

ADV202 JPEG2000 Video Processor User's Guide

Revision 3.4, October 04, 2006

Analog Devices, Inc.
One Technology Way
Norwood, Mass. 02062-9106

Copyright Information

© 2006 Analog Devices, Inc., ALL RIGHTS RESERVED. This document may not be reproduced in any form without prior, express written consent from Analog Devices, Inc.

Printed in the USA.

Disclaimer

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

Trademark and Service Mark Notice

The Analog Devices logo, Blackfin, and the Blackfin logo are registered trademarks of Analog Devices, Inc.

CrossCore, EZ-KIT Lite, and VisualDSP++ are trademarks of Analog Devices, Inc.

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

CONTENTS

ADV202 INTRODUCTION

ADV202 Introduction	1-1
Features	1-2
Typical Applications	1-4
Additional Supporting Documentation	1-5
ADV202 Core Architecture	1-6
Functional Description	1-6
Wavelet Engine	1-6
Entropy Codecs	1-7
Embedded Processor System	1-7
Memory System	1-8
Internal DMA Engine	1-8
Configurable FIFO Block	1-8

ADV202 PHYSICAL INTERFACE

ADV202 Video and Host Interface	2-1
Internal Embedded Processor	2-2

Video Interface (VDATA bus)	2-2
EAV/SAV mode	2-3
HVF mode	2-3
Raw Video mode	2-3
HOST Interface (HDATA bus)	2-4
Pixel Input on HOST Interface	2-4
HOST Bus Configuration	2-5
Pin Configuration and Bus Modes	2-6
32 Bit Host/32 bit Data	2-6
16 Bit Host/32 bit Data	2-6
16 Bit Host/16 bit Data	2-6
16 Bit Host/8 bit Data on JDATA bus mode	2-6
Registers	2-8
Direct Registers	2-8
Indirect Registers	2-8
External DMA Interface	2-9
Video Input Formats	2-9

VIDEO APPLICATIONS

Video Applications	3-1
Encode - multichip application	3-1
Decode - multichip master/slave application	3-4
Digital Still Camera and Camcorder application	3-6
Encode/Decode SDTV Video Application	3-7
32 bit Host/32 bit ASIC application	3-8

HIPI (Host Interface Pixel Interface) Application	3-9
JDATA Application	3-10

SYSTEM REGISTERS

Direct Registers	4-4
Direct Register Memory Map	4-4
Direct Register Descriptions	4-6
Internal Hardware Registers	4-23
Indirect Registers: Configuration and Status	4-26
Indirect Registers: Video Timing and Dimension	4-31
Indirect Registers: External DMA	4-35
EDMOD Register Descriptions	4-38
External DMA Width	4-38
DMA Mode - DREQ/DACK DMA Mode and FLY-BY DMA Mode	
4-39	
DMA Mode - Dedicated Chip Select DMA Mode	4-39
JDATA - DATA Mode	4-39
Single Transfer - DREQ/DACK DMA	4-40
Burst Transfer - DREQ/DACK DMA	4-40
Burst Length	4-41
Indirect Registers: FIFO Block	4-42
Host 32 and 16 bit accesses to Direct/Indirect Registers and Indirect	
Memory	4-45
Accesses Using a 32 bit Host	4-46
Accesses Using a 16 bit Host	4-48

Video Modes	4-53
Encode Mode	4-54
Decode Slave Mode	4-54
Decode Master	4-54
Video Input Formats	4-55
Pin configuration for Video Input Formats on HDATA bus ..	4-55
Pin configuration for Video Input Formats on VDATA bus ..	4-55
8 bit Video Interface	4-63
10 bit Video Interface	4-66
12 bit Video Interface	4-69

1 ADV202 INTRODUCTION

The ADV202 video processor is a single-chip JPEG2000 CODEC targeted at video and high bandwidth image compression applications. Benefits include enhanced quality and feature sets provided by the JPEG2000 (J2K) - ISO/IEC15444-1 image compression standard. The ADV202 implements the computationally intensive operations of the JPEG2000 image compression standard as well as providing fully compliant code stream generation for most applications.

The ADV202 has a dedicated video port providing glueless connection to common digital video standards such as ITU.R-BT656, SMPTE125M, SMPTE293M [525p], ITU.R-BT1358 [625p], SMPTE274M[1080i] or SMPTE296M [720p]. A variety of other high speed synchronous pixel and video formats can also be supported using the programmable framing and validation signals.

The ADV202 can process images at a rate of 40M samples/sec in reversible mode, and at higher rates when used in irreversible mode. The ADV202 contains a dedicated wavelet transform engine, three entropy codecs, on board memory system and an embedded RISC processor which can provide a complete JPEG2000 compression/decompression solution.

The wavelet processor supports the 9/7 irreversible wavelet transform and the 5/3 wavelet transform in reversible and irreversible modes. The entropy codecs support all features in the JPEG2000 Part 1 specification, except Maxshift ROI.

Features

The ADV202 operates on a rectangular array of pixel samples called a tile. A tile may contain a complete image, up to the maximum supported size, or some portion of an image. The maximum horizontal tile size supported depends on the wavelet transform selected and the number of samples in the tile. Images larger than the ADV202's maximum tile size may be broken into individual tiles and then sent sequentially to the chip while still maintaining a single, fully compliant, JPEG2000 code stream for the entire image.

The ADV202 supports a broad set of features that are included in Part 1 of the JPEG2000 standard (ISO/IEC 15444). Depending on the particular application requirements, the ADV202 can provide varying levels of JPEG2000 compression support. It can provide raw code-block and attribute data output which allows the host software to have complete control over the generation of the JPEG2000 code stream and other aspects of the compression process such as bit-rate control. Otherwise the ADV202 can create a complete, fully compliant, JPEG2000 code stream in .j2c or jp2 format.

Features

The ADV202 has the following features:

- Complete single-chip JPEG2000 compression/decompression solution for video and still images.
- Patented SURF™ (Spatial Ultra-efficient Recursive Filtering) technology enables low power and low cost wavelet based compression.
- Supports both 9/7 and 5/3 wavelet transforms with up to 6 levels of transform.

- Programmable tile/image size with widths up to 2048 pixels in three-component 4:2:2 interleaved mode, and up to 4096 pixels in single-component mode. Maximum tile/image height is 4096 pixels.
- Video interface directly supporting ITU.R-BT656, SMPTE125M PAL/ NTSC, SMPTE274M, SMPTE293M [525p], ITU.R-BT1358[625p] or any video format with a max. input rate of 65 Msamples/sec for irreversible mode or 40Msamples/sec for reversible mode. Two or more ADV202s can be combined to support full frame SMPTE274M HDTV [1080i] or SMPTE296M [720p].
- Compression on a field-by-field basis or combined-two-fields basis for SD video sources for improved performance and compression efficiency.
- Flexible asynchronous SRAM style host interface allows glue-less connection to most 16/32-bit microcontrollers and ASICs.
- 2.5-3.3v I/O and 1.5v core supply.
- 12mm x 12mm 121-ball fpBGA, speed grade 115 MHZ or 13mm x 13mm 144-ball fpBGA, speed grades 135 MHz and 150MHz.

Typical Applications

The type of applications the ADV202 is used for includes.

- Networked video and image distribution systems
- Wireless video and image distribution Image archival/retrieval for Standard Definition and HDTV formats
- Digital CCTV and surveillance systems
- Digital Cinema Systems
- Professional/HDTV Video editing and recording
- Digital Camcorders and Digital Still Cameras.

Additional Supporting Documentation

The following supporting documentation can be downloaded from the www.analog.com ADV202 product page under Technical Documentation:

- AN-799 Testmodes - recommended test procedures to verify correct hardware configuration.
- *Getting Started with the ADV202* - a programming guide containing configuration examples for all modes mentioned in this User Guide.
- ADV202 Quick Reference sheet.
- HIPI Mode and Still Image Applications - contain configuration examples for HIPI mode [host interface only configuration].
- ADV202 datasheet containing package specifications and all electrical and timing specifications.
- Firmware for the ADV202. The firmware can be downloaded for free. It is recommended to always use the latest version.
- Product Change Notices and Errata sheets.

ADV202 Core Architecture

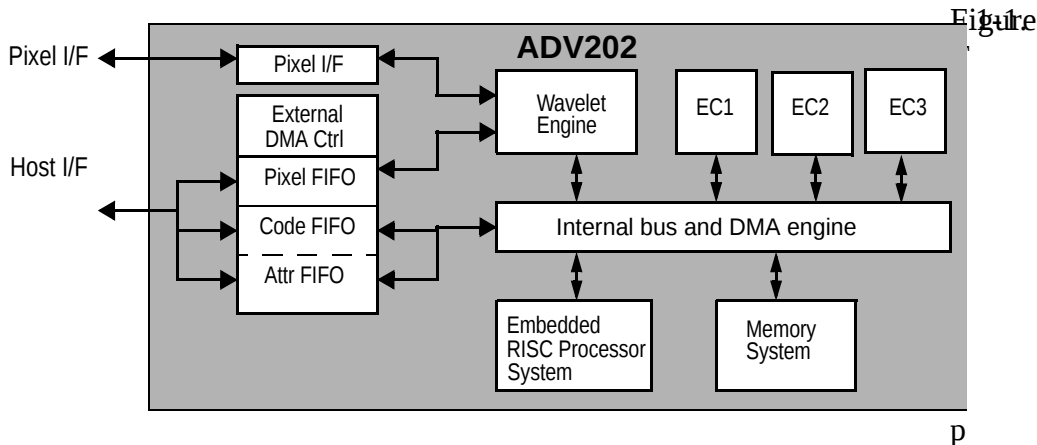


Figure 1-1 Processor Block Diagram

Functional Description

The input video or pixel data is passed on to the ADV202's pixel interface where samples are de-interleaved and passed on to the Wavelet Engine where each tile or frame is decomposed into sub-bands using the 5/3 or 9/7 filters. The resultant wavelet coefficients are then written to internal memory. The entropy coders then code the wavelet data so that it conforms to the JPEG2000 standard. An internal DMA provides high bandwidth memory to memory transfers as well as high performance transfers between functional blocks and memory.

Wavelet Engine

The ADV202 provides a dedicated wavelet transform processor based on Analog Devices' proven and patented SURF^R technology. This processor can perform up to 6 wavelet decomposition-levels on a tile. In

Encode mode, the wavelet transform processor will take in uncompressed samples, perform the wavelet transform and write the wavelet coefficients in all frequency sub-bands to internal memory. Each of these sub-bands is then further broken down into code-blocks. The code-block dimensions can be user-defined, and are used by the wavelet transform processor to organize the wavelet coefficients into code-blocks when writing to internal memory. Each completed code-block is then entropy coded by one of the Entropy Codecs.

In Decode mode, wavelet coefficients are read from internal memory and are recomposed into uncompressed samples.

Entropy Codecs

The entropy codec block performs context modeling and arithmetic coding on a code block of the wavelet coefficients. Additionally, this block also performs the distortion metric calculations during compression that are required for optimal rate-distortion performance. Since the entropy coding process is the most computationally intensive operation in the JPEG2000 compression process, three dedicated hardware entropy codecs are provided on the ADV202.

Embedded Processor System

The ADV202 incorporates an embedded 32-bit RISC processor. This processor is used for configuration, control and management of the dedicated hardware functions as well as for code-stream parsing/generation.

Memory System

The memory system's main functions are to manage wavelet coefficient data and interim code-block/attribute data to provide a temporary work space for creation/parsing/storage of JPEG2000 codestream. The memory system can also be used for program and data memory for the embedded processor

Internal DMA Engine

The internal DMA engine provides high bandwidth memory to memory transfers as well as high performance transfers between memory and functional blocks. This function is critical for high speed code stream generation and parsing.

Configurable FIFO Block

FIFOs are provided for pixel data, code-stream data, and attribute data. The FIFOs can be accessed directly from the host interface using normal addressed read/write cycles or by external host DMA accesses using a DREQ/DACK protocol or by one of the dedicated hardware handshake protocols. Each FIFO also has a programmable threshold that can be used to generate an interrupt.

2 ADV202 PHYSICAL INTERFACE

This chapter describes ADV202 video and host interface.

ADV202 Video and Host Interface

There are several possible modes to interface to the ADV202. The designer can use the VDATA bus and the HDATA bus or the HDATA bus alone.

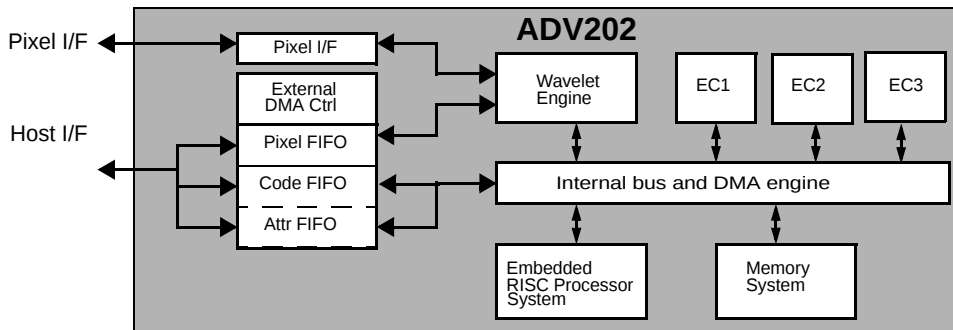


Figure 2-1. ADV202 Internal Block Diagram

Internal Embedded Processor

The ADV202 is a System-on-Chip (SOC) implementation, which means that it incorporates dedicated hardware functions, a processor and on-board firmware/software. It also means that configuring and managing the operations of the chip can be handled by the on-chip processor and its software. Other than basic bus and I/O configuration which the user has to set up, most of the configuration and control of the ADV202 is handled by the firmware. The firmware is responsible for setting other registers in different functional blocks of the ADV202, scheduling events etc. The user has no access to these functional blocks, they are entirely controlled by the firmware.

- ADV202 configuration and firmware load procedures are described in detail in the *Getting Started with the ADV202* programming guide.

Video Interface (VDATA bus)

The video interface can be used in applications where uncompressed pixel data is on a separate bus from compressed data. For example, when an ADV202 is in Encode mode a typical application is to have 8 or 10 bit digital input video on the VDATA bus. Often this comes from a video decoder (i.e. ADV7189B) that converts the analog video into a digital format. When used with the VDATA interface the HDATA bus is used to output compressed data from the ADV202 Encoder.

When the ADV202 is in Decode mode, a typical application is having the compressed JPEG2000 video input on the HDATA bus by a Host processor and the VDATA bus drives a video encoder. This video encoder (i.e. ADV7301A) converts the digital video back into an analog output.

The video interface can support video data or still image data input/output in 8, 10, 12-bit monochrome or YCbCr format. The data format is left justified (msb) on the VDATA bus. If YCbCr data is used, it must be in 4:2:2 format.

Video data can be input/output in several different modes on the VDATA bus. For these modes the pixel clock must be input on the VCLK pin.

