セクション 2. M4K® コア搭載デバイス用 CPU

ハイライト

本セクションには以下の主要項目を記載しています。

2.1 はじめに 2-2
2.2 アーキテクチャの概要 2-3
2.3 PIC32 CPU の詳細 2-6
2.4 CP0 レジスタに書き込む場合の注意点 2-11
2.5 アーキテクチャ リリース 2 の詳細 2-12
2.6 2 つの CPU バス 2-12
2.7 内部システムバス 2-13
2.8 セット / クリア / 反転 2-13
2.9 ALU ステータスビット 2-14
2.10 割り込みおよび例外機構 2-14
2.11 プログラミング モデル 2-14
2.12 コプロセッサ 0 (CP0) レジスタ 2-21
2.13 MIPS16e® の実行 2-55
2.14 メモリモデル 2-55
2.15 CPU 命令の機能による分類 2-56
2.16 CPU の初期化 2-59
2.17 リセットの影響 2-60
2.18 関連アプリケーション ノート 2-61
2.19 改訂履歴 2-62

Note:	本セクションは、デバイス データシートの内容の補足を目的としています。本セクションの内容は、PIC32 ファミリの一部のデバイスには対応していません。 本書の内容がお客様のお使いになるデバイスに対応しているかどうかは、各デバイスの最新のデータシートで**「CPU」**の章の冒頭に記載されている注意書きを参照してください。 デバイス データシートとファミリ リファレンス マニュアルの各セクションは、Microchip社のウェブサイト http://www.microchip.comからダウンロードできます。

2.1 はじめに

PIC32 MCU は、MIPS® Technologies 社が提供する M4K® マイクロプロセッサ コアベースの複合 SoC (system-on-a-chip) です。M4K® は、拡張 MIPS32® リリース 2 命令セット アーキテクチャ (ISA) を備えた最先端の低消費電力型 32 ビット RISC プロセッサコアです。

本書では、M4K® プロセッサコア ベースの PIC32 ファミリ マイクロコントローラの CPU 機能とシステム アーキテクチャの概要を示します。

2.1.1 主な特長

- 最大 1.5 DMIPS/MHz の性能
- フラッシュメモリからの実行を強化するプログラマブルなプリフェッチ キャッシュメモリ (この機能を備えていないデバイスもあります。各デバイスのデータシートを参照してください)
- コンパクトなコードを実現する 16 ビット命令モード (MIPS16e®)
- 最大 96 の割り込み要因を備えたベクタ割り込みコントローラ
- プログラマブルなユーザ動作モードとカーネル動作モード
- 周辺モジュール レジスタのアトミックなビット操作 (シングルサイクル)
- クロックごとに 32 x 16 乗算が可能な最大発行レートを備えた乗除算ユニット
- ハードウェアベースの非侵入型データ監視機能とアプリケーション データ ストリーミング機能を備えた高速 Microchip 社 ICD ポート
- サードパーティ製デバッグ、プログラミング、テスト用ツールに幅広く対応する EJTAG デバッグポート
- 命令による電源管理モードの制御
- 5 段の命令実行パイプライン
- 内部コード保護機能による知的財産保護の強化

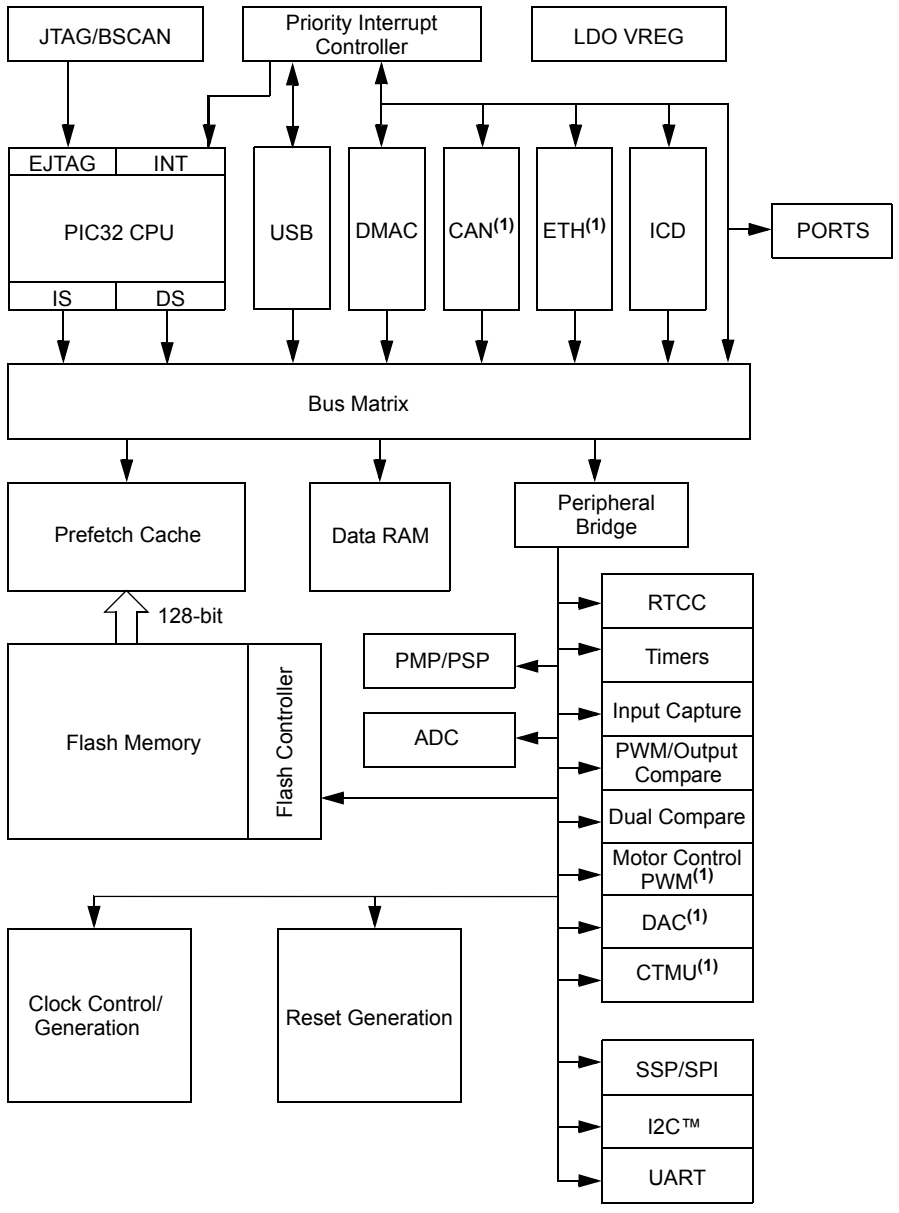
2.1.2 MIPS® 関連の文書

- MIPS32® M4K® プロセッサコア ソフトウェア ユーザマニュアル - MD00249-2B-M4K-SUM
- MIPS® 命令セット - MD00086-2B-MIPS32BIS-AFP
- MIPS16e® - MD00076-2B-MIPS1632-AFP
- MIPS32® 特権リソース アーキテクチャ - MD00090-2B-MIPS32PRA-AFP

2.2 アーキテクチャの概要

PIC32 ファミリは、豊富な機能を備えた複合 SoC です。PIC32 ファミリの全てのプロセッサは、高性能な RISC CPU を搭載し、32 ビットモードと 16 ビットモードでプログラミング可能であり、また両方のモードを混用する事も可能です。PIC32 は、高性能割り込みコントローラ、DMA コントローラ、USB コントローラ、インサーキット デバッガ、高性能スイッチング マトリクス (周辺モジュールへの高速なデータアクセスを提供)、内蔵データ RAM (データとプログラムを格納) を備えます。フラッシュメモリ用にユニークなプリフェッチ キャッシュとプリフェッチ バッファを備える事により、フラッシュのレイテンシを隠し、ゼロ待機ステート相当の性能を提供します。

図 2-1: PIC32 ブロック図



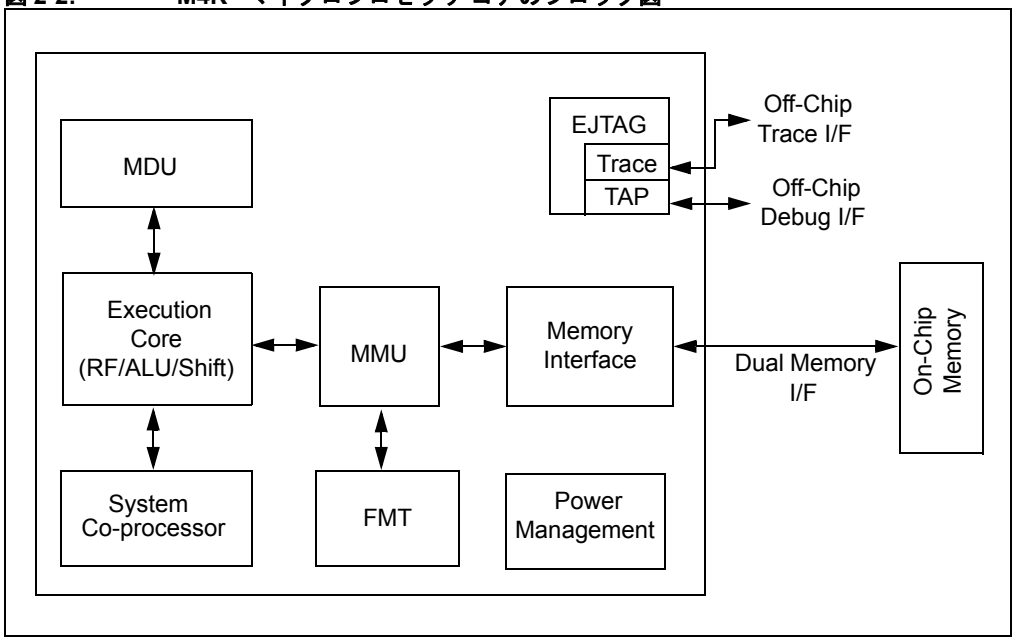
Note 1: デバイスによってはこの周辺モジュールを内蔵していません。詳細は各デバイスのデータシートを参照してください。

全ての周辺モジュールへの接続用に、PIC32 は 2 つの内部バスを備えます。大部分の周辺モジュール ユニットは、メインの周辺モジュールバスを使って、周辺モジュール ブリッジ経由でバスマトリクスに接続されます。割り込みコントローラ、DMA コントローラ、インサーキット デバッガ、USB モジュールを接続する、高速モジュール ブリッジもあります。

一部の PIC32 MCU では M4K® CPU コアが、その心臓部として機能します。この CPU は、プログラムによる制御の下に演算を実行します。命令は CPU によってフェッチされ、同期してデコードおよび実行されます。命令は、プログラム フラッシュメモリまたはデータ RAM のどちらかに格納されます。

PIC32 CPU はロード / ストア アーキテクチャに基づき、一連の内部レジスタを使ってほとんどの演算を実行します。これらの内部レジスタ間のデータ移動と外部とのデータ移動には、専用のロード命令とストア命令を使います。

図 2-2: M4K® マイクロプロセッサ コアのブロック図



2.2.1 バス

PIC32 は 2 つの独立したバスを備えます。一方のバスは CPU への命令のフェッチに使い、もう一方のバスはロードとストア命令用のデータパスとして使います。命令バス (I バス) とデータバス (D バス) は、どちらもバスマトリクス ユニットに接続されています。バスマトリクスはスイッチとして機能し、システム内で複数のアクセスが同時に発生する事を可能にします。バスマトリクスは、互いに異なるターゲットにアクセスを試みる複数のバスマスタからの同時アクセスを可能にします。複数のバスマスタが同一ターゲットに同時にアクセスを試みた場合、バスマトリクスは調停アルゴリズムに基づいてそれらのアクセスを順次処理します。

CPU は、バスマトリクスに接続した 2 種類のデータパスを有するため、実質的にシステムに対して 2 つの異なるバスマスタとして機能します。フラッシュメモリから命令を実行する場合、SRAM と内部周辺モジュールに対するロードおよびストア動作は、フラッシュメモリからの命令フェッチと並列に発生します。

デバイスによっては、CPU に加えて以下もバスマスタとして機能します。

- DMA コントローラ
- インサーキット デバッガ (ICD) ユニット
- USB コントローラ
- CAN コントローラ
- Ethernet コントローラ

2.2.2 プログラミング モデルの概要

PIC32 プロセッサは以下の特長を備えます。

- 5 段のパイプライン
- 32 ビットアドレスおよびデータパス
- DSP のような積和および積差命令 (MADD、MADDU、MSUB、MSUBU)
- デスティネーションに汎用レジスタを指定する乗算命令 (MUL)
- 0 および 1 検出命令 (CLZ、CLO)
- 待機命令 (WAIT)
- 条件付き移動命令 (MOVZ, MOVN)
- MIPS32® 拡張アーキテクチャ (リリース 2) を実装
- ベクタ割り込み
- プログラマブルな例外ベクタベース
- アトミックな割り込みの有効化 / 無効化
- 汎用レジスタ (GPR) シャドーセット
- ビットフィールド操作命令
- MIPS16e® 特定アプリケーション向け拡張機能によるコード密度の向上
- 特殊な PC 相対命令によるアドレスと定数の効率的なロード
- データ型変換命令 (ZEB、SEB、ZEH、SEH)
- コンパクトなジャンプ
- スタックフレームをセットアップ、分解する SAVE および RESTORE マクロ命令
- シンプルな FMT (Fixed Mapping Translation) 機能を備えたメモリ管理ユニット
- プロセッサとコプロセッサ レジスタ間の双方向データ転送
- メモリとコプロセッサ レジスタ間の双方向直接データ転送
- 乗除算ユニットの性能最適化 (高性能ビルド時オプション)
- 32 x 16 乗算の最大発行レート : 1 クロックあたり 1 回
- 32 x 32 乗算の最大発行レート : 1 クロックおきに 1 回
- Early-in 除算制御 - 11 ～ 34 クロックのレイテンシ
- 低消費電力モード (WAIT 命令で開始)
- SDBBP 命令を使ったソフトウェア ブレークポイント

2.2.3 コアタイマ

PIC32 アーキテクチャは、アプリケーション プログラムから利用できるコアタイマを備えています。このタイマは、2 個のコプロセッサ レジスタ (Count レジスタと Compare レジスタ) の形態で実装されています。Count レジスタは、システムクロック (SYSCLK) の 2 サイクルごとにインクリメントします。デバッグモードでは、必要に応じて Count レジスタのインクリメントを中止できます。Compare レジスタは、タイマ割り込みを生成するために使います。Compare レジスタの内容が Count レジスタの内容に一致した時に割り込みが生成されます。割り込みは、割り込みコントローラ モジュール内で有効にされている場合にのみ処理されます。

コアタイマの詳細は、本書の **2.12「コプロセッサ 0 (CP0) レジスタ」**と、『PIC32 ファミリ リファレンス マニュアル』**のセクション 8.「割り込み」**(DS61108) を参照してください。

2.3 PIC32 CPU の詳細

2.3.1 パイプライン段

パイプラインは下記の 5 段で構成されます。

- 命令 (I) 段
- 実行 (E) 段
- メモリ (M) 段
- アライン (A) 段
- 書き戻し (W) 段

2.3.1.1 I 段 - 命令フェッチ

I 段の動作 :

- 命令 SRAM から 1 個の命令をフェッチする
- MIPS16e® 命令を MIPS32® の類似命令に変換する

2.3.1.2 E 段 - 実行

E 段の動作 :

- レジスタファイルからオペランドをフェッチする
- M および A 段からのオペランドをこの段にバイパスする
- 算術論理ユニット (ALU) は、レジスタ間命令の算術または論理演算を始める
- ALU がロードおよびストア命令用のデータ仮想アドレスを求め、MMU が仮想アドレスから物理アドレスへの固定変換を実行する
- ALU は分岐条件の真偽を判定し、分岐命令の分岐先仮想アドレスを求める
- 命令ロジックが命令アドレスを選択し、MMU が仮想アドレスから物理アドレスへの固定変換を実行する
- 全ての乗除演算はこの段で始まる

2.3.1.3 M 段 - メモリフェッチ

M 段の動作 :

- ALU による算術または論理演算が完了する
- ロードおよびストア命令のデータ SRAM アクセスを実行する
- 16 x 16 または 32 x 16 MUL 演算は配列内で完了し、M 段内で 1 クロック分ストールし、M 段内でキャリー伝播加算を完了する
- 32 x 32 MUL 演算は、M 段内で 2 クロック分ストールし、M 段内で配列の第 2 サイクルとキャリー伝播加算を完了する
- MDU で乗算と除算を実行し、IU が命令を M 段から移動する前に計算が完了した場合、IU が命令を A 段に移動するまでの間、MDU は結果をテンポラリ レジスタに保持する (結果は失われません)

2.3.1.4 A 段 - アライン

A 段の動作 :

- ロード用アライナが、ロードしたデータを、そのワード境界で位置合わせする
- MUL 演算が結果を書き戻し可能にする (実際のレジスタ書き戻しは W 段で実行)
- この段から、MDU からロードしたデータまたは得られた結果をバイパスして E 段で利用できる

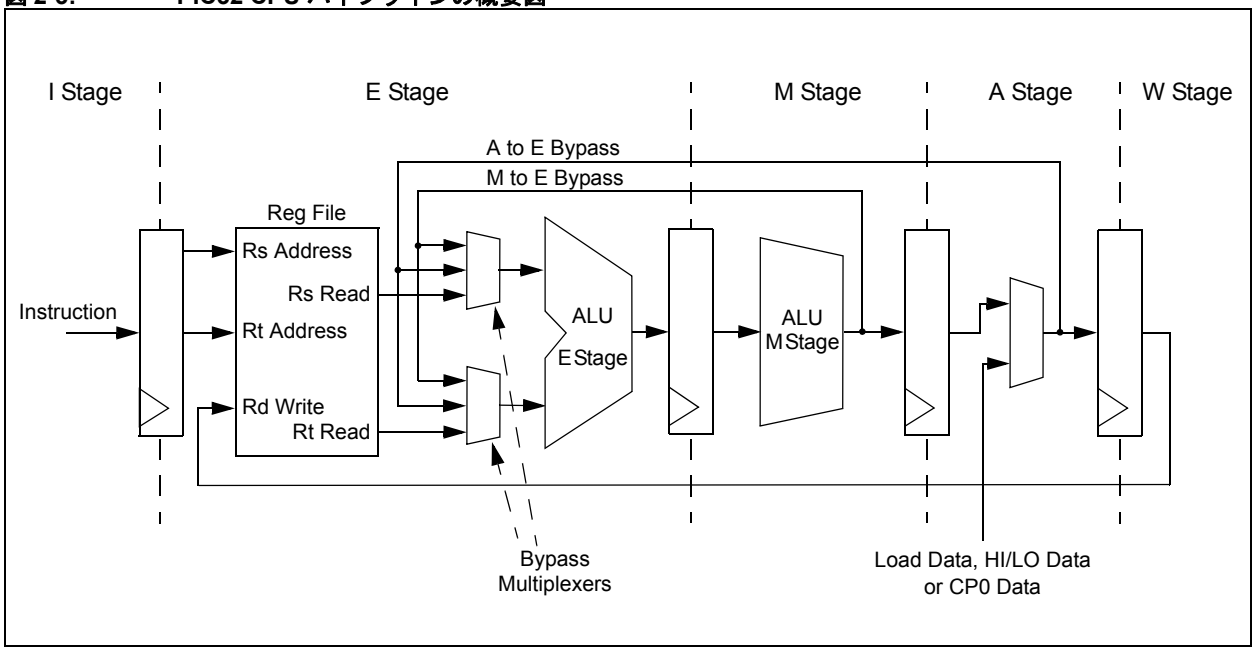
2.3.1.5 W 段 - 書き戻し

W 段の動作 :

