ADuCM4050 パワー・マネージメント搭載の
超低消費電力 ARM Cortex-M4F MCU
ハードウェア・リファレンス

Revision 1.1, November 2018
Part Number
82-100132-01

アナログ・デバイセズ株式会社

本　　社／〒105-6891　東京都港区海岸 1-16-1　ニューピア竹芝サウスタワービル 10F
電話 03（5402）8200
大　阪営業所／〒532-0003　大阪府大阪市淀川区宮原 3-5-36　新大阪トラストタワー 10F
電話 06（6350）6868
名古屋営業所／〒451-6038　愛知県名古屋市西区牛島町 6-1　名古屋ルーセントタワー 38F
電話 052（569）6300

通知条項

著作権情報

印刷国：アメリカ合衆国

免責条項

アナログ・デバイセズは、予告なしに本製品を変更する権利を有します。アナログ・デバイセズの提供する情報は、正確で信頼できるものであることを期していますが、その情報の利用に関して、あるいはその利用によって生じる第三者の特許やその他の権利の侵害に関して一切の責任を負いません。また、アナログ・デバイセズ社の特許の権利の使用を明示的または暗示的に許諾するものでもありません。

商品商標および役務商標に関する表示

アナログ・デバイセズのロゴ、Blackfin、Blackfin+、CrossCore、EngineerZone、EZ-Board、EZ-KIT Lite、EZ-KIT Mini、EZ-Extender、SHARC、SHARC+、A^2B、および VisualDSP++は、アナログ・デバイセズ社の登録商標です。

SigmaStudio は、アナログ・デバイセズ社の商標です。

他のブランドおよび製品名はすべて各社の登録商標または役務商標です。

目次

パワー・マネージメント（PMG）

まえがき

アナログ・デバイセズの ADuCM4050 マイクロコントローラ・ユニット（MCU）をご購入いただき、ありがとうございます。本製品をシステム開発にご利用いただけましたら幸いです。

本マニュアルの目的

ADuCM4050 パワー・マネージメント搭載の超低消費電力 ARM® Cortex®-M4F MCU ハードウェア・リファレンスは、ADuCM4050 マイクロコントローラのアーキテクチャについて説明するもので、マイクロコントローラの主要なアーキテクチャ情報を提供します。

プログラミングの詳細については、ARM Information Center（http://infocenter.arm.com/help/index.jsp）にアクセスしてください。

ARM Cortex-M4F プロセッサのプログラミング関連の技術文書には、次のものがあります。

- ARM Cortex-M4F Devices Generic User Guide
- ARM Cortex-M4F Technical Reference Manual

タイミング仕様、電気仕様、パッケージ仕様については、ADuCM4050 パワー・マネージメント搭載の超低消費電力 ARM Cortex-M4F MCU のデータシートを参照してください。

対象読者

本マニュアルは、アナログ・デバイセズのプロセッサに関する知識のあるプログラマを主な対象としており、読者は該当するプロセッサのアーキテクチャと命令セットに関する実用的な知識を有するものと想定しています。アナログ・デバイセズのプロセッサを使い慣れていないプログラマも本マニュアルを使用できますが、対象となるアーキテクチャについて解説したプログラミング・リファレンス・ブックやデータシートなどの資料を補足的に使用することをお奨めします。

本マニュアルの変更点

本マニュアルは、ADuCM4050 パワー・マネージメントを統合した超低消費電力 ARM Cortex-M4F MCU ハードウェア・リファレンスの最初のリビジョン（1.1）です。このリビジョンでは、暗号化（CRYPTO）、リアルタイム・クロック（RTC）、ユニバーサル非同期レシーバー／トランスミッタ（UART）、I2C（Inter-Integrated Circuit）インターフェース、ウォッチドッグ・タイマー（WDT）の章を更新しています。

技術／カスタマ・サポート

アナログ・デバイセズのプロセッサに関する技術／カスタマ・サポートについては、以下の方法がございます。

- 下記 URL の ***EngineerZone***（英語）では、アナログ・マイクロコントローラの設計上の困難な課題に関する質問をしたり、豊富な技術情報を参照したりすることができます。

http://ez.analog.com/community/analog-microcontrollers（英語）

- 弊社製品に関する技術的なお問い合わせについては、下記 URL にて承っております。製品選定に関するご相談や、スペックに関するご質問などもお気軽にお問い合わせください。
  http://www.analog.com/jp/techsupport
- ソフトウェア／ハードウェア開発ツールの質問については、下記メールアドレス（英語）でも承っております。
  processor.tools.support@analog.com（英語）
- プロセッサに関するお問い合わせは、下記メールアドレス（英語）にて承っております。
  iot_support@analog.com（英語）
- アナログ・デバイセズの販売拠点または正規販売代理店につきましては、下記 URL をご参照ください。
  http://www.analog.com/jp/adi-sales

製品情報

製品情報はアナログ・デバイセズの Web サイトならびにオンライン・ヘルプ・システムで入手できます。

アナログ・デバイセズの Web サイト

アナログ・デバイセズの Web サイト（http://www.analog.com）では、アナログ集積回路、アンプ、コンバータ、デジタル・シグナル・プロセッサなど、広範な製品に関する情報を提供しています。

各プロセッサ・ファミリの詳細な技術ライブラリをご覧いただくには、http://www.analog.com/processors/technical_library にアクセスしてください。各製品に関連した最新のマニュアルと以前のバージョンへのリンクの一覧が開きます。目的のマニュアルのタイトルを指定する際、タイトルの横にある正誤表のチェックマークに注意してください。マークがある場合は、そのマニュアルの最新の修正報告を確認できます。

MyAnalog.com は、アナログ・デバイセズの Web サイトの無料機能で、これにより Web ページをカスタマイズして興味のある製品の最新情報だけを表示することができます。全てのマニュアルの修正点など、ご希望の Web ページの更新をお知らせする電子メールを毎週受信するように選択することも可能です。MyAnalog.com では、書籍、アプリケーション・ノート、データシート、コード例などの様々な情報にアクセスできます。

MyAnalog.com にご登録ください。既に登録済みの場合は、ぜひログオンしてください。お使いのメールアドレスがユーザ名となります。

EngineerZone

EngineerZone はアナログ・デバイセズの技術サポート・フォーラムです。これにより、アナログ・デバイセズの技術サポート・エンジニアに直接問い合わせが可能です。FAQ や技術情報を検索すれば、組み込みプロセッシングや DSP 設計の疑問に対する答えを即座に得ることができます。

同じような設計課題を抱えている他の MCU 開発者と連絡を取り合うには、EngineerZone をご利用ください。また、このオープン・フォーラムを利用して、アナログ・デバイセズのサポート・チームや他の開発者と知識を共有し連携することが可能です。ぜひ、http://ez.analog.com にご登録ください。

表記規則

本マニュアルで使用する文字の表記規則は次のとおりです。なお、本マニュアルでは、特定の章にのみ適用される規則が追加されることがあります。

Example	説明
File > *Close*	表示部分の名称は、CrossCore Embedded Studio IDE のメニュー・システム内の項目の場所を示します（この例では、***Close*** コマンドは ***File*** メニューにあります）。
{this | that}	構文記述での選択必須項目は、波カッコ内に縦線で区切って表示されます。この例では、this または that のどちらかを選択する必要があります。
[this | that]	構文記述でのオプション項目は、角カッコ内に縦線で区切って表示されます。この例は、this または that の任意選択を意味します。
[this, …]	構文記述での任意選択項目の列記は、各括弧内にコンマで区切って表示され、最後に省略形が表示されます。この例では、this のオプションがカンマで区切って列記されます。
.SECTION	コマンド、ディレクティブ、キーワード、機能名の文字は、Letter Gothic フォントで表示します。
filename	キーワード以外のプレースホルダは、イタリック書体で表示します。
注：	関連トピックの補足情報を提供します。 **注：**正しく動作させるには、…
CAUTION: （本書では記載なし）	不測の結果やデバイスの損傷を招くおそれのある、製品の状態や不適切な使用法を示します。 CAUTION: …の場合、デバイスの誤動作を招くおそれがあります。 CAUTION: …の場合、デバイスが損傷するおそれがあります。
ATTENTION: （本書では記載なし）	デバイスの使用者に危険が生じる可能性のある、製品の状態や不適切な使用法を示します。 **ATTENTION:** …の場合、デバイスの使用者がけがをするおそれがあります。

レジスタ図の規則

レジスタ図には次の規則が用いられます。

- レジスタを説明する名称が最初に表示され、続いてその短縮名がカッコ内に表示されます。
- レジスタが読出し専用（RO）、1 を書き込むとセット（W1S）、1 を書き込むとクリア（W1C）のいずれかの場合、この情報が名前の下に表示されます。読出し／書込みはデフォルトであり、記載されません。この情報の後に、追加説明が続くことがあります。
- レジスタ内のいずれかのビットが全体的な読出し／書込み規則に従わない場合、ビット名の後のビット説明にその旨が記されます。
- ビットに短縮名がある場合は、その短縮名がビット説明の最初に表示され、続いてフルネームが括弧内に表示されます。
- リセット値は各ビットにバイナリ表示され、レジスタの右側に 16 進表示されます。
- X で印されたビットには未知のリセット値があります。そのため、そのようなビットを含むレジスタのリセット値は定義されないか、またはリセット時のピンの値で決まります。
- 網掛け表示のビットは予約済みです。

注：将来の実装に対する上位互換性を確保するため、特に指定のない場合、レジスタの予約済みビットで読み出した値は書き戻してください。

レジスタの説明の表では、次の規則が用いられます。

- 各ビットやビット・フィールドのアクセス・タイプは、表のビット番号の下に（読出しアクセス／書込みアクセス）の形で表示されます。アクセス・タイプは次のとおりです。
- R = 読出し、RC = 読出しクリア、RS = 読出しセット、R0 = ゼロを読出し、R1 = 1 を読出し、Rx = 未定義の読出し
- W =書込み、NW = 書込みなし、W1C = 1 を書き込むとクリア、W1S = 1 を書き込むとセット、W0C = ゼロを書き込むとクリア、W0S = ゼロを書き込むとセット、WS = 書き込むとセット、WC = 書き込むとクリア、W1A = 1 を書き込むとアクション。
- ビットの説明やビット・フィールドの説明によっては、ビット値と関連機能が列記されています。プレフィックスで示していない限り、列記されている数値は 10 進値です。

1 はじめに

ADuCM4050 MCU は、処理、制御、接続を統合したパワー・マネージメント機能を備えた超低消費電力のマイクロコントローラ・システムです。この MCU システムは、ARM Cortex-M4F プロセッサをベースとし、一連のデジタル・ペリフェラルおよび組み込み SRAM と組み込みフラッシュ・メモリに加え、クロック、リセット、パワー・マネージメント機能を持つアナログ・サブシステムや A/D コンバータ（ADC）サブシステムを備えています。

動作時と休止時の消費電力を極めて低く抑えるために、ADuCM4050 MCU はダイナミック／ソフトウェア制御のクロック・ゲートや電力ゲートなど、多数の電力モードと機能を備えています。

ADuCM4050 MCU の機能

ADuCM4050 MCU は以下のような機能を備えています。

- 最大 52MHz の ARM Cortex-M4F プロセッサ
- ECC 付き 512KB 組み込みフラッシュ・メモリ
- パリティ付き 128KB システム SRAM
- ユーザ設定可能なパリティ付き 32KB 命令／データ SRAM
  4 キロバイトの SRAM をキャッシュ・メモリに使用して、フラッシュ・メモリへのアクセスを減らすことで、実効的な消費電力を削減できます。
- パワー・マネージメント・ユニット（PMU）
- パワーオン・リセット（POR）と電源モニタ（PSM）
- アクティブ状態と Flexi™ 状態での効率を向上させる降圧コンバータ
- 多層 AMBA バス・マトリックス
- マルチチャンネル中央ダイレクト・メモリ・アクセス（DMA）コントローラ
- プログラマブル GPIO
- 2 つの UART ペリフェラル・インターフェース
- 1 つの I^2C インターフェース
- 幅広いセンサーやコンバータと切れ目なく接続可能な 3 つの SPI インターフェース

- 幅広い無線やコンバータと接続可能なシリアル・ポート（SPORT）2 つの単方向ハーフ SPORT または 1 つの双方向フル SPORT
- シングルトーンとマルチトーンの再生オプションを提供するビーパ・ドライバ
- 正確な実時間を維持できるリアルタイム・クロック（RTC）
- 広範囲のウェイクアップ・タイムに対応する柔軟なリアルタイム・クロック（FLEX_RTC）
- 3 つの汎用タイマーと 1 つのウォッチドッグ・タイマー
- AES-128、様々なモード（ECB、CBC、CTR、CBC-MAC、CCM、CCM*）と SHA-256 を利用した AES-256、HMAC モードに対応する暗号ハードウェア
- 鍵のラップ／アンラップで保護された鍵の保存
- 鍵のアンラップを備えた鍵付き HMAC
- 真性乱数ジェネレータ（TRNG）
- プログラマブル・ジェネレータ多項式によるハードウェア CRC
- マルチパリティ・ビットで保護された SRAM
- 12 ビット、1.8MSPS の SAR ADC
- RGB LED を駆動する RGB タイマー

ADuCM4050 の機能の説明

ここでは、ADuCM4050 MCU の機能に関する情報を説明します。

ADuCM4050 のブロック図

ARM Cortex-M4F コアは、32 ビットの縮小命令セット・コンピュータ（RISC）で、52MHz 時に最大 66MIPS のピーク性能を発揮します。ペリフェラルとメモリ間のデータ移動を効率的に行うため、中央 DMA コントローラが使用されています。

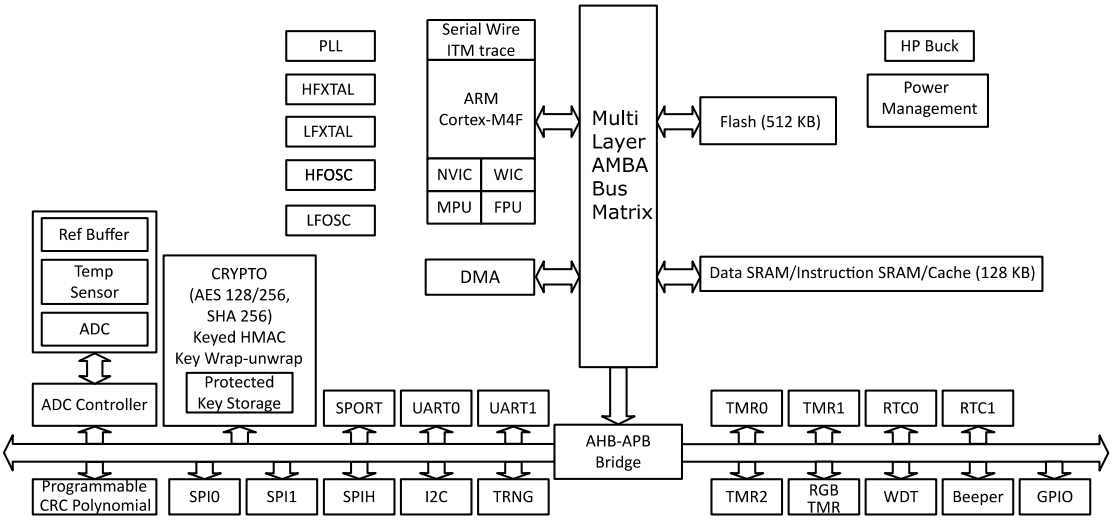


図 1-1：ADuCM4050 のブロック図

メモリ・アーキテクチャ

ADuCM4050 MCU は、プログラム・コードと不揮発性データ・ストレージに使用する 512KB の組み込みフラッシュ・メモリ、96KB のデータ SRAM、32KB の SRAM（命令スペースまたはデータ・スペースとして構成）を実装しています。堅牢性と信頼性を補強するため、フラッシュ・メモリでは ECC がイネーブルされており、SRAM のランダムなソフト・エラーを検出するためにマルチ・ビット・パリティが使用できます。

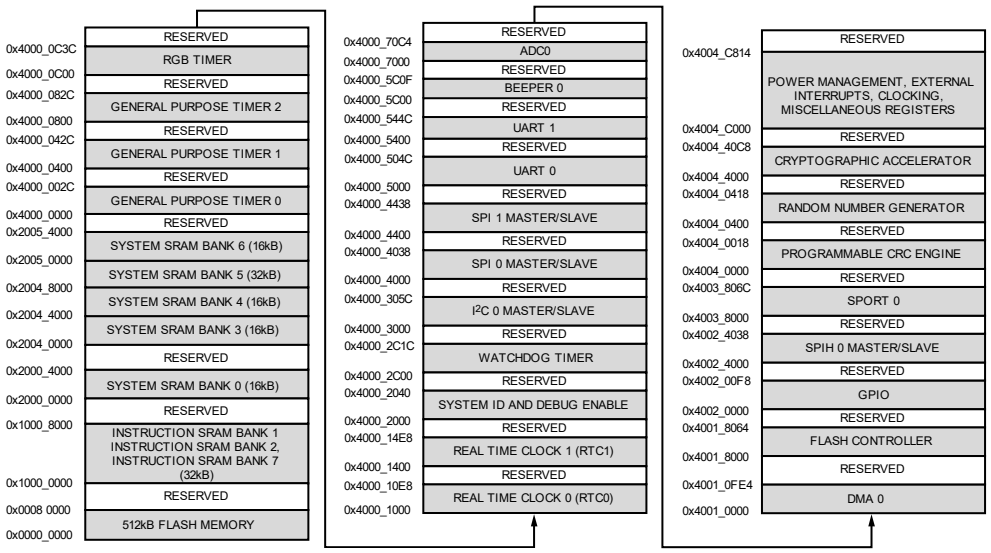


図 1-2：ADuCM4050 メモリ・マップ – SRAM モード 0

SRAM 領域

このメモリ・スペースには、リアルタイムでのアクセスが必要なアプリケーション命令とリテラル（定数）データが含まれています。ARM Cortex-M4F コアによる読出し／書込みアクセスとシステム・ペリフェラルによる読出し／書込み DMA アクセスをサポートしています。SRAM は 96KB のデータ SRAM と 32KB の命令 SRAM に分けられています。命令 SRAM がイネーブルでない場合は、関連付けられた 32KB をデータ SRAM としてマップできるので、結果としてデータ SRAM の容量が 128KB になります。

内部 SRAM データ領域

このスペースには、読出し／書込みデータを格納できます。内部 SRAM は、32KB のブロック内でコードとデータ（M4F スペースの SRAM 領域）にパーティション化できます。この領域へのアクセスは、待機状態なしのコア・クロック速度で実行されます。また、Cortex-M4F コアによる読出し／書込みアクセスとシステム・デバイスによる読出し／書込み DMA アクセスもサポートしています。

図には、ユーザ選択可能な様々な構成に対する SRAM のアドレス・マップを示します。構成の詳細については、スタティック・ランダム・アクセス・メモリ（SRAM）を参照してください。

INI ADDR	END ADDR	SRAM BANK	MODE0 ISRAM Enabled Cache OFF 32 KB ISRAM 96 KB DSRAM	MODE1 ISRAM Enabled 4 KB Cache 28 KB ISRAM 96 KB DSRAM	MODE2 ISRAM Disabled Cache OFF 0 KB ISRAM 128 KB DSRAM	MODE3 ISRAM Disabled 4 KB Cache 0 KB ISRAM 124 KB DSRAM
0x0000_0000	0x0007_FFFF		512 KB Flash	512 KB Flash	512 KB Flash	512 KB Flash
0x1000_0000	0x1000_2FFF	Bank1	12 KB ISRAM	12 KB ISRAM		
0x1000_3000	0x1000_6FFF	Bank7	16 KB ISRAM	16 KB ISRAM		
0x1000_7000	0x1000_7FFF	Bank2	4 KB ISRAM			
0x2000_0000	0x2000_3FFF	Bank0	16 KB DSRAM	16 KB DSRAM	16 KB DSRAM	16 KB DSRAM
0x2000_4000	0x2000_6FFF	Bank1			12 KB DSRAM	12 KB DSRAM
0x2000_7000	0x2000_7FFF	Bank2			4 KB DSRAM	
0x2004_0000	0x2004_3FFF	Bank3	16 KB DSRAM	16 KB DSRAM	16 KB DSRAM	16 KB DSRAM
0x2004_4000	0x2004_7FFF	Bank4	16 KB DSRAM	16 KB DSRAM	16 KB DSRAM	16 KB DSRAM
0x2004_8000	0x2004_FFFF	Bank5	32 KB DSRAM	32 KB DSRAM	32 KB DSRAM	32 KB DSRAM
0x2005_0000	0x2005_3FFF	Bank6	16 KB DSRAM	16 KB DSRAM	16 KB DSRAM	16 KB DSRAM
0x2005_4000	0x2005_7FFF	Bank7			16 KB DSRAM	16 KB DSRAM

Always Retained
Retained if SRAM_RET.RET1 is set
Retained if SRAM_RET.RET2 is set
Retained if SRAM_RET.RET3 is set
Retained if SRAM_RET.RET4 is set
Not Retained
Not Implemented

図 1-3：選択可能な構成

システム領域

ARM Cortex-M4F プロセッサはそのシステム・インターフェースでシステム領域にアクセスし、Cortex- M4F プラットフォーム内で処理されます。

CoreSight™ ROM：ROM のテーブル・エントリは、プロセッサのデバッグ・コンポーネントを指します。

ARM APB ペリフェラル：このスペースは ARM によって定義され、システム領域の下位 256KB を占有します。このスペースは、ARM コアの内部ペリフェラル（SCS、NVIC、WIC）と CoreSight ROM に対する M4F コアの読出し／書込みアクセスをサポートしています。システム DMA からはアクセスできません。

プラットフォーム・コントロール・レジスタ：このスペースには、Cortex-M4F プラットフォーム・コンポーネント内にレジスタがあり、ARM コア、そのメモリ、およびコード・キャッシュを制御します。M4F コアによってアクセス可能です（システム DMA によるアクセスはできません）。

フラッシュ・コントローラ

ADuCM4050 MCU は、フラッシュ・コントローラを使用してアクセス可能な 512KB の組み込みフラッシュ・メモリを備えています。フラッシュ・コントローラは、キャッシュ・コントローラと組み合わせることができます。コードの性能を最適化するため、フラッシュ・コントローラにはプリフェッチ機構が実装されています。フラッシュ書込みは、メモリ・マップド・レジスタへの APB 書込みを介したキーホール機構でサポートされます。フラッシュ・コントローラは、DMA ベースのキーホール書込みをサポートしています。

フラッシュ・コントローラは以下をサポートしています。

- 大容量消去やページ消去などの保護コマンドを実行するために必要な固定のユーザ・キー。
- オプションのユーザ定義可能なユーザ故障分析キー（FAA キー）。アナログ・デバイセズ担当者には、故障分析の実行中にこのキーが必要になります。
- アクセス可能なメモリに対するユーザ定義可能な書込み保護（オプション）
- 8 ビットの誤り訂正符号（ECC）。

キャッシュ・コントローラ

ADuCM4050 MCU には、オプションの 4KB の命令キャッシュがあります。特定のアプリケーションでは、キャッシュをイネーブルにしてコードを実行すると、フラッシュから直接操作する場合よりも消費電力が低くなる場合があります。キャッシュ・コントローラをイネーブルにすると、4KB の命令 SRAM がキャッシュ・メモリとして予約されます。休止モードでは、キャッシュ・メモリは保持されません。

ARM Cortex-M4F メモリ・サブシステム

ADuCM4050 MCU のメモリ・マップは、ARM の Cortex-M4F モデルをベースにしています。標準化されたメモリ・マップを保持することで、M4F プラットフォーム間で簡単にアプリケーションを移植できます。ADuCM4050 のアプリケーション開発は、通常、コード／SRAM 領域全体にわたるメモリ・ブロックをベースにしています。内部 SRAM と内部フラッシュによって、十分な容量の内部メモリを使用できます。

ブート

MCU は次のブート・モードをサポートしています。

- 内蔵フラッシュからのブート
- UART ダウンロードを介したソフトウェアのアップグレード

SYS_BMODE0 ピン（GPIO17）がパワーアップまたはハード・リセットの最中にローになると、MCU はシリアル・ダウンロード・モードに移行します。このモードでは、オンチップ・ローダ・ルーチンがカーネル内で始動し、UART ポートを構成してホストとの通信を行い、特定のシリアル・ダウンロード・プロトコル経由でファームウェアのアップグレードを管理します。

表 1-1：ブート・モード

ブート・モード	説明
0	UART ダウンロード・モード。
1	フラッシュ・ブート。内蔵フラッシュ・メモリからのブート。

セキュリティ機能

ADuCM4050 MCU は、ハードウェアとソフトウェアの保護機構を組み合わせて、セキュア・モードでデバイスへのアクセスをロックアウトし、オープン・モードでアクセスを許可します。これらの機構には、パスワードで保護されたスレーブ・ブート・モード（UART）とパスワードで保護された SWD インターフェースがあります。ブート中、システムは内蔵の発振器でクロックされます。リセットすると情報エリアのハードウェア・チェックサムが計算され、このチェックサムが正常な場合、CPU がフラッシュの情報エリアでアナログ・デバイセズのブート・ローダを実行できるようになります。アナログ・デバイセズのブート・ローダは、ユーザ・コードまたは UART ダウンロードが実行されたかどうかを特定する GPIO ブート・ピン（GPIO17）を検査します。

未許可のユーザによってデバイスの内容（フラッシュ、SRAM、CPU レジスタ、ペリフェラル・レジスタ）が外部インターフェースを介して読み出されないように保護する機構が備わっています。これは読出し保護と呼ばれます。

デバイスを、不正なコードによって回路内で再プログラムされないように保護できます。これは回路内の書込み保護と呼ばれます。

デバイスは、保護なし、読出し保護、読出し／回路内の書込み保護のいずれかで構成できます。読出し保護がない場合、回路内の書込み保護を実装する必要はありません。

安全機能

ADuCM4050 MCU には、特定レベルのシステムの安全性と信頼性を実現するための機能が備わっています。安全レベルは主にシステムの考慮事項によって決定されますが、堅牢性を高めるために次の機能が用意されています。

マルチパリティ・ビットで保護された SRAM

MCU の SRAM とキャッシュ・メモリ・スペースでは、各ワードがマルチパリティ・ビットで保護され、ランダムなソフト・エラーを検出できます。

プログラマブル GPIO

ADuCM4050 MCU には、LFCSP パッケージに 44 本、WLCSP パッケージに 51 本の GPIO ピンがあり、ユーザ・コードで定義可能な複数の機能を持っています。これらは入出力ピンとして構成でき、プログラマブルなプルアップ抵抗を備えています（SWD_CLK はプルダウン抵抗を備えています）。全ての入出力ピンは、電源範囲全体で機能します。

ディープ・スリープ・モードでは、GPIO ピンが状態を保持します。リセットで、トライステートになります。

タイマー

汎用タイマー

ADuCM4050 MCU には、3 個の同じ汎用タイマーが組み込まれ、各タイマーに 16 ビットのアップ・カウンタまたはダウン・カウンタが搭載されています。アップ／ダウン・カウンタは、ユーザが選択可能な 4 つのクロック源のいずれかからクロックできます。選択されたクロック源は、1、16、64、または 256 のプリスケーラを使用してスケール・ダウンできます。

ウォッチドッグ・タイマー（WDT）

ウォッチドッグ・タイマーは、プログラマブル・プリスケーラを備えた 16 ビットのカウント・ダウン・タイマーです。プリスケーラ源は選択可能で、1、16、または 256 の係数でスケール化できます。ウォッチドッグ・タイマーは、32kHz のオンチップ発振器（LFOSC）でクロックされ、無効なソフトウェア状態から回復するために使用されます。WDT には、リセットや MCU への割込みが強制的に行われないように、定期的なサービスが必要です。

RGB タイマー

ADuCM4050 MCU には 1 個の RGB タイマーがあり、アノード・コモンの RGB LED に対応します。1 個のタイマー・カウンタと 3 個のコンペア・レジスタが備わっています。3 つのポートまたはピンに 3 通りの異なる PWM 波形を同時に生成し、アノード・コモンの RGB LED で様々な色を実現できます。

RGB タイマーの動作時、他の 3 個の汎用タイマーはユーザ・ソフトウェアで使用可能です。

全てのタイマーにイベント・キャプチャ機能があり、40 通りの割込みを受けることができます。

パワー・マネージメント

ADuCM4050 MCU には、パワー・マネージメント機能とクロッキング機能があります。

電力モード

PMU は ADuCM4050 MCU の電力モードを制御し、ARM Cortex-M4F がクロックと電力ゲートを制御して、動作時の消費電力と休止時の電力を節約できるようにします。

複数の電力モードを利用できます。各モードは、対応する機能を削減することで、低消費電力のメリットを発揮します。

- **アクティブ・モード：**全てのペリフェラルが有効です。最適化されたクロック管理によって有効電力が管理されます。
- **Flexi モード：**コアはクロック・ゲーティングされますが、システムの残りの部分はアクティブです。このモードでは命令を実行できませんが、DMA 転送はペリフェラルとメモリ間で続行できます。
- **休止モード：**このモードでは、構成可能な SRAM とポート・ピンの保持、限られた数のウェイクアップ割込み、アクティブな RTC（任意指定）が行われます。
- **シャットダウン・モード：**このモードは強力なディープ・スリープ・モードで、全てのデジタル回路とアナログ回路がパワーダウンし、4 通りのウェイクアップ・ソースからウェイクアップするオプションを備えています。このモードでは、RTC は（オプションで）イネーブルになり、デバイスを RTC 割込みによって定期的にウェイクアップできます。
- **シャットダウン・モード – 高速ウェイクアップ：**この機能はシャットダウン・モードと同じですが、ウェイクアップ時間が短縮されます（ただし消費電力は増加します）。

クロッキング

ADuCM4050 MCU には、次のクロック・オプションがあります。

26MHz

- 内部発振器：HFOSC（26MHz）
- 外部水晶発振器：HFXTAL（26MHz または 16MHz）
- GPIO クロック入力：SYS_CLK_IN

32kHz

- 内部発振器：LFOSC
- 外部水晶発振器：LFXTAL

クロック不良検出

LFOSC クロックは、アクティブ、Flexi、休止の各モードで常に LFXTAL を監視しています。LFXTAL が動作を停止すると、割込みを検出・生成したり、ソフトウェアの介入なしに自動的に LFOSC に切り替えたりするオプションがあります。HFOSC クロックは、HFXTAL、GPIO CLK、PLL CLK を監視します。これらのクロックのいずれかをシステム・クロックとして使用し、トグルに失敗した場合、そのクロックは割込みによって検出できます。更に、自動的に HFPSC に切り替わるオプションもあります。

リアルタイム・クロック（RTC）

ADuCM4050 MCU には、RTC0 と RTC1（FLEX_RTC）の 2 つのリアルタイム・クロック・ブロックがあります。

クロック・ブロックは、32,768Hz の外部水晶発振器または LFOSC と組み合わせて動作する低消費電力の水晶発振器回路を共有します。

RTC には、プログラムされたアラート値が RTC カウントと一致するとコアに割込みを実行するアラームがあります。ソフトウェアによって RTC のイネーブルと構成が行われ、カウントが時刻に補正されます。

また、RTC には、固定された間隔で RTC カウントに正または負の調整を適用するデジタル・トリム機能があります。

FLEX_RTC は、SensorStrobe™ ピン（SSx）を介して SensorStrobe 機構で使用できます。この機構を使用することで、シャットダウン・モードを除く全ての電力モードで ADuCM4050 MCU をプログラマブル・クロック・ジェネレータとして使用できます。この方法では、外部センサーのタイミング領域を ADuCM4050 MCU で管理できます。SensorStrobe の出力が、FLEX_RTC からのプログラマブル分周器となるためです。FLEX_RTC は最大 30.7µs の分解能で動作します。センサーとマイクロコントローラは同期しているので、データを再度サンプリングして時間を合わせる必要はありません。

この機構がない場合、次のようになります。

- 外部センサーは、RC 発振器を使用します。MCU は、サンプリングしたデータを使用する前に、MCU の時間領域で再サンプリングする必要があります。

  または、

- MCU の消費電力が高いまま、センサー側で各データ変換を駆動します。

この機構では、ADuCM4050 MCU が長期間にわたり休止モードに置かれ、不要なデータ処理を回避して製品のバッテリ寿命を延ばします。

次表に RTC0 と RTC1 の主な違いを示します。

表 1-2：RTC 機能

機能	RTC0	RTC1（FLEX_RTC）
時間ベースの分解能（プリスケーラ機能）	1Hz で時間（秒単位）をカウントします。操作上、RTC0 は必ず 1Hz にプリスケール（例えば、32,768 で除算）し、常に実際の時間を秒単位でカウントします。	0～15 の任意の数を指数とする 2 のべき乗でクロックをプリスケールできます。これら 16 通りのプリスケール設定の単位で時間をカウントします。例えば、クロックは、1、2、4、8、…、16384、32768 でプリスケールできます。
ウェイクアップ・タイム	時間は秒単位で指定されます。	最小 30.7µs の分解能のアラーム時間をサポートします。つまり、時間は固有の 32kHz のクロック・サイクルまで仕様規定されます。
アラーム数	1 アラームのみ。1 秒単位で指定された絶対的で繰り返しのないアラーム時間を使用します。	2 アラーム。1 つは絶対的なアラーム時間、他の 1 つはプリスケールされた時間の 60 単位ごとに繰り返す周期的アラーム。
SensorStrobe 機構	該当なし。	デューティ・サイクルと周波数（0.5Hz～16 kHz）を微調整した 4 つの独立した SensorStrobe チャンネル。SensorStrobe は RTC のアラーム機構で、出力パルスを SSx ピン経由で外部デバイスに送信し、そのデバイスに特定の時間に計測を行うなどのアクションを実行するよう命令します。SensorStrobe イベントは、RTC のリアルタイム・カウントを基準とする特定のターゲット時間にスケジュール化されています。この機能は、アクティブ、flexi、休止の各モードでイネーブルできます。
入力キャプチャ	該当なし。	入力キャプチャは、外部デバイスが ADuCM4050 MCU の GPIO 入力の 1 つに遷移を介してイベントを送信した場合に、RTC のリアルタイム・カウントのスナップショットを取得します。通常、入力キャプチャ・イベントはそのようなデバイスの自律的な測定またはアクションによってトリガされます。その後、デバイスは、RTC がイベントに対応する時間のスナップショットを取得するよう、ADuCM4050 MCU に信号を送信します。このスナップショットを取得することで、ADuCM4050 MCU がウェイクアップし、CPU への割込みが発生します。続いて、CPU は、入力キャプチャ・イベントの発生サイクルと同一の 32kHz サイクルで、RTC から情報を取得します。
入力サンプリング	該当なし。	各 SensorStrobe チャンネルには、最大 3 つの独立した外部デバイス用 GPIO 入力があり、外部デバイスに送信された出力パルスに基づいてサンプリングが可能です。各 SensorStrobe チャンネルは、外部デバイスからこれらの GPIO 入力に何らかの動きが発生した場合に、ADuCM4050 MCU に割込みをするよう設定できます。これらの入力は、FIFO がフルになった、スイッチがオープンになった、閾値を超えた、などのセンサーの状態を送信できます。

表 1-2：RTC 機能（続き）

機能	RTC0	RTC1（FLEX_RTC）
		この機能によって、MCU は休止モードを維持し、外部デバイスからのプログラムされた特定シーケンスが検出された場合のみウェイクアップしてデータを処理できるようになります。

システム・デバッグ

ADuCM4050 MCU は、シリアル・ワイヤ・デバッグと単線ビューア・ポートを使用したトレースをサポートしています。

ビーパ・ドライバ

ADuCM4050 MCU には、ビーパ用のオーディオ・ドライバが内蔵されています。

ADuCM4050 MCU のビーパ・ドライバ・モジュールは、プログラマブルな周波数で差動方形波を生成します。差動方形波の出力に接続された 2 つの端子を持つ外部圧電音声コンポーネントを駆動します。ビーパ・ドライバは、8kHz～0.25kHz の範囲の周波数を出力するモジュールで構成されています。システム・クロックの変化による影響を受けない独立した 32kHz（32,768Hz）の固定クロック源で動作します。

ビーパ・ドライバにより、トーンの持続時間を 4ms～1.02 秒の範囲で 4ms ごとに設定可能です。シングルトーン（パルス）モードとマルチトーン（シーケンス）モードで多様な再生オプションを提供します。

シーケンス・モードでは、1～254（2～508 トーン）の任意の数のトーン・ペアを再生するか、（ユーザが停止するまで）永久に再生し続けるようにプログラムできます。ビープの開始／終了、シーケンスの終了、またはシーケンスがまもなく完了することを示すために、割込みを使用できます。

暗号化アクセラレータ（Crypto）

Crypto は、32 ビット APB DMA 対応のペリフェラルです。データの入出力用に 128 ビット・バッファを 2 つ備えています。これらのバッファは、4 回のデータ・アクセスで読込みと読出しが行われます。次のモードのように、ビッグエンディアン形式とリトルエンディアン形式がサポートされています。

- 電子コード・ブロック（ECB）モード — Advanced Encryption Standard（AES）モード
- カウンタ（CTR）モード
- 暗号ブロック・チェーン（CBC）モード
- メッセージ認証コード（MAC）モード
- 暗号ブロック・チェーン・メッセージ認証コード（CCM/CCM*）モード
- SHA-256 モード
- 鍵のラップ／アンラップで保護された鍵の保存
- HMAC 署名生成

CRC アクセラレータ

CRC アクセラレータを使用して、メモリ・ロケーションのブロックについて CRC を計算できます。正確なメモリ・ロケーションは、SRAM、フラッシュ、任意の組み合わせのメモリ・マップド・レジスタにあります。CRC アクセラレータは、予想されるシグネチャと比較するために使用できるチェックサムを生成します。

CRC の機能は次のとおりです。

- データ・ブロックの CRC シグネチャを生成します
- 最大 32 ビット長のプログラマブル多項式をサポートします
- 32 ビットのデータについて同時に動作し、任意のデータ長について CRC を生成します
- MSB ファーストと LSB ファーストの CRC 実装をサポートします
- 多様なデータ・ミラーリング機能
- ユーザによってプログラム可能な初期シード
- MCU をオフロードするためのデータ転送に使用する DMA コントローラ（メモリからメモリへの転送）

真性乱数ジェネレータ（TRNG）

TRNG は、確定的でない値が必要な動作で使用されます。セキュアな通信のためにチャレンジを生成したり、暗号化された通信チャンネルで使用するキーを生成したりできます。ジェネレータを複数回実行して、目的の動作の強度のために十分な数のビットを生成できます。TRNG を使用すれば、確定的なランダム・ビット・ジェネレータにシード値を提供できます。

シリアル・ポート（SPORT）

同期シリアル・ポートは、アナログ・デバイセズのオーディオ・コーデック、ADC、DAC など、種々のデジタル／ミックスド・シグナルのペリフェラル・デバイスに対する低価格のインターフェースです。シリアル・ポートは、2 本のデータ・ライン、クロック、フレーム同期で構成されています。データ・ラインは、送信または受信向けにプログラムできます。各データ・ラインは専用の DMA チャンネルを備えています。シリアル・ポート・データは、専用の DMA チャンネル経由で自動的にオンチップ・メモリや外部メモリとの間で転送できます。フレーム同期とクロックは共有されます。一部の ADC や DAC では、変換プロセスに 2 つの制御信号が必要です。このようなデバイスとインターフェース接続するために外部信号（SPT_CNV）が用意されています。この信号を使用するには、タイマー・イネーブル・モードを有効にします。このモードでは、モジュール内部の PWM タイマーを使用してプログラマブルな SPT_CNV 信号を生成します。

SPORT は次の 3 つのモードで動作できます。

- 標準の DSP シリアル・モード
- タイマー・イネーブル・モード
- ゲーテッド・クロック・モード

シリアル・ペリフェラル・インターフェース（SPI）ポート

ADuCM4050 MCU には 3 つの SPI があります。SPI は、8 ビットのデータを同期転送して同時に受信できる、業界標準の全二重同期シリアル・インターフェースです。各 SPI には、DMA コントローラに接続する 2 つの DMA チャンネルが実装されています。一方の DMA チャンネルは送信に、他方のチャンネルは受信に使用されます。MCU に搭載された SPI は、外部シリアル・フラッシュ・デバイスに簡単に接続できます。

SPI で使用できる機能には、次のものがあります。

- シリアル・クロック位相モードとシリアル・クロック極性モード
- ループバック・モード
- 連続転送モード
- ワイヤード OR 出力モード
- 半二重動作用の読出しコマンド・モード（送信の後に受信が続く）
- フロー制御をサポート
- 複数の CS ラインをサポート
- CS ソフトウェア・オーバーライドをサポート
- 3 ピンの SPI をサポート

UART ポート

ADuCM4050 MCU には、PC 標準 UART と互換性のある全二重 UART ポートが 2 ポート搭載されています。

UART ポートは、他のペリフェラルやホストに対する簡略化された UART インターフェースとして機能し、全二重、DMA、シリアル・データの非同期転送をサポートします。UART ポートは、5～8 のデータ・ビット、パリティなし、偶数または奇数のパリティをサポートします。フレームは、1 ストップ・ビット、1. 5 ストップ・ビット、2 ストップ・ビットで終了します。

Inter-Integrated Circuit（I²C）

ADuCM4050 MCU は、I²C インターフェースを備えています。I²C バス・ペリフェラルにはデータ転送用に 2 つのピンがあります。SCL（シリアル・クロック・ピン）と SDA（シリアル・データ・ピン）です。これらのピンは、マルチマスタ・システムでアービトレーションが可能なワイヤード AND 形式で構成されています。マスタ・デバイスを構成して、シリアル・クロックを生成できます。

この周波数は、シリアル・クロック分周器レジスタでプログラミングします。マスタ・チャンネルは、高速モード（400kHz）または標準モード（100kHz）で動作するよう設定できます。

A/D コンバータ（ADC）サブシステム

ADuCM4050 MCU には、12 ビットの SAR ADC と最大 8 個の外部チャンネルが統合されています。変換は、シングル・モードまたは自動サイクル・モードで実行できます。シングル・モードでは、特定のチャンネルで変換を実行するように ADC を設定できます。自動サイクル・モードでは、複数のチャンネル間で変換を行うと共に、個別のチャン

ネル・レジスタのサンプリングと読出しについて MCU のオーバーヘッドを削減できます。また、ADC は専用チャンネルを使用した温度検出とバッテリ電圧の測定にも使用できます。ただし、温度検出とバッテリ監視は、自動サイクル・モードでは実行できません。

リファレンス設計

Circuits from the Lab®では、次の内容をご覧になれます。

- 様々な回路タイプとアプリケーション向けシグナル・チェーンのグラフィカルな回路ブロック図
- ガイドとアプリケーション情報を選択するための各チェーンの部品へのリンク
- ベスト・プラクティスの設計技術を適用するリファレンス設計

開発ツール

このハードウェア・リファレンス・マニュアルの他に、ADuCM4050 MCU の開発サポートとして、評価用ハードウェアと開発ソフトウェア・ツールがあります。

ハードウェア

EV-COG-AD4050LZ（64 ピン LFCSP パッケージ用）と EV-COG-AD4050WZ（72 ボール WLCSP 用）が、ADuCM4050 MCU を用いたセンサー構成のプロトタイプ化に使用できます。

ソフトウェア

EV-COG-AD4050LZ（64 ピン LFCSP パッケージ用）と EV-COG-AD4050WZ（72 ボール WLCSP 用）には、ADuCM4050 MCU の開発環境およびデバッグ環境が完備されています。ADuCM4050 MCU のボード・サポート・パッケージ（BSP）は、IAR Embedded Workbench for ARM、Keil™、CrossCore®Embedded Studio（CCES）環境に対応します。

BSP には、オペレーティング・システム（OS）対応のドライバと、デバイスに搭載された全てのペリフェラルに対応するサンプル・コードも含まれます。

ADuCM4050 のペリフェラル・メモリ・マップ

表 1-3：インスタンスの概要

Name	Module	Address
TMR0	General-Purpose Timer	0x40000000
TMR1	General-Purpose Timer	0x40000400
TMR2	General-Purpose Timer	0x40000800
TMR3	RGB Timer	0x40000C00
RTC0	Real-Time Clock	0x40001000
RTC1	Real-Time Clock	0x40001400
SYS	System	0x40002000

表 1-3：インスタンスの概要（続き）

Name	Module	Address
WDT0	Watchdog Timer	0x40002C00
I2C0	I2C Interface	0x40003000
SPI0	SPI Interface	0x40004000
SPI1	SPI Interface	0x40004400
UART0	UART Interface	0x40005000
UART1	UART Interface	0x40005400
BEEP0	Beeper	0x40005C00
ADC0	ADC	0x40007000
DMA0	DMA	0x40010000
FLCC0	Flash/Cache Controller	0x40018000
GPIO0	GPIO	0x40020000
GPIO1	GPIO	0x40020040
GPIO2	GPIO	0x40020080
SPI2	SPI	0x40024000
SPORT0	SPORT	0x40038000
CRC0	CRC Accelerator	0x40040000
RNG0	Random Number Generator	0x40040400
CRYPT0	Cryptographic Module	0x40044000
PMG0	PMG	0x4004C000
XINT0	External Interrupt Module	0x4004C080
CLKG0	Clocking Subsystem	0x4004C100

2 デバッグ（DBG）とトレース・インターフェース

ADuCM4050 MCU は、2 線式シリアル・ワイヤ・デバッグ（SWD）インターフェースと、単線ビューア・ポートを介したトレース機能をサポートしています。

デバッグとトレース機能

ADuCM4050 MCU は、単線ビューア・ポートを介して Cortex-M4F 向けのトレース機能をサポートしています。この MCU は、低価格のデバッグ、トレースとプロファイリング、ブレークポイント、ウォッチポイント、およびコード・パッチングに役立ついくつかのシステム・デバッグ・コンポーネントを搭載しています。

この製品は、ARM Cortex-M4F プロセッサの軽量実装を目指しています。

サポートされているシステム・デバッグ／トレース・コンポーネントは、次のとおりです。

- フラッシュ・パッチおよびブレークポイント（FPB）ユニットにより、2 つのブレークポイントを実現します。フラッシュ・パッチングはサポートされていません。
- データ・ウォッチポイントおよびトレース（DWT）ユニットにより、1 つのウォッチポイント、トリガ・リソース、およびシステム・プロファイリングを実現します。DWT は 1 つのコンパレータしか備えていないため、ウォッチポイントを発生させるためのデータ・マッチングをサポートしていません。
- 命令トレース・マクロセル（ITM）ユニットにより、オペレーティング・システムとアプリケーション・イベントをトレースする printf 方式のデバッグをサポートし、診断システム情報を生成します。ITM は、4 つの有効なソースの 1 つから得られるトレース・パケットを生成します。
- トレース・ポート・インターフェース・ユニット（TPIU）は、ITM からデータ・ストリームへのオンチップ・トレース・データ間のブリッジとして動作します。

DBG 機能の説明

このセクションでは、ADuCM4050 MCU で使用される SWD インターフェース機能について説明します。

シリアル・ワイヤ・インターフェース

シリアル・ワイヤ・インターフェースは、2 ピンで構成されています。

- SWCLK：SWCLK は、デバッグ・プローブによって駆動されます。
- SWDIO：SWDIO は、プロトコル・フェーズに応じてデバッグ・プローブまたはターゲットによって駆動できる双方向信号です。コンテンションを避けるため、データの方向が変わるたびに非駆動のアイドル・ターンアラウンド・サイクルがバスに挿入されます。

SWD プルアップ

SWDIO ピンには、プルアップ抵抗が必要です。このピンのデフォルト状態は、プルアップです。

SWCLK ピンは、規定された状態（ハイまたはロー）にプルアップまたはプルダウンする必要があります。このピンをローにプルダウンすることを推奨します。プルダウンがこのピンのデフォルト状態です。

SWD タイミング

ターゲットは、SWDIO データを SWCLK の立上がりエッジでサンプリングまたは駆動します。

最適なタイミングにするためには、デバッグ・プローブが SWDIO を SWCLK の立下がりエッジで駆動し、SWDIO を SWCLK の立上がりエッジでサンプリングすることを推奨します。

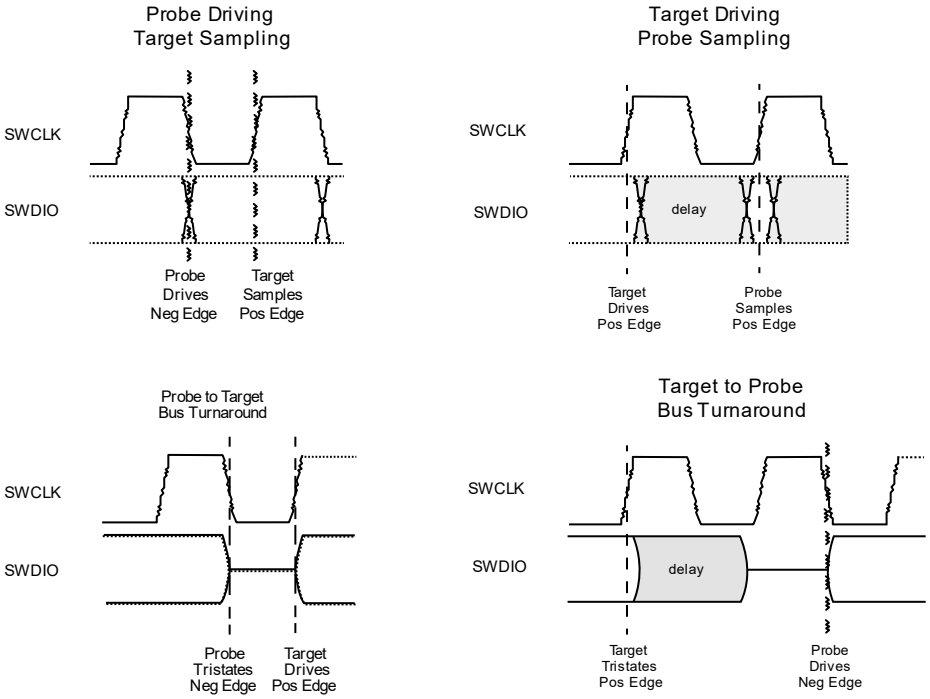


図 2-1：SWD タイミング

DBG 動作モード

シリアル・ワイヤ・プロトコル

シリアル・ワイヤ・プロトコルは、3 つのフェーズで構成されています。

- パケット・リクエスト
- アクノレッジ応答
- データ転送

デバッグ・プローブは、パケット・リクエストを常に送信します。

ターゲットは、アクノレッジ応答を常に送信します。

データ転送方向は、パケット・リクエストとアクノレッジ応答によって決まります。

パケット・リクエスト

デバッグ・プローブによって送信されるパケット・リクエストは 8 ビットであり、ターンアラウンド・ビットが常に続きます。

表 2-1：パケット・リクエスト

ビット	説明
Start	常時ハイ：ロジック 1
APnDP	アクセス・ポート：1 またはデータ・ポート：0
RnW	読出し：1 または書込み：0
A[2:3]	アドレスを LSB ファーストで送信
Parity	パリティ = APnDP + RnW + A[2] + A[3]
Stop	常時ロー：ロジック 0
Park	常時ハイ：ロジック 1
Turn	高インピーダンス／非駆動

アクノレッジ応答

ターゲットによって送信されるアクノレッジ応答は、3 ビットです。ターゲットが読出し（RnW = 1）のためにデータを送信する場合、アクノレッジの後にデータ転送が直接続き、ターンアラウンド・ビットはありません。デバッグ・プローブが書込み（RnW = 0）のためにデータを送信する場合、アクノレッジの後にターンアラウンド・ビットが続きます。

表 2-2：アクノレッジ応答

ビット	説明
ACK[0:2]	アクノレッジを LSB ファーストで送信 100：OK 010：ウェイト 001：フォールト

データ転送

データ転送フェーズは、LSB ファーストで送信される 32 ビットのデータ［0：31］で構成され、1 つのパリティ・ビットが続きます。データ・フィールド内の 1 のビット数が奇数の場合、パリティ・ビットには 1 が設定されます。

ターゲットが読出しのためにデータを送信する場合、データ転送の後にターンアラウンド・ビットが続きます。デバッグ・プローブが書込みのためにデータを送信する場合、ターンアラウンド・ビットはありません。

CTRL/STAT ORUNDETECT ビットが 1 に設定されている場合を除き、ウェイト／フォールト応答の場合は、データ転送フェーズが省略されます。ORUNDETECT ビットが 1 に設定されている場合、データ転送フェーズでは常に送信が継続され、ウェイト／フォールト応答が発生すると、STICKYORUN が 1 に設定されます。

プロトコル・フォーマット

この図は、プロトコル・フォーマットを示しています。

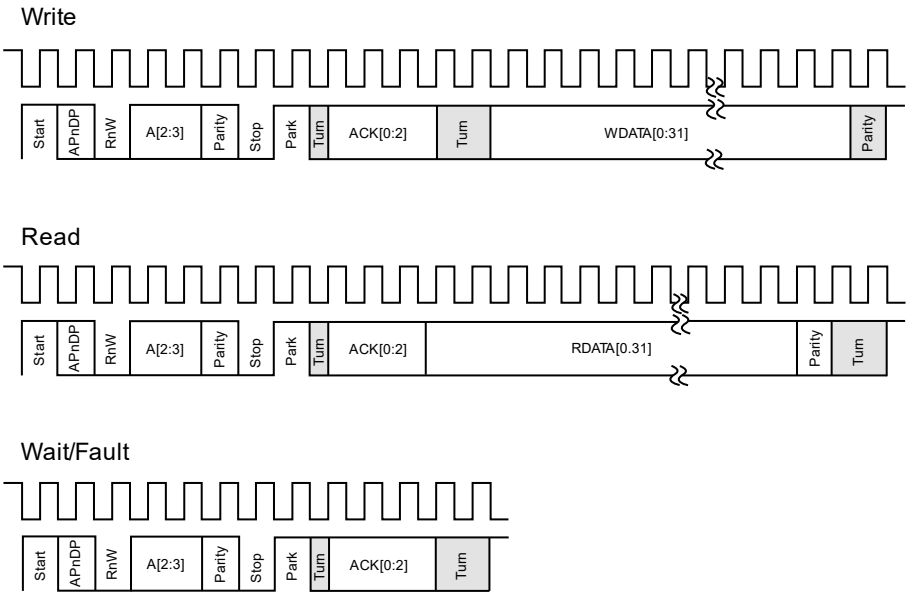


図 2-2：プロトコル・フォーマット

デバッグ・アクセス・ポート（DAP）

DAP は、2 つのポートに分かれています。

- デバッグ・ポート（DP）
- アクセス・ポート（AP）

デバッグ・ポートが 1 つあります。アクセス・ポートは複数にすることができます。

デバッグ・ポートには 2 ビット（4 ワード）のアドレス空間があり、システム・メモリのアドレス空間から分離されています。これらのレジスタは、シリアル・ワイヤ・デバッグ・インターフェースを通してのみアクセスできますが、CPU からはアクセスできません。

各アクセス・ポートには独立した 6 ビットのアドレス空間があるため、アクセス・ポートあたり最大 64 のワード・レジスタをアクセス可能です。このアドレス空間は、デバッグ・ポートとシステム CPU のアドレス空間から分離されており、デバッグ・ポートを介したシリアル・ワイヤ・インターフェースによってのみアクセス可能です。

このデバイスにはアクセス・ポートがあります。メモリ・アクセス・ポートは、`APSEL = 00` に設定することにより選択できます。

表 2-3：アクセス・ポート

APSEL [7:0]	アクセス･ポート
00	メモリ･アクセス･ポート

デバッグ・ポート

デバッグ・ポートには、4 つの 32 ビット・リード／ライト・レジスタが配置されています。これらのレジスタは、インターフェースを識別して設定したり、特定のアクセス・ポートを選択して通信したりするために使用できます。

表 2-4：デバッグ・ポート・レジスタ

Address [3:2]	Read	Write	DPBANKSEL
00	DPIDR	ABORT	X
01	CTRL/STAT		0x0
	DLCR		0x1
	TARGET ID		0x2
	DLPIDR		0x3
	EVENTSTAT		0x4
10	RESEND	SELECT	X
11	RDBUFF	TARGETSEL	X

ADuCM4050 SYS レジスタの説明

システム識別とデバッグ・イネーブル（SYS）は、次のレジスタに含まれています。

表 2-5：ADuCM4050 SYS レジスター一覧

レジスタ名	説明
SYS_ADIID	ADI 識別
SYS_CHIPID	チップ識別子
SYS_SWDEN	シリアル・ワイヤ・デバッグ・イネーブル

ADI 識別

ADI Cortex デバイス識別。

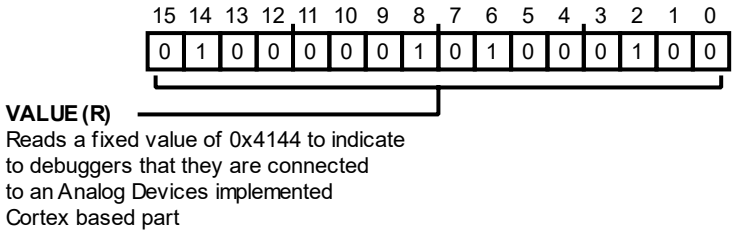


図 2-3：SYS_ADIID レジスタ図

表 2-6：SYS_ADIID レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	VALUE	固定値 0x4144 が読み出されてデバッガに表示されると、デバッガが Cortex ベースの部品を実装したアナログ・デバイセズ製品に接続されていることを示しています。

チップ識別子

チップ識別。

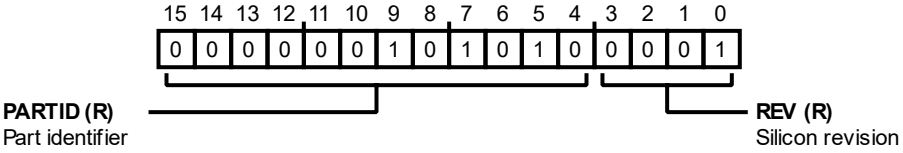


図 2-4：SYS_CHIPID レジスタ図

表 2-7：SYS_CHIPID レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15:4 (R/NW)	PARTID	部品識別子。	
3:0 (R/NW)	REV	シリコン・レビジョン。	
		0	シリコン・レビジョン

シリアル・ワイヤ・デバッグ・イネーブル

SYS_SWDEN レジスタは、シリアル・ワイヤ・デバッグ（SWD）インターフェースをイネーブルにするために使用されます。このレジスタは、内部 POR または外部ピン・リセットでリセットされます。ソフトウェア・リセットの影響は受けません。

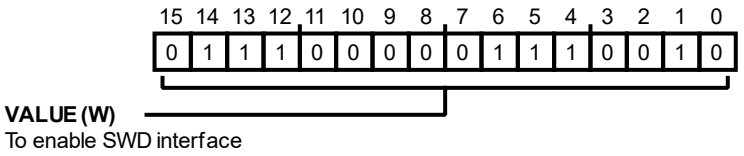


図 2-5：SYS_SWDEN レジスタ図

表 2-8：SYS_SWDEN レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (RX/W)	VALUE	SWD インターフェースをイネーブルにします。 SYS_SWDEN レジスタに"en"（0x6E65）または"EN"（0x4E45）を書き込むと、SWD インターフェースがイネーブルになります。SYS_SWDEN レジスタに"rp"（0x7072）または"RP"（0x5052）を書き込むと、SWD インターフェースがディスエーブルになります。その他の値を書き込むと無視されます。SYS_SWDEN レジスタは、一度"EN"または"RP"が書き込まれると変更することはできません。このレジスタは、POR リセットまたは PIN リセットでリセットされますが、ソフトウェア・リセットまたは WDT リセットではリセットされません。

3 イベント（割込みと例外）

ADuCM4050 MCU は多数のシステム例外とペリフェラル割込みをサポートしています。

イベント機能

ADuCM4050 のイベントには、以下のような機能があります。

- 15 のシステム例外（最も優先順位が高い）。
- 72 のシステム割込み（プログラマブルな優先順位）。
- 4 つの外部割込み（低消費電力モードからコアをウェイクアップするように個別に設定可能）。

イベントの割込みと例外

Cortex の例外

例外 1～15 はシステム例外です。

表 3-1：システム例外

例外番号	IRQ 番号	例外タイプ	優先度	説明
1	−	Reset	−3 (highest)	リセット
2	−14	NMI	−2	VREG 未満または VBAT（1.8V 未満）の組み合わせ（論理和）に連動するマスク不能割込み。
3	−13	Hard fault	−1	対応する故障ハンドラがイネーブルでない場合のすべての故障状態
4	−12	MemManagement fault	Programmable	メモリ管理故障。不正な場所へのアクセス
5	−11	Bus fault	Programmable	プリフェッチ故障、メモリ・アクセス故障、その他のアドレス／メモリ関連の故障
6	−10	Usage fault	Programmable	未定義命令実行や不正な状態遷移の試行などの例外
7 to 10	−	Reserved	Not applicable	

表 3-1：システム例外（続き）

例外番号	IRQ 番号	例外タイプ	優先度	説明
11	−5	SVCALL	Programmable	SVC 命令によるシステム・サービス・コール
12	−4	Debug monitor	Programmable	デバッグ・モニタ（ブレークポイント、ウォッチポイント、または外部デバッグ要求）
13	−	Reserved	Not applicable	
14	−2	PENDSV	Programmable	システム・サービスの保留可能な要求
15	−1	SYSTICK	Programmable	システム・チック・タイマー

ネスト型ベクタ割込みコントローラ

割込みはネスト型ベクタ割込みコントローラ（NVIC）によって制御され、8 つのレベルの優先度を使用できます。限られた数の割込みだけがデバイスを休止モードからウェイクアップさせることができます。デバイスが Flexi モード、休止モード、またはシャットダウン・モードからウェイクアップすると、デバイスは常にアクティブ・モードに戻ります。

表 3-2：割込み源

Exception Number	IRQ Number	Vector	Wake-up From		
			Flexi	Hibernate	Shutdown
16	0	Real Time Clock 1/Wakeup Timer/Hibernate RTC/ LFXTAL Clock Failure	Yes	Yes	No
17	1	External Interrupt 0	Yes	Yes	Yes
18	2	External Interrupt 1	Yes	Yes	Yes
19	3	External Interrupt 2	Yes	Yes	Yes
20	4	External Interrupt 3/ UART0_RX_Wakeup	Yes	Yes	No
21	5	Watchdog Timer	Yes	No	No
22	6	VREG Over	Yes	No	No
23	7	Battery Voltage Range	Yes	Yes	No
24	8	Real Time Clock 0	Yes	Yes	Yes
25	9	GPIO IntA	Yes	No	No
26	10	GPIO IntB	Yes	No	No
27	11	General Purpose Timer 0	Yes	No	No
28	12	General Purpose Timer 1	Yes	No	No
29	13	Flash Controller	Yes	No	No
30	14	UART0	Yes	No	No
31	15	SPI0	Yes*	No	No

表 3-2：割込み源（続き）

Exception Number	IRQ Number	Vector	Wake-up From		
			Flexi	Hibernate	Shutdown
32	16	SPI2	Yes*	No	No
33	17	I2C0 Slave	Yes*	No	No
34	18	I2C0 Master	Yes*	No	No
35	19	DMA Error	Yes	No	No
36	20	DMA Channel 0 Done	Yes	No	No
37	21	DMA Channel 1 Done	Yes	No	No
38	22	DMA Channel 2 Done	Yes	No	No
39	23	DMA Channel 3 Done	Yes	No	No
40	24	DMA Channel 4 Done	Yes	No	No
41	25	DMA Channel 5 Done	Yes	No	No
42	26	DMA Channel 6 Done	Yes	No	No
43	27	DMA Channel 7 Done	Yes	No	No
44	28	DMA Channel 8 Done	Yes	No	No
45	29	DMA Channel 9 Done	Yes	No	No
46	30	DMA Channel 10 Done	Yes	No	No
47	31	DMA Channel 11 Done	Yes	No	No
48	32	DMA Channel 12 Done	Yes	No	No
49	33	DMA Channel 13 Done	Yes	No	No
50	34	DMA Channel 14 Done	Yes	No	No
51	35	DMA Channel 15 Done	Yes	No	No
52	36	SPORT0A	Yes	No	No
53	37	SPORT0B	Yes	No	No
54	38	Crypto	Yes	No	No
55	39	DMA Channel 24 Done	Yes	No	No
56	40	General Purpose Timer 2	Yes	No	No
57	41	Crystal Oscillator	Yes	No	No
58	42	SPI1	Yes	No	No
59	43	PLL	Yes	No	No
60	44	Random Number Generator	Yes	No	No
61	45	Beeper	Yes	No	No

表 3-2：割込み源（続き）

Exception Number	IRQ Number	Vector	Wake-up From		
			Flexi	Hibernate	Shutdown
62	46	ADC	Yes	No	No
63-71	47-55	Reserved	-	-	-
72	56	DMA Channel 16 Done	Yes	No	No
73	57	DMA Channel 17 Done	Yes	No	No
74	58	DMA Channel 18 Done	Yes	No	No
75	59	DMA Channel 19 Done	Yes	No	No
76	60	DMA Channel 20 Done	Yes	No	No
77	61	DMA Channel 21 Done	Yes	No	No
78	62	DMA Channel 22 Done	Yes	No	No
79	63	DMA Channel 23 Done	Yes	No	No
80	64	Reserved	Yes	No	No
81	65	Reserved	Yes	No	No
82	66	UART1	Yes	No	No
83	67	DMA Channel 25 Done	Yes	No	No
84	68	DMA Channel 26 Done	Yes	No	No
85	69	Timer RGB	Yes	No	No
86	70	Reserved	Yes	No	No
87	71	Root Clock Failure	Yes	No	No

＊ 割込みを生成するには、対応する PCLK が必要です。

内部的には、プログラマブルな最高の優先度（0）は、リセット、NMI、ハード故障に次ぐ 4 番目の優先度として扱われます。すべてのプログラマブルな優先度のデフォルトの優先度は 0 であることに注意してください。

2 つ以上の割込みに同じ優先度が割り当てられている場合、ハードウェアの優先度（位置番号が低いほうが優先順位が高い）によって、MCU が割込みをアクティブにする順序が決まります。例えば、SPI0 と SPI1 がどちらも優先度レベル 1 の場合、SPI0 の優先順位のほうが高くなります。

例外および割込みの詳細については、**ARM Cortex-M4 テクニカルリファレンスマニュアル**の第 3.9 章（例外）および第 6 章（ネスト型ベクタ割込みコントローラ）を参照してください。

割込みレジスタの処理

割込み源の割込みサービス・ルーチン（ISR）では、通常、最初に割込み源がクリアされます。1 を書き込んでクリアする割込みの場合、割込み源をクリアするために割込みステータス・レジスタに 1 を書き込む必要があります。バスマトリックスの内部遅延のため、この書込みは書込み先に到達するまでに遅延することがあります。一方、ISR の実行が既に完了している場合、誤って前の割込みが 2 番目の割込みとみなされる可能性があります。したがって、必ず

割込み源がクリアされてから ISR を終了させることを推奨します。これは、ISR の終了時に割込みステータスを再度読み出すと実行できます。

外部割込みの設定

4 つの外部割込みが実装されており、SYS_WAKEx（x = 0、1、2、3）と識別されます。これらの外部割込みは、次の種類のイベントの任意の組み合わせを検出するように個別に設定できます。

- 立上がりエッジ：ロジックはローからハイへの遷移を検出し、パルスを生成します。立上がりエッジごとに 1 つのパルスのみが Cortex-M4F に送信されます。
- 立下がりエッジ：ロジックはハイからローへの遷移を検出し、パルスを生成します。立下がりエッジごとに 1 つのパルスのみが Cortex-M4F に送信されます。
- 立上がりエッジまたは立下がりエッジ：ロジックはローからハイへ、またはハイからローへの遷移を検出し、パルスを生成します。エッジごとに 1 つのパルスのみが Cortex-M4F に送信されます。
- ハイ・レベル：ロジックはハイ・レベルを検出します。適切な割込みがアサートされ、Cortex-M4F に送信されます。割込みラインは、外部ソースからアサート解除されるまでアサートを保持します。検出するには、少なくとも 1 コア・クロック・サイクルの間ハイ・レベルが維持される必要があります。
- ロー・レベル：ロジックはロー・レベルを検出します。適切な割込みがアサートされ、Cortex-M4F に送信されます。割込みラインは、外部ソースからアサート解除されるまでアサートを保持します。検出するには、少なくとも 1 コア・クロック・サイクルの間ロー・レベルが維持される必要があります。

外部割込み検出ユニット・ブロックは常にオンとなっていて、休止モード時に外部割込みによってデバイスをウェイクアップさせることができます。

4 つの外部割込みとは別に、UART0_RX ピンのアクティビティ（上記のイベントの組み合わせ）を設定して、Flexi モードまたは休止モードからデバイスをウェイクアップさせることもできます。

割込み番号 4 は外部割込み 3 と UART0_RX の両方のピンに割り当てられます。どちらのピンも、割込み番号 4 を生成するように設定できます。UART0_RX ピンに基づくウェイクアップを有効にするように `XINT_CFG0.UART_RX_EN` ビットを設定する必要があります。UART0_RX ピンのイベントにより割込み番号 4 がアサートされるたびに、`XINT_EXT_STAT.STAT_UART_RXWKUP` ビットがセットされます。

注：対応するパッドで外部割込みを使用するには、パッドを GPIO として構成し、対応する GPIO 入力イネーブルをハイに設定する必要があります。例えば、パッド P0_15 を外部割込みとして使用する場合は、`GP0CON［31:30］`を 2'b00、`GP0IE［15］`を 1'b1 に設定します。

ADuCM4050 XINT レジスタの説明

外部割込み設定（XINT）には以下のレジスタがあります。

表 3-3：ADuCM4050 XINT レジスター一覧

レジスタ名	説明
XINT_CFG0	外部割込みの設定

表 3-3：ADuCM4050 XINT レジスター一覧（続き）

レジスタ名	説明
XINT_CLR	外部割込みのクリア
XINT_EXT_STAT	外部ウェイクアップ割込みステータス
XINT_NMICLR	マスク不能割込みのクリア

外部割込みの設定

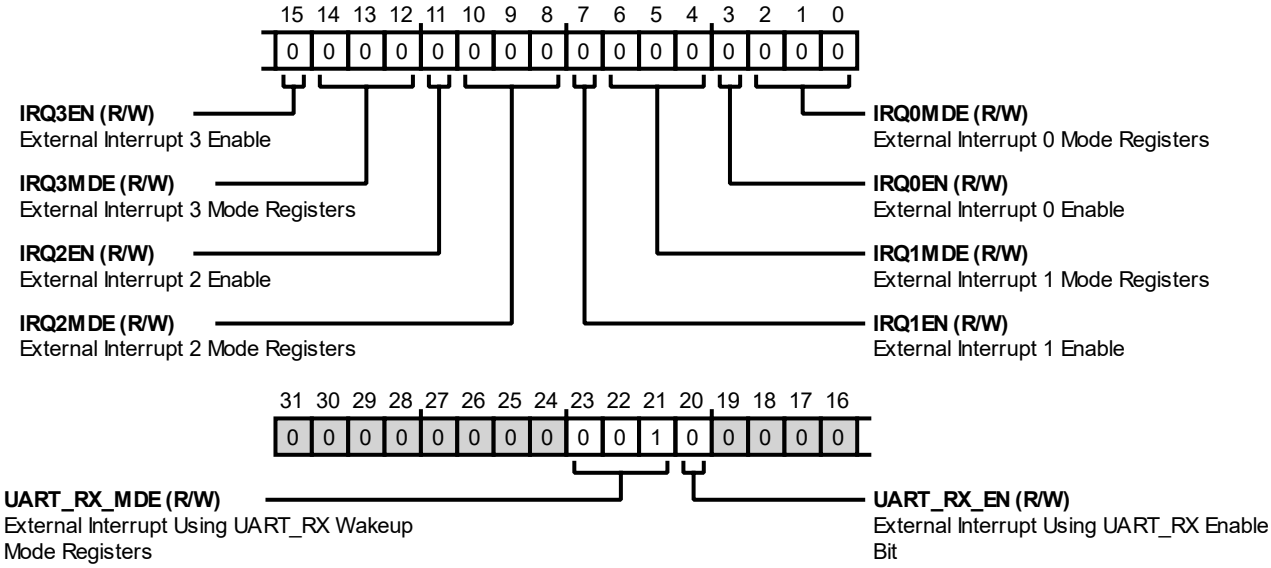


図 3-1：XINT_CFG0 レジスタ図

表 3-4：XINT_CFG0 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
23:21 (R/W)	UART_RX_MDE	UART_RX ウェイクアップ・モード・レジスタを使用した外部割込み。	
		0	立上がりエッジ
		1	立下がりエッジ
		2	立上がりエッジまたは立下がりエッジ
		3	ハイ・レベル
		4	ロー・レベル
		5	立下がりエッジ（001 と同じ）
		6	立上がりエッジまたは立下がりエッジ（010 と同じ）
		7	ハイ・レベル（011 と同じ）
20 (R/W)	UART_RX_EN	UART_RX イネーブル・ビットを使用した外部割込み。 このビットにより、'UART_RX'ピンは割込み（IRQ）番号 4 に割込みを生成します。イベント（割込みと例外）の章の割込み源の表を参照してください。 注：UART RX 割込みは外部割込み 3 との論理和がとられます。両方の割込みを個別に設定して、IRQ 番号 4 で割込みを生成することができます。IRQ 番号 4 での割込みの原因を区別するには、XINT_EXT_STAT レジスタを使用します	
		0	UART_RX ウェイクアップ割込みを無効にします
		1	UART_RX ウェイクアップ割込みを有効にします

表 3-4：XINT_CFG0 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15 (R/W)	IRQ3EN	外部割込み 3 の有効化／無効化。	
		0	外部割込み 3 は無効
		1	外部割込み 3 は有効
14:12 (R/W)	IRQ3MDE	外部割込み 3 のモード・レジスタ。	
		0	立上がりエッジ
		1	立下がりエッジ
		2	立上がりエッジまたは立下がりエッジ
		3	ハイ・レベル
		4	ロー・レベル
		5	立下がりエッジ（001 と同じ）
		6	立上がりエッジまたは立下がりエッジ（010 と同じ）
		7	ハイ・レベル（011 と同じ）
11 (R/W)	IRQ2EN	外部割込み 2 の有効化／無効化。	
		0	外部割込み 2 は無効
		1	外部割込み 2 は有効
10:8 (R/W)	IRQ2MDE	外部割込み 2 のモード・レジスタ。	
		0	立上がりエッジ
		1	立下がりエッジ
		2	立上がりエッジまたは立下がりエッジ
		3	ハイ・レベル
		4	ロー・レベル
		5	立下がりエッジ（001 と同じ）
		6	立上がりエッジまたは立下がりエッジ（010 と同じ）
		7	ハイ・レベル（011 と同じ）
7 (R/W)	IRQ1EN	外部割込み 1 の有効化／無効化。	
		0	外部割込み 0 は無効
		1	外部割込み 0 は有効
6:4 (R/W)	IRQ1MDE	外部割込み 1 のモード・レジスタ。	
		0	立上がりエッジ
		1	立下がりエッジ

表 3-4：XINT_CFG0 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
		2	立上がりエッジまたは立下がりエッジ
		3	ハイ・レベル
		4	ロー・レベル
		5	立下がりエッジ（001 と同じ）
		6	立上がりエッジまたは立下がりエッジ（010 と同じ）
		7	ハイ・レベル（011 と同じ）
3 (R/W)	IRQ0EN	外部割込み 0 の有効化／無効化。	
		0	外部割込み 0 は無効
		1	外部割込み 0 は有効
2:0 (R/W)	IRQ0MDE	外部割込み 0 のモード・レジスタ。	
		0	立上がりエッジ
		1	立下がりエッジ
		2	立上がりエッジまたは立下がりエッジ
		3	ハイ・レベル
		4	ロー・レベル
		5	立下がりエッジ（001 と同じ）
		6	立上がりエッジまたは立下がりエッジ（010 と同じ）
		7	ハイ・レベル（011 と同じ）

外部割込みのクリア

このレジスタには、対応する EXT_STAT ビットをクリアするのに使用される W1C ビットがあります。

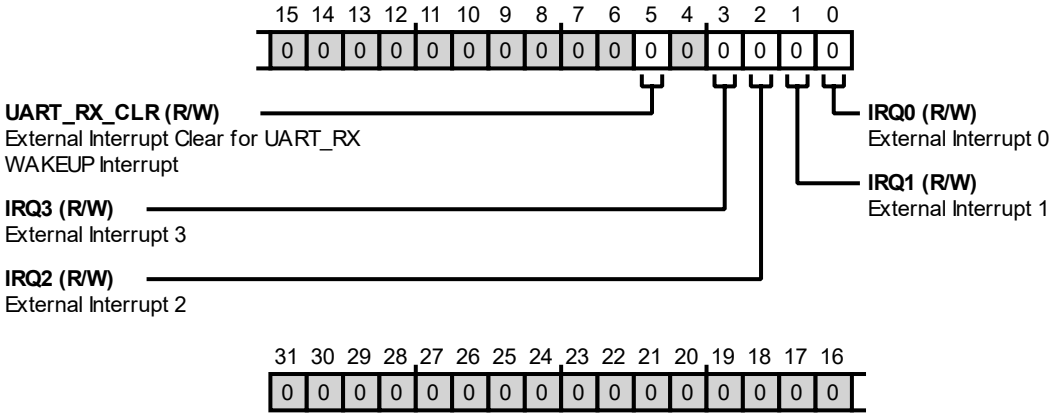


図 3-2：XINT_CLR レジスタ図

表 3-5：XINT_CLR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
5 (R/W)	UART_RX_CLR	UART_RX WAKEUP 割込みの外部割込みクリア。 割込みステータス・フラグをクリアするには 1 にセットします。ハードウェアによって自動的にクリアされます。
3 (R/W)	IRQ3	外部割込み 3。 割込みステータス・フラグをクリアするには 1 にセットします。ハードウェアによって自動的にクリアされます。
2 (R/W)	IRQ2	外部割込み 2。 割込みステータス・フラグをクリアするには 1 にセットします。ハードウェアによって自動的にクリアされます。
1 (R/W)	IRQ1	外部割込み 1。 割込みステータス・フラグをクリアするには 1 にセットします。ハードウェアによって自動的にクリアされます。
0 (R/W)	IRQ0	外部割込み 0。 割込みステータス・フラグをクリアするには 1 にセットします。ハードウェアによって自動的にクリアされます。

外部ウェイクアップ割込みステータス

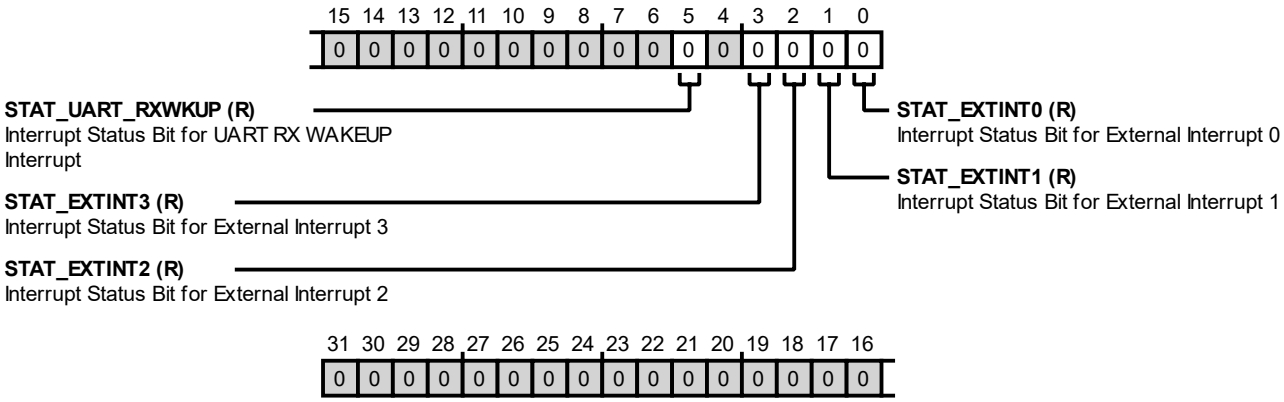


図 3-3：XINT_EXT_STAT レジスタ図

表 3-6：XINT_EXT_STAT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
5 (R/NW)	STAT_UART_RXWKUP	UART RX WAKEUP 割込みの割込みステータス・ビット。 このビットは、イベントの割込みと例外の章で説明したように、割込み番号 4 に割込みがアサートされた場合に有効になります。0 - UART_RX Wakeup は割込みを生成しませんでした。1 - UART_RX Wakeup は割込みを生成しました。これは読出し専用レジスタ・ビットで、EICLR.UART_RX_CLR ビットに 1 を書き込むことでクリアできます。 注：割込み番号 4 は外部割込み 3、UART RX ウェイクアップと共有されます。
3 (R/NW)	STAT_EXTINT3	外部割込み 3 の割込みステータス・ビット。 このビットは、イベントの割込みと例外の章で説明したように、割込み番号 4 に割込みがアサートされた場合に有効になります。0 - 外部割込み 3 を生成しませんでした。1 - 外部割込み 3 を生成しました。これは読出し専用レジスタ・ビットで、EICLR.IRQ3 ビットに 1 を書き込むことでクリアできます。
2 (R/NW)	STAT_EXTINT2	外部割込み 2 の割込みステータス・ビット。 このビットは、イベントの割込みと例外の章で説明したように、割込み番号 3 に割込みがアサートされた場合に有効になります。0 - 外部割込み 2 を生成しませんでした。1 - 外部割込み 2 を生成しました。これは読出し専用レジスタ・ビットで、EICLR.IRQ2 ビットに 1 を書き込むことでクリアできます。
1 (R/NW)	STAT_EXTINT1	外部割込み 1 の割込みステータス・ビット。 このビットは、イベントの割込みと例外の章で説明したように、割込み番号 2 に割込みがアサートされた場合に有効になります。0 - 外部割込み 1 を生成しませんでした。1 - 外部割込み 1 を生成しました。これは読出し専用レジスタ・ビットで、EICLR.IRQ1 ビットに 1 を書き込むことでクリアできます

表 3-6：XINT_EXT_STAT レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
0 (R/NW)	STAT_EXTINT0	外部割込み 0 の割込みステータス・ビット。 このビットは、イベントの割込みと例外の章で説明したように、割込み番号 1 に割込みがアサートされた場合に有効になります。0 - 外部割込み 0 を生成しませんでした。1 - 外部割込み 0 を生成しました。これは読出し専用レジスタ・ビットで、EICLR.IRQ0 ビットに 1 を書き込むことでクリアできます

マスク不能割込みのクリア

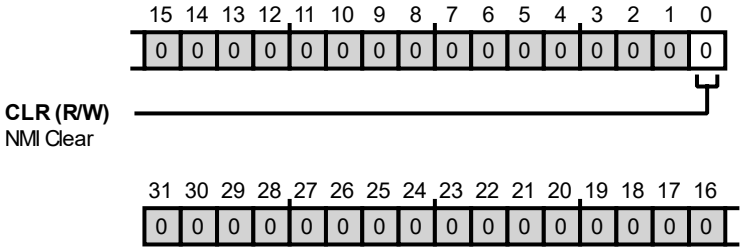


図 3-4：XINT_NMICLR レジスタ図

表 3-7：XINT_NMICLR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
0 (R/W)	CLR	NMI クリア。 NMI 割込みが設定されている場合、割込みステータス・フラグをクリアするには 1 にセットします。ハードウェアによって自動的にクリアされます。

4 パワー・マネージメント（PMG）

この章では、ADuCM4050 MCU で使用されるパワー・マネージメントおよび電力モードの機能アーキテクチャと実装について概説します。

パワー・マネージメント機能

この機能には次のものがあります。

- アクティブ・モードおよび flexi モード時に電力を低減する高効率降圧コンバータ。
- アクティブ・モード向けにカスタマイズされたクロック・ゲート。
- スリープ・モードでリークを減らす電力ゲート。
- 電圧モニタリング。
- スマート・ペリフェラルを備えた柔軟なスリープ・モード。
- 保持を伴うディープ・スリープ・モードと伴わないディープ・スリープ・モード。

パワー・マネージメント機能の説明

このセクションでは、ADuCM4050 MCU のパワー・マネージメント機能について説明します。システムには次の 2 つの電圧ドメインがあります。

- VBAT：1.74V～3.6V の範囲の入力電圧
- VREG：1.2V±10%の範囲の内部生成電圧

パワーアップ・シーケンス

バッテリ電圧が 1.6V を超えると、システムはパワーアップします。降圧コンバータはパワーアップ・シーケンス中はディスエーブルされるため、システムは LDO モードで起動します。必要に応じて、降圧モードに切り替えることができます。降圧モードでは、追加で 5 つのピン、2 個のフライング・コンデンサ、そして 1 個の負荷コンデンサを必要としますが、最小のダイナミック消費電力を提供します。外付け部品と消費電力のトレードオフは、ユーザが決めることができます。降圧モードは、`PMG_CTL1` レジスタのビット 0 に書き込むことによって有効にすることができます。

動作モード

ADuCM4050 MCU は、次の電力モードをサポートしています。

- アクティブ・モード：Cortex-M4F がフラッシュ／SRAM から実行されます。
- Flexi モード：Cortex-M4F はディスエーブルされます。イネーブルにするペリフェラルを選択します。DMA/SPI、DMA/I²C などがあります。
- 休止モード：システム電源が制御されます。16KB の SRAM は常に保持されます。これに加えて、最大 108KB の SRAM を選択して保持することができます。
- シャットダウン・モード：非保持状態。ピンは値を保持します。

アクティブ・モード

このモードでは、デバイスは動作しています。Cortex-M4F はフラッシュや SRAM からの命令を実行します。

このシステムでは、自動クロック・ゲート技術を使用してクロックをゲートするだけでなく、CLKG_CLK_CTL5 レジスタを使用してクロック・ゲートを選択することもできます。ほとんどのペリフェラルは、ディスエーブル時に自動的にクロック・ゲートされます。クロックは、ペリフェラルがイネーブルされているときにのみ動作します。例外として、I²C、UART、Timer_RGB、汎用タイマーがあります。これらのブロックは、CLKG_CLK_CTL5 レジスタを使用して手動でクロック・ゲートする必要があります。

CLKG_CLK_CTL5 レジスタのビットに 1 を書き込むと、ペリフェラルに対応するクロックが停止します。クロックが停止した後、ユーザ／ソフトウェアがそのペリフェラルのレジスタにアクセスすると、クロックは自動的にイネーブルされます。また、CLKG_CLK_CTL5 レジスタのビットに 0 を書き込むと、ペリフェラルに対応するクロックがイネーブルされます。

CLKG_CLK_CTL5.PERCLKOFF ビットを使用して、すべてのペリフェラルをディスエーブルできます。これは、Cortex-M4F プロセッサが SRAM またはフラッシュからのみ実行され、他のペリフェラルが必要ない場合に便利です。

CLKG_CLK_CTL5.PERCLKOFF ビットが 1 のとき、クロックはゲートされます。MCU または DMA がゲートされているペリフェラルのレジスタにアクセスすると、クロックは再びイネーブルされます。

デバイスは LDO でパワーアップされます。PMG_CTL1 レジスタに書き込むことで降圧モードを有効にして、消費電力を削減できます。

HPBUCK 負荷モードの選択

PMG_CTL1.HPBUCK_LD_MODE ビットを **HPBUCK 負荷モードの選択**の表に従って設定します。このビットは、ソフトウェア実行フローの開始時に設定します。

表 4-1：HPBUCK 負荷モードの選択

System Clock Frequency	HPBUCK Load Mode
≤ 26 MHz	0
> 26 MHz	1

Flexi モード

このモードでは、Cortex-M4F は常にディスエーブルされます。コアは、PMG_PWRMOD.MODE ビットを 0 にして WFI 命令を実行することにより、このモードに切り替えることができます。

CLKG_CLK_CTL5 レジスタのビットに 1 を書き込むと、ペリフェラルに対応するクロックが停止します。

クロックが停止した後、ユーザ／ソフトウェアがそのペリフェラルのレジスタにアクセスすると、クロックは自動的にイネーブルされます。また、CLKG_CLK_CTL5 レジスタのビットに 0 を書き込むと、ペリフェラルに対応するクロックがイネーブルされます。

ペリフェラルがイネーブルの場合、システム内の他のすべてのブロックのクロックが作動します。

このモードでは、Cortex の休止中にイネーブルされるブロックを柔軟に決めることができます。例えば、SPI または I^2C を介して DMA トランザクションを実行させたり、MCU の介入なしに DMA に ADC 変換を実行させることができます。このモードを使用すると、アクティビティ（センサーから特定の数のバイトを読み出すなど）を完了してからプロセッサがウェイクアップしてデータを処理することが予想される場合、消費電力を大幅に削減できます。

Flexi の低消費電力モード

SPORT や UART などのシリアル通信ペリフェラルが不要な場合は、低消費電力モードを使用して Flexi での消費電流を低減できます。この低消費電力モードでは、システム・クロック（Root Clk）を HFOSC から供給する必要があります（クロック設定）。

低消費電力モードの入力シーケンスは次のとおりです。

1. 分周された HFOSC 周波数が 1.2MHz 未満となるように、CLKG_CLK_CTL2.HFOSCDIVCLKSEL ビットを設定します。
2. PMG_CTL1.HPBUCK_LOWPWR_MODE ビットをイネーブルにします。

PMG_CTL1.HPBUCK_LOWPWR_MODE ビットがセットされている場合、負荷電流は最小になります。

したがって、このモードでは、以下のペリフェラルのみを動作させることができます。

1. GPT。GPT ブロックのソースは低周波数クロックでなければなりません。
2. ビーパ
3. GPIO
4. 外部割込み源

この機能は、Flexi モードからの高速ウェイクアップと組み合わせて使用できます。この場合、CLKG_CLK_CTL2.HFOSCAUTODIV_EN ビットを 1 に設定すると、高速ウェイクアップを有効にできます。Flexi モードからウェイクアップすると、HFOSC 分周クロックは自動的に 26MHz に戻ります。Flexi モードからウェイクアップした後で通常の動作を再開するには、PMG_CTL1.HPBUCK_LOWPWR_MODE ビットをクリアします。

Flexi からアクティブまでのウェイクアップ時間は、分周された HFOSC クロックの 1 クロック・サイクル分です。

休止モード

このモードでは、Cortex-M4F とすべてのデジタル・ペリフェラルがオフになります。合計 128KB SRAM のうち最大 124KB を休止モードで保持できます。

次の項目を選択できます。

1. 保持する SRAM。制御ビットを使用。これらの制御ビットは、3 つの 32KB 領域と 1 つの 12KB 領域を保持するのに使用できます。これは、常に休止モードで保持される 16 KB の SRAM に追加されます。

   SRAM 保持の制御ビットは次のとおりです。

   `PMG_SRAMRET.RET1`：バンク 1（12KB）の保持を有効化

   `PMG_SRAMRET.RET2`：バンク 3 とバンク 4（32KB）の保持を有効化

   `PMG_SRAMRET.RET3`：バンク 5（32KB）の保持を有効化

   `PMG_SRAMRET.RET4`：バンク 6 とバンク 7（32KB）の保持を有効化
2. 32kHz 水晶発振器を使用する RTC または内蔵 32kHz 発振器。水晶発振器のほうが高精度のクロックを提供します。ただし、消費電力は高くなります。
3. 休止モードでのバッテリ監視の有効化および設定。
4. 1.2V の安定化電源は常に監視され、電源が最小保持電圧を下回ってもデータが破壊されないことが確保されます。安定化電源が一定の閾値を下回ると、データが破損する前にチップがリセットされます。安定化電源は常に監視されますが、休止モードでバッテリを監視するオプションもあります。これは、`PMG_PWRMOD` レジスタを使用して行います。

メモリ設定

休止モードに入っている間、128KB のうち 124KB を保持できます。これは、`PMG_SRAMRET` レジスタで制御されます。

最低 16KB は、`PMG_SRAMRET` レジスタで設定された構成に関係なく、休止モードで必ず保持されます。

詳細については、SRAM 領域を参照してください。

SRAM 休止負荷モード

次の表に従って、`PMG_SRAMRET.HIBERNATE_SRAM_LOAD_MODE` ビットと `PMG_TRIM.HIBERNATE_LOAD_MODE` ビットを選択します。

`PMG_SRAMRET.HIBERNATE_SRAM_LOAD_MODE` ビットを 0 に設定すると、3 ミリ秒間待機してから休止モードに入ります。

表 4-2：SRAM 負荷モードの選択

Size of SRAM Retained in Hibernate	HIBERNATE_SRAM_LOAD_MODE	HIBERNATE_LOAD_MODE
16 KB	1	7
28 KB	1	7
48 KB	1	7
60 KB	1	7
80 KB	0	0
92 KB	0	0
112 KB	0	0
124 KB	0	0

PLL をソースとして使用する休止モード遷移

PMU はパワーダウン要求とパワーアップ要求を監視し、ルート・クロック・マルチプレクサをシームレスに切り替えることができます。これにより、安全な休止モードへのエントリと終了が確保されます。

PMU は、休止モード遷移時に以下の動作を自動的に実行します。

休止モードへのエントリ：

1. パワーダウン要求が検出されると、ルート・クロック・マルチプレクサを HFOSC に切り替えます。
2. PLL をディスエーブルしてパワーダウンします。

休止モードの終了：

1. パワーアップ要求が検出されると、PLL をイネーブルにします。

   PLL 入力が HFXTAL に設定されている場合、HFXTAL ロックが検出された後に PLL がイネーブルされます。
2. PLL のロックを待ちます。
3. ルート・クロック・マルチプレクサを PLL クロックに切り替えます。

注：休止モードからウェイクアップすると、コアは HFOSC で実行を開始します。PLL がロックされると、PMU はルート・クロック・マルチプレクサを PLL クロックに切り替えます。

シャットダウン・モード

これは、ウェイクアップ・コントローラとフェイルセーフを除くすべてのデジタル回路とアナログ回路がパワーダウンする最も強力なスリープ・モードです。これらの回路は VBAT から給電されます。LDO はオフになるため、1.2V の領域はシャットダウンされます。

デジタル・コアの状態は保持されないため、SRAM メモリの内容は保持されません。デバイスがシャットダウン・モードからウェイクアップすると、POR シーケンスに移行し、コードの先頭から実行が開始されます。PMG_SHDN_STAT

レジスタを読み出すことにより、このモードからのウェイクアップ・ソースを知ることができます。可能なウェイクアップ・ソースには次の 4 つがあります。

- 外部割込み。電力を節約するために 3 つの外部割込みに制限されています。
- 外部リセット。
- バッテリが 1.6V を下回る。
- RTC タイマー（このオプションが有効な場合）。水晶発振器でのみ使用できます。シャットダウン中は LF 発振器はオフにされます。

このモード時には、パッドの状態とウェイクアップ割込みの設定が保存されます。

パッドの構成は保存されますが、シャットダウン・モードからのウェイクアップ後にロックされます。`PMG_TST_CLR_LATCH_GPIOS` レジスタに値 0x58FA を書き込んでパッドの状態を解除する必要があります。ISR ルーチン内で書込みを実行することを推奨します。

シャットダウンには、HFOSC を使用した場合にのみ入ることができます。PLL、外部クロックまたは HF XTAL で動作している場合は、シャットダウン・モードにはなりません。シャットダウン・モードに入る前に、HFOSC に切り替える必要があります。

また、シャットダウン・モード中に利用できるスクラッチ・パッドがあります。スクラッチ・パッドは 32 ビットのレジスタで構成されています。スクラッチ・パッドの目的は、シャットダウン・モードですべての情報が失われる前に、3V でデータを保存することです。

`PMG_TST_SCRPAD_IMG` レジスタは常時、書込み可能です。これは読出し／書込みレジスタです。このレジスタに保存された情報は、シャットダウン・モードに入る前に VBAT 内の領域にコピーされます。VBAT のこの情報は、読出し専用レジスタである `PMG_TST_SCRPAD_3V_RD` レジスタを介してアクセスできます。

注：PLL、外部クロック、または HF XTAL で実行する場合は、シャットダウン・モードに入る前に HFOSC に切り替えてください。詳細については、シャットダウン・モードの設定を参照してください。

プログラミング・シーケンス

休止モードの設定

これは、状態を保持する消費電力が最小のパワーダウン・モードです。デジタル情報と SRAM 0 は常に保持されますが、SRAM 1、SRAM 3～SRAM 7 の保持はオプションです。SRAM 領域を参照してください。

休止モードで VBAT を監視することもオプションです。PMG_CTL1 レジスタは、VBAT 状態に関する情報を提供します。この制御レジスタを使用して割込みを生成することもできます。

休止モードに入るシーケンスは以下のとおりです。

1. `PMG_PWRMOD_MODE` = 2 を選択します。
2. Cortex-M4F で `SLEEPDEEP` ビットを選択します。
3. 希望のウェイクアップ割込みを有効にします。

4. バッテリ監視が必要な場合は、PMG_PWRMOD.MONVBATN ビットを 0 に設定します。
5. WFI/WFE 命令を実行します。

これでチップは休止モードになります。

ウェイクアップ割込みが到達すると、デバイスは休止モードを終了し、割込みルーチンを処理します。

シャットダウン・モードの設定

これは最も強力なパワーダウン・モードです。このモードでは状態情報は保持されず、ウェイクアップ後にチップがリセットから再起動します。

使用可能なウェイクアップ・ソースは、外部リセット、3 つのみに制限された外部割込み、水晶発振器を用いた RTC0（アプリケーションが周期的なウェイクアップを必要とする場合）です。内部低周波発振器はこのモードではディスエーブルされます。このモードで使用できるクロックは低周波水晶発振器（オプション）のみです。

このモードに入るシーケンスは以下のとおりです。

1. PMG_PWRMOD_MODE = 3 を選択します。
2. Cortex-M4F で SLEEPDEEP ビットをセットします。
3. 所望のウェイクアップ割込みを有効にします。
4. RTC0 からのウェイクアップが必要な場合は、LFXTAL をイネーブルにします。
5. WFI 命令を実行します。

これでチップはシャットダウン・モードに入ります。

注：PLL、外部クロック、または HFXTAL で実行する場合は、シャットダウン・モードに入る前に HFOSC に切り替えてください。

ウェイクアップ割込みが到達すると、デバイスはシャットダウン・モードを終了し、ウェイクアップして POR シナリオに進みます。このシーケンスは以下のとおりです。

1. PMG_SHDN_STAT レジスタを読み出します。
2. ウェイクアップ・イベントを生成した割込みルーチンを有効にします。
3. チップは有効な ISR にジャンプし、割込みルーチンを処理します。

注：シャットダウン・モードからのウェイクアップ後、ISR で PMG_PWRMOD を 0 に書き戻してください。

ウェイクアップ・シーケンス

ウェイクアップは割込みまたはリセットによってトリガされます。ウェイクアップのシーケンスは電力モードによって異なります。

ウェイクアップが割込みによってトリガされた場合、システムは最初に割込みルーチンを実行します。

ウェイクアップ・シーケンスはシャットダウン・モードでは異なります。情報は保持されないため、システムは常にウェイクアップ後にリセット状態から起動します。ウェイクアップ・シーケンスをトリガする割込みは、コードの再起動時に割込みルーチンが有効になるまで、Cortex-M4F で保留にされます。ステータス・レジスタは、シャットダウンからウェイクアップし、割込みルーチンをイネーブルにした後で読み出すことができます。

電力モードの割込みとウェイクアップ・ソースの詳細については、表 3-2 割込み源を参照してください。

シャットダウン・モードからの高速ウェイクアップ

このモードでは、デバイスはシャットダウン・モードより高速にウェイクアップします。

ウェイクアップ時間は、ウェイクアップ割込みのアサートからユーザ・コードの最初の命令の実行開始までの時間です。

この機能は、PMG_FAST_SHT_WAKEUP レジスタのビット 0 を設定すると有効になります。このレジスタ・フィールドの内容は、すべての電力モードで保持されます。このレジスタ・フィールドは、PMG_RST_STAT.EXTRST ビットまたは PMG_RST_STAT.PORSRC ビットがリセットされるとクリアされます。

電圧モニタ制御

電圧モニタ回路は常にイネーブルされ、VBAT と安定化電源が常に動作レベル内にあることが確保されます。これらの電源を監視する回路は、電源モニタ（PSM）と呼ばれます。

アクティブ・モードでの PSM の主な機能は次のとおりです。

- VBAT 電圧の監視
  - VBAT 電圧が 1.83V 未満の場合、MCU に対してアラームを生成する
  - VBAT 電圧が 1.6V 未満の場合、チップに対してリセットを生成する
  - VBAT ソースの状態を監視する

    PMG_CTL1.HPBUCK_LD_MODE ビットがローのとき、次の電圧範囲ごとに割込みを生成します。

    VBAT が 3.6V～2.8V

    VBAT が 2.8V～2.19V。割込み BR1 を生成

    VBAT が 2.19V～1.6V。割込み BR2 を生成

    PMG_CTL1.HPBUCK_LD_MODE ビットがハイのとき、次の電圧範囲ごとに割込みを生成します。

    VBAT が 3.6V～2.93V

    VBAT が 2.93V～2.39V。割込み BR1 を生成

    VBAT が 2.39V～1.6V。割込み BR2 を生成

- 安定化電源の監視
  - 安定化電源が 1.32V を超えると、割込みを生成（過電圧）
  - 安定化電源が 1.1V を下回ると、割込みを生成（低電圧）
  - 安定化電源が 1.08V 未満の場合にリセットを生成

以下は、休止モードでの PSM の主な機能です。

- VBAT 電圧の監視
  - 電圧が 1.83V を下回ると、MCU に対してアラームを生成（オプション）
  - 電圧が 1.6V を下回ると、チップに対してリセットを発生（オプション）。
  - VBAT 源の状態を監視（オプション）

    `PMG_CTL1.HPBUCK_LD_MODE` ビットがローのとき、次の電圧範囲ごとに割込みを生成します。

    VBAT が 3.6V～2.8V

    VBAT が 2.8V～2.19V。割込み BR1 を生成

    VBAT が 2.19V～1.6V。割込み BR2 を生成

    `PMG_CTL1.HPBUCK_LD_MODE` ビットがハイのとき、次の電圧範囲ごとに割込みを生成します。

    VBAT が 3.6V～2.93V

    VBAT が 2.93V～2.39V。割込み BR1 を生成

    VBAT が 2.39V～1.6V。割込み BR2 を生成
- 安定化電源の監視
  - 安定化電源が 1.08V を下回ると、チップに対してリセットを生成

VBAT および VREG の電源監視を以下に示します。

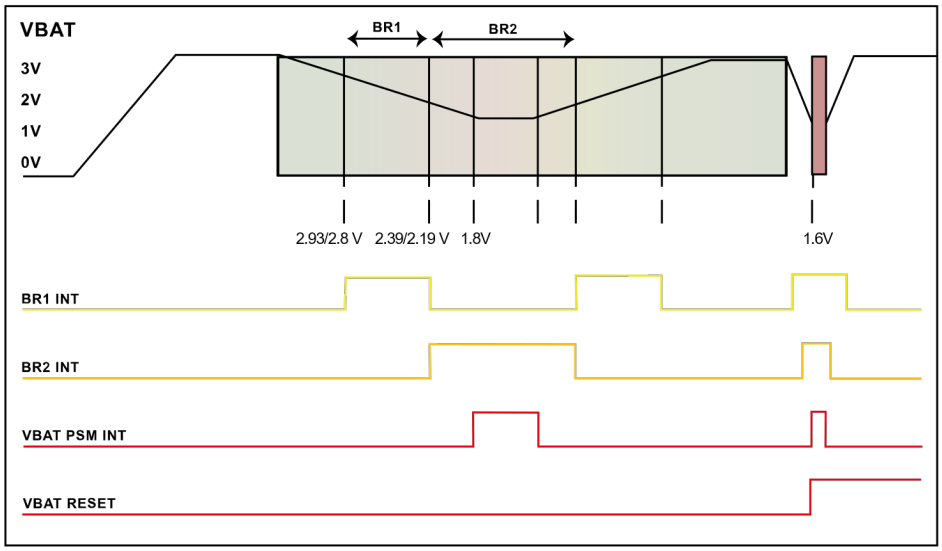


図 4-1：VBAT 電源モニタ

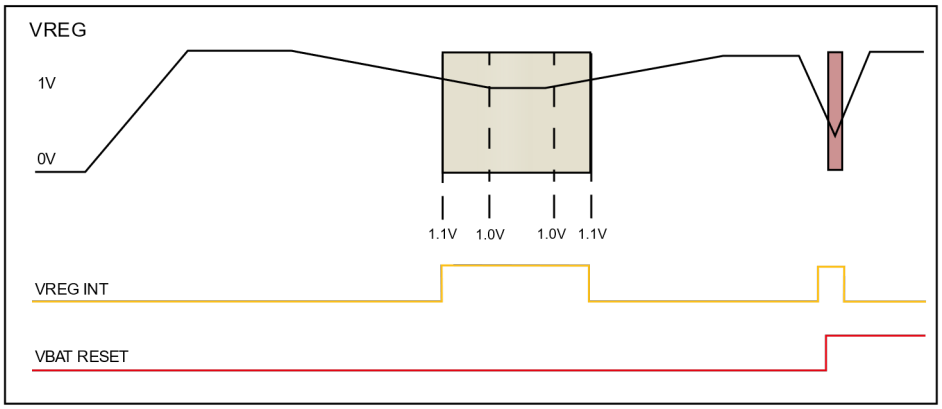


図 4-2：VREG 電源モニタ

保持レジスタの詳細

休止モードとシャットダウン・モードで保持される PMG レジスタを次の表に示します。

表 4-3：PMG 保持レジスタの詳細

Register Address	Register Name	Retention Status	
		Hibernate mode	Shutdown mode
0x4004C000	PMG_IEN	Yes	No
0x4004C004	PMG_PSM_STAT	Yes	No
0x4004C008	PMG_PWRMOD	Yes	No
0x4004C00C	PMG_PWRKEY	Yes	No

表 4-3：PMG 保持レジスタの詳細（続き）

Register Address	Register Name	Retention Status	
		Hibernate mode	Shutdown mode
0x4004C010	PMG_SHDN_STAT	Yes	No
0x4004C014	PMG_SRAMRET	Yes	No
0x4004C040	PMG_RST_STAT	Yes	No
0x4004C044	PMG_CTL1	Yes	No
0x4004C260	PMG_TST_SRAM_CTL.INSTREN	Yes	No
0x4004C260	PMG_TST_SRAM_CTL.PENBNK0	Yes	No
0x4004C260	PMG_TST_SRAM_CTL.PENBNK1	Yes	No
0x4004C260	PMG_TST_SRAM_CTL.PENBNK2	Yes	No
0x4004C260	PMG_TST_SRAM_CTL.PENBNK3	Yes	No
0x4004C260	PMG_TST_SRAM_CTL.PENBNK4	Yes	No
0x4004C260	PMG_TST_SRAM_CTL.PENBNK5	Yes	No
0x4004C260	PMG_TST_SRAM_CTL.PENBNK6	Yes	No
0x4004C260	PMG_TST_SRAM_CTL.PENBNK7	Yes	No
0x4004C260	PMG_TST_SRAM_CTL.AUTOINIT	Yes	No
0x4004C260	PMG_TST_SRAM_CTL.BNK1EN	Yes	No
0x4004C260	PMG_TST_SRAM_CTL.BNK2EN	Yes	No
0x4004C260	PMG_TST_SRAM_CTL.BNK7EN	Yes	No
0x4004C270	PMG_SCRPAD_3V_RD	Yes	Yes
0x4004C274	PMG_FAST_SHT_WAKEUP	Yes	Yes

ADuCM4050 PMG レジスタの説明

パワー・マネージメント・レジスタ（PMG）には以下のレジスタがあります。

表 4-4：ADuCM4050 PMG レジスター覧

レジスタ名	説明
PMG_CTL1	HPBUCK 制御
PMG_IEN	電源モニタ割込みイネーブル
PMG_TRIM	トリミング・ビット
PMG_PSM_STAT	電源モニタのステータス
PMG_PWRKEY	PMG_PWRMOD および PMG_SRAMRET のキー保護
PMG_PWRMOD	電力モードのレジスタ

表 4-4：ADuCM4050 PMG レジスター一覧（続き）

レジスタ名	説明
PMG_RST_STAT	リセット・ステータス
PMG_SHDN_STAT	シャットダウン・ステータス・レジスタ
PMG_SRAMRET	休止モードでの保持 SRAM の制御

HPBUCK 制御

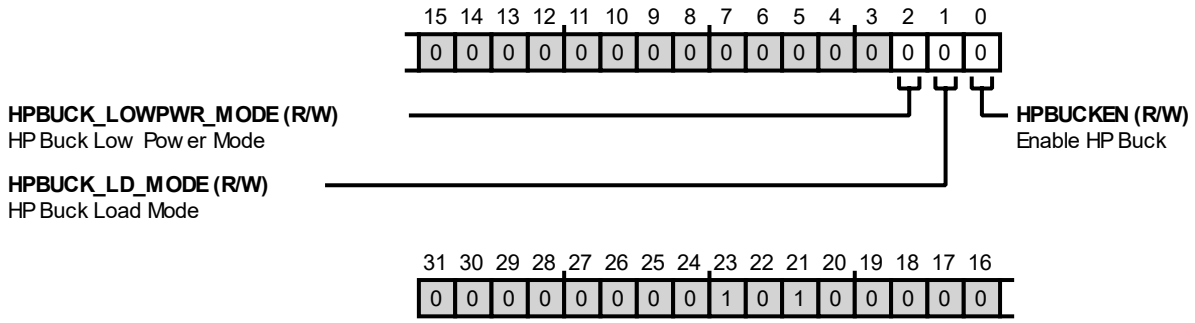


図 4-3：PMG_CTL1 レジスタ図

表 4-5：PMG_CTL1 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
2 (R/W)	HPBUCK_LOWPWR_M ODE	HP 降圧低消費電力モード。 このビットを使用して HPBUCK の低消費電力モードを選択できます。チップが Flexi 電力モードにあり、タイマー、ビーパなどの低消費電力モジュールのみがイネーブルの場合、HPBUCK 低消費電力モードを選択できます。	
		0	HPBUCK 低消費電力モードを無効にする
		1	HPBUCK 低消費電力モードを有効にする
1 (R/W)	HPBUCK_LD_MODE	HP 降圧負荷モード。 HPBUCK の負荷機能を選択するのに使用します。様々な使用事例に対して HPBUCK 負荷モードを選択できます。	
		0	HPBUCK 低負荷モードを有効にします。最大システム・クロック（HCLK）周波数が 26MHz の場合、このビットを 0 に設定します。
		1	HPBUCK 高負荷モードを有効にします。システム・クロック（HCLK）周波数が 26MHz を超える場合、このビットを 1 に設定します。
0 (R/W)	HPBUCKEN	HP 降圧を有効にします。	

電源モニタ割込みイネーブル

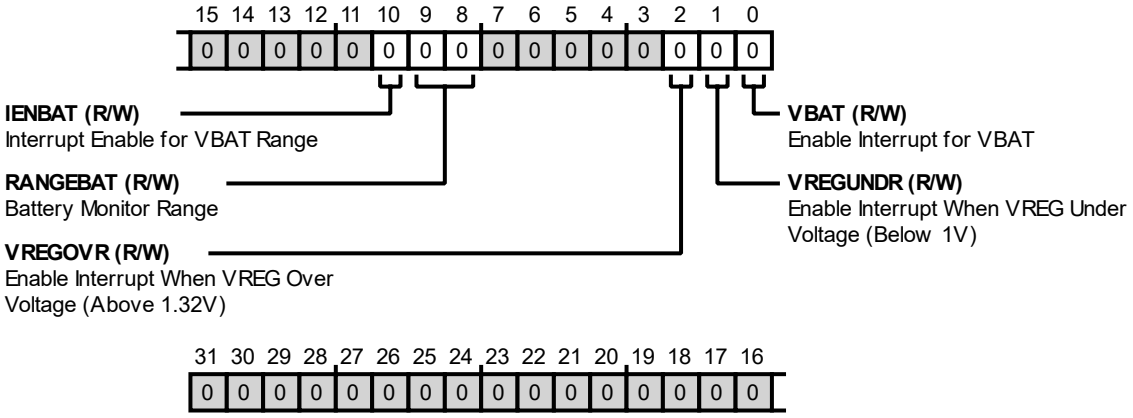


図 4-4：PMG_IEN レジスタ図

表 4-6：PMG_IEN レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
10 (R/W)	IENBAT	VBAT 範囲の割込みイネーブル。 PMG_IEN.RANGEBAT に応じて適切に割込みを生成させる必要がある場合は、このビットを設定します。適切に割込みを生成するように PMG_IEN.RANGEBAT を設定し、このビットをセットします。VBAT が PMG_IEN.RANGEBAT の範囲内に入ると、割込みが生成されます。 例えば、バッテリが良好であるかどうかを監視するには、PMG_PSM_STAT.RANGE2 または PMG_PSM_STAT.RANGE3 で設定し、すべての PMG_PSM_STAT フラグをクリアしてから、この割込みを有効にします。バッテリが不良の場合、バッテリの PMG_PSM_STAT.RANGE1 が割込みを生成し続けます。

表 4-6：PMG_IEN レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
9:8 (R/W)	RANGEBAT	バッテリ監視範囲。 適切に割込みを生成するように PMG_IEN.RANGEBAT を設定します。	
		0	Region1 の VBAT Range1 の VBAT が PMG_CTL1.HPBUCK_LD_MODE ビットに応じて変化する場合、割込みを生成するように設定します。 VBAT > 2.80V（PMG_CTL1.HPBUCK_LD_MODE = 0 のとき）。 VBAT > 2.93V（PMG_CTL1.HPBUCK_LD_MODE = 1 のとき）。
		1	Region2 の VBAT Range2 の VBAT が PMG_CTL1.HPBUCK_LD_MODE ビットに応じて変化する場合、割込みを生成するように設定します。 VBAT は 2.8V～2.19V（PMG_CTL1.HPBUCK_LD_MODE = 0 のとき）。 VBAT は 2.93V～2.39V（PMG_CTL1.HPBUCK_LD_MODE = 1 のとき）。
		2	Region3 の VBAT Range3 の VBAT が PMG_CTL1.HPBUCK_LD_MODE ビットに応じて変化する場合、割込みを生成するように設定します。 VBAT は 2.19V～1.6V（PMG_CTL1.HPBUCK_LD_MODE = 0 のとき）。 VBAT は 2.39V～1.6V（PMG_CTL1.HPBUCK_LD_MODE = 1 のとき）。
		3	該当なし
2 (R/W)	VREGOVR	VREG が過電圧（1.32V 以上）時の割込みを有効にします。	
1 (R/W)	VREGUNDR	VREG が低電圧（1V 未満）時の割込みを有効にします。 有効な場合、IRQ は NMI（マスク不能割込み）に連動します。	
0 (R/W)	VBAT	VBAT の割込みを有効にします。 有効な場合、IRQ は NMI（マスク不能割込み）に連動します。有効な場合、VBAT < 1.83V ならば割込みを生成します。	

トリミング・ビット

パワー・マネージメント用のトリミング・ビットです。

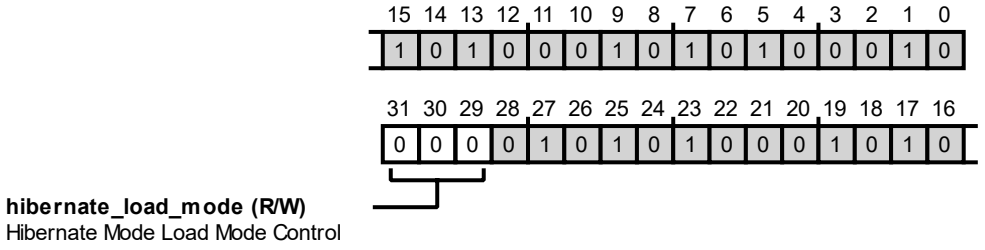


図 4-5：PMG_TRIM レジスタ図

表 4-7：PMG_TRIM レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
31:29 (R/W)	HIBER- NATE_LOAD_MODE	休止モードでの負荷モードの制御。 PMG_SRAMRET.HIBERNATE_SRAM_LOAD_MODE と組み合わせて設定します。 このビット・フィールドを 2 つの休止モード間で動的に変更することは、再構成の遅延を招くため推奨されません。休止時の負荷が低い場合（保持される SRAM が 60KB 以下の場合）、PMG_SRAMRET.HIBERNATE_SRAM_LOAD_MODE をセットし、このビット・フィールドに 3b111 を設定します。 休止時の負荷が高い場合（保持される SRAM が 60KB を超える場合）、PMG_SRAMRET.HIBERNATE_SRAM_LOAD_MODE をリセットし、このビット・フィールドに 3b000 を設定します。	
		0	休止時の負荷が高い
		7	休止時の負荷が低い

電源モニタのステータス

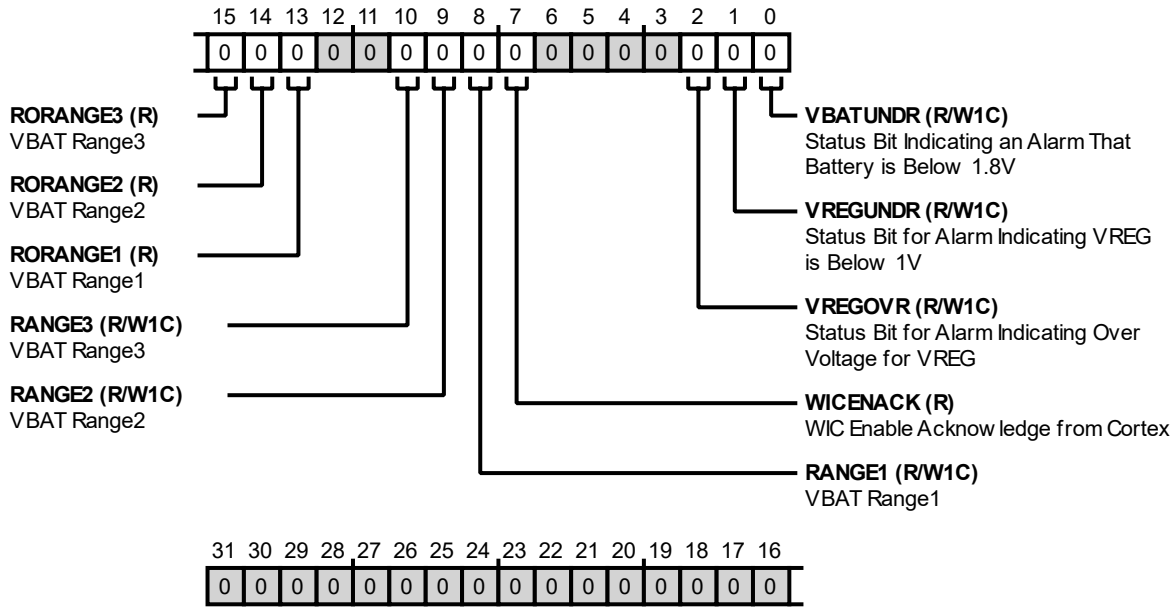


図 4-6：PMG_PSM_STAT レジスタ図

表 4-8：PMG_PSM_STAT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15 (R/NW)	RORANGE3	VBAT Range3。読出し専用ステータス・ビット	
		0	VBAT が指定範囲外
		1	VBAT が指定範囲内
14 (R/NW)	RORANGE2	VBAT Range2。読出し専用ステータス・ビット	
		0	VBAT が指定範囲外
		1	VBAT が指定範囲内
13 (R/NW)	RORANGE1	VBAT Range1。読出し専用ステータス・ビット。	
		0	VBAT が指定範囲外
		1	VBAT が指定範囲内

表 4-8：PMG_PSM_STAT レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
10 (R/W1C)	RANGE3	VBAT Range3。 関連する VBAT 範囲を示します。PMG_IEN.IENBAT が設定されている場合、VBAT 範囲の割込みを生成します。 注：VBAT が指定範囲内にある場合、'1'（フラグ・クリア）が書き込まれてフラグがクリアされた後でも、ステータス・ビットは再びセットされます。 VBAT Range3 は、PMG_CTL1.HPBUCK_LD_MODE ビットによって異なります。 VBAT Range3 の値： 2.19V～1.6V（PMG_CTL1.HPBUCK_LD_MODE = 0 のとき）。 2.39 V～1.6V（PMG_CTL1.HPBUCK_LD_MODE = 1 のとき）。	
		0	VBAT が指定範囲外
		1	VBAT が指定範囲内
9 (R/W1C)	RANGE2	VBAT Range2。 関連する VBAT 範囲を示します。PMG_IEN.IENBAT が設定されている場合、VBAT 範囲の割込みを生成します。 注：VBAT が指定範囲内にある場合、'1'（フラグ・クリア）が書き込まれてフラグがクリアされた後でも、ステータス・ビットは再びセットされます。 VBAT Range2 は、PMG_CTL1.HPBUCK_LD_MODE ビットによって異なります。 VBAT Range2 の値： 2.80V～2.19V（PMG_CTL1.HPBUCK_LD_MODE = 0 のとき）。 2.93V～2.39V（PMG_CTL1.HPBUCK_LD_MODE = 1 のとき）。	
		0	VBAT が指定範囲外
		1	VBAT が指定範囲内
8 (R/W1C)	RANGE1	VBAT Range1。 関連する VBAT 範囲を示します。PMG_IEN.IENBAT が設定されている場合、VBAT 範囲の割込みを生成します。 注：VBAT が指定範囲内にある場合、'1'（フラグ・クリア）が書き込まれてフラグがクリアされた後でも、ステータス・ビットは再びセットされます。 VBAT Range1 は、PMG_CTL1.HPBUCK_LD_MODE ビットによって異なります。 VBAT Range1 の値： VBAT > 2.80V（PMG_CTL1.HPBUCK_LD_MODE = 0 のとき）。 VBAT > 2.93V（PMG_CTL1.HPBUCK_LD_MODE = 1 のとき）。	
		0	VBAT が指定範囲外
		1	VBAT が指定範囲内

表 4-8：PMG_PSM_STAT レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7 (R/NW)	WICENACK	Cortex からの WIC イネーブル・アクノレッジ。
2 (R/W1C)	VREGOVR	VREG の過電圧を示すアラームのステータス・ビット。 VREG（LDO 出力）> 1.32V の場合、PMG_IEN.VREGOVR がセットされている場合に割込みを生成します。 注：VREG > 1.32V の場合、'1'（フラグ・クリア）が書き込まれてフラグがクリアされた後でも、ステータス・ビットは再びセットされます。
1 (R/W1C)	VREGUNDR	VREG が 1V 未満であることを示すアラームのステータス・ビット。 PMG_IEN.VREGUNDR がセットされている場合、割込みを生成します。このビットは、VREG < 1V の場合にセットされます。 注：VREG < 1V の場合、'1'（フラグ・クリア）が書き込まれてフラグがクリアされた後でも、ステータス・ビットは再びセットされます。
0 (R/W1C)	VBATUNDR	バッテリが 1.8V 未満であることを示すアラームのステータス・ビット。 PMG_IEN.VBAT がセットされている場合、割込みを生成します。このビットは、VBAT < 1.83V の場合にセットされます。 注：VBAT < 1.83V の場合、'1'（フラグ・クリア）が書き込まれてフラグがクリアされた後でも、ステータス・ビットは再びセットされます。

PMG_PWRMOD および PMG_SRAMRET のキー保護

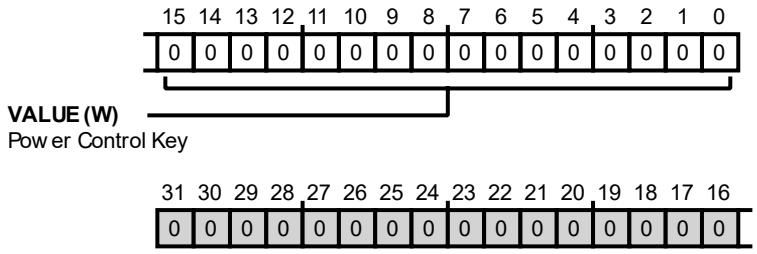


図 4-7：PMG_PWRKEY レジスタ図

表 4-9：PMG_PWRKEY レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (RX/W)	VALUE	パワー・コントロール・レジスタ PMG_PWRMOD レジスタと PMG_SRAMRET レジスタはキーで保護されています。PMG_PWRMOD と PMG_SRAMRET レジスタの値を変更するには、キーへの書込みが 1 回必要です。書き込むキーは 0x4859 です。その後にこれらのレジスタに書き込む必要があります。PMG_PWRMOD または PMG_SRAMRET に書き込む前に、APB バス上の他のレジスタに書き込むと、保護がロック状態に戻ります。

電力モードのレジスタ

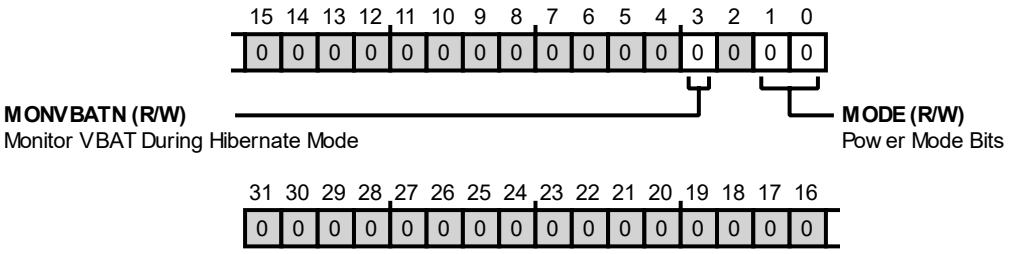


図 4-8：PMG_PWRMOD レジスタ図

表 4-10：PMG_PWRMOD レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
3 (R/W)	MONVBATN	休止モード中に VBAT を監視します。 デフォルトでは VBAT を監視します。VDD 監視は無効にできません。	
		0	PMG ブロックで VBAT 監視を有効にする
		1	PMG ブロックで VBAT 監視を無効にする
1:0 (R/W)	MODE	電力モード・ビット。	
		0	Flexi モード
		1	予備
		2	休止モード
		3	シャットダウン・モード

リセット・ステータス

このレジスタは、リセット要因を判別するためにユーザ・コードの先頭で読み出すことを推奨します。様々なリセット要因があり得るので、このレジスタのデフォルト値は指定されません。

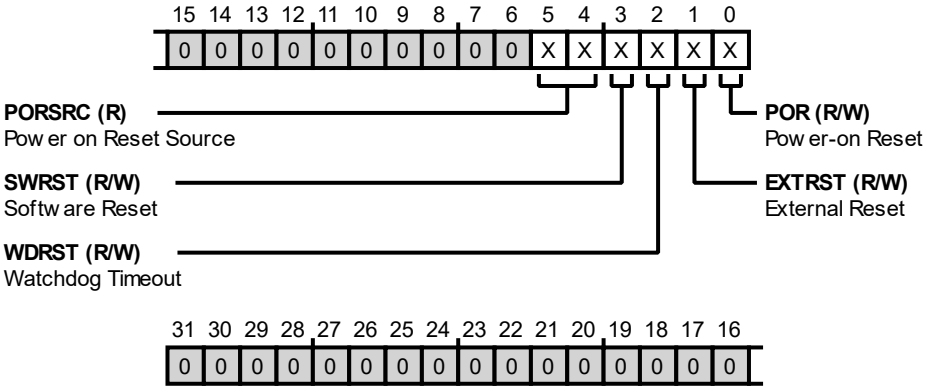


図 4-9：PMG_RST_STAT レジスタ図

表 4-11：PMG_RST_STAT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
5:4 (R/NW)	PORSRC	パワーオン・リセットの要因。 このビットには、パワーオン・リセットが発生した後の詳細情報が追加されています。	
		0	VBAT がフェイルセーフを下回ったために POR がトリガされた
		1	VBAT 電源が原因（VBAT < 1.7V）で POR がトリガされた
		2	VDD 電源が原因（VDD < 1.08V）で POR がトリガされた
		3	VREG がフェイルセーフを下回ったために POR がトリガされた
3 (R/W)	SWRST	ソフトウェア・リセット。 ソフトウェア・リセット。Cortex システムのリセットが生成されたときに自動的に 1 にセットされます。PMG_RST_STAT レジスタの任意のフィールドに 1 を書き込むことでクリアされます。このビットは、パワーオン・リセットがトリガされたときにもクリアされます	
2 (R/W)	WDRST	ウォッチドッグ・タイムアウト。 ウォッチドッグ・タイムアウトが発生すると、自動的に 1 にセットされます。PMG_RST_STAT レジスタの任意のフィールドに 1 を書き込むことでクリアされます。このビットは、パワーオン・リセットがトリガされたときにもクリアされます。	
1 (R/W)	EXTRST	外部リセット。 外部リセットが発生すると自動的に 1 にセットされます。PMG_RST_STAT レジスタの任意のフィールドに 1 を書き込むことでクリアされます。このビットは、パワーオン・リセットがトリガされたときにもクリアされます。	

表 4-11：PMG_RST_STAT レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
0 (R/W)	POR	パワーオン・リセット。 パワーオン・リセットが発生すると自動的にセットされます。PMG_RST_STAT レジスタの任意のフィールドに 1 を書き込むことでクリアされます。

シャットダウン・ステータス・レジスタ

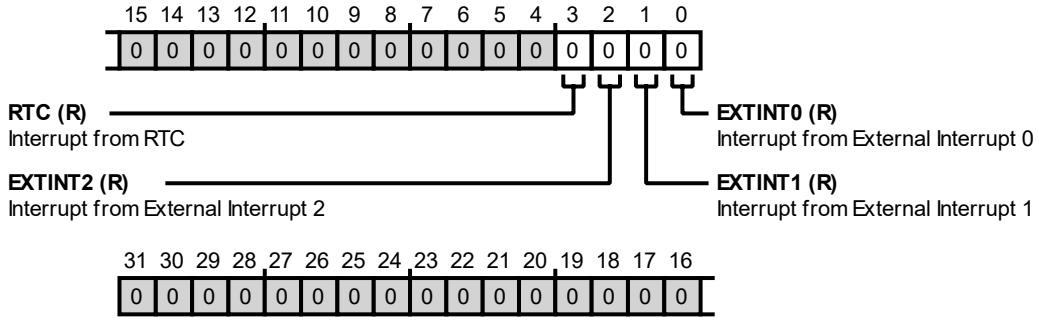


図 4-10：PMG_SHDN_STAT レジスタ図

表 4-12：PMG_SHDN_STAT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
3 (R/NW)	RTC	RTC からの割込み。
2 (R/NW)	EXTINT2	外部割込み 2 からの割込み。
1 (R/NW)	EXTINT1	外部割込み 1 からの割込み。
0 (R/NW)	EXTINT0	外部割込み 0 からの割込み。

休止モードでの保持 SRAM の制御

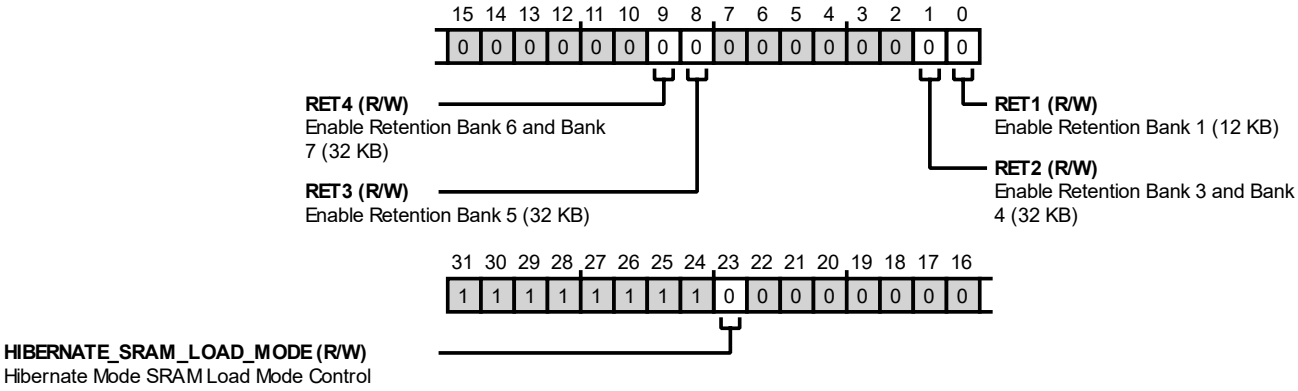


図 4-11：PMG_SRAMRET レジスタ図

表 4-13：PMG_SRAMRET レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
23 (R/W)	HIBERNATE_SRAM_ LOAD_M ODE	休止モード SRAM 負荷モード制御。 休止モード時に適切な負荷電流がサポートされるよう設定します。
9 (R/W)	RET4	保持バンク 6 とバンク 7（32KB）をイネーブルにします。
8 (R/W)	RET3	保持バンク 5（32KB）をイネーブルにします。
1 (R/W)	RET2	保持バンク 3 とバンク 4（32KB）をイネーブルにします。
0 (R/W)	RET1	保持バンク 1（12KB）をイネーブルにします。

ADuCM4050 PMG_TST レジスタの説明

パワー・マネージメント・レジスタ（PMG_TST）には以下のレジスタがあります。

表 4-14：ADuCM4050 PMG_TST レジスター覧

レジスタ名	説明
PMG_TST_CLR_LATCH_GPIOS	シャットダウン・モード後の GPIO のクリア
PMG_TST_FAST_SHT_WAKEUP	高速シャットダウン・ウェイクアップ・イネーブル

表 4-14：ADuCM4050 PMG_TST レジスター一覧（続き）

レジスタ名	説明
PMG_TST_SCRPAD_3V_RD	バッテリ・ドメインに保存されたスクラッチ・パッド
PMG_TST_SCRPAD_IMG	スクラッチ・パッドのイメージ
PMG_TST_SRAM_CTL	SRAM パリティおよび命令 SRAM の制御
PMG_TST_SRAM_INITSTAT	初期化ステータス・レジスタ

シャットダウン・モード後の GPIO のクリア

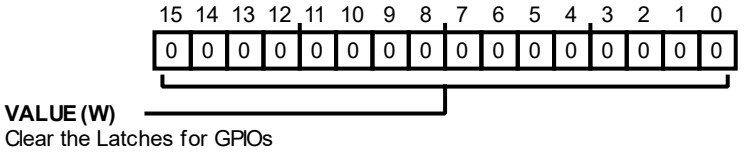


図 4-12：PMG_TST_CLR_LATCH_GPIOS レジスタ図

表 4-15：PMG_TST_CLR_LATCH_GPIOS レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (RX/W)	VALUE	GPIO のラッチをクリアします。 0x58FA が書き込まれると、GPIO はシャットダウン・モード後にラッチされた値を解放します。

高速シャットダウン・ウェイクアップ・イネーブル

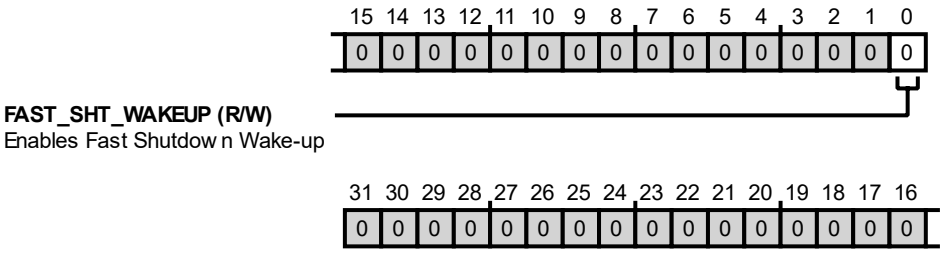


図 4-13：PMG_TST_FAST_SHT_WAKEUP レジスタ図

表 4-16：PMG_TST_FAST_SHT_WAKEUP レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
0 (R/W)	FAST_SHT_WAKEUP	高速シャットダウン・ウェイクアップを有効にします。 有効にすると、シャットダウン・ウェイクアップ時間は通常のシャットダウン・ウェイクアップ時間である 75ms が 1ms に短縮されます。 このレジスタ・フィールドの内容は、すべての電力モードで保持されます。このレジスタ・フィールドは、外部リセットまたはフェイルセーフの VBAT POR リセット時にのみ 0 にクリアされます。デフォルト値は 0 です。	
		0	高速シャットダウン・ウェイクアップを無効にする
		1	高速シャットダウン・ウェイクアップを有効にする

バッテリ・ドメインに保存されたスクラッチ・パッド

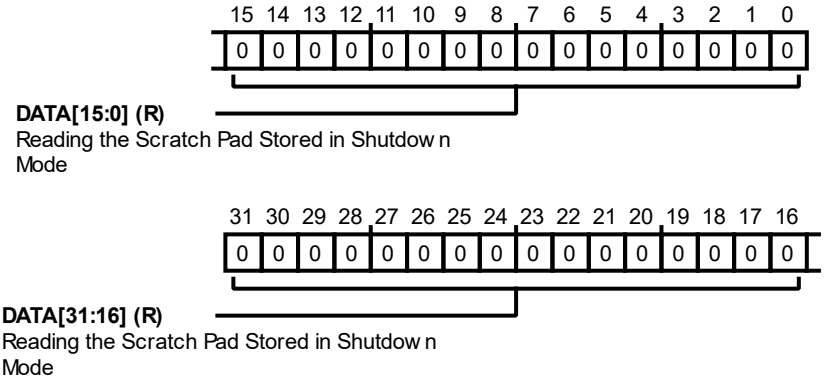


図 4-14：PMG_TST_SCRPAD_3V_RD レジスタ図

表 4-17：PMG_TST_SCRPAD_3V_RD レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	DATA	シャットダウン・モードで保存されたスクラッチ・パッドを読み出します。 読出し専用レジスタ。

スクラッチ・パッドのイメージ

シャットダウン・モードで役立ちます。GPIO 設定の OUT レジスタは、シャットダウン・モード時に内容を失います。32 ビットのスクラッチ・レジスタを使用してその GPIO 設定を保存できます。このビット幅は、GPIO に関連するすべての設定値と出力値を格納するには十分ではないことに注意してください。シャットダウン・ポート設定の組み合わせはごくわずかしかありません。

したがって、これらの組み合わせは、等価な符号化データを 32 ビットのスクラッチ・レジスタに格納することによって記憶できます。SCRATCHPAD_3V_READ レジスタを読み出した後（シャットダウンからのウェイクアップ後）、GPIO の設定を知ることができます。

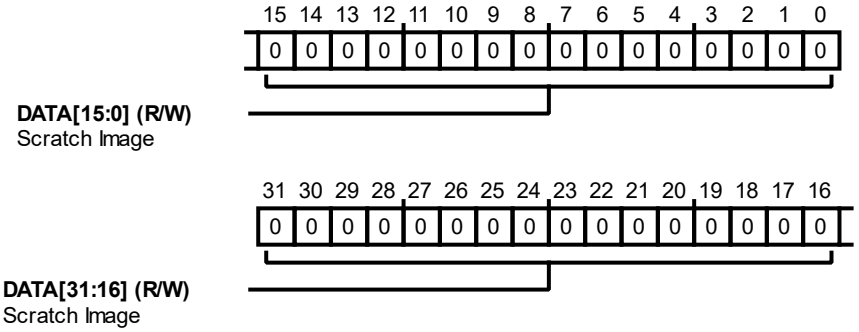


図 4-15：PMG_TST_SCRPAD_IMG レジスタ図

表 4-18：PMG_TST_SCRPAD_IMG レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (R/W)	DATA	スクラッチのイメージ。 このレジスタに書き込まれた値は、シャットダウン・モードに移行するときに 3V で保存されます。

SRAM パリティおよび命令 SRAM の制御

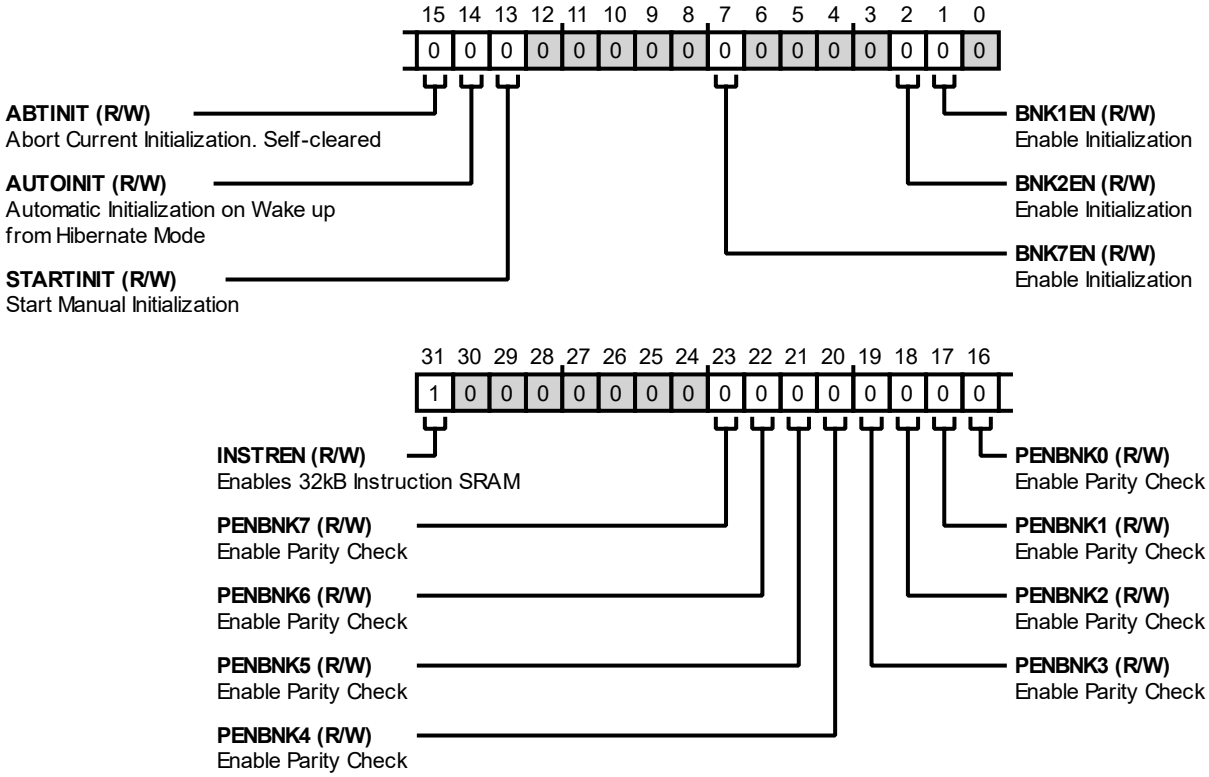


図 4-16：PMG_TST_SRAM_CTL レジスタ図

表 4-19：PMG_TST_SRAM_CTL レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31 (R/W)	INSTREN	32kB 命令 SRAM をイネーブルにします。
23 (R/W)	PENBNK7	パリティ・チェックを有効にします。
22 (R/W)	PENBNK6	パリティ・チェックを有効にします。
21 (R/W)	PENBNK5	パリティ・チェックを有効にします。
20 (R/W)	PENBNK4	パリティ・チェックを有効にします。
19 (R/W)	PENBNK3	パリティ・チェックを有効にします。

表 4-19：PMG_TST_SRAM_CTL レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
18 (R/W)	PENBNK2	パリティ・チェックを有効にします。
17 (R/W)	PENBNK1	パリティ・チェックを有効にします。
16 (R/W)	PENBNK0	パリティ・チェックを有効にします。
15 (R/W)	ABTINIT	現在の初期化を中止します。自動クリアされます。
14 (R/W)	AUTOINIT	休止モードからのウェイクアップ時に自動初期化します。
13 (R/W)	STARTINIT	手動初期化を開始します。 初期化をトリガするには、1 を書き込みます。自動クリアされます。
7 (R/W)	BNK7EN	初期化を有効にします。
2 (R/W)	BNK2EN	初期化を有効にします。
1 (R/W)	BNK1EN	初期化を有効にします。

初期化ステータス・レジスタ

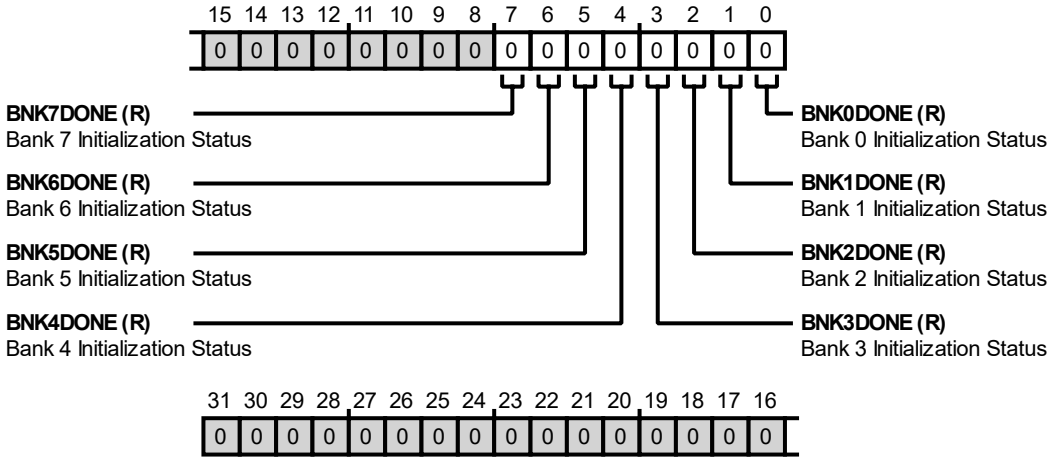


図 4-17：PMG_TST_SRAM_INITSTAT レジスタ図

表 4-20：PMG_TST_SRAM_INITSTAT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
7 (R/NW)	BNK7DONE	バンク 7 の初期化ステータス。	
		0	バンク 7 は未初期化
		1	バンク 7 は初期化済み
6 (R/NW)	BNK6DONE	バンク 6 の初期化ステータス。	
		0	バンク 6 は未初期化
		1	バンク 6 は初期化済み
5 (R/NW)	BNK5DONE	バンク 5 の初期化ステータス。	
		0	バンク 5 は未初期化
		1	バンク 5 は初期化済み
4 (R/NW)	BNK4DONE	バンク 4 の初期化ステータス。	
		0	バンク 4 は未初期化
		1	バンク 4 は初期化済み
3 (R/NW)	BNK3DONE	バンク 3 の初期化ステータス。	
		0	バンク 3 は未初期化
		1	バンク 3 は初期化済み

表 4-20：PMG_TST_SRAM_INITSTAT レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
2 (R/NW)	BNK2DONE	バンク 2 の初期化ステータス。	
		0	バンク 2 は未初期化
		1	バンク 2 は初期化済み
1 (R/NW)	BNK1DONE	バンク 1 の初期化ステータス。	
		0	バンク 1 は未初期化
		1	バンク 1 は初期化済み
0 (R/NW)	BNK0DONE	バンク 0 の初期化ステータス。	
		0	バンク 0 は未初期化
		1	バンク 0 は初期化済み

5 汎用 I/O（GPIO）

このセクションでは、汎用入出力（GPIO）の機能について説明します。

GPIO 機能

GPIO ポートには、以下のような機能があります。

- 入出力モードの GPIO 動作。
- ピン単位で個別に制御されるポート・マルチプレクス。
- すべてのポート・ピンで割込み機能を提供。
- プログラマブルなプルアップ／プルダウン駆動互換。

GPIO 機能の説明

ADuCM4050 MCU は、1 組のメモリ・マップド・レジスタ（MMR）を介して GPIO ピンを制御します。これらの MMR は、APB32 ペリフェラルの一部として実装されています。ほとんどの GPIO ピンに複数の機能がマップされています。これらの機能は、`GPIO_CFG` レジスタの対応するポートを適切に設定することによって選択できます。

ADuCM4050 MCU は最大 51 本の GPIO 双方向ピンを備えています。ほとんどの GPIO ピンは、最上位レベルでマルチプレクスした場合、ユーザ・コードによって設定可能な複数の機能を持ちます。GPIO は、P0、P1、P2、P3 の 4 つのポートにグループ化されています。P0、P1、P2 には 16 の GPIO が含まれます。P3 には 4 つの GPIO が含まれます。

GPIO のブロック図

この図は、プルアップ抵抗とプルダウン抵抗を取り付けた GPIO ピンのブロック図を示しています。

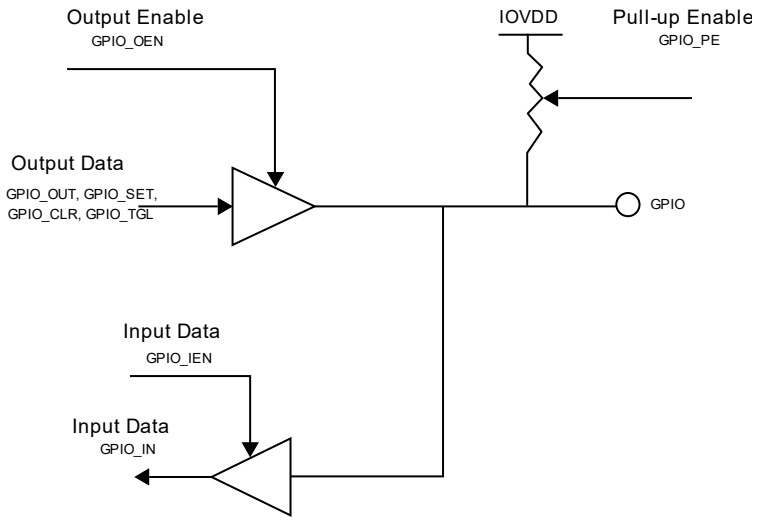


図 5-1：プルアップ抵抗を取り付けた GPIO ピン

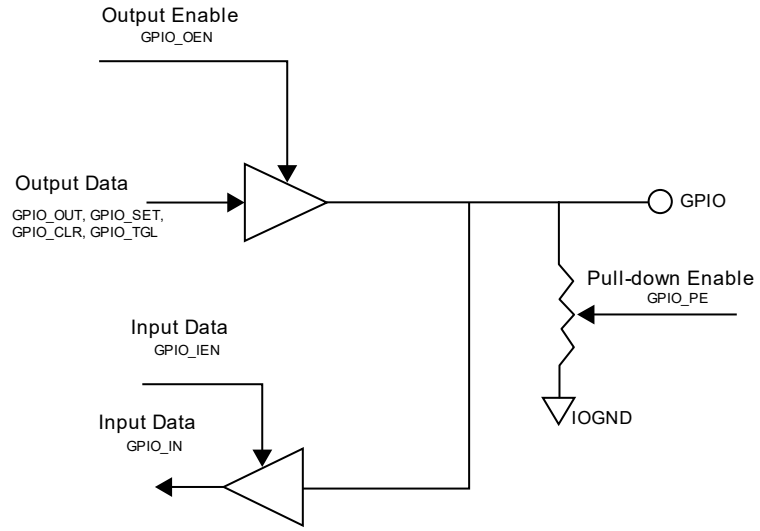


図 5-2：プルダウン抵抗を取り付けた GPIO ピン

入力がシステムによって制御されないアプリケーションでは、入力電圧がデバイスの最大電圧定格を超えることがあります。外部過電圧状態を ADuCM4050 MCU に印加する場合は、アンプと電気的過負荷の間で ESD ダイオードを使用する必要があります。

すべての GPIO は、過電圧および ESD から保護するダイオード・クランプ回路を備えています。

ダイオード・クランプの等価回路を次の図に示します。

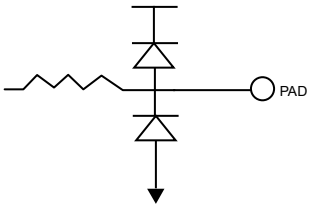


図 5-3：ダイオード・クランプ回路

ADuCM4050 GPIO のマルチプレクス

次の表に、パッケージの汎用 I/O ピンにマルチプレクスされているピン機能を示します。

表 5-1：信号のマルチプレクス - ポート 0

Signal Name	Multiplexed Function 0	Multiplexed Function 1	Multiplexed Function 2	Multiplexed Function 3
P0_00	P0_00	SPI0_CLK	SPT0_BCLK	
P0_01	P0_01	SPI0_MOSI	SPT0_BFS	
P0_02	P0_02	SPI0_MISO	SPT0_BD0	
P0_03	P0_03	SPI0_CS0	SPT0_BCNV	SPI2_RDY
P0_04	P0_04	I2C0_SCL		
P0_05	P0_05	I2C0_SDA		
P0_06	SWD0_CLK	P0_06		
P0_07	SWD0_DATA	P0_07		
P0_08	P0_08	$\overline{\text{BEEP0\_TONE}}$		
P0_09	P0_09	BEEP0_TONE	SPI2_CS1	
P0_10	P0_10	$\overline{\text{UART0\_TX}}$		
P0_11	P0_11	$\overline{\text{UART0\_RX}}$		
P0_12	P0_12	SPT0_AD0		UART_SOUT_EN
P0_13/SYS_WAKE2	P0_13			
P0_14	P0_14	TMR0_OUT	SPI1_RDY	
P0_15/SYS_WAKE0	P0_15			

表 5-2：信号のマルチプレクス - ポート 1

Signal Name	Multiplexed Function 0	Multiplexed Function 1	Multiplexed Function 2	Multiplexed Function 3
P1_00/SYS_WAKE1	P1_00			
P1_01	SYS_BMODE0	P1_01		
P1_02	P1_02	SPI2_CLK		

表 5-2：信号のマルチプレクス - ポート 1（続き）

Signal Name	Multiplexed Function 0	Multiplexed Function 1	Multiplexed Function 2	Multiplexed Function 3
P1_03	P1_03	SPI2_MOSI		
P1_04	P1_04	SPI2_MISO		
P1_05	P1_05	SPI2_CS0		
P1_06	P1_06	SPI1_CLK		RGB_TMR0_1
P1_07	P1_07	SPI1_MOSI		RGB_TMR0_2
P1_08	P1_08	SPI1_MISO		RGB_TMR0_3
P1_09	P1_09	SPI1_CS0		SWV
P1_10	P1_10	SPI0_CS1	SYS_CLKIN	SPI1_CS3
P1_11	P1_11		TMR1_OUT	
P1_12	P1_12		RTC1_SS2	
P1_13	P1_13	TMR2_OUT		
P1_14	P1_14		SPI0_RDY	
P1_15	P1_15	SPT0_ACLK	UART1_TX	

表 5-3：信号のマルチプレクス - ポート 2

Signal Name	Multiplexed Function 0	Multiplexed Function 1	Multiplexed Function 2	Multiplexed Function 3
P2_00	P2_00	SPT0_AFS	UART1_RX	
P2_01/SYS_WAKE3	P2_01		TMR2_OUT	
P2_02	P2_02	SPT0_ACNV	SPI1_CS2	
P2_03	P2_03	ADC0_VIN0		
P2_04	P2_04	ADC0_VIN1		
P2_05	P2_05	ADC0_VIN2		
P2_06	P2_06	ADC0_VIN3		
P2_07	P2_07	ADC0_VIN4	SPI2_CS3	
P2_08	P2_08	ADC0_VIN5	SPI0_CS2	RTC1_SS3
P2_09	P2_09	ADC0_VIN6	SPI0_CS3	
P2_10	P2_10	ADC0_VIN7	SPI2_CS2	
P2_11	P2_11	SPI1_CS1	SYS_CLKOUT	RTC1_SS1

GPIO の動作モード

IO プルアップまたはプルダウンの有効化

すべての GPIO ピンは、プルアップ抵抗付きの GPIO ピンとプルダウン抵抗付きの GPIO ピンの 2 つのカテゴリに分類できます。。各 GPIO ポートには GPIO_PE レジスタが対応しています。GPIO_PE レジスタを使用すると、入力として設定されているときに、ピンのプルアップ／プルダウン・レジスタをイネーブル／ディスエーブルできます。

IO データ入力

GPIO_IEN レジスタで GPIO を入力として設定すると、GPIO 入力レベルが GPIO_IN レジスタで得られます。

IO データ出力

GPIO_OEN レジスタで GPIO を出力として設定すると、GPIO_OUT レジスタの値が GPIO に反映されます。

ビット・セット

各 GPIO ポートには、ビット・セット・レジスタ GPIO_SET が対応しています。ビット・セット・レジスタを使用すると、ポート内の他の GPIO データ出力に影響を与えずに、1 つ以上の GPIO データ出力をセットできます。書込みデータ・ビットが 1 に対応する GPIO のみがセットされ、残りの GPIO は影響を受けません。

ビット・クリア

各 GPIO ポートには、ビット・クリア・レジスタ GPIO_CLR が対応しています。ビット・クリア・レジスタを使用すると、ポート内の他の GPIO データ出力に影響を与えずに、1 つ以上の GPIO データ出力をクリアできます。書込みデータ・ビットが 1 に対応する GPIO のみがクリアされ、残りの GPIO は影響を受けません。

ビット・トグル

各 GPIO ポートには、ビット・トグル・レジスタ GPIO_TGL が対応しています。ビット・トグル・レジスタを使用すると、ポート内の他の GPIO データ出力に影響を与えずに、1 つ以上の GPIO データ出力を反転することができます。書込みデータ・ビットが 1 に対応する GPIO のみがトグルされ、残りの GPIO は影響を受けません。

IO データ出力イネーブル

各 GPIO ポートは、データ出力イネーブル（GPIO_OEN）レジスタを備えており、これによってデータ出力パスがイネーブルされます。データ出力イネーブル・レジスタのビットをセットすると、GPIO_OUT の値が対応する GPIO ピンに反映されます。

割込み

各 GPIO ピンは割込みに関連付けることができます。割込みは GPIO ピンごとに個別に有効にされ、常にエッジ検出されます。各 GPIO ピンの遷移で 1 つの割込みのみが生成されます。検出されるエッジの極性は、正（ローからハイ）または負（ハイからロー）です。各 GPIO の割込みイベントは、2 つの割込み（INTA または INTB）のうちの 1 つにマップできます。これにより、サービスに対して GPIO 割込みをグループ化したり、割込みの優先順位を設定したり

するなど、システムの柔軟性を高めることができます。関連するステータス・レジスタにアクセスすることにより、各 GPIO ピンの割込みステータスを識別およびクリアできます。

割込み極性

割込み極性によって、割込みを立上がりエッジで受け付けるか、立下がりエッジで受け付けるかが決まります。各 GPIO ポートには割込み極性レジスタ（GPIO_POL）が対応しており、これによって各ピンの割込み極性を設定します。0 に設定すると、対応するピンのハイからローへの遷移時に割込みイベントがラッチされます。1 に設定すると、対応するピンのローからハイへの遷移時に割込みイベントがラッチされます。

割込み A の有効化

各 GPIO ポートには、ポートの各ピンをイネーブルまたはマスクするための割込みイネーブル A レジスタ（GPIO_IENA）が対応しています。これらのレジスタ・ビットは、ラッチされたエッジ・イベントがコアに割込みをかける（割込み A）か、マスクすべきかを決定します。どちらの場合でも、イベントの発生は GPIO_INT ステータス・レジスタの対応するビットに取り込まれます。0 に設定すると、割込み A は有効になりません（マスクされます）。この GPIO ピンによってコアへの割込みが発生することがなくなります。1 に設定すると、割込み A が有効になります。有効なエッジを検出すると、コアに割込みが発生します。

割込み B の有効化

各 GPIO ポートには、ポートの各ピンをイネーブルまたはマスクするための割込みイネーブル B レジスタ（GPIO_IENB）が対応しています。これらのレジスタ・ビットは、ラッチされたエッジ・イベントがコアに割込みをかける（割込み B）か、マスクすべきかを決定します。どちらの場合でも、イベントの発生は GPIO_INT ステータス・レジスタの対応するビットに取り込まれます。0 に設定すると、割込み B は有効になりません（マスクされます）。この GPIO ピンによってコアへの割込みが発生することがなくなります。1 に設定すると、割込み B が有効になります。有効なエッジを検出すると、コアに割込みが発生します。

割込みステータス

各 GPIO ポートには、ピンで発生した割込みを取り込むための割込みステータス・レジスタ（GPIO_INT）があります。これらのレジスタ・ビットは、適切に設定された立上がりまたは立下がりエッジが対応する GPIO ピンで検出されたことを示します。

イベントが検出されると、GPIO_INT はクリアされるまでセットされたままになります。これは GPIO ピンが非アクティブ状態に戻っても同じです。プルアップを立下がりエッジ検出と組み合わせた場合、リセット後に GPIO_INT のステータスがクリアされることがありますが、このピンの動作によるものとは限りません。したがって、GPIO_INT のステータスをチェックしてから、最初に割込み（GPIO_IENA と GPIO_IENB）を有効にし、任意の時点で GPIO ピンを設定することを推奨します。

割込みビットは、適切なビット位置に 1 を書き込むことによってクリアされます。0 を書き込んでも影響はありません。割込みがコアに対して有効になっている場合（GPIO_IENA、GPIO_IENB）、割込み（GPIO_INT）の値が 1 になるとコアに割込みが発生します。このビットは、割込み処理中にクリアする必要があります。0 を読み出した場合、このビットが最後にクリアされてから、対応する GPIO ピンで立上がりエッジまたは立下がりエッジが検出されなかったことを示します。1 を読み出した場合、対応する GPIO ピンで立上がりエッジまたは立下がりエッジ（GPIO_POL で選択可能）が検出されたことを示します。このビット位置に 1 を書き込むと、このビットをソフトウェアでクリアで

きます。このビットは、ピンが非アサート状態に復帰してから再びアサート状態に戻った後にのみ、再度セットすることができます。

GPIO プログラミング・モデル

出力機能のプログラミング・シーケンスは次のとおりです。

1. 適切な GPIO ピンの GPIO_OEN レジスタを設定します。
2. 選択したピンのデータは、GPIO_SET/GPIO_CLR/GPIO_TGL レジスタに書き込むことによってハイまたはローに駆動されます。

入力機能のプログラミング・シーケンスは次のとおりです。

1. 適切なポート・ピンの GPIO_IEN レジスタを設定します。
2. 入力ポートのピンによってラッチされたデータは、GPIO_IN レジスタに登録されます。割込みは、GPIO_IENA または GPIO_IENB レジスタに書き込むことによって有効にできます。

ADuCM4050 GPIO レジスタの説明

汎用入出力（GPIO）には以下のレジスタがあります。

表 5-4：ADuCM4050 GPIO レジスター覧

レジスタ名	説明
GPIO_CFG	ポート構成
GPIO_CLR	ポート・データ出力のクリア
GPIO_DS	ポート駆動能力の選択
GPIO_IEN	ポート入力パスのイネーブル
GPIO_IENA	ポート割込み A の有効化
GPIO_IENB	ポート割込み B の有効化
GPIO_IN	ポート・レジスタ・データ入力
GPIO_INT	ポート割込みステータス
GPIO_OEN	ポート出力イネーブル
GPIO_OUT	ポート・データ出力
GPIO_PE	ポート出力のプルアップ／プルダウンの有効化
GPIO_POL	ポート割込み極性
GPIO_SET	ポート・データ出力のセット
GPIO_TGL	ポート・ピンのトグル

ポート構成

GPIO_CFG レジスタは、GPIO ブロックのピンの最上位マルチプレクシング用に予約されています。

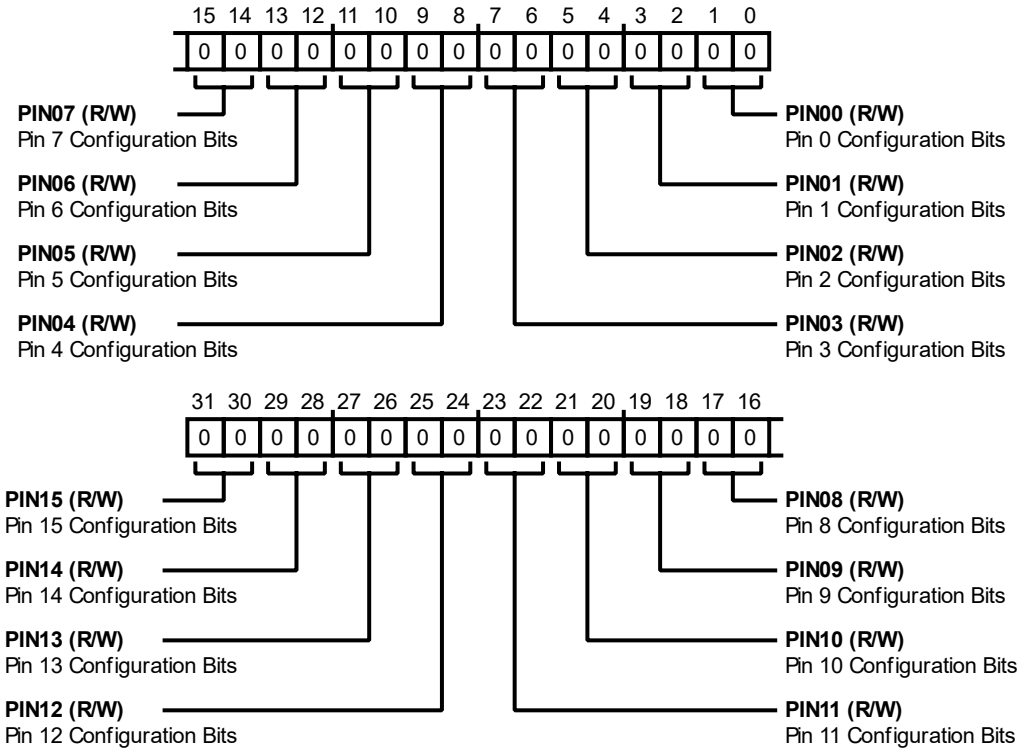


図 5-4：GPIO_CFG レジスタ図

表 5-5：GPIO_CFG レジスタのフィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:30 (R/W)	PIN15	ピン 15 の構成ビット。
29:28 (R/W)	PIN14	ピン 14 の構成ビット。
27:26 (R/W)	PIN13	ピン 13 の構成ビット。
25:24 (R/W)	PIN12	ピン 12 の構成ビット。
23:22 (R/W)	PIN11	ピン 11 の構成ビット。

表 5-5：GPIO_CFG レジスタのフィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応
21:20 (R/W)	PIN10	ピン 10 の構成ビット。
19:18 (R/W)	PIN09	ピン 9 の構成ビット。
17:16 (R/W)	PIN08	ピン 8 の構成ビット。
15:14 (R/W)	PIN07	ピン 7 の構成ビット。
13:12 (R/W)	PIN06	ピン 6 の構成ビット。
11:10 (R/W)	PIN05	ピン 5 の構成ビット。
9:8 (R/W)	PIN04	ピン 4 の構成ビット。
7:6 (R/W)	PIN03	ピン 3 の構成ビット。
5:4 (R/W)	PIN02	ピン 2 の構成ビット。
3:2 (R/W)	PIN01	ピン 1 の構成ビット。
1:0 (R/W)	PIN00	ピン 0 の構成ビット。

ポート・データ出力のクリア

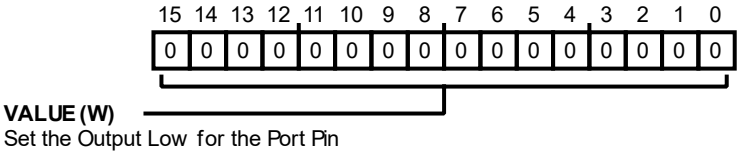


図 5-5：GPIO_CLR レジスタ図

表 5-6：GPIO_CLR レジスタのフィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (RX/W)	VALUE	ポート・ピンの出力をローに設定します。 各ビットは、対応する GPIO ピンをローに駆動するように設定されます。このビットをクリアしても影響はありません。

ポート駆動能力の選択

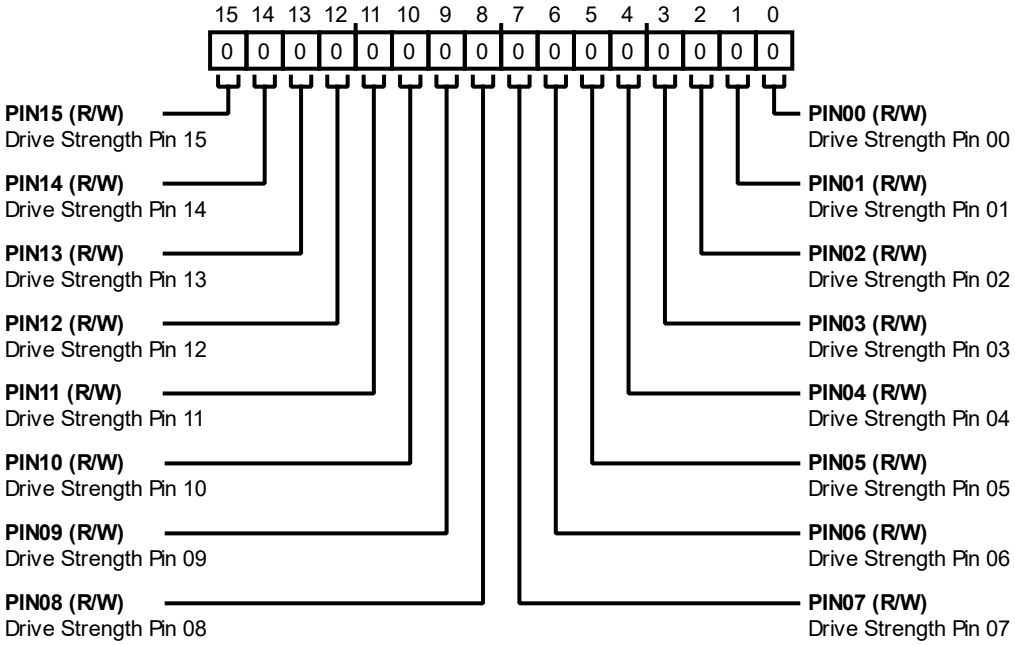


図 5-6：GPIO_DS レジスタ図

表 5-7：GPIO_DS レジスタのフィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
15 (R/W)	PIN15	ピン 15 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。
14 (R/W)	PIN14	ピン 14 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。
13 (R/W)	PIN13	ピン 13 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。
12 (R/W)	PIN12	ピン 12 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。

表 5-7：GPIO_DS レジスタのフィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
11 (R/W)	PIN11	ピン 11 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。
10 (R/W)	PIN10	ピン 10 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。
9 (R/W)	PIN09	ピン 09 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。
8 (R/W)	PIN08	ピン 08 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。
7 (R/W)	PIN07	ピン 07 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。
6 (R/W)	PIN06	ピン 06 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。
5 (R/W)	PIN05	ピン 05 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。
4 (R/W)	PIN04	ピン 04 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。
3 (R/W)	PIN03	ピン 03 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。
2 (R/W)	PIN02	ピン 02 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。

表 5-7：GPIO_DS レジスタのフィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
1 (R/W)	PIN01	ピン 01 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。
0 (R/W)	PIN00	ピン 00 の駆動能力。	
		0	シングル・ドライブ強度。
		1	ダブル・ドライブ強度。

ポート入力パスのイネーブル

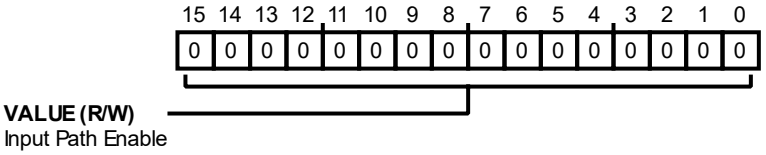


図 5-7：GPIO_IEN レジスタ図

表 5-8：GPIO_IEN レジスタのフィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	入力パス・イネーブル。 各ビットは、セットすると GPIO ピンの入力パスをイネーブルし、クリアすると入力パスをディスエーブルします。

ポート割込み A の有効化

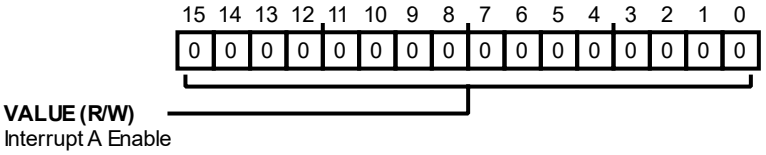


図 5-8：GPIO_IENA レジスタ図

表 5-9：GPIO_IENA レジスタのフィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	割込み A イネーブル。 ラッチされたエッジ・イベントによって、コアへの割込み（割込み A）を許可するか、またはマスクするかを決めます。どちらの場合でも、イベントの発生は GPIO_INT ステータス・レジスタに取り込まれます。クリアすると、割込み A は有効になりません（マスクされます）。セットすると、割込み A が有効になります。

ポート割込み B の有効化

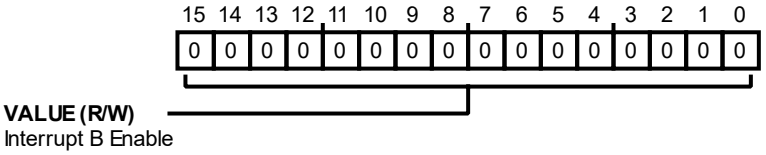


図 5-9：GPIO_IENB レジスタ図

表 5-10：GPIO_IENB レジスタのフィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	割込み B イネーブル ラッチされたエッジ・イベントによって、コアへの割込み（割込み B）を許可するか、またはマスクするかを決めます。どちらの場合でも、イベントの発生は GPIO_INT ステータス・レジスタに取り込まれます。0 に設定すると、割込み B は有効になりません（マスクされます）。1 に設定すると、割込み B が有効になります。

ポート・レジスタ・データ入力

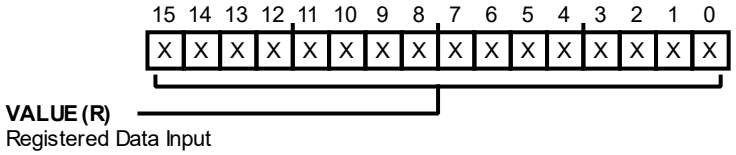


図 5-10：GPIO_IN レジスタ図

表 5-11：GPIO_IN レジスタのフィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	VALUE	レジスタ・データ入力 対応する入力バッファがイネーブルされている場合、各ビットは GPIO ピンの状態を反映します。ピン入力バッファがディスエーブルされている場合、値はゼロになります。

ポート割込みステータス

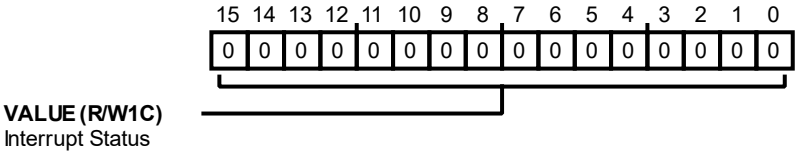


図 5-11：GPIO_INT レジスタ図

表 5-12：GPIO_INT レジスタのフィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W1C)	VALUE	割込みステータス。 対応する GPIO ピンで、適切に設定された立上がりまたは立下がりエッジが検出されたことを示します。イベントが検出されると、GPIO_INT.VALUE ビットはクリアされるまでセットされたままになります。これは、GPIO ピンが非アクティブ状態に戻っても同じです。GPIO_INT.VALUE ビットは、適切なビット位置に 1 を書き込むことによってクリアされます。0 を書き込んでも影響はありません。

ポート出力イネーブル

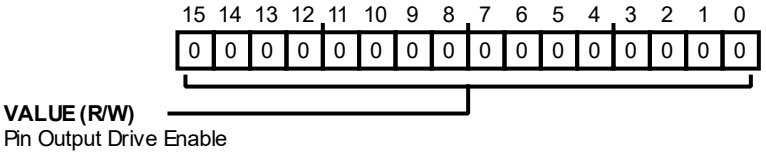


図 5-12：GPIO_OEN レジスタ図

表 5-13：GPIO_OEN レジスタのフィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	ピン出力駆動イネーブル。 各ビットは、セットすると、その特定のピン出力が有効になります。クリアすると、各ピンの出力が無効になります。

ポート・データ出力

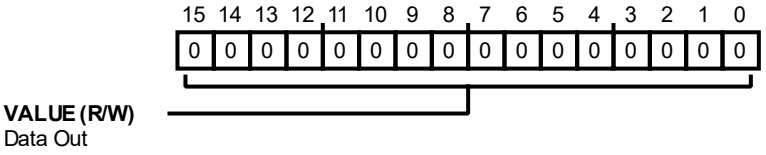


図 5-13：GPIO_OUT レジスタ図

表 5-14：GPIO_OUT レジスタのフィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	データ出力。 ユーザ・コードでセットすると、対応する GPIO がハイに駆動されます。クリアすると、対応する GPIO がローに駆動されます。

ポート出力のプルアップ／プルダウンの有効化

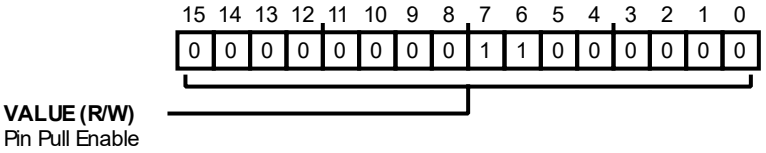


図 5-14：GPIO_PE レジスタ図

表 5-15：GPIO_PE レジスタのフィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	ピンのプルアップ・イネーブル。 各ビットは、セットすると、その特定のピンに対するプルアップ／プルダウンが有効になります。クリアすると、各ピンのプルアップ／プルダウンが無効になります。レジスタ図に示す値は、GPIO ポート 0 の値です。

ポート割込み極性

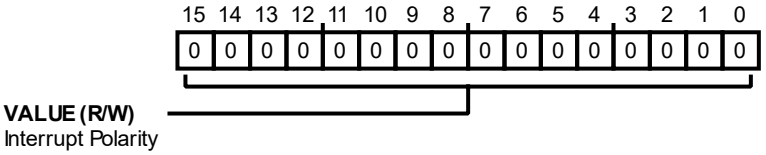


図 5-15：GPIO_POL レジスタの図

表 5-16：GPIO_POL レジスタのフィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	割込み極性。 対応するGPIOピンの立上がりエッジまたは立下がりエッジのどちらで割込みを生成させるかを決定します。クリアすると、割込みイベントがハイからローへの遷移でラッチされます。セットすると、割込みイベントがローからハイへの遷移でラッチされます。

ポート・データ出力のセット

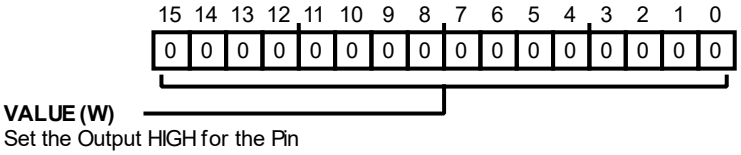


図 5-16：GPIO_SET レジスタ図

表 5-17：GPIO_SET レジスタのフィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (RX/W)	VALUE	ピンの出力をハイに設定します。 ユーザ・コードでセットすると、対応する GPIO がハイに駆動されます。このビットをクリアしても影響はありません。

ポート・ピンのトグル

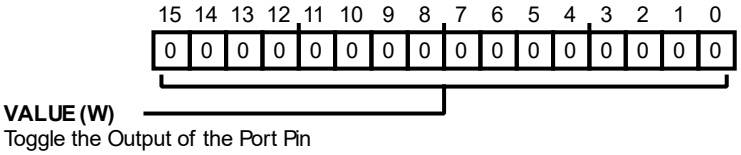


図 5-17：GPIO_TGL レジスタの図

表 5-18：GPIO_TGL レジスタのフィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (RX/W)	VALUE	ポート・ピンの出力をトグルします。 各ビットは、セットすると対応する GPIO ピンが反転します。このビットをクリアしても影響はありません。

6 システム・クロック

ADuCM4050 MCU は、2 つのオンチップ発振器と 2 つの外部水晶発振器用回路を内蔵しています。

システム・クロック機能

システム・クロックには、次の機能があります。

- LFOSC は 32kHz の内部発振器です。
- HFOSC は 26MHz の内部発振器です。
- LFXTAL は 32kHz の外部水晶発振器です。
- HFXTAL は 16／26MHz の外部水晶発振器です。
- SYS_CLKIN 経由の外部クロック入力。
- オンチップのフェーズ・ロック・ループ（PLL）をシステム PLL（SPLL）として利用できます。PLL は、HFOSC、HFXTAL、または SYS_CLKIN を入力クロックとして使用できます。
- 高周波数発振器（HFOSC および HFXTAL）、SYS_CLKIN、それに SPLL の出力をルート・クロックの生成に使用できます。
- 32kHz クロック（LF_CLK）は、LFXTAL または LFOSC から生成され、特定のブロックを駆動します。
- 汎用タイマーは、クロック源を選択するための専用のマルチプレクサを備えています。
- RTC1 は、休止モードとアクティブ・モードにおいて LFXTAL または LFOSC で動作します。RTC0 は、シャットダウン／休止状態／アクティブの各モードにおいて常に LFXTAL で動作します。
- ウォッチドッグ・タイマーは常に LFOSC 発振器で動作します。信頼性の問題を回避するために LFXTAL を使用して動作させることはできません。休止モードまたはシャットダウン・モードでは機能しません。
- シャットダウン・モード以外のすべてのモードで LFXTAL クロック不具合を検出。
- LFXTAL クロックの不具合時に LFMUX クロックを LFOSC に自動切替え。
- アクティブ・モードでルート・クロック不具合を検出。
- ルート・クロック不具合時にルート・クロックを HFOSC クロックに自動切替え。
- ルート・クロックはいくつかの内部クロックに分割されます。

- RCLK（ルート・クロックと混同しないでください）は、フラッシュ・コントローラのリファレンス・タイマー（カウンタ）にクロックを供給します。これは、フラッシュの消去や書込み動作のタイミングを調整するのに使用されます。デフォルトでは、13MHz のクロック源に接続されています。これは、デフォルトで HFOSC（26MHz）に接続された½分周器によって生成されます。したがって、フラッシュ・タイマー・レジスタのデフォルト値は 13MHz クロックになります。

  CLKG_CLK_CTL0.RCLKMUX（RHP_CLK）= 11（HFXTAL = 16MHz）の場合、½ DIV（分周器）は自動的にバイパスされ、HFXTAL（16MHz）は RCLK に直接接続されます。したがって、HFXTAL（16MHz）を使用する場合、フラッシュの消去／書込み操作の前にフラッシュ・タイマー・レジスタを 16MHz に設定する必要があります。
- HP_BUCK_CLK は HP 降圧モジュールにクロックを供給します。イネーブルにした場合、HPBUCK のデフォルトのクロック周波数は 200kHz です。HPBUCK クロック周波数には別のオプションがあり、高負荷モードとして 400kHz の周波数を生成します。
- ACLK は ADC にクロックを供給します。
- GPIOCLKOFF は GPIO 入力ロジックの、PERCLKOFF はペリフェラルのクロック・ゲートに使用されます。対応する読出し／書込みが発生すると、これらのビットは自動的にイネーブルされます。すなわち、ペリフェラル・レジスタに書込み／読出しを行うと PCLK が自動的に再イネーブルされ、GPIO に書込み／読出しを行うと GPIO CLK が再イネーブルされます。

システム・クロック機能の説明

ここでは、パワー・マネージメントに必要なクロックとクロック・ゲートについて説明します。

システム・クロックのブロック図

クロック供給図を以下に示します。

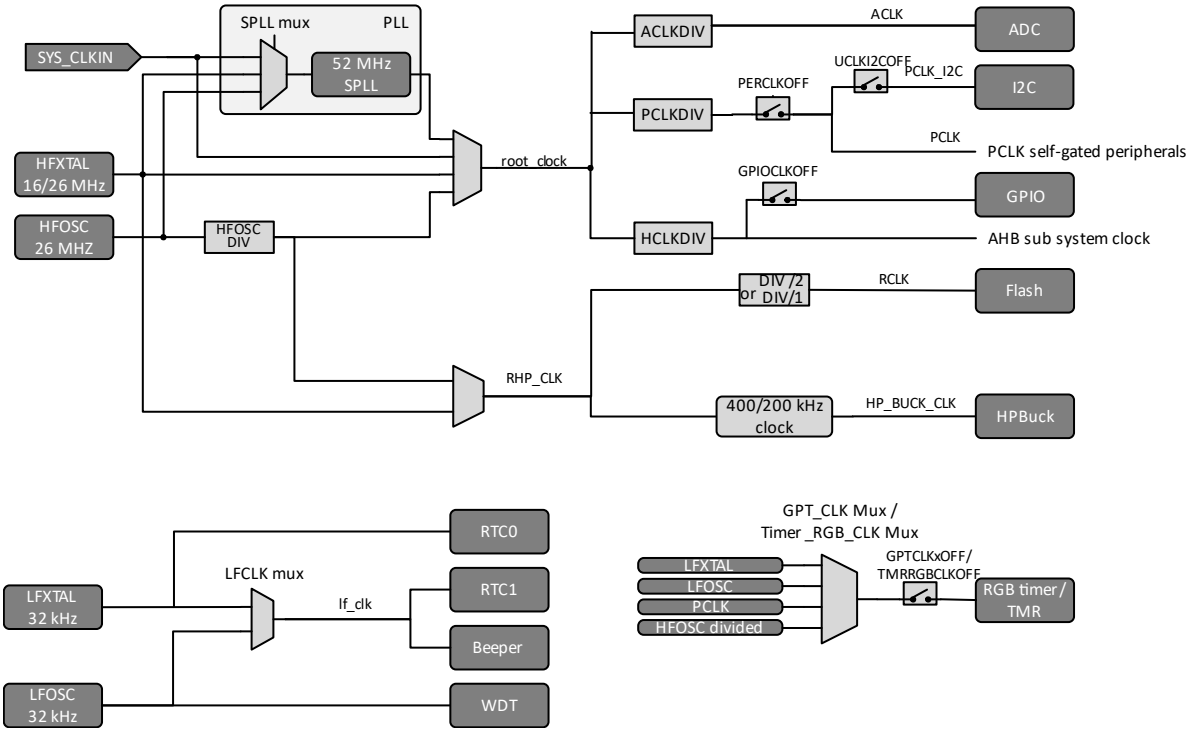


図 6-1：クロック供給図

動作

パワーアップ時には、コアは 6.5MHz のシステム・クロックで動作します。クロック供給レジスタに適切な値を設定することによって、クロック供給方法を設定できます。

クロック・マルチプレクサ

クロック供給図に示すように、クロック源マルチプレクサには次の 4 つがあります。

- ルート・クロック選択マルチプレクサ（ルート・クロック Mux）
- 低周波クロック・マルチプレクサ（LFCLK Mux）
- SPLL 入力マルチプレクサ（SPLL Mux）
- MMR を使用して制御される RCLK マルチプレクサ

汎用タイマー用のマルチプレクサは、タイマー・ブロック内のレジスタを介して制御されます。

注：クロック・マルチプレクサのクロックを選択する際は、希望するクロック入力が使用可能で安定していることを確認してください。そうでない場合、システムはクロックのロックが解除されてしまうことがあります。

RCLK Mux が SPLL 以外のクロックとして設定されている場合、SPLL を無効にする必要があります。

表 6-1：クロック・マルチプレクサ

クロック	レジスタ	選択肢
Root Clock Mux	CLKG_CLK_CTL0.CLKMUX	00：HFOSC 01：HFXTAL 10：SPLL 11：SYS_CLKIN GPIO を介した外部クロック
SPLL Mux	CLKG_CLK_CTL0.PLL_IPSEL	0：HFOSC 1：HFXTAL 2：SYS_CLKIN GPIO を介した外部クロック
LFCLK Mux	CLKG_OSC_CTL.LFCLK_MUX	0：LFOSC 1：LFXTAL
RCLK Mux	CLKG_CLK_CTL0.RCLKMUX	00：HFOSC 01：予備 10：HFXTAL 26 MHz 11：HFXTAL 16 MHz
GPT_CLK Mux Timer_RGB_Clk Mux	TMR_CTL.CLK CLKG_CLK_CTL5. TMRRGBCLKOFF	タイマー0／タイマー1／タイマー2／Timer_RGB による制御 00：PCLK（PDIV（PCLK 分周器）の出力） 01：HFOSC 26MHz（内部の高周波数発振器） 10：LFOSC 32kHz（内部の低周波数発振器） 11：LFXTAL 32kHz

クロック分周器

3 つのプログラマブルなクロック分周器を使用して、システム内のクロックを生成できます。クロック分周器は、入力クロックを新しいクロックに整数で分周します。分周範囲は、HCLK および PCLK 分周器では 1～32、ACLK 分周器では 1～511 です。分周の選択は随時行うことができます。出力はグリッチがなく、ハイ・タイムがストレッチされるため、クロック周期の前後の値よりもハイ・タイムが短くなることはありません。

2 つのクロック分周器はルート・クロックを入力として使用し、コアおよびペリフェラルに同期したクロックを生成します。クロック分周器は、各段がシーケンスの初期に分周クロックを順次開放するようにカスケード接続されています。最後の段は、他のすべての段に対し、分周したクロックを出力する準備ができたことを知らせます。このカスケード接続の効果により、新しい分周器の値を設定した際に、分周クロックが同期して出力されます。各クロックの初期エッジは相互に整列します。

クロック分周器の表に、各クロック分周器の入出力の概要を、それらを設定するためのレジスタ・ビットと併せて示しています。

表 6-2：クロック分周器

Divider	Input Clock	Output Clocks	Bit Field	Retained in Hibernate Mode
0	Root clock	HCLK_CORE, HCLK_BUS, FCLK	CLKG_CLK_CTL1.HCLKDIVCNT	Yes
1	Root clock	PCLK, all peripheral clocks	CLKG_CLK_CTL1.PCLKDIVCNT	Yes
2	Root clock	ACLK (for ADC)	CLKG_CLK_CTL1.ACLKDIVCNT	No

ADC クロックがクリーン・スタートを必要とするため、休止モードから復帰した後に CLKG_CLK_CTL1.ACLKDIVCNT フィールドを設定する必要があります（休止モードでは CLKG_CLK_CTL1.ACLKDIVCNT は保持されません）。また、ADC_CFG.EN ビットも休止モードで保持されないため、ADC を再度イネーブルする必要があります。

PCLK と HCLK の間では、特定の分周比のみが有効です。PCLK の周波数は、HCLK の周波数以下でなければなりません。また、分周器の比率は整数でなければなりません。

一般に、クロック分周は通常の動作中に随時変更できます。

クロック・ゲート

特定のクロックでは、電力モードまたはレジスタの設定に応じてクロックを個別にゲート制御できます。クロック・ゲートおよび電力モードの詳細については、パワー・マネージメント（PMG）を参照してください。

ペリフェラル・クロックのクロック・ゲートは、特定の電力モードでユーザ制御可能です。CLKG_CLK_CTL5 レジスタは、アプリケーションに応じて、特定のクロックをオフにするように設定できます。クロックをディスエーブルするには、CLKG_CLK_CTL5 レジスタの対応するビットを 1 に設定します。

PLL 設定

PLL 図は、SPLL の PLL 構造を概念的に示しています。乗算係数 N と除算係数 M によって出力クロック比 N/M が決定されます。PLL にはオプションの DIV2 が内蔵されているため、PLL 出力クロックを 2 分周するか、直接出力することができます。また、オプションの MUL2 も内蔵されているため、PLL のクロックを 2 倍にして PLL の範囲を拡大できます。

係数またはクロック源を変更するときは、常に PLL を ROOT_CLK から切り離さなければなりません。

位相周波数検出器（PFD）のリファレンス・クロックの入力が 2MHz の場合に、PLL の位相マージンが最良になります。したがって、26MHz の水晶発振器クロック入力の場合は M を 13 に、16MHz の水晶発振器クロック入力の場合は M を 8 に設定することを推奨します。

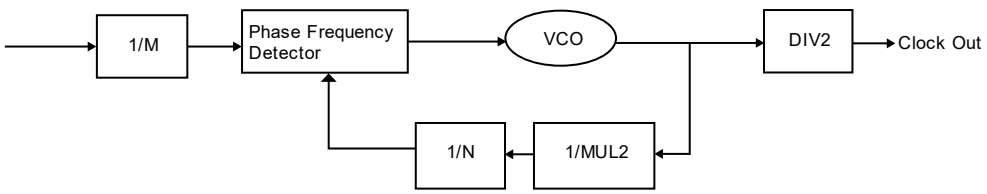


図 6-2：PLL 図

PLL 出力周波数 = クロック入力 *（MUL2*N）/（DIV2*M）

PLL 設定表には、様々な入出力のクロックに推奨される PLL の設定を示しています。PLL は、CLKG_CLK_CTL3 レジスタを使用して設定できます。

SPLL をイネーブルするには、CLKG_CLK_CTL3.SPLLEN ビットをセットする必要があります。SPLL の N 乗数は、CLKG_CLK_CTL3.SPLLNSEL ビットを使用して設定できます。

また、M 分周器は、CLKG_CLK_CTL3.SPLLMSEL ビットを使用して設定できます。SPLL MUL2 は、CLKG_CLK_CTL3.SPLLMUL2 ビットで制御されます。DIV2 は CLKG_CLK_CTL3.SPLLDIV2 ビットで制御されます。

MUL2 = CLKG_CLK_CTL3.SPLLMUL2 + 1、および、DIV2 = CLKG_CLK_CTL3.SPLLDIV2 + 1 です。

表 6-3：26MHz と 52MHz の PLL 設定

PLL	Input Clock (MHz)	M	MUL2	N	DIV2	Output Clock (MHz)
SPLL	26	13	1	26	2	26
SPLL	16	8	1	26	2	26
SPLL	26	13	1	26	1	52
SPLL	16	8	1	26	1	52

N の最小値は 8（MUL2 = 2 の場合）で、最大値は 31 です。1/M 分周器からの推奨出力周波数は 2MHz で、M の最小値は 2 です。

PLL 出力周波数 = クロック入力 *（（MUL2 * NSEL）/（DIV2 * MSEL））、

ここで、

MUL2 = CLKG_CLK_CTL3.SPLLMUL2 + 1

DIV2 = CLKG_CLK_CTL3.SPLLDIV2 + 1

DIV2 = CLKG_CLK_CTL3.SPLLNSEL

MSEL = CLKG_CLK_CTL3.SPLLMSEL

PLL 割込み

PLL は、ロックしたとき、またはロックを喪失したときにコアに割込みを発生させることができます。SPLL 割込みをイネーブルするには、CLKG_CLK_CTL3.SPLLIE ビットをセットする必要があります。CLKG_CLK_STAT0.SPLLUNLK ビットは、SPLL アンロック・イベントが発生したことを示します。

CLKG_CLK_STAT0.SPLLLK ビットは、SPLL ロック・イベントが発生したことを示します。どちらのビットも、PLL 割込みがイネーブルされているときにコアに割込みを発生させるために使用されます。どちらのビットもスティッキーであり、クリアするには 1 を書き込む必要があります。これらのビットは、SPLL ロック信号（1：ロック、0：アンロック）の値を反映する CLKG_CLK_STAT0.SPLL ビットとは異なります。

PLL 設定シーケンス

以下のシーケンスは、HFXTAL からの 26MHz の入力クロックを 26MHz の SPLL 出力に設定し、PLL 出力をルート・クロックとして使用する方法を示しています。HCLK は 26MHz、PCLK は 6.5MHz に設定されます。

1. CLKG_CLK_CTL3.SPLLIE を 0x1 に設定して、PLL 割込みをイネーブルにします。
2. CLKG_CLK_CTL0.PLL_IPSEL を 0x1 に設定して、HFXTAL を PLL の入力として設定します。
3. CLKG_OSC_CTL.HFXTALEN を 0x1 に設定して、HFXTAL をイネーブルにします。
4. クロック分周器の設定が目的のシステム・クロック・レートと一致するように、CLKG_CLK_CTL1.PCLKDIVCNT を 0x04 に、CLKG_CLK_CTL1.HCLKDIVCNT を 0x01 に設定します。
5. PLL をイネーブルにし、CLKG_CLK_CTL3.SPLLEN を 0x1 に、CLKG_CLK_CTL3.SPLLMSEL を 0xD に、CLKG_CLK_CTL3.SPLLNSEL を 0x1A に設定して PLL の M と N の値を設定します。
6. PLL がロックされたことを示す PLL 割込みを待ちます。水晶発振器が安定していない場合は PLL がロックしませんが、必要に応じてこの段階で水晶発振器が安定していることも確認します。
7. PLL 割込みをクリアし、CLKG_CLK_STAT0.SPLLLK を 0x1 に、CLKG_CLK_CTL0.CLKMUX を 0x2 に設定して、PLL をシステム・クロック源として選択します。

以下のシーケンスは、HFOSC からの 26MHz の入力クロックを 16MHz の SPLL 出力に設定し、PLL 出力をルート・クロックとして使用する方法を示しています。HCLKDIV と PCLKDIV を 0x1 に設定します。

1. CLKG_CLK_CTL3.SPLLIE を 0x1 に設定して、PLL 割込みをイネーブルにします。
2. CLKG_CLK_CTL0.PLL_IPSEL を 0x0 に設定して、HFOSC を PLL への入力として設定します。
3. CLKG_CLK_CTL3.SPLLEN を 0x0 に設定して、PLL をディスエーブルします。
4. 適切に MSEL と NSEL を設定して、PLL クロック出力を 16MHz に設定します。
   そのためには、CLKG_CLK_CTL3.SPLLMSEL を 0xD に、CLKG_CLK_CTL3.SPLLNSEL を 0x10 に設定します。
5. CLKG_CLK_CTL3.SPLLEN を 0x1 に設定して、PLL をイネーブルします。
6. PLL がロックされたことを示す PLL 割込みを待ちます。必要に応じて、この段階で水晶発振器が安定していることを確認します。
7. PLL 割込みをクリアし、CLKG_CLK_STAT0.SPLLLK を 0x1 に、CLKG_CLK_CTL0.CLKMUX を 0x2 に設定して、PLL をシステム・クロック源として選択します。

水晶発振器の設定

水晶発振器はデフォルトでディスエーブルされており、CLKG_OSC_CTL レジスタを使用して設定できます。水晶発振器は、CLKG_OSC_CTL.HFX_EN または CLKG_OSC_CTL.LFX_EN ビットをセットすることでイネーブルできます。安定信号のステータス・ビットもこのレジスタに反映されます。

注：SYSRESETREQ を発行する前に CLKG_OSC_CTL.HFOSC_EN ビットをセットし、Cortex-M4F がシステム・リセット・ジェネレータにリセット要求信号をアサートできるようにする必要があります。これにより、すべてのシステム・コンポーネントが正しくリセットされます。これは、ルート・クロック・マルチプレクサと SPLL クロック・マルチプレクサの設定とは無関係です。

割込み

それぞれの水晶発振器は、出力クロックが安定したときにコアに割込みを発生させることができます。この割込みは、

CLKG_CLK_CTL0.HFXTALIE および CLKG_CLK_CTL0.LFXTALIE ビットをセットすることでイネーブルされます。

CLKG_CLK_STAT0.HFXTAL および CLKG_CLK_STAT0.LFXTAL ビットに、水晶発振器の安定信号の現在の状態が入ります。CLKG_CLK_STAT0.HFXTALOK または CLKG_CLK_STAT0.LFXTALOK ビットは、水晶発振器の安定信号でイベントが検出されるとセットされます。

CLKG_CLK_STAT0.HFXTALOK/CLKG_CLK_STAT0.HFXTALNOK および CLKG_CLK_STAT0.LFXTALOK/CLKG_CLK_STAT0.LFXTALNOK ビットはスティッキーであり、1 に書き込むことによってクリアする必要があります。

注：CLKG_CLK_STAT0.HFXTALNOK および CLKG_CLK_STAT0.LFXTALNOK ビットは継続的な XTAL モニタではなく、対応する XTAL が適切にディスエーブルされたことを確認するためにのみセットされます。

PLL クロック保護

PLL へのクロック源が喪失した場合、PLL は約 3～5 ミリ秒の間、VCO 出力周波数を約 20MHz まで低下させて動作を維持しますが、このクロックは PLL がディスエーブルされるまでアクティブであり、作動し続けます。この動作により、CLKG_CLK_STAT0.SPLLUNLK などの PLL 割込み源が処理され、コアによって適切な処理が実行されます。この機能により、XTAL 回路のリードの断線または短絡によってコアが不確定な停止状態にならないように保護されます。

発振器の設定

両方の内部発振器はデフォルトでイネーブルされています。

CLKG_OSC_CTL.HFOSC_EN ビットをセットしてから SYSRESETREQ を発行して、Cortex-M4F がシステム・リセット・ジェネレータにリセット要求信号をアサートできるようにする必要があります。これにより、すべてのシステム・コンポーネントが正しくリセットされます。これは、ルート・クロック・マルチプレクサと SPLL クロック・マルチプレクサの設定とは無関係です。

HFOSC は、CLKG_OSC_CTL.HFOSC_EN ビットを使用してイネーブルします。

注：HFOSC の内部発振器を停止する前に、HFXTAL によってシステムが動作している必要があります。そうしないと、クロックが回復できずに停止するため、デバイスがロックされてしまいます。32kHz クロックで駆動されるペリフェラルは、LFXTAL が安定して動作していることを確認してから、LFXTAL 外付け水晶発振器に切り替える必要があります。LFOSC の内部発振器をディスエーブルすることはできません。

システム・クロックを 26MHz 以上に切り替える

システム周波数は、PLL クロック出力のみを使用して 52MHz にすることができます。

PLL 出力を 26MHz から 52MHz に変更する場合は、フラッシュのウェイト・ステートを 0 から 1 に変更する必要があります。ADC を駆動できる ACLK の最大周波数は 26MHz です。ルート・クロックが 52MHz の場合、周波数が最大許容周波数に制限されるように ACLK 周波数を変更します。

PCLK 周波数が 52MHz の場合、SPI、UART、SPORT、I2C の各モジュールのそれぞれのクロック設定レジスタも適切な値に設定します。また、ADC の PWRUP レジスタで指定する ADC パワーアップ・ウェイト・カウントがその PCLK 周波数に対して設定されていることも確認します。

システム・クロックが 26MHz より大きい場合、PMG_CTL1.HPBUCK_LD_MODE ビットを 1 に設定する必要があります。

以下の手順に、PLL クロック出力周波数を 26MHz から 52MHz に変更する方法を示します。

CLKG_CLK_CTL1.HCLKDIVCNT と CLKG_CLK_CTL1.PCLKDIVCNT が 1 分周に設定され、PLL 入力源が HFXTAL（26MHz）であるものと仮定します。

1. CLKG_CLK_CTL3.SPLLIE を 0x1 に設定して、PLL 割込みをイネーブルにします。
2. HFOSC（26MHz）を PLL クロックではなくルート・クロックとして選択します（CLKG_CLK_CTL0.CLKMUX = 0x0）。
3. CLKG_CLK_CTL3.SPLLEN を 0x0 に設定して、PLL をディスエーブルします。
4. PLL クロック出力を 52MHz にするには、CLKG_CLK_CTL3.SPLLMSEL = 0xD および CLKG_CLK_CTL3.SPLLNSEL = 0x1A に設定します。
5. CLKG_CLK_CTL3.SPLLMUL2 = 0x0 および CLKG_CLK_CTL3.SPLLDIV2 = 0x0 に設定します。
6. CLKG_CLK_CTL3.SPLLEN を 0x1 に設定して、PLL をイネーブルにします。
7. PLL がロックされたことを示す PLL 割込みを待ちます。必要に応じて、水晶発振器が安定していることを確認します。水晶が安定していない場合、PLL はロックしません。
8. PLL 割込みをクリアし、CLKG_CLK_STAT0.SPLLLK ビットを 0x1 に、CLKG_CLK_CTL0.CLKMUX ビットを 0x2 に設定して、システム・クロック源として PLL を選択します。

発振器

HF 発振器

26MHz の高周波発振器は、CLKG_OSC_CTL.HFOSC_EN ビットによってイネーブルします。分周器の値は 1、2、4、8、16、32 です。HFOSC 分周器の値は、CLKG_CLK_CTL2 レジスタを使用して選択できます。

選択した HFOSC 分周器の値が 1 または 2 の場合、RCLK 分周器が自動的に選択されて 13MHz クロックを生成します。HFOSC 分周器の値が 2 より大きい場合、RCLK 周波数は 13MHz 未満となり、この状態を持続します。フラッシュの消去と書込みの動作はできません。読出し動作は可能です。

CLKG_CLK_CTL2.HFOSCAUTODIV_EN ビットでイネーブルすることにより、ウェイクアップ時に 26MHz HFOSC クロックが自動的に選択されて、Flexi モードからの高速ウェイクアップが可能になります。

CLKG_CLK_CTL2.HFOSCDIVCLKSEL の値が更新された後 3µs の間は CLKG_CLK_CTL2 レジスタに書込みをしてはなりません。

Flexi モードからの高速ウェイクアップ

Flexi モードからの Cortex ウェイクアップ時に CLKG_CLK_CTL2.HFOSCAUTODIV_EN ビットが 1 に設定されている場合、1 の分周値が HF 発振器の分周器に自動的にロードされて、高速ウェイクアップを有効にできます。

1 クロックによる分周を選択すると、ウェイクアップ時に高速ウェイクアップを有効にするために、26MHz の HFOSC クロックが使用されます。この更新された 1 分周の値は、HFOSC 分周器の選択レジスタに新しい分周器の値が書き込まれるまで同じ値を維持します。

CLKG_CLK_CTL2.HFOSCAUTODIV_EN ビットが 0 の場合、この高速ウェイクアップ機能は無効になり、ウェイクアップ時に HFOSC 分周器レジスタは変更されません。

LF 発振器

32.768kHz 低周波発振器は常にイネーブルされ、ディスエーブルすることはできません。シャットダウン・モードでは自動的にディスエーブルされます。

HFXTAL 発振器

高周波発振器は CLKG_OSC_CTL.HFX_EN ビットでイネーブルにします。ロック時間は、XTAL がイネーブルされてから安定したクロックを出力するまでに要する時間です。発振器が正しい周波数にロックすると、CLKG_CLK_STAT0.HFXTALOK ステータス・ビットがセットされます。

LFXTAL 発振器

LFXTAL 発振器は RTC のクロック源です。これは、システムの時間を維持するのに使用されます。32.768kHz のクロックが出力されます。これは、CLKG_OSC_CTL.LFX_EN ビットをセットするとイネーブルにされます。

いったんこの発振器がイネーブルされると、オンを持続します（休止モードおよびシャットダウン・モードの場合）。

LFXTAL バイパス機能

LF 発振器はバイパスすることができ、SYS_LFXTAL_IN に外部クロックを供給すると LFXTAL クロックとして使用されます。この機能は、シャットダウン・モード以外のすべてのモードで使用できます。LFXTAL ロックは、供給された外部クロックに同期して自動的に生成されます。この機能は、シャットダウンに入る前に無効にする必要があります。