Optionally the ADV202 can be used in a specific mode [de-interlaced mode] when used in NTSC or PAL, where two consecutive fields are processed at once. This mode yields significantly better compression performance and efficiency for Standard Definition NTSC or PAL sources. For this mode, Jclk must be 4xVclk.

Additionally, high definition digital video, such as SMPTE-274M (1080i) is supported using two or more ADV202 devices.

EAV/SAV mode

The ADV202 can accept video with embedded EAV/SAV codes where the YCbCr data is interleaved onto a single bus. This is setup by the user in the firmware parameters.

HVF mode

The ADV202 can also accept video data accompanied with separate H, V, F signals where video data is interleaved onto a single bus. This is setup by the user in the firmware parameters.

Raw Video mode

This mode is used for still picture data and non-standard video. VFRM, VSTRB, VRDY are used to program the dimension of the image.

HOST Interface (HDATA bus)

The ADV202 can connect directly to a wide variety of 16 and 32 bit host processors and ASICs using an asynchronous SRAM-style interface, DMA accesses or JDATA mode interface. The ADV202 supports 16 and 32-bit buses for control and 8/16/32-bit buses for data transfers. The control and data channel bus widths can be specified independently which allows the ADV202 to support applications that require control and data buses of different width. The host interface is used for configuration, control and status functions as well as for transferring compressed data streams. It can be used for uncompressed data transfers in certain modes. The host interface may be shared for control and status communications, uncompressed data input/compressed data output or compressed data input/uncompressed data output. The data streams may include:

1. uncompressed tile data [for example: still image data]
2. fully encoded JPEG2000 code stream (or unpackaged code blocks)
3. code-block attributes

The ADV202 uses big endian word alignment for 16 and 32-bit transfers. All data is left justified. Meaning, in a 32 bit system, bit 31 is the most significant bit.

Pixel Input on HOST Interface

Pixel input on the host interface supports 8/10/12/14 or 16-bit raw pixel data formats. It can be used for pixel [still image] input/compressed data output in encode mode or compressed data input/ pixel data output in decode mode. Since no timing codes or sync signals are associated with the input data on the host interface, dimension registers and internal counters are used and must be programmed to indicate start and end of frame.

Refer to the TechNote “*ADV202 in HIPI mode*” for detail on how to use the ADV202 in this mode.

HOST Bus Configuration

To provide maximum flexibility, the Host interface provides several configurations to meet specific system requirements. The default bus mode uses the same HDATA pins to read and write control/status and/or data to and from the ADV202.

The ADV202 can support 16 or 32-bit control transfers and 8/16/32-bit data transfers. The size of the control/status and data buses can be selected independently by writing to the direct and indirect registers that configure bus size. This allows a 16-bit micro controller to configure and control the ADV202 while still providing 32-bit data transfers to an ASIC or external memory system.

Pin Configuration and Bus Modes

The ADV202 provides a wide variety of control and data configurations which allows it to be used in many applications with little or no glue logic. The modes described below are configured using the BUSMODE register.

32 Bit Host/32 bit Data

In this mode the HDATA[31..0] pins provide full 32-bit wide data access to Pixel, Code, Attr FIFOs.

16 Bit Host/32 bit Data

This mode allows a 16-bit host to configure and communicate with the ADV202 while still allowing 32-bit accesses to the Pixel, Code, Attr FIFOs using the external DMA capability. All addressed host accesses are 16-bits and therefore only use the HDATA[15..0] pins. The HDATA[31..16] pins are used to provide the additional 16-bits necessary to support the 32-bit external DMA transfers to/from the FIFOs for data.

16 Bit Host/16 bit Data

This mode uses 16-bit transfers if used for host or external DMA data transfers. This mode allows for use of the extended pixel interface modes.

16 Bit Host/8 bit Data on JDATA bus mode

This mode provides separate data input/output and host control interface pins. Host control accesses are 16 bits and use HDATA[15..0] while the dedicated data bus uses JDATA[7..0]. JDATA uses a valid/hold synchronous transfer protocol based on MCLK. The direction of the JDATA bus is determined by the mode of the ADV202. If the ADV202 is encoding (compression) then JDATA[7..0] is an output. If the ADV202 is decoding

(decompression) then JDATA[7..0] is an input. Host control accesses remain asynchronous. Host accesses are possible during JDATA data transfers.

Registers

Direct and indirect registers and internal memory are used to program the ADV202 as well as receive or send compressed video data.

Direct Registers

Direct registers 0x05-0x0A, 0x0D, 0x0E, 0x0F are 16-bit registers and are the ones most commonly used by the external host/controller and are therefore accessible directly. In order to minimize pin count and cost, the number of address pins has been limited to four, which yields a total direct register address space of 16 locations. All indirect registers and indirect memory can be accessed through the use of the IADDR (Indirect Address, 0x0B) and IDATA (Indirect Data, 0x0C) registers.

With the exception of the IADDR, IDATA and the FIFO registers, all direct control/status registers in the ADV202 are 16 bits wide and are addressed with 16-bit ‘words’. When 32-bit host mode is enabled, the upper 16 bits of the HDATA bus are ignored on writes and will return all zeroes on reads of 16-bit registers. The user should mask these bits in software when doing read and write verifys.

Indirect Registers

Since the ADV202 contains both 16 and 32-bit registers and its internal memory is mapped as 32-bit data, a mechanism has been provided to allow 16-bit hosts to access these registers and memory locations. This is accomplished with the staging register (STAGE). The STAGE registers typically is never used when using a 32 bit Host processor.

The STAGE register is a direct register used by the host interface to set the indirect address register (IADDR) as well as for writing and reading data to that indirect memory location. Prior to writing to the desired location, the STAGE register must be written with the most significant 16-bit word of the address for that memory location. This is followed by writing the least significant 16-bit word of the address to the IADDR

register. These two 16-bit words are combined to create the full 32-bit address. Similarly, to write a 32-bit data value, STAGE is first written with the most significant 16-bit word, followed by writing the least significant 16-bit word to the IDATA register.

When a register is read, the upper [most significant] 16-bit word is returned immediately on HDATA and the lower 16-bit word can be retrieved by reading the STAGE register on a subsequent access. Programming examples can be found in the *Getting Started with the ADV202* programming guide.



This does not apply to the PIXEL, CODE and ATTR FIFOs. These channels are always accessed at the specified data width and do not require the use of the STAGE register.

External DMA Interface

The External DMA Interface is provided to enable high-bandwidth data I/O between an external DMA controller and the ADV202 data FIFOs. There are two independent DMA channels which can each be assigned to any one of the data stream FIFOs [Pixel/Code/Attribute].

The controller supports asynchronous DMA using a DREQ/DACK (Data-Request/Data-Acknowledge) protocol in either single or burst access modes. Additional functionality is provided by Fly-By mode and Dedicated Chip Select (DCS) modes.

Video Input Formats

The ADV202 supports a wide variety of formats for uncompressed video and still image data [refer to 4-58 Input formats]. The actual interface and bus modes selected for transferring uncompressed data determines the allowed size of the input data and the number of samples transferred with each access. The host interface can support 8, 10, 12, 14, 16-bit data formats on the HDATA bus.

ADV202 Video and Host Interface

The video interface can support video data or still image data input/output. Supported formats are 8, 10 or 12-bit tYCbCr or single component data. All possible formats are listed in section 4 of this user guide. All formats can support less precision than provided by specifying the actual data width/precision in the PMODE register.

The maximum allowable data input rate is limited by using irreversible or reversible compression modes and the data width (or precision) of the input samples.

The ADV202 datasheet, table 23 and table 24 does list the maximum allowable data input rates.

3 VIDEO APPLICATIONS

This section describes typical video applications using the ADV202 JPEG2000 Video Processor.

Video Applications

A single ADV202 can be in either Encode or Decode mode depending on what firmware the user has loaded into the device. An ADV202 can not change from Encode mode to Decode mode unless new firmware is loaded into the chip and the chip has gone through a reset and configuration process again.

Encode - multichip application

Due to the data input rate limitation an 1080i application will require at least 2 ADV202s to encode or decode full resolution 1080i or 720p video. In encode mode the ADV202 accepts Y and CbCr data on separate buses. An Encode example is shown in [Figure 3-1 on page 3-2](#).

In Decode mode, a master/slave configuration (as shown in the [Figure 3-2 on page 3-4](#)) or a slave/slave configuration can be used in order to synchronize the outputs of the two ADV202s. Refer to the application note AN-796 “Using the *ADV202 in a multi-chip application*” for more detail on how to configure the ADV202s in a multi-chip application.

Video Applications

Applications that have the two separate VDATA outputs sent to an FPGA or buffer before they are sent to an encoder do not require synchronization at the ADV202 outputs

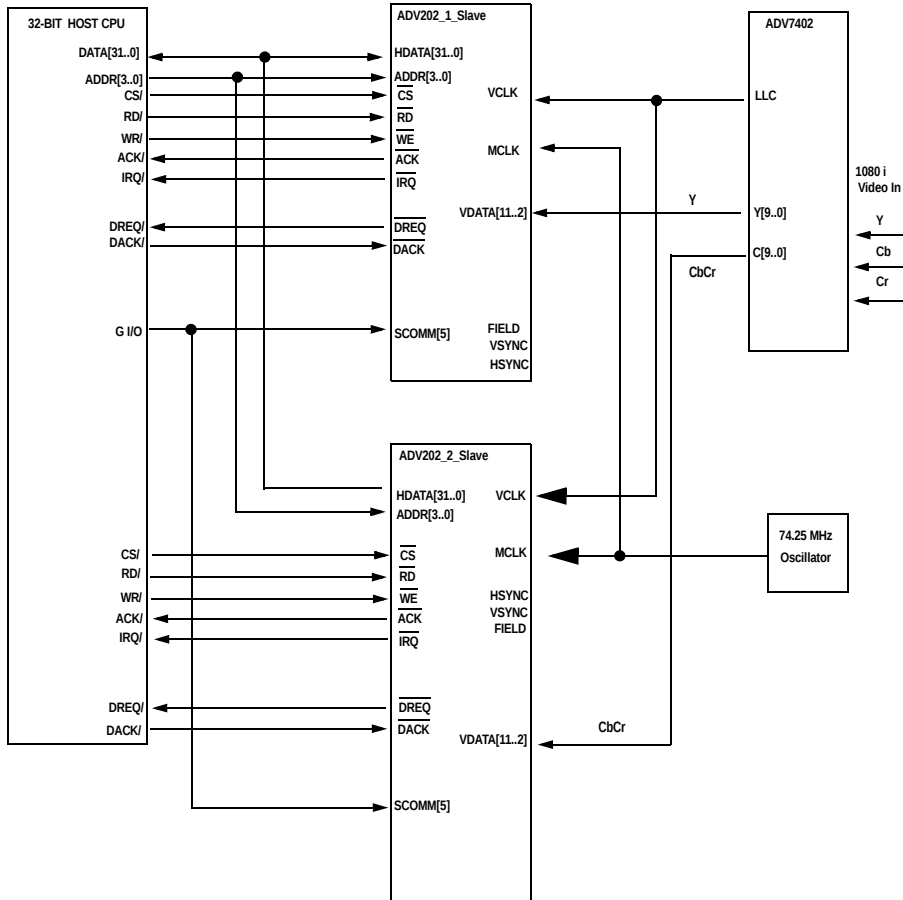


Figure 3-1. Encode - Multi-chip application

Decode - multichip master/slave application

In a master/slave configuration it is expected that the master HVF outputs are connected to the slave HVF inputs and that each SCOMM[5] pin is connected to the same GPIO on the host.

In a slave/slave configuration the common HVF for both ADV202s is generated by an external host sync and each SCOMM[5] is connected to the same GPIO output on the host.

SWIRQ1, Software Interrupt 1 in the EIRQIE register must be unmasked on both devices to enable multi-chip mode in order to synchronize video data at the VDATA outputs.

Video Applications

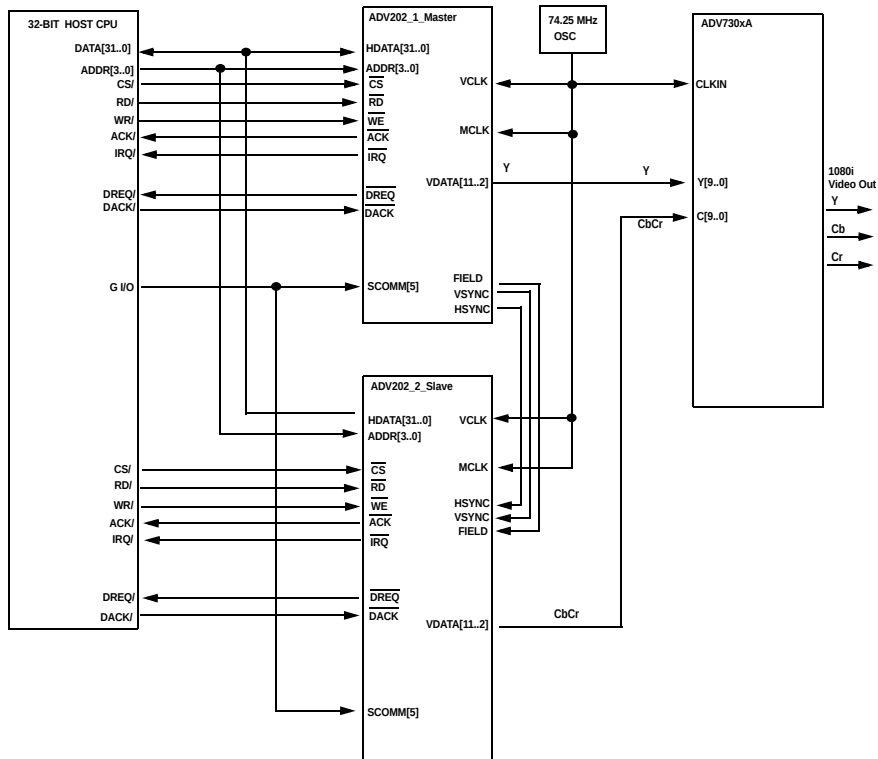


Figure 3-2. Decode - Master/Slave - Multi-chip application

Digital Still Camera and Camcorder application

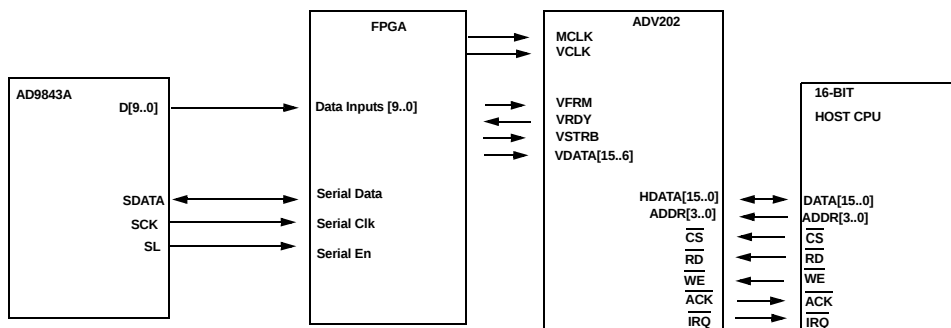


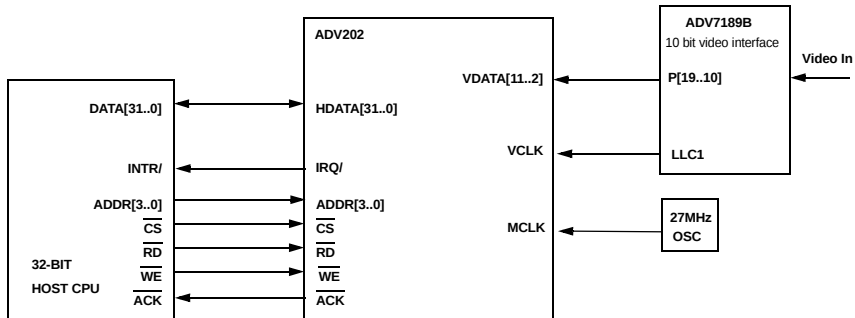
Figure 3-3. Digital Camera and Camcorder Application

For still image applications that do require to encode images that do not contain any blanking or timing information, ie only contain raw image data, the Raw Video Pixel mode can be used.