レジスタ間命令またはロード命令の結果をレジスタファイルに書き戻す

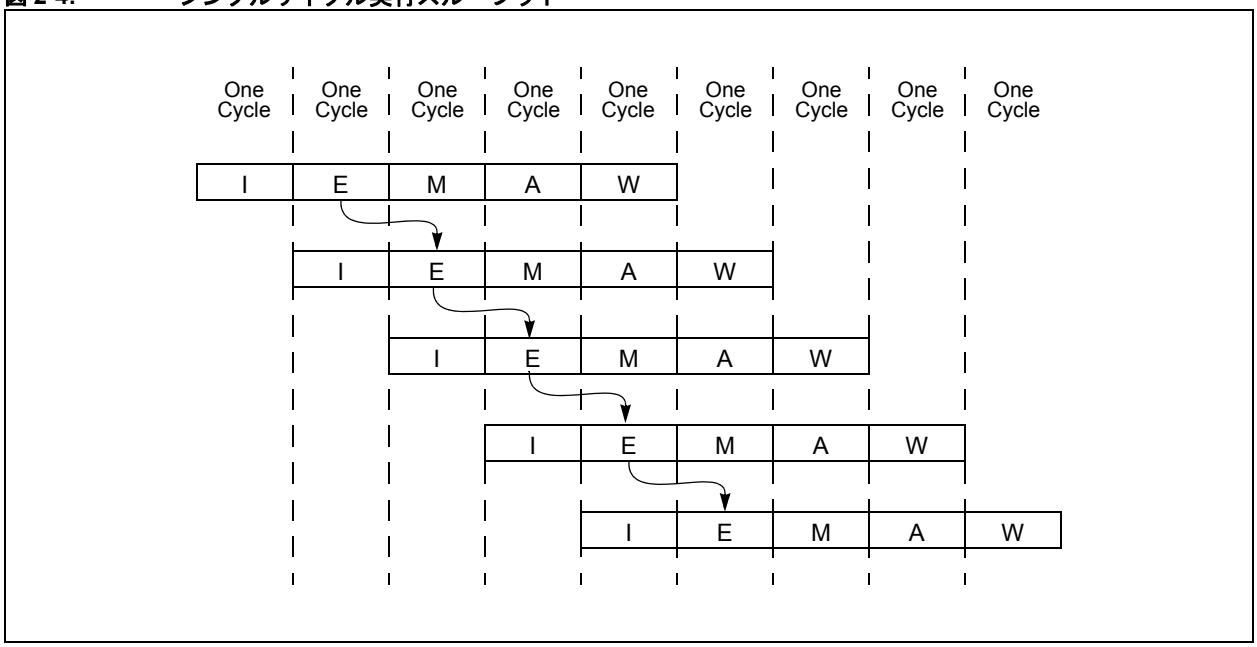
M4K® マイクロプロセッサ コアは「バイパス」機構を備えています。この機構を使うと、演算結果を一度レジスタに書き込んでから読み戻すといった操作なしで、その結果を必要とする別の命令に直接渡す事ができます。

図 2-3: PIC32 CPU パイプラインの概要図



PIC32 コア内の命令パイプラインを使う事により、高速なシングルサイクル命令実行環境を実現しています。

図 2-4: シングルサイクル実行スループット



2.3.2 実行ユニット

PIC32 の実行ユニットは、MIPS® 命令セットの大部分の命令の処理を受け持ちます。実行ユニットは、パイプライン化された実行により、大部分の命令にシングルサイクル スループットを提供します。パイプライン化された実行は「パイプライン処理」と呼ばれ、複雑な演算を複数の「段」に細分化します。一連の処理段は、複数クロックサイクルにまたがって実行されます。

実行ユニットは以下の機能を備えます。

- 32 ビット加算器 - データアドレスの計算に使用
- アドレスユニット - 次の命令アドレスの計算に使用
- ロジック - 分岐の条件判定と分岐先アドレスの計算に使用
- ロード用アライナ
- バイパス マルチプレクサ - データを生成する命令の直後にその結果を使う命令が続く場合、ストールを回避するために使用
- 先頭 0/1 検出ユニット - CLZ および CLO 命令の実装に使用
- 算術論理ユニット (ALU) - ビットワイズ論理演算の実行に使用
- シフタとストア用アライナ

2.3.3 MDU

MDU(乗算 / 除算ユニット) は、乗除演算を実行します。MDU は 32 x 16 乗算器、結果累積レジスタ (HI と LO)、乗算および除算ステートマシン、これらの機能の実行に必要な全てのマルチプレクサと制御ロジックにより構成されます。パイプライン化された高性能 MDU は、1 クロックサイクルに 1 回の 16 x 16 または 32 x 16 乗算演算を実行できます。32 x 32 乗算演算は 1 クロックサイクルおきに 1 回発行できます。連続した 32 x 32 乗算演算の発行をストールするために、適切なインターロックを実装しています。除算演算は、クロックあたり 1 ビットの単純な繰り返しアルゴリズムを使って実装され、最悪条件では演算の完了に 35 クロックサイクルを要します。アルゴリズムの初期で被除数の符号拡張を検出し、被除数が 24、16、8 ビットのどれかである場合、除算器はビット数に応じて 32 回の繰り返しの中の 7、15、23 回をスキップします。除算がまだ実行中である時に、後続の MDU 命令の発行が試みられると、除算演算が完了するまでパイプライン ストールが発生します。

M4K® マイクロプロセッサ コアは、追加の乗算命令として、MUL 命令を備えています。この命令は、乗算結果の下位 32 ビットを HI/LO レジスタペアではなくレジスタファイルに格納するように指示します。この命令は、LO レジスタを使う場合に必要となる LO レジスタからの明示的な移動命令 (MFLO) を不要とし、複数のデスティネーション レジスタをサポートする事により、乗算を多用する演算のスループットを改善します。積和演算には MADD/MADDU 命令、積差演算には MSUB/MSUBU 命令を使います。MADD 命令は、2 つのオペランドを乗算した後に、積を HI と LO レジスタの現在の内容に加算します。同様に、MSUB 命令は、2 つのオペランドを乗算した後に、積を HI および LO レジスタの内容から減算します。MADD/MADDU および MSUB/MSUBU 演算は、デジタルシグナル プロセッサ (DSP) アルゴリズムで一般的に使われます。

2.3.4 シャドーレジスタ セット

PIC32 プロセッサは、高優先度割り込み用に汎用レジスタ (GPR) のコピーを実装しています。このレジスタバンクは、シャドーレジスタ セットと呼びます。高優先度の割り込みが発生すると、プロセッサはソフトウェアからの介入なしで自動的にシャドーレジスタに切り換えます。これにより、割り込みハンドラでのオーバーヘッドが減少し、実効レイテンシを低減できます。

シャドーレジスタ セットは、システム コプロセッサ (CP0) 内のレジスタと、CPU コアの外部にある割り込みコントローラ ハードウェアにより制御されます。

シャドーレジスタ セットの詳細は**セクション 8.「割り込み」** (DS61108) を参照してください。

2.3.5 パイプラインのインターロック処理

データ依存性またはこれに類似する外部条件が原因で、パイプラインのどこかの段で命令処理の進行が妨げられると、円滑なパイプライン フローが割り込まれます。パイプライン割り込みは、完全にハードウェアで処理されます。このような依存性を「インターロック」と呼びます。インターロック条件は、実行中の全ての命令で毎サイクル、チェックされます。先行する命令の結果で決まる命令は、インターロック条件に該当する一例です。

一般的に、MIPS® プロセッサは以下の2種類のハードウェア インターロックをサポートします。

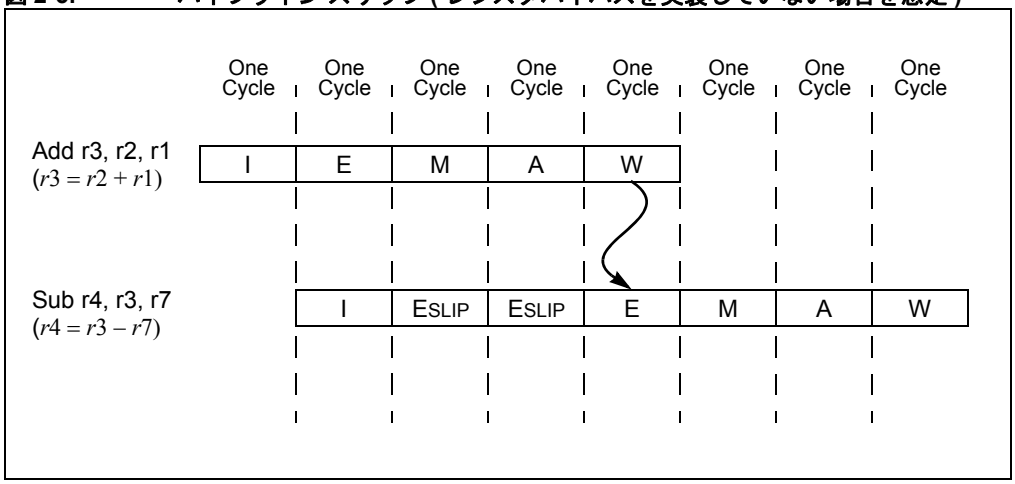
- ストール - これらのインターロックは、パイプライン全体の一時的な中断により解決します。各パイプライン段で現在実行中の全ての命令がストールの影響を受けます。
- スリップ - これらのインターロックを使うと、パイプラインの一部を一時的に停止し、その他の部分の処理を進行できます。

PIC32 プロセッサコアは全てのインターロックをスリップとして処理します。この後で説明するレジスタバイパスと呼ぶ方法を使って、他のパイプライン段から演算結果を取り込む事により、スリップを最小限に抑える事ができます。

Note:	パイプライン スリップの概念を説明するために、仮に PIC32 コアがレジスタバイパスを実装していなかった場合を想定します。

この場合、図 2-5 に示すように、減算命令はその前の加算命令の結果であるレジスタ r3 に対してソースオペランド依存性を持ちます。減算命令は、加算結果がレジスタ r3 に書き戻されるまで待機する事により 2 クロックスリップします。実際の PIC32 ファミリのプロセッサでは、このようなスリップは発生しません。

図 2-5: パイプライン スリップ (レジスタバイパスを実装していない場合を想定)



2.3.6 レジスタバイパス

既に述べたように、PIC32 プロセッサは、実行中のパイプライン スリップを低減するために、レジスタバイパスと呼ばれる機構を備えています。ある命令が現在パイプラインの E 段である場合、オペランドを取得できないと次へ進む事はできません。ある命令が実行パイプライン内の別の命令の計算結果をソースオペランドとする場合、レジスタバイパス機能を使うと、ソースオペランドをパイプラインから直接取得する事ができます。すなわち、E 段を実行中の命令は、パイプラインの M 段または A 段を実行中の別の命令からソースオペランドを取得できるという事です。図 2-6 に示した相互に依存する連続した 3 つの命令は、実行中に一切スリップしません。この場合、A から E と M から E へのレジスタバイパスを使います。図 2-7 は、A から E へのバイパスを使ったロード命令の動作を示しています。ロード命令の結果はパイプラインが A 段になるまで利用できないため、M から E へのバイパスを使いません。

レジスタバイパスの性能上の利点は、たとえレジスタ依存性が存在する場合でも、命令スループットを 1 命令 / クロックの ALU 演算レートまで高速化できる点にあります。

図 2-6: IU パイプライン、M 段から E 段へのバイパス

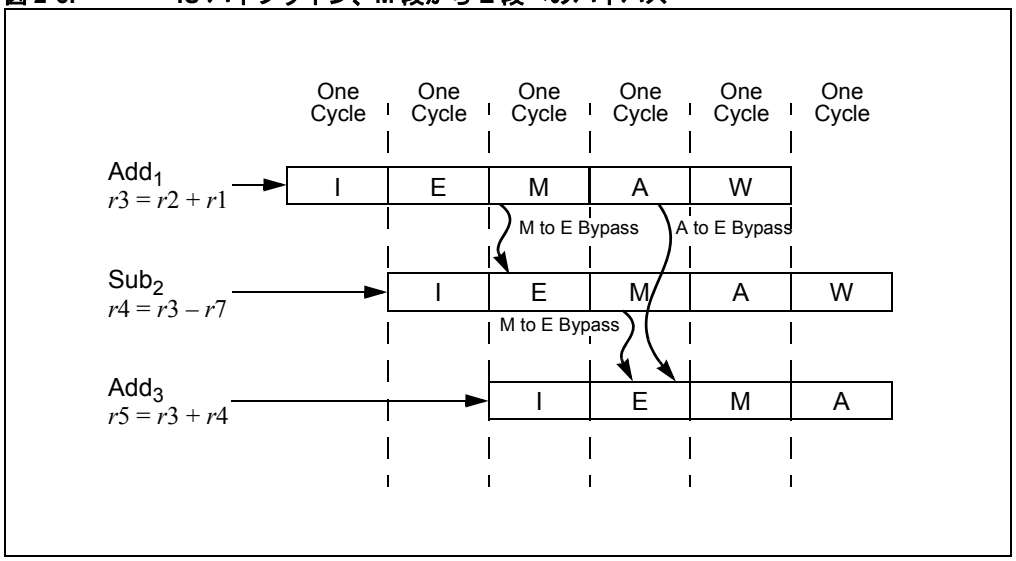
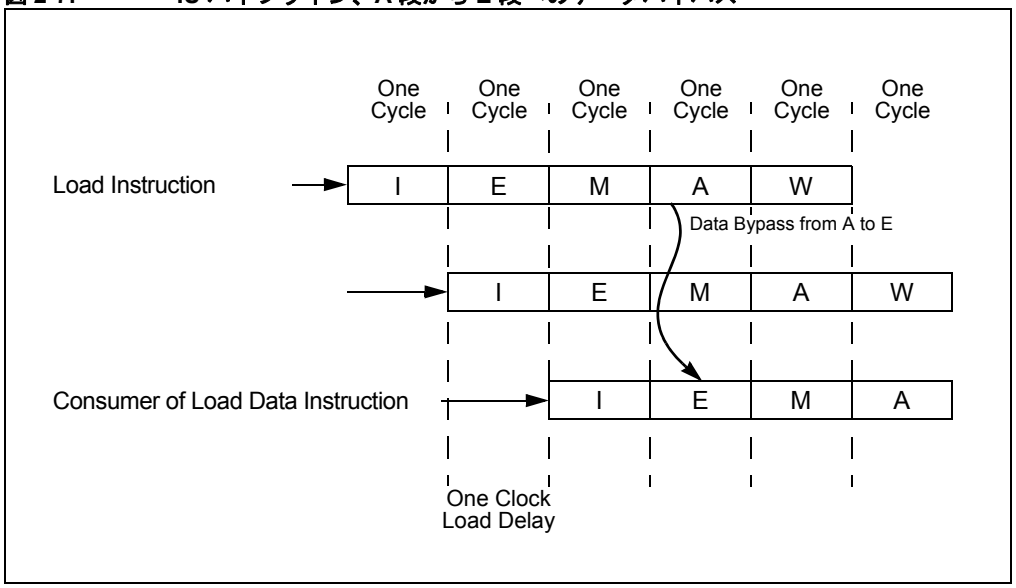


図 2-7: IU パイプライン、A 段から E 段へのデータバイパス



2.4 CP0 レジスタに書き込む場合の注意点

一般的に、PIC32 コアは、完全にシーケンシャルなプログラムモデルに従って命令が実行される事を保証します。このため、プログラム内の各命令は、先行する命令の結果を利用できます。しかし、このモデルに従わない状況も発生します。そのような状況を「ハザード」と呼びます。

特権ソフトウェアでは、以下の 2 種類のハザードが存在します。

- 実行ハザード
- 命令ハザード

2.4.0.1 実行ハザード

実行ハザードは、ある命令の実行に起因して発生し、別の命令の実行によって顕在化するハザードです。表 2-1 に実行ハザードを示します。

表 2-1: 実行ハザード

生成する命令	顕在化させる命令	ハザードの影響	間隔 (命令数)
MTC0	CU0 ビット (Status<28>) の更新値に依存するコプロセッサ命令の実行	CU0 ビット (Status<28>)	1
MTC0	ERET	EPC、DEPC、ErrorEPC	1
MTC0	ERET	Status	0
MTC0、EI、DI	割り込まれた命令	IE ビット (Status<0>)	1
MTC0	割り込まれた命令	IP1 および IP0 ビット (Cause<1>、<0>)	3
MTC0	RDPGPR、WRPGPR	PSS<3:0> ビット (SRSCtl<9:6>)	1
MTC0	コアタイマ割り込みに遭遇していない命令	コアタイマ割り込みをクリアするコンペア値の更新	4
MTC0	変更の影響を受けた命令	その他の CP0 レジスタ	2

2.4.0.2 命令ハザード

命令ハザードは、ある命令の実行に起因して発生し、別の命令のフェッチによって顕在化するハザードです。表 2-2 に命令ハザードを示します。

表 2-2: 命令ハザード

生成する命令	顕在化させるフェッチ	ハザードの影響
MTC0	値の変更に遭遇した命令フェッチ (useg セグメントからの命令フェッチに先行する ERL への変更を含む)	Status

2.5 アーキテクチャ リリース 2 の詳細

PIC32 CPU は、MIPS® 32 ビットプロセッサ アーキテクチャのリリース 2 を採用し、以下のリリース 2 機能を備えます。

- コア外部の割り込みコントローラを使うベクタ割り込み :

  その割り込みのハンドラに対する直接的なベクタ割り込みを可能にします。

- プログラマブルな例外ベクタベース :

  例外ベクタのベースアドレスを、$Status_{BEV}$ が「0」の時に発生した例外へ移動できます。これにより、どのようなシステムでも、例外ベクタをシステム環境に適したメモリ位置に配置できます。

- アトミックな割り込み有効化 / 無効化 :

  割り込みをアトミックに有効または無効にし、ステータス レジスタの以前の値を返すために、2 つの命令を追加しています。

- 消費電力が重視されるアプリケーション向けに Count レジスタを無効にする機能
- GPR シャドーレジスタ :

  GPR シャドーレジスタを追加し、これらのレジスタを束ねてベクタ割り込みまたは例外に提供します。

- フィールド、ローテート、シャッフル命令 :

  レジスタ内のビットフィールドの処理機能を追加しています。

- 明示的なハザード管理 :

  サイクルベースの SSNOP 法の代わりに、明示的にハザードを管理する一式の命令を提供します。

2.6 2 つの CPU バス

PIC32 CPU コアは 2 つのバスを備え、単一バスシステムよりもシステム性能を高める事ができます。性能の向上は、並列処理により得られます。ロードおよびストア動作は、命令フェッチと同時に発生します。2 つのバスは I バスと D バスと呼びます。I バスは命令を CPU に伝達するために使い、D バスはデータの転送に使います。

CPU は、パイプラインの I 段で命令をフェッチします。フェッチは I バスに対して発行され、バスマトリクス ユニットにより処理されます。アドレスに応じて BMX は以下のどれか 1 つを実行します。

- フェッチ要求をプリフェッチ キャッシュ ユニットに送る
- フェッチ要求を DRM ユニットに送る
- 例外を発生させる

命令フェッチは、フェッチされるアドレスに関係なく、常に I バスを使います。BMX は、アドレスと BMX レジスタ内の値に基づいて、各フェッチ要求向けに実行する動作を決めます。**セクション 3.「メモリ構成」**(DS61115) の **3.5「バスマトリクス」**を参照してください。

D バスは、CPU が実行する全てのロードおよびストア動作を処理します。ロードまたはストア命令が実行される時、その要求は D バスを介して BMX に渡されます。この動作はパイプラインの M 段で発生し、以下の各種ターゲット デバイスに向けられます。

- データ RAM
- プリフェッチ キャッシュ / フラッシュメモリ
- 高速周辺モジュールバス (割り込みコントローラ、DMA、デバッグユニット、USB、Ethernet、GPIO ポート)
- 汎用周辺モジュールバス (UART (Universal Asynchronous Receiver Transmitter)、SPI、フラッシュ コントローラ、EPMP/EPSP、RTCC タイマ、入力キャプチャ、PWM/ 出力コンペア、ADC、デュアルコンペア、I^2C、クロック、リセット)