LFXTAL バイパス・モードのイネーブル

1. CLKG_OSC_KEY.VALUE にキーを書き込んでから CLKG_OSC_CTL.LFX_EN ビットをクリアすることで、LFXTAL をディスエーブルします。
2. CLKG_OSC_CTL.LFX_OK ビットがアサート解除されるのを待ちます。
3. CLKG_OSC_KEY.VALUE にキーを書き込み、CLKG_OSC_CTL.LFX_BYP ビットをセットします。
4. CLKG_OSC_CTL.LFX_OK ビットがアサート解除されるのを待ちます。外部クロックはこの段階で LFXTAL に接続されます。

LFXTAL バイパス・モードのディスエーブル

1. CLKG_OSC_KEY.VALUE にキーを書き込み、CLKG_OSC_CTL.LFX_BYP ビットをセットします。

2. CLKG_OSC_CTL.LFX_OK ビットがアサート解除されるのを待ちます。外部クロックが LFXTAL クロックから切り離されます。

LFXTAL クロックの不具合

LFXTAL クロックが動作しているかを監視し、クロックに不具合が発生したときに割込み／ウェイクアップを生成できます。LFXTAL クロックの不具合を監視し報告するには、LFOSC クロック（32kHz）を使用します。この機能は、1kHz 以上の LFXTAL クロック周波数でサポートされています。割込み／ウェイクアップ源は RTC1 割込み源で使用できます。

このクロック不具合は、アクティブ・ハイのステータス・ビット CLKG_OSC_CTL.LFX_FAIL_STA として報告されます。このステータス・ビットは、割込み源のアサートを解除するためにクリアする必要があります。

LFXTAL クロック不具合機能のイネーブル

1. 適切なキーを CLKG_OSC_KEY.VALUE に書き込みます。
2. CLKG_OSC_CTL.LFX_MON_EN ビットをセットします。
3. RTC1 の割込み源をイネーブルします。

LFXTAL クロック不具合機能のディスエーブル

1. 適切なキーを CLKG_OSC_KEY.VALUE に書き込みます。
2. CLKG_OSC_CTL.LFX_MON_EN ビットをクリアします。

ステータス・ビットのクリア

1. LFXTAL クロック不具合機能を無効にします。
2. キーを CLKG_OSC_KEY.VALUE に書き込みます。
3. CLKG_OSC_CTL.LFX_FAIL_STA ビットをセットしてステータスをクリアします。
4. CLKG_OSC_CTL.LFX_FAIL_STA ビットがクリアされるまで待ちます。ステータスと割込みの両方がクリアされます。

LFXTAL 不具合時での LFMUX の LFOSC への自動切替え

この機能は、LFMUX が LFXTAL クロックを使用するように設定されているときに、LFXTAL クロックの不具合時に LFMUX の出力クロックを LFOSC クロックに自動的に切り替える機能です。LFXTAL クロック・モニタ機能も有効にする必要があります。

自動切替え機能

1. CLKG_OSC_CTL.LFX_MON_EN = 1 および CLKG_OSC_CTL.LFX_AUTSW_STA = 1 に設定します。これにより、モニタ機能と自動切替え機能の両方が有効になります。
2. モニタ・ブロックは LFXTAL を追跡し、不具合が発生した場合にエラー割込みを発生させます。

3. CLKG_OSC_CTL.LFCLKMUX = LFXTAL かつ LFXTAL モニタ・エラーが発生すると、自動切替えロジックがトリガされ、LFMUX が LFOSC クロックに切り替わります。
4. LFOSC への切替えが完了すると、ハードウェアは CLKG_OSC_CTL.LFX_AUTSW_STA ビットをセットします。
5. LFMUX の元のクロック源が回復したら、元に戻すために、CLKG_OSC_CTL.LFX_FAIL_STA を 1 に設定します。これにより割込みがクリアされます
6. 割込みがクリアされたら、CLKG_OSC_CTL.LFX_AUTSW_STA に 1 を書き込んで元のクロック源に戻します。

ルート・クロックの不具合

ルート・クロックが動作しているかを監視し、クロックに不具合が発生したときに割込み／ウェイクアップを生成できます。HFOSC クロックを使用してルート・クロックの不具合を監視します。クロック不具合は、アクティブ・ハイのステータス・ビット CLKG_OSC_CTL.ROOT_FAIL_STA として報告されます。このステータス・ビットは、割込み源のアサートを解除するためにクリアする必要があります。

ルート・クロック不具合機能のイネーブル

1. 適切なキーを CLKG_OSC_KEY.VALUE に書き込みます。
2. CCLKG_OSC_CTL.ROOT_MON_EN ビットをセットします。
3. 割込み源 IRQ71 をイネーブルします。

LFXTAL クロック不具合機能のディスエーブル

1. 適切なキーを CLKG_OSC_KEY.VALUE に書き込みます。
2. CLKG_OSC_CTL.ROOT_MON_EN ビットをクリアします。

ステータス・ビットのクリア

1. ルート・クロック不具合機能を無効にします。
2. CLKG_OSC_KEY.VALUE レジスタにキーを書き込みます。
3. CLKG_OSC_CTL.ROOT_FAIL_STA ビットをセットしてステータスをクリアします。
4. CLKG_OSC_CTL.ROOT_FAIL_STA ビットがクリアされるまで待ちます。ステータスと割込みの両方がクリアされます。

ルート・クロック不具合時にルート CLKMUX を HFOSC に自動切替え

この機能は、ルート・クロックの不具合時に、ルート・クロックを HFOSC クロックに自動的に切り替える機能です。ルート・クロック・モニタ機能も有効にする必要があります。

自動切替え機能

1. CLKG_OSC_CTL.ROOT_MON_EN = 1 および CLKG_OSC_CTL.ROOT_AUTSW_EN = 1 に設定します。これにより、モニタ機能と自動切替え機能の両方が有効になります。

2. モニタ・ブロックはルート・クロックを追跡します。不具合が発生した場合は、ライン 71 でエラー割込みが発生します。
3. ルート・クロックのモニタ・エラーが発生すると、自動切替えロジックがトリガされ、ルート・クロックが HFOSC クロックに切り替わります。
4. HFOSC への切替えが完了すると、ハードウェアは CLKG_OSC_CTL.ROOT_AUTSW_STA ビットをセットします。
5. ルート・クロックの元のクロック源が回復したら、元に戻すために、CLKG_OSC_CTL.ROOT_FAIL_STA ビットを 1 に設定します。これにより割込みがクリアされます
6. 割込みがクリアされると、CLKG_OSC_CTL.ROOT_AUTSW_STA ビットに 1 を書き込んで元のクロック源に戻します。

システム・クロックの割込みと例外

詳細については、イベント（割込みと例外）を参照してください。

システム・クロックの設定

システム・クロックを PLL 入力源に設定

次のタイミング図は、システム・クロックを内部発振器から PLL 入力源ベースに変更するためのイベントのシーケンスを示しています。XTAL は、HFXTAL（26MHz または 16MHz 水晶発振器）を指します。タイマー（GPT）には次のような有効期間があります。

HFXTAL ロック・タイマーの有効期間：20 ミリ秒

LFXTAL ロック・タイマーの有効期間：1.5 秒

発振器が有効期間を超えてもロックされない場合は、XTAL に問題があることを示しています。

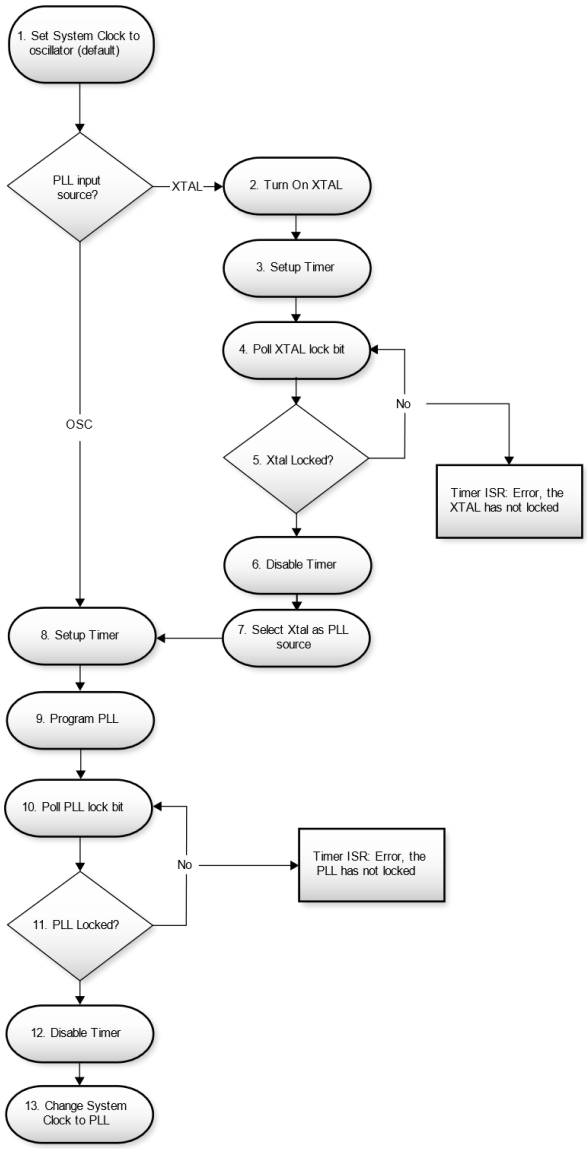


図 6-3：システム・クロックを PLL に変更（ポーリング方式）

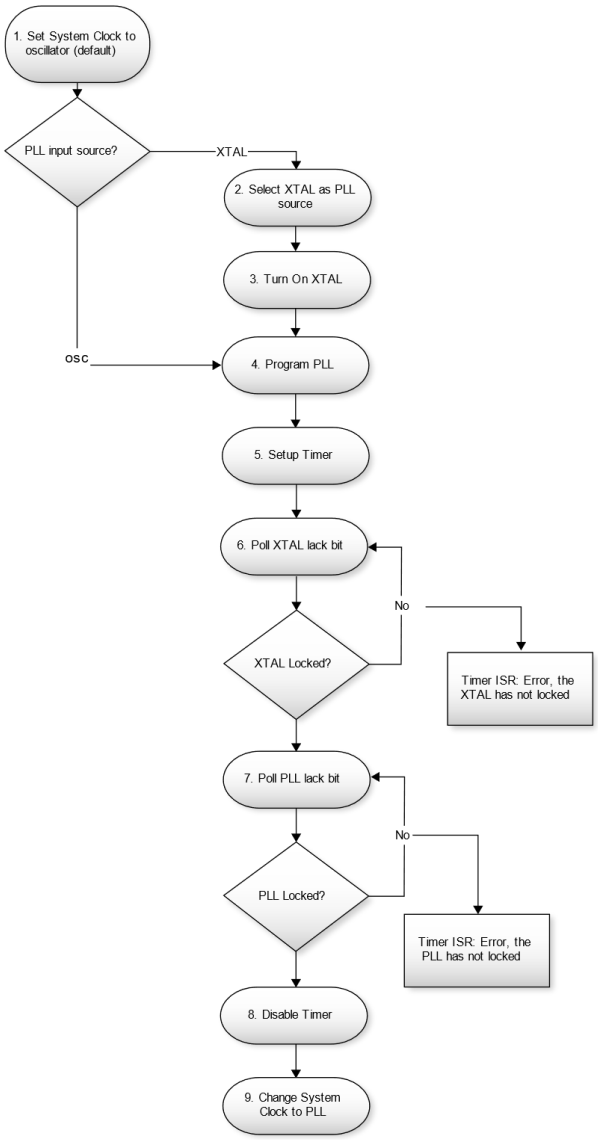


図 6-4：システム・クロックを PLL に変更（代替ポーリング方式）

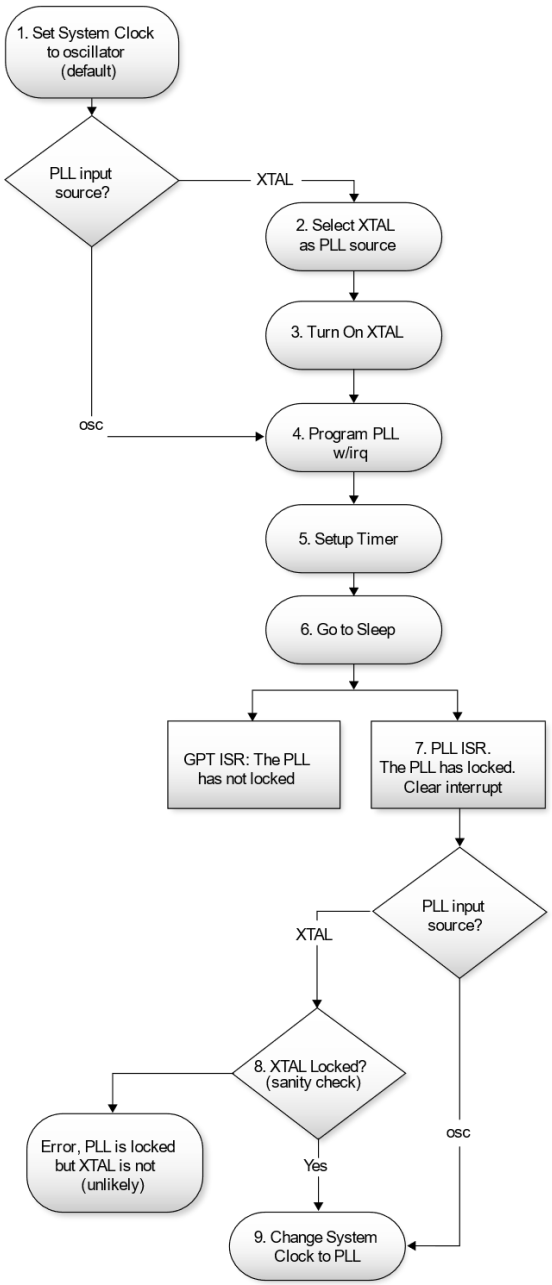


図 6-5：システム・クロックを PLL に変更（IRQ 方式）

システム・クロックを XTAL に設定

次の図は、内部発振器ベースから XTAL ベースのクロック源にシステム・クロックを変更するイベントのシーケンスを示しています。

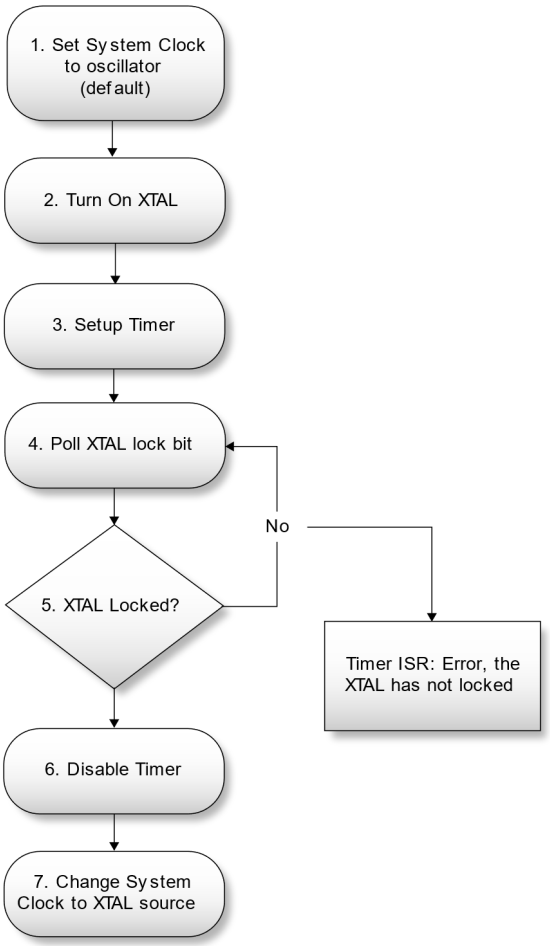


図 6-6：システム・クロックを XTAL に変更（ポーリング方式）

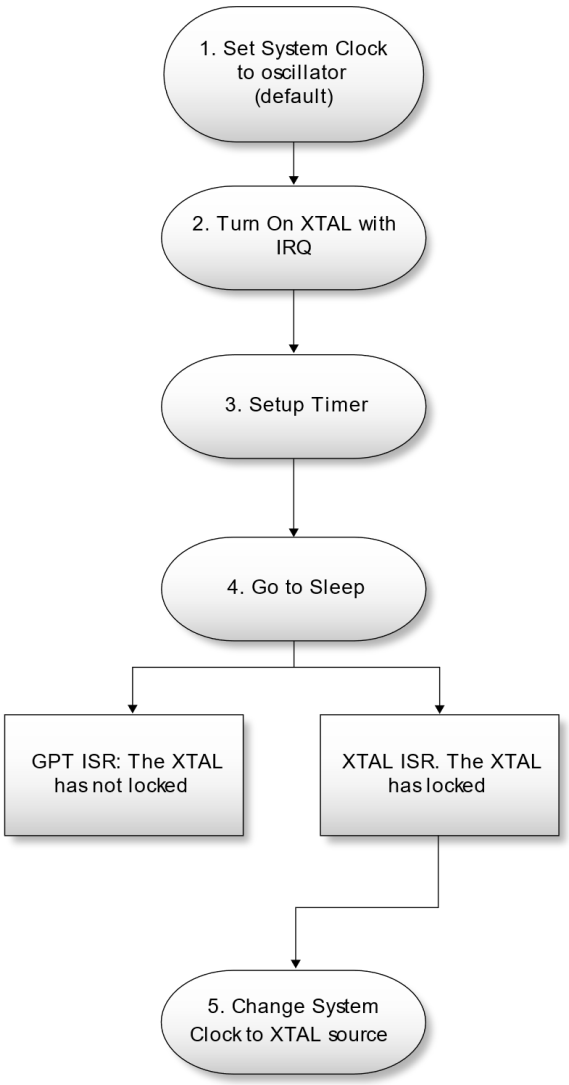


図 6-7：システム・クロックを XTAL に変更（IRQ 方式）

システム・クロック源の変更

この図は、システム・クロック源を発振器から XTAL 入力または PLL ベースの入力源に変更するシーケンスを示しています。

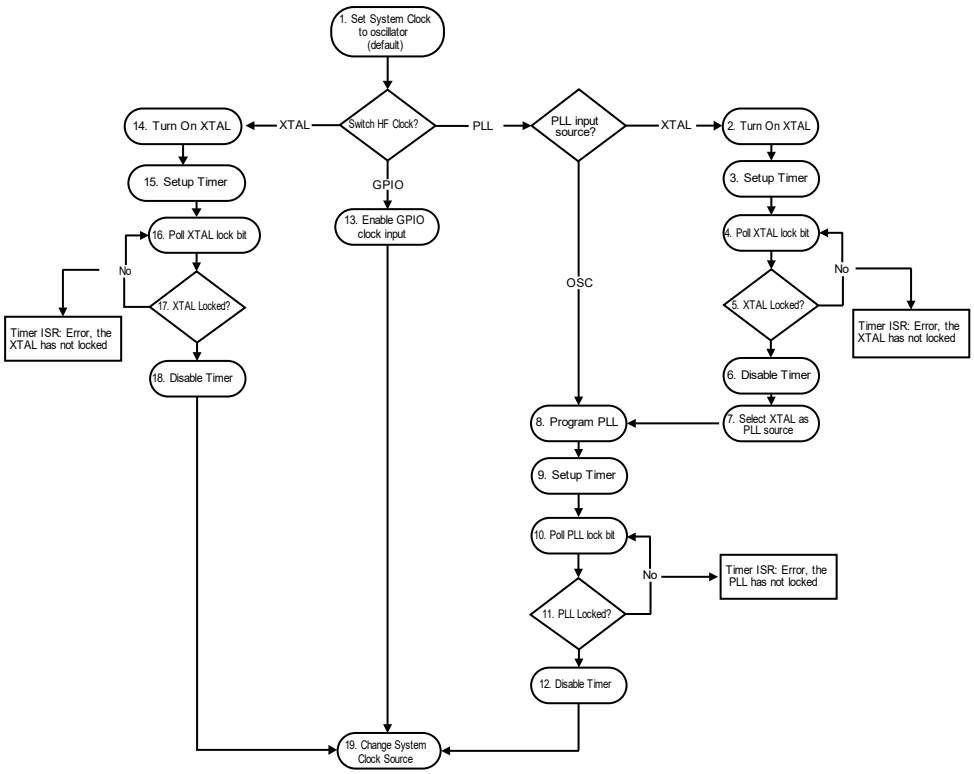


図 6-8：システム・クロックの変更

ADuCM4050 CLKG_OSC レジスタの説明

クロック供給レジスタ（CLKG_OSC）には、次のレジスタがあります。

表 6-4：ADuCM4050 CLKG_OSC レジスター一覧

レジスタ名	説明
CLKG_OSC_CTL	発振器制御
CLKG_OSC_KEY	OSCCTRL のキー保護

発振器制御

CLKG_OSC_CTL レジスタはキーで保護されています。この保護を解除するには、CLKG_OSC_CTL に書き込む前に CLKG_OSC_KEY に 0xCB14 を書き込む必要があります。CLKG_OSC_CTL に書き込む前に APB バスの他のレジスタに書き込むと、保護がロック状態に戻ります。

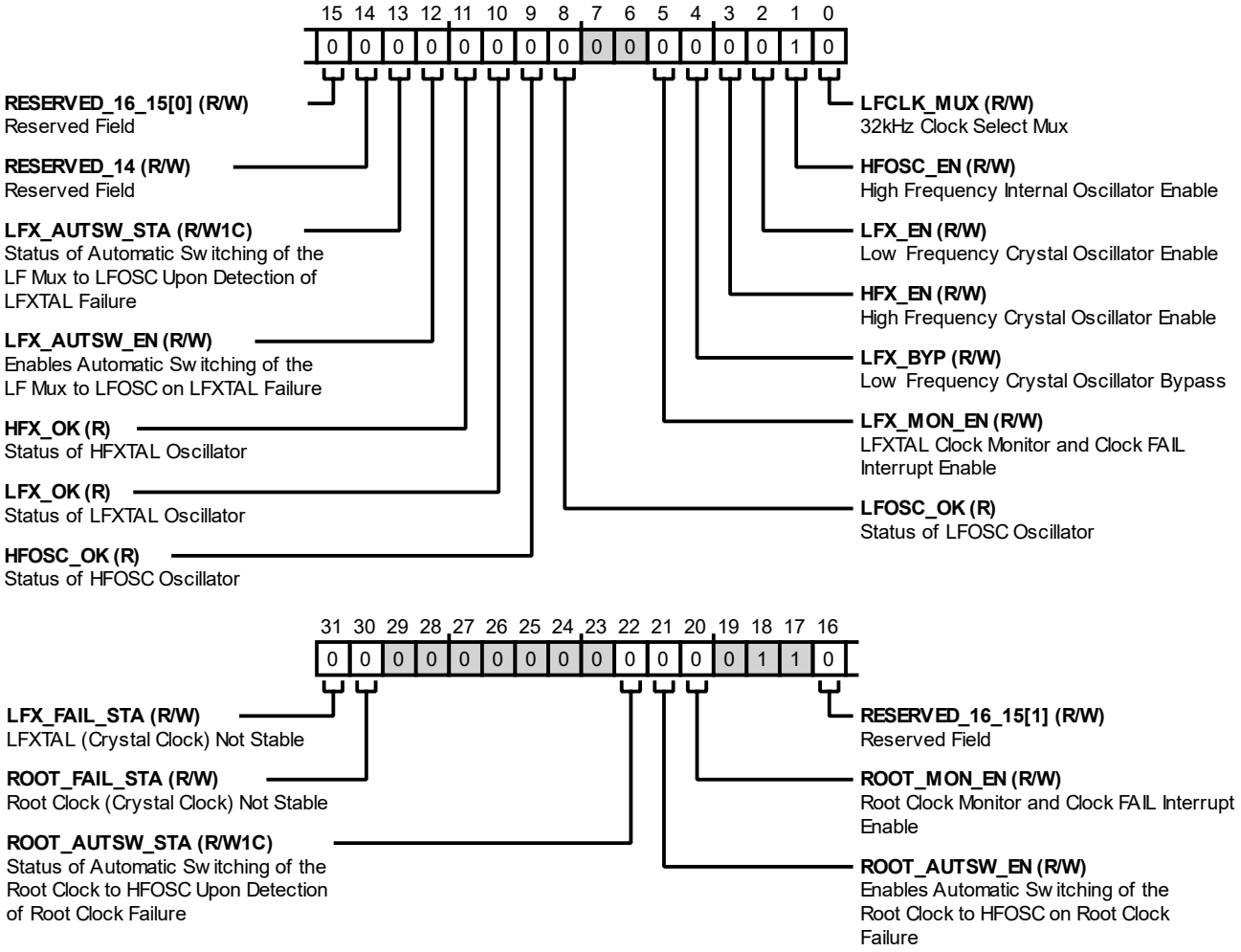


図 6-9：CLKG_OSC_CTL レジスタ図

表 6-5：CLKG_OSC_CTL レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
31 (R/W)	LFX_FAIL_STA	LFXTAL（水晶発振器クロック）が不安定。 これはスティッキー・ビットです。セットした場合、RTC1 IRQ ラインで割込みが発生します。割込みは、休止モードまたはアクティブ・モード時に生成できます。休止モードでこのフラグがセットされている場合、RTC1 IRQ ラインで割込みをアサートすることによって、デバイスは休止モードからウェイクアップします。 割込みとこのビットは、このビットに 1 を書き込むことでクリアできます。 1 - LFXTAL（低周波 XTAL クロック）が動作していません。ボードで水晶発振器からチップへの接続が切断している可能性があります。あるいは、外部クロック・ドライバがクロックの供給を停止しています。 0 - LFXTAL は正常に動作しています。	
30 (R/W)	ROOT_FAIL_STA	ルート・クロック（水晶発振器クロック）が不安定。 これはスティッキー・ビットです。セットした場合、IRQ 71 ラインで割込みが発生します。割込みは、アクティブ・モード時に生成できます。割込みとこのビットは、このビットに 1 を書き込むことでクリアできます。 1 - ルート・クロックが動作していません。 0 - ルート・クロックは正常に動作しています。	
22 (R/W1C)	ROOT_AUTSW_STA	ルート・クロック不具合の検出時における、ルート・クロックの HFOSC への自動切替えの状態。 これはスティッキー・ビットです。このビットは、ハードウェアがルート・クロック不具合の検出時にルート・クロックを HFOSC に自動的に切り替えたかどうかを示しています。このビットは、1 を書き込むことでクリアできます。	
		0	ルート・クロック不具合の検出時に、ハードウェアが自動的に HFOSC に切り替えなかったことを示します
		1	ルート・クロック不具合の検出時に、ハードウェアが自動的に HFOSC に切り替えたことを示します
21 (R/W)	ROOT_AUTSW_EN	ルート・クロック不具合の検出時での、ルート・クロックの HFOSC への自動切替えを有効化。 このビットは、ルート・クロック不具合時にルート・クロックの HF OSC への自動切替えを有効にするのに使用します。この機能は、ルート・クロック・モニタがイネーブルされている場合にのみ有効です。また、ルート・クロックから HF OSC クロックへの自動切替え機能のために、OSCCTRL レジスタの ROOTCLK_MON_EN ビットも 1 に設定する必要があります。	
		0	ルート・クロック不具合時における、ルート・クロックの HF OSC への自動切替えを無効にする
		1	ルート・クロック不具合時における、ルート・クロックの HF OSC への自動切替えを有効にする

表 6-5：CLKG_OSC_CTL レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
20 (R/W)	ROOT_MON_EN	ルート・クロック・モニタおよびクロック不具合割込みをイネーブル。 セットした場合、ルート・クロックは 26MHz の HF 発振器を使用して監視されます。このクロックの監視はアクティブ・モードで行われます。ルート・クロックが 2.5us の間トグルを停止すると、ROOTCLK_MON_FAIL フラグがセットされ、ルート・クロック不具合の IRQ ラインで割込みが発生します。 クリアした場合、モニタ回路はリセットされ、ルート・クロック不具合割込みは発生しません。	
		0	ルート・クロック・モニタおよびクロック不具合割込みをディスエーブル
		1	ルート・クロック・モニタおよびクロック不具合割込みをイネーブル
16:15 (R/W)	RESERVED_16_15	予備フィールド。 注：ADI カーネルは、これらのビット・フィールドに値 3 を書き込みます。これらのフィールドは予約されているため、変更しないでください。	
14 (R/W)	RESERVED_14	予備フィールド。 注：ADI カーネルは、このビットに値 1 を書き込みます。これらのフィールドは予約されているため、変更しないでください。	
13 (R/W1C)	LFX_AUTSW_STA	LFXTAL クロック不具合の検出時の LF Mux の LFOSC への自動切替えの状態。 このビットは、LFXTAL 不具合の検出時にハードウェアが自動的に LFOSC を使用するように切り替えられたかどうかを示します。	
		0	LFXTAL 不具合の検出時に、ハードウェアが自動的に LFOSC に切り替わらなかった
		1	LFXTAL 不具合の検出時に、ハードウェアが自動的に LFOSC に切り替わった
12 (R/W)	LFX_AUTSW_EN	LFXTAL 不具合時における LF Mux の LFOSC への自動切替えを有効化。 このビットは、LFXTAL の不具合時に LF MUX の LFOSC への自動切替えを有効にするために使用します。	
		0	LFXTAL クロック不具合時における、LF Mux の LFOSC への自動切替えを無効にする
		1	LFXTAL クロック不具合時における、LF Mux の LFOSC への自動切替えを有効にする
11 (R/NW)	HFX_OK	HFXTAL 発振器の状態。 このビットは、水晶発振器がイネーブルされた後に安定していることを示します。このビットはモニタではないため、その後に安定性を喪失してもそれを示すことはありません。	
		0	発振器がまだ安定していないか、ディスエーブルされている
		1	発振器はイネーブルされていて安定しており、使用可能

表 6-5：CLKG_OSC_CTL レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
10 (R/NW)	LFX_OK	LFXTAL 発振器の状態。 このビットは、水晶発振器がイネーブルされた後に安定していることを示します。このビットはモニタではないため、その後に安定性を喪失してもそれを示すことはありません。	
		0	発振器がまだ安定していないか、ディスエーブルされている
		1	発振器はイネーブルされていて安定しており、使用可能
9 (R/NW)	HFOSC_OK	HFOSC 発振器の状態。 このビットは、発振器がイネーブルされた後に安定していることを示します。このビットはモニタではないため、その後に安定性を喪失してもそれを示すことはありません。	
		0	発振器がまだ安定していないか、ディスエーブルされている
		1	発振器はイネーブルされていて安定しており、使用可能
8 (R/NW)	LFOSC_OK	LFOSC 発振器の状態。 このビットは、発振器がイネーブルされた後に安定していることを示します。このビットはモニタではないため、その後に安定性を喪失してもそれを示すことはありません。	
		0	発振器がまだ安定していないか、ディスエーブルされている
		1	発振器はイネーブルされていて安定しており、使用可能
5 (R/W)	LFX_MON_EN	LFXTAL クロック・モニタおよびクロック不具合割込み機能を有効化。 セットした場合、LFXTAL クロックはオンチップの 32kHz 低周波発振器を使用して監視されます。このクロックの監視は、休止モードとアクティブ／Flexi モードの両方で行われます。LFXTAL クロックが 2ms の間トグルを停止すると、LFXTAL_MON_FAIL フラグがセットされ、RTC1 IRQ ラインで割込みが発生します。 クリアした場合、モニタ回路はリセットされ、LFXTAL クロック不具合割込みは発生しません。	
		0	LFXTAL クロック・モニタおよびクロック不具合割込み機能を無効にする
		1	LFXTAL クロック・モニタおよびクロック不具合割込み機能を有効にする
4 (R/W)	LFX_BYP	低周波水晶発振器をバイパス。 このビットは、低周波水晶発振器をバイパスし、クロックを LFXTAL ピンの 1 つで外部から供給する場合に使用します。このビットを設定する前に、発振器をディスエーブルする必要があります。 LFXTAL をディスエーブルするにはしばらく時間がかかります。発振器がディスエーブルされたことを、LFXTALEN ビットを読み出し、0 であることによって確認します。	
		0	LFXTAL 発振器をディスエーブルし、低電力状態にする
		1	LFXTAL 発振器をイネーブルする

表 6-5：CLKG_OSC_CTL レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
3 (R/W)	HFX_EN	高周波水晶発振器をイネーブル。 このビットは HFXTAL 発振器をイネーブル／ディスエーブルするために使用します。この発振器は使用前に安定させておく必要があります。。	
		0	HFXTAL 発振器をディスエーブルし、低電力状態にする
		1	HFXTAL 発振器をイネーブルする
2 (R/W)	LFX_EN	低周波水晶発振器をイネーブル。 このビットは LFXTAL 発振器をイネーブル／ディスエーブルするために使用します。この発振器は使用前に安定させておく必要があります。	
		0	LFXTAL 発振器をディスエーブルし、低電力状態にする
		1	LFXTAL 発振器をイネーブルする
1 (R/W)	HFOSC_EN	高周波内部発振器をイネーブル。 このビットは HFOSC 発振器をイネーブル／ディスエーブルするために使用します。この発振器は使用前に安定させておく必要があります。このビットは、SYSRESETREQ システム・リセットを行う前に設定してください。	
		0	HFOSC 発振器をディスエーブルし、低電力状態にする
		1	HFOSC 発振器をイネーブルする
0 (R/W)	LFCLK_MUX	32kHz クロック選択マルチプレクサ。 このクロックはビーパ、RTC に接続されます。 注：ソフト・リセットがアサートされても、このビットはデフォルト値にリセットされません。他のビットはリセットされます。このビットは、パワーアップ、外部リセットなどの他のリセット時にリセットされます。ソフト・リセット後、ソフトウェアを実行するときに適切な値を設定します。ソフト・リセットは、Cortex レジスタの AIRCR.SYSRESETREQ ビットをセットすることによって生成されます。	
		0	内部 32kHz 発振器を選択
		1	外部 32kHz 水晶発振器を選択

OSCCTRL のキー保護

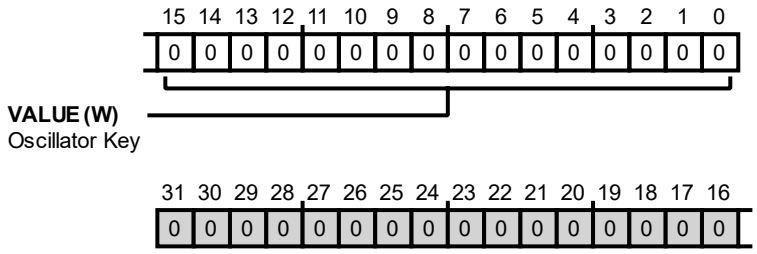


図 6-10：CLKG_OSC_KEY レジスタ図

表 6-6：CLKG_OSC_KEY レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (RX/W)	VALUE	発振器キー。 CLKG_OSC_CTL レジスタは 0xCB14 の値でキー保護されています。適切なキー値を入力した後、CLKG_OSC_CTL レジスタに書き込む必要があります。CLKG_OSC_CTL に書き込む前に APB バスの他のレジスタに書き込むと、保護がロック状態に戻ります。

ADuCM4050 CLKG_CLK レジスタの説明

クロック供給レジスタ（CLKG_CLK）には、次のレジスタがあります。

表 6-7：ADuCM4050 CLKG_CLK レジスター覧

レジスタ名	説明
CLKG_CLK_CTL0	その他のクロック設定
CLKG_CLK_CTL1	クロック分周器
CLKG_CLK_CTL2	HF 発振器の分周クロックの選択
CLKG_CLK_CTL3	システム PLL
CLKG_CLK_CTL5	ユーザ・クロック・ゲート制御
CLKG_CLK_STAT0	クロック・ステータス

その他のクロック設定

クロック・コントロール 0 は、コアやメモリ、ペリフェラルなどの様々なシステムで使用されるクロック源を設定するために使用します。すべての未使用ビットは読出し専用で、0 の値を返します。未使用ビットへの書込みは無効です。

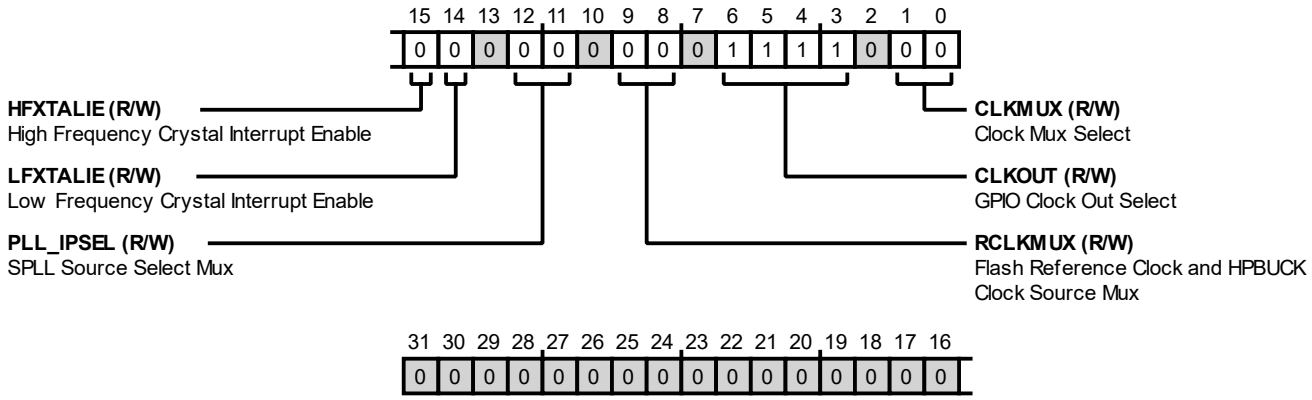


図 6-11：CLKG_CLK_CTL0 レジスタ図

表 6-8：CLKG_CLK_CTL0 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15 (R/W)	HFXTALIE	高周波水晶発振器をイネーブル。 CLKG_CLK_STAT0.HFXTALOK または CLKG_CLK_STAT0.HFXTALNOK ステータスでコアに割込みを発生させるかどうかを制御します。コアへの割込みが保留されている間は、このビットをクリアしないでください。	
		0	HFXTA-LOK または HFXTALNOK によってコアへの割込みを発生させない。
		1	HFXTALOK または HFXTALNOK によってコアへの割込みを発生させる。
14 (R/W)	LFXTALIE	低周波水晶発振器割込みをイネーブル。 CLKG_CLK_STAT0.LFXTALOK または CLKG_CLK_STAT0.LFXTALNOK ステータスでコアに割込みを発生させるか否かを制御します。コアへの割込みが保留されている間は、このビットをクリアしないでください。	
		0	LFXTALOK または LFXTALNOK によってコアへの割込みを発生させない。
		1	LFXTALOK または LFXTALNOK によってコアへの割込みを発生させる。

表 6-8：CLKG_CLK_CTL0 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
12:11 (R/W)	PLL_IPSEL	SPLL クロック源選択マルチプレクス。 CLKG_CLK_CTL0.PLL_IPSEL は、どのクロック源を SPLL に供給するかを選択します。SPLL をイネーブルする前に選択する必要があります。SPLL のイネーブル後は選択を変更しないでください。	
		0	内部 HF 発振器を選択
		1	外部 HF XTAL 発振器を選択
		2	GPIO_CLK 入力を選択
		3	予備
9:8 (R/W)	RCLKMUX	フラッシュ・リファレンス・クロックおよび HPBUCK クロックのクロック源マルチプレクス。 これらのビットは、外部水晶発振器と HF 発振器が共に安定して動作しているときに設定する必要があります。クロック・マルチプレクス出力は、フラッシュ・リファレンス・クロックと HP 降圧クロック（200kHz）を供給します。 オプション 11 を設定してクロック出力を 26MHz ではなく 16MHz とした場合、分周器で次のクロックが自動的に構成されます。 RCLK - 分周器はバイパスされる。RCLK = 16MHz HPBUCK クロック - 分周器は 200kHz クロックを生成するように自動的に構成される 注：外部 XTAL クロックが 26MHz ではなく 16MHz の場合、HP 降圧で重なり合わないフェーズが増加します。HP 降圧をイネーブルしている場合、これは推奨されません。	
		0	HFOSCCLK からクロックを供給（26MHz）。 HFOSC（26MHz）を HPBUCK 分周器およびフラッシュのリファレンス・クロック・ジェネレータのクロック源にする場合は、CLKG_CLK_CTL0.RCLKMUX = 00 に設定します。
		1	予備
		2	外部 HFXTAL からクロックを供給（26MHz）。 HFXTAL を HPBUCK 分周器およびフラッシュ・リファレンス・クロック・ジェネレータのクロック源とし、チップに接続する HFXTAL 周波数を 26MHz に設定する場合は、CLKG_CLK_CTL0.RCLKMUX = 10 に設定します。
		3	外部 HFXTAL からクロックを供給（16MHz）。 HFXTAL を HPBUCK 分周器およびフラッシュ・リファレンス・クロック・ジェネレータのクロック源とし、チップに接続する HFXTAL 周波数を 16MHz に設定する場合は、CLKG_CLK_CTL0.RCLKMUX = 11 に設定します。
6:3 (R/W)	CLKOUT	GPIO クロック出力の選択。 GPIO クロック出力ピンに接続するクロックを選択します。	
		0	ROOT_CLK

表 6-8：CLKG_CLK_CTL0 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
		1	LF_CLK
		2	ACLK
		3	HCLK_BUS
		4	HCLK_CORE
		5	PCLK
		6	RCLK（フラッシュ・コントローラ・タイマーのリファレンス・クロック）
		7	RHP_CLK（HF OSC、HF XTAL クロックのマルチプレクス）
		8	GPT0_CLK
		9	GPT1_CLK
		10	HCLK_P（HCLK で動作するペリフェラルに接続）
		11	PLL 出力クロック
		12	RTC0 1Hz 生成クロック
		13	HPBUCK_CLK
		14	HPBUCK_Non_overlap_clk
		15	RTC1 1Hz 生成クロック
1:0 (R/W)	CLKMUX	クロック・マルチプレクスの選択。 PCLK および HCLK の分周器で使用される共有クロック源を 1 つ決定します。イネーブルされているアクティブな安定クロック源を選択していることを確認してください（AON_CORE_MUX_SEL）。	
		0	高周波内部発振器を選択
		1	高周波外部水晶発振器を選択
		2	システム PLL を選択
		3	外部 GPIO ポートを選択

クロック分周器

クロック・コントロール 1 は、HCLK、PCLK、ACLK の各分周器の分周比を設定するために使用します。このレジスタにはいつでも書き込むことができます。すべての未使用ビットは読出し専用で、0 の値を返します。未使用ビットへの書込みは無効です。

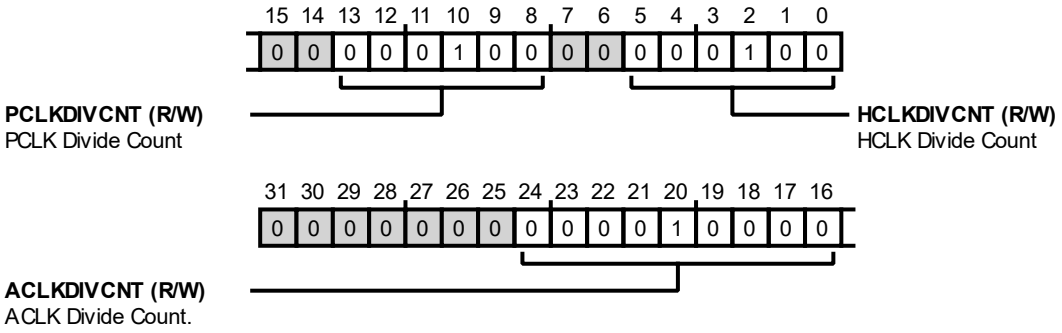


図 6-12：CLKG_CLK_CTL1 レジスタ図

表 6-9：CLKG_CLK_CTL1 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
24:16 (R/W)	ACLKDIVCNT	ACLK の分周カウント。 次式に基づいて ACLK レートを決定します。 ACLK = ROOT_CLK/CLKG_CLK_CTL1.ACLKDIVCNT 例えば、ROOT_CLK が 26MHz で、CLKG_CLK_CTL1.ACLKDIVCNT = 0x1 の場合、ACLK は 26MHz で動作します。 CLKG_CLK_CTL1.ACLKDIVCNT の値は、このレジスタへの書込みアクセス後に有効になり、通常は 1 サイクルの ACLK サイクルを必要とします。このレジスタはいつでも読み出すことができ、いつでも書き込むことができます。リセット分周器のカウントは 0x10 です。 値の範囲は 1～511 です。0 と 1 の値は、1 で分周した場合と同じ結果になります。このレジスタのデフォルト値は、ACLK = 1.625MHz となるように設定されています。設定可能な最大の ACLK 周波数は 26MHz です。
13:8 (R/W)	PCLKDIVCNT	PCLK の分周カウント。 次式に基づいて PCLK レートを決定します。 PCLK = ROOT_CLK/CLKG_CLK_CTL1.PCLKDIVCNT 例えば、ROOT_CLK が 26MHz で、CLKG_CLK_CTL1.PCLKDIVCNT = 0x2 の場合、PCLK は 13MHz で動作します。 CLKG_CLK_CTL1.PCLKDIVCNT の値はこのレジスタへの書込みアクセス後に有効になり、通常は 2～4 サイクルの ACLK サイクルを必要とします。このレジスタはいつでも読み出すことができ、いつでも書き込むことができます。リセット分周器のカウントは 0x4 です。 値の範囲は 1～32 です。32 を超える値は 32 に飽和します。0 と 1 の値は、1 で分周した場合と同じ結果になります。 このレジスタのデフォルト値は、PCLK が 6.5MHz になるように設定されています。

表 6-9：CLKG_CLK_CTL1 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応
5:0 (R/W)	HCLKDIVCNT	HCLK の分周カウント。 次式に基づいて HCLK レートを決定します。 HCLK = ROOT_CLK/CLKG_CLK_CTL1.HCLKDIVCNT 例えば、ROOT_CLK が 26MHz で、CLKG_CLK_CTL1.HCLKDIVCNT = 0x1 の場合、HCLK は 26MHz で動作します。 CLKG_CLK_CTL1.HCLKDIVCNT の値はこのレジスタへの書込みアクセス後に有効になり、通常は 2～4 サイクルの ACLK サイクルを必要とします。このレジスタはいつでも読み出すことができ、いつでも書き込むことができます。リセット分周器のカウントは 0x4 です。 値の範囲は 1～32 です。32 を超える値は 32 に飽和します。0 と 1 の値は、1 で分周した場合と同じ結果になります。 このレジスタのデフォルト値は、HCLK が 6.5MHz になるように設定されています。

HF 発振器の分周クロックの選択

クロック・コントロール 2 は、HF 発振器クロックから得られる周波数分周クロックからクロックを選択するのに使用します。すべての未使用ビットは読出し専用で、0 の値を返します。未使用ビットへの書込みは無効です。

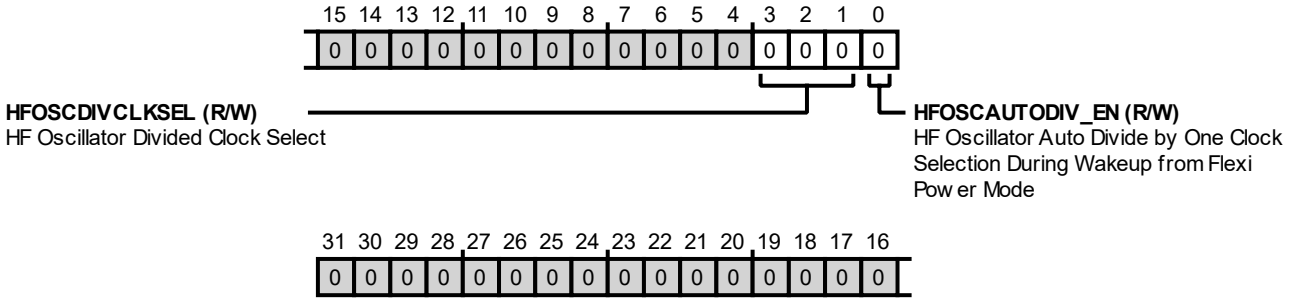


図 6-13：CLKG_CLK_CTL2 レジスタ図

表 6-10：CLKG_CLK_CTL2 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
3:1 (R/W)	HFOSCDIVCLKSEL	HF 発振器の分周クロックの選択。 HFOSC クロックから得られる周波数分周クロックから、クロックを選択するのに使用します。すべての未使用ビットは読出し専用で、0 の値を返します。未使用ビットへの書込みは無効です。 注：HFOSCDIVCLKSEL の値を 3'b010 から 3'b101 に設定すると、フラッシュ・リファレンス・クロック RCLK が 13MHz 未満に変更されます。	
		0	HFOSC/1 クロックを選択
		1	HFOSC/2 クロックを選択
		2	HFOSC/4 クロックを選択
		3	HFOSC/8 クロックを選択
		4	HFOSC/16 クロックを選択
		5	HFOSC/32 クロックを選択
		6	予備
		7	予備

表 6-10：CLKG_CLK_CTL2 レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
0 (R/W)	HFOSCAUTODIV_EN	Flexi 電力モードからのウェイクアップ時に、HF 発振器のクロックとして自動的に 1 分周を選択。 Flexi 電力モードからの高速ウェイクアップを有効にするために使用します。 1：Flexi モードからの高速ウェイクアップを有効にする 0：Flexi モードからの高速ウェイクアップを無効にする Flexi モードからの高速ウェイクアップが有効になると、ウェイクアップ時に未分周の 26MHz の HFOSC クロック周波数が使用されます。 HFOSCDIVCLKSEL ビットをクリアすることで HFOSC/1 クロックが選択され、未分周の HFOSC クロックが自動的に選択されます。この更新された 1 分周のクロック選択は、新しい分周器の値がこのレジスタに書き込まれるまで変更されません。 このビットが 0 の場合、高速ウェイクアップ機能は無効になり、ウェイクアップ時に HFOSCDIVCLKSEL ビットは変更されません。	
		0	Flexi モードからのウェイクアップ時の HFOSC/1 クロックの自動選択を無効にする
		1	Flexi モードからのウェイクアップ時の HFOSC/1 クロックの自動選択を有効にする

システム PLL

クロック・コントロール 3 は、システム PLL を制御するのに使用します。このレジスタへの書込みは、PLL をクロック源（ROOT_CLK）として選択していないときにのみ行う必要があります。すべての未使用ビットは読出し専用で、0 の値を返します。未使用ビットへの書込みは無効です。

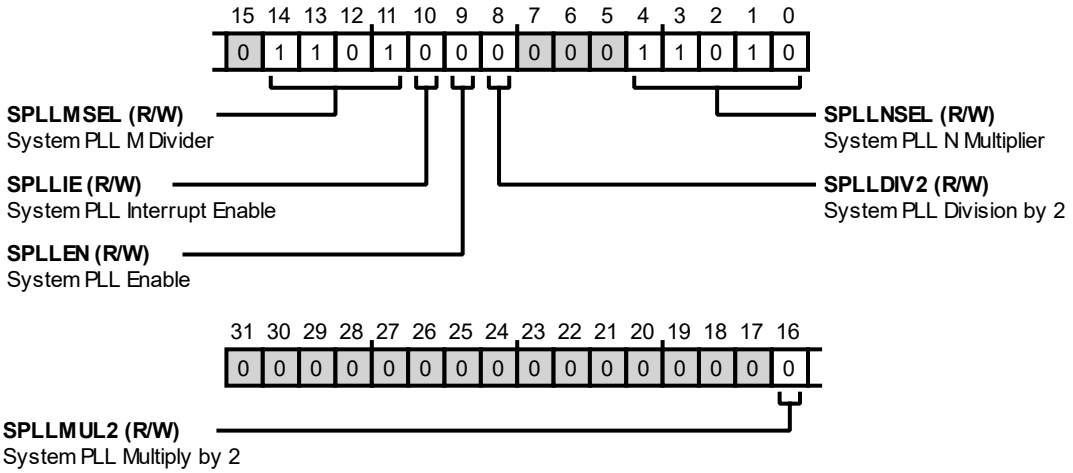


図 6-14：CLKG_CLK_CTL3 レジスタ図

表 6-11：CLKG_CLK_CTL3 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
16 (R/W)	SPLLMUL2	システム PLL を 2 倍にする。 VCO のクロック周波数を 2 倍するか 1 倍するかを設定します。	
		0	PLL 出力周波数 = SPLLNSEL * 1 /（（SPLLDIV2+1）* SPLLMSEL）
		1	PLL 出力周波数 = SPLLNSEL * 2 /（（SPLLDIV2+1）* SPLLMSEL）
14:11 (R/W)	SPLLMSEL	システム PLL の M 分周器。 システム PLL の M 分周器。PLL の倍率 N/M を導くのに使用する M 値を設定します。 CLKG_CLK_CTL3.SPLLMSEL <= 2 の場合、分周器 M の値= 2 です。 例えば 0000：M を 2 に設定する（2 で分周） 0001：M を 2 に設定する（2 で分周） 0010：M を 2 に設定する（2 で分周） 0011：M を 3 に設定する（3 で分周） 0100：M を 4 に設定する（4 で分周） 1111：M を 15 に設定する（15 で分周）	

表 6-11：CLKG_CLK_CTL3 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
10 (R/W)	SPLLIE	システム PLL 割込みをイネーブル。 PLL ロック／PLL アンロックでコアに割込みを発生させるか否かを制御します。コアへの割込みが保留されている間は、このビットをクリアしないでください。	
		0	PLL ロックまたは PLL アンロックでコアへの割込みを発生させない
		1	PLL ロックまたは PLL アンロックでコアへの割込みを発生させる
9 (R/W)	SPLLEN	システム PLL をイネーブル。 PLL をイネーブルするか、低電力状態にするかを制御します。このビットは、システム PLL がシステム・クロック源（CLKG_CLK_CTL0.CLKMUX）として選択されていないときにのみ設定する必要があります。	
		0	PLL をディスエーブルし、パワー・ダウン状態にする
		1	PLL をイネーブルする。最初のうち、PLL は選択した周波数で動作しません。安定化期間の後、PLL は選択した周波数にロックし、システム・クロック源（CLKMUX ビット）として選択できます。
8 (R/W)	SPLLDIV2	システム PLL を 2 で分周。 PLL 出力をオプションで 2 分周するかどうかを制御します。これにより、PLL 周波数（電力）が 2 倍になりますが、平衡出力のデューティ・サイクル出力が確保されます。CLKG_CLK_CTL3.SPLLEN をセットした後は、このビットを変更しないでください。このビットへは、CLKG_CLK_CTL3.SPLLEN のセットと同時に書き込むことができます。 このビットは、次の制約条件に基づいて設定またはリセットします。 1. VCO の出力範囲は 32MHz～60MHz です。 2. SPLL の出力範囲は 16MHz～60MHz です。 デフォルトで設定されています。	
		0	システム PLL を分周しない。その出力周波数は、N/M 比で選択した値に等しくなります。
		1	システム PLL を 2 で分周する。その出力周波数は、N/M 比によって選択した値を 2 分周した値に等しくなります。
4:0 (R/W)	SPLLNSEL	システム PLL の N 乗数。 PLL の乗算係数 N/M を導くのに使用する N 値を設定します。デフォルト値は 0b011010（26 倍）です。最小有効値は 8 で、8 未満の値を書き込むと強制的に 8 になります。有効な最大値は 31 です。SPLL に、16MHz を超える値または 60MHz を上回る値の出力クロックを設定しないでください。	

ユーザ・クロック・ゲート制御

クロック・コントロール 5 は、ペリフェラル UCLK のゲートを制御するのに使用します。

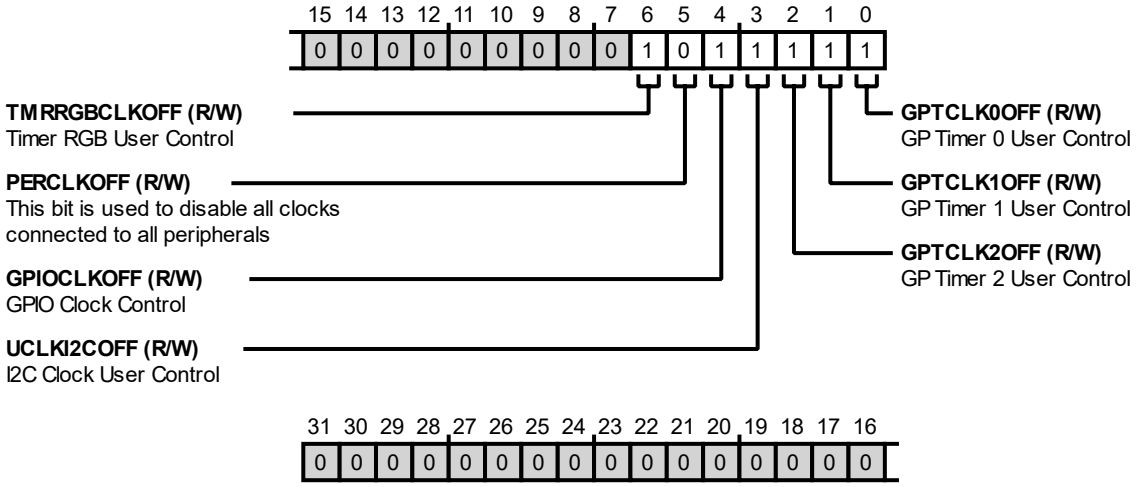


図 6-15：CLKG_CLK_CTL5 レジスタ図

表 6-12：CLKG_CLK_CTL5 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
6 (R/W)	TMRRGBCLKOFF	タイマーRGB ユーザ・コントロール。 このビットは、タイマーRGB（マルチプレクス・バージョン）をディスエーブルします。これは、アクティブ・モードおよび Flexi 電力モードで、タイマーGB クロックのゲートを制御します。休止モードとシャットダウン・モードでは、タイマーRGB クロックは常にオフにされるので、このビットは無効です。 注：タイマーRGB に APB バスを介してアクセスする場合、このビットは自動的にクリアされます。	
		0	TMRRGB クロックをイネーブル
		1	TMRRGB クロックをディスエーブル

表 6-12：CLKG_CLK_CTL5 レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
5 (R/W)	PERCLKOFF	このビットは、すべてのペリフェラルに接続されたすべてのクロックをディスエーブルする場合に使用します。 これは、以下の場合に電力を節約するのに便利です。 (i) プロセッサが、一定時間ペリフェラルと連携動作しない。 (ii) SRAM から SRAM へのコピーやフラッシュから SRAM へのコピー（またはその逆）のような、DMA トランザクション。 このビットをセットすると、GPIO、フラッシュ、DMA、Cortex、SRAM の各ブロックがアクティブになります。 電源モニタ、クロック源モニタのシステム・ステータス・レジスタ： 0：ペリフェラルへのクロックをアクティブにする。 1：次のペリフェラルへのクロックをゲート・オフする。SPI、SPORT、CRC、AES、SPI、I2C、UART、タイマー、APB16 ブリッジ、PCLK 分周器、PM、CLK レジスタ・マップ。 このビットをセットした場合、上記のモジュール・レジスタへ読出し／書込みを行うと、自動的に PERCLKOFF が 0 にリセットされ、その読出し／書込みトランザクションが有効になります。 唯一の例外は、このビットを 1 に設定した後、CLKG_CLK_CTL5 レジスタの読出しが可能になることです。この場合、このビットは自動的にクリアされません。 PERCLKOFF ビットをセットする前に推奨される使用法： 1. DMA トランザクションが完了し、DMA からのトランザクションはそれ以上ないと思われることを確認します。 2. このビット書込みが最後の書込みであることを確認します。また、このビットをセットした後、上記のモジュール・レジスタへの書込み／読出しが行われないことを確認します。それ以外の場合、このビットはクリアされます。	
		0	ペリフェラルへのクロックをアクティブにする
		1	次のペリフェラルへのクロックをゲート・オフする。SPI、SPORT、CRC、AES、I2C、UART、タイマー、APB16 ブリッジ、PCLK 分周器、PM、CLK レジスタ・マップ。
4 (R/W)	GPIOCLKOFF	GPIO クロック・コントロール。 このビットは、GPIO ブロックへのクロックをディスエーブルします。この制御は、Active モードと Flexi モードに適用されます。休止モードとシャットダウン・モードでは、GPIO へのクロックは常にオフにされるため、このビットは無効です。 注：GPIO レジスタが APB バスを介してアクセスされると、このビットは自動的にクリアされて、GPIO ブロックへクロックを供給します。	
		0	GPIO クロック入力をイネーブル
		1	GPIO クロック入力をディスエーブル

表 6-12：CLKG_CLK_CTL5 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
3 (R/W)	UCLKI2COFF	I2C クロック・ユーザ・コントロール。 このビットは、I2C UCLK をディスエーブルします。アクティブ・モードと Flexi モードで I2C UCLK のゲートを制御します。休止モードとシャットダウン・モードでは、I2C UCLK は常にオフにされるため、このビットは無効です。 注：I2C が APB バスを介してアクセスされると、このビットは自動的にクリアされます。	
		0	I2C クロックをイネーブル
		1	I2C クロックをディスエーブル
2 (R/W)	GPTCLK2OFF	GP タイマー2 ユーザ・コントロール。 このビットは、gptclk2（マルチプレクス・バージョン）をディスエーブルします。これは、アクティブ・モードおよび Flexi モードで GP タイマー2 クロックのゲートを制御します。休止モードとシャットダウン・モードでは、GPT クロックは常にオフにされるため、このビットは無効です。 注：GPT2 が APB バスを介してアクセスされると、このビットは自動的にクリアされます。	
		0	TMR2 クロックをイネーブル
		1	TMR2 クロックをディスエーブル
1 (R/W)	GPTCLK1OFF	GP タイマー1 ユーザ・コントロール。 このビットは、gptclk1（マルチプレクス・バージョン）をディスエーブルします。これは、アクティブ・モードおよび Flexi モードで GP タイマー1 クロックのゲートを制御します。休止モードとシャットダウン・モードでは、GPT クロックは常にオフにされるため、このビットは無効です。 注：GPT1 が APB バスを介してアクセスされると、このビットは自動的にクリアされます。	
		0	TMR1 クロックをイネーブル
		1	TMR1 クロックをディスエーブル
0 (R/W)	GPTCLK0OFF	GP タイマー0 ユーザ・コントロール。 このビットは、gptclk0（マルチプレクス・バージョン）をディスエーブルします。これは、アクティブ・モードおよび Flexi モードで GP タイマー0 クロックのゲートを制御します。休止モードとシャットダウン・モードでは、GPT クロックは常にオフにされるため、このビットは無効です。 注：GPT0 が APB バスを介してアクセスされると、このビットは自動的にクリアされます。	
		0	TMR0 クロックをイネーブル
		1	TMR0 クロックをディスエーブル

クロック・ステータス

クロック・ステータスは、PLL と発振器の状態を監視するのに使用します。割込みをイネーブルすると、クロック・コンポーネントが安定するまでの間、初期化コードの実行を継続するか、コアをアイドル状態にすることができます。

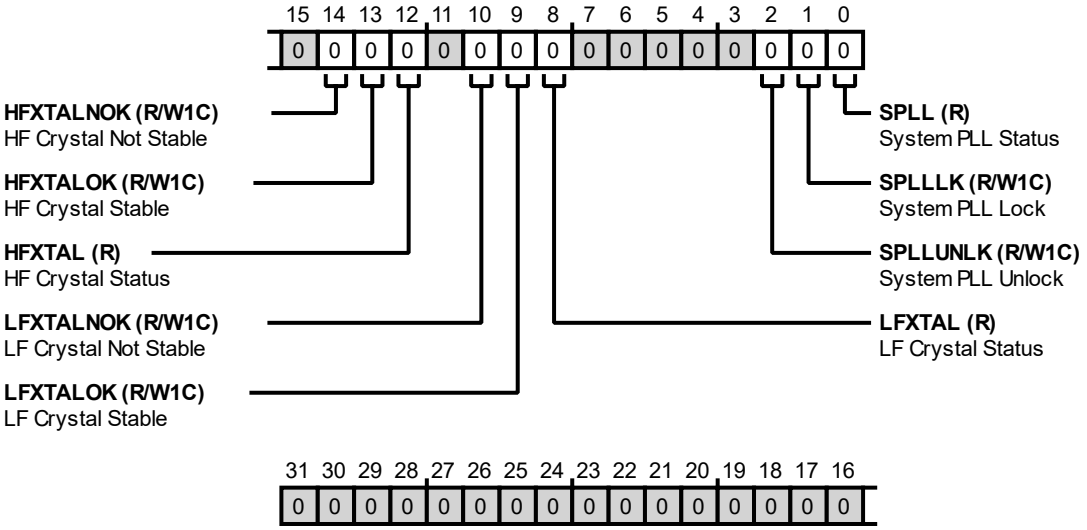


図 6-16：CLKG_CLK_STAT0 レジスタ図

表 6-13：CLKG_CLK_STAT0 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
14 (R/W1C)	HFXTALNOK	HF 水晶発振器が不安定。 このビットは、XTAL が正常にディスエーブルされたことを示します。このビットは XTAL を継続的に監視するものではなく、また XTAL が不安定になったイベントでセットされるわけでもありません。このビットはスティッキーです。クリアするには、この位置に 1 を書き込みます。イネーブルにした場合、このビットに割込みを関連付けることができます。	
		0	HF 水晶発振器の安定信号はアサート解除されていない
		1	HF 水晶発振器の安定信号はアサート解除されている
13 (R/W1C)	HFXTALOK	HF 水晶発振器が安定。 このビットはスティッキーです。これは、割込みがイネーブルされたときにコアに割込みを発生させる場合に使用します。クリアするには、この位置に 1 を書き込みます。	
		0	HF 水晶発振器の安定信号はアサートされていない
		1	HF 水晶発振器の安定信号はアサートされている

表 6-13：CLKG_CLK_STAT0 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
12 (R/NW)	HFXTAL	HF 水晶発振器のステータス。 このビットは、XTAL が安定し、使用できるようになったことを判断するのに役立ちます。このビットは機能を継続的に監視するものではなく、また XTAL が不安定になったイベントでクリアされるわけでもありません。	
		0	HF 水晶発振器が不安定か、イネーブルされていない
		1	HF 水晶発振器は安定状態
10 (R/W1C)	LFXTALNOK	LF 水晶発振器が不安定。 このビットは、XTAL が正常にディスエーブルされたことを示します。このビットは XTAL を継続的に監視するものではなく、また XTAL が不安定になったイベントでセットされるわけでもありません。このビットはスティッキーです。クリアするには、この位置に 1 を書き込みます。イネーブルにした場合、このビットに割込みを関連付けることができます。	
		0	LF 水晶発振器の安定信号はアサート解除されていない
		1	LF 水晶発振器の安定信号はアサート解除されている
9 (R/W1C)	LFXTALOK	LF 水晶発振器は安定。 このビットはスティッキーです。これは、割込みがイネーブルされたときにコアに割込みを発生させる場合に使用します。クリアするには、この位置に 1 を書き込みます。	
		0	LF 水晶発振器の安定信号はアサートされていない。
		1	LF 水晶発振器の安定信号はアサートされている。
8 (R/NW)	LFXTAL	LF 水晶発振器のステータス。 このビットは、XTAL が安定し、使用できるようになったことを判断するのに役立ちます。このビットは機能を継続的に監視するものではなく、また XTAL が不安定になったイベントでクリアされるわけでもありません。	
		0	LF 水晶発振器が不安定か、イネーブルされていない。
		1	LF 水晶発振器は安定状態。
2 (R/W1C)	SPLLUNLK	システム PLL のアンロック。 このビットはスティッキーです。PLL がロックを喪失すると、CLKG_CLK_STAT0.SPLLUNLK がセットされます。CLKG_CLK_STAT0.SPLLUNLK は、ロックが喪失されたことをコアに通知するための割込み源として使用します。このビットに 1 を書き込むとクリアされます。 CLKG_CLK_STAT0.SPLLUNLK は、システム PLL がロックを獲得し、その後再び喪失しない限り、再びセットされることはありません。	
		0	PLL のロック喪失は検出されていない
		1	PLL のロック喪失が検出された

表 6-13：CLKG_CLK_STAT0 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
1 (R/W1C)	SPLLLK	システム PLL のロック。 このビットはスティッキーです。PLL がロックすると CLKG_CLK_STAT0.SPLLLK がセットされます。CLKG_CLK_STAT0.SPLLLK は、ロックが検出されたことをコアに通知する割込み源として使用します。このビットに 1 を書き込むとクリアされます。CLKG_CLK_STAT0.SPLLLK は、システム PLL がロックを喪失し、その後再びロックしない限り、再びセットされることはありません。	
		0	PLL ロック・イベントは検出されていない
		1	PLL ロック・イベントが検出された
0 (R/NW)	SPLL	システム PLL のステータス。 PLL の現在の状態を示します。最初はシステム PLL のロックは解除されます。安定化期間の後、PLL はロックされ、システム・クロック源として使用できる状態になります。これは読出し専用ビットです。書込みは無効です。	
		0	PLL がロックしていないか、正しく構成されていない PLL は、システム・クロック源として使用する準備ができていない。
		1	PLL はロックし、システム・クロック源として使用できる状態にある。

7 リセット（RST）

ADuCM4050 MCU は次のリセット機能を備えています。

- 外部
- パワー・オン
- ウォッチドッグ・タイムアウト
- ソフトウェア・システム

ソフトウェア・システム・リセットは、Cortex-M4F コアの一部として提供されます。

ソフトウェアによりシステム・リセットを生成するには、アプリケーション割込み／リセット制御レジスタを 0x05FA0004 の値に設定する必要があります。このレジスタは NVIC サブシステムの一部であり、アドレス 0xE000ED0C にあります。

ソフトウェア・リセットの詳細については、**ARM Cortex-M4F テクニカル・リファレンス・マニュアル**を参照してください。

ハードウェア・リセットは、アクティブ・ローの SYS_HWRST ピンをトグルすることによって実行されます。

PMG_RST_STAT レジスタは、最後のリセットのソースを示します。このレジスタは、リセット例外サービス・ルーチンの中でリセットのソースを識別するのに使用できます。

表 7-1：リセット・タイプ

Reset	Reset External Pins to Default State	Execute Kernel	Reset All MMRs Except PMG_RST_STAT	Reset All Peripherals	Valid SRAM	PMG_RST_STAT After Reset Event
Software	Yes[*1]	Yes	Yes	Yes	Yes/no[*2]	PMG_RST_STAT.SWRST = 1
Watchdog	Yes	Yes	Yes	Yes	Yes/no[*2]	PMG_RST_STAT.WDRST = 1
External reset pin	Yes	Yes	Yes	Yes	Yes/no[*2]	PMG_RST_STAT.EXTRST = 1
POR	Yes	Yes	Yes	Yes	No	PMG_RST_STAT.POR = 1 PMG_RST_STAT.PORSRC has info on cause of POR reset

*1　GPIOx はデフォルト状態、つまり POR イベントと同じ状態に戻ります。

*2　UART のダウンロード後にリセットされた場合、RAM は無効です。

ADuCM4050 リセット・レジスタの説明

表 7-2：ADuCM4050 リセット・レジスター一覧

レジスタ名	説明	リセット	アクセス
PMG_RST_STAT	リセット・ステータス	0x000000XX	R/W

リセット・ステータス

このレジスタは、リセットの原因を判断するためにユーザ・コードの先頭で読み取ることを推奨します。リセットの原因は様々なので、このレジスタのデフォルト値は指定されていません。

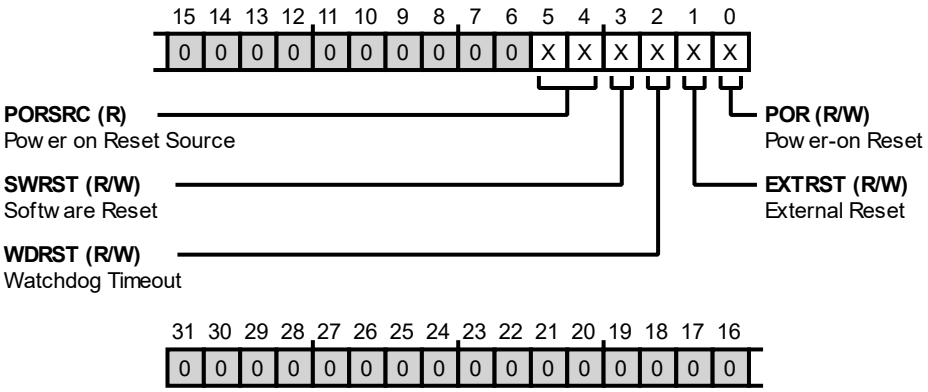


図 7-1：PMG_RST_STAT レジスタ図

表 7-3：PMG_RST_STAT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
5:4 (R/NW)	PORSRC	パワーオン・リセット・ソース。 このビットには、パワーオン・リセットが発生した後の詳細が含まれています。	
		0	VBAT がフェイルセーフを下回ったため POR がトリガされた
		1	VBAT 電源（VBAT < 1.7V）が原因で POR がトリガされた
		2	VDD 電源（VDD < 1.08V）が原因で POR がトリガされた
		3	VREG がフェイルセーフを下回ったため POR がトリガされた
3 (R/W)	SWRST	ソフトウェア・リセット。 システムのリセットが生成されると、自動的に 1 に設定されます。このビットに 1 を書き込むことでクリアされます。このビットは、パワーオン・リセットがトリガされたときにもクリアされます。	
2 (R/W)	WDRST	ウォッチドッグ・タイムアウト・リセット。 ウォッチドッグ・タイムアウトが発生すると、自動的に 1 に設定されます。このビットに 1 を書き込むことでクリアされます。このビットは、パワーオン・リセットがトリガされたときにもクリアされます。	
1 (R/W)	EXTRST	外部リセット。 外部リセットが発生すると、自動的に 1 に設定されます。このビットに 1 を書き込むことでクリアされます。このビットは、パワーオン・リセットがトリガされたときにもクリアされます。	
0 (R/W)	POR	パワーオン・リセット。 パワーオン・リセットが発生すると、自動的にセットされます。ビットに 1 を書き込むことでクリアされます。	

8 フラッシュ・コントローラ（FLASH）

ADuCM4050 MCU は 512kB の組み込みフラッシュ・メモリを内蔵しており、フラッシュ・コントローラを介してアクセスすることができます。組み込みフラッシュには 72 ビット幅のデータ・バスがあり、1 回のアクセスで 2 個の 32 ビット・ワードのデータと、それに対応する 8 ビットの ECC を送ることができます。

フラッシュ・コントローラはキャッシュ・コントローラ・モジュールと組み合わされており、これには次の 2 つの AHB ポートが備わっています。

- データ読出し用の DCode
- 命令読出し用の ICode

ICode の読出し性能を最適化するため、フラッシュ・コントローラにはプリフェッチ機構が実装されています。

フラッシュ書込みは、メモリ・マップド・レジスタ（MMR）への APB 書込みによるキーホール機構でサポートされます。フラッシュ・コントローラは、DMA ベースのキーホール書込みもサポートしています。

フラッシュ機能

ADuCM4050 MCU が使用するフラッシュ・メモリは、以下の機能を備えています。

- ICode インターフェースでの連続アドレスの読出し時、プリフェッチ・バッファにより性能が最大化。
- ICode と DCode の同時読出しアクセス（競合発生時は DCode が優先。ICode がプリフェッチからバッファ・データを返す場合は同時読出しが可能）。
- DMA ベースのキーホール書込み（シーケンシャル・アクセス用のアドレス自動インクリメントを含む）。
- エラーの訂正や検出のための ECC。エラーと訂正は、バス・エラー（ICode/DCode バス上のバス・エラー）または割込みとしてレポートされるか、無視されます。

サポートされているコマンド

- 読出し：ICode または DCode インターフェースを通じてサポート。
- 書込み：メモリ・マップ・レジスタを通じたキーホール機構により提供。
- 一括消去：すべてのユーザ・データとプログラム・コードを消去。
- ページ消去：フラッシュ内の 2KB ページからユーザ・データやプログラム・コードを消去。

- シグネチャ：任意の連続ページ内にあるすべてのユーザ・データやプログラム・コードのためのシグネチャを生成して検証。
- アボート：コマンドの実行を中止。

保護と完全性に関する機能

- 一括消去やページ消去などの保護されたコマンドを実行するには、固定のユーザ・キーが必要。
- アクセス可能なメモリに対するユーザ定義の書込み保護（オプション）。
- 8 ビット・エラー訂正符号（ECC）。

フラッシュ機能の説明

ここでは、ADuCM4050 MCU が使用するフラッシュ・メモリの機能を説明します。

構成

フラッシュ IP には、64 ビットのデータ・バスとそのデータに対応する ECC メタデータ用の 8 ビットが備わっています。メモリは複数のページとして構成されます。各ページのサイズは 2KB で、更に ECC 用に 256 バイトが確保されています。

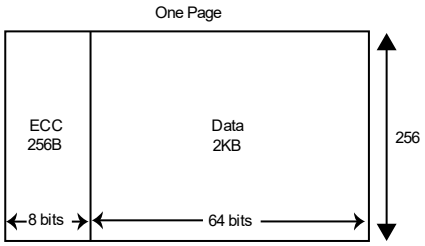


図 8-1：フラッシュ・メモリの 1 ページ

メモリのこれらのページは、情報スペース（Info Space）とユーザ・スペース（User Space）の 2 つの領域に分割されています。一般に、デバイスの合計ストレージ容量は、ユーザ・スペースのサイズとして記述されます。

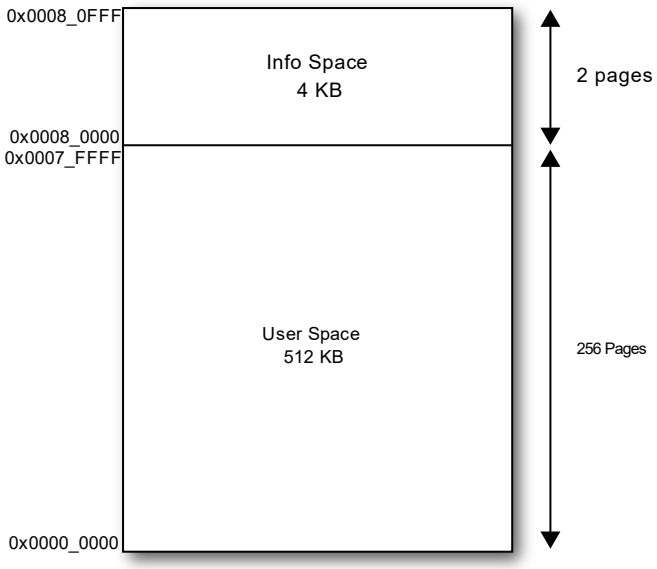


図 8-2：フラッシュの情報スペースとユーザ・スペース

情報スペースはアナログ・デバイセズが使用するために予約されており、通常はブートローダ（カーネル）、トリム値や校正値、その他デバイス固有のメタデータなどが保存されます。情報スペースのほとんどの部分はユーザ・コードによる読出しが可能ですが、消去や書込みはできません。

ユーザ・スペースは、ユーザ・データおよびプログラム・コード用に使うことを意図したフラッシュ・メモリ部分です。ユーザ・スペース内のアドレス範囲のごく一部はメタデータとしてフラッシュ・コントローラが使用し、保護と完全性に関する様々な機能を有効にします。

情報スペース

ユーザ・アクセス可能なデータ

情報スペースは一般にアナログ・デバイセズ専用として予約されていますが、0x0008_0EE0 から 0x0008_0EFF までのアドレス空間内の 32 バイトのデータは、ユーザ・コードによる読出しが可能です。この部分に保存されている情報の概要を下の表に示します。

表 8-1：デバイス情報スペース内のユーザ・アクセス可能パラメータのリスト

アドレス範囲	サイズ	説明
0x0008_0EF0 to 0x0008_0EFF	128 bits (16 bytes)	固有の ID
0x0008_0EE4 to 0x0008_0EEF	96 bits (12 bytes)	メーカーID
0x0008_0EE0 to 0x0008_0EE3	32 bits (4 bytes)	カーネルのレビジョン番号

アドレス空間 0x0008_0F00 から 0x0008_0FFF までの 256 バイトのデバイス情報スペースは保護されており、ユーザ・コードで読み出すことはできません（読み出そうとするとバス・エラーになります）。ユーザが情報スペースをプログラムしたり消去したりすることはできません（コマンドが拒否されます）。

ユーザ・コードが情報スペース内の保護範囲を読み出そうとすると、エラーになります。情報スペースに対する書込みと消去は拒否され、`FLCC_STAT` レジスタ内の該当ビットがセットされます。

ユーザ・コードは、このデバイスに対し、情報スペースの内容に影響を及ぼすことなく一括消去（Mass Erase）コマンドを実行することができます。これは、アナログ・デバイセズのセキュア・ブートローダに影響を与えることなく新しいユーザ・ファームウェアとデータをアップロードするメカニズムを提供します。

ユーザ・スペース

ユーザ・スペースは、ユーザ・データおよびプログラム・コード用に使うことを意図したフラッシュ・メモリ部分です。ユーザ・スペース内のアドレス範囲のごく一部はメタデータとしてフラッシュ・コントローラが使用し、保護と完全性に関する様々な機能を有効にします。

保護された鍵の保存領域

ユーザ・スペースの上位 2 ページは、保護された鍵（キー）の保存領域として使用するために予約されています。これら 2 つのページは物理的なユーザ・スペース内に存在していますが、ユーザが直接アクセスすることはできず、ユーザが使用できるどのフラッシュ・コントローラ機能（書込み保護など）でも制御できません。

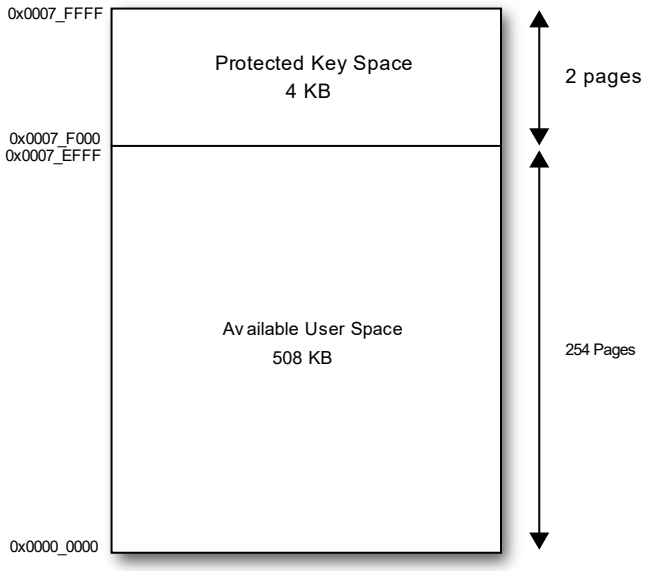


図 8-3：保護された鍵の保存領域として予約された専用ページ（512KB のユーザ・スペースを持つデバイス）

保護された鍵の保存領域に関してユーザが使用するすべてのアクセス機能と制御機能は、暗号化モジュールによって提供されます。

この領域にアクセスしようとすると、次に示すようにアクセスの内容に応じたエラーとなります。

- コマンド：この領域をターゲットとしてコマンドを入力すると、ステータス・レジスタにコマンド・エラー・コードがセットされ、フラッシュ・メモリは何も変化しません。
- ICode：命令コードとしてこのメモリ領域に分岐しようとすると、バス異常となります。あるいは、フラッシュ・コントローラが要求アドレスにアクセスしません。
- DCode：直接 AHB メモリ読出しとしてこの領域を読み出そうとすると、バス異常となります。あるいは、フラッシュ・コントローラが要求アドレスにアクセスしません。

メタデータ

アナログ・デバイセズのセキュア・ブートローダは、様々なセキュリティ機能を実行します。これらのセキュリティ機能は、ソフトウェアでユーザ・スペース内に定義された位置にメタデータを書き込むことを通じて、ユーザにより制御されます。

特に、フラッシュ・コントローラに影響するメタデータには、CRC チェックサムと書込み保護設定という 2 つのタイプがあります。

- 書込み保護：フラッシュ・アドレス 0x0000_019C

  アナログ・デバイセズのセキュア・ブートローダは単一のアドレスを定義し、そこから値を取り出して、ユーザ・コードの実行前にそれをフラッシュ・コントローラの FLCC_WRPROT レジスタに保存します。このフラッシュ位置には、FLCC_WRPROT に必要なデフォルト値がどのような値であっても、その値を書き込むことができます。

- CRC チェックサム

  シグネチャとも呼ばれる CRC チェックサムの保存には、各フラッシュ・ページの最上位ワードを使用できます。フラッシュ・コントローラは、検証対象となる任意の連続ページのシグネチャをソフトウェアが要求できるように、シグネチャ・チェック機能を実装しています。必要なシグネチャは、ユーザ・コードにより、シグネチャ計算領域の最上位ワードに保存する必要があります。

  アナログ・デバイセズのセキュア・ブートローダは、ユーザ・コードを実行する前に、0 ページからユーザが定義した最終ページまでシグネチャ・チェックを実行します。ブートローダは、ユーザ・スペースの 0 ページからメタデータを使って、シグネチャ検証に使用する最終ページを決定します。この値はユーザ・コード長（User Code Length）と呼ばれ、アドレス 0x0000_0194 に保存されます。

  0～255 の範囲にある値をユーザ・コード長アドレスに保存します。これは、シグネチャ生成に使用する最終ページを示します。また、表示された最終ページの最上位ワードに必要なシグネチャ値を書き込む必要もあります。

  シグネチャ・アドレスは次式で計算します。

  ```
  SIG_ADR = [ (PageID + 1) * (0x800) – 4]
  ```

  ここで、PageID はユーザ・コード長に保存された値です。

  例えば、PageID = 0 の場合、SIG_ADR は 0x7FC です。

  [SIG_ADR-4] にある隣接フラッシュ・ワードへの書込みを行わないようにして、シグネチャ値に対応する ECC バイトが破損しないようにしてください。

フラッシュ・アクセス

フラッシュ・メモリのうちの 2 ページは、保護された鍵の保存に使用する専用領域として予約されています。この領域へのすべてのアクセスは、暗号化ブロックを通じて間接的に行われます。フラッシュ・メモリの他のすべての領域へのアクセスは、3 つのインターフェース（ICode、DCode、APB）のいずれかを通じ、フラッシュ・コントローラを介して直接行われます。

これら 2 つのアクセス方法（直接および間接）が影響するのは APB インターフェースだけであり、ソフトウェアの整合性を維持し、保護された鍵の保存領域に必要なセキュリティ機能を提供するために、相互に排他的な形で実装されています。

表 8-2：間接アクセスと直接アクセス

Register	Direct Access	Indirect Access
Status Register	R/W	Read-only
Other Registers	R/W	No access

間接アクセス・モードでは、ユーザ・コードを使ってステータス・レジスタを読み出すことができますが、他のレジスタの書込みや読出しはできません。どのレジスタに書込みを試みても、影響はありません。ステータス・レジスタ以外のレジスタを読み出すと、フラッシュ・コントローラとは関係のない静的データが返されます。

ICode および DCode インターフェースは、アクティブなアクセス・モードの影響を受けることはありません。これらのインターフェースは論理的には直接アクセス・モードの一部ですが、通常は、常に直接アクセス読出しサービスを続けます。

IDLE 状態のときは、フラッシュ・コントローラは実質的に直接アクセス・モードにあります。間接アクセスは、ユーザ・コードにより、暗号化ブロック内のレジスタ書込みを通じて要求されます。フラッシュ・コントローラは、直接アクセス・モードと間接アクセス・モード間の切替えのみを行い、それ以外の場合は IDLE 状態となります（保留されているコマンドの完了時）。したがって、フラッシュ・コントローラが直接アクセス・コマンドを実行中でビジー状態の場合、間接アクセス要求は待機状態になります。

ユーザ・コードは、間接アクセスを、フラッシュ・コントローラが APB トラフィックの処理（フラッシュ・メモリの書込み、プログラム、または消去に使用するコマンド・レジスタを含むメモリ・マップ・レジスタの読出しまたは書込み）を行わないと予想される場合のみ実行される、重要ソフトウェア領域として扱う必要があります。ステータス・レジスタの最上位ビットは、現在のアクセス・モードを示します（0x0 = 直接）。

間接アクセス：保護された鍵の保存領域

ユーザ・スペースでは、保護された鍵の保存領域として使用するために 2 ページが予約されています。これらのページへのユーザ・アクセスは、暗号化モジュールを介した間接アクセスによってのみ行われます。この領域では、ユーザ・コードは以下のいずれかの操作を行うことができます。

- 書込み（WRITE）：鍵保存専用の領域に鍵を保存します。
- 破棄（DESTROY）：鍵保存専用の領域の鍵を 1 つ破棄します。
- 読出し（READ）：鍵保存専用の領域から鍵を 1 つ取り出します。
- 消去（ERASE）：この専用領域内のどちらかのページを消去します。

暗号化モジュールが上記の要求のいずれかを処理する一方で、フラッシュ・コントローラは ICode 要求と DCode 要求の処理を継続し、フラッシュ・メモリがビジー状態の場合は必要に応じて要求を保留するので、鍵保存領域とのやり取りを行いながら、ユーザ・コードをフラッシュ・メモリから透過的に実行することができます。

フラッシュ・コントローラがスリープ・モードにあるときに保護された鍵の保存領域に対して間接アクセスが行われた場合は、フラッシュ・コントローラが自動的にウェイクアップして要求を処理します。場合によっては、ユーザ・コードによってフラッシュ・コントローラをスリープ・モードに戻す必要があります。

保護された鍵保存領域への間接アクセスが行われている間、フラッシュ・コントローラは ABORT_EN レジスタの設定を無視します。間接フラッシュ・アクセスをユーザ・コード割込みによってアボートすることはできません。ユーザ・コードでは、セマフォまたは同等の機能を利用して間接フラッシュ・アクセスを重要コード・セグメントとして

扱い、この間は他のフラッシュ・アクセスが行われないようにすることを推奨します。間接アクセス時に ICode および DCode 直接アクセスを扱うこともできますが、性能は低下します。

フラッシュ・コントローラのステータス・レジスタは、間接アクセスを処理する際に自動的にクリアされます。

直接アクセス

フラッシュ・メモリは、ユーザ・コードによる読出し、書込み、消去が可能です。読出しアクセスは、専用のメモリとデータ・メモリ・バスを通じキャッシュ・コントローラ経由で行われます。書込みアクセスは、キーホール書込みを通じ、DMA ベース書込みのサポートを含めてメモリ・マップ・レジスタ経由で行われます。

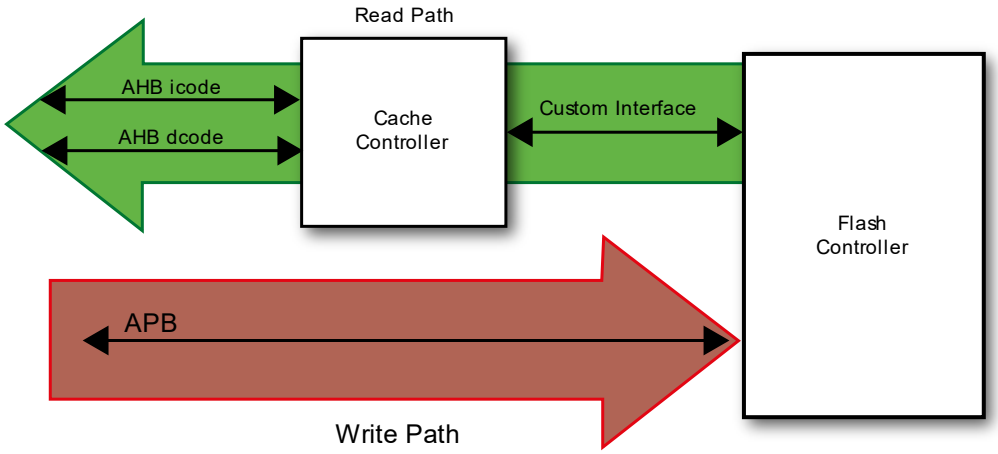


図 8-4：読出しおよび書込みデータ・パス

ユーザ・コードが、保護されたアドレスや範囲外のアドレスから読出しを行おうとした場合は、バス・エラーとなります。保護されたアドレスに対して書込みや消去を行おうとすると、それぞれに対応するフラグが `FLCC_STAT` レジスタにセットされます。書込みと消去のためのアドレス・セットアップは、自動的にフラッシュ・アドレスの範囲内に制限されます。

フラッシュの読出し

フラッシュ・コントローラには、ICode と DCode という 2 つの不揮発性ストレージ読出しインターフェースがあります。

ICode インターフェースと DCode インターフェースへのアクセスは、キャッシュ・コントローラ・モジュールを通じて AHB 経由で行われます。フラッシュ・コントローラには、ICode 用のプリフェッチ・バッファがあります。このプリフェッチ・バッファにより、ICode インターフェースと DCode インターフェースの両方を通じ、同じサイクルでデータを返すことができます。

フラッシュ・メモリは、自動初期化プロセス後にのみ読み出すことができます。フラッシュ・コントローラの初期化中に読出しを行おうとすると、実行が保留されます。フラッシュ・コントローラが別のコマンド（例えばフラッシュへの書込み）を実行していて既にビジー状態になっている場合に読出しを行おうとした場合も、その読出しがプリフェッチ・バッファによって実行される場合を除き、実行が保留されます。

フラッシュの消去

フラッシュ・コントローラは、ユーザ・スペースを消去する際にページレベルでの処理を可能にします（ERASEPAGE コマンド）。あるいは、ユーザ・コードによってユーザ・スペース全体を一度に消去することができます（MASSERASE コマンド）。これら 2 つのコマンドの実行時間は同じです。

書込み保護されたページを消去することはできません。この場合、コマンドが拒否されます。いずれかのページが書込み保護されている場合は、一括消去コマンドも拒否されます。ユーザ・スペースへの書込み保護を行い、一括消去を行うことを予定する場合（例えば、将来ファームウェアのアップグレードを行う場合）は、以上の点を考慮する必要があります。

フラッシュへの書込み

フラッシュ・メモリは、消去時には各ビットを 1 に設定し、データ書込み時にはそれらを 0 にクリアすることで動作します。一般的な書込みアクセスの際には、前もって消去を行う必要があります。

一般に、ユーザ・コンテンツの最初のアップロードは一括消去の実行直後に行われます。

フラッシュへの書込み後にその内容を変更するには、ユーザ・コードによって以下の操作を行う必要があります。

- ページ全体を SRAM へコピー
- 対象ページを消去
- メモリ内の内容を変更
- ページ内容を再びフラッシュへ書込み

ユーザ・スペースが保護されている場合はページ消去が拒否されます。詳細については、保護と完全性のセクションを参照してください。

一括消去またはブランク・チェックが正常に実行されると、すべてのユーザ・スペース保護は自動的にクリアされます（ブランク・チェックは、ユーザ・スペース全体が既に消去済みの場合のみ合格となります）。大容量消去またはブランク・チェックが正常に終了した場合、保護すべきユーザ・スペースの内容は存在しません。

キーホール書込み

キーホール書込みは、ユーザ・コードがターゲット・アドレスとデータ値を使ってメモリ・マップ・レジスタをプログラムし、その後バックグラウンドで書込み動作を行うようフラッシュ・コントローラにコマンドを送る、間接的な書込み動作です。フラッシュ・コントローラは、キーホール書込みによるフラッシュ・メモリへの書込みアクセスだけをサポートしています。書込みアクセスに関するこの制約により、フラッシュ・コントローラによる書込みを、対応する ECC バイトを使ったアトミックな DWORD（64 ビット）動作として正しく行うことができます。

1 つの DWORD（64 ビット）位置に複数の書込みを行う場合、書込みと書込みの間に影響を受けるページを消去しない限り、これら複数の書込みを行うことはできません。消去を行わないと、ECC エラーがレポートされます。

1 つのフラッシュ位置（64 ビット DWORD）に対して許容される書込みは、2 回の消去動作の間に 1 回だけです。任意の位置における 1 回の消去動作に対して複数回の書込みを行うと、ECC エラーとなります。

キーホール動作は、以下への書込みで構成されます。

- `FLCC_KH_ADDR.VALUE`：フラッシュのターゲット・アドレス（例えば 0x104）。フラッシュ・コントローラは、アドレスを DWORD 位置に合わせるために下位ビットを自動的にトリムします。
- `FLCC_KH_DATA0.VALUE`：書込みを行う 64 ビット DWORD のビット［31：0］（例えば 0x7654_3210）。
- `FLCC_KH_DATA1.VALUE`：書込みを行う 64 ビット DWORD のビット［63：32］（例えば 0xFEDC_BA98）。

- FLCC_CMD.VALUE：フラッシュに書込みコマンド（0x4）をアサートします。

書込みコマンドがアサートされると、フラッシュ・コントローラは、FLCC_KH_ADDR.VALUE で提供されるアドレスへの 64 ビットのデュアルワード書込みを開始します。

DMA アクセスが有効になっている（FLCC_UCFG.KHDMAEN がセットされている）場合は、ユーザ・コードによってこれらのキーホール・レジスタへ書込みを行うことはできません。DMA が有効になっているときにこれらのレジスタへ手動書込みを行うとフラッシュ書込みエラーとなり、DMA とフラッシュ・コントローラの同期が失われて、DMA コントローラが長時間（約 20～40µs）ハングすることがあります。

バースト書込み

フラッシュ・メモリの 2KB ページはそれぞれ 8 行で構成されており、1 行は 256 バイトで構成されています。フラッシュ IP のプログラミングに関する設計上の制約により、1 つの行内でバックツーバック書込みを行うことで、行の境界を超えて同様の書込みを行う場合よりも、迅速に書込みを完了させることができます。書込みを最適化するために、ユーザ・コードによるフラッシュ・メモリへの書込みは、最大 256 バイトのブロック単位で行う必要があります。

表 8-3：行ごとのフラッシュ・アドレス

0x7FFFC	0x7FFF8	...	0x7FF04	0x7FF00
...	...	...	...	...
0xFC	0xF8	...	0x4	0x0

この書込み性能の利点を生かすには、フラッシュ・コントローラが最初の書込みを完了する前に、2 回目（およびそれ以降）の 64 ビット書込みを要求する必要があります。最初の 64 ビット書込み動作の完了が近付くと（書込みサイクルの残りが約 20µs になると）、FLCC_STAT.WRALCOMP フラグがセットされます。これは管理可能な時間のウィンドウを提供し、ユーザ・コードはこのウィンドウの中で次の書込み要求をアサートすることができます。このフラグはポーリング可能です。もしくは、要求がアサートされた時点で割込みを生成することができます（オプション）。

以下のステータス・フラグを使用することもできます。

- FLCC_STAT.CMDBUSY：コントローラが FLCC_CMD レジスタからのコマンドを処理している間、ハイになります。
- FLCC_STAT.WRCLOSE：書込みコマンドの前半部分でハイになり、キーホール・レジスタをその後の書込みコマンド用にプログラムできる状態になるとクリアされます。
- FLCC_STAT.CMDCOMP：コマンド（書込みや消去など）が完了するとセットされるスティッキー・フラグです。1 を書き込むとクリアされます。
- FLCC_STAT.WRALCOMP：実行中の書込みコマンドの残り時間が約 20～40µs になるとセットされるスティッキー・フラグです。1 を書き込むとクリアされます。

1 つの行内でバースト書込みとなる可能性がある複数の書込みを行う手順を、以下に示します。

- コマンド・ビジー・フラグがクリアされるのを待ちます（前のコマンドが実行中でないことを確認する）。
- FLCC_KH_ADDR、FLCC_KH_DATA0、FLCC_KH_DATA1、および FLCC_CMD レジスタへのシーケンス書込みに対し、やはりフラッシュ書込みを行う可能性のある ISR による割込みが生成されないように、すべての割込みを無効にします。
- キーホール・アクセスを通じて最初の 64 ビット書込みを要求します。

- FLCC_CMD レジスタに書込みコマンドが書き込まれた後に、フラッシュ・コントローラが書込みプロセスを開始します。
- 書込みコマンドが受け入れられたかどうかをチェックします（FLCC_STAT レジスタを読み出し、何らかのエラー・フラグがセットされていないかどうか確認。コマンドがエラーとなった場合、書込みは実行されません）。
- FLCC_IEN.WRALCMPLT ビットをセットして（WrAIComp がアサートされたときの割込み生成を有効にして）、他の割込みを再び有効にします。
- ユーザ・プログラムを続行します。WrAIComp 割込みが割込みサービス・ルーチンへ送られ、適切なタイミングで次の書込みを要求します。
- バースト書込み ISR：WrAIComp によって割込みが生成されると呼び出される割込みサービス・ルーチン（ISR）。
- FLCC_KH_ADDR、FLCC_KH_DATA0、FLCC_KH_DATA1、および FLCC_CMD レジスタへのシーケンス書込みに対し、やはりフラッシュ書込みを行う可能性のある別の ISR による割込みが生成されないように、すべての割込みを無効にします。
- FLCC_STAT レジスタを読み出して複数のステータス・フラグの状態を確認し、同じ値を書き込むことによってフラグをクリアします。
  - FLCC_STAT.WRALCOMP：フラグがセットされていなければなりません。これは、バースト書込みの次の書込み実行までにまだ時間があることを示します。
  - FLCC_STAT.CMDBUSY：フラグがセットされてなければなりません。これは、現在の書込み動作がまだ実行中であることを示します。
  - FLCC_STAT.CMDCOMP：フラグがセットされていてはいけません。このビットは、コマンドが既に完了し、現在進行中のコマンドが前の書込みコマンドではないこと（別の機能が新しいコマンドを開始したこと）を示します。
  - FLCC_STAT.WRCLOSE：フラグがセットされていてはいけません。セットされている場合はキーホール・レジスタが閉じているので、書込みを行うことができません。FLCC_STAT.WRALCOMP がアサートされたときは、このフラグがクリアされていなければなりません。
- それ以上書込みを行う必要がない場合は、FLCC_STAT.CMDCOMP がセットされるのを待ち、それをクリアしてからサブルーチンを終了します。
- キーホール・アクセス（FLCC_KH_ADDR、FLCC_KH_DATA0、FLCC_KH_DATA1、および FLCC_CMD レジスタへの書込み）を通じて、次の 64 ビット・デュアルワード書込みを要求します。
- 書込みコマンドが受け入れられたかどうかをチェックします（FLCC_STAT レジスタを読み出して、何らかのエラー・フラグがセットされたかどうかを確認）。
- 割込みを再び有効にします。
- ISR を終了します。

フラッシュへの複数の書込みアクセスを伴うユーザ・コードは、可能であれば、フラッシュのスラッシングを防ぐために SRAM から実行する必要があります。次の命令のフェッチに ICode アクセスが必要なときに書込み操作を行うと、ICode 読出しが保留されて性能が低下します（これは、キャッシュ・コントローラにより部分的に緩和できます）。

バースト書込み時はフラッシュ・コントローラが書込み動作を重複させるので、書込み失敗のレポートには、重複するこれら 2 つの書込みのどちらかのステータスが反映されます。ユーザ・コードでは、バースト書込みの失敗を、

ユーザ・コードでは、バースト書込みの失敗を、重複している両方の書込み（つまり現在の書込み要求と 1 つ前の書込み要求）が失敗したと解釈する必要があります。

DMA 書込み

一般に、キーホール書込みでは、1 回のフラッシュ書込みアクセスにつき 4 つのメモリ・マップ・レジスタへ書込みを行う必要があります。オプションのアドレス自動インクリメント機能では、最初の書込みを除く一連の書込みすべてにおいて、1 回のフラッシュ書込みトランザクションに必要な APB トラフィックを 3 つのレジスタ書込み（FLCC_KH_DATA0、FLCC_KH_DATA1、および FLCC_CMD）に減らすことができます（最初の書込みでは開始アドレスのセットアップが必要）。DMA 書込みでは APB トランザクションの数が 2 つに減ります。FLCC_KH_DATA0 レジスタと FLCC_KH_DATA1 レジスタへの書込みをペアで行うごとにフラッシュへの書込みコマンドが 1 回実行され、アドレスが自動的に次の DWORD へインクリメントされます（自動インクリメントの値に関わらず、DMA 書込みは常にこのような形でアドレスのインクリメントを行います）。

DMA ベースの書込みを行うには、まずユーザ・コードによって DMA コントローラ（独立したペリフェラル・モジュール）を基本アクセス用に設定する必要があります（この DMA モードは、フラッシュ・コントローラを含むペリフェラルとの間のデータ転送をサポートしています）。DMA コントローラ・モジュールの設定完了後、このモジュールは、フラッシュ・コントローラによって DMA_REQ 信号がアサートされるまで IDLE 状態になります。

ユーザ・コードは、その都度 FLCC_KH_ADDR への書込みを行って、DMA 書込み用の初期ターゲット・アドレスをセットアップする必要があります。また、DMA コントローラは、すべての出力データを FLCC_KH_DATA1 へ書き込むように設定する必要があります。FLCC_KH_DATA1 への DMA 書込みが 2 回行われるごとにフラッシュ書込みコマンドが 1 回実行され、次の DMA_REQ が行われるまで、対応する遅延時間が取られます。

DMA コントローラからデータを要求するというフラッシュ・コントローラのプロセスを開始するには、フラッシュ・コントローラが DMA アクセス用に設定されている必要があります。DMA モードを有効にするには、ユーザ・コードにより FLCC_UCFG.KHDMAEN ビットをセットする必要があります。

DMA モードが有効になってフラッシュ・コントローラが IDLE 状態になると、フラッシュ・コントローラが DMA モジュールに DMA_REQ 信号を送ります。DMA_REQ は、バースト書込みをサポートするために適切なタイミングで送信されます（現在の書込み動作がほぼ完了すると、新しい要求が行われます）。

DMA 機能の詳細については、ダイレクト・メモリ・アクセス（DMA）を参照してください。

コマンド・アボート

フラッシュ・コマンドの中には、処理に長い時間がかかるものがあります。例えば、消去には 20～40ms かかります（FLCC_TIME_PARAM0 レジスタの内容により異なる）。このように時間を要するコマンドを処理するためにフラッシュ・コントローラがビジー状態にあるときに、低電圧アラームのようなクリティカル・イベント、つまり時間的に制約のあるイベントが発生することがあります。このようなシナリオでは、これらのクリティカル・イベントを適切に処理するために、実行中のフラッシュ・コマンドをアボート（中止）しなければならないことがあります。

コマンド・アボートは、時間的制約のあるクリティカルなイベントに対処するための最後の手段として、慎重に使用する必要があります。フラッシュ・コマンドをアボートすると現在行われているフラッシュ・アクセスを途中で終了させることになり、フラッシュ・データが破損したり、稀にフラッシュ・アレイ自体が損傷したりすることもあります。フラッシュ・コントローラは、次のフラッシュ・アクセスの前に内部の高電圧を安全に下げられるように適切なタイミングでコマンドをアボートすることによって、コマンドのアボートに起因するフラッシュ損傷やデータ破損の可能性を最小限に抑えようとします。このため、コマンドをアボートしようとしても、結果的に正常に完了することがあります。

実行中のフラッシュ・コマンドをアボートするためのメカニズムは 2 つあります。

- コマンド・レジスタ
  アボート・コマンドは、他のフラッシュ・コマンドと同じように、コマンド・レジスタへ書込みを行うことによって送出されます。アボート・コマンドが送出されると、実行中のすべてのコマンドが次の適切なタイミングで中止されます。
- システム割込みアボート
  フラッシュ・コントローラはシステム割込みバスの影響を受けます。ユーザ・コードで IRQ アボート・イネーブル・レジスタに書込みを行って、条件に一致する割込みが行われた場合は実行中のコマンドをフラッシュ・コントローラが自動的にアボートさせるようにプログラムすることができます。IRQ アボート・イネーブル・レジスタを使用すれば、0 から 63 までの任意のシステム IRQ を選択できます。

保護と完全性

情報スペースの完全性

情報スペースの内容をユーザ・コードで制御することはできません。パワーオン・リセット時にこの完全性チェックに失敗した場合は、予め設定された回数に達するか、チェックに合格するまで繰り返しチェックが行われます（これによって、デバイスのパワーアップ時に発生する電源故障に対し、ある程度の回復機能が得られます）。他のすべてのリセットでは（フラッシュ・コントローラを再初期化しないソフトウェア・リセットを除く）、完全性チェックは 1 回しか行われません。

情報スペースの完全性チェックに失敗した場合は、デバイスが故障していることが予想されます。この場合はそのデバイスを廃棄するか、故障解析のためアナログ・デバイセズに返送してください。情報スペースの完全性チェックに合格しなかった場合、フラッシュ・コントローラは特殊なデバッギング・モードになります。このモードではユーザ・スペース保護が自動的にアサートされ、シリアル・ワイヤ・デバッグ（SWD）インターフェースが ICode インターフェース、DCode インターフェース、および APB インターフェースと連携できるように、フラッシュ・コントローラのローカル JTAG 保護が設定されます。

この特別デバッギング・モードには以下のような制約があります。

- すべての ICode 読出しはバス・エラーとなります（完全性チェックに合格しない限り、コードがフラッシュ・メモリから実行されることはありません）。
- ユーザ・スペースに対するすべての DCode 読出しはバス・エラーとなります。
- 最上位の 256 バイトを除く情報スペースに対するすべての DCode 読出しは許可されます。最上位の 256 バイトに対する読出しはバス・エラーとなります。
- 書込みコマンドはすべて拒否されます（書込みを行おうとすると、`FLCC_STAT` レジスタの該当エラー・ビットがセットされます）。
- ユーザ・スペース保護は、セキュリティ要件を満たすことによってのみバイパスできます。

リセット後のシグネチャ・チェックのステータスは、`FLCC_STAT.SIGNERR` ビットから読み出されます。シグネチャ・チェック時の ECC のステータスは、`FLCC_STAT.ECCINFOSIGN` ビットに格納されます。SWD インターフェースがイネーブルされている場合、これらの値は通常の SWD 読出しを通じて読み出されます。

ユーザ・スペース保護

ユーザ・スペース保護は 2 階層で構成されます。

- アクセス保護：すべての読出し動作と書込み動作からユーザ・スペースを保護します。この保護メカニズムは手動でトリガできますが、通常は、システム異常の場合やサービス・ワイヤ・デバッグ・インターフェースがイネーブルされた場合に自動的にアサートされます。
- 書込み保護：ユーザ使用機能が有効になったユーザ・スペース・ページのブロックを、すべての書込みコマンドまたは消去コマンドから保護します。アナログ・デバイセズのセキュア・ブートローダによってランタイムで設定することができます（ブートローダがブート時に使用する必要値をフラッシュに保存）。

アクセス保護

アクセス保護は、第三者が JTAG やシリアル・ワイヤを通じてデータやプログラム・コードを不正に読み出したり変更したりするのを防ぐためのものです。アクセス保護はユーザ・スペース全体に適用されます。アクセス保護は、以下のどちらかのイベントで有効になります。

- シリアル・ワイヤ・デバッグが有効になった
- フラッシュの初期化（情報スペースのサイン・チェック）に失敗した

最初の 2 つの有効化メカニズムは自動機能で、アクセス保護を有効化／実行するために、これらのメカニズムのためにユーザ・コードで設定したり有効にしたりする必要はありません。

アクセス保護が有効になっている間、ユーザ・スペースからの読出しはすべてバス・エラーになり、書込みは拒否されます。また、消去の可否は FLCC_WRPROT.WORD ビットによって決定されます。

MASSERASE コマンドまたは BLANKCHECK コマンドを正常に実行できた場合は、アクセス保護をバイパスできます。FLCC_WRPROT（書込み保護）レジスタがそのリセット値から変更されている場合、MASSERASE コマンドを使用することはできません。BLANKCHECK コマンドは常に実行可能ですが、正常に終了できるのはユーザ・スペースがすべて消去された状態になっている場合に限られます。

書込み保護

ユーザ定義可能なユーザ・スペース領域は、フラッシュ・コントローラがその変更を拒否できるように設定可能です（これは WRITE コマンドと ERASE コマンドの両方に影響します）。書込み保護はランタイムに設定できます。または、デバイスのブート時にアナログ・デバイセズのセキュア・ブートローダによってロードされるユーザ・スペース・メタデータ内に保存できます。

注：FLCC_WRPROT レジスタ内のいずれかのビットがデフォルト値から変更されている場合、MASSERASE コマンドは使用できません。

ランタイム設定

書込み保護は、FLCC_WRPROT メモリ・マップ・レジスタを変更することによって設定されます。
FLCC_WRPROT.WORD は 32 ビット幅のビット・フィールドで、ユーザ・スペース・ページ内にある同様サイズのブロック 32 個に関する書込み保護の状態を表します。512KB のデバイスでは、フラッシュ・メモリは 256 ページのユーザ・スペース・ストレージに分割されます。これらは書込み保護のために、論理的にそれぞれ 8 ページ（16KB）のブロック 32 個に分割されます。書込み保護の制御は、これら 32 個のブロックそれぞれについて独立して行われます。FLCC_WRPROT.WORD の各ビットは、ユーザ・スペース内にある 8 ページのブロックそれぞれのために保護メカニズムを制御します。FLCC_WRPROT.WORD の最下位ビットは、ユーザ・スペースの最下位ページに対応しています。

FLCC_WRPROT レジスタ内のビットはアクティブ・ローです。0 は対応するページ・ブロックの書込み保護が有効であることを表し、1 は無効であることを表します。FLCC_WRPROT レジスタは 0 でスティッキーです。保護が有効になると、デバイスをリセットしない限り無効にすることはできません。

ユーザ・コードは、FLCC_WRPROT.WORD の該当ビットをクリアすることによって、いつでも任意のページのブロックに書込み保護をアサートすることができます。書込み保護は、ユーザ・コードのできるだけ早い段階でアサートする必要があります。また、アナログ・デバイセズのブートローダに依存して FLCC_WRPROT レジスタをセットすることなく書込み保護方式を完全に制御できるよう、ブロック 0（つまり、フラッシュ・ページ 0～3）を書込み保護して、ユーザ・ブートおよび完全性チェックのためのコードをこのブロックに置く必要があります。

メタデータ設定

ユーザ・スペースには、32 個の各論理ブロック（ユーザ・スペースのセクションを参照）に対応する 1 ビット書込み保護フラグのセットを表す 32 ビット・フィールドが 1 つ含まれています。これは、FLCC_WRPROT レジスタの機能と一致します。

これらの書込み保護ビットは、アナログ・デバイセズのセキュア・ブートローダによってフラッシュから読み出されて、リセット動作後に FLCC_WRPROT レジスタに保存されます。フラッシュ・メモリのデフォルト（消去された）状態はすべて 1 です。したがって、デフォルトの FLCC_WRPROT レジスタ値は、ユーザ・スペース内のすべてのページの保護を無効にします。FLCC_WRPROT.WORD の各ビットは、4 つのユーザ・スペース・ページからなるブロック 1 つの保護状態に対応しています。

ユーザ・フラッシュ・アドレス（アドレス 0x0000019C）にある書込み保護ワード（ランタイムには WrProt メタデータ・ワードとも呼ばれる）内のビットは、ユーザ・コードでクリアできます。あるいは、ユーザ・データとプログラム・コードの最初のアップロードにこのワードを含めることができます。ランタイムに WrProt メタデータ・ワードを書き込んでも、それが直ちに書込み保護状態に反映されることはありません。直ちに保護を有効にしたい場合は、ユーザ・コードによって FLCC_WRPROT レジスタに同じ値を書き込む必要があります。WrProt メタデータの書込み時には、最上位ページの書込み保護（これはページ消去やその他の変更からメタデータを保護します）を含めることを考慮する必要があります。

一度保護が有効になると（FLCC_WRPROT.WORD 内の該当ビットがクリアされると）、デバイスをリセットしない限り再び無効にすることはできません。このため、ユーザ・スペース・メタデータを通じてページ内のいずれかのブロックの書込み保護が一度有効になると、ユーザ・スペースの最上位ページ（関連するメタデータをホスティングしているページ）を消去せずに、あるいは、アナログ・デバイセズのセキュア・ブートローダのフローに影響せずに、再び無効にすることはできません。

以下のシーケンスは、書込み保護メタデータ・ワードをフラッシュにプログラムするプロセスの概要を示しています。

- 最後にメタデータが書き込まれた後で、ユーザ・スペースの FLCC_WRPROT アドレスが消去されていることを確認します。
- WrProt メタデータ・ワードに必要な値を直接フラッシュに書き込みます（対応するブロックの保護を有効にするには特定のビットに「0」を書き込みます）。
- ステータス・レジスタにポーリングを行うことによって、書込みが正常に終了したことを確認します。
- 影響するページに対して CRC シグネチャ・チェックを使用する場合は、ユーザ・コードでシグネチャをプログラムする必要があります。
- デバイスをリセットします。WrProt ビットは、アナログ・デバイセズのセキュア・ブートローダによって、ユーザ・スペースから FLCC_WRPROT レジスタへ自動的にアップロードされ、その保護方式を実行するために使われます。

シグネチャ

シグネチャは、フラッシュ・デバイスの内容の完全性をチェックするために使用します。シグネチャ計算には、フラッシュ・データ・バスの ECC 部分や、サインが行われるページのセットから読み出された最上位ワード（このワードはメタデータと見なされ、所定のページ・セットに対する予想シグネチャ値を保持することが目的です）は含まれません。

フラッシュ・コントローラには、シグネチャの生成と確認専用のスタンドアロン CRC エンジンが実装されています。ただし、この実装は CRC アクセラレータ・ペリフェラルと一致します（初期値は 0xFFFF_FFFF）。使用する CRC アルゴリズムの詳細については、巡回冗長検査（CRC）の章を参照してください。

シグネチャは、ユーザ・データとプログラム・コード（アップロード前に生成）の最初のアップロードに含めるか、ランタイムで生成してフラッシュに保存することができます。ランタイムで生成する場合は CRC アクセラレータを利用するか、フラッシュ・コントローラのシグネチャ生成ロジックを呼び出します。

ECC バイトはフラッシュ・メモリ内の 64 ビット DWORD に対応しているので、上位 64 ビット（32 ビット・シグネチャ・ワードを含む）はすべて同時に書き込む必要があります（さもないと、2 回目の書込みで ECC バイトが破損してしまいます）。フラッシュ・コントローラを使ってシグネチャ値を生成する場合は、シグネチャ・ワードとペアになった未使用の 32 ビットを消去状態（0xFFFF_FFFF）のままにします。さもないと（`FLCC_WRPROT` の設定によっては）デバイスのリセット後にスプリアス ECC エラーが発生し、デバイスを使用できなくなります。

サイン・コマンド（Sign）は、開始ページから終了ページまでのすべてのデータ（シグネチャ・メタデータ・ワードを除く）のシグネチャを生成します。ユーザ・コードでは、`FLCC_PAGE_ADDR0.VALUE` への書込みによって開始ページを、`FLCC_PAGE_ADDR1.VALUE` への書込みによって終了ページを定義する必要があります。

シグネチャの生成または検証を行う場合は、以下の手順に従ってください。

- `FLCC_PAGE_ADDR0.VALUE` に書込みを行います。これは一連のページの開始アドレスです（範囲を外れている場合、コマンドは拒否されます）。
- `FLCC_PAGE_ADDR1.VALUE` に書込みを行います。これは一連のページの終了アドレスです（範囲を外れている場合、コマンドは拒否されます）。
- `FLCC_KEY.VALUE` に書込みを行います。`FLCC_KEY` レジスタにユーザ・キーの値（0x676C7565）を書き込みます。
- `FLCC_CMD.VALUE` に書込みを行います。`FLCC_CMD` レジスタに SIGN コマンド（0x3）を書き込みます。
- 待機します。コマンドが完了すると、`FLCC_STAT.CMDCOMP` ビットがセットされます。
  - フラッシュ・メタデータへ書き込むシグネチャをフラッシュ・コントローラで生成する場合、シグネチャの値は `FLCC_SIGNATURE` レジスタから読み出します。
  - 生成されたシグネチャは、サインする領域の上位 32 ビットのワードに保存されたデータと自動的に比較されます。生成された結果が保存値と一致しない場合は、`FLCC_STAT.CMDFAIL` ビット・フィールド（0x2）に検証エラーをアサートすることによって、`FLCC_STAT` レジスタに不一致がレポートされます。

シグネチャの計算中は、フラッシュに対する他のすべてのアクセスが保留されます。512KB ブロック（全ユーザ・スペース）のシグネチャの生成／検証を行うと、保留時間は 64KB のフラッシュ読出し（1 回の読出し動作あたり 64 ビット）、または約 128k の HCLK 周期に相当する時間になります。

注：FLCC_PAGE_ADDR0.VALUE アドレスと FLCC_PAGE_ADDR1.VALUE アドレスはバイト・アドレスとして書き込むことができますが、フラッシュ・コントローラはこれをページ・アドレスとして使用します（下位の 10 アドレス・ビットは無視されます）。FLCC_PAGE_ADDR1.VALUE が FLCC_PAGE_ADDR0.VALUE より小さい場合、SIGN コマンドは拒否されます。シグネチャは、連続したアドレス範囲を対象として、常にページ単位で行われます。

キー・レジスタ

スプリアスを防ぎ、フラッシュ・アクセス破損のおそれをなくすために、いくつかのコマンドとレジスタはキー（鍵）で保護されています。ユーザ・キーはセキュリティ要素ではなく、秘密保持を目的としたものではありません。このキーは、ソフトウェア・バグによる悪影響からユーザを保護するためのものです（特にソフトウェア開発初期）。

ユーザ・キー

これは、フラッシュの一部の機能やアドレスに誤ってアクセスしてしまうのを防ぎます。キーの値は 0x676C_7565 です。保護されたユーザ・コマンド（ERASEPAGE, SIGN、MASSERASE、および ABORT）を実行したり、FLCC_UCFG レジスタ、または FLCC_TIME_PARAM0 および FLCC_TIME_PARAM1 レジスタへの書込みアクセスを有効にしたりするには、このキーを入力する必要があります。一度入力したキーは、キー・レジスタに誤った値が書き込まれるまで有効です。あるいは、何らかのコマンドが要求されて FLCC_CMD レジスタにコマンドが書き込まれると、このキーは自動的にクリアされます。FLCC_UCFG レジスタ、または FLCC_TIME_PARAM0 および FLCC_TIME_PARAM1 レジスタへの書込みアクセスを有効にするためにこのキーが入力された場合は、レジスタの更新後直ちにキーをクリアしてください。

ECC

フラッシュ・コントローラは、フラッシュ読出しについて ECC ベースのエラー検出とエラー訂正を行います。情報スペースの ECC はデフォルトで有効になるので、フラッシュの初期化機能を正しく機能させることができます（情報スペースのシグネチャ・チェックでは、無条件に ECC が考慮されます）。

フラッシュ・コントローラは、8 ビットの修正ハミング・コードを使用して、デュアルワード・フラッシュ・データ・アクセス（64 ビット）の 1 ビット・エラーの訂正や 2 ビット・エラーの検出を行います

シグネチャ・チェック中は ECC エンジンがアクティブになります（シグネチャのセクションを参照）。ユーザ・コードでユーザ・スペース全体のシグネチャ・チェックを要求して、更に FLCC_STAT.ECCERRCMD ビットをチェックし、シングル・ビットまたはデュアル・ビットのデータ破損がユーザ・スペース内に存在するかどうかを判定することができます。

デフォルトと設定

ECC エラーは、オプションで ICode または DCode 読出しのバス・エラーとしてレポートしたり、割込みを生成したりすることができます。1 ビット訂正や 2 ビット・エラー検出には、FLCC_IEN.ECC_ERROR ビットと FLCC_IEN.ECC_CORRECT ビットへの書込みによって、独立したエラー・レポート・オプションを使用することができます。

エラー処理

情報スペースのシグネチャ・チェック時の ECC エラーの影響については、シグネチャのセクションを参照してください。読出し動作時に ECC エンジンが 1 ビット・エラーを検出した場合、そのエラーは自動的に訂正されます（1 ビット・エラーは、ECC バイト自体の中か、ユーザによって読み出される 64 ビット・デュアルワード内で発生します）。2 ビット・エラーを検出した場合、ECC エンジンは検出イベントのレポートのみを行います（2 ビット・エラーは訂正されない）。

読出しがいつ行われたかに応じて（例えば ICode または DCode 読出し時、あるいはシグネチャ・チェックなどの組込みコマンドの一部として）、ステータス・レジスタ（FLCC_STAT.ECCERRCMD や FLCC_STAT.ECCRDERR など）に適切なフラグがセットされます。

FLCC_IEN.ECC_ERROR/FLCC_IEN.ECC_CORRECT ビットで割込み生成が有効になっている場合、割込みを発生させる ECC エラーのソース・アドレスは、割込みサービス・ルーチンによる読出し用に FLCC_ECC_ADDR レジスタ内に格納されます。

サイン・コマンド（Sign）実行時の ECC エラー

シグネチャ・チェック時に発生する ECC エラーは、チェック完了後にそのエラーに対応するステータス・レジスタ・フラグを生成しますが、FLCC_ECC_ADDR レジスタには書き込まれません。

同時エラー

DCODE と ICODE で同時に ECC エラーが発生した場合（例えば、ICODE のプリフェッチ・マッチと DCODE フラッシュ読出しから）、ECC エラー・ステータス情報の優先度付けは以下のように行われます。

- 優先度 1：2 ビット ECC エラーが 1 ビット ECC エラー／訂正よりも優先されます。

  例えば、ICODE 読出しの 1 ビット ECC エラー／訂正と同じサイクルで DCODE 読出しの 2 ビット ECC エラーが検出された場合は、DCODE の ECC エラー・ステータスだけが更新されます。

- 優先度 2：DCODE よりも ICODE が優先されます。

  例えば、同じサイクルの ICODE 読出しと DCODE 読出しで 2 ビット・エラーが検出された場合は、ICODE の ECC エラー・ステータスだけが更新されます。

消去位置の読出し

消去後のフラッシュ・メモリの保持値は、すべての 64 ビット DWORD に付加された ECC バイトを含め、すべて 1 になります。64 個の 1 の正しい ECC メタデータは 0xFF ではないので、フラッシュ・メモリはその消去された状態で、データといくつかのビット・エラーを表す ECC メタデータを保持します。

このため、消去位置のフラッシュ読出しは自動的に ECC エンジンをバイパスします。ユーザ・コードが、64 ビット DWORD と ECC バイトのすべてが 1 になっている位置を読み出すと、ECC エラーなしでデータが返されます。

クロックとタイミング

フラッシュ・コントローラは、26MHz 未満のコア・クロック周波数と 13MHz の基準クロック周波数のすべてのフラッシュ動作に安全なタイミング・パラメータを提供するように、予め設定されています。

コア・クロック周波数が、データシートの HFOSC に指定された制限範囲内にない場合は、ユーザ・コードにより相応のタイミング・パラメータ調整を行う必要があります。

コア・クロック周波数が 26MHz より速い場合（例えば 52MHz）にフラッシュ・アクセスを行うには、まずユーザ・コードによって FLCC_TIME_PARAM1 レジスタへ書込みを行い、WAITSTATES の値をデフォルト（0x0）から 0x1 以上に増やす必要があります。この調整は、フラッシュ・メモリの読出しパスに 1 つ（または複数）の待機状態を挿入

します。読出しパスに少なくとも 1 つの待機状態を追加しないと、26MHz を超えるコア・クロック周波数での読出し中に、フラッシュ・メモリが未定義の動作をする可能性があります。

フラッシュの動作モード

ADuCM4050 MCU が使用するフラッシュ・メモリは、以下の消費電力最適化機能をサポートしています。

スリープ・モード

ユーザ・コードを使い、SLEEP コマンド（0x2）を FLCC_CMD レジスタへ書き込むことによって、フラッシュ IP を低消費電力スリープ・モードにすることができます。フラッシュ・コントローラは、SLEEP コマンド入力後の最初のフラッシュ・アクセス時に、自動的にフラッシュ IP のスリープ状態を解除します。ユーザ・コードは、FLCC_STAT.SLEEPING ビットを読み出すことによって、フラッシュをスリープ状態からウェイクアップすることができます。

注：キャッシュ・コントローラ、DMA 読出し、その他のペリフェラル、あるいはユーザ・コードはいつでもフラッシュ・メモリを読み出すことができ、どの動作もフラッシュのウェイクアップ・イベントをトリガします。ユーザ・コードは、ときどき FLCC_STAT.SLEEPING ビットをポーリングして、フラッシュ IP がスリープ状態にあるかどうか確認できるようにすることを推奨します。

フラッシュ・コントローラは、フラッシュ IP がスリープ・モードにある間は新しいコマンド（WRITE や ERASE など）を受け入れません。このモードでサポートされているコマンドは IDLE と ABORT だけです。スリープ・モードに入ると、DMA 書込み要求は自動的に保留されます。

一般に、システム割込みに基づくアボート（FLCC_ABORT_EN_LO レジスタと FLCC_ABORT_EN_HI レジスタを通じた設定による）は、システム割込みが有効になったときに実行中のフラッシュ・コマンドを中止するために使われます。しかし、このような割込みは、スリープ・モードのフラッシュをウェイクアップしません。フラッシュにアクセスせずにシステム割込みを処理できる場合、フラッシュはスリープ・モードのままになります。割込みを処理するためにフラッシュへアクセスする必要がある場合は、フラッシュ・アクセス自体がフラッシュ IP をウェイクアップします。

スリープ状態からウェイクアップして読出しまたはコマンドを実行するまでには、約 5µs の遅延があります（これはフラッシュ IP の要求事項です）。ユーザ・コードは、IDLE コマンドを実行することによって、フラッシュ IP を早めにウェイクアップすることができます。これは、コントローラに他の影響を与えることなく、フラッシュ IP をウェイクアップします。

整合性を維持するために、ABORT コマンドを使用してフラッシュ IP をウェイクアップすることもできます。ABORT コマンドによるウェイクアップと IDLE コマンドによるウェイクアップの違いは、ステータス・レジスタ値とユーザ・コードの予想値との一致をチェックするために、ステータス・レジスタが FLCC_STAT.CMDFAIL ビット（0x3）をレポートするという点だけです。

パワーダウン・モード

ADuCM4050 MCU は、デバイスが休止状態になると自動的にフラッシュ IP をパワーダウンします。この機能をサポートするために、フラッシュ・コントローラはパワー・マネージメント・ユニットと連携して、実行中のフラッシュ・アクセスが完了するまで休止状態になるのを遅らせます。休止状態に入る前のフラッシュ・ステータス・レジスタの読出しと評価は、ユーザ・コードによって行います（休止時にはステータス・レジスタの内容は維持されません）。アボート・コマンドを使用して実行中のフラッシュ・コマンドを直ちに中止することができますが、最終的にフラッシュ・アレイを損傷させてしまうことがないよう、慎重を期する必要があります。

クロック・ゲート

フラッシュ・コントローラには一連のクロック・ゲートが挿入されており、モジュールの未使用コンポーネントへのクロック供給を自動的に停止します。ユーザによる設定や制御は不要です。フラッシュ・コントローラの未使用部分は、必要に応じてクロック供給が自動的に停止されます（例えば、スリープ・モードの間はフラッシュ・コントローラの大部分は節電のために停止されます）。

フラッシュ割込みと例外

フラッシュ・コントローラは、複数イベントに対して選択的に割込みを生成することができます。割込みを生成するイベントと、各イベントの割込み生成を制御するために使用する FLCC_IEN のビット・フィールドの概要を下の表に示します。

表 8-4：割込みとビット・フィールド

ビット名（ビット・フィールド）	説明
CMDFAIL	コマンドまたは書込み動作がエラー・ステータスと共に完了したときに IRQ が生成。
WRALCOMP	アクティブなフラッシュ書込みがほぼ完了して、キーホール・レジスタが別の書込みのために開かれたときに IRQ が生成（同時に完了した場合はバースト書込みとなります）。
CMDCMPLT	コマンドまたはフラッシュ書込み動作が完了したときに IRQ が生成。
ECC_ERROR	フィールドが 2 にセットされているとき、2 ビット ECC エラーの検出時に IRQ が生成。
ECC_CORRECT	フィールドが 2 にセットされているとき、1 ビット ECC エラーの検出時に IRQ が生成。

フラッシュ・プログラミング・モデル

ここでは、フラッシュ・コントローラを使用してページ消去コマンドを実行するシーケンスの例を示します。多少の修正を加えれば、他のコマンドにも同じシーケンスを使用できます。

プログラミングのガイドライン

プログラミングのガイドラインを以下に示します。

1. 消去が必要なページのアドレスを使って、FLCC_PAGE_ADDR0 レジスタまたは FLCC_PAGE_ADDR1 レジスタをプログラムします。
2. FLCC_KEY.VALUE ビットにフラッシュ・ユーザ・キー（0x676C7565）を書き込みます。
3. 実行するコマンドを FLCC_CMD レジスタに書き込みます。
4. FLCC_STAT.CMDCOMP ビットのセットをポーリングします。

ADuCM4050 FLCC レジスタの説明

フラッシュ・コントローラ（FLCC）には以下のレジスタが含まれています。

表 8-5：ADuCM4050 FLCC レジスター一覧

レジスタ名	説明
FLCC_ABORT_EN_HI	IRQ アボート・イネーブル（上位ビット）
FLCC_ABORT_EN_LO	IRQ アボート・イネーブル（下位ビット）
FLCC_POR_SEC	フラッシュ・セキュリティ
FLCC_VOL_CFG	揮発性フラッシュの設定
FLCC_CMD	コマンド
FLCC_ECC_ADDR	ECC ステータス（アドレス）
FLCC_IEN	割込み許可
FLCC_KEY	キー
FLCC_KH_ADDR	書込みアドレス
FLCC_KH_DATA0	下位データの書込み
FLCC_KH_DATA1	上位データの書込み
FLCC_PAGE_ADDR0	下位ページ・アドレス
FLCC_PAGE_ADDR1	上位ページ・アドレス
FLCC_SIGNATURE	シグネチャ
FLCC_STAT	ステータス
FLCC_TIME_PARAM0	時間パラメータ 0
FLCC_TIME_PARAM1	時間パラメータ 1
FLCC_UCFG	ユーザ設定
FLCC_WRPROT	書込み保護
FLCC_WR_ABORT_ADDR	書込みアボート・アドレス

IRQ アボート・イネーブル（上位ビット）

IRQ アボート・イネーブルの上位ビット

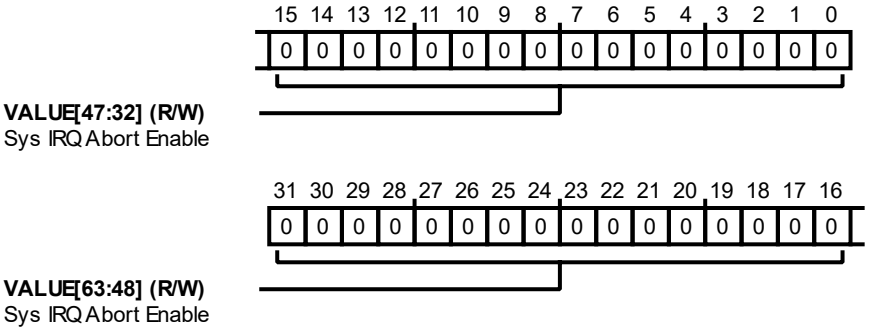


図 8-5：FLCC_ABORT_EN_HI レジスタ図

表 8-6：FLCC_ABORT_EN_HI レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/W)	VALUE	システム IRQ アボート・イネーブル。 実行中のフラッシュ・コマンド（例えば消去、書込み、サイン）が、必要システム IRQ 番号に対応するこのレジスタ内のビットへ 1 を書き込むのを中止するために、システム割込みを許可します。この機能がサポートしているのは、最初の 64 個のシステム IRQ だけです。

IRQ アボート・イネーブル（下位ビット）

IRQ アボート・イネーブルの下位ビット

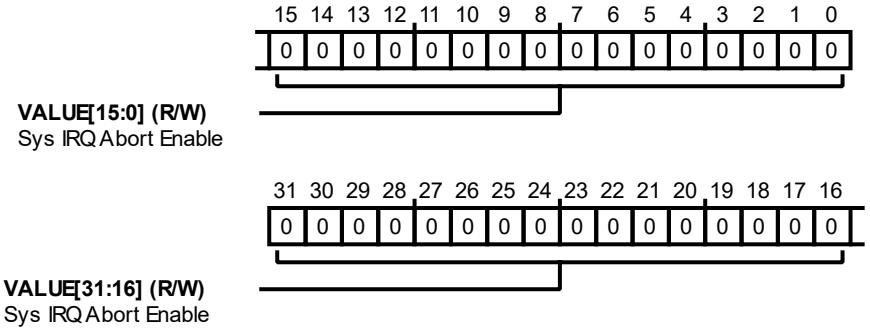


図 8-6：FLCC_ABORT_EN_LO レジスタ図

表 8-7：FLCC_ABORT_EN_LO レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/W)	VALUE	システム IRQ アボート・イネーブル。 実行中のフラッシュ・コマンド（例えば消去、書込み、サイン）が、必要システム IRQ 番号に対応するこのレジスタ内のビットへ 1 を書き込むのを中止するために、システム割込みを許可します。この機能がサポートしているのは、最初の 64 個のシステム IRQ だけです。

フラッシュ・セキュリティ

フラッシュのセキュリティ機能を制御します。

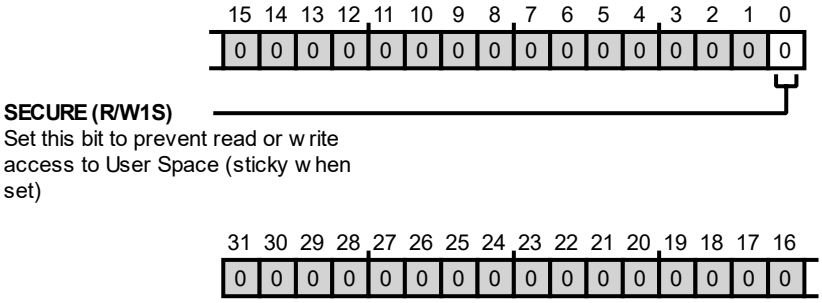


図 8-7：FLCC_POR_SEC レジスタ図

表 8-8：FLCC_POR_SEC レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
0 (R/W1S)	SECURE	ユーザ・スペースへの読出しまたは書込みアクセスを防止するには、このビットをセットします（セット後はスティッキー）。 ユーザ・キーが必要です。 一度セットすると、デバイスをリセットするか（POR または PIN リセットのみ）、ユーザ・スペースの消去／消去状態確認（ブランク・チェックまたは一括消去実行の正常終了）をしない限りクリアすることはできません。 これは、ユーザ・スペースのセキュリティ実行において直接的な役割を果たします。セット後、このビットはユーザ・スペースへのアクセスを防止します。 DCode 読出し：データ・バス == 0 でバス異常を返します。 ICode 読出し：データ・バス == 0 でバス異常を返します。 APB 書込み：コマンドが拒否されます。フラッシュの内容は変わりません。 セットすると、ユーザ・コードは引き続き一括消去動作とページ消去動作を実行します。 注：FLCC_WRPROT レジスタが引き続き適用されます。保護されていないページだけ消去できます（保護されているページが存在する場合、一括消去は使用できません）。

揮発性フラッシュの設定

このレジスタは、（POR／PIN／SW／WDT の）いずれかのリセットまたはすべてのリセットによってリセットされます。このレジスタへの書込み時は、前もって KEY レジスタへユーザ・キーを書き込んでおく必要があります。

図 8-8：FLCC_VOL_CFG レジスタ図

表 8-9：FLCC_VOL_CFG レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
0 (R/W1C)	INFO_REMAP	ユーザ・スペースのベース・アドレスに情報スペースをエイリアスします。 このビットがセットされると、ブートローダがアドレス 0 にミラーされます。このビットは、ブートローダにより自動的にクリアされます。

コマンド

指定されたコマンドを実行するには、このレジスタに書込みを行います。ほとんどのコマンド要求受け入れには、まず FLCC_KEY レジスタへユーザ・キーを書き込む必要があります。

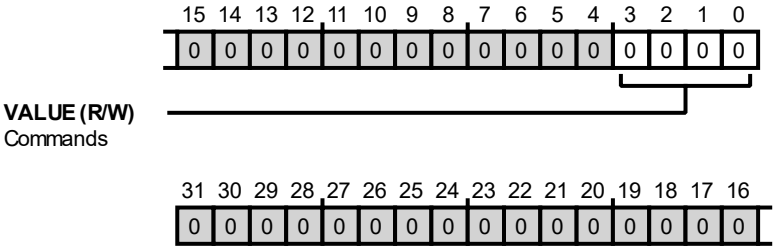


図 8-9：FLCC_CMD レジスタ図

表 8-10：FLCC_CMD レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
3:0 (R/W)	VALUE	コマンド。 特定の動作を開始するには、このレジスタへコマンド値を書き込みます。書込みとアイドルを除くすべてのコマンドでは、まず FLCC_KEY レジスタへユーザ・キーを書き込む必要があります。	
		0	IDLE キーは必要ありません。 どのコマンドも実行されません。IP がスリープ状態にある場合、このコマンドはフラッシュをウェイクアップします（コマンド完了を返し、エラー・コードは返しません）。

表 8-10：FLCC_CMD レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
		1	ABORT。 ［ユーザ・キー］が必要です。 ABORT コマンドは、時間的制約のあるイベントの際にフラッシュ IP にアクセスするための最後の手段として、慎重に使用する必要があります。例えば、低電圧アラームは、実行中のフラッシュ書込みコマンドまたは消去コマンドをアボートして、ユーザ・コードによりデバイスをスムーズにシャットダウンする理由となり得ます。ABORT コマンドを多用すると、フラッシュ・アレイを損傷させるおそれがあります。 このコマンドが送出されると、現在実行中のコマンドが突然停止します（停止可能な場合）。ステータスは、ABORT エラーによってコマンドが完了したことを示します。 ABORT は、別のコマンドを実行中に送出できる唯一のコマンドです（例外の 1 つがユーザ・コードで、ユーザ・コードでは実行中の書込みコマンドの上に別の書込みコマンドを重ねることができます。新しいコマンドが ABORT である場合を除き、他のすべての重複コマンドの組み合わせは無効です）。 書込みまたは消去がアボートされた場合はいくつかのフラッシュ IP タイミング条件に違反することがありますが、その書込みまたは消去が正常に完了したかどうかを知ることはできません。アボートされたコマンドの結果を判定するには、影響を受けた場所をユーザ・コードで読み出す必要があります。コマンドをアボートすると、フラッシュが正しく設定されないことがあります。アボートした場合は影響する領域を消去して、プログラムし直すことを常に推奨します。 フラッシュ・コントローラがコマンドの実行処理をどこまで進めたかによって、アボートできない場合もあります（違反できないフラッシュ IP タイミング・パラメータがいくつかあります）。これをソフトウェアで予測することは（不可能ではないにしても）困難なので、アボート時には実際のコマンド実行時間に影響しないような要求を考える必要があります。

表 8-10：FLCC_CMD レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
		2	スリープ・モードに入るようフラッシュに要求します。 ［ユーザ・キー］が必要です。 フラッシュをスリープ状態（低消費電力モード）にするようフラッシュ・コントローラに要求します。 スリープ状態にあるフラッシュは、ICode、DCode、または DMA トランザクションによって自動的にウェイクアップします。 ウェイクアップ・プロセスには約 5µs かかります（FLCC_TIME_PARAM1 レジスタで設定可能）。ユーザ・コードがフラッシュの必要性を約 5µs 前に予測できる場合は、FLCC_CMD レジスタに IDLE コマンドを書き込んで、フラッシュを手動でウェイクアップすることができます。ABORT コマンドは、デバイスのウェイクアップにも使われます（この場合、スリープ・コマンドがアボートされたことを示す適切なステータスを返します）。 ウェイクアップしたデバイスは、ユーザ・コードが再度 SLEEP コマンドをアサートするまでその状態を維持します。
		3	SIGN。 ［ユーザ・キー］が必要です。 データ・ブロック用のシグネチャを作成するには、このコマンドを使用します。シグネチャを作成できるのは、ブロックが欠落部分のない完全なページだけで構成されている場合に限られます。シグネチャの作成を開始するには、開始ページのアドレスを FLCC_PAGE_ADDR0 レジスタに書き込み、最終ページのアドレスを FLCC_PAGE_ADDR1 レジスタに書き込んでから、このコードを FLCC_CMD レジスタに書き込む必要があります。コマンドの実行が完了すると、FLCC_SIGNATURE レジスタからシグネチャを読み出せるようになります。
		4	WRITE。 キーは必要ありません。 このコマンドは、FLCC_KH_ADDR、FLCC_KH_DATA0、および FLCC_KH_DATA1 レジスタからアドレスとデータを取得して、指定されたアドレスに対して 64 ビットの書込み動作を 1 回実行します。

表 8-10：FLCC_CMD レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
		5	すべてのユーザ・スペースをチェックします。ユーザ・スペース内のいずれかのビットがクリアされている場合はチェック不合格となります。 ［ユーザ・キー］が必要です。 すべてのユーザ・スペースに対してブランク・チェックを実行します。ユーザ・スペース内のいずれかのビットがクリアされている場合は、READ VERIFY（読出し確認）ステータスとなってコマンドを実行できません。すべてのユーザ・スペースが FF の場合、コマンドは正常に実行されます。 このコマンドは、ユーザの迅速なソフトウェア開発の支援を意図したものです。ユーザ・スペースの読出しと書込みを防ぐセキュリティ機能と共に未プログラム部分を起動するときは、このコマンドを使用して、ユーザ・スペースに専有情報が含まれていないことを確認できます。このコマンドが正常に実行されると、ユーザ・スペースの読出しおよび書込み保護がクリアされます。
		6	ERASEPAGE。 ［ユーザ・キー］が必要です。 消去するページのアドレスを FLCC_PAGE_ADDR0 レジスタに書き込み、更にこのコードを FLCC_CMD レジスタに書き込みます。消去が完了すると、すべての内容が消去されたことを確認するために、自動的にページ全体の検証（読出し）が行われます。読出し検証エラーがある場合は FLCC_STAT レジスタで示されます。複数ページを消去するには、前のページの消去が完了するまで待ってからステータスをチェックし、その後に次のページの消去を開始するコマンドを送出します。
		7	MASSERASE。 ［ユーザ・キー］が必要です。 すべてのフラッシュ・スペースを消去します。消去が完了すると、すべての内容が消去されたことを確認するために、自動的にスペース全体の検証（読出し）が行われます。読出し検証エラーがある場合は、ステータス・レジスタで示されます。
		8	予備
		9	予備

ECC ステータス（アドレス）

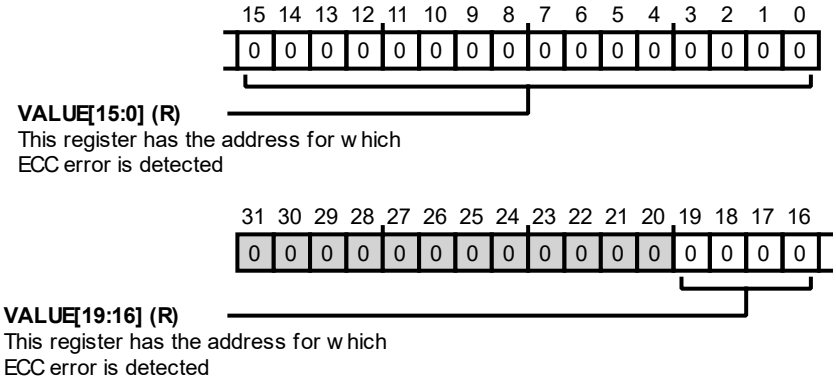


図 8-10：FLCC_ECC_ADDR レジスタ図

表 8-11：FLCC_ECC_ADDR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
19:0 (R/NW)	VALUE	このレジスタには、ECC エラーが検出されたアドレスが格納されます。 このレジスタは、選択に従って ECC エラー時または訂正時に更新され、FLCC_IEN レジスタ内に割込み（IRQ）を生成します。このレジスタは、代わりにバス異常を生成する ECC エラーまたは訂正の場合は更新されません。このレジスタは、最後の ECC ステータス・ビットのクリア後（あるいはリセット後）初めての ECC エラーまたは訂正イベントのアドレスを記録して、割込みを生成します。新しい ECC イベント（IRQ を生成するために選択されたもの）と同じサイクル内でステータス・ビットがクリアされた場合は新しいアドレスが記録され、ステータス・ビットはセットされたままになります。エラーは訂正より優先されます（2 ビット以上の破損 = ERROR で、訂正の場合は 1 ビット訂正後に正しいデータが返されます）。同じサイクル内でエラーと訂正が発生した場合、このレジスタはエラー・アドレスをレポートします。優先度が同じ ECC イベントが 2 つ発生したときは（両方ともエラーまたは両方とも訂正）、DCODE バスより ICODE バスの方が優先されます。したがって、ICODE バスと DCODE バスの両方で同じタイプの ECC イベントが生成された場合、このレジスタには ICODE アドレスが保存されます。レジスタは、リセットしない限りクリアできません。このレジスタには、常に最後にレポートされた ECC 訂正またはエラーのアドレスが格納されます。

割込み許可

いつ割込みを生成するかを指定するために使われます。

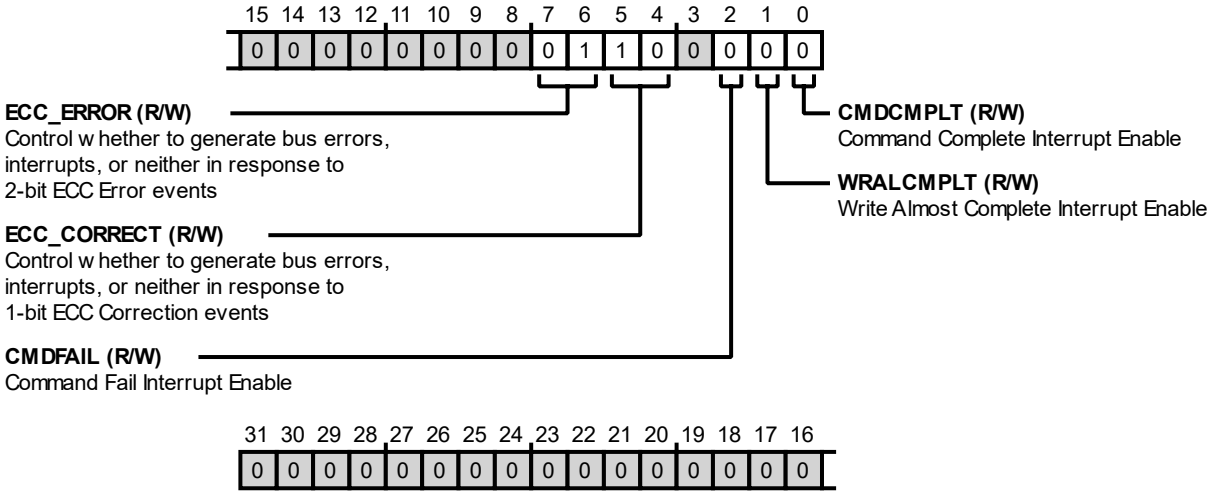


図 8-11：FLCC_IEN レジスタ図

表 8-12：FLCC_IEN レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
7:6 (R/W)	ECC_ERROR	2 ビット ECC エラー・イベントに対してバス・エラーを生成するか、割込みを生成するか、あるいはどちらも生成しないかを制御します。	
		0	ECC イベントに対して応答を生成しません。
		1	ECC イベントに対してバス・エラーを生成します。
		2	ECC イベントに対して IRQ を生成します。
5:4 (R/W)	ECC_CORRECT	1 ビット ECC 訂正イベントに対してバス・エラーを生成するか、割込みを生成するか、あるいはどちらも生成しないかを制御します。 注：ハードウェア・リセット値は 2 ですが、カーネルはこれらのフィールドに値 0 を書き込みます。	
		0	ECC イベントに対して応答を生成しません。
		1	ECC イベントに対してバス・エラーを生成します。
		2	ECC イベントに対して IRQ を生成します。
2 (R/W)	CMDFAIL	コマンド失敗割込みを許可。 このビットをセットすると、コマンドまたはフラッシュ書込みがエラー・ステータスと共に終了したときに割込みが生成されます。	
1 (R/W)	WRALCMPLT	書込みがほぼ終了した時点での割込みを許可。	

表 8-12：FLCC_IEN レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
0 (R/W)	CMDCMPLT	コマンド完了割込みを許可。 このビットをセットすると、コマンドまたはフラッシュ書込みの完了時に割込みが生成されます。

キー

保護された機能へアクセスするためにユーザ・コードでキー（鍵）を書き込む必要がある場合は、このレジスタにキー値を書き込む必要があります。

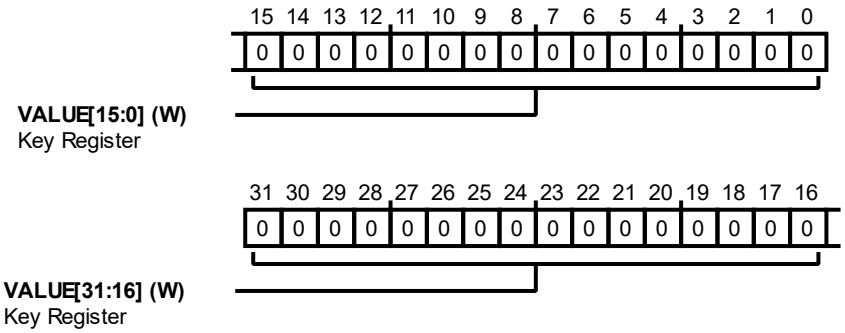


図 8-12：FLCC_KEY レジスタ図

表 8-13：FLCC_KEY レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
31:0 (RX/W)	VALUE	キー・レジスタ。 このレジスタに適切なキー値を書き込むことによって、保護された機能のロックを解除します。	
		1735161189	USERKEY。特定のレジスタを編集できるようにしたり、特定のコマンドを実行できるようにしたりするには、User Key（ユーザ・キー）を書き込みます。 このキーは、フラッシュの内容を誤って変更してしまうのを防ぐためのサニティ・チェックとして使われます。これはセキュリティ・コンポーネントではなく、秘密情報とすることを意図したものではありません。

書込みアドレス

WRITE コマンドのターゲットとなる 64 ビット・デュアルワード・フラッシュ位置にあるバイトのバイト・アドレスを書き込みます。すべての書込みは、フラッシュ・アレイ内の 64 ビット・デュアルワード要素をターゲットとします。フラッシュ・メモリの有効範囲を超えるアドレスを書き込むと、エイリアシングを防ぐためにアドレスが飽和します。ユーザ・コードでは、有効なフラッシュ・アドレス位置をターゲットとするように注意する必要があります。

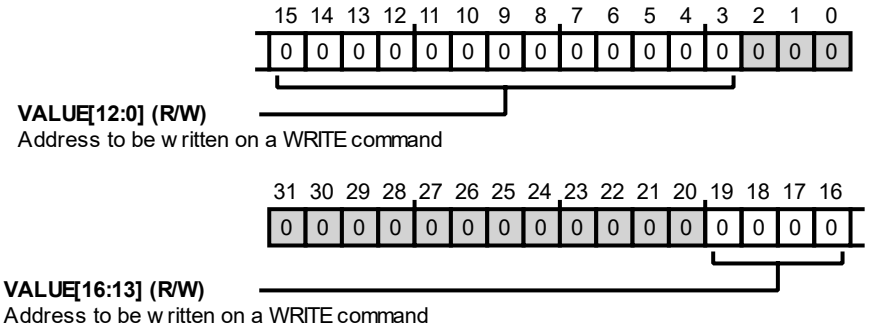


図 8-13：FLCC_KH_ADDR レジスタ図

表 8-14：FLCC_KH_ADDR レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
19:3 (R/W)	VALUE	WRITE コマンドで書き込まれるアドレス。

下位データの書込み

フラッシュに書き込む 64 ビット・デュアルワード・データの下位側半分。

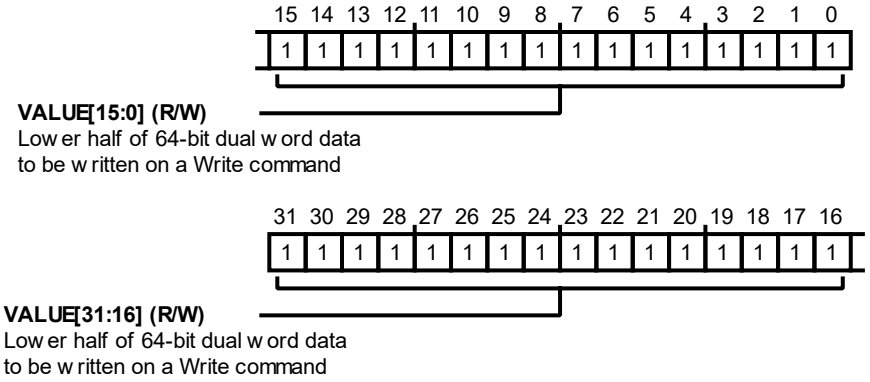


図 8-14：FLCC_KH_DATA0 レジスタ図

表 8-15：FLCC_KH_DATA0 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (R/W)	VALUE	書込みコマンドで書き込む 64 ビット・デュアルワード・データの下位側半分。

上位データの書込み

フラッシュに書き込む 64 ビット・デュアルワード・データの上位側半分。

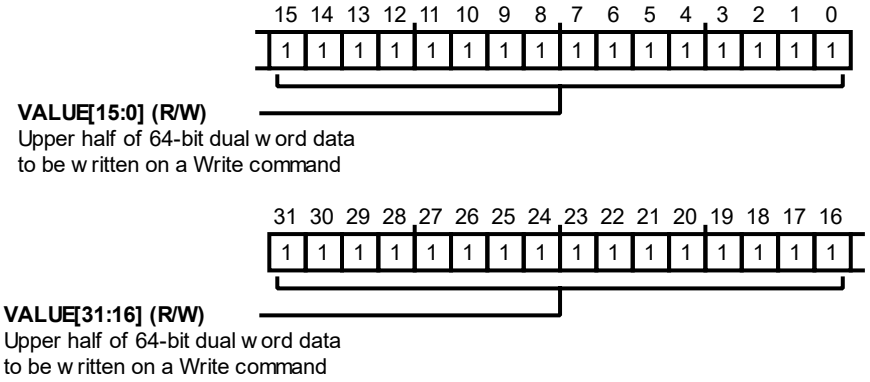


図 8-15：FLCC_KH_DATA1 レジスタ図

表 8-16：FLCC_KH_DATA1 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/W)	VALUE	書込みコマンドで書き込む 64 ビット・デュアルワード・データの上位側半分。 DMA が有効になっている場合、このレジスタは FIFO として動作します。このレジスタへ書込みを行うと、古いデータが 64 ビット・データの下位 32 ビット（FLCC_KH_DATA0）にプッシュされます。このレジスタへの書込みを（DMA モードで）2 回行うと、FIFO がフルになりフラッシュ書込みコマンドが自動的に実行されます。

下位ページ・アドレス

バイトが存在するページを選択するには、そのバイトのアドレスをこのレジスタへ書き込みます。選択されたページは、ERASEPAGE コマンド（どのページを消去するかを選択）または SIGN コマンド（シグネチャを計算するブロックの開始ページを選択）に使用できます。FLCC_PAGE_ADDR0 と FLCC_PAGE_ADDR1 の両方を使用するコマンドの場合は、FLCC_PAGE_ADDR0 が常に FLCC_PAGE_ADDR1 以下となるようにする必要があります。それ以外ではコマンドが拒否されます（フラッシュ・メモリの有効なアドレス範囲を超えるアドレスを書き込むと、アドレス・レジスタが飽和してフラッシュ・メモリ空間のエイリアシングを防ぎます）。

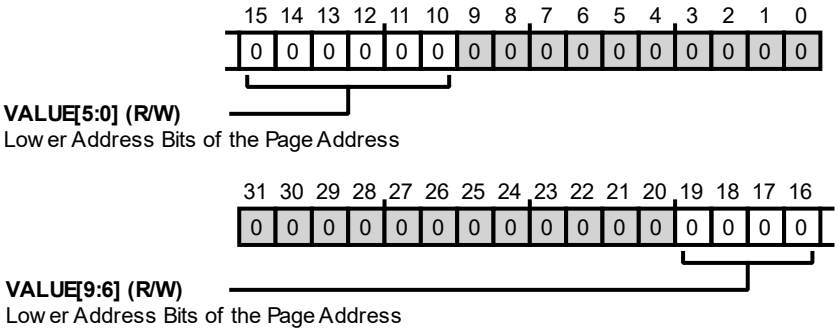


図 8-16：FLCC_PAGE_ADDR0 レジスタ図

表 8-17：FLCC_PAGE_ADDR0 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
19:10 (R/W)	VALUE	ページ・アドレスの下位アドレス・ビット。

上位ページ・アドレス

バイトが存在するページを選択するには、そのバイトのアドレスをこのレジスタへ書き込みます。選択されたページは、SIGN コマンド（シグネチャを計算するブロックの最終ページを選択）に使用できます。FLCC_PAGE_ADDR0 と FLCC_PAGE_ADDR1 の両方を使用するコマンドの場合は、FLCC_PAGE_ADDR0 が常に FLCC_PAGE_ADDR1 以下となるようにする必要があります。それ以外ではコマンドが拒否されます。

フラッシュ・メモリの有効なアドレス範囲を超えるアドレスを書き込むと、アドレス・レジスタが飽和してフラッシュ・メモリ空間のエイリアシングを防ぎます。

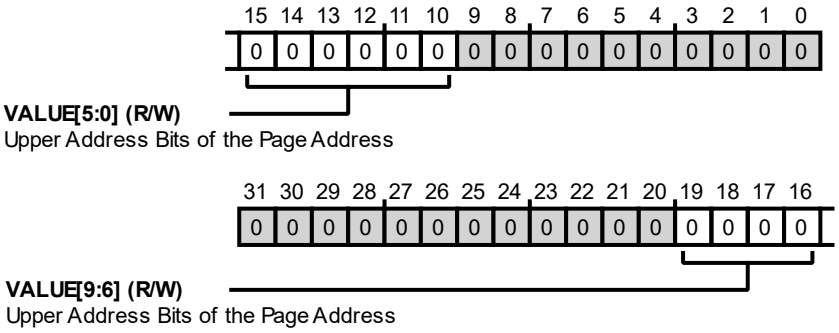


図 8-17：FLCC_PAGE_ADDR1 レジスタ図

表 8-18：FLCC_PAGE_ADDR1 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
19:10 (R/W)	VALUE	ページ・アドレスの上位アドレス・ビット。

シグネチャ

最後に作成されたシグネチャへ読出しアクセスを行います。

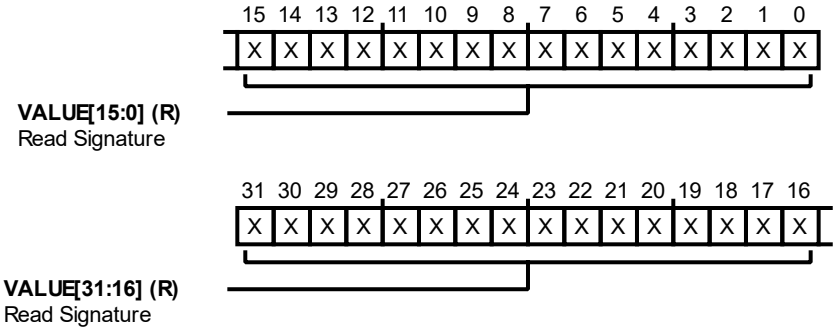


図 8-18：FLCC_SIGNATURE レジスタ図

表 8-19：FLCC_SIGNATURE レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	VALUE	シグネチャを読み出します。

ステータス

現在のコマンド状態とエラー検出／訂正に関する情報を提供します。

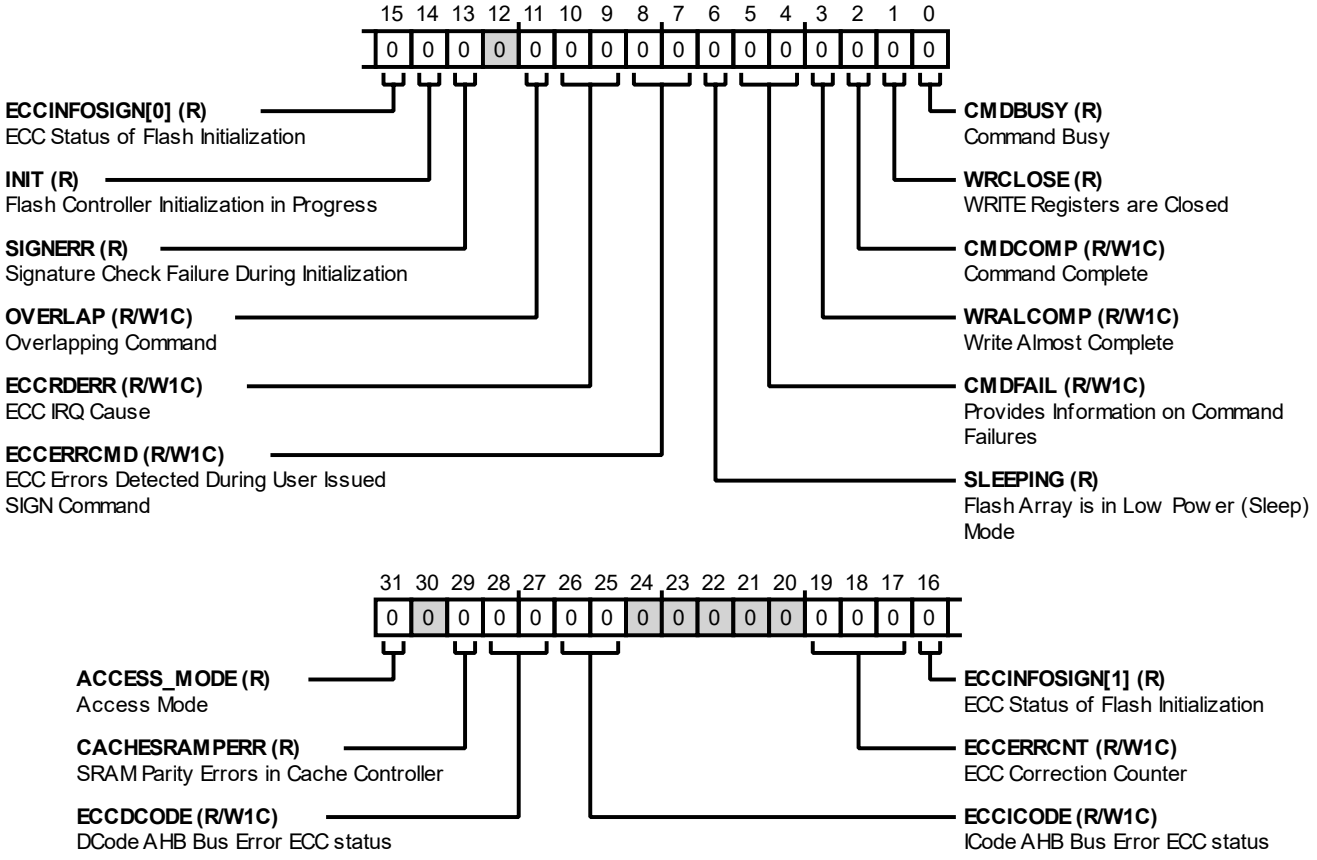


図 8-19：FLCC_STAT レジスタ図

表 8-20：FLCC_STAT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
31 (R/NW)	ACCESS_MODE	アクセス・モード。 コントローラの動作モードが直接アクセス（デフォルト）か間接アクセス（暗号化コマンドを使用）かを示します。	
		0	現在フラッシュ・コントローラは直接アクセス・モードです。すべてのレジスタへのユーザ・アクセスが許可されます。
		1	現在フラッシュ・コントローラは間接モードです。レジスタへのユーザ・アクセスは、ステータス・レジスタの読出しアクセスだけに制限されます。暗号化モジュールによるフラッシュ・コントローラの制御が解除されると（暗号化モジュールが、保護された鍵の保存領域内で現在進行している処理を完了すると）、再びすべてのレジスタへアクセスできるようになります。

表 8-20：FLCC_STAT レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
29 (R/NW)	CACHESRAMPERR	キャッシュ・コントローラ内の SRAM パリティ・エラー。 このレジスタは、ICODE バス上でのキャッシュ SRAM パリティ・エラーによって生成された AHB エラーの詳細を提供します。	
28:27 (R/W1C)	ECCDCODE	DCode AHB バス・エラーの ECC ステータス。 DCODE バス上での ECC エラーまたは訂正、もしくはその両方によって生成された AHB バス・エラーの詳細を提供します。	
		0	エラーなし。 リセット後またはレジスタのクリア後にエラーまたは訂正はレポートされていません。
		1	2 ビット・エラー。 AHB 読出しアクセス時に 2 ビット ECC エラーが検出され、レポートされました。
		2	1 ビット訂正。 AHB 読出しアクセス時に 1 ビット ECC 訂正が検出され、レポートされました。
26:25 (R/W1C)	ECCICODE	ICode AHB バス・エラーの ECC ステータス。 ICODE バス上での ECC エラーまたは訂正、もしくはその両方によって生成された AHB バス・エラーの詳細を提供します。	
		0	エラーなし。 リセット後またはレジスタのクリア後にエラーまたは訂正はレポートされていません。
		1	2 ビット・エラー。 AHB 読出しアクセス時に 2 ビット ECC エラーが検出され、レポートされました。
		2	1 ビット訂正。 AHB 読出しアクセス時に 1 ビット ECC 訂正が検出され、レポートされました。
19:17 (R/W1C)	ECCERRCNT	ECC 訂正カウンタ。 このカウンタは、重複する ECC 1 ビット訂正レポートを追跡します。ECC 訂正イベント発生時に IRQ または AHB バス・エラーを生成するように設定されている場合、このフィールドは、最初にレポートされた訂正の後に発生した ECC 訂正の回数をカウントします。カウンタは、オーバーフロー時にはフル・スケールのままになり、FLCC_STAT.ECCICODE または FLCC_STAT.ECCDCODE ステータス・ビットがクリアされると自動的にクリアされます。	

表 8-20：FLCC_STAT レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
16:15 (R/NW)	ECCINFOSIGN	フラッシュ初期化の ECC ステータス。 情報スペースでの自動シグネチャ・チェック終了後の ECC ステータス。	
		0	エラーなし。 エラーはレポートされていません。
		1	2 ビット・エラー。 シグネチャ・チェック中に 2 ビット ECC エラーが 2 回以上検出されました（シグネチャ・チェック不合格）。
		2	1 ビット・エラー。 シグネチャ・チェック中に 1 ビット ECC 訂正が 1 回以上検出されました。（チェックサムが一致している場合はシグネチャ・チェック合格）。
		3	1 ビットおよび 2 ビット・エラー。 シグネチャ・チェック中にそれぞれの ECC イベント（1 ビット訂正と 2 ビット・エラー）が少なくとも 1 回ずつ検出されました（シグネチャ・チェックは不合格）。
14 (R/NW)	INIT	フラッシュ・コントローラ初期化中。 フラッシュ・コントローラの初期化が進行中です。このビットのアサートが解除されるまで AHB アクセスは保留され、APB コマンドは無視されます。	
13 (R/NW)	SIGNERR	初期化中のシグネチャ・チェックに失敗。 フラッシュ・コントローラの初期化中、自動シグネチャ・チェックに失敗したことを示します。レジスタ値が有効になるのは、シグネチャ・チェックの完了後に限られます。	
11 (R/W1C)	OVERLAP	コマンドの重複。 あるコマンドがビジー状態にあるときに別のコマンドが要求されると、このビットがセットされます（重複コマンドは無視されます）。	

表 8-20：FLCC_STAT レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
10:9 (R/W1C)	ECCRDERR	ECC IRQ の原因。 このフィールドは、最近生成された割込みの原因をレポートします。コントローラは、FLCC_IEN.ECC_CORRECT と FLCC_IEN.ECC_ERROR に適切な値を書き込むことによって、1 ビットまたは 2 ビット ECC イベントに対して割込みを生成するように設定できます。これらのビットは、ユーザ・コードによってクリアされるまでスティッキー・ハイです。	
		0	エラーなし。
		1	2 ビット・エラー。 AHB 読出しアクセス時に、ECC エンジンが訂正不能な 2 ビット・エラーを検出しました。
		2	1 ビット訂正。 AHB 読出しアクセス時に、ECC エンジンが 1 ビット・エラーを訂正しました。
		3	1 ビット・イベントと 2 ビットイベント。 ECC エンジンが 1 ビット・データと 2 ビット・データ両方の破損を検出し、これらのエラーによって IRQ がトリガされました。（1 回の読出しでレポートできるイベント・タイプは 1 つだけです。このステータスは、その後に続く読出しアクセスが別の ECC エラー・イベントを発生させたことを示します）。 デフォルトでは、1 ビット ECC 訂正が IRQ としてレポートされ、2 ビット ECC エラーはバス異常としてレポートされます。両方のタイプを IRQ としてレポートすることは推奨しません。どの異常が最初に発生したのかを診断しようとしても、ステータス・ビットから判断できなくなるためです。

表 8-20：FLCC_STAT レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
8:7 (R/W1C)	ECCERRCMD	ユーザによる SIGN コマンド入力時に検出された ECC エラー。 ECC エラーがシグネチャ・コマンド実行時に生成された場合、そのエラーはこれらのビットによってレポートされます。これらのビットに基づいて割込みを生成するには、FLCC_IEN レジスタ内の対応ビットをユーザ・コードでセットする必要があります。	
		0	成功。 エラーは発生せず、シグネチャ・チェック時のフラッシュ読出し動作は正常に終了しました。
		1	2 ビット・エラー。 シグネチャ・コマンド実行中に、1 箇所以上のフラッシュ位置で 2 ビット・エラーが検出されました。エラーは訂正されていません。
		2	1 ビット・エラー。 シグネチャ・コマンド実行中に、1 箇所以上のフラッシュ位置で 1 ビット・エラーが訂正されました。
		3	1 ビットまたは 2 ビット・エラー。 シグネチャ・コマンド実行中に、1 箇所以上のフラッシュ位置で 1 ビット・エラーと 2 ビット・エラーが検出されました。
6 (R/NW)	SLEEPING	フラッシュ・アレイは低消費電力（スリープ）モードです。 フラッシュ・アレイが低消費電力（スリープ）モードにあることを示します。別のデータ・トランザクションのために必要な場合は、フラッシュ・コントローラが自動的にフラッシュをウェイクアップします。フラッシュは、FLCC_CMD レジスタに IDLE コマンドを書き込むことによって、いつでもウェイクアップできます。フラッシュのウェイクアップに要する時間は一定しませんが、通常は約 5µs です。性能を最適化するには、使用の約 5µs 前にフラッシュをウェイクアップする必要があります。	

表 8-20：FLCC_STAT レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
5:4 (R/W1C)	CMDFAIL	コマンド失敗に関する情報を提供します。 このフィールドは、実行完了時のコマンドのステータスを示します。これらのビットをクリアせずに複数のコマンドが実行された場合は、最初に発生したエラーだけが保存されます。	
		0	成功。 コマンドが正常に終了しました（例えば WRITE）。
		1	無視。 保護されたメモリ位置または範囲外のメモリ位置にアクセスしようとしました（このようなコマンドは無視されます）。
		2	検証エラー。 読出し検証エラーが発生しました。 このステータスが返される原因は 2 つあります。消去またはシグネチャ・チェック（もしくはその両方）の失敗です。 消去の失敗：フラッシュ・ページの消去後、コントローラは対応ワードを読み出して、消去が正常に完了したことを確認します。まだデータが残っている場合は消去失敗で、このフィールドは失敗をレポートします。 シグネチャ・チェックの失敗：Sign コマンドが実行された結果得られたシグネチャが、サイン・チェックを行ったブロックの最上位の 32 ビット・ワードに保存されているデータと一致しない場合、サイン・チェックは失敗となり、このフィールドが失敗をレポートします。
		3	アボート。 ユーザ・コードによってコマンドの実行が中止された（アボート・コマンドが入力された）こと、またはシステム割込みが生成されたことを示します。
3 (R/W1C)	WRALCOMP	書込みがほぼ完了。 実行中の書込みが間もなく完了するので、WRITE データ・レジスタへのアクセスのため、同レジスタが再度開かれました。FLCC_STAT.CMDCOMP がアサートされる前にもう 1 つの書込み動作を要求すると、バースト書込みとなります。バースト書込みは、フラッシュ・メモリの低レベル・プロトコルの利点を生かして性能を大幅に向上させます（それぞれの書込み動作を約 15µs 短縮）。 注：バースト書込みによって性能が向上するのは、フラッシュ・アレイの同一行内のバックツーバック書込みに限られます。	

表 8-20：FLCC_STAT レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応
2 (R/W1C)	CMDCOMP	コマンド完了。 このビットはコマンド完了時にアサートされます（新しいコマンドが要求されると自動的にクリアされる）。 注：パワーオン・リセット後、フラッシュ・コントローラは複数の動作を実行します（例えば、情報スペース内のコードの完全性確認）。このプロセスの終了時に、コントローラは FLCC_STAT.CMDCOMP ビットをセットし、プロセスが完了したことを示します。
1 (R/NW)	WRCLOSE	WRITE レジスタが閉じました。 WRITE データ／アドレス・レジスタとコマンド・レジスタが閉じて、アクセスできなくなりました。このビットは、書込み進行中に一時的にアサートされます。このビットがハイの場合、関係するレジスタはフラッシュ・コントローラが使用中で、書き込みできません。このビットは FLCC_STAT.WRALCOMP フラグがハイになるとクリアされ、進行中の書込みコマンドが対応するデータの書込みを完了して、これらのレジスタが上書き可能な状態になったことを示します。
0 (R/NW)	CMDBUSY	コマンド・ビジー。 このビットは、コマンド・レジスタを介して入力されたコマンドをフラッシュ・ブロックが能動的に実行しようとしているときにアサートされます。 注：コマンドを要求してからこのビットがアサートされるまでには、わずかな遅延があります。コマンドの完了をポーリングするユーザ・コマンドは、FLCC_STAT.CMDBUSY ビットではなく FLCC_STAT.CMDCOMP ビットをウォッチする必要があります。

時間パラメータ 0

このレジスタへの書込みには［ユーザ・キー］が必要です。書込みまたは消去コマンドの進行中にはこのレジスタを変更しないでください。このレジスタは、フラッシュ・メモリに送られる信号のタイミング制御に使用するレジスタのセットを定義します。デフォルト値は、26MHz のシステム・クロックと 13MHz で動作するリファレンス・クロックに合わせて設定されます。各タイミング・パラメータの値は、ユーザ・プログラマブル・ニブル（4 ビット）といくつかのハードコード・ビットで構成されます。ユーザ・プログラマブル・ビットは、各パラメータの最上位ビットです。

時間パラメータは、対応するフラッシュ・メモリ自体の時間的制約を満たすための待機時間をリファレンス・クロックの周期数で記述します。実際の遅延は、クロック領域をまたぐ場合の制約や、これらのパラメータで記述されていない信号の制約によって、わずかに増加します。時間パラメータ・レジスタを設定するときは、各制約について最短時間に近い値を選択する必要があります。

注：このレジスタの設定が不適切な場合、プログラム（PROGRAM）または消去（ERASE）動作中にフラッシュ・メモリを損傷させるおそれがあります。

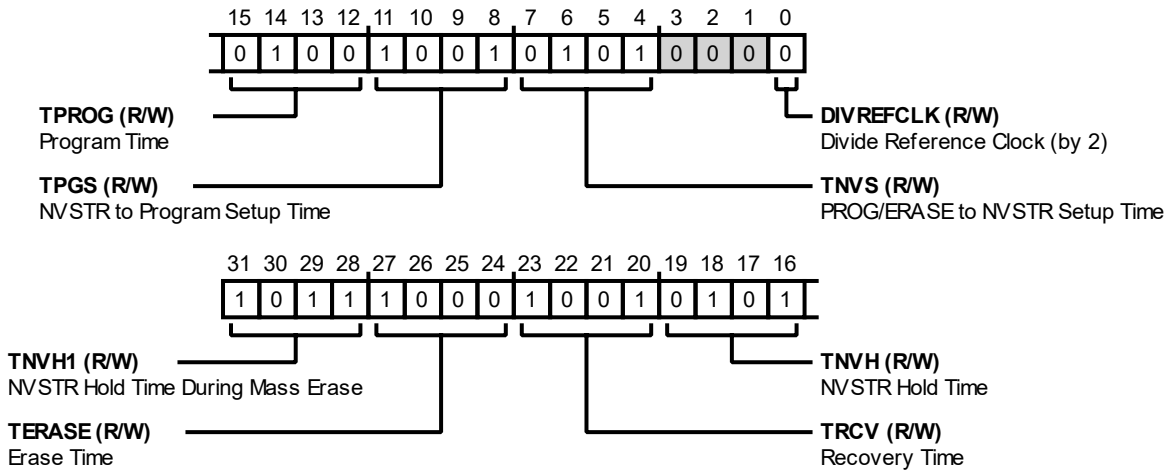


図 8-20：FLCC_TIME_PARAM0 レジスタ図

表 8-21：FLCC_TIME_PARAM0 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:28 (R/W)	TNVH1	一括消去時の NVSTR 保持時間。 タイマーにロードされる 11 ビット値の上位 4 ビットを決定します。下位ビットは 0x14 にハードコードされています。 13MHz の理想リファレンス・クロック使用時は以下のようになります。 最短［0x0］= 1.5µs デフォルト［0xB］= 110µs 最長［0xF］= 149µs

表 8-21：FLCC_TIME_PARAM0 レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
27:24 (R/W)	TERASE	消去時間。 タイマーにロードされる 19 ビット値の上位 4 ビットを決定します。下位ビットは 0x2000 にハードコードされています。 13MHz の理想リファレンス・クロック使用時は以下のようになります。 最短［0x0］= 0.6ms デフォルト［0x8］= 20.8ms 最長［0xF］= 38.4ms
23:20 (R/W)	TRCV	回復時間。 タイマーにロードされる 8 ビット値の上位 4 ビットを決定します。下位ビットは 0x2 にハードコードされています。 13MHz の理想リファレンス・クロック使用時は以下のようになります。 最短［0x0］= 154ns デフォルト［0x9］= 1.1µs 最長［0xF］= 18.6µs
19:16 (R/W)	TNVH	NVSTR ホールド時間 タイマーにロードされる 8 ビット値の上位 4 ビットを決定します。下位ビットは 0x1 にハードコードされています。 13MHz の理想リファレンス・クロック使用時は以下のようになります。 最短［0x0］= 77ns デフォルト［0x5］= 5.5µs 最長［0xF］= 18.5µs
15:12 (R/W)	TPROG	設定時間。 タイマーにロードされる 10 ビット値の上位 4 ビットを決定します。下位ビットは 0x0D にハードコードされています。 13MHz の理想リファレンス・クロック使用時は以下のようになります。 最短［0x0］= 1µs デフォルト［0x4］= 20.7µs 最長［0xF］= 75.3µs

表 8-21：FLCC_TIME_PARAM0 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応
11:8 (R/W)	TPGS	セットアップ時間を設定するための NVSTR。 タイマーにロードされる 8 ビット値の上位 4 ビットを決定します。下位ビットは 0x2 にハードコードされています。 13MHz の理想リファレンス・クロック使用時は以下のようになります。 最短［0x0］= 154ns デフォルト［0x9］= 1.1µs 最長［0xF］= 18.6µs
7:4 (R/W)	TNVS	NVSTR セットアップ時間のプログラム／消去（PROG/ERASE）。 タイマーにロードされる 8 ビット値の上位 4 ビットを決定します。下位ビットは 0x1 にハードコードされています。 13MHz の理想リファレンス・クロック使用時は以下のようになります。 最短［0x0］= 77ns デフォルト［0x5］= 5.5µs 最長［0xF］= 18.5µs
0 (R/W)	DIVREFCLK	リファレンス・クロックの分周（1/2）。 1 に設定すると、リファレンス・クロックは 2 分周されます。すべての時間パラメータはリファレンス・クロックが基準になります。クロックを分周すると、必要に応じて長い時間枠を選択することができます。 注：ユーザ・コードで時間パラメータを修正する必要が生じることはほとんどありません。必要になった場合はサポートが提供されます。

時間パラメータ 1

このレジスタへの書込みには［ユーザ・キー］が必要です。書込みまたは消去コマンドの進行中にはこのレジスタを変更しないでください。このレジスタは、フラッシュ・メモリに送られる信号のタイミング制御に使用するレジスタのセットを定義します。デフォルト値は、26MHz のシステム・クロックと 13MHz で動作するリファレンス・クロックに合わせて設定されます。各タイミング・パラメータの値は、ユーザ・プログラマブル・ニブル（4 ビット）といくつかのハードコード・ビットで構成されます。ユーザ・プログラマブル・ビットは、各パラメータの最上位ビットです。

時間パラメータは、対応するフラッシュ・メモリ自体の時間的制約を満たすための待機時間をリファレンス・クロックの周期数で記述します。実際の遅延は、クロック領域をまたぐ場合の制約や、これらのパラメータで記述されていない信号の制約によって、わずかに増加します。時間パラメータ・レジスタを設定するときは、各制約について最短時間に近い値を選択する必要があります。

注：このレジスタの設定が不適切な場合、プログラム（PROGRAM）または消去（ERASE）動作中にフラッシュ・メモリを損傷させる可能性があります。

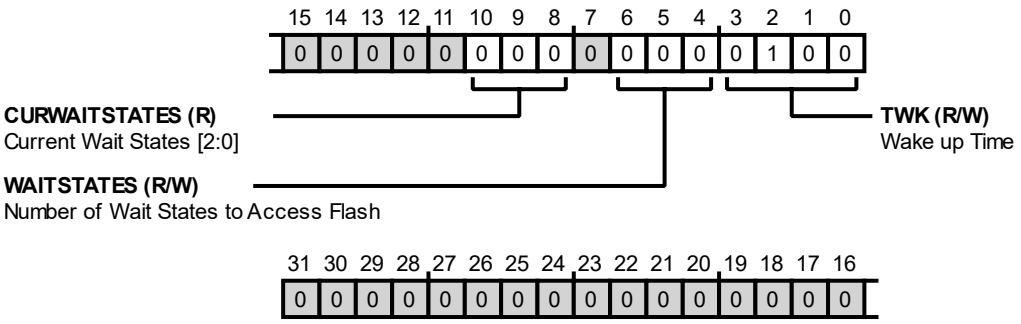


図 8-21：FLCC_TIME_PARAM1 レジスタ図

表 8-22：FLCC_TIME_PARAM1 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
10:8 (R/NW)	CURWAITSTATES	現在の待機状態［2：0］。 FLCC_TIME_PARAM1.WAITSTATES を変更すると、その変更が有効になるまでに数クロックの遅延があります。このコントローラは、現在有効なすべての読出し、書込み、または消去が完了するまで待機してから、待機状態の数を変更します。これらのビットは、FLCC_TIME_PARAM1.WAITSTATES［2：0］への変更が有効になっていて、安全にクロック周波数を変更できることを確認するために使用できます。

表 8-22：FLCC_TIME_PARAM1 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応
6:4 (R/W)	WAITSTATES	フラッシュへのアクセス待機状態の数。 このフィールドは、アナログ・デバイセズのキーが入力されるまで読出し専用です。 これは読出しに必要な待機状態の数を決定し、フラッシュ・アクセス時間とフラッシュ・コアへ送るシステム・クロックの周波数に依存します。WAITSTATES［2：0］が変更された場合は、FLCC_TIME_PARAM1.CURWAITSTATES が同じ値になるまでポーリングを行います。同じ数になった後でなければ、システム・クロック周波数を安全に変更することはできません。 これが設定できるのは 3'b100 までで、3'b100 を超える設定を行おうとしてもこのレジスタは更新されません。
3:0 (R/W)	TWK	ウェイクアップ時間。 タイマーにロードされる 8 ビット値の上位 4 ビットを決定します。下位ビットは 0xB にハードコードされています。 13MHz の理想リファレンス・クロック使用時は以下のようになります。 最短［0x0］= 847ns デフォルト［0x4］= 5.7µs 最長［0xF］= 19.3µs

ユーザ設定

ユーザ・キーが必要です。DMA 機能と自動インクリメント機能のユーザ制御を有効にするには、このレジスタへ書込みを行います。ユーザ・コードがこのレジスタへのアクセスを終了したときは、保護を再度アサートするために FLCC_KEY レジスタへガーベージ・データを書き込む必要があります。

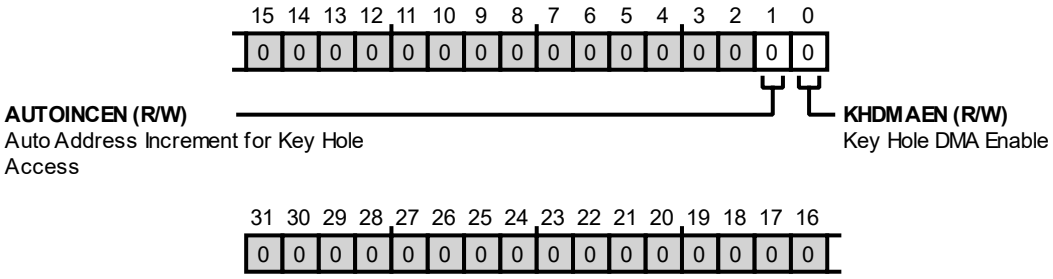


図 8-22：FLCC_UCFG レジスタ図

表 8-23：FLCC_UCFG レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
1 (R/W)	AUTOINCEN	キーホール・アクセスのための自動アドレス・インクリメント。 このビットをセットすると、それぞれの書込みコマンド実行時または読出しコマンド終了後に、FLCC_KH_ADDR が自動的に 0x8 ずつインクリメントされます。これにより、ユーザ・コードがそれぞれの書込みのためにフラッシュ・アドレスをその都度設定しなくても、一連のフラッシュ位置への書込みが可能になります。 書込みコマンド実行時に FLCC_STAT.WRALCOMP がアサートされるか、読出しコマンド実行後に FLCC_STAT.WRALCOMP がアサートされると、FLCC_KH_ADDR がインクリメントされます。これはユーザ・コードによって確認できます。 このビットがセットされると、ユーザ・コードで直接 FLCC_KH_ADDR を変更することはできません。

表 8-23：FLCC_UCFG レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応
0 (R/W)	KHDMAEN	キーホール DMA の有効化。 このビットがセットされると、フラッシュ・コントローラは DMA コントローラと連携して動作します。 この設定を行う場合は、事前にユーザ・コードで以下の操作をしておく必要があります。 - 開始アドレスを FLCC_KH_ADDR に書き込みます - FLCC_KH_DATA1 にデータを書き込むように DMA コントローラを設定します（アドレスを DWORD に合わせる必要があります） - 常に 32 ビット・ワードのペアを書き込むように DMA コントローラを設定します（R_Power = 1） - 整数値のデータ・ペアを書き込むように DMA コントローラを設定します（ワード数が奇数の場合は、DMA の助けを借りずにユーザ・コードが個別に 1 ワードを書き込む必要があります） 注：すべての DMA 書込みは、自動的にターゲット・アドレスをインクリメントします（FLCC_UCFG.AUTOINCEN の動作と同様）。DMA コントローラを使用できるのは、FLCC_KH_ADDR の値から始まる一連のアドレスを書き込む場合に限られます。 フラッシュ・コントローラは、DMA コントローラが書込み対象の 2 個のワードを提供するごとに、自動的に書込み動作を開始します。DMA コントローラとの連携動作はバースト書込みを使用するように設計されており、これによって全体的なプログラミング時間が大幅に短縮されます。

書込み保護

このレジスタを変更するにはユーザ・キーが必要です。FLCC_WRPROT レジスタは、デバイス起動時に自動的に設定されます。この設定時にはブートローダがユーザ・スペースからデータを読み込んで、このレジスタにロードします。

ユーザ・コードは、フラッシュ・メモリ内の該当位置に書込みを行うことによって、不揮発性の書込み保護に影響を与えます。デフォルトでは、これに関係するフラッシュ内の位置は 0x3FFF0 ですが（ユーザ・スペースの 4 番目の最上位ワード）、アナログ・デバイセズのセキュア・ブートローダによって位置を変更できます。

あるいは、ユーザ・コードでこのレジスタへ書込みを行うことによって、保護されていないブロックの保護をランタイムでアサートすることができます。ブロックには保護を追加できますが、保護を解除することはできません。リセットすると変更内容は失われます。このアプローチは、特にユーザ・コードの開発時に推奨されます。

すべての書込み保護はパワーオン・リセットでクリアされますが、フラッシュ・メモリへのユーザ・アクセスを許可する前に、ユーザ・スペースの FLCC_WRPROT ワードの定義に従ってアナログ・デバイセズのセキュア・ブートローダが再度書込み保護をアサートします。したがって、書込み保護の解除は、ユーザ・スペース最上位ページの ERASEPAGE コマンドか（その時点でページが保護されていない場合）、MASSERASE コマンドによってのみ行うことができます。MASSERASE コマンドが正常に実行されると、ユーザ・スペース内にあるページのすべての保護が直ちにクリアされます（ユーザは、デバイスをリセットすることなく、このような消去の直後にユーザ・スペースに書込みを行うことができます）。

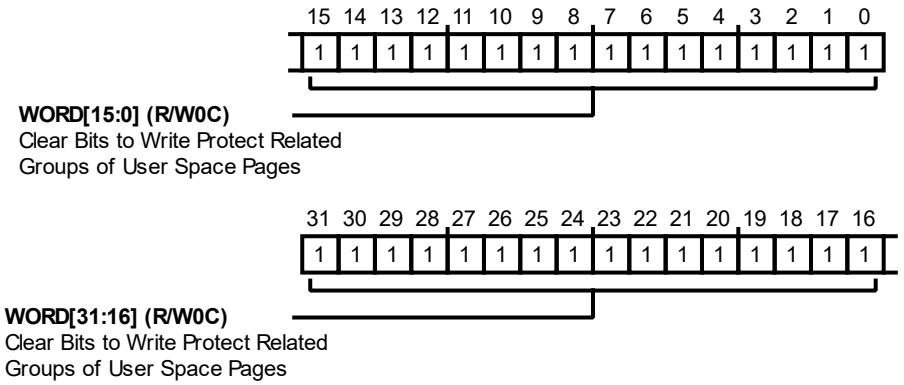


図 8-23：FLCC_WRPROT レジスタ図

表 8-24：FLCC_WRPROT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (R/W0C)	WORD	ユーザ・スペース・ページの関係グループを書込み保護するには、ビットをクリアします。 一度クリアした後は、デバイスをリセットしない限りこれらのビットを再びセットすることはできません。 この 32 ビット・ワードの各ビットは、使用可能な全ユーザ・スペースの 32 番目を表します。 2KB ページで構成される 512KB のデバイスでは（256 ページ）、各ビットは、8 ページからなるグループの書込み保護状態を表します。 2KB ページで構成される 256KB のデバイスでは（128 ページ）、各ビットは、4 ページからなるグループの書込み保護状態を表します。 2KB ページで構成される 128KB のデバイスでは（64 ページ）、各ビットは、2 ページからなるグループの書込み保護状態を表します。 このレジスタの最上位ビットは、ユーザ・スペース内のページの最上位グループを表します。

書込みアボート・アドレス

最後にアボートされた書込みコマンドのアドレス。このアドレスは、アボートされた書込みコマンドが再開された場合のみ書き込まれます。コマンドが十分に早い段階でアボートされてフラッシュ IP への影響がない場合、このアドレスは更新されません。

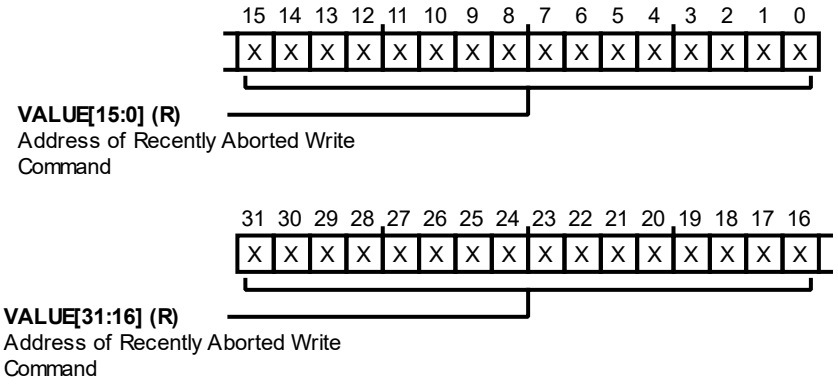


図 8-24：FLCC_WR_ABORT_ADDR レジスタ図

表 8-25：FLCC_WR_ABORT_ADDR レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	VALUE	最後にアボートされた書込みコマンドのアドレス。 このレジスタには現在実行中の書込みコマンドのターゲットとなるアドレスが格納され、アボート・イベント後もその値が維持されます。ユーザ・コードでこのレジスタを読み出せば、書込みアボートに影響を与えたフラッシュ位置を知ることができます。 新しいフラッシュ・コマンドが要求された場合、このレジスタの値が維持されるとは限らないため、書込みをアボートした直後にユーザ・コードでこの値を読み出す必要があります。

9 スタティック・ランダム・アクセス・メモリ（SRAM）

この章では、ADuCM4050 MCU の SRAM 機能の概要を説明します。

SRAM 機能

ADuCM4050 MCU で使用する SRAM は次の機能に対応しています。

- データ SRAM、命令 SRAM、キャッシュ SRAM 用の低消費電力コントローラ
- 使用可能な合計メモリ容量：128KB
- 休止モード時に保持される最大メモリ容量：124KB
- データ SRAM は 128KB で構成されます。16KB は休止モードで必ず保持されます。休止モード時に保持される最大メモリ容量は 124KB です。
- 命令 SRAM がイネーブルになっている場合、データ SRAM のうち 32KB が命令 SRAM にマップされ、アクセス可能なデータ SRAM は 96KB のみとなります。命令 SRAM には、休止モードで保持される容量として、0、12、16、28KB のオプションがあります。
- キャッシュ・コントローラをイネーブルにすると、命令 SRAM のうち 4KB がキャッシュ・データ用に予約されます。この 4KB のキャッシュ・データは休止モードでは保持されません。
- パリティ・ビット誤差検出（オプション）には、すべての SRAM メモリを使用できます。パリティ・チェックは様々なメモリ領域で有効／無効にするよう構成できます。
- バイト、ハーフワード、ワードのアクセスがサポートされています。

詳細については、SRAM 領域のセクションを参照してください。

SRAM の構成

命令 SRAM とデータ SRAM の関係

`PMG_TST_SRAM_CTL.INSTREN` ビットがアサートされている場合、32KB の SRAM が開始アドレス 0x1000_0000 に命令 SRAM としてマップされます（メモリ・マップの MODE0 を参照）。96KB のデータ SRAM は、0x2000_0000 を開始アドレスとするセクションと 0x2004_0000 を開始アドレスとするセクションの 2 つのセクションにマップされます。キャッシュ・メモリ機能を使用する場合は、命令 SRAM 用に使用できるのは 28KB のみとなります（メモリ・マップの MODE1 を参照）。

PMG_TST_SRAM_CTL.INSTREN ビットが 0 でキャッシュがディスエーブルの場合は、128KB の SRAM がデータ SRAM としてマップされます。メモリは、最初 32KB 分が開始アドレス 0x2000_0000 に、残りの 96KB 分が開始アドレス 0x2004_0000 にマップされます（メモリ・マップの MODE2 を参照）。キャッシュ・メモリ機能を使用する場合は、最初のセクションにマップされるのは 28KB のみです。そのため、使用可能なデータ SRAM は合計 124KB となります（メモリ・マップの MODE3 を参照）。

デフォルトでは、起動時とハードウェア・リセット時に 32KB の SRAM が命令 SRAM として使用可能となります。合計 128KB のデータ SRAM を使用するオプションを選択する必要がある場合、ユーザ・コードの最初の部分で PMG_TST_SRAM_CTL.INSTREN ビットをゼロと設定する必要があります。

キャッシュ・コントローラがイネーブルの場合、SRAM のバンク 2 にはアクセスできません。アクセスが試行されるとバス・エラー（マップされていないアドレス）が生成されます。

詳細については、SRAM 領域のセクションを参照してください。

休止モード時の SRAM 保持

休止モード時に保持される SRAM 容量に関しては、様々な設定が可能です。

0x2000_0000 にマップされたデータ SRAM の最初の 16KB（バンク 0）の内容は、常に保持されます。SRAM のバンク 2（4KB）は休止モードでは保持されません。これは動作モードに応じて、データ SRAM、命令 SRAM、またはキャッシュにマップされる場合があります。

PMG_SRAMRET.RET1 ビットがイネーブルの場合は、0x2000_4000～0x2000_6FFF（バンク 1）にマップされた 12KB のデータ SRAM が休止モード時に保持されます。命令 SRAM がイネーブルの場合、これは 0x1000_0000～0x1000_2FFF にマップされます。

PMG_SRAMRET.RET2 ビットがイネーブルの場合は、0x2004_0000～0x2004_7FFF（バンク 3、バンク 4）にマップされた 32 KB のデータ SRAM が保持されます。

PMG_SRAMRET.RET3 ビットがイネーブルの場合は、0x2004_8000～0x2004_FFFF（バンク 5）にマップされた 32KB のデータ SRAM が保持されます。

PMG_SRAMRET.RET4 ビットがイネーブルの場合は、0x2005_0000～0x2005_7FFF（バンク 6、バンク 7）にマップされた 32KB のデータ SRAM が保持されます。命令 SRAM がイネーブルの場合、バンク 7 は 0x1000_3000～0x1000_6FFF にマップされます。

PMG_SRAMRET.RET1、PMG_SRAMRET.RET2、PMG_SRAMRET.RET3、PMG_SRAMRET.RET4 の各ビットは PWRKEY で書込み保護されます。

SRAM プログラミング・モデル

データ SRAM および命令 SRAM の開始アドレスは、それぞれ 0x2000_0000 および 0x1000_0000 に設定されています。

SRAM パリティ

堅牢性を確保するため、すべての SRAM バンクまたはユーザが選択した SRAM バンクのグループについて、パリティ・チェックを有効にできます。バンク 0、1、2 の各バンクについては、パリティ・チェックにより 1 ワードあたり最大 2 個のエラーを検出でき、バイトやハーフワードの書込みを実行する前に、データがリードバックされます。

バンク 3〜7 については、パリティ・チェックにより 1 ワードあたり最大 4 個のエラーを検出でき、メモリは 1 サイクルでのバイト単位の書込みに対応します。そのため、これらのバンクではバイトやハーフワードの書込みに対してリードバックは行われません。

パリティ・チェック機能は、SRAM バンクごとに PMG_TST_SRAM_CTL.PENBNK0〜PMG_TST_SRAM_CTL.PENBNK5 のビットをアサートすることで有効にできます。プログラム・コードの最初でこれらのビットを設定する必要があります。

パリティ・チェックはデータの読出し時に実行されます。また、1 バイトやハーフワードのデータがバンク 0、1、2 に書き込まれる場合にも実行されます。1 ワード（32 ビット）の書込みが行われる場合には、パリティ・チェックは実行されません。パリティ・エラーが検出されるとバス・エラーが生成されます。バイトやハーフワードの書込み時にパリティ・エラーが検出された場合でも、書込み動作は最後まで行われ、パリティ・ビットは新規データに従って更新されます。バス・フォールト割込みルーチンでパリティ・エラーを管理する必要があります。

SRAM の初期化

パリティ・チェックが有効な場合、バンク 0、1、2 へのバイト書込みやハーフワード書込みを実行中に誤ったパリティ・エラーが発生するのを防ぐため、SRAM の内容が初期化されます。他のバンクは単一サイクルでのバイト書込みやハーフワード書込みに対応します。これらの初期化は不要です。バンク 7 を命令 SRAM として使用し、パリティ・チェックがこのバンクについて有効になっている場合は、初期化が必要です。専用ハードウェアによって選択した SRAM バンクを自動的に初期化することも可能です。このプロセスを完了するには 2048HCLK サイクルが必要です。このハードウェアは完全にプログラマブルなため、初期化の開始を自動でも手動でも行うことができます。

初期化によって、選択した SRAM バンクの内容は上書きされます。このプロセスはこれらの SRAM への書込み前に実行する必要があります。初期化シーケンスの間、初期化されている SRAM バンクへの書込みアクセスまたは読出しアクセスが検出された場合、初期化シーケンスが完了するまでバス・エラー応答が発生します。初期化が選択されていない SRAM バンクは、他のバンクの初期化プロセス中も通常どおりアクセス可能です。

特定の SRAM バンクの初期化が完了したことは、PMG_TST_SRAM_INITSTAT.BNK0DONE〜PMG_TST_SRAM_INITSTAT.BNK7DONE の対応するビットをポーリングすることでモニタできます。特定の SRAM バンクが初期化されるたびに、PMG_TST_SRAM_INITSTAT.BNK0DONE〜PMG_TST_SRAM_INITSTAT.BNK7DONE の関連するビットがクリアされ、初期化が完了するまでローを維持します。

起動後は、SRAM バンク 0、1、2、7 は自動的に初期化されます。SRAM バンク 0（16KB）は常に保持され、ここにはスタック・ポインタおよび重要な情報が保存されます。初期化が既に実行されているため、内容は上書きされません。ユーザ情報が既に保存されている可能性があるため、初期化済みの SRAM バンクは初期化しないでください。

休止後は、非保持の SRAM バンクの内容は失われます。バンク 1（非保持の場合）、バンク 2、バンク 7（非保持で命令 SRAM にマップされている場合）のパリティを有効にする場合は、初期化が必要です。

休止モード後に必要な SRAM バンクを初期化するには、次の 2 つの選択肢があります。

1. 休止モード終了後に PMG_TST_SRAM_CTL.STARTINIT レジスタに書き込むことで初期化します。1 にセットされている SRAM バンク（PMG_TST_SRAM_CTL.BNK0EN〜PMG_TST_SRAM_CTL.BNK7EN ビット）が初期化されます。

2. 休止モード後の自動初期化。休止モードが終了するたびに PMG_TST_SRAM_CTL に書き込む必要がありません。この自動モードを選択するには、事前に PMG_TST_SRAM_CTL.AUTOINIT ビットをセットしておく必要があります。このレジスタで初期化を選択した SRAM は休止モードの終了後、自動的に初期化されます。

初期化によって、選択したメモリ・バンクの内容はリセットされます。ユーザ情報が失われないよう、初期化するメモリ・バンクは適切に選択する必要があります。

初期化シーケンスは、PMG_TST_SRAM_CTL.ABTINIT ビットに書き込むことでいつでもアボートできます。このビットは書き込み後自動クリアされます。

キャッシュの初期化

キャッシュ・メモリを使用する場合、その関連する SRAM バンク（バンク 2）でパリティを有効にすることもできます。この場合（SRAM バンク 2 をキャッシュ・メモリとして使用する場合）、初期化は不要です。初期化が必要なのは、パリティ・チェックが有効になっている SRAM にバイトまたはハーフワードのアクセスを実行する場合のみです。SRAM バンク 2 がキャッシュ・メモリとして使用される場合、すべてのアクセスはワード・アクセスです。