The *Getting Started with the ADV202* programming guide should be consulted for Raw Video Pixel configuration and programming sequence when using this mode.

Encode/Decode SDTV Video Application

Encode mode



Decode mode

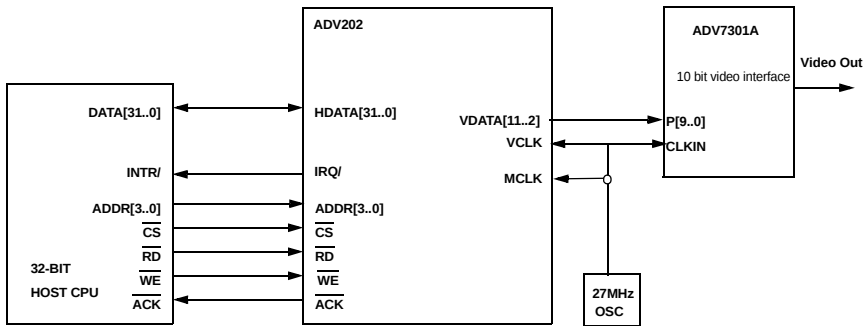


Figure 3-4. Encode/Decode - SDTV - 10 bit CCIR656 - Normal Host Mode Application

32 bit Host/32 bit ASIC application

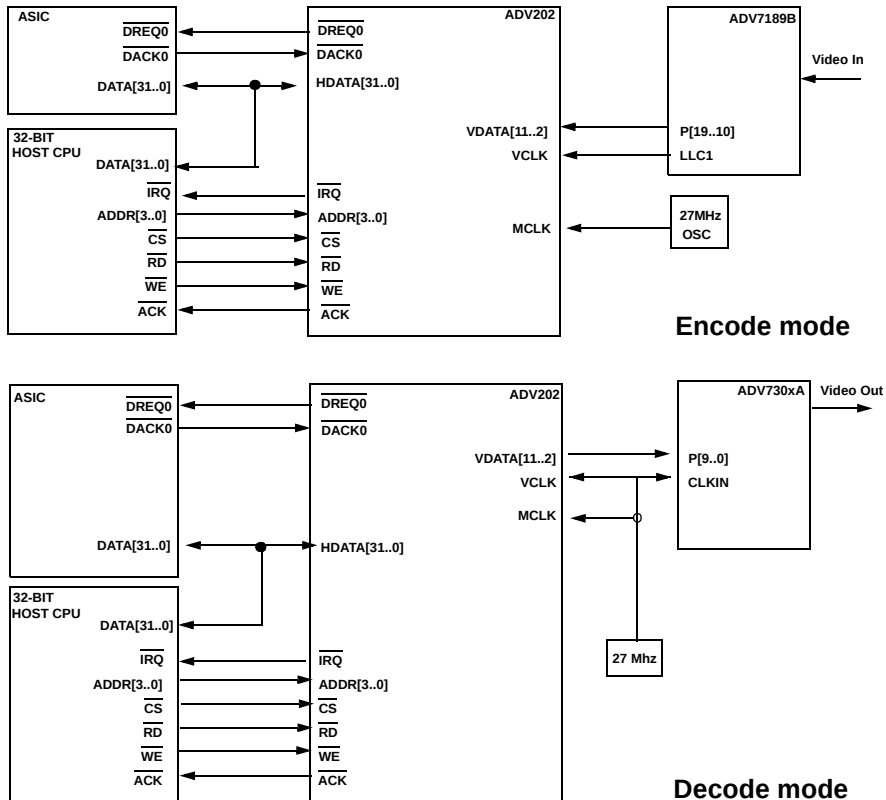


Figure 3-5. 32bit HOST/32 bit ASIC - 10 bit CCIR656 using DMA DREQ/DACK mode for compressed data transfer

The figure above shows a SDTV application using the external DMA channels of the ADV202 [EDMOD registers] to transfer compressed data to/from the ADV202. The *Getting Started with the ADV202* programming guide should be consulted for configuration and programming sequence when using this configuration.

HIPI (Host Interface Pixel Interface) Application

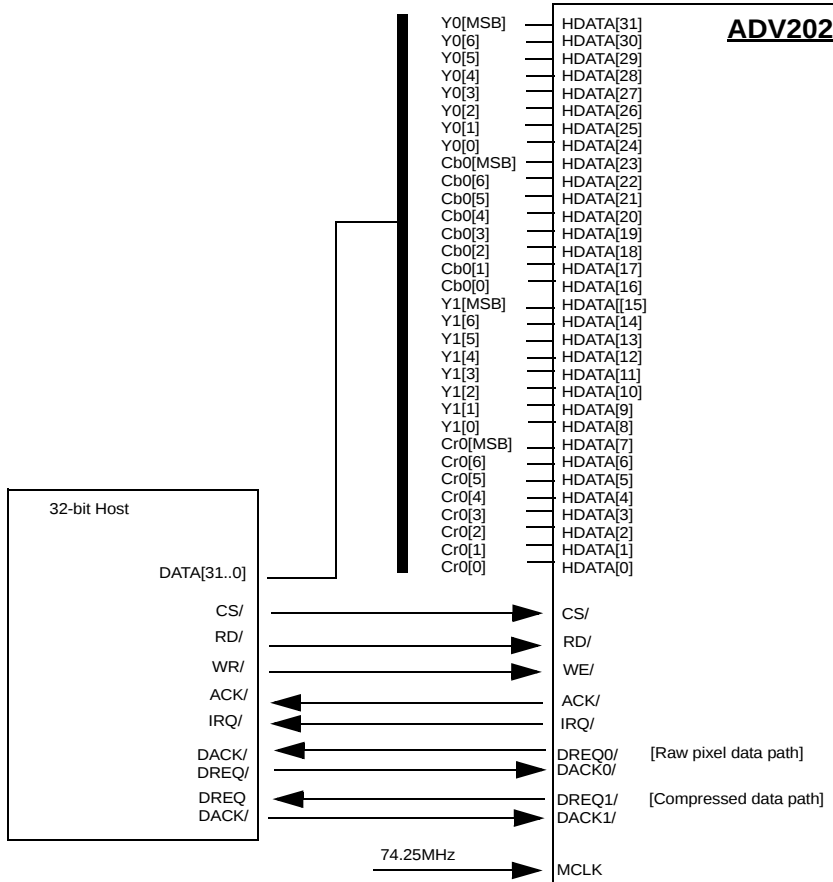


Figure 3-6. Encode - HIPI application

For more information about how to use the ADV202 in HIPI mode, the technical notes *ADV202 HIPI mode* and *HIPI mode overview* should be consulted.

JDATA Application

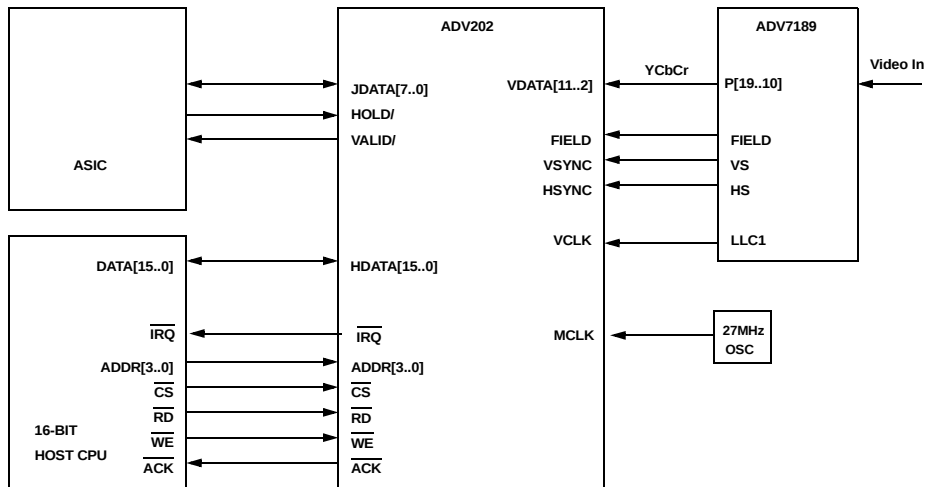


Figure 3-7. JDATA - 16 bit Host - Dedicated JDATA Output - 10 bit CCIR656 application

For JDATA configuration and programming sequence the *Getting Started with the ADV202* programming guide should be used.

4 SYSTEM REGISTERS

This section provides a functional description of the direct and indirect registers of the ADV202.

- There are 16 direct registers with a *direct* address range from 0x0 to 0xF. The registers are accessed using the ADDR [3..0] pins for address and the HDATA bus for data. These registers are 16 or 32 bits wide and aligned on word (4-byte) boundaries. See [Table 4-1, “Direct Register Memory Map,” on page 4-4](#)
- The Indirect register address range is from *indirect* addresses 0xFFFF0400 to 0xFFFF14FC. These registers are 16 bits wide and aligned on word (4-byte) boundaries. See [Table 4-5, “Internal Hardware Registers,” on page 4-24](#). Indirect address and data registers can be accessed by properly setting the IADDR and IDATA direct registers.
- The indirect memory locations are used for the storage of the firmware and firmware parameters. The 32KB encode or decode firmware is stored at starting address 0x00050000. The firmware parameters are stored at starting address 0x00057F00. These memory locations are 32 bits wide and aligned on 4-byte bound-

aries. Indirect memory address and data registers can be accessed by properly setting the IADDR and IDATA direct registers.

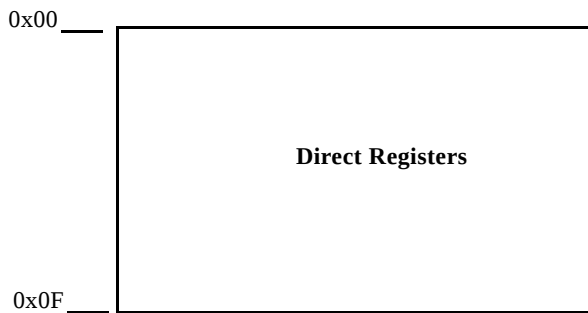


Figure 4-1. Direct Register Memory Map

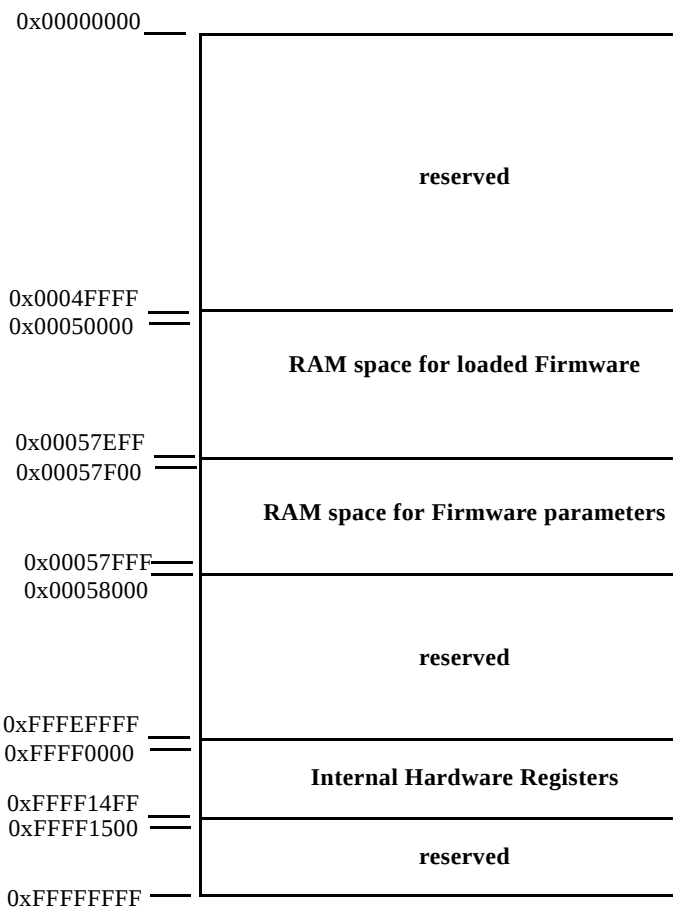


Figure 4-2. Indirect Registers and Firmware memory map

Direct Registers

The ADV202 has 16 Direct Registers as shown in [Table 4-1](#). The Direct Registers are accessed by the host interface using ADDR [3..0], HDATA[31..0], CS/, RD/, WE/, ACK/ pins.



The host must first initialize the Direct Registers before any application specific operation can be implemented. The PLL_HI and PLL_LO registers are the only registers that can be read/write verified on bootup.

Direct Register Memory Map

Table 4-1. Direct Register Memory Map

ADDRESS	NAME	DESCRIPTION	R/W	LENGTH
0x00	PIXEL	Pixel FIFO Access Register	R/W	[31..0]
0x01	CODE	Compressed Code Stream Access Register	R/W	[31..0]
0x02	ATTR	Attribute FIFO Access Register	R/W	[31..0]
0x03	Reserved	Reserved	R/W	[31..0]
0x04	Reserved	Reserved	R/W	[31..0]
0x05	EIRQIE	External Interrupt Enabled	R/W	[15..0]
0x06	EIRQFLG	External interrupt Flags	R/W	[15..0]
0x07	SWFLAG	Software Flag Register	R	[15..0]
0x08	BMODE	Bus Mode Configuration Register	R/W	[15..0]
0x09	MMODE	Miscellaneous Mode Register	R/W	[15..0]
0x0A	STAGE	Staging Register	R/W	[15..0]
0x0B	IADDR	Indirect Address Register	R/W	[31..0]
0x0C	IDATA	Indirect Data Register	R/W	[31..0]
0x0D	BOOT	Boot Mode Register	R/W	[15..0]

Table 4-1. Direct Register Memory Map

ADDRESS	NAME	DESCRIPTION	R/W	LENGTH
0x0E	PLL_HI	PLL Control Register - High Byte	R/W	[15..0]
0x0F	PLL_LO	PLL Control Register - Low Byte	R/W	[15..0]

Direct Register Descriptions

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0x0	PIXEL	Pixel FIFO Access Register	R/W	undefined

The 32 bit PIXEL register is used to access the Pixel data FIFO via normal addressed accesses and generally is used for pixel data input and output on the host interface. The actual bus width when accessing this register depends on the host control data width selected [BUSMODE/HWIDTH].