2.7 内部システムバス

PIC32 プロセッサの内部バスは、周辺モジュールをバスマトリクス ユニットに接続します。バスマトリクスは、デバイス全体の各種データパスを使って、イニシエータからターゲットへのバスアクセスを切り換える事により、性能上のボトルネックを排除します。

バスマトリクスが使うパスには、用途が固定されているものと、各種ターゲットで共有するものがあります。

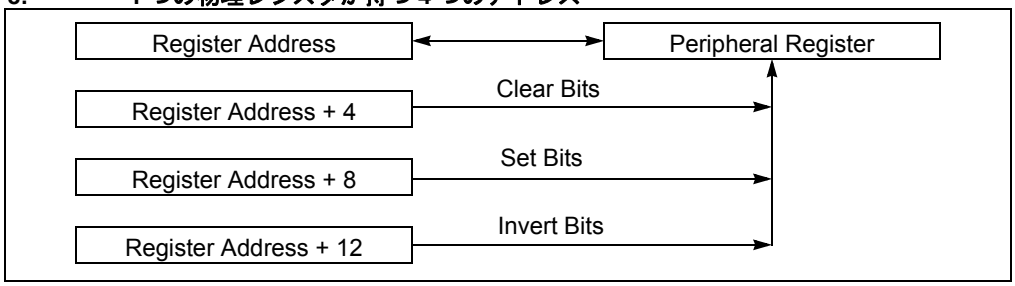
データ RAM とフラッシュメモリの読み出しパスは専用パスです。このため、周辺モジュールバスの動作によって遅延する事なく、低レイテンシでメモリリソースにアクセスできます。高帯域幅の周辺モジュールは高速バス上に配置されます。これには、割り込みコントローラ、デバッグユニット、DMA エンジン、USB ホスト / 周辺モジュール ユニット、その他の高帯域幅周辺モジュール (CAN、Ethernet エンジン等) を含みます。

高帯域幅を必要としない周辺モジュールは、消費電力を節約するために、別の周辺モジュールバス上に配置されます。

2.8 セット / クリア / 反転

周辺モジュールに対するシングルサイクルのビット操作を可能にするために、周辺モジュール ユニット内の各レジスタには、ビット操作専用の 3 種類の周辺モジュール アドレスを使ってアクセスできます。従って各レジスタは、ベースアドレスを含めて計 4 つの異なるアドレスを持ちます。このため、1 つのレジスタが 4 つの異なるレジスタのように見えますが、実際には同一物理レジスタのアドレスを 4 つの異なる方法で指定できるという事です。

図 2-8: 1 つの物理レジスタが持つ 4 つのアドレス



ベース レジスタアドレスは、通常の読み書きアクセスに使い、他の 3 つのアドレスは特殊な書き込み専用機能に使います。

- 通常アクセス
- アトミックな RMW アクセスによるビットのセット
- アトミックな RMW アクセスによるビットのクリア
- アトミックな RMW アクセスによるビットの反転

周辺モジュールの読み出しは、各周辺モジュール レジスタのベースアドレスから実行する必要があります。ビットのセット / クリア / 反転用アドレスからの読み出し結果は未定義であり、周辺モジュールごとに異なる可能性があります。

ベースアドレスへの書き込みは、周辺モジュール レジスタに値全体を書き込みます (全てのビットに書き込みます)。例として、現在 0xAAAA5555 を格納しているレジスタに 0x000000FF を書き込んだ場合を考えます。書き込み後、そのレジスタの値は 0x000000FF となります (全てのビットが読み書き可能ビットであると仮定)。

周辺モジュール レジスタのビットセット用アドレスに対する書き込みでは、「1」を書き込んだビットに対応する、デスティネーション レジスタのビットだけがセットされます。例として、現在 0xAAAA5555 を格納しているレジスタのビットセット用レジスタアドレスに 0x000000FF を書き込んだ場合を考えます。その周辺モジュール レジスタの値は 0xAAAA55FF となります。

周辺モジュール レジスタのビットクリア用アドレスに対する書き込みでは、「1」を書き込んだビットに対応する、デスティネーション レジスタのビットだけが「0」にクリアされます。例として、現在 0xAAAA5555 を格納しているレジスタのビットクリア用レジスタアドレスに 0x000000FF を書き込んだ場合を考えます。その周辺モジュール レジスタの値は 0xAAAA5500 となります。

周辺モジュール レジスタのビット反転用アドレスに対する書き込みでは、「1」を書き込んだビットに対応するデスティネーション レジスタのビットだけが反転 (トグル) されます。例として、現在 0xAAAA5555 を格納しているレジスタのビット反転用レジスタアドレスに 0x000000FF を書き込んだ場合を考えます。その周辺モジュール レジスタの値は 0xAAAA55AA となります。

2.9 ALU ステータスビット

他のほとんどの PIC® マイクロコントローラとは異なり、PIC32 プロセッサはステータス レジスタ フラグを使いません。多くのプロセッサは、プログラム実行中に各種の条件を判断するために、条件フラグを使います。これらのフラグは、比較演算または何らかの算術演算の結果に基づいてセットされます。このようなプロセッサにおける条件分岐命令は、定められた一式の条件コードの値に基づいて条件判断します。

これとは異なり、PIC32 プロセッサの命令は、比較と同時に汎用レジスタにフラグまたは値を保存します。条件分岐は、この汎用レジスタをオペランドとして使って実行されます。

2.10 割り込みおよび例外機構

Note:	本書では、「識別可能」と「識別困難」という用語を使って例外を説明します。識別可能な例外とは、例外発生の原因となった命令を EPC (CP0 レジスタ 14、Select 0) を使って識別できるものを指します。識別困難な例外とは、例外を発生させた命令を識別できないものを指します。ほとんどの例外は識別可能です。バスエラー例外は識別困難である場合があります。

PIC32 ファミリのプロセッサは、効率的で柔軟な割り込みと例外処理機構を備えています。割り込みと例外は、それらを処理する特殊なプロシージャを実行するために現在の命令フローを一時的に変更するという点で、どちらも同じ挙動を示します。両者の違いは、割り込みが一般的に正常動作の結果として発生するのに対し、例外はバスエラー等のエラー条件の結果として発生する点です。

割り込みまたは例外が発生した時、プロセッサは以下のように対応します。

1. ハンドラが処理を元のルーチンに返した直後に実行する命令の PC をコプロセッサ レジスタに保存する
2. 例外または割り込みの原因を示すために Cause レジスタを更新する
3. カーネルモード実行を開始するために Status レジスタの EXL または ERL ビットをセットする
4. ハンドラの PC を Ebase と SPACING 値から求める
5. プロセッサは更新された PC から実行を始める

上記は、割り込みと例外機構の簡単なまとめです。割り込みと例外の処理の詳細は、**セクション 8. 「割り込み」** (DS61108) を参照してください。

2.11 プログラミング モデル

PIC32 ファミリのプロセッサは、C プログラミング言語等の高級言語の適用を前提に設計されています。このため、高級言語で必要となる各種のデータ型とシンプルで柔軟なアドレッシング モードをサポートします。また、32 個の汎用レジスタと、乗除算用に 2 個の特殊レジスタを備えています。

PIC32 プロセッサ用のマシン語命令には、以下の 3 種類のフォーマットが存在します。

- 即値 (I 型) CPU 命令
- ジャンプ (J 型) CPU 命令
- レジスタ (R 型) CPU 命令

大部分の動作にはレジスタを使います。レジスタ型 CPU 命令は 3 個のオペランド (2 個のソースオペランドと 1 個のデスティネーション オペランド) を使います。

3 個のオペランドと豊富なレジスタセットを使うと、アセンブリ言語プログラマとコンパイラは CPU リソースを効率的に使う事ができます。これにより、中間結果をレジスタで保持できるため、メモリとの間でデータを頻繁にやり取りする必要がなく、従ってより高速でコンパクトなプログラムを作成できます。

即値型命令は、即値オペランド、ソースオペランド、デスティネーション オペランドを使います。ジャンプ型命令は、ジャンプ先の計算用に 26 ビットの相対命令オフセット フィールドを使います。

2.11.1 CPU 命令のフォーマット

CPU 命令は、32 ビット幅に揃えられた 1 つのワードです。CPU 命令には下記の種類があります。

- 即値 (図 2-9 参照)
- ジャンプ (図 2-10 参照)
- レジスタ (図 2-11 参照)

表 2-3 に、これらの命令が使うフィールドを示します。

表 2-3: CPU 命令のフォーマット フィールド

フィールド	説明
opcode	6 ビットのプライマリ オペコード
rd	デスティネーション レジスタを指定する 5 ビットの指定子
rs	ソースレジスタを指定する 5 ビットの指定子
rt	ターゲット (ソース / デスティネーション) レジスタを指定する 5 ビットの指定子 (または、プライマリ オペコード REGIMM 内で関数を指定するためにも使用)
immediate	論理オペランド、算術符号付きオペランド、ロード / ストア アドレスバイト オフセット、PC 相対分岐符号付き命令オフセットに使う、16 ビットの符号付き即値
instr_index	ジャンプ先アドレスの下位 28 ビットを提供するために左に 2 ビットシフトした 26 ビットのインデックス
sa	5 ビットのシフト量
function	プライマリ オペコード SPECIAL 内で関数を指定するために使う 6 ビットの関数フィールド

図 2-9: 即値 (I 型) CPU 命令のフォーマット

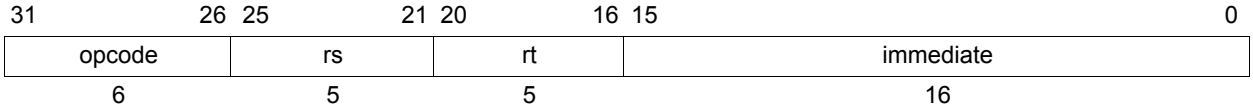


図 2-10: ジャンプ (J 型) CPU 命令のフォーマット

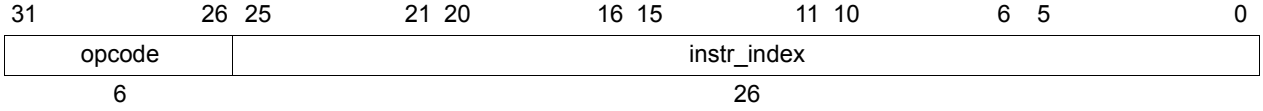
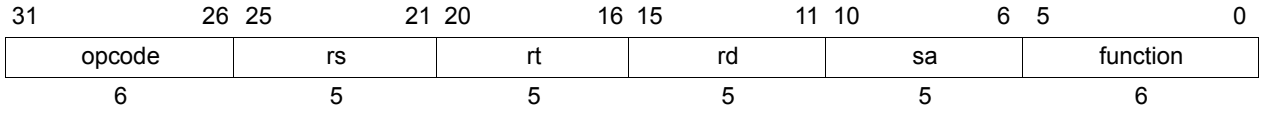


図 2-11: レジスタ (R 型) CPU 命令のフォーマット



2.11.2 CPU レジスタ

PIC32 アーキテクチャでは以下の CPU レジスタを定義しています。

- 32 個の 32 ビット汎用レジスタ (GPR)
- 2 個の専用レジスタ (HI と LO): 整数乗除演算と乗算累積演算の結果を保持するために使用
- 1 個の特殊なプログラム カウンタ (PC): 特定の命令実行により間接的に影響を受ける (アーキテクチャ的には不可視のレジスタ)

2.11.2.1 CPU 汎用レジスタ

CPU 汎用レジスタのうち以下の 2 個に特定の機能が割り当てられています。

- r0 - このレジスタは、ハードウェアで値「0」に結線されており、全ての命令でターゲットレジスタとして使えますが、その場合、命令の結果は破棄されます。r0 は、値として「0」が必要な時に、ソースとして使う事もできます。
- r31 - JAL、BLTZAL、BLTZALL、BGEZAL、BGEZALL はこのレジスタをデスティネーションレジスタとして使います (命令ワードで明示的に指定する必要はありません)。その他の命令は r31 を通常の汎用レジスタとして使います。

上記以外のレジスタは全て汎用的に使えます。

2.11.2.2 推奨するレジスタの使用方法

PIC32 アーキテクチャ内の大部分のレジスタは汎用レジスタとして設計されていますが、Microchip 社 C コンパイラ等の高級言語で作成したソフトウェアを適正に動作させるために、レジスタの使用方法には表 2-4 に示す、いくつかの推奨事項があります。

表 2-4: 推奨するレジスタの使用方法

CPU レジスタ	レジスタの記号表記	使用方法
r0	zero	常に「0」(1)
r1	at	アセンブラでテンポラリ レジスタとして使用
r2 - r3	v0-v1	関数の戻り値
r4 - r7	a0-a3	関数への引数
r8 - r15	t0-t7	テンポラリ - 関数呼び出し元は内容を保持する必要はない
r16 - r23	s0-s7	保存テンポラリ - 関数呼び出し元は内容を保持する必要がある
r24 - r25	t8-t9	テンポラリ - 関数呼び出し元は内容を保持する必要はない
r26 - r27	k0-k1	カーネル テンポラリ - 割り込みと例外処理に使用
r28	gp	グローバル ポインタ - 共通データへの高速アクセスに使用
r29	sp	スタックポインタ - ソフトウェア スタック
r30	s8 または fp	保存テンポラリ - 関数呼び出し元は内容を保持する必要がある またはフレームポインタ - スタック上のプロシージャ フレームへのポインタ
r31	ra	リターンアドレス (1)

Note 1: これは推奨する用法ではなく、ハードウェアで強制される用法です。

2.11.2.3 CPU 専用レジスタ

CPU は以下の 3 個の専用レジスタを備えています。

- PC - プログラム カウンタ レジスタ
- HI - 乗除算レジスタ (結果の上位)
- LO - 乗除算レジスタ (結果の下位)
  - 乗算の場合、HI および LO レジスタは整数乗算の積を格納します。
  - 積和または積差演算の場合、HI および LO レジスタは整数積和または積差の結果を格納します。
  - 除算の場合、LO レジスタは整数除算の商を格納し、HI レジスタは剰余を格納します。
  - 累積乗算の場合、HI および LO レジスタは演算の累積結果を格納します。

図 2-12 に CPU レジスタの配置を示します。

図 2-12: CPU レジスタ

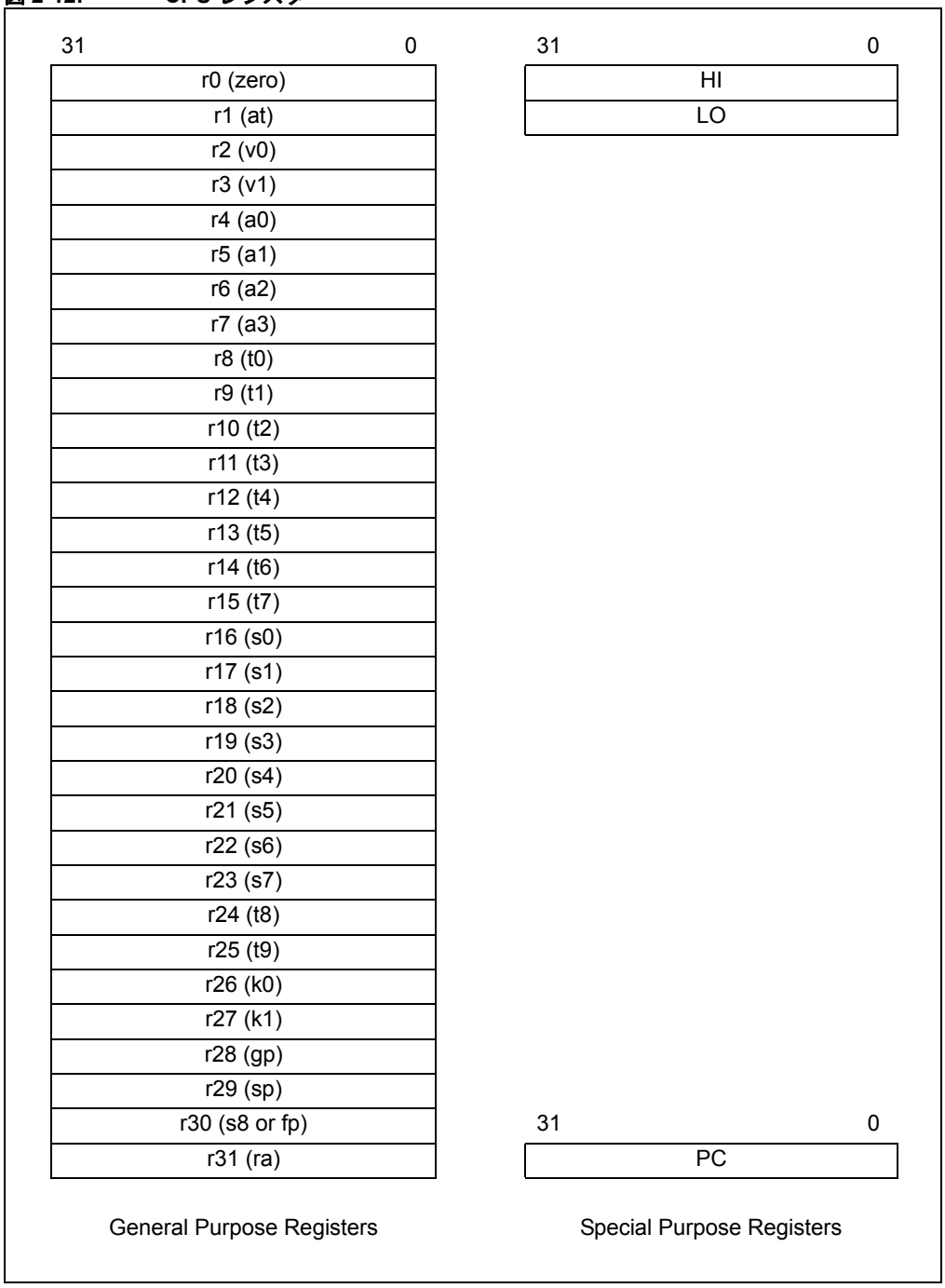


表 2-5: MIPS16e® レジスタの使用方法

MIPS16e® レジスタ エンコード	32 ビット MIPS® レジスタ エンコード	シンボル名	説明
0	16	s0	汎用レジスタ
1	17	s1	汎用レジスタ
2	2	v0	汎用レジスタ
3	3	v1	汎用レジスタ
4	4	a0	汎用レジスタ
5	5	a1	汎用レジスタ
6	6	a2	汎用レジスタ
7	7	a3	汎用レジスタ
なし	24	t8	MIPS16e® 条件コードレジスタ ; BTEQZ、BTNEZ、CMP、CMPI、SLT、SLTU、SLTI、SLTIU 命令が暗黙的に参照
なし	29	sp	スタックポインタ レジスタ
なし	31	ra	リターンアドレス レジスタ

表 2-6: MIPS16e® 専用レジスタ

シンボル名	目的
PC	プログラム カウンタ : PC 相対命令はこのレジスタをオペランドとして使用可能
HI	乗算または除算結果の上位ワードを格納
LO	乗算または除算結果の下位ワードを格納

2.11.3 スタックの実装方法 /MIPS® の呼び出し規則

PIC32 CPU はハードウェア スタックを備えません。プロセッサは、ソフトウェアを使ってこの機能を提供します。ハードウェアはスタック動作そのものを実行しないため、システム内の全てのソフトウェアで同じスタック動作を得るために、統一した規則が必要です。例えば、スタックはアドレスが大きくなる方向にも小さくなる方向にも変化できます。ソフトウェアの一部のルーチンでは「スタックはアドレスが小さくなる方向に変化する」という事を前提とし、そのルーチンが「スタックはアドレスが大きくなる方向に変化する」という事を前提とする別のルーチンを呼び出した場合、スタックは正しく機能しません。

