

# Cortex<sup>™</sup>-M3

リビジョン : r1p1

## テクニカルリファレンスマニュアル



# Cortex-M3

## テクニカルリファレンス マニュアル

Copyright © 2005, 2006 ARM Limited. All rights reserved.  
リリース情報

### 改訂履歴

日付	発行	公開の有無	改訂内容
2005 年 11 月 15 日	A	非公開	初版
2006 年 1 月 13 日	B	公開	非公開から公開に変更
2006 年 5 月 10 日	C	公開	rlp0 の最初のリリース
2006 年 9 月 27 日	D	公開	rlp1 の最初のリリース
2007 年 6 月 13 日	E	公開	技術面での変更を伴わない細部の更新

### 著作権表記

® または ™ の付いた用語とロゴは、本著作権条項で特に明記されていない限り、EU および他諸国における ARM Limited の登録商標または商標です。本書に記載されている他の商標その他の名前は、対応する所有者の商標の場合があります。

本書に記載されている情報の全部または一部、ならびに本書で紹介する製品は、著作権所有者の文書による事前の許可を得ない限り、転用・複製することを禁じます。

本書に説明されている製品は、継続的に開発と改良が行われています。本書に含まれている製品およびその利用方法についての情報は、ARM Limited が利用者の利益のために提供するものです。したがって当社では、製品の商品性または目的への適合性を含め、暗黙的・明示的に関係なく一切の保証を行いません。

本書は、本製品の利用者をサポートすることだけを目的としています。本書に記載されている情報の使用、情報の誤りまたは省略、あるいは本製品の誤使用によって発生したいかなる損失や損害についても、ARM Limited は一切責任を負いません。

### 守秘義務

このドキュメントは公開文書です。本書を使用、複製、公開する権利には、ARM と ARM が本書を配布した当事者との間で締結された契約の条項に従って、ライセンス制約が適用されることがあります。

### 製品ステータス

本書には最終情報（完成製品に関する情報）が記載されています。

### ARM ホームページ

<http://www.arm.com>

# 目次

## Cortex-M3 テクニカルリファレンス マニュアル

### 序章

本書について .....	xviii
ご意見・ご質問 .....	xxiii

### 第 1 章

#### はじめに

1.1	プロセッサについて .....	1-2
1.2	コンポーネント、階層、実装 .....	1-4
1.3	構成可能なオプション .....	1-12
1.4	パイプラインステージの実行 .....	1-13
1.5	プリフェッチユニット .....	1-15
1.6	分岐ターゲットのフォワーディング .....	1-16
1.7	ストアバッファ .....	1-19
1.8	命令セットの概要 .....	1-20
1.9	製品リビジョン .....	1-31

### 第 2 章

#### プログラマモデル

2.1	プログラマモデル .....	2-2
2.2	特権アクセスおよびユーザアクセス .....	2-3
2.3	レジスタ .....	2-4
2.4	データタイプ .....	2-10
2.5	メモリフォーマット .....	2-11
2.6	命令セット .....	2-13

<b>第 3 章</b>	<b>システム制御</b>	
	3.1 プロセッサレジスタの概要 .....	3-2
<b>第 4 章</b>	<b>メモリマップ</b>	
	4.1 メモリマップ .....	4-2
	4.2 ビットバンド .....	4-5
	4.3 ROM メモリテーブル .....	4-7
<b>第 5 章</b>	<b>例外</b>	
	5.1 例外モデルについて .....	5-2
	5.2 例外のタイプ .....	5-4
	5.3 例外の優先度 .....	5-6
	5.4 特権とスタック .....	5-9
	5.5 横取り .....	5-11
	5.6 テールチェイン .....	5-14
	5.7 後着 .....	5-15
	5.8 退出 .....	5-17
	5.9 リセット .....	5-20
	5.10 例外制御の移行 .....	5-24
	5.11 複数のスタックの設定 .....	5-25
	5.12 アボートモデル .....	5-27
	5.13 起動レベル .....	5-32
	5.14 フローチャート .....	5-34
<b>第 6 章</b>	<b>クロックとリセット</b>	
	6.1 クロック .....	6-2
	6.2 リセット .....	6-4
	6.3 Cortex-M3 のリセットモード .....	6-5
<b>第 7 章</b>	<b>電力管理</b>	
	7.1 電力管理について .....	7-2
	7.2 システム電力管理 .....	7-3
<b>第 8 章</b>	<b>ネスト型ベクタ割り込みコントローラ</b>	
	8.1 NVIC について .....	8-2
	8.2 NVIC のプログラマモデル .....	8-3
	8.3 レベル割り込みとパルス割り込みの比較 .....	8-41
<b>第 9 章</b>	<b>メモリ保護ユニット</b>	
	9.1 MPU について .....	9-2
	9.2 MPU のプログラマモデル .....	9-3
	9.3 MPU のアクセス許可 .....	9-13
	9.4 MPU アボート .....	9-15
	9.5 MPU 領域の更新 .....	9-16
	9.6 割り込みと MPU の更新 .....	9-19

<b>第 10 章</b>	<b>コアデバッグ</b>	
10.1	コアデバッグについて .....	10-2
10.2	コアデバッグ レジスタ .....	10-3
10.3	コア デバッグアクセスの例 .....	10-12
10.4	コアデバッグでのアプリケーションレジスタの使用法 .....	10-13
<b>第 11 章</b>	<b>システムデバッグ</b>	
11.1	システムデバッグについて .....	11-2
11.2	システムデバッグ アクセス .....	11-3
11.3	システムデバッグのプログラマモデル .....	11-5
11.4	FPB .....	11-6
11.5	DWT .....	11-13
11.6	ITM .....	11-29
11.7	AHB-AP .....	11-38
<b>第 12 章</b>	<b>デバッグポート</b>	
12.1	DP について .....	12-2
<b>第 13 章</b>	<b>トレースポート インタフェースユニット</b>	
13.1	TPIU について .....	13-2
13.2	TPIU レジスタ .....	13-8
13.3	シリアルワイヤ出力接続 .....	13-17
<b>第 14 章</b>	<b>バスインタフェース</b>	
14.1	バスインタフェースについて .....	14-2
14.2	AMBA 3 への準拠 .....	14-3
14.3	ICode バスインタフェース .....	14-4
14.4	DCode バスインタフェース .....	14-6
14.5	システムインタフェース .....	14-7
14.6	コードバスの統合 .....	14-9
14.7	外部専用ペリフェラル インタフェース .....	14-10
14.8	アクセスのアライメント .....	14-11
14.9	領域にまたがるアンアラインドアクセス .....	14-12
14.10	ビットバンドアクセス .....	14-13
14.11	ライトバッファ .....	14-14
14.12	メモリ属性 .....	14-15
14.13	AHB のタイミング特性 .....	14-16
<b>第 15 章</b>	<b>エンベデッドトレース マクロセル</b>	
15.1	ETM について .....	15-2
15.2	データトレース .....	15-7
15.3	ETM リソース .....	15-8
15.4	トレース出力 .....	15-11
15.5	ETM のアーキテクチャ .....	15-12
15.6	ETM のプログラマモデル .....	15-16

<b>第 16 章</b>	<b>エンベデッドトレース マクロセルのインタフェース</b>	
16.1	ETM インタフェースについて .....	16-2
16.2	CPU の ETM インタフェースポートの説明 .....	16-3
16.3	分岐ステータスインタフェース .....	16-6
<b>第 17 章</b>	<b>AHB トレースマクロセル インタフェース</b>	
17.1	AHB トレースマクロセル インタフェースについて .....	17-2
17.2	CPU AHB トレースマクロセル インタフェースのポートの説明 .....	17-3
<b>第 18 章</b>	<b>命令のタイミング</b>	
18.1	命令のタイミングについて .....	18-2
18.2	プロセッサ命令のタイミング .....	18-3
18.3	ロード / ストアのタイミング .....	18-7
<b>第 19 章</b>	<b>AC 特性</b>	
19.1	プロセッサのタイミングパラメータ .....	19-2
19.2	プロセッサのタイミングパラメータ .....	19-3
<b>付録 A</b>	<b>信号の説明</b>	
A.1	クロック .....	A-2
A.2	リセット .....	A-3
A.3	その他 .....	A-4
A.4	割り込みインタフェース .....	A-6
A.5	ICode インタフェース .....	A-7
A.6	DCode インタフェース .....	A-8
A.7	システムバス インタフェース .....	A-9
A.8	専用ペリフェラルバス インタフェース .....	A-10
A.9	ITM インタフェース .....	A-11
A.10	AHB-AP インタフェース .....	A-12
A.11	ETM インタフェース .....	A-13
A.12	AHB トレースマクロセルインタフェース .....	A-15
A.13	テストインタフェース .....	A-16
	<b>用語集</b>	

# 表目次

## Cortex-M3 テクニカルリファレンス マニュアル

	改訂履歴 .....	ii
表 1-1	Cortex-M3 の 16 ビット命令の概要 .....	1-20
表 1-2	Cortex-M3 の 32 ビット命令の概要 .....	1-23
表 2-1	アプリケーションプログラム ステータスレジスタのビット割り当て .....	2-6
表 2-2	割り込みプログラム ステータスレジスタのビット割り当て .....	2-7
表 2-3	EPSR のビットの機能 .....	2-8
表 2-4	サポートされていない Thumb 命令 .....	2-13
表 2-5	サポートされている Thumb-2 命令 .....	2-13
表 3-1	NVIC レジスタ .....	3-2
表 3-2	コアデバッグ レジスタ .....	3-5
表 3-3	フラッシュパッチ レジスタの概要 .....	3-6
表 3-4	DWT レジスタの概要 .....	3-7
表 3-5	ITM レジスタの概要 .....	3-9
表 3-6	AHB-AP レジスタの概要 .....	3-10
表 3-7	デバッグインタフェースポート レジスタの概要 .....	3-11
表 3-8	MPU レジスタ .....	3-11
表 3-9	TPIU レジスタ .....	3-12
表 3-10	ETM レジスタ .....	3-13
表 4-1	メモリアンタフェース .....	4-3
表 4-2	メモリ領域のアクセス許可 .....	4-4
表 4-3	ROM テーブル .....	4-7
表 5-1	例外のタイプ .....	5-4
表 5-2	優先度に応じた例外の動作 .....	5-6

表 5-3	優先度のグループ化 .....	5-8
表 5-4	例外の開始手順 .....	5-12
表 5-5	例外の退出手順 .....	5-17
表 5-6	例外からの復帰動作 .....	5-19
表 5-7	リセット時の動作 .....	5-20
表 5-8	リセット時のブートアップ動作 .....	5-21
表 5-9	例外処理への移行 .....	5-24
表 5-10	フォールト .....	5-28
表 5-11	デバッグフォールト .....	5-30
表 5-12	フォールトステータス レジスタとフォールトアドレス レジスタ .....	5-31
表 5-13	各起動レベルでの特権とスタック .....	5-32
表 5-14	例外の遷移 .....	5-32
表 5-15	例外のサブタイプの遷移 .....	5-33
表 6-1	Cortex-M3 プロセッサのクロック .....	6-2
表 6-2	Cortex-M3 マクロセルのクロック .....	6-2
表 6-3	リセット入力 .....	6-4
表 6-4	リセットモード .....	6-5
表 7-1	サポートされているスリープモード .....	7-3
表 8-1	NVIC レジスタ .....	8-3
表 8-2	割り込みコントローラタイプ レジスタのビット割り当て .....	8-8
表 8-3	SysTick 制御およびステータスレジスタのビット割り当て .....	8-9
表 8-4	SysTick リロード値レジスタのビット割り当て .....	8-10
表 8-5	SysTick 現在値レジスタのビット割り当て .....	8-11
表 8-6	SysTick 較正值レジスタのビット割り当て .....	8-11
表 8-7	割り込みイネーブルセットレジスタのビット割り当て .....	8-13
表 8-8	割り込みイネーブルクリアレジスタのビット割り当て .....	8-13
表 8-9	割り込み保留セットレジスタのビット割り当て .....	8-14
表 8-10	割り込み保留クリアレジスタのビット割り当て .....	8-15
表 8-11	アクティブビット レジスタのビット割り当て .....	8-15
表 8-12	割り込み優先度レジスタの 0 ~ 31 ビットの割り当て .....	8-17
表 8-13	CPUID ベースレジスタのビット割り当て .....	8-17
表 8-14	割り込み制御状態レジスタのビット割り当て .....	8-19
表 8-15	ベクタテーブルオフセットレジスタのビット割り当て .....	8-21
表 8-16	アプリケーション割り込みおよびリセット制御レジスタのビット割り当て .....	8-22
表 8-17	システム制御レジスタのビット割り当て .....	8-24
表 8-18	構成制御レジスタのビット割り当て .....	8-26
表 8-19	システムハンドラ優先度レジスタのビット割り当て .....	8-28
表 8-20	システムハンドラ制御および状態レジスタのビット割り当て .....	8-29
表 8-21	メモリ管理フォールトステータス レジスタのビット割り当て .....	8-32
表 8-22	バスフォールトステータス レジスタのビット割り当て .....	8-33
表 8-23	用法フォールトステータス レジスタのビット割り当て .....	8-35
表 8-24	ハードフォールトステータス レジスタのビット割り当て .....	8-36
表 8-25	デバッグフォールトステータス レジスタのビット割り当て .....	8-38
表 8-26	メモリ管理フォールトアドレス レジスタのビット割り当て .....	8-39
表 8-27	バスフォールトアドレス レジスタのビット割り当て .....	8-39
表 8-28	補助フォールトステータス レジスタのビット割り当て .....	8-40
表 8-29	ソフトウェアトリガ割り込みレジスタのビット割り当て .....	8-40



表 9-1	MPU レジスタ .....	9-3
表 9-2	MPU タイプレジスタのビット割り当て .....	9-4
表 9-3	MPU 制御レジスタのビット割り当て .....	9-6
表 9-4	MPU 領域番号レジスタのビット割り当て .....	9-7
表 9-5	MPU 領域ベースアドレス レジスタのビット割り当て .....	9-8
表 9-6	MPU 領域属性およびサイズレジスタのビット割り当て .....	9-9
表 9-7	MPU 保護領域サイズフィールド .....	9-10
表 9-8	TEX、C、B のエンコード .....	9-13
表 9-9	メモリ属性エンコードのキャッシュポリシー .....	9-14
表 9-10	AP のエンコード .....	9-14
表 9-11	XN のエンコード .....	9-14
表 10-1	コアデバッグ レジスタ .....	10-2
表 10-2	デバッグホールド制御およびステータスレジスタ .....	10-4
表 10-3	デバッグコアレジスタ セレクタレジスタ .....	10-6
表 10-4	デバッグ例外およびモニタ制御レジスタ .....	10-9
表 10-5	コアデバッグで使用されるアプリケーションレジスタ .....	10-13
表 11-1	FPB レジスタの概要 .....	11-7
表 11-2	フラッシュパッチ制御レジスタのビット割り当て .....	11-8
表 11-3	COMP マッピング .....	11-10
表 11-4	フラッシュパッチリマップレジスタのビット割り当て .....	11-11
表 11-5	フラッシュパッチ コンパレータレジスタのビット割り当て .....	11-12
表 11-6	DWT レジスタの概要 .....	11-13
表 11-7	DWT 制御レジスタのビット割り当て .....	11-16
表 11-8	DWT カレント PC サンプラサイクル カウントレジスタのビット割り当て .....	11-19
表 11-9	DWT CPI カウントレジスタのビット割り当て .....	11-20
表 11-10	DWT 例外オーバーヘッドカウント レジスタのビット割り当て .....	11-20
表 11-11	DWT スリープカウント レジスタのビット割り当て .....	11-21
表 11-12	DWT LSU カウントレジスタのビット割り当て .....	11-22
表 11-13	DWT フォールドカウント レジスタのビット割り当て .....	11-23
表 11-14	DWT プログラムカウンタ サンプルレジスタのビット割り当て .....	11-23
表 11-15	DWT コンパレータレジスタ 0～3 のビット割り当て .....	11-24
表 11-16	DWT マスクレジスタ 0～3 のビット割り当て .....	11-24
表 11-17	DWT 機能レジスタ 0～3 のビット機能 .....	11-25
表 11-18	DWT 機能レジスタの設定 .....	11-27
表 11-19	ITM レジスタの概要 .....	11-29
表 11-20	ITM トレースイネーブルレジスタのビット割り当て .....	11-31
表 11-21	ITM トレース特権レジスタのビット割り当て .....	11-32
表 11-22	ITM トレース制御レジスタのビット割り当て .....	11-33
表 11-23	ITM 統合書き込みレジスタのビット割り当て .....	11-35
表 11-24	ITM 統合読み出しレジスタのビット割り当て .....	11-35
表 11-25	ITM 統合モード制御レジスタのビット割り当て .....	11-36
表 11-26	ITM ロックアクセス レジスタのビット割り当て .....	11-36
表 11-27	ITM ロックステータス レジスタのビット割り当て .....	11-37
表 11-28	AHB-AP レジスタの概要 .....	11-38
表 11-29	AHB-AP 制御およびステータスワード レジスタのビット割り当て .....	11-39
表 11-30	AHB-AP 転送アドレスレジスタのビット割り当て .....	11-41
表 11-31	AHB-AP データ読み出し / 書き込みレジスタのビット割り当て .....	11-41

表 11-32	AHB-AP バンクデータ レジスタのビット割り当て .....	11-42
表 11-33	AHB-AP デバッグ ROM アドレスレジスタのビット割り当て .....	11-42
表 11-34	AHB-AP ID レジスタのビット割り当て .....	11-43
表 13-1	トレース出力ポートの信号 .....	13-5
表 13-2	ATB ポートの信号 .....	13-6
表 13-3	その他のコンフィギュレーション入力 .....	13-7
表 13-4	TPIU レジスタ .....	13-8
表 13-5	非同期クロックプリスケアラ レジスタのビット割り当て .....	13-10
表 13-6	選択ピンプロトコル レジスタのビット割り当て .....	13-11
表 13-7	フォーマッタおよびフラッシュステータス レジスタのビット割り当て .....	13-12
表 13-8	フォーマッタおよびフラッシュ制御レジスタのビット割り当て .....	13-13
表 13-9	統合テストレジスタ -ITATBCTR2 のビット割り当て .....	13-15
表 13-10	統合テストレジスタ -ITATBCTR0 のビット割り当て .....	13-16
表 14-1	命令フェッチ .....	14-4
表 14-2	バスマッパのアンアラインドアクセス .....	14-11
表 14-3	メモリ属性 .....	14-15
表 14-4	インタフェースのタイミング特性 .....	14-16
表 15-1	ETM コアインタフェースの入力と出力 .....	15-4
表 15-2	その他の構成入力 .....	15-4
表 15-3	トレースポート信号 .....	15-5
表 15-4	その他の信号 .....	15-5
表 15-5	クロックとリセット .....	15-6
表 15-6	APB インタフェース信号 .....	15-6
表 15-7	Cortex-M3 リソース .....	15-8
表 15-8	例外トレースのマッピング .....	15-13
表 15-9	ETM レジスタ .....	15-16
表 16-1	ETM インタフェースポート .....	16-3
表 16-2	分岐ステータス信号の機能 .....	16-6
表 16-3	オペコードシーケンスの例 .....	16-10
表 17-1	AHB インタフェースのポート .....	17-3
表 18-1	命令のタイミング .....	18-3
表 19-1	その他の入力ポートのタイミングパラメータ .....	19-3
表 19-2	割り込み入力ポートのタイミングパラメータ .....	19-3
表 19-3	AHB 入力ポートのタイミングパラメータ .....	19-4
表 19-4	PPB 入力ポートのタイミングパラメータ .....	19-4
表 19-5	デバッグ入力ポートのタイミングパラメータ .....	19-5
表 19-6	テスト入力ポートのタイミングパラメータ .....	19-5
表 19-7	ETM 入力ポートのタイミングパラメータ .....	19-6
表 19-8	その他の出力ポートのタイミングパラメータ .....	19-6
表 19-9	AHB 出力ポートのタイミングパラメータ .....	19-6
表 19-10	PPB 出力ポートのタイミングパラメータ .....	19-8
表 19-11	デバッグインタフェース出力ポートのタイミングパラメータ .....	19-8
表 19-12	ETM インタフェース出力ポートのタイミングパラメータ .....	19-9
表 19-13	HTM インタフェース出力ポートのタイミングパラメータ .....	19-10
表 19-14	テスト出力ポートのタイミングパラメータ .....	19-10
表 A-1	クロック信号 .....	A-2
表 A-2	リセット信号 .....	A-3

表 A-3	その他の信号 .....	A-4
表 A-4	割り込みインタフェース .....	A-6
表 A-5	ICode インタフェース .....	A-7
表 A-6	DCode インタフェース .....	A-8
表 A-7	システムバス インタフェース .....	A-9
表 A-8	専用ペリフェラルバス インタフェース .....	A-10
表 A-9	ITM インタフェース .....	A-11
表 A-10	AHB-AP インタフェース .....	A-12
表 A-11	ETM インタフェース .....	A-13
表 A-12	HTM インタフェース .....	A-15
表 A-13	テストインタフェース .....	A-16



## 図目次

# Cortex-M3 テクニカルリファレンス マニュアル

	タイミング図の規則を表す記号 .....	xxi
図 1-1	Cortex-M3 のブロック図 .....	1-5
図 1-2	Cortex-M3 のパイプラインステージ .....	1-13
図 2-1	プロセッサのレジスタセット .....	2-4
図 2-2	アプリケーションプログラム ステータスレジスタのビット割り当て .....	2-6
図 2-3	割り込みプログラム ステータスレジスタのビット割り当て .....	2-6
図 2-4	実行プログラム ステータスレジスタ .....	2-8
図 2-5	リトルエンディアンおよびビッグエンディアンのメモリフォーマット .....	2-12
図 4-1	プロセッサのメモリマップ .....	4-2
図 4-2	ビットバンドのマッピング .....	4-6
図 5-1	横取り後のスタックの内容 .....	5-11
図 5-2	例外開始のタイミング .....	5-13
図 5-3	テールチェーンのタイミング .....	5-14
図 5-4	後着例外のタイミング .....	5-15
図 5-5	例外退出のタイミング .....	5-18
図 5-6	割り込み処理のフローチャート .....	5-34
図 5-7	横取りのフローチャート .....	5-35
図 5-8	割り込みからの復帰のフローチャート .....	5-36
図 6-1	リセット信号 .....	6-6
図 6-2	パワーオンリセット .....	6-6
図 6-3	内部リセットの同期 .....	6-7
図 7-1	SLEEPING 電力制御の例 .....	7-4
図 7-2	SLEEPDEEP 電力制御の例 .....	7-5

図 8-1	割り込みコントローラタイプ レジスタのビット割り当て .....	8-7
図 8-2	SysTick 制御およびステータスレジスタのビット割り当て .....	8-8
図 8-3	SysTick リロード値レジスタのビット割り当て .....	8-10
図 8-4	SysTick 現在値レジスタのビット割り当て .....	8-10
図 8-5	SysTick 較正值レジスタのビット割り当て .....	8-11
図 8-6	割り込み優先度レジスタの 0 ~ 31 ビットの割り当て .....	8-16
図 8-7	CPUID ベースレジスタのビット割り当て .....	8-17
図 8-8	割り込み制御状態レジスタのビット割り当て .....	8-19
図 8-9	ベクタテーブルオフセットレジスタのビット割り当て .....	8-21
図 8-10	アプリケーション割り込みおよびリセット制御レジスタのビット割り当て .....	8-22
図 8-11	システム制御レジスタのビット割り当て .....	8-24
図 8-12	構成制御レジスタのビット割り当て .....	8-26
図 8-13	システムハンドラ優先度レジスタのビット割り当て .....	8-28
図 8-14	システムハンドラ制御および状態レジスタのビット割り当て .....	8-29
図 8-15	構成可能フォールトステータス レジスタのビット割り当て .....	8-31
図 8-16	メモリ管理フォールトレジスタのビット割り当て .....	8-32
図 8-17	バスフォールトステータス レジスタのビット割り当て .....	8-33
図 8-18	用法フォールトステータス レジスタのビット割り当て .....	8-35
図 8-19	ハードフォールトステータス レジスタのビット割り当て .....	8-36
図 8-20	デバッグフォールトステータス レジスタのビット割り当て .....	8-37
図 8-21	ソフトウェアトリガ割り込みレジスタのビット割り当て .....	8-40
図 9-1	MPU タイプレジスタのビット割り当て .....	9-4
図 9-2	MPU 制御レジスタのビット割り当て .....	9-5
図 9-3	MPU 領域番号レジスタのビット割り当て .....	9-7
図 9-4	MPU 領域ベースアドレス レジスタのビット割り当て .....	9-8
図 9-5	MPU 領域属性およびサイズレジスタのビット割り当て .....	9-9
図 10-1	デバッグホールト制御およびステータスレジスタのフォーマット .....	10-4
図 10-2	デバッグコアレジスタ セレクタレジスタのフォーマット .....	10-6
図 10-3	デバッグ例外およびモニタ制御レジスタのフォーマット .....	10-8
図 11-1	システムデバッグ アクセスのブロック図 .....	11-4
図 11-2	フラッシュパッチ制御レジスタのビット割り当て .....	11-8
図 11-3	フラッシュパッチリマップレジスタのビット割り当て .....	11-10
図 11-4	フラッシュパッチ コンパレータレジスタのビット割り当て .....	11-11
図 11-5	DWT 制御レジスタのビット割り当て .....	11-15
図 11-6	DWT CPI カウントレジスタのビット割り当て .....	11-19
図 11-7	DWT 例外オーバーヘッドカウンタ レジスタのビット割り当て .....	11-20
図 11-8	DWT スリープカウンタ レジスタのビット割り当て .....	11-21
図 11-9	DWT LSU カウントレジスタのビット割り当て .....	11-22
図 11-10	DWT フォールドカウンタ レジスタのビット割り当て .....	11-22
図 11-11	DWT マスクレジスタ 0 ~ 3 のビット割り当て .....	11-24
図 11-12	DWT 機能レジスタ 0 ~ 3 のビット割り当て .....	11-25
図 11-13	ITM トレース特権レジスタのビット割り当て .....	11-32
図 11-14	ITM トレース制御レジスタのビット割り当て .....	11-33
図 11-15	ITM 統合書き込みレジスタのビット割り当て .....	11-34
図 11-16	ITM 統合読み出しレジスタのビット割り当て .....	11-35
図 11-17	ITM 統合モード制御レジスタのビット割り当て .....	11-36
図 11-18	ITM ロックステータス レジスタのビット割り当て .....	11-37

図 11-19	AHB-AP 制御およびステータスワード レジスタ .....	11-39
図 11-20	AHB-AP ID レジスタ .....	11-43
図 13-1	TPIU のブロック図 (非 ETM バージョン) .....	13-3
図 13-2	TPIU のブロック図 (ETM バージョン) .....	13-4
図 13-3	サポートされる同期化ポートサイズ レジスタのビット割り当て .....	13-9
図 13-4	非同期クロックプリスケラ レジスタのビット割り当て .....	13-10
図 13-5	選択ピンプロトコル レジスタのビット割り当て .....	13-10
図 13-6	フォーマッタおよびフラッシュステータス レジスタのビット割り当て .....	13-12
図 13-7	フォーマッタおよびフラッシュ制御レジスタのビット割り当て .....	13-13
図 13-8	統合テストレジスタ -ITATBCTR2 のビット割り当て .....	13-15
図 13-9	統合テストレジスタ -ITATBCTR0 のビット割り当て .....	13-16
図 13-10	TRACESWO 専用ピン .....	13-17
図 13-11	TRACEPORT と SWO の共用 .....	13-18
図 13-12	JTAG-TDO と SWO の共用 .....	13-18
図 14-1	ICode/Dcode マルチプレクサ .....	14-9
図 15-1	ETM のブロック図 .....	15-3
図 15-2	例外からの復帰を示すパケットのエンコード .....	15-12
図 15-3	分岐パケットの例外のエンコード .....	15-14
図 16-1	条件付き後方分岐が行われなかった場合 .....	16-7
図 16-2	条件付き後方分岐が行われた場合 .....	16-7
図 16-3	条件付き前方分岐が行われなかった場合 .....	16-8
図 16-4	条件付き前方分岐が行われた場合 .....	16-8
図 16-5	パイプラインがストールしない場合の無条件分岐 .....	16-8
図 16-6	パイプラインがストールした場合の無条件分岐 .....	16-9
図 16-7	実行段での無条件分岐、飛び先がアラインドな命令 .....	16-9
図 16-8	実行段での無条件分岐、飛び先がアンアラインドな命令 .....	16-9
図 16-9	オペコードシーケンスの例 .....	16-11





# 序章

本章では、*Cortex-M3* テクニカルリファレンス マニュアル (TRM) について説明します。  
本章は以下のセクションから構成されています。

- 本書について : P.xviii
- ご意見・ご質問 : P.xxiii

## 本書について

本書は、Cortex-M3 プロセッサ用の TRM です。

## 製品リビジョン ステータス

**mpn** 識別子は、本書に記載されている製品のリビジョン ステータスを示しています。各識別子の意味は次のとおりです。

**rn** 製品が大幅に修正されたことを示しています。

**pn** 製品に小さな修正または変更が加えられていることを示しています。

## 対象読者

本書は、Cortex-M3 プロセッサをベースとしたシステムオンチップ(SoC) デバイスの実装を計画しているシステム設計者、システムインテグレータ、および検証技術者を対象としています。

## 本書の使用法

本書は以下の章に分かれています。

### 第1章 「はじめに」

プロセッサのコンポーネントおよび命令セットについて説明します。

### 第2章 「プログラマモデル」

プロセッサのレジスタセットや動作モード、そしてプロセッサのプログラミングに関するその他の情報について説明します。

### 第3章 「システム制御」

システム制御のために、レジスタやプログラマモデルについて説明します。

### 第4章 「メモリマップ」

プロセッサのメモリマップおよびビットバンディング機能について説明します。

### 第5章 「例外」

プロセッサの例外モデルについて説明します。

### 第6章 「クロックとリセット」

プロセッサのクロックおよびリセットについて説明します。

### 第7章 「電力管理」

プロセッサの電力管理および省電力について説明します。

### 第8章 「ネスト型ベクタ割り込みコントローラ」

プロセッサの割り込み処理およびその制御について説明します。

## 第9章 「メモリ保護ユニット」

プロセッサのメモリ保護ユニット (MPU) について説明します。

## 第10章 「コアデバッグ」

プロセッサコアのデバッグおよびテストについて説明します。

## 第11章 「システムデバッグ」

プロセッサのシステムデバッグ コンポーネントについて説明します。

## 第12章 「デバッグポート」

プロセッサのデバッグポート、シリアルワイヤ *JTAG* デバッグポート (SWJ-DP)、シリアルワイヤデバッグポート(SW-DP)について説明します。

## 第13章 「トレースポート インタフェースユニット」

プロセッサのトレースポート インタフェースユニット (TPIU) について説明します。

## 第14章 「バスインタフェース」

プロセッサのバスインタフェースについて説明します。

## 第15章 「エンベデッドトレース マクロセル」

プロセッサのエンベデッドトレース マクロセル(ETM)について説明します。

## 第16章 「エンベデッドトレース マクロセルのインタフェース」

プロセッサの ETM インタフェースについて説明します。

## 第17章 「AHB トレースマクロセルインタフェース」

プロセッサのアドバンスド ハイパフォーマンス バス (AHB) トレースマクロセルインタフェースについて説明します。

## 第18章 「命令のタイミング」

プロセッサの命令のタイミングおよびクロックサイクルについて説明します。

## 第19章 「AC 特性」

プロセッサの AC 特性について説明します。

## 付録 A 「信号の説明」

プロセッサの信号の一覧についてはこの付録をご覧ください。

## 表記規則

本書では次の表記規則が採用されています。

- 書体の一般的な規則
- タイミング図: P.xx

- 信号: P.xxi
- 序章: P.xvii

## 書体の一般的な規則

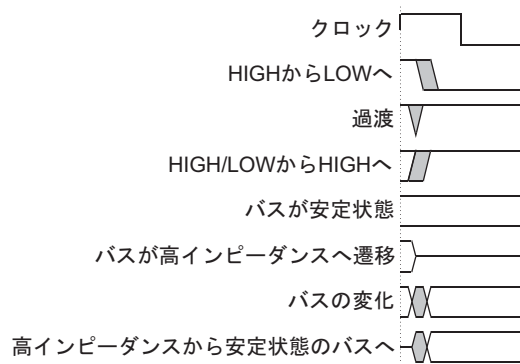
本書で使用されている書体の一般的な規則は次のとおりです。

<b>斜体</b>	重要な注釈の強調、特別な用語の初出時、本書内での相互参照と引用に使用されます。
<b>太字</b>	メニュー名などのインタフェース要素を強調するために太字が使用されます。信号名を示すためにも使用されています。また、必要に応じて説明表の項目名にも太字が使用されています。
<b>monospace</b>	コマンド、ファイル名、プログラム名、ソースコードなどの、キーボードから入力可能なテキストを示しています。
<b><u>monospace</u></b>	コマンドまたはオプションに使用可能な略語を示しています。コマンドやオプションの名前を全部入力する代わりに、下線部分のテキストだけを入力してこれらを指定できます。
<b><i>monospace italic</i></b>	特定の値で置き換え可能な引数を示しています。
<b>monospace bold</b>	サンプルコード以外で使用されている場合、言語のキーワードを示しています。
<b>&lt; and &gt;</b>	コードまたはコード片の中で不等号の括弧で囲まれた言葉は、アセンブラ構文内で置き換え可能なことを示しています。文章の中では、通常のフォントで記載されています。いくつかの例を挙げます。 <ul style="list-style-type: none"> <li>• MRC p15, 0 &lt;Rd&gt;, &lt;CRn&gt;, &lt;CRm&gt;, &lt;Opcode_2&gt;</li> <li>• Opcode_2 の値によって、アクセスされるレジスタが選択されます。</li> </ul>

## タイミング図

タイミング図の規則を表す記号は、タイミング図で使用される構成要素を示しています。この図と異なる意味で使用されている場合は、その都度明記されています。タイミング図に明示されていないタイミング情報については、推測で判断しないで下さい。

影が付いたバスと信号の部分は定義されていないため、バスと信号は、その時点で影付きの領域内の任意の値を取り得ます。実際のレベルは重要ではなく、通常の動作には影響しません。



タイミング図の規則を表す記号

### 信号

信号の表記規則は次のとおりです。

信号レベル	アサートされる信号レベルは、その信号がアクティブ <b>HIGH</b> かアクティブ <b>LOW</b> かに依存します。「アサートされた (asserted)」とは、アクティブ <b>HIGH</b> の信号に対しては <b>HIGH</b> 、アクティブ <b>LOW</b> の信号に対しては <b>LOW</b> ということを意味します。
小文字の n	アクティブ <b>LOW</b> の信号を表します。
接頭文字 H	アドバンスド ハイパフォーマンスバス (AHB) 信号を表します。
接頭文字 P	アドバンスド ペリフェラルバス (APB) 信号を表します。

### 数値の表記

数値の表記規則は次のとおりです。

< ビット数 > < 基数 > < 数値 >

これは、Verilog 方式による定数の省略表記です。いくつかの例を挙げます。

- 'h7B4 は、ビット数が決められていない 16 進数の値です。
- 'o7654 は、ビット数が決められていない 8 進数の値です。
- 8'd9 は、8 ビット幅の 10 進数の値で 9 です。
- 8'h3F は、8 ビット幅の 16 進数の値で 0x3F です。これは、b00111111 に等価です。
- 8'b1111 は、8 ビット幅の 2 進数の値で b00001111 です。

## 参考資料

このセクションでは、ARM Limited やサードパーティが発行している出版物を紹介します。

ARM Limited は自社出版物の定期的な更新と修正を行っています。最新の正誤表、追補情報、および ARM Limited によく寄せられる質問 (FAQ) については、<http://www.arm.com> を参照して下さい。

### ARM の刊行物

本書は、Cortex-M3 プロセッサ固有の情報を提供しています。他の関連情報については、以下の出版物を参照して下さい。

- *ARM Architecture Reference Manual* (ARM DDI 0100)
- *ARM Architecture Reference Manual, Thumb-2® Supplement* (ARM DDI 0308)
- *ARMv7-M Architecture Reference Manual* (ARM DDI 0403)
- *ARM AMBA® 3 AHB-Lite Protocol (v1.0)* (ARM IHI 0033)
- *ARM CoreSight™ Components Technical Reference Manual* (ARM DDI 0314)
- *ARM Debug Interface v5, Architecture Specification* (ARM IHI 0031)
- *ARM Embedded Trace Macrocell Architecture Specification* (ARM IHI 0014).

### その他の刊行物

このセクションでは、サードパーティが発行している関連する出版物を紹介します。

- IEEE Standard, *Test Access Port and Boundary-Scan Architecture specification* 1149.1-1990 (JTAG).

## ご意見・ご質問

ARM Limited では、Cortex-M3 プロセッサと本書に関するご意見等をお待ちしています。

### Cortex-M3 プロセッサに関するご意見・ご質問

本製品に関するご意見・ご質問等がございましたら、次の情報とともに製品購入元までご連絡下さい。

- 製品名
- ご意見についての簡潔な説明

### 本書に関するご意見

本書に関するご意見等がございましたら、電子メールに次の情報をご記入の上、[errata@arm.com](mailto:errata@arm.com) までお寄せ下さい。

- 題名
- 資料番号
- ご意見のあるページ番号
- ご意見についての簡潔な説明

補足または改善すべき点についての一般的なご意見もお待ちしております。





# 第 1 章

## はじめに

本章では、プロセッサおよび命令セットについて説明します。本章は以下のセクションから構成されています。

- プロセッサについて : P.1-2
- コンポーネント、階層、実装 : P.1-4
- 構成可能なオプション : P.1-12
- パイプラインステージの実行 : P.1-13
- プリフェッチユニット : P.1-15
- 分岐ターゲットのフォワーディング : P.1-16
- ストアバッファ : P.1-19
- 命令セットの概要 : P.1-20
- 製品リビジョン : P.1-31

## 1.1 プロセッサについて

このプロセッサは、少ないゲート数、短い割り込みレイテンシ、および低コストのデバッグを特徴とする低電力プロセッサです。高速な割り込み応答機能を要求する組み込み用途に向いています。プロセッサには、ARM v7-M アーキテクチャが実装されています。

このプロセッサには次の機能が組み込まれています。

- プロセッサコア。ゲート数が少ないコアであり、短いレイテンシで割り込みを処理します。プロセッサコアには次の特徴があります。
  - ARMv7-M。Thumb®-2 命令セットアーキテクチャ(ISA) のサブセットで、16 ビットおよび 32 ビットの基本 Thumb-2 命令がすべて含まれますが、メディア、単一命令複数データ処理(SIMD)、拡張デジタル信号プロセッサ(DSP) 命令(Eバリエーション)、ARM システムアクセスの部分は取り除かれています。
  - レジスタのうち、スタックポインタ(SP)のみがバンク切替え
  - ハードウェア除算命令の SDIV および UDIV(Thumb-2 命令)
  - ハンドラモードとスレッドモード
  - Thumb 状態とデバッグ状態
  - 短い割り込みレイテンシのために、中断可能で中断後から継続可能な複数転送(LDM/STM)、PUSH/POP
  - 短いレイテンシでの割り込み処理ルーチン(ISR)(Interrupt Service Routine)への入退出のために、自動的にプロセッサ状態を保存/復元
  - エンディアンは、ARMv6 アーキテクチャ方式の BE8/LE をサポート
  - メモリアccessは、ARMv6 のアンアラインドアクセスをサポート
- ネスト型ベクタ割り込みコントローラ(NVIC)。NVIC が密接にプロセッサコアに統合されているので、短いレイテンシの割り込み処理を実現しています。これには、次の機能が含まれています。
  - 外部割り込みは、1 ~ 240 の間でサイズを構成可能
  - 優先度は、3 ~ 8 ビットの間でサイズを構成可能
  - 割り込みの優先度を動的に再設定
  - 優先度のグループ化。これにより、横取りする割り込みレベルと横取りしない割り込みのレベルを選択できるようになります。
  - 割り込みで、テールチェイン(Tail-Chaining)と後着(Late Arrival)をサポート。これにより、割り込みと割り込みの間における状態の保存と復元のオーバーヘッドなしで、連続して割り込み処理が可能になります。
  - 自動的にプロセッサ状態が割り込みエントリ時に保存され、割り込み復帰時に復元され、命令のオーバーヘッドがありません。

- メモリ保護ユニット(MPU)。メモリ保護に使用されるオプションの MPU で、次の特徴があります。
  - 8つのメモリ領域
  - サブ領域無効化 (SRD)(Sub Region Disable) によるメモリ領域の効率的な利用
  - バックグラウンド領域の許可により、デフォルトでのメモリマップ属性を与えることが可能
- バスインタフェース
  - アドバンスド ハイパフォーマンズバス ライト (AHB-Lite)。ICode バス、Dcode バスおよびシステムバス インタフェース
  - アドバンスド ペリフェラルバス (APB) および専用ペリフェラルバス (PPB)(Private Peripheral Bus) インタフェース
  - 不可分なビットバンド書き込みと読み出しを備えたビットバンドのサポート
  - メモリアクセスのアライメント
  - 書き込みデータをバッファするためのライトバッファ
- 低コストのデバッグソリューション。次の機能が含まれています。
  - デバッグアクセスは、システムのすべてのメモリとレジスタへアクセス可能であり、コアの実行中、ホールド中、またはリセット中の Cortex-M3 のレジスタ バンクも含みます。
  - シリアルワイヤ デバッグポート (SW-DP)(Serial Wire Debug Port) および、シリアルワイヤ JTAG デバッグポート (SWJ-DP)(Serial Wire JTAG Debug Port) の、一方または両方によるデバッグアクセス
  - フラッシュパッチおよびブレイクポイント (FPB)(Flash Patch and Breakpoint) ユニットにより、ブレイクポイントおよびコードへのパッチを実現
  - データウォッチポイントおよびトレース (DWT)(Data Watchpoint and Trace) ユニットにより、ウォッチポイント、データトレース、およびシステムプロファイリングを実現
  - 計装トレース マクロセル(ITM)(Instrumentation Trace Macrocell)により、printf 方式のデバッグをサポート
  - トレースポート インタフェースユニット (TPIU) が、トレースポート アナライザ(TPA)(Trace Port Analyzer) へのブリッジとなります。
  - オプションのエンベデッドトレース マクロセル(ETM)による命令トレース

## 1.2 コンポーネント、階層、実装

このセクションでは、プロセッサのコンポーネント、階層、実装について説明します。  
このセクションは以下の項目から構成されています。

- プロセッサコア : P.1-7
- *NVIC* : P.1-8
- バスマトリックス : P.1-8
- *FPB* : P.1-9
- *DWT* : P.1-9
- *ITM* : P.1-10
- *MPU* : P.1-10
- *ETM* : P.1-10
- *TPIU* : P.1-10

プロセッサの構造を、P.1-5 図 1-1 に示します。

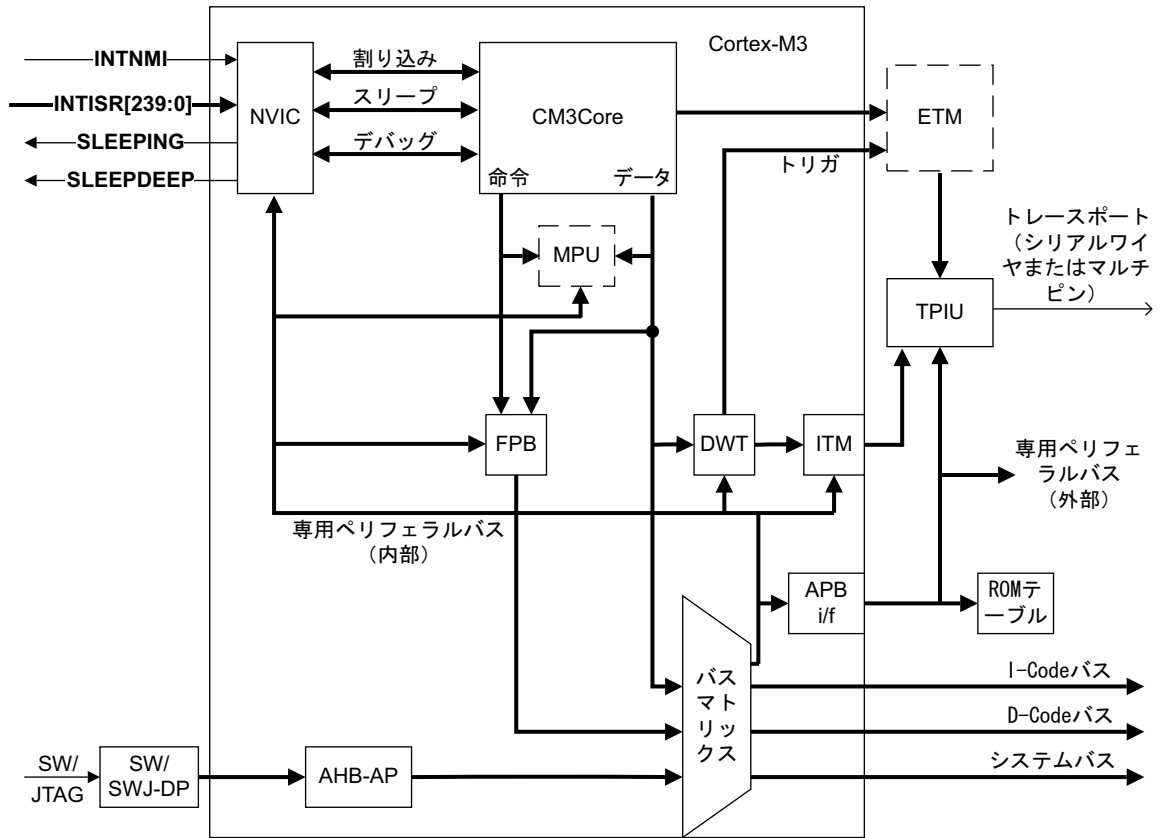


図 1-1 Cortex-M3 のブロック図

—— 注 ——

EMT および MPU はオプションのコンポーネントなので、実装によっては存在しないこともあります。

### 1.2.1 Cortex-M3 の階層と実装オプション

図 1-1 は、プロセッサのコンポーネントが2つのレベルの階層に存在することを示しています。これは設計の RTL 階層を表します。4 つのコンポーネント、ETM、TPIU、SW/SWJ-DP、ROM テーブルは、Cortex-M3 レベルの外側に存在しています。これは、これらのコンポーネントがオプションであるか、あるいは柔軟な実装や使用が可能であるためです。実装によっては図 1-1 に示された内容とは異なることがあります。実装可能なオプションに関しては以下に示されています。

- TPIU : P.1-6

- *SW/SWJ-DP* : P.1-6
- *ROM* テーブル

## TPIU

TPIU の実装オプションは次のとおりです。

- ETM がシステムに存在する場合、TPIU への二つの入力ポートはともに存在します。それ以外の場合、片方のポートのみが使用され、1 つの入力用 FIFO のゲートコストが節約できます。
- 単一または複数の TPIU により、マルチコアの実装をトレースすることができます。
- ARM TPIU ブロックは、パートナー独自の CoreSight™ に準拠した TPIU と置き換えることができます。
- 製品版のデバイスでは、TPIU が取り除かれていることがあります。

### —— 注 ——

TPIU が取り除かれている場合、Cortex-M3 のトレース機能は使用できません。

## SW/SWJ-DP

SW/SWJ-DP の実装オプションを以下に示します。

- 実装は SW-DP または SWJ-DP のどちらかが含まれます。
- ARM SW-DP は、パートナー独自の CoreSight に準拠した SW-DP と置き換えることができます。
- ARM SWJ-DP は、パートナー独自の CoreSight に準拠した SWJ-DP と置き換えることができます。
- パートナー独自のテストインタフェースを、SW-DP または SWJ-DP と同時に使用することができます。

## ROM テーブル

ROM テーブルは次の場合に、P.4-7「ROM メモリテーブル」に記載されたものから変更されます。

- システムにデバッグコンポーネントが追加された場合

## 1.2.2 プロセッサコア

プロセッサコアには、ARM v7-M アーキテクチャが実装されています。主な機能を次に示します。

- Thumb-2 (ISA) のサブセットであり、16 ビットおよび 32 ビットの基本 Thumb-2 命令がすべて含まれています。
- ハーバードプロセッサアーキテクチャにより、命令フェッチと、データのロード / ストアが同時に実行できます。
- 3 ステージのパイプライン
- 単一サイクルでの 32 ビット乗算
- ハードウェア除算
- Thumb 状態とデバッグ状態
- ハンドラモードとスレッドモード
- 短いレイテンシの ISR 入退出
  - プロセッサ状態の保存と復元は、命令フェッチのオーバーヘッドがありません。例外ベクタは、状態保存と同時にメモリからフェッチされるため、ISR エントリが高速になります。
  - 後着割り込みのサポート
  - 割り込みコントローラと密接に結合されたインタフェースにより、後着割り込みの効率的な処理が可能
  - 割り込みのテールチェインにより、割り込み間での状態の保存と復元のオーバーヘッドなしで、連続して割り込み処理が可能になります。
- 中断可能で中断後から継続可能な複数転送 (LDM/STM)、PUSH/POP
- エンディアンは、ARMv6 方式の BE8/LE をサポート
- メモリアクセスは、ARMv6 のアンアラインドをサポート

### レジスタ

プロセッサには次のレジスタがあります。

- 13 個の 32 ビット汎用レジスタ
- リンクレジスタ (LR)
- プログラムカウンタ (PC)
- プログラムステータス レジスタ、xPSR
- 2 つのバンクを持つ SP レジスタ

## メモリアインタフェース

このプロセッサはハーバードインタフェースを採用しているため、命令フェッチと、データのロード/ストアが同時に可能です。メモリアクセスは次のユニットにより制御されます。

- 独立したロードストアユニット (LSU)(Load Store Unit) があり、ロードおよびストア動作が ALU (Arithmetic and Logic Unit) から分離されています
- 3 エントリのプリフェッチユニット (PFU)(Prefetch Unit)。1 回につき 1 ワードがフェッチされます。この 1 ワードというのは、Thumb 命令が 2 つか、ワードアラインの Thumb-2 命令が 1 つか、ハーフワードアラインの Thumb-2 命令の上位か下位半分のハーフワードが 1 つに加えて Thumb 命令 1 つか、あるいは（加えるのが Thumb 命令ではなくて）ハーフワードにアラインされた別の Thumb-2 命令の下位か上位半分のハーフワードが 1 つかのいずれかです。コアからのすべてのフェッチアドレスはワードアラインです。Thumb-2 命令がハーフワードアラインの場合、その Thumb-2 命令をフェッチするには 2 回のフェッチが必要になります。しかし、3 エントリのプリフェッチバッファにより、最初にフェッチする Thumb-2 命令のハーフワードのみにストールサイクルが必要なことが保証されます。

### 1.2.3 NVIC

NVIC はプロセッサ コアに密接に結合されています。これにより、低レイテンシの例外処理を容易に行うことができます。主な機能を次に示します。

- 1 ~ 240 の間で外部割り込みの数を構成可能
- 3 ~ 8 ビットの間で優先度のビット数を構成可能
- レベルによる割り込みとパルスによる割り込みをサポート
- 割り込みの動的な優先度再設定
- 優先度のグループ化
- 割り込みのテールチェインをサポート
- プロセッサ状態は割り込み開始時に自動的に保存され、割り込みからの復帰時には自動的に復元され、命令のオーバーヘッドがありません。

NVIC については第 8 章「ネスト型ベクタ割り込みコントローラ」で詳しく説明します。

### 1.2.4 バスマトリックス

バスマトリックスは、プロセッサおよびデバッグインタフェースを外部バスに接続します。バスマトリックスは以下の外部バスへのインタフェースになります。

- ICode バス。このバスはコード空間からの、命令およびベクタのフェッチに使用されます。これは、32 ビット AHB-Lite バスです。
- DCode バス。このバスはコード空間への、データのロード/ストアおよびデバッグアクセスに使用されます。これは、32 ビット AHB-Lite バスです。



- システムバス。このバスはシステム空間への、命令およびベクタのフェッチ、データのロード / ストア、デバッグアクセスに使用されます。これは、32 ビット AHB-Lite バスです。
- PPB。このバスは PPB 空間への、データのロード / ストアおよびデバッグアクセスに使用されます。これは、32 ビット APB (v2.0) バスです。

バスマトリックスには次の制御機能も含まれています。

- アンアラインドアクセス。バスマトリックスはアラインされていないプロセッサアクセスを、アラインドアクセスに変換します。
- ビットバンディング。バスマトリックスは、ビットバンドエイリアスへのアクセスを、ビットバンド領域へのアクセスに変換します。この際、次の処理が行われます。
  - ビットバンドロード用のビットフィールド抽出
  - ビットバンドストアに対する不可分な読み出し - 変更 - 書き込み (atomic read-modify-write)
- ライトバッファ。バスマトリックスには、バスのストールをプロセッサコアから切り離すため、1 エントリの書き込みバッファが含まれています。

バスインタフェースについては、第 14 章 「バスインタフェース」で説明します。

### 1.2.5 FPB

FPB ユニットはハードウェアブレイクポイントを実装し、コード空間からシステム空間へ、アクセスにパッチをあてます。FPB には次の 8 つのコンパレータがあります。

- 6 つの命令コンパレータを個別に構成して、命令フェッチをコード空間からシステム空間にマップし直したり、ハードウェアブレイクポイントを実行したりすることができます。
- 2 つのリテラルコンパレータで、リテラルアクセスをコード空間からシステム空間へマップし直すことができます。

FPB については、第 11 章 「システムデバッグ」で説明します。

### 1.2.6 DWT

DWT ユニットには、次のデバッグ機能が含まれています。

- 4 つのコンパレータがあり、ハードウェアウォッチポイント、ETM トリガ、PC サンプラのイベントトリガ、またはデータアドレス サンプラのイベントトリガとして構成することができます。
- 性能測定のため、複数のカウンタやデーター一致イベントトリガが含まれています。
- 定義された間隔で PC サンプルを送信し、割り込みイベント情報を送信するように構成可能です。

DWT については、第 11 章 「システムデバッグ」で説明します。

### 1.2.7 ITM

ITM は、アプリケーションのイベントトレースや `printf` 方式のデバッグをサポートする、アプリケーション駆動型のトレースソースです。

ITM では、次のトレース情報ソースが提供されます。

- ソフトウェアトレース。ソフトウェアは ITM のスティムラスレジスタに直接書き込むことができます。これにより、パケットが送信されます。
- ハードウェアトレース。パケットは DWT により生成され、ITM により送信されます。
- タイムスタンプ。タイムスタンプはパケットを基準として送信されます。

ITM については、第 11 章 「システムデバッグ」で説明します。

### 1.2.8 MPU

オプションの MPU を利用すると、プロセッサがメモリ保護を行えるようになります。MPU はアクセス許可およびメモリ属性をチェックします。8 つの領域が含まれおり、これに加えてバックグラウンド領域を設定すると、デフォルトのメモリマップ属性を設定できます。

MPU については、第 9 章 「メモリ保護ユニット」で説明します。

### 1.2.9 ETM

ETM は、命令トレースのみをサポートする低コストのトレースマクロセルです。

ETM については、第 15 章 「エンベデッドトレース マクロセル」で説明します。

### 1.2.10 TPIU

TPIU は、ITM（と、存在する場合は ETM）からの Cortex-M3 トレースデータと、チップ外のトレースポートアナライザとの間でブリッジとして動作します。TPIU を構成して、低コストのデバッグ用にシリアルピンのトレース、または、より広帯域のトレース用にマルチピンのトレースをサポートすることができます。TPIU は CoreSight 互換です。

TPIU については、第 13 章 「トレースポートインタフェースユニット」で説明します。

### 1.2.11 SW/SWJ-DP

プロセッサを構成して、**SW-DP** または **SWJ-DP** のデバッグポート インタフェースをプロセッサに持たせることができます。デバッグポートは、プロセッサのレジスタを含む、システム中のすべてのレジスタおよびメモリに対するデバッグアクセスを提供します。

SW/SWJ-DP については、第 12 章 「デバッグポート」 で説明します。

## 1.3 構成可能なオプション

このセクションでは、プロセッサの構成オプションを紹介します。お使いになる実装の構成については、実装者に確認して下さい。

### 1.3.1 割り込み

実装時に、外部割り込みの本数を 1 ～ 240 の間で構成することができます。また、割り込み優先度のビット数を 3 ～ 8 ビットの間で構成することができます。

### 1.3.2 MPU

実装に MPU を含めるように構成することができます。

MPU については、第 9 章 「メモリ保護ユニット」で説明します。

### 1.3.3 DWT

DWT 実装にデーター一致を含めて構成することも、除外することもできます。

DWT については、P.11-13 「DWT」で説明します。

### 1.3.4 ETM

実装時に、ETM を含めるようにシステムを構成することができます。

ETM については、第 15 章 「エンベデッドトレース マクロセル」で説明します。

### 1.3.5 AHB トレースマクロセル インタフェース

実装時に、AHB トレース マクロセル (HTM) (AHB Trace Macrocell) インタフェースを含めるように Coretex-M3 システムを構成することができます。実装時にこのオプションを有効にしないと、必要な回路が含まれないため、HTM インタフェースは動作しません。

## 1.4 パイプラインステージの実行

パイプラインは次のステージにより構成されます。

- フェッチステージ
- デコードステージ
- 実行ステージ

プロセッサのパイプラインステージ、および各ステージで実行されるパイプライン処理について、図 1-2 に示します。

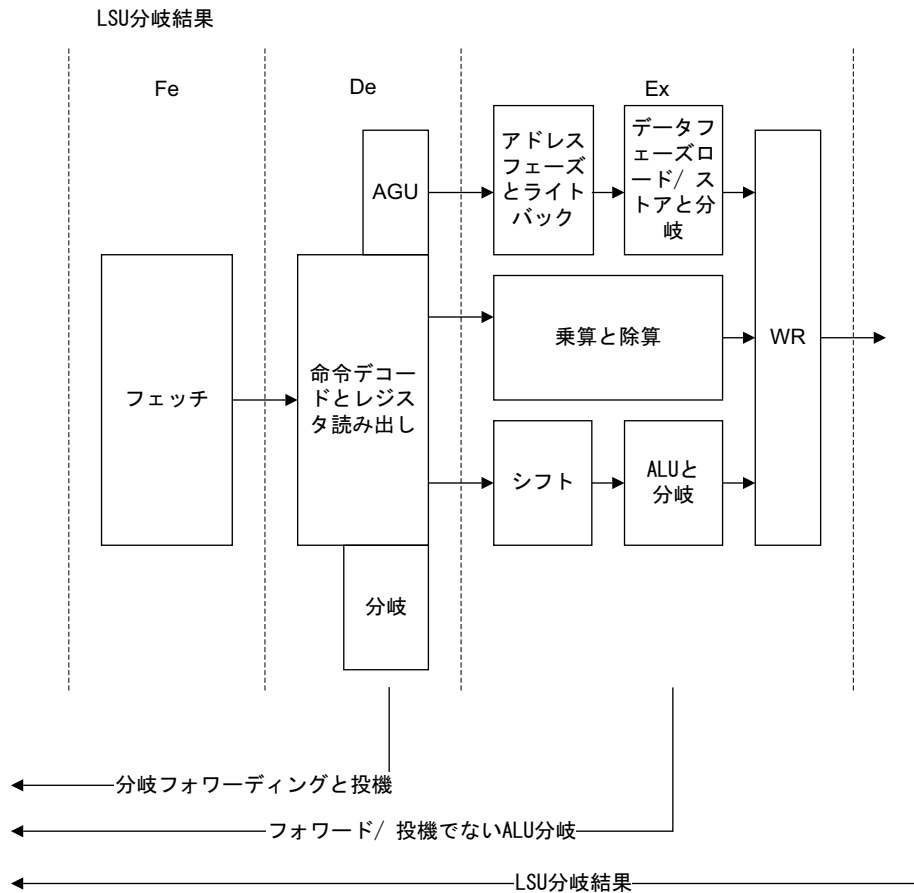


図 1-2 Cortex-M3 のパイプラインステージ

パイプラインステージの名前と機能は、次のとおりです。

- Fe**            命令フェッチで、命令メモリからデータが入ってきます。
- De**            命令デコーで、フォワードされたレジスタポートを使用した **LSU** アドレスの生成、および即値のオフセットやリンクレジスタ (**LR**) での分岐によるフォワーディング
- Ex**            命令実行で、複数サイクルのストールがある単一パイプライン、**AHB** インタフェースへの **LSU** アドレス / データのパイプライン処理、乗算 / 除算、および分岐結果付き **ALU**

**ALU** 使用ペナルティの無いパイプライン化された 2 サイクルのメモリアクセス、および間接ポインタ用のアドレス生成フォワーディングがパイプラインの構造により提供されます。

## 1.5 プリフェッチユニット

プリフェッチユニット (PFU) は、次の目的に使用されます。

- あらかじめ命令をフェッチし、PC 相対分岐命令をフォワードする。条件分岐の場合、フェッチは投機的になります。
- Thumb-2 命令を検出し、単一命令ワードとして出力します。
- ベクタロードを実行します。

PFU は、サイクル毎に 1 ワードを供給可能なメモリシステムから命令をフェッチします。PFU は 3 ワードまでのフェッチを FIFO にバッファすることができます。つまり、最大 3 つの Thumb-2 命令または 6 つの Thumb 命令をバッファすることができます。

PC+ 即値の ALU 加算として生成される分岐の大部分は、実際には、分岐オペコードのデコードフェーズより後に生成されることはありません。条件付きで実行される分岐の場合、アドレスは（バスのフェッチスロットを消費して）投機的に出力されます。そして、(ALU から) フォワードされた結果に基づき、その分岐バスがフェッチキューをフラッシュするか、保存するかが決定されます。

短いサブルーチン復帰は、BX LR の場合におけるフォワーディングの動作を有効に利用するように最適化されています。

## 1.6 分岐ターゲットのフォワーディング

オペコードが実行ステージに到達するよりも少なくとも 1 サイクル前に、分岐のメモリトランザクションを出力することで、プロセッサはある種の分岐タイプを前もって出力します。分岐は組み込みコントローラのアプリケーションにおいてかなりの部分を占めるので、分岐フォワーディングによってコアのパフォーマンスが向上します。影響がある分岐は、即値をオフセットとする PC 相対分岐、または LR をターゲットレジスタとして使用する分岐です。条件分岐はオペコード定義によって分岐したり、IT (If Then) ブロックの中で分岐したりしますが、条件分岐をフォワードするには、内部のクリティカルパスで条件の評価と平行して投機的にアドレスが出力される必要があります。

条件付のオペコードにおいて投機フェッチをした場合、分岐フォワーディングは、1 回のフェッチの機会を失います。しかし、3 エントリのフェッチキュー、16 ビットと 32 ビットオペコードの混在、および 1 サイクルの ALU によって、これを軽減することができます。追加のペナルティは、パイプラインストールの 1 サイクルです。最悪のケースは、3 つの 32 ビットの単一ロード / ストアオペコードで、その命令がワードアラインドでなく、データウェイトステータのない場合です。BRCHSTAT インタフェースが提供する情報は、条件実行にフォワードされた分岐、条件分岐の場合は分岐方向、および先行する条件付きオペコードがその後成功した場合には判定が立ち下がりにレジスタ出力されます。BRCHSTAT の詳細については、P.16-6「分岐ステータスインタフェース」を参照して下さい。

プリフェッチでレジスタ出力される ICODE を使ったコアの性能は、10% ほど遅いだけで、分岐フォワーディングインタフェースを持たないコアと事実上同じです。分岐フォワーディングは、アドレスインタフェースでレジスタ出力される前の内部アドレス生成回路と見なすことができます。1 サイクル早めの情報を利用できるタイミング余裕があれば、メモリコントローラに対する柔軟性が向上します。例えば、0.13  $\mu\text{m}$  プロセスからさらに 65nm プロセスの、より低い MHz で消費電力を抑えなければならないターゲット。ほかには、この早い時期のアドレスへのアクセスで、システムへレジスタ出力される前に、メモリコントローラがルックアップできるという柔軟性も得られます。

予測の失敗が発生するので、分岐の投機はウェイトステータがあるようなメモリに対してはコストが掛かります。このオーバーヘッドを避けるには、コントローラにおける規則として、条件分岐は投機的にフェッチせずレジスタ出力をするようにすれば、サブルーチンのコールや復帰は予測失敗のペナルティなしで分岐フォワーディングの利点を得ることができます。改良できる点は、後方へ戻る条件分岐のみを予測して、ループを高速化することです。もう 1 つの選択肢として、ARM コンパイラでは、ループの底部では後方へ無条件分岐して、ループの脱出の判定で前方への条件分岐を行なうようになっているので、ループの開始でコアがフェッチのキューを進めておくと、動作が良くなります。

BRCHSTAT には、次に実行段階に達するオペコードについての、他の情報も含まれています。BRCHSTAT がそのトランザクションに対応しているフォワードされた分岐と違って、BRCHSTAT はオペコードの実行という点ではどのトランザクションとも関連のないひとつのヒントであり、複数サイクルの間アサートされる可能性があります。コントローラは分岐がすぐに発生することを知っているので、この情報を使用して、追加のプリフェッチを抑制することができます。これにより、分岐先のターゲットが実行段階で生成されても。



以下のシナリオでは、分岐フォワーディングと **BRCHSTAT** 制御を使って、自分のメモリシステムで最高の性能を実現するための方法が示されています。シナリオでは、理想的なハードアーキテクチャの構成を前提としているので、命令が **ICODE** から実行され、リテラルが (**ICODE** に統合された) **DCODE** から実行され、そしてスタック / ヒープ / アプリケーションのデータが **SYSTEM** から実行されます。

- 0 ウェイトステート
- 0 ウェイトステート、レジスタ出力のフェッチインタフェース (**ICODE**)
- 1 ウェイトステートのフラッシュメモリ
- 1 ウェイトステートのフラッシュメモリ、レジスタ出力されるフェッチインタフェース (**ICODE**)
- P.1-1 「はじめに」

### 1.6.1 0 ウェイトステート

分岐予測により、それががない場合と比べて約 10% の向上が得られます。そして、極端な場合を除いて、プロセッサは 100% 分岐予測のすべての恩恵を受けますが、分岐の投機によるペナルティはありません。

### 1.6.2 0 ウェイトステート、レジスタ出力のフェッチインタフェース (**ICODE**)

分岐フォワーディングは、**ICODE** インタフェースでのタイミングがより厳しくなります。もしこのバスがシステムのクリティカルパスの場合、**ICODE** インタフェースがレジスタ出力されているかもしれません。ウェイトステート追加の約 25% のペナルティを避けるために、単一エントリのプリフェッチャとして機能する回路を追加して、レジスタ出力されるインタフェースのプッシュプル問題を回避することができます。

### 1.6.3 1 ウェイトステートのフラッシュメモリ

フラッシュメモリへのウェイトステートの追加は、どんなコアでもパフォーマンスに影響します。キャッシュを使用して、このペナルティを軽減することもできますが、リアルタイム性の保証およびシリコン面積に大きな影響を及ぼします。2 ラインのエントリを持つキャッシュラインのプリフェッチャにより、少ないゲートを使用して、キャッシュに匹敵するパフォーマンスを提供することができます。例えば **ARM7** ターゲットの場合には、32 ビット命令セットが使用されているので、128 ビットが一般的なプリフェッチ幅です。**Cortex-M3** プロセッサは、16 ビットと 32 ビットの混在命令セットである **Thumb-2** の恩恵を受けます。つまり、64 ビットのプリフェッチ幅により、128 ビットインタフェースとほぼ同等の恩恵を受けることができます。

### 1.6.4 1 ウェイトステートのフラッシュメモリ、レジスタ出力されるフェッチインタフェース (**ICODE**)

**ICODE** インタフェースをレジスタ出力にする必要がある場合、予測ミスのコストをプリフェッチコントローラのスレーブ側のみに軽減することができます。この場合も 0 ウェイトステートの場合と同様に、コアは **ICODE** インタフェースでのフェッチキュー要求の機会を失います。しかし、後に続く条件付き実行のレジスタ出力された **BRCHSTAT[3]** ステータスにより、コントローラのレジスタ出力されたシステムインタフェースでアイドルサイクルのように見せて、外部の予測ミスをマスクすることができます。

## 1.6.5 2 ウェイトステートのフラッシュメモリ

これは 1 ウェイトステートの場合と同じですが、分岐に対するペナルティが大きくなります。コンパイラツールが分岐のオーバーヘッドを減らし、条件ループをハードウェアの都合の良いほうにする度合いに応じて、コアとメモリシステムの速度のミスマッチの影響は減ります。128 ビットのインタフェースはこれらの点において優れています。

## 1.7 ストアバッファ

プロセッサには、2つのストアバッファが組み込まれています。

- 即値オフセットを持つオペコード用の、Cortex-M3 コア LSU のストアバッファ
- ウェイトステートおよびアンアラインドのトランザクション用の、バスマトリックスのストアバッファ

コアのストアバッファにより、コンパイルされたコードによくある `STR rx,[ry,#imm]` の場合が最適化されます。つまり、次のオペコードがストアのデータフェーズと重なることが可能なので、パイプラインの観点から見ると、オペコードを 1 サイクルにすることができます。

プロセッサ内のバスマトリックス相互接続により、コアのアンアラインド動作とビットバンドが管理されます。バスマトリックスのストアバッファは、システムのウェイトステートの解消および複数のトランザクションに分かれたアンアラインドアクセスに有用です。

バッファ可としてマークされたトランザクションのみがストアバッファを使用します。スタック操作は本質的にバッファ不可なので、どちらのバッファも使用しません。

1.8 命令セットの概要

このセクションでは、次の内容について説明します。

- プロセッサの 16 ビット命令の概要
- プロセッサの 32 ビット命令の概要

Cortex-M3 の 16 ビット命令の一覧を、表 1-1 に示します。

表 1-1 Cortex-M3 の 16 ビット命令の概要

操作	アセンブラ
レジスタの値と C フラグをレジスタの値に加算	ADC <Rd>, <Rm>
3 ビット即値をレジスタに加算	ADD <Rd>, <Rn>, #<immed_3>
8 ビット即値をレジスタに加算	ADD <Rd>, #<immed_8>
下位レジスタの値を下位レジスタの値に加算	ADD <Rd>, <Rn>, <Rm>
上位レジスタの値を下位または上位レジスタの値に加算	ADD <Rd>, <Rm>
PC 相対 4 × (8 ビット即値) でレジスタに加算	ADD <Rd>, PC, #<immed_8> * 4
SP 相対 4 × (8 ビット即値) でレジスタに加算	ADD <Rd>, SP, #<immed_8> * 4
4 × (7 ビット即値) を SP に加算	ADD SP, #<immed_7> * 4
レジスタの値をビット単位に AND	AND <Rd>, <Rm>
即値の数値により算術右シフト	ASR <Rd>, <Rm>, #<immed_5>
レジスタの数値により算術右シフト	ASR <Rd>, <Rs>
条件付き分岐	B<cond> <target address>
無条件分岐	B <target_address>
ビットクリア	BIC <Rd>, <Rm>
ソフトウェアブレイクポイント	BKPT <immed_8>
リンク付き分岐	BL <Rm>
リンク付き分岐と状態遷移	BLX <Rm>
比較して非 0 で分岐	CBNZ <Rn>, <label>
比較して 0 で分岐	CBZ <Rn>, <label>
レジスタ値の 2 の補数（負数、negation）を別のレジスタの値と比較	CMN <Rn>, <Rm>
8 ビット即値と比較	CMP <Rn>, #<immed_8>
レジスタと比較	CMP <Rn>, <Rm>
上位レジスタを下位または上位レジスタと比較	CMP <Rn>, <Rm>
プロセッサ状態変更	CPS <effect>, <iflags>

表 1-1 Cortex-M3 の 16 ビット命令の概要

操作	アセンブラ
上位または下位レジスタの値を別の上位または下位レジスタにコピー	CPY <Rd> <Rm>
レジスタの値をビット単位で排他的論理和	EOR <Rd>, <Rm>
次の命令を条件付け、次の 2 つの命令を条件付け、次の 3 つの命令を条件付け、次の 4 つの命令を条件付け	IT <cond> IT<x> <cond> IT<x><y> <cond> IT<x><y><z> <cond>
複数ワードを連続するメモリからロード	LDMIA <Rn>!, <registers>
ベースレジスタのアドレス + 5 ビット即値オフセットのメモリからワードをロード	LDR <Rd>, [<Rn>, #<immed_5> * 4]
ベースレジスタのアドレス + レジスタオフセットのメモリからワードをロード	LDR <Rd>, [<Rn>, <Rm>]
PC アドレス + 8 ビット即値オフセットのメモリからワードをロード	LDR <Rd>, [PC, #<immed_8> * 4]
SP アドレス + 8 ビット即値オフセットのメモリからワードをロード	LDR, <Rd>, [SP, #<immed_8> * 4]
レジスタアドレス + 5 ビット即値オフセットのメモリバイト [7:0] ロード	LDRB <Rd>, [<Rn>, #<immed_5>]
レジスタアドレス + レジスタオフセットのメモリからバイト [7:0] をロード	LDRB <Rd>, [<Rn>, <Rm>]
レジスタアドレス + 5 ビット即値オフセットのメモリハーフワード [15:0] ロード	LDRH <Rd>, [<Rn>, #<immed_5> * 2]
レジスタアドレス + レジスタオフセットのメモリからハーフワード [15:0] をロード	LDRH <Rd>, [<Rn>, <Rm>]
レジスタアドレス + レジスタオフセットのメモリから符号付きバイト [7:0] をロード	LDRSB <Rd>, [<Rn>, <Rm>]
レジスタアドレス + レジスタオフセットのメモリから符号付きハーフワード [15:0] をロード	LDRSH <Rd>, [<Rn>, <Rm>]
即値により論理左シフト	LSL <Rd>, <Rm>, #<immed_5>
レジスタの値により論理左シフト	LSL <Rd>, <Rs>
即値により論理右シフト	LSR <Rd>, <Rm>, #<immed_5>
レジスタの値により論理右シフト	LSR <Rd>, <Rs>
8 ビット即値をレジスタに移動	MOV <Rd>, #<immed_8>
下位レジスタの値を下位レジスタに移動	MOV <Rd>, <Rn>
上位または下位レジスタの値を上位または下位レジスタに移動	MOV <Rd>, <Rm>
レジスタの値を乗算	MUL <Rd>, <Rm>

表 1-1 Cortex-M3 の 16 ビット命令の概要

操作	アセンブラ
レジスタの値の否定（1 の補数、complement）をレジスタに移動	MVN <Rd>, <Rm>
レジスタの値を負（negative、2 の補数）にしてレジスタに保存	NEG <Rd>, <Rm>
無操作	NOP <c>
レジスタの値をビット単位で論理和	ORR <Rd>, <Rm>
スタックから複数レジスタをポップ	POP <registers>
スタックから複数レジスタおよび PC をポップ	POP <registers, PC>
複数レジスタをスタックへプッシュ	PUSH <registers>
LR および複数レジスタをスタックへプッシュ	PUSH <registers, LR>
ワード中のバイト順を反転して、レジスタへコピー	REV <Rd>, <Rn>
2 つのハーフワード中でそれぞれのバイト順を反転して、レジスタへコピー	REV16 <Rd>, <Rn>
下位ハーフワード [15:0] でバイト順を反転、符号拡張して、レジスタへコピー	REVSH <Rd>, <Rn>
レジスタ内の量により、右ローテート	ROR <Rd>, <Rs>
レジスタの値と C フラグをレジスタの値から減算	SBC <Rd>, <Rm>
イベント送信	SEV <c>
複数のレジスタからワードを連続するメモリ位置へストア	STMIA <Rn>!, <registers>
レジスタのアドレス+5 ビット即値のオフセットへレジスタのワードをストア	STR <Rd>, [<Rn>, #<immed_5> * 4]
レジスタのアドレスへレジスタのワードをストア	STR <Rd>, [<Rn>, <Rm>]
SP アドレス+8 ビット即値のオフセットへレジスタのワードをストア	STR <Rd>, [SP, #<immed_8> * 4]
レジスタのアドレス+5 ビット即値のオフセットへレジスタのバイト [7:0] をストア	STRB <Rd>, [<Rn>, #<immed_5>]
レジスタのアドレスへレジスタのバイト [7:0] をストア	STRB <Rd>, [<Rn>, <Rm>]
レジスタのアドレス+5 ビット即値のオフセットへレジスタのハーフワード [15:0] をストア	STRH <Rd>, [<Rn>, #<immed_5> * 2]
レジスタのアドレス+レジスタによるオフセットへレジスタのハーフワード [15:0] をストア	STRH <Rd>, [<Rn>, <Rm>]
3 ビット即値をレジスタから減算	SUB <Rd>, <Rn>, #<immed_3>
8 ビット即値をレジスタの値から減算	SUB <Rd>, #<immed_8>
レジスタの値を減算	SUB <Rd>, <Rn>, <Rm>
4 ×（7 ビット即値）を SP から減算	SUB SP, #<immed_7> * 4

表 1-1 Cortex-M3 の 16 ビット命令の概要

操作	アセンブラ
8 ビット即値の呼び出しコードにより、オペレーティングシステムのサービス呼び出し	SVC <immed_8>
レジスタからバイト [7:0] を抽出し、レジスタに移動して、32 ビットに符号拡張	SXTB <Rd>, <Rm>
レジスタからハーフワード [15:0] を抽出し、レジスタに移動して、32 ビットに符号拡張	SXTH <Rd>, <Rm>
別のレジスタの値と論理積を実行して、レジスタの設定されているビットをテスト	TST <Rn>, <Rm>
レジスタからバイト [7:0] を抽出し、レジスタに移動して、32 ビットにゼロ拡張	UXTB <Rd>, <Rm>
レジスタからハーフワード [15:0] を抽出し、レジスタに移動して、32 ビットにゼロ拡張	UXTH <Rd>, <Rm>
イベント待ち	WFE <c>
割り込み待ち	WFI <c>

Cortex-M3 の 32 ビット命令の一覧を、表 1-2 に示します。

表 1-2 Cortex-M3 の 32 ビット命令の概要

操作	アセンブラ
レジスタの値、12 ビット即値、C ビットを加算	ADC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
レジスタの値、シフトしたレジスタの値、C ビットを加算	ADC{S}.W <Rd>, <Rn>, <Rm>[, <shift>]
レジスタの値と 12 ビット即値を加算	ADD{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
レジスタの値とシフトしたレジスタの値を加算	ADD{S}.W <Rd>, <Rm>[, <shift>]
レジスタの値と 12 ビット即値を加算	ADDW.W <Rd>, <Rn>, #<immed_12>
レジスタの値と 12 ビット即値をビット単位論理積	AND{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
レジスタの値とシフトされたレジスタの値をビット単位論理積	AND{S}.W <Rd>, <Rn>, <Rm>[, <shift>]
レジスタの数値により算術右シフト	ASR{S}.W <Rd>, <Rn>, <Rm>
条件分岐	B{cond}.W <label>
ビットフィールドをクリア	BFC.W <Rd>, #<lsb>, #<width>
ビットフィールドを 1 つのレジスタの値から別のレジスタの値に挿入	BFI.W <Rd>, <Rn>, #<lsb>, #<width>

表 1-2 Cortex-M3 の 32 ビット命令の概要

操作	アセンブラ
レジスタの値と 12 ビット即値の否定（1 の補数）とをビット単位の論理積	BIC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
レジスタの値とシフトされたレジスタの値の否定（1 の補数）とをビット単位の論理積	BIC{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
リンク付き分岐	BL <label>
リンク付き分岐（即値）	BL<c> <label>
無条件分岐	B.W <label>
排他クリアは、実行中のプロセッサのローカルレコードで、アドレスが排他アクセスの要求を受けているものをクリアします。	CLREX <c>
レジスタの値に含まれる先行ゼロの数を返す	CLZ.W <Rd>, <Rn>
レジスタの値と 12 ビット即値の 2 の補数とを比較	CMN.W <Rn>, #<modify_constant(immed_12)>
レジスタの値とシフトされたレジスタの値の 2 の補数とを比較	CMN.W <Rn>, <Rm>{, <shift>}
レジスタの値と 12 ビット即値とを比較	CMP.W <Rn>, #<modify_constant(immed_12)>
レジスタの値とシフトされたレジスタの値とを比較	CMP.W <Rn>, <Rm>{, <shift>}
データメモリ バリア	DMB <c>
データ同期化バリア	DSB <c>
レジスタの値と 12 ビット即値とを排他的論理和	EOR{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
レジスタの値とシフトされたレジスタの値とを排他的論理和	EOR{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
命令同期化バリア	ISB <c>
ポストインクリメント (IA) またはプリデクリメント (DB) で、メモリからレジスタへ複数ロード	LDM{IA DB}.W <Rn>{!}, <registers>
ベースレジスタのアドレス + 12 ビット即値のオフセットのメモリからワードをロード	LDR.W <Rxf>, [<Rn>, #<offset_12>]
ベースレジスタのアドレス + 12 ビット即値のオフセットのメモリから PC へワードをロード（分岐）	LDR.W PC, [<Rn>, #<offset_12>]
ポストインデックスで、ベースレジスタのアドレスに 8 ビット即値をオフセットしたメモリから PC へワードをロード（分岐）	LDR.W PC, [Rn], #<+/-<offset_8>