キャッシュ・メモリに初期化が不要なため、休止モードから復帰後もキャッシュ機能を使用できます（初期化の時間ペナルティはなし）。予期しないバス・エラーを防止するため、キャッシュがイネーブルになっている場合、コントローラは SRAM バンク 2 の初期化をすべて無視します。

ADuCM4050 SRAM レジスタの説明

表 9-1：ADuCM4050 SRAM レジスタ・リスト

レジスタ名	説明	リセット	アクセス
PMG_TST_SRAM_INITSTAT	初期化ステータス・レジスタ	0x00000000	R/W
PMG_TST_SRAM_CTL	SRAM パリティと命令 SRAM の制御	0x80000000	R/W
PMG_SRAMRET	休止モードでの保持 SRAM の制御	0x00000000	R/W

初期化ステータス・レジスタ

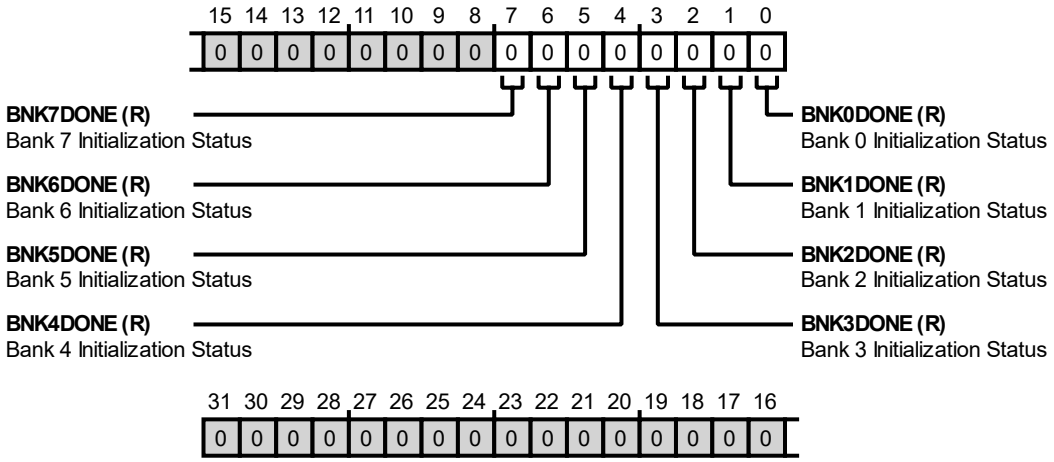


図 9-1：PMG_TST_SRAM_INITSTAT レジスタ図

表 9-2：PMG_TST_SRAM_INITSTAT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
7 (R/NW)	BNK7DONE	バンク 7 初期化ステータス	
		0	バンク 7 は未初期化
		1	バンク 7 は初期化済
6 (R/NW)	BNK6DONE	バンク 6 初期化ステータス	
		0	バンク 6 は未初期化
		1	バンク 6 は初期化済
5 (R/NW)	BNK5DONE	バンク 5 初期化ステータス	
		0	バンク 5 は未初期化
		1	バンク 5 は初期化済
4 (R/NW)	BNK4DONE	バンク 4 初期化ステータス	
		0	バンク 4 は未初期化
		1	バンク 4 は初期化済
3 (R/NW)	BNK3DONE	バンク 3 初期化ステータス	
		0	バンク 3 は未初期化
		1	バンク 3 は初期化済

表 9-2：PMG_TST_SRAM_INITSTAT レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
2 (R/NW)	BNK2DONE	バンク 2 初期化ステータス	
		0	バンク 2 は未初期化
		1	バンク 2 は初期化済
1 (R/NW)	BNK1DONE	バンク 1 初期化ステータス	
		0	バンク 1 は未初期化
		1	バンク 1 は初期化済
0 (R/NW)	BNK0DONE	バンク 0 初期化ステータス	
		0	バンク 0 は未初期化
		1	バンク 0 は初期化済

SRAM パリティと命令 SRAM の制御

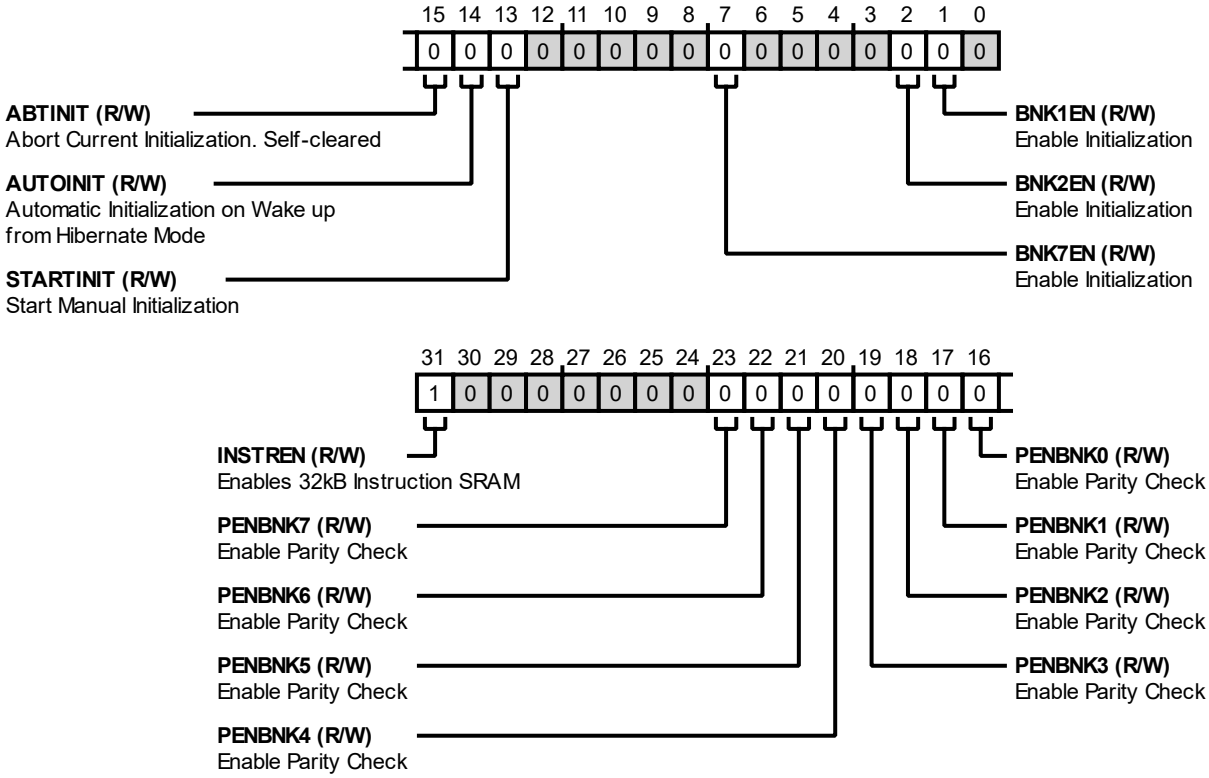


図 9-2：PMG_TST_SRAM_CTL レジスタ図

表 9-3：PMG_TST_SRAM_CTL レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31 (R/W)	INSTREN	32KB 命令 SRAM を有効化。
23 (R/W)	PENBNK7	パリティ・チェックを有効化。
22 (R/W)	PENBNK6	パリティ・チェックを有効化。
21 (R/W)	PENBNK5	パリティ・チェックを有効化。
20 (R/W)	PENBNK4	パリティ・チェックを有効化。
19 (R/W)	PENBNK3	パリティ・チェックを有効化。

表 9-3：PMG_TST_SRAM_CTL レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応
18 (R/W)	PENBNK2	パリティ・チェックを有効化。
17 (R/W)	PENBNK1	パリティ・チェックを有効化。
16 (R/W)	PENBNK0	パリティ・チェックを有効化。
15 (R/W)	ABTINIT	現在の初期化をアボート。自動クリア。
14 (R/W)	AUTOINIT	休止モードからの復帰時に自動初期化。
13 (R/W)	STARTINIT	1 を書き込み、初期化をトリガ。自動クリア。
7 (R/W)	BNK7EN	初期化を有効化。
2 (R/W)	BNK2EN	初期化を有効化。
1 (R/W)	BNK1EN	初期化を有効化。

休止モードでの保持 SRAM の制御

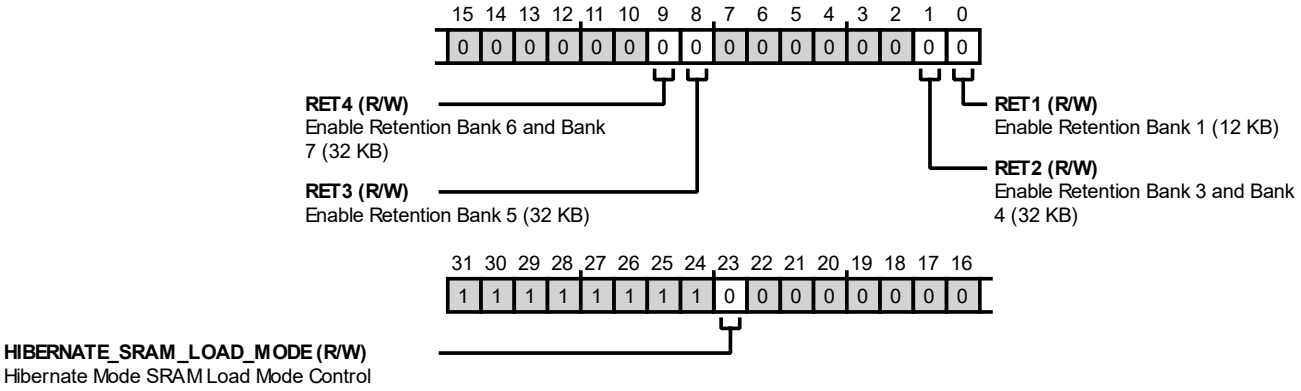


図 9-3：PMG_SRAMRET レジスタ図

表 9-4：PMG_SRAMRET レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
23 (R/W)	HIBERNATE_SRAM_LOAD_MODE	休止モードで SRAM 負荷モード制御。 休止モード時に適切な負荷電流がサポートされるよう設定します。
9 (R/W)	RET4	保持バンク 6 とバンク 7（32KB）をイネーブルにします。
8 (R/W)	RET3	保持バンク 5（32KB）をイネーブルにします。
1 (R/W)	RET2	保持バンク 3 とバンク 4（32KB）をイネーブルにします。
0 (R/W)	RET1	保持バンク 1（12KB）をイネーブルにします。

10 キャッシュ

キャッシュをイネーブルにすることで、フラッシュから実行するアプリケーションの性能が向上します。キャッシュ・メモリは SRAM と共存します。キャッシュがイネーブルになると、SRAM の一部がキャッシュに割り当てられます。そのため、キャッシュを他の目的に使用することはできません。命令キャッシュ（I キャッシュ）のサイズは 4KB です。休止からの復帰時に I キャッシュの内容は保持されません。

キャッシュ・プログラミング・モデル

起動時には命令キャッシュはディスエーブルになっています。

プログラミング・ガイドライン

キャッシュをイネーブルにする手順は次のとおりです。

1. `FLCC_CACHE_STAT.ICEN` ビットを読出し、キャッシュがディスエーブルであることを確認します。このビットがクリアされるまでポーリングします。
2. USER KEY（0xF123F456）を `FLCC_CACHE_KEY` レジスタに書き込みます。
3. `FLCC_CACHE_SETUP.ICEN` ビットをセットしてキャッシュをイネーブルにします。

ADuCM4050 FLCC_CACHE レジスタの説明

キャッシュ・コントローラ（FLCC_CACHE）には次のレジスタがあります。

表 10-1：ADuCM4050 FLCC_CACHE レジスタ・リスト

レジスタ名	説明
FLCC_CACHE_KEY	キャッシュ・キー・レジスタ
FLCC_CACHE_SETUP	キャッシュ・セットアップ・レジスタ
FLCC_CACHE_STAT	キャッシュ・ステータス・レジスタ

キャッシュ・キー・レジスタ

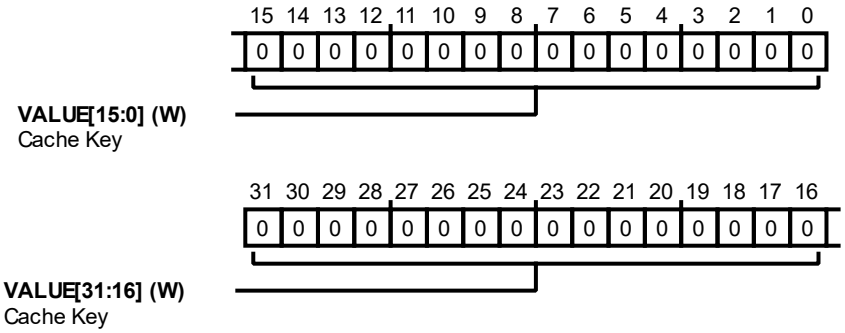


図 10-1：FLCC_CACHE_KEY レジスタ図

表 10-2：FLCC_CACHE_KEY レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	VALUE	キャッシュ・キー。 0xF123_F456 を入力して UserKey をセットします。読み出すと 0x0 が返されます。 FLCC_CACHE_SETUP レジスタに書込み後、このキーは自動的にクリアされます。

キャッシュ・セットアップ・レジスタ

このロケーションへの書込みを有効にするにはキャッシュの User key が必要です。キーはこのレジスタへの書込み後、クリアされます。

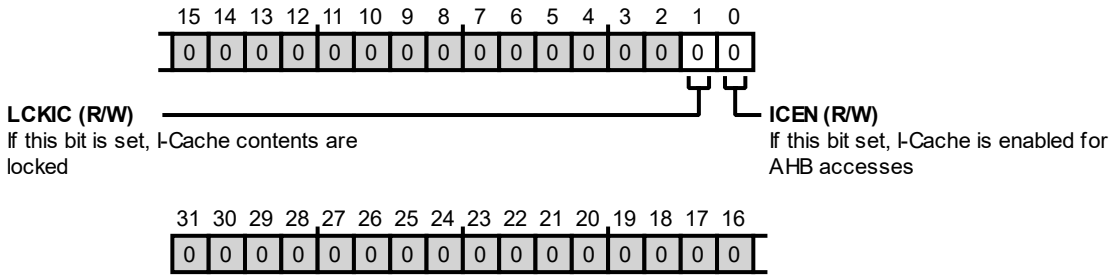


図 10-2：FLCC_CACHE_SETUP レジスタ図

表 10-3：FLCC_CACHE_SETUP レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
1 (R/W)	LCKIC	このビットをセットした場合、I キャッシュの内容はロックされます。 新たに発生したミスは I キャッシュでは置き換えられません。
0 (R/W)	ICEN	このビットをセットした場合、I キャッシュは AHB アクセスを有効にします。 0 の場合は、I キャッシュはディスエーブルになり、すべての AHB アクセスはフラッシュ・メモリ経由となります。このビットは Init ビットがセットされるとクリアされます。

キャッシュ・ステータス・レジスタ

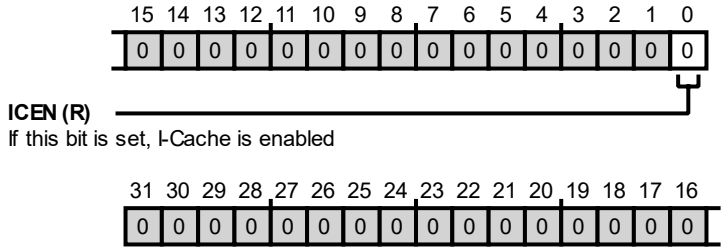


図 10-3：FLCC_CACHE_STAT レジスタ図

表 10-4：FLCC_CACHE_STAT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
0 (R/NW)	ICEN	このビットをセットした場合、I キャッシュはイネーブルになります。 このビットがクリアされると、I キャッシュはディスエーブルになります。

11 ダイレクト・メモリ・アクセス（DMA）

ダイレクト・メモリ・アクセス（DMA）コントローラは、データ転送タスクを実行し、それらを ADuCM4050 MCU からオフロードするために使用されます。DMA により、ペリフェラルとメモリ間の高速データ転送が実行されます。データは、CPU による動作を必要とせず、DMA によって迅速に移動可能です。このため、CPU リソースを他の動作に自由に使用できます。

DMA 機能

DMA は次の機能をサポートしています。

- 27 の独立した DMA チャンネル
- DMA チャンネルごとに 2 つの優先度をプログラム可能
- DMA チャンネル番号で決められた固有の優先度を使用した各優先度のアービトレーション
- 各 DMA チャンネルはプライマリおよび／または代替のチャンネル制御データ構造にアクセス可能
- 次の複数の転送タイプをサポート：
  - メモリからメモリ
  - メモリからペリフェラル
  - ペリフェラルからメモリ
- 次の複数の DMA サイクル・タイプをサポート：
  - ベーシック
  - 自動リクエスト
  - ピンポン
  - スキャッタギャザ
- 複数の DMA 転送データ幅（8 ビット、16 ビット、32 ビット）をサポート
- 各 DMA チャンネルは、ソースとディスティネーションのインクリメント／デクリメントを個別に制御可能

DMA 機能の説明

DMA は合計 27 チャンネルあり、各チャンネルはペリフェラルからのメモリ・アクセス要求の管理を専門に行います。DMA チャンネルは割り当てを示します。

各 DMA チャンネルは専用のハードウェア DMA リクエストに接続され、ソフトウェア・トリガも各チャンネルでサポートされています。この構成はソフトウェアによって行います。各 DMA チャンネルは、デフォルト優先度または高優先度に設定できます。各優先度は、DMA チャンネル番号によって決定される固有の優先度を使用してアービトレートされます。

DMA コントローラは複数の DMA 転送データ幅をサポートしています。ただし、ソースとディスティネーションの転送サイズは同じでなければなりません。また、ソースとディスティネーションのアドレスが常にデータ転送サイズに合っている必要があります。エラー状態の発生時にそれを示す 1 つの出力として、DMA エラー割込みがあります。エラー状態は、DMA 転送中にバス・エラーが生じた場合、または DMA コントローラが無効なサイクル制御を読み出した場合に発生する可能性があります。DMA コントローラは、メモリからメモリ、ペリフェラルからメモリ、およびメモリからペリフェラルへの転送をサポートし、フラッシュまたは SRAM をソースおよびディスティネーションとしてアクセスします。

表 11-1：DMA チャンネル

Channel Node	Peripheral
0	SPI2 Tx
1	SPI2 Rx
2	SPORT0A
3	SPORT0B
4	SPI0 Tx
5	SPI0 Rx
6	SPI1 Tx
7	SPI1 Rx
8	UART0 Tx
9	UART0 Rx
10	I2C Slave Tx
11	I2C Slave Rx
12	I2C Master
13	Crypto IN
14	Crypto OUT
15	Flash
16 to 23	Software DMA
24	ADC Rx
25	UART1 Tx
26	UART1 Rx

DMA アーキテクチャの概念

DMA チャンネルは、システム・インターフェースを使用して、メモリ空間の間またはメモリとペリフェラルの間でデータを転送する手段を提供します。DMA チャンネルは、システム全体にデータを分散する効率的な手段を提供し、コアを他の操作のために解放します。DMA 転送に対応する各ペリフェラルには、DMA 転送の動作モードを構成し制御するレジスタ・セットを持った、1 つまたは複数の専用 DMA チャンネルがあります。

DMA 動作モード

DMA コントローラには 2 つのバスがあり、1 つは Cortex-M4F コアと共有されるシステム・バスに接続され、もう 1 つのバスは 16 ビットのペリフェラルに接続されます。CPU と DMA が同じディスティネーション（メモリまたはペリフェラル）をターゲットにする場合など、一部のバス・サイクルでは DMA リクエストによってシステム・バスへの CPU アクセスが停止されることがあります。

DMA コントローラは、システム・メモリに配置されたチャンネル制御データ構造を読み出してデータ転送を実行します。DMA に対応したペリフェラルは、DMA の使用が可能になると、DMA コントローラに転送を要求できます。あるチャンネルに対し、設定した数の DMA 転送が終了した時点で、DMA コントローラは、そのチャンネルに対応する単一サイクルの dma_done 割込みを生成します。この割込みは DMA 転送の完了を示します。NVIC には、各 DMA チャンネルに対して個別の割込みイネーブル・ビットが用意されています。

チャンネル制御データ構造体

すべてのチャンネルには、プライマリ・データ構造体と代替データ構造体の 2 つの制御データ構造が関連付けられています。単純な転送モードの場合、DMA コントローラはプライマリまたは代替のいずれかのデータ構造体を使用します。ピンポンやスキャッタギャザなどのより複雑なデータ転送モードでは、DMA コントローラはプライマリと代替の両方のデータ構造体を使用します。各制御データ構造体（プライマリまたは代替）は、**チャンネル制御データ構造体**の表に示すように、メモリ内で 4 つの 32 ビット・ロケーションを占有します。

表 11-2：チャンネル制御データ構造体

オフセット	名前	説明
0x00	SRC_END_PTR	ソース・エンド・ポインタ
0x04	DST_END_PTR	ディスティネーション・エンド・ポインタ
0x08	CHNL_CFG	制御データの構成
0x0C	RESERVED	予備

プライマリ DMA 構造のメモリ・マップの図は、16 チャンネルの場合のすべてのチャンネル制御データ構造体を例示しています。これは 256 バイトのシステム・メモリを使用します。

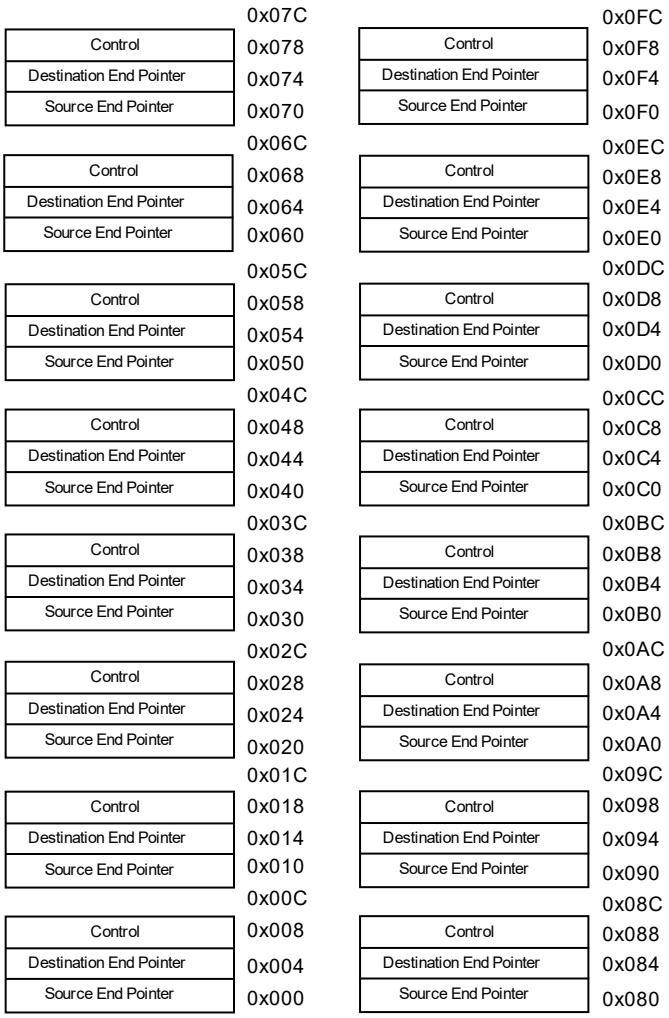


図 11-1：プライマリ DMA 構造のメモリ・マップ

コントローラが DMA 転送を実行する前に、DMA チャンネルに関連するデータ構造体をシステム・メモリの指定された場所に書き込む必要があります。ソース・エンド・ポインタのメモリ・ロケーションには、ソース・データの終了アドレスが含まれます。ディスティネーション・エンド・ポインタのメモリ・ロケーションには、ディスティネーション・データの終了アドレスが含まれます。コントロール・メモリ・ロケーションには、チャンネル構成制御データが含まれます。これにより、各アービトレーションのソースとディスティネーションのデータ・サイズ、転送数、およびデータ転送数が決定されます。

DMA コントローラは、チャンネルのリクエストを受け取ると、システム・メモリから対応するデータ構造体を内部キャッシュに読み込みます。`dma_done` 割込みまでにシステム・メモリ内のディスクリプタを更新すると、予期した動作が実行されないおそれがあります。`dma_done` を受信する前にディスクリプタを更新しないことを推奨します。

ソース・データ終了ポインタ

SRC_END_PTR のメモリ・ロケーションには、DMA 転送の一部としてデータが読み出される最後のロケーションのアドレスが格納されます。コントローラが DMA 転送を実行する前に、このメモリ・ロケーションをソース・データの終了アドレスに設定する必要があります。コントローラは、最初の DMA データ転送を開始する際にこのメモリ・ロケーションを読み出します。DMA コントローラがこのメモリ・ロケーションに書き込むことはありません。

表 11-3：ソース・データ終了ポインタ

ビット	名前	説明
[31:0]	SRC_END_PTR	ソース・データの終了アドレス。

ディスティネーション・データ終了ポインタ

DST_END_PTR のメモリ・ロケーションには、DMA 転送の一部としてデータが書き込まれる最後のロケーションのアドレスが格納されます。コントローラが DMA 転送を実行する前に、このメモリ・ロケーションをディスティネーション・データの終了アドレスに設定する必要があります。コントローラは、最初の DMA データ転送を開始するときにこのメモリ・ロケーションに書き込みます。DMA コントローラがこのメモリ・ロケーションを読み出すことはありません。

表 11-4：ディスティネーション・データ終了ポインタ

ビット	名前	説明
[31:0]	DST_END_PTR	ディスティネーション・データの終了アドレス。

制御データの構成

DMA 転送ごとに、制御データ構造体（CHNL_CFG）のメモリ・ロケーションにある DMA 転送の制御情報がコントローラに提供されます。

表 11-5：制御データの構成

ビット	名前	説明
31:30	DST_INC	ディスティネーション・アドレスのインクリメント。このアドレス・インクリメントは、次のようにソース・データ幅に依存します。 SRC_SIZE： 00：バイト 01：ハーフワード 10：ワード 11：インクリメントなし。アドレスは、DST_END_PTR メモリ・ロケーションに含まれる値に設定されたままです。
29:28	RESERVED	未定義。ゼロ書込み。

表 11-5：制御データの構成（続き）

ビット	名前	説明
27:26	SRC_INC	ソース・アドレスのインクリメント。このアドレス・インクリメントは、次のようにソース・データ幅に依存します。 SRC_SIZE： 00：バイト 01：ハーフワード 10：ワード 11：インクリメントなし。アドレスは、SRC_END_PTR メモリ・ロケーションに含まれる値に設定されたままです。
25:24	SRC_SIZE	ソース・データのサイズ。 00：バイト 01：ハーフワード 10：ワード 11：予備
23:18	RESERVED	未定義。ゼロ書込み。
17:14	R_Power	x 回の DMA 転送の後にアービトレーションを実行。
		0000：x = 1
		0001：x = 2
		0010：x = 4
		0011：x = 8
		0100：x = 16
		0101：x = 32
		0110：x = 64
		0111：x = 128
		1000：x = 256
		1001：x = 512
		1010～1111：x = 1024
		注：ソフトウェア DMA 転送では、0000～1111 の任意の値を使用できます。ペリフェラルが関わる DMA 転送では、暗号ブロックを除いて常に 0000 を使用する必要があります。詳細については、暗号化の章を参照してください。ペリフェラルが関わる DMA 転送で 0000 以外の値を設定した場合、DMA 動作は不定になります。
13:4	N_minus_1	現在の DMA サイクルでの転送の合計数 − 1。 この値は、DMA サイクルにおける転送の合計数を示すもので、合計バイト数ではありません。 使用可能な値は 0～1023 です。

表 11-5：制御データの構成（続き）

ビット	名前	説明
3	RESERVED	未定義。ゼロ書込み。
2:0	Cycle_ctrl	DMA サイクルの動作モード。
		000：停止（無効）
		001：ベーシック
		010：自動リクエスト
		011：ピンポン
		100：メモリ・スキャッタギャザ・プライマリ
		101：メモリ・スキャッタギャザ代替
		110：ペリフェラル・スキャッタギャザ・プライマリ
		111：ペリフェラル・スキャッタギャザ代替

DMA データ転送中にエラーが発生すると、CHNL_CFG がシステム・メモリに書き戻され、N_minus_1 フィールドが未完了の転送数を反映するように更新されます。全 DMA サイクルが完了すると、転送の完了を示すために cycle_ctrl ビットが無効になります。

DMA 優先順位

チャンネルの優先順位は、その番号と優先度によって決まります。各チャンネルには、デフォルトの優先度と高優先度の 2 つの優先度があります。高優先度のチャンネルはすべて、デフォルト優先度のどのチャンネルよりも優先順位が高くなります。同じ優先度では、番号の小さいチャンネルが番号の大きいチャンネルより優先順位が高くなります。DMA チャンネルの優先度は、DMA_PRI_SET レジスタの該当するビットに書き込むことにより変更できます。

DMA 転送タイプ

DMA コントローラは、プライマリおよび代替の制御データ構造体および DMA 転送を完了するためのリクエストの数に基づいて、様々なタイプの DMA 転送をサポートしています。DMA 転送タイプは、制御データ構造体の CHNL_CFG ロケーションにある cycle_ctrl ビットに適切な値をプログラムすることにより設定します。

cycle_ctrl ビットに基づいて、以下の DMA 転送タイプがサポートされています。

- 無効
- ベーシック
- 自動リクエスト
- ピンポン
- メモリ・スキャッタギャザ

- ペリフェラル・スキャッタギャザ

DMA 転送タイプの使用の表は、ペリフェラルおよびソフトウェア DMA で使用すべき様々な DMA 転送タイプを示しています。ペリフェラルにソフトウェア DMA 転送タイプを使用すること、およびその逆の使い方をすることは推奨されません。

表 11-6：DMA 転送タイプの使用

Software	Peripheral
Auto Request	Basic
Ping-Pong	Ping-Pong*1
Memory Scatter-Gather	Peripheral Scatter-Gather

*1 ソフトウェアのピンポン DMA 転送動作は、ペリフェラルのピンポン DMA 転送動作とは異なります。

無効

これは、当該チャンネルで DMA 転送が有効になっていないことを意味します。コントローラが DMA サイクルを完了した後、サイクル・タイプを無効に設定して、同じ DMA サイクルが繰り返されないようにします。いずれかのチャンネルで無効なディスクリプタを読み出した場合、DMA エラー割込みが生成され、`DMA_INVALIDDESC_CLR` の対応するビットがセットされます。

ベーシック

このモードでは、対応するチャンネルの `DMA_ALT_CLR` レジスタを設定することにより、プライマリまたは代替のデータ構造体のいずれかを使用するようにコントローラを構成できます。このモードはペリフェラルの DMA 転送に使用されます。このモードをペリフェラルに使用するときは、`CHNL_CFG` の R_Power を 0 に設定する必要があります。これは、ペリフェラルがデータ転送ごとにリクエストを送信する必要があることを意味します。

チャンネルをイネーブルすると、コントローラはリクエストを受信したときに以下の操作を実行します。

1. コントローラは転送を実行します。残りの転送数がゼロの場合は、**ステップ 3** に進みます。
2. コントローラは以下のようにアービトレートします。
   a. 優先順位の高いチャンネルがサービスをリクエストしている場合、コントローラはそのチャンネルにサービスを提供します。
   b. ペリフェラルまたはソフトウェアがコントローラにリクエストを送信した場合は、**ステップ 1** に進みます。
3. 転送の終了時に、コントローラは対応するチャンネルの `dma_done` 割込みを使用して MCU に割込みを生成します。

自動リクエスト

このモードでは、DMA コントローラが単一のリクエストを受信するだけで、DMA サイクル全体を完了させることができます。これにより、優先度のより高いリクエストを処理するために遅延が大幅に増加したり、MCU またはペリフェラルからの複数のリクエストを必要としたりすることなく、大きなデータの転送が可能となります。このモードは、ソフトウェア DMA リクエストを使用するメモリ間コピーのアプリケーションで役立ちます。

このモードでは、対応するチャンネルの `DMA_ALT_CLR` レジスタを設定することで、プライマリまたは代替のいずれかのデータ構造体を使用するようにコントローラを構成することができます。

チャンネルをイネーブルすると、コントローラはリクエストを受信したときに以下の操作を実行します。

1. コントローラは、チャンネルに対して min（2^{R_Power}, N）回、転送を実行します。残りの転送数がゼロの場合は、**ステップ 3** に進みます。
2. チャンネルのリクエストが自動的に生成されます。コントローラはアービトレーションを実行します。そのチャンネルの優先度が最も高い場合、DMA サイクルは**ステップ 1** に進みます。
3. 転送の最後に、コントローラはチャンネルの `dma_done` 割込みを使用して MCU に割込みを生成します。

ピンポン

このモードでは、コントローラはデータ構造体の 1 つを使用して DMA サイクルを実行してから、別のデータ構造体を使用して DMA サイクルを実行します。コントローラは、ベーシック／自動リクエストのデータ構造体を読み出すか、または MCU がチャンネルをディスエーブルするまで、プライマリと代替の切替えを継続します。

このモードは、メモリ内の異なるバッファを使用してデータを転送する場合に有用で、連続的なデータの送受信を遅延なく行うのに最適です。通常のアプリケーションでは、転送を開始する前に、MCU でプライマリおよび代替の両方のデータ構造体を設定する必要があります。ホストはその後、対応する転送が終了した際に、割込みサービス・ルーチン内のプライマリまたは代替の制御データ構造体を構成することができます。

DMA コントローラは、それぞれの制御データ構造体に関連した転送の完了後に、dma_done 割込みを使用して MCU に割込みを生成します。プライマリまたは代替の制御データ構造体を使用する個々の転送は、ベーシック DMA 転送と全く同じように機能します。

ソフトウェア・ピンポン DMA 転送

このモードでは、ソフトウェアから DMA リクエストが行われた場合、プライマリまたは代替のディスクリプタ・タスクが完了するまで、アービトレーション・サイクル後ごとにリクエストが自動的に生成されます。この最後のディスクリプタでは、自動リクエスト転送タイプを使用する必要があります。

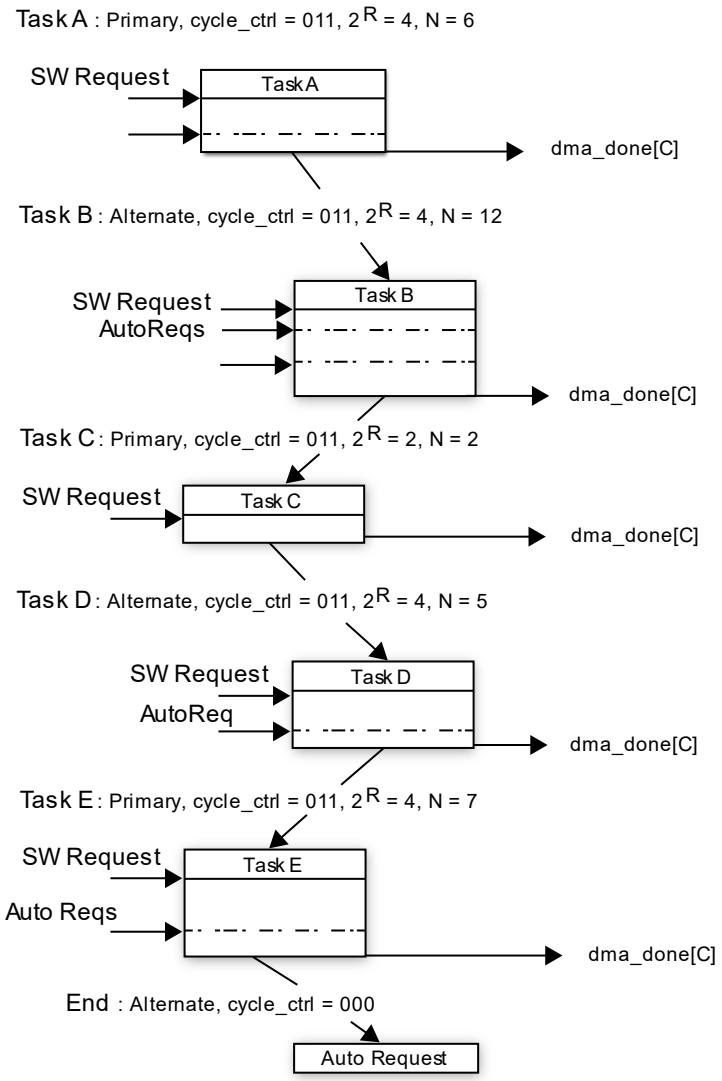


図 11-2：ソフトウェア・ピンポン DMA 転送

ペリフェラル・ピンポン DMA 転送

このモードでは、ペリフェラルから DMA リクエストが行われた場合、データ転送終了ごとに DMA リクエストを送信してプライマリまたは代替のディスクリプタ・タスクを完了させ、最後のディスクリプタでベーシック転送タイプを使用するように設定する必要があります。

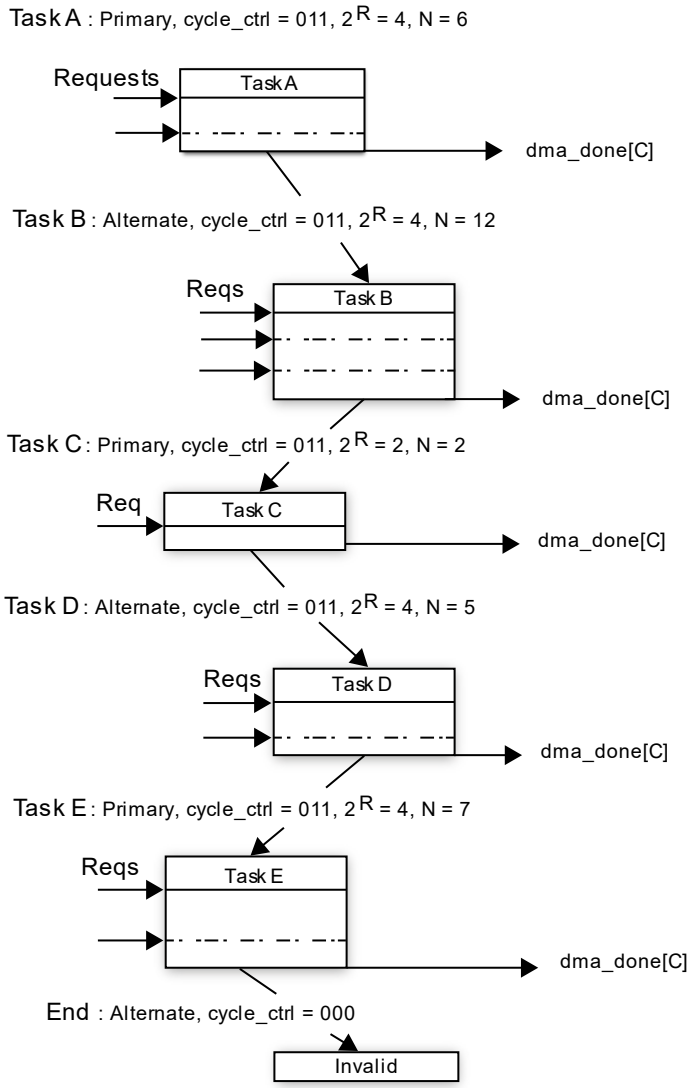


図 11-3：ペリフェラル・ピンポン DMA 転送

メモリ・スキャッタギャザ

このモードでは、DMA コントローラはプライマリと代替の両方のデータ構造体を使用するように構成する必要があります。DMA コントローラは、プライマリ・データ構造体を使用して代替データ構造体の制御構成を設定します。代替データ構造体は、自動リクエスト DMA 転送と同様の転送を使用するデータ転送に使用されます。コントローラはプライマリ転送が終了するたびにアービトレーションを行います。コントローラは、1 回のリクエストだけですべての転送を完了できます。このモードは、複数のメモリ間コピー・タスクを実行するときに使用します。MCU はそれらすべてをまとめて構成可能で、介入する必要はありません。DMA コントローラは、ベーシック・サイクルを使用してスキャッタギャザ・トランザクション全体が完了したときに、`dma_done` 割込みを使用して MCU に割込みを生成します。

このモードでは、コントローラは最初のリクエストを受け取ると、プライマリ・データ構造体を使用して DMA 転送を 4 回実行し、代替データ構造体の制御構造をプログラムします。この転送が完了すると、代替データ構造体を使用して

DMA サイクルが開始されます。サイクルの完了後、コントローラはプライマリ・データ構造体を使用して更に DMA 転送を 4 回実行します。

コントローラは、以下の状態になるまでプライマリから代替へ、更にプライマリへの切替えを継続します。

- MCU がベーシック・サイクルの代替データ構造体を構成するまで。

  または、

- DMA が無効なデータ構造体を読み込むまで。

表 11-7：メモリ・スキャッタギャザ・モードでのプライマリ・データ構造体の CHNL_CFG

ビット	名前	値	説明
31:30	DST_INC	10	アドレスに対してワード単位でアクセスするようにコントローラを設定します。
29:28	RESERVED	00	未定義。ゼロ書込み。
27:26	SRC_INC	10	アドレスに対してワード単位でアクセスするようにコントローラを設定します。
25:24	SRC_SIZE	10	ワード転送を使用するようにコントローラを設定します。
23:18	RESERVED	00	未定義。ゼロ書込み。
17:14	R_power	0010	DMA コントローラが転送を 4 回実行するように指定します。
13:4	N_minus_1	User	DMA 転送を N 回実行するようにコントローラを設定します（N は 4 の倍数）。
3	RESERVED	00	未定義。ゼロ書込み。
2:0	Cycle_ctrl	100	メモリ･スキャッタギャザ DMA サイクルを実行するようにコントローラを設定します。

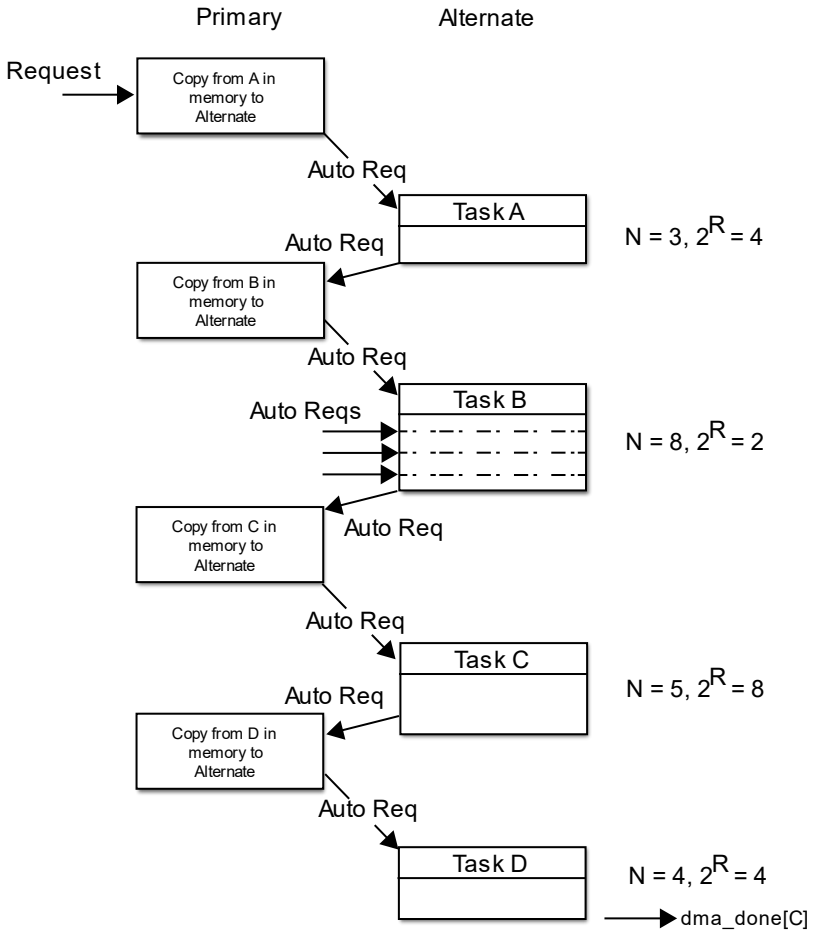


図 11-4：メモリ・スキャッタギャザ転送プロセス

ペリフェラル・スキャッタギャザ

このモードでは、プライマリ・データ構造体と代替データ構造体の両方を使用するように DMA コントローラを構成する必要があります。コントローラは、プライマリ・データ構造体を使用して、代替データ構造体の制御構造をプログラムします。実際のデータ転送には代替データ構造体が使用され、各転送はベーシック DMA 転送を伴う代替データ構造体を使用して行われます。コントローラはプライマリ転送ごとにアービトレーションは行いません。このモードは、複数のペリフェラルからメモリへの DMA タスクを実行する場合に使用します。MCU は、それらすべてを同時に構成可能で、介入する必要はありません。

このモードはメモリ・スキャッタギャザ・モードと同様ですが、アービトレーションとリクエストの条件が異なります。DMA コントローラは、ベーシック・サイクルを使用してスキャッタギャザ・トランザクション全体が完了したときに、dma_done 割込みを使用して MCU に割込みを生成します。

ペリフェラル・スキャッタギャザ・モードでは、コントローラはペリフェラルの最初のリクエストを受け取ってから、プライマリ・データ構造体を使用して DMA 転送を 4 回実行し、代替データ構造体の制御構造をプログラムします。次に、アービトレーションを再度行うことなく、代替データ構造体を使用して DMA サイクルを直ちに開始します。

このサイクルの完了後、コントローラは再度アービトレーションを行い、ペリフェラルからのリクエストを受信して、その優先度が最も高い場合、プライマリ・データ構造体を使用して次の DMA 転送を 4 回実行します。次に、アービトレーションを再度行うことなく、代替データ構造体を使用して DMA サイクルを直ちに開始します。

コントローラは、以下の状態になるまでプライマリから代替へ、更にプライマリへの切替えを継続します。

- MCU がベーシック・サイクルの代替データ構造体を構成するまで。

  または、

- DMA が無効なデータ構造体を読み出すまで。

表 11-8：ペリフェラル・スキャッタギャザ・モードでのプライマリ・データ構造体の CHNL_CFG

ビット	名前	値	説明
31:30	DST_INC	10	アドレスに対してワード単位でアクセスするようにコントローラを設定します。
29:28	RESERVED	00	未定義。ゼロ書込み。
27:26	SRC_INC	10	アドレスに対してワード単位でアクセスするようにコントローラを設定します。
25:24	SRC_SIZE	10	ワード転送を使用するようにコントローラを設定します。
23:18	RESERVED	00	未定義。ゼロ書込み。
17:14	R_power	0010	DMA コントローラが転送を 4 回実行するように指定します。
13:4	N_minus_1	user	DMA 転送を N 回実行するようにコントローラを設定します（N は 4 の倍数）。
3	RESERVED	00	未定義。ゼロ書込み。
2:0	Cycle_ctrl	110	メモリ・スキャッタギャザ DMA サイクルを実行するようにコントローラを設定します。

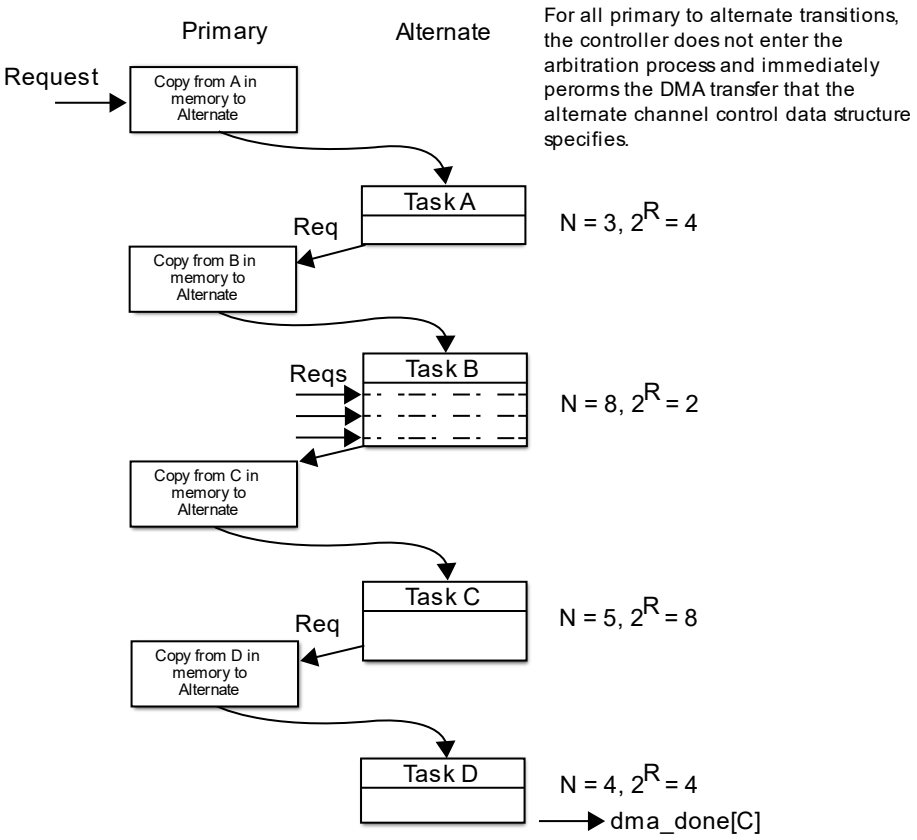


図 11-5：ペリフェラル・スキャッタギャザ転送プロセス

DMA の割込みと例外

エラー管理

DMA コントローラは、DMA エラーが発生するとコアにエラー割込みを生成します。DMA エラーは、バス・エラーまたは無効なディスクリプタ・フェッチによって発生する可能性があります。バス・エラーは、ディスクリプタのフェッチ時またはデータ転送中に発生します。また、予備のアドレス・ロケーションで読出しまたは書込みが発生すると、バス・エラーが発生することがあります。

バス・エラーが発生すると、不具合のあるチャンネルは自動的にディスエーブルされ、`DMA_ERRCHNL_CLR` の対応するステータス・ビットがセットされます。NVIC で DMA エラーが有効になっている場合も、エラーにより割込みが生成されます。更に、対応するチャンネルの `CHNL_CFG` データ構造体が最新の n_count で更新されます。これをチェックすることで、バス・エラーが発生する前に成功したデータ転送数を知ることができます。

コントローラが無効なディスクリプタをフェッチすると、不具合のあるチャンネルは自動的にディスエーブルされ、`DMA_INVALIDDESC_CLR` レジスタの対応するステータス・ビットがセットされます。NVIC で DMA エラーが有効になっている場合も、エラーにより割込みが生成されます。

アドレス計算

DMA コントローラは、`SRC_END_PTR`、`CHNL_CFG` のソース・アドレス・インクリメントの設定、および N_Minus_1 の現在の値に基づいて、ソースの読出しアドレスを計算します。

$$\text{Source read address} = \begin{cases} \text{SRC\_END\_PTR} - (\text{N\_minus\_1} << (\text{SRC\_INC})) \text{ for SRC\_INC} = 0, 1, 2 \\ \text{SRC\_END\_PTR for SRC\_INC} = 3 \end{cases}$$

同様に、ディスティネーションの書込みアドレスは、DST_END_PTR、CHNL_CFG のディスティネーション・アドレス・インクリメント設定、および N_Minus_1 の現在の値に基づいて計算されます。

$$\text{Destination write address} = \begin{cases} \text{DST\_END\_PTR} - (\text{N\_minus\_1} << (\text{DST\_INC})) \text{ for DST\_INC} = 0, 1, 2 \\ \text{DST\_END\_PTR for DST\_INC} = 3 \end{cases}$$

N_minus_1 は、実行する転送の現在のカウントです。

アドレス・デクリメント

アドレス・デクリメントは、ソース・アドレスとディスティネーション・アドレスに対して有効にすることができます。DMA_SRCADDR_SET レジスタの該当するビットをセットすることにより、チャンネルに対してソース・アドレス・デクリメントを有効にできます。同様に、DMA_DSTADDR_SET レジスタの必要なビットをセットすることにより、チャンネルに対してディスティネーション・アドレスのデクリメントを有効にできます。ソース・データ終了ポインタ（SRC_END_PTR）およびディスティネーション・データ終了ポインタ（DST_END_PTR）に書き込まれた値は、DMA サイクルの一部として最後の転送のアドレスとしても使用されます。ただし、開始アドレスは、ソース読出しまたはディスティネーション書込みのアドレス・インクリメント方式とは異なる方法で計算されます。

チャンネルの DMA_SRCADDR_SET レジスタのソース・デクリメント・ビットをセットした場合、ソース・アドレスは次のように計算されます。

$$\text{Source read address} = \begin{cases} \text{SRC\_END\_PTR} + (\text{N\_minus\_1} << (\text{SRC\_INC})) \text{ for SRC\_INC} = 0, 1, 2 \\ \text{SRC\_END\_PTR for SRC\_INC} = 3 \end{cases}$$

チャンネルの DMA_DSTADDR_SET レジスタのディスティネーション・デクリメント・ビットをセットした場合、ソース・アドレスは次のように計算されます。

$$\text{Destination write address} = \begin{cases} \text{DST\_END\_PTR} + (\text{N\_minus\_1} << (\text{DST\_INC})) \text{ for DST\_INC} = 0, 1, 2 \\ \text{DST\_END\_PTR for DST\_INC} = 3 \end{cases}$$

注：

N_minus_1 は、実行する転送の現在のカウントです。

バイト・スワップとアドレス・デクリメントは、いかなるチャンネルでも共用しないでください。共用すると、DMA データ転送動作は予測できなくなります。

イメージ・デクリメント図は、ソースとディスティネーションのデクリメントとそのデータ移動方向のすべての組み合わせを示しています。

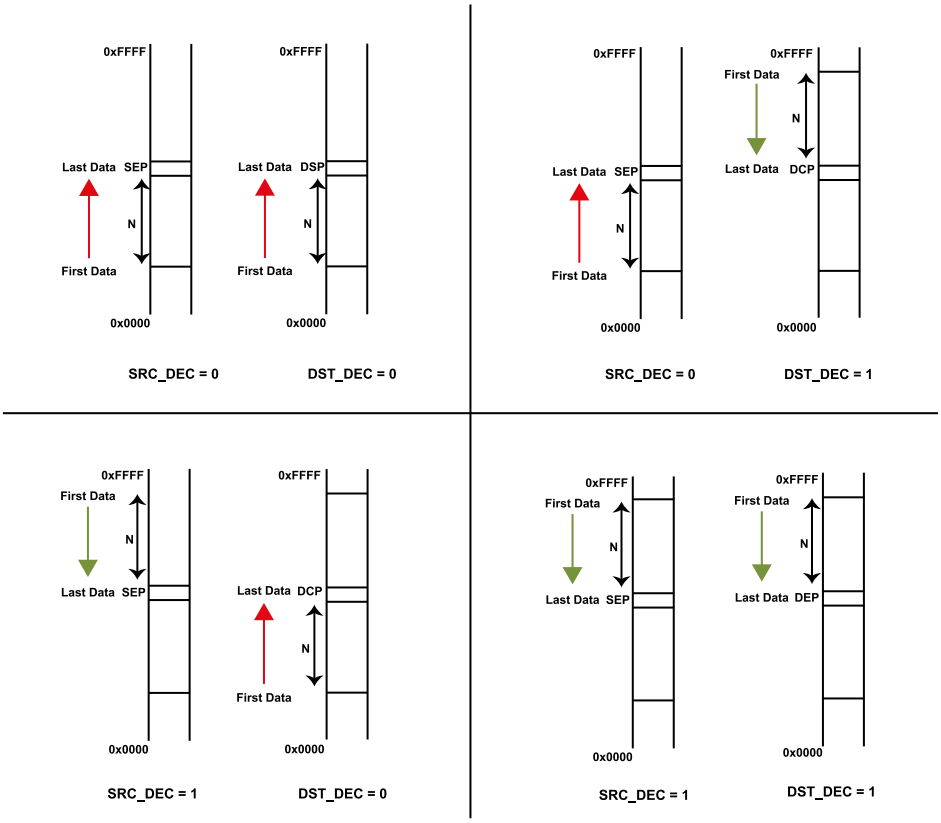


図 11-6：イメージ・デクリメント

エンディアン操作

DMA コントローラは、デフォルトでリトル・エンディアン方式の転送を行います。ただし、このデフォルト動作は、`DMA_BS_SET` レジスタの対応するチャンネル・ビットをセットすることにより変更できます。エンディアン操作はバイト・スワップと呼ばれます。

バイト・スワップの無効化

バイト・スワップはデフォルトで無効になっています。この場合、データ転送はリトル・エンディアンと見なされます。ペリフェラルから到着するデータは、32 ビット・ワードの LSB から順に配置されます。

例えば、SPI に 16 バイトのデータが 0x01（開始）、0x02、0x03、0x04 ... 0x0F、0x10 として到着した場合、DMA によってメモリに次のように格納されます。

1. `04_03_02_01`
2. `08_07_06_05`
3. `0C_0B_0A_09`
4. `10_0F_0E_0D`

バイト・スワップの有効化

バイト・スワップは、DMA_BS_SET レジスタの対応するビットをセットすることにより、任意のチャンネルで有効になります。ビットをセットすることにより、ビッグ・エンディアンのデータ転送が行われます。ペリフェラルから到着するデータは、32 ビット・ワードの MSB から順に配置されます。

例えば、SPI に 16 バイトのデータが 0x01（開始）、0x02、0x03、0x04 ... 0x0F、0x10 として到着した場合、DMA によってメモリに次のように格納されます。

1. 01_02_03_04
2. 05_06_07_08
3. 09_0A_0B_0C
4. 0D_0E_0F_10

注：

バイト・スワップは、32 ビットのデータ境界で発生します。転送サイズは 4 の倍数でなければなりません。

バイト・スワップとアドレス・デクリメントは、いかなるチャンネルでも共用しないでください。共用すると、DMA データ転送動作は予測できなくなります。

バイト・スワップを使用する場合、完全なデータ転送を行うためには、ソース・データ・アドレスが一定になるように注意する必要があります。例えば、SPI の場合、データ・ソースが常に同じロケーションから（FIFO）である必要があります。

バイト・スワップ機能は DMA 転送サイズとは無関係で、8 ビット、16 ビット、32 ビットのいずれでもかまいません。

DMA チャンネルのイネーブル／ディスエーブル

DMA チャンネルは、DMA リクエストを発行する前にイネーブルする必要があります。イネーブルされない場合、対応するチャンネルの DMA リクエストにより DONE 割込みが発生します。DMA チャンネルはいずれも、対応するビットを DMA_EN_SET レジスタに書き込むことによってイネーブルできます。DMA コントローラは、対応する dma_done 割込みが発生すると、チャンネルをディスエーブルします。ただし、DMA_EN_CLR レジスタの対応するビットに書き込むことで、任意のチャンネルをディスエーブルすることも可能です。

チャンネルがディスエーブルされると、DMA コントローラの現在の状態に基づいて以下が実行されます。

- チャンネルをディスエーブルし、そのチャンネルで保留中のリクエストがない場合は、直ちにディスエーブルされます。
- 未処理のチャンネルをディスエーブルしたにも関わらず、そのチャンネルのリクエストが送られた場合、その保留のリクエストはクリアされ、チャンネルは直ちにディスエーブルされます。
- アービトレーション後に選択したチャンネルをディスエーブルしたが、転送がまだ開始されていない場合、コントローラはアービトレーション・サイクルを完了させてからチャンネルをディスエーブルします。
- 処理中のチャンネルをディスエーブルすると、コントローラは現在のアービトレーション・サイクルを完了させてからチャンネルをディスエーブルします。

DMA マスタのイネーブル

DMA_CFG.MEN ビットは、DMA コントローラへのソフト・リセットとして機能します。DMA コントローラ内のアクティビティは、このビットが 1 にセットされている場合にのみ実行できます。このビットを 0 にクリアすると、コントローラ内のキャッシュされたディスクリプタがすべてクリアされ、コントローラがリセットされます。

パワーダウン・モードに関する考慮事項

チップをパワーダウンして休止モードにする前に、処理中のすべての DMA 転送を完了させます。ただし、できるだけ早く（現在のデータ転送を無視して）休止させる場合は、休止モードに入る前に DMA_CFG.MEN ビットをクリアして DMA コントローラをディスエーブルする必要があります。

注：DMA 転送の処理中に休止モードを選択すると、休止モードからの復帰時に転送が中断されます。DMA はディスエーブル状態に戻ります。

休止モード（または POR）の後で DMA_CFG.MEN ビットをセットすることにより、DMA を再度イネーブルする必要があります。

休止モードでは、以下のレジスタが保持されます。

- DMA_PDBPTR
- DMA_ADBPTR
- DMA_RMSK_SET
- DMA_RMSK_CLR
- DMA_PRI_SET
- DMA_PRI_CLR
- DMA_BS_SET
- DMA_BS_CLR
- DMA_SRCADDR_SET
- DMA_SRCADDR_CLR
- DMA_DSTADDR_SET
- DMA_DSTADDR_CLR

DMA のプログラミング・モデル

ここでは、DMA を構成するためのプログラミング・シーケンスについて説明します。

プログラミングのガイドライン

1. DMA_CFG レジスタで DMA コントローラをイネーブルします。
2. 目的の DMA チャンネルをイネーブルします。

3. DMA ベース・ポインタを設定します。
4. データ転送用の DMA ディスクリプタを設定します。
5. DMA_SWREQ レジスタでソフトウェア DMA リクエストを生成させるか、または DMA コントローラへの割込みを生成するペリフェラルをイネーブルします。

ADuCM4050 DMA レジスタの説明

DMA には以下のレジスタがあります。

表 11-9：ADuCM4050 DMA レジスター覧

レジスタ名	説明
DMA_ADBPTR	DMA チャンネルの代替制御データベース・ポインタ
DMA_ALT_CLR	DMA チャンネルのプライマリ代替をクリア
DMA_ALT_SET	DMA チャンネルでのプライマリ代替をセット
DMA_BS_CLR	DMA チャンネルのバイト・スワップ・イネーブルをクリア
DMA_BS_SET	DMA チャンネルのバイト・スワップ・イネーブルをセット
DMA_CFG	DMA 構成
DMA_DSTADDR_CLR	DMA チャンネルのディスティネーション・アドレス・デクリメント・イネーブルをクリア
DMA_DSTADDR_SET	DMA チャンネルのディスティネーション・アドレス・デクリメント・イネーブルをセット
DMA_EN_CLR	DMA チャンネル・イネーブルをクリア
DMA_EN_SET	DMA チャンネル・イネーブルをセット
DMA_ERRCHNL_CLR	DMA エラーをチャンネル単位でクリア
DMA_ERR_CLR	DMA バス・エラーをクリア
DMA_INVALIDDESC_CLR	DMA の無効なディスクリプタをチャンネル単位でクリア
DMA_PDBPTR	DMA チャンネルのプライマリ制御データベース・ポインタ
DMA_PRI_CLR	DMA チャンネルの優先度をクリア
DMA_PRI_SET	DMA チャンネルの優先度を設定
DMA_REVID	DMA コントローラのリビジョン ID
DMA_RMSK_CLR	DMA チャンネルのリクエスト・マスクをクリア
DMA_RMSK_SET	DMA チャンネルのリクエスト・マスクをセット
DMA_SRCADDR_CLR	DMA チャンネルのソース・アドレス・デクリメント・イネーブルをクリア
DMA_SRCADDR_SET	DMA チャンネルのソース・アドレス・デクリメント・イネーブルをセット
DMA_STAT	DMA ステータス
DMA_SWREQ	DMA チャンネルのソフトウェア・リクエスト

DMA チャンネルの代替制御データベース・ポインタ

DMA_ADBPTR 読出し専用レジスタは、代替チャンネル制御データ構造体のベース・アドレスを返します。このレジスタによって、アプリケーション・ソフトウェアが代替データ構造体のベース・アドレスを計算することが不要になります。DMA コントローラがリセット状態のときは、このレジスタを読み出すことはできません。

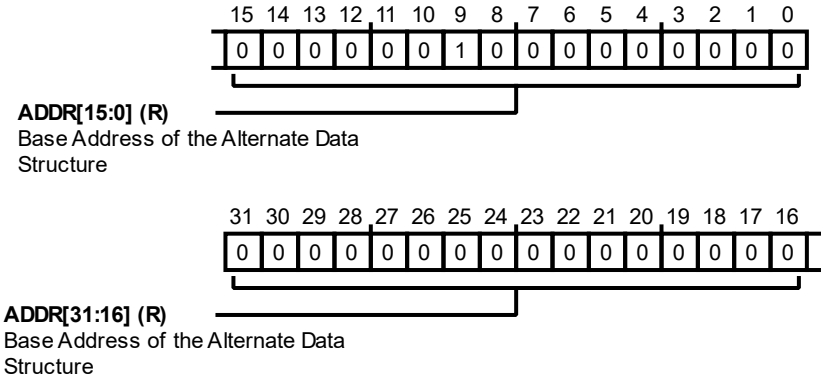


図 11-7：DMA_ADBPTR レジスタ図

表 11-10：DMA_ADBPTR レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	ADDR	代替データ構造体のベース・アドレス。

DMA チャンネルのプライマリ代替をクリア

DMA_ALT_CLR 書込み専用レジスタによって、プライマリ制御データ構造体を使用するように該当する DMA チャンネルを構成できます。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。

注：DMA コントローラは、ピンポン、メモリ・スキャッタギャザ、およびペリフェラル・スキャッタギャザの転送を行うために、必要に応じてこれらのビットを自動的にセット／クリアします。

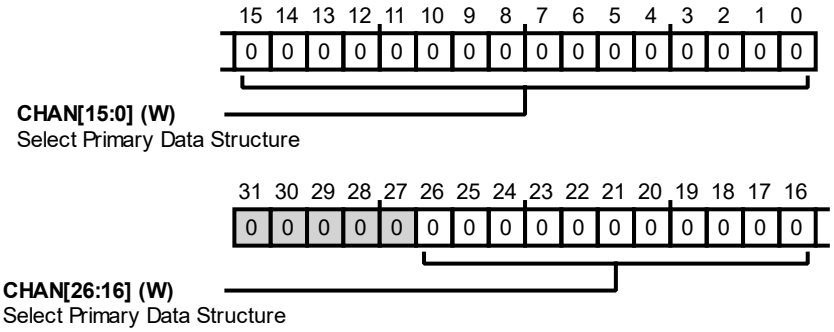


図 11-8：DMA_ALT_CLR レジスタ図

表 11-11：DMA_ALT_CLR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (RX/W)	CHAN	プライマリ・データ構造体の選択。 適切なビットをセットして、対応する DMA チャンネルのプライマリ・データ構造体を選択します。 ビット 0 は DMA チャンネル 0 に対応します。 ビット M-1 は、DMA チャンネル M-1 に対応します。 書込み時： DMA_ALT_CLR.CHAN［C］= 0、影響なし。代替データ構造体を選択するには、DMA_ALT_SET レジスタを使用します。 DMA_ALT_CLR.CHAN［C］= 1、チャンネル C のプライマリ・データ構造体を選択します。

DMA チャンネルでのプライマリ代替をセット

DMA_ALT_SET レジスタにより、代替制御データ構造体を使用するように該当する DMA チャンネルを構成できます。このレジスタを読み出すと、対応する DMA チャンネルに対して使用されているデータ構造体のステータスが返されます。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。

注：DMA コントローラは、ピンポン、メモリ・スキャッタギャザ、およびペリフェラル・スキャッタギャザの転送を行うために、これらのビットを必要に応じて自動的にセット／クリアします。

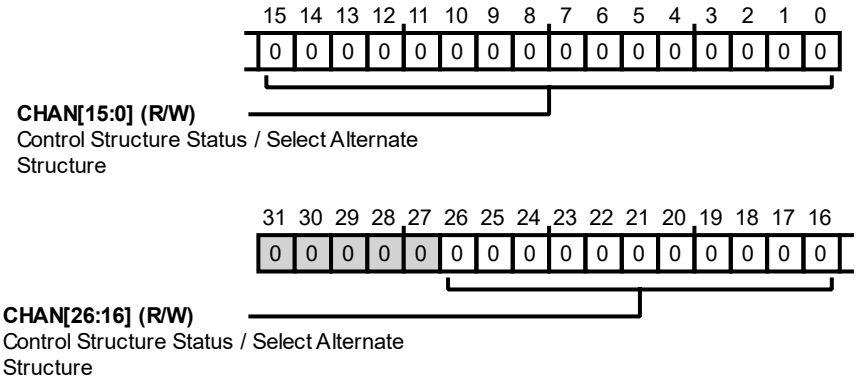


図 11-9：DMA_ALT_SET レジスタ図

表 11-12：DMA_ALT_SET レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
26:0 (R/W)	CHAN	制御構造体のステータス／代替構造体の選択。 チャンネル制御データ構造体のステータスを返すか、対応する DMA チャンネルの代替データ構造体を選択します。 ビット 0 は DMA チャンネル 0 に対応します。 ビット M-1 は、DMA チャンネル M-1 に対応します。 読出し時： DMA_ALT_SET.CHAN［C］= 0、DMA チャンネル C でプライマリ・データ構造体を使用しています。 DMA_ALT_SET.CHAN［C］= 1、DMA チャンネル C で代替データ構造体を使用しています。 書込み時： DMA_ALT_SET.CHAN［C］= 0、影響なし。ビット［C］を 0 に設定するには、DMA_ALT_CLR レジスタを使用します。 DMA_ALT_SET.CHAN［C］= 1、チャンネル C で代替データ構造体を選択します。

DMA チャンネルのバイト・スワップ・イネーブルをクリア

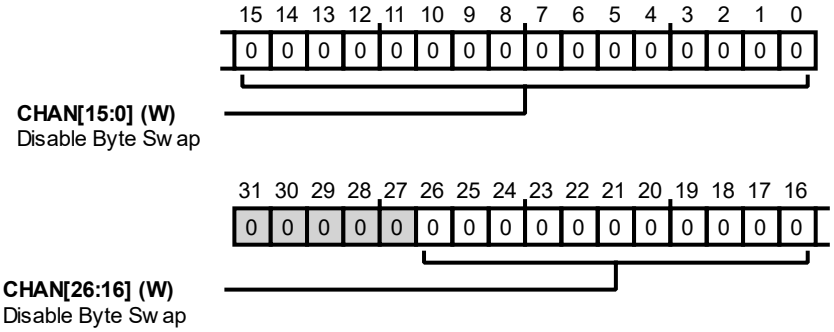


図 11-10：DMA_BS_CLR レジスタ図

表 11-13：DMA_BS_CLR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (RX/W)	CHAN	バイト・スワップを無効化。 DMA_BS_CLR 書込み専用レジスタを使用すると、バイト・スワップを使用しないように DMA チャンネルを構成したり、デフォルトの動作を使用したりできます。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。ビット 0 は DMA チャンネル 0 に対応し、ビット M-1 は DMA チャンネル M-1 に対応します。 書込み時： DMA_BS_CLR.CHAN［C］= 0、影響なし。チャンネル C でバイト・スワップを有効にするには DMA_BS_SET レジスタを使用します。 DMA_BS_CLR.CHAN［C］= 1、チャンネル C でバイト・スワップを無効にします。

DMA チャンネルのバイト・スワップ・イネーブルをセット

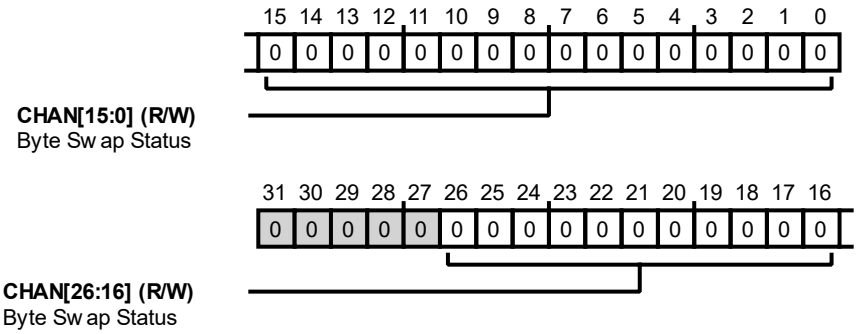


図 11-11：DMA_BS_SET レジスタ図

表 11-14：DMA_BS_SET レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (R/W)	CHAN	バイト・スワップのステータス。 このレジスタは、DMA チャンネルでバイト・スワップを使用するよう構成する場合に使用します。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。ビット 0 は DMA チャンネル 0 に対応し、ビット M-1 は DMA チャンネル M-1 に対応します。 読出し時： DMA_BS_SET.CHAN［C］= 0、チャンネル C でバイト・スワップが無効になっています。 DMA_BS_SET.CHAN［C］= 1、チャンネル C でバイト・スワップが有効になっています。 書込み時： DMA_BS_SET.CHAN［C］= 0、影響なし。チャンネル C でバイト・スワップを無効にするには、DMA_BS_CLR レジスタを使用します。 DMA_BS_SET.CHAN［C］= 1、チャンネル C でバイト・スワップを有効にします。

DMA 構成

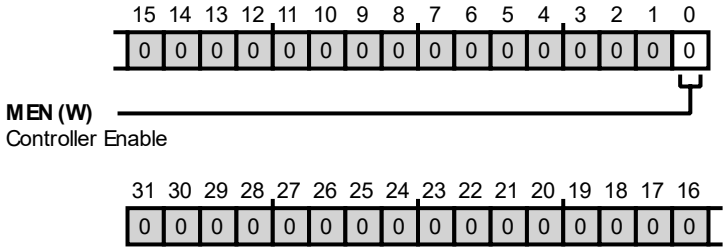


図 11-12：DMA_CFG レジスタ図

表 11-15：DMA_CFG レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
0 (RX/W)	MEN	コントローラのイネーブル。	
		0	コントローラをディスエーブル
		1	コントローラをイネーブル

DMA チャンネルのディスティネーション・アドレス・デクリメント・イネーブルをクリア

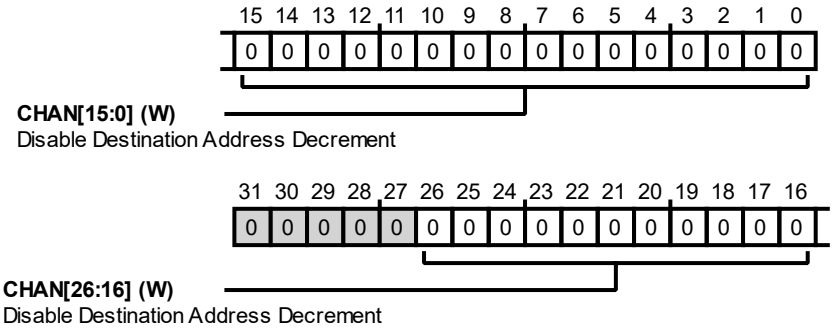


図 11-13：DMA_DSTADDR_CLR レジスタ図

表 11-16：DMA_DSTADDR_CLR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (RX/W)	CHAN	ディスティネーション・アドレスのデクリメントを無効化。 DMA_DSTADDR_CLR 書込み専用レジスタを使用すると、ディスティネーション・アドレスをデフォルトのインクリメント・モードで使用するように DMA チャンネルを構成できます。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。 ビット 0 は DMA チャンネル 0 に対応します。 ビット M-1 は、DMA チャンネル M-1 に対応します。 書込み時： DMA_DSTADDR_CLR.CHAN［C］= 0、影響なし。チャンネル C でディスティネーション・アドレスのデクリメントを有効にするには、DMA_DSTADDR_SET レジスタを使用します。 DMA_DSTADDR_CLR.CHAN［C］= 1、チャンネル C でディスティネーション・アドレスのデクリメントを無効にします。

DMA チャンネルのディスティネーション・アドレス・デクリメント・イネーブルをセット

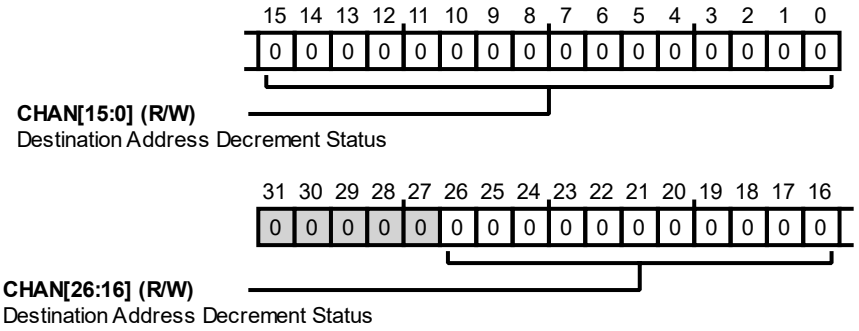


図 11-14：DMA_DSTADDR_SET レジスタ図

表 11-17：DMA_DSTADDR_SET レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (R/W)	CHAN	ディスティネーション・アドレス・デクリメントのステータス。 DMA_DSTADDR_SET レジスタは、DMA チャンネルのディスティネーション・アドレスを、各アクセス後にインクリメントではなくデクリメントするように構成するために使用します。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。ビット 0 は DMA チャンネル 0 に対応し、ビット M-1 は DMA チャンネル M-1 に対応します。 読出し時： DMA_DSTADDR_SET.CHAN［C］= 0、チャンネル C でディスティネーション・アドレスのデクリメントが無効になっています。 DMA_DSTADDR_SET.CHAN［C］= 1、チャンネル C でディスティネーション・アドレスのデクリメントが有効になっています。 書込み時： DMA_DSTADDR_SET.CHAN［C］= 0、影響なし。チャンネル C でディスティネーション・アドレスのデクリメントを無効にするには、DMA_DSTADDR_CLR レジスタを使用します。 DMA_DSTADDR_SET.CHAN［C］= 1、チャンネル C でディスティネーション・アドレスのデクリメントを有効にします。

DMA チャンネル・イネーブルをクリア

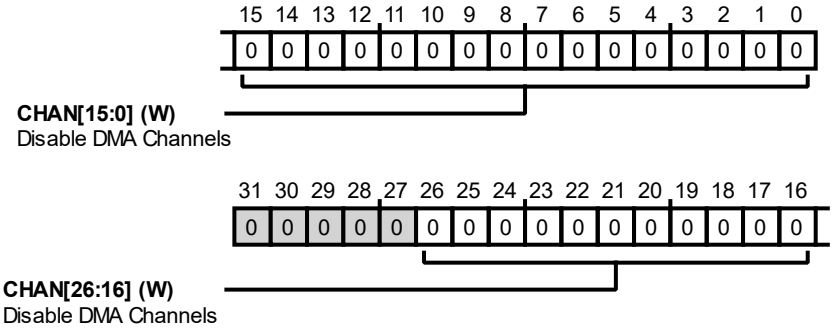


図 11-15：DMA_EN_CLR レジスタ図

表 11-18：DMA_EN_CLR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (RX/W)	CHAN	DMA チャンネルをディスエーブル。 このレジスタで DMA チャンネルをディスエーブルできます。このレジスタは書込み専用です。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。注：コントローラは、DMA サイクルを完了すると、適切なビットをセットしてチャンネルを自動的にディスエーブルします。適切なビットをセットすると、対応するチャンネルをディスエーブルします。 ビット 0 は DMA チャンネル 0 に対応します。 ビット M-1 は、DMA チャンネル M-1 に対応します。 書込み時： DMA_EN_CLR.CHAN［C］= 0、影響なし。チャンネルをイネーブルするには DMA_EN_SET レジスタを使用します。 DMA_EN_CLR.CHAN［C］= 1、チャンネル C をディスエーブルします。

DMA チャンネル・イネーブルをセット

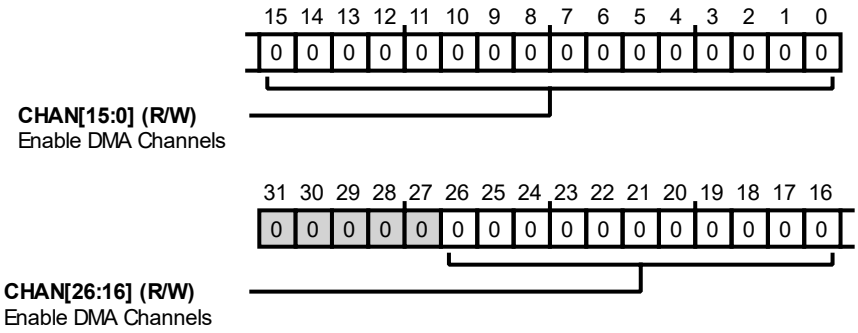


図 11-16：DMA_EN_SET レジスタ図

表 11-19：DMA_EN_SET レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (R/W)	CHAN	DMA チャンネルをイネーブル。 このレジスタで DMA チャンネルをイネーブルできます。このレジスタを読み出すと、チャンネルのイネーブル・ステータスが返されます。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。適切なビットをセットすると、対応するチャンネルがイネーブルされます。 ビット 0 は DMA チャンネル 0 に対応します。 ビット M-1 は、DMA チャンネル M-1 に対応します。 読出し時： DMA_EN_SET.CHAN［C］= 0、チャンネル C はディスエーブルされています。 DMA_EN_SET.CHAN［C］= 1、チャンネル C はイネーブルされています。 書込み時： DMA_EN_SET.CHAN［C］= 0、影響なし。チャンネルをディスエーブルするには、DMA_EN_CLR レジスタを使用します。 DMA_EN_SET.CHAN［C］= 1、チャンネル C をイネーブルします。

DMA エラーをチャンネル単位でクリア

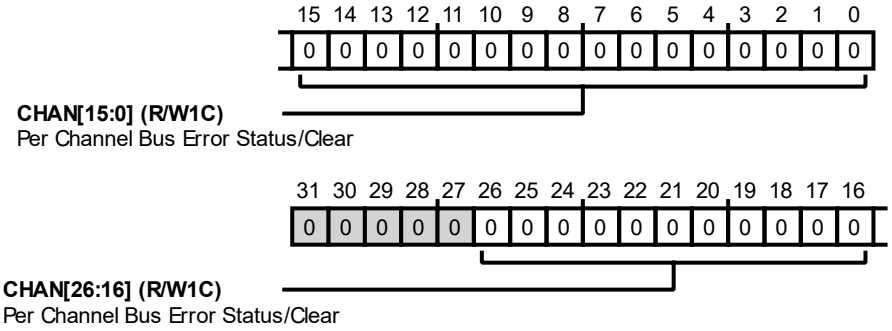


図 11-17：DMA_ERRCHNL_CLR レジスタ図

表 11-20：DMA_ERRCHNL_CLR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (R/W1C)	CHAN	チャンネル単位でのバス・エラー・ステータス／クリア。 このレジスタは、DMA バスのエラー状態をチャンネル単位で読出しおよびクリアするのに使用します。エラー状態は、コントローラが転送中にバス・エラーを検出した場合にセットされます。あるチャンネルでバス・エラーが発生した場合、そのチャンネルはコントローラによって自動的にディスエーブルされます。他のチャンネルは影響を受けません。ビットをクリアするには 1 を書き込みます。 読出し時： 0：バス・エラーは発生していません。 1：バス・エラー制御が保留中です。 書込み時： 0：影響なし。 1：ビットをクリア。

DMA バス・エラーをクリア

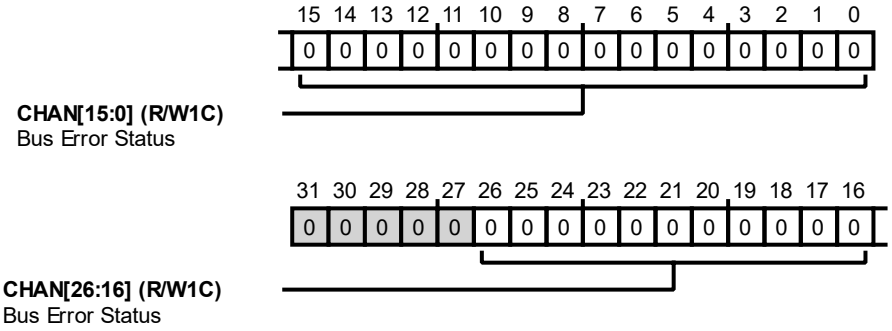


図 11-18：DMA_ERR_CLR レジスタ図

表 11-21：DMA_ERR_CLR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (R/W1C)	CHAN	バス・エラー・ステータス。 このレジスタは、DMA バス・エラー・ステータスの読出しおよびクリアに使用します。エラー・ステータスは、コントローラが転送中にバス・エラーを検出した場合、または無効なディスクリプタ（サイクル制御が 3'b000）を読み出した場合にセットされます。バス・エラーが発生した場合、またはチャンネルで無効なサイクル制御を読み出した場合、そのチャンネルはコントローラによって自動的にディスエーブルされます。他のチャンネルは影響を受けません。ビットをクリアするには 1 を書き込みます。 読出し時： 0：バス・エラー／無効なサイクル制御は発生していません。 1：バス・エラー／無効なサイクル制御が保留中です。 書込み時： 0：影響なし。 1：ビットをクリア。

DMA の無効なディスクリプタをチャンネル単位でクリア

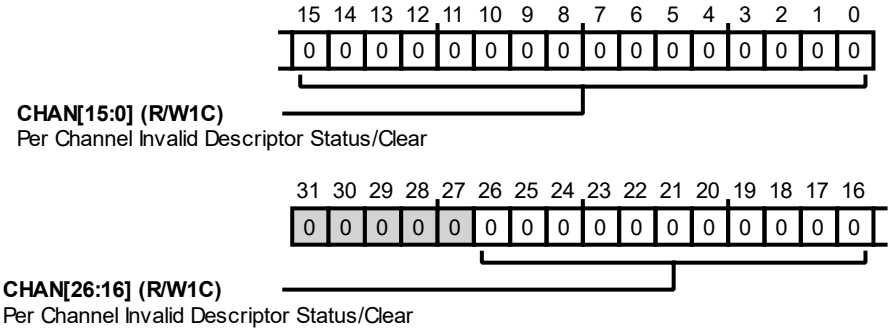


図 11-19：DMA_INVALIDDESC_CLR レジスタ図

表 11-22：DMA_INVALIDDESC_CLR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (R/W1C)	CHAN	無効なディスクリプタのチャンネル単位でのステータス／クリア。 このレジスタは、無効なディスクリプタを DMA チャンネル単位で読出しおよびクリアするのに使用します。コントローラが無効なディスクリプタ（サイクル制御が 3'b000）を読み出すと、無効なディスクリプタのステータスがチャンネル単位でセットされます。あるチャンネルで無効なサイクル制御を読み出した場合、そのチャンネルはコントローラによって自動的にディスエーブルされます。他のチャンネルは影響を受けません。ビットをクリアするには 1 を書き込みます。 読出し時： 0：無効なサイクル制御は発生していません。 1：無効なサイクル制御が保留中です。 書込み時： 0：影響なし。 1：ビットをクリア。

DMA チャンネルのプライマリ制御データベース・ポインタ

DMA_PDBPTR レジスタは、システム・メモリにおけるプライマリ・チャンネル制御ベース・ポインタを指すように設定する必要があります。DMA コントローラに割り当てなければならないシステム・メモリの量は、使用する DMA チャンネルの数、および代替チャンネル制御データ構造体を使用するかどうかによって異なります。DMA コントローラがリセット状態のときは、このレジスタを読み出すことはできません。

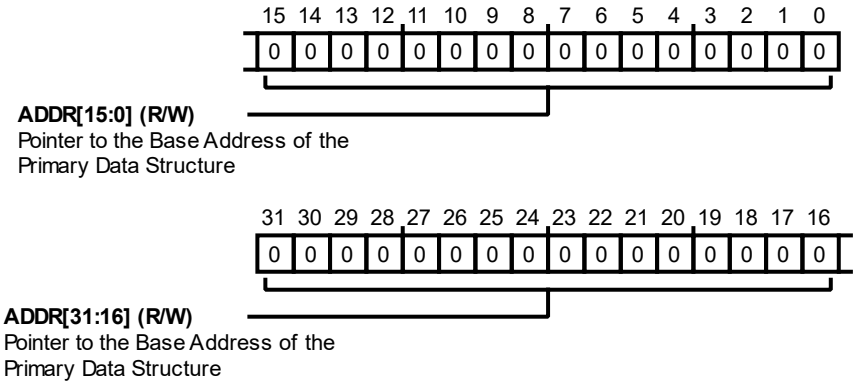


図 11-20：DMA_PDBPTR レジスタ図

表 11-23：DMA_PDBPTR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (R/W)	ADDR	プライマリ・データ構造体のベース・アドレスへのポインタ。 5 + log（2）M LSB は予約されているため、0 を書き込む必要があります。M はチャンネル数です。

DMA チャンネルの優先度をクリア

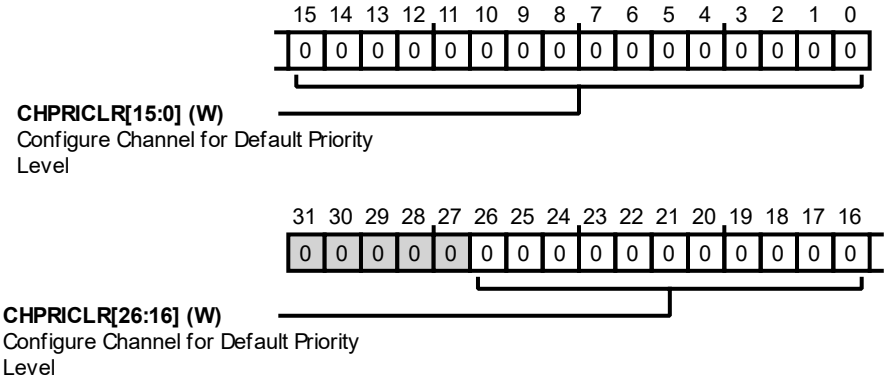


図 11-21：DMA_PRI_CLR レジスタ図

表 11-24：DMA_PRI_CLR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (RX/W)	CHPRICLR	チャンネルをデフォルト優先度に構成。 DMA_PRI_CLR 書込み専用レジスタは、デフォルト優先度を使用するように DMA チャンネルを構成します。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。適切なビットをセットすると、指定した DMA チャンネルのデフォルト優先度を選択します。 ビット 0 は DMA チャンネル 0 に対応し、ビット M-1 は DMA チャンネル M-1 に対応します。 書込み時： DMA_PRI_CLR.CHPRICLR［C］= 0、影響なし。チャンネル C を高優先度に設定するには、DMA_PRI_SET レジスタを使用します。 DMA_PRI_CLR.CHPRICLR［C］= 1、チャンネル C にデフォルト優先度を設定します。

DMA チャンネルの優先度を設定

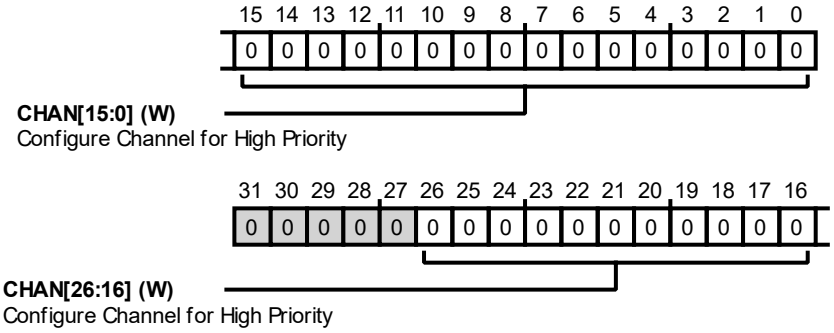


図 11-22：DMA_PRI_SET レジスタ図

表 11-25：DMA_PRI_SET レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (RX/W)	CHAN	チャンネルを高優先度に設定。 このレジスタを使用すると、高優先度を使用するように DMA チャンネルを設定できます。レジスタを読み出すと、チャンネルのプライオリティ・マスクのステータスが返されます。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。チャンネルのプライオリティ・マスク・ステータスを返すか、チャンネルの優先度を高優先度に設定します。 ビット 0 は DMA チャンネル 0 に対応し、ビット M-1 は DMA チャンネル M-1 に対応します。 読出し時： DMA_PRI_SET.CHAN［C］= 0、DMA チャンネル C はデフォルト優先度を使用しています。 DMA_PRI_SET.CHAN［C］= 1、DMA チャンネル C は高優先度を使用しています。 書込み時： DMA_PRI_SET.CHAN［C］= 0、影響なし。チャンネル C をデフォルト優先度に設定するには、DMA_PRI_CLR レジスタを使用します。 DMA_PRI_SET.CHAN［C］= 1、チャンネル C で高優先度を使用します。

DMA コントローラのリビジョン ID

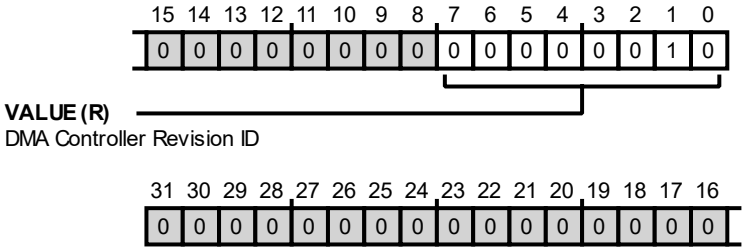


図 11-23：DMA_REVID レジスタ図

表 11-26：DMA_REVID レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
7:0 (R/NW)	VALUE	DMA コントローラのリビジョン ID。

DMA チャンネルのリクエスト・マスクをクリア

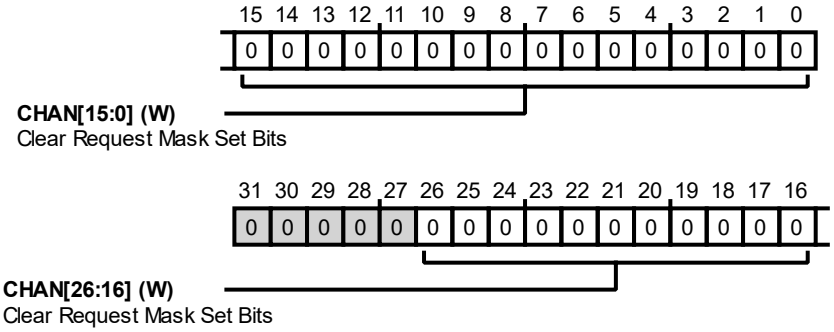


図 11-24：DMA_RMSK_CLR レジスタ図

表 11-27：DMA_RMSK_CLR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (RX/W)	CHAN	リクエスト・マスクのセット・ビットをクリア。 このレジスタは、DMA_RMSK_SET レジスタにセットされたマスクをクリアすることにより、ペリフェラルからの DMA リクエストを有効にします。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。適切なビットを設定すると、対応する DMA_RMSK_SET.CHAN ビットをクリアします。 ビット 0 は DMA チャンネル 0 に対応します。 ビット M-1 は、DMA チャンネル M-1 に対応します。 書込み時： DMA_RMSK_CLR.CHAN［C］= 0、影響なし。DMA リクエストを無効にするには、DMA_RMSK_SET レジスタを使用します。 DMA_RMSK_CLR.CHAN［C］= 1、チャンネル C に関連するペリフェラルをイネーブルして、DMA リクエストを生成させます。

DMA チャンネルのリクエスト・マスクをセット

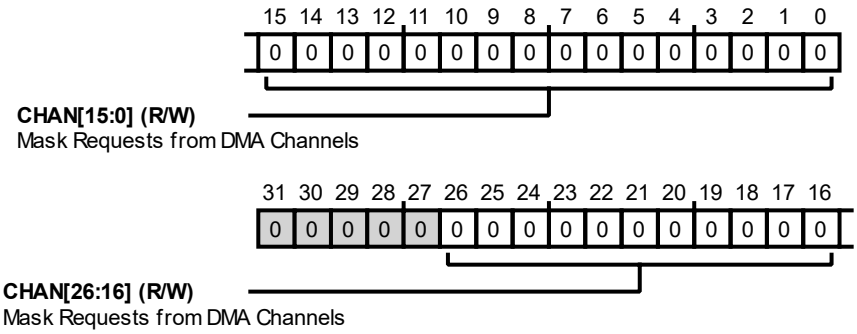


図 11-25：DMA_RMSK_SET レジスタ図

表 11-28：DMA_RMSK_SET レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (R/W)	CHAN	DMA チャンネルからのリクエストをマスク。 このレジスタは、ペリフェラルからの DMA リクエストを無効にします。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。適切なビットをセットすると、対応する DMA チャンネルからのリクエストをマスクします。 ビット 0 は DMA チャンネル 0 に対応します。 ビット M-1 は、DMA チャンネル M-1 に対応します。 読出し時： DMA_RMSK_SET.CHAN［C］= 0、チャンネル C でリクエストが無効になっています。 DMA_RMSK_SET.CHAN［C］= 1、チャンネル C でリクエストが有効になっています。 書込み時： DMA_RMSK_SET.CHAN［C］= 0、影響なし。DMA リクエストを有効にするには、DMA_RMSK_CLR レジスタを使用します。 DMA_RMSK_SET.CHAN［C］= 1、チャンネル C に関連するペリフェラルによる DMA リクエストの生成を無効にします。

DMA チャンネルのソース・アドレス・デクリメント・イネーブルをクリア

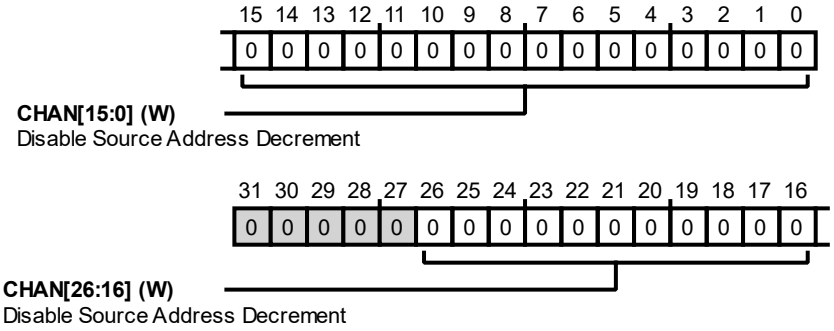


図 11-26：DMA_SRCADDR_CLR レジスタ図

表 11-29：DMA_SRCADDR_CLR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (RX/W)	CHAN	ソース・アドレスのデクリメントを無効化。 DMA_SRCADDR_CLR 書込み専用レジスタを使用すると、デフォルトのインクリメント・モードでソース・アドレスを扱うように DMA チャンネルを設定できます。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。 ビット 0 は DMA チャンネル 0 に対応します。 ビット M-1 は、DMA チャンネル M-1 に対応します。 書込み時： DMA_SRCADDR_CLR.CHAN［C］= 0、影響なし。チャンネル C でソース・アドレスのデクリメントを有効にするには、DMA_SRCADDR_SET レジスタを使用します。 DMA_SRCADDR_CLR.CHAN［C］= 1、チャンネル C でソース・アドレスのデクリメントを無効にします。

DMA チャンネルのソース・アドレス・デクリメント・イネーブルをセット

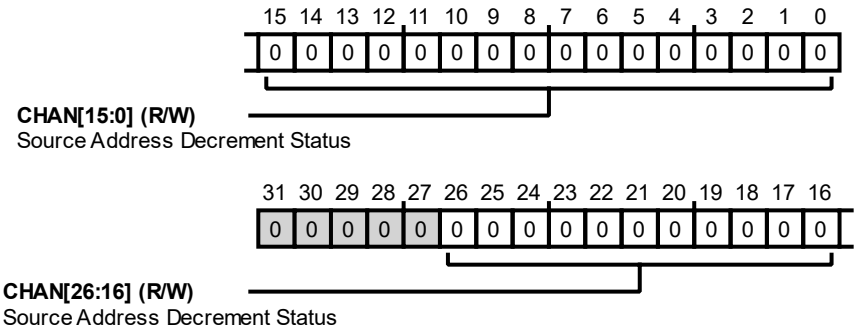


図 11-27：DMA_SRCADDR_SET レジスタ図

表 11-30：DMA_SRCADDR_SET レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (R/W)	CHAN	ソース・アドレス・デクリメントのステータス。 DMA_SRCADDR_SET レジスタは、DMA チャンネルのソース・アドレスを、各アクセス後にインクリメントではなくデクリメントするように構成するために使用します。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。ビット 0 は DMA チャンネル 0 に対応し、ビット M-1 は DMA チャンネル M-1 に対応します。 読出し時： DMA_SRCADDR_SET.CHAN［C］= 0、チャンネル C でソース・アドレスのデクリメントが無効になっています。 DMA_SRCADDR_SET.CHAN［C］= 1、チャンネル C でソース・アドレスのデクリメントが有効になっています。 書込み時： DMA_SRCADDR_SET.CHAN［C］= 0、影響なし。チャンネル C でソース・アドレスのデクリメントを無効にするには、DMA_SRCADDR_CLR レジスタを使用します。 DMA_SRCADDR_SET.CHAN［C］= 1、チャンネル C でソース・アドレスのデクリメントを有効にします。

DMA ステータス

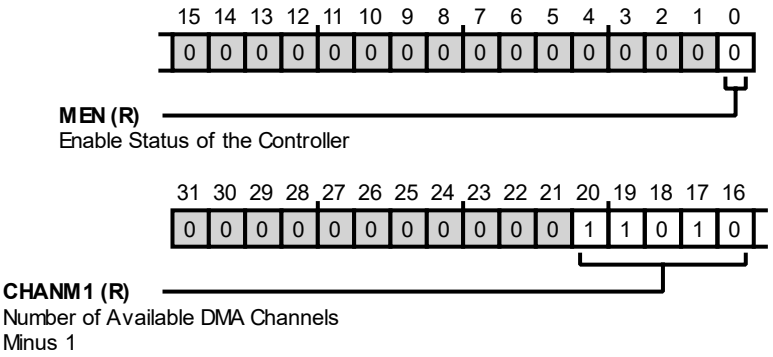


図 11-28：DMA_STAT レジスタ図

表 11-31：DMA_STAT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
20:16 (R/NW)	CHANM1	使用可能な DMA チャンネル数から 1 を引いた値。 27 チャンネルを使用できる場合、レジスタから 0x1A が読み出されます。	
0 (R/NW)	MEN	コントローラのステータスを有効化。	
		0	コントローラが無効
		1	コントローラが有効

DMA チャンネルのソフトウェア・リクエスト

DMA_SWREQ レジスタは、ソフトウェア DMA リクエストの生成を有効にします。レジスタの各ビットは、DMA コントローラ内の対応するチャンネル番号を表します。M は、DMA チャンネルの数です。

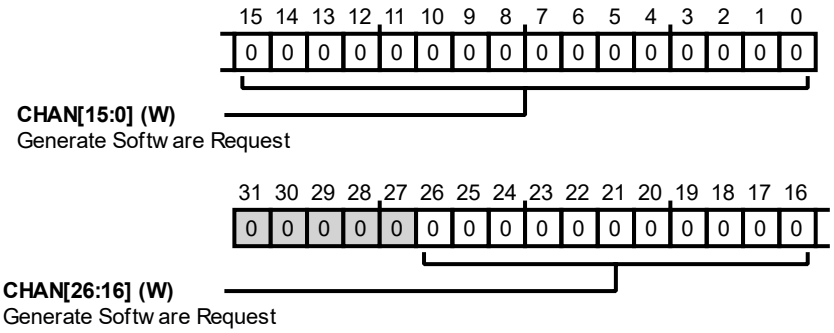


図 11-29：DMA_SWREQ レジスタ図

表 11-32：DMA_SWREQ レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
26:0 (RX/W)	CHAN	ソフトウェア・リクエストを生成。 適切なビットをセットすると、対応する DMA チャンネルでソフトウェア DMA リクエストが生成されます。 ビット 0 は DMA チャンネル 0 に対応します。 ビット M-1 は、DMA チャンネル M-1 に対応します。 書込み時： DMA_SWREQ.CHAN［C］= 0、チャンネル C で DMA リクエストを生成しません。 DMA_SWREQ.CHAN［C］= 1、チャンネル C で DMA リクエストを生成します。 これらのビットは、対応するソフトウェア・リクエストが完了後、ハードウェアによって自動的にクリアされます。

12 暗号化（CRYPTO）

暗号化ブロックは、AES 暗号、NIST ブロック動作モード、SHA-256 ハッシュ関数ジェネレータ、鍵付き HMAC アクセラレータ、鍵ラップ／アンラップ・モジュール、および保護された鍵の保存領域を包含するハードウェア・アクセラレータ・ブロックです。

高度暗号化標準（Advanced Encryption Standard：AES）は電子データを暗号化するための対称鍵アルゴリズムの仕様で、米国標準技術局（National Institute of Standards and Technology：NIST）が、従来の DES 標準に代えて 2001 年に公表したものです。AES アルゴリズムは 128 ビットの固定データ・ブロックに対して動作します。

ブロック動作モードは、対称鍵ブロック暗号アルゴリズムと共に使用する秘密保持および認証のための動作モードです。このブロックでサポートされている対称鍵暗号は AES です。

セキュア・ハッシュ・アルゴリズム（SHA）は、メッセージを 512 ビットのデータ・ブロックに分解しダイジェストを作成するハッシュ・アルゴリズムの一種です。

鍵付き HMAC（Keyed HMAC）は SHA-256 ベースのアルゴリズムで、秘密暗号鍵との組み合わせでメッセージ認証コード（Message Authentication Code：MAC）を生成します。

鍵ラップ／アンラップ・モジュールは、NIST が認定した鍵の安全な保管および移動のための鍵暗号化／復号アルゴリズムです。

保護された鍵の保存領域は暗号化モジュールだけがアクセスできる専用スペースで、暗号鍵の安全な不揮発性保存のために使用します。

ここでは、AES 標準、NIST ブロック暗号の動作モード、SHA-256 アルゴリズム、SHA-256 ベースの鍵付き HMAC、および鍵ラップ／アンラップ・アルゴリズムについて十分な知識を有していることを前提に、説明を進めます。詳細については以下の文献を参照してください。

表 12-1：NIST 発行文献リスト

AES standard	Federal Information Processing Standard (FIPS) 197
NIST Block Modes of Operation	Special Publication-800-38A, B & C
SHA	FIPS 180-4
HMAC	FIPS 198-1 (Keyed HMAC)
Key Wrap	SP800-38F (Methods for Key Wrapping)
CCM*	Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)

暗号化機能

ADuCM4050 MCU は以下の機能を備えています。

- 32 ビット APB スレーブ
- 128 ビット・バッファ × 2：
  - 入力バッファ
  - 出力バッファ
- HMAC モードおよび SHA モード用の入力データを保存する 512 ビット・バッファ × 1
- DMA チャンネル × 2：
  - 入力バッファ
  - 出力バッファ
- 対応可能な AES 暗号鍵の長さ：
  - 128 ビット
  - 256 ビット
- AES 暗号鍵ソース
  - MMR を通じてプログラム可能
  - 読出し不可（ソフトウェアでゼロとして読出し）
- 対応モード：
  - ECB モードの AES 暗号化／復号
  - CBC モードの AES 暗号化／復号
  - CTR モードの AES 暗号化／復号
  - AES 暗号 MAC 生成モード
  - CCM モードの AES 暗号化／復号
  - CCM*モードの AES 暗号化／復号
  - SHA-256 モード
  - 鍵のラップ／アンラップにより保護された鍵の保存領域
  - HMAC シグネチャの生成

暗号化機能の説明

ここでは、ADuCM4050 MCU が使用する暗号化ブロックの機能について説明します。

暗号化ブロック図

この図は、ADuCM4050 MCU が使用する暗号化ブロックを示しています。

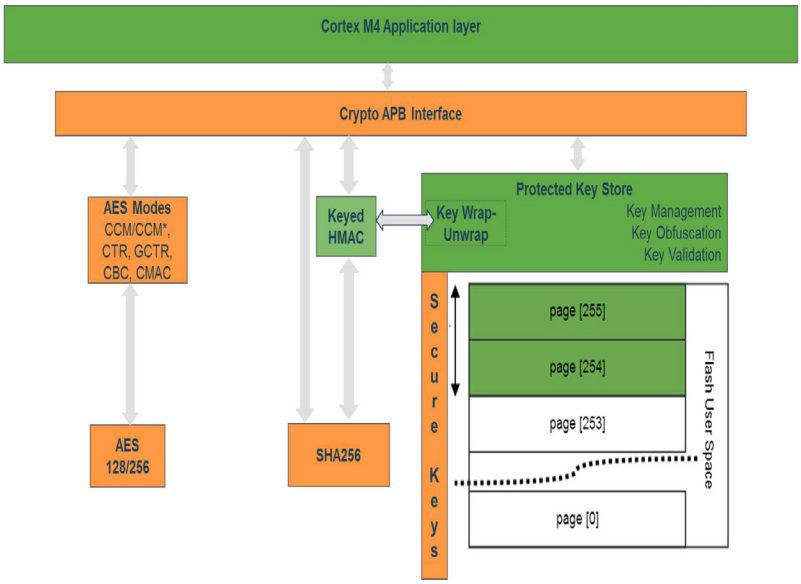


図 12-1：暗号化ブロック

暗号化ブロックは 32 ビット APB DMA 対応のペリフェラルで、データ I/O 動作用に 2 つのバッファを備えています。これらのバッファは 32 ビット幅で、4 回のデータ・アクセスで 128 ビットの読込みと読出しを行います。入力データと出力データのバイト・スワッピングは、`CRYPT_CFG.AES_BYTESWAP` 設定ビットおよび `CRYPT_CFG.SHA_BYTESWAP` 設定ビットを使って行います。更に、`CRYPT_CFG.KEY_BYTESWAP` ビットを使用すると、MMR への鍵設定時にソフトウェアで鍵のバイト・スワップを行うことができます。

イネーブルされている場合、このブロックは入力バッファからデータを取得し、出力バッファを通じて処理済みデータを出力します。これらのデータ処理には DMA を使用できますが、これは消費電力を低減する上で非常に重要です。ブロックをイネーブルしてデータ転送を開始する前に、設定ビット、鍵、その他の関連レジスタが希望の値に設定されていることを確認してください。

暗号化ブロックの動作モード

以下のブロック動作モードがサポートされています。

- CTR モード：カウンタ・モード
- CBC モード：暗号ブロック・チェーン・モード
- MAC モード：メッセージ認証コード・モード
- CCM モード：暗号ブロック・チェーン・メッセージ認証コード・モード
- ECB モード：電子コード・ブック・モード
- CCM*モード：WPAN 標準用修正 CCM モード

暗号化ブロックは以下の動作モードもサポートしています。

- PrKStor モード：鍵ラップ／アンラップ用ハードウェア・アクセラレータを備えた、保護された鍵の保存領域
- SHA-256：SHA-256 アルゴリズムを使用するハッシュ生成
- 鍵付き HMAC：SHA-256 ベースの認証モード

CRYPT_CFG レジスタのモード有効化ビット・フィールドは、希望の動作モードが有効になるように設定します。該当のビット・フィールドを 1 に設定すると、そのフィールドに対応するブロック・モードが有効になります。

同時に複数のブロック動作モードを有効にすることはできません。

CRYPT_CFG.ECBEN、CRYPT_CFG.CTREN、CRYPT_CFG.CBCEN、CRYPT_CFG.CCMEN、CRYPT_CFG.CMACEN、CRYPT_CFG.SHA256EN、CRYPT_CFG.PRKSTOREN、CRYPT_CFG.HMACEN の各ビット・フィールドは、それぞれ異なる動作モードをイネーブルします。

複数のブロック・モードを同時にイネーブルすると、ブロックは予測不能な動作をします。エラーや警告がフラグされることはありません。CCM*モードは CRYPT_CFG.CCMEN ビットを使って有効にします。

暗号化／復号

CRYPT_CFG.ENCR ビットは、AES 暗号を使用する様々なブロック・モードにおける暗号化／復号動作を選択するために使用します。復号を有効にすると、最終的な鍵のスケジュール値が計算され、オリジナルの AESKEY 値に上書きする形で AESKEY レジスタ・セットに書き込まれます。したがって、新しい暗号化処理を行うには、AESKEY の値を設定し直す必要があります。

暗号化データの転送

入力バッファと出力バッファは 128 ビットのデータを保持します。32 ビット APB バスのペリフェラルであるこのブロックには、32 ビット・データ・バスでアクセスします。したがって出力／入力バッファを空あるいはフルにするには、4 回の読出し／書込み動作が必要です。

SHA256 モードでは、結果は CRYPT_SHAH0.SHAHASH0～CRYPT_SHAH7.SHAHASH7 のレジスタで更新され、出力バッファには書き込まれません。ブロック動作モードの結果は、出力バッファからストリーム出力されます。

暗号化データ・レート

ブロック・モードと、AES 暗号用に選択した鍵の長さに応じて、データを処理する速度は変化します。

各種ブロック動作モードに必要な計算時間を以下に示します。これらの数値には、バッファの読出しと書込みのための時間は含まれていません。読出し／書込み時間は DMA またはコアのデータ・レートによって制御されます。下に示すレートは、鍵長 128/256 バイトに対する値です。

- ECB モード：40/56 クロック・サイクル
- CBC モード：40/56 クロック・サイクル

- CTR モード：40/56 クロック・サイクル
- MAC モード：40/56 クロック・サイクル
- CCM モード：80/102 クロック・サイクル

入力バッファへのデータ入力レートが 1 クロック・サイクルあたり 1 ワードだとすると、SHA-256 の計算には 64 クロック・サイクルが必要です。暗号化入力データ・バッファが APB プロトコルをサポートしているので、入力データ・レートは 2 クロック・サイクルあたり 1 ワード未満です。

データ・パイプライン

ブロック動作モードでは、入力バッファに 128 ビットのデータ（仮にブロック B1 とします）が書き込まれると、直ちにそのデータが内部で計算用にコピーされて、次の 128 ビットのデータ（ブロック B2）を入力バッファに書き込むことができます。

ブロック B2 が書き込まれる時点までにブロック B1 の計算が完了しない場合、入力バッファはフルになったままで、別のデータを書き込むことはできません。この場合は、該当する割込みレジスタとステータス・レジスタが更新されます。ブロック B1 の処理が完了すると、その結果はすぐに出力バッファに移されて B2 の計算が開始されます。B2 の処理が完了するまでに B1 が出力バッファから読み出されない場合、そのブロックは保留状態になります。この場合は、該当する割込みレジスタとステータス・レジスタがセットされます。

DMA 機能

このブロックは、DMA を使用してバッファへデータを移動したり、バッファからデータを移動したりすることができます。DMA を使用すると、入力バッファと出力バッファの両方で、1 回の DMA 要求につき 4 ワードを転送できます。これらのバッファに関する DMA 要求は、CRYPT_CFG.INDMAEN ビットと CRYPT_CFG.OUTDMAEN ビットをセットすることによって有効にできます。ブロックは、入力バッファがフルになるまでその DMA 要求ラインをハイにします。同様に、出力バッファに関する DMA 要求は、出力バッファが空になるまでハイに維持されます。

これらの設定ビットは、データ・トランザクションの途中で変更しないようにしなければなりません。変更すると誤った内容が出力される結果となりますが、エラーや警告は生成されません。

コア転送

DMA を使用しない場合は、ブロックに転送されるワード数とブロックから読み出されるワード数をソフトウェアで追跡する必要があります。ステータスとエラー状態を示すビット・フィールドは、CRYPT_STAT レジスタ内にあります。様々な状態に対する割込みを、CRYPT_INTEN レジスタでイネーブルすることができます。

DMA ベースの転送は性能の面でも消費電力の面でも効率的なので、その使用が推奨されます。

データ・フォーマット

ここでは、データと鍵を暗号化レジスタへマッピングする方法と、出力バッファから送られるデータのフォーマットについて説明します。

以下に示す鍵、入力データ、および出力データを考えます。鍵、平文、および暗号文には、NIST 規格に定めるデータを使用します。

```
Key [0：15] ：2b7e151628aed2a6abf7158809cf4f3c
Plaintext [0：15] ：6bc1bee22e409f96e93d7e117393172a
```

```
Cipher  Text［0：15］：3ad77bb40d7a3660a89ecaf32466ef97
```

この 128 ビットの鍵を、以下のように AESKEY レジスタへマップします。

```
AESKEYi =  Key［i*4+3：i*4］;
```

上の例では次のようになります。

```
 AESKEY0［31：0］= 0x16157e2b;
 AESKEY1［31：0］= 0xa6d2ae28;
 AESKEY2［31：0］= 0x8815f7ab;
 AESKEY3［31：0］= 0x3c4fcf09;
```

平文は、以下に示すように DMA／コアを使用し、APB 書込みを介して入力バッファへ送る必要があります。

```
for（i=0; i<4; i++）
 {
Input_buffer = plain_text［i*4+3：i*4］;
}
```

例に示す平文は、以下のように入力バッファ書込みへマップできます。

```
write（Input_buffer, 0xe2bec16b）;
write（Input_buffer, 0x969f402e）;
write（Input_buffer, 0x117e3de9）;
write（Input_buffer, 0x2a179373）;
```

暗号文・ブロックは、4 回の連続読出しで出力バッファから読み出すことができます。

```
for（i=0; i<4; i++）
 {
Cipher_text［i*4+3：i*4］= read（output_buffer）;
}
```

バイト・スワップ設定

暗号化バッファと鍵レジスタへ書き込むデータのバイト・スワップ用に、3 つの設定ビットが用意されています。

バイト・スワップ操作は以下のように定義されます。

```
Input  String：0xB3B2B1B0
Output String：0xB0B1B2B3
```

AES_BYTESWAP

このビットをセットすると、ブロック動作モード用の入力バッファに書き込まれるデータと、出力データから読み出されるデータがバイト・スワップされます。

SHA_BYTESWAP

このビットをセットすると、SHA 動作と HMAC 動作用の入力バッファに書き込まれるデータがバイト・スワップされます。

KEY_BYTESWAP

CRYPT_AESKEY0～CRYPT_AESKEY7、CRYPT_KUW0～CRYPT_KUW15、CRYPT_KEYValStr1、および CRYPT_KEYValStr2 レジスタに書き込まれるデータのバイト・スワップに関して、更に高い柔軟性が実現されます。

割込み

INRDY

この割込みは入力バッファがフルでないことを示します。この割込みを有効にすると、入力バッファがフルにならない限り割込みがトリガされます。この割込みは、コアがバッファへ書込みを行ったときに使われるので、入力バッファへの入力は ISR から行ってください。

AES ベースの動作については、すべてこの割込みを使用して暗号化アクセラレータへデータを送る必要があります。SHA ベースの動作や HMAC アクセラレータには、この割込みを使用しないでください。

OUTRDY

この割込みは、出力バッファがデータを保持していること、およびデータの読出しを待っていることを示します。これは、出力バッファが空でない限りセットされたままになります。この割込みは、コアがバッファを使用できるようになったときに使われるので、ISR 内の 128 ビットすべてを読み出してください。

この割込みは AES ブロック暗号モードだけに使用し、SHA には使用しません。

SHADONE

この割込みは、SHA の計算が終了したことを示します。新しいデータを入力バッファに書き込むことができます。あるいは、ハッシュ結果を CRYPT_SHAH0 レジスタから CRYPT_SHAH7 レジスタへ読み出すことができます。

INOVR

この割込みは、入力バッファにオーバーフロー・イベントが発生したことを示します。原因として以下が挙げられます。

- 出力バッファが読み出されていない。この場合、ブロックは動作を停止します。
- 入力データ・レートが暗号化ブロックの処理可能レートより大きい。

DMA またはコアが推奨に従ってプログラムされている場合、この割込みを生成することはできません。

入力がオーバーフローすると、暗号化動作の信頼性が失われます。

HMACDONE

この割込みは、HMAC 動作の最終ステップが完了したことを示します。HMAC 動作の結果は、SHARES レジスタ・セットから読み出すことができます。

HMACMSGRDY

このステータス・ビットは、HMAC ブロックが KEY IPAD のハッシュを完了して、データ入力の準備ができたことを示します。対応するステータス・ビットは、割込みルーチン内でクリアする必要があります。HMAC コントローラが現在のデータ・ブロックのハッシュを完了すると、この割込みが再度トリガされてイネーブルになります。最後のメ

ッセージ・データ・ブロックについては、ソフトウェアが CRYPT_SHA_LAST_WORD.O_LAST_WORD ビットと CRYPT_SHA_LAST_WORD.O_BITS_VALID ビットを使用して、これを示す必要があります。

PRKSTOR_CMD_DONE

このビットは、保護された鍵の保存領域が操作されたことを示します。鍵のラップとアンラップ、フラッシュ内の安全な領域への鍵の保存を含む、複数の操作を行うことができます。

暗号化エラー状態

出力バッファが空のときに読出しを行おうとすると、アンダーフロー・エラーとなります。出力バッファが空の場合、ブロックは出力バッファ・チャンネル上で DMA 要求を生成しません。

計算の任意の段階で暗号化ブロックがディスエーブルされている場合は、データ出力の信頼性が失われます。ステート・マシンはリセットされます。

入力バッファがフルのときに書込みを行おうとすると、オーバーフロー・エラーとなります。入力バッファがフルの場合、ブロックは入力バッファ・チャンネル上で DMA 要求を生成しません。

入力バッファと出力バッファは、CRYPT_CFG レジスタ内の対応フラッシュ・ビットに 1 を書き込むことによってフラッシュできます。

割込みは、CRYPT_INTEN レジスタ内の対応ビットに 1 を書き込むことによってイネーブルにできます。

割込みのクリアは、CRYPT_STAT レジスタ内のそれぞれのビットに 1 を書き込むことによって可能です。

CRYPT_STAT.PRKSTORBUSY ビットがセットされている場合は、保護された鍵の保存領域モード（CRYPT_CFG.PRKSTOREN ビットまたは CRYPTO_CFG.BLKEN ビット）をソフトウェアで無効にすることはできません。

暗号化ステータス・ビット

INWORDS/OUTWORDS

これらのフィールドには、入出力バッファにその時点で格納されているワード数に関する情報が保持されています。この情報は、入出力バッファが空でない状況下、あるいはフルでない状況下で、書き込む必要のあるワード数または読み出す必要のあるワード数を知るために使用できます。

SHADONE

このフィールドは、現在提供されているデータに関するハッシュ計算が完了したことを示します。

SHABUSY

このフィールドは、ハッシュ計算が進行中であることを示します。この信号がローになった場合は、ハッシュ計算が完了してその結果を読み出せる状態になったことを示しています。

INOVR
このフィールドは、入力バッファ上でオーバーフロー・イベントが発生したことを示します。このステータス・ビットはスティッキーで、ビット・フィールドに 1 を書き込むことによってクリアできます。

HMACDONE
このビットは、現在のメッセージ入力に関する HMAC 処理が完了して、SHARES レジスタ・セットから結果を読み出せる状態であることを示します。

HMACBUSY
このビットは、HMAC アクセラレータが現在ビジー状態にあることを示します。このビットがセットされた場合の SHARES レジスタ・セット内の結果は中間的なもので、ゼロとして読み出されます。

HMACMSGRDY
このビットは、HMAC ブロックがユーザからメッセージ入力を取得する準備ができたこと、つまり、key xor ipad を使用する SHA256 計算の最初の段階が完了したことを示します。更に、SHA256 が現在のメッセージ入力ブロックを処理している間も、このビットはローになります。

PRKSTOR_CMD_DONE
このビットがセットされた場合は、現在の PrKStor コマンドが完了して、PrKStor に新たなコマンドを送出できるようになったことを示します。

PRKSTOR_CMD_FAIL
このビットが `CRYPT_STAT.PRKSTOR_CMD_DONE` ビットと共にセットされている場合は、PrKStor コマンドの実行に失敗したことを示します。

失敗の理由は以下のとおりです。

- アンラップ・コマンド：アンラップ後に取得した検証文字列が、本書に指定されている 3 つの検証文字列のどれにも一致しなかった場合は失敗します。
- RETRIEVE_KEY：このコマンドは、フラッシュ・コントローラから応答がない場合や ECC エラーによって鍵が壊れた場合、失敗する可能性があります。この場合は、読み出した鍵が KUW レジスタ内でゼロになります。
- 消去および破棄コマンド：これらのコマンドは、コマンドが送出されたときにフラッシュがビジー状態であるなどの理由でフラッシュから応答がない場合は失敗します。

PRKSTOR_RET_STATUS
これらのステータス・ビットはフラッシュ読出しによる ECC ステータスを示します。これはフラッシュ内の ECC ステータスと同様です。

CMD_ISSUED
これは、最後に送出されたコマンドを、モードが無効になるかブロックがディスエーブルされるまで保持します。あるいは新しいコマンドが送出されるまで保持します。

PRKSTOR_BUSY
このステータス・ビットは、コマンドが実行中で PrKStor をディスエーブルできない状態にあることを示します。現在実行中の PrKStor コマンドは、PrKStor の完全性を維持するために、休止状態への移行前に完了させる必要があります。

暗号鍵

AES 暗号に使用する鍵は、CRYPT_AESKEY0 から CRYPT_AESKEY7 までの鍵レジスタ内に設定する必要があります。

鍵の長さ

AES 暗号は、128 ビットと 256 ビットの 2 種類の鍵長で使用することができます。鍵長は、CRYPT_CFG.KEYLEN ビットを設定して選択します。レジスタ CRYPT_AESKEY0 から CRYPT_AESKEY7 には、AES 暗号に使用する鍵が格納されます。

CRYPT_CFG.KEYLEN は、0x2 に設定された場合は暗号処理に 256 ビットの鍵が使われ、0x0 にリセットされた場合は 128 ビットの鍵が使われることを示します。

鍵の設定

鍵レジスタは任意の順番で設定できます。選択した鍵長が 128 ビットのときは、長さ 256 ビットの鍵に使用するレジスタを設定することはできません。

AES 暗号処理に使用する鍵（K）を AESKEY レジスタ・セットから作成する方法を以下に示します。

CRYPT_AESKEY0 = K［31：0］

CRYPT_AESKEY1 = K［63：32］

CRYPT_AESKEY2 = K［95：64］

CRYPT_AESKEY3 = K［127：96］

CRYPT_AESKEY4 = K［159：128］（鍵長が 256 の場合のみ使用）

CRYPT_AESKEY5 = K［191：160］（鍵長が 256 の場合のみ使用）

CRYPT_AESKEY6 = K［223：192］（鍵長が 256 の場合のみ使用）

CRYPT_AESKEY7 = K［255：224］（鍵長が 256 の場合のみ使用）

注：鍵レジスタの一部だけに書込みを行うことはできません。新しい鍵（長さが同じか異なるかは問わず）を使用する場合は、所定の AESKEY レジスタの前の値または新しい値が同じであったとしても、すべての鍵レジスタを更新する必要があります。

鍵ラップ／アンラップ・レジスタ

CRYPT_KUW0～CRYPT_KUW15 レジスタ・セットは、アンラップする予定のラップされた鍵、もしくはラップする予定のアンラップされた鍵を格納します。

ラップ／アンラップ処理の結果も同じレジスタ・セットに格納されます。更に、CRYPT_KUWVALSTR1 レジスタと CRYPT_KUWVALSTR2 レジスタには、鍵をラップするために使う検証文字列によって生成される追加内容が格納されます。鍵をラップする間、ラップ処理に必要な検証文字列はこれらのレジスタ内に格納されます。

検証文字列レジスタは、アンラップ処理後にのみ読み出すことができます。セキュリティが確保されたソフトウェアでは、この検証文字列を使用し、ソフトウェア・ベースで鍵の検証方法を実装することができます。

鍵レジスタの属性

このフィールドは、入力バッファ上でオーバーフロー・イベントが発生したことを示します。このステータス・ビットはスティッキーで、ビット・フィールドに 1 を書き込むことによってクリアできます。

CRYPT_AESKEY0～CRYPT_AESKEY7 レジスタ・セット、CRYPT_KUW0～CRYPT_KUW15 レジスタ・セット、CRYPT_KUWVALSTR1 レジスタ、および CRYPT_KUWVALSTR2 レジスタを鍵レジスタと呼びます。

暗号鍵を安全に使用するために、以下の機能が実装されています。

- ソフトウェアでは鍵レジスタを読み出すことはできません。
- ラップ／アンラップ処理に使用する検証文字列レジスタは、アンラップ・コマンドの実行後から、次の PrKStor コマンドが送出されるまで、またはブロックがディスエーブルされるまでの特別な読出しウィンドウ内でのみ読み出すことができます。読出し値は、鍵の使用を検証するためのメタデータを所定の鍵に対応させるために使用できます。
- 検証文字列レジスタを除く鍵レジスタは、部分的な書込みでリセットされます。ハードウェア・アクセラレータと共に新しい鍵を使用する場合、ソフトウェアは完全な鍵を設定する必要があります。
- 鍵長を変更すると、鍵レジスタと鍵検証レジスタがデフォルト値にリセットされます。

暗号化パワー・セーブ・モード

CRYPT_CFG.BLKEN レジスタのパワー・セーブ・モードを有効にすると自動クロック・ゲート機能が有効になり、ブロック内のクロッキング動作が最小限に抑えられます。このモードをオンにしても性能には影響しないので、常時オンにしておくことを推奨します。

モード固有の役割を持つレジスタ

DATALEN

CRYPT_DATALEN レジスタは、ペイロード・データの長さを指定するために使われます。この情報は、CCM 動作モードと MAC 動作モードで使われます。

- CCM モード：最終的に得られるアライン済みペイロード内の 128 ビット・データ・ブロックの数を設定します。CRYPT_DATALEN［15：0］は、この目的のために使用します。CCM は、16 バイトの倍数以外のデータ長をサポートしています。ブロック数は、次の整数に切り上げる必要があります。例えば、データのブロック数が 4.5 の場合は、データ長を 5 に設定します。
- CMAC モード：ペイロード・データの合計ビット数が 128 の整数倍となるようにパディングを行い、アライン済みペイロード・データ内の 128 ビット・データ・ブロックの数を設定します。CRYPT_DATALEN［19：0］は、この目的のために使用します。例えば、データ長が 56 バイトの場合は、データ長を 4 に設定する必要があります。

PREFIXLEN

CRYPT_PREFIXLEN レジスタは、認証だけが終了した対応データの長さを指定するために使われます。この情報は、CCM 動作モードで使われます。

CCM モード：対応データの合計ビット数が 128 の整数倍となるように 0 でパディングを行い、最終的に得られるアライン済みペイロード・データ内の 128 ビット・ブロックの数を設定します。
CRYPT_PREFIXLEN.VALUE ビットは、この目的のために使用します。

NONCEx

CRYPT_NONCE0 から CRYPT_NONCE3 までのレジスタは、CTR、CBC、および CCM 動作モードのノンスを設定するために使用します。

ノンスは、次のように構成されます。

Nonce［127：0］= {CRYPT_NONCE3［31：0］, CRYPT_NONCE2［31：0］, CRYPT_NONCE1［31：0］, CRYPT_NONCE0［31：0］}

異なるブロック動作モードには、異なる長さのノンスが使われます。

- CTR モード：このモードは長さ 108 ビットのノンスを使用します。このモードでは、暗号化処理は以下のフォーマットで行われます。

  Ciphertext_Block = Ciph（ {Nonce［107：0］, CRYPT_CNTRINIT［19：0］）xor Payload_Block
- CBC モード：このモードは長さ 128 ビットのノンスを使用します。ノンスは初期化ベクトルとして使われます。
  Initialization_Vector = Nonce［127：0］
- CCM モード：このモードは長さ 112 ビットのノンスを使用します。ノンスはカウンタ・モードの計算に使われます。データは以下のフォーマットで構成されます。

  Ciphertext_Block = Ciph（ {Nonce［111：0］, Counter［15：0］} ）xor Payload_Block

SHARESx

SHA ハッシュ処理の結果は、以下のフォーマットで SHARES レジスタ・セットから読み出されます。

HASH［255：0］= {SHARES7［31：0］, SHARES6［31：0］, SHARES5［31：0］, SHARES4［31：0］, SHARES3［31：0］, SHARES2［31：0］, SHARES1［31：0］, SHARES0［31：0］}

CNTRINIT

この値はカウンタ生成機能を初期化します。このブロック内に実装されたカウンタ生成機能はインクリメント機能です。つまり、この機能は、新しいデータ・ブロックごとに 1 つずつカウント値を加算していきます。カウンタの初期値は、CRYPT_CNTRINIT レジスタ内に設定できます。

CTR モードと CCM モードのカウンタ幅は以下のとおりです。

- CTR モード：20 ビット・カウンタ。カウンタがゼロからスタートした場合は、オーバーフローするまでに 16GB のデータが処理されます。また、カウンタ初期化用に 20 ビットのレジスタが用意されています。
- CCM モード：16 ビット・カウンタ。カウンタがゼロからスタートした場合は、オーバーフローするまでに 1GB のデータが処理されます。また、カウンタ初期化用に 16 ビットのレジスタが用意されています。

  カウンタのビット幅が固定されているということは、同じカウンタ値を繰り返さずに暗号化できるデータの最大長に限界があることを示唆しています。つまり、CTR モードのペイロード・データ最大長は 16MB、CCM モードの秘密データ最大長は 1MB です。

注：カウンタが最大値に達すると、次のデータ入力時にゼロにロールオーバーします。

SHAINIT

これは自動クリアのレジスタ・フィールドです。制御信号は、新しい計算開始時に SHA 計算レジスタを初期化して値をリセットするために使われます。

SHA_LAST_WORD

SHA-256 モジュールは、長さカウンタとパディング・マスクを含めて更新されます。ソフトウェアでメッセージの最後にパディングとビット長を追加する必要はありません。

SHA 制御レジスタには以下のフィールドがあります。

- last_word
- bits_valid［4：0］

ソフトウェアは、入力データを 1 ワード（32 ビット）として暗号化アクセラレータ入力バッファにロードします。入力データは、32 ビットの倍数であってもなくても構いません。ユーザは last_word ビットをセットし、bits_valid ビットを最終ワード（0-31）内の有効なビット数に設定して、その最終ワードを暗号化アクセラレータへ書き込む必要があります。このワードの LSB は、SHA 標準に従って無視されます（パディングに置き換えられる）。データ長が 32 ビットの倍数の場合は、最終処理を完了させるためにダミー・ワードを書き込む必要があります（すべてのビットがパディングに置き換えられる）。

注：メッセージの最終ワードでは、last_word をアサートして、bits_valid に last_word 内の有効ビット数を反映させる必要があります。すべてのビットが有効な場合は、最初にワードを書き込んでから、有効なビットを持たないダミー・ワード（最終ワード）を書き込む必要があります。無効なビットは無視されます。

保護された鍵の保存領域（PrKStor）

このプラットフォームは、フラッシュ・メモリの保護された鍵の保存領域（PrKStor）という形で、暗号鍵を保存するための安全な不揮発性領域を提供します。PrKStor は 2 つの 2KB ページで構成され、1 ページあたり最大で 51 個（合計で 102 個）の鍵を保存できる、保護された保存領域を提供します。それぞれの鍵レコードはサイズ 320 ビットの鍵に対応しており、ページとインデックスの値によって識別されます。ソフトウェアは、128 ビットまたは 256 ビット

のラップされた鍵とアンラップされた鍵をこの 320 ビットのスペースに保存できます。鍵使用ポリシーに関するすべてのメタデータは、ソフトウェアによって維持する必要があります。このモジュールは、鍵関連の様々な処理を行うためのコマンド・インターフェースを備えており、鍵ラップ／アンラップ・モジュールと共にこのモジュールを使用することで、安全な鍵更新ポリシーを設計することができます。

PrKStor モジュールは、暗号化（Crypto）モジュールのスレーブとして実装されています。PrKStor とのすべての連携は、暗号化モジュールのレジスタ・アドレス・スペース内でのレジスタ・アクセスを通じて行われます。PrKStor が使用するために予約されたフラッシュ・メモリを、ユーザが設定することはできません。また、フラッシュ・コントローラを通じてアクセスすることもできません。

鍵のラップ／アンラップ

この機能は、鍵の暗号化と復号を行うために PrKStor に追加されたものです。このラップされた鍵は、検証プレフィックスと共に KUW レジスタ・セットに保存されます。すなわち、CRYPT_KUW0〜CRYPT_KUW15 レジスタ・セット、CRYPT_KUWVALSTR1 レジスタ、および CRYPT_KUWVALSTR2 レジスタです。これらのレジスタは、ソフトウェアによって選択的に読み出すことができます。この読出しウィンドウの範囲は、アンラップ処理の後から次の PrKStor コマンドが送出されるまでです。

鍵の検証

ソフトウェアの意図せぬ変更を防ぐために安全なブートローダが組み込まれている場合は、ソフトウェアに安全な鍵検証ポリシーを実装することができます。この安全検証を有効にするために、アンラップ・コマンド送出後から他の PrKStor コマンドが送出されるまで、あるいはブロック／モードが無効になるまでのウィンドウ内で、CRYPT_KUWVALSTR1 レジスタと CRYPT_KUWVALSTR2 レジスタを読み込むことができます。ソフトウェアはこのレジスタを読み出して、鍵検証ポリシーに関係するメタデータを、その鍵に関係する他のメタデータと共に保存することができます。ソフトウェアは、その検証鍵が現在の暗号化処理条件を満たさない場合、AESKEY および KUW レジスタ・セットをワイプします。

保護と完全性

- 鍵のラップ：フラッシュ内に保存される鍵は任意のランダム鍵で暗号化できますが、これらの鍵は、ラップまたはアンラップ処理の前に AESKEY レジスタ・セットに設定する必要があります。
- 読出し保護：PrKStor フラッシュ・メモリをターゲットとするときは、暗号化データ・パスだけがイネーブルされます（フラッシュ・コントローラ・データ・パスは FF にプルされます）。
- 書込み／消去保護：PrKStor フラッシュ・メモリをターゲットとするときに使用できるコマンドは、暗号化コマンド・インターフェースからのコマンドだけです（他のフラッシュ処理はすべて保留されます）。一括消去は行わないでください（ユーザ・スペース内の任意のページ・セットを書込み保護する）。
- ECC を使用するデータ完全性：1 ビット・エラー訂正と複数ビット・エラー検出は、下位層のフラッシュ・コントローラによって行われます。

動作

PrKStor は、物理的にはフラッシュ・メモリ内のユーザ・スペースにある最上位の 2 ページに置かれています。これら 2 ページへは、通常のフラッシュ・メモリへのアクセス方法ではアクセスできず、代わりに暗号化モジュール専用のバス／プロトコルを使用してアクセスします。

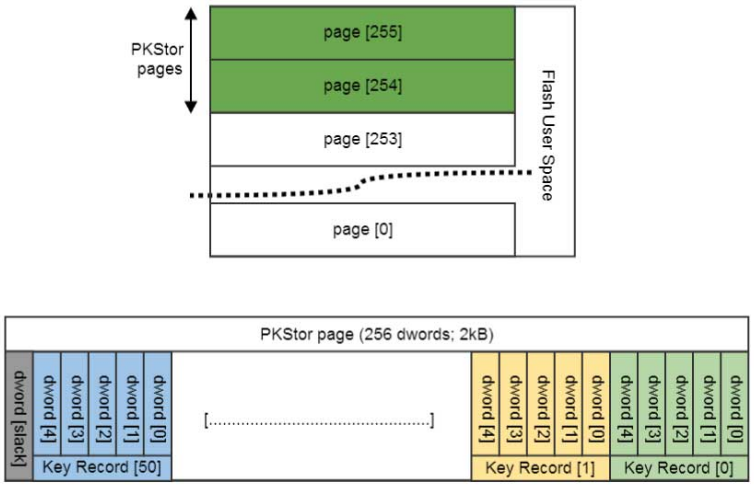


図 12-2：PrKStor の構成

PrKStor の 2 つのページには、それぞれ 51 個ずつの暗号鍵を保存できます。それぞれの PrKStor には 1DWORD のスラック・スペースが存在します。これらのスペースは予約されています。暗号化モジュールは、どちらかの PrKStor ページの任意の DWORD へインデックスすることができます。安全のため、暗号化モジュールは、より抽象的なインターフェースをユーザに提供しています。ユーザ・コードは、ページ番号とインデックスによって個々の鍵をアトミックに参照することができます（例えば、ページ 0 の最下位の鍵は［0：0］、ページ 1 の最上位の鍵は［1：50］）。

メタデータは PrKStor 領域内には保存されません。保存されている鍵の位置と特定用途に関わる有効性を追跡できるようにするには、そのユーザ固有のメタデータを維持する必要があります。

保護された鍵の保存領域の設定

設定ビット

設定ビット・フィールドは、`CRYPT_PRKSTORCFG` レジスタと `CRYPT_CFG` レジスタ内にあります。

- デバイス鍵：デバイス鍵は RTL の 256 ビット固定鍵です。これは、鍵を PrKStor フラッシュ保存領域に保存する前に、その鍵をローカルでラップ／アンラップするための鍵です。デバイス鍵はソフトウェアにとっては未知のものですが、PrKStor コマンド LD_DEVICE_KEY を使って AESKEY レジスタ・セットへロードすることができます。ソフトウェアは、AESKEY レジスタ・セットへ別の鍵をロードすることによって、その鍵をラップまたはアンラップ処理に使用することができます。
- PRKSTOREN：PrKStor コマンドを送出する前に、このビットをセットする必要があります。
- KEY_INDEX［6：0］：PrKStor コマンドによる処理の対象となった鍵のインデックスを示します。MSB はページ ID を示し、残りのビットは該当ページ内の鍵の位置を示します。
- CMD［3：0］：PrKStor 内で実行するコマンドを示します。コマンドが実行されると、これらのビットは自動的にクリアされます。最後に送出されたコマンドは、`CRYPT_STAT.CMD_ISSUED` ビットを読み出すことによって確認できます。

- KUW0～KUW7、KUWValStr1、および KUWValStr2 レジスタ：書込み専用レジスタ。これらのレジスタは、ラップまたはアンラップ操作を実行する必要があるユーザからの鍵を格納します。例えば、256 + 64 ビットの鍵をアンラップする場合は、KUW0～KUW7 レジスタ・セットと、KUWValStr1 および KUWValStr2 レジスタにその鍵をロードする必要があります。
- KUWKEYLEN［2：0］：アンラップされた鍵の長さを 64 ビット・ブロック単位で示します。

  KUWKEYLEN = 0x3 の場合、鍵長は 512 ビットです。512 ビットの鍵は HMAC 計算用にのみサポートされており、PrKStor フラッシュ保存領域には保存できません。

  鍵長が 256 ビットの場合は、KUWKEYLEN を 0x2 に設定する必要があります。

  128 ビット鍵の場合、KUWKEYLEN は 0x1 に設定します。

ステータス・ビット

ステータス・ビットは CRYPT_STAT レジスタ内にあります。

- PRKSTOR_BUSY：このビットがセットされた場合は、PrKStor がコマンド実行中でビジー状態にあり、新しいコマンドを受け取ることができないことを示します。
- PRKSTOR_CMD_DONE：このビットがセットされた場合は、PrKStor が、最後に送出されたコマンドの実行を終了したことを示します。
- PRKSTOR_CMD_FAIL：このビットがセットされた場合は、最後に送出されたコマンドの実行に失敗したことを示します。これらのコマンドのいくつかは、フラッシュの動作に依存します。失敗が繰り返される場合は通常のフラッシュ動作を試み、（問題が存在する場合は）フラッシュ・コントローラ内部の問題をデバッグしてください。
- PRKSTOR_RET_STATUS［1：0］：ステータスが 0x2 の場合は、RETRIVE_KEY 動作に関して 1 ビット ECC エラーが少なくとも 1 つ確認され、訂正されたことを示しています。ステータスが 0x1 の場合は、RETRIEVE_KEY 動作のリードバック時に複数ビットの ECC エラーが少なくとも 1 つ確認されたことを示しています。得られる鍵は無効なものとなる可能性があります。
- CMD_ISSUED［3：0］：このフィールドは、PrKStor に送出された最後のコマンドを記録します。CMD フィールドはコマンドが完了すると自動的にクリアされるので、これは、送出された最後のコマンドを読み出すソフトウェアのオプションとして提供されます。

割込み許可

PRKSTRCMDONEEN：このビットがセットされた場合は、PRKSTOR コマンド完了時にコアに割込みがかけられます。

保護された鍵の保存領域用のコマンド

PrKStor モジュールは以下のコマンドをサポートしています。

- WRAP：CMD = 0x1。このコマンドは、CRYPT_KUW0～CRYPT_KUW7 レジスタ・セットに保存された鍵に対し、ラップ処理を行う必要があることを示します。
- UNWRAP：CMD = 0x2。このコマンドは、CRYPT_KUW0～CRYPT_KUW15 レジスタ・セット、CRYPT_KUWVALSTR1 レジスタ、および CRYPT_KUWVALSTR2 レジスタに保存されたラップ鍵に対し、アンラップ処理を行う必要があることを示します。アンラップ後は、他の PrKStor コマンドが送出されるまで、あるいはモード／ブロックが無効化されるまで、ソフトウェアが検証文字列を読み出すことができます。

検証文字列のデフォルト値は、NIST 標準の検証文字列 0xA6A6A6A6A6A6A6A6 です。

- RST_DECRYPTED_KEY：CMD = 0x3。CRYPT_KUW0～CRYPT_KUW15 レジスタ・セットの内容は、このコマンドを送出することによってクリアできます。復号／アンラップされた鍵を使用した後は、このビットをセットすることを推奨します。
- USE_DECRYPTED_KEY：CMD = 0x4。このコマンドを送出することによって、復号／アンラップした鍵をその後の暗号化処理に使用することができます。このコマンドは、KUW レジスタの内容を AESKEY レジスタにロードします。このコマンドを実行すると、CRYPT_CFG.AESKEYLEN が、CRYPT_CFG.KUWKEYLEN に保存されている値に更新されます。この鍵は、一部だけを変更することができない構造になっています。AES がサポートしている鍵長は 128 ビットと 256 ビットなので、このコマンドの実行前にソフトウェアが 512 ビットの鍵を設定することはできません。
- LD_DEV_KEY：CMD = 0x5。このコマンドは、CRYPT_AESKEY0～CRYPT_AESKEY7 レジスタ・セットにデバイス鍵をロードします。デバイス鍵は 256 ビット鍵なので、CRYPT_CFG.AESKEYLEN ビットは 256 ビット・オプションに設定する必要があります。デバイス鍵は、CRYPT_KUW0～CRYPT_KUW15 レジスタ・セット内のあらゆる鍵のラップ／アンラップに使用できます。
- RETRIEVE_KEY：CMD = 0x8。CRYPT_PRKSTORCFG.KEY_INDEX ビットによって指定される位置はフラッシュから読み出され、CRYPT_KUW0～CRYPT_KUW15 レジスタ・セット、CRYPT_KUWVALSTR1 レジスタ、および CRYPT_KUWVALSTR2 レジスタ内に置かれます。
- STOR_KEY：CMD = 0x9。このコマンドが送出されると、CRYPT_KUW0～CRYPT_KUW15 レジスタ・セット、CRYPT_KUWVALSTR1 レジスタ、および CRYPT_KUWVALSTR2 レジスタの内容が、CRYPT_PRKSTORCFG.KEY_INDEX ビットによって指定されるフラッシュ内位置に保存されます。
- DESTROY_KEY：CMD = 0xA。このコマンドが送出されると、CRYPT_PRKSTORCFG.KEY_INDEX フィールドによって指定されるインデックス位置にある鍵が 0 で上書きされます。
- ERASE_PAGE：CMD = 0xB。このコマンドが送出されると、CRYPT_PRKSTORCFG.KEY_INDEX［6］フィールドで指定されるページが消去されます。

プログラミング・フロー

PrKStor はコア・モードで使用することを推奨します。

- 有効な KUW 鍵のプログラミング：KUW 鍵の場合、許容されるのは鍵全体の書込みだけです。いずれかの KUW 鍵レジスタを上書きすると、すべての KUW 鍵レジスタが 0 にリセットされるので、KUW 鍵の一部だけを変更することはできません。KUW 鍵を設定する前に、CRYPT_CFG.KUWKEYLEN フィールドを適切な値に設定する必要があります。また、関連するすべてのレジスタを設定する必要があります。

  CRYPT_CFG.KUWKEYLEN = 3'b001：KUW 鍵の長さは 512 で、CRYPT_KUW0 から CRYPT_KUW15 までのすべてのレジスタを設定する必要があります。

  CRYPT_CFG.KUWKEYLEN = 3'b010：KUW 鍵の長さは 256 で、CRYPT_KUW0 から CRYPT_KUW7 までのすべてのレジスタを設定する必要があります。

  CRYPT_CFG.KUWKEYLEN = 3'b100：KUW 鍵の長さは 128 で、CRYPT_KUW0 から CRYPT_KUW3 までのすべてのレジスタを設定する必要があります。

MMR を介して設定された KUW 鍵に対してラップまたはアンラップ・コマンドを実行するには、KUW 検証文字列を設定する必要があります。検証文字列の一部だけを変更することはできず、64 ビットすべてを一度に設定する必要があります。

USE_DECRYPTED_KEY コマンドの実行時は、CRYPT_CFG.KUWKEYLEN を正しくプログラムする必要があります。このコマンドを実行すると、CRYPT_CFG.AESKEYLEN ビット・フィールドが、CRYPT_CFG.KUWKEYLEN に保存された値に更新されます。

- PrKStor コマンドを実行するには、CRYPT_CFG.PRKSTOREN ビットと CRYPT_CFG.BLKEN ビットを設定する必要があります。設定レジスタが最初に設定され、PrKStor へコマンドを送出する前にブロックがイネーブルされます。
- PrKStor がイネーブルされると、一連のコマンド実行が可能になります。
- USE_DECRYPTED_KEY コマンドと LD_DEV_KEY コマンドを使用するときは、設計によって CRYPT_CFG.AESKEYLEN ビット・フィールドが内部的に変更されます。復号された鍵の長さは、USE_DECRYPTED_KEY コマンドの場合のみ 128 ビットまたは 256 ビットになります。デバイス鍵は固定の 256 ビット鍵です。LD_DEV_KEY が実行されると、CRYPT_CFG.AESKEYLEN フィールドは 256 ビット値に変更されます。
- PrKStor コマンドは、休止状態になる前にすべて完了させる必要があります。コマンドが途中で停止した場合、PrKStor 保存領域の完全性は確保されません。CRYPT_STAT.PRKSTOR_BUSY ビットがセットされた場合は、コマンドが実行中であることを示します。
- コマンドのステータスは、CRYPT_STAT.PRKSTOR_CMD_DONE ビットを使って読み出すことができます。ソフトウェアは、CRYPT_STAT.PRKSTOR_CMD_FAIL ビットを使ってコマンドの合否ステータスも確認する必要があります。このビットが CRYPT_STAT.PRKSTOR_CMD_DONE ビットと共にセットされている場合は、PrKStor コマンドの実行が失敗したことを示しています。
- RETRIEVE_KEY 動作時に ECC エラー（訂正可能か否かを問わず）が発生した場合で、KEY_CORRUPT_INT_EN ビットがセットされている場合は、コアに割込みがかけられます。正しい鍵で改めて鍵位置を書き込み、再確認することを推奨します。
- 何回か再試行してもエラーが再発してコマンドが正しく実行されない場合は、フラッシュのデバッグを行う必要があります。

コマンド失敗ステータス

CRYPT_STAT.PRKSTOR_CMD_FAIL ビットが CRYPT_STAT.PRKSTOR_CMD_DONE ビットと共にセットされた場合は、最後に送出されたコマンドの実行が失敗したことを示しています。

PrKStor がディスエーブル

PrKStor ページへのタンパリングのために PrKStor がディスエーブルされると、RETRIEVE_KEY、STOR_KEY、DESTROY_KEY、および ERASE_PAGE コマンドを実行できなくなります。

ゼロ以外の ECC ステータス

RETRIEVE_KEY コマンド送出時にフラッシュから読み出されたデータのいずれかの ECC ステータスがゼロでない場合、その RETRIEVE コマンドの実行は失敗です。

フラッシュ書込み失敗

フラッシュが書込みコマンドを正しく実行できない場合、STOR_KEY コマンドの実行は失敗です。

メタデータの維持

ソフトウェアは、どの鍵がどのインデックスに保存され、それらが何に使われるのかを追跡する必要があります。このメタデータは、鍵破棄（Destroy Key）コマンド実行時と不良鍵（Bad Key）ステータス（修正不能 ECC エラー）時に更新する必要があります。すべてのコマンドの開始と終了をメタデータに反映することを推奨します。明確に終了しなかったコマンドがある場合は、それがメタデータに反映され、再使用時には対応する PrKStor インデックスが期待通りに動作しない可能性があることを意味します。

鍵の検証

何らかの PrKStor 処理を行った後は、鍵を使って既知の平文と暗号文のペアを暗号化／復号することにより、その鍵を検証することができます。

HMAC

より安全なデータ認証方法として、HMAC 生成モードが追加されています。セキュリティ強化のために、512 ビット鍵のラップ／アンラップ・モードも追加されています。これは、鍵暗号化鍵を使用することで、HMAC 秘密鍵 K の暗号化／復号を容易にします。

このモジュールは、任意の長さのユーザ鍵 K をサポートしています。サポートされている鍵の最大ラップ長さが 512 なので、ラップされた HMAC 鍵 K を使用するときは、ソフトウェアで鍵 K0 を処理して提供する必要があります。

K：作成者と意図した受領者の間で共有される秘密鍵。

K0：B バイト鍵を作成するために必要な前処理を行った鍵 K。

HMAC アルゴリズム

入力仕様

- HMAC 鍵：前処理済み 512 ビット鍵。

  key_len < 512 の場合、HMAC 鍵 = `[Key, {512 - key_len} 'b0]`

  key_len > 512 の場合、HMAC 鍵 = `[SHA256（KEY）, {512 – 256} 'b0} ]`

- 平文のメッセージ

出力仕様

HMAC 認証文字列。SHA256 の結果と同じ。

```
o_key_pad = [0x5c * blocksize] ⊕ key // Blocksize = 256

i_key_pad = [0x36 * blocksize] ⊕ key // ⊕ represents XOR operation

HMAC_Output = SHA256（o_key_pad ‖ SHA256 （i_key_pad ‖ message））// ‖ represents concatenation
```

プログラミング・モデル

設定ビット

- HMACEN：暗号化アクセラレータの HMAC モードを有効にします。
- KEY_BYTESWAP：鍵レジスタにロードされたデータのバイト・スワッピングを有効にします。

  HMAC 鍵は 512 ビット長であるものとします。これは、NIST の仕様書に従って前処理する必要があります。ゼロのパディングはハードウェアにより自動的に行われます。
- SHA_BYTESWAP：メッセージ入力を HMAC アクセラレータへ送る前にバイト・スワップする必要がある場合は、SHA_BYTESWAP をそのために再利用することができます。

ステータス・ビット

- HMACMSGRDY：暗号化ブロックが、ハッシュ処理する必要のあるテキストを取り込む準備ができたことを示します。
- HMACDONE：現在の HMAC 動作が完了したことを示します。
- HMACBUSY：HMAC 動作が進行中であることを示します。

動作

HMAC は、設定レジスタの CRYPT_CFG.HMACEN ビットをセットすることによって有効にします。設計が前処理済みの HMAC 鍵をサポートしているので、HMAC 鍵の長さが 512 ビットより長い場合は、ソフトウェアで鍵のハッシュ処理を行って 256 ビットの出力を生成し、256 個の 0 を追加して HMAC 鍵 K0 を作成する必要があります。

コア・モード

1. KUW レジスタに有効な鍵 K0 をロードします。
2. HMAC を有効にします。
3. ソフトウェアは、メッセージ・テキストを設計に入力する前に、CRYPT_STAT.HMACMSGRDY ビットまたは CRYPT_INTEN.HMACMSGRDYEN（割込みモード）を待つ必要があります。
4. ソフトウェアは CRYPT_STAT.HMACMSGRDY ビットをクリアして、1 ブロックのデータを入力する必要があります（HMAC アクセラレータに 512 ビットを入力）。
5. HMAC アクセラレータへのメッセージ・ブロックの入力が完了したら、ソフトウェアは、次の CRYPT_INTEN.HMACMSGRDYEN を待って次のデータ・ブロックを HMAC アクセラレータへ入力する必要があります。
6. 最後のデータ・ブロックでは、SHA に対して行ったものと同じ手順を行う必要があります。最終ワードを除くすべてのデータが設計に入力されたら、CRYPT_SHA_LAST_WORD をセットして、CRYPT_SHA_LAST_WORD.O_BITS_VALID を使って最終ワード内の有効ビット数に関する情報を入力します。
7. HMAC アクセラレータにデータを入力したら、ソフトウェアは、HMAC の動作が完了したことを示す HMACDONE 割込みまで待機する必要があります。

DMA モード

1. KUW レジスタに鍵 K0 をロードします。
2. 必要データの転送用に DMA を設定します（最終ワードを除く）。
3. HMAC と DMA を有効にします。
4. 暗号化ブロックをイネーブルします。
5. CRYPT_INTEN.HMACMSGRDYEN をディスエーブルします。DMA は、コアからの割込みなしで必要なすべてのデータを転送します。
6. SHA へ最終ワードを入力するには、コア動作モードで説明した手順と同じ手順に従う必要があります。
7. ソフトウェアは、ポーリングまたは割込みのどちらかを通じて CRYPT_STAT.HMACDONE ビットが設定されるのを待つ必要があります。

注：テキスト・データは、通常の SHA 処理と同じ方法で入力できます。テキスト・データが入力されたら、ソフトウェアは、CRYPT_STAT.HMACDONE ビットをチェックする必要があります。このビットは、ハッシュ処理が完了して、SHARES レジスタ・セットから最終結果を読み出せるようになったことを示します。メッセージ長が 512 ビットの倍数でない場合は、ハードウェアが最終データ・ブロック用に入力されたメッセージの処理を開始できるように、SHA 計算時に行った CRYPT_SHA_LAST_WORD および CRYPT_SHA_LAST_WORD.O_BITS_VALID による処理と同様の処理を行う必要があります。

CCM/CCM*モード

図には以下のカラー・コードを使用します。

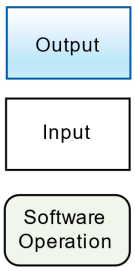


図 12-3：カラー・コード

CCM モードの NIST 標準は、CCM 処理を以下のように記述しています。

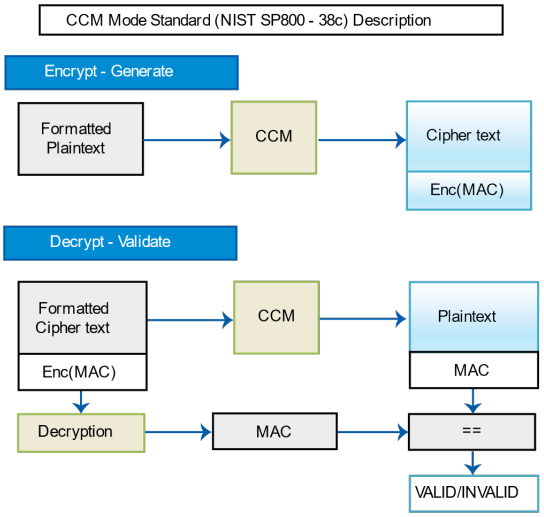


図 12-4：CCM モードの標準記述

CCM モードのハードウェア実装を下の図に示します。

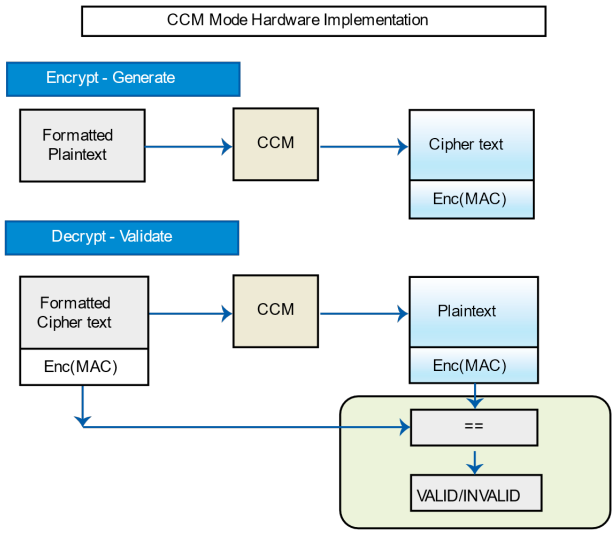


図 12-5：CCM モードのハードウェア実装

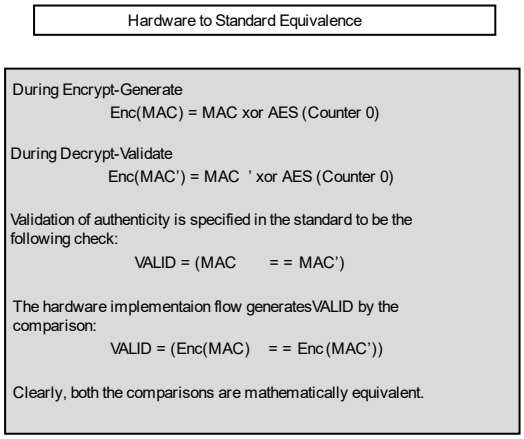


図 12-6：ハードウェアと標準の等価性

復号-検証段階で有効（VALID）信号を生成するために採用した実装フローに違いがある理由を以下に示します。

- CMAC モードは MAC の結果を提供し、受信データを使ってその結果をチェックしませんが、CCM モードではこのチェックを行うのが妥当です。
- このアプローチの方が高速です。復号が必要なデータをシステムが受信する場合、システムは暗号化された MAC を受信します。ハードウェア実装では、受信した MAC を復号する必要はありません。
- MAC の長さは選択可能です。ハードウェアが内部で比較を行う必要がある場合は、MAC 出力に関する情報を設定する必要があります。これによって新たな制御パラメータが追加されると共に、可変長のコンパレータを備えることでハードウェアがより複雑なものとなります。
- 比較動作は単純なもので、必要な処理はごくわずかです。ハードウェア・アクセラレータは必要ありません。
- MAC が CCM 処理の出力として必要とされることはありません。必要なのは有効（VALID）または無効（INVALID）信号です。

ブロック動作モードのプログラミング・フロー

ブロック動作モードには、以下の処理を行うソフトウェアが必要です。

- 暗号化ブロックの設定
- 入力データの作成
- データの読出し

暗号化ブロックの設定

暗号鍵の設定：あらゆる処理を行う前に、鍵レジスタ内に完全な鍵を設定する必要があります。鍵の一部だけを設定することはできません。

暗号化ブロックは以下の順番で設定する必要があります。

1. 鍵長、暗号化／復号の別、モードなどのパラメータを設定します。ブロックはイネーブルしないでください。

2. 鍵レジスタ内に鍵を設定します。所定の鍵長に関係する鍵レジスタだけを設定するようにしてください。
3. 設定レジスタの読出し、変更、書込みを行うことによってブロックをイネーブルします。

暗号化または復号

CRYPT_CFG.ENCR フィールド

データ転送

1. CRYPT_CFG.ENDIAN のエンディアン
2. CRYPT_CFG.INDMAEN/CRYPT_CFG.OUTDMAEN フィールド

ブロック動作モード

暗号化ブロックは以下の動作モードをサポートしています。

- ECB モードの AES 暗号化／復号
- CBC モードの AES 暗号化／復号
- CTR モードの AES 暗号化／復号
- AES 暗号 CMAC 生成モード
- CCM モードの AES 暗号化／復号
- CCM*モードの AES 暗号化／復号

注：同時に複数のモードを有効にすることはできません。

モード固有パラメータ

CTR モード

- CRYPT_CNTRINIT
- CRYPT_NONCE0～CRYPT_NONCE3

CCM/CCM*モード

- CRYPT_DATALEN
- CRYPT_PREFIXLEN
- CRYPT_CNTRINIT
- CRYPT_NONCE0～CRYPT_NONCE3

CBC モード

CRYPT_NONCE0～CRYPT_NONCE3 レジスタの初期化ベクトル。

CMAC モード

CRYPT_DATALEN：この情報をブロックが使用し、CMAC の結果をいつ出力バッファに送ればよいかを決定します。

暗号化ブロックのイネーブル

CRYPT_CFG.BLKEN フィールド。

このビットは、ブロックの設定が完了した場合のみイネーブルされるようにしてください。

ペイロードと認証済みデータのフォーマット

どの動作モードでも（CMAC を除く）、ペイロードや認証済みデータの長さを 128 ビットの倍数にするために、十分な個数の 0 でパディングを行う必要があります。CMAC モードの場合は、最後のメッセージ・ブロックを別にパディングし XOR する必要があります。

モード固有のデータ・フォーマット

- CBC モード：CRYPT_NONCE0～CRYPT_NONCE3 レジスタを初期化ベクトルによって初期化します。
- CMAC モード：ブロックは、得られた 128 ビット MAC を選択されたフォーマット（Big_Endian または Little_Endian）で出力バッファに出力します。ソフトウェアは、必要なビット数が使われていることを確認する必要があります。

  サブキーの生成：最終メッセージ・データ・ブロック（DataBlock）の生成には、K1 と K2 の値が使われます。ソフトウェアは K1 と K2 の生成を処理して上位ビットを'b10....j（j = パディングで最終的なメッセージ・データ・ブロックを 128 ビット長にするための 0 の数）でパディングし、更に XOR 処理を行って最終的なメッセージ・データ・ブロックを生成します。
- CCM/CCM*モード：CCM/CCM*モードのセクションを参照してください。

SHA 単独動作のプログラミング・フロー

コア・モード

設定

1. SHA を設定します。
2. CRYPT_CFG.SHA256EN ビットをイネーブルします。
3. データの最初のブロックを設定します。
4. CRYPT_STAT.SHADONE 割込みビットがセットされるのを待ちます。
5. 512 ビットの連続したデータ・ブロックを入力バッファに設定します。
6. データ・ブロック内の最終ワードに対しては、CRYPT_SHA_LAST_WORD.O_LAST_WORD ビットをセットして、そのワード内の有効ビット数を設定します。例えば 12 ビットが有効な場合は、CRYPT_SHA_LAST_WORD.O_BITS_VALID ビットに 12 をロードします。

すべてのビットが有効な場合は、最後の書込みをダミー書込みとする必要があります。

7. 新しい SHA 計算の場合は、CRYPT_CFG.SHAINIT ビットを使って SHA を初期化します。

別の SHA 処理を開始する前に、現在進行中の SHA 処理を完了させてください。

DMA モード

設定
以下のパラメータを設定する必要があります。

1. 入力バッファの DMA（CRYPT_CFG.INDMAEN フィールド）。
2. 必要な SHA モードを有効化（CRYPT_CFG.SHA256EN フィールド）。
3. 暗号化ブロックをイネーブル（CRYPT_CFG.BLKEN フィールド）。
4. SHA 処理を初期化（CRYPT_CFG.SHAINIT フィールド）。

   新しいデータ・ブロックを計算するためにフレッシュ・スタートとする必要がある場合は、CRYPT_CFG レジスタ内にある自動クリア・ビットを使用してデータ・ブロックを初期化する必要があります。

データの作成
SHA アクセラレータは、データをブロック・サイズの倍数とするためのソフトウェア・パディングを必要としません。

データの読出し
データ処理が完了すると、SHAH レジスタ・セットからメッセージ・ダイジェストを読み出すことができます。

特定ブロックのメッセージ・ダイジェストが完了したら、CRYPT_CFG.SHAINIT ビットに 1 を書き込むことによって、内部 SHAH レジスタを再初期化します。

休止モードでの保持

休止モードでは、CRYPT_CFG.BLKEN を除く CRYPT_CFG レジスタの内容が保持されます。AESKEY レジスタ・セットも保持されます。KUW レジスタ・セットを含む他のすべてのレジスタ・フィールドはリセットされます。

トランザクションの途中でデバイスが休止モードになった場合、そのトランザクションを再開することはできません。CRYPT_CFG および AESKEY レジスタ・セットを除くすべての内部レジスタがリセットされます。

PrKStor がビジー状態の場合、デバイスを休止モードにすることはできません。デバイスを休止状態にするには、その前に PrKStor コマンドを完了させる必要があります。

ADuCM4050 CRYPT レジスタの説明

暗号化ブロック（CRYPT）のレジスタ・マップには以下のレジスタが含まれています。

表 12-2：ADuCM4050 CRYPT レジスター一覧

レジスタ名	説明
CRYPT_AESKEY0	AES 鍵ビット［31：0］
CRYPT_AESKEY1	AES 鍵ビット［63：32］
CRYPT_AESKEY2	AES 鍵ビット［95：64］
CRYPT_AESKEY3	AES 鍵ビット［127：96］
CRYPT_AESKEY4	AES 鍵ビット［159：128］
CRYPT_AESKEY5	AES 鍵ビット［191：160］
CRYPT_AESKEY6	AES 鍵ビット［223：192］
CRYPT_AESKEY7	AES 鍵ビット［255：224］
CRYPT_CCM_NUM_VALID_BYTES	NUM_VALID_BYTES
CRYPT_CFG	設定レジスタ
CRYPT_CNTRINIT	カウンタ初期化ベクトル
CRYPT_DATALEN	ペイロード・データ長
CRYPT_INBUF	入力バッファ
CRYPT_INTEN	割込みイネーブル・レジスタ
CRYPT_KUW0	鍵ラップ／アンラップ・レジスタ 0
CRYPT_KUW1	鍵ラップ／アンラップ・レジスタ 1
CRYPT_KUW10	鍵ラップ／アンラップ・レジスタ 10
CRYPT_KUW11	鍵ラップ／アンラップ・レジスタ 11
CRYPT_KUW12	鍵ラップ／アンラップ・レジスタ 12
CRYPT_KUW13	鍵ラップ／アンラップ・レジスタ 13
CRYPT_KUW14	鍵ラップ／アンラップ・レジスタ 14
CRYPT_KUW15	鍵ラップ／アンラップ・レジスタ 15
CRYPT_KUW2	鍵ラップ／アンラップ・レジスタ 2
CRYPT_KUW3	鍵ラップ／アンラップ・レジスタ 3
CRYPT_KUW4	鍵ラップ／アンラップ・レジスタ 4
CRYPT_KUW5	鍵ラップ／アンラップ・レジスタ 5
CRYPT_KUW6	鍵ラップ／アンラップ・レジスタ 6
CRYPT_KUW7	鍵ラップ／アンラップ・レジスタ 7
CRYPT_KUW8	鍵ラップ／アンラップ・レジスタ 8
CRYPT_KUW9	鍵ラップ／アンラップ・レジスタ 9
CRYPT_KUWVALSTR1	鍵ラップ／アンラップ検証文字列［63：32］

表 12-2：ADuCM4050 CRYPT レジスター一覧（続き）

レジスタ名	説明
CRYPT_KUWVALSTR2	鍵ラップ／アンラップ検証文字列［31：0］
CRYPT_NONCE0	ノンス・ビット［31：0］
CRYPT_NONCE1	ノンス・ビット［63：32］
CRYPT_NONCE2	ノンス・ビット［95：64］
CRYPT_NONCE3	ノンス・ビット［127：96］
CRYPT_OUTBUF	出力バッファ
CRYPT_PREFIXLEN	認証データ長
CRYPT_PRKSTORCFG	PRKSTOR 構成
CRYPT_SHAH0	SHA ビット［31：0］
CRYPT_SHAH1	SHA ビット［63：32］
CRYPT_SHAH2	SHA ビット［95：64］
CRYPT_SHAH3	SHA ビット［127：96］
CRYPT_SHAH4	SHA ビット［159：128］
CRYPT_SHAH5	SHA ビット［191：160］
CRYPT_SHAH6	SHA ビット［223：192］
CRYPT_SHAH7	SHA ビット［255：224］
CRYPT_SHA_LAST_WORD	SHA 最終ワードおよび有効ビットの情報
CRYPT_STAT	ステータス・レジスタ

AES 鍵ビット［31：0］

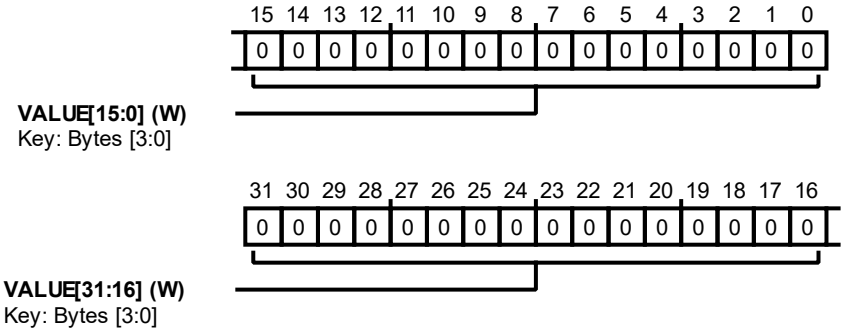


図 12-7：CRYPT_AESKEY0 レジスタ図

表 12-3：CRYPT_AESKEY0 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	VALUE	鍵：バイト［3：0］。

AES 鍵ビット［63：32］

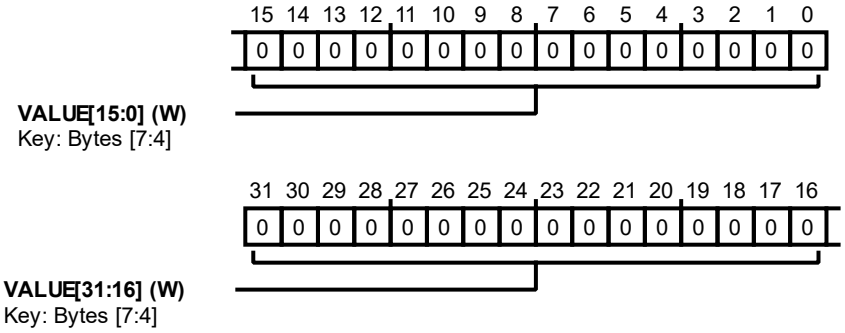


図 12-8：CRYPT_AESKEY1 レジスタ図

表 12-4：CRYPT_AESKEY1 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	VALUE	鍵：バイト［7：4］。

AES 鍵ビット［95：64］

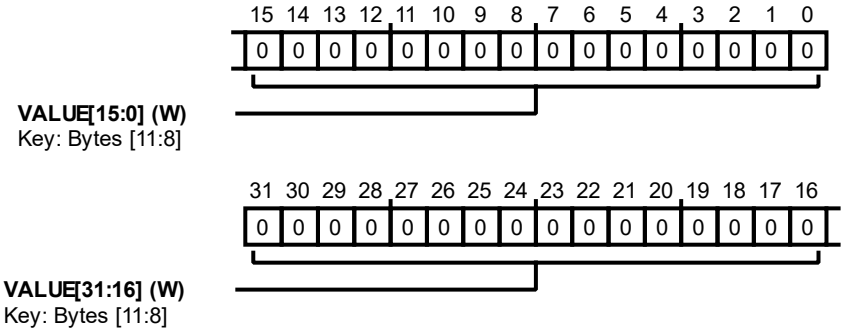


図 12-9：CRYPT_AESKEY2 レジスタ図

表 12-5：CRYPT_AESKEY2 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	VALUE	鍵：バイト［11：8］。

AES 鍵ビット［127：96］

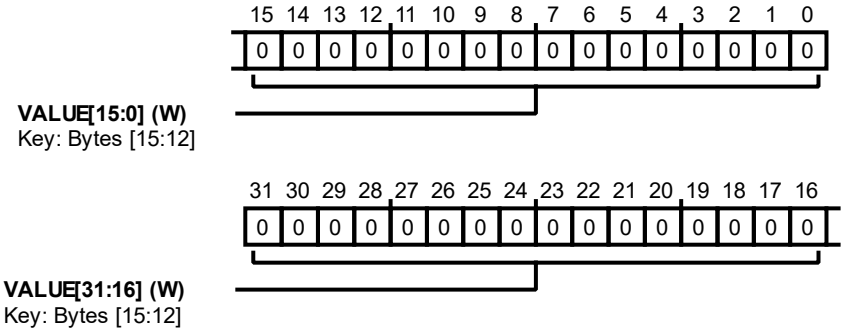


図 12-10：CRYPT_AESKEY3 レジスタ図

表 12-6：CRYPT_AESKEY3 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	VALUE	鍵：バイト［15：12］。

AES 鍵ビット［159：128］

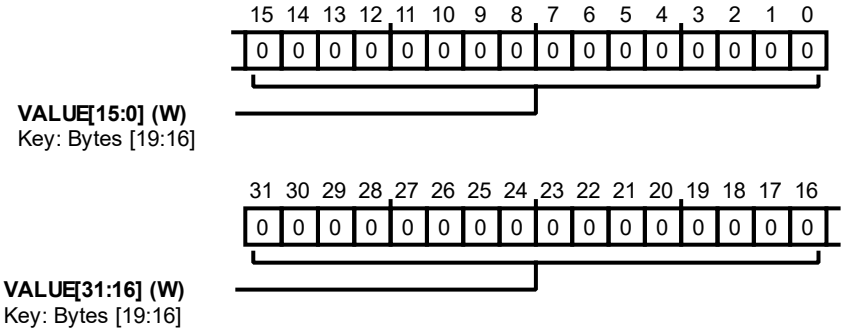


図 12-11：CRYPT_AESKEY4 レジスタ図

表 12-7：CRYPT_AESKEY4 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	VALUE	鍵：バイト［19：16］。

AES 鍵ビット［191：160］

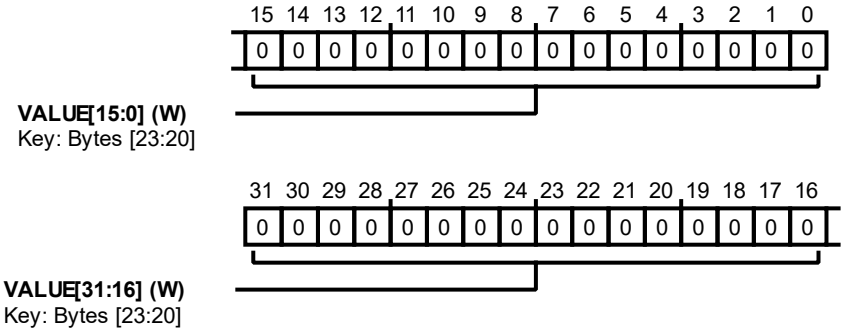


図 12-12：CRYPT_AESKEY5 レジスタ図

表 12-8：CRYPT_AESKEY5 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	VALUE	鍵：バイト［23：20］。

AES 鍵ビット［223：192］

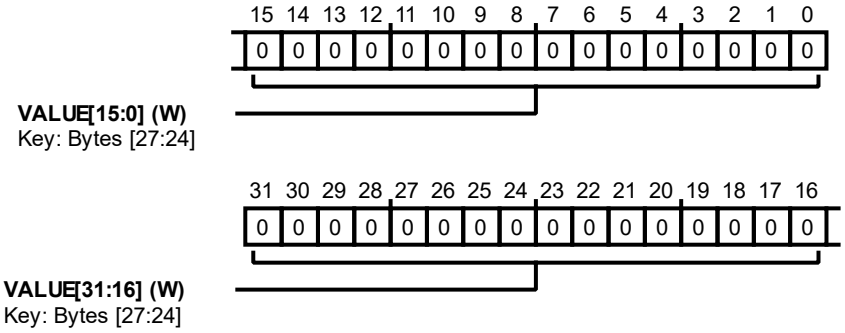


図 12-13：CRYPT_AESKEY6 レジスタ図

表 12-9：CRYPT_AESKEY6 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	VALUE	鍵：バイト［27：24］。

AES 鍵ビット［255：224］

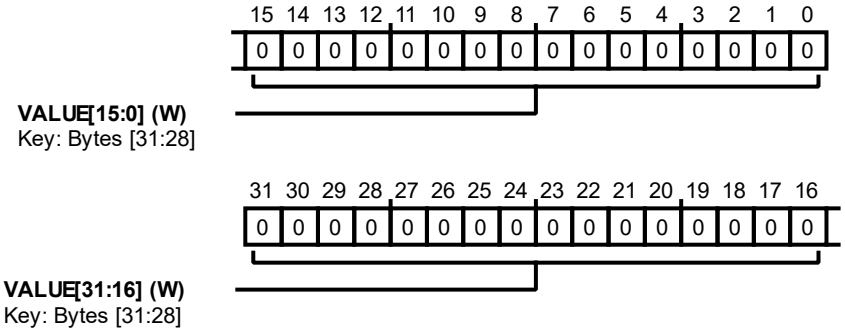


図 12-14：CRYPT_AESKEY7 レジスタ図

表 12-10：CRYPT_AESKEY7 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	VALUE	鍵：バイト［31：28］。

NUM_VALID_BYTES

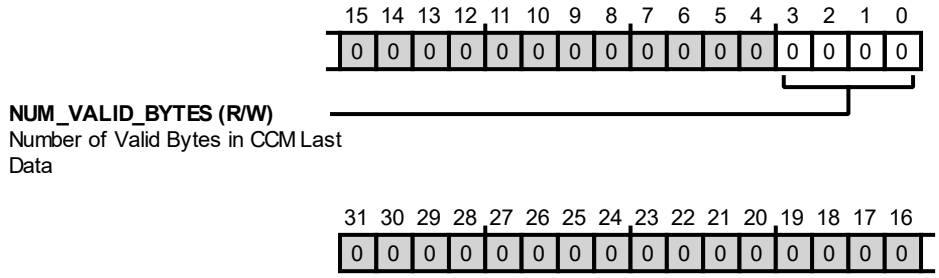


図 12-15：CRYPT_CCM_NUM_VALID_BYTES レジスタ図

表 12-11：CRYPT_CCM_NUM_VALID_BYTES レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
3:0 (R/W)	NUM_VALID_BYTES	CCM 最終データの有効バイト数。

設定レジスタ

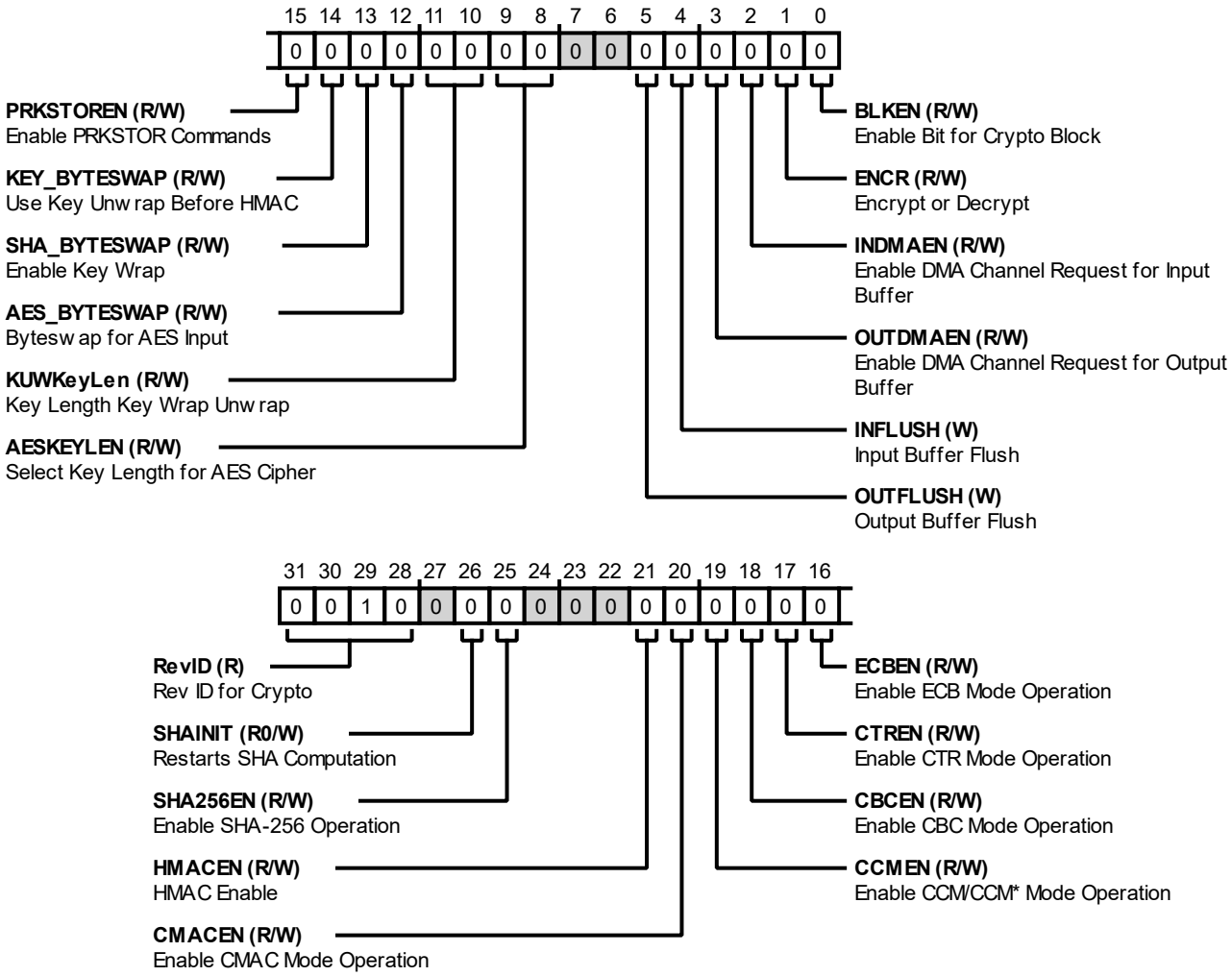


図 12-16：CRYPT_CFG レジスタ図

表 12-12：CRYPT_CFG レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:28 (R/NW)	REVID	暗号化ブロックのリビジョン ID。 このバージョンでは、リビジョン ID は 0x2 に設定されています。
26 (R0/W)	SHAINIT	SHA 計算を再開。 新しいデータ・ブロックの SHA 計算のフレッシュ・スタート用に、SHA モジュールを初期化します。このビットは、SHA 結果レジスタをデフォルト値にリセットします。これは、SHA 結果レジスタのリセット後に自動的にクリアされます。

表 12-12：CRYPT_CFG レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
25 (R/W)	SHA256EN	SHA-256 処理を有効にします。	
21 (R/W)	HMACEN	HMAC を有効化。 HMAC モードを有効にします。このビットは、HMAC 計算終了時に自動的にリセットされます。新たに HMAC 計算を開始するには、このビットをソフトウェアによってセットし直す必要があります。	
20 (R/W)	CMACEN	CMAC モード動作を有効化。	
19 (R/W)	CCMEN	CCM/CCM*モード動作を有効化。	
18 (R/W)	CBCEN	CBC モード動作を有効化。	
17 (R/W)	CTREN	CTR モード動作を有効化。	
16 (R/W)	ECBEN	ECB モード動作を有効化。	
15 (R/W)	PRKSTOREN	PRKSTOR コマンドを有効化。 保護された鍵の保存領域内でコマンドを実行するには、このビットをイネーブルする必要があります。	
14 (R/W)	KEY_BYTESWAP	HMAC の前に鍵のアンラップを使用。 このビットがセットされると、AES 鍵、KUW 鍵、および KUW 検証文字列レジスタへの入力ワードは、バイト・スワップされます。	
13 (R/W)	SHA_BYTESWAP	鍵ラップを有効化。 SHA モジュールへの各ワード入力のバイト・スワップを有効にします。｛B3, B2, B1, B0｝と入力されるデータは、｛B1, B2, B3, B4｝と記録されます。	
12 (R/W)	AES_BYTESWAP	AES 入力のバイト・スワップ。 入力バッファへの各ワード入力のバイト・スワップを有効にします。｛B3, B2, B1, B0｝と入力されるデータは、｛B1, B2, B3, B4｝と記録されます。	
11:10 (R/W)	KUWKEYLEN	ラップまたはアンラップされた鍵の長さ。 KUW レジスタ内の鍵の長さを、64 ビット・ブロック単位で示します。これらは、それぞれのモードでラップまたはアンラップされた鍵を格納するレジスタです。サポートする最大鍵長は 512 ビットです。鍵長を変更すると KUW レジスタがリセットされます。	
		1	KUW 鍵のサイズは 128 ビットです。
		2	KUW 鍵のサイズは 256 ビットです。
		3	KUW 鍵のサイズは 512 ビットです。

表 12-12：CRYPT_CFG レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
9:8 (R/W)	AESKEYLEN	AES 暗号の鍵長を選択。 AES 鍵の長さを指定します。サポートされている鍵長は 128 ビットと 256 ビットです。鍵長に変更を加えると AES 鍵がリセットされます。	
		0	長さ 128 ビットの鍵を使用
		2	長さ 256 ビットの鍵を使用
5 (RX/W)	OUTFLUSH	出力バッファをフラッシュ。	
4 (RX/W)	INFLUSH	入力バッファをフラッシュ。	
3 (R/W)	OUTDMAEN	出力バッファの DMA チャンネル要求を有効化。	
		0	出力バッファの DMA 要求を無効にします。
		1	出力バッファの DMA 要求を有効にします。
2 (R/W)	INDMAEN	入力バッファの DMA チャンネル要求を有効化。	
		0	入力バッファの DMA 要求を無効にします。
		1	入力バッファの DMA 要求を有効にします。
1 (R/W)	ENCR	暗号化または復号。 所定のモードの暗号化／復号を選択します。保護された鍵の保存領域内でラップ／アンラップ処理を行うためにこのビットをセット／リセットする必要はありません。	
		0	復号
		1	暗号化
0 (R/W)	BLKEN	暗号化ブロックのイネーブル・ビット。 暗号化ハードウェア・アクセラレータをイネーブルします。このビットは、すべての暗号化モードの動作開始として使用することを意図したものです。このビットをイネーブルしない限り、どの暗号化モードも機能しません。	
		0	暗号化ブロックをディスエーブル
		1	暗号化ブロックをイネーブル

カウンタ初期化ベクトル

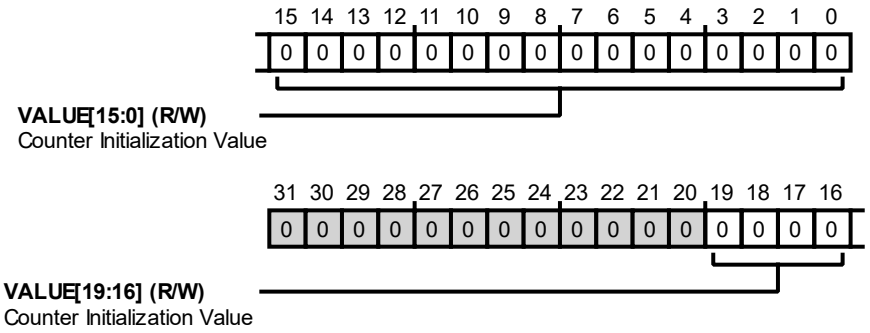


図 12-17：CRYPT_CNTRINIT レジスタ図

表 12-13：CRYPT_CNTRINIT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
19:0 (R/W)	VALUE	カウンタ初期化値。 これは、一部の動作モードに使われる内部カウンタ生成機能に使用する初期化値です。 1.CCM/CCM*モード：このモードのカウンタ初期化ベクトルの初期化には、CRYPT_CNTRINIT.VALUE［15：0］だけが使われます。これより上位のビットは無視されます。 2.CTR モード：このモードのカウンタ初期化ベクトルの初期化には、CRYPT_CNTRINIT.VALUE［19：0］が使われます。ブロックをリセットすると、カウンタはこのレジスタに設定された値に初期化されます。

ペイロード・データ長

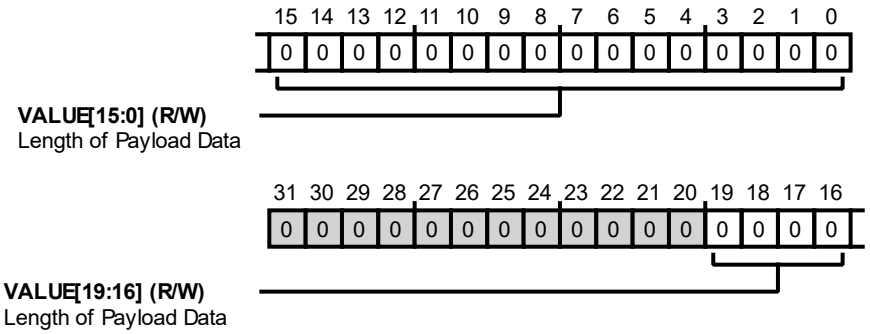


図 12-18：CRYPT_DATALEN レジスタ図

表 12-14：CRYPT_DATALEN レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
19:0 (R/W)	VALUE	ペイロード・データの長さ。 暗号化する必要のあるペイロードの長さを以下の値で設定します。 1.MAC モード：ペイロード内の 128 ビット・ブロックの数を設定するには、CRYPT_DATALEN.VALUE［19：0］を使用します。 2.CCM/CCM*モード：ペイロード内の 128 ビット・ブロックの数を設定するには、CRYPT_DATALEN.VALUE［15：0］を使用します。

入力バッファ

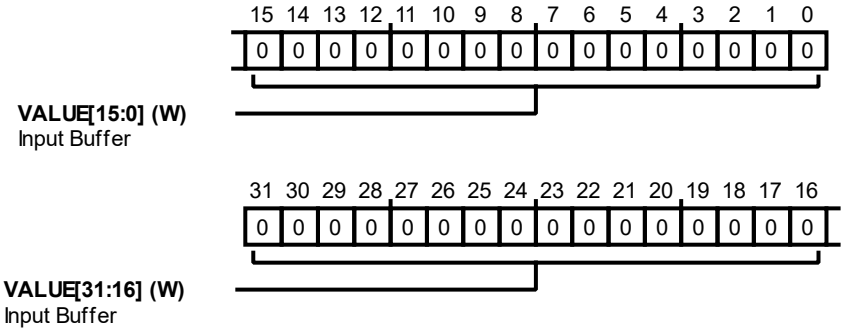


図 12-19：CRYPT_INBUF レジスタ図

表 12-15：CRYPT_INBUF レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	VALUE	入力バッファ。

割込みイネーブル・レジスタ

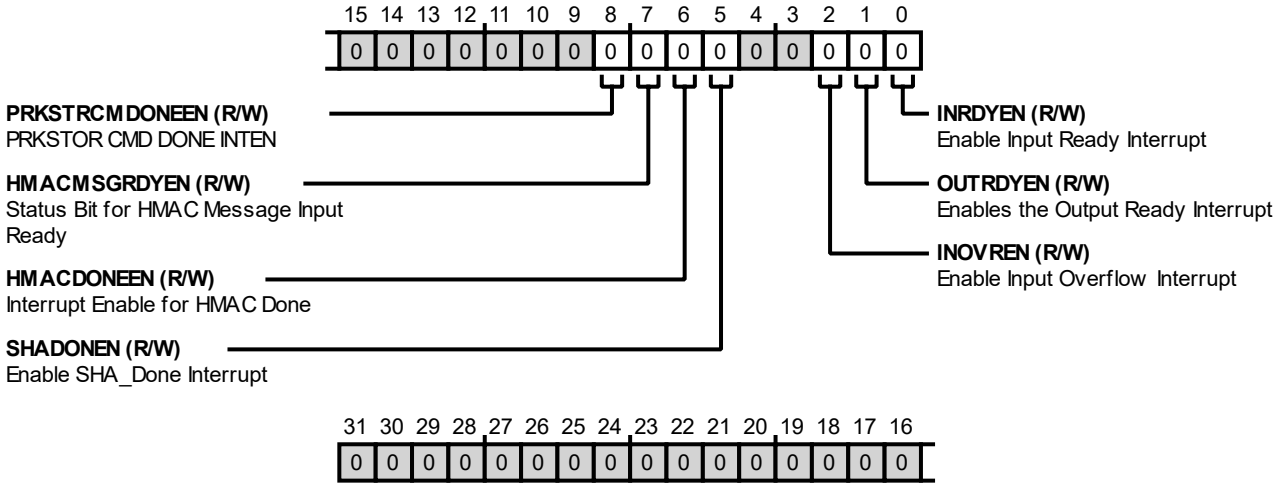


図 12-20：CRYPT_INTEN レジスタ図

表 12-16：CRYPT_INTEN レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
8 (R/W)	PRKSTRCMDONEEN	PRKSTOR CMD DONE INTEN。 このビットがセットされた場合は、PRKSTOR コマンド完了時にコアに割込みがかけられます。
7 (R/W)	HMACMSGRDYEN	HMAC メッセージ入力準備完了を示すステータス・ビット。 このビットがセットされた場合、HMAC アクセラレータにユーザ・データを取り込む準備が完了すると、コアに割込みがかけられます。ユーザ入力データの最終ブロックについては、SHA_LAST_WORD のセクションの説明に従って SHA_LAST_WORD レジスタを設定する必要があります。
6 (R/W)	HMACDONEEN	HMAC 完了時に割込みイネーブル。 このビットがセットされた場合は、HMAC 計算完了時にコアに割込みがかけられます。
5 (R/W)	SHADONEN	SHA_Done 割込みをイネーブル。 このビットがセットされた場合は、SHA が現在のデータ・ブロックの処理を完了するとコアに割込みがかけられます。
2 (R/W)	INOVREN	入力オーバーフロー割込みをイネーブル。 このビットがセットされた場合は、入力バッファがオーバーフローするとコアに割込みがかけられます。
1 (R/W)	OUTRDYEN	出力準備完了割込みをイネーブル。 このビットがセットされた場合は、出力バッファが更にデータを出力できる状態になると、コアに割込みがかけられます。このビットは、出力バッファが空になるとリセットされます。
0 (R/W)	INRDYEN	入力準備完了割込みをイネーブル。 このビットがセットされた場合は、入力バッファが更にデータを受け入れ可能な状態になると、コアに割込みがかけられます。このビットは、入力バッファがフルになるとリセットされます。

鍵ラップ／アンラップ・レジスタ 0

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

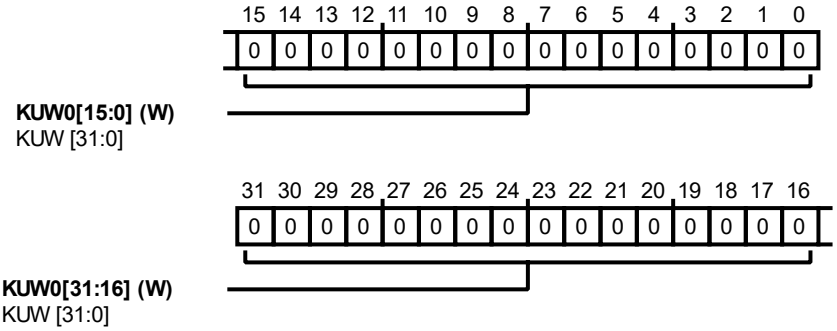


図 12-21：CRYPT_KUW0 レジスタ図

表 12-17：CRYPT_KUW0 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW0	KUW［31：0］。

鍵ラップ／アンラップ・レジスタ 1

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

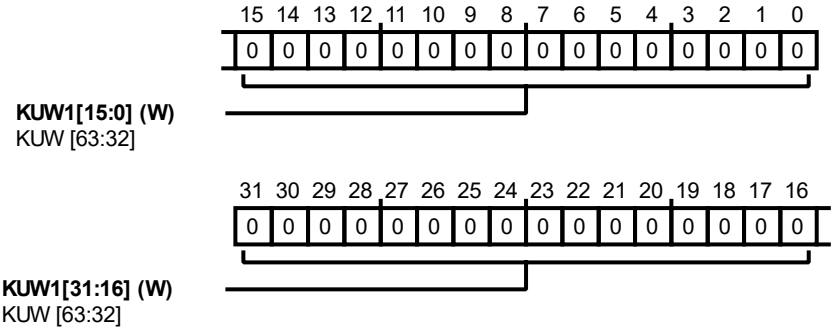


図 12-22：CRYPT_KUW1 レジスタ図

表 12-18：CRYPT_KUW1 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW1	KUW［63：32］。

鍵ラップ／アンラップ・レジスタ 10

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

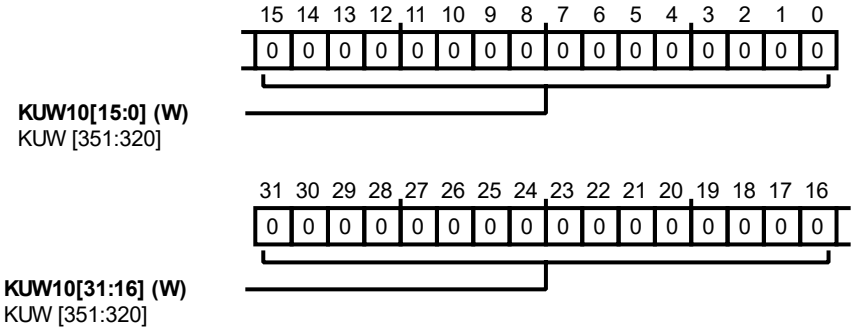


図 12-23：CRYPT_KUW10 レジスタ図

表 12-19：CRYPT_KUW10 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW10	KUW［351：320］。

鍵ラップ／アンラップ・レジスタ 11

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

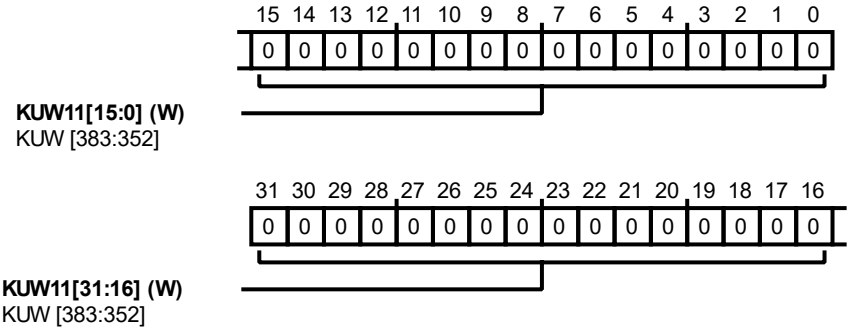


図 12-24：CRYPT_KUW11 レジスタ図

表 12-20：CRYPT_KUW11 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW11	KUW［383：352］。 KUW レジスタ・セットのレジスタ 11。

鍵ラップ／アンラップ・レジスタ 12

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

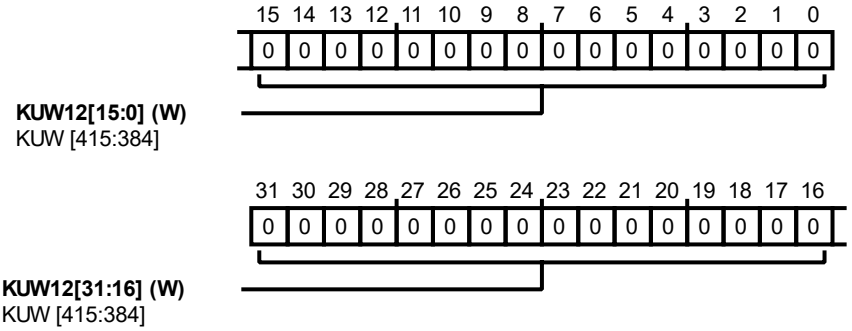


図 12-25：CRYPT_KUW12 レジスタ図

表 12-21：CRYPT_KUW12 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW12	KUW［415：384］。

鍵ラップ／アンラップ・レジスタ 13

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

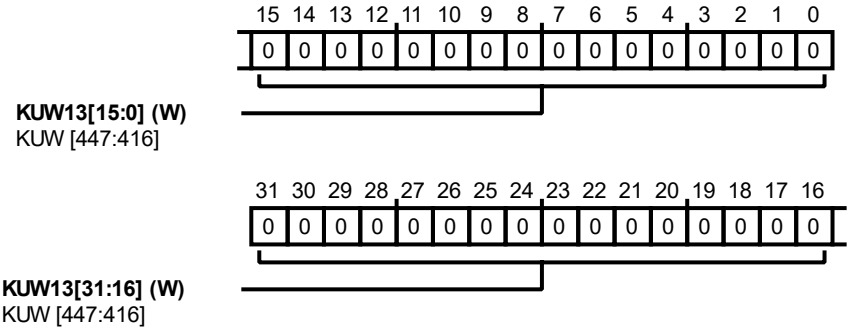


図 12-26：CRYPT_KUW13 レジスタ図

表 12-22：CRYPT_KUW13 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW13	KUW［447：416］。

鍵ラップ／アンラップ・レジスタ 14

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

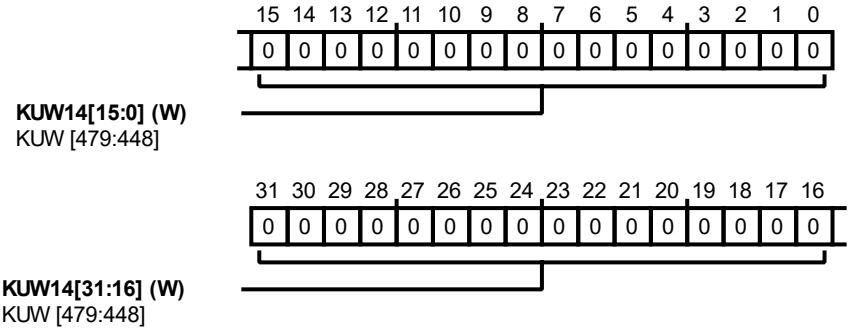


図 12-27：CRYPT_KUW14 レジスタ図

表 12-23：CRYPT_KUW14 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW14	KUW［479：448］。

鍵ラップ／アンラップ・レジスタ 15

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

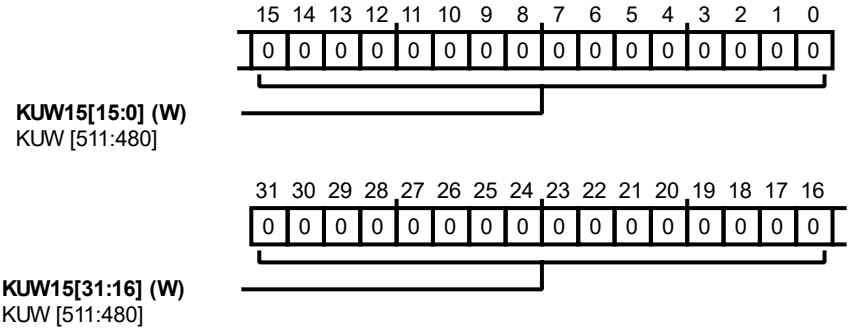


図 12-28：CRYPT_KUW15 レジスタ図

表 12-24：CRYPT_KUW15 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW15	KUW［511：480］。