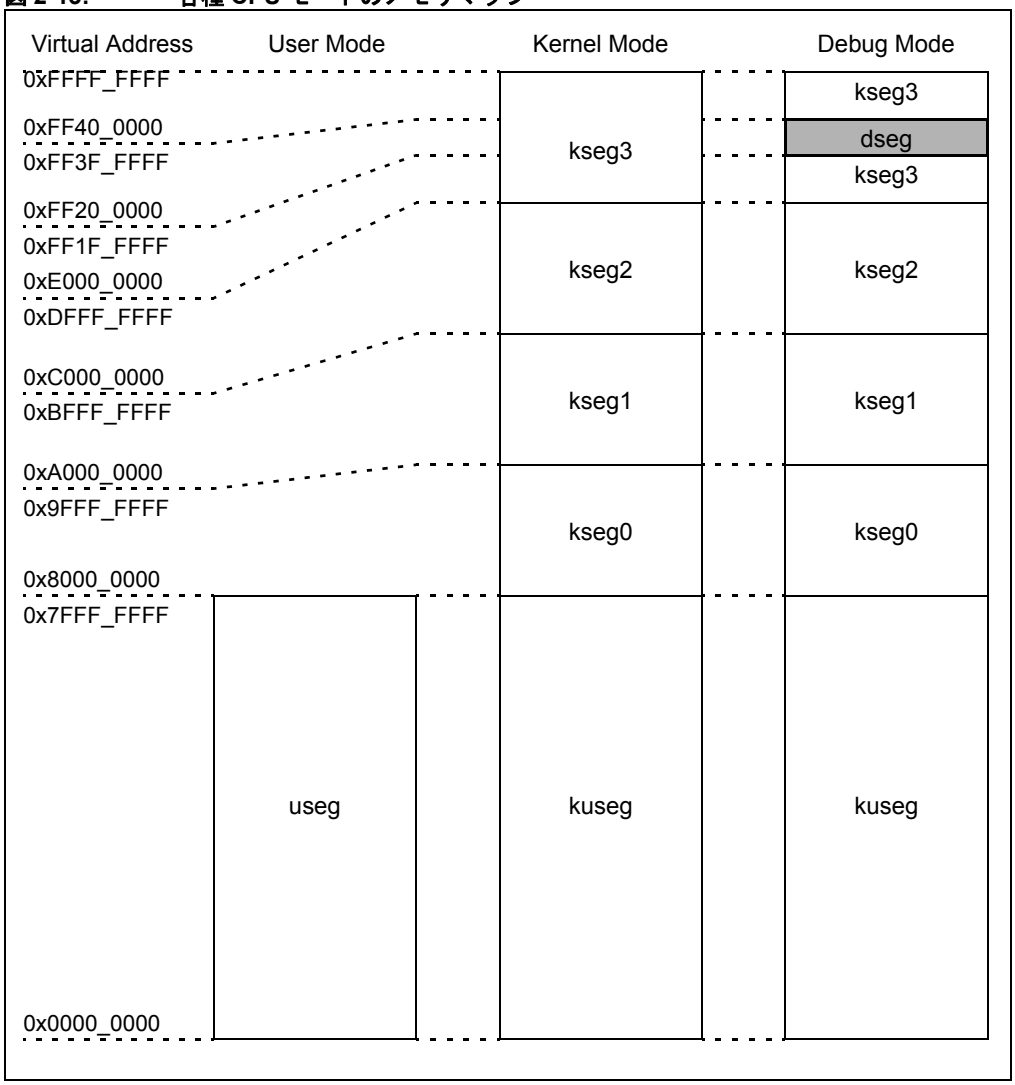
システム全体で呼び出し時の規則を統一する事により、このような問題を防げます。Microchip 社のＣコンパイラでは､｢スタックはアドレスが小さくなる方向に変化する｣事を前提とします。

2.11.4 プロセッサモード

PIC32 ファミリの CPU には、2 つの動作モード (ユーザモード、カーネルモード) と、1 つの特殊な実行モード (デバッグモード) が存在します。プロセッサはカーネルモードで実行を始め、通常動作時もカーネルモードで動作できます。ユーザモードはオプションのモードです。このモードを使うと､コードを特権ソフトウェアと非特権ソフトウェアに分割できます｡デバッグモードは、通常はデバッガまたはモニタだけが使います。

これらの各動作モードでは、ソフトウェアからアクセスできるメモリアドレスが異なります。ユーザモードでは、周辺モジュールにアクセスできません。図 2-13 に、各モードにおけるメモリマップを示します。プロセッサのメモリマップに関する詳細は、**セクション 3.「メモリ構成」** (DS61115) を参照してください。

図 2-13:　　各種 CPU モードのメモリマップ



2.11.4.1 カーネルモード

ハードウェア リソースの多くにアクセスするには、プロセッサをカーネルモードで動作させる必要があります。カーネルモードの場合、ソフトウェアはプロセッサのアドレス空間の全域にアクセスできます。また、特権命令へのアクセスも可能です。

Debug レジスタの DM ビットが「0」かつ Status レジスタで以下の条件が 1 つまたは複数成立する場合、プロセッサはカーネルモードで動作します。

- UM = 0
- ERL = 1
- EXL = 1

デバッグ例外ではない例外が検出されると、EXL または ERL がセットされ、プロセッサはカーネルモードに移行します。例外ハンドラルーチンの最後では、一般的に例外からの復帰 (ERET) 命令が実行されます。ERET 命令は、例外 PC (例外の種類に応じて EPC または ErrorPC) へジャンプして、ERL をクリアし、ERL= 0 であれば EXL をクリアします。

UM = 1 の場合、ERL と EXL が「0」にクリアされて例外から復帰した後に、プロセッサはユーザモードに戻ります。

2.11.4.2 ユーザモード

ユーザモードで動作する場合、ソフトウェアはプロセッサ リソースの 1 つのサブセットだけを使えます。多くの場合、アプリケーション レベルのコードはユーザモードで実行する事を推奨します。そうする事により、発生したエラーを封じ込めて、カーネルモードのコードに影響してしまう事を防げます。

アプリケーションは、SYSCALL 機構等の制御されたインターフェイスを介してカーネルモードの関数にアクセスできます。

図 2-13 が示すように、ユーザモード ソフトウェアは USEG メモリ空間にアクセスできます。

ユーザモードで動作するには、Status レジスタのビット値を以下のように設定する必要があります。

- UM = 1
- EXL = 0
- ERL = 0

2.11.4.3 デバッグモード

デバッグモードはプロセッサの特殊なモードであり、通常はデバッガまたはシステムモニタだけが使います。デバッグモードにはデバッグ例外を介して移行します。このモードでは、全てのカーネルモード リソースに加えて、デバッグ アプリケーションが使う特殊なハードウェア リソースにアクセスできます。

Debug レジスタの DM ビットが「1」の時に、プロセッサはデバッグモードで動作します。

通常、デバッグモードは、デバッグハンドラから DERET 命令を実行する事により終了します。

2.12 コプロセッサ 0 (CP0) レジスタ

PIC32 は、システム ソフトウェアと CPU の間でステータス情報と制御情報を交換するために、特殊なレジスタ インターフェイスを使います。このインターフェイスを、コプロセッサ 0 (CP0) と呼びます。コプロセッサ 0 を介して可視となる CPU 機能は以下の通りです。

- コアタイマ
- 割り込みと例外制御
- 仮想メモリのコンフィグレーション
- シャドーレジスタ セットの制御
- プロセッサ ID
- デバッガ制御
- パフォーマンス カウンタ

システム ソフトウェアは、MFC0 と MTC0 等のコプロセッサ命令を使って、CP0 内のレジスタにアクセスします。表 2-7 に、PIC32 が備える CP0 レジスタを示します。

表 2-7: CP0 レジスタ

レジスタ番号	レジスタ名	機能
0-6	予約済み	M4K® マイクロプロセッサ コア内で予約済み
7	HWREna	非特権モードでの RDHWR 命令による一部ハードウェア レジスタへのアクセスを可能にします。
8	BadVAddr	直前に発生したアドレス関連の例外のアドレスを格納します。
9	Count	プロセッサ サイクル数
10	予約済み	M4K® マイクロプロセッサ コア内で予約済み
11	Compare	コアタイマ割り込みの制御
12	Status	プロセッサのステータスと制御
	IntCtl	ベクタ空間の割り込み制御
	SRSCtl	シャドーレジスタ セットの制御
	SRSMap	シャドーレジスタ割り当ての制御
13	Cause	直前の例外の原因を示します。
14	EPC	直前の例外時のプログラム カウンタ
15	PRID	プロセッサの ID とリビジョン
	Ebase	例外ベクタの例外ベースアドレス
16	Config	コンフィグレーション レジスタ
	Config1	コンフィグレーション レジスタ 1
	Config2	コンフィグレーション レジスタ 2
	Config3	コンフィグレーション レジスタ 3
17-22	予約済み	M4K® マイクロプロセッサ コア内で予約済み
23	Debug	デバッグ制御 / 例外ステータス
	TraceControl	EJTAG トレース制御
	TraceControl2	EJTAG トレース制御 2
	UserTraceData	ユーザ フォーマット タイプによるトレースレコードのトリガ
	TraceBPC	EJTAG ハードウェア ブレークポイントを使ったトレースの制御
	Debug2	デバッグ制御 / 例外ステータス 1
24	DEPC	直前のデバッグ例外時のプログラム カウンタ
25-29	予約済み	M4K® マイクロプロセッサ コア内で予約済み
30	ErrorEPC	直前のエラー時のプログラム カウンタ
31	DeSAVE	デバッグハンドラのスクラッチパッド レジスタ

2.12.1 HWREna レジスタ (CP0 レジスタ 7、Select 0)

HWREna レジスタは、RDHWR 命令によるアクセスを許可するハードウェア レジスタを指定するためのビットマスクを格納します。

レジスタ 2-1: HWREna: ハードウェア アクセス許可レジスタ ; CP0 レジスタ 7、Select 0

ビット レンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
7:0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	-	-	-	-	MASK<3:0>			

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　x = ビットは未知

bit 31-4 **未実装 :** 「0」として読み出し

bit 3-0 **MASK<3:0>:** ビットマスク ビット

1 = 対応するハードウェア レジスタに対するアクセスを有効にする
0 = アクセスを無効にする
これらの各ビットは、RDHWR 命令による特定ハードウェア レジスタ (仮想レジスタの場合もある) へのアクセスを有効にします。利用できるハードウェア レジスタのリストは、RDHWR 命令の説明を参照してください。

2.12.2 BadVAddr レジスタ (CP0 レジスタ 8、Select 0)

BadVAddr レジスタは読み出し専用レジスタです。このレジスタは、直前のアドレスエラー例外を引き起こした仮想アドレスをキャプチャします。アドレスエラーは不整合なアドレスからのロード、ストア、フェッチ動作により発生します。また、ユーザモードからのカーネルモード アドレスへのアクセスによっても発生します。

BadVAddr レジスタは、バスエラーに関するアドレス情報をキャプチャしません (バスエラーはアドレッシング エラーではないため)。

レジスタ 2-2: BadVAddr: 不正仮想アドレスレジスタ ; CP0 レジスタ 8、Select 0

ビット レンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadVAddr<31:24>							
23:16	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadVAddr<23:16>							
15:8	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadVAddr<15:8>							
7:0	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadVAddr<7:0>							

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　　x = ビットは未知

bit 31-0 **BadVAddr<31:0>:** 不正仮想アドレスビット
このビットは、直前のアドレスエラー例外を引き起こした仮想アドレスをキャプチャします。

2.12.3 Count レジスタ (CP0 レジスタ 9、Select 0)

Count レジスタはタイマとして機能し、一定レートでインクリメントします。命令が実行されても破棄されても、あるいはパイプラインで処理が進行してもしなくても、一切関係なくインクリメントします。Cause レジスタの DC ビットが「0」である場合、カウンタは 1 クロックおきにインクリメントします。

機能上の目的あるいは診断目的でカウント値を書き込む事ができます。あるいは、リセット時またはプロセッサを同期させるために書き込む事ができます。

Debug レジスタの COUNTDM ビットの設定により、プロセッサがデバッグモードの時にカウントを継続するかどうかを指定できます。

レジスタ 2-3: Count: インターバル カウンタ レジスタ ; CP0 レジスタ 9、Select 0

ビット レンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COUNT<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COUNT<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COUNT<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COUNT<7:0>							

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　　x = ビットは未知

bit 31-0 **COUNT<31:0>:** インターバル カウンタ ビット
この値は 1 クロックサイクルおきにインクリメントします。

2.12.4 Compare レジスタ (CP0 レジスタ 11、Select 0)

Compare レジスタと Count レジスタの組み合わせにより、タイマとタイマ割り込み機能を実装します。Compare レジスタは一定の値を保持し、自身の内容を変更しません。

Count レジスタの値が Compare レジスタの値に一致した時、CPU はシステム割り込みコントローラに対して割り込み信号をアサートします。この信号は、Compare レジスタに対して書き込みが発生するまでアサート状態を維持します。

レジスタ 2-4: Compare: インターバル カウントコンペア レジスタ ; CP0 レジスタ 11、Select 0

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COMPARE<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COMPARE<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COMPARE<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COMPARE<7:0>							

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　　x = ビットは未知

bit 31-0 **COMPARE<31:0>:** インターバル カウントコンペア値ビット

2.12.5 Status レジスタ (CP0 レジスタ 12、Select 0)

読み書き可能な Status レジスタは、プロセッサの動作モード、割り込み有効状態、診断ステートを格納します。このレジスタのビットの組み合わせにより、プロセッサの動作モードが決まります。

2.12.5.1 割り込みの有効化

割り込みは、以下の条件が全て成立する時に有効です。

- IE = 1
- EXL = 0
- ERL = 0
- DM = 0

これらの条件が全て成立する場合、IPL ビットの設定に応じて割り込みが生成されます。

2.12.5.2 動作モード

Debug レジスタの DM ビットが「1」の場合、プロセッサはデバッグモードです。これ以外の場合、プロセッサはカーネルモードかユーザモードのどちらかです。

ユーザとカーネルのどちらのモードかは、下表のCPUステータス レジスタビットの状態によって判別できます。

表 2-8: CPU ステータス レジスタビットとプロセッサモードの関係

モード	ビット設定		
ユーザモード (右の条件の全てが成立した場合)	UM = 1	EXL = 0	ERL = 0
カーネルモード (右の条件のどれか 1 つでも成立した場合)	UM = 0	EXL = 1	ERL = 1

Note: Status レジスタの CU0 ビット (Status<28>) は、コプロセッサに対するアクセスを制御します。使用可能なコプロセッサが存在しない場合、コプロセッサにアクセスする命令を実行すると例外が発生します。

レジスタ 2-5: STATUS: Status レジスタ ; CP0 レジスタ 12、Select 0

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	R/W-x	R/W-0(1)	U-0	R/W-x	U-0
	-	-	-	CU0	RP	-	RE	-
23:16	U-0	R/W-1	U-0	R/W-1	R/W-0	U-0	U-0	U-0
	-	BEV	-	SR	NMI	-	-	-
15:8	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	U-0	U-0
	-	-	-	IPL<2:0>			-	-
7:0	U-0	U-0	U-0	R/W-x	U-0	R/W-x	R/W-x	R/W-x
	-	-	-	UM	-	ERL	EXL	IE

凡例 :

R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　x = ビットは未知

bit 31-29 **未実装 :** 「0」として読み出し

レジスタ 2-5: STATUS: Status レジスタ ; CP0 レジスタ 12、Select 0 (続き)

bit 28 **CU0:** コプロセッサ 0 使用可能ビット
コプロセッサ 0 に対するアクセスを制御します。
1 = アクセスを許可する
0 = アクセスを許可しない
プロセッサがカーネルモードで動作している場合、CU0 ビットの状態に関係なく、コプロセッサ 0 は常に使用可能です。

bit 27 **RP:** 省電力ビット
1 = 省電力モードを有効にする
0 = 省電力モードを無効にする

bit 26 **未実装 :** 「0」として読み出し

bit 25 **RE:** 逆エンディアン メモリ参照イネーブルビット
このビットは、プロセッサがユーザモードで動作している時に、逆エンディアンのメモリ参照を有効にします。
1 = ユーザモードは逆エンディアンを使う
0 = ユーザモードは設定されたエンディアンを使う
このビットの状態はデバッグ、カーネル、スーパーバイザ モードでのメモリ参照には影響しません。

bit 24-23 **未実装 :** 「0」として読み出し

bit 22 **BEV:** ブートストラップ例外ベクタ制御ビット
例外ベクタの位置を制御します。
1 = ブートストラップ
0 = 通常

bit 21 **未実装 :** 「0」として読み出し

bit 20 **SR:** ソフトリセット ビット
リセット例外ベクタへのリセットがソフトリセットに起因したかどうかを示します。
1 = ソフトリセットに起因(このビットはPIC32コア上の全てのタイプのリセット時に常にセットされます)
0 = PIC32 では未使用
ソフトウェアは、このビットに「0」を書き込んでクリアできますが、「0」を「1」に書き換える事はできません。

bit 19 **NMI:** ノンマスカブル割り込みビット
リセット例外ベクタを介するエントリがノンマスカブル割り込みに起因したかどうかを示します。
1 = NMI に起因する
0 = NMI に起因しない (ソフトリセットまたはリセットに起因)
ソフトウェアは、このビットに「0」を書き込んでクリアできますが、「0」を「1」に書き換える事はできません。

bit 18-13 **未実装 :** 「0」として読み出し

bit 12-10 **IPL<2:0>:** 割り込み優先度ビット
このビットは現在の割り込み優先度 (IPL) のエンコード値 (0-7) です。要求された割り込みの優先度がこの値より高い場合にのみ、割り込み信号が生成されます。

bit 9-5 **未実装 :** 「0」として読み出し

bit 4 **UM:** ユーザモード ビット
このビットは、プロセッサのベース動作モードを示します。このビットは以下のようにエンコードされます。
1 = ベースモードはユーザモード
0 = ベースモードはカーネルモード
ERL または EXL がセットされている場合、UM ビットの状態に関係なく、プロセッサはカーネルモードです。

bit 3 **未実装 :** 「0」として読み出し

レジスタ 2-5: STATUS: Status レジスタ ; CP0 レジスタ 12、Select 0 (続き)

bit 2 **ERL:** エラーレベル ビット

リセット、ソフトリセット、NMI (ノンマスカブル割り込み)、キャッシュエラー例外が発生すると、プロセッサはこのビットをセットします。
1 = エラーレベル
0 = 正常レベル

ERL がセットされている場合 :
- プロセッサはカーネルモードで動作中
- 割り込みは無効
- ERET命令は、EPCレジスタではなくErrorEPCレジスタで保持されているリターンアドレスを使います。
- kuseg の下位 2^{29} バイトは、マッピングもキャッシュもされない領域として扱われます。これにより、キャッシュエラーが存在する時にメインメモリへアクセスできます。プロセッサが kuseg から命令を実行している時に ERL ビットがセットされた場合、プロセッサの動作は不確定です。

bit 1 **EXL:** 例外レベルビット

リセット、ソフトリセット、NMI 以外の例外が発生すると、プロセッサはこのビットをセットします。
1 = 例外レベル
0 = 正常レベル

EXL がセットされている場合 :
- プロセッサはカーネルモードで動作中
- 割り込みは無効

別の例外が発生しても EPC、BD、SRSCtl は更新されません。

bit 0 **IE:** 割り込みイネーブルビット

ソフトウェアとハードウェア割り込みのマスタ イネーブルビットとして機能します。
1 = 割り込みを有効にする
0 = 割り込みを無効にする
このビットは、DI 命令を使って無効に、EI 命令を使って有効に変更できます。

2.12.6 IntCtl: 割り込み制御レジスタ (CP0 レジスタ 12、Select 1)

IntCtl レジスタは、PIC32 アーキテクチャのベクタ間隔を制御します。

レジスタ 2-6: IntCtl: 割り込み制御レジスタ ; CP0 レジスタ 12、Select 1

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
23:16	U=0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
15:8	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
	-	-	-	-	-	-	VS<4:4>	
7:0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
	VS<2:0>			-	-	-	-	-

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　x = ビットは未知

bit 31-10 **未実装 :** 「0」として読み出し

bit 9-5 **VS<4:0>:** ベクタ間隔ビット
これらのビットは、各割り込みベクタ間の間隔を指定します。

エンコード	ベクタの間隔 (16 進数)	ベクタの間隔 (10 進数)
0x00	0x000 0x	0
0x01	0x020	32
0x02	0x040	64
0x04	0x080	128
0x08	0x100	256
0x10	0x200	512