表 1-2 Cortex-M3 の 32 ビット命令の概要

操作	アセンブラ
ポストインデックスで、ベースレジスタのアドレスに 8 ビット即値をオフセットしたメモリからワードをロード	LDR.W <Rxf>, [<Rn>], #+/-<offset_8>
プレインデックスで、ベースレジスタのアドレスに 8 ビット即値をオフセットしたメモリからワードをロード	LDR.W <Rxf>, [<Rn>, #+/-<offset_8>]! LDRT.W <Rxf>, [<Rn>, #<offset_8>]
プレインデックスで、ベースレジスタのアドレスに 8 ビット即値をオフセットしたメモリから PC ヘワードをロード (分岐)	LDR.W PC, [<Rn>, #+/-<offset_8>]!
位置が 0、1、2、または 3 つ左にシフトされたレジスタのアドレスのメモリからワードをロード	LDR.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
位置が 0、1、2、または 3 つ左にシフトされたレジスタのアドレスのメモリから PC ヘワードをロード (分岐)	LDR.W PC, [<Rn>, <Rm>{, LSL #<shift>}]
PC アドレスに 12 ビット即値をオフセットしたメモリからワードをロード	LDR.W <Rxf>, [PC, #+/-<offset_12>]
PC アドレスに 12 ビット即値をオフセットしたメモリから PC ヘワードをロード (分岐)	LDR.W PC, [PC, #+/-<offset_12>]
ベースレジスタのアドレス + 12 ビット即値のオフセットのメモリからバイト [7:0] をロード	LDRB.W <Rxf>, [<Rn>, #<offset_12>]
ポストインデックスで、ベースレジスタのアドレスを 8 ビット即値でオフセットしたメモリからバイト [7:0] をロード	LDRB.W <Rxf>, [<Rn>], #+/-<offset_8>
位置が 0、1、2、または 3 つ左にシフトされたレジスタのアドレスのメモリからバイト [7:0] をロード	LDRB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
プレインデックスで、ベースレジスタのアドレスに 8 ビット即値でオフセットしたメモリからバイト [7:0] をロード	LDRB.W <Rxf>, [<Rn>, #+/-<offset_8>]!
PC アドレスに 12 ビット即値でオフセットしたメモリからバイトをロード	LDRB.W <Rxf>, [PC, #+/-<offset_12>]
プレインデックスで、レジスタのアドレスに、8 ビット即値 × 4 をオフセットしたメモリからダブルワードをロード	LDRD.W <Rxf>, <Rxf2>, [<Rn>, #+/-<offset_8> * 4]{!}
ポストインデックスで、レジスタのアドレスに、8 ビット即値 × 4 をオフセットしたメモリからダブルワードをロード	LDRD.W <Rxf>, <Rxf2>, [<Rn>], #+/-<offset_8> * 4

表 1-2 Cortex-M3 の 32 ビット命令の概要

操作	アセンブラ
排他レジスタロードは、ベースレジスタの値と即値オフセットからアドレスを計算し、ワードをメモリからロードして、レジスタに書き込みます。	LDREX<c> <Rt>,[<Rn>{,<#imm>}]
排他レジスタロード ハーフワードは、ベースレジスタの値と即値オフセットからアドレスを計算し、ハーフワードをメモリからロードして、レジスタに書き込みます。	LDREXH<c> <Rt>,[<Rn>{,<#imm>}]
排他レジスタロード バイトは、ベースレジスタの値と即値オフセットからアドレスを計算し、バイトをメモリからロードして、レジスタに書き込みます。	LDREXB<c> <Rt>,[<Rn>{,<#imm>}]
ベースレジスタのアドレス + 12 ビット即値のオフセットのメモリからハーフワード [15:0] をロード	LDRH.W <Rxf>,[<Rn>,<#offset_12>]
ブレインデクスで、ベースレジスタのアドレスに 8 ビット即値でオフセットしてハーフワード [15:0] をロード	LDRH.W <Rxf>,[<Rn>,<#<+/-<offset_8>]!
ポストインデクスで、ベースレジスタのアドレスに 8 ビット即値でオフセットしてハーフワード [15:0] をロード	LDRH.W <Rxf>,[<Rn>],<#<+/-<offset_8>
位置が 0、1、2、または 3 つ左にシフトされたレジスタのアドレスのメモリからハーフワード [15:0] をロード	LDRH.W <Rxf>,[<Rn>,<Rm>{,<LSL #<shift>}]
PC アドレスに 12 ビット即値をオフセットしたメモリからハーフワードをロード	LDRH.W <Rxf>,[PC,<#<+/-<offset_12>]
ベースレジスタのアドレス + 12 ビット即値のオフセットのメモリから符号付きバイト [7:0] をロード	LDRSB.W <Rxf>,[<Rn>,<#<offset_12>]
ポストインデクスで、ベースレジスタのアドレスに 8 ビット即値をオフセットしたメモリから符号付きバイト [7:0] をロード	LDRSB.W <Rxf>,[<Rn>],<#<+/-<offset_8>
ブレインデクスで、ベースレジスタのアドレスに 8 ビット即値をオフセットしたメモリから符号付きバイト [7:0] をロード	LDRSB.W <Rxf>,[<Rn>,<#<+/-<offset_8>]!
位置が 0、1、2、または 3 つ左にシフトされたレジスタのアドレスのメモリから符号付きバイト [7:0] をロード	LDRSB.W <Rxf>,[<Rn>,<Rm>{,<LSL #<shift>}]
PC アドレスに 12 ビット即値をオフセットしたメモリから符号付きバイトをロード	LDRSB.W <Rxf>,[PC,<#<+/-<offset_12>]

表 1-2 Cortex-M3 の 32 ビット命令の概要

操作	アセンブラ
ベースレジスタのアドレス + 12 ビット即値をオフセットしたメモリから符号付きハーフワード [15:0] をロード	LDRSH.W <Rxf>, [<Rn>, #<offset_12>]
ポストインデクスで、ベースレジスタのアドレスを 8 ビット即値でオフセットして、符号付きハーフワード [15:0] をロード	LDRSH.W <Rxf>, [<Rn>], #+/-<offset_8>
プレインデクスで、ベースレジスタのアドレスに 8 ビット即値でオフセットして、符号付きハーフワード [15:0] をロード	LDRSH.W <Rxf>, [<Rn>, #<+/-<offset_8>]!
位置が 0、1、2、または 3 つ左にシフトされたレジスタのアドレスからメモリ符号付きハーフワード [15:0] をロード	LDRSH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
PC アドレスを 12 ビット即値でオフセットしたメモリから符号付きハーフワードをロード	LDRSH.W <Rxf>, [PC, #+/-<offset_12>]
レジスタの数値によりレジスタ値を論理左シフト	LSL{S}.W <Rd>, <Rn>, <Rm>
レジスタの数値によりレジスタ値を論理右シフト	LSR{S}.W <Rd>, <Rn>, <Rm>
2 つの符号付きまたは符号なしレジスタ値を乗算し、下位 32 ビットをレジスタの値に加算（積和）	MLA.W <Rd>, <Rn>, <Rm>, <Racc>
2 つの符号付きまたは符号なしレジスタ値を乗算し、下位 32 ビットをレジスタの値から減算	MLS.W <Rd>, <Rn>, <Rm>, <Racc>
12 ビット即値をレジスタに移動	MOV{S}.W <Rd>, #<modify_constant(immed_12)>
シフトしたレジスタの値をレジスタに移動	MOV{S}.W <Rd>, <Rm>{, <shift>}
16 ビット即値をレジスタの上位ハーフワード [31:16] に移動	MOVT.W <Rd>, #<immed_16>
16 ビット即値をレジスタの下位ハーフワード [15:0] に移動し、上位ハーフワード [31:16] をクリア	MOVW.W <Rd>, #<immed_16>
ステータスレジスタからレジスタに移動	MRS<c> <Rd>, <psr>
ステータスレジスタに移動	MSR<c> <psr>_<fields>, <Rn>
2 つの符号付きまたは符号なしレジスタ値を乗算	MUL.W <Rd>, <Rn>, <Rm>
無操作	NOP.W
レジスタの値と 12 ビット即値の NOT とを論理和	ORN{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
レジスタの値とシフトしたレジスタの値の NOT とを論理和	ORN{S}.W <Rd>, <Rn>, <Rm>{, <shift>}

表 1-2 Cortex-M3 の 32 ビット命令の概要

操作	アセンブラ
レジスタの値と 12 ビット即値を論理和	ORR{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
レジスタの値とシフトしたレジスタの値を論理和	ORR{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
ビットの順序を反転	RBIT.W <Rd>, <Rm>
ワード中のバイト順を反転	REV.W <Rd>, <Rm>
各ハーフワード中のバイト順をそれぞれ反転	REV16.W <Rd>, <Rn>
下位ハーフワードのバイト順を反転して、符号拡張	REVSH.W <Rd>, <Rn>
レジスタ内の数により、右ローテート	ROR{S}.W <Rd>, <Rn>, <Rm>
拡張付き右ローテート	RRX{S}.W <Rd>, <Rm>
レジスタの値を 12 ビット即値から減算	RSB{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
レジスタの値をシフトしたレジスタの値から減算	RSB{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
12 ビット即値と C ビットをレジスタの値から減算	SBC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
シフトしたレジスタの値と C ビットをレジスタの値から減算	SBC{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
選択されたビットをレジスタにコピーし、符号拡張	SBFX.W <Rd>, <Rn>, #<lsb>, #<width>
符号付き除算	SDIV<c> <Rd>, <Rn>, <Rm>
イベント送信	SEV<c>
符号付きワードを乗算して、符号拡張された値を、一対のレジスタ値に累算	SMLAL.W <RdLo>, <RdHi>, <Rn>, <Rm>
2 つの符号付きレジスタ値を乗算	SMULL.W <RdLo>, <RdHi>, <Rn>, <Rm>
符号付き飽和	SSAT.W <c> <Rd>, #<imm>, <Rn>{, <shift>}
複数レジスタのワードを連続メモリ位置へストア	STM{IA DB}.W <Rn>{!}, <registers>
レジスタのアドレス + 12 ビット即値のオフセットに、レジスタのワードをストア	STR.W <Rxf>, [<Rn>, #<offset_12>]
ポストインデックスで、レジスタのアドレス + 8 ビット即値のオフセットに、レジスタのワードをストア	STR.W <Rxf>, [<Rn>], #+/-<offset_8>
位置が 0、1、2、または 3 つシフトされたレジスタのアドレスに、レジスタのワードをストア	STR.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
プレインデックスまたはポストインデックスで、レジスタのアドレス + 8 ビット即値のオフセットに、レジスタのワードをストア	STR.W <Rxf>, [<Rn>, #+/-<offset_8>]{!} STRT.W <Rxf>, [<Rn>, #<offset_8>]
プレインデックスで、レジスタのアドレスを 8 ビット即値でオフセットして、レジスタのバイト [7:0] をストア	STRB{T}.W <Rxf>, [<Rn>, #+/-<offset_8>]{!}

表 1-2 Cortex-M3 の 32 ビット命令の概要

操作	アセンブラ
レジスタのアドレス + 12 ビット即値のオフセットに、レジスタのバイト [7:0] をストア	STRB.W <Rxf>, [<Rn>, #<offset_12>]
ポストインデクスで、レジスタのアドレスを 8 ビット即値でオフセットして、レジスタのバイト [7:0] をストア	STRB.W <Rxf>, [<Rn>], #+/-<offset_8>
位置が 0、1、2、または 3 つシフトされたレジスタのアドレスに、レジスタのバイト [7:0] をストア	STRB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
ダブルワードをブレインデクスでストア	STRD.W <Rxf>, <Rxf2>, [<Rn>, #+/-<offset_8> * 4]{!}
ダブルワードをポストインデクスでストア	STRD.W <Rxf>, <Rxf2>, [<Rn>, #+/-<offset_8> * 4]
排他レジスタストアは、ベースレジスタの値と即値オフセットからアドレスを計算し、実行中のプロセッサがアドレス指定されたメモリに対する排他アクセスを持っている場合、ワードをレジスタからメモリにストアします。	STREX <c> <Rd>, <Rt>, [<Rn>{, #<imm>}]
排他レジスタストア バイトは、ベースレジスタの値からアドレスを導出し、実行中のプロセッサがアドレス指定されたメモリに対する排他アクセスを持っている場合、バイトをレジスタからメモリにストアします。	STREXB <c> <Rd>, <Rt>, [<Rn>]
排他レジスタストア ハーフワードは、ベースレジスタの値からアドレスを導出し、実行中のプロセッサがアドレス指定されたメモリに対する排他アクセスを持っている場合、ハーフワードをレジスタからメモリにストアします。	STREXH <c> <Rd>, <Rt>, [<Rn>]
レジスタのアドレス + 12 ビット即値のオフセットに、レジスタのハーフワード [15:0] をストア	STRH.W <Rxf>, [<Rn>, #<offset_12>]
位置が 0、1、2、または 3 つシフトされたレジスタのアドレスに、レジスタのハーフワード [15:0] をストア	STRH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
ブレインデクスで、レジスタのアドレスを 8 ビット即値でオフセットして、レジスタのハーフワード [15:0] をストア	STRH{T}.W <Rxf>, [<Rn>, #+/-<offset_8>]{!}
ポストインデクスで、レジスタのアドレスを 8 ビット即値でオフセットして、レジスタのハーフワード [15:0] をストア	STRH.W <Rxf>, [<Rn>], #+/-<offset_8>
12 ビット即値をレジスタの値から減算	SUB{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
シフトしたレジスタの値をレジスタの値から減算	SUB{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
12 ビット即値をレジスタの値から減算	SUBW.W <Rd>, <Rn>, #<immed_12>

表 1-2 Cortex-M3 の 32 ビット命令の概要

操作	アセンブラ
バイトを 32 ビットに符号拡張	SXTB.W <Rd>, <Rm>{, <rotation>}
ハーフワードを 32 ビットに符号拡張	SXTH.W <Rd>, <Rm>{, <rotation>}
バイトでテーブル分岐	TBB [<Rn>, <Rm>]
ハーフワードでテーブル分岐	TBH [<Rn>, <Rm>, LSL #1]
レジスタの値と 12 ビット即値で排他的論理和	TEQ.W <Rn>, #<modify_constant(immed_12)>
レジスタの値とシフトしたレジスタの値で排他的論理和	TEQ.W <Rn>, <Rm>{, <shift>}
レジスタの値と 12 ビット即値で論理積	TST.W <Rn>, #<modify_constant(immed_12)>
レジスタの値とシフトしたレジスタの値で論理積	TST.W <Rn>, <Rm>{, <shift>}
レジスタの値のビットフィールドをレジスタにコピーし、32 ビットにゼロ拡張	UBFX.W <Rd>, <Rn>, #<lsb>, #<width>
符号なし除算	UDIV<c> <Rd>,<Rn>,<Rm>
2 つの符号なしレジスタ値を乗算して、一対のレジスタ値に累算	UMLAL.W <RdLo>, <RdHi>, <Rn>, <Rm>
2 つの符号なしレジスタ値を乗算	UMULL.W <RdLo>, <RdHi>, <Rn>, <Rm>
符号なし飽和	USAT <c> <Rd>, #<imm>, <Rn>{, <shift>}
符号なしバイトをレジスタにコピーして、32 ビットにゼロ拡張	UXTB.W <Rd>, <Rm>{, <rotation>}
符号なしハーフワードをレジスタにコピーして、32 ビットにゼロ拡張	UXTH.W <Rd>, <Rm>{, <rotation>}
イベント待ち	WFE.W
割り込み待ち	WFI.W

## 1.9 製品リビジョン

このセクションでは、プロセッサのリリースごとの機能の違いについて、概要を説明します。

- *r0p0* と *r1p0* の間の機能面での相違点
- *r1p0* と *r1p1* の間の機能面での相違点: P.1-32

### 1.9.1 *r0p0* と *r1p0* の間の機能面での相違点

機能面の相違点の概要は次のとおりです。

- 構成可能なデータ値比較を **DWT** モジュールに追加。P.11-13「**DWT**」参照。
- **DWT\_FUNCTION** へ **MATCHED** ビットを追加。P.11-13「**DWT**」参照。
- Cortex-M3 への入力として **ETMFIFOFULL** を追加。P.A-13「**ETM** インタフェース」参照。
- Cortex-M3 への出力として **ETMISTALL** を追加。P.A-13「**ETM** インタフェース」参照。
- **ITM** に **SWVMode** を追加。**SWVMode** をサポートするために、**TPIUEMIT** と **TPIUBAUD** を **TPIU** からの出力およびプロセッサの入力として追加。P.11-29「**ITM**」参照。
- **CPUID** ベースレジスタのバリエーション (**VARIANT**) フィールドが Rev1 を示すよう変更。P.8-7「**NVIC** レジスタの説明」参照。
- Cortex-M3 Rev0 のビットバンドアクセスは、**BE8** モードでは、アクセスサイズがバイトの必要がありました。Cortex-M3 Rev1 では、**BE8** のビットバンドアクセスがどのアクセスサイズでも機能するように変更されました。
- すべての例外が 8 バイトのスタックアライメントを持つように、**STKALIGN** と呼ばれる構成ビットを追加。P.8-7「**NVIC** レジスタの説明」参照。
- 補助フォールトステータス レジスタをアドレス 0xE000ED3C に追加。このレジスタを設定するために、**AUXFAULT** と呼ばれる 32 ビットバスが追加されました。P.8-7「**NVIC** レジスタの説明」参照。
- **HTM** サポートの追加。第 17 章「**AHB** トレースマクロセル インタフェース」参照。
- **I-Code** と **D-Code** のキャッシュ可とバッファ可の **HPROT** 値が、恒久的にライトスルーに接続されました。P.14-4「**ICode** バスインタフェース」および P.14-6「**DCode** バスインタフェース」参照。
- **IFLUSH** という名前の新しい入力が追加されました。P.A-4「その他」参照。

- HMASTER ポートの追加。P.A-8「*DCode* インタフェース」と P.A-9「システムバス インタフェース」参照。
- SWJ-DP の追加。これは、JTAG-DP と SW-DP を結合させる標準的な CoreSight™ デバッグポートです。P.12-2「*DP* について」参照。
- アドレス 0xE000101C に、DWT\_PCSR レジスタを追加。P.11-13「*DWT*」参照。
- **DNOTITRANS** と呼ばれる新しい入力を追加。P.14-9「コードバスの統合」参照。
- r0p0 リリースまでのエラーを修正。

## 1.9.2 r1p0 と r1p1 の間の機能面での相違点

機能面の相違点の概要は、次のとおりです。

- ウォッチポイントの生成用のデータ値一致が実装時に構成になりました。P.11-13「*DWT*」参照。
- ETM にアーキテクチャクロックゲーティングをオプションで実装するための定義を追加。以前のリリースでは、ETM にはアーキテクチャクロックゲーティングが常に存在していました。
- r0p0 および r1p0 では、**DAPCLKEN** は静的な信号の必要がありました。この要件は r1p1 では取り除かれています。
- 現在の未完了の命令フェッチが完了するまでは、**SLEEPING** 信号が抑制されるようになりました。
- r1p0 リリースまでのエラーを修正。



## 第 2 章

# プログラマモデル

本章では、プロセッサのプログラマモデルについて説明します。本章は以下のセクションから構成されています。

- プログラマモデル : P.2-2
- 特権アクセスおよびユーザアクセス : P.2-3
- レジスタ : P.2-4
- データタイプ : P.2-10
- メモリフォーマット : P.2-11
- 命令セット : P.2-13

## 2.1 プログラマモデル

Cortex-M3 プロセッサには、ARM v7-M アーキテクチャが実装されています。これには、すべての 16 ビット Thumb 命令セット、および基本 32 ビット Thumb-2 命令セットのアーキテクチャが組み込まれています。このプロセッサは、ARM 命令を実行することはできません。

Thumb 命令セットは ARM 命令セットのサブセットで、16 ビットにエンコードし直したものです。コード密度が高く、メモリに 16 ビット幅またはそれより狭いデータバスを使用するシステムをサポートしています。

Thumb-2 は、Thumb 命令セットアーキテクチャ (ISA) に対する大幅な拡張になっています。Thumb-2 により Thumb より高いコード密度が可能になり、16/32 ビット命令で高度な性能が提供されます。

### 2.1.1 動作モード

プロセッサは、スレッドモードとハンドラモードの 2 つの動作モードをサポートしています。

- スレッドモードはリセットで開始され、例外復帰の結果として入ることができます。スレッドモードでは、特権コードとユーザ（非特権）コードの両方が実行できます。
- ハンドラモードには、例外の結果として入ります。ハンドラモードでは、すべてのコードが特権コードです。

### 2.1.2 動作状態

プロセッサは、次の 2 つの動作状態で動作できます。

- Thumb 状態。これは、16 ビットおよび 32 ビットのハーフワードにアラインされた Thumb および Thumb-2 命令を実行する標準の実行状態です。
- デバッグ状態。これは、デバッグホールド中の状態です。

## 2.2 特権アクセスおよびユーザアクセス

コードは、特権または非特権で実行できます。非特権実行の場合は、一部のリソースへのアクセスが制限されるか排除されます。特権実行の場合は、すべてのリソースにアクセスできます。ハンドラモードは常に特権です。スレッドモードは、特権または非特権が可能です。

スレッドモードはリセットにより特権になりますが、**MSR** 命令を使用して **CONTROL[0]** ビットをクリアすることにより、ユーザすなわち非特権に変更することができます。ユーザでのアクセスにより、次のことが防止されます。

- **FAULTMASK** と **PRIMASK** をセットする **CPS** などの一部の命令の使用
- システム制御空間 (SCS)(System Control Space) での大部分のレジスタへのアクセス

スレッドモードが特権からユーザに変更されると、ユーザでは特権に戻すことはできません。ハンドラのみが、スレッドモードを特権に変更できます。ハンドラモードは常に特権です。

### 2.2.1 メインスタックおよびプロセススタック

リセットにより、すべてのコードはメインスタックを使用します。**SVC** などの例外ハンドラは、ハンドラ退出時に使用する **EXC\_RETURN** を変更することにより、スレッドモードで使用されているスタックをメインスタックからプロセススタックに変更することができます。すべての例外はメインスタックを継続して使用します。スタックポインタ **r13** は、**SP\_main** と **SP\_process** との間で切り替わるバンクレジスタです。プロセススタックまたはメインスタックのいずれか一方のスタックのみが、**r13** を使用して常に可視です。

スレッドモードでは、**MSR** 命令を使用して **CONTROL[1]** に書き込むことにより、メインスタックからプロセススタックに切り替えが可能であり、さらにハンドラモードから退出時に **EXC\_RETURN** 値を使用して選択も可能です。

2.3 レジスタ

プロセッサには、次の 32 ビットのレジスタがあります。

- 汎用レジスタは 13 個で、r0 ~ r12
- スタックポインタは、バンクになっているレジスタのエイリアスで、実体は SP\_process および SP\_main
- リンクレジスタ、r14
- プログラムカウンタ、r15
- 1つのプログラムステータス レジスタ、xPSR

プロセッサのレジスタセットを、図 2-1 に示します。

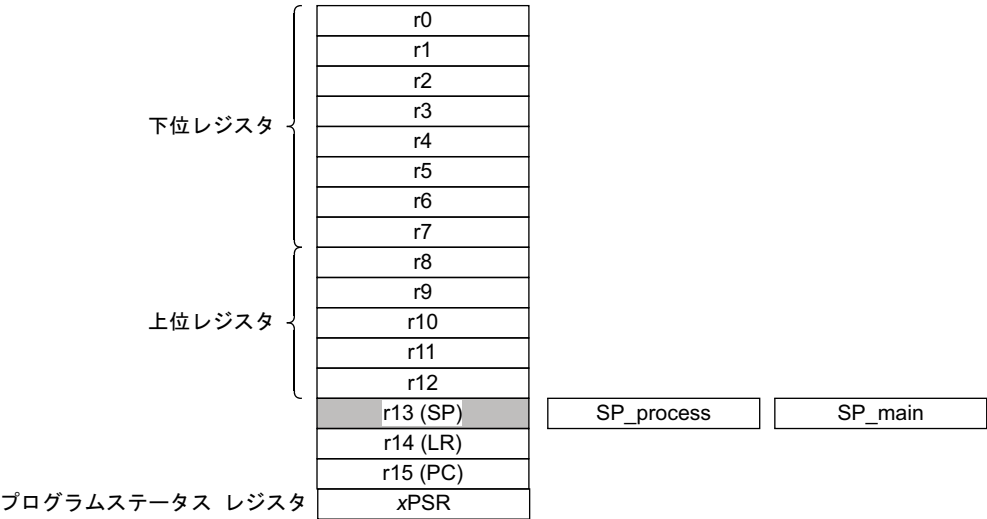


図 2-1 プロセッサのレジスタセット

2.3.1 汎用レジスタ

汎用レジスタ r0 ~ r12 には、アーキテクチャで定義された特別な使用方法はありません。汎用レジスタを指定する大部分の命令は、r0 ~ r12 を使用できます。

**下位レジスタ**      レジスタ r0 ~ r7 は、下位レジスタといい、汎用レジスタを指定するすべての命令によってアクセス可能です。

**上位レジスタ**      レジスタ r8 ~ r12 は、上位レジスタといい、汎用レジスタを指定するすべての 32 ビット命令によってアクセス可能です。  
レジスタ r8 ~ r12 は、16 ビット命令ではアクセスできません。

レジスタ r13、r14、r15 には、次のような特別な機能があります。

- スタックポインタ** レジスタ r13 は、スタックポインタ (SP) として使用されます。SP はビット [1:0] への書き込みを無視するため、4 バイト境界のワードに自動的にアライン (整列) されます。  
ハンドラモードでは常時 SP\_main が使用されます。スレッドモードでは SP\_main または SP\_process のどちらでも使用できるように構成できます。
- リンクレジスタ** レジスタ r14 は、サブルーチンとリンクするためのリンクレジスタ (LR) です。  
LR は、リンク付き分岐 (BL) またはリンク付き分岐と状態遷移 (BLX) 命令が実行されると、PC から復帰アドレスを受け取ります。  
LR は例外復帰にも使用されます。  
それ以外の場合は、r14 を汎用レジスタとして使用できます。
- プログラムカウンタ** レジスタ r15 はプログラムカウンタ (PC) です。  
ビット [0] は常時 0 のため、命令はワードまたはハーフワード境界にアラインされます。

### 2.3.2 専用プログラムステータス レジスタ (xPSR)

xPSR に関して、システムレベルのプロセッサステータスは、次の 3 つのカテゴリに分けられます。

- P.2-5 「アプリケーション PSR」
- P.2-6 「割り込み PSR」
- P.2-7 「実行 PSR」

これらは、個別のレジスタ、3 つのうちの任意の 2 つの組み合わせ、または 3 つすべての組み合わせとして、ステータスからレジスタへの移動 (MRS) 命令および MSR 命令を使用してアクセスできます。

#### アプリケーション PSR

アプリケーション PSR (APSR) には、条件コードフラグが含まれています。例外処理に入る前に、プロセッサがスタック上に条件コードフラグを保存します。MSR(2) および MRS(2) 命令を使用して APSR にアクセスできます。

APSR のフィールドを、P.2-6 図 2-2 に示します。

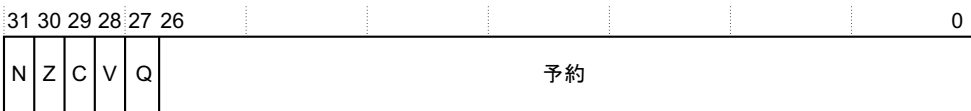


図 2-2 アプリケーションプログラム ステータスレジスタのビット割り当て

APSR のフィールドの説明を、表 2-1 に示します。

表 2-1 アプリケーションプログラム ステータスレジスタのビット割り当て

フィールド	名前	定義
[31]	N	負または、より小さいフラグ 1 = 結果が負または、より小さい 0 = 結果が正または、より大きい
[30]	Z	ゼロフラグ 1 = 結果が 0 0 = 結果が 0 でない
[29]	C	キャリー / ボローフラグ 1 = キャリーまたはボロー 0 = キャリーおよびボローなし
[28]	V	オーバフローフラグ 1 = オーバフロー 0 = オーバフローなし
[27]	Q	スティッキー飽和フラグ
[26:0]	-	予約

割り込み PSR

割り込み PSR (IPSR) には、現在アクティブになっている例外の割り込み処理ルーチン (ISR) 番号が含まれています。

IPSR のフィールドを、図 2-2 に示します。

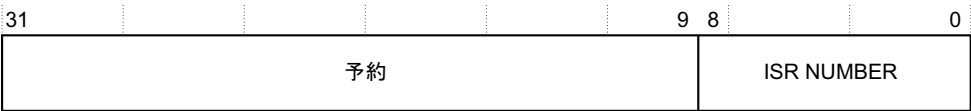


図 2-3 割り込みプログラム ステータスレジスタのビット割り当て

IPSR のフィールドの説明を、表 2-2 に示します。

表 2-2 割り込みプログラム ステータスレジスタのビット割り当て

フィールド	名前	定義
[31:9]	-	予約
[8:0]	ISR NUMBER	横取りされる例外の番号 ベースレベル = 0 NMI = 2 SVCall = 11 INTISR[0] = 16 INTISR[1] = 17 . . . INTISR[15] = 31 . . . INTISR[239] = 255

実行 PSR

実行 PSR(EPSR) には、次の重複する 2 つのフィールドが含まれています。

- 割り込みされた複数ロードおよび複数ストア命令用の **中断可能で中断後から継続可能な命令 (ICI)**(Interruptible-Continuable Instruction) フィールド
- **If-Then(IT)** 命令用の実行状態フィールド、および **Thumb 状態ビット (T-bit)**.

**中断可能で中断後から継続可能な命令フィールド**

複数ロード(LDM) 操作および複数ストア(STM) 操作は割り込みで中断可能です。EPSR の ICI フィールドには、中断が発生した箇所から複数ロードまたは複数ストアを継続するために必要な情報が保持されています。

**If-then 状態フィールド**

EPSR の IT フィールドには、If-Then 命令用の実行状態ビットが含まれています。

——— 注 ———

ICI フィールドと IT フィールドは重複しているため、If-Then ブロック内の複数ロードまたは複数ストアは中断継続可能ではありません。

EPSR のフィールドを、図 2-4 に示します。

31	27	26	25	24	23	16	15	10	9	0
予約		ICI/IT	T	予約			ICI/IT		予約	

図 2-4 実行プログラムステータスレジスタ

EPSR には直接アクセスできません。EPSR を変更できるのは、次の 2 つのイベントです。

- LDM または STM 命令中に発生する割り込み
- If-Then 命令の実行

EPSR のフィールドの説明を、表 2-3 に示します。

表 2-3 EPSR のビットの機能

フィールド	名前	定義
[31:27]	-	予約
[26:25], [15:10]	ICI	中断可能で中断後から継続可能な命令ビット。LDM または STM 転送中に割り込みが発生すると、複数転送は一時的に中止されます。EPSR はビット [15:12] を使用して、複数転送のオペランドになっているレジスタの次の番号を記憶します。割り込み処理の後、プロセッサは [15:12] によって示されるレジスタに復帰して複数転送を再開します。ICI フィールドが命令のレジスタ一覧に含まれていないレジスタを示している場合は、プロセッサはリストに現に存在する次のレジスタを使用して転送を継続します。
[26:25], [15:10]	IT	If-Then ビット。これらは If-Then 命令の実行状態ビットです。ここには、If-Then ブロック内の命令数および実行条件が含まれています。
[24]	T	T ビットは、書き込まれた PC のビット [0] が 0 であるインタワーキング命令を使用してクリアすることができます。スタックされた T ビットが 0 である例外のスタックから復元することによってもクリアできます。 T ビットがクリアされている間に命令を実行すると、INVSTATE 例外が発生します。
[23:16]	-	予約
[9:0]	-	予約



### **LDM および STM 転送中のベースレジスタ更新**

次のような場合に、LDM または STM によりベースレジスタが更新されます。

- 命令によりベースレジスタのライトバックが指定された場合。この場合、ベースレジスタは更新されたアドレスに変更されます。アボートがあると、元のベース値に復元されます。
- LDM のレジスタリストにベースレジスタがあるが、リストの最後のレジスタではない場合。この場合、ベースレジスタはロードされた値に変更されます。

次のような場合は、LDM/STM は継続されずに始めから再開されます。

- LDM/STM でフォールトが発生した場合
- LDM/STM が IT の内部にある場合

LDM がベースのロードを完了した後でも、転送はベースのロード前の状態から継続されます。

### **xPSR ビットの保存**

例外処理が開始されると、プロセッサにより 3 つのステータスレジスタを結合した情報がスタックに保存されます。

## 2.4 データタイプ

プロセッサは次のデータタイプをサポートしています。

- 32 ビットワード
- 16 ビットハーフワード
- 8 ビットバイト

---

### 注

---

メモリシステムは、すべてのデータタイプをサポートする必要があります。特に、システムはワード内で隣接するバイトを破損しないように、ワードより小さい書き込みをサポートする必要があります。

---

## 2.5 メモリフォーマット

プロセッサはメモリを、線形に配列されていて 0 から昇順に番号が付けられたバイトの集合として参照します。例えば、次のとおりです。

- バイト 0 ～ 3 は最初にストアされるワードです。
- バイト 4 ～ 7 は 2 番目にストアされるワードです。

プロセッサは、データには、リトルエンディアン形式またはビッグエンディアン形式でメモリ上のワードにアクセスできます。コードには常時リトルエンディアンでアクセスします。

---

### 注

ARM プロセッサでは、リトルエンディアンがデフォルトのメモリフォーマットです。

リトルエンディアン形式では、ワード内の最も下のアドレスを持つバイトはそのワードの最下位のバイトです。ワード内の最も上のアドレスを持つバイトは最上位です。メモリシステムのアドレス 0 のバイトは、データ線の 7 ～ 0 に接続されます。

ビッグエンディアン形式では、ワード内の最も下のアドレスを持つバイトはそのワードの最上位のバイトです。ワード内の最も上のアドレスを持つバイトは最下位です。メモリシステムのアドレス 0 のバイトは、データ線の 31 ～ 24 に接続されます。

P.2-12 図 2-5 に、リトルエンディアンのメモリフォーマットとビッグエンディアンのメモリフォーマットとの違いを示します。

プロセッサには構成設定ピン **BIGEND** があり、それを使用するとリトルエンディアンまたは BE-8 のビッグエンディアン形式のいずれかが選択できます。この構成設定ピンの状態は、リセット時に取り込まれます。リセットの後ではエンディアンを変更できません。

---

### 注

- システム制御空間 (SCS) へのアクセスは常時リトルエンディアンです。
- リセット以外でエンディアンを変更しようとしても無視されます。
- 専用ペリフェラルバス (PPB) 空間は **BIGEND** の設定に関係なくリトルエンディアンです。

リトルエンディアンのデータフォーマット



ビッグエンディアンのデータフォーマット

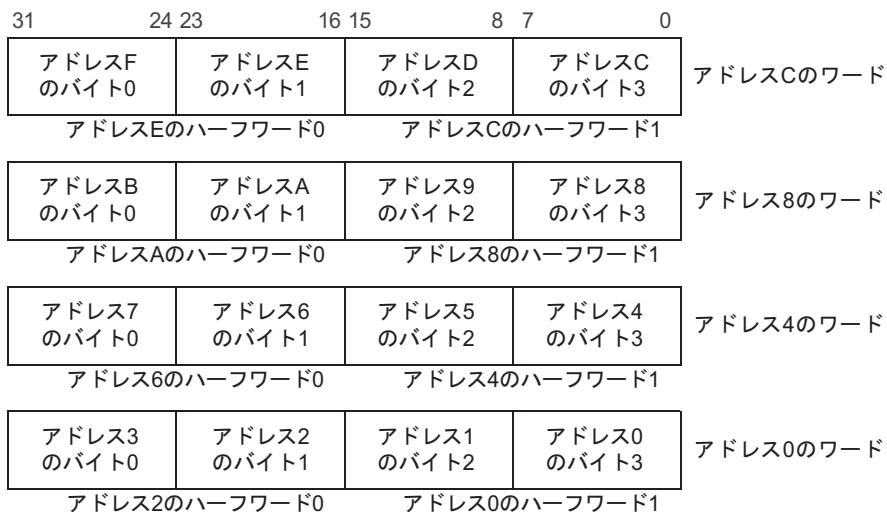


図 2-5 リトルエンディアンおよびビッグエンディアンのメモリフォーマット

## 2.6 命令セット

このプロセッサは ARM 命令をサポートしていません。

プロセッサは表 2-4に一覧表記されている命令以外のすべてのARMv6 Thumb 命令をサポートしています。

### 表 2-4 サポートされていない Thumb 命令

命令		実行時の動作
BLX(1)	リンク付き分岐と状態遷移	BLX(1) は常時フォールトが発生します。
SETEND	エンディアン形式の設定	SETEND は常時フォールトが発生します。構成設定ピンにより、Cortex-M3 のエンディアン方式が選択されます。

プロセッサは、表 2-5 に一覧表記されている Thumb-2 命令をサポートしています。

表 2-5 サポートされている Thumb-2 命令

命令の種類	サイズ	命令
データ処理	16	ADC、ADD、AND、ASR、BIC、CMN、CMP、CPY、EOR、LSL、LSR、MOV、MUL、MVN、NEG、ORR、ROR、SBC、SUB、TST、REV、REVH、REVSH、SXTB、SXTH、UXTB と UXTH。
分岐	16	B<cond>、B、BL、BX、および BLX。即値を使用する BLX は存在しないことに注意して下さい。
単一ロード/ストア	16	LDR、LDRB、LDRH、LDRSB、LDRSH、STR、STRB、STRH。
複数ロード/ストア	16	LDMIA、POP、PUSH と STMIA。
例外生成	16	BKPT は、デバッグ可の場合はデバッグ内で停止し、デバッグ不可の場合はフォールトが発生します。SVC は SVCcall ハンドラを呼び出すフォールトが発生します。
即値を使用するデータ処理	32	ADC{S}、ADD{S}、CMN、RSB{S}、SBC{S}、SUB{S}、CMP、AND{S}、TST、BIC{S}、EOR{S}、TEQ、ORR{S}、MOV{S}、ORN{S} と MVN{S}。
大きな即値を使用するデータ処理	32	MOVW、MOVT、ADDW と SUBW。 MOVW および MOVT は 16 ビット即値を持ちます。これは、メモリからのリテラルロードを置き換え可能であることを意味します。 ADDW および SUBW は 12 ビット即値を持ちます。これは、メモリからのリテラルロードの多くを置き換え可能であることを意味します。

表 2-5 サポートされている Thumb-2 命令

命令の種類	サイズ	命令
ビットフィールド操作	32	BFI、BFC、UBFX、SBFX これらはビット単位の操作で、ビットにおける位置と大きさの制御が可能です。これらによって C/C++ をサポートし、構造体のビットフィールド、多くの比較やいくつかの AND/OR の代入文で使えます。
3 つのレジスタを使用するデータ処理	32	ADC{S}、ADD{S}、CMN、RSB{S}、SBC{S}、SUB{S}、CMP、AND{S}、TST、BIC{S}、EOR{S}、TEQ、ORR{S}、MOV{S}、ORN{S}、MVN{S}PKxxx 命令は含まれません。
シフト操作	32	ASR{S}、LSL{S}、LSR{S}、RRX {S} と ROR {S}。
その他の命令	32	REV、RE VH、REVSH、RBIT、CLZ、SXTB、SXTH、UXTB と UXTH。 拡張命令は、対応する v6 の 16 ビット命令と同じです。
テーブル分岐	32	switch/case 文用の TBB および TBH のテーブル分岐。これらはシフトしてレジスタにロード (LDR) した後に分岐です。
乗算	32	MUL、ML と MLS。
結果が 64 ビットの乗算	32	UMULL、SMULL、UMLAL と SMLAL。
ロード/ストアのアドレス方式	32	アドレス方式のフォーマットは、PC +/- imm12、Rbase + imm12、Rbase +/- imm8、およびシフトなどのレジスタの調整をサポートしています。 命令に T が付くバリエーションは特権モードで使用されます。
単一ロード/ストア	32	LDR、LDRB、LDRSB、LDRH、LDRSH、STR、STRB、STRH、および T が付くバリエーション。PLD および PLI は両方ともヒント命令なので、NOP として動作します。
複数ロード/ストア	32	STM、LDM、LDRD と STRD。
排他ロード/ストア	32	LDREX、STREX、LDREXB、LDREXH、STREXB、STREXH、CLREX。 ローカルモニタがない場合はフォールトが発生します。これは実装定義です。 LDREXD および STREXD は、このプロファイルに含まれていません。
分岐	32	B、BL、B<cond>BLX(1) は必ず ARM 状態へ遷移することになるので、存在しません。BXJ はありません。

表 2-5 サポートされている Thumb-2 命令

命令の種類	サイズ	命令
システム	32	MSR(2) および MRS(2) は MSR/MRS を置き換えて、より高度な動作をします。これらは、他のスタックおよびステータスレジスタにもアクセスします。 CPSIE/CPSID の 32 ビット形式はサポートされていません。 RFE および SRS はありません。
システム	16	CPSIE および CPSID は MSR(2) 命令の短縮バージョンで、標準 Thumb-2 エンコーディングを使用します。しかし a はなく、i (PRIMASK) および f (FAULTMASK) の使用のみを許可します。(訳注) : a については ARMv6 の Current PSR 参照
32 ビット拡張	32	NOP (すべての形式)、コプロセッサ (MCR、MCR2、MCRR、MRC、MRC2、MRRC)、および YIELD (ヒント NOP)。注 : MRS(1)、MSR(1)、SUBS (PC 復帰リンク) は含まれません。
組み合わせ分岐	16	CBZ および CBNZ (比較してレジスタが 0 または 0 でないかで分岐)
拡張	16	IT および NOP。YIELD を含みます。
除算	32	SDIV および UDIV。32/32 ビットの除算で、それぞれ符号付きおよび符号なしです。結果は 32 ビットの商です。余りはありませんが、引き算で求めることができます。早期終了が許可されています。
スリープ	16、32	WFI、WFE、SEV は、ヒントのある NOP 命令のクラスにはいり、スリープ動作を制御します。
バリア	32	ISB、DSB、DMB はバリア命令で、次の命令が実行される前に、確実にある動作が完了していることを保証します。
飽和	32	SSAT および USAT は、レジスタで飽和を実行します。次の項目を実行します。 ある選択したビット位置、Q 値からのオーバーフローをシフトでテストしながら値を正規化する。 オーバーフローが検出された場合は、xPSR の Q ビットをセットして値を飽和。 飽和とは、選択されたサイズに対して、最大の符号なしの値、または最大 / 最小の符号付きの値を指します。

## 注

すべてのコプロセッサ命令は、コプロセッサなし (NOCP) フォールトを生成します。





## 第 3 章

# システム制御

本章では、プロセッサをプログラムするレジスタについて説明します。本章は以下のセクションから構成されています。

- プロセッサレジスタの概要: P.3-2

3.1 プロセッサレジスタの概要

このセクションでは、機能を制御するレジスタについて説明します。このセクションは以下の項目から構成されています。

- ネスト型ベクタ割り込みコントローラレジスタ : P.3-2
- コアデバッグレジスタ : P.3-5
- システムデバッグレジスタ : P.3-6
- デバッグインタフェースポートレジスタ : P.3-11
- メモリ保護ユニットレジスタ : P.3-11
- トレースポートインタフェースユニットレジスタ : P.3-12
- エンベデッドトレースマクロセルレジスタ : P.3-13

3.1.1 ネスト型ベクタ割り込みコントローラレジスタ

ネスト型ベクタ割り込みコントローラ (NVIC) レジスタの概要を表 3-1 に示します。NVIC レジスタの詳細については、第 8 章「ネスト型ベクタ割り込みコントローラ」を参照して下さい。

表 3-1 NVIC レジスタ

レジスタ名	タイプ	アドレス	リセット時の値
割り込み制御タイプレジスタ	読み出し専用	0xE000E004	a
SysTick 制御およびステータスレジスタ	読み出し / 書き込み	0xE000E010	0x00000000
SysTick リロード値レジスタ	読み出し / 書き込み	0xE000E014	Unpredictable
SysTick 現在値レジスタ	読み出し / 書き込みクリア	0xE000E018	Unpredictable
SysTick 較正值レジスタ	読み出し専用	0xE000E01C	STCALIB
割り込み要求 (IRQ)0 ~ 31 イネーブルセットレジスタ	読み出し / 書き込み	0xE000E100	0x00000000
.	.	.	.
.	.	.	.
.	.	.	.
割り込み要求 (IRQ)224 ~ 239 イネーブルセットレジスタ	読み出し / 書き込み	0xE000E11C	0x00000000
割り込み要求 (IRQ)0 ~ 31 イネーブルクリアレジスタ	読み出し / 書き込み	0xE000E180	0x00000000
.	.	.	.
.	.	.	.
.	.	.	.

表 3-1 NVIC レジスタ

レジスタ名	タイプ	アドレス	リセット時の値
割り込み要求 (IRQ)224 ～ 239 イネーブルクリア レジスタ	読み出し / 書き込み	0xE000E19C	0x00000000
割り込み要求 (IRQ)0 ～ 31 保留セットレジスタ	読み出し / 書き込み	0xE000E200	0x00000000
.	.	.	.
.	.	.	.
.	.	.	.
割り込み要求 (IRQ)224 ～ 239 保留セットレジスタ	読み出し / 書き込み	0xE000E21C	0x00000000
割り込み要求 (IRQ)0 ～ 31 保留クリアレジスタ	読み出し / 書き込み	0xE000E280	0x00000000
.	.	.	.
.	.	.	.
.	.	.	.
割り込み要求 (IRQ)224 ～ 239 保留クリアレジスタ	読み出し / 書き込み	0xE000E29C	0x00000000
割り込み要求 (IRQ)0 ～ 31 アクティブビット レジスタ	読み出し専用	0xE000E300	0x00000000
.	.	.	.
.	.	.	.
.	.	.	.
割り込み要求 (IRQ)224 ～ 239 アクティブビット レジスタ	読み出し専用	0xE000E31C	0x00000000
割り込み要求 (IRQ)0 ～ 31 優先度レジスタ	読み出し / 書き込み	0xE000E400	0x00000000
.	.	.	.
.	.	.	.
.	.	.	.
通常割り込み要求 (IRQ)236 ～ 239 優先度レジスタ	読み出し / 書き込み	0xE000E4F0	0x00000000
CPUID ベースレジスタ	読み出し専用	0xE000ED00	0x411FC231
割り込み制御状態レジスタ	読み出し / 書き込み、または読み出し専用	0xE000ED04	0x00000000

表 3-1 NVIC レジスタ

レジスタ名	タイプ	アドレス	リセット時の値
ベクタテーブルオフセット レジスタ	読み出し / 書き込み	0xE000ED08	0x00000000
アプリケーション割り込み / リセット制御レジスタ	読み出し / 書き込み	0xE000ED0C	0x00000000
システム制御レジスタ	読み出し / 書き込み	0xE000ED10	0x00000000
構成制御レジスタ	読み出し / 書き込み	0xE000ED14	0x00000000
システムハンドラ 4 ~ 7 優先度レジスタ	読み出し / 書き込み	0xE000ED18	0x00000000
システムハンドラ 8 ~ 11 優先度レジスタ	読み出し / 書き込み	0xE000ED1C	0x00000000
システムハンドラ 12 ~ 15 優先度レジスタ	読み出し / 書き込み	0xE000ED20	0x00000000
システムハンドラ制御および状態レジスタ	読み出し / 書き込み	0xE000ED24	0x00000000
構成可能フォールトステータス レジスタ	読み出し / 書き込み	0xE000ED28	0x00000000
ハードフォールトステータス レジスタ	読み出し / 書き込み	0xE000ED2C	0x00000000
デバッグフォールトステータス レジスタ	読み出し / 書き込み	0xE000ED30	0x00000000
メモリ管理アドレスレジスタ	読み出し / 書き込み	0xE000ED34	予測不能
バスフォールトアドレス レジスタ	読み出し / 書き込み	0xE000ED38	予測不能
補助フォールトステータス レジスタ	読み出し / 書き込み	0xE000ED3C	0x00000000
PFR0 : プロセッサ機能レジスタ 0	読み出し専用	0xE000ED40	0x00000030
PFR1 : プロセッサ機能レジスタ 1	読み出し専用	0xE000ED44	0x00000200
DFR0 : デバッグ機能レジスタ 0	読み出し専用	0xE000ED48	0x00100000
AFR0 : 補助機能レジスタ 0	読み出し専用	0xE000ED4C	0x00000000
MMFR0 : メモリモデル機能レジスタ 0	読み出し専用	0xE000ED50	0x00000030
MMFR1 : メモリモデル機能レジスタ 1	読み出し専用	0xE000ED54	0x00000000
MMFR2 : メモリモデル機能レジスタ 2	読み出し専用	0xE000ED58	0x00000000
MMFR3 : メモリモデル機能レジスタ 3	読み出し専用	0xE000ED5C	0x00000000

表 3-1 NVIC レジスタ

レジスタ名	タイプ	アドレス	リセット時の値
ISAR0 : ISA 機能レジスタ 0	読み出し専用	0xE000ED60	0x01141110
ISAR1 : ISA 機能レジスタ 1	読み出し専用	0xE000ED64	0x02111000
ISAR2 : ISA 機能レジスタ 2	読み出し専用	0xE000ED68	0x21112231
ISAR3 : ISA 機能レジスタ 3	読み出し専用	0xE000ED6C	0x01111110
ISAR4 : ISA 機能レジスタ 4	読み出し専用	0xE000ED70	0x01310102
ソフトウェアトリガ割り込みレジスタ	書き込み専用	0xE000EF00	-
ペリフェラル識別レジスタ (PID4)	読み出し専用	0xE000EFD0	0x04
ペリフェラル識別レジスタ (PID5)	読み出し専用	0xE000EFD4	0x00
ペリフェラル識別レジスタ (PID6)	読み出し専用	0xE000EFD8	0x00
ペリフェラル識別レジスタ (PID7)	読み出し専用	0xE000EFD C	0x00
ペリフェラル識別レジスタビット [7:0] (PID0)	読み出し専用	0xE000EFE0	0x00
ペリフェラル識別レジスタビット [15:8] (PID1)	読み出し専用	0xE000EFE4	0xB0
ペリフェラル識別レジスタビット [23:16] (PID2)	読み出し専用	0xE000EFE8	0x1B
ペリフェラル識別レジスタビット [31:24] (PID3)	読み出し専用	0xE000EFEC	0x00
コンポーネント識別レジスタビット [7:0] (CID0)	読み出し専用	0xE000EFF0	0x0D
コンポーネント識別レジスタビット [15:8] (CID1)	読み出し専用	0xE000EFF4	0xE0
コンポーネント識別レジスタビット [23:16] (CID2)	読み出し専用	0xE000EFF8	0x05
コンポーネント識別レジスタビット [31:24] (CID3)	読み出し専用	0xE000EFFC	0xB1

a. リセット時の値は、定義されている割り込みの数によって異なります。

### 3.1.2 コアデバッグ レジスタ

コアデバッグ レジスタの概要を表 3-2 に示します。これらのレジスタの詳細については、第 10 章 「コアデバッグ」を参照して下さい。

表 3-2 コアデバッグ レジスタ

レジスタ名	タイプ	アドレス	リセット時の値
デバッグホールト制御およびステータスレジスタ	読み出し / 書き込み	0xE000EDF0	0x00000000 <sup>a</sup>

表 3-2 コアデバッグレジスタ

レジスタ名	タイプ	アドレス	リセット時の値
デバッグコアレジスタ セレクタレジスタ	書き込み専用	0xE000EDF4	-
デバッグコアレジスタ データレジスタ	読み出し / 書き込み	0xE000EDF8	-
デバッグ例外およびモニタ制御レジスタ	読み出し / 書き込み	0xE000EDFC	0x00000000 <sup>b</sup>

- a. ビット [5]、[3]、[2]、[1]、[0] は、**PORESETn** によってリセットされます。ビット [1] は **SYSRESETn** によって、およびアプリケーション割り込みおよびリセット制御レジスタの **VECTRESET** ビットに 1 を書き込むことによってリセットされます。
- b. ビット [16]、[17]、[18]、[19] は **SYSRESETn** によって、およびアプリケーション割り込みおよびリセット制御レジスタの **VECTRESET** ビットに 1 を書き込むことによってリセットされます。

### 3.1.3 システムデバッグレジスタ

このセクションでは、システムデバッグレジスタの一覧を示します。

#### フラッシュパッチおよびブレイクポイントレジスタ

表 3-3 フラッシュパッチおよびブレイクポイント(FPB)レジスタの概要を表 3-3に示します。FPB レジスタの詳細については、第 11 章「システムデバッグ」を参照して下さい。

表 3-3 フラッシュパッチレジスタの概要

名前	タイプ	アドレス	リセット時の値	説明
FP_CTRL	読み出し / 書き込み	0xE0002000	ビット [0] は 1'b0 にリセットされます。	フラッシュパッチ制御レジスタ
FP_REMAP	読み出し / 書き込み	0xE0002004	-	フラッシュパッチリマップレジスタ
FP_COMP0	読み出し / 書き込み	0xE0002008	ビット [0] は 1'b0 にリセットされます。	フラッシュパッチコンパレータレジスタ
FP_COMP1	読み出し / 書き込み	0xE000200C	ビット [0] は 1'b0 にリセットされます。	フラッシュパッチコンパレータレジスタ
FP_COMP2	読み出し / 書き込み	0xE0002010	ビット [0] は 1'b0 にリセットされます。	フラッシュパッチコンパレータレジスタ
FP_COMP3	読み出し / 書き込み	0xE0002014	ビット [0] は 1'b0 にリセットされます。	フラッシュパッチコンパレータレジスタ
FP_COMP4	読み出し / 書き込み	0xE0002018	ビット [0] は 1'b0 にリセットされます。	フラッシュパッチコンパレータレジスタ
FP_COMP5	読み出し / 書き込み	0xE000201C	ビット [0] は 1'b0 にリセットされます。	フラッシュパッチコンパレータレジスタ

表 3-3 フラッシュパッチ レジスタの概要

名前	タイプ	アドレス	リセット時の値	説明
FP_COMP6	読み出し / 書き込み	0xE0002020	ビット [0] は 1'b0 にリセットされます。	フラッシュパッチ コンパレータレジスタ
FP_COMP7	読み出し / 書き込み	0xE0002024	ビット [0] は 1'b0 にリセットされます。	フラッシュパッチ コンパレータレジスタ
PID4	読み出し専用	0xE0002FD0	-	値 0x04
PID5	読み出し専用	0xE0002FD4	-	値 0x00
PID6	読み出し専用	0xE0002FD8	-	値 0x00
PID7	読み出し専用	0xE0002FDC	-	値 0x00
PID0	読み出し専用	0xE0002FE0	-	値 0x03
PID1	読み出し専用	0xE0002FE4	-	値 0xB0
PID2	読み出し専用	0xE0002FE8	-	値 0x0B
PID3	読み出し専用	0xE0002FEC	-	値 0x00
CID0	読み出し専用	0xE0002FF0	-	値 0x0D
CID1	読み出し専用	0xE0002FF4	-	値 0xE0
CID2	読み出し専用	0xE0002FF8	-	値 0x05
CID3	読み出し専用	0xE0002FFC	-	値 0xB1

### データウォッチポイントおよびトレースレジスタ

データウォッチポイントおよびトレース (DWT) レジスタの概要を、表 3-4 に示します。DWT レジスタの詳細については、第 11 章 「システムデバッグ」を参照して下さい。

表 3-4 DWT レジスタの概要

名前	タイプ	アドレス	リセット時の値	説明
DWT_CTRL	読み出し / 書き込み	0xE0001000	0x00000000	DWT 制御レジスタ
DWT_CYCCNT	読み出し / 書き込み	0xE0001004	0x00000000	DWT カレント PC サンプラサイクルカウンタ レジスタ
DWT_CPICNT	読み出し / 書き込み	0xE0001008	-	DWT カレント CPI カウントレジスタ
DWT_EXCCNT	読み出し / 書き込み	0xE000100C	-	DWT カレント割り込みオーバーヘッドカウンタ レジスタ
DWT_SLEEPCNT	読み出し / 書き込み	0xE0001010	-	DWT カレントスリープカウンタ レジスタ

表 3-4 DWT レジスタの概要

名前	タイプ	アドレス	リセット 時の値	説明
DWT_LSUCNT	読み出し / 書き込み	0xE0001014	-	DWT カレント LSU カウントレジスタ
DWT_FOLDCNT	読み出し / 書き込み	0xE0001018	-	DWT カレントフォールドカウント レジスタ
DWT_PCSR	読み出し / 書き込み	0xE000101C	-	DWT PC サンプルレジスタ
DWT_COMP0	読み出し / 書き込み	0xE0001020	-	DWT コンパレータレジスタ
DWT_MASK0	読み出し / 書き込み	0xE0001024	-	DWT マスクレジスタ
DWT_FUNCTION0	読み出し / 書き込み	0xE0001028	0x00000000	DWT 機能レジスタ
DWT_COMP1	読み出し / 書き込み	0xE0001030	-	DWT コンパレータレジスタ
DWT_MASK1	読み出し / 書き込み	0xE0001034	-	DWT マスクレジスタ
DWT_FUNCTION1	読み出し / 書き込み	0xE0001038	0x00000000	DWT 機能レジスタ
DWT_COMP2	読み出し / 書き込み	0xE0001040	-	DWT コンパレータレジスタ
DWT_MASK2	読み出し / 書き込み	0xE0001044	-	DWT マスクレジスタ
DWT_FUNCTION2	読み出し / 書き込み	0xE0001048	0x00000000	DWT 機能レジスタ
DWT_COMP3	読み出し / 書き込み	0xE0001050	-	DWT コンパレータレジスタ
DWT_MASK3	読み出し / 書き込み	0xE0001054	-	DWT マスクレジスタ
DWT_FUNCTION3	読み出し / 書き込み	0xE0001058	0x00000000	DWT 機能レジスタ
PID4	読み出し専用	0xE0001FD0	0x04	値 0x04
PID5	読み出し専用	0xE0001FD4	0x00	値 0x00
PID6	読み出し専用	0xE0001FD8	0x00	値 0x00
PID7	読み出し専用	0xE0001FDC	0x00	値 0x00
PID0	読み出し専用	0xE0001FE0	0x02	値 0x02
PID1	読み出し専用	0xE0001FE4	0xB0	値 0xB0
PID2	読み出し専用	0xE0001FE8	0x0B0	値 0x1B
PID3	読み出し専用	0xE0001FEC	0x00	値 0x00
CID0	読み出し専用	0xE0001FF0	0x0D	値 0x0D



表 3-4 DWT レジスタの概要

名前	タイプ	アドレス	リセット時の値	説明
CID1	読み出し専用	0xE0001FF4	0xE0	値 0xE0
CID2	読み出し専用	0xE0001FF8	0x05	値 0x05
CID3	読み出し専用	0xE0001FFC	0xB1	値 0xB1

### 計装トレースマクロセル レジスタ

計装トレースマクロセル (ITM) レジスタの概要を、表 3-5 に示します。ITM レジスタの詳細については、第 11 章 「システムデバッグ」を参照して下さい。

表 3-5 ITM レジスタの概要

名前	タイプ	アドレス	リセット時の値
スティムラサポート 0 ～ 31	読み出し / 書き込み	0xE0000000-0xE000007C	-
トレースイネーブル	読み出し / 書き込み	0xE0000E00	0x00000000
トレース特権	読み出し / 書き込み	0xE0000E40	0x00000000
トレース制御レジスタ	読み出し / 書き込み	0xE0000E80	0x00000000
統合書き込み	書き込み専用	0xE0000EF8	0x00000000
統合読み出し	読み出し専用	0xE0000EFC	0x00000000
統合モード制御	読み出し / 書き込み	0xE0000F00	0x00000000
ロックアクセス レジスタ	書き込み専用	0xE0000FB0	0x00000000
ロックステータス レジスタ	読み出し専用	0xE0000FB4	0x00000003
PID4	読み出し専用	0xE0000FD0	0x00000004
PID5	読み出し専用	0xE0000FD4	0x00000000
PID6	読み出し専用	0xE0000FD8	0x00000000
PID7	読み出し専用	0xE0000FDC	0x00000000
PID0	読み出し専用	0xE0000FE0	0x00000001
PID1	読み出し専用	0xE0000FE4	0x000000B0
PID2	読み出し専用	0xE0000FE8	0x0000001B

表 3-5 ITM レジスタの概要

名前	タイプ	アドレス	リセット時の値
PID3	読み出し専用	0xE000FEC	0x00000000
CID0	読み出し専用	0xE000FF0	0x0000000D
CID1	読み出し専用	0xE000FF4	0x000000E0
CID2	読み出し専用	0xE000FF8	0x00000005
CID3	読み出し専用	0xE000FFC	0x000000B1

### アドバンストハイパフォーマンスバス アクセスポート レジスタ

アドバンストハイパフォーマンスバス アクセスポート (AHB-AP) レジスタの概要を表 3-6 に示します。AHB-AP レジスタの詳細については、第 11 章「システムデバッグ」を参照して下さい。

表 3-6 AHB-AP レジスタの概要

名前	タイプ	アドレス	リセット時の値
制御およびステータスワード	読み出し / 書き込み	0x00	レジスタ参照
転送アドレス	読み出し / 書き込み	0x04	0x00000000
データ読み出し / 書き込み	読み出し / 書き込み	0x0C	-
バンクデータ 0	読み出し / 書き込み	0x10	-
バンクデータ 1	読み出し / 書き込み	0x14	-
バンクデータ 2	読み出し / 書き込み	0x18	-
バンクデータ 3	読み出し / 書き込み	0x1C	-
デバッグ ROM アドレス	読み出し専用	0xF8	0xE000E000
ID レジスタ	読み出し専用	0xFC	0x14770011

### 3.1.4 デバッグインタフェースポート レジスタ

デバッグインタフェースポート レジスタの概要を、表 3-7 に示します。デバッグインタフェースポート レジスタの詳細については、第 12 章 「デバッグポート」を参照して下さい。

表 3-7 デバッグインタフェースポート レジスタの概要

名前	SWJ-DP	SW-DP	説明
ABORT	はい	はい	アボートレジスタ
IDCODE	はい	はい	ID コードレジスタ
CTRL/STAT	はい	はい	制御 / ステータスレジスタ
SELECT	はい	はい	AP 選択レジスタ
RDBUFF	はい	はい	読み出しバッファレジスタ
WCR	いいえ	はい	ワイヤ制御レジスタ
RESEND	いいえ	はい	読み出し再送レジスタ

### 3.1.5 メモリ保護ユニットレジスタ

メモリ保護ユニット (MPU) レジスタの概要を、表 3-8 に示します。MPU レジスタの詳細については、第 9 章 「メモリ保護ユニット」を参照して下さい。

表 3-8 MPU レジスタ

名前	タイプ	アドレス	リセット時の値
MPU タイプレジスタ	読み出し専用	0xE000ED90	0x00000800
MPU 制御レジスタ	読み出し専用	0xE000ED94	0x00000000
MPU 領域番号レジスタ	読み出し専用	0xE000ED98	-
MPU 領域ベースアドレス レジスタ	読み出し専用	0xE000ED9C	-
MPU 領域属性およびサイズレジスタ	読み出し専用	0xE000EDA0	-
MPU エイリアス 1 領域ベースアドレス レジスタ	D9C のエイリアス	0xE000EDA4	-
MPU エイリアス 1 領域属性およびサイズレジスタ	DA0 のエイリアス	0xE000EDA8	-
MPU エイリアス 2 領域ベースアドレス レジスタ	D9C のエイリアス	0xE000EDAC	-

表 3-8 MPU レジスタ

名前	タイプ	アドレス	リセット時の値
MPU エイリアス 2 領域属性およびサイズレジスタ	DA0 のエイリアス	0xE000EDB0	-
MPU エイリアス 3 領域ベースアドレスレジスタ	D9C のエイリアス	0xE000EDB4	-
MPU エイリアス 3 領域属性およびサイズレジスタ	DA0 のエイリアス	0xE000EDB8	-

### 3.1.6 トレースポート インタフェースユニット レジスタ

トレースポート インタフェースユニット (TPIU) レジスタの概要を、表 3-9 に示します。TPIU レジスタの詳細については、第 13 章「トレースポート インタフェースユニット」を参照して下さい。

表 3-9 TPIU レジスタ

レジスタ名	タイプ	アドレス	リセット時の値
サポートされる同期化ポートサイズ レジスタ	読み出し専用	0xE0040000	0bxx0x
カレント同期化ポートサイズ レジスタ	読み出し / 書き込み	0xE0040004	0x01
非同期クロックプリスケアラ レジスタ	読み出し / 書き込み	0xE0040010	0x0000
選択ピンプロトコル レジスタ	読み出し / 書き込み	0xE00400F0	0x01
フォーマッタおよびフラッシュステータス レジスタ	読み出し / 書き込み	0xE0040300	0x08
フォーマッタおよびフラッシュ制御レジスタ	読み出し専用	0xE0040304	0x00 or 0x102
フォーマッタ同期化カウンタレジスタ	読み出し専用	0xE0040308	0x00
統合レジスタ : ITATBCTR2	読み出し専用	0xE0040EF0	0x0
統合レジスタ : ITATBCTR0	読み出し専用	0xE0040EF8	0x0

### 3.1.7 エンベデッドトレースマクロセル レジスタ

エンベデッドトレースマクロセル (ETM) レジスタの概要を、表 3-10 に示します。ETM レジスタの詳細については、第 15 章 「エンベデッドトレース マクロセル」を参照して下さい。

表 3-10 ETM レジスタ

名前	タイプ	アドレス	存在
ETM 制御	読み出し / 書き込み	0xE0041000	はい
コンフィギュレーションコード	読み出し専用	0xE0041004	はい
トリガイメント	書き込み専用	0xE0041008	はい
ASIC 制御	書き込み専用	0xE004100C	いいえ
ETM ステータス	読み出し専用または 読み出し / 書き込み	0xE0041010	はい
システム構成	読み出し専用	0xE0041014	はい
TraceEnable	書き込み専用	0xE0041018, 0xE004101C	いいえ
TraceEnable イベント	書き込み専用	0xE0041020	はい
TraceEnable 制御 1	書き込み専用	0xE0041024	はい
FIFOFULL 領域	書き込み専用	0xE0041028	いいえ
FIFOFULL レベル	書き込み専用または 読み出し / 書き込み	0xE004102C	はい
ViewData	書き込み専用	0xE0041030-0xE004103C	いいえ
アドレスコンパレータ	書き込み専用	0xE0041040- 0xE004113C	いいえ
カウンタ	書き込み専用	0xE0041140-0xE004157C	いいえ
シーケンサ	読み出し / 書き込み	0xE0041180- 0xE0041194, 0xE0041198	いいえ
外部出力	書き込み専用	0xE00411A0-0xE00411AC	いいえ
CID コンパレータ	書き込み専用	0xE00411B0-0xE00411BC	いいえ
実装固有	書き込み専用	0xE00411C0-0xE00411DC	いいえ
同期周波数	書き込み専用	0xE00411E0	いいえ
ETM ID	読み出し専用	0xE00411E4	はい
コンフィギュレーションコード拡張機能	読み出し専用	0xE00411E8	はい

表 3-10 ETM レジスタ

名前	タイプ	アドレス	存在
拡張外部入力セクタ	書き込み専用	0xE00411EC	いいえ
TraceEnable 開始 / 中止エンベデッド ICE	読み出し / 書き込み	0xE00411F0	はい
エンベデッド ICE 動作制御	書き込み専用	0xE00411F4	いいえ
CoreSight トレース ID	読み出し / 書き込み	0xE0041200	はい
OS 保存 / 復元	書き込み専用	0xE0041304-0xE0041308	いいえ
ITMISCIN	読み出し専用	0xE0041EE0	はい
ITTRIGOUT	書き込み専用	0xE0041EE8	はい
ITATBCTR2	読み出し専用	0xE0041EF0	はい
ITATBCTR0	書き込み専用	0xE0041EF8	はい
統合モード制御	読み出し / 書き込み	0xE0041F00	はい
クレームタグ	読み出し / 書き込み	0xE0041FA0-0xE0041FA4	はい
ロックアクセス	書き込み専用	0xE0041FB0-0xE0041FB4	はい
認証ステータス	読み出し専用	0xE0041FB8	はい
デバイスタイプ	読み出し専用	0xE0041FCC	はい
ペリフェラル ID 4	読み出し専用	0xE0041FD0	はい
ペリフェラル ID 5	読み出し専用	0xE0041FD4	はい
ペリフェラル ID 6	読み出し専用	0xE0041FD8	はい
ペリフェラル ID 7	読み出し専用	0xE0041FDC	はい
ペリフェラル ID 0	読み出し専用	0xE0041FE0	はい
ペリフェラル ID 1	読み出し専用	0xE0041FE4	はい
ペリフェラル ID 2	読み出し専用	0xE0041FE8	はい
ペリフェラル ID 3	読み出し専用	0xE0041FEC	はい
コンポーネント ID 0	読み出し専用	0xE0041FF0	はい
コンポーネント ID 1	読み出し専用	0xE0041FF4	はい
コンポーネント ID 2	読み出し専用	0xE0041FF8	はい
コンポーネント ID 3	読み出し専用	0xE0041FFC	はい

## 第 4 章

# メモリマップ

本章では、プロセッサに固定のメモリマップおよびそのビットバンド機能について説明します。本章は以下のセクションから構成されています。

- メモリマップ: P.4-2
- ビットバンド: P.4-5
- ROM メモリテーブル: P.4-7

4.1   メモリマップ

固定のメモリマップを、図 4-1 に示します。

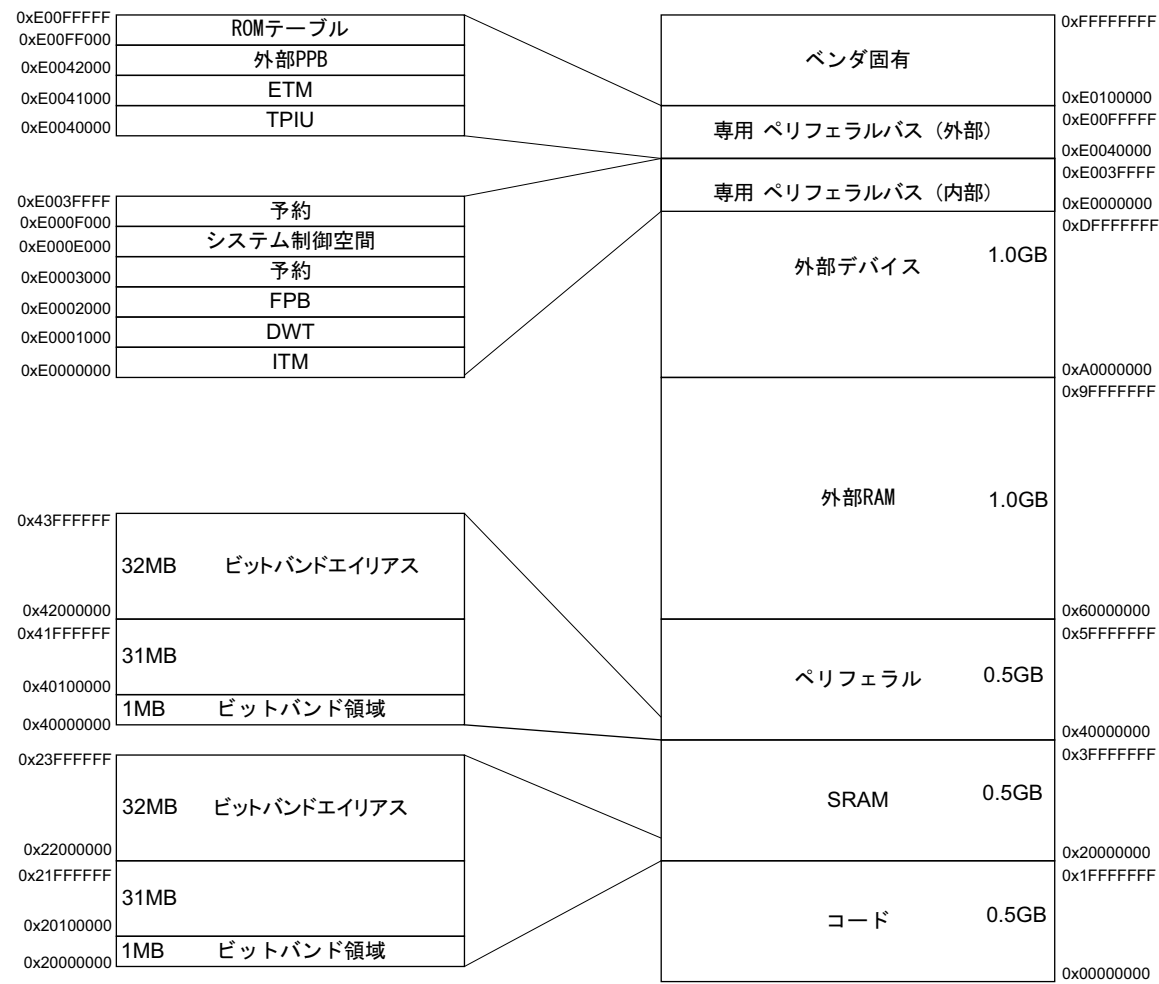


図 4-1 プロセッサのメモリマップ



各アドレスに割り当てられるメモリマップの領域におけるプロセッサのインタフェースを、表 4-1 に示します。

表 4-1 メモリインタフェース

メモリマップ	インタフェース
コード	命令フェッチは ICode バス上で実行され、データアクセスは DCode バス上で実行されます。
SRAM	命令フェッチおよびデータアクセスは、システムバス上で実行されます。
SRAM_bitband	エイリアス領域。データアクセスはエイリアスで行われます。命令アクセスにはエイリアスは使用されません。
ペリフェラル	命令フェッチおよびデータアクセスは、システムバス上で実行されます。
Periph_bitband	エイリアス領域。データアクセスはエイリアスで行われます。命令アクセスにはエイリアスは使用されません。
外部 RAM	命令フェッチおよびデータアクセスは、システムバス上で実行されます。
外部デバイス	命令フェッチおよびデータアクセスは、システムバス上で実行されます。
専用ペリフェラルバス	<p>次のアクセスは、プロセッサの内部専用ペリフェラルバス (PPB) で行われます。</p> <ul style="list-style-type: none"> <li>計装トレース マクロセル (ITM)</li> <li>ネスト型ベクタ割り込みコントローラ (NVIC)</li> <li>フラッシュパッチおよびブレイクポイント (FPB) ユニット</li> <li>データウォッチポイントおよびトレース (DWT) ユニット</li> <li>メモリ保護ユニット (MPU)</li> </ul> <p>次のアクセスは、外部 PPB インタフェース上で行われます。</p> <ul style="list-style-type: none"> <li>トレースポート インタフェースユニット (TPIU)</li> <li>エンベデッドトレース マクロセル (ETM)</li> <li>PPB メモリマップのシステムエリア</li> </ul> <p>メモリ領域は実行不可 (XN)(Execute Never) に設定されているため、命令フェッチは禁止です。MPU がある場合でも、これは変更できません。</p>
システム	システムセグメントはベンダのシステムペリフェラル用です。このメモリ領域は XN に設定されているため、命令フェッチは禁止です。MPU がある場合でも、これは変更できません。

プロセッサのメモリ領域のアクセス許可を、表 4-2 に示します。

表 4-2 メモリ領域のアクセス許可

名前	領域	デバイスのタイプ	XN	キャッシュ
コード	0x00000000-0x1FFFFFFF	ノーマル	-	WT
SRAM	0x20000000-0x3FFFFFFF	ノーマル	-	WBWA
SRAM_1M	+0000000	-	-	-
SRAM_31M	+0100000	-	-	
SRAM_bitband	+2000000	内部	-	-
SRAM	+4000000	-	-	-
ペリフェラル	0x40000000-0x5FFFFFFF	デバイス	XN	-
Periph_1IM	+0000000	-	-	-
Periph_31IM	+0100000	-	-	-
Periph_bit band	+2000000	内部	-	-
ペリフェラル	+4000000	-	-	-
外部 RAM	0x60000000-0x7FFFFFFF	ノーマル	-	WBWA
外部 RAM	0x80000000-0x9FFFFFFF	ノーマル	-	WT
外部デバイス	0xA0000000-0xBFFFFFFF	デバイス	XN	-
外部デバイス	0xC0000000-0xDFFFFFFF	デバイス	XN	-
専用ペリフェラルバス	+4000000	SO	XN	-
システム	+0100000	-	XN	-

—— 注 ——

0xE0000000 - 0xFFFFFFFF 範囲の専用ペリフェラルバスとシステム空間は、恒久的に実行不可 (XN) です。MPU を使って、これを変更することはできません。

プロセッサのバスインタフェースの詳細については、第 14 章 「バスインタフェース」を参照して下さい。

## 4.2 ビットバンド

プロセッサのメモリマップには、2つのビットバンド領域が含まれます。これらの領域は、**SRAM**およびペリフェラルのメモリ領域の最下位 **1MB** をそれぞれ占有します。ビットバンド領域は、メモリのエイリアス領域の各ワードをメモリのビットバンド領域のビットにマッピングします。

メモリマップは、2つの **1MB** ビットバンド領域にマップされる 2つの **32MB** エイリアス領域を持っています。

- **32MB** の **SRAM** エイリアス領域へのアクセスは、**1MB** の **SRAM** ビットバンド領域にマッピングされます。
- **32MB** のペリフェラルエイリアス領域へのアクセスは、**1MB** のペリフェラルビットバンド領域にマッピングされます。

マッピングの式により、エイリアス領域の各ワードが、対応するビットすなわちターゲットのビットに、ビットバンド領域でどのように参照されるのかを示します。マッピングする式は次のとおりです

$$\text{bit\_word\_offset} = (\text{byte\_offset} \times 32) + (\text{bit\_number} \times 4)$$

$$\text{bit\_word\_addr} = \text{bit\_band\_base} + \text{bit\_word\_offset}$$

ここで、

- **Bit\_word\_offset** は、ビットバンドメモリ領域での、ターゲットビットの位置
- **Bit\_word\_addr** は、エイリアスメモリ領域での、ターゲットビットがマップされるワードのアドレス
- **Bit\_band\_base** は、エイリアス領域の開始アドレス
- **Byte\_offset** は、ビットバンド領域での、ターゲットビットを含むバイトの位置 (バイト数)
- **Bit\_number** は、(ビットバンド領域の) ターゲットビット (を含むバイト中での) のビット位置 (0 ~ 7)

**SRAM** のビットバンド エイリアス領域と **SRAM** のビットバンド領域との間におけるビットバンドのマッピング例を、**P.4-6 図 4-2** に示します。

- **0x23FFFFE0** にあるエイリアスのワードは、**0x200FFFFF** にあるビットバンドのバイトのビット [0] にマップされます。 $0x23FFFFE0 = 0x22000000 + (0xFFFF * 32) + 0 * 4$
- **0x23FFFFFC** にあるエイリアスのワードは、**0x200FFFFF** にあるビットバンドのバイトのビット [7] にマップされます。 $0x23FFFFFC = 0x22000000 + (0xFFFF * 32) + 7 * 4$
- **0x22000000** にあるエイリアスのワードは、**0x20000000** にあるビットバンドのバイトのビット [0] にマップされます。 $0x22000000 = 0x22000000 + (0 * 32) + 0 * 4$
- **0x2200001C** にあるエイリアスのワードは、**0x20000000** にあるビットバンドのバイトのビット [7] にマップされます。 $0x2200001C = 0x22000000 + (0 * 32) + 7 * 4$

