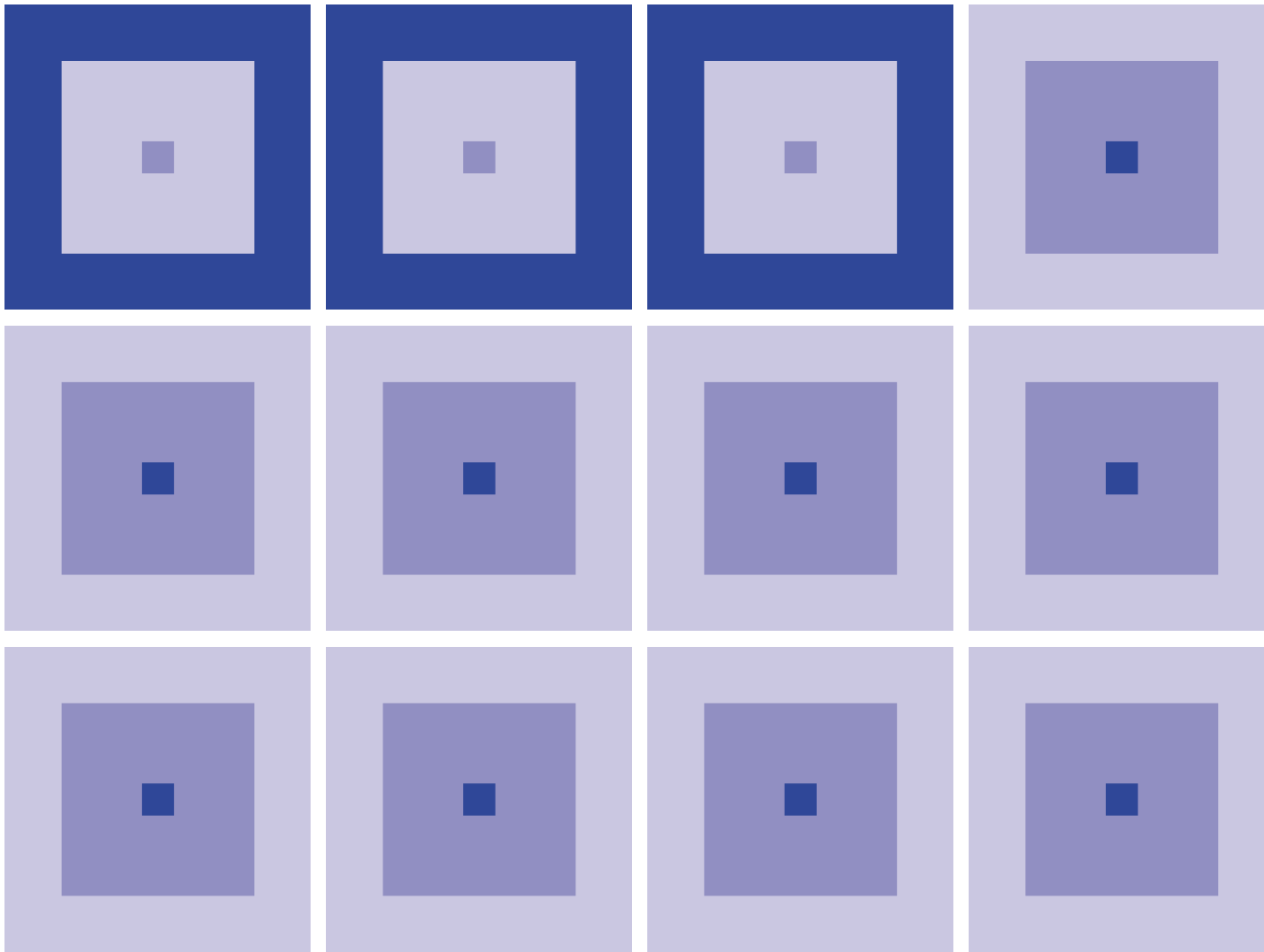
EPSON

ARM POWERED®

ARM720T Revision 4

(AMBA AHB Bus Interface Version)

コアCPUマニュアル



セイコーエプソン株式会社

本資料のご使用につきましては、次の点にご留意願います。

1. 本資料の内容については、予告なく変更することがあります。
2. 本資料の一部、または全部を弊社に無断で転載、または複製など他の目的に使用することは、堅くお断りします。
3. 本資料に掲載される応用回路、プログラム、使用方法等はあくまでも参考情報であり、これらに起因する第三者の権利（工業所有権を含む）侵害あるいは損害の発生に対し、弊社は如何なる保証を行うものではありません。また、本資料によって第三者または弊社の工業所有権の実施権の許諾を行うものではありません。
4. 特性表の数値の大小は、数直線上の大小関係で表しています。
5. 本資料に掲載されている製品のうち、「外国為替および外国貿易法」に定める戦略物資に該当するものについては、輸出する場合、同法に基づく輸出許可が必要です。
6. 本資料に掲載されている製品は、一般民生用です。生命維持装置その他、きわめて高い信頼性が要求される用途を前提としていません。よって、弊社は本（当該）製品をこれらの用途に用いた場合の如何なる責任についても負いかねます。

目次

図一覧

表一覧

このページは白紙です。

はじめに

はじめに

この「はじめに」では、*ARM720T Revision 4 (AMBA AHB Bus Interface Version) コア CPU マニュアル*を紹介します。ここでは、以下の項について記載しています。

本書について

このマニュアルは、ARM720T r4p3 プロセッサのテクニカルマニュアルです。

対象

このマニュアルは、ARM 製品に関するアーキテクチャ、コンフィギュレーション、インテグレーション、および命令セットについての知識の有無に関係なくハードウェアおよびソフトウェアの経験豊かなエンジニア向けに作成されています。本書では、設計者がプロセッサをできるだけ速やかにターゲットシステムに組み込むことができるようにするための情報を提供しています。

このマニュアルの使い方

このマニュアルは、以下の章から構成されています。

第 1 章「概要」

ARM720T プロセッサの概要を理解するには、この章をお読みください。

第 2 章「プログラマーズモデル」

32 ビット ARM 命令セットと 16 ビット Thumb 命令セットの説明については、この章をお読みください。

第 3 章「コンフィギュレーション」

ARM1156F-S 制御プロセッサ C15 レジスタ構成の説明およびプログラミングの詳細については、この章をお読みください。

第 4 章「命令・データキャッシュ」

命令・データ混在キャッシュの説明については、この章をお読みください。

第 5 章「ライトバッファ」

ライトバッファを使用して、ARM720T プロセッサのシステムパフォーマンスを拡張する方法に関する説明については、この章をお読みください。

第 6 章「バスインタフェース」

ARM720T プロセッサバスインタフェースの説明については、この章をお読みください。

第 7 章「メモリ管理ユニット」

メモリ管理ユニット（MMU）の機能および使用方法の説明については、この章をお読みください。

第 8 章「コプロセッサインタフェース」

ARM1156F-S コプロセッサインタフェースへのコプロセッサの接続方法に関する説明については、この章をお読みください。

第 9 章「システムのデバッグ」

ARM720T プロセッサのハードウェア拡張機能とインテグレーテッドオンチップデバッグサポートの説明については、この章をお読みください。

第 10 章「ETM インタフェース」

ARM720T プロセッサの組み込み型トレースマクロセルサポートの説明については、この章をお読みください。

第 11 章「テストサポート」

デバイス固有のテスト動作の実行方法に関する説明については、この章をお読みください。

付録 A「信号の説明」

ARM720T プロセッサインタフェース信号の一覧については、この付録をお読みください。

表記規則

このマニュアルでは、以下の表記規則を使用しています。

ボールド体	ARM プロセッサ信号名、およびメニュー名などのインタフェースエレメントを強調表示しています。また、適宜記述的なリストの用語にも使用しています。
イタリック体	特殊な用語、クロスリファレンス、および引用を強調表示しています。
monospace	コマンド、ファイル名やプログラム名、およびソースコードなど、キーボードから入力することができるテキストを示しています。
mono**space**	コマンドやオプションの許容省略文字を示しています。コマンドやオプションのフルネームの代わりに、下線部のテキストを入力することができます。
monospace italic	コマンドや関数の引数を示します。引数は、特定の値で置き換えられます。
monospace bold	例題コード外で使用される言語キーワードを示します。

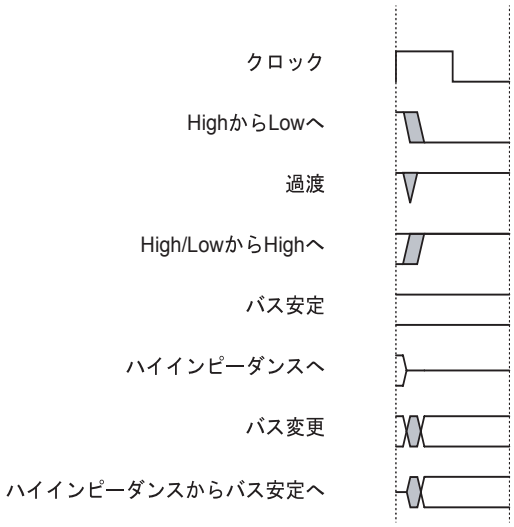
製品のレビジョン状態

r*n*p*n* 識別子は、このマニュアルで説明する製品のレビジョン状態を示します。

r*n*	製品の主レビジョンを識別します。
p*n*	製品の副レビジョンや修正状態を識別します。

タイミングチャート規則

このマニュアルでは、1 つ以上のタイミングチャートを掲載しています。以下のキーは、これらのタイミングチャートで使用される基本的な状態を説明しています。これらの状態は明らかに、いくつかに分類されます。したがって、特に記述がないかぎりは、特別な意味が追加されることはありません。



タイミングチャートキーの規則

網掛けされているバスや信号部分は定義されていませんので、バスや信号はその時の網掛け部分で任意の値を想定することができます。実際のレベルは重要ではなく、通常の動作に影響は与えません。

参考文献

この項では、ARM Limited およびサードパーティ社の出版物を掲載しています。

ARM 社では、本書のアップデートや修正を定期的に行っています。最新の正誤表、追補、ARM の FAQ（よくある質問）については、http://www.arm.com で確認してください。

ARM 社の出版物

このマニュアルは、ARM720T プロセッサに固有な情報を記載しています。他の該当情報については、以下のドキュメントを参照してください。

- *ARM Architecture Reference Manual*（ARM DDI 0100）
- *AMBA Specification（Rev 2.0）*（ARM IHI 0011）
- *ETM7 (Rev 1) Technical Reference Manual*（ARM DDI 0158）
- *ARM7TDMI-S (Rev 4) Technical Reference Manual*（ARM DDI 0234）

その他の出版物

この項では、サードパーティ社から出版されている関連ドキュメントをリストします。

- *Standard Test Access Port and Boundary Scan Architecture*（IEEE Std. 1149.1.1990）

図 9-8（9-19 ページ）は、IEEE の許可により IEEE Std. 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture Copyright 2001, by IEEE から転載されたものです。IEEE は、記載した方法により実装、使用したことによって生じるいかなる責任も負いません。

このページは白紙です。

1
概要

1 概要

この章では、ARM720T プロセッサの概要を説明します。ここでは、以下の項について記載しています。

1.1 ARM720T プロセッサについて

ARM720T プロセッサは、単一のチップに 8KB キャッシュ、拡張ライトバッファ、*メモリ管理ユニット*（MMU）を搭載した汎用 32 ビットマイクロプロセッサです。このプロセッサは ARM7TDMI-S CPU を使用しており、ARM プロセッサファミリとソフトウェアの互換性があります。

オンチップデータ・命令混在キャッシュは、ライトバッファとともに、大幅に平均実行速度を向上し、プロセッサに必要な平均メモリバンド幅を低減します。これにより、外部メモリはパフォーマンスの損失を最小限に抑えて追加プロセッサや*ダイレクトメモリアクセス*（DMA）チャンネルに対応することができます。

MMU は、従来の 2 段階ページテーブル構造と、ハイエンドな組み込みアプリケーションや複雑なオペレーティングシステムを理想的に実行させるようにするいくつかの拡張機能をサポートしています。

異なるタスク ID により仮想アドレスを割り当てることによって、キャッシュをイネーブルにしてタスク切り換え動作のパフォーマンスを向上させています。これらの再配置仮想アドレスは、EmbeddedICE-RT ブロックによって監視されます。

メモリインタフェースは、メモリシステムに高いコストをかけずにパフォーマンスの潜在能力を実現することができます。処理速度がクリティカルな制御信号はパイプライン処理を行って、システム制御機能を標準の低電力ロジックで実現できるようにしています。これらの制御信号によって、業界標準 DRAM で提供されるページモードアクセスを活用することができます。

ARM720T プロセッサには、必要な信号を ARM コアから ARM720T プロセッサの周辺部に出力する*組み込み型トレースマクロセル*（ETM）インタフェースが搭載されています。これによって、標準の ETM7 マクロセルを接続することができます。

ARM720T プロセッサは完全にスタティックな部品であり、電力消費を最小限に抑えるように設計されています。これによって、MRM720T プロセッサは低電力消費が必要不可欠な携帯型アプリケーションに理想的なものになっています。

ARM720T プロセッサのアーキテクチャは、*「縮小命令セットコンピュータ」*（RISC）の原理に基づいています。命令セットとその関連デコードメカニズムは、マイクロプログラム化された*「複雑命令セットコンピュータ」*（CISC）と比べて大幅に簡略化されています。

ARM720T プロセッサのブロック図を図 1-1 に示します。

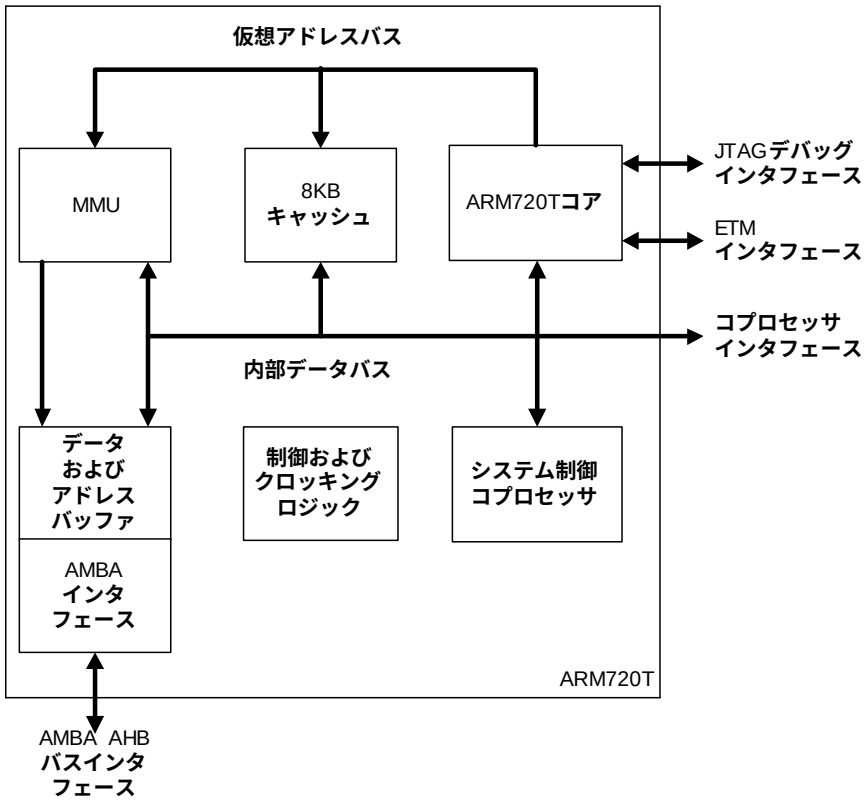


図 1-1 ARM 720T ブロック図

ARM720T プロセッサの機能信号を図 1-2 に示します。

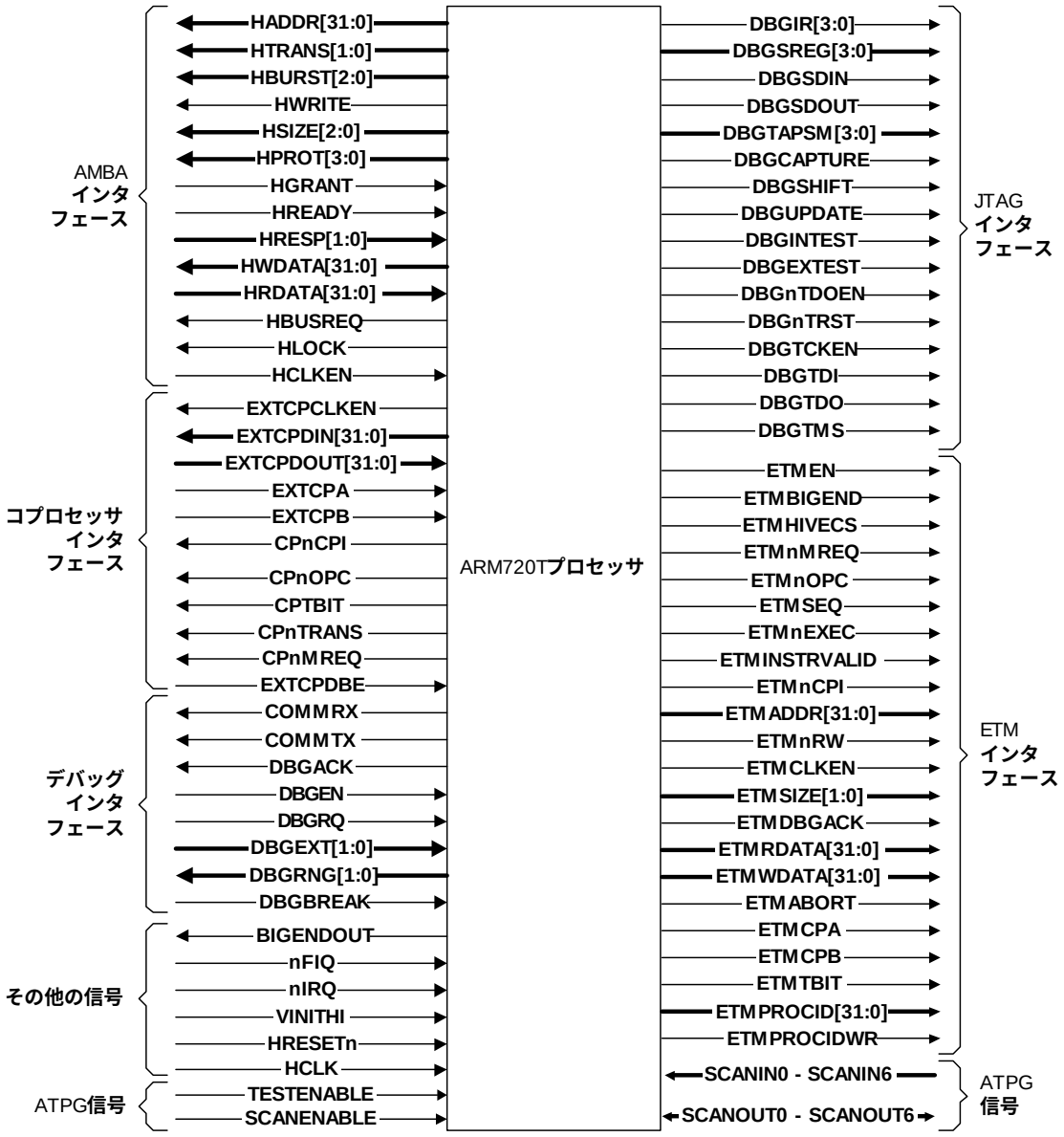


図 1-2 ARM720T **プロセッサ機能信号**

1.1.1 EmbeddedICE-RT ロジック

EmbeddedICE-RT ロジックは、ARM720T コアに対してインテグレーテッドオンチップデバッグサポートを提供します。EmbeddedICE-RT ロジックにより、ブレークポイントやウォッチポイントを発生させることができる条件をプログラムすることができます。

EmbeddedICE-RT ロジックは EmbeddedICE の拡張処理系であり、監視モードでデバッギングを実行することができます。監視モードでは、ARM720T コアはハルトモードのときのようにデバッグ状態になるのではなく、ブレークポイントやウォッチポイントで例外処理をとります。

ARM720T コアがウォッチポイントやブレークポイントを検出したときにデバッグ状態にならなければ、ハードウェア割り込み要求を通常通り継続して処理することができます。コアを停止させるとシステム傷害を招く可能性がある、機械的システムのフィードバックループの一部をコアが形成する場合は、監視モードでのデバッギングが有効です。

EmbeddedICE-RT ロジックには、*「デバッグ通信チャネル」*（DCC）があります。DCC は、ターゲットとホストデバッガ間で情報を受け渡すために使用されます。EmbeddedICE-RT ロジックは、*「ジョイントテストアクショングループ」*（JTAG）試験アクセスポートを介して制御されます。

プログラマーズモデルの変更

EmbeddedICE-RT マクロセルのサポートを提供するために、ARM720T プロセッサのプログラマーズモデルでは以下の変更が行われています。

デバッグ制御レジスタ

デバッグ制御レジスタには、以下のように新しいビットが 2 つあります。

ビット 4 監視モードイネーブル。このビットを使用して、ブレークポイントやウォッチポイントでどのようにデバイスが対応するかを制御します。

- セットすると、ARM720T コアは命令またはデータアボート例外処理をとります。
- クリアすると、ARM720T コアはデバッグ状態になります。

ビット 5 EmbeddedICE-RT ディセーブル。このビットは、ウォッチポイントやブレークポイントを変更する場合に使用します。

- セットすると、このビットはブレークポイントとウォッチポイントを無効にし、ブレークポイントレジスタやウォッチポイントレジスタを有効にして新しい値を設定することができます。
- クリアすると、新しいブレークポイント値やウォッチポイント値が動作可能になります。

このレジスタの詳細については、*「デバッグ制御レジスタ」*の項（9-40 ページ）を参照してください。

コプロセッサレジスタマップ

CP14 コプロセッサレジスタマップの新しい r2 レジスタは、実際のアボートによって、あるいはブレークポイントやウォッチポイントによってプロセッサがプリフェッチまたはデータアボート例外処理に入ったかどうかを示します。この詳細については、9-40 ページの *「アボートステータスレジスタ」*を参照してください。

詳細については、第 9 章 *「システムのデバッグ」*を参照してください。

1.2 コプロセッサ

ARM720T プロセッサには、デバイスを内部制御するための内部コプロセッサ専用 CP15 があります（第 3 章*「コンフィギュレーション」*を参照してください）。

ARM720T プロセッサには、オンチップ外部コプロセッサを接続するためのポートもあります。これによって、アーキテクチャの整合性を保ちながら ARM720T の機能を拡張させることができます。

1.3 命令セット

命令セットは、以下の 10 種類の基本命令から構成されます。

- 2 種類の命令はオンチップ算術論理ユニット、バレルシフタ、乗算器を使用して、それぞれが32ビット幅の31個のレジスタから成るバンクのデータを高速で処理します。
- 3 種類の命令は、メモリとレジスタ間のデータ転送を制御します。
  - 1 つはメモリアクセスを柔軟に最適化します。
  - 1 つは高速コンテキスト切り換え用です。
  - 1 つはデータスワッピング用です。
- 2 種類の命令は、実行のフローと特権レベルを制御します。
- 3 種類の命令は、外部コプロセッサの制御専用です。これらの命令によって、命令セットの機能をオープンで統一的にオフチップに拡張することができます。

ARM 命令セットは、さまざまな高水準言語のコンパイラに適しています。クリティカルなコードセグメントが必要な場合でも、簡単にアセンブリコードプログラミングが行えます。

1.3.1 形式概要

この項では、ARM 命令セットと Thumb（サム）命令セットの概略を説明します。

- *ARM 命令セット*、1-7 ページ
- *Thumb（サム）命令セット*、1-14 ページ

命令セットテーブルの記号を表 1-1 に示します。

ARM720T プロセッサの ARM7TDMI-S コアは、ARM アーキテクチャ v4T の処理系です。2 つの命令セットの詳細については、*「ARM Architecture Reference Manual」*を参照してください。

表 1-1 命令セットテーブルの記号

項目	説明
{cond}	表 1-11（1-13 ページ）を参照してください。
<Oprnd2>	表 1-9（1-12 ページ）を参照してください。
{field}	表 1-10（1-12 ページ）を参照してください。
S	条件コードを設定する（オプション）。
B	バイト演算（オプション）
H	ハーフワード演算（オプション）
T	アドレス変換を強制する。プリインデックスド・アドレスには使用することはできない。
<a_mode2>	表 1-3（1-10 ページ）を参照してください。
<a_mode2P>	表 1-4（1-11 ページ）を参照してください。
<a_mode3>	表 1-5（1-11 ページ）を参照してください。
<a_mode4L>	表 1-6（1-11 ページ）を参照してください。
<a_mode4S>	表 1-7（1-12 ページ）を参照してください。
<a_mode5>	表 1-8（1-12 ページ）を参照してください。
#<32bit_Imm>	8 ビット値を偶数ビット回数だけ右ロテートさせて形成された32ビット定数
<reglist>	中括弧（{ および } ）で囲まれ、カンマで区切られたレジスタのリスト

1.3.2 ARM 命令セット

この項では、使用可能な ARM 命令の概要を説明します。これらの命令の詳細については、*「ARM Architecture Reference Manual」*を参照してください。

ARM 命令セットの形式を図 1-3 に示します。

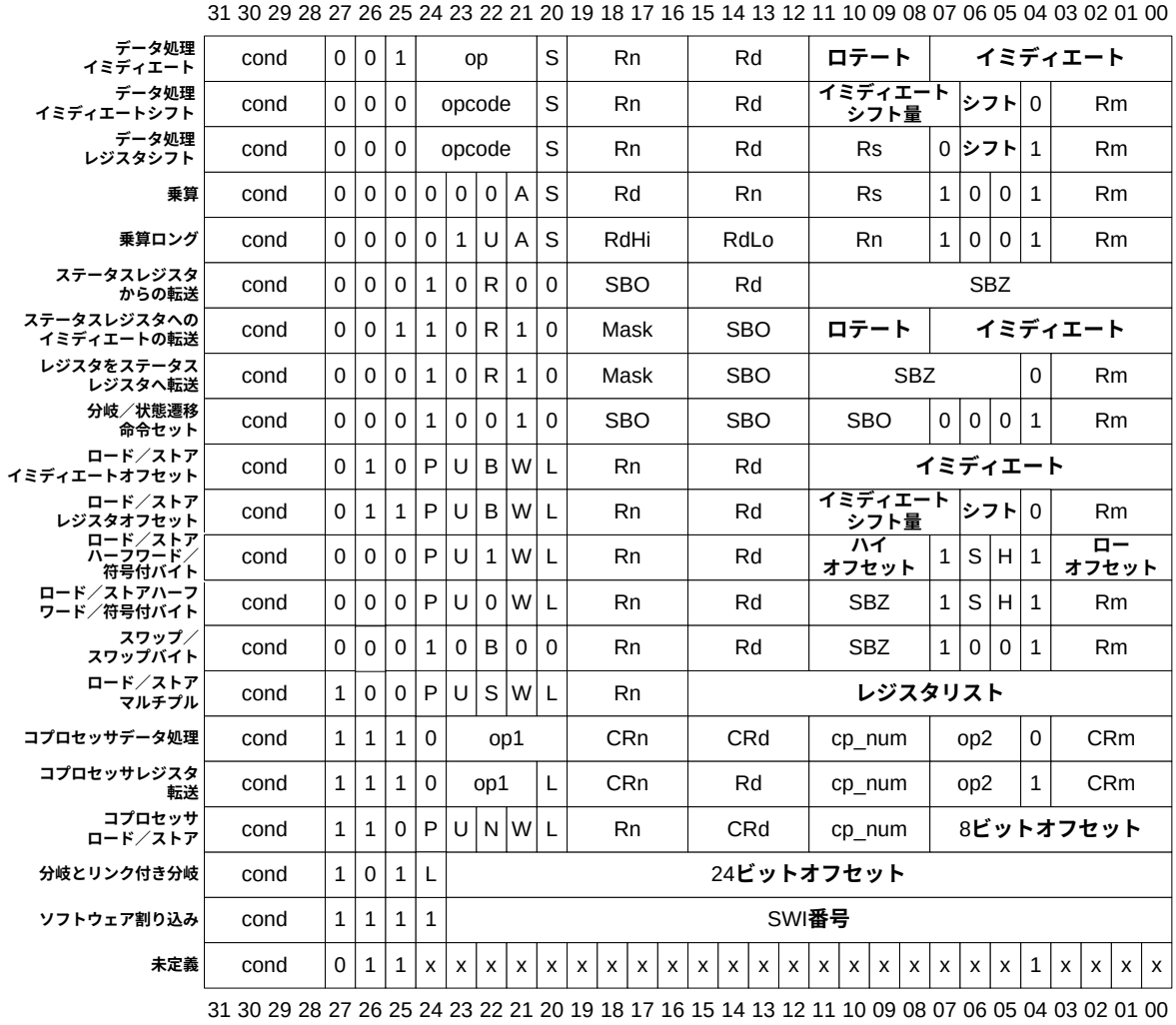


図 1-3 ARM 命令セットの形式

注意： 命令コードの中には定義されておらず、未定義命令トラップで処理されていないものがあります。たとえばビット 6 をセットした乗算命令。これらの命令の動作は将来 ARM のインプリメントで変更される可能性がありますので、使用しないでください。

ARM 命令セットのまとめを表 1-2 に示します。

表 1-2 ARM 命令一覧

動作		アセンブラ
転送	転送	MOV{cond}{S} <Rd>, <Oprnd2>
	レジスタ値の補数の転送	MVN{cond}{S} <Rd>, <Oprnd2>
	SPSR をレジスタに転送	MRS{cond} <Rd>, SPSR
	CPSR をレジスタに転送	MRS{cond} <Rd>, CPSR
	レジスタを SPSR に転送	MSR{cond} SPSR{field}, <Rm>
	レジスタを CPSR に転送	MSR{cond} CPSR{field}, <Rm>
	SPSR フラグにイミディエート定数を転送	MSR{cond} SPSR_f, #<32bit_Imm>
	CPSR フラグにイミディエート定数を転送	MSR{cond} CPSR_f, #<32bit_Imm>
算術	加算	ADD{cond}{S} <Rd>, <Rn>, <Oprnd2>
	キャリー付き加算	ADC{cond}{S} <Rd>, <Rn>, <Oprnd2>
	減算	SUB{cond}{S} <Rd>, <Rn>, <Oprnd2>
	キャリー付き減算	SBC{cond}{S} <Rd>, <Rn>, <Oprnd2>
	逆減算	RSB{cond}{S} <Rd>, <Rn>, <Oprnd2>
	キャリー付き逆減算	RSC{cond}{S} <Rd>, <Rn>, <Oprnd2>
	乗算	MUL{cond}{S} <Rd>, <Rm>, <Rs>
	積和乗算	MLA{cond}{S} <Rd>, <Rm>, <Rs>, <Rn>
	符号なしロングの乗算	UMULL{cond}{S} <RdLo>, <RdHi>, <Rm>, <Rs>
	符号なし積和ロングの乗算	UMLAL{cond}{S} <RdLo>, <RdHi>, <Rm>, <Rs>
	符号付きロングの乗算	SMULL{cond}{S} <RdLo>, <RdHi>, <Rm>, <Rs>
	符号付き積和ロングの乗算	SMLAL{cond}{S} <RdLo>, <RdHi>, <Rm>, <Rs>
	比較	CMP{cond} <Rd>, <Oprnd2>
	補数の比較	CMN{cond} <Rd>, <Oprnd2>
論理	テスト	TST{cond} <Rn>, <Oprnd2>
	等価のテスト	TEQ{cond} <Rn>, <Oprnd2>
	論理積	AND{cond}{S} <Rd>, <Rn>, <Oprnd2>
	排他的論理和	EOR{cond}{S} <Rd>, <Rn>, <Oprnd2>
	論理和	ORR{cond}{S} <Rd>, <Rn>, <Oprnd2>
	ビットクリア	BIC{cond}{S} <Rd>, <Rn>, <Oprnd2>
分岐	分岐	B{cond} <label>
	リンク付き分岐	BL{cond} <label>
	分岐、および命令セットの状態遷移	BX{cond} <Rn>

表 1-2 ARM 命令一覧（続き）

動作		アセンブラ
ロード	ワード	LDR{cond} <Rd>, <a_mode2>
	ユーザモード特権付きワード	LDR{cond}T <Rd>, <a_mode2P>
	バイト	LDR{cond}B <Rd>, <a_mode2>
	ユーザモード特権付きバイト	LDR{cond}BT <Rd>, <a_mode2P>
	符号付きバイト	LDR{cond}SB <Rd>, <a_mode3>
	ハーフワード	LDR{cond}H <Rd>, <a_mode3>
	符号付きハーフワード	LDR{cond}SH <Rd>, <a_mode3>
マルチ ブロック データ操作	ロード前インクリメント	LDM{cond}IB <Rd>{!}, <reglist>{^}
	ロード後インクリメント	LDM{cond}IA <Rd>{!}, <reglist>{^}
	ロード前デクリメント	LDM{cond}DB <Rd>{!}, <reglist>{^}
	ロード後デクリメント	LDM{cond}DA <Rd>{!}, <reglist>{^}
	スタック操作	LDM{cond}<a_mode4L> <Rd>{!}, <reglist>
	スタック操作と、CPSR リストア	LDM{cond}<a_mode4L> <Rd>{!}, <reglist+pc>^
	ユーザレジスタ	LDM{cond}<a_mode4L> <Rd>{!}, <reglist>^
ストア	ワード	STR{cond} <Rd>, <a_mode2>
	ユーザモード特権付きワード	STR{cond}T <Rd>, <a_mode2P>
	バイト	STR{cond}B <Rd>, <a_mode2>
	ユーザモード特権付きバイト	STR{cond}BT <Rd>, <a_mode2P>
	ハーフワード	STR{cond}H <Rd>, <a_mode3>
マルチ ブロック データ操作	ストア前インクリメント	STM{cond}IB <Rd>{!}, <reglist>{^}
	ストア後インクリメント	STM{cond}IA <Rd>{!}, <reglist>{^}
	ストア前デクリメント	STM{cond}DB <Rd>{!}, <reglist>{^}
	ストア後デクリメント	STM{cond}DA <Rd>{!}, <reglist>{^}
	スタック操作	STM{cond}<a_mode4S> <Rd>{!}, <reglist>
	ユーザレジスタ	STM{cond}<a_mode4S> <Rd>{!}, <reglist>^
スワップ	ワード	SWP{cond} <Rd>, <Rm>, [<Rn>]
	バイト	SWP{cond}B <Rd>, <Rm>, [<Rn>]

表 1-2 ARM 命令一覧（続き）

動作		アセンブラ
コプロセッサ	データ操作	CDP{cond} p<cpnum>, <op1>, <CRd>, <CRn>, <CRm>, <op2>
	コプロセッサレジスタから ARM レジスタへの転送	MRC{cond} p<cpnum>, <op1>, <Rd>, <CRn>, <CRm>, <op2>
	ARM レジスタからコプロセッサレジスタへの転送	MCR{cond} p<cpnum>, <op1>, <Rd>, <CRn>, <CRm>, <op2>
	ロード	LDC{cond} p<cpnum>, <CRd>, <a_mode5>
	ストア	STC{cond} p<cpnum>, <CRd>, <a_mode5>
ソフトウェア割り込み		SWI <24bit_Imm>

アドレッシングモード 2 <a_mode2> を表 1-3 に示します。

表 1-3 アドレッシングモード 2

動作	アセンブラ
イミディエートオフセット	[<Rn>, #+/-<12bit_Offset>]
レジスタオフセット	[<Rn>, +/-<Rm>]
スケーリング済みレジスタオフセット	[<Rn>, +/-<Rm>, LSL #<5bit_shift_imm>]
	[<Rn>, +/-<Rm>, LSR #<5bit_shift_imm>]
	[<Rn>, +/-<Rm>, ASR #<5bit_shift_imm>]
	[<Rn>, +/-<Rm>, ROR #<5bit_shift_imm>]
	[<Rn>, +/-<Rm>, RRX]
プリインデックスドイミディエートオフセット	[<Rn>, #+/-<12bit_Offset>]!
プリインデックスドレジスタオフセット	[<Rn>, +/-<Rm>]!
プリインデックスドスケーリング済みレジスタオフセット	[<Rn>, +/-<Rm>, LSL #<5bit_shift_imm>]!
	[<Rn>, +/-<Rm>, LSR #<5bit_shift_imm>]!
	[<Rn>, +/-<Rm>, ASR #<5bit_shift_imm>]!
	[<Rn>, +/-<Rm>, ROR #<5bit_shift_imm>]!
	[<Rn>, +/-<Rm>, RRX]!
ポストインデックスドイミディエートオフセット	[<Rn>], #+/-<12bit_Offset>
ポストインデックスドレジスタオフセット	[<Rn>], +/-<Rm>
ポストインデックスドスケーリング済みレジスタオフセット	[<Rn>], +/-<Rm>, LSL #<5bit_shift_imm>
	[<Rn>], +/-<Rm>, LSR #<5bit_shift_imm>
	[<Rn>], +/-<Rm>, ASR #<5bit_shift_imm>
	[<Rn>], +/-<Rm>, ROR #<5bit_shift_imm>
	[<Rn>, +/-<Rm>, RRX]

アドレッシングモード 2（特権付き）<a_mode2P> を表 1-4 に示します。

表 1-4 アドレッシングモード 2（特権付き）

動作	アセンブラ
イミディエートオフセット	[<Rn>, #+/-<12bit_Offset>]
レジスタオフセット	[<Rn>, +/-<Rm>]
スケーリング済みレジスタオフセット	[<Rn>, +/-<Rm>, LSL #<5bit_shift_imm>]
	[<Rn>, +/-<Rm>, LSR #<5bit_shift_imm>]
	[<Rn>, +/-<Rm>, ASR #<5bit_shift_imm>]
	[<Rn>, +/-<Rm>, ROR #<5bit_shift_imm>]
	[<Rn>, +/-<Rm>, RRX]
ポストインデックスドイミディエートオフセット	[<Rn>], #+/-<12bit_Offset>
ポストインデックスドレジスタオフセット	[<Rn>], +/-<Rm>
ポストインデックスドスケーリング済みレジスタオフセット	[<Rn>], +/-<Rm>, LSL #<5bit_shift_imm>
	[<Rn>], +/-<Rm>, LSR #<5bit_shift_imm>
	[<Rn>], +/-<Rm>, ASR #<5bit_shift_imm>
	[<Rn>], +/-<Rm>, ROR #<5bit_shift_imm>
	[<Rn>, +/-<Rm>, RRX]

アドレッシングモード 3(符号付きバイト、およびハーフワードデータ転送)<a_mode3> を表 1-5 に示します。

表 1-5 アドレッシングモード 3

動作	アセンブラ
イミディエートオフセット	[<Rn>, #+/-<8bit_Offset>]
プリインデックスド	[<Rn>, #+/-<8bit_Offset>]!
ポストインデックスド	[<Rn>], #+/-<8bit_Offset>
レジスタ	[<Rn>, +/-<Rm>]
プリインデックスド	[<Rn>, +/-<Rm>]!
ポストインデックスド	[<Rn>], +/-<Rm>

アドレッシングモード 4（ロード）<a_mode4L> を表 1-6 に示します。

表 1-6 アドレッシングモード 4（ロード）

アドレッシングモード		スタックタイプ	
IA	事後インクリメント	FD	フル下降
IB	事前インクリメント	ED	エンプティ下降
DA	事後デクリメント	FA	フル上昇
DB	事後デクリメント	EA	エンプティ上昇

アドレッシングモード 4（ストア）<a_mode4S> を表 1-7 に示します。

表 1-7 アドレッシングモード 4（ストア）

アドレッシングモード		スタックタイプ	
IA	事後インクリメント	EA	エンプティ上昇
IB	事前インクリメント	FA	フル上昇
DA	事後デクリメント	ED	エンプティ下降
DB	事前デクリメント	FD	フル下降

アドレッシングモード 5（コプロセッサデータ転送）<a_mode5> を表 1-8 に示します。

表 1-8 アドレッシングモード 5

動作	アセンブラ
イミディエートオフセット	[<Rn>, #+/-<8bit_Offset*4>]
プリインデックスド	[<Rn>, #+/-<8bit_Offset*4>]!
ポストインデックスド	[<Rn>], #+/-<8bit_Offset*4>

オペランド 2 <Oprnd2> を表 1-9 に示します。

表 1-9 オペランド 2

動作	アセンブラ
イミディエート値	#<32bit_Imm>
論理左シフト	<Rm> LSL #<5bit_Imm>
論理右シフト	<Rm> LSR #<5bit_Imm>
算術右シフト	<Rm> ASR #<5bit_Imm>
右ロテート	<Rm> ROR #<5bit_Imm>
レジスタ	<Rm>
論理左シフト	<Rm> LSL <Rs>
論理右シフト	<Rm> LSR <Rs>
算術右シフト	<Rm> ASR <Rs>
右ロテート	<Rm> ROR <Rs>
拡張右ロテート	<Rm> RRX

フィールド {field} を表 1-10 に示します。

表 1-10 フィールド

接尾辞	セット
_c	制御フィールドマスクビット（ビット 3）
_f	フラグフィールドマスクビット（ビット 0）
_s	ステータスフィールドマスクビット（ビット 1）
_x	拡張フィールドマスクビット（ビット 2）

条件フィールド {cond} を表 1-11 に示します。

表 1-11 条件フィールド

接尾辞	内容	条件
EQ	等号	Z セット
NE	不等号	Z クリア
CS	符号なし、より高いか同じ	C セット
CC	符号なし、より低い	C クリア
MI	負	N セット
PL	正またはゼロ	N クリア
VS	オーバーフロー	V セット
VC	オーバーフローなし	V クリア
HI	符号なし、より高い	C セット、Z クリア
LS	符号なし、より低いか同じ	C クリア、Z セット
GE	以上	N=V（N と V セットまたは N と V クリア）
LT	未満	N<>V（N セットと V クリア）または（N クリアと V セット）
GT	超過	Z クリア、N=V（N と V セットまたは N と V クリア）
LE	以下	Z セットまたは N<>V（N セットと V クリア）または（N クリアと V セット）
AL	常時	常時

1.3.3 Thumb（サム）命令セット

この項では、使用可能な Thumb（サム）命令セットの概要を説明します。これらの命令の詳細については、「*ARM Architecture Reference Manual*」を参照してください。

Thumb（サム）命令セットの形式を図 1-4 に示します。

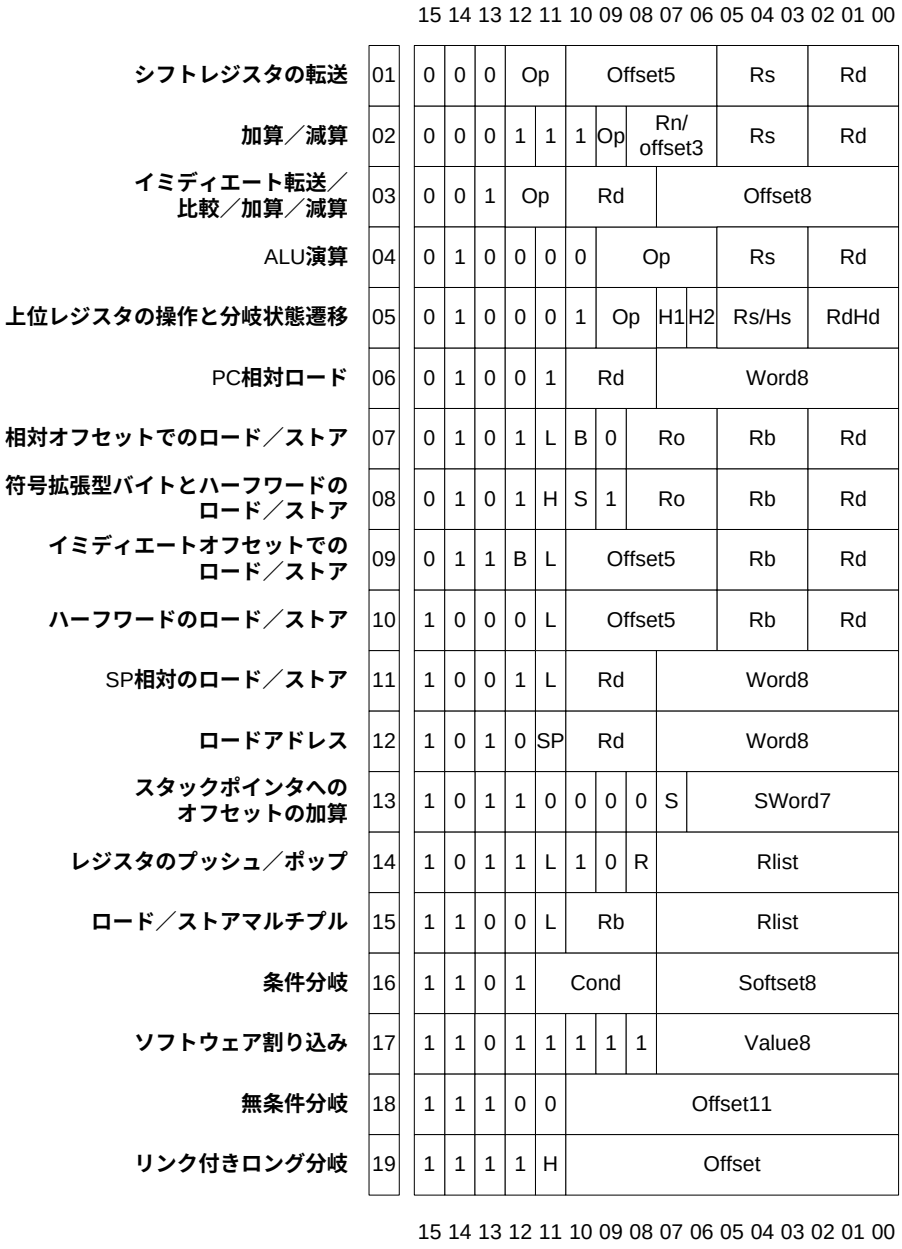


図 1-4 Thumb（サム）命令セットの形式

Thumb（サム）命令セットのまとめを表 1-12 に示します。

表 1-12 Thumb（サム）命令一覧

動作		アセンブラ
転送	イミディエート	MOV <Rd>, #<8bit_Imm>
	上位レジスタ（R8-15）から下位レジスタ（R0-7）へ	MOV <Rd>, <Hs>
	下位レジスタ（R0-7）から上位レジスタ（R8-15）へ	MOV <Hd>, <Rs>
	上位レジスタ（R8-15）から上位レジスタ（R8-15）へ	MOV <Hd>, <Hs>
算術	加算	ADD <Rd>, <Rs>, #<3bit_Imm>
	下位レジスタ（R0-7）と下位レジスタ（R0-7）の加算	ADD <Rd>, <Rs>, <Rn>
	上位レジスタ（R8-15）から下位レジスタ（R0-7）への加算	ADD <Rd>, <Hs>
	下位レジスタ（R0-7）から上位レジスタ（R8-15）への加算	ADD <Hd>, <Rs>
	上位レジスタ（R8-15）から上位レジスタ（R8-15）への加算	ADD <Hd>, <Hs>
	イミディエート値の加算	ADD <Rd>, #<8bit_Imm>
	SP への値加算	ADD SP, #<7bit_Imm> ADD SP, #-<7bit_Imm>
	キャリー付き加算	ADC <Rd>, <Rs>
	減算	SUB <Rd>, <Rs>, <Rn> SUB <Rd>, <Rs>, #<3bit_Imm>
	イミディエート値の減算	SUB <Rd>, #<8bit_Imm>
	キャリー付き減算	SBC <Rd>, <Rs>
	否定	NEG <Rd>, <Rs>
	乗算	MUL <Rd>, <Rs>
	下位レジスタ（R0-7）と下位レジスタ（R0-7）の比較	CMP <Rd>, <Rs>
	下位レジスタ（R0-7）と上位レジスタ（R8-15）の比較	CMP <Rd>, <Hs>
	上位レジスタ（R8-15）と下位レジスタ（R0-7）の比較	CMP <Hd>, <Rs>
	上位レジスタ（R8-15）と上位レジスタ（R8-15）の比較	CMP <Hd>, <Hs>
	補数の比較	CMN <Rd>, <Rs>
	イミディエート値の比較	CMP <Rd>, #<8bit_Imm>
論理	論理積	AND <Rd>, <Rs>
	排他的論理和	EOR <Rd>, <Rs>

表 1-12 Thumb（サム）命令一覧（続き）

動作			アセンブラ
		論理和	ORR <Rd>, <Rs>
		ビットクリア	BIC <Rd>, <Rs>
		レジスタ値の補数の転送	MVN <Rd>, <Rs>
		テストビット	TST <Rd>, <Rs>
シフト／ロテート		論理左シフト	LSL <Rd>, <Rs>, #<5bit_shift_imm> LSL <Rd>, <Rs>
		論理右シフト	LSR <Rd>, <Rs>, #<5bit_shift_imm> LSR <Rd>, <Rs>
		算術右シフト	ASR <Rd>, <Rs>, #<5bit_shift_imm> ASR <Rd>, <Rs>
		右ロテート	ROR <Rd>, <Rs>
分岐	条件分岐		
		Z セット時	BEQ <label>
		Z クリア時	BNE <label>
		C セット時	BCS <label>
		C クリア時	BCC <label>
		N セット時	BMI <label>
		N クリア時	BPL <label>
		V セット時	BVS <label>
		V クリア時	BVC <label>
		C セット、Z クリア時	BHI <label>
		C クリア、Z セット時	BLS <label>
		N セットと V セット時、または N クリアと V クリア時	BGE <label>
		N セットと V クリア時、または N クリアと V セット時	BLT <label>
		Z クリアおよび N または V セット時、あるいは Z クリアおよび N または V クリア時	BGT <label>
		Z セットまたは N セットと V クリアあるいは N クリアと V セット時	BLE <label>
	無条件分岐		B <label>
		リンク付きロング分岐	BL <label>
		任意の状態変化	
		下位レジスタに保持されたアドレスへ	BX <Rs>
		上位レジスタに保持されたアドレスへ	BX <Hs>