その他の値は全て予約済みです。予約済みの値をこれらのビットに書き込んだ場合、プロセッサの動作は不確定です。

bit 4-0 **未実装 :** 「0」として読み出し

2.12.7 SRSCtl レジスタ (CP0 レジスタ 12、Select 2)

SRSCtl レジスタは、プロセッサ内の GPR シャドーセットの動作を制御します。

表 2-9: 例外または割り込み時の CSS 更新値のソース

例外タイプ	ビットソース	条件	説明
例外	ESS	全て	-
非ベクタ割り込み	ESS	IV ビット = 0 (Cause<23>)	例外として扱う
ベクタ EIC 割り込み	EICSS	IV ビット (Cause<23>) = 1 かつ VEIC ビット (Config3<6>) = 1	外部の割り込みコントローラからの割り込み要求

レジスタ 2-7: SRSCtl: シャドーレジスタ セット レジスタ ; CP0 レジスタ 12、Select 2

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	R-0	R-0	R-0	R-1	U-0	U-0
	-	-	HSS<3:0>				-	-
23:16	U-0	U-0	R-x	R-x	R-x	R-x	U-0	U-0
	-	-	EICSS<3:0>				-	-
15:8	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
	ESS<3:0>				-	-	PSS<3:2>	
7:0	R/W-0	R/W-0	U-0	U-0	R-0	R-0	R-0	R-0
	PSS<1:0>		-	-	CSS<3:0>			

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　x = ビットは未知

bit 31-30 **未実装 :** 「0」として読み出し

bit 29-26 **HSS<3:0>:** 最大シャドーセット番号ビット
このビットは、プロセッサが実装しているシャドーセットの最大番号を格納します。これらのビットの値が「0000」の場合、通常の GPR のみ実装している事を意味します。
1111 = 予約済み
•
•
•
0100 = 予約済み
0011 = 4 個のシャドーセットが存在する
0010 = 予約済み
0001 = 2 個のシャドーセットが存在する
0000 = 1 個のシャドーセット (通常の GPR セット) が存在する

このビットの値は、このレジスタの ESS<3:0>、EICSS<3:0>、PSS<3:0>、CSS<3:0> ビットまたは SRSMap レジスタの任意のビットに書き込み可能な最大値でもあります。これらのビットにこの HSS ビットよりも大きな値を書き込んだ場合、プロセッサの動作は不確定です。

bit 25-22 **未実装 :** 「0」として読み出し

bit 21-18 **EICSS<3:0>:** 外部割り込みコントローラ シャドーセット ビット
EIC 割り込みモードのシャドーセットです。このビットは、割り込み要求ごとに外部割り込みコントローラから読み込まれ、それらの割り込みに現在のシャドーセットを選択するために、SRSMap の代わりに使われます。

bit 17-16 **未実装 :** 「0」として読み出し

レジスタ 2-7: SRSCtl: シャドーレジスタ セット レジスタ ; CP0 レジスタ 12、Select 2 (続き)

bit 15-12 **ESS<3:0>:** 例外シャドーセット ビット

このビットは、ベクタ割り込み以外の全ての例外に起因するカーネルモードへの移行時に使うシャドーセットを指定します。ソフトウェアがこのビットに HSS<3:0> ビットの値よりも大きな値を書き込んだ場合、プロセッサの動作は不確定です。

bit 11-10 **未実装 :** 「0」として読み出し

bit 9-6 **PSS<3:0>:** 前回のシャドーセット ビット

GPR シャドーレジスタが実装されているため、例外または割り込みが発生した時に、このビットには CSS ビットの内容がコピーされます。BEV ビット (Status<22>) = 0 の場合、ERET 命令はこのビットの値を CSS ビットに書き戻します。

このビットは、ERL ビット (Status<2>) を「1」に設定する全ての例外 (リセット、ソフトリセット、NMI、キャッシュエラー)、EJTAG デバッグモードへの移行、EXL ビット (Status<1>) = 1 または BEV = 1 に設定する全ての例外または割り込みが発生しても更新されません。このビットは、ERL = 1 の時に例外が発生しても更新されません。

ソフトウェアがこのビットに HSS<3:0> ビットの値よりも大きな値を書き込んだ場合、プロセッサの動作は不確定です。

bit 5-4 **未実装 :** 「0」として読み出し

bit 3-0 **CSS<3:0>:** 現在のシャドーセット ビット

GPR シャドーレジスタが実装されているため、このビットは現在の GPR セットの番号を格納します。このビットの値は、全ての割り込みまたは例外の発生時に更新され、ERET 命令によって PSS ビットから復元されます。

表 2-9 に、例外または割り込み発生時に CSS<3:0> ビットを更新する各種要因を示します。

このビットは、ERL ビット (Status<2>) を「1」に設定する全ての例外 (リセット、ソフトリセット、NMI、キャッシュエラー)、EJTAG デバッグモードへの移行、EXL ビット (Status<1>) = 1 または BEV = 1 に設定する全ての例外または割り込みが発生しても更新されません。また、ERL = 1 または BEV = 1 である場合、この値は ERET 命令によって更新されません。このビットは、ERL = 1 の時に例外が発生しても更新されません。

CSS<3:0> ビットをソフトウェアで直接変更するには、PSS<3:0> ビットに値を書き込んでから ERET 命令を実行する必要があります。

2.12.8 SRSMap: レジスタ (CP0 レジスタ 12、Select 3)

SRSMap レジスタは、ベクタ番号からシャドーセット番号へのマッピングを定義する 8 個の 4 ビットフィールドを格納します。これらは割り込みをサービスする際に使います。このレジスタの値は、非割り込み例外または非ベクタ割り込み (IV ビット = 0、Cause<23> または VS<4:0> ビット = 0、IntCtl<9:5>) には使いません。そのような場合、シャドーセット番号は ESS<3:0> ビット (SRSCtl<15:12>) から提供されます。

HSS<3:0> ビット (SRSCTL29:26) が「0」の場合、このレジスタに対するソフトウェア読み出しまたは書き込みの結果は不確定です。

このレジスタ内の任意のビットに HSS<3:0> ビットの値よりも大きな値を書き込んだ場合、プロセッサの動作は不確定です。

SRSMap レジスタは､ベクタ番号 7 ～ 0 に対応するシャドーレジスタ セット番号を格納します。ベクタから単一シャドーレジスタ セット番号への多対一対応マッピングを作成する事により、同じシャドーセット番号を複数の割り込みベクタに割り当てる事ができます。

レジスタ 2-8: SRSMap: シャドーレジスタ セットマップ レジスタ ; CP0 レジスタ 12、Select 3

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	SSV7<3:0>				SSV6<3:0>			
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	SSV5<3:0>				SSV4<3:0>			
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	SSV3<3:0>				SSV2<3:0>			
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	SSV1<3:0>				SSV0<3:0>			

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　　x = ビットは未知

bit 31-28 **SSV7<3:0>:** シャドーセット ベクタ 7 ビット
ベクタ番号 7 に割り当てるシャドーレジスタ セット番号

bit 27-24 **SSV6<3:0>:** シャドーセット ベクタ 6 ビット
ベクタ番号 6 に割り当てるシャドーレジスタ セット番号

bit 23-20 **SSV5<3:0>:** シャドーセット ベクタ 5 ビット
ベクタ番号 5 に割り当てるシャドーレジスタ セット番号

bit 19-16 **SSV4<3:0>:** シャドーセット ベクタ 4 ビット
ベクタ番号 4 に割り当てるシャドーレジスタ セット番号

bit 15-12 **SSV3<3:0>:** シャドーセット ベクタ 3 ビット
ベクタ番号 3 に割り当てるシャドーレジスタ セット番号

bit 11-8 **SSV2<3:0>:** シャドーセット ベクタ 2 ビット
ベクタ番号 2 に割り当てるシャドーレジスタ セット番号

bit 7-4 **SSV1<3:0>:** シャドーセット ベクタ 1 ビット
ベクタ番号 1 に割り当てるシャドーレジスタ セット番号

bit 3-0 **SSV0<3:0>:** シャドーセット ベクタ 0 ビット
ベクタ番号 0 に割り当てるシャドーレジスタ セット番号

2.12.9 Cause レジスタ (CP0 レジスタ 13、Select 0)

Cause レジスタは主に直前に発生した例外の原因に関する情報を提供します。また、割り込みを生成するためのソフトウェア割り込み要求とベクタを制御します。IP1、IP0、DC、IV ビットを除き、Cause レジスタ内のビットは全て読み出し専用です。

表 2-10: Cause レジスタ EXCCODE<4:0> ビット

例外コード値		ニーモニック	説明
10 進数	16 進数		
0	0x00	Int	割り込み
1-3	0x01	-	予約済み
4	0x04	AdEL	アドレスエラー例外 (ロードまたは命令フェッチ)
5	0x05	AdES	アドレスエラー例外 (ストア)
6	0x06	IBE	バスエラー例外 (命令フェッチ)
7	0x07	DBE	バスエラー例外 (データ参照 : ロードまたはストア)
8	0x08	Sys	システムコール例外
9	0x09	Bp	ブレークポイント例外
10	0x0A	RI	予約済み命令例外
11	0x0B	CPU	コプロセッサ使用不可例外
12	0x0C	Ov	算術オーバーフロー例外
13	0x0D	Tr	トラップ例外
14-31	0x0E-0x1F	-	予約済み

レジスタ 2-9: Cause: 例外原因レジスタ ; CP0 レジスタ 13、Select 0

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-x	R-x	R-x	R-x	R/W-0	U-0	U-0	U-0
	BD	TI	CE<1:0>		DC	-	-	-
23:16	R/W-x	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	IV	-	-	-	-	-	-	-
15:8	U-0	U-0	U-0	R-x	R-x	R-x	R/W-x	R/W-x
	-	-	-	RIPL<2:0>			IP1	IP0
7:0	U-0	R-x	R-x	R-x	R-x	R-x	U-0	U-0
	-	EXCCODE<4:0>					-	-

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　x = ビットは未知

bit 31 **BD:** 分岐遅延ビット
直前の例外が分岐遅延スロット内で発生したのかどうかを示します。
1 = 遅延スロット内で発生した
0 = 遅延スロット内で発生したのではない
例外が発生した時に EXL ビット (Status<1>) が「0」であった場合のみ、プロセッサは BD を更新します。

bit 30 **TI:** タイマ割り込みビット
タイマ割り込み このビットは、タイマ割り込みが保留中かどうかを示します (他の割り込みタイプ用の IP ビットと同様)。
1 = タイマ割り込みは保留中
0 = タイマ割り込みは保留中ではない

レジスタ 2-9: Cause: 例外原因レジスタ ; CP0 レジスタ 13、Select 0 (続き)

bit 29-28 **CE<1:0>:** コプロセッサ例外ビット
コプロセッサ使用不可例外が発生した時に参照されるコプロセッサのユニット番号です。ハードウェアは例外が発生するたびにこのビットを読み込みますが、コプロセッサ使用不可以外の全ての例外における値は不確定です。

bit 27 **DC:** カウント無効化レジスタビット
Count レジスタを使わない省電力性が求められる一部のアプリケーションでは、このビットを使ってカウントを停止する事により、不必要なトグリングを防げます。
1 = Count レジスタのカウントを無効にする
0 = Count レジスタのカウントを有効にする

bit 26-24 **未実装 :** 「0」として読み出し

bit 23 **IV:** 割り込みベクタビット
割り込み例外が通常の例外ベクタと特殊な割り込みベクタのどちらを使うのかを示します。
1 = 特殊な割り込みベクタ (0x200) を使う
0 = 通常の例外ベクタ (0x180) を使う
IV ビット (Cause<23>) が「1」かつ BEV ビット (Status<22>) が「0」の場合、特殊な割り込みベクタはベクタ割り込みテーブルのベースを表します。

bit 22-13 **未実装 :** 「0」として読み出し

bit 12-10 **RIPL<2:0>:** 要求中割り込みの優先度ビット
このビットは、現在要求されている割り込みの優先度のエンコード値 (7-0) です。「0」は要求中の割り込みが存在しない事を意味します。

bit 9-8 **IP<1:0>:** ソフトウェア割り込み要求制御ビット
ソフトウェア割り込み要求を制御します。
1 = ソフトウェア割り込みを要求する
0 = 割り込みを要求しない
これらのビットはシステム割り込みコントローラへエクスポートされ、そこで他の割り込み要因と一緒に EIC 割り込みモードで優先順位付けされます。

bit 7 **未実装 :** 「0」として読み出し

bit 6-2 **EXCCODE<4:0>:** 例外コードビット
例外コードは表 2-10 に記載しています。

bit 1-0 **未実装 :** 「0」として読み出し

2.12.10 EPC レジスタ (CP0 レジスタ 14、Select 0)

例外プログラム カウンタ (EPC) は読み書き可能レジスタです。このレジスタは、例外をサービスした後の処理の再開位置を指すアドレスを格納します。EPC レジスタの全てのビットは効果を持ち、書き込み可能です。

同期した (識別可能な) 例外の場合、EPC レジスタは以下のどれかを格納します。

- 例外の直接原因となった命令の仮想アドレス
- 例外を引き起こした命令が分岐遅延スロット内にあり、かつ、Cause レジスタの分岐遅延ビットがセットされている場合、直前の BRANCH または JUMP 命令の仮想アドレス

新しい例外が発生した時、Status レジスタの EXL ビットがセットされていると、プロセッサは EPC レジスタに書き込みませんが、MTC0 命令を使ってこのレジスタに書き込む事ができます。

PIC32 ファミリは MIPS16e® または microMIPS™ ASE を実装しているため、MFC0 命令による EPC レジスタの読み出しは、デスティネーション GPR に以下の値を返します。

$$GPR[rt] \leftarrow ExceptionPC_{31..1} \,||\, ISAMode_0$$

すなわち、例外 PC の上位 31 ビットに ISA<1:0> ビット (Config3<15:14>) の下位ビットを連結した内容が GPR に書き込まれます。

同様に、MTC0 命令による EPC レジスタへの書き込みは、GPR から読み出した値を、下記のように例外 PC と ISA<1:0> ビット (Config3<15:14>) に分配します。

$$ExceptionPC \leftarrow GPR[rt]_{31..1} \,||\, 0$$
$$ISAMode \leftarrow 2\#0 \,||\, GPR[rt]_0$$

すなわち、GPR の上位 31 ビットを例外 PC の上位 31 ビットに書き込み、下位 1 ビットをクリアします。また、ISA<1:0> ビット (Config3<15:14>) の上位ビットをクリアし、下位ビットには GPR の下位ビットを読み込みます。

レジスタ 2-10: EPC: 例外プログラム カウンタ レジスタ ; CP0 レジスタ 14、Select 0

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	EPC<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	EPC<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	EPC<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	EPC<7:0>							

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　　x = ビットは未知

bit 31-0 **EPC<31:0>:** 例外プログラム カウンタビット

2.12.11 PRID レジスタ (CP0 レジスタ 15、Select 0)

プロセッサ ID (PRID) レジスタは 32 ビットの読み出し専用レジスタです。このレジスタは、プロセッサに関する情報 (製造者、製造者オプション、ID、リビジョンレベル) を格納します。

レジスタ 2-11: PRID: プロセッサ ID レジスタ ; CP0 レジスタ 15、Select 0

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	R-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
23:16	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-1
	COMPANYID<23:16>							
15:8	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	PROCESSORID<15:8>							
7:0	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	MAJORREV<2:0>			MINORREV<2:0>			PATCHREV<1:0>	

凡例 :	凡例 :	
	R = 読み出し可能ビット	W = 書き込み可能ビット P = プログラム可能ビット r = 予約済みビット
	U = 未実装ビット	n = POR 時のビット値 (「0」、「1」または x = ビットは未知)

bit 31-24 **未実装 :** 「0」として読み出し

bit 23-16 **COMPANYID<7:0>:** 製造者 ID ビット
PIC32 の場合、これらのビットは「1」を格納し、プロセッサの製造 / 設計者として MIPS® Technologies, Inc. を表します。

bit 15-8 **PROCESSORID<7:0>:** プロセッサ ID ビット
これらのビットを使うと、ソフトウェアが MIPS® Technologies 社製プロセッサの各種タイプを識別できます。M4K® コアを搭載した PIC32 の値は 0x87 です。

bit 7-5 **MAJORREV<2:0>:** プロセッサ メジャー リビジョン ID ビット
ソフトウェアは、これらのビットを使って同一プロセッサタイプのリビジョン間の違いを識別できます。この値は、プロセッサコアの主要な変更のたびに増えます。

bit 4-2 **MINORREV<2:0>:** プロセッサ マイナー リビジョン ID ビット
この値は、プロセッサの小規模な変更のたびに増えます。メジャー リビジョンが新しくなるたびに、この値はリセットされます。

bit 1-0 **PATCHREV<1:0>:** プロセッサ パッチレベル ID ビット
プロセッサの旧リビジョンにパッチによる変更が加えられるたびに、これらのビットの値が増えます。

2.12.12 Ebase レジスタ (CP0 レジスタ 15、Select 1)

Ebase レジスタは読み書き可能なレジスタです。このレジスタは、BEV ビット (Status<22>) が「0」の時に使う例外ベクタのベースアドレスと、読み出し専用の CPU 番号を格納します。CPU 番号は、マルチプロセッサ システムでソフトウェアが異なるプロセッサを識別するために使います。

ソフトウェアは、Ebase レジスタを使って、マルチプロセッサ システム内の特定のプロセッサを識別でき、特に異種プロセッサから構成されるシステムでは、プロセッサごとに例外ベクタを変える事ができます。Ebase レジスタの bit 31 ～ 12 にゼロを連結する事により、BEV ビットが「0」の場合の例外ベクタのベースアドレスを形成します。BEV ビットが「1」の場合、または EJTAG デバッグ例外が発生した場合、例外ベクタのベースアドレスには固定された既定値を使います。Ebase レジスタの bit 31 ～ 12 のリセット状態は、例外ベースレジスタを 0x80000000 に初期化します。

Ebase レジスタの bit 31 ～ 30 の値は 2#10 に固定されており、例外ベースアドレスを kseg0 または kseg1 (マッピングされない仮想アドレス セグメント) 内に配置します。

例外ベースレジスタの値は、BEV ビットが「1」の時に変更する必要があります。BEV ビット (Status<22>) が「0」の時に Ebase<17:0> ビットに現在の値とは異なる値を書き込んだ場合、プロセッサの動作は不確定です。

Ebase レジスタの bit 31 ～ 20 を組み合わせる事により、例外ベクタのベースアドレスを任意の 4 KB ページ境界に配置できます。

レジスタ 2-12: Ebase: 例外ベースレジスタ ; CP0 レジスタ 15、Select 1

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-1	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	-	-	EBASE<17:12>					
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	EBASE<11:4>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R-0	R-0
	EBASE<3:0>				-	-	CPUNUM<9:8>	
7:0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	CPUNUM<7:0>							

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　x = ビットは未知

bit 31 **未実装 :** 「1」として読み出し

bit 30 **未実装 :** 「0」として読み出し

bit 29-12 **EBASE<17:0>:** 例外ベクタ ベースアドレス ビット
bit 31 ～ 30 との組み合わせにより、BEV ビット (Status<22>) が「0」の時の例外ベクタのベースアドレスを指定します。

bit 11-10 **未実装 :** 「0」として読み出し

bit 9-0 **CPUNUM<9:0>:** CPU 番号ビット
これらのビットは、マルチプロセッサ システム内の CPU の番号を指定します。ソフトウェアは、この番号を使って特定のプロセッサを識別できます。シングル プロセッサ システムでは、この値は「0」に設定されます。

2.12.13 Config レジスタ (CP0 レジスタ 16、Select 0)

Config レジスタは、各種のコンフィグレーションと機能に関する情報を定義します。Config レジスタ内の大部分のフィールドは、リセット例外処理中にハードウェアによって初期化されます (一部のフィールドは値を保持)。