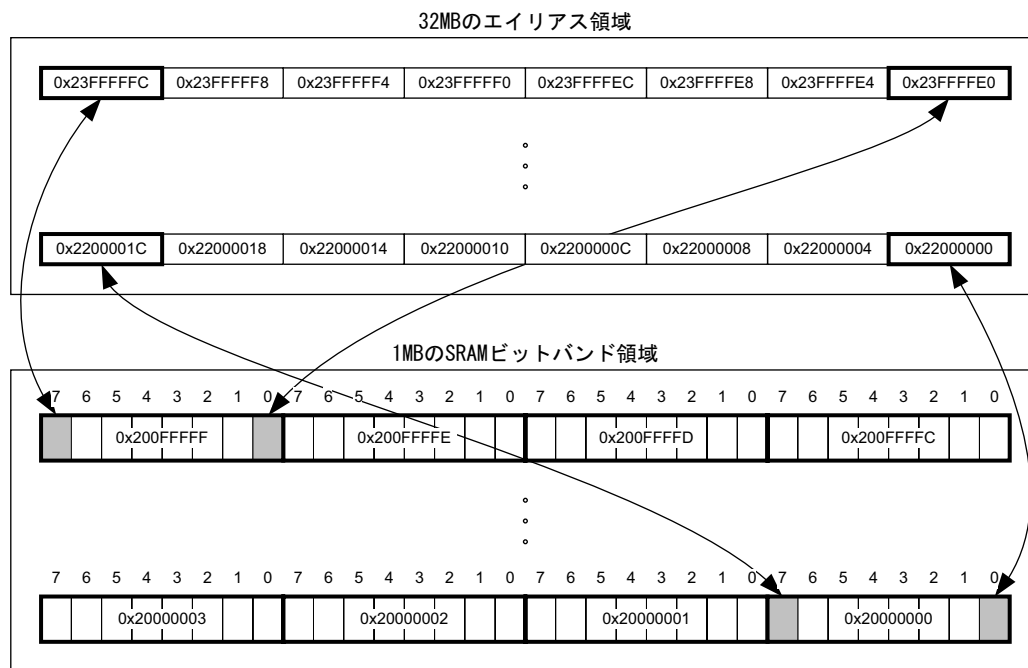


図 4-2 ビットバンドのマッピング

#### 4.2.1 エイリアス領域への直接アクセス

エイリアス領域へのワードの書き込みは、ビットバンド領域のターゲットビットに対する読み出し - 変更 - 書き込み操作と同じ効果があります。

エイリアス領域のワードに書き込まれる値のビット [0] により、ビットバンド領域のターゲットビットに書き込まれる値が決定されます。ビット [0] が 1 にセットされた値を書き込むと、ビットバンドのビットに 1 が書き込まれ、ビット [0] が 0 にクリアされた値を書き込むと、ビットバンドのビットに 0 が書き込まれます。

エイリアスのワード中のビット [31:1] は、ビットバンドのビットには影響しません。0x01 の書き込みは、0xFF の書き込みと同じ効果があります。0x00 の書き込みは、0x0E の書き込みと同じ効果があります。

エイリアス領域でのワードの読み出しは、0x01 または 0x00 を返します。0x01 という値は、ビットバンド領域のターゲットビットが 1 にセットされていることを示します。0x00 という値は、ターゲットビットが 0 にクリアされていることを示します。読み出しでは、ビット [31:1] は常に 0 です。

#### 4.2.2 ビットバンド領域への直接アクセス

ビットバンド領域に対して通常のリードやライトで直接アクセスしたり、その領域に書き込んだりすることもできます。

## 4.3 ROM メモリテーブル

ROM メモリの説明を、表 4-3 に示します。

表 4-3 ROM テーブル

オフセット	値	名前	説明
0x000	0xFFFF0F03	NVIC	0xE000E000. にある NVIC を指し示します。
0x004	0xFFFF0203	DWT	0xE0001000 にあるデータウォッチポイントおよびトレースブロックを指し示します。
0x008	0xFFFF0303	FPB	0xE0002000 にあるフラッシュパッチおよびブレークポイントブロックを指し示します。
0x00C	0xFFFF0103	ITM	0xE0000000 にある計装トレース ブロックを指し示します。
0x010	存在しなければ 0xFFFF41002、 存在すれば 0xFFFF41003	TPIU	TPIU を指し示します。TPIU が搭載されている場合、値のビット [0] は 1 にセットされています。TPIU は 0xE0040000 に位置します。
0x014	0xFFFF4203	ETM	ETM を指し示します。ETM が搭載されている場合、値のビット [0] は 1 にセットされています。ETM は 0xE0041000 に位置します。
0x018	0	End	ROM テーブルの終わりのマーカーです。CoreSight コンポーネントが追加されている場合、コンポーネントはこの位置から追加され、End のマーカーは追加されたコンポーネントの後の次の位置に移動されます。
0xFCC	0x1	MEM TYPE	ビット [31:1] は 0 として読み出されます (RAZ)。(デバッグが) DAP 経由でシステムメモリをアクセス可能な場合、ビット [0] が 1 にセットされます。DAP を使ってアクセス可能なのがデバッグコンポーネントのみでシステムメモリにアクセスできない場合、ビット [0] は 0 にクリアされます。
0xFD0	0x0	PID4	-
0xFD4	0x0	PID5	-
0xFD8	0x0	PID6	-
0xFDC	0x0	PID7	-
0xFE0	0x0	PID0	-
0xFE4	0x0	PID1	-
0xFE8	0x0	PID2	-
0xFEC	0x0	PID3	-
0xFF0	0x0D	CID0	-

表 4-3 ROM テーブル

オフセット	値	名前	説明
0xFF4	0x10	CID1	-
0xFF8	0x05	CID2	-
0xFFC	0xB1	CID3	-

## 第 5 章

### 例外

本章では、プロセッサの例外モデルについて説明します。本章は以下のセクションから構成されています。

- 例外モデルについて : P.5-2
- 例外のタイプ : P.5-4
- 例外の優先度 : P.5-6
- 特権とスタック : P.5-9
- 横取り : P.5-11
- テールチェイン : P.5-14
- 後着 : P.5-15
- 退出 : P.5-17
- リセット : P.5-20
- 例外制御の移行 : P.5-24
- 複数のスタックの設定 : P.5-25
- アボートモデル : P.5-27
- 起動レベル : P.5-32
- フローチャート : P.5-34

## 5.1 例外モデルについて

プロセッサとネスト型ベクタ割り込みコントローラ(NVIC)はすべての例外に優先度を付けて処理します。すべての例外はハンドラモードで処理されます。プロセッサの状態は、例外の発生時に自動的にスタックに保存され、割り込み処理ルーチン (ISR) の終了時に自動的にスタックから復元されます。割り込みへのエントリを効率的にするため、ベクタのフェッチは状態の保存と並行して行われます。プロセッサはテールチェインをサポートしているため、状態の保存と復元のオーバーヘッドなしに連続した割り込みが可能です。次に示す機能により、効率的でレイテンシの短い割り込み処理が可能になっています。

- 自動的な状態の保存と復元。プロセッサは ISR へ入る前にレジスタをスタックにプッシュし、ISR の退出時にポップします。これらの処理に命令オーバーヘッドは発生しません。
- 自動的なベクタテーブルのエントリ読み出し。ここには、コードメモリやデータ SRAM にある ISR のアドレスが存在します。この処理は状態の保存と並行して行われます。

### —— 注 ——

ベクタテーブルのエントリは ARM/Thumb インタワーキング互換です。

このため、ベクタの値のビット [0] は例外の開始時に EPSR の T ビットにロードされます。ビット [0] がクリアされたテーブルエントリを作成すると、そのベクタに対応するハンドラの最初の命令で INVSTATE フォールトが生成されます。

- テールチェインをサポート。テールチェインでは、プロセッサは ISR と ISR との間でレジスタのポップとプッシュを行わず、連続して割り込みを処理します。
- 割り込みの動的な優先度変更
- プロセッサコアと NVIC との間の密接なインタフェースにより、割り込みの早期処理と、高い優先度を持つ後着割り込みの処理が可能
- 割り込みの数は 1 ～ 240 で構成可能
- 割り込み優先度の番号は 3 ～ 8 ビットで構成可能 (8 ～ 256 レベル)
- ハンドラモードとスレッドモードのための、独立したスタックと特権レベル
- ARM アーキテクチャプロシージャ呼び出し標準 (AAPCS)(ARM Architecture Procedure Call Standard) で、C/C++ 標準の呼び出し規則を使用した ISR 制御の移行
- クリティカル領域をサポートする優先度マスク



---

**注**

---

割り込みの数と割り込み優先度のビットは実装時に構成されます。ソフトウェアで選択できるのは、構成されている割り込みの一部を許可することと、構成されている優先度のうち何ビットを使用するかのみです。

---

5.2 例外のタイプ

プロセッサには、各種の例外が存在します。フォールトは、命令の実行によってエラー条件が発生したことによる例外です。フォールトは、原因となった命令と同期報告されることもあれば、非同期に報告されることもあります。一般的にはフォールトは同期して報告されます。不正確なバスフォールトは、ARMv7-M プロファイルでサポートされている非同期フォールトです。同期フォールトは常に、フォールトの原因となった命令とともに報告されます。非同期フォールトでは、フォールトの原因となった命令に関して、どのように報告されるかの保証はありません。

例外の詳細については、『ARMv7-M アーキテクチャ リファレンスマニュアル』を参照して下さい。

例外のタイプ、位置、優先度を、表 5-1 に示します。位置は、ベクタテーブルの先頭から計算したワードオフセットを意味しています。優先度の列では、小さい数値が高い優先度を示しています。各タイプの例外がどのようにアクティブになるか、同期か非同期かも示されています。優先度の正確な意味と使用法については、P.5-6「*例外の優先度*」を参照して下さい。

表 5-1 例外のタイプ

例外のタイプ	位置	優先度	説明
-	0	-	リセット時に、ベクタテーブルの最初のエントリからスタックの先頭がロードされます。
リセット	1	-3 (最高)	電源投入時とウォームリセット時に呼び出されます。最初の命令で最低の優先度に落とされます (スレッドモード)。この例外は非同期です。
マスク不能割り込み	2	-2	リセット以外のどの例外によっても、中断も横取りもできません。この例外は非同期です。
ハードフォールト	3	-1	優先度の関係で、または構成可能なフォールトのハンドラが無効にされているためにフォールトを発生できないときの、すべてのクラスのフォールト。この例外は同期です。
メモリ管理	4	構成可能 <sup>a</sup>	メモリ保護ユニット (MPU) の不整合。これにはアクセス違反と不一致が含まれます。この例外は同期です。この例外は、デフォルトのメモリマップの <i>実行不可</i> (XN)(eXecute Never) 領域をサポートするため、MPU が無効にされているか存在しないときでも使用されます。
バスフォールト	5	構成可能 <sup>b</sup>	プリフェッチフォールト、メモリアクセス フォールト、その他のアドレスやメモリ関連のフォールト。この例外は正確な場合は同期、不正確な場合は非同期です。
用法フォールト	6	構成可能	用法フォールト、例えば未定義命令の実行や不正な状態への遷移の試み。この例外は同期です。
-	7-10	-	予約

表 5-1 例外のタイプ

例外のタイプ	位置	優先度	説明
SVCall（スーパーバイザ コール）	11	構成可能	SVC 命令によるシステムサービス呼び出し。この例外は同期です。
デバッグモニタ	12	構成可能	ホールド中でないときのデバッグモニタ。この例外は同期ですが、許可されているときのみアクティブになります。現在アクティブなものより優先度が低い場合はアクティブになりません。
-	13	-	予約
PendSV	14	構成可能	システムサービスへの保留可能な要求。この例外は非同期で、ソフトウェアによってのみ保留されます。
SysTick	15	構成可能	システムタイマの報知。この例外は非同期です。
外部割り込み	16 以上	構成可能	コアの外部 (INTISR[239:0]) からアサートされ、NVIC 経由で（優先度付けされて）供給されます。これらの例外はすべて非同期です。

- a. この例外の優先度は変更可能です。詳細については、P.8-28「システムハンドラ優先度レジスタのビット割り当て」を参照して下さい。設定可能なのは 0 ～ N までの NVIC 優先度の値で、N は実装されている最大の優先度です。内部的には、ユーザが設定可能な最高の優先度 (0) は 4 として扱われます。
- b. このフォールトは使用可能にも禁止にもできます。詳細については、P.8-29「システムハンドラ制御および状態レジスタのビット割り当て」を参照して下さい。

5.3 例外の優先度

プロセッサが例外を処理する時期と方法に優先度が与える影響を、表 5-2 に示します。  
この表には、例外が優先度に応じてどのような動作を行うかが一覧表示されています。

表 5-2 優先度に応じた例外の動作

動作	説明
横取り	<p>新しい例外が現在の例外優先度やスレッドよりも高い優先度を持っていて、現在のフローに割り込みます。</p> <p>これは保留中の割り込みへの応答で、保留中の割り込みがアクティブな <b>ISR</b> やスレッドよりも優先度が高い場合に、<b>ISR</b> が開始されることになります。ある <b>ISR</b> が別の <b>ISR</b> を横取りした場合、割り込みがネストされます。</p> <p>例外の開始時にプロセッサは自動的にプロセッサ状態をスタックにプッシュして、保存します。この動作と並行して、割り込みに対応するベクタがフェッチされます。<b>ISR</b> の最初の命令の実行は、プロセッサの状態が保存され、<b>ISR</b> の最初の命令がプロセッサのパイプラインの実行ステージに入った時点で開始されます。状態の保存は、システムバスおよび <b>DCode</b> バス経由で行われます。ベクタフェッチは、ベクタテーブルの場所に応じて、システムバスか <b>ICode</b> バスのいずれかを經由して行われます。これについては、P8-20「ベクタテーブル オフセットレジスタ」を参照して下さい。</p>
テールチェイン	<p>プロセッサが割り込み処理を高速化するために使用する機構。<b>ISR</b> の完了時に、<b>ISR</b> または復帰先のスレッドよりも優先度が高い保留中の割り込みが存在すれば、スタックのポップをスキップして、新しい <b>ISR</b> に制御が移されます。</p>
復帰	<p>保留中の例外がない場合や、スタックされている <b>ISR</b> よりも優先度の高い保留中の例外が存在しない場合、プロセッサはスタックをポップしてスタックされた <b>ISR</b> やスレッドモードに復帰します。</p> <p><b>ISR</b> の完了時に、プロセッサは自動的にスタックをポップして、<b>ISR</b> の開始の原因となった割り込みが発生する前の時点にプロセッサの状態を復元します。状態の復元中に新しい割り込みが到着し、その割り込みの優先度が <b>ISR</b> や復帰先のスレッドよりも高い場合、状態の復元は中止され、新しい割り込みがテールチェーンとして処理されます。</p>
後着	<p>プロセッサの横取りを高速化するために使用される機構。前の横取りによる状態保存の間に、より高い優先度の割り込みが到着した場合、プロセッサはより高い優先度の割り込みの処理に切り替わり、その割り込みのベクタフェッチを開始します。状態の保存はどちらの割り込みについても同じなので、後着割り込みの影響を受けずに状態の保存が続行されます。後着割り込みは、<b>ISR</b> の最初の命令がプロセッサパイプラインの実行ステージに入るまで有効です。復帰時には、テールチェーンの通常の規則が適用されます。</p>

プロセッサの例外モデルでは、プロセッサが例外を処理する時期と方法は、優先度によって決定されます。次の操作を実行できます。

- 割り込みにソフトウェアの優先度レベルを割り当てる。
- 優先度レベルを横取り優先度とサブ優先度とに分け、優先度をグループ化する。

### 5.3.1 優先度レベル

NVIC は、ソフトウェアで割り当てる優先度レベルをサポートしています。割り込み優先度レジスタ (Interrupt Priority Register) の 8 ビットの PRI\_N フィールドに値を書き込むことで、割り込みに 0 ~ 255 の優先度レベルを割り当てることができます。詳細については P.8-16 「割り込み優先度レジスタ」を参照して下さい。ハードウェア優先度は、割り込み番号が大きくなるほど優先度が低くなります。優先度レベル 0 は最も高い優先度レベルを、優先度レベル 255 は最も低い優先度レベルを意味します。この優先度レベルは、ハードウェア優先度より優先されます。例えば、IRQ[0] に優先度レベル 1 を、IRQ[31] に優先度レベル 0 を割り当てた場合、IRQ[31] の優先度が IRQ[0] よりも高くなります。

---

#### 注

---

ソフトウェアによる優先度付けは、リセット、マスク不能割り込み (NMI)、ハードフォールトには影響しません。これらは常に、外部割り込みよりも高い優先度で処理されます。

複数の割り込みについて優先度番号が同じ場合、最も低い割り込み番号を持つ保留中の割り込みが優先されます。例えば、IRQ[0] と IRQ[1] が両方とも優先度レベル 1 の場合、IRQ[0] が IRQ[1] よりも優先されます。

PRI\_N フィールドの詳細については、P.8-16 「割り込み優先度レジスタ」を参照して下さい。

### 5.3.2 優先度のグループ化

多数の割り込みの優先度をシステムで適切に制御するため、NVIC では優先度のグループ化がサポートされています。P.8-21 「アプリケーション割り込みおよびリセット制御レジスタ」に示す PRIGROUP フィールドを使用して、すべての PRI\_N フィールドの値を横取り優先度フィールドとサブ優先度フィールドに分割することができます。横取り優先度グループは、グループ優先度と呼ばれます。複数の保留中の例外が同じグループ優先度を共有している場合、グループ内の優先度はサブ優先度ビットフィールドによって解決されます。これを、グループ内のサブ優先度と呼びます。グループ優先度とサブ優先との組み合わせが、一般に優先度と呼ばれます。2 つの保留中の例外があり、優先度が同じの場合、例外番号の低いものが高いものより優先されます。これは、優先度の順位決定体系と整合しています。

PRIGROUP への書き込みにより、8 ビットの PRI\_N フィールドがどのように横取り優先度フィールド (x) とサブ優先度フィールド (y) に分割されるかを表 5-3 に示します。

表 5-3 優先度のグループ化

割り込み優先度レベルフィールド、PRI_N[7:0]					
PRIGROUP[2:0]	2 進小数点の位置	横取りフィールド	サブ優先度フィールド	横取り優先度の数	サブ優先度の数
b000	bxxxxxxx.y	[7:1]	[0]	128	2
b001	bxxxxxx.yy	[7:2]	[1:0]	64	4
b010	bxxxxx.yyy	[7:3]	[2:0]	32	8
b011	bxxxx.yyyy	[7:4]	[3:0]	16	16
b100	bxxx.yyyyy	[7:5]	[4:0]	8	32
b101	bxx.yyyyyy	[7:6]	[5:0]	4	64
b110	bx.yyyyyyy	[7]	[6:0]	2	128
b111	b.yyyyyyyy	なし	[7:0]	0	256

——— 注 ———

- 8 ビットの優先度で構成されたプロセッサの優先度を、表 5-3 に示します。
- 8 ビットより少ない優先度で構成されたプロセッサの場合、レジスタの下位ビットは常に 0 です。例えば、4 ビットで優先度が実装されている場合、PRI\_N[7:4] で優先度が設定され、PRI\_N[3:0] は 4'b0000 です。

割り込みが進行中の他の割り込みを横取りできるのは、横取り優先度が進行中の割り込みのものよりも高い場合のみです。

優先度の最適化、優先度レベルのグループ化、優先度マスクの詳細については、『ARMv7-M アーキテクチャ リファレンスマニュアル』を参照して下さい。

## 5.4 特権とスタック

プロセッサは、次の 2 つの独立したスタックをサポートしています。

### プロセススタック

スレッドモードでプロセススタックを使用するように構成できます。リセット以降、スレッドモードはメインスタックを使用しています。SP\_process は、プロセススタック用の *Stack Pointer*(SP) レジスタです。

### メインスタック

ハンドラモードはメインスタックを使用します。SP\_main は、メインスタック用の SP レジスタです。

どの時点でも、可視になっているのはプロセススタックまたはメインスタックのいずれか 1 つです。8 つのレジスタをプッシュした後で、ISR はメインスタックを使用し、以後の割り込みの横取りもすべてメインスタックを使用します。コンテキストを保存するためのスタックは次のとおりです。

- スレッドモードは、CONTROL のビット [1] の値に従って、メインスタックまたはプロセススタックのいずれかを使用します。CONTROL[1] には、MSR(Move to Status Register) または MRS(Move to Register from Status) でアクセス可能です。ISR を退出するとき、適切な EXC\_RETURN の値によって、このビットをセットすることもできます。ユーザスレッドを横取りする例外は、スレッドモードが使用しているスタックに、ユーザスレッドのコンテキストを保存します。
- すべての例外は、自分のローカル変数用にメインスタックを使用します。

スレッドモードにはプロセススタックを、例外にはメインスタックを使用することで、オペレーティングシステム(OS) のスケジューリングをサポートします。再スケジューリングを行うとき、カーネルはハードウェアによってプッシュされない 8 つのレジスタ(r4 ~ r11) を保存し、SP\_process をスレッド制御ブロック(TCB)(Thread Control Block) にコピーするだけで済みます。仮にプロセッサがメインスタックにコンテキストを保存すると想定した場合、カーネルは 16 のレジスタを TCB にコピーする必要があります。

#### 注

MSR および MRS 命令は、両方のスタックが可視でアクセスできます。

### 5.4.1 スタック

スタックのモデルは特権のモードと独立しています。つまり、スレッドモードはプロセススタックとメインスタックのどちらでも使用でき、またユーザモードと特権モードのどちらも可能です。スタックと特権の 4 つの組み合わせが、いずれも可能です。基本的な保護されたスレッドモデルでは、ユーザスレッドはプロセススタックを使用してスレッドモードで実行され、カーネルおよび割り込みはメインスタックを使用して特権モードで実行されます。

---

**注**

---

特権のみでは、悪意によるまたは事故によるスタックの破壊を防止できません。ユーザコードを分離するため、何らかの形式のメモリ保護方式が1つか2つは必要です。つまり、ユーザコードが他のスタックなど、自分の所有していないメモリに書き込むことを防止する必要があります。

---

## 5.4.2 特権

特権はアクセス権を制御するもので、ARMv7-Mの他のすべての概念と分離されています。コードは、特権となって完全なアクセス権を持つことも、非特権で制限されたアクセス権を持つこともできます。次のような操作を行えるかどうかは、アクセス権により決定されます。

- MSR フィールドなど、特定の命令の使用または不使用
- システム制御空間(SCS)(System Control Space) レジスタへのアクセス
- システムの設計に基づいて、メモリやペリフェラルへのアクセス。プロセッサはシステムに対して、アクセスを行っているコードが特権コードかどうかを通知し、システムはそれに基づいて非特権のアクセスに対して制限を加えることができます。
- MPU に基づいて、メモリ位置へのアクセス規則。MPU が搭載されている場合、アクセス制限によって読み出し、書き込み、実行が可能なメモリを制御できます。

非特権モードにすることができるのはスレッドモードのみです。例外はすべて特権モードです。



5.5 横取り

以下のセクションでは、プロセッサが例外を取得したときの動作について説明します。

- スタック操作
- 後着: P.5-15
- デールチェイン: P.5-14

5.5.1 スタック操作

プロセッサは、例外を呼び出すとき、次の 8 つのレジスタを自動的に SP へ、次に示す順序でプッシュします。

- プログラムカウンタ (PC)
- プログラムステータス レジスタ (xPSR)
- 0 ~ r3
- r12
- リンク レジスタ (LR)

スタックのプッシュが完了すると、SP は 8 ワード減らされます。例外によって現在のプログラムフローが横取りされた後のスタックの内容を、図 5-1 に示します。

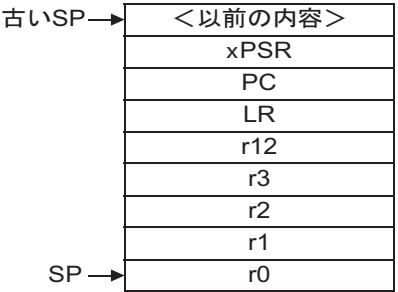


図 5-1 横取り後のスタックの内容

注

- 図 5-1 に示されているのは、スタックでの順序です。
- 構成制御レジスタ (Configuration Control Register) の STKALIGN がセットされている場合、スタック処理が行われる前に追加ワードを挿入できます。詳細については、P.8-25 「構成制御レジスタ」を参照して下さい。

ISR からの復帰後に、プロセッサは 8 つのレジスタをスタックから自動的にポップします。割り込みからの復帰は LR 中のデータフィールドとして渡されるため、ISR 機能を通常の C/C++ 関数として作成でき、ベニアを必要としません。

プロセッサが **ISR** を開始する前に実行する手順を、表 5-4 に示します。

表 5-4 例外の開始手順

動作	やり直し可能か	説明
8 つのレジスタを プッシュ <sup>a</sup>	不可	xPSR、PC、r0、r1、r2、r3、r12、LR を選択されているスタックにプッシュします。
ベクタテーブルの 読み出し	可。後着例外により やり直しになることが あります。	テーブルのベース + (例外番号 × 4) のアドレスからベクタ テーブルを読み出します。ICode バスで読み出すと同時に、 DCode バスでレジスタのプッシュを実行できます。
ベクタテーブル から SP を読み出 し	不可	リセット時のみ、SP の値を、ベクタテーブルから読み出した スタックの先頭に更新します。他の例外は、スタック選択、 プッシュ、ポップを除いて、SP を変更しません。
PC の更新	不可	ベクタテーブルから読み出した値に PC を更新します。後着例 外は、最初の命令の実行が開始されるまで処理できません。
パイプラインの ロード	可。横取りが発生す ると、新たに読み出 されたベクタテーブ ルでパイプラインが 再びロードされます。	ベクタテーブルが指し示す場所から命令をロードします。こ の処理は、レジスタのプッシュと並行して行われます。
LR の更新	不可	例外から退出のため、LR が EXC_RETURN に設定されます。 EXC_RETURN は、『ARMv7-M アーキテクチャリファレンスマ ニュアル』で定義されている 16 の値のいずれかです。

a. テールチェインの場合、この手順はスキップされます。

例外開始のタイミング例を、P.5-13 図 5-2 に示します。

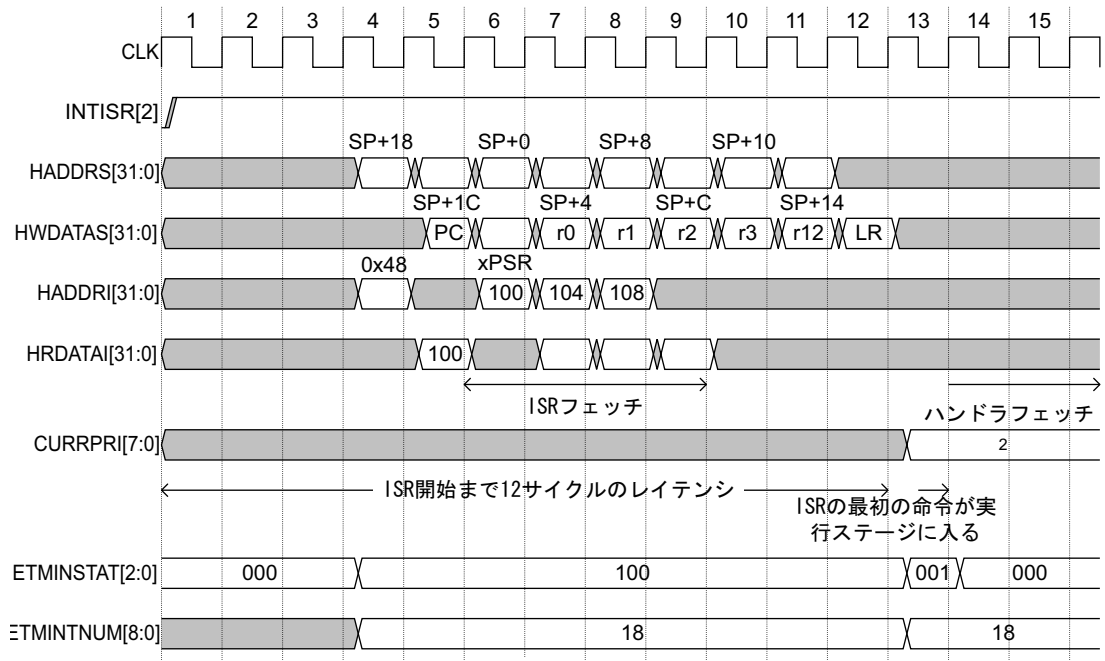


図 5-2 例外開始のタイミング

NVIC は、**INTISR[2]** が受信された後のサイクルで、割り込みが受信されたことをプロセッサコアに通知します。プロセッサコアはその後のサイクルで、スタックへのプッシュとベクタのフェッチとを開始します。

スタックへのプッシュが完了すると、ISR の最初の命令がパイプラインの実行ステージに入ります。ISR の実行が開始されるサイクルに、次の信号が出力されます。

- **ETMINSTAT[2:0]** は、ISR が開始されたことを示します (3'b001)。これは 1 サイクルのパルスです。
- **CURRPRI[7:0]** は、アクティブな割り込みの優先度を示します。**CURRPRI** は ISR の継続期間中ずっとアサート状態に維持されます。**CURRPRI** は、**ETMINTSTAT** によって ISR の開始が示されたとき (3'b001) に有効になります。
- **ETMINTNUM[8:0]** は、アクティブな割り込みの番号を示します。**ETMINTNUM** は ISR の継続期間中ずっとアサート状態に維持されます。**ETMINTNUM** は、**ETMINTSTAT** によって ISR の開始が示されたとき (3'b001) に有効になります。それ以前は、この信号はどの ISR がフェッチされているかを示します。

図 5-2 では、割り込みのアサートから ISR の最初の命令が実行されるまで、12 サイクルのレイテンシがあることが示されています。

## 5.6 テールチェーン

テールチェーンは、割り込みと割り込みとの間で状態保存および復元のオーバーヘッドを伴わない連続割り込み処理です。プロセッサは、1つのISRを退出して別のISRの実行を開始するとき、8つのレジスタをポップしてまた8つのレジスタをプッシュするという操作を省きます。このような処理をしてもスタックの内容には何も影響を与えないからです。

プロセッサは、保留中の割り込みが、スタックされているどの例外よりも優先度が高いとき、テールチェーンを行います。

テールチェーンの例を、図 5-3 に示します。保留中の割り込みが、スタックされている最も優先度が高い例外よりもさらに優先度が高い場合、スタックのプッシュとポップは省略され、プロセッサは保留中の割り込みのベクタを直ちにフェッチします。テールチェーンされるISRは、前のISRの終了から6サイクル後に実行開始されます。

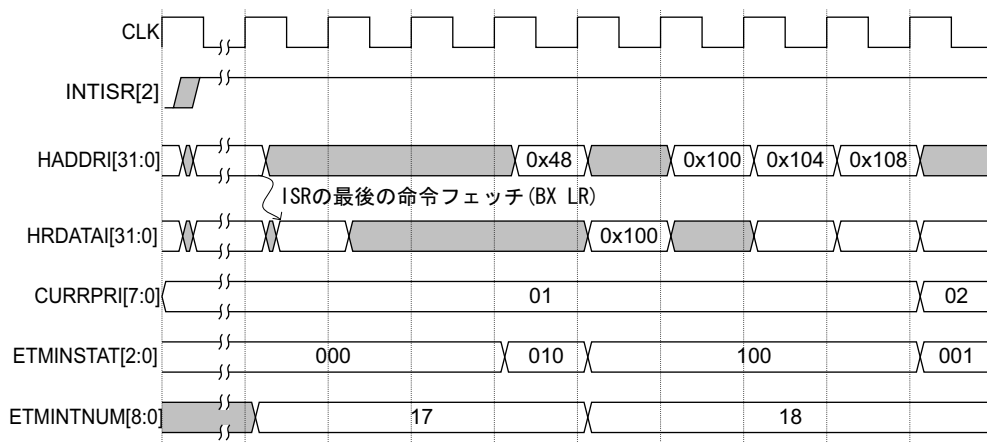


図 5-3 テールチェーンのタイミング

最後のISRからの復帰時に、INTISR[2]の優先度が、スタックされているどのISRや他の保留中の割り込みよりも高いため、プロセッサはINTISR[2]に対応するISRへテールチェーンを行います。INTISR[2]に対応するISRの実行が開始されるサイクルに、次の信号が出力されます。

- **ETMINSTAT[2:0]** はISRが開始されたことを示します (3'b001)。これは1サイクルのパルスです。
- **CURRPRI[7:0]** は、アクティブな割り込みの優先度を示します。CURRPRIはISRの継続期間中ずっとアサート状態に維持されます。
- **ETMINTNUM[8:0]** は、アクティブな割り込みの番号を示します。ETMINTNUMはISRの継続期間中ずっとアサート状態に維持されます。

図 5-3 は、前のISRから復帰した後、新しいISRの実行までのレイテンシが6サイクルであることを示しています。

## 5.7 後着

後着割り込みは、先行する ISR の最初の命令がまだ実行ステージに入っておらず、後着割り込みの優先度が先行する割り込みよりも高い場合に、前の割り込みを横取りでできます。

後着割り込みによって、新しいベクタアドレスのフェッチと ISR のプリフェッチが行われます。状態保存は、後着割り込みについては行われません。これは、最初の割り込みによってすでに状態保存が行われているため、繰り返し実行する必要がないからです。

後着割り込みの例を、図 5-4 に示します。

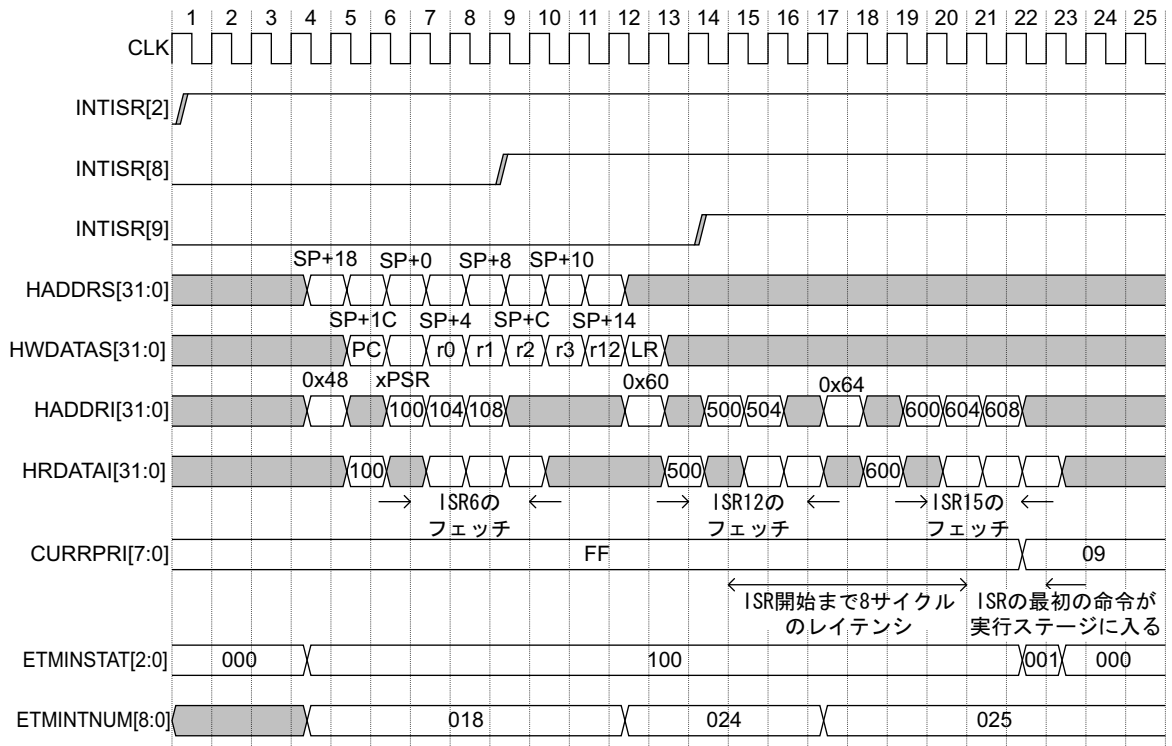


図 5-4 後着例外のタイミング

図 5-4 で、**INTISR[8]** は **INTISR[2]** を横取りします。状態保存は **INTISR[2]** に対してすでに行われているため、繰り返す必要はありません。図 5-4 は、**INTISR[2]** の ISR の最初の命令が実行ステージに入る前の、**INTISR[8]** が横取りできる最後の時点を示しています。この時点より後に、より高い優先度の割り込みが発生すると、横取りとして管理されます。

P.5-15 図 5-4 は、**INTISR[8]** に対する ISR の最初の命令がフェッチステージに入る前の、**INTISR[9]** が横取りできる最後の時点を示しています。**INTISR[9]** が受信されると **INTISR[8]** に対する ISR のフェッチはアボートされ、プロセッサは **INTISR[9]** に対するベクタのフェッチを開始します。この時点より後に、より高い優先度の割り込みが発生すると、横取りとして管理されます。

**INTISR[9]** に対応する ISR の実行が開始されるサイクルに、次の信号が出力されます。

- **ETMINSTAT[2:0]** は ISR が開始されたことを示します (3'b001)。これは 1 サイクルのパルスです。
- **CURRPRI[7:0]** は、アクティブな割り込みの優先度を示します。**CURRPRI** は ISR の継続期間中ずっとアサート状態に維持されます。
- **ETMINTNUM[8:0]** は、アクティブな割り込みの番号を示します。**ETMINTNUM** は ISR の継続期間中ずっとアサート状態に維持されます。

## 5.8 退出

ISR の最後の命令は、PC に 0xFFFFFEX をロードします。これは、例外開始時の LR の値です。この動作は、ISR が完了したことをプロセッサに通知し、プロセッサは例外退出シーケンスを開始します。プロセッサが ISR から復帰するために使用できる命令については、P.5-18「プロセッサのISRからの復帰」を参照して下さい。

### 5.8.1 例外退出

例外から復帰するとき、プロセッサは次のいずれかの動作を行います。

- 保留中の例外が、スタックされているどの例外よりも優先度が高い場合、テールチェーンが行われます。
- 保留中の例外がない場合、またはスタックされている例外のうち最も高い優先度のものが、保留中の例外のうち最も高い優先度のものよりも優先度が高い場合、最後にスタックされた ISR への復帰が行われます。
- 保留中またはスタックされている例外が存在しない場合、スレッドモードへの復帰が行われます。

コンテキストの復元のシーケンスを、表 5-5 に示します。

**表 5-5 例外の退出手順**

動作	説明
8 つのレジスタをポップ	横取りが行われない場合、EXC_RETURN によって選択されているスタックから PC、xPSR、r0、r1、r2、r3、r12、LR をポップし、SP を調整します。
現在アクティブな割り込み番号 <sup>a</sup> をロードし、逆方向のスタックアライメント調整	現在アクティブな割り込み番号を、スタックされている IPSR のワードのビット [8:0] からロードします。プロセッサはこの番号を使用して、どの例外へ復帰するかを追跡し、復帰時にアクティブを示すビットをクリアします。ビット [8:0] が 0 の場合、プロセッサはスレッドモードへ復帰します。
SP を選択	例外へ復帰する場合、SP は SP_main です。スレッドモードへ復帰する場合、SP は SP_main または SP_process です。

- a. 動的な優先度の変更があるため、NVIC は割り込み優先度ではなく割り込み番号を使用して、どの ISR が現在のものかを判定します。

例外退出のタイミング例を、P.5-18 図 5-5 に示します。

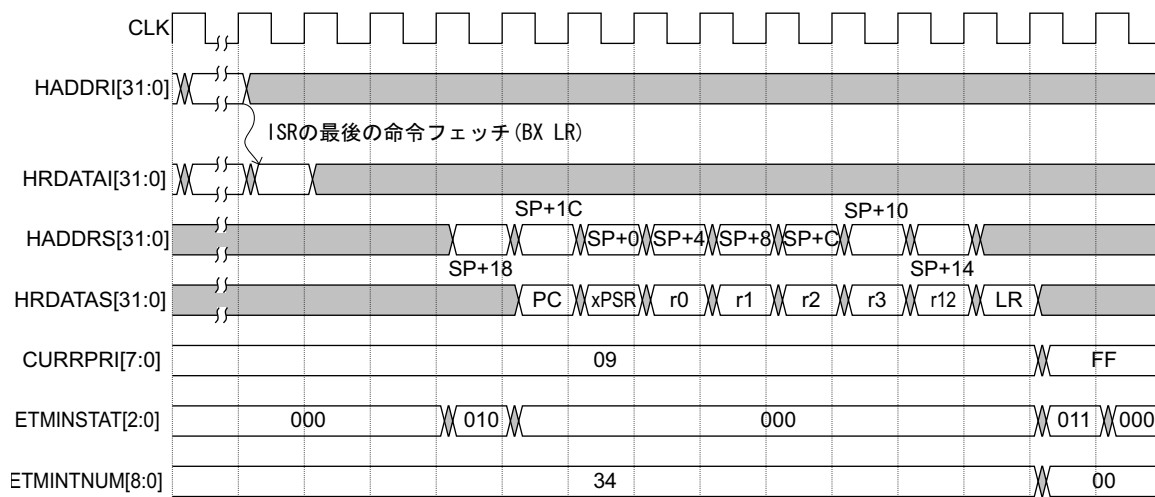


図 5-5 例外退出のタイミング

**ETMINSTAT** は次の内容を意味しています。

- 3'b010 の場合、ISR が退出したことを示します。**ETMINTNUM** は、退出した ISR の番号を示します。
- 前にスタックされた ISR へ復帰する場合、割り込み退出後のサイクルで 3'b011 になります。**ETMINTNUM** は、復帰先の割り込みの番号を示します。

#### 注

スタックのポップ中に、より優先度の高い例外が発生した場合、プロセッサはスタックのポップを中止し、スタックポインタを元に戻してから、テールチェーンとして例外の処理を行います。

## 5.8.2 プロセッサの ISR からの復帰

例外からの復帰は、次のいずれかの命令によって PC に値 0xFFFFFFFF がロードされたときに発生します。

- PC のロードを含む POP/LDM
- PC を転送先とする LDR
- 任意のレジスタでの BX (分岐と状態遷移)



この方法で使用される場合、PC に書き込まれる値は途中で介在を受け、**EXC\_RETURN** 値と呼ばれます。EXC\_RETURN[3:0] は、表 5-6 に定義されているように復帰情報を示します。

表 5-6 例外からの復帰動作

EXC_RETURN[3:0]	説明
0bXXX0	予約
0b0001	ハンドラモードへの復帰。 例外からの復帰時には、メインスタックから状態が復元されます。 復帰時の実行はメインスタックを使用します。
0b0011	予約
0b01X1	予約
0b1001	スレッドモードへの復帰。 例外からの復帰時には、メインスタックから状態が復元されます。 復帰時の実行はメインスタックを使用します。
0b1101	スレッドモードへの復帰。 例外からの復帰時には、プロセススタックから状態が復元されます。 復帰時の実行はプロセススタックを使用します。
0b1X11	予約

この表に含まれている予約の項目を使うと、用法フォールトへの連鎖的な例外を引き起こします。

EXC\_RETURN の値を PC にロードするのが（例外からの復帰でなく）スレッドモードの場合、ベクタテーブルからの場合、または他の命令によるものである場合、その値は特別な値ではなくアドレスと見なされます。このアドレスの範囲には**実行不可 (XN)** のアクセス許可が定義されているため、**MemManage** フォールトが発生します。

5.9      リセット

NVIC はコアと同時にリセットされ、コアへのリセットの解除を制御します。その結果、リセットの動作は予測可能です。リセット時の動作を表 5-7 に示します。

表 5-7 リセット時の動作

動作	説明
NVIC がリセットされ、コアをリセット状態に保持する	NVIC は自分のレジスタのほとんどをクリアします。プロセッサはスレッドモード、優先度は特権モード、スタックはメインスタックに設定されます。
NVIC がコアをリセットから解除する	NVIC はコアのリセットを解除します。
コアがスタックを設定する	コアは、ベクタテーブルのオフセット 0 から SP の初期値を読み出します。SP は SP_main です。
コアが PC と LR を設定する	コアは、ベクタテーブルのオフセットから PC の初期値を読み出します。LR は 0xFFFFFFFF に設定されます。
リセットルーチンを実行する	NVIC は割り込みを禁止します。NMI とハードフォールトは禁止されません。

リセットの詳細については、第 6 章 「クロックとリセット」を参照して下さい。

5.9.1      5.16.1 ベクタテーブルとリセット

ベクタテーブルの位置 0 には、4 つの値のみが必要です。

- スタックの先頭アドレス
- リセットルーチンの位置
- NMI の ISR の位置
- ハードフォールトの ISR の位置

割り込みが許可されている場合、位置にかかわらず、ベクタテーブルはすべてのマスク可能な例外を指し示しています。また、SVC 命令が使用されている場合、SVCall の ISR の位置も書き込まれています。

完全なベクタテーブルの例を次に示します。

```
unsigned int stack_base[STACK_SIZE];
void ResetISR(void);
void NmiISR(void);
...
ISR_VECTOR_TABLE vector_table_at_0
{
    stack_base + sizeof(stack_base),
    ResetISR,
    NmiISR,
    FaultISR,
    0,           // Populate if using MemManage (MPU)
```

```

0,          // Populate if using Bus fault
0,          // Populate if using Usage Fault
0, 0, 0, 0, // reserved slots
SVCallISR,
0,          // Populate if using a debug monitor
0,          // Reserved
0,          // Populate if using pendable service request
0,          // Populate if using SysTick
// external interrupts start here
Timer1ISR,
GpioInISR
GpioOutISR,
I2CIsr
};

```

### 5.9.2 想定している起動シーケンス

通常のリセットルーチンは、表 5-8 に示されている手順に従います。C/C++ ランタイムは、最初の 3 つの手順を実行してから main() を呼び出すことができます。

**表 5-8 リセット時のブートアップ動作**

動作	説明
変数の初期化	すべてのグローバルまたは静的な変数を設定する必要があります。これには、BSS セクションの変数を 0 に初期化する動作と、定数でない変数の初期値を ROM から RAM へコピーする動作が含まれます。
[ スタックの設定 ]	複数のスタックを使用する場合、他のバンクになっている SP も初期化する必要があります。現在の SP をプロセスからメインに変更することもできます。
ランタイムの初期化	必要なら、C/C++ ランタイムの初期化コードを呼び出して、ヒープ、浮動小数点、その他の機能を有効にすることもできます。この動作は通常、C/C++ ライブラリの _main によって行われます。
[ ペリフェラルの初期化 ]	割り込みを許可する前に、ペリフェラルを設定します。この動作は、アプリケーションで使用するペリフェラルのそれぞれについて設定のため呼び出すことができます。
[ISR ベクタテーブルの切り替え]	ベクタテーブルをコード領域のアドレス 0 から、SRAM 内の位置に変更してもかまいません。この動作が必要なのは、性能を最適化したり、動的な変更を許可したりするときだけです。
[ 構成可能なフォールトの設定 ]	構成可能なフォールトを許可し、優先度を設定します。
割り込みの設定	優先度レベルとマスクを設定します。

表 5-8 リセット時のブートアップ動作

動作	説明
割り込みの許可	割り込みを許可します。NVIC での割り込み処理を許可します。割り込み許可の処理を行っている間に割り込みが発生するのは望ましくありません。32 よりも多い割り込みが存在する場合、セット - イネーブル レジスタが 2 つ以上必要になります。CPS または MSR 経由で PRIMASK を使用して、準備が整うまで割り込みをマスクすることができます。
[ 特権の変更 ]	[ 特権の変更 ]。必要なら、スレッドモードの特権をユーザーに変更できます。この動作は、通常は SVCall ハンドラ呼び出しによって処理する必要があります。
ループ	退出時スリープが許可されている場合、最初の割り込みまたは例外を処理した後で制御は戻ってきません。退出時スリープが選択的に許可 / 禁止されている場合、このループでクリーンアップと実行のタスクを管理できます。退出時スリープを使用しない場合、ループは自由で、必要に応じて WFI（即時スリープ、sleep-now）を使用できます。

——— 注 ———

P.5-21 表 5-8 の項目のうち、角括弧で囲まれているものはオプションの動作です。

リセットルーチンの例

リセットルーチンは、アプリケーションを起動して、割り込みを有効にする役割を担います。割り込み処理が実行された後で、リセット ISR を使用する方法は 3 つあります。これはリセット ISR (Reset ISR) のメインループ部分と呼ばれます。例 5-1、P. 5-23 例 5-2、P. 5-23 例 5-3 に 3 つの例を示します。

例 5-1 純粋な退出時スリープのリセットルーチン（リセットルーチンはメインループとしての処理をしない）

```
void reset()
{
    // do setup work (initialize variables, initialize runtime if wanted,
    setup peripherals, etc)
    nvic[INT_ENA] = 1;    // enable interrupts
    nvic_regs[NV_SLEEP] |= NV_SLEEP_ON_EXIT; // will not normally come back after
                                           1st exception

    while (1)
        wfi();
}
```

---

**例 5-2 WFI を使用した選択スリープモデルを持つリセットルーチン**


---

```
void reset()
{
    extern volatile unsigned exc_req;
    // do setup work (initialize variables, initialize runtime if wanted,
    // setup peripherals, etc)
    nvic[INT_ENA] = 1;    // enable interrupts
    while (1)
    {
        // do some work for (exc_req = FALSE; exc_req == FALSE; )
        wfi(); // sleep now - wait for interrupt
        // do some post exception checking/cleanup
    }
}
```

---

**例 5-3 選択された退出時スリープが、通知を必要とする ISR によって  
キャンセルされるリセットルーチン**


---

```
void reset()
{
    // do setup work (initialize variables, initialize runtime if wanted,
    // setup peripherals, etc)
    nvic[INT_ENA] = 1;    // enable interrupts
    while (1)
    {
        // We are slept until an exception clears sleep on exit state so that we
        // can post-process/cleanup.
        nvic_regs[NV_SLEEP] |= NVSLEEP_ON_EXIT;
        while (nvic_regs[NV_SLEEP] & NVSLEEP_ON_EXIT)
            wfi(); // sleep now - wait for interrupt which clears
        // do some post exception checking/cleanup
    }
}
```

---



---

**注**


---

ISR の起動が優先度レベルの変更として機能するので、リセットルーチンにカーネルは必要ありません。これによって、負荷の変化に迅速に対応し、優先度の上昇を使用して優先度の反転を解決し、粒度の細かいサポートをすることが十分可能になります。スレッドと特権を使用するリアルタイムオペレーティングシステム (RTOS) モデルでは、ユーザコードにスレッドモードが使用されます。

---

5.10 例外制御の移行

プロセッサは、表 5-9 に示す規則に従って制御を ISR へ移行します。

表 5-9 例外処理への移行

例外アサート時のプロセッサの動作	例外処理への移行
メモリを使用しない命令	次の命令の前の、サイクルの完了時に例外を処理します。
単一ロード / ストア	バスのステータスにより、転送を完了または破棄します。バスのウェイトステート次第で、次のサイクルで例外を処理します。
複数ロード / ストア	現在転送中のレジスタを完了または破棄し、継続カウンタ (ICI フィールド) を EPSR に設定します。バスのアクセス許可と (ICI)(Interruptible-Continuable Instruction) の規則に応じて、次のサイクルで例外を処理します。ICI 規則の詳細については、『ARMv7-M アーキテクチャ リファレンスマニュアル』を参照して下さい。
例外の開始	これは後着例外のひとつです。現在開始中の例外よりも優先度が高い場合、プロセッサは例外の開始動作をキャンセルし、後着例外を処理します。後着例外は、割り込み処理時での判断変更 (ベクタテーブル) を引き起こします。新しいハンドラが開始されるとき、つまり ISR の最初の命令で、通常の横取り規則が適用され、後着には分類されなくなります。
テールチェイン	これは後着例外のひとつです。テールチェインされているものよりも優先度が高い場合、プロセッサはコンテキスト保存をキャンセルし、後着例外を処理します。
例外のコンテキスト復元	新しい例外が、プロセッサの復帰先であるスタックされた例外よりも優先度が高い場合、プロセッサは新しい例外へのテールチェインを行います。

## 5.11 複数のスタックの設定

複数のスタックを実装するには、アプリケーションが次の動作を行う必要があります。

- MSR 命令を使用して、Process\_SP レジスタを設定します。
- MPU を使用している場合、スタックを適切に保護します。
- スレッドモードのスタックと特権を初期化します。

スレッドモードの特権が特権モードからユーザモードに変更された場合、SVCall など別の ISR のみが特権モードに戻すことができます。

スレッドモードのスタックはメインからプロセスへ、または逆に変更できますが、この操作を行うとスレッドのローカル変数へのアクセスに影響します。スレッドモードで使用するスタックは、別の ISR を使用して変更することをお勧めします。ブートシーケンスの例を次に示します。

1. 設定ルーチンを呼び出し、次の動作を行います。
  - a. MSR を使用して他のスタックを設定します。
  - b. MPU が存在する場合、MPU によるベース領域のサポートを許可します。
  - c. すべてのブートルーチンを起動します。
  - d. セットアップルーチンへ復帰します。
2. スレッドモードを非特権モードに変更します。
3. SVC を使用してカーネルを起動します。次に、カーネルは以下の処理を行います。
  - a. スレッドを開始します。
  - b. MRS を使用して現在のユーザスレッドの SP を読み出し、TCB に保存します。
  - c. MSR を使用して、次のスレッド用の SP を設定します。これは通常、SP\_process です。
  - d. 必要なら、新しい現在のスレッド用に MPU を設定します。
  - e. 新しい現在のスレッドに復帰します。

PSP を使用して復帰先の ISR の EXC\_RETURN 値を変更する方法を、例 5-4 に示します。

### 例 5-4 ISR の EXC\_RETURN 値の変更

---

```

; First time use of PSP, run from a Handler with RETTBASE == 1

LDR r0, PSPValue      ; acquire value for new Process stack
MSR PSP, r0            ; set Process stack value
ORR lr, lr, #4         ; change EXC_RETURN for return on PSP
BX lr                 ; return from Handler to Thread

```

---

PSP のスレッドに切り替えた後で、単純なコンテキストスイッチャを実装する方法を P. 5-26 例 5-5 に示します。

**例 5-5 単純なコンテキストスイッチャの実装**

---

```
; Example Context Switch (Assumes Thread is already on PSP)

MRS r12, PSP                ; Recover PSP into R12
STMDB r12!, {r4-r11, LR}    ; Push non-stack registers
LDR r0, =OldPSPValue        ; Get pointer to old Thread Control Block
STR r12, [r0]               ; Store SP into Thread Control Block
LDR r0, =NewPSPValue        ; Get pointer to new Thread Control Block
LDR r12, [r0]               ; Acquire new Process SP
LDMIA r12!, {r4-r11, LR}    ; Restore non-stacked registers
MSR PSP, r12                ; Set PSP to R12
BX lr                      ; Return back to Thread
```

---

---

**注**

---

P. 5-25 例 5-4 および例 5-5 において、スレッドを MSP から PSP へ移動する決定を行える時期、またはスタックされていないレジスタがスタックされたハンドラによって変更されていないことを保証できる時期は、アクティブな ISR/ ハンドラが 1 つしか存在しないときです。

---



## 5.12 アボートモデル

フォールトを生成する可能性があるイベントは4つです。

- 命令フェッチまたはベクタテーブルロードのバスエラー
- データアクセスのバスエラー
- 未定義命令や、BX 命令による状態変更の試みなど、内部で検出されたエラー。NVIC のフォールトステータスレジスタにフォールトの原因が示されます。
- 特権違反や管理されていない領域の結果として発生する MPU フォールト

フォールトハンドラには次の2種類があります。

- 固定優先度のハードフォールト
- 優先度を設定できるローカルフォールト

### 5.12.1 ハードフォールト

固定優先度のハードフォールトは、リセットおよびNMIのみが横取りできます。ハードフォールトは、リセット、NMI、他のハードフォールトを除くすべての例外を横取りできます。

---

#### 注

---

FAULTMASK を使用するコードはハードフォールトとして動作し、ハードフォールトと同じ規則に従います。

---

横取りをしている同じタイプのフォールトは自分自身を横取りできないため、派生したバスフォールトは昇格されません。つまり、破損したスタックによってフォールトが発生した場合、そのフォールトハンドラはスタックへのプッシュが失敗した場合でも実行されます。フォールトハンドラは動作できますが、スタックの内容は破損しています。

### 5.12.2 ローカルフォールトと昇格

ローカルフォールトは、原因によって分類されます。詳細については、P.5-28 表 5-10 を参照して下さい。ローカルフォールト ハンドラが許可されている場合、このハンドラが通常のフォールトをすべて処理します。ただし、ローカルフォールトは次の条件でハードフォールトに昇格されます。

- ローカルフォールト ハンドラで、処理中のものと同じ種類のフォールトが発生した場合。
- ローカルフォールト ハンドラで、同じまたはより優先度の高いフォールトが発生した場合。
- 例外ハンドラで、同じまたはより優先度の高いフォールトが発生した場合。
- ローカルフォールトが許可されていない場合。

ローカルフォールトの一覧を、表 5-10 に示します。

表 5-10 フォールト

フォールト	ビット名	ハンドラ	注	トラップイネーブルビット
リセット	リセットの原因	リセット	任意の形式のリセット	RESETVCATCH
ベクタ読み出しエラー	VECTTBL	HardFault	ベクタテーブル エントリの読み出し時にバスエラーが返されました。	INTERR
uCode スタック プッシュ エラー	STKERR	BusFault	ハードウェアを使用してコンテキストを保存するときに、バスエラーが返されました。	INTERR
uCode スタック プッシュ エラー	MSTKERR	MemManage	ハードウェアを使用してコンテキストを保存するときに、MPU アクセス違反が発生しました。	INTERR
uCode スタック ポップ エラー	UNSTKERR	BusFault	ハードウェアを使用してコンテキストを復元するときに、バスエラーが返されました。	INTERR
uCode スタック ポップ エラー	MUNSKERR	MemManage	ハードウェアを使用してコンテキストを復元するときに、MPU アクセス違反が発生しました。	INTERR
ハードフォールト への昇格	FORCED	HardFault	フォールトが発生し、ハンドラの優先度が現在のものと同じか、より高い。優先度が許可されていないときにフォールト内でフォールトが発生した場合や、構成可能なフォールトが禁止されている場合を含みます。SVC、BKPT、その他のフォールトを含みます。	HARDERR
MPU 不一致	DACCVIOL	MemManage	データアクセスの結果、MPU にアクセス違反またはフォールトが発生しました。	MMERR
MPU 不一致	IACCVIOL	MemManage	命令アドレスが原因で、MPU にアクセス違反またはフォールトが発生しました。	MMERR
プリフェッチエラー	IBUSERR	BusFault	命令フェッチでバスエラーが返されました。実行される場合のみ、フォールトが発生します。分岐シャドウでは、フォールトが発生しても無視されることがあります。	BUSERR

表 5-10 フォールト

フォールト	ビット名	ハンドラ	注	トラップイネーブルビット
正確なデータバスエラー	PRECISERR	BusFault	データアクセスの結果バスエラーが返されました。このエラーは正確で、命令を指し示しています。	BUSERR
不正確なデータバスエラー	IMPRECISERR	BusFault	データアクセスの結果、遅れてバスエラーが返されました。正確な命令はもはや不明です。このフォールトは保留され、非同期です。FORCED を起こしません。	BUSERR
コプロセッサなし	NOCP	UsageFault	コプロセッサが実際に存在しないか、存在ビットがセットされていません。	NOCERR
未定義命令	UNDEFINSTR	UsageFault	未定義命令	STATERR
無効な ISA 状態、例えば Thumb でない状態から命令の実行が試みられた	INVSTATE	UsageFault	無効な EPSR 状態で実行が試みられました。例えば、BX タイプの命令で状態が変更された後などが挙げられます。これには、インタローキング状態を含む、例外からの復帰後の状態が含まれます。	STATERR
許可されていない、または無効なマジックナンバーを持つ、PC=EXC_RETURN への復帰	INVPC	UsageFault	不正な EXC_RETURN の値、EXC_RETURN とスタックされている EPSR の値の不一致、または現在の EPSR が現在アクティブな例外の一覧に含まれていないときの終了により発生した、不正な終了。	STATERR
不正なアンアラインドロード/ストア	UNALIGNED	UsageFault	複数ロード/ストア命令で、ワードアラインドでない位置へのアクセスを試みたときに発生します。UNALIGN_TRP ビットを使用して、サイズにアラインしていない任意のロード/ストアについて発生を許可できます。	CHKERR
0 による除算	DIVBYZERO	UsageFault	SDIV または UDIV 命令が実行され、除数が 0 であり、DIV_0_TRP ビットがセットされているときに発生します。	CHKERR
SVC	-	SVCall	システム要求（サービス呼び出し）	-

デバッグフォールトを、表 5-11 に示します。

表 5-11 デバッグフォールト

フォールト	フラグ	注	トラップイネーブルビット
内部停止要求	HALTED	ステップやコア停止などからの NVIC 要求	-
ブレイクポイント	BKPT	パッチされた命令や FPB からの SW ブレイクポイント	-
ウォッチポイント	DWTTRAP	DWT でのウォッチポイント一致	-
外部	EXTERNAL	EDBGRQ ラインがアサートされました。	-
ベクタキャッチ	VCATCH	ベクタキャッチがトリガされました。対応する FSR には、例外の主な原因が含まれています。	VC_xxx ビットまたは RESETVCATCH がセットされている

5.12.3    フォールトステータス レジスタとフォールトアドレス レジスタ

各フォールトについて、フォールトステータス レジスタに対応するフラグが存在します。これらは、次のレジスタです。

- 3 つの構成可能なフォールトハンドラに対応する、3 つの構成可能なフォールトステータス レジスタ
- 1 つのハードフォールト ステータスレジスタ
- 1 つのデバッグフォールト ステータスレジスタ

原因に応じて、5 つのステータスレジスタのうちの 1 ビットがセットされます。

フォールトアドレス レジスタ (FAR)(Fault Address Register) は 2 つあります。

- バスフォールトアドレス レジスタ (BFAR)(Bus Fault Address Register)
- メモリフォールトアドレス レジスタ (MFAR)(Memory Fault Address Register)

フォールトステータス レジスタの対応するフラグは、フォールトアドレス レジスタのアドレスがいつ有効であるかを示しています。

——— 注 ———

BFAR と MFAR は同じ物理レジスタです。このため、BFARVALID ビットと MFARVALID ビットは相互に排他的です。

フォールトステータス レジスタと、2 つのフォールトアドレス レジスタを、P.5-31 表 5-12 に示します。

表 5-12 フォールトステータス レジスタとフォールトアドレス レジスタ

ステータスレジスタ名	ハンドラ	アドレスレジスタ名	説明
HFSR	ハードフォールト	-	昇格と特殊
MMSR	メモリ管理	MMAR	MPU フォールト
BFSR	バスフォールト	BFAR	バスフォールト
UFSR	用法フォールト	-	用法フォールト
DFSR	デバッグモニタまたは停止	-	デバッグトラップ

5.13 起動レベル

アクティブな例外がない場合、プロセッサはスレッドモードです。ISR またはフォールトハンドラがアクティブな場合、プロセッサはハンドラモードに入ります。各起動レベルにおける特権とスタックの一覧を、表 5-13 に示します。

表 5-13 各起動レベルでの特権とスタック

アクティブな例外	起動レベル	特権	スタック
なし	スレッドモード	特権またはユーザ	メインまたはプロセス
ISR がアクティブ	非同期の横取りレベル	特権	メイン
フォールトハンドラがアクティブ	同期の横取りレベル	特権	メイン
リセット	スレッドモード	特権	メイン

すべての例外タイプに関する遷移規則と、それらのアクセス規則およびスタックモデルとの関係の要約を、表 5-14 に示します。

表 5-14 例外の遷移

アクティブな例外	例外をトリガするイベント	遷移タイプ	特権	スタック
リセット	リセット信号	スレッド	特権またはユーザ	メインまたはプロセス
ISR <sup>a</sup> または NMI <sup>b</sup>	保留セットソフトウェア命令、またはハードウェア信号	非同期の横取り	特権	メイン
フォールト				
ハードフォールト	昇格			
バスフォールト	メモリアクセス エラー	同期の横取り	特権	メイン
非 CP <sup>c</sup> フォールト	CP 不在のアクセス			
未定義命令フォールト	未定義命令			
デバッグモニタ	停止が許可されていないときのデバッグイベント	同期	特権	メイン
SVC <sup>d</sup>	SVC 命令			
外部割り込み				

- a. 割り込み処理ルーチン。
- b. マスク不能割り込み。
- c. コプロセッサ。
- d. ソフトウェア割り込み。

例外のサブタイプの遷移を、表 5-15 に示します。

表 5-15 例外のサブタイプの遷移

起動するサブタイプ	例外をトリガするイベント	起動	優先度への影響
スレッド	リセット信号	非同期	即時、スレッドの優先度は最低
ISR/NMI	HW 信号またはセット保留	非同期	優先度に応じて横取りまたはテールチェイン
モニタ	デバッグイベント <sup>a</sup>	同期	優先度が現在のものと同じか低い場合、ハードフォールト
SVCall	SVC 命令	同期	優先度が現在のものと同じか低い場合、ハードフォールト
PendSV	ソフトウェア保留要求	チェイン	優先度に応じて横取りまたはテールチェイン
UsageFault	未定義命令	同期	優先度が現在のものと同じか高い場合、ハードフォールト
NoCpFault	存在しない CP へのアクセス	同期	優先度が現在のものと同じか高い場合、ハードフォールト
BusFault	メモリアクセスエラー	同期	優先度が現在のものと同じか高い場合、ハードフォールト
MemManage	MPU 不一致	同期	優先度が現在のものと同じか高い場合、ハードフォールト
HardFault	昇格	同期	NMI を除くすべての例外より高い
FaultEscalate	構成可能なフォールトハンドラからの昇格要求	チェイン	ローカルハンドラの優先度をハードフォールトと同じにブーストし、構成可能なフォールトハンドラへ復帰してチェインできるようにします。

a. 停止が許可されていないとき。

5.14 フローチャート

このセクションでは、次の条件における割り込みのフローについて説明します。

- 割り込み処理
- 横取り：P.5-35
- 復帰：P.5-35

5.14.1 割り込み処理

より優先度が高い割り込みによって横取りされるまで、命令がどのように実行されるかを図 5-6 に示します。

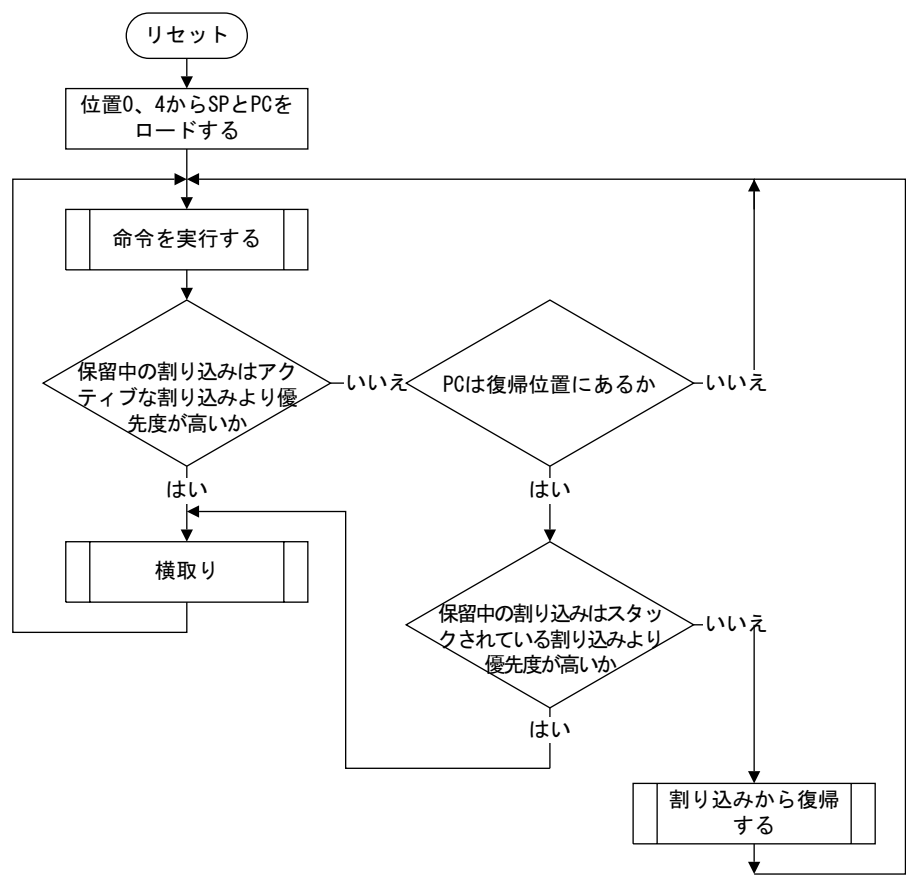


図 5-6 割り込み処理のフローチャート



### 5.14.2 横取り

例外が現在の ISR を横取りするときの動作を、図 5-7 に示します。

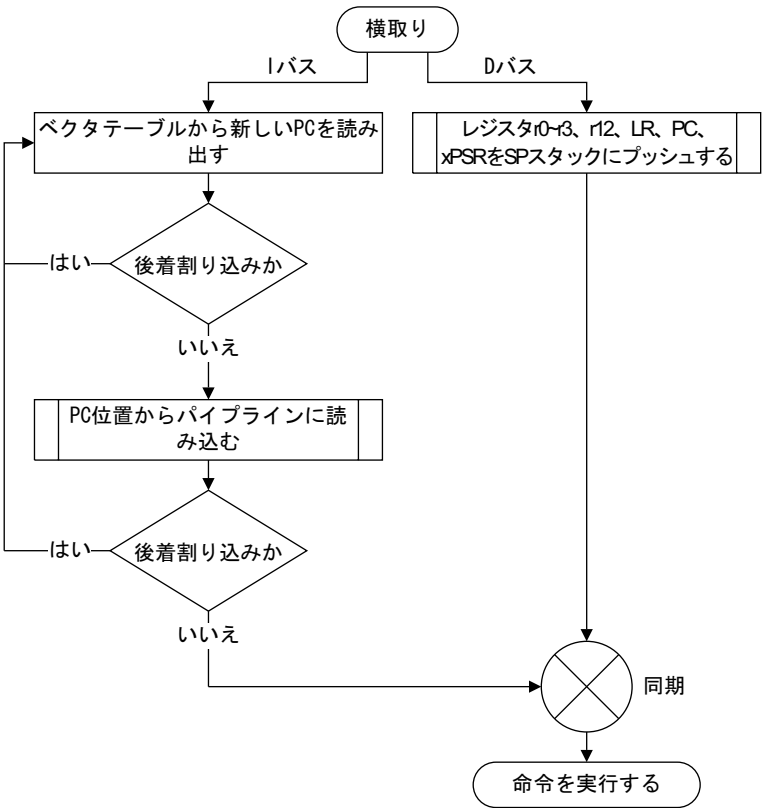


図 5-7 横取りのフローチャート

### 5.14.3 復帰

プロセッサがスタックされている ISR を復元する動作、またはスタックされている ISR よりも優先度が高い後着割り込みへのテールチェーンを行うときの動作を、P.5-36 図 5-8 に示します。

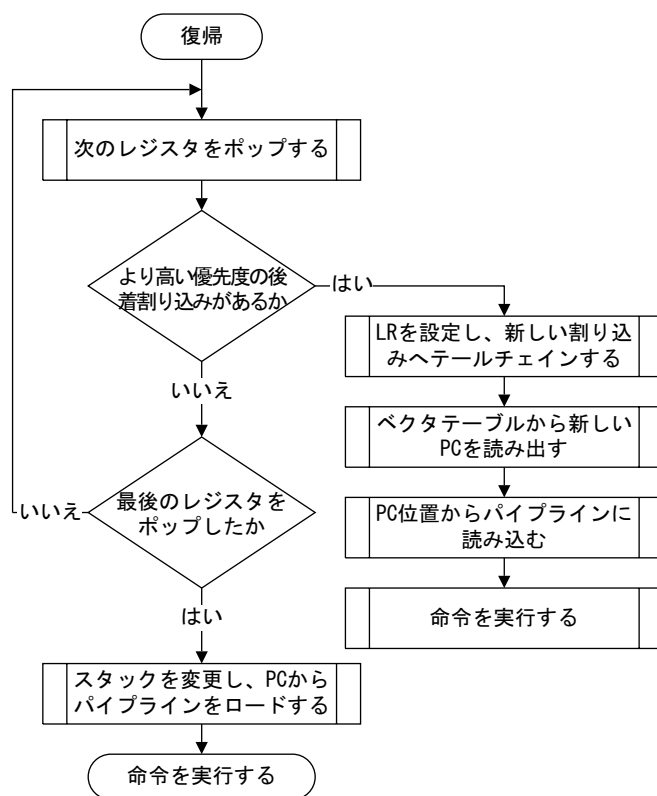


図 5-8 割り込みからの復帰のフローチャート

## 第 6 章

# クロックとリセット

本章では、プロセッサのクロックとリセットについて説明します。本章は以下のセクションから構成されています。

- クロック : P.6-2
- リセット : P.6-4
- *Cortex-M3* のリセットモード : P.6-5

6.1 クロック

プロセッサには、クロック入力があります。プロセッサのクロックの説明を、表 6-1 に示します。

表 6-1 Cortex-M3 プロセッサのクロック

クロック	ドメイン	説明
FCLK	プロセッサ	フリーランのプロセッサクロックで、割り込みのサンプリングとデバッグ回路へのクロック供給に使用されます。このクロックは、プロセッサがスリープ状態のときに、割り込みのサンプリングとスリープイベントのトレースが可能であることを保証するために使用されます。
HCLK	プロセッサ	プロセッサクロック
DAPCLK	プロセッサ	デバッグポートのアドバンストハイパフォーマンスバス アクセスポート (AHB-AP)(Advanced High-Performance Bus Access Port) クロック

FCLK と HCLK は互いに同期しています。FCLK は HCLK のフリーランクロックです。詳細については、第 7 章 「電力管理」を参照して下さい。FCLK と HCLKHCLK 間はプロセッサへのレイテンシが等しくなるようにバランスしている必要があります。

プロセッサには、デバッグとトレース用のコンポーネントが組み込まれています。お使いのマクロセルには、表 6-2 に示されているクロックの一部またはすべてを持っている可能性があります。

表 6-2 Cortex-M3 マクロセルのクロック

クロック	ドメイン	説明
TRACECLKIN	TPIU	TPIU の出力のクロック
DBGCLK	SW-DP	デバッグクロック
SWCLKTCK	SWJ-DP	デバッグクロック

SWCLKTCK は、SWJ-DP のデバッグインタフェース ドメインのクロックです。これは、JTAG モードでは TCK と等価で、シリアルワイヤ モードではシリアルワイヤ クロックです。このクロックは、他のすべてのクロックと非同期です。DBGCLK は、SW-DP のデバッグインタフェース ドメインのクロックです。このクロックは、他のクロックと非同期です。

TRACECLKIN はトレースポート インタフェースユニット (TPIU) のリファレンス クロックです。このクロックは、他のクロックと非同期です。

---

**注**

---

**SWCLKTCK**、**DBGCLK**、**TRACECLKIN** の駆動が必要になるのは、実装にそれぞれシリアルワイヤ JTAG デバッグポート (SWJ-DP)、シリアルワイヤ デバッグポート SW-DP)、TPIU ブロックが含まれている場合のみです。それ以外の場合、クロック入力をオフに接続する必要があります。

---

**注**

---

プロセッサには、**STCLK** 入力も組み込まれています。このポートはクロックではありません。これは SysTick カウンタへのリファレンス入力で、周波数は **FCLK** の半分未満の必要があります。**STCLK** は、プロセッサによって内部的に **FCLK** と同期されます。

---

6.2 リセット

プロセッサには3つのリセット入力があります。リセット入力の説明を、表 6-3 に示します。

表 6-3 リセット入力

リセット入力	説明
PORESETn	SWJ-DP を除いて、プロセッサシステム全体をリセットします。
SYSRESETn	以下の回路のデバッグ回路を除いて、プロセッサシステム全体をリセットします。 <ul style="list-style-type: none"><li>•     ネスト型ベクタ割り込みコントローラ (NVIC)</li><li>•     フラッシュパッチおよびブレークポイントユニット (FPB)</li><li>•     データウォッチポイントおよびトレース (DWT)</li><li>•     計装トレース マクロセル (ITM)</li><li>•     AHB-AP</li></ul>
nTRST	SWJ-DP リセット

—— 注 ——

nTRST は SWJ-DP をリセットします。お使いの実装に SWJ-DP が含まれていない場合、このリセットをオフに接続する必要があります。

6.3 Cortex-M3 のリセットモード

プロセッサの設計に存在するリセット信号によって、設計のさまざまなパーツを個別にリセットできます。リセット信号、およびその組み合わせと考えられる用途を表 6-4 に示します。

表 6-4 リセットモード

リセット モード	SYSRESETn	nTRST	PORESETn	用途
パワーオン リセット	x	0	0	パワーオン時のリセットで、完全なシステムリ セットを行います。コールドリセット
システム リセット	0	x	1	デバッグを除く、プロセッサコアとシステムコン ポーネントをリセットします。
SWJ-DP リセット	1	0	1	SWJ-DP 回路をリセットします。

——— 注 ———

**PORESETn** は、**SYSRESETn** 回路のスーパーセットをリセットします。

6.3.1 パワーオンリセット

マクロセルのリセット信号を、P.6-6 図 6-1 に示します。

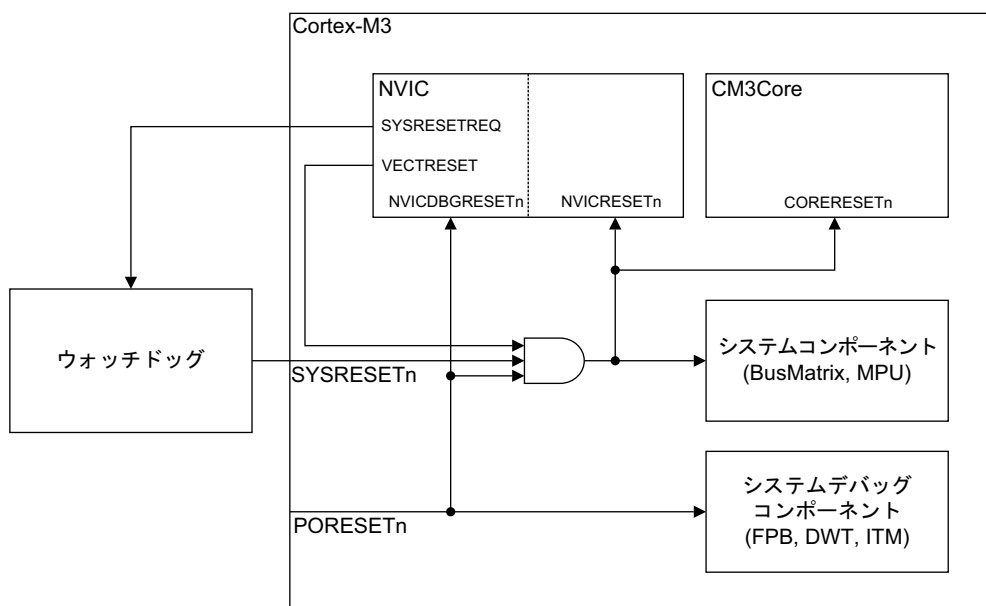


図 6-1 リセット信号

システムの電力が最初にオンになったときに、パワーオンリセットすなわちコールドリセットをプロセッサに適用する必要があります。パワーオンリセットの場合、リセット信号 **PORESETn** の立ち下がりエッジは **HCLK** と同期している必要はありません。**PORESETn** はプロセッサ内で同期化されるため、この信号を同期化する必要はありません。パワーオンリセットの使用法を、図 6-2 に示します。P.6-7 図 6-3 には、リセットがプロセッサ内で同期していることが示されています。

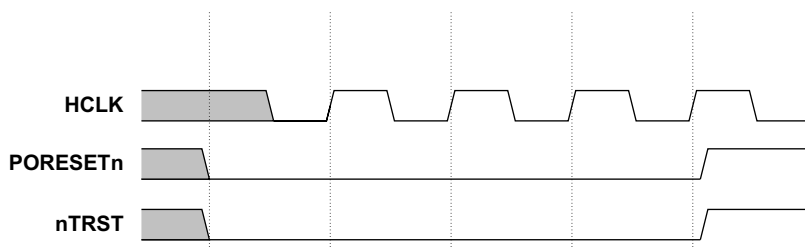


図 6-2 パワーオンリセット

正しいリセット動作を保証するため、リセット信号は少なくとも 3 **HCLK** サイクルの間アサートすることをお勧めします。内部でのリセットの同期を、P.6-7 図 6-3 に示します。