表 1-12 Thumb（サム）命令一覧（続き）

動作			アセンブラ
ロード	イミディエートオフセット付き		
		ワード	LDR <Rd>, [<Rb>, #<7bit_offset>]
		ハーフワード	LDRH <Rd>, [<Rb>, #<6bit_offset>]
		バイト	LDRB <Rd>, [<Rb>, #<5bit_offset>]
	レジスタオフセット付き		
		ワード	LDR <Rd>, [<Rb>, <Ro>]
		ハーフワード	LDRH <Rd>, [<Rb>, <Ro>]
		符号付きハーフワード	LDRSH <Rd>, [<Rb>, <Ro>]
		バイト	LDRB <Rd>, [<Rb>, <Ro>]
		符号付きバイト	LDRSB <Rd>, [<Rb>, <Ro>]
		PC 相対	LDR <Rd>, [PC, #<10bit_offset>]
		SP 相対	LDR <Rd>, [SP, #<10bit_offset>]
	アドレス		
		PC の使用	ADD <Rd>, PC, #<10bit_offset>
		SP の使用	ADD <Rd>, SP, #<10bit_offset>
		マルチプル	LDMIA Rb!, <reglist>
ストア	イミディエートオフセット付き		
		ワード	STR <Rd>, [<Rb>, #<7bit_offset>]
		ハーフワード	STRH <Rd>, [<Rb>, #<6bit_offset>]
		バイト	STRB <Rd>, [<Rb>, #<5bit_offset>]
	レジスタオフセット付き		
		ワード	STR <Rd>, [<Rb>, <Ro>]
		ハーフワード	STRH <Rd>, [<Rb>, <Ro>]
		バイト	STRB <Rd>, [<Rb>, <Ro>]
	SP 相対		STR <Rd>, [SP, #<10bit_offset>]
	マルチプル		STMIA <Rb>!, <reglist>
プッシュ／ポップ		スタックへレジスタ値をプッシュ	PUSH <reglist>
		スタックへLRおよびレジスタ値をプッシュ	PUSH <reglist, LR>
		スタックからレジスタへのポップ	POP <reglist>
		スタックからレジスタと PC へのポップ	POP <reglist, PC>
ソフトウェア割り込み			SWI <8bit_Imm>

注意： Thumb フェッチはすべて 32 ビット thumb プリフェッチバッファを使用して 32 ビットバストランザクションとして実行されます。

1.4 シリコンレビジョン

このマニュアルは、ARM720T マクロセルのレビジョン r4p3 に対応したものです。レビジョンナンバリングの詳細については、*「製品のレビジョン状態」*の項（xii ページ）を参照してください。前回のレビジョンとの機能上の違いはありません。

2
プログラマーズモデル

2 プログラマーズモデル

この章では、ARM720T プロセッサのプログラマーズモデルを説明します。ここでは、以下の項を記載しています。

2.1 プロセッサの動作状態

プログラマの視点から見た場合、ARM720T は次の 2 つのいずれかの状態になります。

ARM 状態 32 ビットのワード境界で整列した ARM 命令を実行します。

Thumb 状態 16 ビットのハーフワード境界で整列した Thumb 命令で動作します。この状態では、PC はビット 1 を使用してハーフワード間の選択を行います。

2.1.1 プロセッサ状態間の切り換え

プロセッサ状態間で遷移が行われても、プロセッサモードやレジスタの内容には影響を与えません。

Thumb 状態への移行

Thumb 状態へ移行するには、オペランドレジスタの状態ビット（ビット 0）をセットして BX 命令を実行します。

プロセッサが Thumb 状態のときに例外が発生した場合、たとえば *Interrupt ReQuest*（IRQ）、*Fast Interrupt reQuest*（FIQ）、UNDEF、ABORT、あるいは *SoftWare Interrupt*（SWI）のような例外から復帰する際にも自動的に Thumb 状態へ遷移します。

ARM 状態への移行

以下の場合、ARM 状態へ移行します。

- オペランドレジスタの状態ビットをクリアして BX 命令を実行した場合。
- プロセッサが例外を、たとえば IRQ、FIQ、RESET、UNDEF、ABORT、または SW1 を受け付けた場合。この場合、PC は例外モードのリンクレジスタに配置され、実行は例外のベクタアドレスから始まります。

2.2 メモリ形式

ARM720T プロセッサは、メモリを以下のように 0 から大きくなる方へ番号付けされたバイトの直線的な集まりとしてとらえます。

バイト 0 ～ 3　　最初に格納されたワードを保持します。

バイト 4 ～ 7　　2 番目に格納されたワードを保持します。

バイト 8 ～ 11　　3 番目に格納されたワードを保持します。

ワードは、次の項で説明するようにビッグエンディアンまたはリトルエンディアンとしてメモリに格納されます。

- *ビッグエンディアン形式*
- *リトルエンディアン形式*（2-3 ページ）

使用するエンディアンは、システム制御プロセッサの制御レジスタの B ビットの状態によって異なります。詳細については、*「制御レジスタ」*の項（3-4 ページ）を参照してください。

2.2.1 ビッグエンディアン形式

ビッグエンディアン形式では、ワードの最上位バイトが最も番号の小さいバイトに格納され、最下位バイトが最も番号の大きいバイトに格納されます。このため、メモリシステムのバイト 0 はデータライン 31 から 24 に接続されます。

ビッグエンディアン形式を図 2-1 に示します。

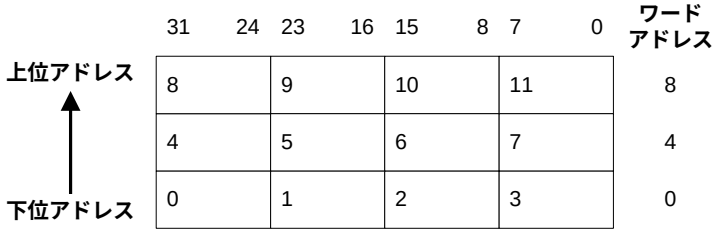


図 2-1 ワード内のバイトのビッグエンディアンアドレス

注意：

- 最上位バイトは、最も小さいアドレスに格納されます。
- ワードは、最上位バイトのバイトアドレスでアドレス指定されます。

2.2.2 リトルエンディアン形式

リトルエンディアン形式では、ワードの最も番号の小さいバイトがそのワードの最下位バイトとみなされ、最も番号の大きいバイトが最上位バイトとみなされます。このため、メモリシステムのバイト 0 はデータライン 7 から 0 に接続されます。

リトルエンディアン形式を図 2-2 に示します。

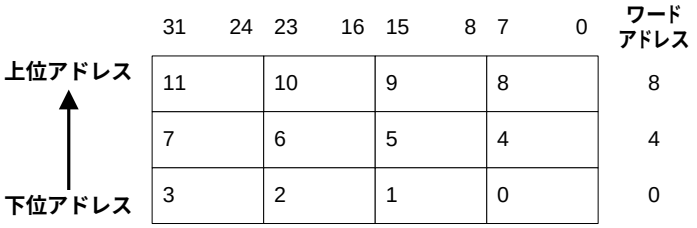


図 2-2 ワード内のバイトのリトルエンディアンアドレス

注意：

- 最下位バイトは、最も小さいアドレスに格納されます。
- ワードは、最下位バイトのバイトアドレスでアドレス指定されます。

2.3 命令長

命令は以下の長さを持ちます。

- ARM 状態では 32 ビット長
- Thumb 状態では 16 ビット長

2.4 データタイプ

ARM720T プロセッサは、以下のデータタイプをサポートしています。

- バイト（8 ビット）
- ハーフワード（16 ビット）
- ワード（32 ビット）

これらのデータタイプは、以下のように整列させます。

- ワードは 4 バイト境界に合わせる。
- ハーフワードは 2 バイト境界に合わせる。
- バイトはどのバイト境界に合わせてもよい。

2.5 動作モード

ARM720T プロセッサは、表 2-1 に示すように 7 種類の動作モードをサポートしています。

表 2-1 ARM720T の動作モード

モード	タイプ	説明
ユーザ	usr	通常の ARM プログラム実行モード
FIQ	fiq	システムで最もパフォーマンスがクリティカルな割り込みに使用されるモード
IRQ	irq	汎用割り込み処理に使用されるモード
スーパーバイザ	svc	オペレーティングシステムのプロテクトモード
アボートモード	abt	データアボートや命令プリフェッチアボート後に移行するモード
システム	sys	オペレーティングシステムの特権ユーザモード
未定義	und	未定義命令が実行されると移行するモード

2.5.1 動作モードの変更

動作モードの変更は、ソフトウェアの制御によって行ったり、外部割込みによってまたは例外処理中に行ったりすることができます。大部分のアプリケーションはユーザモードで実行されます。特権モードといわれる非ユーザモードは、割り込みや例外処理を扱ったり、保護リソースにアクセスしたりする場合に移行します。

2.6 レジスタ

ARM720T プロセッサには、以下のように全部で 37 個のレジスタがあります。

- 31 個の汎用 32 ビットレジスタ
- 6 個のプログラムステータスレジスタ

これらのレジスタがすべて同時に使用できるわけではありません。プロセッサの状態と動作モードにより、常時プログラマが使用できるレジスタが決まります。

2.6.1 ARM 状態のレジスタセット

ARM 状態では、16 個の汎用レジスタと 1 個または 2 個のステータスレジスタを常時見ることができます。特権（非ユーザ）モードでは、モード固有のバンクレジスタに切り替わります。図 2-3（2-6 ページ）に、各モードで使用可能なレジスタを示します。バンクレジスタには、網かけされた三角形の印がついています。

ARM 状態のレジスタセットには、16 個の直接アクセス可能なレジスタ r0 ～ r15 があります。r15 を除き、これらはすべて汎用レジスタであり、データまたはアドレス値を保持するために使用されます。また、r14 と r15 レジスタには以下のような特別な役割があります。

レジスタ r14 このレジスタは、サブルーチンリンクレジスタとして使用されます。このレジスタは、*リンク付き分岐*（BL）コード命令が実行されると r15 のコピーを受け取ります。それ以外の場合は、汎用レジスタとして扱われます。対応するバンクレジスタ r14_svc、r14_irq、r14_fiq、r14_abt、および r14_und は、割り込みや例外が発生した場合や、割り込みルーチンや例外ルーチン内で BL 命令が実行された場合に r15 の戻り値を保持するために同じように使用されます。

レジスタ r15 このレジスタは、*プログラムカウンタ*（PC）を保持します。ARM 状態では、r15 のビット [1:0] はゼロであり、ビット [31:2] は PC 値を格納しています。Thumb 状態では、ビット 0 はゼロであり、ビット [31:1] は PC 値を格納しています。

これらに加えて、*カレントプログラムステータスレジスタ*（CPSR）がステータス情報を格納するために使用されます。このレジスタは、条件コードフラグと現在のモードビットを格納しています。

割り込みモード

FIQ モードには、r8 ～ r14（r8_fiq ～ r14_fiq）にマップされた 7 個のバンクレジスタがあります。ARM 状態では、多数の FIQ ハンドラがこれらのバンクレジスタを使用して、レジスタをスタックに保存しなくてよいようにしています。ユーザ、IRQ、スーパーバイザ、アボート、および未定義モードには、それぞれ r13 と r14 にマップされた 2 つのバンクレジスタがあります。これらのバンクレジスタにより、これらの各モードは専用のスタックポインタとリンクレジスタを持つことができます。

図 2-3 ARM 状態でのレジスタ構成

2.6.2 Thumb 状態のレジスタセット

Thumb 状態のレジスタセットは、ARM 状態のレジスタセットのサブセットです。以下のレジスタに直接アクセスすることができます。

- 8 個の汎用レジスタ（r0 ～ r7）
- PC
- *スタックポインタ*（SP）*レジスタ*
- *リンクレジスタ*（LR）
- CPSR

各特権モードには、バンク構成の SP、LR、および*セーブプログラムステータスレジスタ*（SPSR）があります。これを図 2-4 に示します。

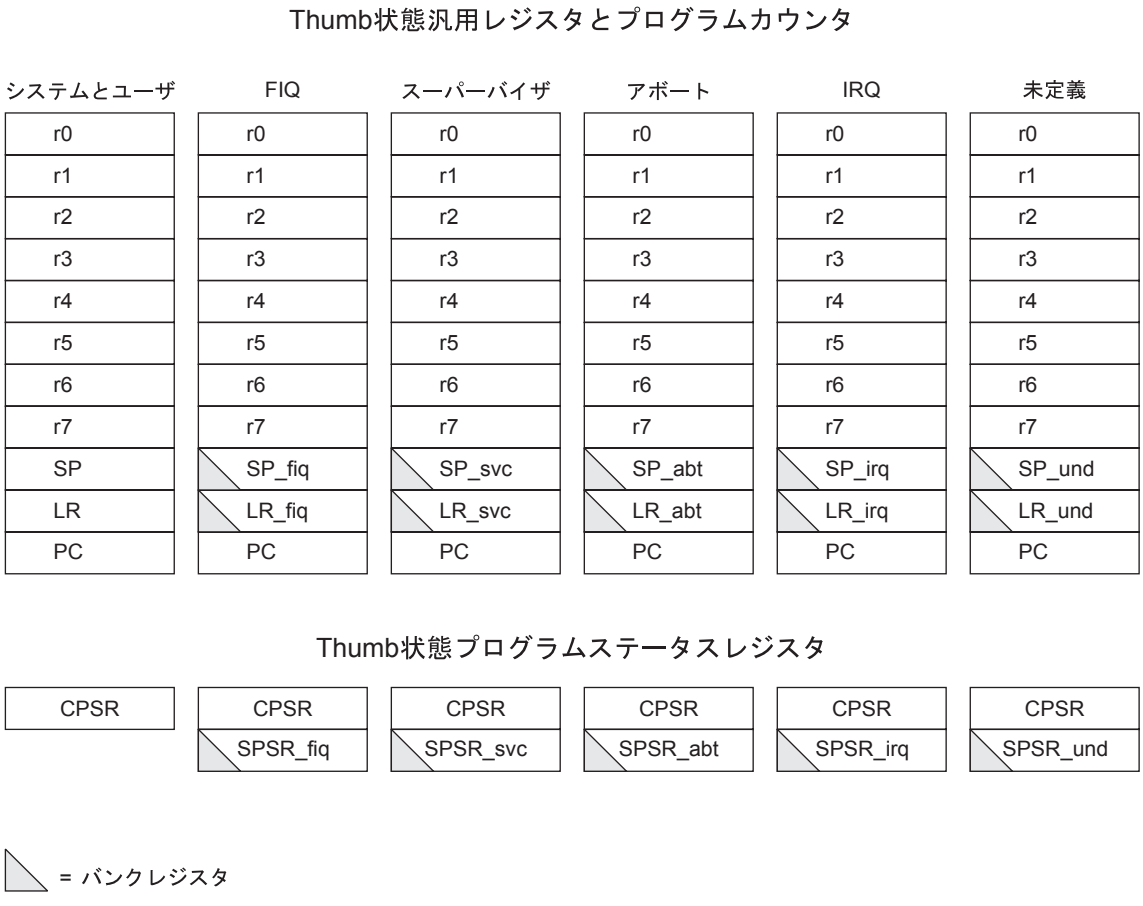


図 2-4 Thumb 状態でのレジスタ構成

2.6.3 ARM 状態レジスタと Thumb 状態レジスタ間の関係

Thumb 状態のレジスタは、以下のように ARM 状態のレジスタと関係しています。

- Thumb 状態レジスタ r0 ～ r7 と ARM 状態レジスタ r0 ～ r7 は同じです。
- Thumb 状態 CPSR、SPSR と ARM 状態 CPSR、SPSR は同じです。
- Thumb 状態 SP は、ARM 状態 r13 レジスタに対応します。
- Thumb 状態 LR は、ARM 状態 r14 レジスタに対応します。
- Thumb 状態 PC は、ARM 状態 PC（r15）に対応します。

この関係を図 2-5 に示します。

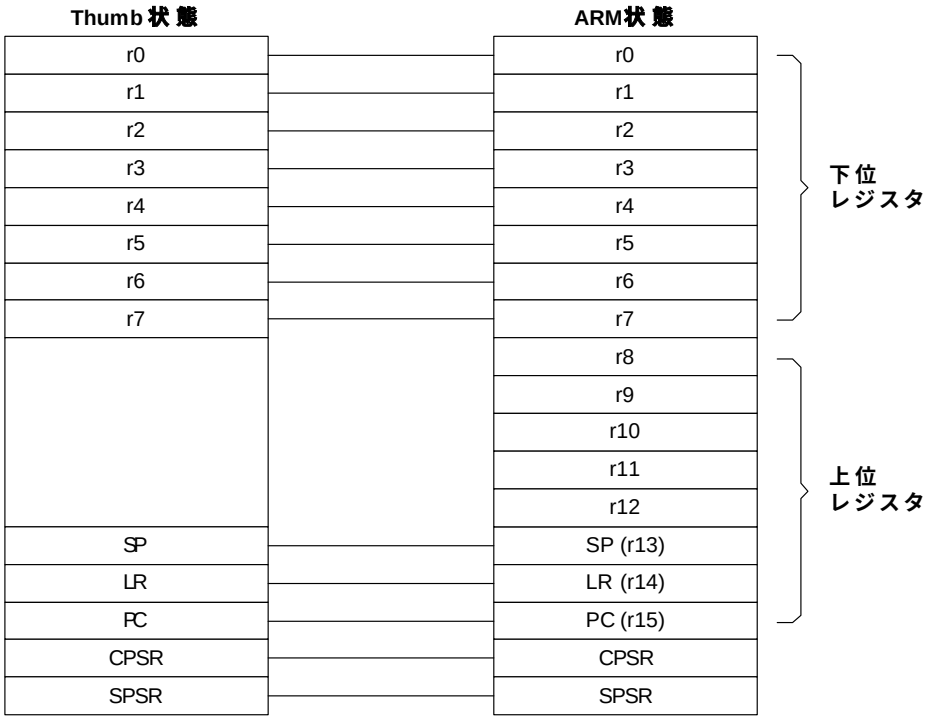


図 2-5 Thumb 状態レジスタから ARM 状態レジスタへのマッピング

2.6.4 Thumb 状態の上位レジスタへのアクセス

Thumb 状態では、ARM レジスタ r8 ～ r15（上位レジスタ）は標準のレジスタセットの一部ではありませんが、アセンブリ言語プログラマは制限付きでこれらのレジスタにアクセスして、高速の一時記憶用に使用できます。

値は、特殊な形式の MOV 命令を使用して r0 ～ r7 の範囲のレジスタ（下位レジスタ）から上位レジスタへ、また上位レジスタから下位レジスタへ値を転送することができます。上位レジスタの値は、CMP 命令や ADD 命令を使用して下位レジスタと比較したり、下位レジスタに加算したりすることもできます。上位レジスタの動作の詳細については、*「ARM Architecture Reference Manual」* を参照してください。

2.7 プログラムステータスレジスタ

ARM720T プロセッサには、CPSR と、例外ハンドラによって使用される 5 つの SPSR があります。これらのレジスタは以下の動作を行います。

- ALU 演算で実行された最新の情報を保持する。
- 割り込みのイネーブル、ディセーブルを制御する。
- プロセッサの動作モードを設定する。

図 2-6 はビット配列を示します。

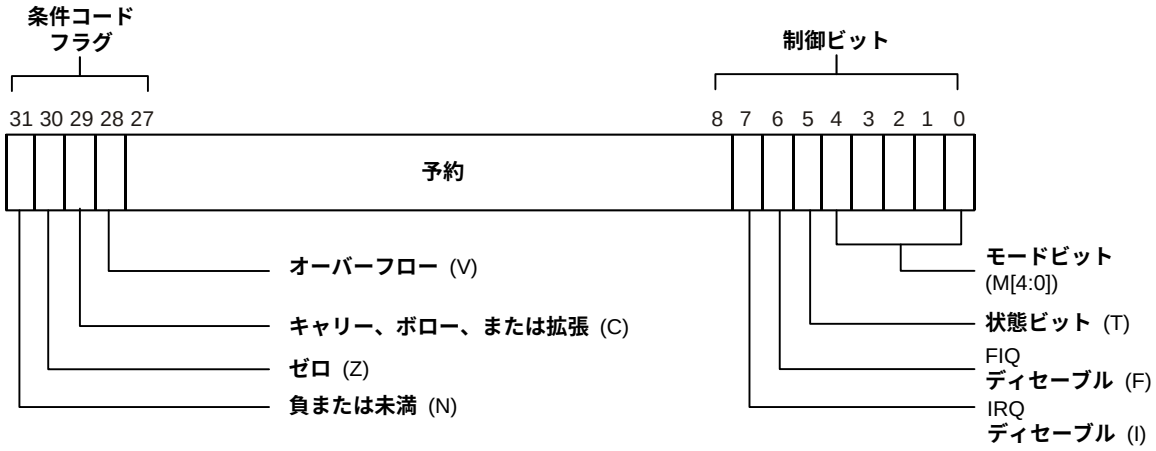


図 2-6 プログラムステータスレジスタ形式

2.7.1 条件コードフラグ

N、Z、C、V ビットは条件コードフラグです。これらのビットは、算術演算や論理演算の結果変更されたり、命令を実行すべきかどうかを決定するために調べられたりします。

ARM 状態では、すべての命令を条件付で実行できます。Thumb 状態では、分岐命令のみが条件付で実行できます。詳細については、*「ARM Architecture Reference Manual」*を参照してください。

2.7.2 制御ビット

PSR の下位 8 ビット（I、F、T、および M[4:0] を含む）は、ひとまとめに制御ビットといわれています。これらのビットは、例外が発生すると変化します。プロセッサが特権モードで動作している場合、これらのビットはソフトウェアから操作することもできます。

I、F ビット これらのビットは、割り込みディセーブルビットです。セットされると、それぞれ IRQ、FIQ 割り込みを無効にします。

T ビット このビットは動作状態を反映します。このビットがセットされると、プロセッサは Thumb 状態で実行しており、セットされていなければ ARM 状態で実行しています。これは、CPTBIT 外部信号に反映されます。ソフトウェアは、CPSR の CPTBIT の状態を変更してはいけません。変更した場合、プロセッサは予期できない状態になります。

M[4:0] ビット これらのビットはモードビットです。これらは、表 2-2（2-10 ページ）に示すようにプロセッサの動作モードを決定します。すべてのモードビットの組み合わせが有効なプロセッサモードを定義するわけではありません。明示された組み合わせのみ使用することができます。

注意： モードビット M[4:0] に不当な値を設定すると、プロセッサは予期できない状態になります。このような場合はリセットを行ってください。

2.7.3 予約ビット

PSR の残りのビットは予約されています。PSR のフラグや制御ビットを変更する場合は、これらの未使用のビットが変更されていないことを確認してください。また、将来プロセッサがこれらのビットを 1 または 0 として読み出す可能性がありますので、プログラムは特定の値を格納したこれらのビットに頼ることはできません。

表 2-2 PSR モードビット値

M[4:0]	モード	使用可能な Thumb 状態レジスタ	使用可能な ARM 状態レジスタ
b10000	ユーザ	r7 to r0, LR, SP PC, CPSR	r14 to r0, PC, CPSR
b10001	FIQ	r7 to r0, LR_fiq, SP_fiq PC, CPSR, SPSR_fiq	r7 to r0, r14_fiq..r8_fiq, PC, CPSR, SPSR_fiq
b10010	IRQ	r7 to r0, LR_irq, SP_irq PC, CPSR, SPSR_irq	r12 to r0, r14_irq, r13_irq, PC, CPSR, SPSR_irq
b10011	スーパーバイザ	r7 to r0, LR_svc, SP_svc, PC, CPSR, SPSR_svc	r12 to r0, r14_svc, r13_svc, PC, CPSR, SPSR_svc
b10111	アボート	r7 to r0, LR_abt, SP_abt, PC, CPSR, SPSR_abt	r12 to r0, r14_abt..r13_abt, PC, CPSR, SPSR_abt
b11011	未定義	r7 to r0 LR_und, SP_und, PC, CPSR, SPSR_und	r12 to r0, r14_und, r13_und, PC, CPSR, SPSR_und
b11111	システム	r7 to r0, LR, SP PC, CPSR	r14 to r0, PC, CPSR

2.8 例外

例外は、通常のプログラムフローを一時的に停止しなければならない時に必ず発生します。たとえば、周辺機器からの割り込みを処理する場合がそうです。例外を処理する前には、ハンドラルーチンが終了した時に元のプログラムを再開できるように現在のプロセッサ状態が保存されます。

複数の例外が同時に発生することがありますが、この場合は、決まった順序で処理されます。詳細については、*「例外の優先順位」*の項（2-15 ページ）を参照してください。

例外動作は、以下の項で説明しています。

- *例外開始時の動作*
- *例外終了時の動作*、2-12 ページ
- *例外の開始／終了の概要*、2-12 ページ
- *高速割り込み要求*、2-13 ページ
- *割り込み要求*、2-13 ページ
- *アボート*、2-13 ページ
- *ソフトウェア割り込み*、2-14 ページ
- *未定義命令*、2-14 ページ
- *例外ベクタ*、2-14 ページ
- *例外の優先順位*、2-15 ページ
- *例外の制限*、2-15 ページ

2.8.1 例外開始時の動作

例外を処理する場合、ARM720T プロセッサは以下のように動作します。

1 該当する LR に次の命令のアドレスを保存します。

   a. ARM 状態で例外が発生すると、次の命令のアドレスは LR（すなわち、例外により現在の PC+4 または PC+8）にコピーされます。詳細については、表 2-3（2-12 ページ）を参照してください。

   b. Thumb 状態で例外が発生すると、LR に書き込まれた値は、例外から戻ったときにプログラムが正しい場所から再開できるような値でオフセットされた現在の PC 値になります。これは、例外ハンドラが、どの状態で例外が発生したかを決める必要がないことを意味しています。

   たとえば、SWI の場合：

   ```
   MOVS PC, r14_svc
   ```

   この命令は、SWI が ARM 状態か Thumb 状態で実行されたかにかかわらず必ず次の命令に戻します。

2 CPSR の内容を該当する SPSR にコピーします。

3 強制的に CPSR モードビットを例外に対応した値にします。

4 強制的に PC に、該当する例外ベクタから次の命令をフェッチさせます。

また、制御不可能な例外のネスティングを妨ぐために割り込みディセーブルフラグを設定することもできます。

例外が発生したときにプロセッサが Thumb 状態の場合は、PC に例外ベクタアドレスがロードされると自動的に ARM 状態に切り換わります。

2.8.2 例外終了時の動作

例外終了時、例外ハンドラは以下の動作を行います。

1 必要に応じてオフセットをマイナスして、LR の内容を PC に移します。オフセットは例外のタイプによって異なります。

2 SPSR の内容を CPSR に書き戻します。

3 例外発生時に割り込みディセーブルフラグがセットされていた場合は、それらをクリアします。

注意： Thumb 状態に切り換える必要はまったくありません。それは、SPSR の内容を CPSR にリストアすることによって、T ビットが例外の直前に保持していた値に自動的にセットされるからです。

2.8.3 例外の開始／終了の概要

表 2-3 は、例外開始時に該当する r14 レジスタに保存される PC 値と、例外ハンドラを終了するための推奨命令をまとめたものです。

表 2-3 例外の開始と終了

例外	リターン命令	例外発生前の状態	
		ARM r14_x	Thumb r14_x
BL[a]	MOV PC, r14	PC + 4	PC + 2
SWI[a]	MOVS PC, r14_svc	PC + 4	PC + 2
UDEF[a]	MOVS PC, r14_und	PC + 4	PC + 2
FIQ[b]	SUBS PC, r14_fiq, #4	PC + 4	PC + 4
IRQ[b]	SUBS PC, r14_irq, #4	PC + 4	PC + 4
PABT[a]	SUBS PC, r14_abt, #4	PC + 4	PC + 4
DABT[c]	SUBS PC, r14_abt, #8	PC + 8	PC + 8
RESET[d]	NA	-	-

a. PC は、プリフェッチアボートを発生した BL、SWI、未定義命令、またはフェッチのアドレスです。

b. PC は、FIQ または IRQ が優先されたために実行されなかった命令のアドレスです。

c. PC は、データアボートを発生させたロードまたはストア命令のアドレスです。

d. リセット時に r14_svc に保存される値は、予測不可能です。

2.8.4 高速割り込み要求

FIQ 例外は、システムの中で最もパフォーマンスが重要な割り込みに使用されます。ARM 状態では、プロセッサはレジスタ退避の必要がないように十分な専用レジスタを備えていますので、コンテキスト切り換えのオーバーヘッドが最小限に抑えられます。

FIQ は、外部で **nFIQ** 入力を Low にすることにより発生します。**nFIQ** と **nIRQ** は非同期とみなされますので、割り込みがプロセッサフローに影響を及ぼす前に同期化のためのサイクル遅延が発生します。

例外が ARM 状態から、あるいは Thumb 状態から発生したかにかかわらず、FIQ ハンドラは次の命令を実行して割り込みを終了しなければなりません。

```
SUBS PC, r14_fiq, #4
```

FIQ は、CPSR の F フラグをセットすることによって無効にすることができます。

注意： この操作は、ユーザモードでは行えません。

F フラグがクリアされると、ARM720T プロセッサは各命令の終了時に FIQ 同期回路の出力が Low レベルになっているかどうかをチェックします。

2.8.5 割り込み要求

IRQ 例外は、**nIRQ** 入力が Low レベルのときに発生する通常の割り込みです。IRQ は FIQ よりも優先順位が低いため、FIQ シーケンスが開始されるとマスクされます。IRQ は CPSR の I ビットをセットすることによりいつでも無効にすることができますが、これは特権（非ユーザ）モードからしか行うことができません。

例外が ARM 状態から、あるいは Thumb 状態から発生したかどうかにかかわらず、IRQ ハンドラは次の命令を実行して割り込みから戻らなければなりません。

```
SUBS PC, r14_irq, #4
```

2.8.6 アボート

アボートは、現在のメモリアクセスを完了できないことを示しています。アボートは、プロテクションユニットか、または **HRESP** バスのいずれかによって信号を送ることができます。ARM720T プロセッサは、メモリアクセスサイクル中にアボート例外が発生していないかどうかをチェックします。

アボートには、以下の 2 つのタイプがあります。

プリフェッチアボート このアボートは、命令プリフェッチ時に発生します。プリフェッチ命令は無効にすることができますが、例外は命令がパイプラインの先頭に来るまで実行されません。たとえば、命令がパイプラインにあるときに分岐が発生したために命令が実行されなかった場合、アボートは発生しません。

データアボート このアボートは、データアクセス中に発生します。実行される動作は、以下のように命令タイプによって異なります。

- シングルデータ転送命令（LDR、STR）は、変更されたベースレジスタを書き戻します。アボートハンドラは、このことを認識していなければなりません。
- スワップ命令（SWP）は、あたかも実行されなかったかのようにアボートされます。
- ブロックデータ転送命令（LDM、STM）は完了します。ライトバックが設定されると、ベースは更新されます。命令がデータによってベースを上書きしようとすると（すなわち、転送リストにベースがあると）、上書きは阻止されます。アボートが示された後、すべてのレジスタの上書きは行われません。これは、特に r15（常に、最後に転送されるレジスタ）は LDM 命令がアボートされても保存されることを意味しています。

アボートの原因が解決された後、ハンドラはプロセッサの状態（ARM または Thumb）にかかわらず以下の命令を実行しなければなりません。

プリフェッチアボートの場合： SUBS PC, r14_abt, #4

データアボートの場合： SUBS PC, r14_abt, #8

これによって、PC と CPSR の両方がリストアされ、アボートされた命令を再試行します。

注意： 外部アボート信号の使用に関しては制限があります。詳細については、*「外部アボート」*の項（7-21 ページ）を参照してください。

2.8.7 ソフトウェア割り込み

SWI 命令はスーバーバイザモードを開始するために使用され、通常特定のスーパーバイザ機能を要求します。SWI ハンドラは、プロセッサの状態（ARM または Thumb）にかかわりなく以下の命令を実行することによって復帰しなければなりません。

```
MOV PC, r14_svc
```

これによって PC と CPSR の両方がリストアされ、SWI の次の命令に復帰します。

2.8.8 未定義命令

ARM720T プロセッサは処理することができない命令を検出した場合、未定義命令トラップを使用します。このメカニズムを使用して、ソフトウェアエミュレーションによって設定されている Thumb 命令か ARM 命令のいずれかを拡張することができます。

実行に失敗した命令をエミュレートした後、トラップハンドラはプロセッサの状態（ARM または Thumb）にかかわりなく以下の命令を実行しなければなりません。

```
MOVS PC, r14_und
```

これによって CPSR がリストアされ、未定義命令の次の命令に復帰します。

2.8.9 例外ベクタ

ARM720T プロセッサは、システム制御コプロセッサの制御レジスタの V ビットで制御される下位アドレスか上位アドレスのいずれかにマップされた例外ベクタを持つことができます（*「制御レジスタ」*の項（3-4 ページ）を参照してください）。表 2-4 に、例外ベクタアドレスを示します。

表 2-4 例外ベクタアドレス

上位アドレス	下位アドレス	例外	エントリーモード
0xFFFF0000	0x00000000	リセット	スーパーバイザ
0xFFFF0004	0x00000004	未定義命令	未定義
0xFFFF0008	0x00000008	ソフトウェア割り込み	スーパーバイザ
0xFFFF000C	0x0000000C	アボート（プリフェッチ）	アボート
0xFFFF0010	0x00000010	アボート（データ）	アボート
0xFFFF0014	0x00000014	予約	予約
0xFFFF0018	0x00000018	IRQ	IRQ
0xFFFF001C	0x0000001C	FIQ	FIQ

注意： 下位アドレスはデフォルトです。

2.8.10 例外の優先順位

複数の例外が同時に発生した場合、決められた優先順位システムによって処理される順番は以下のように決まっています。

1 リセット（最も優先順位が高い）
2 データアボート
3 FIQ
4 IRQ
5 プリフェッチアボート
6 未定義命令、SWI （最も優先順位が低い）

2.8.11 例外の制限

未定義命令と SWI は同時に指定できません。それぞれが現在の命令の特定の（非オーバラッピング）デコーディングに対応しているためです。

データアボートが FIQ と同時に発生し、かつ FIQ が有効になっている、すなわち CPSR の F フラグがクリアされている場合、ARM720T プロセッサはデータアボートハンドラを開始し、その後すぐに FIQ ベクタ処理に移ります。FIQ から正常復帰すると、データアボートハンドラは実行を再開します。データアボートの優先順位を FIQ よりも高く設定して、転送エラーの検出が確実に行われるようにする必要があります。この例外開始に要する時間は、最悪の場合を想定した FIQ レイテンシの計算に加えてください。

2.9 FCSE PID による下位仮想アドレスの再配置

ARM720T プロセッサは*高速コンテキスト切り換え拡張機能*（FCSE）というメカニズムを提供して、*FCSE プロセス識別子*（PID）の現在の値に基づいて仮想アドレスを物理アドレスに変換します。

プロセッサコアによって生成される IDC、MMU への仮想アドレスは、仮想アドレスの下位 32MB に存在する場合再配置させることができます。すなわち、CP15 コプロセッサの FCSE PID レジスタの 7 ビット [31:25] で置き換えることによって、仮想アドレスビット [31:25] = b0000000 になります。

以下の場合、FCSE PID が変更されると、遅延分岐と同様な動作を示します。

- FCSE PID を変更するための命令のすぐ後でフェッチされた 2 つの命令は、以前の FCSE PID に再配置されてフェッチされる。
- フェッチ中の命令のアドレスは、再配置するアドレスの範囲内にある。

リセット時、FCSE PID レジスタビット [31:25] は b0000000 にセットされ、すべての再配置を無効にします。このことから、下位アドレスリセット例外ベクタは、このメカニズムでは効果的に再配置されません。

注意： プロセッサコアによって生成されるすべてのアドレスは、適切なアドレスレンジに入っている限りこの変換にかけられます。例外ベクタが仮想メモリマップの下位に来るように構成されている場合は、例外ベクタもこれに含まれます。この構成は、CP15 制御レジスタ c1 の V ビットによって決定されます。

2.10 リセット

HRESETn 信号が Low になると、ARM720T プロセッサは以下の動作を行います。

1. 実行中の命令を中止します。
2. キャッシュと*トランスレーションルックアサイドバッファ*（TLB）をフラッシュします。
3. *ライトバッファ*（WB）、キャッシュ、MMU をディセーブルにします。
4. FCSE PID をリセットします。
5. インクリメントするワードアドレスから命令フェッチを継続します。

HRESETn 信号が Low になると、プロセッサは **VINITHI** 外部入力をサンプリングし、その結果を CP15 レジスタ 1 の V ビットに格納します。

HRESETn 信号が再度 High になると、ARM720T プロセッサは以下の動作を行います。

1. PC と CPSR の現在の値をコピーして r14_svc と SPSR_svc を上書きします。保存された PC と SPSR の値は、定義されていません。
2. 強制的に M[4:0] を b10011 にし（スーパーバイザモード）、CPSR の I ビットと F ビットをセットし、CPSR の T ビットをクリアします。
3. 強制的に PC にリセット例外ベクタから次の命令をフェッチさせます。例外ベクタは、CP15 レジスタ 1 の V ビット状態に応じて上位アドレスか下位アドレスのいずれかに再配置されます（LOW = 下位アドレス、HIGH = 上位アドレス）。
4. ARM 状態で実行を再開します。

2.11 インプリメンテーション時定義の命令動作

*"ARM Architecture Reference Manual"*では、ARM720T プロセッサの命令セットが定義されています。

- *"ARM Architecture Reference Manual"*でインプリメンテーション時定義としてみなされる命令の動作については、*「データアボート時のインデックスドアドレッシング」*の項を参照してください。
- ARM720T プロセッサでの符号付き早期終了、符号なし早期終了を定義するこれらの機能については、*「早期終了」*の項を参照してください。

2.11.1 データアボート時のインデックスドアドレッシング

プリインデックスドアドレッシングやポストインデックスドアドレッシングでデータアボートが発生した場合、Rn にある値は、以下の命令について更新されたベースレジスタ値として定義されます。

- LDC
- LDM
- LDR
- LDRB
- LDRBT
- LDRH
- LDRSB
- LDRSH
- LDRT
- STC
- STM
- STR
- STRB
- STRBT
- STRH
- STRT.

2.11.2 早期終了

ARM720T では、早期終了を以下のように定義しています。

MLA、MUL	符号付き早期終了
SMULL、SMLAL	符号付き早期終了
UMULL, UMLAL	符号なし早期終了

3
コンフィギュレーション

3 コンフィギュレーション

この章では、ARM720T プロセッサのコンフィギュレーションについて説明します。ここでは、以下の項を記載しています。

3.1 コンフィギュレーションについて

ARM720T プロセッサの動作とコンフィギュレーションは、以下のように制御されます。

- 直接、システム制御コプロセッサ CP15 へのコプロセッサ命令を使用
- 間接的に MMU ページテーブルを使用

コプロセッサ命令は、以下のコンフィギュレーションを制御する複数の内蔵レジスタを操作します。

- キャッシュ
- ライトバッファ
- MMU
- 他のコンフィギュレーションオプション

3.1.1 互換性

将来の CPU との下位互換性を確保するために、以下のようにしてください。

- レジスタおよびコプロセッサ命令の予約ビットや未使用ビットはすべて “0” に設定してください。
- 無効なレジスタは、読み出したり書き込んだりしないでください。
- 以下のビットは “0” に設定してください。
  - レジスタ 1 のビット [31:14] とビット [12:10]
  - レジスタ 2 のビット [13:0]
  - レジスタ 5 のビット [31:9]
  - レジスタ 7 のビット [31:0]
  - レジスタ 13 FCSE PID のビット [24:0]

3.1.2 表記法

この項では、以下の用語と省略形を使用します。

予測不能（UNP） リード指定された場合、この位置から読み出されたときに返されるデータは予測できません。データはどんな値でも持つことができます。ライト指定された場合、この位置に書き込みを行うとデバイスコンフィギュレーションで予測不能な動作をしたり、変化が生じたりします。

ゼロ指定（SBZ） この位置に書き込む場合、このフィールドのビットはすべて “0” でなければなりません。

3.2 内部コプロセッサ命令

ARM720T プロセッサの命令セットによって、コプロセッサを使用する特殊な追加命令をインプリメントすることができます。これらは ARM720T プロセッサに連結された独立した処理ユニットです。ただし、CP15 は ARM720T プロセッサに組み込まれています。

注意： CP15 レジスタマップは将来の ARM プロセッサでは変更される可能性がありますので、CP15 をアクセスするコードが単一のモジュールに格納されるようにソフトウェアを構築して、容易にアップデートできるようにしておくことを強くお勧めします。

CP15 レジスタは、特権モードで MRC 命令や MCR 命令でのみアクセスすることができます。MRC と MCR 命令の命令ビットパターンを図 3-1 に示します。

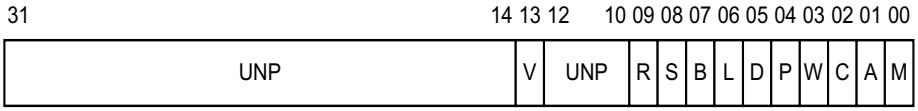


図 3-1 MRC と MCR ビットパターン

CCP15 に対する非特権の MRC や MCR 命令と同様に CDP、LDC、STC 命令を使用すると、未定義命令トラップとして処理されます。

MRC 命令、MCR 命令の CRn フィールドは、アクセスするコプロセッサレジスタを指定します。CRm フィールドと opcode_2 フィールドは、一部のレジスタをアドレッシングするときに特定の動作を指定します。

CP15 にアクセスするすべての命令では、以下のようにしてください。

- opcode_1 フィールドは、ゼロでなければなりません（SBZ）。
- opcode_2 と CRm フィールドは、指定値を使用して希望するキャッシュ、TLB、あるいはプロセス識別子動作を選択するときにレジスタ 7、8、13 をアクセスするとき以外はゼロでなければなりません。

3.3 レジスタ

ARM720T プロセッサは、キャッシュや MMU 動作を制御するレジスタを備えています。これらのレジスタは、プロセッサが特権モードの状態で CP15 への MCR 命令、MRC 命令を使用してアクセスすることができます。

表 3-1 に、有効な CP15 レジスタの概要を示します。無効なレジスタのリード／ライト行うと予測不能な動作を引き起こしますので、絶対に行わないでください。

表 3-1 キャッシュと MMU 制御レジスタ

レジスタ	レジスタリード	レジスタライト
0	ID レジスタ	予約
1	制御レジスタ	制御レジスタ
2	変換テーブルベースレジスタ	変換テーブルベースレジスタ
3	ドメインアクセス制御レジスタ	ドメインアクセス制御レジスタ
4	予約	予約
5	フォールトステータスレジスタ	フォールトステータスレジスタ
6	フォールトアドレスレジスタ	フォールトアドレスレジスタ
7	予約	キャッシュ動作レジスタ
8	予約	TLB 動作レジスタ
9 – 12	予約	予約
13	プロセス識別子レジスタ	プロセス識別子レジスタ
14	予約	予約
15	テストレジスタ	テストレジスタ

3.3.1 ID レジスタ

CP15 レジスタ 0 からリードを行うと、次の値が返されます。

0x41807204

注意： 最後のニブル（4 ビット）は、コアレビジョンを表しています。

CP15 レジスタ 0 をリードする場合、CRm および opcode_2 フィールドはゼロでなければなりません。ID レジスタリードフォーマットを図 3-2 に示します。

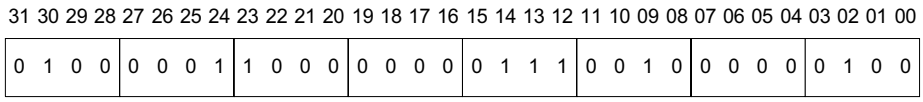


図 3-2 ID レジスタリードフォーマット

CP15 レジスタ 0 にライトを行うと、予測不能な動作になります。ID レジスタライトフォーマットを図 3-3 に示します。

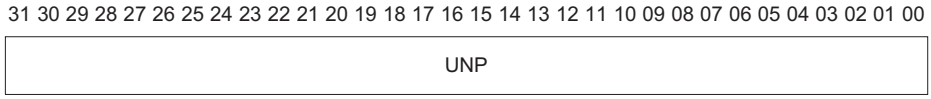


図 3-3 ID レジスタライトフォーマット

3.3.2 制御レジスタ

CP15 レジスタ 1 からリードを行うと、制御ビットが読み出されます。CP15 レジスタ 1 をリードする場合、CRm および opcode_2 フィールドはゼロでなければなりません。制御レジスタリードフォーマットを図 3-4 に示します。

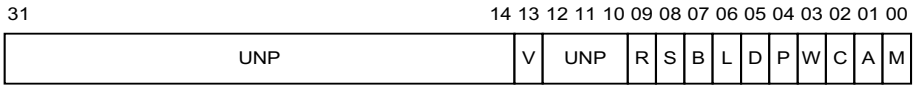


図 3-4 制御レジスタリードフォーマット

CP15 レジスタ 1 にライトを行うと、制御ビットがセットされます。CP15 レジスタ 1 にライトを行う場合、CRm および opcode_2 フィールドはゼロでなければなりません。制御レジスタライトフォーマットを図 3-5 に示します。

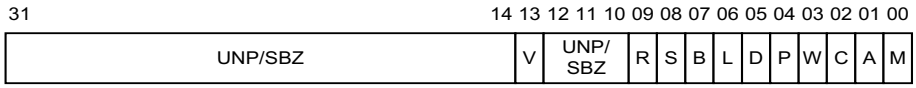


図 3-5 制御レジスタライトフォーマット

V ビット以外のすべての定義済み制御ビットは、リセット時ゼロにセットされます。これらの制御ビットには、以下の機能があります。

M ビット 0 MMU イネーブル／ディセーブル：
0 ＝ MMU ディセーブル
1 ＝ MMU イネーブル

A ビット 1 アラインメントフォールトイネーブル／ディセーブル：
0 ＝アドレスアラインメントフォールト検査ディセーブル
1 ＝アドレスアラインメントフォールト検査イネーブル

C ビット 2 キャッシュイネーブル／ディセーブル：
0 ＝命令および／またはデータキャッシュ（IDC）ディセーブル
1 ＝命令および／またはデータキャッシュ（IDC）イネーブル

W ビット 3 ライトバッファイネーブル／ディセーブル：
0 ＝ライトバッファディセーブル
1 ＝ライトバッファイネーブル

P ビット 4 リードすると 1 を返し、ライトは無視されます。

D ビット 5 リードすると 1 を返し、ライトは無視されます。

L ビット 6 リードすると 1 を返し、ライトは無視されます。

B ビット 7 ビッグエンディアン／リトルエンディアン：

0 ＝リトルエンディアン動作

1 ＝ビッグエンディアン動作

S ビット 8 システム保護：MMU 保護システムを変更します。

R ビット 9 ROM 保護：MMU 保護システムを変更します。

ビット 12:10 リードすると、予測不能な値を返します。ライトされた場合、0 か、同じプロセッサのこれらのビットからリードした値でなければなりません。

注意： このレジスタを変更する場合にリードライトモディファイシーケンスを使用すると、将来の互換性を最大限もたせることができます。

V ビット 13 例外ベクタの位置：

0 ＝下位アドレス

1 ＝上位アドレス

V ビットの値は、**HRESETn** が Low の間にサンプリングされた **VINITHI** 外部入力の状態を反映します。

ビット 31:14 リードすると、予測不能な値を返します。ライトされた場合、0 か、同じプロセッサのこれらのビットからリードした値でなければなりません。

MMU のイネーブル

変換されたアドレスが未変換のアドレスと異なる場合は、MMU のイネーブルの後に続く命令がアドレス変換を使用しないでフェッチされますので注意してください。MMU のイネーブルは、遅延実行のある分岐とみなすことができます。

MMU をディセーブルにした場合も、同様な状況が起こります。MMU をイネーブル／ディセーブルにするための適切なコードシーケンスは、*「MMU とキャッシュの相互作用」*の項（7-21 ページ）に説明しています。

注意：

- MMU がディセーブルされると、キャッシュもディセーブルされます。
- MMU がイネーブルになっていない場合にキャッシュとライトバッファがイネーブルになると、予測不能な結果を招きます。

3.3.3 変換テーブルベースレジスタ

CP15 レジスタ 2 からリードを行うと、ビット [31:14] の現在アクティブな第 1 レベル変換テーブルのポインタおよびビット [13:0] の予測不能な値が返されます。CP15 レジスタ 2 をリードする場合、CRm および opcode_2 フィールドはゼロでなければなりません。