表 2-11: キャッシュ コヒーレンシ属性

K0<2:0> の値	キャッシュ コヒーレンシ属性
2	キャッシュしない
3	キャッシュ可能

レジスタ 2-13: Config: コンフィグレーション レジスタ ; CP0 レジスタ 16、Select 0

ビット レンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	r-1	R-0	R-1	R-0	R/W-0	R/W-1	R/W-0	U-0
	-	K23<2:0>			KU<2:0>			-
23:16	U-0	R-0	R-0	r-0	U-0	U-0	U-0	R-1
	-	UDI	SB	-	-	-	-	DS
15:8	R-0	R-0	R-0	R-0	R-0	R-1	R-0	R-1
	BE	AT<1:0>		AR<2:0>			MT<2:1>	
7:0	R-1	U-0	U-0	U-0	U-0	R/W-0	R/W-1	R/W-0
	MT<1>	-	-	-	-	K0<2:0>		

凡例 :			
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し	
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア	x = ビットは未知

bit 31 **予約済み :** このビットは、Config1 レジスタが存在する事を示すために、ハードウェアで「1」に結線されています。

bit 30-28 **K23<2:0>:** kseg2 と kseg3 ビット
これらのビットは、kseg2 と kseg3 アドレス セグメントのキャッシュの可否を制御します。
ビットのエンコードは表 2-11 を参照してください。

bit 27-25 **KU<2:0>:** Kuseg と useg ビット
これらのビットは、kuseg と useg アドレス セグメントのキャッシュの可否を制御します。
ビットのエンコードは表 2-11 を参照してください。

bit 24-23 **未実装 :** 「0」として読み出し

bit 22 **UDI:** ユーザ定義ビット
このビットは、CorExtend ユーザ定義命令が実装されている事を示します。
1 = ユーザ定義命令が実装されている
0 = ユーザ定義命令は実装されていない

bit 21 **SB:** SimpleBE ビット
このビットは、SimpleBE バスモードが有効かどうかを示します。
1 = 内部バス インターフェイス上で SimpleBE (Simple バイトイネーブル) だけを許可する
0 = 内部バス インターフェイス上で予約済みバイトイネーブルを使わない

bit 20 **予約済み :** このビットは、高速で高性能な乗除算ユニット (MDU) の存在を示すために、ハードウェアで「0」に結線されています。

bit 19-17 **未実装 :** 「0」として読み出し

レジスタ 2-13: Config: コンフィグレーション レジスタ ; CP0 レジスタ 16、Select 0 (続き)

bit 16 **DS:** デュアル SRAM ビット
1 = 命令 / データ用にデュアル SRAM 内部バス インターフェイスを使う
0 = 命令 / データ用に一元化された SRAM 内部バス インターフェイスを使う
M4K® コアに基づく PIC32 はデュアル SRAM 型インターフェイスを内部で使います。

bit 15 **BE:** ビッグエンディアン ビット
プロセッサがどちらのエンディアン モードで動作しているのかを示します。PIC32は常にリトルエンディアンです。
1 = ビッグエンディアン
0 = リトルエンディアン

bit 14-13 **AT<1:0>:** アーキテクチャ タイプビット
プロセッサが実装しているアーキテクチャのタイプを示します。これらのビットは常に「00」に設定され、MIPS32® アーキテクチャである事を示します。

bit 12-10 **AR<2:0>:** アーキテクチャ リビジョンレベル ビット
アーキテクチャのリビジョンレベルを示します。これらのビットは常に「001」に設定され、MIPS32® リリース 2 である事を示します。
111 = 予約済み
110 = 予約済み
101 = 予約済み
100 = 予約済み
011 = 予約済み
010 = 予約済み
001 = リリース 2
000 = リリース 1

bit 9-7 **MT<2:0>:** MMU タイプビット
111 = 予約済み
110 = 予約済み
101 = 予約済み
100 = 予約済み
011 = 固定
010 = 予約済み
001 = 予約済み
000 = 予約済み

bit 6-3 **未実装 :** 「0」として読み出し

bit 2-0 **K0<2:0>:** Kseg0 ビット
Kseg0 コヒーレンシ アルゴリズムです。ビットのエンコードは表 2-11 を参照してください。

2.12.14 Config1 レジスタ (CP0 レジスタ 16、Select 1)

Config1 レジスタは、Config レジスタの補助として、コアが実装する機能に関する追加の情報をエンコードします。Config1 レジスタ内のフィールドは全て読み出し専用です。

レジスタ 2-14: Config1: コンフィグレーション レジスタ 1; CP0 レジスタ 16、Select 1

ビット レンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	r-1	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
7:0	U-0	U-0	U-0	U-0	U-0	R-1	R-x	U-0
	-	-	-	-	-	CA	EP	-

凡例 :	r = 予約済みビット	
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　x = ビットは未知

bit 31 **予約済み :** このビットは、Config2 レジスタが存在する事を示すために、ハードウェアで「1」に結線されています。

bit 30-3 **未実装 :** 「0」として読み出し

bit 2 **CA:** コード圧縮実装ビット
1 = MIPS16e® を実装している
0 = MIPS16e® を実装していない

bit 1 **EP:** EJTAG 実装ビット
このビットは常に「1」に設定され、コアが EJTAG を実装している事を示します。

bit 0 **未実装 :** 「0」として読み出し

2.12.15 Config2 (CP0 レジスタ 16、Select 2)

Config2 レジスタは、Config レジスタの補助として、コアが実装する追加機能に関する情報をエンコードするために予約済みです。Config2 レジスタは、レベル 2/3 のキャッシュのコンフィグレーションを示します。PIC32 コアは、L2/L3 キャッシュをサポートしないため、これらのビットは「0」にリセットされます。Config2 レジスタ内のビットは全て読み出し専用です。

レジスタ 2-15: Config2: コンフィグレーション レジスタ 2; CP0 レジスタ 16、Select 2

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	r-1	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
7:0	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-

凡例：	r = 予約済みビット		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し	
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア	x = ビットは未知

bit 31 **予約済み：**このビットは、Config3 レジスタが存在する事を示すために、ハードウェアで「1」に結線されています。

bit 30-0 **未実装：**「0」として読み出し

2.12.16 Config3 レジスタ (CP0 レジスタ 16、Select 3)

Config3 レジスタは、追加機能に関する情報をエンコードします。Config3 レジスタ内のフィールドは全て読み出し専用です。

レジスタ 2-16: Config3: コンフィグレーション レジスタ 3; CP0 レジスタ 16、Select 3

ビット レンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	R-0
	-	-	-	-	-	-	-	ITL
7:0	U-0	R-1	R-1	U-0	U-0	U-0	U-0	U-0
	-	VEIC	VINT	-	-	-	-	-

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　x = ビットは未知

bit 31-9 **未実装 :** 「0」として読み出し

bit 8 **ITL:** iFlowTrace ハードウェアを実装しているかどうかを示します。
1 = コアは iFlowTrace を実装している
0 = コアは iFlowTrace を実装していない

bit 7 **未実装 :** 「0」として読み出し

bit 6 **VEIC:** 外部ベクタ割り込みコントローラ ビット
外部割り込みコントローラをサポートするかどうかを示します。
1 = EIC 割り込みモードをサポートする
0 = EIC 割り込みモードをサポートしない
PIC32 は MIPS® 「外部割り込みコントローラ」を内蔵しています。従って、このビットの読み出し値は「1」です。

bit 5 **VINT:** ベクタ割り込みビット
ベクタ割り込みを実装しているかどうかを示します。このビットは、ベクタ割り込みを実装しているかどうかを示します。
1 = ベクタ割り込みを実装している
0 = ベクタ割り込みを実装していない
PIC32 コアはベクタ割り込みを実装しているため、このビットは常に「1」です。

bit 4-0 **未実装 :** 「0」として読み出し

2.12.17 Debug レジスタ (CP0 レジスタ 23、Select 0)

Debug レジスタは、デバッグ例外を制御し、その原因に関する情報を提供します。また、デバッグモードで通常の例外が発生したためにデバッグ例外ベクタ位置で再入した時の情報を提供します。読み出し専用の情報ビットは、デバッグ例外が発生するたびに、または、既にデバッグモードで動作している間に通常の例外が発生した時に、更新されます。

デバッグ以外のモードからの読み出しに対して、DM ビットと VER<2:0> ビットだけは有効な値を返しますが、その他のビットとフィールドの読み出し値は全て予測不能です。デバッグ以外のモードから Debug レジスタが書き込まれた場合、プロセッサの動作は不確定です。

一部のビットとフィールドは、「デバッグ例外」または「デバッグモードでの通常例外」(または、その両方) が発生した時にのみ、下記のように更新されます。

- DSS、DBP、DDBL、DDBS、DIB、DINT は、「デバッグ例外」と「デバッグモードでの通常例外」の両方で更新されます。
- DEXCCODE<4:0> は、「デバッグモードでの通常例外」で更新されますが、「デバッグ例外」後の状態は未定義です。
- HALT と DOZE は「デバッグ例外」で更新され、「デバッグモードでの通常例外」後の状態は未定義です。
- DBD は、「デバッグ例外」と「デバッグモードでの通常例外」の両方で更新されます。

VER<2:0> と DM 以外の全てのビットは、通常モードからの読み出しに対して未定義です。

レジスタ 2-17: Debug: デバッグ例外レジスタ ; CP0 レジスタ 23、Select 0

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-0	R-0	R-0	R/W-0	U-0	U-0	R/W-1	R/W-0
	DBD	DM	NODCR	LSNM	DOZE	HALT	COUNTDM	IBUSEP
23:16	R-0	R-0	R/W-0	R/W-0	R-0	R-0	R-0	R-1
	MCHECKP	CACHEEP	DBUSEP	IEXI	DDBSIMPR	DDBLIMPR	VER<2:1>	
15:8	R-0	U-0	U-0	U-0	U-0	U-0	R-0	R/W-0
	VER	DEXCCODE<4:0>					NOSST	SST
7:0	U-0	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	-	DIBIMPR	DINT	DIB	DDBS	DDBL	DBP	DSS

凡例 :			
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し	
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア	x = ビットは未知

bit 31 **DBD:** 分岐遅延デバッグ例外ビット
直前の「デバッグ例外」または「デバッグモードでの通常例外」が分岐遅延スロット内で発生したのかどうかを示します。
1 = 遅延スロット内で発生した
0 = 遅延スロット内で発生したのではない

bit 30 **DM:** デバッグモード ビット
プロセッサがデバッグモードで動作しているかどうかを示します。
1 = プロセッサはデバッグモードで動作している
0 = プロセッサはデバッグモード以外で動作している

bit 29 **NODCR:** デバッグ制御レジスタビット
dseg メモリセグメントが存在する (デバッグ制御レジスタにアクセス可能) かどうかを示します。
1 = dseg は存在しない
0 = dseg は存在する

レジスタ 2-17: Debug: デバッグ例外レジスタ ; CP0 レジスタ 23、Select 0 (続き)

bit 28 **LSNM:** ロード / ストアアクセス制御ビット
dseg とメインメモリ間のロード / ストアアクセスを制御します。
1 = dseg アドレス領域内のロード / ストアはメインメモリにアクセスする
0 = dseg アドレス領域内のロード / ストアは dseg にアクセスする

bit 27 **DOZE:** 低消費電力モードデバッグ例外ビット
デバッグ例外が発生した時にプロセッサが何らかの低消費電力モードであったかどうかを示します。
1 = デバッグ例外発生時にプロセッサは低消費電力モードであった
0 = デバッグ例外発生時にプロセッサは低消費電力モードでなかった

bit 26 **HALT:** システムバス クロック停止ビット
デバッグ例外発生時に内部システムバス クロックが停止していたかどうかを示します。
1 = 内部システムバス クロックは動作していた
0 = 内部システムバス クロックは動作していた

bit 25 **COUNTDM:** Count レジスタ挙動ビット
デバッグモードにおける Count レジスタの挙動を示します。
1 = Count レジスタはデバッグモードで動作する
0 = Count レジスタはデバッグモードで停止する

bit 24 **IBUSEP:** 命令フェッチ バスエラー例外保留ビット
このビットは、命令フェッチ バスエラー イベントが発生した時、またはソフトウェアでこのビットに「1」を書き込んだ時にセットされます。このビットは、プロセッサが命令フェッチにおけるバスエラー例外を処理した時、およびリセットによってクリアされます。IEXI がクリアされている時に IBUSEP がセットされた場合、プロセッサが命令フェッチ時のバスエラー例外を処理した後に、IBUSEP はクリアされます。

bit 23 **MCHECKP:** マシンチェック例外保留ビット
PIC32 プロセッサでは、全てのマシンチェック例外は識別可能であるため、このビットは常に「0」として読み出されます。

bit 22 **CACHEEP:** キャッシュエラー保留ビット
PIC32 コアはキャッシュエラーを処理できないため、このビットは常に「0」として読み出されます。

bit 21 **DBUSEP:** データアクセス バスエラー例外保留ビット
このビットは、データアクセス バス上の識別困難なバスエラーに対応します。このビットの挙動は、命令フェッチで識別困難なバスエラーが発生した時の IBUSEP ビットの挙動と同様です。

bit 20 **IEXI:** 識別困難エラー例外抑止制御ビット
識別困難なエラー表示に起因する例外の発生を抑制します。このビットは、プロセッサが「デバッグ例外」または「デバッグモードでの通常例外」を処理する際にセットされます。このビットは、DERET 命令の実行によりクリアされます。それ以外に、デバッグモード ソフトウェアによる変更が可能です。IEXI がセットされている場合、命令フェッチまたはデータアクセス時のバスエラー、キャッシュエラー、マシンチェックに起因する識別困難なエラー例外は抑止され、このビットがクリアされるまで延期されます。

bit 19 **DDBSIMPR:** デバッグデータ ブレークストア例外ビット
識別困難なデバッグデータ ブレークストア例外が処理された事を示します。PIC32 コアでは、全てのデータブレークは識別可能であるため、このビットは常に「0」として読み出されます。

bit 18 **DDBLIMPR:** デバッグデータ ブレークロード例外ビット
識別困難なデバッグデータ ブレークロード例外が処理された事を示します。PIC32 コアでは、全てのデータブレークは識別可能であるため、このビットは常に「0」として読み出されます。

bit 17-15 **VER<2:0>:** EJTAG バージョンビット
EJTAG のバージョン番号を格納します。

bit 14-10 **DEXCCODE<4:0>:** デバッグモードにおける直前例外ビット
デバッグモードで発生した直前の例外の原因を示します。このビットは、デバッグモードで発生する可能性のある通常例外の原因を、Cause レジスタの EXCCODE<4:0> ビットと同様のコードで示します。「デバッグ例外」後のこのビットの値は未定義です。

bit 9 **NOSST:** シングルステップ機能制御ビット
この実装で SST ビットによるシングルステップ機能の制御が可能かどうかを示します。
1 = シングルステップ機能は利用できない
0 = シングルステップ機能を利用できる

レジスタ 2-17: Debug: デバッグ例外レジスタ ; CP0 レジスタ 23、Select 0 (続き)

bit 8 **SST:** デバッグ シングルステップ制御ビット
デバッグ シングルステップ例外を有効または無効にします。
1 = デバッグ シングルステップ例外を有効にする
0 = デバッグ シングルステップ例外を無効にする

bit 7 **未実装 :** 「0」として読み出し

bit 6 **DIBImpr:** 識別困難なデバッグ命令ブレーク例外ビット
複雑なブレークポイントに起因する識別困難なデバッグ命令ブレーク例外の発生を示します。「デバッグモードでの通常例外」時にクリアされます。

bit 5 **DINT:** デバッグ割り込み例外ビット
デバッグ割り込み例外の発生を示します。「デバッグモードでの通常例外」時にクリアされます。
1 = デバッグ割り込み例外が発生した
0 = デバッグ割り込み例外は発生していない

bit 4 **DIB:** デバッグ命令ブレーク例外ビット
デバッグ命令ブレーク例外の発生を示します。「デバッグモードでの通常例外」時にクリアされます。
1 = デバッグ命令例外が発生した
0 = デバッグ命令例外は発生していない

bit 3 **DDBS:** ストア時デバッグデータ ブレーク例外ビット
ストア時にデバッグデータ ブレーク例外が発生した事を示します。「デバッグモードでの通常例外」時にクリアされます。
1 = ストア時にデバッグ命令例外が発生した
0 = ストア時にデバッグデータ例外は発生していない

bit 2 **DDBL:** ロード時デバッグデータ ブレーク例外ビット
ロード時にデバッグデータ ブレーク例外が発生した事を示します。「デバッグモードでの通常例外」時にクリアされます。
1 = ロード時にデバッグ命令例外が発生した
0 = ロード時にデバッグデータ例外は発生していない

bit 1 **DBP:** デバッグ ソフトウェア ブレークポイント例外ビット
デバッグ ソフトウェア ブレークポイント例外の発生を示します。「デバッグモードでの通常例外」時にクリアされます。
1 = デバッグ ソフトウェア ブレークポイント例外が発生した
0 = デバッグ ソフトウェア ブレークポイント例外は発生していない

bit 0 **DSS:** デバッグ シングルステップ例外ビット
デバッグ シングルステップ例外の発生を示します。「デバッグモードでの通常例外」時にクリアされます。
1 = デバッグ シングルステップ例外が発生した
0 = デバッグ シングルステップ例外は発生していない

2.12.18 TraceControl レジスタ (CP0 レジスタ 23、Select 1)

TraceControl レジスタは、ソフトウェア トレース制御を可能にします。このレジスタは、EJTAG トレース機能を備えたデバイスだけが実装しています。

レジスタ 2-18: TraceControl: トレース制御レジスタ ; CP0 レジスタ 23、Select 1

ビット レンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	TS	UT	-	-	TB	IO	D(1)	E(1)
23:16	R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
	K(1)	-	U(1)	-	-	-	-	-
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
7:0	U-0	U-0	U-0	U-1	R/W-0	R/W-0	R/W-0	R/W-0
	-	-	-	-	MODE<2:0>			IB

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　　x = ビットは未知

bit 31 **TS:** トレース選択ビット
このビットでは、ハードウェアまたはソフトウェアどちらかのトレース制御を選択します。
1 = トレース制御ビットを選択する
0 = 外部のハードウェア トレースブロック信号を選択する

bit 30 **UT:** ユーザタイプ選択ビット
このビットでは、ユーザトリガ トレースレコードのタイプを選択します。
1 = ユーザタイプ 2
0 = ユーザタイプ 1

bit 29-28 **未実装 :** 「0」として読み出し

bit 27 **TB:** トレース分岐ビット
1 = 全ての分岐の実行で PC 値をトレースする
0 = 分岐先アドレスが予測不能な静的アドレスである場合に PC 値をトレースする

bit 26 **IO:** オーバーフロー抑制ビット
この信号は、コアのトレースロジックに、速度を落としてでも完全なトレース実行を指示するために使います。
1 = FIFO オーバーフローまたはトレースデータの破棄を抑制する
0 = FIFO オーバーフローまたはトレースデータの破棄を容認する

bit 25 **D:** デバッグモード トレース イネーブルビット (1)
1 = デバッグモードでのトレースを有効にする
0 = デバッグモードでのトレースを無効にする

bit 24 **E:** 例外モード トレース イネーブルビット (1)
1 = 例外モードでのトレースを有効にする
0 = 例外モードでのトレースを無効にする

bit 23 **K:** カーネルモード トレース イネーブルビット (1)
1 = カーネルモードでのトレースを有効にする
0 = カーネルモードでのトレースを無効にする

bit 22 **未実装 :** 「0」として読み出し

Note 1: トレースを有効にするには、ON ビットを「1」にセットする必要があります。

レジスタ 2-18: TraceControl: トレース制御レジスタ ; CP0 レジスタ 23、Select 1 (続き)

bit 21 **U:** ユーザモード トレース イネーブルビット (1)
1 = ユーザモードでのトレースを有効にする
0 = ユーザモードでのトレースを無効にする

bit 20-5 **未実装 :** 「0」として読み出し

bit 4 **未実装 :** 「1」として読み出し

bit 3-1 **MODE<2:0>:** トレースモード制御ビット
111 = PC をトレースし、アドレスとデータをロードおよびストアする
110 = PC をトレースし、アドレスとデータをストアする
101 = PC をトレースし、アドレスとデータをロードする
100 = PC をトレースし、データをロードする
011 = PC をトレースし、アドレスをロードおよびストアする
010 = PC をトレースし、アドレスをストアする
001 = PC をトレースし、アドレスをロードする
000 = PC をトレースする

bit 0 **ON:** マスタトレース イネーブルビット
1 = 他のトレース イネーブルビットが「1」に設定された時にトレースを有効にする
0 = トレースを無効にする