- 16-bit mode will accept/return data on HDATA[15..0]
- 32-bit mode will accept/return data on HDATA[31..0]

In 16 bit mode, the upper unused bytes will be ignored on writes and return zeros on reads. The depth of the Pixel Data FIFO is fixed to 256 x 32-bits.



A read operation when the PIXEL FIFO is empty or a write when it is full will cause the PFERR bit in the EIRQFLG direct register to be set.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0x1	CODE	Compressed Code Stream Access	R/W	undefined

The 32 bit CODE register is used to access the JPEG2000 compressed code stream FIFO via normal addressed accesses. It is also used for accessing raw code blocks when the ADV202 is in HIPI mode [Host Interface Pixel Interface mode]. The actual bus width when accessing this register depends on the host control data width selected [BUSMODE/HWIDTH].

In Encode mode, a host processor can read the compressed JPEG2000 stream using the CODE register in 16- or 32-bit mode. In Decode mode, it can write this compressed stream to the CODE register.

- 16-bit mode will accept/return data on HDATA[15..0]
- 32-bit mode will accept/return data on HDATA[31..0]

In 16 bit mode, the upper unused bytes will be ignored on writes and return zeros on reads.



A read operation when the CODE FIFO is empty or a write when it is full, will cause the DFERR bit in the EIRQFLG direct register to be set.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0x2	ATTR	Attribute FIFO Access	R/W	undefined

The 32 bit ATTR register is used to access the attribute FIFO via normal addressed accesses. The actual bus width when accessing this register depends on the host control data width selected [BUSMODE/HWIDTH].

- 16-bit mode will accept/return data on HDATA[15..0]
- 32-bit mode will accept/return data on HDATA[31..0]

Direct Registers

In 16 bit mode, the upper unused bytes will be ignored.



A read operation when the ATTR FIFO is empty or a write when it is full, will cause the AFERR bit in the EIRQFLG direct register to be set.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0x5	EIRQIE	External Interrupt Enables	R/W	see bit field

This 16 bit EIRQIE register is used to enable conditions that will cause an external interrupt to occur on the IRQ/ pin. Refer to the “*Getting Started with the ADV202*” programming guide.

EIRQIE Bit Field			
Bit	Name	Description	Reset Value
0	PFTH	Pixel FIFO threshold condition exists (Level sensitive)	0
1	DFTH	Code FIFO threshold condition exists (Level sensitive)	0
2	AFTH	Attribute FIFO threshold condition exists (Level sensitive)	0
3	Reserved	Reserved	0
4	PFERR	Pixel FIFO has overflowed or underflowed	0
5	DFERR	Data FIFO has overflowed or underflowed.	0
6	AFERR	Attribute FIFO has overflowed or underflowed.	0
7	Reserved	Reserved	0
8	Reserved	Always write 0	0
9	Reserved	Always write 0	0
10	SWIRQ0	Software interrupt 0	0
11	SWIRQ1	Software interrupt 1	0
12	SWIRQ2	Software interrupt 2 (<i>reserved for future use</i>)	0
13	Reserved	Reserved	0
14	Reserved	Reserved	0
15	Reserved	Reserved	0

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0x6	EIRQFLG	External Interrupt Flags	R/W	see bit field

Direct Registers

The 16 bit EIRQFLG register indicates which external interrupt conditions are currently active. The bits in this register correspond directly to the bits in the external interrupt enable register (EIRQIE).



Individual interrupts are cleared by writing a ‘1’ in the proper bit position of this register. A common mistake is trying to clear the interrupt flag before acting on what caused the interrupt. An example would be using the HOST processor to read JPEG2000 data out of the Code FIFO after the DFTH interrupt is set [CODE FIFO threshold condition is set]. First the data must be read out of the FIFO, then the interrupt must be cleared by writing to the according bit in the EIRQFLG register. Otherwise, if the interrupt flag is cleared before data is read out, the interrupt never gets cleared because the data count in the FIFO is still above the FIFO threshold.

EIRQFLG Bit Field			
<i>Bit</i>	<i>Name</i>	<i>Description</i>	<i>Reset Value</i>
0	PFTH	Pixel FIFO threshold condition exists (Level sensitive)	1
1	DFTH	Data FIFO threshold condition exists (Level sensitive)	1
2	AFTH	Attribute FIFO threshold condition exists (Level sensitive)	1
3	Reserved	Reserved	1
4	PFERR	Pixel FIFO has overflowed or underflowed	0
5	DFERR	Data FIFO has overflowed or underflowed.	0
6	AFERR	Attribute FIFO has overflowed or underflowed.	0
7	Reserved	Reserved	0
8	Reserved	Reserved	0
9	Reserved	Reserved	0
10	SWIRQ0	Software interrupt 0	0
11	SWIRQ1	Software interrupt 1	0
12	SWIRQ2	Software interrupt 2	0
13	Reserved	Reserved	0
14	<i>Reserved</i>		0
15	<i>Reserved</i>		0

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0x7	SWFLAG	<i>Application ID</i>	R	see bit field

The 16 bit SWFLAG register is used to read the Application ID after the ADV202 Encode or Decode firmware has been uploaded by the Host processor. After the firmware has been uploaded and a soft reboot has been initiated, the embedded firmware takes control of this register and will write the Application ID into this register. Application IDs are different for Encode and Decode. This register is often used by the 16 or 32 bit Host processor to ensure the firmware upload was successful.

Direct Registers

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0x8	BUSMODE	Bus Mode Configuration	R/W	see bit field

The 16 bit BUSMODE register configures host control and data bus.

BUS MODE Bit Field			
Bit	Name	Description	Reset Value
1..0	HWIDTH	Host control data width 00 = Invalid 01 = 16-bits 10 = 32-bits 11 = Invalid	'01'
3..2	DWIDTH	DMA data width 00 = 8-bits 01 = 16-bits 10 = 32-bits 11 = Invalid	'01'
6..4	BCFG	Bus configuration 000 = Normal. HDATA [31..0] are available for host or data transfers according to the settings of HWIDTH and DWIDTH. For normal read/write access HWIDTH and DWIDTH are set to 1 or 2. 001 = JDATA Mode. Enables JDATA[7..0]. HWIDTH must be set to 1 and DWIDTH must be set to 0. 010 = Raw Video Mode. HWIDTH should be set to 16-bit and DWIDTH should be set to 16- or 8-bit mode. 011-111= Reserved	'000'
7	Reserved	Reserved for future use; always write 0.	0
15..8	N/A	Reserved for future use; always write 0.	undef

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0x9	MMODE	Misc. Mode Configuration	R/W	see bit field

This 16 bit MMODE register configures miscellaneous functions for indirect access for 32 bit or 16 bit Host interfaces.

Proper use and examples of this register can be found in section: [“Host 32 and 16 bit accesses to Direct/Indirect Registers and Indirect Memory”](#) on page 4-45.

Direct Registers

MMODE Bit Field			
Bit	Name	Description	Reset Value
1..0	IWIDTH	Indirect access widths 00 = Invalid 01 = 16-bits 10 = 32-bits 11 = Invalid	'01'
3..2	IAUTOSTP	IADDR increment/decrement size. IADDR can automatically increment/decrement the IADDR location. This would require only one write to the IADDR register/memory. It can also be disabled., in which case each indirect register/memory access requires a write to IADDR and IDATA. 00 = 8-bit increment/decrement 01 = 16-bit increment/decrement 10 = 32-bit increment/decrement 11 = Disable auto increment/decrement	'01'
4	IAUTOMOD	Indirect address auto increment/decrement 0 = increment 1 = decrement	0
5	CTLREGAM	Control register address mode. Enable full indirect address map. <i>The control register address space is 0xFFFF0000 and above.</i> 0 = Full address mode. Provides full access to the ADV202's internal address space 1 = Control register mode. The upper 16-bits of the internal address are forced to the beginning of the control register address map. This eliminates setting the upper 16-bits of the indirect address register prior to each access.	0
15..6	Reserved	Reserved for future use; always write 0.	undef

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xA	STAGE	Staging	R/W	undefined

The 16 bit STAGE register is used when accessing 32-bit registers in 16-bit host mode. It is not required for a 32 bit host. It is used to access 32 bit words in indirect memory, indirect registers, and 32 bit direct registers. The STAGE register acts as a holding or “staging” register. For example, when using a 16 bit host the STAGE register is used in conjunction with the IADDR register to configure the 32 bit address or the IDATA register.

Proper use and examples of this register can be found in section: [“Host 32 and 16 bit accesses to Direct/Indirect Registers and Indirect Memory”](#) on page 4-45.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xB	IADDR	Indirect Address	R/W	undefined

The 32 bit IADDR register is used to set the address for indirect register and indirect memory read/write accesses.

The indirect address may optionally be auto-incremented/decremented by setting the IAUTOMOD and IAUTOSTP fields in the MMODE register.

Proper use and examples of this register can be found in section: [“Host 32 and 16 bit accesses to Direct/Indirect Registers and Indirect Memory”](#) on page 4-45.

Direct Registers

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xC	IDATA	Indirect Data	R/W	undefined

The 32 bit IDATA direct register is used to read and write data to an indirect register or indirect memory. The data value written to the IDATA register is written to the memory location as specified by the data value in the IADDR register.

The indirect address may optionally be auto-incremented by setting the IAUTOMOD and IAUTOSTP fields in the MMODE register.

Proper use and examples of this register can be found in section: [“Host 32 and 16 bit accesses to Direct/Indirect Registers and Indirect Memory” on page 4-45.](#)

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xD	BOOT	Boot Mode	R/W	see bit field

The 16 bit BOOT register is used to configure the boot mode and initiate a soft or hard reset of the ADV202.



A hard reset, via the reset pin or setting the HARDRST bit causes bits [2..1] to be loaded from the configuration pins as specified below, all other bits in this register will be set as specified in the bit field table

The bits in the BOOT register are used to select various boot modes of the ADV202 after reset and to load specific instruction sets into memory. The boot mode can be configured via hardware [over the CFG pins] or via software [via firmware]. In a hardware configuration after a hard reset, the boot mode will be set to the values on the configuration pins CFG[2..1]. These bits are not reset on a soft reset; see HARDRST

and SOFTRST bit definitions below. The first boot mode after power-up is set by the CFG pins. Only boot mode 2 is available in hardware bootmode.



A soft reset, via setting the SOFTRST bit will only clear the SOFTRST bit, all other bits will remain unchanged. A SOFT RESET affects all ADV202 internal registers except for BOOT/PLL_LO/PLL_HI. A HARD RESET via pin or BOOT register will reset all of the ADV202 internal registers.

BOOT Bit Field			
Bit	Name	Description	Reset Value
2..0	Bootmode(s)	SOFTWARE BOOTMODE 000 Reserved. For internal use only. Do not use. 001 Reserved. For internal use only. Do not use. 010 No-Boot Host mode, ADV202 does not boot but all internal registers and memory are accessible through normal host I/O operations. 011 Reserved. For internal use only. Do not use. 100 Reserved. 101 Co-Processor Boot Used in conjunction with the No-Boot Host mode and starts executing the loaded firmware. 110 Reserved. For internal use only. Do not use 111 Reserved. For internal use only. Do not use HARDWARE BOOTMODE 000 Reserved. For internal use only. Do not use. 001 Not Available 010 No-Boot Host mode, ADV202 does not boot but all internal registers and memory are accessible through normal host I/O operations. 101 Not Available 100 Reserved. 101 Not Available 110 Not Available 111 Not Available	000
3	reserved	Reserved for future use; always write 1.	1
5..4	reserved	Reserved for future use; always write 0.	0

Direct Registers

BOOT Bit Field			
Bit	Name	Description	Reset Value
6	HARDRST	Setting this bit will cause the ADV202 to execute a hard reset. This is identical to asserting a reset on the RESET pin. The value in BOOT MODE will be ignored since it will be immediately reloaded from the configuration pins. This bit will override the SOFTRST bit if both bits are set concurrently. The PLL control register is reset to its reset value as well as the BOOT MODE [2..1] bits representing the values on the CFG [2..1] pins.	0
7	SOFTRST	Setting this bit will cause the ADV202 to execute a soft reset. This is similar to asserting the RESET/ pin except that the value in BOOT MODE will be used to determine how the ADV202 will boot instead of using the configuration pins. BOOT MODE [2..0] and the PLL control register will not be reset. This bit will be cleared after the soft reset, but all other bits in this register will remain unchanged.	0
15..8	reserved	Reserved for future use; always write 0.	undef

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xE	PLL_HI	PLL Control high byte	R/W	see bit field

The 16 bit PLL_HI register is used to configure the on chip PLL. Internally, the ADV202 uses two clock domains which are generated by an on chip Phase Locked Loop from the input clock MCLK.

JCLK is used to clock all of the JPEG2000 specific hardware blocks and HCLK is used to clock the embedded processor system. A block diagram of the PLL and its control parameters are shown on [Figure 4-3 on page 4-21](#).



Any time the PLL_HI register is changed, the host must wait at least 20us before reading or writing any other register. If this delay is not implemented, erratic behavior may result.

PLL_HI Bit Field			
<i>Bit</i>	<i>Name</i>	<i>Description</i>	<i>Reset Value</i>
0	PLLPDN	Power down enable = 1	0
1	BYPASS	Bypass enabled = 1	0
2	<i>reserved</i>	always write 0	0
3	HCLKD	HCLK divider enabled = 1	1
4	ACK_FN	'0' - ACKn is actively driven '1' - ACK/ is configured as an open drain output	'0'
6..5	reserved	ADI use only; always write 0	undef
7	SM	ADI use only; always write 0	0
15..8	N/A	Not Available	undef

Direct Registers

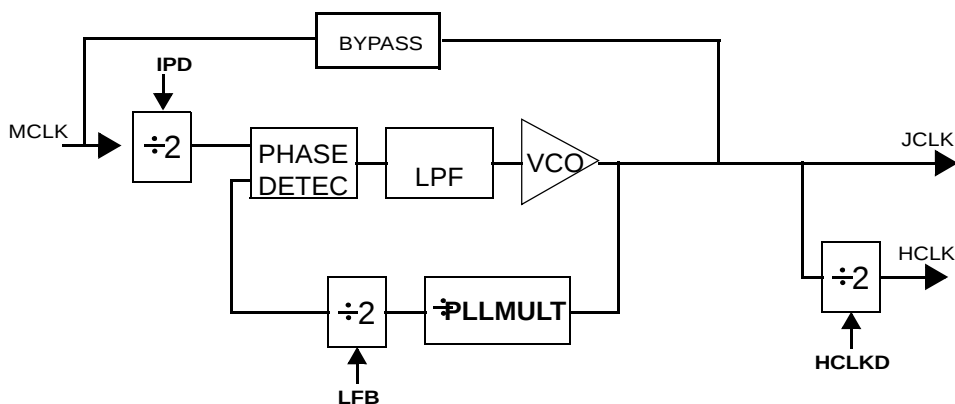
ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xF	PLL_LO	PLL Control low byte	R/W	see bit field

This 16 bit PLL_LO register is used to configure the on chip PLL.