CP15 レジスタ 2 にライトを行うと、現在アクティブな第 1 レベル変換テーブルのポインタを、ライトされた値の [31:14] ビットの値から更新します。ビット [13:0] は 0 でなければなりません。CP15 レジスタ 2 をライトする場合、CRm および opcode_2 フィールドはゼロでなければなりません。変換テーブルベースレジスタの形式を図 3-6 に示します。

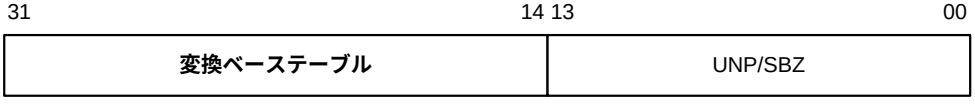


図 3-6 変換テーブルベースレジスタの形式

3.3.4 ドメインアクセス制御レジスタ

CP15 レジスタ 3 からリードを行うと、ドメインアクセス制御レジスタの値を返します。

CP15 レジスタ 3 にライトを行うと、ドメインアクセス制御レジスタの値をライトします。

ドメインアクセス制御レジスタは 16 個の 2 ビットフィールドから成り、各フィールドは 16 個のドメイン（D15 ～ D0）のうちの 1 つのアクセス許可を定義しています。

CP15 レジスタ 3 をリードしたり、ライトする場合、CRm および opcode_2 フィールドはゼロでなければなりません。ドメインアクセス制御レジスタの形式を図 3-7 に示します。

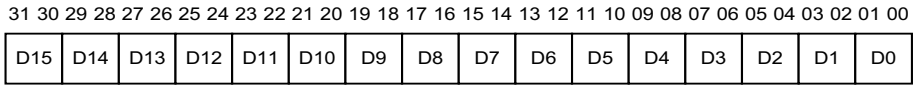


図 3-7 ドメインアクセス制御レジスタの形式

3.3.5 フォールトステータスレジスタ

CP15 レジスタ 5 からリードを行うと、*フォールトステータスレジスタ*（FSR）の値を返します。FRS は、最後のフォルトのソースを含んでいます。

注意： 下位 9 ビットの値だけが返されます。上位 23 ビットの値は予測不能です。

FSR は、アボートが発生したときに試みていたアクセスのドメインとタイプを示します。

ビット 8 このビットは常に 0 としてリードされます。ビット 8 はライト時には無視されます。

ビット [7:4] これらのビットは、フォルトが発生した場合に 16 のドメイン（D15 ～ D0）のうちのどれがアクセスされていたかを指定します。

ビット [3:1] これらのビットは、試行されていたアクセスのタイプを示します。

これらのビットのエンコーディングを、*「フォールトアドレスとフォールトステータスレジスタ」*の項（7-16 ページ）に示します。FAR は、データフォルトに対してのみ更新されます。プリフェッチフォルトの場合は更新されません。

CP15 レジスタ 5 にライトを行うと、FSR はライトされたデータの値に設定されます。これは、デバッガが FSR の値をリストアしなければならない場合に有用です。ライトされる上位 24 ビットは、ゼロでなければなりません。

CP15 レジスタ 5 をリードしたり、ライトする場合、CRm および opcode_2 フィールドはゼロでなければなりません。フォールトステータスレジスタの形式を図 3-8 に示します。

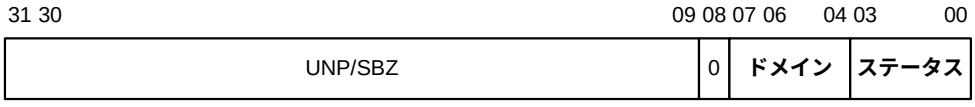


図 3-8 フォールトステータスレジスタの形式

3.3.6 フォールトアドレスレジスタ

CP15 レジスタ 6 からリードを行うと、*フォールトアドレスレジスタ*（FAR）の値を返します。FAR は、障害が発生したときに試みていたアクセスの仮想アドレスを保持しています。FAR は、データフォルトに対してのみ更新されます。プリフェッチフォルトの場合は、更新されません。

CP15 レジスタ 6 にライトを行うと、FAR はライトされたデータの値に設定されます。これは、デバッガが FAR の値をリストアしなければならない場合に有用です。

CP15 レジスタ 6 をリードしたり、ライトする場合、CRm および opcode_2 フィールドはゼロでなければなりません。フォールトアドレスレジスタの形式を図 3-9 に示します。

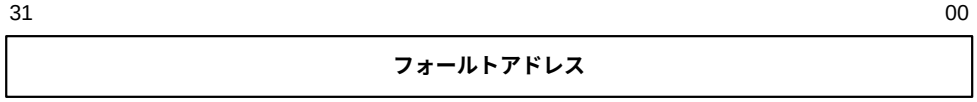


図 3-9 フォールトアドレスレジスタの形式

注意： レジスタ 6 は、FCSE PID レジスタがゼロ以外の値であれば、修正仮想アドレスを格納しています。

3.3.7 キャッシュ動作レジスタ

CP15 レジスタ 7 へのライトは、ARM720T のユニファイド（命令およびデータ）キャッシュを管理します。CP15 レジスタ 7 にライトを行う MCR 命令の以下の opcode_2 と CRm フィールドを使用して、1 つのキャッシュ動作のみを定義します。

注意： 無効 ID キャッシュ機能はキャッシュデータをすべて無効にしますので、この機能は注意して使用してください。

レジスタ 7 を表 3-2 に示します。

表 3-2 キャッシュ動作

機能	opcode_2 値	CRm 値	データ	命令
無効 ID キャッシュ	b000	b0111	SBZ	MCR p15, 0, <Rd>, c7, c7, 0

CP15 レジスタ 7 からのリードは未定義です。

3.3.8 TLB 動作レジスタ

CP15 レジスタ 8 へのライトは、*トランスレーションロックサイドバッファ*（TLB）を制御します。ARM720T プロセッサは、ユニファイド（命令とデータ）TLB をインプリメントします。

2 つの TLB 動作が定義されます。実行する機能は、CP15 レジスタ 8 の書き込みに使用された MCR 命令の opcode_2 と CRm フィールドによって選択されます。

使用することができる TLB 動作と命令を表 3-3 に示します。

表 3-3 TLB 動作

機能	opcode_2 値	CRm 値	データ	命令
無効 TLB	b000	b1000	SBZ	MCR p15, 0, <Rd>, c8, c5, 0 MCR p15, 0, <Rd>, c8, c6, 0 MCR p15, 0, <Rd>, c8, c7, 0
無効 TLB シングルエントリ	b001	b1000	修正仮想アドレス	MCR p15, 0, <Rd>, c8, c5, 1 MCR p15, 0, <Rd>, c8, c6, 1 MCR p15, 0, <Rd>, c8, c7, 1

表 3-3 に示す命令では、c7 はユニファイド MMU を示すので、CRn フィールドの優先値になります。

CP15 レジスタ 8 からのリードは未定義です。

無効 TLB シングルエントリ機能は、Rd で指定される*修正仮想アドレス*（MVA）に対応する TLB エントリをすべて無効にします。

3.3.9 プロセス ID レジスタ

レジスタ 13 を使用して、2 つの独立したプロセス ID レジスタにアクセスすることができます。

- *高速コンテキスト切り換え拡張機能プロセス ID レジスタ*
- *トレースプロセス ID レジスタ*、3-8 ページ

高速コンテキスト切り換え拡張機能プロセス ID レジスタ

opcode_2=0 で CP15 レジスタ 13 からリードを行うと、*高速コンテキスト切り換え拡張機能*（FCSE）*プロセスID*（PID）の値が返されます。FCSCE PIDレジスタの形式を図 3-10に示します。

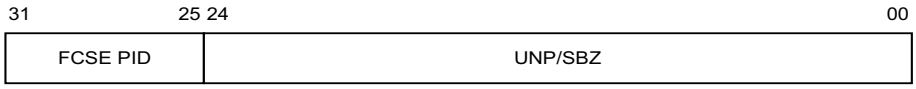


図 3-10 FCSCE PID レジスタの形式

注意： ビット [31:25] のみを返します。残りの 25 ビットは予測不能です。

opcode_2=0 で CP15 レジスタ 13 にライトを行うと、FCSE PID をビット [31:25] の値から更新します。ビット [24:0] はゼロでなければなりません。FCSE PID は、リセット時 b0000000 にセットされます。

FCSE PIDをリードしたり、ライトする場合、CRmおよびopcode_2はゼロでなければなりません。

FCSE PID の変更

以降の命令は従来の FCSE PID でフェッチされていますので、FCSE PID を変更する場合は注意してください。このように、FCSE PID の変更は遅延実行のある分岐と似ています。詳細については、*「FCSE PID による下位仮想アドレスの再配置」*の項（2-16 ページ）を参照してください。

トレースプロセス ID レジスタ

ETM7 へトレース*プロセス ID*（PROCID）を最大 32 ビットの長さまで保持するために、32 ビットのリード／ライトレジスタを設けています。これは、opcode_2 を 1 に設定して PROCID レジスタからリードやライトを行うことによって実行されます。PROCID レジスタの形式を図 3-11 に示します。

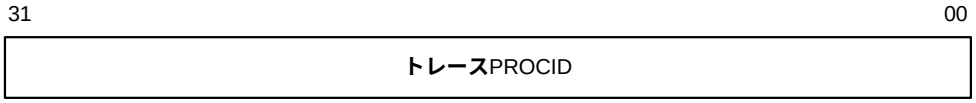


図 3-11 PROCID レジスタの形式

PROCIDWR 信号は、トレース PROCID がライトされたことを ETM7 に通知するために出力されます。

3.3.10 レジスタ 14（予約）

このレジスタへのアクセスは未定義です。レジスタ 14 へのライトも未定義です。

3.3.11 テストレジスタ

CP15 レジスタ 15 はデバイス固有のテスト動作に使用されます。詳細については、第 11 章「*テストサポート*」を参照してください。

このページは白紙です。

4

命令・
データキャッシュ

4　命令・データキャッシュ

この章では、命令・データキャッシュについて説明します。ここでは、以下の項を記載しています。

4.1　命令・データキャッシュについて

キャッシュは、リードミスアロケーションポリシーとランダム置換アルゴリズムによってライトスルーベースでのみ動作します。

4.1.1　IDC 動作

ARM720T プロセッサは、8KB の*命令・データ混在キャッシュ（IDC）を搭載しています。*

このキャッシュは、それぞれが 64 ラインから成る 4 つのセグメントから構成されており、各ラインは 8 ワードを格納することができます。IDC は、常に一度に 1 ラインずつリロードされます。IDC は、ARM720T 制御レジスタを使用してイネーブル／ディセーブルされます。また、**HRESETn** でディセーブルされます。

注意：　命令・データキャッシュがオンになっていると、MMU をディセーブルにすることはできません。ただし、制御レジスタに一回ライトを行うだけで 2 つのデバイスを同時にイネーブルにすることができます（*「制御レジスタ」*の項（3-4 ページ）を参照してください）。

4.1.2　キャッシャブルビット

C ビットは、リード中のデータを IDC に格納して以降のリード動作に使用するかどうかを決定します。通常、メインメモリはシステムパフォーマンスを向上するためにキャッシャブルとしてマークされ、I/O 空間は ARM720T キャッシュへのデータ保存をしないようにノンキャッシャブルとしてマークされます。

たとえば、プロセッサが I/O 空間のハードウェアフラグをポーリングする場合、キャッシュに保持された初期データのコピーではなく、外部周辺機器から強制的にデータをリードすることが重要です。キャッシャブルビットはページとセクションの両方で設定することができます。

キャッシャブルリード（C=1）

メモリのキャッシャブル領域でキャッシュミスが発生すると、8 ワードごとのラインフェッチが行われ、ランダムにキャッシュバンクに配置されます。

注意：　キャッシュラインフェッチではメモリアボートはサポートされていないため、無視されます。

アンキャッシャブルリード（C=0）

外部メモリアクセスが実行され、キャッシュへのライトは行われません。

4.1.3 リードロックライト

IDC は、以下のようにリードロックライト命令を特殊なケースとして扱います。

リードフェーズ キャッシュにデータが格納されているかどうかにかかわらず、常に強制的に外部メモリからリードします。

ライトフェーズ 通常のライト動作として扱われます。データが既にキャッシュに格納されている場合、キャッシュは更新されます。

2 つのフェーズは、**HLOCK** 信号をアサートすることによって分割できないものとして外部でフラグが立てられます。

4.2 IDC 有効性

IDC は仮想アドレスで動作しますので、その内容が、MMU で実行される物理マッピングに対する仮想アドレスとの整合性を保つようにしておいてください。メモリマッピングが変更された場合、IDC の有効性を確保する必要があります。

4.2.1 ソフトウェア IDC フラッシュ

IDC 全体を、キャッシュ動作レジスタ c7 にライトすることによって無効にすることができます。キャッシュはレジスタにライトが行われるとすぐにフラッシュされますが、次に続く 2 命令のフェッチがレジスタへのライト前にキャッシュから行われます。

4.2.2 二重にマップされた空間

キャッシュは仮想アドレスで動作しますので、各仮想アドレスは個別の物理アドレスにマップされるものとみなされます。同じ物理位置を 2 つ以上の仮想アドレスでアクセスすると、キャッシュは整合性を維持することができません。各仮想アドレスはキャッシュの中で個別のエントリをもちますので、1 回のプロセッサライト動作で更新されるのは 1 つのエントリだけです。

キャッシュの整合性を失わないようにするため、二重にマップされた仮想アドレスは両方ともアンキャッシャブルにする必要があります。

4.3 IDC イネーブル／ディセーブルおよびリセット

IDC は、**HRESETn** で自動的にディセーブルされ、フラッシュされます。イネーブルになると、キャッシャブルリードアクセスによってラインがキャッシュ内に配置されます。

IDC をイネーブルするには、

1. 制御レジスタのビット 0 をセットして、まず MMU がイネーブルになっていることを確認します。
2. 制御レジスタのビット 2 をセットして、IDC をイネーブルにします。制御レジスタへのシングルライトにより、MMU と IDC を同時にイネーブルにすることができます。

IDC をディセーブルにするには、

1. 制御レジスタのビット 2 をクリアします。
2. キャッシュ動作レジスタにライトを行ってフラッシュを行います。

5
ライトバッファ

5 ライトバッファ

この章ではライトバッファを説明します。ここでは、以下の項を記載しています。

5.1 ライトバッファについて

ARM720T プロセッサのライトバッファは、システムパフォーマンスを向上させるために用意されています。このライトバッファは、以下の内容をバッファリングすることができます。

- 最大 8 ワードのデータ
- 最大 8 個の独立アドレス

制御レジスタの W ビット、すなわちビット 3 を使用して、ライトバッファをイネーブル、ディセーブルにすることができます。ライトバッファは、リセット時にディセーブルされフラッシュされます。

ライトバッファの動作は、MMU ページテーブルに格納された*バッファ可能*（B）ビットでさらに制御することができます。このため、MMU はライトバッファを使用する前にイネーブルにしなければなりません。ただし、制御レジスタへのシングルライトによりこの 2 つの機能を同時にイネーブルにすることができます。

ライトバッファを使用するためのライトを行うには、制御レジスタの W ビットおよび対応するページテーブルの B ビットをセットしてください。

注意： バッファに格納されたライトデータを外部からアボートすることはできません。**HRESP[1:0]** に対するエラー応答は無視されます。アボートを生成するメモリ領域は、MMU ページテーブル内でバッファ不可能としてマークしなければなりません。

5.1.1 バッファ可能ビット

このビットは、ライト動作がライトバッファを使用するかしないかを制御します。通常、メインメモリはバッファ可能であり、I/O 空間はバッファ不可能です。B ビットはページとセクションの両方で設定することができます。

5.2 ライトバッファ動作

ライトバッファの動作は、CP15 レジスタ 1、すなわち制御レジスタによって制御することができます（*「制御レジスタ」*の項（3-4 ページ）を参照してください）。

CPU がライト動作を実行する場合、そのアドレスの変換エントリが検査され、B ビットの状態によって以後の動作が決定されます。制御レジスタを使用してライトバッファをディセーブルにすると、バッファリングされたライトはバッファリングされていないライトと同様に扱われます。

ライトバッファをイネーブルにするには、次の手順を取ります。

1. 制御レジスタのビット 0 をセットして、MMU がイネーブルになっていることを確認します。
2. 制御レジスタのビット 3 をセットして、ライトバッファをイネーブルにします。

   制御レジスタにシングルライトを行うことにより、MMU とライトバッファを同時にイネーブルにすることができます。

ライトバッファをディセーブルにするには、制御レジスタのビット 3 をクリアします。ライトバッファにある既存のライトは正常に完了しています。ライトバッファは、データが存在する限りライト動作を試行します。

5.2.1 バッファ可能ライト

ライトバッファがイネーブルになっている時にプロセッサがバッファ可能領域へのライトを行うと、データは **HCLK** 速度でライトバッファに格納され、CPU は実行を継続します。その後、ライトバッファは並行して外部ライトを実行します。

ライトバッファがフル状態になっていると、プロセッサはラインバッファに空のラインが確保されるまで停止します。

5.2.2 バッファ不可能ライト

ライトバッファがディセーブルになったり、CPU がバッファ不可能領域にライトを行ったりすると、プロセッサはライトバッファが空になり、外部からのライトが完了するまで機能停止します。この場合、同期化と複数の外部クロックサイクルが要求される場合があります。

5.2.3 リードロックライト

リードロックライトシーケンス（SWP 命令）のライトフェーズは、バッファ可能であってもバッファ不可能ライトとして扱われます。

5.2.4 ノンキャッシャブル領域からの読み出し

CPU がノンキャッシャブル領域からリードを行うと、ライトバッファに格納されているライトをすべて実行し、その間プロセッサは停止します。

5.2.5 ライトバッファのドレーン

ノンキャッシャブル位置からリードを行うことによって、強制的にライトバッファに格納されているライトをすべて実行します。

5.2.6 マルチワードライト

すべてのアクセスは不連続なものとして扱われます。これは、ライトでワードごとにアドレススロットとデータスロットが要求されることを意味しています。このため、バッファに格納された STM アクセスはバッファリングされていない STM アクセスよりも効率が悪い場合があります。大きなデータブロックを移す前には、（CP15 レジスタ 1 のビット 3 をクリアして）ライトバッファをディセーブルすることをお勧めします。

6
バスインタフェース

6 バスインタフェース

この章では、ARM720T プロセッサのバスインタフェースの信号について説明します。ここでは、以下の項を記載しています。

6.1 バスインタフェースについて

ARM720T プロセッサは、*アドバンスドハイパフォーマンスバス*（AHB）バスマスタです。レビジョンの違いを含め他の ARM プロセッサで設計を再使用できるようにするためには、設計サイクルの初めから完全に AMBA 準拠のペリフェラルとインタフェースを使用することを強くお勧めします。この章で解説する AHB タイミングはあくまでも例なので、すべての可能なアクセスを完全に記載している訳ではありません。

AMBA インタフェースとインテグレーションの詳細については、*「AMBA Specification」*を参照してください。

6.1.1 AHB 転送メカニズムの概要

AHB 転送は、以下のように構成されます。

アドレスフェーズ これは 1 サイクルしか継続されません。アドレスフェーズにウェイトを挿入することはできませんので、すべてのスレーブはアドレスフェーズ時にアドレスをサンプルしなければなりません。

データフェーズ このフェーズは、**HREADY** 信号を使用してウェイトを挿入することができます。**HREADY** 信号が Low の場合、転送にウェイトステートが挿入され、スレーブがデータを提供したりサンプルしたりできるように余分な時間が確保されます。

ライトデータバスは、マスタからスレーブへデータを移動するために使用されます。

リードデータバスは、スレーブからマスタへデータを移動するために使用されます。

図 6-1 は、ウェイトステートのない転送を示します（これは最もシンプルな転送タイプです）。

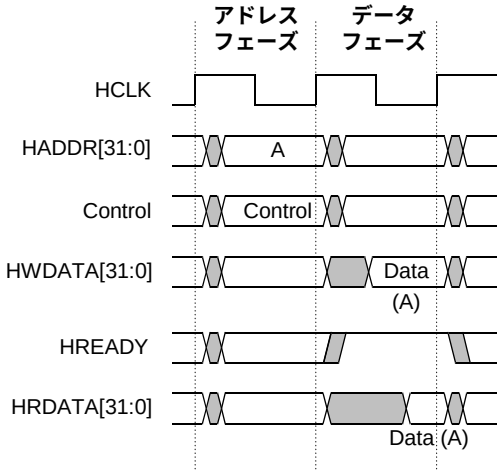


図 6-1 シンプルな AHB 転送

許可を受けたバスマスタは、アドレス信号と制御信号を駆動して AHB 転送を開始します。これらの信号は、転送に関する以下の情報を提供します。

- アドレス
- 方向
- 転送幅
- 転送がバーストの一部を形成するかどうか
- バーストタイプ

バーストとは連続した転送のことです。ARM720T プロセッサは、以下のタイプのバーストを行うことができます。

- 長さが指定されていないインクリメントバースト
- ラインフィル時にのみ使用される 8 ビートインクリメントバースト

  インクリメントバーストはアドレス境界でラップアラウンドされません。バーストでの各転送のアドレスは、バーストでの前回の転送アドレスがインクリメントされたものです。

詳細については、*「アドレスと制御信号」*の項（6-7 ページ）を参照してください。

AHB 転送メカニズムの詳細については、*「AMBA Specification（Rev 2.0）」*を参照してください。

6.2 バスインタフェース信号

ARM720T プロセッサバスインタフェースの信号は、以下のカテゴリに分類することができます。

転送タイプ **HTRANS[1:0]**

*「転送タイプ」*の項（6-5 ページ）を参照してください。

アドレスと制御

HADDR[31:0]
HWRITE
HSIZE[2:0]
HBURST[2:0]
HPROT[3:0]

*「アドレスと制御信号」*の項（6-7 ページ）を参照してください。

スレーブ転送応答

HREADY
HRESP[1:0]

*「スレーブ転送応答信号」*の項（6-9 ページ）を参照してください。

データ

HRDATA[31:0]
HWDATA[31:0]

*「データバス」*の項（6-10 ページ）を参照してください。

アービトレーション

HBUSREQ
HGRANT
HLOCK

*「アービトレーション」*の項（6-12 ページ）を参照してください。

クロック

HCLK
HCLKEN

*「バスクロッキング」*の項（6-13 ページ）を参照してください。

リセット

HRESETn

*「リセット」*の項（6-13 ページ）を参照してください。

これらの各信号グループは、バスインタフェースサイクルについて共通のタイミング関係を共有しています。ARM720T プロセッサバスインタフェースの信号はすべて **HCLK** の立ち上がりエッジで生成され、サンプリングされます。

AHB バスマスタインタフェース信号を図 6-2 に示します。

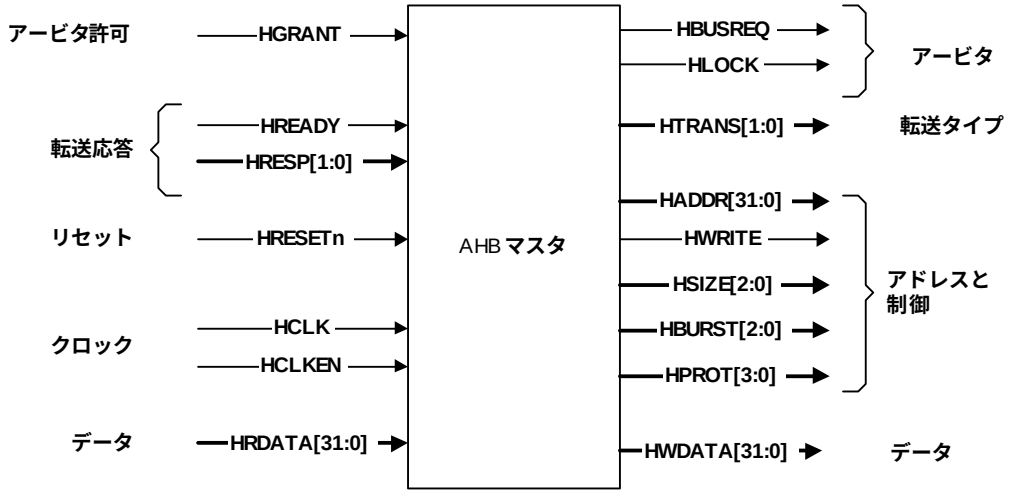


図 6-2 AHB **バスマスタインタフェース**

6.3 転送タイプ

ARM720T プロセッサバスインタフェースはパイプライン処理されますので、アドレス信号とメモリリクエスト信号はそれらが参照（使用）されるバスサイクルより前のバスサイクルで処理されます。これによって、メモリサイクルは、アドレスをデコードし、アクセス要求に応答するための時間を最大限得ることができます。

シングルメモリサイクルを図 3-1 に示します。

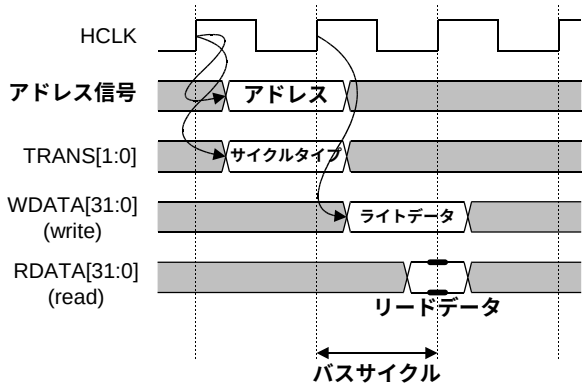


図 6-3 単純なメモリサイクル

転送には 3 つのタイプがあります。転送タイプは、表 6-1 に示す **HTRANS[1:0]** 信号によって示されます。

表 6-1 転送タイプエンコーディング

HTRANS[1:0]	転送タイプ	説明
b00	IDLE	データ転送が必要でないことを示します。バスマスタにバスの許可が出たが、データ転送を実行したくないときには、IDLE 転送タイプが使用されます。 スレーブは常にIDLE転送に対してウェイトステートなしの OKAY 応答を返し、転送はスレーブで無視されます。
b10	NONSEQ	バーストの最初の転送またはシングル転送を示します。アドレス信号と制御信号は、前回の転送とは無関係です。 バス上の複数のシングル転送は、1 つの転送から成るバーストとして扱われます。
b11	SEQ	バーストでは、最初の転送以外の転送はすべて SEQUENTIAL です。 アドレスは前回の転送と関係があり、前回の転送のアドレスにそのサイズ（バイト単位）を加えたものと同じです。ラップ式バーストの場合、転送のアドレスは転送のビート数（4、8、または 16 のいずれか）で乗算されたサイズ（バイト単位）に等しいアドレス境界でラップされます。制御情報については、前回の転送と同じです。

注意： *AMBA Specification（Rev 2.0）*では、**HTRANS[1:0]**=b01 は BUSY サイクルを示しますが、ARM720T プロセッサがこれらの信号を挿入することはありません。

図 6-4 は、異なる転送タイプの例を示しています。

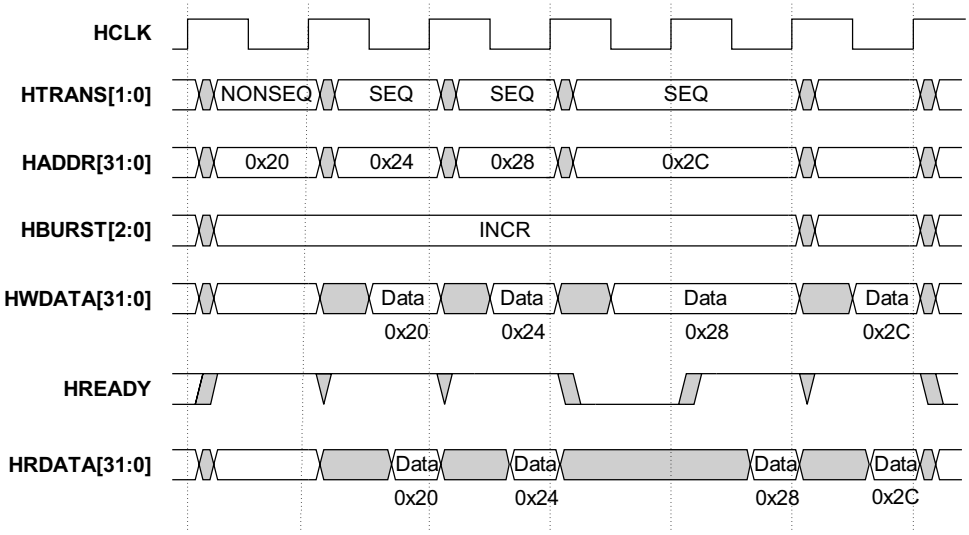


図 6-4 転送タイプの例

図 6-4 では、

- 最初の転送はバーストの始まりなので、連続していません。
- マスタは、すぐにバーストの 2 番目の転送を実行します。
- マスタはすぐにバーストの 3 番目の転送も実行しますが、このときスレーブは完了していないため、**HREADY** を使って 1 ウェイトステートを挿入します。
- バーストの最終転送は、ウェイトステートなしで完了します。

6.4 アドレスと制御信号

アドレス信号と制御信号は、以下の項で説明します。

- *HADDR[31:0]*
- *HWRITE*
- *HSIZE[2:0]*
- *HBURST[2:0]*、6-8 ページ
- *HPROT[3:0]*、6-8 ページ

6.4.1 HADDR[31:0]

HADDR[31:0] は、転送のアドレスを指定する 32 ビットアドレスバスです。すべてのアドレスはバイトアドレスなので、ワードアドレスのバーストは各サイクルで 4 ずつインクリメントするアドレスバスになります。

アドレスバスは、4GB のリニアアドレス指定空間を提供します。これは以下のことを意味しています。

- ワードアクセスの信号を受けると、メモリシステムは下位 2 ビット、**HADDR[1:0]** を無視します。
- ハーフワードアクセスの信号を受けると、メモリシステムは下位 1 ビット、**HADDR[0]** を無視します。

6.4.2 HWRITE

HWRITE は、以下のように転送方向を指定します。

HWRITE HIGH ARM720T プロセッサのライトサイクルを示します。

HWRITE LOW ARM720T プロセッサのリードサイクルを示します。

S サイクルのバーストは、必ずリードバーストとライトバーストのいずれかです。転送方向は、バーストの途中で変更することはできません。

6.4.3 HSIZE[2:0]

SIZE[2:0] バスは、転送のサイズをエンコードします。ARM720T プロセッサはワード、ハーフワード、バイトを転送することができます。これは、表 6-2 に示すように **SIZE[2:0]** でエンコードされます。

注意： C コンパイラと ARM デバッグツールチェーンを使用するためには、システムが任意のバイトやハーフワードのライトをサポートしていなければなりません。このため、各個別バイトのレベルまでライトイネーブルを確保して、バスサイズまでのすべての転送サイズをサポートするようにしなければなりません。

表 6-2 転送サイズエンコーディング

HSIZE[2:0]	サイズ	転送幅
b000	8 bits	**バイト**
b001	16 bits	**ハーフワード**
b010	32 bits	**ワード**

6.4.4 HBURST[2:0]

HBURST[2:0] は、表 6-3 に示すように ARM720T コアによって生成されるバーストのタイプを示します。

表 6-3 バーストタイプエンコーディング

HBURST[2:0]	タイプ	説明
b000	SINGLE	シングル転送
b001	INCR	長さが指定されていないインクリメントバースト
b101	INCR8	8ビートインクリメントバースト

バースト動作の詳細については、*「AMBA Specification （Rev 2.0）」*を参照してください。

6.4.5 HPROT[3:0]

HPROT[3:0] は保護制御バスです。これらの信号はバスアクセスの追加情報を与えるものであり、主としてモジュールにアクセス許可機構を実現させることを目的としています。

これらの信号は、転送が以下のどちらであるかを示します。

- オペコードフェッチまたはデータアクセス
- 特権モードアクセスまたはユーザモードアクセス

メモリ管理ユニット付きのバスマスタの場合、これらの信号は現在のアクセスがキャッシュ可能か、あるいはバッファ可能かどうかも示します。

表 6-4 は、ARM720T コアから生成される保護制御エンコーディングを示しています。

表 6-4 保護制御エンコーディング

HPROT[3] キャッシャブル	HPROT[2] バッファ可能	HPROT[1] 特権モード	HPROT[0] データ／オペコード	説明
-	-	-	0	オペコードフェッチ
-	-	-	1	データアクセス
-	-	0	-	ユーザアクセス
-	-	1	-	特権アクセス
-	0	-	-	バッファ不可能
-	1	-	-	バッファ可能
0	-	-	-	キャッシュ不可能
1	-	-	-	キャッシュ可能

一部のバスマスタは正確な保護情報を生成することができないため、どうしても必要でない限り、スレーブは **HPROT[3:0]** 信号を使用しないことをお勧めします。

6.5 スレーブ転送応答信号

マスタが転送を開始すると、スレーブは転送の進行具合を確認します。転送が開始されてからバスマスタが転送を取り消す手段については、AHB 仕様では規定されていません。

スレーブがアクセスされたときは必ず、以下の信号を使用して応答をしなければなりません。

HRESP[1:0] 転送の状態を示します。

HREADY 転送を延長するために使用されます。この信号は、**HRESP[1:0]** と組み合わされて機能します。

スレーブは、以下のような方法で転送を完了することができます。

- すぐに転送を完了する。
- 1 つ以上のウェイトステートを挿入して、転送を完了する時間を確保する。
- エラー信号を出して、転送が失敗したことを示す。
- 転送の完了を遅らせる。ただし、マスタとスレーブがバスを戻し、他の転送でバスを利用できるようにする。

6.5.1 HREADY

HREADY 信号は、以下のように AHB 転送のデータフェーズにウェイトを挿入するために使用されます。

HREADY LOW 転送データフェーズにウェイトが挿入されることを示しています。転送にウェイトステートを挿入して、スレーブがデータを提供したり、サンプリングしたりするための時間を確保できるようにします。

HREADY HIGH 転送が完了できることを示しています。

各スレーブは、バスアクセスのレイテンシを計算できるようにするために、バスを戻す前に挿入する、あらかじめ決められた最大数のウェイトステートを持っていなければなりません。一回のアクセスで多数のクロックサイクルによりバスがロックされないようにするために、スレーブが 16 を超えるウェイトステートを挿入しないようにすることをお勧めします。

6.5.2 HRESP[1:0]

HRESP[1:0] は、スレーブが転送状態を示すために使用します。**HRESP[1:0]** エンコーディングを表 6-5 に示します。

表 6-5 応答エンコーディング

HRESP[1:0]	応答	説明
b00	OKAY	**HREADY が** High **のとき、この応答は転送が正常に完了したことを示します。** OKAY **応答は、他の** 3 **つの応答のうちのどれかを出す前に、HREADY が** Low **となり挿入される追加サイクルに対しても使用されます。**
b01	ERROR	**この応答は、転送エラーが発生し、転送が正常に完了しなかったことを示します。通常、これはリードオンリメモリ位置へのライトをしようとしたなどの保護エラーに対して使用されます。エラー状態は、転送が正常に完了しなかったことを認識させるために、バスマスタへ信号を送らなければいけません。** **エラー状態には、**2 **サイクル応答が必要です。**
b10	RETRY	RETRY **応答は転送がまだ完了していないことを示します。このため、バスマスタは転送を再試行しなければなりません。バスマスタは、転送が完了するまで転送を再試行し続けます。** 2 **サイクルの** RETRY **応答が必要です。**
b11	SPLIT	**転送がまだ完了していません。バスマスタは、次にバスをアクセスする許可を受けたときに転送を再試行しなければなりません。スレーブは、転送が完了するときにマスタのためにバスへのアクセスを要求します。** 2 **サイクルの** SPLIT **応答が必要です。**

スレーブ転送応答の詳細については、*「AMBA Specification（Rev 2.0）」*を参照してください。

6.6 データバス

トライステートドライバを使用せずに AHB システムを実現できるようにするには、独立した 32 ビットのリードデータバスとライトデータバスが必要になります。

6.6.1 HWDATA[31:0]

ライトデータバスは、ライト転送中にバスマスタによって駆動されます。転送が延長されると、バスマスタは **HREADY** が High になり転送が完了するまでデータを有効にしておかなければなりません。

すべての転送は、転送サイズに等しいアドレス境界に合わせなければなりません。たとえば、ワード転送はワードアドレス境界（すなわち、A[1:0]=b00）に合わせなければなりません。また、ハーフワード転送はハーフワードアドレス境界（すなわち、A[0]=0）に合わせなければなりません。

バスマスタは、転送サイズにかかわりなくすべてのバイトレーンを駆動します。

- ハーフワード転送では、たとえば 0x1234 の場合は、**HWDATA[31:0]** はエンディアンにかかわりなく値 0x12341234 で駆動されます。
- バイト転送では、たとえば 0x12 の場合、**HWDATA[31:0]** はエンディアンにかかわりなく値 0x12121212 で駆動されます。

6.6.2 HRDATA[31:0]

リードデータバスは、リード転送時に適切なスレーブによって駆動されます。スレーブが **HREADY** を Low に保持してリード転送を延長した場合、スレーブは **HREADY** が High を示す転送の最終サイクルの終了時点でのみ有効なデータを提供する必要があります。

バス幅よりも狭い転送については、スレーブはアクティブなバイトレーンでのみ有効なデータを提供する必要があります。バスマスタは、適切なバイトレーンからデータを選択する役割を担っています。以下の表は、アクティブなバイトレーンを明らかにしています。

- 表 6-6（6-11 ページ）は、リトルエンディアンタイプの場合のアクティブバイトレーンを示しています。
- 表 6-7（6-12 ページ）は、ビッグエンディアンタイプの場合のアクティブバイトレーンを示しています。

スレーブは、**HRESP[1:0]** による OKAY 応答で転送が完了したときにのみ有効なデータを提供する必要があります。SPLIT、RETRY、ERROR 応答は有効なリードデータを必要としません。

6.6.3 エンディアン

モジュールはすべて同じエンディアンタイプを使うものであり、かつデータ経路やブリッジも同じエンディアンタイプを使うものであることが不可欠です。

ダイナミックなエンディアンタイプはサポートしていません。ダイナミックなエンディアンタイプをサポートするとほとんどの組み込みシステムで重大なシリコンオーバーヘッドを引き起こし、冗長になるからです。

幅広いアプリケーションで使用するモジュールのみをバイ－エンディアンにしておいて、設定端子や内部制御ビットによってエンディアンを選択することを推奨します。より特定用途向けのブロックについては、エンディアンをリトルエンディアンかビッグエンディアンのいずれかに固定することにより、より小さく電力消費が少ない、またより高いパフォーマンスのインタフェースが得られます。

表 6-6 は、リトルエンディアンタイプの場合のアクティブなバイトレーンを示します。

表 6-6 32 ビットリトルエンディアンデータバスのアクティブバイトレーン

転送サイズ	アドレスオフセット	データ [31:24]	データ [23:16]	データ [15:8]	データ [7:0]
ワード	0	✓	✓	✓	✓
ハーフワード	0	-	-	✓	✓
ハーフワード	2	✓	✓	-	-
バイト	0	-	-	-	✓
バイト	1	-	-	✓	-
バイト	2	-	✓	-	-
バイト	3	✓	-	-	-

表 6-7 は、ビッグエンディアンタイプのアクティブなバイトレーンを示します。

表 6-7 32 ビットビッグエンディアンデータバスのアクティブバイトレーン

転送サイズ	アドレス オフセット	データ [31:24]	データ [23:16]	データ [15:8]	データ [7:0]
ワード	0	✓	✓	✓	✓
ハーフワード	0	✓	✓	-	-
ハーフワード	2	-	-	✓	✓
バイト	0	✓	-	-	-
バイト	1	-	✓	-	-
バイト	2	-	-	✓	-
バイト	3	-	-	-	✓

6.7 アービトレーション

アービトレーションメカニズムは、*「AMBA Specification（Rev 2.0）」*で詳細に説明しています。このメカニズムは、常に一度に 1 つのマスタだけがバスにアクセスできるようにするために使用されます。アービタは、バスを使用する多数の異なる要求を監視し、バスを要求しているどのマスタが最も優先順位の高いマスタであるかを決定することによってこの機能を実行しています。またアービタは、SPLIT 転送を完了したいスレーブからの要求も受け付けます。

SPLIT 転送を実行することができないスレーブはアービトレーション処理を意識する必要はありません。ただし、バスの所有権が変更されるとバーストの転送が完了しない恐れがあるということに注意する必要があります。

6.7.1 HBUSREQ

バス要求信号は、バスへのアクセスを要求するバスマスタにより使用されます。各バスマスタはアービタに対する独自の **HBUSREQ** 信号を持っており、システムには最大 16 の独立したバスマスタが存在可能です。

6.7.2 HLOCK

ロック信号は、バス要求信号と同時にマスタによってアサートされます。これは、マスタが多くの分割できない転送を実行しており、ロックされた転送の最初の転送がいったん開始されたら、アービタは他のバスマスタにバスへのアクセスを許可してはならないということをアービタに示しています。**HLOCK** は、アービタが参照するアドレスよりも少なくとも 1 サイクル前にアサートされ、アービタが許可信号を変更できないようにしなければなりません。

6.7.3 HGRANT

許可信号はアービタによって生成されます。この信号はロックされた転送と SPLIT 転送を考慮して、該当するマスタが現在バスを要求している最も優先順位の高いマスタであることを示します。

HGRANT が High で、**HREADY** が **HCLK** の立ち上がりエッジで High の場合、マスタはアドレスバスの所有権を得ます。

6.8 バスクロッキング

ARM720T プロセッサバスインタフェースには、2 つのクロック入力があります。

6.8.1 HCLK

バスは、システムクロック、すなわち **HCLK** によってクロック制御されます。このクロックはすべてのバス転送のタイミングをとります。すべての信号タイミングは、**HCLK** の立ち上がりエッジと関係があります。

6.8.2 HCLKEN

HCLK は **HCLKEN** 信号によってイネーブルになります。**HCLKEN** を使用して、バスインタフェースの **HCLK** を分周することによってバス転送レートを遅くすることができます。

注意： **HCLKEN** は CPU 自体に対してではなく、バスに対してのみクロックイネーブルになっています。バスにウェイトステートを挿入するには、**HREADY** 信号を使用します。

6.9 リセット

バスリセット信号は **HRESETn** です。この信号は、システムとバスのリセットに使用されるグローバルリセットです。この信号は非同期にアサートすることができますが、ディアサートは **HCLK** の立ち上がりエッジの後に同期をとって行われます。完全なシステムリセットは、**DBGnTRST** が **HRESETn** と同じようにアサートされたときに行われます。

リセット時、すべてのマスタは以下の点を確保しなければなりません。

- アドレス信号と制御信号が有効なレベルにある。
- **HTRANS[1:0]** が IDLE を示している。

HRESETn は、AMBA AHB 仕様では唯一のアクティブ Low 信号です。

このページは白紙です。

7
メモリ管理ユニット

7 メモリ管理ユニット

この章では、*メモリ管理ユニット*（MMU）を解説します。ここでは、以下の項を記載しています。

7.1 MMU について

ARM720T プロセッサは拡張された ARM アーキテクチャ v4 MMU をインプリメントしており、コアの命令およびデータアドレスポートの変換およびアクセス許可チェックを提供できます。MMU は、CP15 レジスタ 1 の M ビットでイネーブルにされる、メインメモリに格納された 2 段階ページテーブルによって制御されます。これによって、シングルアドレス変換および保護機構が実現されます。

MMU の機能は、以下のとおりです。

- 標準の ARMv4 MMU マッピングサイズ、ドメイン、およびアクセス保護機構
- マッピングサイズは、1MB（セクション）、64KB（ラージページ）、4KB（スモールページ）、および 1KB（タイニーページ）
- セクションのアクセス許可
- ラージページとスモールページのアクセス許可は、ページの 1/4 ごとに単独に指定可能（これらの 1/4 ページをサブページと呼ぶ）
- 最大 16 ドメインがハードウェアによりインプリメント可能
- 64 エントリ TLB
- ハードウェアページテーブルウォーク
- ランダムロビン置換アルゴリズム（サイクリックとも呼ばれる）
- CP15 レジスタ 8 を使用した TLB 全体の無効化
- CP15 レジスタ 8 を使用した、*修正仮想アドレス*（MVA）で選択された TLB エントリの無効化

7.1.1 アクセス許可とドメイン

ラージページやスモールページについては、アクセス許可はサブページ（スモールページの場合は 4KB、ラージページの場合は 16KB）ごとに定義されます。セクションとタイニーページは、1 セットでアクセス許可が与えられます。

すべてのメモリ領域には関連するドメインを持っています。ドメインとは、あるメモリ領域の主要なアクセス制御メカニズムのことであり、アクセスを進行させるために必要な条件を定義しています。ドメインは以下の項目を決定します。

- アクセス許可がアクセスに権限を与えるために使用されるかどうか。
- アクセスを無条件で続行を許可するかどうか。
- アクセスを無条件でアボートするかどうか。

後の方の 2 項目については、アクセス許可属性は無視されます。

ドメインは 16 あります。これらのドメインは、ドメインアクセス制御レジスタを使用して設定されます。

7.1.2 変換エントリ

TLB は 64 の変換エントリを一時保存します。CPU メモリアクセス時、TLB はアクセス制御ロジックに対して保護情報を提供します。

TLB が MVA 用の変換エントリを含む場合、アクセス制御ロジックは以下のようにアクセスが許可されているかどうかを確認します。

- アクセスが許可され、チップ外部へのアクセスが要求される場合、MMU は MVA に対応した適切な物理アドレスを出力します。
- アクセスが許可され、チップ外部へのアクセスが要求されない場合、キャッシュはアクセスを処理します。
- アクセスが許可されない場合、MMU は CPU コアにアボートするように信号を送ります。

TLB がミスした（VA のエントリを含んでいない）場合、変換テーブルウォークハードウェアが呼び出されて、物理メモリの変換テーブルから変換情報を検索します。その変換情報は、検索された場合 TLB に書き込まれ、既存の値を上書きします。

上書きされるエントリは、TLB 位置を巡回して選択されます。

リセット時などにMMUがオフになると、アドレスマッピングは行われず、すべての領域はキャッシュ不可能およびバッファ不可能としてマークされます。

7.2 MMU プログラムアクセス可能レジスタ

表 7-1 は、CP15 レジスタについてまとめています。これらは MMU の動作を決定するためにメモリに保存されるページテーブルディスクリプタと組み合わせて使用されます。

表 7-1 CP15 レジスタの機能

レジスタ	番号	ビット	レジスタの説明
制御レジスタ	1	M, A, S, R	MMU をイネーブルにするビット（M ビット）、データアドレスアラインメントチェックをイネーブルにするビット（A ビット）、およびアクセス保護機構を制御するビット（S ビット、R ビット）を格納しています。
変換テーブルベースレジスタ	2	31:14	メインメモリに保持されている変換テーブルのベースの物理アドレスを保持しています。このベースアドレスは、16KB 境界にある必要があります。
ドメインアクセス制御レジスタ	3	31:0	16 の 2 ビットフィールドから成ります。各フィールドは、16 のドメイン（D15 ～ D0）のうちの 1 つのアクセス制御属性を定義しています。
フォールトステータスレジスタ	5	7:0	アボートが発生した時のデータアボートやプリフェッチアボートの原因およびアボートされたアクセスのドメイン番号を示します。ビット 7：4 は、フォールトが発生したときに 16 個のドメイン（D15 ～ D0) のうちのどのドメインにアクセスしていたのかを指定します。ビット 3：0 は、試行していたアクセスのタイプを示します。他のすべてのビットの値は予測不能です。これらのビットのエンコーディングを表 7-9（7-16 ページ）に示します。
フォールトアドレスレジスタ	6	31:0	アボートを引き起こしたアクセスと関連のある MVA を保持しています。各フォールトタイプに関して保存されたアドレスの詳細については、表 7-9（7-16 ページ）を参照してください。 バンクレジスタ c14 を使用すると、プリフェッチアボートと関連のある VA を確認することができます。
TLB 動作レジスタ	8	31:0	このレジスタにライトを行って、MMU に TLB メンテナンス動作を行わせることができます。メンテナンス動作は次の通りです。 • TLB 内のすべてのエントリを無効化する。 • 特定のエントリを無効化する。

レジスタ c8 以外のすべての CP15 MMU レジスタは、状態を格納しています。これらのレジスタについては、MRC 命令を使用してリードしたり、MCR 命令を使用してライトすることができます。また、レジスタ c5、c6 は、すべてのアボート発生時に MMU によりライトされます。レジスタ c8 にライトを行うと、MMU は TLB 動作を実行して TLB エントリを操作します。このレジスタからリードすることはできません。

CP15 については、レジスタの形式およびレジスタをアクセスするために使用することができるコプロセッサ命令に関して、第 3 章*「コンフィギュレーション」*で詳細に説明しています。

7.3 アドレス変換

MMU は、CPU コアおよび CP15 レジスタ c13 によって生成される VA を外部メモリにアクセスするために物理アドレスに変換します。また、TLB を使用してアクセス許可を取得、確認します。

MMU テーブルウォーキングハードウェアは、TLB にエントリを追加するために使用されます。アドレス変換データとアクセス許可データの両方からなる変換情報は、物理メモリに配置された変換テーブルに常駐しています。MMU は、この変換テーブルを検索したり、TLB にエントリをロードしたりするためのロジックを提供します。

ハードウェアテーブルウォーキング処理やアクセス許可確認処理には、1 つまたは 2 つのステージがあります。ステージの数は、アドレスがセクションマップドアクセスまたはページマップドアクセスとしてマークされているかどうかによって異なります。