---

**注**


---

外部デバッガが付属していない場合、何らかの外部のウォッチドッグ回路には、Cortex-M3 システムからの LOCKUP を含めることを検討する必要があります。

---

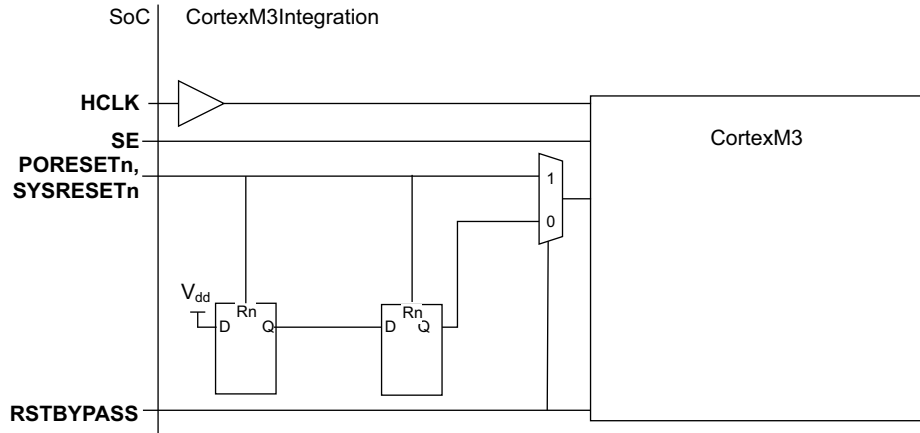


図 6-3 内部リセットの同期

### 6.3.2 システムリセット

システムリセットすなわちウォームリセットは、NVIC のデバッグ回路、FPB、DWT、ITM を除く大部分のマクロセルを初期化します。システムリセットは通常、例えばウォッチドッグリセットのように、一定期間稼動し続けているシステムをリセットします。

**SYSRESETn** はプロセッサの外部で同期する必要があります。CortexM3Integration で提供されているリセット同期の例を、図 6-3 に示します。

Cortex-M3 は、アプリケーション割り込みおよびリセット制御レジスタ (Application Interrupt and Reset Control Register) の **SYSRESETREQ** ビットがセットされるとアサートされる **SYSRESETREQ** 信号を外部に出力しています。例えば、P.6-6 図 6-1 に示すように、この信号をウォッチドッグタイマへの入力として使用できます。

### 6.3.3 SWJ-DP リセット

**nTRST** リセットは、SWJ-DP コントローラの状態を初期化します。**nTRST** リセットは一般に、RealView™ICE モジュールによって、デバッガをシステムへホットプラグ接続するために使用されます。

**nTRST** は、プロセッサの通常の動作に影響を与えずに、SWJ-DP コントローラを初期化できます。

SWJ-DP は同期を実行しないため、**nTRST** 信号は **SWCLKTCK** クロックに同期してアサートする必要があります。

#### 6.3.4 SW-DP リセット

SW-DP は **DBGRESETn** でリセットされます。このリセットは、**DBGCLK** と同期している必要があります。

#### 6.3.5 通常動作

通常動作中は、プロセッサリセットとパワーオンリセットはいずれもアサートされません。SWJ-DP ポートが使用されていない場合、**nTRST** の値は関係しません。

# 第 7 章

## 電力管理

本章では、プロセッサの電力管理機能について説明します。本章は以下のセクションから構成されています。

- 電力管理について：P.7-2
- システム電力管理：P.7-3

## 7.1 電力管理について

プロセッサは、ゲート付きクロックを広く活用して、使用されていない機能を無効にし、使用されていない機能ブロックへの入力を無効にして、アクティブで使用されている回路のみが動的な電力を消費するようにします。

ARMv7-M アーキテクチャでは、消費電力をより少なくするために、Cortex-M3 とシステムクロックを停止することができるシステム スリープモードがサポートされています。詳細については、P.7-3 「システム電力管理」を参照して下さい。

## 7.2 システム電力管理

Cortex-M3 システムの電力状態は、システム制御レジスタ (P.8-24 「システム制御レジスタ」) への書き込みにより制御されます。サポートされているスリープモードを、表 7-1 に示します。

表 7-1 サポートされているスリープモード

スリープ機構	説明
Sleep-now	割り込み待ち (WFI) またはイベント待ち (WFE) 命令は、Sleep-now モデルを要求します。これらの命令が実行されると、ネスト型ベクタ割り込みコントローラ (NVIC) は、他の例外を保留して、プロセッサを低電力状態に移行します。 <sup>a</sup>
Sleep-on-exit	システム制御レジスタの SLEEPONEXIT ビットがセットされている場合、最も優先度が低い ISR から退出直後に、プロセッサは低電力状態に移行します。プロセッサは、レジスタをポップせずに低電力状態へ移行し、レジスタをプッシュする必要なしに後続の例外を処理します。コアは、次の例外が保留されるまでスリープ状態に置かれます。これは自動的な WFI モードです。
<p style="text-align: center;">——— 注 ———</p> <p>Sleep-on-exit は、デバッグなど各種の状況下で、ベースに戻ることがあります。このため、アイドルループやアイドルスレッドなどのベースコードを用意しておく必要があります。</p>	
Deep-sleep	Deep-sleep は、Sleep-now および Sleep-on-exit と組み合わせて使用されます。システム制御レジスタの SLEEPDEEP ビットがセットされている場合、プロセッサはシステムに、より深いスリープ状態への移行が可能なことを通知します。

- a. WFI 命令は、例外がアクティブにならなくても完了することがあります。この命令を、例外の発生を検出する方法として使用しないで下さい。WFI は通常、スレッドモードのアイドルループで使用されます。WFI、WFE、BASEPRI、PRIMASK の詳細については、『ARMv7-M アーキテクチャリファレンス マニュアル』を参照して下さい。

プロセッサは、いつプロセッサがスリープしているのかを通知するため、次の信号を出力しています。

**SLEEPING** この信号は、Sleep-now または Sleep-on-exit モードでアサートされ、プロセッサへのクロックを停止できることを示します。新しい割り込みを受信すると、NVIC はコアをスリープから解除し、この信号をデアサートします。**SLEEPING** の使用例を、P.7-4 「**SLEEPING**」に示します。

### SLEEPDEEP

システム制御レジスタの SLEEPDEEP ビットがセットされている場合、Sleep-now または Sleep-on-exit モードでこの信号がアサートされます。この信号はクロックマネージャにまで配線され、プロセッサおよびフェーズロックドループ (PLL) を含むシステムコンポーネントをゲートして、より電力を節約することができます。新しい割り込みを受信すると、ネスト型ベクタ割り込みコントローラ (NVIC) はこの信号をデアサートし、クロックマネージャからクロックが安定しているという通知を受けた時点で、コアをスリープから解除し、**SLEEPDEEP** の使用例を、P.7-4 「**SLEEPDEEP**」に示します。

## 7.2.1 SLEEPING

低電力状態において **SLEEPING** によりプロセッサへの **HCLK** クロックをゲートすることで消費電力を低減する方法例を、図 7-1 に示します。必要に応じて、**SLEEPING** を使用して、他のシステムコンポーネントをゲートすることも可能です。

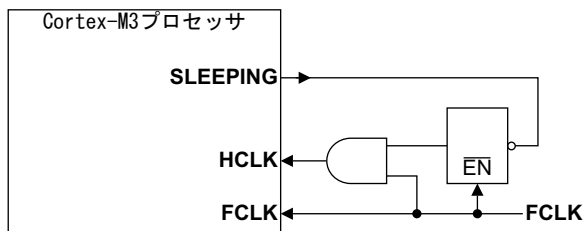


図 7-1 SLEEPING 電力制御の例

割り込みを検出するため、プロセッサはフリーラン **FCLK** を常に受信している必要があります。**FCLK** は次の回路にクロックを供給します。

- 割り込みを検出するための、NVIC 内の小さな回路。
- データウォッチポイントおよびトレース (DWT) ブロックと計装トレース マクロセル (ITM) ブロック。これらのブロックは、許可されていればスリープ中にトレースパケットを生成することができます。デバッグ例外およびモニタ制御レジスタの **TRCENA** ビットが有効な場合、これらのブロックの電力消費は最小化されます。詳細については、P.10-8「デバッグ例外およびモニタ制御レジスタ」を参照して下さい。

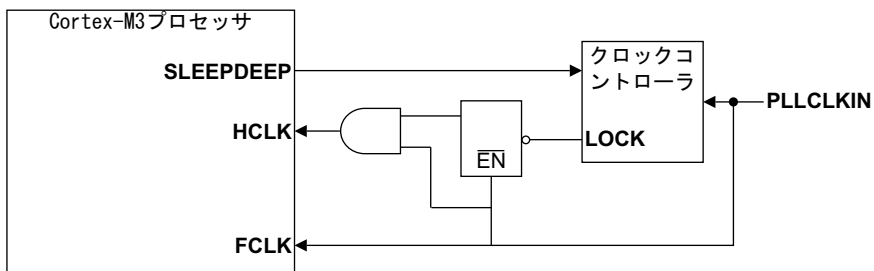
**SLEEPING** がアサートされている間は、**FCLK** の周波数を下げることができます。

### —— 注 ——

図 7-1 のクロックゲート方式を使用して **HCLK** を止めると、デバッグアクセスが禁止されます。CoreSight デバッグポート (DP) から供給される起動信号を使って、システムは図 7-1 のクロックゲート回路をバイパスすることができます。

## 7.2.2 SLEEPDEEP

低電力状態において **SLEEPDEEP** を使用してクロックコントローラを停止することで、消費電力を低減する方法を、P.7-5 図 7-2 に示します。低電力状態を抜けるとき、**LOCK** シグナルは、PLL が安定していて、Cortex-M3 クロックを有効にしても安全であることを示します。これにより、クロックが安定するまで、プロセッサが再起動しないことが保証されます。

**図 7-2 SLEEPDEEP 電力制御の例**

割り込みを検出するには、プロセッサが低電力状態でフリーランの **FCLK** を受信している必要があります。**SLEEPDEEP** がアサートされている間は、**FCLK** の周波数を下げることができます。





## 第 8 章

# ネスト型ベクタ割り込みコントローラ

本章では、ネスト型ベクタ割り込みコントローラ(NVIC)について説明します。本章は以下のセクションから構成されています。

- NVIC について: P.8-2
- NVIC のプログラマモデル: P.8-3
- レベル割り込みとパルス割り込みの比較: P.8-41

## 8.1 NVIC について

NVIC には次の特徴があります。

- レイテンシが短い例外および割り込み処理を実現する
- 電力管理を制御する
- システム制御レジスタ (System Control Register) を実装している

NVIC は最大 240 の割り込みをサポートし、各割り込みには最大 256 レベルの優先度を設定でき、動的に優先度を再設定することもできます。NVIC とプロセッサコア インタフェースは密接に結合されているため、レイテンシの短い割り込み処理や、後着割り込みの効率的な処理が可能です。NVIC は、割り込みのテールチェーンを可能にするため、スタックされた（ネストされた）割り込みの情報を保持します。

NVIC への完全なアクセスは特権モードからのみ実行できますが、構成制御レジスタを許可すると、ユーザモードで割り込みを保留することもできます (P.8-25「構成制御レジスタ」)。それ以外のユーザモードでのアクセスは、すべてバスフォールトを引き起こします。

すべての NVIC レジスタは、特に明記されていない限り、バイト、ハーフワード、ワードを使用してアクセスできます。

すべての NVIC レジスタとシステムデバッグ レジスタは、プロセッサのエンディアン形式の状態にかかわらず、リトルエンディアンです。

プロセッサ例外の処理については、第 5 章「例外」を参照して下さい。

## 8.2 NVIC のプログラマモデル

このセクションでは、NVIC レジスタのリストを示し、説明を行います。このセクションは以下の項目から構成されています。

- NVIC レジスタのマップ
- NVIC レジスタの説明：P.8-7

### 8.2.1 NVIC レジスタのマップ

NVIC レジスタの一覧を、表 8-1 に示します。NVIC はシステム制御空間に含まれています。NVIC 空間は次のように分けられています。

- 0xE000E000 - 0xE000E00F 割り込みタイプレジスタ
- 0xE000E010 - 0xE000E0FF システムタイマ
- 0xE000E100 - 0xE000ECFF NVIC
- 0xE000ED00 - 0xE000ED8F システム制御ブロック（次の項目が含まれます）
  - CPUID
  - システム制御、コンフィギュレーション、ステータス
  - フォールト通知
- 0xE000EF00 - 0xE000EF0F ソフトウェアトリガ例外レジスタ
- 0xE000EFD0 - 0xE000EFFF.ID 空間

表 8-1 NVIC レジスタ

レジスタ名	タイプ	アドレス	リセット時の値	ページ
割り込み制御タイプレジスタ (Interrupt Control Type Register)	読み出し専用	0xE000E004	<sup>a</sup>	P. 8-7
SysTick 制御およびステータスレジスタ (SysTick Control and Status Register)	読み出し / 書き込み	0xE000E010	0x00000000	P. 8-8
SysTick リロード値レジスタ (SysTick Reload Value Register)	読み出し / 書き込み	0xE000E014	予測不能	P. 8-9
SysTick 現在値レジスタ (SysTick Current Value Register)	読み出し / 書き込みクリア	0xE000E018	予測不能	P. 8-10
SysTick 較正值レジスタ (SysTick Calibration Value Register)	読み出し専用	0xE000E01C	STCALIB	P. 8-11
IRQ 0 ～ 31 イネーブルセット レジスタ (Set Enable Register)	読み出し / 書き込み	0xE000E100	0x00000000	P. 8-12
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
IRQ 224 ～ 239 イネーブルセット レジスタ	読み出し / 書き込み	0xE000E11C	0x00000000	P. 8-12
IRQ 0 ～ 31 イネーブルクリア レジスタ (Clear Enable Register)	読み出し / 書き込み	0xE000E180	0x00000000	P. 8-13
.	.	.	.	.

表 8-1 NVIC レジスタ

レジスタ名	タイプ	アドレス	リセット時の値	ページ
.	.	.	.	.
.	.	.	.	.
IRQ 224 ~ 239 イネーブルクリア レジスタ	読み出し / 書き込み	0xE000E19C	0x00000000	P. 8-13
IRQ 0 ~ 31 保留セットレジスタ (Set Pending Register)	読み出し / 書き込み	0xE000E200	0x00000000	P. 8-14
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
IRQ 224 ~ 239 保留セットレジスタ	読み出し / 書き込み	0xE000E21C	0x00000000	P. 8-14
IRQ 0 ~ 31 保留クリアレジスタ (Clear Pending Register)	読み出し / 書き込み	0xE000E280	0x00000000	P. 8-14
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
IRQ 224 ~ 239 保留クリアレジスタ	読み出し / 書き込み	0xE000E29C	0x00000000	P. 8-14
IRQ 0 ~ 31 アクティブビット レジスタ (Active Bit Register)	読み出し専用	0xE000E300	0x00000000	P. 8-15
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
IRQ 224 ~ 239 アクティブビット レジスタ	読み出し専用	0xE000E31C	0x00000000	P. 8-15
IRQ 0 ~ 31 優先度レジスタ (Priority Register)	読み出し / 書き込み	0xE000E400	0x00000000	P. 8-16
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
IRQ 236 ~ 239 優先度レジスタ	読み出し / 書き込み	0xE000E4F0	0x00000000	P. 8-15
CPUID ベースレジスタ (Base Register)	読み出し専用	0xE000ED00	0x411FC231	P. 8-17
割り込み制御状態レジスタ (Interrupt Control State Register)	読み出し / 書き込み、 または読み出し専用	0xE000ED04	0x00000000	P. 8-18
ベクタテーブル オフセットレジスタ (Vector Table Offset Register)	読み出し / 書き込み	0xE000ED08	0x00000000	P. 8-20
アプリケーション割り込み / リセット制御 レジスタ (Application Interrupt/Reset Control Register)	読み出し / 書き込み	0xE000ED0C	0x00000000 <sup>b</sup>	P. 8-21

表 8-1 NVIC レジスタ

レジスタ名	タイプ	アドレス	リセット時の値	ページ
システム制御レジスタ (System Control Register)	読み出し / 書き込み	0xE000ED10	0x00000000	P. 8-24
構成制御レジスタ (Configuration Control Register)	読み出し / 書き込み	0xE000ED14	0x00000000	P. 8-25
システムハンドラ 4 ~ 7 優先度レジスタ (System Handlers Priority Register)	読み出し / 書き込み	0xE000ED18	0x00000000	P. 8-27
システムハンドラ 8 ~ 11 優先度レジスタ	読み出し / 書き込み	0xE000ED1C	0x00000000	P. 8-27
システムハンドラ 12 ~ 15 優先度レジスタ	読み出し / 書き込み	0xE000ED20	0x00000000	P. 8-27
システムハンドラ制御および状態レジスタ (System Handler Control and State Register)	読み出し / 書き込み	0xE000ED24	0x00000000	P. 8-28
構成可能フォールトステータス レジスタ (Configurable Fault Status Register)	読み出し / 書き込み	0xE000ED28	0x00000000	P. 8-31
ハードフォールトステータス レジスタ (Hard Fault Status Register)	読み出し / 書き込み	0xE000ED2C	0x00000000	P. 8-36
デバッグフォールトステータス レジスタ (Debug Fault Status Register)	読み出し / 書き込み	0xE000ED30	0x00000000	P. 8-37
メモリ管理アドレスレジスタ (Mem Manage Address Register)	読み出し / 書き込み	0xE000ED34	予測不能	P. 8-38
バスフォールトアドレスレジスタ (Bus Fault Address Register)	読み出し / 書き込み	0xE000ED38	予測不能	P. 8-39
補助フォールトステータス レジスタ (Auxiliary Fault Status Register)	読み出し / 書き込み	0xE000ED3C	0x00000000	P. 8-39
PFR0 : プロセッサ機能レジスタ 0 (Processor Feature Register)	読み出し専用	0xE000ED40	0x00000030	-
PFR1 : プロセッサ機能レジスタ 1	読み出し専用	0xE000ED44	0x00000200	-
DFR0 : デバッグ機能レジスタ 0 (Debug Feature Register)	読み出し専用	0xE000ED48	0x00100000	-
AFR0 : 補助機能レジスタ 0 (Auxiliary Feature Register)	読み出し専用	0xE000ED4C	0x00000000	-
MMFR0 : メモリモデル機能レジスタ 0 (Memory Model Feature Register)	読み出し専用	0xE000ED50	0x00000030	-
MMFR1 : メモリモデル機能レジスタ 1	読み出し専用	0xE000ED54	0x00000000	-
MMFR2 : メモリモデル機能レジスタ 2	読み出し専用	0xE000ED58	0x00000000	-
MMFR3 : メモリモデル機能レジスタ 3	読み出し専用	0xE000ED5C	0x00000000	-
ISAR0 : ISA 機能レジスタ 0	読み出し専用	0xE000ED60	0x01141110	-
ISAR1 : ISA 機能レジスタ 1	読み出し専用	0xE000ED64	0x02111000	-

表 8-1 NVIC レジスタ

レジスタ名	タイプ	アドレス	リセット時の値	ページ
ISAR2 : ISA 機能レジスタ 2	読み出し専用	0xE000ED68	0x21112231	-
ISAR3 : ISA 機能レジスタ 3	読み出し専用	0xE000ED6C	0x01111110	-
ISAR4 : ISA 機能レジスタ 4	読み出し専用	0xE000ED70	0x01310102	-
ソフトウェアトリガ割り込みレジスタ (Software Trigger Interrupt Register)	書き込み専用	0xE000EF00	-	P. 8-40
ペリフェラル識別レジスタ (PID4)	読み出し専用	0xE000EFD0	0x04	-
ペリフェラル識別レジスタ (PID5)	読み出し専用	0xE000EFD4	0x00	-
ペリフェラル識別レジスタ (PID6)	読み出し専用	0xE000EFD8	0x00	-
ペリフェラル識別レジスタ (PID7)	読み出し専用	0xE000EFD8	0x00	-
ペリフェラル識別レジスタ ビット [7:0] (PID0)	読み出し専用	0xE000EFE0	0x00	-
ペリフェラル識別レジスタ ビット [15:8] (PID1)	読み出し専用	0xE000EFE4	0xB0	-
ペリフェラル識別レジスタ ビット [23:16] (PID2)	読み出し専用	0xE000EFE8	0x1B	-
ペリフェラル識別レジスタ ビット [31:24] (PID3)	読み出し専用	0xE000EFEC	0x00	-
コンポーネント識別レジスタビット [7:0] (CID0)	読み出し専用	0xE000EFF0	0x0D	-
コンポーネント識別レジスタ ビット [15:8] (CID1)	読み出し専用	0xE000EFF4	0xE0	-
コンポーネント識別レジスタ ビット [23:16] (CID2)	読み出し専用	0xE000EFF8	0x05	-
コンポーネント識別レジスタ ビット [31:24] (CID3)	読み出し専用	0xE000EFFC	0xB1	-

- リセット時の値は、定義されている割り込みの数によって異なります。
- ビット [10:8] はリセットされます。エンディアン形式ビット（ビット [15]）は、**BIGEND** のサンプリングによりリセット時にセットされます。

### 8.2.2 NVIC レジスタの説明

以下のセクションでは、NVIC レジスタの用法について説明します。

- 注

メモリ保護ユニット (MPU) レジスタ、およびデバッグレジスタについては、それぞれ第 9 章「メモリ保護ユニット」と第 10 章「コアデバッグ」を参照して下さい。

### 割り込みコントローラタイプレジスタ (Interrupt Controller type Register)

NVIC がサポートしている割り込み線の数を確認するには、割り込みコントローラタイプレジスタを読み込みます。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE000E004

アクセス 読み出し専用

**リセット時の状態** そのプロセッサ実装で定義されている割り込みの数に依存します。

割り込みコントローラタイプレジスタのフィールドを図 8-1 に示します。

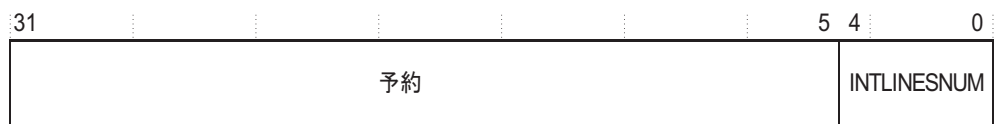


図 8-1 割り込みコントローラタイプレジスタのビット割り当て

割り込みコントローラタイプレジスタのフィールドの説明を、表 8-2 に示します。

表 8-2 割り込みコントローラタイプレジスタのビット割り当て

ビット	フィールド	機能
[31:5]	-	予約
[4:0]	INTLINESNUM	割り込み線の合計数で、32 個のグループ単位で示されます。 b00000 = 0...32 <sup>a</sup> b00001 = 33...64 b00010 = 65...96 b00011 = 97...128 b00100 = 129...160 b00101 = 161...192 b00110 = 193...224 b00111 = 225...256a

a. プロセッサは、1 ～ 240 の外部割り込みのみをサポートします。

SysTick 制御およびステータスレジスタ

SysTick 機能を有効にするには、SysTick 制御およびステータスレジスタを使用します。  
このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス                    0xE000E010  
アクセス                    読み出し / 書き込み  
リセット時の状態        0x00000000

SysTick 制御およびステータスレジスタのフィールドを、図 8-2 に示します。

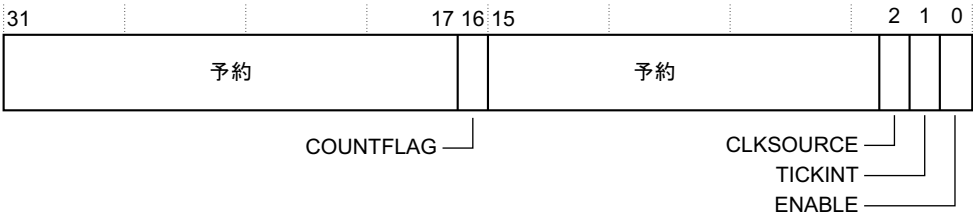


図 8-2 SysTick 制御およびステータスレジスタのビット割り当て



SysTick 制御およびステータスレジスタのフィールドの説明を、表 8-3 に示します。

**表 8-3 SysTick 制御およびステータスレジスタのビット割り当て**

ビット	フィールド	機能
[31:17]	-	予約
[16]	COUNTFLAG	最後の読み出しの後にタイマが 0 になった場合、1 を返します。アプリケーションにより SysTick 制御およびステータスレジスタのどの部分でも読み出されると、クリアされます。デバッガが DAP を使用して読み出したとき、AHB-AP 制御レジスタの MasterType ビットが 0 にセットされている場合だけは、このビットは読み出し時にクリアされます。それ以外の場合、COUNTFLAG ビットはデバッガ読み出しにより変更されません。
[2]	CLKSOURCE	0 = 外部参照クロック 1 = コアクロック 参照クロックが提供されない場合は、コアクロックと同じクロックが供給されるように 1 のまま保持します。コアクロックには、参照クロックと比べて最低でも 2.5 倍以上のスピードが要求されます。この要件が満たされない場合、カウント値は予測不能です。
[1]	TICKINT	1 = 0 までカウントダウンすると、SysTick ハンドラを保留します。 0 = 0 までカウントダウンしても、SysTick ハンドラを保留しません。ソフトウェアは、COUNTFLAG を使用して、0 までカウントされたかどうかを判断できます。
[0]	ENABLE	1 = カウンタはマルチショット方式で動作します。つまり、カウンタにリロード値がロードされてから、カウントダウンが開始されます。0 になると、COUNTFLAG を 1 にセットします。また、オプションで、TICKINT に基づいて SysTick ハンドラを保留することもできます。その後、リロード値を再びロードして、カウントを開始します。 0 = カウンタは禁止されています。

## SysTick リロード値レジスタ

カウンタが 0 になったときに現在値レジスタへロードされる開始値は、SysTick リロード値レジスタで指定します。これは、1 ~ 0x00FFFFFF の任意の値です。開始値を 0 にすることもできますが、SysTick 割り込みと COUNTFLAG は 1 から 0 へのカウントの際にアクティブとなるため、この値は無効です。

このため、このレジスタがマルチショットタイマとして動作する場合、N + 1 回目のクロックパルス (N は 1 ~ 0x00FFFFFF の任意の値) ごとにアクティブとなります。したがって、ティック割り込みを 100 クロックパルスごとに発生する必要がある場合、RELOAD に 99 を書き込む必要があります。ティック割り込みごとに新しい値が書き込まれる場合は、シングルショットとして扱われるので、実際のカウントダウンを書き込む必要があります。例えば、400 クロックパルス後にティックを発生する必要がある場合は、RELOAD に 400 を書き込む必要があります。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス	0xE000E014
アクセス	読み出し / 書き込み
リセット時の状態	予測不能

SysTick リロード値レジスタのフィールドを、図 8-3 に示します。

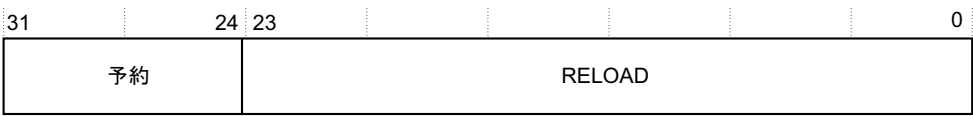


図 8-3 SysTick リロード値レジスタのビット割り当て

SysTick リロード値レジスタのフィールドの説明を、表 8-4 に示します。

表 8-4 SysTick リロード値レジスタのビット割り当て

ビット	フィールド	機能
[31:24]	-	予約
[23:0]	RELOAD	カウンタが 0 になったときに、SysTick 現在値レジスタにロードする値

SysTick 現在値レジスタ

レジスタの現在の値を調べるには、SysTick 現在値レジスタを使用します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

- アドレス 0xE000E018
- アクセス 読み出し / 書き込みクリア
- リセット時の状態 予測不能

SysTick 現在値レジスタのフィールドを、図 8-4 に示します。

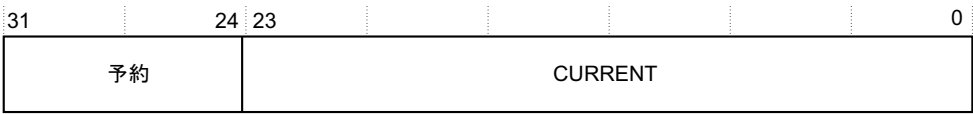


図 8-4 SysTick 現在値レジスタのビット割り当て

SysTick 現在値レジスタのフィールドの説明を、表 8-5 に示します。

**表 8-5 SysTick 現在値レジスタのビット割り当て**

ビット	フィールド	機能
[31:24]	-	予約
[23:0]	CURRENT	レジスタがアクセスされたときの現在値。読み出し - 変更 - 書き込み保護は提供されないで、変更は慎重に行ってください。 このレジスタは書き込みクリアです。このレジスタに任意の値を書き込むと、レジスタが 0 にクリアされます。レジスタをクリアすると、SysTick 制御およびステータスレジスタの COUNTFLAG ビットもクリアされます。

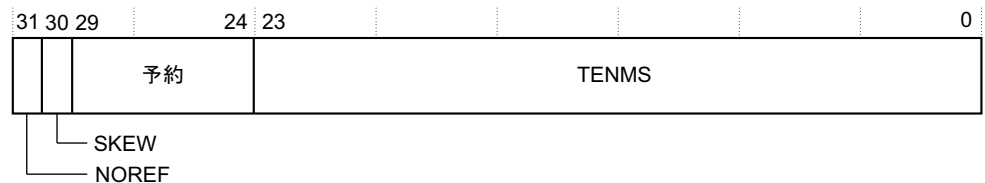
### SysTick 較正值レジスタ

SysTick 較正值レジスタは、ソフトウェアが乗算と除算を使って要求されるスピードにスケールリングすることができるように使用します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス            0xE000E01C  
アクセス            読み出し  
リセット状態        STCALIB

SysTick 較正值レジスタのフィールドを、図 8-5 に示します。



**図 8-5 SysTick 較正值レジスタのビット割り当て**

SysTick 較正值レジスタのフィールドの説明を、表 8-6 に示します。

**表 8-6 SysTick 較正值レジスタのビット割り当て**

ビット	フィールド	機能
[31]	NOREF	1 = 参照クロックが提供されていません。
[30]	SKEW	1 = クロック周波数が原因で、較正值が正確に 10ms ではありません。これは、ソフトウェアのリアルタイムクロックの安定性に影響を与える可能性があります。
[29:24]	-	予約

表 8-6 SysTick 較正值レジスタのビット割り当て

ビット	フィールド	機能
[23:0]	TENMS	この値は、10ms タイミングのために使用するリロード値です。SKEW の値に応じて、正確に 10ms の場合と、10ms に最も近くなる値の場合があります。 この値が 0 として読み出される場合、較正值は不明です。これは通常、参照クロックがシステムからの未知の入力であるか、動的にスケーリングが可能であるかのいずれかが理由です。

割り込みイネーブルセットレジスタ

割り込みイネーブルセットレジスタは、次の目的に使用します。

- 割り込みを許可する。
- どの割り込みが現在許可されているのかを判別する。

レジスタの各ビットは、32 個の割り込みのうちの 1 つと対応します。割り込みイネーブルセットレジスタにビットをセットすることで、対応する割り込みが許可されます。

保留中の割り込みのイネーブルビットがセットされると、プロセッサは優先度に基づいて割り込みをアクティブにします。イネーブルビットがクリアされている場合、割り込み信号をアサートすると、その割り込みは保留されます。しかし、優先度にかかわらず、この割り込みをアクティブにすることはできません。このため、禁止された割り込みは、ラッチされる汎用 I/O ビットとして使用することができ、割り込みを呼び出すことなくこのビットを読み込んでクリアできます。

割り込みイネーブルセットレジスタのビットをクリアするには、割り込みイネーブルクリアレジスタの対応するビットに 1 を書き込みます (P.8-13「割り込みイネーブルクリアレジスタ」参照)。

注

割り込みイネーブルセットレジスタのビットをクリアしても、現在アクティブな割り込みは影響を受けません。新しい割り込みがアクティブになることのみが禁止されます。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス	0xE000E100-0xE000E11C
アクセス	読み出し / 書き込み
リセット時の状態	0x00000000

割り込みイネーブルセットレジスタのフィールドの説明を、表 8-7 に示します。

**表 8-7 割り込みイネーブルセットレジスタのビット割り当て**

ビット	フィールド	機能
[31:0]	SETENA	<p>割り込みイネーブルセット ビット。書き込み操作の場合、次の意味を持ちます。</p> <p>1 = 割り込みを許可</p> <p>0 = 無効</p> <p>読み出し操作の場合、次の意味を持ちます。</p> <p>1 = 割り込みを許可</p> <p>0 = 割り込みを禁止</p> <p>SETENA ビットに対する 0 の書き込みは無効です。このビットを読み出すと、現在の許可状態が返されます。SETENA フィールドはリセット時にクリアされます。</p>

### 割り込みイネーブルクリアレジスタ

割り込みイネーブルクリアレジスタは、次の目的に使用します。

- 割り込みを禁止する。
- どの割り込みが現在禁止されているのかを判別する。

レジスタの各ビットは、32 個の割り込みのうちの 1 つと対応します。割り込みイネーブルクリアレジスタのビットをセットすると、対応する割り込みが禁止されます。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE000E180-0xE000E19C

アクセス 読み出し / 書き込み

リセット時の状態 0x00000000

割り込みイネーブルクリアレジスタのフィールドを、表 8-8 に示します。

**表 8-8 割り込みイネーブルクリアレジスタのビット割り当て**

ビット	フィールド	機能
[31:0]	CLRENA	<p>割り込みイネーブルクリアビット。書き込み操作の場合、次の意味を持ちます。</p> <p>1 = 割り込みを禁止</p> <p>0 = 無効</p> <p>読み込み操作の場合</p> <p>1 = 割り込みを許可</p> <p>0 = 割り込みを禁止</p> <p>CLRENA ビットに対する 0 の書き込みは無効です。このビットを読み出すと、現在の許可状態が返されます。</p>

割り込み保留セットレジスタ

割り込み保留セットレジスタは、次の目的に使用します。

- 割り込みを強制的に保留状態に変更する。
- どの割り込みが現在保留されているのかを判別する。

レジスタの各ビットは、32 個の割り込みのうちの 1 つと対応します。割り込み保留セットレジスタのビットをセットすると、対応する割り込みが保留されます。

割り込み保留セットレジスタのビットをクリアするには、割り込み保留クリアレジスタの対応するビットに 1 を書き込みます（*割り込み保留クリアレジスタ*参照）。割り込み保留セットレジスタのビットをクリアすると、割り込みは非保留状態になります。

注

割り込み保留セットレジスタへの書き込みは、すでに保留または禁止されている割り込みに対しては無効です。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス	0xE000E200-0xE000E21C
アクセス	読み出し / 書き込み
リセット時の状態	0x00000000

割り込み保留セットレジスタのフィールドの説明を、表 8-9 に示します。

表 8-9 割り込み保留セットレジスタのビット割り当て

ビット	フィールド	機能
[31:0]	SETPEND	割り込み保留セットビット 1 = 対応する割り込みを保留 0 = 対応する割り込みは保留していない SETPEND ビットに対する 0 の書き込みは無効です。このビットを読み出すと、現在の状態が返されます。

割り込み保留クリアレジスタ

割り込み保留クリアレジスタは、次の目的に使用します。

- 保留中の割り込みをクリアする。
- どの割り込みが現在保留されているのかを判別する。

レジスタの各ビットは、32 個の割り込みのうちの 1 つと対応します。割り込み保留クリアレジスタのビットをセットすると、対応する保留中の割り込みは非アクティブの状態になります。

注

割り込み保留クリアレジスタへの書き込みは、アクティブな割り込みに対しては、その割り込みが保留されていない限り無効です。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE000E280-0xE000E29C  
 アクセス 読み出し / 書き込み  
 リセット時の状態 0x00000000

割り込み保留クリアレジスタのフィールドの説明を、表 8-10 に示します。

表 8-10 割り込み保留クリアレジスタのビット割り当て

ビット	フィールド	機能
[31:0]	CLRPEND	割り込み保留クリアビット 1 = 保留中の割り込みをクリア 0 = 保留中の割り込みをクリアしない CLRPEND ビットに対する 0 の書き込みは無効です。このビットを読み出すと、現在の状態が返されます。

## アクティブビット レジスタ

どの割り込みがアクティブであるかを判別するには、アクティブビット レジスタを読み出します。レジスタの各フラグは、32 個の割り込みのうちの 1 つと対応します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE000E300-0xE00031C  
 アクセス 読み出し専用  
 リセット時の状態 0x00000000

アクティブビット レジスタのフィールドの説明を、表 8-11 に示します。

表 8-11 アクティブビット レジスタのビット割り当て

ビット	フィールド	機能
[31:0]	ACTIVE	割り込みアクティブフラグ 1 = 割り込みはアクティブであるか、横取りされてスタックされています。 0 = 割り込みは非アクティブであるか、スタックされています。

割り込み優先度レジスタ

割り込み優先度レジスタは、利用可能な各割り込みに、0 ～ 255 の優先度を割り当てるために使用します。0 が最も高い優先度で、255 が最も低い優先度です。

優先度レジスタは、最上位ビット (MSB) から順に格納されます。つまり、4 ビットの優先度がある場合、優先度の値はバイトのビット [7:4] に保存されます。ただし、3 ビットの優先度がある場合は、優先度の値はバイトのビット [7:5] に保存されます。これは、アプリケーションがいくつかの優先度を利用できるのか知る必要なしに動作できることを意味します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス                    0xE000E400-0xE000E41F  
アクセス                    読み出し / 書き込み  
リセット時の状態        0x00000000

割り込み優先度レジスタ 0 ～ 7 のフィールドを、図 8-6 に示します。

	31	24 23	16 15	8 7	0
E000E400	PRI_3	PRI_2	PRI_1	PRI_0	
E000E404	PRI_7	PRI_6	PRI_5	PRI_4	
E000E408	PRI_11	PRI_10	PRI_9	PRI_8	
E000E40C	PRI_15	PRI_14	PRI_13	PRI_12	
E000E410	PRI_19	PRI_18	PRI_17	PRI_16	
E000E414	PRI_23	PRI_22	PRI_21	PRI_20	
E000E418	PRI_27	PRI_26	PRI_25	PRI_24	
E000E41C	PRI_31	PRI_30	PRI_29	PRI_28	

図 8-6 割り込み優先度レジスタの 0 ～ 31 ビットの割り当て

下位の PRI\_n ビットにより、優先度グループ化のサブ優先度を指定することができます。詳細については、P.5-6 「例外の優先度」を参照して下さい



割り込み優先度レジスタのフィールドの説明を、表 8-12 に示します。

表 8-12 割り込み優先度レジスタの 0 ～ 31 ビットの割り当て

ビット	フィールド	機能
[7:0]	PRI <sub><i>n</i></sub>	割り込み <i>n</i> の優先度 <i>n</i>

CPU ID ベースレジスタ

次の内容を判別するには、CPU ID ベースレジスタを読み出します。

- プロセッサコアの ID 番号
- プロセッサコアのバージョン番号
- プロセッサコアの実装の詳細

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス	0xE000ED00
アクセス	読み出し専用
リセット時の状態	0x411FC231

CPUID ベースレジスタのフィールドを、図 8-7 に示します。

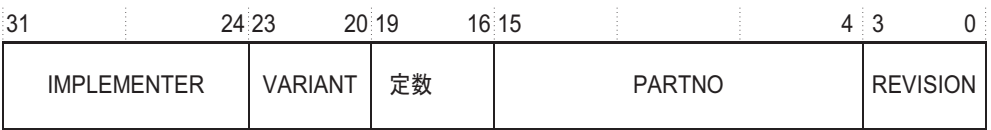


図 8-7 CPUID ベースレジスタのビット割り当て

CPUID ベースレジスタのフィールドの説明を、表 8-13 に示します。

表 8-13 CPUID ベースレジスタのビット割り当て

ビット	フィールド	機能
[31:24]	IMPLEMENTER	実装者コード。ARM の場合は 0x41 です。
[23:20]	VARIANT	実装定義のバリエーション番号

表 8-13 CPUID ベースレジスタのビット割り当て

ビット	フィールド	機能
[19:16]	定数	0xF として読み出されます。
[15:4]	PARTNO	ファミリ内のプロセッサ番号 [11:10] b11 = Cortex ファミリ [9:8] b00 = パージョン [7:6] b00 = 予約 [5:4] b10 = M (v7-M) [3:0] X = ファミリ番号。Cortex-M3 ファミリの場 合は b0011 です。
[3:0]	REVISION	実装定義のリビジョン番号

割り込み制御状態レジスタ

割り込み制御状態レジスタは、次の目的に使用します。

- 保留中のマスク不能割り込み (NMI)(Non-Maskable Interrupt) をセットする。
- 保留中の SVC をセットまたはクリアする。
- 保留中の SysTick をセットまたはクリアする。
- 保留中の例外をチェックする。
- 最も優先度が高い、保留中の例外のベクタ番号をチェックする。
- アクティブな例外のベクタ番号をチェックする。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス                    0xE000ED04  
アクセス                    読み出し / 書き込み、または読み出し専用  
リセット時の状態    0x00000000

割り込み制御状態レジスタのフィールドを、P.8-19 図 8-8 に示します。

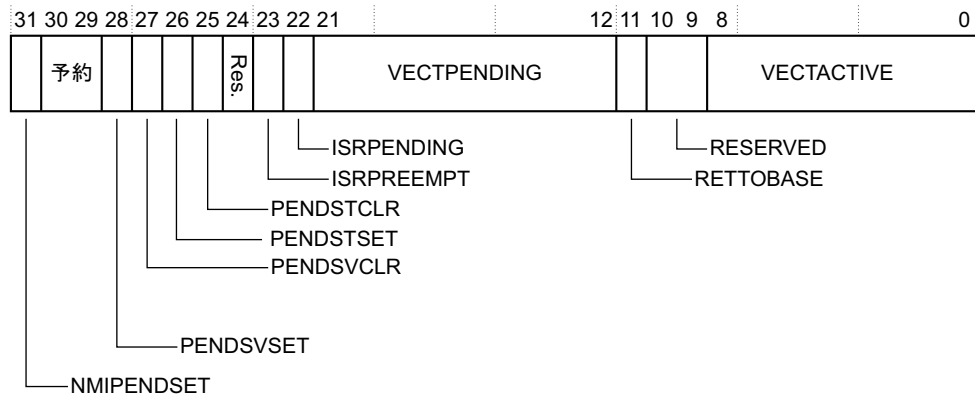


図 8-8 割り込み制御状態レジスタのビット割り当て

割り込み制御状態レジスタのフィールドの説明を、表 8-14 に示します。

表 8-14 割り込み制御状態レジスタのビット割り当て

ビット	フィールド	タイプ	機能
[31]	NMIPENDSET	読み出し / 書き込み	保留中の NMI をセットするビット 1 = 保留中の NMI をセットします。 0 = 保留中の NMI をセットしません。 NMIPENDSET は、NMI を保留し、アクティブにします。NMI は最優先の割り込みなので、設定すると直ちに効果を発揮します。
[30:29]	-	-	予約
[28]	PENDSVSET	読み出し / 書き込み	保留中の PendSV をセットするビット 1 = 保留中の PendSV をセットします。 0 = 保留中の PendSV をセットしません。
[27]	PENDSVCLR	書き込み専用	保留中の PendSV をクリアするビット 1 = 保留中の PendSV をクリアします。 0 = 保留中の PendSV をクリアしません。
[26]	PENDSTSET	読み出し / 書き込み	保留中の SysTick をセットするビット 1 = 保留中の SysTick をセットします。 0 = 保留中の SysTick をセットしません。
[25]	PENDSTCLR	書き込み専用	保留中の SysTick をクリアするビット 1 = 保留中の SysTick をクリアします。 0 = 保留中の SysTick をクリアしません。

表 8-14 割り込み制御状態レジスタのビット割り当て

ビット	フィールド	タイプ	機能
[24]	-	-	予約
[23]	ISRPREEMPT	読み出し専用	このビットは、デバッグ時のみ使用できます。保留中の割り込みが、次の実行サイクルにアクティブになることを示します。デバッグホールド制御およびステータスレジスタ (Debug Halting Control and Status Register) の C_MASKINTS がクリアされている場合、割り込みは処理されます。
[22]	ISRPENDING	読み出し専用	割り込み保留フラグ。NMI とフォールトは除外します。 1 = 割り込みは保留中。 0 = 割り込みは保留中ではない。
[21:12]	VECTPENDING	読み出し専用	保留中の ISR 番号フィールド。VECTPENDING には、最も優先度が高い保留中の ISR の割り込み番号が格納されています。
[11]	RETTOBASE	読み出し専用	すべてのアクティブな例外のセットから IPSR_current_exception を引いた結果が、空のセットである場合、このビットは 1 になります。
[10]	-	-	予約
[9]	-	-	予約
[8:0]	VECTACTIVE	読み出し専用	アクティブな ISR 番号フィールド。VECTACTIVE には、NMI およびハードフォールトを含む、現在実行されている ISR の割り込み番号が格納されています。共有ハンドラは、VECTACTIVE を使用して、その ISR がどの割り込みにより呼び出されたのかを特定できます。VECTACTIVE フィールドから 16 を減算して、割り込みイネーブルクリア/セットレジスタ、割り込み保留クリア/セットレジスタ、割り込み優先度レジスタへのインデックスとすることができます。INTISR[0] のベクタ番号は 16 です。 VECTACTIVE フィールドはリセット時にクリアされます。

ベクタテーブル オフセットレジスタ

ベクタテーブル オフセットレジスタは、次の内容を決定するために使用します。

- ベクタテーブルが RAM またはコードメモリにあるかどうか。
- ベクタテーブルのオフセット。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス                    0xE000ED08  
アクセス                    読み出し / 書き込み  
リセット時の状態        0x00000000

ベクタテーブル オフセットレジスタのフィールドを、P.8-21 図 8-9 に示します。

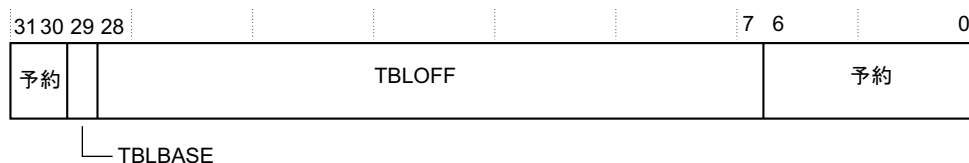


図 8-9 ベクタテーブル オフセットレジスタのビット割り当て

ベクタテーブル オフセットレジスタのフィールドの説明を、表 8-15 に示します。

表 8-15 ベクタテーブル オフセットレジスタのビット割り当て

ビット	フィールド	機能
[31:30]	-	予約
[29]	TBLBASE	テーブルベースはコード (0) または RAM(1) のいずれかに存在します。
[28:7]	TBLOFF	ベクタテーブルベース オフセットフィールド。SRAM または CODE 空間の最下位からテーブルのベースまでのオフセットが含まれています。
[6:0]	-	予約

ベクタテーブル オフセットレジスタは、CODE または SRAM 空間にあるベクタテーブルの位置を示しています。リセット時のデフォルト値は 0 (CODE 空間) です。位置を設定するとき、オフセットはテーブルにある例外の数に基づいてアラインされる必要があります。つまり、16 個までの割り込みが使える、最小のアライメントは 32 ワードになります。割り込みの数がより多い場合は、次の 2 のべき乗まで切り上げて、アライメントを調整する必要があります。例えば、21 個の割り込みが必要な場合、テーブルサイズが 37 ワードになり、次の 2 のべき乗は 64 なので、アライメントは 64 ワード境界の必要があります。

## アプリケーション割り込みおよびリセット制御レジスタ

アプリケーション割り込みおよびリセット制御レジスタは、次の目的に使用します。

- データのエンティアン形式を判別する。
- デバッグまたはハードウェア障害からの回復のため、すべてのアクティブな状態情報をクリアする。
- システムリセットを実行する。
- 優先度のグループ分け位置（2 進小数点）を変更する。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE000ED0C

アクセス      読み出し / 書き込み

リセット時の状態 0x00000000

アプリケーション割り込みおよびリセット制御レジスタのフィールドを、図 8-10 に示します。

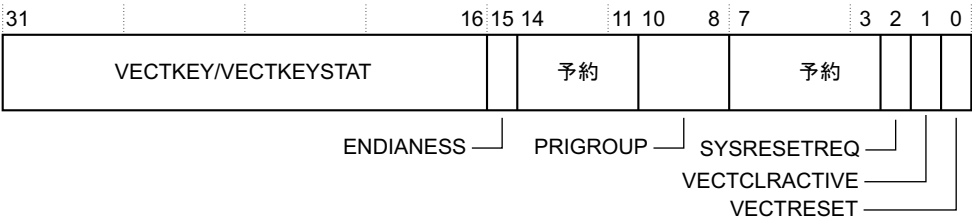


図 8-10 アプリケーション割り込みおよびリセット制御レジスタのビット割り当て

アプリケーション割り込みおよびリセット制御レジスタのフィールドの説明を、表 8-16 に示します。

表 8-16 アプリケーション割り込みおよびリセット制御レジスタのビット割り当て

ビット	フィールド	機能
[31:16]	VECTKEY	レジスタキー。このレジスタへ書き込みを行うには、VECTKEY フィールドに 0x5FA を指定する必要があります。それ以外の場合、書き込み値は無視されます。
[31:16]	VECTKEYSTAT	0xFA05 として読み出されます。
[15]	ENDIANESS	データエンディアン形式ビット 1 = ビッグエンディアン 0 = リトルエンディアン  ENDIANESS は、リセット時に BIGEND 入力ポートからサンプリングされます。リセット期間外では ENDIANESS を変更できません。
[14:11]	-	予約
[10:8]	PRIGROUP	割り込み優先度グループ分けフィールドで、次の意味を持ちます。 PRIGROUP    サブ優先度と横取り優先度の分割 0            7.1 で、横取り優先度が 7 ビット、サブ優先度が 1 ビットを示します。 1            6.2 で、横取り優先度が 6 ビット、サブ優先度が 2 ビットを示します。 2            5.3 で、横取り優先度が 5 ビット、サブ優先度が 3 ビットを示します。 3            4.4 で、横取り優先度が 4 ビット、サブ優先度が 4 ビットを示します。 4            3.5 で、横取り優先度が 3 ビット、サブ優先度が 5 ビットを示します。 5            2.6 で、横取り優先度が 2 ビット、サブ優先度が 6 ビットを示します。 6            1.7 で、横取り優先度が 1 ビット、サブ優先度が 7 ビットを示します。 7            0.8 で、横取り優先度なし、サブ優先度が 8 ビットを示します。

表 8-16 アプリケーション割り込みおよびリセット制御レジスタのビット割り当て

ビット	フィールド	機能
		PRIGROUP フィールドは、同じ横取りレベルを持つ例外のサブ優先度を作成するために、2 進小数点の位置を示します。割り込み優先度レジスタの PRI_n フィールドを、横取りレベルとサブ優先度レベルに分割します。2 進小数点の位置は、値の左です。つまり、PRIGROUP の値は、最下位ビット (LSB) の左から始まる小数点を表しています。これは 7:0 のビット [0] になります。 優先度に割り当てられているビットの数および選択されている実装によっては、最下位の値は 0 ではないこともあります。
[7:3]	-	予約
[2]	SYSRESETREQ	リセットが必要なことを示している外部システムに対して、信号のアサートを引き起こします。デバッグ以外の主要なコンポーネントすべてについて、強制的に大規模なシステムリセットを行うことを目的としています。このビットをセットしても、ホールトデバッグが実行されるのを防ぐことはできません。
[1]	VECTCLRACTIVE	次のように、アクティブなベクタビットをクリアします。 1 = アクティブな NMI、フォールト、および割り込みの、すべての状態情報をクリアします。 0 = クリアしません。 スタックの再初期化はアプリケーションで行う必要があります。 VECTCLRACTIVE ビットは、デバッグ中に既知の状態に戻るために使用されます。VECTCLRACTIVE ビットは、自身によりクリアされます。 IPSR は、この操作ではクリアされません。このため、アプリケーションで使用する場合は、ベースレベルの起動またはアクティブビットを設定可能なシステムハンドラの内部でのみ使用する必要があります。
[0]	VECTRESET	システムリセット ビットで、デバッグコンポーネント以外のシステムをリセットします。 1 = システムをリセットします。 0 = システムをリセットしません。 VECTRESET ビットは、自身によりクリアされます。VECTRESET ビットはリセット時にクリアされます。 デバッグの場合は、コアが停止したときのみこのビットを書き込んで下さい。

注

**SYSRESETREQ** は、システムリセットによりクリアされます。これは、**VECTRESET** を同時にアサートすると、**SYSRESETREQ** が書き込まれたのと同じサイクルにクリアされる可能性があることを意味します。この場合、外部のシステムが **SYSRESETREQ** を観測できなくなる可能性があります。したがって、**VECTRESET** と **SYSRESETREQ** は排他的に使用し、同時に両方に 1 を書き込まないようにする必要があります。

システム制御レジスタ

システム制御レジスタは、次のような電力管理機能のため使用します。

- プロセッサが低電力状態にいつ移行できるかをシステムへ知らせる
- プロセッサが低電力状態に入る、または退出する方法の制御

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス                    0xE000ED10  
アクセス                    読み出し / 書き込み  
リセット時の状態    0x00000000

システム制御レジスタのフィールドを、図 8-11 に示します。

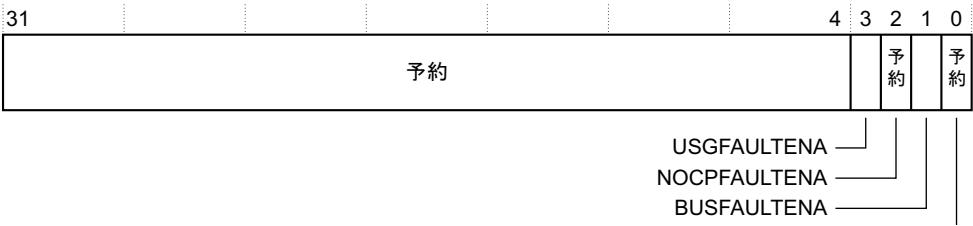


図 8-11 システム制御レジスタのビット割り当て

システム制御レジスタのフィールドの説明を、表 8-17 に示します。

表 8-17 システム制御レジスタのビット割り当て

ビット	フィールド	機能
[31:5]	-	予約
[4]	SEVONPEND	このビットが許可されると、割り込みが非アクティブから保留中に移行したときに、WFE がウェークアップされます。許可されていない場合は、WFE は外部の SEV 命令で生成されたイベント信号のみによりウェークアップされます。このイベント入力 (RXEV) は、イベント待ちでないときであっても受け付けられ、次の WFE に対して効果を発揮します。



表 8-17 システム制御レジスタのビット割り当て

ビット	フィールド	機能
[2]	SLEEPDEEP	<p>ディープスリープビット</p> <p>1 = Cortex-M3 クロックの停止が可能なことをシステムに通知します。このビットをセットすると、プロセッサを停止可能なときに <b>SLEEPDEEP</b> ポートがアサートされます。</p> <p>0 = システムクロックを停止することはできません。</p> <p><b>SLEEPDEEP</b> の使用法の詳細については、第 7 章「電力管理」を参照して下さい。</p>
[1]	SLEEPONEXIT	<p>ハンドラモードからスレッドモードに戻るとき、退出時スリープに移行します。</p> <p>1 = ISR の退出時スリープに移行します。</p> <p>0 = スレッドモードへ戻るときにスリープへ移行しません。</p> <p>割り込み駆動型アプリケーションは、この機能を使用して、空のメインアプリケーションに戻ることを避けられます。</p>
[0]	-	予約

## 構成制御レジスタ

構成制御レジスタは、次の目的に使用します。

- NMI、ハードフォールト、**FAULTMASK** がバスフォールトを無視するのを許可する。
- 0 による除算およびアンアラインドアクセスをトラップする。
- ソフトウェアトリガ例外レジスタへのユーザアクセスを許可する。
- スレッドモードへのエントリを制御する。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE000ED14

アクセス 読み出し / 書き込み

リセット時の状態 0x00000000

構成制御レジスタのフィールドを、P.8-26 図 8-12 に示します。

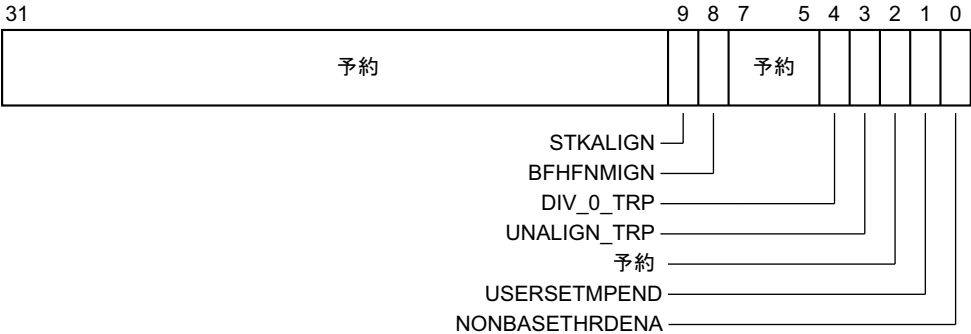


図 8-12 構成制御レジスタのビット割り当て

構成制御レジスタのフィールドの説明を、表 8-18 に示します。

表 8-18 構成制御レジスタのビット割り当て

ビット	フィールド	機能
[9]	STKALIGN	1 = 例外エントリのときに、例外の前に使用されていた SP が 8 バイトアラインドに調整され、復元のためのコンテキストが保存されます。SP は対応する例外からの復帰時に復元されます。 0 = 例外の開始時に、例外の前に使用されていた SP には、4 バイトのアライメントのみが保証されます。
[8]	BFHFNMIGN	許可された場合、優先度 -1 および -2 で実行されているハンドラ（ハードフォールト、NMI、および FAULTMASK から昇格されたハンドラ）が、ロードおよびストア命令により生じたデータバス フォールトを無視します。禁止されている場合、これらのバスフォールトはロックアップを引き起こします。このビットを有効にする場合は、十分な注意を払うようにして下さい。すべてのデータバス フォールトが無視されるため、ハンドラおよびそのデータが絶対的に安全なメモリにある場合のみ、使用して下さい。通常、制御パスの問題の検出や修正のためにシステムデバイスやブリッジを調査するのに使用されます。
[4]	DIV_0_TRP	0 による除算でのトラップ。これにより、0 による除算が試みられた場合のフォールト / 停止が許可されます。関連する用法フォールトステータス レジスタのビットは DIVBYZERO です。詳細については、P.8-34 「用法フォールトステータス レジスタ」を参照して下さい。

表 8-18 構成制御レジスタのビット割り当て

ビット	フィールド	機能
[3]	UNALIGN_TRP	アンアラインドアクセス用のトラップ。これにより、アンアラインドなハーフまたはフルワードアクセスに対するフォールト/停止が許可されます。アンアラインドな複数ロード/ストアは常にフォールトを引き起こします。関連する用法フォールトステータスレジスタのビットは UNALIGNED です。詳細については、P.8-34「用法フォールトステータスレジスタ」を参照して下さい。
[1]	USERSETMPEND	1 が書き込まれた場合、メインスタックポインタと対応しているメイン例外をトリガ（保留）するために、ユーザコードがソフトウェアトリガ割り込みレジスタに書き込みを行うことを許可します。
[0]	NONEBASETHRDENA	0（デフォルト値）の場合、最後の例外から戻ったときはスレッドモードのみに移行することができます。1 に設定されている場合、戻り値の制御によってハンドラモードのどのレベルからでもスレッドモードに移行することが可能です。

### システムハンドラ優先度レジスタ (System Handler Priority Registers)

システムハンドラ優先度レジスタは 3 つあり、次のシステムハンドラの優先度付けに使用します。

- メモリ管理
- バスフォールト
- 用法フォールト
- デバッグモニタ
- SVC
- SysTick
- PendSV

システムハンドラは、どの優先度レベルにも設定可能な例外ハンドラの特長なクラスです。ほとんどのシステムハンドラは、オン（許可）とオフ（禁止）のいずれにもマスクできます。禁止された場合、そのフォールトは常にハードフォールトとして取り扱われます。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE000ED18, 0xE000ED1C, 0xE000ED20

アクセス 読み出し / 書き込み

リセット時の状態 0x00000000

システムハンドラ優先度レジスタのフィールドを、P.8-28 図 8-13 に示します。

	31	24 23	16 15	8 7	0
E000ED18	PRI_7	PRI_6	PRI_5	PRI_4	
E000ED1C	PRI_11	PRI_10	PRI_9	PRI_8	
E000ED20	PRI_15	PRI_14	PRI_13	PRI_12	

図 8-13 システムハンドラ優先度レジスタのビット割り当て

システムハンドラ優先度レジスタのフィールドの説明を、表 8-19 に示します。

表 8-19 システムハンドラ優先度レジスタのビット割り当て

ビット	フィールド	機能
[31:24]	PRI_N3	システムハンドラ 7、11、15 の優先度。それぞれ、予約、SVCall、SysTick に対応します。
[23:16]	PRI_N2	システムハンドラ 6、10、14 の優先度。それぞれ、用法フォールト、予約、PendSV に対応します。
[15:8]	PRI_N1	システムハンドラ 5、9、13 の優先度。それぞれ、バスフォールト、予約、予約に対応します。
[7:0]	PRI_N	システムハンドラ 4、8、12 の優先度。それぞれ、メモリ管理、予約、デバッグモニタに対応します。

システムハンドラ制御および状態レジスタ

システムハンドラ制御状態および状態レジスタは、次の目的に使用します。

- システムハンドラを許可または禁止する。
- バスフォールト、メモリ管理フォールト、SVC の保留ステータスを判定する。
- システムハンドラのアクティブステータスを判定する。

フォールトハンドラが禁止されているときに対応するフォールト状態が発生した場合、そのフォールトはハードフォールトに昇格されます。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス	0xE000ED24
アクセス	読み出し / 書き込み
リセット時の状態	0x00000000

システムハンドラ制御および状態レジスタのフィールドを、P.8-29 図 8-14 に示します。

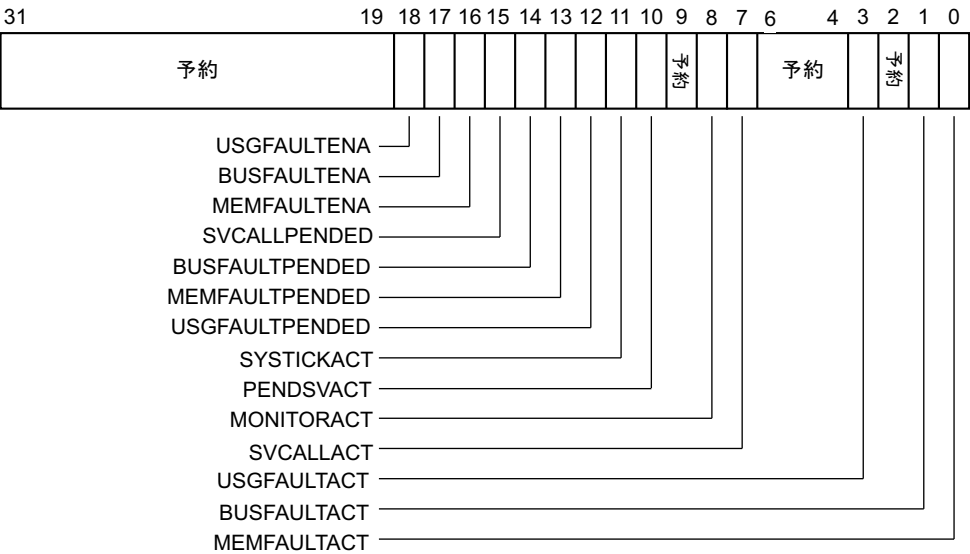


図 8-14 システムハンドラ制御および状態レジスタのビット割り当て

システムハンドラ制御および状態レジスタのフィールドの説明を、表 8-20に示します。

表 8-20 システムハンドラ制御および状態レジスタのビット割り当て

ビット	フィールド	機能
[31:19]	-	予約
[18]	USGFAULTENA	禁止する場合は 0 に、許可する場合は 1 にセットします。
[17]	BUSFAULTENA	禁止する場合は 0 に、許可する場合は 1 にセットします
[16]	MEMFAULTENA	禁止する場合は 0 に、許可する場合は 1 にセットします
[15]	SVCALLPENDEDED	SVCall が保留されている場合、1 として読み出されます。
[14]	BUSFAULTPENDEDED	BusFault が保留されている場合、1 として読み出されます。
[13]	MEMFAULTPENDEDED	MemManage が保留されている場合、1 として読み出されます。
[12]	USGFAULTPENDEDED	用法フォールトが保留されている場合、1 として読み出されます。
[11]	SYSTICKACT	SysTick がアクティブな場合、1 として読み出されます。
[10]	PENDSVACT	PendSV がアクティブな場合、1 として読み出されます。
[9]	-	予約

表 8-20 システムハンドラ制御および状態レジスタのビット割り当て

ビット	フィールド	機能
[8]	MONITORACT	モニタがアクティブな場合、1として読み出されます。
[7]	SVCALLACT	SVCall がアクティブな場合、1として読み出されます。
[6:4]	-	予約
[3]	USGFAULTACT	UsageFault がアクティブな場合、1として読み出されます。
[2]	-	予約
[1]	BUSFAULTACT	BusFault がアクティブな場合、1として読み出されます。
[0]	MEMFAULTACT	MemManage がアクティブな場合、1として読み出されます。

アクティブビットは、アクティブか、現在実行中か、または横取りによりスタックされているシステムハンドラがあるかどうかを示します。この情報はデバッグに使用され、アプリケーションハンドラによっても使用されます。保留ビットは、リトライできないフォールトがより高い優先度の割り込みの後着が原因で引き伸ばされている場合のみセットされます。

————警告————

アクティブビットは書き込み、クリア、またはセットが可能です。しかし、実行の際は十分に注意して下さい。これらのビットのクリアおよびセットは、スタックの内容の修復や、他のデータ構造のクリーンアップを行いません。コンテキストスイッチャが、スレッドのコンテキストを保存するためにクリアとセットを使用することを意図しています。フォールトハンドラ中での使用も考えられます。最も一般的な状況は、未定義の命令およびコプロセッサのエミュレーションのために、SVCall ハンドラまたは UsageFault ハンドラにあるスレッドのコンテキストを保存することです。この操作を実行するためのモデルは、現在の状態を保存して、ハンドラのコンテキストを格納しているスタックを切り替えて、新しいスレッドの状態をロードして、新しいスレッドのスタックに切り替えてから、スレッドに戻ることです。IPSRはこの操作を反映するように変更されないので、現在のハンドラのアクティブビットを絶対クリアしないようにしてください。スタックされたアクティブなハンドラを変更するためだけに使用して下さい。

表で示されているように、SVCALLPENDED と BUSFAULTPENDED ビットは、対応するハンドラが後着割り込みによって保留されている場合にセットされます。これらのビットは、元になるハンドラが実際に起動されるまで、クリアされません。つまり、もし SVCall またはバスフォールト ハンドラが開始される前に、スタックエラーまたはベクタ読み込みエラーが発生した場合、これらのビットはクリアされません。これにより、プッシュエラーまたはベクタ読み出しエラーハンドラでは、ビットをクリアするか再試行するかを選択できます。

構成可能フォールトステータス レジスタ

構成可能フォールトステータス レジスタは次の 3 つがあり、ローカルフォールトに関する情報を取得するために使用します。

- メモリ管理フォールトステータス レジスタ
- バスフォールトステータス レジスタ : P.8-33
- P.8-34 「用法フォールトステータス レジスタ」

これらのレジスタのフラグは、ローカルフォールトの原因を示します。複数のフォールトが発生した場合は、複数のフラグが設定されることがあります。これらのレジスタは読み出し / 書き込みクリアです。つまり、これらのレジスタは通常に読み出せますが、任意のビットに 1 を書き込むと、そのビットがクリアされます。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス	0xE000ED28	メモリ管理フォールトステータス レジスタ
	0xE000ED29	バスフォールトステータス レジスタ
	0xE000ED2A	用法フォールトステータス レジスタ
アクセス	読み出し / 1 を書き込んでクリア	
リセット時の状態	0x00000000	

構成可能フォールトステータス レジスタのフィールドを、図 8-15 に示します。

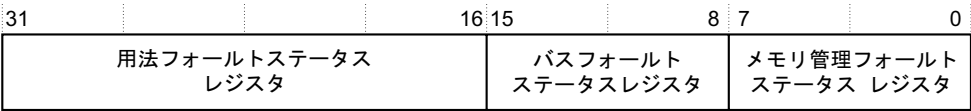


図 8-15 構成可能フォールトステータス レジスタのビット割り当て

メモリ管理フォールトステータス レジスタ

メモリ管理フォールトステータス レジスタのフラグは、メモリアクセス フォールトの原因を示します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス	0xE000ED28
アクセス	読み出し / 1 を書き込んでクリア
リセット時の状態	0x00000000

メモリ管理フォールトステータス レジスタのフィールドを、P.8-32 図 8-16 に示します。

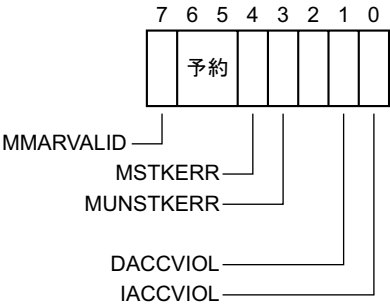


図 8-16 メモリ管理フォールトレジスタのビット割り当て

メモリ管理フォールトステータス レジスタのフィールドの説明を、表 8-21に示します。

表 8-21 メモリ管理フォールトステータス レジスタのビット割り当て

ビット	フィールド	機能
[7]	MMARVALID	メモリ管理アドレスレジスタ(MMAR) アドレス有効フラグ  1 = MMAR での有効なフォールトアドレス。バスフォールトなどの後着フォールトは、メモリ管理フォールトをクリアすることができます。 0 = MMAR での有効なフォールトアドレスなし。  優先度が原因でハードフォールトに昇格される MemManage フォールトが発生した場合、ハードフォールトハンドラは、このビットをクリアする必要があります。これによって、MMAR 値が上書きされてしまっている、スタックされたアクティブな MemManage ハンドラに戻るという問題を防止できます。
[4]	MSTKERR	例外のスタッキングで、1 つ以上のアクセス違反が発生しました。SP はまだ調整されていて、スタックのコンテキスト領域の値は正しくない可能性があります。MMAR は書き込まれていません。
[3]	MUNSTKERR	例外からの復帰のアンスタッキングで、1 つ以上のアクセス違反が発生しました。これは、ハンドラにチェーンされているので、元の復帰スタックはまだ存在しています。SP は失敗した復帰により調整されておらず、新しい保存は実行されていません。MMAR は書き込まれていません。
[1]	DACCVIOL	データアクセス違反フラグ。許可されていない場所に対してロード/ストア操作が試みられると、DACCVIOL フラグがセットされます。復帰 PC は、フォールトの発生した命令を示しています。このエラーでは、試みられたアクセスのアドレスが MMAR にロードされます。
[0]	IACCVIOL	命令アクセス違反フラグ。実行が許可されていない場所から命令のフェッチが試みられると、IACCVIOL フラグがセットされます。このエラーは、XN 領域へのすべてのアクセスで発生し、MPU が禁止されている場合や存在しない場合でも発生します。復帰 PC は、フォールトの発生した命令を示しています。MMAR は書き込まれていません。



バスフォールトステータス レジスタ

バスフォールトステータス レジスタのフラグは、バスアクセス フォールトの原因を示します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

- アドレス 0xE000ED29
- アクセス 読み出し /1 を書き込んでクリア
- リセット時の状態 0x00000000

バスフォールトステータス レジスタのフィールドを、図 8-17 に示します。

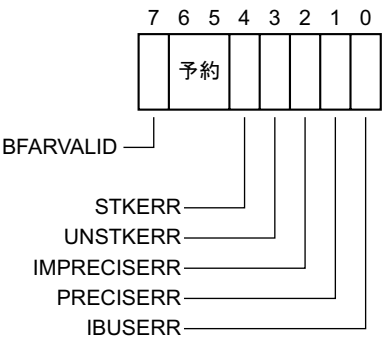


図 8-17 バスフォールトステータス レジスタのビット割り当て

バスフォールトステータス レジスタのフィールドの説明を、表 8-22 に示します。

表 8-22 バスフォールトステータス レジスタのビット割り当て

ビット	フィールド	機能
[7]	BFARVALID	このビットは、バスフォールトアドレス レジスタ (BFAR) に有効なアドレスが含まれている場合にセットされます。これは、そのアドレスがわかっているバスフォールトの後に真になります。メモリ管理フォールトなど他のフォールトが後で発生すると、このビットがクリアされる可能性があります。 優先度が原因でハードフォールトに昇格されるバスフォールトが発生した場合、ハードフォールト ハンドラは、このビットをクリアする必要があります。これによって、BFAR 値が上書きされているスタックされたアクティブなバスフォールト ハンドラに復帰する場合の問題が回避されます。
[6:5]	-	予約
[4]	STKERR	例外でのスタッキングで、1 つ以上のバスフォールトが発生しました。SP は調整された状態のままで、スタックのコンテキスト領域の値は正しくない可能性があります。BFAR は書き込まれていません。

表 8-22 バスフォールトステータス レジスタのビット割り当て

ビット	フィールド	機能
[3]	UNSTKERR	例外復帰でのアンスタッキングで、1 つ以上のバスフォールトが発生しました。これは、ハンドラにチェインされているので、元の復帰スタックはまだ存在しています。SP は失敗した復帰によって調整されず、新しい保存は実行されていません。BFAR は書き込まれていません。
[2]	IMPRECISERR	不正確なデータバス エラー。これは BusFault ですが、復帰 PC は原因である命令と関係がありません。これは、同期フォールトではありません。そのため、現在のアクティベーションの優先度がバスフォールトより高いことが検出された場合、保留のみが行われます。より低い優先度のアクティベーションに戻った際に、バスフォールトはアクティブになります。優先度の低い例外に戻る前に正確なフォールトが発生した場合、IMPRECISERR と正確なフォールトステータスビットの 1 つが同時にセットされていることが、ハンドラによって検出されます。BFAR は書き込まれていません。
[1]	PRECISERR	正確なデータバス エラーの復帰
[0]	IBUSERR	命令バスエラー フラグ 1 = 命令バスエラー 0 = 命令バスエラーなし  IBUSERR フラグは、プリフェッチエラーによりセットされます。フォールトはその命令で停止するため、分岐シャドウでエラーが発生した場合は、フォールトは発生しません。BFAR は書き込まれていません。

用法フォールトステータス レジスタ

用法フォールトステータス レジスタのフラグは、次のエラーを示します。

- EPSR と命令の不正な組み合わせ
- 不正な PC ロード
- 不正なプロセッサ状態
- 命令デコードエラー
- コプロセッサ命令使用の試み
- 不正なアンアラインドアクセス

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス                    0xE000ED2B  
アクセス                    読み出し / 書き込みクリア  
リセット状態                0x00000000

用法フォールトステータス レジスタのフィールドを、P.8-35 図 8-18 に示します。

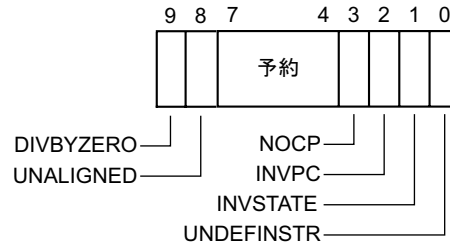


図 8-18 用法フォールトステータス レジスタのビット割り当て

用法フォールトステータス レジスタのフィールドを、表 8-23 に示します。

表 8-23 用法フォールトステータス レジスタのビット割り当て

ビット	フィールド	機能
[9]	DIVBYZERO	DIV_0_TRP (P.8-25 「構成制御レジスタ」 参照) が許可されていて、SDIV または UDIV 命令が除数 0 で使用された場合、このフォールトが発生します。命令は実行され、復帰 PC はこの命令を指し示します。DIV_0_TRP が設定されていない場合は、除算により 0 の商が戻されます。
[8]	UNALIGNED	UNALIGN_TRP (P.8-25 「構成制御レジスタ」 参照) が許可されていて、アンアラインドなメモリアクセスが試みられた場合、このフォールトが発生します。アンアラインドな LDM/STM/LDRD/STRD 命令は、UNALIGN_TRP の設定にかかわらず、常にフォールトになります。
[7:4]	-	予約
[3]	NOCP	コプロセッサ命令使用の試み。プロセッサはコプロセッサ命令をサポートしていません。
[2]	INVPC	EXC_RETURN を PC に不正にロードする試み。無効な命令、無効なコンテキスト、無効な値。復帰 PC は、PC の設定を試みた命令を指し示しています。
[1]	INVSTATE	UNDEFINED 命令以外の理由による、EPSR と命令の無効な組み合わせ。復帰 PC は、フォールトを引き起こした無効な状態の命令を指し示します。
[0]	UNDEFINSTR	UNDEFINSTR フラグは、プロセッサが未定義の命令を実行しようとした場合に設定されます。これは、プロセッサがデコードできない命令です。復帰 PC は、未定義の命令を指し示します。

### 注

このレジスタがクリアされる前に複数のフォールトが発生した場合、このフォールトビットは加算的に動作します。

ハードフォールトステータス レジスタ

ハードフォールトステータス レジスタ (HFSR) は、ハードフォールト ハンドラをアクティブにするイベントについての情報を取得するために使用します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

- アドレス 0xE000ED2C
- アクセス 読み出し /1 を書き込んでクリア
- リセット時の状態 0x00000000

HFSR は書き込みクリアレジスタです。これは、ビットに 1 を書き込むと、そのビットがクリアされることを意味します。ハードフォールトステータス レジスタのフィールドを、図 8-19 に示します。

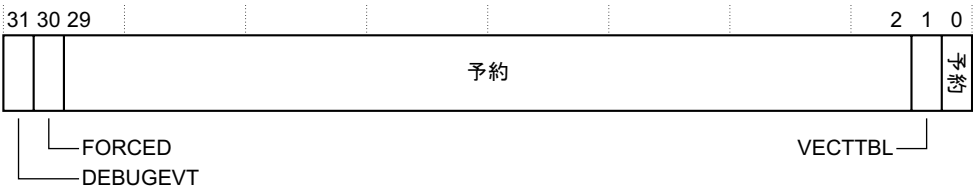


図 8-19 ハードフォールトステータス レジスタのビット割り当て

ハードフォールトステータス レジスタのフィールドの説明を、表 8-24 に示します。

表 8-24 ハードフォールトステータス レジスタのビット割り当て

ビット	フィールド	機能
[31]	DEBUGEVT	このビットは、デバッグに関連するフォールトがあった場合にセットされます。これは、ホールトデバッグが許可されていない場合のみ起こり得ます。モニタが許可されたデバッグの場合、現在の優先度がモニタより高いときに、BKPT に対してのみ起こります。ホールトおよびモニタデバッグの両方が禁止されている場合、無視されないデバッグイベント（最低でも BKPT）に対してのみ起こります。デバッグフォールトステータス レジスタは更新されます。
[30]	FORCED	構成可能フォールトが受信されましたが、優先度または構成可能フォールトが禁止されていることが原因でアクティブにできないため、ハードフォールトがアクティブになりました。 この場合、ハードフォールト ハンドラは、他のフォールトステータス レジスタを読み出して、原因を特定する必要があります。
[29:2]	-	予約
[1]	VECTTBL	このビットは、ベクタテーブルが例外処理（バスフォールト）で読み出されたことが原因でフォールトが発生した場合にセットされます。この場合、常にハードフォールトになります。復帰 PC は、横取りされた命令を指し示します。
[0]	-	予約

## デバッグフォールトステータス レジスタ

デバッグフォールトステータス レジスタは、次の内容をモニタするのに使用します。

- 外部デバッグ要求
- ベクタキャッチ
- データウォッチポイント一致
- BKPT 命令の実行
- ホールト要求

複数のフォールト条件が発生した場合、デバッグフォールトステータス レジスタの複数のフラグがセットされることがあります。このレジスタは読み出し / 書き込みクリアです。これは、通常どおりの読み込みが可能であることを示します。1 をビットに書き込むと、そのビットがクリアされます。

### 注

これらのビットは、イベントがキャッチされない限りセットされません。つまり、これはある種の停止の原因になります。ホールトデバッグが許可されている場合、これらのイベントはプロセッサを停止して、デバッグ状態に移行させます。デバッグが禁止され、デバッグモニタが許可されている場合、優先度によって許されれば、これはデバッグモニタ ハンドラ呼び出しになります。デバッグとモニタが両方とも禁止されている場合は、これらのイベントのうち一部はハードフォールトで、ハードフォールトステータス レジスタの DBGEVT ビットがセットされます。一部のイベントは無視されます。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE000ED30