Note 1: トレースを有効にするには、ON ビットを「1」にセットする必要があります。

2.12.19 TraceControl2 レジスタ (CP0 レジスタ 23、Select 2)

TraceControl2 レジスタは、トレースに関する追加の制御とステータス情報を提供します。TraceControl2 レジスタ内の一部のフィールドは読み出し専用ですが、リセット状態が「未定義」である事に注意が必要です。これは、これらの値がトレース制御ブロック (TCB) から読み込まれるためです。このため、TraceControl2 レジスタ内のこれらのフィールドは、TCB がこれらの値をアサートするまで、有効な値を格納しません。

レジスタ 2-19: TraceControl2: トレース制御レジスタ 2; CP0 レジスタ 23、Select 2

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
7:0	U-0	R-1	R-0	R-x	R-x	R-x	R-x	R-x
	-	VALIDMODES<1:0>		TBI	TBU	SYP<2:0>		

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　　x = ビットは未知

bit 31-7 **未実装 :** 「0」として読み出し

bit 6-5 **VALIDMODES<1:0>:** 有効トレースモード選択ビット
11 =予約済み
10 =PC、アドレスのロードおよびストア、データのロードおよびストアをトレースする
01 =PC とアドレスのロードとストアだけをトレースする
00 =PC だけをトレースする

bit 4 **TBI:** トレースバッファ実装ビット
1 = TCB は内蔵と外部のトレースバッファを実装する
0 = 1 つのトレースバッファだけを実装する

bit 3 **TBU:** トレースバッファ使用ビット
1 = トレースデータを外部のトレースバッファに転送する
0 = トレースデータを内蔵トレースバッファに転送する

bit 2-0 **SYP<2:0>:** 同期周期ビット
下表の「内蔵」列の値は、トレースデータを内蔵トレースバッファに書き込む場合 ($TraceControl2_{TBU}$ = 0) に適用されます。「外部」列の値は、トレースデータを外部のトレースバッファに書き込む場合 ($TraceControl2_{TBU}$ = 1) に適用されます。

ビット設定	内蔵	外部
111 =	2^2	2^7
110 =	2^3	2^8
101 =	2^4	2^9
100 =	2^5	2^{10}
011 =	2^6	2^{11}
010 =	2^7	2^{12}
001 =	2^8	2^{13}
000 =	2^9	2^{14}

2.12.20 UserTraceData レジスタ (CP0 レジスタ 23、Select 3)

ソフトウェアが UserTraceData レジスタ内のビットのどれかに書き込むと、タイプ 1 またはタイプ 2 のユーザ フォーマットでトレースレコードの書き込みが始まります。タイプは、TraceControl レジスタの UT ビットによって決まります。このレジスタには、連続したサイクルで書き込む事はできません。このレジスタに連続したサイクルで書き込んだ場合のトレース出力データは不確定です。

このビットは、EJTAG トレース機能を備えたデバイスだけが実装しています。

レジスタ 2-20: UserTraceData: ユーザトレース データ制御レジスタ ; CP0 レジスタ 23、Select 3

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DATA<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DATA<23:10>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DATA<15:6>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DATA<7:0>							

凡例 :			
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し	
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア	x = ビットは未知

bit 31-0 **DATA:** ソフトウェア読み書き可能データビット
このレジスタに書き込むと、ユーザ フォーマットによるトレースレコードの出力が始まり、PDtrace インターフェイスからトレースメモリにデータビットが転送されます。

2.12.21 TraceBPC レジスタ (CP0 レジスタ 23、Select 4)

このレジスタは､EJTAG ハードウェア ブレークポイントを使ってトレースの開始 / 終了を制御するために使います。ハードウェア ブレークポイントはトリガ源として設定されます。オプションにより、デバッグ例外ブレークポイントとして設定する事ができます。

このレジスタは、ハードウェア ブレークポイント機能と EJTAG トレース機能の両方を備えたデバイスだけが実装しています。

レジスタ 2-21: TraceBPC: トレース ブレークポイント制御レジスタ ; CP0 レジスタ 23、Select 4

ビット レンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	DE	-	-	-	-	-	-	-
23:16	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
	-	-	-	-	-	-	DBPO1	DBPO0
15:8	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	IE	-	-	-	-	-	-	-
7:0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	-	-	IBPO5	IBPO4	IBPO3	IBPO2	IBPO1	IBPO0

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　　x = ビットは未知

bit 31 **DE:** EJTAG データ ブレークポイント トリガ選択ビット
1 = データ ブレークポイントからのトリガ信号を有効にする
0 = データ ブレークポイントからのトリガ信号を無効にする

bit 15 **IE:** EJTAG 命令ブレークポイント選択ビット
1 = 命令ブレークポイントからのトリガ信号を有効にする
0 = 命令ブレークポイントからのトリガ信号を無効にする

bit 14-6 **未実装 :** 「0」として読み出し

bit 5 **IBPO5:** 命令ブレークポイント 6 ビット
1 = 対応する命令ブレークポイント トリガを有効にしてトレースを始める
0 = トリガ信号によるトレースを無効にする

bit 4 **IBPO4:** 命令ブレークポイント 5 ビット
1 = 対応する命令ブレークポイント トリガを有効にしてトレースを始める
0 = トリガ信号によるトレースを無効にする

bit 3 **IBPO3:** 命令ブレークポイント 4 ビット
1 = 対応する命令ブレークポイント トリガを有効にしてトレースを始める
0 = トリガ信号によるトレースを無効にする

bit 2 **IBPO2:** 命令ブレークポイント 3 ビット
1 = 対応する命令ブレークポイント トリガを有効にしてトレースを始める
0 = トリガ信号によるトレースを無効にする

bit 1 **IBPO1:** 命令ブレークポイント 2 ビット
1 = 対応する命令ブレークポイント トリガを有効にしてトレースを始める
0 = トリガ信号によるトレースを無効にする

bit 0 **IBPO0:** 命令ブレークポイント 1 ビット
1 = 対応する命令ブレークポイント トリガを有効にしてトレースを始める
0 = トリガ信号によるトレースを無効にする

Note 1: 利用可能なトリガ源は、各デバイスのデータシートを参照してください。

2.12.22 Debug2 レジスタ (CP0 レジスタ 23、Select 5)

このレジスタは、複雑なブレークポイント例外に関する追加の情報を保持します。このレジスタは、複雑なハードウェア ブレークポイントが存在する場合にのみ実装されています。

レジスタ 2-22: Debug2: デバッグ ブレークポイント例外レジスタ ; CP0 レジスタ 23、Select 5

ビット レンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	r-1	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	-	-	-	-	-	-	-	-
7:0	U-0	U-0	U-0	U-0	R-x	R-x	R-x	R-x
	-	-	-	-	PRM	DQ	TUP	PACO

凡例 :	r = 予約済みビット	
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　　x = ビットは未知

bit 31 **予約済み :** 「1」として読み出し

bit 30-4 **未実装 :** 「0」として読み出し

bit 3 **PRM:** Primed ビット
直前のデバッグ例外で、アクティブな priming 条件を持つ複雑なブレークポイントに遭遇したかどうかを示します。

bit 2 **DQ:** データ修飾子ビット
直前のデバッグ例外で、アクティブなデータ修飾子を持つ複雑なブレークポイントに遭遇したかどうかを示します。

bit 1 **TUP:** タプル ブレークポイント ビット
直前のデバッグ例外で、タプル ブレークポイントに遭遇したかどうかを示します。

bit 0 **PACO:** パスカウンタ ビット
直前のデバッグ例外で、アクティブなパスカウンタを持つ複雑なブレークポイントに遭遇したかどうかを示します。

2.12.23 DEPC レジスタ (CP0 レジスタ 24、Select 0)

デバッグ例外プログラム カウンタ (DEPC) レジスタは読み書き可能レジスタです。このレジスタは、「デバッグ例外」または「デバッグモードでの通常例外」をサービスした後の処理の再開位置を指すアドレスを格納します。

同期した (識別可能な)「デバッグ例外」と「デバッグモードでの通常例外」の場合、DEPC レジスタは下記のどれかを格納します。

- デバッグ例外の直接的原因となった命令の仮想アドレス
- デバッグ例外を引き起こした命令が分岐遅延スロット内にあり、かつ、Debug レジスタのデバッグ分岐遅延 (DBD) ビットがセットされていた場合、直前の分岐またはジャンプ命令の仮想アドレス

非同期のデバッグ例外 (デバッグ割り込み) の場合、DEPC レジスタは、デバッグハンドラ コードを実行した後に実行を再開する命令の仮想アドレスを格納します。

PIC32 ファミリの一部のデバイスは MIPS16e® ASE を実装しているため、MFC0 命令による DEPC レジスタの読み出しは、デスティネーション GPR に以下の値を返します。

$GPR[rt] = DebugExceptionPC_{31..1} \ || \ ISAMode_0$

すなわち、デバッグ例外 PC の上位 31 ビットに ISA<1:0> ビット (Config3<15:14>) の下位ビットを連結した内容が GPR に書き込まれます。

同様に、MTC0 命令による DEPC レジスタへの書き込みは、GPR から読み出した値を、下記のようにデバッグ例外 PC と ISA<1:0> ビット (Config3<15:14>) に分配します。

$DebugExceptionPC = GPR[rt]_{31..1} \ || \ 0$
$ISAMode = 2\#0 \ || \ GPR[rt]_0$

すなわち、GPR の上位 31 ビットをデバッグ例外 PC の上位 31 ビットに書き込み、デバッグ例外 PC の下位ビットはクリアします。また、ISA<1:0> ビット (Config3<15:14>) の上位ビットをクリアし、下位ビットには GPR の下位ビットを読み込みます。

レジスタ 2-23: DEPC: デバッグ例外プログラム カウンタレジスタ ; CP0 レジスタ 24、Select 0

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DEPC<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DEPC<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DEPC<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DEPC<7:0>							

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　　x = ビットは未知

bit 31-0 **DEPC<31:0>:** デバッグ例外プログラム カウンタビット

DEPC レジスタの値は、デバッグ例外を引き起こした命令の仮想アドレス値に更新されます。その命令が分岐遅延スロット内にある場合、このレジスタは、直前の分岐またはジャンプ命令の仮想アドレスを格納します。

DERET 命令を実行すると、DEPC レジスタ内のアドレスへのジャンプが発生します。

2.12.24 ErrorEPC (CP0 レジスタ 30、Select 0)

ErrorEPC レジスタは読み書き可能レジスタです。ErrorEPC レジスタはエラー例外時に使います。その挙動は通常の例外時に使う EPC レジスタに類似しています。ErrorEPC レジスタの全てのビットは効果を持ち、書き込み可能です。このレジスタは、リセット、ソフトリセット、ノンマスカブル割り込み (NMI) 例外時にプログラム カウンタを保存するためにも使います。

ErrorEPC レジスタは、エラー処理後に命令処理を再開可能な位置を指す仮想アドレスを格納します。このアドレスは以下のどれかです。

- エラー例外の原因となった命令の仮想アドレス
- エラー例外の原因となった命令が分岐遅延スロット内である場合、直前の分岐またはジャンプ命令の仮想アドレス

EPC レジスタとは異なり、ErrorEPC レジスタには、対応する分岐遅延スロットを示すものはありません。

PIC32 ファミリの一部のデバイスは MIPS16e® ASE を実装しているため、MFC0 命令による ErrorEPC レジスタの読み出しは、デスティネーション GPR に以下の値を返します。

$GPR[rt] = ErrorExceptionPC_{31..1} \,||\, ISAMode_0$

すなわち、エラー例外 PC の上位 31 ビットに ISA<1:0> ビット (Config3<15:14>) の下位ビットを連結した内容が GPR に書き込まれます。

同様に、MTC0 命令による ErrorEPC レジスタへの書き込みは、GPR から読み出した値を、以下のようにエラー例外 PC と ISA<1:0> ビット (Config3<15:14>) に分配します。

$ErrprExceptionPC = GPR[rt]_{31..1} \,||\, 0$
$ISAMode = 2\#0 \,||\, GPR[rt]_0$

すなわち、GPR の上位 31 ビットをエラー例外 PC の上位 31 ビットに書き込み、エラー例外 PC の下位ビットはクリアします。また、ISA<1:0> ビット (Config3<15:14>) の上位ビットをクリアし、下位ビットには GPR の下位ビットを読み込みます。

レジスタ 2-24: ErrorEPC: エラー例外プログラム カウンタ レジスタ ; CP0 レジスタ 30、Select 0

ビット レンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	ErrorEPC<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	ErrorEPC<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	ErrorEPC<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	ErrorEPC<7:0>							

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　　x = ビットは未知

bit 31-0 **ErrorEPC<31:0>:** エラー例外プログラム カウンタ ビット

2.12.25 DeSAVE レジスタ (CP0 レジスタ 31、Select 0)

DeSAVE レジスタは読み書き可能レジスタです。このレジスタは単純なメモリアドレスとして機能します。デバッグ例外ハンドラは、このレジスタを使って 1 つの GPR を保存し、その GPR を使って、残りのコンテクストを事前に決められたメモリ領域 (EJTAG プローブ等) に保存します。このレジスタを使うと、例外ハンドラと、その他のコードタイプ (コンテクストの保存用に有効なスタックを確保できそうにないコード) を安全にデバッグできます。

レジスタ 2-25: DeSAVE: デバッグ例外保存レジスタ ; CP0 レジスタ 31、Select 0

ビットレンジ	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DESAVE<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DESAVE<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DESAVE<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DESAVE<7:0>							

凡例 :		
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未実装ビット、「0」として読み出し
-n = POR 時の値	「1」= ビットはセット	「0」= ビットはクリア　　x = ビットは未知

bit 31-0 **DESAVE<31:0>:** デバッグ例外保存ビット
デバッグ例外コードが使うスクラッチパッド レジスタ

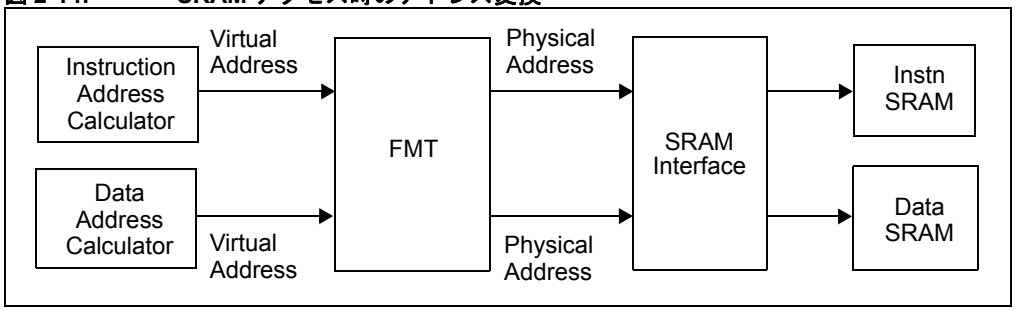
2.13 MIPS16e® の実行

コアが MIPS16e® モードで動作する場合、命令フェッチは、データの 16 ビットだけの戻りしか要求しません。しかし、効率を改善するために、アドレスがワードアラインされている場合、コアは命令データの 32 ビットをフェッチします。従って、MIPS16e® で連続した命令を実行する場合、フェッチは 1 つおきの命令にしか発生せず、結果として性能が向上し、システム消費電力が低減されます。

2.14 メモリモデル

ソフトウェアが使う仮想アドレスは、CPU バスに転送される前に、メモリ管理ユニット (MMU) によって物理アドレスに変換されます。PIC32 CPU は、この変換に固定された割り当てを使います。システム メモリモデルの詳細は**セクション 3.「メモリ構成」** (DS61115) を参照してください。

図 2-14: SRAM アクセス時のアドレス変換



2.14.1 キャッシュ適用性

CPU は命令フェッチ、ロード、ストアの仮想アドレスに基づいて、キャッシュにアクセスすべきかどうかを判断します。kseg0 または useg/kuseg 内のメモリアクセスはキャッシュ可能ですが、kseg1 内のアクセスはキャッシュできません。CPU は、メモリセグメントのキャッシュ適用性を判断するために、Config レジスタの CCA ビットを使います。対応する CCA が 011_2 であれば、メモリアクセスはキャッシュ可能です。キャッシュ動作の詳細は**セクション 4.「プリフェッチ キャッシュ モジュール」** (DS61119) を参照してください。

2.14.1.1 リトルエンディアンのバイトオーダ

バイト単位でメモリをアドレッシングする CPU では、複数バイトのデータアイテムを処理するために、バイトの並び順に関する規則が存在します。ビッグエンディアンのバイトオーダでは、複数バイトデータの MSB が最も低いアドレスに格納されます。リトルエンディアンのバイトオーダでは、複数バイトデータの LSB が最も低いアドレスに格納されます。PIC32 CPU はリトルエンディアンのバイトオーダをサポートします。

図 2-15: ビッグエンディアンのバイトオーダ

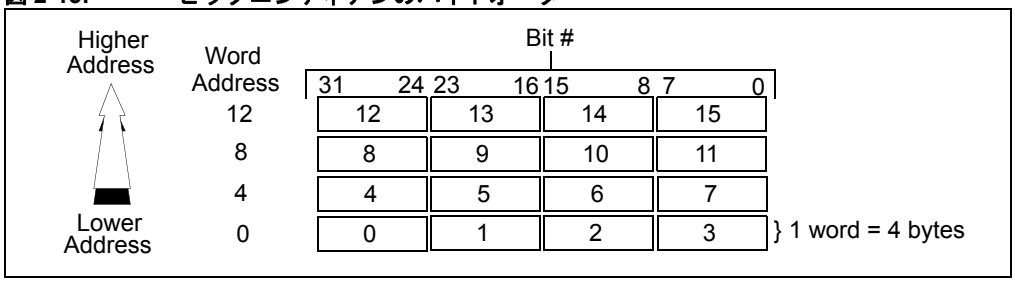
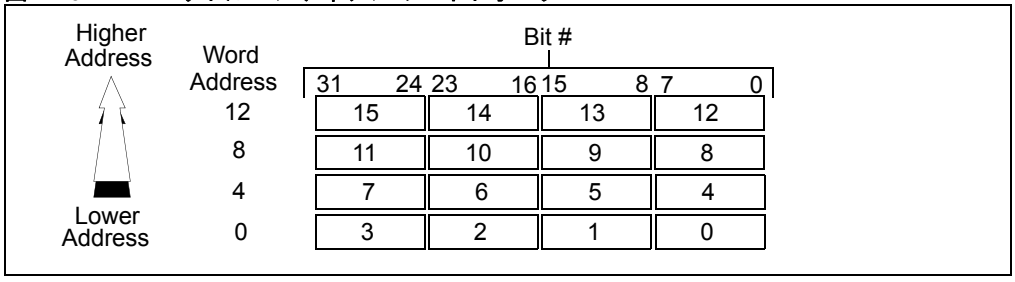


図 2-16: リトルエンディアンのバイトオーダ



2.15 CPU 命令の機能による分類

CPU 命令は、以下の機能グループに分類されます。

- ロードとストア
- 計算
- ジャンプと分岐
- その他
- コプロセッサ

各命令は 32 ビット長です。

2.15.1 CPU ロードとストア命令

ロード / ストア アーキテクチャを採用した MIPS® プロセッサは、全ての演算をプロセッサ レジスタが保持するオペランドに対して実行し、メインメモリにはロードおよびストア命令のみを介してアクセスします。