ページマップドアクセスには 3 つのサイズがあり、セクションマップドアクセスには 1 つのサイズがあります。ページマップドアクセスは、以下のものに用意されています。

- ラージページ
- スモールページ
- タイニーページ

変換プロセスは、レベル 1 フェッチで常に同じように開始されます。セクションマップドアクセスはレベル 1 フェッチのみが必要ですが、ページマップドアクセスは次のレベル 2 フェッチを要求します。

7.3.1 変換テーブルベースレジスタ

ハードウェア変換プロセスは、要求される MVA 用の変換を TLB が含まない場合に開始されます。*「変換テーブルベース」*レジスタは、セクションまたはページディスクリプタあるいはその両方を格納した物理メモリのテーブルのベースアドレスを指し示します。変換テーブルベースレジスタの下位 14 ビットは、リード時にはゼロにセットされます。このテーブルは 16KB 境界に常駐していなければなりません。図 7-1 は、変換テーブルベースレジスタの形式を示しています。

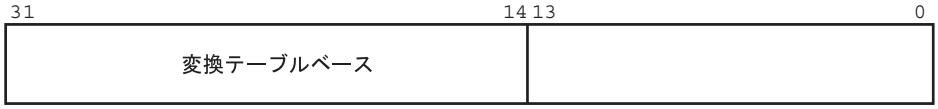
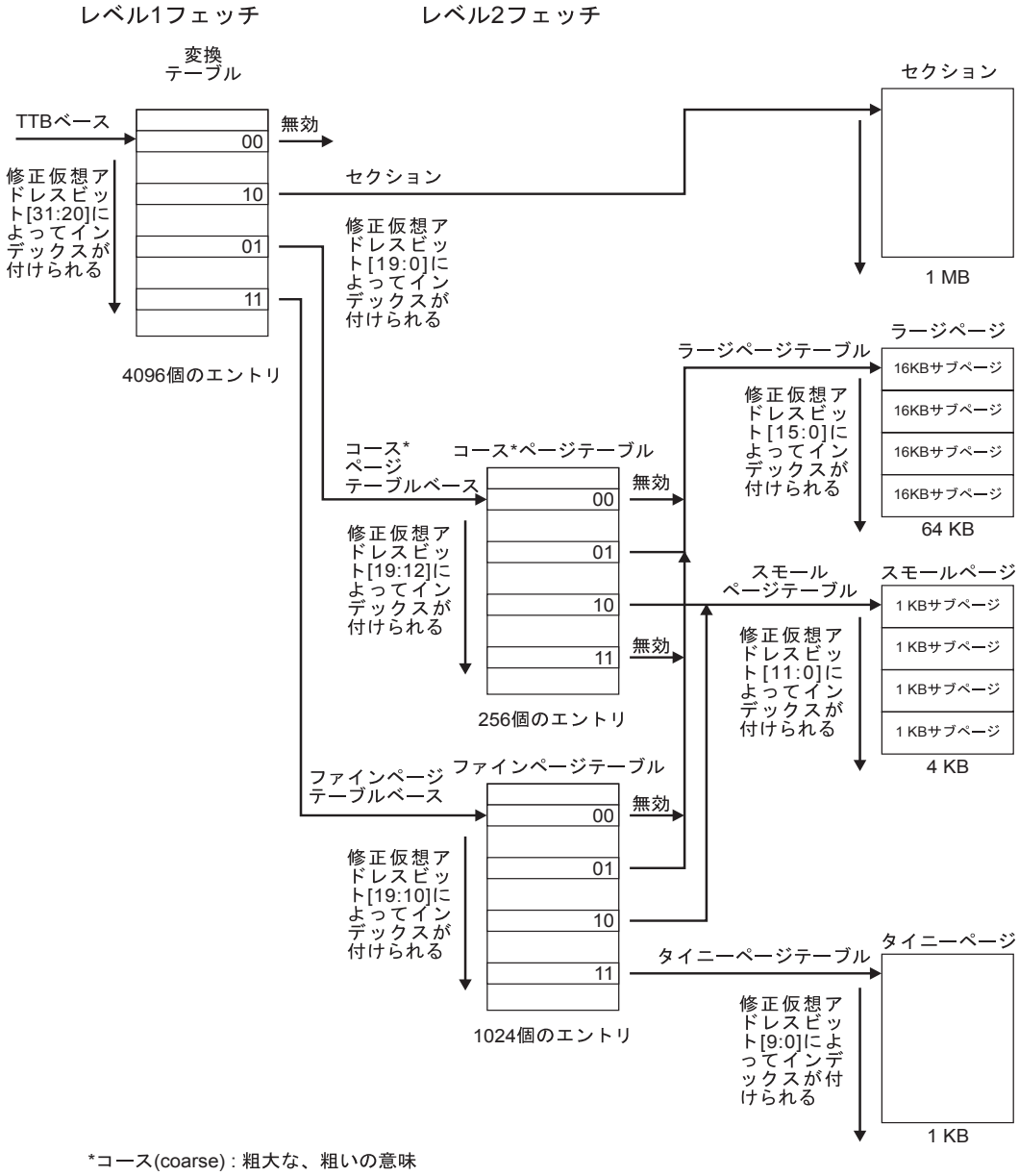


図 7-1 変換テーブルベースレジスタ

変換テーブルは最大 4096 x 32 ビットのエントリをもち、各エントリは 1MB の仮想メモリを記述しています。これによって、最大 4GB の仮想メモリをアドレス指定することができます。図 7-2 は、テーブルウォーク処理を示しています。



*コース(coarse) : 粗大な、粗いの意味

図 7-2 変換ページテーブル

7.3.2 レベル 1 フェッチ

図 7-3 に示すように、変換テーブルベースレジスタのビット [31:14] は MVA のビット [31:20] と連結され、30 ビットアドレスを生成します。

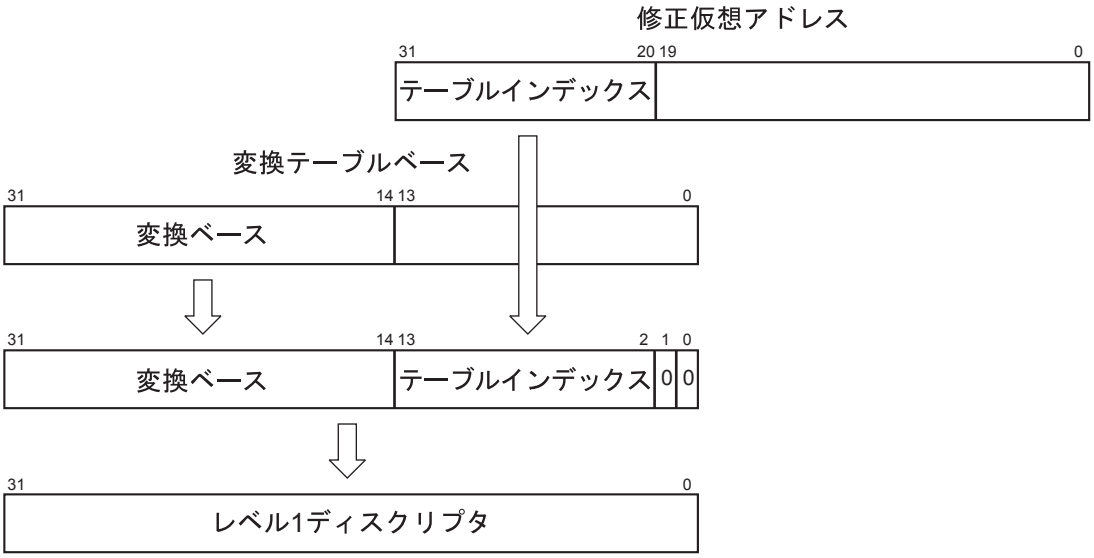


図 7-3 変換テーブルレベル 1 ディスクリプタのアクセス

このアドレスは、4 バイトの変換テーブルエントリを選択します。この変換テーブルエントリは、セクションまたはページテーブルのいずれかのためのレベル 1 ディスクリプタです。

7.3.3 レベル 1 ディスクリプタ

返されたレベル 1 ディスクリプタは、セクションディスクリプタ、コースページテーブルディスクリプタ、あるいはファインページテーブルディスクリプタのいずれかであるか、または無効です。図 7-4 は、レベル 1 ディスクリプタの形式を示しています。

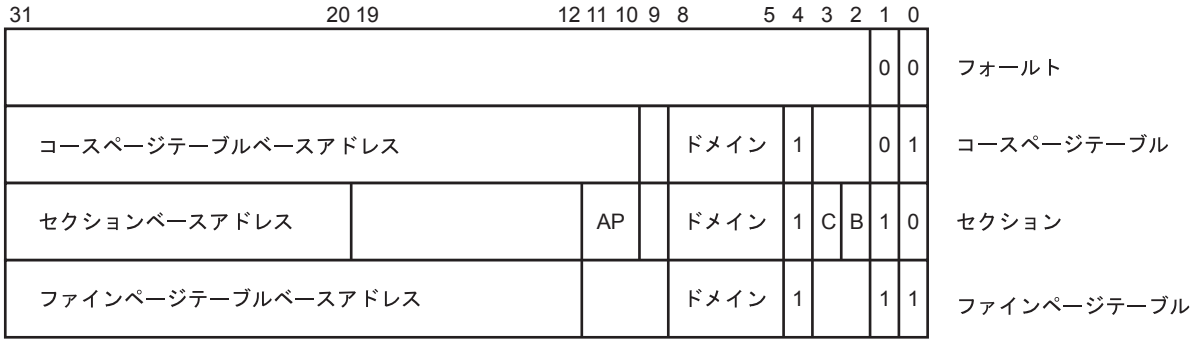


図 7-4 レベル 1 ディスクリプタ

セクションディスクリプタは、1MB メモリブロックのベースアドレスを提供します。

ページテーブルディスクリプタは、レベル 2 ディスクリプタを格納したページテーブルのベースアドレスを提供します。ページテーブルには、以下の 2 つのサイズがあります。

- コースページテーブルは 256 個のエントリをもち、テーブルが記述する 1MB を 4KB のブロックに分割しています。
- ファインページテーブルは 1024 個のエントリをもち、テーブルが記述する 1MB を 1KB のブロックに分割しています。

レベル 1 ディスクリプタのビット割り当てを表 7-2 に示します。

表 7-2 レベル 1 ディスクリプタのビット割当

ビット			説明
セクション	コース	ファイン	
31:20	31:10	31:12	これらのビットは、物理アドレスの対応ビットを形成します。
19:12	-	-	ゼロでなければなりません。
11:10	-	-	アクセス許可ビットです。*「ドメインアクセス制御」*の項（7-17 ページ）および *「フォールトチェックシーケンス」*の項（7-19 ページ）に、アクセス許可ビットの説明をしています。
9	9	11:9	ゼロでなければなりません。
8:5	8:5	8:5	ドメイン制御ビット
4	4	4	1 でなければなりません。
3:2	-	-	これらのビット C、B は、このページでマップされたメモリ領域がキャッシュ可能かキャッシュ不可能なものとして、およびバッファ可能かバッファ不可能ものとして扱われるかどうかを示しています（システムは常にライトスルーです）。
-	3:2	3:2	ゼロでなければなりません。
1:0	1:0	1:0	これらのビットはページサイズと有効性を示しており、表 7-3 に示すように解釈されます。

レベル 1 ディスクリプタの最下位 2 ビットは、表 7-3 に示すようにディスクリプタタイプを示しています。

表 7-3 レベル 1 ディスクリプタビット [1:0] の解釈

値	意味	説明
0 0	無効	セクション変換フォールトを生成します。
0 1	コースページテーブル	コースページテーブルディスクリプタであることを示します。
1 0	セクション	セクションディスクリプタであることを示します。
1 1	ファインページテーブル	ファインページテーブルディスクリプタであることを示します。

7.3.4 セクションディスクリプタ

セクションディスクリプタは、1MB メモリブロックのベースアドレスを示します。図 7-5 は、セクションディスクリプタの形式を示しています。

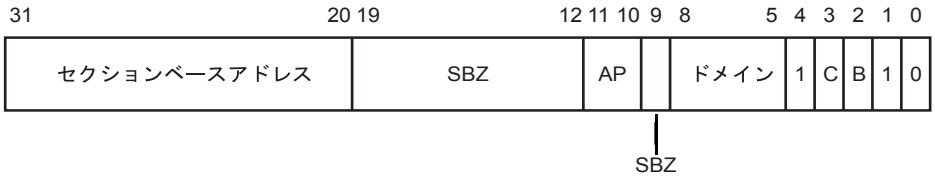


図 7-5 セクションディスクリプタ

セクションディスクリプタのビット割当を表 7-4 に説明します。

表 7-4 セクションディスクリプタのビット

ビット	説明
31:20	セクションの物理アドレスの対応ビットを形成します。
19:12	常に “0” がライトされます。
11:10	（AP）このセクションのアクセス許可を指定します。
9	常に “0” がライトされます。
8:5	主要なアクセス制御を含む 16 のドメイン（ドメインアクセス制御レジスタに保持されます）のうちの 1 つを指定します。
4	下位互換性を得るために “1” がライトされなければなりません。
3:2	これらのビット C、B は、このページでマップされるメモリ領域がキャッシュ可能かキャッシュ不可能なものとして、およびバッファ可能かバッファ不可能なものとして扱われるかどうかを示しています（システムは常にライトスルーです）。
1:0	これらのビットは、セクションディスクリプタを示すために b10 でなければなりません。

7.3.5 コースページテーブルディスクリプタ

コースページテーブルディスクリプタは、ラージページアクセスまたはスモールページアクセスのいずれかのレベル 2 ディスクリプタを含むページテーブルのベースアドレスを示します。コースページテーブルは 256 個のエントリをもち、テーブルが記述する 1MB を 4KB のブロックに分割しています。図 7-6 は、コースページテーブルディスクリプタの形式を示します。

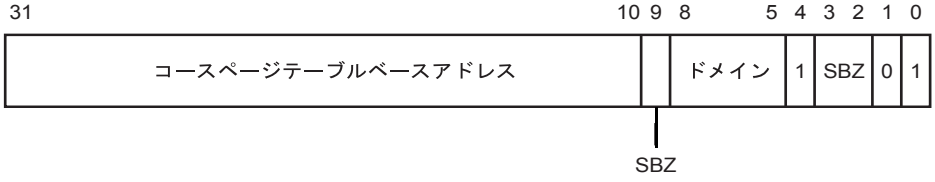


図 7-6 コースページテーブルディスクリプタ

注意： コースページテーブルディスクリプタがレベル 1 フェッチから戻されると、レベル 2 フェッチが開始されます。

コースページテーブルディスクリプタのビット割当を表 7-5 に示します。

表 7-5 コースページテーブルディスクリプタのビット割当

ビット	説明
31:10	これらのビットは、レベル 2 ディスクリプタを参照するためのベースを形成します（エントリのコースページテーブルインデックスは MVA から得られます）。
9	常に “0” がライトされます。
8:5	これらのビットは、主要なアクセス制御を含んでいる 16 のドメイン（ドメインアクセス制御レジスタに保持されます）のうちの 1 つを指定します。
4	常に “1” がライトされます。
3:2	常に “0” がライトされます。
1:0	これらのビットは、コースページテーブルディスクリプタを示すために 01 でなければなりません。

7.3.6 ファインページテーブルディスクリプタ

ファインページテーブルディスクリプタは、ラージページ、スモールページ、あるいはタイニーページのアクセス用のレベル 2 ディスクリプタを含むページテーブルのベースアドレスを示します。ファインページテーブルは 1024 個のエントリをもち、テーブルが記述する 1MB を 1KB のブロックに分割しています。図 7-7 は、ファインページテーブルディスクリプタの形式を示します。

図 7-7 ファインページテーブルディスクリプタ

注意： ファインページテーブルディスクリプタがレベル 1 フェッチから返されると、レベル 2 フェッチが開始されます。

ファインページテーブルディスクリプタのビット割当を表 7-6 に示します。

表 7-6 ファインページテーブルディスクリプタのビット割当

ビット	説明
31:12	これらのビットは、レベル 2 ディスクリプタを参照するためのベースを形成します（エントリのファインページテーブルインデックスは MVA から得られます）。
11:9	常に “0” がライトされます。
8:5	これらのビットは、主要なアクセス制御を含んでいる 16 のドメイン（ドメインアクセス制御レジスタに保持されています）のうちの 1 つを指定します。
4	常に “1” がライトされます。
3:2	常に “0” がライトされます。
1:0	これらのビットは、ファインページテーブルディスクリプタを示すために b11 でなければなりません。

7.3.7 セクションリファレンスの変換

図 7-8 は、完全なセクション変換シーケンスを示しています。

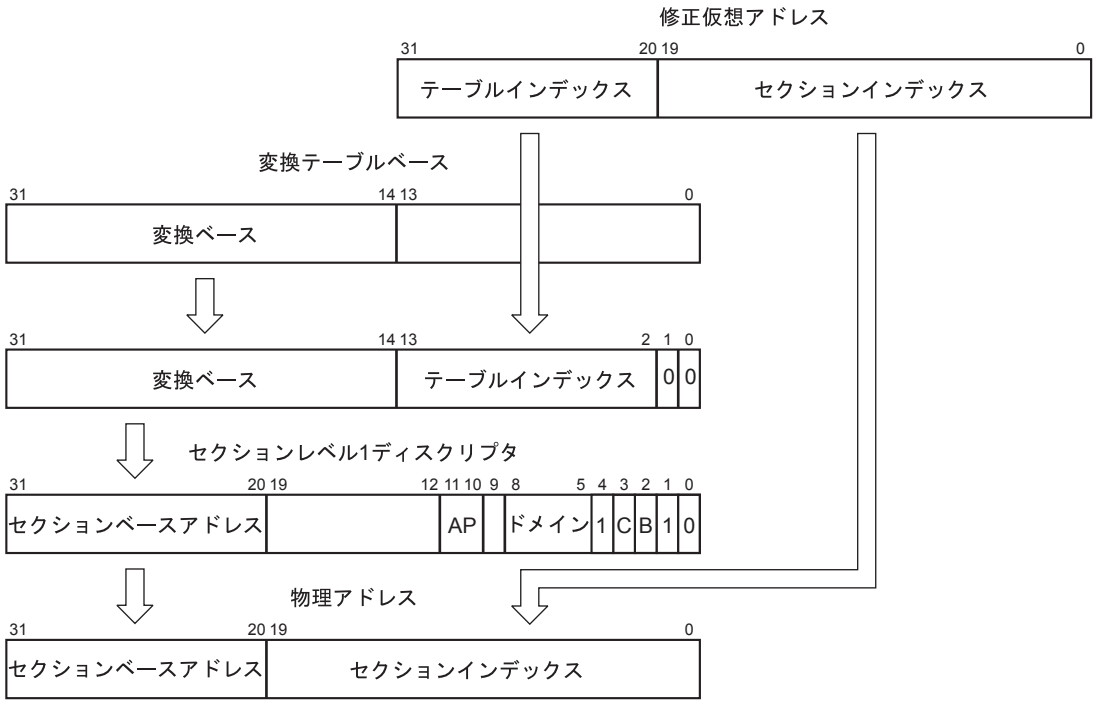


図 7-8 セクション変換

注意： 物理アドレスを生成する前に、レベル1ディスクリプタに含まれるアクセス許可をチェックしてください。

7.3.8 レベル 2 ディスクリプタ

レベル 1 フェッチがコースページテーブルディスクリプタかファインページテーブルディスクリプタのいずれかを返す場合、使用するページテーブルのベースアドレスが示されます。その後ページテーブルがアクセスされ、レベル 2 ディスクリプタが返されます。図 7-9 は、レベル 2 ディスクリプタの形式を示しています。

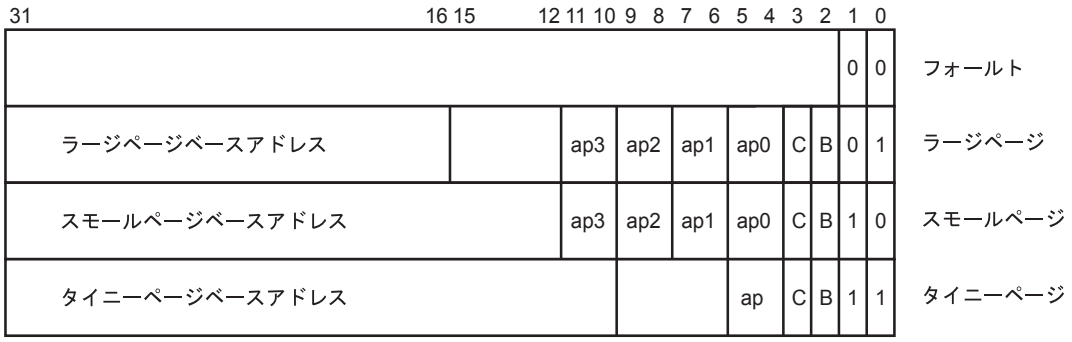


図 7-9 レベル 2 ディスクリプタ

レベル 2 ディスクリプタは、以下のようにタイニーページディスクリプタ、スモールページディスクリプタ、またはラージページディスクリプタを定義しているか、あるいは無効になっています。

- ラージページディスクリプタは、64KB のメモリブロックのベースアドレスを示します。
- スモールページディスクリプタは、4KB のメモリブロックのベースアドレスを示します。
- タイニーページディスクリプタは、1KB のメモリブロックのベースアドレスを示します。

コースページテーブルは、スモールページかラージページのいずれかのベースアドレスを示します。ラージページディスクリプタは、16 連続エントリごとにリピートされなければなりません。スモールページディスクリプタは、各連続エントリでリピートされなければなりません。

ファインページテーブルは、ラージページ、スモールページ、またはタイニーページのベースアドレスを示します。ラージページディスクリプタは、64 連続エントリごとにリピートさなければなりません。スモールページディスクリプタは 4 連続エントリごとに、またタイニーページディスクリプタは各連続エントリでリピートされなければなりません。

レベル 2 ディスクリプタのビット割当を表 7-7 に説明します。

表 7-7 レベル 2 ディスクリプタのビット割当

ビット			説明
ラージ	スモール	タイニー	
31:16	31:12	31:10	これらのビットは物理アドレスの対応ビットを形成します。
15:12	-	9:6	0 でなければなりません。
11:4	11:4	5:4	アクセス許可ビットです。*「ドメインアクセス制御」*の項（7-17 ページ）および *「フォールトチェックシーケンス」*の項（7-19 ページ）は、アクセス許可ビットの解釈方法を示しています。
3:2	3:2	3:2	これらのビット C、B は、このページでマップされるメモリ領域がキャッシュ可能かキャッシュ不可能なものとして、およびバッファ可能かバッファ不可能なものとして扱われるかどうかを示しています（システムは常にライトスルーです）。
1:0	1:0	1:0	これらのビットはページサイズと有効性を示しており、表 7-8 に示すように解釈されます。

レベル 2 ディスクリプタの最下位 2 ビットは、表 7-8 に示すようにディスクリプタタイプを示しています。

表 7-8 ページテーブルエントリビット [1:0] の解釈

値	意味	説明
0 0	無効	ページ変換フォールトを生成します。
0 1	ラージページ	64KB ページであることを示します。
1 0	スモールページ	4KB ページであることを示します。
1 1	タイニーページ	1KB ページであることを示します。

注意： タイニーページはサブページ許可に対応していません。したがって、アクセス許可ビットを 1 つもつのみです。

7.3.9 ラージページリファレンスの変換

図 7-10 は、64KB ラージページの完全な変換シーケンスを示しています。

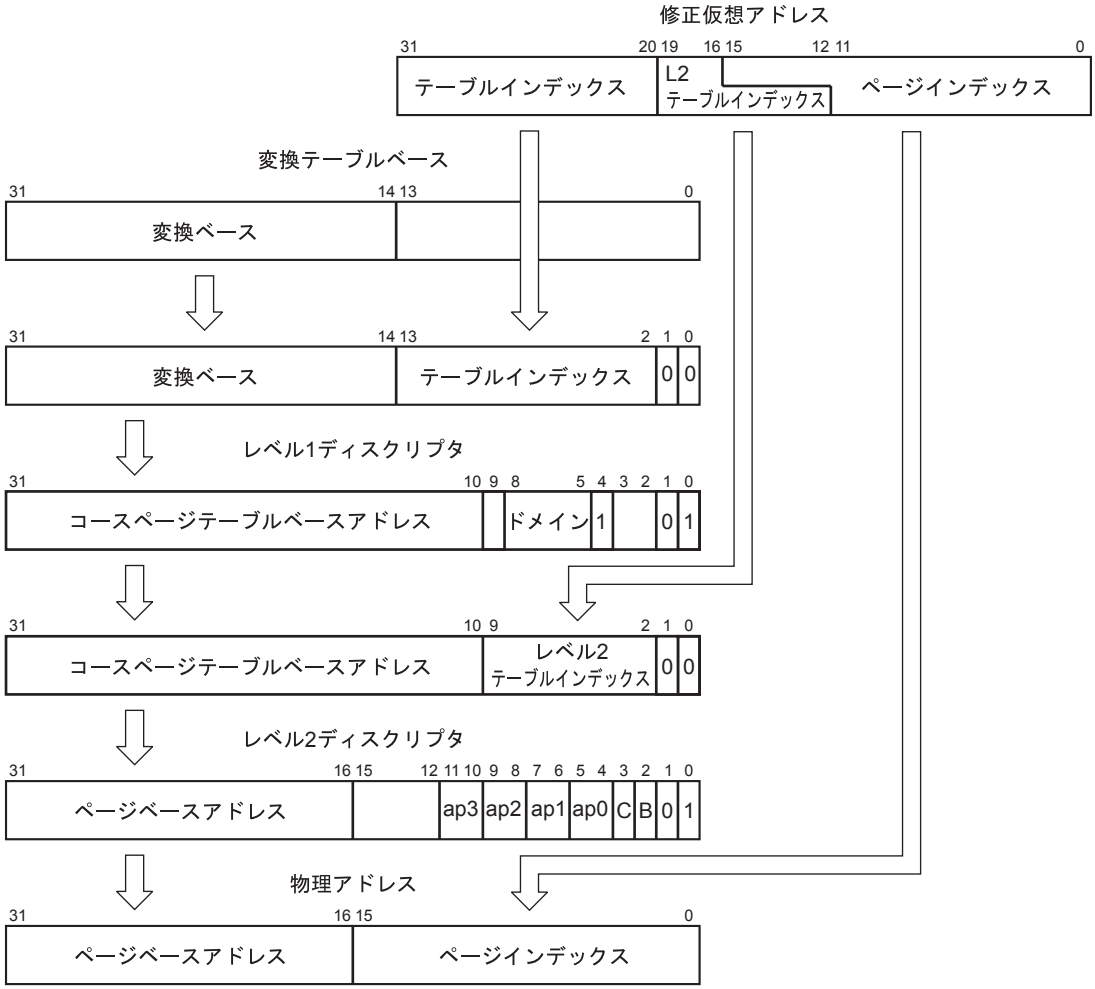


図 7-10 コースページテーブルからのラージページ変換

ページインデックスの上位 4 ビットとコースページテーブルインデックスの下位 4 ビットはオーバーラップしていますので、ラージページ用の各コースページテーブルエントリはコースページテーブル（の連続メモリ位置）で 16 回コピーされなければなりません。

ファインページテーブルにラージページディスクリプタが取り込まれている場合、ページインデックスの上位 6 ビットとファインページテーブルインデックスの下位 6 ビットはオーバーラップしています。したがって、ラージページ用の各ファインページテーブルエントリは 64 回コピーされなければなりません。

7.3.10 スモールページリファレンスの変換

図 7-11 は、4KB スモールページの完全な変換シーケンスを示しています。

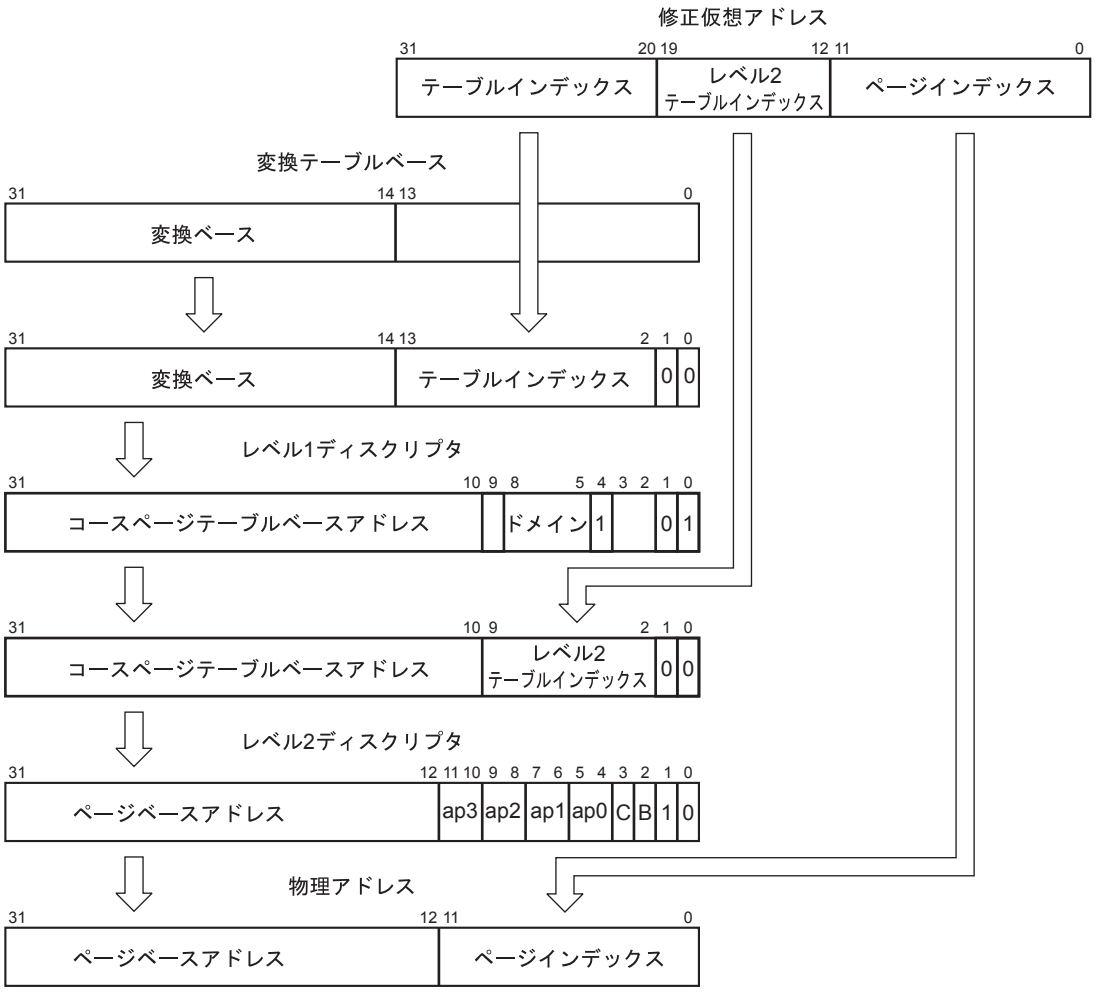


図 7-11 コースページテーブルからのスモールページ変換

ファインページテーブルにスモールページディスクリプタが取り込まれている場合、ページインデックスの上位 2 ビットとファインページテーブルインデックスの下位 2 ビットはオーバーラップしています。したがって、スモールページ用の各ファインページテーブルエントリは 4 回コピーされなければなりません。

7.3.11 タイニーページリファレンスの変換

図 7-12 は、1KB タイニーページの完全な変換シーケンスを示しています。

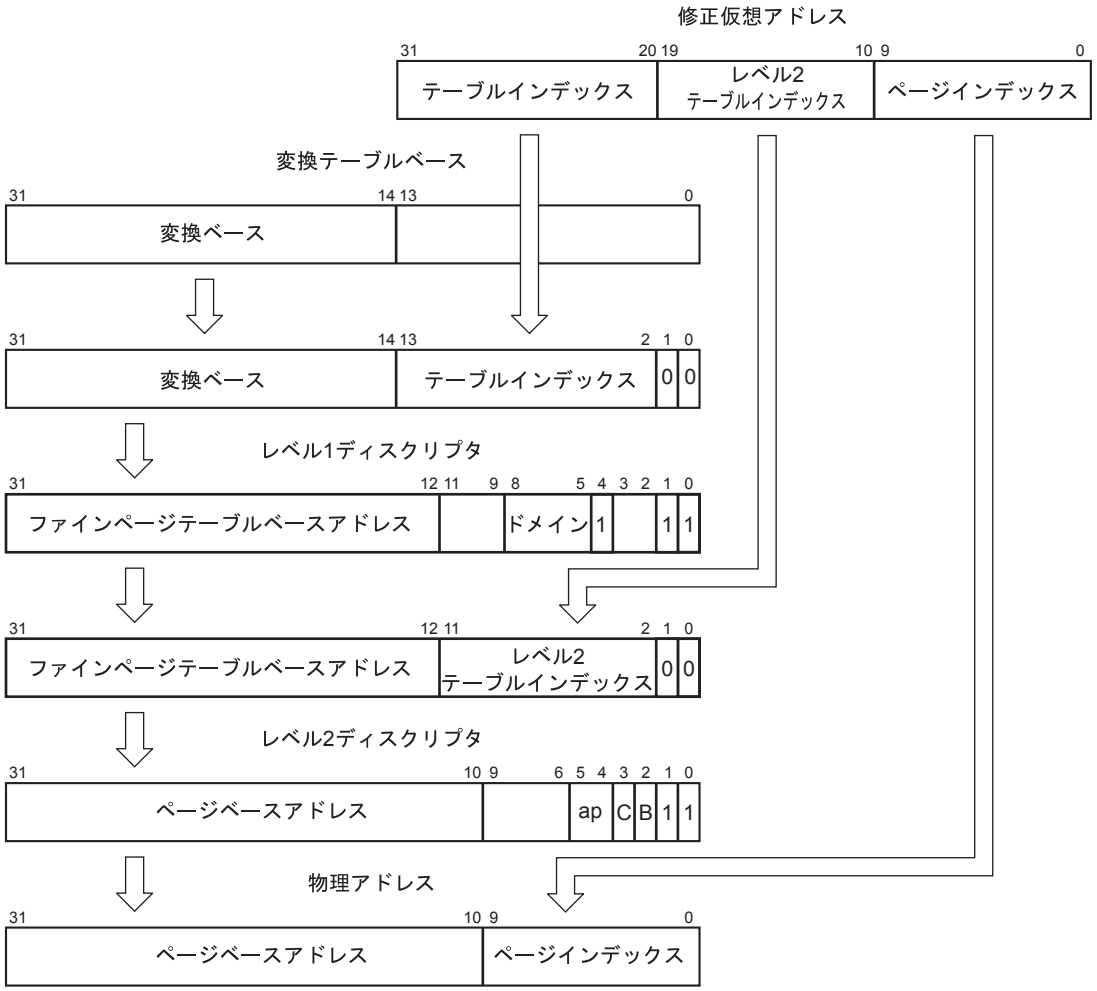


図 7-12 ファインページテーブルからのタイニーページ変換

ページ変換には、セクション変換の場合よりも 1 つステップが追加されています。レベル 1 ディスクリプタはファインページテーブルディスクリプタであり、これはレベル 1 ディスクリプタを指し示すために使用されます。

注意： レベル 1 記述に指定されるドメインとレベル 1 記述に指定されるアクセス許可はともに、アクセスが次へ進むための許可をもっているかどうかを決定します。詳細については、*「ドメインアクセス制御」*の項（7-17 ページ）を参照してください。

7.3.12 サブページ

スモールページとラージページのサブページについて、アクセス許可を定義することができます。ページウォーク中に、スモールページやラージページが、異なったサブページ許可を持っている場合、アクセス中のサブページだけが TLB に書き込まれます。たとえば、サブページ許可が異なる場合は 16KB（ラージページ）サブページエントリが TLB に書き込まれます。サブページ許可が同じ場合は、64KB エントリが TLB に配置されます。

サブページ許可を使用しており、ページエントリを無効にしなければならない場合は、4 つの全サブページを個別に無効化してください。

7.4 MMU フォールトと CPU アボート

MMU は、以下の 4 種類のフォールトが起こるとアボートを発生します。

- 整列フォールト（データアクセスのみ）
- 変換フォールト
- ドメインフォールト
- 許可フォールト

さらに、外部システムが外部アボートを発生させることもできます。これは、以下のようにコアが外部システムと同期化されたアクセスタイプの場合にのみ起こります。

- ノンキャッシャブルロード
- バッファ不可能ライト

整列フォールトチェックは、CP15 レジスタ c1 のビット A によってイネーブルされます。整列フォールトチェックは、MMU がイネーブルかどうかに影響されることはありません。MMU がイネーブルの場合、変換フォールト、ドメインフォールト、および許可フォールトが発生するだけです。

MMU のアクセス制御メカニズムは、これらのフォールトを発生させる状態を検出します。メモリアクセスの結果、フォールトが検出されると、MMU はアクセスをアボートし、CPU コアにフォールト状態を信号で伝えます。MMU は、データアクセスによって発生したフォールトに関するステータスとアドレス情報を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（*「フォールトアドレスとフォールトステータスレジスタ」*の項（7-16 ページ）を参照）に保持します。

任意のメモリアクセスでアクセス違反があると、対応する外部アクセスは禁止され、アボートにより CPU コアに返されます。

7.5 フォールトアドレスとフォールトステータスレジスタ

アボート時、MMU は 4 ビットにエンコードされたドメイン番号とともに、エンコードされた 4 ビット値 FS[3:0] をデータ FSR に置きます。また、このアボートと関連のある MVA が FAR にラッチされます。アクセス違反によって 2 つ以上のアボートソースが同時に生成されると、アボートは表 7-9 に示す優先順位でエンコードされます。

7.5.1 フォールトステータス

表 7-9 は、データ MMU がサポートする各種のアクセス許可と制御、およびこれらの許可や制御を解釈してフォールトを生成させる詳細な方法を説明しています。

表 7-9 フォールトステータスの優先順位のエンコーディング

優先順位	アボートソース	サイズ	ステータス	ドメイン	FAR
最高	整列	-	b00x1	無効	アボートを引き起こすアクセスの MVA
	変換	セクション ページ	b0101 b0111	無効 有効	アボートを引き起こすアクセスの MVA
	ドメイン	セクション ページ	b1001 b1011	有効 有効	アボートを引き起こすアクセスの MVA
	許可	セクション ページ	b1101 b1111	有効 有効	アボートを引き起こすアクセスの MVA
最低	キャッシュ不可能かつバッファ不可能なアクセス時の外部アボート、またはキャッシュ不可能ではあるがバッファ可能なリード時の外部アボート	セクション ページ	b1000 b1010	有効 有効	アボートを引き起こすアクセスの MVA

注意： 整列フォールト FS[3:0] に、b0001 または b0011 のいずれかをライトします。フォールトはページテーブルディスクリプタから有効なドメインフィールドがリードされる前に起こりますので、ドメイン [3:0] の値は無効になります。優先順位エンコーディングによってマスクされたアボートは、上位のアボートを修復し命令をリスタートさせることによって再生させることができます。

7.6 ドメインアクセス制御

MMU アクセスは、主にドメインを使用することによって制御されます。ドメインは 16 個あり、各ドメインはドメインへのアクセスを定義するための 2 ビットフィールドをもちます。2 種類のユーザ、すなわちクライアントとマネージャをサポートしています。ドメインは、ドメインアクセス制御レジスタに定義されます。図 7-13 は、レジスタの 32 ビットが 16 の 2 ビットドメインを定義するためにどのように割り当てられるかを示しています。

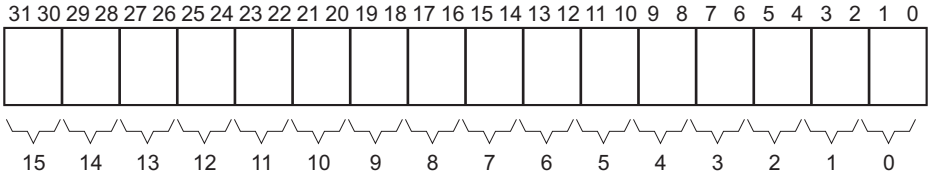


図 7-13 ドメインアクセス制御レジスタの形式

表 7-10 は、アクセス許可を指定するために各ドメインがどのように解釈されるかを定義しています。

表 7-10 ドメインアクセス制御レジスタのアクセス制御ビットの解釈

値	意味	説明
b00	ノーアクセス	どのアクセスもドメインフォールトを生成します。
b01	クライアント	アクセスは、セクションまたはページディスクリプタのアクセス許可ビットと照合されます。
b10	予約	予約済み。現在、ノーアクセスモードのように動作します。
b11	マネージャ	アクセスはアクセス許可ビットと照合されませんので、許可フォールトは生成されません。

表 7-11 は、*アクセス許可*（AP）ビットの解釈方法と、その解釈がどのように S ビットと R ビット（制御レジスタビット 8、9）に依存しているかを示しています。

表 7-11 アクセス許可（AP）ビットの解釈

AP	S	R	スーパーバイザ許可	ユーザ許可	説明
b00	0	0	ノーアクセス	ノーアクセス	どのアクセスにも許可フォールトが発生します。
b00	1	0	リードオンリ	ノーアクセス	スーパーバイザリードのみが許可されます。
b00	0	1	リードオンリ	リードオンリ	どの書き込みにも許可フォールトが発生します。
b00	1	1	予約	-	-[a]
b01	x	x	リード／ライト	ノーアクセス	アクセスはスーパーバイザモードでのみ許可されます。
b10	x	x	リード／ライト	リードオンリ	ユーザモードでライトを行うと、許可フォールトが発生します。
b11	x	x	リード／ライト	リード／ライト	両モードですべてのアクセスタイプが許可されます。
bxx	1	1	予約	-	-[a]

a. このエンコーディングは使用しないでください。[S:R] = b11 のとき、任意のアクセスに対してフォールトが生成されます。

7.7 フォールトチェックシーケンス

MMU がアクセスフォールトの有無をチェックするために使用するシーケンスは、セクションとページ間で異なります。両タイプのアクセスに関するシーケンスを図 7-14 に示します。

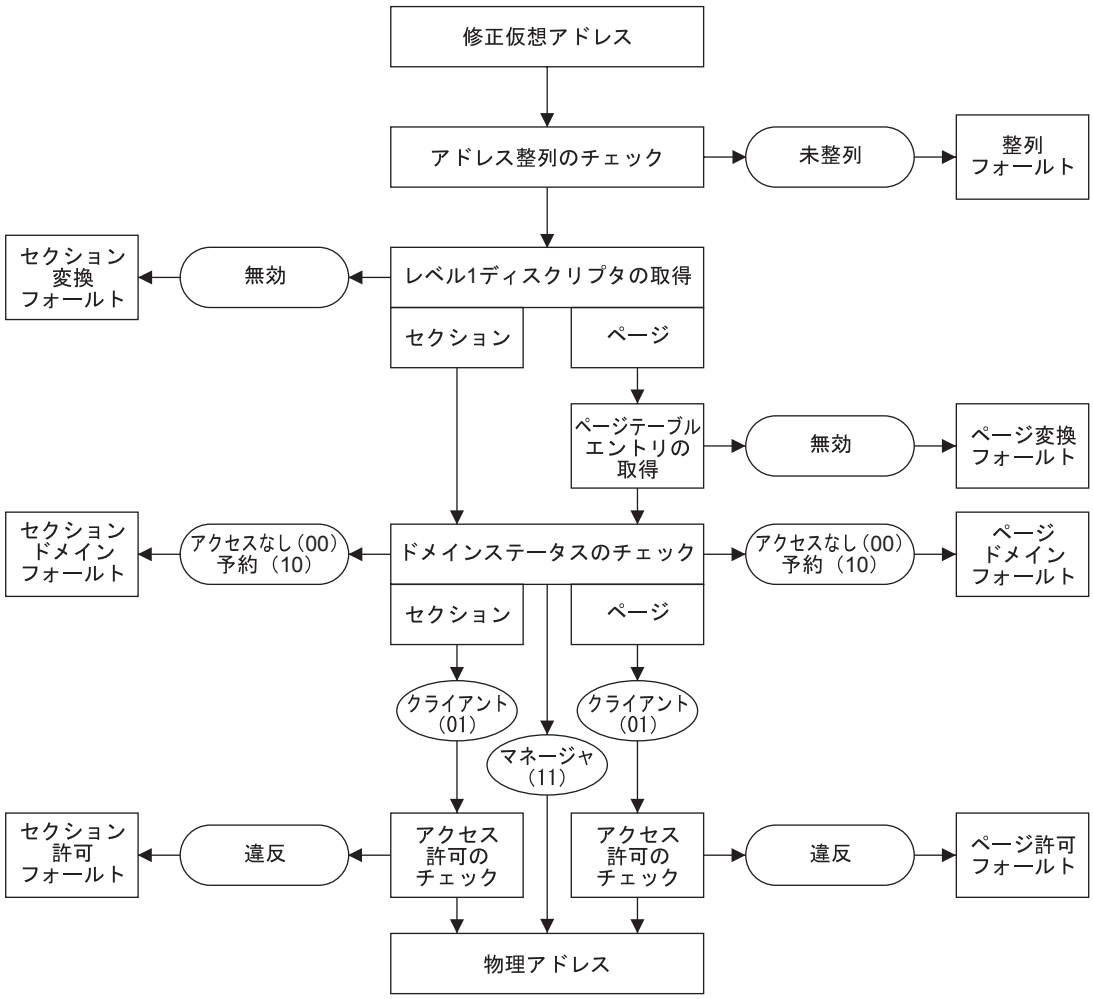


図 7-14 フォールトチェックシーケンス

各フォールトを発生させる条件は、以下の項で説明します。

- *整列フォールト*（7-19 ページ）
- *変換フォールト*（7-20 ページ）
- *ドメインフォールト*（7-20 ページ）
- *許可フォールト*（7-20 ページ）

7.7.1 整列フォールト

整列フォールトがイネーブルになっていると（CP15 レジスタ c1 のビット A がセットされていると）、データワードアクセス時にアドレスがワード整列されていない場合、MMU は整列フォールトを発生させます。また、ハーフワードアクセス時にアドレスがハーフワード整列されていない場合、MMU は整列フォールトを発生させます。これらは、MMU がイネーブルになっているかどうかにかかわらず行われます。整列フォールトは、命令フェッチ時やバイトアクセス時には発生しません。

注意： アクセスによって整列フォールトが発生すると、アクセスシーケンスはさらなる許可チェックに関係なくアボートします。

7.7.2 変換フォールト

変換フォールトには、以下の 2 種類があります。

セクション セクション変換フォールトは、レベル 1 ディスクリプタが無効とマークされる場合に発生します。これは、レベル 1 ディスクリプタのビット [1:0] がともに “0” になっていると起こります。

ページ ページ変換フォールトは、レベル 2 ディスクリプタが無効とマークされる場合に発生します。これは、レベル 2 ディスクリプタのビット [1:0] がともに “0” になっていると起こります。

7.7.3 ドメインフォールト

ドメインフォールトには、以下の 2 種類があります。

セクション レベル 1 ディスクリプタは、4 ビットドメインフィールドを保持しています。このフィールドは、ドメインアクセス制御レジスタの 16 個の 2 ビットドメインのうちの 1 つを選択します。そして、表 7-11（7-18 ページ）に説明するように、指定されたドメインの 2 ビットはアクセス許可を調べるために照合されます。ドメインは、レベル 1 ディスクリプタが返されたときに照合されます。

ページ レベル 1 ディスクリプタは、4 ビットドメインフィールドを保持しています。このフィールドは、ドメインアクセス制御レジスタの 16 個の 2 ビットドメインのうちの 1 つを選択します。そして、表 7-11（7-18 ページ）に説明するように、指定されたドメインの 2 ビットはアクセス許可を調べるために照合されます。ドメインは、レベル 1 ディスクリプタが返されたときに照合されます。

指定されたアクセスがアクセスなし（b00）または予約（b10）のいずれかの場合、セクションドメインフォールトまたはページドメインフォールとのいずれかが発生します。

7.7.4 許可フォールト

2 ビットドメインフィールドが 01（クライアント）を返してきた場合、アクセス許可は以下のようにチェックされます。

セクション レベル1ディスクリプタがセクションマップドアクセスを定義している場合、レベル 1 ディスクリプタの AP ビットは表 7-11（7-18 ページ）に従ってアクセスが許可されているかどうかを定義します。ビットの解釈は S ビットと R ビット（制御レジスタのビット 8、ビット 9）の設定に依存します。アクセスが許可されない場合、セクション許可フォールトが発生します。

ラージページとスモールページ

レベル 1 ディスクリプタがページマップドアクセスを定義し、レベル 2ディスクリプタがラージページやスモールページ用のものである場合、4 つのアクセス許可フィールド（AP3 ～ AP0) が指定され、各フィールドは 1/4 ページに対応しています。スモールページの場合、ap3 はページの上位 1KB で選択され、ap0 はページの下位 1KB で選択されます。ラージページの場合、ap3 はページの上位 16KB で選択され、ap0 はページの下位 16KB で選択されます。そして、選択された AP ビットはセクションの場合と同じように解釈されます(表 7-11（7-18 ページ）を参照してください)。発生するフォールトがページ許可フォールトであるという点が唯一異なります。