Any time the PLL_LO register is modified, the host must wait at least 20us before reading or writing any other register. If this delay is not implemented, erratic behavior may result.

PLL_LO Bit Field			
Bit	Name	Description	Reset Value
4..0	PLLMULT	Multiplier [values from 0 to 31]	'00011'
5	LFB	Loop feedback divider enable = 1	0
6	reserved	Always write 0	0
7	IPD	Input clock divider enable = 1	0
15..8	N/A	Not applicable	undef



Equivalent transfer function :
$$JCLK = \frac{MCLK * PLLMULT * LFB}{IPD}$$

JCLK & MCLK clk values in Hz

Figure 4-3. PLL Architecture (MCLK is external clock source)

Table 4-2. Recommended Register Settings for PLL register

IPD	LFB	PLLMULT	HCLKD	HCLK	JCLK
0	0	N	0	N * MCLK	N * MCLK
0	0	N	1	N * MCLK / 2	N * MCLK
0	1	N	0	2 * N * MCLK	2 * N * MCLK
0	1	N	1	N * MCLK	2 * N * MCLK
1	0	N	0	N * MCLK / 2	N * MCLK / 2
1	0	N	1	N * MCLK / 4	N * MCLK / 2
1	1	N	0	N * MCLK	N * MCLK
1	1	N	1	N * MCLK / 2	N * MCLK

Direct Registers

Table 4-3. PLL RULES FOR VARIOUS SPEED GRADES

PLL SETTING	115MHZ version	135MHZ version	150MHZ version
JCLK	Greater than 50 Mhz & less than 115 Mhz	Greater than 50 Mhz & less than 135 Mhz	Greater than 50 Mhz & less than 150 Mhz
JCLK	Greater than or equal to 2x VCLK		
HCLK	Less than 81 Mhz	Less than 100 Mhz	Less than 108 Mhz
JDATA Mode, JCLK must be greater than 4x MCLK	All		
Maximum DMA burst frequency is ≤ 0.36 JCLK	All		
If clock input is >50 MHz, the input clock divider must be set (IPD = '1')	All		
Clock input must be ≥ 10 MHz	All		
IPD can not be enabled for clock inputs less than 20 MHz	All		

Listing 4-1.

For example, to achieve the lowest power consumption, an MCLK frequency of 27MHz is recommended for a standard definition CCIR656 input. The PLL circuit is recommended to have a multiplier of 4. This will set JCLK 108MHz, the HCLK divider should be enabled.

Table 4-4. Recommended values for PLL_HI and PLL_LO Registers

VIDEO STANDARD	CLKIN FREQUENCY ON MCLK	PLL_HI	PLL_LO	JCLK
SMPTE125M or ITU-R.BT656 [NTSC or PAL]	27MHz	0x0008	0x0004	108MHz
SMPTE293M [525p]	27MHz	0x0008	0x0004	108MHz
ITU-R.BT1358 [625p]	27MHz	0x0008	0x0004	108MHz
SMPTE274M [1080i]	74.25MHz	0x0008	0x0084	148MHz

Internal Hardware Registers

The indirect registers are accessed by the Host processor through the IADDR and IDATA direct registers. If a 16 bit host is used, the STAGE direct register is also used. See [Figure 4-2 on page 4-3](#). A common indirect access by the external host processor is to write indirect registers, such as the EDMOD and FIFO threshold registers.

In certain modes, such as Custom Specific or HIPI mode, indirect registers are accessed by the Host. In standard modes, many of the indirect registers are controlled by the internal ADV202 processor. In these standard modes, the host processor simply needs to configure the indirect DMA or FIFO threshold registers.

Both 32 bit and 16 bit hosts can access the indirect registers. 32 bit access use the IADDR and IDATA registers while 16 bit hosts use IADDR, IDATA, and the STAGE register. Additional information on accessing and configuring these registers can be found in the “*Getting Started with the ADV202*” programming guide.

Internal Hardware Registers

If the CTRLREGAM bit is set (1), the user only needs to load the IADDR register with the lower 16 bits address [15..0]. the upper address bits default to 0xFFFF. This can be advantageous for 16 bit Hosts. If the CTRLREGAM bit is cleared (0), a full 32-bit address must be written to the IADDR register.

Table 4-5. Internal Hardware Registers

INDIRECT REGISTER ADDRESS	NAME	DESCRIPTION
0xFFFF0400	PMODE1	Pixel/Video Format
0xFFFF0404	COMP_CNT_STATUS	Component count
0xFFFF0408	LINE_CNT_STATUS	Line count
0xFFFF040C	XTOT	Total Samples per line
0xFFFF0410	YTOT	Total Lines per frame
0xFFFF0414	F0_START	Start Line of Field 0 [F0]
0xFFFF0418	F1_START	Start Line of Field 1 [F1]
0xFFFF041C	V0_START	Start of active video Field 0 [F0]
0xFFFF0420	V1_START	Start of active video Field 1 [F1]
0xFFFF0424	V0_END	End of active video Field 0 [F0]
0xFFFF0428	V1_END	End of active video Field 1 [F1]
0xFFFF042C	PIXEL_START	Horizontal start of active video
0xFFFF0430	PIXEL_END	Horizontal end of active video
0xFFFF0440	MS_CNT_DEL	Master/Slave Delay
0xFFFF0444	Reserved	Reserved
0xFFFF0448	PMODE2	Pixel Mode 2
0xFFFF044C	VMODE	Video Mode
0xFFFF1408	EDMOD0	External DMA mode register 0
0xFFFF140C	EDMOD1	External DMA mode register 1

Table 4-5. Internal Hardware Registers (Cont'd)

INDIRECT REGISTER ADDRESS	NAME	DESCRIPTION
0xFFFF1410	FFTHRP	FIFO Threshold for Pixel FIFO
0xFFFF1414	Reserved	Reserved
0xFFFF1418	Reserved	Reserved
0xFFFF141C	FFTHRC	FIFO Threshold for Code FIFO
0xFFFF1420	FFTHRA	FIFO Threshold for ATTR FIFO
0xFFFF1424	Reserved	Reserved
0xFFFF1428	Reserved	Reserved
0xFFFF142C	Reserved	Reserved
0xFFFF1430	Reserved	Reserved
0xFFFF1434 - 0xFFFF14F3	Reserved	Reserved
0xFFFF14F4	HWREV	Hardware Revision Register

Indirect Registers: Configuration and Status

These indirect configuration and status registers are typically controlled by the ADV202, except when in a custom specific video input mode or HIPI mode. In standard video input modes, they do not need to be programmed by the external host processor except for EDMOD and the FIFO threshold registers to configure compressed data transfer to/from the ADV202.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF0400	PMODE1	Pixel Mode 1	R/W	see bit field

The PMODE1 register configures the VDATA or HDATA bus for a specific pixel format. [“Video Input Formats” on page 4-55.](#) for a thorough explanation of the types of video formats used on the VDATA bus and HDATA bus.

PMODE1 Bit Field				
Bit	Name	Description	Reset Value	
4..0	PFMT	VDATA 0x00 - 0x03 Reserved 0x04 Single Component (8, 10, or 12 bit) 0x05 Cb/Y/Cr/Y interleaved (8, 10, or 12 bit) 0x06 Reserved 0x07 Cb/Cr interleaved (8, 10, or 12 bit)	0x05	
		HDATA 0x14 32-bit 4x8-bit Single Component 0x15 32-bit 4x8-bit Packed YCbYCr 0x16 32-bit 4x8-bit Packed YYCbCr 0x17 32-bit 4x8-bit Packed CbCrCbCr 0x18 32-bit 2x16-bit Packed Single Component 0x19 32-bit 2x16-bit Packed YCbYCr 0x1A 32-bit 2x16-bit Packed YY/CbCr 0x1B 32-bit 2x16-bit Packed CbCr 0x1C -0x1F Reserved		
7..5	Reserved	Always write 0	0	

PMODE1 Bit Field			
<i>Bit</i>	<i>Name</i>	<i>Description</i>	<i>Reset Value</i>
10..8	PREC	0x0 8-bit precision 0x1 10-bit precision 0x2 12-bit precision 0x3 14-bit precision 0x4 16-bit precision 0x05 - 0x7 Reserved	0x01
15..11	Reserved	Always write 0	0

Internal Hardware Registers

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF0404	COMP_CNT_STATUS	Sample Count	R	0x0000

The 16 bit COMP_CNT_STATUS register indicates the current value of the Pixel Interface horizontal sample counter. This register should not be polled during applications.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF0408	LINE_CNT_STAT	Line Count.	R	0x0000

The 16 bit LINE_CNT_STAT register indicates the current value of the Pixel Interface line counter. This register should not be polled during applications.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF040C	XTOT	Total samples per line	R/W	0x064B

The 16 bit XTOT register is used to set total number of samples per line. For monochrome input, the number of samples is equivalent to the number of pixels. For YCbCr video, the number of samples is twice the number of pixels per line.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF0410	YTOT	Total lines per frame	R/W	0x020D

The 16 bit YTOT register is used to set the total lines per frame.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF0414	F0_START	Start of Field 0. Only used in Decode mode to identify at which line number the Field bit will transition from Field 1 to Field 0.	R/W	0x0004

The 16 bit F0_START register is used to indicate the start of Field 0 in units of number of lines. Line count starts with line 1. [Table 4-4 on page 4-53.](#)

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFFF0418	F1_START	Start of field 1. Only used in Decode mode to identify at which line number the Field bit will transition from Field 0 to Field 1.	R/W	0x010A

The 16 bit F1_START register is used to indicate the start of Field 1 in units of number of lines. Line count starts with line 1. [Table 4-4 on page 4-53](#)

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFFF041C	V0_START	Start of active video in field 0	R/W	0x0014

The 16 bit V0_START register is used to set the first active video line in Field 0 that will be captured by the ADV202. Line count starts with line 1. [Table 4-4 on page 4-53](#)

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFFF0420	V1_START	Start of active video in field 1	R/W	0x011B

The 16 bit V1_START register is used to set the first active video line in Field 1 that will be captured by the ADV202. Line count starts with line 1. [Table 4-4 on page 4-53](#)

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFFF0424	V0_END	End of active video in field 0	R/W	0x0107

Internal Hardware Registers

The 16 bit V0_END register is used to set the vertical end of the active video in Field 0. This register should hold the value of the last active line number. Line count starts with line 1.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFFF0428	V1_END	End of active video in field 1	R/W	0x020D

The 16 bit V1_END register is used to set the vertical end of the active video in Field 1. This register should hold the value of the last active line number. Line count starts with line1. [Table 4-4 on page 4-53](#).

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFFF042C	PIXEL_START	Start of horizontal active video	R/W	0x0001

The 16 bit PIXEL_START register is used to set the horizontal start (i.e. first active sample in the line) of the active video in units of samples (horizontal counter starts at sample 1). [Table 4-4 on page 4-53](#).

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFFF0430	PIXEL_END	End of horizontal active video	R/W	0x05A0

The 16 bit PIXEL_END register is used to set the horizontal end (i.e. last active sample in the line) of the active video in units of samples (horizontal counter starts at sample 1). [Table 4-4 on page 4-53](#)

Indirect Registers: Video Timing and Dimension

This section describes the indirect registers that affect video timing and video dimension. For more detailed information, [“Video Modes” on page 4-53](#). These indirect configuration and status registers are typically controlled by the ADV202, except when in a custom specific video input mode or HIPI mode. In standard video input modes, they do not need to be programmed by the external host processor.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF0440	MS_CNT_DEL	Multi-chip control	R	0

The 16 bit MS_CNT_DEL register is used only in multi-chip sync mode. Refer to Application Note “ADV202 multi-chip application” for additional information.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF0448	PMODE2	Pixel Mode 2	R/W	0

The 16 bit PMODE2 register is mainly used to set up the programmable edge control for the input clock and sync signals.

PMODE2 Bit Field			
Bit	Name	Description	Reset Value
0	VCLK_POL	VCLK active edge 1=rising, 0=falling	1
1	VSYNC_POL	VSYNC active edge 1=rising, 0=falling	0
2	HSYNC_POL	HSYNC active edge 1=rising, 0=falling	0
3	FIELD_POL	FIELD active edge 1=rising, 0=falling	0

Internal Hardware Registers

PMode2 Bit Field			
<i>Bit</i>	<i>Name</i>	<i>Description</i>	<i>Reset Value</i>
4	YUNI	Y Unipolar 1=on, 0=off This bit should be set to '1' for most applications.	0
5	CUNI	C Unipolar 1=on, 0=off This bit should be set to '1' for most applications.	0
6	reserved	Always write 0	0
7	reserved	Always write 0	0
8	reserved	Always write 0	0
9	reserved	Always write 0	0
10	reserved	Always write 0	0
11	reserved	Always write 0	0
12	reserved	Always write 0	0
13	reserved	Always write 0	0
14	reserved	Always write 0	0
15	reserved	Always write 0	0

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF044C	VMODE	Video Mode	R/W	see Bit field

The 16 bit VMODE register is controlled internally by the ADV202 (except in custom specific mode or HIPI mode). It sets the video interface basic operating mode.

VMODE Bit Field			
Bit	Name	Description	Reset Value
0	MAS_SLV	Master or Slave operation is only selectable in Decode mode. 1 = MASTER 0 = SLAVE	0
1	ENC_DEC	Sets ENCODE or DECODE mode 1 = ENCODE 0 = DECODE	1
2	MP_656	Video input timing control. 0 = EAV/SAV mode = 0; for video input with embedded timing codes 1 = HVF mode = 1; for video input with separate H,V,F sync signals.	0
3	Reserved	Reserved	0
4	HOST_MODE	Pixel data over HDATA bus. 1 = on 0 = off	0
5	RAW_MODE	Raw video input on the VDATA bus HVF mode is not required to be set in this mode. 1 = on 0 = off	0
6		Always write 0.	0
7	PRGRSV_SCN	Progressive scan mode; Must be set if video input on the VDATA bus is non-interlaced, i.e 1field/frame. 1 = on 0 = off	0
8	Reserved	Always write 0.	0

Internal Hardware Registers

VMODE Bit Field			
<i>Bit</i>	<i>Name</i>	<i>Description</i>	<i>Reset Value</i>
9	Reserved	Always write 0.	0
10	Reserved	Always write 0.	0
11	CNT_RD_EN	Read PI control counters [CO_CNT_STATUS and LINE_CNT_STATUS]. 1 = on 0 = off	0
12..14	Reserved	Always write 0	0
15	Reserved	Always write 0	0

Indirect Registers: External DMA

This Section describes the ADV202's interface to DMA. These registers are required to be programmed by the external host processor, when used for compressed data transfers or for pixel data/ compressed data transfers in HIPI mode. Alternatively Normal Host mode accesses in conjunction with the Threshold registers can be used for compressed data transfers which does not require programming of the EDMOD registers.