アクセス 読み出し / 1 を書き込んでクリア

リセット時の状態 0x00000000

デバッグフォールトステータス レジスタのフィールドを、図 8-20 に示します。

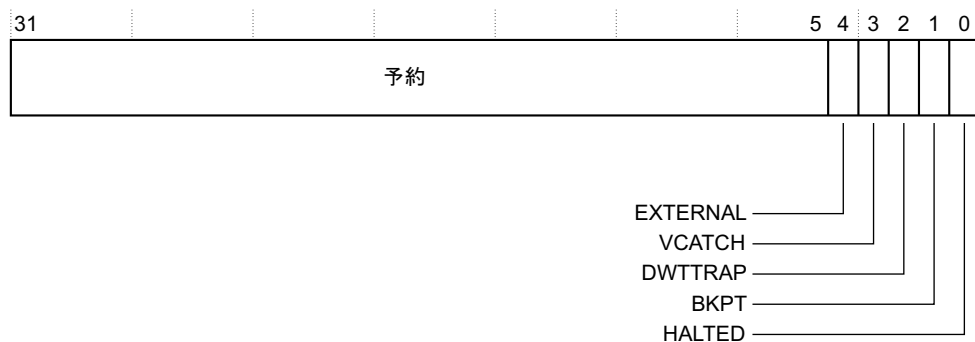


図 8-20 デバッグフォールトステータス レジスタのビット割り当て

デバッグフォールトステータス レジスタのフィールドの説明を、表 8-25 に示します。

表 8-25 デバッグフォールトステータス レジスタのビット割り当て

ビット	フィールド	機能
[31:5]	-	予約
[4]	EXTERNAL	外部デバッグ要求フラグ 1 = <b>EDBGRQ</b> シグナルがアサートされます。 0 = <b>EDBGRQ</b> シグナルはアサートされません。 プロセッサは次の命令の境界で停止します。
[3]	VCATCH	ベクタキャッチ フラグ 1 = ベクタキャッチが発生しました。 0 = ベクタキャッチは発生していません。 VCATCH フラグがセットされている場合、フォールトのタイプを示すため、ローカルフォールトステータス レジスタの 1 つのフラグもセットされます。
[2]	DWTTRAP	データウォッチポイントおよびトレース (DWT) フラグ 1 = DWT 一致 0 = DWT 不一致 プロセッサは、現在の命令または次の命令で停止します。
[1]	BKPT	BKPT フラグ 1 = BKPT 命令を実行します。 0 = BKPT 命令を実行しません。 BKPT フラグは、フラッシュパッチ コードおよび通常コード中の BKPT 命令でセットされます。復帰 PC は、命令を含むブレークポイントを指し示します。
[0]	HALTED	ホールト要求フラグ 1 = ステップを含む、NVIC により要求されたホールト。プロセッサは次の命令でホールトします。 0 = ホールト要求なし

メモリ管理フォールトアドレス レジスタ

メモリ管理フォールトアドレス レジスタは、メモリ管理フォールトの原因となった位置のアドレスを読み出すために使用します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス	0xE000ED34
アクセス	読み出し / 書き込み
リセット時の状態	予測不能

メモリ管理フォールトアドレス レジスタのフィールドの説明を、表 8-26 に示します。

表 8-26 メモリ管理フォールトアドレス レジスタのビット割り当て

ビット	フィールド	機能
[31:0]	ADDRESS	メモリ管理フォールトアドレス フィールド。ADDRESS は、フォールトが発生したロード/ストアを試みたデータアドレスです。アンアラインドなアクセスでフォールトが発生した場合、このアドレスは実際にフォールトが発生したアドレスです。アクセスはアラインされた複数の部分に分割されている可能性があるため、このアドレスは要求されたサイズの範囲内の任意のオフセットの可能性があります。メモリ管理フォールトステータス レジスタのフラグは、フォールトの原因を示します。詳細については、P.8-31「メモリ管理フォールトステータス レジスタ」を参照して下さい。

バスフォールトアドレス レジスタ

バスフォールトアドレス レジスタは、バスフォールトが発生した位置のアドレスを読み出すために使用します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xEEE0ED38

アクセス 読み出し / 書き込み

リセット時の状態 予測不能

バスフォールトアドレス レジスタのフィールドの説明を、表 8-27 に示します。

表 8-27 バスフォールトアドレス レジスタのビット割り当て

ビット	フィールド	機能
[31:0]	ADDRESS	バスフォールトアドレス フィールド。ADDRESS は、フォールトが発生したロード/ストアを試みたデータアドレスです。アンアラインドなアクセスでフォールトが発生した場合のアドレスは、仮にそれがフォールトの発生したアドレスでなくても、その命令により要求されたアドレスになります。バスフォールトステータス レジスタのフラグは、フォールトの原因を示します。詳細については、P.8-33「バスフォールトステータス レジスタ」を参照して下さい。

補助フォールトステータス レジスタ

補助フォールトステータス レジスタ (AFSR) は、ソフトウェアに対して追加のシステムフォールト情報を判定するために使用します。

AFSR フラグは、プロセッサの AUXFAULT 入力に直接マップされます。外部ピンが 1 サイクルの間 HIGH レベルになると、対応する AFSR ビットが 1 としてラッチされます。このビットは、対応する AFSR ビットに 1 を書き込むことによってのみクリアされます。

AFSR ビットが書き込まれた場合、または 1 としてラッチされた場合、例外は発生しません。例外が必要な場合は、割り込みを使う必要があります。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xEEE0ED3C  
アクセス 読み出し / 書き込みクリア  
リセット時の状態 0x00000000

AFSR のフィールドの説明を、表 8-28 に示します。

表 8-28 補助フォールトステータス レジスタのビット割り当て

ビット	フィールド	機能
[31:0]	IMPDEF	実装定義。これらのビットは、AUXFAULT 入力への信号割り当てに直接マッピングされます。詳細については、P.A-4「その他」を参照して下さい。

ソフトウェアトリガ割り込みレジスタ

ソフトウェアトリガ割り込みレジスタは、トリガを行う割り込みを保留するために使用します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE000EF00  
アクセス 書き込み専用  
リセット時の状態 0x00000000

ソフトウェアトリガ割り込みレジスタのフィールドを、図 8-21 に示します。

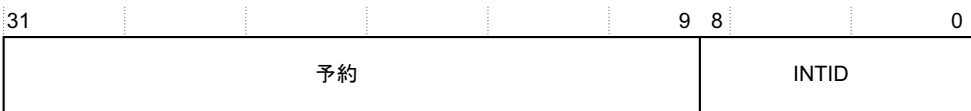


図 8-21 ソフトウェアトリガ割り込みレジスタのビット割り当て

ソフトウェアトリガ割り込みレジスタのフィールドの説明を、表 8-29 に示します。

表 8-29 ソフトウェアトリガ割り込みレジスタのビット割り当て

ビット	フィールド	機能
[31:9]	-	予約
[8:0]	INTID	割り込み ID フィールド。INTID フィールドに値を書き込むことは、割り込み保留セットレジスタの対応する割り込みビットをセットして、割り込みを手動で保留することと同じです。



### 8.3 レベル割り込みとパルス割り込みの比較

プロセッサは、レベル割り込みとパルス割り込みの両方をサポートしています。レベル割り込みは、デバイスにアクセスする **ISR** によってクリアされるまで、アサートされ続けます。パルス割り込みは、エッジモデルの一種です。エッジは、非同期のままではなく、**Cortex-M3** クロック (**HCLK**) の立ち上がりエッジでサンプリングされる必要があります。

レベル割り込みの場合、割り込みルーチンから戻る前に信号がアサート解除されないときは、割り込みは再保留され、再度アクティブになります。これは **FIFO** およびバッファベースのデバイスの場合に特に有用です。これにより、それらのデバイスが余分な作業なしに、1 つの **ISR** によって、または起動の繰り返しによってドレインされることが保証されるのが理由です。これは、デバイスは空になるまで、信号をアサートのまま保持することを意味します。

パルス割り込みは、**ISR** の間に再アサートすることができます。このため、割り込みが同時に保留中とアクティブの両方の状態になる場合があります。アプリケーションの設計では、最初のパルスがアクティブになる前に 2 番目のパルスが到着しないことを保証する必要があります。最初の割り込みがすでに保留しているので、2 番目の保留は無効です。ただし、割り込みが 1 サイクル以上アサートされた場合、**NVIC** は保留ビットをラッチします。**ISR** がアクティブになると、保留ビットはクリアされます。**ISR** がアクティブな間に割り込みが再度アサートされた場合、保留ビットが再度ラッチされることがあります。

パルス割り込みは、ほとんどの場合外部信号によって、一定の間隔で発生する、または繰り返し発生する信号に使用されます。



## 第 9 章

# メモリ保護ユニット

本章では、メモリ保護ユニット (MPU) について説明します。本章は以下のセクションから構成されています。

- *MPU について* : P.9-2
- *MPU のプログラマモデル* : P.9-3
- *割り込みと MPU の更新* : P.9-19
- *MPU のアクセス許可* : P.9-13
- *MPU アボート* : P.9-15
- *MPU 領域の更新* : P.9-16

## 9.1 MPU について

MPU は、メモリを保護するコンポーネントです。プロセッサは、標準の ARMv7 保護メモリシステムアーキテクチャ(Protected Memory System Architecture)(PMSAv7) モデルをサポートしています。MPU は、次の機能を完全にサポートしています。

- 保護領域
- 保護領域のオーバーラップ
- アクセス許可
- システムへのメモリ属性のエクスポート

MPU で不一致やアクセス許可違反が発生すると、優先度をプログラム可能なメモリ管理フォールトハンドラが呼び出されます。詳細については、P.8-38「メモリ管理フォールトアドレスレジスタ」を参照して下さい。

MPU は、次の目的に使用できます。

- 特権ルールの強制
- プロセスの分離
- アクセス規則の強制

## 9.2 MPU のプログラマモデル

このセクションでは、MPU を制御するレジスタについて説明します。このセクションは以下の項目から構成されています。

- MPU レジスタの概要
- MPU レジスタの説明

### 9.2.1 MPU レジスタの概要

MPU レジスタの概要を、表 9-1 に示します。

表 9-1 MPU レジスタ

レジスタ名	タイプ	アドレス	リセット時の値	ページ
MPU タイプレジスタ (MPU Type Register)	読み出し専用	0xE000ED90	0x00000800	P.9-3
MPU 制御レジスタ (MPU Control Register)	読み出し / 書き込み	0xE000ED94	0x00000000	P.9-4
MPU 領域番号レジスタ (MPU Region Number register)	読み出し / 書き込み	0xE000ED98	-	P.9-6
MPU 領域ベースアドレス レジスタ (MPU Region Base Address register)	読み出し / 書き込み	0xE000ED9C	-	P.9-7
MPU 領域属性およびサイズレジスタ (MPU Region Attribute and Size register)	読み出し / 書き込み	0xE000EDA0	-	P.9-8
MPU エイリアス 1 領域ベースアドレス レジスタ (MPU Alias 1 Region Base Address register)	D9C のエイリアス	0xE000EDA4	-	P.9-11
MPU エイリアス 1 領域属性およびサイズレジスタ (MPU Alias 1 Region Attribute and Size register)	DA0 のエイリアス	0xE000EDA8	-	P.9-11
MPU エイリアス 2 領域ベースアドレス レジスタ	D9C のエイリアス	0xE000EDAC	-	P.9-11
MPU エイリアス 2 領域属性およびサイズレジスタ	DA0 のエイリアス	0xE000EDB0	-	P.9-11
MPU エイリアス 3 領域ベースアドレス レジスタ	D9C のエイリアス	0xE000EDB4	-	P.9-11
MPU エイリアス 3 領域属性およびサイズレジスタ	DA0 のエイリアス	0xE000EDB8	-	P.9-11

### 9.2.2 MPU レジスタの説明

このセクションでは、それぞれの MPU レジスタについて説明します。

#### MPU タイプレジスタ

MPU タイプレジスタは、MPU がサポートする領域の数を確認するために使用します。MPU が存在するかどうかを判定するには、ビット [15:8] を読み出します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス	0xE000ED90
アクセス	読み出し専用

リセット時の状態 0x00000800

MPU タイプレジスタのフィールドを、図 9-1 に示します。

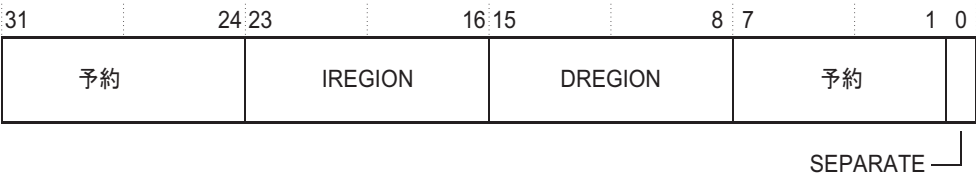


図 9-1 MPU タイプレジスタのビット割り当て

MPU タイプレジスタのフィールドの説明を、表 9-2 に示します。

表 9-2 MPU タイプレジスタのビット割り当て

ビット	フィールド	機能
[31:24]	-	予約
[23:16]	IREGION	このプロセッサコアでは統合 MPU しか使用されないため、IREGION は常に 0x00 に設定されています。
[15:8]	DREGION	サポートされている MPU 領域の数を示すフィールド。8 つの領域をサポートする MPU が実装されている場合は DREGION が 0x08 に設定されますが、そうでない場合は 0x00 に設定されています。
[7:0]	-	予約
[0]	SEPARATE	このプロセッサコアでは統合 MPU しか使用されないため、SEPARATE は常に 0 に設定されています。

MPU 制御レジスタ

MPU 制御レジスタは次の目的に使用します。

- MPU を許可する。
- デフォルトのメモリマップ（バックグラウンド領域）を許可する。
- ハードフォールト、マスク不能割り込み(NMI)、FAULTMASK で昇格されたハンドラ内で MPU を許可する。

MPU が許可されている場合、PRIVDEFENA ビットがセットされていないかぎり、MPU が動作するには、少なくともメモリマップの 1 つの領域が MPU で許可されている必要があります。PRIVDEFENA ビットがセットされており、どの領域も許可されていない場合、特権コード以外は実行できません。

MPU が禁止されている場合は、MPU が存在しない場合と同様に、デフォルトのアドレスマップが使用されます。

MPU が許可されている場合、システムパーティションへのアクセスと、ベクタテーブルのロードアクセスだけがいつでも実行できます。その他の部分へアクセスが可能かどうかは、領域と、PRIVDEFENA ビットがセットされているかどうかに基づいて決定されます。

HFNMIENA がセットされていない場合、例外の優先度が -1 または -2 のときには MPU が許可されません。これらの優先度は、ハードフォールトや NMI が発生した場合、または FAULTMASK が許可されている場合にのみ起こり得ます。この 2 つの優先度で動作している場合、HFNMIENA ビットによって MPU が許可されます。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE000ED94  
 アクセス 読み出し / 書き込み  
 リセット時の状態 0x00000000

MPU 制御レジスタのフィールドを、図 9-2 に示します。

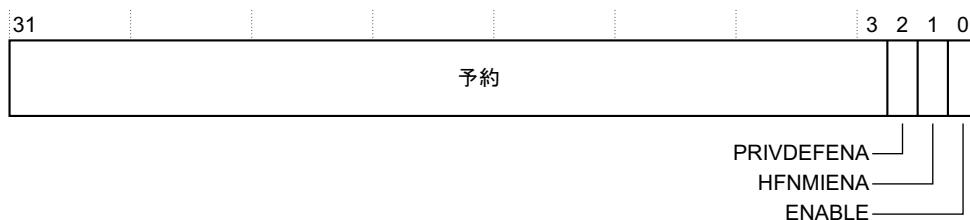


図 9-2 MPU 制御レジスタのビット割り当て

MPU 制御レジスタのフィールドの説明を、表 9-3 に示します。

表 9-3 MPU 制御レジスタのビット割り当て

ビット	フィールド	機能
[31:2]	-	予約
[2]	PRIVDEFENA	<p>このビットは、MPU が許可されている場合に、特権アクセス用のデフォルトのメモリマップをバックグラウンド領域として許可します。バックグラウンド領域は、どの設定可能な領域よりも前に領域番号 1 として存在するかのように機能します。他の領域を設定すると、このデフォルトのマップ上にオーバーレイされ、デフォルト設定より優先されます。</p> <p>このビットを 0 にセットした場合は、デフォルトのメモリマップは禁止され、メモリは領域フォールトでカバーされません。</p> <p>MPU が許可され、PRIVDEFENA が許可されている場合のデフォルトのメモリマップは、第 4 章「メモリマップ」で説明されているとおりです。このメモリマップは、メモリタイプ、実行不可 (XN)、キャッシュ、および共有可の規則が適用されます。ただし、これは特権モード（フェッチとデータアクセス）でのみ適用されます。ユーザモードでは、そのコードとデータ用の領域を設定しない限り、コードはフォールトを発生します。</p> <p>MPU が禁止されている場合は、特権モードとユーザモードの両方のコードでデフォルトのマップが使用されます。</p> <p>XN 規則と SO 規則は、MPU が許可されているかどうかに関係なく、常にシステムパーティションに適用されます。</p> <p>MPU が禁止されている場合、このビットは無視されます。</p> <p>PRIVDEFENA ビットはリセット時にクリアされます。</p>
[1]	HFNMIENA	<p>このビットは、ハードフォールト、NMI、および FAULTMASK で昇格されたハンドラ内で MPU を許可します。このビットが 1 で、ENABLE ビットが 1 の場合は、これらのハンドラ内で MPU が許可されます。このビットが 0 の場合は、ENABLE の値に関係なく、これらのハンドラ内で MPU が禁止されます。このビットが 1 で ENABLE ビットが 0 の場合、動作は予測不能です。</p> <p>HFNMIENA ビットはリセット時にクリアされます。</p>
[0]	ENABLE	<p>MPU 許可ビット</p> <p>1 = MPU を許可</p> <p>0 = MPU を禁止</p> <p>ENABLE ビットはリセット時にクリアされます。</p>

MPU 領域番号レジスタ

MPU 領域番号レジスタは、どの保護領域をアクセスするかを選択するのに使用します。保護領域を選択したら、MPU 領域ベースアドレスレジスタまたは MPU 属性およびサイズレジスタを使用して、その領域の特性を設定します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス                    0xE000ED98

アクセス                    読み出し / 書き込み

リセット時の状態          予測不能





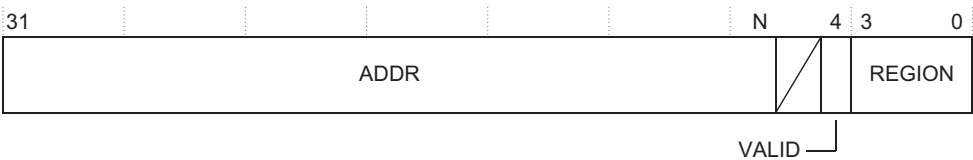


図 9-4 MPU 領域ベースアドレス レジスタのビット割り当て

MPU 領域ベースアドレス レジスタのフィールドの説明を、表 9-5 に示します。

表 9-5 MPU 領域ベースアドレス レジスタのビット割り当て

ビット	フィールド	機能
[31:N]	ADDR	領域ベースアドレス フィールド。ベースアドレスは領域サイズの偶数倍でアラインされるため、N の値は領域サイズに依存します。MPU 領域属性およびサイズレジスタの SZENABLE フィールドで指定されたサイズの 2 のべき乗によって、ベースアドレスで何ビットが使用されるかが決まります。
[4]	VALID	MPU 領域番号有効ビット 1 = MPU 領域番号レジスタのビット [3:0] (REGION 値) が書き換えられます。 0 = MPU 領域番号レジスタは変化せず、解釈されています。
[3:0]	REGION	MPU 領域オーバーライド フィールド

MPU 領域属性およびサイズレジスタ

MPU 領域属性およびサイズレジスタは、MPU アクセス許可を制御するために使用します。このレジスタは、それぞれがハーフワードサイズの 2 つの部分レジスタで構成されています。これらの部分レジスタには、それぞれのサイズを使用してアクセスすることも、ワード操作を使用して同時にアクセスすることもできます。

領域サイズが 32、64、128 バイトの場合、サブ領域禁止ビットは予測不能です。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス                    0xE00EDA0  
アクセス                    読み出し / 書き込み  
リセット時の状態          予測不能

MPU 領域属性およびサイズレジスタのフィールドを、P.9-9 図 9-5 に示します。

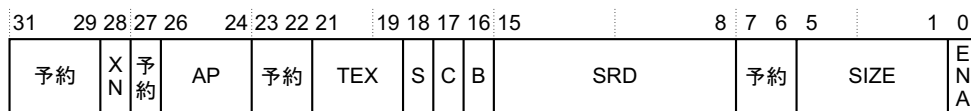


図 9-5 MPU 領域属性およびサイズレジスタのビット割り当て

MPU 領域属性およびサイズレジスタのフィールドの説明を、表 9-6 に示します。詳細については、P.9-13「MPU のアクセス許可」を参照して下さい。

表 9-6 MPU 領域属性およびサイズレジスタのビット割り当て

ビット	フィールド	機能	
[31:29]	-	予約	
[28]	XN	命令アクセス禁止ビット 1 = 命令フェッチを禁止します。 0 = 命令フェッチを許可します。	
[27]	-	予約	
[26:24]	AP	データアクセス許可フィールド	
	値	特権アクセス許可	ユーザアクセス許可
	b000	アクセス不可	アクセス不可
	b001	読み出し / 書き込み	アクセス不可
	b010	読み出し / 書き込み	読み出し専用
	b011	読み出し / 書き込み	読み出し / 書き込み
	b100	予約	予約
	b101	読み出し専用	アクセス不可
	b110	読み出し専用	読み出し専用
	b111	読み出し専用	読み出し専用
[23:22]	-	予約	
[21:19]	TEX	タイプ拡張フィールド	
[18]	S	共有可ビット 1 = 共有可 0 = 共有不可	
[17]	C	キャッシュ可ビット 1 = キャッシュ可 0 = キャッシュ不可	

表 9-6 MPU 領域属性およびサイズレジスタのビット割り当て

ビット	フィールド	機能
[16]	B	バッファ可ビット 1 = バッファ可 0 = バッファ不可
[15:8]	SRD	サブ領域禁止 (SRD) (Sub-Region Disable) フィールド。SRD ビットをセットすると、対応するサブ領域が禁止されます。領域は、同じサイズの 8 つのサブ領域に分割されます。128 バイト以下の領域サイズについては、サブ領域はサポートされません。詳細については、P.9-12 「サブ領域」を参照して下さい。
[7:6]	-	予約
[5:1]	SIZE	MPU 保護領域サイズフィールド。詳細については、表 9-7 を参照して下さい。
[0]	ENABLE	領域許可ビット

アクセス許可の詳細については、P.9-13 「MPU のアクセス許可」を参照して下さい。

表 9-7 MPU 保護領域サイズフィールド

領域	サイズ
b00000	予約
b00001	予約
b00010	予約
b00011	予約
b00100	32B
b00101	64B
b00110	128B
b00111	256B
b01000	512B
b01001	1KB
b01010	2KB
b01011	4KB
b01100	8KB

表 9-7 MPU 保護領域サイズフィールド

領域	サイズ
b01101	16KB
b01110	32KB
b01111	64KB
b10000	128KB
b10001	256KB
b10010	512KB
b10011	1MB
b10100	2MB
b10101	4MB
b10110	8MB
b10111	16MB
b11000	32MB
b11001	64MB
b11010	128MB
b11011	256MB
b11100	512MB
b11101	1GB
b11110	2GB
b11111	4GB

### 9.2.3 エイリアスレジスタを使用した MPU へのアクセス

レジスタのエイリアシングを使用して、MPU レジスタのロード速度を最適化することができます。ネスト型ベクタ割り込みコントローラ (NVIC) のエイリアスレジスタが 3 セットあります。これらのレジスタの詳細については、P.8-7 「NVIC レジスタの説明」を参照して下さい。

エイリアスによるレジスタへのアクセスは、通常の方法と全く同じです。これらのエイリアスを使用すると、シーケンシャル書き込み (STM) によって 1 ～ 4 個の領域を更新することができます。この方法は、禁止 / 変更 / 許可の必要がない場合に使用します。

領域の内容を読み出す場合は領域番号を書き込む必要があるため、エイリアスは使用できません。

4 つの領域を更新するためのコードシーケンスの例を次に示します。

```
; R1 = 4 region pairs from process control block (8 words)
MOVR0, #NVIC_BASE
ADD R0, #MPU_REG_CTRL
LDM R1, [R2-R9]; load region information for 4 regions
STM R0, [R2-R9]; update all 4 regions at once
```

---

**注**

---

C/C++ コンパイラでは、通常はこのシーケンスに `memcpy()` 関数を使用することができます。ただし、コンパイラでワード転送を使用していることを確認する必要があります。

---

## 9.2.4 サブ領域

領域属性およびサイズレジスタにある 8 つのサブ領域禁止 (SRD) ビットによって、領域がそのサイズに基づいて、同じサイズの 8 つのユニットに分割されます。これによって、いくつかの 1/8 サブ領域を選択的に禁止することができます。最下位ビットは最初の 1/8 サブ領域に、最上位ビットは最後の 1/8 サブ領域に対応します。あるサブ領域を禁止すると、その範囲とオーバーラップしている別の領域が代わりに一致をとるようにします。そのサブ領域とオーバーラップする別の領域が存在しない場合は、デフォルトの動作が採用され、不一致フォールトが発生します。サブ領域は、最も小さい 3 つの領域サイズ、つまり 32、64、128 のサイズでは使用できません。これらのサブ領域を使用した場合、結果は予測不能です。

### SRD の使用例

同じベースアドレスを持つ 2 つの領域がオーバーラップしています。1 つは 64KB の領域で、もう 1 つは 512KB の領域です。512KB の領域の下位 64KB を禁止して、64KB の領域の属性を適用します。これは、512KB の領域の SRD を b11111110 に設定することによって実現されます。

### 9.3 MPU のアクセス許可

このセクションでは、MPU のアクセス許可について説明します。領域アクセス制御レジスタ（P.9-8「MPU 領域属性およびサイズレジスタ」参照）のアクセス許可ビットである TEX、C、B、AP、XN によって、対応するメモリ領域へのアクセスが制御されます。必要な許可なしにメモリ領域へアクセスを行うと、アクセス許可フォールトが発生します。

TEX、C、B のエンコードの説明を、表 9-8 に示します。

表 9-8 TEX、C、B のエンコード

TEX	C	B	説明	メモリタイプ	領域の共有可能性
b000	0	0	ストロングリオーダ	ストロングリオーダ	共有可
b000	0	1	共有デバイス	デバイス	共有可
b000	1	0	外部および内部ライトスルー、書き込み割り当てなし	ノーマル	S
b000	1	1	外部および内部ライトバック、書き込み割り当てなし	ノーマル	S
b001	0	0	外部および内部でキャッシュ不可	ノーマル	S
b001	0	1	予約	予約	予約
b001	1	0	実装定義		
b001	1	1	外部および内部ライトバック、読み出し / 書き込み割り当て	ノーマル	S
b010	0	0	共有不可デバイス	デバイス	共有不可
b010	0	1	予約	予約	予約
b010	1	X	予約	予約	予約
b1BB	A	A	キャッシュされたメモリ、BB = 外部ポリシー、AA = 内部ポリシー	ノーマル	S

#### 注

表 9-8 の S は、MPU 領域属性およびサイズレジスタの S ビット [2] を意味します。

メモリ属性エンコードのキャッシュポリシーの説明を、表 9-9 に示します。

表 9-9 メモリ属性エンコードのキャッシュポリシー

メモリ属性エンコード (AA と BB)	キャッシュポリシー
00	キャッシュ不可
01	ライトバック、読み出し / 書き込み割り当て
10	ライトスルー、書き込み割り当てなし
11	ライトバック、書き込み割り当てなし

——— 注 ———

**HPROT** と **MEMATTR** で示されるすべてのキャッシュポリシーは、外部キャッシュに関係します。

AP のエンコードの説明を、表 9-10 に示します。

表 9-10 AP のエンコード

AP[2:0]	特権アクセス許可	ユーザアクセス許可	説明
000	アクセス不可	アクセス不可	すべてのアクセスでアクセス許可フォールトが発生します。
001	読み出し / 書き込み	アクセス不可	特権アクセスのみ
010	読み出し / 書き込み	読み出し専用	ユーザモードで書き込みを行うと、アクセス許可フォールトが発生します。
011	読み出し / 書き込み	読み出し / 書き込み	完全アクセス
100	予測不能	予測不能	予約
101	読み出し専用	アクセス不可	特権読み出しのみ
110	読み出し専用	読み出し専用	特権 / ユーザ読み出しのみ
111	読み出し専用	読み出し専用	特権 / ユーザ読み出しのみ

XN のエンコードの説明を、表 9-11 に示します。

表 9-11 XN のエンコード

XN	説明
0	すべての命令フェッチを許可
1	すべての命令フェッチを禁止



## 9.4 MPU アボート

MPU アボートの詳細については、P.8-38「メモリ管理フォールトアドレス レジスタ」を参照して下さい。

## 9.5 MPU 領域の更新

MPU 領域をプログラムするための、メモリマップされた 3 ワードで構成される 3 つのレジスタが存在します。これらのレジスタは部分レジスタで、個別にプログラムやアクセスが可能です。つまり、既存の ARMv6、ARMv7、および CP15 のコードを移植することができます。この場合、MRC 命令と MCR 命令が LDRx 命令と STRx 命令に置き換えられます。

また、これらのレジスタに 3 つのワードとしてアクセスし、そのうちの 2 ワードだけを使用してプログラムすることもできます。STM 命令を使用して一組の領域を同時にプログラムできるエイリアスが提供されています。

### 9.5.1 CP15 と等価なコードを使用した MPU 領域の更新

CP15 と等価なコードの使用例を次に示します。

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
MOV R0,#NVIC_BASE
ADD R0,#MPU_REG_CTRL
STR R1,[R0,#0]; region number
STR R4,[R0,#4]; address
STRHR2,[R0,#8]; size and enable
STRHR3,[R0,#10]; attributes
```

---

#### 注

この期間に割り込みが横取り可能な場合、この領域はその動作に影響することがあります。つまり領域を、まず禁止し、書き込んでから、許可する必要があります。通常、コンテキストスイッチャではこの操作は必要ありませんが、他の場所で更新が行われる場合は必要になります。

---

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
MOV R0,#NVIC_BASE
ADD R0,#MPU_REG_CTRL
STR R1,[R0,#0]; region number
BIC R2,R2, #1; disable
STRHR2,[R0,#8]; size and enable
STR R4,[R0,#4]; address
STRHR3,[R0,#10]; attributes
ORR R2,#1 ; enable
STRHR2,[R0,#8]; size and enable
```

専用ペリフェラルバスはストロングリオーダータイプのメモリエリアのため、DMB/DSB は必要ありません。ただし、コンテキストスイッチャの終了などによって MPU への変更が適用される前に、DSB が必要になります。

ISB は、MPU 領域をプログラムするコードが分岐または呼び出しによって開始された場合に必要です。例外からの復帰または例外の取得によってコードが開始された場合は、ISB は必要ありません。

### 9.5.2 2 ワードまたは 3 ワードを使用した MPU 領域の更新

情報がどう分割されているかによって、2 ワードまたは 3 ワードを使用して直接プログラムすることができます。

```
; R1 = region number
; R2 = address
; R3 = size, attributes in one
MOV R0,#NVIC_BASE
ADD R0,#MPU_REG_CTRL
STR R1,[R0,#0]; region number
STR R2,[R0,#4]; address
STR R3,[R0,#8]; size, attributes
```

次のように STM で最適化することができます。

```
; R1 = region number
; R2 = address
; R3 = size, attributes in one
MOV R0,#NVIC_BASE
ADD R0,#MPU_REG_CTRL
STM R0,{R1-R3}; region number, address, size, and attributes
```

パック済みの情報については、2 ワードでこの操作を実行できます。つまり、ベースアドレスレジスタは領域 - 有効ビットに加えて領域番号を格納しています。この方法は、ブートリストやプロセス制御ブロック (PCB)(Process Control Block) のようにデータが静的にパックされている場合に有効です。

```
; R1 = address and region number in one
; R2 = size and attributes in one
MOV R0,#NVIC_BASE
ADD R0,#MPU_REG_CTRL
STR R1,[R0,#4]; address and region number
STR R2,[R0,#8]; size and attributes
```

次のように STM で最適化することができます。

```
; R1 = address and region number in one
; R2 = size and attributes in one
MOV R0,#NVIC_BASE
ADD R0,#MPU_REG_CTRL
STM R0,{R1-R2}; address, region number, size
```

割り込みと MPU の更新の詳細については、P.9-19「割り込みと MPU の更新」を参照して下さい。

## 9.6 割り込みと MPU の更新

MPU 領域にはクリティカルなデータが含まれていることがあります。これは、更新に複数のバス トランザクションが必要なためです。これは、通常は 2 ワードです。そのため、スレッド切り替えには注意が必要です。つまり、割り込みによって 2 つのワードが分断され、その領域を不整合な情報のままにしてしまう可能性があります。これによって、2 種類の問題が発生します。

- 割り込みが発生し、その割り込みでも MPU が更新される可能性があります。この場合は、読み出し - 変更 - 書き込みの問題だけでなく、その割り込みルーチンが同じ領域を変更しないことを保証しているような場合にも影響します。これは、プログラムは、レジスタに書き込まれる領域番号に依存しているので、どの領域を更新するかを知っているからです。したがって、このケースでは各更新ルーチンの間は割り込みを禁止する必要があります。
- 更新されている領域を使用する、または、ベースフィールドあるいはサイズフィールドのいずれかのみが更新されてしまったので影響を受けてしまう割り込みが発生する可能性があります。サイズフィールドが変更され、ベースフィールドが変更されなかった場合は、このベースとサイズから構成されるアドレスが、通常は他の領域で処理されるエリア中にまでオーバーラップしてしまう可能性があります。このケースでは、禁止 - 変更 - 許可の手法を使用する必要があります。

ただし、ユーザ領域を切り替える標準の OS コンテキストスイッチ コードでは、これらの領域がユーザ特権かつユーザエリア アドレスに事前設定されているため、問題はありません。つまり、割り込みが発生しても副作用は起きません。そのため、許可 / 禁止コードを使用したり、割り込みを禁止したりする必要はありません。

最も一般的な手法は、ブートコードとコンテキストスイッチャのみで MPU のプログラムを行うことです。プログラムを行う場所がこの 2 か所のみで、コンテキストスイッチャではユーザ領域しか更新されない場合、割り込みの禁止は必要ありません。コンテキストスイッチャは最初からクリティカルな領域であり、ブートコードは割り込み禁止状態で実行されるためです。



## 第 10 章

# コアデバッグ

本章では、プロセッサのデバッグとテストを行う方法について説明します。本章は以下のセクションから構成されています。

- コアデバッグについて : P.10-2
- コアデバッグ レジスタ : P.10-3
- コア デバッグアクセスの例 : P.10-12
- コアデバッグでのアプリケーションレジスタの使用法 : P.10-13

10.1 コアデバッグについて

コアデバッグには、コアデバッグ レジスタ経由でアクセスします。これらのレジスタへデバッグアクセスを行うは、アドバンスト ハイパフォーマンス バス (AHB-AP) ポートを使用します。詳細については、P.11-38「*AHB-AP*」を参照して下さい。プロセッサは、内部の専用ペリフェラルバス (PPB) 経由で、これらのレジスタに直接アクセスできます。

コアデバッグ レジスタの説明を、表 10-1 に示します。

表 10-1 コアデバッグ レジスタ

アドレス	タイプ	リセット時の値	説明
0xE000EDF0	読み出し / 書き込み	0x00000000 <sup>a</sup>	デバッグホールド制御およびステータスレジスタ
0xE000EDF4	書き込み専用	-	デバッグコアレジスタ セレクタレジスタ (Debug Core Register Selector Register)
0xE000EDF8	読み出し / 書き込み	-	デバッグコアレジスタ データレジスタ (Debug Core Register Data Register)
0xE000EDFC	読み出し / 書き込み	0x00000000 <sup>b</sup>	デバッグ例外およびモニタ制御レジスタ (Debug Exception and Monitor Control Register)

- a. ビット 5、3、2、1、0 は **PORESETn** によってリセットされます。ビット [1] も **SYSRESETn** によって、またはアプリケーション割り込みおよびリセット制御レジスタの **VECTRESET** ビットに 1 を書き込むことによってリセットされます。
- b. ビット 16、17、18、19 も **SYSRESETn** によって、またはアプリケーション割り込みおよびリセット制御レジスタの **VECTRESET** ビットに 1 を書き込むことによってリセットされます。

デバッグフォールト ステータスレジスタも使用されます。詳細については、P.8-37「デバッグフォールトステータス レジスタ」を参照して下さい。

10.1.1 ホールトモードでのデバッグ

デバッグは、デバッグホールド制御およびステータスレジスタの **C\_DEBUGEN** および **C\_HALT** ビットをセットすることでコアをホールトできます。コアは、ホールトされたときにデバッグホールド制御およびステータスレジスタの **S\_HALT** ビットをセットして応答します。

コアをホールトしてから、**C\_STEP** ビットを 1 にセットし、**C\_HALT** ビットを 0 にクリアすると、コアをシングルステップ実行できます。コアは、ステップ実行を完了したときにデバッグホールド制御およびステータスレジスタの **S\_HALT** ビットをセットして応答し、再度ホールト状態になります。

10.1.2 コアデバッグの退出

コアは、デバッグホールド制御およびステータスレジスタの **C\_DEBUGEN** ビットをクリアすることでホールトから抜けられます。



## 10.2 コアデバッグ レジスタ

デバッグ機能は次のレジスタによって提供されます。

- デバッグホールト制御およびステータスレジスタ：P.10-3
- デバッグ例外およびモニタ制御レジスタ：P.10-8
- デバッグコアレジスタ データレジスタ：P.10-7
- デバッグ例外およびモニタ制御レジスタ：P.10-8

### 10.2.1 デバッグホールト制御およびステータスレジスタ

デバッグホールト制御およびステータスレジスタ (DHCSR) の目的は次のとおりです。

- プロセッサの状態についてのステータス情報を提供する。
- コアデバッグを許可する。
- プロセッサのホールトとステップ実行を行う。

DHCSR には次の特徴があります。

- 32 ビットの読み出し / 書き込みレジスタである。
- アドレスは 0xE000EDF0 である。

---

#### —— 注 ——

DHCSR は、パワーオンを含むシステムリセットでのみリセットされます。DHCSR のビット 16 のリセット時の値は予測不能です。

---

このレジスタのビット配置を、図 10-1 に示します。

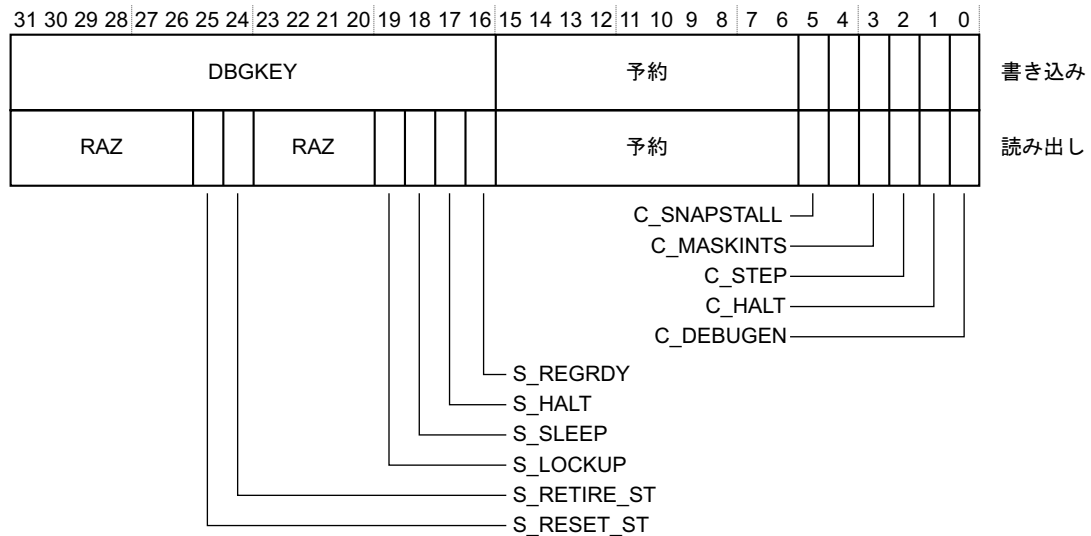


図 10-1 デバッグホールト制御およびステータスレジスタのフォーマット

デバッグ ID レジスタのビットの機能を、表 10-2 に示します。

表 10-2 デバッグホールト制御およびステータスレジスタ

ビット	タイプ	フィールド	機能
[31:16]	書き込み	<b>DBGKEY</b>	デバッグキー。このレジスタに書き込むときは、必ず 0xA05F を書き込む必要があります。ステータスビット [25:16] として読み出されます。キーとして書き込まれない場合、書き込み操作は無視され、どのビットもレジスタに書き込まれません。
[31:26]	-	-	予約、RAZ
[25]	読み出し	<b>S_RESET_ST</b>	このビットが最後に読み出された後でコアがリセットされた、または現在リセット中であることを示します。このビットはスティッキービットで、読み出し時にクリアされます。このため、2 回読み出して最初に 1、次に 0 が読み出された場合、以前にリセットされたことを示します。2 回読み出して 2 回とも 1 が読み出された場合、現在リセット中である（まだリセット状態に置かれている）ことを示します。
[24]	読み出し	<b>S_RETIRED_ST</b>	最後に読み出された後で命令が完了したことを示します。このビットはスティッキービットで、読み出し時にクリアされます。このビットを使用して、コアがロード/ストアやフェッチでストールしているかどうかを判断できます。
[23:20]	-	-	予約、RAZ
[19]	読み出し	<b>S_LOCKUP</b>	コアが（ホールト中ではなく）実行中で、ロックアップ条件が存在している場合に 1 が読み出されます。

表 10-2 デバッグホールド制御およびステータスレジスタ

ビット	タイプ	フィールド	機能
[18]	読み出し	<b>S_SLEEP</b>	コアがスリープ状態（WFI、WFE、または SLEEP-ON-EXIT）であることを示します。 <b>C_HALT</b> を使用して制御を取り戻すか、割り込みでウェークアップするのを待つ必要があります。SLEEP-ON-EXIT の詳細については、P.7-3 表 7-1 を参照して下さい。
[17]	読み出し	<b>S_HALT</b>	<b>S_HALT</b> がセットされているときは、コアはデバッグ状態です。
[16]	読み出し	<b>S_REGRDY</b>	デバッグコアレジスタ セレクタレジスタで、レジスタの読み出し / 書き込みが使用できます。最後の転送が完了しました。
[15:6]	-	-	予約
[5]	読み出し / 書き込み	<b>C_SNAPSTALL</b>	ロード / ストア操作でコアがストールしている場合、ストールが解除され、命令が強制的に完了します。これによって、ホールドデバッグがコアの制御を取り戻せます。このビットは、次の条件が満たされている場合のみセットできます。 <b>C_DEBUGEN</b> = 1 <b>C_HALT</b> = 1 コアは、 <b>S_RETIRE_ST</b> を 0 として読み出します。これは、進行した命令がないことを示しています。これによって、使用ミスが回避されます。 これが使用された場合、バス状態は予測不能です。 <b>S_RETIRE</b> は、ロード / ストア操作時のコアのストールを検出できます。
[4]	-	-	予約
[3]	読み出し / 書き込み	<b>C_MASKINTS</b>	ステップ実行時、またはホールドデバッグでの実行時に割り込みをマスクします。NMI はマスク不可なので、NMI には影響しません。プロセッサがホールドしているとき（ <b>S_HALT</b> == 1）にのみ変更する必要があります。
[2]	読み出し / 書き込み	<b>C_STEP</b>	ホールドデバッグでコアをステップ実行します。 <b>C_DEBUGEN</b> = 0 のときは、このビットは無効です。プロセッサがホールドしているとき（ <b>S_HALT</b> == 1）にのみ変更する必要があります。
[1]	読み出し / 書き込み	<b>C_HALT</b>	コアをホールドします。このビットは、ブレークポイントなど、コアがホールドするときに自動的にセットされます。コアのリセット時には、このビットはクリアされます。このビットは、 <b>C_DEBUGEN</b> が 1 のときにのみ書き込み可能で、それ以外のときは無視されます。このビットを 1 にセットするとき、 <b>C_DEBUGEN</b> にも同じ値として 1 をセットする必要があります（値 [1:0] が 2'b11）。 <b>C_DEBUGEN</b> がすでに 1 で、b11 が書き込まれたときのみ、コアがホールドできます。
[0]	読み出し / 書き込み	<b>C_DEBUGEN</b>	デバッグを許可します。このビットは、AHB-AP によってのみ書き込み可能で、コアからは書き込めません。コアはこのビットをセットまたはクリアできないため、コアからの書き込みは無視されます。コアが自分をホールドするために <b>C_HALT</b> へ書き込むときは、コアはこのビットに 1 を書き込む必要があります。

**C\_DEBUGEN** = 1 で、ホールドモードが許可されていない場合、他のすべてのフィールドは無効です。

このレジスタは、システムリセット時にリセットされません。パワーオンリセット時にはリセットされます。ただし、**C\_HALT** ビットはシステムリセット時に常にクリアされます。

リセット時にホールドするには、次のビットを許可する必要があります。

- デバッグ例外およびモニタ制御レジスタのビット [0] (**VC\_CORERESET**)
- デバッグホールド制御およびステータスレジスタのビット [0] (**C\_DEBUGEN**)

——— 注 ———

このレジスタにワード以外のサイズで書き込みを行った場合、結果は予測不能です。読み出しは任意のサイズで許可され、この方法を使用してスティッキービットの意図しない変更を防止できます。

10.2.2 デバッグコアレジスタ セレクタレジスタ

デバッグコアレジスタ セレクタレジス (DCRSR) は、データの転送先または転送元となるプロセッサレジスタを選択するために使用します。

DCSRC には次の特徴があります。

- 17 ビットの書き込み専用レジスタである。
- アドレスは 0xE000EDF4 である。

このレジスタのビット配置を、図 10-2 に示します。

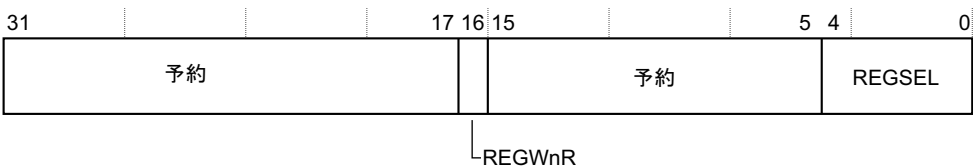


図 10-2 デバッグコアレジスタ セレクタレジスタのフォーマット

デバッグコアレジスタ セレクタレジスタのビットの機能を、表 10-3 に示します。

表 10-3 デバッグコアレジスタ セレクタレジスタ

ビット	タイプ	フィールド	機能
[31:17]	-	-	予約

表 10-3 デバッグコアレジスタ セレクタレジスタ

ビット	タイプ	フィールド	機能
[16]	書き込み	REGWnR	1 = 書き込み 0 = 読み出し
[15:5]	-	-	-
[4:0]	書き込み	REGSEL	5b00000 = R0 5b00001 = R1 ... 5b01111 = DebugReturnAddress() 5b10000 = xPSR/ フラグ、実行番号、状態情報 5b10001 = MSP (メイン SP) 5b10010 = PSP (プロセス SP) 5b10011 = RAZ/WI 未使用の値はすべて予約されています。

このレジスタは書き込み専用で、データをデバッグコアレジスタ データレジスタと選択されたレジスタとの間で転送するため、コアとのハンドシェークを生成します。コアのトランザクションが完了するまで、DHCSR のビット [16] (**S\_REGRDY**) は 0 に保持されます。

注

- このレジスタにワード以外のサイズで書き込みを行った場合、結果は予測不能です。
- PSR レジスタはこの方法で完全にアクセス可能です。これに対して、MRS 命令を使用した場合、一部は 0 として読み出されます。
- すべてのビットは書き込み可能ですが、一部の組み合わせでは実行が再開されたときにフォールトが発生します。
- IT は IT ブロック中でも、書き込まれて、動作する可能性があります。
- ICI は書き込み可能ですが、無効な値が書き込まれた場合や、LDM/STM で使用されていない場合は、例外からの復帰のように、フォールトが発生します。ICI をある値から 0 に変更すると、元になる LDM/STM は継続ではなく、最初から開始します。

10.2.3 デバッグコアレジスタ データレジスタ

デバッグコアレジスタ データレジスタ (DCRDR) の目的は、プロセッサとの間でレジスタの読み出しと書き込みを行うためのデータを保持することです。

DCRDR には次の特徴があります。

- 32 ビットの読み出し / 書き込みレジスタである。
- アドレスは 0xE000EDF8 である。

これは、デバッグコアレジスタ セレクタレジスタによって選択されたレジスタに書き込まれるデータの値です。

プロセッサがデバッグコアレジスタ セレクタからの要求を受信すると、このレジスタは通常のロード / ストアユニットの操作を使用して、プロセッサにより読み出しまたは書き込みされます。

コアレジスタの転送が実行されない場合、OS RSD や Real View Monitor などのソフトウェアベースのデバッグモニタが、このレジスタを非ホールドデバッグでの通信に使用できます。これによって、フラグやビットにより状態を応答し、コマンドに対して受け付け、応答、または両方が行われたかどうかを示すことができます。

## 10.2.4 デバッグ例外およびモニタ制御レジスタ

デバッグ例外およびモニタ制御レジスタ (DEMCR) は次の目的に使用されます。

- ベクタキャッチ。つまり、指定のベクタが実行のために読み出されるとき、デバッグエントリを発生します。
- デバッグモニタの制御

DEMCR には次の特徴があります。

- 32 ビットの読み出し / 書き込みレジスタである。
- アドレスは 0xE000EDFC である。

このレジスタのビット配置を、P.10-6 図 10-2 に示します。

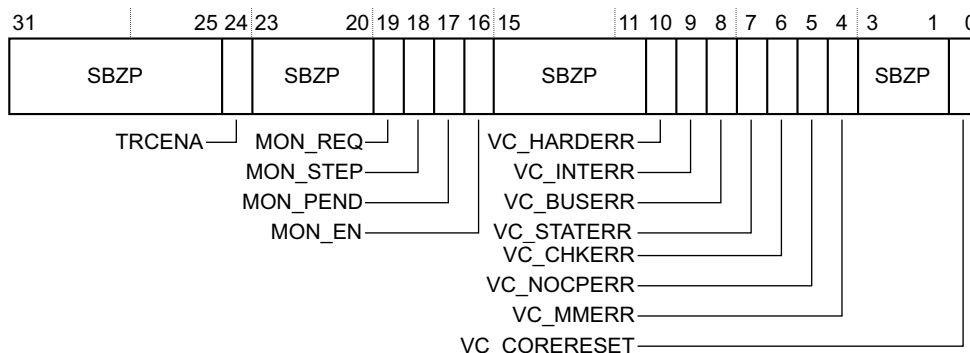


図 10-3 デバッグ例外およびモニタ制御レジスタのフォーマット

デバッグ例外およびモニタ制御レジスタのビットの機能を、表 10-4 に示します。

**表 10-4 デバッグ例外およびモニタ制御レジスタ**

ビット	タイプ	フィールド	機能
[31:25]	-	-	予約、SBZP
[24]	読み出し / 書き込み	<b>TRCENA</b>	<p>次のトレースおよびデバッグブロックの使用を許可するには、このビットを 1 にセットする必要があります。</p> <ul style="list-style-type: none"> <li>データウォッチポイントおよびトレース (DWT)</li> <li>計装トレース マクロセル (ITM)</li> <li>エンベデッドトレース マクロセル (ETM)</li> <li>トレースポート インタフェースユニット (TPIU)</li> </ul> <p>これによって、トレースが必要ない場合に消費電力の管理が許可されます。アプリケーションは、ITM、またはデバッガで使用するために、この機能を許可できます。</p> <p>——— 注 ———</p> <p>実装時に、CM3Defs.v で TIEOFF_TRCENA の定義がコメント解除されている場合、TRCENA はセットできません。</p>
[23:20]	-	-	予約、SBZP
[19]	読み出し / 書き込み	<b>MON_REQ<sup>a</sup></b>	<p>モニタによって、ウェークアップの発生した原因を識別するために使用されます。</p> <p>1 = MON_PEND によるウェークアップ</p> <p>0 = デバッグ例外によるウェークアップ</p>
[18]	読み出し / 書き込み	<b>MON_STEP<sup>a</sup></b>	<p><b>MON_EN</b>=1 の場合、このビットによってコアがステップ実行されます。<b>MON_EN</b>=0 の場合、このビットは無視されます。これは、<b>C_STEP</b> と等価です。割り込みは、モニタの優先度と、<b>PRIMAST</b>、<b>FAULTMASK</b>、または <b>BASEPRI</b> の設定に従ってのみステップ実行されます。</p>
[17]	読み出し / 書き込み	<b>MON_PEND<sup>a</sup></b>	<p>優先度で許可されたときにアクティブになるようにモニタを保留します。これにより、<b>AHB-AP</b> ポートを経由して、モニタをウェークアップすることができます。モニタデバッグ用の <b>C_HALT</b> と等価です。</p> <p>このレジスタは、システムリセット時にリセットされません。パワーオンリセット時にのみリセットされます。リセットハンドラまたはそれ以後のソフトウェア、あるいは <b>DAP</b> によってデバッグモニタを許可する必要があります。</p>

表 10-4 デバッグ例外およびモニタ制御レジスタ

ビット	タイプ	フィールド	機能
[16]	読み出し / 書き込み	<b>MON_EN<sup>a</sup></b>	<p>デバッグモニタを許可します。許可されると、優先度レベルはシステムハンドラの優先度レジスタによって制御されます。禁止されている場合、すべてのデバッグイベントはハードフォールトになります。デバッグホールト制御およびステータスレジスタの <b>C_DEBUGEN</b> は、このビットに優先します。</p> <p>ベクタキャッチは半同期です。一致するイベントが検出されると、ホールト要求が発生します。プロセッサは命令の境界でのみホールト可能なため、次の命令の境界まで待つ必要があります。その結果、プロセッサは例外ハンドラの最初の命令で停止します。ただし、ベクタキャッチがトリガされたときは、2つの特別なケースがあります。</p> <ul style="list-style-type: none"> <li>ベクタ、ベクタ読み出し、またはスタックプッシュエラーの間にフォールトが取得された場合、ベクタエラーまたはスタックプッシュに対応するフォールトハンドラでホールトが発生します。</li> <li>ベクタ処理の間に到着した後着割り込みは取得されません。つまり、後着の最適化をサポートする実装では、この場合に後着割り込みを抑制する必要があります。</li> </ul>
[15:11]	-	-	予約、SBZP
[10]	読み出し / 書き込み	<b>VC_HARDERR<sup>b</sup></b>	ハードフォールトでのデバッグトラップ
[9]	読み出し / 書き込み	<b>VC_INTERR<sup>b</sup></b>	割り込み / 例外へのサービスのエラーによるデバッグトラップ。これらは他のフォールトのサブセットで、 <b>BUSERR</b> または <b>HARDERR</b> の前にキャッチされます。
[8]	読み出し / 書き込み	<b>VC_BUSERR<sup>b</sup></b>	通常のバスエラーによるデバッグトラップ
[7]	読み出し / 書き込み	<b>VC_STATERR<sup>b</sup></b>	用法フォールト状態エラーでのデバッグトラップ
[6]	読み出し / 書き込み	<b>VC_CHKERR<sup>b</sup></b>	用法フォールトイネーブルチェックエラーでのデバッグトラップ
[5]	読み出し / 書き込み	<b>VC_NOCERR<sup>b</sup></b>	存在しないか、または <b>CAR</b> レジスタで存在しないとマークされているコプロセッサへの用法フォールトアクセスでのデバッグトラップ
[4]	読み出し / 書き込み	<b>VC_MMERR<sup>b</sup></b>	メモリ管理フォールトでのデバッグトラップ
[3:1]	-	-	予約、SBZP
[0]	読み出し / 書き込み	<b>VC_CORERESET<sup>b</sup></b>	リセットベクタキャッチ。コアリセットが発生すると、実行中のシステムがホールトします。

a. コアリセット時には、このビットはクリアされます。

b. **C\_DEBUGEN** = 1 の場合のみ使用できます。



このレジスタは、デバッグ下の例外の動作を管理します。

ベクタキャッチは、ホールトデバッグでのみ使用できます。上位ハーフワードはモニタ制御用に、下位ハーフワードはホールト例外サポートに使用されます。

このレジスタは、システムリセット時にリセットされません。

パワーオンリセット時にはリセットされます。ビット [19:16] は、コアリセット時に常にクリアされます。デバッグモニタは、リセットハンドラまたはそれ以降のソフトウェア、あるいは AHB-AP ポートによって許可されます。

ベクタキャッチは半同期です。一致するイベントが検出されると、ホールト要求が発生します。プロセッサは命令の境界でのみホールト可能なため、次の命令の境界まで待つ必要があります。その結果、プロセッサは例外ハンドラの最初の命令で停止します。ただし、ベクタキャッチがトリガされたときは、2つの特別なケースがあります。

1. ベクタ読み出しまたはスタックプッシュの間にフォールトが検出された場合、ベクタエラーまたはスタックプッシュエラーのためのフォールトハンドラで、ホールトが発生します。
2. ベクタ読み出しまたはスタックプッシュ エラー時に検出された後着割り込みは処理されません。つまり、後着の最適化をサポートする実装では、この場合に後着割り込みを抑える必要があります。

## 10.3 コア デバッグアクセスの例

プロセッサをホールトしてレジスタの 1 つに値を書き込むには、次のシーケンスを実行します。

1. デバッグホールト制御およびステータスレジスタに 0xA05F0003 を書き込みます。これによって、デバッグが許可され、コアがホールトします。
2. デバッグホールト制御およびステータスレジスタの **S\_HALT** ビットがセットされるまで待ちます。このビットは、コアがホールトしたことを示します。
3. 書き込む値を、デバッグコアレジスタ データレジスタに書き込みます。
4. 書き込み先のレジスタ番号を、デバッグコアレジスタ セクタレジスタに書き込みます。

## 10.4 コアデバッグでのアプリケーションレジスタの使用法

アプリケーションレジスタは、ステータスへのアクセスと、システムへ変更を適用するためにも使用できます。

アプリケーションレジスタをコアデバッグに使用する場合、次の点に注意してください。

- AHB-AP とアプリケーションの両方がこれらのレジスタを変更する場合、読み出し - 変更 - 書き込みの問題が発生します。
- PENDSET や PENDCLR のような書き込みレジスタの場合、これらのレジスタは最初に読み出されないため、読み出し - 変更 - 書き込みの問題が発生します。
- 優先度を含むレジスタと、他の読み出し / 書き込みレジスタの場合、読み出し - 変更 - 書き込み操作を実行するときに、読み出しと書き込みとの間でレジスタに変更が発生する可能性があります。いくつかの場合には、この問題を軽減するためにバイトアクセスが許可されており、プロセッサが動作しているときには、デバッグはこの問題に注意を払う必要があります。

コアデバッグで主に使用されるアプリケーションレジスタと、レジスタのビットを表 10-5 に示します。アプリケーションレジスタの完全な一覧については、『*ARMv7-M アーキテクチャ リファレンスマニュアル*』を参照して下さい。

**表 10-5 コアデバッグで使用されるアプリケーションレジスタ**

レジスタ	コアデバッグで使用されるビットまたはフィールド
割り込み制御状態	<b>ISRPREEMPT</b> <b>ISRPENDING</b> <b>VECTPENDING</b>
ベクタテーブル オフセット	ベクタテーブルの検索用
アプリケーション割り込み / リセット制御	<b>VECTCLRACTIVE</b> <b>ENDIANESS</b>
構成制御	<b>DIV_0_TRP</b> <b>UNALIGN_TRP</b>
システムハンドラ制御および状態	<b>ACTIVE</b> <b>PENDED</b>



## 第 11 章

# システムデバッグ

本章では、プロセッサのシステムデバッグについて説明します。本章は以下のセクションから構成されています。

- システムデバッグについて : P.11-2
- システムデバッグ アクセス : P.11-3
- システムデバッグのプログラマモデル : P.11-5
- *FPB* : P.11-6
- *DWT* : P.11-13
- *ITM* : P.11-29
- *AHB-AP* : P.11-38

## 11.1 システムデバッグについて

プロセッサにはいくつかのシステムデバッグ コンポーネントが組み込まれており、次のような目的に役立ちます。

- 低コストでのデバッグ
- トレースおよびプロファイリング
- ブレークポイント
- ウォッチポイント
- コードへのパッチ

システムデバッグ コンポーネントには、次のものが含まれます。

- ブレークポイントおよびコードへのパッチを実装するためのフラッシュパッチおよびブレークポイント (FPB) ユニット
- ウォッチポイント、トリガリソース、システムプロファイリングを実装するためのデータウォッチポイントおよびトレース (DWT) ユニット
- printf 形式のデバッグをサポートするアプリケーション駆動型トレースソース用の計装トレース マクロセル (ITM)
- 命令トレース用のエンベデッドトレース マクロセル (ETM)。プロセッサは、ETM を持つバージョンと持たないバージョンの両方でサポートされています。

すべてのデバッグコンポーネントは内部の専用ペリフェラルバス (PPB) 上に存在し、特権コードを使用してアクセスすることができます。

---

### 注

- コアデバッグの詳細については、第 10 章 「コアデバッグ」を参照して下さい。
-

## 11.2 システムデバッグ アクセス

デバッグ制御およびデータアクセスは、アドバンスドハイパフォーマンスバス アクセスポート (AHB-AP) インタフェースを経由して行われます。このインタフェースは、シリアルワイヤ デバッグポート (SW-DP) またはシリアルワイヤ JTAG デバッグポート (SWJ-DP) コンポーネントのいずれかによって行われます。SW-DP および SWJ-DP コンポーネントの情報については、P.12-1「デバッグポート」を参照して下さい。アクセスには次のものが含まれます。

- 内部 PPB。デバッガはこのバスを経由して、次に示すコンポーネントへアクセスできます。
  - ネスト型ベクタ割り込みコントローラ (NVIC)。プロセッサコアへのデバッグアクセスは、NVIC を経由して行われます。詳細については、第 10 章「コアデバッグ」を参照して下さい。
  - DWT ユニット
  - FPB ユニット
  - ITM
  - メモリ保護ユニット (MPU)

### ——— 注 ———

デバッガは、システムリセット中に PPB 空間内のすべてのレジスタを読み出すことができます。また、パワーオンリセットによってのみリセットされる、PPB 空間内のレジスタに書き込むこともできます。

- 外部専用ペリフェラルバス。デバッグでは、このバスを経由して、次のコンポーネントにアクセスできます。
  - ETM。命令トレースのみをサポートする低コストのトレースマクロセル。詳細については、第 15 章「エンベデッドトレース マクロセル」を参照して下さい。
  - トレースポート インタフェースユニット (TPIU)。このコンポーネントは、Cortex-M3 トレースデータ (ITM、および存在する場合は ETM からのデータ) とオフチップのトレースポート アナライザとの間のブリッジとして動作します。詳細については、第 13 章「トレースポート インタフェースユニット」を参照して下さい。
  - ROM テーブル
- DCode バス。デバッグでは、このバスを経由して、コード空間内に位置するメモリへアクセスできます。
- システムバス。システムバス空間内に位置するバス、メモリ、ペリフェラルへアクセスできます。

システムデバッグ アクセスの構造、および AHB-AP がシステムコンポーネントおよび外部バスのそれぞれにアクセスする方法を、図 11-1 に示します。

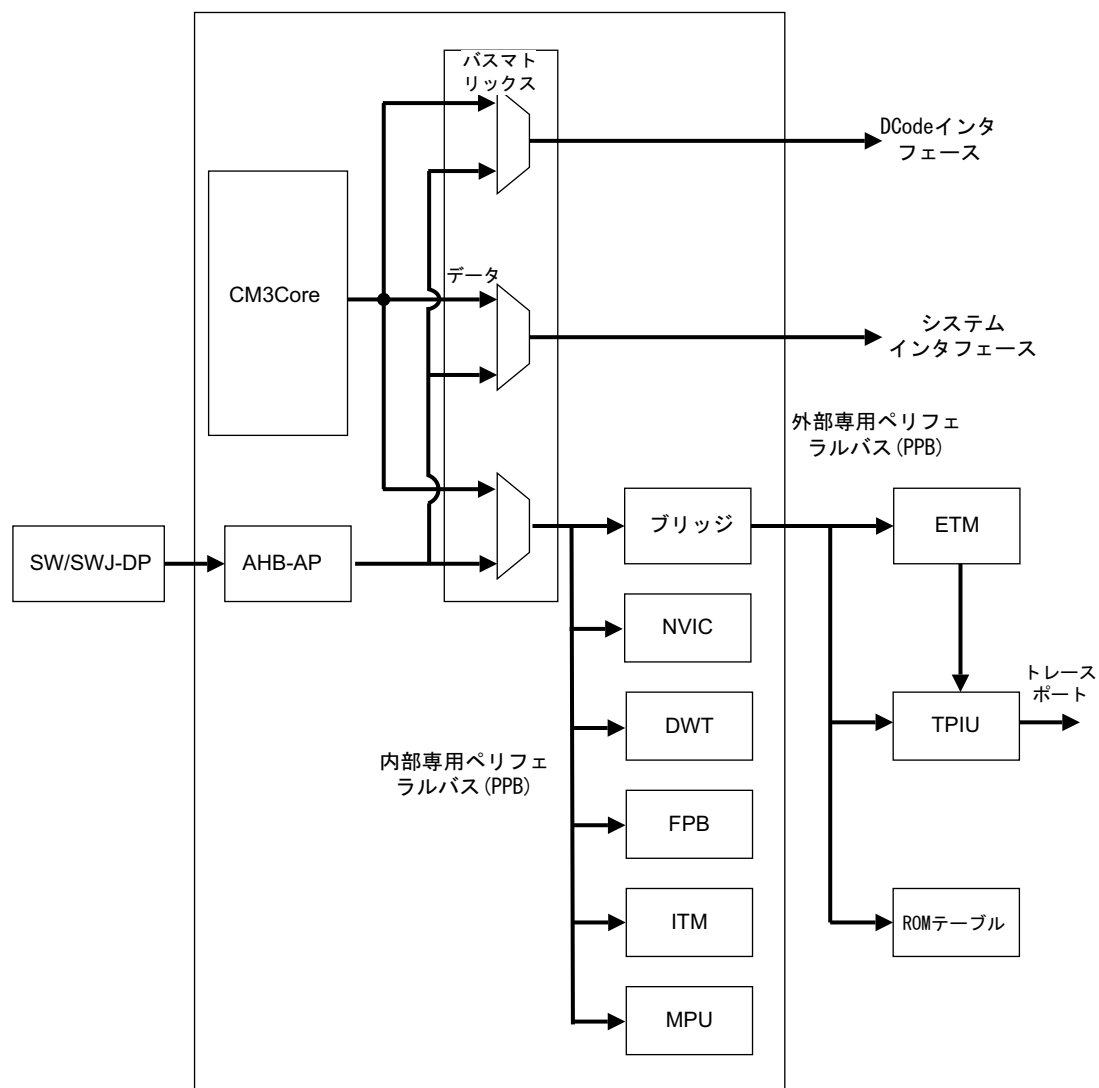


図 11-1 システムデバッグアクセスのブロック図



## 11.3 システムデバッグのプログラマモデル

このセクションでは、すべてのシステムデバッグ コンポーネントについて、デバッグレジスタを一覧表記し、説明を加えます。このセクションは以下の項目から構成されています。

- P.11-6 「*FPB*」
- P.11-13 「*DWT*」
- P.11-29 「*ITM*」
- P.11-38 「*AHB-AP*」

---

### 注

- コアデバッグ レジスタの詳細については、P.10-3 「コアデバッグ レジスタ」を参照して下さい。
  - SWJ-DP および SW-DP レジスタの詳細については、第 12 章 「デバッグポート」を参照して下さい。
  - TPIU の詳細については、第 13 章 「トレースポート インタフェースユニット」を参照して下さい。
-

## 11.4 FPB

### FPB

- ハードウェアブレークポイントを実装
- コード空間からシステム空間へ、コードおよびデータにパッチを適用

FPB ユニットには、次のものが含まれています。

- コード空間からのリテラルロードとの一致を検出し、システム空間の対応するエリアにリマッピングするための、2つのリテラルコンパレータ
- コード空間からの命令フェッチとの一致を検出し、システム空間の対応するエリアにリマッピングするための 6 つの命令コンパレータ。または、個別にコンパレータを構成して、一致が発生した場合はブレークポイント命令(BKPT)をプロセッサコアへ戻すようにすると、ハードウェアブレークポイントの機能を提供することができます。

FPB はグローバル許可を持ちますが、8 つのコンパレータに対して個別に許可することもできます。エントリの比較が一致すると、リマップレジスタに設定されているアドレスと、一致したコンパレータに対応するオフセットを加えたアドレスに、アドレスをリマッピングするか、または機能が許可されている場合は BKPT 命令にリマップします。比較は動作中に行われますが、比較の結果の発生が遅すぎるため、コード空間で発生する元の命令フェッチおよびリテラルロードは中止できません。ただし、プロセッサはこのトランザクションを無視し、リマップされたトランザクションのみが使用されます。

MPU が存在する場合は、リマップされたアドレスではなく元のアドレスに対して MPU ルックアップが実行されます。

#### ——— 注 ———

- アンアラインドリテラルアクセスはリマップされません。この場合は、DCode バスに対する元のアクセスが実行されます。
- 排他ロードは FPB に対しては予測不能です。アドレスはリマップされますが、排他ロードとしてのアクセスは実行されません。
- ビットバンドエイリアスへのリマッピングは、直接エイリアスアドレスにアクセスを行い、ビットバンド領域へのリマップは行いません。

### 11.4.1 FPB のプログラマモデル

フラッシュパッチ レジスタの一覧を、表 11-1 に示します。

**表 11-1 FPB レジスタの概要**

名前	タイプ	アドレス	説明
FP_CTRL	読み出し / 書き込み	0xE0002000	P.11-8 「フラッシュパッチ制御レジスタ」 参照
FP_REMAP	読み出し / 書き込み	0xE0002004	P.11-9 「フラッシュパッチリマップレジスタ」 参照
FP_COMP0	読み出し / 書き込み	0xE0002008	P.11-11 「フラッシュパッチ コンパレータレジスタ」 参照
FP_COMP1	読み出し / 書き込み	0xE000200C	P.11-11 「フラッシュパッチ コンパレータレジスタ」 参照
FP_COMP2	読み出し / 書き込み	0xE0002010	P.11-11 「フラッシュパッチ コンパレータレジスタ」 参照
FP_COMP3	読み出し / 書き込み	0xE0002014	P.11-11 「フラッシュパッチ コンパレータレジスタ」 参照
FP_COMP4	読み出し / 書き込み	0xE0002018	P.11-11 「フラッシュパッチ コンパレータレジスタ」 参照
FP_COMP5	読み出し / 書き込み	0xE000201C	P.11-11 「フラッシュパッチ コンパレータレジスタ」 参照
FP_COMP6	読み出し / 書き込み	0xE0002020	P.11-11 「フラッシュパッチ コンパレータレジスタ」 参照
FP_COMP7	読み出し / 書き込み	0xE0002024	P.11-11 「フラッシュパッチ コンパレータレジスタ」 参照
PID4	読み出し専用	0xE0002FD0	値 0x04
PID5	読み出し専用	0xE0002FD4	値 0x00
PID6	読み出し専用	0xE0002FD8	値 0x00
PID7	読み出し専用	0xE0002FDC	値 0x00
PID0	読み出し専用	0xE0002FE0	値 0x03
PID1	読み出し専用	0xE0002FE4	値 0xB0
PID2	読み出し専用	0xE0002FE8	値 0x0B
PID3	読み出し専用	0xE0002FEC	値 0x00
CID0	読み出し専用	0xE0002FF0	値 0x0D
CID1	読み出し専用	0xE0002FF4	値 0xE0
CID2	読み出し専用	0xE0002FF8	値 0x05
CID3	読み出し専用	0xE0002FFC	値 0xB1

フラッシュパッチ制御レジスタ

フラッシュパッチ制御レジスタは、フラッシュパッチ ブロックを許可するために使用します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE0002000

アクセス 読み出し / 書き込み

リセット時の状態 ビット [0] (ENABLE) が 1'b0 にリセットされます。

フラッシュパッチ制御レジスタのフィールドを、図 11-2 に示します。

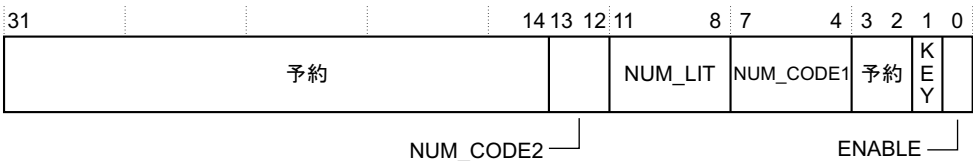


図 11-2 フラッシュパッチ制御レジスタのビット割り当て

フラッシュパッチ制御レジスタのフィールドの説明を、表 11-2 に示します。

表 11-2 フラッシュパッチ制御レジスタのビット割り当て

ビット	フィールド	機能
[31:15]	-	予約。0 として読み出され、書き込みは無視されます。
[14:12]	NUM_CODE2	コードコンパレータの全バンク数で、バンクごとに 16 個のコンパレータが含まれます。コードコンパレータが 16 個より少ない場合は、バンクカウントは 0 で、現在の数は NUM_CODE で示されます。この読み出し専用フィールドは 3'b000 で、Cortex-M3 プロセッサのバンクが 0 であることを示します。
[11:8]	NUM_LIT	リテラルスロット フィールドの数。この読み出し専用フィールドは b0010 で、2 つのリテラルスロットが存在することを示します。
[7:4]	NUM_CODE1	コードスロット フィールドの数。この読み出し専用フィールドは b0110 で、6 つのコードスロットが存在することを示します。

表 11-2 フラッシュパッチ制御レジスタのビット割り当て

ビット	フィールド	機能
[3:2]	-	予約
[1]	KEY	キーフィールド。フラッシュパッチ制御レジスタに書き込むには、この書き込み専用ビットに 1 を書き込む必要があります。
[0]	ENABLE	フラッシュパッチユニットのイネーブルビット 1 = フラッシュパッチ ユニットの許可 0 = フラッシュパッチ ユニットの禁止 ENABLE ビットはリセット時にクリアされます。
<p>——— 注 ———</p> <p>実装時に CM3Defs.v で TIEOFF_FPBEN の定義がコメント解除されている場合、ENABLE はセットできません。</p>		

### フラッシュパッチリマップレジスタ

フラッシュパッチリマップレジスタは、一致したアドレスがシステム空間のどの位置にリマップされるかを指定するために使用します。REMAP アドレスは 8 ワードアラインドで、8 つの FPB コンパレータのそれぞれに対して 1 ワードが割り当てられます。比較が一致すると、次の位置にリマップされます。

```
{3'b001, REMAP, COMP[2:0], HADDR[1:0]}
```

式で使用されている要素は以下のとおりです。

- 3'b001 により、リマップされたアクセスがシステム空間に固定されています。
- REMAP は 24 ビットで 8 ワードアラインドのリマップアドレスです

- COMP は一致しているコンパレータです。詳細については、表 11-3 を参照して下さい。

表 11-3 COMP マッピング

COMP[2:0]	コンパレータ	説明
000	FP_COMP0	命令コンパレータ
001	FP_COMP1	命令コンパレータ
010	FP_COMP2	命令コンパレータ
011	FP_COMP3	命令コンパレータ
100	FP_COMP4	命令コンパレータ
101	FP_COMP5	命令コンパレータ
110	FP_COMP6	リテラルコンパレータ
111	FP_COMP7	リテラルコンパレータ

- HADDR[1:0]** は、元のアドレスの 2 つの最下位ビット (LSB) です。**HADDR[1:0]** は、命令フェッチに対しては常時 2'b00 です。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE0002004

アクセス 読み出し / 書き込み

リセット時の状態 このレジスタはリセットされません。

フラッシュパッチリマップレジスタのフィールドを、図 11-3 に示します。

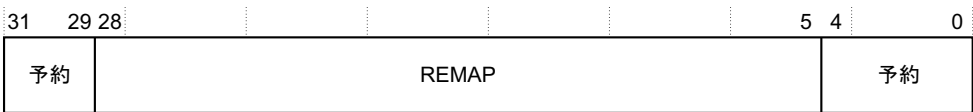


図 11-3 フラッシュパッチリマップレジスタのビット割り当て

フラッシュパッチリマップレジスタのフィールドの説明を、表 11-4 に示します。

**表 11-4 フラッシュパッチリマップレジスタのビット割り当て**

ビット	フィールド	機能
[31:29]	-	予約。読み出し値 b001。リマップをシステム空間へ接続します。
[28:5]	REMAP	ベースアドレス フィールドをリマップします。
[4:0]	-	予約。0 として読み出され、書き込みは無視されます。

## フラッシュパッチ コンパレータレジスタ

フラッシュパッチ コンパレータレジスタは、PC のアドレスと比較する値を格納するために使用されます。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アクセス      読み出し / 書き込み

アドレス 0xE0002008、0xE000200C、0xE0002010、0xE0002014、0xE0002018、0xE000201C、  
0xE0002020、0xE0002024

**リセット時の状態** ビット [0] (ENABLE) は 1'b0 にリセットされます。

フラッシュパッチ コンパレータレジスタのフィールドを、図 11-4 に示します。

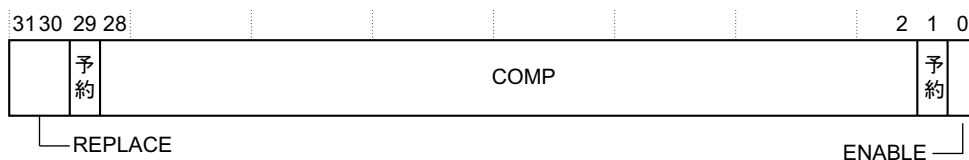


図 11-4 フラッシュパッチ コンパレータレジスタのビット割り当て

フラッシュパッチ コンパレータレジスタのフィールドの説明を、P.11-12 表 11-5 に示します。

表 11-5 フラッシュパッチ コンパレータレジスタのビット割り当て

ビット	フィールド	機能
[31:30]	REPLACE	COMP アドレスが一致した場合に発生させるイベントを選択します。 次のように解釈されます。 b00 = リマップアドレスにリマップします。FP_REMAP を参照して下さい。 b01 = BKPT を下位ハーフワードに設定します。上位ハーフワードには影響しません。 b10 = BKPT を上位ハーフワードに設定します。下位ハーフワードには影響しません。 b11 = BKPT を下位および上位ハーフワードの両方に設定します。 b00 以外の設定は、命令コンパレータに対してのみ有効です。リテラルコンパレータでは、b00 以外の設定は無視されます。 アドレスリマッピングは、b00 設定に対してのみ有効です。
[29]	-	予約
[28:2]	COMP	比較アドレス
[1]	-	予約
[0]	ENABLE	フラッシュパッチ コンパレータレジスタ <i>n</i> の比較およびリマップの許可 1 = フラッシュパッチ コンパレータレジスタ <i>n</i> の比較およびリマップを許可 0 = フラッシュパッチ コンパレータレジスタ <i>n</i> の比較およびリマップを禁止 FP_CTRL の ENABLE ビットも、比較を許可するようにセットする必要があります。 ENABLE ビットはリセット時にクリアされます。



## 11.5 DWT

DWT ユニットにより、次のデバッグ機能が実行されます。

- DWT ユニットには4つのコンパレータが組み込まれており、ハードウェアウォッチポイント、ETM トリガ、PC サンプライベントトリガ、またはデータアドレス サンプライベントトリガとして構成することができます。最初のコンパレータ DWT\_COMP0 は、クロックサイクルカウンタ CYCCNT に対して比較することもできます。2 番目のコンパレータ DWT\_COMP1 は、データコンパレータとしても使用できます。
- DWT には、次のものに対して使用するカウンタが組み込まれています。
  - クロックサイクル (CYCCNT)
  - フォールドされた命令
  - ロード/ストアユニット (LSU)(Load Store Unit) 操作
  - スリープサイクル
  - CPI (最初のサイクルを除くすべての命令サイクル)
  - 割り込みオーバーヘッド