タイニーページ レベル 1 ディスクリプタがページマップドアクセスを定義し、レベル 2 ディスクリプタがタイニーページ用のものである場合、レベル 1 ディスクリプタの AP ビットはセクションの場合と同じようにアクセスが許可されているかどうかを定義します。発生するフォールトは、ページ許可フォールトです。

7.8 外部アボート

MMU が発生するアボートに加えて、ARM720T プロセッサを AMBA バスによって外部からアボートさせることができます。これを使用して、外部へのメモリアクセス時に発生するエラーに対してフラグを立てることができます。ただし、すべてのアクセスがこの方法でアボートできるわけではなく、*バスインタフェースユニット*（BIU）は処理することができない外部アボートを無視します。

以下のアクセスをアボートすることができます。

- キャッシュに保存されない読み出し
- バッファに格納されない書き込み
- ノンキャッシャブルメモリに対するリードロックライトシーケンス

リードロックライト（SWP）シーケンスの場合、読み出しがアボートされると、書き込みは行われません。

7.9 MMU とキャッシュの相互作用

MMU は、以下の項に説明するように CP15 制御レジスタ c1 のビット 0 を使用してイネーブルにしたり、ディセーブルにしたりできます。

- *MMU のイネーブル*
- *MMU のディセーブル*

7.9.1 MMU のイネーブル

MMU をイネーブルにするには、以下の手順をとります。

1 TTB とドメインアクセス制御レジスタを設定します。

2 レベル 1 とレベル 2 ページテーブルを必要に応じて設定します。

3 制御レジスタのビット 0 を設定して、MMU をイネーブルにします。

MMU のイネーブルの後に続くいくつかの命令は MMU をオフにして（フィジカル = VA – フラット変換を使用して）プリフェッチされている場合がありますので、変換アドレスが未変換アドレスと異なる場合は注意しなければなりません。

この場合、MMU のイネーブル化は遅延実行のある分岐とみなすことができます。MMU をディセーブルにするときにも同じような状況が発生します。以下のコードシーケンスを考慮してください。

```
MRC p15, 0, r1, c1, c0, 0; Read control register
ORR R1, R1, #0x01
MCR p15,0, r1, c1, c0, 0 ; Enable MMUS
Fetch Flat
Fetch Flat
Fetch Translated
```

7.9.2 MMU のディセーブル

MMU をディセーブルにするには、制御レジスタのビット 0 をクリアします。データキャッシュは、制御レジスタのビット 2 をクリアすることによって MMU をディセーブルにする前に、あるいは MMU のディセーブルと同時にディセーブルにしてください。プリフェッチの影響に関する*「MMU のイネーブル」*の項を参照してください。

注意： MMU をイネーブル、ディセーブルにし、その後再度イネーブルにした場合、TLB のコンテンツは保存されます。今、これらが無効な場合は、MMU を再度イネーブルにする前に TLB を無効にしてください。詳細については、*「TLB 動作レジスタ」*の項（3-7 ページ）を参照してください。

このページは白紙です。

8
コプロセッサインタフェース

8 コプロセッサインタフェース

この章では、ARM720T プロセッサのコプロセッサインタフェースを説明します。ここでは、以下の項を記載しています。

8.1 コプロセッサについて

ARM720T プロセッサの命令セットには、コプロセッサを使用する特殊な追加命令がインプリメントされています。これらは、ARM720T プロセッサと密接に連結した独立した処理ユニットといえます。代表的なコプロセッサは以下の機能をもちます。

- 命令パイプライン
- 命令デコーディングロジック
- ハンドシェイクロジック
- レジスタバンク
- 独自のデータパスをもった特殊な処理ロジック

コプロセッサはシステムの ARM720T プロセッサと同じデータバスに接続され、ARM720T コアのパイプラインをトラッキングします。これは、コプロセッサが命令ストリームの命令をデコードし、自らがサポートしている命令を実行することができることを意味しています。各命令は、ARM720T プロセッサパイプラインとコプロセッサパイプラインの両方で同時に進行します。

命令の実行は、以下のように ARM720T コアとコプロセッサ間で共有されます。

ARM720T コア：

1 条件コードを評価して命令がコプロセッサで実行されなければならないかどうかを決定し、これをシステムのコプロセッサに（**CPnCPI** を使用して）信号で伝えます。

2 命令で要求されるアドレスを生成します。これには、パイプラインを再充てんするための次の命令のプリフェッチも含まれます。

3 コプロセッサが命令を受け付けない場合は、未定義の命令トラップを実行します。

コプロセッサ：

1. 命令をデコードして、この命令を受け付けることができるかどうかを確認します。
2. 命令を受け付けることができるかどうかを（**EXTCPA** と **EXTCPB** で信号を送って）示します。
3. 独自のレジスタバンクから必要な値をフェッチします。
4. 命令で要求される動作を実行します。

コプロセッサが命令を実行することができない場合、命令は未定義の命令トラップを実行します。コプロセッサ機能をソフトウェアでエミュレートするか、あるいは専用のコプロセッサを設計するかを選択することができます。

8.1.1 コプロセッサの利用

システムには最大 16 個のコプロセッサを接続することができ、各コプロセッサは独自のコプロセッサ ID 番号をもっています。

コプロセッサ番号の中には予約されているものがあります。たとえば、コプロセッサ番号 14、15 は ARM720T プロセッサ内部で使用されていますので、外部コプロセッサをこれらの番号に割り付けることはできません。

- CP14 は通信チャネルコプロセッサです。
- CP15 は、キャッシュおよび MMU 機能向けシステム制御コプロセッサです。

コプロセッサの利用割り当てについて表 8-1 に示します。

表 8-1 コプロセッサの利用

コプロセッサ番号	割当
15	システム制御
14	デバッグコントローラ
13:8	予約
7:4	ユーザが使用可能
3:0	予約

注意： コプロセッサを設計する場合は、件名に「coprocessor」と記載して E メールを info@arm.com に送信して、既に割り当てられているコプロセッサ番号の最新情報を取得してください。

8.2 コプロセッサインタフェース信号

ARM720T コアをコプロセッサとインタフェースをとるために使用する信号は、4 種類に分類されます。

クロック信号、クロック制御信号には、メインプロセッサクロックとバスリセットが含まれます。

- **HCLK**
- **EXTCPCLKEN**
- **HRESETn.**

パイプライン監視信号は次の通りです。

- **CPnMREQ**
- **CPnTRANS**
- **CPnOPC**
- **CPTBIT.**

ハンドシェイク信号は次の通りです。

- **CPnCPI**
- **EXTCPA**
- **EXTCPB.**

データ信号は次の通りです。

- **EXTCPDIN[31:0]**
- **EXTCPDOUT[31:0]**
- **EXTCPDBE.**

これらの信号とその使い方は、以下の項で説明しています。

- *パイプライン監視信号*、8-4 ページ
- *コプロセッサインタフェースハンドシェイキング*、8-5 ページ
- *コプロセッサの接続*、8-9 ページ
- *外部コプロセッサを使用しない場合*、8-10 ページ
- *未定義命令*、8-10 ページ
- *特権命令*、8-10 ページ

8.3 パイプライン監視信号

システムの各コプロセッサは、ARM720T プロセッサパイプラインで実行される命令を監視するためにパイプライン監視機能をもっていなければなりません。コプロセッサは、ARM720T プロセッサ入力データバス **EXTCPDOUT[31:0]**、**HCLK**、および **EXTCPCLKEN** に接続され、命令は ARM720T プロセッサ入力データバスを介してフェッチされます。

2 つのパイプラインが常に同期していることが不可欠です。コプロセッサ用のパイプライン監視機能を設計する場合は、以下の規則を遵守してください。

- リセット時 (**HRESETn** が Low)、パイプラインは無効になっているか、あるいは該当のコプロセッサを有効にするデコード不要の命令（たとえば NOP 命令等）で満たされていなければなりません。
- コプロセッサの状態は、(リセットを除き)**EXTCPCLKEN** が High のときにのみ変化するようにしてください。
- 命令は、**HCLK** の立ち上がりエッジで、かつ **CPnOPC**、**CPnMREQ**、および **CPTBIT** が前回のバスサイクルですべてLowになった場合にのみパイプラインにロードされなければなりません。

  これらの条件は、このサイクルが ARM 状態オペコードフェッチであり、新しいオペコードがパイプラインにサンプリングされなければならないことを示しています。
- パイプラインは、**CPnOPC**、**CPnMREQ**、および **CPTBIT** が現在のバスサイクルですべて Low になっているときに **HCLK** の立ち上がりエッジで進むようにさせてください。

  これらの条件は現在の命令がその実行を完了しようとしていることを示していますが、これは命令フェッチを実行する命令の最初の動作がパイプラインを再充てんすることであるためです。

ARM720T プロセッサパイプラインからフラッシュされる命令は、実行段階に入ったことを **CPnCPI** で伝えないため、これらの命令はパイプラインを再充てんするために要求されるプリフェッチによって自動的にコプロセッサパイプラインからフラッシュされます。

Thumb 命令セットにはコプロセッサ命令がないため、コプロセッサは **CPTBIT** 信号の状態を監視して、ARM 命令のようにペアの Thumb 命令をデコードしないようにしなければなりません。

8.4 コプロセッサインタフェースハンドシェイキング

システムの ARM720T コアとコプロセッサは、表 8-2 に示す信号を使用してハンドシェイクを実行します。

表 8-2 ハンドシェイキング信号

信号	方向	意味
CPnCPI	ARM720T コアからコプロセッサへ	コプロセッサ命令ではない
EXTCPA	コプロセッサから ARM720T コアへ	コプロセッサなし
EXTCPB	コプロセッサから ARM720T コアへ	コプロセッサビジー

これらの信号は、*「コプロセッサ発信信号」*の項（8-6 ページ）で詳細に説明しています。

8.4.1 コプロセッサ

コプロセッサは、現在そのパイプラインのデコード段階にある命令をデコードし、その命令がコプロセッサ命令かどうかを照合します。コプロセッサ命令には、コプロセッサの ID とマッチしたコプロセッサ番号があります。

現在デコード段階にある命令がコプロセッサ命令の場合、以下の動作がとられます。

1 コプロセッサは命令を実行しようとします。

2 コプロセッサは、**EXTCPA** と **EXTCPB** を使用して ARM720T コアに信号を返します。

8.4.2 ARM720T コア

コプロセッサ命令は、コプロセッサパイプラインと同期して ARM720T プロセッサパイプラインを進行します。コプロセッサ命令は、以下のものが真であれば実行されます。

1 コプロセッサ命令がパイプラインの実行段階に達した (その前に分岐があると達していない場合もあります)。

2 命令が、その条件実行テストを合格した。

3 システムのコプロセッサは命令を受け入れ可能であることを示す **EXTCPA** と **EXTCPB** で信号を送った。

これらの要件がすべて満たされると、ARM720T プロセッサは **CPnCPI** を Low にして信号を送ります。これによって、コプロセッサはコプロセッサ命令の実行をゆだねられます。

8.4.3 コプロセッサ発信信号

コプロセッサは、以下のように信号を送ります。

コプロセッサなし　コプロセッサが、現在デコード段階にある命令を受け付けることができない場合は、**EXTCPA** も **EXTCPB** も High のままにしておかなければなりません。

コプロセッサ有り

コプロセッサが命令を受け付け、直ちにその命令を開始することができる場合は、**EXTCPA** も **EXTCPB** も Low にしてこれを知らせなければなりません。

コプロセッサビジー（ビジーウェイト）

コプロセッサが命令を受け付けることはできるが、現在その命令を処理することができない場合は、ビジーウェイトをアサートして ARM720T コアを機能停止させることができます。これは **EXTCPA** を Low にし、**EXTCPB** は High のままにしておいて信号を送ります。コプロセッサがいつでも命令を実行する準備ができたときは、**EXTCPB** を Low にしてこれを知らせます。これを図 8-1 に示します。

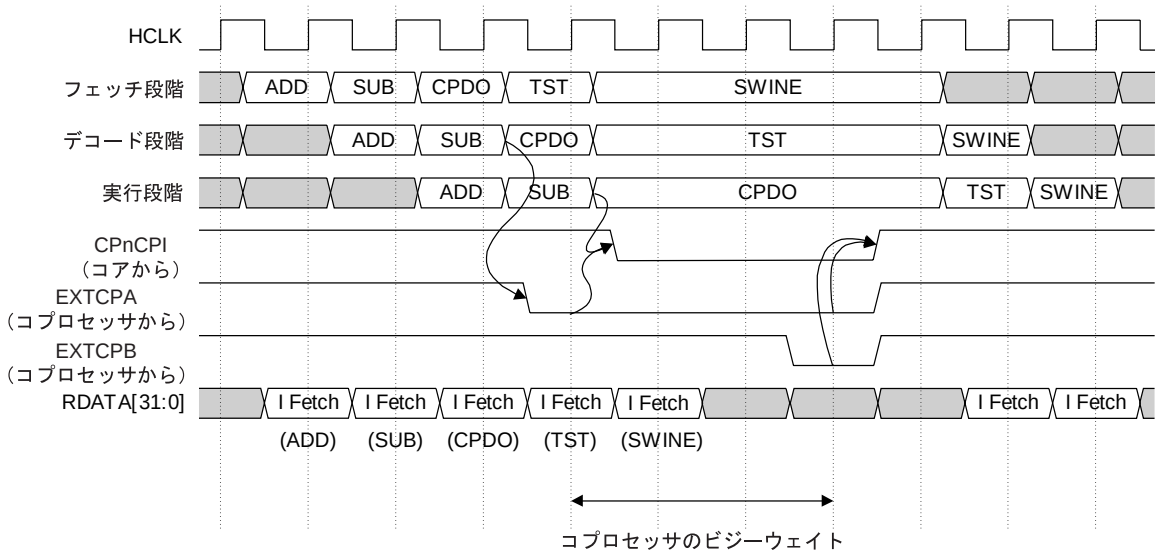


図 8-1 コプロセッサのビジーウェイトシーケンス

8.4.4 ビジーウェイトの結果

ビジーウェイト状態のコプロセッサ命令を、割り込みすることができます。有効な FIQ または IRQ が発生すると（CSPR で該当のビットがクリアされます）、ARM720T プロセッサはコプロセッサ命令を破棄し、**CPnCPI** を High にしてこれを知らせます。ビジーウェイトすることができるコプロセッサは **CPnCPI** を監視して、この状態を検出しなければなりません。ARM720T コアがコプロセッサ命令を破棄すると、コプロセッサも命令を破棄し、ARM720T プロセッサパイプラインの監視を続けます。

注意：　コプロセッサがビジーウェイトしている間にとる動作は、同じ効果があることが必須です。コプロセッサがとる動作によってコプロセッサの状態を転化されてはならず、繰り返し同じ結果が得られなければなりません。コプロセッサは、命令が実行された後のみ自己の状態を変更することができます。

8.4.5 コプロセッサのレジスタ転送命令

コプロセッサのレジスタ転送命令 MCR と MRC は、ARM720T プロセッサレジスタバンクのレジスタとコプロセッサレジスタバンクのレジスタ間でデータを転送します。コプロセッサレジスタ転送のシーケンス例を図 8-2 に示します。

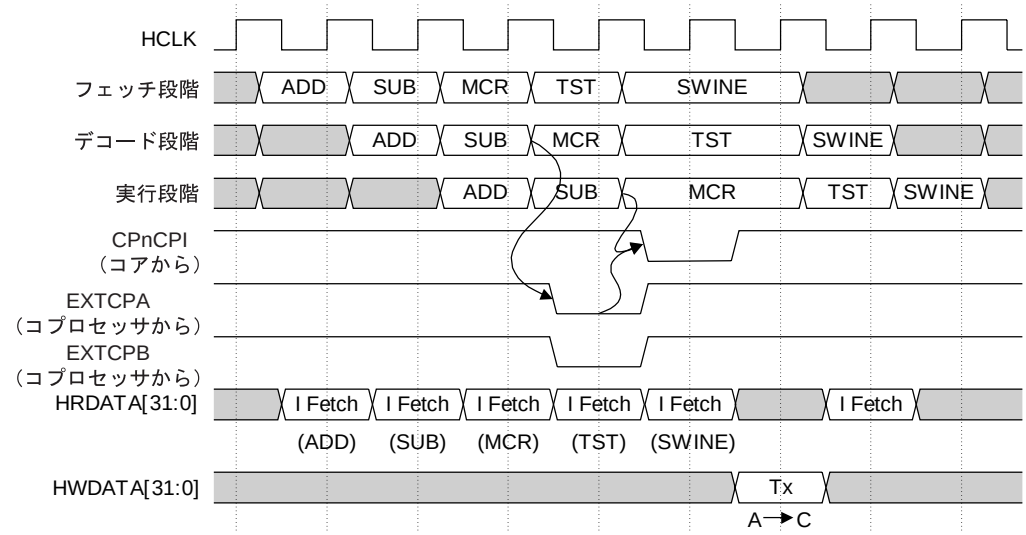


図 8-2 コプロセッサのレジスタ転送シーケンス

8.4.6 コプロセッサデータ操作

コプロセッサデータ処理命令 CDP は、コプロセッサレジスタバンクに保持されているデータの処理操作を実行します。この操作の結果、ARM720T コアとコプロセッサ間で情報は転送されません。シーケンス例を図 8-3 に示します。

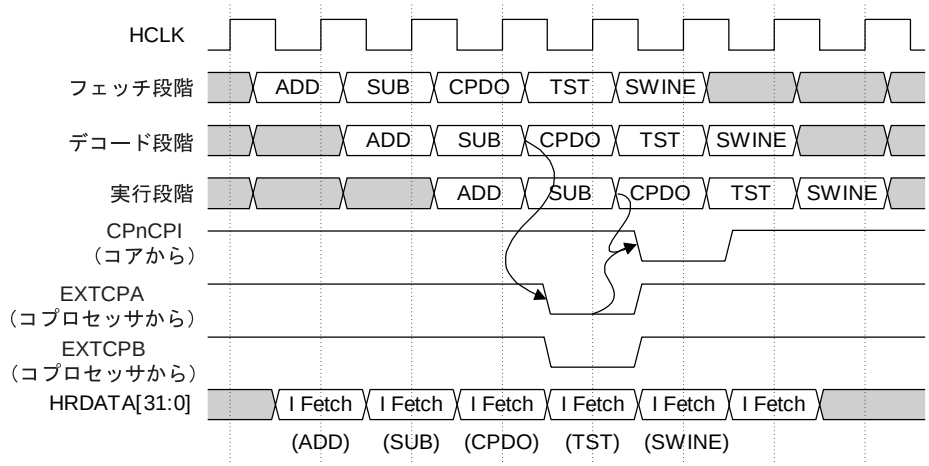


図 8-3 コプロセッサのデータ操作シーケンス

8.4.7 コプロセッサのロード、ストア操作

コプロセッサのロードおよびストア命令 LDC と STC は、コプロセッサとメモリ間でデータを転送するために使用されます。これらの命令は、1 ワードのデータか、あるいは複数のコプロセッサレジスタのいずれかを転送するために使用されます。1 つの LDC 命令や STC 命令で転送することができるデータのワード数には制限はありませんが、規定によれば、コプロセッサは 1 つの命令で16ワードを超えるデータを転送することはできません。シーケンス例を図 8-4に示します。

注意：

- LDC、STC 命令が CP15 操作としてデコードできない限り、外部コプロセッサはこれらの命令でアボートしてはいけません。そうでないと、ビジーウェイト時にデッドロックが発生します。
- 1 つの命令で 16 ワードを超えるデータを転送しようとすると、ARM720T プロセッサの最悪状態割り込みレイテンシが増加します。

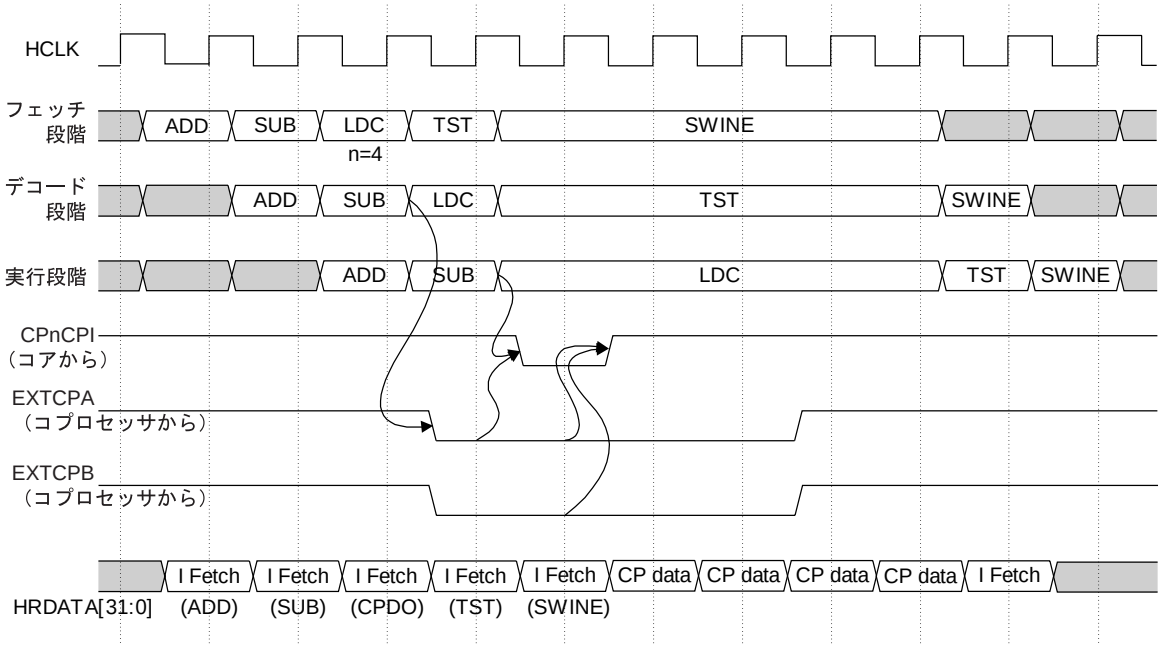


図 8-4 コプロセッサのロードシーケンス

8.5 コプロセッサの接続

ARM720T プロセッサに基づいたシステム内のコプロセッサは、以下の動作を行うために 32 ビットの接続を持つ必要があります。

- メモリからのデータ転送（命令ストリームと LDC）
- ARM720T プロセッサからのデータのライト（MCR）
- ARM720T プロセッサへのデータのリード（MRC）

8.5.1 1 つのコプロセッサを接続する

図 8-5 に示すように、ロジックを追加せずに 1 つのコプロセッサを直接 ARM720T プロセッサのコプロセッサインタフェースに接続することができます。外部コプロセッサが **EXTCPDOUT** のデータを駆動する場合、**EXTCPDBE** は外部プロセッサで High にしてください。

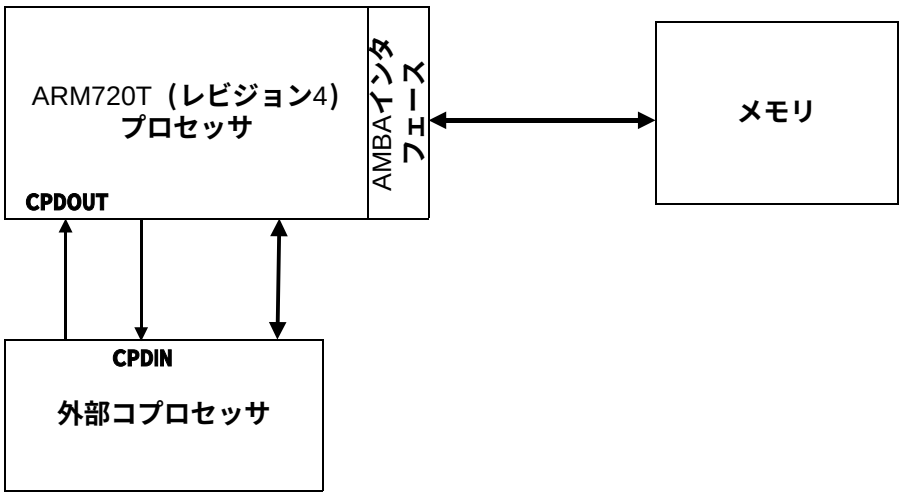


図 8-5 コプロセッサの接続例

. **注意：** ETM7 と ARM720T コアでシステムを構築している場合は、以下のバスを直接接続してください。

- ETM7 の入力 **RDATA[31:0]** を ARM720T プロセッサ出力 **ETMRDATA[31:0]** に
- ETM7 の入力 **WDATA[31:0]** を ARM720T プロセッサ出力 **ETMWDATA[31:0]** に

これによって、ETM はコプロセッサ命令を正しくトレースすることができます。

8.5.2 複数のコプロセッサの接続

システムに複数のコプロセッサがある場合は、表 8-3 に示すようにハンドシェイク信号を接続します。

表 8-3 ハンドシェイク信号の接続

信号	接続
CPnCPI	この信号を、システムに存在するすべてのプロセッサに接続します。
CPA および CPB	各コプロセッサからの個々の CPA および CPB 出力はそれぞれ論理積をとって、ARM720T プロセッサの EXTCPA および EXTCPB 入力に接続しなければなりません。

また、複数のコプロセッサからの出力データはマルチプレクスさせてください。

8.6 外部コプロセッサを使用しない場合

外部プロセッサを含まないシステムをインプリメントする場合は、**EXTCPA** と **EXTCPB** の両方を High に固定してください。これは、システムに外部コプロセッサが存在していないことを示します。コプロセッサ命令を受信すると、これらの命令は必要に応じてソフトウェアでエミュレートできるように未定義命令トラップを実行します。

ARM720T コアからのコプロセッサ用の以下の出力は、未接続のままにすることができます。

- **CPnMREQ**
- **CPnTRANS**
- **CPnOPC**
- **CPnCPI**
- **CPTBIT.**

EXTCPDOUT をオフに固定してください。

外部コプロセッサデータバスイネーブル **EXTCPDBE** を Low に固定してください。

8.7 STC 操作

外部コプロセッサを使用している場合は、キャッシュがイネーブルの場合にキャッシャブル領域で STC 操作を行うことができます。ただし、STC 操作は AMBA バス上で一連のノンシーケンシャル転送として扱われます。

8.8 未定義命令

ARM720T プロセッサは、ARM アーキテクチャ v4T 未定義命令処理を完全にインプリメントしています。これは、*「ARM Architecture Reference Manual」*で「**未定義**」と定義されている命令によってARM720Tプロセッサは自動的に未定義命令トラップを実行するということを意味しています。コプロセッサで受け付けられないコプロセッサ命令の場合も、ARM720T プロセッサは未定義命令トラップを実行します。

8.9 特権命令

出力信号 **CPnTRANS** を使用して、特権モードでのみアクセス可能なコプロセッサ、またはコプロセッサ命令をインプリメントすることができます。この信号の意味を、表 8-4 に示します。

表 8-4 CPnTRANS **信号の意味**

CPnTRANS	意味
Low **の状態**	**ユーザモード命令**
High **の状態**	**特権モード命令**

CPnTRANS 信号は、命令と同時にサンプリングされ、要因としてコプロセッサパイプラインのデコード段階に入れられます。

注意： ユーザモードプロセス（**CPnTRANS** は Low の状態）が特権モードでしか実行することができないコプロセッサ命令にアクセスをしようとすると、コプロセッサは **EXTCPA** と **EXTCPB** を High にして応答しなければなりません。これによって、ARM720T プロセッサは未定義命令トラップを実行します。

9
システムのデバッグ

9 システムのデバッグ

この章では、ARM720T プロセッサをベースにしたシステムをデバッグする方法を解説します。ここでは、以下の項を記載しています。

9.1 システムのデバッギングについて

ARM720T プロセッサの高度デバッギング機能によって、アプリケーションソフトウェア、オペレーティングシステム、およびハードウェア自体を容易に開発することができます。

9.1.1 代表的なデバッグシステム

ARM720T プロセッサは、実行される高レベルデバッギングから ARM720T プロセッサがサポートする低レベルインタフェースまでを接続するデバッグシステムの 1 コンポーネントを形成します。図 9-1 に代表的なデバッグシステムを示します。

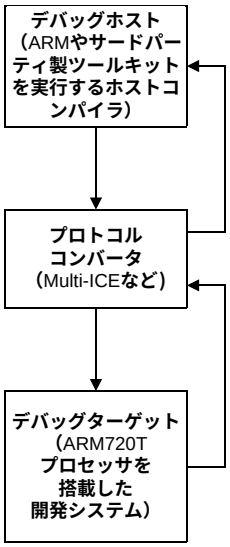


図 9-1 代表的なデバッグシステム

デバッグシステムには、通常以下の 3 つの部分があります。

デバッグホスト *ARM Debugger for Windows* (ADW) などのソフトウェアデバッガを実行するコンピュータです。デバッグホストによって、ブレークポイントの設定やメモリの内容の検査などの高度なコマンドを出すことができます。

プロトコルコンバータ これは、デバッグホストが発行する高度なコマンドと ARM720T プロセッサ JTAG インタフェースの低レベルコマンド間のインタフェースをとります。通常、拡張パラレルポートなどのインタフェースを介してホストに接続されます。

デバッグターゲット ARM720T プロセッサには、最低レベルでのデバッギングを容易にするハードウェア拡張機能があります。これらの拡張機能を使用すると、以下のことが可能です。

- プログラム実行の停止
- コアの内部状態の調査と変更
- メモリシステムの状態の調査
- アボート例外を実行して、コアのリアルタイムモニタリングを可能にする。
- プログラム実行の再開

デバッグホストとプロトコルコンバータはシステムに依存します。

9.2 デバッギングの制御

ARM720T プロセッサの主要なブロックは、以下の通りです。

ARM CPU コア　これは、デバッグのハードウェアサポートを備えています。

EmbeddedICE-RT マクロセル

デバッグ例外（ブレークポイントなど）を生成するために使用するレジスタとコンパレータのセット。このユニットについては、*「EmbeddedICE-RT マクロセル」*の項（9-10 ページ）に説明しています。

TAP コントローラ　JTAG シリアルインタフェースを使用してスキャンチェインの動作を制御します。詳細については、*「TAP コントローラ」*の項（9-19 ページ）を参照してください。

これらのブロックを図 9-2 に示します。

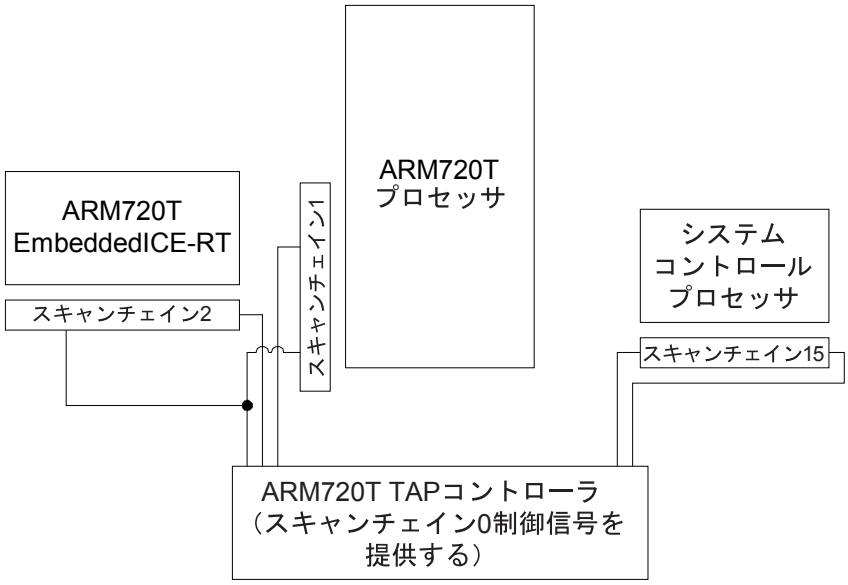


図 9-2 ARM720T **プロセッサブロック図**

9.2.1 デバッグモード

デバッギングは、以下のモードのいずれかで行うことができます。

ホルトモード

システムがホルトモードのときに、ARM720T コアはブレークポイントやウォッチポイントを検出するとデバッグ状態になります。デバッグ状態では、コアは停止し、システムの他の部分から切り離されます。デバッグが完了すると、デバッグホストはコアとシステム状態を復元しますので、プログラムの実行が再開されます。

詳細については、*「デバッグ状態の開始」*の項（9-5 ページ）を参照してください。

モニタモード

システムがモニタモードのときに、ARM720T コアはブレークポイントやウォッチポイントを検出してもデバッグ状態にはなりません。代わりに、命令アボートやデータアボートが生成され、コアは通常通りに割り込みの受け付け、処理を継続します。アボートステータスレジスタを使用すると、例外がブレークポイントやウォッチポイントによるものか、あるいは純粋なメモリアボートによるものかを確かめることができます。

詳細については、*「モニタモードデバッギング」*の項（9-12 ページ）を参照してください。

9.2.2 デバッギング時のシステム状態の調査

ホルトモードでもモニタモードでも、JTAG 形式シリアルインタフェースを使用すると、システムが稼動している間にコアの内部状態やシステムの外部状態を調べることができます。

ホルトモードでは、これによって、外部データバスを使用せずに命令をコアパイプラインにシリアルに挿入することができます。たとえば、デバッグ状態では、*「マルチプルストア」*（STM）を命令パイプラインに挿入して、ARM720T プロセッサレジスタの内容を取り出すことができます。このデータは、システムの残りの部分に影響を与えずにシリアルにシフトアウトさせることができます。詳細については、*「デバッグ状態のコアおよびシステムの調査」*の項（9-27 ページ）を参照してください。

モニタモードでは、JTAG インタフェースを使用して、デバッガと、ARM720T コアで実行されるシンプルな監視プログラム間でデータを転送します。

スキャンチェインと JTAG インタフェースの詳細については、*「スキャンチェインと JTAG インタフェース」*の項（9-17 ページ）を参照してください。

9.3 デバッグ状態の開始

システムがホルトモードの場合、以下の割り込みタイプのいずれかでも発生すると、プロセッサを強制的にデバッグ状態にします。

- ブレークポイント（ある特定の命令フェッチ）
- ウォッチポイント（データアクセス）
- 外部デバッグ要求

注意： モニタモードでは、プロセッサは引き続きリアルタイムで命令を実行し、アボート例外を処理します。アボートステータスレジスタを使用すると、例外がブレークポイントやウォッチポイントによるものか、あるいは純粋なメモリアボートによるものかを明らかにすることができます。

EmbeddedICE-RT ロジックを使用して、ブレークポイントやウォッチポイントが発生する条件を設定することができます。あるいは、**DBGBREAK** 信号を使用して外部ロジックによりブレークポイントやウォッチポイントにフラグを立てたり、以下の項目を監視したりすることができます。

- アドレスバス
- データバス
- 制御信号

外部で生成されたブレークポイントやウォッチポイントのタイミングは同じです。データは、常に **HCLK** の立ち上がりエッジ周辺で有効でなければなりません。このデータが、ブレークポイントが設定された命令のときは、**DBGBREAK** 信号は **HCLK** の立ち上がりエッジ周辺で High にならなければなりません。同様に、データがロードやストア用であるときには、**HCLK** の立ち上がりで **DBGBREAK** 信号をアサートすることによって、そのデータをウォッチポイントとしてマークします。

ブレークポイントやウォッチポイントが生成されると、ARM720T コアがデバッグ状態になる前に遅延が発生することがあります。コアがデバッグ状態になると、**DBGACK** 信号がアサートされます。外部で生成されたブレークポイントのタイミングを図 9-3 に示します。

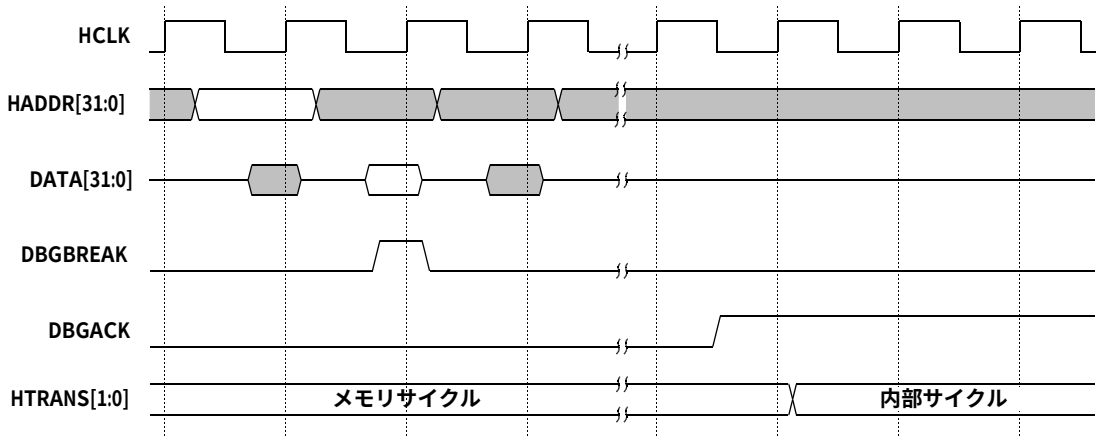


図 9-3 デバッグ状態の開始

9.3.1 ブレークポイントによるデバッグ状態の開始

AMM720T プロセッサは命令が命令パイプラインに入るとブレークポイントが設定されたものとして命令をマークしますが、コアは命令が実行段階になるまでデバッグ状態にはなりません。

ブレークポイントが設定された命令は実行されません。代わりに、ARM720T コアがデバッグ状態になります。内部状態を調べてみると、ブレークポイントが設定された命令の前の状態を確認することができます。検査が完了したら、ブレークポイントを外します。プログラムの実行は、前回ブレークポイントを設定した命令から再開されます。

ブレークポイントが設定された条件付命令がパイプラインの実行段階に達すると、システムがホルトモードであればブレークポイントは必ず機能します。ARM720T コアは、命令条件が満たされているかどうかにかかわらずデバッグ状態に入ります。

以下の場合、ブレークポイントが設定された命令によって ARM720T コアがデバッグ状態になることはありません。

- 分岐やPCへの書き込みがブレークポイントが設定された命令よりも前に起こります。この場合、分岐が実行されると、ARM720T プロセッサは命令パイプラインをフラッシュしますので、ブレークポイントは取り消されます。
- 例外が発生して、ARM720T プロセッサは命令パイプラインをフラッシュし、ブレークポイントを取り消します。通常、例外終了時、ARM720T コアは例外が発生する前次に実行されることになっていた命令に再度分岐されます。この場合、パイプラインは再充てんされ、ブレークポイントには再度フラグが立ちます。

9.3.2 ウォッチポイントによるデバッグ状態の開始

ウォッチポイントは、データアクセス時に発生します。ホルトモードでは、コア処理は停止します。モニタモードでは、アボート例外が実行されます（*「アボート」*の項（2-13 ページ）を参照してください）。ウォッチポイントは必ず機能しますが、ホルトモードになっているコアはすぐにはデバッグ状態にならない場合があります。これは、現在の命令が必ず完了する必要があるからです。現在の命令がマルチワードのロードやストア（LDM や STM）の場合、多くのサイクルが経過してからウォッチポイントが実行されます。

ウォッチポイントでは、以下のシーケンスが発生します。

1 現在の命令が完了します。
2 コア状態の変更がすべて行われます。
3 ロードデータは、デスティネーションレジスタに書き込まれます。
4 ベースライトバックが実行されます。

注意： ウォッチポイントはデータアボートに似ています。異なる点は、データアボートが発生すると、命令は完了しますが、ARM720T コアは ARM720T プロセッサ状態の以降の変更がいっさい行われないようにするという点です。この動作によって、アボートハンドラはアボートの原因を解決することができますので、命令は再度実行されます。

例外を保留にしているときにウォッチポイントが発生すると、コアは例外と同じモードでデバッグ状態になります。

9.3.3 デバッグ要求によるデバッグ状態の開始

ホルトモードになっている ARM720T コアは、デバッグ要求時、以下のいずれかの方法で強制的にデバッグ状態にさせることができます。

- EmbeddedICE-RT をプログラムする（*「ブレークポイントの設定」*の項（9-37 ページ）および *「ウォッチポイントの設定」*の項（9-39 ページ）を参照してください）。
- **DBGRQ** ピンをアサートする。

DBGRQ は、**DBGACK** がアサートされるのと同じクロックでディアサートされなければなりません。

DBGRQ ピンがアサートされていると、通常、コアは現在の命令が終わるとデバッグ状態に入ります。ただし、現在の命令がコプロセッサに対するビジーウェイティングアクセスの場合、命令は終了し、ARM720T コアはすぐにデバッグ状態になります。これは、**nIRQ** と **nFIQ** の動作に似ています。

9.3.4 デバッグ状態での ARM720T プロセッサの動作

ARM720T プロセッサがデバッグ状態になると、コアは強制的に **HTRANS[1:0]** に内部サイクルを示させます。この動作によって、メモリシステムの他の部分は ARM720T コアを無視し、通常通り機能することができます。メモリシステムの他の部分は動作を継続することから、ARM720T は強制的にアボートと割り込みを無視するようになります。

警告： デバッギング中はコアをリセットしないでください。リセットすると、デバッガはコアを監視できなくなります。

注意： システムは、デバッグ中に **ETMBIGEND** 信号を変化させてはいけません。プログラマの視点から見ると、**ETMBIGEND** が変更された場合、ARM720T プロセッサが変化し、デバッガはコアがリセットされていることを認識できません。また、デバッグ中 **HRESETn** を安定させておいてください。システムが ARM720T プロセッサにリセットを適用すると（すなわち、**HRESETn** は Low になります）、ARM720T プロセッサ状態が変わり、デバッガはコアがリセットされていることを認識できません。

9.3.5 クロック

システムクロックとテストクロックは、プロセッサの外部で同期させる必要があります。ARM Multi-ICE デバッグエージェントは、ASIC 設計内の 1 つ以上のコアを直接サポートしています。オフチップのデバッグクロックを ARM720T プロセッサと同期させるには、3 段シンクロナイザが必要になります。オフチップデバイス（たとえば、Multi-ICE）は **TCK** 信号を出して、**RTCK**（Returned **TCK**）信号が返ってくるのを待ちます。オフチップデバイスは **RTCK** を受信するまで次の **TCK** には進まないため、同期化は維持されます。

図 9-4 にこの同期化を示します。

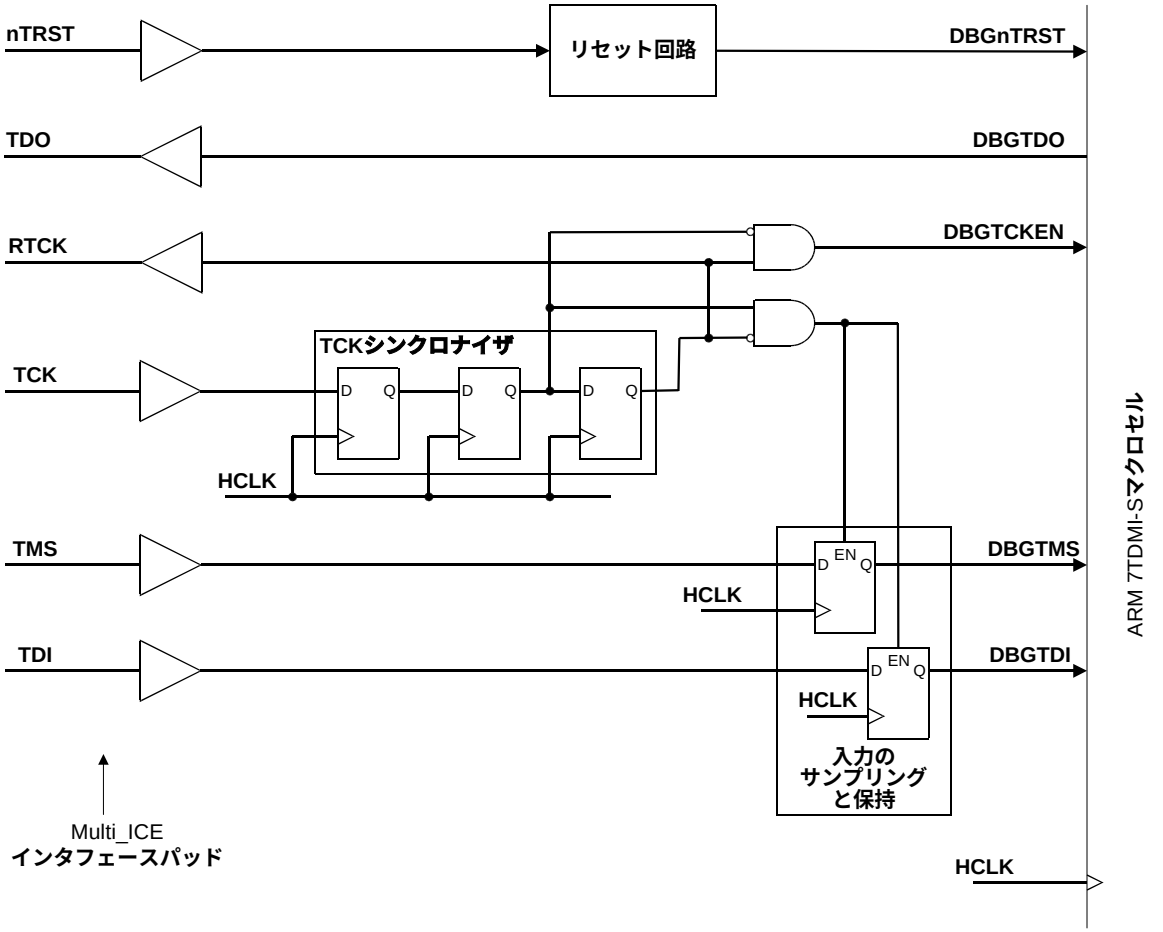


図 9-4 クロックの同期化

注意： 図 9-4 に示す D タイプはすべて **DBGnTRST** によってリセットされます。

9.4 デバッグインタフェース

ARM720Tプロセッサデバッグインタフェースは、IEEE規格 1149.1-1990 *"Standard Test Access Port and Boundary-Scan Architecture"* に基づいています。この章で使用する用語や TAP コントローラの状態の説明については、この規格を参照してください。

9.4.1 デバッグインタフェース信号

デバッグインタフェースと関連のある外部信号には、主に以下の 3 つの外部信号があります。

- **DBGBREAK** および **DBGRQ** は、ARM720T コアがデバッグ状態になるためのシステム要求です。

注意： コアがデバッグ状態になったときは、**DBGRQ** と **DBGBREAK** はともに Low になっていなければなりません。そうでないと、これらの信号はコアがデバッグに入ったり、抜けたりする方法を制御するスキャンチェイン 1 での **DBGBREAK** フラグの使用に影響を与えます。その結果、コアは予測できない一連のデバッグとシステムスピードアクセスを行い、デバッガはコアを制御できなくなります。

- **DBGACK** は、デバッグ状態になっていることをシステムにフラグバックするために ARM720T コアによって使用されます。

9.5 ARM720T コアクロックドメイン

ARM720T プロセッサは、以下の 2 つのクロックイネーブルで制御される単一のクロック **HCLK** をもっています。

- **HCLKEN** は、メモリシステムへのアクセスを制御します。
- **DBGTCKEN** は、デバッグ動作を制御します。

ARM720T プロセッサがデバッグ状態のとき、**DBGTCKEN** はコアのクロックである **HCLK** を調整します。

9.6 EmbeddedICE-RT マクロセル

ARM720T プロセッサの EmbeddedICE-RT マクロセルモジュールは、ARM720T コアに対して内蔵のデバッグサポートを提供します。

EmbeddedICE-RT モジュールは直接コアに接続されますので、FCSE PID による再配置の前にプロセッサの仮想アドレスで機能します。EmbeddedICE-RT マクロセルについては、ARM720T プロセッサ TAP コントローラを使用してシリアルにプログラムします。

図 9-5 はコア、EmbeddedICE-RT、および TAP コントローラ間の関係を示しており、EmbeddedICE-RT と関係のある信号のみを記載しています。

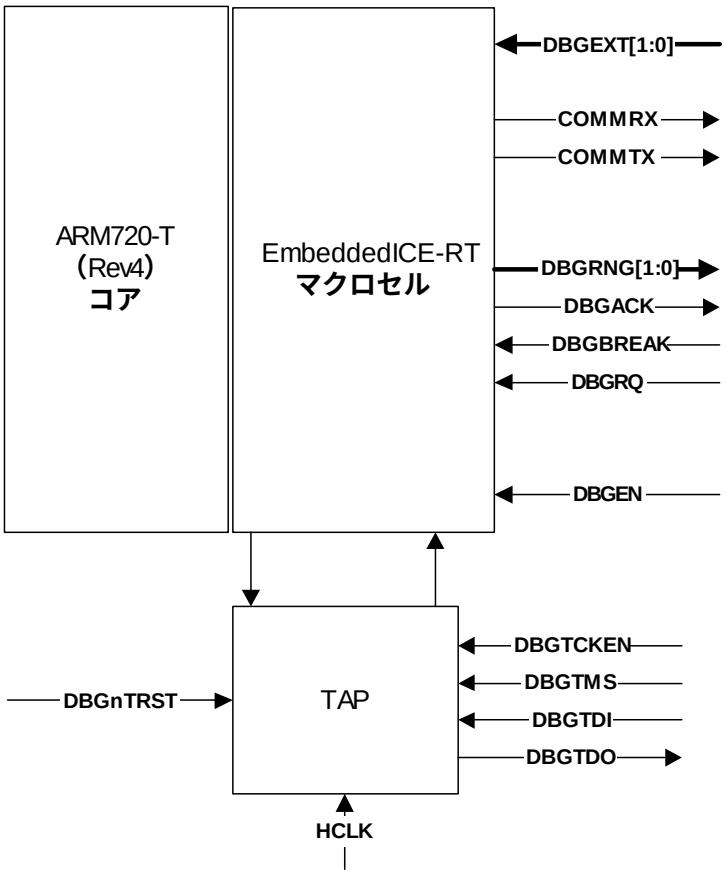


図 9-5 ARM720T コア、TAP コントローラ、および EmbeddedICE-RT マクロセルの関係

EmbeddedICE-RT ロジックは、以下の構成品から成ります。

2 つのリアルタイムウォッチポイントユニット

1 つまたは両方のウォッチポイントユニットを設定して、コアによる命令の実行を停止させることができます。命令の実行は、EmbeddedICE-RT ロジックに設定された値が現在アドレスバス、データバス、および各種の制御信号で現れている値と一致したときに停止します。ビットの値が比較に影響を与えないようにマスクすることができます。

各ウォッチポイントユニットをウォッチポイント（データアクセスの監視）か、あるいはブレークポイント（命令フェッチの監視）のどちらにも設定することができます。ウォッチポイントやブレークポイントは、データに依存させることができます。

詳細については、*「ウォッチポイントユニットレジスタ」*の項（9-34 ページ）を参照してください。

アボートステータスレジスタ

このレジスタは、アボート例外がブレークポイント、ウォッチポイント、あるいは真のアボートによって引き起こされたかどうかを明らかにします。詳細については、*「アボートステータスレジスタ」*の項（9-40 ページ）を参照してください。

デバッグ通信チャネル（DCC)

DCC は、ターゲットとホストデバッガ間で情報を受け渡します。詳細については、*「デバッグ通信チャネル」*の項（9-14 ページ）を参照してください。

さらに、2 つの独立レジスタによって EmbeddedICE-RT 操作を包括的に制御することができます。これについては、以下の項で説明しています。

- *デバッグ制御レジスタ*、9-40 ページ
- *デバッグステータスレジスタ*、9-42 ページ .