.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF1408	EDMOD0	External DMA Mode	R/W	See bit Field

The 16 bit EDMOD0 register is used to assign a data FIFO to Channel 0 and to select transfer modes.

EDMOD0 Bit Field			
Bit	Name	Description	Reset Value
0	DMEN0	Enable external DMA channel 0 0=disabled 1=enabled	0
2..1	DMSEL0	DMA Channel 0 Assignment 00 = Pixel Data 01 = Compressed Data 10 = Attribute Data 11 = Reserved	'00'
5..3	DMMOD0	DMA Channel 0 mode 000 - Dedicated Chip Select DMA mode 001 - Single transfer DREQ/DACK DMA mode 010 - Burst transfer DREQ/DACK DMA mode 011 - JDATA mode 100 - reserved 101 - Single transfer Fly-by DMA mode 110 - Burst transfer Fly-by DMA mode 111 - reserved	'000'

Internal Hardware Registers

EDMOD0 Bit Field			
Bit	Name	Description	Reset Value
8..6	DMBL0	DMA Channel 0 burst length [in number of accesses] 000 = 8 - not available for DWIDTH of 8-bit 001 = 16 - for DWIDTH of 8/16 or 32-bits 010 = 32 - for DWIDTH of 8/16 or 32-bits 011 = 64 - for DWIDTH of 8/16 or 32-bits 100 = 128 - for DWIDTH of 8/16 or 32-bits 110 = 512 - for DWIDTH of 8/16-bit, for 32-bits, not recommended 111 - 1024 - for DWIDTH of 8-bit; not recommended for 16 bit host, not available for 32 bit hostsable.	'000'
9	DR0POL	$\overline{\text{FSRQ0}} / \overline{\text{VALID}}$ polarity 0 = Active Low 1 = Active High	0
10	DA0POL	$\overline{\text{FCS0}} / \overline{\text{HOLD}}$ polarity 0 = Active Low 1 = Active High	0
14..11	DR0PULS	DREQ0 pulse width [in Jclkcycles] If DR0PULS==0, then DREQ0 will remain asserted until DACK0 and RD/ in read mode or WR/ in write mode are asserted.	'0001'
15	Reserved	Reserved for future use. Always write 0.	0

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF140C	EDMOD1	External DMA Mode	R/W	See bit Field

The 16 bit EDMOD1 register is used to assign a data FIFO to Channel 1 and to select transfer modes.



JDATA mode is not available in EDMOD1

EDMOD1 Bit Field			
Bit	Name	Description	Reset Value
0	DMEN1	Enable external DMA channel 1 0=disabled 1=enabled	0
2..1	DMSEL1	DMA Channel 1 assignment 00 = Pixel Data 01 = Compressed Data 10 = Attribute Data 11 = Reserved	'01'
5..3	DMMOD1	DMA Channel 1 mode 000 - Dedicated Chip Select DMA mode 001 - Single transfer DREQ/DACK DMA mode 010 - Burst transfer DREQ/DACK DMA mode 011 - reserved 100 - reserved 101 - Single transfer Fly-by DMA mode 110 - Burst transfer Fly-by DMA mode 111 - reserved	'000'
7..6	DMBL1	DMA Channel 1 burst length [in number of accesses] 000 = 8 - not available for DWIDTH of 8-bit 001 = 16 - for DWIDTH of 8/16 or 32-bits 010 = 32 - for DWIDTH of 8/16 or 32-bits 011 = 64 - for DWIDTH of 8/16 or 32-bits 100 = 128 - for DWIDTH of 8/16 or 32-bits 110 = 512 - for DWIDTH of 8/16-bit, for 32-bits, not recommended 111 - 1024 - for DWIDTH of 8-bit, not recommended for 16-bit hosts, not available for 23 bit hosts	'00'

Internal Hardware Registers

EDMOD1 Bit Field			
Bit	Name	Description	Reset Value
9	DR1POL	$\overline{\text{FSRQ1}}$ polarity 0 = Active Low 1 = Active High	0
10	DA1POL	$\overline{\text{FCS1}}$ polarity 0 = Active Low 1 = Active High	0
14..11	DR1PULS	DREQ1 pulse width [in Jclkcycles]. If DR1PULS==0, then DREQ1 will remain asserted until DACK1 and RD/ in read mode or WR/ in write mode are asserted.	'0001'
15	Reserved	Reserved for future use. Always write 0.	0

EDMOD Register Descriptions

EDMOD0 and EDMOD1 registers configure the ADV202 for external DMA operation. DMA allows transfers to and from memory without the on-board processor being required to perform the transactions, as it is the case in Normal Host mode.

The external DMA registers include settings for Single/Burst DMA, Fly-By Mode DMA, and Dedicated Chip Select (DCS) DMA. These registers are used in conjunction with the Bus Mode (BUSMODE) and MMODE registers.

External DMA Width

The desired data width when accessing data FIFOs is set in the BUS-MODE register. The DWIDTH field in this register (direct address location 0x08) can be programmed to byte, 16-bit , or word (32-bit) data widths. The default data-width is 16-bit.

When accessing the FIFOs through the Direct Registers, the HWIDTH parameter in the BUSMODE register sets the data-width. DWIDTH is used to set the data width when DMA modes are used

DMA Mode - DREQ/DACK DMA Mode and FLY-BY DMA Mode

These registers are used to enable the External DMA interface, assign the desired FIFO to each channel, and select modes of operation. All DMA modes make use of the Data Request ($\overline{\text{DREQ0}}/\overline{\text{DREQ1}}$) and Data Acknowledge ($\overline{\text{DACK0}}/\overline{\text{DACK1}}$) pins.

DREQ/DACK DMA mode and Fly-By DMA mode can be used in single transfer or burst transfer modes.

A programming example for DREQ/DACK DMA mode can be found in the “*Getting Started with the ADV202*” programming guide.

DMA Mode - Dedicated Chip Select DMA Mode

This mode is available for devices which use the level-sensitive logic to monitor the FSRQx signals (as opposed to edge-triggered logic). The $\overline{\text{FSRQx}}$ ($\overline{\text{DREQx}}$) pins will be asserted when data/space is available in the selected FIFO and will be de-asserted when the FIFO is empty/full. The $\overline{\text{FCSx}}$ ($\overline{\text{DACKx}}$) pins are used as FIFO Chip Select signals and should be asserted in conjunction with the $\overline{\text{RD}}/\overline{\text{WE}}$ pins. This mode is enabled via the EDMOD0/EDMOD1 registers. The FSRQx/ polarities can be programmed via the EDMOD0/EDMOD1 registers.

A programming example for DCS mode can be found in the “*Getting Started with the ADV202*” programming guide.

JDATA - DATA Mode

This mode allows streaming input/output of compressed data using the VALID/HOLD protocol and the JDATA[7..0] pins. Note that 32-bit data/host modes and the Dual-Lane Pixel Interface mode are not available

Internal Hardware Registers

when JDATA mode is enabled. To assign HDATA [31..24] pins to JDATA[7..0], the BUSMODE register must first be programmed to JDATA. The polarity of the VALID (DREQ0 pin) and HOLD (DACK0 pin) is set in the EDMOD0 register through bits DR0POL/DA0POL. The DMSEL0 field in EDMOD0 must be set to “1” (Compressed Data). The DMMOD0 field must be set to “3” (JDATA Mode). It is recommended that the user set up all mode bits in EDMOD0 without enabling the interface (DMEN0). Then, the HOLD pin should be asserted [HOLD should be in its active state], and then de-asserted [de-activated] after the JDATA interface has been enabled by either the Host or the ADV202. Data transfers occur when the ADV202 asserts VALID and the external device de-asserts HOLD.

The VALID signal will always function as an output from the ADV202 and the HOLD signal will always function as an input to the ADV202. Data transfers occur when VALID is asserted and HOLD is de-asserted at the rising edge of MCLK.

When JDATA mode is used, the Compressed Data (CODE) FIFO is dedicated to DMA Channel 0. The other data FIFOs (ATTR) are not available when JDATA mode is used. Attempting to access these FIFOs will interfere with JDATA operation.

JDATA is referenced with MCLK and requires a PLL_MULT setting of $Jclk \geq 4 * Mclk$.

BUSMODE/DWIDTH should be set to “8-bit”.

A programming example for JDATA mode can be found in the “*Getting Started with the ADV202*” programming guide.

Single Transfer - DREQ/DACK DMA

In this mode, the ADV202 will assert $\overline{DREQx}$ when there is at least one data transfer available for the assigned FIFO. In encode mode, the $\overline{DREQx}$ assertion indicates that output data is present in the FIFO. In decode mode-

this indicates that space is available in the FIFO for at least one data transfer. The external device must respond with $\overline{\text{DACKx}}$, in conjunction with $\overline{\text{RD}}/\overline{\text{WE}}/\text{HDATA}$ activity. In Single Transfer Fly-By Mode, the functionality of the $\overline{\text{RD}}$ and $\overline{\text{WE}}$ pins is reversed.

Burst Transfer - DREQ/DACK DMA

In this mode, the ADV202 will assert $\overline{\text{DREQx}}$ when there is at least one burst data transfer available for the assigned FIFO. In encode mode, the $\overline{\text{DREQx}}$ assertion indicates that output data is present in the FIFO. In decode mode, this indicates that space is available in the FIFO for at least one burst data transfer. The external device must respond with $\overline{\text{DACKx}}$, in conjunction with $\overline{\text{RD}}/\overline{\text{WE}}/\text{HDATA}$ activity.

In Burst Transfer Fly-By Mode, the functionality of the $\overline{\text{RD}}$ and $\overline{\text{WE}}$ pins is reversed.

Burst Length

FIFO width is always 32-bit words, but the programmed Burst Lengths correspond to the number of accesses (which may be 8/16/32 bits wide). A given FIFO depth can support the same number of word transfers, twice the number of 16-bit ‘word’ transfers, and four times the number of byte transfers.

For example, Burst Length (EDMODx/DMBLx) can be programmed to up to 64 word transfers or 128 16-bit ‘word’ transfers.

If the assigned FIFO does not contain the programmed number of accesses [i.e. words] for the last portion of the compressed field, the FIFO is packed with 0s to ensure that the field ends on a burst boundary.

Indirect Registers: FIFO Block

This Section describes the FIFO Block registers on the ADV202. For additional details on how to use FIFO accesses, [“Indirect Registers: External DMA” on page 4-35](#). These registers need to be programmed by the external host processor.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF1410	FFTHRP	FIFO Threshold for PIXEL FIFO	R/W	0x0000

The 16 bit FFTHRP register sets the threshold detect level for the PIXEL data channel.

If the FIFO is in input mode then the threshold condition will be true whenever the empty count is greater than, or equal to, the threshold value.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF141C	FFTHRC	FIFO Threshold for CODE FIFO Mode	R/W	0x0000

The 16 bit FFTHRC register sets the threshold detect level for the CODE data channel. When the ADV202 is in encode mode, this register is used to assert the interrupt when the number of words or spaces in the FIFO is equal to or greater than the threshold. When the ADV202 is in decode mode, this register is used to assert the interrupt when the number of empty locations in the FIFO is equal to or greater than the threshold. Maximum threshold value for the Code FIFO is 512 words, when the ATTR FIFO is not used.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF1420	FFTHRA	FIFO Threshold for ATTR FIFO	R/W	0x0000

The 16 bit FFTHRA register sets the threshold detect level for the ATTR data channel. When the ADV202 is in encode mode, this register is used to assert the interrupt when the threshold amount of data is in the FIFO. When the ADV202 is in decode mode, this register is used to assert the interrupt when there is threshold amount of space is available in the FIFO (i.e number of empty locations). Maximum threshold value for ATTR FIFO is 256 words. If this threshold setting is used for the ATTR FIFO, the maximum threshold value for the CODE FIFO becomes 256 words.

Other Indirect Registers

This Section describes other registers within the ADV202.

ADDR	REGISTER NAME	REGISTER DESCRIPTION	R/W	RESET VALUE
0xFFFF14F4	HWREV	Hardware Revision Register	R/W	0x000000??

This 32-bit indirect register describes what ADV202 hardware revision is of the chip. This is the hardware revision and has no relation to the firm-ware loaded into the chip.

Table 4-6. HWREV register values

VALUE	DESCRIPTION
0x00000000	ES1 (REVA)
0x00000001	ES2 (REVB)
0x00000002	ES3 (REVC)
0x00000003	ES4 (REVD)
0x00000004	ES5 (REVE)
.....	

Host 32 and 16 bit accesses to Direct/Indirect Registers and Indirect Memory

The ADV202 is versatile in that it can support both 32 and 16 bit host interface sizes. The following tables provide some examples of common bus interface configurations for 32 and 16 bit host processor.

The Direct Registers are accessed by the physical host interface using pins ADDR [3..0], HDATA[31..0], CS/, RD/, WE/, ACK/ pins. The Indirect Registers are accessed by writing and reading the IADDR register and the IDATA register. The IADDR register acts as an indirect pointer to the internal indirect registers and memory. If a 16 bit host is used, the STAGE direct register is also used.

Host 32 and 16 bit accesses to Direct/Indirect Registers and Indirect Memory

Accesses Using a 32 bit Host

The table below shows the common register settings for accessing direct, indirect register and indirect memory for a 32 bit Host. All accesses to indirect registers and indirect memory are done using the IADDR and IDATA direct registers.