鍵ラップ／アンラップ・レジスタ 2

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

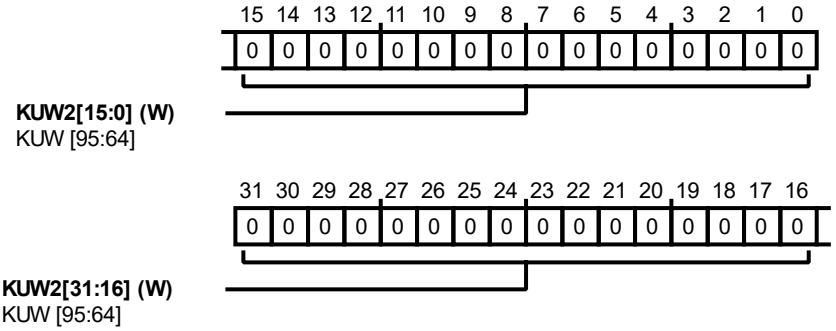


図 12-29：CRYPT_KUW2 レジスタ図

表 12-25：CRYPT_KUW2 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW2	KUW［95：64］。

鍵ラップ／アンラップ・レジスタ 3

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

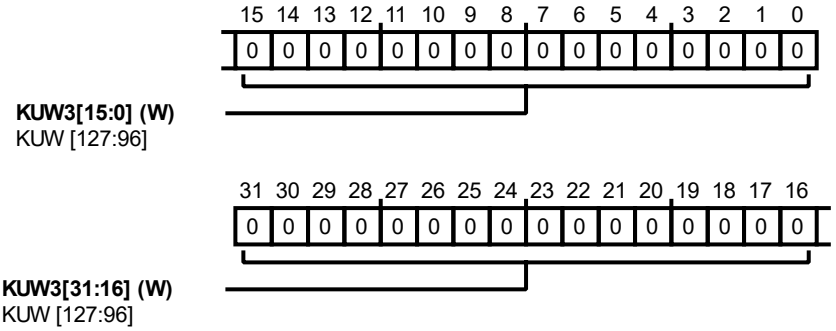


図 12-30：CRYPT_KUW3 レジスタ図

表 12-26：CRYPT_KUW3 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW3	KUW［127：96］。

鍵ラップ／アンラップ・レジスタ 4

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

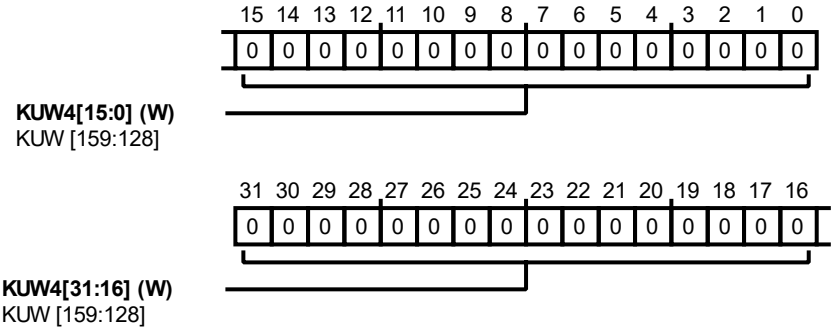


図 12-31：CRYPT_KUW4 レジスタ図

表 12-27：CRYPT_KUW4 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW4	KUW［159：128］。

鍵ラップ／アンラップ・レジスタ 5

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

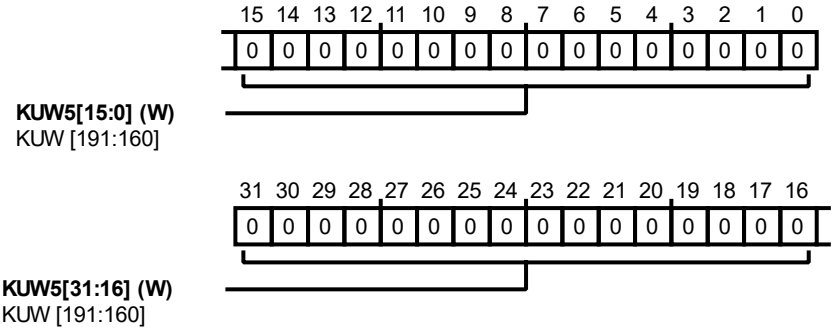


図 12-32：CRYPT_KUW5 レジスタ図

表 12-28：CRYPT_KUW5 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW5	KUW［191：160］。

鍵ラップ／アンラップ・レジスタ 6

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

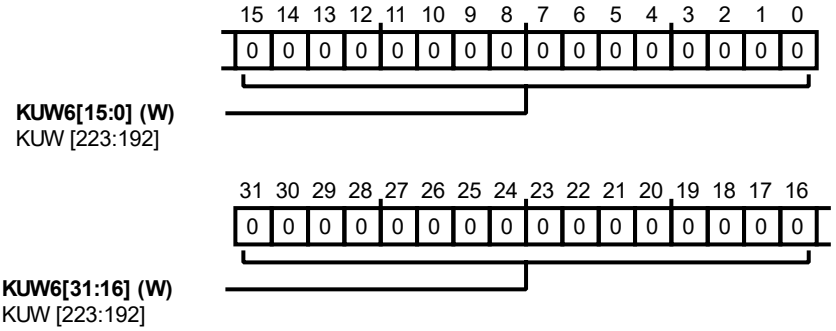


図 12-33：CRYPT_KUW6 レジスタ図

表 12-29：CRYPT_KUW6 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW6	KUW［223：192］。

鍵ラップ／アンラップ・レジスタ 7

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

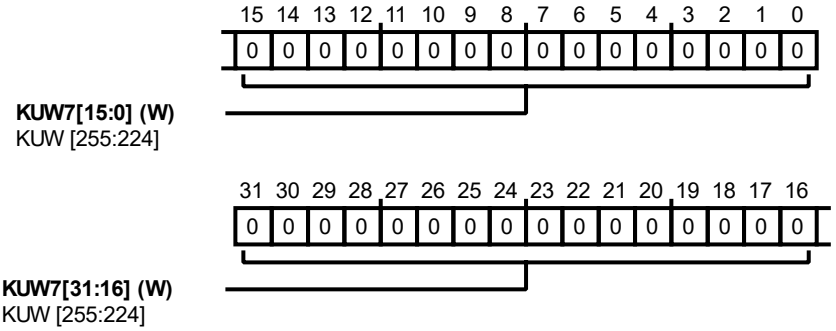


図 12-34：CRYPT_KUW7 レジスタ図

表 12-30：CRYPT_KUW7 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW7	KUW［255：224］。

鍵ラップ／アンラップ・レジスタ 8

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

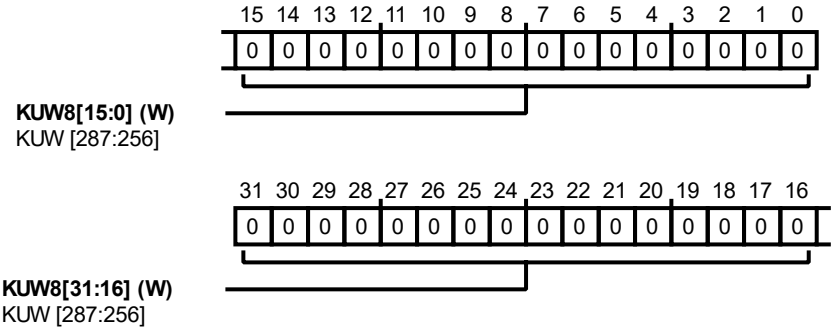


図 12-35：CRYPT_KUW8 レジスタ図

表 12-31：CRYPT_KUW8 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW8	KUW［287：256］。

鍵ラップ／アンラップ・レジスタ 9

このレジスタは、それぞれのモードでラップまたはアンラップされた鍵の一部を保持します。

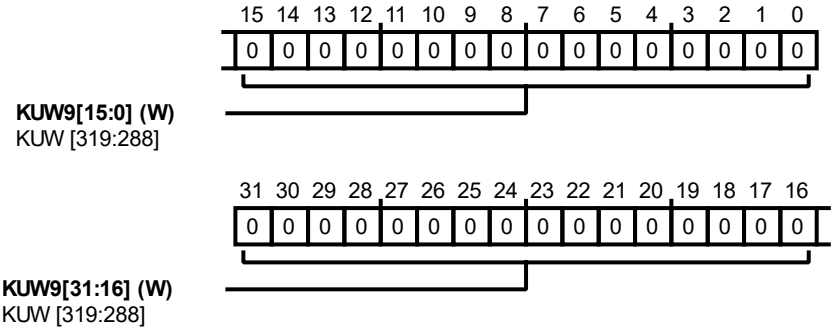


図 12-36：CRYPT_KUW9 レジスタ図

表 12-32：CRYPT_KUW9 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	KUW9	KUW［319：288］。

鍵ラップ／アンラップ検証文字列［63：32］

KUW 検証文字列［31：0］

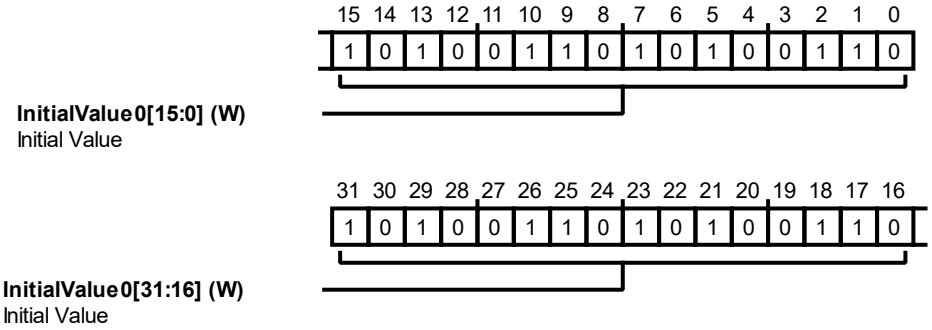


図 12-37：CRYPT_KUWVALSTR1 レジスタ図

表 12-33：CRYPT_KUWVALSTR1 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	INITIALVALUE0	初期値。 鍵ラップ／アンラップの初期値文字列。ラップ・モードではこれは完全性チェック文字列となり、ソフトウェアが使用して、ラップする鍵に完全性チェックを追加できます。アンラップ・モードでは、得られる文字列をソフトウェアによって照合し、鍵の完全性を確認する必要があります。

鍵ラップ／アンラップ検証文字列［31：0］

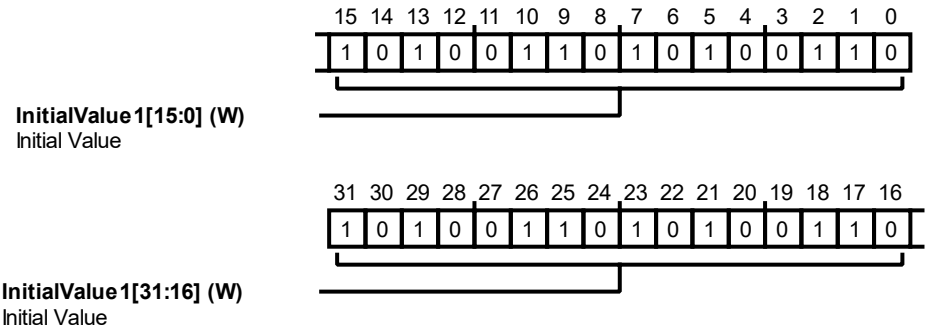


図 12-38：CRYPT_KUWVALSTR2 レジスタ図

表 12-34：CRYPT_KUWVALSTR2 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	INITIALVALUE1	初期値。 鍵ラップ／アンラップの初期値文字列。ラップ・モードではこれは完全性チェック文字列となり、ソフトウェアが使用して、ラップする鍵に完全性チェックを追加できます。アンラップ・モードでは、得られる文字列をソフトウェアによって照合し、鍵の完全性を確認する必要があります。

ノンス・ビット［31：0］

ノンスは一部の動作モードで使用し、モードに応じて異なる長さのノンスが使われます。

1. CTR モード：このモードでは 108 ビットのノンスを使用します。このノンスは次のように構成されます。

{CRYPT_NONCE3［11：0］, CRYPT_NONCE2, CRYPT_NONCE1, CRYPT_NONCE0}

2. CBC モード：このモードでは 128 ビットのノンスを使用します。このノンスは次のように構成されます。

{CRYPT_NONCE3, CRYPT_NONCE2, CRYPT_NONCE1, CRYPT_NONCE0}

3. CTR モード：このモードでは 108 ビットのノンスを使用します。このノンスは次のように構成されます。

{CRYPT_NONCE3［15：0］, CRYPT_NONCE2, CRYPT_NONCE1, CRYPT_NONCE0}

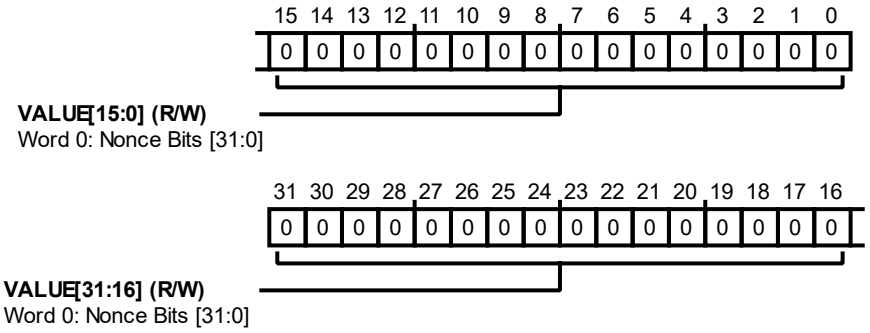


図 12-39：CRYPT_NONCE0 レジスタ図

表 12-35：CRYPT_NONCE0 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/W)	VALUE	ワード 0：ノンス・ビット［31：0］

ノンス・ビット［63：32］

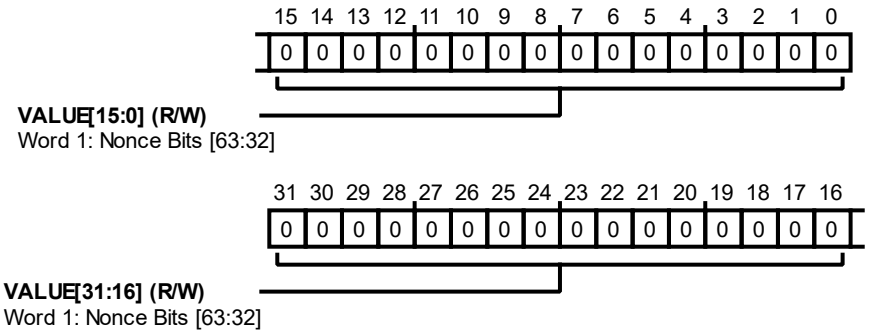


図 12-40：CRYPT_NONCE1 レジスタ図

表 12-36：CRYPT_NONCE1 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (R/W)	VALUE	ワード 1：ノンス・ビット［63：32］

ノンス・ビット［95：64］

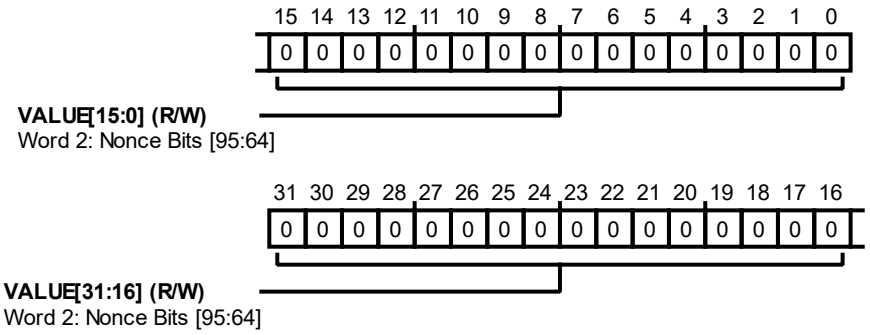


図 12-41：CRYPT_NONCE2 レジスタ図

表 12-37：CRYPT_NONCE2 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (R/W)	VALUE	ワード 2：ノンス・ビット［95：64］

ノンス・ビット［127：96］

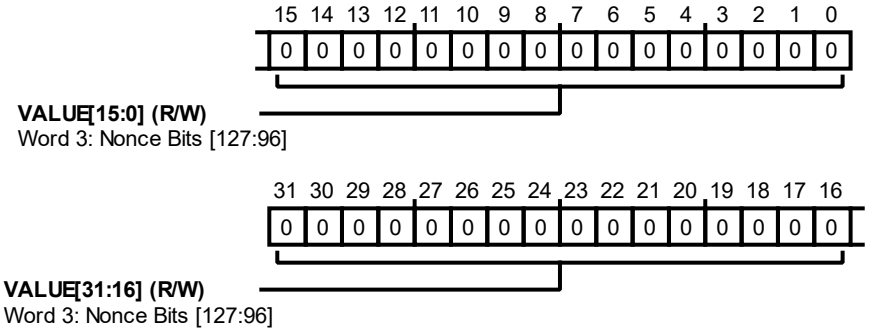


図 12-42：CRYPT_NONCE3 レジスタ図

表 12-38：CRYPT_NONCE3 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/W)	VALUE	ワード 3：ノンス・ビット［127：96］

出力バッファ

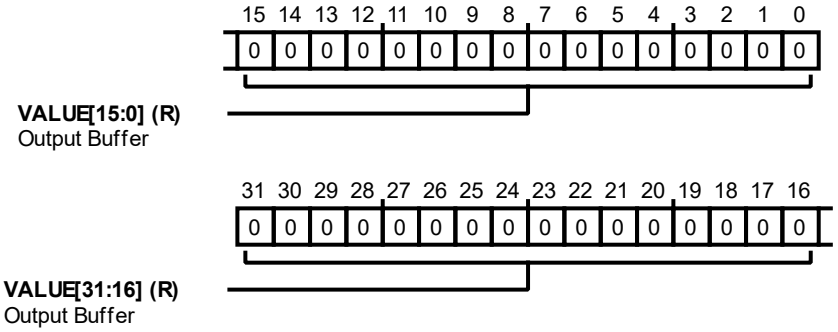


図 12-43：CRYPT_OUTBUF レジスタ図

表 12-39：CRYPT_OUTBUF レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	VALUE	出力バッファ。

認証データ長

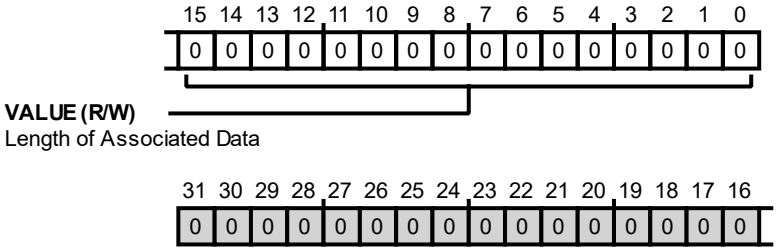


図 12-44：CRYPT_PREFIXLEN レジスタ図

表 12-40：CRYPT_PREFIXLEN レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	対応データの長さ。 暗号化する必要のある対応データの長さを以下の値で設定します。 CCM/CCM*モード：対応データの 128 ビット・ブロックの個数設定用に 20 ビットを使用できます。上位の 4 ビットは無視されます。

PRKSTOR 構成

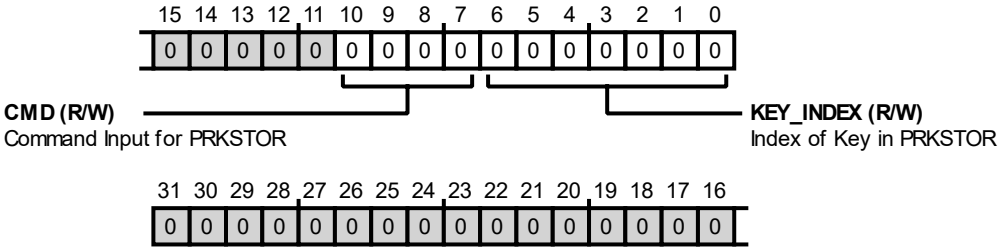


図 12-45：CRYPT_PRKSTORCFG レジスタ図

表 12-41：CRYPT_PRKSTORCFG レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
10:7 (R/W)	CMD	PRKSTOR のコマンド入力
6:0 (R/W)	KEY_INDEX	PRKSTOR 内の鍵のインデックス。

SHA ビット［31：0］

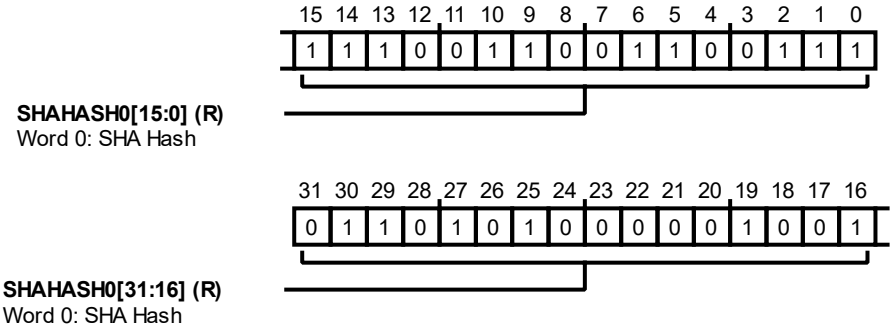


図 12-46：CRYPT_SHAH0 レジスタ図

表 12-42：CRYPT_SHAH0 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	SHAHASH0	ワード 0：SHA ハッシュ。

SHA ビット［63：32］

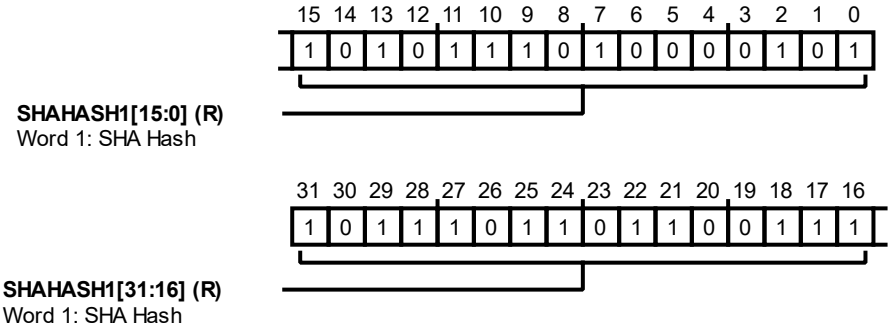


図 12-47：CRYPT_SHAH1 レジスタ図

表 12-43：CRYPT_SHAH1 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	SHAHASH1	ワード 1：SHA ハッシュ。

SHA ビット［95：64］

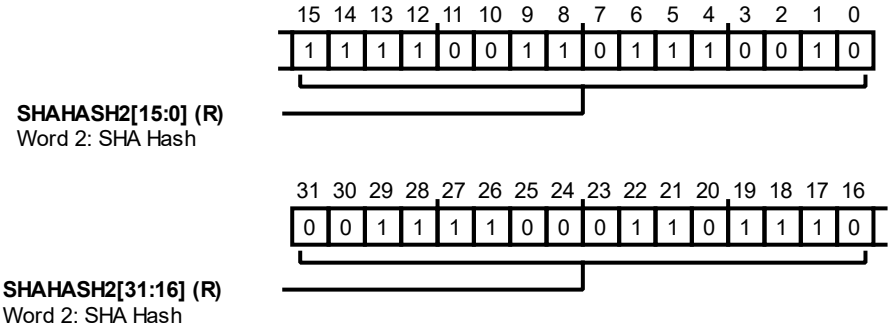


図 12-48：CRYPT_SHAH2 レジスタ図

表 12-44：CRYPT_SHAH2 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	SHAHASH2	ワード 2：SHA ハッシュ。

SHA ビット［127：96］

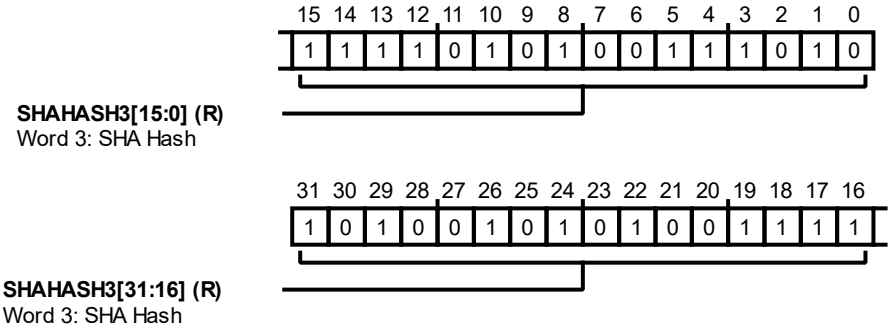


図 12-49：CRYPT_SHAH3 レジスタ図

表 12-45：CRYPT_SHAH3 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	SHAHASH3	ワード 3：SHA ハッシュ。

SHA ビット［159：128］

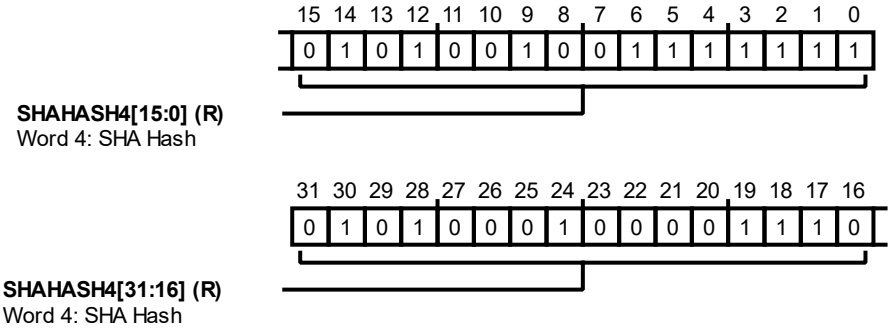


図 12-50：CRYPT_SHAH4 レジスタ図

表 12-46：CRYPT_SHAH4 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	SHAHASH4	ワード 4：SHA ハッシュ。

SHA ビット［191：160］

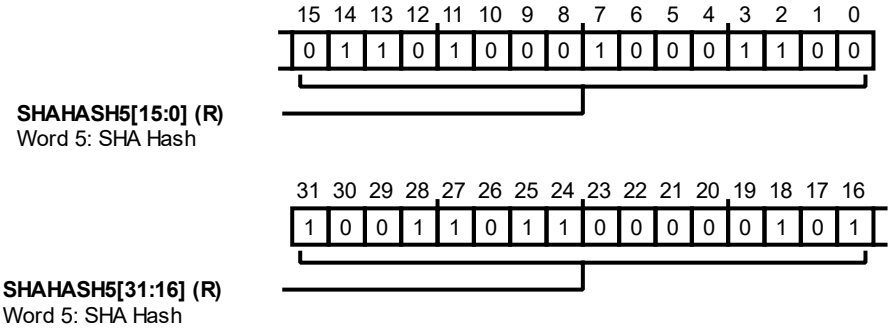


図 12-51：CRYPT_SHAH5 レジスタ図

表 12-47：CRYPT_SHAH5 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	SHAHASH5	ワード 5：SHA ハッシュ。

SHA ビット［223：192］

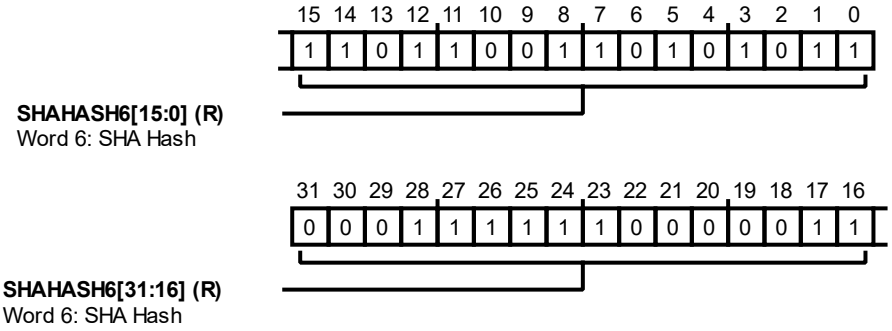


図 12-52：CRYPT_SHAH6 レジスタ図

表 12-48：CRYPT_SHAH6 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	SHAHASH6	ワード 6：SHA ハッシュ。

SHA ビット［255：224］

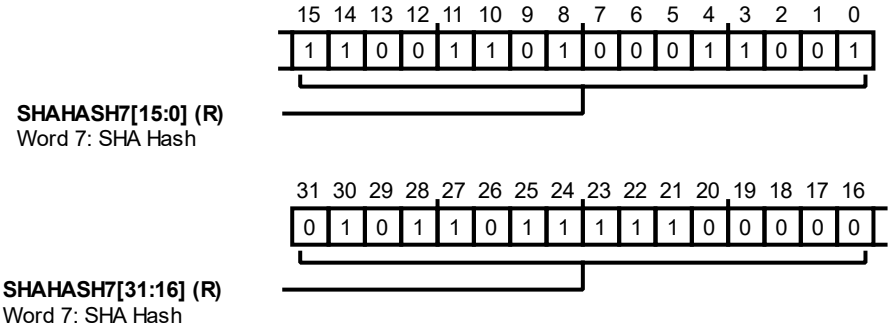


図 12-53：CRYPT_SHAH7 レジスタ図

表 12-49：CRYPT_SHAH7 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	SHAHASH7	ワード 7：SHA ハッシュ。

SHA 最終ワードおよび有効ビットの情報

このレジスタへの書込みは、SHA 入力レジスタへの最終ワード書込み前に行います。これは、CRYPT_SHA_LAST_WORD.O_LAST_WORD への書込みを行うことによって、最終ワードが間もなく書き込まれることを SHA エンジンに知らせるために使用します。なお、CRYPT_SHA_LAST_WORD.O_BITS_VALID に有効ビットの数を設定する必要があります。

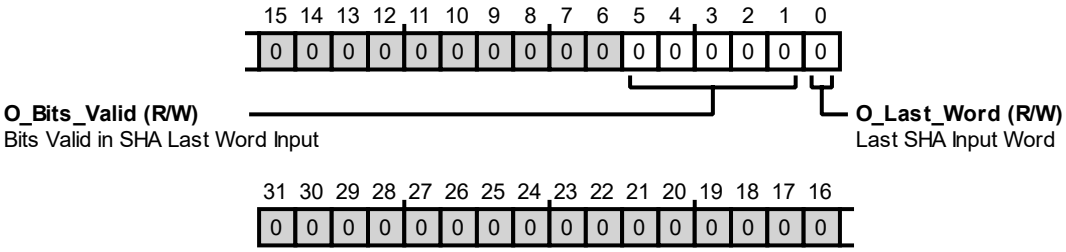


図 12-54：CRYPT_SHA_LAST_WORD レジスタ図

表 12-50：CRYPT_SHA_LAST_WORD レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
5:1 (R/W)	O_BITS_VALID	SHA 最終ワード入力の有効ビット。 SHA への最終入力ワード内の有効ビット数を示します。これは、CRYPT_STAT.SHADONE がセットされると自動クリアされます。
0 (R/W)	O_LAST_WORD	最終 SHA 入力ワード。 これは、最終ワードが間もなく SHA エンジンに書き込まれることを示すためにセットします。CRYPT_STAT.SHADONE がセットされると自動クリアされます。

ステータス・レジスタ

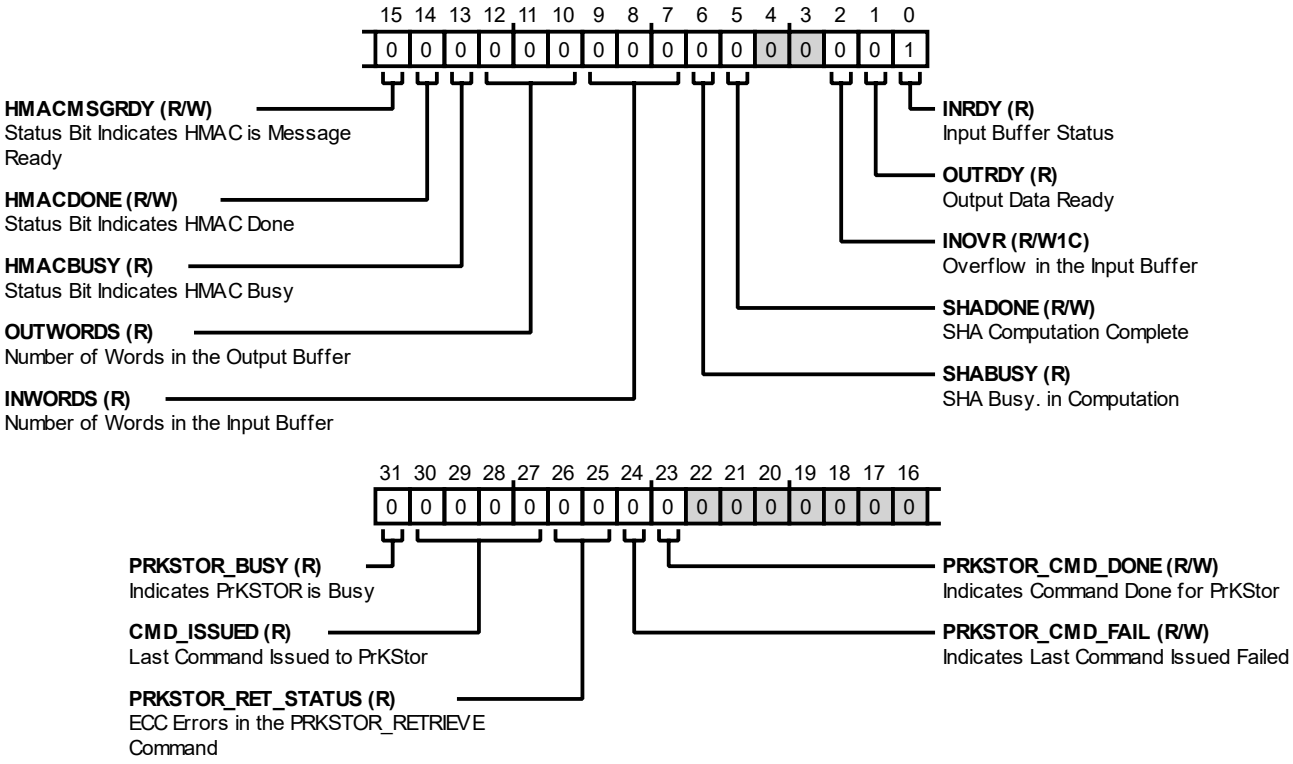


図 12-55：CRYPT_STAT レジスタ図

表 12-51：CRYPT_STAT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31 (R/NW)	PRKSTOR_BUSY	PrKSTOR がビジーであることを示します。 このビットがセットされた場合は、保護された鍵の保存領域がコマンドを実行中で、ビジー状態にあることを示します。このビットがセットされている間、保護された鍵の保護領域設定レジスタへのすべての書込みは無視されます。
30:27 (R/NW)	CMD_ISSUED	PrKStor に送出された最後のコマンド。 このフィールドは、保護された鍵の保存領域に送出された最後のコマンドを記録します。ブロックがディスエーブルされるとクリアされます。
26:25 (R/NW)	PRKSTOR_RET_STATUS	PRKSTOR_RETRIEVE コマンドの ECC エラー。 このフィールドは、ECC エラーが発生したかどうかを示します。
24 (R/W)	PRKSTOR_CMD_FAIL	最後に送出したコマンドの実行に失敗したことを示します。

表 12-51：CRYPT_STAT レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
23 (R/W)	PRKSTOR_CMD_DONE	PrKStor に対して実行されたコマンドを示します。
15 (R/W)	HMACMSGRDY	HMAC のメッセージに関する準備が完了したことを示すステータス・ビット。
14 (R/W)	HMACDONE	HMAC の実行が完了したことを示すステータス・ビット。
13 (R/NW)	HMACBUSY	HMAC がビジーであることを示すステータス・ビット。
12:10 (R/NW)	OUTWORDS	出力バッファ内のワード数。
9:7 (R/NW)	INWORDS	入力バッファ内のワード数。
6 (R/NW)	SHABUSY	SHA ビジー、計算中。 最新データの計算が進行中であることを示します。このビットがセットされている間、CRYPT_SHAHx レジスタ内の値は無効です。
5 (R/W)	SHADONE	SHA 計算が完了。 最新データのハッシュ計算が完了したことを示します。ハッシュ計算が完了すると、ハッシュの現在値を読み出すか新しいデータを入力バッファに書き込んで、更に計算を進めることができます。
2 (R/W1C)	INOVR	入力バッファのオーバーフロー。
1 (R/NW)	OUTRDY	出力データの準備完了。 出力データを読み出す準備ができました。
0 (R/NW)	INRDY	入力バッファ・ステータス。 計算を開始するには、入力バッファに更にデータが必要です。バッファがフルになるまでセットされたままになります。

13 真性乱数ジェネレータ（TRNG）

非決定論的値が必要な動作では、真性乱数ジェネレータ（TRNG）が用いられます。これによって、セキュアな通信のためにチャレンジを生成したり、暗号化された通信チャンネルで使用するキーを生成したりできます。ジェネレータを複数回実行して、目的の動作の強度に十分な数のビットを生成できます。真性乱数ジェネレータを使用して、NIST CRC SP-800-90A に記載されているような決定論的ランダム・ビット・ジェネレータのシードとすることができます。

TRNG 機能

TRNG には次のような特長があります。

- 十分なエントロピーが得られるよう、長さをプログラム可能。
- サンプリング・ジッタの特性を評価する発振器カウンタを内蔵。
- 割込みは生成不可。

TRNG 機能の説明

ここでは、ADuCM4050 MCU で使用される TRNG の機能について説明します。

TRNG のブロック図

次の図は、ADuCM4050 MCU で使用される TRNG のブロック図を示しています。

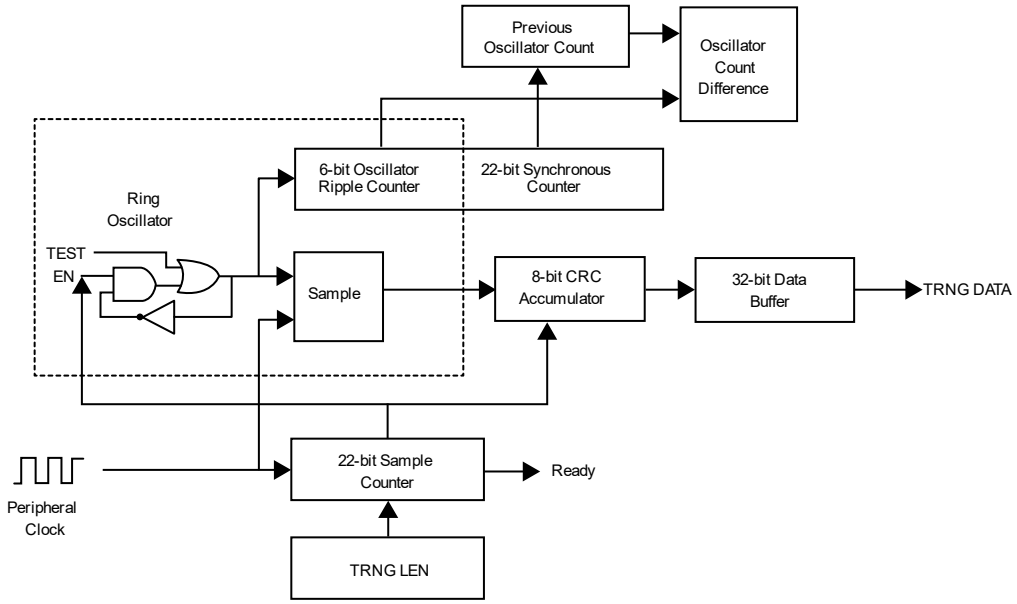


図 13-1：TRNG のブロック図

TRNG はサンプリングされた非同期クロック（この実装の場合はリング発振器）に基づきます。CRC を圧縮関数として使用し、大量の低エントロピー・ビットを少量の高エントロピー・ビットに変換することができます。CRC は、サンプル長レジスタで設定可能な数のサンプルを蓄積します。

TRNG はコントロール・レジスタによってイネーブルされます。サンプル長レジスタがサンプルの数を指定し、このサンプル数に十分なエントロピーが蓄積されるよう TRNG が実行されます。ペリフェラル・クロックあたり 1 つのサンプルが取得され、CRC によって圧縮されます。TRNG が設定された数のサンプルを蓄積すると、この蓄積された 8 ビットの CRC 結果は読出し可能となります。TRNG がレディ状態になると、ソフトウェアはステータス・ビット（RNG_STAT.RNRDY）をポーリングしたり、割込みを受けたりすることが可能となります。

任意の数だけ反復可能です。例えば、8 ビットの TRNG を最低 14 回読み出すことで、112 ビットのエントロピーを取得することができます。

TRNG 発振器カウンタ

TRNG ペリフェラルには発振器カウンタがあります。この内蔵カウンタを使用すれば、オフチップでは測定困難なサンプリング・ジッタの特性を評価することができます。サンプリング・ジッタは乱数ジェネレータのエントロピー源となります。

リング発振器のクロック数を所定のサンプリング周期にわたってカウントすることで、リング発振器のクロックとペリフェラルのサンプリング・クロックとの周波数比を定めることができます（ペリフェラルのクロック周波数が既知であれば、リング発振器の周波数が求まります）。

$$f_{OSCCLK} = f_{PCLK} \times \frac{OSCCNT}{SAMPCNT}$$

ここで、

OSCCNT は発振器のカウント数（RNG_OSCCNT.VALUE）です。

サンプリング・カウンタが使用する SAMPCNT（サンプリング時間の長さ）は、RNG_LEN レジスタで決まります。サンプリング・ジッタは、発振器の複数のカウント値の標準偏差を計算して求めます。

N 回の有効なループを実行して平均の発振器カウントとジッタを決定するための疑似コードは、次のとおりです。

```
sum=0; sum_sqr=0;
for（i=0;i<N;i++）
{
gen_rng（）；
sum += osc_cnt;
sum_sqr += osc_cnt*osc_cnt;
}
avg=sum/N;
std=sqrt（（sum_sqr-avg*sum）/（N-1））；
```

明確化のため、コードは単純化してあります。C 言語で実装する場合、

- sum および sum_sqr 変数は整数型であることが必要です。浮動小数点変数では、仮数部分がオーバーフローして結果を切り捨てるほどの長さに和が蓄積された場合、精度を欠くことになります。仮数部分全体を維持することが望ましく、そうすれば、浮動小数点型データの指数部は削除できます。
- sum 変数を格納するには、$\log_2$（osc_cnt）+ $\log_2$（N）ビットが必要です。
- sum_sqr 変数を格納するには、2 × $\log_2$（osc_cnt）+ $\log_2$（N）ビットが必要です。
- 2 乗演算（osc_cnt × osc_cnt および avg × avg）には、少なくとも 2 × $\log_2$（osc_cnt）のサイズを持つ型への変換が必要です。
- avg および std 変数は小数値（実数型または浮動小数点型）となる可能性があります。
- 割り算では小数型への変換が必要です。

正確なジッタ測定のためには、RNG_OSCCNT の標準偏差が 1 以上であることが必要です。1 未満の場合は、SAMPCNT を増加する必要があります。

発振器カウンタのジッタには、リング発振器のジッタとペリフェラル・クロックのジッタの両方が含まれています。これを利用してサンプリング・ジッタを求めることができます。ペリフェラル・クロックのジッタがわかれば、発振器クロックのジッタは次式から求めることができます。

$$\frac{\sigma_{OSCCNT}}{\sqrt{SAMPCNT}}\frac{SAMPCNT}{OSCCNT}\frac{1}{f_{PCLK}} = \sigma_{SAMPLE} = \sqrt{\sigma_{PCLK}^2 + \sigma_{OSCCLK}^2\frac{OSCCNT}{SAMPCNT}}$$

TRNG のエントロピーとサプライザル

サンプリング・クロックのジッタが乱数ジェネレータのエントロピー源となります。発振器を構成するトランジスタのノイズがジッタの一因です。長期ジッタは、時間またはクロック数の平方根に比例して蓄積されます。

$$\sigma_{TOTAL} = \sigma_{CLK}\sqrt{N}$$

1.0/ビットの理想的なエントロピーを実現するために必要な蓄積サンプル数は、システムのジッタ量によって決まります。エントロピーは次式で計算できます。

$$\text{average entropy} = -\sum p_i \log_2(p_i)$$

$$\text{minimum entropy} = \min(-\log_2(p_i))$$

各数値の発生確率は、理想的なジェネレータの場合、全数値に対して等しい、すなわち均一であることが必要です。不完全なジェネレータでは、ある数値を出力する頻度が他の数値に比べ高くなります。このような乱数ジェネレータは、この方法でエントロピーを測定または定量化できるような状態を維持できるよう設計されているとは言えません。乱数が生成されるごとに、CRC がリセットされリング発振器が同じ位相で起動します。システムのエントロピーが十分でない場合、ジェネレータは同じ数値をより高頻度に出力します。これは設計が原因となるもので、これによりエントロピーの評価が可能になります。不十分なエントロピーは SAMPCNT を低い値に設定すると生じるもので、この場合、ある乱数が他の乱数に比べ高頻度で発生します。SAMPCNT を増加させて、各数値の発生確率を均一にすることが必要です。

TRNG の最小エントロピーは、生成した乱数のヒストグラムを作成し最頻値の確率を計算することで、求めることができます。最小エントロピーの式は、どれだけ多くの真性乱数を決定論的乱数ジェネレータのシードとするかを決める際に使用します。

8 ビット・アキュムレータの理論上の最小サプライザルを取得するのに必要な SAMPCNT（RNG_LEN レジスタで設定）の最小値は、次の式で定まります。SAMPCNT を（様々な動作条件でのジッタ変動を考慮して）これより大きくしないまでも、少なくともこれと同じ大きさに設定することが、安全サイドの設計といえます。

$$SAMPCNT_{min} = \frac{1}{\sigma_{SAMPLE}^2 f_{OSC}^2}$$

攻撃者がシステムを物理的に不正に改変し、ペリフェラル・クロックを制御するようなことになった場合、上記の計算のサンプル・ジッタからペリフェラル・クロックによるジッタを除去する必要があります。これは、攻撃者が水晶発振器を低ジッタのクロック源に置き換える可能性があることを想定しています。また、物理的な攻撃者はシステムのノイズ量を削減するような良い電源を使用する可能性もあります。攻撃者がペリフェラル・クロックのジッタを除去または削減するような場合、乱数ジェネレータのエントロピー源の能力は低下してしまいます。システムはリング

発振器のジッタのみに依存すべきで、安全サイドで考えれば、ペリフェラル・クロックのジッタはゼロと想定する必要があります。

発振器カウントの差

乱数ジェネレータには、発振器の連続するカウント値の差を計算するロジックが組み込まれています。これを使用して分散を計算し、システムのジッタとそれに伴うエントロピーの量を求めることができます。これは乱数ジェネレータ・エントロピー源の健全性の指標となります。

サンプルの統計的な分散は、通常次式で表されます。

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

内蔵回路によって発振器カウントの現在と以前のサンプル値の差が計算されます。サンプルはそれぞれ独立に同じように生成されるため、平均の項は省略できます。

$$E[X] - E[X] = 0$$

$$E[(X - X)^2] = 2\sigma^2$$

発振器カウント値の差を 2 乗し、その 2 分の 1 の平均値を計算することで、分散が求まります。

$$OSCVAR = AVERAGE\left(\frac{(OSCCNT_i - OSCCNT_{i-1})^2}{2}\right)$$

この平均をローパス IIR フィルタで置き換えれば、分散の時間変動を追跡できます。

ADuCM4050 RNG レジスタの説明

乱数ジェネレータ（RNG）には次のレジスタがあります。

表 13-1：ADuCM4050 RNG レジスター一覧

レジスタ名	説明
RNG_CTL	RNG コントロール・レジスタ
RNG_DATA	RNG データ・レジスタ
RNG_LEN	RNG サンプル長レジスタ
RNG_OSCCNT	発振器カウント
RNG_OSCDIFF[n]	発振器差

表 13-1：ADuCM4050 の RNG レジスター一覧（続き）

レジスタ名	説明
RNG_STAT	RNG ステータス・レジスタ

RNG コントロール・レジスタ

RNG_CTL レジスタを使用して乱数ジェネレータをイネーブルできます。

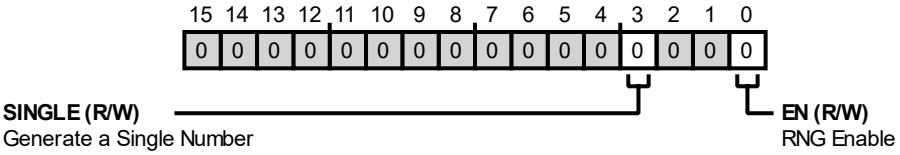


図 13-2：RNG_CTL レジスタ図

表 13-2：RNG_CTL レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
3 (R/W)	SINGLE	単一の数値を生成。 デフォルトでは、RNG は 4 個の 8 ビット値を生成しバッファして、32 ビットの乱数を供給します。このビットをセットすることで、RNG は単一の 8 ビット乱数を生成するのみとなります。	
		0	バッファ・ワード
		1	単一バイト
0 (R/W)	EN	RNG イネーブル。 RNG_CTL.EN をセットし、RNG_STAT.RNRDY をクリアした場合、リング発振器が起動し、RNG_LEN で指定されたサンプル数が RNG_DATA レジスタに蓄積されます。	
		0	RNG をディスエーブル
		1	RNG をイネーブル

RNG データ・レジスタ

RNG_DATA レジスタは、エントロピー・アキュムレータ（8 ビット CRC）とデータ・バッファへの読出し専用アクセスを CPU に許可します。データ・バッファがイネーブルでない場合は、8 ビットの結果が提供されます。データ・バッファがイネーブルの場合は、32 ビット（4 個の 8 ビット値）が提供されます。このレジスタの内容は、RNG_STAT.RNRDY ビットがセットされている場合に有効です。RNG_STAT.RNRDY ビットがクリアされると、このレジスタはリセットされます。このレジスタが読み出され CPU がデバッグ・ホールト状態でない場合、RNG_STAT.RNRDY ビットは自動的にクリアされます。RNG_CTL.EN がセットされているときに CPU がこのレジスタを読み出すと、新しい乱数が生成されます。

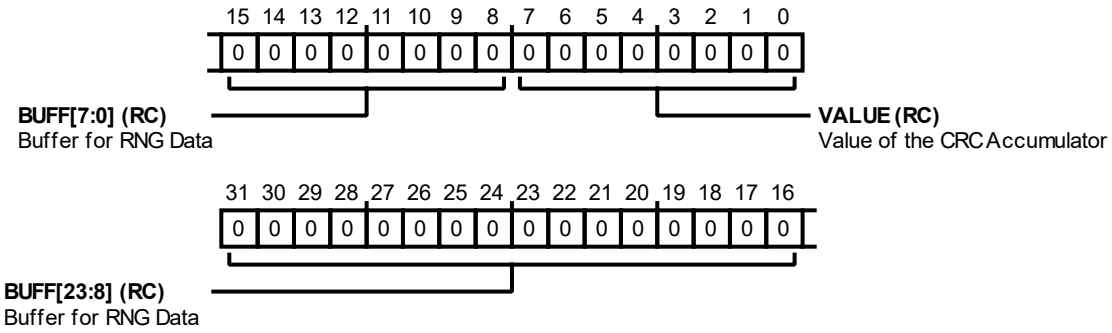


図 13-3：RNG_DATA レジスタ図

表 13-3：RNG_DATA レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:8 (RC/NW)	BUFF	RNG データ用バッファ。 32 ビット値を生成するよう設定されている場合、それまでに生成された 3 個の数値が RNG_DATA.BUFF に格納されます。
7:0 (RC/NW)	VALUE	CRC アキュムレータの値。 このレジスタで、エントロピー圧縮関数（8 ビット CRC アキュムレータ）のデータが提供されます。

RNG サンプル長レジスタ

RNG_LEN レジスタは、乱数発生時に CRC レジスタに蓄積するサンプルの数を指定します。蓄積サンプル数は、RNG_LEN.RELOAD の値に 2^RNG_LEN.PRESCALE の倍率を乗じたものになります。

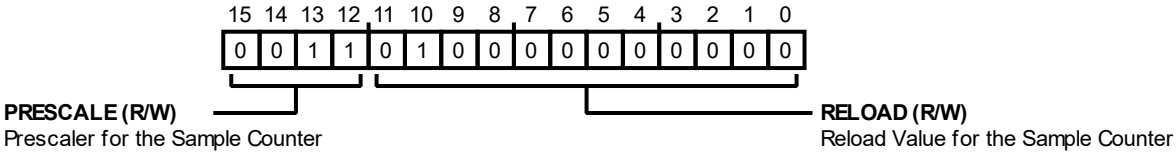


図 13-4：RNG_LEN レジスタ図

表 13-4：RNG_LEN レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:12 (R/W)	PRESCALE	サンプル・カウンタのプリスケーラ サンプル・カウンタのリロード値 RNG_LEN.RELOAD に 2^RNG_LEN.PRESCALE の倍率が乗じられます。プリスケーラは 10 ビット・カウンタです。プリスケーラに有効な値は 0～10 です。10 を超える値は最大プリスケーラ値に抑えられます。
11:0 (R/W)	RELOAD	サンプル・カウンタのリロード値 乱数発生時に CRC に蓄積するサンプルの数を指定します。

発振器カウント

発振器カウンタは、乱数発生時のリング発振器の周期数をカウントします。発振器カウンタは 28 ビットです。オーバーフローを防止するため、発振器カウンタは最大値で飽和します。

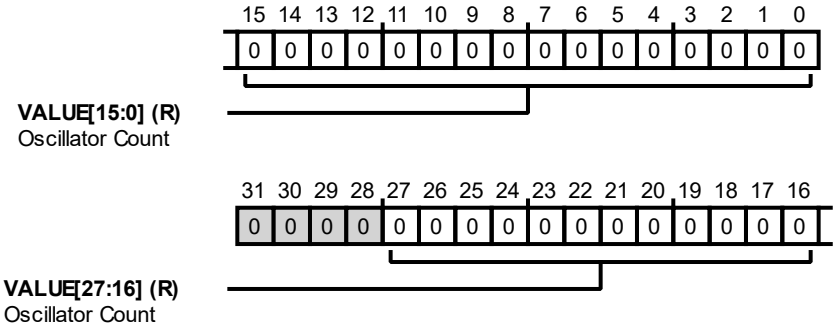


図 13-5：RNG_OSCCNT レジスタ図

表 13-5：RNG_OSCCNT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
27:0 (R/NW)	VALUE	発振器カウント このレジスタは RNG_STAT.RNRDY がセットされている場合のみ有効です。

発振器差

発振器差のレジスタには、RNG_OSCCNT の現在の値とその直前の値との差（RNG_OSCCNT［n］– RNG_OSCCNT［n-1］）が格納されます。この差は符号付きの 8 ビット値で表されます。これは最大値と最小値で飽和します。これを使用して、RNG_DATA バッファに現在格納されている値向けに RNG_OSCCNT を修復できます。この情報を使用して RNG_OSCCNT の分散を計算し、乱数ジェネレータの健全性をチェックしてエントロピーが適切であることを確認できます。

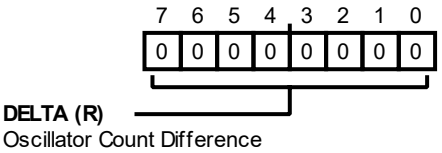


図 13-6：RNG_OSCDIFF［n］レジスタ図

表 13-6：RNG_OSCDIFF［n］レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
7:0 (R/NW)	DELTA	発振器カウントの差

RNG ステータス・レジスタ

RNG_STAT レジスタは RNG が乱数の生成を終えたことを示します。

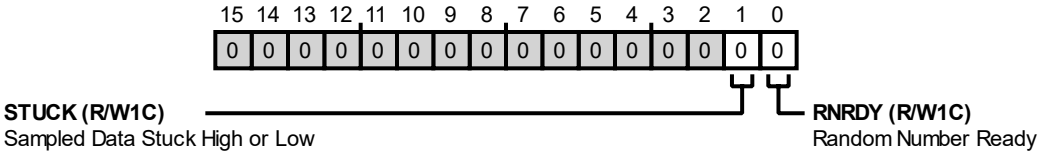


図 13-7：RNG_STAT レジスタ図

表 13-7：RNG_STAT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
1 (R/W1C)	STUCK	サンプリング・データがハイまたはローでスタック。 乱数発生時、ある回路がサンプリングされるリング発振器の出力をモニタします。この回路は、リング発振器の出力がハイおよびローでサンプリングされていて、一定値にスタックしてはいないことを確認するためのチェックを行います。このビットがセットされるとスティッキーとなり、クリアするには 1 を書き込みます。	
0 (R/W1C)	RNRDY	乱数レディ。 このビットは、RNG_DATA の値が読出し可能であることを示します。このビットがセットされると割込みが生成されます。リング発振器は、このビットがセットされると消費電力の節約のため停止します。RNG_DATA レジスタが読み出され CPU がデバッグ・ホールト状態で停止していない場合、このビットは自動的にクリアされます。このビットも 1 を書き込むことでクリアできます。	
		0	データ・レジスタは読出し不可。
		1	データ・レジスタは読出し可能。

14 巡回冗長検査（CRC）

CRC アクセラレータを使用して、メモリ・ロケーションのブロックについて CRC を計算できます。正確なメモリ・ロケーションは、SRAM、フラッシュ、または任意の組み合わせのメモリ・マップド・レジスタにあります。CRC アクセラレータが生成するチェックサムを使用して、シグネチャの予測値と比較します。最終的な CRC の比較は ADuCM4050 MCU が実行します。

CRC 機能

ADuCM4050 MCU で使用する CRC は、以下に示す機能を備えています。

- データ・ブロックの CRC シグネチャを生成。
- 最大 32 ビット長のプログラマブル多項式。
- 32 ビットのデータについて同時に動作し、また 1～8 ビットのデータ長にも対応。
- MSB ファーストと LSB ファーストの CRC を実装。
- 多様なデータ・ミラーリング機能。
- ユーザによるプログラム可能な初期シード。
- MCU をオフロードするためのデータ転送に使用する DMA コントローラ（ソフトウェア DMA を使用）

CRC 機能の説明

ここでは、ADuCM4050 MCU で使用する CRC アクセラレータの機能について説明します。メモリ・ブロックの CRC 計算においてアドレスのデクリメント／インクリメントを制御するオプションは、DMA コントローラにあります。

このオプションの詳細については、ダイレクト・メモリ・アクセス（DMA）を参照してください。

CRC のブロック図

CRC のブロック図を以下に示します。

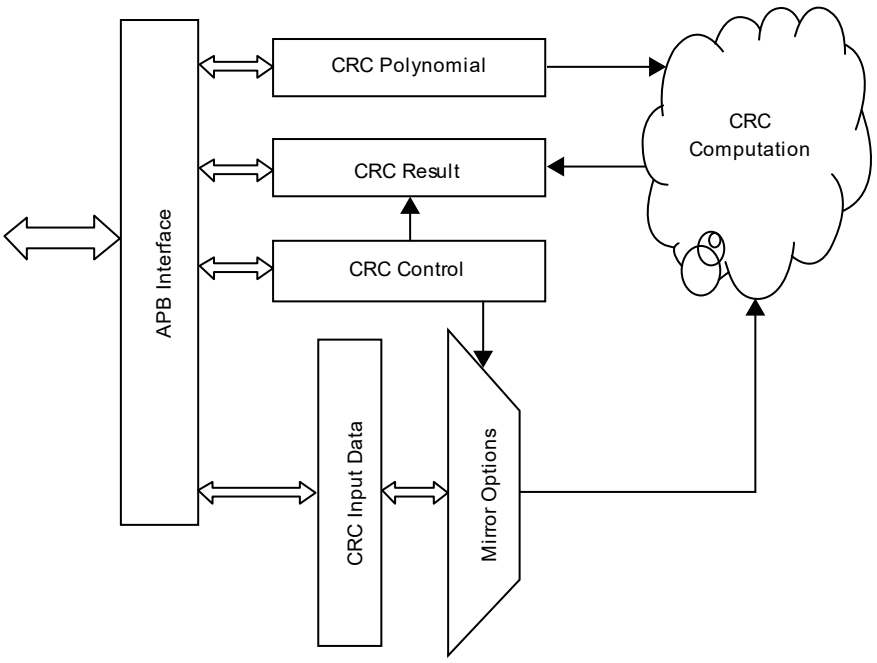


図 14-1：CRC のブロック図

CRC アクセラレータは、32 ビットのデータ・ワードを処理します。このデータ・ワードは、CRC アクセラレータ専用の DMA チャンネルを通じて、または MCU から直接ブロックに供給されます。そして、アクセラレータは CRC 出力を即座に提供します。

CRC 動作モード

このアクセラレータは、受信したデータ・ストリームの CRC を、通常は 32 ビット同時に計算します。データは、DMA エンジンを使用して、または MCU から直接ブロックに書き込まれます。

また、このアクセラレータは 1～8 ビットのデータ長に対応します。これは、あらゆるデータ長の CRC を計算できることを示しています。例えば、23 バイトのデータの CRC を計算するには、5 ワードと 3 バイトのデータとして CRC アクセラレータに供給します。また、このデータを 23 の個別のバイト書込みとして供給することもできます。ただし、より多くのサイクル数が必要となり、CRC の動作方法として効率的ではありません。データ長が 23 ビットの場合は、2 バイトと 7 ビットのデータとして入力できます。このアクセラレータは、MSB ファーストまたは LSB ファーストによる 1～32 ビット長の多項式を使用して、あらゆるデータ長を処理できる柔軟性を備えています。

入力データのミラーリングは、CRC エンジンで使用される前にビット、バイト、およびワード・レベルで処理することができ、CRC_CTL.BITMIRR、CRC_CTL.BYTMIRR、および CRC_CTL.W16SWP ビットで設定します。1 バイトに満たないデータにはミラーリング・オプションを適用できません。バイト・データに対してバイトのミラーリング・オプションやワードのスワップ・オプションの処理はできません。バイト・データに対しては、ビットのミラーリングのみ適用できます。

CRC アルゴリズムは、CRC_IPDATA レジスタに書き込まれた入力データ・ストリームに対して動作します。新しいデータ・ワードを受信するたびに、CRC を計算して結果を CRC_RESULT レジスタに更新します。CRC アクセラレータ

は現在のデータの CRC を計算して、結果を CRC 結果レジスタに即座に格納します。この値は MCU によって読み出すことができます。

新しいデータ・ワードを受信すると、CRC エンジンは現在の CRC_RESULT を使用して次の CRC_RESULT を生成します。CRC_RESULT レジスタは初期シードを使ってプログラムできます。n ビット多項式のシード値のビット幅は n でなければなりません。また、シードは端を揃えて CRC_RESULT レジスタに格納します。

多項式

この CRC アクセラレータは、どのような長さの多項式を使用する CRC の計算にも対応できます。多項式は CRC_POLY レジスタに書き込みます。MSB ファーストで実行する場合、CRC 多項式レジスタへのプログラミング時に最高次の値は削除され、多項式は左詰めに格納されます。LSB ファーストで実行する場合、多項式は右詰めに格納され、LSB が削除されます。CRC_RESULT レジスタには、n ビットの CRC 多項式のチェックサムとして使用する n ビットの MSB が格納されます。

以下に CRC 多項式の例を示します。

MSB ファースト計算用の 16 ビット多項式のプログラミング

多項式：CRC-16-CCITT、$x^{16} + x^{12} + x^5 + 1$ = （1）0001 0000 0010 0001 = 0x1021

最大次数（x^{16} の項）を省略するため、0001 0000 0010 0001 となります

多項式レジスタに左詰めに格納すると、以下のようになります

CRC 多項式レジスタ（CRC_POLY）

0001 0000	0010 0001	8b0	8b0

CRC 結果レジスタ（CRC_RESULT）

CRC	Result	8b0	8b0

CRC 結果レジスタ（CRC_RESULT）にプログラムされた初期シード

CRC	SEED	8b0	8b0

LSB ファースト計算用の 16 ビット多項式のプログラミング

多項式：CRC-16-CCITT、$x^{16} + x^{12} + x^5 + 1$ = 1000 0100 0000 1000（1） = 0x8408

最小次数（x^0 の項）を省略するため、1000 0100 0000 1000 となります
多項式レジスタに右詰めに格納すると、以下のようになります

CRC 多項式レジスタ（CRC_POLY）

8b0	8b0	1000 0100	0000 1000

CRC 結果レジスタ（CRC_RESULT）

8b0	8b0	CRC	Result

CRC 結果レジスタ（CRC_RESULT）にプログラムされた初期シード

8b0	8b0	CRC	SEED

MSB ファースト計算用の 8 ビット多項式のプログラミング

多項式：CRC-8-ATM、$x^8 + x^2 + x + 1$ = （1）0000 0111 = 0x07

最大次数（x^8 の項）を省略するため、0000 0111 となります

多項式レジスタに左詰めに格納すると、以下のようになります

CRC 多項式レジスタ（CRC_POLY）

0000 0111	8b0	8b0	8b0

CRC 結果レジスタ（CRC_RESULT）

CRC RESULT	8b0	8b0	8b0

CRC 結果レジスタ（CRC_RESULT）にプログラムされた初期シード

CRC SEED	8b0	8b0	8b0

LSB ファースト計算用の 8 ビット多項式のプログラミング

多項式：CRC-8-ATM、$x^8 + x^2 + x + 1$ = 1000 0011（1） = 0x83

最小次数（x^0 の項）を省略するため、1000 0011 となります

多項式レジスタに右詰めに格納すると、以下のようになります

CRC 多項式レジスタ（CRC_POLY）

8b0	8b0	8b0	1000 0011

CRC 結果レジスタ（CRC_RESULT）

8b0	8b0	8b0	CRC RESULT

CRC 結果レジスタ（CRC_RESULT）にプログラムされた初期シード

8b0	8b0	8b0	CRC SEED

CRC エンジンは、デフォルトとして以下に示す 32 ビットの CRC 多項式を使用します（IEEE 802.3）。

$g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

これは、デフォルトで MSB ファースト計算用に以下のようにプログラムされています。

0x04	0xC1	0x1D	0xB7

リセットおよび休止モード

1. ブロック・イネーブル・ビット（CRC_CTL.EN）以外の CRC 設定ビットは保持されます。ブロックは、休止モードの終了後に再びイネーブルする必要があります。
2. CRC 多項式および CRC 結果レジスタは、休止モードの終了後も保持されています。

表 14-1：リセットおよび休止モード

レジスタ	リセット	休止
CRC_CTL	0x0	BLKEN を除くすべてのビットが保持される
CRC_POLY	0x04C11DB7	保持される
CRC_IPDATA CRC_IPBYTE CRC_IPBITS[n]	0x0	保持されない（0x0）
CRC_RESULT	0x0	保持される

入力データ長

様々なデータ長とミラーリング・オプションによる CRC アクセラレータの使用方法の例を以下に示します。

1. ワード（4 バイト）の倍数のデータ長

   多項式：最大 32 ビットの任意の長さ、MSB ファーストまたは LSB ファースト計算
   データ長：512 ビット

   CRC_CTL レジスタは、適切なオプションを選択して設定してください。

   多項式は、多項式のセクションの説明に従ってプログラムしてください。

   データ長は 16 ワードです。DMA をセットアップして、この 16 ワードを CRC_IPDATA レジスタに転送できます。DMA による channel_done 割込みを受信後、コアは結果を読み出すことができます。
2. ワードの倍数ではなくバイトの倍数のデータ長

   多項式：最大 32 ビットの任意の長さ、MSB ファーストまたは LSB ファースト計算

   データ長：496 ビット

   CRC_CTL レジスタは、適切なオプションを選択して設定してください。

   多項式は、多項式のセクションの説明に従ってプログラムしてください。

データ長は 15 ワードと 2 バイトです。DMA をセットアップして、この 15 ワードを CRC_IPDATA レジスタに転送できます。dma_done 割込みを受信後、コアは残りの 2 バイトを CRC_IPBYTE レジスタに書き込むことができます。コアが残りの 2 バイトの書込みを完了した直後に、結果を読み出すことができます。

DMA をバイト転送モードに設定して CRC アクセラレータにデータを供給できます。このモードでは、DMA は一度に 1 バイトずつ CRC_IPBYTE レジスタにデータを転送します。このモードは、他の方法と比較して CRC アクセラレータをアクティブにする回数が約 4 倍になるため、推奨しません。また、バイト・データの CRC 計算に適用できるのは、ビットのミラーリング・オプションのみです。

3. バイトの倍数でもワードの倍数でもないデータ長

   多項式：最大 32 ビットの任意の長さ、MSB ファーストまたは LSB ファースト計算

   データ長：51 ビット

   CRC_CTL レジスタは、適切なオプションを選択して設定してください。

   多項式は、多項式のセクションの説明に従ってプログラムしてください。

   データ長は 1 ワード、2 バイト、3 ビットです。データは、以下の順番で CRC アクセラレータに供給する必要があります。まず 1 ワードを CRC_IPDATA レジスタに書き込みます。次に 2 バイトを CRC_IPBYTE レジスタに書き込み、その後、CRC_IPBIT3 レジスタに書き込みます。残りのデータが n ビット（ここで n ε［1、7］）のときは、CRC_IPBITS［n］レジスタに書き込みます。

   CRC 計算が MSB ファーストのとき、n ビットのデータは入力データの MSB に位置を合わせます。LSB ファーストのときは LSB に合わせます。

   例として、残りのデータが 3 ビット（例えば 3'b101）の場合を考えてみます。MSB ファースト計算の場合、入力バイトは 8' b101_00000 となるように格納する必要があります。LSB ファーストの場合は、8' b00000_101 となるように格納します。このバイトは CRC_IPBIT3 レジスタに書き込まれます。ミラーリング・オプションはこの残りのデータには適用できません。

CRC のデータ転送

データ・ストリームは、DMA コントローラを使用して、または MCU から直接、ブロックに書き込むことができます。

CRC の割込み、および例外

DMA チャンネルは、CRC ブロックへのデータ転送を完了すると割込みを生成します。

CRC のプログラミング・モデル

このブロックは、コアが他のタスクを処理している間に、バックグラウンドでデータ・ブロックの CRC シグネチャを計算します。

CRC ブロックは、コア・アクセスと DMA アクセスの 2 つのモードの CRC 計算に対応できます。

コア・アクセス

1. 多項式のセクションの例で示したように、必要な多項式を、適切に端を揃えて CRC_POLY レジスタにプログラムします。
2. 初期シードは CRC_RESULT レジスタにプログラムします。このシードは、多項式で説明したように、CRC_RESULT レジスタ内で適切に端を揃えて書き込まなければなりません。
3. CRC_CTL レジスタに書き込むことによって CRC アクセラレータ・ブロックを始動します。
   a. CRC_CTL.EN ビットをハイに設定します。
   b. CRC_CTL.W16SWP ビット、CRC_CTL.BITMIRR ビット、CRC_CTL.BITMIRR ビットを変更することで、アプリケーションを様々なミラーリング・オプションに設定できます。詳細については、ミラーリング・オプションを参照してください。
   c. CRC_CTL.LSBFIRST ビットをセット／リセットして、CRC 計算における LSB ファースト／MSB ファーストを決定します。

   注：ステップ 3 のサブステップで CRC_CTL レジスタに書き込む必要があるのは、1 回だけです。

   CRC_IPDATA レジスタに書き込むことによって、コアは CRC ブロックへのデータ送信を開始できます。CRC アクセラレータは、CRC_IPDATA レジスタにデータが書き込まれる限り、CRC の計算を続けます。CRC ブロックに書き込まれたワード数のカウントは、アプリケーションで行わなければなりません。すべてのワードが書き込まれると、アプリケーションは CRC_RESULT レジスタを読み出すことができます。
4. CRC_RESULT レジスタを読み出します。CRC 計算において、MSB ファーストの場合は n 個の MSB ビット、LSB ファーストの場合は n 個の LSB ビットで構成された、n ビットの結果が格納されています。
5. 次のデータ・ブロックの CRC を計算します。次のデータ・ブロックの CRC を計算するには、*ステップ 1～4* を繰り返します。
6. CRC_CTL.EN ビットをクリアして、CRC アクセラレータ・ブロックをディスエーブルします。これにより、ブロックを低消費電力の状態にできます。

DMA アクセス

CRC アクセラレータ・ブロックはソフトウェア DMA をサポートしています。

1. 多項式のセクションの例で示したように、必要な多項式を左詰めで CRC_POLY レジスタにプログラムします。
2. CRC_RESULT レジスタに初期シード値をプログラムします。このシードは、多項式で説明したように、CRC_RESULT レジスタ内で適切に端を揃えて書き込まなければなりません。
3. CRC_CTL レジスタに書き込むことにより、アクセラレータ機能を有効にします。
   a. CRC_CTL.EN ビットをハイに設定します。
   b. CRC_CTL.W16SWP ビット、CRC_CTL.BITMIRR ビット、CRC_CTL.BITMIRR ビットを変更することで、様々なミラーリング・オプションにアプリケーションを設定できます。詳細については、ミラーリング・オプションを参照してください。
   c. CRC_CTL.LSBFIRST ビットをセット／リセットして、CRC 計算における LSB ファースト／MSB ファーストを決定します。

注：*ステップ3* のサブステップで CRC_CTL レジスタに書き込む必要があるのは、1 回だけです。

CRC_IPDATA に書き込むことによって DMA は CRC データの送信を開始します。CRC アクセラレータ・ブロックは、CRC_IPDATA レジスタにデータが書き込まれる限り、CRC の計算を続けます。

4. 必要に応じて DMA チャンネルをセットアップします。DST_END_PTR の値は、CRC_IPDATA レジスタ・アドレスです。データ・サイズはワードです。使用するチャンネルにはアドレス・デスティネーションをインクリメントしないオプションを使用してください。DMA のプログラミングの詳細については、プログラミングのガイドラインを参照してください。
5. dma_done は DMA チャンネルの割込み信号で、CRC ブロックへのデータ転送が完了したことを示します。
6. すべてのデータがアクセラレータ・ブロックに送信されるまで*ステップ1～4* を繰り返します。
7. CRC_RESULT レジスタを読み出します。CRC 計算において、MSB ファーストの場合は n 個の MSB ビット、LSB ファーストの場合は n 個の LSB ビットで構成された、n ビットの結果が格納されています。
8. 次のデータ・ブロックの CRC を計算します。次のデータ・ブロックの CRC を計算するには、*ステップ1～5* を繰り返します。
9. CRC_CTL.EN ビットをクリアして、CRC アクセラレータ・ブロックをディスエーブルします。これにより、ブロックを低消費電力の状態にできます。

ミラーリング・オプション

CRC_CTL.W16SWP、CRC_CTL.BITMIRR、および CRC_CTL.BYTMIRR ビットを使用して、CRC を計算するビットの順番を決定します。

32 ビット多項式を使用する 32 ビット入力データのミラーリング・オプションの表に、32 ビット多項式に対してこのブロックで使用するミラーリングの全オプションを示します。

DIN［31：0］を CRC_IPDATA レジスタに書き込まれたデータ、CIN［31：0］をミラーリングが行われた後のデータとします。シリアル・エンジンは、MSB ビットから開始して LSB ビットで終了する順番、すなわち、CIN［31］、CIN［30］、...、CIN［1］、CIN［0］の順番で CIN［31：0］を計算します。

表 14-2：32 ビット多項式を使用する 32 ビット入力データのミラーリング・オプション

W16SWP	BYTMIRR	BITMIRR	Input Data DIN[31:0]	CRC Input Data (CIN[31:0])
0	0	0	DIN[31:0]	CIN[31:0] = DIN[31:0]
0	0	1	DIN[31:0]	CIN[31:0] = DIN[24:31], DIN[16:23], DIN[8:15], DIN[0:7]
0	1	0	DIN[31:0]	CIN[31:0] = DIN[23:16], DIN[31:24], DIN[7:0], DIN[15:8]
0	1	1	DIN[31:0]	CIN[31:0] = DIN[16:23], DIN[24:31], DIN[0:7], DIN[8:15]
1	0	0	DIN[31:0]	CIN[31:0] = DIN[15:0], DIN[31:16]
1	0	1	DIN[31:0]	CIN[31:0] = DIN[8:15], DIN[0:7], DIN[24:31], DIN[16:23]
1	1	0	DIN[31:0]	CIN[31:0] = DIN[7:0], DIN[15:8], DIN[23:16], DIN[31:24]
1	1	1	DIN[31:0]	CIN[31:0] = DIN[0:7], DIN[8:15], DIN[16:23], DIN[24:31]

ADuCM4050 の CRC レジスタの説明

CRC アクセラレータ（CRC）は以下のレジスタを備えています。

表 14-3：ADuCM4050 の CRC レジスター覧

レジスタ名	説明
CRC_CTL	CRC コントロール
CRC_IPBITS[n]	入力データ・ビット
CRC_IPBYTE	入力データ・バイト
CRC_IPDATA	入力データ・ワード
CRC_POLY	プログラマブル CRC 多項式
CRC_RESULT	CRC 結果

CRC コントロール

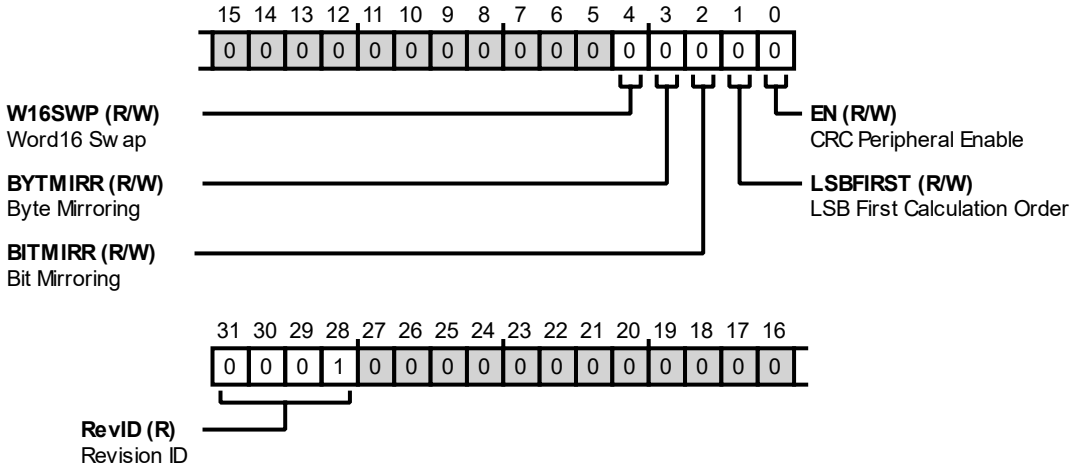


図 14-2：CRC_CTL のレジスタ図

表 14-4：CRC_CTL のレジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
31:28 (R/NW)	REVID	リビジョン ID。	
4 (R/W)	W16SWP	ワード 16 スワップ。 このビットは、32 ビット・ワード内で 16 ビット・ハーフワードをスワップします。	
		0	ワード 16 スワップを無効にします
		1	ワード 16 スワップを有効にします
3 (R/W)	BYTMIRR	バイト・ミラーリング。 このビットは、各 16 ビット・ハーフワード内で 8 ビット・バイトをスワップします。	
		0	バイト・ミラーリングを無効にします
		1	バイト・ミラーリングを有効にします
2 (R/W)	BITMIRR	ビット・ミラーリング。 このビットは、各バイト内でビットをスワップします。	
		0	ビット・ミラーリングを無効にします
		1	ビット・ミラーリングを有効にします
1 (R/W)	LSBFIRST	LSB ファーストの計算順序。	
		0	MSB ファーストの CRC 計算を行います
		1	LSB ファーストの CRC 計算を行います

表 14-4：CRC_CTL のレジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
0 (R/W)	EN	CRC ペリフェラル・イネーブル。	
		0	CRC ペリフェラルをディスエーブルにします
		1	CRC ペリフェラルをイネーブルにします

入力データ・ビット

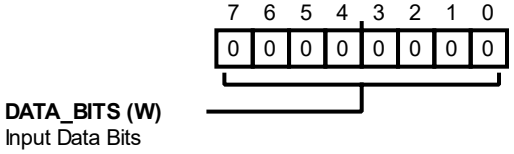


図 14-3：CRC_IPBITS［n］のレジスタ図

表 14-5：CRC_IPBITS［n］のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (RX/W)	DATA_BITS	入力データ・ビット。 このフィールドを使用して、入力データの 1～7 ビットから成るデータ・バイトの一部で CRC を計算します。CRC_IPBITS［n］.DATA_BITS の n ビットにバイトを書き込むことによって、入力データの n ビットの CRC を計算します。

入力データ・バイト

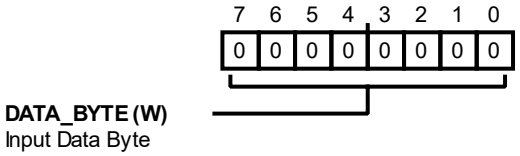


図 14-4：CRC_IPBYTE のレジスタ図

表 14-6：CRC_IPBYTE のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (RX/W)	DATA_BYTE	入力データ・バイト。 このフィールドに書き込まれたデータを使用して、データ・バイトの CRC を計算します。

入力データ・ワード

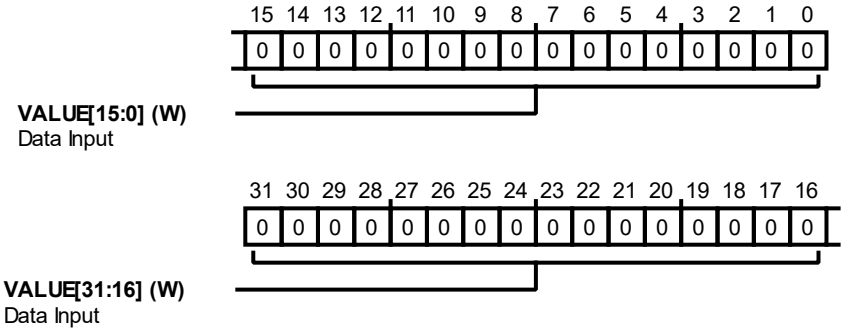


図 14-5：CRC_IPDATA のレジスタ図

表 14-7：CRC_IPDATA のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	VALUE	入力データ。

プログラマブル CRC 多項式

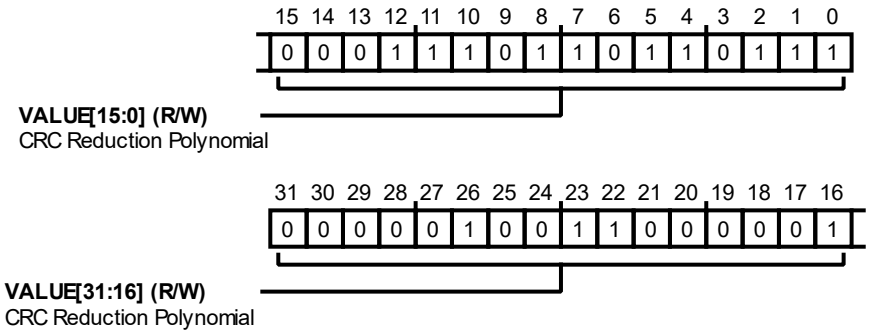


図 14-6：CRC_POLY のレジスタ図

表 14-8：CRC_POLY のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (R/W)	VALUE	CRC の既約多項式。

CRC 結果

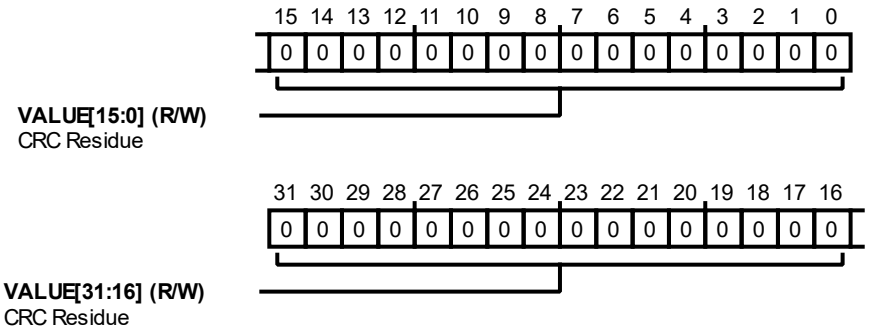


図 14-7：CRC_RESULT のレジスタ図

表 14-9：CRC_RESULT のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (R/W)	VALUE	CRC の剰余。

15 シリアル・ペリフェラル・インターフェース（SPI）

シリアル・ペリフェラル・インターフェース（SPI）は、業界標準の同期シリアル・リンクであり、複数の SPI 互換デバイスとの通信をサポートします。ベースラインの SPI ペリフェラルは、2 本のデータ・ピン、1 本のデバイス・セレクト・ピン、およびゲーティド・クロック・ピンで構成される同期 4 線式インターフェースです。2 本のデータ・ピンにより、他の SPI 互換デバイスとの間で全二重動作が可能です。フロー制御、高速モード、読出しコマンド・モード（半二重動作）などの拡張モードもサポートしています。更に、DMA モードでは、CPU へのアクセスを最小にしながら複数ワードを転送することができます。

SPI ポートは設定可能な様々なオプションを備えており、マスタ・モード、スレーブ・モード、マルチスレーブの環境で、他の SPI 互換デバイスとの円滑なハードウェア・インターフェースを提供します。SPI ペリフェラルには、プログラマブルなボー・レート、クロック位相、クロック極性が含まれています。このペリフェラルは、マスタ・デバイスまたはスレーブ・デバイスとして動作する他のいくつかのデバイスとインターフェースすることによって、マルチスレーブ環境で動作可能です。マルチスレーブ環境では、SPI ペリフェラルはオープンドレイン出力を使用してデータ・バス競合を回避します。フロー制御機能により、低速スレーブ・デバイスは、転送を柔軟に制御する SPI 対応ピンを備えることで、高速マスタ・デバイスとインターフェースできます。

SPI 機能

SPI モジュールは次の機能をサポートしています。

- シリアル・クロック位相モードとシリアル・クロック極性モード
- ループバック・モード
- 連続転送モード
- ワイヤード OR 出力モード
- 半二重動作の読出しコマンド・モード
- フロー制御
- 複数の CS ライン
- CS ソフトウェア・オーバーライド
- 3 ピン SPI のマスタまたはスレーブ・モードのサポート
- LSB ファースト転送のオプション

- 割込みモード：1、2、3、4、5、6、7、8 バイトの後で割込み。
- 8 バイト／ハーフワード・ディープ FIFO（Tx および Rx 用）

SPI 機能の説明

SPI 転送時、データの送信（シリアルでシフト・アウト）および受信（シリアルでシフト・イン）は同時に行われます。シリアル・クロック・ラインは、2 本のシリアル・データ・ラインでの情報のシフトとサンプリングを同期させます。データ転送時、一方の SPI システムはデータ・フローを制御するリンク・マスタとして機能し、他方のシステムはスレーブとして動作し、マスタによってデータのシフト・インおよびシフト・アウトが行われます。異なるデバイスが順番にマスタになることが可能で、1 つのマスタが同時に複数のスレーブにデータをシフト・インできます（ブロードキャスト・モード）。

ただし、出力を駆動してマスタにデータを書き戻すことができるのは常に 1 つのスレーブのみです。これはブロードキャスト・モードで実行され、マスタからのデータ受信には複数のスレーブを選択できますが、マスタにデータを返すことができるのは 1 つのスレーブのみです。

SPI ポートは、マスタまたはスレーブの動作用に構成可能で、MISO、MOSI、SCLK、CS の 4 本のピンで構成されています。SPI ペリフェラルをイネーブルする場合は、SPI 通信に使用する GPIO を予め SPI モードで設定する必要があることに注意してください。SPI ペリフェラルは、`SPI_CTL.SPIEN` ビットをセットしてイネーブルにします。SPI ペリフェラルを使用しない場合は、このビットをクリアしてオフにする必要があります。

SPI ブロック図

図は SPI ブロック図を示しています。

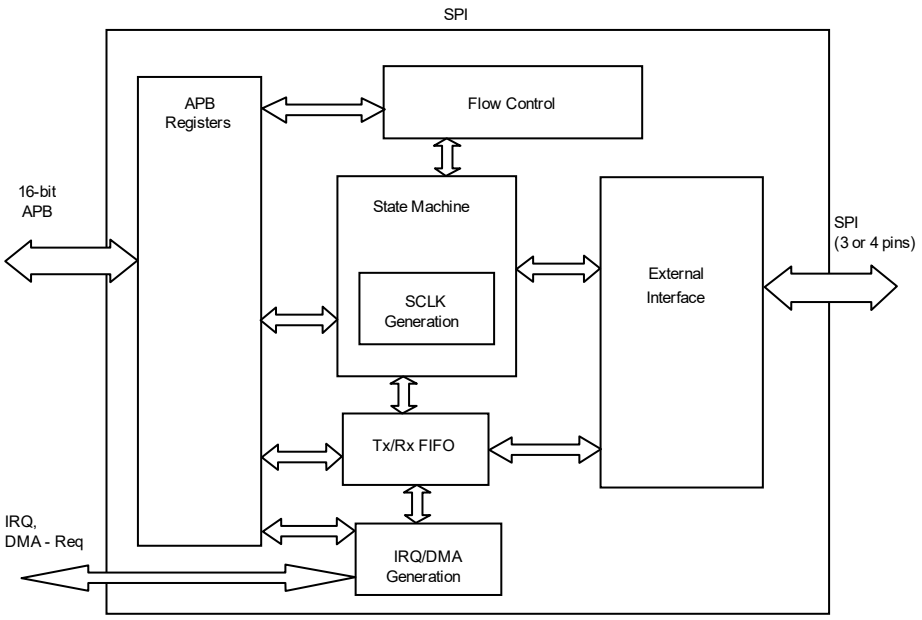


図 15-1：SPI ブロック図

MISO（マスタ・イン、スレーブ・アウト）ピン

MISO ピンは、マスタ・モードでは入力ライン、スレーブ・モードでは出力ラインとして構成されます。マスタでの MISO ライン（データ入力）は、スレーブ・デバイスの MISO ライン（データ出力）に接続する必要があります。データはバイト幅（8 ビット）のシリアル・データとして転送されます。

MOSI（マスタ・アウト、スレーブ・イン）ピン

MOSI ピンは、マスタ・モードでは出力ライン、スレーブ・モードでは入力ラインとして構成されます。マスタでの MOSI ライン（データ出力）は、スレーブ・デバイスの MOSI ライン（データ入力）に接続する必要があります。データはバイト幅（8 ビット）のシリアル・データとして転送されます。

SCLK（シリアル・クロック I/O）ピン

マスタ・シリアル・クロック（SCLK）は、MOSI SCLK の周期中に送受信されるデータを同期させます。したがって、1 バイトは 8 つの SCLK 周期後に送受信されます。SCLK ピンは、マスタ・モードでは出力、スレーブ・モードでは入力として構成されます。

マスタ・モードでは、クロックの極性と位相は SPI_CTL レジスタで制御され、ビット・レートは SPI_DIV レジスタで次のように定義されます。

$f_{シリアル・クロック} = CLK /（2（1 + SPIx\_DIV））$

CLK は、SPI0 と SPI1 では PCLK、SPIH では HCLK です。

スレーブ・モードでは、予定入力クロックの位相と極性を SPI_CTL レジスタに設定しなければなりません。

マスタ・モードとスレーブ・モードのどちらにおいても、データは SCLK 信号の片方のエッジで送信され、もう片方のエッジでサンプリングされます。したがって、マスタとスレーブのデバイスで極性と位相を同じ設定にします。

注：タイミング仕様については、データシートを参照してください。

チップ・セレクト（CS I/O）ピン

SPI スレーブ・モードでは、アクティブ・ローの入力信号である CS のアサートによって転送が開始されます。その後、CS のアサート解除によって転送が終了するまで、SPI ポートで 8 ビットのデータが送受信されます。スレーブ・モードでは、CS は常に入力です。

SPI マスタ・モードでは、CS はアクティブ・ローの出力信号です。これは、転送の開始時に自動的にアサートされ、完了時に自動的にアサート解除されます。

マルチスレーブ環境では、最大 4 つの異なるスレーブがサポートされます。SPI マスタは、SPI_CS_CTL レジスタを使用して最大 4 本の CS ライン（CS0、CS1、CS2、CS3）を駆動するように設定できます。ブロードキャスト・アクセスでは、複数の CS ラインを同時に駆動できます。アクティブな CS ラインを 0 または 1 にソフトウェア駆動するためのオーバーライド・フィールドもありますが、これはいくつかの特殊な用途で必要になります。他のすべての SPI ピンはスレーブ間で共有されます。

SPI 動作モード

SPI は以下に示す機能を備えています。

ワイヤード OR モード（WOM）

マルチスレーブ・システムで SPI を使用する際の競合を防ぐために、MOSI および MISO のデータ出力ピンは、オープン・サーキット・ドライバとして動作するように構成できます。この機能を選択するときは、外付けのプルアップ抵抗が必要です。コントロール・レジスタの SPI_CTL.WOM ビットは、データ・ラインのパッド・イネーブル出力を制御します。

一般的注意事項

1. SPI モジュールは PCLK で動作します。したがって、PCLK が停止している場合、このブロックは機能しません。
2. SPI の設定を変更する必要がある場合は常に、CS がアサート解除されていることを確認してください。これにより、SPI 転送の突然の中断を回避できます。SPI 転送の実行中に構成を変更すると、ペリフェラルの動作は不確定になります。
3. 構成を変更する際は、必ず SPI_CTL レジスタと SPI_IEN レジスタを変更した後で SPI_CNT レジスタを設定してください。そうしないと、構成を変更する前に転送が開始されることがあります。

パワーダウン・モード

マスタ・モードでは、パワーダウン・モードに入る前に、適切な割込み／ステータス・ビットをチェックして転送が完了していることを確認する必要があります。その後、SPI_CTL.SPIEN ビットをクリアすることで SPI ブロックをディスエーブルする必要があります。そうした場合にのみ、パワーダウンへのエントリはクリーンになります。

スレーブ・モードでは、どちらの動作モード（割込み駆動または DMA）でも SPI が通信していないことを確認するために、SPI_STAT.CS ビットを使用して CS ラインのレベルをチェックする必要があります。CS ラインがハイの場合にのみ、SPI ブロックをディスエーブルする必要があります。

パワーダウン中は、以下のフィールドが保持されます。

- SPI_CTL.SPIEN を除く、SPI_CTL レジスタのすべてのビット・フィールド。パワーアップ時には、SPI_CTL.SPIEN ビットがリセットされます。これにより、ウェイクアップ時に新たにペリフェラルを開始できます。
- SPI_IEN.IRQMODE ビット
- SPI_DIV.VALUE ビット
- SPI_RD_CTL.THREEPIN ビット
- SPI_FLOW_CTL.RDYPOL ビット

他のフィールドはいずれも保持されず、パワーアップ時にリセットされます。パワーダウン・モードの終了時には、ソフトウェアにより、保持されていないすべてのレジスタを必要に応じて再設定する必要があります。SPI ブロックは、SPI_CTL.SPIEN ビットをセットすることにより再度イネーブルする必要があります。

SPI スレーブとのインターフェース

SPI 転送は通常は全二重ですが、多くの場合、スレーブはコマンド、アドレス、およびデータ読出し／書込みからなるプロトコルで動作します。書込みコマンドは常に単方向です。ただし、読出しコマンドでは、1 つの CS フレーム内で送信後に受信を行うことが必要です。プロトコルの例を以下に示します。

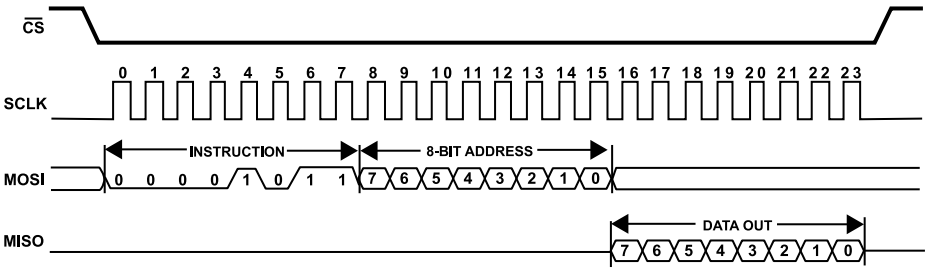


図 15-2：SPI レジスタ読出し

上記のような半二重動作に似た転送をサポートするには、SPI_RD_CTL.CMDEN ビットをセットする必要があります。また、SPI_RD_CTL.TXBYTES フィールドに送信するバイト数（上記の例では 1）を設定する必要があります。受信するバイト数（送信完了後）は、SPI_CNT.VALUE ビットで指定します。この場合は、SPI_CNT.VALUE は 1 でなければなりません。

SPI_RD_CTL.OVERLAP ビットは、コマンドおよびアドレスのバイト送信中に受信したバイトを保存するか無視するかを指定します。このビットをセットした場合、SPI_CNT.VALUE ビットは、フレーム全体の合計バイト数、すなわち SPI_CNT.VALUE = 送信バイト数 + 受信バイト数を示します。

SPI 読出しの例を以下にいくつか示します。

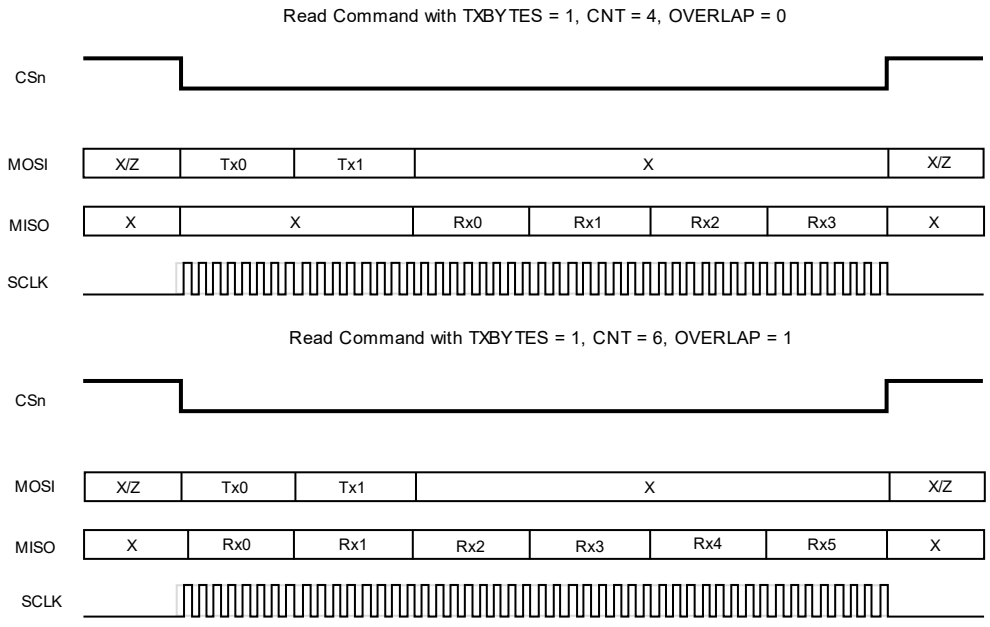


図 15-3：SPI 読出しの例

読出しコマンド・モードの擬似コードの例

```
SPI0_DIV = 0x0001              //SPI シリアル・クロック周波数 = PCLK 周波数の¼
SPI0_CTL = 0x0883             //SPI をマスタ・モード、ZEN = 1、
                             //連続モード、TIM = 0 に設定
SPI0_CNT = 0x0040           //64 バイトを受信
SPI0_DMA = 0x0005          //DMA モードと Rx-DMA リクエストを有効化
SPI0_RD_CTL = 0x000D     //TXBYTES = 3、CMDEN = 1、2 回の 16 ビット書込みで
                        //4 Tx バイトを書込み（DMA モード）
SPI0_TX = 0xB6A5
SPI0_TX = 0xD8C7
read_data = SPI0_RX      //受信 FIFO のダミー読出しを行って
                        //SPI 転送を開始
```

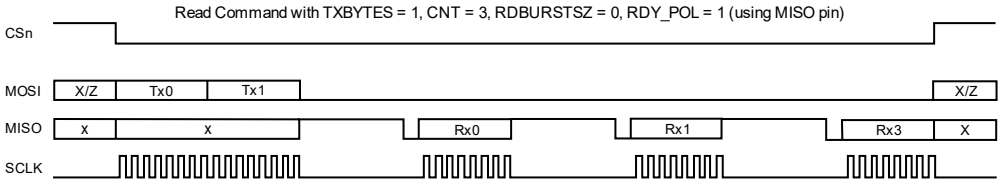
この場合、TXBYTES が 3 なので、4 バイトのみが送信されます。ただし、受信する次の 64 バイトに対しては、MOSI に 0 が送信されます。合計で、オーバーラップしないモードでは 4 バイトが送信され、64 バイトが受信されます。

フロー制御

一部のコンバータでは、フロー制御メカニズムを使用して必要なデータ／サンプル・レートにマッチさせています。SPI マスタは、以下のタイプのフロー制御をサポートしています。

- データの読出し中に待機状態を設定するために、シリアル・クロック・レートでクロック制御された 16 ビット・タイマーを使用。マスタは、タイマーが終了するまで待機してから、SPI_FLOW_CTL.RDBURSTSZ + 1 のバイト数を読み出します。待機状態に戻り、タイマーを再起動します。これは、SPI_CNT.VALUE バイト数を受信するまで継続します。
- GPIO の 1 つに接続された個別の RDY ピンを使用。マスタは、このピンがアクティブなレベルになるまで待ちます。遷移を検出すると、SPI_FLOW_CTL.RDBURSTSZ + 1 バイト数を読み出してから、次のアクティブ・レベルが検出されるまで待機状態に戻ります。これは、SPI_CNT.VALUE バイト数を受信するまで継続します。
- MISO ピンの使用。このモードでは、マスタは MISO ピンがアクティブ・レベルになるのを待ちます。遷移を検出すると、SPI_FLOW_CTL.RDBURSTSZ + 1 バイト数を読み出してから、次のアクティブ・レベルが検出されるまで待機状態に戻ります。これは、SPI_CNT.VALUE バイト数を受信するまで継続します。

上記のすべての場合において、SPI_CNT.VALUE は SPI_FLOW_CTL.RDBURSTSZ + 1 の整数倍でなければなりません。フロー制御転送の例を以下に示します。



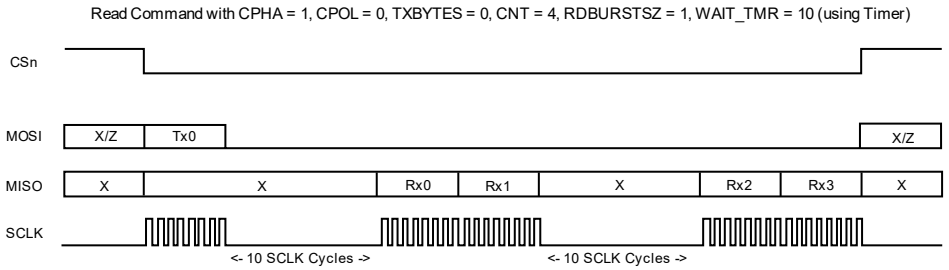


図 15-4：フロー制御転送の例

注：フロー制御の SCLK 出力が停止したときや FIFO にデータ／スペースがなくなったときには、常に最後の SCLK エッジがサンプリング・エッジになります。これは、停止期間の前に駆動エッジが生じないようにするためです。これにより、停止期間の後でスレーブに SCLK 駆動エッジが与えられます。

SCLK 信号は SPI_CTL.CPOL = 0 のときにローでアイドル状態になり、SPI_CTL.CPOL = 1 のときにハイでアイドル状態になりますが、SPI_CTL.CPHA ビットによってサンプリングと駆動エッジのシーケンスが決定されます。SPI_CTL.CPHA = 1 の場合、SCLK 信号はアイドル・レベルと同じレベルで停止します。SPI_CTL.CPHA = 0 の場合、SCLK はアイドル・レベルと逆のレベルで停止します。転送終了時（CS がアサート解除されたとき）、SCLK は常に SPI_CTL.CPOL に従ってアイドリング状態になります。**フロー制御**の表でも同様に説明されています。

表 15-1：フロー制御

CPHA	CPOL	SCLK Idle level	SCLK Stalled level
0	0	0	1
0	1	1	0
1	0	0	0
1	1	1	1

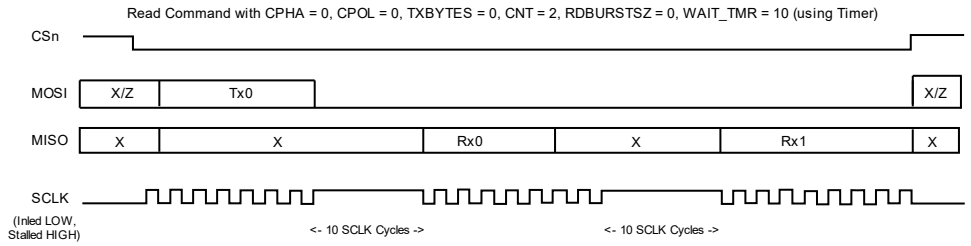


図 15-5：サンプル転送

3 ピン・モード

SPI は、データの送受信を単一の双方向ラインで行う 3 ピン・モードで使用できます。SPI マスタは、読出しコマンド・モードでこれをサポートしています。このコマンド・モードでは、MOSI ラインが「コマンド+アドレス」バイトの送信に使用され、同じラインが読出しデータの受信に使用されます。これをイネーブルするには、SPI_RD_CTL レジスタの SPI_RD_CTL.THREEPIN フィールドと SPI_RD_CTL.CMDEN フィールドをセットする必要があります。

本来は、送信フェーズの最後のサンプリング・エッジから受信フェーズの最初の駆動エッジまでが SCLK サイクルの半分となります。しかし、スレーブがより長いターンアラウンド・タイムを必要とする場合は、タイマー・ベースのフロー制御を有効にし、必要に応じて SPI_WAIT_TMR.VALUE を設定する必要があります。3 ピン SPI 転送の例を以下に示します。

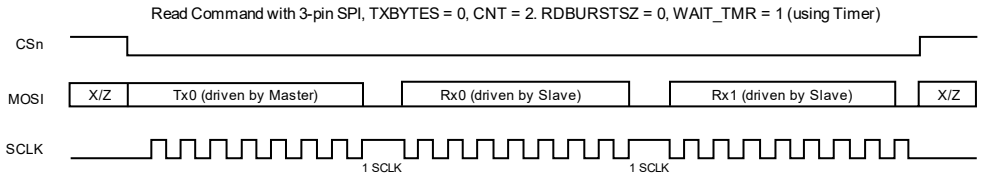


図 15-6：3 ピン SPI 転送

SPI データ転送

マスタ・モードでは、転送および割込みモード・ビット（SPI_CTL.TIM）によって、SPI シリアル転送の開始方法が決まります。SPI_CTL.TIM ビットをセットすると、送信 FIFO への書込み後にシリアル転送が開始されます。SPI_CTL.TIM ビットをクリアすると、受信 FIFO の読出し後にシリアル転送が開始されます。読出しは、SPI インターフェースがアイドル状態の間に行わなければなりません。アクティブな転送中に読出しを行うと、次の転送は開始されません。

マスタ・モード・イネーブル（SPI_CTL.MASEN）ビットと SPI_CTL.TIM ビットの設定によらず、SPI はデータの受信と送信を同時に行います（SPI_RD_CTL.CMDEN ビットが 0 の場合）。したがって、SPI はデータ送信中にデータの受信も行い、受信 FIFO を満たしていきます。受信 FIFO からデータが読み出されないと、FIFO がオーバーフローし始めた時点でオーバーフロー割込みが発生します。受信データの読出しやオーバーフロー割込みの受信が不要の場合、受信 FIFO フラッシュ・イネーブル（SPI_CTL.RFLUSH）ビットをセットすると、受信データは受信 FIFO に保存されません。あるいは（オーバーフロー状態は考慮せずに）受信データの読出しのみが必要な場合は、SPI_IEN.RXOVR ビットをクリアすることで、この割込みをディスエーブルできます。同様に、データを受信するだけで送信 FIFO にデータを書き込む必要のない場合は、送信 FIFO フラッシュ・イネーブル（SPI_CTL.TFLUSH）ビットをセットすると、送信 FIFO からのアンダーフロー割込みを回避できます。また、FIFO データを送信する必要はあっても、アンダーフロー割込みを望まない場合は、SPI_IEN.TXUNDR ビットをクリアする必要があります。

送信起動転送

送信 FIFO への書込みによって転送を起動する場合、最初のバイトが FIFO に書き込まれると直ちに SPI は送信します。最初のバイトの SPI 転送は即座に実行されます。

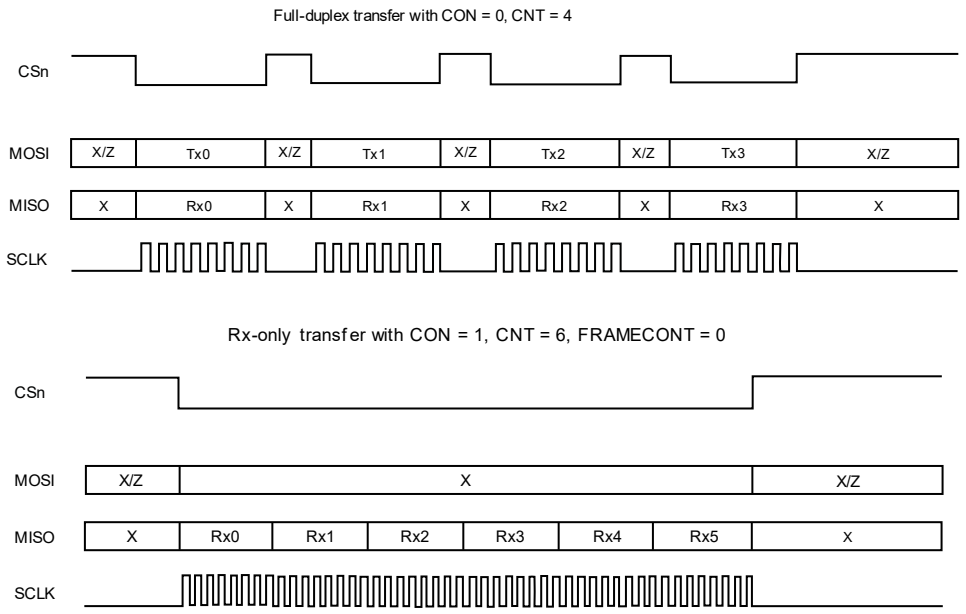
連続転送イネーブル・ビット（SPI_CTL.CON）をセットすると、転送は完了するまで継続されます。この完了とは、SPI_CNT.VALUE バイト数の最後（SPI_CNT.VALUE > 0 の場合）または送信 FIFO に有効なデータがない場合（SPI_CNT.VALUE = 0 の場合）のいずれかです。チップ・セレクトは、転送中はアサートされたままになります。SPI_CNT.FRAMECONT をクリアし、かつ SPI_CNT.VALUE > 0 の場合、すべての SPI_CNT.VALUE バイトを転送すると、転送が停止します。SPI_CNT.FRAMECONT をセットした場合、SPI_CNT.VALUE バイトごとに新しいフレームが開始されます。したがって、SPI_CNT.VALUE バイトの倍数が転送されます。FIFO にデータ／スペースがない場合、転送は使用可能になるまで停止します。逆に、FIFO に有効なデータがある場合は転送が継続されます。

SPI_CTL.CON をクリアした場合、各転送は単一の 8 ビット・シリアル転送で構成されます。有効なデータが送信 FIFO に存在する場合、チップ・セレクトがアサート解除される停止期間後に、新しい転送が開始されます。

受信起動転送

受信 FIFO の読出しによって起動される転送は、FIFO に受信するバイト数に依存します。SPI_IEN.IRQMODE を 7 に設定し、受信 FIFO への読出しが発生すると、SPI マスタは 8 バイトの転送を開始します。連続モードを設定している場合（SPI_CTL.CON）、バイト間でチップ・セレクトがアサート解除されずに 8 バイトが連続して転送されます。連続モードを設定していない場合は、8 バイトの各転送の間に停止期間が置かれ、その際チップ・セレクトがアサート解除されます。ただし、連続モードでは、SPI_CNT.VALUE > 0 の場合、CS はフレーム期間全体にわたってアサートされます。SPI は、FIFO 領域が利用可能になるまで SCL をクロック供給しないことで、停止期間を設けます。

SPI_IEN.IRQMODE を 6 に設定した場合、上記と同様に受信 FIFO の読出しにより 7 バイト転送が開始されます。SPI_IEN.IRQMODE を 1 に設定した場合、受信 FIFO の読出しにより 2 バイト転送が開始されます。SPI_IEN.IRQMODE を 0 に設定して FIFO を読み出すと、1 バイト転送が開始されます。SPI がデータを受信している間に受信 FIFO を読み出すと、現在の転送が完了した後、次の転送は開始されません。連続モードでは、SPI_CNT.VALUE > 0 かつ SPI_CNT.FRAMECONT = 1 の場合、（受信した最後のバイト・セットを取得するため）SPI フレームの最後で受信 FIFO を読み出すと、常に新しい SPI フレームが開始されます。したがって、任意のフレームで SPI 転送を停止するには、SPI_CNT.FRAMECONT ビットをクリアしてから、受信した最後のバイト・セットを読み出す必要があります。



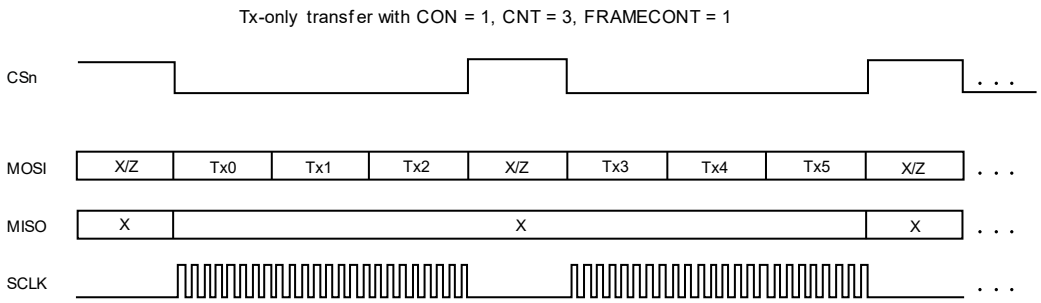


図 15-7：SPI 転送

スレーブ・モードでの転送

スレーブ・モードでは、デバイスのチップ・セレクトのアサートによって転送が開始されます。

マスタは最大 4 本の CS 出力ラインをサポートできますが、スレーブ・モードでは CS 入力は 1 本のみ（CS0）使用されます。

スレーブとしてのデバイスは、チップ・セレクトのアサート解除によって転送が終了するまで、8 ビット・データを送受信します。

以下の SPI 転送プロトコル図は、SPI のデータ転送プロトコルと、T1、T2、T3 に示すように、制御レジスタ内の SPI_CTL.CPHA ビットおよび SPI_CTL.CPOL ビットがこのプロトコルに及ぼす影響を示しています（**SPI 転送プロトコル CPHA = 0** および **SPI 転送プロトコル CPHA = 1** を参照）。

注：チップ・セレクトはグラウンドに接続しないでください。

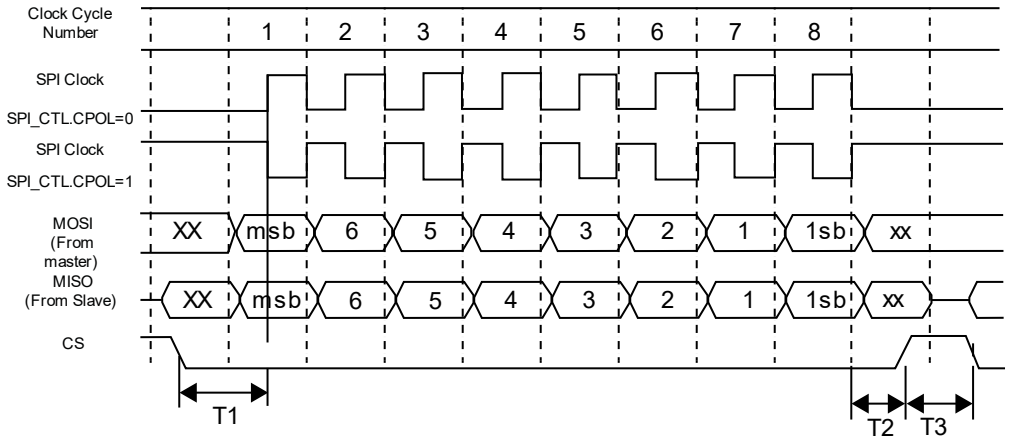


図 15-8：SPI 転送プロトコル CPHA = 0

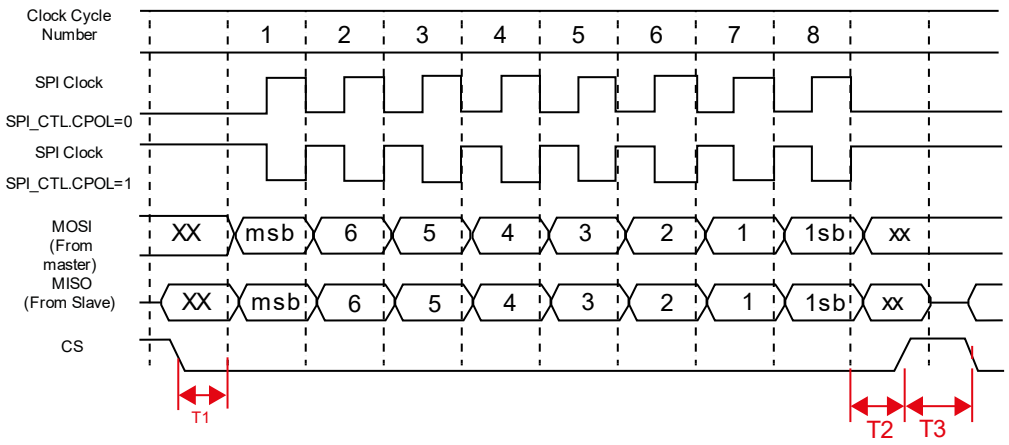


図 15-9：SPI 転送プロトコル CPHA = 1

SPI データのアンダーフローとオーバーフロー

送信ゼロ・アンダーフロー・モード・ビット（SPI_CTL.ZEN）をクリアした場合、FIFO に有効なデータがない状態で転送が開始されると、最後の無効バイトがシフト・アウトされます。SPI_CTL.ZEN をセットした場合、FIFO に有効なデータがない状態で転送が開始されると、ゼロが送信されます。

受信オーバーフロー・オーバライト・イネーブル・ビット（SPI_CTL.RXOF）をセットすると、FIFO に空きがない場合、受信 FIFO 内の有効データは受信した新しいシリアル・バイトによって上書きされます。SPI_CTL.RXOF をクリアすると、受信 FIFO に空きがない場合、受信した新しいシリアル・バイトは破棄されます。

受信 FIFO で有効なデータが上書きされる場合、最も古いバイトから最初に上書きされ、次に古いバイトが続きます。

SPI の割込みと例外

SPI ごとに 1 本の割込みラインと 11 の割込み源があります。

SPI_STAT.IRQ ビットは、割込みラインの状態を反映します。

SPI_STAT [15:12] ビットと SPI_STAT [7:1] ビットは、11 の割込み源の状態を反映します。

SPI は、送信割込みリクエスト（TIRQ）または受信割込みリクエスト（RIRQ）を生成します。両方の割込みを同時にイネーブルすることはできません。SPI_CTL.TIM ビットを使用して、適切な割込みをイネーブルします。SPI_CTL.TIM = 1 の場合、TIRQ がイネーブルされます。SPI_CTL.TIM = 0 の場合、RIRQ がイネーブルされます。

すべての割込みはスティッキーで、SPI_STAT レジスタの適切な割込みビットを 1 にセットしている場合にのみクリアされます。デバイスからの割込みラインは、すべての割込み源がクリアされた場合にのみクリアされます。

送信割込み

SPI_CTL.TIM ビットをセットした場合、送信 FIFO ステータスによって割込みが発生します。SPI_IEN.IRQMODE ビットによって割込みの発生タイミングを制御します。SPI_IEN.IRQMODE ビットの設定によって、以下のように割込みが発生します。

000：1 バイトが送信されるたびに割込みが発生します。FIFO からバイトが読み出され、シフト・レジスタに書き込まれると、割込みが発生します。

001：2 バイトが送信されるたびに割込みが発生します。

...

110：7 バイトが送信されるたびに割込みが発生します。

111：8 バイトが送信されるたびに割込みが発生します。

割込みは、送信されたバイト数に応じて生成され、FIFO のバイト数には依存しません。これは受信割込みとは異なります。受信割込みは受信 FIFO 内のバイト数に依存し、受信バイト数には依存しません。

この割込みの状態は、TXIRQ ステータス・ビットを読み出すことで知ることができます。送信 FIFO フラッシュ・イネーブル（TFLUSH）がハイの場合、割込みはディスエーブルされます。

注：SPI_CTL レジスタへの書込みによって、送信バイト・カウンタがゼロにリセットされます。例えば、SPI_IEN.IRQMODE ビットを 0x3 に設定した場合、3 バイトの送信後に SPI_CTL レジスタが再初期化されるため、次の 4 バイトが送信されるまで送信割込みは発生しません。

受信割込み

SPI_CTL.TIM をクリアすると、受信 FIFO ステータスによって割込みが発生します。この場合も、SPI_IEN.IRQMODE によって割込みが発生するタイミングを制御します。この割込みの状態は、SPI_STAT.RXIRQ ステータス・ビットを読み出すことで知ることができます。

割込みは、データが FIFO に書き込まれたときにのみ発生します。例えば、SPI_IEN.IRQMODE を 0x0 に設定している場合、最初のバイトが受信された後に割込みが発生します。1 を書き込むことによって割込みがクリアされ、受信 FIFO からデータ・バイトが読み出されなかった場合、再度割込みが発生することはありません。次のバイトが FIFO に受信されると、次の割込みが発生します。

この割込みは、SPI がデータを受信しているときの FIFO の有効バイト数に依存します。SPI で受信されたバイト数には依存しません。

SPI_CTL.RFLUSH をハイに維持すると、割込みはディスエーブルされます。

アンダーフロー／オーバーフロー割込み

送信 FIFO にデータがない状態で転送が開始されると、ステータス・レジスタの SPI_STAT.TXUNDR ビットがアンダーフロー状態を示すようにセットされます。SPI_IEN.TXUNDR をセットしている場合は、これにより割込みが発生します。この割込みは、どの SPI_CTL.TIM ビットをセットしているかにかかわらず発生します。SPI_CTL.TFLUSH ビットをセットしている場合、この割込みはディスエーブルされます。

受信 FIFO が既にフルになっているときにデータを受信すると、ステータス・レジスタの SPI_STAT.RXOVR ビットがハイになり、オーバーフロー状態を示します。SPI_IEN.RXOVR をセットしている場合は、これにより割込みが発生します。この割込みは、SPI_CTL.TIM ビットの設定に無関係に発生します。SPI_CTL.RFLUSH ビットをセットしている場合、この割込みはディスエーブルされます。

該当するフラッシュ・ビットがアサートされているか、SPI がディスエーブルされている場合、受信割込みと送信割込みはクリアされます。それ以外の場合は、SPI が再構成されても割込みはアクティブを維持します。

SPI プログラミング・モデル

次のセクションでは、SPI DMA の詳細について説明します。

SPI DMA

2 つの DMA チャンネルは SPI の送受信専用です。2 つの SPI DMA チャンネルは、DMA コントローラで設定する必要があります。

SPI_DMA レジスタの受信用または送信用の DMA リクエスト・ビットをセットすることにより、同時に 1 チャンネルまたは 2 チャンネルの DMA リクエストを有効にできます。DMA 送信リクエスト（SPI_DMA.TXEN）のみを有効にした場合、ユーザ・コードによって受信データを読み出さない限り、SPI 転送中に Rx FIFO がオーバーフローし、オーバーフロー割込みが発生します。オーバーフロー割込みが発生しないようにするには、Rx FIFO フラッシュ・ビット（SPI_CTL.RFLUSH）をセットするか、SPI_IEN.RXOVR ビットをクリアするか、コアの NVIC で SPI 割込みをディスエーブルする必要があります。DMA 受信リクエスト（SPI_DMA.RXEN）のみを有効にした場合、Tx FIFO はアンダーフローします。この場合も、アンダーフロー割込みを回避するには、SPI_IEN.TXUNDR ビットをクリアするか、NVIC で SPI 割込みをディスエーブルする必要があります。

注：Tx FIFO の状態を確認してください。ダミー読出しを発行する前は、これが空の状態にならないようにして、意図しない Tx アンダーフロー割込みが発生しないようにしてください。

DMA が使用されているときは、SPI の Tx および Rx 割込みは生成されません。SPI_IEN.IRQMODE は送信モードでは使用せず、受信モードでは 3'b000 に設定する必要があります。

DMA ビット（SPI_DMA.TXEN）は、DMA 転送の開始を制御します。DMA リクエストは、SPI_DMA.EN = 1 の場合にのみ生成されます。DMA 転送の最後、つまり DMA SPI 転送割込みを受信したとき、μDMA コントローラに DMA リクエストが追加されないように、このビットをクリアする必要があります。Tx FIFO にまだ存在しているデータは、Tx モードの場合に送信されます。

すべての DMA データ転送は 16 ビット転送であり、それに従って DMA を設定する必要があります。例えば、16 バイトのデータを SPI を介して転送する場合、ハーフワード（16 ビット）転送を 8 回実行するように DMA を設定する必要があります。17 バイトを転送する場合は 9 回のハーフワード転送が必要であり、追加のバイトは廃棄されます。DMA 転送をバイト幅転送として設定していると、データ・エラーが発生します。

DMA モードでは、Tx/Rx FIFO は 2 バイト幅です。ビット［7:0］が最初に SPI によってアクセスされ、次にビット［15：8］がアクセスされます。これは、カウントまたは SPI_CTL.LSB の設定とは無関係です。例えば、SPI_CNT.VALUE = 3 の場合、送受信の順序は次のようになります（Byte-3 は無視されます）。

Byte-1	Byte-0
Byte-3	Byte-2

注：SPI_CTL.LSB ビットは DMA モードでの FIFO のアクセス順序に影響しません。SPI で各バイトがどのように転送されるかにのみ影響します。

DMA マスタ送信の構成

DMA SPI Tx チャンネルを設定する必要があります。また、DMA Tx マスタ割込みをイネーブルするように NVIC を設定する必要があります。

SPI ブロックは以下のように設定する必要があります。

```
SPI_DIV = SPI_SERIAL_FREQ;       //シリアル・クロック周波数を設定。
SPI_CTL = 0x1043;                //SPI をマスタ・モードおよび送信モードでイネーブル、
                                 //Rx FIFO フラッシュをイネーブル。
SPI_CNT.VALUE = NUM_BYTES_TO_TRANSFER      //転送するバイト数を設定。
SPI_DMA = 0x1;                   //（オプション）16 ビットのコア・データ書込み
                                 //を受け付けるように FIFO をイネーブル。
SPI_TX = 0xXXXX;                 //（オプション）FIFO にプリロードするために、
                                 //最大 4 回の 16 ビット・コア書込みを実行できる。
SPI_DMA = 0x3;                   //DMA モードを有効にし、Tx DMA リクエストを有効にする。
```

すべての DMA 転送は 16 ビット転送となります。DMA バッファに存在するすべてのデータが送信されると、DMA は割込みを生成します。ユーザ・コードでは、DMA リクエストを無効にする必要があります。Tx FIFO に空き領域ができるたびに DMA リクエストが生成されるため、Tx FIFO は常にデータでフルに保たれます。

DMA マスタ受信の設定

SPI_CNT レジスタには、SPI マスタが必要とする受信バイト数を設定します。リクエストしたバイト数を受信すると、それ以上の転送は開始されません。DMA マスタ受信転送を開始するには、ダミー読出しをユーザ・コードで実行する必要があります。このダミー読出しは SPI_CNT の数に加えないでください。

受信したバイト数のカウンタのリセットは、SPI_CNL レジスタの SPI_CTL.SPIEN ビットを使用して SPI をディスエーブルしている場合、または SPI_CNT レジスタがユーザ・コードによって変更された場合に行われます。

SPI DMA マスタ受信を実行するには、以下のようにします。

DMA SPI Rx チャンネルを設定する必要があります。コアの NVIC は、DMA Rx マスタ割込みをイネーブルするように構成する必要があります。

SPI ブロックは以下のように設定する必要があります。

```
SPI_DIV = SPI_SERIAL_FREQ;     //シリアル・クロック周波数を設定。
SPI_CTL = 0x2003;              //SPI をマスタ・モードおよび
                               //受信モードでイネーブル、1 バイト転送に設定。
SPI_DMA = 0x5;                 //DMA モードを有効にし、Rx DMA リクエストを有効にする。
SPI_CNT.VALUE = XXX;           //受信するバイト数。
A = SPI_RX;                    //ダミー読出し。
```

適切な数のクロック・サイクルが生成されると、DMA 転送が停止します。すべての DMA データ転送は 16 ビット転送であり、それに従って DMA を設定する必要があります。例えば、16 バイトのデータを SPI を介して受信する場合、16 ビット転送を 8 回実行するように DMA を設定する必要があります。

17 バイトを受信する場合、16 ビット転送が 9 回必要です。最後の DMA 転送に追加バイトが付加されます。DMA 転送をバイト幅転送として設定している場合、データ・エラーが発生します。

注：転送が完了したときに DMA 割込みを生成するためには、DMA バッファは SPI_CNT.VALUE と同じサイズ（SPI_CNT.VALUE が奇数の場合はこれに 1 を加えたもの）でなければなりません。

DMA スレーブ送信の構成

DMA SPI Tx チャンネルを設定する必要があります。また、DMA Tx 割込みをイネーブルするように NVIC を設定する必要があります。

SPI ブロックは以下のように設定する必要があります。

```
SPI_CTL = 0x1241;                                    //スレーブ・モードと転送モードで SPI をイネーブルし、
                                                     //Rx FIFO、フラッシュをイネーブル。
SPI_CNT.VALUE = NUM_BYTES_TO_TRANSFER;       //送信するバイト数を設定。
SPI_TX = 0xXXXX;                 //（オプション）FIFO にプリロードするために、16 ビット書込みを最大 4 回実行できる。
                                 //これは、DMA の N-1 を設定する際にも考慮する必要があります。
SPI_DMA = 0x3;                 //DMA モードを有効にし、Tx DMA リクエストを有効にする。
```

すべての SPI DMA 転送は 16 ビット転送となります。DMA バッファに存在するすべてのデータが送信されると、DMA は割込みを生成します。ユーザ・コードでは、DMA リクエストを無効にする必要があります。Tx FIFO に空き領域ができるたびに DMA リクエストが生成されるため、Tx FIFO は常にデータでフルに保たれます。

DMA スレーブ受信の設定

SPI_CNT レジスタには、必要とする受信バイト数を設定します。

受信したバイト数のカウンタのリセットは、SPI_CTL レジスタの SPI_CTL.SPIEN ビットを使用して SPI がディスエーブルされている場合、または SPI_CNT レジスタがユーザ・コードによって変更された場合に行われます。

SPI DMA マスタ受信を実行するには、以下のようにします。

DMA SPI Rx チャンネルを設定する必要があります。コアの NVIC は、DMA Rx マスタ割込みをイネーブルするように構成する必要があります。

SPI ブロックは以下のように設定する必要があります。

```
SPI_CTL = 0x2001;                 //SPI をスレーブ・モードおよび受信モードでイネーブル、1 バイト転送に設定。
SPI_DMA = 0x5;                  //DMA モードを有効にし、Rx DMA リクエストを有効にする。
SPI_CNT.VALUE = XXX;          //受信するバイト数。
A = SPI_RX;                   //ダミー読出し。
```

適切な数のハーフワードを受信すると、DMA 転送は停止します。すべての DMA データ転送は 16 ビット転送であり、それに従って DMA を設定する必要があります。例えば、16 バイトのデータを SPI を介して受信する場合、16 ビット転送を 8 回実行するように DMA を設定する必要があります。17 バイトを受信する場合、16 ビット転送が 9 回必要で

す。最後の DMA 転送に追加バイトが付加されます。DMA 転送をバイト幅転送として設定している場合、データ・エラーが発生します。

注：転送が完了したときに DMA 割込みを生成するためには、DMA バッファは SPI_CNT.VALUE と同じサイズ（SPI_CNT.VALUE が奇数の場合はこれに 1 を加えたもの）でなければなりません。

ADuCM4050 SPI レジスタの説明

シリアル・ペリフェラル・インターフェース（SPI）には以下のレジスタがあります。

表 15-2：ADuCM4050 SPI レジスター一覧

レジスタ名	説明
SPI_CNT	転送バイト数
SPI_CS_CTL	マルチスレーブ接続のチップ・セレクト制御
SPI_CS_OVERRIDE	チップ・セレクト・オーバーライド
SPI_CTL	SPI 設定
SPI_DIV	SPI ボー・レート選択
SPI_DMA	SPI DMA イネーブル
SPI_FIFO_STAT	FIFO ステータス
SPI_FLOW_CTL	フロー制御
SPI_IEN	SPI 割込みイネーブル
SPI_RD_CTL	読出し制御
SPI_RX	受信
SPI_STAT	ステータス
SPI_TX	送信
SPI_WAIT_TMR	フロー制御の待機タイマー

転送バイト数

このレジスタはマスタ・モードでのみ使用します。

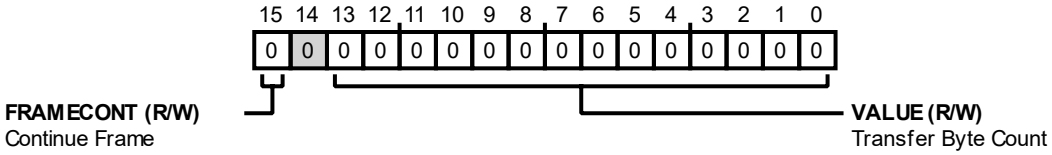


図 15-10：SPI_CNT レジスタ図

表 15-3：SPI_CNT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15 (R/W)	FRAMECONT	フレーム継続。 このビットは、SPI_CTL.CON および SPI_CNT.VALUE フィールドと共に使用する必要があります。SPI データのフレーム制御に使用します。このビットをクリアした場合、SPI マスタは SPI_CNT.VALUE バイトの 1 フレームのみを転送します。セットした場合、SPI マスタは SPI_CNT.VALUE バイトのフレーム単位でデータを転送します。 注：SPI_CNT.VALUE = 0 の場合、Tx/Rx FIFO がレディ状態になっている限り SPI マスタは転送を続けるため、このフィールドは無効です。SPI_CTL.CON = 0 の場合、他の制御フィールドに関係なく、すべての SPI フレームが 1 バイト幅となるため、このフィールドは無効です。	
		0	SPI_CNT.VALUE > 0 の場合、SPI_CNT.VALUE バイト数の後で SPI 転送を停止させます。
		1	Tx/Rx FIFO がレディ状態になっている限り、SPI 転送を継続させます。
13:0 (R/W)	VALUE	転送バイト数。 このフィールドは、転送するバイト数を指定します。これは、受信と送信の両方の転送タイプで使用されます。この値を設定することにより、マスタ・モード転送を適切な時間に終了させたり、奇数転送カウントと一致させるために 16 ビットの SPI_DMA 転送でバイトを付加または破棄したりすることができます。	

マルチスレーブ接続のチップ・セレクト制御

このレジスタはマスタ・モードでのみ使用します。

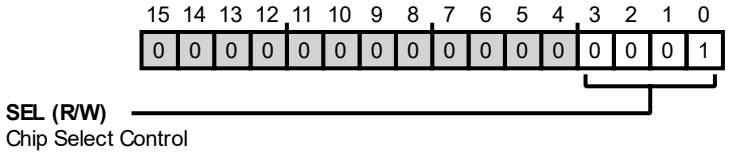


図 15-11：SPI_CS_CTL レジスタ図

表 15-4：SPI_CS_CTL レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
3:0 (R/W)	SEL	チップ・セレクト制御。 このフィールドは、現在の SPI 転送に使用される CS ラインを指定します。これは、様々なスレーブ間で CS ラインのみが固有であるようなマルチスレーブ設定で有効です。SPI マスタは、最大 4 つの異なる CS ラインをサポートできます。いずれのビットもセットしていない場合、デフォルトで CS0 が使用されます。 注：複数のビットをセットした場合、それぞれの CS ラインは同時にアクティブになります。

チップ・セレクト・オーバーライド

このレジスタはマスタ・モードでのみ使用します。

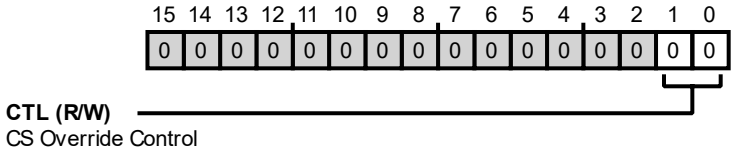


図 15-12：SPI_CS_OVERRIDE レジスタ図

表 15-5：SPI_CS_OVERRIDE レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
1:0 (R/W)	CTL	CS オーバーライド制御。 このビットは、マスタ・ステート・マシンからの実際の CS 出力をオーバーライドします。特殊な SPI 転送で必要とされる場合があります。注：注意して使用してください。	
		0	CS は強制されません。
		1	CS は強制的に 1'b1 を駆動します。
		2	CS は強制的に 1'b0 を駆動します。
		3	CS は強制されません。

SPI 設定

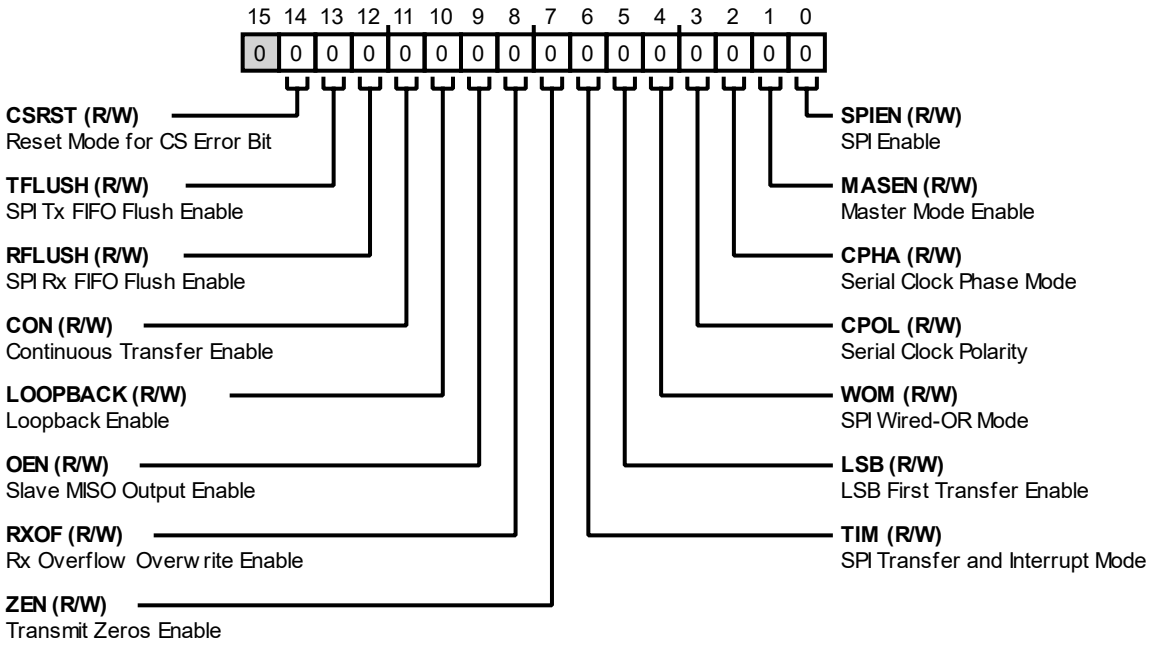


図 15-13：SPI_CTL レジスタ図

表 15-6：SPI_CTL レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
14 (R/W)	CSRST	CS エラー・ビットのリセット・モード。 このビットをセットすると、CS エラー状態の後にビット・カウンタがリセットされ、Cortex が SPI_CTL.SPIEN をクリアすることが期待されます。このビットがクリアされると、ビット・カウンタは停止した位置から継続します。CS がアサートされると SPI は残りのビットを受信可能で、Cortex は SPI_STAT.CSERR 割込みを無視する必要があります。ただし、CS エラー後の回復のために、このビットをセットすることを推奨します。
13 (R/W)	TFLUSH	SPI Tx FIFO フラッシュ・イネーブル。 このビットをセットすると、Tx FIFO がフラッシュされます。このビットは自動的にクリアされないため、1 回のみのフラッシュが必要な場合はトグルする必要があります。このビットをハイにしたままにすると、最後に送信された値または 0x00 が SPI_CTL.ZEN ビットに応じて送信されます。このビットをセットしていると、Tx FIFO への書込みは無視されます。 このビットをクリアすると、Tx FIFO のフラッシングが無効になります。

表 15-6：SPI_CTL レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応
12 (R/W)	RFLUSH	SPI Rx FIFO フラッシュ・イネーブル。 このビットをセットすると、Rx FIFO がフラッシュされます。このビットは自動的にクリアされないため、1 回のみのフラッシュが必要な場合はトグルする必要があります。このビットをセットしていると、すべての入力データは無視され、割込みは発生しません。セットし、SPI_CTL.TIM = 0 の場合、Rx FIFO の読出しによって転送が開始されます。 Rx FIFO フラッシュを無効にするには、このビットをクリアします。
11 (R/W)	CON	連続転送イネーブル。 セットすると、連続転送が無効になります。マスタ・モードでは、SPI_TX レジスタに有効なデータがなくなるまで（SPI_CTL.TIM = 1）、または Rx FIFO がフルになるまで（SPI_CTL.TIM = 0）、転送が継続されます。CS はアサートされると、Tx FIFO が空になるか、または Rx FIFO がフルになるまで、各 8 ビット・シリアル転送の期間アサート状態を維持します。 クリアすると、連続転送は無効になります。各 SPI フレームは、単一の 8 ビット・シリアル転送で構成されます。有効なデータが SPI_TX レジスタに存在する場合（SPI_CTL.TIM = 1）、または Rx FIFO がフルでない場合（SPI_CTL.TIM = 0）、1 シリアル・クロック・サイクルの停止期間後に新しい転送が開始されます。
10 (R/W)	LOOPBACK	ループバック・イネーブル。 セットすると、MISO を MOSI とテスト・ソフトウェアに接続します。クリアすると、通常のモードになります。
9 (R/W)	OEN	スレーブ MISO 出力イネーブル。 このビットをセットすると、MISO が通常どおりに動作します。このビットをクリアすると、MISO ピンの出力ドライバがディスエーブルされます。このビットをクリアした場合、MISO ピンはオープン・サーキットになります。
8 (R/W)	RXOF	Rx オーバーフロー・オーバーライトのイネーブル。 セットすると、SPI_RX レジスタ内の有効なデータは、受信した新しいシリアル・バイトによって上書きされます。クリアすると、受信した新しいシリアル・バイトは破棄されます。
7 (R/W)	ZEN	ゼロ送信イネーブル。 セットすると、Tx FIFO に有効なデータがない場合、0x00 を送信します。クリアすると、Tx FIFO に有効なデータがない場合、最後に送信した値を送信します。
6 (R/W)	TIM	SPI 転送および割込みモード。 セットすると、SPI_TX レジスタへの書込みで転送を開始定します。割込みは、SPI_IEN.IRQMODE + 1 バイト数が送信されたときにのみ発生します。 クリアすると、SPI_RX レジスタからの読出しで転送を開始します。割込みは、Rx-FIFO に SPI_IEN.IRQMODE + 1 バイト数以上が受信されたときにのみ発生します。
5 (R/W)	LSB	LSB ファースト転送イネーブル。 0：MSB を最初に送信 1：LSB を最初に送信

表 15-6：SPI_CTL レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
4 (R/W)	WOM	SPI ワイヤード OR モード。	
		0	通常の出力レベル
		1	オープン・サーキットのデータ出力をイネーブルします。データ出力ピンに外付けプルアップが必要です。
3 (R/W)	CPOL	シリアル・クロックの極性。	
		0	シリアル・クロックのアイドル状態をロー
		1	シリアル・クロックのアイドル状態をハイ
2 (R/W)	CPHA	シリアル・クロックの位相モード。	
		0	各シリアル・ビット転送の終了時にシリアル・クロック・パルス
		1	各シリアル・ビット転送の開始時にシリアル・クロック・パルス
1 (R/W)	MASEN	マスタ・モード・イネーブル。 注：このビットをクリアすると、デバイスとほとんどのステータス・ビットに同期リセットが発行されますが、他の MMR は影響を受けません。	
		0	スレーブ・モードが有効
		1	マスタ・モードが有効
0 (R/W)	SPIEN	SPI イネーブル。	
		0	SPI をディスエーブル
		1	SPI をイネーブル

SPI ボー・レート選択

このレジスタはマスタ・モードでのみ使用します。

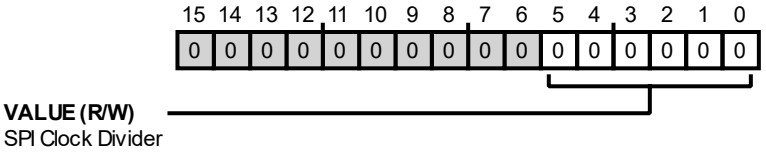


図 15-14：SPI_DIV レジスタ図

表 15-7：SPI_DIV レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
5:0 (R/W)	VALUE	SPI クロック・ドライバ。 `SPI_DIV.VALUE` は、シリアル・クロックを生成するために PCLK の分周に使用される係数です。

SPI DMA イネーブル

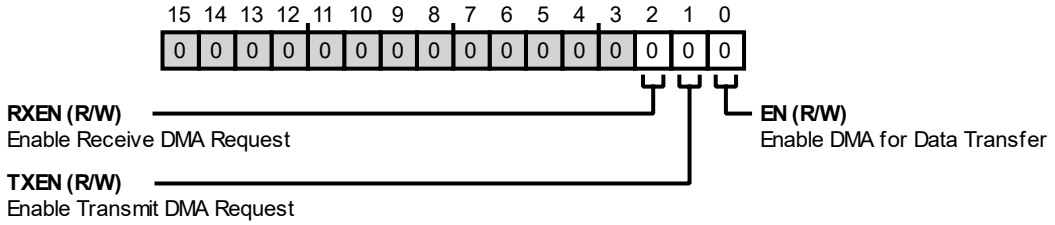


図 15-15：SPI_DMA レジスタ図

表 15-8：SPI_DMA レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
2 (R/W)	RXEN	受信 DMA リクエスト・イネーブル。 DMA をイネーブルしているときにこのビットをセットした場合、Rx FIFO に有効なデータがあると Rx DMA リクエストが発生します。
1 (R/W)	TXEN	送信 DMA リクエスト・イネーブル。 DMA をイネーブルしているときにこのビットをセットした場合、Tx FIFO に空があると Tx DMA リクエストが発生します。 注：このビットは、DMA コントローラへ追加の Tx DMA リクエストが行われないように、Tx DMA DONE 割込みを受け取るとすぐにクリアする必要があります。
0 (R/W)	EN	DMA データ転送イネーブル。 ユーザ・コードでセットすると、DMA 転送を開始します。

FIFO ステータス

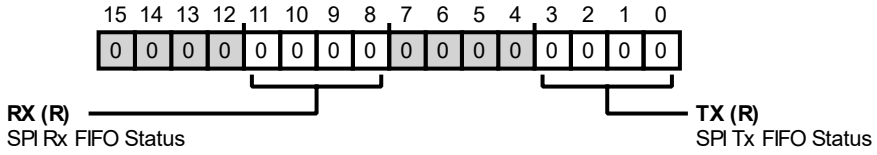


図 15-16：SPI_FIFO_STAT レジスタ図

表 15-9：SPI_FIFO_STAT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
11:8 (R/NW)	RX	SPI Rx FIFO ステータス。 このフィールドは、DMA がディスエーブルのときは、Rx FIFO 内のバイト数を示します。DMA モードでは、Rx FIFO 内のハーフワード数を示します。	
		0	Rx FIFO は空
		1	Rx FIFO に 1 つの有効なバイト／ハーフワードが存在
		2	Rx FIFO に 2 つの有効なバイト／ハーフワードが存在
		3	Rx FIFO に 3 つの有効なバイト／ハーフワードが存在
		4	Rx FIFO に 4 つの有効なバイト／ハーフワードが存在
		5	Rx FIFO に 5 つの有効なバイト／ハーフワードが存在
		6	Rx FIFO に 6 つの有効なバイト／ハーフワードが存在
		7	Rx FIFO に 7 つの有効なバイト／ハーフワードが存在
		8	Rx FIFO に 8 つの有効なバイト／ハーフワードが存在（Rx FIFO フル）
3:0 (R/NW)	TX	SPI Tx FIFO ステータス。 このフィールドは、DMA がディスエーブルのときは、Tx FIFO 内のバイト数を示します。DMA モードでは、Tx FIFO 内のハーフワード数を示します。	
		0	Tx FIFO は空
		1	Tx FIFO に 1 つの有効なバイト／ハーフワードが存在
		2	Tx FIFO に 2 つの有効なバイト／ハーフワードが存在
		3	Tx FIFO に 3 つの有効なバイト／ハーフワードが存在
		4	Tx FIFO に 4 つの有効なバイト／ハーフワードが存在
		5	Tx FIFO に 5 つの有効なバイト／ハーフワードが存在
		6	Tx FIFO に 6 つの有効なバイト／ハーフワードが存在
		7	Tx FIFO に 7 つの有効なバイト／ハーフワードが存在
		8	Tx FIFO に 8 つの有効なバイト／ハーフワードが存在（Tx FIFO がフル）

フロー制御

このレジスタはマスタ・モードでのみ使用します。

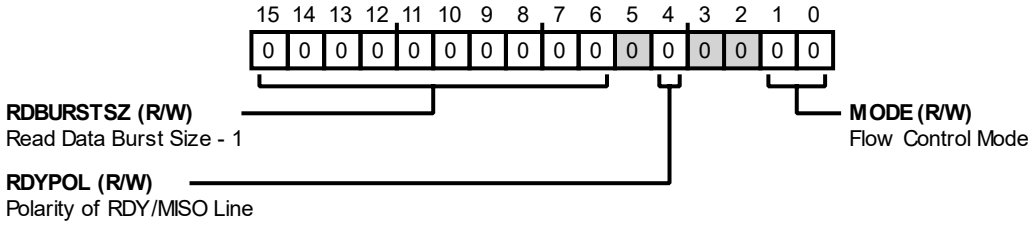


図 15-17：SPI_FLOW_CTL レジスタ図

表 15-10：SPI_FLOW_CTL レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15:6 (R/W)	RDBURSTSZ	読出しデータのバースト・サイズ －1。 フロー制御の待機前にスレーブから単一バーストで受信するバイト数 － 1 を指定します。 SPI_FLOW_CTL.MODE が 2'b00 の場合、これは無効です。SPI_FLOW_CTL.MODE が他の値の場合はすべて、このフィールドは有効です。これは 0～1023 の値で、それぞれ 1～1024 バイトの読出しバーストを意味します。 注：このモードは、固定幅の変換結果を定期的に読み出す場合に便利です。	
4 (R/W)	RDYPOL	RDY/MISO ラインの極性。 RDY/MISO ピンの極性を指定します。このピンはスレーブの読出しデータが準備できていることを示します。 SPI_FLOW_CTL.MODE = 2'b10 の場合、RDY ピンの極性を示します。 SPI_FLOW_CTL.MODE = 2'b11 の場合、MISO（DOUT）ラインの極性を示します。 SPI_FLOW_CTL.MODE が他の値の場合はすべて、このビットは無視されます	
		0	極性はアクティブ・ハイ。SPI マスタは RDY/MISO がハイになるまで待機します。
		1	極性はアクティブ・ロー。SPI マスタは RDY/MISO がローになるまで待機します。

表 15-10：SPI_FLOW_CTL レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
1:0 (R/W)	MODE	フロー制御モード。 データ読出しのためのフロー制御の設定。 注：RDY 信号をフロー制御に使用する場合、SPI モジュールのこの RDY 入力には任意の信号を接続できます。例えば、オフチップ入力、オンチップ・タイマー出力、または他の制御信号を接続できます。	
		0	フロー制御を無効化。
		1	フロー制御にタイマー（WAIT_TMR）を使用。
		2	フロー制御に RDY 信号を使用。
		3	フロー制御に MISO ピンを使用。

SPI 割込みイネーブル

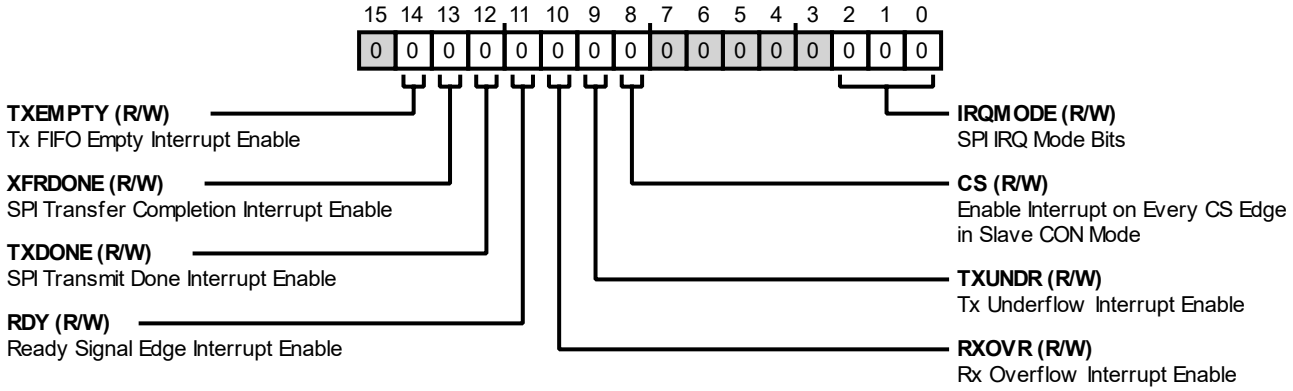


図 15-18：SPI_IEN レジスタ図

表 15-11：SPI_IEN レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
14 (R/W)	TXEMPTY	Tx FIFO エンプティ割込みイネーブル。 このビットは、Tx FIFO が空きになるごとに SPI_STAT.TXEMPTY 割込みをイネーブルします。	
		0	TXEMPTY 割込みをディスエーブルします。
		1	TXEMPTY 割込みをイネーブルします。
13 (R/W)	XFRDONE	SPI 転送完了割込みイネーブル。 このビットは、SPI_STAT.XFRDONE 割込みをイネーブルします。	
		0	XFRDONE 割込みをディスエーブルします。
		1	XFRDONE 割込みをイネーブルします。
12 (R/W)	TXDONE	SPI 送信完了割込みイネーブル。 このビットは、読出しコマンド・モードで、SPI_STAT.TXDONE 割込みをイネーブルします。 注：これは、読出しコマンド・モードにおいて SPI 転送方向の変化を知るために使用できます。	
		0	TXDONE 割込みをディスエーブルします。
		1	TXDONE 割込みをイネーブルします。

表 15-11：SPI_IEN レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
11 (R/W)	RDY	レディ信号エッジ割込みイネーブル。 RDY/MISO 信号でアクティブ・エッジが発生するたびに、SPI_STAT.RDY 割込みをイネーブルします。 SPI_FLOW_CTL.MODE = 2'b10 の場合、エッジ検出は RDY 信号で行われます。 SPI_FLOW_CTL.MODE = 2'b11 の場合、RDY 信号の代わりに MISO 信号が使用されます。 SPI_FLOW_CTL.MODE が他の値の場合はすべて、このビットは無効です。 アクティブ・エッジ（立上がり／立下がり）は SPI_FLOW_CTL.RDYPOL ビットで決められます。	
		0	RDY 信号エッジ割込みをディスエーブルします。
		1	RDY 信号エッジ割込みをイネーブルします。
10 (R/W)	RXOVR	Rx オーバーフロー割込みイネーブル。	
		0	Rx オーバーフロー割込みをディスエーブルします。
		1	Rx オーバーフロー割込みをイネーブルします。
9 (R/W)	TXUNDR	Tx アンダーフロー割込みイネーブル。	
		0	Tx アンダーフロー割込みをディスエーブルします。
		1	Tx アンダーフロー割込みイネーブルします。
8 (R/W)	CS	スレーブ CON モードで CS エッジごとに割込みをイネーブル。 このビットをセットし、SPI モジュールを連続モードでスレーブとして構成した場合、CS のエッジごとに割込みが生成され、対応するステータス・ビット（SPI_STAT.CSRISE、SPI_STAT.CSFALL）がアサートされます。 このビットをセットしない場合、割込みは発生せず、ステータス・ビットはアサートされません。 このビットは、SPI が連続モードでない場合、またはマスタである場合は無効です。	
2:0 (R/W)	IRQMODE	SPI IRQ モード・ビット。 これらのビットは、転送中に Tx/Rx 割込みを発生させるタイミングを設定します。DMA Rx 転送の場合、これらのビットは 3'b000 でなければなりません。	
		0	Tx 割込みは、1 バイトが転送されたときに発生します。Rx 割込みは、1 バイト以上が FIFO に受信されたときに発生します。
		1	Tx 割込みは、2 バイトが転送されたときに発生します Rx 割込みは、2 バイト以上が FIFO に受信されたときに発生します。
		2	Tx 割込みは、3 バイトが転送されたときに発生します Rx 割込みは、3 バイト以上が FIFO に受信されたときに発生します。

表 15-11：SPI_IEN レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
		3	Tx 割込みは、4 バイトが転送されたときに発生します Rx 割込みは、4 バイト以上が FIFO に受信されたときに発生します。
		4	Tx 割込みは、5 バイトが転送されたときに発生します Rx 割込みは、5 バイト以上が FIFO に受信されたときに発生します。
		5	Tx 割込みは、6 バイトが転送されたときに発生します Rx 割込みは、6 バイト以上が FIFO に受信されたときに発生します。
		6	Tx 割込みは、7 バイトが転送されたときに発生します Rx 割込みは、7 バイト以上が FIFO に受信されたときに発生します。
		7	Tx 割込みは、8 バイトが転送されたときに発生します Rx 割込みは、FIFO がフルになると発生します。

読出し制御

このレジスタはマスタ・モードでのみ使用します。

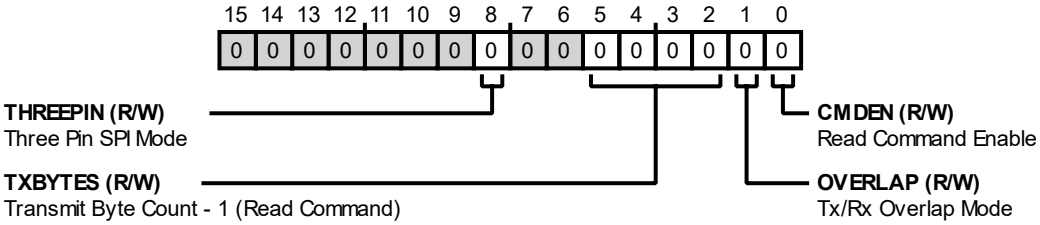


図 15-19：SPI_RD_CTL レジスタ図

表 15-12：SPI_RD_CTL レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
8 (R/W)	THREEPIN	3 ピン SPI モード。 SPI インターフェースは、双方向データピン（3 ピン・インターフェース）か、Tx および Rx（4 ピン・インターフェース）専用の単一方向データ・ピンかを指定します。これは、読出しコマンド・モードかつ SPI_FLOW_CTL.MODE = 2'b01 の場合にのみ有効です。 3 ピン・モードを選択した場合、MOSI ピンはマスタによって送信フェーズ中に駆動され、SPI_WAIT_TMR の SCLK サイクルの待機時間後にスレーブが同じ MOSI ピンを駆動することが期待されます。 注：ターンアラウンド時間を許容するための待機状態を設けるために、SPI_FLOW_CTL.MODE を 2'b01 に設定する必要があります。それ以外の場合、スレーブには SCLK の半分の時間（SCLK のサンプリング・エッジから駆動エッジまでの間）のターンアラウンド時間しかありません。このモードを使用する場合は、SPI_RD_CTL.OVERLAP = 0 に設定します。	
		0	SPI を 4 ピン・インターフェースにします。
		1	SPI を 3 ピン・インターフェースにします。
5:2 (R/W)	TXBYTES	送信バイト数 －1（読出しコマンド）。 スレーブからデータを読み出す前に送信するバイト数 －1 を指定します。このフィールドには、1～16 Tx バイトに応じて 0～15 の値を設定できます。これには、スレーブに送信すべきすべてのバイト、すなわち、コマンド、およびアドレス（必要な場合）が含まれます。設計ではこれら 2 つを区別しません。Tx FIFO から指定バイト数を送信するだけです。 注：コマンド送信からデータ受信までの間に遅延がある場合、その遅延のために付加される Tx バイト（ほとんど 0）の数もこれにカウントされます。	

表 15-12：SPI_RD_CTL レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
1 (R/W)	OVERLAP	Tx/Rx オーバーラップ・モード。 このビットは、Tx と Rx の開始時点をオーバーラップさせるかどうかを指定します。ほとんどのスレーブでは、マスタが「コマンド+アドレス」の送信を完了した後にのみ、データの読出しを開始します。これはオーバーラップのない転送です。スレーブによっては、コマンドの受信中にステータス・バイトを送信することがあります。したがって、CS フレームの先頭からバイトの受信を開始することが必要な場合があります。これがオーバーラップ・モードです。 注：オーバーラップ・モードの場合、SPI_CNT.VALUE は受信するバイトの合計数を示します。したがって、SPI_CNT.VALUE の設定の際には（実際の読出しバイトに加えて）追加のステータス・バイトを考慮する必要があります。	
		0	Tx/Rx オーバーラップをディスエーブルします。
		1	Tx/Rx オーバーラップをイネーブルします。
0 (R/W)	CMDEN	読出しコマンド・イネーブル。 コマンド+アドレスを送信し、同じ CS フレーム内でデータ読出しが期待される SPI 読出しコマンド・モード。このビットをクリアした場合、SPI_RD_CTL、SPI_FLOW_CTL、SPI_WAIT_TMR レジスタの他のいずれのフィールドにも影響しません。	
		0	読出しコマンド・モードをディスエーブルします。
		1	読出しコマンド・モードをイネーブルします。

受信

このレジスタを使用すると、深さ 8 の受信 FIFO にアクセスできます。

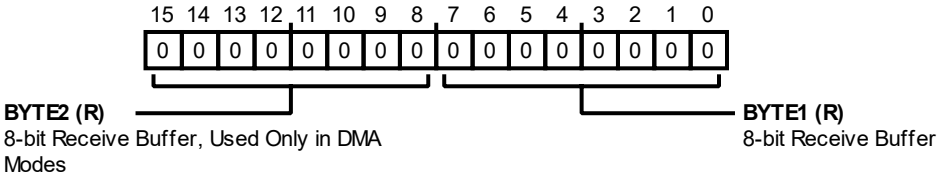


図 15-20：SPI_RX レジスタ図

表 15-13：SPI_RX レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:8 (R/NW)	BYTE2	8 ビット受信バッファ。DMA モードでのみ使用。 これらの 8 ビットは SPI_DMA モードでのみ使用し、すべての FIFO アクセスはハーフワード・アクセスで行われます。SPI_DMA をディスエーブルした場合、ゼロを返します。
7:0 (R/NW)	BYTE1	8 ビット受信バッファ。

ステータス

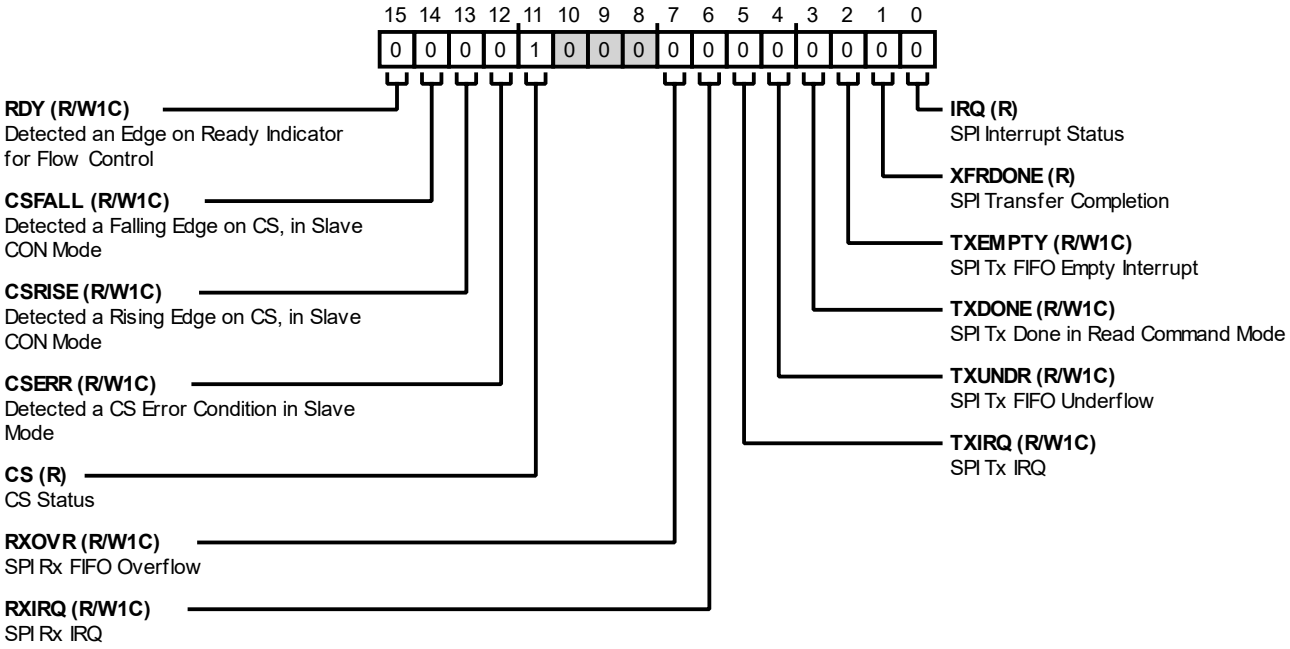


図 15-21：SPI_STAT レジスタ図

表 15-14：SPI_STAT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W1C)	RDY	フロー制御でレディ・インジケータのエッジを検出。 このビットは、フロー制御モードに応じて RDY/MISO ラインにアクティブ・エッジが存在したことを示します。SPI_FLOW_CTL.MODE = 2'b10 の場合、RDY 信号でアクティブ・エッジが検出されると、このビットがセットされます。 SPI_FLOW_CTL.MODE = 2'b11 の場合、MISO 信号でアクティブ・エッジが検出されると、このビットがセットされます。 他のすべての MODE 値では、このビットは常に 0 です。 アクティブ・エッジ（立上がり／立下がり）は SPI_FLOW_CTL.RDYPOL で決められます。 SPI_IEN.RDY をセットした場合、このビットにより割込みが発生し、このビットに 1 を書き込んだときにのみクリアされます。 注：これは、送信側でのスタッガード・フロー制御に、必要に応じて CS オーバーライドと共に使用できます。

表 15-14：SPI_STAT レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
14 (R/W1C)	CSFALL	スレーブ CON モードで CS の立下がりエッジを検出。 このビットは、デバイスが連続モードでスレーブであり、SPI_IEN.CS がアサートされたときに、CS ラインに立下がりエッジがあったことを示します。このビットにより割込みが発生します。このビットに 1 を書き込むか、SPI_CTL.SPIEN をクリアすると、このビットはクリアされます。 これは、SPI データ・フレームの開始を識別するのに使用できます。
13 (R/W1C)	CSRISE	スレーブ CON モードで CS の立上がりエッジを検出。 このビットは、デバイスが連続モードでスレーブであり、SPI_IEN.CS がアサートされたときに、CS ラインに立上がりエッジがあったことを示します。このビットにより割込みが発生します。このビットに 1 を書き込むか、SPI_CTL.SPIEN をクリアすると、このビットはクリアされます。 これは、SPI データ・フレームの終了を識別するのに使用できます。
12 (R/W1C)	CSERR	スレーブ・モードで CS エラー状態を検出。 このビットは、フル・バイトのデータが完全に送信される前であるにもかかわらず、CS ラインが外部マスタによって突然アサート解除されたことを示します。 このビットにより割込みが発生します。このビットに 1 を書き込むか、SPI_CTL.SPIEN をクリアすると、このビットはクリアされます。
11 (R/NW)	CS	CS ステータス。 このビットは、SPI モジュールが認識する実際の CS ステータスを反映します。 注：これは、SCLK-PCLK 同期を使用します。したがって、CS が状態を変えると際にわずかな遅延が生じます。 0：CS ラインがロー。 1：CS ラインがハイ。
7 (R/W1C)	RXOVR	SPI Rx FIFO オーバーフロー。 新しいデータが FIFO にロードされたとき、既に Rx FIFO がフルになっている場合にセットされます。SPI_CTL.RFLUSH をセットしている場合を除き、SPI_IEN.RXOVR をセットしている場合、このビットにより割込みが発生します。 このビットに 1 を書き込むか、SPI_CTL.SPIEN をクリアすると、このビットはクリアされます。
6 (R/W1C)	RXIRQ	SPI Rx IRQ。 受信割込みが発生したときにセットされます。DMA モードでは使用できません。このビットは、SPI_CTL.TIM がクリアされ、必要なバイト数が受信されたときにセットされます。このビットに 1 を書き込むか、SPI_CTL.SPIEN をクリアすると、このビットはクリアされます。
5 (R/W1C)	TXIRQ	SPI Tx IRQ。 SPI Tx IRQ ステータス・ビット。DMA モードでは使用できません。 送信割込みが発生したときにセットされます。このビットは、SPI_CTL.TIM がセットされ、必要なバイト数が送信されたときにセットされます。 このビットに 1 を書き込むか、SPI_CTL.SPIEN をクリアすると、このビットはクリアされます。

表 15-14：SPI_STAT レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
4 (R/W1C)	TXUNDR	SPI Tx FIFO アンダーフロー。 このビットは、Tx FIFO に有効なデータが存在しないまま送信が開始されたときにセットされます。SPI_CTL.TFLUSH をセットしている場合を除き、SPI_IEN.TXUNDR をセットしている場合、このビットにより割込みが発生します。 このビットに 1 を書き込むか、SPI_CTL.SPIEN をクリアすると、このビットはクリアされます。
3 (R/W1C)	TXDONE	SPI 読出しコマンド・モードで SPI Tx 完了。 このビットは、読出しコマンドで送信全体が完了したときにセットされます。SPI_IEN.TXDONE をセットしている場合、このビットにより割込みが生成されます。 このビットは、SPI_RD_CTL.CMDEN をセットしている場合にのみ有効です。このビットに 1 を書き込んだときにのみクリアされます。
2 (R/W1C)	TXEMPTY	SPI Tx FIFO エンプティ割込み。 SPI_CTL.TFLUSH をセットしている場合を除き、SPI_IEN.TXEMPTY をセットしている場合、Tx-FIFO が空になるとこのビットがセットされます。このビットにより割込みが生成されます。 このビットに 1 を書き込むか、SPI_CTL.SPIEN をクリアすると、このビットはクリアされます。
1 (R/NW)	XFRDONE	SPI 転送完了。 このビットはマスタ・モードでの SPI 転送完了の状態を示します。これは、SPI_CNT.VALUE バイト数の転送が完了したときにセットされます。スレーブ・モードまたは SPI_CNT.VALUE = 0 の場合、このビットは無効です。 SPI_IEN.XFRDONE をセットしている場合、このビットにより割込みが生成されます。マスタ・ステート・マシンのステートを使用して、SPI 転送の完了を判断します。したがって、CS のオーバーライドはこのビットに影響しません。このビットに 1 が書き込まれた場合にのみクリアされます。
0 (R/NW)	IRQ	SPI 割込みステータス SPI により割込みが発生すると 1 にセットされます。すべての割込み源をクリアすると、クリアされます。

送信

このレジスタを使用すると、深さ 8 の送信 FIFO にアクセスできます。

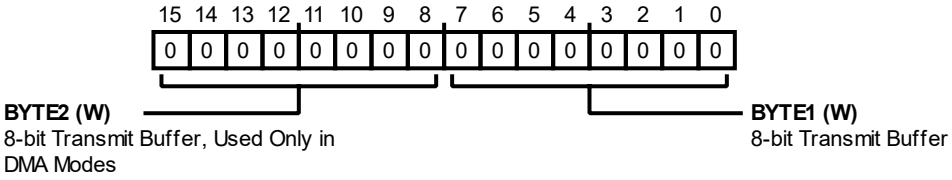


図 15-22：SPI_TX レジスタ図

表 15-15：SPI_TX レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:8 (RX/W)	BYTE2	8 ビット送信バッファ。DMA モードでのみ使用。 これらの 8 ビットは SPI_DMA モードでのみ使用し、すべての FIFO アクセスはハーフワード・アクセスで行われます。
7:0 (RX/W)	BYTE1	8 ビット送信バッファ。

フロー制御の待機タイマー

このレジスタはマスタ・モードでのみ使用します。

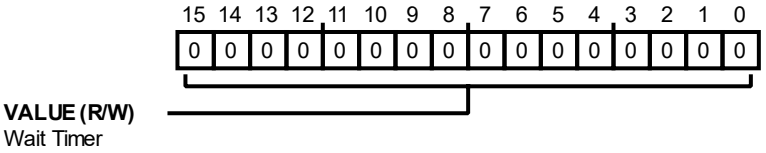


図 15-23：SPI_WAIT_TMR レジスタ図

表 15-16：SPI_WAIT_TMR レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	待機タイマー このフィールドは、SPI 読出しを続行する前に待機する SCLK サイクル数を指定します。このフィールドの値には 0～65535 を指定できます。 このフィールドは、SPI_FLOW_CTL.MODE = 2'b01 の場合にのみ有効です。 SPI_FLOW_CTL.MODE が他の値の場合はすべて、このフィールドは無視されます。0 の値は、1 SCLK サイクルの待機時間を意味します。

16 シリアル・ポート（SPORT）

ADuCM4050 MCU は、様々なシリアル・データ通信プロトコルをサポートするシリアル・ポート（SPORT）を搭載しています。更に、SPORT は、業界標準の様々なデータ・コンバータ、コーデック、および DSP をはじめとする他のプロセッサに対する直接のハードウェア・インターフェースとなります。SPORT のトップ・モジュールは、同一機能を持つ SPORT_A および SPORT_B という 2 つのハーフ SPORT（HSPORT）で構成されています。各ハーフ SPORT は、トランスミッタまたはレシーバーとして個別に設定することも、同一の SPORT 内でもう一方の HSPORT と組み合わせることも可能です。各ハーフ SPORT は同じ性能を備えており、同じ方法でプログラムすることができます。

SPORT の機能

各 HSPORT は、以下の機能をサポートしています。

- トランスミッタまたはレシーバーとして設定可能な 1 本の双方向データ・ライン。更に、2 つのハーフ SPORT を組み合わせると、全二重動作を実現できます。
- 動作モード：
  - 標準的な DSP シリアル・モード
  - タイマー・イネーブル・モード
  - ゲーテッド・クロック・モード
- 4 ビット～32 ビットの長さのシリアル・データ・ワード。
- 内部クロックの生成がきめ細かなため、端数のない PCLK／SPT_CLK 比を実現。SPORT は、外部ソースからクロックを入力することも可能です。
- データの駆動またはサンプリング、およびフレーム同期のために、SPT_CLK の立上がりエッジまたは立下がりエッジを設定可能。
- 内部クロック・モードの場合にゲーテッド・クロック・モードをサポート。
- フレーム同期オプション：
  - フレームなしモード
  - 内部／外部フレーム同期オプション
  - 極性をプログラム可能

- 早期／遅延フレーム同期

- 外部フレーム同期信号をレベル・センシティブな信号として設定。
- 外部フレーム同期を早まって受信した場合と、送信アンダーフロー（または受信オーバーフロー）の場合にステータス・フラグが立ち、オプションで割込みが発生。
- オプションとして、SPORT がレシーバーとして設定されている場合、16 ビット・ワードを 32 ビット・ワードに、または 8 ビット・ワードを 32 ビット・ワードにパッキングします。SPORT がトランスミッタとして設定されている場合は、32 ビット・ワードを 16 ビット・ワードに、または 32 ビット・ワードを 8 ビット・ワードにアンパッキング。
- 割込みで駆動する転送をサポート。
- 割込み制御。
- 転送終了割込み（TFI）：プログラムされた転送数の最終ビットが送出されたとき、割込みが発生します。
- MSB/LSB ファーストのビット順をプログラム可能。
- 1 ワード～4095 ワードの転送カウントをプログラム可能。
- ハーフ SPORT ごとに専用の DMA チャンネルを用いて、コア転送と DMA 転送の両方を実現可能。
- SPORT モジュールの 2 つのハーフ SPORT 間で、クロックやフレーム同期の経路設定と共有を行う機能。
- タイマー・イネーブル・モードの場合に SPT_CNV 信号を生成。
- 各ハーフ SPORT には専用の 1 組のコントロール・レジスタとデータ・バッファを実装。

信号の説明

各ハーフ SPORT モジュールは、次の表に示すように 4 本の専用ピンを備えています。

表 16-1：SPORT ピンの説明

内部ノード	方向	説明
SPT_CLK	I/O	送信／受信シリアル・クロック。データとフレーム同期は、このクロックを基準にして駆動／サンプリングされます。この信号は、内部でも外部でも生成できます。
SPT_FS	I/O	送信／受信フレーム同期。フレーム同期パルスにより、シリアル・データのシフトが開始されます。この信号は、内部でも外部でも生成できます。
SPT_D0	I/O	送信／受信データ・チャンネル。双方向データ・ピン。この信号は、送信シリアル・データのための出力、または受信シリアル・データのための入力として設定できます。
SPT_CNV	Output	この信号は、タイマー・イネーブル・モードの場合にのみ使用される付加制御信号で、制御情報のために 2 つの信号を必要とするペリフェラルに対して用いられます。

- クロックとフレーム同期信号は、ハーフ SPORT のペア間で相互接続が可能です。
- SPT_CNV 信号は、タイマー・イネーブル・モードでのみ使用されます（タイマー・イネーブル・モードを参照）。

SPORT 機能の説明

ここでは、MCU のシリアル・ポートの機能の概要を示します。

SPORT のブロック図

SPORT のトップレベル・ブロック図を下記に示します。

この図は、APB および DMA チャンネルとのインターフェースを示しています。また、ペリフェラルとの接続も示しています。

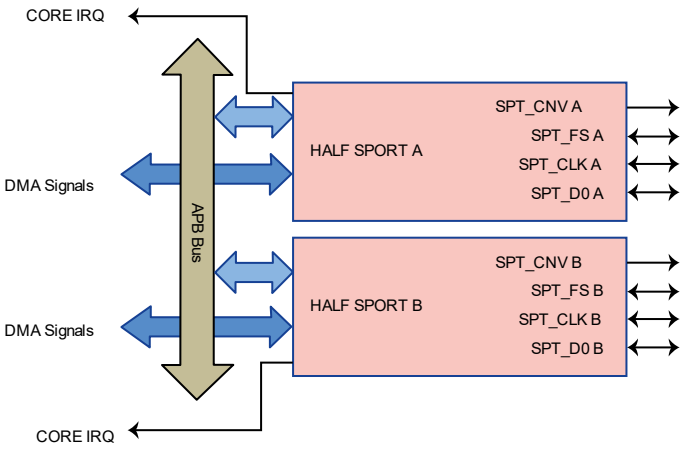


図 16-1：SPORT のトップレベル・ブロック図

次の図は、ハーフ SPORT のブロック図を示しています。

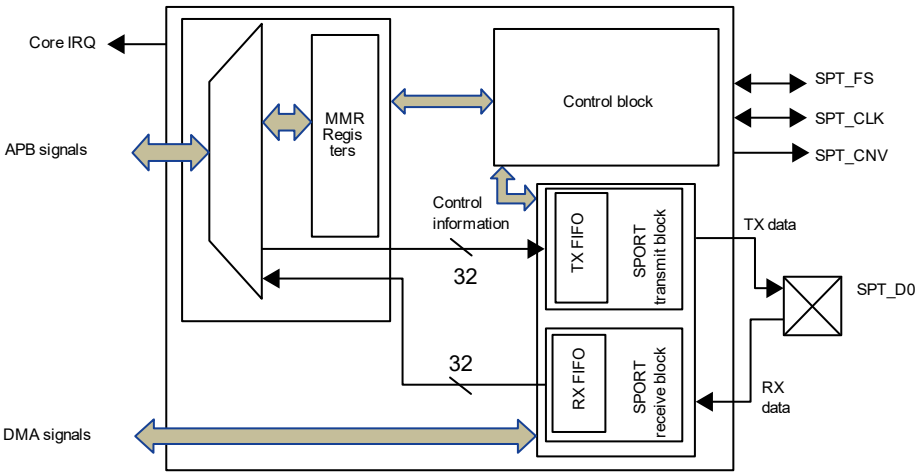


図 16-2：ハーフ SPORT のブロック図

各 SPORT モジュールは、同一機能を持つハーフ SPORT（HSPORT）の A および B と呼ばれる 2 つの独立したブロックで構成されています。これらのブロックは、トップレベル・ブロック図に示すように、トランスミッタまたはレシーバーとして個別に設定することができます。

各 HSPORT は、専用の 1 組のコントロール・レジスタとデータ・バッファを備えています。全二重動作を実現するためには、2 つの HSPORT を組み合わせる必要があります。

SPORT の転送

SPORT_CTL_A.SPTRAN 制御ビット、または SPORT_CTL_B.SPTRAN 制御ビットを 1 に設定すると、SPORT は送信モードに設定されます。このビットをクリアすると、シリアル・ポートは受信モードに設定されます。パスがアクティブになると、データはシリアル・クロックのレートでフレーム同期に応答してシフトします。SPORT が送信モードで動作している場合は受信データ・バッファが停止し、SPORT が受信モードで動作している場合は送信データ・バッファが停止します。停止しているデータ・バッファは使用できないため、アクセスしないでください。アプリケーション・プログラムには、適切なデータ・バッファを使用する必要があります。

送信パス

SPORT_CTL_A.SPTRAN 制御ビット、または SPORT_CTL_B.SPTRAN 制御ビットを 1 に設定すると、SPORT は送信モードに設定されます。

SPORT_TX_A レジスタと SPORT_TX_B レジスタは、（深さ 3 の FIFO である）送信データ・バッファとして使用されます。

送信されるデータは、SPORT_TX_A レジスタと SPORT_TX_B レジスタに書き込まれます。このデータは、更に送信シフト・レジスタに転送されます。シフト・レジスタは SPT_CLK 信号をクロック源としているため、このデータはクロックに同期して、SPT_D0 からシリアルにシフト・アウトされます。フレーミング信号を使用する場合、SPT_FS 信号がシリアル・ワードの送信開始を示します。

SPORT がトランスミッタとして設定されると、イネーブルになった SPORT のデータ・ピン SPT_D0 は常に駆動されます。

注：シリアル・ワード長よりも長い時間が経過した後に次のフレーム同期が割り当てられると、停止しているシリアル・クロック・サイクル（現在のフレーム内のデータを送信した後のクロック・サイクル）の間、SPORT はデータ・ピンから送信される次のワードの最初のビットを駆動します。これによって、レシーバー側で問題が発生することはありません。SPORT は、有効なフレーム同期を検出した後にのみデータ・ピンのサンプリングを開始するためです。なお、これは、フレーム同期の停止中はクロックが存在しないゲーテッド・クロック・モードには該当しません。

シリアル・ポートは、送信データ・バッファのステータスと、アンダーランなどの送信エラーに対応するエラー検出ロジックのステータスを提供します。詳細な情報については、エラー検出を参照してください。

シリアル・ポートが送信モードに設定されると、受信データ・パスは停止し、シリアル・クロック信号にもフレーム同期信号にも応答しません。したがって、Tx モードにおいてエンプティ状態の受信データ・バッファから読み出すことは推奨しません。

受信パス

SPORT_CTL_A.SPTRAN ビットをクリアすると、SPORT は受信モードに設定されます。

SPORT_RX_A レジスタと SPORT_RX_B レジスタは、受信データ・バッファ（深さ 3 の FIFO）で、APB を通してアクセス可能です。

Rx モードでは、入力シフト・レジスタが SPT_CLK に同期してデータ・ビットを SPT_D0 からシフト・インします。フレーミング信号を使用する場合、SPT_FS x 信号は受信したシリアル・ワードの開始を示します。チャンネル上でワード全体がシフト・インすると、データは SPORT_RX_x から読出し可能になります。

シリアル・ポートは、受信データ・バッファのステータスと、オーバーフローなどの受信エラーに対応するエラー検出ロジックのステータスを提供します。詳細な情報については、エラー検出を参照してください。

シリアル・ポートが受信モードに設定されると、送信パスは停止し、シリアル・クロック信号にもフレーム同期信号にも応答しません。したがって、Rx モードで送信データ・バッファにアクセスすることは推奨しません。

マルチプレクサ・ロジック

SPORT と PinMux ロジックの間に、マルチプレクスするためのブロックが内蔵されています。これにより、2 つのハーフ SPORT のペア間でクロック信号とフレーム同期信号の経路設定と共有を柔軟に行うことができます。この機能は、インターフェースでのピンの総数を低減するために使用することができ、また、ハーフ SPORT のペアを全二重動作向けに使用するときに効果的です。

この機能を設定するために、`SPORT_CTL_A.CKMUXSEL` と、`SPORT_CTL_A.FSMUXSEL` が使用されます。

下図は、2 つのハーフ SPORT のクロックの間に接続されたこのマルチプレクサの動作を示しています。ハーフ SPORT A は、隣のハーフ SPORT B からクロックを入力するようにプログラムできます。

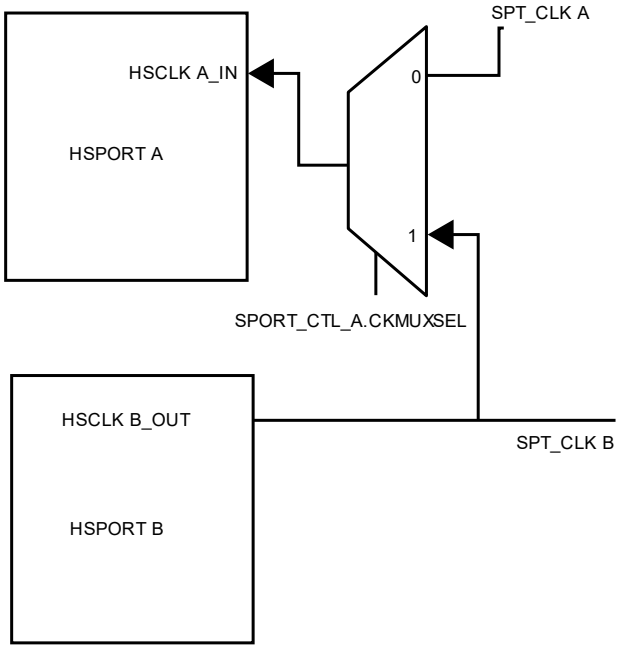


図 16-3：HSPORT A が HSPORT B の内部クロックを使用する場合のマルチプレクサ・ロジック

`SPORT_CTL_A.CKMUXSEL` が 1 の場合、`SPORT_CTL_B.ICLK` の値を 1 にすると、HSPORT A は HSPORT B の内部クロックを受信します。

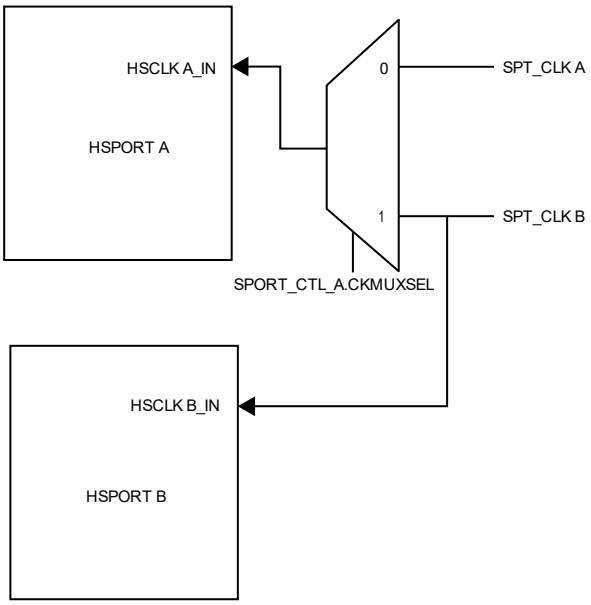


図 16-4：HSPORT A と HSPORT B が同じ外部クロックを使用する場合のマルチプレクサ・ロジック

SPORT_CTL_A.CKMUXSEL が 1 で、SPORT_CTL_B.ICLK が 0 の場合、両方のハーフ SPORT は同じ外部クロックを受信します。

SPORT_CTL_A.CKMUXSEL が 0 の場合、両方のハーフ SPORT でのクロックは通常動作になります。

下図は、2 つのハーフ SPORT のフレーム同期の間に接続されたマルチプレクサの動作を示しています。ハーフ SPORT A は、隣のハーフ SPORT B からフレーム同期を入力するようにプログラムできます。

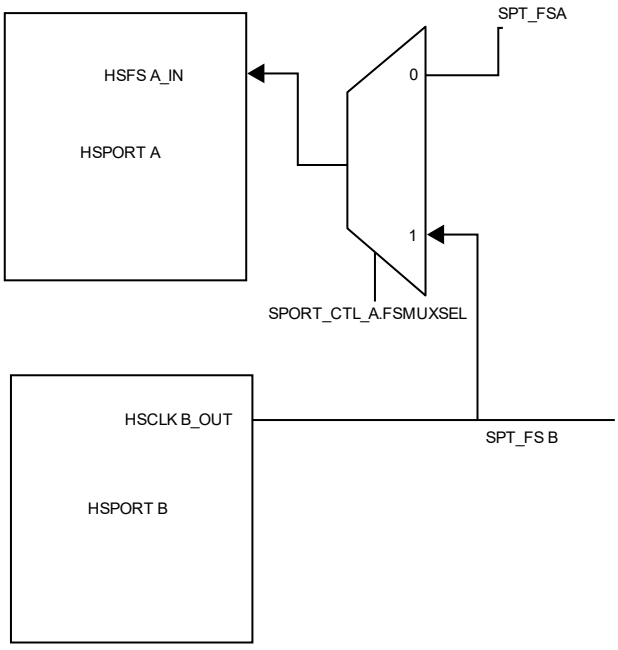


図 16-5：HSPORT A が HSPORT B の内部フレーム同期を使用する場合のマルチプレクサ・ロジック

SPORT_CTL_A.FSMUXSEL が 1 の場合、SPORT_CTL_B.IFS の値を 1 にすると、HSPORT A は HSPORT B の内部フレーム同期を受信します。

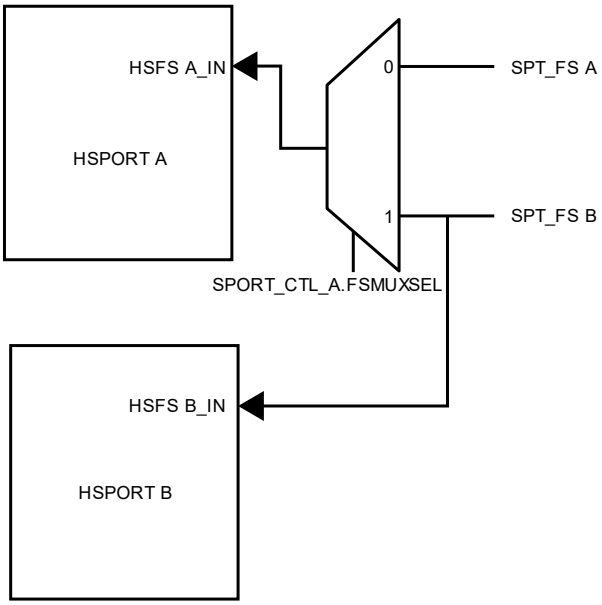


図 16-6：HSPORT A と HSPORT B が同じ外部フレーム同期を使用する場合のマルチプレクサ・ロジック

SPORT_CTL_A.FSMUXSEL が 1 で、SPORT_CTL_B.IFS が 0 の場合、両方のハーフ SPORT は同じ外部フレーム同期を受信します。

SPORT_CTL_A.FSMUXSEL が 0 の場合、両方のハーフ SPORT でのフレーム同期は通常動作になります。

2 つのハーフ SPORT 間をマルチプレクスするとき、SPORT_CTL_A.CKRE、SPORT_CTL_B.CKRE、SPORT_CTL_A.LFS、および SPORT_CTL_B.LFS などの極性ビットは、同じ値に設定する必要があります。また、同じ内部クロック／フレーム同期が両方のハーフ SPORT で使用されているとき、転送数（NUMTRAN）の値も両方のハーフ SPORT で同じ数値にプログラムする必要があります。

注：HSPORT A には、外部クロック・モードに設定されているときだけ、HSPORT B からシリアル・クロック信号を入力することができます。同様に、HSPORT A には、外部フレーム同期モードに設定されているときだけ、HSPORT B からフレーム同期信号を入力することができます。この共有を可能にするためには、HSPORT A に対してのみ SPORT_CTL_A レジスタ（SPORT_CTL_A.CKMUXSEL と SPORT_CTL_A.FSMUXSEL）の設定が必要です（HSPORT B に対する設定は不要）。HSPORT B には、HSPORT A からのシリアル・クロック信号とフレーム同期信号を入力することはできません。

シリアル・クロック

シリアル・クロック（SPT_CLK）信号は、データ・ビットをどのように駆動またはサンプリングするかを制御します。フレーム同期信号は、シリアル・クロックを基準にして（内部フレーム同期モードで）駆動されたり、（外部フレーム同期モードで）サンプリングされたりします。シリアル・クロックは、MCU のシステム・クロックから内部で生成することも、SPORT_CTL_x.ICLK（SPORT_CTL_A.ICLK または SPORT_CTL_B.ICLK）ビットに基づいて外部から供給することもできます。SPORT が内部クロック・モードに設定された場合（SPORT_CTL_x.ICLK = 1）、SPORT_DIV_x.CLKDIV（SPORT_DIV_A.CLKDIV または SPORT_DIV_B.CLKDIV）フィールドを用いて分周器を設定すると、基本的なクロック HCLK からシリアル・ポート・クロック信号を生成できます。この除数が 16 ビットの値であるため、広範囲のシリアル・クロック・レートを実現できます。

次式を使用すると、シリアル・クロックの周波数を算出できます。

$$f_{SPT\_CLK} = \frac{HCLK}{2x(1+SPORT\_DIV\_xCLKDIV)}$$

ゲーテッド・クロック・モードでは、シリアル・ポートはデータが有効な期間だけアクティブになるゲーテッド・クロックを生成するように設定できます。

SPORT が外部クロック・モード（SPORT_CTL_x.ICLK = 0）に設定された場合、シリアル・クロックは入力信号になり、SPORT はスレーブ・モードで動作します。SPORT_DIV_x.CLKDIV は無視されます。

外部から供給されるシリアル・クロックは、MCU のシステム・クロックに対して、常に非同期クロックであると見なされます。正確な AC タイミング仕様については、**ADuCM4050 パワー・マネージメントを統合した超低消費電力 ARM Cortex-M4F MCU** のデータシートを参照してください。

フレーム同期

フレーム同期は、新しいワードまたはフレームの開始を特定するために使用する制御信号です。シリアル・ポートは、この信号を検出すると同時に、選択された方向に基づいて、データ・ビットをシリアルにシフト・インまたはシフト・アウトし始めます。フレーム同期信号は、シリアル・クロック（SPT_CLK x）から内部で生成することも、SPORT_CTL_x.IFS（SPORT_CTL_A.IFS または SPORT_CTL_B.IFS）ビットの設定値に基づいて外部から供給することもできます。

SPORT が内部フレーム同期モード（SPORT_CTL_x.IFS = 1）に設定された場合、SPORT_DIV_x.FSDIV（SPORT_DIV_A.FSDIV または SPORT_DIV_B.FSDIV）フィールドを用いて分周器を設定すると、シリアル・クロックから SPT_FS x 信号を生成できます。この除数が 8 ビットの値であるため、広範囲なフレーム同期レートで、周期的な転送を開始できます。

フレーム同期の間のシリアル・クロック数 =（SPORT_DIV_x.FSDIV + 1）。

SPORT がタイマー・イネーブル・モードで動作しているとき、SPT_CNV 信号を内部で生成するために FSDIV が使用されます。したがって、タイマー・イネーブル・モードの場合、このフィールドは SPT_CNV パルス間のシリアル・クロックのサイクル数を示しています。

SPORT_DIV_x.FSDIV の値は、SPORT_CTL_x.SLEN（SPORT_CTL_A.SLEN または SPORT_CTL_B.SLEN）ビット・フィールド（シリアル・ワード長 − 1）の値よりも小さくしないでください。これにより外部のデバイスが現在の動作をアボートしたり、別の予測できない結果が生じたりする可能性があるためです。

注：SPORT をイネーブルにした後、最初の内部フレーム同期（または、タイマー・イネーブル・モードの場合の CNV）は（SPORT_DIV_x.FSDIV + 1）個のシリアル・クロック期間が経過した後に現れます。

SPORT が外部フレーム同期モード（SPORT_CTL_x.IFS = 0）に設定された場合、SPT_FS x は入力信号になり、SPORT_DIV_x（SPORT_DIV_A または SPORT_DIV_B）レジスタの SPORT_DIV_x.FSDIV フィールドは無視されます。デフォルト設定によって、この外部信号はレベル・センシティブです。フレーム同期は、シリアル・クロックと同期することが望まれます。そうでない場合、フレーム同期は、**ADuCM4050 パワー・マネージメントを統合した超低消費電力 ARM Cortex-M4F MCU のデータシート**に記載されたタイミング条件を満たす必要があります。

注：SPORT が外部フレーム同期でイネーブルになっている場合、フレーム同期をアサート解除する必要があります。

フレーム同期オプション

フレーミング信号は、アクティブ・ハイにもアクティブ・ローにも設定することができます。SPORT_CTL_x.LFS ビット（SPORT_CTL_A.LFS または SPORT_CTL_B.LFS）によって、フレーム同期信号のロジック・レベルを選択します。

- SPORT_CTL_x.LFS = 0 にすると、対応するフレーム同期信号はアクティブ・ハイになります。フレーム同期信号のデフォルトの極性です。
- SPORT_CTL_x.LFS = 1 にすると、対応するフレーム同期信号はアクティブ・ローになります。

以降の 3 セクションでは、ある動作モードの場合にシリアル・ポートでフレーム同期信号がどのように使用されるかを説明します。

データ依存のフレーム同期とデータ独立のフレーム同期

SPORT がトランスミッタとして設定された場合、すなわち SPORT_CTL_x.SPTRAN ビット（SPORT_CTL_A.SPTRAN または SPORT_CTL_B.SPTRAN）が 1 の場合で、更にデータ依存のフレーム同期が選択された場合、すなわち SPORT_CTL_x.DIFS ビット（SPORT_CTL_A.DIFS または SPORT_CTL_B.DIFS）が 0 の場合に、新しいデータ・ワードが SPORT の送信バッファにロードされたときだけ、内部で生成された送信フレーム同期が駆動されます。すなわち、フレーム同期信号の生成とデータの送信はデータに依存します。データが準備されていないとき、この動作モードにより待機状態を確保できます。

SPORT がレシーバー（SPORT_CTL_x.SPTRAN = 0）として設定された場合で、更に SPORT_CTL_x.DIFS = 0 の場合は、受信データ・バッファ・ステータスがフルでないときだけ、受信フレーム同期信号が生成されます。

データ独立のフレーム同期モードでは、バッファ・ステータスとは無関係に、フレーミング信号を連続的に生成できます。SPORT_CTL_x.DIFS を 1 に設定すると、このモードをアクティブにできます。SPORT_CTL_x.DIFS = 1 のとき、送信フレーム同期信号は送信データ・バッファ・ステータスとは無関係に生成され、受信フレーム同期信号も受信データ・バッファ・ステータスとは無関係に生成されます。

注：DMA では通常、送信バッファがフル状態に保たれます。アプリケーションは、送信バッファをデータで満たす必要があります。さもなければ、アンダーフロー／オーバーフローが発生して、フラグが立つ可能性があります。

早期フレーム同期と遅延フレーム同期

フレーム同期信号は、早くすることも遅くすることもできます。シリアル・ポート・コントロール・レジスタの SPORT_CTL_x.LAFS ビットにより、このオプションを設定できます。

デフォルト設定によって、SPORT_CTL_x.LAFS がクリアされているとき、フレーム同期信号は早期フレーミング信号として設定されます。これは通常動作モードです。このモードでは、フレーム同期がアサートされた後、次のシリアル・クロック・サイクルで送信データ・ワードの先頭ビットが有効になります（また、受信データ・ワードの先頭ビットがラッチされます）。ワード全体が送信（または受信）されるまで、フレーム同期が再度チェックされることはありません。

早期フレーミング・モードでデータの送信が連続している（すなわち、各ワードの最終ビットの直後に次のワードの先頭ビットが続く）場合には、各ワードの最終ビットの期間にフレーム同期信号が生成されます。早期フレーミング・モードでは、内部で生成されたフレーム同期は 1 クロック・サイクルの間にアサートされます。

SPORT_CTL_x.LAFS を 1 に設定すると、遅延フレーム同期が設定されます。これはオルタネート動作モードです。このモードでは、フレーム同期がアサートされたのと同じシリアル・クロック・サイクルで送信データ・ワードの先頭ビットが有効になります（また、受信データ・ワードの先頭ビットがラッチされます）。受信データ・ビットは、シリアル・クロックのエッジによってラッチされます。遅延フレーミング・モードでは、内部で生成されたフレーム同期がデータ・ワードの長さ全体にわたってアサートされたままになります。

注：遅延フレーム同期の場合、外部で生成されたフレーム同期は、データ転送のワードの長さ全体にわたってアサートされるようにしてください。さもなければ、予測できない結果が生じる可能性があります。

次の図は、2 つのモードのフレーム信号のタイミングを示しています。

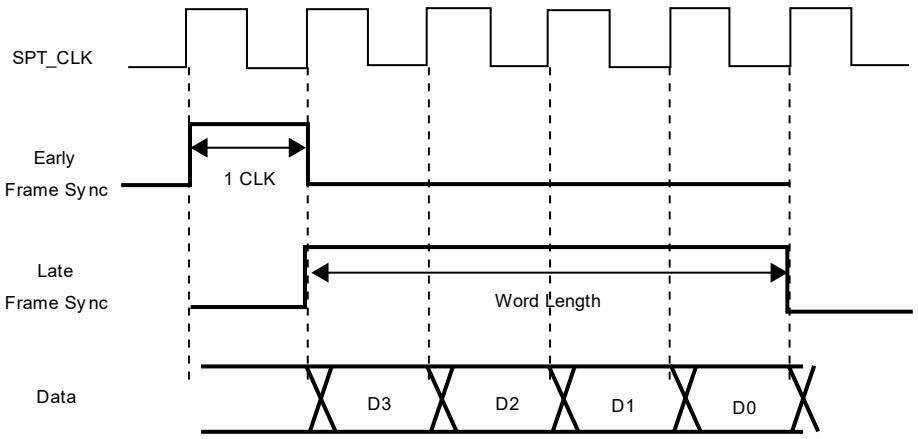


図 16-7：ノーマル・フレーミング（早期フレーム同期）とオルタネート・フレーミング（遅延フレーム同期）

フレーム付きのフレーム同期とフレームなしのフレーム同期

シリアル・ポート通信では、フレーム同期信号の使用はオプションです。SPORT_CTL_x.FSR ビット（SPORT_CTL_A.FSR または SPORT_CTL_B.FSR）は、フレーミング信号が必要かどうかを指定します。

SPORT_CTL_x.FSR ビットを 1 に設定すると、フレーム同期信号はすべてのデータ・ワードで必要とされます。

SPORT_CTL_x.FSR をクリアすると、対応するフレーム同期信号は不要です。通信を開始するのに 1 つのフレーム同期が必要ですが、最初のビットが転送された後は無視されます。その後、データ・ワードはフレームなしモードと呼ばれるモードで連続的に転送されます。フレームなしモードは、連続的な受信／送信に適しています。

次の図は、シリアル・ポート動作でのフレーム付きモードとフレームなしモードを比較したものです。

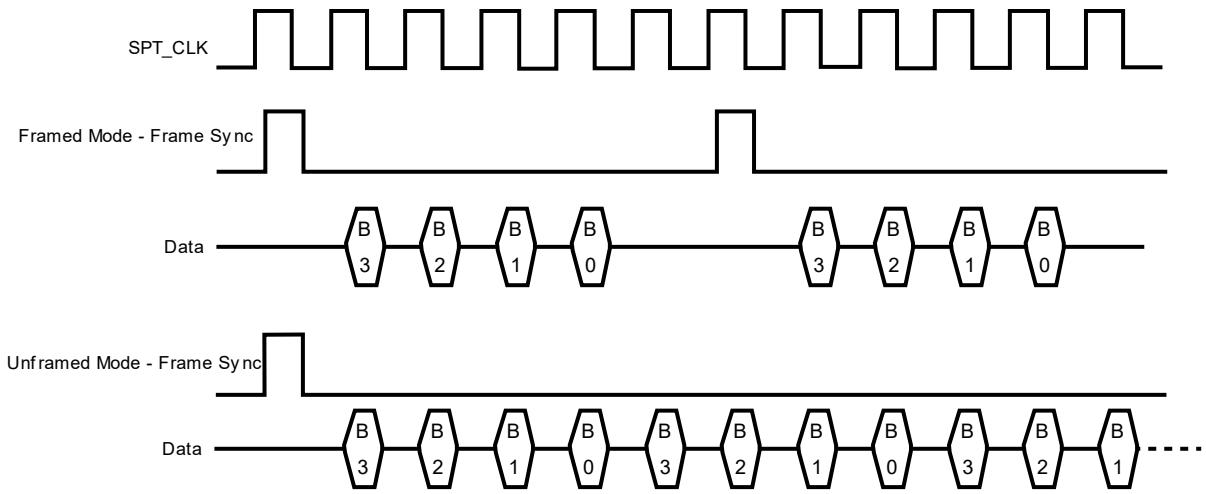


図 16-8：フレーム付きデータ・ストリームとフレームなしデータ・ストリーム

注：フレームなし動作モードでは、SPORT_CTL_x.DIFS ビットを常に 1 に設定する必要があります。フレームなしモードでは、SPORT_CTL_x.DIFS = 0 はサポートされていません。また、SPORT_CTL_x.LAFS ビットを常にゼロに設定する必要があります。フレームなしモードでは、遅延フレーム同期はサポートされていません。