### —— 注 ——

イベントは、カウンタがオーバーフローするたびに送信されます。

- DWT を構成すると、定義された間隔で PC サンプルおよび割り込みイベント情報を送信することができます。

### 11.5.1 DWT レジスタの概要と説明

DWT レジスタの一覧を、表 11-6 に示します。

表 11-6 DWT レジスタの概要

名前	タイプ	アドレス	リセット時の値	説明
DWT_CTRL	読み出し / 書き込み	0xE0001000	0x00000000	P.11-15 「DWT 制御レジスタ」 参照
DWT_CYCCNT	読み出し / 書き込み	0xE0001004	0x00000000	P.11-18 「DWT カレント PC サンプライベントトリガ サンプルカウンタレジスタ」 参照
DWT_CPICNT	読み出し / 書き込み	0xE0001008	-	P.11-19 「DWT CPI カウントレジスタ」 参照
DWT_EXCCNT	読み出し / 書き込み	0xE000100C	-	P.11-20 「DWT 例外オーバーヘッドカウントレジスタ」 参照
DWT_SLEEPCNT	読み出し / 書き込み	0xE0001010	-	P.11-21 「DWT スリープカウント レジスタ」 参照
DWT_LSUCNT	読み出し / 書き込み	0xE0001014	-	P.11-21 「DWT LSU カウントレジスタ」 参照

表 11-6 DWT レジスタの概要

名前	タイプ	アドレス	リセット時の値	説明
DWT_FOLDCNT	読み出し / 書き込み	0xE0001018	-	P.11-22 「DWT フォールドカウンタ レジスタ」 参照
DWT_PCSR	読み出し専用	0xE000101C	-	P.11-23 「DWT プログラムカウンタ サンプルレジスタ」 参照
DWT_COMP0	読み出し / 書き込み	0xE0001020	-	P.11-23 「DWT コンパレータレジスタ」 参照
DWT_MASK0	読み出し / 書き込み	0xE0001024	-	P.11-24 「DWT マスクレジスタ 0 ～ 3」 参照
DWT_FUNCTION0	読み出し / 書き込み	0xE0001028	0x00000000	P.11-25 「DWT 機能レジスタ 0 ～ 3」 参照
DWT_COMP1	読み出し / 書き込み	0xE0001030	-	P.11-23 「DWT コンパレータレジスタ」 参照
DWT_MASK1	読み出し / 書き込み	0xE0001034	-	P.11-24 「DWT マスクレジスタ 0 ～ 3」 参照
DWT_FUNCTION1	読み出し / 書き込み	0xE0001038	0x00000000	P.11-25 「DWT 機能レジスタ 0 ～ 3」 参照
DWT_COMP2	読み出し / 書き込み	0xE0001040	-	P.11-23 「DWT コンパレータレジスタ」 参照
DWT_MASK2	読み出し / 書き込み	0xE0001044	-	P.11-24 「DWT マスクレジスタ 0 ～ 3」 参照
DWT_FUNCTION2	読み出し / 書き込み	0xE0001048	0x00000000	P.11-25 「DWT 機能レジスタ 0 ～ 3」 参照
DWT_COMP3	読み出し / 書き込み	0xE0001050	-	P.11-23 「DWT コンパレータレジスタ」 参照
DWT_MASK3	読み出し / 書き込み	0xE0001054	-	P.11-24 「DWT マスクレジスタ 0 ～ 3」 参照
DWT_FUNCTION3	読み出し / 書き込み	0xE0001058	0x00000000	P.11-25 「DWT 機能レジスタ 0 ～ 3」 参照
PID4	読み出し専用	0xE0001FD0	0x04	値 0x04
PID5	読み出し専用	0xE0001FD4	0x00	値 0x00
PID6	読み出し専用	0xE0001FD8	0x00	値 0x00
PID7	読み出し専用	0xE0001FDC	0x00	値 0x00
PID0	読み出し専用	0xE0001FE0	0x02	値 0x02
PID1	読み出し専用	0xE0001FE4	0xB0	値 0xB0
PID2	読み出し専用	0xE0001FE8	0x1B	値 0x1B
PID3	読み出し専用	0xE0001FEC	0x00	値 0x00

表 11-6 DWT レジスタの概要

名前	タイプ	アドレス	リセット時の値	説明
CID0	読み出し専用	0xE0001FF0	0x0D	値 0x0D
CID1	読み出し専用	0xE0001FF4	0xE0	値 0xE0
CID2	読み出し専用	0xE0001FF8	0x05	値 0x05
CID3	読み出し専用	0xE0001FFC	0xB1	値 0xB1

DWT 制御レジスタ

DWT 制御レジスタは、DWT ユニットを許可するために使用します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE0001000

アクセス 読み出し / 書き込み

リセット時の状態 0x40000000

DWT 制御レジスタのフィールドを、図 11-5 に示します。

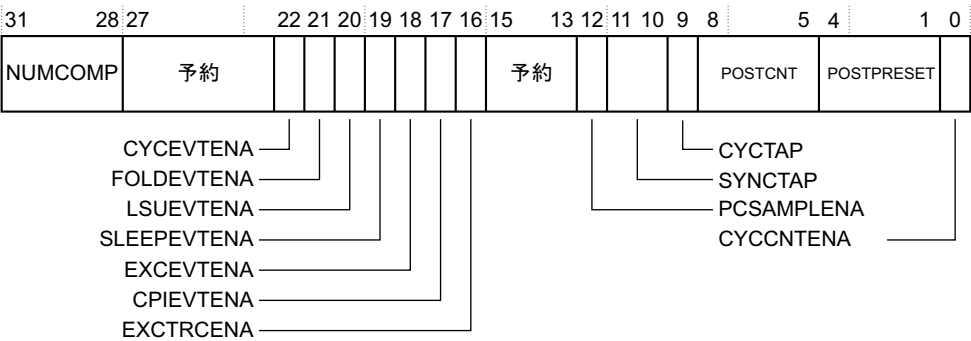


図 11-5 DWT 制御レジスタのビット割り当て

DWT 制御レジスタのフィールドの説明を、表 11-7 に示します。

**表 11-7 DWT 制御レジスタのビット割り当て**

ビット	フィールド	機能
[31:28]	NUMCOMP	コンパレータ数のフィールド。この読み出し専用フィールドは b0100 で、4 つのコンパレータを示します。
[27:23]	-	予約
[22]	CYCEVTEN	<p>サイクルカウント イベントを許可します。POSTCNT カウンタによってトリガされると、イベントを送信します。詳細については、CYCTAP（ビット [9]）および POSTPRESET、ビット [4:1] を参照して下さい。</p> <p>1 = サイクルカウント イベントを許可 0 = サイクルカウント イベントを禁止</p> <p>このイベントは、PCSAMPLENA、ビット [12] が禁止されている場合にのみ送信されます。</p> <p>PCSAMPLENA は、このビットの設定に優先します。</p> <p>CYCEVTENA ビットは、リセット時にクリアされます。</p>
[21]	FOLDEVTENA	<p>フォールドされた命令カウント イベントを許可します。DWT_FOLDCNT がオーバフローすると、イベントを送信します（フォールドされた命令の 256 サイクルごと）。フォールドされた命令とは、実行に 1 サイクルも掛からなかった命令です。例えば、IT 命令はフォールドされるため 1 サイクルも使用されません。</p> <p>1 = フォールドされた命令のカウント イベントを許可 0 = フォールドされた命令のカウント イベントを禁止</p> <p>FOLDEVTENA ビットは、リセット時にクリアされます。</p>
[20]	LSUEVTENA	<p>LSU カウント イベントを許可します。DWT_LSUCNT がオーバフローすると、イベントを送信します（LSU 演算の 256 サイクルごと）。LSU カウントには、命令の初期サイクル後にかかるすべての LSU サイクルをカウントします。</p> <p>1 = LSU カウント イベントを許可 0 = LSU カウント イベントを禁止</p> <p>LSUEVTENA ビットは、リセット時にクリアされます。</p>
[19]	SLEEPEVTENA	<p>スリープカウント イベントを許可します。DWT_SLEEP CNT がオーバフローすると、イベントを送信します（プロセッサがスリープ中の 256 サイクルごと）。</p> <p>1 = スリープカウント イベントを許可 0 = スリープカウント イベントを禁止</p> <p>SLEEPEVTENA ビットは、リセット時にクリアされます。</p>
[18]	EXCEVTENA	<p>割り込みオーバヘッド イベントを許可します。DWT_EXCCNT がオーバフローすると、イベントを送信します（割り込みオーバヘッドの 256 サイクルごと）。</p> <p>1 = 割り込みオーバヘッド イベントを許可 0 = 割り込みオーバヘッド イベントを禁止</p> <p>EXCEVTENA ビットは、リセット時にクリアされます。</p>

表 11-7 DWT 制御レジスタのビット割り当て

ビット	フィールド	機能
[17]	CPIEVTENA	CPI カウントイベントを許可します。DWT_CPICNT がオーバフローすると、イベントを送信します（マルチサイクル命令の 256 サイクルごと）。 1 = CPI カウントイベントを許可 0 = CPI カウントイベントを禁止 CPIEVENTA ビットは、リセット時にクリアされます。
[16]	EXCTRCENA	割り込みイベントトレーシングを許可します。 1 = 割り込みイベントトレースを許可 0 = 割り込みイベントトレースを禁止 EXCEVENTA ビットは、リセット時にクリアされます。
[15:13]	-	予約
[12]	PCSAMPLEENA	PC サンプリングイベントを許可します。POSTCNT カウンタによってトリガされると、PC サンプルイベントが送信されます。詳細については、CYCTAP（ビット [9]）および POSTPRESET、ビット [4:1] を参照して下さい。このビットによる許可は、CYCEVENTA（ビット [20]）より優先されます。 1 = PC サンプリングイベントを許可 0 = PC サンプリングイベントを禁止 PCSAMPLEENA ビットは、リセット時にクリアされます。
[11:10]	SYNCTAP	ITM の SYNCENA 制御に同期化パルスを供給します。ここで選択された値により、DWT_CYCCNT レジスタ上でタップを選択して周期（約 1/ 秒またはそれより小さい）が決定されます。同期化（ハートビートおよび動的接続同期化）を使用するには、CYCCNTENA を 1 に、SYNCTAP をその値のいずれかに、SYNCENA を 1 に設定する必要があります。 0b00 = 禁止。同期化カウントなし 0b01 = CYCCNT ビット 24 でタップ 0b10 = CYCCNT ビット 26 でタップ 0b11 = CYCCNT ビット 28 でタップ
[9]	CYCTAP	DWT_CYCCNT レジスタのタップを選択します。これらは、次のようにビット [6] とビット [10] の間隔が空いています。 CYCTAP = 0 の場合、ビット [6] がタップとして選択 CYCTAP = 1 の場合、ビット [10] がタップとして選択 CYCCNT レジスタで選択したビットが、0 から 1 または 1 から 0 に変更されると、POSTCNT のビット [8:5]、ポストスカラカウンタへ送信され、そのカウンタはカウントダウンされます。ポストスカラが 0 の場合にビットが変更されると、PC サンプリングまたは CYCEVTCNT のイベントがトリガされます。

表 11-7 DWT 制御レジスタのビット割り当て

ビット	フィールド	機能
[8:5]	POSTCNT	CYCTAP のポストスカラカウンタ。 選択したタップビットが 0 から 1 または 1 から 0 に変更されると、ポストスカラカウンタは、0 でなければカウントダウンされます。 0 の場合は、PCSAMPLENA または CYCEVTENA を使用するためのイベントがトリガされます。同時に、POSTPRESET（ビット [4:1]）の値がこのカウンタにリロードされます。
[4:1]	POSTPRESET	ビット [8:5] の POSTCNT（ポストスカラカウンタ）のリロードに使用される値。この値が 0 の場合、イベントは各タップ変更の際にトリガされます（2 のべき乗、例えば $1 << 6$ または $1 << 10$ など）。 このフィールドが 0 以外の値の場合、その値がカウントダウン値となり、0 になるたびに POSTCNT にリロードされます。例えば、このレジスタの値が 1 の場合は、タップが 2 回変更されるごとにイベントが生成されます。
[0]	CYCCNTENA	CYCCNT カウンタを許可します。許可されていない場合は、カウンタはカウントされず、PS サンプリングまたは CYCCNTENA のイベントは生成されません。通常に使用する場合は、デバッガにより CYCCNT カウンタを 0 に初期化する必要があります。

——— 注 ———

デバッグ例外およびモニタ制御レジスタの TRCENA ビットは、DWT を使用する前にセットする必要があります。P.10-8「デバッグ例外およびモニタ制御レジスタ」を参照して下さい。

——— 注 ———

DWT は、ITM とは独立に許可されます。DWT を許可してイベントを送信する場合は、ITM も許可する必要があります。

DWT カレント PC サンプラサイクル カウントレジスタ

DWT カレント PC サンプラサイクル カウントレジスタは、コアサイクルの数をカウントするために使用します。これにより、経過した実行時間が測定できます。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス                    0xE0001004  
アクセス                    読み出し専用  
リセット時の状態        0x00000000

DWT カレント PC サンプラサイクル カウントレジスタのフィールドの説明を、表 11-8 に示します。

**表 11-8 DWT カレント PC サンプラサイクル カウントレジスタのビット割り当て**

ビット	フィールド	機能
[31:0]	CYCCNT	<p>カレント PC サンプラサイクル カウンタのカウンタ値。許可されている場合は、このカウンタにより、コアが停止しているときを除くコアサイクルの数がカウントされます。</p> <p>CYCCNT は、加算式のフリーランニング カウンタです。オーバフローすると 0 にラップアラウンドします。</p> <p>最初に許可されるときに、デバッグにより 0 に初期化する必要があります。</p>

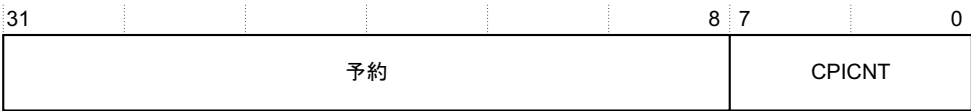
- これはフリーランカウンタです。このカウンタには、次の 3 つの機能があります。
- PCSAMPLENA がセットされると、選択したタップビットの値が（0 から 1 または 1 から 0 へ）変更され、いずれかのポストスカラ値のカウントが 0 になったときに PC がサンプリングされて送信されます。
  - CYCEVTENA がセットされる（および PCSAMPLENA がクリアされる）と、選択したタップビットの値が（0 から 1 または 1 から 0 へ）変更され、いずれかのポストスカラ値のカウントが 0 になったときにイベントが送信されます。
  - アプリケーションおよびデバッグは、カウンタを使用して経過した実行時間を測定することができます。開始時刻と終了時刻の差を計算することにより、アプリケーションはコア内クロック間の時間を測定することができます（デバッグでホールド中を除く）。これは  $2^{32}$  コアクロック サイクルまで有効です（例えば、50MHz において約 86 秒）。

### DWT CPI カウントレジスタ

DWT CPI カウントレジスタは、最初のサイクル以後の命令サイクルの総数をカウントするために使用します。

- このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。
- アドレス                      0xE0001008
- アクセス                      読み出し / 書き込み
- リセット時の状態              -

DWT CPI カウントレジスタのフィールドを、図 11-6 に示します。



**図 11-6 DWT CPI カウントレジスタのビット割り当て**

DWT CPI カウントレジスタのフィールドの説明を、表 11-9 に示します。

表 11-9 DWT CPI カウントレジスタのビット割り当て

ビット	フィールド	機能
[31:8]	-	予約
[7:0]	CPICNT	現在の CPI カウンタ値。DWT_LSUCNT によって記録される命令以外のすべての命令を実行するのに必要な追加サイクル（最初のサイクルはカウントされません）でインクリメントされます。このカウンタは、すべての命令フェッチストールでもインクリメントされます。 CPIEVTENA がセットされていると、カウンタがオーバフローしたときにイベントが送信されます。 許可すると 0 にクリアされます。

DWT 例外オーバヘッドカウンタ レジスタ

DWT 例外オーバヘッドカウンタ レジスタは、割り込み処理中に消費される合計サイクル数をカウントするために使用します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス                    0xE000100C  
アクセス                    読み出し / 書き込み  
リセット時の状態        -

DWT 例外オーバヘッドカウンタ レジスタのフィールドを、図 11-7 に示します。



図 11-7 DWT 例外オーバヘッドカウンタ レジスタのビット割り当て

DWT 例外オーバヘッドカウンタ レジスタのフィールドの説明を、表 11-10 に示します。

表 11-10 DWT 例外オーバヘッドカウンタ レジスタのビット割り当て

ビット	フィールド	機能
[31:8]	-	予約
[7:0]	EXCCNT	現在の割り込みオーバヘッドのカウンタ値。割り込み処理中に消費される合計サイクル数をカウントします（エントリスタック、復帰アンスタック、横取りなど）。カウンタがオーバフローすると、イベントが送信されます（256 サイクルごと）。許可されると、このカウンタは 0 に初期化されます。 許可すると 0 にクリアされます。





DWT LSU カウントレジスタのフィールドを、図 11-9 に示します。

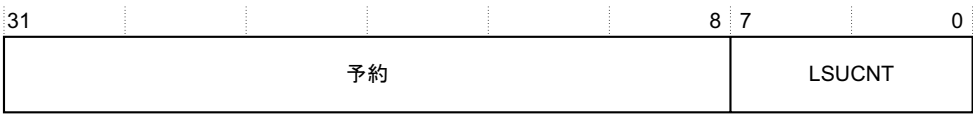


図 11-9 DWT LSU カウントレジスタのビット割り当て

DWT LSU カウントレジスタのフィールドの説明を、表 11-12 に示します。

表 11-12 DWT LSU カウントレジスタのビット割り当て

ビット	フィールド	機能
[31:8]	-	予約
[7:0]	LSUCNT	LSU カウンタ。プロセッサが LSU 操作を処理する合計サイクル数をカウントします。命令の初期実行コストはカウントされません。 例えば、完了するのに 2 サイクルを要する LDR では、このカウンタが 1 サイクル分インクリメントされます。同様に、2 サイクル分ストールする LDR（つまり、4 サイクル分）では、このカウンタが 3 サイクル分インクリメントされます。カウンタがオーバフローすると、イベントが送信されます（256 サイクルごと）。 許可すると 0 にクリアされます。

DWT フォールドカウント レジスタ

DWT フォールドカウント レジスタは、フォールドされた命令の総数をカウントするために使用します。0 サイクルを要する各命令は、1 としてカウントされます。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス                    0xE0001018  
アクセス                    読み出し / 書き込み  
リセット時の状態        -

DWT フォールドカウント レジスタのフィールドを、図 11-10 に示します。

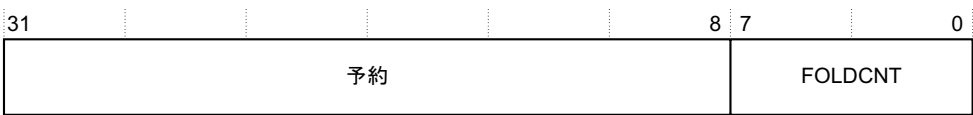


図 11-10 DWT フォールドカウント レジスタのビット割り当て

DWT フォールドカウント レジスタのフィールドの説明を、表 11-13 に示します。

**表 11-13 DWT フォールドカウント レジスタのビット割り当て**

ビット	フィールド	機能
[31:8]	-	予約
[7:0]	FOLDCNT	フォールドされた命令の総数をカウントします。許可されると、このカウンタは 0 に初期化されます。

### DWT プログラムカウンタ サンプルレジスタ

DWT プログラムカウンタ サンプルレジスタ (PCSR) は、現在実行中のコードを変更せずに、デバッグエージェントを使用して粒度の粗いソフトウェアプロファイリングを行うために使用します。

コアがデバッグ状態でない場合は、返される値は直前に実行された命令の命令アドレスです。

コアがデバッグ状態の場合は、返される値は 0xFFFFFFFF です。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE000101C

アクセス 読み出し専用

リセット時の状態 予測不能

DWT PCSR のフィールドの説明を、表 11-14 に示します。

**表 11-14 DWT プログラムカウンタ サンプルレジスタのビット割り当て**

ビット	フィールド	機能
[31:0]	EIASAMPLE	実行命令アドレスサンプル、またはコアが停止している場合は 0xFFFFFFFF。

### DWT コンパレータレジスタ

DWT コンパレータレジスタ 0 ～ 3 は、ウォッチポイントイベントをトリガする値を書き込むために使用します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE0001020、0xE0001030、0xE0001040、0xE0001050

アクセス 読み出し / 書き込み

リセット時の状態 -

DWT コンパレータレジスタ 0 ～ 3 のフィールドの説明を、表 11-15 に示します。

表 11-15 DWT コンパレータレジスタ 0 ～ 3 のビット割り当て

フィールド	名前	定義
[31:0]	COMP	DWT_FUNCTIONx によって指定されるデータアドレス、および PC に対して比較するためのデータ値。 DWT_COMP0 は、PC サンプラカウンタ (DWT_CYCCNT) の値に対しても比較できます。 DWT_COMP1 は、データマッチングが実行されるためのデータ値 (DATAVMATCH) に対しても比較できます。

DWT マスクレジスタ 0 ～ 3

DWT マスクレジスタ 0 ～ 3 は、COMP との比較の際に、マスクをデータアドレスに適用するために使用します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE0001024、0xE0001034、0xE0001044、0xE0001054

アクセス 読み出し / 書き込み

リセット時の状態 -

DWT マスクレジスタ 0 ～ 3 のフィールドを、図 11-11 に示します。

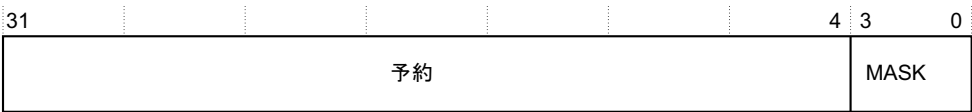


図 11-11 DWT マスクレジスタ 0 ～ 3 のビット割り当て

DWT マスクレジスタ 0 ～ 3 のフィールドの説明を、表 11-16 に示します。

表 11-16 DWT マスクレジスタ 0 ～ 3 のビット割り当て

ビット	フィールド	機能
[31:4]	-	予約
[3:0]	MASK	COMP との比較時のデータアドレスのマスク。これは、無視マスクのサイズです。そのため、使用するアドレスに対して、 $\sim 0 \ll \text{MASK}$ によりマスクが形成されます。つまり、DWT マッチングは $(\text{ADDR} \& (\sim 0 \ll \text{MASK})) = \text{COMP}$ として実行されます。 ただし、バス上のどこに出現してもアドレスの比較を可能にするため、実際の比較はもう少し複雑です。そのため、COMP が 3 の場合、0 のワード内であるため、0 のワードアクセスでも一致します。

DWT 機能レジスタ 0 ～ 3

DWT 機能レジスタ 0 ～ 3 は、コンパレータの動作を制御するために使用します。各コンパレータは、次の操作を実行できます。

- PC またはデータアドレスのいずれかに対する比較。これは CYCMATCH によって制御されます。この機能はコンパレータ 0 (DWT\_COMP0) でのみ使用可能です。
- 関連付けられたアドレスコンパレータでアドレスが一致した場合、データの値の比較を実行します。この機能はコンパレータ 1 (DWT\_COMP1) でのみ使用可能です。
- データまたは PC カップルの送信、ETM のトリガ、あるいは FUNCTION で定義された操作に依存するウォッチポイントの生成。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE0001028、0xE0001038、0xE0001048、0xE0001058  
アクセス 読み出し / 書き込み  
リセット時の状態 0x00000000

DWT 機能レジスタ 0 ～ 3 のフィールドを、図 11-12 に示します。

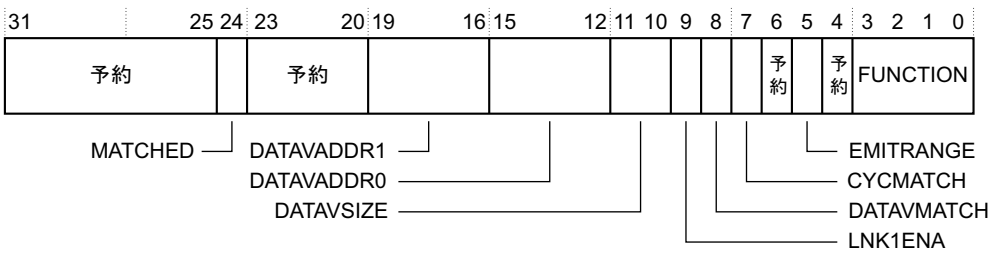


図 11-12 DWT 機能レジスタ 0 ～ 3 のビット割り当て

DWT 機能レジスタ 0 ～ 3 のフィールドの説明を、表 11-17 に示します。

表 11-17 DWT 機能レジスタ 0 ～ 3 のビット機能

ビット	フィールド	機能
[31:25]	-	予約
[24]	MATCHED	このビットはコンパレータが一致した場合にセットされ、このビットが最後に読み出された後に、FUNCTION によって定義された操作が発生したことを示します。このビットは読み出し時にクリアされます。
[23:20]	-	予約
[19:16]	DATAVADDR1	DATAVMATCH == 1 および LNK1ENA == 1 の場合に、データ値の一致に対する 2 番目のリンクされるアドレス コンパレータの ID を示します。

表 11-17 DWT 機能レジスタ 0 ～ 3 のビット機能

ビット	フィールド	機能
[15:12]	DATAVADDR0	DATAVMATCH == 1 の場合に、データ値の一致に対するリンクされるアドレス コンパレータの ID を示します。
[11:10]	DATAVSIZE	一致する COMP レジスタのデータサイズを定義します。 00 = バイト 01 = ハーフワード 10 = ワード 11 = 予測不能
[9]	LNK1ENA	読み出し専用で、次の意味を持ちます。 0 = DATAVADDR1 はサポートされていません。 1 = DATAVADDR1 はサポート（許可）されています。
[8]	DATAVMATCH	このビットは、コンパレータ 1 でのみ使用可能です。DATAVMATCH がセットされている場合は、このコンパレータでデータ値の比較が実行されます。DATAVADDR0 および DATAVADDR1 によって指定されたコンパレータにより、データ比較のアドレスが提供されます。DWT_FUNCTION1 で DATAVMATCH がセットされている場合は、DATAVADDR0 および DATAVADDR1 によって指定されたコンパレータに対する FUNCTION 設定は上書きされ、これらのコンパレータはデータ比較用のアドレス一致のみを提供します。
[7]	CYCMATCH	コンパレータ 0 でのみ使用可能です。セットされている場合、このコンパレータはクロックサイクルカウンタに対して比較されます。
[6]	-	予約
[5]	EMITRANGE	送信範囲フィールド。範囲一致が発生した場合にオフセットの送信を許可するように予約されています。EMITRANGE ビットは、リセット時にクリアされます。EMITRANGE が許可されている場合、PC サンプリングはサポートされません。 EMITRANGE は次の場合にのみ適用されます。 FUNCTION = b0001、b0010、および b0011
[4]	-	予約
[3:0]	FUNCTION	FUNCTION 設定については、P.11-27 表 11-18 を参照して下さい。

DWT 機能レジスタの機能設定の説明を、表 11-18 に示します。

**表 11-18 DWT 機能レジスタの設定**

値	機能
b0000	禁止
b0001	EMITRANGE = 0、ITM を通して PC をサンプリングし送信します。 EMITRANGE = 1、ITM を通してアドレスオフセットを送信します。
b0010	EMITRANGE = 0、読み出し / 書き込み時に ITM を通してデータを送信します。 EMITRANGE = 1、読み出し / 書き込み時に ITM を通してデータおよびアドレスオフセットを送信します。
b0011	EMITRANGE = 0、読み出し / 書き込み時に ITM を通して PC およびデータ値をサンプリングします。 EMITRANGE = 1、読み出し / 書き込み時に ITM を通してアドレスオフセットおよびデータ値を送信します。
b0100	PC 一致時のウォッチポイント
b0101	読み出し時のウォッチポイント
b0110	書き込み時のウォッチポイント
b0111	読み出し / 書き込み時のウォッチポイント
b1000	PC 一致時の ETM トリガ
b1001	読み出し時の ETM トリガ
b1010	書き込み時の ETM トリガ
b1011	読み出し / 書き込み時の ETM トリガ
b1100	予約
b1101	予約
b1110	予約
b1111	予約

#### 注

- ETM が組み込まれていない場合、ETM トリガは使用できません。
- データ値は、フォールト（MPU またはバスフォールト）が発生しないアクセスについてのみサンプリングされます。PC は、どのフォールトかには関係なくサンプリングされます。PC はバーストの最初のアドレスについてのみサンプリングされます。

- DWT\_FUNCTION1 でも DATAVMATCH がセットされている場合、DWT\_FUNCTION1 の DATAVADDR0 および DATAVADDR1 によって指定されたコンパレータについて FUNCTION が上書きされます。DATAVADDR0 および DATAVADDR1 によって指定されたコンパレータは、コンパレータ 1 のデータ一致に対してのみアドレスコンパレータ一致を実行できます。
  - 実装時に CM3Defs.v で TIEOFF\_DMATCH の定義がコメント解除されている場合、DWT\_FUNCTION1 の DATAVADDR0、DATAVADDR1、または DATAVMATCH をセットすることはできません。これは、その実装ではデータ一致機能を使用できないことを意味します。データ一致が使用できるかどうかをテストするには、DWT\_FUNCTION1 の DATAVMATCH ビットを書き込んでから読み出します。このビットをセットできない場合は、データ一致は使用できません。
  - ウォッチポイントは命令の後で停止するため、ウォッチポイントでの PC 一致は推奨されません。これは、主に ETM をガードしてトリガします。
-



## 11.6 ITM

ITM は `printf` 形式のデバッグをサポートするアプリケーション駆動型トレースソースで、オペレーティングシステム (OS) およびアプリケーションのイベントをトレースして、システム診断情報を送信します。ITM はトレース情報をパケットとして送信します。パケットを生成可能なソースは 3 つあります。複数のソースが同時にパケットを生成した場合は、ITM によってパケットの出力順序が調整されます。ソースは、優先順位の高いものから順に、次の 3 つです。

- ソフトウェアトレース。ソフトウェアは ITM スティムラスレジスタに直接書き込むことができます。これによりパケットが送信されます。
- ハードウェアトレース。これらのパケットは DWT により生成され、ITM によって送信されます。
- タイムスタンプ。タイムスタンプはパケットに関連して送信されます。ITM にはタイムスタンプを生成する 21 ビットカウンタが組み込まれています。Cortex-M3 クロックまたはシリアルワイヤ ビューア (SWV) 出力のビットクロックレートでカウンタを駆動します。

### 11.6.1 ITM レジスタの概要と説明

——— 注 ———

デバッグ例外およびモニタ制御レジスタの TRCENA は、ITM をプログラムするかまたは使用する前に許可する必要があります。P.10-8「デバッグ例外およびモニタ制御レジスタ」を参照して下さい。

ITM レジスタの一覧を、表 11-19 に示します。

表 11-19 ITM レジスタの概要

名前	タイプ	アドレス	リセット時の値	説明
スティムラスポート 0 ~ 31	読み出し / 書き込み	0xE0000000-0xE000007C	-	P.11-30「ITM スティムラスポート 0 ~ 31」参照
トレースイネーブル	読み出し / 書き込み	0xE0000E00	0x00000000	P.11-31「ITM トレースイネーブル レジスタ」参照
トレース特権	読み出し / 書き込み	0xE0000E40	0x00000000	P.11-32「ITM トレース特権レジスタ」参照
トレース制御レジスタ	読み出し / 書き込み	0xE0000E80	0x00000000	P.11-33「ITM トレース制御レジスタ」参照
統合書き込み	書き込み専用	0xE0000EF8	0x00000000	P.11-34「ITM 統合書き込みレジスタ」参照

表 11-19 ITM レジスタの概要

名前	タイプ	アドレス	リセット時の値	説明
統合読み出し	読み出し専用	0xE0000EFC	0x00000000	P.11-35 「ITM 統合読み出しレジスタ」 参照
統合モード制御	読み出し / 書き込み	0xE0000F00	0x00000000	P.11-35 「ITM 統合モード制御レジスタ」 参照
ロックアクセスレジスタ	書き込み専用	0xE0000FB0	0x00000000	P.11-36 「ITM ロックアクセス レジスタ」 参照
ロックステータス レジスタ	読み出し専用	0xE0000FB4	0x00000003	P.11-36 「ITM ロックステータス レジスタ」 参照
PID4	読み出し専用	0xE0000FD0	0x00000004	値 0x04
PID5	読み出し専用	0xE0000FD4	0x00000000	値 0x00
PID6	読み出し専用	0xE0000FD8	0x00000000	値 0x00
PID7	読み出し専用	0xE0000FDC	0x00000000	値 0x00
PID0	読み出し専用	0xE0000FE0	0x00000001	値 0x01
PID1	読み出し専用	0xE0000FE4	0x000000B0	値 0xB0
PID2	読み出し専用	0xE0000FE8	0x0000001B	値 0x1B
PID3	読み出し専用	0xE0000FEC	0x00000000	値 0x00
CID0	読み出し専用	0xE0000FF0	0x0000000D	値 0x0D
CID1	読み出し専用	0xE0000FF4	0x000000E0	値 0xE0
CID2	読み出し専用	0xE0000FF8	0x00000005	値 0x05
CID3	読み出し専用	0xE0000FFC	0x000000B1	値 0xB1

—— 注 ——

ITM レジスタは特権モードでは完全にアクセス可能です。ユーザモードではすべてのレジスタは読み出せますが、書き込みはスティムラスレジスタおよびトレースイネーブル レジスタに対してのみ行えます。また、対応するトレース特権レジスタのビットがセットされている場合にのみ書き込みが可能です。ITM レジスタへの無効なユーザモード書き込みは破棄されます。

ITM スティムラサポート 0 ～ 31

32 個のスティムラサポートには、それぞれ固有のアドレスがあります。トレースイネー

ブル レジスタの対応するビットがセットされている場合は、これらの位置のいずれかに書き込むと、データが FIFO に書き込まれます。スティムラポートのいずれかを読み出すと、ビット [0] に次の FIFO ステータスが返されます。

- 0 = フル。
- 1 = フルではない。

ポーリングされる FIFO インタフェースでは不可分な読み出し - 変更 - 書き込みは提供されないため、ポーリングされる printf が、割り込みまたは他のスレッドによる ITM の使用と同時に使われている場合は、Cortex-M3 の排他モニタを使用する必要があります。以下のポーリングされるコードは、ITM へのポーリングによるアクセスによって、スティムラが消失しないことを保証します。

```

; r0 = Value to write to port

; r1 and r2 = Temporary scratch registers

        MOV r1, #0xE0000000      ; r1 = Stimulus port base
Retry    LDREX r2, [r1, #Port*4]  ; Load FIFO full status

        CMP r2, #0                ; Compare with full

        ITT NE                    ; If (not full)

        STREXNE r2, [r1, #Port*4]; Try sending value to port

        CMPNE r2, #1              ; and check for failure

        BEQ Retry                 ; If full or failed then retry

```

### ITM トレースイネーブル レジスタ

トレースイネーブル レジスタは、対応するスティムポートに書き込みを行い、トレースデータを生成するために使用します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アクセス	読み出し / 書き込み
アドレス	0xE0000E00
リセット時の状態	0x00000000

ITM トレースイネーブルレジスタのフィールドを、表 11-20 に示します。

表 11-20 ITM トレースイネーブルレジスタのビット割り当て

ビット	フィールド	機能
[31:0]	STIMENA	ITM スティムラポートでのトレーシングを許可するビットマスク。スティムラポートごとに 1 つのビットが使用されます。

### —— 注 ———

ITMENA がセットされている場合は、このレジスタへの特権書き込みが受け付けられ

まず、ITMENA がセットされ、適切な特権マスクがクリアされている場合は、このレジスタへのユーザ書き込みが受け付けられます。スティムラサポートへの特権アクセスを使用すると、RTOS カーネルは、必要に応じて計装スロットすなわち帯域幅を保証することができます。

## ITM トレース特権レジスタ

ITM トレース特権レジスタは、オペレーティングシステムによって、ユーザコードによりアクセス可能なスティムラスポートの制御を行うために使用されます。

注

特権モードでのみ、このレジスタに書き込むことができます。

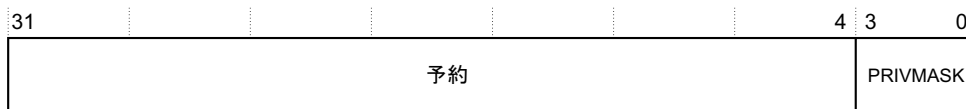
このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アクセス      読み出し / 書き込み

アドレス 0xE0000E40

リセット時の状態 0x00000000

ITM トレース特権レジスタのビット割り当てを、図 11-13 に示します。



**図 11-13 ITM トレース特権レジスタのビット割り当て**

ITM トレース特権レジスタのフィールドの説明を、表 11-21 に示します。

表 11-21 ITM トレース特権レジスタのビット割り当て

ビット	フィールド	機能
[31:4]	-	予約
[3:0]	PRIVMASK	ITM スティムラスポートでのトレーシングを許可するビットマスク ビット [0] = スティムラスポート [7:0] ビット [1] = スティムラスポート [15:8] ビット [2] = スティムラスポート [23:16] ビット [3] = スティムラスポート [31:24]

ITM トレース制御レジスタ

このレジスタは、ITM 転送の構成と制御に使用します。

注

特権モードでのみ、このレジスタに書き込むことができます。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アクセス           読み出し / 書き込み

アドレス           0xE0000E80

リセット時の状態   0x00000000

ITM トレース制御レジスタのビット割り当てを、図 11-14 に示します。

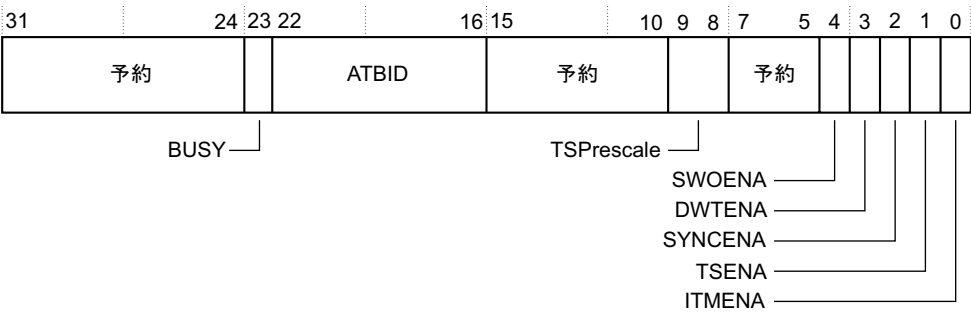


図 11-14 ITM トレース制御レジスタのビット割り当て

ITM 制御レジスタのフィールドの説明を、表 11-22 に示します。

表 11-22 ITM トレース制御レジスタのビット割り当て

ビット	フィールド	機能
[31:24]	-	0b00000000
[23]	BUSY	ITM イベントが存在し、出力されているときにセットされます。
[22:16]	ATBID	CoreSight システム用の ATB ID
[15:10]	-	0b000000

表 11-22 ITM トレース制御レジスタのビット割り当て

ビット	フィールド	機能
[9:8]	TSPrescale	タイムスタンプのプリスケアラ 0b00 = プリスケーリングなし 0b01 = 4 で除算 0b10 = 16 で除算 0b11 = 64 で除算
[7:5]	-	予約
[4]	SWOENA	SWV 動作 -TPIUEMIT および TPIUBAUD でのカウントを許可します。
[3]	DWTENA	DWT スティムラスを許可します。
[2]	SYNCENA	TPIU の同期化パケットを許可します。
[1]	TSENA	差分タイムスタンプを許可します。パケットが、0 ではないタイムスタンプ カウンタで FIFO に書き込まれ、タイムスタンプカウンタがオーバフローした場合に、差分タイムスタンプが送信されます。 決まったサイクル数の後のアイドル時間中に、タイムスタンプが送信されます。これにより、パケットおよびパケット間ギャップの時間参照が提供されます。
[0]	ITMENA	ITM を許可します。これはマスタイネーブルであり、ITM スティムラスおよびトレースイネーブル レジスタに書き込まれる前にセットする必要があります。

——— 注 ———

DWT は ITM ブロック内では許可されません。ただし、FIFO への DWT スティムラス エントリは DWTENA によって制御されます。DWT がタイムスタンプを要求する場合は、TSSSEN ビットをセットする必要があります。

ITM 統合書き込みレジスタ

このレジスタを使用して、ATVALIDM ビットの動作を決定します。  
ITM 統合書き込みレジスタのビット割り当てを、図 11-15 に示します。

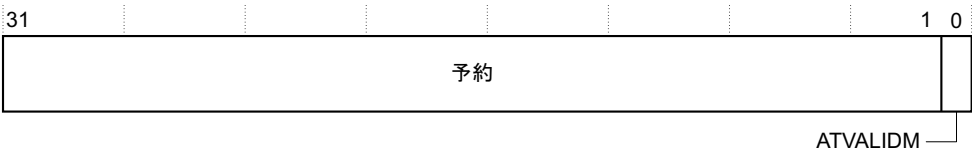


図 11-15 ITM 統合書き込みレジスタのビット割り当て

ITM 統合書き込みレジスタのフィールドの説明を、表 11-23 に示します。

表 11-23 ITM 統合書き込みレジスタのビット割り当て

ビット	フィールド	機能
[31:1]	-	予約
[0]	ATVALIDM	統合モードが設定されるのは、次の場合です。 0 = ATVALIDM クリア 1 = ATVALIDM セット

注

モードが設定されると、ビット [0] により ATVALIDM が駆動されます。

ITM 統合読み出しレジスタ

このレジスタは、ATREADYM の値を読み出すために使用します。

ITM 統合読み出しレジスタのビット割り当てを、図 11-16 に示します。

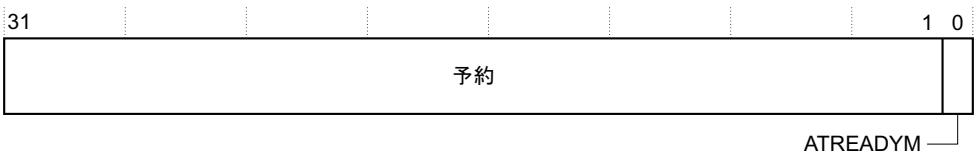


図 11-16 ITM 統合読み出しレジスタのビット割り当て

ITM 統合読み出しレジスタのフィールドの説明を、表 11-24 に示します。

表 11-24 ITM 統合読み出しレジスタのビット割り当て

ビット	フィールド	機能
[31:1]	-	予約
[0]	ATREADYM	ATREADYM の値

ITM 統合モード制御レジスタ

このレジスタは、制御レジスタへの書き込みアクセスを許可するために使用します。

ITM 統合モード制御レジスタのビット割り当てを、P.11-36 図 11-17 に示します。

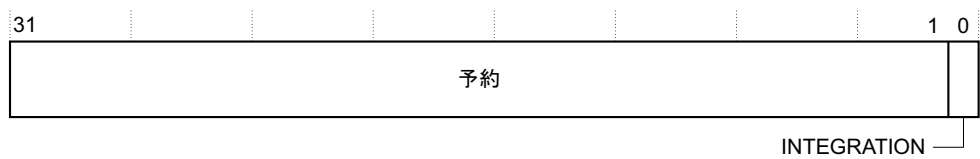


図 11-17 ITM 統合モード制御レジスタのビット割り当て

ITM 統合モード制御レジスタのフィールドの説明を、表 11-25 に示します。

表 11-25 ITM 統合モード制御レジスタのビット割り当て

ビット	フィールド	機能
[31:1]	-	予約
[0]	INTEGRATION	0 = ATVALIDM 標準 1 = ATVALIDM を統合書き込みレジスタから駆動

ITM ロックアクセス レジスタ

このレジスタは、制御レジスタへの書き込みアクセスを禁止するために使用します。

ITM ロックアクセス レジスタのフィールドの説明を、表 11-26 に示します。

表 11-26 ITM ロックアクセス レジスタのビット割り当て

ビット	フィールド	機能
[31:0]	ロックアクセス	0xC5ACCE55 の特権書き込みにより、制御レジスタ 0xE00::0xFFC への書き込みアクセスが許可されます。無効な書き込みにより、書き込みアクセスが取り消されます。

ITM ロックステータス レジスタ

このレジスタは、制御レジスタへの書き込みアクセスを許可するために使用します。

ITM ロックステータス レジスタのビット割り当てを、P.11-37 図 11-18 に示します。



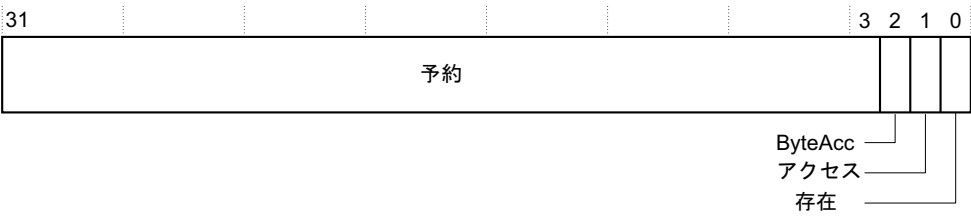


図 11-18 ITM ロックステータス レジスタのビット割り当て

ITM ロックステータス レジスタのフィールドの説明を、表 11-27 に示します。

表 11-27 ITM ロックステータス レジスタのビット割り当て

ビット	フィールド	機能
[31:3]	-	予約
[2]	ByteAcc	8 ビットロック アクセスは実装できません。
[1]	アクセス	コンポーネントへの書き込みアクセスはブロックされます。すべての書き込みが無視され、読み出しは許可されます。
[0]	存在	このコンポーネントのロック機構が存在することを示します。

11.7 AHB-AP

AHB-AP は Cortex-M3 システムへのデバッグアクセス ポートで、NVIC を経由して、プロセッサレジスタを含むシステム内のすべてのメモリおよびレジスタにアクセスできます。システムアクセスは、プロセッサステータスから独立しています。SW-DP または SWJ-DP が AHB-AP にアクセスします。

AHB-AP はバスマトリックスのマスタです。AHB-AP プログラマモデルを使用してトランザクションが実行され、このモデルによりバスマトリックスへの AHB-Lite トランザクションが生成されます。詳細については、*AHB-AP レジスタの概要と説明*を参照して下さい。

11.7.1 AHB-AP トランザクションタイプ

AHB-AP はバス上で連続するトランザクションを実行しないため、すべてのトランザクションはノンシーケンシャルです。AHB-AP は、アンアラインドおよびビットバンドのトランザクションが実行できます。これらのトランザクションは、バスマトリックスにより処理されます。AHB-AP トランザクションは MPU 管理の対象ではありません。AHB-AP トランザクションは FPB をバイパスするため、FPB は AHB-AP トランザクションをリマップできません。

SWJ/SW-DP が起動するトランザクションアポートは、**HABORT** と呼ばれる、AHB-AP でサポートされるサイドバンド信号を駆動します。この信号はバスマトリックスに接続されていて、バスマトリックスの状態をリセットするので、AHB-AP が専用ペリフェラルバスにアクセスして、コアの読み出し / 中止 / リセットなどの最後の手段のデバッグを行えます。

AHB-AP トランザクションはリトルエンディアンです。

11.7.2 AHB-AP レジスタの概要と説明

AHB-AP レジスタの一覧を、表 11-28 に示します。

表 11-28 AHB-AP レジスタの概要

名前	タイプ	アドレス	リセット時の値	説明
制御およびステータスワード	読み出し / 書き込み	0x00	レジスタ参照	P.11-39「 <i>AHB-AP 制御およびステータスワードレジスタ</i> 」参照
転送アドレス	読み出し / 書き込み	0x04	0x00000000	P.11-41「 <i>AHB-AP 転送アドレスレジスタ</i> 」参照
データ読み出し / 書き込み	読み出し / 書き込み	0x0C	-	P.11-41「 <i>AHB-AP データ読み出し / 書き込みレジスタ</i> 」参照
バンクデータ 0	読み出し / 書き込み	0x10	-	P.11-41「 <i>AHB-AP バンクデータ レジスタ 0 ~ 3</i> 」参照
バンクデータ 1	読み出し / 書き込み	0x14	-	P.11-41「 <i>AHB-AP バンクデータ レジスタ 0 ~ 3</i> 」参照
バンクデータ 2	読み出し / 書き込み	0x18	-	P.11-41「 <i>AHB-AP バンクデータ レジスタ 0 ~ 3</i> 」参照

表 11-28 AHB-AP レジスタの概要

名前	タイプ	アドレス	リセット時の値	説明
バンクデータ 3	読み出し / 書き込み	0x1C	-	P.11-41「AHB-AP バンクデータ レジスタ 0 ~ 3」参照
デバッグ ROM アドレス	読み出し専用	0xF8	0xE00FF003	P.11-42「AHB-AP デバッグ ROM アドレス レジスタ」参照
識別レジスタ	読み出し専用	0xFC	0x14770011	P.11-42「AHB-AP ID レジスタ」参照

AHB-AP 制御およびステータスワード レジスタ

このレジスタを使用して、AHB インタフェースを経由で転送の構成と制御を行います。  
AHB-AP 制御およびステータスワード レジスタのフィールドを、図 11-19 に示します。

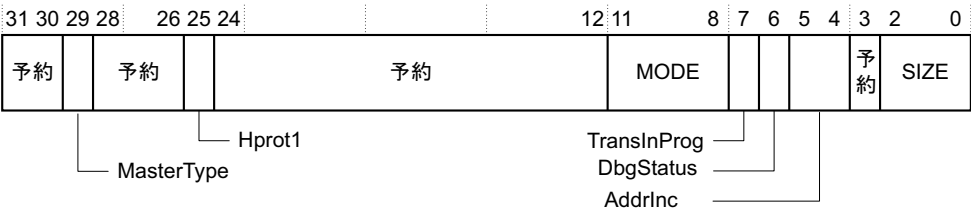


図 11-19 AHB-AP 制御およびステータスワード レジスタ

AHB-AP 制御およびステータスワード レジスタのフィールドの説明を、表 11-29 に示します。

表 11-29 AHB-AP 制御およびステータスワード レジスタのビット割り当て

ビット	フィールド	機能
[31:30]	-	予約。読み出し値 b010
[29]	MasterType <sup>a</sup>	0 = コア 1 = デバッグ COREACCEN = 0 の場合は、このビットはクリアできません。受け付けられたかの確認のため再度読み出して下さい。トランザクションが終了していない場合は、変更できません。デバッガは最初に TransInProg をチェックする必要があります。 リセット時の値 = 0b1
[28:26]	-	予約、0b000
[25]	Hprot1	ユーザ / 特権制御 -HPROT[1] リセット時の値 = 0b1
[24]	-	予約、0b1
[23:12]	-	予約、0x000

表 11-29 AHB-AP 制御およびステータスワード レジスタのビット割り当て

ビット	フィールド	機能
[11:8]	Mode	演算モードのビット b0000 = 標準ダウンロード / アップロードモード b0001 ~ b1111 は予約されています。 リセット時の値 = 0b0000
[7]	TransINProg	転送が進行中です。このフィールドは、APB マスタポート上で転送が進行中であるかどうかを示します。
[6]	DbgStatus	DBGEN ポートの状態を示します。DbgStatus が LOW の場合、AHB 転送は実行されていません。 1 = AHB 転送を許可する 0 = AHB 転送を許可しない
[5:4]	AddrInc	読み出し / 書き込みデータアクセス時の自動アドレスインクリメントおよびバックモード。現在のトランザクションがエラーなしで完了した場合にのみインクリメントされます。 自動アドレスインクリメントおよびバック転送は、バンクデータ レジスタ 0x10-0x1C へのアクセスの際は実行されません。この場合、これらのビットの状態は無視されます。4KB アドレス境界内でインクリメントとラップアラウンドが行われます。例えば、ワードが 0x1000 から 0x1FFC までインクリメントされます。開始値が 0x14A0 の場合、カウンタは 0x1FFC までインクリメントされてから、0x1000 にラップアラウンドし、0x149C までインクリメントが続行されます。 0b00 = 自動インクリメントオフ 0b01 = インクリメントシングル。対応するバイトレーンからのシングル転送 0b10 = インクリメントバック 0b11 = 予約。転送なし アドレスインクリメントのサイズは、サイズフィールド [2:0] によって定義されます。 リセット時の値 : 0b00
[3]	-	予約
[2:0]	SIZE	アクセスフィールドのサイズ b000 = 8 ビット b001 = 16 ビット b010 = 32 ビット b011 ~ 111 は予約されています。 リセット時の値 : b000

- a. クリアされている場合、このビットによりデバッグはデバッグホールド制御およびステータスレジスタの C\_DEBUGEN ビットをセットすることを禁止されるため、コアをホールドすることができません。

## AHB-AP 転送アドレスレジスタ

このレジスタは、現在の転送のアドレスをプログラムするために使用します。

AHB-AP 転送アドレスレジスタのフィールドの説明を、表 11-30 に示します。

表 11-30 AHB-AP 転送アドレスレジスタのビット割り当て

ビット	フィールド	機能
[31:0]	ADDRESS	現在の転送アドレス リセット時の値 = 0x00000000

## AHB-AP データ読み出し / 書き込みレジスタ

このレジスタは、現在の転送データの読み出し / 書き込みに使用します。

AHB-AP データ読み出し / 書き込みレジスタのフィールドの説明を、表 11-31 に示します。

表 11-31 AHB-AP データ読み出し / 書き込みレジスタのビット割り当て

ビット	フィールド	機能
[31:0]	DATA	書き込みモード：現在の転送の書き込みデータ値 読み出しモード：現在の転送の読み出しデータ値 リセット時の値 = 0x00000000

## AHB-AP バンクデータ レジスタ 0 ～ 3

これらのレジスタは、AHB-AP 転送アドレスレジスタ (TAR) に再書き込みせずに、AHB-AP アクセスを直接 AHB 転送へマップするために使用します。

AHB-AP バンクデータ レジスタのフィールドの説明を、表 11-32 に示します。

表 11-32 AHB-AP バンクデータ レジスタのビット割り当て

ビット	フィールド	機能
[31:0]	DATA	<p>BD0 ～ BD3 により、4 つの位置境界内の TAR を再書き込みせずに、DAP アクセスを經由して直接 AHB 転送へマップする機構が提供されます。例えば、TAR からの BD0 読み出し / 書き込み、TAR + 4 からの BD1 読み出し / 書き込みがそれに相当します。</p> <p><b>DAPADDR[7:4] == 0x0001</b> の場合は、範囲 0x10-0x1C の AHB-AP レジスタにアクセスして取得される <b>HADDR[31:0]</b> は次のとおりです。</p> <p>読み出しモード：外部アドレス TAR[31:4] + <b>DAPADDR[3:0]</b> から現在転送されている読み出しデータ値。BD0 ～ BD3 への DAP アクセスでは、自動アドレスインクリメントは実行されません。</p> <p>書き込みモード：外部アドレス TAR[31:4] + <b>DAPADDR[3:0]</b> へ現在転送されている書き込みデータ値。</p> <p>バンク転送は、ワード転送についてのみサポートされます。現在のところ、ワード以外のバンク転送サイズは無視され、ワードアクセスと見なされます。</p> <p>リセット時の値 = 0x00000000</p>

AHB-AP デバッグ ROM アドレスレジスタ

このレジスタは、デバッグインタフェースのベースアドレスを指定します。読み出し専用です。

AHB-AP デバッグ ROM アドレスレジスタのフィールドの説明を、表 11-33 に示します。

表 11-33 AHB-AP デバッグ ROM アドレスレジスタのビット割り当て

ビット	フィールド	機能
[31:0]	デバッグ ROM アドレス	デバッグインタフェースのベースアドレス

AHB-AP ID レジスタ

このレジスタは、アクセスポートの外部インタフェースを定義します。

AHB-AP ID レジスタのフィールドを、P.11-43 図 11-20 に示します。

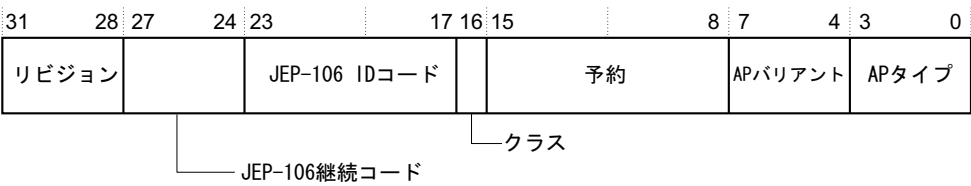


図 11-20 AHB-AP ID レジスタ

AHB-AP ID レジスタのフィールドの説明を、表 11-34 に示します。

表 11-34 AHB-AP ID レジスタのビット割り当て

ビット	フィールド	機能
[31:28]	リビジョン	このフィールドは AP 設計の最初の実装では 0 ですが、設計の主要な改訂ごとに更新されます。
[27:24]	JEP-106 継続コード	ARM で設計された AP の場合は、このフィールドの値は 0b0100、0x4 です。
[23:17]	JEP-106ID コード	ARM で設計された AP の場合は、このフィールドの値は 0b0111011、0x3B です。
[16]	クラス	0b1: この AP はメモリアクセス ポートです。
[15:8]	-	予約。SBZ
[7:4]	AP バリエーション	0x1:Cortex-M3 バリエーション
[3:0]	AP タイプ	0x1:AMBA AHB バス





## 第 12 章

# デバッグポート

本章では、プロセッサのデバッグポート (DP) について説明します。本章は以下のセクションから構成されています。

- DP について: P.12-2

## 12.1 DP について

プロセッサには、アドバンスト ハイパフォーマンスバス アクセスポート (AHB-AP) インタフェースがデバッグアクセス用に組み込まれています。外部の DP コンポーネントは、このインタフェースにアクセスします。Cortex-M3 システムは、次の 2 つの DP 実装をサポートします。

- シリアルワイヤ JTAG デバッグポート (SWJ-DP)。SWJ-DP は、JTAG-DP とシリアルワイヤ デバッグポート (SW-DP) を組み合わせた、標準的な CoreSight デバッグポートです。
- SW-DP。AHB-AP ポートへの 2 ピン (クロック + データ) インタフェースを提供します。

---

### 注

SWJ-DP は、JTAG デバッグアクセスに使用されていないときは、**JTAG-TDO** と **JTAG-TDI** のピンの共有ができるように設計されています。Cortex-M3 TPIU とともに使用するときには、シリアルワイヤ出力 (SWO) の接続に別のオプションがあります。詳細については、P.13-17「シリアルワイヤ出力接続」を参照して下さい。

これら 2 つの DP 実装では、プロセッサへのデバッグアクセスを行うために提供される機構が異なります。実装には、一方のコンポーネントのみを搭載する必要があります。

---

### 注

実装には、SW-DP や SWJ-DP の代わりに、実装者固有の DP が搭載されていることもあります。詳細については、実装者に問い合わせして下さい。

DP コンポーネントの詳細については、『CoreSight コンポーネント テクニカルリファレンスマニュアル』を参照して下さい。

AHB-AP の詳細については、P.11-38「AHB-AP」を参照して下さい。

DP と AP は、組み合わせてデバッグアクセス ポート (DAP) と呼ばれます。

デバッグインタフェースの詳細については、『ARM デバッグインタフェース v5、アーキテクチャ仕様書』を参照して下さい。

## 第 13 章

# トレースポート インタフェースユニット

本章では、トレースポート インタフェースユニット (TPIU)(Trace Port Interface Unit) について説明します。本章は以下のセクションから構成されています。

- *TPIU* について : P.13-2
- *TPIU* レジスタ : P.13-8
- シリアルワイヤ出力接続 : P.13-17

## 13.1 TPIU について

TPIU は、エンベデッドトレース マクロセル (ETM) および計装トレース マクロセル (ITM) からのオンチップ トレースデータ間のブリッジとして動作し、個別の ID でひとつのデータストリームにし、必要に応じて ID をカプセル化しながら、トレースポート アナライザ (TPA)(Trace Port Analyzer) によってキャプチャされます。

TPIU は低コストのデバッグ用に特に設計されています。また、CoreSight TPIU の特別バージョンであり、システム要件により CoreSight TPIU の追加機能が必要とされる場合は、CoreSight コンポーネントで置き換えることができます。

TPIU には、次の 2 つの構成があります。

- ITM デバッグトレースをサポートする構成
- ITM および ETM デバッグトレースの両方をサポートする構成

—— 注 ——

Cortex-M3 システムでオプションの ETM コンポーネントを使用する場合は、ITM および ETM デバッグトレースの両方をサポートする TPIU の構成を使用する必要があります。ETM の詳細については、第 15 章 「エンベデッドトレース マクロセル」を参照して下さい。

### 13.1.1 TPIU のブロック図

TPIU の 2 つの構成に対応するコンポーネントの構成を、P.13-3 図 13-1 と P.13-4 図 13-2 に示します。

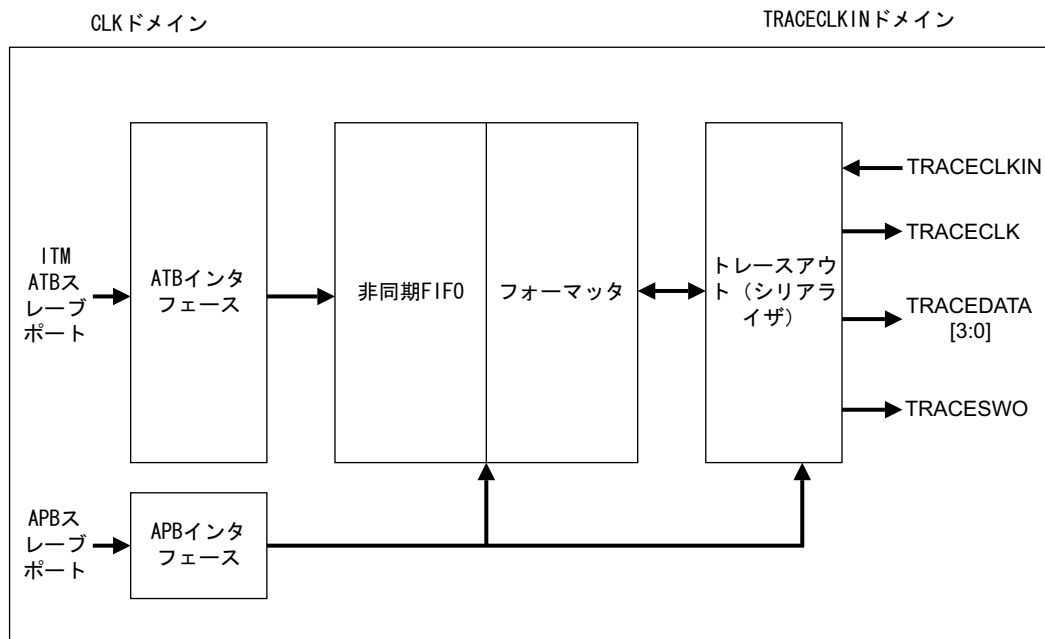


図 13-1 TPIU のブロック図 (非 ETM バージョン)

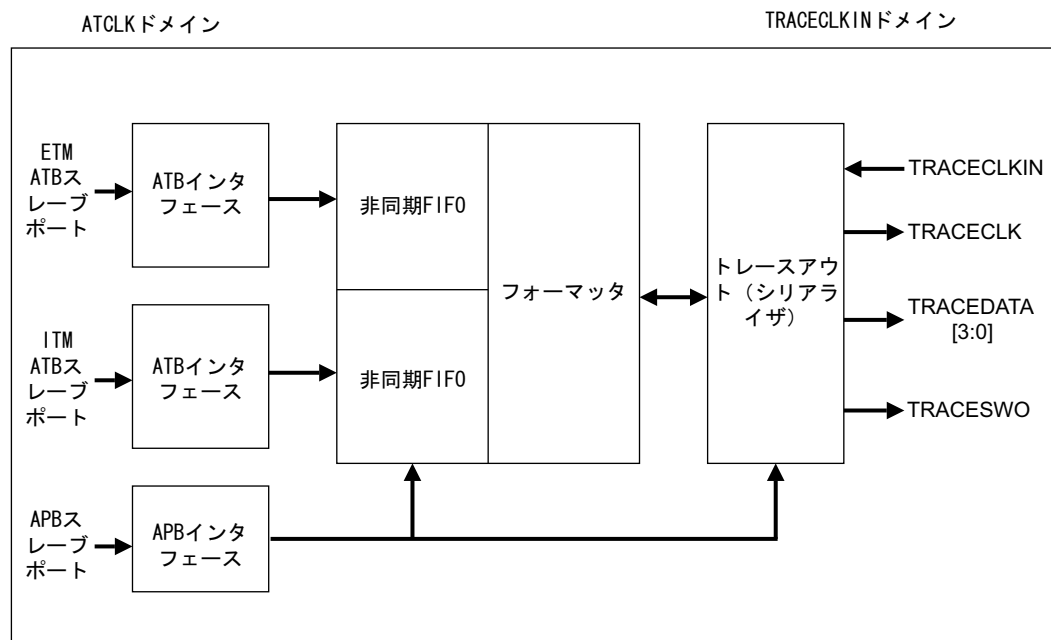


図 13-2 TPIU のブロック図 (ETM バージョン)

### 13.1.2 TPIU のコンポーネント

以下のセクションでは、TPIU の主要なコンポーネントについて説明します。

- **非同期 FIFO** : P.13-4
- **フォーマッタ**
- **トレース出力** : P.13-5
- **AMBA トレースバス インタフェース** : P.13-5
- **アドバンストペリフェラルバス インタフェース** : P.13-5

#### 非同期 FIFO

非同期 FIFO を使用すると、トレースデータをコアクロックの速度に依存しない速度で取り出すことができます。

#### フォーマッタ

フォーマッタは、トレースデータをトレースソースと再び対応付けできるように、ソース ID 信号をデータパケットに挿入します。TRACEPORT モードがアクティブな場合、フォーマッタは常時アクティブです。

トレース出力

トレース出力ブロックは、データがチップの外に送られる前に、フォーマットされたデータの直列化を行います。

AMBA トレースバス インタフェース

TPIU は、トレースソースからのトレースデータを、トレースソース (ETM または ITM) から直接受け取るか、またはトレースファンネルを使用して受け取ります。詳細については、AMBA トレースバス インタフェースを参照して下さい。

アドバンストペリフェラルバス インタフェース

APB インタフェースは、TPIU 用のプログラミングインタフェースです。詳細については、アドバンストペリフェラルバス インタフェースを参照して下さい。

13.1.3 TPIU 入力および出力

このセクションでは、TPIU 入力および出力について説明します。このセクションは次の項目から構成されています。

- トレース出力ポート : P.13-5
- AMBA トレースバス インタフェース
- その他の構成入力 : P.13-6

トレース出力ポート

トレース出力ポートの信号の説明を、表 13-1 に示します。

表 13-1 トレース出力ポートの信号

名前	タイプ	説明
TRACECLKIN	入力	トレースポート速度の簡単な制御を可能にするための ATB からの分離クロック。一般的に、このクロックはチップ上の制御可能なクロックソースから作成されますが、高速度ピンを使用する場合は外部クロックジェネレータでも駆動できます。データは立ち上がりエッジでのみ変更されます。
TRESETn	入力	これは TRACECLKIN ドメインのリセット信号です。この信号は通常はパワーオン リセットで駆動され、TRACECLKIN に同期している必要があります。
TRACECLK	出力	TRACEDATA は、TRACECLK の両方のエッジで変更されます。
TRACEDATA[3:0]	出力	クロックモードの出力データ
TRACESWO	出力	非同期モードの出力データ

ATB インタフェース

TPIU の構成に応じて、1 つまたは 2 つの ATB インタフェースが存在します。ATB ポートの信号の説明を、表 13-2 に示します。TPIU が単一の ATB インタフェースで構成されている場合は、ポート 2 の信号は使用されません。

表 13-2 ATB ポートの信号

名前	タイプ	説明
CLK	入力	トレースバスおよび ATB インタフェースクロック
nRESET	入力	CLK ドメイン (ATB/APB インタフェース) のリセット
CLKEN	入力	CLK ドメインのクロックイネーブル
ATVALID1S	入力	トレースソース 1 からのデータが、このサイクルで有効なことを示します。
ATREADY1S	出力	この信号がアサートされている (ATVALID が HIGH) 場合、データはこのサイクルにトレースソース 1 から受け付けられました。
ATDATA1S[7:0]	入力	ソース 1 からのトレースデータ入力
ATID1S[6:0]	入力	ソース 1 のトレースソース ID。これは動的には変更できません。
ATVALID2S	入力	トレースソース 2 からのデータが、このサイクルで有効なことを示します。
ATREADY2	出力	この信号がアサートされている (ATVALID が HIGH) 場合、データはこのサイクルにトレースソース 2 から受け付けられました。
ATDATA2S[7:0]	入力	ソース 2 からのトレースデータ入力
ATID2S[6:0]	入力	ソース 2 のトレースソース ID。これは動的には変更できません。

その他の構成入力

その他の構成入力の説明を、表 13-3 に示します。



表 13-3 その他のコンフィギュレーション入力

名前	タイプ	説明
<b>MAXPORTSIZE</b> [1:0]	入力	同期トレース出力で使用可能なピンの最大数を定義します。
<b>SyncReq</b>	入力	グローバルトレース同期化トリガ。同期化パッケージを、フォーマットされたデータストリームに挿入します。フォーマッタがアクティブな場合にのみ使用されます。この信号は、Cortex-M3 の <b>DSYNC</b> 出力に接続する必要があります。
<b>TRIGGER</b>	入力	フォーマッタがアクティブな場合に、トリガパッケージをトレースストリームに挿入します。
<b>SWOACTIVE</b>	出力	<b>SWO</b> モードが選択されます（ピン多重化に使用）。
<b>TPIUACTIV</b>	出力	<b>TPIU</b> に出力プロセス中のデータがあることを示します。
<b>TPIUBAUD</b>	出力	ボー周波数で切り換わります（ <b>TRACECLKIN</b> ドメイン）。

13.2 TPIU レジスタ

このセクションでは、TPIU レジスタについて説明します。このセクションは次の項目から構成されています。

- TPIU レジスタの概要
- TPIU レジスタの説明

13.2.1 TPIU レジスタの概要

TPIU レジスタの概要を、表 13-4 に示します。

表 13-4 TPIU レジスタ

レジスタ名	タイプ	アドレス	リセット時の値	ページ
サポートされる同期化ポートサイズ レジスタ (Supported Sync Port Sizes Register)	読み出し専用	0xE0040000	0bxx0x	P.13-9
カレント同期化ポートサイズ レジスタ (Current Sync Port Size Register)	読み出し / 書き込み	0xE0040004	0x01	P.13-9
非同期クロックプリスケアラ レジスタ (Async Clock Prescaler Register)	読み出し / 書き込み	0xE0040010	0x0000	P.13-9
選択ピンプロトコル レジスタ (Selected Pin Protocol Register)	読み出し / 書き込み	0xE00400F0	0x01	P.13-10
トリガ制御レジスタ	読み出し専用	0xE0040100 ～ 0xE0040108	-	P.13-11
EXTCTL ポートレジスタ	読み出し専用	-	-	P.13-11
テストパターン レジスタ	読み出し専用	0xE0040200 ～ 0xE0040208	-	P.13-11
フォーマッタおよびフラッシュステータス レジスタ (Formatter and Flush Status Register)	読み出し専用	0xE0040300	0x08	P.13-11
フォーマッタおよびフラッシュ制御レジスタ (Formatter and Flush Control Register)	読み出し / 書き込み	0xE0040304	0x102	P.13-12
フォーマッタ同期化カウンタレジスタ (Formatter Synchronization Counter Register)	読み出し専用	0xE0040308	0x00	P.13-14
統合レジスタ : ITATBCTR2	読み出し専用	0xE0040EF0	0x0	P.13-15
統合レジスタ : ITATBCTR0	読み出し専用	0xE0040EF8	0x0	P.13-15

13.2.2 TPIU レジスタの説明

このセクションでは、TPIU レジスタについて説明します。

## サポートされる同期化ポートサイズレジスタ

このレジスタは読み出し / 書き込み可能です。各ビット位置が、そのデバイスでサポートされているポートサイズを表していて、ビット位置 [3:0] では 4、2、または 1 を表します。ビットがセットされると、そのポートサイズが許可されます。デフォルトでは、RTL はすべてのポートサイズをサポートするように設計されており、0x0000000B に設定されます。このレジスタは、入力タイオフ **MAXPORTSIZE** によって制約を受けます。外部のタイオフ **MAXPORTSIZE** は、物理ピンに結線された **TRACEDATA** 信号の実際の数を示すように、ASIC の最終段階で設定される必要があります。これにより、接続された TPA がキャプチャできないポート幅の選択をツールが試みないことが保証されます。**MAXPORTSIZE** の値により、より広い幅、つまりサポートされていない幅を表す、サポートされる同期化ポートサイズ レジスタ内のビットがクリアされます。

ビット割り当てを図 13-3 に示します。

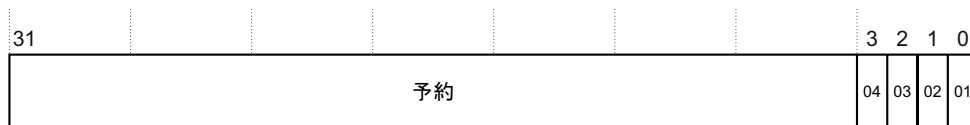


図 13-3 サポートされる同期化ポートサイズレジスタのビット割り当て

## カレント同期化ポートサイズレジスタ

このレジスタは読み出し / 書き込み可能です。カレント同期化ポートサイズ レジスタは、サポートされる同期化ポートサイズ レジスタと同じフォーマットを持ちますが、1 ビットのみがセットされ、他のすべてのビットは 0 の必要があります。複数のビットがセットされている値の書き込み、およびサポートされていることが示されていないビットのセットはサポートされておらず、予測不能な動作を引き起こします。

このレジスタは、サポートされる同期化ポートサイズ レジスタと同じフォーマットを使用するのが便利です。これは、デバイス内で後にサイズをデコードする必要性を省き、有効な割り当てをチェックする他のレジスタバンクのフォーマットも維持するからです。

リセット時、このレジスタのデフォルト値は可能な最小ポートサイズ、つまり 1 ビットであるため、読み出し値は 0x00000001 です。

## 非同期クロックプリスケーラ レジスタ

非同期クロックプリスケアラ レジスタは、非同期出力のボーレート进行调整するために使用します。

非同期クロックプリスケアラ レジスタのフィールドを、P.13-10 図 13-4 に示します。

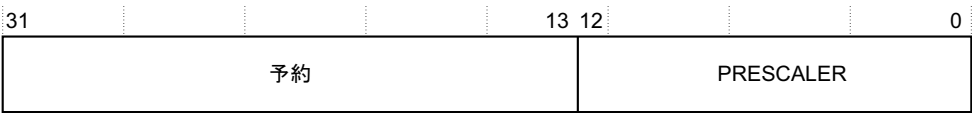


図 13-4 非同期クロックプリスケラ レジスタのビット割り当て

非同期クロックプリスケラ レジスタのフィールドの説明を、表 13-5 に示します。

表 13-5 非同期クロックプリスケラ レジスタのビット割り当て

ビット	フィールド	機能
[31:13]	-	予約。RAZ/SBZP
[12:0]	PRESCALER	TRACECLKIN の除数は、PRESCALER + 1 で指定されます。

選択ピンプロトコル レジスタ

選択ピンプロトコル レジスタを使用して、トレース出力に使用するプロトコルを選択します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE00400F0  
アクセス 読み出し / 書き込み  
リセット時の状態 0x01

選択ピンプロトコル レジスタのフィールドを、図 13-5 に示します。



図 13-5 選択ピンプロトコル レジスタのビット割り当て

選択ピンプロトコル レジスタのフィールドの説明を、表 13-6 に示します。

表 13-6 選択ピンプロトコル レジスタのビット割り当て

ビット	フィールド	機能
[31:2]	-	予約
[1:0]	PROTOCOL	00-TracePort モード 01-SerialWire 出力（マンチェスタ）。これはリセット時の値です。 10-SerialWire 出力 (NRZ) 11- 予約

—— 注 ——

トレースデータが出力中にこのレジスタが変更されると、データが破損します。

トリガ制御レジスタ

この TPIU はトリガ遅延をサポートしていません。これを示すために、トリガ制御レジスタは 0 として読み出されます。

EXTCTL ポートレジスタ

この TPIU は EXTCTL ポートをサポートしていません。これを示すために、EXTCTL ポートレジスタは 0 として読み出されます。

テストパターン レジスタ

この TPIU には、テストパターン ジェネレータが組み込まれていません。これを示すために、テストパターン ジェネレータは 0 として読み出されます。

フォーマッタおよびフラッシュステータス レジスタ

フォーマッタおよびフラッシュステータス レジスタは、TPIU フォーマッタのステータスを読み出すために使用します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE0040300  
アクセス 読み出し専用  
リセット時の状態 0x08

フォーマッタおよびフラッシュステータス レジスタのフィールドを、P.13-12 図 13-6 に示します。

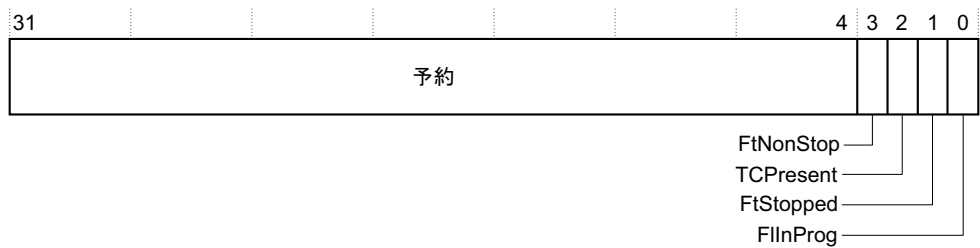


図 13-6 フォーマッタおよびフラッシュステータス レジスタのビット割り当て

フォーマッタおよびフラッシュステータス レジスタのフィールドの説明を、表 13-7 に示します。

表 13-7 フォーマッタおよびフラッシュステータス レジスタのビット割り当て

ビット	フィールド	機能
[31:4]	-	予約
[3]	FtNonStop	フォーマッタは中止できません。
[2]	TCPresent	このビットは、常に 0 として読み出されます。
[1]	FtStopped	このビットは、常に 0 として読み出されます。
[0]	FIInProg	このビットは、常に 0 として読み出されます。

フォーマッタおよびフラッシュ制御レジスタ

フォーマッタおよびフラッシュ制御レジスタ。  
このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE0040304  
アクセス 読み出し / 書き込み  
リセット時の状態 0x102

フォーマッタおよびフラッシュ制御レジスタのフィールドを、P.13-13 図 13-7 に示します。

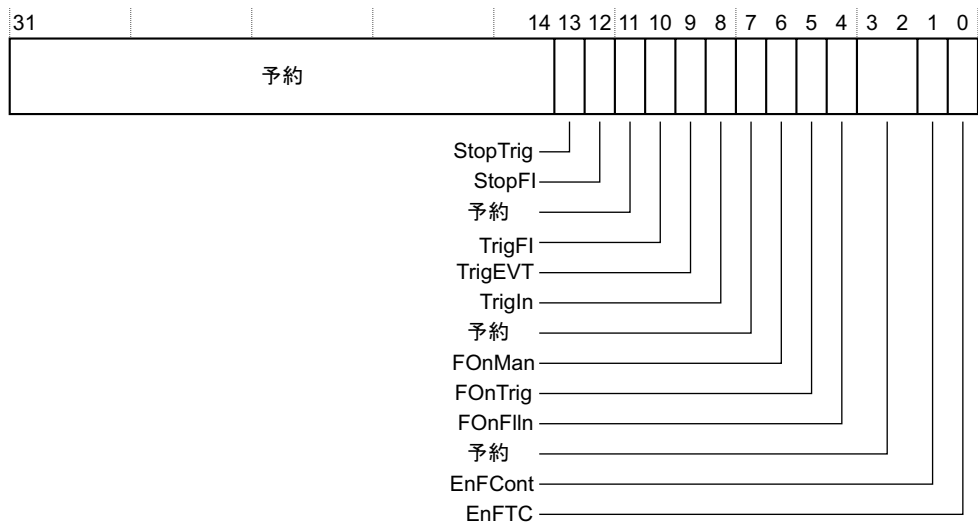


図 13-7 フォーマットおよびフラッシュ制御レジスタのビット割り当て

フォーマットおよびフラッシュ制御レジスタのフィールドの説明を、表 13-8 に示します。

表 13-8 フォーマットおよびフラッシュ制御レジスタのビット割り当て

ビット	フィールド	機能
[31:14]	-	予約
[13]	StopTrig	トリガイイベントを観測後に、フォーマットを停止します。
[12]	StopFI	フラッシュの完了後に、フォーマットを停止します。
[11]	-	予約
[10]	TrigFI	フラッシュ完了時のトリガを示します。
[9]	TrigEVT	トリガイイベント時のトリガを示します。
[8]	TrigIN	<b>TRIGIN</b> アサート時のトリガを示します。
[7]	-	予約
[6]	FOnMan	システムのフラッシュを手動で生成します。
[5]	FOnTrig	トリガイイベントを使用してフラッシュを生成します。
[4]	FOnFlIn	<b>FLUSHIN</b> インタフェースを使用してフラッシュを生成します。