EmbeddedICE-RT レジスタの位置を、*「EmbeddedICE-RT タイミング」*の項（9-45 ページ）に示します。

9.7 EmbeddedICE-RT のディセーブル

EmbeddedICE-RT を次の 2 つの方法でディセーブルにすることができます。

恒久的

DBGEN 入力を Low に配線することによって

DBGEN が Low の場合、次のようになります。

- **DBGBREAK** および **DBGRQ** はコアによって無視されます。

注意： **DBGACK** は、ARM720T コアによって強制的に Low にされます。

- 割り込みは、禁止されていないプロセッサへ通過します。
- EmbeddedICE-RT ロジックは低消費電力モードに入ります。

警告： **DBGEN** 入力を恒久的に Low にハードウェアにより固定すると、デバッグ状態情報はディセーブルされます。ただし、システムセキュリティを確保するためには、これに依存しないでください。

一時的

デバッグ制御レジスタ（*「デバッグ制御レジスタ」*の項（9-40 ページ）に説明しています）のビット 5 を設定することによって。ビット 5 は、EmbeddedICE-RT ディセーブルビットとも呼ばれます。

以下のいずれかの操作を行う前に、ビット 5 を設定してください。

- ブレークポイントやウォッチポイントレジスタを設定する。
- デバッグ制御レジスタのビット 4 を変更する。

9.8 EmbeddedICE-RT レジスタマップ

EmbeddedICE-RT レジスタについて表 9-1 に示します。

表 9-1 EmbeddedICE-RT レジスタの機能とマッピング

アドレス	幅	機能
b00000	6	デバッグ制御
b00001	5	デバッグステータス
b00100	32	デバッグ通信チャネル（DCC）制御レジスタ
b00101	32	デバッグ通信チャネル（DCC）データレジスタ
b01000	32	ウォッチポイント 0 アドレス値
b01001	32	ウォッチポイント 0 アドレスマスク
b01010	32	ウォッチポイント 0 データ値
b01011	32	ウォッチポイント 0 データマスク
b01100	9	ウォッチポイント 0 制御値
b01101	8	ウォッチポイント 0 制御マスク
b10000	32	ウォッチポイント 1 アドレス値
b10001	32	ウォッチポイント 1 アドレスマスク
b10010	32	ウォッチポイント 1 データ値
b10011	32	ウォッチポイント 1 データマスク
b10100	9	ウォッチポイント 1 制御値
b10101	8	ウォッチポイント 1 制御マスク

9.9 モニタモードデバッギング

ARM720T プロセッサは、コア全体を停止させずにシステムのデバッグをイネーブルにすることができるロジックを備えています。これは、コアがデバッガによって問い合わせを受けている間に重要な割り込みルーチンが処理し続けることを意味しています。

9.9.1 モニタモードのイネーブル

デバッギングモードは、デバッグ制御レジスタ（*「デバッグ制御レジスタ」*の項（9-40 ページ）に説明しています）のビット 4 で制御されます。このレジスタのビット 4 は、モニタモードイネーブルビットとも呼ばれます。

ビット 4 セット　ARM720T プロセッサのモニタモード機能をイネーブルにします。このビットがセットされると、EmbeddedICE-RT ロジックはブレークポイントまたはウォッチポイントによって ARM720T コアがアボートモードになり、それぞれプリフェッチアボートベクタやデータアボートベクタを処理するように構成されます。

ビット 4 クリア　モニタモードデバッギングはディセーブルされ、システムはホルトモードになります。ホルトモードでは、コアはブレークポイントまたはウォッチポイントを検出するとデバッグ状態に入ります。

9.9.2 モニタモードデバッギングの制限事項

ARM コアをモニタモードデバッギングに構成する場合、以下のような知っておくべき制限事項があります。

- ブレークポイントやウォッチポイントは、モニタモードではデータに依存するようには設定できません。レンジ機能の使用はサポートされていません。ブレークポイントとウォッチポイントは、以下の項目にのみ基づくことができます。
  - 命令またはデータアドレス
  - 外部ウォッチポイントコンディショナ（**DBGEXT[0]** または **DBGEXT[1]**）
  - ユーザモードまたは特権モードアクセス（**CPnTRANS**）
  - ウォッチポイントのリード／ライトアクセス（**HWRITE**）
  - アクセスサイズ（ウォッチポイント **SIZE[1:0]**）
- 外部ブレークポイントやウォッチポイントはサポートされていません。
- ホルトモードとモニタモード機能の混合はサポートされていません。

モニタモードによってアボートが発生した場合は、コプロセッサ 14 のアボートステータスレジスタに記録されます（*「アボートステータスレジスタ」*の項（9-40 ページ）を参照）。

モニタモードイネーブルビットによって、ARM720T プロセッサがデバッグ状態になることはありません。このため、ウォッチポイントレジスタの内容は、コアが停止しているデバッグ状態のときではなく、外部メモリアクセスが実行されている間に変更する必要があります。

ウォッチポイントレジスタの変更時に（一部のレジスタの古いデータと残りのレジスタの新しいデータによって）誤った一致が生じる恐れがある場合は、以下の手順をとってください。

1. デバッグ制御レジスタのビット 5（EmbeddedICE-RT ディセーブルビットとも呼ばれます）をセットすることによってウォッチポイントユニットをディセーブルにします。
2. EmbeddedICE-RT ディセーブルビットがセットどおりに読み返されるまで、デバッグ制御レジスタをポーリングします。
3. その他のレジスタを変更します。
4. デバッグ制御レジスタのEmbeddedICE-RTディセーブルビットをクリアして、ウォッチポイントユニットを再度イネーブルにします。

ブレークポイントとウォッチポイントでのコア動作の制御に関する詳細については、*「デバッグ制御レジスタ」*の項（9-40 ページ）を参照してください。

9.10 デバッグ通信チャネル

ARM720T EmbeddedICE-RT マクロセルには、ターゲットとホストデバッガ間で情報をやり取りするための*デバッグ通信チャネル*（DCC）があります。これは、コプロセッサ 14 としてインプリメントされています。

DCC は、以下の 2 つのレジスタから構成されます。

DCC 制御レジスタ

プロセッサと非同期デバッガ間の同期化ハンドシェイキングに使用される 32 ビットレジスタです。詳細については、*「ドメインアクセス制御レジスタ」*の項を参照してください。

DCC データレジスタ

デバッガとプロセッサ間のデータ転送に使用される 32 ビットレジスタです。詳細については、*「DCC による通信」*の項（9-16 ページ）を参照してください。

これらのレジスタは、表 9-1（9-12 ページ）に示すように EmbeddedICE-RT メモリマップの固定位置を占有しています。これらは、コプロセッサ 14 への MCR 命令、MRC 命令を使用してプロセッサからアクセスされます。

レジスタは次のようにアクセスされます。

デバッガにより　通常の方法でスキャンチェイン 2 を介して

プロセッサにより　コプロセッサレジスタ転送命令を介して

9.10.1 ドメインアクセス制御レジスタ

ドメインアクセス制御レジスタは読み出し専用であり、プロセッサとデバッガ間で同期化ハンドシェイキングが行えるようにしています。そのレジスタ形式を図 9-6 に示します。

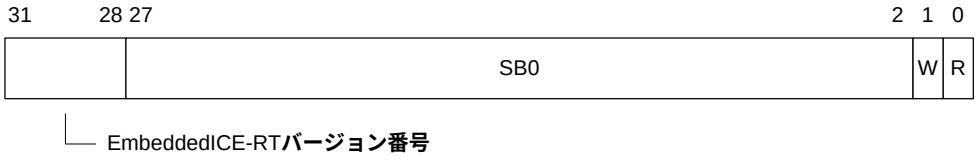


図 9-6 ドメインアクセス制御レジスタ

ドメインアクセス制御レジスタのビット割当を表 9-2 に示します。

表 9-2 ドメインアクセス制御レジスタのビット割当

ビット	機能
31:28	EmbeddedICE-RT バージョン番号を示した固定パターンを格納しています。これは以下の通りでなければなりません。 • MRC 動作を使用して読み出す場合は、b0111 • スキャン動作を使用して読み出す場合は、b0100
27:2	SBZ
1	ライト制御ビット このビットがクリアされると、DCC データライトレジスタはいつでもプロセッサからデータを受け付けることができます。 このビットがセットされると、DCC データライトレジスタにデータが存在し、デバッガはそれをスキャンアウトすることができます。
0	リード制御ビット このビットがクリアされると、DCC データリードレジスタはいつでもデバッガからデータを受け付けることができます。 このビットがセットされると、DCC データリードレジスタはまだプロセッサが読み出していない新しいデータを格納していますので、デバッガは待機しなければなりません。

注意： 実行が停止すると、ビット 0 はアサートされたままになる場合があります。デバッガはドメインアクセス制御レジスタに書き込みを行ってこれをクリアすることができます。

このレジスタへの書き込みはほとんど必要ありません。通常の動作ではプロセッサはレジスタの読み出し後、ビット 0 をクリアするからです。

命令

以下の命令を使用してください。

MRC CP14, 0, <Rd>, C0, C0

ドメインアクセス制御レジスタからの値をデスティネーションレジスタ Rd に返します。

MCR CP14, 0, <Rn>, C1, C0

ソースレジスタ Rn の値を DCC データライトレジスタに書き込みます。

MRC CP14, 0, <Rd>, C1, C0

DCC データリードレジスタからの値をデスティネーションレジスタ Rd に返します。

注意： Thumb 命令セットはコプロセッサ命令を含んでいないため、Thumb 状態のときに SWI 命令を使用して、アクセスすることを推奨します。

9.10.2 DCC による通信

メッセージは DCC を介して送受信することができます。

デバッガにメッセージを送信する

メッセージは、以下のようにプロセッサからデバッガへ送信されます。

1 プロセッサが EmbeddedICE-RT にメッセージを送りたい時には、まず、通信データライトレジスタが使用できるかどうかを確認します。プロセッサは、ドメインアクセス制御レジスタの W ビットの状態を確認することでこれを実施します。

   a. W ビットがクリアされていると、DCC データライトレジスタは空なので、メッセージはコプロセッサへのレジスタ転送によってライトされます。

   b. Wビットがセットされていると、前回書き込まれたデータがまだデバッガでリードされていないことを示しています。プロセッサは、W ビットがクリアされるまで繰り返しドメインアクセス制御レジスタをリードしなければなりません。

2 W ビットがクリアされると、メッセージはコプロセッサ 14 へのレジスタ転送によってライトされます。データ転送はプロセッサから DCC データライトレジスタへ行われますので、ドメインアクセス制御レジスタの W ビットはセットされます。

3 デバッガが JTAG インタフェースを介してドメインアクセス制御レジスタをリードすると、R ビットと W ビットの同期化バージョンを確認します。

   a. デバッガは W ビットがセットされていることを確認すると、通信データライトレジスタをリードし、データをスキャンアウトします。

   b. このデータレジスタのリード動作によって、ドメインアクセス制御レジスタの W ビットがクリアされます。この時点で、通信プロセスを再開することができます。

デバッガからメッセージを受信する

デバッガからプロセッサへのメッセージの転送は、メッセージをプロセッサからデバッガへ送信するのと似ています。この場合、デバッガはデバッグ通信制御レジスタの R ビットをリードします。

デバッガからメッセージを受信するシーケンスは、以下の通りです。

1 デバッガは、ドメインアクセス制御レジスタの R ビットをリードします。

   a. R ビットがクリアされていると、データリードレジスタは解放されているため、データをレジスタに置いてプロセッサがリードするようにすることができます。

   b. R ビットがセットされていると、前回置かれたデータはまだ収集されていないため、デバッガは待機しなければなりません。

2 通信データリードレジスタが解放されている場合は、JTAG インタフェースを使用してデータがこのレジスタへライトされます。このライト動作によって、ドメインアクセス制御レジスタの R ビットがセットされます。

3 プロセッサは、ドメインアクセス制御レジスタをリードします。

   a. R ビットがセットされていると、コプロセッサ 14 への MRC 命令を使用してリードすることができるデータが存在しています。このロード動作によって、デバッグ通信制御レジスタの R ビットはクリアされます。

   b. R ビットがクリアされていると、データはリードされているため、プロセスを繰り返すことができることを示しています。

9.11 スキャンチェインと JTAG インタフェース

ARM720T プロセッサ内には、3 つの JTAG 方式のスキャンチェインがあります。これらのスキャンチェインによって、デバッギングや EmbeddedICE-RT プログラミングが行えます。

JTAG 方式の*テストアクセスポート*（TAP）コントローラは、スキャンチェインを制御します。JTAG 仕様の詳細については、IEEE 規格 1149.1-1990 *"Standard Test Access Port and Boundary-Scan Architecture"*を参照してください。

9.11.1 スキャンチェインのインプリメンテーション

ARM720T プロセッサの 3 つのスキャンパスは、スキャンチェイン 1、スキャンチェイン 2、スキャンチェイン 15 と呼ばれます。これらのスキャンチェインを図 9-7 に示します。

デバッグスキャンチェイン 0 は ARM720T プロセッサではインプリメントされていませんが、制御信号はすべてマクロセルバウンダリで提供されています。これによって、必要に応じて独自のバウンダリスキャンチェインラッパを設計することができます。

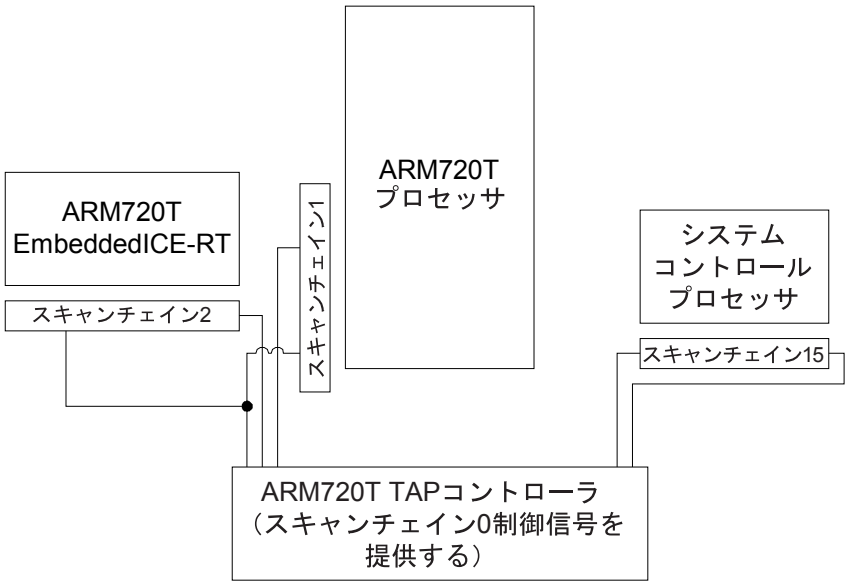


図 9-7 ARM720T プロセッサスキャンチェインの配置

スキャンチェイン 1

スキャンチェイン 1 は、コアデータバス **HRDATA/HWDATA** へのシリアルアクセスおよび **DBGBREAK** 信号を提供します。

このスキャンチェインには 33 ビットあり、順序（シリアルデータインからアウトまで）は以下の通りです。

- データバスビット 0 ～ 31
- **DBGBREAK** ビット（シフトアウトされる先頭ビット）

スキャンチェイン 2

スキャンチェイン 2 によって、EmbeddedICE-RT レジスタをアクセスすることができます。詳細については、*「テストデータレジスタ」*の項（9-22 ページ）を参照してください。

スキャンチェイン 15

スキャンチェイン 15 は、システム制御コプロセッサレジスタ（CP15 レジスタ）専用です。

スキャンチェイン 15 には 37 ビットあり、**DBGTDI** から **DBGTDO** までのビットの順序は以下の通りです。

- リード／ライトビット
- 命令エンコーディングビット [3:0]（表 9-3 を参照）
- データバスビット 31 ～ 0

データフィールドのビット 0 は、スキャンインおよびスキャンアウトされる先頭ビットです。

スキャンチェイン 15 の 4 ビット命令エンコーディングを表 9-3 に示します。

表 9-3 スキャンチェイン 15 の命令エンコーディング

エンコーディング	命令
b0000	ID レジスタアクセス（リードオンリ）
b0001	制御レジスタアクセス（リード／ライト）
b0010	変換テーブルベースレジスタアクセス(リード／ライト)
b0011	DAC レジスタアクセス（リード／ライト）
b0100	FSR レジスタアクセス（リード／ライト）
b0101	FAR レジスタアクセス（リード／ライト）
b0110	FCSE PID レジスタアクセス（リード／ライト）
b0111	TRACE PROCID レジスタアクセス（リード／ライト）
b1000	キャッシュの無効化（ライトオンリ）
b1001	TLB の無効化（ライトオンリ）
b1010	TLB シングルエントリの無効化（ライトオンリ）

注意： 表 9-3 に示す命令は、更新時にのみ実行することができます。リードを実行するには、プロセッサは状態をキャプチャするために戻り、その後結果をシフトアウトしなければなりません。このキャプチャ段階では、スキャンチェイン 15 の命令フィールドはゼロとしてリードされます。

9.11.2 JTAG インタフェースの制御

JTAG インタフェースは、命令レジスタ（*「命令レジスタ」*の項（9-23 ページ）に解説します）に現在ロードされている命令で駆動されます。命令のローディングは、*テストアクセスポート*（TAP）コントローラで制御されます。

TAP コントローラの詳細については、*「TAP コントローラ」*の項（9-19 ページ）を参照してください。

9.12 TAP コントローラ

TAP コントローラは、バウンダリスキャンテスト信号 **DBGTDI** と **DBGTDO** の状態を決定するステートマシンです。図 9-8 は、TAP コントローラで起こる状態遷移を示しています。

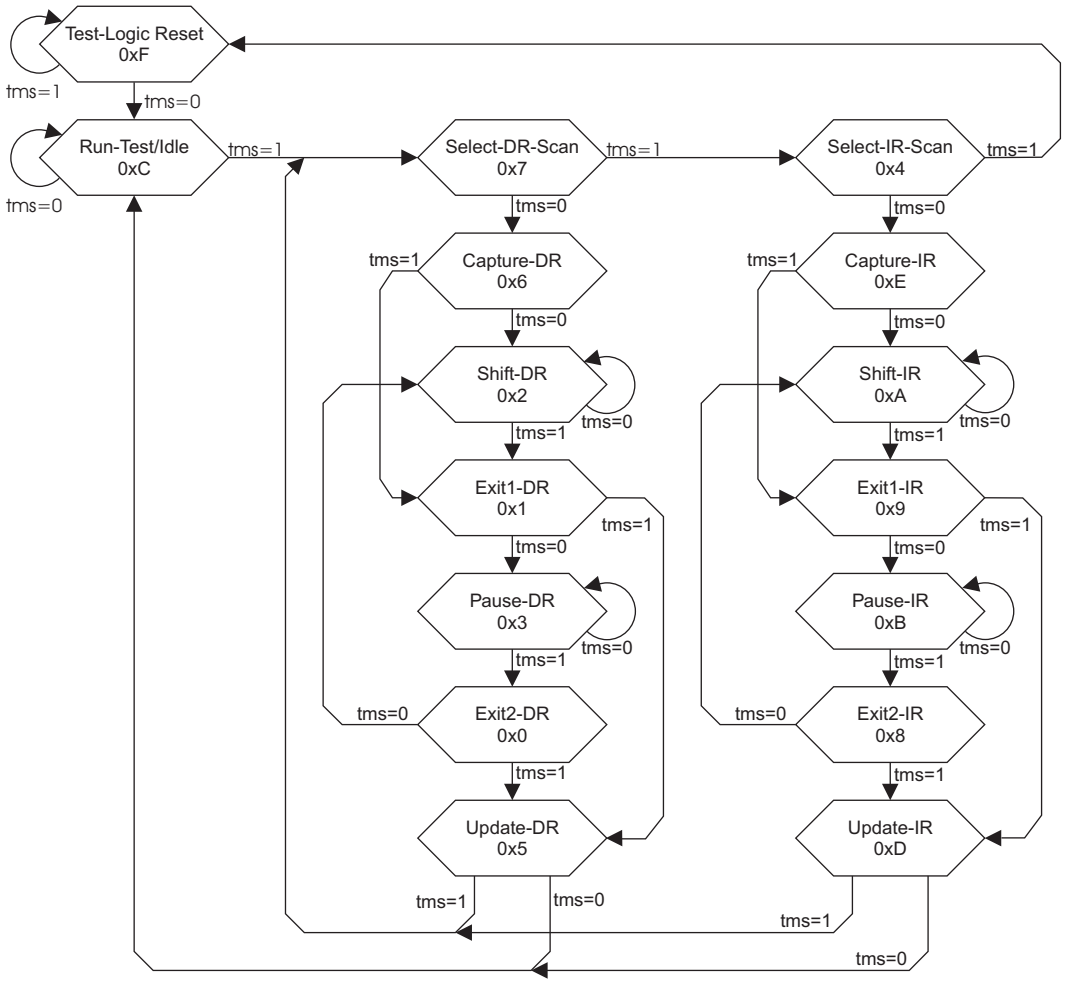


図 9-8 テストアクセスポートコントローラの状態遷移

IEEE 規格 1149.1-1990 より。禁転載。

9.12.1 TAP コントローラのリセット

電源投入後、強制的に TAP コントローラを適切な状態にするには、**DBGnTRST** 信号にリセットパルスを加えてください。

- バウンダリスキャンインタフェースを使用する場合は、**DBGnTRST** を Low にして、その後再度 High にしてください。
- バウンダリスキャンインタフェースを使用しない場合は、**DBGnTRST** 入力を Low に固定することができます。

リセット動作は以下の通りです。

1. システムモードが選択されます。これは、バウンダリスキャンセルが外部システムとコア間で受け渡しされるどの信号も妨げないことを意味しています。
2. IDCODE 命令が選択されます。

   TAP コントローラが SHIFT-DR 状態におかれ、**HCLK** が **DBGTCKEN** によってイネーブルされている間にパルス出力されると、ID レジスタの内容は **DBGTDO** からクロックアウトされます。

9.13 パブリック JTAG 命令

表 9-4 は、パブリック JTAG 命令を示しています。

表 9-4 パブリック命令

命令	バイナリコード
SCAN_N	b0010
INTEST	b1100
IDCODE	b1110
BYPASS	b1111
RESTART	b0100

以降の説明では、ARM720T プロセッサは **DBGTCKEN** を High にした状態で **DBGTDI** と **DBGTMS**を**HCLK**の立ち上がりエッジでサンプリングします。TAPコントローラの状態を図 9-8（9-19 ページ）に示します。

9.13.1 SCAN_N (b0010)

SCAN_N 命令は、**DBGTDI** と **DBGTDO** 間にスキャンパス選択レジスタを接続します。

- CAPTURE-DR ステートでは、固定値 b1000 はレジスタにロードされます。
- SHIFT-DR ステートでは、希望するスキャンパスの ID 番号はスキャンパス選択レジスタにシフトインされます。
- UPDATE-DR ステートでは、選択されたスキャンチェインのスキャンレジスタは **DBGTDI** と **DBGTDO** 間に接続され、次の SCAN_N 命令が発行されるまで接続されたままになっています。
- リセット時、スキャンチェイン 0 はデフォルトで選択されます。

スキャンパス選択レジスタはこのインプリメンテーションでは 4 ビット長になりますが、レジスタ長は特に限定されていません。

9.13.2 INTEST (b1100)

INTEST 命令は、選択したスキャンチェインをテストモードにします。

- INTEST命令は、**DBGTDI**と**DBGTDO**間に選択したスキャンチェインを接続します。
- INTEST 命令が命令レジスタにロードされると、すべてのスキャンセルは動作テストモードに入ります。
- CAPTURE-DR ステートでは、コアロジックから出力スキャンセルに適用されるデータの値、およびシステムロジックから入力スキャンセルに適用されるデータの値がキャプチャされます。
- SHIFT-DR ステートでは、前回キャプチャされたテストデータは **DBGTDO** ピンを介してスキャンチェインからシフトアウトされ、新しいテストデータは **DBGTDI** ピンを介してシフトインされます。

コアのシングルステップ動作は、INTEST 命令を使用して可能になります。

9.13.3 IDCODE (b1110)

IDCODE 命令は、**DBGTDI** と **DBGTDO** 間にデバイス識別コードレジスタ（すなわち、ID レジスタ）を接続します。ID レジスタは 32 ビットレジスタであり、このレジスタにより TAP を介してコンポーネントの製造業者名、パーツ番号、バージョンをリードすることができます。ID レジスタ形式の詳細については、*「ARM720T プロセッサデバイス識別（ID）コードレジスタ」*の項（9-22 ページ）を参照してください。

IDCODE 命令が命令レジスタにロードされると、すべてのスキャンセルは通常（システム）動作モードに設定されます。

- CAPTURE-DR ステートでは、デバイス識別コードは ID レジスタでキャプチャされます。
- SHIFT-DR ステートでは、前回キャプチャされたデバイス識別コードは **DBGTDO** ピンを介して ID レジスタからシフトアウトされ、データは **DBGTDI** ピンを介して ID レジスタにシフトインされます。
- UPDATE-DR ステートでは、ID レジスタは影響を受けません。

9.13.4 BYPASS (b1111)

BYPASS 命令は、**DBGTDI** と **DBGTDO** 間に 1 ビットシフトレジスタ（バイパスレジスタ）を接続します。

BYPASS 命令が命令レジスタにロードされると、すべてのスキャンセルは通常（システム）動作モードに設定されます。BYPASS 命令は、システム端子には影響を与えません。

- CAPTURE-DR ステートでは、ロジック 0 がバイパスレジスタでキャプチャされます。
- SHIFT-DR ステートでは、テストデータは **DBGTDI** ピンを介してバイパスレジスタにシフトインされ、**HCLK** の 1 サイクル遅延後に **DBGTDO** でシフトアウトされます。シフトアウトされる先頭ビットはゼロです。
- バイパスレジスタは、UPDATE-DR ステートでは影響を受けません。

未使用命令コードはすべて BYPASS 命令にデフォルト設定されます。

9.13.5 RESTART (b0100)

RESTART 命令は、デバッグ状態の終了時にプロセッサを再始動します。この命令は、**DBGTDI** と **DBGTDO** 間にバイパスレジスタを接続します。TAP コントローラは、あたかも BYPASS 命令がロードされたかのように動作します。

RUN-TEST/IDLE ステートになると、プロセッサはデバッグ状態を終了します。

詳細については、*「デバッグ状態の終了」*の項（9-29 ページ）を参照してください。

9.14 テストデータレジスタ

以降の項では、**DBGTDI** と **DBGTDO** 間に接続することができる 6 個のテストデータレジスタを説明します。

- *バイパスレジスタ*
- *ARM720T プロセッサデバイス識別（ID）コードレジスタ*
- *命令レジスタ*、9-23 ページ
- *スキャンパス選択レジスタ*、9-23 ページ
- *スキャンチェイン 1*、9-17 ページ
- *スキャンチェイン 2*、9-17 ページ

以降の説明では、**DBGTCKEN** イネーブルが High になっている場合、データは各 **HCLK** サイクル時にシフトされます。

9.14.1 バイパスレジスタ

目的 **DBGTDI** と **DBGTDO** 間にパスを確保して、スキャンテスト時にデバイスをバイパスします。

長さ 1 ビット

動作モード BYPASS 命令が命令レジスタ内の現在の命令の場合、シリアルデータは **DBGTCKEN** でイネーブルにされる **HCLK** の 1 サイクル遅延で SHIFT-DR ステートの **DBGTDI** から **DBGTDO** まで転送されます。
バイパスレジスタからパラレル出力は行われません。
ロジック 0 は、CAPTURE-DR ステートでバイパスレジスタのパラレル入力からロードされます。

9.14.2 ARM720T プロセッサデバイス識別（ID）コードレジスタ

目的 32 ビットデバイス識別コードをリードします。プログラム可能な補足識別コードはありません。

長さ 32 ビット。ID コードレジスタの形式を図 9-9 に示します。

図 9-9 ID コードレジスタ形式

デフォルトデバイス識別コードは 0x7f1f0f0f です。

動作モード IDCODE 命令がカレントになっていると、ID レジスタが **DBGTDI** と **DBGTDO** 間のシリアルパスとして選択されます。

ID レジスタからパラレル出力は行われません。

32 ビットデバイス識別コードは、CAPTURE-DR ステート時にパラレル入力から ID レジスタにロードされます。

9.14.3 命令レジスタ

目的 現在の TAP 命令を変更します。

長さ ４ビット

動作モード SHIFT-IR ステートでは、命令レジスタが **DBGTDI** と **DBGTDO** 間のシリアルパスとして選択されます。

CAPTURE-IR ステート時、バイナリ値 0001 はこのレジスタにロードされます。この値は SHIFT-IR ステート時に（最下位ビットから先に）シフトアウトされ、新しい命令は（最下位ビットから先に）シフトインされます。

UPDATE-IR ステート時、命令レジスタの値は現在の命令になります。

リセット時、IDCODE は現在の命令になります。

パリティビットはありません。

9.14.4 スキャンパス選択レジスタ

目的 現在のアクティブスキャンチェインを変更します。

長さ ４ビット

動作モード SHIFT-DR ステートの現在の命令として、SCAN_N はスキャンパス選択レジスタを **DBGTDI** と **DBGTDO** 間のシリアルパスとして選択します。

CAPTURE-DR ステート時、バイナリ値 b1000 はこのレジスタにロードされます。この値は SHIFT-DR ステート時（最下位ビットから先に）ロードアウトされ、新しい値は（最下位ビットから先に）ロードインされます。UPDATE-DR ステート時、レジスタの値はスキャンチェインを選択して現在アクティブなスキャンチェインになります。INTEST などの追加命令はすべてこのスキャンチェインに適用されます。

現在選択されているスキャンチェインは、SCAN_N 命令が実行されている場合にのみ、あるいはリセット発生時に変更されます。リセット時、スキャンチェイン 0 はアクティブなスキャンチェインとして選択されます。

表 9-5 は、スキャンチェイン番号の割当を示しています。

表 9-5 スキャンチェイン番号の割当

スキャンチェイン番号	機能
0	（ユーザインプリメンテッド）
1	デバッグ
2	EmbeddedICE-RT プログラミング
3	予約[a]
4	予約[a]
8	予約[a]

a. 選択されると、予約スキャンチェインはゼロをスキャンアウトします。

9.14.5 スキャンチェイン 1、2

スキャンチェインを使用して、コアロジックおよびプログラミングのために EmbeddedICE-RT ハードウェアへシリアルアクセスを行うことができます。各スキャンチェインセルは単純で、シリアルレジスタとマルチプレクサから構成されます。

スキャンセルは、以下の 3 つの基本機能を実行します。

- キャプチャ
- シフト
- 更新

入力セルの場合、キャプチャ段階にはコアへのシステム入力の値をシリアルレジスタへコピーすることが含まれます。シフト時、この値はシリアルに出力されます。入力セルからコアに設定される値は、システム入力か、あるいはマルチプレクサ制御下でのパラレルレジスタの内容（UPDATE-DR ステート後にシフトレジスタからのロード）のいずれかです。

出力セルの場合、キャプチャにはコア出力の値をシリアルレジスタに格納することが含まれます。シフト時、この値は前と同じようにシリアルに出力されます。出力セルからシステムに設定される値は、コア出力か、あるいはシリアルレジスタの内容のいずれかです。

スキャンセルの制御信号はすべて TAP コントローラによって内部で生成されます。TAP コントローラの動作は、現在の命令と TAP ステートマシンの状態によって決定されます。

スキャンチェイン 1

目的
スキャンチェイン 1 は、デバッガと ARM720T コア間の通信に使用されます。このスキャンチェインは、データのリード／ライトをしたり、命令をパイプラインにスキャンインしたりするために使用されます。SCAN_N TAP 命令を使用すると、スキャンチェイン 1 を選択することができます。

長さ
33 ビット。32 ビットはデータ値用で、1 ビットは **DBGBREAK** コア入力時のスキャンセル用です。

スキャンチェインの順序
DBGTDI から **DBGTDO** まで、ARM720T プロセッサデータビットがビット 0 ～ 31 であり、33 番目のビットは **DBGBREAK** スキャンセル用です。

スキャンチェイン 1 のビット 33 は、以下の 3 つの目的を果たします。

- 通常の INTEST テスト条件下では、ビット 33 によって既知の値を **DBGBREAK** 入力にスキャンインすることができます。
- デバッギング時、33 番目のビットに置かれた値は ARM720T コアが命令の実行前にシステム速度と同期をとるかどうかを決定します。詳細については、*「システムスピードアクセス」*の項（9-32 ページ）を参照してください。
- ARM720T コアがデバッグ状態になった後、33 番目のビットが初めてキャプチャされ、スキャンアウトされると、その値はコアがブレークポイント（ビット 33 は Low）から、あるいはウォッチポイント（ビット 33 は High）からデバッグ状態になったかをデバッガに通知します。

スキャンチェイン 2

目的 スキャンチェイン 2 によって、EmbeddedICE-RT レジスタにアクセスすることができます。これを行うには、SCAN_N TAP コントローラ命令を使用してスキャンチェイン 2 を選択し、その後 TAP コントローラを INTEST モードにしなければなりません。

長さ 38 ビット

スキャンチェインの順序

DBGTDI から **DBGTDO** まで、リード／ライトビットとレジスタアドレスビットがビット 4 ～ 0 であり、データビットがビット 0 ～ 31 です。

CAPTURE-DR 時には、動作は行われません。

SHIFT-DR 時、データ値はシリアルレジスタにシフトインされます。ビット 32 ～ 36 はアクセスする EmbeddedICE-RT レジスタのアドレスを指定します。

UPDATE-DR 時、このレジスタはビット 37 の値（0= リード、1= ライト）によってリードされたり、ライトされたりします。詳細については、図 9-12（9-35 ページ）を参照してください。

9.15 スキャンタイミング

図 9-10 は、一般的なスキャンタイミング情報を示しています。

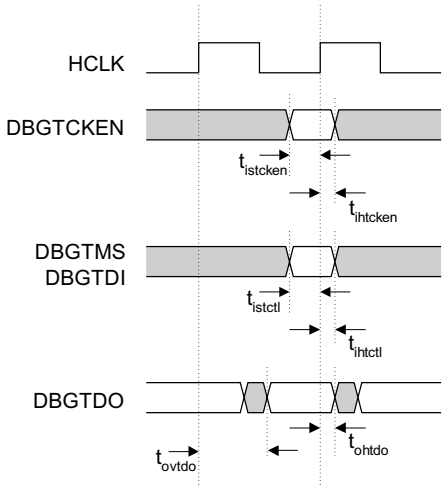


図 9-10 スキャンタイミング

9.15.1 スキャンチェイン 1 セル

ARM720T プロセッサは、表 9-6 に示すようにスキャンチェイン 1 セルにデータを提供します。

表 9-6 スキャンチェイン 1 セル

番号	信号	タイプ
1	DATA[0]	入出力
2	DATA[1]	入出力
3	DATA[2]	入出力
4	DATA[3]	入出力
5	DATA[4]	入出力
6	DATA[5]	入出力

表 9-6 スキャンチェイン 1 セル（続き）

番号	信号	タイプ
7	DATA[6]	入出力
8	DATA[7]	入出力
9	DATA[8]	入出力
10	DATA[9]	入出力
11	DATA[10]	入出力
12	DATA[11]	入出力
13	DATA[12]	入出力
14	DATA[13]	入出力
15	DATA[14]	入出力
16	DATA[15]	入出力
17	DATA[16]	入出力
18	DATA[17]	入出力
19	DATA[18]	入出力
20	DATA[19]	入出力
21	DATA[20]	入出力
22	DATA[21]	入出力
23	DATA[22]	入出力
24	DATA[23]	入出力
25	DATA[24]	入出力
26	DATA[25]	入出力
27	DATA[26]	入出力
28	DATA[27]	入出力
29	DATA[28]	入出力
30	DATA[29]	入出力
31	DATA[30]	入出力
32	DATA[31]	入出力
33	DBGBREAK	入力

9.16 デバッグ状態のコアおよびシステムの調査

ARM720T プロセッサがデバッグ状態にあるときは、ロードマルチプル、ストアマルチプルを強制的に命令パイプラインに入れてコアとシステム状態を調べることができます。

コアやシステム状態を調べられるようになる前に、デバッガは以下のように EmbeddedICE-RT デバッグステータスレジスタのビット 4 を調べて、プロセッサが Thumb 状態または ARM 状態のどちらからデバッグ状態になったのかを確認しなければなりません。

ビット 4 が High の場合　コアは Thumb 状態からデバッグ状態を開始。

ビット 4 が Low の場合　コアは ARM 状態からデバッグ状態を開始。

9.16.1 コア状態の確認

プロセッサが Thumb 状態からデバッグ状態になった場合、最も単純な処理手順はデバッガが強制的にコアを ARM 状態に戻すことです。その後、デバッガはプロセッサ状態を確認するために同じ命令シーケンスを実行できます。

プロセッサを強制的に ARM 状態に戻すには、コアで以下の Thumb 命令シーケンスを実行してください。

```
STR     R0, [R0]    ; Save R0 before use
MOV     R0, PC      ; Copy PC into R0
STR     R0, [R0]    ; Now save the PC in R0
BX      PC          ; Jump into ARM state
MOV     R8, R8      ; NOP
MOV     R8, R8      ; NOP
```

注意： すべての Thumb 命令は 16 ビット長のみですから、スキャンチェイン 1 をシフトするときに命令を繰り返すことができます。たとえば、BX R0 のエンコーディングは 0x4700 ですので、0x47004700 がスキャンチェイン 1 にシフトインされると、デバッガはプロセッサがデータをリードしようとしているバスの半分を追跡する必要がありません。

プロセッサの状態を確認するには、以下の ARM 命令のシーケンスを使用することができます。

プロセッサが ARM 状態にあるときに、最初に実行する命令は通常、以下の通りです。

```
STM R0, {r0-r15}
```

この命令によって、レジスタの内容がデータバス上で参照できるようになります。次にこれらの値をサンプリングし、シフトアウトさせることができます。

注意： 上記では r0 を STM のベースレジスタとして使用していますが、単なる一例にすぎず、実際にはどのレジスタを使用しても構いません。

現在のレジスタバンクの値を確認した後、バンクレジスタにアクセスしたい場合があります。これを行うには、モードを変更してください。通常、モード変更はコアがすでに特権モードになっているときにのみ行うことができます。ただし、デバッグ状態にあるときに、あるモードから別のモードにモード変更を行うことができます。

デバッガは、デバッグ状態を終了する前に元のモードを復元しなければなりません。たとえば、デバッガがユーザモードレジスタと FIQ モードレジスタの状態を戻すことを求められ、かつスーパーバイザモードでデバッグ状態に入っていた場合、命令シーケンスは以下のようにすることができます。

```
STM R0, {r0-r15}    ; Save current registers
MRS R0, CPSR
STR R0, R0          ; Save CPSR to determine current mode
BIC R0, 0x1F        ; Clear mode bits
ORR R0, 0x10        ; Select user mode
MSR CPSR, R0        ; Enter USER mode
STM R0, {r13,r14}   ; Save register not previously visible
ORR R0, 0x01        ; Select FIQ mode
MSR CPSR, R0        ; Enter FIQ mode
STM R0, {r8-r14}    ; Save banked FIQ registers
```

これらの命令はすべてデバッグスピードで実行されます。デバッグスピードはシステムスピードよりもかなり遅いですが、これは各コアクロック間で命令をシフトインしたり、データをシフトアウトしたりするのに 33 クロック発生するためです。ARM720T プロセッサは完全にスタティックですので、このようにゆっくりと命令を実行してもコア状態にアクセスするには問題はありません。ただし、システムの他の部分を確認するのにこの方法を使用することはできません。

デバッグ状態にある間、以下の命令のみを命令パイプラインにスキャンインして実行することができます。

- すべてのデータ処理操作
- すべてのロード、ストア、ロードマルチプル、およびストアマルチプル命令
- MSR と MRS

9.16.2 システム状態の確認

メモリシステムのダイナミックタイミング要件を満たすには、**HCLKEN** により制御されるクロックでシステム状態にアクセスしなければなりません。メモリアクセスを行うには、**HCLKEN** を使用して ARM720T プロセッサを通常の動作モードで強制的に実行させなければなりません。これは、スキャンチェイン 1 のビット 33 で制御されます。

ビット 33、すなわち **DBGBREAK** ビットを Low にしてスキャンチェイン 1 に設定された命令は、デバッグスピードで実行されます。この命令をシステムスピードで実行するためには、その前の命令をビット 33 を High にセットした状態でスキャンチェイン 1 にスキャンインしなければなりません。

システムスピード命令をデータバスにスキャンインし、パイプラインにクロックインした後、RESTART 命令を TAP コントローラにロードしなければなりません。RESTART によって、ARM720T プロセッサは以下の動作を実行します。

1. 自動的に **HCLKEN** 制御に切り替わる。
2. システムスピードで命令を実行する。
3. 再度デバッグ状態に入る。

命令が完了すると、**DBGACK** は High になり、コアは **DBGTCKEN** 制御に復帰します。これで、TAP コントローラで INTEST を選択して、デバッギングを再開することができます。

デバッガは、**DBGACK** および **HTRANS[1:0]** の両方を確認して、システムスピード命令が完了したかどうかを確認しなければなりません。メモリにアクセスするためには、ARM720T コアはシステムスピードに同期した後、**HTRANS[1:0]** の両ビットを Low にします。この遷移は、ARM720T コアが次のサイクルでバスを使用することができるかどうかを調停するためにメモリコントローラで使用されます。バスが利用できない場合、ARM720T プロセッサは無期限にそのクロックを機能停止させてしまうことがあります。メモリアクセスが完了したかどうかを確認する唯一の方法は、**HTRANS[1:0]** と **DBGACK** の両方の状態を調べることです。両方が High になっていると、アクセスは完了しています。

デバッガは通常 EmbeddedICE-RT を使用してデバッギングを制御し、**HTRANS[1:0]** と **DBGACK** の状態は EmbeddedICE-RT ステータスレジスタを読み出して確認することができます。詳細については、*「デバッグステータスレジスタ」*の項（9-42 ページ）を参照してください。

システムメモリの状態は、システムスピードのロードマルチプルとデバッグスピードのストアマルチプルを使用して、デバッグホストにフィードバックすることができます。

ビット 33 をセットできる命令は限られています。このビットをセットさせる有効な命令は、次の通りです。

- ロード
- ストア
- ロードマルチプル
- ストアマルチプル

*「デバッグ状態の終了」*の項（9-29 ページ）も参照してください。

システムスピードアクセス後に ARM720T プロセッサがデバッグ状態に戻ると、スキャンチェイン 1 のビット 33 は High にセットされます。ビット 33 の状態は、このスキャンチェインが初めて読み出されたときにコアがデバッグ状態になった原因についてのデバッグ情報を示します。

9.17 デバッグ状態の終了

デバッグ状態の終了には、以下の操作を伴います。

- ARM720T プロセッサの内部状態の復元
- 次の命令への分岐の実行
- 通常の動作への復帰

内部状態を復元後、分岐命令がパイプラインにロードされる必要があります。分岐の計算に関する詳細については、*「デバッグ時のプログラムカウンタ」*の項（9-31 ページ）を参照してください。

スキャンチェイン 1 のビット 33 は、ARM720T プロセッサを強制的に **HCLKEN**、すなわちクロックイネーブルに再同期させます。デバッグシーケンスの最後から二番目の命令は、ビット 33 が High になっているときにスキャンインされます。デバッグシーケンスの最後の命令は、ビット 33 が Low になっているときにスキャンインされる分岐です。そして、コアは分岐命令をパイプラインにロードするためにクロックを開始します。RESTART 命令が、TAP コントローラで選択されます。

ステートマシンが RUN-TEST/IDLE 状態になると、スキャンチェインはシステムモードに戻ります。そして ARM720T プロセッサは通常の動作を再開して、メモリから命令をフェッチします。この遅延により、すなわちステートマシンが RUN-TEST/IDLE 状態になるまでの間に、マイクロプロセッサシステムの他のデバイスが即時実施しないように条件を設定することができます。そして、ステートマシンが RUN-TEST/IDLE 状態になると、すべてのプロセッサは同時に動作を再開します。

ARM720T プロセッサがデバッグ状態になっていると、**DBGACK** はシステムの他の部分に通知します。この情報は、ウォッチドッグタイマなどの、リアルタイム特性をもつ周辺機器を抑止するために使用することができます。また **DBGACK** は、デバッギング処理によって生じたメモリアクセスをマスクすることもできます。

たとえば、ARM720T プロセッサがブレークポイント後にデバッグ状態になると、命令パイプラインはブレークポイントが設定された命令と、プリフェッチされた他の 2 つの命令を含んでいます。デバッグ状態の開始時にパイプラインはフラッシュされます。デバッグ状態終了時にはパイプラインは以前の状態に戻らなければなりません。

デバッギング処理のため、メモリアクセスは通常考えられるよりも多くなります。**DBGACK** は、メモリアクセス数の影響を受けやすいシステム周辺機器を抑止させることができます。たとえば、メモリサイクル数をカウントする周辺機器は、プログラムがデバッギング付きで実行された後も、デバッギングなしで実行された後も同じ応答を返さなければなりません。図 9-11 は、デバッグ状態終了時の ARM720T プロセッサの動作を示しています。

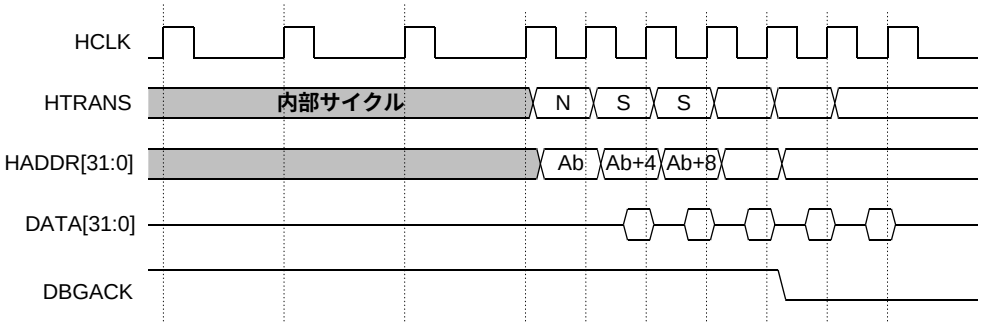


図 9-11 デバッグ終了シーケンス

図 9-3（9-5 ページ）は、**DBGACK** が High になった後に内部サイクルで最終メモリアクセスが発生することを示しています。ここで、サイクルカウンタがディセーブルされなければなりません。図 9-11 は、**DBGACK** が Low になった後、サイクルカウンタが以前確認していなかった初めてのメモリアクセスが内部サイクルに発生していることを示しています。ここでサイクルカウンタを再度イネーブルにします。

注意： デバッグ状態からのシステムスピードアクセスが発生すると、ARM720T プロセッサは一時的にデバッグ状態から抜け出すため、**DBGACK** は Low になります。メモリアクセス数に影響されやすい周辺機器がある場合、これらの周辺機器は ARM720T プロセッサがまだデバッグ状態にあることを認識しなければなりません。これは、EmbeddedICE-RT 制御レジスタを設定して、**DBGACK** の値を強制的に High にすることによって行うことができます。詳細については、*「デバッグステータスレジスタ」*の項（9-42 ページ）を参照してください。

9.18 デバッグ時のプログラムカウンタ

デバッガは PC の状態を追跡して、ARM720T コアがデバッグによってプログラムフローが中断された場所まで強制的に分岐して復帰します。プログラムフローは、以下の項目によっても中断されます。