サンプリング・エッジ

シリアル・ポートでは、シリアル・データをサンプリングまたは駆動するために 2 つの制御信号が使用されます。

- シリアル・クロック（SPT_CLK）により、シリアル・データごとにビット・クロックが加えられます。
- フレーム同期（SPT_FS）により、データ・ストリームがフレームに分割されます。

これらの制御信号は、内部で生成することも、外部から供給することも可能で、SPORT_CTL_x.ICLK（SPORT_CTL_A.ICLK または SPORT_CTL_B.ICLK）と SPORT_CTL_x.IFS（SPORT_CTL_A.IFS または SPORT_CTL_B.IFS）ビットの設定値によって指定できます。

データとフレーム同期は、シリアル・ポートのクロック信号の立上がりエッジまたは立下がりエッジでサンプリングできます。SPORT_CTL_x.CKRE（SPORT_CTL_A.CKRE または SPORT_CTL_B.CKRE）ビットは、サンプリング・エッジを制御します。デフォルト設定による SPORT_CTL_x.CKRE = 0 のとき、MCU は受信データと外部フレーム同期のサンプリングに SPT_CLK 信号の立下がりエッジを選択しています。SPORT_CTL_x.CKRE = 1 のとき、受信データとフレーム同期は SPT_CLK の立上がりエッジでサンプリングされます。

送信データと内部フレーム同期信号は、選択されていないシリアル・クロックのエッジで駆動されます（状態が変化します）。SPORT は、デフォルト設定（SPORT_CTL_x.CKRE = 0）では SPT_CLK 信号の立上がりエッジでデータとフレーム同期信号を駆動し、SPORT_CTL_x.CKRE = 1 のときは、立下がりエッジで駆動します。

したがって、相互接続された任意の 2 つのシリアル・ポートの送信と受信の機能では、SPORT_CTL_x.CKRE に対して常に同じ値を設定することを推奨します。これにより、内部で生成された信号は一方のエッジで駆動され、受信された信号は反対のエッジでサンプリングされます。

次の図は、SPORT_CTL_x.CKRE = 0 の場合に、シリアル通信の 2 つのサイドでの代表的な SPORT 信号を示しています。

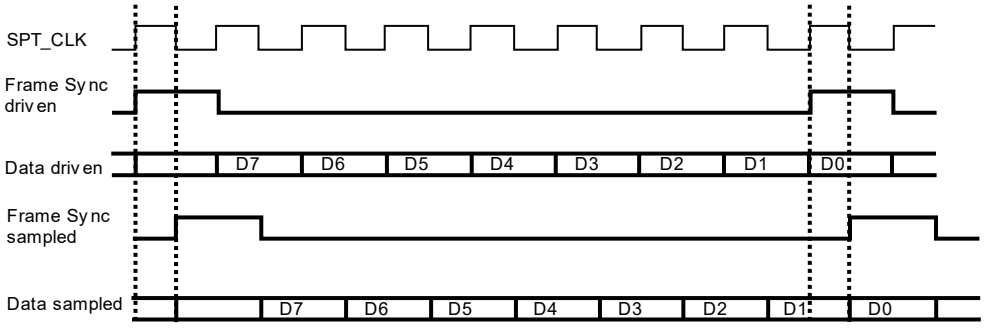


図 16-9：立上がりエッジで駆動されるフレーム同期とデータ

スレーブがフレーム同期信号をサンプリングしたとき、ワード・カウンタには最大値が再ロードされます。各 SPT_CLK は、フレーム全体が受信されるまで、このワード・カウンタをデクリメントします。

したがって、トランスミッタがシリアル・クロックの立上がりエッジで内部フレーム同期とデータを駆動した場合、レシーバーは外部フレーム同期とデータをサンプリングするために、立下がりエッジを使用する必要があります。また、逆の場合も同様です。

早すぎるフレーム同期エラーの検出

SPORT のフレーミング信号は、送信データまたは受信データを同期させるために使用されます。外部 FS モードでは、有効なフレームが続いているときにフレーム同期が早まって受信された場合、または遅延フレーム同期を使用した転送が行われている間にフレーム同期が途切れた場合、これは早すぎるフレーム同期と呼ばれ、無効になります。

早すぎるフレーム同期が受信された場合、SPORT_STAT_x.FSERR ビット（SPORT_STAT_A.FSERR または SPORT_STAT_B.FSERR）にフラグが立てられて、このフレーミング・エラーが表示されます。SPORT_IEN_x.FSERRMSK ビット（SPORT_IEN_A.FSERRMSK と SPORT_IEN_B.FSERRMSK）を 1 に設定することによって、このイベントに対応するオプションのエラー割込みを発生させることができます。

図に示すように、データ転送（送信または受信）中に、早期フレーム同期が生成されたとき、または、遅延フレーム同期の場合にフレーム同期の期間がシリアル・ワード長（SPORT_CTL_A.SLEN または SPORT_CTL_B.SLEN）よりも短い場合に、フレーム同期エラーがトリガされます（これにより、エラー・ビットが 1 に設定されます）。

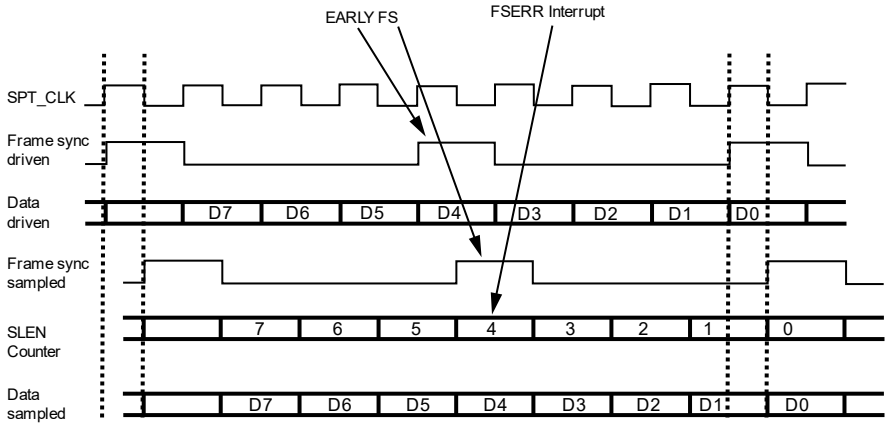


図 16-10：フレーム同期エラーの検出

フレーム同期エラーが発生したとき、受信動作の場合にのみ、SPORT_CTL_x_FSERMODE ビット（SPORT_CTL_A.FSERRMODE または SPORT_CTL_B.FSERRMODE）はこのエラーを処理する方法を指定します。このビットを 1 に設定すると、受信データは廃棄されます。次のフレーム同期が入力されると、新たな受信転送が行われます。

このビットをゼロに設定すると、フレーム同期エラーにフラグが立ちますが、転送は継続します。転送に対しては何も行われません。

シリアル・ワード長

シリアル・ポート・コントロール・レジスタの SPORT_CTL_A.SLEN フィールドまたは SPORT_CTL_B.SLEN フィールドは、送信および受信するためのシリアル・データのワード長を指定します。各ハーフ SPORT は、個別に最大 32 ビットのワード長を処理できます。32 ビット未満のワードは、送信バッファまたは受信バッファにある間、最下位ビット（LSB）の位置に右揃えされます。ただし、データは SPORT_CTL_A.LSBF または SPORT_CTL_B.LSBF ビットの設定値に基づき、MSB ファーストまたは LSB ファーストのフォーマットで、シフト・インまたはシフト・アウトできます。

SPORT_CTL_A.SLEN フィールドまたは SPORT_CTL_B.SLEN フィールドの値は次のように算出できます。

SLEN = シリアル・ポートのワード長 – 1

標準シリアル・モードでの有効なワード長の範囲は、4～32 です。

転送数

SPORT の SPORT_NUMTRAN_A レジスタまたは SPORT_NUMTRAN_B レジスタは、転送ワード数を指定します。ワード長は、SPORT_CTL_A.SLEN フィールドまたは SPORT_CTL_B.SLEN フィールドで指定できます。各ハーフ SPORT はこのフィールドを備えており、SPORT_CTL_A.SPTRAN ビットまたは SPORT_CTL_B.SPTRAN ビットに応じて、受信または送信に使用できます。このフィールドには、送信／受信に必要な転送ワード数をプログラムできます。プログラムされた数の転送が終了すると、SPORT はすべての動作を無効にします。すなわち、クロックは、内部クロック・モードの場合は生成されず、外部クロック・モードの場合はラッチされません。フレーム同期は生成されず、フレーム同期が受信されても何の動作も行われません。

例えば、転送数が 20、ワード長が 20 ビットにプログラムされている場合、400 ビットが転送された後、SPORT はすべての動作を無効にする可能性があります。

このレジスタには、転送数を指定するためのプログラム可能なビットが 12 ビットあります。

次の組の転送を開始するためには、SPORT_NUMTRAN_A レジスタと SPORT_NUMTRAN_B レジスタに必要な転送数を再プログラムする必要があります。同じ転送数をプログラムする必要がある場合は、同じ値を再プログラムする必要があります。

また、外部クロック・モードでは、その数の転送が終了すると、余分なフレーム同期を SPORT に送信する必要はありません。新しく転送する場合は、NUMTRAN が再プログラムされた場合のみ、フレーム同期を送信する必要があります。

注：フレームなしモードでは、次の組の転送を行うために、転送数を再プログラムすることはできません。フレームなしモードの場合に次の組の転送動作を行うには、SPORT をディスエーブルにしてから再度イネーブルにする必要があります。

SPORT の動作モード

ADuCM4050 MCU の SPORT は、次の動作モードをサポートしています。

- 標準シリアル・モード
- タイマー・イネーブル・モード
- ゲーテッド・クロック・モード

SPORT 内の 2 つのハーフ SPORT は、SPMUX ロジックを使用した連結を行わない場合、個別に設定できます。

注：

- これらのモードは、クロックとフレーム同期が両方とも内部であっても外部であってもサポートされています。外部フレーム同期、内部クロック、および内部フレーム同期、外部クロックを用いたモードは、この SPORT ではサポートしていません。フレームなしモードの場合のみ、転送開始のために生成される外部クロック（SPORT_CTL_x.ICLK = 0（SPORT_CTL_A.ICLK または SPORT_CTL_B.ICLK））と内部フレーム同期パルス（SPORT_CTL_x.IFS = 1（SPORT_CTL_A.IFS または SPORT_CTL_B.IFS））がサポートされています。
- SPORT の設定を変更するには（例えば、SPORT_CTL_x.SLEN（SPORT_CTL_A.SLEN または SPORT_CTL_B.SLEN）、または、SPORT_CTL_x.LSBF（SPORT_CTL_A.LSBF または SPORT_CTL_B.LSBF）を変更）、SPORT_CTL_x.SPEN ビット（SPORT_CTL_A.SPEN または SPORT_CTL_B.SPEN）をクリアし、再度セットすることによって再びイネーブルにします。
- SPORT を外部クロック・モードで動作させる場合、SPORT がイネーブルになってから通常動作が開始可能になるまでに、SPORT に 3 つのシリアル・クロックを供給する必要があります。

モード選択

動作モードは、SPORT_CTL_x.OPMODE（SPORT_CTL_A.OPMODE または SPORT_CTL_B.OPMODE）で設定できます。

標準シリアル・モード

SPORT は、SPORT_CTL_x.OPMODE をクリアすることによって、標準の DSP シリアル・モードに設定できます。標準シリアル・モードでは、プログラムによりシリアル・ポートをシリアル・データ・コンバータやオーディオ・コーデックなど、様々なシリアル・デバイスに接続するように設定できます。これらのデバイスに接続するために、様々なクロッキング、フレーミング、およびデータ・フォーマッティングのオプションが使用可能です。これらすべてのオプションは、上記のセクションで信号や様々な機能を示しながら説明されています。いずれも標準シリアル・モードで使用できます。

タイマー・イネーブル・モード

変換プロセスにおいて、2 つの制御信号を必要とする ADC/DAC は、わずかですが存在します。そのようなデバイスとインターフェースするには、追加の信号（SPT_CNV）が必要です。この信号を使用する場合は、SPORT_CTL_x.OPMODE を 1 にプログラムして、タイマー・イネーブル・モードを有効にする必要があります。モジュール内部の PWM タイマーを使用すると、プログラマブルな CNV 信号を生成できます。

SPT_CNV 信号の幅は、SPORT_CNVT_x.WID（SPORT_CNVT_A.WID または SPORT_CNVT_B.WID）フィールドにプログラムできます。SPT_CNV のアサーションとフレーム同期のアサーションの間の遅延は、SPORT_CNVT_x.CNVT2FS（SPORT_CNVT_A.CNVT2FS または SPORT_CNVT_B.CNVT2FS）にプログラムできます。このプログラム機能により、これらの信号を柔軟に使用することができます。次の図は、これらの値を示しています。

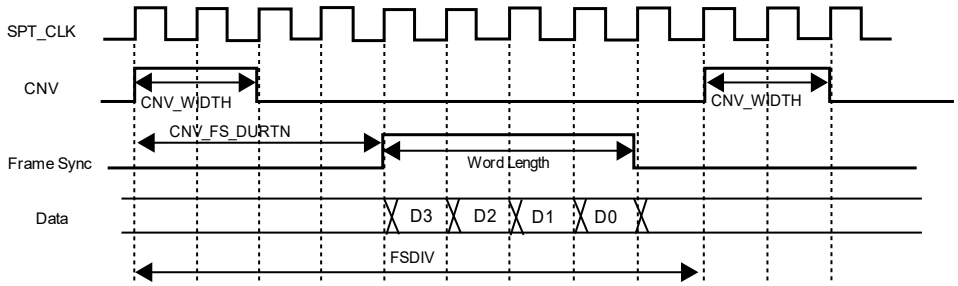


図 16-11：タイマー・イネーブル・モードでインターフェースされる信号

SPT_CNV 信号の極性は、SPORT_CNVT_x.POL（SPORT_CNVT_A.POL または SPORT_CNVT_B.POL）ビット・フィールドにプログラムできます。

CNV パルス間のシリアル・クロックのサイクル数は、FSDIV の値で指定できます。DSP シリアル・モードでのフレーム同期用に規定された他のすべてのプログラマブル・オプションは、以下のオプションを除いて、このモードで使用可能です。

- 外部フレーム同期と外部クロックは使用できません。
- 早期フレーム同期は使用できません。
- フレームなしモードは使用できません。

注：DSP シリアル・モードの使用中、SPT_CNV 信号は使用できません。

ゲーテッド・クロック・モード

ADC/DAC には、インターフェース用に SPI 互換プロトコルをサポートしているものがあります。そのようなコンバータとの通信用に、シリアル・ポートはデータ有効情報がクロックに埋め込まれたゲーテッド・クロック動作モードをサポートしています。したがって、ゲーテッド・クロック・モードでは、有効なデータが送信または受信されているときのみ、クロックはアクティブになります。

SPORT_CTL_x.GCLKEN（ゲーテッド・クロック・モード・セレクト）制御ビットは、シリアル・ポートをゲーテッド・クロック・モードに設定するために使用されます。

ゲーテッド・クロック・モードでは、他の制御ビットに対して次の設定値があります。ゲーテッド・モードが正常に動作するように、それらのビットを適切にプログラムしてください。

- シリアル・クロック信号とフレーム同期信号のどちらも内部生成。
- シリアル・クロック信号とフレーム同期信号のどちらも外部供給。
- フレーム同期は不要（モード SPORT_CTL_A.FSR = 0 はサポートされていません）。

SPORT データ転送

SPORT は、コア駆動または DMA 駆動のデータ転送を使用します。DMA 転送では、シリアル・ポートの送信データ・バッファと内部メモリの間、またはシリアル・ポートの受信データ・バッファと内部メモリの間で、設定可能なシリアル・ワード数を自動的に転送するようにセットアップできます。コア駆動転送では、SPORT 割込みを使用して MCU コアに信号を送り、シリアル・ポートのデータ・バッファと 1 ワード転送のやり取りを行います。

1 ワード（コア）転送

シリアル・ポートでは、各データ・ワードが送信または受信されたときに割込みを発生させることにより、個別のデータ・ワードも送信または受信させることができます。シリアル・ポートがイネーブルで対応する DMA がディスエーブルの場合、完全な 1 ワードが受信データ・バッファに受信されたとき、または、送信データ・バッファがフルでないときは、常に SPORT 割込みが発生します。

シリアル・ポートのデータ・パッキングが有効な場合、送信割込みと受信割込みが 32 ビットにパッキングされたワードでは発生しますが、16 ビット・ワードごと、および 8 ビット・ワードごとには発生しません。コア駆動転送を行う場合、SPORT_CTL_A.SPTRAN（ハーフ SPORT が A の場合）ビットの設定値で指定されたバッファに書込みを行います。不測の動作を避けるため、MCU コアがシリアル・ポートの受信バッファから 1 ワードを読み出そうとするとき、または送信バッファに 1 ワードを書き込もうとするときは、対応するデータ・バッファのステータスをチェックします。SPORT_STAT_x.DXS（SPORT_STAT_A.DXS または SPORT_STAT_B.DXS）ビットを使用することによって、フル／エンプティ・ステータスを読み出すことができます。

DMA 転送

ダイレクト・メモリ・アクセス（DMA）では、シリアル・データのブロック全体を受信または送信するためのメカニズムが実行されてから、割込みが発生します。SPORT DMA がイネーブルでない場合、SPORT はデータ・ワードを受信するたび、またはデータ・ワードを送信し始めるたびに割込みを発生させます。MCU に内蔵されている DMA コントローラが DMA 転送の処理を行うため、データのブロック全体が送信または受信されるまで、MCU コアはスリープするか、または他のタスクを実行し続けることができます。したがって、サービス・ルーチンは、1 ワードごとではなくデータのブロックごとで動作できるため、オーバーヘッドを大幅に低減できます。

このため、方向ビット、DMA イネーブル・ビット、およびシリアル・ポート・イネーブル・ビットを設定してから SPORT データ・バッファで任意の動作を開始します。シリアル・ポートに関連する DMA チャンネルがイネーブルの場合、データ・バッファのアクセスは試みないでください。各ハーフ SPORT は、Tx データ・パスと Rx データ・パスに対して専用の DMA チャンネルを備えています。DMA コントローラは、送信モードでは送信データ・バッファに書込みを行います。同様に、受信モードでは受信データ・バッファから読出しを行います。DMA コントローラは、DMA 転送の終了時に割込みを発生させます。

SPORT は、（SPORT_CTL_x.SLEN（SPORT_CTL_A.SLEN または SPORT_CTL_B.SLEN）フィールドで規定されるように）4 ビット～32 ビットのワード・サイズを処理できます。シリアル・データ長が 16 ビット以下の場合、2 つのデータ・バイトは DMA 転送ごとに 32 ビット・ワードにパッキングできます。シリアル・データ長が 8 ビット以下の場合、4 つのデータは DMA 転送ごとに 32 ビット・ワードにパッキングできます。シリアル・ポートのデータ・パッキングが有効なとき、送信 DMA リクエストと受信 DMA リクエストは 32 ビットにパッキングされたワードでは発生しますが、16 ビット・ワードごと、および 8 ビット・ワードごとには発生しません。

パッキングが有効かどうかに応じて、対応する DMA 転送幅を DMA コントローラで選択する必要があります。パッキングが有効な場合、または 16 ビットを超える転送が必要な場合は、DMA でワード転送を行います。パッキングが有効でない場合で、かつ転送が 16 ビット未満で 8 ビットを超える場合は、ハーフ・ワード転送を行います。パッキングが有効でない場合で、かつ転送長が 8 ビット未満の場合は、DMA コントローラ内部でバイト転送を行います。

注：連続的な送信／受信の場合に、連続的なデータ・フローを確保できるほど頻繁に DMA リクエストが処理されない場合があります。このような転送を行うときは、DMA チャンネルの優先順位が最も高いことを確認してください。

SPORT のデータ・バッファ

データ・バッファ・ステータス

SPORT は、データ・バッファのステータス情報を備えています。SPORT_CTL_A.SPTRAN ビットの設定値に応じて、SPORT_STAT_x.DXS ビットには送信データ・バッファまたは受信データ・バッファのステータスが反映されます。これらのビットは、バッファがフルか、部分的にフルか、またはエンプティかどうかを示します。

不測の状態を避けるため、バッファ・ステータスを常にチェックして、アクセスできるかどうかを判断してください。SPORT_STAT_A.DXS フィールドのステータス・ビットは、常に FIFO のステータスを示します。

4 番目のワードがシフト・インされている間に、3 つの完全な 32 ビット・ワードを受信バッファに格納することができます。したがって、オーバーフローが発生する前に、受信バッファから読み出すことなしに 4 つの完全なワードを受信できます。4 番目のワードがすべて受信された後、（MCU コアまたは DMA コントローラによって）最初のワードが読み出されなかった場合、シフト・レジスタの内容が 3 番目のワードに上書きされます。この場合、SPORT_STAT_A.DERR レジスタのエラー・ステータス・ビットを通じて、受信オーバーフロー・ステータスにフラグが立ちます。4 番目のワードの最終ビットが受信された後、オーバーフロー・ステータスが発生します。

データ・バッファ・パッキング

SPORT が 16 ビット以下、または更に 8 ビット以下の長さのシリアル・データ・ワードを受信するレシーバーとして設定されると、受信したデータ・ワードを 32 ビット・ワードにパッキングすることができます。同様に、SPORT が 16 ビット以下の長さのシリアル・データ・ワードを送信するトランスミッタとして設定されると、送信する 32 ビット・ワードを 2 つの 16 ビット・ワードまたは 4 つの 8 ビット・ワードにアンパッキングすることができます。この機能は、SPORT_CTL_x.PACK（SPORT_CTL_A.PACK または SPORT_CTL_B.PACK）ビットで選択できます。

注：長さの短いデータの転送でパッキングが有効でない場合は、バスの帯域幅も FIFO バッファも効率的に使用することができません。

SPORT_CTL_x.PACK = 01 のとき、受信された連続する 4 ワードが 1 つの 32 ビット・ワードにパッキングされたり、各 32 ビット・ワードがアンパッキングされて 4 つの 8 ビット・ワードとして送信されたりします。8 ビット未満のワードは、［SLEN：0］、［（SLEN+8）：8］、［（SLEN+16）：16］、および［（SLEN+24）：24］のようにパッキングされます。これは、受信（パッキング）動作と送信（アンパッキング）動作の両方に適用されます。

次の図は、8 ビット・パッキングを示しています。濃い範囲がパッキングされたビットに相当します。

図 16-12：パッキングされたデータ（8 ビット・パッキング）

SPORT_CTL_x.PACK を 10 に指定したとき、受信された連続する 2 ワードが 1 つの 32 ビット・ワードにパッキングされたり、各 32 ビット・ワードがアンパッキングされて 2 つの 16 ビット・ワードとして送信されたりします。16 ビ

ット未満のワードは、［SLEN：0］と［（SLEN+16）：16］のようにパッキングされます。これは、受信（パッキング）動作と送信（アンパッキング）動作の両方に適用されます。

次の図は、16 ビット・パッキングを示しています。

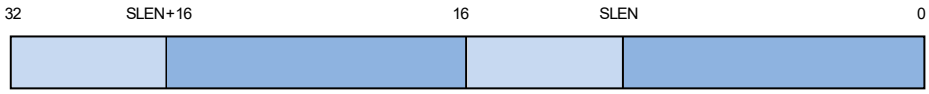


図 16-13：パッキングされたデータ（16 ビット・パッキング）

パッキングにおいて、送信割込みと受信割込みは 32 ビットにパッキングされたワードには発生しますが、16 ビット・ワードごと、および 8 ビット・ワードごとには発生しません。パッキング機能により、システムの帯域幅を有効に利用できます。

注：パッキングが有効な場合、ワード長はプログラムされたパッキングよりも短く設定する必要があります。例えば、8 ビット・パッキングに設定する場合、SLEN は 7 以下にする必要があり、また、16 ビット・パッキングに設定する場合、SLEN は 15 以下にする必要があります。

SPORT 割込みと例外

各ハーフ SPORT は、関連する割込みを備えています。割込み源を特定するために、アプリケーションは割込みステータス・レジスタをチェックする必要があります。ここでは、様々な割込み源について説明します。コア・モードでは、SPORT は受信動作または送信動作においてデータ割込みを発生させることができます。更に、SPORT モジュールはエラー状態を生成することができ、割込み源ごとに個別のステータス・ビットを備えています。

注：割込み信号をクリアするには、対応するステータス・ビットに 1 を書き込みます。これにより、ステータス・ビットの値がクリアされます。すなわち、対応するステータス・ビットはゼロになります。

エラー検出

SPORT がトランスミッタとして設定されている場合、SPORT_STAT_A.DERR ビットまたは SPORT_STAT_B.DERR ビットは、データ・パスでの送信データ・バッファのアンダーフロー・ステータスを表示します（これは送信データ・バッファがエンプティであったとき、フレーム同期信号が生成されたことを示します）。SPORT は、フレーミング信号を検出するたびにデータを送信します。

同様に、SPORT がレシーバーとして設定されている場合、SPORT_STAT_A.DERR ビットまたは SPORT_STAT_B.DERR ビットは、受信データ・バッファの受信オーバーフロー・ステータスを表示します。すなわち、SPORT は受信バッファがフルのとき、1 つのチャンネルが新しいデータを受信して、新しいデータが既存のデータに上書きされたことを示します。SPORT はフレーミング信号を検出すると、データを受信します。

外部フレーム同期の場合、または DIFS（データ独立のフレーム同期）が 1 に設定されている場合に、これらのエラー状態が発生します。

SPORT_IEN_x.DERRMSK（SPORT_IEN_A.DERRMSK または SPORT_IEN_B.DERRMSK）（データ・エラー割込みイネーブル）ビットは、データ・エラーに対応するステータス割込みのマスクを解除するために使用できます。

データ・アンダーフロー・エラーとデータ・オーバーフロー・エラーに加えて、フレーム同期エラーが検出されると、オプションとしてステータス割込みもトリガされます。SPORT_IEN_x.FSERRMSK（SPORT_IEN_A.FSERRMSK または SPORT_IEN_B.FSERRMSK）（フレーム同期エラー割込みイネーブル）ビットは、このフレーム同期エラーに対応

するステータス割込みのマスクを解除します。早すぎるフレーム同期が原因で、このフレーム同期エラーが発生します。これについての詳細な情報は、早すぎるフレーム同期エラーの検出を参照してください。

注：有効なデータの送信／受信が行われなかったり、入力信号内のノイズによってフレーム同期パルスが生成されたりしたときは、フレーム同期エラーは検出されません。

システム転送割込み

SPORT が送信用に設定されている場合、コアがフル状態の送信 FIFO に書込みを試みると、割込みが発生します。SPORT_STAT_A.SYSDATERR ステータスまたは SPORT_STAT_B.SYSDATERR ステータスは、この送信 FIFO オーバーフロー・エラーを表示します。

同様に、SPORT が受信用に設定されている場合、コアがエンプティ状態の受信 FIFO から読出しを試みると、割込みが発生します。この場合、SPORT_STAT_A.SYSDATERR ステータス・フィールドまたは SPORT_STAT_B.SYSDATERR ステータス・フィールドは、受信 FIFO アンダーフロー・エラーを表示します。

注：DMA モードでは、送信 FIFO がフル、または受信 FIFO がエンプティの場合、SPORT は DMA リクエストを発生させません。DMA 転送の間は、割込みは発生しません。

転送終了割込み（TFI）

転送終了割込み機能は、送信／受信終了の信号を送るために使用されます。この機能は、SPORT_IEN_A ビットまたは SPORT_IEN_B ビットを 1 に設定することによって有効にできます。SPORT_NUMTRAN_A フィールドまたは SPORT_NUMTRAN_B フィールドにプログラムされた数の転送が終了すると、SPORT は FIFO 内（送信シフト・レジスタも含む）のすべてのデータが送出される、または受信レジスタ内に完全に受信されるまで待ち、転送終了割込みを発生させます。この機能を用いると、SPORT がプログラムされた転送数に対応するデータをすべて送出したか、または受信したかの確認に対応するすべてのデータを特定することができます。

注：転送終了割込みが発生すると、（必要に応じて）転送数を再プログラムしたり、SPORT をディスエーブルにして再度イネーブルにしたりすることもできます。転送数を再プログラムするには、転送数を参照してください。

SPORT を再度イネーブルにするには、SPORT プログラミング・モデルを参照してください。

SPORT プログラミング・モデル

SPORT のプログラミング・ガイドラインを次に示します。

- SPORT のすべてのレジスタに必要な設定値を書き込みます。ただし、コントロール・レジスタを除きます。
- 設定の後、コントロール・レジスタ（SPORT_CTL_A または SPORT_CTL_B）に転送に必要な設定値を書き込みます。SPORT_CTL_A.SPEN フィールドまたは SPORT_CTL_B.SPEN フィールドは 1 に設定する必要があります。
- SPORT_CTL_A.SPEN フィールドまたは SPORT_CTL_B.SPEN フィールドに 1 を書き込むと、プログラムされた設定値に基づいて SPORT は通常動作を開始できます。

設定値を書き込む必要のある SPORT レジスタを以下に示します。

- SPORT_DIV_A または SPORT_DIV_B に書込みを行って、必要な CLKDIV 値と FSDIV 値をプログラムします。

- SPORT_CNVT_A レジスタまたは SPORT_CNVT_B レジスタには、タイマー・イネーブル・モードでのみ書き込む必要があります。
- IEN レジスタには、割込みに基づいて書込みを行います。例えば、アンダーフロー・エラー／オーバーフロー・エラーに対する割込みが必要であれば、SPORT_IEN_A または SPORT_IEN_B を 1 に設定する必要があります。
- SPORT_NUMTRAN_A または SPORT_NUMTRAN_B に必要な転送数を書き込みます。このレジスタにはゼロは設定しないでください。

SPORT のパワー・マネージメント

チップが休止モードに入った場合でも、設定値は保持されます。次のレジスタが保持されます。

- SPORT_CTL_A（DMAEN ビット・フィールドと SPEN ビット・フィールドを除く）
- SPORT_DIV_A
- SPORT_CNVT_A
- SPORT_CTL_B（DMAEN ビット・フィールドと SPEN ビット・フィールドを除く）
- SPORT_DIV_B
- SPORT_CNVT_B

注：チップが休止モードに入る前に転送が行われていても、その転送は休止からの復帰後に再開されません。SPORT が休止から復帰したら、新たな転送を始めから行う必要があります。SPORT_NUMTRAN_A レジスタまたは SPORT_NUMTRAN_B レジスタに適切にプログラムします。次に SPORT_CTL_A.SPEN ビットまたは SPORT_CTL_B.SPEN ビットをプログラムして SPORT 動作を有効にします。

ADuCM4050 SPORT レジスタの説明

シリアル・ポート（SPORT）には、次のレジスタがあります。

表 16-2：ADuCM4050 SPORT レジスター一覧

レジスタ名	説明
SPORT_CTL_A	ハーフ SPORT'A'コントロール・レジスタ
SPORT_CTL_B	ハーフ SPORT 'B'コントロール・レジスタ
SPORT_DIV_A	ハーフ SPORT 'A'除数レジスタ
SPORT_DIV_B	ハーフ SPORT 'B'除数レジスタ
SPORT_IEN_A	ハーフ SPORT A 割込みイネーブル・レジスタ
SPORT_IEN_B	ハーフ SPORT B 割込みイネーブル・レジスタ
SPORT_NUMTRAN_A	ハーフ SPORT A 転送数レジスタ
SPORT_NUMTRAN_B	ハーフ SPORT B 転送数レジスタ
SPORT_RX_A	ハーフ SPORT 'A' Rx バッファ・レジスタ

表 16-2：ADuCM4050 SPORT レジスター一覧（続き）

レジスタ名	説明
SPORT_RX_B	ハーフ SPORT 'B' Rx バッファ・レジスタ
SPORT_STAT_A	ハーフ SPORT 'A'ステータス・レジスタ
SPORT_STAT_B	ハーフ SPORT 'B'ステータス・レジスタ
SPORT_CNVT_A	ハーフ SPORT 'A' CNV 幅レジスタ
SPORT_CNVT_B	ハーフ SPORT 'B' CNV 幅レジスタ
SPORT_TX_A	ハーフ SPORT 'A' Tx バッファ・レジスタ
SPORT_TX_B	ハーフ SPORT 'B' Tx バッファ・レジスタ

ハーフ SPORT 'A'コントロール・レジスタ

SPORT_CTL_A には、シリアル・ポートのモードを選択するなどの、ハーフ SPORT 'A'用の送信コントロール・ビットと受信コントロール・ビットがあります。

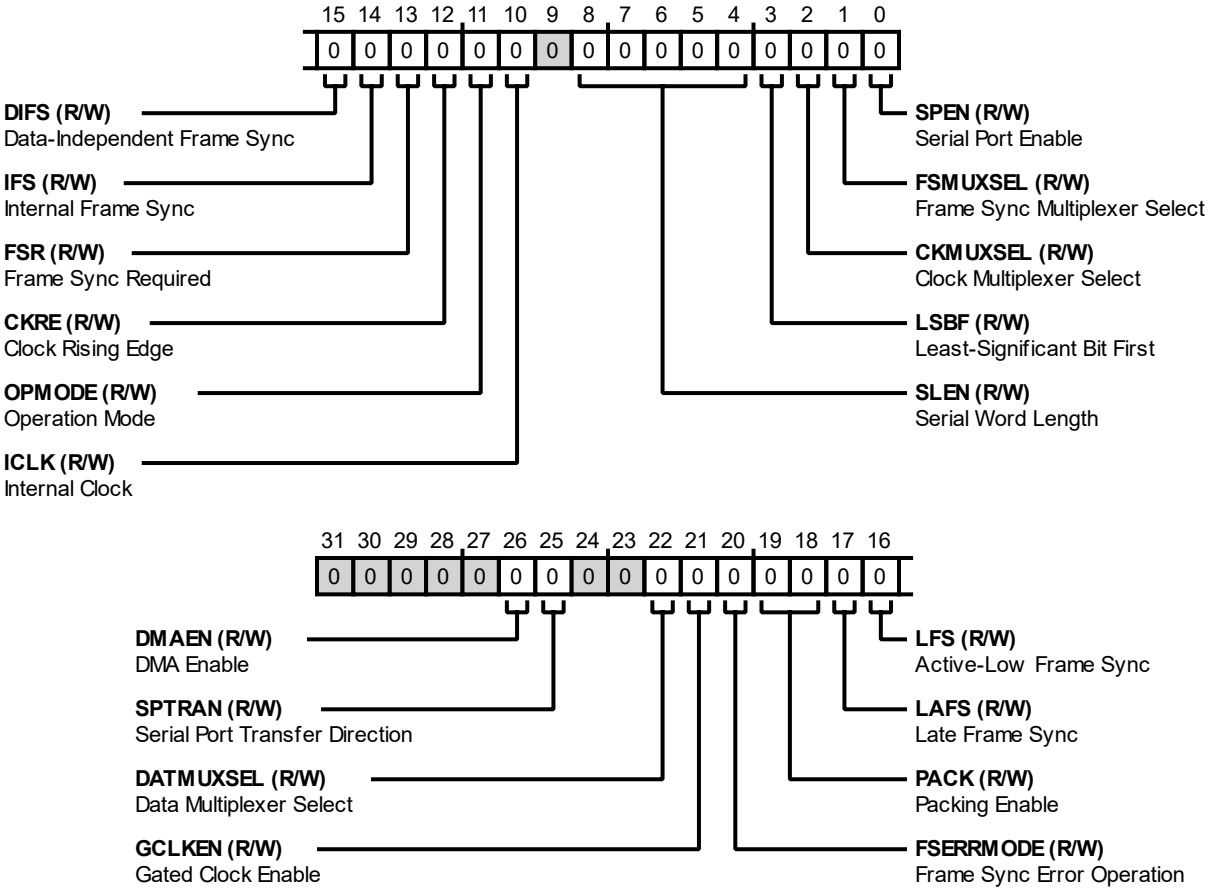


図 16-14：SPORT_CTL_A レジスタ図

表 16-3：SPORT_CTL_A レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
26 (R/W)	DMAEN	DMA イネーブル。 送信 FIFO がデータを必要とするとき、または受信 FIFO がデータを DMA に送信する必要があるときに、このビットはハーフ SPORT が DMA リクエスト信号を送信するかどうかを指定します。	
		0	DMA リクエストを無効にします
		1	DMA リクエストを有効にします

表 16-3：SPORT_CTL_A レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
25 (R/W)	SPTRAN	シリアル・ポートの転送方向。 SPORT_CTL_A.SPTRAN ビットは、ハーフ SPORT のチャンネルにおける転送方向（受信または送信）を選択します。	
		0	受信
		1	送信
22 (R/W)	DATMUXSEL	データ・マルチプレクサ・セレクト。 このビットを設定すると、データ・マルチプレクサを使用して、データをハーフ SPORT B からハーフ SPORT A にルーティングできます。	
		0	データ・マルチプレクサ・セレクトを無効にします。
		1	データ・マルチプレクサ・セレクトを有効にします。
21 (R/W)	GCLKEN	ゲーテッド・クロック・イネーブル。 SPORT_CTL_A.GCLKEN ビットは、ハーフ SPORT でのゲーテッド・クロック動作をイネーブルにします。 SPORT_CTL_A.GCLKEN をイネーブルにすると、SPORT がデータを転送しているとき、またはフレーム同期が変化（アクティブ状態に遷移）するときに、SPORT クロックがアクティブになります。	
		0	ディスエーブルにします
		1	イネーブルにします
20 (R/W)	FSERRMODE	フレーム同期エラー時の動作。 SPORT_CTL_A.FSERRMODE ビットは、フレーム同期エラーが発生したときに SPORT が応答する方法を指定します。	
		0	フレーム同期エラーのフラグを立てて通常動作を続けます
		1	フレーム同期エラーが発生したとき、受信データを廃棄します
19:18 (R/W)	PACK	パッキング・イネーブル。 SPORT_CTL_A.PACK ビットは、ハーフ SPORT が受信データ上で 16 ビットを 32 ビットに、または 8 ビットを 32 ビットにパッキングすることを有効にし、また、送信データ上で 32 ビットを 16 ビットに、または 32 ビットを 8 ビットにアンパッキングすることを有効にします。	
		0	無効にします
		1	8 ビット・パッキングを有効にします
		2	16 ビット・パッキングを有効にします
		3	予備

表 16-3：SPORT_CTL_A レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
17 (R/W)	LAFS	遅延フレーム同期。 SPORT_CTL_A.LAFS ビットは、ハーフ SPORT が遅延フレーム同期信号（最初のデータ・ビットの期間に SPORT_AFS）を生成するか、または、早期フレーム同期信号（最初のデータ・ビットの前のシリアル・クロック・サイクル期間に SPORT_AFS）を生成するかを選択します。	
		0	早期フレーム同期
		1	遅延フレーム同期
16 (R/W)	LFS	アクティブ・ローのフレーム同期。 SPORT_CTL_A.LFS ビットは、ハーフ SPORT がアクティブ・ローのフレーム同期を使用するか、アクティブ・ハイのフレーム同期を使用するかを選択します。	
		0	アクティブ・ハイのフレーム同期
		1	アクティブ・ローのフレーム同期
15 (R/W)	DIFS	データ独立のフレーム同期。 SPORT_CTL_A.DIFS ビットは、ハーフ SPORT がデータ独立のフレーム同期を使用するか、データ依存のフレーム同期を使用するかを選択します。データ独立のフレーム同期を使用すると、ハーフ SPORT は SPORT_DIV_A.FSDIV で選択された間隔で同期を生成します。データ依存のフレーム同期を使用すると、ハーフ SPORT は送信バッファがエンプティでないとき、または、受信バッファがフルでないときに選択された間隔で同期を生成します。	
		0	データ依存のフレーム同期
		1	データ独立のフレーム同期
14 (R/W)	IFS	内部フレーム同期。 SPORT_CTL_A.IFS ビットは、ハーフ SPORT が内部フレーム同期を使用するか、外部フレーム同期を使用するかを選択します。 外部で生成されたフレーム同期は、プロセッサのシステム・クロックと同期する必要はありません。	
		0	外部フレーム同期
		1	内部フレーム同期
13 (R/W)	FSR	フレーム同期の要求。 SPORT_CTL_A.FSR は、ハーフ SPORT がデータを転送するためにフレーム同期を必要とするかどうかを選択します。	
		0	フレーム同期は不要です
		1	フレーム同期が必要です

表 16-3：SPORT_CTL_A レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
12 (R/W)	CKRE	クロックの立上がりエッジ。 SPORT_CTL_A.CKRE は、ハーフ SPORT が受信データとフレーム同期をサンプリングするために、SPORT_ACLK クロックの立上がりエッジを使用するか、立下がりエッジを使用するかを選択します。	
		0	クロックの立下がりエッジ
		1	クロックの立上がりエッジ
11 (R/W)	OPMODE	動作モード。 SPORT_CTL_A.OPMODE ビットは、ハーフ SPORT が DSP 標準モードで動作するか、タイマー・イネーブル・モードで動作するかを選択します。	
		0	DSP 標準モード
		1	タイマー・イネーブル・モード
10 (R/W)	ICLK	内部クロック。 SPORT_CTL_A.ICLK ビットは、ハーフ SPORT が内部クロックを使用するか、外部クロックを使用するかを選択します。	
		0	外部クロック
		1	内部クロック
8:4 (R/W)	SLEN	シリアル・ワード長。 SPORT_CTL_A.SLEN ビットは、ハーフ SPORT がデータを転送するときのビット単位のワード長を選択します。ワード長は、4 ビット～32 ビットにすることができます。ビット単位のワード長を選択するための式は、次のようになります。 SPORT_CTL_A.SLEN = （ビット単位のシリアル・ワード長） – 1	
3 (R/W)	LSBF	最下位ビット・ファースト。 SPORT_CTL_A.LSBF ビットは、ハーフ SPORT がデータを送信または受信するときに、LSB ファーストで行うか、MSB ファーストで行うかを選択します。	
		0	MSB ファーストで送信または受信します
		1	LSB ファーストで送信または受信します
2 (R/W)	CKMUXSEL	クロック・マルチプレクサ・セレクト。 SPORT_CTL_A.CKMUXSEL ビットは、ハーフ SPORT のシリアル・クロックのマルチプレクスを有効にします。このモードでは、ハーフ SPORT 自身のシリアル・クロックの代わりに、関連するハーフ SPORT のシリアル・クロックが使用されます。例えば、SPORT_CTL_A.CKMUXSEL がイネーブルの場合、ハーフ SPORT 'A'は SPORT_ACLK の代わりに、SPORT_BCLK を使用します。	
		0	シリアル・クロックのマルチプレクスを無効にします
		1	シリアル・クロックのマルチプレクスを有効にします

表 16-3：SPORT_CTL_A レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
1 (R/W)	FSMUXSEL	フレーム同期マルチプレクサ・セレクト。 SPORT_CTL_A.FSMUXSEL ビットは、ハーフ SPORT のフレーム同期のマルチプレクスを有効にします。このモードでは、ハーフ SPORT 自身のフレーム同期の代わりに、関連するハーフ SPORT のフレーム同期が使用されます。例えば、SPORT_CTL_A.FSMUXSEL がイネーブルの場合、ハーフ SPORT 'A'は SPORT_AFS の代わりに、SPORT_BFS を使用します。	
		0	フレーム同期のマルチプレクスを無効にします
		1	フレーム同期のマルチプレクスを有効にします
0 (R/W)	SPEN	シリアル・ポート・イネーブル。 SPORT_CTL_A.SPEN ビットは、ハーフ SPORT のデータ・チャンネルをイネーブルにします。注：このビットをクリアする（= 1 から= 0 に変更する）と、ハーフ SPORT はチャンネルのデータ・バッファを自動的に消去して、SPORT 内のクロック、フレーム同期、およびカウンタをディスエーブルにします。	
		0	ディスエーブルにします
		1	イネーブルにします

ハーフ SPORT 'B'コントロール・レジスタ

SPORT_CTL_B には、チャンネルにおけるシリアル・ポートのモードを選択するなどの、ハーフ SPORT 'B'用の送信コントロール・ビットと受信コントロール・ビットがあります。SPORT の動作モードに応じて、SPORT_CTL_B の一部のビットの機能が変わります。詳細については、SPORT 動作モードの説明を参照してください。予備ビットを読み出した場合、読み出された値はこれらのビットに最後に書き込んだ値か、またはこれらのビットをリセットした値になります。

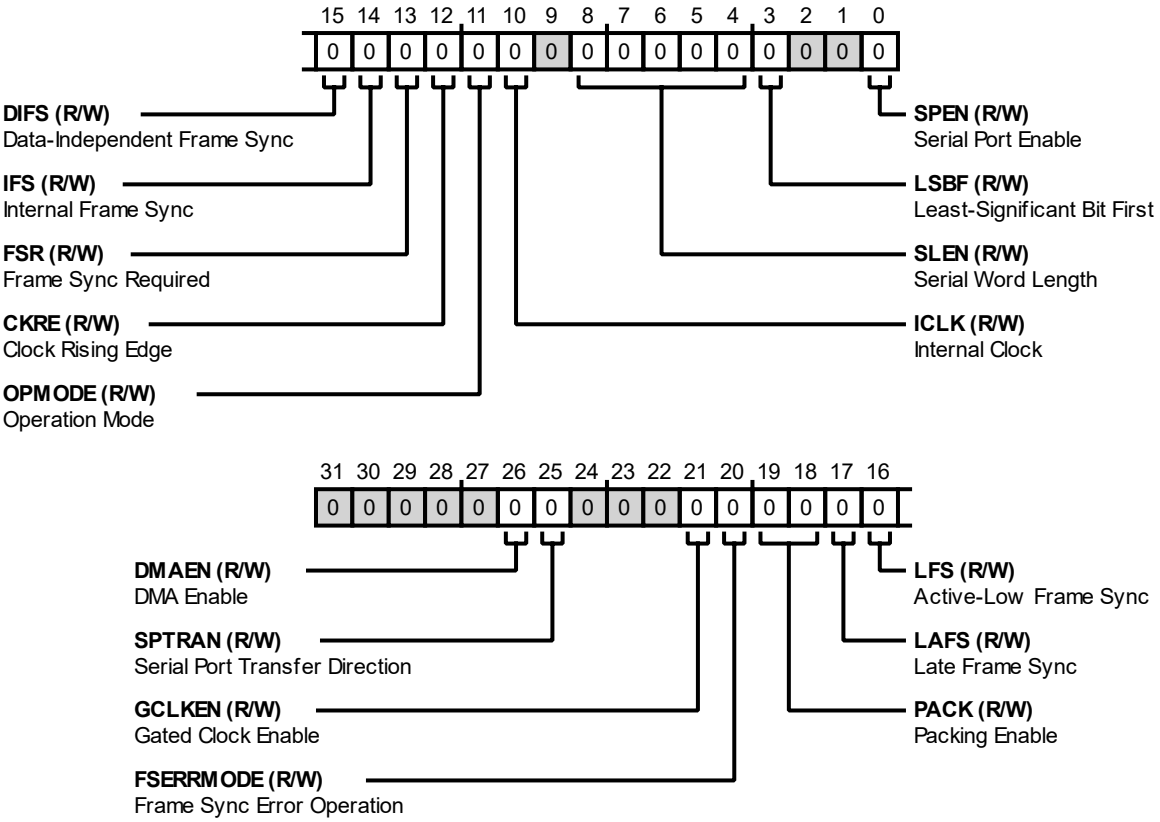


図 16-15：SPORT_CTL_B レジスタ図

表 16-4：SPORT_CTL_B レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
26 (R/W)	DMAEN	DMA イネーブル。 送信 FIFO がデータを必要とするとき、または受信 FIFO がデータを DMA に送信する必要があるときに、このビットはハーフ SPORT が DMA リクエスト信号を送信するかどうかを指定します。	
		0	DMA リクエストを無効にします
		1	DMA リクエストを有効にします

表 16-4：SPORT_CTL_B レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
25 (R/W)	SPTRAN	シリアル・ポートの転送方向。 SPORT_CTL_B.SPTRAN ビットは、ハーフ SPORT のチャンネルにおける転送方向（受信または送信）を選択します。	
		0	受信
		1	送信
21 (R/W)	GCLKEN	ゲーテッド・クロック・イネーブル。 SPORT_CTL_B.GCLKEN ビットは、DSP シリアル・モードのときにハーフ SPORT でのゲーテッド・クロック動作をイネーブルにします。 SPORT_CTL_B.GCLKEN をイネーブルにすると、SPORT がデータを転送しているとき、またはフレーム同期が変化（アクティブ状態に遷移）するときに、SPORT クロックがアクティブになります。	
		0	ディスエーブルにします
		1	イネーブルにします
20 (R/W)	FSERRMODE	フレーム同期エラー時の動作。 SPORT_CTL_B.FSERRMODE ビットは、フレーム同期エラーが発生したときに SPORT が応答する方法を指定します。	
		0	フレーム同期エラーのフラグを立てて通常動作を続けます
		1	フレーム同期エラーが発生したとき、受信データを廃棄します
19:18 (R/W)	PACK	パッキング・イネーブル。 SPORT_CTL_B.PACK ビットは、ハーフ SPORT が受信データ上で 16 ビットを 32 ビットに、または 8 ビットを 32 ビットにパッキングすることを有効にし、また、送信データ上で 32 ビットを 16 ビットに、または 32 ビットを 8 ビットにアンパッキングすることを有効にします。	
		0	無効にします
		1	8 ビット・パッキングを有効にします
		2	16 ビット・パッキングを有効にします
		3	予備
17 (R/W)	LAFS	遅延フレーム同期。 ハーフ SPORT が DSP 標準モード（SPORT_CTL_B.OPMODE = 0）の場合、SPORT_CTL_B.LAFS ビットは、ハーフ SPORT が遅延フレーム同期信号（最初のデータ・ビットの期間に SPORT_BFS）を生成するか、または、早期フレーム同期信号（最初のデータ・ビットの前のシリアル・クロック・サイクル期間に SPORT_BFS）を生成するかを選択します。	
		0	早期フレーム同期
		1	遅延フレーム同期

表 16-4：SPORT_CTL_B レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
16 (R/W)	LFS	アクティブ・ローのフレーム同期。 ハーフ SPORT が DSP 標準モード（SPORT_CTL_B.OPMODE = 0）の場合、SPORT_CTL_B.LFS ビットは、ハーフ SPORT がアクティブ・ローのフレーム同期を使用するか、アクティブ・ハイのフレーム同期を使用するかを選択します。	
		0	アクティブ・ハイのフレーム同期
		1	アクティブ・ローのフレーム同期
15 (R/W)	DIFS	データ独立のフレーム同期。 SPORT_CTL_B.DIFS ビットは、ハーフ SPORT がデータ独立のフレーム同期を使用するか、データ依存のフレーム同期を使用するかを選択します。データ独立のフレーム同期を使用すると、ハーフ SPORT は SPORT_DIV_B.FSDIV によって選択された間隔で同期を生成します。データ依存のフレーム同期を使用すると、ハーフ SPORT は送信バッファがエンプティでないとき、または、受信バッファがフルでないときに選択された間隔で同期を生成します。	
		0	データ依存のフレーム同期
		1	データ独立のフレーム同期
14 (R/W)	IFS	内部フレーム同期。 SPORT_CTL_B.IFS ビットは、ハーフ SPORT が内部フレーム同期を使用するか、外部フレーム同期を使用するかを選択します。 外部で生成されたフレーム同期は、プロセッサのシステム・クロックと同期する必要はありません。	
		0	外部フレーム同期
		1	内部フレーム同期
13 (R/W)	FSR	フレーム同期の要求。 SPORT_CTL_B.FSR は、ハーフ SPORT がデータを転送するためにフレーム同期を必要とするかどうかを選択します。	
		0	フレーム同期は不要です
		1	フレーム同期が必要です
12 (R/W)	CKRE	クロックの立上がりエッジ。 SPORT_CTL_B.CKRE は、ハーフ SPORT が受信データとフレーム同期をサンプリングするために、SPORT_BCLK クロックの立上がりエッジを使用するか、立下がりエッジを使用するかを選択します。	
		0	クロックの立下がりエッジ
		1	クロックの立上がりエッジ

表 16-4：SPORT_CTL_B レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
11 (R/W)	OPMODE	動作モード。 SPORT_CTL_B.OPMODE ビットは、ハーフ SPORT が DSP 標準モードで動作するか、タイマー・イネーブル・モードで動作するかを選択します。	
		0	DSP 標準モード
		1	タイマー・イネーブル・モード
10 (R/W)	ICLK	内部クロック。 ハーフ SPORT が DSP 標準モード（SPORT_CTL_B.OPMODE = 0）の場合、SPORT_CTL_B.ICLK ビットは、ハーフ SPORT が内部クロックを使用するか、外部クロックを使用するかを選択します。内部クロックがイネーブルの場合は、ハーフ SPORT が SPORT_BCLK クロック信号を生成するため、SPORT_BCLK が出力になります。SPORT_DIV_B.CLKDIV シリアル・クロックの除数値は、クロック周波数を指定します。内部クロックがディスエーブルの場合は、SPORT_BCLK クロック信号が入力となり、シリアル・クロックの除数値は無視されます。外部で生成されたシリアル・クロックは、プロセッサのシステム・クロックと同期する必要はありません。	
		0	外部クロック
		1	内部クロック
8:4 (R/W)	SLEN	シリアル・ワード長。 SPORT_CTL_B.SLEN ビットは、ハーフ SPORT がデータを転送するときのビット単位のワード長を選択します。ワード長は、4 ビット～32 ビットにすることができます。ビット単位のワード長を選択するための式は、次のようになります。 SPORT_CTL_B.SLEN = （ビット単位のシリアル・ワード長） – 1	
3 (R/W)	LSBF	最下位ビット・ファースト。 SPORT_CTL_B.LSBF ビットは、ハーフ SPORT がデータを送信または受信するときに、LSB ファーストで行うか、MSB ファーストで行うかを選択します。	
		0	MSB ファーストで送信または受信します
		1	LSB ファーストで送信または受信します
0 (R/W)	SPEN	シリアル・ポート・イネーブル。 SPORT_CTL_B.SPEN ビットは、ハーフ SPORT のデータ・チャンネルをイネーブルにします。注：このビットをクリアする（= 1 から= 0 に変更する）と、ハーフ SPORT はチャンネルのデータ・バッファを自動的に消去して、SPORT 内のクロック、フレーム同期、およびカウンタをディスエーブルにします。	
		0	ディスエーブルにします
		1	イネーブルにします

ハーフ SPORT 'A'除数レジスタ

SPORT_DIV_A には、ハーフ SPORT 'A'のために内部で生成されるクロックと、フレーム同期の周波数を指定する除数値が格納されています。

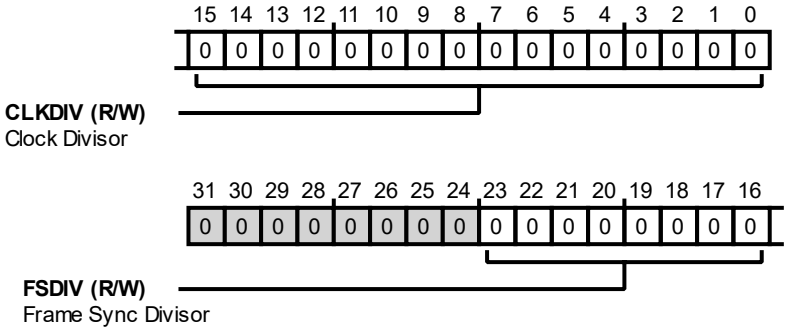


図 16-16：SPORT_DIV_A レジスタ図

表 16-5：SPORT_DIV_A レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
23:16 (R/W)	FSDIV	フレーム同期の除数。 SPORT_DIV_A.FSDIV ビットは、フレーム同期パルスを生成する前にハーフ SPORT がカウントする送信クロック・サイクル数または受信クロック・サイクル数を選択します。ハーフ SPORT は、これらが内部で生成されたシリアル・クロックでも外部で生成されたシリアル・クロックでも、シリアル・クロック・サイクルをカウントします。このフィールドは、タイマー・イネーブル・モードで CNV 信号を生成する前に、シリアル・クロック・サイクル数を計数するために使用されます。
15:0 (R/W)	CLKDIV	クロックの除数。 SPORT_DIV_A.CLKDIV ビットは、プロセッサのシステム・クロック（PCLK）を信号源とするシリアル・クロック（SPORT_ACLK）を算出するために、ハーフ SPORT が使用する除数を選択します。

ハーフ SPORT 'B'除数レジスタ

SPORT_DIV_B には、ハーフ SPORT 'B'のために内部で生成されるクロックとフレーム同期の周波数を指定する除数値が格納されています。

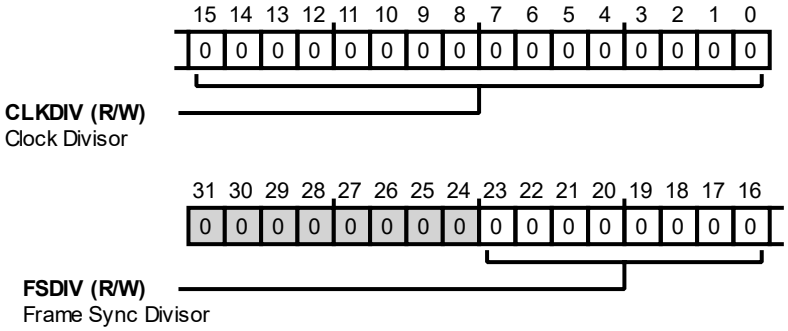


図 16-17：SPORT_DIV_B レジスタ図

表 16-6：SPORT_DIV_B レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
23:16 (R/W)	FSDIV	フレーム同期の除数。 SPORT_DIV_B.FSDIV ビットは、フレーム同期パルスを生成する前にハーフ SPORT がカウントする送信クロック・サイクル数または受信クロック・サイクル数を選択します。ハーフ SPORT は、これらが内部で生成されたシリアル・クロックでも外部で生成されたシリアル・クロックでも、シリアル・クロック・サイクルをカウントします。このフィールドは、タイマー・イネーブル・モードで CNV 信号を生成する前に、シリアル・クロック・サイクル数を計数するために使用されます。
15:0 (R/W)	CLKDIV	クロックの除数。 SPORT_DIV_B.CLKDIV ビットは、プロセッサのシステム・クロック（PCLK）を信号源とするシリアル・クロック（SPORT_BCLK）を算出するために、ハーフ SPORT が使用する除数を選択します。

ハーフ SPORT A 割込みイネーブル・レジスタ

このレジスタには、ハーフ SPORT A に存在するエラーとデータ・リクエストに関連する様々な割込みをイネーブルにすることに関わるすべてのフィールドが含まれています。

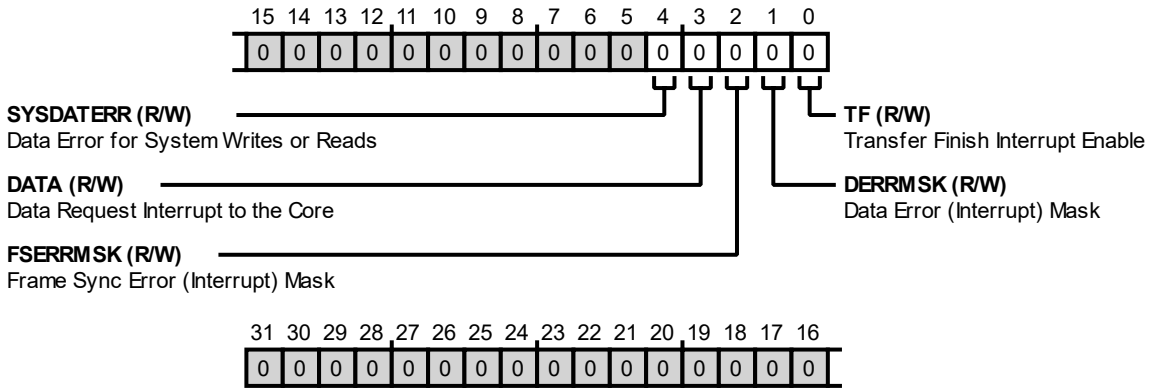


図 16-18：SPORT_IEN_A レジスタ図

表 16-7：SPORT_IEN_A レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
4 (R/W)	SYSDATERR	システム書込みまたはシステム読出しにおけるデータ・エラー。 このフィールドは、ハーフ SPORT がシステム書込みによる TX FIFO のオーバーフロー、またはシステム読出しによる RX FIFO のアンダーフローに対応するデータ・エラー割込みの発生をイネーブルにします。	
		0	システム・データ・エラー割込みをディスエーブルにします
		1	システム・データ・エラー割込みをイネーブルにします
3 (R/W)	DATA	コアへのデータ・リクエスト割込み。 このビットは、データを送信用の送信 FIFO に書き込むため、またはデータを受信 FIFO から読み出すために、コアへの割込みをイネーブルにします。	
		0	データ・リクエスト割込みをディスエーブルにします
		1	データ・リクエスト割込みをイネーブルにします
2 (R/W)	FSERRMSK	フレーム同期エラー（割込み）マスク。 SPORT_IEN_A.FSERRMSK は、ハーフ SPORT がフレーム同期エラー割込みを生成できるようにマスクを解除します（割込みをイネーブルにします）。	
		0	マスクします（ディスエーブルにします）
		1	マスクを解除します（イネーブルにします）

表 16-7：SPORT_IEN_A レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
1 (R/W)	DERRMSK	データ・エラー（割込み）マスク。 SPORT_IEN_A.DERRMSK は、ハーフ SPORT がデータ・チャンネルでのデータ・エラー割込みを生成できるようにマスクを解除します（割込みをイネーブルにします）。	
		0	マスクします（ディスエーブルにします）
		1	マスクを解除します（イネーブルにします）
0 (R/W)	TF	転送終了割込みイネーブル。 SPORT_IEN_A.TF ビットは、プログラムされた数の転送が終了したとき、ハーフ SPORT が転送終了割込みを発行するかどうかを選択します。イネーブル（SPORT_IEN_A.TF = 1）のとき、プログラムされた数の転送の最終ワードの最終ビットが完全にシフト・アウトされるか受信されると、割込みが発生します。ディスエーブル（SPORT_IEN_A.TF = 0）の場合、SPORT は割込みを発生させません。	
		0	転送終了割込みをディスエーブルにします
		1	転送終了割込みをイネーブルにします

ハーフ SPORT B 割込みイネーブル・レジスタ

このレジスタには、ハーフ SPORT B に存在するエラーとデータ・リクエストに関連する様々な割込みをイネーブルにすることに関わるすべてのフィールドが含まれています。

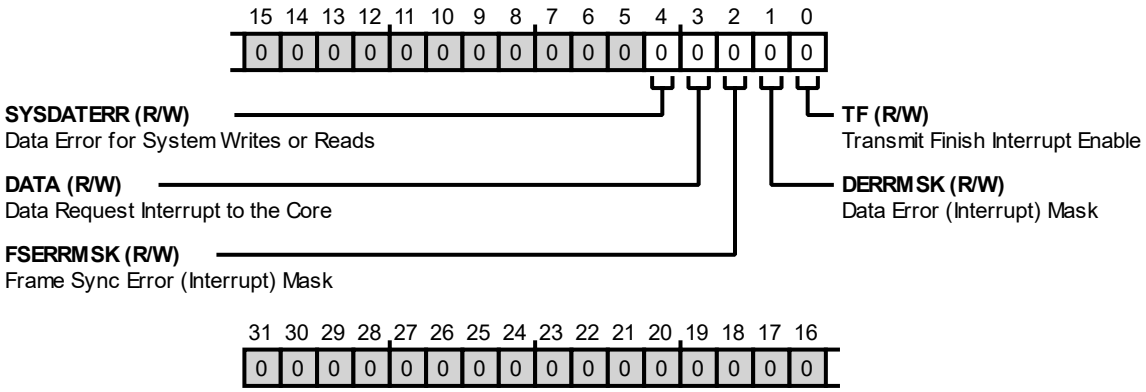


図 16-19：SPORT_IEN_B レジスタ図

表 16-8：SPORT_IEN_B レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
4 (R/W)	SYSDATERR	システム書込みまたはシステム読出しにおけるデータ・エラー。 このフィールドは、システム書込みによる TX FIFO のオーバーフロー、またはシステム読出しによる RX FIFO のアンダーフローに対応する、ハーフ SPORT のデータ・エラー割込み生成をイネーブルにします。	
		0	システム・データ・エラー割込みをディスエーブルにします
		1	システム・データ・エラー割込みをイネーブルにします
3 (R/W)	DATA	コアへのデータ・リクエスト割込み。 このビットは、データを送信用の送信 FIFO に書き込むため、またはデータを受信 FIFO から読み出すために、コアへの割込みをイネーブルにします。	
		0	データ・リクエスト割込みをディスエーブルにします
		1	データ・リクエスト割込みをイネーブルにします
2 (R/W)	FSERRMSK	フレーム同期エラー（割込み）マスク。 SPORT_IEN_B.FSERRMSK は、ハーフ SPORT がフレーム同期エラー割込みを生成できるようにマスクを解除します（割込みをイネーブルにします）。	
		0	マスクします（ディスエーブルにします）
		1	マスクを解除します（イネーブルにします）

表 16-8：SPORT_IEN_B レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
1 (R/W)	DERRMSK	データ・エラー（割込み）マスク。 SPORT_IEN_B.DERRMSK は、ハーフ SPORT がデータ・チャンネルでのデータ・エラー割込みを生成できるようにマスクを解除します（割込みをイネーブルにします）。	
		0	マスクします（ディスエーブルにします）
		1	マスクを解除します（イネーブルにします）
0 (R/W)	TF	転送終了割込みイネーブル。 SPORT_IEN_B.TF ビットは、プログラムされた数の転送が終了したとき、ハーフ SPORT が転送終了割込みを発行するかどうかを選択します。イネーブル（SPORT_IEN_B.TF = 1）のとき、プログラムされた数の転送の最終ワードの最終ビットが完全にシフト・アウトされるか受信されると、割込みが発生します。ディスエーブル（SPORT_IEN_B.TF = 0）の場合、SPORT は割込みを発生させません。	
		0	転送終了割込みをディスエーブルにします
		1	転送終了割込みをイネーブルにします

ハーフ SPORT A 転送数レジスタ

このレジスタは、SPORT_CTL_A.SPTRAN に応じて送信または受信するための転送ワード数を指定します。

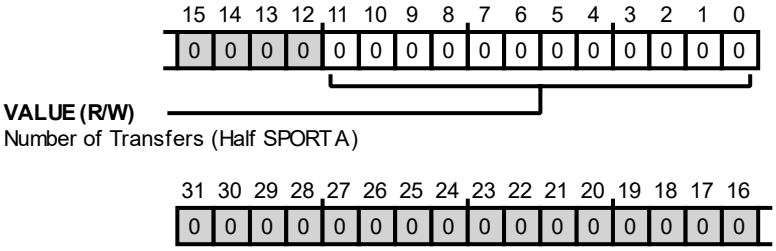


図 16-20：SPORT_NUMTRAN_A レジスタ図

表 16-9：SPORT_NUMTRAN_A レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
11:0 (R/W)	VALUE	転送数（ハーフ SPORT A）。

ハーフ SPORT B 転送数レジスタ

このレジスタは、SPORT_CTL_B.SPTRAN に応じて送信または受信するための転送ワード数を指定します。

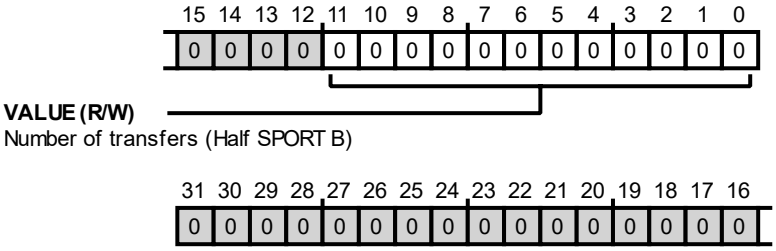


図 16-21：SPORT_NUMTRAN_B レジスタ図

表 16-10：SPORT_NUMTRAN_B レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
11:0 (R/W)	VALUE	転送数（ハーフ SPORT B）。

ハーフ SPORT 'A' Rx バッファ・レジスタ

SPORT_RX_A レジスタは、ハーフ SPORT の受信データをバッファリングします。ハーフ SPORT がデータを受信するように設定されている場合、このバッファはアクティブになります。受信シフタに完全なワードが受信された後、そのワードは SPORT_RX_A レジスタに置かれます。このデータは、コア・モード（割込みベースまたはポーリング・ベースのメカニズム）で、または、DMA コントローラを使用して、プロセッサのメモリに直接 DMA することによって読み出すことができます。

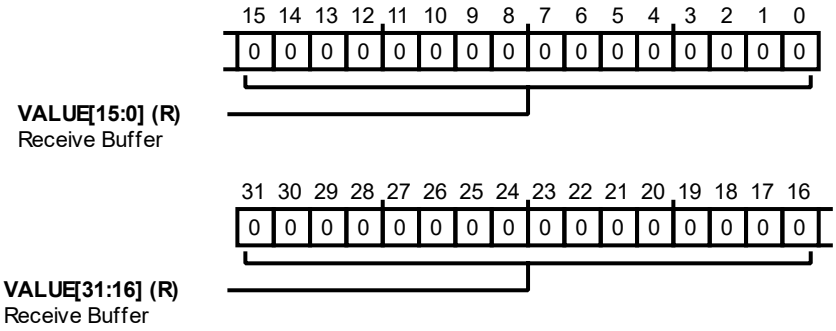


図 16-22：SPORT_RX_A レジスタ図

表 16-11：SPORT_RX_A レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	VALUE	受信バッファ。 SPORT_RX_A.VALUE ビットは、ハーフ SPORT のチャンネル受信データを保持します。

ハーフ SPORT 'B' Rx バッファ・レジスタ

SPORT_RX_B レジスタは、ハーフ SPORT のチャンネル受信データをバッファリングします。ハーフ SPORT がデータを受信するように設定されている場合、このバッファはアクティブになります。受信シフタに完全なワードが受信された後、そのワードは SPORT_RX_B レジスタに置かれます。このデータは、コア・モードで、または、DMA コントローラを使用して、プロセッサのメモリに直接 DMA することによって読み出すことができます。

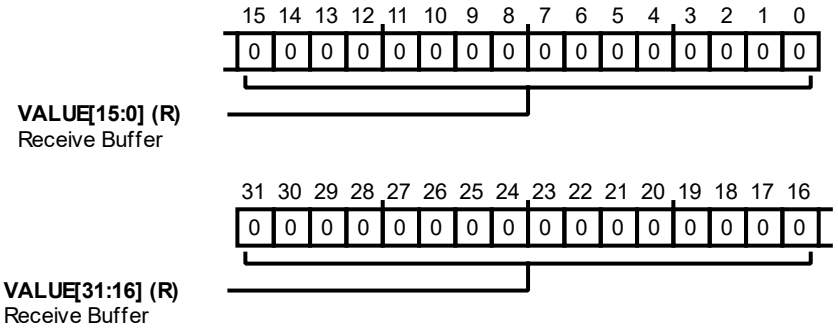


図 16-23：SPORT_RX_B レジスタ図

表 16-12：SPORT_RX_B レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
31:0 (R/NW)	VALUE	受信バッファ。 SPORT_RX_B.VALUE ビットは、ハーフ SPORT の受信データを保持します。

ハーフ SPORT 'A'ステータス・レジスタ

このレジスタには、ハーフ SPORT A のすべてのステータス・フィールドが含まれています。検出されるエラーは、フレーム同期違反、またはバッファ・オーバーフロー状態／バッファ・アンダーフロー状態です。

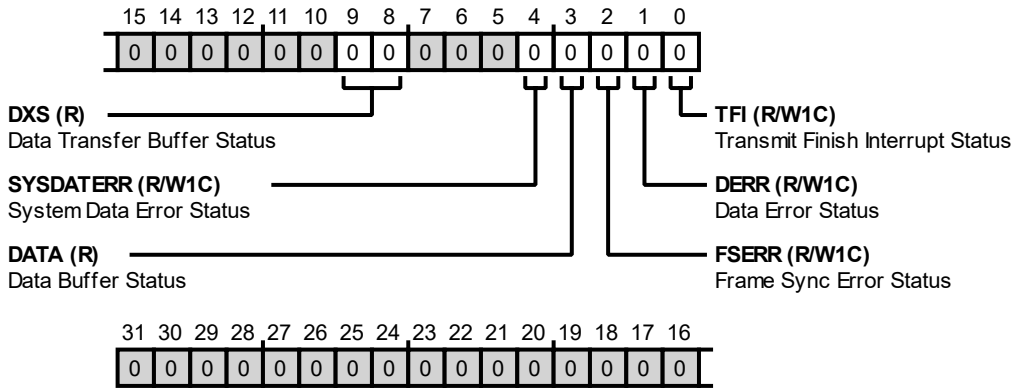


図 16-24：SPORT_STAT_A レジスタ図

表 16-13：SPORT_STAT_A レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
9:8 (R/NW)	DXS	データ転送バッファ・ステータス。 SPORT_STAT_A.DXS は、ハーフ SPORT A のデータ・バッファのステータスを示します。	
		0	エンプティ
		1	予備
		2	部分的にフル
		3	フル
4 (R/W1C)	SYSDATERR	システム・データ・エラー・ステータス。 このビットは、システムのデータ転送中におけるハーフ SPORT のデータ・バッファのエラー・ステータスを示します。送信（SPORT_CTL_A.SPTRAN = 1）の場合は、SPORT_STAT_A.SYSDATERR は送信アンダーフロー・ステータスを示します。受信（SPORT_CTL_A.SPTRAN = 0）の場合は、SPORT_STAT_A.SYSDATERR は受信オーバーフロー・ステータスを示します。	
		0	エラーなし
		1	TXFIFO でシステム送信アンダーフローが発生しているか、または RXFIFO でシステム受信オーバーフローが発生しています
3 (R/NW)	DATA	データ・バッファ・ステータス。 このフィールドは、ハーフ SPORT A のデータ・バッファのステータスのみを示します。	
		0	送信 FIFO がフルか、または受信 FIFO がエンプティです。
		1	送信 FIFO はフルではなく、かつ受信 FIFO はエンプティではありません。

表 16-13：SPORT_STAT_A レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
2 (R/W1C)	FSERR	フレーム同期エラー・ステータス。 SPORT_STAT_A.FSERR ビットは、ビット・カウント（フレーム内に残っているビット）がゼロでない場合にハーフ SPORT がフレーム同期を検出したか、または、遅延フレーム同期の場合にハーフ SPORT がフレーム同期のアクティブ状態をワード長よりも短期間しか検出できなかったことを示します。	
		0	エラーなし
		1	フレーム同期エラーが発生しています
1 (R/W1C)	DERR	データ・エラー・ステータス。 SPORT_STAT_A.DERR ビットは、ハーフ SPORT A のデータ・バッファのエラー・ステータスを示します。送信（SPORT_CTL_A.SPTRAN = 1）中は、SPORT_STAT_A.DERR は送信アンダーフロー・ステータスを示します。受信（SPORT_CTL_A.SPTRAN = 0）中は、SPORT_STAT_A.DERR は受信オーバーフロー・ステータスを示します。	
		0	エラーなし
		1	エラー（送信アンダーフローまたは受信オーバーフロー）
0 (R/W1C)	TFI	転送終了割込みステータス。 SPORT_STAT_A.TFI ビットは、プログラムされた数の転送が終了したとき、ハーフ SPORT が転送終了割込みを発行したことを示します。このビットが 1 のときは、プログラムされた数の転送の最終ワードの最終ビットが完全にシフト・アウトされているか、または受信されています。このビットが 0 のときは、全数の転送は終了していません。	
		0	最終ビットが送信／受信されていません。
		1	最終ビットが送信／受信されました。

ハーフ SPORT 'B'ステータス・レジスタ

このレジスタには、ハーフ SPORT B に存在するすべてのステータス・フィールドが含まれています。検出されるエラーは、フレーム同期違反、またはバッファ・オーバーフロー状態／バッファ・アンダーフロー状態です。

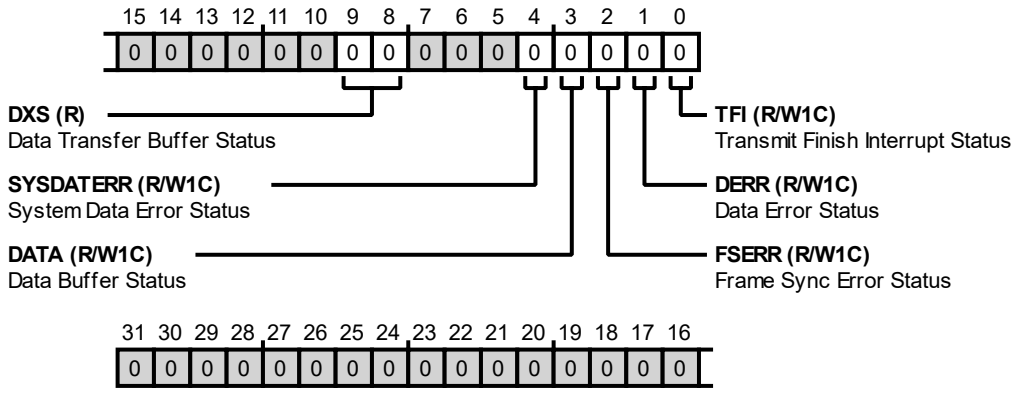


図 16-25：SPORT_STAT_B レジスタ図

表 16-14：SPORT_STAT_B レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
9:8 (R/NW)	DXS	データ転送バッファ・ステータス。 SPORT_STAT_B.DXS は、ハーフ SPORT B のデータ・バッファのステータスを示します。	
		0	エンプティ
		1	予備
		2	部分的にフル
		3	フル
4 (R/W1C)	SYSDATERR	システム・データ・エラー・ステータス。 このビットは、システムのデータ転送中におけるハーフ SPORT のデータ・バッファのエラー・ステータスを示します。送信（SPORT_CTL_B.SPTRAN = 1）の場合は、SPORT_STAT_B.SYSDATERR は送信アンダーフロー・ステータスを示します。受信（SPORT_CTL_B.SPTRAN = 0）の場合は、SPORT_STAT_B.SYSDATERR は受信オーバーフロー・ステータスを示します。	
		0	エラーなし
		1	TXFIFO でシステム送信アンダーフローが発生しているか、または RXFIFO でシステム受信オーバーフローが発生しています
3 (R/W1C)	DATA	データ・バッファ・ステータス。 このフィールドは、ハーフ SPORT B のデータ・バッファのステータスのみを示します。	
		0	送信 FIFO がフルか、または受信 FIFO がエンプティです。
		1	送信 FIFO はフルではなく、かつ受信 FIFO はエンプティではありません。

表 16-14：SPORT_STAT_B レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
2 (R/W1C)	FSERR	フレーム同期エラー・ステータス。 SPORT_STAT_B.FSERR ビットは、ビット・カウント（フレーム内に残っているビット）がゼロでない場合にハーフ SPORT がフレーム同期を検出したか、または、遅延フレーム同期の場合にハーフ SPORT がフレーム同期のアクティブ状態をワード長よりも短期間しか検出できなかったことを示します。	
		0	エラーなし
		1	エラー（フレーム同期の時点でビット・カウントがゼロではありません）
1 (R/W1C)	DERR	データ・エラー・ステータス。 SPORT_STAT_B.DERR ビットは、ハーフ SPORT B のデータ・バッファのエラー・ステータスを示します。送信（SPORT_CTL_B.SPTRAN = 1）中は、SPORT_STAT_B.DERR は送信アンダーフロー・ステータスを示します。受信（SPORT_CTL_B.SPTRAN = 0）中は、SPORT_STAT_B.DERR は受信オーバーフロー・ステータスを示します。	
		0	エラーなし
		1	エラー（送信アンダーフローまたは受信オーバーフロー）
0 (R/W1C)	TFI	転送終了割込みステータス。 SPORT_STAT_B.TFI ビットは、プログラムされた転送数が終了したとき、ハーフ SPORT が転送終了割込みを発行したことを示します。このビットが 1 のときは、プログラムされた転送数の最終ワードの最終ビットが完全にシフト・アウトされているか、または受信されています。このビットが 0 のときは、全数の転送は終了していません。	
		0	最終ビットが送信／受信されていません。
		1	最終ビットが送信／受信されました。

ハーフ SPORT 'A' CNV 幅レジスタ

このレジスタには、ハーフ SPORT A での CNV 信号に関連する設定値が格納されています。

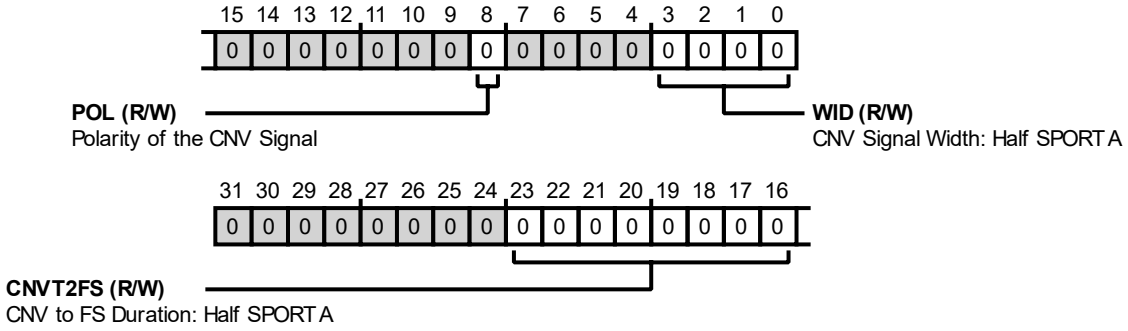


図 16-26：SPORT_CNVT_A レジスタ図

表 16-15：SPORT_CNVT_A レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
23:16 (R/W)	CNVT2FS	CNV から FS までの時間幅：ハーフ SPORT A。 このフィールドには、ハーフ SPORT A において、CNV 信号のアサートからフレーム同期信号までの必要な時間幅に応じてプログラムされるクロック数の値が格納されます。	
8 (R/W)	POL	CNV 信号の極性。 このビットは、CNV 信号の極性を指定します。	
		0	アクティブ・ハイの極性
		1	アクティブ・ローの極性
3:0 (R/W)	WID	CNV 信号の幅：ハーフ SPORT A。 このフィールドには、ハーフ SPORT A において、CNV 信号がアクティブになる必要のあるシリアル・クロック数の値が格納されます。	

ハーフ SPORT 'B' CNV 幅レジスタ

このレジスタには、ハーフ SPORT B での CNV 信号に関連する設定値が格納されています。

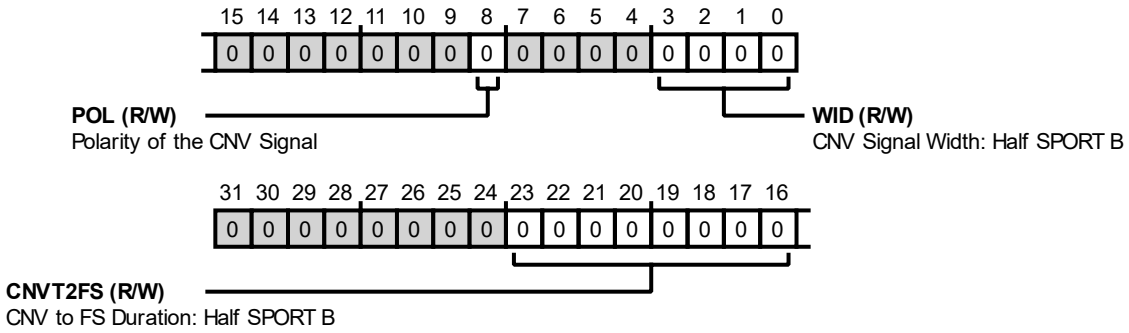


図 16-27：SPORT_CNVT_B レジスタ図

表 16-16：SPORT_CNVT_B レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
23:16 (R/W)	CNVT2FS	CNV から FS までの時間幅：ハーフ SPORT B。 このフィールドには、ハーフ SPORT B において、CNV 信号のアサートからフレーム同期信号までの必要な時間幅に応じてプログラムされるクロック数の値が格納されます。	
8 (R/W)	POL	CNV 信号の極性。 このビットは、CNV 信号の極性を指定します。	
		0	アクティブ・ハイの極性
		1	アクティブ・ローの極性
3:0 (R/W)	WID	CNV 信号の幅：ハーフ SPORT B。 このフィールドには、ハーフ SPORT B において、CNV 信号の必要な幅に応じてプログラムされるクロック数の値が格納されます。	

ハーフ SPORT 'A' Tx バッファ・レジスタ

SPORT_TX_A レジスタは、ハーフ SPORT の送信データをバッファリングします。ハーフ SPORT が送信用に設定された場合、このレジスタに送信するデータをロードする必要があります。プロセッサ・コアで動作しているプログラムがデータをバッファにロードする場合（ワードごとのプロセス）も、DMA コントローラがデータをバッファに自動的にロードする場合（DMA プロセス）もあります。

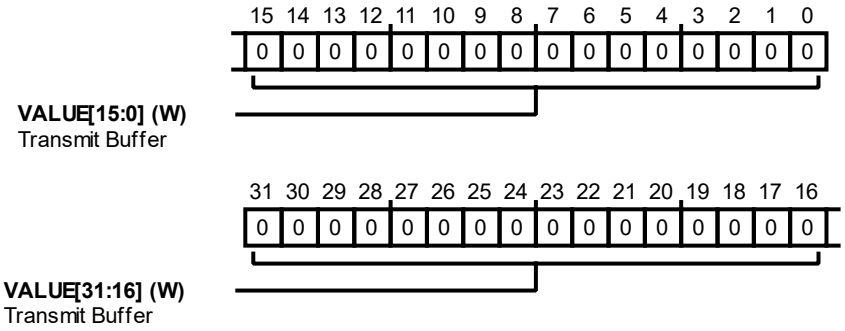


図 16-28：SPORT_TX_A レジスタ図

表 16-17：SPORT_TX_A レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	VALUE	送信バッファ。 SPORT_TX_A.VALUE ビットは、ハーフ SPORT のチャンネル送信データを保持します。

ハーフ SPORT 'B' Tx バッファ・レジスタ

SPORT_TX_B レジスタは、ハーフ SPORT のチャンネル送信データをバッファリングします。このレジスタに送信するデータをロードする必要があります。プロセッサ・コアで動作しているプログラムがデータをバッファにロードする場合（ワードごとのプロセス）と、DMA コントローラがデータをバッファに自動的にロードする場合（DMA プロセス）があります。

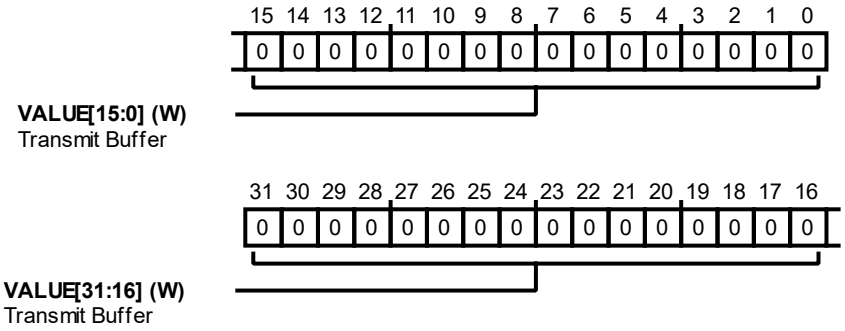


図 16-29：SPORT_TX_B レジスタ図

表 16-18：SPORT_TX_B レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
31:0 (RX/W)	VALUE	送信バッファ。 SPORT_TX_B.VALUE ビットは、ハーフ SPORT の送信データを保持します。

17 ユニバーサル非同期レシーバー／トランスミッタ（UART）

UART ペリフェラルは、業界標準の 16450/16550 と互換性のあるシリアル全二重のユニバーサル非同期レシーバー／トランスミッタです。シリアル通信は、様々なワード長、ストップ・ビット、およびパリティ生成オプションをサポートする非同期プロトコルに準拠しています。ADuCM4050 MCU には、2 つの UART コアがあります。

UART の機能

- シリアル全二重のユニバーサル非同期レシーバー／トランスミッタ。
- 業界標準の 16450/16550 と互換性あり。
- データ・バッファ・フル／エンプティ、転送エラー検出、およびブレーク検出など、複数のユニークなイベントに対応するハードウェアを処理する割込みを搭載。
- 高精度のボー・レートの生成を容易にする分数分周器。
- 7 ビット～12 ビットのワード長。
- 受信用と送信用に 2 つの専用 DMA チャンネル。
- 5～8 のデータ・ビット。
- 1 ストップ・ビット、2 ストップ・ビット、または 1.5 ストップ・ビット。
- パリティなし、偶数パリティ、または奇数パリティ。
- 4 倍、8 倍、16 倍、32 倍のオーバー・サンプル・レートでプログラム可能。

UART 機能の説明

UART ペリフェラルは、業界標準の非同期シリアル通信をサポートし、送信ピンと受信ピンを通してデータを転送することができます。UART では、7 ビット～12 ビットのワード長をサポートします。送信動作は、送信保持レジスタ（UART_TX）に書き込むことにより開始されます。受信動作では、常に 1 であるストップ・ビットの数を除いて、送信設定と同じデータ・フォーマットを使用します。

UART のブロック図

UART のブロック図を以下に示します。

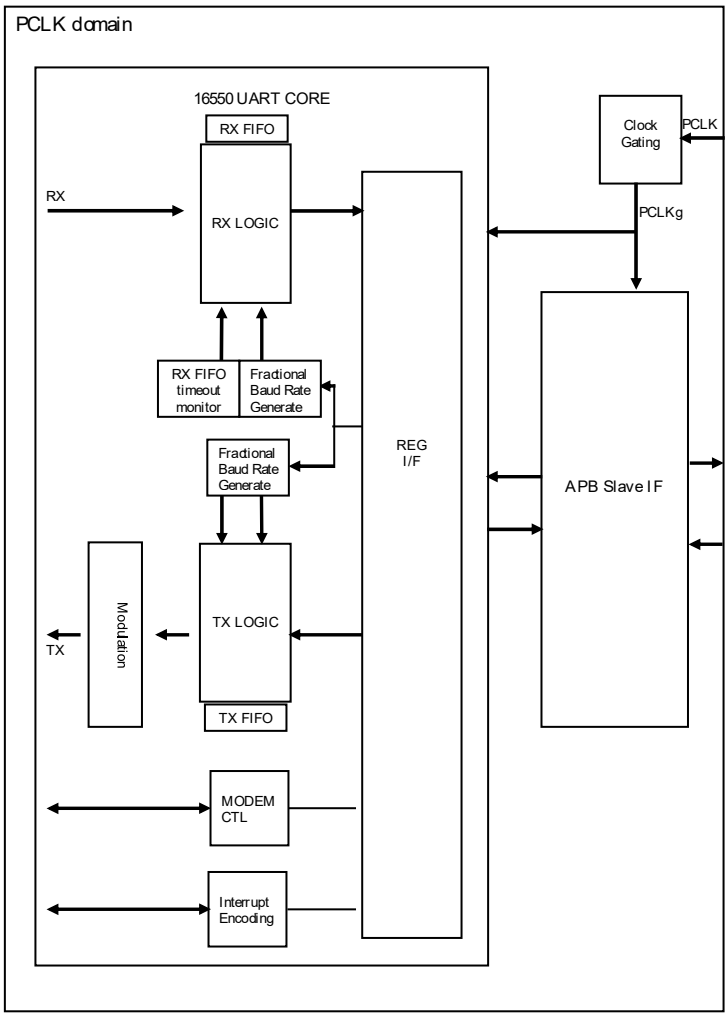


図 17-1：UART のブロック図

UART の動作

シリアル通信

非同期シリアル通信プロトコルは、次のオプションをサポートしています。

- 5～8 のデータ・ビット
- 1 ストップ・ビット、2 ストップ・ビット、または 1.5 ストップ・ビット
- パリティなし、偶数パリティ、または奇数パリティ
- 4 倍、8 倍、16 倍、32 倍のオーバー・サンプル・レートでプログラム可能。

- ボー・レート = PCLK /（（M + N / 2048）× 2^{OSR+2} × DIV）
  ここで、
  OSR（UART_LCR2.OSR）= 0～3
  DIV（UART_DIV）= 1～65535
  M（UART_FBR.DIVM）= 1～3
  N（UART_FBR.DIVN）= 0～2047

すべてのデータ・ワードには、1 つのスタート・ビットと 1 つ以上のストップ・ビットが必要です。これにより、各ワードは 7 ビット～12 ビットの範囲で生成されます。

送信動作は、送信保持レジスタ（UART_TX）に書き込むことにより開始されます。同期遅延の後に、データは送信シフト・レジスタに移動します。この移動において、データは必要に応じて付加されたスタート・ビット、ストップ・ビット、およびパリティ・ビットと共に PCLK /（（M + N/2048）× 2^{OSR+2} × DIV）に等しいボー（ビット）レートでシフト・アウトされます。すべてのデータ・ワードは、スタート・ビットの立下がりから始まります。送信保持レジスタから送信シフト・レジスタへの転送が行われると、送信レジスタ・エンプティ・ステータス・ビット（UART_LSR.THRE）が 1 に設定されます。

受信動作では、常に 1 であるストップ・ビットの数を除いて、送信設定と同じデータ・フォーマットを使用します。スタート・ビットが検出された後に、受信されたワードは受信シフト・レジスタにシフトされます。適正な数のビット（ストップ・ビットを含む）が受信された後に、適正な同期遅延が経過すると、受信シフト・レジスタは受信バッファ・レジスタに転送を行い、受信バッファ・レジスタ・フル・ステータス・フラグ（UART_IIR.STAT）が更新されます。

ボー・レートの 2^{OSR+2} 倍に等しいサンプリング・クロックを使用して、できる限りビットの中点近くでデータをサンプリングします。このサンプリング・クロック周期よりも短いスプリアス・パルスを除去する受信フィルタも搭載されています。

注記：データは、最下位ビット、すなわち送信シフト・レジスタのビット 0 が最初に送信および受信されます。

ボー・レート・ジェネレータ

ボー・レート・ジェネレータは分数分周器を備えており、高精度のボー・レートを生成します。ボー・レートは、次のように算出することができます。

分数分周器をバイパスするためには、UART_FBR.FBEN = 0 に設定します。

この設定により、ボー・レート = PCLK /（2^{OSR+2} × DIV）になります。

整数値の UART_DIV では、上記のレートに対して誤差が大きすぎる場合、UART_FBR.DIVN の値と UART_FBR.DIVM の値を見積もって分数ボー・レート・ジェネレータを使用すると、ボー・レートを微調整できます。

注記：UART_DIV を 0 に設定すると、UART ロジックは無効になります。

次の表は、入力クロックが 52MHz、26MHz、および 16MHz であると仮定した場合のボー・レートの例を示しています。

表 17-1：52MHz の PCLK に基づいたボー・レートの例

Baud Rate	OSR	DIV	DIVM	DIVN	Actual	Error
1,000,000	3	1	1	1280	1000000	0.0%
1,500,000	3	1	1	171	1499774	-0.002%
3,000,000	2	1	1	171	2999549	-0.015%
3,500,000	1	1	1	1755	3500394	0.0011%

表 17-2：26MHz の PCLK に基づいたボー・レートの例

Baud Rate	OSR	DIV	DIVM	DIVN	Actual	Error
9600	3	28	3	46	9600.74	0.0077%
19200	3	14	3	46	19201.48	0.0077%
38400	3	7	3	46	38402.95	0.0077%
57600	3	14	1	15	57613.74	0.0024%
115,200	3	3	2	719	115195.6	-0.004%
230,400	3	3	1	359	230439	0.0017%
460,800	3	1	1	1563	460814	0.003%
921,600	2	1	1	1563	921628	0.003%
1,000,000	2	1	1	1280	1000000	0.0%
1,500,000	2	1	1	171	1499774	-0.002%

表 17-3：16MHz の PCLK に基づいたボー・レートの例

Baud Rates	OSR	DIV	DIVM	DIVN	Actual	% Error
9600	3	17	3	131	9599.25	-0.0078%
19200	3	8	3	523	19199.04	-0.0050%
38400	3	4	3	523	38398.08	-0.0050%
57600	3	8	1	174	57605.76	0.0100%
115,200	3	2	2	348	115211.5	0.0100%
230,400	3	2	1	174	230423	0.0100%
460,800	3	1	1	174	460846.1	0.0100%
921,600	2	1	1	174	921692.2	0.0100%
1,000,000	2	1	1	0	1000000	0.0000%
1,500,000	1	1	1	683	1499816.9	-0.012%

UART の動作モード

ADuCM4050 MCU で使用されている UART は、次のモードをサポートしています。

IO モード

このモードでは、ソフトウェアは UART とデータのやり取りを行います。これは割込みサービス・ルーチンによって行われ、このルーチンは必要に応じて、データの読出しや書込みにより発生する送信割込みと受信割込みに応答します。このモードでは、受信チャンネルでのオーバーラン・エラーを防ぐために、ソフトウェアは一定時間内に応答する必要があります。

また、IO モードでは、データの移動に支障のないタイミングを判断するために、ステータス・フラグをポーリングする必要があります。

このモードはコアに集中するため、システムがオーバーヘッドを許容可能な場合しか使用できません。割込みは、UART 割込みイネーブル・レジスタ（UART_IEN）を使用して制御することができます。

送信保持レジスタがエンプティでないときに送信保持レジスタに書き込んだり、受信バッファ・レジスタがフルでないときに受信バッファ・レジスタから読み出したりすると、誤った結果が生じます。前者の場合、送信保持レジスタは新しいワードで上書きされ、以前のワードは決して送信されません。後者の場合は、以前受信されたワードが再び読み出されます。これらのエラーは、割込みを使用するか、またはステータス・レジスタのポーリングを行って、ソフトウェアで適切に回避する必要があります。これらのエラーは、ハードウェアでは検出されません。

DMA モード

このモードでは、ユーザ・コードは UART とデータのやり取りを行いません。DMA ブロックへの DMA 要求信号は、UART がデータを送信または受信する準備ができていることを示すために生成されます。これらの DMA 要求信号は、UART_IEN レジスタ内で制御することができます。

FIFO モード（16550）

この UART には、業界標準の 16550 との互換性を持たせるために、16 バイトの深さの TX FIFO と RX FIFO が実装されています。

デフォルトでは、FIFO はディスエーブルになっています。FIFO は、UART_FCR.FIFOEN ビットを 1 に設定することによってイネーブルになります。イネーブルになると、内部の FIFO がアクティブになり、受信モードと送信モードの両方で 16 バイト（および、RX FIFO 内のバイトあたり 3 ビットのエラー・データ）が記憶されます。これによりシステムのオーバーヘッドを最小限に抑え、システム効率を最大限に高めることができます。

RX FIFO の割込みおよび／または DMA のトリガ・レベルは、UART_FCR.RFTRIG ビットで設定可能です。2 つの DMA モードが使用可能です。必要な DMA モードは、UART_FCR.FDMAMD ビットを使用してプログラムできます。FIFO がイネーブルのとき、DMA モード 1 だけが（バースト・モードで）動作します。

注記：詳細については、UART_FCR レジスタを参照してください。

UART の割込み

UART ペリフェラルは、すべての Rx 割込みと Tx 割込みを代表するコア割込みコントローラに対して、1 つの出力信号を備えています。割込みの原因を判断するためには、ソフトウェアで UART 割込み識別レジスタ（UART_IIR）を読み出す必要があります。

IO モードでは、以下の場合に割込みが発生する可能性があります。

- 受信バッファ・レジスタ・フル
- 受信オーバーラン・エラー
- 受信パリティ・エラー
- 受信フレーミング・エラー
- FIFO（16550）がイネーブルの場合の受信 FIFO タイムアウト
- ブレーク割込み（SIN がローを維持）
- 送信保持レジスタ・エンプティ

自動ボー・レート検出（ABD）

自動ボー・レート検出ブロックは、事前に想定していない 2 つの UART デバイスのボー・レートを自動的に一致させるために使用されます。

UART_ACR.ABE ビットにより、レシーバーを自動ボー・レート検出モードで動作させることができます。20 ビット・カウンタのロジックは、設定されたスタート・エッジとエンド・エッジ（立上がりまたは立下がり）の間で PCLK の周期数をカウントします。この 20 ビット・カウンタは、UART_ASRH.CNT（上位 8 ビット）と UART_ASRL.CNT（下位 12 ビット）に記憶されます。

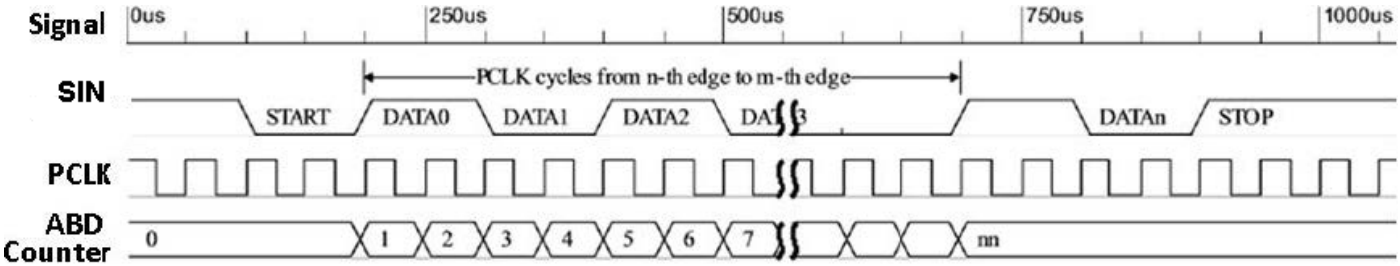


図 17-2：自動ボー・レートの例

目的のエッジに達すると、割込みが発生します。タイムアウト割込みがイネーブル（UART_ACR.TOIEN が 1 に設定されている）の間、カウンタがオーバーフローした場合、長時間のブレーク（UART_ASRL.BRKTO）、有効なスタート・エッジなし（UART_ASRL.NSETO）、または、有効なエンディング・エッジなし（UART_ASRL.NEETO）が原因で、ブロックはタイムアウト割込みを発生させます。CNT がリード・バックされると、ボー・レートを取得するために、ソフトウェアはアクティブなスターティング・エッジとエンディング・エッジの間の有効なビット数（CountedBits と定義される）を認識する必要があります。CountedBits は、自動ボー・レート検出に使用するキャラクタ、および選択されたスターティング・エッジとエンディング・エッジに依存します。

自動ボー・レートは、（必要に応じて）内部カウンタをクリアする場合は無効にし、別の実行をする場合は再び有効にする必要があります。

次の図は、使用するキャラクタとエッジに応じて異なるパラメータを設定する方法を示しています。

例えば、8 ビット・モード、パリティ・ビットなし、LSB ファーストで、受信されたデータ・バイトが 0x0D（0b'00001101、キャリッジ・リターン）であった場合に、3 つの立上がりエッジと 3 つの立下がりエッジがあります。キャリッジ・リターンの最初のエッジから最後のエッジまでをカウントするためには、UART_ACR.SEC フィールドに 1（スタート・

ビットから 2 番目のエッジ）を書き込む必要があります。また、ストップ・ビットのエッジ（スターティング・エッジから 4 番目のエッジ）までをカウントするためには、UART_ACR.EEC フィールドに 3 を書き込む必要があります。この場合、スターティング・エッジとエンディング・エッジの間の CountedBits は、8 ビットになります。

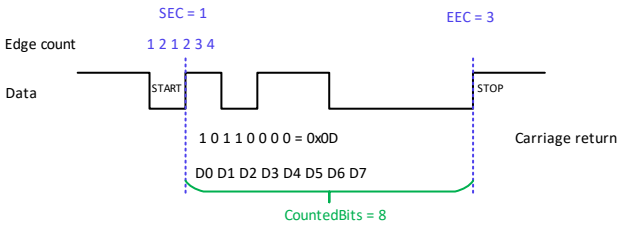


図 17-3：自動ボー・レートの設定例

スタート・ビットからストップ・ビットまでをカウントすることが可能です。この場合、UART_ACR.SEC フィールドに 0（スタート・ビットから 1 番目のエッジ）を書き込み、UART_ACR.EEC フィールドには 4（スタート・エッジから 5 番目のエッジ）を書き込む必要があります。この結果、CountedBits は 9 になるはずです。

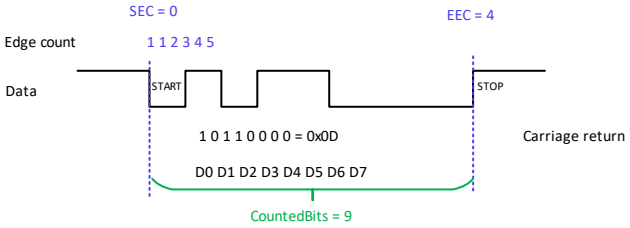


図 17-4：自動ボー・レートの設定例

その他のキャラクタは、自動ボー・レート検出用に使用できます。例えば、バックスペース 0x08（0b'00010000）について、データの最初のエッジとストップ・ビットの間を測定するために、UART_ACR.SEC フィールドを 1 に設定し、UART_ACR.EEC フィールドを 1 に設定すると、CountedBits は 5 になるはずです。

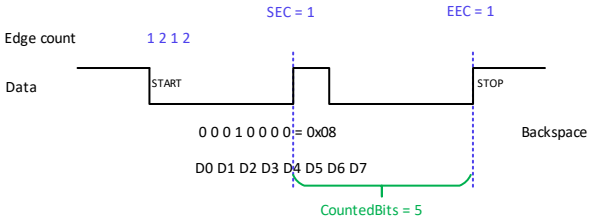


図 17-5：自動ボー・レートの設定例

20 ビット・カウンタでは、次式を使用します。

$$CNT\,[19:0] = CountedBits \times 2^{OSR+2} \times UART\_DIV \times (UART\_FBR.DIVM + UART\_FBR.DIVN \div 2048)$$

UART のボー・レート設定に基づいて、自動ボー・レート検出結果は次のように計算できます。

注記：打ち切り誤差を低減するために、UART_FBR.DIVM フィールドを 1 に設定します。

if CNT < 8 × CountedBits, OSR = 0, UART_FBR.DIVN = 512 × CNT ÷ CountedBits − 2048,

else if CNT < 16 × CountedBits, OSR = 1, UART_DIV = 1, UART_FBR.DIVN = 256 × CNT ÷ CountedBits − 2048,

else if CNT < 32 × CountedBits, OSR = 2, UART_DIV = 1, UART_FBR.DIVN = 128 × CNT ÷ CountedBits − 2048,

else if CNT > = 32 × CountedBits, OSR = 3,

if CNT が（32 × CountedBits）で割り切れる, UART_DIV = CNT ÷ 32 ÷ CountedBits,

else UART_DIV = $2^{\log2（CNT \div 32 \div CountedBits）}$ //打ち切り誤差を低減するために、UART_DIV フィールドを最も近い 2 の累乗に設定します。

UART_FBR.DIVN = 64 × CNT ÷ UART_DIV ÷ CountedBits − 2048

RS485 の半二重モード

UART ペリフェラルは、3 つの GPIO（2 つは UART_TX（SOUT）と UART_RX（SIN）、1 つは UART_SOUT_EN）を使って、RS485 トランシーバーと通信する機能を提供します。UART_SOUT_EN は、RS485 トランシーバーの送信／受信の方向を制御します。UART_SOUT_EN がアサートされると常に、トランシーバーの送信ドライバのイネーブル（アクティブ・ハイ制御）と受信ドライバのディスエーブル（アクティブ・ロー制御）を同時に行うことができるので、RS485 バスへのデータ送信が容易になります。UART_SOUT_EN がディアサートされる場合は常に、トランシーバーの受信ドライバのイネーブルと送信ドライバのディスエーブルを同時に行うことができます。受信時に Tx をホールド・オフするためには、UART_RSC_DISTX ビットをセットし、送信時に Rx をディスエーブルするためには、UART_RSC_DISRX ビットをセットします。RS485 の半二重モードのインターフェース図の一例を以下に示します。

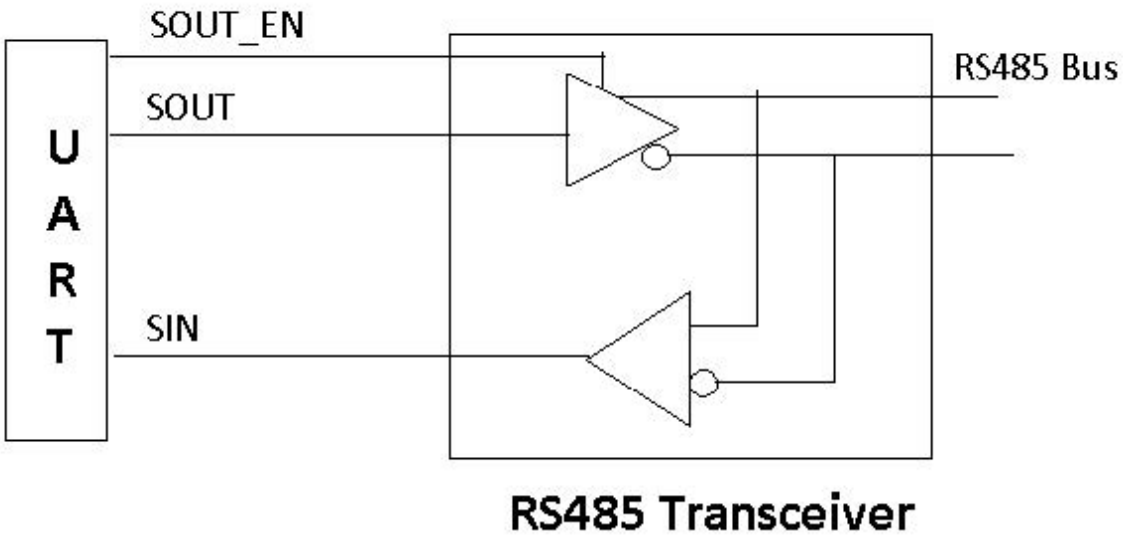


図 17-6：RS485 トランシーバーを実装した UART

受信ラインの反転

光リンクを通した UART 通信などの特定のアプリケーションでは、受信ラインは反対レベル（すなわち、ロー・レベルでアイドリング）で動作する可能性があります。UART_CTL.RXINV レジスタは、この目的のために受信ラインを反転させることができます。

注記：RS485 のアプリケーションでは、受信ラインの反転を使用しないでください。

UART_CTL.RXINV レジスタを設定してから UART を設定し、自動ボー・レートを有効にします。

クロック・ゲーティング

UART ロジックを駆動するクロックは、アイドル時に自動的にゲート・オフされ、アクセスされません。この自動クロック・ゲーティングは、UART_CTL.FORCECLKON フィールドで無効にできます。

UART とパワーダウン・モード

進行中の UART 転送が終了してから、チップを休止モードにしてパワーダウンします。あるいは、UART_DIV レジスタをクリアすることによって UART をディスエーブルにしてから休止モードにします。

注記：UART の転送中に休止モードが選択されると、休止からの復帰時に転送は継続されません。UART 内のすべての中間データ、状態、ステータス・ロジックがクリアされます。ただし、TX パッド（ピン・マルチプレクスが選択されている場合の SOUT と SOUT_EN）は、送信時に休止モードでアクティブのままになる可能性があります。

休止後、UART_DIV レジスタを 1 に設定することによって（予めクリアされている場合）、UART をイネーブルにすることができます。DMA モードにする必要がある場合、UART_IEN［5:4］を設定する必要があります。

クリーンなウェイクアップのためには、

- 休止に先立ち、UART_DIV レジスタをクリアすることによって UART ブロックをディスエーブルにします。

  または
- 休止から復帰した後、すべての UART の処理の前に、システムのウェイクアップ・タイム（代表値 10µs）よりも 1 フレーム期間以上長いブレークを適用します。これにより、UART は処理開始条件を確実に認識できます。

  ウェイクアップ・タイムは、クロック源と命令コード用に使用されているメモリに依存します。

次のレジスタは、休止モードで保持されます。その他のレジスタと内部ロジックは、ハードウェア・デフォルト値にクリアされます。

- UART_IEN.ELSI、UART_IEN.ERBFI
- UART_LCR.BRK、UART_LCR.SP、UART_LCR.EPS、UART_LCR.PEN、UART_LCR.STOP、UART_LCR.WLS
- UART_FCR.RFTRIG、UART_FCR.FDMAMD、UART_FCR.FIFOEN
- UART_FBR.FBEN、UART_FBR.DIVM、UART_FBR.DIVN

- UART_DIV.DIV
- UART_LCR2.OSR
- UART_CTL.RXINV、UART_CTL.FORCECLK
- UART_RSC.DISTX、UART_RSC.DISRX、UART_RSC.OENSP、UART_RSC.OENP

ADuCM4050 UART レジスタの説明

ユニバーサル非同期レシーバー／トランスミッタ（UART）には、次のレジスタがあります。

表 17-4：ADuCM4050 UART のレジスタ・リスト

レジスタ名	説明
UART_ACR	自動ボー・レート制御
UART_ASRH	自動ボー・レート・ステータス（ハイ）
UART_ASRL	自動ボー・レート・ステータス（ロー）
UART_CTL	UART コントロール・レジスタ
UART_DIV	ボー・レート分周器
UART_FBR	分数ボー・レート
UART_FCR	FIFO 制御
UART_IEN	割込みイネーブル
UART_IIR	割込み ID
UART_LCR	ライン制御
UART_LCR2	第 2 のライン制御
UART_LSR	ライン・ステータス
UART_RFC	RX FIFO のバイト・カウント
UART_RSC	RS485 の半二重制御
UART_RX	受信バッファ・レジスタ
UART_SCR	スクラッチ・バッファ
UART_TFC	TX FIFO のバイト・カウント
UART_TX	送信保持レジスタ

自動ボー・レート制御

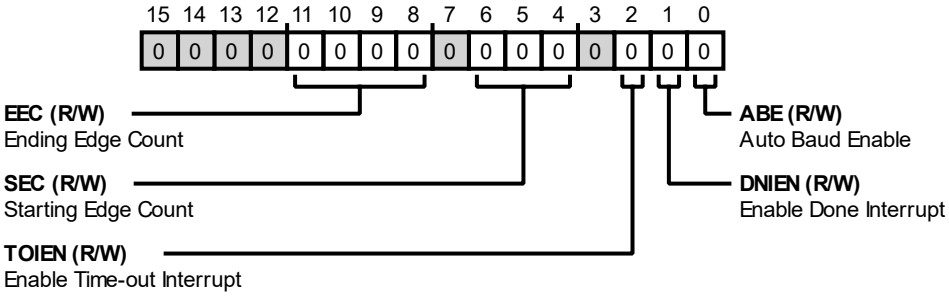


図 17-7：UART_ACR レジスタ図

表 17-5：UART_ACR レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
11:8 (R/W)	EEC	エンディング・エッジ・カウント。 UART_ACR.SEC ビットフィールドで設定されるスタート・エッジ・カウントからカウントされます。	
		0	1 番目のエッジ。スターティング・エッジ（UART_ACR.SEC 内に設定）の後の 1 番目のエッジに一致します。
		1	2 番目のエッジ
		2	3 番目のエッジ
		3	4 番目のエッジ
		4	5 番目のエッジ
		5	6 番目のエッジ
		6	7 番目のエッジ
		7	8 番目のエッジ
		8	9 番目のエッジ
6:4 (R/W)	SEC	スターティング・エッジ・カウント。 START ビットの立下がりエッジからカウントされます。	
		0	1 番目のエッジ。START ビットの立下がりエッジに一致します。
		1	2 番目のエッジ
		2	3 番目のエッジ
		3	4 番目のエッジ
		4	5 番目のエッジ
		5	6 番目のエッジ

表 17-5：UART_ACR レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
		6	7 番目のエッジ
		7	8 番目のエッジ
2 (R/W)	TOIEN	タイムアウト割込みのイネーブル。	
		0	タイムアウト割込みをディスエーブルにします
		1	タイムアウト割込みをイネーブルにします
1 (R/W)	DNIEN	終了割込みのイネーブル。	
		0	終了割込みをディスエーブルにします
		1	終了割込みをイネーブルにします
0 (R/W)	ABE	自動ボー・レートのイネーブル。	
		0	自動ボー・レートを無効にします
		1	自動ボー・レートを有効にします

自動ボー・レート・ステータス（ハイ）

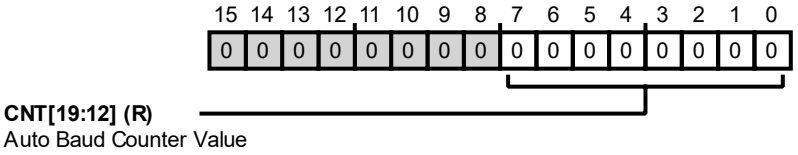


図 17-8：UART_ASRH レジスタ図

表 17-6：UART_ASRH レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (R/NW)	CNT	自動ボー・レート・カウンタ値。

自動ボー・レート・ステータス（ロー）

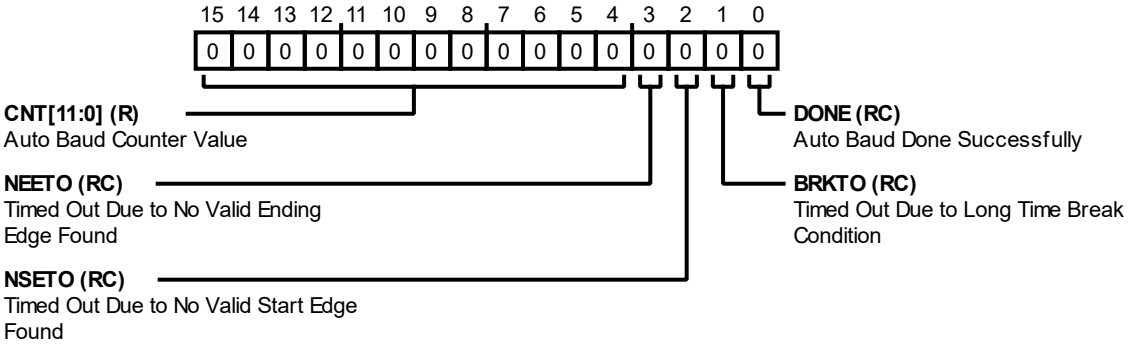


図 17-9：UART_ASRL レジスタ図

表 17-7：UART_ASRL レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:4 (R/NW)	CNT	自動ボー・レート・カウンタ値。
3 (RC/NW)	NEETO	有効なエンディング・エッジが検出されないことが原因のタイムアウト。
2 (RC/NW)	NSETO	有効なスタート・エッジが検出されないことが原因のタイムアウト。
1 (RC/NW)	BRKTO	長時間のブレーク状態が原因のタイムアウト。
0 (RC/NW)	DONE	自動ボー・レートが正常に終了。

UART コントロール・レジスタ

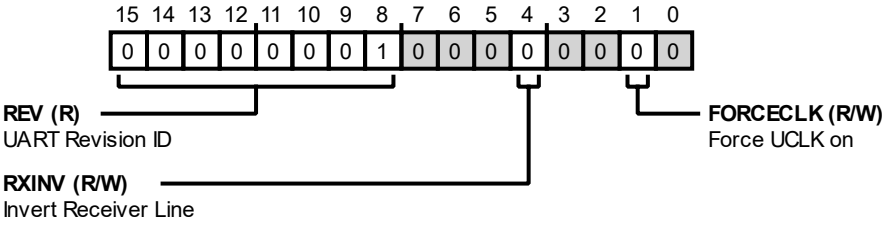


図 17-10：UART_CTL レジスタ図

表 17-8：UART_CTL レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15:8 (R/NW)	REV	UART リビジョン ID。	
4 (R/W)	RXINV	レシーバー・ラインの反転。	
		0	レシーバー・ラインを反転しません（ハイでアイドリング）。
		1	レシーバー・ラインを反転します（ローでアイドリング）。
1 (R/W)	FORCECLK	UCLK の強制的なオン。	
		0	UCLK が自動的にゲートされます
		1	UCLK が常時動作します

ボー・レート分周器

内部 UART ボー・レート生成カウンタは、同じ値または異なる値にかかわらず、UART_DIV レジスタが書込みによってアクセスされるたびに、再スタートします。

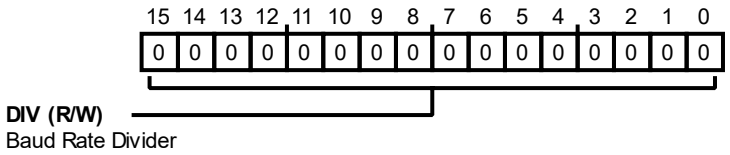


図 17-11：UART_DIV レジスタ図

表 17-9：UART_DIV レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	DIV	ボー・レート分周器。 UART_DIV レジスタを 0 に設定しないでください。0 は仕様規定されていません。設定可能な範囲は 1～65535 です。

分数ボー・レート

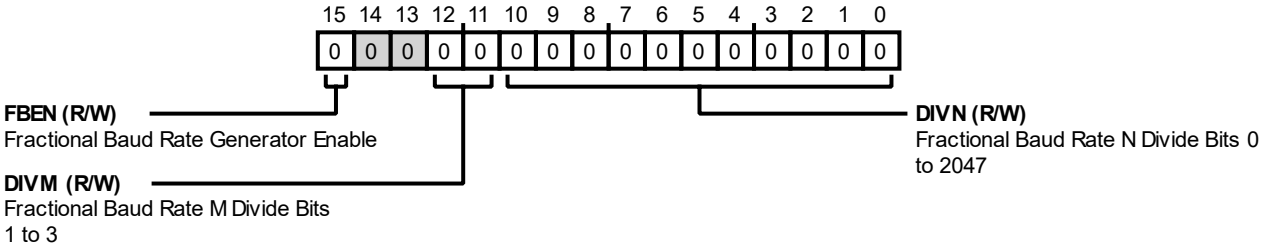


図 17-12：UART_FBR レジスタ図

表 17-10：UART_FBR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15 (R/W)	FBEN	分数ボー・レート・ジェネレータ・イネーブル。 分数ボー・レートの生成は次式で記述することが可能で、UART 動作の最終的なボー・レートは次のように算出できます。ボー・レート =（UCLK /（2 *（UART_FBR.DIVM + UART_FBR.DIVN / 2048）* 16 * UART_DIV））
12:11 (R/W)	DIVM	分数ボー・レート M 分周ビット 1～3。 分数ボー・レートが有効なとき、このビットを 0 に設定しないでください。
10:0 (R/W)	DIVN	分数ボー・レート N 分周ビット 0～2047。

FIFO 制御

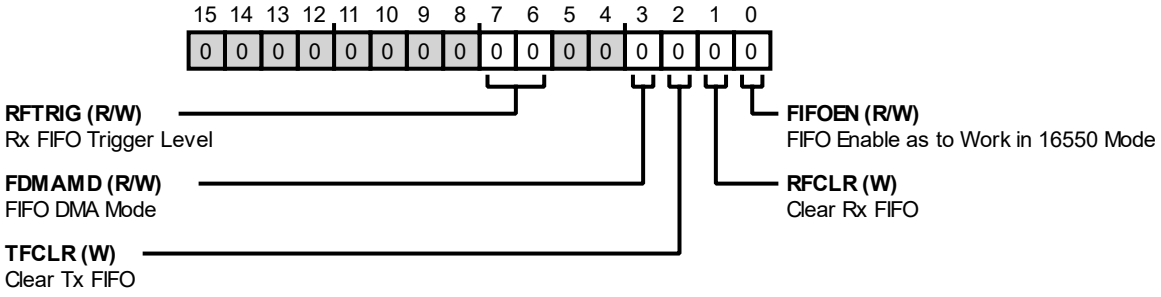


図 17-13：UART_FCR レジスタ図

表 17-11：UART_FCR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
7:6 (R/W)	RFTRIG	Rx FIFO のトリガ・レベル。FIFOEN=1 の場合のみ書込み可能です。	
		0	RX 割込みのトリガまで 1 バイト
		1	RX 割込みのトリガまで 4 バイト
		2	RX 割込みのトリガまで 8 バイト
		3	RX 割込みのトリガまで 14 バイト
3 (R/W)	FDMAMD	FIFO DMA モード。 FIFOEN = 1 の場合のみ書込み可能です。	
		0	DMA モード 0 では、RX DMA リクエストは、RBR または RX FIFO 内にデータが存在するときは常にアサートされ、RBR または RX FIFO がエンプティのときは常にディアサートされます。TX DMA リクエストは、THR または TX FIFO がエンプティのときは常にアサートされ、データが書き込まれているときは常にディアサートされます。
		1	DMA モード 1 では、RX DMA リクエストは、RX FIFO トリガ・レベルまたはタイムアウトに達したときは常にアサートされ、その後、RX FIFO がエンプティになるとディアサートされます。TX DMA リクエストは、TX FIFO がエンプティのときは常にアサートされ、その後、TX FIFO が完全にフルになるとディアサートされます。
2 (RX/W)	TFCLR	Tx FIFO をクリアします。	
1 (RX/W)	RFCLR	Rx FIFO をクリアします。	

表 17-11：UART_FCR レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
0 (R/W)	FIFOEN	16550 モードで動作するように FIFO をイネーブルにします。

割込みイネーブル

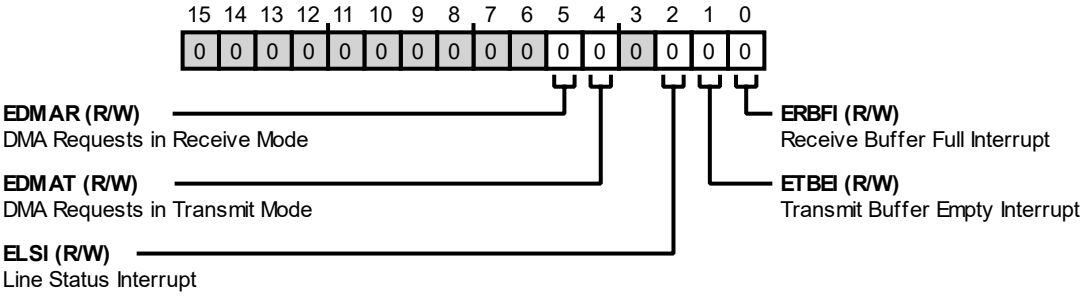


図 17-14：UART_IEN レジスタ図

表 17-12：UART_IEN レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
5 (R/W)	EDMAR	受信モードでの DMA リクエスト。	
		0	DMA リクエストを無効にします
		1	DMA リクエストを有効にします
4 (R/W)	EDMAT	送信モードでの DMA リクエスト。	
		0	DMA リクエストを無効にします
		1	DMA リクエストを有効にします
2 (R/W)	ELSI	ライン・ステータス割込み。	
		0	割込みをディスエーブルにします
		1	割込みをイネーブルにします
1 (R/W)	ETBEI	送信バッファ・エンプティ割込み。	
		0	割込みをディスエーブルにします
		1	割込みをイネーブルにします
0 (R/W)	ERBFI	受信バッファ・フル割込み。	
		0	割込みをディスエーブルにします
		1	割込みをイネーブルにします

割込み ID

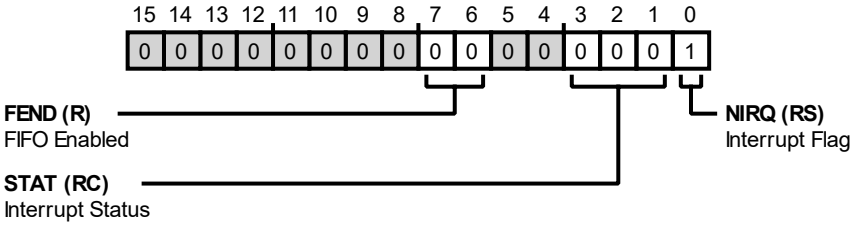


図 17-15：UART_IIR レジスタ図

表 17-13：UART_IIR レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
7:6 (R/NW)	FEND	FIFO のイネーブル。	
		0	FIFO はイネーブルになっていません、16450 モード
		3	FIFO はイネーブルになっています、16550 モード
3:1 (RC/NW)	STAT	割込みステータス。 UART_IIR.NIRQ がロー（アクティブ・ロー）のときは、割込みが発生していることを示します。 以下の UART_IIR.STAT ビットを使用してデコードします。	
		0	モデム・ステータス割込み（MSR レジスタを読み出すとクリアされます）
		1	送信バッファ・エンプティ割込み（Tx レジスタに書き込むか、または IIR レジスタを読み出すとクリアされます）
		2	受信バッファ・フル割込み（Rx レジスタを読み出すとクリアされます）
		3	受信ライン・ステータス割込み（LSR レジスタを読み出すとクリアされます）
		6	受信 FIFO タイムアウト割込み（Rx レジスタを読み出すとクリアされます）
0 (RS/NW)	NIRQ	割込みフラグ。	

ライン制御

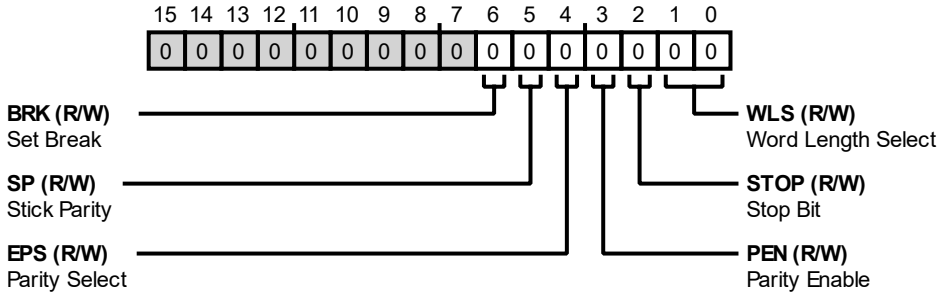


図 17-16：UART_LCR レジスタ図

表 17-14：UART_LCR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
6 (R/W)	BRK	ブレークの設定。	
		0	TxD は通常動作します
		1	TxD を強制的に 0 にします
5 (R/W)	SP	パリティの固定。 パリティを強制的に規定値に設定するために使用します。1 に設定すると、パリティは次のビット設定値に基づきます：UART_LCR.EPS = 1 かつ UART_LCR.PEN = 1 の場合、パリティは強制的に 0 になります。UART_LCR.EPS = 0 かつ UART_LCR.PEN = 1 の場合、パリティは強制的に 1 になります。UART_LCR.EPS = X かつ UART_LCR.PEN = 0 の場合、パリティは送信されません。	
		0	パリティは、パリティ・セレクト・ビットとパリティ・イネーブル・ビットに基づいて強制されることはありません。
		1	パリティは、パリティ・セレクト・ビットとパリティ・イネーブル・ビットに基づいて強制されます。
4 (R/W)	EPS	パリティ・セレクト。 パリティがイネーブルの（UART_LCR.PEN が 1 に設定されている）場合、このビットだけが意味を持ちます。	
		0	奇数パリティが送信されてチェックされます。
		1	偶数パリティが送信されてチェックされます。
3 (R/W)	PEN	パリティ・イネーブル。 送信されてチェックされるパリティ・ビットを制御するために使用されます。送信される値とチェックされる値は、UART_LCR.EPS と UART_LCR.SP の設定値に基づきます。	
		0	パリティは送信されることもチェックされることもありません。
		1	パリティは送信されてチェックされます。

表 17-14：UART_LCR レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
2 (R/W)	STOP	ストップ・ビット。 送信されるストップ・ビットの数を制御するために使用されます。すべての場合で、受信されたデータの最初のストップ・ビットだけが評価されます。	
		0	ワード長セレクト・ビットにかかわらず 1 ストップ・ビットを送信します。
		1	次のようなワード長に基づいて複数のストップ・ビットを送信します。WLS = 00 の場合は、1.5 ストップ・ビット（5 ビットのワード長）を送信します。WLS = 01、10、11 の場合はそれぞれ、2 ストップ・ビット（6、7、8 ビットのワード長）を送信します。
1:0 (R/W)	WLS	ワード長セレクト。 送信ごとのビット数を選択します。	
		0	5 ビット
		1	6 ビット
		2	7 ビット
		3	8 ビット

第 2 のライン制御

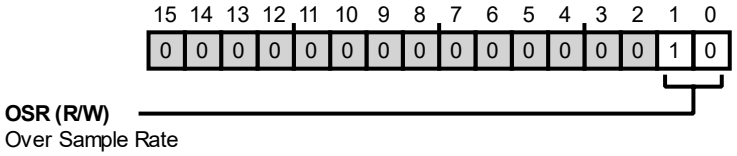


図 17-17：UART_LCR2 レジスタ図

表 17-15：UART_LCR2 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
1:0 (R/W)	OSR	オーバー・サンプル・レート。	
		0	4 倍のオーバー・サンプル。
		1	8 倍のオーバー・サンプル。
		2	16 倍のオーバー・サンプル。
		3	32 倍のオーバー・サンプル。

ライン・ステータス

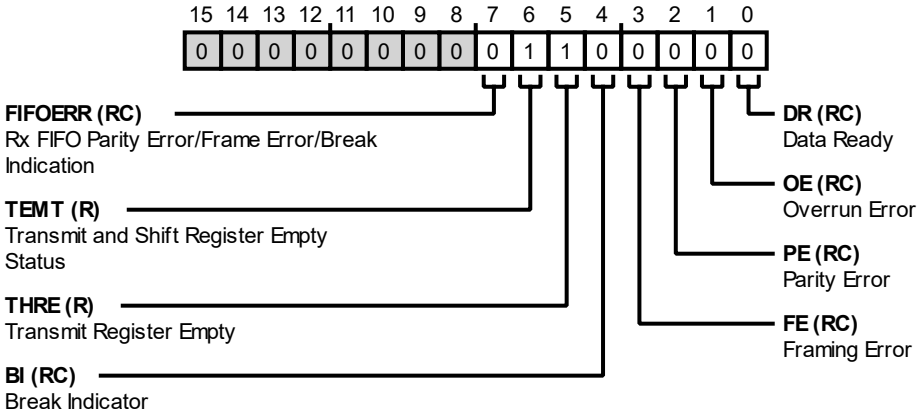


図 17-18：UART_LSR レジスタ図

表 17-16：UART_LSR レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
7 (RC/NW)	FIFOERR	Rx FIFO のパリティ・エラー／フレーム・エラー／ブレーク表示。 RX FIFO 内の 1 つまたは複数のデータ・バイトがパリティ・エラー、フレーム・エラー、またはブレーク表示になっています。16550 モードのみで使用します。RX FIFO 内にエラーがなくなった場合、読み出すとクリアされます。	
6 (R/NW)	TEMT	送信レジスタと送信シフト・レジスタのエンプティ・ステータス。	
		0	Tx レジスタとには、送信するデータが書き込まれて格納されています。その値に上書きしないように注意する必要があります。
		1	Tx レジスタと送信シフト・レジスタがエンプティであるため、新しいデータを Tx レジスタに書き込んでも安全です。データは送信済みです。
5 (R/NW)	THRE	送信レジスタ・エンプティ。 UART_RX が読み出されると、THRE はクリアされます。	
		0	Tx レジスタには、送信するデータが書き込まれて格納されています。その値に上書きしないように注意する必要があります。
		1	Tx レジスタがエンプティであるため、Tx レジスタに新しいデータを書き込んでも安全です。以前のデータは、まだ送信されていない可能性があり、シフト・レジスタ内にまだ存在している可能性があります。

表 17-16：UART_LSR レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
4 (RC/NW)	BI	ブレーク・インジケータ。 1 が設定されている場合、UART_LSR が読み出された後、このビットはセルフ・クリアされます。	
		0	最大ワード長よりも長い間、SIN が検出されませんでした。
		1	最大ワード長よりも長い間、SIN がローを維持しました。
3 (RC/NW)	FE	フレーミング・エラー。 1 が設定されている場合、UART_LSR が読み出された後、このビットはセルフ・クリアされます。	
		0	無効なストップ・ビットは検出されませんでした。
		1	受信されたワード上で無効なストップ・ビットが検出されました。
2 (RC/NW)	PE	パリティ・エラー。 1 が設定されている場合、UART_LSR が読み出された後、このビットはセルフ・クリアされます。	
		0	パリティ・エラーは検出されませんでした。
		1	受信されたワード上でパリティ・エラーが発生しました。
1 (RC/NW)	OE	オーバーラン・エラー。 1 が設定されている場合、UART_LSR が読み出された後、このビットはセルフ・クリアされます。	
		0	受信データは上書きされていません。
		1	受信データは、Rx レジスタから読み出される前に、新しいデータで上書きされました。
0 (RC/NW)	DR	データ・レディ。 このビットは、UART_RX が読み出されたときだけクリアされます。このビットは、セルフ・クリアされません。	
		0	Rx レジスタには新しい受信データがありません。
		1	Rx レジスタには読み出すべき受信データがあります。

RX FIFO のバイト・カウント

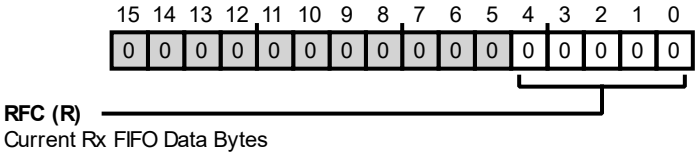


図 17-19：UART_RFC レジスタ図

表 17-17：UART_RFC レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
4:0 (R/NW)	RFC	現在の Rx FIFO のデータ・バイト。

RS485 の半二重制御

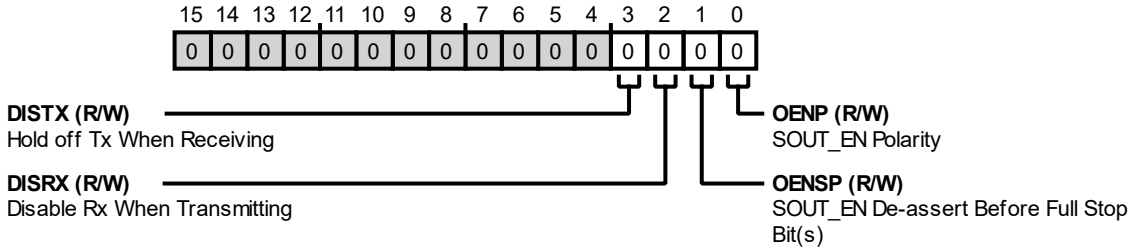


図 17-20：UART_RSC レジスタ図

表 17-18：UART_RSC レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
3 (R/W)	DISTX	受信時に Tx をホールド・オフにします。	
2 (R/W)	DISRX	送信時に Rx をディスエーブルにします。	
1 (R/W)	OENSP	1 つまたは複数のフル・ストップ・ビットよりも早い SOUT_EN のディアサート。	
		0	1 つまたは複数のフル・ストップ・ビットと同時に SOUT_EN をディアサートします
		1	1 つまたは複数のフル・ストップ・ビットよりも半ビット早く SOUT_EN をディアサートします
0 (R/W)	OENP	SOUT_EN の極性。	
		0	ハイ・アクティブ。
		1	ロー・アクティブ。

受信バッファ・レジスタ

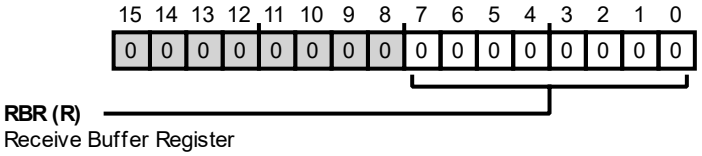


図 17-21：UART_RX レジスタ図

表 17-19：UART_RX レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (R/NW)	RBR	受信バッファ・レジスタ。

スクラッチ・バッファ

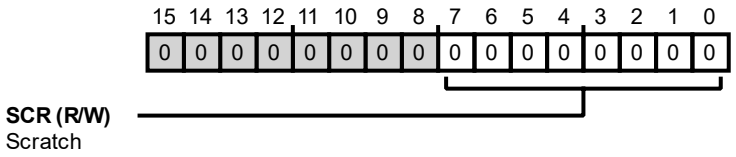


図 17-22：UART_SCR レジスタ図

表 17-20：UART_SCR レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (R/W)	SCR	スクラッチ。 中間結果を保存するために使用する 8 ビット・レジスタ。UART_SCR に格納される値は、UART の機能性や性能に影響を与えません。このレジスタの 8 ビットのみが実装されます。ビット［15:8］は読出し専用で、読み出すと常に 0x00 が返されます。このレジスタには、0～255 の任意の値が書込み可能です。読み出すと、最後に書き込まれた値が返されます。

TX FIFO のバイト・カウント

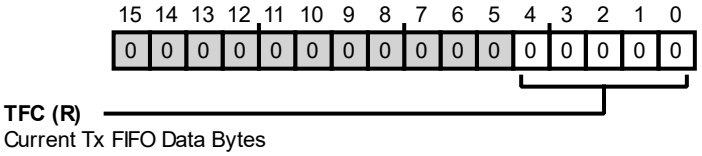


図 17-23：UART_TFC レジスタ図

表 17-21：UART_TFC レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
4:0 (R/NW)	TFC	現在の Tx FIFO のデータ・バイト。

送信保持レジスタ

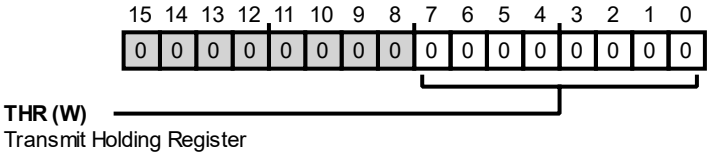


図 17-24：UART_TX レジスタ図

表 17-22：UART_TX レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (RX/W)	THR	送信保持レジスタ。

18 I2C（Inter-Integrated Circuit）インターフェース

ADuCM4050 MCU は、マスタとスレーブのどちらの機能にも設定できる I2C（Inter-Integrated Circuit）インターフェースを備えています。このペリフェラルは、**I2C Bus Specification バージョン 2.1** に準拠しています。

I2C の機能

ADuCM4050 MCU の I2C インターフェースは、以下に示す機能を備えています。

- マスタおよびスレーブで使用する、2 バイトの送信 FIFO および受信 FIFO。
- 反復開始に対応。
- 10 ビットのアドレス指定に対応。
- マスタ・アービトレーションをサポート。
- マスタの連続読出しモード、または最大 512 バイト固定の読出しモード。
- スレーブとマスタのクロック・ストレッチングをサポート。
- スレーブ・アドレスの設定：4 個の 7 ビット・アドレス、または 1 個の 10 ビット・アドレスと 2 個の 7 ビット・アドレスに設定可能。
- 内部および外部ループバックに対応。
- DMA に対応。
- バス・クリアに対応。

I2C 機能の説明

ここでは、ADuCM4050 MCU で使用する I2C 機能について説明します。

I2C のブロック図

I2C のブロック図を以下に示します。

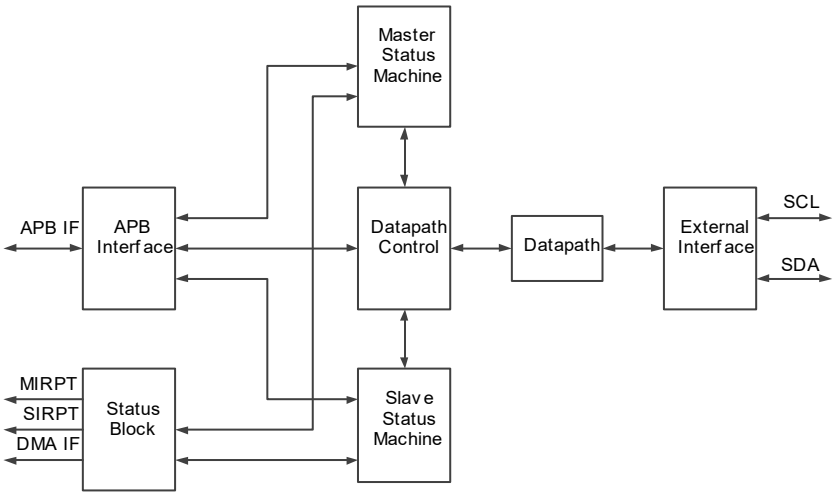


図 18-1：I2C のブロック図

I2C バス・ペリフェラルには、データ転送に使用する 2 本のピンがあります。SCL はシリアル・クロック・ピンで、SDA はシリアル・データ・ピンです。この 2 本のピンは、ワイヤード AND フォーマットで構成されており、マルチマスタ・システムでアービトレーションが可能です。

マスタ・デバイスを構成して、シリアル・クロックを生成できます。この周波数は、シリアル・クロック分周器レジスタでプログラミングします。マスタ・チャンネルは、高速モード（400kHz）または標準モード（100kHz）で動作するように設定できます。

I2C バス・システムにおける I2C バス・ペリフェラルのアドレスは、ユーザが設定します。この ID は、転送の実行中以外いつでも変更可能です。最大 4 個のスレーブ・アドレスを設定して、このペリフェラルに認識させることができます。ペリフェラルには、送信シフト・レジスタと受信シフト・レジスタのそれぞれに 2 バイトの FIFO が実装されています。FIFO の処理が必要なときは、コントロール・レジスタの IRQ ピンとステータス・ビットを使用して MCU コアに信号を送信します。

I2C の動作モード

I2C 通信に使用する GPIO は、I2C ペリフェラルをイネーブルする前に I2C モードに設定しなければなりません。

マスタ転送開始

マスタ・イネーブル・ビット（I2C_MCTL.MASEN）がセットされると、I2C_ADDR1 および I2C_ADDR2 レジスタに有効な値が書き込まれてマスタ転送シーケンスを開始します。I2C_MTX レジスタに有効なデータがあるときは、このデータが、書込みシーケンスにおいてアドレス・バイトに続いて転送される最初のバイトになります。

スレーブ転送開始

スレーブ・イネーブル・ビット（I2C_SCTL.SLVEN）がセットされると、スレーブ転送シーケンスは I2C_ID0、I2C_ID1、I2C_ID2、または I2C_ID3 レジスタのデバイス・アドレスをモニタします。デバイス・アドレスを認識すると、デバイスはスレーブ転送シーケンスに入ります。

スレーブ動作は常に、3 つの割込み源（I2C_MSTAT.MRXREQ/I2C_SSTAT.SRXREQ、I2C_MSTAT.MTXREQ/I2C_SSTAT.STXREQ、または I2C_SSTAT.GCINT）の 1 つがアサートされることによって開始されます。ソフトウェアは、

STOP 割込みを探してトランザクションが正しく完了したことを確認すると、STOP 割込みステータス・ビットをアサート解除します。

Rx/Tx データ FIFO

マスタとスレーブの送信データパスは、深さが 2 バイトの Tx FIFO、MTX と STX、送信シフタで構成されています。送信ステータス・ビット（I2C_MSTAT.MTXF と I2C_SSTAT.STXFSEREQ）は、Tx FIFO 内の有効なデータの有無を示します。シリアル・バイトの送信が開始されると、Tx FIFO から Tx シフタにデータがロードされます。転送シーケンスの実行中に Tx FIFO が一杯になっていないときは、送信要求ビット（I2C_MSTAT.MTXREQ または I2C_SSTAT.STXREQ）がアサートされます。

Tx シフタにロードされたときに有効な送信データがない場合は、スレーブは送信アンダーフロー・ステータス・ビット（I2C_SSTAT.STXUNDR）をアサートします。

マスタは、データ書込みを開始したときに送信 FIFO にデータがない場合、STOP 条件を生成します。

受信データパスは、マスタおよびスレーブの Rx FIFO（それぞれの深さは 2 バイト）と、I2C_MRX および I2C_SRX で構成されています。受信要求割込みビット（I2C_MSTAT.MRXREQ または I2C_SSTAT.SRXREQ）は、Rx FIFO 内の有効なデータの有無を示します。各バイトの受信後、データは Rx FIFO にロードされます。

Rx FIFO の有効なデータが Rx シフタによって上書きされると、受信オーバーフロー・ステータス・ビット（I2C_MSTAT.MRXOVR または I2C_SSTAT.SRXOVR）がアサートされます。

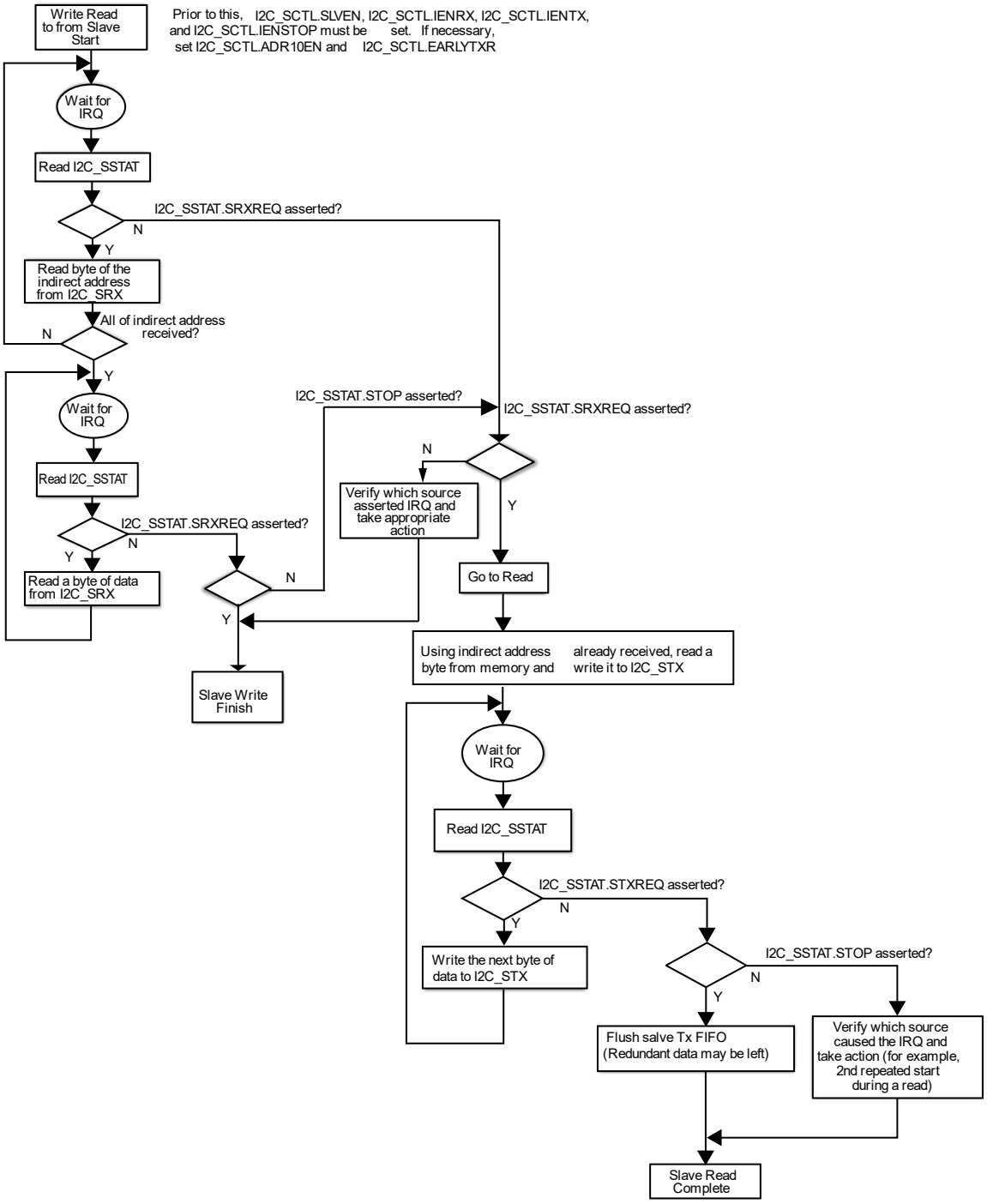


図 18-2：スレーブの読出し／書込みフロー

マスタによるノー・アクノレッジ

データの受信中にマスタの FIFO が一杯になり、更に FIFO へのバイトの書込みが試行されると、マスタは NACK で応答します。この最後の受信バイトは FIFO に書き込まれず、失われます。

スレーブによるノー・アクノレッジ

スレーブで読出しアクセスをアクノレッジしたくない場合は、スレーブ送信 FIFO にデータを書き込まないだけで NACK になります。

スレーブでマスタの書込みをアクノレッジしたくない場合は、スレーブ・コントロール・レジスタ内の I2C_SCTL.NACK ビットをアサートします。

通常、スレーブは受信 FIFO に書き込まれたバイトのすべてをアクノレッジします。受信 FIFO が一杯になるとスレーブはそれ以上バイトを書き込むことができず、FIFO に書き込めなかったバイトはアクノレッジされません。そして、マスタはトランザクションを停止する必要があります。

方向ビットが 1（読出し）で送信 FIFO が空の場合、スレーブはデバイス・アドレスが一致してもアクノレッジしません。そのため、マイクロコントローラがスレーブの送信要求に対応しアクノレッジをアサートする時間が短くなります。I2C_SCTL.EARLYTXR ビットをアサートすることを推奨します。

ジェネラル・コール

ジェネラル・コール・イネーブル・ビット（I2C_SCTL.GCEN）とスレーブ・イネーブル・ビット（I2C_SCTL.SLVEN）がセットされると、デバイスはジェネラル・コールに応答します。ジェネラル・コールの 2 番目のバイトが 0x06 の場合、I2C インターフェース（マスタおよびスレーブ）はリセットされます。そして、ジェネラル・コール割込みステータス・ビット（I2C_SSTAT.GCINT）がアサートされ、ジェネラル・コール ID ビット（I2C_SSTAT.GCID）が 0x1 にセットされます。ユーザ・コードを使用してシステム全体をリセットするか、I2C インターフェースを再度イネーブルする必要があります。

2 番目のバイトが 0x04（ハードウェアによるスレーブ・アドレスのプログラマブルな部分への書込み）の場合、ジェネラル・コール割込みステータス・ビット（I2C_SSTAT.GCINT）がアサートされ、ジェネラル・コール ID（I2C_SSTAT.GCID）が 0x2 にセットされます。

2 番目のバイトを受信すると、ジェネラル・コール割込みステータス・ビット（I2C_SSTAT.GCINT）はあらゆるジェネラル・コールに対してセットされます。ユーザ・コードは、デバイス・アドレスの再プログラミングなど、正しいアクションがとれるようにしなければなりません。

I2C_SCTL.GCEN がアサートされている場合、スレーブは常にジェネラル・コールの最初のバイトをアクノレッジします。2 番目のバイトが 0x04 または 0x06 の場合、あるいは 2 番目のバイトがハードウェア・ジェネラル・コールで、I2C_SCTL.HGCEN がアサートされている場合は、ジェネラル・コールの 2 番目のバイトをアクノレッジします。

I2C_ALT レジスタには、ハードウェア・ジェネラル・コール・シーケンス用の代替デバイス ID が含まれています。I2C_SCTL.HGCEN、I2C_SCTL.GCEN、または I2C_SCTL.SLVEN ビットがセットされると、デバイスはハードウェア・ジェネラル・コールを認識します。ジェネラル・コール・シーケンスが実行されると、シーケンスの 2 番目のバイトは ALT と同一となり、ハードウェア・コール・シーケンスはそのデバイスを認識します。

マスタによる反復開始の生成

マスタがトランザクションの処理でビジーなときに 1 番目のマスタ・アドレス・バイト・レジスタに書込みが行われると、マスタは反復開始を生成します。ステート・マシンがデバイス・アドレスの送信を開始した後であれば、1 番目のマスタ・アドレス・バイト・レジスタに書き込んでも問題ありません。

例えば、書込み-反復開始-読出し／書込みのトランザクションが要求された場合、ステート・マシンがデバイス・アドレスの送信を開始するか、最初の TXREQ 割込みを受信した後に、1 番目のマスタ・アドレス・バイト・レジスタに書き込みます。送信 FIFO が空になると、反復開始が生成されます。

同様に、読出し-反復開始-読出し／書込みのトランザクションが要求された場合も、ステート・マシンがデバイス・アドレスの送信を開始するか最初の RXREQ 割込みを受信した後に、1 番目のマスタ・アドレス・バイト・レジスタに書き込みます。必要な受信数に達すると、反復開始が生成されます。

DMA 要求

4 つの DMA チャンネル（マスタ用の MAS_DMA_RX_REQ および MAS_DMA_TX_REQ と、スレーブ用の SLV_DMA_RX_REQ および SLV_DMA_TX_REQ）が利用可能です。DMA イネーブル・ビットは、I2C_SCTL.STXDMA、I2C_SCTL.SRXDMA、I2C_MCTL.MTXDMA、および I2C_MCTL.MRXDMA レジスタにあります。

I2C リセット・モード

I2C_SCTL.SLVEN に 0 が書き込まれるとスレーブのステート・マシンが、I2C_MCTL.MASEN に 0 が書き込まれるとマスタのステート・マシンが、リセットされます。

I2C テスト・モード

I2C_MCTL.LOOPBACK ビットを設定することにより、デバイスを内部ループバック・モードにセットできます。4 つの FIFO（マスタの Tx と Rx、およびスレーブの Tx と Rx）があるため、I2C ペリフェラル同士で通信するようにセットアップできます。マスタがスレーブのアドレスを指定するようにセットアップすると、外部ループバックを実行できます。

I2C 低消費電力モード

マスタとスレーブの両方をディスエーブルすると（I2C_MCTL.MASEN = I2C_SCTL.SLVEN = 0）、デバイスは低消費電力モードになります。

自動クロック・ストレッチング

自動クロック・ストレッチングは、SCL ラインをローに保つことによってトランザクションを一時停止させます。トランザクションは、ラインがハイになるまで再開できません。

I2C スレーブが高速レートでデータを受信する場合、受信バイトを保存したり他の送信バイトを準備したりするために、より多くの時間が必要になることがあります。その場合、スレーブはバイトを受信してアクノレッジを返してから、SCL ラインをローに保つことができます。これにより、スレーブが次のバイト転送に対してレディ状態となるまで、マスタを待機状態にします。

MCU が低周波数で動作している場合や I2C 割込みの優先順位が低い場合、自動クロック・ストレッチング機能を使用して、I2C の割込み処理が完了するまで I2C バスを保持できます。

I2C のバス・クリア動作

外部のスレーブが SDA をローに保ったまま 0（または ACK）を送信する場合、このスレーブが SCL の次の立下がりエッジを取得するまで SDA は解放されません。その結果、バスはハングアップします。I2C マスタが新しい転送を開始しても SDA は送信されたアドレスと一致しないため、マスタはアービトレーションを失った状態になります。

I2C_MCTL.BUSCLR ビットがセットされているときにマスタがアービトレーションを失うと、マスタは SCL クロックを 9 回送信します。これにより、SDA ラインを保持しているスレーブは、この 9 回の SCL の内にそれを解放できるようになります。I2C_MCTL.STOPBUSCLR ビットがアサートされている場合、マスタは SDA が解放されると SCL クロックの送信を止めます。

パワーダウンに関する考慮事項

デバイスをパワーダウンして休止モードにする場合には、以下の点を考慮してください。マスタ／スレーブが IDLE になっている場合（それぞれのステータス・レジスタで確認できます）、マスタ・コントロール・レジスタの I2C_MCTL.MASEN ビット、またはスレーブ・コントロール・レジスタの I2C_SCTL.SLVEN ビットをクリアすることで、直ちにディスエーブルできます。

マスタ／スレーブがアクティブの場合、以下の 4 つのケースが考えられます。

- I2C がマスタで、Rx を実行している。

  この場合、デバイスは I2C_MRXCNT レジスタでプログラムされた数のデータを受信します。I2C_MRXCNT.EXTEND ビットがセットされている場合は、連続読出しモードになっています。読出し転送を停止するには、I2C_MRXCNT.EXTEND ビットをクリアして I2C_MRXCNT レジスタに I2C_MCRXCNT.VALUE + 1 を割り当てます。ここで、I2C_MCRXCNT.VALUE は現在の読出し回数です。

  「+1」は、完了するまでに余分に受信する必要があることを示します。新たにプログラムされた値が現在値より小さい場合は、現在値がオーバーフローして、プログラムされた回数に達するまで受信します。プログラムされた回数に達してから次のバイトを受信すると、転送を終了します。トランザクション完了割込みを受信したら、コアは I2C_MCTL.MASEN ビットをクリアして、マスタをディスエーブルしなければなりません。

- I2C がマスタで、Tx を実行している。

  ソフトウェアで I2C_STAT.MFLUSH ビットをセットして Tx-FIFO をフラッシュすると共に、I2C_MCTL.IENMTX ビットをクリアして Tx 要求をディスエーブルする必要があります。これにより、実行中のバイト送信が完了した後、転送を終了します。トランザクション完了割込みを受信したら、I2C_MCTL.MASEN ビットをクリアしなければなりません。

  注：完了前にマスタをディスエーブルすると、バスがハングアップしたままになる可能性があります。

- I2C がスレーブで、Rx を実行している。

  ソフトウェアで I2C_SCTL.NACK ビットをセットする必要があります。これにより、次の通信で NACK を生成することで、その後、外部マスタが STOP を生成することになります。STOP 割込みを受信すると、コアは I2C_SCTL.SLVEN ビットをクリアしてスレーブをディスエーブルします。

- I2C がスレーブで、Tx を実行している。

  スレーブ送信が開始されると、その後のトランザクションを NACK（ノー・アクノレッジ）することはできません（ACK はマスタによってのみ生成されます）。そのため、外部マスタが STOP 条件を生成するまで待つ必要があります。STOP 割込みを受信すると、スレーブをディスエーブルできます。これが本来の終了方法です。しかし、スレーブを直ちにディスエーブルしなければならない場合は、不正なデータ（すべて FF）を送信するだけで終了させることができます。これは、SDA ラインがこれ以上駆動されずにデータ・フェーズの間プルアップされてしまうためです。この場合、バスはハングアップしません。

I2C データ転送

I2C ペリフェラルは、データ転送をコア・モードと DMA モードの両方にプログラムできます。マスタおよびスレーブ機能専用の DMA チャンネルを備えています。DMA チャンネル番号については、ダイレクト・メモリ・アクセス（DMA）を参照してください。

I2C 割込みおよび例外

I2C ブロックは、マスタおよびスレーブ・モードにおいて、様々な条件で割込みを生成できます。

ADuCM4050 の I2C レジスタの説明

I2C マスタ／スレーブ（I2C）は以下のレジスタを備えています。

表 18-1：ADuCM4050 の I2C レジスター覧

レジスタ名	説明
I2C_ADDR1	マスタ・アドレス・バイト 1
I2C_ADDR2	マスタ・アドレス・バイト 2
I2C_ALT	ハードウェア・ジェネラル・コール ID
I2C_ASTRETCH_SCL	SCL の自動ストレッチング
I2C_BYT	開始バイト
I2C_DIV	シリアル・クロック分周器
I2C_ID0	1 番目のスレーブ・アドレス・デバイス ID
I2C_ID1	2 番目のスレーブ・アドレス・デバイス ID
I2C_ID2	3 番目のスレーブ・アドレス・デバイス ID
I2C_ID3	4 番目のスレーブ・アドレス・デバイス ID
I2C_MCTL	マスタ・コントロール
I2C_MCRXCNT	現在のマスタ受信データ・カウント
I2C_MRX	マスタ受信データ
I2C_MRXCNT	マスタ受信データ・カウント
I2C_MSTAT	マスタ・ステータス
I2C_MTX	マスタ送信データ
I2C_SCTL	スレーブ・コントロール
I2C_SHCTL	共通コントロール
I2C_SRX	スレーブ受信
I2C_SSTAT	スレーブの I2C ステータス／エラー／IRQ
I2C_STAT	マスタおよびスレーブ FIFO ステータス
I2C_STX	スレーブ送信
I2C_TCTL	タイミング・コントロール・レジスタ

マスタ・アドレス・バイト 1

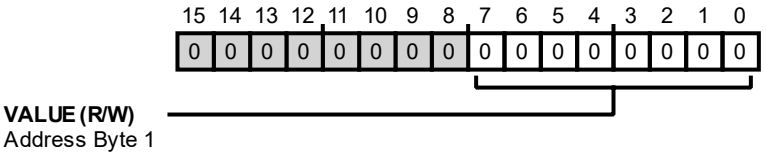


図 18-3：I2C_ADDR1 のレジスタ図

表 18-2：I2C_ADDR1 のレジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
7:0 (R/W)	VALUE	アドレス・バイト 1。 7 ビット・アドレスを使用する場合は、I2C_ADDR1.VALUE のビット 7 からビット 1 にアドレスがプログラムされ、I2C_ADDR1.VALUE のビット 0 に方向（読出し、または書込み）がプログラムされます。 10 ビット・アドレスを使用する場合は、I2C_ADDR1.VALUE のビット 7 からビット 3 に 11110、I2C_ADDR1.VALUE のビット 2 からビット 1 にアドレスの 2 個の MSB、I2C_ADDR1.VALUE のビット 0 に 0 がプログラムされます。

マスタ・アドレス・バイト 2

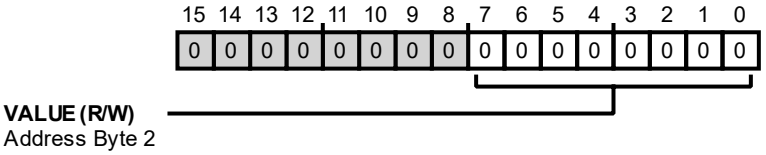


図 18-4：I2C_ADDR2 のレジスタ図

表 18-3：I2C_ADDR2 のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (R/W)	VALUE	アドレス・バイト 2。 このレジスタは、10 ビット・アドレスでスレーブのアドレス指定を行う場合にのみ使用します。`I2C_ADDR2.VALUE` のビット 7 からビット 0 にアドレスの下位 8 ビットがプログラムされます。

ハードウェア・ジェネラル・コール ID

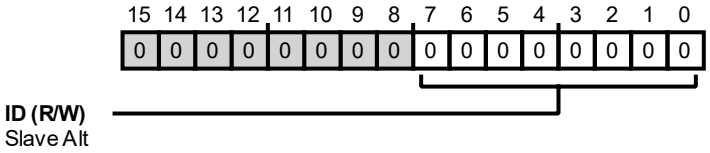


図 18-5：I2C_ALT のレジスタ図

表 18-4：I2C_ALT のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (R/W)	ID	スレーブの代替アドレス。 このレジスタは、I2C_SCTL.HGCEN と共に使用して、ハードウェア・ジェネラル・コールを生成しているマスタを検索します。マスタ・デバイスにスレーブ・アドレスがプログラムされていない状態で、スレーブがマスタ・アドレスを認識しなければならない場合に使用します。

SCL の自動ストレッチング

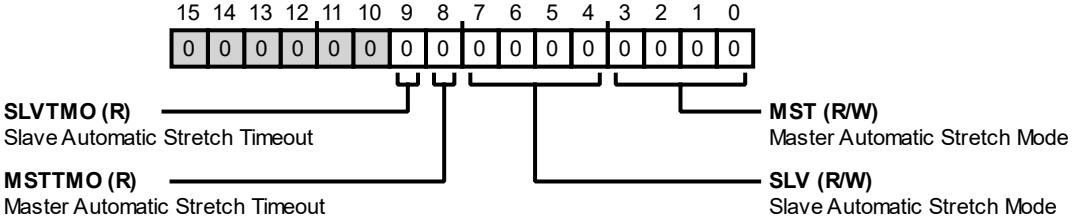


図 18-6：I2C_ASTRETCH_SCL のレジスタ図

表 18-5：I2C_ASTRETCH_SCL のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
9 (R/NW)	SLVTMO	スレーブの自動ストレッチング・タイムアウト。 スレーブの自動ストレッチング・タイムアウトが発生するとアサートされ、ビットが読み出されるとクリアされます。
8 (R/NW)	MSTTMO	マスタの自動ストレッチング・タイムアウト。 マスタの自動ストレッチング・タイムアウトが発生するとアサートされ、ビットが読み出されるとクリアされます。
7:4 (R/W)	SLV	スレーブの自動ストレッチング・モード。 スレーブの自動ストレッチング・モードを定義します。スレーブの送信時、アドレス・バイトに対する ACK/NACK を送信する前、またはデータ・バイトにデータを送信する前にスレーブ Tx FIFO が空になっている場合、SCL クロックは SCL の立下がりエッジから自動ストレッチングを開始します。 スレーブ Tx FIFO が空でなくなった場合、またはタイムアウトが発生した場合は、ストレッチングを停止します。スレーブの受信時、ACK/NACK を送信する前にスレーブ Rx FIFO がオーバーフローすると、SCL クロックは SCL の立下がりエッジから自動ストレッチングを開始します。スレーブ Rx FIFO のオーバーフローが解消された場合、またはタイムアウトが発生した場合は、ストレッチングを停止します。 I2C_SSTAT.STXREQ がアサートされると、スレーブのクロック・ストレッチング・モードは自動的に無効になります。I2C_SSTAT.STXREQ がクリアされると、ストレッチング・モードは前に設定された値に戻ります。 注：このビットが 4b0000 になっている場合は、I2C_SCTL.EARLYTXR を 1 にセットする必要があります。そうでない場合、CPU はアドレスを受信して読出し要求を行うときにスレーブ送信割込みを処理できません。

表 18-5：I2C_ASTRETCH_SCL のレジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応
3:0 (R/W)	MST	マスタの自動ストレッチング・モード。 マスタの自動ストレッチング・モードを定義します。ストレッチングは SCL ラインをローに保持することなので、割込みを処理するには時間がかかります。また、バスのロックアップを防ぐにはタイムアウトを使用します。 0000：SCL クロック・ストレッチングを行わない。 0001：最大 2^1 = 2 ビット時間の SCL ストレッチング 0010：最大 2^2 = 4 ビット時間の SCL ストレッチング 1110：最大 2^14 ビット時間の SCL ストレッチング 1111：SCL ストレッチングは無限大（タイムアウトなし） ビット時間は I2C_DIV.HIGH + I2C_DIV.LOW で決まり、PCLK によってカウントされます。 タイムアウトの最大値 = 2^14/400kHz = 40ms。SMBus（タイムアウトは 35ms）と互換性があります。 マスタの送信時、データ送信前にマスタ Tx FIFO が空のとき、SCL クロックは SCL の立下がりエッジから自動ストレッチングを開始します。マスタ Tx FIFO が空でなくなった場合、またはタイムアウトが発生した場合、ストレッチングを停止します。 マスタの受信時、ACK/NACK を送信する前にマスタ Rx FIFO がオーバーフローすると、SCL クロックは SCL の立下がりエッジから自動ストレッチングを開始します。マスタ Rx FIFO のオーバーフローが解消された場合、またはタイムアウトが発生した場合は、ストレッチングを停止します。 I2C_MSTAT.MTXREQ がアサートされた場合、または新しいアドレスが書き込まれた場合に、マスタのクロック・ストレッチング・モードは自動的に無効になります。I2C_MSTAT.MTXREQ がクリアされるかアドレスがロードされる（RESTART が送信される）と、ストレッチング・モードは前に設定された値に戻ります。

開始バイト

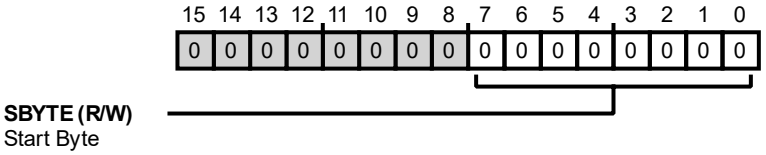


図 18-7：I2C_BYT のレジスタ図

表 18-6：I2C_BYT のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (R/W)	SBYTE	開始バイト。 トランザクションの開始時に開始バイトを生成するために使用します。開始バイトに続いて通常のアドレスを生成するには、最初に I2C_BYT.SBYTE に書き込んでから、アドレス・レジスタ（I2C_ADDR1）に書き込みます。これにより、バス上で I2C_BYT.SBYTE に書き込まれたバイトとその後に続く反復開始が生成されます。このレジスタは、（開始バイト 00000001 だけでなく）I2C バス上で反復開始の前のあらゆるバイトを生成するために使用できます。

シリアル・クロック分周器

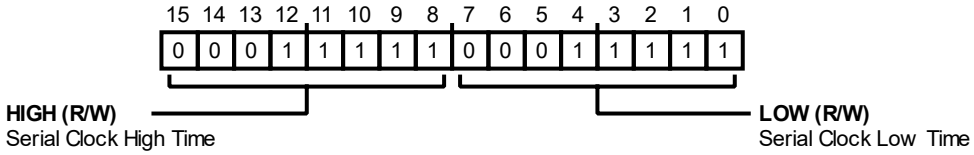


図 18-8：I2C_DIV のレジスタ図

表 18-7：I2C_DIV のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:8 (R/W)	HIGH	シリアル・クロック・ハイ時間。 このレジスタは、クロック・ハイの時間を制御します。タイマーはコア・クロック（UCLK）によって生成されます。次式を使って必要なハイ時間を求めます。 I2C_DIV.HIGH = （REQD_HIGH_TIME/PCLK_PERIOD） − 2 例えば、50MHz のコア・クロック周波数、1300ns のロー時間と 1200ns のハイ時間で 400kHz の SCL を生成する場合： LOTIME = 1300ns/20ns − 1 = 0x40（10 進法で 64） I2C_DIV.HIGH = 1200ns/20ns − 2 = 0x3A（10 進法で 58）。 このレジスタを 0x1F にリセットすると、SCL のハイ時間は PCLK の刻みで 33 になります。tHD：STA も次式のように I2C_DIV.HIGH で決まります。 tHD：STA = （HIGH − 1）× pclk_period。 tHD：STA は 600ns にする必要があるため、UCLK = 50MHz では、I2C_DIV.HIGH の最小値は 31 になります。これにより、SCL のハイ時間は 660ns になります。
7:0 (R/W)	LOW	シリアル・クロック・ロー時間。 このレジスタは、クロック・ローの時間を制御します。タイマーはコア・クロック（UCLK）によって生成されます。次式を使って必要なロー時間を求めます。 I2C_DIV.LOW = （REQD_LOW_TIME/PCLK_PERIOD） − 1 このレジスタを 0x1F にリセットすると、SCL のロー時間は PCLK の刻みで 32 になります。

1 番目のスレーブ・アドレス・デバイス ID

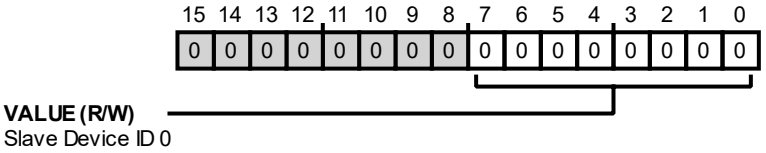


図 18-9：I2C_ID0 のレジスタ図

表 18-8：I2C_ID0 のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (R/W)	VALUE	スレーブ・デバイス ID 0。 I2C_ID0.VALUE［7：1］にデバイス ID をプログラムします。I2C_ID0.VALUE［0］はドント・ケアです。このレジスタを 10 ビット・アドレスでプログラムする方法については、I2C_SCTL.ADR10EN ビットを参照してください。

2 番目のスレーブ・アドレス・デバイス ID

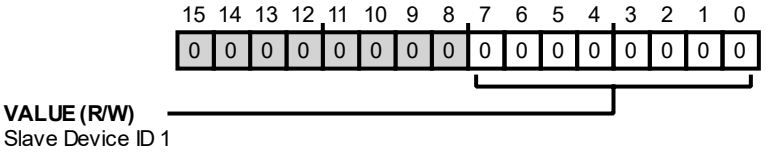


図 18-10：I2C_ID1 のレジスタ図

表 18-9：I2C_ID1 のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (R/W)	VALUE	スレーブ・デバイス ID 1。 I2C_ID1.VALUE［7：1］にデバイス ID をプログラムします。I2C_ID1.VALUE［0］はドント・ケアです。このレジスタを 10 ビット・アドレスでプログラムする方法については、I2C_SCTL.ADR10EN ビットを参照してください。

3 番目のスレーブ・アドレス・デバイス ID

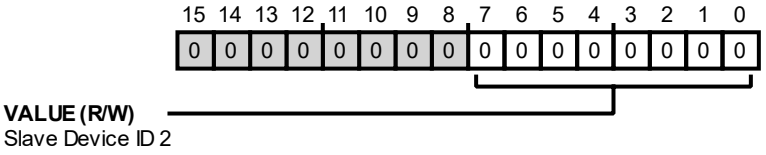


図 18-11：I2C_ID2 のレジスタ図

表 18-10：I2C_ID2 のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (R/W)	VALUE	スレーブ・デバイス ID 2。 I2C_ID2.VALUE［7：1］にデバイス ID をプログラムします。I2C_ID2.VALUE［0］はドント・ケアです。このレジスタを 10 ビット・アドレスでプログラムする方法については、I2C_SCTL.ADR10EN ビットを参照してください。

4 番目のスレーブ・アドレス・デバイス ID

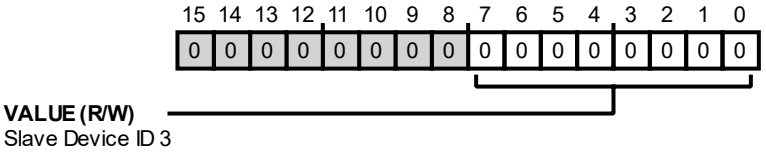


図 18-12：I2C_ID3 のレジスタ図

表 18-11：I2C_ID3 のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (R/W)	VALUE	スレーブ・デバイス ID 3。 I2C_ID3.VALUE［7：1］にデバイス ID をプログラムします。I2C_ID3.VALUE［0］はドント・ケアです。このレジスタを 10 ビット・アドレスでプログラムする方法については、I2C_SCTL.ADR10EN ビットを参照してください。

マスタ・コントロール

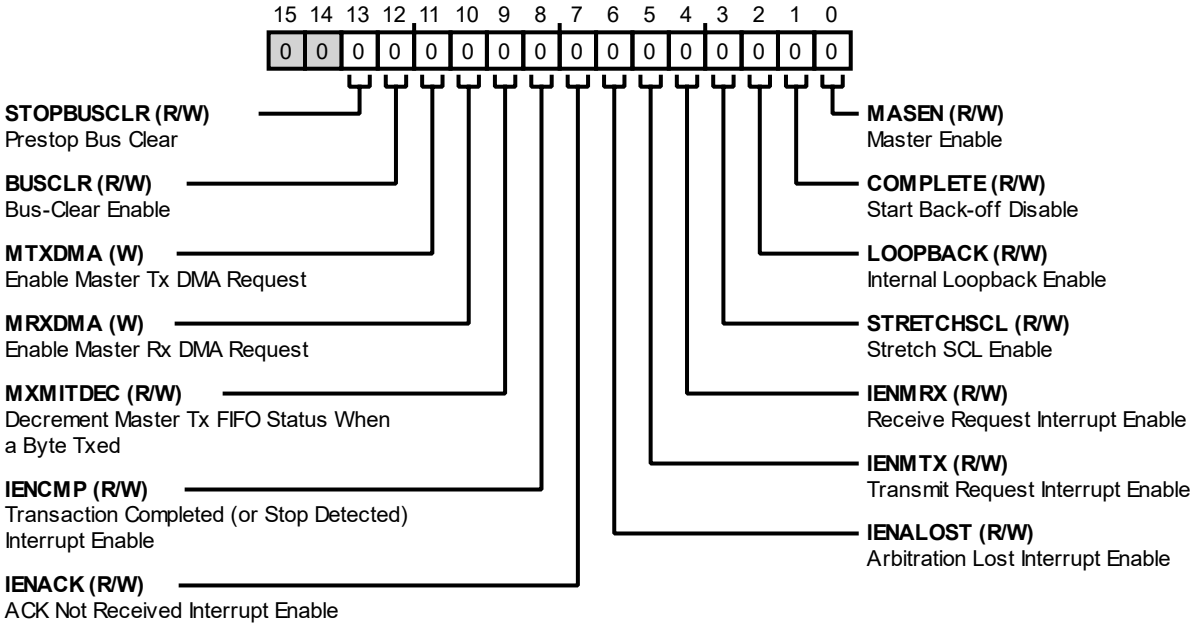


図 18-13：I2C_MCTL のレジスタ図

表 18-12：I2C_MCTL のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
13 (R/W)	STOPBUSCLR	バス・クリア事前停止。 このビットは I2C_MCTL.BUSCLR と組み合わせて使用します。このビットがセットされている場合、SCL クロックが 9 回送信される前に SDA が解放されると、マスタはそれ以上の SCL クロック送信を停止します。
12 (R/W)	BUSCLR	バス・クリア・イネーブル。 このビットがセットされている場合、マスタのアービトレーションが失われると、SCL を最大 9 回送信することによりマスタはバス・クリア動作を開始します。このビットは、スレーブの誤動作によって SDA が動かなくなってしまった状態から抜け出すために追加されます。したがって、注意して使用しなければなりません。他にアクティブな I2C マスタがない場合にのみ、このビットをセットします。
11 (RX/W)	MTXDMA	マスタ Tx DMA 要求イネーブル。 ユーザ・コードで 1 にセットすることで、I2C のマスタ DMA Tx 要求を有効にします。ユーザ・コードでクリアすることにより、DMA モードを無効にします。
10 (RX/W)	MRXDMA	マスタ Rx DMA 要求イネーブル。 ユーザ・コードで 1 にセットすることで、I2C のマスタ DMA Rx 要求を有効にします。ユーザ・コードでクリアすることにより、DMA モードを無効にします。

表 18-12：I2C_MCTL のレジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
9 (R/W)	MXMITDEC	バイト送信時のマスタ Tx FIFO ステータス・デクリメント。 バイトの送信が完了すると、マスタ Tx FIFO ステータスをデクリメントします。このビットが 1 にセットされている場合、バイトの送信が完了すると、マスタ送信 FIFO ステータスはデクリメントされます。0 にセットされている場合、バイトの送信が開始されたときにバイトが FIFO からシャドウ・レジスタにアンロードされると、マスタ送信 FIFO ステータスがデクリメントされます。
8 (R/W)	IENCMP	トランザクション完了（または STOP 条件検出）割込みイネーブル。 トランザクション完了割込みをイネーブルします。このビットがアサートされている場合、STOP 条件を検出すると割込みを生成します。
7 (R/W)	IENACK	ACK 未受信割込みイネーブル。
6 (R/W)	IENALOST	アービトレーション喪失割込みイネーブル。
5 (R/W)	IENMTX	送信要求割込みイネーブル。
4 (R/W)	IENMRX	受信要求割込みイネーブル。
3 (R/W)	STRETCHSCL	SCL ストレッチング・イネーブル。 このビットをセットすると、SCL が 0 のときは 0 に保持、SCL が 1 のときは次に 0 になったら 0 に保持するようにデバイスに指示します。
2 (R/W)	LOOPBACK	内部ループバック・イネーブル。 このビットをセットすると、SCL ラインを SCL 入力に、SDA ラインを SDA 入力にマルチプレクスします。デバイス・アドレスが対応している限り、マスタがスレーブに転送をループバックする、すなわち外部ループバックを行うことも可能です。
1 (R/W)	COMPLETE	バックオフ・スタート・ディスエーブル このビットをセットすると、デバイスは、他のデバイスが START 条件を生成しているときでもバスの使用権取得を試行します。
0 (R/W)	MASEN	マスタ・イネーブル。 このビットが 1 の場合、マスタがイネーブルされます。このビットが 0 の場合、すべてのマスタのステート・マシンのフロップがリセット状態に保持され、マスタはディスエーブルされます。マスタを使用しない場合は、ディスエーブルすることによりマスタのクロックをゲーティングできるため、消費電力を削減できます。このビットはトランザクションが完了するまでクリアしないようにしてください（I2C_MSTAT.TCOMP 参照）。APB 書込み可能レジスタのビットは、このビットによってリセットされません。デフォルトではクリアされています。

現在のマスタ受信データ・カウント

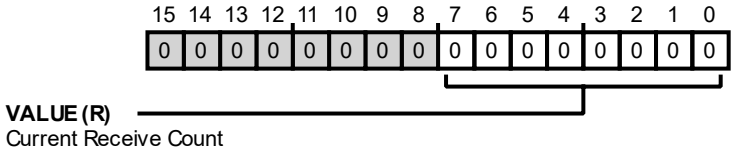


図 18-14：I2C_MCRXCNT のレジスタ図

表 18-13：I2C_MCRXCNT のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (R/NW)	VALUE	現在の受信カウント。 このレジスタは、受信したバイトの総数を示します。トランザクションの完了時に 256 バイトになる場合、このレジスタは 0 を読み出します。

マスタ受信データ

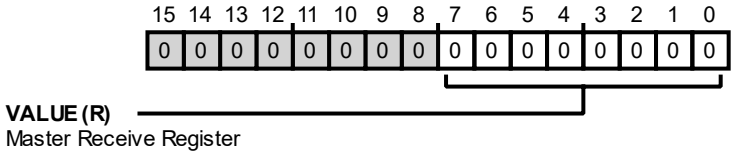


図 18-15：I2C_MRX のレジスタ図

表 18-14：I2C_MRX のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (R/NW)	VALUE	マスタ受信レジスタ。 このレジスタにより、受信データ FIFO にアクセスできるようになります。FIFO は 2 バイトを保持できます。

マスタ受信データ・カウント

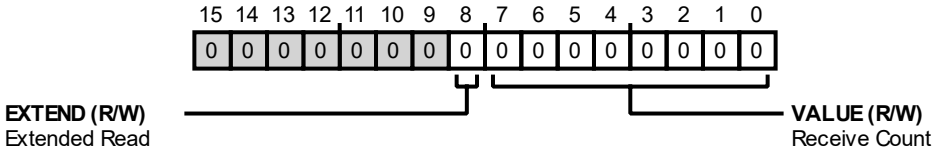


図 18-16：I2C_MRXCNT のレジスタ図

表 18-15：I2C_MRXCNT のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
8 (R/W)	EXTEND	拡張読出し。 256 より大きいバイト数を読み出す必要があるときに、このビットを使用します。例えば、412 バイトを受信する場合は、I2C_MRXCNT.EXTEND に 1 を書き込みます。最初のバイトを受信するまで待ってから、その後、バイトを受信するたびに I2C_MCRXCNT レジスタをチェックします。I2C_MRXCNT.VALUE が 0 に戻っている場合は、256 バイトが受信されたことになります。次いで、I2C_MRXCNT レジスタに 0x09C が書き込まれます。
7:0 (R/W)	VALUE	受信カウント。 必要なバイト数 − 1 をこのレジスタにプログラムします。必要なバイト数が 1 のときは、0 を書き込みます。256 より大きなバイト数が必要な場合は、I2C_MRXCNT.EXTEND を使用します。

マスタ・ステータス

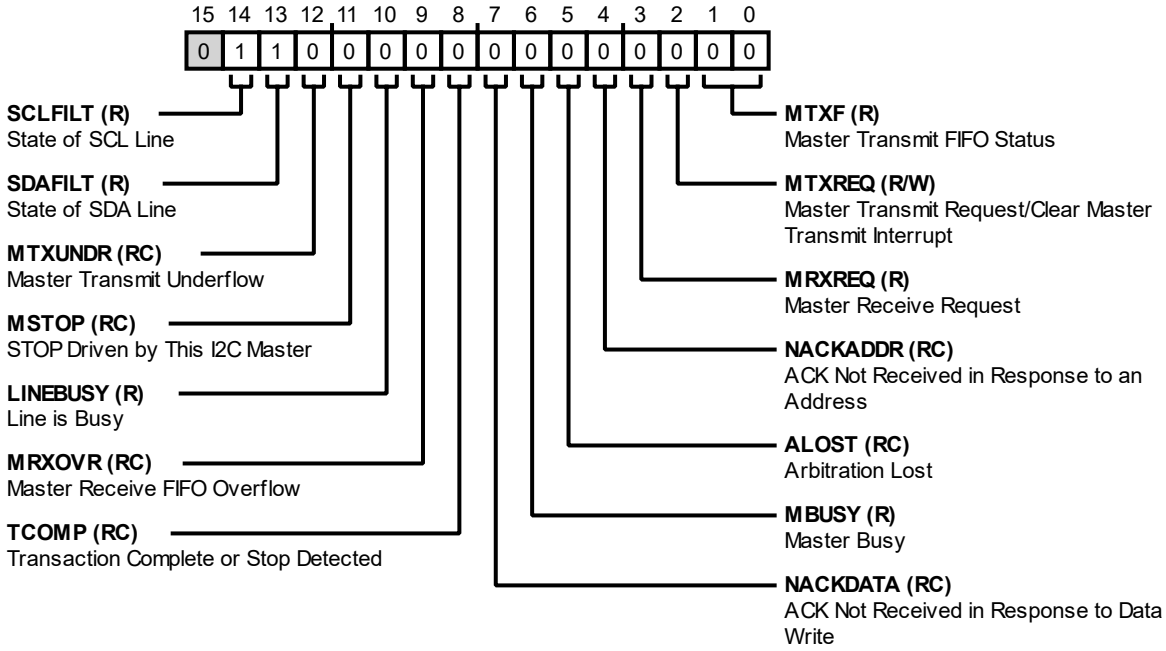


図 18-17：I2C_MSTAT のレジスタ図

表 18-16：I2C_MSTAT のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
14 (R/NW)	SCLFILT	SCL ラインの状態。 このビットは、SCL のグリッチ・フィルタ出力です。SCL は、駆動されていない状態では常にハイにプルアップされています。
13 (R/NW)	SDAFILT	SDA ラインの状態。 このビットは、SDA のグリッチ・フィルタ出力です。SDA は、駆動されていない状態では常にハイにプルアップされています。
12 (RC/NW)	MTXUNDR	マスタ送信アンダーフロー。 Tx FIFO が空の状態のため I2C マスタがトランザクションを停止すると、アサートされます。このビットは、I2C_MCTL.IENMTX ビットがセットされている場合にのみアサートされます。
11 (RC/NW)	MSTOP	この I2C マスタが生成する STOP。 この I2C マスタが I2C バスの STOP 条件を生成すると、アサートされます。このビットがアサートされている場合、トランザクションの完了、Tx のアンダーフロー、Rx のオーバーフロー、またはスレーブによる NACK を表します。他の I2C マスタによる STOP 条件の発生ではアサートされない点で、このビットは I2C_MSTAT.TCOMP と異なります。このビットがアサートされても割込みは生成されません。しかし、I2C_MCTL.IENCMP = 1 の場合、STOP 条件発生のたびに割込みが生成され、このビットを読み出すことができます。 このビットは、読み出されるとステータスをクリアします。

表 18-16：I2C_MSTAT のレジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
10 (R/NW)	LINEBUSY	ライン・ビジー。 I2C バスで START を検出するとアサートされます。I2C バスで STOP を検出すると、アサート解除されます。
9 (RC/NW)	MRXOVR	マスタ受信 FIFO オーバーフロー。 既に一杯になっている受信 FIFO に新たにバイトが書き込まれると、アサートされます。このビットは、読み出されるとステータスをクリアします。
8 (RC/NW)	TCOMP	トランザクション完了、または STOP 検出。 トランザクション完了。I2C バスで STOP 条件を検出すると、このビットがアサートされます。I2C_MCTL.IENCMP = 1 の場合にこのビットがアサートされると、割込みが生成されます。このビットは、マスタがイネーブルの場合（I2C_MCTL.MASEN = 1）にのみアサートされます。 このビットを使用して、マスタを安全にディスエーブルできるタイミングを決定します。このマスタがアービトレーションを失った場合には、このビットを使用して I2C バス上にある他のマスタのトランザクションが完了するまで待機させることもできます。このビットは、読み出されるとステータスをクリアします。このビットは割込みを生成できます。
7 (RC/NW)	NACKDATA	データ書込みに対応する ACK 未受信。 データ書込み転送に対応する ACK の受信がない場合に、このビットがアサートされます。I2C_MCTL.IENACK = 1 の場合、このビットがアサートされると割込みが生成されます。 このビットは割込みを生成できます。I2C_MSTAT レジスタを読み出すことで、このビットはクリアされます。
6 (R/NW)	MBUSY	マスタ・ビジー。 このビットは、マスタのステート・マシンがトランザクションを処理していることを示します。ステート・マシンがアイドル状態になるか、他のデバイスが I2C バスを制御するとクリアされます。
5 (RC/NW)	ALOST	アービトレーション喪失。 マスタがアービトレーションを失うとこのビットがアサートされます。I2C_MCTL.IENALOST = 1 の場合、このビットがアサートされると割込みが生成されます。I2C_MSTAT レジスタを読み出すことで、このビットはクリアされます。このビットは割込みを生成できます。
4 (RC/NW)	NACKADDR	アドレスに対応する ACK 未受信。 アドレスに対応する ACK の受信がない場合に、このビットがアサートされます。I2C_MCTL.IENACK = 1 の場合、このビットがアサートされると割込みが生成されます。I2C_MSTAT レジスタを読み出すことで、このビットはクリアされます。このビットは割込みを生成できます。
3 (R/NW)	MRXREQ	マスタ受信要求。 受信 FIFO にデータがあると、このビットがアサートされます。I2C_MCTL.IENMRX = 1 の場合、このビットがアサートされると割込みが生成されます。 このビットは割込みを生成できます。
2 (R/W)	MTXREQ	マスタ送信要求／マスタ送信割込みクリア。 読出しのときはマスタ送信要求、書込みのときはマスタ送信割込みビットのクリアが行われます。方向ビットが 0 で送信 FIFO が一杯になっていないとき、このビットがアサートされます。I2C_MCTL.IENMTX = 1 の場合、このビットがアサートされると割込みを生成します。 このビットに 1 が書き込まれると、マスタ送信割込みとクロック・ストレッチングはクリアされます。

表 18-16：I2C_MSTAT のレジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
1:0 (R/NW)	MTXF	マスタ送信 FIFO ステータス。 この 2 ビットは、マスタ送信 FIFO のステータスを示します。	
		0	FIFO は空。
		2	FIFO 内に 1 バイト。
		3	FIFO はフル。

マスタ送信データ

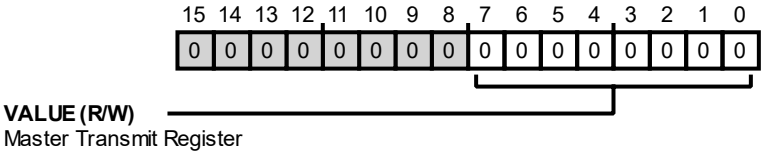


図 18-18：I2C_MTX のレジスタ図

表 18-17：I2C_MTX のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (R/W)	VALUE	マスタ送信レジスタ。 テストやデバッグに使用します。このレジスタが読み出されると、現在マスタが送信しているバイトを返します。送信レジスタに書き込まれたバイトがライン上で送信されているときは、しばらくしてからリードバックすることができます。このレジスタにより、送信データ FIFO にアクセスできるようになります。FIFO は 2 バイトを保持できます。

スレーブ・コントロール

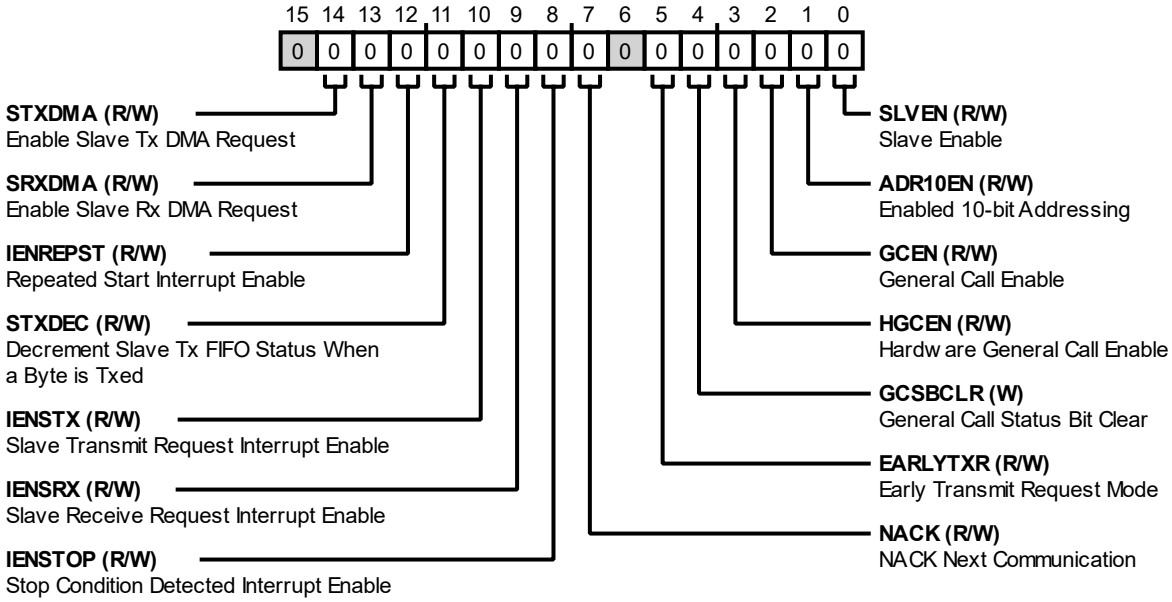


図 18-19：I2C_SCTL のレジスタ図

表 18-18：I2C_SCTL のレジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
14 (R/W)	STXDMA	スレーブ Tx DMA 要求イネーブル。 ユーザ・コードで 1 にセットすることで、I2C のスレーブ DMA Rx 要求を有効にします。ユーザ・コードでクリアすることにより、DMA モードを無効にします。
13 (R/W)	SRXDMA	スレーブ Rx DMA 要求イネーブル。 ユーザ・コードで 1 にセットすることで、I2C のスレーブ DMA Rx 要求を有効にします。ユーザ・コードでクリアすることにより、DMA モードを無効にします。
12 (R/W)	IENREPST	反復開始割込みイネーブル。 このビットが 1 の場合、`I2C_SSTAT.REPSTART` ステータス・ビットがアサートされると割込みが生成されます。0 の場合、`I2C_SSTAT.REPSTART` ステータス・ビットがアサートされても割込みは生成されません。
11 (R/W)	STXDEC	バイト送信時のスレーブ Tx FIFO ステータス・デクリメント。 バイトの送信が完了すると、スレーブ Tx FIFO ステータスをデクリメントします。このビットが 1 にセットされている場合、バイトの送信が完了すると、送信 FIFO ステータスはデクリメントされます。0 にセットされている場合、バイトの送信開始時にバイトが FIFO からシャドウ・レジスタにアンロードされると、送信 FIFO ステータスがデクリメントされます。
10 (R/W)	IENSTX	スレーブ送信要求割込みイネーブル。

表 18-18：I2C_SCTL のレジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応
9 (R/W)	IENSRX	スレーブ受信要求割込みイネーブル。
8 (R/W)	IENSTOP	STOP 条件検出割込みイネーブル。
7 (R/W)	NACK	次の通信の NACK。 このビットがセットされると、次の通信は NACK されます。これを使用できるのは、例えば、24xx 形式のアクセスが行われているときに、システム・メモリ内の読出し専用位置や存在しない位置への書込みが試行された場合です。これは、24xx 形式の書込みにおいて、書込みできないメモリ位置を指定した間接アドレスです。
5 (R/W)	EARLYTXR	早期送信要求モード。 このビットをセットすると、方向ビットの SCL クロック・パルスの立上がりエッジの直後に送信要求が有効になります。
4 (RX/W)	GCSBCLR	ジェネラル・コール・ステータス・ビット・クリア このビットに 1 が書き込まれると、I2C_SSTAT.GCINT ビットと I2C_SSTAT.GCID ビットがクリアされます。I2C_SSTAT.GCINT ビットと I2C_SSTAT.GCID ビットは、このビットに書き込むかフル・リセットする以外の方法ではリセットできません。
3 (R/W)	HGCEN	ハードウェア・ジェネラル・コール・イネーブル。 このビットと I2C_SCTL.GCEN ビットがセットされているときに、ジェネラル・コール（アドレス 00h）と 1 つのデータ・バイトを受信すると、デバイスは I2C_ALT の内容と受信シフト・レジスタの内容をチェックします。これらの内容が一致した場合、デバイスはハードウェア・ジェネラル・コールを受信したことになります。これは、デバイスがマスタ・デバイスを緊急に必要としているにもかかわらず、どのマスタを参照すればよいかわからないときに使用します。これは「関係があるかもしれないデバイスに対する」呼び出しです。マスタを必要としているこのデバイスは、メッセージの中に自分のアドレスを埋め込みます。I2C 仕様（2000 年 1 月）に従って、I2C_ALT レジスタの LSB には必ず 1 を書き込んでください。
2 (R/W)	GCEN	ジェネラル・コール・イネーブル。 このビットは、I2C スレーブによるアドレス 0x00（書込み）の I2C ジェネラル・コールの ACK を有効にします。
1 (R/W)	ADR10EN	10 ビット・アドレス指定イネーブル。 このビットがクリアされると、スレーブは I2C_ID0～I2C_ID3 にプログラムされた 4 つのスレーブ・アドレスに対応可能になります。このビットがセットされると、10 ビットのアドレス指定が有効になります。1 つの 10 ビット・アドレスがスレーブに対応し、I2C_ID0 と I2C_ID1 に保存されます。ここで、I2C_ID0 にはアドレスの最初のバイトが格納され、上位 5 ビットには 11110 をプログラムしなければなりません。同時に、I2C_ID2 と I2C_ID3 には 7 ビット・アドレスをプログラムできます。
0 (R/W)	SLVEN	スレーブ・イネーブル。 このビットが 1 のとき、スレーブがイネーブルされます。0 のとき、すべてのスレーブのステート・マシンのフロップがリセット状態に保持され、スレーブはディスエーブルされます。APB 書込み可能レジスタ・ビットはリセットされないことに注意してください。

共通コントロール

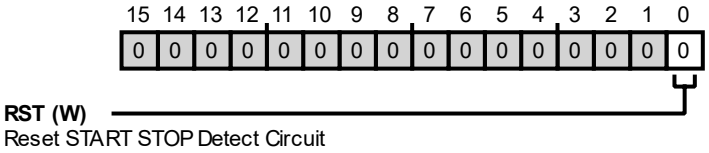


図 18-20：I2C_SHCTL のレジスタ図

表 18-19：I2C_SHCTL のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
0 (RX/W)	RST	START・STOP 検出回路リセット。 このビットに 1 が書き込まれると、SCL と SDA の同期回路、START・STOP 検出回路、および LINEBUSY 検出回路をリセットします。マスタがイネーブルされていないときでも LINEBUSY をアサートする必要があるため、マスタとスレーブの両方がディスエーブルの場合は、これらの回路はリセットされません。SCL/SDA が適切にパワーアップされなかった場合にのみ、パワーオン・リセット後にこれらの回路をリセットする必要があります。このビットを読み出すと、常に 0 がリードバックされます。

スレーブ受信

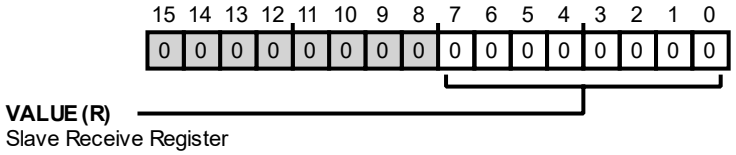


図 18-21：I2C_SRX のレジスタ図

表 18-20：I2C_SRX のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7:0 (R/NW)	VALUE	スレーブ受信レジスタ。

スレーブの I2C ステータス／エラー／IRQ

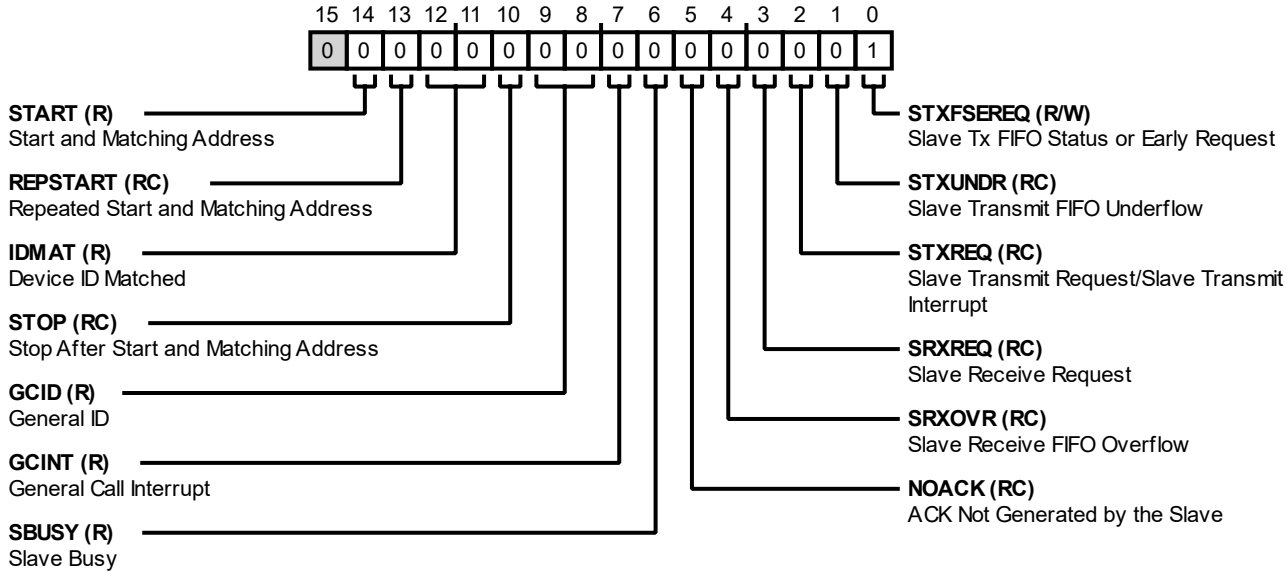


図 18-22：I2C_SSTAT のレジスタ図

表 18-21：I2C_SSTAT のレジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
14 (R/NW)	START	開始とマッチング・アドレス。 このビットは、SCL/SDA で START 条件が検出され、デバイス・アドレスが一致した場合、ジェネラル・コール（GC-0000_0000）コードを受信して GC がイネーブルになった場合、高速（HS-0000_1XXX）コードを受信した場合、あるいは開始バイト（0000_0001）を受信した場合にアサートされます。STOP または START 条件を受信するとクリアされます。	
13 (RC/NW)	REPSTART	反復開始とマッチング・アドレス。 このビットは、I2C_SSTAT.START が既にアサートされているときに反復開始を検出すると、アサートされます。読み出されるか、STOP 条件を受信するとクリアされます。このビットは割込みを生成できます。	
12:11 (R/NW)	IDMAT	一致したデバイス ID。	
		0	受信したアドレスが ID レジスタ 0 と一致
		1	受信したアドレスが ID レジスタ 1 と一致
		2	受信したアドレスが ID レジスタ 2 と一致
		3	受信したアドレスが ID レジスタ 3 と一致