表 13-8 フォーマッタおよびフラッシュ制御レジスタのビット割り当て

ビット	フィールド	機能
[3:2]	-	予約
[1]	EnFCont	連続フォーマッティング、 <b>TRACECTL</b> なし。このビットはリセット時にセットされます。
[0]	EnFTC	フォーマッティングを許可します。 <b>TRACECTL</b> が存在しないため、このビットは 0 として読み出されます。

このレジスタのビット [8] は常時セットされており、**TRIGGER** がアサートされたときのトリガを示しています。

2 つのシングルワイヤ出力モードのうち 1 つが選択されると、このレジスタのビット [1] によりフォーマッタのバイパスが許可されます。フォーマッタがバイパスされると、ITM/DWT トレースソース (ATDATA2) のみが通過します。TPIU は、ETM ポート (ATDATA1) 上に存在するデータを受け付けて廃棄します。この機能は、ETM を含むデバイスを、シリアルワイヤ出力データのみをキャプチャできるトレースキャプチャデバイスに接続する必要がある場合に使用することを想定しています。フォーマッタを許可または禁止すると、データが一時的に破損します。

——— 注 ———

選択ピンプロトコル レジスタが 0x00 (TracePort モード) に設定されると、フォーマッタおよびフラッシュ制御レジスタの読み出し値は常時 0x102 になります。これはフォーマッタが自動的に許可されるためです。その後で、シリアルワイヤ モードのいずれかが選択されると、このレジスタは以前にプログラムされた値に戻ります。

フォーマッタ同期化カウンタレジスタ

グローバル同期化トリガは、プログラムカウンタ (PC) サンプラブロックによって生成されます。これは、TPIU に同期化カウンタがないことを意味します。

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

- アドレス 0xE0040308
- アクセス 読み出し専用
- リセット時の状態 0x00



統合テストレジスタ

統合テストレジスタは、Cortex-M3 システム内で他のデバイスを使用し、TPIU のトポロジ検出を行うために使用します。これらのレジスタにより、出力の直接制御および入力値の読み出し機能が許可されます。プロセッサには、次の 2 つの統合テストレジスタがあります。

- 統合テストレジスタ -ITATBCTR2
- 統合テストレジスタ -ITATBCTR0

統合テストレジスタ -ITATBCTR2

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE0040EF0  
アクセス 読み出し専用  
リセット時の状態 0x0

統合テストレジスタのビット割り当てのフィールドを、図 13-8 に示します。

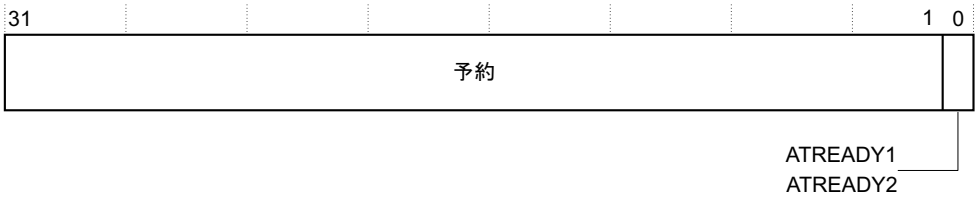


図 13-8 統合テストレジスタ -ITATBCTR2 のビット割り当て

統合テストレジスタのビット割り当てフィールドの説明を、表 13-9 に示します。

表 13-9 統合テストレジスタ -ITATBCTR2 のビット割り当て

ビット	フィールド	機能
[31:1]	-	予約
[0]	ATREADY1	このビットにより、ATREADYS1 および ATREADYS2 の値が読み出し、または設定されます。

統合テストレジスタ -ITATBCTR0

このレジスタのアドレス、アクセスタイプ、リセット時の状態は次のとおりです。

アドレス 0xE0040EF8  
アクセス 読み出し専用  
リセット時の状態 0x0

統合テストレジスタのビット割り当てのフィールドを、図 13-9 に示します。

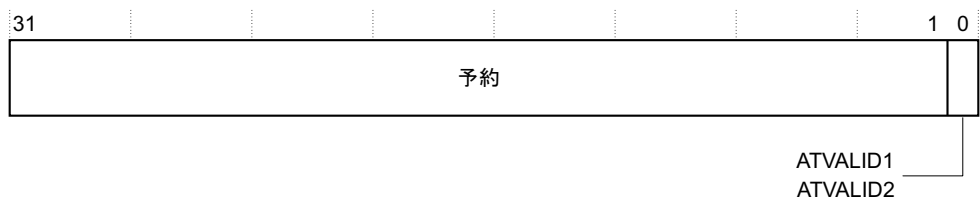


図 13-9 統合テストレジスタ -ITATBCTR0 のビット割り当て

統合テストレジスタのビット割り当てフィールドの説明を、表 13-10 に示します。

表 13-10 統合テストレジスタ -ITATBCTR0 のビット割り当て

ビット	フィールド	機能
[31:1]	-	予約
[0]	ATVALID1、 ATVALID2	このビットには、ATVALIDS1 と ATVALIDS2 の値に対して OR 演算を行った結果が、読み出しましたは設定されます。

## 13.3 シリアルワイヤ出力接続

Cortex-M3 の TPIU にはシリアルワイヤ出力モードが提供されており、一本の外部ピンが必要になります。このピンの接続には、次の 3 つのオプションが使用できます。

- *TRACESWO 専用ピンの使用*
- *TRACEPORT と SWO の共用*
- *JTAG-TDO と SWO の共用* : P.13-18

### 13.3.1 TRACESWO 専用ピンの使用

これは最も単純なオプションですが、パッケージに追加のピンが必要になります。専用ピンオプションを、図 13-10 に示します。

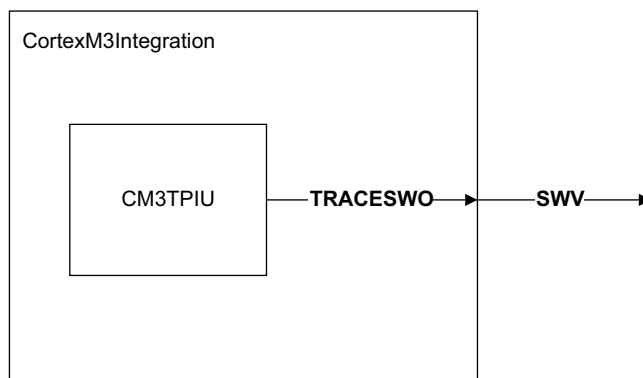


図 13-10 TRACESWO 専用ピン

### 13.3.2 TRACEPORT と SWO の共用

**TRACEDATA[0]** と **TRACESWO** でピンを共用できます。これら 2 つのピンのうち、同時に使用できるのは 1 つだけなので、このオプションの使用によって機能が失われることはありません。パッケージに専用トレースポートが存在する場合は、このオプションを使用するのが適切です。

このオプションを実装するには、Cortex-M3 の TPIU の **SWOACTIVE** 出力を使用して、マルチプレクサを制御します。TRACEPORT と SWO を共用するオプションを、P.13-18 図 13-11 に示します。

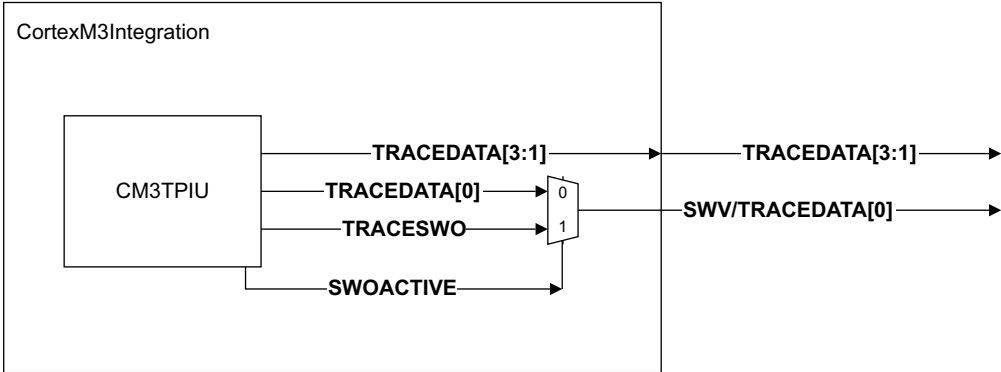


図 13-11 TRACEPORT と SWO の共用

13.3.3 JTAG-TDO と SWO の共用

ピンの数をできるだけ減らしたい場合は、パッケージの同一ピンに JTAG デバッグと SWO をオーバーレイすることが可能です。この手法は、従来のトレースポート用の機構、またはより複雑なシステムレベルのデバッグコンフィギュレーション制御で使用するための機構が存在しない場合にのみ推奨されます。

このオプションを選択した場合、JTAG 構成でデバッグポートが使用されている間は、計装トレースにアクセスできません。シリアルワイヤデバッグと SWO は同時に使用できます。

このオプションを実装するには、SWJ-DP の JTAGNSW 出力を使用してマルチプレクサを制御します。JTAG-TDO と SWO を共用するオプションを、図 13-12 に示します。

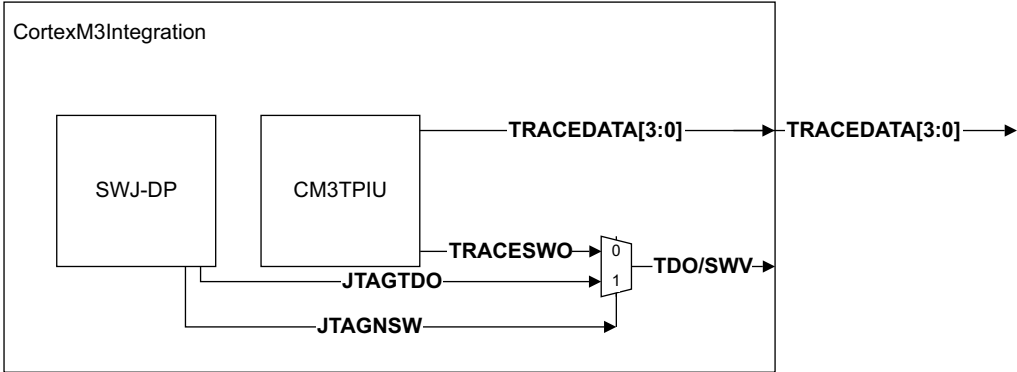


図 13-12 JTAG-TDO と SWO の共用

# 第 14 章

## バスインタフェース

本章では、プロセッサのバスインタフェースについて説明します。本章は以下のセクションから構成されています。

- バスインタフェースについて : P.14-2
- *AMBA 3* への準拠 : P.14-3
- *ICode* バスインタフェース : P.14-4
- *DCode* バスインタフェース : P.14-6
- システムインタフェース : P.14-7
- コードバスの統合 : P.14-9
- 外部専用ペリフェラルインタフェース : P.14-10
- アクセスのアライメント : P.14-11
- 領域にまたがるアンアラインドアクセス : P.14-12
- ビットバンドアクセス : P.14-13
- ライトバッファ : P.14-14
- メモリ属性 : P.14-15
- *AHB* のタイミング特性 : P.14-16

## 14.1 バスインタフェースについて

プロセッサには、次の4つのバスインタフェースが組み込まれています。

- **ICode** メモリインタフェース。コードメモリ空間 (0x00000000 ~ 0xFFFFFFFF) からの命令フェッチは、この 32 ビットアドバンスド ハイパフォーマンスバス ライト (AHB-Lite) バス経由で行われます。詳細については、P.14-4「*ICode* バスインタフェース」を参照して下さい。
- **DCode** メモリインタフェース。コードメモリ空間 (0x00000000 ~ 0xFFFFFFFF) へのデータおよびデバッグアクセスは、この 32 ビット AHB-Lite バス経由で行われます。詳細については、P.14-6「*DCode* バスインタフェース」を参照して下さい。
- **システム** インタフェース。システム空間 (0x20000000 ~ 0xDFFFFFFF, 0xE0100000 ~ 0xFFFFFFFF) への命令フェッチとデータおよびデバッグアクセスは、この 32 ビット AHB-Lite バス経由で行われます。詳細については、P.14-7「**システム** インタフェース」を参照して下さい。
- **外部専用** ペリフェラルバス (PPB)。外部 PPB 空間 (0xE0040000 ~ 0xE00FFFFFF) へのデータおよびデバッグアクセスは、この 32 ビットアドバンスド ペリフェラルバス (APB)(AMBA v2.0) 経由で行われます。トレースポート インタフェースユニット (TPIU) およびベンダ固有のペリフェラルは、このバスに接続されます。詳細については、P.14-10「**外部専用** ペリフェラルインタフェース」を参照して下さい。

---

### 注

プロセッサには、**ネスト型**ベクタ割り込みコントローラ (NVIC)、データウォッチポイントおよびトレース (DWT)、計装トレース マクロセル (ITM)、フラッシュパッチおよびブレークポイント (FPB)、メモリ保護ユニット (MPU) へのアクセス用の内部 PPB が組み込まれています。

---

## 14.2 AMBA 3 への準拠

プロセッサは、ウェイトされている転送中の制御情報の保持を除いて、AMBA 3 仕様を満たしています。AMBA 3 AHB-Lite プロトコルでは、スレーブがウェイト状態を要求している場合、マスタは次の場合を除いては転送タイプを変更できません。

- IDLE 転送 - マスタは、転送タイプを IDLE から NONSEQ へ変更することが許可されています。
- BUSY 転送、固定長バースト - マスタは、転送タイプを BUSY から SEQ へ変更することが許可されています。
- BUSY 転送、不定長バースト - マスタは、BUSY から他の転送タイプへ変更することが許可されています。

プロセッサは、待機転送時にアクセスタイプを SEQ または NONSEQ から IDLE へ変更することがあるため、上の定義を満たしていません。これによって、以前のアクセスがウェイト状態で完了待ちであるためにまだ発生していない予定されている転送は、結果的に取り消されます。これによって、ウェイト状態のシステムでプロセッサの割り込みレイテンシを短くし、パフォーマンスが向上します。

---

### 注

必要ならば、Cortex-M3 の外部に回路を実装して完全な準拠を実現できますが、その方法が必要になるのは、ペリフェラルがウェイトされている転送中にも制御情報の維持を必要とする場合のみです。このロジックを実装する方法の 1 つは、**HTRANS** が **LOW** の間、**HREADY** などの制御情報をマスクすることです。

---

14.3 ICode バスインタフェース

ICode インタフェースは、32 ビット AHB-Lite バスインタフェースです。コードメモリ空間 (0x00000000 ~ 0x1FFFFFFF) からの命令フェッチとベクタフェッチは、このバスを経由して行われます。

CM3Core 命令フェッチバスのみが、ICode インタフェースにアクセスし、最適なコードフェッチ パフォーマンスを実現できます。すべてのフェッチはワード幅です。ワードあたりのフェッチされる命令数は、実行されているコードと、メモリ内でのコードのアライメントによって異なります。この関係の説明を、表 14-1 に示します。

——— 注 ———

I-Code および D-Code の AHB バスインタフェース間での外部アービトレーションでは、D-Code の優先度を必ず I-Code よりも高くすることを強くお勧めします。

表 14-1 命令フェッチ

32 ビット命令 フェッチ [31:16]	32 ビット命令 フェッチ [15:0]	説明
Thumb[15:0]	Thumb[15:0]	すべての Thumb 命令はメモリ上でハーフワードアラインドなので、2 つの Thumb 命令が同時にフェッチされます。シーケンシャルなコードの場合、命令フェッチは 2 サイクルごとに実行されます。割り込みや分岐があると、命令フェッチは連続したサイクルで実行できます。
Thumb-2[31:16]	Thumb-2[15:0]	Thumb-2 コードがメモリ上でワードアラインドの場合、完全な Thumb-2 命令がサイクルごとにフェッチされます。
Thumb-2[15:0]	Thumb-2[31:16]	Thumb-2 コードがハーフワードアラインドの場合、最初の 32 ビットフェッチでは Thumb-2 命令の最初のハーフワードのみが返されます。2 番目のハーフワードをフェッチするには、2 回目のフェッチを実行する必要があります。このシナリオでは、実行されている命令によってはウェイトサイクル (CM3Core が命令を実行できないサイクル) が発生します。このレイテンシの追加サイクルは、ハーフワードアラインドの Thumb-2 命令が最初にフェッチされるときにのみ発生します。CM3Core には 3 エントリのフェッチバッファが搭載されているため、ハーフワードアラインドの Thumb-2 命令に含まれる上位のハーフワードはフェッチバッファに保持され、次のシーケンシャルな Thumb-2 命令に使用されます。

すべての ICode 命令フェッチはキャッシュ可でバッファ不可としてマークされ (HPROTI[3:2] = 2'b10)、また割り当て可能で共有不可としてマークされます (MEMATTRI = 2'b01)。これらの属性は固定です。MPU が搭載されている場合、ICode バスについて MPU の領域属性は無視されます。

HPROTI[0] は、フェッチの対象を示します。

- 0- 命令フェッチ



- 1- ベクタフェッチ

すべての ICode トランザクションは、ノンシーケンシャルに実行されます。

### 14.3.1 分岐状態信号

分岐状態信号 **BRCHSTAT** は、~~計装~~トレース マクロセル (ETM) インタフェースに出力され、パイプラインに分岐が含まれているかどうかを示します。例えば、プリフェッチユニットはこの信号を利用して、分岐がフェッチされようとしているときのプリフェッチを防止できます。分岐状態信号の詳細については、第 16 章「エンベデッドトレース マクロセルのインタフェース」を参照して下さい。

## 14.4 DCode バスインタフェース

DCode インタフェースは、32 ビット AHB-Lite バスです。コードメモリ空間 (0x00000000 ~ 0x1FFFFFFF) へのデータおよびデバッグアクセスは、この 32 ビット AHB-Lite バスを経由して行われます。コアデータアクセスは、デバッグアクセスよりも高い優先度で行われます。つまり、このバスに対してコアアクセスとデバッグアクセスが同時に行われると、コアアクセスが完了するまでデバッグアクセスは待たされます。

このインタフェースの制御ロジックは、アンアラインドなデータおよびデバッグアクセスを、2 つまたは 3 つの (アンアラインドアクセスのサイズとアライメントによって異なります) アラインドアクセスに変換します。このため、アンアラインドアクセスが完了するまで、以後のデータまたはデバッグアクセスはすべてストールします。

アンアラインドアクセスの説明については、P.14-11 「アクセスのアライメント」を参照して下さい。

---

### —— 注 ——

I-Code および D-Code の AHB バスインタフェース間での外部アービトレーションでは、D-Code の優先度を必ず I-Code よりも高くすることを強くお勧めします。

---

### 14.4.1 排他アクセス

DCode バスは排他アクセスをサポートしています。これは、**EXREQD** および **EXRESPD** という 2 つのサイドバンド信号を使用して行われます。詳細については、P.A-8 「DCode インタフェース」を参照して下さい。

### 14.4.2 メモリ属性

すべての DCode メモリアccessはキャッシュ可でバッファ不可としてマークされ (**HPROTD[3:2] = 2'b10**)、また割り当てで共有不可としてマークされます (**MEMATTRD = 2'b01**)。

これらの属性は固定です。MPU が搭載されている場合、DCode バスについて MPU の領域属性は無視されます。

## 14.5 システムインタフェース

システムインタフェースは、32ビットAHB-Liteバスです。システムメモリ空間(0x20000000 ~ 0xDFFFFFFF、0xE0100000 ~ 0xFFFFFFFF)への命令およびベクタフェッチと、データおよびデバッグアクセスは、このバスを経由して行われます。

このバスへ同時にアクセスが行われた場合、アービトレーションは優先順位の高いものから低いものへ、次の順序で行われます。

- データアクセス
- 命令およびベクタフェッチ
- デバッグ

システムバスインタフェースには、アンアラインドアクセス、FPB のリマップされたアクセス、ビットバンドアクセス、パイプライン化された命令フェッチを処理するための制御ロジックが含まれています。

### 14.5.1 アンアラインドアクセス

アンアラインドのデータおよびデバッグアクセスは、そのサイズとアライメントによって 2 つまたは 3 つのアラインドアクセスに変換されます。このため、アンアラインドアクセスが完了するまで、以後のアクセスはすべてストールします。アンアラインドアクセスの詳細については、P.14-11「アクセスのアライメント」を参照して下さい。

### 14.5.2 ビットバンドアクセス

ビットバンドエイリアス領域へのアクセスは、ビットバンド領域へのアクセスに変換されます。ビットバンド書き込みには 2 サイクルが必要です。これらの書き込みは読み出し - 変更 - 書き込み操作に変換されるため、ビットバンド書き込みアクセスが完了するまで、以後のアクセスはすべてストールします。ビットバンドアクセスの詳細については、P.14-13「ビットバンドアクセス」を参照して下さい。

### 14.5.3 フラッシュパッチのリマップ

システムメモリ空間へリマップされたコードメモリ空間へのアクセスでは、リマップのために 1 サイクルのペナルティが発生します。このため、フラッシュパッチアクセスが完了するまで、以後のアクセスはすべてストールします。フラッシュパッチの説明については、P.11-6「FPB」を参照して下さい。

### 14.5.4 排他アクセス

システムバスは排他アクセスをサポートしています。これは、**EXREQS** および **EXRESPS** という 2 つのサイドバンド信号を使用して行われます。詳細については、P.A-9「システムバスインタフェース」を参照して下さい。

## 14.5.5 メモリ属性

プロセッサは、**MEMATTRS** というサイドバンドバスを使用して、システムバスのメモリ属性を出力します。詳細については、P.14-15「メモリ属性」を参照して下さい。

## 14.5.6 パイプライン化された命令フェッチ

システムバスでクリーンなタイミングインタフェースを提供するため、このバスへの命令およびベクタフェッチ要求はレジスタ出力されます。その結果、システムバスからの命令フェッチには2サイクルを要し、1サイクルのレイテンシが追加されます。これは、システムバスからは連続した命令フェッチが不可能であることも意味します。

---

### —— 注 ——

ICode バスへの命令フェッチ要求はレジスタ出力されません。パフォーマンスが重要なコードは、ICode インタフェースから実行する必要があります。

---

## 14.6 コードバスの統合

一部のシステムでは、プロセッサコアの ICode および Dcode バスを組み合わせて単一の統合コードバスにすることが必要な場合があります。高速動作でこれをサポートするため、プロセッサに **DNOTITRANS** 入力があり、**HTRANSI** がアクティブのとき、**HTRANSD** ラインを抑えます。**DNOTITRANS** がアサートされている状態で、対応する単一サイクルアドレスフェーズで **HTRANSI** および **HTRANSD** が同時にアクティブになろうとすると、**HTRANSD** のみがアサートされます。ICode トランザクションはプロセッサ内でウェイトされます。言い換えれば、外部 ICode バスは強制的にアイドル状態になります。したがって、2 つの **HTRANS** 信号が同時にアクティブとなることは絶対にないため、バスマルチプレクサを非常に簡単なデバイスとして設計できます。

### 注

**DNOTITRANS** は静的な入力で、この動作を保証するために HIGH へ接続する必要があります。

外部の ICode/DCode バスマルチプレクサは、図 14-1 に示すように Cortex-M3 システムに統合できます。

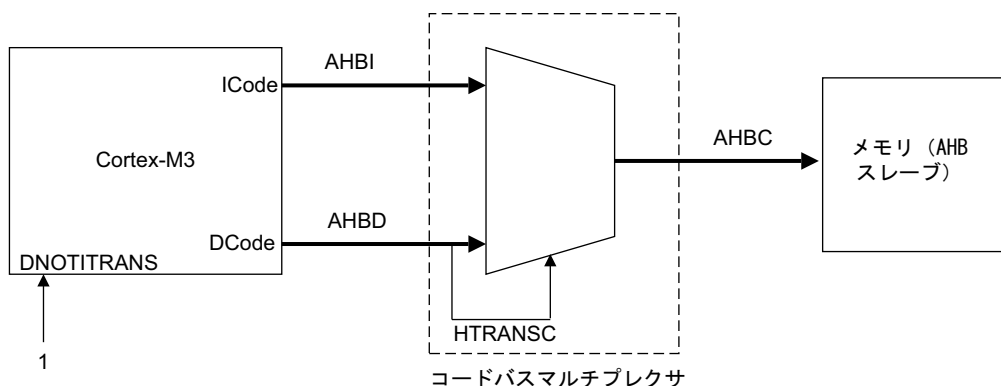


図 14-1 ICode/Dcode マルチプレクサ

## 14.7 外部専用ペリフェラル インタフェース

外部専用ペリフェラル インタフェースは APB (AMBA v2.0) バスです。外部ペリフェラルメモリ空間 0xE0040000 - 0xE00FFFFF へのデータおよびデバッグアクセスは、このバスを経由して行われます。このバスではウェイトステートはサポートされていません。TPIU および任意のベンダ固有コンポーネントはこのバスを使用します。コアデータアクセスはデバッグアクセスよりも優先度が高いため、このバスに対してコアアクセスとデバッグアクセスが同時に発生すると、コアアクセスが完了するまでデバッグアクセスは待たされます。このインタフェースでは、外部 **PPB** 空間をデコードするために必要なアドレスビットである、**PADDR** の [19:2] のみがサポートされています。

**PADDR31** は、このバスのサイドバンド信号として駆動されます。この信号が **HIGH** のとき、AHB-AP デバッグがマスタを要求していることを示します。この信号が **LOW** のとき、コアがマスタを要求していることを示します。

このバスへのアンアラインドアクセスはアーキテクチャの関係で予測不能であり、サポートされていません。プロセッサはコアからの元の **HADDR[1:0]** 要求を出力し、要求を複数のアラインドアクセスに変換する動作は行いません。

14.8    アクセスのアライメント

プロセッサは、ARMv6 モデルを使用したアンアラインドデータアクセスをサポートしています。DCode およびシステムバス インタフェースには、アンアラインドアクセスをアラインドアクセスに変換するロジックが組み込まれています。

アンアラインドデータアクセスの説明を、表 14-2 に示します。この表は、最初の列にアンアラインドアクセスの種類を示し、残りの列ではそのアクセスがどのように変換されるかを示しています。アンアラインドアクセスは、そのサイズとアライメントによって、2 つまたは 3 つのアラインドアクセスに変換されます。

表 14-2 バスマップパーのアンアラインドアクセス

アンアラインドアクセス		アラインドアクセス					
		サイクル 1		サイクル 2		サイクル 3	
サイズ	ADDR [1:0]	HSIZE	HADDR [1:0]	HSIZE	HADDR[1:0]	HSIZE	HADDR[1:0]
ハーフワード	00	ハーフワード	00	-	-	-	-
ハーフワード	01	バイト	01	バイト	10	-	-
ハーフワード	10	ハーフワード	10	-	-	-	-
ハーフワード	11	バイト	11	バイト	{(Addr+4)[31:2],2b00}	-	-
ワード	00	ワード	00	-	-	-	-
ワード	01	バイト	01	ハーフワード	10	バイト	{(Addr+4)[31:2],2b00}
ワード	10	ハーフワード	10	ハーフワード	{(Addr+4)[31:2],2b00}	-	-
ワード	11	バイト	11	ハーフワード	{(Addr+4)[31:2],2b00}	バイト	{(Addr+4)[31:2],2b10}

—— 注 ——

ビットバンドエイリアス領域にまたがるアンアラインドアクセスはビットバンド要求とは見なされず、そのアクセスはビットバンド領域にリマップされません。その代わりに、そのアクセスはビットバンドエイリアス領域へのハーフワードまたはバイトアクセスと見なされます。

## 14.9 領域にまたがるアンアラインドアクセス

CM3Core は ARMv6 アンアラインドアクセスをサポートしており、すべてのアクセスを単一のアンアラインドアクセスとして実行します。これらは、DCode およびシステムバス インタフェースにより、2 つまたはそれ以上のアラインドアクセスに変換されます。

---

### 注

---

Cortex-M3 のすべての外部アクセスはアラインドアクセスです。

---

アンアラインドのサポートは、単一ロード/ストア (LDR、STR) でのみ利用できます。ダブルロード/ストアはすでにワードアラインドのアクセスをサポートしていますが、他のアンアラインドアクセスを許可せず、試みられた場合はフォールトが生成されます。

メモリマップの境界にまたがるアンアラインドアクセスは、アーキテクチャの関係で予測不能です。プロセッサの動作は、以下に示すように境界に依存しています。

- DCode アクセスは領域内でラップアラウンドします。例えば、コード空間の最後のバイト (0x1FFFFFFF) に対して行われたアンアラインドのハーフワードアクセスは、DCode インタフェースによって、0x1FFFFFFF へのバイトアクセスと、0x00000000 へのバイトアクセスに変換されます。
- PPB 空間にまたがるシステムアクセスは、システム空間内でラップアラウンドしません。例えば、システム空間の最後のバイト (0xDFFFFFFF) に対して行われたアンアラインドのハーフワードアクセスは、システムインタフェースによって、0xDFFFFFFF へのバイトアクセスと、0xE0000000 へのバイトアクセスに変換されます。0xE0000000 は、システムバス上で有効なアドレスではありません。
- コード空間にまたがるシステムアクセスは、システム空間内でラップアラウンドしません。例えば、システム空間の最後のバイト (0xFFFFFFFF) に対して行われたアンアラインドのハーフワードアクセスは、システムインタフェースによって、0xFFFFFFFF へのバイトアクセスと、0x00000000 へのバイトアクセスに変換されます。0x00000000 は、システムバス上で有効なアドレスではありません。
- PPB 空間ではアンアラインドアクセスはサポートされていないため、PPB アクセスについては境界にまたがる例はありません。

ビットバンドエイリアス領域にまたがるアンアラインドアクセスも、アーキテクチャの関係で予測不能です。プロセッサはそのアクセスをビットバンドエイリアスのアドレスへのアクセスとして実行しますが、この動作はビットバンド操作とはなりません。例えば、0x21FFFFFF へのアンアラインドのハーフワードアクセスは、0x21FFFFFF へのバイトアクセスと、それに続く 0x22000000 (ビットバンドエイリアスの最初のバイト) へのバイトアクセスとして実行されます。

FPB のリテラルコンパレータに対して一致するアンアラインドロードはリマップされません。FPB は、アラインドアドレスのみをリマップします。



## 14.10 ビットバンドアクセス

システムバスインタフェースには、次のようにビットバンドアクセスを制御するロジックが組み込まれています。

- ビットバンドエイリアスのアドレスを、ビットバンド領域にリマップします。
- 読み出しの場合、読み出したバイトから要求されたビットを抽出し、読み出しデータの最下位ビット (LSB) としてコアへ返します。
- 書き込みの場合、書き込みを不可分な読み出し-変更-書き込み操作に変換します。

ビットバンドの詳細については、P.4-5「ビットバンド」を参照して下さい。

---

### 注

- CM3Core は、ビットバンド操作の実行中にシステムバスへのアクセスを試みない限り、ビットバンド操作中にストールすることはありません。
  - ビットバンドエイリアス領域へのビッグエンディアン アクセスはバイトサイズの必要があります。そうでない場合、そのアクセスは予測不能です。
-

## 14.11 ライトバッファ

データストア時にバスウェイト サイクルがプロセッサをストールさせることを防止するため、**DCode** およびシステムバスへのバッファされたストアは、1 エントリのライトバッファを経由します。ライトバッファがフルの場合、ライトバッファがドレインされるまで、バスへの以後のアクセスはストールします。バスがバッファされたストアのデータフェーズを待っている場合のみ、ライトバッファが使用されます。それ以外の場合、トランザクションはバス上で完了します。

**DMB** および **DSB** 命令は、完了する前にライトバッファのドレインを待ちます。**DMB/DSB** がライトバッファのドレインを待っている間に割り込みが発生した場合、**DMB/DSB** より後のオペコードは割り込みの完了時に実行されます。これは、割り込み処理が一種のメモリバリア操作となるためです。

14.12 メモリ属性

プロセッサは、**MEMATTR** というサイドバンドバスの追加により、システムバスのメモリ属性を出力します。

**MEMATTR[0]** と **HPROT[3:2]** との関係を、表 14-3 に示します。

表 14-3 メモリ属性

MEMATTR[0]	HPROT[3]	HPROT[2]	説明
0	0	0	ストロングリオーダー
0	0	1	デバイス
0	1	0	L1 キャッシュ可、L2 キャッシュ不可
1	0	0	無効
1	0	1	無効
1	1	0	キャッシュライトスルー (WT)、読み出し割り当て
0	1	1	キャッシュライトバック (WB)、読み出し / 書き込み割り当て
1	1	1	キャッシュライトバック (WB)、読み出し割り当て

14.13 AHB のタイミング特性

プロセッサは、マクロセル内にメモリを含みません。高いシステムパフォーマンスを達成し、実装者がメモリアーキテクチャの完全な柔軟性を得られるようにするため、プロセッサからのメモリ要求はレジスタ出力されず、AHB インタフェースへ直接送られます。

このため、Cortex-M3 AHB 出力はサイクルの約 50% まで有効で、AHB 入力はクロック期間の約 50% のセットアップ要件があります。

各インタフェースのタイミング特性の説明を、表 14-4 に示します。

表 14-4 インタフェースのタイミング特性

インタフェース	タイミング特性
ICODE	命令アドレスと制御信号は ALU から生成されるため、クロックサイクルの約 50% まで有効です。読み出しデータ (HRDATAI) および読み出し応答 (HRESPI) はプロセッサへ直接送られ、クロック期間の約 50% のセットアップを持ちます。
DCODE	コアデータおよびデバッグ要求はこのバスを経由して出力されます。データとデバッグの要求はどちらも、サイクルの比較的早い時間に出力され、これらはレジスタと、その後にある少量の組み合わせロジックにより生成されます。このバス上の要求は、ICODE バス上のものよりも余裕が多くなっています。書き込みデータ (HWDATAD) は ALU から直接出力され、クロックサイクルの約 50% まで有効です。読み出しデータ (HRDATAD) および読み出し応答 (HRESPD) はプロセッサへ直接送られ、クロック期間の約 50% のセットアップを持ちます。
SYSTEM	このバスからの命令フェッチは P.14-8 「パイプライン化された命令フェッチ」で説明されているようにパイプライン化され、このバスへのデータおよびデバッグ要求はサイクルの早い時点で出力されるため、このバス上の要求は ICODE バスに出力されるものよりも余裕が多くなっています。書き込みデータ (HWDATAS) は ALU から直接出力され、クロックサイクルの約 50% まで有効です。読み出しデータ (HRDATAS) および読み出し応答 (HRESPS) はプロセッサへ直接送られ、クロック期間の約 50% のセットアップを持ちます。
PPB	このバスへのデータおよびデバッグ要求はサイクルの早い時点で出力されるため、このバスへの要求は ICODE バスへ出力されるものよりも余裕が多くなっています。書き込みデータ (PWDATA) は ALU から直接出力され、クロックサイクルの約 50% まで有効です。読み出しデータ (PRDATA) はプロセッサへ直接送られ、クロック期間の約 50% のセットアップを持ちます。

# 第 15 章

## エンベデッドトレース マクロセル

本章では、エンベデッドトレース マクロセル(ETM)について説明します。本章は以下のセクションから構成されています。

- *ETM* について : P.15-2
- データトレース : P.15-7
- *ETM* リソース : P.15-8
- トレース出力 : P.15-11
- *ETM* のアーキテクチャ : P.15-12
- *ETM* のプログラマモデル : P.15-16

## 15.1 ETM について

ETM はオプションのデバッグコンポーネントで、プログラム実行の再構築を可能にします。ETM は高速で低消費電力のデバッグツールとして設計されており、命令トレースのみをサポートします。これによって、占有面積を最小にし、ゲートの数も削減しています。

### 15.1.1 ETM のブロック図

ETM のブロック図、および ETM とトレースポート インタフェースユニット (TPIU) とのインタフェースを、P.15-3 図 15-1 に示します。

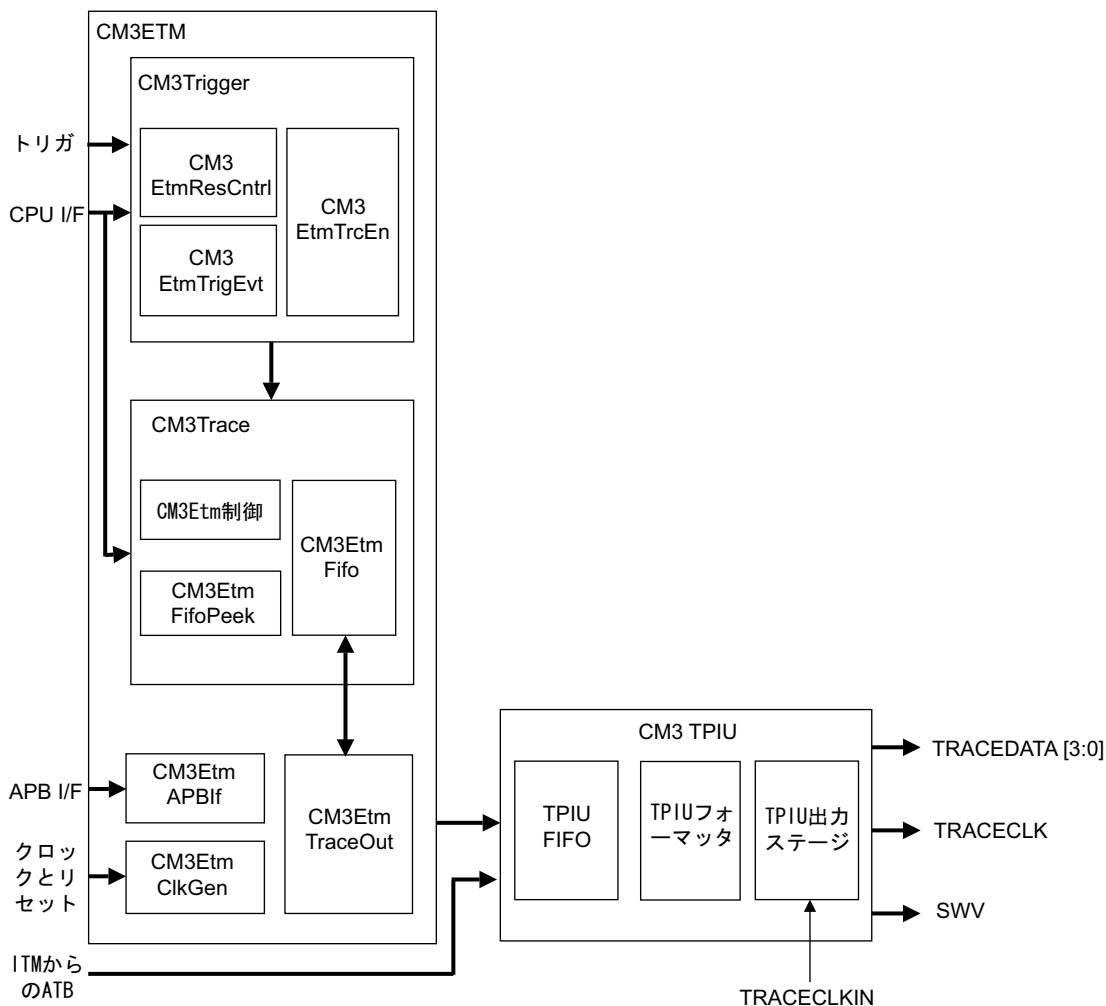


図 15-1 ETM のブロック図

### 15.1.2 ETM の入力と出力

このセクションでは、ETM の入力と出力について説明します。

- ETM コアインタフェース。詳細については、P.15-4 表 15-1 を参照して下さい。
- その他の構成入力。詳細については、P.15-4 表 15-2 を参照して下さい。
- トレースポート信号。詳細については、P.15-4 表 15-2 を参照して下さい。
- その他の信号。詳細については、P.15-5 表 15-4 を参照して下さい。

- クロックとリセット。詳細については、P.15-6 表 15-5 を参照して下さい。
- アドバンスド ペリフェラルバス (APB) インタフェース信号。詳細については、P.15-6 表 15-6 を参照して下さい。

表 15-1 ETM コアインタフェースの入力と出力

名前	説明	認証する信号	方向
ETMIA[31:1]	コアの命令のアドレスバス	ETMIVALID	入力
ETMIVALID	現在の命令データは命令を示しています。	-	入力
ETMDVALID	現在の命令データは命令を示しています。	-	入力
ETMICCFAIL	命令で条件コードが失敗しました。	ETMIVALID	入力
ETMIBRANCH	命令は分岐ターゲットです。	ETMIVALID	入力
ETMIINDBR	命令は間接分岐ターゲットです。	ETMIBRANCH	入力
ETMFLUSH	次の命令の前に PC が変更されます。	-	入力
ETMISTALL	コアによって信号を出された最後の命令がまだ実行を開始されていないことを示します。	-	入力
ETMFINDBR	間接操作によって PC が変更されました。	ETMFLUSH	入力
ETMINTSTAT[2:0]	例外の開始と終了	-	入力
ETMINTNUM[8:0]	例外のタイプ	ETMINTSTAT	入力
ETMCANCEL	例外はキャンセル例外です。	ETMINTSTAT	入力
COREHALT	コアがホールドしています。	-	入力
DWTMATCH [3:0]	データウォッチポイントおよびトレース (DWT) トリガユニットが、アドレス、データ、制御バスに現在提示されている条件に一致したことを示します。	-	入力
DWTINOTD[3:0]	DWT トリガユニットが PC の値 (セット) またはデータアドレス (クリア) の比較を実行していることを示します。	-	入力

表 15-2 その他の構成入力

名前	説明	方向	クロックドメイン
NIDEN	非侵襲デバッグが有効です。	入力	HCLK
EXTIN[1:0]	外部入力リソース	入力	HCLK



表 15-2 その他の構成入力

名前	説明	方向	クロックドメイン
<b>MAXEXTIN[1:0]</b>	サポートされている最大の外部入力	入力	<b>HCLK</b>
<b>CGBYPASS</b>	アーキテクチャ上のクロックゲートセルをバイパスします。	入力	<b>CLK</b>
<b>FIFOFULLEN</b>	<b>ETMFIFOFULL</b> を有効にします。	入力	<b>CLK</b>

## 注

ETM への **EXTIN** 入力の 1 つをコアからの **LOCKUP** 出力で駆動し、ロックアップ条件が発生したときにトリガしたり、トレースキャプチャを停止したりできます。**EXTIN** 入力は ETM 内で同期されていません。これらの入力が ETM クロックで駆動されない場合、ETM の外部で同期する必要があります。

表 15-3 トレースポート信号

名前	説明	方向	クロックドメイン
<b>ATDATAM[7:0]</b>	8 ビットのトレースデータ	出力	<b>HCLK</b>
<b>ATVALIDM</b>	<b>ATDATA</b> が有効	出力	<b>HCLK</b>
<b>ATIDM[6:0]</b>	トレースソース ID	出力	<b>HCLK</b>
<b>ATREADYM</b>	トレースポートが <b>ATDATA</b> のデータを受け付け可能なことを示します。	入力	<b>HCLK</b>
<b>AFREADYM</b>	ETM FIFO が空であることを示します。	出力	<b>HCLK</b>

表 15-4 その他の信号

名前	説明	方向	クロックドメイン
<b>FIFOPEEK[9:0]</b>	検証専用	出力	<b>HCLK</b>
<b>ETMPWRUP</b>	ETM の電力がオンであることを示します。	出力	<b>HCLK</b>
<b>ETMTRIGOUT</b>	トリガが発生したことを示すステータス信号	出力	<b>HCLK</b>
<b>ETMDBGRQ</b>	コアへのデバッグ要求	出力	<b>HCLK</b>
<b>ETMEN</b>	ETM トレースポートが許可されています。	出力	<b>HCLK</b>

表 15-5 クロックとリセット

名前	説明	方向
HCLK	ETM ロジックのクロックで、Cortex-M3 と同じ <b>FCLK</b> に接続する必要があります。	入力
HRESETn	<b>HCLK</b> ドメインのパワーオン リセット。コアの <b>HCLK</b> リセット ( <b>SYSRESETn</b> ) と同じにすることはできません。	入力

表 15-6 APB インタフェース信号

名前	説明	方向	クロックドメイン
PSEL	APB デバイスの選択	入力	CLK
PENABLE	APB 制御信号	入力	CLK
PADDR[11:2]	APB アドレスバス	入力	CLK
PWRITE	APB 転送の方向 (!Read/Write)	入力	CLK
PWDATA[31:0]	APB 書き込みデータバス	入力	CLK
PRDATA[31:0]	APB 読み出しデータバス	出力	CLK

## 15.2 データトレース

Cortex-M3 システムは、データウォッチポイントおよびトレース (DWT)、および計装トレースマクロセル (ITM) コンポーネントを使用して、低帯域幅のデータトレースを実行できます。少ないピン数で命令トレースをサポートするため、データトレースは ETM に組み込まれていません。これによって、トリガを行うリソースが簡素化され、ETM のゲート数が大幅に削減されています。

ETM がプロセッサに実装されるとき、ITM および ETM という 2 つのトレースソースはどちらも TPIU にフィードされ、TPIU で結合されて、通常はトレースポート経由で出力されます。DWT は、FIFO オーバフローの問題の可能性があります、集中したデータトレース、またはグローバルなデータトレースのどちらも提供できます。TPIU は、単一コアの Cortex-M3 システムの要件に最適化されています。

15.3 ETM リソース

ETM はデータトレース情報を生成しないため、帯域幅が低く、トリガ機能を比較的簡素化できます。これは、ETM に次の機能が含まれていないことを意味します。

- 内部コンパレータ
- カウンタ
- シーケンサ

Cortex-M3 リソースの一覧を、表 15-7 に示します。

表 15-7 Cortex-M3 リソース

機能	Cortex-M3 ETM に存在するか
アーキテクチャのバージョン	ETMv3.4
アドレスコンパレータ ペア	0
データコンパレータ	0
コンテキスト ID コンパレータ	0
MMD	0
カウンタ	0
シーケンサ	いいえ
開始 / 停止ブロック	はい
エンベデッド ICE コンパレータ	4
外部入力	2
外部出力	0
拡張外部入力	0
拡張外部入力セクタ	0
FIFOFULL	はい
FIFOFULL のレベル設定	はい
分岐ブロードキャスト	はい
ASIC 制御レジスタ	いいえ
データ抑制	いいえ

表 15-7 Cortex-M3 リソース

機能	Cortex-M3 ETM に存在するか
ソフトウェアからレジスタへのアクセス	はい
読み出し可能なレジスタ	はい
FIFO サイズ	24 バイト
最小ポートサイズ	8 バイト
最大ポートサイズ	8 バイト
通常のポートサイズ	-
通常のハーフレートクロック /1:1	はい - 非同期
Demux ポートモード	-
Demux ハーフレートクロック /1:2	いいえ
Mux ポートモード /2:1	いいえ
1:4 ポートモード	いいえ
動的ポートモード（ストールを含む）	いいえ - 非同期ポートモードでサポート
CPRT データ	いいえ
PC を最初にロード	いいえ
フェッチ比較	いいえ
データロードのトレース	いいえ

### 15.3.1 定期的な同期

ETM は、トレース 1024 バイトごとの固定周期で同期パケットを生成します。

### 15.3.2 データおよび命令アドレス比較のリソース

DWT は、データバス上で 4 つのアドレスコンパレータを提供し、デバッグ機能を実現しています。DWT ユニット内では、一致によりトリガされる機能を指定でき、これらの機能の 1 つは ETM 一致入力生成です。これらの入力、エンベデッド *In Circuit Emulator*(ICE) コンパレータ入力として ETM へ送られます。

ひとつの DWT リソースが ETM イベントをトリガし、同時に同じイベントから直接計装トレースを生成することもできます。

また、4 つの DWT コンパレータを個別に構成し、実行 PC との比較により PC 比較リソースへの ETM によるアクセスを許可できます。これらの入力、エンベデッド ICE コンパレータ入力として ETM へ送られます。

---

**注**

DWT コンパレータを PC コンパレータとして使用すると、使用可能なデータアドレス比較の数が減少します。

---

See DWT ユニットの詳細については、P.11-13 「DWT」を参照して下さい。

### 外部入力

2 つの外部入力 (ETMEXTIN[1:0]) により、追加のオンチップ IP で ETM 用のトリガ / イネーブル信号を生成できます。

### 開始 / 停止ブロック

開始 / 停止ブロックは、ETM へのエンベデッド ICE 入力を使用して開始 / 停止動作を制御します。DWT はこれらの入力を制御します。

## 15.3.3 FIFO 機能

FIFO のサイズは 24 バイトです。

FIFO が特定の深さに達したときコアをストールできるように、FIFOFULL 出力が用意されています。一般的なアプリケーションでは、ほとんどの場合コアのストールは許容できないものですが、この機構によって 100% のトレースを行い、ストールを行わない実行で得られた部分的なトレースとの比較が可能になります。

## 15.4 トレース出力

ETM 出力データは 8 ビットずつ、コアのクロック速度で出力されます。トレースポートのサイズやモードの変更はサポートされていません。TPIU はトレース出力をチップから出力します。この出力は、**AMBA** トレースバス (ATB) プロトコル互換です。

AFVALID 機能はサポートされていないため、トレースポートは ETM FIFO からデータをフラッシュできません。ただし、8 ビット ATB ポートにより FIFO は常にドレインされるため、AFVALID は不要になっています。

Cortex-M3 システムには、ETM および ITM とともに使用されるために最適化された TPIU が搭載されています。この TPIU は、追加のトレースソースをサポートしていません。ただし、TPIU をより複雑なバージョンに置き換え、さらにトレース設備を追加すれば、トレースソースを追加できます。

---

### 注

トレース ID レジスタと出力は、複数のトレースソースを使用するシステム用に提供されています。

---

TPIU は、フォーマットされたトレース出力プロトコルを使用します。このため、**TRACECTL** 信号用の追加ピンは必要ありません

ETM からのトレース出力は、コアクロックに同期しています。トレースポートインターフェースには非同期 FIFO が組み込まれています。ETM をマルチコアシステムに統合する場合は、非同期 ATB ブリッジを使用することが必要な可能性があります。

15.5 ETM のアーキテクチャ

ETM は命令専用 ETM で、ARM ETM アーキテクチャ v3.4 を実装しています。このアーキテクチャは、ARM ETM アーキテクチャ仕様に基づいたものです。詳細については、『ARM エンベデッドトレースマクロセル アーキテクチャ仕様書』を参照して下さい。

すべての Thumb-2 命令は単一の命令としてトレースされます。IT 命令以後の命令は、通常の条件付き命令としてトレースされます。デコンプレッサは、IT 命令を参照する必要はありません。

15.5.1 再始動可能な命令

ARMv7-M アーキテクチャは、例外によって中断された LSM 命令を再始動できます。ETM は、例外によって中断された命令をトレースするために、命令がキャンセルされたことを通知します。例外からの復帰時に、ETM は同じ命令を、再始動または再開が行われたかどうかにかかわらず再度トレースします。

15.5.2 例外からの復帰

ETM は、例外からの復帰を、トレースストリームに明示的に示します。これは、例外からの復帰機能がデータ依存の方法でエンコードされ、単純な分岐とは異なる動作を行うためです。

パケットのエンコードにより、例外からの復帰が示されます。このエンコードを、図 15-2 に示します。

7	6	5	4	3	2	1	0
0	1	1	1	0	1	1	0

図 15-2 例外からの復帰を示すパケットのエンコード

新しい、より優先度の高い例外がスタックポップを横取りした場合、例外ハンドラへの分岐は、最後の命令がキャンセルされたことを示す必要があります。これは、例外からの復帰パケットはキャンセルされたが、例外からの復帰命令はキャンセルされなかったことを示します。例外からの復帰パケットが存在している場合、前の命令が完了しなかったことを意味します。

15.5.3 例外のトレース

例外をトレースするため、分岐パケットにはオプションのフィールドが追加されています。この追加フィールドは、例外に関する情報を示しています。通常分岐パケットは 1 ～ 5 バイトのトレースデータにエンコードされますが、例外への分岐は次のようにエンコードされます。

- 2 ～ 5 バイトのアドレス



- 1 ～ 2 バイトの例外

例外のマッピングは、最も多い例外が 1 バイトでエンコードされるように設計されています。ETM の例外トレースのマッピングマッピングを、表 15-8 に示します。

表 15-8 例外トレースのマッピング

バイト数	例外	ETMINTNUM	トレースされる 値
1 バイトの例外	なし	-	0
1 バイトの例外	IRQ1	17	1
1 バイトの例外	IRQ2	18	2
1 バイトの例外	IRQ3	19	3
1 バイトの例外	IRQ4	20	4
1 バイトの例外	IRQ5	21	5
1 バイトの例外	IRQ6	22	6
1 バイトの例外	IRQ7	23	7
1 バイトの例外	IRQ0	16	8
1 バイトの例外	用法フォールト	6	9
1 バイトの例外	NMI	2	10
1 バイトの例外	SVC	11	11
1 バイトの例外	DebugMon	12	12
1 バイトの例外	MemManage	4	13
1 バイトの例外	PendSV	14	14
1 バイトの例外	SysTick	15	15
2 バイトの例外	予約	8	16
2 バイトの例外	リセット	0	17
2 バイトの例外	予約	10	18
2 バイトの例外	HardFault	3	19
2 バイトの例外	予約	9	20
2 バイトの例外	BusFault	5	21

表 15-8 例外トレースのマッピング

バイト数	例外	ETMINTNUM	トレースされる値
2 バイトの例外	予約	7	22
2 バイトの例外	予約	13	23
2 バイトの例外	IRQ8	24	24
2 バイトの例外	IRQ9	25	25
2 バイトの例外	IRQ10	26	26
.	.	.	.
.	.	.	.
.	.	.	.
2 バイトの例外	IRQ239	512	512

例外パケットを持つ完全な分岐を、図 15-3 に示します。

7	6	5	4	3	2	1	0	
C	Addr[6:1]						1	アドレスのバイト0
C	E/ Addr[13]	Addr[12:7]						アドレスのバイト1 (オプション)
C	E/ Addr[20]	Addr[19:14]						アドレスのバイト2 (オプション)
C	E/ Addr[27]	Addr[26:21]						アドレスのバイト3 (オプション)
C	E	0	1	Addr[31:28]				アドレスのバイト4 (オプション)
C	T2EE	Canc	Excp[3:0]				NS	例外情報バイト0
C	0	SBZ	Excp[8:4]					例外情報バイト1 (オプション)

図 15-3 分岐パケットの例外のエンコード

最後のアドレスバイトのビット [7:6] は、アドレスフィールドの終わりを示すため 0b01 にセットされます。このフィールドの後に例外のデータが置かれます。例外のバイト 0 のビット [7] は、その後に 2 番目の例外バイトが続く場合は 1 にセットされます。例外が存在せず、アドレスビット [6:1] のみを変更された場合、単一のバイトが使用されます。例外が存在する場合、アドレスを示すために最低 2 バイトが使用されます。

例外ハンドラ開始の直前にトレースを停止すると、例外を開始するまで ETM は許可状態に維持されます。これによって、ETM は分岐アドレス、例外のタイプ、再開情報をトレースできます。

15.6 ETM のプログラマモデル

ETM のプログラマモデルの詳細については、『*ARM エンベデッドトレース マクロセル アーキテクチャ仕様書*』を参照して下さい。このセクションでは、ETM プログラマモデルの実装固有の機能について定義します。

15.6.1 アドバンスト ペリフェラルバス インタフェース

ETM には、APB スレーブインタフェースが組み込まれており、ETM レジスタの読み出しと書き込みを実行できます。このインタフェースは、プロセッサのクロックに同期しています。コアおよび外部デバッグインタフェースは、シリアルワイヤデバッグポート/JTAG デバッグポート(SW-DP/JTAG-DP)を経由してこのインタフェースにアクセスできます。

15.6.2 ETM レジスタの一覧

ETM レジスタの一覧を、表 15-9 に示します。完全な詳細については、『*ARM エンベデッドトレース マクロセル アーキテクチャ仕様書*』を参照して下さい。

表 15-9 ETM レジスタ

名前	タイプ	アドレス	存在	説明
ETM 制御	読み出し / 書き込み	0xE0041000	はい	説明については、P. 15-19 を参照して下さい。
コンフィギュレーションコード	読み出し専用	0xE0041004	はい	説明については、P. 15-19 を参照して下さい。
トリガイメント	書き込み専用	0xE0041008	はい	トリガを制御するイベントを定義します。
ASIC 制御	-	0xE004100C	いいえ	-
ETM ステータス	読み出し専用 または読み出し / 書き込み	0xE0041010	はい	トレースおよびトリガロジックの現在のステータスに関する情報を提供します。
システム構成	読み出し専用	0xE0041014	はい	説明については、P. 15-19 を参照して下さい。
TraceEnable	-	0xE0041018、 0xE004101C	いいえ	-
TraceEnable イベント	書き込み専用	0xE0041020	はい	TraceEnable 許可イベントを記述します。
TraceEnable 制御 1	書き込み専用	0xE0041024	はい	説明については、P. 15-20 を参照して下さい。
FIFOFULL 領域	書き込み専用	0xE0041028	いいえ	-

表 15-9 ETM レジスタ

名前	タイプ	アドレス	存在	説明
FIFOFULL レベル	書き込み専用 または読み出し / 書き込み	0xE004102C	はい	FIFO がフルと見なされるしきい値を保持しています。この値を下回った場合、FIFO はフルと見なされます。
ViewData	-	0xE0041030 ～ 0xE004103C	いいえ	-
アドレスコンパレータ	-	0xE0041040 ～ 0xE004113C	いいえ	-
カウンタ	-	0xE0041140 ～ 0xE0041157C	いいえ	-
シーケンサ	-	0xE0041180 ～ 0xE0041194、 0xE0041198	いいえ	-
外部出力	-	0xE00411A0 ～ 0xE00411AC	いいえ	-
CID コンパレータ	-	0xE00411B0 ～ 0xE00411BC	いいえ	-
実装固有	-	0xE00411C0 ～ 0xE00411DC	いいえ	すべて RAZ。書き込みは無視されます。
同期化周波数	読み出し専用	0xE00411E0	はい	0x00000400 として読み出されます。
ETM ID	読み出し専用	0xE00411E4	はい	説明については、P. 15-20 を参照して下さい。
コンフィギュレーションコード 拡張機能	読み出し専用	0xE00411E8	はい	説明については、P. 15-20 を参照して下さい。
拡張外部入力セクタ	-	0xE00411EC	いいえ	拡張外部入力は実装されていません。
TraceEnable 開始 / 停止エンベ デッド ICE	読み出し / 書き込み	0xE00411F0	はい	ビット [19:16] は、停止リソースとして使用するよう E-ICE 入力を構成します。ビット [3:0] は、開始リソースとして使用するよう E-ICE 入力を構成します。
エンベデッド ICE 動作制御	-	0xE00411F4	いいえ	エンベデッド ICE 入力はデフォルトの動作を使用します。
CoreSight トレース ID	読み出し / 書き込み	0xE0041200	はい	通常のもので実装されます。
OS 保存 / 復元	-	0xE0041304 ～ 0xE0041308	いいえ	OS 保存 / 復元は実装されていません。RAZ、書き込みは無視されます。

表 15-9 ETM レジスタ

名前	タイプ	アドレス	存在	説明
電力オフ ステータスレジスタ (Power Down Status Register)	読み出し専用	0xE0041314	はい	説明については、P. 15-20 を参照して下さい。
ITMISCIN	読み出し専用	0xE0041EE0	はい	[1:0] を EXTIN[1:0] に、[4] を COREHALT にセットします。
ITTRIGOUT	書き込み専用	0xE0041EE8	はい	[0] を TRIGGER にセットします。
ITATBCTR2	読み出し専用	0xE0041EF0	はい	[0] を ATREADY にセットします。
ITATBCTR0	書き込み専用	0xE0041EF8	はい	[0] を ATVALID にセットします。
統合モード制御	読み出し / 書き込み	0xE0041F00	はい	通常のものとして実装されます。
クレームタグ	読み出し / 書き込み	0xE0041FA0 ~ 0xE0041FA4	はい	4 ビットのクレームタグを実装します。
ロックアクセス	書き込み専用	0xE0041FB0 ~ 0xE0041FB4	はい	通常のものとして実装されます。
認証ステータス	読み出し専用	0xE0041FB8	はい	通常のものとして実装されます。
デバイスタイプ	読み出し専用	0xE0041FCC	はい	リセット時の値 : 0x13
ペリフェラル ID 4	読み出し専用	0xE0041FD0	はい	0x04
ペリフェラル ID 5	読み出し専用	0xE0041FD4	はい	0x00
ペリフェラル ID 6	読み出し専用	0xE0041FD8	はい	0x00
ペリフェラル ID 7	読み出し専用	0xE0041FDC	はい	0x00
ペリフェラル ID 0	読み出し専用	0xE0041FE0	はい	0x24
ペリフェラル ID 1	読み出し専用	0xE0041FE4	はい	0xB9
ペリフェラル ID 2	読み出し専用	0xE0041FE8	はい	0x1B
ペリフェラル ID 3	読み出し専用	0xE0041FEC	はい	0x00
コンポーネント ID 0	読み出し専用	0xE0041FF0	はい	0x0D
コンポーネント ID 1	読み出し専用	0xE0041FF4	はい	0x90
コンポーネント ID 2	読み出し専用	0xE0041FF8	はい	0x05
コンポーネント ID 3	読み出し専用	0xE0041FFC	はい	0xB1

### 15.6.3 ETM レジスタの説明

以下のセクションでは、ETM レジスタのいくつかについて追加説明を行います。詳細については、『ARM エンベデッドトレースマクロセルアーキテクチャ仕様書』を参照して下さい。

#### ETM 制御レジスタ

ETM 制御レジスタは、ETM の一般的な動作、例えばトレースを許可するかどうかなどを制御します。

リセット時の値：0x00002411

実装されているビット：[21]、[17:16]、[13]、[11:4]、[0]

他のビットはすべて RAZ で、書き込みは無視されます。

#### コンフィギュレーションコード レジスタ

ETM コンフィギュレーションコード レジスタによって、デバッガは ETM の実装固有の構成を読み出すことが出来ます。

リセット時の値：0x8C800000

ビット [22:20] は 0 に固定されており、ASIC からは供給されません。ビット [18:17] は **MAXEXTIN[1:0]** 入力バスによって供給され、MAXEXTIN と 2 (EXTIN の数) のどちらか低い値として読み出されます。これは、次の内容を示しています。

- ソフトウェアアクセスがサポートされています。
- トレース開始/停止ブロックが存在します。
- CID コンパレータは存在しません。
- FIFOFULL ロジックが存在します。
- 外部出力は存在しません。
- 0 ~ 2 の外部入力 (MAXEXTIN によって制御される) が存在します。
- シーケンサは存在しません。
- カウンタは存在しません。
- MMD は存在しません。
- データコンパレータは存在しません。
- アドレスコンパレータ ペアは存在しません。

#### システム構成レジスタ

システム構成レジスタは、ASIC でサポートされている ETM 機能を示しています。

リセット時の値：0x00020D09

ビット [11:10] は通常のもので実装されます。ビット [9] および [2:0] は 4'b0001 に固定されています。

## TraceEnable 制御 1 レジスタ

TraceEnable 制御 1 レジスタは、TraceEnable を構成するレジスタの 1 つです。

ビット [25] のみが実装されています。このビットは開始 / 停止リソース制御トレーシングを制御します。

## ETM ID レジスタ

ETM ID レジスタは ETM のアーキテクチャバージョンを持ち、ETM のプログラマモデルを正確に定義しています。

リセット時の値 : 0x4114F241

この値は次の内容を示しています。

- ARM 実装者
- 特殊分岐のエンコード（各バイトのビット [7:6] に影響します）。
- Thumb-2 がサポートされています。
- コアファミリは別の場所に示されています。
- ETMv3.4
- 実装リビジョン 1

## コンフィギュレーションコード拡張レジスタ

コンフィギュレーションコード拡張レジスタは、ETM コンフィギュレーションコードの追加ビットを保持しています。これらのビットは、拡張外部入力を記述しています。

リセット時の値 : 0x00018800

このレジスタは次の内容を示しています。

- 開始 / 停止ブロックは、エンベデッドインサーキット エミュレータ (ICE) 入力を使用します。
- 4 つのエンベデッド ICE 入力が存在します。
- データ比較はサポートされていません。
- すべてのレジスタが読み出し可能です。
- 拡張外部入力はサポートされていません。

## 電力オフ ステータスレジスタ

電力オフ ステータスレジスタ (PDSR) は、ETM の電力がオンになっているかどうかを示します。

リセット時の値 : 0x00000001



ビット [0] のみが実装されています。これは、ETM デバッグ電力ドメインの電力がオンになっているかどうかを示します。

- 0 = ETM デバッグ電力ドメインの電力がオンになっていません。
- 1 = ETM デバッグ電力ドメインの電力がオンになっています。

---

注

ETM の電力がオンになっていない場合、ETM レジスタにはアクセスできません。

---



# 第 16 章

## エンベデッドトレース マクロセルのインタフェース

本章では、エンベデッドトレース マクロセルエンベデッドトレース マクロセル

- P.16-2 「*ETM* インタフェースについて」
- P.16-3 「*CPU* の *ETM* インタフェースポートの説明」
- P.16-6 「分岐ステータスインタフェース」

## 16.1 ETM インタフェースについて

ETM インタフェースは、ETM からプロセッサへの簡単な接続に使用できます。このインタフェースには、命令トレースから ETM へのチャンネルが用意されています。

## 16.2 CPU の ETM インタフェースポートの説明

プロセッサには、ETM が命令実行シーケンスを判定するためのポートがあります。これらのポートの説明を、表 16-1 に示します。

表 16-1 ETM インタフェースポート

ポート名	方向	認証する信号	説明
<b>ETMIVALID</b>	出力	認証なし	実行中の命令は有効です。オペコードが実行の最初のサイクルに入ったことを示します。
<b>ETMIBRANCH</b>	出力	<b>ETMIVALID</b>	オペコードは分岐ターゲットです。現在のコードがプログラムカウンタ (PC) 変更イベント (分岐、割り込み処理) のデスティネーションであることを示します。
<b>ETMIINDBR</b>	出力	<b>ETMIBRANCH</b>	オペコードの分岐ターゲットは間接です。現在のオペコードが分岐ターゲットで、PC のコンテンツからそのデスティネーションを推定できないことを示します。例えば、LSU、レジスタ移動、割り込み処理などが挙げられます。
<b>ETMDVALID</b>	出力	認証なし	データウォッチポイントおよびブレース (DWT) から見た現在のデータアドレスが、このサイクルについて有効であることを示します。
<b>ETMICCFAIL</b>	出力	<b>ETMIVALID</b>	オペコードの条件コードの失敗または成功を示します。現在のオペコードが条件付き実行チェックで失敗したか成功したかを示します。オペコードは、条件付き分岐の場合、または IT ブロック内に見つかった他のオペコードの場合に、条件付きで実行されます。
<b>ETMINTSTAT[2:0]</b>	出力	認証なし	割り込みステータス。現在のサイクルの割り込みステータスを次のように示します。 000- ステータスなし 001- 割り込み開始 010- 割り込み終了 011- 割り込み復帰 100- ベクタフェッチおよびスタックプッシュ。 <b>ETMINTSTAT</b> 開始 / 復帰は、新規割り込みコンテキストの最初のサイクルでアサートされます。 <b>ETMIVALID</b> なしで終了します。
<b>ETMINTNUM[8:0]</b>	出力	<b>ETMINTSTAT</b>	割り込み番号。現在の実行コンテキストの割り込み番号を示します。

表 16-1 ETM インタフェースポート

ポート名	方向	認証する信号	説明
<b>ETMIA[31:1]</b>	出力	認証なし	<p>命令アドレス。実行されているオペコード、または最後に実行されたオペコードの現在のフェッチアドレスを示します。コンテキストは、次の信号を調べることで判定できます。</p> <p><b>ETMIVALID</b>  <b>HALTED</b>  <b>SLEEPING.</b></p> <p>ETM は、<b>ETMIVALID</b> がアサートされたとき、このネットを調べます。DWT は、PC サンプルとバスウォッチ用にこのネットを調べます。</p>
<b>ETMFOLD</b>	出力	<b>ETMIVALID</b>	<p>オペコードはフォールドされます。このサイクルで IT オペコードがフォールドされたことを示します。現在の（16 ビット）オペコードと IT 命令（16 ビット）を超えて PC が進行します。これは ETMIA に影響します。</p>
<b>ETMFLUSH</b>	出力	認証なし	<p>PC イベントのフラッシュマーカ。PC を変更するオペコードが実行されたか、または割り込みプッシュ/ポップが開始されたことを示します。ETM はこのコントロールを使用して、<b>ETMIBRANCH</b> イベントの準備のために未解決のパケットを完了できます。</p>
<b>ETMFINDBR</b>	出力	<b>ETMFLUSH</b>	<p>フラッシュは間接的です。PC がフラッシュヒントのデスティネーションを推定できないことを示します。</p>
<b>ETMCANCEL</b>	出力	認証なし	<p>現在実行されているオペコードがキャンセルされました。割り込みされたオペコードは、この実行コンテキストに復帰したときに再始動または続行されます。これには、次のオペコードが含まれます。</p> <p>LDR/STR  LDRD/STRD  LDM/STM  U/SMULL  MLA  U/SDIV  MSR  CPSID</p>

表 16-1 ETM インタフェースポート

ポート名	方向	認証する信号	説明
ETMISTALL	出力	認証なし	コアによって信号を出された最後の命令がまだ実行を開始されていないことを示します。 <b>ETMICANCEL</b> が <b>ETMISTALL</b> と共にアサートされた場合、ストールした命令が実行されなかったことと、前の命令がキャンセルされたことを示します。
ETMTRIGGER[3:0]	出力	認証なし	DWT からの出力トリガ。4 つの DWT コンパレータそれぞれについて 1 ビットが使用されています。
ETMTRIGINOTD[3:0]	出力	認証なし	ETM が命令またはデーター一致でトリガされるかどうかを示す出力です。

16.3 分岐ステータスインタフェース

分岐ステータス信号 **BRCHSTAT** は、デコードにあるオペコードについてのフェッチ時情報を示します。デコード時分岐には、それに暗黙的に対応するフェッチサイクルがあるため、**BRCHSTAT** は問題のメモリランザクションに関する唯一のインシデントです。実行時分岐にはマルチサイクルの **BRCHSTAT** が含まれている可能性があり、これは実行中の前のオペコードのストールに依存します。信号の機能を、表 16-2 に示します。

表 16-2 分岐ステータス信号の機能

名前	方向	説明
BRCHSTAT	出力	0000 = ヒントなし 0001 = デコード時の後方への条件付き分岐 <sup>a</sup> 0010 = デコード時の条件付き分岐 <sup>b</sup> 0011 = 実行時の条件付き分岐 <sup>c</sup> 0100 = デコード時の無条件分岐 <sup>d</sup> 0101 = 実行時の無条件分岐 <sup>e</sup> 0110 = 予約 0111 = 予約 1000 = デコード時の条件付き分岐が、 <b>IHTRANS</b> の b0001 または b0010 後のサイクルに取得されます。 <sup>f</sup>

- a. IT ブロック内ではない T1B1 後方または T2B0 後方。
- b. IT ブロック内ではない T1B1 前方または T2B0 前方。IT ブロック内の T1B1 または T2B0。IT ブロック内の T1B2、T1MOV LR、または T2BLX LR。
- c. T1CBZ。IT ブロックの T1BLX !LR。IT ブロックの T1MOV !LR。
- d. IT ブロック内ではない T1B2、T2BL、T1MOV LR、または T2BLX LR。
- e. IT ブロック内ではない T1BLX !LR。IT ブロック内ではない T1MOV !LR。
- f. b0001 および b0010 後のサイクルにのみアサートされます。

——— 注 ———

- T1B1 および T2B0 は条件付き分岐です。
- T1B2 および T2BL は無条件分岐です。
- T1CBZ は 0 と比較して分岐です。

——— 注 ———

- エンコード b1000 は、条件付きデコード分岐で分岐が行われた後のサイクルにのみアサートされます。これはレジスタ出力なので、メモリコントローラのアドレスのマルチプレクサを駆動するために使用できます。
- ALU のレジスタをベースとした分岐と、LSU の PC を変更するオペコードは、IT ブロック内を除いて b0101 にエンコードされます。IT ブロック内では b0011 にエンコードされます。



- b0101 エンコードのマルチサイクル LSU は、実行中のフェッチを抑制します。
- 実行エンコードは、デコードのマルチサイクル期間示されます。

条件付き後方分岐が行われた場合と、行われなかった場合について図 16-1 および図 16-2 に示します。分岐は、オペコードのデコードフェーズにおいて投機的に行われます。分岐ターゲットは、ハーフワードのアンアラインドの 16 ビットオペコードです。

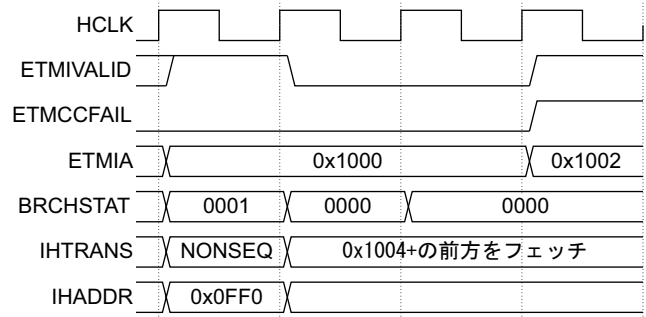


図 16-1 条件付き後方分岐が行われなかった場合

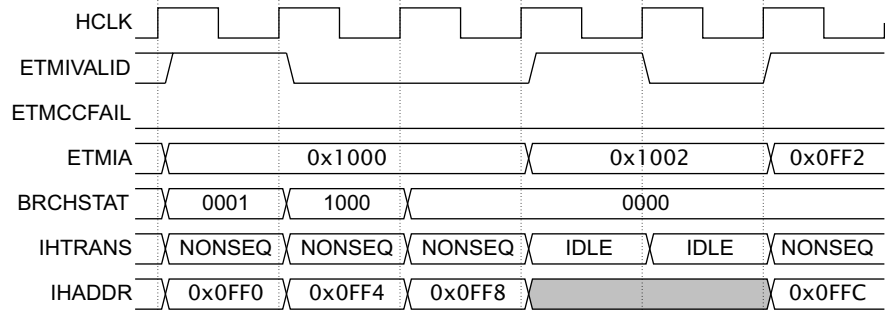


図 16-2 条件付き後方分岐が行われた場合

#### 注

**HADDRICore** および **HTRANSICore** は、プロセッサからのアドレスおよびトランザクション要求信号で、外部 Cortex-M3 インタフェースの信号ではありません。

条件付き前方分岐が行われた場合と、行われなかった場合について P.16-8 図 16-3 および P.16-8 図 16-4 に示します。分岐は、オペコードのデコードフェーズにおいて投機的に行われます。分岐ターゲットは、ハーフワードのアンアラインドの 16 ビットオペコードです。

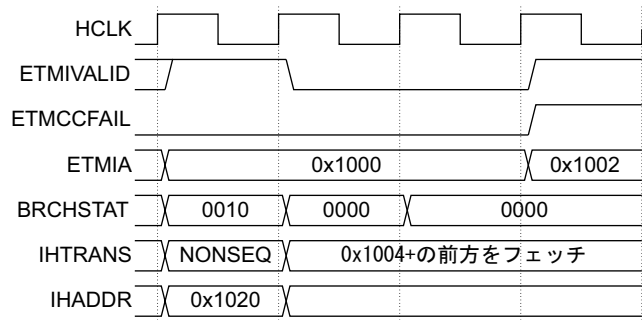


図 16-3 条件付き前方分岐が行われなかった場合

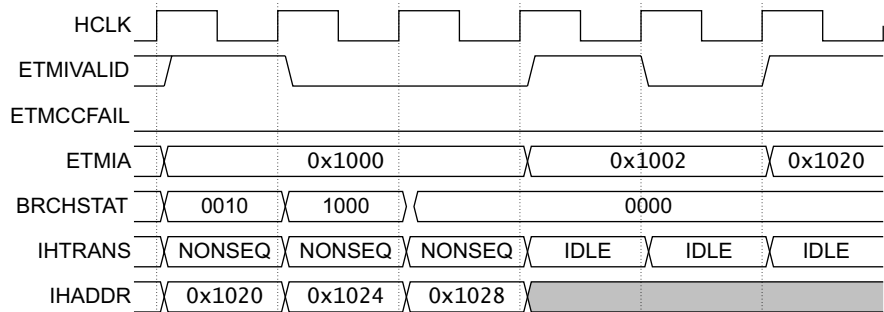


図 16-4 条件付き前方分岐が行われた場合

図 16-5 および P.16-9 図 16-6 は、前のオペコードの実行フェーズにおけるこのサイクルの無条件分岐を、パイプラインがストールした場合としない場合について示しています。分岐は、オペコードのデコードフェーズに行われます。分岐ターゲットは、アラインされた 32 ビットオペコードです。

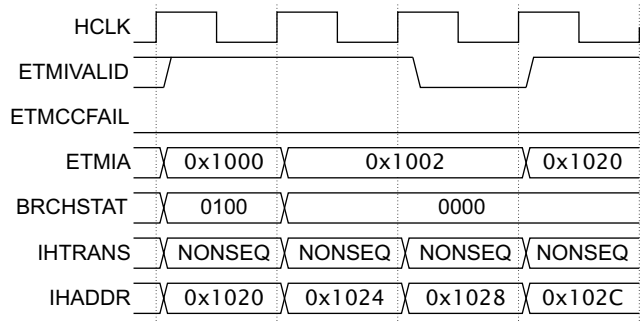


図 16-5 パイプラインがストールしない場合の無条件分岐

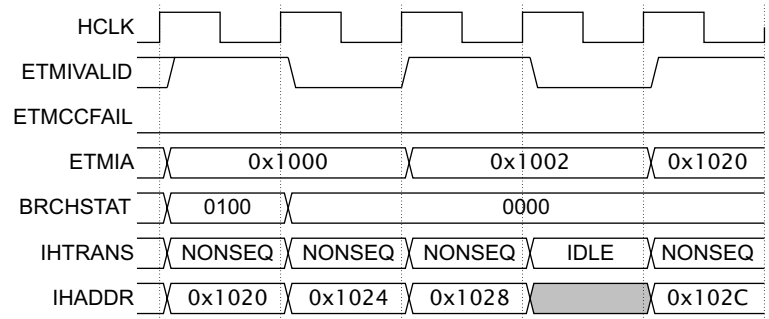


図 16-6 パイプラインがストールした場合の無条件分岐

次のオペコードでの無条件分岐を、図 16-7 および図 16-8 に示します。分岐は、オペコードの実行フェーズに行われます。分岐ターゲットは、アラインドおよびアンアラインドの 32 ビット ALU オペコードです。

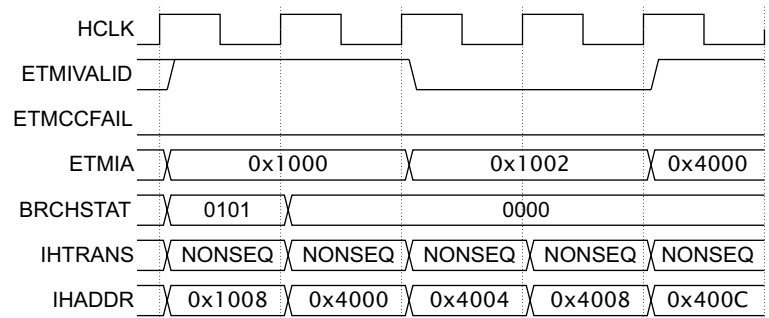


図 16-7 実行段での無条件分岐、飛び先がアラインドな命令

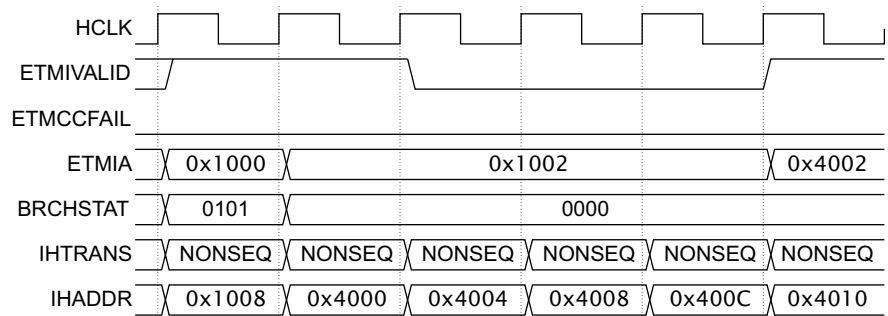


図 16-8 実行段での無条件分岐、飛び先がアンアラインドな命令

オペコードシーケンスの例を、表 16-3 に示します。

表 16-3 オペコードシーケンスの例

実行サイクル	フェッチアドレス	オペコード
1	0x1020	ADD r1,#1
2	0x1022	LDR r3,[r4]
3	0x1024	ADD r2,#3
4	0x1026	CMP r3,r2
5	0x1028	BEQ = Target1
6	0x1040	CMP r1,r2
7	0x1042	ITE // folded
8	0x1044	LDR r3,[r4,r1]
9	0x1046	LDR r3,[r4,r2] // not taken
10	0x1048	ADD r6,r3
11	0x104A	NOP // folded
12	0x104C	BX r14
13	0x0FC4	CMP
14	0x0FC6	BEQ = Target2 // not taken
15	0x0FC8	BX r5