- *ブレークポイント*
- *ウォッチポイント*
- *別の例外をもったウォッチポイント*、9-32 ページ
- *デバッグ要求*、9-32 ページ
- *システムスピードアクセス*、9-32 ページ

9.18.1 ブレークポイント

ブレークポイントからデバッグ状態になると、PC は 4 アドレスまたは 16 バイト加算されます。デバッグ状態で各命令が実行されると、PC は 1 アドレスまたは 4 バイト加算されます。

ブレークポイント後にデバッグ状態を終了するための通常の方法は、ブレークポイントを削除し、以前にブレークポイントが設定されていたアドレスに分岐して復帰することです。

たとえば、ARM720T プロセッサが特定のアドレスに設定されていたブレークポイントからデバッグ状態に入り、かつ 2 デバッグスピード命令が実行されていた場合、−7 アドレスの分岐を行う必要があります（デバッグに入るための 4、命令用の 2、および最後の分岐用の 1）。

以下のシーケンスは、最上位ビットから先にスキャンチェイン 1 にスキャンインされるデータを示しています。最初の桁の値は **DBGBREAK** ビットに入り、命令データはスキャンチェイン 1 の残りの部分に入ります。

```
0 E0802000   ; ADD r2, r0, r0

1 E1826001   ; ORR r6, r2, r1

0 EAFFFFF9  ; B -7 (2's complement)
```

ARM720T プロセッサがデバッグ状態になった後は、最低 2 つの命令を、両方とも NOP（MOV R0、R0）でも、分岐前に実行しなければなりません。小さい分岐の場合、PC をあて先（上記の例では SUB PC、PC、#28）にして、最後の分岐を減算で置き換えることができます。

9.18.2 ウォッチポイント

ウォッチポイントからデバッグ状態になった後にプログラムの実行に戻る方法は、*「ブレークポイント」*で説明した手順と同じように行います。

デバッグ状態に入ると、4 アドレスが PC に加算され、命令はすべて 1 アドレス加算されます。ブレークポイントとの違いは、ウォッチポイントを引き起こした命令が実行され、プログラムがその次の命令に戻らなければならないという点です。

9.18.3 別の例外をもったウォッチポイント

ウォッチポイントが設定されたアクセスが同時にデータアボートを引き起こした場合、ARM720T プロセッサはアボートモードでデバッグ状態になります。コアがアボートモードに変わり、アボートベクタから命令をフェッチするまでデバッグは開始されません。

ウォッチポイントが設定されたメモリアクセス時に割り込みや他の例外が発生すると、同様なシーケンスが行われます。ARM720T プロセッサは、例外のモードでデバッグ状態になります。デバッガは、（CPSR および SPSR の）現在および以前のモードならびに PC の値を調べることによって、例外が発生したかどうかをチェックしなければなりません。例外が実行されると、ユーザにはデバッギング前に例外を処理する選択肢が与えられます。

例外発生時にデバッグ状態になると、PC は 4 命令ではなく 3 命令分だけインクリメントされます。このことは、デバッグ状態を終了するときに戻り分岐計算で考慮しなければなりません。たとえば、ウォッチポイントが設定されたアクセスでアボートが発生し、10 命令が実行されてこの結果が確認されたとします。この場合、以下のシーケンスを使用してプログラムの実行に戻ることができます。

```
0 E1A00000   ; MOV R0, R0

1 E1A00000   ; MOV R0, R0

0 EAFFFFF0  ; B -16
```

このコードは強制的に分岐をアボートベクタに戻し、その場所にある命令が再度フェッチされ実行されます。

注意： アボートサービスルーチンの後、アボートとウォッチポイントを引き起こした命令は再度フェッチされ、実行されます。これによって、ウォッチポイントは再度トリガとなり、ARM720T プロセッサは再度デバッグ状態になります。

9.18.4 デバッグ要求

デバッグ要求を使用してデバッグ状態を開始するのは、ブレークポイントの場合と似ています。ただし、ブレークポイントとは異なり、最後の命令はその実行を完了しているため、デバッグ状態の終了時に再度フェッチされないようにします。したがって、デバッグ状態になることによって 3 アドレスが PC に加算され、デバッグ状態で命令が実行されるたびに 1 アドレスが加算されると考えることができます。

たとえば、デバッグ要求を呼び出し、すぐにプログラム実行への復帰を決めたと仮定すると、以下のシーケンスを使用することができます。

```
0 E1A00000   ; MOV R0, R0

1 E1A00000   ; MOV R0, R0

0 EAFFFFFA  ; B -6
```

このコードによって、PC は復元され、プログラムは次の命令から再開されます。

9.18.5 システムスピードアクセス

デバッグ状態でシステムスピードアクセスが実行されると、PC の値は 3 アドレスだけ増加します。システムスピード命令はメモリシステムにアクセスするため、アボートを実行させることができます。アボートがシステムスピードメモリアクセス時に発生すると、ARM720T プロセッサはデバッグ状態に戻る前にアボートモードに入ります。

この方法は、アボートされたウォッチポイントに似ていますが、アボートはメインプログラムの命令で引き起こされたわけではないので、PC はアボートの原因となった命令を示さないことから、問題の解決はさらに難しくなります。アボートハンドラは、通常 PC を調べてアボートを引き起こした命令およびアボートアドレスを確認します。この場合、PC の値は無効になります。しかし、デバッガはアクセスされていた場所を決めることができるため、デバッガはアボートハンドラがメモリシステムを修復できるように、手助けします。

9.18.6 復帰アドレス計算の要約

デバッグ状態になったことがブレークポイント、ウォッチポイント、あるいはデバッグ要求（**DBGRQ**）によるものかどうかを確認するために、デバッグステータスレジスタ（スキャンチェイン 2 のレジスタ 1）のビット 12（**DBGMOE**）とともに、スキャンチェイン 1 のビット 33（**DBGBREAK**）を調べる必要があります。

表 9-7 は、デバッグ状態になった原因によってどのように **DBGMOE** および **DBGBREAK** が変わるかを示しています。

注意： **DBGMOE** および **DBGBREAK** は、デバッグ状態に入った後、およびスキャンチェイン 1 への別のアクセス前に読み出されなければなりません。

表 9-7 デバッグ状態開始の原因の決定

DBGMOE	DBGBREAK	説明
0	0	ブレークポイント
0	1	ウォッチポイント
1	X	デバッグ要求（DBGRQ）

分岐リターンアドレスは、以下のように計算されます。

- 通常のブレークポイントとウォッチポイントの場合、分岐は以下の通りです。
  - (4 + N + 3S)
- デバッグ要求（**DBGRQ**）や、例外をもったウォッチポイントの場合、分岐は以下の通りです。
  - (3 + N + 3S)

ここで、N は実行されたデバッグスピード命令数（最後の分岐を含みます）であり、S は実行されたシステムスピード命令数です。

9.19 優先順位と例外

ブレークポイントやデバッグ要求が発生すると、通常のプログラムフローは中断します。したがって、デバッグは別のタイプの例外として処理することができます。デバッガと他の例外との相互作用については、*「デバッグ時のプログラムカウンタ」*の項（9-31 ページ）に説明しています。この項では、以下の優先順位を説明しています。

- *プリフェッチアボート付きブレークポイント*
- *割り込み*、9-34 ページ
- *データアボート*、9-34 ページ

9.19.1 プリフェッチアボート付きブレークポイント

ブレークポイントが設定された命令フェッチによってプリフェッチアボートが発生すると、アボートは実行され、ブレークポイントは無視されます。通常、プリフェッチアボートは、たとえば物理的に存在しない仮想アドレスに対してアクセスが行われ、その結果返されたデータが無効になると発生します。このような場合、オペレーティングシステムの通常の動作ではメモリページでスワップを行って、以前無効だったアドレスに戻ります。このとき、命令がフェッチされ、ブレークポイントの設定が有効となり（データに依存することがあります）、ARM720T プロセッサはデバッグ状態になります。

したがって、プリフェッチアボートはブレークポイントよりも優先順位が高くなっています。

9.19.2 割り込み

ARM720T プロセッサがデバッグ状態になると、割り込みは自動的にディセーブルされます。

割り込みが、デバッグ状態になる前の命令実行時に保留になっていると、ARM720T プロセッサは割り込みのモードでデバッグ状態になります。デバッグ状態になるとき、デバッガは ARM720T がユーザのプログラムが予想するモードにあることを想定することができません。ARM720T コアは、PC、CPSR、SPSR をチェックして、例外の原因を正確に確認しなければなりません。

このため、デバッグは割り込みよりも優先順位が高くなりますが、ARM720T プロセッサは割り込みが発生したことを記憶しています。

9.19.3 データアボート

ウォッチポイントが設定されたアクセスでデータアボートが発生すると、ARM720T プロセッサはアボートモードでデバッグ状態になります。したがって、ウォッチポイントはアボートよりも優先順位は高くなりますが、ARM720T プロセッサはアボートが発生したことを記憶しています。

9.20 ウォッチポイントユニットレジスタ

ウォッチポイントユニットには、*「ウォッチポイント 0」*と *「ウォッチポイント 1」*の 2 つがあります。これらのウォッチポイントユニットは、ウォッチポイント（データアクセスの監視）またはブレークポイント（命令フェッチの監視）のいずれかで構成することができます。ウォッチポイントとブレークポイントは、データ依存型にすることができます。

各ウォッチポイントユニットには、3 ペアのレジスタがあります。

- アドレス値とアドレスマスク
- データ値とデータマスク
- 制御値と制御マスク

各レジスタは個別に設定することができ、独自のアドレスをもちます。ウォッチポイントユニットレジスタの機能とマッピングを表 9-1（9-12 ページ）に示します。

9.20.1 ウォッチポイントレジスタの設定とリード

ウォッチポイントレジスタは、データを EmbeddedICE-RT スキャンチェイン（スキャンチェイン 2）にシフトインすることによって設定されます。スキャンチェインは、以下のものから構成される 38 ビットシフトレジスタです。

- 32 ビットデータフィールド
- 5 ビットアドレスフィールド
- リード／ライトビット

このセットアップを図 9-12 に示します。

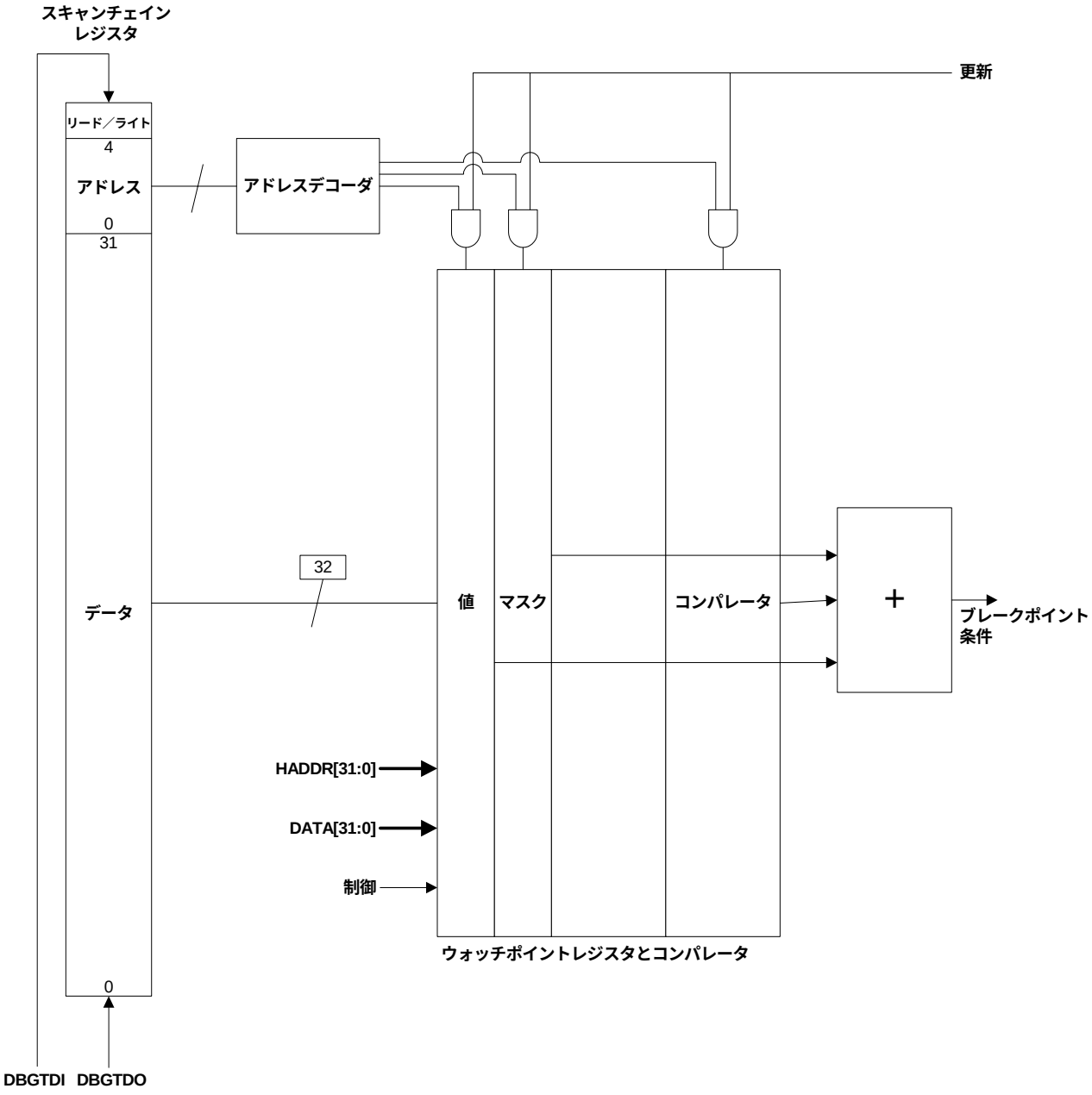


図 9-12 EmbeddedICE-RT **ブロック図**

ライトデータは 32 ビットデータフィールドにシフトインされ、レジスタのアドレスは 5 ビットアドレスフィールドにシフトインされ、そしてリード／ライトビットがセットされます。

ライトデータは 32 ビットデータフィールドにスキャンインされ、レジスタのアドレスは 5 ビットアドレスフィールドにスキャンインされ、そしてリード／ライトビットがセットされます。

レジスタは、そのアドレスをアドレスフィールドにシフトインし、そして 0 をリード／ライトビットにシフトインすることによってリードされます。32 ビットデータフィールドは無視されます。

レジスタアドレスを表 9-1（9-12 ページ）に示します。

注意： リードやライトは、TAP コントローラが UPDATE-DR ステートになると実行されます。

9.20.2 データおよびアドレスマスクレジスタの使用

一対のレジスタの各値レジスタには、同じ形式のマスクレジスタがあります。マスクレジスタでビットを 1 にセットすると、値レジスタの対応ビットを比較で無視させる効果があります。

たとえば、特定のメモリ位置でウォッチポイントが要求されるが、データ値が無関係な場合、データマスクレジスタを 0xFFFFFFFF（全ビットがセットされます）に設定してデータバスフィールド全体を無視することができます。

注意： マスクは、従来の AND マスクではなく XNOR マスクです。マスクビットが 1 にセットされると、そのビット位置に関してコンパレータは値レジスタや入力値にかかわりなく常に一致と判定します。

マスクビットのクリアは、入力値が、値レジスタに設定された値と一致する場合に限りコンパレータは一致と判定することを意味しています。

9.20.3 ウォッチポイントユニット制御レジスタ

制御値と制御マスクレジスタは、図 9-13に示すように下位8ビットに同じようにマップされます。

8	7	6	5	4	3	2	1	0
ENABLE	RANGE	CHAIN	DBGEXT	PROT[1]	PROT[0]	SIZE[1]	SIZE[0]	WRITE

図 9-13 ウォッチポイント制御値とマスク形式

制御値レジスタのビット 8 は ENABLE ビットであり、マスクすることはできません。

このレジスタのビットには、以下の機能があります。

WRITE バス動作の方向を検出するために、コアからのライト信号と比較されます。**WRITE** はリードサイクルでは “0” であり、ライトサイクルでは “1” です。

SIZE[1:0] バス動作のサイズを検出するために、コアからの **HSIZE[1:0]** 信号と比較されます。

エンコーディングを表 9-8 に示します。

表 9-8 SIZE[1:0] 信号のエンコーディング

bit 1	bit 0	データサイズ
0	0	バイト
0	1	ハーフワード
1	0	ワード
1	1	（予約）

PROT[0] 現在のサイクルが命令フェッチ（**PROT[0]** = 0）か、あるいはデータアクセス（**PROT[0]** = 1）かを検出するために使用されます。

PROT[1] ユーザモード（**PROT[1]** = 0）アクセスか、あるいは非ユーザモード（**PROT[1]** = 1）アクセスかを区別するために、コアからの未変換信号と比較する際に使用されます。

DBGEXT[1:0] ウォッチポイントを外部条件に依存させるようにすることができるEmbeddedICE-RT ロジックへの外部入力です。

ウォッチポイント 0 の **DBGEXT** 入力は、**DBGEXT[0]** と記されます。

ウォッチポイント 1 の **DBGEXT** 入力は、**DBGEXT[1]** と記されます。

CHAIN たとえば、breakpoint on address YYY only when in process XXX 形式のデバッガ要求をインプリメントするために、別のウォッチポイントのチェイン出力に接続することができます。

AMR720T プロセッサ EmbeddedICE-RT マクロセルでは、ウォッチポイント 1 の **CHAINOUT** 出力はウォッチポイント 0 の **CHAIN** 入力に接続されます。

CHAINOUT 出力は、レジスタから制御されます。アドレス／制御フィールドコンパレータはレジスタのライトイネーブルを駆動します。レジスタへの入力は、データフィールドコンパレータの値です。

CHAINOUT レジスタは、制御値レジスタにライトが行われたり、**DBGnTRST** が Low になっていたりするとクリアされます。

RANGE ARM720T プロセッサ EmbeddedICE-RT ロジックでは、ウォッチポイント 1 の **DBGRNG** 出力はウォッチポイント 0 の **RANGE** 入力に接続されます。この接続によって、2 つのウォッチポイントが連結され、たとえば、範囲チェックで同時に発生する状態を検出します。

ENABLE ウォッチポイントが一致していると、**ENABLE** ビットがセットされた場合に限り内部 **DBGBREAK** 信号がアサートされます。このビットは値レジスタでのみ存在します。マスクすることはできません。

制御値レジスタの各ビット [7:0] については、制御マスクレジスタに対応するビットがあります。これによって、特定の信号への依存性をなくしています。

9.21 ブレークポイントの設定

ブレークポイントは、ハードウェアブレークポイントとソフトウェアブレークポイントに分類されます。

- *ハードウェアブレークポイント*は一般的にアドレス値を監視し、ROM にあるコードやセルフモディファイイングコードなど、どんなコードででも設定することができます。詳細については、*「ハードウェアブレークポイント」*の項を参照してください。
- *ソフトウェアブレークポイント*は、任意のアドレスからフェッチされる特定のビットパターンを監視します。したがって、1 つの EmbeddedICE-RT ウォッチポイントが、ソフトウェアブレークポイントを何回もサポートするために使用することができます。詳細については、*「ソフトウェアブレークポイント」*の項（9-38 ページ）を参照してください。

  ソフトウェアブレークポイントを発生させるために選択された特殊なビットパターンは命令に置き換える必要があることから、通常、ソフトウェアブレークポイントは RAM 内にのみ設定されます。

9.21.1 ハードウェアブレークポイント

ウォッチポイントユニットに（命令フェッチで）ハードウェアブレークポイントを起こさせるようにするには、次の手順をとります。

1 ブレークポイントを設定する命令のアドレスでアドレス値レジスタを設定します。

2 以下のように、各状態のブレークポイントビットを設定します。

ARM 状態ブレークポイント

アドレスマスクレジスタのビット [1:0] をセットします。

Thumb 状態ブレークポイント

アドレスマスクレジスタのビット 0 をセットします。

どちらの場合も残りのビットはクリアしてください。

3 データ依存型ブレークポイントが必要な場合のみデータ値レジスタを設定します。すなわち、アドレスだけでなくフェッチされた実際の命令コードをマッチさせなければならない場合にのみ設定します。データ値が不要な場合は、データマスクレジスタを 0xFFFFFFFF に設定します（すべてのビットをセット）。そうでない場合は、このレジスタを 0x00000000 に設定してください。

4 制御値レジスタを **PROT[0]**=0 に設定します。

5 制御マスクレジスタを **PROT[0]**=0 に設定します。他のビットはすべてセットします。

6 ユーザモード命令フェッチと非ユーザモード命令フェッチを区別する必要がある場合は、**PROT[1]** 値を設定し、適宜ビットをマスクします。

7 必要に応じて、**DBGEXT**、**RANGE**、および **CHAIN** ビットを同じように設定します。

8 すべての未使用制御値のマスクビットをセットします。

注意： モニタモードでは、レジスタ値を変更する前に、EmbeddedICE-RT ディセーブルビット（デバッグ制御レジスタのビット 5）をセットし、プログラミング完了時それをクリアしてください。

9.21.2 ソフトウェアブレークポイント

ウォッチポイントユニットに（特定のビットパターンの命令フェッチで）ソフトウェアブレークポイントを起こさせるようにするには、次の手順をとります。

1 アドレスが無視されるように、ウォッチポイントユニットのアドレスマスクレジスタを 0xFFFFFFFF に設定します（すべてのビットをセット）。

2 ソフトウェアブレークポイントを表すために選択された特定のビットパターンでデータ値レジスタを設定します。

Thumb ソフトウェアブレークポイントを設定している場合、データ値レジスタの半分にそれぞれ16ビットパターンを繰り返します。たとえば、ビットパターンが0xDFFFならば、0xDFFFDFFF を設定します。16 ビット命令をフェッチする場合、EmbeddedICE-RT はデータバスの有効な半分のみをデータ値レジスタの内容と比較します。このように、1 つのウォッチポイントレジスタを使用して、データバスの上位半分と下位半分の両方のソフトウェアブレークポイントを捕捉することができます。

3 データマスクレジスタを 0x00000000 に設定します。

4 制御値レジスタを **PROT[0]**=0 に設定します。

5 制御マスクレジスタを **PROT[0]**=0 に設定します。他のビットはすべてセットします。

6 ユーザモード命令フェッチと非ユーザモード命令フェッチを区別したい場合は、制御値レジスタの **PROT[1]** 値を設定し、それに応じて制御マスクレジスタを設定します。

7 必要に応じて、**DBGEXT**、**RANGE**、および **CHAIN** ビットを同じように設定します。

注意： アドレス値レジスタは、設定する必要はありません。

ブレークポイントのセット

ソフトウェアブレークポイントをセットするには、以下の手順をとります。

1 希望するアドレスで命令を読み出し、ストアします。

2 そのアドレスでソフトウェアブレークポイントを表した特別なビットパターンをライトします。

ブレークポイントのクリア

ソフトウェアブレークポイントをクリアするには、命令をそのアドレスにリストアします。

9.22 ウォッチポイントの設定

この項では、ウォッチポイントユニットを設定してブレークポイントやウォッチポイントを生成する方法の例を示します。ウォッチポイントユニットレジスタの設定方法は、それ以外にもたくさんあります。たとえば、1 つ以上のアドレスマスクビットをセットすることによって簡単な範囲ブレークポイントを設定することができます。

ウォッチポイントユニットにウォッチポイントを（データアクセスで）起こさせるようにするには、次の手順をとります。

1 ウォッチポイントを設定するデータアクセスのアドレスでアドレス値レジスタを設定します。

2 アドレスマスクレジスタを 0x00000000 に設定します。

3 データ依存型ウォッチポイントが必要な場合にのみデータ値レジスタを設定します。すなわち、アドレスだけでなくリード／ライトされた実際のデータ値も一致させなければならない場合のみ設定します。データ値が関係ない場合は、データマスクレジスタを 0xFFFFFFFF に設定します（すべてのビットをセット）。そうでない場合は、0x00000000 に設定してください。

4 制御値レジスタを以下のように設定します。

PROT[0] セット

HWRITE リードの場合はクリア
ライトの場合はセット

SIZE[1:0] 該当のデータサイズに対応した値で設定

5 制御マスクレジスタを以下のように設定します。

PROT[0] クリア

HWRITE クリア

注意： リードやライトによってウォッチポイントを設定する場合は、このビットをセットすることができます。

SIZE[1:0] クリア

注意： データサイズアクセスにウォッチポイントを設定する場合は、これらのビットをセットすることができます。

他のすべてのビット
セット

6 ユーザモードデータアクセスと非ユーザモードデータアクセスを区別する必要がある場合は、制御値レジスタの **PROT[1]** ビットを設定し、それに応じて制御マスクレジスタを設定します。

7 必要であれば、**DBGEXT**、**RANGE**、および **CHAIN** ビットを同じように設定します。

9.23 アボートステータスレジスタ

この 32 ビットリード／ライトレジスタのビット 0 のみが使用されます。このビットは、アボート例外エントリがブレークポイント、ウォッチポイント、あるいは真のアボートによって起こされたかどうかを明らかにします。このレジスタの形式を図 9-14 に示します。

31:1	0
SBZ/RAZ	**DbgAbt**

図 9-14 デバッグアボートステータスレジスタ

ブレークポイントやウォッチポイントの結果、ARM720T コアがプリフェッチアボートやデータアボートを実行すると、ビット 0 がセットされます。デバッグアボートと外部アボート信号の両方が特定の命令やデータフェッチでアサートされると、外部アボートが優先され、DbgAbt（デバッグアボートビット）はセットされません。デバッグアボートビットはいったんセットされると、ユーザがリセットするまでセットされたままになります。レジスタは、MRC 命令や MCR 命令でアクセスされます。

9.24 デバッグ制御レジスタ

デバッグ制御レジスタは、6 ビット幅です。ウォッチポイントユニットレジスタにライトが行われると、デバッグ制御レジスタにライトが行われます。ウォッチポイントユニットレジスタがリードされると、デバッグ制御レジスタからのリードが行われます。詳細については、*「ウォッチポイントユニットレジスタ」*の項（9-34 ページ）を参照してください。

図 9-15 は、デバッグ制御レジスタの各ビットの機能を示しています。

5	4	3	2	1	0
EmbeddedICE-RT**ディセーブル**	**モニタモードイネーブル**	SBZ/RAZ	**INTDIS**	**DBGRQ**	**DBGACK**

図 9-15 デバッグ制御レジスタの形式

デバッグ制御レジスタのビット割当を表 9-9 に示します。

表 9-9 デバッグ制御レジスタのビット割当

ビット	機能
5	**ウォッチポイントやブレークポイントレジスタが設定されている間、**EmbeddedICE-RT **コンパレータの出力をディセーブルにするために使用されます。このビットは、**JTAG **を介してリードしたり、ライトすることができます。** **以下の場合は、ビット** 5 **をセットします。** • **ブレークポイントやウォッチポイントレジスタの設定** • **デバッグ制御レジスタのビット** 4 **の変更** **変更を行った後は、ビット** 5 **をクリアして** EmbeddedICE-RT **ロジックを再度イネーブルにし、新しいブレークポイントとウォッチポイントを実行可能にしてください。**
4	**ブレークポイントやウォッチポイントに達したときにコアの動作を確認するために使用します。** • **クリアしていると、ブレークポイントやウォッチポイントに達したときにコアはデバッグ状態になります。** • **セットしていると、ブレークポイントやウォッチポイントに達したときにコアはアボート例外を実行します。** **このビットは、**JTAG **からリードしたり、ライトすることができます。**

表 9-9 デバッグ制御レジスタのビット割当（続き）

ビット	機能
3	このビットはクリアしてください。
2	割り込みをディセーブルにするために使用します。 • セットされていると、コアの割り込みイネーブル信号（IFEN）は強制的に Low にさせられます。IFEN 信号は、表 9-10 に示すように駆動されます。 • クリアされていると、割り込みが有効になります。
1	DBGRQ の値を制御します。
0	DBGACK の値を制御します。

9.24.1 割り込みのディセーブル

IRQ と FIQ は、以下の条件下ではディセーブルになります。

- デバッギング中（**DBGACK** は High)
- **INTDIS** ビットがセットされているとき

コア割り込みイネーブル信号 **IFEN** は、表 9-10 に示すように駆動されます。

表 9-10 割り込み信号制御

DBGACK	INTDIS	IFEN	割り込み
0	0	1	許可
1	x	0	禁止
x	1	0	禁止

9.24.2 DBGRQ の制御

図 9-17（9-43 ページ）は、デバッグ制御レジスタのビット 1 に保存されている値が同期化され、外部 **DBGRQ** と論理和がとられた後、プロセッサに適用されることを示しています。この OR ゲートの出力は、マクロセルから外部へ出力される **DBGRQI** 信号です。

デバッグ制御レジスタのビット 1 と **DBGRQI** 間の同期化は、マルチプロセッサ環境でアシストされます。同期ラッチは、TAP コントローラステートマシンが RUN-TEST-IDLE 状態にあるときにのみ開かれます。これによって、システムのすべてのプロセッサがまだ稼動している間に、デバッグ状態になる条件をこれらのプロセッサでセットアップすることができます。条件がすべてのプロセッサにセットアップされると、この条件は RUN-TEST-IDLE 状態になることによってこれらのプロセッサに同時に適用されます。

9.24.3 DBGACK の制御

図 9-17（9-43 ページ）は、コアからの内部信号 **DBGACKI** の値はデバッグ制御レジスタのビット 0 に保持された値と論理和がとられ、ARM720T コアの周辺部での **DBGACK** の外部値を生成することを示しています。これによって、デバッグシステムは（コアからの内部信号 **DBGACK** が Low になっている時に）システムスピードアクセスが実行されていても、コアが依然としてデバッグ中であることをシステムの残りの部分に信号で伝えることができます。

9.25 デバッグステータスレジスタ

デバッグステータスレジスタは 13 ビット幅です。（リード／ライトビットをセットして）ライトのためにアクセスされると、ステータスビットにライトが行われます。（リード／ライトビットをクリアして）リードのためにアクセスされると、ステータスビットがリードされます。デバッグステータスレジスタの形式を図 9-16 に示します。

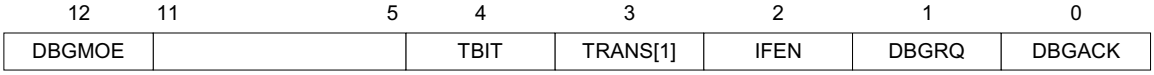


図 9-16 デバッグステータスレジスタの形式

このレジスタの各ビットの機能を、表 9-11 に示します。

表 9-11 デバッグステータスレジスタのビット割当

ビット	機能
12	コアが DBGRQ のアサートによってデバッグ状態になっているかどうかを確認するために、デバッガをイネーブルにします。
4	リードする TBIT をイネーブルにします。これによって、デバッガはプロセッサの状態と実行する命令を確認することができます。
3	コアからの、リードする HTRANS[1] 信号の状態をイネーブルにします。これによって、デバッガはデバッグ状態からのメモリアクセスが完了したかどうかを確認することができます。
2	リードするコア割り込みイネーブル信号 IFEN の状態をイネーブルにします。
1	リードする DBGRQ 同期バージョンの値をイネーブルにします。
0	リードする DBGACK 同期バージョンの値をイネーブルにします。

デバッグ制御レジスタとデバッグステータスレジスタの構造を、図 9-17 に示します。

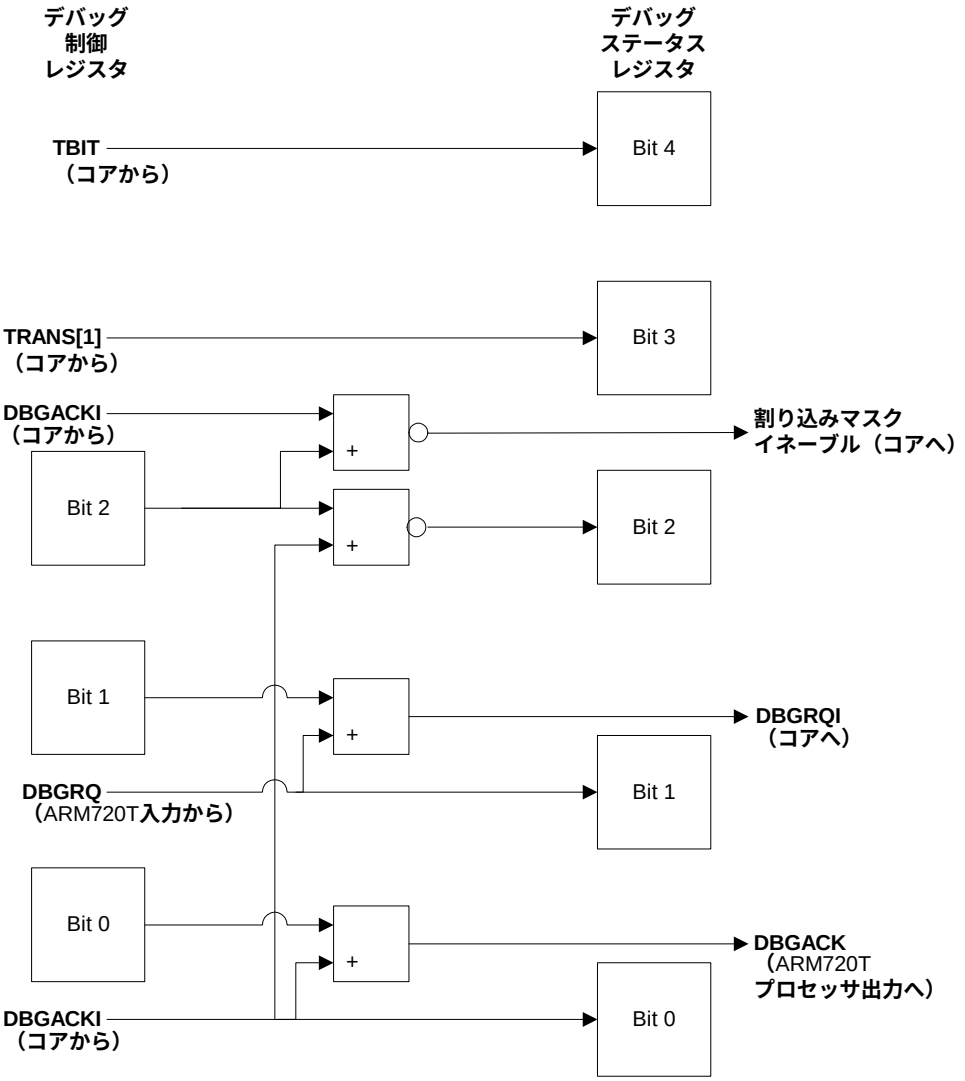


図 9-17 デバッグ制御レジスタとデバッグステータスレジスタの構造

9.26 ブレークポイントとウォッチポイントの組み合わせ

ウォッチポイントユニット 1 と 0 は、**CHAIN** 入力と **RANGE** 入力を使用して組み合わせることができます。**CHAIN** を使用すると、ウォッチポイント 1 があらかじめマッチしている場合に限りウォッチポイント 0 をトリガすることができます。**RANGE** を使用すると、両ウォッチポイントの出力を組み合わせることにより簡単な範囲チェックを行うことができます。

9.26.1 ブレークポイントとウォッチポイントの組み合わせ例

以下のように設定します。

Av[31:0]	アドレス値レジスタの値
Am[31:0]	アドレスマスクレジスタの値
A[31:0]	ARM720T プロセッサからのアドレスバス
Dv[31:0]	データ値レジスタの値
Dm[31:0]	データマスクレジスタの値
D[31:0]	ARM720T プロセッサからのデータバス
Cv[8:0]	制御値レジスタの値
Cm[7:0]	制御マスクレジスタの値
C[9:0]	ARM720T コア、他のウォッチポイントレジスタ、および **DBGEXT** 信号からの結合制御バス

CHAINOUT 信号

CHAINOUT 信号は、以下のように得られます。

WHEN (({Av[31:0],Cv[4:0]} XNOR {A[31:0],C[4:0]}) OR {Am[31:0],Cm[4:0]} == 0xFFFFFFFFF)

CHAINOUT = ((({Dv[31:0],Cv[6:4]} XNOR {D[31:0],C[7:5]}) OR {Dm[31:0],Cm[7:5]}) == 0x7FFFFFFF)

ウォッチポイントレジスタ 1 の **CHAINOUT** 出力は、ウォッチポイント 0 に **CHAIN** 入力を供給します。この **CHAIN** 入力によって、ブレークポイントとウォッチポイントをかなり複雑な構成で使用することができます。

注意： ウォッチポイント 1 への **CHAIN** 入力、およびウォッチポイント 0 からの **CHAIN** 出力はありません。

たとえば、マルチプロセスシステムのプロセス XXX を実行する場合に、YYY 位置にある命令のブレークポイントへのデバッガの要求を考えてみます。現在のプロセス ID がメモリに保存されている場合、ウォッチポイントとブレークポイントを組み合わせて上記の機能を実現することができます。ウォッチポイントアドレスは現在のプロセス ID を含む既知のメモリ位置を指し示し、ウォッチポイントデータは必要なプロセス ID を指し示し、そして **ENABLE** ビットはクリアされます。

ウォッチポイントのアドレスコンパレータ出力は、**CHAINOUT** ラッチのライトイネーブルを駆動するために使用されます。ラッチの入力は、同じウォッチポイントからのデータコンパレータの出力です。ラッチの出力は、ブレークポイントコンパレータの **CHAIN** 入力を駆動します。アドレス YYY はブレークポイントレジスタに保存されます。**CHAIN** 入力がアサートされると、ブレークポイントアドレスが一致しますので、ブレークポイントは正しくトリガされます。

9.26.2 DBGRNG 信号

DBGRNG 信号は、以下のように得られます。

DBGRNG = ((({Av[31:0],Cv[4:0]} XNOR {A[31:0],C[4:0]}) OR {Am[31:0],Cm[4:0]}) == 0xFFFFFFFFF) AND
((({Dv[31:0],Cv[7:5]} XNOR {D[31:0],C[7:5]}) OR
Dm[31:0],Cm[7:5]}) == 0x7FFFFFFFF)

ウォッチポイントレジスタ 1 の **DBGRNG** 出力は、ウォッチポイントレジスタ 0 に **RANGE** 入力を供給します。この **RANGE** 入力によって、2 つのブレークポイントを組み合わせて、範囲ブレークポイントを形成することができます。

注意： 選択された範囲は、2 のべき乗に制限されます。

たとえば、アドレスがメモリの先頭の 256 バイトにあるときにブレークポイントを発生させなければならないが、先頭の 32 バイトでは発生させないときは、ウォッチポイントレジスタを以下のように設定します。

ウォッチポイント 1：

1. ウォッチポイント 1 をアドレス値 0x00000000 とアドレスマスク 0x0000001F で設定します。
2. **ENABLE** ビットをクリアします。
3. 他のすべてのウォッチポイント 1 レジスタをブレークポイントに対して通常どおりに設定します。

   先頭の 32 バイト内にあるアドレスによって、**RANGE** 出力は High になりますが、ブレークポイントはトリガされません。

ウォッチポイント 0：

1. ウォッチポイント 0 をアドレス値 0x00000000 とアドレスマスク 0x000000FF で設定します。
2. ENABLE ビットをセットします。
3. RANGE ビットを 0 とマッチするように設定します。
4. 他のすべてのウォッチポイント 0 レジスタをブレークポイントに対して通常どおりに設定します。

ウォッチポイント 0 はマッチするが、ウォッチポイント 1 はマッチしない場合（すなわち、ウォッチポイント 0 への **RANGE** 入力が 0 の場合）、ブレークポイントがトリガされます。

9.27 EmbeddedICE-RT タイミング

EmbeddedICE-RT は、**HCLK** の立ち上がりエッジで **DBGEXT[1]** と **DBGEXT[0]** 入力をサンプリングします。

このページは白紙です。

10
ETM インタフェース

10 ETM インタフェース

この章では、ARM720T プロセッサに搭載される ETM インタフェースを説明します。ここでは、以下の項を記載しています。

10.1 ETM インタフェースについて

外部*組み込み型トレースマクロセル*（ETM）は ARM720T プロセッサに接続できますので、プロセッサが実行しているコードをリアルタイムでトレースすることができます。

一般に、ETM7 を ARM720T プロセッサに接続するには、グルーロジックはほとんど必要になりません。ETM は、JTAG インタフェースを介してプログラムします。このインタフェースは ARM ATP コントローラの拡張機能であり、スキャンチェイン 6 が割り当てられます。

注意： システムに 2 つ以上の ARM プロセッサがある場合、各プロセッサは独自の専用 ETM を装備していなければなりません。

ETM7 と ARM720T プロセッサの統合の詳細については、*"ETM7 (Rev 1) Technical Reference Manual"* を参照してください。

10.2 ETM7 インタフェースのイネーブル、ディセーブル

ARM デバッグツールの制御下では、ETM7 **PWRDOWN** 出力を使用して ETM をイネーブル、ディセーブルしています。**PWRDOWN** が High になっていると、ETM が現在イネーブルではないことを示していますので、**CLK** 入力を停止して、他の ETM 信号を安定化させることができます。これによって、トレースを実行していないときの電力消費を低減させることができます。

TAP リセット（**nTRST**）が発生すると、ETM7 制御レジスタが設定されるまで **PWRDOWN** は強制的に High にされます（このレジスタの詳細については、*"Embedded Trace Macrocell Specification"* を参照してください。

PWRDOWN は、デバッグ動作の開始時に自動的にクリアされます。

ARM720T プロセッサでは、ETM インタフェース端子は **ETMEN** 入力によってゲート制御されます。これは、**ETMEN** 入力が Low ならば、ETM インタフェースの出力端子はすべて安定していることを示しています。この **ETMEN** 入力は、以下の出力のいずれか 1 つに接続することによって制御することができます。

- ETM7 の **ETMEN** 出力
- ETM7 の **PWRDOWN** 反転出力

10.3 ETM7 マクロセルと ARM720T プロセッサ間の接続

表 10-1 は、ETM7 マクロセルと ARM720T プロセッサ間で行わなければならない接続を示しています。

表 10-1 ETM7 マクロセルと ARM720T プロセッサ間の接続

ETM7 マクロセル信号名	ARM720T プロセッサ信号名
A[31:0]	ETMADDR[31:0]
ABORT	ETMABORT
ARMTDO	DBGTDO
BIGEND	ETMBIGEND
CLK[a]	HCLK[a]
CLKEN	ETMCLKEN
CPA	ETMCPA
CPB	ETMCPB
DBGACK	ETMDBGACK
DBGRQ[b]	DBGRQ[b]
nMREQ	ETMnMREQ
SEQ	ETMSEQ
MAS[1:0]	ETMSIZE[1:0]
nCPI	ETMnCPI
nEXEC	ETMnEXEC
nOPC	ETMnOPC
nRESET	HRESETn
nRW	ETMnRW
nTRST[a]	DBGnTRST[a]
PROCID[31:0]	ETMPROCID[31:0]
PROCIDWR	ETMPROCIDWR
ETMEN または PWRDOWN 反転信号	ETMEN[c]
-	ETMHIVECS[d]
RANGEOUT[0]	DBGRNG[0]
RANGEOUT[1]	DBGRNG[1]
RDATA[31:0]	ETMRDATA[31:0]
TBIT	ETMTBIT
TCK[a]	HCLK[a]
TCKEN	DBGTCKEN
TDI	DBGTDI

表 10-1 ETM7 マクロセルと ARM720T プロセッサ間の接続（続き）

ETM7 マクロセル信号名	ARM720T プロセッサ信号名
TDO[e]	DBGTDO
TMS	DBGTMS
WDATA[31:0]	ETMWDATA[31:0]
INSTRVALID	ETMINSTRVALID

a. 「クロックとリセット」の項（10-3 ページ）を参照してください。

b. 「デバッグ要求ワイヤリング」の項（10-3 ページ）を参照してください。

c. 「ETM7 インタフェースのイネーブル、ディセーブル」の項（10-1 ページ）を参照してください。

d. このピンは未接続のままにしておきます。

e. 「TAP インタフェースワイヤリング」の項（10-3 ページ）を参照してください。

10.4 クロックとリセット

ARM720T プロセッサは、シングルクロック **HCLK** をメインシステムクロックとしても JTAG クロックとしても使用します。プロセッサクロックを ETM の **HCLK** と **TCK** の両方に接続してください。そして、**TCKEN** を使用すると、JTAG インタフェースを制御することができます。

ARM720T プロセッサのウォームリセットによってトレースするには、TAP リセット（**nTRST** を **DBGnTRST** に接続します）を使用して ETM7 状態をリセットします。

ETM7 クロックとリセットの詳細については、*"ETM7 Technical Reference Manual"* を参照してください。

10.5 デバッグ要求ワイヤリング

ETM7 の **DBGRQ** 出力を ARM720T プロセッサの **DBGRQ** 入力に接続することを推奨します。この入力を既に使用している場合は、**DBGRQ** 入力の論理和をとることができます。詳細については、*"ETM7 Technical Reference Manual"* を参照してください。

10.6 TAP インタフェースワイヤリング

ARM720T プロセッサは、スキャンチェイン拡張入力を供給していません。ARM 社では、ARM720T プロセッサと ETM7 TAP コントローラを並列に接続することを勧めています。詳細については、*"ETM7 (Rev 1) Technical Reference Manual"* を参照してください。

このページは白紙です。

11
テストサポート

11 テストサポート

この章では、ARM720T プロセッサ合成ロジックおよび TCM のテスト方法論および CP15 テストレジスタを解説します。ここでは、以下の項を記載しています。

11.1 ARM720T テストレジスタについて

ARM720T プロセッサのコプロセッサ 15 レジスタ c15 を使用して、デバイス固有のテスト動作を提供します。このレジスタを使用して、以下の項目にアクセスし、制御することができます。

- *テストステートレジスタ*、11-3 ページ
- *キャッシュテストレジスタと動作*、11-3 ページ
- *MMU テストレジスタと動作*、11-8 ページ

これらの動作はテストでのみ使用してください。*“ARM Architecture Reference Manual”*では、このレジスタをインプリメンテーション時定義依存として説明しています。

CP15 テスト動作の形式は、以下の通りです。

MCR/MRC p15, opcode_1, <Rd>, c15, <CRm>, <opcode_2>

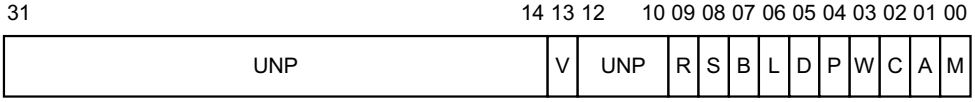


図 11-1 CP15 **の** MRC **および** MCR **ビットパターン**

L ビットにより、MCR（L ビットは 1 にセット）と MRC（L ビットは 0 にセット）を区別します。

11.2 自動テストパターン生成（ATPG）

スキャン挿入はすでに実行され、ARM720T プロセッサで決定されています。*自動テストパターン生成*（ATPG）ツールを使用すると、必要なスキャンパターンを作成してすべてのレジスタからロジック出力をテストすることができます。

ARM720T ATPG テスト信号の概要を表 11-1 に示します。

表 11-1 ATPG テスト信号の概要

テスト信号	方向	説明
TESTENABLE	Input	この信号は、スキャンテスト時クロックが確実にフリーランニングするようにします。TESTENABLE は、以下の通りでなければなりません： スキャンテスト期間を通して High になっている。 機能モード時は Low になっている。
SCANENABLE	Input	この信号により、スキャンチェインを介してベクタをシリアルにシフトさせることができます。この信号は、I/O ピンを使用して制御してください。機能モード時、この信号は Low に固定してください。
SCANIN0-SCANIN6	Inputs	プロセッサコアスキャンチェイン入力
SCANOUT-SCANOUT6	Outputs	プロセッサコアスキャンチェイン出力
HCLK	Input	システムクロック。すべての信号は、HCLK の立ち上がりエッジと関連があります。
HCLKEN	Input	AHB 転送の同期イネーブル。High になっていると、HCLK の次の立ち上がりが、ARM720T プロセッサが組み込まれた AHB システムの立ち上がりでもあることを示しています。この信号は、AMBA バスとコアを同じ周波数にするシステムでは High に固定してください。
DBGTCKEN	Input	デバッグロジックの同期イネーブル。スキャンテスト中この信号は High に固定してください。
HRESETn	Input	これは、システムおよびバスのアクティブ Low リセット信号です。
DBGnTRST	Input	これは、内部状態のアクティブ Low リセット信号です。この信号は、非同期リセットレベル入力です。

ATPG モードでは、**HRESETn**、**DBGnTRST**、**TESTENABLE** 信号は 1 に制限されます。**TESTENABLE** 信号は内部クロックモジュール内部にのみ進み、設計のすべてのスキャンフリップフロップが同じ位相を使用するようにします。2 つの機能クロックドメイン間にはロックアップラッチはありません。

11.2.1 ARM720T プロセッサ INTEST/EXTEST ラッパ

オート挿入スキャンチェインに加えて、ARM720T プロセッサはオプションの INTEST/EXTEST スキャンチェイン、すなわちスキャンチェイン 0 用の信号をすべて持っています。