表 18-21：I2C_SSTAT のレジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応
10 (RC/NW)	STOP	開始およびマッチング・アドレス後に停止。 スレーブが、前の START 条件を受信してアドレスが一致した後に STOP 条件を受信した場合、ハードウェアによってセットされます。I2C_SSTAT レジスタを読み出すとクリアされます。スレーブ・コントロール・レジスタの I2C_SCTL.IENSTOP がアサートされているときにこのビットがセットされると、スレーブ割込み要求がアサートされます。このビットは割込みを生成できます。
9:8 (R/NW)	GCID	ジェネラル ID。 I2C_SCTL.GCSBCLR に 1 が書き込まれると、I2C_SSTAT.GCID はクリアされます。これらのステータス・ビットは、ジェネラル・コール・リセットではクリアされません。 0：ジェネラル・コールなし 1：ジェネラル・コールはアドレスをリセットしてプログラムします 2：ジェネラル・コールはアドレスをプログラムします 3：ジェネラル・コールは代替 ID のマッチングを行います
7 (R/NW)	GCINT	ジェネラル・コール割込み。 このビットは常に割込みを生成します。スレーブ・デバイスがいずれかのタイプのジェネラル・コールを受信すると、このビットがアサートされます。スレーブ・コントロール・レジスタの I2C_SCTL.GCSBCLR に 1 を書き込むと、クリアされます。ジェネラル・コール・リセットだった場合、すべてのレジスタはデフォルト値になります。ハードウェア・ジェネラル・コールだった場合は、Rx FIFO はジェネラル・コールの 2 番目のバイトを保持し、I2C_ALT レジスタと比較されます。
6 (R/NW)	SBUSY	スレーブ・ビジー。 スレーブ・デバイスが I2C START 条件を受信した場合に、ハードウェアによってセットされます。アドレスが ID レジスタと一致しなかった場合、スレーブ・デバイスが I2C STOP 条件を受信した場合、あるいは反復開始アドレスが一致しなかった場合に、ハードウェアによってクリアされます。
5 (RC/NW)	NOACK	スレーブによる ACK 生成なし。 このビットがアサートされている場合は、スレーブがデバイス・アドレスに NACK で応答したことを示しています。送信データがないときにシーケンスがスレーブ読出しを行った場合、または I2C_SCTL.NACK ビットがセットされているときにデバイスがアドレス指定された場合に、このビットがアサートされます。このビットは、I2C_SSTAT レジスタを読み出すことでクリアされます。
4 (RC/NW)	SRXOVR	スレーブ受信 FIFO オーバーフロー。 既に一杯になっているスレーブ受信 FIFO にバイトが新たに書き込まれると、アサートされます。
3 (RC/NW)	SRXREQ	スレーブ受信要求。 I2C_SSTAT.SRXREQ は、スレーブ受信 FIFO が空にならない限り、常にアサートされています。スレーブ受信 FIFO の読出し、またはフラッシュが行われると、クリアされます。このビットは、バイトの最後のデータ・ビットに対応する SCL クロック・パルスの立下がりエッジでアサートされます。このビットは割込みを生成できます。

表 18-21：I2C_SSTAT のレジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応
2 (RC/NW)	STXREQ	スレーブ送信要求／スレーブ送信割込み 読出しのときはスレーブ送信要求、書込みのときはスレーブ送信割込みビットのクリアが行われます。このビットは読出し専用です。 I2C_SCTL.EARLYTXR = 0 の場合、転送の方向ビットがハイを受信すると、このビットがセットされます。 その後、FIFO が一杯にならない限り、このビットはアサートされた状態に保たれます。初期状態では（デバイス・アドレスが一致した場合にも）、このビットは、方向ビットに対応する SCL クロック・パルスの立下がりエッジでアサートされます。 I2C_SCTL.EARLYTXR = 1 の場合、転送の方向ビットがハイを受信すると、このビットがセットされます。 その後、FIFO が一杯にならない限り、このビットはアサートされた状態に保たれます。初期状態では（デバイス・アドレスが一致した場合にも）、このビットは、方向ビットに対応する SCL クロック・パルスの立上がりエッジの後にアサートされます。 このビットは、I2C_SSTAT レジスタを読み出すとクリアされます。slv_txint_clr は書込み専用です。このビットが 1 のとき、スレーブ送信割込みとクロック・ストレッチングはクリアされます。そして、STOP または（RE）START を受信（転送が完了）した場合、TX FIFO が書き込まれた（ソフトウェアが更にデータを送信することを決定した）場合、TX FIFO が空で、NACK ではなく ACK を受信した場合、のいずれかに該当するとき、このビットはクリアされます。
1 (RC/NW)	STXUNDR	スレーブ送信 FIFO アンダーフロー。 このビットは、マスタがデバイスにデータを要求したとき、SCL の立上がりエッジで Tx FIFO が空の場合にセットされます。
0 (R/W)	STXFSEREQ	スレーブ Tx FIFO ステータス、または早期要求。 I2C_SCTL.EARLYTXR = 0 の場合、スレーブ Tx FIFO が空であれば常にこのビットがアサートされています。 I2C_SCTL.EARLYTXR = 1 の場合、転送の方向ビットがハイを受信すると、このビットがセットされます。 方向ビットに対応する SCL クロックの立上がりエッジ（デバイス・アドレスが一致した場合にも）でアサートされます。このビットは、1 回の転送で 1 回だけアサートされます。I2C_SCTL.EARLYTXR がアサートされているかどうかを読み出すと、このビットはクリアされます。

マスタおよびスレーブ FIFO ステータス

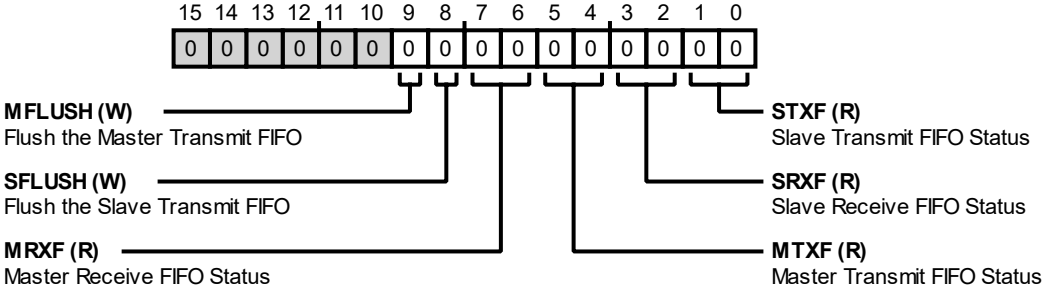


図 18-23：I2C_STAT のレジスタ図

表 18-22：I2C_STAT のレジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
9 (RX/W)	MFLUSH	マスタ送信 FIFO フラッシュ。 このビットに 1 を書き込むと、マスタ送信 FIFO をフラッシュします。アービトレーションが失われた場合やスレーブが NACK で応答した場合には、マスタ送信 FIFO をフラッシュする必要があります。	
8 (RX/W)	SFLUSH	スレーブ送信 FIFO フラッシュ。 このビットに 1 を書き込むと、スレーブ送信 FIFO をフラッシュします。	
7:6 (R/NW)	MRXF	マスタ受信 FIFO ステータス。 このステータスは、FIFO 内のバイト数を示します。	
		0	FIFO は空
		1	FIFO 内に 1 バイト
		2	FIFO 内に 2 バイト
		3	予備
5:4 (R/NW)	MTXF	マスタ送信 FIFO ステータス。 このステータスは、FIFO 内のバイト数を示します。	
		0	FIFO は空
		1	FIFO 内に 1 バイト
		2	FIFO 内に 2 バイト
		3	予備

表 18-22：I2C_STAT のレジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
3:2 (R/NW)	SRXF	スレーブ受信 FIFO ステータス。 このステータスは、FIFO 内のバイト数を示します。	
		0	FIFO は空
		1	FIFO 内に 1 バイト
		2	FIFO 内に 2 バイト
		3	予備
1:0 (R/NW)	STXF	スレーブ送信 FIFO ステータス。 このステータスは、FIFO 内のバイト数を示します。	
		0	FIFO は空
		1	FIFO 内に 1 バイト
		2	FIFO 内に 2 バイト
		3	予備

スレーブ送信

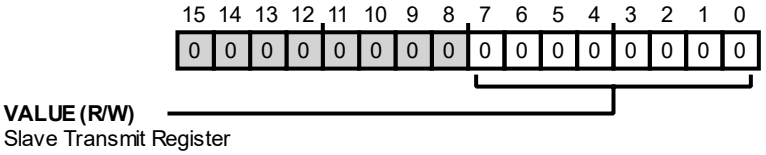


図 18-24：I2C_STX のレジスタ図

表 18-23：I2C_STX のレジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
7:0 (R/W)	VALUE	スレーブ送信レジスタ。

タイミング・コントロール・レジスタ

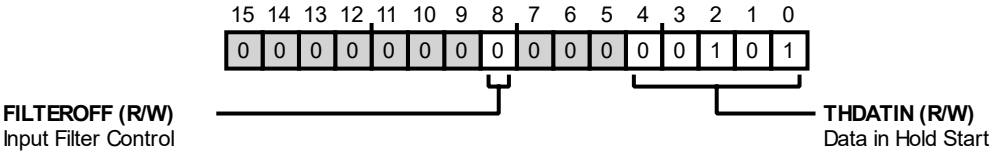


図 18-25：I2C_TCTL のレジスタ図

表 18-24：I2C_TCTL のレジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
8 (R/W)	FILTEROFF	入力フィルタ制御。 入力フィルタを制御します。PCLK レートが 16MHz より低いときは、デジタル・フィルタをディスエーブルしたほうが望ましい場合があります。	
		0	デジタル・フィルタをイネーブルして 1 PCLK に等しくする。
		1	デジタル・フィルタをディスエーブルする。パッドのフィルタ処理は影響を受けません。
4:0 (R/W)	THDATIN	ホールド・スタートのデータ。 I2C_TCTL.THDATIN によって、START または STOP 条件を認識する前に満足する必要があるホールド時間条件を決定します。有効な開始条件として認識されるには、SDA と SCL の立下がり間のホールド時間がプログラムされた値を超えなければなりません。 I2C_TCTL.THDATIN = （tHD;STA） / （クロック周期）。例えば、16MHz の PCLK（62.5ns）で最小の tHD;STA 限界を 300ns とすると、（300ns） / （62.5ns）= 5 になります。	

19 ビーパ・ドライバ（BEEP）

ビーパ・ドライバ・モジュールは、プログラマブルな周波数で差動方形波を生成し、2 つの端子が差動方形波出力に接続された外部圧電音声コンポーネントを駆動します。このモジュールは、標準 APB インターフェースでシステム・バスに接続します。

ビーパの機能

MCU に組み込まれたビーパ・ドライバは、以下の機能を備えています。

- 8kHz～約 0.25kHz の周波数を生成するモジュール。
- システム・クロックの変化に影響を受けない独立した 32kHz の固定クロック源で動作。
- 4ms～1.02s のトーン持続時間を 4ms 刻みで設定可能。
- シングルトーン・モード（パルス・モード）とマルチトーン・モード（シーケンス・モード）による汎用性の高い再生オプション。
- シーケンス・モードでは、1～254（2～508 トーン）の任意の数のトーン・ペアを再生するか、ユーザが停止するまで再生し続けるようにプログラム可能。
- ビープの開始／終了、シーケンスの終了、またはシーケンスがまもなく完了することを示す割込みが使用可能。

ビーパ機能の説明

ここでは、ADuCM4050 MCU が使用するビーバ・ドライバの機能について説明します。

ビーパのブロック図

ADuCM4050 MCU が使用するビーパ・ドライバのブロック図を以下に示します。

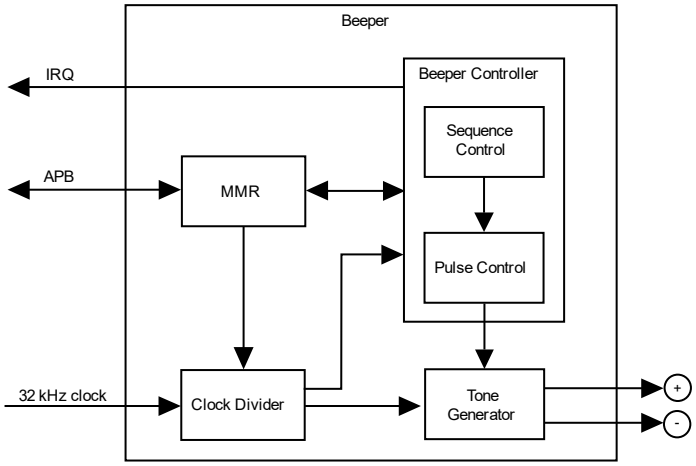


図 19-1：ビーパ・ドライバ

クロック・ドライバは、外部の圧電コンポーネントを駆動するために、32kHz の入力クロックを 8kHz～約 0.25kHz の範囲に分周します。ビーパ制御ブロックはトーン・ジェネレータを制御してその動作をイネーブル／ディスエーブルし、クロック分周器から適切な周波数出力を選択します。

また、ビーパは、トーンの持続時間と反復シーケンスを追跡します。ビーパ・モジュールは、1 つの割込みラインでマイクロコントローラに割込みをかけることができます。このモジュールがディスエーブルされている（圧電コンポーネントに電圧がかからない）ときは外部出力ラインがローに保持され、トーン再生時には差動方形波を駆動します。

ビーパ動作モード

ビーパの基本的な動作は、1 つまたは両方のトーン・レジスタへトーン・データを書き込み、その後に BEEP_CFG.EN ビットをセットすることによってビーパをイネーブルするという 2 つの動作で構成されます。モジュールをイネーブルすると、動作モードに応じて 1 つまたは複数のトーンが再生されます。再生中、モジュールは（VSS から VCC へ）差動方形波を出力しますが、その周波数と持続時間は、BEEP_TONEA または BEEP_TONEB レジスタで設定します。

ビーパには 2 つの動作モードがあります。ビーパがイネーブルされたときに BEEP_CFG.SEQREPEAT ビット・フィールドの値がゼロでない場合、ビーパはシーケンス・モードで動作します。BEEP_CFG.SEQREPEAT が 0x0 に設定された状態でビーパがイネーブルされると、ビーパはパルス・モードで動作します。

パルス・モード動作は、トーンを 1 つだけ再生して動作を完了します。シーケンス・モードの動作完了までのトーン再生数は、ユーザが設定できます。パルス・モードまたはシーケンス・モードでの動作時に、BEEP_STAT.BUSY ビットがアサートされます。このビットは動作完了時にクリアされます。

現在再生中のトーンは、BEEP_TONEA.DUR ビットと BEEP_TONEB.DUR ビットに 0x0 を書き込むことによって、途中で終了させることができます。終了時には BEEP_STAT.AENDED ビットと BEEP_STAT.BENDED ビットがセットされ、BEEP_CFG.AENDIRQ ビットと BEEP_CFG.BENDIRQ ビットを使用して割込みが設定されている場合は、割込みが生成されます。あるいは、ビーパをディスエーブル（BEEP_CFG.EN ビットをクリア）することによって、すべての再生を直ちに終了させることができます。ビーパをディスエーブルするとイベントが発生しないので、割込みが生成されることもありません。

割込みは、イベントが要求されたときに生成されます。BEEP_STAT レジスタは、レジスタの最後のクリア後に発生した 6 つの追跡イベントに関するフィードバックを提供します。BEEP_CFG レジスタでこれら 6 個のイベントのいずれかを選択すれば、そのイベントを使って割込みを生成することができます。

ビーパの動作モードはビーパがイネーブルされたときに設定され、ビーパを再びディスエーブルするまで変更できません。ビーパは、BEEP_CFG.EN ビットに 0x0 を書き込むか、現在のパルスまたはシーケンスが完了してビーパがディスエーブルされるまで待つことによって、ディスエーブルできます。

パルス・モード

パルス・モードでは、ビーパがトーンを 1 つだけ再生します。パルス・モードの動作は、BEEP_CFG.SEQREPEAT ビットをゼロに設定してビーパをイネーブルすることにより開始します。イネーブルされると、BEEP_TONEA レジスタの記述に従ってビーバがシングルトーンを再生します。このトーンの再生が完了すると、BEEP_STAT.BUSY ビットがクリアされます。

割込みは BEEP_CFG レジスタの要求に従って生成されます。このモードで発生する（かつ有効な割込みを生成する）イベントは、BEEP_TONEA 再生の開始（BEEP_CFG.ASTARTIRQ）と終了（BEEP_CFG.AENDIRQ）だけです。

シーケンス・モード

シーケンス・モードでは、設定された数のトーンをビーパが再生します。シーケンス・モードの動作は、BEEP_CFG.SEQREPEAT フィールドをゼロ以外の値に設定してビーパをイネーブルすることにより開始します。イネーブルされると、ビーバが BEEP_TONEA レジスタと BEEP_TONEB レジスタの記述に従って、一連のツートーン・シーケンスを再生します。それぞれのツートーンの反復は BEEP_TONEA を再生することによって開始され、BEEP_TONEB を再生することによって終了します。

このシーケンスは、BEEP_CFG.SEQREPEAT フィールドの設定に従って繰り返されます。0xFF は特別なケースで、ユーザ・コードにより停止されるまでシーケンサを実行するために使われます。すべての繰り返しが完了すると、BEEP_STAT.BUSY ビットがクリアされます。

シーケンス繰返しの残り回数は、BEEP_STAT.SEQREMAIN ビットから読み出すことができます。無限のシーケンスを実行する場合、BEEP_STAT.SEQREMAIN ビットは常に 0xFF を返します。ビーパをシーケンス・モードで実行中に BEEP_CFG.SEQREPEAT へ書込みを行うと、繰返しカウンタが再起動されてその値までのカウントが開始され、BEEP_STAT.SEQREMAIN ビットの値が直ちに更新されます。

シーケンス・モードは、BEEP_CFG.SEQREPEAT ビットに 0x0 を書き込むことによって中止できます。BEEP_CFG.SEQREPEAT ビットを 0x0 に設定すると、ビーパは再生を停止し、現在再生中のツートーン・シーケンスが完了してからビーバがディスエーブルされます。終了時には BEEP_STAT.SEQENDED ビットがセットされ、BEEP_CFG.SEQATENDIRQ ビットを使用して割込みが設定されている場合は、割込みが生成されます。あるいは、ビーパをディスエーブル（BEEP_CFG.EN ビットをクリア）すれば、すべての再生が直ちに中止されます。ビーパをディスエーブルするとイベントが発生しなくなるので、割込みも生成されません。

割込みは、BEEP_CFG レジスタの要求に従ってシーケンス・モードで生成されます。このモードでは、すべてのビーパ・イベントが割込みを生成できます。

トーン

ビーパによって再生されるトーンの周波数と持続時間は、BEEP_TONEA レジスタと BEEP_TONEB レジスタに保存された値によって決まります。これらのレジスタは、それぞれ独立した 1 つのトーンを記述します。パルス・モードはレジスタ BEEP_TONEA のトーンだけを再生し、シーケンス・モードは両方のトーン・レジスタのトーンを再生します。

これらのトーン・レジスタは、3 つのフィールド、すなわち持続時間フィールド（BEEP_TONEA.DUR、BEEP_TONEB.DUR）、 周波数フィールド（BEEP_TONEA.FREQ、BEEP_TONEB.FREQ）、およびディスエーブル・フィールド（BEEP_TONEA.DIS、BEEP_TONEB.DIS）に分割されています。

トーン・レジスタへの書込みはいつでも可能です。現在再生中のトーンに関する持続時間フィールド（BEEP_TONEA.DUR、BEEP_TONEB.DUR）への 0x0 の書込み、あるいはディスエーブル・ビット（BEEP_TONEA.DIS、BEEP_TONEB.DIS）のセット／クリアは、直ちに有効になります。BEEP_TONEA および BEEP_TONEB レジスタに対する他のすべての変更は、次回のトーン再生時に有効になります。したがって、ほとんどの場合は、現在のトーンを再生中に次のトーン・データを書き込むことができます。

持続時間フィールドは、再生選択時のトーンの再生時間を設定するために使用します。持続時間は 4ms 単位で測定されます。トーン・レジスタには、0 から 255 までの任意の値を保存できます。持続時間を 0x0 にすると、直ちにトーン再生が終了します。255（0xFF）は特殊なケースの値で、トーン持続時間を無限に設定するために使われます。この場合、トーンの再生が開始されると、ユーザ・コードがそのトーンを停止させる（BEEP_TONEA.DUR、BEEP_TONEB.DUR に 0x0 を書き込む）か、ビーパがディスエーブルされる（BEEP_CFG.EN ビットをクリアする）まで再生され続けます。

周波数フィールドは、ソース・クロック（32.768kHz）を基準とするトーンの相対周波数を設定するために使用します。周波数フィールドに値 0、1、2、または 3 を書き込んだ場合は、いずれも同じ結果が得られます。つまり、再生中の出力振動数が 0 になります（「レスト・トーン」とも呼ばれます）。ソース・クロックを希望のトーン周波数に分周するには、4～127（0x7F）の任意の値にします。これにより、8kHz から約 0.25kHz までの再生範囲が得られます。

ディスエーブル・フィールドは、再生中にビーパの出力ピンを制御するために使用します。BEEP_TONEA.DIS ビットと BEEP_TONEB.DIS ビットをセットしてトーンを再生すると、出力ピンはビーパがディスエーブルされているかのように動作します。つまり、どちらのピンもロジック 0 となってピン間に DC 電位は発生せず、振動もしません。この機能は、設定されたシーケンスに長いサイレンス時間が存在する場合に使用することを想定しています。この機能を使用することで、これらのサイレンス時間に圧電コンポーネントが損傷するのを防ぐことができます。

クロッキングと消費電力

ビーパ・ドライバの周波数と持続時間は、32.768kHz の入力クロックに基づいています。クロック・ソースはシステム全体で設定され、該当するクロック制御モジュール・レジスタ（CLKG_OSC_CTL.LFCLKMUX）に書込みを行うことによって、外部水晶発振器または内部発振器を選択します。選択されたクロックは、タイマー用の安定したリファレンスを提供するだけです。ビーパの内部ロジックは、システムのペリフェラル・クロック（PCLK）によってクロックされます。このため、システムが低消費電力モードになっている間はペリフェラル・クロック（PCLK）が供給されなくなるので、ビーパを作動させることができません。

タイマー開始イベントは 32kHz クロックに同期されます。したがって、ビーパをイネーブルするときは、32kHz クロックの 1 周期分（30,520ns）だけ出力が遅延します。つまり、4MHz の PCLK の場合は、ビーパの起動から実際にオ

ーディオ出力が生成されるまでに最大 122 PCLK 周期分の遅延が生じることになります。この間に BEEP_TONEA レジスタと BEEP_TONEB レジスタに変更を加えると、保留中のオーディオに影響が生じます。ユーザ・コードは、BEEP_STAT.ASTARTED ビットと BEEP_STAT.BSTARTED ビットにポーリングを行うかこれらのビットに割込みを設定することによって、トーン・レジスタに変更を加える前に、トーンが再生されているかどうかを確認する必要があります。

低消費電力状態になるとペリフェラル・クロックが供給されなくなるので、ユーザ・コードはその前にビーパをディスエーブルする必要があります。ビーパがトーンを再生しているときにシステムが低消費電力モードになると、圧電コンポーネントの端子に一定の DC 電圧が長時間にわたって加わるので、圧電コンポーネントを損傷させる可能性があります。

パワーダウンに関する考慮事項

以下の状態でビーパをイネーブルしてトーン出力を駆動した場合は、外部圧電ビーパ装置に修復不能な損傷を与える可能性があります。

- 休止モード
- シャットダウン・モード
- PCLK オフ

これらのモードでは、ピンとビーパ・モジュールの接続が自動的に解除されません

ビーパ割込みとイベント

ビーパ動作中に発生し得る追跡イベントは 6 種類あります。トーン A の開始または終了、トーン B の開始または終了、シーケンスの終了または終了 1 ステップ前です。これら 6 つのイベントは常にモニタされており、発生すると、後で読み出せるように BEEP_STAT レジスタにスティッキー・ビットがセットされます。これらのビットは、ユーザ・コードによってクリアされるまでそのまま残ります。

これら 6 つのイベントは割込みの生成にも使われます。どのイベントで割込みを生成するかは、BEEP_CFG レジスタ内の該当ビットをセットすることにより決定します。イベントが割込みをトリガしたときは、ユーザ・コードでそのイベント・ビットをクリアするか、割込み処理時に割込み選択ビットをディスエーブルする必要があります。

ビーパ割込みを処理するときは、BEEP_STAT レジスタ内のすべてのイベント・ビットをユーザ・コードでチェックして、最終的にそれらをクリアする必要があります。割込みは、複数のイベントによってトリガすることもできます（割込みがイネーブルされている場合）。BEEP_STAT に 0xFF を書き込むと、すべてのイベントがクリアされます。

すべての追跡イベントは、割込み生成のために個別に選択できます。

ビーパ・プログラミング・モデル

以下では、一般的なプログラミングのガイドラインと手順を示します。

タイミング図

トーンを生成するために設定されたビーパの動作を図に示します。

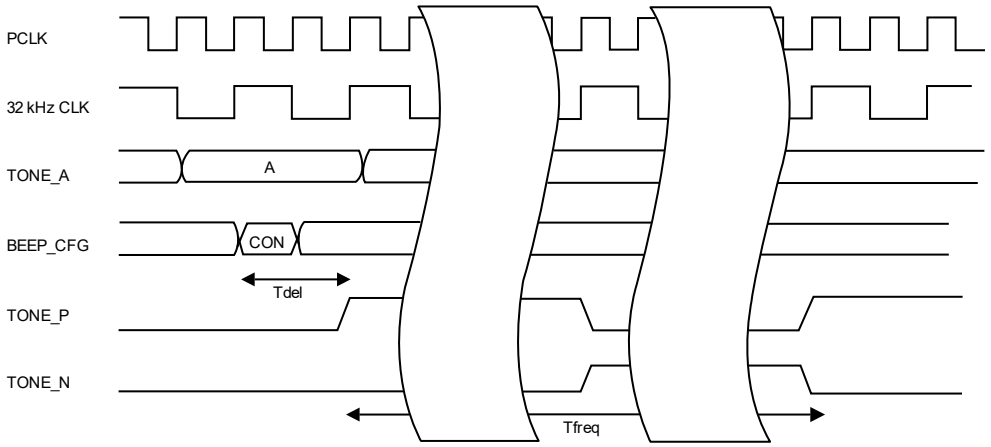


図 19-2：タイミング図

この例では、まず BEEP_TONEA レジスタにトーンが書き込まれて、次に BEEP_CFG レジスタへの書込みによってビーパがイネーブルされます。Tdel は、ビーパをイネーブルしてから実際に再生が開始されるまでに、最大で 32kHz クロック 1 周期分の遅延があることを示しています。Tfreq は、2 つの差動ピンの出力時間を示しています。1kHz（1024Hz）トーンの場合、Tfreq = 0.9766ms です。

プログラミング・ガイドライン

プログラミング例

ビーパ・ドライバは長さ 1 秒の 1kHz シングル・ビープに対して設定され、トーンの再生が終了すると割込みが生成されます。

1. トーン A の持続時間を 1 秒に設定します。BEEP_TONEA.DUR = 0xFA //（250 × 4ms）
2. トーン A の周波数を 1kHz に設定します。BEEP_TONEA.FREQ = 0x20 //（32kHz/32）
3. トーン A の終了時に入力割込みを設定します。BEEP_CFG.AENDIRQ = 0x1
4. シーケンスの繰返しをゼロに設定します。BEEP_CFG.SEQREPEAT = 0x0
5. ビーパをイネーブルします。BEEP_CFG.EN = 0x1

レジスタ・アクセス・シーケンスは BEEP_TONEA = 0x20FA および BEEP_CFG = 0x0900 です。

ビーパ・ドライバは、音階 G#が 500ms、F#が 1000ms のツートーン・シーケンス 32 個からなるメロディに設定され、メロディの終了時に割込みが生成されます。

1. トーン A の持続時間を 500ms に設定します。BEEP_TONEA.DUR = 0x7D //（125 × 4ms）
2. トーン A の周波数を G#に設定します。BEEP_TONEA.FREQ = 0x27 //（32kHz/39 = 840Hz）
3. トーン B の持続時間を 1000ms に設定します。BEEP_TONEB.DUR = 0xFA //（250 × 4ms）
4. トーン B の周波数を F#に設定します。BEEP_TONEB.FREQ = 0x2C //（32kHz/44 = 745Hz）

5. シーケンス終了時の入力割込みを設定します。BEEP_CFG.SEQATENDIRQ = 0x1
6. シーケンスの繰返しを 32 に設定します。BEEP_CFG.SEQREPEAT = 0x20
7. ビーパをイネーブルします。BEEP_CFG.EN = 0x1

レジスタ・アクセス・シーケンスは BEEP_TONEA = 0x277D、BEEP_TONEB = 0x2CFA、BEEP_CFG = 0x8120 です。

ADuCM4050 BEEP レジスタの説明

ビーパ・ドライバ（BEEP）には以下のレジスタが含まれています。

表 19-1：ADuCM4050 BEEP レジスター一覧

レジスタ名	説明
BEEP_CFG	Beeper Configuration
BEEP_STAT	Beeper Status
BEEP_TONEA	Tone A Data
BEEP_TONEB	Tone B Data

ビーパ設定

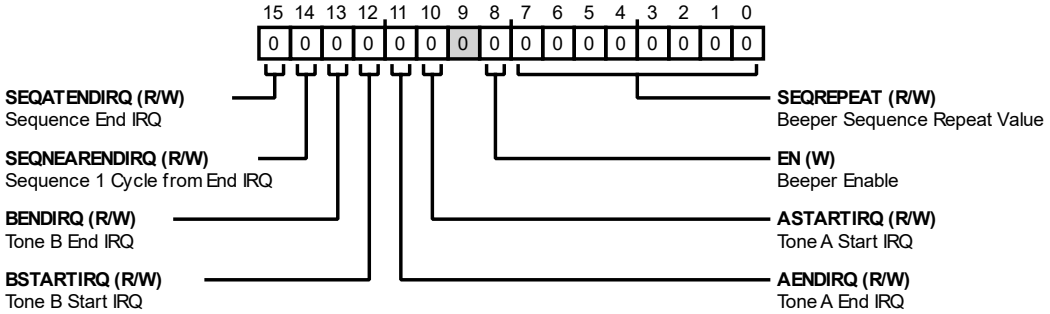


図 19-3：BEEP_CFG レジスタ図

表 19-2：BEEP_CFG レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15 (R/W)	SEQATENDIRQ	シーケンス終了 IRQ。 シーケンサが動作を停止したときに割込みを要求するには、このビットをセットします。	
		0	シーケンス終了時に IRQ を生成しません。
		1	シーケンス終了時に IRQ を生成します。
14 (R/W)	SEQNEARENDIRQ	終了から 1 サイクル前のシーケンスで IRQ。 シーケンス実行中、完了まであと 1 回の繰り返しが残された状態で割込みを要求するには、このビットをセットします。	
		0	シーケンス終了の 1 トーン・ペア前に IRQ を生成しません。
		1	シーケンス終了の 1 トーン・ペア前に IRQ を生成します。
13 (R/W)	BENDIRQ	トーン B 終了 IRQ。 BEEP_TONEB が動作を停止したときに割込みを要求するには、このビットをセットします。	
		0	トーン B 終了時に IRQ を生成しません。
		1	トーン B 終了時に IRQ を生成します。
12 (R/W)	BSTARTIRQ	トーン B 開始 IRQ。 BEEP_TONEB が動作を開始したときに割込みを要求するには、このビットをセットします。	
		0	トーン B 開始時に IRQ を生成しません。
		1	トーン B 開始時に IRQ を生成します。

表 19-2：BEEP_CFG レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
11 (R/W)	AENDIRQ	トーン A 終了 IRQ。 トーン A が動作を停止したときに割込みを要求するには、このビットをセットします。	
		0	トーン A 終了時に IRQ を生成しません。
		1	トーン A 終了時に IRQ を生成します。
10 (R/W)	ASTARTIRQ	トーン A 開始 IRQ。 トーン A が動作を開始したときに割込みを要求するには、このビットをセットします。	
		0	トーン A 開始時に IRQ を生成しません。
		1	トーン A 開始時に IRQ を生成します。
8 (RX/W)	EN	ビーパ・イネーブル。 トーン A の再生を開始するには、このビットに 0x1 を書き込みます。BEEP_CFG.SEQREPEAT = 0 の場合はシングルトーンが再生され、それ以外の場合は一連のツートーン・シーケンスが再生されます。このビットに 0x0 を書き込めば、いつでもオーディオの再生を終了できます。このビットを読み出すと、常に 0x0 が返されます。ビーパが現在イネーブルされているかどうかを知るには、BEEP_STAT.BUSY を読み出します。	
		0	モジュールをディスエーブルします。
		1	モジュールをイネーブルします。
7:0 (R/W)	SEQREPEAT	ビーパ・シーケンスの繰り返し回数。 ビーパがイネーブル状態に移行する際、このフィールドの値が、ビーパをパルス・モードで作動させるかシーケンス・モードで作動させるかを選択します。0x0 の場合ビーパはパルス・モードで動作し、ディスエーブルされるまで 1 つのトーン（トーン A）だけを再生します。このフィールドにゼロ以外の値を書き込むと、ビーパはシーケンス・モードで動作し、ディスエーブルされるまでツートーン・シーケンス（トーン A とトーン B）を要求された回数だけ再生します。 パルス・モードで動作中にこのフィールドを更新しても、結果に影響はありません。シーケンス・モードで動作中にこのフィールドを更新すると、更新内容が直ちに有効になります。	
		0	イネーブルするとパルス・モードで動作を開始します。
		1-254	イネーブルするとシーケンス・モードで動作を開始し、設定された回数だけ音階を再生します。
		255	イネーブルするとシーケンス・モードで動作を開始し、ユーザ・コードにより停止されるまで再生を続けます。

ビーパ・ステータス

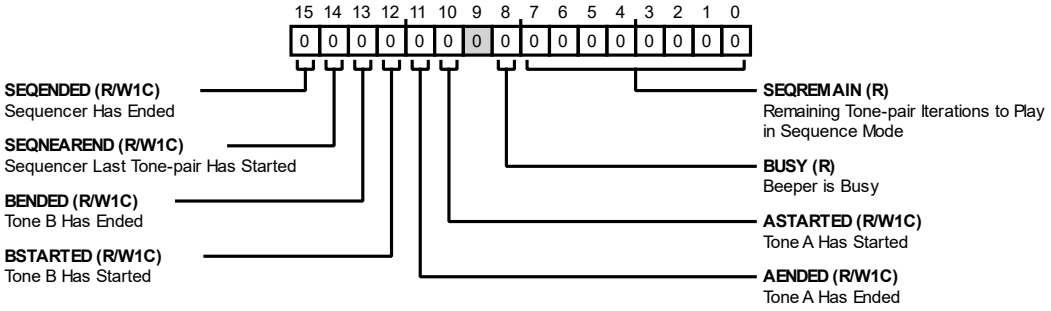


図 19-4：BEEP_STAT レジスタ図

表 19-3：BEEP_STAT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15 (R/W1C)	SEQENDED	シーケンサの動作が終了。 シーケンス・モードでビーパが動作を停止すると、常にこのビットがアサートされます。このビットは、ユーザ・コードでこの位置に 0x1 を書き込んだ場合のみクリアされます。
14 (R/W1C)	SEQNEAREND	シーケンサ最終トーン・ペアの開始。 ビーパがシーケンス・モードで動作中に残りの再生シーケンスが 1 回になると、常にこのビットがアサートされます。このビットは、ユーザ・コードでこの位置に 0x1 を書き込んだ場合のみクリアされます。
13 (R/W1C)	BENDED	トーン B が終了。 ビーパがトーン B の再生を停止すると、常にこのビットがアサートされます。このビットは、ユーザ・コードでこの位置に 0x1 を書き込んだ場合のみクリアされます。
12 (R/W1C)	BSTARTED	トーン B の開始。 ビーパがトーン B の再生を開始すると、常にこのビットがアサートされます。このビットは、ユーザ・コードでこの位置に 0x1 を書き込んだ場合のみクリアされます。
11 (R/W1C)	AENDED	トーン A が終了。 ビーパがトーン A の再生を停止すると、常にこのビットがアサートされます。このビットは、ユーザ・コードでこの位置に 0x1 を書き込んだ場合のみクリアされます。
10 (R/W1C)	ASTARTED	トーン A の開始。 ビーパがトーン A の再生を開始すると、常にこのビットがアサートされます。このビットは、ユーザ・コードでこの位置に 0x1 を書き込んだ場合のみクリアされます。

表 19-3：BEEP_STAT レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
8 (R/NW)	BUSY	ビーパがビジー。 BEEP_CFG.EN に 0x1 を書き込むことによってビーパ・モジュールがイネーブルされると、このビットがアサートされます。このビットは、モジュールがその動作を完了するか、ユーザ・コードが BEEP_CFG.EN に 0x0 を書き込むとクリアされます。	
		0	ビーバは現在ディスエーブルされています。
		1	ビーバは現在イネーブルされています。
7:0 (R/NW)	SEQREMAIN	シーケンス・モードでの残りのトーン・ペア繰返し再生回数。 シーケンス終了後、このフィールドは BEEP_CFG.SEQREPEAT にリセットされます。このフィールドは、ビーバがシーケンス・モードでオーディオを再生するのに合わせて更新されます。それぞれのツートーンの繰り返しはトーン A の再生によって開始され、トーン B の再生によって終了します。このフィールドはトーン B 再生の終了時に更新されるので、最後の繰返しにおいて、トーン A とトーン B の再生時に 0x1 が返されます。無限のシーケンスを実行する場合、このフィールドは常に 0xFF を返します。	

トーン A のデータ

トーン A は、シーケンス・モードで再生される最初のトーンであり、パルス・モードで再生される唯一のトーンです。トーン A の再生中に BEEP_TONEA.DUR フィールドへ 0x0 を書き込むと、再生中のトーンが直ちに停止します。その他の BEEP_TONEA への書込みはすべて次のトーン再生にのみ影響し、現在再生中のトーンには影響しません。

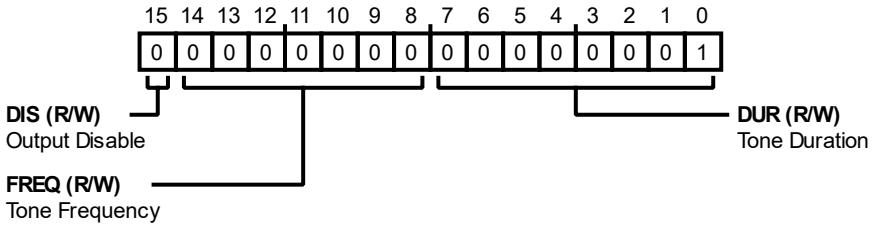


図 19-5：BEEP_TONEA レジスタ図

表 19-4：BEEP_TONEA レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15 (R/W)	DIS	出力ディスエーブル。 このビットは、トーン A の再生中にビーパ出力をディスエーブルするためにセットします。ディスエーブルすると、ビーバはその 2 つの出力ピンを両方ともロジック 0 に保持します。トーン A の再生中にこのビットへ書込みを行うと、その内容が直ちに有効になります。	
		0	出力イネーブル。
		1	出力ディスエーブル。出力ピンに DC 電位は生じません。
14:8 (R/W)	FREQ	トーン周波数。 このフィールドは、32.768kHz クロックの整数分周値としてトーン A の周波数を定義します。値 0～3 は「レスト・トーン」と解釈され、ビーパ出力ピンに振動は生じません。その他すべての値は、そのまま 32kHz 入力クロックの分周に使われます。トーン A の再生中にこのビットへ書込みを行うと、その内容が直ちに再生に影響します。	
		0-3	レスト・トーン（持続時間のみで振動なし）
		4-127	32kHz/（FREQ）の振動
7:0 (R/W)	DUR	トーン持続時間。 このフィールドは、トーン A の持続時間を 4ms 刻みで定義します。値 0x0 を書き込むと、トーン A を再生中の場合は直ちにトーンが終了します。0xFF を書き込むと、トーン A が永久に再生されます。あるいは、ユーザ・コードによりトーンが停止されるまで再生されます。現在再生中のトーンに影響する書込み値は 0x0 だけで、その他すべての値は次回トーン A が再生されるときだけ使われます。	
		0-254	トーンの再生時間は（DUR）*4ms です。
		255	トーンは無限に再生され続けます。

トーン B のデータ

トーン B はシーケンス・モードで再生される 2 番目のトーンであり、パルス・モードでは再生されません。トーン B の再生中に BEEP_TONEB.DUR フィールドへ 0x0 を書き込むと、再生中のトーンが直ちに停止します。BEEP_TONEB へのその他の書込みはすべて次のトーン再生にのみ影響し、現在再生中のトーンには影響しません。

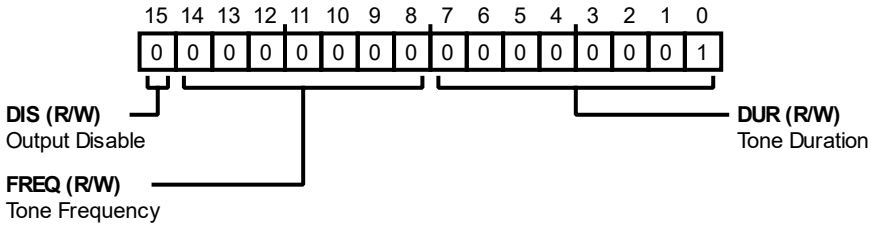


図 19-6：BEEP_TONEB レジスタ図

表 19-5：BEEP_TONEB レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15 (R/W)	DIS	出力ディスエーブル。 このビットは、トーン B の再生中にビーパ出力をディスエーブルするためにセットします。ディスエーブルすると、ビーバはその 2 つの出力ピンを両方ともロジック 0 に保持します。トーン B の再生中にこのビットへ書込みを行うと、その内容が直ちに有効になります。	
		0-3	出力イネーブル。
		4-127	出力ディスエーブル。出力ピンに DC 電位は生じません。
14:8 (R/W)	FREQ	トーン周波数。 このフィールドは、32.768kHz クロックの整数分周値としてトーン B の周波数を定義します。値 0～3 は「レスト・トーン」と解釈され、ビーパ出力ピンに振動は生じません。その他すべての値は、そのまま 32kHz 入力クロックの分周に使われます。トーン B の再生中にこのビットへ書込みを行うと、その内容が直ちに再生に影響します。	
		0-3	レスト・トーン（持続時間のみで振動なし）
		4-127	32kHz/（FREQ）の振動
7:0 (R/W)	DUR	トーン持続時間。 このフィールドは、トーン B の持続時間を 4ms 刻みで定義します。値 0x0 を書き込むと、トーン B を再生中の場合は直ちにトーンが終了します。0xFF を書き込むとトーン B が永久に再生されます。あるいは、ユーザ・コードによりトーンが停止されるまで再生されます。現在再生中のトーンに影響する書込み値は 0x0 だけで、その他すべての値は次回トーン B が再生されるときだけ使われます。	
		0-254	トーンの再生時間は（DUR）*4ms です。
		255	トーンは無限に再生され続けます。

20 ADC サブシステム

ADuCM4050 MCU は、最大 1.8MSPS で動作可能な高速、マルチチャンネルの 12 ビット A/D コンバータ（ADC）を内蔵しています。この ADC コントローラは、一連の変換を実行して専用の DMA チャンネル経由でシステムにデータを転送するように設定できます。これにより、MCU を（デバイスの全体的な消費電力を最低限に抑える）Flexi モードにしたり、他のタスクを実行したりすることができます。

ADC 機能

この ADC は以下に示す機能を備えています。

- 12 ビット分解能。
- 最大 1.8MSPS のプログラマブルな ADC 更新レート。
- 最大 8 チャンネルをサポートする入力マルチプレクサを内蔵。
- 温度検出のサポート。
- バッテリ監視のサポート。
- ソフトウェアで選択可能なオンチップ・リファレンス電圧生成：1.25 V、2.5 V、および VBAT。
- ソフトウェアで選択可能な内部／外部リファレンス。
- 自動サイクル・モード：変換用の入力チャンネルのシーケンスを自動的に選択。
- 平均化機能：1 つまたは複数のチャンネルの変換データを、最大 256 個のサンプルに対し平均化可能。
- アラート機能：ADC0_VIN0、ADC0_VIN1、ADC0_VIN2、ADC0_VIN3 チャンネルの内部デジタル・コンパレータ。ADC の結果がユーザ定義の閾値を超過したことをデジタル・コンパレータが検出した場合に、割込みを生成可能。
- 専用 DMA チャンネルのサポート。
- 温度センサーとバッテリ監視を含む各チャンネルに変換結果専用のデータ・レジスタを搭載。

ADC 機能の説明

ここでは、ADuCM4050 MCU で使用される ADC サブシステムの機能について説明します。

ADC サブシステムのブロック図

次の図に ADC サブシステムを示します。

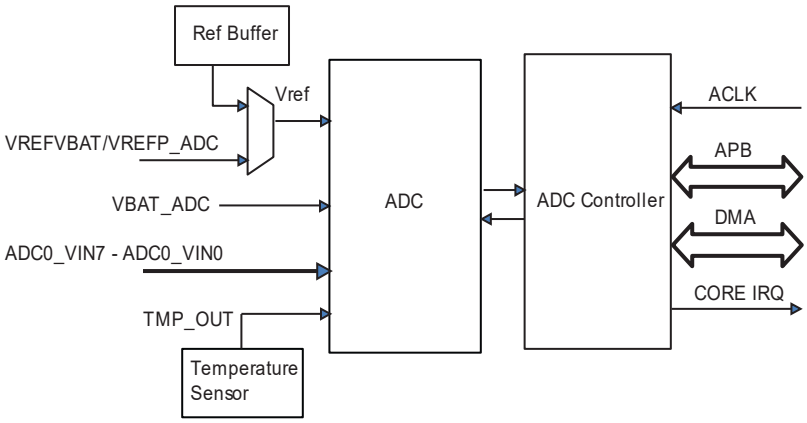


図 20-1：ADC サブシステム

ADC サブシステムのコンポーネント

ADC サブシステムは、以下のコンポーネントで構成されています。

- ADC コア
- ADC デジタル・コントローラ
- リファレンス・バッファ
- 温度センサー

ADC 電圧リファレンスの選択

ADC は、動作用に内部電圧リファレンス、バッテリ電圧、または外部電圧リファレンスを使用できます。

内部ソース

ADuCM4050 MCU には、内蔵バンドギャップ・リファレンスを使用して 1.25V または 2.5V をリファレンス電圧として生成できるリファレンス・バッファが内蔵されています。この内部リファレンス・バッファは、`ADC_CFG.REFBUFEN` ビットをセットするとイネーブルされます。内部リファレンス・バッファを使用する場合は、0.1µF のコンデンサと並列に 4.7µF のコンデンサを VREF_ADC ピンのできるだけ近くに配置することを推奨します。

ADC_CFG.VREFVBAT ビットと ADC_CFG.VREFVBAT_DEL ビットをセットすると、バッテリ電圧（VBAT）を内部リファレンスとして選択することもできます。

内部リファレンスとして 2.5V を選択するための最小 VBAT 電圧は 2.75V です。

低消費電力モード

100KSPS 未満の変換レートの場合、リファレンス・バッファは通常の動作に比べて消費電流が少ない機能を提供します。この動作モードは、ADC_CFG1.RBUFLP ビットをセットして有効にできます。

シンク・イネーブル

反転構成が必要な外部バイアス電圧を生成する場合、**外部バイアス付き ADC サブシステム・リファレンス・バッファ**の図に示すように、リファレンス・バッファは電流をシンクできます。1.25V リファレンスで 50µA のシンク電流能力、2.5V リファレンスで 100µA のシンク電流機能が ADC サブシステムで提供されています。これは、ADC_CFG.SINKEN ビットをセットすると有効にすることができます。

次の図に、外部バイアス電圧からの電流シンク構成とした ADC リファレンス・バッファの例を示します。

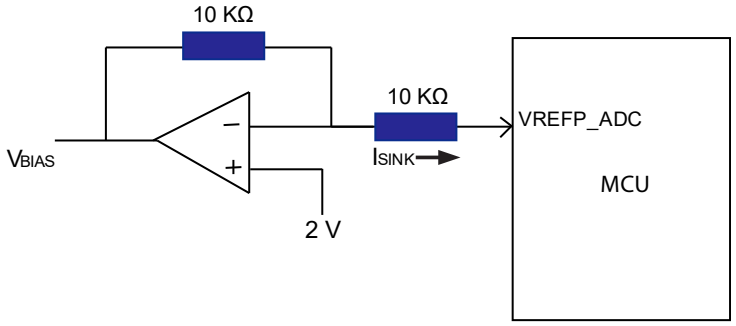


図 20-2：外部バイアス付き ADC サブシステム・リファレンス・バッファ

外部電圧リファレンス

ADC リファレンスとして外部リファレンス電圧源を選択するには、ADC_CFG.REFBUFEN ビットと ADC_CFG.VREFVBAT ビットをクリアし、VREFP_ADC ピンと GND_VREFADC ピンに外部電圧リファレンスを接続します。VBAT_ADC は、外部で VREFP_ADC に接続すると、リファレンス電圧として使用できます。外部電圧リファレンスが VBAT より大きくならないようにしてください。外部電圧リファレンスを使用する場合、0.1µF のコンデンサを VREFP_ADC ピンのできるだけ近くに配置することを推奨します。

リファレンス電圧選択の表に、リファレンス電圧を選択するための ADC_CFG レジスタ内の様々なビット・フィールドの設定を示します。

表 20-1：リファレンス電圧選択

Voltage	REFBUFEN	VREFSEL	VREFVBAT
1.25 V	1	1	0
2.5 V	1	0	0

表 20-1：リファレンス電圧選択（続き）

Voltage	REFBUFEN	VREFSEL	VREFVBAT
External Voltage Reference	0	X	0
VBAT	0	X	1

注：外部電圧リファレンスは、VREFP_ADC ピンを使用して ADC に接続します。内部リファレンスまたは VBAT を ADC リファレンスとして使用する際は、VREFP_ADC ピンをフロート状態のままにしておき、いずれの電圧源またはグラウンドにも接続しないでください。

ADC とリファレンス・バッファを使用しない場合、VREFP_ADC ピンは（外付けコンデンサに接続せずに）フロート状態のままにしておくことができます。

デジタル・オフセットのキャリブレーション

オフセット補正ブロックを使用して、ADC のリファレンス電圧のオフセット誤差を測定し補正することができます。正確さを維持するためにはキャリブレーションが必要です。

ADC のパワーアップ後、新しいキャリブレーション・サイクルを開始するには、ADC の準備完了後に ADC_CFG.STARTCAL ビットをセットする必要があります。キャリブレーションが完了すると、ADC_STAT.CALDONE ビットがセットされます。ADC_IRQ_EN.CALDONE ビットをセットすると、割込みをイネーブルできます。キャリブレーションの終了後に、ADC_CAL_WORD レジスタに新しいキャリブレーション・ワードがロードされます。このレジスタは、休止モードで保持されます。ADC_CAL_WORD レジスタは、ユーザが設定することもできます。

注：変換中に ADC_CAL_WORD レジスタを設定すると、次の変換サイクルから新しいキャリブレーション・ワードが有効になります。

ADC への電源供給

ADC、リファレンス・バッファ、および温度センサーは、リセット時にパワーダウンされます。これらの各ブロックは明示的にパワーアップする必要があります。パワーアップ・シーケンスは、使用する内部または外部のリファレンスに依存します。

リファレンスとして外部リファレンスまたは VBAT を使用

外部リファレンスを ADC リファレンスとして使用するには、パワーアップ時に以下のシーケンスを使用する必要があります。

1. ADC_CFG.PWRUP ビットをセットして、ADC をパワーアップします。
2. CLKG_CLK_CTL1 レジスタで設定された PCLK 周波数を基準として待機時間が 20µs 以上となるように、ADC_PWRUP.WAIT ビットを設定します。
3. ADC_CFG.VREFVBAT ビットをセットして、VBAT をリファレンス電圧として使用します。
4. 少なくとも 700µs 待機してから、ADC_CFG.VREFVBAT_DEL ビットをセットします。
5. ADC_CFG.EN ビットをアサートし、ADC をイネーブルします。

6. 割込みを待ち、ADC_STAT.RDY ビットに 1 を書き込んでクリアします。ADC_IRQ_EN.RDY ビットをセットして、割込みをイネーブルする必要があります。
7. ADC_CFG.STARTCAL ビットをセットしてキャリブレーション・サイクルを開始します。
8. ADC_STAT.CALDONE ビットがハイになるのを待ちます。

これで、ADC サブシステムは動作する準備ができました。

内部リファレンス・バッファの使用

内部リファレンス・バッファを使用するには、パワーアップ時に以下のシーケンスを使用する必要があります。

1. ADC_CFG.PWRUP ビットをセットして、ADC をパワーアップします。
2. CLKG_CLK_CTL1 レジスタで設定された PCLK 周波数を基準として待機時間が 20µs 以上となるように、ADC_PWRUP.WAIT フィールドを設定します。
3. ADC_CFG.VREFSEL ビットを使用して、リファレンス電圧として 1.25V または 2.5V を選択します。
4. ADC_CFG.REFBUFEN ビットをアサートします。
5. ADC_CFG.EN ビットをアサートします。
6. 少なくとも 3.5ms 待機します。

   汎用（GP）タイマーのいずれかを使用すると 3.5ms 間待機できます。この待機期間中、システムを Flexi モードに移行して電力を節約し、汎用タイマーの割込みでウェイクアップできます。
7. ADC_STAT.RDY ビットをポーリングし、セットされたら ADC は使用できる状態になります。ADC_STAT.RDY ビットに 1 を書き込んでクリアします。このビットに対する割込みは、ADC_IRQ_EN.RDY ビットをセットしてイネーブルする必要があります。
8. ADC_CFG.STARTCAL ビットをセットしてキャリブレーション・サイクルを開始します。
9. ADC_STAT.CALDONE ビットがハイになるのを待ちます。

これで ADC は変換の準備ができました。

休止状態

デバイスが休止状態になると、ADC サブシステムのすべてのコンポーネントが自動的にパワーダウンされます。これは、消費電力を最小限に抑えるために行われます。休止状態から復帰した後は、ADC への電源供給のセクションの説明のとおり、すべてのコンポーネントを明示的にパワーアップして変換が可能な状態にする必要があります。

ただし、ADC が休止状態にある間、キャリブレーション係数は ADC の内部レジスタに保持されています。このため、ADC は休止状態からウェイクアップして、ウェイクアップ時間が経過した直後に補正を行った変換サイクルを開始することができます。ADC_CFG.STARTCAL ビットをアサートすることにより、必要に応じて新しいキャリブレーション・サイクルを実行することができます。

ウェイクアップ時での新しいキャリブレーション・サイクルの実行が推奨されるのは、システムが長期間休止モードにあり、最後のキャリブレーション・サイクル以降に ADC の動作条件、温度、リファレンス電圧が変化している可能性がある場合です。

チップが休止モードになった場合、以下のレジスタ・フィールドが保持されます。

- ADC_CFG.VREFSEL
- ADC_CFG.SINKEN
- ADC_PWRUP.WAIT
- ADC_CAL_WORD.VALUE
- ADC_CNV_TIME.SAMPTIME
- ADC_AVG_CFG.FACTOR
- ADC_AVG_CFG.OS
- ADC_LIMx_LO.VALUE
- ADC_LIMx_HI.VALUE
- ADC_HYSx.VALUE
- ADC_HYSx.MONCYC
- ADC_CFG1.RBUFLP

注：デバイスが休止モードに移行する前に変換が実行されていても、その変換はデバイスの起動後には再開されません。

ADC_CFG.EN ビットをクリアして ADC をディスエーブルしてから、休止モードまたはシャットダウン・モードに移行する必要があります。

サンプリングと変換時間

ADC は以下のいずれかのフェーズで動作します。

- **アクイジション・フェーズ**：アクイジション・フェーズでは、コンデンサ・アレイを入力に直接接続して、完全に充電させます。必要な最小アクイジション時間は出力インピーダンスに依存します。このフェーズのタイミングは、ADC_CNV_TIME.SAMPTIME ビットでクロック・サイクル数の単位で設定します。

  アクイジション・フェーズは、（SAMPTIME + 1）ACLK サイクルです。

  この時間は、SAMPTIME の値と選択したクロック周波数（ACLK）に依存します。

  ACLK 周波数 = システム・クロック周波数/ACLKDIV

  ACLKDIV は CLKG_CLK_CTL1.ACLKDIVCNT レジスタ（9 ビット）に設定できます。

- **変換フェーズ**：アクイジション・フェーズの終了時に変換フェーズが開始されます。変換は逐次比較によって完了し、13 ACLK サイクルを要します。

  注：2 回の変換の間の時間は 100µs を超えてはなりません。言い換えれば、ADC の最小サンプリング・レートは 10kSPS です。

ADC を 10KSPS 未満のサンプリング・レート（100µs 以上のサンプリング時間）で動作させる必要がある場合、サンプリング間でダミー変換を実行する必要があります。必要なスループットを満たすために、このダミー・サンプルは破棄してください。

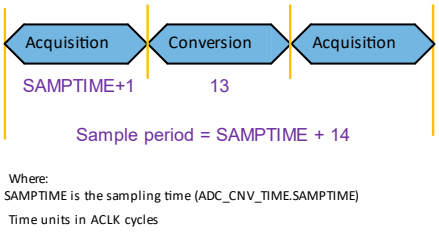


図 20-3：オーバーサンプリング無効およびゼロ遅延

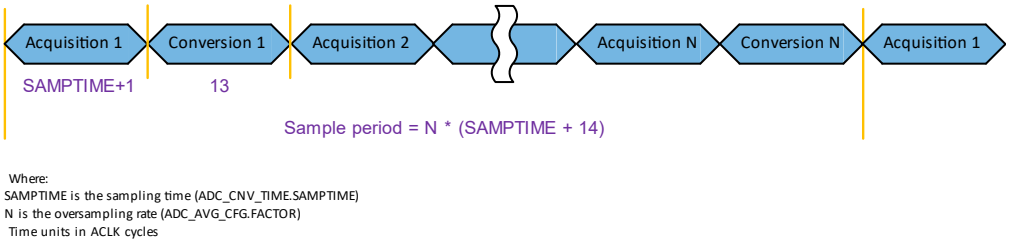


図 20-4：オーバーサンプリング有効およびゼロ遅延

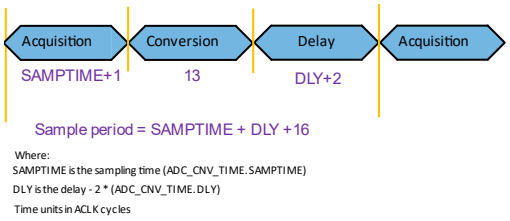


図 20-5：オーバーサンプリング無効および遅延 > 0

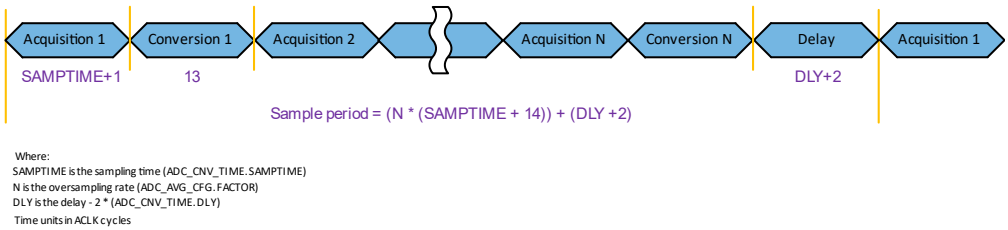


図 20-6：オーバーサンプリング無効および遅延 > 0

単一チャンネルの場合：

- `ADC_CNV_TIME.DLY` = 0 の場合、

$$\text{ADC Sampling Rate} = \frac{(\text{Root\_clock} / \text{ACLKDIV})}{(14 + \text{SAMPTIME}) \times (\text{Oversampling Rate})}$$

ここで、

Root_clock は、MCU が動作するシステム・クロックです。

Oversampling Rate は、必要な分解能を得るためにオーバーサンプリングおよび平均化が行われる ADC サンプルの数です。2 の累乗ステップで 1～256 の値を指定できます。

- ADC_CNV_TIME.DLY > 0 の場合、

$$\text{ADC Sampling Rate} = \frac{(\text{Root\_clock} / \text{ACLKDIV})}{((14 + \text{SAMPTIME}) \times (\text{Oversampling Rate})) + (\text{DLY} + 2)}$$

例えば、Root_clock が 26MHz、ACLKDIV が 10、SAMPTIME が 6 ACLK サイクル、DLY が 4 ACLK サイクル、Oversampling Rate が 1 の場合、ADC Sampling Rate は 100KSPS となります。

上記の例で 14 ビットの分解能が必要な場合、Oversampling Rate は 16 となり、ADC Sampling Rate は 7.975KSPS になります。

複数チャンネルの場合：

- ADC_CNV_TIME.DLY = 0 の場合、

$$\text{ADC Sampling Rate} = \frac{(\text{Root\_clock} / \text{ACLKDIV})}{(14 + \text{SAMPTIME}) \times (\text{Oversampling Rate}) \times (\text{No of Channels})}$$

- ADC_CNV_TIME.DLY > 0 の場合、

$$\text{ADC Sampling Rate} = \frac{(\text{Root\_clock} / \text{ACLKDIV})}{((14 + \text{SAMPTIME}) \times (\text{Oversampling Rate}) \times (\text{No of Channels})) + (\text{DLY} + 3)}$$

動作

ADC コントローラはいくつかの用例をサポートしています。

単一チャンネル・モード

ADC_CNV_CFG.SEL ビットを使用してチャンネルの 1 つを選択することで、特定のチャンネルで変換を実行するように ADC を構成できます。ADC はアナログ入力を変換し、CNV_DONE 割込みを発生させ、結果を該当する CHx_OUT レジスタに格納します。ADC_CNV_CFG.SEL ビットの特定のビットに書き込むことによって、そのチャンネルを有効にすることができます。例えば、チャンネル 2 を有効にするには、ADC_CNV_CFG.SEL［2］ビットをセットします。単一チャンネルでの変換（ADC_CNV_CFG.AUTOMODE = 0）では、（SEL［7：0］、BAT、TMP、TMP2 のうち）1 つのビットのみをセットするようにしてください。

選択したチャンネルで複数変換を実行するには、ADC_CNV_CFG.MULTI ビットをセットします。割込みをイネーブルした場合は割込みが生成され、ADC_STAT レジスタの該当するビットが各変換後にセットされます。変換出力は CHx_OUT レジスタに格納されます。

次の変換が終了する前に結果が出力レジスタから読み出されず、ステータス・ビットがクリアされないと、変換出力がオーバーフローし、新しい結果が CHx_OUT レジスタに格納され、データが失われます。

複数変換には DMA を使用することを推奨します。連続的な変換の間に遅延を設定することもできます。

注：必要な変換回数が完了したら、ADC_CNV_CFG.MULTI ビットをクリアして ADC サブシステムの変換を停止する必要があります。

自動サイクル・モード

自動サイクル・モードでは、個々のチャンネル・レジスタのサンプリングと読出しの MCU オーバーヘッドを削減できます。このモードにより、ユーザは ADC 入力チャンネルのシーケンスを選択し、すべてのチャンネルでの変換終了時に単一の割込みを発生させることができます。ただし、温度検出とバッテリ監視は、自動サイクル・モードでは実行できません。自動サイクル・モードは、ADC_CNV_CFG.AUTOMODE ビットをセットすることで有効になります。

有効化されたチャンネルのうち、最も小さい番号のチャンネルから変換が開始され、最も大きい番号のチャンネルまで実行されます。各チャンネルの出力はそれぞれ該当する CHx_OUT レジスタに格納され、すべてのチャンネル（1 つのシーケンス）の変換完了後に ADC_STAT レジスタの該当するビットがセットされます。

選択したチャンネルで複数変換を開始するには、ADC_CNV_CFG.MULTI ビットをセットします。各シーケンスの完了後に割込みが生成されます。連続するシーケンスの間に遅延を設定することができます。生成されるデータ量に対応するために、複数変換には DMA を使用することを推奨します。

注：必要な変換回数が完了したら、ADC_CNV_CFG.MULTI ビットをクリアして ADC サブシステムの変換を停止する必要があります。ADC_CNV_CFG.MULTI は、最低 1 ACLK サイクルの間ローでなければなりません。変換完了、アラート、およびオーバーフローの各ステータス・ビットは、次の変換を開始する前にクリアする必要があります。

変換間の遅延

ADC_CNV_TIME.DLY ビットを設定すると、1 つのチャンネル・シーケンスの完了から次のシーケンスの開始までの間、または 1 つのチャンネルでの複数変換の間に遅延を設定できます。この遅延は、ACLK サイクル数単位で設定します。

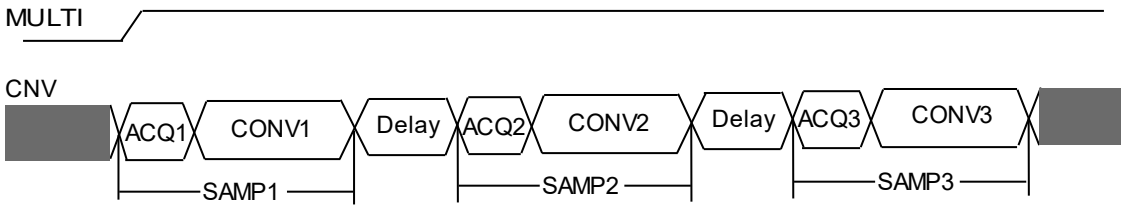


図 20-7：複数変換モードにおける単一チャンネルでの変換間の遅延

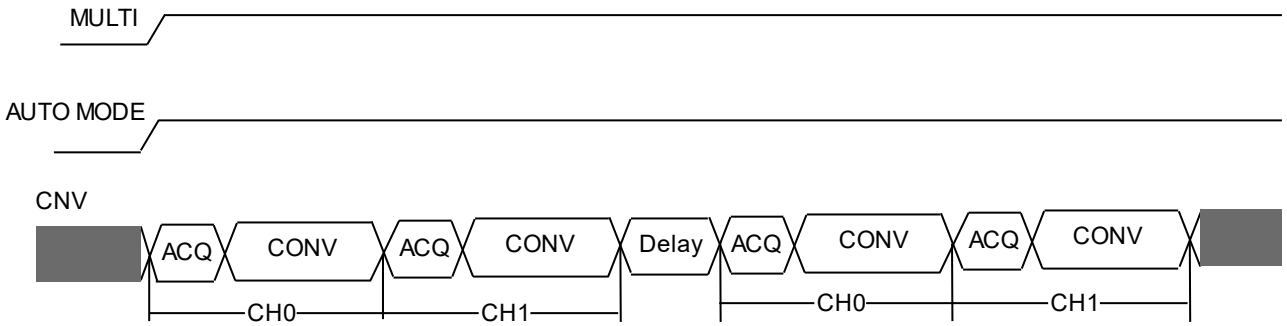


図 20-8：自動サイクル・モードにおけるシーケンス間の遅延

DMA

DMA チャンネルを使用すると、MCU オーバーヘッドを削減できます。この場合、必要な数の ADC 変換の完了後に単一割込みをアサートして ADC の結果を SRAM に直接移動します。DMA は、ADC_CNV_CFG.DMAEN ビットによりイネーブルできます。変換結果は、ADC_DMA_OUT.RESULT ビットから読み出す必要があります。DMA チャンネルでは、変換の数を設定できます。設定した変換回数が完了し、DMA 完了割込みが受信されるまでの期間、MCU を Flexi モードにすることができます。詳細については、プログラミング・フローのセクションを参照してください。

割込み

ADC コントローラには関連する割込みが備わっています。割込みステータス・レジスタは、割込み源を示します。DMA を使用しない場合、ADC コントローラは変換後にデータ割込みを生成します。

注：割込みはすべて、1 を書き込んでクリアする（Write One to Clear）タイプです

変換

ADC_IRQ_EN.CNVDONE ビットをセットすると、各変換の完了後に割込みを生成できます。変換完了後に、ADC_STAT レジスタの該当する *DONE* ビットがセットされます。

オーバーフロー

チャンネル上で次の変換が実行される前に、出力データがユーザまたは DMA によって出力レジスタから読み出されない場合、そのデータは上書きされます。ADC_OVF レジスタの各チャンネルでオーバーフロー・ビットを設定すると、割込みが生成されます。ユーザがクリアするまでセットされたままになります。この割込みは、ユーザが ADC_IRQ_EN.OVF ビットをセットしてイネーブルする必要があります。

注：変換終了時、ADC_CNV_CFG.SEL ビットは変更しないでそのままにしておく必要があります。

プログラミング・フロー

パワーアップとキャリブレーション（必要な場合）の後、ADC は変換の準備ができます。

単一チャンネルでの単一変換

このモードは、選択した 1 つのチャンネルで単一変換を実行するのに使用します。変換を実行するには、ADC_CNV_CFG.SINGLE ビットを 1 にセットする必要があります。このビットは、1 を書き込んでアクションするビットとして機能し、ゼロが読み出されます。単一データを読み出すだけで済むため、このモードでは DMA は推奨されません。変換結果は、該当するチャンネル出力レジスタの RESULT ビットから読み出すことができます。

以下の手順は、単一チャンネル（AIN2 など）で単一変換を実行するように ADC を設定する方法を説明しています。

1. ADC_CNV_CFG.SEL = 0x2 と設定して、変換にチャンネル 2 を選択します。
2. ADC_IRQ_EN.CNVDONE = 0x1 と設定して、変換完了時の割込みをイネーブルします。
3. ADC_CNV_CFG.SINGLE = 0x1 と設定して、変換を開始します。
4. 変換が終了すると、割込みが発生し、STAT［2］がセットされます。
5. cnv_result = ADC_CH0_OUT.RESULT を設定して、変換結果を読み出します。
6. ADC_STAT.DONE2 = 0x1 と設定して、割込みをクリアします。

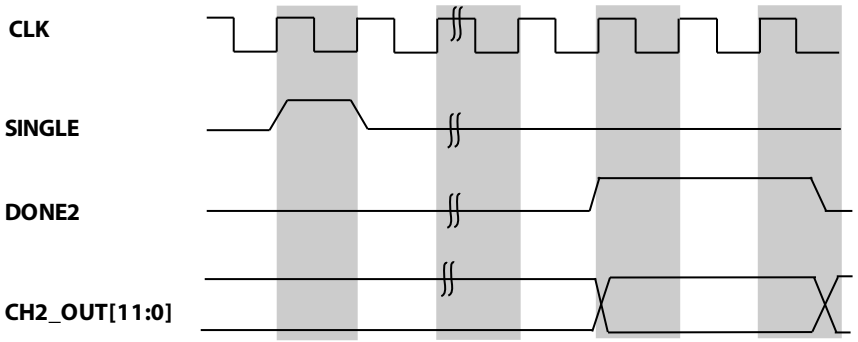


図 20-9：単一チャンネルでの単一変換

単一チャンネルでの複数変換

選択したチャンネルで複数変換を実行するには、ADC_CNV_CFG.MULTI を使用します。このビットがアサート解除されるまで、変換は継続します。このモードでは DMA を使用します。このモードは、チャンネルで複数のバックツーバックの単一変換を実行するのに似て、データの読出し（そうしないと、上書きされてしまいます）と変換の再開のためのオーバーヘッドが追加されます。DMA を使用する場合、実行する変換の数を DMA のカウント・ディスクリプタに設定する必要があります。

以下の手順は、単一チャンネル（ADC0_VIN2 など）で複数変換を実行するように ADC を設定する方法について説明しています。

1. ADC_CNV_CFG.SEL = 0x2 と設定して、変換にチャンネル 2 を選択します。
2. 次のように DMA を設定します。
   a. 変換数が 3 の場合は、DMA カウント = 2（DMA カウント = 変換数 − 1）。
   b. ソース・アドレス = ADC_DMA_OUT レジスタのアドレス。

c. ソース・サイズ = ハーフワード

d. 変換結果を格納するための DMA のディスティネーション・アドレスを SRAM メモリのロケーション・アドレスに設定。

e. 必要なインクリメント（ハーフワード）を DMA チャンネル制御データ構造体に設定。

3. ADC_CNV_CFG.DMAEN = 0x1 と設定して、DMA をイネーブルします。
4. ADC_CNV_TIME.DLY = 0x14 と設定して、2 回の変換の間の遅延を設定します。
5. ADC_CNV_CFG.MULTI = 0x1 と設定して、変換を開始します。

   3 回の変換（カウント= 2）後に dma_done 割込みが発生します。
6. ADC_CNV_CFG.MULTI = 0 と設定して MULTI をクリアし、ISR でのそれ以上の変換を無効にします。

変換結果は SRAM から読み出すことができます。

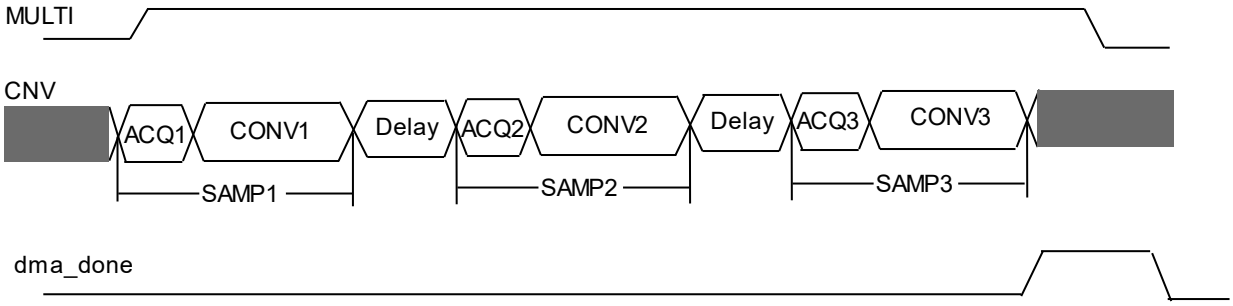


図 20-10：単一チャンネルでの複数変換

自動サイクル・モードでの単一変換

該当する ADC_CNV_CFG.SEL チャンネル・ビットを設定して、シーケンスに含めるチャンネルをイネーブルにします。

以下の手順は、自動サイクル・モード（ADC0_VIN2、ADC0_VIN4、および ADC0_VIN7 など）において一連のチャンネルで単一変換を実行するように ADC を設定する方法を説明しています。

1. ADC_CNV_CFG.SEL = 0x94 と設定します。
2. 次のように DMA を設定します。

   a. 変換数が 3 の場合は、DMA カウント = 2（DMA カウント = 変換数 − 1）。

   b. ソース・アドレス = ADC_DMA_OUT レジスタのアドレス。

   c. ソース・サイズ = ハーフワード

   d. 変換結果を格納するための DMA のディスティネーション・アドレスを、SRAM メモリのロケーション・アドレスに設定。

   e. 必要なインクリメント（ハーフワード）を DMA チャンネル制御データ構造体に設定。
3. ADC_CNV_CFG.DMAEN = 0x1 と設定して、DMA をイネーブルします（DMA を使用する場合）。
4. ADC_CNV_CFG.AUTOMODE = 0x1 と設定します。

5. ADC_CNV_TIME.DLY = 0x0 と設定します。
6. ADC_CNV_CFG.SINGLE = 0x1 と設定して変換を開始します。

   変換後に dma_done が生成されます。

変換結果は、SRAM または該当するチャンネル出力レジスタから読み出すことができます。

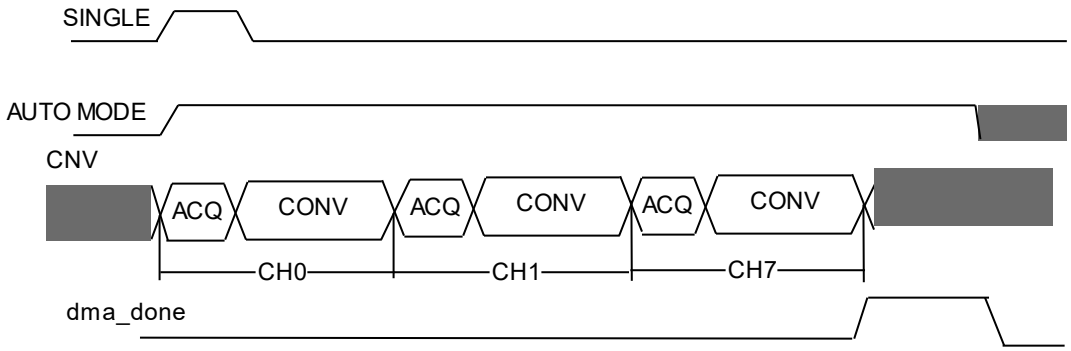


図 20-11：自動サイクル・モードでの単一変換

自動サイクル・モードでの複数変換

シーケンス数に選択したチャンネル数を掛けた数を DMA のカウント・ディスクリプタに設定します。

以下の手順は、自動サイクル・モード（ADC0_VIN0、ADC0_VIN2、および ADC0_VIN3 など）において、一連のチャンネルで複数変換を実行するように ADC を設定する方法を説明しています。

1. ADC_CNV_CFG.SEL = 0x0D と設定します。
2. ADC_CNV_TIME.DLY = 0x7E と設定します。
3. 次のように DMA を設定します。
   a. ソース・アドレス = ADC_DMA_OUT レジスタのアドレス。
   b. ソース・サイズ = ハーフワード
   c. 変換結果を格納するための DMA のディスティネーション・アドレスを、SRAM メモリのロケーション・アドレスに設定。
   d. 必要なインクリメント（ハーフワード）を DMA チャンネル制御データ構造体に設定。
   e. カウント = 5（2 シーケンスの場合）
4. ADC_CNV_CFG.DMAEN = 0x1 と設定します。
5. ADC_CNV_CFG.MULTI = 0x1 と設定して変換を開始します。

   変換後に dma_done が生成されます。
6. ADC_CNV_CFG.MULTI = 0 と設定して、ISR でそれ以上の変換を無効にします。

変換結果は SRAM から読み出すことができます。

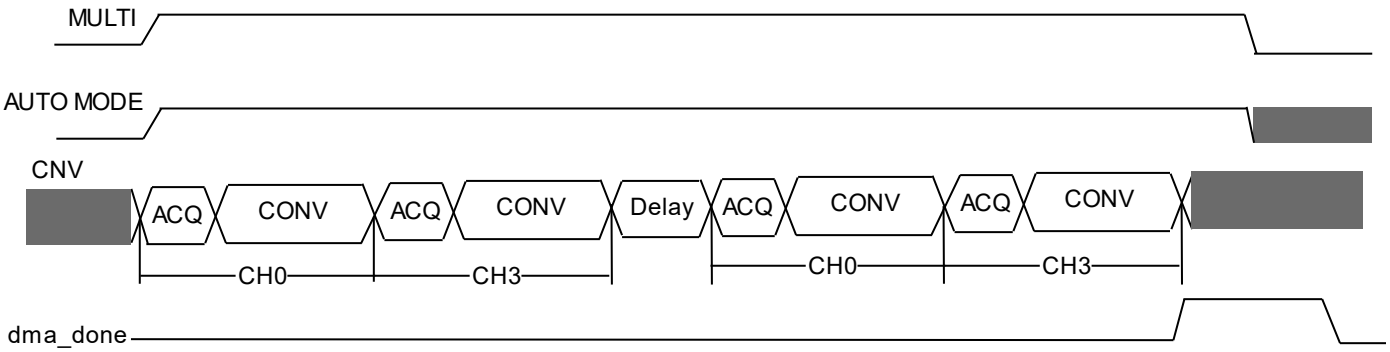


図 20-12：自動サイクル・モードでの複数変換

温度センサー

ADC サブシステムは、ダイ温度を測定する温度センサーを備えています。温度検出を有効にするには、ADC_CFG.TMPEN ビットをセットします。変換が開始されるまで 300µs 待機します。温度センサーを ADC またはリファレンス・バッファと一緒にイネーブルすると、時間を節約できます。

この温度センサーは、絶対周囲温度の計算ツールとして使用するようには設計されていません。これは、ADuCM4050 MCU ダイの温度のおおよその目安として使用します。

アクイジション時間が最小 65µs となるように ADC_CNV_TIME.SAMPTIME ビットを設定します。

温度を測定する方法を以下の手順で説明します。

1. ADC_CNV_CFG.TMP ビットを 1 にセットします。
2. ADC_CNV_CFG.SINGLE ビットをセットします。
3. 変換後に ADC_STAT.TMPDONE ビットがセットされます。

   ADC_IRQ_EN.CNVDONE ビットをセットしている場合、割込みが発生します。
4. ADC_TMP_OUT レジスタから結果を読み出します。
5. ADC_CNV_CFG.TMP2 ビットを 1 にセットします。
6. ADC_CNV_CFG.SINGLE ビットを 1 にセットします。
7. 変換後に ADC_STAT.TMP2DONE ビットがセットされます。

   ADC_IRQ_EN.CNVDONE ビットをセットしている場合、割込みが発生します。
8. 変換後に ADC_TMP2_OUT レジスタから結果を読み出します。

自動モードも使用できます。

ADC_CNV_CFG.TMP ビットと ADC_CNV_CFG.TMP2 ビットをセットします。

ADC_CNV_CFG.SINGLE ビットまたは ADC_CNV_CFG.MULTI ビット（複数変換の場合）をセットして、変換を開始します。

温度は次式で計算できます。

T（ºC）= [[[ADC_TMP_OUT − （$Offset_{code}$ * 1.25 / Vref）] / （ADC_TMP2_OUT + （RG * ADC_TMP_OUT）] * ［$Rvirtual_{reference}$/$ideal_{sensitivity}$］] − 273.15

ここで、

- RG = 1.1829
- $Rvirtual_{reference}$ = 1.2256
- $ideal_{sensitivity}$ = 0.0012411
- $Offset_{code}$ = 161

例えば、ADC_TMP_OUT が 1620、ADC_TMP2_OUT が 2102 の場合、温度は 85ºC です。

バッテリ監視

バッテリ監視を有効にするには、ADC_CNV_CFG.BAT ビットをセットする必要があります。アクイジション時間を 500ns にするように ADC_CNV_TIME.SAMPTIME ビットを設定する必要があります。ADC_CNV_CFG.SINGLE をセットして、変換を開始します。ADC 変換出力は、変換モードに応じて ADC_BAT_OUT レジスタから読み出すことができます。

変換完了後に割込みを生成させるために、変換完了割込みをイネーブルすることができます。変換後に ADC_CNV_CFG.BAT ビットをクリアすると、消費電力が低減します。

ADC の結果をバッテリ電圧に変換するには、次式を使用します。

VBAT = 4 * ADC_BAT_OUT* Vref /（2^{12} − 1）

ここで、

- VBAT はバッテリ電圧
- Vref は選択したリファレンス電圧（1.25V を推奨）
- ADC_BAT_OUT に ADC 変換出力が得られます。

オーバーサンプリング

オーバーサンプリングによって 12 ビットを超える分解能が実現できます。オーバーサンプリングは、ADC_AVG_CFG.OS ビットと ADC_AVG_CFG.EN ビットに 1 を書き込むことで有効にできます。オーバーサンプリングを有効にした場合、ADC は複数回サンプリングし、結果を平均化して、CHx_OUT レジスタに必要な分解能で出力します。必要な分解能を得るために ADC_AVG_CFG.FACTOR フィールドに設定する値を、**高分解能化係数**の表に示します。

表 20-2：高分解能化係数

Resolution Required	AVG_CFG_FACTOR to be programmed	Number of Samples Used
13-bit	h02	4
14-bit	h08	16
15-bit	h20	64
16-bit	h80	256

平均化機能

複数変換を実行する場合、選択したすべてのチャンネルで平均化を有効にすることができます。平均化は、2 の累乗（つまり、2、4、8、16...256）のステップで、最大 256 のサンプルに対して実行できます。累算された値は切り捨てられ、CHx_OUT レジスタに 12 ビットの平均化出力が得られます。ADC_AVG_CFG.OS ビットを 0 に、ADC_AVG_CFG.EN ビットを 1 に書き込むことで、平均化を有効にできます。

平均化係数は、ADC_AVG_CFG.FACTOR ビットに係数/2 として設定します。

例えば、64 サンプルを平均化するには、ADC_AVG_CFG.FACTOR を 32（0x20）に設定します。平均化の完了後に割込みが発生します。

自動サイクル・モードの場合、DMA をイネーブルすることができます。

例えば、16 サンプルを平均化するには、16 個の値に対して変換と加算が実行され、更に 16 で除算されます。

自動モード（複数チャンネルの場合）で平均化を有効にした場合、1 つのチャンネルで設定した変換回数が実行され（および平均化され）てから次のチャンネルに移動します。

単一チャンネルでの複数変換の平均化

以下の手順は、単一チャンネル（ADC0_VIN2 など）で複数変換を平均化する方法について説明しています。

1. ADC_CNV_CFG.SEL = 0x2 と設定して、変換にチャンネル 2 を選択します。
2. ADC_IRQ_EN.CNVDONE = 0x1 と設定して、変換完了時の割込みをイネーブルします。
3. ADC_AVG_CFG.OS = 0 と設定します。
4. ADC_AVG_CFG.EN = 0x1 と設定して、平均化を有効にします。
5. 64 サンプルを平均化するように ADC_AVG_CFG.FACTOR = 0x20 と設定します。
6. ADC_CNV_CFG.SINGLE = 0x1 と設定して、変換を開始します。
7. 変換が終了すると、割込みが発生し、ADC_STAT.DONE2 がセットされます。
8. cnv_result = ADC_CH2_OUT.RESULT と設定して、変換結果を読み出します。
9. ADC_STAT.DONE2 = 0x1 と設定して、割込みをクリアします。

自動サイクル・モードでの複数変換の平均化

以下の手順は、自動サイクル・モードにおいて一連のチャンネルで複数変換を平均化する方法を説明しています。

1. ADC_CNV_CFG.SEL = 0x30 と設定して、変換にチャンネル 5 とチャンネル 4 を選択します。
2. 次のように DMA を設定します。
   a. DMA カウント = 1 として、2 つのチャンネルを平均化（DMA カウント = 変換数 − 1）。
   b. ソース・アドレス = ADC_DMA_OUT レジスタのアドレス。
   c. ソース・サイズ = ハーフワード
   d. 変換結果を格納するための DMA のディスティネーション・アドレスを、SRAM メモリのロケーション・アドレスに設定。
   e. ディスティネーションでの必要なインクリメントを設定します。
3. ADC_CNV_CFG.DMAEN = 0x1 と設定して、DMA をイネーブルします（DMA を使用する場合）。
4. ADC_CNV_CFG.AUTOMODE = 0x1 と設定して、変換を開始します。
5. ADC_AVG_CFG.OS = 0 と設定します。
6. ADC_AVG_CFG.EN = 0x1 と設定します。
7. 16 サンプルを平均化するように ADC_AVG_CFG.FACTOR = 0x08 と設定します。
8. ADC_CNV_CFG.SINGLE = 0x1 と設定して変換を開始します。

   変換後に dma_done が生成されます。

変換結果は、SRAM または該当するチャンネル出力レジスタから読み出すことができます。

注：平均化／オーバーサンプリングと監視を同時に使用することはできません。これらは相互に排他的です。

ADC デジタル・コンパレータ

ADC 入力が設定可能な閾値を上回るか下回ると、デバイスのデジタル・コンパレータによって割込みをトリガできます。ADC0_VIN0、ADC0_VIN1、ADC0_VIN2、ADC0_VIN3 の各入力チャンネルがデジタル・コンパレータで使用できます。これを使用すると、それらのチャンネル（またはチャンネル・セット）のいずれかが正常な値の範囲内にあるかどうかを継続的に監視できます。コンパレータは、単一チャンネルまたは自動サイクル・モードのいずれかでの複数変換でのみイネーブルにできます。ユーザが監視中に結果を読み出すことは想定されていないため、オーバーフロー表示は無効にされます。

ADC デジタル・コンパレータを設定するには、以下のようにします。

- 下限閾値は、12 ビットの ADC_LIMx_LO.VALUE フィールドに書き込む必要があります。下限との比較を可能にするには、該当するレジスタの EN ビットをセットする必要があります。
- 上限閾値は、12 ビットの ADC_LIMx_HI.VALUE フィールドに書き込む必要があります。比較を有効にするには、該当するレジスタの EN ビットをセットする必要があります。

- ヒステリシス値についても、ADC_HYSx.VALUE ビットでこれらの各チャンネルに設定できます。これは、ADC_HYSx.EN ビットをセットして有効にする必要があります。
- ADC の結果が閾値（ADC_LIMx_LO.VALUE または ADC_LIMx_HI.VALUE）を超え、ADC_HYSx.MONCYC ビットに設定されたクロック・サイクル数内に通常値に戻らない場合、アラートが示されます。

  通常値は以下のように定義されています。

  - 下限閾値の場合：ADC_LIMx_LO.VALUE + ADC_HYSx.VALUE を超えている
  - 上限閾値の場合：ADC_LIMx_HI.VALUE − ADC_HYSx.VALUE 未満

変換結果が ADC_LIMx_HI.VALUE を超えている場合、次の MONCYC 変換で監視されます。

監視中に変換結果が ADC_LIMx_HI.VALUE − ADC_HYSx.VALUE を超えている場合、アラートが示されます。

監視中の変換結果が ADC_LIMx_HI.VALUE − ADC_HYSx.VALUE 未満の場合、アラートは示されず、監視が停止します。

同様に、変換結果が ADC_LIMx_LO.VALUE 未満の場合は、次の MONCYC 変換で監視されます。

監視中に変換結果が ADC_LIMx_LO.VALUE + ADC_HYSx.VALUE 未満のままである場合は、アラートが示されます。

監視中の変換結果が ADC_LIMx_LO.VALUE + ADC_HYSx.VALUE を超えている場合、アラートは示されず、監視が停止します。

ADC_ALERT レジスタを読み出すと、アラート源を特定することができます。ADC_IRQ_EN.ALERT ビットをセットしている場合、割込みが発生します。

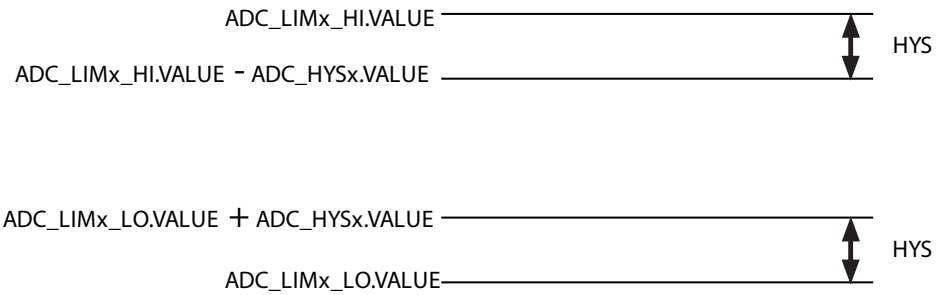


図 20-13：下限値およびヒステリシス

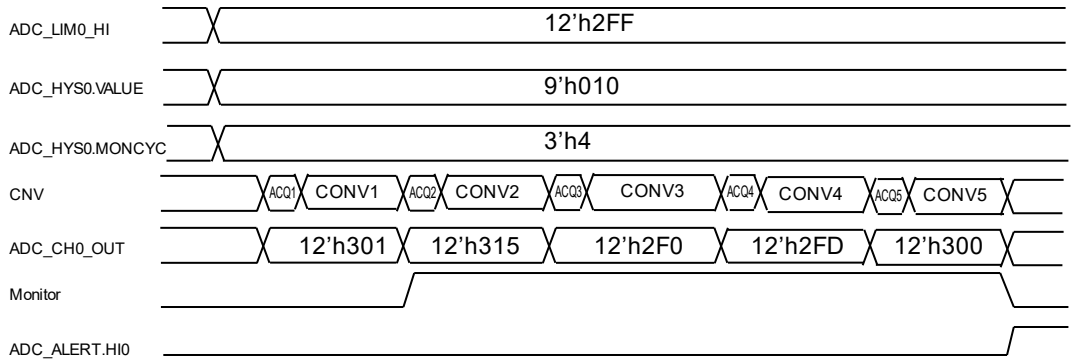


図 20-14：アラート機能

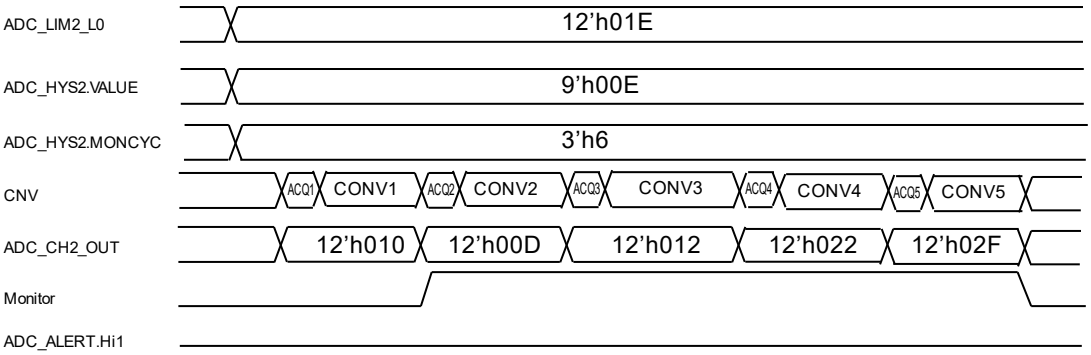


図 20-15：変換結果が MONCYC サイクル内に通常値に戻ったときにはアラートは示されない

自動サイクル・モードでは、チャンネルは順に変換されていきます。いずれかのチャンネルが閾値を超えた場合、シーケンスの次のチャンネルに移動する前に監視が行われます。チャンネル入力が MONCYC サイクル内に通常値に戻るか、MONCYC サイクル後にアラートが発生すると、監視は停止します。

注：ヒステリシスは、DMA をイネーブルしている場合はサポートされません。

次のフローチャートは、自動サイクル・モードにおいて 3 つのチャンネルで比較を有効にしている場合の変換フローを説明しています。

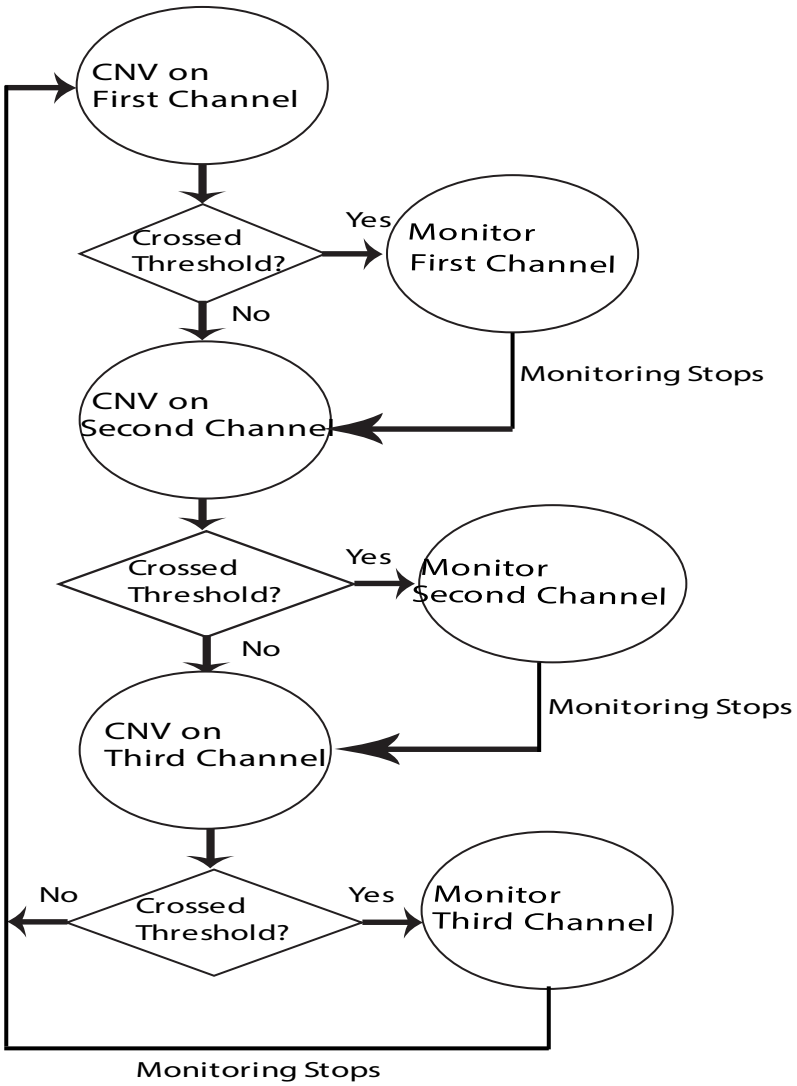


図 20-16：自動サイクル・モードにおいて 3 つのチャンネルで比較を有効にしている場合の変換フロー

ADuCM4050 ADC レジスタの説明

ADC 用のデジタル・コントローラには、以下のレジスタがあります。

表 20-3：ADuCM4050 ADC レジスター覧

レジスタ名	説明
ADC_ALERT	アラート表示
ADC_AVG_CFG	平均化の設定
ADC_BAT_OUT	バッテリ監視結果
ADC_CAL_WORD	キャリブレーション・ワード
ADC_CFG	ADC の設定

表 20-3：ADuCM4050 ADC レジスター一覧（続き）

レジスタ名	説明
ADC_CFG1	リファレンス・バッファ低消費電力モード
ADC_CH0_OUT	変換結果チャンネル 0
ADC_CH1_OUT	変換結果チャンネル 1
ADC_CH2_OUT	変換結果チャンネル 2
ADC_CH3_OUT	変換結果チャンネル 3
ADC_CH4_OUT	変換結果チャンネル 4
ADC_CH5_OUT	変換結果チャンネル 5
ADC_CH6_OUT	変換結果チャンネル 6
ADC_CH7_OUT	変換結果チャンネル 7
ADC_CNV_CFG	ADC 変換の設定
ADC_CNV_TIME	ADC 変換時間
ADC_DMA_OUT	DMA 出力レジスタ
ADC_HYS0	チャンネル 0 のヒステリシス
ADC_HYS1	チャンネル 1 のヒステリシス
ADC_HYS2	チャンネル 2 のヒステリシス
ADC_HYS3	チャンネル 3 のヒステリシス
ADC_IRQ_EN	割込みイネーブル
ADC_LIM0_HI	チャンネル 0 の上限
ADC_LIM0_LO	チャンネル 0 の下限
ADC_LIM1_HI	チャンネル 1 の上限
ADC_LIM1_LO	チャンネル 1 の下限
ADC_LIM2_HI	チャンネル 2 の上限
ADC_LIM2_LO	チャンネル 2 の下限
ADC_LIM3_HI	チャンネル 3 の上限
ADC_LIM3_LO	チャンネル 3 の下限
ADC_OVF	出力レジスタのオーバーフロー
ADC_PWRUP	ADC パワーアップ時間
ADC_STAT	ADC ステータス
ADC_TMP2_OUT	温度結果 2
ADC_TMP_OUT	温度結果

アラート表示

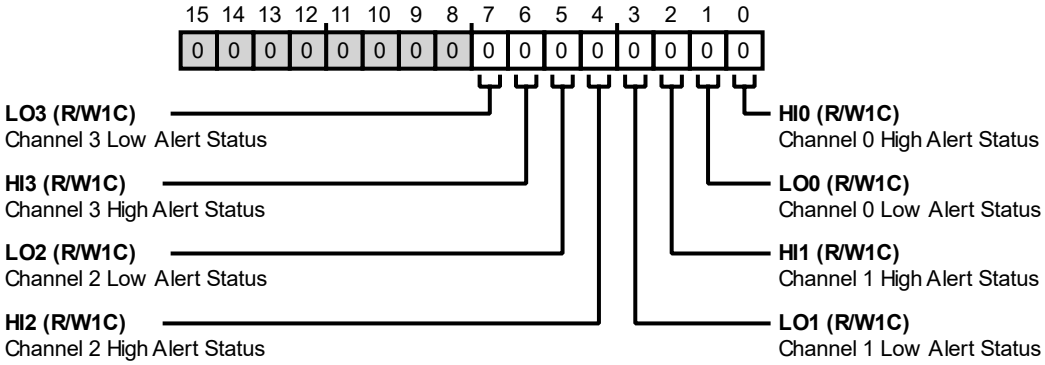


図 20-17：ADC_ALERT レジスタ図

表 20-4：ADC_ALERT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
7 (R/W1C)	LO3	チャンネル 3 ロー・アラート・ステータス。
6 (R/W1C)	HI3	チャンネル 3 ハイ・アラート・ステータス。
5 (R/W1C)	LO2	チャンネル 2 ロー・アラート・ステータス。
4 (R/W1C)	HI2	チャンネル 2 ハイ・アラート・ステータス。
3 (R/W1C)	LO1	チャンネル 1 ロー・アラート・ステータス。
2 (R/W1C)	HI1	チャンネル 1 ハイ・アラート・ステータス。
1 (R/W1C)	LO0	チャンネル 0 ロー・アラート・ステータス。
0 (R/W1C)	HI0	チャンネル 0 ハイ・アラート・ステータス。

平均化の設定

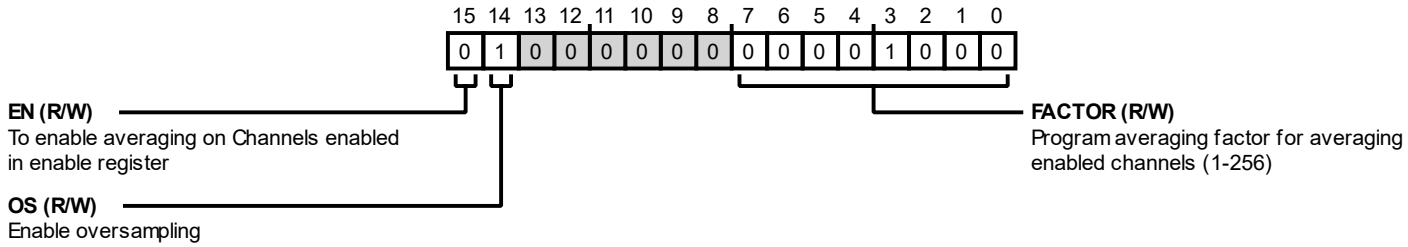


図 20-18：ADC_AVG_CFG レジスタ図

表 20-5：ADC_AVG_CFG レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W)	EN	イネーブル・レジスタで有効にされているチャンネルで平均化を有効化。
14 (R/W)	OS	オーバーサンプリングを有効化。
7:0 (R/W)	FACTOR	平均化が有効にされているチャンネルで平均化係数を設定（1～256）。

バッテリ監視結果

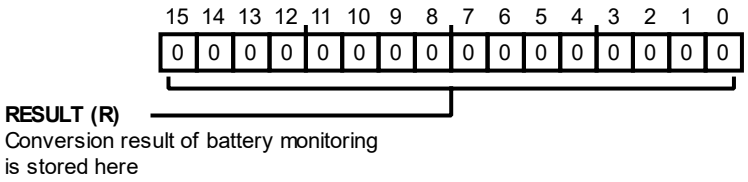


図 20-19：ADC_BAT_OUT レジスタ図

表 20-6：ADC_BAT_OUT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	RESULT	バッテリ監視の変換結果がここに格納されます。

キャリブレーション・ワード

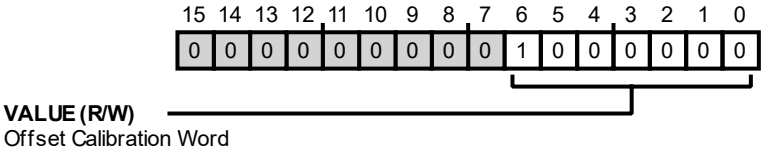


図 20-20：ADC_CAL_WORD レジスタ図

表 20-7：ADC_CAL_WORD レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
6:0 (R/W)	VALUE	オフセット・キャリブレーション・ワード

ADC の設定

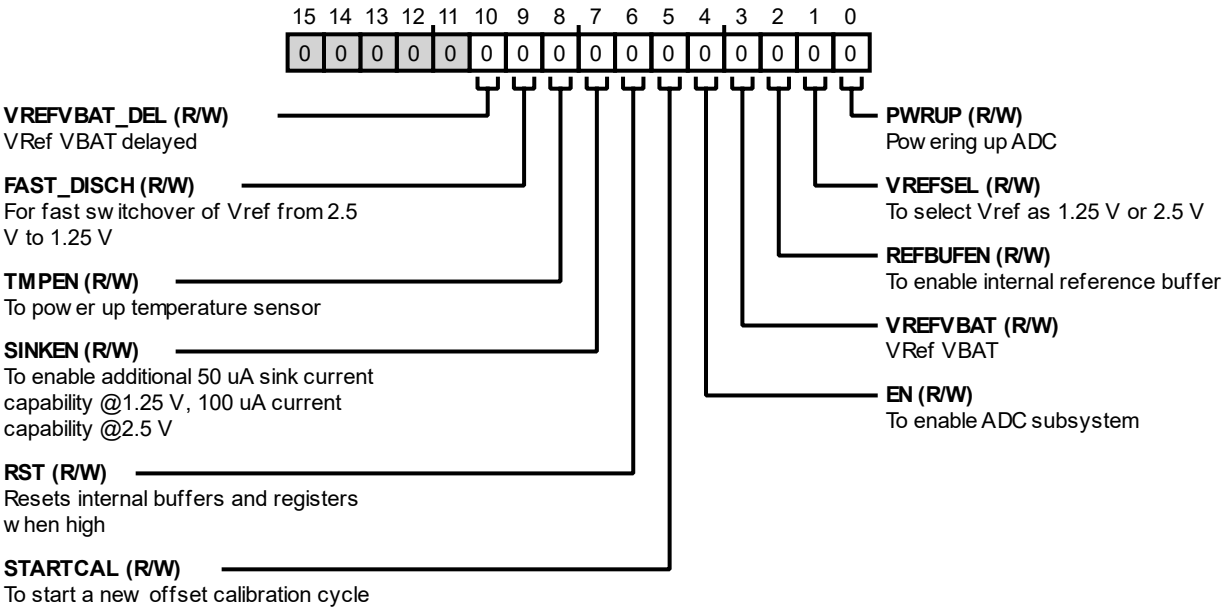


図 20-21：ADC_CFG レジスタ図

表 20-8：ADC_CFG レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
10 (R/W)	VREFVBAT_DEL	VRef VBAT の遅延。 `ADC_CFG.VREFVBAT` フィールドを 1 にセットしてから 700µs 以上遅延した後に、1 にセットされます。注：`ADC_CFG.VREFVBAT_DEL` と `ADC_CFG.VREFVBAT` は共に 0 にリセットする必要があります。
9 (R/W)	FAST_DISCH	Vref を 2.5V から 1.25V に高速切替え。
8 (R/W)	TMPEN	温度センサーのパワーアップ。
7 (R/W)	SINKEN	1.25V での追加の 50µA シンク電流能力、2.5V での追加の 100µA シンク電流能力をイネーブル。
6 (R/W)	RST	ハイのときに内部バッファとレジスタをリセット。
5 (R/W)	STARTCAL	新しいオフセット・キャリブレーション・サイクルを開始。
4 (R/W)	EN	ADC サブシステムをイネーブル。

表 20-8：ADC_CFG レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
3 (R/W)	VREFVBAT	VRef VBAT。 1 をセットすると、VBAT を Vref として選択。このフィールドを設定してから 700µs 以上遅延した後、ADC_CFG.VREFVBAT_DEL フィールドを 1 にセットする必要があります。 注：ADC_CFG.VREFVBAT_DEL と ADC_CFG.VREFVBAT は共に 0 にリセットする必要があります。	
2 (R/W)	REFBUFEN	内部リファレンス・バッファをイネーブル。	
		0	外部リファレンスを使用
		1	リファレンス・バッファをイネーブル
1 (R/W)	VREFSEL	Vref に 1.25V または 2.5V を選択。	
		0	Vref = 2.5 V
		1	Vref = 1.25 V
0 (R/W)	PWRUP	ADC をパワーアップ。	

リファレンス・バッファ低消費電力モード

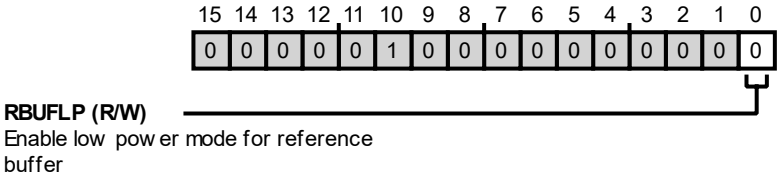


図 20-22：ADC_CFG1 レジスタ図

表 20-9：ADC_CFG1 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
0 (R/W)	RBUFLP	リファレンス・バッファの低消費電力モードを有効化。

変換結果チャンネル 0

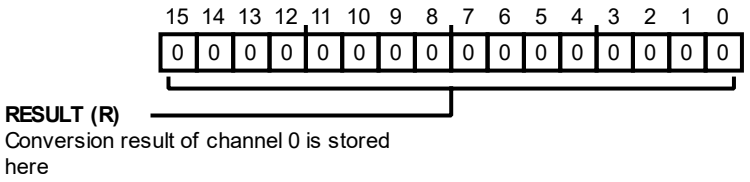


図 20-23：ADC_CH0_OUT レジスタ図

表 20-10：ADC_CH0_OUT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	RESULT	チャンネル 0 の変換結果がここに格納されます。

変換結果チャンネル 1

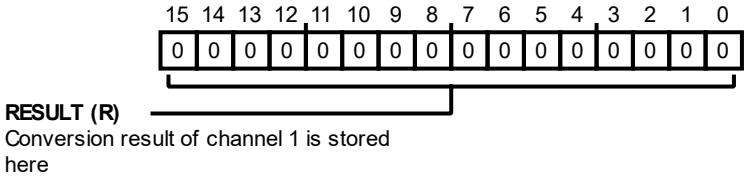


図 20-24：ADC_CH1_OUT レジスタ図

表 20-11：ADC_CH1_OUT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	RESULT	チャンネル 1 の変換結果がここに格納されます。

変換結果チャンネル 2

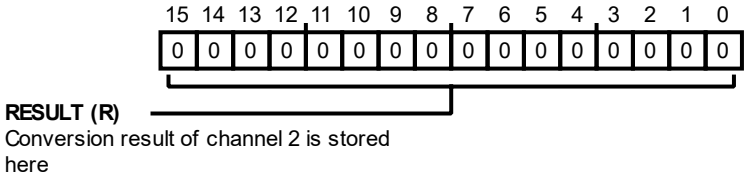


図 20-25：ADC_CH2_OUT レジスタ図

表 20-12：ADC_CH2_OUT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	RESULT	チャンネル 2 の変換結果がここに格納されます。

変換結果チャンネル 3

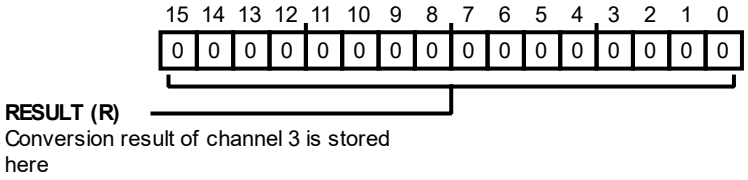


図 20-26：ADC_CH3_OUT レジスタ図

表 20-13：ADC_CH3_OUT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	RESULT	チャンネル 3 の変換結果がここに格納されます。

変換結果チャンネル 4

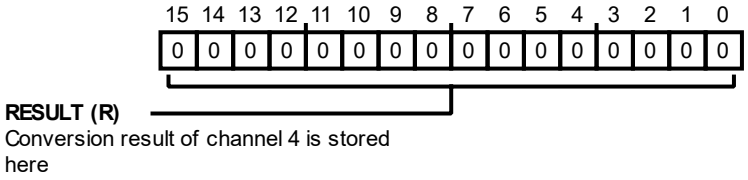


図 20-27：ADC_CH4_OUT レジスタ図

表 20-14：ADC_CH4_OUT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	RESULT	チャンネル 4 の変換結果がここに格納されます。

変換結果チャンネル 5

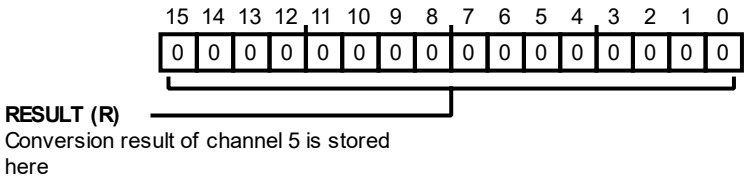


図 20-28：ADC_CH5_OUT レジスタ図

表 20-15：ADC_CH5_OUT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	RESULT	チャンネル 5 の変換結果がここに格納されます。

変換結果チャンネル 6

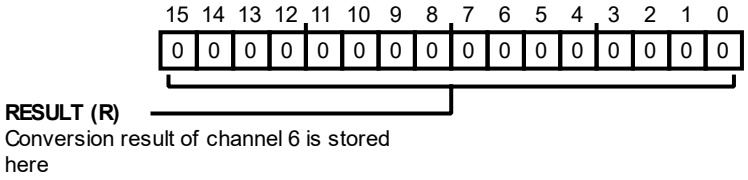


図 20-29：ADC_CH6_OUT レジスタ図

表 20-16：ADC_CH6_OUT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	RESULT	チャンネル 6 の変換結果がここに格納されます。

変換結果チャンネル 7

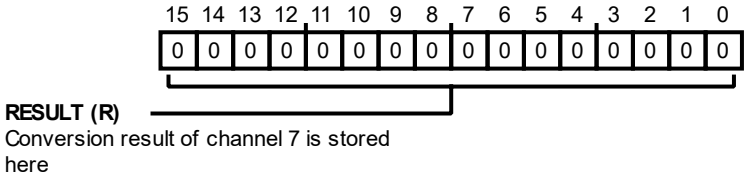


図 20-30：ADC_CH7_OUT レジスタ図

表 20-17：ADC_CH7_OUT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	RESULT	チャンネル 7 の変換結果がここに格納されます。

ADC 変換の設定

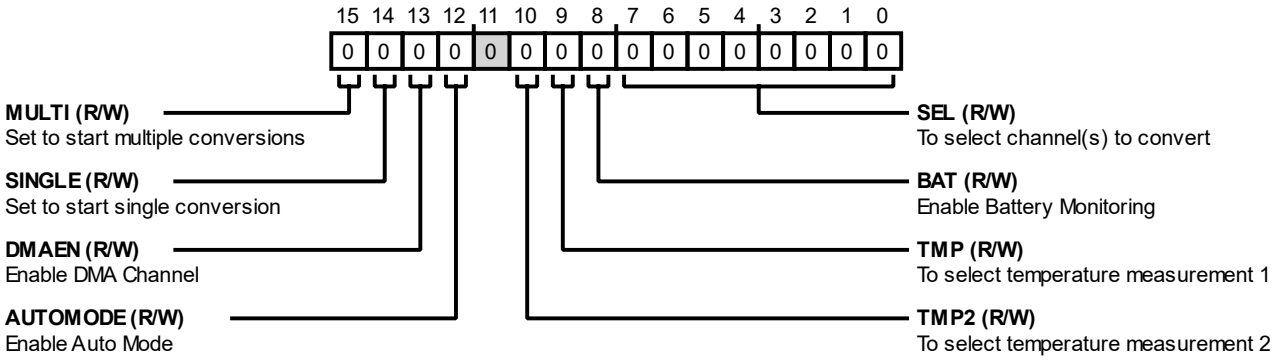


図 20-31：ADC_CNV_CFG レジスタ図

表 20-18：ADC_CNV_CFG レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15 (R/W)	MULTI	セットすると、複数変換を開始。
14 (R/W)	SINGLE	セットすると、単一変換を開始。
13 (R/W)	DMAEN	DMA チャンネルを有効化。
12 (R/W)	AUTOMODE	自動モードを有効化。
10 (R/W)	TMP2	温度測定 2 を選択。
9 (R/W)	TMP	温度測定 1 を選択。
8 (R/W)	BAT	バッテリ監視を有効化。
7:0 (R/W)	SEL	変換するチャンネルを選択。

ADC 変換時間

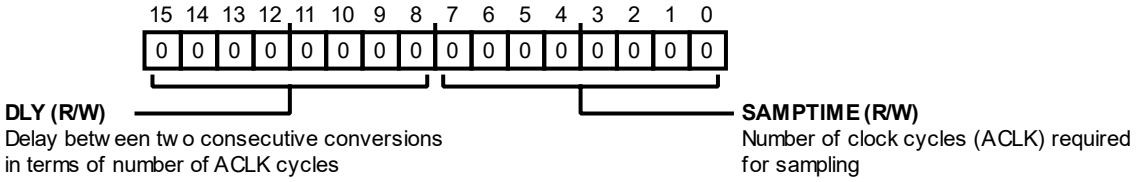


図 20-32：ADC_CNV_TIME レジスタ図

表 20-19：ADC_CNV_TIME レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:8 (R/W)	DLY	2 つの連続する変換の間の遅延（ACLK サイクル数単位）。
7:0 (R/W)	SAMPTIME	サンプリングに必要なクロック・サイクル数（ACLK）。

DMA 出力レジスタ

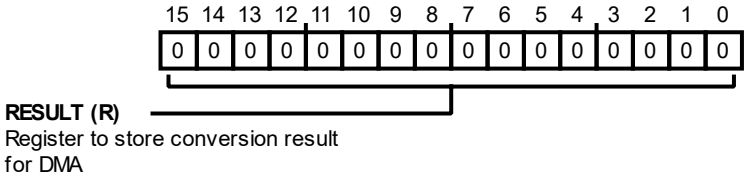


図 20-33：ADC_DMA_OUT レジスタ図

表 20-20：ADC_DMA_OUT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	RESULT	DMA の変換結果を保存するためのレジスタ。

チャンネル 0 のヒステリシス

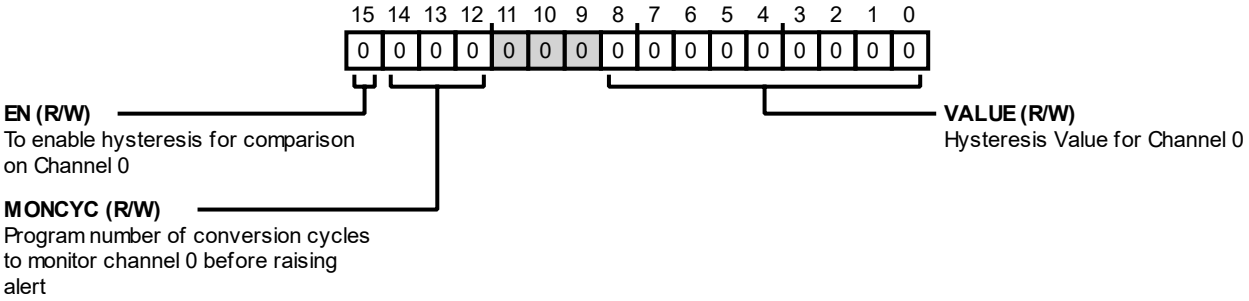


図 20-34：ADC_HYS0 レジスタ図

表 20-21：ADC_HYS0 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W)	EN	チャンネル 0 での比較にヒステリシスを有効化。
14:12 (R/W)	MONCYC	アラートを発生させるまでにチャンネル 0 を監視する変換サイクル数。
8:0 (R/W)	VALUE	チャンネル 0 のヒステリシス値。

チャンネル 1 のヒステリシス

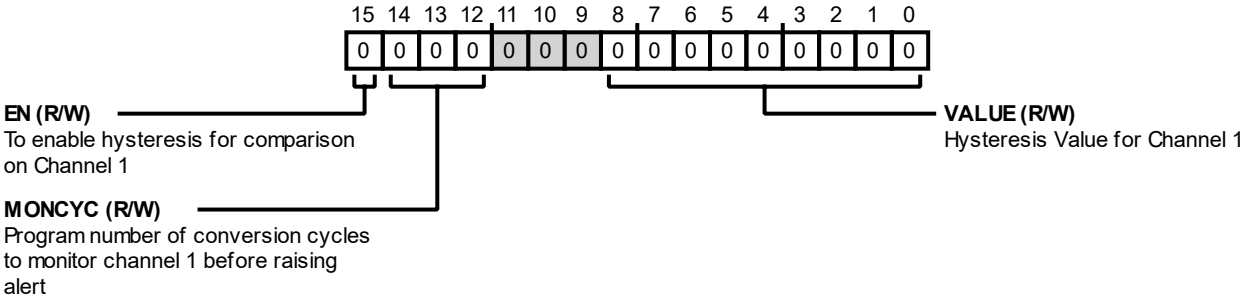


図 20-35：ADC_HYS1 レジスタ図

表 20-22：ADC_HYS1 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W)	EN	チャンネル 1 での比較にヒステリシスを有効化。
14:12 (R/W)	MONCYC	アラートを発生させるまでにチャンネル 1 を監視する変換サイクル数。
8:0 (R/W)	VALUE	チャンネル 1 のヒステリシス値。

チャンネル 2 のヒステリシス

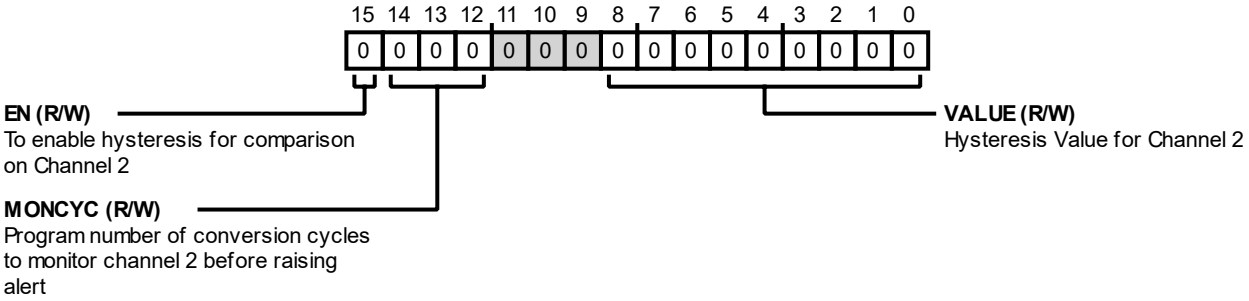


図 20-36：ADC_HYS2 レジスタ図

表 20-23：ADC_HYS2 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W)	EN	チャンネル 2 での比較にヒステリシスを有効化。
14:12 (R/W)	MONCYC	アラートを発生させるまでにチャンネル 2 を監視する変換サイクル数。
8:0 (R/W)	VALUE	チャンネル 2 のヒステリシス値。

チャンネル 3 のヒステリシス

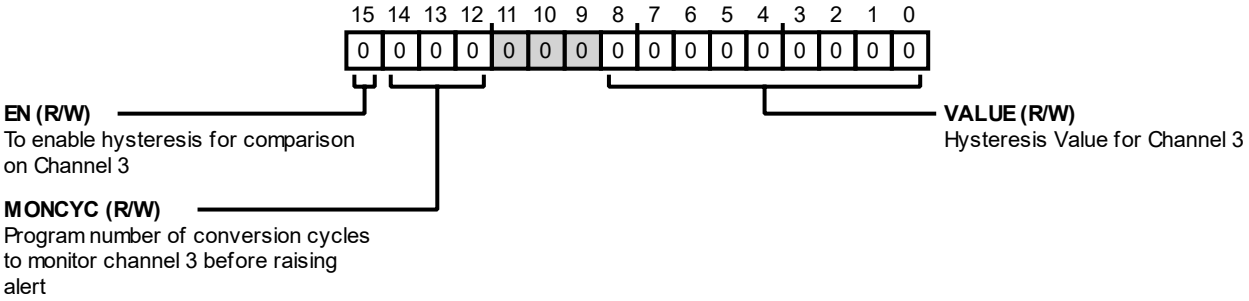


図 20-37：ADC_HYS3 レジスタ図

表 20-24：ADC_HYS3 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W)	EN	チャンネル 3 での比較にヒステリシスを有効化。
14:12 (R/W)	MONCYC	アラートを発生させるまでにチャンネル 3 を監視する変換サイクル数。
8:0 (R/W)	VALUE	チャンネル 3 のヒステリシス値。

割込みイネーブル

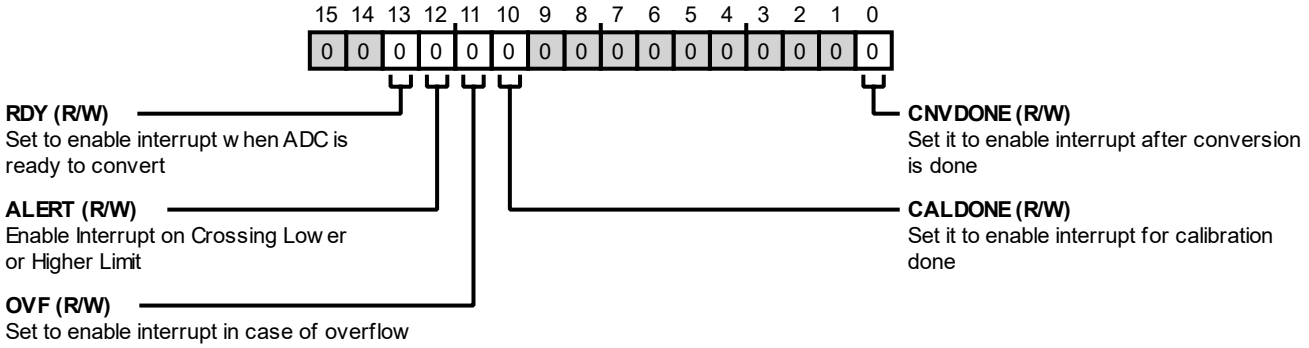


図 20-38：ADC_IRQ_EN レジスタ図

表 20-25：ADC_IRQ_EN レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
13 (R/W)	RDY	セットすると、ADC の変換準備ができたときの割込みをイネーブル。
12 (R/W)	ALERT	セットすると、下限または上限を超えたときの割込みをイネーブル。
11 (R/W)	OVF	セットすると、オーバーフロー時の割込みをイネーブル。
10 (R/W)	CALDONE	セットすると、キャリブレーション完了時の割込みをイネーブル。
0 (R/W)	CNVDONE	セットすると、変換完了後の割込みをイネーブル。

チャンネル 0 の上限

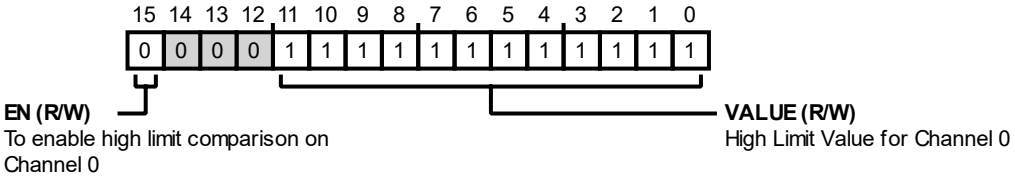


図 20-39：ADC_LIM0_HI レジスタ図

表 20-26：ADC_LIM0_HI レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W)	EN	チャンネル 0 での上限比較を有効化。
11:0 (R/W)	VALUE	チャンネル 0 の上限値。

チャンネル 0 の下限

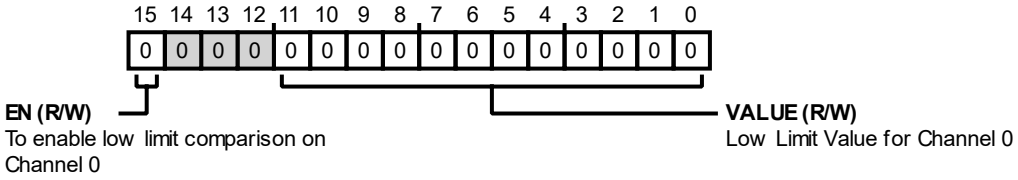


図 20-40：ADC_LIM0_LO レジスタ図

表 20-27：ADC_LIM0_LO レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W)	EN	チャンネル 0 での下限比較を有効化。
11:0 (R/W)	VALUE	チャンネル 0 の下限値。

チャンネル 1 の上限

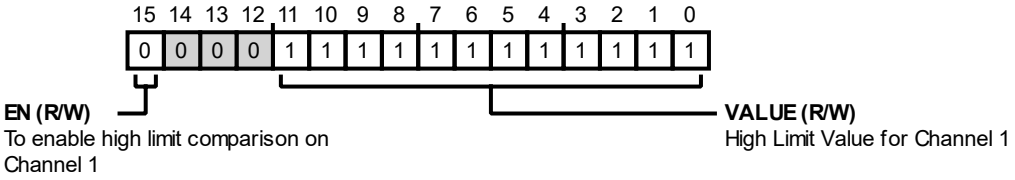


図 20-41：ADC_LIM1_HI レジスタ図

表 20-28：ADC_LIM1_HI レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W)	EN	チャンネル 1 での上限比較を有効化。
11:0 (R/W)	VALUE	チャンネル 1 の上限値。

チャンネル 1 の下限

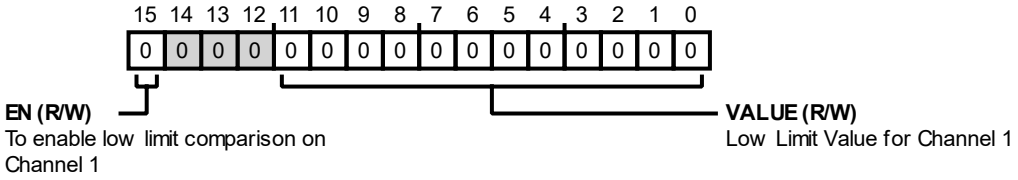


図 20-42：ADC_LIM1_LO レジスタ図

表 20-29：ADC_LIM1_LO レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W)	EN	チャンネル 1 での下限比較を有効化。
11:0 (R/W)	VALUE	チャンネル 1 の下限値。

チャンネル 2 の上限

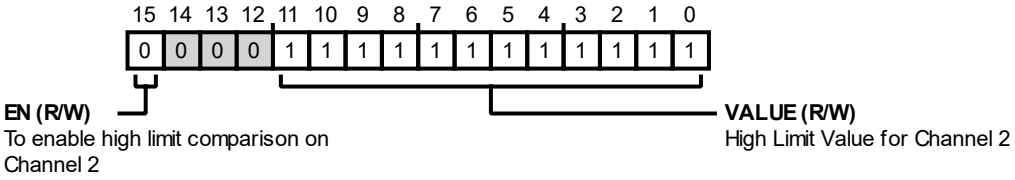


図 20-43：ADC_LIM2_HI レジスタ図

表 20-30：ADC_LIM2_HI レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W)	EN	チャンネル 2 での上限比較を有効化。
11:0 (R/W)	VALUE	チャンネル 2 の上限値。

チャンネル 2 の下限

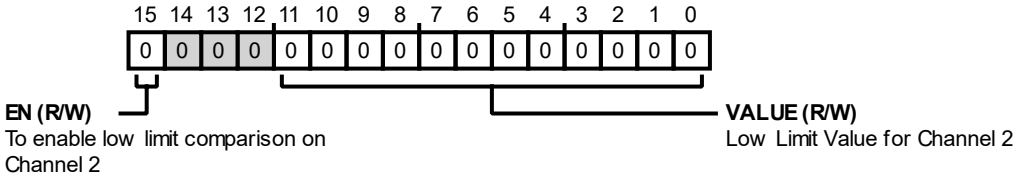


図 20-44：ADC_LIM2_LO レジスタ図

表 20-31：ADC_LIM2_LO レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W)	EN	チャンネル 2 での下限比較を有効化。
11:0 (R/W)	VALUE	チャンネル 2 の下限値。

チャンネル 3 の上限

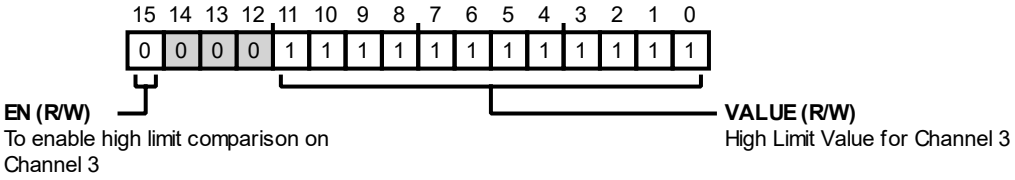


図 20-45：ADC_LIM3_HI レジスタ図

表 20-32：ADC_LIM3_HI レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W)	EN	チャンネル 3 での上限比較を有効化。
11:0 (R/W)	VALUE	チャンネル 3 の上限値。

チャンネル 3 の下限

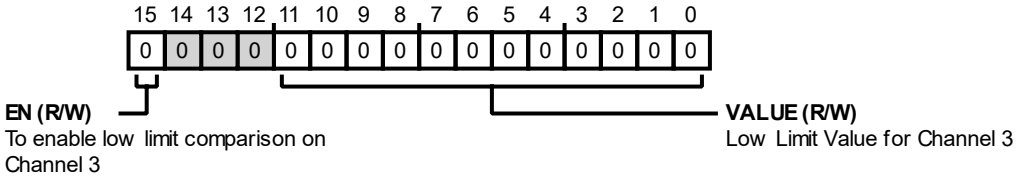


図 20-46：ADC_LIM3_LO レジスタ図

表 20-33：ADC_LIM3_LO レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W)	EN	チャンネル 3 での下限比較を有効化。
11:0 (R/W)	VALUE	チャンネル 3 の下限値。

出力レジスタのオーバーフロー

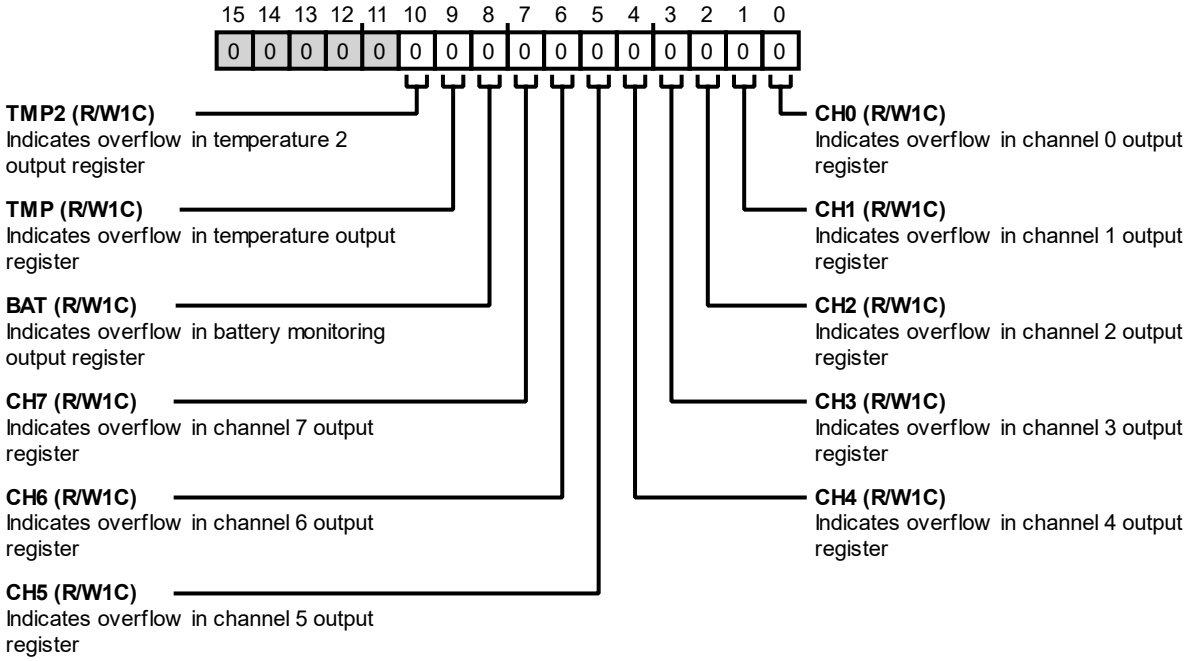


図 20-47：ADC_OVF レジスタ図

表 20-34：ADC_OVF レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
10 (R/W1C)	TMP2	温度 2 出力レジスタでのオーバーフローを示します。
9 (R/W1C)	TMP	温度出力レジスタでのオーバーフローを示します。
8 (R/W1C)	BAT	バッテリ監視出力レジスタでのオーバーフローを示します。
7 (R/W1C)	CH7	チャンネル 7 出力レジスタでのオーバーフローを示します。
6 (R/W1C)	CH6	チャンネル 6 出力レジスタでのオーバーフローを示します。
5 (R/W1C)	CH5	チャンネル 5 出力レジスタでのオーバーフローを示します。
4 (R/W1C)	CH4	チャンネル 4 出力レジスタでのオーバーフローを示します。

表 20-34：ADC_OVF レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
3 (R/W1C)	CH3	チャンネル 3 出力レジスタでのオーバーフローを示します。
2 (R/W1C)	CH2	チャンネル 2 出力レジスタでのオーバーフローを示します。
1 (R/W1C)	CH1	チャンネル 1 出力レジスタでのオーバーフローを示します。
0 (R/W1C)	CH0	チャンネル 0 出力レジスタでのオーバーフローを示します。

ADC パワーアップ時間

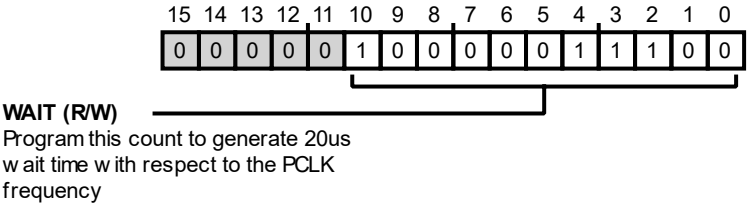


図 20-48：ADC_PWRUP レジスタ図

表 20-35：ADC_PWRUP レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
10:0 (R/W)	WAIT	PCLK 周波数を基準として 20us の待機時間を生成するには、このカウントを設定します。

ADC ステータス

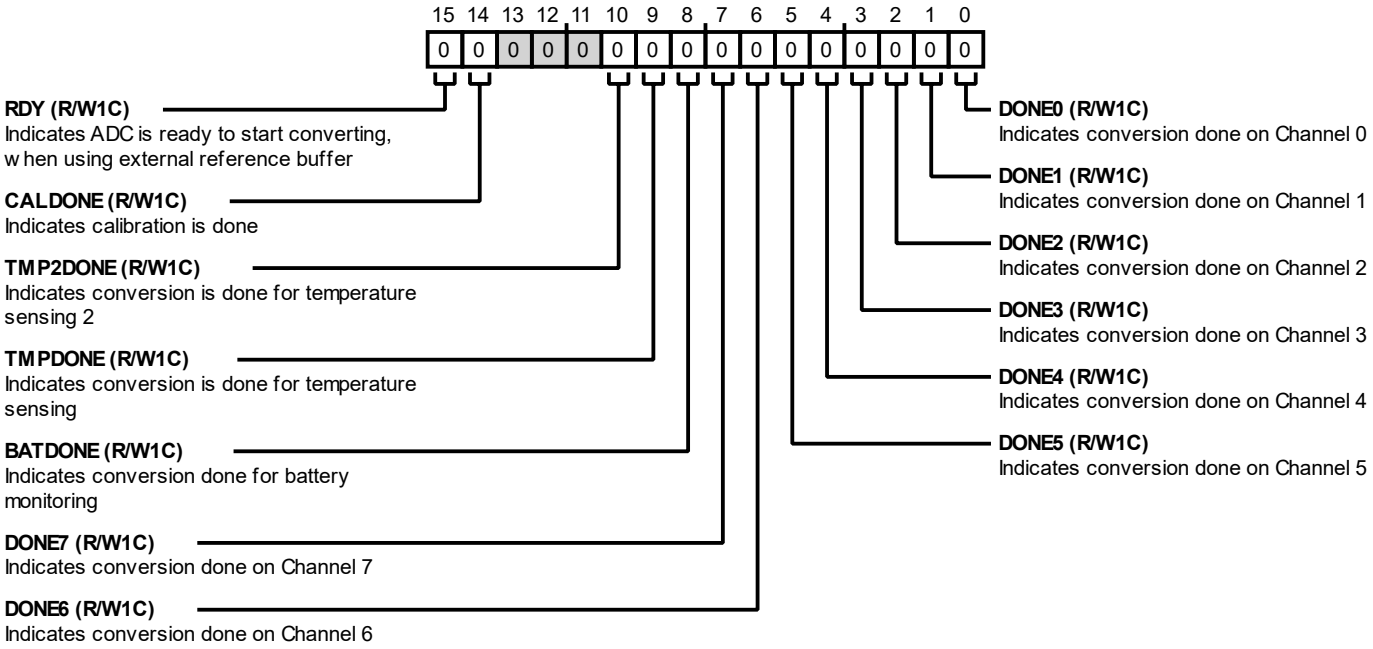


図 20-49：ADC_STAT レジスタ図

表 20-36：ADC_STAT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15 (R/W1C)	RDY	外部リファレンス・バッファの使用時、ADC が変換を開始する準備ができていることを示します。
14 (R/W1C)	CALDONE	キャリブレーションが完了したことを示します。
10 (R/W1C)	TMP2DONE	温度検出 2 の変換が完了したことを示します。
9 (R/W1C)	TMPDONE	温度検出の変換が完了したことを示します。
8 (R/W1C)	BATDONE	バッテリ監視の変換が完了したことを示します。
7 (R/W1C)	DONE7	チャンネル 7 での変換が完了したことを示します。
6 (R/W1C)	DONE6	チャンネル 6 での変換が完了したことを示します。

表 20-36：ADC_STAT レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
5 (R/W1C)	DONE5	チャンネル 5 での変換が完了したことを示します。
4 (R/W1C)	DONE4	チャンネル 4 での変換が完了したことを示します。
3 (R/W1C)	DONE3	チャンネル 3 での変換が完了したことを示します。
2 (R/W1C)	DONE2	チャンネル 2 での変換が完了したことを示します。
1 (R/W1C)	DONE1	チャンネル 1 での変換が完了したことを示します。
0 (R/W1C)	DONE0	チャンネル 0 での変換が完了したことを示します。

温度結果 2

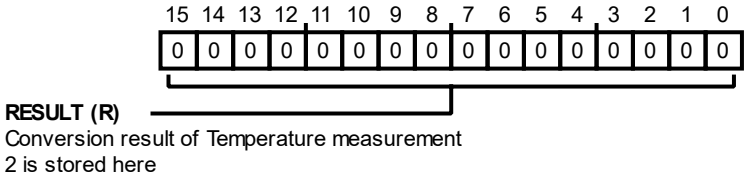


図 20-50：ADC_TMP2_OUT レジスタ図

表 20-37：ADC_TMP2_OUT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	RESULT	温度測定 2 の変換結果がここに保存されます。

温度結果

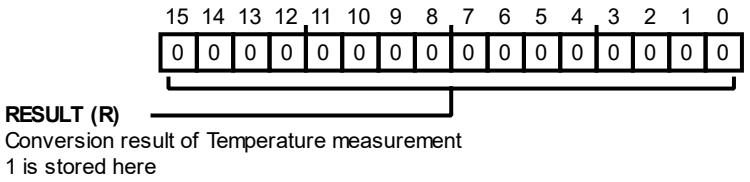


図 20-51：ADC_TMP_OUT レジスタの図

表 20-38：ADC_TMP_OUT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	RESULT	温度測定 1 の変換結果がここに保存されます。

21 リアルタイム・クロック（RTC）

ADuCM4050 MCU は、経過時間を追跡するリアルタイム・クロック（RTC）を使用しています。時間の単位は設定が可能で、32,768Hz の外付け水晶発振器を使用して時間ベースを生成します。RTC をイネーブルすると、CPU が最後に再定義（再初期化）したカウント値またはパワーアップ値のどちらか該当するほうからの経過時間が積算されます。RTC は設定された任意の初期値からのカウントが可能で、RTC カウントの繰り越しもサポートされています。RTC は 16 ビット・ペリフェラルですが、経過時間カウントやアラーム値などの 47 ビット値用に 3 つのレジスタを内蔵しています。

ADuCM4050 MCU には、RTC0 と RTC1（FLEX_RTC とも呼ばれる）の 2 つのリアルタイム・クロック・ブロックがあります。

各 RTC は設定可能な 2 つのアラーム機能を備えており、絶対（時間の正確な一致による）アラームまたは周期的（RTC カウントが 60 インクリメントごとの）アラームを生成することができます。

RTC のイネーブルと設定、およびそのカウント値の解釈と時刻への変換は、ソフトウェアで行います。RTC ロジックは、時間追跡に関してより高い PPM 精度を実現できるように補正された、デジタル・トリム機能も備えています。

RTC 機能

ADuCM4050 MCU は、以下の機能をサポートしています。

- RTC カウント・レジスタ。プログラマブルな基準点（初期値）からの経過時間を表す、32 個の整数ビットと 15 個の小数ビットを備えており、時間単位は設定可能です。このレジスタはソフトウェア制御の下でプログラムされます。
  - RTC は、RTC ベース・クロック（公称 32,768Hz）を分周した周期を単位として時間をカウント可能で、分周は 2 の 0 乗から 15 乗までの任意のべき乗で行うことができます。したがって、RTC の時間単位の範囲は、30.52µs～1s です。

    RTC カウントをプログラム（初期化）するときや再度有効にするとき、あるいはプリスケール分周比を変更するときは、プリスケーラが自動的にゼロに設定されます。これは、RTC カウント値を、プリスケール・シーケンスの開始およびモジュロ 60 のカウント繰り越しの両方のタイミングと、正確に一致させるためです。

    実際的な言い方をすると、1 秒の時間ベースを使用する際に、CPU による RTC カウント（経過時間）の再定義を、1 秒の境界と 1 分の境界が一致する位置に合わせることができます。この RTC では、どの値にプリスケールされた時間ベースに対しても同じ機能がサポートされています。

- 2 つの独立したアラーム機能（1 つは絶対時間、他方はモジュロ 60 の周期時間による）。この機能はオプションで有効にして、RTC カウントがアラーム値と等しくなった時点でプロセッサに割込みをかけることができます。プロセッサがスリープ状態にある場合、このような割込みはそのプロセッサをウェイクアップします。
- デジタル・トリム機能。RTC カウントに固定間隔で正または負の調整を加え（設定された RTC 時間単位を使用）、RTC PPM 時間精度を目標の範囲内に維持することができます。調整値と間隔値は、共に CPU のキャリブレーション・アルゴリズムを実行することによって計算されます。
- RTC は、CPU から要求があった場合、経過したリアルタイム・カウントのスナップショットを取得して保存することができます。CPU はこれにより、データ・パケット受信などのイベントにタイム・スタンプを対応させることができます。RTC は、CPU によるリードバックのためにスナップショットを保存します。スナップショットはそのまま維持され、CPU が新しい値のキャプチャを要求した場合のみ上書きされます。
- RTC は、アラーム機能と共に 4 つの独立した SensorStrobe チャンネルを備えています。各チャンネルは GPIO 経由で外部デバイスに出力パルスを送り、特定の時点で測定または何らかの動作を実行するよう、そのデバイスに命令します。SensorStrobe イベントは ADuCM4050 MCU の CPU によってスケジュールされます。CPU は、RTC のリアルタイム・カウントを基準とする特定のターゲット時間に SensorStrobe イベントをアクティブにするよう RTC に命令することによって、これを行います。出力パルスのデューティ・サイクルと周波数（0.5Hz～16384Hz）は設定可能です。
- 各 SensorStrobe チャンネルは外部デバイスからの GPIO 入力を最大 3 つ使用可能で、これらの入力は、外部デバイスに送信される出力パルスに基づいてサンプリングすることができます。外部デバイスからこれらの GPIO 入力への特定アクティビティが検出された場合に ADuCM4050 MCU に割込みをするよう、各 SensorStrobe チャンネルを設定できます。これらの入力は、FIFO フル、スイッチ・オープン、閾値超過などのセンサー状態を送信できます。MCU はこの機能によって低消費電力状態を維持し、特定アクティビティの検出時にのみウェイクアップしてデータを処理できるようになります。
- 入力サンプリングは RTC のオプション機能で、SensorStrobe イベントの立上がり／立下がりエッジでサンプリングを行うように専用 GPIO を設定することができます。
- RTC は入力キャプチャ機能を備えています。入力キャプチャは、外部デバイスが ADuCM4050 MCU の GPIO 入力の 1 つの状態遷移を介してイベントを送信した場合に、RTC のリアルタイム・カウントのスナップショットを取得するプロセスです。

RTC 機能の説明

ここでは、ADuCM4050 MCU の RTC 機能について説明します。

RTC のブロック図

RTC の概略ブロック図を下に示します。カウント、アラーム、トリム、スナップショット、およびウェイクアップ割込みのためのすべての機能は、32kHz クロックを使用する常時オンの専用 RTC 電源領域に置かれています。ポストされたレジスタ書込みのためのキューイングおよびディスパッチ・ロジックで構成される CPU との APB インターフェースは、Cortex NVIC への割込みと共に、すべてメイン・パワーゲート・コア領域の PCLK/FCLK クロック（同期クロック）セクションに置かれています。

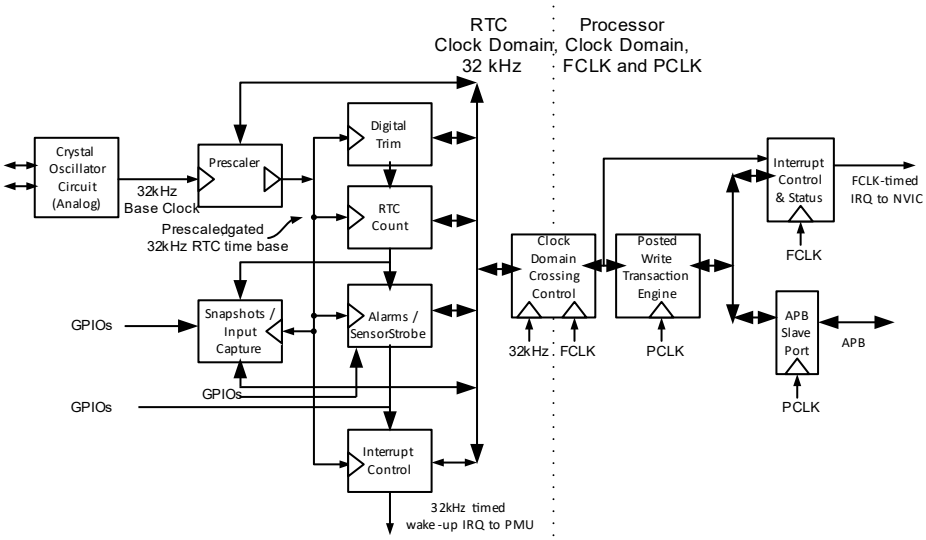


図 21-1：RTC の概略ブロック図

RTC の主な用途は、計時機能を提供して、正確かつ信頼できる方法と最小限の消費電力で時刻と日付を維持することにあります。計時機能に加えて、ストップウォッチ機能とアラーム機能も備えています。RTC は内部カウンタを使用して、日にち、時間、分、秒を単位として日付と時刻を維持します。これは、ユーザ・アプリケーションが RTC から日付と時刻の情報を引き出すのに十分なデータです。また、周期的に割込みを行うこともできます。

プリスケーラにより、RTC は 1 秒未満の時間単位でカウントできます（RTC1 のみに適用)。プリスケールが設定されている場合（RTC_CR1.PRESCALE2EXP)、整数カウント・レジスタ RTC_CNT0 および RTC_CNT1 は、RTC_CR1.PRESCALE2EXP で選択される 2 のべき乗でベース・クロック（32768Hz）を分周したレートで時間単位をカウントし、小数カウント・レジスタ RTC_CNT2 は、RTC_CR1.PRESCALE2EXP で定義された小数ビットの数を格納します。RTC_CNT2 レジスタは、RTC_CNT0 内のカウントをインクリメントするために、32768Hz のベース・クロック周波数で 0 から RTC_CR1.PRESCALE2EXP で選択された 2 のべき乗まで順番にカウントします。よって、RTC カウントは、プリスケールされた時間単位で表示され、{RTC_CNT1, RTC_CNT0}小数点{RTC_CNT2}で与えられます。したがって、RTC_CNT2 の小数ビットを含むリアルタイム・カウントの全体的な分解能は、32kHz クロックの 1 周期です。

RTC の動作モード

ADuCM4050 MCU が使用する RTC は、以下の機能をサポートしています。

初期 RTC パワーアップ

RTC は専用の電圧領域で動作し、この領域は通常状態（設定可能）では常時電源オンになっています。ただし最初のバッテリ取り付け時または交換時は、パワーオン・リセットが行われてすべての RTC レジスタがリセットされます。

RTC 故障が検出されると、通常、CPU は RTC カウント・レジスタとデジタル・トリム・レジスタを再プログラムして、RTC コントロール・レジスタ内の故障フラグをクリアします。CPU はオプションで、アラーム値とカウント値が一致すると割込みを生成するように、RTC のアラーム・レジスタをプログラムすることができます。

持続的なスティッキーRTC ウェイクアップ・イベント

パワーダウン・モードのデバイスに発生した RTC アラーム・イベントが失われることはありません。割込みがイネーブルされている場合、アラームによって生成される割込みは、アサートされた状態で RTC により維持されるので、後でプロセッサ電源が回復された時点で、NVIC がその割込み（FCLK で時間情報が付与されたバージョン）を認識します。これを容易にするために、RTC は同じ割込みに 32Khz クロックで時間情報を付与したバージョンを PMU のウェイクアップ・コントローラに送信し、これによってデジタル・コアに再度電源が供給されます。CPU がウェイクアップすると、CPU は PMU と RTC の両方を確認して、ウェイクアップに関する割込みイベントの原因を知ることができます。

CPU がポストした書込みへの RTC の対応能力

CPU により RTC の 32kHz ソースの MMR にポストされた書込みが、32kHz 領域に対する他の同様のレジスタ書込みのキューによる（RTC 内の）保留ディスパッチである場合は、CPU による同じ RTC レジスタへの 2 番目の書込みやそれ以降の書込みをスタック・アップしたり、保留トランザクションを上書きしたりすることはできません。このような試みは RTC によって拒否され、RTC 内で `RTC_SR0.WPNDERRINT` 割込みイベントが生成されます（`RTC_SR0` の MMR の詳細を参照）。

パケット定義の時間基準に対する RTC カウントのリアライメント

CPU は、`RTC_GWY` MMR にソフトウェア・キー0x7627 を書き込むことによって、経過時間カウントのスナップショットを取得するよう RTC に命令することができます。この場合、組み合わされた 3 つのスナップ・レジスタ（`RTC_SNAP2`、`RTC_SNAP1`、および `RTC_SNAP0`）が、3 つのカウント・レジスタ（`RTC_CNT2`、`RTC_CNT1`、および `RTC_CNT0`）の現在値に更新され、後で CPU が上書きを命じるまでそのスナップショットが維持されます。

RTC に関する推奨事項：クロックと消費電力

RTC の使用に関しては、以下の事項を推奨します。

PCLK 周波数

RTC 書込みの進行中は、PCLK 周波数を 1MHz 未満に落とさないでください。RTC クロックは、PCLK の 1/30 以下の周波数にする必要があります。

PCLK の停止

CPU を PCLK が停止するようなモードにする場合は、過去にポストされた書込みがすべて完了しているという確認が RTC から得られるまで、待機する必要があります。CPU は、`RTC_SR0` レジスタと `RTC_SR2` レジスタの両方を読み出すことによって、これをチェックできます。

パワーダウン時に RTC 電源境界越しの通信がないことを確認

パワーダウンに関する情報を CPU が事前に得ている場合は、RTC の常時電源オン部分の動作に問題がないことを確認するために、以下の措置を講じる必要があります。

- すなわち、
  CPU は、実行待ちの書込みが RTC 内にポストされていないことを確認する必要があります。
- または、

RTC 内にポストされた書込みのうち、キューに入っているものと実行中のものをキャンセルします。これは、RTC_GWY レジスタにキャンセル・キー0xA2C5 を書き込むことによって行います。この書込みは直ちに有効になります。

- および、
  コアの電源が切れるまで、RTC に追加のレジスタ書込みをポストしないでください。

これらのステップによって、CPU と RTC の間で進行中の通信がないこと、したがってその後 RTC 電源領域の絶縁障壁がアクティブになったときに、通信が中断されるおそれもないことを徹底できます。

RTC 割込みと例外

RTC ブロックは複数のソースからの割込みを生成可能で、これらの割込みのマスクは、コントロール・レジスタをプログラムすることによって解除できます。割込みソースはステータス・レジスタに反映されます。

RTC デジタル・トリミング

RTC ブロックは、RTC を補償するための LF 水晶発振器のトリム値の計算に使用できます。これは、静的測定（周波数カウンタ）であるリアルタイムのドリフト測定をベースとする外部リファレンスを使用して求めることができます。

通常、市販の水晶発振器は 20～100ppm で動作します。一例として、未トリム時には約+58.6ppm（0.00586%）の誤差が生じる特別な水晶発振器とボードからなる構成をトリミングする場合を考えます。これは、1 週間に約 35.5 秒（1 年で 30 分）進むクロックに相当します。

表 21-1：トリム補正

	Trim Interval (second)				
Trim Value (second)	16384	32768	65536	131072	
0	0.00	0.00	0.00	0.00	Trim Correction (ppm)
1	61.04	30.52	15.26	7.63	
2	122.07	61.04	30.52	15.26	
3	183.11	91.55	45.78	22.89	
4	244.14	122.07	61.04	30.52	
5	305.18	152.59	76.29	38.15	
6	366.21	183.11	91.55	45.78	
7	427.25	213.62	106.81	53.41	

トリム補正の表によれば、上記の例に最も近い ppm 修正は 61.04 です。同じ修正結果が得られる異なる組み合わせがある場合は、瞬時ドリフトが最小限に抑えられるので、最も短い時間間隔のもの（結果としてトリム値も最小となるもの）が望ましい選択です。

この例では 2^14 秒のトリム間隔、負のトリム値 1 秒を選択して 4.5 時間ごとに 1 秒を引き、進んでいる水晶発振器を「遅らせ」て、より妥当な速度にします。この特別なトリムによって残る誤差はマイナス 2.44ppm（0.000244%）で、トリム後のクロックはわずかに遅くなりますが（週に 1.5 秒未満、1 年で約 1.3 分）、トリムしない場合の精度 30 分／年よりははるかに正確です。

RTC プログラミング・モデル

以下のセクションでは、アラーム・イベント用に RTC を設定するためのプログラミング・シーケンスを示します。

プログラミングのガイドライン

プログラミング・ガイドラインを以下に示します。

1. RTC_CNT レジスタをリセットして 0 にします。
2. RTC_CR1 レジスタで、RTC ベース・クロックを分周するようにプリスケーラを設定します。
3. MMR 書込みは遅い方の RTC 領域で発生するので、RTC_SR1 レジスタで RTC 同期ビットの設定をポーリングします。
4. 使用する予定のアラーム時間を使って、RTC_ALM0、RTC_ALM1、RTC_ALM2 レジスタをプログラムします。
5. RTC_CR0.ALMINTEN ビットをセットすることによって、アラームの割込みをイネーブルします。
6. RTC_CR0.ALMEN ビットと RTC_CR0.CNTEN ビットをセットします。
7. RTC_CNT が RTC_ALM の値と一致したときにトリガされる RTC アラーム割込みを待ちます。

RTC SensorStrobe

SensorStrobe は RTC 内のアラーム機構です。これは GPIO 経由で外部デバイスに出力パルスを送り、特定の時点で測定または何らかの動作を実行するようそのデバイスに命令します。SensorStrobe イベントは ADuCM4050 MCU の CPU によってスケジュールされます。CPU は、RTC のリアルタイム・カウントを基準とする特定のターゲット時間に SensorStrobe イベントをアクティブにするよう RTC に命令することによって、これを行います。

アラーム時間は CPU によって RTC にプログラムされますが、通常これは休止状態になる前に行われます。SensorStrobe イベントが発生すると、RTC はオプションで CPU をウェイクアップさせて、GPIO を介して得られるセンサーの結果を確認するために割込みをかけることができます。

外部デバイスを特定の時間間隔でモニタする場合は、入力サンプリング機能が使われます。この周期は SensorStrobe チャンネル出力によって制御されます。外部デバイスからの入力は専用の GPIO 入力を通じ、定義されたレートでサンプリングされます。これらの GPIO 入力を介して外部デバイスからの特定のプログラム・シーケンスが検出されると、RTC はオプションで CPU をウェイクアップし、割込みをかけて受信データを処理させることができます。

ADuCM4050 MCU の RTC1 には 5 つの SensorStrobe チャンネル、つまり SS0（アラームのみ）、SS1、SS2、SS3、および SS4（アラームのみ）があります。これらのチャンネルは、CPU により RTC にスケジュールされたアラームとして動作します。SensorStrobe イベント（アラーム）が発生すると、チャンネルが出力制御ラインをトグルします。この出力は関係するチャンネル用の固定 GPIO ピンを通じて転送され、チップからエクスポートされます。

現時点で ADuCM4050 MCU がサポートしている GPIO は、SS1、SS2、および SS3 です。SS0 は RTC の 47 ビット・アラーム機能、SS4 は 16 ビット・アラーム機能で、GPIO には接続されていません。

表 21-2：SensorStrobe 出力 GPIO

SensorStrobe Channel	MCU GPIO Pin
SS1	P2_11 (GPIO43)
SS2	P1_12 (GPIO28)
SS3	P2_08 (GPIO40)

RTC0 には SensorStrobe 機能はありません。

図は、SensorStrobe チャンネルと、それらをどのようにメイン RTC カウントにアラインするかを示しています。

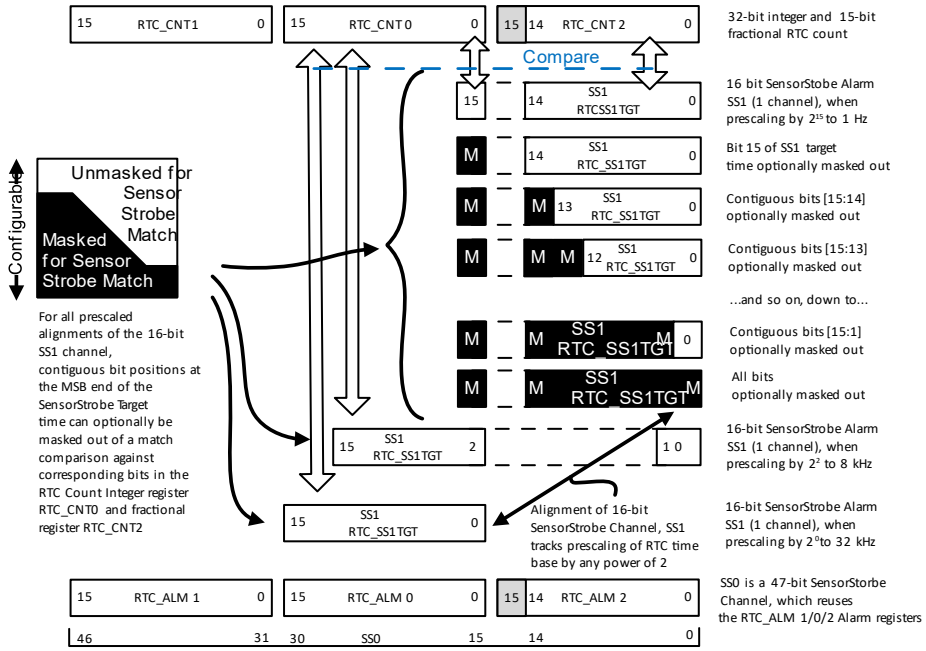


図 21-2：RTC SensorStrobe チャンネル

47 ビット SensorStrobe チャンネル（SS0）は、`RTC_ALM1`、`RTC_ALM0`、および `RTC_ALM2` レジスタによって定義される RTC の 47 ビット・アラーム機能と同義です。これは絶対時間アラームで、任意のビット位置でマスク解除することができ、アラームが繰り返されるまでの時間幅は RTC カウントの長さと同じです。

SensorStrobe を頻繁にアクティブにするために SS0 を使用するには、CPU が、SS0 イベント発生時に RTC によってウェイクアップされ、割込みをかけられるたびに、`RTC_ALM1`/`RTC_ALM0`/`RTC_ALM2` レジスタでアラーム時間をプログラムし直す必要があります。SS0 で GPIO を使用することはできません。

16 ビット SensorStrobe チャンネルの SS1、SS2、SS3、および SS4 は時間幅は短いものの、SS0 より高い汎用性を備えています。これらは、RTC の整数カウント時間ベースのプリスケーリング用に、16 個の使用可能ビットをアラインします。

RTC_SSMSK レジスタは、繰返しアラームの周期を短縮するために、そのターゲット時間の連続 MSB ビットをマスクします（**RTC SensorStrobe チャンネル**の図を参照）。RTC_SSMSKOT レジスタも、パルス幅を制御するために、そのターゲット時間の連続 LSB ビットをマスクします。16 ビットの SensorStrobe チャンネル SS1、SS2、SS3、SS4 は、アラーム周期と 32kHz クロック周期の関係が 2 のべき乗でない場合のために、オプションで自動再ローディングをサポートしています。各 SensorStrobe チャンネルには、その SensorStrobe ラインの 1 サイクル・アクティブ化をエクスポートするための固定された GPIO があります。

RTC SensorStrobe チャンネルの図には、RTC 時間ベースの 3 つのプリスケーリング度の例（2^{15}、2^{2}、および 2^{0}）が示されています。この図には、SS1、SS2、SS3、SS4 の 16 ビット・アラーム時間を 32kHz クロックのプリスケーリング度にアラインする方法が示されています。

SensorStrobe の一致をチェックする場合は、設定可能なプリスケーリングと一致する RTC_CNT2 レジスタのすべての小数ビットが、SS1、SS2、SS3、SS4 によって指定されるアラーム時間の対応ビット位置と比較されます。

残りのビットは、RTC_CNT0 レジスタの LSB 側と比較されます。プリスケーリングに基づくこの SS1、SS2、SS3、SS4 と RTC_CNT2 および RTC_CNT0 レジスタの自動アラインは、2 の 0 乗から 15 乗までのすべてのプリスケールに対してサポートされています。

2 種類のアラーム（SensorStrobe イベント）周期に対する SensorStrobe：

- **32kHz クロックとの関係が 2 のべき乗になっている周期**：比較は、RTC カウントの SSx ビット（x = 1、2、3、4）の値の間で繰り返し行われ、プリスケーリングに合わせてアラインされます。

  周期 =（（RTC_CR4SS.SSxMSKEN ?2$^{\text{RTC\_SSMSK.SSxMSK}}$ *：2*^{16}*））/ 32768 秒*

  オン時間 = 2$^{\text{SSMSKOT.SSxMSKOT}}$ */ 32768 秒*

- **2 のべき乗でない周期**：比較は、SensorStrobe イベントが発生するたびに RTC_SSxHIGHDUR および RTC_SSxLOWDUR（x = 1、2、3）内のオフセット値を SSx（x = 1、2、3、4）に自動リロードすることによって行われます。（SSx 内の既存の内容に累積的に加算）

  周期 =（RTC_SSxLOWDUR + RTC_SSxHIGHDUR）/ 32768 秒

  オン時間 = RTC_SSxHIGHDUR / 32768 秒

SensorStrobe チャンネルの次回イベントのターゲット時間は、CPU が RTC_SSxTGT レジスタを介して読み出すことができます。自動リロードを使用しない SensorStrobe では、イベント間のステップ・サイズが SSx によって決まるので、RTC_SSxTGT レジスタ内のライブ値が SSx（x = 1、2、3、4）と同じになります。自動リロードを使用する場合、RTC_SSxTGT レジスタのライブ値は、繰り返し累積的に加算される RTC_SSxHIGHDUR と RTC_SSxLOWDUR のモジュロ 16 の結果を SSx の開始値に反映します。RTC_SSxTGT を使用すれば、CPU は、RTC_SSxTGT と RTC カウントの一致によって SensorStrobe イベントの期限を知ることができます。

すべての SensorStrobe チャンネルは、ターゲットの MSB 側と LSB 側の両方で部分的にマスクすることができます。マスキングを行えば、SensorStrobe イベントの反復周期を短縮できます。これは、ターゲット時間と RTC カウントのビット位置をドント・ケアと見なし、LSB マスキングの場合は連続マッチングをドント・ケアと見なすことにより可能です。RTC カウントに対して SSx の 16 ビット SensorStrobe マッチ・チェックを行う場合は、SSx と RTC_SSxTGT の MSB 側にある連続ビットをオプションでマスクして、ドント・ケアとすることができます（図では M ビット位置として示されています）。SSx および RTC_SS1TGT 内のマスクされる連続ビットの数は、RTC_SSMSK と RTC_SSMSKOT

によって指定されます。それぞれの SensorStrobe チャンネルには専用の GPIO 出力があり、これを介してチップ外部に出力パルスがエクスポートされます。

RTC SensorStrobe プログラミング・モデル

1. SensorStrobe 用 GPIO を、**SensorStrobe 出力 GPIO** の表に従って設定します。
2. `RTC1_CNT` レジスタを 0 にリセットします。
3. `RTC1_SSxHIGHDUR` レジスタと `RTC1_SSxLOWDUR` レジスタの値を、SensorStrobe パルスのハイ時間とロー時間に設定します。これらのレジスタへの書込みは、SensorStrobe 出力パルスのデューティ・サイクルも設定します。ここで、x は SensorStrobe のチャンネル番号です。
4. `RTC1_CR4SS.SSxARLEN` に 1 を書き込むことによって、自動リロードを有効にします。ここで、x は SensorStrobe のチャンネル番号です。
5. `RTC1_CR3SS.SSxEN` に 1 を書き込むことによって、SensorStrobe チャンネルをイネーブルします。ここで、x は SensorStrobe のチャンネル番号です。
6. `RTC1_CR0.CNTEN` に 1 を書き込むことによって、RTC をイネーブルします。これにより、RTC カウントと選択したチャンネルの SensorStrobe 出力がイネーブルされます。

RTC 入力サンプリング

これは SensorStrobe チャンネルに関連付けられたオプション機能です。各 SensorStrobe チャンネル（`SSx`、x = 1、2）には最大 3 個の GPIO 入力があります（`SSxGPINy`、y = 0、1、2）。これらの入力は、出力キャプチャ・イベントに基づいてサンプリングされます。

入力のサンプリングには、立下がりエッジ、立上がりエッジ、またはその両方の SensorStrobe イベントを使用することができます。各チャンネルのサンプリング入力は、`RTC_SR7.SSxSMP` フィールドの非スティッキー・リードバック・ステータスとして使用できます（次のサンプルを使用するときに更新）。これは、センサーにより生成される外部アクティビティの周期的かつ管理されたサンプリングを可能にします。それぞれの出力パルスには、6 個のサンプリング・エッジを使用することができます。

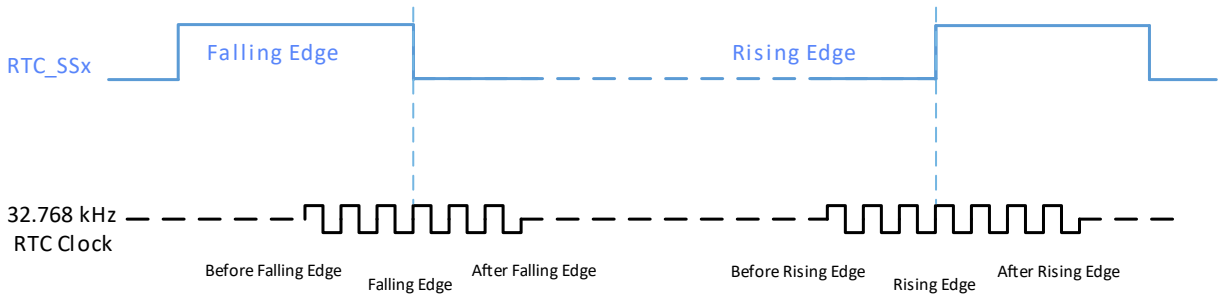


図 21-3：サンプリング・エッジ

センサー・アクティビティ・モニタリングは、最新サンプルが特定の動作と一致したときにオプションの割込みを生成することによって行われます。これには、以下のいずれかのシーケンスを選ぶことができます。

- 最新サンプル（$SMPn_t$）がその直前のサンプル値（$SMPn_{t-1}$）と異なる。または、
- 最新サンプル（$SMPn_t$）がその直前のサンプル値（$SMPn_{t-1}$）と一致する。または、
- 最新サンプル（$SMPn_t$）が予想サンプル値（EXP_SMPn）と一致する。または、
- 最新サンプル（$SMPn_t$）が予想サンプル値（EXP_SMPn）と一致しない。

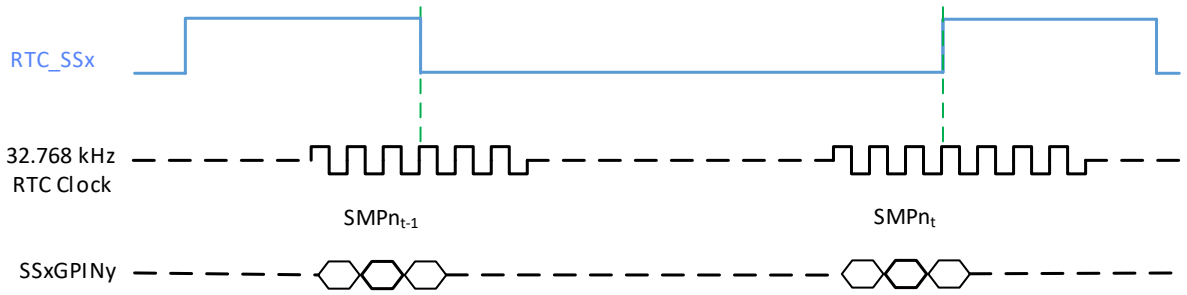


図 21-4：RTC 入力サンプリング

RTC_SR7 レジスタのスティッキー・ステータス・ビットを使用できるということは、プログラムされたシーケンスがセンサー・インターフェースで発生したことを意味します。このステータス・ビットは、オプションの割込みとして CPU にエクスポートできます。これは、CPU を低消費電力モードにしておいて、プログラムされたシーケンスがセンサー・インターフェースで発生したときだけウェイクアップすることを可能にします。

各 SensorStrobe チャンネル（SS1、SS2、SS3、SS4）は、RTC_GPMUX0 レジスタと RTC_GPMUX1 レジスタを設定することによって、使用可能な 8 つの専用 GPIO の中から最大 3 つの GPIO 入力を選択できます。選択した GPIO は、外部アクティビティをモニタするための入力として設定する必要があります。

GPMUX0/1.SSxGPINySEL	3'b000	3'b001	3'b010	3'b011	3'b100	3'b101	3'b110	3'b111
SSxGPINy	P0_12	P2_00	P0_09	P0_08	P1_13	P1_02	P2_07	P2_09

ADuCM4050 MCU で休止状態にあるリアルタイム・クロック（RTC1）の SensorStrobe 機能を以下に示します。

- GPIO がアクティブな状態で、4 つの SensorStrobe チャンネル（SS1、SS2、SS3、SS4）をサポートします。
  - 16 ビット・チャンネル。16 ビット・チャンネルのターゲット時間は以下のように指定されます。

    {least_significant_integer_bits_to_bring_total_to_16, fractional_bits_due_to_prescaling}

    この連結されたキャプチャ時間の合計ビット数は 16 です。

    ターゲット内の小数ビットの数は、32kHz ベース・クロックに対して設定されたプリスケーリングの程度によって異なり、入力キャプチャ機能に使用する小数ビットの数もプリスケーリングに従います。16 ビットのターゲット時間の残りビットは、RTC の整数カウントの LSB 側にある整数で構成されます。小数ビットと整数ビットの（プリスケーリングによる）合計数は、16 でなければなりません。
  - RTC には 47 ビットのアラーム機能があります。しかし、この機能に関連付けられて、SensorStrobe チャンネルとして動作する GPIO はありません。この 47 ビット・アラームのターゲット時間は以下のように指定されます。

{32_integer_bits, 15_fractional_bits}

- SensorStrobe チャンネルの場合、ターゲット時間には関連するすべての小数ビットが含まれるので、ターゲット時間は常に個々の 32kHz クロック・サイクルまでの精度で指定されます。
- 16 ビット SensorStrobe チャンネル、SS1、SS2、SS3、SS4 の場合は、16 ビット・ターゲット時間内の連続ビットを、チャンネルごとにターゲットの MSB 側からサーモメータ・コードでマスクすることができます。

  このオプション・マスキングの結果として、ターゲット時間の MSB 側にある連続ビットはドント・ケアとして扱われ、更にこのために、マスクされたすべてのビットで SensorStrobe イベントの反復周期が 1/2 になります。

  つまり、ターゲット時間の MSB 側ビットのマスク数を増やすと、そのチャンネルの反復 SensorStrobe イベントのモジュラ・シーケンスが 1/2 に短縮されます。
- 16 ビット SensorStrobe チャンネルの場合は、16 ビット・ターゲット時間内の連続ビットを、チャンネルごとにターゲットの LSB 側からサーモメータ・コードでマスクすることができます。このオプション・マスキングの結果として、ターゲット時間の LSB 側にある連続ビットはドント・ケアとして扱われ、これによって連続マッチとなるので、最終的には出力パルスのハイ時間を制御する結果となります。SensorStrobe チャンネルのハイ時間は、LSB 側のビットをマスクするごとに 2 倍になります。

  つまり、ターゲット時間の LSB 側ビットのマスク数を増やすと、そのチャンネルの SensorStrobe イベントのハイ時間が 2 倍になります。
- 自動リロード：イベントがトリガされるごとに設定可能な 16 ビットのデルタ（リロード値）がターゲット時間に加算されて、次のイベントの修正ターゲットが計算されます。リロード値は、ロー時間レジスタとハイ時間レジスタから導かれます。立下がり／立上がりエッジ・イベントによって出力パルスがローになると、ロー／ハイ時間の値がリロード値として使われ、立上がり／立下がりイベントに対して次のターゲットが設定されます。これは、32kHz クロック周期との関係が 2 のべき乗ではない反復イベントの周期性とハイ時間を、このチャンネルで生成することを可能にします。

  CPU は、リロードされたターゲット時間を読み込んで、その SensorStrobe チャンネルのイベントごとにリローディングによる積算が行われているのを確認することができます。リロードされたターゲットも、MSB 側から連続的にマスクできます。
- SensorStrobe チャンネルには読出し可能なスティッキー割込みソース・ビットがあり、SensorStrobe イベントが発生すると、常にこれがアクティブになります（ハイに固定される）。立下がりエッジ・イベントと立上がりエッジ・イベントには、別々のビットを使用できます。この割込みソース・ビットは、割込みをかけて RTC からのラインをウェイクアップするために、オプションでイネーブルすることができます。
- SS1、SS2、SS3、および SS4 出力パルスは、RTC により GPIO 経由で外部デバイスへエクスポートされて、そのデバイスを作動させます。

  パルスのエクスポートは、RTC からチャンネルに接続された外部デバイスへの専用チャンネルを使って行われます。1 つの SensorStrobe チャンネル上のパルスが、他の SensorStrobe チャンネルに影響を及ぼすことはありません。出力パルスは、チャンネルごとに非反転バージョンや反転バージョンを個別にエクスポートできます。オプションで、SSx（x = 1、3）の反転バージョンを SSy（y = 2、4）の GPIO を経由してエクスポートすることができます。これらのチャンネル・ペアを使用すれば、差動ペアを作成可能です。ブロードキャスト機能はありません。ブロードキャスト機能は、1 つのチャンネル上の SensorStrobe イベントをすべての SensorStrobe チャンネルと外部デバイスに送信することを可能にします。

- 各 SensorStrobe チャンネルは、使用可能な 8 つの GPIO から最大 3 つの GPIO 入力を選択できます。これらの入力は、設定に応じ、出力パルス・イベントに関してサンプリングされます。最新のサンプル値は、非スティッキー・リードバック・ステータスとして使用できます。サンプル値に指定されたシーケンスが発生したときは、オプションで CPU に割込みを送ることができます。
- SensorStrobe チャンネルは、オンザフライで設定し直してイネーブル／ディスエーブルすることができます。再設定に含まれないチャンネルへの割込みは行われません。
- SensorStrobe チャンネルは、RTC の入力キャプチャ・チャンネルから独立しています。

RTC SensorStrobe チャンネルの波形の図に、RTC における SS1 チャンネルの SensorStrobe の動作を示します。

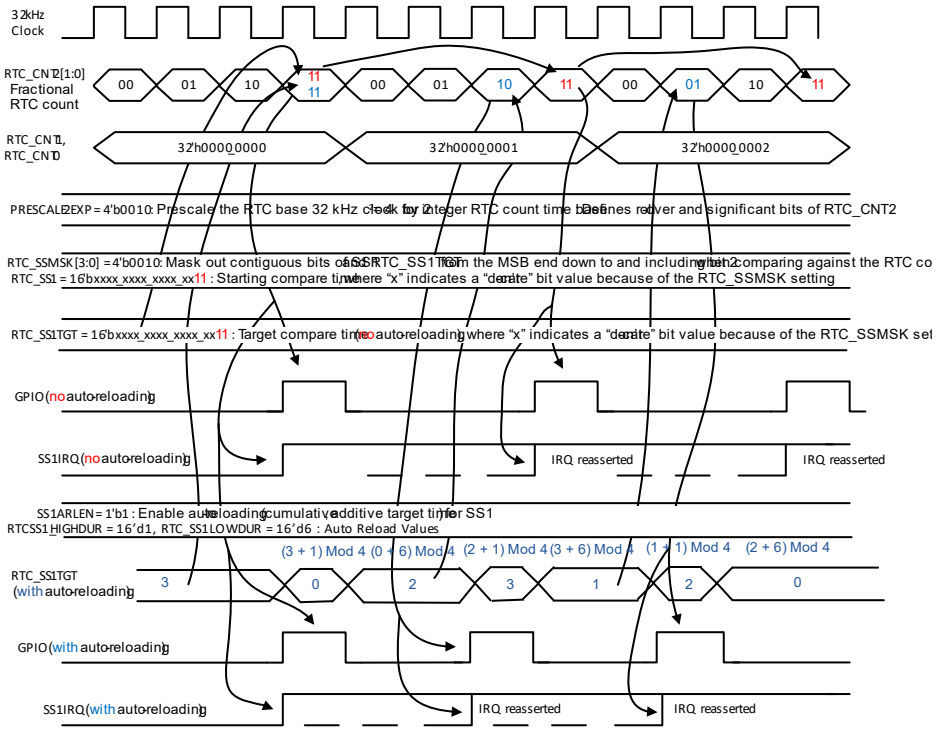


図 21-5：RTC SensorStrobe チャンネルの波形

図には、RTC が 32kHz のベース・クロックを 2^2 でプリスケール（RTC_CNT2 レジスタに 2 つの有効小数ビットを指定）したときの、SensorStrobe イベントの発生が示されています。SensorStrobe は、RTC_CR1.PRESCALE2EXP を介して選択された、2 の 0 乗から 15 乗までのすべてのプリスケーリングに対してサポートされています。

赤字の設定は、32kHz クロックの 4 サイクルごとに SS1 チャンネルで発生する SensorStrobe イベントを示しています。自動リローディングが RTC_CR4SS.SS1ARLEN を介して無効にされているので、イベントの周期は 2 のべき乗サイクルです。SS1 レジスタの値はターゲット時間を決定します。RTC_SSMSK レジスタのマスク設定により、SS1 の下位 2 ビットだけが RTC カウントと比較されます。SS1 の LSB 側にあるマスクされていないターゲット時間 2'b11 は、RTC_CNT2 レジスタが 4 回インクリメントされるたびにマッチングされるので、マスクしていないビットの数（2）により比較イベントの周期性が決まります。

青字の設定は、自動リローディングを有効にした SensorStrobe イベントの等価シーケンスを示しています。ターゲット時間への累積加算による影響は、RTC_SS1TGT レジスタで確認できます。自動リローディングを有効にすると、ターゲット時間の初期値は SS1 の値になりますが、各イベントがアクティブになると、RTC_SS1HIGHDUR と RTC_SS1LOWDUR の値が RTC_SS1TGT レジスタ内の内容に加算されて、加算の繰り越しが可能になります。

SensorStrobe マッチでは RTC_SSMSK レジスタで指定されたマスキングを考慮するので、自動リローディング時の累積加算ではモジュロ演算（モジュラ加算）が行われます。モジュロ演算は 2 比較時のマスクされていないビットの数です。

図には、自動リロード値（2'b11）のマスクされていない LSB が示されており、これが RTC_SS1TGT のマスクされていないビットに累積的に加算されます。結果にはモジュロ 4 演算が適用されます。比較イベントの周期は、自動リロード値の未マスク・ビットによって指定されます。

マスキングと自動リローディングは RTC のオプション機能です。

以下では、RTC カウントの LSB 側への自動アライメントが 16 ビットごとに行われます（プリスケーリングを考慮）。

- RTC_SS1 の初期ターゲット時間
- RTC_SS1TGT の累積ターゲット時間
- RTC_SSMSK からデコード可能なすべてのマスク

SensorStrobe イベントが発生すると、32kHz の 1 サイクル分の GPIO 出力がアクティブにされてスティッキー割込みソース（RTC_CR3SS.SS1IRQEN）がアサートされ、CPU がこの割込みソース・ビットを最後にクリアした後に発生した新しい比較イベントが記録されます。CPU はオプションで RTC_CR3SS.SS1IRQEN をイネーブルして、ウェイクアップ・コントローラおよび NVIC への RTC 割込みラインのアクティブ化に関与することができます。これは、CPU を休止状態からウェイクアップさせ、RTC SensorStrobe イベントによりアクションの実行を求められた外部センサー・デバイスからの結果を検証する場合に使用します。

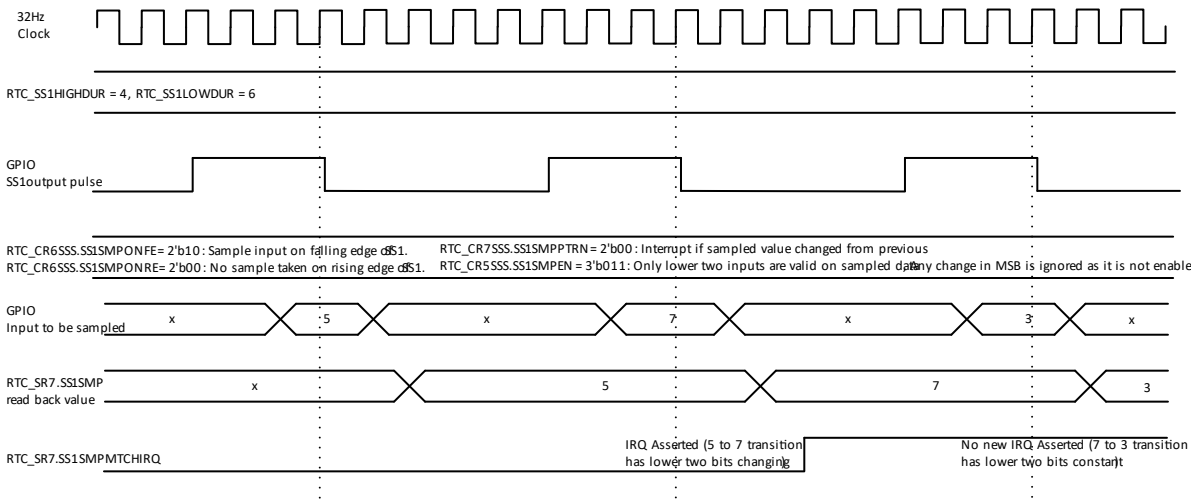


図 21-6：RTC 入力サンプリング

図は、SensorStrobe のサンプリング機構を示しています。RTC_CR6SSS.SSxSMPONFE フィールドと RTC_CR6SSS.SSxSMPONRE フィールドは、送信された出力パルスの立下がりエッジと立上がりエッジ前後における入力 GPIO のサンプリングを有効にします。この例では、設定に従い、出力パルスの立下がりエッジだけで入力がサンプリングされ、記録されます。サンプリング・ポイントとサンプル値は青でマークされています。最初のサンプリング・ポイントにおける入力値は 5 で、次のサンプリング・ポイントにおける値は 7 と 3 です。これらの値は、RTC_SR7.SS1SMP

フィールドでユーザへのリードバックとして使用できます。コアは、入力が変化したときだけ割込みが行われるように設定されています（RTC_CR7SSS.SS1SMPPTRN が 2'b00）。使用可能な 3 つの入力のうち、2 つだけがイネーブルされます。RTC_CR5SSS.SS1SMPEN の値は 3'b011 です。これは 2 つの LSB だけが考慮され、MSB はドント・ケアであることを示しています。

最初のサンプル 5（3'b101）では、前のサンプルに対する基準がないので割込みは生成されません。

次のサンプル 7（3'b111）では、下位 2 ビット目が変化しているので、コアに割込みが送られます。

3 つめのサンプル 3（3'b011）では、下位 2 ビットが両方ともその前の値と同じなので、新しい割込みは生成されません。

SensorStrobe の概要を下の表に示します。

表 21-3：SensorStrobe 機構

SensorStrobe Channel	Polarity Control	On/Off Time Mask Control	Interrupt Edge Selection	On/Off Time Fine control	Input Sampling	Differential Output Pairs
SS1	Yes	Yes	Yes	Yes	Yes	Yes
SS2	Yes	Yes	Yes	Yes	Yes	Yes
SS3	Yes	Yes	Yes	Yes	Yes	Yes
SS4	Yes	Yes	Yes	No	No	Yes

表 21-4：SensorStrobe ルックアップ・テーブル

RTC_SSx SensorStrobe Parameters	RTC_CR4SS.SSxMSKEN = 0 RTC_CR4SS.SSxARLEN = 0 (x = 1, 2, 3, 4)	RTC_CR4SS.SSxMSKEN = 1 RTC_CR4SS.SSxARLEN = 0 (x = 1, 2, 3, 4)	RTC_CR4SS.SSxMSKEN = 0 RTC_CR4SS.SSxARLEN = 1 (x = 1, 2, 3)	RTC_CR4SS.SSxMSKEN = 1 RTC_CR4SS.SSxARLEN = 1 (x = 1, 2, 3)
On Time (RTC Clocks)	$2^{RTC\_SSMSKOT.SSxMSKOT}$		RTC_SSxHIGHDUR.SSxHIGHDUR	(RTC_SSxHIGHDUR.SSxHIGHDUR) % 2^{SSxMSK}
Period (RTC Clocks)	RTC_CR4SS.RTC_SSxMSKEN? ($2^{RTC\_SSMSK.SSxMSK}$) : 2^{16}		RTC_SSxHIGHDUR.SSxHIGHDUR + RTC_SSxLOWDUR.SSxLOWDUR	(RTC_SSxHIGHDUR.SSxHIGHDUR + RTC_SSxLOWDUR.SSxLOWDUR) % 2^{SSxMSK}
Illegal Case	NA	RTC_SSMSK.SSxMSK ≤ RTC_SSMSKOT.SSxMSKOT	On Time = 0 Or Period = 0	RTC_SSMSKOT.SSxMSKOT! = 0 Or On Time = 0 Or Period = 0

RTC 入力サンプリング・プログラミング・モデル

1. SensorStrobe 用 GPIO を、**SensorStrobe 出力 GPIO** の表に従って設定します。
2. RTC1_CNT レジスタを 0 にリセットします。
3. RTC1_SSxHIGHDUR レジスタと RTC1_SSxLOWDUR レジスタの値を、SensorStrobe パルスのハイ時間とロー時間に設定します。これらのレジスタへの書込みは、SensorStrobe 出力パルスのデューティ・サイクルも設定します。ここで、x は SensorStrobe のチャンネル番号です。
4. RTC1_CR4SS.SSxARLEN に 1 を書き込むことによって、自動リロードを有効にします。ここで、x は SensorStrobe のチャンネル番号です。
5. GPIO を入力サンプリングに使用する入力として設定します。
6. RTC1_GPMUX0 と RTC1_GPMUX1 へ書込みを行うことによって、GPIO 選択用の GPIO ピン・マルチプレクサを、SensorStrobe チャンネルがサンプリングするデータとして設定します。
7. RTC1_CR6SSS レジスタへ書込みを行うことによって、選択した SS チャンネルに対して、サンプリング・エッジを立上がりエッジまたは立下がりエッジ、もしくはその両方として設定します。
8. RTC1_CR7SSS レジスタへ書込みを行うことによって、選択した SensorStrobe チャンネルに関する GPIO サンプリングのパターン・マッチング条件を設定します。
9. RTC1_CR5SSS へ書込みを行うことによって、選択した SensorStrobe チャンネルの GPIO 入力サンプリングとアクティビティ割込みをイネーブルします。
10. RTC1_CR3SS.SSxEN に 1 を書き込むことによって、SensorStrobe チャンネルをイネーブルします。ここで、x は SensorStrobe のチャンネル番号です。
11. RTC1_CR0.CNTEN に 1 を書き込むことによって、RTC をイネーブルします。これは、RTC カウントと選択したチャンネルの SensorStrobe 出力をイネーブルします。

RTC 入力キャプチャ

入力キャプチャは、外部デバイスが ADuCM4050 MCU の GPIO 入力の 1 つの状態遷移を介してイベントを送信した場合に、RTC のリアルタイム・カウントのスナップショットを取得するプロセスです。入力キャプチャ・イベントはデバイスの自律的な測定またはアクションによってトリガされます。その後、デバイスは、RTC がイベントに対応する時間のスナップショットを取得するよう、ADuCM4050 MCU に信号を送信します。スナップショットを取得することで、ADuCM4050 MCU がウェイクアップして CPU への割込みが生成されます。CPU はその後、正確な 32kHz サイクルと入力キャプチャ・イベントの正確な時間に関する情報を、RTC から取得することができます。

ADuCM4050 MCU の RTC1 には、4 つの入力キャプチャ・チャンネル、RTCIC0、RTCIC2、RTCIC3、RTCIC4 があります。RTCIC2、RTCIC3、RTCIC4 は 16 ビット入力キャプチャ・チャンネルで、RTCIC0 は 47 ビット幅のチャンネルです。

表 21-5：RTC 入力キャプチャ GPIO

RTC Input Capture Channel	MCU GPIO Pin
RTCIC0	P0_15 (GPIO15)

表 21-5：RTC 入力キャプチャ GPIO（続き）

RTC Input Capture Channel	MCU GPIO Pin
RTCIC2	P1_00 (GPIO16)
RTCIC3	P0_13 (GPIO13)
RTCIC4	P2_01 (GPIO33)

これらの独立チャンネルはそれぞれ、そのチャンネルに対応する GPIO 上で立上がりエッジまたは立下がりエッジ・イベント（チャンネルごとにどちらかのタイプを選択）が発生したときに、RTC カウントのスナップショットを取得できます。スナップショットは、外部センサー・デバイスからの入力に対し、その後の CPU による確認と処理用にタイム・スタンプを付与するために使われます（CPU のウェイクアップ後）。

入力キャプチャ・スナップショットは、RTC による 32kHz クロックのすべてのプリスケーリングに関し、このクロック 1 周期分の最大精度を備えています。プリスケーリングは、32kHz クロックを 2 のべき乗（0 乗～15 乗）で分周して低周波の時間ベースを生成し、整数で表される時間を RTC_CNT1 レジスタと RTC_CNT0 レジスタでカウントするプロセスです。この時間ベースの小数で表される時間も、32kHz クロックの周期を単位として RTC_CNT2 レジスタで同時にカウントされます。各入力キャプチャ・チャンネルには固定の GPIO ピンがあって、GPIO にエッジ（立上がりまたは立下がり）が発生した時点で RTC 時間のスナップショットを取得するよう要求します。

RTCIC0 チャンネルは他の RTCICx チャンネルと異なり、CPU が RTC に特別なキーを書き込むことで、ソフトウェア起動スナップショットを取得することもできます。

16 ビット入力キャプチャ・チャンネルは、整数カウント時間ベースのための RTC のプリスケーリング度に基づいてスナップショットを取得し、それに応じてチャンネルをアラインします。16 ビット・スナップショットには、プリスケーリングに関係する RTC_CNT2 レジスタの小数ビットと、RTC_CNT0 レジスタの LSB 側にある追加ビットが含まれます。

RTCIC0 は、RTC_CNT1、RTC_CNT0、および RTC_CNT2 レジスタのすべてのビットのスナップショットを取得します。これにより、RTCIC0 チャンネルで生成されるイベントの絶対時間スナップショットがもれなく提供されます。その他の 16 ビット入力キャプチャ・チャンネルは、32kHz クロックの 2^{16} 周期に等しい時間幅（スナップショット固有）だけを取得します。

スナップショットとメイン RTC カウントのアライメントを、**入力キャプチャ・チャンネル：RTC カウントの 16 ビットおよび 47 ビット・スナップショット**の図に示します。この図には、RTC 時間ベースの 3 つのプリスケーリング例（2^{15}、2^{2}、および 2^{0}）が示されています。16 ビット入力キャプチャ・チャンネルでは、プリスケーリングに使われる 2 の 0 乗から 15 乗までのすべてのべき乗がサポートされています。

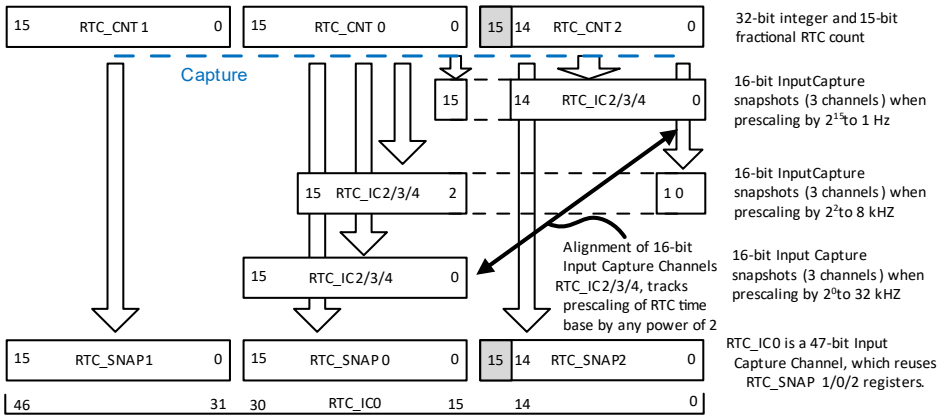


図 21-7：RTC 入力キャプチャ・チャンネル：RTC カウントの 16 ビットおよび 47 ビット・スナップショット

それぞれの入力キャプチャ・チャンネルは以下の機能を独自に制御可能で、他のチャンネルに影響を与えることなくオンザフライで変更できます。

- チャンネルのイネーブル／ディスエーブル
- スナップショットの要求をアクティブにする GPIO のエッジ極性（立上がりエッジと立下がりエッジの別）
- キャプチャ・イベント発生の割込みイネーブル
- キャプチャ・イベント用のスティッキー割込みソース
- CPU によるキャプチャ・スナップショットの読出し／未読出しステータス

変更可能な共通設定 `RTC_CR2IC.RTCICOWUSEN` は、すべての入力キャプチャ・チャンネルに適用されます。これは、まだ読み出されていない過去のスナップショットをチャンネルごとに新しいスナップショットで上書きするか、あるいは CPU が入力キャプチャ・スナップショットを読み出すまでそのスナップショットを維持するかの選択に使われます。スナップショットの上書きに関してはすべてのチャンネルに同じポリシーが適用されますが、ポリシーの実行はチャンネルごとに独自に行われます。

ADuCM4050 MCU の休止状態にあるリアルタイム・クロック（RTC1）の入力キャプチャ機能を以下に示します。

- 4 つの独立した入力キャプチャ・チャンネルをサポート（3 つの 16 ビット・チャンネルと 1 つの 47 ビット・チャンネル）。
  - 3 つの 16 ビット・チャンネルは、以下のように 16 個のビットを使って RTC 経過時間カウントのスナップショットを取得します。

    ```
    {least_significant_integer_bits_to_bring_total_to_16,
    fractional_bits_due_to_prescaling}
    ```

    この連結されたキャプチャ時間の合計ビット数は 16 です。

    ターゲット内の小数ビットの数は、32kHz ベース・クロックに対して設定されたプリスケーリングの程度によって異なり、入力キャプチャ機能に使用する小数ビットの数もプリスケーリングに従います。

    16 ビットのターゲット時間の残りビットは、RTC の整数カウントの LSB 側にある整数で構成されます。小数ビットと整数ビットの（プリスケーリングによる）合計数は、16 でなければなりません。

  - 47 ビット・チャンネルは入力キャプチャ・チャンネル用に絶対時間をキャプチャします。ここで、キャプチャ時間は {32_integer_bits, 15_fractional_bits} で与えられます。
- 4 つある入力キャプチャ・チャンネルのそれぞれでは、キャプチャ時間にすべての関連小数ビットが含まれるので、スナップショットの最小分解能は常に個々の 32kHz クロック・サイクルとなります。
- RTC が入力キャプチャ・イベントを検出すると、そのイベントのスナップショット時間の精度は以下のようになります。
  - 3 つの 16 ビット・チャンネルでは、各チャンネルが、イベント発生時間のスナップショットを 32kHz の 1 サイクルまでの精度でキャプチャします。
  - 1 つの 47 ビット入力チャンネルでは、イベントに対してキャプチャされたスナップショット値に、32kHz の 1 サイクルに相当する小数部の時間分の固定遅延があります。

    小数部カウントの 1LSB に相当するこの遅延は、ソフトウェアにより後で差し引くことができます。
- 入力キャプチャ・イベントは、そのチャンネルの GPIO 入力上でのローからハイまたはハイからローへの遷移として設定できます。この遷移の極性はチャンネルごとに設定可能です。入力キャプチャ・チャンネルには、イベントを発生させる遷移のタイプを個別に指定できます。
- 4 つある入力キャプチャ・チャンネルのそれぞれには読出し可能でスティッキーな割込みソース・ビットがあり、入力キャプチャ・イベントが発生すると常にこれがアクティブになります（ハイに固定される）。この割込みソース・ビットは、割込みをかけて（割込みの関与項となって）RTC からのラインをウェイクアップするために、オプションでイネーブルすることができます。
- ユーザは、RTC のすべての入力キャプチャ・チャンネルに対して、以下のいずれかのグローバル上書きポリシーを設定することができます。
  - 所定のチャンネル上で新しい入力キャプチャ・イベントが発生した場合は、そのチャンネルで直前にキャプチャされた値に新しいスナップショット値が上書きされます。
  - 所定のチャンネルにおいて、新しいイベントでスナップショットを上書きできるのは、そのチャンネルの既存のスナップショット値を既に CPU が読み出した場合に限られます。

    注：RTC は CPU に対し、新しい入力キャプチャ・イベントの受信時にアクティブに固定される読出し可能割込みソースに加えて、入力キャプチャ・スナップショットの読出し／未読出しステータスを確認するフラグも提供します。CPU が、まだ読み出していないスナップショットを入力キャプチャ・チャンネルの新しいイベントで上書きするのを許可しないポリシーを使用している場合、これはその助けとなります。
- 入力キャプチャ・チャンネルはオンザフライで設定し直してイネーブル／ディスエーブルすることができ、再設定に含まれないチャンネルに対する割込みは行われません。
- それぞれの入力キャプチャ・チャンネルは互いに独立して動作し、RTC の SensorStrobe チャンネルからも独立しています。

RTC 入力キャプチャ・チャンネル波形の図に、入力キャプチャの動作を示します。この図は、RTC が 32kHz のベース・クロックを 2^2（RTC_CNT2 に 2 つの有効小数ビットを指定）でプリスケールしたときのキャプチャ・イベントの発生を示していますが、入力キャプチャは、2 のべき乗（0〜15）のすべてのプリスケーリング（RTC_CR1.PRESCALE2EXP を介して選択）に対してサポートされています。

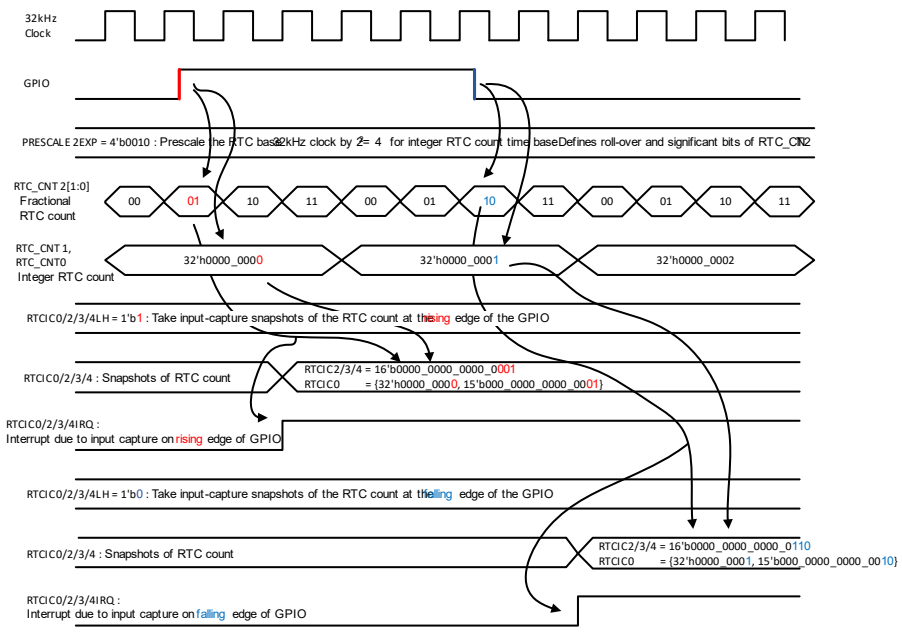


図 21-8：RTC 入力キャプチャ・チャンネル波形

所定の入力キャプチャ・チャンネルに関しては、RTC_CR2IC.RTCIC2LH/RTC_CR2IC.RTCIC3LH/RTC_CR2IC.RTCIC4LH フィールドを使用し、GPIO の立上がりエッジまたは立下がりエッジで（両方のエッジを使用することはできない）RTC 時間のスナップショットを取得することができます。入力キャプチャ・イベントが発生すると、16 ビット入力キャプチャ・チャンネルが、GPIO エッジが発生した個々の 32kHz サイクルを特定するのに必要な RTC_CNT2 レジスタと RTC_CNT0 レジスタから、16 個の有効 LSB（プリスケーリングを考慮）の最新値を保存します。47 ビット入力キャプチャ・チャンネル（RTCIC0）は、RTC_CNT1、RTC_CNT0、および RTC_CNT2 レジスタのすべてのビットを保存し、1 つの 32kHz サイクルを特定するためのスナップショット幅を広げます。