Table 4-7. 32 bit Host Interface Settings

MEMORY	LOCATION	BUSMODE REGISTER	MMODE REGISTER	EXAMPLES
32 bit Direct Register	0x00 - 0x0F	DWIDTH = 32 bits HWIDTH = 32 bits	IWIDTH = 32 bits IAUTOSTP = 32 bits	Example: Write 0xFFFF1408 to the IADDR register, then read the IADDR register. Sequence: 1) Write 0xFFFF1408 to the IADDR register 2) Read IADDR register (value should be 0xFFFF1408)
16-bit Direct Register	0x00 - 0x0F	DWIDTH = 32 bits HWIDTH = 32 bits	IWIDTH = 32 bits IAUTOSTP = 32 bits	Example: Write 0x000A to the MMODE register Sequence: 1) Write 0x0000000A to the MMODE register <i>Note: (HDATA[31..16]) bits are ignored</i>

Table 4-7. 32 bit Host Interface Settings

MEMORY	LOCATION	BUSMODE REGISTER	MMODE REGISTER	EXAMPLES
16-bit Indirect Register	0xFFFF0000 - 0xFFFF14C0	DWIDTH = 32 bits HWIDTH = 32 bits	IWIDTH = 32 bits IAUTOSTP = 32 bits	Example: Write 0x0802 to the EDMOD0 register (0xFFFF1408) Sequence: 1) Write 0xFFFF1408 to the IADDR register 2) Write 0x08020000 to the IDATA register Example: Read the EDMOD0 register (0xFFFF1408) Sequence: 1) Write 0xFFFF1408 to the IADDR register 2) Read the IDATA register (value should be 08020000 from previous write) <i>Note: (HDATA[15..0]) bits are ignored on Indirect writes.</i> <i>Indirect Registers are 16-bits aligned on 32-bit boundaries utilizing the [31.15] byte lanes.</i>

Host 32 and 16 bit accesses to Direct/Indirect Registers and Indirect Memory

Table 4-7. 32 bit Host Interface Settings

MEMORY	LOCATION	BUSMODE REGISTER	MMODE REGISTER	EXAMPLES
32-bit Indirect Memory	0x00057000 - 0x00057FF	DWIDTH = 32 bits HWIDTH = 32 bits	IWIDTH = 32 bits IAUTOSTP = 32 bits	Example: Write 0x12345678 to memory location 0x00057F00 Sequence: 1) Write 0x00057F00 to the IADDR register 2) Write 0x12345678 to the IDATA register Example: Read memory location 0x00057F00. Sequence: 1) Write 0x00057F00 to the IADDR register 2) Read the IDATA register (value should be 0x12345678 from previous write)

Accesses Using a 16 bit Host

The table below shows the common register settings for accessing direct, indirect register and indirect memory for a 16 bit Host.



MMODE register settings: Accesses to Indirect Registers and Indirect memory will change depending on the setting of the IWIDTH and IAUTOSTP bits.

When the IAUTOSTP bit (MMODE register) is not disabled, the address pointer will increment 32 or 16 bits, depending on the value loaded. This decreases the HOST overhead by automatically incrementing the address pointer. The HOST can simply continue sequential address writes and reads until finished.

Table 4-8. 16 bit Host Interface Settings

MEMORY	LOCATION	BUSMODE REGISTER	MMODE REGISTER	EXAMPLES
32-bit Direct Register	0x00 - 0x0F	DWIDTH = 16 bits HWIDTH = 16 bits	IWIDTH = 16 bits IAUTOSTP = 16 bits	Example: Write 0xFFFF1408 to the IADDR register Sequence: 1) Write 0xFFFF to the STAGE register. 2) Write 0x1408 to the IADDR register. Example: Read the value written to the IADDR register Sequence: 1) Read the IADDR register (value should be 0xFFFF) 2) Read the STAGE register (value should be 0x1408)
16-bit Direct Register	0x00 - 0x0F	DWIDTH = 16 bits HWIDTH = 16 bits	IWIDTH = 16 bits IAUTOSTP = 16 bits	Example: Write 0x000A to the MMODE register Sequence: 1) Write 0x000A to the MMODE register

Host 32 and 16 bit accesses to Direct/Indirect Registers and Indirect Memory

Table 4-8. 16 bit Host Interface Settings

MEMORY	LOCATION	BUSMODE REGISTER	MMODE REGISTER	EXAMPLES
16-bit Indirect Register	0xFFFF0000 - 0xFFFF14C0	DWIDTH = 16 bits HWIDTH = 16 bits	IWIDTH = 16 bits IAUTOSTP = 32 bits	Example: Write 0x0802 to the EDMOD0 register (0xFFFF1408) Sequence: 1) Write 0xFFFF to the STAGE register. 2) Write 0x1408 to the IADDR register. 3) Write 0x0802 to the IDATA register. Example: Read the value written to the EDMOD0 register (0xFFFF1408) Sequence: 1) Write 0xFFFF to the STAGE register. 2) Write 0x1408 to the IADDR register. 3) Read the IDATA register (value should be 0x0802) <i>Note: Since Indirect Registers are on 32-bit boundaries, the IAUTOSTP bit should be set for 32 bit increments.</i>

Table 4-8. 16 bit Host Interface Settings

MEMORY	LOCATION	BUSMODE REGISTER	MMODE REGISTER	EXAMPLES
32-bit Indirect Memory	0x00057000 - 0x00057FFF	DWIDTH = 16 bits HWIDTH = 16 bits	IWIDTH = 16 bits IAUTOSTP = 16 bits	Example: Write 0x12345678 to memory location 0x00057F00 Sequence: 1) Write 0x0005 to the STAGE register. 2) Write 0x7F00 to the IADDR register. 3) Write 0x1234 to the IDATA register. 4) Write 0x5678 to the IDATA register. Example: Read the value written to memory location 0x00057F00 Sequence: 1) Write 0x0005 to the STAGE register. 2) Write 0x7F00 to the IADDR register. 3) Read the IDATA register.(value should be 0x5678) 4) Read the IDATA register.(value should be 0x1234) <i>Note: In this mode, the read back value is swapped. The written order is Big-Endian, the read order is Little Endian.</i>

Host 32 and 16 bit accesses to Direct/Indirect Registers and Indirect Memory

Table 4-8. 16 bit Host Interface Settings

MEMORY	LOCATION	BUSMODE REGISTER	MMODE REGISTER	EXAMPLES
32-bit Indirect Memory	0x00057000 - 0x00057FFF	DWIDTH = 16 bits HWIDTH = 16 bits	IWIDTH = 32 bits IAUTOSTP = 32 bits	Example: Write 0x12345678 to memory location 0x00057F00 Sequence: 1) Write 0x0005 to the STAGE register. 2) Write 0x7F00 to the IADDR register. 3) Write 0x1234 to the STAGE register. 4) Write 0x5678 to the IDATA register. Continuous writes would be achieved with a write to the STAGE register, followed by a write to the IDATA register. Example: Read the value written to memory location 0x00057F00 Sequence: 1) Write 0x0005 to the STAGE register. 2) Write 0x7F00 to the IADDR register. 3) Read the IDATA register.(value should be 0x1234) 4) Read the STAGE register.(value should be 0x5678) <i>Note: By setting the MMODE register to 0x000A, the STAGE register is utilized for reads and writes instead of just the IDATA register only</i>

Video Modes

This section describes the different modes used by the ADV202.

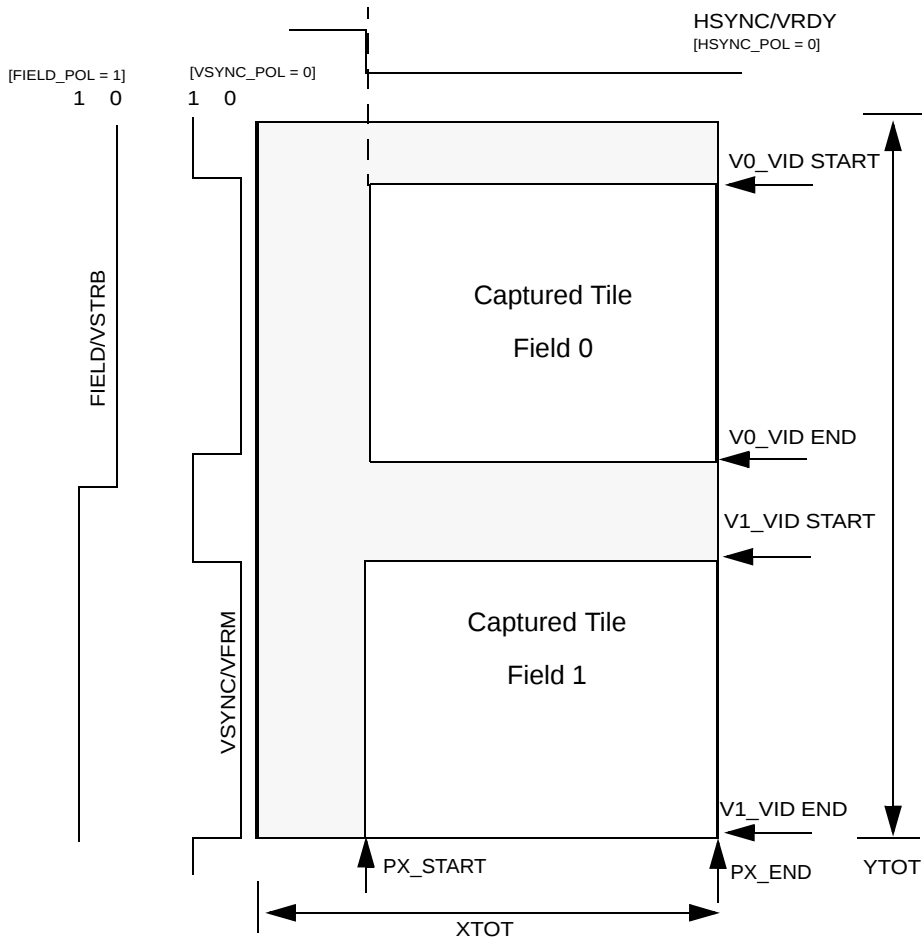


Figure 4-4. Video dimension attributes used in register settings

Video Modes

Encode Mode

In encode mode the ADV202 is always in slave configuration. Input data can be accompanied by separate H,V, F signals or embedded timing codes. In both cases, XTOT, YTOT, V0_START, V1_START, V0_END, V1_END, PIXEL_START, PIXEL_END must reflect the video standard of the input. The ADV202 synchronizes itself to the incoming sync signals or by the embedded sync codes in the video data stream.

Using the value of registers V0_START, V0_END, V1_START, V1_END, PIXEL_START, PIXEL_END the active video region to be processed is calculated. This does apply to all input modes using the VDATA bus; HIPI mode only requires programmed values for XTOT and YTOT while all other dimension register values are ignored [refer to the “ADV202_HIPI_mode” application note].

Decode Slave Mode

In decode slave mode video output is synchronized to the incoming HVF signals. In both cases, XTOT, YTOT, V0_START, V1_START, V0_END, V1_END, PIXEL_START, PIXEL_END must reflect the video standard of the input. The part synchronizes itself to the incoming sync signals. Fields are identified by the incoming FIELD signal.

Decode Master

VSYNC, HSYNC, FIELD, or alternatively EAV/SAV codes, are generated according to the register settings of: XTOT, YTOT, F0_START, F1_START, V0_START, V1_START, V0_END, V1_END, PIXEL_START, PIXEL_END. To enable the generation of these timing signals in decode mode, VMODE register must be programmed to decode master mode and MP_656 must be set to 1 for HVF generation or 0 for EAV/SAV insertion into the video output stream .

Video Input Formats

Pin Input Formats on HDATA and VDATA bus

These tables show the input pin configurations using HDATA and VDATA bus

Table 4-9. Input Pin Configuration and Assignment on VDATA bus

DATA FORMAT	FORMAT	VDATA PINS [MSB]...[LSB]
Video	8-bit	VDATA[11]...[4]
	10-bit	VDATA[11]...[2]
	12-bit	VDATA[11]...[0]
Pixel	8-bit	VDATA[15]...[8]
	10-bit	VDATA[15]...[6]
	12-bit	VDATA[15]...[4]
	16-bit	VDATA[15]...[0]

Table 4-10. Configuration and Assignment on HDATA bus

DATA FORMAT	FORMAT	HDATA (31..24)	HDATA (23..16)	HDATA (15..8)	HDATA (7..0)	
Pixel	8-bit	HDATA [31]...[24]	HDATA [23]...[16]	HDATA [15]...[8]	HDATA [7]...[0]	
		Sn	Sn+1.	Sn+2	Sn+3	1 component
		Cbn	Crn	Cbn+1	Crn+1	2 component

Video Input Formats

Table 4-10. Configuration and Assignment on HDATA bus

DATA FORMAT	FORMAT	HDATA (31..24)	HDATA (23..16)	HDATA (15..8)	HDATA (7..0)	
		Yn, Yn+2...	Yn+1, Yn+3.	Cbn, Cbn+1.	Crn, Crn+1...	3 component
		Yn, Yn+2...	Cbn, Cbn+1.	Yn+1, Yn+3.	Crn, Crn+1...	3 component

Table 4-11. 10, 12, 14 and 16 bit Configuration and Assignment on HDATA bus

DATA FORMAT	FORMAT	HDATA UPPER (msb..lsb)	HDATA LOWER (msb..lsb)	
Pixel	10-bit	HDATA [31]...[22]	HDATA [15]...[6]	
		Sn	Sn+1	1 compo- nent
		Cbn, Cbn+1	Cr, Crn+1	2 compo- nent
		Yn, Yn+1...	Cbn, Crn	3 compo- nent
		Yn, Cbn	Yn+1, Crn	3 compo- nent
Pixel	12-bit	HDATA [31]...[20]	HDATA [15]...[4]	
		Sn	Sn+1	1 compo- nent
		Cbn, Cbn+1	Cr, Crn+1	2 compo- nent
		Yn, Yn+1...	Cbn, Crn	3 compo- nent
		Yn, Cbn	Yn+1, Crn	3 compo- nent

Table 4-11. 10, 12, 14 and 16 bit Configuration and Assignment on HDATA bus

DATA FORMAT	FORMAT	HDATA UPPER (msb..lsb)	HDATA LOWER (msb..lsb)	
Pixel	14-bit	HDATA [31]...[18]	HDATA [15]...[2]	
		Sn	Sn+1	1 compo- nent
		Cbn, Cbn+1	Cr, Crn+1	2 compo- nent
		Yn, Yn+1...	Cbn, Crn	3 compo- nent
		Yn, Cbn	Yn+1, Crn	3 compo- nent
Pixel	16-bit	HDATA [31]...[16]	HDATA [15]...[0]	
		Sn	Sn+1	1 compo- nent
		Cbn, Cbn+1	Cr, Crn+1	2 compo- nent
		Yn, Yn+1...	Cbn, Crn	3 compo- nent
		Yn, Cbn	Yn+1, Crn	3 compo- nent

Video Input Formats on HDATA bus

These Tables show the 8,10, 12, 14, 2x8, and 16 bit video input formats on the HDATA bus. These input formats are controlled by the PFMT and PREC register settings in the PMODE1 register

Table 4-12. Data Alignment for 8 bit component input formats on HDATA bus

8 bit three component YCbYCr4:2:2	8 bit three component YYCbCr4:2:2
msb(7) Y_n lsb(0)	msb(7) Y_n lsb(0)
msb(7) Cb_n lsb(0)	msb(7) Y_{n+1} lsb(0)
msb(7) Y_{n+1} lsb(0)	msb(7) Cb_n lsb(0)
msb(7) Cr_n lsb(0)	msb(7) Cr_n lsb(0)
Repeat	Repeat
8 bit single component	8 bit two component CbCr
msb(7) S_n lsb(0)	msb(7) Cb_n lsb(0)
Repeat	msb(7) Cr_n lsb(0)
	Repeat