P.16-11図 16-9のオペコードシーケンス例について、タイミングシーケンスを表 16-3に示します。

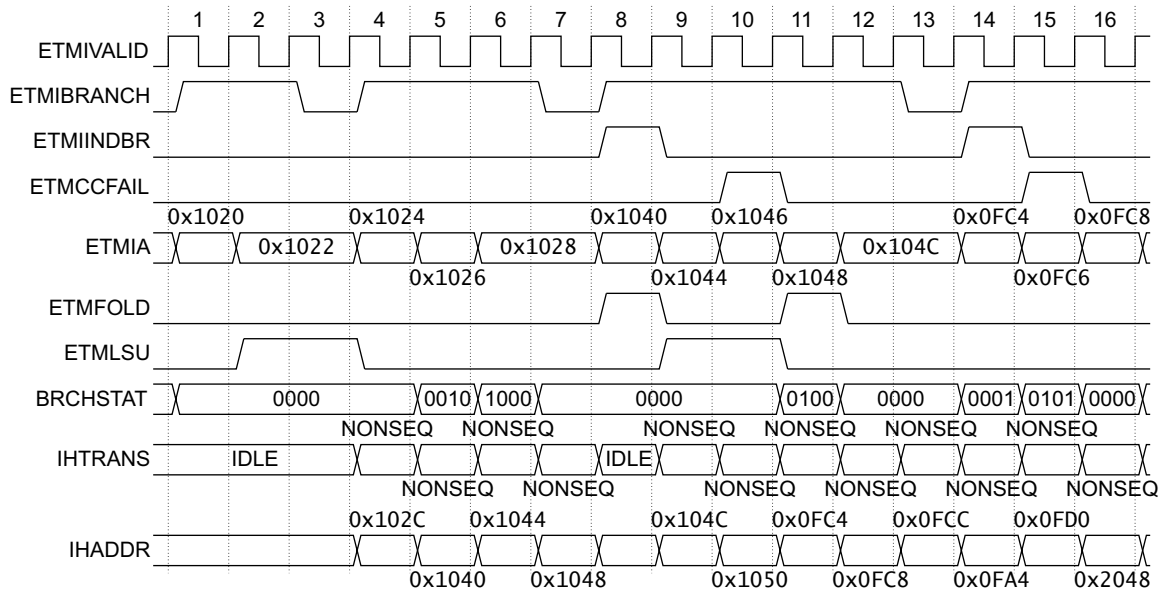


図 16-9 オペコードシーケンスの例



## 第 17 章

# AHB トレースマクロセル インタフェース

本章では、AHB トレースマクロセル インタフェースについて説明します。本章は以下のセクションから構成されています。

- P.17-2 「AHB トレースマクロセル インタフェースについて」
- P.17-3 「CPU AHB トレースマクロセル インタフェースのポートの説明」

## 17.1 AHB トレースマクロセル インタフェースについて

AHB トレースマクロセル (HTM) インタフェースを使用すると、AHB トレースマクロセルをプロセッサへ簡単に接続できます。このインタフェースは、HTM へのデータトレース用チャンネルとして動作します。

HTM インタフェースを使用するには、実装の前に CM3Defs.v の HTM\_PORT\_ENABLE 定義をコメント解除しておく必要があります。また、HTM が HTM ポートを許可してトレースデータを供給できるようにするため、TRCENA は 1 にセットする必要があります。



## 17.2 CPU AHB トレースマクロセル インタフェースのポートの説明

AHB インタフェースのポートの一覧を、表 17-1 に示します。

表 17-1 AHB インタフェースのポート

ポート名	方向	説明
HTMDHADDR[31:0]	出力	32 ビットのアドレス
HTMDHTRANS[1:0]	出力	現在のデータ転送のタイプを示す出力。この出力の値は IDLE、NONSEQUENTIAL、SEQUENTIAL のいずれかです。
HTMDHSIZE[1:0]	出力	アクセスのサイズを示します。値は 8、16、32 ビットのいずれかです。
HTMDHBURST[2:0]	出力	転送がバーストの一部かどうかを示す出力
HTMDHPROT[3:0]	出力	アクセスに関する情報を提供します。
HTMDHWDATA[31:0]	出力	32 ビットの書き込みデータバス
HTMDHWRITE	出力	書き込み、読み出しではありません。
HTMDHRDATA[31:0]	出力	読み出しデータバス
HTMDHREADY	出力	HIGH の場合は、バス上で転送が完了したことを示します。転送を延長するために、この信号が LOW に駆動されます。
HTMDHRESP[1:0]	出力	転送応答ステータス。OKAY または ERROR です。
HTMDHADDR[31:0]	出力	32 ビットのアドレス
HTMDHTRANS[1:0]	出力	現在のデータ転送のタイプを示す出力。この出力の値は IDLE、NONSEQUENTIAL、SEQUENTIAL のいずれかです。



## 第 18 章

# 命令のタイミング

本章では、プロセッサの命令のタイミングについて説明します。本章は以下のセクションから構成されています。

- 命令のタイミングについて: P.18-2
- プロセッサ命令のタイミング: P.18-3
- ロード/ストアのタイミング: P.18-7

## 18.1 命令のタイミングについて

本章に記載されているタイミング情報は、命令間の相互動作に加え、各命令をカバーしています。また、タイミングに影響を与える要因についての情報も含まれています。

タイミングについて考える際は、システムアーキテクチャが果たす役割を理解することが重要になります。すべての命令はフェッチされる必要があります、すべてのロード/ストアはシステムに送出される必要があります。これらの要因については、意図されているシステム設計およびタイミングへの影響とともに、本章で説明されています。

## 18.2 プロセッサ命令のタイミング

ARMv7-M アーキテクチャでサポートされている Thumb-2 サブセットを、表 18-1 に示します。ここでは、命令ストリームの相互動作がどのようにタイミングに影響するのかを説明するための注釈を含む、サイクル情報が示されています。遅いメモリからコードを実行する場合のような、システムの影響についても考慮されています。

表 18-1 命令のタイミング

命令タイプ	サイズ	サイクル数	説明
データ処理	16	1 (PC がデステーションの場合 は +P <sup>a</sup> )	ADC、ADD、AND、ASR、BIC、CMN、CMP、CPY、EOR、LSL、LSR、MOV、MUL、MVN、NEG、ORR、ROR、SBC、SUB、TST、REV、REVH、REVSH、SXTB、SXTH、UXTB、UXTH。MUL は 1 サイクルです。
分岐命令	16	1+P <sup>a</sup>	B<cond>、B、BL、BX、BLX。イミディエートを使用する BLX は含まれません。分岐が発生した場合、パイプラインがリロードされます (2 サイクル追加されます)。
単一ロード/ストア	16	2 <sup>b</sup> (PC がデステーションの場合 は +P <sup>a</sup> )	LDR、LDRB、LDRH、LDRSB、LDRSH、STR、STRB、STRH、および T バリエーション
複数ロード/ストア	16	1+N <sup>b</sup> (PC がロードされる場合は +P <sup>a</sup> )	LDMIA、POP、PUSH、STMIA
例外生成	16	-	BKPT は、デバッグが許可されている場合はデバッグで停止し、デバッグが禁止されている場合はフォールトが発生します。 SVC は SVCall ハンドラにフォールトを発生させます (詳細については、『ARMv7-M アーキテクチャ仕様書』を参照して下さい)。
イミディエートを使用するデータ操作	32	1 (PC がデステーションの場合 は +P <sup>a</sup> )	ADC{S}、ADD{S}、CMN、RSB{S}、SBC{S}、SUB{S}、CMP、AND{S}、TST、BIC{S}、EOR{S}、TEQ、ORR{S}、MOV{S}、ORN{S}、MVN{S}
大きなイミディエートを使用するデータ操作	32	1	MOVW、MOVT、ADDW、SUBW。MOVW および MOVT は、16 ビットイミディエートを持ちます (このため、メモリからのリテラルロードの代わりに使用できます)。ADDW および SUBW は、12 ビットイミディエートを持ちます (このため、メモリからのリテラルロードのうち、多くの操作の代わりに使用できます)。
ビットフィールド操作	32	1	BFI、BFC、UBFX、SBFX。これらはビット単位の操作で、ビット内での位置およびサイズの制御を許可します。C/C++ の (構造体の) ビットフィールド操作や、その他多くの比較および一部の AND/OR 代入式をサポートします。
3 つのレジスタを使用するデータ操作	32	1 (PC がデステーションの場合 は +P <sup>a</sup> )	ADC{S}、ADD{S}、CMN、RSB{S}、SBC{S}、SUB{S}、CMP、AND{S}、TST、BIC{S}、EOR{S}、TEQ、ORR{S}、MOV{S}、ORN{S}、MVN{S}。PKxxx 命令は含まれません。

表 18-1 命令のタイミング

命令タイプ	サイズ	サイクル数	説明
シフト操作	32	1	ASR{S}、LSL{S}、LSR{S}、ROR{S}、RRX{S}
その他	32	1	REV、REVH、REVSH、RBIT、CLZ、SXTB、SXTH、UXTB、UXTH。対応する ARMv6 16 ビット命令と同じ拡張機能命令です。
テーブル分岐	16	4+P <sup>a</sup>	switch/case を使用する場合のテーブル分岐。これらはシフトしてから分岐する LDR です。
乗算	32	1 または 2	MUL、MLA、MLS。MUL は 1 サイクルで、MLA と MLS は 2 サイクルです。
結果が 64 ビットの乗算	32	3-7 <sup>c</sup>	UMULL、SMULL、UMLAL、SMLAL。サイクル数は入力サイズによって異なります。つまり、ABS（入力）< 64K の場合は早く終了します。
ロード/ストアアドレッシング	32	-	PC +/- imm12、Rbase + imm12、Rbase +/- imm8 のフォーマット、およびシフトを含む調整が行われたレジスタをサポートしています。T バリエントは特権モードで使用されます。
単一ロード/ストア	32	2 <sup>b</sup> （PC がデステーションの場合には +P <sup>a</sup> ）	LDR、LDRB、LDRSB、LDRH、LDRSH、STR、STRB、STRH、および T バリエント。PLD および PLI は両方ともヒントなので、NOP として動作します。
複数ロード/ストア	32	1+N <sup>b</sup> （PC がロードされる場合には +P <sup>a</sup> ）	STM、LDM、LDRD、STRD
特殊なロード/ストア	32	1+N <sup>b</sup>	LDREX、STREX、LDREXB、LDREXH、STREXB、STREXH、CLREX。ローカルモニタ (IMP DEF) がいない場合、これらはフォールトを発生します。LDREXD および STREXD は、このプロファイルに含まれていません。
分岐命令	32	1+P <sup>a</sup>	B、BL、B<cond>。状態が常に変化するため、BLX (1) は含まれません。BXJ は含まれません。
システム	32	1-2	MSR(2) および MRS(2) は MSR/MRS に置き換わる命令で、機能が拡張されています。これらは、他のスタックおよびステータスレジスタにもアクセスします。CPSIE/CPSID の 32 ビット形式はサポートされていません。RFE および SRS は含まれません。
システム	16	1-2	CPSIE および CPSID は MSR(2) 命令の短縮バージョンで、標準の Thumb-2 エンコーディングを使用しますが、i および f の使用のみが許可され、a の使用は許可されません。
拡張 32	32	1	NOP および YIELD（ヒント NOP）。MRS(1)、MSR(1)、SUBS（PC 復帰リンク）は含まれません。
結合分岐	16	1+P <sup>a</sup>	CBZ
拡張	16	0-1 <sup>d</sup>	IT および NOP（YIELD を含む）

表 18-1 命令のタイミング

命令タイプ	サイズ	サイクル数	説明
除算	32	2-12 <sup>e</sup>	SDIV および UDIV。32/32 は、符号付きおよび符号なしの値を除算して、32 ビット商の結果を生成します（余りは減算により導出できるため、生成されません）。被除数と除数のサイズが近い場合、これにより時間が短縮されます。
スリープ	32	1+W <sup>f</sup>	WFI、WFE、SEV は、スリープの動作を制御するヒント NOP 命令のクラスに含まれます。
バリア	16	1+B <sup>g</sup>	ISB、DSB、DMB はバリア命令で、次の命令が実行される前に特定のアクションが発生することを保証します。
飽和	32	1	SSAT および USAT は、レジスタ上で飽和を実行します。これらは 3 つのタスクを実行します。シフトを使用して値を正規化し、選択されたビット位置からのオーバーフロー（Q 値）をテストし、xPSR の Q ビットをセットします。飽和は、選択されたサイズに対して最大の符号なし値、または最大 / 最小の符号付き値を参照します。

- 分岐は、命令のために 1 サイクル、続いてターゲット命令のパイプラインリロードのために 1 サイクルを使用します。分岐が行われなかった場合は、合計で 1 サイクルです。イミディエート値を使用した分岐が行われた場合、通常は 1 サイクルのパイプラインリロード（合計 2 サイクル）です。レジスタオペランドを使用した分岐が行われた場合、通常は 2 サイクルのパイプラインリロード（合計 3 サイクル）です。アンアラインドな 32 ビット命令へ分岐する場合と、低速のメモリへアクセスする場合、パイプラインのリロード時間は長くなります。低速なシステムでは、プリロードを許可するための分岐ヒントをコードバスへ発行できます。これにより、遅いメモリに対する分岐ターゲットペナルティを低減することができますが、ここに示されたものより少なくすることはできません。
- 一般的には、ロード / ストア命令は最初のアクセスは 2 サイクル、それ以降の各アクセスは 1 サイクルで実行されます。イミディエートオフセットを使用するストアの場合は、1 サイクルで実行されます。
- UMULL/SMULL/UMLAL/SMLAL は、ソースの値のサイズに応じて早期終了を行います。これらは、割り込み可能（破棄 / 再始動）で、レイテンシは最長でも 1 サイクルです。MLAL バージョンは 4 ~ 7 サイクルで、MULL バージョンは 3 ~ 5 サイクルでそれぞれ実行されます。MLAL の場合、符号付きバージョンは符号なしバージョンより 1 サイクル長くなります。
- IT 命令はフォールド可能です。
- DIV タイミングは、被除数と除数に依存します。DIV は、割り込み可能（破棄 / 再始動）で、レイテンシは最長でも 1 サイクルです。被除数と除数がほぼ同じサイズの場合、除算は少ないサイクル数で終了します。除数が被除数より長い場合や除数が 0 の場合、時間は最短になります。除数が 0 の場合は（フォールトではなく）0 が返されますが、このケースをキャッチするためにデバッグトラックを利用することができます。
- スリープは命令自体に 1 サイクルを使用し、それにスリープのサイクルが必要な分だけ加わります。イベントがパスした場合、WFE は 1 サイクルのみを使用します。WFI の開始と同時に割り込みが発生して保留された場合を除き、WFI は通常 1 サイクルより多くを使用します。
- ISB は 1 サイクルで実行されます（分岐として動作）。データがライトバッファまたは LSU で保留されない限り、DMB と DSB も 1 サイクルで実行されます。バリア中に割り込みが発生した場合は、破棄 / 再始動されます。

サイクル数の記号は、次の意味を表します。

- P = パイプラインのリロード
- N = エレメントの数

- W = スリープウェイト
- B = バリアクリアランス

一般に、P.18-3 表 18-1 に示されているように、各命令は実行を開始するために 1 サイクル（1 コアクロック）を使用します。フェッチのストールのために、追加のサイクルが使用されることがあります。



## 18.3 ロード/ストアのタイミング

このセクションでは、どのように命令をペアにするのが最適かについて説明します。これにより、より時間が短縮されます。

- **STR Rx,[Ry,#imm]** は、常に 1 サイクルです。これは、アドレス生成は最初のサイクルで実行され、データストアは次の命令の実行と同時に実行されるためです。ストアバッファがフルのときに、そのストアバッファに対してストアが行われた場合、次の命令はストアが完了するまで遅延します。ストアがコードセグメントなど、ストアバッファ以外に対して実行され、そのトランザクションがストールした場合は、完了前に別のロード/ストア操作が実行された場合のみ、タイミングへの影響が発生します。
- **LDR Rx!,[any]** は、通常はパイプライン処理されません。つまり、ベース更新ロードには通常、最低でも 2 サイクルが使用されます（ストールした場合はより長くなります）。しかし、次の命令がレジスタからの読み出しを必要としない場合、ロードは 1 サイクルに短縮されます。レジスタを読み出さない命令には、**CMP**、**TST**、**NOP**、および **IT** 制御の命令が実行されない場合が含まれます。
- **LDR PC,[any]** は、常にブロッキング操作です。これは、ロードに 2 サイクル、パイプラインのリロードに 3 サイクルが、最低でも使用されることを意味します。つまり、最低でも 5 サイクルが使用されます（ロードまたはフェッチでストールが発生した場合はより長くなります）。
- フェッチユニットとの競合が原因で、**LDR Rx,[PC,#imm]** が 1 サイクル長くなることがあります。
- **TBB** および **TBH** もブロッキング操作です。これらの命令では、ロードに 2 サイクル、加算に 1 サイクル、パイプラインのリロードに 3 サイクルが、最低でも使用されます。つまり、最低でも 6 サイクルが使用されます（ロードまたはフェッチでストールが発生した場合はより長くなります）。
- 可能な場合、すべての **LDR** はパイプライン処理されます。つまり、もし次の命令が **LDR** または非ベース更新 **STR** で、最初の **LDR** のデスティネーションが次の命令のアドレスを計算するために使用されないときは、次の命令のコストから 1 サイクルが差し引かれます。このため、**LDR** の後に **STR** が続くと、**LDR** によってロードされた内容が **STR** によって書き込まれます。より多くの数の **LDR** を一緒にパイプライン処理することができます。最適化された例をいくつか挙げます。
  - **LDR R0,[R1]; LDR R1,[R2]** - 通常は合計 3 サイクル
  - **LDR R0,[R1,R2]; STR R0,[R3,#20]** - 通常は合計 3 サイクル
  - **LDR R0,[R1,R2]; STR R1,[R3,R2]** - 通常は合計 3 サイクル
  - **LDR R0,[R1,R5]; LDR R1,[R2]; LDR R2,[R3,#4]** - 通常は合計 4 サイクル

- レジスタオフセットが使用されている **STR** の後でパイプライン処理を行うことはできません。**STR** へのパイプライン処理は **LDR** の後のみ可能で、ストアの後でパイプライン処理を行うことはできません。ストアバッファ（ビットバンド、データセグメント、アンアラインド）があるため、ストールしている **STR** の場合でも通常は2サイクルしか使用しません。
- LDREX** と **STREX** は、**LDR** と全く同様にパイプライン処理することができます。これは、**STREX** は **LDR** のように扱われるためで、**STREX** は **LDR** の説明と同じようにパイプライン処理することができます。**LDREX** も同様に、**LDR** と全く同じように扱われるので、パイプライン処理することができます。
- LDRD** や **STRD** を、先行する命令や後に続く命令と一緒にパイプライン処理することはできません。しかし、2 つのワードは一緒にパイプライン処理されます。このため、ストールが発生しない場合は3サイクルです。
- LDM** や **STM** を、先行する命令や後に続く命令と一緒にパイプライン処理することはできません。しかし、最初のエレメント以降のすべてのエレメントと一緒にパイプライン処理されます。このため、ストールが発生しない場合、3 エレメントの **LDM** は  $2 + 1 + 1$  または5サイクルで実行されます。同様に、8 エレメントのストアは、ストールが発生しない場合は9サイクルで実行されます。割り込みがあった場合は、**LDM** および **STM** 命令は復帰の際に中断したところから再開されます。再開動作により、最初のエレメントを開始するために、1 または2サイクルが加えられます。
- アンアラインドなワードまたはハーフワードのロード/ストアには、ペナルティサイクルが追加されます。バイトアラインドのハーフワードロード/ストアは2つのバイトとして操作が実行されるため、実行時間に1サイクルが追加されます。ハーフワードアラインドのワードロード/ストアは2つのハーフワードとして操作が実行されるため、実行時間に1サイクルが追加されます。バイトアラインドのワードロード/ストアは、バイト、ハーフワード、バイトの操作に分割されて実行されるため、実行時間に2サイクルが追加されます。メモリがストールした場合、これらの数字は増加します。**STR** または **STRH** では、ストアバッファがあるため、プロセッサへの遅延は発生しません。

# 第 19 章

## AC 特性

本章では、プロセッサのタイミングパラメータについて説明します。本章は以下のセクションから構成されています。

- P.19-2 「プロセッサのタイミングパラメータ」

## 19.1 プロセッサのタイミングパラメータ

このセクションでは、プロセッサの入出力ポートのタイミングパラメータについて説明します。P.19-3 表 19-1 ～ P.19-10 表 19-14 は、SoC に適用される各プロセッサ信号のタイミングパラメータまたは各種制約条件による遅延の最大値を、パーセンテージで示しています。入力遅延と出力遅延の列は、各信号に関して SoC に入力されるプロセッサのクロックサイクルに対する最大時間と最小時間の割合を示しています。

## 19.2 プロセッサのタイミングパラメータ

### 19.2.1 入力ポートのタイミングパラメータ

その他の入力ポートのタイミングパラメータを、表 19-1 に示します。

表 19-1 その他の入力ポートのタイミングパラメータ

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	10%	<b>PORESETn</b>
クロックの不確定分	10%	<b>SYSRESETn</b>
クロックの不確定分	50%	<b>BIGEND</b>
クロックの不確定分	50%	<b>EDBGRQ</b>
クロックの不確定分	50%	<b>STCLK</b>
クロックの不確定分	50%	<b>STCALIB[25:0]</b>
クロックの不確定分	50%	<b>RXEV</b>
クロックの不確定分	50%	<b>AUXFAULT[31:0]</b>
クロックの不確定分	50%	<b>IFLUSH</b>
クロックの不確定分	50%	<b>PPBLOCK[5:0]</b>

割り込み入力ポートのタイミングパラメータを、表 19-2 に示します。

表 19-2 割り込み入力ポートのタイミングパラメータ

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	50%	<b>INTISR[239:0]</b>
クロックの不確定分	50%	<b>INTNMI</b>
クロックの不確定分	20%	<b>VECTADDR[9:0]</b>
クロックの不確定分	20%	<b>VECTADDREN</b>

アドバンスド ハイパフォーマンス バス (AHB) ポートのタイミングパラメータを、表 19-3 に示します。

表 19-3 AHB 入力ポートのタイミングパラメータ

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	10%	DNOTITRANS
クロックの不確定分	50%	HRDATAI[31:0]
クロックの不確定分	50%	HREADYI
クロックの不確定分	50%	HRESPI[1:0]
クロックの不確定分	50%	HRDATAD[31:0]
クロックの不確定分	50%	HREADYD
クロックの不確定分	50%	HRESPD[1:0]
クロックの不確定分	50%	EXRESPD
クロックの不確定分	50%	HRDATAS[31:0]
クロックの不確定分	50%	HREADYS
クロックの不確定分	50%	HRESPS[1:0]
クロックの不確定分	50%	EXRESPS

専用ペリフェラルバス (PPB) ポートのタイミングパラメータを、表 19-4 に示します。

表 19-4 PPB 入力ポートのタイミングパラメータ

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	50%	PRDATA[31:0]

デバッグ入力ポートのタイミングパラメータを、表 19-5 に示します。

**表 19-5 デバッグ入力ポートのタイミングパラメータ**

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	10%	<b>nTRST</b>
クロックの不確定分	50%	<b>SWDITMS</b>
クロックの不確定分	50%	<b>TDI</b>
クロックの不確定分	50%	<b>DAPRESETn</b>
クロックの不確定分	50%	<b>DAPSEL</b>
クロックの不確定分	50%	<b>DAPEN</b>
クロックの不確定分	50%	<b>DAPENABLE</b>
クロックの不確定分	50%	<b>DAPCLKEN</b>
クロックの不確定分	50%	<b>DAPWRITE</b>
クロックの不確定分	50%	<b>DAPABORT</b>
クロックの不確定分	50%	<b>DAPADDR[31:0]</b>
クロックの不確定分	50%	<b>DAPWDATA[31:0]</b>
クロックの不確定分	50%	<b>ATREADY</b>

テスト入力ポートのタイミングパラメータを、表 19-6 に示します。

**表 19-6 テスト入力ポートのタイミングパラメータ**

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	10%	<b>SE</b>
クロックの不確定分	10%	<b>SI</b>
クロックの不確定分	10%	<b>RSTBYPASS</b>
クロックの不確定分	10%	<b>CGBYPASS</b>
クロックの不確定分	10%	<b>WSII</b>
クロックの不確定分	10%	<b>WSOI</b>

エンベデッドトレース マクロセル(ETM)のタイミングパラメータを、表 19-7 に示します。

表 19-7 ETM 入力ポートのタイミングパラメータ

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	30%	ETMPWRUP

その他の出力ポートのタイミングパラメータを、表 19-8 に示します。

表 19-8 その他の出力ポートのタイミングパラメータ

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	50%	LOCKUP
クロックの不確定分	50%	SLEEPING
クロックの不確定分	50%	SLEEPDEEP
クロックの不確定分	50%	SYSRESETREQ
クロックの不確定分	50%	BRCHSTAT[3:0]
クロックの不確定分	50%	HALTED
クロックの不確定分	50%	TXEV
クロックの不確定分	50%	ATIDITM[6:0]
クロックの不確定分	50%	CURRPRI[7:0]
クロックの不確定分	70%	TRCENA

AHB 出力ポートのタイミングパラメータを、表 19-9 に示します。

表 19-9 AHB 出力ポートのタイミングパラメータ

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	50%	HTRANSI[1:0]
クロックの不確定分	50%	HSIZEI[2:0]
クロックの不確定分	50%	HPROTI[3:0]
クロックの不確定分	50%	MEMATTRI[1:0]
クロックの不確定分	50%	HBURSTI[2:0]



表 19-9 AHB 出力ポートのタイミングパラメータ

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	50%	<b>HADDRI[31:0]</b>
クロックの不確定分	50%	<b>HMASTERD[1:0]</b>
クロックの不確定分	50%	<b>HTRANS[1:0]</b>
クロックの不確定分	50%	<b>HSIZED[2:0]</b>
クロックの不確定分	50%	<b>HPROTD[3:0]</b>
クロックの不確定分	50%	<b>MEMATTRD[1:0]</b>
クロックの不確定分	50%	<b>EXREQD</b>
クロックの不確定分	50%	<b>HBURSTD[2:0]</b>
クロックの不確定分	50%	<b>HADDRD[31:0]</b>
クロックの不確定分	50%	<b>HWDATAD[31:0]</b>
クロックの不確定分	50%	<b>HWRITED</b>
クロックの不確定分	50%	<b>HMASTERS[1:0]</b>
クロックの不確定分	50%	<b>HTRANS[1:0]</b>
クロックの不確定分	50%	<b>HSIZES[2:0]</b>
クロックの不確定分	50%	<b>HPROTS[3:0]</b>
クロックの不確定分	50%	<b>MEMATTRS[1:0]</b>
クロックの不確定分	50%	<b>EXREQS</b>
クロックの不確定分	50%	<b>HBURSTS[2:0]</b>
クロックの不確定分	50%	<b>HMASTLOCKS</b>
クロックの不確定分	50%	<b>HADDRS[31:0]</b>
クロックの不確定分	50%	<b>HWDATAS[31:0]</b>
クロックの不確定分	50%	<b>HWRITES</b>

PPB 出力ポートのタイミングパラメータを、表 19-10 に示します。

表 19-10 PPB 出力ポートのタイミングパラメータ

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	50%	PADDR31
クロックの不確定分	50%	PADDR[19:2]
クロックの不確定分	50%	PSEL
クロックの不確定分	50%	PENABLE
クロックの不確定分	50%	PWRITE
クロックの不確定分	50%	PWDATA[31:0]

デバッグインタフェース出力ポートのタイミングパラメータを、表 19-11 に示します。

表 19-11 デバッグインタフェース出力ポートのタイミングパラメータ

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	50%	SWV
クロックの不確定分	50%	TRACECLK
クロックの不確定分	50%	TRACEDATA[3:0]
クロックの不確定分	50%	TDO
クロックの不確定分	50%	SWDO
クロックの不確定分	50%	nTDOEN
クロックの不確定分	50%	SWDOEN
クロックの不確定分	50%	DAPREADY
クロックの不確定分	50%	DAPSLVERR
クロックの不確定分	50%	DAPRDATA[31:0]
クロックの不確定分	50%	ATVALID
クロックの不確定分	50%	AFREADY
クロックの不確定分	50%	ATDATA[7:0]

ETM インタフェース出力ポートのタイミングパラメータを、表 19-12 に示します。

**表 19-12 ETM インタフェース出力ポートのタイミングパラメータ**

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	30%	<b>ETMTRIGGER[3:0]</b>
クロックの不確定分	30%	<b>ETMTRIGINOTD[3:0]</b>
クロックの不確定分	30%	<b>ETMIVALID</b>
クロックの不確定分	30%	<b>ETMDVALID</b>
クロックの不確定分	30%	<b>ETMFOLD</b>
クロックの不確定分	30%	<b>ETMCANCEL</b>
クロックの不確定分	30%	<b>ETMIA[31:1]</b>
クロックの不確定分	30%	<b>ETMICCFAIL</b>
クロックの不確定分	30%	<b>ETMIBRANCH</b>
クロックの不確定分	30%	<b>ETMIINDBR</b>
クロックの不確定分	30%	<b>ETMFLUSH</b>
クロックの不確定分	30%	<b>ETMFINDBR</b>
クロックの不確定分	30%	<b>ETMINTSTAT[2:0]</b>
クロックの不確定分	30%	<b>ETMINTNUM[8:0]</b>
クロックの不確定分	30%	<b>ETMISTALL</b>
クロックの不確定分	30%	<b>DSYNC</b>

AHB トレースマクロセル(HTM)インタフェース出力ポートのタイミングパラメータを、表 19-13 に示します。

表 19-13 HTM インタフェース出力ポートのタイミングパラメータ

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	50%	HTMDHADDR[31:0]
クロックの不確定分	50%	HTMDHTRANS[1:0]
クロックの不確定分	50%	HTMDHSIZE[2:0]
クロックの不確定分	50%	HTMDHBURST[2:0]
クロックの不確定分	50%	HTMDHPROT[3:0]
クロックの不確定分	50%	HTMDHWDATA[31:0]
クロックの不確定分	50%	HTMDHWRITE
クロックの不確定分	50%	HTMDHRDATA[31:0]
クロックの不確定分	50%	HTMDHREADY
クロックの不確定分	50%	HTMDHRESP[1:0]

テスト出力ポートのタイミングパラメータを、表 19-14 に示します。

表 19-14 テスト出力ポートのタイミングパラメータ

最小入力遅延	最大入力遅延	信号名
クロックの不確定分	10%	SO
クロックの不確定分	10%	WSOO
クロックの不確定分	10%	WSIO

# 付録 A

## 信号の説明

この付録では、プロセッサインタフェースの信号を一覧表記し、説明を加えます。この付録は以下のセクションから構成されています。

- クロック : P.A-2
- リセット : P.A-3
- その他 : P.A-4
- 割り込みインタフェース : P.A-6
- ICode インタフェース : P.A-7
- DCode インタフェース : P.A-8
- システムバス インタフェース : P.A-9
- 専用ペリフェラルバス インタフェース : P.A-10
- ITM インタフェース : P.A-11
- AHB-AP インタフェース : P.A-12
- ETM インタフェース : P.A-13
- AHB トレースマクロセルインタフェース : P.A-15
- テストインタフェース : P.A-16

A.1 クロック

クロック信号の一覧を、表 A-1 に示します。

表 A-1 クロック信号

名前	方向	説明
HCLK	入力	メイン Cortex-M3 クロック
FCLK	入力	フリーラン Cortex-M3 クロック
DAPCLK	入力	AHB-AP クロック

## A.2     リセット

リセット信号の一覧を、表 A-2 に示します。

表 A-2 リセット信号

名前	方向	説明
<b>PORESETn</b>	入力	パワーオンリセット。全 Cortex-M3 システムをリセットします。
<b>SYSRESETn</b>	入力	システムリセット。プロセッサ、NVIC の非デバッグ部分、バスマトリックス、MPU をリセットします。デバッグコンポーネントはリセットされません。
<b>SYSRESETREQ</b>	出力	システムリセット要求
<b>DAPRESETn</b>	入力	AHB-AP リセット

## A.3 その他

その他の信号の一覧を、表 A-3 に示します。

表 A-3 その他の信号

名前	方向	説明
<b>LOCKUP</b>	出力	<b>LOCKUP</b> により、カーネルソフトウェアに重大な誤りのあることが直ちに示されます。これは、プロセッサに組み込まれているシステム状態保護ハードウェアの起動に続く、回復不能な例外によって発生するコアのロックアップの結果です。ARMv7-M のアーキテクチャにおけるロックアップ条件の詳細については、『ARMv7-M アーキテクチャ リファレンス マニュアル』を参照して下さい。
<b>SLEEPDEEP</b>	出力	Cortex-M3 クロックを停止できることを示します。
<b>SLEEPING</b>	出力	Cortex-M3 クロックを停止できることを示します。
<b>CURRPRI[7:0]</b>	出力	現在使用されている割り込みの優先度（またはベースブースト）を示します。 <b>CURRPRI</b> は横取り優先度を表しており、2 次的な優先度は示していません。
<b>HALTED</b>	出力	ホールドデバッグ モード。 <b>HALTED</b> は、コアがデバッグ中はアサート状態に維持されます。
<b>TXEV</b>	出力	<b>SEV</b> 命令の結果として送信されるイベント。これは単一サイクルパルスです。
<b>TRCENA</b>	出力	トレースイネーブル。この信号は、デバッグ例外およびモニタ制御レジスタのビット [24] の設定を反映します。この信号は <b>TPIU</b> および <b>ETM</b> ブロックへのクロックをゲートして、トレースが禁止されているときに電力消費を低減させます。
<b>BIGEND</b>	入力	静的なエンディアン形式の選択 1 = ビッグエンディアン 0 = リトルエンディアン この信号はリセット時にサンプリングされ、リセットが非アクティブのときは変更できません。
<b>EDBGRQ</b>	入力	外部デバッグ要求
<b>PPBLOCK[5:0]</b>	入力	予約。6'b000000 に固定する必要があります。
<b>STCLK</b>	入力	システムティッククロック
<b>STCALIB[25:0]</b>	入力	システムティック較正
<b>RXEV</b>	入力	<b>WFE</b> 命令からのウェークアップを引き起こします。
<b>VECTADDR[9:0]</b>	入力	予約。10'b000000000000 に固定する必要があります。
<b>VECTADDREN</b>	入力	予約。1'b0 に固定する必要があります。



表 A-3 その他の信号

名前	方向	説明
<b>DNOTITRANS</b>	入力	I-Code および D-Code の AHB トランザクションが同時に発生しないように、プロセッサを強制設定する静的なタイオフ。これにより、単純なバスマルチプレクサをプロセッサの外部に接続することができます。
<b>AUXFAULT[31:0]</b>	入力	システムからの補助フォールトステータス情報
<b>IFLUSH</b>	入力	予約。命令フラッシュ、0 に固定する必要があります。

A.4 割り込みインタフェース

外部割り込みインタフェースの信号の一覧を、表 A-4 に示します。

表 A-4 割り込みインタフェース

名前	方向	説明
INTISR[239:0]	入力	外部割り込み信号
INTNMI	入力	マスク不能割り込み

## A.5 ICode インタフェース

ICode インタフェースの信号の一覧を、表 A-5 に示します。

表 A-5 ICode インタフェース

名前	方向	説明
<b>HADDRI[31:0]</b>	出力	32 ビット命令アドレスバス
<b>HTRANSI[1:0]</b>	出力	現在の転送が IDLE または NONSEQUENTIAL であるかどうかを示します。
<b>HSIZEI[2:0]</b>	出力	命令フェッチのサイズを示します。Cortex-M3 では、命令フェッチはすべて 32 ビットです。
<b>HBURSTI[2:0]</b>	出力	転送がバーストの一部かどうかを示します。Cortex-M3 では、命令フェッチおよびリテラルロードはすべて SINGLE として実行されます。
<b>HPROTI[3:0]</b>	出力	アクセスに関する情報を提供します。このバス上ではキャッシュ可かつバッファ不可であることが常に示されています。 <b>HPROTI[0] = 0</b> は命令フェッチを示します。 <b>HPROTI[0] = 1</b> はベクタフェッチを示します。
<b>MEMATTRI[1:0]</b>	出力	メモリ属性。このバス（共有不可、割り当て）では常時 01 です。
<b>BRCHSTAT[3:0]</b>	出力	現在および次の AHB フェッチ要求に関するヒント情報を提供します。条件付きオペコードは推測であり、後で破棄される可能性があります。 0000 ヒントなし 0001 デコード時の条件付き後方分岐 0010 デコード時の条件付き分岐 0011 実行時の条件付き分岐 0100 デコード時の無条件分岐 0101 実行時の無条件分岐 0110 予約 0111 予約 1000 デコード時の条件付き分岐が実行された場合（IHTRANS 後のサイクル） 1001 ... 1111 予約
<b>HRDATAI[31:0]</b>	入力	命令読み出しバス
<b>HREADYI</b>	入力	HIGH の場合、バス上で転送が完了したことを示します。この信号を LOW に駆動すると、転送が延長されます。
<b>HRESPI[1:0]</b>	入力	転送応答ステータス。OKAY または ERROR です。

A.6 DCode インタフェース

DCode インタフェースの信号の一覧を、表 A-6 に示します。

表 A-6 DCode インタフェース

名前	方向	説明
HADDRD[31:0]	出力	32 ビットデータ アドレスバス
HTRANS[1:0]	出力	現在の転送が IDLE、NONSEQUENTIAL、SEQUENTIAL のどれであるかを示します。
HWRITED	出力	書き込み、読み出しではありません。
HSIZED[2:0]	出力	アクセスのサイズを示します。値は 8、16、32 ビットのいずれかです。
HBURSTD[2:0]	出力	転送がバーストの一部かどうかを示します。Cortex-M3 では、データアクセスは INCR として実行されます。
HPROTD[3:0]	出力	アクセスに関する情報を提供します。このバス上ではキャッシュ可かつバッファ不可であることが常に示されています。
EXREQD	-	排他要求
MEMATTRD[1:0]	出力	メモリ属性 このバス（共有不可、割り当て）では常時 01 です。
HMASTERD[1:0]	出力	現在の D-Code バスマスタを示します。 <ul style="list-style-type: none"><li>0 = コアデータ側アクセス</li><li>1 = DAP アクセス</li><li>2 = コア命令側アクセス。これらには、HPROTD[0] がデータとして示されるベクタフェッチが含まれます。この値は HMASTERD では出現しません。</li><li>3 = 予約。この値は HMASTERD では出現しません。</li></ul>
HWDATAD[31:0]	入力	データ読み出しバス
HREADYD	入力	HIGH の場合、バス上で転送が完了したことを示します。この信号を LOW に駆動すると、転送が延長されます。
HRESPD[1:0]	入力	転送応答ステータス。OKAY または ERROR です。
HRDATAD[31:0]	入力	読み出しデータ
EXRESPD	入力	排他応答

## A.7 システムバス インタフェース

システムバス インタフェースの信号の一覧を、表 A-7 に示します。

表 A-7 システムバス インタフェース

名前	方向	説明
<b>HADDRS[31:0]</b>	出力	32 ビットのアドレス
<b>HTRANS[1:0]</b>	出力	現在の転送のタイプを示します。値は IDLE、NONSEQUENTIAL、SEQUENTIAL のいずれかです。
<b>HSIZES[2:0]</b>	出力	アクセスのサイズを示します。値は 8、16、32 ビットのいずれかです。
<b>HBURSTS[2:0]</b>	出力	転送がバーストの一部かどうかを示します。
<b>HPROTS[3:0]</b>	出力	アクセスに関する情報を提供します。
<b>HWDATAS[31:0]</b>	出力	32 ビットの書き込みデータバス
<b>HWRITES</b>	出力	書き込み、読み出しではありません。
<b>HMASTLOCKS</b>	出力	バス上で不可分の必要があるトランザクションを示します。これはビットバンド書き込みに対してのみ使用されます（読み出し - 変更 - 書き込みとして実行）。
<b>EXREQS</b>	出力	排他要求
<b>MEMATTRS[1:0]</b>	出力	メモリ属性。ビット 0 = 割り当て、ビット 1 = 共有可
<b>HMASTERS[1:0]</b>	出力	現在のシステムバス マスタを示します。 <ul style="list-style-type: none"> <li>0 = コアのデータ側アクセス、またはマスタタイプが 0 に設定された DAP アクセス。</li> <li>1 = マスタタイプが 1 に設定された DAP のアクセス</li> <li>2 = コアの命令側アクセス。これには、<b>HPROTS[0]</b> がデータとして示されるベクタフェッチが含まれます。</li> <li>3 = 予約。この値は <b>HMASTERS</b> では出現しません。</li> </ul>
<b>HRDATAS[31:0]</b>	入力	読み出しデータバス
<b>HREADYs</b>	入力	HIGH の場合、バス上で転送が完了したことを示します。この信号を LOW に駆動すると、転送が延長されます。
<b>HRESPS[1:0]</b>	入力	転送応答ステータス。OKAY または ERROR です。
<b>EXRESPS</b>	入力	排他応答

A.8 専用ペリフェラルバス インタフェース

PPB インタフェースの信号の一覧を、表 A-8 に示します。

表 A-8 専用ペリフェラルバス インタフェース

名前	方向	説明
PADDR[19:2]	出力	17 ビットのアドレス。外部専用ペリフェラルバスに関連するビットのみが駆動されます。
PADDR31	出力	この信号は、AHB-AP が要求をしているマスタの場合に HIGH に駆動されます。DCore が要求をしているマスタの場合は LOW に駆動されます。
PSEL	出力	データ転送が要求されていることを示します。
PENABLE	出力	すべてのアクセスのタイミング用のストロブ。APB 転送の 2 番目のサイクルを示します。
PWDATA[31:0]	出力	32 ビットの書き込みデータバス
PWRITE	出力	書き込み、読み出しではありません。
PRDATA[31:0]	入力	読み出しデータバス

## A.9 ITM インタフェース

ITM インタフェースの信号の一覧を、表 A-9 に示します。

表 A-9 ITM インタフェース

名前	方向	説明
ATVALID	出力	ATB は有効です。
AFREADY	出力	ATB フラッシュ
ATDATA[7:0]	出力	ATB データ
ATIDITM[6:0]	出力	TPIU 用の ITM ID
ATREADY	入力	ATB 準備完了
TPIUEMIT	入力	最初のパケットがエミッタ FIFO に入力されるまで、タイムスタンプのカウントを 0 に保持します。
TPIUBAUD	入力	タイムスタンプを外部プロトコルの観察可能なボーレートにするための、タイムスタンプカウンタの基準

A.10 AHB-AP インタフェース

AHB-AP インタフェースの信号の一覧を、表 A-10 に示します。

表 A-10 AHB-AP インタフェース

名前	方向	説明
DAPRDATA[31:0]	出力	読み出しバスは、 <b>DAPWRITE</b> が <b>LOW</b> のとき、選択された AHB-AP によって読み出しサイクルの間駆動されます。
DAPREADY	出力	AHB-AP はこの信号を使用して、DAP 転送を延長します。
DAPSLVERR	出力	エラー応答で、次の理由で発生します。 <ul style="list-style-type: none"><li>• マスタポートがエラー応答を生成したか、または転送を妨げる <b>DAPEN</b> のために転送が起動されなかった。</li><li>• <b>DAPABORT</b> 操作の後で、AP レジスタへのアクセスが受け付けられなかった。</li></ul>
DAPCLKEN	入力	DAP クロック許可（省電力化）
DAPEN	入力	AHB-AP 許可
DAPADDR[31:0]	入力	DAP アドレスバス
DAPSEL	入力	DAP デコーダから各 AP に対して生成される信号を選択します。この信号は、スレーブデバイスが選択され、データ転送が要求されていることを示します。各スレーブに対して <b>DAPSEL</b> 信号が存在します。この信号は、駆動中の DP によっては生成されません。デコーダはアドレスバスをモニタして、関連する <b>DAPSEL</b> をアサートします。
DAPENABLE	入力	この信号は、DP から AHB-AP への DAP 転送の 2 番目および以後のサイクルを示します。
DAPWRITE	入力	<b>HIGH</b> の場合、DP から AHB-AP への DAP 書き込みアクセスを示します。 <b>LOW</b> の場合は読み出しアクセスを示します。
DAPWDATA[31:0]	入力	書き込みバスは、 <b>DAPWRITE</b> が <b>HIGH</b> のとき、書き込みサイクル中に DP ブロックによって駆動されます。
DAPABORT	入力	現在の転送をアボートします。AHB-AP は、AHB マスタポートで進行中の転送の状態に影響を与えずに <b>DAPREADY HIGH</b> を返します。



## A.11 ETM インタフェース

ETM インタフェースの信号の一覧を、表 A-11 に示します。

表 A-11 ETM インタフェース

名前	方向	説明
<b>ETMTRIGGER[3:0]</b>	出力	DWT からのトリガ。4 つの DWT コンパレータのそれぞれに対して 1 ビットが使用されます。
<b>ETMTRIGINOTD[3:0]</b>	出力	ETM が命令またはデータ一致でトリガされるかどうかを示します。
<b>ETMIVALID</b>	出力	命令は有効です。
<b>ETMIA[31:1]</b>	出力	実行中の命令の PC
<b>ETMICCFAIL</b>	出力	条件コードの失敗。現在の命令で条件付実行のチェックが失敗したかパスしたかを示します。
<b>ETMIBRANCH</b>	出力	オペコードは分岐ターゲットです。
<b>ETMIINDBR</b>	出力	オペコードは間接分岐ターゲットです。
<b>ETMINTSTAT[2:0]</b>	出力	割り込みステータス。現在のサイクルの割り込みステータスをマークします。 000- ステータスなし 001- 割り込み開始 010- 割り込み終了 011- 割り込み復帰 100- ベクタフェッチおよびスタックプッシュ <b>ETMINTSTAT</b> の開始 / 復帰は、新規割り込みコンテキストの最初のサイクルでアサートされます。 <b>ETMIVALID</b> なしで終了します。
<b>ETMINTNUM[8:0]</b>	出力	現在の実行コンテキストの割り込み番号を示します。
<b>ETMISTALL</b>	出力	コアが示している最後の命令が、まだ実行ステージに入っていないことを示します。
<b>ETMFLUSH</b>	出力	PC を変更するオペコードが実行されたか、または割り込みプッシュ / ポップが開始されました。
<b>ETMPWRUP</b>	入力	ETM が許可されています。
<b>ETMDVALID</b>	出力	データは有効です。
<b>ETMCANCEL</b>	出力	命令がキャンセルされました。

表 A-11 ETM インタフェース

名前	方向	説明
ETMFINDBR	出力	フラッシュは間接的です。フラッシュヒント デスティネーションが PC から推測できないことを示します。
ETMFOLD	出力	オペコードのフォールド。このサイクルで IT オペコードがフォールドされました。現在の（16 ビット）オペコードと IT 命令（16 ビット）を超えて PC が進みます。これは ETMIA に反映されます。
ETMFIFOFULL	入力	ETM によって駆動されます（接続されている場合）。 <b>ETMFIFOFULL</b> は ETM FIFO がフルの場合にアサートされ、FIFO がドレインされるまでプロセッサはストールします。これによって、トレースが欠落しないことが保証されます。
DSYNC	出力	DWT からの同期化パルス

## A.12 AHB トレースマクロセルインタフェース

AHB トレースマクロセル (HTM) インタフェースの信号の一覧を、表 A-12 に示します。

**表 A-12 HTM インタフェース**

名前	方向	説明
HTMDHADDR[31:0]	出力	32 ビットのアドレス
HTMDHTRANS[1:0]	出力	現在のデータ転送のタイプを示す出力。値は IDLE、NONSEQUENTIAL、SEQUENTIAL のいずれかです。
HTMDHSIZE[1:0]	出力	アクセスのサイズを示します。値は 8、16、32 ビットのいずれかです。
HTMDHBURST[2:0]	出力	転送がバーストの一部かどうかを示す出力
HTMDHPROT[3:0]	出力	アクセスに関する情報を提供します。
HTMDHWDATA[31:0]	出力	32 ビットの書き込みデータバス
HTMDHWRITE	出力	書き込み、読み出しではありません。
HTMDHRDATA[31:0]	出力	読み出しデータバス
HTMDHREADY	出力	準備完了信号
HTMDHRESP[1:0]	出力	転送応答ステータス。OKAY または ERROR です。

# A.13    テストインタフェース

テストインタフェースの信号の一覧を、表 A-13 に示します。

表 A-13 テストインタフェース

名前	方向	説明
SE	入力	スキャンは許可されています。
RSTBYPASS	入力	スキャンテストのリセットバイパス。 <b>PORESETn</b> は、スキャンテスト中に使用される唯一のリセットです。
CGBYPASS	入力	スキャンテスト用のアーキテクチャ上のクロックゲート バイパス

# 用語集

この用語集では、ARM Limited の発行する技術文書で使用されている用語のいくつかについて説明します。

## アボート

Abort。試みられたメモリアクセスが無効であるか許可されていない、もしくはメモリアクセスによって返されたデータが無効であることをコアに通知する機構。アボートは、無効なまたは保護された命令やデータメモリへのアクセスを実行した結果として、外部メモリシステムまたは内部メモリシステムによって引き起こされる可能性があります。

アボート (Data Abort)、外部アボート (External Abort)、プリフェッチ アボート (Prefetch Abort) も参照して下さい。

## アドレッシングモード

Addressing modes。命令で使用する値を生成するために、複数のさまざまな命令で共有される、各種の機構。

## アドバンスド ハイパフォーマンスバス (AHB)

Advanced High-performance Bus (AHB)。アドレス / 制御フェーズとデータフェーズとの間である決まったパイプラインを使用するバスプロトコル。AMBA AXI プロトコルで提供されている機能のサブセットのみをサポートします。完全な AMBA AHB プロトコル仕様には、一般的なマスタ / スレーブの設計資産 (IP) を使った開発では必要とされない機能が多く含まれているため、通常はプロトコルのサブセットだけを使用することをお勧めします。このサブセットは、AMBA AHB-Lite プロトコルとして定義されています。

アドバンスド マイクロコントローラバス アーキテクチャ (Advanced Microcontroller Bus Architecture) と AHB-Lite も参照して下さい。

**アドバンスト マイクロコントローラバス アーキテクチャ (AMBA)**

Advanced Microcontroller Bus Architecture (AMBA)。相互接続のための方針が記載された、プロトコル仕様のファミリ。AMBA は、オンチップバスに関する ARM のオープンな規格です。システムオンチップ (SoC)(System-on-Chip) を構築する機能ブロックの相互接続と管理のための方針が詳しく記載された、オンチップバスの仕様です。1 つまたは複数の CPU または信号プロセッサ、および複数のペリフェラルを含む組み込みプロセッサの開発に役立ちます。AMBA は、SoC モジュール用の共通バックボーンを定義することによって、再利用可能な設計手法をより完全なものにします。

**アドバンスト ペリフェラルバス (APB)**

Advanced Peripheral Bus (APB)。AXI や AHB よりも単純なバスプロトコル。タイマ、割り込みコントローラ、UART、I/O ポートなどの補助的な、または汎用のペリフェラルで使用するために設計されています。メインのシステムバスへの接続は、システムとペリフェラルとの間のバスブリッジを経由して行なわれていて、システムの消費電力を抑えられます。

**AHB** アドバンスト ハイパフォーマンスバス (Advanced High-performance Bus) 参照。

**AHB アクセスポート (AHB-AP)**

AHB Access Port (AHB-AP)。DAP のオプションコンポーネントで、SoC への AHB インタフェースを提供します。

**AHB-AP** AHB アクセスポート (AHB Access Port) 参照。

**AHB-Lite** 完全な AMBA AHB プロトコル仕様のサブセット。大部分の AMBA AHB マスタ / スレーブ設計に必要なすべての基本機能を提供しており、特に複数レイヤの AMBA 相互接続で使用されます。ほとんどの場合、完全な AMBA AHB インタフェースで提供されている追加の機能は、AMBA AXI プロトコルインタフェースで実装するとより効率的になります。

**AHB トレースマクロセル**

AHB Trace Macrocell。プロセッサコアに接続されたときに、トレースポートにデータのトレース情報を出力するハードウェアマクロセルです。

**アラインド** Aligned。データサイズで決まるバイト数で割り切れるアドレスに保存されたデータ項目を、アラインド、またはアラインしていると呼びます。アラインしているワードとハーフワードのアドレスは、それぞれ 4 と 2 で割り切れます。したがって、ワードアラインドとハーフワードアラインドという用語は、それぞれ 4 と 2 で割り切れるアドレスを規定しています。

**AMBA** アドバンスト マイクロコントローラバス アーキテクチャ (Advanced Microcontroller Bus Architecture) 参照。

**アドバンスト トレースバス (ATB)**

Advanced Trace Bus (ATB)。CoreSight のトレースキャプチャ資源を共有するため、トレースデバイスによって使用されるバス。

**APB** アドバンスト ペリフェラルバス (Advanced Peripheral Bus) 参照。

**特定用途向け集積回路 (ASIC)**

Application Specific Integrated Circuit (ASIC)。特定用途の機能を実行するために設計された集積回路。特注または量産が可能です。

**特定用途用標準部品 / 製品 (ASSP)**

Application Specific Standard Part/Product (ASSP)。特定用途の機能を発揮するために設計された集積回路。通常は、2つ以上の個別の回路機能から成り、いくつかの特定用途市場の製品群に使うのに適するようにビルディングブロックとして統合されています。

**アーキテクチャ**

Architecture。プロセッサとプロセッサに接続されている構成要素を特徴づけるハードウェアおよびソフトウェアの構成。装置の動作を記述するときに、装置を似たような特徴でまとめて扱えるようにします。例えば、ハーバードアーキテクチャ、命令セットアーキテクチャ、ARMv7-M アーキテクチャなど。

**ARM 命令**

ARM instruction。ARM 命令セットアーキテクチャ (ISA)(Instruction Set Architecture) の命令。これらの命令は Cortex-M3 では実行できません。

**ARM 状態**

ARM state。プロセッサ状態のうち、プロセッサが ARM ISA の命令を実行する状態。Cortex-M3 プロセッサは Thumb 状態でのみ動作し、ARM 状態では動作しません。

**ASIC**

特定用途向け集積回路 (Application Specific Integrated Circuit) 参照。

**ASSP**

特定用途用標準部品 / 製品 (Application Specific Standard Part/Product) 参照。

**ATB**

アドバンスト トレースバス (Advanced Trace Bus) 参照。

**ATB ブリッジ**

ATB bridge。同期 ATB ブリッジは、パイプラインステージの追加によって、タイミング収束を容易にするレジスタスライスを提供します。また、2つの同期 ATB ドメイン間の単方向リンクを提供します。

非同期 ATB ブリッジは、非同期クロックを使用した2つの ATB ドメイン間の単方向リンクを提供します。これは、異なるクロック ドメインにあるコンポーネントを ATB ポートで接続するのをサポートすることが目的です。

**ベースレジスタ**

Base register。命令のアドレス計算の基準値を保持するため、ロード / ストア命令で指定されるレジスタ。命令とそのアドレッシングモードによっては、ベースレジスタ値にオフセットが加算または減算されて、メモリに送信するアドレスが形成される場合があります。

**ベースレジスタ ライトバック**

Base register write-back。アドレスが連続したメモリ内の次の上位アドレスまたは次の下位アドレスを指し示すように、命令のターゲットアドレスの計算に使用するベースレジスタの内容を更新すること。これによって、連続的な命令による転送のためのターゲットアドレスをフェッチする必要がなく、連続するメモリに対してより高速なバーストアクセスが可能になります。

**ビート**

Beat。1つのバースト転送中の個々のデータ転送に対するもう1つの言い方。例えば、INCR4 バーストは4ビートで構成されます。

**BE-8**

バイト不変システムでの、ビッグエンディアン形式のメモリビュー。

BE-32、LE、バイト不変 (Byte-invariant)、ワード不変 (Word-invariant) も参照して下さい。

<b>BE-32</b>	ワード不変システムでの、ビッグエンディアン形式のメモリビュー。  BE-8、LE、バイト不変 (Byte-invariant)、ワード不変 (Word-invariant) も参照して下さい。
<b>ビッグエンディアン</b>	Big-endian。データ ワード内の最上位バイトから最下位バイトまでが、メモリ内のアドレスの昇順に保存されるバイト配列方式。  リトルエンディアン (Little-endian) とエンディアン形式 (Endianness) も参照して下さい。
<b>ビッグエンディアンメモリ</b>	Big-endian memory。次のようなメモリです。 <ul style="list-style-type: none"> <li>ワードアラインされたアドレスのバイトまたはハーフワードが、そのアドレスのワード内で最上位のバイトまたはハーフワードである。</li> <li>ハーフワードアラインされたアドレスのバイトが、そのアドレスのハーフワード内で最上位バイトである。</li> </ul> リトルエンディアン メモリ (Little-endian memory) も参照して下さい。
<b>バウンダリスキャン チェイン</b>	Boundary scan chain。バウンダリスキャン チェインは、標準の JTAG TAP インタフェースを使用してバウンダリスキャン技術を実装しているデバイスのシリアル接続で構成されます。各デバイスには少なくとも 1 つの TAP コントローラがあり、 <b>TDI</b> と <b>TDO</b> との間の接続チェーンを形成するシフトレジスタを搭載しています。このチェーンを通して、テストデータがシフトされます。プロセッサには数個のシフトレジスタを搭載できるため、デバイスの指定した部分にアクセスすることができます。
<b>分岐フォールディング</b>	Branch folding。実行パイプラインに送られる命令ストリームから、分岐命令を完全に取り除く手法です。
<b>ブレイクポイント</b>	Breakpoint。プログラムの実行を停止させようとする位置にある命令を識別するために、デバッガによって提供される機構。プログラマはブレイクポイントを挿入することによって、プログラムの実行中の決まった位置で、レジスタの内容、メモリの位置、変数の値を確認して、プログラムが正常に動作しているかどうかをテストすることができます。プログラムのテストが完了した後で、ブレイクポイントは削除されます。  ウォッチポイント (Watchpoint) も参照して下さい。
<b>バースト</b>	Burst。連続アドレスに対する一連の転送。アドレスが連続しているため、2 回目以降の転送ではアドレスを指定する必要がありません。この方法によって、一連の転送の実行速度が向上します。AMBA 経由のバーストは、バーストの長さでアドレスのインクリメント方法を示す信号によって制御されます。  ビート (Beat) も参照して下さい。
<b>バイト</b>	Byte。8 ビットのデータ項目。



<b>バイト不変</b>	Byte-invariant。バイト不変のシステムでは、リトルエンディアンとビッグエンディアンの動作が切り替えられても、メモリの各バイトのアドレスは変更されません。1 バイトを超えるデータ項目をメモリからロード、またはメモリにストアするとき、そのデータ項目を構成するバイトが、メモリアクセスのエンディアン形式に応じて正しい順序に配列されます。ARM アーキテクチャでは、ARMv6 およびそれ以降のバージョンでバイト不変システムがサポートされています。バイト不変のサポートが選択されている場合は、アンアラインドのハーフワードとワードによるメモリアクセスもサポートされます。複数ワードアクセスは、ワードアラインしている必要があります。
	ワード不変 (Word-invariant) も参照して下さい。
<b>クロックゲート</b>	Clock gating。制御信号でマクロセルのクロック信号をゲートし、そのクロックを使用することによって、マクロセルの動作状態が制御されます。
<b>命令当たりのクロック数 (CPI)</b>	Clocks Per Instruction (CPI)。命令当たりのサイクル数 (CPI)(Cycles Per Instruction) 参照。
<b>コールドリセット</b>	Cold reset。パワーオンリセット (power-on reset) と呼ばれることもあります。ウォームリセット (Warm reset) 参照。
<b>コンテキスト</b>	Context。マルチタスクのオペレーティングシステム上で各プロセスが動作する環境。高速コンテキストスイッチ (Fast context switch) も参照して下さい。
<b>コア</b>	Core。ALU、データパス、汎用レジスタ、プログラムカウンタ、命令デコードおよび制御回路を含む、プロセッサの一部。
<b>コアリセット</b>	Core reset。ウォームリセット (Warm reset) 参照。
<b>CoreSight</b>	完全なシステム オンチップ (SoC) のモニタ、トレース、デバッグを行うためのインフラストラクチャ。
<b>CPI</b>	命令当たりのサイクル数 (Cycles per instruction) 参照。
<b>命令当たりのサイクル数 (CPI)</b>	Cycles Per Instruction (CPI)。命令当たりのサイクル数 (または命令当たりのクロック数) は、1 クロック サイクルで実行可能なコンピュータ命令の数の指標です。この性能指標は、同じ命令セットを実装した異なる CPU の性能を比較するために使用できます。値が低いほど、パフォーマンスが高いことを意味します。
<b>データアボート</b>	Data Abort。データメモリの不正な位置にアクセスが試みられたことを、メモリシステムからコアへ通知する方法。プロセッサがアボートを引き起こしたデータを使用しようとする、例外をおこす必要があります。アボート (Abort) も参照して下さい。
<b>DCode メモリ</b>	DCode Memory。0x00000000 ~ 0x1FFFFFFF のメモリ空間。

**デバッグアクセス ポート (DAP)**

Debug Access Port (DAP)。一種の TAP ブロックであり、システムバスへのアクセスでは AMBA の AHB または AHB-Lite マスタとして動作します。DAP は、システム規模のデバッグをサポートする回路ブロック群を総称するために使用される用語です。DAP はモジュラーコンポーネントで、単一のインタフェースを通して、メモリマップされた AHB や CoreSight APB などの複数システムへの任意のアクセスをサポートするように拡張可能なように意図されています。

**デバッガ**

Debugger。ソフトウェアの障害を検出し、場所を特定し、修正するのに使用されるプログラムと、ソフトウェアのデバッグをサポートするカスタムハードウェアを組み合わせたデバッグシステム。

**エンベデッドトレース マクロセル (ETM)**

エンベデッド Trace Macrocell (ETM)。ハードウェアマクロセルで、プロセッサコアに接続されたときに、トレースポートに命令トレース情報を出力します。

**エンディアン形式**

Endianness。バイトの順序。データワードの連続するバイトがメモリに保存される順序を決定する方式。システムのメモリマッピングの見え方。リトルエンディアン (Little-endian) とビッグエンディアン (Big-endian) も参照して下さい。

**ETM**

エンベデッドトレース マクロセル( エンベデッド Trace Macrocell) 参照。

**例外**

Exception。エラーまたはイベントで、これが発生するとプロセッサは現在実行中の命令ストリームを停止し、特定の例外ハンドラまたは割り込み処理ルーチンを実行します。例外には外部割り込みや NMI のほか、プログラムの実行を中断する必要があるほど深刻だと見なされるフォールトやエラーイベントが含まれます。例として、無効なメモリアクセス、外部割り込み、未定義命令を実行しようとした場合が挙げられます。例外が発生すると、通常のプログラムフローが中断され、対応する例外ベクタで実行が再開されます。例外ベクタには、例外を処理する割り込み処理ルーチンの最初の命令が含まれています

**例外ハンドラ**

Exception handler。割り込み処理ルーチン (Interrupt service routine) 参照。

**例外ベクタ**

Exception vector。割り込みベクタ (Interrupt vector) 参照。

**外部 PPB**

External PPB。0xE0040000 ～ 0xE00FFFFFF の PPB メモリ空間。

**フラッシュパッチとブレイクポイントユニット (FPB)**

Flash Patch and Breakpoint unit (FPB)。アドレス一致タグのセットであり、フラッシュメモリへのアクセスを SRAM の特別な部分へのアクセスに変更します。これによって、フラッシュメモリの位置にパッチをあてることで、ブレイクポイントを設定したり、すばやい修正または変更が可能になります。

**フォーマッタ**

Formatter。フォーマッタは ETB と TPIU の内部入力ブロックであり、トレースソースの ID をデータに埋め込んで単一のトレースストリームを形成します。

**ハーフワード**

Halfword。16 ビットのデータ項目。

<b>ホールトモード</b>	Halt mode。互いに排他的な 2 つのデバッグモードのうちの 1 つ。ホールトモードでは、ブレークポイントまたはウォッチポイントに遭遇したときに、すべてのプロセッサの実行が停止します。すべてのプロセッサの状態、コプロセッサの状態、メモリと I/O の位置を、JTAG インタフェースから検査および変更できます。  モニタデバッグモード (Monitor debug-mode) も参照して下さい。
<b>ホスト</b>	Host。データや他のサービスを別のコンピュータに提供するコンピュータ。特に、デバッグ対象のターゲットにデバッグサービスを提供しているコンピュータ。
<b>HTM</b>	AHB トレースマクロセル (AHB Trace Macrocell) 参照。
<b>ICode メモリ</b>	ICode Memory。0x00000000 ～ 0x1FFFFFFF のメモリ空間。
<b>不正命令</b>	Illegal instruction。アーキテクチャで未定義の命令。
<b>実装定義</b>	Implementation-defined。動作がアーキテクチャで定義されておらず、個別の実装によって定義や文書化が行われます。
<b>実装固有</b>	Implementation-specific。動作がアーキテクチャで定義されていないが、実装ごとに文書化する必要があることを意味します。使用可能な実装オプションが多数あり、選択したオプションによってソフトウェアの互換性に影響がない場合に使用されます。
<b>命令サイクル数</b>	Instruction cycle count。命令がパイプラインの実行ステージを占有するサイクル数。
<b>計装トレース</b>	Instrumentation trace。リアルタイムシステムをデバッグするためのコンポーネントであり、メモリマップの簡単なトレースインタフェースで、printf 形式のデバッグを提供します。
<b>インテリジェント電力管理 (IEM)</b>	Intelligent Energy Management (IEM)。デバイスの消費電力を低減するために使用される技術で、動的な電圧スケーリングとクロック周波数の変更を可能にします。
<b>内部 PPB</b>	Internal PPB。0xE0000000 ～ 0xE003FFFF の PPB メモリ空間。
<b>割り込み処理ルーチン</b>	Interrupt service routine。割り込みが発生したときに、プロセッサの制御が渡されるプログラム。
<b>割り込みベクタ</b>	Interrupt vector。下位メモリにある多くの固定アドレスの 1 つで、対応する割り込み処理ルーチンの最初の命令が含まれています
<b>ジョイントテストアクショングループ (JTAG)</b>	Joint Test Action Group (JTAG)。IEEE 1149.1 規格を策定した団体の名前。この規格では、集積回路デバイスのインサーキットテストに使用される、バウンダリスキャン アーキテクチャが定義されています。頭文字の JTAG で広く知られています。

<b>JTAG</b>	ジョイントテスト アクショングループ (Joint Test Action Group) 参照。
<b>JTAG デバッグポート (JTAG-DP)</b>	JTAG Debug Port (JTAG-DP)。デバッグアクセス用に標準の JTAG インタフェースを提供する DAP 用のオプションの外部インタフェース。
<b>JTAG-DP</b>	JTAG デバッグポート (JTAG Debug Port) 参照。
<b>LE</b>	バイト不変とワード不変の両方のシステムにおける、リトルエンディアン形式のメモリビュー。バイト不変 (Byte-invariant) とワード不変 (Word-invariant) も参照して下さい。
<b>リトルエンディアン</b>	Little-endian。データワード内の最下位バイトから最上位バイトまでが、メモリ内のアドレスの昇順に保存されるバイト配列方式。 ビッグエンディアン (Big-endian) とエンディアン形式 (Endianness) も参照して下さい。
<b>リトルエンディアンメモリ</b>	Little-endian memory。次のようなメモリです。 <ul style="list-style-type: none"><li>• ワードアラインされたアドレスのバイトまたはハーフワードが、そのアドレスにあるワード内の最下位のバイトまたはハーフワードである。</li><li>• ハーフワードアラインされたアドレスのバイトが、そのアドレスにあるハーフワード内の最下位バイトである。</li></ul> ビッグエンディアンメモリ (Big-endian memory) も参照して下さい。
<b>ロード/ストアアーキテクチャ</b>	Load/store architecture。データ処理動作が、直接メモリの内容に対して行われるのではなく、レジスタの内容に対してのみ行われるプロセッサアーキテクチャ。
<b>ロードストア ユニット (LSU)</b>	Load Store Unit (LSU)。ロード/ストア転送を処理するプロセッサの部分。
<b>LSU</b>	ロードストア ユニット (Load Store Unit) 参照。
<b>マクロセル</b>	Macrocell。インタフェースと動作が定義された複合回路ブロック。一般的な VLSI システムは、複数のマクロセル (プロセッサ、ETM、メモリブロックなど) と、特定用途の論理回路で構成されます。
<b>メモリコヒーレンシ</b>	Memory coherency。メモリは、データ読み出しまたは命令フェッチによって読み出された値が、最後にその位置に書き込まれた値と一致していれば、コヒーレントです。メモリコヒーレンシは、メインメモリ、ライトバッファ、キャッシュを搭載したシステムのように、使用可能な物理位置が複数存在する場合には、実現が容易ではありません。
<b>メモリ保護ユニット (MPU)</b>	Memory Protection Unit (MPU)。メモリブロックに対するアクセス許可を制御するハードウェア。MMU とは異なり、MPU はアドレスを変更しません。

<b>マイクロプロセッサ</b>	Microprocessor。プロセッサ (Processor) 参照。
<b>モニタデバッグモード</b>	Monitor debug-mode。互いに排他的なデバッグモードのうちの 1 つ。モニタデバッグモードでは、プロセッサは、デバッグモニタまたはオペレーティングシステムのデバッグタスクで提供されるソフトウェア アボートハンドラを許可します。これによって、ブレークポイントまたはウォッチポイントに遭遇して、通常のプログラム実行が中断している間であっても、重要なシステム割り込み処理を継続することができます。  ホールトモード (Halt mode) も参照して下さい。
<b>MPU</b>	メモリ保護ユニット (Memory Protection Unit) 参照。
<b>マルチレイヤ</b>	Multi-layer。クロスバースイッチに似た相互接続方式。相互接続の各マスタにはそれぞれのスレーブへの直接リンクがあり、このリンクは他のマスタとは共有されません。これによって、各マスタは他のマスタと並列に転送を処理できます。マルチレイヤ相互接続での競合は、ペイロードの宛先、通常はスレーブ、でのみ発生します。
<b>ペナルティ</b>	Penalty。命令フローが仮定または予想と異なるため、実行ステージの有効なパイプライン動作が発生しないサイクル数。
<b>PFU</b>	プリフェッチユニット (Prefetch Unit) 参照。
<b>パワーオンリセット</b>	Power-on reset。コールドリセット (Cold reset) 参照。
<b>PPB</b>	専用ペリフェラルバス (Private Peripheral Bus) 参照。
<b>プリフェッチ</b>	Prefetching。パイプライン処理のプロセッサで、先行する命令の実行が完了する前に、その後の命令をメモリからフェッチしてパイプラインに送り込む処理。命令のプリフェッチは、その命令が必ず実行されることを意味しません。
<b>プリフェッチアボート</b>	Prefetch Abort。メモリの不正な位置から命令がフェッチされたことを、メモリシステムからコアへ通知すること。プロセッサがその命令を実行しようとしたら、例外を取得する必要があります。プリフェッチアボートは、無効な命令メモリへのアクセスを試みた結果として、外部または内部のメモリシステムが引き起こす可能性があります。データアボート (Data Abort)、アボート (Abort) も参照して下さい。
<b>プリフェッチユニット (PFU)</b>	Prefetch Unit (PFU)。サイクル毎に 1 ワードを供給可能なメモリシステムから、命令をフェッチします。PFU は 3 ワードまでのフェッチを FIFO にバッファすることができます。つまり、最大 3 つの Thumb-2 命令または 6 つの Thumb 命令をバッファすることができます。
<b>専用ペリフェラルバス</b>	Private Peripheral Bus。0xE0000000 ～ 0xE00FFFFF のメモリ空間。
<b>プロセッサ</b>	Processor。コンピュータ命令を使用してデータを処理するために必要なコンピュータシステムの回路。プロセッサは、マイクロプロセッサの略称です。完全に機能する最小のコンピュータシステムを作成するには、クロックソース、電源、メインメモリも必要です。

<b>RealView ICE</b>	JTAG インタフェースを使用して、組み込みプロセッサコアをデバッグするためのシステム。
<b>予約</b>	Reserved。制御レジスタまたは命令フォーマット内のフィールドが実装で定義される場合、そのフィールドは予約されます。このようなフィールドの内容が 0 ではない場合、予測不能な結果が引き起こされます。これらのフィールドは、アーキテクチャの将来の拡張に備えて予約される場合と、実装固有の場合があります。実装で使用されないすべての予約ビットは、0 として読み書きする必要があります。
<b>SBO</b>	常に 1 (Should Be One) 参照。
<b>SBZ</b>	常に 0 (Should Be Zero) 参照。
<b>SBZP</b>	常に 0 または保持 (Should Be Zero or Preserved) 参照。
<b>スキャンチェーン</b>	Scan chain。スキャンチェーンはシリアル接続したデバイスで構成され、標準の JTAG TAP インタフェースを使用してバウンダリスキャン技術を実装しています。各デバイスには少なくとも 1 つの TAP コントローラがあり、 <b>TDI</b> と <b>TDO</b> との間の接続チェーンを形成するシフトレジスタを搭載しています。このチェーンを通して、テストデータがシフトされます。プロセッサには数個のシフトレジスタを搭載できるため、デバイスの指定した部分にアクセスすることができます。
<b>常に 1(SBO)</b>	Should Be One (SBO)。ソフトウェアで 1 (ビットフィールドの場合はすべてのビットに 1) を書き込む必要があります。0 を書き込んだ場合、結果は予測不能です。
<b>常に 0(SBZ)</b>	Should Be Zero (SBZ)。ソフトウェアで 0 (ビットフィールドの場合はすべてのビットに 0) を書き込む必要があります。1 を書き込んだ場合、結果は予測不能です。
<b>常に 0 または保持 (SBZP)</b>	Should Be Zero or Preserved (SBZP)。ソフトウェアで 0 (ビットフィールドの場合はすべてのビットに 0) を書き込むか、同じプロセッサの同じフィールドから以前読み出した値をそのまま書き戻して保持する必要があります。
<b>シリアルワイヤ デバッグポート</b>	Serial-Wire Debug Port。DAP 用のオプションの外部インタフェースで、シリアルワイヤの双方向デバッグインタフェースを提供します。
<b>シリアルワイヤ JTAG デバッグポート</b>	Serial-Wire JTAG Debug Port。JTAG-DP および SW-DP を組み合わせた、標準のデバッグポート。
<b>SW-DP</b>	シリアルワイヤ デバッグポート (Serial-Wire Debug Port) 参照。
<b>SWJ-DP</b>	シリアルワイヤ JTAG デバッグポート (Serial-Wire JTAG Debug Port) 参照。
<b>同期化基本命令</b>	Synchronization primitive。メモリの同期化基本命令は、メモリの同期を保証するために使用される命令です。LDREX および STREX 命令が該当します。
<b>システムメモリ</b>	System memory。0x20000000 ~ 0xFFFFFFFF のメモリ空間、ただし 0xE0000000 ~ 0xE00FFFFFFF の PPB 空間を除きます。

<b>TAP</b>	テスト アクセスポート (Test Access Port) 参照。
<b>テスト アクセスポート (TAP)</b>	Test Access Port (TAP)。JTAG バウンダリスキャン アーキテクチャの入出力インタフェースと制御インタフェースを形成する、4 つの必須端子と 1 つのオプション端子の集合。必須端子は、 <b>TDI</b> 、 <b>TDO</b> 、 <b>TMS</b> 、 <b>TCK</b> です。オプション端子は、 <b>TRST</b> です。この信号は、デバッグ回路のリセットに使用されるため、ARM コアには不可欠です。
<b>スレッド制御ブロック</b>	Thread Control Block。オペレーティングシステムのカーネルが、実行しているひとつのスレッドに固有の情報を保持するために使用するデータ構造。
<b>Thumb 命令</b>	Thumb instruction。ARM プロセッサが Thumb 状態で実行する動作を指定するハーフワード。Thumb 命令は、ハーフワードアラインしている必要があります。
<b>Thumb 状態</b>	Thumb state。Thumb (16 ビット) ハーフワードアラインド命令を実行しているプロセッサは、Thumb 状態で動作しています。
<b>TPA</b>	トレースポート アナライザ(Trace Port Analyzer)。
<b>TPIU</b>	トレースポート インタフェースユニット参照。
<b>トレースポート インタフェースユニット (TPIU)</b>	Trace Port Interface Unit (TPIU)。トレースデータを出力し、オンチップのトレースデータと、TPA によりキャプチャされるデータストリームとのブリッジとして機能します。
<b>アンアラインド</b>	Unaligned。データサイズで決まるバイト数で割り切れないアドレスに保存されたデータ項目は、アンアラインドと呼びます。例えば、4 で割り切れないアドレスに保存されているワードはアンアラインドです。
<b>UNP</b>	予測不能 (Unpredictable) 参照。
<b>予測不能</b>	Unpredictable。読み出しの場合は、この位置から読み出しによって返されるデータが予測不能なことを意味します。データはどのような値にもなり得ます。書き込みの場合は、この位置への書き込みによって予測不能な動作が発生するか、デバイスの構成に予測不能な変化が発生することを意味します。予測不能な命令によって、プロセッサまたはシステムのいずれかの部分に停止やハングが発生しないようにする必要があります。
<b>ウォームリセット</b>	Warm reset。コアリセット (core reset) と呼ばれることもあります。デバッグコントローラとデバッグ回路を除くプロセッサの大部分を初期化します。このタイプのリセットは、プロセッサのデバッグ機能を使用している場合に便利です。
<b>ウォッチポイント</b>	Watchpoint。ウォッチポイントは、デバッグで提供されている機構で、特定のメモリアドレスに保存されているデータが変更されたときにプログラムの実行を停止します。プログラマはウォッチポイントを挿入することによって、メモリが書き込まれたときの、レジスタの内容、メモリの位置、変数の値を検査して、プログラムが正常に動作しているかどうかをテストすることができます。プログラムのテストが完了した後で、ウォッチポイントは削除されます。ブレークポイント (Breakpoint) も参照して下さい。
<b>ワード</b>	Word。32 ビットのデータ項目。

**ワード不変**

**Word-invariant.** ワード不変システムでは、リトルエンディアン動作とビッグエンディアン動作の切り替え時に、各メモリバイトのアドレスが変更されます。例えば、一方のエンディアン形式でアドレス **A** が割り当てられたバイトは、他方のエンディアン形式ではアドレス **A EOR 3** が割り当てられます。このため、メモリのアラインされたワードは、エンディアン形式に関係なく、必ず、メモリ上の同じ 4 バイトに同じ順序で構成されます。エンディアン形式の切り替えは、バイト配列が変わるためではなく、バイトアドレスが変更されるために起こります。

**ARM** アーキテクチャでは、**ARMv3** およびそれ以降のバージョンでワード不変システムがサポートされています。ワード不変のサポートが選択されている場合、アンアラインドアドレスが指定されたロード命令またはストア命令の動作は命令によって異なり、通常は、アンアラインドアクセスに対して予測される動作にはなりません。ワード不変システムでは、エンディアン形式が設定される前のリセットハンドラの冒頭部分を除き、常に期待通りのバイトアドレスが生成されるエンディアン形式を使用することをお勧めします。リセットハンドラの冒頭部分では、アラインしたワードメモリアクセスのみを使用する必要があります。

バイト不変 (**Byte-invariant**) も参照して下さい。

**ライトバッファ**

**Write buffer.** 書き込みデータをバッファし、バスのストール (**stall**) によってプロセッサがストールすることを防止するためのパイプラインステージ。