RTC_CR2IC.RTCICOWUSEN が新しい GPIO エッジによる未読出しスナップショットの上書きを許可しない場合、RTC 時間のスナップショットは、CPU によって読み出されるまでスティッキーなものとなります（上書きされない）。スナップショットが取得されると、スティッキー割込みソース（RTC_SR3.RTCIC0IRQ/RTC_SR3.RTCIC2IRQ/RTC_SR3.RTCOC3IRQ/RTC_SR3.RTCIC4IRQ）がハイになり、CPU が最後に割込みソース・ビットをクリアした後に新しい入力キャプチャ・イベントが発生したことが記録されます。CPU はオプションでこれらのソースをイネーブルして、RTC からウェイクアップ・コントローラおよび NVIC への割込みラインに関与することができます。

チャンネルは、4 つの入力キャプチャ・チャンネルと 4 つの SensorStrobe チャンネルから任意の数（0～8）のチャンネルを選んで、同時に使用することができます。

RTC 消費電力モードの動作

休止モードとシャットダウン・モードにおける MMR の内容を表に示します。

表 21-6：RTC レジスタの一覧

Register	RTC0 Retention Status		RTC1 Retention Status	
	Hibernate Mode	Shutdown Mode	Hibernate Mode	Shutdown Mode
RTC_CR0	Yes	Yes	Yes	No
RTC_SR0	Yes	Yes	Yes	No
RTC_SR1	Yes	Yes	Yes	No
RTC_CNT0	Yes	Yes	Yes	No
RTC_CNT1	Yes	Yes	Yes	No
RTC_ALM0	Yes	Yes	Yes	No
RTC_ALM1	Yes	Yes	Yes	No
RTC_TRM	Yes	Yes	Yes	No
RTC_GWY	No	No	No	No
RTC_CR1	Yes	Yes	Yes	No
RTC_SR2	Yes	Yes	Yes	No
RTC_SNAP0	Yes	Yes	Yes	No
RTC_SNAP1	Yes	Yes	Yes	No
RTC_SNAP2	Yes	Yes	Yes	No
RTC_MOD	Yes	Yes	Yes	No
RTC_CNT2	Yes	Yes	Yes	No
RTC_ALM2	Yes	Yes	Yes	No
RTC_SR3	NA	NA	Yes	No
RTC_CR2IC	NA	NA	Yes	No
RTC_CR3SS	NA	NA	Yes	No
RTC_CR4SS	NA	NA	Yes	No
RTC_SSMSK	NA	NA	Yes	No
RTC_IC2	NA	NA	Yes	No
RTC_IC3	NA	NA	Yes	No
RTC_IC4	NA	NA	Yes	No
RTC_SS1	NA	NA	Yes	No
RTC_SS2	NA	NA	Yes	No
RTC_SS3	NA	NA	Yes	No
RTC_SS4	NA	NA	Yes	No
RTC_SR4	NA	NA	Yes	No

表 21-6：RTC レジスタの一覧（続き）

Register	RTC0 Retention Status		RTC1 Retention Status	
	Hibernate Mode	Shutdown Mode	Hibernate Mode	Shutdown Mode
RTC_SR5	NA	NA	Yes	No
RTC_SR6	NA	NA	Yes	No
RTC_SS1TGT	NA	NA	Yes	No
RTC_FRZCNT	NA	NA	No	No
RTC_SS2TGT	NA	NA	Yes	No
RTC_SS3TGT	NA	NA	Yes	No
RTC_SS1LOWDUR	NA	NA	Yes	No
RTC_SS2LOWDUR	NA	NA	Yes	No
RTC_SS3LOWDUR	NA	NA	Yes	No
RTC_SS1HIGHDUR	NA	NA	Yes	No
RTC_SS2HIGHDUR	NA	NA	Yes	No
RTC_SS3HIGHDUR	NA	NA	Yes	No
RTC_SSMSKOT	NA	NA	Yes	No
RTC_CR5SSS	NA	NA	Yes	No
RTC_CR6SSS	NA	NA	Yes	No
RTC_CR7SSS	NA	NA	Yes	No
RTC_SR7	NA	NA	Yes	No
RTC_SR8	NA	NA	Yes	No
RTC_SR9	NA	NA	Yes	No
RTC_GPMUX0	NA	NA	Yes	No
RTC_GPMUX1	NA	NA	Yes	No

ADuCM4050 RTC レジスタの説明

リアルタイム・クロック（RTC）には以下のレジスタが含まれています。

表 21-7：ADuCM4050 の RTC レジスター覧

レジスタ名	説明
RTC_ALM0	RTC アラーム 0
RTC_ALM1	RTC アラーム 1
RTC_ALM2	RTC アラーム 2
RTC_CNT0	RTC カウント 0

表 21-7：ADuCM4050 の RTC レジスター一覧（続き）

レジスタ名	説明
RTC_CNT1	RTC カウント 1
RTC_CNT2	RTC カウント 2
RTC_CR0	RTC 制御 0
RTC_CR1	RTC 制御 1
RTC_CR2IC	入力キャプチャ・チャンネル設定用の RTC 制御 2
RTC_CR3SS	SensorStrobe チャンネル設定用の RTC 制御 3
RTC_CR4SS	SensorStrobe チャンネル設定用の RTC 制御 4
RTC_CR5SSS	SensorStrobe チャンネル GPIO サンプリング設定用の RTC 制御 5
RTC_CR6SSS	SensorStrobe チャンネル GPIO サンプリング・エッジ設定用の RTC 制御 6
RTC_CR7SSS	SensorStrobe チャンネル GPIO サンプリング・アクティビティ設定用の RTC 制御 7
RTC_FRZCNT	RTC フリーズ・カウント
RTC_GPMUX0	RTC GPIO ピン・マルチプレクサ・コントロール・レジスタ 0
RTC_GPMUX1	RTC GPIO ピン・マルチプレクサ・コントロール・レジスタ 1
RTC_GWY	RTC ゲートウェイ
RTC_IC2	RTC 入力キャプチャ・チャンネル 2
RTC_IC3	RTC 入力キャプチャ・チャンネル 3
RTC_IC4	RTC 入力キャプチャ・チャンネル 4
RTC_MOD	RTC モジュロ
RTC_SS1	RTC SensorStrobe チャンネル 1
RTC_SS1HIGHDUR	SensorStrobe チャンネル 1 の RTC 自動リロード・ハイ時間
RTC_SS1LOWDUR	SensorStrobe チャンネル 1 の RTC 自動リロード・ロー時間
RTC_SS1TGT	RTC SensorStrobe チャンネル 1 のターゲット
RTC_SS2	RTCSensorStrobe チャンネル 2
RTC_SS2HIGHDUR	SensorStrobe チャンネル 2 の RTC 自動リロード・ハイ時間
RTC_SS2LOWDUR	SensorStrobe チャンネル 2 の RTC 自動リロード・ロー時間
RTC_SS2TGT	RTC SensorStrobe チャンネル 2 のターゲット
RTC_SS3	RTCSensorStrobe チャンネル 3
RTC_SS3HIGHDUR	SensorStrobe チャンネル 3 の RTC 自動リロード・ハイ時間
RTC_SS3LOWDUR	SensorStrobe チャンネル 3 の RTC 自動リロード・ロー時間
RTC_SS3TGT	RTC SensorStrobe チャンネル 3 のターゲット
RTC_SS4	RTCSensorStrobe チャンネル 4

表 21-7：ADuCM4050 の RTC レジスター一覧（続き）

レジスタ名	説明
RTC_SSMSK	SensorStrobe チャンネル用 RTC マスク
RTC_SSMSKOT	SensorStrobe チャンネル・オン時間制御用 RTC マスク
RTC_SNAP0	RTC スナップショット 0
RTC_SNAP1	RTC スナップショット 1
RTC_SNAP2	RTC スナップショット 2
RTC_SR0	RTC ステータス 0
RTC_SR1	RTC ステータス 1
RTC_SR2	RTC ステータス 2
RTC_SR3	RTC ステータス 3
RTC_SR4	RTC ステータス 4
RTC_SR5	RTC ステータス 5
RTC_SR6	RTC ステータス 6
RTC_SR7	RTC ステータス 7
RTC_SR8	RTC ステータス 8
RTC_SR9	RTC ステータス 9
RTC_TRM	RTC トリム

RTC アラーム 0

RTC_ALM0 には、非小数（プリスケールされた）RTC アラーム・ターゲット時間値の下位 16 ビットが格納されます。

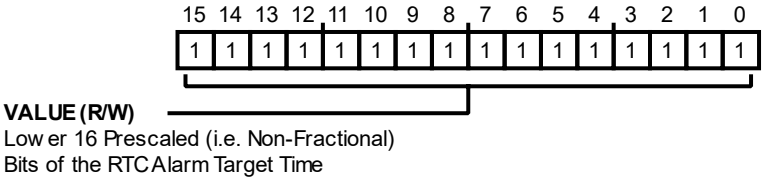


図 21-9：RTC_ALM0 レジスタ図

表 21-8：RTC_ALM0 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	RTC アラーム・ターゲット時間のプリスケールされた（つまり非小数部の）下位 16 ビット。 アラーム・レジスタは、誤アラームを避けるために RTC カウントと異なるリセット値を使用します。

RTC アラーム 1

RTC_ALM1 には、非小数（プリスケールされた）RTC アラーム・ターゲット時間値の上位 16 ビットが格納されます。

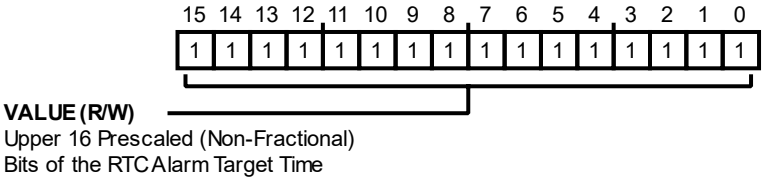


図 21-10：RTC_ALM1 レジスタ図

表 21-9：RTC_ALM1 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	RTC アラーム・ターゲット時間のプリスケールされた（非小数部）上位 16 ビット。 アラーム・レジスタは、誤アラームを避けるために RTC カウントと異なるリセット値を使用します。

RTC アラーム 2

RTC_ALM2 は、RTC アラーム・ターゲット時間値の小数部（プリスケールされていない）ビットを、個々の 32kHz クロック・サイクルまで指定します。

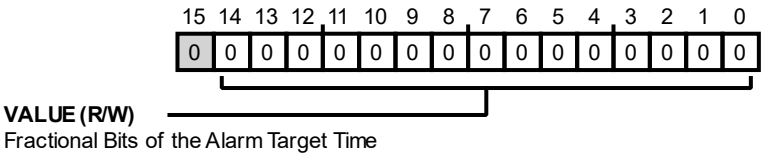


図 21-11：RTC_ALM2 レジスタ図

表 21-10：RTC_ALM2 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
14:0 (R/W)	VALUE	アラーム・ターゲット時間の小数ビット。 RTC アラーム・ターゲット時間の小数部（プリスケールされていない）ビット。

RTC カウント 0

RTC_CNT0 には、経過したプリスケール RTC 時間単位でリアルタイム・カウントを維持する RTC カウンタの下位 16 ビットが格納されます。

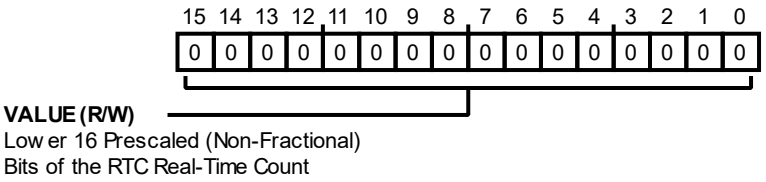


図 21-12：RTC_CNT0 レジスタ図

表 21-11：RTC_CNT0 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	RTC リアルタイム・カウントのプリスケールされた（非小数部）下位 16 ビット。

RTC カウント 1

RTC_CNT1 には、経過したプリスケール RTC 時間単位でリアルタイム・カウントを維持する RTC カウンタの上位 16 ビットが格納されます。

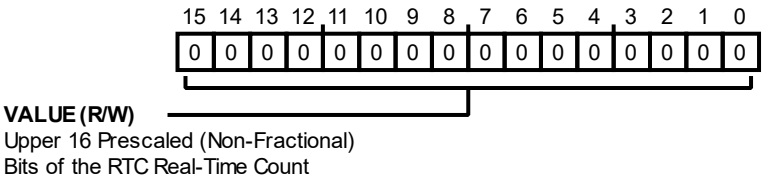


図 21-13：RTC_CNT1 レジスタ図

表 21-12：RTC_CNT1 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	RTC リアルタイム・カウントのプリスケールされた（非小数部）上位 16 ビット。

RTC カウント 2

RTC_CNT2 には、RTC カウントの小数部が格納されます。このカウントはプリスケールされた時間単位で表示され、{RTC_CNT1, RTC_CNT0} . {RTC_CNT2}で与えられます。RTC_CNT2 の小数ビットを含むリアルタイム・カウントの全体的な分解能は、32kHz クロックの 1 周期です。

RTC_CNT2 レジスタは RTC1 にのみ存在します。RTC0 では、CPU が RTC カウントの小数部を読み出すことはできません。

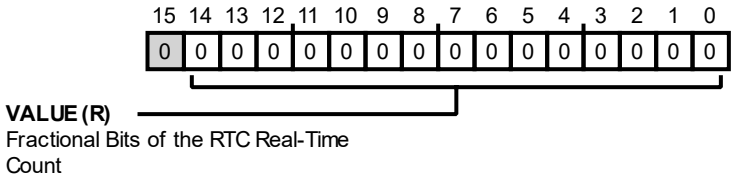


図 21-14：RTC_CNT2 レジスタ図

表 21-13：RTC_CNT2 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
14:0 (R/NW)	VALUE	RTC リアルタイム・カウントの小数ビット。 以下のいずれかのイベントが発生すると、RTC_CNT2 はゼロに設定されます。 (i) CPU が RTC_CNT1 レジスタと RTC_CNT0 レジスタに新しい値ペアを書き込んで経過時間単位カウントを再定義する一方で、RTC がイネーブルされ、ポストされたツイン書込みが実行される。 (ii) CPU が、RTC_CR0.CNTEN フィールドを使ってディスエーブルされた RTC をイネーブルする。 (iii) RTC がイネーブルの場合に、RTC_CR1.PRESCALE2EXP フィールドを介して RTC のプリスケール度が変更される。

RTC 制御 0

RTC_CR0 は 2 つある RTC 用コントロール・レジスタのメインとなる側で、もう 1 つが RTC_CR1 です。

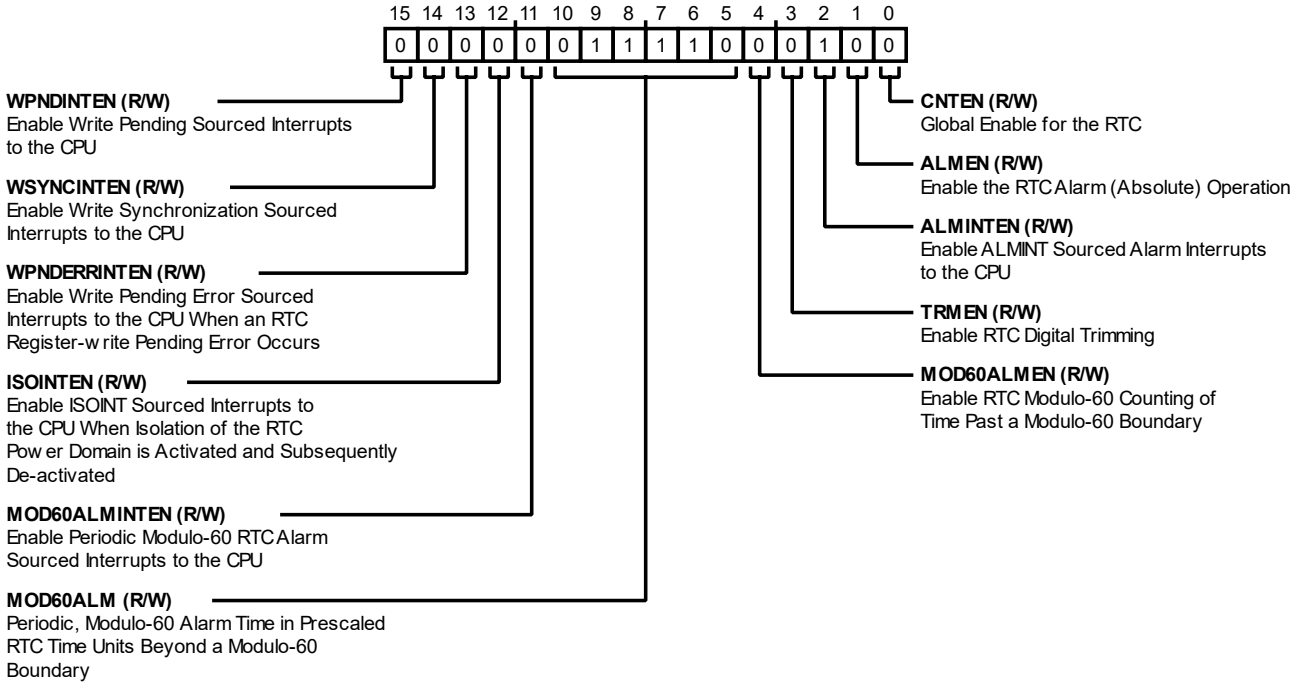


図 21-15：RTC_CR0 レジスタ図

表 21-14：RTC_CR0 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15 (R/W)	WPNDINTEN	CPU への書込み保留ソース割込みをイネーブルします。 このフィールドは、RTC_SR0.WPNDINT スティッキー割込みソース・フィールドに基づいて、CPU への RTC 割込みをイネーブルします。	
		0	RTC_SR0.WPNDINT をソースとする CPU への割込みをディスエーブルします。
		1	RTC_SR0.WPNDINT をソースとする CPU への割込みをイネーブルします。

表 21-14：RTC_CR0 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
14 (R/W)	WSYNCINTEN	CPU への書込み同期ソース割込みをイネーブルします。 このフィールドは、RTC_SR0.WSYNCINT スティッキー割込みソース・フィールドに基づいて、CPU への RTC 割込みをイネーブルします。	
		0	RTC_SR0.WSYNCINT をソースとする CPU への割込みをディスエーブルします。
		1	RTC_SR0.WSYNCINT をソースとする CPU への割込みをイネーブルします。
13 (R/W)	WPNDERRINTEN	RTC レジスタ書込み保留エラーの発生時に、CPU への書込み保留エラー・ソース割込みをイネーブルします。 このフィールドは、RTC_SR0.WPNDINT スティッキー割込みソース・フィールドに基づいて CPU への RTC 割込みをイネーブルします。	
		0	RTC で書込み保留エラーが発生した場合に割込みをディスエーブルします。
		1	RTC で書込み保留エラーが発生した場合に割込みをイネーブルします。
12 (R/W)	ISOINTEN	RTC 電源領域が一度アクティブになり、その後、非アクティブになったときに、CPU への ISOINT ソース割込みをイネーブルします。 このフィールドは、RTC_SR0.ISOINT スティッキー割込みソースに基づいて、CPU への RTC 割込みをイネーブルします。 このビット・フィールドは RTC1 にのみ存在します。RTC0 ではこのフィールドは予備で、ゼロとして読み出されます。	
		0	RTC_SR0.ISOINT をソースとする CPU への割込みをディスエーブルします。
		1	RTC_SR0.ISOINT をソースとする CPU への割込みをイネーブルします。
11 (R/W)	MOD60ALMINTEN	CPU への定期的なモジュロ 60 RTC アラーム・ソース割込みをイネーブルします。 このフィールドは、RTC_SR0.MOD60ALMINT スティッキー割込みソースに基づいて、CPU への RTC 割込みをイネーブルします。 このビット・フィールドは RTC1 にのみ存在します。RTC0 ではこのフィールドは予備で、ゼロとして読み出されます。	
		0	モジュロ 60 RTC 経過時間による定期的割込みをディスエーブルします。
		1	モジュロ 60 RTC 経過時間による定期的割込みをイネーブルします。

表 21-14：RTC_CR0 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応
10:5 (R/W)	MOD60ALM	モジュロ 60 境界を超えるプリスケール RTC 時間単位で表した、周期的モジュロ 60 アラーム時間。 このフィールドは、CPU が、RTC からの周期的な（繰り返し）アラーム割込みを、RTC_CNT1 および RTC_CNT0 内の RTC 整数カウントのモジュロ 60 の境界（繰り越しイベント）からプリスケール RTC 時間単位の任意の整数個分離れた位置に配置することを可能にします。 使用できる値は 0～59 です。これより大きい値を設定すると、その値はゼロ・プリスケール RTC 時間単位として扱われます。 以下のいずれかのイベントが発生した場合は、常に境界が定義されます。 (i) CPU が新しい値ペアを RTC_CNT1 レジスタと RTC_CNT0 レジスタに書き込む。 (ii) CPU が RTC_CR0.CNTEN フィールドを使って、ディスエーブルされた RTC をイネーブルする。 (iii) RTC_CR0.CNTEN によって RTC がイネーブルされている場合に、RTC_CR1.PRESCALE2EXP フィールドを介して RTC のプリスケール度が変更される。 このビット・フィールドは RTC1 にのみ存在します。RTC0 ではこのフィールドは予備で、すべてゼロとして読み出されます。
4 (R/W)	MOD60ALMEN	モジュロ 60 境界後の時間の RTC モジュロ 60 カウントを有効化。 RTC_CR0.MOD60ALMEN は、RTC_CNT1 と RTC_CNT0 の周期的割込み条件が、RTC_CR0.MOD60ALM で定義されたモジュロ 60 境界後のプリスケール RTC 単位数に達したことを RTC が検出し記録する動作を、CPU によってクリアされるまで有効にします。 このビット・フィールドは RTC1 にのみ存在します。RTC0 ではこのフィールドは予備で、ゼロとして読み出されます。 0：モジュロ 60 RTC 経過時間のカウントを無効にします。 1：モジュロ 60 RTC 経過時間のカウントを有効にします。
3 (R/W)	TRMEN	RTC デジタル・トリミングを有効化。 RTC のトリミングは、プリスケール RTC 時間単位によるリアルタイム・カウントを定期的に調整し、より高い精度で時間を追跡することを可能にします。RTC_CR0.CNTEN を介して RTC がイネーブルされている場合は、RTC_CR0.TRMEN がこの調整を有効にします。 RTC_CR0.CNTEN もアクティブな状態で、ディスエーブルされた RTC_CR0.TRMEN をアクティブにした場合は、トリム間隔境界となり、新しいトリム間隔が開始されます。このような RTC_CR0.TRMEN をアクティブにしても、RTC カウントのトリム調整は行われません。調整は、有効にされたトリム間隔が最初から最後まで経過した時点で初めて行われます。 0：RTC カウントのデジタル・カウントを無効にします。 1：トリムを有効にします。

表 21-14：RTC_CR0 レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
2 (R/W)	ALMINTEN	ALMINT をソースとする CPU へのアラーム割込みをイネーブルします。 RTC_CR0.ALMINTEN は、RTC_SR0.ALMINT スティッキー割込みソースに基づいて、CPU への割込みをイネーブルします。	
		0	アラーム割込みをディスエーブル。
		1	RTC アラーム値とカウント値が一致した場合に、割込みをイネーブルします。
1 (R/W)	ALMEN	RTC アラーム（絶対）動作をイネーブルします。 アラーム・ロジックを機能させてあらゆるアラーム・イベントが検出されるようにするには、RTC_CR0.ALMEN をアクティブに設定する必要があります。このようなイベントは、RTC カウントの値とアラーム・レジスタの値の一致として定義されます。アラーム・レジスタは、RTC_CNT1、RTC_CNT0、RTC_CNT2、RTC_ALM1、RTC_ALM0、および RTC_ALM2 です。	
		0	アラーム・イベントの検出を無効にします。
		1	アラーム・イベントの検出を有効にします。
0 (R/W)	CNTEN	RTC のグローバル・イネーブル。 RTC_CR0.CNTEN は、経過したリアルタイムのカウントを有効にし、RTC のマスタ・イネーブルとして機能します。 すべての割込みソースは、CNTEN の値に関係なく CPU によってクリアすることができます。 RTC_CR0.CNTEN をアクティブにすることにより RTC をイネーブルした場合は、このイベントが発生すると、RTC が使用するプリスケーラ、トリム間隔、およびモジュロ 60 カウンタのリアライメントが行われ、RTC_SR0.MOD60ALMINT をソースとする割込みが生成されます。	
		0	RTC をディスエーブルします。
		1	RTC をイネーブルします。

RTC 制御 1

RTC_CR1 レジスタは、RTC_CR0 を介して既に設定されている RTC 制御の細かさを拡張します。

RTC_CR1 は RTC1 でのみ設定可能です。RTC0 では、これに該当するレジスタは読出し専用の固定（リセット）設定です。

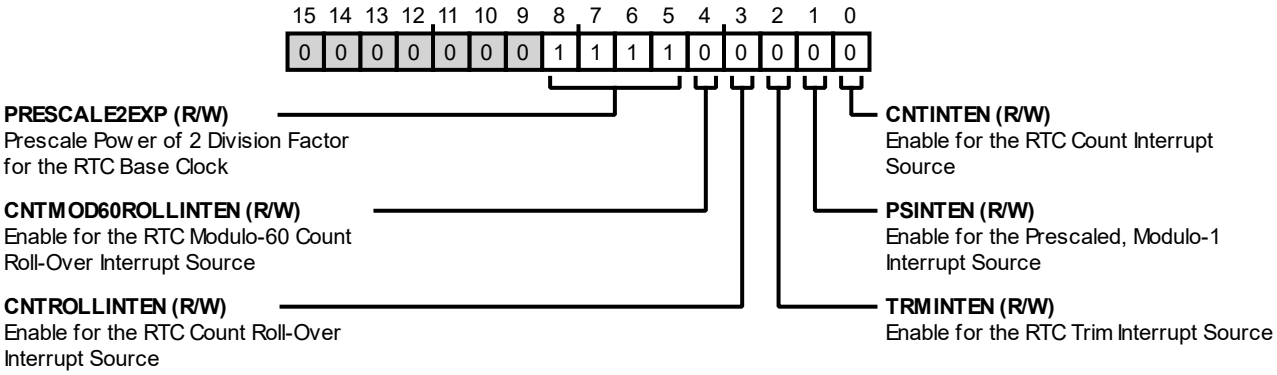


図 21-16：RTC_CR1 レジスタ図

表 21-15：RTC_CR1 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
8:5 (R/W)	PRESCALE2EXP	2 のべき乗で定義される RTC ベース・クロック用プリスケール係数。 RTC_CR1.PRESCALE2EXP は 2 のべき乗を定義し、RTC_CNT1 レジスタと RTC_CNT0 レジスタの内容を増加することによって時間のカウントに使用される前に、このべき乗によって RTC ベース・クロック（32kHz）がプリスケール（周波数で分周）されます。このビットは、また、RTC_CNT2 レジスタに格納されるカウントの小数ビットの数も定義します。 例えば、RTC_CR1.PRESCALE2EXP が 0xF の場合、RTC_CNT1 と RTC_CNT2 は 1Hz のレート（32768/2^15）でインクリメントし、RTC_CNT2 は、RTC_CNT0 の値を 1 だけ増加させるために、32768Hz のベース・クロック周波数で 0 から 2^15 までの時間単位を順番にカウントします。 このビット・フィールドは RTC1 でのみ設定可能です。RTC0 ではこのフィールドは読出し専用で、値 0xF が格納されています。これは、プリスケーリングが 2 の 15 乗に固定されていることを示します。これは、RTC0 が常に 1Hz でリアルタイムのカウントを行うからです。これに対し、RTC1 は［15,0］を範囲とする任意の 2 のべき乗数でクロックをプリスケールし、その時間ベースとして使用することができます。	
		0	RTC ベース・クロックを 2^0 = 1 でプリスケールします。
		1	RTC ベース・クロックを 2^1 = 2 でプリスケールします。
		2	RTC ベース・クロックを 2^2 = 4 でプリスケールします。
		3	RTC ベース・クロックを 2^3 = 8 でプリスケールします。
		4	RTC ベース・クロックを 2^4 = 16 でプリスケールします。
		5	RTC ベース・クロックを 2^5 = 32 でプリスケールします。

表 21-15：RTC_CR1 レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
		6	RTC ベース・クロックを 2^6 = 64 でプリスケールします。
		7	RTC ベース・クロックを 2^7 = 128 でプリスケールします。
		8	RTC ベース・クロックを 2^8 = 256 でプリスケールします。
		9	RTC ベース・クロックを 2^9 = 512 でプリスケールします。
		10	RTC ベース・クロックを 2^10 = 1024 でプリスケールします。
		11	RTC ベース・クロックを 2^11 = 2048 でプリスケールします。
		12	RTC ベース・クロックを 2^12 = 4096 でプリスケールします。
		13	RTC ベース・クロックを 2^13 = 8192 でプリスケールします。
		14	RTC ベース・クロックを 2^14 = 16384 でプリスケールします。
		15	RTC ベース・クロックを 2^15 = 32768 でプリスケールします。
4 (R/W)	CNTMOD60ROLLINT- EN	RTC モジュロ 60 カウント繰り越し割込みソースをイネーブルします。 割込みソースは RTC_SR2.CNTMOD60ROLLINT ビット・フィールドです。このビット・フィールドは RTC1 にのみ存在します。RTC0 ではこのフィールドは予備で、ゼロとして読み出されます。	
		0	RTC ペリフェラル割込みのファン・イン項としての RTC_SR2.CNTMOD60ROLLINT をディスエーブルします。
		1	RTC ペリフェラル割込みのファン・イン項としての RTC_SR2.CNTMOD60ROLLINT をイネーブルします。
3 (R/W)	CNTROLLINTEN	RTC カウント繰り越し割込みソースをイネーブルします。 このビット・フィールドは RTC1 にのみ存在します。RTC0 ではこのフィールドは予備で、ゼロとして読み出されます。	
		0	RTC ペリフェラル割込みのファン・イン項としての RTC_SR2.CNTROLLINT をディスエーブルします。
		1	RTC ペリフェラル割込みのファン・イン項としての RTC_SR2.CNTROLLINT をイネーブルします。
2 (R/W)	TRMINTEN	RTC トリム割込みソースをイネーブルします。 このビット・フィールドは RTC1 にのみ存在します。RTC0 ではこのフィールドは予備で、ゼロとして読み出されます。	
		0	RTC ペリフェラル割込みのファン・イン項としての RTC_SR2.TRMINT をディスエーブルします。
		1	RTC ペリフェラル割込みのファン・イン項としての RTC_SR2.TRMINT をイネーブルします。

表 21-15：RTC_CR1 レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
1 (R/W)	PSINTEN	RTC_SR2.PSINT のプリスケールされたモジュロ 1 割込みソースをイネーブルします。 このビット・フィールドは RTC1 にのみ存在します。RTC0 ではこのフィールドは予備で、ゼロとして読み出されます。	
		0	RTC ペリフェラル割込みのファン・イン項としての RTC_SR2.PSINT をディスエーブルします。
		1	RTC ペリフェラル割込みのファン・イン項としての RTC_SR2.PSINT をイネーブルします。
0 (R/W)	CNTINTEN	RTC カウント割込みソースをイネーブルします。 このビット・フィールドは RTC1 にのみ存在します。RTC0 ではこのフィールドは予備で、ゼロとして読み出されます。	
		0	RTC ペリフェラル割込みのファン・イン項としての RTC_SR2.CNTINT をディスエーブルします。
		1	RTC ペリフェラル割込みのファン・イン項としての RTC_SR2.CNTINT をイネーブルします。

入力キャプチャ・チャンネル設定用の RTC 制御 2

RTC_CR2IC は、入力キャプチャ・チャンネルに関係するイネーブルを設定するためのコントロール・レジスタです。RTC_CR2IC には、各チャンネルの入力キャプチャ機能自体のイネーブル機能と、チャンネル上のイベントを RTC から CPU およびウェイクアップ・コントローラへの割込みラインに関与させるかどうかに関するイネーブル機能の両方があります。

RTC 入力キャプチャ・チャンネルと GPIO パッド：入力キャプチャ・チャンネル［0,2,3,4］は、それぞれ外部割込みピン［0,1,2,3］に接続されます。

このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

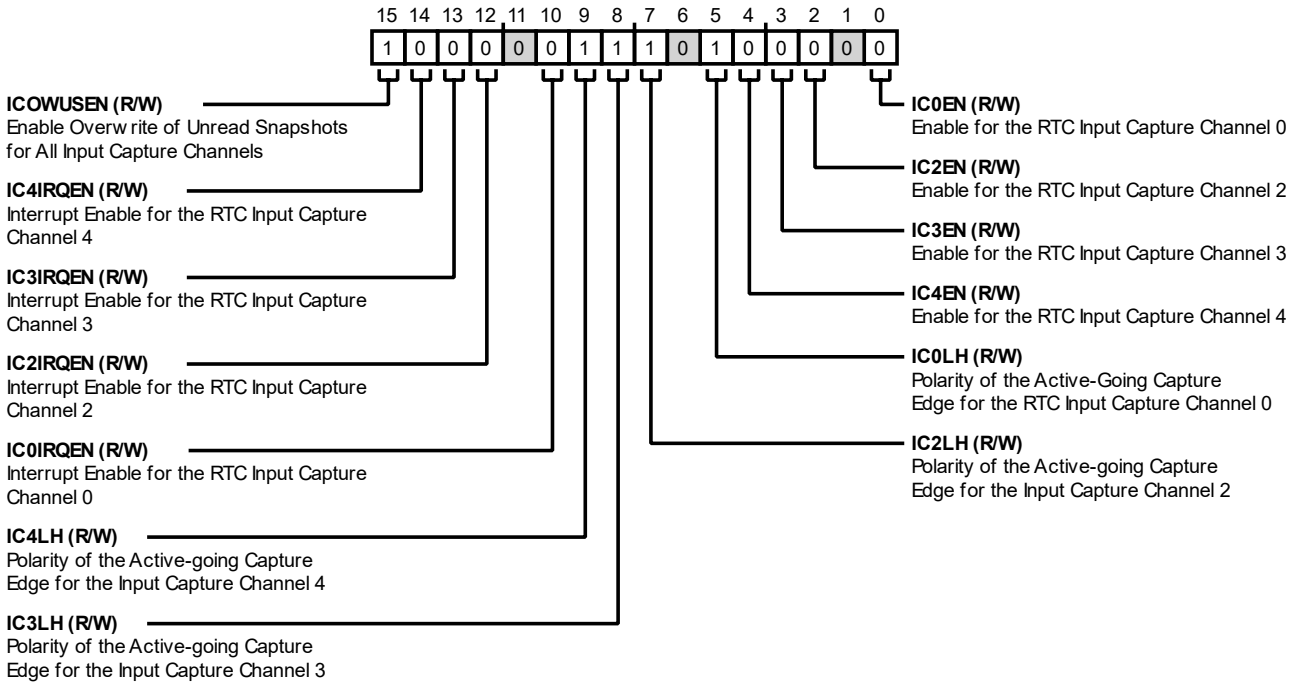


図 21-17：RTC_CR2IC レジスタ図

表 21-16：RTC_CR2IC レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
15 (R/W)	ICOWUSEN	すべての入力キャプチャ・チャンネルについて、未読出しのスナップショットの上書きをイネーブルします。 このフィールドは、新しいキャプチャ・イベントの発生時に、その前にキャプチャしたスナップショットを CPU がまだ読み出していなくても、その入力キャプチャ・チャンネルのスナップショットを自動的に上書きするかどうかを制御します。	
		0	スナップショットは最初の読出しまで維持され、その後はキャプチャ・イベントに支配されます。未読出しスナップショットの上書きは有効化されません。
		1	スナップショットは、新しいキャプチャ・イベントが発生するまで維持されます。未読出しスナップショットの上書きが有効になります。
14 (R/W)	IC4IRQEN	RTC 入力キャプチャ・チャンネル 4 の割込みイネーブル。 このフィールドはアクティブ・ハイで、スティッキー割込みソース RTC_SR3.IC4IRQ をイネーブルし、関与項として CPU への RTC IRQ 割込みラインへファン・インするかどうかを決定します。	
		0	CPU とウェイクアップ・コントローラ両方への RTC 割込みに対する RTC_SR3.IC4IRQ の関与をディスエーブルします。
		1	CPU とウェイクアップ・コントローラ両方への RTC 割込みに関与するファン・イン項としての RTC_SR3.IC4IRQ をイネーブルします。
13 (R/W)	IC3IRQEN	RTC 入力キャプチャ・チャンネル 3 の割込みイネーブル。 このフィールドはアクティブ・ハイで、スティッキー割込みソース RTC_SR3.IC3IRQ をイネーブルし、関与項として CPU への RTC IRQ 割込みラインへファン・インするかどうかを決定します。	
		0	CPU とウェイクアップ・コントローラ両方への RTC 割込みに対する RTC_SR3.IC3IRQ の関与をディスエーブルします。
		1	CPU とウェイクアップ・コントローラ両方への RTC 割込みに関与するファン・イン項としての RTC_SR3.IC3IRQ をイネーブルします。

表 21-16：RTC_CR2IC レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
12 (R/W)	IC2IRQEN	RTC 入力キャプチャ・チャンネル 2 の割込みイネーブル。 このフィールドはアクティブ・ハイで、スティッキー割込みソース RTC_SR3.IC2IRQ をイネーブルし、関与項として CPU への RTC IRQ 割込みラインへファン・インするかどうかを決定します。	
		0	CPU とウェイクアップ・コントローラ両方への RTC 割込みに対する RTC_SR3.IC2IRQ の関与をディスエーブルします。
		1	CPU とウェイクアップ・コントローラ両方への RTC 割込みに関与するファン・イン項としての RTC_SR3.IC2IRQ をイネーブルします。
10 (R/W)	IC0IRQEN	RTC 入力キャプチャ・チャンネル 0 の割込みイネーブル。 このフィールドはアクティブ・ハイで、スティッキー割込みソース RTC_SR3.IC0IRQ をイネーブルし、関与項として CPU への RTC IRQ 割込みラインへファン・インするかどうかを決定します。	
		0	RTC 割込みに対する RTC_SR3.IC0IRQ の関与をディスエーブルします。
		1	RTC 割込みに関与するファン・イン項として RTC_SR3.IC0IRQ をイネーブルします。
9 (R/W)	IC4LH	入力キャプチャ・チャンネル 4 をアクティブにするキャプチャ・エッジの極性。 このフィールドは、RTC_IC4 チャンネルの入力キャプチャ・イベントを、そのチャンネルの GPIO 入力のローからハイへの（LH）遷移として定義するか、またはハイからローへの遷移として定義するかを選択します。	
		0	RTC_IC4 チャンネルは、GPIO ピンのハイからローへの遷移を使用して入力キャプチャ・イベントの信号を送ります。
		1	RTC_IC4 チャンネルは、GPIO ピンのローからハイへの遷移を使用して入力キャプチャ・イベントの信号を送ります。
8 (R/W)	IC3LH	入力キャプチャ・チャンネル 3 をアクティブにするキャプチャ・エッジの極性。 このフィールドは、RTC_IC3 チャンネルの入力キャプチャ・イベントを、そのチャンネルの GPIO 入力のローからハイへの（LH）遷移として定義するか、またはハイからローへの遷移として定義するかを選択します。	
		0	RTC_IC3 チャンネルは、GPIO ピンのハイからローへの遷移を使用して入力キャプチャ・イベントの信号を送ります。
		1	RTC_IC3 チャンネルは、GPIO ピンのローからハイへの遷移を使用して入力キャプチャ・イベントの信号を送ります。

表 21-16：RTC_CR2IC レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
7 (R/W)	IC2LH	入力キャプチャ・チャンネル 2 をアクティブにするキャプチャ・エッジの極性。 このフィールドは、RTC_IC2 チャンネルの入力キャプチャ・イベントを、そのチャンネルの GPIO 入力のローからハイへの（LH）遷移として定義するか、またはハイからローへの遷移として定義するかを選択します。	
		0	RTC_IC2 チャンネルは、GPIO ピンのハイからローへの遷移を使用して入力キャプチャ・イベントの信号を送ります。
		1	RTC_IC2 チャンネルは、GPIO ピンのローからハイへの遷移を使用して入力キャプチャ・イベントの信号を送ります。
5 (R/W)	IC0LH	RTC 入力キャプチャ・チャンネル 0 をアクティブにするキャプチャ・エッジの極性。 このフィールドは、IC0 チャンネルの入力キャプチャ・イベントを、そのチャンネルの GPIO 入力のローからハイへの（LH）遷移として定義するか、またはハイからローへの遷移として定義するかを選択します。	
		0	IC0 チャンネルは、GPIO ピンのハイからローへの遷移を使用して入力キャプチャ・イベントの信号を送ります。
		1	IC0 チャンネルは、GPIO ピンのローからハイへの遷移を使用して入力キャプチャ・イベントの信号を送ります。
4 (R/W)	IC4EN	RTC 入力キャプチャ・チャンネル 4 のイネーブル。 このフィールドはアクティブ・ハイで、入力キャプチャ 4 チャンネル RTC_IC4 のグローバル・イネーブルです。	
		0	16 ビット入力キャプチャ・チャンネル RTC_IC4 をディスエーブルします。
		1	16 ビット入力キャプチャ・チャンネル RTC_IC4 をイネーブルします。
3 (R/W)	IC3EN	RTC 入力キャプチャ・チャンネル 3 のイネーブル。 このフィールドはアクティブ・ハイで、入力キャプチャ 3 チャンネル RTC_IC3 のグローバル・イネーブルです。	
		0	16 ビット入力キャプチャ・チャンネル RTC_IC3 をディスエーブルします。
		1	16 ビット入力キャプチャ・チャンネル RTC_IC3 をイネーブルします。
2 (R/W)	IC2EN	RTC 入力キャプチャ・チャンネル 2 のイネーブル。 このフィールドはアクティブ・ハイで、入力キャプチャ 2 チャンネル RTC_IC2 のグローバル・イネーブルです。	
		0	16 ビット入力キャプチャ・チャンネル RTC_IC2 をディスエーブルします。
		1	16 ビット入力キャプチャ・チャンネル RTC_IC2 をイネーブルします。

表 21-16：RTC_CR2IC レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
0 (R/W)	IC0EN	RTC 入力キャプチャ・チャンネル 0 のイネーブル。 47 ビット・スナップショット・レジスタ RTC_SNAP1、RTC_SNAP0、および RTC_SNAP2 への入力キャプチャに関する GPIO 要求のみに関わる IC0 機能のグローバル・イネーブル（アクティブ・ハイ）。RTC_CR2IC.IC0EN は、CPU に起因するスナップショットには影響しません。CPU が起因のスナップショットは、特別なキー値 0x7627 を RTC_GWY レジスタへ書き込むことによって要求されます。	
		0	47 ビット入力キャプチャ・チャンネル IC0 の外部要求による（GPIO、非 CPU）スナップショットをディスエーブルします。
		1	47 ビット入力キャプチャ・チャンネル IC0 の外部要求による（GPIO、非 CPU）スナップショットをイネーブルします。

SensorStrobe チャンネル設定用の RTC 制御 3

RTC_CR3SS は 16 ビット SensorStrobe チャンネルに関係するイネーブルを設定するコントロール・レジスタです。47 ビット SensorStrobe チャンネル 0 は RTC_CR3SS による制御を受けません。SensorStrobe チャンネル 0 はメイン 47 ビット RTC アラーム（その割込みソースは RTC_SR0.ALMINT）と同義で、RTC_CR0.ALMEN および RTC_CR0.ALMINTEN によって制御されます。RTC_CR3SS は RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

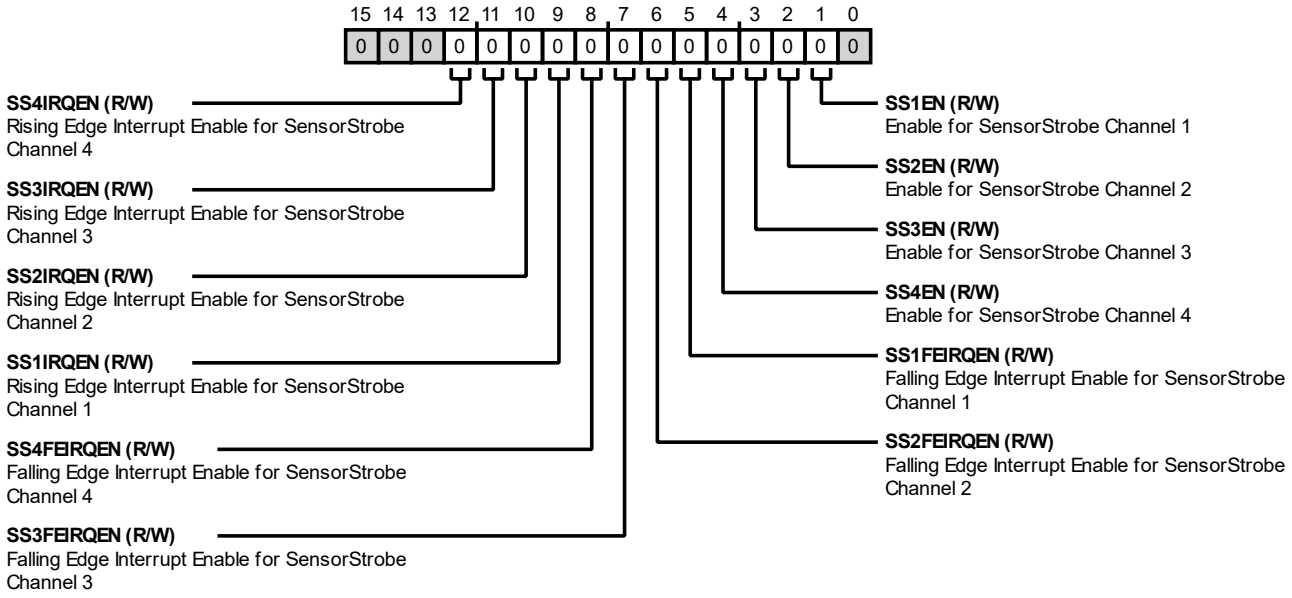


図 21-18：RTC_CR3SS レジスタ図

表 21-17：RTC_CR3SS レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
12 (R/W)	SS4IRQEN	SensorStrobe チャンネル 4 の立上がりエッジ割込みイネーブル。 アクティブ・ハイで、スティッキー割込みソース RTC_SR3.SS4IRQ をイネーブルし、関与項として CPU およびウェイクアップ・コントローラ両方への RTC IRQ 割込みラインへファン・インするかどうかを決定します。	
		0	CPU とウェイクアップ・コントローラ両方への RTC 割込みに対する RTC_SR3.SS4IRQ の関与をディスエーブルします。
		1	CPU とウェイクアップ・コントローラ両方への RTC 割込みに関与するファン・イン項としての RTC_SR3.SS4IRQ をイネーブルします。

表 21-17：RTC_CR3SS レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
11 (R/W)	SS3IRQEN	SensorStrobe チャンネル 3 の立上がりエッジ割込みイネーブル。 アクティブ・ハイで、スティッキー割込みソース RTC_SR3.SS3IRQ をイネーブルし、関与項として CPU およびウェイクアップ・コントローラ両方への RTC IRQ 割込みラインへファン・インするかどうかを決定します。	
		0	CPU とウェイクアップ・コントローラ両方への RTC 割込みに対する RTC_SR3.SS3IRQ の関与をディスエーブルします。
		1	CPU とウェイクアップ・コントローラ両方への RTC 割込みに関与するファン・イン項としての RTC_SR3.SS3IRQ をイネーブルします。
10 (R/W)	SS2IRQEN	SensorStrobe チャンネル 2 の立上がりエッジ割込みイネーブル。 アクティブ・ハイで、スティッキー割込みソース RTC_SR3.SS2IRQ をイネーブルし、関与項として CPU およびウェイクアップ・コントローラ両方への RTC IRQ 割込みラインへファン・インするかどうかを決定します。	
		0	CPU とウェイクアップ・コントローラ両方への RTC 割込みに対する RTC_SR3.SS2IRQ の関与をディスエーブルします。
		1	CPU とウェイクアップ・コントローラ両方への RTC 割込みに関与するファン・イン項としての RTC_SR3.SS2IRQ をイネーブルします。
9 (R/W)	SS1IRQEN	SensorStrobe チャンネル 1 の立上がりエッジ割込みイネーブル。 アクティブ・ハイで、スティッキー割込みソース RTC_SR3.SS1IRQ をイネーブルし、関与項として CPU およびウェイクアップ・コントローラ両方への RTC IRQ 割込みラインへファン・インするかどうかを決定します。	
		0	CPU とウェイクアップ・コントローラ両方への RTC 割込みに対する RTC_SR3.SS1IRQ の関与をディスエーブルします。
		1	CPU とウェイクアップ・コントローラ両方への RTC 割込みに関与するファン・イン項としての RTC_SR3.SS1IRQ をイネーブルします。

表 21-17：RTC_CR3SS レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
8 (R/W)	SS4FEIRQEN	SensorStrobe チャンネル 4 の立下がりエッジ割込みイネーブル。 アクティブ・ハイで、スティッキー割込みソース RTC_SR3.SS4FEIRQ をイネーブルし、関与項として CPU およびウェイクアップ・コントローラ両方への RTC IRQ 割込みラインへファン・インするかどうかを決定します。	
		0	CPU とウェイクアップ・コントローラ両方への RTC 割込みに対する RTC_SR3.SS4FEIRQ の関与をディスエーブルします。
		1	CPU とウェイクアップ・コントローラ両方への RTC 割込みに関与するファン・イン項としての RTC_SR3.SS4FEIRQ をイネーブルします。
7 (R/W)	SS3FEIRQEN	SensorStrobe チャンネル 3 の立下がりエッジ割込みイネーブル。 アクティブ・ハイで、スティッキー割込みソース RTC_SR3.SS3FEIRQ をイネーブルし、関与項として CPU およびウェイクアップ・コントローラ両方への RTC IRQ 割込みラインへファン・インするかどうかを決定します。	
		0	CPU とウェイクアップ・コントローラ両方への RTC 割込みに対する RTC_SR3.SS3FEIRQ の関与をディスエーブルします。
		1	CPU とウェイクアップ・コントローラ両方への RTC 割込みに関与するファン・イン項としての RTC_SR3.SS3FEIRQ をイネーブルします。
6 (R/W)	SS2FEIRQEN	SensorStrobe チャンネル 2 の立下がりエッジ割込みイネーブル。 アクティブ・ハイで、スティッキー割込みソース RTC_SR3.SS2FEIRQ をイネーブルし、関与項として CPU およびウェイクアップ・コントローラ両方への RTC IRQ 割込みラインへファン・インするかどうかを決定します。	
		0	CPU とウェイクアップ・コントローラ両方への RTC 割込みに対する RTC_SR3.SS2FEIRQ の関与をディスエーブルします。
		1	CPU とウェイクアップ・コントローラ両方への RTC 割込みに関与するファン・イン項としての RTC_SR3.SS2FEIRQ をイネーブルします。

表 21-17：RTC_CR3SS レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
5 (R/W)	SS1FEIRQEN	SensorStrobe チャンネル 1 の立下がりエッジ割込みイネーブル。 アクティブ・ハイで、スティッキー割込みソース RTC_SR3.SS1FEIRQ をイネーブルし、関与項として CPU およびウェイクアップ・コントローラ両方への RTC IRQ 割込みラインへファン・インするかどうかを決定します。	
		0	CPU とウェイクアップ・コントローラ両方への RTC 割込みに対する RTC_SR3.SS1FEIRQ の関与をディスエーブルします。
		1	CPU とウェイクアップ・コントローラ両方への RTC 割込みに関与するファン・イン項としての RTC_SR3.SS1FEIRQ をイネーブルします。
4 (R/W)	SS4EN	SensorStrobe チャンネル 4 のイネーブル。 SensorStrobe チャンネル 4（スケジュール・アラーム）RTC_SS4 のアクティブ・ハイのグローバル・イネーブル。	
		0	16 ビット SensorStrobe チャンネル 4 をディスエーブルします。
		1	16 ビット SensorStrobe チャンネル 4 をイネーブルします。
3 (R/W)	SS3EN	SensorStrobe チャンネル 3 のイネーブル。 SensorStrobe チャンネル 3（スケジュール・アラーム）RTC_SS3 のアクティブ・ハイのグローバル・イネーブル。	
		0	16 ビット SensorStrobe チャンネル 3 をディスエーブルします。
		1	16 ビット SensorStrobe チャンネル 3 をイネーブルします。
2 (R/W)	SS2EN	SensorStrobe チャンネル 2 のイネーブル。 SensorStrobe チャンネル 2（スケジュール・アラーム）RTC_SS2 のアクティブ・ハイのグローバル・イネーブル。	
		0	16 ビット SensorStrobe チャンネル 2 をディスエーブルします。
		1	16 ビット SensorStrobe チャンネル 2 をイネーブルします。
1 (R/W)	SS1EN	SensorStrobe チャンネル 1 のイネーブル。 SensorStrobe チャンネル 1（スケジュール・アラーム）RTC_SS1 のアクティブ・ハイのグローバル・イネーブル。	
		0	16 ビット SensorStrobe チャンネル 1 をディスエーブルします。
		1	16 ビット SensorStrobe チャンネル 1 をイネーブルします。

SensorStrobe チャンネル設定用の RTC 制御 4

RTC_CR4SS は 16 ビット SensorStrobe チャンネル RTC_SS1 のマスキングと自動リローディングに関係する有効化を設定するためコントロール・レジスタです。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

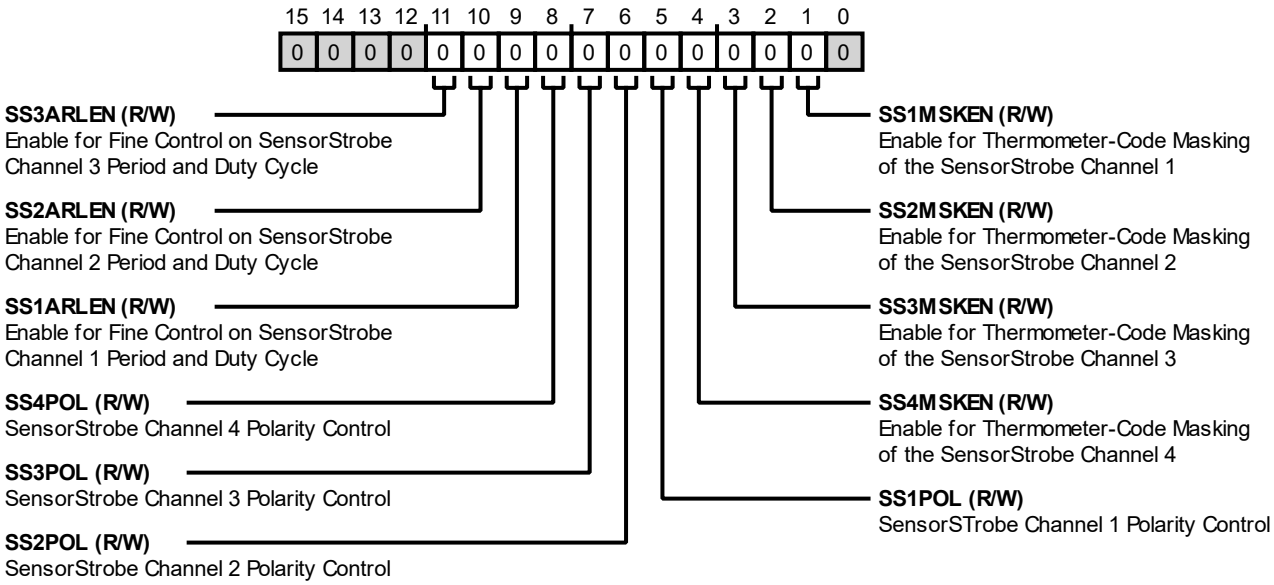


図 21-19：RTC_CR4SS レジスタ図

表 21-18：RTC_CR4SS レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
11 (R/W)	SS3ARLEN	SensorStrobe チャンネル 3 の周期とデューティ・サイクルの精密な制御をイネーブルします。 RTC_SS3HIGHDUR と RTC_SS3LOWDUR を設定して RTC_SS3 SensorStrobe チャンネルのハイ時間とロー時間の精密制御を行うには、このビットをセットします。	
		0	イネーブルした SensorStrobe イベントがそのチャンネルで発生したときは、常に RTC_SS1 の自動リローディングを無効にし、そのターゲット値を維持します。
		1	イネーブルした SensorStrobe イベントがそのチャンネルで発生したときは、常に RTC_SS3LOWDUR および RTC_SS3HIGHDUR からの RTC_SS3 の自動リローディングを有効にします。

表 21-18：RTC_CR4SS レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
10 (R/W)	SS2ARLEN	SensorStrobe チャンネル 2 の周期とデューティ・サイクルの精密な制御をイネーブルします。 RTC_SS2HIGHDUR と RTC_SS2LOWDUR を設定して SensorStrobe チャンネル 2 のハイ時間とロー時間の精密制御を行うには、このビットをセットします。	
		0	イネーブルした SensorStrobe イベントがそのチャンネルで発生したときは、常に RTC_SS1 の自動リローディングを無効にし、そのターゲット値を維持します。
		1	イネーブルした SensorStrobe イベントがそのチャンネルで発生したときは、常に RTC_SS2LOWDUR および RTC_SS2HIGHDUR からの RTC_SS2 の自動リローディングを有効にします。
9 (R/W)	SS1ARLEN	SensorStrobe チャンネル 1 の周期とデューティ・サイクルの精密な制御をイネーブルします。 RTC_SS1HIGHDUR と RTC_SS1LOWDUR を設定して SensorStrobe チャンネル 1 のハイ時間とロー時間の精密制御を行うには、このビットをセットします。	
		0	イネーブルした SensorStrobe イベントがそのチャンネルで発生したときは、常に RTC_SS1 の自動リローディングを無効にし、そのターゲット値を維持します。
		1	イネーブルした SensorStrobe イベントがそのチャンネルで発生したときは、常に RTC_SS1LOWDUR および RTC_SS1HIGHDUR からの RTC_SS1 の自動リローディングを有効にします。
8 (R/W)	SS4POL	SensorStrobe チャンネル 4 の極性制御。 RTC_SS4 の極性を初期化するには、このビットをセットします。	
		0	SensorStrobe チャンネル 4 の極性はローから開始されます。
		1	SensorStrobe チャンネル 4 の極性はハイから開始されます。
7 (R/W)	SS3POL	SensorStrobe チャンネル 3 の極性制御。 RTC_SS3 の極性を初期化するには、このビットをセットします。	
		0	SensorStrobe チャンネル 3 の極性はローから開始されます。
		1	SensorStrobe チャンネル 3 の極性はハイから開始されます。
6 (R/W)	SS2POL	SensorStrobe チャンネル 2 の極性制御。 RTC_SS2 の極性を初期化するには、このビットをセットします。	
		0	SensorStrobe チャンネル 2 の極性はローから開始されます。
		1	SensorStrobe チャンネル 2 の極性はハイから開始されます。

表 21-18：RTC_CR4SS レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
5 (R/W)	SS1POL	SensorStrobe チャンネル 1 の極性制御。 RTC_SS1 の極性を初期化するには、このビットをセットします。	
		0	SensorStrobe チャンネル 1 の極性はローから開始されます。
		1	SensorStrobe チャンネル 1 の極性はハイから開始されます。
4 (R/W)	SS4MSKEN	SensorStrobe チャンネル 4 のサーモメータ・コード・マスキングを有効化。 これは、SensorStrobe チャンネル 4 のスケジュール・アラームをアクティブにするかどうかを決定する際に、RTC_SS4 と RTC カウントのマッチングのマスキングを有効にするために使われます。 このフィールドを介して有効にした場合は、RTC_SSMSK に組み込まれた 1 つの 4 ビット・コードが 16 ビットのサーモメータ・コード・マスクにデコードされて、RTC_SS4 内のビット位置と、RTC_CNT0 および RTC_CNT2 によって与えられる RTC カウントの末尾にある下位 16 個の整数および小数ビットの両方に適用されます。	
		0	16 ビット SensorStrobe チャンネル 4 にマスクを適用しません。
		1	RTC_CR3SS.SS4EN を介してチャンネルがイネーブルされている場合は、サーモメータ・デコードされたマスクを 16 ビット SensorStrobe チャンネル 4 に適用します。
3 (R/W)	SS3MSKEN	SensorStrobe チャンネル 3 のサーモメータ・コード・マスキングを有効化。 これは、SensorStrobe チャンネル 3 のスケジュール・アラームをアクティブにするかどうかを決定する際に、RTC_SS3 と RTC カウントのマッチングのマスキングを有効にするために使われます。 このフィールドを介して有効にした場合は、RTC_SSMSK に組み込まれた 1 つの 4 ビット・コードが 16 ビットのサーモメータ・コード・マスクにデコードされて、RTC_SS3 内のビット位置と、RTC_CNT0 および RTC_CNT2 によって与えられる RTC カウントの末尾にある下位 16 個の整数および小数ビットの両方に適用されます。	
		0	16 ビット SensorStrobe チャンネル 3 にマスクを適用しない。
		1	RTC_CR3SS.SS3EN を介してチャンネルがイネーブルされている場合は、サーモメータ・デコードされたマスクを 16 ビット SensorStrobe チャンネル 3 に適用します。

表 21-18：RTC_CR4SS レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
2 (R/W)	SS2MSKEN	SensorStrobe チャンネル 2 のサーモメータ・コード・マスキングを有効化。 これは、SensorStrobe チャンネル 2 のスケジュール・アラームをアクティブにするかどうかを決定する際に、RTC_SS2 と RTC カウントのマッチングのマスキングを有効にするために使われます。 このフィールドを介して有効にした場合は、RTC_SSMSK に組み込まれた 1 つの 4 ビット・コードが 16 ビットのサーモメータ・コード・マスクにデコードされて、RTC_SS2 内のビット位置と、RTC_CNT0 および RTC_CNT2 によって与えられる RTC カウントの末尾にある下位 16 個の整数および小数ビットの両方に適用されます。	
		0	16 ビット SensorStrobe チャンネル 2 にマスクを適用しません。
		1	RTC_CR3SS.SS2EN を介してチャンネルがイネーブルされている場合は、サーモメータ・デコードされたマスクを 16 ビット SensorStrobe チャンネル 2 に適用します。
1 (R/W)	SS1MSKEN	SensorStrobe チャンネル 1 のサーモメータ・コード・マスキングを有効化。 このフィールドは、SensorStrobe チャンネル 1 のスケジュール・アラームをアクティブにするかどうかを決定する際に、RTC_SS1 と RTC カウントのマッチングのマスキングを有効にするために使われます。 このフィールドを介して有効にした場合は、RTC_SSMSK に組み込まれた 1 つの 4 ビット・コードが 16 ビットのサーモメータ・コード・マスクにデコードされて、RTC_SS1 内のビット位置と、RTC_CNT0 および RTC_CNT2 によって与えられる RTC カウントの末尾にある下位 16 個の整数および小数ビットの両方に適用されます。	
		0	SensorStrobe チャンネル 1 レジスタにマスクを適用しません。
		1	RTC_CR3SS.SS1EN を介してチャンネルがイネーブルされている場合は、サーモメータ・デコードされたマスクを RTC_SS1 に適用します。

SensorStrobe チャンネル GPIO サンプリング設定用の RTC 制御 5

RTC_CR5SSS は、すべての 16 ビット SensorStrobe チャンネルの GPIO サンプリングに関係する有効化を設定するコントロール・レジスタです。RTC_CR5SSS には、各チャンネルの GPIO サンプリングの有効化機能と、チャンネル上のイベントを RTC から CPU およびウェイクアップ・コントローラへの割込みラインに関与させるかどうかに関する有効化機能の両方があります。

47 ビット SensorStrobe チャンネル 0 は RTC_CR5SSS による制御を受けません。SensorStrobe チャンネル 0 はメイン 47 ビット RTC アラーム（その割込みソースは ALMINT）と同義で、RTC_CR0.ALMEN および RTC_CR0.ALMINTEN フィールドによって制御されます。

このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

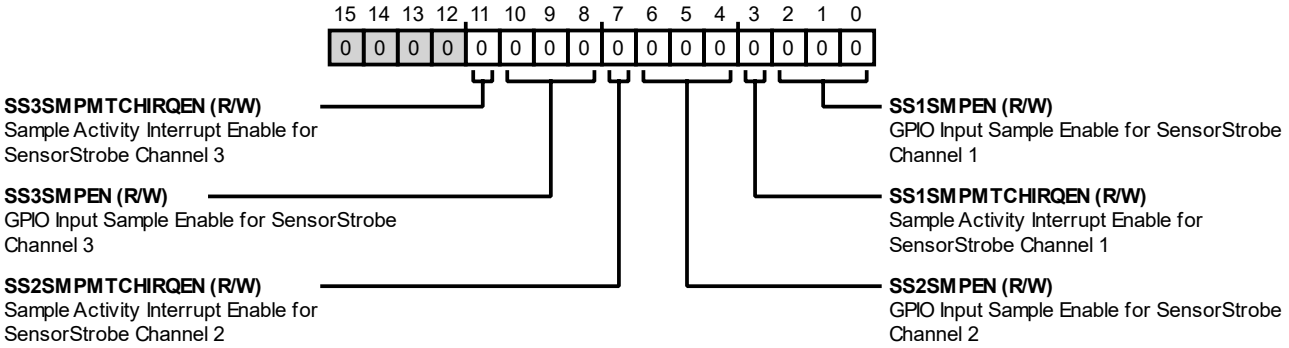


図 21-20：RTC_CR5SSS レジスタ図

表 21-19：RTC_CR5SSS レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
11 (R/W)	SS3SMPMTCHIRQEN	SensorStrobe チャンネル 3 のサンプル・アクティビティ割込みイネーブル。 このビットをセットすると、RTC_SR7.SS3SMPMTCHIRQ がセットされた時点でコアに割込みが送られます。コアに送られる割込みはスティッキー割込みです。
10:8 (R/W)	SS3SMPEN	SensorStrobe チャンネル 3 の GPIO 入力サンプル・イネーブル。 各 SensorStrobe チャンネルには、サンプリング用の専用 GPIO 入力が 3 つあります。このレジスタを設定することによって、これらの GPIO の 1 つまたは複数を無視することができます。この 3 ビット値は、それぞれの GPIO をマスクするために使われます。ビットをローに設定すると、そのビットに対応する入力をサンプリングと比較からマスクできます。
7 (R/W)	SS2SMPMTCHIRQEN	SensorStrobe チャンネル 2 のサンプル・アクティビティ割込みイネーブル。 このビットをセットすると、RTC_SR7.SS2SMPMTCHIRQ がセットされた時点でコアに割込みが送られます。コアに送られる割込みはスティッキー割込みです。

表 21-19：RTC_CR5SSS レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
6:4 (R/W)	SS2SMPEN	SensorStrobe チャンネル 2 の GPIO 入力サンプル・イネーブル。 各 SensorStrobe チャンネルには、サンプリング用の専用 GPIO 入力が 3 つあります。このレジスタを設定することによって、これらの GPIO の 1 つまたは複数を無視することができます。この 3 ビット値は、それぞれの GPIO をマスクするために使われます。ビットをローに設定すると、そのビットに対応する入力をサンプリングと比較からマスクできます。
3 (R/W)	SS1SMPMTCHIRQEN	SensorStrobe チャンネル 1 のサンプル・アクティビティ割込みイネーブル。 このビットをセットすると、`RTC_SR7.SS1SMPMTCHIRQ` がセットされた時点でコアに割込みが送られます。コアに送られる割込みはスティッキー割込みです。
2:0 (R/W)	SS1SMPEN	SensorStrobe チャンネル 1 の GPIO 入力サンプル・イネーブル。 各 SensorStrobe チャンネルには、サンプリング用の専用 GPIO 入力が 3 つあります。このレジスタを設定することによって、これらの GPIO の 1 つまたは複数を無視することができます。この 3 ビット値は、それぞれの GPIO をマスクするために使われます。ビットをローに設定すると、そのビットに対応する入力をサンプリングと比較からマスクできます。

SensorStrobe チャンネル GPIO サンプリング・エッジ設定用の RTC 制御 6

RTC_CR6SSS は、16 ビット SensorStrobe チャンネルすべてのアクティビティに関する GPIO サンプリング・エッジを設定するコントロール・レジスタです。RTC_CR6SSS には、SensorStrobe チャンネルのすべてについて、立上がりエッジまたは立下がりエッジ、もしくはその両方でサンプリングを行うよう設定するための有効化機能があります。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

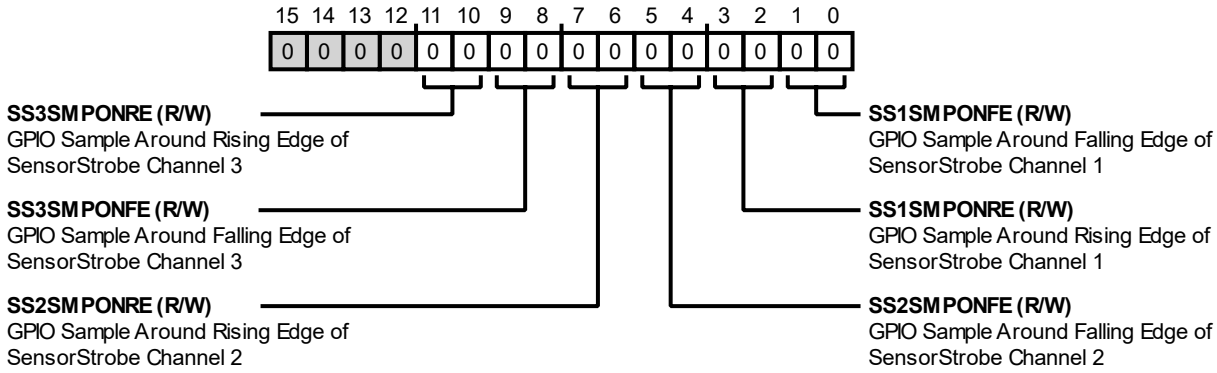


図 21-21：RTC_CR6SSS レジスタ図

表 21-20：RTC_CR6SSS レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
11:10 (R/W)	SS3SMPONRE	SensorStrobe チャンネル 3 の立上がりエッジでの GPIO サンプル。 SensorStrobe チャンネル 3 パルスの立上がりエッジによる SensorStrobe GPIO 入力のサンプリングを有効にします。サンプリングしたデータは、RTC_SR7.SS3SMP レジスタからリードバックできます。新しいサンプルが取得されると、このデータは上書きされます。	
		0	立上がりエッジでのサンプリングを行いません。
		1	入力は SensorStrobe チャンネル 3 の立上がりエッジの 1 クロック・サイクル前でサンプルされます。
		2	入力は SensorStrobe チャンネル 3 の立上がりエッジでサンプルされます。
		3	入力は、SensorStrobe チャンネル 3 の立上がりエッジの 1 クロック・サイクル後でサンプルされます。

表 21-20：RTC_CR6SSS レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
9:8 (R/W)	SS3SMPONFE	SensorStrobe チャンネル 3 の立下がりエッジでの GPIO サンプル。 SensorStrobe チャンネル 3 パルスの立下がりエッジによる SensorStrobe GPIO 入力のサンプリングを有効にします。サンプリングしたデータは、RTC_SR7.SS3SMP レジスタからリードバックできます。新しいサンプルが取得されると、このデータは上書きされます。	
		0	立下がりエッジでのサンプリングを行いません。
		1	入力は SensorStrobe チャンネル 3 の立下がりエッジの 1 クロック・サイクル前でサンプルされます。
		2	入力は SensorStrobe チャンネル 3 の立下がりエッジでサンプルされます。
		3	入力は SensorStrobe チャンネル 3 の立下がりエッジの 1 クロック・サイクル後でサンプルされます。
7:6 (R/W)	SS2SMPONRE	SensorStrobe チャンネル 2 の立上がりエッジでの GPIO サンプル。 SensorStrobe チャンネル 2 パルスの立上がりエッジによる SensorStrobe GPIO 入力のサンプリングを有効にします。サンプリングしたデータは、RTC_SR7.SS2SMP レジスタからリードバックできます。新しいサンプルが取得されると、このデータは上書きされます。	
		0	立上がりエッジでのサンプリングを行いません。
		1	入力は SensorStrobe チャンネル 2 の立上がりエッジの 1 クロック・サイクル前でサンプルされます。
		2	入力は SensorStrobe チャンネル 2 の立上がりエッジでサンプルされます。
		3	入力は、SensorStrobe チャンネル 2 の立上がりエッジの 1 クロック・サイクル後でサンプルされます。
5:4 (R/W)	SS2SMPONFE	SensorStrobe チャンネル 2 の立下がりエッジでの GPIO サンプル。 SensorStrobe チャンネル 2 パルスの立下がりエッジによる SensorStrobe GPIO 入力のサンプリングを有効にします。サンプリングしたデータは、RTC_SR7.SS2SMP レジスタからリードバックできます。新しいサンプルが取得されると、このデータは上書きされます。	
		0	立下がりエッジでのサンプリングを行いません。
		1	入力は SensorStrobe チャンネル 2 の立下がりエッジの 1 クロック・サイクル前でサンプルされます。
		2	入力は SensorStrobe チャンネル 2 の立下がりエッジでサンプルされます。
		3	入力は SensorStrobe チャンネル 2 の立下がりエッジの 1 クロック・サイクル後でサンプルされます。

表 21-20：RTC_CR6SSS レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
3:2 (R/W)	SS1SMPONRE	SensorStrobe チャンネル 1 の立上がりエッジでの GPIO サンプル。 SensorStrobe パルスの立上がりエッジによる SensorStrobe GPIO 入力のサンプリングを有効にします。サンプリングしたデータは、RTC_SR7.SS1SMP レジスタからリードバックできます。新しいサンプルが取得されると、このデータは上書きされます。	
		0	立上がりエッジでのサンプリングを行いません。
		1	入力は SensorStrobe チャンネル 1 の立上がりエッジの 1 クロック・サイクル前でサンプルされます。
		2	入力は SensorStrobe チャンネル 1 の立上がりエッジでサンプルされます。
		3	入力は、SensorStrobe チャンネル 1 の立上がりエッジの 1 クロック・サイクル後でサンプルされます。
1:0 (R/W)	SS1SMPONFE	SensorStrobe チャンネル 1 の立下がりエッジでの GPIO サンプル。 SensorStrobe パルスの立下がりエッジによる SensorStrobe GPIO 入力のサンプリングを有効にします。サンプリングしたデータは、RTC_SR7.SS1SMP レジスタからリードバックできます。新しいサンプルが取得されると、このデータは上書きされます。	
		0	立下がりエッジでのサンプリングを行いません。
		1	入力は SensorStrobe チャンネル 1 の立下がりエッジの 1 クロック・サイクル前でサンプルされます。
		2	入力は SensorStrobe チャンネル 1 の立下がりエッジでサンプルされます。
		3	入力は SensorStrobe チャンネル 1 の立下がりエッジの 1 クロック・サイクル後でサンプルされます。

SensorStrobe チャンネル GPIO サンプリング・アクティビティ設定用の RTC 制御 7

RTC_CR7SSS は、コアに割込みを送る GPIO アクティビティを設定するコントロール・レジスタです。GPIO アクティビティとは、変化する入力または変化しない入力、あるいは予想値と一致する入力または予想値と一致しない入力のことを意味します。このアクティビティは RTC_SR7 レジスタに記録され、オプションでコアへの割込みとして送信することができます。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

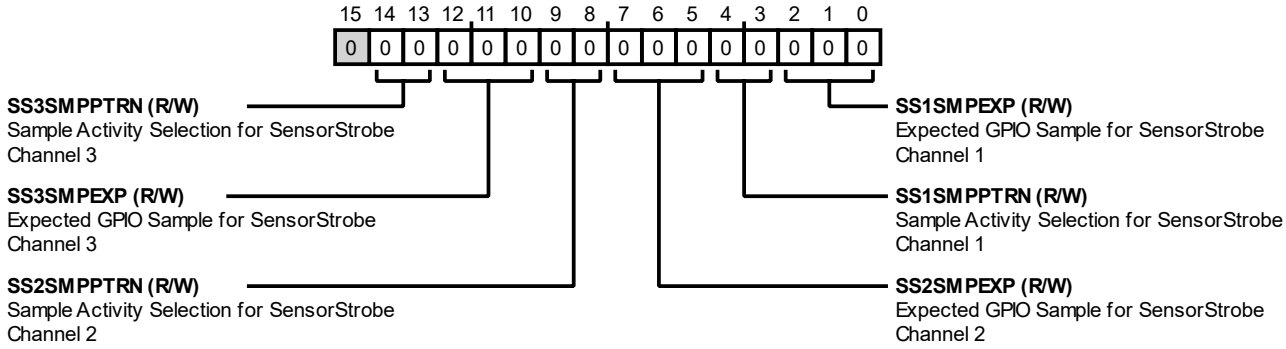


図 21-22：RTC_CR7SSS レジスタ図

表 21-21：RTC_CR7SSS レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
14:13 (R/W)	SS3SMPPTRN	SensorStrobe チャンネル 3 のサンプル・アクティビティ選択。 このフィールドは、予想パターン RTC_CR7SSS.SS3SMPEXP またはサンプル値の変化に基づいてステータス・ビット RTC_SR7.SS3SMPMTCHIRQ をセットするために使われます。	
		0	現在の GPIO サンプルは 1 つ前のサンプルと異なります。
		1	現在の GPIO サンプルは 1 つ前のサンプルと同じです。
		2	現在の GPIO サンプルは予想サンプルと同じです。
		3	現在の GPIO サンプルは予想サンプルと異なります。
12:10 (R/W)	SS3SMPEXP	SensorStrobe チャンネル 3 の予想 GPIO サンプル。 RTC_SR7.SS3SMP が RTC_CR7SSS.SS3SMPPTRN に従って RTC_CR7SSS.SS3SMPEXP と一致する場合は、RTC_SR7.SS3SMPMTCHIRQ がセットされます。このレジスタは、パターン・マッチング用の予想 GPIO 値を提供します。	

表 21-21：RTC_CR7SSS レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
9:8 (R/W)	SS2SMPPTRN	SensorStrobe チャンネル 2 のサンプル・アクティビティ選択。 このフィールドは、予想パターン RTC_CR7SSS.SS2SMPEXP またはサンプル値の変化に基づいてステータス・ビット RTC_SR7.SS2SMPMTCHIRQ をセットするために使われます。	
		0	現在の GPIO サンプルは 1 つ前のサンプルと異なります。
		1	現在の GPIO サンプルは 1 つ前のサンプルと同じです。
		2	現在の GPIO サンプルは予想サンプルと同じです。
		3	現在の GPIO サンプルは予想サンプルと異なります。
7:5 (R/W)	SS2SMPEXP	SensorStrobe チャンネル 2 の予想 GPIO サンプル。 RTC_SR7.SS2SMP が RTC_CR7SSS.SS2SMPPTRN に従って RTC_CR7SSS.SS2SMPEXP と一致する場合は、RTC_SR7.SS2SMPMTCHIRQ がセットされます。このレジスタは、パターン・マッチング用の予想 GPIO 値を提供します。	
4:3 (R/W)	SS1SMPPTRN	SensorStrobe チャンネル 1 のサンプル・アクティビティ選択。 このフィールドは、予想パターン RTC_CR7SSS.SS1SMPEXP またはサンプル値の変化に基づいてステータス・ビット RTC_SR7.SS1SMPMTCHIRQ をセットするために使われます。	
		0	現在の GPIO サンプルは 1 つ前のサンプルと異なります。
		1	現在の GPIO サンプルは 1 つ前のサンプルと同じです。
		2	現在の GPIO サンプルは予想サンプルと同じです。
		3	現在の GPIO サンプルは予想サンプルと異なります。
2:0 (R/W)	SS1SMPEXP	SensorStrobe チャンネル 1 の予想 GPIO サンプル。 RTC_SR7.SS1SMP が RTC_CR7SSS.SS1SMPPTRN に従って RTC_CR7SSS.SS1SMPEXP と一致する場合は、RTC_SR7.SS1SMPMTCHIRQ がセットされます。このレジスタは、パターン・マッチング用の予想 GPIO 値を提供します。	

RTC フリーズ・カウント

RTC フリーズ・カウント MMR を使用すれば、RTC_CNT2、RTC_CNT1、および RTC_CNT0 に格納された 47 ビット RTC カウントのコヒーレントなトリプル 16 ビット読出しを行うことができます。

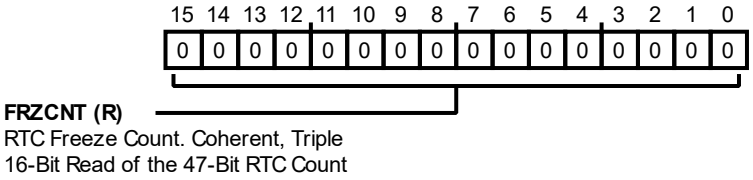


図 21-23：RTC_FRZCNT レジスタ図

表 21-22：RTC_FRZCNT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	FRZCNT	RTC フリーズ・カウント。47 ビット RTC カウントのコヒーレントなトリプル 16 ビット読出し。 このフィールドは常に 3 つの読出しのシーケンスで行われ、シーケンス内の最初の読出しが RTC_CNT0 の最新の値を返します。 同時に、この最初の読出しで RTC_CNT1 と RTC_CNT2 の値のスナップショットが取得されてフリーズされ、RTC_FRZCNT.FRZCNT のシーケンスの 2 回目と 3 回目の読出しでそれぞれ RTC_CNT1 と RTC_CNT2 のスナップショット値が返されます。これら 3 つの読出しのシーケンス番号は、RTC_SR6.FRZCNTPTR を介して知ることができます。

RTC GPIO ピン・マルチプレクサ・コントロール・レジスタ 0

RTC_GPMUX0 は、SensorStrobe チャンネルによってサンプリングされるデータとして GPIO（ピン）を選択するためのコントロール・レジスタです。このレジスタは、該当する SensorStrobe チャンネルがディスエーブルされている場合にのみ更新してください。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

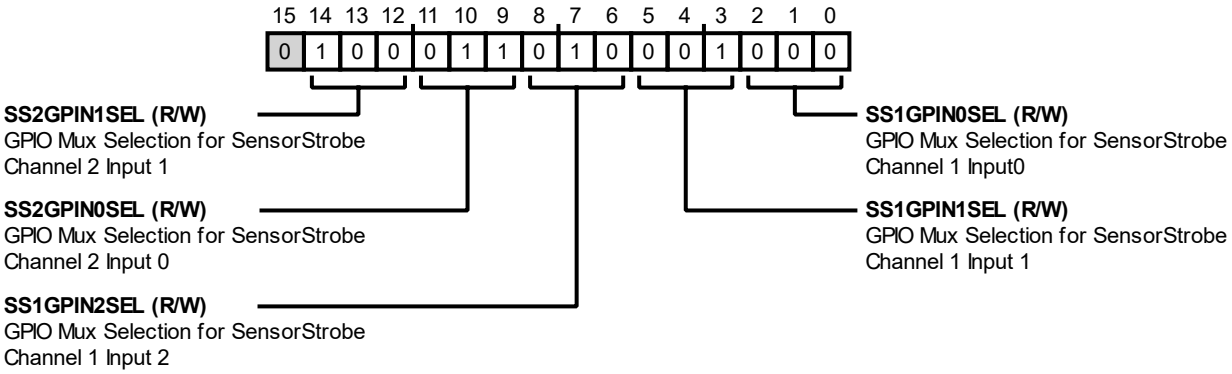


図 21-24：RTC_GPMUX0 レジスタ図

表 21-23：RTC_GPMUX0 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
14:12 (R/W)	SS2GPIN1SEL	SensorStrobe チャンネル 2 入力 1 の GPIO マルチプレクサ選択。 サンプリングされる SensorStrobe チャンネル 2 の GPIO 入力は、 GPIO［RTC_GPMUX0.SS2GPIN1SEL］です。
11:9 (R/W)	SS2GPIN0SEL	SensorStrobe チャンネル 2 入力 0 の GPIO マルチプレクサ選択。 サンプリングされる SensorStrobe チャンネル 2 の GPIO 入力は、 GPIO［RTC_GPMUX0.SS2GPIN0SEL］です。
8:6 (R/W)	SS1GPIN2SEL	SensorStrobe チャンネル 1 入力 2 の GPIO マルチプレクサ選択。 サンプリングされる SensorStrobe チャンネル 1 の GPIO 入力は、 GPIO［RTC_GPMUX0.SS1GPIN2SEL］です。
5:3 (R/W)	SS1GPIN1SEL	SensorStrobe チャンネル 1 入力 1 の GPIO マルチプレクサ選択。 サンプリングされる SensorStrobe チャンネル 1 の GPIO 入力は、 GPIO［RTC_GPMUX0.SS1GPIN1SEL］です。
2:0 (R/W)	SS1GPIN0SEL	SensorStrobe チャンネル 1 入力 0 の GPIO マルチプレクサ選択。 サンプリングされる SensorStrobe チャンネル 1 の GPIO 入力は、 GPIO［RTC_GPMUX0.SS1GPIN0SEL］です。

RTC GPIO ピン・マルチプレクサ・コントロール・レジスタ 1

RTC_GPMUX1 は、SensorStrobe チャンネルによってサンプリングされるデータとして GPIO（ピン）を選択するためのコントロール・レジスタです。このレジスタは、該当する SensorStrobe チャンネルがディスエーブルされている場合にのみ更新してください。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

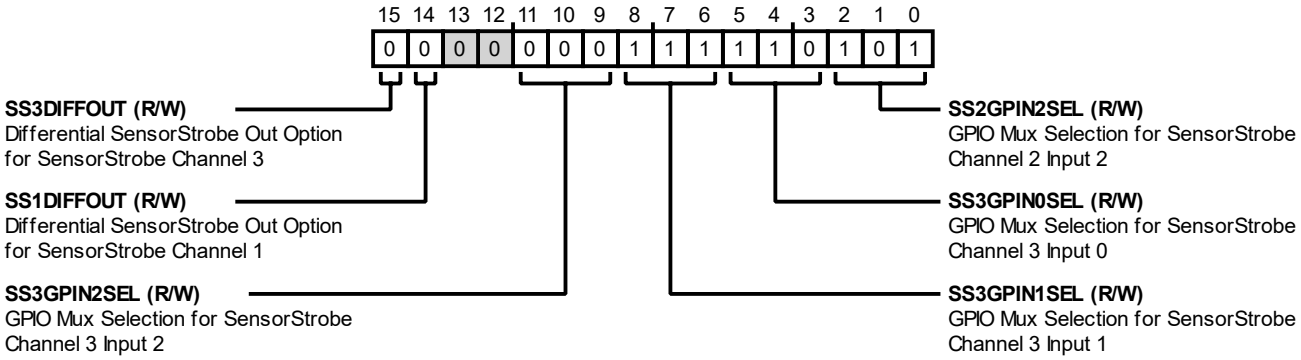


図 21-25：RTC_GPMUX1 レジスタ図

表 21-24：RTC_GPMUX1 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15 (R/W)	SS3DIFFOUT	SensorStrobe チャンネル 3 の差動 SensorStrobe 出力オプション。 SensorStrobe チャンネル 3 は差動信号として使われ、このチャンネルの実際の RTC_SS3 出力は対応する GPIO から得られます。RTC_SS4 は、提供された RTC_SS3 の反転信号に対して使われます。
14 (R/W)	SS1DIFFOUT	SensorStrobe チャンネル 1 の差動 SensorStrobe 出力オプション。 SensorStrobe チャンネル 1 は差動信号として使われ、このチャンネルの実際の RTC_SS1 出力は対応する GPIO から得られます。SensorStrobe チャンネル 2 の RTC_SS2 は、提供された RTC_SS1 の反転信号に対して使われます。
11:9 (R/W)	SS3GPIN2SEL	SensorStrobe チャンネル 3 入力 2 の GPIO マルチプレクサ選択。 サンプリングされる SensorStrobe チャンネル 3 の GPIO 入力は、GPIO［RTC_GPMUX1.SS3GPIN2SEL］です。
8:6 (R/W)	SS3GPIN1SEL	SensorStrobe チャンネル 3 入力 1 の GPIO マルチプレクサ選択。 サンプリングされる SensorStrobe チャンネル 3 の GPIO 入力は、GPIO［RTC_GPMUX1.SS3GPIN1SEL］です。
5:3 (R/W)	SS3GPIN0SEL	SensorStrobe チャンネル 3 入力 0 の GPIO マルチプレクサ選択。 サンプリングされる SensorStrobe チャンネル 3 の GPIO 入力は、GPIO［RTC_GPMUX1.SS3GPIN0SEL］です。
2:0 (R/W)	SS2GPIN2SEL	SensorStrobe チャンネル 2 入力 2 の GPIO マルチプレクサ選択。 サンプリングされる SensorStrobe チャンネル 2 の GPIO 入力は、GPIO［RTC_GPMUX1.SS2GPIN2SEL］です。

RTC ゲートウェイ

RTC_GWY はゲートウェイ MMR のアドレスで、CPU は、RTC 内で行うべき動作をこのアドレスを通じて指示することができます。CPU は、RTC_GWY に特定のキーを書き込むことによってこれを行います。RTC_GWY はすべてゼロとしてリードバックされます。

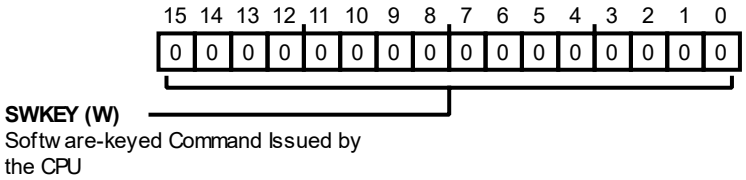


図 21-26：RTC_GWY レジスタ図

表 21-25：RTC_GWY レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (RX/W)	SWKEY	CPU により発行されるソフトウェア・キー・コマンド。 RTC_GWY レジスタは、CPU が RTC に対して発行するソフトウェア・キー・コマンドをアクティブにするための書込みターゲットです。使用可能なキーは次のとおりです。 (i) 値が 0xa2c5 の FLUSH_RTC ソフトウェア・キー（RTC_GWY へのレジスタ書込みを介して供給）を使用すると、RTC はポストされたすべての書込みトランザクションをフラッシュ（廃棄）して、現在実行中のすべてのトランザクションを停止します。 (ii) 値が 0x7627 の SNAPSHOT_RTC キー（RTC_GWY へのレジスタ書込みを介して供給）を使用すると、RTC は RTC_CNT1、RTC_CNT0、および RTC_CNT2 の値のスティッキー・スナップショットを取得して、それを RTC_SNAP1、RTC_SNAP0、および RTC_SNAP2 に保存します。 (iii) RTC_GWY への書込みによって得られる値 0x9376 のキーは、RTC_SR6.FRZCNTPTR ポインタをゼロにするので、RTC_FRZCNT 読出し用の 0-1-2 シーケンスを再初期化します。

RTC 入力キャプチャ・チャンネル 2

RTC_IC2 は {integer_bits, fractional_bits} の下位 16 ビットの読出し専用スナップショットで、これは入力キャプチャ・チャンネル 2 の最新イベントにおけるメインの 47 ビット RTC カウントを意味します。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

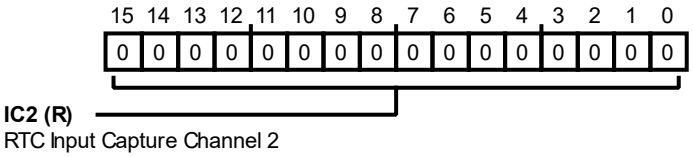


図 21-27：RTC_IC2 レジスタ図

表 21-26：RTC_IC2 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	IC2	RTC 入力キャプチャ・チャンネル 2。 RTC 時間のスナップショットは RTC_IC2.IC2 へのプリスケーリングによって決定されるアライメントで配置されますが、これは、入力キャプチャ・チャンネル 2 の RTC への外部要求入力によって行われます。このようなイベントは、それ以前に RTC_IC2.IC2 にキャプチャされたあらゆる値を上書きします（RTC_CR2IC.ICOWUSEN によって許可されている場合）。

RTC 入力キャプチャ・チャンネル 3

RTC_IC3 は {integer_bits, fractional_bits} の下位 16 ビットの読出し専用スナップショットで、これは入力キャプチャ・チャンネル 3 の最新イベントにおけるメインの 47 ビット RTC カウントを意味します。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

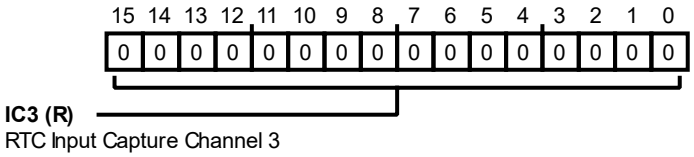


図 21-28：RTC_IC3 レジスタ図

表 21-27：RTC_IC3 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	IC3	RTC 入力キャプチャ・チャンネル 3。 RTC 時間のスナップショットは RTC_IC3.IC3 へのプリスケーリングによって決定されるアライメントで配置されますが、これは、入力キャプチャ・チャンネル 3 の RTC への外部要求入力によって行われます。このようなイベントは、それ以前に RTC_IC3.IC3 にキャプチャされたあらゆる値を上書きします（RTC_CR2IC.ICOWUSEN によって許可されている場合）。

RTC 入力キャプチャ・チャンネル 4

RTC_IC4 は {integer_bits, fractional_bits} の下位 16 ビットの読出し専用スナップショットで、これは入力キャプチャ・チャンネル 4 の最新イベントにおけるメインの 47 ビット RTC カウントを意味します。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

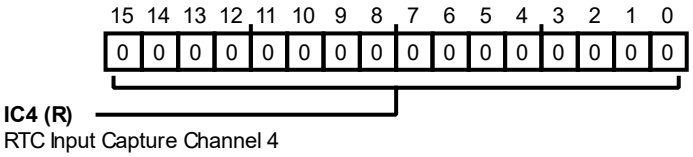


図 21-29：RTC_IC4 レジスタ図

表 21-28：RTC_IC4 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	IC4	RTC 入力キャプチャ・チャンネル 4。 RTC 時間のスナップショットは RTC_IC4.IC4 へのプリスケーリングによって決定されるアライメントで配置されますが、これは、入力キャプチャ・チャンネル 4 の RTC への外部要求入力によって行われます。このようなイベントは、それ以前に RTC_IC4.IC4 にキャプチャされたあらゆる値を上書きします（RTC_CR2IC.ICOWUSEN によって許可されている場合）。

RTC モジュロ

RTC_MOD は読出し専用レジスタで、RTC_CNT1 と RTC_CNT0 のカウント値のモジュロ 60 に相当する RTC_MOD.CNTMOD60 を使用できるようにします。このモジュロ 60 値は、最後に発生したモジュロ 60 繰り越しイベント後のプリスケール RTC 時間単位の変位に等しくなります。繰り越しはモジュロ 60 境界と同義です。

境界は以下の要領で定義されます。RTC は、以下のいずれかのイベントが発生した場合は常に RTC 自体を再アラインして、モジュロ 60 境界とモジュロ 1 境界を同時に作成します。

(i) CPU が RTC_CNT1 レジスタと RTC_CNT0 レジスタに新しい値ペアを書き込んで経過時間単位カウントを再定義する一方で、RTC がイネーブルされ、このポストされたツイン書込みが続いて実行される。

(ii) CPU が、RTC_CR0.CNTEN フィールドを使ってディスエーブルされた RTC をイネーブルする。

(iii) RTC_CR0.CNTEN によって RTC がイネーブルされている場合に、RTC_CR1.PRESCALE2EXP フィールドを介して RTC のプリスケール度が変更される。

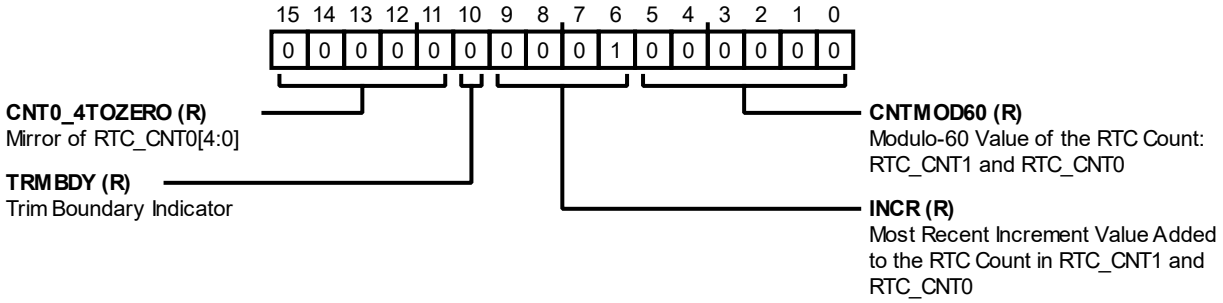


図 21-30：RTC_MOD レジスタ図

表 21-29：RTC_MOD レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:11 (R/NW)	CNT0_4TOZERO	RTC_CNT0［4：0］のミラー。 RTC_MOD.CNT0_4TOZERO は RTC_CNT0［4：0］のミラーで、RTC_MOD.CNTMOD60 フィールドとの同時リードバックに使用できます。 同時にこのミラーを使用できるようにすると、モジュロ 60 と絶対 RTC カウントの関係をより良く理解し、デバッグすることが可能になります。

表 21-29：RTC_MOD レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
10 (R/NW)	TRMBDY	トリム境界インジケータ。 最後に行われた RTC カウントのインクリメントが、カウント値のトリミングと同時に行われたことを示す、トリム境界インジケータ。	
		0	最後に行われた RTC カウントのインクリメント時に、トリミングが行われませんでした。
		1	最後に行われた RTC カウントのインクリメント時に、トリミングが行われました。
9:6 (R/NW)	INCR	RTC_CNT1 と RTC_CNT0 の RC カウントに最後に加算されたインクリメント値。 RTC_MOD.INCR は、RTC カウントを最後にインクリメントした読出し専用値です。通常の状況では、RTC がイネーブルされるとこの値は 1 になります。しかしトリミングを行う場合、RTC_MOD.INCR は、RTC_TRM レジスタのトリム設定に応じて 0 から 8 までの任意の整数値を取り得ます。	
5:0 (R/NW)	CNTMOD60	RTC カウント、RTC_CNT1 と RTC_CNT0 のモジュロ 60 値。 RTC_MOD.CNTMOD60 は 0 から 59 までカウントされ、その後は繰り越されて再び 0 になります。この値は、RTC_CNT1、RTC_CNT0、および RTC_CNT2 内のメイン RTC カウントと並行して増加（およびトリム）されます。 以下のいずれかのイベントが発生すると、RTC_MOD.CNTMOD60 は常にゼロに設定されます。 (i) プリスケールされた時間単位使用時に、値 59 から行われる通常の繰り越し。 (ii) CPU が RTC_CNT1 レジスタと RTC_CNT0 レジスタに新しい値ペアを書き込んで経過時間カウントを再定義する一方で、RTC がイネーブルされ、このポストされたツイン書込みが実行される。 (iii) CPU が、RTC_CR0.CNTEN フィールドを使ってディスエーブルされた RTC をイネーブルする。 (iv) RTC_CR0.CNTEN を介して RTC がイネーブルされ、RTC_CR1.PRESCALE2EXP フィールドを介して RTC のプリスケール度が変更される。 このビット・フィールドは RTC1 にのみ存在します。RTC0 ではこのフィールドは予備で、すべてゼロとして読み出されます。	

RTC SensorStrobe チャンネル 1

RTC_SS1 は {integer_bits, fractional_bits} の下位 16 ビットに関する SensorStrobe チャンネル 1 のスケジュール・アラーム時間で、これはメインの 47 ビット RTC カウントを意味します。 これらの 16 ビットの位置より上位にあるメイン RTC カウントのビットは、16 ビット RTC_SS1 SensorStrobe チャンネルに関してはドント・ケアです。

このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

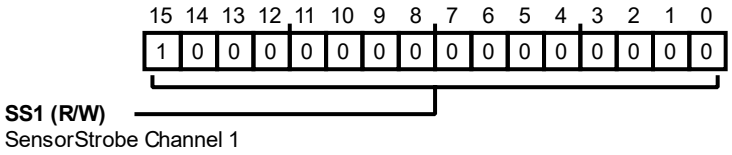


図 21-31：RTC_SS1 レジスタ図

表 21-30：RTC_SS1 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	SS1	SensorStrobe チャンネル 1。 SensorStrobe チャンネル 1 のスケジュール・アラーム・ターゲット時間で、オプションの自動リローディングおよびマスキング機能を備えています。 SensorStrobe と RTC カウントが一致すると、RTC_SS1.SS1 の値が維持されるか、別の 16 ビット・レジスタから自動リロードされます。この更新は次のように表されます。 RTC_SS1.SS1 = RTC_CR4SS.SS1ARLEN? RTC_SS1.SS1 + ((RTC_SS1? RTC_SS1HIGHDUR：RTC_SS1LOWDUR) ：RTC_SS1.SS1

SensorStrobe チャンネル 1 の RTC 自動リロード・ハイ時間

RTC_SS1HIGHDUR には 16 ビットのリロード値が格納されます。有効にされた SensorStrobe イベントがそのチャンネル上で発生すると、この値はオプションで（RTC_CR4SS.SS1ARLEN で有効化）RTC_SS1 の積算値に加えられ、RTC_SS1TGT レジスタで確認することができます。RTC_SS1HIGHDUR を使用すると、2 のべき乗またはそれ以外の値を周期とする RTC_SS1 用の繰り返しアラームを有効にすることができます。リローディングが有効でない場合、RTC_SS1 と RTC_SS1TGT のリードバック値は同じです。つまり、RTC_SS1 内にある SensorStrobe の開始値です（有効でないのでリロードされません）。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

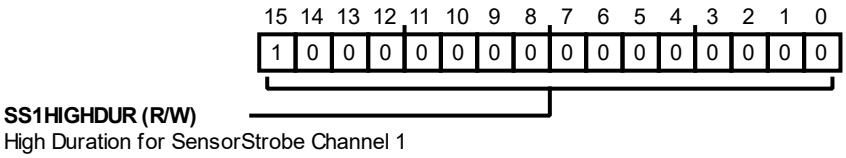


図 21-32：RTC_SS1HIGHDUR レジスタ図

表 21-31：RTC_SS1HIGHDUR レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	SS1HIGHDUR	SensorStrobe チャンネル 1 のハイ時間。 RTC_SS1 の自動リローディングが RTC_CR4SS.SS1ARLEN を介して有効にされている場合は、RTC_SS1HIGHDUR の内容が SensorStrobe チャンネル 1 のハイ時間を制御します。RTC_SS1TGT の初期値は RTC_SS1 です。 これは次のように表されます。 RTC_SS1TGT = RTC_CR4SS.SS1ARLEN?（RTC_SS1TGT +（SensorStrobe channel 1 output level ? RTC_SS1LOWDUR：RTC_SS1HIGHDUR））：RTC_SS1.SS1 SensorStrobe チャンネル 1 は、RTC タイマーの値が RTC_SS1TGT に達して、RTC_SS1TGT の新しい値が上述のように更新されると、トグルします。 このような状況においては、RTC_SS1TGT が、値 RTC_SS1LOWDUR と RTC_SS1HIGHDUR を交互にロードする反復アラームとしての役割を果たし、この際、リロード値内のマスクされていないビットが、現在時間から次の SensorStrobe アラームまでのステップ・サイズ（オフセット）を実質的に定義します。

SensorStrobe チャンネル 1 の RTC 自動リロード・ロー時間

RTC_SS1LOWDUR には 16 ビットのリロード値が格納されます。有効にされた SensorStrobe イベントがそのチャンネル上で発生すると、この値はオプションで（RTC_CR4SS.SS1ARLEN で有効化）RTC_SS1 の積算値に加えられ、RTC_SS1TGT レジスタで確認することができます。

RTC_SS1LOWDUR を使用すると、2 のべき乗またはそれ以外の値を周期とする RTC_SS1 用の繰返しアラームを有効にすることができます。

リローディングが有効でない場合、RTC_SS1 と RTC_SS1TGT のリードバック値は同じです。つまり、RTC_SS1 内にある SensorStrobe の開始値です（有効でないのでリロードされません）。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

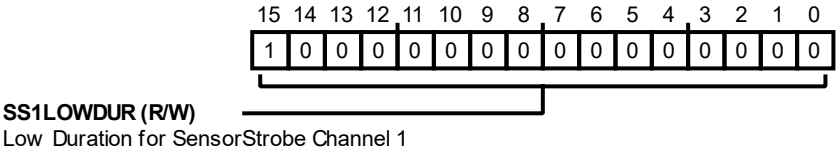


図 21-33：RTC_SS1LOWDUR レジスタ図

表 21-32：RTC_SS1LOWDUR レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	SS1LOWDUR	SensorStrobe チャンネル 1 のロー時間。 RTC_SS1 の自動リローディングが RTC_CR4SS.SS1ARLEN を介して有効にされている場合は、RTC_SS1LOWDUR の内容が RTC_SS1 の SensorStrobe チャンネルのロー時間を制御します。RTC_SS1TGT の初期値は RTC_SS1 です。 これは次のように表されます。 RTC_SS1TGT = RTC_CR4SS.SS1ARLEN?（RTC_SS1TGT +（SensorStrobe channel 1 level ? RTC_SS1LOWDUR：RTC_SS1HIGHDUR））：RTC_SS1.SS1 SensorStrobe チャンネル 1 出力は、RTC タイマーの値が RTC_SS1TGT に達して、RTC_SS1TGT の新しい値が上述のように更新されると、トグルします。 このような状況においては、RTC_SS1TGT が、値 RTC_SS1LOWDUR と RTC_SS1HIGHDUR を交互にロードする反復アラームとしての役割を果たし、この際、リロード値内のマスクされていないビットが、現在時間から次の SensorStrobe アラームまでのステップ・サイズ（オフセット）を実質的に定義します。

RTC SensorStrobe チャンネル 1 のターゲット

SensorStrobe チャンネル 1 の現在の累積ターゲット・アラーム時間を反映します。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

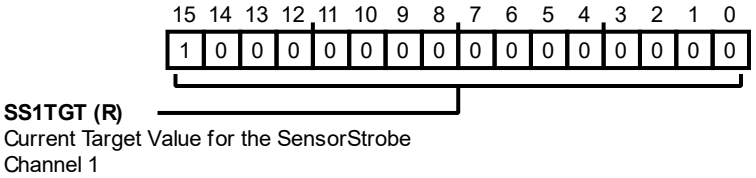


図 21-34：RTC_SS1TGT レジスタ図

表 21-33：RTC_SS1TGT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	SS1TGT	SensorStrobe チャンネル 1 の現在のターゲット値。 考え得るすべての自動リローディングを考慮に入れた上で、SensorStrobe チャンネル 1 の現在のターゲット値を CPU が確認できるようにします。 RTC_CR4SS.SS1ARLEN によって自動リローディングが無効にされている場合、このフィールドによって返される値は RTC_SS1 レジスタの開始値と同じです。 自動リローディングが有効な場合、このフィールドをリードバックすると、SensorStrobe チャンネル 1 の現在の累積ターゲット値が返されます。この値は、RTC_SS1 レジスタ内に格納された値から SensorStrobe シーケンスを開始してリロードした値です。つまり、SensorStrobe イベントの発生ごとに新しいターゲット時間（RTC_SS1LOWDUR と RTC_SS1HIGHDUR の値によりオフセット）を加算して累積したものです。

RTC SensorStrobe チャンネル 2

このレジスタは {integer_bits, fractional_bits} の下位 16 ビットに関する SensorStrobe チャンネル 2 のスケジュール・アラーム時間で、これはメインの 47 ビット RTC カウントを意味します。これらの 16 ビットの位置より上位にあるメイン RTC カウントのビットは、16 ビット SensorStrobe チャンネル 2 に関してはドント・ケアです。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

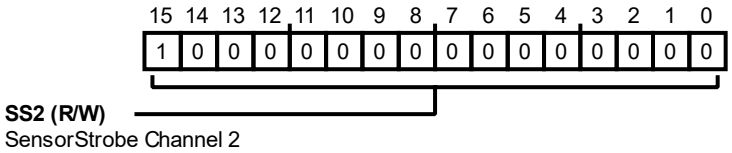


図 21-35：RTC_SS2 レジスタ図

表 21-34：RTC_SS2 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	SS2	SensorStrobe チャンネル 2。 {RTC_CNT0［（15－PRESCALE2EXP）：0］, RTC_CNT2［15：0］} と {RTC_SS2［（15－PRESCALE2EXP）：0］, {（16－PRESCALE2EXP）{1'b0}} , RTC_SS2［（PRESCALE2EXP－1）：0］} の間にマスク・マッチが存在する場合、ここには SensorStrobe アラームを作動させる 16 ビットのターゲット値が格納されます。 このマッチのアクティブ・ハイ・マスク、つまり比較のためのドント・ケア・ビットの位置は、{TCODEMSK2［（15－PRESCALE2EXP）：0］, {（16－PRESCALE2EXP）{1'b1}} , TCODEMSK2［（PRESCALE2EXP－1）：0］} と定義されます。更にここで、TCODEMSK2［15：0］はサーモメータ・デコードされたマスクで、RTC_CR4SS.SS2MSKEN?（16'hFFFF << RTC_SSMSK［7：4］）：16'h0000 と定義されます。 SensorStrobe が一致すると、RTC_SS2.SS2 の値が維持されるか、別の 16 ビット・レジスタから自動リロードされます。この更新は次のように表されます。 RTC_SS2.SS2 = RTC_CR4SS.SS2ARLEN? RTC_SS2.SS2 +（RTC_SS2? RTC_SS2HIGHDUR：RTC_SS2LOWDUR）：RTC_SS2.SS2。 RTC_SS2 を使用すれば繰返しアラームをスケジュールすることができ、そのターゲット時間は、RTC_CNT1、RTC_CNT0、および RTC_CNT2（RTC_CNT2 の小数ビット）によって定義される RTC カウントの {integer_bits, fractional_bits} 下位 16 ビットに対してチェックされます。これは、32kHz ベース・クロックのプリスケール度を考慮することを意味します。

SensorStrobe チャンネル 2 の RTC 自動リロード・ハイ時間

RTC_SS2HIGHDUR には 16 ビットのリロード値が格納されます。有効にされた SensorStrobe イベントがそのチャンネル上で発生すると、この値はオプションで（RTC_CR4SS.SS2ARLEN で有効化）RTC_SS2 の積算値に加えられ、RTC_SS2TGT レジスタで確認することができます。

RTC_SS2HIGHDUR を使用すると、2 のべき乗またはそれ以外の値を周期とする RTC_SS2 用の繰返しアラームを有効にすることができます。

リローディングが有効でない場合、RTC_SS2 と RTC_SS2TGT のリードバック値は同じです。つまり、RTC_SS2 内にある SensorStrobe の開始値です（有効でないのでリロードされません）。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

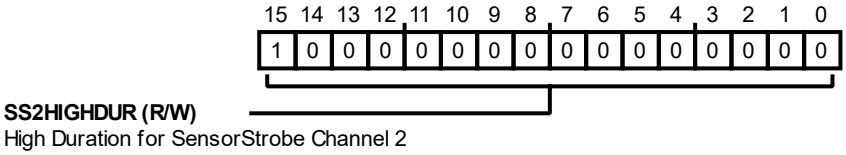


図 21-36：RTC_SS2HIGHDUR レジスタ図

表 21-35：RTC_SS2HIGHDUR レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	SS2HIGHDUR	SensorStrobe チャンネル 2 のハイ時間。 RTC_SS2 の自動リローディングが RTC_CR4SS.SS2ARLEN を介して有効にされている場合は、RTC_SS2HIGHDUR の内容が SensorStrobe チャンネル 2 のハイ時間を制御します。RTC_SS2TGT の初期値は RTC_SS2 です。 これは次のように表されます。 RTC_SS2TGT = RTC_CR4SS.SS2ARLEN?（RTC_SS2TGT +（SensorStrobe channel 2 level ? RTC_SS2LOWDUR：RTC_SS2HIGHDUR））：RTC_SS2.SS2。 SensorStrobe チャンネル 2 は、RTC タイマーの値が RTC_SS2TGT に達して、RTC_SS2TGT の新しい値が上述のように更新されると、トグルします。 このような状況においては、RTC_SS2TGT が、値 RTC_SS2LOWDUR と RTC_SS2HIGHDUR を交互にロードする反復アラームとしての役割を果たし、この際、リロード値内のマスクされていないビットが、現在時間から次の SensorStrobe アラームまでのステップ・サイズ（オフセット）を実質的に定義します。

SensorStrobe チャンネル 2 の RTC 自動リロード・ロー時間

RTC_SS2LOWDUR には 16 ビットのリロード値が格納されます。有効にされた SensorStrobe イベントがそのチャンネル上で発生すると、この値はオプションで（RTC_CR4SS.SS2ARLEN で有効化）RTC_SS2 の積算値に加えられ、RTC_SS2TGT レジスタで確認することができます。

RTC_SS2LOWDUR を使用すると、2 のべき乗またはそれ以外の値を周期とする RTC_SS2 用の繰返しアラームを有効にすることができます。

リローディングが有効でない場合、RTC_SS2 と RTC_SS2TGT のリードバック値は同じです。つまり、RTC_SS2 内にある SensorStrobe の開始値です（有効でないのでリロードされません）。

このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

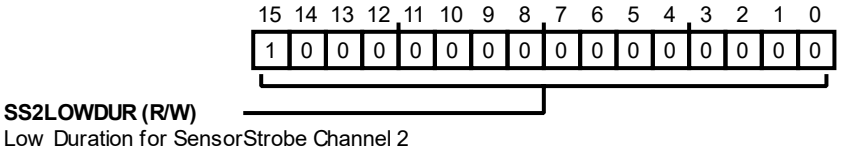


図 21-37：RTC_SS2LOWDUR レジスタ図

表 21-36：RTC_SS2LOWDUR レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	SS2LOWDUR	SensorStrobe チャンネル 2 のロー時間。 RTC_SS2 の自動リローディングが RTC_CR4SS.SS2ARLEN を介して有効にされている場合は、RTC_SS2LOWDUR の内容が RTC_SS2 の SensorStrobe チャンネルのロー時間を制御します。RTC_SS2TGT の初期値は RTC_SS2 です。 これは次のように表されます。 RTC_SS2TGT = RTC_CR4SS.SS2ARLEN? (RTC_SS2TGT + (SensorStrobe channel 2 level ? RTC_SS2LOWDUR : RTC_SS2HIGHDUR)) : RTC_SS2.SS2。 SensorStrobe チャンネル 2 出力は、RTC タイマーの値が RTC_SS2TGT に達して、RTC_SS2TGT の新しい値が上述のように更新されると、トグルします。 このような状況においては、RTC_SS2TGT が、値 RTC_SS2LOWDUR と RTC_SS2HIGHDUR を交互にロードする反復アラームとしての役割を果たし、この際、リロード値内のマスクされていないビットが、現在時間から次の SensorStrobe アラームまでのステップ・サイズ（オフセット）を実質的に定義します。

RTC SensorStrobe チャンネル 2 のターゲット

SensorStrobe イベント発生時の自動リローディングを考慮した上で SensorStrobe チャンネル 2 の現在の累積ターゲット・アラーム時間が反映される、読出し専用レジスタ。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

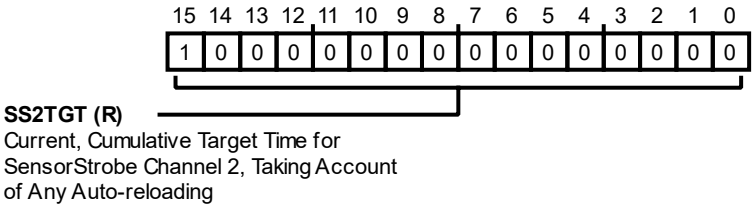


図 21-38：RTC_SS2TGT レジスタ図

表 21-37：RTC_SS2TGT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	SS2TGT	自動リローディングを考慮した SensorStrobe チャンネル 2 の現在の累積ターゲット時間。 RTC_SS2TGT.SS2TGT は読出し専用レジスタで、考え得るすべての自動リローディングを考慮に入れた上で、SensorStrobe チャンネルの RTC_SS2 の現在のターゲット値を CPU が確認できるようにします。 RTC_CR4SS.SS2ARLEN によって自動リロードが有効にされていない場合、RTC_SS2TGT.SS2TGT によって返される値は RTC_SS2 レジスタの値、つまり SensorStrobe チャンネル 2 の開始値と同じです。 自動リローディングが有効にされている場合、RTC_SS2TGT.SS2TGT のリードバックは、RTC_SS2 チャンネルの現在の累積値を返します。これは、RTC_SS2 レジスタに格納された値から SensorStrobe シーケンスを開始してリロードした値です。 つまり、SensorStrobe イベントの発生ごとに、新しいターゲット時間（RTC_SS2LOWDUR と RTC_SS2HIGHDUR の値によりオフセット）を加算して累積したものです。 32kHz 領域に同期されているときに以下のいずれかの設定変更を行うと、累積された RTC_SS2TGT.SS2TGT ターゲット時間が再初期化されて、その後の自動リロード・シーケンス時に RTC_SS2 レジスタの開始値に再設定されます。つまり、以下のいずれかの場合は新しいリロード・シーケンスが開始されて、RTC_SS2TGT.SS2TGT が RTC_SS2 の値に再初期化されます。 ［1］RTC_SS2 MMR への書込みによって開始値自体が再定義される。 ［2］RTC_CR3SS.SS2EN が 0 から 1 に遷移して、RTC_SS2 SensorStrobe チャンネルがイネーブルされる。 ［3］RTC_CR4SS.SS2ARLEN が 1 から 0 に遷移して、RTC_SS2 の自動リローディングが無効になる。 RTC_SS2TGT.SS2TGT のソースは 32kHz 領域であり、このような再定義が RTC_SS2TGT.SS2TGT に反映された RTC_SS2 チャンネルのターゲット時間に影響する場合は、関係する MMR の再定義の WSYNC フラグが 1'b1 であること（PCLK 領域に同期された結果）をチェックする必要があります。

RTC SensorStrobe チャンネル 3

このレジスタは {integer_bits, fractional_bits} の下位 16 ビットに関する SensorStrobe チャンネル 3 のスケジュール・アラーム時間で、これはメインの 47 ビット RTC カウントを意味します。これらの 16 ビットの位置より上位にあるメイン RTC カウントのビットは、16 ビット SensorStrobe チャンネル 3 に関してはドント・ケアです。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

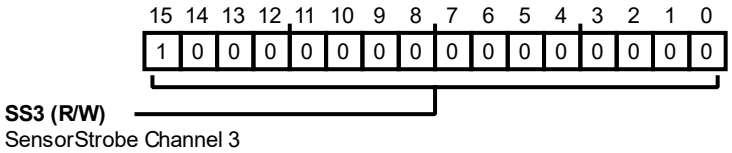


図 21-39：RTC_SS3 レジスタ図

表 21-38：RTC_SS3 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	SS3	SensorStrobe チャンネル 3 {RTC_CNT0［（15－PRESCALE2EXP）：0］, RTC_CNT2［15：0］} と {RTC_SS3［（15－PRESCALE2EXP）：0］, {（16－PRESCALE2EXP）{1'b0}}, RTC_SS3［（PRESCALE2EXP－1）：0］} の間にマスク・マッチが存在する場合、ここには SensorStrobe アラームを作動させる 16 ビットのターゲット値が格納されます。 このマッチのアクティブ・ハイ・マスク、つまり比較のためのドント・ケア・ビットの位置は、{TCODEMSK3［（15－PRESCALE2EXP）：0］, {（16－PRESCALE2EXP）{1'b1}}, TCODEMSK3［（PRESCALE2EXP－1）：0］} と定義されます。更にここで、TCODEMSK3［15：0］はサーモメータ・デコードされたマスクで、RTC_CR4SS.SS3MSKEN?（16'hFFFF << RTC_SSMSK［11：8］）：16'h0000 と定義されます。 SensorStrobe が一致すると、RTC_SS3.SS3 の値が維持されるか、別の 16 ビット・レジスタから自動リロードされます。この更新は次のように表されます。RTC_SS3.SS3 = RTC_CR4SS.SS3ARLEN? RTC_SS3.SS3 + （（RTC_SS3）? RTC_SS3HIGHDUR：RTC_SS3LOWDUR）：RTC_SS3.SS3。 RTC_SS3 を使用すれば繰返しアラームをスケジュールすることができ、そのターゲット時間は、RTC_CNT1、RTC_CNT0、および RTC_CNT2（RTC_CNT2 の小数ビット）によって定義される RTC カウントの {integer_bits, fractional_bits} 下位 16 ビットに対してチェックされます。これは、32kHz ベース・クロックのプリスケール度を考慮することを意味します。

SensorStrobe チャンネル 3 の RTC 自動リロード・ハイ時間

RTC_SS3HIGHDUR には 16 ビットのリロード値が格納されます。有効にされた SensorStrobe イベントがそのチャンネル上で発生すると、この値はオプションで（RTC_CR4SS.SS3ARLEN で有効化）RTC_SS3 の積算値に加えられ、RTC_SS3TGT レジスタで確認することができます。RTC_SS3HIGHDUR を使用すると、2 のべき乗またはそれ以外の値を周期とする RTC_SS3 用の繰り返しアラームを有効にすることができます。リローディングが有効でない場合、RTC_SS3 と RTC_SS3TGT のリードバック値は同じです。つまり、RTC_SS3 内にある SensorStrobe の開始値です（有効でないのでリロードされません）。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

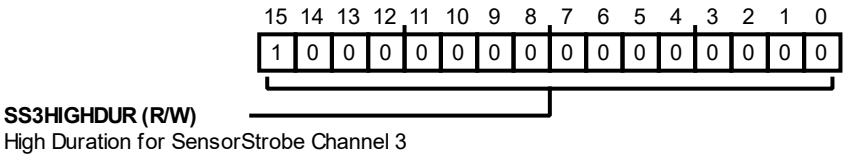


図 21-40：RTC_SS3HIGHDUR レジスタ図

表 21-39：RTC_SS3HIGHDUR レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	SS3HIGHDUR	SensorStrobe チャンネル 3 のハイ時間。 RTC_SS3 の自動リローディングが RTC_CR4SS.SS3ARLEN を介して有効にされている場合は、RTC_SS3HIGHDUR の内容が SensorStrobe チャンネル 3 のハイ時間を制御します。RTC_SS3TGT の初期値は RTC_SS3 です。 これは次のように表されます。 RTC_SS3TGT = RTC_CR4SS.SS3ARLEN?（RTC_SS3TGT +（SensorStrobe channel 3 level ? RTC_SS3LOWDUR：RTC_SS3HIGHDUR））：RTC_SS3.SS3。 SensorStrobe チャンネル 3 は、RTC タイマーの値が RTC_SS3TGT に達して、RTC_SS3TGT の新しい値が上述のように更新されると、トグルします。 このような状況においては、RTC_SS3TGT が、値 RTC_SS3LOWDUR と RTC_SS3HIGHDUR を交互にロードする反復アラームとしての役割を果たし、この際、リロード値内のマスクされていないビットが、現在時間から次の SensorStrobe アラームまでのステップ・サイズ（オフセット）を実質的に定義します。

SensorStrobe チャンネル 3 の RTC 自動リロード・ロー時間

RTC_SS3LOWDUR には 16 ビットのリロード値が格納されます。有効にされた SensorStrobe イベントがそのチャンネル上で発生すると、この値はオプションで（RTC_CR4SS.SS3ARLEN で有効化）RTC_SS3 の積算値に加えられ、RTC_SS3TGT レジスタで確認することができます。RTC_SS3LOWDUR を使用すると、2 のべき乗またはそれ以外の値を周期とする RTC_SS3 用の繰返しアラームを有効にすることができます。リローディングが有効でない場合、RTC_SS3 と RTC_SS3TGT のリードバック値は同じです。つまり、RTC_SS3 内にある SensorStrobe の開始値です（有効でないのでリロードされません）。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

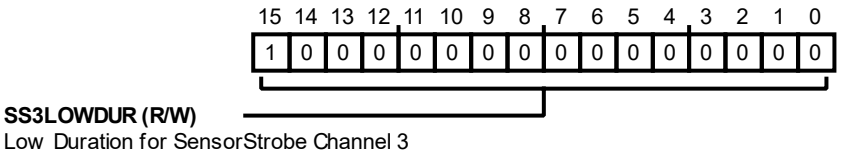


図 21-41：RTC_SS3LOWDUR レジスタ図

表 21-40：RTC_SS3LOWDUR レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	SS3LOWDUR	SensorStrobe チャンネル 3 のロー時間。 RTC_SS3 の自動リローディングが RTC_CR4SS.SS3ARLEN を介して有効にされている場合は、RTC_SS3LOWDUR の内容が RTC_SS3 の SensorStrobe チャンネルのロー時間を制御します。RTC_SS3TGT の初期値は RTC_SS3 です。 これは次のように表されます。 RTC_SS3TGT = RTC_CR4SS.SS3ARLEN?（RTC_SS3TGT +（SensorStrobe channel 3 level ? RTC_SS3LOWDUR：RTC_SS3HIGHDUR））：RTC_SS3.SS3。 SensorStrobe チャンネル 3 出力は、RTC タイマーの値が RTC_SS3TGT に達して、RTC_SS3TGT の新しい値が上述のように更新されると、トグルします。 このような状況においては、RTC_SS3TGT が、値 RTC_SS3LOWDUR と RTC_SS3HIGHDUR を交互にロードする反復アラームとしての役割を果たし、この際、リロード値内のマスクされていないビットが、現在時間から次の SensorStrobe アラームまでのステップ・サイズ（オフセット）を実質的に定義します。

RTC SensorStrobe チャンネル 3 のターゲット

SensorStrobe イベント発生時の自動リローディングを考慮した上で SensorStrobe チャンネル 3 の現在の累積ターゲット・アラーム時間が反映される、読出し専用レジスタ。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

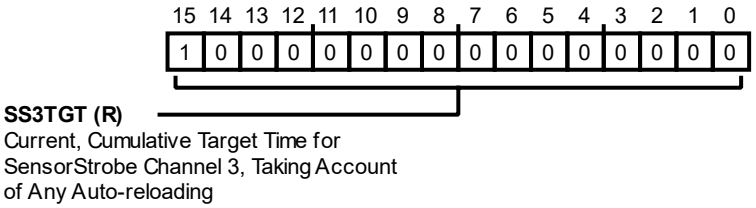


図 21-42：RTC_SS3TGT レジスタ図

表 21-41：RTC_SS3TGT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	SS3TGT	自動リローディングを考慮した SensorStrobe チャンネル 3 の現在の累積ターゲット時間。 RTC_SS3TGT.SS3TGT は読出し専用レジスタで、考え得るすべての自動リローディングを考慮に入れた上で、SensorStrobe チャンネルの RTC_SS3 の現在のターゲット値を CPU が確認できるようにします。 RTC_CR4SS.SS3ARLEN によって自動リロードが有効にされていない場合、RTC_SS3TGT.SS3TGT によって返される値は RTC_SS3 レジスタの値、つまり SensorStrobe チャンネル 3 の開始値と同じです。 ただし、自動リローディングが有効な場合、RTC_SS3TGT.SS3TGT をリードバックすると、RTC_SS3 チャンネルの現在の累積ターゲット値が返されます。この値は、RTC_SS3 レジスタ内に格納された値から SensorStrobe シーケンスを開始してリロードした値です。つまり、SensorStrobe イベントの発生ごとに新しいターゲット時間（RTC_SS3LOWDUR と RTC_SS3HIGHDUR の値によりオフセット）を加算して累積したものです。 32kHz 領域に同期されているときに以下のいずれかの設定変更を行うと、累積された RTC_SS3TGT.SS3TGT ターゲット時間が再初期化されて、その後の自動リロード・シーケンス時に RTC_SS3 レジスタの開始値に再設定されます。つまり、以下のいずれかの場合は新しいリロード・シーケンスが開始されて、RTC_SS3TGT.SS3TGT が RTC_SS3 の値に再初期化されます。 ［1］RTC_SS3 MMR への書込みによって開始値自体が再定義される。 ［2］RTC_CR3SS.SS3EN が 0 から 1 に遷移して、RTC_SS3 SensorStrobe チャンネルがイネーブルされる。 ［3］RTC_CR4SS.SS3ARLEN が 1 から 0 に遷移して、RTC_SS3 の自動リローディングが無効になる。 RTC_SS3TGT.SS3TGT のソースは 32kHz 領域であり、このような再定義が RTC_SS3TGT.SS3TGT に反映された RTC_SS3 チャンネルのターゲット時間に影響する場合は、関係する MMR の再定義の WSYNC フラグが 1'b1 であること（つまり、PCLK 領域に同期された結果）をチェックする必要があります。

RTC SensorStrobe チャンネル 4

このレジスタは {integer_bits, fractional_bits} の下位 16 ビットに関する SensorStrobe チャンネル 4 のスケジュール・アラーム時間で、これはメインの 47 ビット RTC カウントを意味します。これらの 16 ビットの位置より上位にあるメイン RTC カウントのビットは、16 ビット SensorStrobe チャンネル 4 に関してはドント・ケアです。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

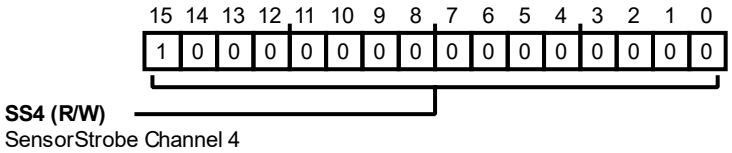


図 21-43：RTC_SS4 レジスタ図

表 21-42：RTC_SS4 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	SS4	SensorStrobe チャンネル 4。 {RTC_CNT0［（15－PRESCALE2EXP）：0］, RTC_CNT2［15：0］} と {RTC_SS4［（15－PRESCALE2EXP）：0］, {（16－PRESCALE2EXP）{1'b0}}, RTC_SS4［（PRESCALE2EXP－1）：0］} の間にマスク・マッチが存在する場合、ここには SensorStrobe アラームを作動させる 16 ビットのターゲット値が格納されます。 このマッチのアクティブ・ハイ・マスク、つまり比較のためのドント・ケア・ビット位置は、{TCODEMSK4［（15－PRESCALE2EXP）：0］, {（16－PRESCALE2EXP）{1'b1}}, TCODEMSK4［（PRESCALE2EXP－1）：0］} と定義されます。更にここで、TCODEMSK4［15：0］はサーモメータ・デコードされたマスクで、RTC_CR4SS.SS4MSKEN?（16'hFFFF << RTC_SSMSK［15：12］）：16'h0000 と定義されます。 RTC_SS1 と異なり、RTC_SS2、RTC_SS3、または RTC_SS4 では、SensorStrobe イベント時に自動リローディングを行うことはできません。代わりに、SensorStrobe が一致したときは、RTC_SS4 の値が維持されます。したがって、SensorStrobe チャンネル 4 は、RTC_SS4 のマスクされていないビットにより与えられる 2 のべき乗を周期とする繰返しアラームとして機能します。 RTC_SS4 を使用すれば繰返しアラームをスケジュールすることができ、そのターゲット時間は、RTC_CNT1、RTC_CNT0、および RTC_CNT2（RTC_CNT2 の小数ビット）によって定義される RTC カウントの {integer_bits, fractional_bits} 下位 16 ビットに対してチェックされます。これは、32kHz ベース・クロックのプリスケール度を考慮することを意味します。

SensorStrobe チャンネル用 RTC マスク

RTC_SSMSK には 4 ビットのエンコード・マスクが格納されます。このマスクは、16 ビットのサーモメータ・コード・マスクにデコードされて、16 ビット SensorStrobe チャンネル 1 のターゲット・アラーム時間用に連続ドント・ケア・ビット位置を定義します。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

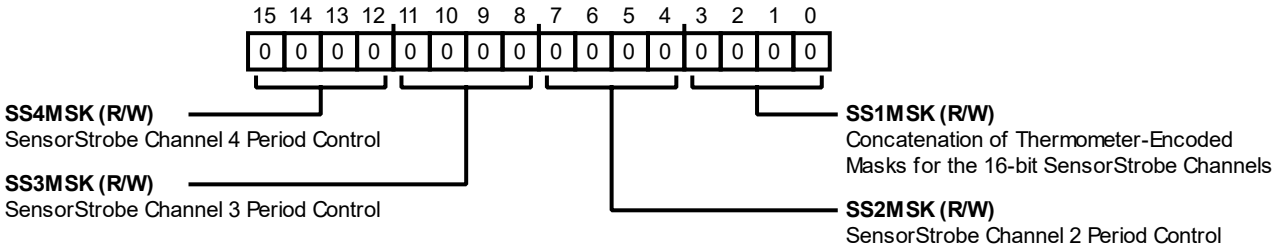


図 21-44：RTC_SSMSK レジスタ図

表 21-43：RTC_SSMSK レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:12 (R/W)	SS4MSK	SensorStrobe チャンネル 4 の周期制御。 SensorStrobe チャンネル 4 の周期は、2 × RTC_SSMSK.SS4MSK RTC クロック周期です。SensorStrobe チャンネル 1 の RTC_SSMSK［3：0］と同様に、RTC_SSMSK［7：4］は SensorStrobe チャンネル 2 のオプション・マスクとしてデコードされ、RTC_SSMSK［11：8］は SensorStrobe チャンネル 3、RTC_SSMSK［15：12］は SensorStrobe チャンネル 4 のオプション・マスクとしてデコードされます。
11:8 (R/W)	SS3MSK	SensorStrobe チャンネル 3 の周期制御。 SensorStrobe チャンネル 3 の周期は、2 × RTC_SSMSK.SS3MSK RTC クロック周期です。SensorStrobe チャンネル 1 の RTC_SSMSK［3：0］と同様に、RTC_SSMSK［7：4］は SensorStrobe チャンネル 2 のオプション・マスクとしてデコードされ、RTC_SSMSK［11：8］は SensorStrobe チャンネル 3、RTC_SSMSK［15：12］は SensorStrobe チャンネル 4 のオプション・マスクとしてデコードされます。
7:4 (R/W)	SS2MSK	SensorStrobe チャンネル 2 の周期制御。 SensorStrobe チャンネル 2 の周期は、2 × RTC_SSMSK.SS2MSK RTC クロック周期です。SensorStrobe チャンネル 1 の RTC_SSMSK［3：0］と同様に、RTC_SSMSK［7：4］は SensorStrobe チャンネル 2 のオプション・マスクとしてデコードされ、RTC_SSMSK［11：8］は SensorStrobe チャンネル 3、RTC_SSMSK［15：12］は SensorStrobe チャンネル 4 のオプション・マスクとしてデコードされます。

表 21-43：RTC_SSMSK レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応
3:0 (R/W)	SS1MSK	16 ビット SensorStrobe チャンネル用にサーモメータ・エンコードされた一連のマスク。 SensorStrobe チャンネル 1 の周期は、2 × RTC_SSMSK.SS1MSK RTC クロック周期です。RTC_SSMSK.SS1MSK にはサーモメータ・エンコードされた 4 ビットのマスクが 4 個格納されて、16 ビットのレジスタを構成します。16 ビット値にサーモメータ・エンコードされた場合、ビット［3：0］は RTC_CR4SS.SS1MSKEN によってイネーブルされるオプション・マスクとして機能し、チャンネル RTC_SS1 の RTC カウントとの一致の判定に使われます。 SensorStrobe チャンネル<n>では、RTC カウントとのマッチのアクティブ・ハイ・マスク、つまり比較のためのドント・ケア・ビットの位置は、｛TCODEMSK<n>［（15 − PRESCALE2EXP）：0］,｛（16 − PRESCALE2EXP）｛1'b1｝｝, TCODEMSK<n>［（PRESCALE2EXP − 1）：0］｝と定義されます。ここで、TCODEMSK<n>［15：0］はサーモメータ・デコードされたマスクで、RTC_SS<n>MSKEN ?（16'hFFFF << RTC_SSMSK［（n × 4）+：4］）：16'h0000 で定義されます。更にここで、インデックス n の範囲は 0〜3 です。 4 つの 16 ビット SensorStrobe チャンネルごとに個別にデコードされるサーモメータ・マスクは、以下のいずれかの値を取ります。16'h0000（ビット・マスクなし）、16'8000（MSBit マスク）、16'hC000（2 MSBit マスク）、16'hE000、16'hF000、...、16'hFFF0、16'hFFF8、16'hFFFC（2 LS ビットだけ未マスク）、16'hFFFE（LS ビットだけ未マスク）、16'hFFFF（すべてのビットをマスク、連続的な SensorStrobe の一致を示唆しています）。

SensorStrobe チャンネル・オン時間制御用 RTC マスク

ここには 4 個の 4 ビット・エンコード・マスクが格納されます。このマスクは 4 個の 16 ビット・サーモメータ・コード・マスクにデコードされて、16 ビット SensorStrobe チャンネルのターゲット・アラーム時間用に連続ドント・ケア・ビットの位置を定義します。このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

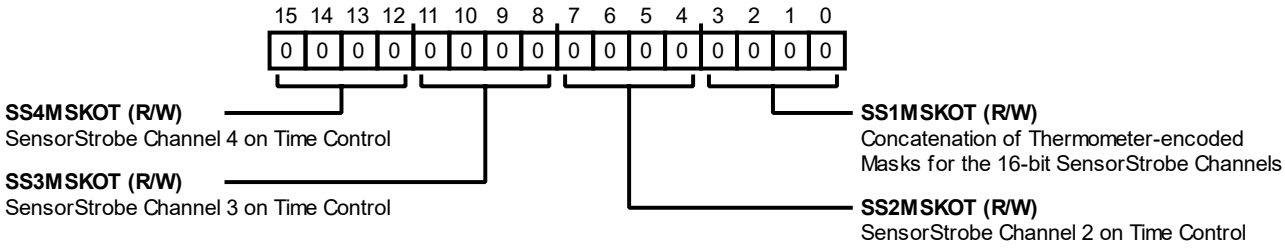


図 21-45：RTC_SSMSKOT レジスタ図

表 21-44：RTC_SSMSKOT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:12 (R/W)	SS4MSKOT	SensorStrobe チャンネル 4 のオン時間制御。 このフィールドは、デフォルトで SensorStrobe チャンネル 4 出力のオン時間を定義します。SensorStrobe チャンネル 4 のオン時間は、2 × RTC_SSMSKOT.SS4MSKOT RTC クロック周期です。RTC_CR4SS.SS4POL をセットした場合、このレジスタは SensorStrobe チャンネル 4 のオフ時間を定義します。
11:8 (R/W)	SS3MSKOT	SensorStrobe チャンネル 3 のオン時間制御。 このフィールドは、デフォルトで SensorStrobe チャンネル 3 出力のオン時間を定義します。SensorStrobe チャンネル 3 のオン／オフ時間は、2 × RTC_SSMSKOT.SS3MSKOT RTC クロック周期です。RTC_CR4SS.SS3POL をセットした場合、このレジスタは SensorStrobe チャンネル 3 のオフ時間を定義します。RTC_CR4SS.SS3ARLEN をセットしたときは、このフィールドを 0 にする必要があります。
7:4 (R/W)	SS2MSKOT	SensorStrobe チャンネル 2 のオン時間制御。 このフィールドは、デフォルトで SensorStrobe チャンネル 2 出力のオン時間を定義します。SensorStrobe チャンネル 2 のオン／オフ時間は、2 × RTC_SSMSKOT.SS2MSKOT RTC クロック周期です。RTC_CR4SS.SS2POL をセットした場合、このレジスタは SensorStrobe チャンネル 2 のオフ時間を定義します。RTC_CR4SS.SS2ARLEN をセットしたときは、このフィールドを 0 にする必要があります。
3:0 (R/W)	SS1MSKOT	16 ビット SensorStrobe チャンネル用にサーモメータ・エンコードされた一連のマスク。 このフィールドは、デフォルトで SensorStrobe チャンネル 1 出力のオン時間を定義します。SensorStrobe チャンネル 1 のオン／オフ時間は、2 × RTC_SSMSKOT.SS1MSKOT RTC クロック周期です。RTC_CR4SS.SS1POL をセットした場合、このレジスタは SensorStrobe チャンネル 1 のオフ時間を定義します。RTC_CR4SS.SS1ARLEN をセットしたときは、このフィールドを 0 にする必要があります。

RTC スナップショット 0

RTC_SNAP0 は、値 RTC_CNT0 のスティッキー・スナップショットです。以下のいずれかのイベントが発生すると、対応する RTC_SNAP1 および RTC_SNAP2 と共に更新されて、前の値を上書きします。

(i) CPU がスナップショット要求キー16'h7627 を RTC_GWY MMR に書き込む。

(ii) 入力キャプチャ・チャンネル 0 がイネーブルされた状態で入力キャプチャ・イベントが発生する。ただし、RTC_CR2IC.ICOWUSEN の設定によってこのような上書きを許可されている場合。

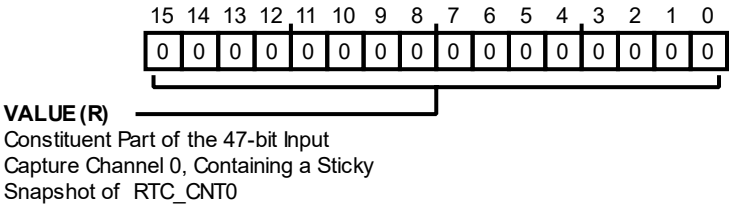


図 21-46：RTC_SNAP0 レジスタ図

表 21-45：RTC_SNAP0 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	VALUE	47 ビット入力キャプチャ・チャンネル 0 の構成部分で、RTC_CNT0 のスティッキー・スナップショットが格納されます。 RTC_SNAP0.VALUE は、47 ビット入力キャプチャ・チャンネル 0 の一部です。

RTC スナップショット 1

RTC_SNAP1 は、値 RTC_CNT1 のスティッキー・スナップショットです。以下のいずれかのイベントが発生すると、対応する RTC_SNAP0 および RTC_SNAP2 と共に更新されて、前の値を上書きします。

(i) CPU がスナップショット要求キー16'h7627 を RTC_GWY MMR に書き込む。

(ii) 入力キャプチャ・チャンネル 0 がイネーブルされた状態で入力キャプチャ・イベントが発生する。ただし、RTC_CR2IC.ICOWUSEN の設定によってこのような上書きを許可されている場合。

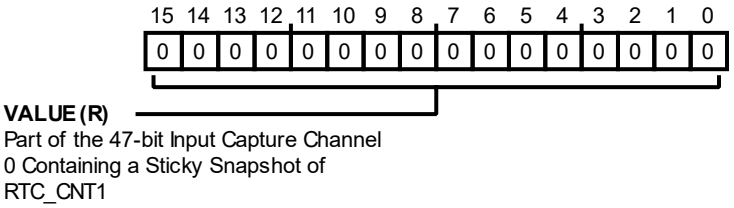


図 21-47：RTC_SNAP1 レジスタ図

表 21-46：RTC_SNAP1 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	VALUE	47 ビット入力キャプチャ・チャンネル 0 の一部で、RTC_CNT1 のスティッキー・スナップショットが格納されます。 RTC_SNAP1.VALUE は、47 ビット入力キャプチャ・チャンネル 0 の一部です。

RTC スナップショット 2

RTC_SNAP2 は、値 RTC_CNT2 のスティッキー・スナップショットです。以下のいずれかのイベントが発生すると、対応する RTC_SNAP0 および RTC_SNAP1 と共に更新されて、前の値を上書きします。

(i)　CPU がスナップショット要求キー16'h7627 を RTC_GWY MMR に書き込む。

(ii)　入力キャプチャ・チャンネル 0 がイネーブルされた状態で入力キャプチャ・イベントが発生する。ただし、RTC_CR2IC.ICOWUSEN の設定によってこのような上書きを許可されている場合。

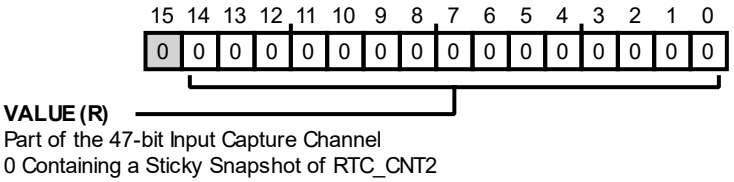


図 21-48：RTC_SNAP2 レジスタ図

表 21-47：RTC_SNAP2 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
14:0 (R/NW)	VALUE	47 ビット入力キャプチャ・チャンネル 0 の一部で、RTC_CNT2 のスティッキー・スナップショットが格納されます。 RTC_SNAP2.VALUE は、47 ビット入力キャプチャ・チャンネル 0 の一部です。

RTC ステータス 0

RTC 動作に関する情報は、3 つのステータス・レジスタ、RTC_SR0、RTC_SR1、および RTC_SR2 を介して CPU に提供されます。これらのレジスタには、RTC 内の CPU 割込みソースとエラー条件に関係するすべてのフラグが含まれています。

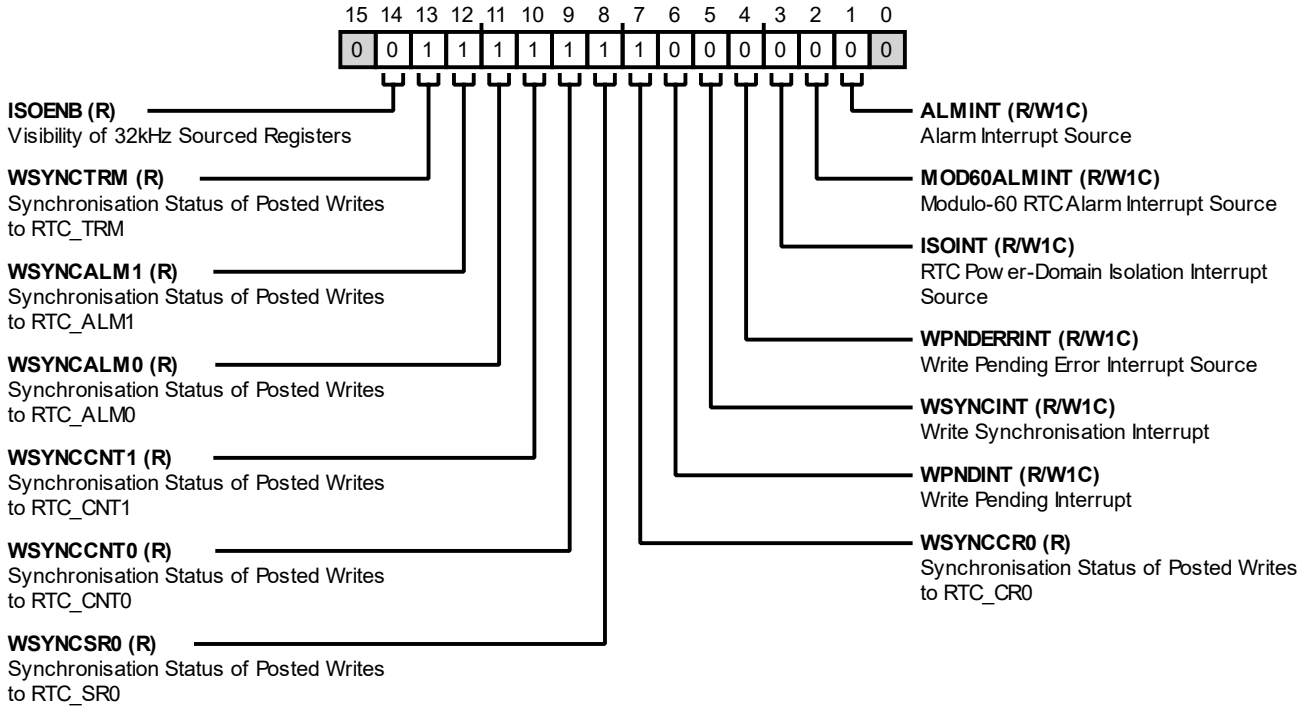


図 21-49：RTC_SR0 レジスタ図

表 21-48：RTC_SR0 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
14 (R/NW)	ISOENB	32kHz をソースとするレジスタの確認。 電源領域の絶縁を考慮した 32kHz ソース・レジスタの確認ステータス。	
		0	絶縁されているので、CPU は RTC の常時オン側にある 32kHz ソース MMR の状態を確認できません。
		1	CPU は RTC の常時オン側にある 32kHz ソース MMR の状態を確認できます。
13 (R/NW)	WSYNCTRM	RTC_TRM に対してポストされた書込みの同期ステータス。 このフィールドは、RTC_TRM に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、ポストされた書込みの結果をまだ確認できません。
		1	CPU は、ポストされた書込みの結果を確認できます。

表 21-48：RTC_SR0 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
12 (R/NW)	WSYNCALM1	RTC_ALM1 に対してポストされた書込みの同期ステータス。 このフィールドは、RTC_ALM1 に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、ポストされた書込みの結果をまだ確認できません。
		1	CPU は、ポストされた書込みの結果を確認できます。
11 (R/NW)	WSYNCALM0	RTC_ALM0 に対してポストされた書込みの同期ステータス。 このフィールドは、RTC_ALM0 に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、ポストされた書込みの結果をまだ確認できません。
		1	CPU は、ポストされた書込みの結果を確認できます。
10 (R/NW)	WSYNCCNT1	RTC_CNT1 に対してポストされた書込みの同期ステータス。 このフィールドは、RTC_CNT1 に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、ポストされた書込みの結果をまだ確認できません。
		1	CPU は、ポストされた書込みの結果を確認できます。
9 (R/NW)	WSYNCCNT0	RTC_CNT0 に対してポストされた書込みの同期ステータス。 このフィールドは、RTC_CNT0 に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、ポストされた書込みの結果をまだ確認できません。
		1	CPU は、ポストされた書込みの結果を確認できます。
8 (R/NW)	WSYNCSR0	RTC_SR0 に対してポストされた書込みの同期ステータス。 このフィールドは、RTC_SR0 に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、ポストされた書込みの結果をまだ確認できません。
		1	CPU は、ポストされた書込みの結果を確認できます。
7 (R/NW)	WSYNCCR0	RTC_CR0 に対してポストされた書込みの同期ステータス。 このフィールドは、RTC_CR0 に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU はポストされた書込みの結果をまだ確認できません。
		1	CPU はポストされた書込みの結果を確認できます。

表 21-48：RTC_SR0 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
6 (R/W1C)	WPNDINT	書込み保留割込み。 CPU が RTC の 32kHz ソース MMR または MMR ビット・フィールドへの新しい書込みトランザクションをポストするための空間が解放されるときにアクティブ化される、スティッキー割込みソース。 RTC_SR0.WPNDINT 割込みをイネーブルするには、RTC_CR0.WPNDINTEN を 1 に設定します。RTC_SR0.WPNDINT は、1 を書き込むことによってクリアされます。	
		0	WPENDINT が最後にクリアされてから、RTC 内のポストされた書込みトランザクションの保留ステータスに変更はありません。
		1	WPENDINT が最後にクリアされてから、ポストされた書込みトランザクションがディスパッチされ、新しくポストされる CPU から同じ MMR への書込みのためにスロットが解放されました。
5 (R/W1C)	WSYNCINT	書込み同期割込み。 32kHz ソースの MMR または MMR ビット・フィールドに対してポストされた書込みトランザクションが完了して、CPU がその影響を確認できるようになったときにアクティブ化される、スティッキー割込みソース。 RTC_SR0.WSYNCINT 割込みをイネーブルするには、RTC_CR0.WSYNCINTEN を 1 に設定します。RTC_SR0.WSYNCINT は、1 を書き込むことによってクリアされます。	
		0	CPU が RTC_SR0.WSYNCINT を最後にクリアしてから、CPU のクロック領域が、32kHz ソース MMR または MMR ビット・フィールドに対してポストされた書込みトランザクションの結果を新たに確認できる状態になっていません。
		1	CPU が RTC_SR0.WSYNCINT を最後にクリアした後に、CPU のクロック領域が、32kHz ソース MMR または MMR ビット・フィールドに対してポストされた書込みトランザクションの結果を新たに確認できる状態になっています。

表 21-48：RTC_SR0 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
4 (R/W1C)	WPNDERRINT	書込み保留エラー割込みソース。 RTC レジスタへの書込み実行が保留されているトランザクションがあるときに、CPU が同じレジスタに対して新しいトランザクションの書込みを試行したためにエラーが発生したことを示す、スティッキー割込みソース。 書込みが MMR に対するもので、それが 32kHz 領域をソースとするものである場合、RTC がサポートしている保留可能書込みトランザクションは MMR あたり 1 つだけです。複数の書込み保留エラーが発生した場合（つまり、ポストされた書込みが拒否された場合）は、最初のエラー発生時に RTC_SR0.WPNDERRINT がアクティブになり、以後はそのまま維持されます。RTC_SR0.WPNDERRINT は、1 を書き込むことによってクリアされます。	
		0	RTC_SR0.WPNDERRINT が CPU によって最後にクリアされてから、ポストされた書込みで拒否されたものはありません。
		1	書込みが拒否されました。同じ MMR に対して前にポストされた書込みがまだ実行待機中のため、新しくポストされた書込みが RTC によって拒否されました。この拒否は、CPU が最後に RTC_SR0.WPNDERRINT をクリアした後に発生しています。
3 (R/W1C)	ISOINT	RTC 電源領域絶縁割込みソース。 コアの電源喪失のために、RTC がその電源領域の絶縁障壁をアクティブにしなければならないかどうかを示すスティッキー割込みソース。CPU は、コアの電源復帰時に ISOINT を読み出して、このような電源イベントに関する情報を得ることができます。 RTC_SR0.ISOINT は、CPU が 1 を書き込むことによってクリアされます。 このビット・フィールドは RTC1 にのみ存在します。RTC0 ではこのフィールドは予備で、ゼロとして読み出されます。	
		0	RTC_SR0.ISOINT 割込みソースが CPU によって最後にクリアされてから、常時オン RTC 電源領域がコアとの絶縁をアクティブにしていません。
		1	電源イベントのために常時オン RTC 電源領域がアクティブになり、その後、コアとの絶縁が非アクティブになりました。このイベントは、CPU が最後に RTC_SR0.ISOINT をクリアした後に発生しました。

表 21-48：RTC_SR0 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
2 (R/W1C)	MOD60ALMINT	モジュロ 60 RTC アラーム割込みソース。 オプションでイネーブルされた CPU への割込みのソースとなるスティッキー・フラグ。この割込みは、RTC_CR0.MOD60ALM の変位がインクリメントされてモジュロ 60 境界を超え、RTC_CNT1 と RTC_CNT0 の整数カウントが 60 までインクリメントされるたびに、アクティブになります。 このソースのイネーブルは RTC_CR0.MOD60ALMINTEN を使用して行い、そのターゲット時間の設定は RTC_CR0.MOD60ALM を使用して行います。また、クリアは、そのビット位置に 1 を書き込むことによって行います。 このビット・フィールドは RTC1 にのみ存在します。RTC0 ではこのフィールドは予備で、ゼロとして読み出されます。	
		0	CPU が最後にこのビットをクリアした後、RTC_SR0.MOD60ALMINT 割込みイベントは発生していません。
		1	CPU が最後にこのビットをクリアした後、RTC_SR0.MOD60ALMINT 割込みイベントが発生しました。
1 (R/W1C)	ALMINT	アラーム割込みソース。 オプションでイネーブルされた CPU への割込みのソースとなるスティッキー・フラグ。これは、RTC カウントとアラーム・レジスタの値が一致したためにアラーム・イベントが発生したことを示します。RTC_CNT1、RTC_CNT0、および RTC_CNT2 の値が、RTC_ALM1、RTC_ALM0、および RTC_ALM2 によって与えられるアラーム時間と等しい場合は、一致と判断されます。このようなイベントの検出は、RTC_CR0 の RTC_CR0.ALMEN によって有効になりますが、これは RTC_CR0.CNTEN もイネーブルされていることが前提となります。 RTC_SR0.ALMINT は、値 1 を書き込むことによってクリアされます。	
		0	CPU が最後にこのビットをクリアした後、RTC_SR0.ALMINT 割込みイベントは発生していません。
		1	CPU が最後にこのビットをクリアした後、RTC_SR0.ALMINT 割込みイベントが発生しました。

RTC ステータス 1

RTC 動作に関する情報は、3 つのステータス・レジスタ、RTC_SR0、RTC_SR1、および RTC_SR2 を介して CPU に提供されます。これらのレジスタには、RTC 内の CPU 割込みソースとエラー条件に関係するすべてのフラグが含まれています。

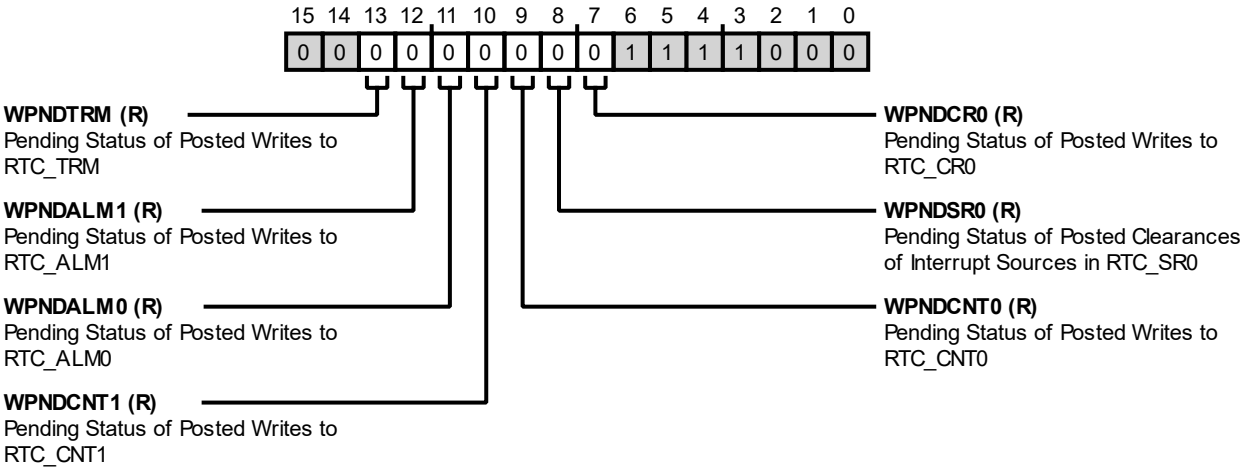


図 21-50：RTC_SR1 レジスタ図

表 21-49：RTC_SR1 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
13 (R/NW)	WPNDTRM	RTC_TRM に対してポストされた書込みの保留ステータス。 RTC_TRM に対してポストされたレジスタ書込みが現在保留されて（バッファされキューに入れられて）実行待ち状態かどうか、したがってこの時点では同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_TRM MMR に対して新たにポストされた書込みを受け入れ可能です。
		1	過去に RTC_TRM に対してポストされた書込みがまだ実行待ち状態なので、この MMR には新たな書込みをポストすることはできません。
12 (R/NW)	WPNDALM1	RTC_ALM1 に対してポストされた書込みの保留ステータス。 RTC_ALM1 に対してポストされたレジスタ書込みが現在保留されて（バッファされキューに入れられて）実行待ち状態かどうか、したがってこの時点では同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_ALM1 MMR に対して新たにポストされた書込みを受け入れ可能です。
		1	過去に RTC_ALM1 に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。

表 21-49：RTC_SR1 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
11 (R/NW)	WPNDALM0	RTC_ALM0 に対してポストされた書込みの保留ステータス。 RTC_ALM0 に対してポストされたレジスタ書込みが現在保留されて（バッファされキューに入れられて）実行待ち状態かどうか、したがってこの時点では同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_ALM0 MMR に対して新たにポストされた書込みを受け入れ可能です。
		1	過去に RTC_ALM0 に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
10 (R/NW)	WPNDCNT1	RTC_CNT1 に対してポストされた書込みの保留ステータス。 RTC_CNT1 に対してポストされたレジスタ書込みが現在保留されて（バッファされキューに入れられて）実行待ち状態かどうか、したがってこの時点では同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_CNT1 MMR に対して新たにポストされた書込みを受け入れ可能です。
		1	過去に RTC_CNT1 に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
9 (R/NW)	WPNDCNT0	RTC_CNT0 に対してポストされた書込みの保留ステータス。 RTC_CNT0 に対してポストされたレジスタ書込みが現在保留されて（バッファされキューに入れられて）実行待ち状態かどうか、したがってこの時点では同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_CNT0 MMR に対して新たにポストされた書込みを受け入れ可能です。
		1	過去に RTC_CNT0 に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
8 (R/NW)	WPNDSR0	RTC_SR0 の割込みソースについてポストされたクリアの保留ステータス。 RTC_SR0 の割込みソースにポストされたクリアが現在保留されていて（バッファされてキューに入っていて）、実行待ち状態にあるかどうかを示します。	
		0	RTC は、32kHz 領域に置かれた RTC_SR0 の割込みソースにポストされたクリアを受け入れ可能です。
		1	32kHz 領域に置かれている RTC_SR0 の割込みソースに対して過去にポストされたクリアが、まだ実行待ち状態です。追加のクリアは、既存の保留トランザクションにまとめることができます。

表 21-49：RTC_SR1 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
7 (R/NW)	WPNDCR0	RTC_CR0 に対してポストされた書込みの保留ステータス。 RTC_CR0 に対してポストされたレジスタ書込みが現在保留されて（バッファされキューに入れられて）実行待ち状態かどうか、したがってこの時点では同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_CR0 に対して新たにポストされた書込みを受け入れ可能です。
		1	過去に RTC_CR0 に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。

RTC ステータス 2

RTC_SR2 は、RTC_SR0 と RTC_SR1 によって提供されるステータス情報を補完するステータス・レジスタです。RTC1 はすべての RTC_SR2 機能を備えていますが、RTC0 では機能が限定されています。

RTC_SR2 のすべての割込みソースは、スティッキーでアクティブ・ハイのレベル信号です。各ソースは 1 を書き込むことによってクリアできます。

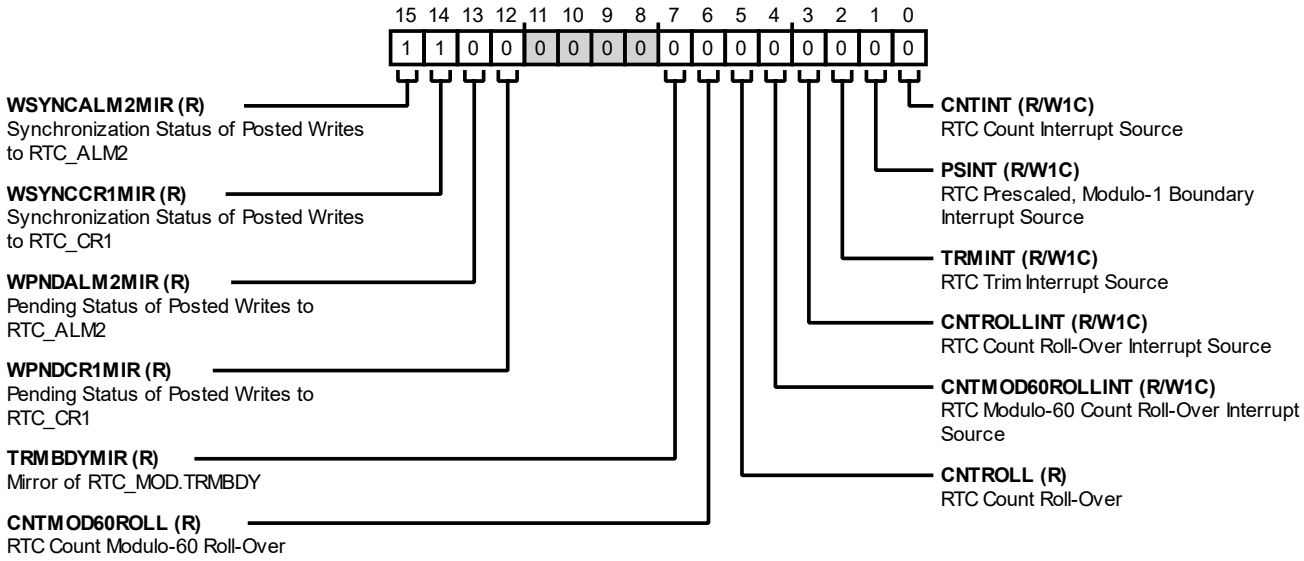


図 21-51：RTC_SR2 レジスタ図

表 21-50：RTC_SR2 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15 (R/NW)	WSYNCALM2MIR	RTC_ALM2 に対してポストされた書込みの同期ステータス。 RTC_ALM2 に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	ポストされた書込みの結果をまだ確認できません。
		1	ポストされた書込みの結果を確認できます。
14 (R/NW)	WSYNCCR1MIR	RTC_CR1 に対してポストされた書込みの同期ステータス。 RTC_CR1 に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	ポストされた書込みの結果をまだ確認できません。
		1	ポストされた書込みの結果を確認できます。

表 21-50：RTC_SR2 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
13 (R/NW)	WPNDALM2MIR	RTC_ALM2 に対してポストされた書込みの保留ステータス。 RTC_ALM2 に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_ALM2 に対して新たにポストされた書込みを受け入れ可能です。
		1	過去にポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
12 (R/NW)	WPNDCR1MIR	RTC_CR1 に対してポストされた書込みの保留ステータス。 WPENDCR1 は、RTC_CR1 に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_CR1 に対して新たにポストされた書込みを受け入れ可能です。
		1	過去にポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
7 (R/NW)	TRMBDYMIR	RTC_MOD.TRMBDY のミラー。 このビット・フィールドは、RTC_MOD.TRMBDYMMR の値の読出し専用ミラーです。これがここに含められていることで、RTC_SR2 の読出し時に、メイン RTC カウントまたはそれに相当するモジュロ 60 の繰り越しが、トリミングによってどのような影響を受けるかを知ることができます。	
6 (R/NW)	CNTMOD60ROLL	RTC カウントのモジュロ 60 繰り越し。 RTC_SR2.CNTMOD60ROLL は、最新のプリスケールされた時間単位でインクリメントしたときに可能最大値から可能最小値へ繰り越すことによって、RTC_MOD.CNTMOD60 MMR で与えられる RTC カウントのモジュロ 60 値が生成されたかどうかを示します。 このビット・フィールドは RTC1 にのみ存在します。RTC0 ではこのフィールドは予備で、ゼロとして読み出されます。	
		0	繰り越しによる RTC_MOD.CNTMOD60 内の RTC カウントのモジュロ 60 値の生成はありませんでした。
		1	現在 RTC_MOD.CNTMOD60 内にある RTC カウントのモジュロ 60 値が、その最大値のトリミング距離内の値から最小値のトリミング距離内の値へ繰り越しました。

表 21-50：RTC_SR2 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
5 (R/NW)	CNTROLL	RTC カウントの繰り越し。 RTC_SR2.CNTROLL は、最新のプリスケールされた時間単位でインクリメントしたときに可能最大値から可能最小値へ繰り越すことによって、RTC_CNT1、RTC_CNT0、および RTC_CNT2 で与えられる RTC リアルタイム・カウントの現在値が生成されたかどうかを示します。	
		0	繰り越しによる RTC リアルタイム・カウントの現在値は生成されませんでした。
		1	RTC リアルタイム・カウントの現在値が、その最大値のトリミング距離内の値から最小値のトリミング距離内の値へ繰り越しました。
4 (R/W1C)	CNTMOD60ROLLINT	RTC モジュロ 60 カウント繰り越し割込みソース。 このソースは、RTC_CNT1 および RTC_CNT0 内の整数カウント値に相当するモジュロ 60 値が 59 から 0 に繰り越されるか、これらの値が両端となるようにトリムされると、アクティブ・ハイに固定されます。このような繰り越しイベントは、プリスケールされた RTC カウントの 60 インクリメントごとに発生しますが、正の（加算）トリミングがアクティブの場合は、これより少なくなります。 RTC から RTC_SR2.CNTMOD60ROLLINT に割込みを行わせるには、この割込みファン・イン項に対応するイネーブル・ビット RTC_CR1.CNTMOD60ROLLINTEN がアクティブ・ハイでなければなりません。 この割込みソースは、1 を書き込むことによってクリアされます。 このビット・フィールドは RTC1 にのみ存在します。RTC0 ではこのフィールドは予備で、ゼロとして読み出されます。	
		0	RTC_SR2.CNTMOD60ROLLINT が最後にクリアされた後、RTC_MOD.CNTMOD60 内にある RTC_CNT1 と RTC_CNT0 のモジュロ 60 値は繰り越していません。
		1	RTC_SR2.CNTMOD60ROLLINT が最後にクリアされた後、RTC_MOD.CNTMOD60 内にある RTC_CNT1 と RTC_CNT0 のモジュロ 60 値は繰り越しました。

表 21-50：RTC_SR2 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
3 (R/W1C)	CNTROLLINT	RTC カウント・繰り越し割込みソース。 このソースは、RTC_CNT1 および RTC_CNT0 内の整数カウント値が（2^32-1）から 0 に繰り越されるか、トリム・インクリメントによって RTC がこれらの最大値と最小値を通過する（場合によってはこれらの値を両端とする）ようにトリムされると、アクティブ・ハイに固定されます。 RTC から RTC_SR2.CNTROLLINT に割込みを行わせるには、この割込みファン・イン項に対応するイネーブル・ビット RTC_CR1.CNTROLLINTEN がアクティブ・ハイでなければなりません。 この割込みソースは、1 を書き込むことによってクリアされます。 RTC0 で CPU が取得できる情報は、RTC_SR2.CNTROLLINT を読み出すことによって得られる情報だけです。RTC_CR1.CNTROLLINTEN がないので、RTC_SR2.CNTROLLINT を RTC0 の割込みファン・イン項としてイネーブルすることはできません。これに対し、RTC1 ではすべての割込み機能を使用できます。	
		0	RTC_SR2.CNTROLLINT が最後にクリアされた後、RTC_CNT1 と RTC_CNT0 の整数カウントは繰り越していません。
		1	RTC_SR2.CNTROLLINT が最後にクリアされた後、RTC_CNT1 と RTC_CNT0 の整数カウントが繰り越しました。
2 (R/W1C)	TRMINT	RTC トリム割込みソース。 イネーブルされたトリム間隔の終了時にトリム境界に達すると、このソースがアクティブ・ハイに固定され、RTC_CNT1 と RTC_CNT0 の整数値は RTC_TRM の設定に従って調整されます。 RTC から RTC_SR2.TRMINT に割込みを行わせるには、この割込みファン・イン項に対応するイネーブル・ビット RTC_CR1.TRMINTEN がアクティブ・ハイでなければなりません。 この割込みソースは、1 を書き込むことによってクリアされます。 RTC0 で CPU が取得できる情報は、RTC_SR2.TRMINT を読み出すことによって得られる情報だけです。RTC_CR1.TRMINTEN がないので、RTC_SR2.TRMINT を RTC0 の割込みファン・イン項としてイネーブルすることはできません。これに対し、RTC1 ではすべての割込み機能を使用できます。	
		0	RTC_SR2.TRMINT が最後にクリアされた後は、RTC トリム間隔境界に達していません。
		1	RTC_SR2.TRMINT が最後にクリアされた後に、RTC トリム間隔境界に達しました。

表 21-50：RTC_SR2 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
1 (R/W1C)	PSINT	RTC のプリスケールされたモジュロ 1 境界割込みソース。 プリスケールされた RTC 時間単位が経過して RTC 整数カウントがインクリメントまたはトリムされると、このソースがアクティブ・ハイに固定されます。 RTC から RTC_SR2.PSINT に割込みを行わせるには、この割込みファン・イン項に対応するイネーブル・ビット RTC_CR1.PSINTEN がアクティブ・ハイでなければなりません。 この割込みソースは、RTC_SR2 の該当ビット位置に 1 を書き込むことによってクリアされます。 RTC0 で CPU が取得できる情報は、RTC_SR2.PSINT を読み出すことによって得られる情報だけです。RTC_CR1.PSINTEN がないので、RTC_SR2.PSINT を RTC0 の割込みファン・イン項としてイネーブルすることはできません。これに対し、RTC1 ではすべての割込み機能を使用できます。	
		0	カウントはインクリメントまたはトリムされていません。 RTC_SR2.PSINT が最後にクリアされてから、RTC 整数カウントはインクリメントまたはトリムされていません。
		1	RTC_SR2.PSINT が最後にクリアされた後に、RTC 整数カウントがインクリメントまたはトリムされました。
0 (R/W1C)	CNTINT	RTC カウント割込みソース。 RTC_CNT1 と RTC_CNT0 の整数カウント値が変化すると、このソースがアクティブ・ハイに固定されます。これらのイベントは、プリスケールされた RTC 時間単位の発生（RTC_SR2.PSINT）と同じではありません。RTC カウントは再定義またはトリムできますが、これによって値が変更されることもあれば、されないこともあるからです。 この割込みソースは、1 を書き込むことによってクリアされます。 RTC0 で CPU が取得できる情報は、RTC_SR2.CNTINT を読み出すことによって得られる情報だけです。RTC_CR1.CNTINTEN がないので、RTC_SR2.CNTINT を RTC0 の割込みファン・イン項としてイネーブルすることはできません。これに対し、RTC1 ではすべての割込み機能を使用できます。	
		0	RTC 整数カウントの値は、最後に RTC_SR2.CNTINT がクリアされてから変化していません。
		1	RTC 整数カウントの値は、 最後に RTC_SR2.CNTINT がクリアされた後に変化しています。

RTC ステータス 3

RTC_SR3 は、1 を書き込むことによってクリアされる割込みソースを格納するステータス・レジスタで、イネーブルされた入力キャプチャや SensorStrobe チャンネルに関するイベントが発生すると、アクティブ・ハイに固定されます。

このレジスタは RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

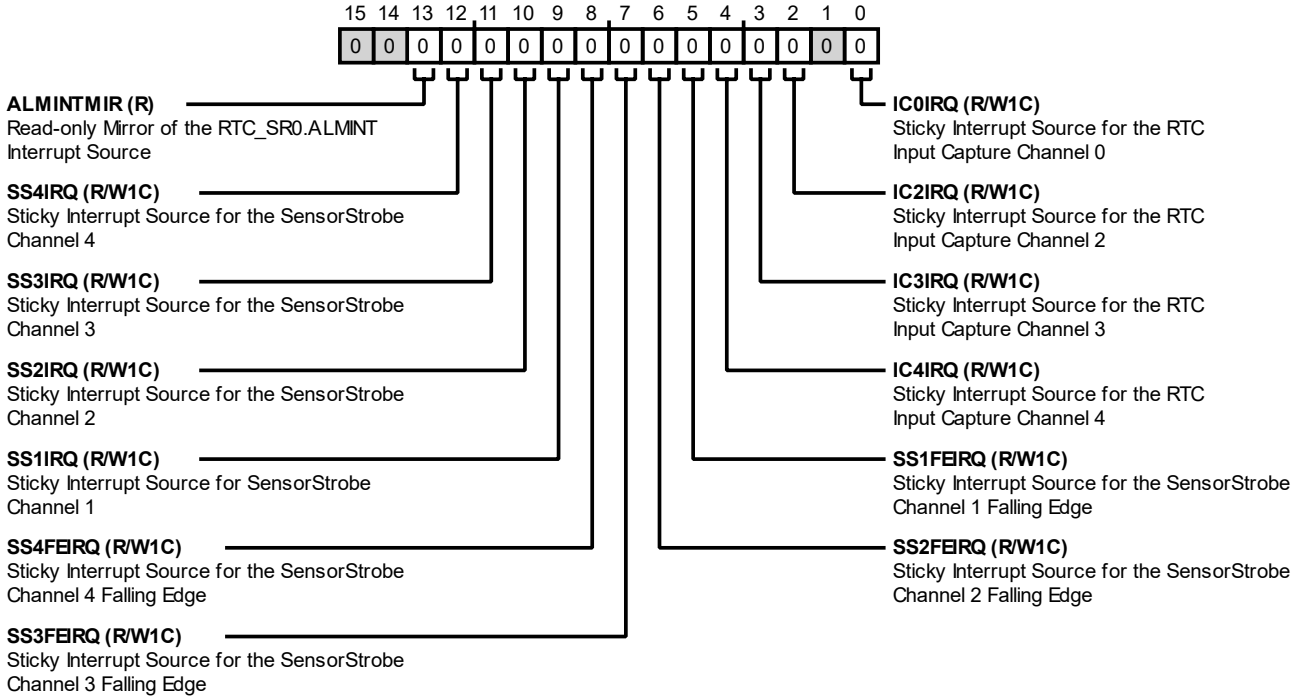


図 21-52：RTC_SR3 レジスタ図

表 21-51：RTC_SR3 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
13 (R/NW)	ALMINTMIR	RTC_SR0.ALMINT 割込みソースの読出し専用ミラー。 RTC_SR0.ALMINT をアクティブにする RTC の 47 ビット絶対時間アラーム機能は、SensorStrobe チャンネル 0 として複製されます。	
		0	CPU が RTC_SR0.ALMINT 割込みソース・ビットを最後にクリアしてから、ALMINT 割込みイベントは発生していません。
		1	CPU が RTC_SR0.ALMINT 割込みソース・ビットを最後にクリアした後、ALMINT 割込みイベントが発生しています。

表 21-51：RTC_SR3 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
12 (R/W1C)	SS4IRQ	SensorStrobe チャンネル 4 のスティッキー割込みソース。 この割込みソースは、（RTC_CR3SS.SS4EN を介して）イネーブルされた SensorStrobe チャンネル 4 に立上がりエッジを生成するスケジュール・アラーム・イベントが発生すると、ハイに固定されます。 RTC_SR3.SS4IRQ は、そのビット位置に値 1'b1 を書き込むことによってクリアされます。RTC_CR3SS.SS4IRQEN を介してイネーブルされた場合、RTC_SR3.SS4IRQ ソースは、CPU とウェイクアップ・コントローラへ送られる RTC 割込みラインへの関与項として含まれます。	
		0	CPU が最後にこのビットをクリアしてから、SensorStrobe チャンネル 4 上で、有効な SensorStrobe イベント（SensorStrobe 出力の立上がりエッジ）は発生していません。
		1	CPU が最後にこのビットをクリアした後に、SensorStrobe チャンネル 4 上で、有効な SensorStrobe イベント（SensorStrobe 出力の立上がりエッジ）が発生しました。
11 (R/W1C)	SS3IRQ	SensorStrobe チャンネル 3 のスティッキー割込みソース。 この割込みソースは、（RTC_CR3SS.SS3EN を介して）イネーブルされた SensorStrobe チャンネル 3 に立上がりエッジを生成するスケジュール・アラーム・イベントが発生すると、ハイに固定されます。 RTC_SR3.SS3IRQ は、そのビット位置に値 1'b1 を書き込むことによってクリアされます。RTC_CR3SS.SS3IRQEN を介してイネーブルされた場合、RTC_SR3.SS3IRQ ソースは、CPU とウェイクアップ・コントローラへ送られる RTC 割込みラインへの関与項として含まれます。	
		0	CPU が最後にこのビットをクリアしてから、SensorStrobe チャンネル 3 上で、有効な SensorStrobe イベント（SensorStrobe 出力の立上がりエッジ）は発生していません。
		1	CPU が最後にこのビットをクリアした後に、SensorStrobe チャンネル 3 上で、イネーブルされた SensorStrobe イベント（SensorStrobe 出力の立上がりエッジ）が発生しました。
10 (R/W1C)	SS2IRQ	SensorStrobe チャンネル 2 のスティッキー割込みソース。 この割込みソースは、（RTC_CR3SS.SS2EN を介して）イネーブルされた SensorStrobe チャンネル 2 に立上がりエッジを生成するスケジュール・アラーム・イベントが発生すると、ハイに固定されます。 RTC_SR3.SS2IRQ は、そのビット位置に値 1'b1 を書き込むことによってクリアされます。RTC_CR3SS.SS2IRQEN を介してイネーブルされた場合、RTC_SR3.SS2IRQ ソースは、CPU とウェイクアップ・コントローラへ送られる RTC 割込みラインへの関与項として含まれます。	
		0	CPU が最後にこのビットをクリアしてから、SensorStrobe チャンネル 2 上で、有効な SensorStrobe イベント（SensorStrobe 出力の立上がりエッジ）は発生していません。
		1	CPU が最後にこのビットをクリアした後に、SensorStrobe チャンネル 2 上で、有効な SensorStrobe イベント（SensorStrobe 出力の立上がりエッジ）が発生しました。

表 21-51：RTC_SR3 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
9 (R/W1C)	SS1IRQ	SensorStrobe チャンネル 1 のスティッキー割込みソース。 この割込みソースは、（RTC_CR3SS.SS1EN を介して）イネーブルされた SensorStrobe チャンネル 1 に立上がりエッジを生成するスケジュール・アラーム・イベントが発生すると、ハイに固定されます。 RTC_SR3.SS1IRQ は、1 を書き込むことによってクリアされます。RTC_CR3SS.SS1IRQEN を介してイネーブルされた場合、RTC_SR3.SS1IRQ ソースは、CPU とウェイクアップ・コントローラへ送られる RTC 割込みラインへの関与項として含まれます。	
		0	CPU が最後にこのビットをクリアしてから、RTC_SS1 チャンネル上で、有効な SensorStrobe イベント（SensorStrobe 出力の立上がりエッジ）は発生していません。
		1	CPU が最後にこのビットをクリアした後に、RTC_SS1 チャンネル上で、有効な SensorStrobe イベント（SensorStrobe 出力の立上がりエッジ）が発生しました。
8 (R/W1C)	SS4FEIRQ	SensorStrobe チャンネル 4 立下がりエッジのスティッキー割込みソース。 この割込みソースは、（RTC_CR3SS.SS4EN を介して）イネーブルされた SensorStrobe チャンネル 4 の立下がりエッジを生成するスケジュール・アラーム・イベントが発生すると、ハイに固定されます。 RTC_SR3.SS4IRQ は、そのビット位置に値 1'b1 を書き込むことによってクリアされます。RTC_CR3SS.SS4IRQEN を介してイネーブルされた場合、RTC_SR3.SS4FEIRQ ソースは、CPU とウェイクアップ・コントローラへ送られる RTC 割込みラインへの関与項として含まれます。	
		0	CPU が最後にこのビットをクリアしてから、SensorStrobe チャンネル 4 上で、有効な SensorStrobe イベント（SensorStrobe 出力の立下がりエッジ）は発生していません。
		1	CPU が最後にこのビットをクリアした後に、SensorStrobe チャンネル 4 上で、有効な SensorStrobe イベント（SensorStrobe 出力の立下がりエッジ）が発生しました。
7 (R/W1C)	SS3FEIRQ	SensorStrobe チャンネル 3 立下がりエッジのスティッキー割込みソース。 この割込みソースは、（RTC_CR3SS.SS3EN を介して）イネーブルされた SensorStrobe チャンネル 3 の立下がりエッジを生成するスケジュール・アラーム・イベントが発生すると、ハイに固定されます。 RTC_SR3.SS3IRQ は、そのビット位置に値 1'b1 を書き込むことによってクリアされます。RTC_CR3SS.SS3IRQEN を介してイネーブルされた場合、RTC_SR3.SS3FEIRQ ソースは、CPU とウェイクアップ・コントローラへ送られる RTC 割込みラインへの関与項として含まれます。	
		0	CPU が最後にこのビットをクリアしてから、SensorStrobe チャンネル 3 上で、有効な SensorStrobe イベント（SensorStrobe 出力の立下がりエッジ）は発生していません。
		1	CPU が最後にこのビットをクリアした後に、SensorStrobe チャンネル 3 上で、有効な SensorStrobe イベント（SensorStrobe 出力の立下がりエッジ）が発生しました。

表 21-51：RTC_SR3 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
6 (R/W1C)	SS2FEIRQ	SensorStrobe チャンネル 2 立下がりエッジのスティッキー割込みソース。 この割込みソースは、（RTC_CR3SS.SS2EN を介して）イネーブルされた SensorStrobe チャンネル 2 の立下がりエッジを生成するスケジュール・アラーム・イベントが発生すると、ハイに固定されます。 RTC_SR3.SS2IRQ は、そのビット位置に値 1'b1 を書き込むことによってクリアされます。RTC_CR3SS.SS2IRQEN を介してイネーブルされた場合、RTC_SR3.SS2FEIRQ ソースは、CPU とウェイクアップ・コントローラへ送られる RTC 割込みラインへの関与項として含まれます。	
		0	CPU が最後にこのビットをクリアしてから、SensorStrobe チャンネル 2 上で、有効な SensorStrobe イベント（SensorStrobe 出力の立下がりエッジ）は発生していません。
		1	CPU が最後にこのビットをクリアした後に、SensorStrobe チャンネル 2 上で、有効な SensorStrobe イベント（SensorStrobe 出力の立下がりエッジ）が発生しました。
5 (R/W1C)	SS1FEIRQ	SensorStrobe チャンネル 1 立下がりエッジのスティッキー割込みソース。 この割込みソースは、（RTC_CR3SS.SS1EN を介して）イネーブルされた SensorStrobe チャンネル 1 の立下がりエッジを生成するスケジュール・アラーム・イベントが発生すると、ハイに固定されます。 RTC_SR3.SS1IRQ は、そのビット位置に値 1'b1 を書き込むことによってクリアされます。RTC_CR3SS.SS1IRQEN を介してイネーブルされた場合、RTC_SR3.SS1FEIRQ ソースは、CPU とウェイクアップ・コントローラへ送られる RTC 割込みラインへの関与項として含まれます。	
		0	CPU が最後にこのビットをクリアしてから、RTC_SS1 チャンネル上で、有効な SensorStrobe イベント（SensorStrobe 出力の立下がりエッジ）は発生していません。
		1	CPU が最後にこのビットをクリアした後に、RTC_SS1 チャンネル上で、有効な SensorStrobe イベント（SensorStrobe 出力の立下がりエッジ）が発生しました。
4 (R/W1C)	IC4IRQ	RTC 入力キャプチャ・チャンネル 4 のスティッキー割込みソース。 RTC_SR3 のこの割込みソースは、（RTC_CR2IC.IC4EN を介して）イネーブルされた RTC への入力要求元が、RTC_IC4 入力キャプチャ・チャンネルの RTC カウントのスナップショットを取得するように要求すると、ハイに固定されます。このソースは、そのビット位置に値 1'b1 を書き込むことによってクリアされます。	
		0	CPU が最後にこのビットをクリアしてから、有効な入力キャプチャ・イベントは RTC_IC4 チャンネル上で発生していません。
		1	CPU が最後にこのビットをクリアした後に、有効な入力キャプチャ・イベントが RTC_IC4 チャンネル上で発生しました。

表 21-51：RTC_SR3 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
3 (R/W1C)	IC3IRQ	RTC 入力キャプチャ・チャンネル 3 のスティッキー割込みソース。 RTC_IC3 のこの割込みソースは、（RTC_CR2IC.IC3EN を介して）イネーブルされた RTC への入力要求元が、RTC_IC3 入力キャプチャ・チャンネルの RTC カウントのスナップショットを取得するように要求すると、ハイに固定されます。このソースは、そのビット位置に値 1'b1 を書き込むことによってクリアされます。	
		0	CPU が最後にこのビットをクリアしてから、有効な入力キャプチャ・イベントは RTC_IC3 チャンネル上で発生していません。
		1	CPU が最後にこのビットをクリアした後に、有効な入力キャプチャ・イベントが RTC_IC3 チャンネル上で発生しました。
2 (R/W1C)	IC2IRQ	RTC 入力キャプチャ・チャンネル 2 のスティッキー割込みソース。 RTC_SR3 のこの割込みソースは、（RTC_CR2IC.IC2EN を介して）イネーブルされた RTC への入力要求元が、RTC_IC2 入力キャプチャ・チャンネルの RTC カウントのスナップショットを取得するように要求すると、ハイに固定されます。このソースは、そのビット位置に値 1'b1 を書き込むことによってクリアされます。	
		0	CPU が最後にこのビットをクリアしてから、有効な入力キャプチャ・イベントは RTC_IC2 チャンネル上で発生していません。
		1	CPU が最後にこのビットをクリアした後に、有効な入力キャプチャ・イベントが RTC_IC2 チャンネル上で発生しました。
0 (R/W1C)	IC0IRQ	RTC 入力キャプチャ・チャンネル 0 のスティッキー割込みソース。 RTC_SR3 のこの割込みソースは、（RTC_CR2IC.IC0EN を介して）イネーブルされた RTC への入力要求元（CPU 以外）が、IC0 入力キャプチャ・チャンネルの RTC カウントのスナップショットを取得するように要求すると、ハイに固定されます。このソースは、そのビット位置に値 1'b1 を書き込むことによってクリアされます。このフィールドは、CPU が要求するスナップショットの影響を受けません（この要求は、0x7627 という特別なキー値を GWY レジスタへ書き込むことによって行われます）。	
		0	CPU が最後にこのビットをクリアしてから、CPU 以外の有効な入力キャプチャ・イベントは IC0 チャンネル上で発生していません。
		1	CPU が最後にこのビットをクリアした後に、CPU 以外の有効な入力キャプチャ・イベントが IC0 チャンネル上で発生しました。

RTC ステータス 4

RTC_SR4 はステータス・レジスタで、入力キャプチャと出力制御に関係し、更に RTC の 32kHz 常時オン側をソースとするレジスタに対して、ポストされた書込みと読出しの同期ステータスを提供します。

RTC_SR4 は RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

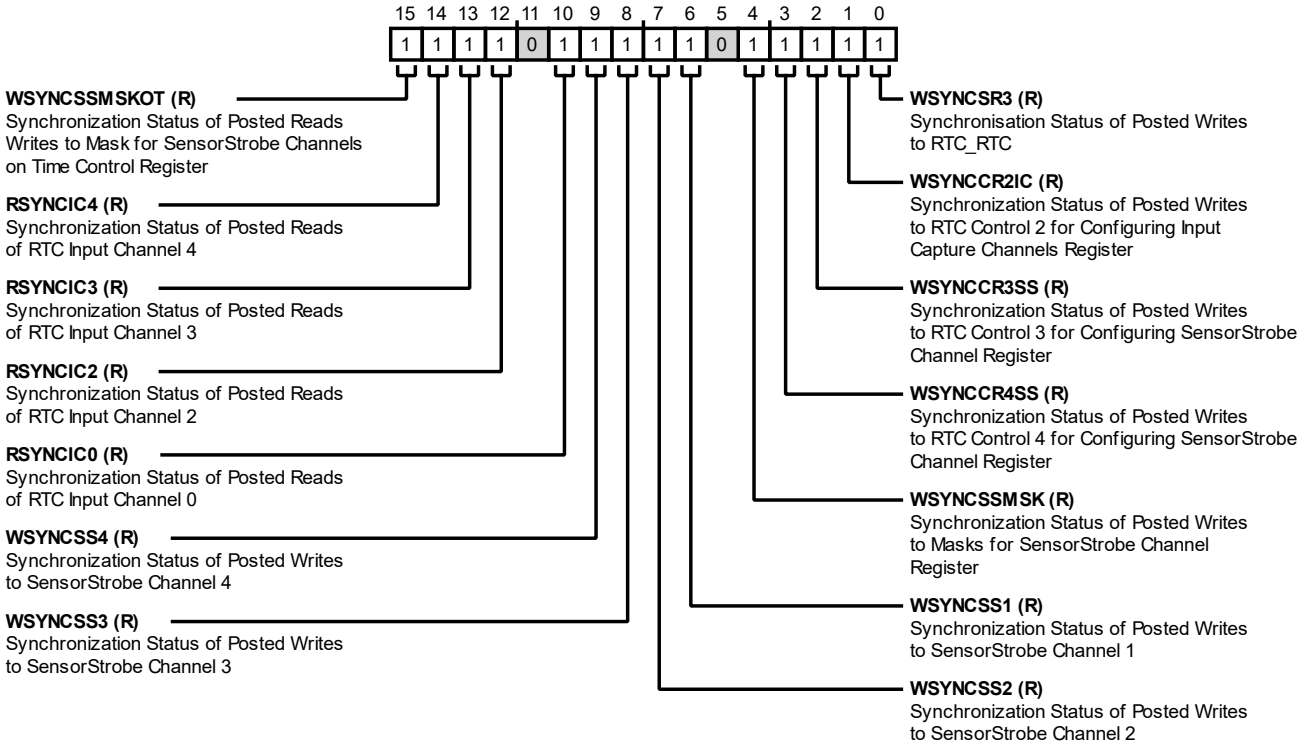


図 21-53：RTC_SR4 レジスタ図

表 21-52：RTC_SR4 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15 (R/NW)	WSYNCSSMSKOT	時間コントロール・レジスタの SensorStrobe チャンネルのマスクに対してポストされた読出しと書込みの同期ステータス。 このフィールドは、RTC_SSMSKOT に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、RTC_SSMSKOT に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_SSMSKOT に対してポストされた書込みの結果を確認できます。
14 (R/NW)	RSYNCIC4	RTC 入力チャンネル 4 のポストされた読出しの同期ステータス。 このフィールドは、RTC_IC4 の読出し結果を CPU が確認できるかどうか、具体的には RTC_SR6.IC4UNR が更新されて、その入力キャプチャ・チャンネルの未読出しステータスが反映されたかどうかを示します。	
		0	CPU は、IC4 の読出しの結果をまだ確認できません。
		1	CPU は、IC4 の読出しの結果を確認できます。
13 (R/NW)	RSYNCIC3	RTC 入力チャンネル 3 のポストされた読出しの同期ステータス。 このフィールドは、RTC_IC3 の読出し結果を CPU が確認できるかどうか、具体的には RTC_SR6.IC3UNR が更新されて、その入力キャプチャ・チャンネルの未読出しステータスが反映されたかどうかを示します。	
		0	CPU は、IC3 の読出しの結果をまだ確認できません。
		1	CPU は、IC3 の読出しの結果を確認できます。
12 (R/NW)	RSYNCIC2	RTC 入力チャンネル 2 のポストされた読出しの同期ステータス。 このフィールドは、RTC_IC2 の読出し結果を CPU が確認できるかどうか、具体的には RTC_SR6.IC2UNR が更新されて、その入力キャプチャ・チャンネルの未読出しステータスが反映されたかどうかを示します。	
		0	CPU は、IC2 の読出しの結果をまだ確認できません。
		1	CPU は、IC2 の読出しの結果を確認できます。
10 (R/NW)	RSYNCIC0	RTC 入力チャンネル 0 のポストされた読出しの同期ステータス。 このフィールドは、RTC_IC0 の 47 ビットすべての読出し結果を CPU が確認できるかどうか、具体的には RTC_SR6.IC0UNR が更新されて、その入力キャプチャ・チャンネルの未読出しステータスが反映されたたかどうかを示します。	
		0	CPU は、IC0 の読出しの結果をまだ確認できません。
		1	CPU は、IC0 の読出しの結果を確認できます。

表 21-52：RTC_SR4 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
9 (R/NW)	WSYNCSS4	SensorStrobe チャンネル 4 に対してポストされた書込みの同期ステータス。 RTC_SR4.WSYNCSS4 は、RTC_SS4 に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、RTC_SS4 に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_SS4 に対してポストされた書込みの結果を確認できます。
8 (R/NW)	WSYNCSS3	SensorStrobe チャンネル 3 に対してポストされた書込みの同期ステータス。 このフィールドは、RTC_SS3 に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、RTC_SS3 に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_SS3 に対してポストされた書込みの結果を確認できます。
7 (R/NW)	WSYNCSS2	SensorStrobe チャンネル 2 に対してポストされた書込みの同期ステータス。 このフィールドは、RTC_SS2 に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、RTC_SS2 に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_SS2 に対してポストされた書込みの結果を確認できます。
6 (R/NW)	WSYNCSS1	SensorStrobe チャンネル 1 に対してポストされた書込みの同期ステータス。 このフィールドは、RTC_SS1 に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、RTC_SS1 に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_SS1 に対してポストされた書込みの結果を確認できます。
4 (R/NW)	WSYNCSSMSK	SensorStrobe チャンネル・レジスタのマスクに対してポストされた書込みの同期ステータス。 このフィールドは、RTC_SSMSK に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、RTC_SSMSK に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_SSMSK に対してポストされた書込みの結果を確認できます。

表 21-52：RTC_SR4 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
3 (R/NW)	WSYNCCR4SS	SensorStrobe チャンネル・レジスタの設定用に RTC 制御 4 に対してポストされた書込みの同期ステータス。 このフィールドは、RTC_CR4SS に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、RTC_CR4SS に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_CR4SS に対してポストされた書込みの結果を確認できます。
2 (R/NW)	WSYNCCR3SS	SensorStrobe チャンネル・レジスタの設定用に RTC 制御 3 に対してポストされた書込みの同期ステータス。 このフィールドは、RTC_CR3SS に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、RTC_CR3SS に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_CR3SS に対してポストされた書込みの結果を確認できます。
1 (R/NW)	WSYNCCR2IC	入力キャプチャ・チャンネル・レジスタの設定用に RTC 制御 2 に対してポストされた書込みの同期ステータス。 このフィールドは、RTC_CR2IC に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、RTC_CR2IC に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_CR2IC に対してポストされた書込みの結果を確認できます。
0 (R/NW)	WSYNCSR3	RTC_SR3 に対してポストされた書込みの同期ステータス。 このフィールドは、RTC_SR3 に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。	
		0	CPU は、RTC_SR3 に対してポストされた割込みのクリアの結果をまだ確認できません。
		1	CPU は、RTC_SR3 に対してポストされた割込みのクリアの結果を確認できます。

RTC ステータス 5

RTC_SR5 はステータス・レジスタで、入力キャプチャと出力制御に関係するレジスタで、RTC の 32kHz 常時オン側をソースとするものに対してポストされた書込みの保留（バッファされてキューに入っている）ステータスを提供します。

RTC_SR5 は RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

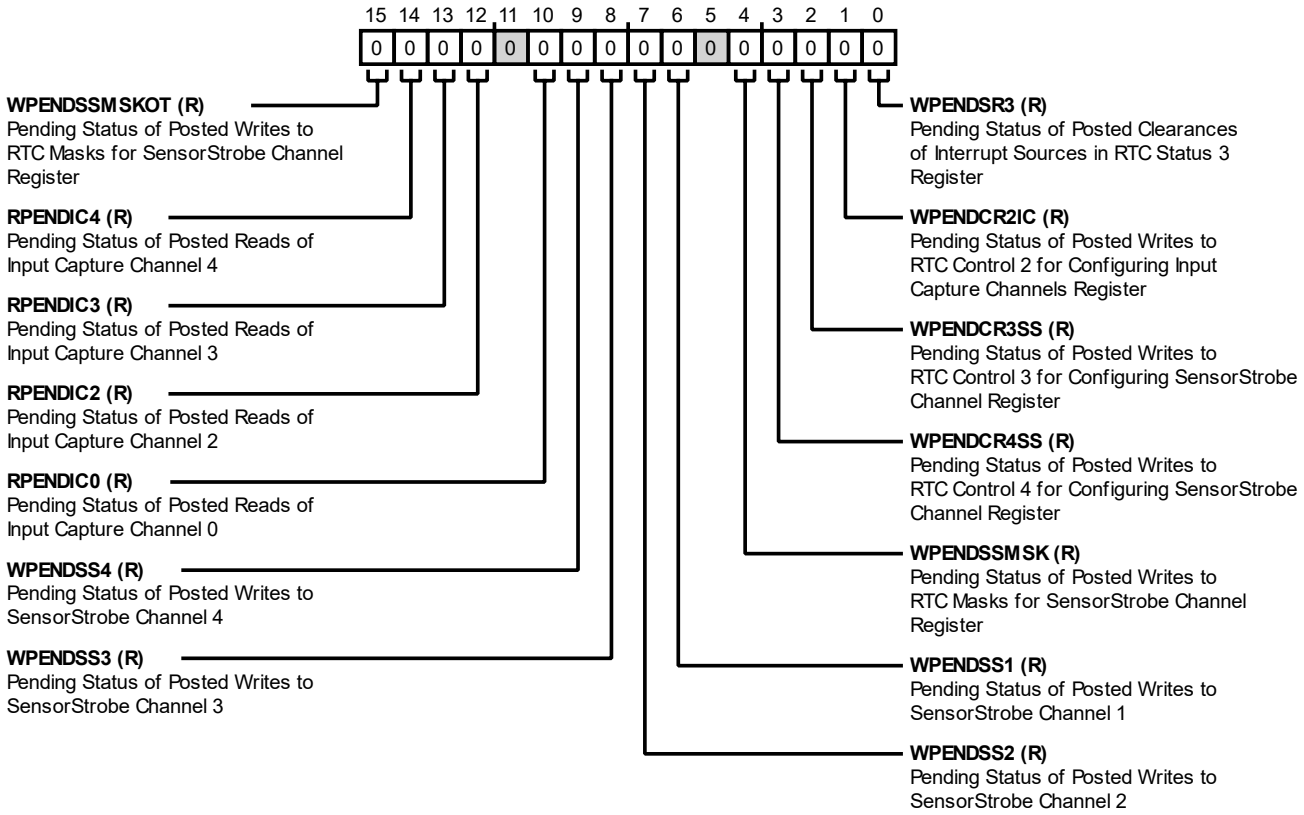


図 21-54：RTC_SR5 レジスタ図

表 21-53：RTC_SR5 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15 (R/NW)	WPENDSSMSKOT	SensorStrobe チャンネル・レジスタの RTC マスクに対してポストされた書込みの保留ステータス。 RTC_SR5.WPENDSSMSKOT は、RTC_SSMSKOT に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_SSMSKOT に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_SSMSKOT に対してポストされた書込みがまだ実行待ち状態なので、この MMR への新たな書込みをポストすることはできません。
14 (R/NW)	RPENDIC4	入力キャプチャ・チャンネル 4 のポストされた読出しの保留ステータス。 このフィールドは 32kHz 領域をソースとし、RTC_SR6.IC4UNR の値を更新するためにポストされた RTC_IC4 の読出しが現在保留され（バッファされてキューに入っており）、まだ実行待ち状態であるかどうかを示します。	
		0	RTC は、RTC_IC4 の新しい読出しを受け入れて、未読出しステータスの（読出しへの）この変更を、32kHz をソースとする RTC_SR6.IC4UNR ビット・フィールドにポストすることができます。
		1	これ以前にポストされた RTC_IC4 の未読出しステータスの（読出しへの）変更で、32kHz 領域の RTC_SR6.IC4UNR に保持されているものは、そのまま実行待ちとなります。
13 (R/NW)	RPENDIC3	入力キャプチャ・チャンネル 3 のポストされた読出しの保留ステータス。 このフィールドは 32kHz 領域をソースとし、RTC_SR6.IC3UNR の値を更新するためにポストされた RTC_IC3 の読出しが現在保留され（バッファされてキューに入っており）、まだ実行待ち状態であるかどうかを示します。	
		0	RTC は、RTC_IC3 の新しい読出しを受け入れて、未読出しステータスの（読出しへの）この変更を、32kHz をソースとする RTC_SR6.IC3UNR ビット・フィールドにポストすることができます。
		1	これ以前にポストされた RTC_IC3 の未読出しステータスの（読出しへの）変更で、32kHz 領域の RTC_SR6.IC3UNR に保持されているものは、そのまま実行待ちとなります。

表 21-53：RTC_SR5 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
12 (R/NW)	RPENDIC2	入力キャプチャ・チャンネル 2 のポストされた読出しの保留ステータス。 このフィールドは 32kHz 領域をソースとし、RTC_SR6.IC2UNR の値を更新するためにポストされた RTC_IC2 の読出しが現在保留され（バッファされてキューに入っており）、まだ実行待ち状態であるかどうかを示します。	
		0	RTC は RTC_IC2 の新しい読出しを受け入れて、未読出しステータスの（読出しへの）この変更を、32kHz をソースとする RTC_SR6.IC2UNR ビット・フィールドにポストすることができます。
		1	これ以前にポストされた RTC_IC2 の未読出しステータスの（読出しへの）変更で、32 kHz 領域の RTC_SR6.IC2UNR に保持されているものは、そのまま実行待ちとなります。
10 (R/NW)	RPENDIC0	入力キャプチャ・チャンネル 0 のポストされた読出しの保留ステータス。 このフィールドは 32kHz 領域をソースとし、RTC_SR6.IC0UNR の値を更新するために、ポストされた入力キャプチャ・チャンネル 0 の読出しが現在保留され（バッファされてキューに入っており）、まだ実行待ち状態であるかどうかを示します。	
		0	RTC は、IC0 の新しい読出しを受け入れて、未読出しステータスのこの（読出しへの）変更を、32kHz をソースとする RTC_SR6.IC0UNR ビット・フィールドにポストすることができます。
		1	これ以前にポストされた IC0 の未読出しステータスの（読出しへの）変更で、32 kHz 領域の RTC_SR6.IC0UNR に保持されているものは、そのまま実行待ちとなります。
9 (R/NW)	WPENDSS4	SensorStrobe チャンネル 4 に対してポストされた書込みの保留ステータス。 このフィールドは、RTC_SS4 に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_SS4 に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_SS4 に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
8 (R/NW)	WPENDSS3	SensorStrobe チャンネル 3 に対してポストされた書込みの保留ステータス。 このフィールドは、RTC_SS3 に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_SS3 に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_SS3 に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。

表 21-53：RTC_SR5 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
7 (R/NW)	WPENDSS2	SensorStrobe チャンネル 2 に対してポストされた書込みの保留ステータス。 このフィールドは、RTC_SS2 に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_SS2 に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_SS2 に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
6 (R/NW)	WPENDSS1	SensorStrobe チャンネル 1 に対してポストされた書込みの保留ステータス。 このフィールドは、ポストされた RTC_SS1 へのレジスタ書込みが現在保留されて（バッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_SS1 に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_SS1 に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
4 (R/NW)	WPENDSSMSK	SensorStrobe チャンネル・レジスタの RTC マスクに対してポストされた書込みの保留ステータス。 このフィールドは、ポストされた RTC_SSMSK へのレジスタ書込みが現在保留されて（バッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_SSMSK に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_SSMSK に対してポストされた書込みがまだ実行待ち状態なので、この MMR へ新たな書込みをポストすることはできません。
3 (R/NW)	WPENDCR4SS	SensorStrobe チャンネル・レジスタの設定用に RTC 制御 4 に対してポストされた書込みの保留ステータス。 このフィールドは、ポストされた RTC_CR4SS へのレジスタ書込みが現在保留されて（バッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_CR4SS に対して新たにポストされた書込みを受け入れることができます。
		1	RTC_CR4SS への書込み保留。過去に RTC_CR4SS に対してポストされた書込みがまだ実行待ち状態なので、この MMR へ新たな書込みをポストすることはできません。