Table 4-13. Data Alignment for 10 bit component input formats on HDATA bus

10 bit three component YCbYCr4:2:2			10 bit three component YYCbCr4:2:2		
msb(9)	Y_n	lsb(0)	msb(9)	Y_n	lsb(0)
msb(9)	Cb_n	lsb(0)	msb(9)	Y_{n+1}	lsb(0)
msb(9)	Y_{n+1}	lsb(0)	msb(9)	Cb_n	lsb(0)
msb(9)	Cr_n	lsb(0)	msb(7)	Cr_n	lsb(0)
Repeat			Repeat		
10 bit single component			10 bit two component CbCr		
msb(9)	S_n	lsb(0)	msb(9)	Cb_n	lsb(0)
<i>Repeat</i>			msb(9)	Cr_n	lsb(0)
			<i>Repeat</i>		

Table 4-14. Data Alignment for 12 bit component input formats on HDATA bus

12 bit three component YCbYCr4:2:2			12 bit three component YYCbCr4:2:2		
msb(11)	Y_n	lsb(0)	msb(11)	Y_n	lsb(0)
msb(11)	Cb_n	lsb(0)	msb(11)	Y_{n+1}	lsb(0)
msb(11)	Y_{n+1}	lsb(0)	msb(11)	Cb_n	lsb(0)
msb(11)	Cr_n	lsb(0)	msb(11)	Cr_n	lsb(0)
Repeat			Repeat		
12 bit single component			12 bit two component CbCr		
msb(11)	S_n	lsb(0)	msb(11)	Cb_n	lsb(0)
<i>Repeat</i>			msb(11)	Cr_n	lsb(0)
			<i>Repeat</i>		

Video Input Formats

Table 4-15. Data Alignment for 14 bit component input formats on HDATA bus

14 bit three component YCbYCr4:2:2			14 bit three component YYCbCr4:2:2		
msb(13)	Y_n	lsb(0)	msb(13)	Y_n	lsb(0)
msb(13)	Cb_n	lsb(0)	msb(13)	Y_{n+1}	lsb(0)
msb(13)	Y_{n+1}	lsb(0)	msb(13)	Cb_n	lsb(0)
msb(13)	Cr_n	lsb(0)	msb(13)	Cr_n	lsb(0)
Repeat			Repeat		
14 bit single component			14 bit two component CbCr		
msb(13)	S_n	lsb(0)	msb(13)	Cb_n	lsb(0)
Repeat			msb(13)	Cr_n	lsb(0)
			Repeat		

Table 4-16. Data Alignment for 2x8 bit three component input formats on HDATA bus

2x8 bit three component YCbYCr4:2:2					
msb(7)	Y_n	lsb(0)	msb(7)	Cb_n	lsb(0)
msb(7)	Y_{n+1}	lsb(0)	msb(7)	Cr_n	lsb(0)
Repeat					
2x8 bit three component YYCbYCr4:2:2					
msb(7)	Y_n	lsb(0)	msb(6)	Y_{n+1}	lsb(0)
msb(7)	Cb_n	lsb(0)	msb(6)	Cr_n	lsb(0)
			Repeat		

Table 4-17. Data Alignment for 2x8 bit single and two component input formats on HDATA bus

2x8 bit two component CbCr					
msb(7)	Cb _n	lsb(0)	msb(7)	Cr _n	lsb(0)
<i>Repeat</i>					
2x8 bit single component					
msb(7)	S _n	lsb(0)	msb(7)	S _{n+1}	lsb(0)
<i>Repeat</i>					

Table 4-18. Data Alignment for 16 bit input formats on HDATA bus

16 bit three component YCbYCr4:2:2			16 bit three component YYCbCr4:2:2		
msb(15)	Y _n	lsb(0)	msb(15)	Y _n	lsb(0)
msb(15)	Cb _n	lsb(0)	msb(15)	Y _{n+1}	lsb(0)
msb(15)	Y _{n+1}	lsb(0)	msb(15)	Cb _n	lsb(0)
msb(15)	Cr _n	lsb(0)	msb(15)	Cr _n	lsb(0)
<i>Repeat</i>			<i>Repeat</i>		
16 bit single component			16 bit two component CbCr		
msb(15)	S _n	lsb(0)	msb(15)	Cb _n	lsb(0)
<i>Repeat</i>			msb(15)	Cr _n	lsb(0)
			<i>Repeat</i>		

Video Input Formats on VDATA bus

These Tables show the 8, 10, 12 bit video input formats. These input formats are controlled by the PFMT and PREC register settings in the PMODE1 register



All data transmitted on the VDATA bus is always MSB aligned (left justified); unused pins on the bus are read as 0 while writes on the unused pins are ignored by the ADV202. In HVF and EAV/SAV modes, the VDATA[11..0] pins are used.

For RAW Pixel Modes, the Bus Configuration, BCFG, must be programmed to 010. (See the BUSMODE direct register[6:4], BCFG). This setting enables the extended VDATA bus. If not set, VDATA[15:13] will be ignored by the Pixel Interface.

8 bit Video Interface

Table 4-19. 8 bit three component - CbYCrY - EAV/SAV Mode

VDATA(11)		VDATA(4)	VDATA(3..0)
msb(7)	0xFF	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	SAV	lsb(0)	'0000'
msb(7)	Cb _n	lsb(0)	'0000'
msb(7)	Y _n	lsb(0)	'0000'
msb(7)	Cr _n	lsb(0)	'0000'
msb(7)	Y _{n+1}	lsb(0)	'0000'
<i>Repeat</i>			
msb(7)	0xFF	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	EAV	lsb(0)	'0000'

Table 4-20. 8 bit two component - CbCr - EAV/SAV Mode

VDATA(11)		VDATA(4)	VDATA(3..0)
msb(7)	0xFF	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	SAV	lsb(0)	'0000'
msb(7)	Cb _n	lsb(0)	'0000'
msb(7)	Cr _n	lsb(0)	'0000'

Video Input Formats

Table 4-20. 8 bit two component - CbCr - EAV/SAV Mode

VDATA(11)		VDATA(4)	VDATA(3..0)
msb(7)	Cb_{n+1}	lsb(0)	'0000'
msb(7)	Cr_{n+1}	lsb(0)	'0000'
<i>Repeat</i>			
msb(7)	0xFF	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	EAV	lsb(0)	'0000'

Table 4-21. 8 bit single component - EAV/SAV Mode

VDATA(11)		VDATA(4)	VDATA(3..0)
msb(7)	0xFF	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	SAV	lsb(0)	'0000'
msb(7)	S_n	lsb(0)	'0000'
msb(7)	S_{n+1}	lsb(0)	'0000'
msb(7)	S_{n+2}	lsb(0)	'0000'
<i>Repeat</i>			
msb(7)	0xFF	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	EAV	lsb(0)	'0000'

Table 4-22. 8 bit three component - CbYCrY - HVF Mode

VDATA(11)		VDATA(4)	VDATA(3..0)
msb(7)	Cb _n	lsb(0)	'0000'
msb(7)	Y _n	lsb(0)	'0000'
msb(7)	Cr _n	lsb(0)	'0000'
msb(7)	Y _{n+1}	lsb(0)	'0000'
<i>Repeat</i>			

Table 4-23. 8 bit two component - CbCr - HVF Mode

VDATA(11)		VDATA(4)	VDATA(3..0)
msb(7)	Cb _n	lsb(0)	'0000'
msb(7)	Cr _n	lsb(0)	'0000'
msb(7)	Cb _{n+1}	lsb(0)	'0000'
msb(7)	Cr _{n+1}	lsb(0)	'0000'
<i>Repeat</i>			

Table 4-24. 8 bit single component - HVF Mode

VDATA(11)		VDATA(4)	VDATA(3..0)
msb(7)	S _n	lsb(0)	'0000'
<i>Repeat</i>			

Table 4-25. 8 bit single component - Raw Mode

VDATA(15)		VDATA(8)	VDATA(7..0)
msb(7)	S _n	lsb(0)	'00'
<i>Repeat</i>			

Video Input Formats

10 bit Video Interface

Table 4-26. 10 bit three component YCbCr EAV/SAV Mode

VDATA(11)		VDATA(4)	VDATA(3..2)	VDATA(1..0)
msb(7)	0xFF	lsb(0)	‘00’	‘00’
msb(7)	0x00	lsb(0)	‘00’	‘00’
msb(7)	0x00	lsb(0)	‘00’	‘00’
msb(7)	SAV	lsb(0)	‘00’	‘00’
msb(9)	Cb _n		lsb(0)	‘00’
msb(9)	Y _n		lsb(0)	‘00’
msb(9)	Cr _n		lsb(0)	‘00’
msb(9)	Y _{n+1}		lsb(0)	‘00’
Repeat				
msb(7)	0xFF	lsb(0)	‘00’	‘00’
msb(7)	0x00	lsb(0)	‘00’	‘00’
msb(7)	0x00	lsb(0)	‘00’	‘00’
msb(7)	EAV	lsb(0)	‘00’	‘00’

Table 4-27. 10 bit two component CbCr EAV/SAV Mode

VDATA(11)		VDATA(4)	VDATA(3..2)	VDATA(1..0)
msb(7)	0xFF	lsb(0)	'00'	'00'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	SAV	lsb(0)	'00'	'00'
msb(9)	Cb _n		lsb(0)	'00'
msb(9)	Cr _n		lsb(0)	'00'

Table 4-27. 10 bit two component CbCr EAV/SAV Mode

VDATA(11)		VDATA(4)	VDATA(3..2)	VDATA(1..0)
msb(9)	Cb _{n+1}	lsb(0)		'00'
msb(9)	Cr _{n+1}	lsb(0)		'00'
<i>Repeat</i>				
msb(7)	0xFF	lsb(0)	'00'	'00'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	EAV	lsb(0)	'00'	'00'

Table 4-28. 10 bit single component - EAV/SAV Mode

VDATA(11)		VDATA(4)	VDATA(3..2)	VDATA(1..0)
msb(7)	0xFF	lsb(0)	'00'	'00'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	SAV	lsb(0)	'00'	'00'
msb(9)	S _n	lsb(0)		'00'
msb(9)	S _{n+1}	lsb(0)		'00'
msb(9)	S _{n+2}	lsb(0)		'00'
msb(9)	S _{n+3}	lsb(0)		'00'
<i>Repeat</i>				
msb(7)	0xFF	lsb(0)	'00'	'00'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	0x00	lsb(0)	'00'	'00'
msb(7)	EAV	lsb(0)	'00'	'00'

Video Input Formats

Table 4-29. 10 bit three component - CbYCrY - HVF Mode

VDATA(11)		VDATA(2)	VDATA(1..0)
msb(9)	Cb _n	lsb(0)	'00'
msb(9)	Y _n	lsb(0)	'00'
msb(9)	Cr _n	lsb(0)	'00'
msb(9)	Y _{n+1}	lsb(0)	'00'
<i>Repeat</i>			

Table 4-30. 10 bit two component - CbCr - HVF Mode

VDATA(11)		VDATA(2)	VDATA(1..0)
msb(9)	Cb _n	lsb(0)	'00'
msb(9)	Cr _n	lsb(0)	'00'
msb(9)	Cb _{n+1}	lsb(0)	'00'
msb(9)	Cr _{n+1}	lsb(0)	'00'
<i>Repeat</i>			

Table 4-31. 10 bit single component - HVF Mode

VDATA(11)		VDATA(2)	VDATA(1..0)
msb(9)	S _n	lsb(0)	'00'
<i>Repeat</i>			

Table 4-32. 10 bit single component - Raw Mode

VDATA(15)		VDATA(6)	VDATA(5..0)
msb(9)	S _n	lsb(0)	'00'
<i>Repeat</i>			

12 bit Video Interface

Table 4-33. 12 bit three component YCbCr EAV/SAV Mode

VDATA(11)			VDATA(3..0)
msb(7)	0xFF	lsb(0)	‘0000’
msb(7)	0x00	lsb(0)	‘0000’
msb(7)	0x00	lsb(0)	‘0000’
msb(7)	SAV	lsb(0)	‘0000’
msb(11)	Cb _n		lsb(0)
msb(11)	Y _n		lsb(0)
msb(11)	Cr _n		lsb(0)
msb(11)	Y _{n+1}		lsb(0)
Repeat			
msb(7)	0xFF	lsb(0)	‘0000’
msb(7)	0x00	lsb(0)	‘0000’
msb(7)	0x00	lsb(0)	‘0000’
msb(7)	EAV	lsb(0)	‘0000’

Table 4-34. 12 bit two component CbCr EAV/SAV Mode

VDATA(11)			VDATA(3..0)
msb(7)	0xFF	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	SAV	lsb(0)	'0000'
msb(11)	Cb _n		lsb(0)
msb(11)	Cr _n		lsb(0)

Video Input Formats

Table 4-34. 12 bit two component CbCr EAV/SAV Mode

VDATA(11)			VDATA(3..0)
msb(11)	Cb _{n+1}	lsb(0)	
msb(11)	Cr _{n+1}	lsb(0)	
<i>Repeat</i>			
msb(7)	0xFF	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	EAV	lsb(0)	'0000'

Table 4-35. 12 bit single component - EAV/SAV Mode

VDATA(11)			VDATA(3..0)
msb(7)	0xFF	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	SAV	lsb(0)	'0000'
msb(11)	S _n	lsb(0)	
msb(11)	S _{n+1}	lsb(0)	
msb(11)	S _{n+2}	lsb(0)	
msb(11)	S _{n+3}	lsb(0)	
<i>Repeat</i>			
msb(7)	0xFF	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	0x00	lsb(0)	'0000'
msb(7)	EAV	lsb(0)	'0000'

Table 4-36. 12 bit three component - CbYCrY - HVF Mode

VDATA(11)			VDATA(0)
msb(11)	Cb _n		lsb(0)
msb(11)	Y _n		lsb(0)
msb(11)	Cr _n		lsb(0)
msb(11)	Y _{n+1}		lsb(0)
<i>Repeat</i>			

Table 4-37. 12 bit two component - CbCr - HVF Mode

VDATA(11)			VDATA(0)
msb(11)	Cb _n		lsb(0)
msb(11)	Cr _n		lsb(0)
msb(11)	Cb _{n+1}		lsb(0)
msb(11)	Cr _{n+1}		lsb(0)
<i>Repeat</i>			

Table 4-38. 12 bit single component - HVF Mode

VDATA(11)			VDATA(1..0)
msb(11)	S _n		lsb(0)
<i>Repeat</i>			

Table 4-39. 12 bit single component - Raw Mode

VDATA(15)			VDATA(4)
msb(11)	S _n		lsb(0)
<i>Repeat</i>			