ATPG

ATPG には、テストイネーブルとシングルスキャンイネーブルとともに 7 つのスキャンチェインが提供されます。

11.3 テストステートレジスタ

テストステートレジスタは、1 ビット、すなわちビット 0 を格納しているのみです。

ビット 0 セットの場合　　MMU とキャッシュテストをイネーブルにします。

ビット 0 クリアの場合　　MMU とキャッシュテストをディセーブルにします。

リセット時（**HRESETn** が Low の時）、ビット 0 はクリアされます。

テストステートレジスタの動作を表 11-2 に示します。

表 11-2 テストステートレジスタの動作

動作	命令
テストレジスタのライト	MCR p15, 7, <Rd>, c15, c15, 7
テストレジスタのリード	MRC p15, 7, <Rd>, c15, c15, 7

注意： キャッシュと MMU テスト動作は、テストステータスレジスタがオンになっている場合にのみサポートされます。

11.4 キャッシュテストレジスタと動作

キャッシュは、ARM v4T プログラマーズモデルで定義された CP15 レジスタ c7 および c9 に対して MCR 命令、MRC 命令を使用することによって維持されます。CP15 レジスタ c15 に対して MCR 命令、MRC 命令を使用して、追加の動作を実行することができます。これらの動作を、レジスタ c7 および c9 を使用した動作と組み合わせると、キャッシュのテストを完全にソフトウェアで行うことができます。

CP15 レジスタ c7 はライトオンリであり、以下の 1 つの機能しか提供しません。

- キャッシュの無効化

CP15 レジスタ c9 の動作はリードおよびライトです。可能なリード／ライト動作は、以下の通りです。

- ビクティムとロックダウンベースのライト
- ビクティムのライト

CP15 レジスタ c15 の動作は、以下の通りです。

- レジスタ C15.C へのライト
- レジスタ C15.C からのリード
- C15.C への CAM リード
- CAM ライト
- C15.C への RAM リード
- C15.C からの RAM ライト
- CAM マッチ、C15.C への RAM リード

注意： CAM マッチ、RAM リード動作の場合、それぞれの MMU はルックアップは実行せず、キャッシュミスが発生してもラインフィルは起こりません。

レジスタ c15 の動作は、すべて MCR として発行されます。Rd フィールドは動作のアドレスを定義します。したがって、データは CP15 の CP15.C から供給されるか、あるいは CP15.C にラッチされるかのどちらかです。これらの 32 ビットレジスタは、CP15 の MCR、MRC 命令でアクセスされます。

表 11-3 は、レジスタ c7、c9、c15 の動作を要約しています。

表 11-3 CP15 レジスタ c7、c9、c15 動作の要約

機能	Rd	命令
キャシュの無効化	SBZ	MCR p15, 0, <Rd>, c7, c7, 0
キャッシュビクティムおよびロックダウンベースのライト	ビクティム = ベース	MCR p15, 0, <Rd>, c9, c0, 0
キャシュビクティムのライト	ビクティム、セグメント	MCR p15, 0, <Rd>, c9, c1, 0
C15.C への CAM リード	セグメント	MCR p15, 2, <Rd>, c15, c7, 2
CAM ライト	タグ、セグメント、ダーティ	MCR p15, 2, <Rd>, c15, c7, 6
C15.C への RAM リード	セグメント、ワード	MCR p15, 2, <Rd>, c15, c11, 2
C15.C からの RAM ライト	セグメント、ワード	MCR p15, 2, <Rd>, c15, c11, 6
CAM マッチ、C15.C への RAM リード	タグ、セグメント、ワード	MCR p15, 2, <Rd>, c15, c7, 5
レジスタ C15.C へのライト	データ	MCR p15, 3, <Rd>, c15, c3, 0
レジスタ C15.C からのリード	データリード	MRC p15, 3, <Rd>, c15, c3, 0

CAM リード Rd 形式を図 11-2 に示します。

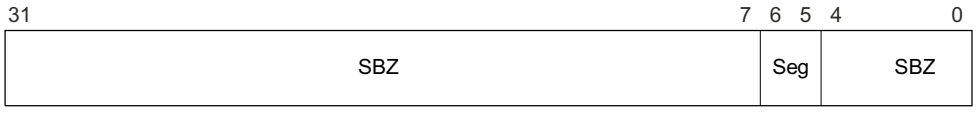


図 11-2 CAM リードの Rd 形式

CAM ライト Rd 形式を図 11-3 に示します。

図 11-3 CAM ライトの Rd 形式

図 11-3 では、ビットラベルは以下の意味をもちます。

V 有効

De *ダーティイーブン*（ワード [3:0]）。未使用

Do *ダーティーオッド*（ワード [7:4]）。未使用

WB ライトバック。未使用

RAM リード Rd 形式を図 11-4 に示します。

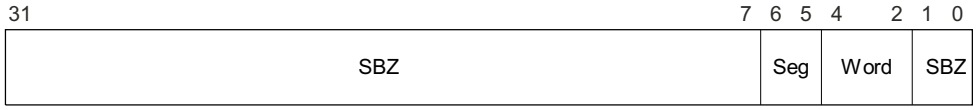


図 11-4 RAM リードの Rd 形式

RAM ライト Rd 形式を図 11-5 に示します。

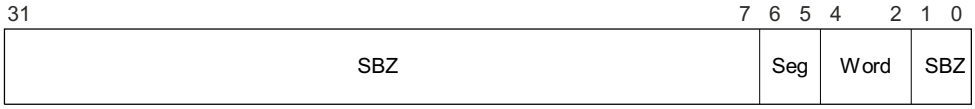


図 11-5 RAM ライトの Rd 形式

CAM マッチ、RAM リード Rd 形式を図 11-6 に示します。

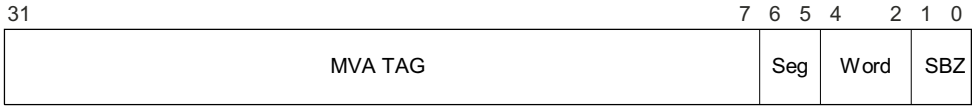


図 11-6 CAM マッチと RAM リードの Rd 形式

CAM リードデータ形式を図 11-7 に示します。

図 11-7 CAM リードのデータ形式

RAM リードデータ形式を図 11-8 に示します。

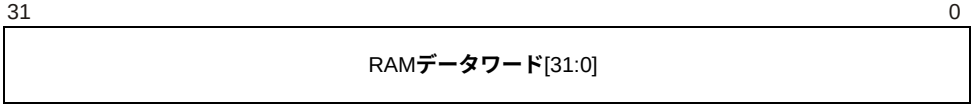


図 11-8 RAM リードのデータ形式

CAM マッチ、RAM リードデータ形式を図 11-9 に示します。

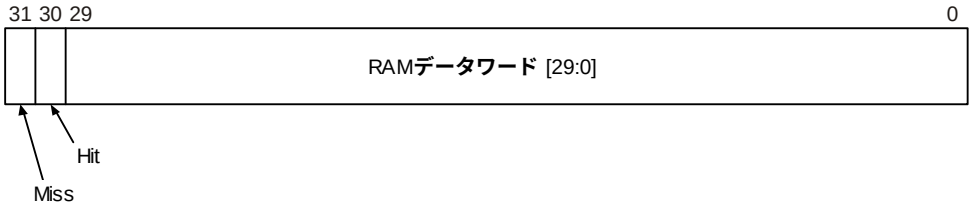


図 11-9 CAM マッチと RAM リードのデータ形式

11.4.1 CAM と RAM のアドレッシング

CAM リード／ライト、また RAM リード／ライト動作では、セグメント、インデックス、ワード（RAM 動作の場合）を指定してください。CAM や RAM 動作では、該当のセグメントのビクティムポインタの値が使用されますので、CAM や RAM 動作が行われる前にビクティムポインタに値をライトするようにしてください。

MCR ライトビクティムおよびロックダウンベースを使用すると、ビクティムポインタは CAM リード／ライトが行われるたびに、また RAM リード／ライトが行われるたびにインクリメントされます。MCR ライトビクティムを使用すると、ビクティムポインタは CAM リード／ライトのみが行われるたびにインクリメントします。これによって、セグメント全体に対して CAM および RAM のリード／ライトを効率的に行えるようにしています。キャッシュビクティムとロックダウンライト動作を表 11-4 に示します。

表 11-4 キャッシュビクティムとロックダウンライト動作

動作	命令
キャッシュビクティムとロックダウンベースのライト	MCR p15, 0, <Rd>, c9, c0, 0 MCR p15, 0, <Rd>, c9, c0, 1
キャッシュビクティムのライト	MCR p15, 0, <Rd>, c9, c1, 0 MCR p15, 0, <Rd>, c9, c1, 1

キャッシュビクティムとロックダウンベースライト Rd 形式を、図 11-10 に示します。

図 11-10 キャッシュビクティムとロックダウンベースライトの Rd 形式

キャッシュビクティムライト Rd 形式を図 11-11 に示します。

図 11-11 キャッシュビクティムライトの Rd 形式

別のキャッシュテストレジスタ C15.C には、MCR CAM リードが実行されると必ずアドレス指定されたセグメントの現在のビクティムでライトが行われます。これは、CAM および RAM の値のリード前に各セグメントの現在のビクティムポインタの値を設定するためにデバッグでの使用を目的としたものです。このため、値は後でリストアすることができます。

例 11-1 は、キャッシュのソフトウェアテストを実行するためのコードサンプルを示しています。この例は、レジスタ C15.C による典型的な動作を示しています。

例 11-1 キャッシュテスト動作

```
; CAM write, read and check for segment 2

; Write cache victim pointer with index 0, segment 2
            MOV r0,#0
            ORR r1,r0,#2 :SHL: 0x5
            MCR p15,0,r1,c9,c1,0

; Write pattern in 0xFFFFFF9E in all 64 CAM lines
            MVN r2,#1               ; bit 0 should be '0'
            BIC r2,r2,#0x20         ; write segment 2
            MOV r8,#64
loop0       MCR p15,2,r2,c15,c7,6   ; write CAM, index auto-incremented
            SUBS r8,r8,#1
            BNE loop0

; Now read and check
; Reset victim pointer to index 0, segment 2
            MOV r0,#0
            ORR r1,r0,#2 :SHL:0x5
            MCR p15,0,r1,c9,c1,0
            MOV r8,#64
            MOV r3,#0x40                ; read segment 2
            BIC r2,r2,#0x60             ; clear bit 5 and 6 (always read as '0')
loop1       MCR p15,3,r0,c15,c3,0       ; write C15.C to '0'
            MCR p15,2,r3,c15,c7,2       ; read CAM to C15.C
            MRC p15,3,r4,c15,c3,0       ; read C15.C to R4
            BIC r4,r4,#1                ; clear LFSR bit
            CMP r4,r2
            BNE TEST_FAIL
            SUBS r8,r8,#1
            BNE loop1
            B TEST_PASS

; RAM write, read and check for segment 1

; Write cache victim pointer with index 0, segment 1
            MOV r0,#0
            ORR r1,r0,#1 :SHL: 0x5
            MCR p15,0,r1,c9,c1,0

; Write pattern 0x5A5A5A5A in RAM line (eight words)
            LDR r0,=0x5A5A5A5A
            MOV r8,#8
            MOV r2,#0x10                ;write segment 1,word 0
            MCR p15,3,r0,c15,c3,0       ; write RAM data in C15.C
loop0       MCR p15,2,r2,c15,c11,6      ; write RAM
            ADD r2,r2,#0x04             ; next word
            SUBS r8,r8,#1
            BNE loop0
```

```
; Now read and check

            MOV r8,#8
            MOV r2,#0x10
            MOV r1,#0
loop1       MCR p15,3,r1,c15,c3,0       ; write C15.C to '0'
            MCR p15,2,r2,c15,c11,2      ; read RAM to C15.C
            MRC p15,3,r5,c15,c3,0       ; read C15.C to R4
            ADD r2,r2,#0x04
            CMP r5,r0
            BNE TEST_FAIL
            SUBS r8,r8,#1
            BNE loop1
            B TEST_PASS
```

11.5 MMU テストレジスタと動作

TLB は、ARM v4T プログラマーズモデルで定義された CP15 レジスタ c2、c3、c5、c6、c8、および c10 に対して MCR 命令、MRC 命令を使用することによって維持されます。

CP15 レジスタ c2 の動作は、*変換テーブルベース*（TTB）を制御します。これらの動作は、以下の通りです。

- 変換テーブルベースレジスタのライト
- 変換テーブルベースレジスタのリード

CP15 レジスタ c3 の動作は、*ドメインアクセス制御*（DAC）レジスタを制御します。これらの動作は、以下の通りです。

- DAC レジスタライト
- DAC レジスタリード

CP15 レジスタ c5 の動作は、*フォールトステータスレジスタ*（FSR）を制御します。これらの動作は、以下の通りです。

- FSR ライト
- FSR リード

CP15 レジスタ c6 の動作は、*フォールトアドレスレジスタ*（FAR）を制御します。これらの動作は、以下の通りです。

- FAR ライト
- FAR リード

CP15 レジスタ c8 の動作は TLB を制御しますが、すべてライトオンリです。これらの動作は、以下の通りです。

- TLB の無効化
- MVA を使用したシングルエントリの無効化

CP15 レジスタ c10 の動作は、TLB ロックダウンを制御します。これらの動作は、以下の通りです。

- ビクティム、ロックダウンベース、および保存ビットのリード
- ビクティム、ロックダウンベース、および保存ビットのライト

CAM、RAM1、および RAM2 で動作させる CP15 レジスタ c15 の動作を、表 11-5 に示します。

表 11-5 CAM、RAM1、および RAM2 レジスタ c15 の動作

機能	Rd	データ
C15.M への CAM リード	SBZ	タグ、サイズ、V、P
CAM ライト	Tag, Size, V, P	
C15.M への RAM1 リード	SBZ	プロテクション
RAM1 ライト	Protection	
C15.M への RAM2 リード	SBZ	PA タグ、サイズ
RAM2 ライト	PA Tag, Size	PA タグ、サイズ
CAM マッチ、C15.M への RAM1 リード	MVA	フォールト、ミス、プロテクション

注意： CAM マッチ、RAM1 リード動作については、TLB ミスになってもページウォーキングは起こりません。

これらのレジスタ c15 動作はすべて MCR として発行されるものであり、これはリード、マッチ動作が CP15 の CP15.M レジスタにラッチされなければならないことを意味しています。これは、以下の CP15 MRC 命令でリードされる 32 ビットレジスタです。

Read from register CP15.M

表 11-6 は、レジスタ c2、c3、c5、c6、c8、c10、および c15 動作を要約しています。

表 11-6 レジスタ c2、c3、c5、c6、c8、c10、および c15 動作

機能	Rd	命令
変換テーブルベースレジスタのリード	TTB	MRC p15, 0, <Rd>, c2, c0, 0
変換テーブルベースレジスタのライト	TTB	MCR p15, 0, <Rd>, c2, c0, 0
ドメイン [15:0] アクセス制御のリード	DAC	MRC p15, 0, <Rd>, c3, c0, 0
ドメイン [15:0] アクセス制御のライト	DAC	MCR p15, 0, <Rd>, c3, c0, 0
FSR リード	FSR	MRC p15, 0, <Rd>, c5, c0, 0
FSR ライト	FSR	MCR p15, 0, <Rd>, c5, c0, 0
FAR リード	FAR	MRC p15, 0, <Rd>, c6, c0, 0
FAR ライト	FAR	MCR p15, 0, <Rd>, c6, c0, 0
TLB の無効化	SBZ	MCR p15, 0, <Rd>, c8, c5, 0 MCR p15, 0, <Rd>, c8, c6, 0 MCR p15, 0, <Rd>, c8, c7, 0
（MVA を使用した）TLB シングルエントリの無効化	MVA 形式	MCR p15, 0, <Rd>, c8, c5, 1 MCR p15, 0, <Rd>, c8, c6, 1 MCR p15, 0, <Rd>, c8, c7, 1
TLB ロックダウンのリード	TLB ロックダウン	MRC p15, 0, <Rd>, c10, c0, 0
TLB ロックダウンのライト	TLB ロックダウン	MCR p15, 0, <Rd>, c10, c0, 0
C15.M への CAM リード	SBZ	MCR p15, 4, <Rd>, c15, c7, 4
CAM ライト	Tag, Size, V, P	MCR p15, 4, <Rd>, c15, c7, 0
C15.M への RAM1 リード	SBZ	MCR p15, 4, <Rd>, c15, c11, 4

表 11-6 レジスタ c2、c3、c5、c6、c8、c10、および c15 動作（続き）

機能	Rd	命令
RAM1 ライト	Protection	MCR p15, 4, <Rd>, c15, c11, 0
C15.M への RAM2 リード	SBZ	MCR p15, 4, <Rd>, c15, c3, 5
RAM2 ライト	PA Tag, Size	MCR p15, 4, <Rd>, c15, c3, 1
CAM マッチ、C15.M への RAM1 リード	MVA	MCR p15, 4, <Rd>, c15, c13, 4
C15.M リード	Data	MRC p15, 4, <Rd>, c15, c3, 0

図 11-12 は、CAM ライトの Rd 形式および CAM リードのデータ形式を示しています。

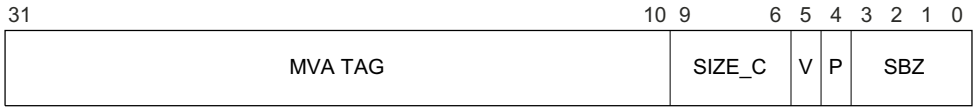


図 11-12 CAM ライトの Rd 形式および CAM リードのデータ形式

図 11-12 では、V は有効ビットであり、P は保存ビットであり、SIZE_C はメモリ領域サイズを設定します。SIZE_C の許容値を表 11-7 に示します。

表 11-7 CAM メモリ領域サイズ

SIZE_C[3:0]	メモリ領域サイズ
b1111	1MB
b0111	64KB
b0011	16KB
b0001	4KB
b0000	1KB

図 11-13 は、RAM1 ライトの Rd 形式を示しています。

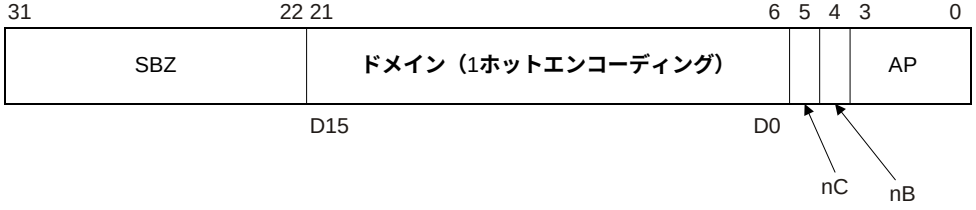


図 11-13 RAM1 ライトの Rd 形式

図 11-13 では、AP[3:0] はメモリ領域のアクセス許可ビットの設定を決めます。許可値を表 11-8 に示します。

表 11-8 アクセス許可ビットの設定

AP[3:0]	アクセス許可ビット
b1000	b11
b0100	b10
b0010	b01
b0001	b00

図 11-14 は、RAM1 リードのデータ形式を示しています。

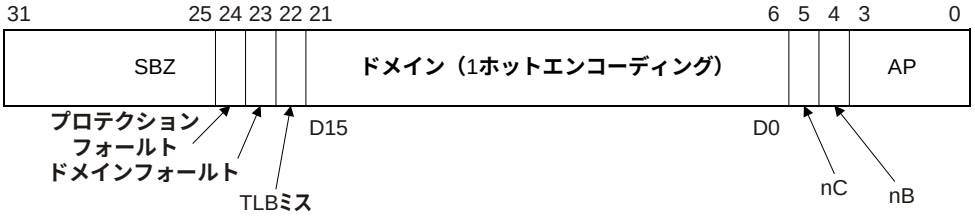


図 11-14 RAM1 リードのデータ形式

図 11-14 では、ビット [24:22] はマッチ動作に対してのみ有効です。この場合、表 11-9 に示す値が適用されます。

表 11-9 ミスおよびフォールトエンコーディング

プロテクション フォールト	ドメイン フォールト	TLB ミス	機能
0	0	0	ヒット、OK
0	1	0	ヒット、ドメインフォールト
1	0	0	ヒット、プロテクションフォールト
1	1	0	ヒット、プロテクション、およびドメインフォールト
-	-	1	TLB ミス

図 11-15 は、RAM2 ライトの Rd 形式と RAM2 リードのデータ形式を示しています。

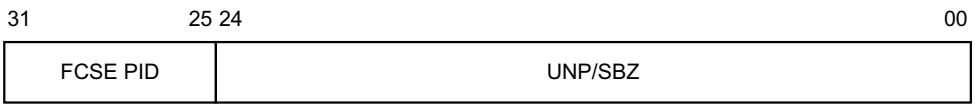


図 11-15 RAM2 ライトの Rd 形式と RAM2 リードのデータ形式

図 11-15 では、SIZE_R2 はメモリ領域サイズを設定します。SIZE_R2 の許容値を表 11-10 に示します。

表 11-10 RAM2 メモリ領域サイズ

SIZE_R2[3:0]	メモリ領域サイズ
b1000	1MB
b0100	64KB
b0010	16KB
b0000	4KB
b0001	1KB

注意： SIZE_R2 のエンコーディングは、SIZE_C とは異なります。

11.5.1 CAM、RAM1、および RAM2 のアドレッシング

CAM リードやライト、RAM1 リードやライト、および RAM2 リードやライト動作では、インデックスを指定してください。CAM、RAM1 動作ではビクティムポインタの値が使用されますので、CAM や RAM1 操作の前にこれをライトしなければなりません。RAM2 は、CAM や RAM1 操作に使用するビクティムポインタのパイプラインバージョンを使用します。これは、RAM2 アレイのインデックス N からリードするには、最初に CAM または RAM1 のいずれかのインデックス N にアクセスを行わなければならないことを示しています。

TLB ロックダウンライト動作は、以下の通りです。

MCR p15, 0, <Rd>, c10, c0, 0

TLB ロックダウンライト Rd 形式を図 11-16 に示します。

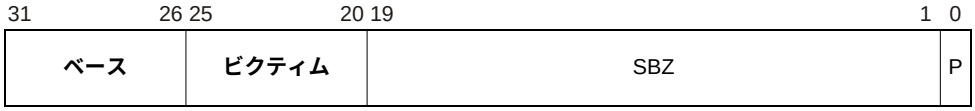


図 11-16 TLB ロックダウンライトの Rd 形式

例 11-2 は、MMU のソフトウェアテストを実行するためのコードサンプルを示しています。この例では、C15.M での代表的な操作を記載しています。

例 11-2 MMU テスト動作

```
; MMU write, read and check for CAM, RAM1 and RAM2

; Load victim pointer with 0
            MOV r0,#0
            MCR p15,0,r0,c10,c0,0

; Write pattern 0x5A5A5A50 in CAM
; Write pattern 0x0025A5A5 in RAM1
; Write pattern 0xF0F0F0C0 in RAM2
            LDR r2,=0x5A5A5A50
            LDR r3,=0x0025A5A5
            LDR r4,=0xF0F0F0C0
            MOV r5,#64
; Write all 64 lines
loop0       MCR p15,4,r2,c15,c7,0       ; write CAM
            MCR p15,4,r3,c15,c11,0      ; write RAM1
            MCR p15,4,r4,c15,c3,1       ; write RAM2, pointer auto-incremented here
            SUBS r5,r5,#1
            BNE loop0

; Now read and check
; Reset victim pointer
            MOV r0,#0
            MCR p15,0,r0,c10,c0,0
            MOV r8,#64
loop1       MCR p15,4,r5,c15,c7,4       ; read CAM to C15.M
            MRC p15,4,r5,c15,c3,6       ; read C15.M to R5
            MCR p15,4,r6,c15,c11,4
            MRC p15,4,r6,c15,c3,6       ; read RAM1 to R6
            BIC r5,r5,#0x01c00000       ; mask fault/miss bits

            MCR p15,4,r7,c15,c3,5
            MRC p15,4,r7,c15,c3,6       ; read RAM2 to R7

            CMP r5,r2
            CMPEQ r6,r3
            CMPEQ r7,r4
            BNE TEST_FAIL

            SUBS r8,r8,#1
            BNE loop1
            B TEST_PASS
```

このページは白紙です。

付録 A
信号の説明

A　信号の説明

この付録では、ARM720T プロセッサのインタフェース信号を説明します。ここでは、以下の項を記載しています。

A.1　AMBA インタフェース信号

AMBA インタフェース信号を表 A-1 に示します。

表 A-1 AMBA インタフェース信号

信号名	タイプ	説明
HCLK	Input	バスクロック。これは ARM720T プロセッサの唯一のクロックです。
HADDR[31:0]	Output	32 ビットシステムアドレスバス
HTRANS[1:0]	Output	現在の転送のタイプを示します。
HBURST[2:0]	Output	現在の転送のバースト長を示します。
HWRITE	Output	現在の転送の方向を示します。
HSIZE[2:0]	Output	現在の転送のサイズを示します。
HPROT[3:0]	Output	プロテクション制御信号
HGRANT	Input	許可されたバス転送
HREADY	Input	現在の転送が終了したことを示します。
HRESP[1:0]	Input	転送ステータスを示します。
HWDATA[31:0]	Output	ライトデータバス
HRDATA[31:0]	Input	リードデータバス
HBUSREQ	Output	バス転送要求
HLOCK	Output	ロックされたアクセスを示します。
HCLKEN	Input	バスクロックイネーブル
HRESETn	Input	グローバルリセット（ローアクティブ信号）

A.2 コプロセッサインタフェース信号

コプロセッサインタフェース信号を表 A-2 に示します。

表 A-2 コプロセッサインタフェース信号の説明

名前	タイプ	説明
EXTCPA	Input	**外部コプロセッサなし** **外部コプロセッサが存在しない場合、この信号は** High **でなければなりません。**
EXTCPB	Input	**外部コプロセッサビジー**
EXTCPCLKEN	Output	**外部コプロセッサクロックイネーブル**
EXTCPDIN[31:0]	Output	**外部コプロセッサデータイン**
EXTCPDOUT[31:0]	Input	**外部コプロセッサデータアウト**
CPnCPI	Output	**コプロセッサ命令（ローアクティブ信号）** Low **の時、この信号は** ARM720T **プロセッサがコプロセッサ命令を実行中であることを示します。**
CPnOPC	Output	**演算コードフェッチ（ローアクティブ信号）** Low **の時、この信号はプロセッサがメモリから命令をフェッチ中であることを示しています。**High **の時、データが存在しているときはそのデータを転送中です。この信号は、**ARM **パイプラインをトラッキングするためにコプロセッサで使用されます。**
CPTBIT	Output	Thumb **ステート** High **の時、この信号は** Thumb **命令セットをプロセッサが実行中であることを示します。**Low **の時、プロセッサは** ARM **命令セットを実行中です。**
CPnTRANS	Output	**コプロセッサ変換（ローアクティブ信号）** High **の時、コプロセッサインタフェースは非特権モードになっています。**Low **の時、コプロセッサインタフェースは特権モードになっています。** **コプロセッサは、コプロセッサの応答を確認するときにサイクルごとにこの信号をサンプリングします。**
CPnMREQ	Output	**コプロセッサメモリ要求（ローアクティブ信号）**
EXTCPDBE	Input	**外部コプロセッサデータバスイネーブル** High **の時、この信号はコプロセッサが自己のデータバス CPDATA を駆動しようとしていることを示しています。コプロセッサインタフェースが使用されていない場合は、この信号は** Low **でなければなりません。**

A.3 JTAG およびテスト信号

JTAG およびテスト信号の説明を表 A-3 に示します。

表 A-3 JTAG およびテスト信号の説明

名前	タイプ	説明
DBGIR[3:0]	Output	TAP 命令レジスタ これらの信号は、TAP コントローラ命令レジスタにロードされた現在の命令を反映しています。TAP ステートマシンが UPDATE-DR ステートにある時に、これらの信号は HCLK の立下りエッジで変化します。この信号を使用すると、ARM720T プロセッサ TAP コントローラを使用してより多くのスキャンチェインを追加することができます。
DBGSREG[3:0]	Output	スキャンチェインレジスタ これらの信号は、TAP コントローラで選択された現在のスキャンチェインの ID 番号を反映しています。TAP ステートマシンが UPDATE-DR ステートにある時に、これらの信号はXTCKの立下りエッジで変化します。
DBGSDIN	Output	バウンダリスキャンシリアルデータイン この信号は、外部スキャンチェインに適用されるシリアルデータです。
DBGSDOUT	Input	バウンダリスキャンシリアルデータアウト この信号は、外部スキャンチェインからのシリアルデータです。この信号によって、1 つの DBGTDO ポートを使用することができます。外部スキャンチェインが接続されていない場合、この入力は Low になっていなければなりません。
DBGTAPSM[3:0]	Output	コントローラステータス これらの信号は、TAP コントローラマシンの現在の状態を表しています。これらの信号は XTCK の立ち上がりエッジで変化します。この信号を使用すると、ARM720T プロセッサ TAP コントローラを使用してより多くのスキャンチェインを追加することができます。
DBGCAPTURE[a]	Output	CAPTURE ステート信号 High の時、この信号は TAP コントローラステートマシンが CAPTURE ステートにあることを示します（図 9-8（9-19 ページ）を参照）。
DBGSHIFT[a]	Output	SHIFT ステート信号 High の時、この信号は TAP コントローラステートマシンが SHIFT ステートにあることを示します（図 9-8（9-19 ページ）を参照）。
DBGUPDATE[a]	Output	UPDATE ステート信号 High の時、この信号は TAP コントローラステートマシンが UPDATE ステートにあることを示します（図 9-8（9-19 ページ）を参照）。
DBGINTEST[a]	Output	INTEST ステート信号
DBGEXTEST[a]	Output	EXTEST ステート信号
DBGnTDOEN	Output	テストデータアウトイネーブル
DBGnTRST	Input	テストリセット（ローアクティブ信号） Low の時、この信号は JTAG インタフェースをリセットします。
DBGTCKEN	Input	テストクロックイネーブル
DBGTDI	Input	テストデータイン JTAG テストデータイン信号

表 A-3 JTAG およびテスト信号の説明（続き）

名前	タイプ	説明
DBGTDO	Output	**テストデータアウト** JTAG **テストデータアウト信号**
DBGTMS	Input	**テストモードセレクト** JTAG **テストモード選択信号**

a. これらの信号は、スキャンチェイン 0 が選択されている場合にのみアクティブになります。

A.4 デバッガ信号

デバッガ信号の説明を表 A-4 に示します。

表 A-4 デバッガ信号の説明

名前	タイプ	説明
DBGBREAK	Input	**ブレークポイント** **この信号によって、外部ハードウェアはデバッグを行うためにプロセッサの実行を停止することができます。**High **の時、この信号によって現在のメモリアクセスにブレークポイントが設定されます。メモリアクセスが命令フェッチの場合、命令がコアパイプラインの実行段階に達すると、コアはデバッグ状態に入ります。メモリアクセスがデータフェッチの場合、コアは現在の命令が実行を完了した後にデバッグ状態になります。これによって、**EmbeddedICE-RT **モジュールが提供する内部ブレークポイントが実行されます。** **ほとんどのシステムでは、この入力は** Low **に固定されています。**
COMMRX	Output	**通信受信フル** High **の時、この信号は通信チャネル受信バッファにコアが読み取るデータが格納されていることを示しています。**
COMMTX	Output	**通信送信エンプティ** High **の時、この信号は通信チャネル送信バッファがエンプティになっていることを示しています。**
DBGACK	Output	**デバッグ応答** High **の時、この信号は** ARM **がデバッグ状態にあることを示しています。**
DBGEN	Input	**デバッグイネーブル** Low **の時、プロセッサのデバッグ機能をディセーブルにするスタティック構成信号です。** EmbeddedICE **ロジックを機能させるには、この信号は** High **でなければなりません。**
DBGRQ	Input	**デバッグ要求** **この信号によって、コアは現在の命令を実行した後デバッグ状態になります。これにより、**EmbeddedICE-RT **ロジックが提供するデバッグ機能に加えて、外部ハードウェアはコアを強制的にデバッグ状態にさせることができます。** **ほとんどのシステムでは、この入力は** Low **に固定されています。** **DBGRQ は、DBGACK がアサートされる同じクロックでディアサートされなければなりません。**
DBGEXT[1:0]	Input	**外部状態** **これらの信号により、ブレークポイントとウォッチポイントは外部状態に依存することができます。**

表 A-4 デバッガ信号の説明

名前	タイプ	説明
DBGRNG[1:0]	Output	レンジアウト これらの信号は、該当の EmbeddedICE-RT ウォッチポイントレジスタが、現在アドレス、データ、および制御バスに存在するデータとマッチしていることを示しています。これらの信号は、ウォッチポイントイネーブル制御ビットの状態から独立しています。

A.5 組み込み型トレースマクロセルインタフェース信号

ETM インタフェース信号を表 A-5 に示します。

表 A-5 ETM インタフェース信号の説明

出力名	タイプ	説明
ETMnMREQ	Output	メモリ要求（ローアクティブ信号）。Low の時、この信号はプロセッサが次のサイクル時にメモリアクセスを要求することを示しています。
ETMSEQ	Output	シーケンシャルアドレス。High の時、次のメモリサイクルのアドレスが最後のメモリサイクルのアドレスと関連があることを示しています。新しいアドレスは、以下のアドレスのいずれか 1 つです。 • 以前のアドレスと同じ • ARM 状態で 4 つ大きい • Thumb 状態で 2 つ大きい この信号は、次のサイクルで高速メモリモードを使用してアドレス変換システムをバイパスすることができることを示すために下位アドレスラインで使用することができます。
ETMnEXEC	Output	実行（ローアクティブ信号）。High の時、実行ユニット内の命令がまだ実行されていないことを示しています。たとえば、条件チェックコードでフェイルしている場合もあります。
ETMnCPI	Output	コプロセッサ命令（ローアクティブ信号）。ARM720T プロセッサがコプロセッサ命令を実行すると、ETMnCPI を Low にして、コプロセッサからの応答を待ちます。実行される処置は、CPA や CPB 入力のコプロセッサ信号であるこの応答に依存します。
ETMADDR[31:0]	Output	アドレス。これは、リタイムされた内部アドレスバスです。
ETMnOPC	Output	オペコードフェッチ（ローアクティブ信号）。Low の時、プロセッサがメモリから命令をフェッチ中であることを示しています。High の時、データが存在している場合はそれを転送中であることを示しています。
ETMDBGACK	Output	デバッグ応答。High の時、プロセッサがデバッグ状態にあることを示しています。Low の時、プロセッサが通常のシステム状態にあることを示しています。
ETMABORT	Output	メモリアボートまたはバスエラー。要求されたアクセスが禁止されていることを示しています。
ETMCPA	Output	コプロセッサ不在ハンドシェイク。コプロセッサ不在信号です。この信号は、バッファされたコプロセッサ不在信号です。
ETMCPB	Output	コプロセッサビジーハンドシェイク コプロセッサビジー信号。この信号は、バッファされたコプロセッサビジー信号です。
ETMPROCID[31:0]	Output	トレース PROCID バス
ETMPROCIDWR	Output	トレース PROCID ライト。トレース PROCID 、すなわち CP15 レジスタ c13 に書き込みが行われたことを ETM7 に知らせます。

表 A-5 ETM インタフェース信号の説明（続き）

出力名	タイプ	説明
ETMTBIT	Output	Thumb 状態 High の時、この信号はプロセッサが Thumb 命令セットを実行中であることを示しています。Low の時、プロセッサは ARM 命令セットを実行中です。
ETMBIGEND	Output	ビッグエンディアン形式 この信号が High の時、プロセッサはメモリ内のバイトをビッグエンディアン形式で扱います。Low の時、メモリはリトルエンディアン形式で処理されます。
ETMEN	Input	ETM7 イネーブル信号
ETMHIVECS	Output	Low の時、この信号は例外ベクタがアドレス 0x00000000 から始まることを示します。High の時、例外ベクタはアドレス 0xFFFF0000 から始まります。
ETMSIZE[1:0]	Output	ARM720Tプロセッサによって駆動されるバスのメモリアクセスサイズ
ETMRDATA[31:0]	Output	プロセッサリードデータバス
ETMWDATA[31:0]	Output	プロセッサライトデータバス
ETMINSTRVALID	Output	ARM720T プロセッサで駆動された命令の有効信号です。High の時、実行段階の命令が有効であり、フラッシュされていないことを示しています。
ETMnRW	Output	リード／ライト（ローアクティブ信号）。High の時、プロセッサライトサイクルであることを示します。Low の時、プロセッサリードサイクルであることを示します。
ETMCLKEN	Output	この信号は、コアがウェイト状態にあることを ETM に知らせるために使用されます。この信号は、ETM に対する真のクロックイネーブルではありません。

A.6 ATPG テスト信号

ARM720T プロセッサが使用する ATPG テスト信号を表 A-6 に示します。

表 A-6 ATPG テスト信号の説明

名前	タイプ	説明
TESTENABLE	Input	この信号は、スキャンテスト時クロックが確実にフリーランニングするようにします。 TESTENABLE は、以下の通りでなければなりません。 • スキャンテスト期間を通して High に固定。 • 機能モード時は Low に固定。
SCANENABLE	Input	この信号により、スキャンチェインを介してベクタをシリアルにシフトさせることができます。この信号は、I/O ピンを使用して制御してください。機能モード時、この信号は Low に固定していなければなりません。
SCANIN0-SCANIN6	Inputs	プロセッサコアスキャンチェイン入力
SCANOUT0-SCANOUT6	Outputs	プロセッサコアスキャンチェイン出力
HCLK	Input	システムクロック。すべての信号は、HCLK の立ち上がりエッジと関連があります。
HCLKEN	Input	AHB 転送の同期イネーブル。High の時、HCLK の次の立ち上がりエッジが、ARM720T プロセッサが組み込まれた AHB システムの立ち上がりエッジでもあることを示しています。この信号は、AMBA バスとコアを同じ周波数にするシステムでは High に固定してください。
DBGTCKEN	Input	デバッグロジックの同期イネーブル。スキャンテスト時、この信号は High に固定しなければなりません。
HRESETn	Input	この信号は、システムおよびバス用のアクティブ Low リセット信号です。
DBGnTRST	Input	この信号は、内部状態のアクティブ Low リセット信号です。この信号は、非同期リセットレベル入力です。

A.7 その他の信号

ARM720T プロセッサが使用するその他の信号を、表 A-7 に示します。

表 A-7 その他の信号の説明

名前	タイプ	説明
BIGENDOUT	Output	ビッグエンディアン形式 この信号が High の時、プロセッサはメモリ内のバイトをビッグエンディアン形式で扱います。Lowの時、メモリはリトルエンディアン形式で処理されます。
nFIQ	Input	ARM 高速割り込み要求信号（ローアクティブ信号）
nIRQ	Input	ARM 割り込み要求信号（ローアクティブ信号）
VINITHI	Input	リセット時、CP15 レジスタ c1 の V ビットの状態を決定します。High の時、V ビットはリセット終了時にセットされます。Low の時、リセット終了時に V ビットはクリアされます。

このページは白紙です。

用語集

用語集

この用語集は、このマニュアルで使用している用語について解説します。用語に複数の意味がある場合は、ここで説明する意味を意図しています。

0 でなければならないフィールド（SBZ）

ソフトウェアによって 0（またはビットフィールドの場合はすべて 0）がライトされなければなりません。0 以外の値がライトされると、予測不能な結果になります。

「1 でなければならないフィールド（SBO）」も参照してください。

1 でなければならないフィールド（SBO）

ソフトウェアによって 1（またはビットフィールドの場合はすべて 1）がライトされなければなりません。1 以外の値がライトされると、予測不能な結果になります。

「0 でなければならないフィールド（SBZ）」も参照してください。

ALU 「算術論理ユニット」を参照してください。

ARM 状態 ARM（32 ビット）命令を実行しているプロセッサは、ARM 状態で動作します。

CISC 「複雑命令セットコンピュータ（CISC）」を参照してください。

CPSR 「プログラムステータスレジスタ」を参照してください。

DCC デバッグ通信チャンネル

EmbeddedICE EmbeddedICE ロジックは、MultiICE のようなプロトコルコンバータを使用して、JTAG テストアクセスポートを介して制御されます。ソフトウェアツールに、ターゲットプロセッサで実行するコードのデバッギングを行わせる追加のハードウェア。

「ICE」、「JTAG」も参照してください。

EmbeddedICE-RT

リアルタイムデバッギングのサポートを向上させた EmbeddedICE ロジックのバージョン

FIQ 高速割り込み

ICE 「インサーキットエミュレータ」を参照してください。

IRQ 割り込み要求

JTAG 「Joint Test Action Group」を参照してください。

Joint Test Action Group

IEEE 1149.1 規格を作成した組織の名称。この規格は、集積回路のインサーキットテストに使用するバウンダリスキャンアーキテクチャを定義しています。

LR 「リンクレジスタ」を参照してください。

MMU 「メモリ管理ユニット」を参照してください。

PC 「プログラムカウンタ」を参照してください。

PSR 「プログラムステータスレジスタ」を参照してください。

RAZ ゼロとしてリードされます。

RISC 「縮小命令セットコンピュータ（RISC）」を参照してください。

SBO 「1 でなければならないフィールド（SBO）」を参照してください。

SBZ 「0 でなければならないフィールド（SBZ）」を参照してください。

SP 「スタックポインタ」を参照してください。

SPSR 「保存プログラムステータスレジスタ（SPSR）」を参照してください。

SWI 「ソフトウェア割り込み命令」を参照してください。

TAP 「テストアクセスポート（TAP）」を参照してください。

Thumb 状態 Thumb（16 ビット）命令を実行するプロセッサは、Thumb 状態で動作します。

Thumb 命令 Thumb 状態の ARM プロセッサが実行する動作を指定するハーフワードです。Thumb 命令はハーフワード整列されていなければなりません。

UND 「未定義」を参照してください。

UNP　「予測不能」を参照してください。

アドレッシングモード

命令で使用される値を生成するために多くの異なる命令で共用されるプロシージャ。4 種類の ARM アドレッシングモードについては、生成される値はメモリアドレスです（これは、アドレッシングモードの従来の規則です）。5 番目のアドレッシングモードは、データ処理命令によってオペランドとして使用される値を生成します。

アボート　不正なメモリアクセスによって発生します。アボートは、外部メモリシステム、外部 MMU、または EmbeddedICE-RT ロジックによって引き起こされます。

インサーキットエミュレータ

インサーキットエミュレータ（ICE）とは、ハードウェアやソフトウェアのデバッギングを支援するデバイスのことです。ARM720T プロセッサなどのデバッグ可能な ARM プロセッサには、この処理を支援するための追加ハードウェアがあります。

「EmbeddedICE-RT」も参照してください。

ウォッチポイント

モニタされているイメージ内の位置。そこに保存されている値が変化すると、デバッガはイメージの実行を停止します。

「ブレークポイント」も参照してください。

外部アボート　外部メモリシステムによって生成されるアボート

カレントプログラムステータスレジスタ

「プログラムステータスレジスタ」を参照してください。

算術論理ユニット

加算、乗算、およびすべての比較演算などのあらゆる算術演算を実行するコンピュータの一部分。

縮小命令セットコンピュータ（RISC）

複雑命令セットコンピュータと比べて少ない数の命令を認識するタイプのマイクロプロセッサです。RISC アーキテクチャの利点は、以下の通りです。

- 命令が極めて簡単ですので、命令を非常に高速に実行させることができます。
- 少ない個数のトランジスタしか必要としないため、製造費が安く、よりパワー効率に優れています。

「複雑命令セットコンピュータ（CISC）」も参照してください。

スタックポインタ

スタックの一番上を指し示すレジスタまたは変数です。スタックがフルスタックの場合、SP はプッシュされた最新のアイテムを指し示します。スタックが空の場合、SP は次のアイテムがプッシュされてくる先頭の空位置を指し示します。

ステータスレジスタ

「プログラムステータスレジスタ」を参照してください。

制御ビット

プログラムステータスレジスタの下位 8 ビット。例外が起こると、制御ビットは変化します。このビットは、プロセッサが特権モードになっているときにのみソフトウェアで変更することができます。

ソフトウェア割り込み命令

この命令（SWI）はスーパーバイザモードに入り、特定のオペレーティングシステム機能を要求します。

テストアクセスポート（TAP）

入出力を形成し、JTAG バウンダリスキャンアーキテクチャへのインタフェースを制御する 4 つの必須端子と 1 つのオプション端子の集まりです。必須端子とは **TDI**、**TDO**、**TMS**、および **TCK** です。オプション端子とは **nTRST** です。

デバッガ

ソフトウェアデバッギングをサポートするカスタムハードウェアとともに、ソフトウェアフォールトを検出、位置決めし、修復するために使用されるプログラムをもったデバッギングシステム。

デバッグ状態

プロセッサの実行をモニタリングし、制御することができる状態。通常、アプリケーションプログラムフローのエラーを見つけるために使用されます。プロセッサは*ホルトモード*からデバッグ状態になりますが、*モニタモード*からはなりません。

特権モード

ユーザモード以外のプロセッサモード。一般に、メモリシステムは、より制限のあるユーザアクセス許可ではなくスーパーバイザアクセス許可に対して特権モードからのメモリアクセスをチェックします。一部の命令は、その使用が特権モードに制限されています。

バンクレジスタ

物理レジスタが現在のプロセッサモードで定義されているレジスタ番号。バンクレジスタは、プロセッサモードによってレジスタ r8 ～ r14 またはレジスタ r13 ～ r14 になります。

ビッグエンディアン

ワードの最下位バイトが最上位バイトよりも大きなアドレスに置かれるメモリ構成。

複雑命令セットコンピュータ（CISC）

多数の命令を認識するマイクロプロセッサです。

「縮小命令セットコンピュータ（RISC）」も参照してください。

ブレークポイント

プログラム内の位置。プログラムの実行がこの位置までくると、デバッガはコードイメージの実行を停止します。

「ウォッチポイント」も参照してください。

プログラムカウンタ

レジスタ 15、すなわちプログラムカウンタは、現在の命令の後の 2 命令のポインタとしてほとんどの命令で使用されます。

プログラムステータスレジスタ

現在のプログラムの情報および現在のプロセッサの情報を格納しています。プロセッサステータスレジスタとも言われます。

また、*カレント PSR*（CPSR）とも呼ばれます。この PSR と *保存 PSR*（SPSR）との区別を重視するときにこのように呼ばれます。SPSR は、現在の機能が呼び出されたときに PSR がもっていた値を保持します。この値は、制御が戻るとリストアされます。

プロセッサステータスレジスタ

「プログラムステータスレジスタ」を参照してください。

冪等（idempotent）

特定のバイナリ演算がそれ自体に適用されたときに、そのものに等しくなるという数学的量

保存プログラムステータスレジスタ

現在のプロセッサモードと関連のある保存プログラムステータスレジスタです。ユーザモードやシステムモードのように、保存プログラムステータスレジスタがない場合には定義されません。

「プログラムステータスレジスタ」も参照してください。

ホルトモード

2 つのデバッギングモードのうちの 1 つ。デバッギングをホルトモードで実行すると、コアはウォッチポイントやブレークポイントを検出すると停止し、システムの残りの部分から切り離されます。*「モニタモード」*も参照してください。

マクロセル

定義済みインタフェースと動作をもった複雑な論理ブロックです。一般的な VLSI システムは、いくつかのマクロセル（ARM7TDMI-S コア、ETM7、メモリブロックなど）とアプリケーション固有のロジックから構成されます。

未定義

未定義命令トラップを生成する命令を指します。

メモリ管理ユニット

メモリシステムの管理を可能にします。多くのメモリ管理は、メモリ内の変換テーブルを介して行われます。

モニタモード

2 つのデバッギングモードのうちの 1 つ。デバッギングをモニタモードで実行すると、コアはウォッチポイントやブレークポイントを検出しても停止せず、代わりにアボート例外ルーチンを実行します。「ホルトモード」も参照してください。

予測不能

命令の結果が信頼できないことを意味しています。予測不能な命令は、プロセッサやシステムのいかなる部分も停止させたり、ハングさせたりしてはいけません。

予測不能フィールド

有効なデータは格納されておらず、値は始終、命令ごとに、およびインプリメンテーションごとに変化します。

リトルエンディアンメモリ

ワードの最上位バイトが最下位バイトよりも大きなアドレスに置かれるメモリ構成。

リンクレジスタ

このレジスタは、リンク命令付きの分岐後の次の命令のアドレスを保持しています。

例外モード

特定の例外が発生したときになる特権モード

例外

イベントを処理します。たとえば、例外は外部割込みや未定義命令を扱うことができます。

索引

索引

項目はアルファベット順，50 音順で配列されています。項目に続く数字はページ数を示しています。

そ

た

つ

て

と

な

は

ひ

ふ

へ

ほ

ま

み

ARM720T Revision 4
(AMBA AHB Bus Interface Version)
コアCPUマニュアル

セイコーエプソン株式会社
電子デバイス営業本部

〈ED東日本営業部〉
東　京　　〒191-8501　東京都日野市日野421-8
TEL（042）587-5313（直通）　FAX（042）587-5116

仙　台　　〒980-0013　宮城県仙台市青葉区花京院1-1-20　花京院スクエア19F
TEL（022）263-7975（代表）　FAX（022）263-7990

〈ED西日本営業部〉
大　阪　　〒541-0059　大阪市中央区博労町3-5-1　エプソン大阪ビル15F
TEL（06）6120-6000（代表）　FAX（06）6120-6100

名古屋　　〒461-0005　名古屋市東区東桜1-10-24　栄大野ビル4F
TEL（052）953-8031（代表）　FAX（052）953-8041

インターネットによる電子デバイスのご紹介　http://www.epsondevice.com/domcfg.nsf

ドキュメントコード：405018400
2004年 6月作成