表 21-53：RTC_SR5 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
2 (R/NW)	WPENDCR3SS	SensorStrobe チャンネル・レジスタの設定用に RTC 制御 3 に対してポストされた書込みの保留ステータス。 このフィールドは、ポストされた RTC_CR3SS へのレジスタ書込みが現在保留されて（バッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_CR3SS に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_CR3SS に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
1 (R/NW)	WPENDCR2IC	入力キャプチャ・チャンネル・レジスタの設定用に RTC 制御 2 に対してポストされた書込みの保留ステータス。 このフィールドは、ポストされた RTC_CR2IC へのレジスタ書込みが現在保留されて（バッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_CR2IC に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_CR2IC に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
0 (R/NW)	WPENDSR3	RTC ステータス 3 レジスタの割込みソースにポストされたクリアの保留ステータス。 このフィールドは、RTC_SR3 の割込みソースにポストされたクリアが現在保留されていて（バッファされてキューに入っていて）、実行待ち状態にあるかどうかを示します。	
		0	RTC は、32kHz 領域に置かれた RTC_SR3 の割込みソースにポストされたクリアを受け入れることができます。
		1	32kHz 領域に置かれている RTC_SR3 の割込みソースに対して過去にポストされたクリアが、まだ実行待ち状態です。追加のクリアは、既存の保留トランザクションにまとめることができます。

RTC ステータス 6

SR6 はステータス・レジスタで、入力キャプチャ・チャンネル IC0、IC2、IC3、IC4 のスナップショットの未読出しステータスを提供します。RTC_SR6 は RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

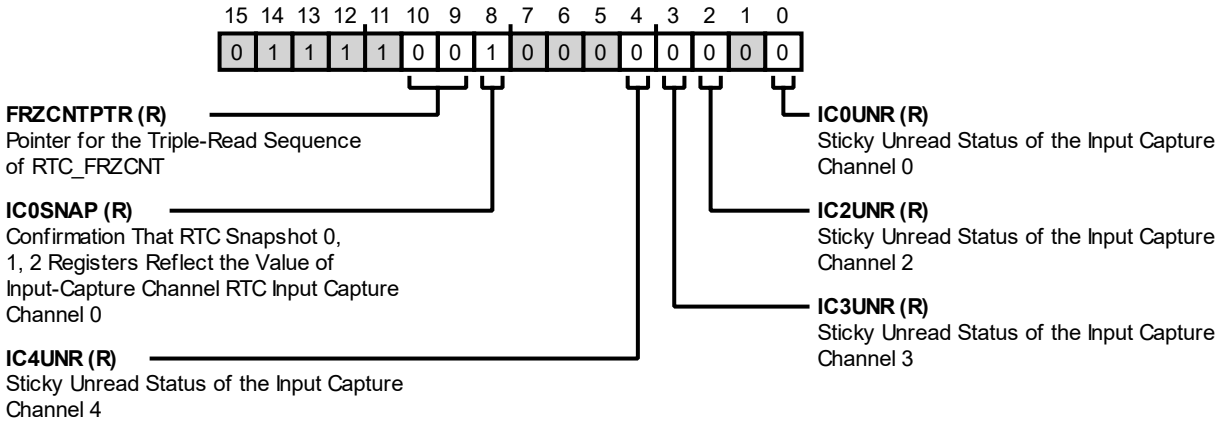


図 21-55：RTC_SR6 レジスタ図

表 21-54：RTC_SR6 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
10:9 (R/NW)	FRZCNTPTR	RTC_FRZCNT のトリプル読出しシーケンス用のポインタ。 このフィールドは、RTC_FRZCNT レジスタのトリプル読出しシーケンスにおける次の読出しのシーケンス番号を示します。 このフィールドは、RTC_GWY に値 0x9376 のソフトウェア・キーを書き込むことによってゼロにできます。	
		0	RTC_FRZCNT の次の読出し時は、以下のような形で行われます。 (i) RTC_FRZCNT の読出しデータは RTC_CNT0 の現在値になります。なおかつ、 (ii) RTC_FRZCNT のその後の 2 回の読出し時には、RTC_CNT2 と RTC_CNT1 の現在値のコヒーレント・スナップショットが取得されて返されます。
		1	RTC_FRZCNT の次の読出しがトリプル読出しシーケンスの 2 回目の読出しとなり、そのシーケンスの最初の RTC_FRZCNT 読出し時に取得された RTC_CNT1 のスナップショットが返されます。
		2	RTC_FRZCNT の次の読出しがトリプル読出しシーケンスの 3 回目の読出しとなり、そのシーケンスの最初の RTC_FRZCNT 読出し時に取得された RTC_CNT2 のスナップショットが返されます。

表 21-54：RTC_SR6 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
8 (R/NW)	IC0SNAP	RTC スナップショット 0、1、2 レジスタが入力キャプチャ・チャンネル（RTC 入力キャプチャ・チャンネル 0）の値を反映していることの確認。 このフラグは、RTC_SNAP0、RTC_SNAP1、および RTC_SNAP2 レジスタが、47 ビット IC0 チャンネルの値、または RTC カウントのソフトウェア起動スナップショットの値を反映しているかどうかを示します。	
		0	ソフトウェア起動スナップショット。RTC_SNAP0、RTC_SNAP1、および RTC_SNAP2 から読み出すことのできる値は、ソフトウェア起動スナップショットによるものです。
		1	IC0 入力キャプチャ・イベントにより取得されるスナップショット。RTC_SNAP0、RTC_SNAP1、および RTC_SNAP2 から読み出すことのできる値は、IC0 入力キャプチャ・イベントによるものです。
4 (R/NW)	IC4UNR	RTC 入力キャプチャ・チャンネル 4 のスティッキー未読出しステータス。 このフィールドはスティッキーな 32kHz ソース・フラグで、入力キャプチャ・チャンネル IC4 に新しい未読出しのスナップショットが存在するかどうかを示します。	
		0	RTC_IC4 には、その入力キャプチャ・チャンネル用の新しい未読出しのスナップショットは格納されていません。
		1	RTC_IC4 には、その入力キャプチャ・チャンネル用の新しい未読出しのスナップショットが格納されています。
3 (R/NW)	IC3UNR	RTC 入力キャプチャ・チャンネル 3 のスティッキー未読出しステータス。 このフィールドはスティッキーな 32kHz ソース・フラグで、入力キャプチャ・チャンネル IC3 に新しい未読出しのスナップショットが存在するかどうかを示します。	
		0	RTC_IC3 には、その入力キャプチャ・チャンネル用の新しい未読出しのスナップショットは格納されていません。
		1	RTC_IC3 には、その入力キャプチャ・チャンネル用の新しい未読出しのスナップショットが格納されています。
2 (R/NW)	IC2UNR	RTC 入力キャプチャ・チャンネル 2 のスティッキー未読出しステータス。 このフィールドはスティッキーな 32kHz ソース・フラグで、入力キャプチャ・チャンネル IC2 に新しい未読出しのスナップショットが存在するかどうかを示します。	
		0	RTC_IC2 には、その入力キャプチャ・チャンネル用の新しい未読出しのスナップショットは格納されていません。
		1	RTC_IC2 には、その入力キャプチャ・チャンネル用の新しい未読出しのスナップショットが格納されています。

表 21-54：RTC_SR6 レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
0 (R/NW)	IC0UNR	RTC 入力キャプチャ・チャンネル 0 のスティッキー未読出しステータス。 このフィールドはスティッキーな 32kHz ソース・フラグで、入力キャプチャ・チャンネル IC0 に新しい未読出しのスナップショットが存在するかどうかを示します。	
		0	IC0 には、その入力キャプチャ・チャンネル用の新しい未読出しのスナップショットは格納されていません。
		1	IC0 には、その入力キャプチャ・チャンネル用の新しい未読出しのスナップショットが格納されています。

RTC ステータス 7

これは、すべての SensorStrobe チャンネルのサンプリングされた GPIO データを提供するステータス・レジスタです。また、これらの値のパターン・マッチのステータスも提供します。RTC_SR7 は RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

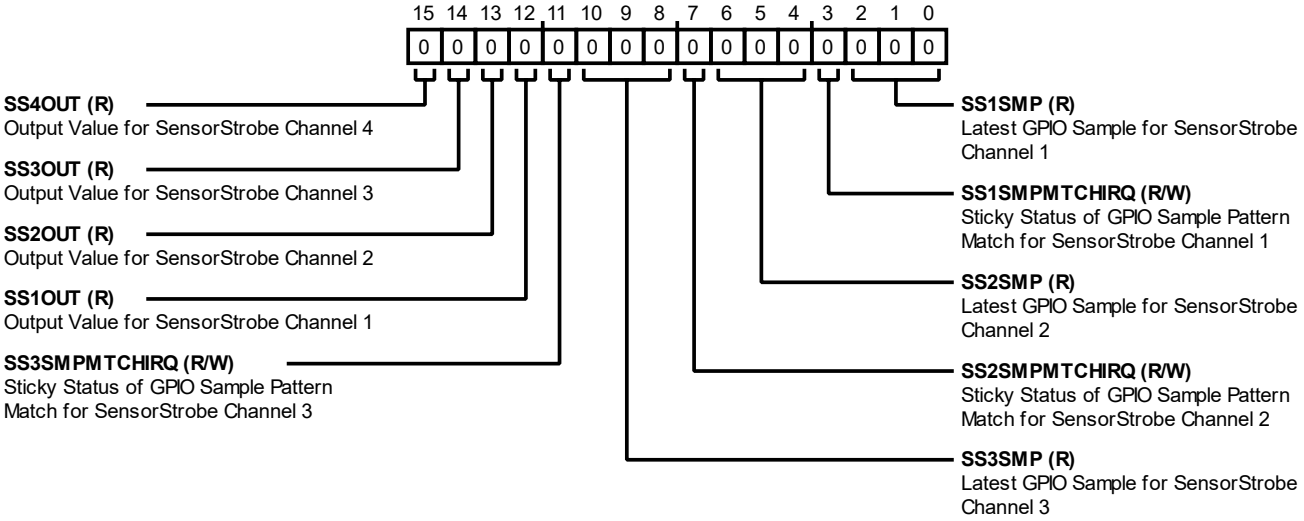


図 21-56：RTC_SR7 レジスタ図

表 21-55：RTC_SR7 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15 (R/NW)	SS4OUT	SensorStrobe チャンネル 4 の出力値。 SensorStrobe チャンネル 4 の出力状態。
14 (R/NW)	SS3OUT	SensorStrobe チャンネル 3 の出力値。 SensorStrobe チャンネル 3 の出力状態。
13 (R/NW)	SS2OUT	SensorStrobe チャンネル 2 の出力値。 SensorStrobe チャンネル 2 の出力状態。
12 (R/NW)	SS1OUT	SensorStrobe チャンネル 1 の出力値。 SensorStrobe チャンネル 1 の出力状態。
11 (R/W)	SS3SMPMTCHIRQ	SensorStrobe チャンネル 3 の GPIO サンプル・パターン・マッチのスティッキー・ステータス。 SensorStrobe チャンネル 3 のパターン・マッチのスティッキー・ステータス。RTC_CR7SSS.SS3SMPPTRN、RTC_CR7SSS.SS3SMPEXP、および前回の RTC_SR7.SS3SMP に基づいて、新しいサンプルが予想パターンと一致する場合。このステータス・ビットをクリアするには、1 を書き込みます。

表 21-55：RTC_SR7 レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応
10:8 (R/NW)	SS3SMP	SensorStrobe チャンネル 3 の最新の GPIO サンプル。 このフィールドは、SensorStrobe チャンネル 3 の GPIO データの最新サンプルを提供します。新しいサンプルが取得されると、上書きされます。
7 (R/W)	SS2SMPMTCHIRQ	SensorStrobe チャンネル 2 の GPIO サンプル・パターン・マッチのスティッキー・ステータス。 SensorStrobe チャンネル 2 のパターン・マッチのスティッキー・ステータス。RTC_CR7SSS.SS2SMPPTRN、RTC_CR7SSS.SS2SMPEXP、および前回の RTC_SR7.SS2SMP に基づいて、新しいサンプルが予想パターンと一致する場合。このステータス・ビットをクリアするには、1 を書き込みます。
6:4 (R/NW)	SS2SMP	SensorStrobe チャンネル 2 の最新の GPIO サンプル。 このフィールドは、SensorStrobe チャンネル 2 の GPIO データの最新サンプルを提供します。新しいサンプルが取得されると、上書きされます。
3 (R/W)	SS1SMPMTCHIRQ	SensorStrobe チャンネル 1 の GPIO サンプル・パターン・マッチのスティッキー・ステータス。 SensorStrobe チャンネル 1 のパターン・マッチのスティッキー・ステータス。RTC_CR7SSS.SS1SMPPTRN、RTC_CR7SSS.SS1SMPEXP、および前回の RTC_SR7.SS1SMP に基づいて、新しいサンプルが予想パターンと一致する場合。このステータス・ビットをクリアするには、1 を書き込みます。
2:0 (R/NW)	SS1SMP	SensorStrobe チャンネル 1 の最新の GPIO サンプル。 このフィールドは、SensorStrobe チャンネル 1 の GPIO データの最新サンプルを提供します。新しいサンプルが取得されると、上書きされます。

RTC ステータス 8

これはステータス・レジスタで、入力キャプチャと出力制御に関係し、なおかつ RTC の 32kHz 常時オン側をソースとするレジスタに対してポストされた書込みと読出しの同期ステータスを提供します。

あるレジスタの WSYNC の値が 0 の場合は、前回そのレジスタに対してポストされた書込みの結果を、CPU がまだ確認できないことを意味します。レジスタの WSYNC 値が 1 の場合、レジスタに対してポストされた書込みは実行が完了していることを意味します。

あるレジスタの RSYNC の値が 0 の場合は、前回そのレジスタに対してポストされた読出しの結果を、CPU がまだ確認できないことを意味します。レジスタの RSYNC 値が 1 の場合、レジスタに対してポストされた読出しは実行が完了していることを意味します。

RTC_SR8 は RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

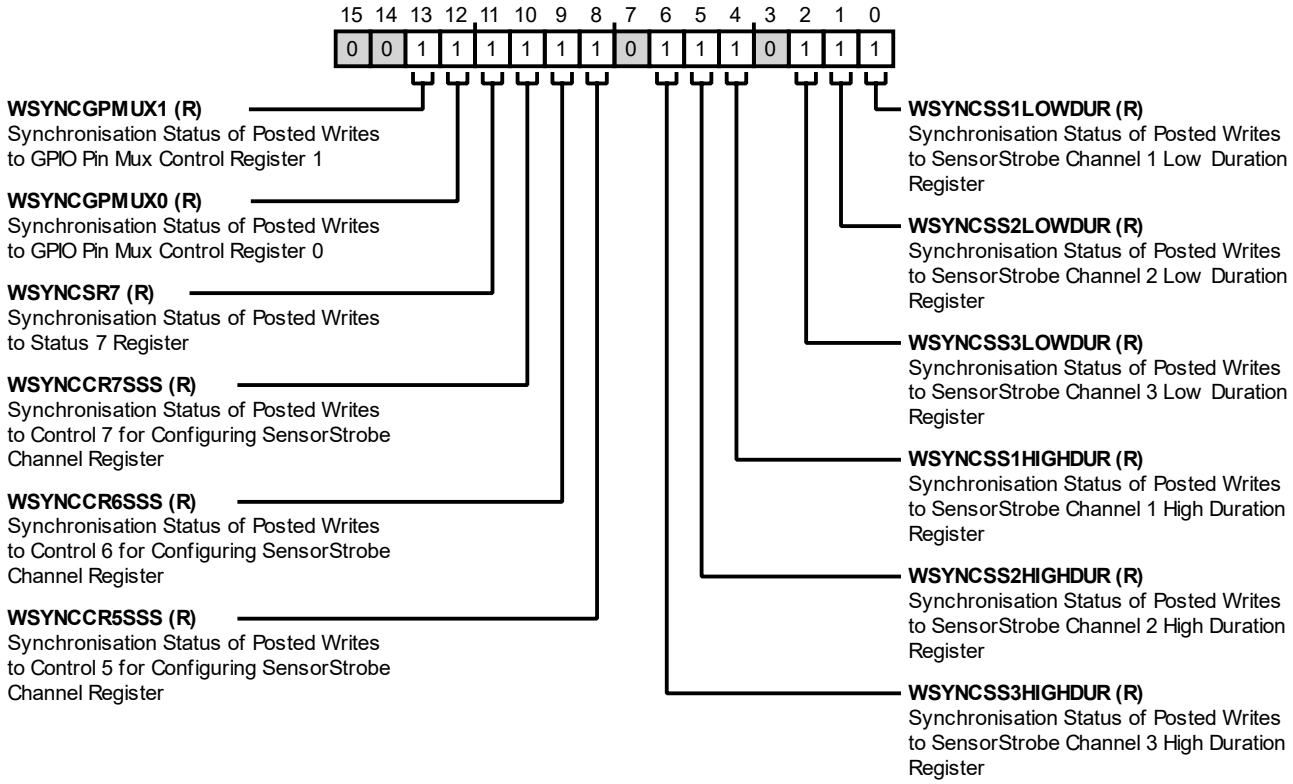


図 21-57：RTC_SR8 レジスタ図

表 21-56：RTC_SR8 レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
13 (R/NW)	WSYNCGPMUX1	GPIO ピン・マルチプレクサ・コントロール・レジスタ 1 に対してポストされた書込みの同期ステータス。 これは、RTC_GPMUX1 に対してポストされた書込みの結果を、CPU が確認できるかどうかを示します。 RTC_SR8.WSYNCGPMUX1 がローの場合、RTC_GPMUX1 に対してポストされた書込みはその時点でキューに入っているか実行中ですが、CPU はまだこのトランザクションの結果を確認できません。RTC_SR8.WSYNCGPMUX1 がハイになって WSYNCINT スティッキー割込みソースがアクティブになると、プロセッサは RTC_GPMUX1 への書込みの結果を確認することができます。	
		0	CPU は、RTC_GPMUX1 に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_GPMUX1 に対してポストされた書込みの結果を確認できます。
12 (R/NW)	WSYNCGPMUX0	GPIO ピン・マルチプレクサ・コントロール・レジスタ 0 に対してポストされた書込みの同期ステータス。 これは、RTC_GPMUX0 に対してポストされた書込みの結果を、CPU が確認できるかどうかを示します。 RTC_SR8.WSYNCGPMUX0 がローの場合、RTC_GPMUX0 に対してポストされた書込みはその時点でキューに入っているか実行中ですが、CPU はまだこのトランザクションの結果を確認できません。RTC_SR8.WSYNCGPMUX0 がハイになって WSYNCINT スティッキー割込みソースがアクティブになると、プロセッサは RTC_GPMUX0 への書込みの結果を確認することができます。	
		0	CPU は、RTC_GPMUX0 に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_GPMUX0 に対してポストされた書込みの結果を確認できます。
11 (R/NW)	WSYNCSR7	ステータス 7 レジスタに対してポストされた書込みの同期ステータス。 これは、RTC_SR7 に対してポストされた書込みの結果を、CPU が確認できるかどうかを示します。 RTC_SR8.WSYNCSR7 がローの場合、RTC_SR7 に対してポストされた書込みはその時点でキューに入っているか実行中ですが、CPU はまだこのトランザクションの結果を確認できません。RTC_SR8.WSYNCSR7 がハイになって WSYNCINT スティッキー割込みソースがアクティブになると、プロセッサは RTC_CR7SSS への書込みの結果を確認することができます。	
		0	CPU は、RTC_SR7 に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_SR7 に対してポストされた書込みの結果を確認できます。

表 21-56：RTC_SR8 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
10 (R/NW)	WSYNCCR7SSS	SensorStrobe チャンネル・レジスタの設定用に制御 7 に対してポストされた書込みの同期ステータス。 これは、RTC_CR7SSS に対してポストされた書込みの結果を、CPU が確認できるかどうかを示します。 RTC_SR8.WSYNCCR7SSS がローの場合、RTC_CR7SSS に対してポストされた書込みはその時点でキューに入っているか実行中ですが、CPU はまだこのトランザクションの結果を確認できません。RTC_SR8.WSYNCCR7SSS がハイになって WSYNCINT スティッキー割込みソースがアクティブになると、プロセッサは RTC_CR7SSS への書込みの結果を確認することができます。	
		0	CPU は、RTC_CR7SSS に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_CR7SSS に対してポストされた書込みの結果を確認できます。
9 (R/NW)	WSYNCCR6SSS	SensorStrobe チャンネル・レジスタの設定用に制御 6 に対してポストされた書込みの同期ステータス。 これは、RTC_CR6SSS に対してポストされた書込みの結果を、CPU が確認できるかどうかを示します。 RTC_SR8.WSYNCCR6SSS がローの場合、RTC_CR6SSS に対してポストされた書込みはその時点でキューに入っているか実行中ですが、CPU はまだこのトランザクションの結果を確認できません。RTC_SR8.WSYNCCR6SSS がハイになって WSYNCINT スティッキー割込みソースがアクティブになると、プロセッサは RTC_CR6SSS への書込みの結果を確認することができます。	
		0	CPU は、RTC_CR6SSS に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_CR6SSS に対してポストされた書込みの結果を確認できます。
8 (R/NW)	WSYNCCR5SSS	SensorStrobe チャンネル・レジスタの設定用に制御 5 に対してポストされた書込みの同期ステータス。 これは、RTC_CR5SSS に対してポストされた書込みの結果を、CPU が確認できるかどうかを示します。 RTC_SR8.WSYNCCR5SSS がローの場合、RTC_CR5SSS に対してポストされた書込みはその時点でキューに入っているか実行中ですが、CPU はまだこのトランザクションの結果を確認できません。RTC_SR8.WSYNCCR5SSS がハイになって WSYNCINT スティッキー割込みソースがアクティブになると、プロセッサは RTC_CR5SSS への書込みの結果を確認することができます。	
		0	CPU は、RTC_CR5SSS に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_CR5SSS に対してポストされた書込みの結果を確認できます。

表 21-56：RTC_SR8 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
6 (R/NW)	WSYNCSS3HIGHDUR	SensorStrobe チャンネル 3 のハイ時間レジスタに対してポストされた書込みの同期ステータス。 これは、RTC_SS3HIGHDUR に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。 RTC_SR8.WSYNCSS3HIGHDUR がローの場合、RTC_SS3HIGHDUR に対してポストされた書込みはその時点でキューに入っているか実行中ですが、CPU はまだこのトランザクションの結果を確認できません。RTC_SR8.WSYNCSS3HIGHDUR がハイになって WSYNCINT スティッキー割込みソースがアクティブになると、プロセッサは RTC_SS3HIGHDUR への書込みの結果を確認することができます。	
		0	CPU は、RTC_SS3HIGHDUR に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_SS3HIGHDUR に対してポストされた書込みの結果を確認できます。
5 (R/NW)	WSYNCSS2HIGHDUR	SensorStrobe チャンネル 2 のハイ時間レジスタに対してポストされた書込みの同期ステータス。 これは、RTC_SS2HIGHDUR に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。 RTC_SR8.WSYNCSS2HIGHDUR がローの場合、RTC_SS2HIGHDUR に対してポストされた書込みはその時点でキューに入っているか実行中ですが、CPU はまだこのトランザクションの結果を確認できません。RTC_SR8.WSYNCSS2HIGHDUR がハイになって WSYNCINT スティッキー割込みソースがアクティブになると、プロセッサは RTC_SS2HIGHDUR への書込みの結果を確認することができます。	
		0	CPU は、RTC_SS2HIGHDUR に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_SS2HIGHDUR に対してポストされた書込みの結果を確認できます。
4 (R/NW)	WSYNCSS1HIGHDUR	SensorStrobe チャンネル 1 のハイ時間レジスタに対してポストされた書込みの同期ステータス。 これは、RTC_SS1HIGHDUR に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。 RTC_SR8.WSYNCSS1HIGHDUR がローの場合、RTC_SS1HIGHDUR に対してポストされた書込みはその時点でキューに入っているか実行中ですが、CPU はまだこのトランザクションの結果を確認できません。RTC_SR8.WSYNCSS1HIGHDUR がハイになって WSYNCINT スティッキー割込みソースがアクティブになると、プロセッサは RTC_SS1HIGHDUR への書込みの結果を確認することができます。	
		0	CPU は、RTC_SS1HIGHDUR に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_SS1HIGHDUR に対してポストされた書込みの結果を確認できます。

表 21-56：RTC_SR8 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
2 (R/NW)	WSYNCSS3LOWDUR	SensorStrobe チャンネル 3 のロー時間レジスタに対してポストされた書込みの同期ステータス。 これは、RTC_SS3LOWDUR に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。 RTC_SR8.WSYNCSS3LOWDUR がローの場合、RTC_SS3LOWDUR に対してポストされた書込みはその時点でキューに入っているか実行中ですが、CPU はまだこのトランザクションの結果を確認できません。RTC_SR8.WSYNCSS3LOWDUR がハイになって WSYNCINT スティッキー割込みソースがアクティブになると、プロセッサは RTC_SS3LOWDUR への書込みの結果を確認することができます。	
		0	CPU は、RTC_SS3LOWDUR に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_SS3LOWDUR に対してポストされた書込みの結果を確認できます。
1 (R/NW)	WSYNCSS2LOWDUR	SensorStrobe チャンネル 2 のロー時間レジスタに対してポストされた書込みの同期ステータス。 これは、RTC_SS2LOWDUR に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。 RTC_SR8.WSYNCSS2LOWDUR がローの場合、RTC_SS2LOWDUR に対してポストされた書込みはその時点でキューに入っているか実行中ですが、CPU はまだこのトランザクションの結果を確認できません。RTC_SR8.WSYNCSS2LOWDUR がハイになって WSYNCINT スティッキー割込みソースがアクティブになると、プロセッサは RTC_SS2LOWDUR への書込みの結果を確認することができます。	
		0	CPU は、RTC_SS2LOWDUR に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_SS2LOWDUR に対してポストされた書込みの結果を確認できます。
0 (R/NW)	WSYNCSS1LOWDUR	SensorStrobe チャンネル 1 のロー時間レジスタに対してポストされた書込みの同期ステータス。 これは、RTC_SS1LOWDUR に対してポストされた書込みの結果を CPU が確認できるかどうかを示します。 RTC_SR8.WSYNCSS1LOWDUR がローの場合、RTC_SS1LOWDUR に対してポストされた書込みはその時点でキューに入っているか実行中ですが、CPU はまだこのトランザクションの結果を確認できません。RTC_SR8.WSYNCSS1LOWDUR がハイになって WSYNCINT スティッキー割込みソースがアクティブになると、プロセッサは RTC_SS1LOWDUR への書込みの結果を確認することができます。	
		0	CPU は、RTC_SS1LOWDUR に対してポストされた書込みの結果をまだ確認できません。
		1	CPU は、RTC_SS1LOWDUR に対してポストされた書込みの結果を確認できます。

RTC ステータス 9

これはステータス・レジスタで、入力キャプチャと出力制御に関係し、RTC の 32kHz 常時オン側をソースとするレジスタに対してポストされた書込みの保留（バッファされてキューに入っている）ステータスを提供します。あるレジスタの WPEND 値が 1 の場合は、そのレジスタに対して前回ポストされた書込みが現在深度 1 のバッファのキューに入っていて、まだ実行が開始されていないことを意味します。したがって WPEND が 1 の場合、RTC は、既存の書込みの実行が開始されてバッファが空くまで、対象レジスタへのそれ以上の書込みを処理できません。レジスタの WPEND が 0 の場合、RTC には、そのレジスタに対して新たにポストされた書込みを受け入れる余地があります。RTC_SR9 は RTC1 にのみ存在します。RTC0 ではこのレジスタ・アドレスは予備で、レジスタのリセット値として読み出されます。

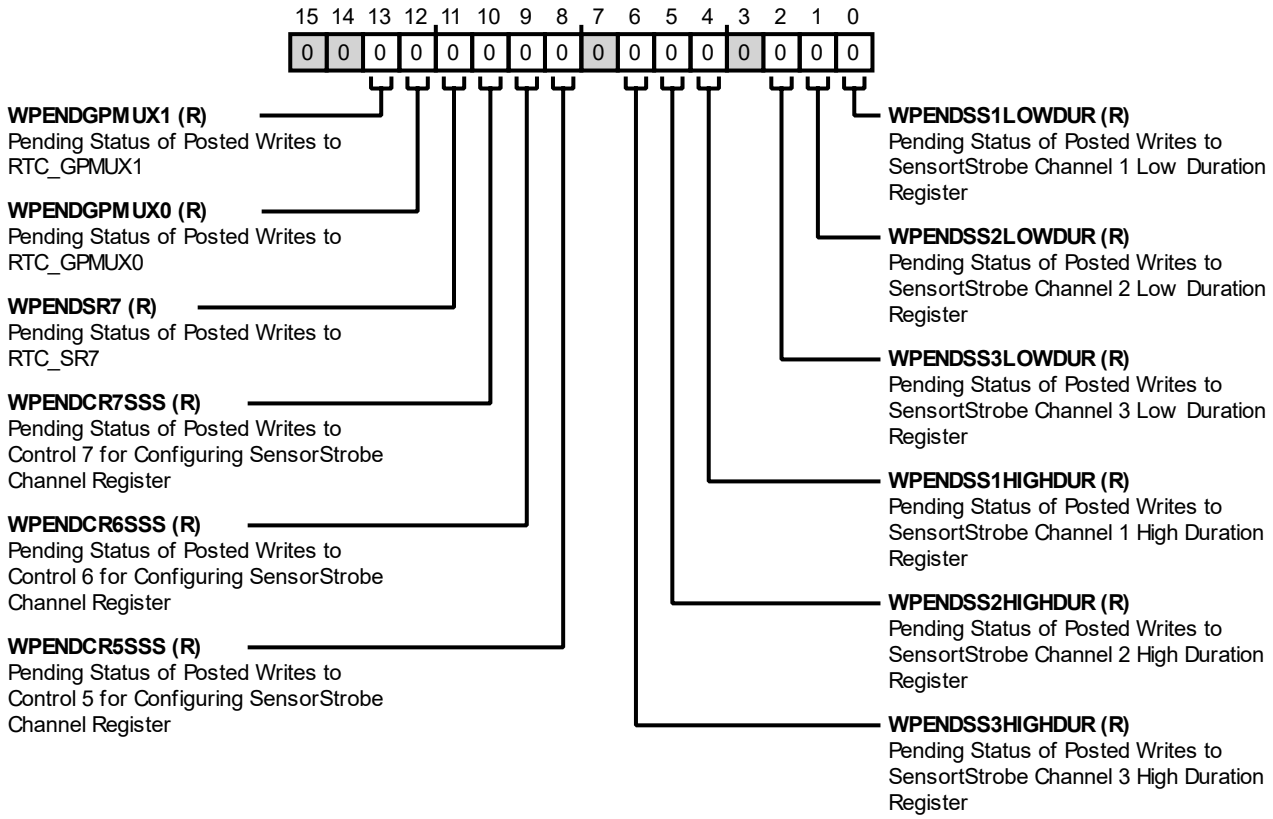


図 21-58：RTC_SR9 レジスタ図

表 21-57：RTC_SR9 レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
13 (R/NW)	WPENDGPMUX1	RTC_GPMUX1 に対してポストされた書込みの保留ステータス。 これは、RTC_GPMUX1 に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_GPMUX1 に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_GPMUX1 に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
12 (R/NW)	WPENDGPMUX0	RTC_GPMUX0 に対してポストされた書込みの保留ステータス。 これは、RTC_GPMUX0 に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_GPMUX0 に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_GPMUX0 に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
11 (R/NW)	WPENDSR7	RTC_SR7 に対してポストされた書込みの保留ステータス。 これは、RTC_SR7 に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点では同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_SR7 に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_SR7 に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
10 (R/NW)	WPENDCR7SSS	SensorStrobe チャンネル・レジスタの設定用に制御 7 に対してポストされた書込みの保留ステータス。 これは、RTC_CR7SSS に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_CR7SSS に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_CR7SSS に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。

表 21-57：RTC_SR9 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
9 (R/NW)	WPENDCR6SSS	SensorStrobe チャンネル・レジスタの設定用に制御 6 に対してポストされた書込みの保留ステータス。 これは、RTC_CR6SSS に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_CR6SSS に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_CR6SSS に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
8 (R/NW)	WPENDCR5SSS	SensorStrobe チャンネル・レジスタの設定用に制御 5 に対してポストされた書込みの保留ステータス。 これは、RTC_CR5SSS に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点で同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_CR5SSS に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_CR5SSS に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
6 (R/NW)	WPENDSS3HIGHDUR	SensorStrobe チャンネル 3 のハイ時間レジスタに対してポストされた書込みの保留ステータス。 これは、RTC_SS3HIGHDUR に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点では同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_SS3HIGHDUR に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_SS3HIGHDUR に対してポストされた書込みがまだ実行待ち状態なので、この MMR へ新たな書込みをポストすることはできません。
5 (R/NW)	WPENDSS2HIGHDUR	SensorStrobe チャンネル 2 のハイ時間レジスタに対してポストされた書込みの保留ステータス。 これは、RTC_SS2HIGHDUR に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点では同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_SS2HIGHDUR に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_SS2HIGHDUR に対してポストされた書込みがまだ実行待ち状態なので、この MMR へ新たな書込みをポストすることはできません。

表 21-57：RTC_SR9 レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
4 (R/NW)	WPENDSS1HIGHDUR	SensorStrobe チャンネル 1 のハイ時間レジスタに対してポストされた書込みの保留ステータス。 これは、RTC_SS1HIGHDUR に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点では同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_SS1HIGHDUR に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_SS1HIGHDUR に対してポストされた書込みがまだ実行待ち状態なので、この MMR へ新たな書込みをポストすることはできません。
2 (R/NW)	WPENDSS3LOWDUR	SensorStrobe チャンネル 3 のロー時間レジスタに対してポストされた書込みの保留ステータス。 これは、RTC_SS3LOWDUR に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点では同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_SS3LOWDUR に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_SS3LOWDUR に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
1 (R/NW)	WPENDSS2LOWDUR	SensorStrobe チャンネル 2 のロー時間レジスタに対してポストされた書込みの保留ステータス。 これは、RTC_SS2LOWDUR に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点では同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_SS2LOWDUR に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_SS2LOWDUR に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。
0 (R/NW)	WPENDSS1LOWDUR	SensorStrobe チャンネル 1 のロー時間レジスタに対してポストされた書込みの保留ステータス。 これは、RTC_SS1LOWDUR に対してポストされたレジスタ書込みが現在保留されて（つまりバッファされてキューに入っていて）実行待ち状態かどうか、したがってこの時点では同じ MMR への書込みをそれ以上受け入れられない状態かどうかを示します。	
		0	RTC は、RTC_SS1LOWDUR に対して新たにポストされた書込みを受け入れることができます。
		1	過去に RTC_SS1LOWDUR に対してポストされた書込みがまだ実行待ち状態なので、この MMR に新たな書込みをポストすることはできません。

RTC トリム

RTC_TRM には、必要な精度で時間を追跡するために、RTC カウント値の定期調整用のトリム値と間隔が格納されます。トリミングの有効化と無効化は RTC_CR0.TRMEN ビットを介して行われます。トリミングが行われるようにするには、RTC 用のグローバル・イネーブル RTC_CR0.CNTEN もアクティブにする必要があります。

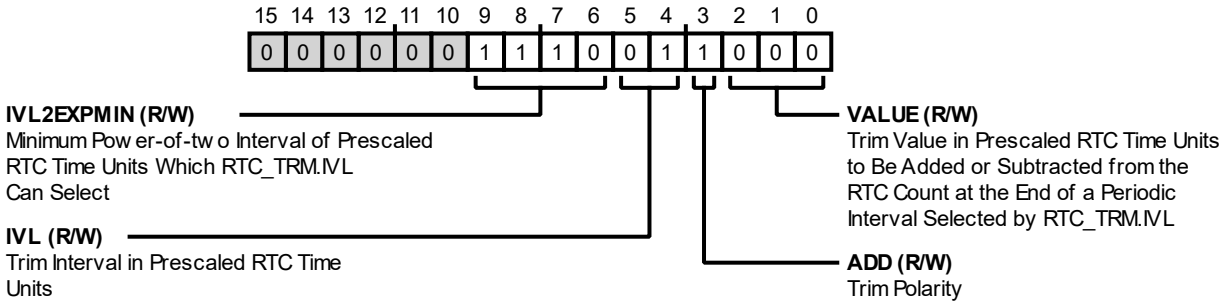


図 21-59：RTC_TRM レジスタ図

表 21-58：RTC_TRM レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
9:6 (R/W)	IVL2EXPMIN	2 のべき乗で表されるプリスケール RTC 時間単位の最小間隔。この値は RTC_TRM.IVL で選択できます。 RTC_TRM.IVL2EXPMIN は、RTC がいつでも使用できるトリム間隔の範囲を設定します。範囲は 2 の（RTC_TRM.IVL2EXPMIN + 0、1、2、または 3）乗で、これは RTC_TRM.IVL によって選択します。 サポートされている RTC_TRM.IVL2EXPMIN の最小値は 2 で、サポートされている最大値（およびデフォルト値）は 14 です。RTC の設定が［2,14］の範囲を外れている場合は、デフォルト値の 14 が使われます。	
5:4 (R/W)	IVL	プリスケール RTC 時間単位の時間間隔。 RTC_TRM.IVL は、RTC カウントに対するプリスケール RTC 時間単位 RTC_TRM.VALUE の定期調整の間隔を指定します。	
		0	トリム間隔は 2^RTC_TRM.IVL2EXPMIN プリスケール RTC 時間単位。カウント時間に対するプリスケーリングが 1Hz の場合、これは 4 時間 33 分 04 秒に相当します。
		1	トリム間隔は 2^（RTC_TRM.IVL2EXPMIN + 1）プリスケール RTC 時間単位。カウント時間に対するプリスケーリングが 1Hz の場合、これは 9 時間 06 分 08 秒に相当します。
		2	トリム間隔は 2^（RTC_TRM.IVL2EXPMIN + 2）プリスケール RTC 時間単位。カウント時間に対するプリスケーリングが 1Hz の場合、これは 18 時間 12 分 16 秒に相当します。
		3	トリム間隔は 2^（RTC_TRM.IVL2EXPMIN + 3）プリスケール RTC 時間単位。カウント時間に対するプリスケーリングが 1Hz の場合、これは 36 時間 24 分 32 秒に相当します。

表 21-58：RTC_TRM レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
3 (R/W)	ADD	トリム極性。 RTC_TRM.ADD は、RTC_TRM.VALUE が RTC カウントの正の調整（加算）か負の調整（減算）かを指定します。	
		0	RTC_TRM.IVL によって定義される周期で、RTC カウントから RTC_TRM.VALUE プリスケール RTC 時間単位を差し引きます（待機する）。
		1	RTC カウントにトリム値を加えます。RTC_TRM.IVL によって定義される周期で、RTC カウントに RTC_TRM.VALUE プリスケール RTC 時間単位を加えます。
2:0 (R/W)	VALUE	RTC_TRM.IVL によって選択された定期調整間隔終了時に RTC カウントに加える（あるいは RTC カウントから引く）、プリスケール RTC 時間単位のトリム値。 RTC カウントに対する+/− 0、1、2、3、4、5、6、7 プリスケール RTC 時間単位のトリム調整（RTC_TRM.IVL によって定義される周期で実行）は、RTC_TRM.VALUE を使って指定することができます。 トリムと同時に RTC 割込みイベントを発生させるように設定する場合、割込み動作は次のようになります。正のトリムによって割込み時間がトリム（スキップ）された場合は、トリム終了時に割込みが生成されます。割込みイベントに対してターゲット時間に RTC カウントを止める負のトリムの場合は、最初にターゲット時間に達した時点で割込みが生成されます。	
		0	時間間隔終了時に RTC カウントの調整を行いません。
		1	RTC カウントに 1 プリスケール時間単位を加えます。あるいは RTC カウントから 1 プリスケール時間単位を差し引きます。
		2	RTC カウントに 2 プリスケール時間単位を加えます。あるいは RTC カウントから 2 プリスケール時間単位を差し引きます。
		3	RTC カウントに 3 プリスケール時間単位を加えます。あるいは RTC カウントから 3 プリスケール時間単位を差し引きます。
		4	RTC カウントに 4 プリスケール時間単位を加えます。あるいは RTC カウントから 4 プリスケール時間単位を差し引きます。
		5	RTC カウントに 5 プリスケール時間単位を加えます。あるいは RTC カウントから 5 プリスケール時間単位を差し引きます。
		6	RTC カウントに 6 プリスケール時間単位を加えます。あるいは RTC カウントから 6 プリスケール時間単位を差し引きます。
		7	RTC カウントに 7 プリスケール時間単位を加えます。あるいは RTC カウントから 7 プリスケール時間単位を差し引きます。

22 タイマー（TMR）

ADuCM4050 MCU には 3 個の汎用タイマー（GP タイマー）があります。それぞれ 16 ビットのアップ／ダウン・カウンタと最大 8 ビットのクロック・プリスケーラを搭載し、1、4、16、64、256 のプリスケール・オプションを備えています。個別のクロック供給ブロック内で、タイマーに最大 4 つのクロック源を選択できます。

タイマーの最大タイムアウト範囲は（$2^{Counter\_width}$/クロック周波数）です。

タイマー・クロック周波数が 26MHz の場合、最大タイムアウト範囲は $2^{(16+8)}$/26MHz = 0.6 秒です。

タイマー・クロック周波数が 32kHz の場合、最大タイムアウト範囲は $2^{(16+8)}$/32kHz = 512 秒です。

これらのタイマーの使用例としては、以下があります。

- 1µs クロック
- ファームウェアに 50～60µs のリファレンス・タイムベースを供給するための SysTick タイマー
- データ・レート・カウンタ

  データ・レートは、ワイヤレス・プロトコルに応じて 100bps～400Kbps の範囲で設定できます。
- PWM 生成
- PWM 復調

TMR 機能

ADuCM4050 MCU で使用される汎用タイマーは、以下の機能を備えています。

- 自走モードと周期モードの 2 つの動作モードをサポート
- PWM 生成機能では、アイドル状態をロジック 0 またはロジック 1 に設定可能
- PWM 出力をトグル・モードまたはマッチ・モードで生成
- 選択可能な IRQ 源のアサーションをタイマーのリセットに使用可能
- リセット機能とキャプチャ機能を使用して、3 つのタイマーすべてで PWM 復調を実行可能

タイマーには、自走モードと周期モードの 2 つの動作モードがあります。自走モードでは、カウンタは最大値／最小値からゼロ／フルスケールまでデクリメント／インクリメントし、最大値／最小値から再スタートします。

周期モードでは、カウンタはロード・レジスタ TMR_LOAD の値からゼロ／フルスケールまでデクリメント／インクリメントし、ロード・レジスタに格納された値から再スタートします。

カウンタの値は、その値レジスタ（TMR_ACURCNT.VALUE）にアクセスすることにより、常時読出し可能です。これは同期クロックを前提としており、非同期の（同期していない）タイマー値を直接返すことで同期遅延を回避しています。他に、TMR_CURCNT.VALUE を読み出すことも可能で、この場合は若干遅延したタイマー値を返します。タイマーとコアが異なるクロックで動作しているときには、TMR_ACURCNT.VALUE の読出しはサポートされていません。この場合、戻り値は定義されません。

TMR_CTL.EN ビットに 1 を書き込むと、アップ／ダウン・カウンタが開始されます。カウント・ダウン時にはカウンタ値がゼロになるたびに、またはカウント・アップ時にはカウンタ値がフルスケールに達するたびに、IRQ が生成されます。TMR_CLRINT.TIMEOUT ビットに 1 を書き込むと、IRQ をクリアできます。IRQ は、選択したイベントがイベント入力で発生したときにもセットされます。

PWM 生成機能は、TMR_PWMCTL.IDLESTATE ビットへの書込みによって、ロジック 0 またはロジック 1 でアイドルになるように構成できます。オプションのマッチ値を TMR_PWMMATCH.VALUE ビットに書き込むと、TMR_PWMCTL.MATCH ビットへの書込みによりこの値が有効になります。

PWM 出力は、トグル・モードまたはマッチ・モードのいずれかで生成されます。トグル・モードでは、PWM 出力はタイマーのゼロ／フルスケールでトグルするため、設定可能な周期で 50%のデューティ・サイクルが得られます。マッチ・モードでは、設定可能なデューティ・サイクルと設定可能な周期の PWM 出力が得られます。マッチ・モードでは、PWM 出力は TMR_CTL レジスタに設定されたアイドル状態から開始し、タイマーとマッチ値が等しくなるとアサートされ、タイマーがゼロ／フルスケールに達するとアサート解除されて再びアイドル状態になります。

TMR_CTL.RSTEN ビットをセットした場合､選択可能な IRQ 源のアサーションをタイマーのリセットに使用できます。ブロックへの入力には、トリガ・ソースとして 40 種類のイベントから 1 つを選択できます。この割込み源は、キャプチャ機能の一部としても使用できます。このキャプチャ機能は、選択した IRQ 源のアサーションによってもトリガされます。トリガされると、現在のタイマー値が TMR_CAPTURE.VALUE ビットにコピーされ、タイマーは実行を継続します。この機能を使用すると、イベントのアサーションを高精度で判断できます。

TMR 機能の説明

ここでは、ADuCM4050 MCU で使用される汎用タイマーの機能を説明します。

TMR ブロック図

この図は、ADuCM4050 MCU で使用される汎用タイマーを示しています。

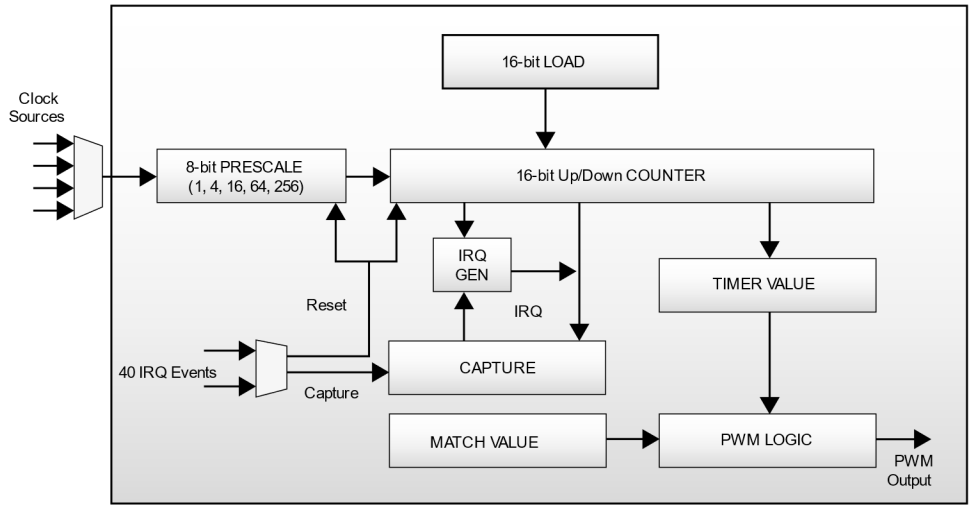


図 22-1：タイマー1 のブロック図

TMR 動作モード

ADuCM4050 MCU で使用される汎用タイマーは、以下の動作モードを備えています。

自走モード

自走モードでは、タイマーは TMR_CTL.EN ビットへの書込みによって開始します。カウント・アップ／ダウン（TMR_CTL.UP ビットで設定）すると、タイマーはゼロ／フルスケールからフルスケール／ゼロまでインクリメントします。フルスケールは、2^{16} − 1 または 16 進数形式の 0xFFFF です。フルスケール（またはゼロ）に達すると、タイムアウト割込みが発生し、TMR_STAT.TIMEOUT ビットがセットされます。TMR_STAT.TIMEOUT ビットをクリアするには、ユーザ・コードで TMR_CLRINT.TIMEOUT ビットに 1 を書き込む必要があります。タイムアウト割込みが発生すると、タイマーに最大値／最小値が再ロードされます。TMR_CTL.RLD ビットをセットすると、TMR_CLRINT.TIMEOUT ビットに 1 を書き込んだときにもタイマーへの再ロードが行われます。

周期モード

周期モードでは、タイマーを使用可能にする前に最初の TMR_LOAD.VALUE 値をロードする必要があります。タイマーは、TMR_CTL.EN ビットへの書込みによって開始されます。カウンタの値は、TMR_LOAD.VALUE レジスタに格納されている値からフルスケールまでインクリメントします。あるいは、TMR_CTL.UP の設定（カウント・アップ／ダウン）に応じて、TMR_LOAD.VALUE レジスタに格納された値からゼロまでデクリメントします。カウンタがフルスケールまたはゼロに達すると、タイマーは割込みを生成します。TMR_LOAD.VALUE がカウンタに再ロードされ、カウント・アップ／ダウンを継続します。タイマーは、TMR_LOAD.VALUE レジスタを変更する前にディスエーブルする必要があります。デフォルトでは、IRQ の生成時にカウンタは自動的に再ロードされます。TMR_CTL.RLD ビットを 1 に設定すると、ユーザ・コードが TMR_CLRINT.TIMEOUT ビットに書込みを行ったときにも、カウンタは再ロードされます。ロード値を細かいタイミングで変更できるようにするため、非同期書込み用として TMR_ALOAD が用意されています。TMR_ALOAD に書き込むことにより、同期ロジックがバイパスされ、ロード値をきめ細かく制御できます。タイマーがイネーブルされ、コアとは異なるクロックで動作しているときに TMR_ALOAD レジスタを変更すると、未定

義の結果が生じることがあります。タイマーのイネーブルのセットまたはクリアに続いてタイマー・レジスタに書き込む前に、TMR_STAT を読み出す必要があります。TMR_STAT.PDOK ビットがゼロを返すと、レジスタを変更できます。これにより、タイマーは、コアとタイマーのクロック・ドメイン間でタイマー制御を同期させることができます。代表的な同期時間は 2 タイマー・クロック周期です。TMR_CTL.UP または TMR_CTL.MODE ビットの変更は、タイマーのディスエーブル時に実行する必要があります。

TMR_CURCNT.VALUE にはコア・クロックに同期した有効な値が入っており、常時読出し可能です。TMR_CTL レジスタで、カウンタのイネーブル、モードやプリスケール値の選択、イベント・キャプチャ機能の制御を行います。

トグル・モード

トグル・モードでは、PWM は設定可能な周期で 50%のデューティ・サイクル出力を提供します。タイマーによってタイムアウト割込みが発生すると、PWM 出力が反転します。このため、周期は選択したクロックとプリスケーラによって定義されます。タイマーが周期モードで実行している場合、PWM 出力は TMR_LOAD.VALUE 値にも依存します。タイムアウト・イベントは、タイマー期間（コア・クロック期間よりも長い）の終了時に評価されます。

次の図は、タイマーをカウント・アップ／ダウンに設定しているときのトグル・モードでの PWM 出力を示しています。

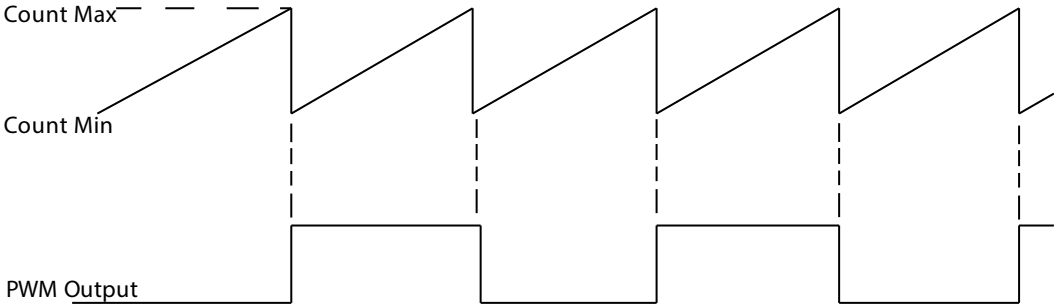


図 22-2：タイマーをカウント・アップに設定しているときのトグル・モードでの PWM 出力

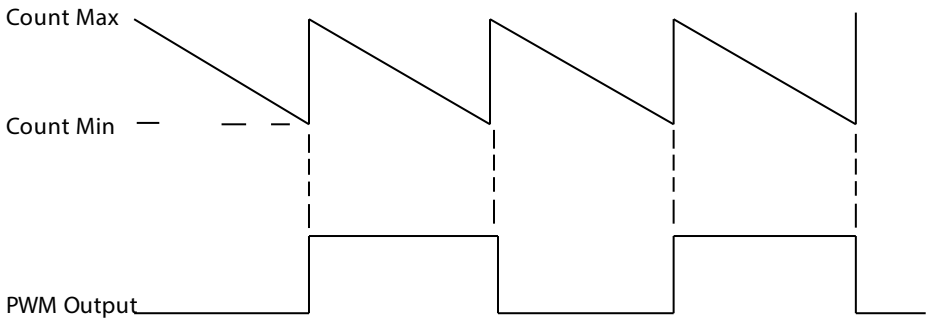


図 22-3：タイマーをカウント・ダウンに設定しているときのトグル・モードでの PWM 出力

マッチ・モード

マッチ・モードでの PWM 出力では、デューティ・サイクルと周期を設定可能です。トグル・モードと同様に、周期は選択したクロックとプリスケーラ、および TMR_LOAD.VALUE（タイマーが周期モードで実行している場合）によって定義されます。デューティ・サイクルは、TMR_PWMMATCH.VALUE 値によって定義されます。

タイマーのカウンタ値が TMR_PWMMATCH.VALUE に格納されている値と等しい場合、PWM 出力がアサートされます。タイマーはアサートを維持し、ゼロ／フルスケールに達するとアサート解除されます。タイマー期間（コア・クロック期間よりも長い）の終了時にマッチ・イベントおよびタイムアウト・イベントが評価されます。

マッチ・イベントは、同じサイクルでトリガされた場合はタイムアウト・イベントよりも優先されるため、周期モードでの動作時は TMR_PWMMATCH.VALUE を TMR_LOAD.VALUE と同じ値に設定し、自走モード時にはゼロ／フルスケールに設定することにより、PWM を 100%デューティ・サイクルに設定できます。0%のデューティ・サイクルを設定できるのは周期モードでの実行時に限られ、TMR_PWMMATCH.VALUE をタイマー・カウンタの範囲外（カウント・ダウン時は TMR_LOAD.VALUE + 1 に、カウント・アップ時は TMR_LOAD.VALUE − 1）に設定します。

以下の図は、タイマーをカウント・アップ／ダウンに設定し、PWM アイドル状態を 0 に設定した場合の、マッチ・モードでの PWM 出力を示しています。

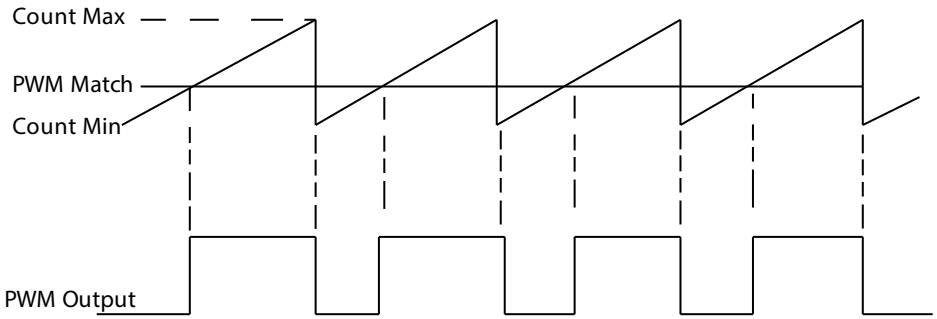


図 22-4：タイマー設定がカウント・アップで、PWM アイドル状態が 0 の場合のマッチ・モードでの PWM 出力

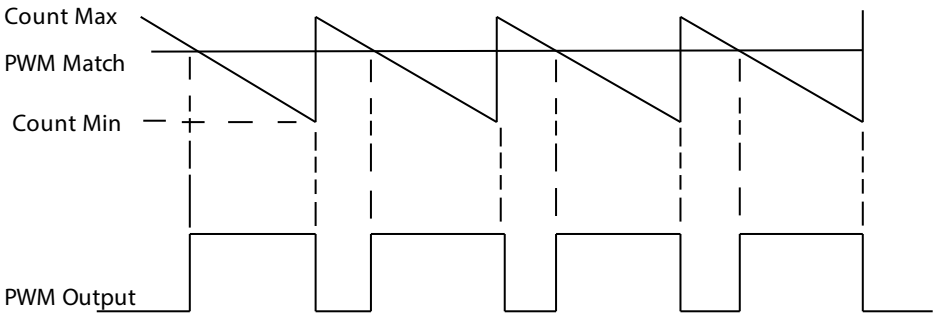


図 22-5：タイマー設定がカウント・ダウンで、PWM アイドル状態が 0 の場合のマッチ・モードでの PWM 出力

ある与えられたクロック源、プリスケーラ、および TMR_LOAD.VALUE 値では、トグル・モードの PWM 周期はマッチ・モードの周期の 2 倍になります。これは、トグル・モードではタイマー・カウンタ周期ごとに反転が 1 回しか行われないのに対して、マッチ・モードでは PWM 出力のアサートとアサート解除の両方が行われるためです。タイマー実行中の TMR_CTL および TMR_PWMMATCH.VALUE レジスタへの書込みはサポートされています。ただし、次のマッチ・イベントまたはタイムアウト・イベントまで、TMR_CTL への変更は PWM 出力に反映されません。これにより、

PWM デューティ・サイクルを決めた上で PWM 動作を変更できます。TMR_CLRINT に書き込むか、タイマーをイネーブル／ディスエーブルすることで、PWM 出力を新たに書き込んだ設定と一致するように強制することができます。

注：PWM はトグル・モードまたはマッチ・モードで動作するように構成できます。いずれのモードでも、PWM 出力を使用するには、GPIO 制御モジュールで適切なマルチプレクサ設定を選択する必要があります。

PWM 変調

PWM 生成機能では、それぞれの PWM 制御レジスタの対応するビットに書き込むことにより、アイドル状態をロジック 0 またはロジック 1 に設定できます。オプションのマッチ値は、それぞれの PWM マッチ・レジスタに書込み可能で、PWM 制御レジスタの別のビットに書き込むことによって有効になります。PWM 出力は、トグル・モードまたはマッチ・モードのいずれかで生成されます。トグル・モードでは、PWM 出力はタイマーのゼロ／フルスケールでトグルするため、設定可能な周期で 50%のデューティ・サイクルが得られます。マッチ・モードでは、設定可能なデューティ・サイクルと設定可能な周期の PWM 出力が得られます。マッチ・モードでは、PWM 出力は PWM 制御レジスタで設定されたアイドル状態から開始し、タイマーとマッチ値が等しくなるとアサートされ、タイマーがゼロ／フルスケールに達するとアサート解除されて再びアイドル状態になります。

このシーケンスは以下のとおりです。

1. PWM レジスタを設定します。
2. タイマーをイネーブルします。

PWM 復調

PWM 復調は、イベント・ロジックでタイマー・リセット機能とキャプチャ機能の両方を有効にすることで実行できます。これにより、PWM 入力のハイ・レベルとロー・レベルのカウント値を別々に読み出すことができます。実際には、MCU は適切な PWM（GPIO）割込み源を選択し、PWM パルスの立上がりエッジで割込みをトリガします。キャプチャを検出すると、カウンタがリセットされ、MCU への割込みを実行します。MCU は、PWM パルスの立下がりエッジでトリガするように（エッジ発生前に）割込みを変更します。この割込みにより、ハイ・レベル・カウントが TMR_CAPTURE.VALUE にキャプチャされてカウンタがリセットされ、更新されたキャプチャ値で MCU に再度割込みが実行されます。MCU はこのキャプチャされたカウントを読み出し、再び正のエッジで動作するように（エッジ発生前に）割込みを修正します。この割込みにより、ロー・レベル・カウントが TMR_CAPTURE.VALUE にキャプチャされてカウンタがリセットされ、MCU に再度割込みが実行されます。MCU はこのキャプチャされたカウントを読み出すことができます。両方の読出し値を比較することにより、PWM 値が示されます。PWM パルス幅は、様々な入力エッジ間で必要とされる同期遅延、IRQ 遅延、およびソフトウェア設定のオーバーヘッドに対応できる十分に広い幅である必要があります。

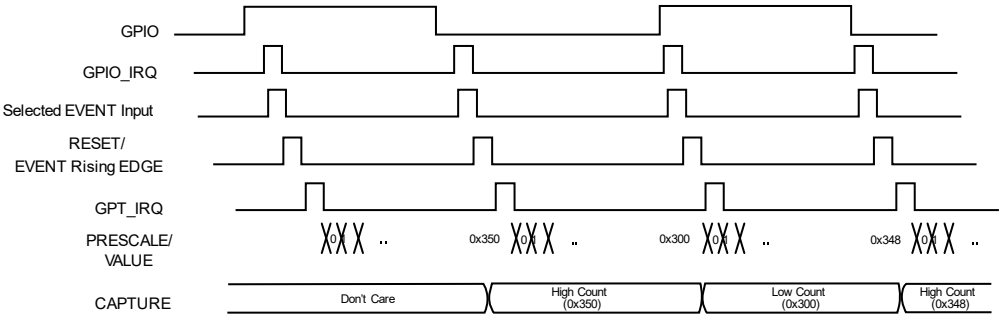


図 22-6：PWM 復調波形

クロック選択

4 つのタイマーのそれぞれで、4 通りのクロックから選択できます。選択したクロックは、プリスケーラとタイマー・カウンタ・ロジックの周波数を制御するのに使用されます。使用するクロックは、TMR_CTL.CLK ビットで選択します。使用するのは同期クロックのみです。32kHz クロックを選択した場合、最初にシステム・クロックに同期されます。クロック構成の設定は、クロック制御モジュールの一部です。詳細については、システム・クロックを参照してください。

表 22-1：クロック源

Clock Select	GPT Clock Source
00	PCLK
01	HFOSC
10	LFOSC
11	LFXTAL

イベントのキャプチャ

40 個の割込みイベントを汎用タイマーでキャプチャできます。汎用タイマーに関連付けられた 40 イベントのいずれか 1 つで、40 ビットの TMR_CURCNT レジスタを 16 ビットの TMR_CAPTURE レジスタにキャプチャさせることができます。TMR_CTL.EVTRANGE には、キャプチャする 40 個のイベントの 1 つを選択する 6 ビットのフィールドがあります。

選択した IRQ が発生すると、TMR_CURCNT レジスタが TMR_CAPTURE.VALUE レジスタにコピーされます。TMR_STAT.CAPTURE ビットがセットされます。TMR_CLRINT.EVTCAPT ビットに 1 を書き込むことで、IRQ はクリアされます。TMR_CAPTURE.VALUE ビットもその値を保持し、TMR_CLRINT.EVTCAPT ビットが 1 に書き込むまでは上書きできません。

表 22-2：キャプチャおよびリセット・イベント機能

Event Select Bits (EVTSEL)	GPT0 Capture Source	GPT1 Capture Source	GPT2 Capture Source
0	RTC1/lfxtal_err	External Interrupt 0	External Interrupt 1
1	External Interrupt 3	RTC0	UART0

表 22-2：キャプチャおよびリセット・イベント機能（続き）

Event Select Bits (EVTSEL)	GPT0 Capture Source	GPT1 Capture Source	GPT2 Capture Source
2	SPI0	SPI1	SPI2
3	SPORT0B	GPIO-IntA	GPIO-IntB
4	I2C0 Slave	P0_8 Input	Reserved
5	Reserved	DMA Error	DMA Channel 0 Done
6	DMA Channel 2 Done	DMA Channel 3 Done	DMA Channel 4 Done
7	DMA Channel 6 Done	DMA Channel 7 Done	DMA Channel 8 Done
8	DMA Channel 10 Done	DMA Channel 11 Done	DMA Channel 12 Done
9	DMA Channel 14 Done	DMA Channel 15 Done	DMA Channel 16 Done
10	DMA Channel 18 Done	DMA Channel 19 Done	DMA Channel 20 Done
11	DMA Channel 22 Done	DMA Channel 23 Done	DMA Channel 24 Done
12	DMA Channel 26 Done	Flash Controller	PLL
13	VREG Over	Battery Voltage Range	Crypto
14	Reserved	Reserved	Reserved
15	Reserved	Reserved	Reserved
16	Beeper	Timer0	Timer1
17	TimerRGB	WDT	RTC1
18	External Interrupt 0	RTC1	External Interrupt 0
19	External Interrupt 1	External Interrupt 1	External Interrupt 2
20	External Interrupt 2	External Interrupt 2	External Interrupt 3
21	RTC	External Interrupt 3	RTC
22	UART0	UART0	UART1
23	UART1	UART1	SPI0
24	SPI1	SPI0	SPI1
25	SPI2	SPI2	SPORT0A
26	SPORT0A	SPORT0A	SPORT0B
27	GPIO-IntA	SPORT0B	GPIO-IntA
28	GPIO-IntB	GPIO-IntB	I2C0 Master
29	I2C0 Master	I2C0 Master	I2C0 Slave
30	P0_8 Input	I2C0 Slave	P0_8 Input
31	Reserved	Reserved	Reserved
32	Reserved	Reserved	Reserved

表 22-2：キャプチャおよびリセット・イベント機能（続き）

Event Select Bits (EVTSEL)	GPT0 Capture Source	GPT1 Capture Source	GPT2 Capture Source
33	DMA Error	Reserved	DMA Error
34	DMA Channel 0 Done	DMA Channel 0 Done	DMA Channel 1 Done
35	DMA Channel 1 Done	DMA Channel 1 Done	DMA Channel 2 Done
36	DMA Channel 3 Done	DMA Channel 2 Done	DMA Channel 3 Done
37	DMA Channel 4 Done	DMA Channel 4 Done	DMA Channel 5 Done
38	DMA Channel 5 Done	DMA Channel 5 Done	DMA Channel 6 Done
39	DMA Channel 7 Done	DMA Channel 6 Done	DMA Channel 7 Done

TMR の割込みと例外

詳細については、イベント（割込みと例外）を参照してください。

TMR 電源モードの動作

3 つの汎用タイマーのすべてのレジスタは、休止モードで保持されます。

ADuCM4050 TMR レジスタの説明

汎用タイマー（TMR）には以下のレジスタがあります。

表 22-3：ADuCM4050 TMR レジスター覧

レジスタ名	説明
TMR_ALOAD	16 ビット・ロード値、非同期
TMR_ACURCNT	16 ビット・タイマー値、非同期
TMR_CAPTURE	キャプチャ
TMR_CLRINT	割込みクリア
TMR_CTL	制御
TMR_EVENTSELECT	タイマー・イベント選択レジスタ
TMR_LOAD	16 ビット・ロード値
TMR_PWMCTL	PWM 制御レジスタ
TMR_PWMMATCH	PWM マッチ値
TMR_STAT	ステータス
TMR_CURCNT	16 ビット・タイマー値

16 ビット・ロード値、非同期

同期クロック源を選択している場合にのみ使用します（TMR_CTL.CLK = 00）。

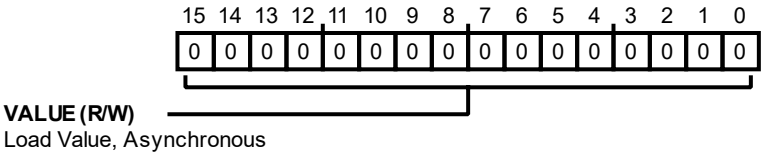


図 22-7：TMR_ALOAD レジスタ図

表 22-4：TMR_ALOAD レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	ロード値、非同期。 周期モードを選択している場合（TMR_CTL.MODE = 1）、この値がアップ／ダウン・カウンタに定期的にロードされます。このビットフィールドへの書込みは、これ以外の場合では必要とされるクロック同期ロジックをバイパスすることによって、タイマーが PCLK で動作することを利用しています。

16 ビット・タイマー値、非同期

同期クロック源を選択している場合にのみ使用します（TMR_CTL.CLK = 00）。

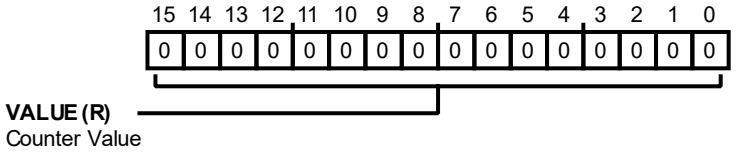


図 22-8：TMR_ACURCNT レジスタ図

表 22-5：TMR_ACURCNT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	VALUE	カウンタ値。 現在のアップ／ダウン・カウンタ値を反映します。このビットフィールドの読出しは、これ以外の場合では必要とされるクロック同期ロジックをバイパスすることによって、タイマーが PCLK で動作することを利用しています。

キャプチャ

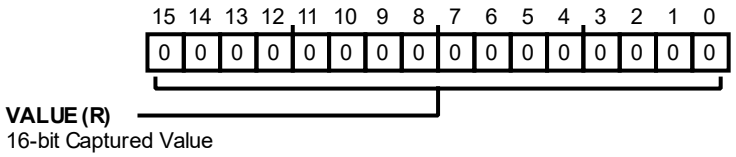


図 22-9：TMR_CAPTURE レジスタ図

表 22-6：TMR_CAPTURE レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	VALUE	16 ビットのキャプチャ値。 このビットフィールドは、ユーザ・コードによって TMR_CLRINT.EVTCAPT がセットされるまでその値を保持します。TMR_CLRINT.EVTCAPT に書き込まない限り、別のイベントが発生してもこのビットフィールドが上書きされることはありません。

割込みクリア

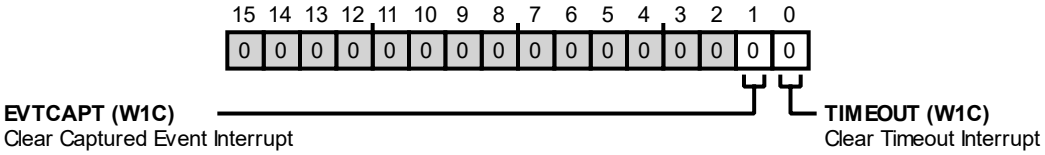


図 22-10：TMR_CLRINT レジスタ図

表 22-7：TMR_CLRINT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
1 (RX/W1C)	EVTCAPT	キャプチャ・イベント割込みクリア。 このビットは、キャプチャ・イベント割込みをクリアするのに使用します。	
		0	影響なし
		1	キャプチャ・イベント割込みをクリア
0 (RX/W1C)	TIMEOUT	タイムアウト割込みクリア。 このビットは、タイムアウト割込みをクリアするのに使用します。	
		0	影響なし
		1	タイムアウト割込みをクリア

制御

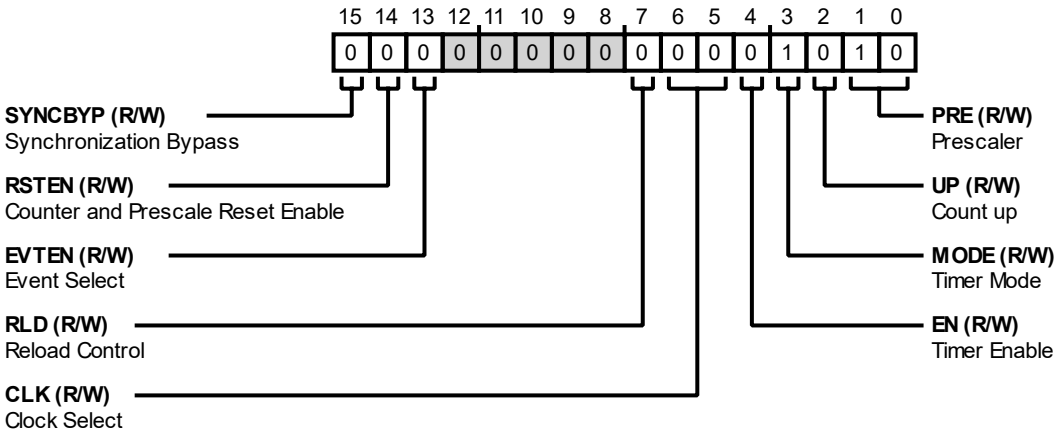


図 22-11：TMR_CTL レジスタ図

表 22-8：TMR_CTL レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
15 (R/W)	SYNCBYP	同期バイパス。 ブロック内の同期ロジックをバイパスするのに使用します。同期クロック、つまり、コア・クロック（PCLK）をタイマーのクロック源として選択している場合にのみ使用します。このビットを 1'b1 に設定し、TMR_CTL.PRE を Source_Clock/1 に選択すると、プリスケーラのカウントである TMR_CTL.PRE が 3 から 0 に変更されます。 このビットの値は、TMR_CTL.PRE を Source_Clock/1 に選択した場合にプリスケーラ・カウント値に影響します。 このビットをセットすると、Source_Clock/1 が選択され、ゼロのプリスケール・カウント値がプリスケーラで使用されます。 このビットをクリアすると、Source_Clock/4 が選択され、3 のプリスケール・カウント値がプリスケーラで使用されます。	
		0	同期バイパスを無効にする
		1	同期バイパスを有効にする
14 (R/W)	RSTEN	カウンタおよびプリスケールのリセット・イネーブル。 リセット機能を有効または無効にするのに使用します。TMR_CTL.EVTEN および TMR_EVENTSELECT.EVTRANGE と一緒に使用します。選択したイベントが発生すると、16 ビット・カウンタと 8 ビット・プリスケールがリセットされます。これは PWM 復調モードで必要になります。	

表 22-8：TMR_CTL レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
13 (R/W)	EVTEN	イベント選択。 イベントのキャプチャを有効または無効にするのに使用します。TMR_EVENTSELECT.EVTRANGE と組み合わせて使用します。選択したイベントが発生すると、アップ／ダウン・カウンタの現在の値が TMR_CAPTURE にキャプチャされます。	
		0	イベントをキャプチャしない
		1	イベントをキャプチャする
7 (R/W)	RLD	再ロード制御。 このビットは周期モードでのみ使用します。アップ／ダウン・カウンタをタイムアウト・イベントでのみリセットするか、TMR_CLRINT.TIMEOUT をセットした場合にもリセットするかを選択できます。	
		0	アップ／ダウン・カウンタを、タイムアウト・イベントでのみリセットする。
		1	アップ／ダウン・カウンタを、「タイムアウト割込みクリア」ビットをセットした場合にもリセットする。
6:5 (R/W)	CLK	クロック選択。 使用可能な 4 つのクロック源からタイマー・クロックを選択するのに使用します。	
		0	CLK ソース 0 を選択（デフォルト）
		1	CLK ソース 1 を選択
		2	CLK ソース 2 を選択
		3	CLK ソース 3 を選択
4 (R/W)	EN	タイマー・イネーブル。 タイマーをイネーブルまたはディスエーブルするのに使用します。このビットをクリアすると、TMR_CURCNT レジスタを含め、タイマーがリセットされます。	
		0	タイマーをディスエーブル（デフォルト）
		1	タイマーをイネーブル
3 (R/W)	MODE	タイマー・モード。 このビットは、タイマーを周期的モードで動作させるか自走モードで動作させるかを制御する場合に使用します。周期モードでは、アップ／ダウン・カウンタは定義された TMR_LOAD.VALUE から開始します。自走モードでは、アップ／ダウン・カウンタは、タイマーがカウント・アップする場合は 0x0000 から、カウント・ダウンする場合は 0xFFFF から開始します。	
		0	タイマーを自走モードで実行
		1	タイマーを周期モードで実行（デフォルト）

表 22-8：TMR_CTL レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
2 (R/W)	UP	カウント・アップ。 アップ／ダウン・カウンタをインクリメント（カウント・アップ）するかデクリメント（カウント・ダウン）するかを制御します。	
		0	タイマーをカウント・ダウンするように設定（デフォルト）
		1	タイマーをカウント・アップするように設定
1:0 (R/W)	PRE	プリスケーラ。 タイマーの選択したクロックに適用されるプリスケーラの分周率を制御します。	
		0	source_clock / 1 または source_clock / 4。TMR_CTL.SYNCBYP をセットした場合は Source_Clock / 1、クリアした場合は Source_Clock / 4。
		1	Source_clock / 16
		2	Source_clock / 64
		3	Source_clock / 256

タイマー・イベント選択レジスタ

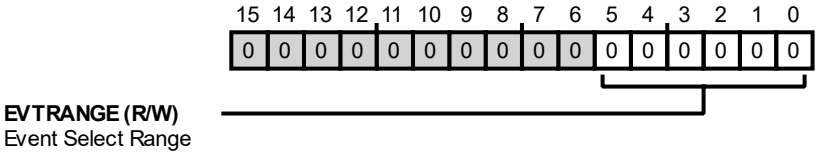


図 22-12：TMR_EVENTSELECT レジスタ図

表 22-9：TMR_EVENTSELECT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
5:0 (R/W)	EVTRANGE	イベント選択範囲。 タイマー・イベント選択範囲（0～39）。このレジスタ値により、モジュールに接続された 40 個のイベント入力の 1 つを選択します。イベントの選択範囲は 0～39 のみです。39 を超える値を書き込むと、レジスタの内容はゼロにクリアされます。

16 ビット・ロード値

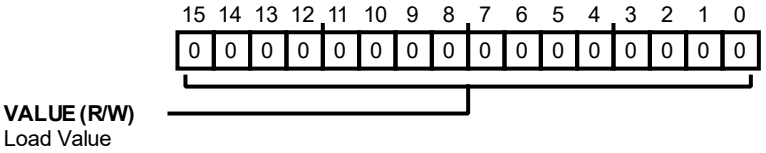


図 22-13：TMR_LOAD レジスタ図

表 22-10：TMR_LOAD レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	ロード値。 周期モードを選択している場合（TMR_CTL.MODE = 1）、この値がアップ／ダウン・カウンタに定期的にロードされます。アップ／ダウン・カウンタ・タイムアウト・イベント中の TMR_LOAD.VALUE への書込みは、イベントが経過するまで遅延します。

PWM 制御レジスタ

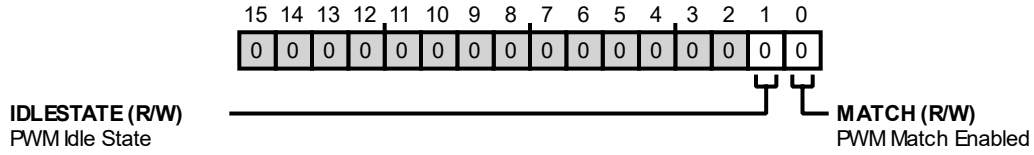


図 22-14：TMR_PWMCTL レジスタ図

表 22-11：TMR_PWMCTL レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
1 (R/W)	IDLESTATE	PWM アイドル状態。 このビットは、PWM アイドル状態を設定するのに使用します。	
		0	PWM のアイドル状態をローにする
		1	PWM のアイドル状態をハイにする
0 (R/W)	MATCH	PWM マッチ・イネーブル。 このビットは、PWM の動作モードを制御するのに使用します。	
		0	PWM をトグル・モードにする
		1	PWM をマッチ・モードにする

PWM マッチ値

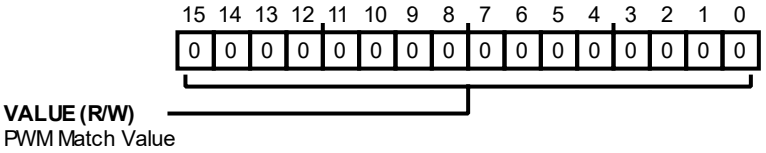


図 22-15：TMR_PWMMATCH レジスタ図

表 22-12：TMR_PWMMATCH レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	PWM マッチ値。 この値は、PWM がマッチ・モードで動作しているときに使用します。PWM 出力は、アップ／ダウン・カウンタがこのマッチ値と等しくなるとアサートされます。タイムアウト・イベントが発生すると、PWM 出力は再びアサート解除されます。マッチ値に達しなかった場合、またはタイムアウト・イベントと同時に発生した場合、PWM 出力はアイドル状態を維持します。

ステータス

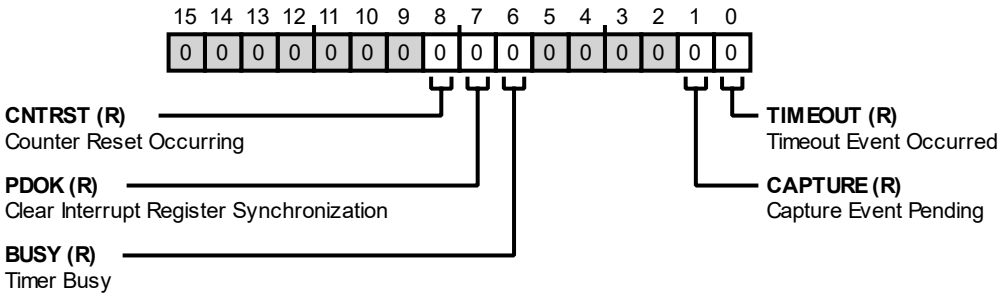


図 22-16：TMR_STAT レジスタ図

表 22-13：TMR_STAT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
8 (R/NW)	CNTRST	カウンタ・リセット発生。 イベントを検出したためカウンタが現在リセットされていることを示します。これを機能させるには、TMR_CTL.RSTEN をセットする必要があります。	
7 (R/NW)	PDOK	割込みクリア・レジスタ同期。 このビットは、TMR_CLRINT.TIMEOUT = 1 に設定すると自動的にセットされます。割込みクリア要求がクロック・ドメインを超え、タイマー・クロック・ドメインで有効になると、自動的にクリアされます。	
		0	タイマー・クロック・ドメインで割込みがクリアされた
		1	タイマー・クロック・ドメインでタイムアウト割込みクリア・ビットが更新された
6 (R/NW)	BUSY	タイマー・ビジー。 このビットは、TMR_CTL への書込みがタイマー・クロック・ドメインでまだ有効であることを知らせます。このビットは、TMR_CTL に書き込んだ後にチェックする必要があり、このビットがクリアされるまでは次の書込みを抑制する必要があります。	
		0	タイマーの、制御レジスタへのコマンドを受信する準備ができている
		1	タイマーの、制御レジスタへのコマンドを受信する準備はできていない
1 (R/NW)	CAPTURE	キャプチャ・イベント保留。 現在のタイマー値のキャプチャが発生したことを示します。	
		0	キャプチャ・イベントは保留されていない
		1	キャプチャ・イベントが保留されている

表 22-13：TMR_STAT レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
0 (R/NW)	TIMEOUT	タイムアウト・イベント発生。 このビットは、カウント・ダウン時にカウンタの値がゼロに達するか、またはカウント・アップ時にフルスケールに達すると自動的にセットされます。このビットは、TMR_CLRINT.TIMEOUT をセットするとクリアされます。	
		0	タイムアウト・イベントは発生していない
		1	タイムアウト・イベントが発生した

16 ビット・タイマー値

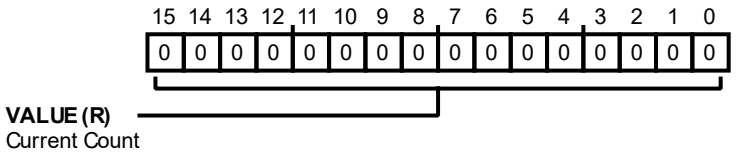


図 22-17：TMR_CURCNT レジスタ図

表 22-14：TMR_CURCNT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	VALUE	現在のカウント。 現在のアップ／ダウン・カウンタ値を反映します。この値は、クロック同期のために 2 PCLK サイクル遅延します。

23 RGB タイマー

ここでは、ADuCM4050 MCU の RGB タイマー機能について説明します。

RGB タイマー機能

RGB タイマーは以下の機能を備えています。

- 16 ビットのアップ／ダウン・カウンタおよび最大 8 ビットのクロック・プリスケーラ（1、16、64、256 のプリスケール・オプション）。
- 個別のクロック供給ブロック内で、タイマーに最大 4 つのクロック源を選択可能。
- 独立した制御レジスタおよびマッチ・レジスタを備えた 3 つの PWM 出力。
- 選択するクロックによって異なる最大タイムアウト時間。

  タイムアウト =（$2^{Counter\_width}$）/クロック周波数

  例）26MHz クロック時、タイムアウト =（2^{16+8}）/ 26MHz = 0.6s

  32kHz クロック時、タイムアウト =（2^{16+8}）/ 32kHz = 512s
- 自走モードと周期モードの 2 つの動作モードをサポート。
- リセット機能とキャプチャ機能により、タイマーで PWM 復調可能。
- プロセッサ・クロックと非同期のクロックをタイマーのソースとして使用可能。

RGB タイマー機能の説明

ここでは、ADuCM4050 MCU で使用される RGB タイマーの機能について説明します。

RGB タイマーのブロック図

次の図に RGB タイマーのブロック図を示します。

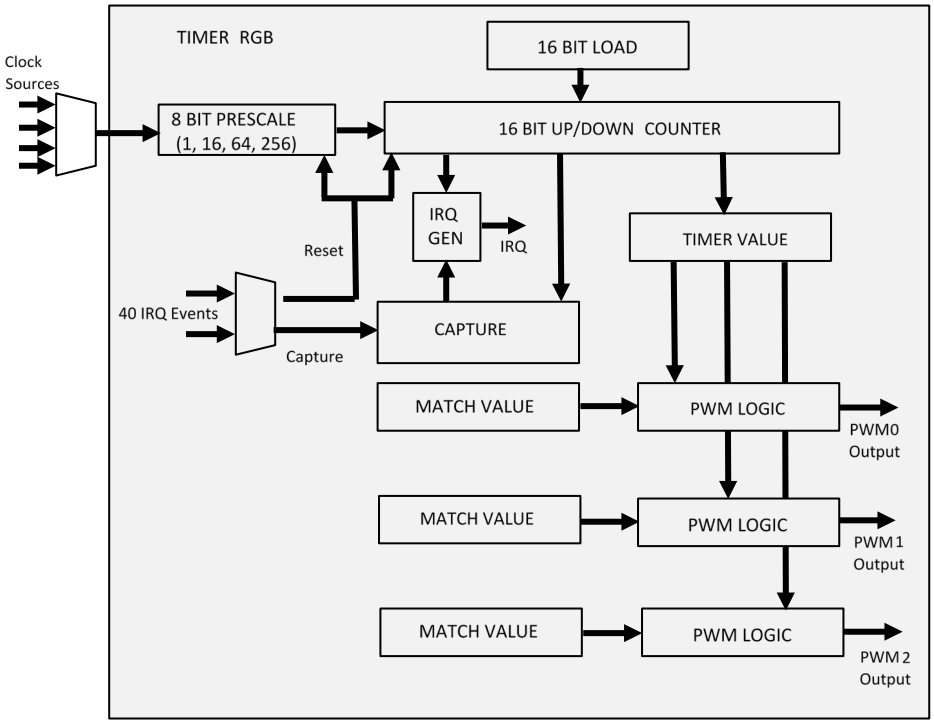


図 23-1：RGB タイマーのブロック図

RGB タイマーの動作

RGB タイマーには、自走モードと周期モードの 2 つの動作モードがあります。

自走モードでは、タイマーは TMR_RGB_CTL.EN ビットへの書込みによって開始します。カウント・アップ／ダウンすると、タイマーはゼロ／フルスケールからフルスケール／ゼロまでインクリメントします。フルスケールは、2^{16} − 1 すなわち 0xFFFF です。フルスケール（またはゼロスケール）に達すると、タイムアウト割込みが発生し、TMR_RGB_STAT.TIMEOUT ビットがセットされます。このビットをクリアするには、TMR_RGB_CLRINT.TIMEOUT ビットに 1 を書き込む必要があります。生成されるタイマー割込みは常にレベルであり、クリアするまではハイを維持します。タイムアウト割込みが発生すると、タイマーに最大値／最小値が再ロードされます。TMR_RGB_CTL.RLD ビットをセットすると、TMR_RGB_CLRINT.TIMEOUT ビットに 1 を書き込んだときにもタイマーへの再ロードが行われます。

周期モードでは、タイマーをイネーブルする前に初期値を TMR_RGB_LOAD.VALUE ビットにロードする必要があります。タイマーは、TMR_RGB_CTL.EN ビットへの書込みによって開始されます。カウンタの値は TMR_RGB_CTL.UP ビットに応じて、TMR_RGB_LOAD レジスタに格納されている値からフルスケールまでインクリメントするか、TMR_RGB_LOAD レジスタに格納されている値からゼロまでデクリメントします。カウンタがフルスケールまたはゼロに達すると、タイマーは割込みを生成します。TMR_RGB_LOAD.VALUE ビットがカウンタに再ロードされ、カウント・アップ／ダウンを継続します。タイマーは、TMR_RGB_CTL レジスタまたは TMR_RGB_LOAD レジスタを変更する前にディスエーブルする必要があります。デフォルトでは、IRQ の生成時にカウンタは自動的に再ロードされます。TMR_RGB_CTL.RLD ビットを 1 に設定すると、ユーザ・コードが TMR_RGB_CLRINT レジスタに書込みを行ったときにも、カウンタは再ロードされます。ロード値を細かいタイミングで変更できるようにするため、非同期書込み用として TMR_RGB_ALOAD レジスタが用意されています。

TMR_RGB_ALOAD レジスタに書き込むことにより、同期ロジックがバイパスされ、ロード値をきめ細かく制御できます。タイマーがイネーブルされ、コアとは異なるクロックで動作しているときに TMR_RGB_ALOAD レジスタを変更すると、未定義の結果が生じることがあります。

TMR_RGB_CTL.EN ビットのセットまたはクリアに続いて TMR_RGB_STAT を読み出してから、タイマー・レジスタに書き込む必要があります。TMR_RGB_STAT.PDOK ビットがゼロを返すと、レジスタを変更できます。これにより、タイマーは、コアとタイマーのクロック・ドメイン間でタイマー制御を同期させることができます。代表的な同期時間は 2 タイマー・クロック周期です。TMR_RGB_CTL.UP ビットまたは TMR_RGB_CL.MODE ビットの変更は、タイマーのディスエーブル時に実行する必要があります。

TMR_RGB_CURCNT レジスタにはコア・クロックに同期した有効な値が入っており、常時読出し可能です。
TMR_RGB_CTL レジスタで、カウンタのイネーブル、モードやプリスケール値の選択、イベント・キャプチャ機能の制御を行います。

PWM はトグル・モードまたはマッチ・モードで動作するように構成できます。いずれのモードでも、PWM 出力を使用するには、GPIO 制御モジュールで適切なマルチプレクサ設定を選択する必要があります。

トグル・モードでは、PWM は設定可能な周期で 50%のデューティ・サイクル出力を提供します。タイマーによってタイムアウト割込みが発生すると、PWM 出力が反転します。このため、周期は選択したクロックとプリスケーラによって定義されます。タイマーが周期モードで実行されている場合、PWM 出力は TMR_RGB_LOAD.VALUE 値にも依存します。タイムアウト・イベントは、タイマー期間（コア・クロック期間よりも長い）の終了時に評価されます。

マッチ・モードでの PWM 出力では、デューティ・サイクルと周期を設定可能です。トグル・モードと同様に、周期は選択したクロックとプリスケーラ、および TMR_RGB_LOAD.VALUE ビット（タイマーが周期モードで実行されている場合）によって定義されます。デューティ・サイクルは、各 PWM 出力の該当する PWM マッチ・レジスタ値によって定義されます。タイマーのカウンタ値がマッチ・レジスタに格納されている値と等しい場合、PWM 出力がアサートされます。PWM 出力はアサートを維持し、ゼロ／フルスケールに達するとアサート解除されます。タイマー期間（コア・クロック期間よりも著しく長い可能性がある）の終了時にマッチ・イベントおよびタイムアウト・イベントが評価されます。マッチ・イベントは、同じサイクルでトリガされた場合はタイムアウト・イベントよりも優先されます。したがって、周期モードでの動作時はマッチ・レジスタの値を TMR_RGB_LOAD.VALUE と同じ値に設定し、自走モード時にはゼロ／フルスケールに設定することにより、PWM を 100%デューティ・サイクルに設定できます。0%のデューティ・サイクルを設定できるのは周期モードでの実行時に限られ、マッチ・レジスタをタイマー・カウンタの範囲外（カウント・ダウン時は TMR_RGB_LOAD.VALUE + 1 に、カウント・アップ時は TMR_RGB_LOAD.VALUE − 1）に設定します。

ある与えられたクロック源、プリスケーラ、および TMR_RGB_LOAD.VALUE 値では、トグル・モードの PWM 周期はマッチ・モードの周期の 2 倍になります。トグル・モードでは、タイマー・カウンタ周期ごとに反転が 1 回しか行われません。マッチ・モードでは、PWM 出力のアサートとアサート解除が行われます。タイマー実行中の PWM 制御レジスタとマッチ・レジスタへの書込みがサポートされています。

注：次のマッチ・イベントまたはタイムアウト・イベントまで、制御レジスタへの変更は PWM 出力に反映されません。これにより、決められた PWM デューティ・サイクルを維持した上で PWM 動作を変更できます。TMR_RGB_CLRINT レジスタに書き込むか、タイマーをイネーブル／ディスエーブルすることで、PWM 出力を新たに書き込んだ設定と一致するように強制することができます。

PWM 変調

PWM 生成機能では、それぞれの PWM 制御レジスタの対応するビットに書き込むことにより、アイドル状態をロジック 0 またはロジック 1 に設定できます。オプションのマッチ値は、それぞれの PWM マッチ・レジスタに書込み可能で、PWM 制御レジスタの別のビットに書き込むことによって有効になります。PWM 出力は、トグル・モードまたはマッチ・モードのいずれかで生成されます。トグル・モードでは、PWM 出力がタイマーのゼロ／フルスケールでトグルするためデューティ・サイクルは 50%となり、周期が設定できます。マッチ・モードでは、PWM 出力のデューティ・サイクルと周期を設定できます。マッチ・モードでは、PWM 出力は PWM 制御レジスタで設定されたアイドル状態から開始し、タイマーとマッチ値が等しくなるとアサートされ、タイマーがゼロ／フルスケールに達するとアサート解除されて、再びアイドル状態になります。

PWM 出力 PWM0、PWM1、PWM2 に対応する PWM 制御レジスタは、それぞれ `TMR_RGB_PWM0CTL`、`TMR_RGB_PWM1CTL`、`TMR_RGB_PWM2CTL` です。

PWM 出力 PWM0、PWM1、PWM2 に対応する PWM マッチ・レジスタは、それぞれ `TMR_RGB_PWM0MATCH`、`TMR_RGB_PWM1MATCH`、`TMR_RGB_PWM2MATCH` です。

開始点が不確定にならないように、タイマーをイネーブルする前に PWM レジスタに適切な値を設定します。

このシーケンスは以下のとおりです。

1. PWM レジスタを設定する。
2. タイマーをイネーブルする。

次の図は、3 つの PWM 出力に異なるデューティ・サイクルを設定し、すべての PWM 出力の立上がりエッジが同時に発生する波形を示しています。

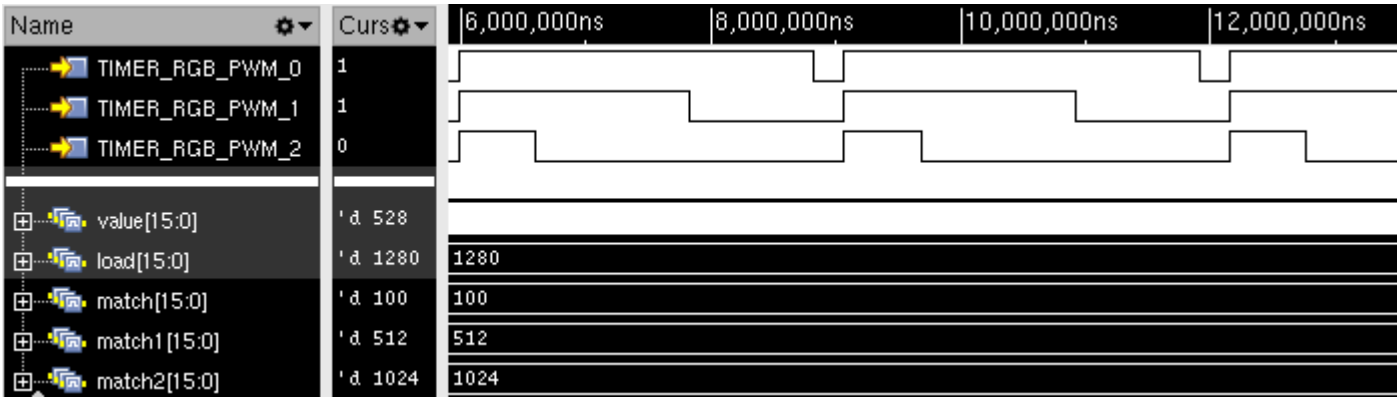


図 23-2：PWM 出力波形

PWM 復調

PWM 復調は、イベント・ロジックでタイマー・リセット機能とキャプチャ機能を両方とも有効にすることで実行できます。これにより、PWM 入力のハイ・レベルとロー・レベルのカウント値を別々に読み出すことができます。実際には、MCU は適切な PWM（GPIO）割込み源を選択し、PWM パルスの立上がりエッジで割込みをトリガします。キャプチャを検出すると、カウンタがリセットされ、MCU への割込みを実行します。MCU は、PWM パルスの立下がりエ

ッジでトリガするように（エッジ発生前に）割込みを変更します。この割込みにより、ハイ・レベル・カウントが TMR_RGB_CAPTURE レジスタにキャプチャされてカウンタがリセットされ、更新されたキャプチャ値で MCU に再度割込みが実行されます。MCU はこのキャプチャされたカウントを読み出し、再び正のエッジで動作するように（エッジ発生前に）割込みを修正します。この割込みにより、ロー・レベル・カウントが TMR_RGB_CAPTURE レジスタにキャプチャされてカウンタがリセットされ、MCU に再度割込みが実行されます。MCU はこのキャプチャされたカウントを読み出すことができます。両方の読出し値を比較することにより、PWM 値が示されます。PWM パルス幅は、様々な入力エッジ間で必要とされる同期遅延、IRQ 遅延、およびソフトウェア設定のオーバーヘッドに対応するだけの十分に広い幅である必要があります。

次の図は PWM 復調波形を示しています。

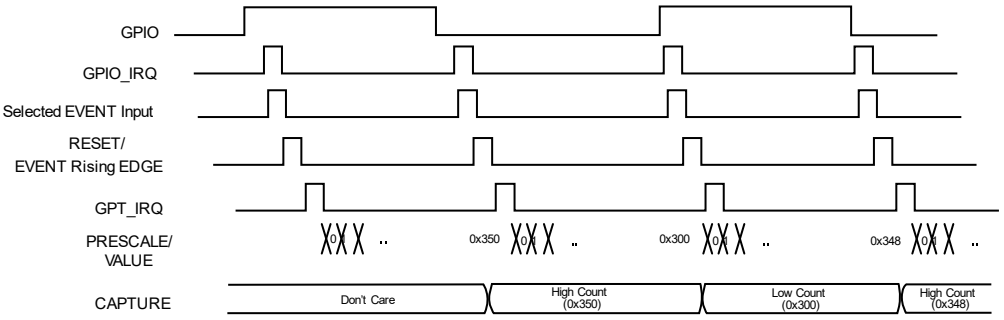


図 23-3：PWM 復調波形

クロック選択

各タイマーにつき 4 通りのクロックが使用できます。選択したクロックはプリスケーラと共に、タイマー・カウンタ・ロジックの周波数を制御するのに使用されます。使用するクロックは、TMR_RGB_CTL.CLK ビットを設定して選択します。使用するのは同期クロックのみです。32kHz クロックを選択した場合、外部からこのブロックへのシステム・クロックに最初に同期されます。

イベントのキャプチャ

40 個の割込みイベントをこのタイマーでキャプチャできます。タイマーに関連付けられた 40 イベントのいずれか 1 つで、16 ビットの TMR_RGB_CURCNT レジスタを 16 ビットの TMR_RGB_CAPTURE レジスタにキャプチャさせることができます。TMR_RGB_EVENTSELECT レジスタには、40 個のイベントの 1 つを選択する 6 ビットのフィールドがあります。いずれかのイベントが出力イネーブルをトリガすると、event_out がアサートされます。割込みイベントは、Cortex NVIC 割込みバスの 32 個のソースに接続されており、これらはすべてメインのシステム・クロックに同期しています。

選択した IRQ が発生すると、TMR_RGB_CURCNT レジスタが TMR_RGB_CAPTURE レジスタにコピーされます。TMR_RGB_STAT.CAPTURE ビットがセットされます。TMR_RGB_CLRINT.EVTCAPT ビットに 1 を書き込むことで、IRQ はクリアされます。TMR_RGB_CAPTURE レジスタもその値を保持し、TMR_RGB_CLRINT.EVTCAPT に 1 が書き込まれるまでは上書きできません。

表 23-1：イベント一覧

Event Select Bits (EVTSEL)	Timer_RGB Capture Source
0	External Interrupt 2

表 23-1：イベント一覧（続き）

Event Select Bits (EVTSEL)	Timer_RGB Capture Source
1	UART1
2	SPORT0A
3	I2C0 Master
4	Reserved
5	DMA Channel 1 Done
6	DMA Channel 5 Done
7	DMA Channel 9 Done
8	DMA Channel 13 Done
9	DMA Channel 17 Done
10	DMA Channel 21 Done
11	DMA Channel 25 Done
12	XTAL OSC
13	RNG
14	Reserved
15	Reserved
16	Timer2
17	RTC1
18	External Interrupt 0
19	External Interrupt 1
20	External Interrupt 3
21	RTC0
22	UART0
23	SPI0
24	SPI1
25	SPI2
26	SPORT0B
27	GPIO-IntA
28	GPIO-IntB
29	I2C0 Slave
30	P0_8 Input
31	Reserved

表 23-1：イベント一覧（続き）

Event Select Bits (EVTSEL)	Timer_RGB Capture Source
32	Reserved
33	DMA Error
34	DMA Channel 0 Done
35	DMA Channel 2 Done
36	DMA Channel 3 Done
37	DMA Channel 4 Done
38	DMA Channel 6 Done
39	DMA Channel 7 Done

ADuCM4050 TMR_RGB レジスタの説明

3 PWM 出力を備えた RGB タイマー（TMR_RGB）には、以下のレジスタがあります。

表 23-2：ADuCM4050 TMR_RGB レジスター一覧

レジスタ名	説明
TMR_RGB_ALOAD	16 ビット・ロード値、非同期
TMR_RGB_ACURCNT	16 ビット・タイマー値、非同期
TMR_RGB_CAPTURE	キャプチャ
TMR_RGB_CLRINT	割込みクリア
TMR_RGB_CTL	制御
TMR_RGB_EVENTSELECT	タイマー・イベント選択レジスタ
TMR_RGB_LOAD	16 ビット・ロード値
TMR_RGB_PWM0CTL	PWM0 制御レジスタ
TMR_RGB_PWM0MATCH	PWM0 マッチ値
TMR_RGB_PWM1CTL	PWM1 制御レジスタ
TMR_RGB_PWM1MATCH	PWM1 マッチ値
TMR_RGB_PWM2CTL	PWM2 制御レジスタ
TMR_RGB_PWM2MATCH	PWM2 マッチ値
TMR_RGB_STAT	ステータス
TMR_RGB_CURCNT	16 ビット・タイマー値

16 ビット・ロード値、非同期

同期クロック源を選択している場合にのみ使用します（TMR_RGB_CTL.CLK = 00）。

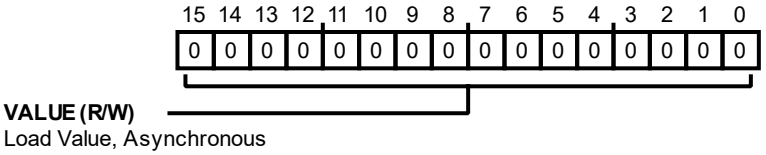


図 23-4：TMR_RGB_ALOAD レジスタ図

表 23-3：TMR_RGB_ALOAD レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	ロード値、非同期。 周期モードを選択している場合（TMR_RGB_CTL.MODE = 1）、この値がアップ／ダウン・カウンタに定期的にロードされます。TMR_RGB_ALOAD.VALUE への書込みは、これ以外の場合では必要とされるクロック同期ロジックをバイパスすることで、タイマーが PCLK で動作することを利用します。

16 ビット・タイマー値、非同期

同期クロック源を選択している場合にのみ使用します（TMR_RGB_CTL.CLK = 00）。

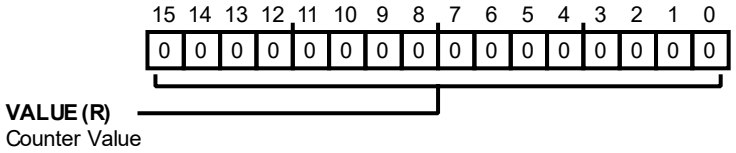


図 23-5：TMR_RGB_ACURCNT レジスタ図

表 23-4：TMR_RGB_ACURCNT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	VALUE	カウンタ値。 現在のアップ／ダウン・カウンタ値を反映します。TMR_RGB_ACURCNT の読出しは、これ以外の場合では必要とされるクロック同期ロジックをバイパスすることで、タイマーが PCLK で動作することを利用します。

キャプチャ

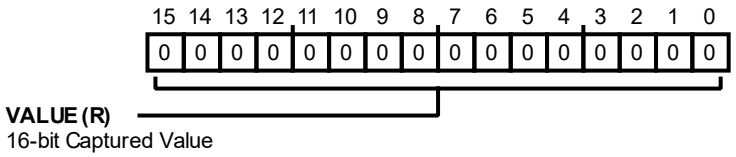


図 23-6：TMR_RGB_CAPTURE レジスタ図

表 23-5：TMR_RGB_CAPTURE レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	VALUE	16 ビットのキャプチャ値。 TMR_RGB_CAPTURE は、ユーザ・コードによって TMR_RGB_CLRINT.EVTCAPT がセットされるまでその値を保持します。TMR_RGB_CLRINT.EVTCAPT に書き込まない限り、別のイベントが発生しても TMR_RGB_CAPTURE が上書きされることはありません。

割込みクリア

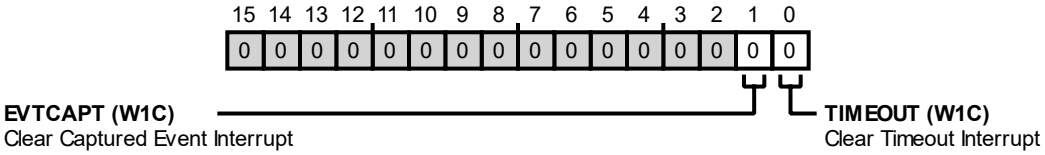


図 23-7：TMR_RGB_CLRINT レジスタ図

表 23-6：TMR_RGB_CLRINT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
1 (RX/W1C)	EVTCAPT	キャプチャ・イベント割込みクリア。 このビットは、キャプチャ・イベント割込みをクリアするのに使用します。	
		0	影響なし
		1	キャプチャ・イベント割込みをクリア
0 (RX/W1C)	TIMEOUT	タイムアウト割込みクリア。 このビットは、タイムアウト割込みをクリアするのに使用します。	
		0	影響なし
		1	タイムアウト割込みをクリア

制御

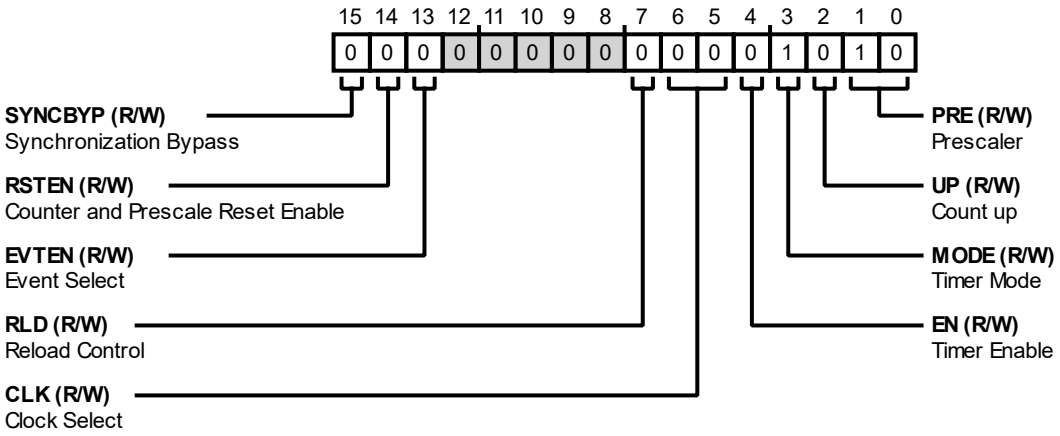


図 23-8：TMR_RGB_CTL レジスタ図

表 23-7：TMR_RGB_CTL レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
15 (R/W)	SYNCBYP	同期バイパス。 ブロック内の同期ロジックをバイパスするのに使用します。同期クロックの場合のみ使用します。また、このビット・フィールドにより、TMR_RGB_CTL.PRE の最大プリスケーラ・カウントが 3 から 0 に変更されます。	
14 (R/W)	RSTEN	カウンタおよびプリスケールのリセット・イネーブル。 リセット機能を有効または無効にするのに使用します。TMR_RGB_CTL.EVTEN および TMR_RGB_EVENTSELECT.EVTRANGE と共に使用します。選択したイベントが発生すると、16 ビット・カウンタと 8 ビット・プリスケールがリセットされます。これは PWM 復調モードで必要になります。	
13 (R/W)	EVTEN	イベント選択。 イベントのキャプチャを有効または無効にするのに使用します。TMR_RGB_EVENTSELECT.EVTRANGE と共に使用します。選択したイベントが発生すると、アップ／ダウン・カウンタの現在の値が TMR_RGB_CAPTURE にキャプチャされます。	
		0	イベントをキャプチャしません
		1	イベントをキャプチャします

表 23-7：TMR_RGB_CTL レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
7 (R/W)	RLD	再ロード制御。 TMR_RGB_CTL.RL D は周期モードでのみ使用します。このビットにより、アップ／ダウン・カウンタをタイムアウト・イベントでのみリセットするか、TMR_RGB_CLRINT.TIMEOUT をセットした場合にもリセットするかを選択できます。	
		0	アップ／ダウン・カウンタを、タイムアウト・イベントでのみリセットします
		1	アップ／ダウン・カウンタを、Clear Timeout Interrupt ビットをセットした場合にもリセットします
6:5 (R/W)	CLK	クロック選択。 使用可能な 4 つのクロック源からタイマー・クロックを選択するのに使用します。使用可能なクロック源の詳細については、クロック源の表を参照してください。	
		0	CLK 源 0 を選択（デフォルト）
		1	CLK 源 1 を選択
		2	CLK 源 2 を選択
		3	CLK 源 3 を選択
4 (R/W)	EN	タイマー・イネーブル。 タイマーをイネーブルまたはディスエーブルするのに使用します。このビットをクリアすると、TMR_RGB_CURCNT レジスタを含め、タイマーがリセットされます。	
		0	タイマーをディスエーブル（デフォルト）
		1	タイマーをイネーブル
3 (R/W)	MODE	タイマー・モード。 このビットは、タイマーを周期的モードで動作させるか自走モードで動作させるかを制御する場合に使用します。周期モードでは、アップ／ダウン・カウンタは定義された TMR_RGB_LOAD.VALUE から開始します。自走モードでは、アップ／ダウン・カウンタは、タイマーがカウント・アップする場合は 0x0000 から、カウント・ダウンする場合は 0xFFFF から開始します。	
		0	タイマーを自走モードで実行
		1	タイマーを周期モードで実行（デフォルト）
2 (R/W)	UP	カウント・アップ。 アップ／ダウン・カウンタをインクリメント（カウント・アップ）するかデクリメント（カウント・ダウン）するかを制御します。	
		0	タイマーをカウント・ダウンするように設定（デフォルト）
		1	タイマーをカウント・アップするように設定

表 23-7：TMR_RGB_CTL レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
1:0 (R/W)	PRE	プリスケーラ。 タイマーの選択したクロックに適用されるプリスケーラの分周率を制御します。	
		0	source_clock / 1 または source_clock / 4。TMR_RGB_CTL.SYNCBYPP をセットした場合は source_clock / 1、クリアした場合は source_clock / 4。
		1	source_clock / 16
		2	source_clock / 64
		3	source_clock / 256

タイマー・イベント選択レジスタ

このレジスタ値により、モジュールに接続された 40 個のイベント入力の 1 つを選択します。イベントの選択範囲は 0～39 のみです。39 を超える値を書き込むと、レジスタの内容はゼロにクリアされます。

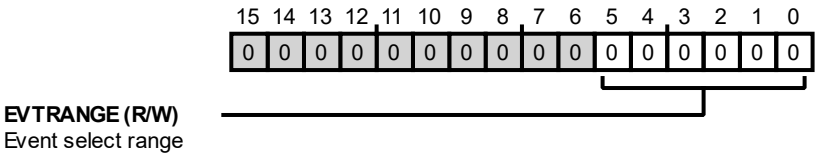


図 23-9：TMR_RGB_EVENTSELECT レジスタ図

表 23-8：TMR_RGB_EVENTSELECT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
5:0 (R/W)	EVTRANGE	イベント選択範囲。 タイマー・イベント選択範囲（0～39）。このジスタ値により、モジュールに接続された 40 個のイベント入力の 1 つを選択します。イベントの選択範囲は 0～39 のみです。39 を超える値を書き込むと、レジスタの内容はゼロにクリアされます。

16 ビット・ロード値

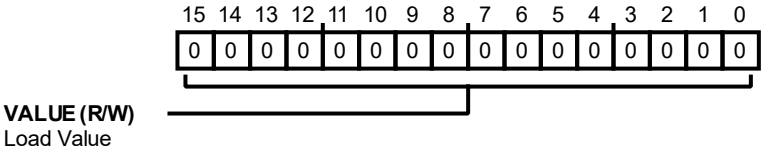


図 23-10：TMR_RGB_LOAD レジスタ図

表 23-9：TMR_RGB_LOAD レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	ロード値。 周期モードを選択している場合（TMR_RGB_CTL.MODE = 1）、この値がアップ／ダウン・カウンタに定期的にロードされます。アップ／ダウン・カウンタ・タイムアウト・イベント中の TMR_RGB_LOAD.VALUE への書込みは、イベントが経過するまで遅延します。

PWM0 制御レジスタ

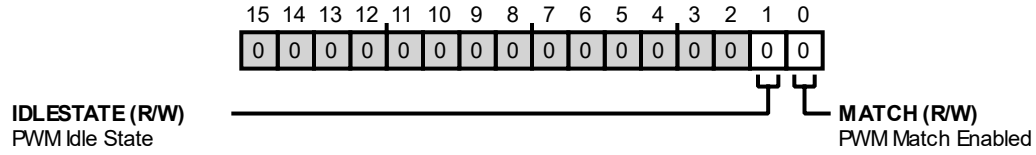


図 23-11：TMR_RGB_PWM0CTL レジスタ図

表 23-10：TMR_RGB_PWM0CTL レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
1 (R/W)	IDLESTATE	PWM アイドル状態。 このビットは、PWM アイドル状態を設定するのに使用します。0：PWM のアイドル状態をローにします、1：PWM のアイドル状態をハイにします
0 (R/W)	MATCH	PWM マッチ・イネーブル。 このビットは、PWM の動作モードを制御するのに使用します。0：PWM をトグル・モードにします、1：PWM をマッチ・モードにします

PWM0 マッチ値

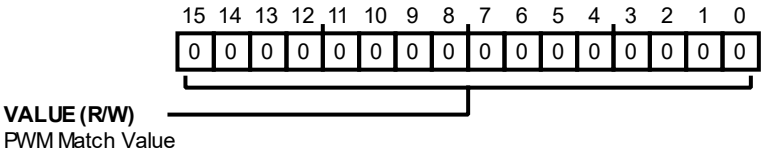


図 23-12：TMR_RGB_PWM0MATCH レジスタ図

表 23-11：TMR_RGB_PWM0MATCH レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	PWM マッチ値。 この値は、PWM がマッチ・モードで動作しているときに使用します。PWM 出力は、アップ／ダウン・カウンタがこのマッチ値と等しくなるとアサートされます。タイムアウト・イベントが発生すると、PWM 出力は再びアサート解除されます。マッチ値に達しなかった場合、またはタイムアウト・イベントと同時に発生した場合、PWM 出力はアイドル状態を維持します。

PWM1 制御レジスタ

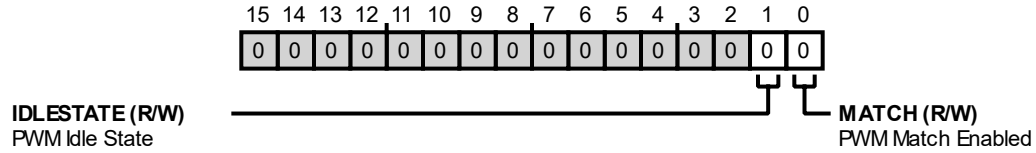


図 23-13：TMR_RGB_PWM1CTL レジスタ図

表 23-12：TMR_RGB_PWM1CTL レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
1 (R/W)	IDLESTATE	PWM アイドル状態。 このビットは、PWM アイドル状態を設定するのに使用します。0：PWM のアイドル状態をローにします、1：PWM のアイドル状態をハイにします
0 (R/W)	MATCH	PWM マッチ・イネーブル。 このビットは、PWM の動作モードを制御するのに使用します。0：PWM をトグル・モードにします、1：PWM をマッチ・モードにします

PWM1 マッチ値

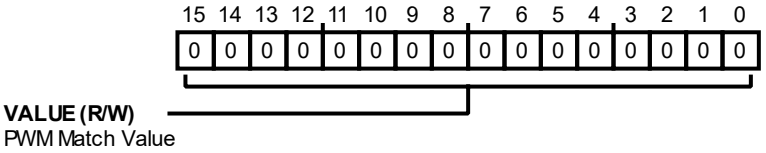


図 23-14：TMR_RGB_PWM1MATCH レジスタ図

表 23-13：TMR_RGB_PWM1MATCH レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	PWM マッチ値。 この値は、PWM がマッチ・モードで動作しているときに使用します。PWM 出力は、アップ／ダウン・カウンタがこのマッチ値と等しくなるとアサートされます。タイムアウト・イベントが発生すると、PWM 出力は再びアサート解除されます。マッチ値に達しなかった場合、またはタイムアウト・イベントと同時に発生した場合、PWM 出力はアイドル状態を維持します。

PWM2 制御レジスタ

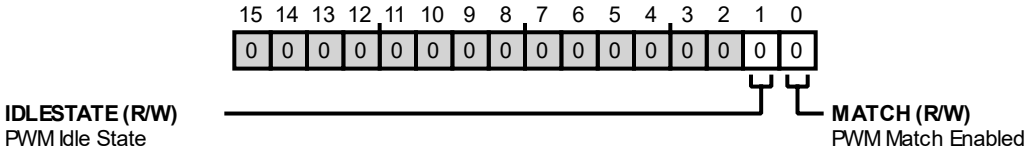


図 23-15：TMR_RGB_PWM2CTL レジスタ図

表 23-14：TMR_RGB_PWM2CTL レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
1 (R/W)	IDLESTATE	PWM アイドル状態。 このビットは、PWM アイドル状態を設定するのに使用します。0：PWM のアイドル状態をローにします、1：PWM のアイドル状態をハイにします
0 (R/W)	MATCH	PWM マッチ・イネーブル。 このビットは、PWM の動作モードを制御するのに使用します。0：PWM をトグル・モードにします、1：PWM をマッチ・モードにします

PWM2 マッチ値

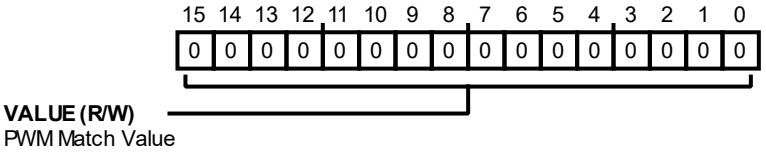


図 23-16：TMR_RGB_PWM2MATCH レジスタ図

表 23-15：TMR_RGB_PWM2MATCH レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	PWM マッチ値。 この値は、PWM がマッチ・モードで動作しているときに使用します。PWM 出力は、アップ／ダウン・カウンタがこのマッチ値と等しくなるとアサートされます。タイムアウト・イベントが発生すると、PWM 出力は再びアサート解除されます。マッチ値に達しなかった場合、またはタイムアウト・イベントと同時に発生した場合、PWM 出力はアイドル状態を維持します。

ステータス

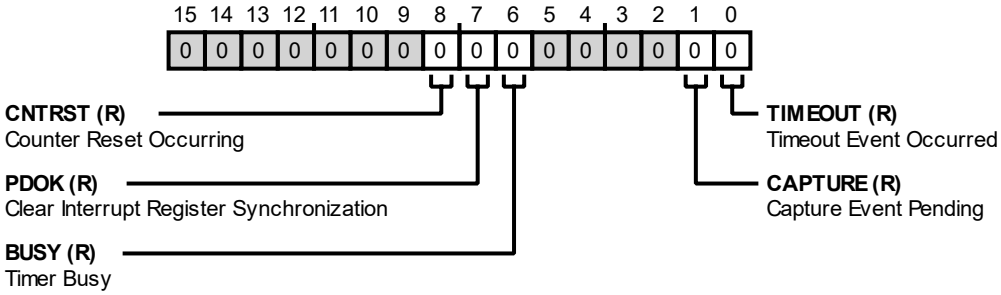


図 23-17：TMR_RGB_STAT レジスタ図

表 23-16：TMR_RGB_STAT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
8 (R/NW)	CNTRST	カウンタ・リセット発生。 イベントを検出したためカウンタが現在リセットされていることを示します。これを機能させるには、TMR_RGB_CTL.RSTEN をセットする必要があります。	
7 (R/NW)	PDOK	割込みクリア・レジスタ同期。 このビットは、TMR_RGB_CLRINT.TIMEOUT = 1 に設定すると自動的にセットされます。割込みクリア要求がクロック・ドメインを超え、タイマー・クロック・ドメインで有効になると、自動的にクリアされます。	
		0	タイマー・クロック・ドメインで割込みがクリアされました
		1	タイマー・クロック・ドメインでタイムアウト割込みクリア・ビットが更新されました
6 (R/NW)	BUSY	タイマー・ビジー。 このビットは、TMR_RGB_CTL への書込みがタイマー・クロック・ドメインでまだ有効であることを知らせます。このビットは、TMR_RGB_CTL に書き込んだ後にチェックする必要があり、このビットがクリアされるまでは次の書込みを抑制する必要があります。	
		0	制御レジスタへのコマンドを受信する準備がタイマーにできています
		1	制御レジスタへのコマンドを受信する準備がタイマーにできていません
1 (R/NW)	CAPTURE	キャプチャ・イベント保留。 現在のタイマー値のキャプチャが発生したことを示します。	
		0	キャプチャ・イベントは保留されていません
		1	キャプチャ・イベントが保留されています

表 23-16：TMR_RGB_STAT レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
0 (R/NW)	TIMEOUT	タイムアウト・イベント発生。 このビットは、カウント・ダウン時にカウンタの値がゼロに達するか、またはカウント・アップ時にフルスケールに達すると、自動的にセットされます。このビットは、TMR_RGB_CLRINT.TIMEOUT をセットするとクリアされます。	
		0	タイムアウト・イベントは発生していません
		1	タイムアウト・イベントが発生しました

16 ビット・タイマー値

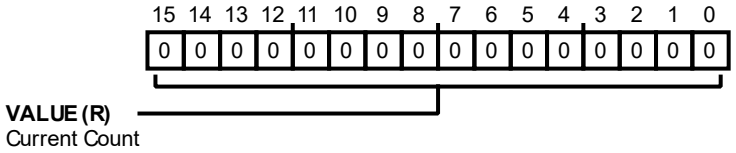


図 23-18：TMR_RGB_CURCNT レジスタ図

表 23-17：TMR_RGB_CURCNT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	VALUE	現在のカウント。 現在のアップ／ダウン・カウンタ値を反映します。この値は、クロック同期化のために 2 PCLK サイクル遅延します。

24 ウォッチドッグ・タイマー（WDT）

ウォッチドッグ・タイマーは無効なソフトウェア状態からの復帰に使用されます。ユーザ・コードでイネーブルにした場合、リセットや MCU への割込みが強制的に行われないように定期的な処置が必要です。

WDT 機能

ウォッチドッグ・タイマーは 32kHz のオンチップ発振器（LFOSC）でクロックされます。ウォッチドッグは、リセット、休止、シャットダウン、デバッグ・モード以外のときは、常にクロックされます。このタイマーは、プログラマブル・プリスケーラを備えた 16 ビットのカウント・ダウン・タイマーです。プリスケーラ源は選択可能で、1、16、または 256 の係数でスケール化できます。WDT のタイムアウトにより、リセットまたは割込みを生成できます。`WDT_CTL.IRQ` ビットで、リセットの代わりに割込みを選択できますが、これはデバッグ機能を目的としたものです。`WDT_RESTART` 書込み専用レジスタに 0xCCCC を書き込むことにより、割込みをクリアできます。

WDT 機能の説明

ここでは、ADuCM4050 MCU で使用される WDT の機能について説明します。

WDT ブロック図

この図は、ADuCM4050 MCU で使用される WDT のブロック図を示しています。

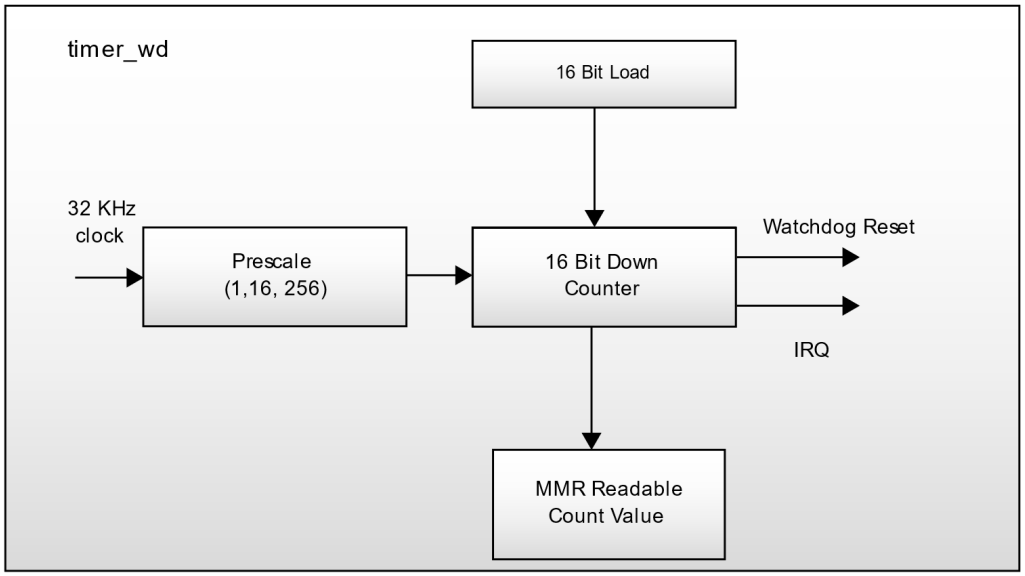


図 24-1：ウォッチドッグ・タイマーのブロック図

WDT 動作モード

有効なリセット後、次のようにハードウェアでウォッチドッグ・タイマーをリセットします。

1. WDT_CTLWDT_CTL = 0x00E9 を設定
2. WDT_LOAD = 0x1000 を設定
3. WDT_CCNT = 0x1000 を設定

この設定により、タイムアウト値が 32 秒のウォッチドッグ・タイマーがイネーブルされます。ただし、ADI カーネルによってウォッチドッグ・タイマーをディスエーブルすれば、ユーザーによるタイマー設定が可能です。したがって、レジスタにアクセスしていたとしたら、次の値が読み取れるはずです。

1. WDT_CTL = 0x0
2. WDT_LOAD = 0x1000
3. WDT_CCNT = 0x1000

カーネル内のタイマーをディスエーブルすることで、ウォッチドッグ・タイマーの書込み保護機能の利点を活用できます。ユーザ・コードによって WDT_CTL.EN ビットをセットすると、WDT_STAT.LOCKED ビットもセットされ、WDT_CTL および WDT_LOAD.VALUE（ロード値）が書込み保護されます。リセットでのみ、書込み保護と WDT_STAT.LOCKED ビットをクリアして、タイマーを再設定することができます。

WDT_CTL レジスタを変更していない場合、ユーザ・コードでいつでも WDT_LOAD.VALUE を変更できます。WDT_CTL.EN ビットをクリアすると（WDT_CTL レジスタの書込み前に）、タイマーはディスエーブルされます。これにより、タイマーの設定が変更され、タイマーを再度イネーブルできます。WDT_CTL レジスタへの書込みによってタイマーが再びイネーブルされると同時にウォッチドッグの構成がロックされ、レジスタ値が誤って変更されることがなくなります。

ウォッチドッグ・タイマーを自走モード（WDT_CTL.MODE = 0）に設定している場合、ウォッチドッグ・タイマーの値は 0x1000 からゼロにデクリメントし、0x1000 にラップ・アラウンドしてデクリメントを継続します。

タイムアウト値を 0x1000（デフォルトのプリスケール = 2 で約 32 秒）より大きくするか小さくするには、周期モードを使用する必要があります（WDT_CTL.MODE = 1）。WDT_LOAD.VALUE および WDT_CTL.PRE には、希望するタイムアウト時間に応じた値を書き込みます。最大のタイムアウト時間は約 8 分です（WDT_LOAD.VALUE = 0xFFFF、WDT_CTL.PRE = 2）。周期モードを選択すると、タイマー値は WDT_LOAD.VALUE 値からゼロまでデクリメントし、再び WDT_LOAD.VALUE にラップ・アラウンドしてデクリメントを継続します。

WDT_CCNT.VALUE には、APB クロックに同期し、いつでも読み出すことのできる有効な値が含まれます。

ウォッチドッグ・タイマーが 0 になると、リセットまたは割込みが発生します。このリセットは、有効期間の満了前に WDT_RESTART レジスタに値 0xCCCC を書き込むことによって抑止できます。WDT_RESTART に書き込むと、ウォッチドッグ・タイマーに WDT_LOAD.VALUE 値（または自走モードの場合は 0x1000）が再ロードされ、直ちに新しいタイムアウト時間で再びカウントが開始されます。0xCCCC 以外の値を書き込むと、リセットまたは割込みが発生します。

WDT_CTL.EN ビットをクリアしてタイマーをディスエーブルすると、内部プリスケーラとカウンタの両方がクリアされます。再度イネーブルすると、カウンタに WDT_CTL.MODE ビットの設定に対応する値が再ロードされます。

WDT の設定が再初期化／リセットされるのは、POR またはシステム・リセット後に限られます。

誤った WDT_RESTART への書込みによってウォッチドッグ・リセットがアサートされた場合、アサート期間は 1PCLK 期間になります。タイムアウトによるリセットは 32kHz のフル・クロック周期になります。

ウォッチドッグ同期

ウォッチドッグ・タイマーには、WDT_STAT.COUNTING、DT_STAT.LOADING、および WDT_STAT.CLRIRQ の 3 つのステータス・ビットがあり、WDT_CTL、WDT_LOAD.VALUE、および WDT_RESTART レジスタのそれぞれについて、高速クロック領域と低速クロック領域の同期が処理中であることを示します。

WDT_CTL または WDT_LOAD.VALUE レジスタには、対応する同期ビットがセットされているときは書き込まないでください。ただし、WDT_RESTART レジスタへの値 0xCCCC の連続書込みは許可されます。以前のアクセスの同期中に行われた WDT_RESTART への値 0xCCCC の書込みは、無視されます。

ウォッチドッグの電源モードでの動作

ウォッチドッグ・タイマーは、休止モードとシャットダウン・モードでは自動的にディスエーブルされます。ウォッチドッグ・タイマー・レジスタは、休止モードでは保持されません。休止モードおよびシャットダウン・モードの終了後に、WDT カウンタを再設定する必要があります。それ以外の場合、WDT のタイムアウト値には約 32 秒のデフォルト値が使用されます。

ADuCM4050 WDT レジスタの説明

ウォッチドッグ・タイマー（WDT）には以下のレジスタがあります。

表 24-1：ADuCM4050 WDT レジスター覧

レジスタ名	説明
WDT_CCNT	現在のカウント値
WDT_CTL	制御
WDT_LOAD	ロード値
WDT_RESTART	割込みクリア
WDT_STAT	ステータス

現在のカウント値

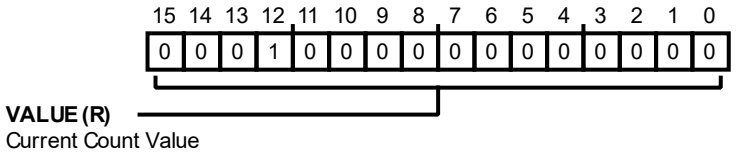


図 24-2：WDT_CCNT レジスタ図

表 24-2：WDT_CCNT レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/NW)	VALUE	現在のカウント値。 このレジスタは APB クロックに同期されます。

制御

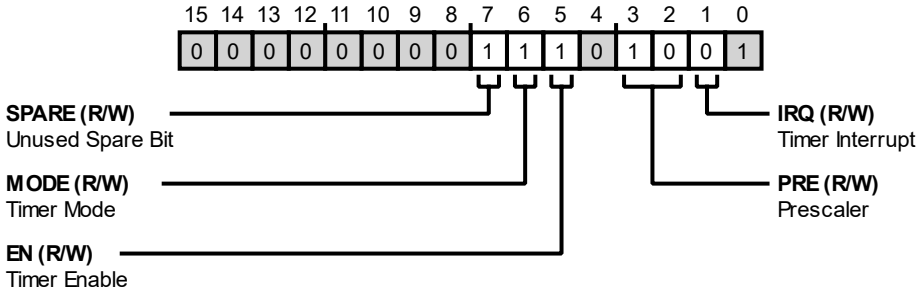


図 24-3：WDT_CTL レジスタ図

表 24-3：WDT_CTL レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
7 (R/W)	SPARE	未使用の予備ビット。 注：ADI カーネルは、このビット・フィールドに'0x0'を書き込みます。	
6 (R/W)	MODE	タイマー・モード。 セットすると、周期モードで動作します（デフォルト）。クリアすると、自走モードで動作します。自走モードでは、0x1000 でラップ・アラウンドします。 注：ADI カーネルは、このビット・フィールドに'0x0'を書き込みます（デフォルト）。	
		0	自走モード
		1	周期モード
5 (R/W)	EN	タイマー・イネーブル。 セットすると、タイマーをイネーブルします（デフォルト）。クリアすると、タイマーをディスエーブルします。セットしてしまうと、システム・リセット以外ではウォッチドッグをディスエーブルできません。これにより、ソフトウェアが誤ってウォッチドッグをディスエーブルすることが防止されます。 注：ADI カーネルは、このビット・フィールドに'0x0'を書き込みます（デフォルト）。	
		0	WDT をディスエーブル
		1	WDT をイネーブル
3:2 (R/W)	PRE	プリスケーラ。 注：ADI カーネルは、これらのビット・フィールドに'0x0'を書き込みます（デフォルト）。	
		0	ソース・クロック/1
		1	ソース・クロック/16
		2	ソース・クロック/256（デフォルト）
		3	予備

表 24-3：WDT_CTL レジスタ・フィールド（続き）

ビット番号 （アクセス）	ビット名	ビット値／機能対応	
1 (R/W)	IRQ	タイマー割込み。 セットすると、タイマーがタイムアウトしたときに割込みを生成します。この機能はデバッグの目的で提供されており、PCLK がアクティブな場合にのみ使用できます。 クリアすると、タイマーがタイムアウトしたときにリセットが生成されます（デフォルト）。	
		0	タイムアウトしたときに WDT がリセットをアサート
		1	タイムアウトしたときに WDT が割込みを生成

ロード値

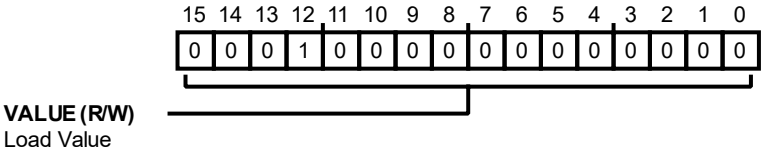


図 24-4：WDT_LOAD レジスタ図

表 24-4：WDT_LOAD レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (R/W)	VALUE	ロード値。

割込みクリア

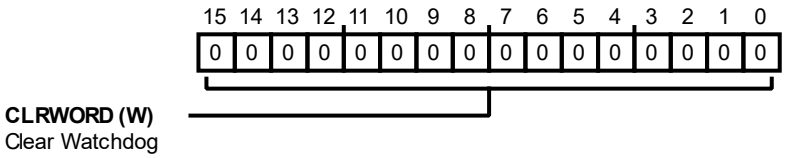


図 24-5：WDT_RESTART レジスタ図

表 24-5：WDT_RESTART レジスタ・フィールド

ビット番号 （アクセス）	ビット名	ビット値／機能対応
15:0 (RX/W)	CLRWORD	ウォッチドッグ・クリア。 0xCCCC を書き込むと、リセット、再ロード、WDT 再起動、または IRQ クリアを実行します。他の値を書き込むと、ウォッチドッグのリセット／IRQ を実行します。書込み専用ビットで、読み出すと 0 を返します。

ステータス

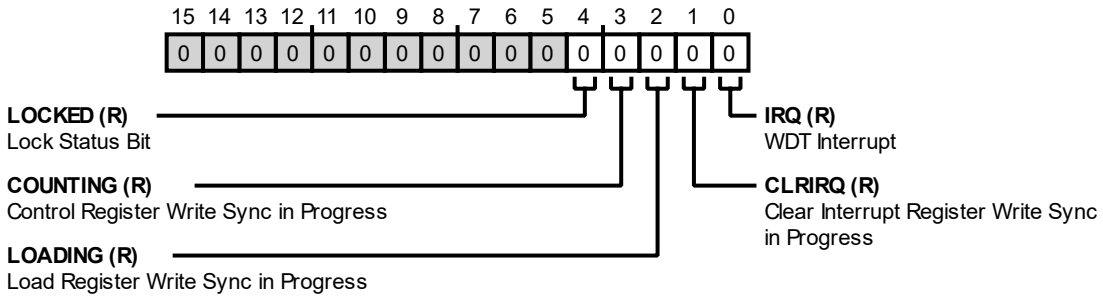


図 24-6：WDT_STAT レジスタ図

表 24-6：WDT_STAT レジスタ・フィールド

ビット番号（アクセス）	ビット名	ビット値／機能対応	
4 (R/NW)	LOCKED	ロック・ステータス・ビット。 ユーザ・コードで WDT_CTL.EN をセットした場合、ハードウェアで自動的にセットされます。デフォルトではクリアされ、ユーザ・コードで WDT_CTL.EN をセットするまではクリアを維持します。	
		0	ソフトウェアによりイネーブル・ビットがセットされていない
		1	ソフトウェアによりイネーブル・ビットがセットされている。WDT がロックされている
3 (R/NW)	COUNTING	制御レジスタ書込み同期の処理中。 同期の問題を回避するためにこのビットをセットしている場合は、ソフトウェアで WDT_CTL レジスタに書込みを行わないでください。	
		0	APB と WDT のクロック・ドメインで WDT_CTL 値が一致している
		1	APB の WDT_CTL レジスタ値を WDT のクロック・ドメインの値に同期する処理を実行中
2 (R/NW)	LOADING	ロード・レジスタ書込み同期の処理中。 同期の問題を回避するためにこのビットをセットしている場合は、ソフトウェアで WDT_LOAD レジスタに書込みを行わないでください。	
		0	APB と WDT のクロック・ドメインで WDT_LOAD 値が一致している
		1	APB の WDT_LOAD レジスタ値を WDT のクロック・ドメインの値に同期する処理を実行中

表 24-6：WDT_STAT レジスタ・フィールド（続き）

ビット番号（アクセス）	ビット名	ビット値／機能対応	
1 (R/NW)	CLRIRQ	割込みクリア・レジスタ書込み同期の処理中。 同期問題を回避するためにこのビットをセットしている場合は、ソフトウェアで RESTART レジスタに書込みを行わないでください。	
		0	APB の WDT_RESTART 書込みの同期が完了していない。
		1	APB の WDT_RESTART 書込みが WDT のクロック・ドメインに同期されている。同期が完了すると、WDT が再起動される（0xCCCC を書き込んだ場合）。
0 (R/NW)	IRQ	WDT 割込み このビットをクリアするには、WDT_RESTART レジスタに値 0xCCCC を書き込みます。	
		0	WDT 割込みは保留されていない。
		1	WDT 割込みが保留中。

25 ADuCM4050 レジスター一覧

ここでは、メモリマップド・レジスタのアドレスとレジスタ名を一覧で示します。モジュールはアルファベット順に記載されています。

表 25-1：ADuCM4050 ADC0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40007000	ADC0_CFG	ADC0 ADC の設定	0x00000000
0x40007004	ADC0_PWRUP	ADC0 ADC パワーアップ時間	0x0000041C
0x40007008	ADC0_CAL_WORD	ADC0 キャリブレーション・ワード	0x00000040
0x4000700C	ADC0_CNV_CFG	ADC0 ADC 変換の設定	0x00000000
0x40007010	ADC0_CNV_TIME	ADC0 ADC 変換時間	0x00000000
0x40007014	ADC0_AVG_CFG	ADC0 平均化の設定	0x00004008
0x40007020	ADC0_IRQ_EN	ADC0 割込みイネーブル	0x00000000
0x40007024	ADC0_STAT	ADC0 ADC ステータス	0x00000000
0x40007028	ADC0_OVF	ADC0 出力レジスタのオーバーフロー	0x00000000
0x4000702C	ADC0_ALERT	ADC0 アラート表示	0x00000000
0x40007030	ADC0_CH0_OUT	ADC0 変換結果チャンネル 0	0x00000000
0x40007034	ADC0_CH1_OUT	ADC0 変換結果チャンネル 1	0x00000000
0x40007038	ADC0_CH2_OUT	ADC0 変換結果チャンネル 2	0x00000000
0x4000703C	ADC0_CH3_OUT	ADC0 変換結果チャンネル 3	0x00000000
0x40007040	ADC0_CH4_OUT	ADC0 変換結果チャンネル 4	0x00000000
0x40007044	ADC0_CH5_OUT	ADC0 変換結果チャンネル 5	0x00000000
0x40007048	ADC0_CH6_OUT	ADC0 変換結果チャンネル 6	0x00000000
0x4000704C	ADC0_CH7_OUT	ADC0 変換結果チャンネル 7	0x00000000
0x40007050	ADC0_BAT_OUT	ADC0 バッテリ監視結果	0x00000000
0x40007054	ADC0_TMP_OUT	ADC0 温度結果	0x00000000

表 25-1：ADuCM4050 ADC0 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40007058	ADC0_TMP2_OUT	ADC0 温度結果 2	0x00000000
0x4000705C	ADC0_DMA_OUT	ADC0 DMA 出力レジスタ	0x00000000
0x40007060	ADC0_LIM0_LO	ADC0 チャンネル 0 の下限	0x00000000
0x40007064	ADC0_LIM0_HI	ADC0 チャンネル 0 の上限	0x00000FFF
0x40007068	ADC0_HYS0	ADC0 チャンネル 0 のヒステリシス	0x00000000
0x40007070	ADC0_LIM1_LO	ADC0 チャンネル 1 の下限	0x00000000
0x40007074	ADC0_LIM1_HI	ADC0 チャンネル 1 の上限	0x00000FFF
0x40007078	ADC0_HYS1	ADC0 チャンネル 1 のヒステリシス	0x00000000
0x40007080	ADC0_LIM2_LO	ADC0 チャンネル 2 の下限	0x00000000
0x40007084	ADC0_LIM2_HI	ADC0 チャンネル 2 の上限	0x00000FFF
0x40007088	ADC0_HYS2	ADC0 チャンネル 2 のヒステリシス	0x00000000
0x40007090	ADC0_LIM3_LO	ADC0 チャンネル 3 の下限	0x00000000
0x40007094	ADC0_LIM3_HI	ADC0 チャンネル 3 の上限	0x00000FFF
0x40007098	ADC0_HYS3	ADC0 チャンネル 3 のヒステリシス	0x00000000
0x400070C0	ADC0_CFG1	ADC0 リファレンス・バッファ低消費電力モード	0x00000400

表 25-2：ADuCM4050 BEEP0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40005C00	BEEP0_CFG	BEEP0 ビーパ設定	0x00000000
0x40005C04	BEEP0_STAT	BEEP0 ビーパ・ステータス	0x00000000
0x40005C08	BEEP0_TONEA	BEEP0 トーン A のデータ	0x00000001
0x40005C0C	BEEP0_TONEB	BEEP0 トーン B のデータ	0x00000001

表 25-3：ADuCM4050 BUSM0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x4004C800	BUSM0_ARBIT0	BUSM0 FLASH および SRAM0 のアービトレーション優先度設定	0x00240024
0x4004C804	BUSM0_ARBIT1	BUSM0 SRAM1 のアービトレーション優先度設定	0x00240024
0x4004C808	BUSM0_ARBIT2	BUSM0 APB32 および APB16 のアービトレーション優先度設定	0x00240024

表 25-3：ADuCM4050 BUSM0 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x4004C80C	BUSM0_ARBIT3	BUSM0 コアと DMA1 の APB16 優先度に関するアービトレーション優先度設定	0x00010002
0x4004C814	BUSM0_ARBIT4	BUSM0 SRAM1 および SIP のアービトレーション優先度設定	0x00000024

表 25-4：ADuCM4050 CLKG0_OSC MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x4004C10C	CLKG0_OSC_KEY	CLKG0 OSCCTRL のキー保護	0x00000000
0x4004C110	CLKG0_OSC_CTL	CLKG0 発振器制御	0x00060002

表 25-5：ADuCM4050 CLKG0_CLK MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x4004C300	CLKG0_CLK_CTL0	CLKG0 その他のクロック設定	0x00000078
0x4004C304	CLKG0_CLK_CTL1	CLKG0 クロック分周器	0x00100404
0x4004C308	CLKG0_CLK_CTL2	CLKG0 HF 発振器の分周クロックの選択	0x00000000
0x4004C30C	CLKG0_CLK_CTL3	CLKG0 システム PLL	0x0000681A
0x4004C314	CLKG0_CLK_CTL5	CLKG0 ユーザ・クロック・ゲート制御	0x0000005F
0x4004C318	CLKG0_CLK_STAT0	CLKG0 クロック・ステータス	0x00000000

表 25-6：ADuCM4050 CRC0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40040000	CRC0_CTL	CRC0 CRC コントロール	0x10000000
0x40040004	CRC0_IPDATA	CRC0 入力データ・ワード	0x00000000
0x40040008	CRC0_RESULT	CRC0 CRC 結果	0x00000000
0x4004000C	CRC0_POLY	CRC0 プログラマブル CRC 多項式	0x04C11DB7
0x40040010	CRC0_IPBITS[n]	CRC0 入力データ・ビット	0x00000000
0x40040010	CRC0_IPBYTE	CRC0 入力データ・バイト	0x00000000
0x40040011	CRC0_IPBITS[n]	CRC0 入力データ・ビット	0x00000000
0x40040012	CRC0_IPBITS[n]	CRC0 入力データ・ビット	0x00000000
0x40040013	CRC0_IPBITS[n]	CRC0 入力データ・ビット	0x00000000

表 25-6：ADuCM4050 CRC0 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40040014	CRC0_IPBITS[n]	CRC0 入力データ・ビット	0x00000000
0x40040015	CRC0_IPBITS[n]	CRC0 入力データ・ビット	0x00000000
0x40040016	CRC0_IPBITS[n]	CRC0 入力データ・ビット	0x00000000
0x40040017	CRC0_IPBITS[n]	CRC0 入力データ・ビット	0x00000000

表 25-7：ADuCM4050 CRYPT0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40044000	CRYPT0_CFG	CRYPT0 設定レジスタ	0x20000000
0x40044004	CRYPT0_DATALEN	CRYPT0 ペイロード・データ長	0x00000000
0x40044008	CRYPT0_PREFIXLEN	CRYPT0 認証データ長	0x00000000
0x4004400C	CRYPT0_INTEN	CRYPT0 割込みイネーブル・レジスタ	0x00000000
0x40044010	CRYPT0_STAT	CRYPT0 ステータス・レジスタ	0x00000001
0x40044014	CRYPT0_INBUF	CRYPT0 入力バッファ	0x00000000
0x40044018	CRYPT0_OUTBUF	CRYPT0 出力バッファ	0x00000000
0x4004401C	CRYPT0_NONCE0	CRYPT0 ノンス・ビット［31：0］	0x00000000
0x40044020	CRYPT0_NONCE1	CRYPT0 ノンス・ビット［63：32］	0x00000000
0x40044024	CRYPT0_NONCE2	CRYPT0 ノンス・ビット［95：64］	0x00000000
0x40044028	CRYPT0_NONCE3	CRYPT0 ノンス・ビット［127：96］	0x00000000
0x4004402C	CRYPT0_AESKEY0	CRYPT0 AES 鍵ビット［31：0］	0x00000000
0x40044030	CRYPT0_AESKEY1	CRYPT0 AES 鍵ビット［63：32］	0x00000000
0x40044034	CRYPT0_AESKEY2	CRYPT0 AES 鍵ビット［95：64］	0x00000000
0x40044038	CRYPT0_AESKEY3	CRYPT0 AES 鍵ビット［127：96］	0x00000000
0x4004403C	CRYPT0_AESKEY4	CRYPT0 AES 鍵ビット［159：128］	0x00000000
0x40044040	CRYPT0_AESKEY5	CRYPT0 AES 鍵ビット［191：160］	0x00000000
0x40044044	CRYPT0_AESKEY6	CRYPT0 AES 鍵ビット［223：192］	0x00000000
0x40044048	CRYPT0_AESKEY7	CRYPT0 AES 鍵ビット［255：224］	0x00000000
0x4004404C	CRYPT0_CNTRINIT	CRYPT0 カウンタ初期化ベクトル	0x00000000
0x40044050	CRYPT0_SHAH0	CRYPT0 SHA ビット［31：0］	0x6A09E667
0x40044054	CRYPT0_SHAH1	CRYPT0 SHA ビット［63：32］	0xBB67AE85
0x40044058	CRYPT0_SHAH2	CRYPT0 SHA ビット［95：64］	0x3C6EF372

表 25-7：ADuCM4050 CRYPT0 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x4004405C	CRYPT0_SHAH3	CRYPT0 SHA ビット［127：96］	0xA54FF53A
0x40044060	CRYPT0_SHAH4	CRYPT0 SHA ビット［159：128］	0x510E527F
0x40044064	CRYPT0_SHAH5	CRYPT0 SHA ビット［191：160］	0x9B05688C
0x40044068	CRYPT0_SHAH6	CRYPT0 SHA ビット［223：192］	0x1F83D9AB
0x4004406C	CRYPT0_SHAH7	CRYPT0 SHA ビット［255：224］	0x5BE0CD19
0x40044070	CRYPT0_SHA_LAST_WOR D	CRYPT0 SHA 最終ワードおよび有効ビットの情報	0x00000000
0x40044074	CRYPT0_CCM_NUM_VALID_ BYTES	CRYPT0 NUM_VALID_BYTES	0x00000000
0x40044078	CRYPT0_PRKSTORCFG	CRYPT0 PRKSTOR 構成	0x00000000
0x40044080	CRYPT0_KUW0	CRYPT0 鍵ラップ／アンラップ・レジスタ 0	0x00000000
0x40044084	CRYPT0_KUW1	CRYPT0 鍵ラップ／アンラップ・レジスタ 1	0x00000000
0x40044088	CRYPT0_KUW2	CRYPT0 鍵ラップ／アンラップ・レジスタ 2	0x00000000
0x4004408C	CRYPT0_KUW3	CRYPT0 鍵ラップ／アンラップ・レジスタ 3	0x00000000
0x40044090	CRYPT0_KUW4	CRYPT0 鍵ラップ／アンラップ・レジスタ 4	0x00000000
0x40044094	CRYPT0_KUW5	CRYPT0 鍵ラップ／アンラップ・レジスタ 5	0x00000000
0x40044098	CRYPT0_KUW6	CRYPT0 鍵ラップ／アンラップ・レジスタ 6	0x00000000
0x4004409C	CRYPT0_KUW7	CRYPT0 鍵ラップ／アンラップ・レジスタ 7	0x00000000
0x400440A0	CRYPT0_KUW8	CRYPT0 鍵ラップ／アンラップ・レジスタ 8	0x00000000
0x400440A4	CRYPT0_KUW9	CRYPT0 鍵ラップ／アンラップ・レジスタ 9	0x00000000
0x400440A8	CRYPT0_KUW10	CRYPT0 鍵ラップ／アンラップ・レジスタ 10	0x00000000
0x400440AC	CRYPT0_KUW11	CRYPT0 鍵ラップ／アンラップ・レジスタ 11	0x00000000
0x400440B0	CRYPT0_KUW12	CRYPT0 鍵ラップ／アンラップ・レジスタ 12	0x00000000
0x400440B4	CRYPT0_KUW13	CRYPT0 鍵ラップ／アンラップ・レジスタ 13	0x00000000
0x400440B8	CRYPT0_KUW14	CRYPT0 鍵ラップ／アンラップ・レジスタ 14	0x00000000
0x400440BC	CRYPT0_KUW15	CRYPT0 鍵ラップ／アンラップ・レジスタ 15	0x00000000
0x400440C0	CRYPT0_KUWVALSTR1	CRYPT0 鍵ラップ／アンラップ検証文字列［63：32］	0xA6A6A6A6
0x400440C4	CRYPT0_KUWVALSTR2	CRYPT0 鍵ラップ／アンラップ検証文字列［31：0］	0xA6A6A6A6

表 25-8：ADuCM4050 DMA0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40010000	DMA0_STAT	DMA0 DMA ステータス	0x001A0000
0x40010004	DMA0_CFG	DMA0 DMA 構成	0x00000000
0x40010008	DMA0_PDBPTR	DMA0 DMA チャンネルのプライマリ制御データベース・ポインタ	0x00000000
0x4001000C	DMA0_ADBPTR	DMA0 DMA チャンネルの代替制御データベース・ポインタ	0x00000200
0x40010014	DMA0_SWREQ	DMA0 DMA チャンネルのソフトウェア・リクエスト	0x00000000
0x40010020	DMA0_RMSK_SET	DMA0 DMA チャンネルのリクエスト・マスクをセット	0x00000000
0x40010024	DMA0_RMSK_CLR	DMA0 DMA チャンネルのリクエスト・マスクをクリア	0x00000000
0x40010028	DMA0_EN_SET	DMA0 DMA チャンネル・イネーブルをセット	0x00000000
0x4001002C	DMA0_EN_CLR	DMA0 DMA チャンネル・イネーブルをクリア	0x00000000
0x40010030	DMA0_ALT_SET	DMA0 DMA チャンネルでのプライマリ代替をセット	0x00000000
0x40010034	DMA0_ALT_CLR	DMA0 DMA チャンネルのプライマリ代替をクリア	0x00000000
0x40010038	DMA0_PRI_SET	DMA0 DMA チャンネルの優先度を設定	0x00000000
0x4001003C	DMA0_PRI_CLR	DMA0 DMA チャンネルの優先度をクリア	0x00000000
0x40010048	DMA0_ERRCHNL_CLR	DMA0 DMA エラーをチャンネル単位でクリア	0x00000000
0x4001004C	DMA0_ERR_CLR	DMA0 DMA バス・エラーをクリア	0x00000000
0x40010050	DMA0_INVALID- DESC_CLR	DMA0 DMA の無効なディスクリプタをチャンネル単位でクリア	0x00000000
0x40010800	DMA0_BS_SET	DMA0 DMA チャンネルのバイト・スワップ・イネーブルをセット	0x00000000
0x40010804	DMA0_BS_CLR	DMA0 DMA チャンネルのバイト・スワップ・イネーブルをクリア	0x00000000
0x40010810	DMA0_SRCADDR_SET	DMA0 DMA チャンネルのソース・アドレス・デクリメント・イネーブルをセット	0x00000000
0x40010814	DMA0_SRCADDR_CLR	DMA0 DMA チャンネルのソース・アドレス・デクリメント・イネーブルをクリア	0x00000000
0x40010818	DMA0_DSTADDR_SET	DMA0 DMA チャンネルのデスティネーション・アドレス・デクリメント・イネーブルをセット	0x00000000
0x4001081C	DMA0_DSTADDR_CLR	DMA0 DMA チャンネルのデスティネーション・アドレス・デクリメント・イネーブルをクリア	0x00000000
0x40010FE0	DMA0_REVID	DMA0 DMA コントローラのリビジョン ID	0x00000002

表 25-9：ADuCM4050 XINT0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
	XINT0_CFG0	XINT0 外部割込みの設定	0x00200000

表 25-9：ADuCM4050 XINT0 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x4004C080			
0x4004C084	XINT0_EXT_STAT	XINT0 外部ウェイクアップ割込みステータス	0x00000000
0x4004C090	XINT0_CLR	XINT0 外部割込みのクリア	0x00000000
0x4004C094	XINT0_NMICLR	XINT0 マスク不能割込みのクリア	0x00000000

表 25-10：ADuCM4050 FLCC0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40018000	FLCC0_STAT	FLCC0 ステータス	0x00000000
0x40018004	FLCC0_IEN	FLCC0 割込み許可	0x00000060
0x40018008	FLCC0_CMD	FLCC0 コマンド	0x00000000
0x4001800C	FLCC0_KH_ADDR	FLCC0 書込みアドレス	0x00000000
0x40018010	FLCC0_KH_DATA0	FLCC0 下位データの書込み	0xFFFFFFFF
0x40018014	FLCC0_KH_DATA1	FLCC0 上位データの書込み	0xFFFFFFFF
0x40018018	FLCC0_PAGE_ADDR0	FLCC0 下位ページ・アドレス	0x00000000
0x4001801C	FLCC0_PAGE_ADDR1	FLCC0 上位ページ・アドレス	0x00000000
0x40018020	FLCC0_KEY	FLCC0 キー	0x00000000
0x40018024	FLCC0_WR_ABORT_ADD R	FLCC0 書込みアボート・アドレス	0x00000000
0x40018028	FLCC0_WRPROT	FLCC0 書込み保護	0xFFFFFFFF
0x4001802C	FLCC0_SIGNATURE	FLCC0 シグネチャ	0x00000000
0x40018030	FLCC0_UCFG	FLCC0 ユーザ設定	0x00000000
0x40018034	FLCC0_TIME_PARAM0	FLCC0 時間パラメータ 0	0xB8954950
0x40018038	FLCC0_TIME_PARAM1	FLCC0 時間パラメータ 1	0x00000004
0x4001803C	FLCC0_ABORT_EN_LO	FLCC0 IRQ アボート・イネーブル（下位ビット）	0x00000000
0x40018040	FLCC0_ABORT_EN_HI	FLCC0 IRQ アボート・イネーブル（上位ビット）	0x00000000
0x40018048	FLCC0_ECC_ADDR	FLCC0 ECC ステータス（アドレス）	0x00000000
0x40018050	FLCC0_POR_SEC	FLCC0 フラッシュ・セキュリティ	0x00000000
0x40018054	FLCC0_VOL_CFG	FLCC0 揮発性フラッシュの設定	0x00000001

表 25-11：ADuCM4050 FLCC0_CACHE MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40018058	FLCC0_CACHE_STAT	FLCC0 キャッシュ・ステータス・レジスタ	0x00000000
0x4001805C	FLCC0_CACHE_SETUP	FLCC0 キャッシュ・セットアップ・レジスタ	0x00000000
0x40018060	FLCC0_CACHE_KEY	FLCC0 キャッシュ・キー・レジスタ	0x00000000

表 25-12：ADuCM4050 GPIO0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40020000	GPIO0_CFG	GPIO0 ポート構成	0x00000000
0x40020004	GPIO0_OEN	GPIO0 ポート出力イネーブル	0x00000000
0x40020008	GPIO0_PE	GPIO0 ポート出力のプルアップ／プルダウンの有効化	0x000000C0
0x4002000C	GPIO0_IEN	GPIO0 ポート入力パスのイネーブル	0x00000000
0x40020010	GPIO0_IN	GPIO0 ポート・レジスタ・データ入力	0x00000000
0x40020014	GPIO0_OUT	GPIO0 ポート・データ出力	0x00000000
0x40020018	GPIO0_SET	GPIO0 ポート・データ出力のセット	0x00000000
0x4002001C	GPIO0_CLR	GPIO0 ポート・データ出力のクリア	0x00000000
0x40020020	GPIO0_TGL	GPIO0 ポート・ピンのトグル	0x00000000
0x40020024	GPIO0_POL	GPIO0 ポート割込み極性	0x00000000
0x40020028	GPIO0_IENA	GPIO0 ポート割込み A の有効化	0x00000000
0x4002002C	GPIO0_IENB	GPIO0 ポート割込み B の有効化	0x00000000
0x40020030	GPIO0_INT	GPIO0 ポート割込みステータス	0x00000000
0x40020034	GPIO0_DS	GPIO0 ポート駆動能力の選択	0x00000000

表 25-13：ADuCM4050 GPIO1 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40020040	GPIO1_CFG	GPIO1 ポート構成	0x00000000
0x40020044	GPIO1_OEN	GPIO1 ポート出力イネーブル	0x00000000
0x40020048	GPIO1_PE	GPIO1 ポート出力のプルアップ／プルダウンの有効化	0x00000002
0x4002004C	GPIO1_IEN	GPIO1 ポート入力パスのイネーブル	0x00000000
0x40020050	GPIO1_IN	GPIO1 ポート・レジスタ・データ入力	0x00000000
0x40020054	GPIO1_OUT	GPIO1 ポート・データ出力	0x00000000
0x40020058	GPIO1_SET	GPIO1 ポート・データ出力のセット	0x00000000

表 25-13：ADuCM4050 GPIO1 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x4002005C	GPIO1_CLR	GPIO1 ポート・データ出力のクリア	0x00000000
0x40020060	GPIO1_TGL	GPIO1 ポート・ピンのトグル	0x00000000
0x40020064	GPIO1_POL	GPIO1 ポート割込み極性	0x00000000
0x40020068	GPIO1_IENA	GPIO1 ポート割込み A の有効化	0x00000000
0x4002006C	GPIO1_IENB	GPIO1 ポート割込み B の有効化	0x00000000
0x40020070	GPIO1_INT	GPIO1 ポート割込みステータス	0x00000000
0x40020074	GPIO1_DS	GPIO1 ポート駆動能力の選択	0x00000000

表 25-14：ADuCM4050 GPIO2 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40020080	GPIO2_CFG	GPIO2 ポート構成	0x00000000
0x40020084	GPIO2_OEN	GPIO2 ポート出力イネーブル	0x00000000
0x40020088	GPIO2_PE	GPIO2 ポート出力のプルアップ／プルダウンの有効化	0x00000000
0x4002008C	GPIO2_IEN	GPIO2 ポート入力パスのイネーブル	0x00000000
0x40020090	GPIO2_IN	GPIO2 ポート・レジスタ・データ入力	0x00000000
0x40020094	GPIO2_OUT	GPIO2 ポート・データ出力	0x00000000
0x40020098	GPIO2_SET	GPIO2 ポート・データ出力のセット	0x00000000
0x4002009C	GPIO2_CLR	GPIO2 ポート・データ出力のクリア	0x00000000
0x400200A0	GPIO2_TGL	GPIO2 ポート・ピンのトグル	0x00000000
0x400200A4	GPIO2_POL	GPIO2 ポート割込み極性	0x00000000
0x400200A8	GPIO2_IENA	GPIO2 ポート割込み A の有効化	0x00000000
0x400200AC	GPIO2_IENB	GPIO2 ポート割込み B の有効化	0x00000000
0x400200B0	GPIO2_INT	GPIO2 ポート割込みステータス	0x00000000
0x400200B4	GPIO2_DS	GPIO2 ポート駆動能力の選択	0x00000000

表 25-15：ADuCM4050 GPIO3 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x400200C0	GPIO3_CFG	GPIO3 ポート構成	0x00000000
0x400200C4	GPIO3_OEN	GPIO3 ポート出力イネーブル	0x00000000
0x400200C8	GPIO3_PE	GPIO3 ポート出力のプルアップ／プルダウンの有効化	0x00000000

表 25-15：ADuCM4050 GPIO3 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x400200CC	GPIO3_IEN	GPIO3 ポート入力パスのイネーブル	0x00000000
0x400200D0	GPIO3_IN	GPIO3 ポート・レジスタ・データ入力	0x00000000
0x400200D4	GPIO3_OUT	GPIO3 ポート・データ出力	0x00000000
0x400200D8	GPIO3_SET	GPIO3 ポート・データ出力のセット	0x00000000
0x400200DC	GPIO3_CLR	GPIO3 ポート・データ出力のクリア	0x00000000
0x400200E0	GPIO3_TGL	GPIO3 ポート・ピンのトグル	0x00000000
0x400200E4	GPIO3_POL	GPIO3 ポート割込み極性	0x00000000
0x400200E8	GPIO3_IENA	GPIO3 ポート割込み A の有効化	0x00000000
0x400200EC	GPIO3_IENB	GPIO3 ポート割込み B の有効化	0x00000000
0x400200F0	GPIO3_INT	GPIO3 ポート割込みステータス	0x00000000
0x400200F4	GPIO3_DS	GPIO3 ポート駆動能力の選択	0x00000000

表 25-16：ADuCM4050 TMR0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40000000	TMR0_LOAD	TMR0 16 ビット・ロード値	0x00000000
0x40000004	TMR0_CURCNT	TMR0 16 ビット・タイマー値	0x00000000
0x40000008	TMR0_CTL	TMR0 制御	0x0000000A
0x4000000C	TMR0_CLRINT	TMR0 割込みクリア	0x00000000
0x40000010	TMR0_CAPTURE	TMR0 キャプチャ	0x00000000
0x40000014	TMR0_ALOAD	TMR0 16 ビット・ロード値、非同期	0x00000000
0x40000018	TMR0_ACURCNT	TMR0 16 ビット・タイマー値、非同期	0x00000000
0x4000001C	TMR0_STAT	TMR0 ステータス	0x00000000
0x40000020	TMR0_PWMCTL	TMR0 PWM 制御レジスタ	0x00000000
0x40000024	TMR0_PWMMATCH	TMR0 PWM マッチ値	0x00000000
0x40000028	TMR0_EVENTSELECT	TMR0 タイマー・イベント選択レジスタ	0x00000000

表 25-17：ADuCM4050 TMR1 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40000400	TMR1_LOAD	TMR1 16 ビット・ロード値	0x00000000
0x40000404	TMR1_CURCNT	TMR1 16 ビット・タイマー値	0x00000000

表 25-17：ADuCM4050 TMR1 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40000408	TMR1_CTL	TMR1 制御	0x0000000A
0x4000040C	TMR1_CLRINT	TMR1 割込みクリア	0x00000000
0x40000410	TMR1_CAPTURE	TMR1 キャプチャ	0x00000000
0x40000414	TMR1_ALOAD	TMR1 16 ビット・ロード値、非同期	0x00000000
0x40000418	TMR1_ACURCNT	TMR1 16 ビット・タイマー値、非同期	0x00000000
0x4000041C	TMR1_STAT	TMR1 ステータス	0x00000000
0x40000420	TMR1_PWMCTL	TMR1 PWM 制御レジスタ	0x00000000
0x40000424	TMR1_PWMMATCH	TMR1 PWM マッチ値	0x00000000
0x40000428	TMR1_EVENTSELECT	TMR1 タイマー・イベント選択レジスタ	0x00000000

表 25-18：ADuCM4050 TMR2 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40000800	TMR2_LOAD	TMR2 16 ビット・ロード値	0x00000000
0x40000804	TMR2_CURCNT	TMR2 16 ビット・タイマー値	0x00000000
0x40000808	TMR2_CTL	TMR2 制御	0x0000000A
0x4000080C	TMR2_CLRINT	TMR2 割込みクリア	0x00000000
0x40000810	TMR2_CAPTURE	TMR2 キャプチャ	0x00000000
0x40000814	TMR2_ALOAD	TMR2 16 ビット・ロード値、非同期	0x00000000
0x40000818	TMR2_ACURCNT	TMR2 16 ビット・タイマー値、非同期	0x00000000
0x4000081C	TMR2_STAT	TMR2 ステータス	0x00000000
0x40000820	TMR2_PWMCTL	TMR2 PWM 制御レジスタ	0x00000000
0x40000824	TMR2_PWMMATCH	TMR2 PWM マッチ値	0x00000000
0x40000828	TMR2_EVENTSELECT	TMR2 タイマー・イベント選択レジスタ	0x00000000

表 25-19：ADuCM4050 I2C0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40003000	I2C0_MCTL	I2C0 マスタ・コントロール	0x00000000
0x40003004	I2C0_MSTAT	I2C0 マスタ・ステータス	0x00006000
0x40003008	I2C0_MRX	I2C0 マスタ受信データ	0x00000000
0x4000300C	I2C0_MTX	I2C0 マスタ送信データ	0x00000000

表 25-19：ADuCM4050 I2C0 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40003010	I2C0_MRXCNT	I2C0 マスタ受信データ・カウント	0x00000000
0x40003014	I2C0_MCRXCNT	I2C0 現在のマスタ受信データ・カウント	0x00000000
0x40003018	I2C0_ADDR1	I2C0 マスタ・アドレス・バイト 1	0x00000000
0x4000301C	I2C0_ADDR2	I2C0 マスタ・アドレス・バイト 2	0x00000000
0x40003020	I2C0_BYT	I2C0 開始バイト	0x00000000
0x40003024	I2C0_DIV	I2C0 シリアル・クロック分周器	0x00001F1F
0x40003028	I2C0_SCTL	I2C0 スレーブ・コントロール	0x00000000
0x4000302C	I2C0_SSTAT	I2C0 スレーブの I2C ステータス／エラー／IRQ	0x00000001
0x40003030	I2C0_SRX	I2C0 スレーブ受信	0x00000000
0x40003034	I2C0_STX	I2C0 スレーブ送信	0x00000000
0x40003038	I2C0_ALT	I2C0 ハードウェア・ジェネラル・コール ID	0x00000000
0x4000303C	I2C0_ID0	I2C0 1 番目のスレーブ・アドレス・デバイス ID	0x00000000
0x40003040	I2C0_ID1	I2C0 2 番目のスレーブ・アドレス・デバイス ID	0x00000000
0x40003044	I2C0_ID2	I2C0 3 番目のスレーブ・アドレス・デバイス ID	0x00000000
0x40003048	I2C0_ID3	I2C0 4 番目のスレーブ・アドレス・デバイス ID	0x00000000
0x4000304C	I2C0_STAT	I2C0 マスタおよびスレーブ FIFO ステータス	0x00000000
0x40003050	I2C0_SHCTL	I2C0 共通コントロール	0x00000000
0x40003054	I2C0_TCTL	I2C0 タイミング・コントロール・レジスタ	0x00000005
0x40003058	I2C0_ASTRETCH_SCL	I2C0 SCL の自動ストレッチング	0x00000000

表 25-20：ADuCM4050 NVIC0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0xE000E004	NVIC0_INTNUM	NVIC0 割込み制御タイプ	0x00000000
0xE000E010	NVIC0_STKSTA	NVIC0 Systick 制御およびステータス	0x00000000
0xE000E014	NVIC0_STKLD	NVIC0 Systick リロード値	0x00000000
0xE000E018	NVIC0_STKVAL	NVIC0 Systick 現在値	0x00000000
0xE000E01C	NVIC0_STKCAL	NVIC0 Systick キャリブレーション値	0x00000000
0xE000E100	NVIC0_INTSETE0	NVIC0 IRQ0～31 イネーブル・セット	0x00000000
0xE000E104	NVIC0_INTSETE1	NVIC0 IRQ32～63 イネーブル・セット	0x00000000
0xE000E108	NVIC0_INTSETE2	NVIC0 IRQ64～95 イネーブル・セット	0x00000000

表 25-20：ADuCM4050 NVIC0 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0xE000E180	NVIC0_INTCLRE0	NVIC0 IRQ0～31 イネーブル・クリア	0x00000000
0xE000E184	NVIC0_INTCLRE1	NVIC0 IRQ32～63 イネーブル・クリア	0x00000000
0xE000E188	NVIC0_INTCLRE2	NVIC0 IRQ64～95 イネーブル・クリア	0x00000000
0xE000E200	NVIC0_INTSETP0	NVIC0 IRQ0～31 保留セット	0x00000000
0xE000E204	NVIC0_INTSETP1	NVIC0 IRQ32～63 保留セット	0x00000000
0xE000E208	NVIC0_INTSETP2	NVIC0 IRQ64～95 保留セット	0x00000000
0xE000E280	NVIC0_INTCLRP0	NVIC0 IRQ0～31 保留クリア	0x00000000
0xE000E284	NVIC0_INTCLRP1	NVIC0 IRQ32～63 保留クリア	0x00000000
0xE000E288	NVIC0_INTCLRP2	NVIC0 IRQ64～95 保留クリア	0x00000000
0xE000E300	NVIC0_INTACT0	NVIC0 IRQ0～31 アクティブ・ビット	0x00000000
0xE000E304	NVIC0_INTACT1	NVIC0 IRQ32～63 アクティブ・ビット	0x00000000
0xE000E308	NVIC0_INTACT2	NVIC0 IRQ64～95 アクティブ・ビット	0x00000000
0xE000E400	NVIC0_INTPRI0	NVIC0 IRQ0～3 優先度	0x00000000
0xE000E404	NVIC0_INTPRI1	NVIC0 IRQ4～7 優先度	0x00000000
0xE000E408	NVIC0_INTPRI2	NVIC0 IRQ8～11 優先度	0x00000000
0xE000E40C	NVIC0_INTPRI3	NVIC0 IRQ12～15 優先度	0x00000000
0xE000E410	NVIC0_INTPRI4	NVIC0 IRQ16～19 優先度	0x00000000
0xE000E414	NVIC0_INTPRI5	NVIC0 IRQ20～23 優先度	0x00000000
0xE000E418	NVIC0_INTPRI6	NVIC0 IRQ24～27 優先度	0x00000000
0xE000E41C	NVIC0_INTPRI7	NVIC0 IRQ28～31 優先度	0x00000000
0xE000E420	NVIC0_INTPRI8	NVIC0 IRQ32～35 優先度	0x00000000
0xE000E424	NVIC0_INTPRI9	NVIC0 IRQ36～39 優先度	0x00000000
0xE000E428	NVIC0_INTPRI10	NVIC0 IRQ40～43 優先度	0x00000000
0xE000E42C	NVIC0_INTPRI11	NVIC0 IRQ44～47 優先度	0x00000000
0xE000E430	NVIC0_INTPRI12	NVIC0 IRQ48～51 優先度	0x00000000
0xE000E434	NVIC0_INTPRI13	NVIC0 IRQ52～55 優先度	0x00000000
0xE000E438	NVIC0_INTPRI14	NVIC0 IRQ56～59 優先度	0x00000000
0xE000E43C	NVIC0_INTPRI15	NVIC0 IRQ60～63 優先度	0x00000000
0xE000E440	NVIC0_INTPRI16	NVIC0 IRQ64～67 優先度	0x00000000
0xE000E444	NVIC0_INTPRI17	NVIC0 IRQ68～71 優先度	0x00000000
0xE000ED00	NVIC0_INTCPID	NVIC0 CPUID ベース	0x00000000

表 25-20：ADuCM4050 NVIC0 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0xE000ED04	NVIC0_INTSTA	NVIC0 割込み制御状態	0x00000000
0xE000ED08	NVIC0_INTVEC	NVIC0 ベクタ･テーブル・オフセット	0x00000000
0xE000ED0C	NVIC0_INTAIRC	NVIC0 アプリケーション割込み／リセット制御	0x00000000
0xE000ED10	NVIC0_INTCON0	NVIC0 システム制御	0x00000000
0xE000ED14	NVIC0_INTCON1	NVIC0 構成制御	0x00000000
0xE000ED18	NVIC0_INTSHPRIO0	NVIC0 システム･ハンドラ 4-7 優先度	0x00000000
0xE000ED1C	NVIC0_INTSHPRIO1	NVIC0 システム･ハンドラ 8-11 優先度	0x00000000
0xE000ED20	NVIC0_INTSHPRIO3	NVIC0 システム･ハンドラ 12-15 優先度	0x00000000
0xE000ED24	NVIC0_INTSHCSR	NVIC0 システム･ハンドラ制御および状態	0x00000000
0xE000ED28	NVIC0_INTCFSR	NVIC0 構成可能フォールト・ステータス	0x00000000
0xE000ED2C	NVIC0_INTHFSR	NVIC0 ハード・フォールト・ステータス	0x00000000
0xE000ED30	NVIC0_INTDFSR	NVIC0 デバッグ・フォールト・ステータス	0x00000000
0xE000ED34	NVIC0_INTMMAR	NVIC0 メモリ管理アドレス	0x00000000
0xE000ED38	NVIC0_INTBFAR	NVIC0 バス･フォールト･アドレス	0x00000000
0xE000ED3C	NVIC0_INTAFSR	NVIC0 補助フォールト・ステータス	0x00000000
0xE000ED40	NVIC0_INTPFR0	NVIC0 プロセッサ機能レジスタ 0	0x00000000
0xE000ED44	NVIC0_INTPFR1	NVIC0 プロセッサ機能レジスタ 1	0x00000000
0xE000ED48	NVIC0_INTDFR0	NVIC0 デバッグ機能レジスタ 0	0x00000000
0xE000ED4C	NVIC0_INTAFR0	NVIC0 補助機能レジスタ 0	0x00000000
0xE000ED50	NVIC0_INTMMFR0	NVIC0 メモリ･モデル機能レジスタ 0	0x00000000
0xE000ED54	NVIC0_INTMMFR1	NVIC0 メモリ･モデル機能レジスタ 1	0x00000000
0xE000ED58	NVIC0_INTMMFR2	NVIC0 メモリ･モデル機能レジスタ 2	0x00000000
0xE000ED5C	NVIC0_INTMMFR3	NVIC0 メモリ･モデル機能レジスタ 3	0x00000000
0xE000ED60	NVIC0_INTISAR0	NVIC0 ISA 機能レジスタ 0	0x00000000
0xE000ED64	NVIC0_INTISAR1	NVIC0 ISA 機能レジスタ 1	0x00000000
0xE000ED68	NVIC0_INTISAR2	NVIC0 ISA 機能レジスタ 2	0x00000000
0xE000ED6C	NVIC0_INTISAR3	NVIC0 ISA 機能レジスタ 3	0x00000000
0xE000ED70	NVIC0_INTISAR4	NVIC0 ISA 機能レジスタ 4	0x00000000
0xE000EF00	NVIC0_INTTRGI	NVIC0 ソフトウェア・トリガ割込みレジスタ	0x00000000
0xE000EFD0	NVIC0_INTPID4	NVIC0 ペリフェラル識別レジスタ 4	0x00000000
0xE000EFD4	NVIC0_INTPID5	NVIC0 ペリフェラル識別レジスタ 5	0x00000000

表 25-20：ADuCM4050 NVIC0 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0xE000EFD8	NVIC0_INTPID6	NVIC0 ペリフェラル識別レジスタ 6	0x00000000
0xE000EFDC	NVIC0_INTPID7	NVIC0 ペリフェラル識別レジスタ 7	0x00000000
0xE000EFE0	NVIC0_INTPID0	NVIC0 ペリフェラル識別 ビット 7：0	0x00000000
0xE000EFE4	NVIC0_INTPID1	NVIC0 ペリフェラル識別 ビット 15：8	0x00000000
0xE000EFE8	NVIC0_INTPID2	NVIC0 ペリフェラル識別 ビット 16：23	0x00000000
0xE000EFEC	NVIC0_INTPID3	NVIC0 ペリフェラル識別 ビット 24：31	0x00000000
0xE000EFF0	NVIC0_INTCID0	NVIC0 コンポーネント識別 ビット 7：0	0x00000000
0xE000EFF4	NVIC0_INTCID1	NVIC0 コンポーネント識別 ビット 15：8	0x00000000
0xE000EFF8	NVIC0_INTCID2	NVIC0 コンポーネント識別 ビット 16：23	0x00000000
0xE000EFFC	NVIC0_INTCID3	NVIC0 コンポーネント識別 ビット 24：31	0x00000000

表 25-21：ADuCM4050 PMG0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x4004C000	PMG0_IEN	PMG0 電源モニタ割込みイネーブル	0x00000000
0x4004C004	PMG0_PSM_STAT	PMG0 電源モニタのステータス	0x00000000
0x4004C008	PMG0_PWRMOD	PMG0 電力モードのレジスタ	0x00000000
0x4004C00C	PMG0_PWRKEY	PMG0 PMG_PWRMOD および PMG_SRAMRET のキー保護	0x00000000
0x4004C010	PMG0_SHDN_STAT	PMG0 シャットダウン・ステータス・レジスタ	0x00000000
0x4004C014	PMG0_SRAMRET	PMG0 休止モードでの保持 SRAM の制御	0xFF000000
0x4004C038	PMG0_TRIM	PMG0 トリミング・ビット	0x0A8AA2A2
0x4004C040	PMG0_RST_STAT	PMG0 リセット・ステータス	0x00000000
0x4004C044	PMG0_CTL1	PMG0 HPBUCK 制御	0x00A00000

表 25-22：ADuCM4050 PMG0_TST MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x4004C260	PMG0_TST_SRAM_CTL	PMG0 SRAM パリティおよび命令 SRAM の制御	0x80000000
0x4004C264	PMG0_TST_SRAM_INIT-STAT	PMG0 初期化ステータス・レジスタ	0x00000000

表 25-22：ADuCM4050 PMG0_TST MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x4004C268	PMG0_TST_CLR_LATCH_GPIOS	PMG0 シャットダウン・モード後の GPIO のクリア	0x00000000
0x4004C26C	PMG0_TST_SCRPAD_IMG	PMG0 スクラッチ・パッドのイメージ	0x00000000
0x4004C270	PMG0_TST_SCRPAD_3V_ RD	PMG0 バッテリ・ドメインに保存されたスクラッチ・パッド	0x00000000
0x4004C274	PMG0_TST_FAST_SHT_W AKEUP	PMG0 高速シャットダウン・ウェイクアップ・イネーブル	0x00000000

表 25-23：ADuCM4050 RNG0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40040400	RNG0_CTL	RNG0 RNG コントロール・レジスタ	0x00000000
0x40040404	RNG0_LEN	RNG0 RNG サンプル長レジスタ	0x00003400
0x40040408	RNG0_STAT	RNG0 RNG ステータス・レジスタ	0x00000000
0x4004040C	RNG0_DATA	RNG0 RNG データ・レジスタ	0x00000000
0x40040410	RNG0_OSCCNT	RNG0 発振器カウント	0x00000000
0x40040414	RNG0_OSCDIFF[n]	RNG0 発振器差	0x00000000
0x40040415	RNG0_OSCDIFF[n]	RNG0 発振器差	0x00000000
0x40040416	RNG0_OSCDIFF[n]	RNG0 発振器差	0x00000000
0x40040417	RNG0_OSCDIFF[n]	RNG0 発振器差	0x00000000

表 25-24：ADuCM4050 RTC0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40001000	RTC0_CR0	RTC0 RTC 制御 0	0x000003C4
0x40001004	RTC0_SR0	RTC0 RTC ステータス 0	0x00003F80
0x40001008	RTC0_SR1	RTC0 RTC ステータス 1	0x00000078
0x4000100C	RTC0_CNT0	RTC0 RTC カウント 0	0x00000000
0x40001010	RTC0_CNT1	RTC0 RTC カウント 1	0x00000000
0x40001014	RTC0_ALM0	RTC0 RTC アラーム 0	0x0000FFFF
0x40001018	RTC0_ALM1	RTC0 RTC アラーム 1	0x0000FFFF
0x4000101C	RTC0_TRM	RTC0 RTC トリム	0x00000398
0x40001020	RTC0_GWY	RTC0 RTC ゲートウェイ	0x00000000

表 25-24：ADuCM4050 RTC0 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40001028	RTC0_CR1	RTC0 RTC 制御 1	0x000001E0
0x4000102C	RTC0_SR2	RTC0 RTC ステータス 2	0x0000C000
0x40001030	RTC0_SNAP0	RTC0 RTC スナップショット 0	0x00000000
0x40001034	RTC0_SNAP1	RTC0 RTC スナップショット 1	0x00000000
0x40001038	RTC0_SNAP2	RTC0 RTC スナップショット 2	0x00000000
0x4000103C	RTC0_MOD	RTC0 RTC モジュロ	0x00000040
0x40001040	RTC0_CNT2	RTC0 RTC カウント 2	0x00000000
0x40001044	RTC0_ALM2	RTC0 RTC アラーム 2	0x00000000
0x40001048	RTC0_SR3	RTC0 RTC ステータス 3	0x00000000
0x4000104C	RTC0_CR2IC	RTC0 入力キャプチャ・チャンネル設定用の RTC 制御 2	0x000083A0
0x40001050	RTC0_CR3SS	RTC0 SensorStrobe チャンネル設定用の RTC 制御 3	0x00000000
0x40001054	RTC0_CR4SS	RTC0 SensorStrobe チャンネル設定用の RTC 制御 4	0x00000000
0x40001058	RTC0_SSMSK	RTC0 SensorStrobe チャンネル用 RTC マスク	0x00000000
0x40001064	RTC0_IC2	RTC0 RTC 入力キャプチャ・チャンネル 2	0x00000000
0x40001068	RTC0_IC3	RTC0 RTC 入力キャプチャ・チャンネル 3	0x00000000
0x4000106C	RTC0_IC4	RTC0 RTC 入力キャプチャ・チャンネル 4	0x00000000
0x40001070	RTC0_SS1	RTC0 RTC SensorStrobe チャンネル 1	0x00008000
0x40001074	RTC0_SS2	RTC0 RTC SensorStrobe チャンネル 2	0x00008000
0x40001078	RTC0_SS3	RTC0 RTC SensorStrobe チャンネル 3	0x00008000
0x4000107C	RTC0_SS4	RTC0 RTC SensorStrobe チャンネル 4	0x00008000
0x40001080	RTC0_SR4	RTC0 RTC ステータス 4	0x0000F7DF
0x40001084	RTC0_SR5	RTC0 RTC ステータス 5	0x00000000
0x40001088	RTC0_SR6	RTC0 RTC ステータス 6	0x00007900
0x4000108C	RTC0_SS1TGT	RTC0 RTC SensorStrobe チャンネル 1 のターゲット	0x00008000
0x40001090	RTC0_FRZCNT	RTC0 RTC フリーズ・カウント	0x00000000
0x40001094	RTC0_SS2TGT	RTC0 RTC SensorStrobe チャンネル 2 のターゲット	0x00008000
0x40001098	RTC0_SS3TGT	RTC0 RTC SensorStrobe チャンネル 3 のターゲット	0x00008000
0x400010A0	RTC0_SS1LOWDUR	RTC0 SensorStrobe チャンネル 1 の RTC 自動リロード・ロー時間	0x00008000

表 25-24：ADuCM4050 RTC0 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x400010A4	RTC0_SS2LOWDUR	RTC0 SensorStrobe チャンネル 2 の RTC 自動リロード・ロー時間	0x00008000
0x400010A8	RTC0_SS3LOWDUR	RTC0 SensorStrobe チャンネル 3 の RTC 自動リロード・ロー時間	0x00008000
0x400010B0	RTC0_SS1HIGHDUR	RTC0 SensorStrobe チャンネル 1 の RTC 自動リロード・ハイ時間	0x00008000
0x400010B4	RTC0_SS2HIGHDUR	RTC0 SensorStrobe チャンネル 2 の RTC 自動リロード・ハイ時間	0x00008000
0x400010B8	RTC0_SS3HIGHDUR	RTC0 SensorStrobe チャンネル 3 の RTC 自動リロード・ハイ時間	0x00008000
0x400010C0	RTC0_SSMSKOT	RTC0 SensorStrobe チャンネル・オン時間制御用 RTC マスク	0x00000000
0x400010C4	RTC0_CR5SSS	RTC0 SensorStrobe チャンネル GPIO サンプリング設定用の RTC 制御 5	0x00000000
0x400010C8	RTC0_CR6SSS	RTC0 SensorStrobe チャンネル GPIO サンプリング・エッジ設定用の RTC 制御 6	0x00000000
0x400010CC	RTC0_CR7SSS	RTC0 SensorStrobe チャンネル GPIO サンプリング・アクティビティ設定用の RTC 制御 7	0x00000000
0x400010D0	RTC0_SR7	RTC0 RTC ステータス 7	0x00000000
0x400010D4	RTC0_SR8	RTC0 RTC ステータス 8	0x00003F77
0x400010D8	RTC0_SR9	RTC0 RTC ステータス 9	0x00000000
0x400010E0	RTC0_GPMUX0	RTC0 RTC GPIO ピン・マルチプレクサ・コントロール・レジスタ 0	0x00004688
0x400010E4	RTC0_GPMUX1	RTC0 RTC GPIO ピン・マルチプレクサ・コントロール・レジスタ 1	0x000001F5

表 25-25：ADuCM4050 RTC1 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40001400	RTC1_CR0	RTC1 RTC 制御 0	0x000003C4
0x40001404	RTC1_SR0	RTC1 RTC ステータス 0	0x00003F80
0x40001408	RTC1_SR1	RTC1 RTC ステータス 1	0x00000078
0x4000140C	RTC1_CNT0	RTC1 RTC カウント 0	0x00000000
0x40001410	RTC1_CNT1	RTC1 RTC カウント 1	0x00000000
0x40001414	RTC1_ALM0	RTC1 RTC アラーム 0	0x0000FFFF
0x40001418	RTC1_ALM1	RTC1 RTC アラーム 1	0x0000FFFF

表 25-25：ADuCM4050 RTC1 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x4000141C	RTC1_TRM	RTC1 RTC トリム	0x00000398
0x40001420	RTC1_GWY	RTC1 RTC ゲートウェイ	0x00000000
0x40001428	RTC1_CR1	RTC1 RTC 制御 1	0x000001E0
0x4000142C	RTC1_SR2	RTC1 RTC ステータス 2	0x0000C000
0x40001430	RTC1_SNAP0	RTC1 RTC スナップショット 0	0x00000000
0x40001434	RTC1_SNAP1	RTC1 RTC スナップショット 1	0x00000000
0x40001438	RTC1_SNAP2	RTC1 RTC スナップショット 2	0x00000000
0x4000143C	RTC1_MOD	RTC1 RTC モジュロ	0x00000040
0x40001440	RTC1_CNT2	RTC1 RTC カウント 2	0x00000000
0x40001444	RTC1_ALM2	RTC1 RTC アラーム 2	0x00000000
0x40001448	RTC1_SR3	RTC1 RTC ステータス 3	0x00000000
0x4000144C	RTC1_CR2IC	RTC1 入力キャプチャ・チャンネル設定用の RTC 制御 2	0x000083A0
0x40001450	RTC1_CR3SS	RTC1 SensorStrobe チャンネル設定用の RTC 制御 3	0x00000000
0x40001454	RTC1_CR4SS	RTC1 SensorStrobe チャンネル設定用の RTC 制御 4	0x00000000
0x40001458	RTC1_SSMSK	RTC1 SensorStrobe チャンネル用 RTC マスク	0x00000000
0x40001464	RTC1_IC2	RTC1 RTC 入力キャプチャ・チャンネル 2	0x00000000
0x40001468	RTC1_IC3	RTC1 RTC 入力キャプチャ・チャンネル 3	0x00000000
0x4000146C	RTC1_IC4	RTC1 RTC 入力キャプチャ・チャンネル 4	0x00000000
0x40001470	RTC1_SS1	RTC1 RTC SensorStrobe チャンネル 1	0x00008000
0x40001474	RTC1_SS2	RTC1 RTC SensorStrobe チャンネル 2	0x00008000
0x40001478	RTC1_SS3	RTC1 RTC SensorStrobe チャンネル 3	0x00008000
0x4000147C	RTC1_SS4	RTC1 RTC SensorStrobe チャンネル 4	0x00008000
0x40001480	RTC1_SR4	RTC1 RTC ステータス 4	0x0000F7DF
0x40001484	RTC1_SR5	RTC1 RTC ステータス 5	0x00000000
0x40001488	RTC1_SR6	RTC1 RTC ステータス 6	0x00007900
0x4000148C	RTC1_SS1TGT	RTC1 RTC SensorStrobe チャンネル 1 のターゲット	0x00008000
0x40001490	RTC1_FRZCNT	RTC1 RTC フリーズ・カウント	0x00000000
0x40001494	RTC1_SS2TGT	RTC1 RTC SensorStrobe チャンネル 2 のターゲット	0x00008000
0x40001498	RTC1_SS3TGT	RTC1 RTC SensorStrobe チャンネル 3 のターゲット	0x00008000

表 25-25：ADuCM4050 RTC1 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x400014A0	RTC1_SS1LOWDUR	RTC1 SensorStrobe チャンネル 1 の RTC 自動リロード・ロー時間	0x00008000
0x400014A4	RTC1_SS2LOWDUR	RTC1 SensorStrobe チャンネル 2 の RTC 自動リロード・ロー時間	0x00008000
0x400014A8	RTC1_SS3LOWDUR	RTC1 SensorStrobe チャンネル 3 の RTC 自動リロード・ロー時間	0x00008000
0x400014B0	RTC1_SS1HIGHDUR	RTC1 SensorStrobe チャンネル 1 の RTC 自動リロード・ハイ時間	0x00008000
0x400014B4	RTC1_SS2HIGHDUR	RTC1 SensorStrobe チャンネル 2 の RTC 自動リロード・ハイ時間	0x00008000
0x400014B8	RTC1_SS3HIGHDUR	RTC1 SensorStrobe チャンネル 3 の RTC 自動リロード・ハイ時間	0x00008000
0x400014C0	RTC1_SSMSKOT	RTC1 SensorStrobe チャンネル・オン時間制御用 RTC マスク	0x00000000
0x400014C4	RTC1_CR5SSS	RTC1 SensorStrobe チャンネル GPIO サンプリング設定用の RTC 制御 5	0x00000000
0x400014C8	RTC1_CR6SSS	RTC1 SensorStrobe チャンネル GPIO サンプリング・エッジ設定用の RTC 制御 6	0x00000000
0x400014CC	RTC1_CR7SSS	RTC1 SensorStrobe チャンネル GPIO サンプリング・アクティビティ設定用の RTC 制御 7	0x00000000
0x400014D0	RTC1_SR7	RTC1 RTC ステータス 7	0x00000000
0x400014D4	RTC1_SR8	RTC1 RTC ステータス 8	0x00003F77
0x400014D8	RTC1_SR9	RTC1 RTC ステータス 9	0x00000000
0x400014E0	RTC1_GPMUX0	RTC1 RTC GPIO ピン・マルチプレクサ・コントロール・レジスタ 0	0x00004688
0x400014E4	RTC1_GPMUX1	RTC1 RTC GPIO ピン・マルチプレクサ・コントロール・レジスタ 1	0x000001F5

表 25-26：ADuCM4050 SPI0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40004000	SPI0_STAT	SPI0 ステータス	0x00000800
0x40004004	SPI0_RX	SPI0 受信	0x00000000
0x40004008	SPI0_TX	SPI0 送信	0x00000000
0x4000400C	SPI0_DIV	SPI0 SPI ボー・レート選択	0x00000000
0x40004010	SPI0_CTL	SPI0 SPI 設定	0x00000000
0x40004014	SPI0_IEN	SPI0 SPI 割込みイネーブル	0x00000000

表 25-26：ADuCM4050 SPI0 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40004018	SPI0_CNT	SPI0 転送バイト数	0x00000000
0x4000401C	SPI0_DMA	SPI0 SPI DMA イネーブル	0x00000000
0x40004020	SPI0_FIFO_STAT	SPI0 FIFO ステータス	0x00000000
0x40004024	SPI0_RD_CTL	SPI0 読出し制御	0x00000000
0x40004028	SPI0_FLOW_CTL	SPI0 フロー制御	0x00000000
0x4000402C	SPI0_WAIT_TMR	SPI0 フロー制御の待機タイマー	0x00000000
0x40004030	SPI0_CS_CTL	SPI0 マルチスレーブ接続のチップ・セレクト制御	0x00000001
0x40004034	SPI0_CS_OVERRIDE	SPI0 チップ・セレクト・オーバーライド	0x00000000

表 25-27：ADuCM4050 SPI1 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40004400	SPI1_STAT	SPI1 ステータス	0x00000800
0x40004404	SPI1_RX	SPI1 受信	0x00000000
0x40004408	SPI1_TX	SPI1 送信	0x00000000
0x4000440C	SPI1_DIV	SPI1 SPI ボー・レート選択	0x00000000
0x40004410	SPI1_CTL	SPI1 SPI 設定	0x00000000
0x40004414	SPI1_IEN	SPI1 SPI 割込みイネーブル	0x00000000
0x40004418	SPI1_CNT	SPI1 転送バイト数	0x00000000
0x4000441C	SPI1_DMA	SPI1 SPI DMA イネーブル	0x00000000
0x40004420	SPI1_FIFO_STAT	SPI1 FIFO ステータス	0x00000000
0x40004424	SPI1_RD_CTL	SPI1 読出し制御	0x00000000
0x40004428	SPI1_FLOW_CTL	SPI1 フロー制御	0x00000000
0x4000442C	SPI1_WAIT_TMR	SPI1 フロー制御の待機タイマー	0x00000000
0x40004430	SPI1_CS_CTL	SPI1 マルチスレーブ接続のチップ・セレクト制御	0x00000001
0x40004434	SPI1_CS_OVERRIDE	SPI1 チップ・セレクト・オーバーライド	0x00000000

表 25-28：ADuCM4050 SPI2 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40024000	SPI2_STAT	SPI2 ステータス	0x00000800
0x40024004	SPI2_RX	SPI2 受信	0x00000000

表 25-28：ADuCM4050 SPI2 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40024008	SPI2_TX	SPI2 送信	0x00000000
0x4002400C	SPI2_DIV	SPI2 SPI ボー・レート選択	0x00000000
0x40024010	SPI2_CTL	SPI2 SPI 設定	0x00000000
0x40024014	SPI2_IEN	SPI2 SPI 割込みイネーブル	0x00000000
0x40024018	SPI2_CNT	SPI2 転送バイト数	0x00000000
0x4002401C	SPI2_DMA	SPI2 SPI DMA イネーブル	0x00000000
0x40024020	SPI2_FIFO_STAT	SPI2 FIFO ステータス	0x00000000
0x40024024	SPI2_RD_CTL	SPI2 読出し制御	0x00000000
0x40024028	SPI2_FLOW_CTL	SPI2 フロー制御	0x00000000
0x4002402C	SPI2_WAIT_TMR	SPI2 フロー制御の待機タイマー	0x00000000
0x40024030	SPI2_CS_CTL	SPI2 マルチスレーブ接続のチップ・セレクト制御	0x00000001
0x40024034	SPI2_CS_OVERRIDE	SPI2 チップ・セレクト・オーバーライド	0x00000000

表 25-29：ADuCM4050 SPORT0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40038000	SPORT0_CTL_A	SPORT0 ハーフ SPORT'A'コントロール・レジスタ	0x00000000
0x40038004	SPORT0_DIV_A	SPORT0 ハーフ SPORT 'A'除数レジスタ	0x00000000
0x40038008	SPORT0_IEN_A	SPORT0 ハーフ SPORT A 割込みイネーブル・レジスタ	0x00000000
0x4003800C	SPORT0_STAT_A	SPORT0 ハーフ SPORT 'A'ステータス・レジスタ	0x00000000
0x40038010	SPORT0_NUMTRAN_A	SPORT0 ハーフ SPORT A 転送数レジスタ	0x00000000
0x40038014	SPORT0_CNVT_A	SPORT0 ハーフ SPORT 'A' CNV 幅レジスタ	0x00000000
0x40038020	SPORT0_TX_A	SPORT0 ハーフ SPORT 'A' Tx バッファ・レジスタ	0x00000000
0x40038028	SPORT0_RX_A	SPORT0 ハーフ SPORT 'A' Rx バッファ・レジスタ	0x00000000
0x40038040	SPORT0_CTL_B	SPORT0 ハーフ SPORT 'B'コントロール・レジスタ	0x00000000
0x40038044	SPORT0_DIV_B	SPORT0 ハーフ SPORT 'B'除数レジスタ	0x00000000
0x40038048	SPORT0_IEN_B	SPORT0 ハーフ SPORT B 割込みイネーブル・レジスタ	0x00000000
0x4003804C	SPORT0_STAT_B	SPORT0 ハーフ SPORT 'B'ステータス・レジスタ	0x00000000
0x40038050	SPORT0_NUMTRAN_B	SPORT0 ハーフ SPORT B 転送数レジスタ	0x00000000
0x40038054	SPORT0_CNVT_B	SPORT0 ハーフ SPORT 'B' CNV 幅レジスタ	0x00000000
0x40038060	SPORT0_TX_B	SPORT0 ハーフ SPORT 'B' Tx バッファ・レジスタ	0x00000000

表 25-29：ADuCM4050 SPORT0 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40038068	SPORT0_RX_B	SPORT0 ハーフ SPORT 'B' Rx バッファ・レジスタ	0x00000000

表 25-30：ADuCM4050 SYS MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40002020	SYS_ADIID	ADI 識別	0x00004144
0x40002024	SYS_CHIPID	チップ識別子	0x000002A1
0x40002040	SYS_SWDEN	シリアル・ワイヤ・デバッグ・イネーブル	0x00007072

表 25-31：ADuCM4050 TMR_RGB MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40000C00	TMR_RGB_LOAD	TMR_RGB 16 ビット・ロード値	0x00000000
0x40000C04	TMR_RGB_CURCNT	TMR_RGB 16 ビット・タイマー値	0x00000000
0x40000C08	TMR_RGB_CTL	TMR_RGB 制御	0x0000000A
0x40000C0C	TMR_RGB_CLRINT	TMR_RGB 割込みクリア	0x00000000
0x40000C10	TMR_RGB_CAPTURE	TMR_RGB キャプチャ	0x00000000
0x40000C14	TMR_RGB_ALOAD	TMR_RGB 16 ビット・ロード値、非同期	0x00000000
0x40000C18	TMR_RGB_ACURCNT	TMR_RGB 16 ビット・タイマー値、非同期	0x00000000
0x40000C1C	TMR_RGB_STAT	TMR_RGB ステータス	0x00000000
0x40000C20	TMR_RGB_PWM0CTL	TMR_RGB PWM0 制御レジスタ	0x00000000
0x40000C24	TMR_RGB_PWM0MATCH	TMR_RGB PWM0 マッチ値	0x00000000
0x40000C28	TMR_RGB_EVENTSE- LECT	TMR_RGB タイマー・イベント選択レジスタ	0x00000000
0x40000C2C	TMR_RGB_PWM1CTL	TMR_RGB PWM1 制御レジスタ	0x00000000
0x40000C30	TMR_RGB_PWM1MATCH	TMR_RGB PWM1 マッチ値	0x00000000
0x40000C34	TMR_RGB_PWM2CTL	TMR_RGB PWM2 制御レジスタ	0x00000000
0x40000C38	TMR_RGB_PWM2MATCH	TMR_RGB PWM2 マッチ値	0x00000000

表 25-32：ADuCM4050 UART0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40005000	UART0_TX	UART0 送信保持レジスタ	0x00000000

表 25-32：ADuCM4050 UART0 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40005000	UART0_RX	UART0 受信バッファ・レジスタ	0x00000000
0x40005004	UART0_IEN	UART0 割込みイネーブル	0x00000000
0x40005008	UART0_IIR	UART0 割込み ID	0x00000001
0x4000500C	UART0_LCR	UART0 ライン制御	0x00000000
0x40005014	UART0_LSR	UART0 ライン・ステータス	0x00000060
0x4000501C	UART0_SCR	UART0 スクラッチ･バッファ	0x00000000
0x40005020	UART0_FCR	UART0 FIFO 制御	0x00000000
0x40005024	UART0_FBR	UART0 分数ボー・レート	0x00000000
0x40005028	UART0_DIV	UART0 ボー・レート分周器	0x00000000
0x4000502C	UART0_LCR2	UART0 第 2 のライン制御	0x00000002
0x40005030	UART0_CTL	UART0 UART コントロール・レジスタ	0x00000100
0x40005034	UART0_RFC	UART0 RX FIFO のバイト・カウント	0x00000000
0x40005038	UART0_TFC	UART0 TX FIFO のバイト・カウント	0x00000000
0x4000503C	UART0_RSC	UART0 RS485 の半二重制御	0x00000000
0x40005040	UART0_ACR	UART0 自動ボー・レート制御	0x00000000
0x40005044	UART0_ASRL	UART0 自動ボー・レート・ステータス（ロー）	0x00000000
0x40005048	UART0_ASRH	UART0 自動ボー・レート・ステータス（ハイ）	0x00000000

表 25-33：ADuCM4050 UART1 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40005400	UART1_TX	UART1 送信保持レジスタ	0x00000000
0x40005400	UART1_RX	UART1 受信バッファ・レジスタ	0x00000000
0x40005404	UART1_IEN	UART1 割込みイネーブル	0x00000000
0x40005408	UART1_IIR	UART1 割込み ID	0x00000001
0x4000540C	UART1_LCR	UART1 ライン制御	0x00000000
0x40005414	UART1_LSR	UART1 ライン・ステータス	0x00000060
0x4000541C	UART1_SCR	UART1 スクラッチ･バッファ	0x00000000
0x40005420	UART1_FCR	UART1 FIFO 制御	0x00000000
0x40005424	UART1_FBR	UART1 分数ボー・レート	0x00000000
0x40005428	UART1_DIV	UART1 ボー・レート分周器	0x00000000

表 25-33：ADuCM4050 UART1 MMR レジスタのアドレス（続き）

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x4000542C	UART1_LCR2	UART1 第 2 のライン制御	0x00000002
0x40005430	UART1_CTL	UART1 UART コントロール・レジスタ	0x00000100
0x40005434	UART1_RFC	UART1 RX FIFO のバイト・カウント	0x00000000
0x40005438	UART1_TFC	UART1 TX FIFO のバイト・カウント	0x00000000
0x4000543C	UART1_RSC	UART1 RS485 の半二重制御	0x00000000
0x40005440	UART1_ACR	UART1 自動ボー・レート制御	0x00000000
0x40005444	UART1_ASRL	UART1 自動ボー・レート・ステータス（ロー）	0x00000000
0x40005448	UART1_ASRH	UART1 自動ボー・レート・ステータス（ハイ）	0x00000000

表 25-34：ADuCM4050 WDT0 MMR レジスタのアドレス

メモリマップド・アドレス	レジスタ名	説明	リセット値
0x40002C00	WDT0_LOAD	WDT0 ロード値	0x00001000
0x40002C04	WDT0_CCNT	WDT0 現在のカウント値	0x00001000
0x40002C08	WDT0_CTL	WDT0 制御	0x000000E9
0x40002C0C	WDT0_RESTART	WDT0 割込みクリア	0x00000000
0x40002C18	WDT0_STAT	WDT0 ステータス	0x00000000

索引

C

D

S

T

U

W

X

あ

お

か

き

く

け

こ

し

す

せ

そ

た

ち

て

と

に

の

は

ひ

ふ

へ

ほ

ま

ゆ

よ

ら

り

ろ

わ