2.15.1.1 ロードおよびストアのタイプ

ロードおよびストア命令には以下に示す種類があり、それぞれ異なる目的に対して設計されています。

- 可変長フィールドの転送 (例 : LB、SW)
- 符号付きまたは符号なし整数として転送されたデータの変換 (例 : LHU)
- アライメントされていないフィールドへのアクセス (例 : LWR、SWL)
- アトミックなメモリ更新 (例 : LL/SC 等の読み出し - 変更 - 書き込み命令)

2.15.1.2 CPU ロードおよびストア命令の種類

CPU のロードおよびストア命令は以下のデータサイズ (AccessLength フィールドで定義) を転送します。

- バイト
- ハーフワード
- ワード

各サイズの符号付きと符号なし整数は、符号拡張またはゼロ拡張したデータをレジスタに格納するロード命令によってサポートされます。

アライメントされていないワードとダブルワードは、特定の 2 つの命令をペアで使う事により、2 命令だけでロードまたはストアできます。ロードする場合、LWL 命令と LWR 命令をペアで使います。これらのロード命令は、それぞれ左側と右側バイト (レジスタの左側と右側) をアライメントされたワードから読み出し、デスティネーション レジスタのバイトに正しく適合するようにそれらのバイトを結合します。

2.15.1.3 アトミックな更新に使うロードおよびストア命令

LL (Load Linked) 命令と SC (Store Conditional) 命令のペアを使うと、ワードまたはダブルワードでキャッシュされたメモリアドレスをアトミックに読み出し - 変更 - 書き込みできます。これらの命令は、Test-and-Set、ビットレベル ロック、セマフォ、シーケンサ / イベントカウントを含む各種同期化プリミティブのどれかを提供するために、厳密にコーディングされたシーケンス内で使います。

2.15.1.4 コプロセッサのロードとストア

有効化されていない特定のコプロセッサに対してロードおよびストアを実行する事はできません。そのようなロードまたはストアが試行された場合、コプロセッサ使用不可例外が発生します。コプロセッサの有効化は、システム制御コプロセッサ (CP0) が提供する特権動作です。

2.15.2 計算命令

2 の補数演算は、2 の補数で表現された整数に対して実行されます。これらは、以下の演算の符号付きバージョンです。

- 加算
- 減算
- 乗算
- 除算

「符号なし」の加算と減算演算は、実質的にオーバーフロー検出なしのモジュロ演算です。

乗算と除算およびシフトの完全補完と論理演算にも「符号なし」バージョンが存在します。論理演算はレジスタの幅を選びません。

MIPS32® は、32 ビット整数と 32 ビット算術演算を提供します。

2.15.2.1 シフト命令

ISA では、下記の 2 種類のシフト命令が定義されています。

- 命令ワード内の 5 ビットフィールドで固定シフト量を指定する (例 : SLL、SRL)
- 汎用レジスタの下位ビットでシフト量を指定する (例 : SRAV、SRLV)

2.15.2.2 乗算および除算命令

乗算命令は 32 ビット値に 32 ビット値を乗算し、64 ビットまたは 32 ビットの結果を生成します。除算命令は、64 ビット値を 32 ビット値で除算し、32 ビットの結果を生成します。1 つの例外を除き、これらの演算は HI と LO 特殊レジスタに結果を格納します。MUL 命令は、結果の下位半分を直接 GPR に格納します。

- 乗算は、入力オペランドの 2 倍のビット幅を持つ積を生成し、その下位半分を LO レジスタ、上位半分を HI レジスタに格納します。
- 積和および積差演算は、入力オペランドの 2 倍のビット幅を持つ積を生成し、これを HI と LO レジスタを連結した値に対して加算または減算します。そして、その結果の下位半分を LO レジスタに、上位半分を HI レジスタに書き戻します。
- 除算は、商を LO レジスタに、剰余を HI レジスタに格納します。

これらの結果には、HI/LO レジスタと汎用レジスタの間でデータを転送する命令を使ってアクセスします。

2.15.3 ジャンプおよび分岐命令

2.15.3.1 ISA によって定義されているジャンプおよび分岐命令のタイプ

アーキテクチャでは、以下のジャンプおよび分岐命令が定義されています。

- PC 相対条件分岐
- PC 領域無条件ジャンプ
- 絶対 (レジスタ) 無条件ジャンプ
- リターンリンク アドレスを汎用レジスタに記録する一連のプロシージャ コール

2.15.3.2 分岐遅延と分岐遅延スロット

全ての分岐は、構造的に 1 命令分の遅延を生じます。分岐直後の命令は、「分岐遅延スロット内にある」と表現します。分岐またはジャンプ命令が「分岐遅延スロット内にある」場合、それらの命令の動作は未定義です。

例外または割り込みが分岐遅延スロット内の命令の完了を妨げた場合、規則により、分岐命令フローは再実行されます。これを可能とするために、分岐は再実行可能である事が必要です。また、プロシージャ コールは、リターンリンクを格納しているレジスタ (通常は GPR 31) を使って分岐先アドレスを決定する事はできません。

2.15.3.3 分岐命令と BLANCH LIKELY 命令

条件分岐には以下の 2 種類が存在します。これらは、分岐が採択されず分岐の実行が不成立に終わった時に、遅延スロット内の命令を処理する方法において以下のように異なります。

- 分岐命令 : 遅延スロット内の命令を実行する
- Branch likely 命令 : 分岐が採択されなかった場合に遅延スロット内の命令を実行しない (これを「遅延スロット内の命令を無効に (nullify) する」と言う)

  Branch Likely 命令は本仕様書に含まれていますが、MIPS® アーキテクチャの将来のリビジョンでは廃止される予定であるため、ソフトウェアでは Branch Likely 命令を使わない事を強く推奨します。

2.15.4 その他の命令

2.15.4.1 同期化命令 (SYNC と SYNCI)

通常の動作では、実行中のプロセッサの外部から見た (例えば、マルチプロセッサ システムの場合) メモリに対するロードとストアのアクセス順序は、アーキテクチャによって定義されていません。

SYNC 命令を使うと、実行中の命令ストリーム内に、一部のロードおよびストア命令の相対的順番を特定可能にするためのポイントを作成できます。SYNC より前で実行されたロードとストアが完了するまで、SYNC より後のロードとストアは開始できません。

SYNCI 命令は、プロセッサ キャッシュを前回の書き込みまたは命令ストリームに対するその他の変更に同期させます。

2.15.4.2 例外命令

例外命令は、制御をカーネル内のソフトウェア例外ハンドラに渡します。例外には、条件付きと無条件の 2 種類があります。条件付き例外は比較結果に基づいて `trap` 命令で生成されます。無条件例外は、`syscall` と `break` 命令で生成されます。

2.15.4.3 条件付き移動命令

MIPS32® には、1 つの CPU 汎用レジスタの内容を、別のレジスタに、第 3 の汎用レジスタの値に基づいて条件付きで移動する命令があります。

2.15.4.4 NOP 命令

`NOP` 命令は、実質的に全てゼロとしてエンコードされます。MIPS® プロセッサは、このエンコードを「何もしない」特殊なケースとみなし、命令の実行を最適化します。加えて、`SSNOP` 命令は、スーパースカラ アーキテクチャを採用したプロセッサを含む全てのプロセッサ上で、「何もしない」サイクルを挿入します。

2.15.5 コプロセッサ命令

2.15.5.1 コプロセッサの役割

コプロセッサは、CPU とは別のレジスタファイルを使う代替実行ユニットです。抽象的に言うと、MIPS® アーキテクチャは最大 4 個のコプロセッサ ユニット (番号 0 ～ 3) を提供します。これらのコプロセッサの番号は、ISA の各レベルで定義されます。コプロセッサ 0 は常にシステム制御用に使い、コプロセッサ 1 と 3 は浮動小数点ユニット用に使います。コプロセッサ 2 は、実装固有の使用法に備えて予約済みです。

1 つのコプロセッサは以下の 2 種類のレジスタセットを備えます。

- コプロセッサ汎用レジスタ
- コプロセッサ制御レジスタ

各レジスタセットは最大 32 個のレジスタを収めます。コプロセッサ計算命令は、どちらのセット内のレジスタも使えます。

2.15.5.2 システム制御コプロセッサ 0 (CP0)

全ての MIPS® プロセッサ用システム コントローラは、コプロセッサ 0 (CP0: システム制御コプロセッサ) として実装されます。このコプロセッサはプロセッサ制御、メモリ管理、例外ハンドラ機能を提供します。

2.15.5.3 コプロセッサ ロードおよびストア命令

CP0 向けには明示的なロードおよびストア命令は定義されておらず、CP0 レジスタの書き込みと読み出しには、コプロセッサへの移動命令とコプロセッサからの移動命令を使う必要があります。その他のコプロセッサ向けのロードとストアは、**2.15.1.4「コプロセッサのロードとストア」**に概要を記載しています。

2.16 CPU の初期化

ソフトウェアは、リセットイベントの後に、デバイスの以下の部分を初期化する必要があります。

2.16.1 汎用レジスタ

電源投入時の CPU レジスタファイルの状態は不確定です。ただし r0 だけは常に「0」です。ハードウェアの適正動作という観点からは、その他のレジスタファイルを初期化する必要はありません。しかし、ソフトウェア環境によっては、以下を含む一部のレジスタの初期化が必要です。例えば、以下のレジスタです。

- sp - スタックポインタ
- gp - グローバル ポインタ
- fp - フレームポインタ

2.16.2 コプロセッサ 0 のステート

ブートコードを終了する前に、各種の CP0 ステートを初期化する必要があります。ERL = 1 または EXL = 1 によってブロックされる各種の例外が存在しますが、これらはリセットによってクリアされません。これらをクリアする事により、ブートコード終了時の誤った例外の発生を防げます。

表 2-12: CPU の初期化

CP0 レジスタ	操作
Cause	WP (ウォッチ保留) と SW0/1 (ソフトウェア割り込み) をクリアする必要があります。
Config	一般的に、K0、KU、K23 フィールドを必要なキャッシュ コヒーレンシ アルゴリズム (CCA) 値に設定した後に、対応するメモリ領域にアクセスする必要があります。
Count(1)	タイマ割り込みを使う場合、特定の値に設定する必要があります。
Compare(1)	タイマ割り込みを使う場合、特定の値に設定する必要があります。Compare レジスタに書き込むと、保留中のタイマ割り込みは全てクリアされます (従って、予期せぬ割り込みを防ぐために、Compare レジスタよりも先に Count レジスタを設定する必要があります)。
Status	デバイスのステートを必要な状態に設定する必要があります。
その他の CP0 ステート	その他のレジスタは、読み出される前に書き込んでおく必要があります。レジスタによっては、明示的に書き込む事はできず、命令の実行または例外の発生によって副次的に更新されるものもあります。そのようなレジスタを読み出した場合、初期化されていないビットを無視する必要があります。

Note 1: Count レジスタの値が Compare レジスタの値に一致した時に、タイマ割り込み信号が生成されます。必要に応じて、割り込みコントローラ内のマスクビットを使う事により、この割り込み信号の CPU への伝達を無効にできます。

2.16.3 バスマトリクス

バスマトリクス (BMX) は、ユーザモードへ切り換える前、または DRM から実行する前に、初期化する必要があります。バスマトリクスに書き込む値は、実行するアプリケーションのメモリ配置に基づきます。

2.17 リセットの影響

2.17.1 マスタクリア リセット

ハードウェア リセットでは、PIC32 コアは完全には初期化されません。プロセッサ ステートの最小限のサブセットだけがクリアされます。マッピングもキャッシュもされないコード空間内で動作させる場合のコアの起動方法としては、これで十分です。その他のプロセッサ ステートは、全て起動後にソフトウェアで初期化できます。パワーアップ リセットはデバイスを既知の状態にします。ソフトリセットは、MCLR ピンをアサートする事により生成できます。このようにリセットを区別するのは、他の MIPS® プロセッサとの互換性を維持するためです。実際は、どちらのリセットも同様に処理します。

2.17.1.1 コプロセッサ 0 のステート

ハードウェア初期化の大部分はコプロセッサ 0 で発生します。表 2-13 に、その内容を示します。

表 2-13: リセットによってクリアまたはセットされるビット

レジスタ名	ビット名	クリア / セット	値	リセットのタイプ
Status	BEV	クリア	1	リセットまたはソフトリセット
	TS	クリア	0	リセットまたはソフトリセット
	SR	セット	1	リセットまたはソフトリセット
	NMI	クリア	0	リセットまたはソフトリセット
	ERL	セット	1	リセットまたはソフトリセット
	RP	クリア	0	リセットまたはソフトリセット
全てのコンフィグレーション レジスタ : Config Config1 Config2 Config3	静的入力に関連する コンフィグレーション フィールド	セット	入力値	リセットまたはソフトリセット
Config	K0	セット	010 (キャッシュせず)	リセットまたはソフトリセット
	KU	セット	010 (キャッシュせず)	リセットまたはソフトリセット
	K23	セット	010 (キャッシュせず)	リセットまたはソフトリセット
Debug	DM	クリア	0	リセットまたはソフトリセット(1)
	LSNM	クリア	0	リセットまたはソフトリセット
	IBUSEP	クリア	0	リセットまたはソフトリセット
	IEXI	クリア	0	リセットまたはソフトリセット
	SSt	クリア	0	リセットまたはソフトリセット

Note 1: デバッグモードへの起動に EJTAGBOOT オプションを使わない場合

2.17.1.2 バス ステートマシン

リセットまたはソフトリセット例外が発生すると、保留中のバス トランザクションは全て中止され、SRAM インターフェイス ユニット内のステートマシンはリセットされます。

2.17.2 フェッチアドレス

EJTAGBOOTオプションを使わない場合、リセットとソフトリセット時にフェッチはVA 0xBFC00000 (PA 0x1FC00000) にアドレッシングされます。このアドレスは、マッピングもキャッシュもされない kseg1 内に位置します。

2.17.3 ウォッチドッグ タイマリセット

ウォッチドッグ タイマ (WDT) イベント後の CPU レジスタの状態は、WDT イベント前の CPU の動作モードによって決まります。

デバイスがスリープ中ではなかった場合、WDT イベントによってレジスタはリセット値に設定されます。

2.18 関連アプリケーション ノート

本書に関連するアプリケーション ノートの一覧を以下に記載します。これらのアプリケーション ノートは特に PIC32 ファミリ向けとして書かれたものではありません。ただし概念は共通しており、変更が必要であったり制限事項が存在するものの利用が可能です。PIC32 M4K® コア搭載デバイス用 CPU に関連する最新のアプリケーション ノートは以下の通りです。

タイトル	アプリケーション ノート番号

現在、関連するアプリケーション ノートはありません。

Note: PIC32 ファミリ向けのその他のアプリケーション ノートとサンプルコードは、Microchip 社のウェブサイト (www.microchip.com) をご覧ください。

2.19 改訂履歴

リビジョン A (2007 年 10 月)

本書の初版です。

リビジョン B (2008 年 4 月)

本書のステータスを Preliminary に変更しました。セクション 2.1「主な特長」と図 2-1 を変更しました。U-0 を r-x に変更しました。

リビジョン C (2008 年 5 月)

図 2-1 を変更しました。セクション 2.2.3「コアタイマ」を追加しました。予約済みビットを「Maintain as」から「Write」に変更しました。

リビジョン D (2011 年 8 月)

このリビジョンでの変更内容は以下の通りです。

PIC32 ファミリに M14K™ マイクロプロセッサコアを採用したデバイスが追加された事を反映して、本書の下記の内容を更新しました。

- セクション :
  - **2.1.1「主な特長」**を更新しました。
  - **2.1.2「MIPS® 関連の文書」**を更新しました。
  - **2.2.2「プログラミング モデルの概要」**を更新しました。
  - **2.3.1.1「I 段 - 命令フェッチ」**を更新しました。
  - **2.10「割り込みおよび例外機構」**を更新しました。
  - **2.14「microMIPS™ 実行 (M14K™ 用デバイスのみ)」**を追加しました。
  - **2.15「MCU™ ASE の実行 (M14K™ 用デバイスのみ)」**を追加しました。
- 図 :
  - 図 2-1: PIC32 ブロック図を更新しました。
  - 図 2-2: M4K® および M14K™ マイクロプロセッサ コアのブロック図を更新しました。
- 表 :
  - 表 2-6: microMIPS™ 16 ビット命令レジスタの使用方法 (M14K™ 用デバイスのみ) を追加しました。
  - 表 2-8: CP0 レジスタを更新しました。
  - 表 2-14: パフォーマンス カウンタでカウント可能なイベントを追加しました。
  - 表 2-15: イベントの説明を追加しました。
  - 表 2-17: リセットによるビットのクリア / セットを更新しました。
- レジスタ :
  - レジスタ 2-1、レジスタ 2-10、レジスタ 2-11、レジスタ 2-13、レジスタ 2-22、レジスタ 2-24、レジスタ 2-25、レジスタ 2-26、レジスタ 2-27、レジスタ 2-28、レジスタ 2-30、レジスタ 2-31 を追加しました。
  - M4K® または M14K™ マイクロプロセッサ コアのどちらかだけに実装されているビットを反映するために、既存レジスタの関連する部分だけを更新しました。
- 表現および体裁の変更等、本書全体の細部を修正しました。

リビジョン E (2012 年 9 月)

このリビジョンでの変更内容は以下の通りです。

- 文書のタイトルを変更しました。
- 文書全体を通して M14K™ マイクロプロセッサ コアに対する言及を全て削除しました。このコアに関する内容は、ファミリ リファレンス マニュアルの別のセクションで解説する予定です。
- 表現および体裁の変更等、本書全体の細部を修正しました。

Microchip 社製デバイスのコード保護機能に関して以下の点にご注意ください。

- Microchip 社製品は、該当する Microchip 社データシートに記載の仕様を満たしています。
- Microchip 社では、通常の条件ならびに仕様に従って使用した場合、Microchip 社製品のセキュリティ レベルは、現在市場に流通している同種製品の中でも最も高度であると考えています。
- しかし、コード保護機能を解除するための不正かつ違法な方法が存在する事もまた事実です。弊社の理解では、こうした手法は Microchip 社データシートにある動作仕様書以外の方法で Microchip 社製品を使用する事になります。このような行為は知的所有権の侵害に該当する可能性が非常に高いと言えます。
- Microchip 社は、コードの保全性に懸念を抱いているお客様と連携し、対応策に取り組んでいきます。
- Microchip 社を含む全ての半導体メーカーで、自社のコードのセキュリティを完全に保証できる企業はありません。コード保護機能とは、Microchip 社が製品を「解読不能」として保証するものではありません。

コード保護機能は常に進歩しています。Microchip 社では、常に製品のコード保護機能の改善に取り組んでいます。Microchip 社のコード保護機能の侵害は、デジタル ミレニアム著作権法に違反します。そのような行為によってソフトウェアまたはその他の著作物に不正なアクセスを受けた場合、デジタル ミレニアム著作権法の定めるところにより損害賠償訴訟を起こす権利があります。

商標

ISBN: 978-1-62077-468-7

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Microchip 社では、Chandler および Tempe (アリゾナ州)、Gresham (オレゴン州) の本部、設計部およびウェハー製造工場そしてカリフォルニア州とインドのデザインセンターが ISO/TS-16949: 2009 認証を取得しています。Microchip 社の品質システム プロセスおよび手順は、PIC® MCU および dsPIC® DSC、KEELOQ® コード ホッピング デバイス、シリアル EEPROM、マイクロペリフェラル、不揮発性メモリ、アナログ製品に採用されています。さらに、開発システムの設計と製造に関する Microchip 社の品質システムは ISO 9001:2000 認証を取得しています。

各国の営業所とサービス

北米

本社
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
技術サポート :
http://www.microchip.com/support
URL:
www.microchip.com

アトランタ
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

オースティン、TX
Tel: 512-257-3370

ボストン
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

シカゴ
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

クリーブランド
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

ダラス
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

デトロイト
Novi, MI
Tel: 248-848-4000

ヒューストン、TX
Tel: 281-894-5983

インディアナポリス
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

ロサンゼルス
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

ニューヨーク、NY
Tel: 631-435-6000

サンノゼ、CA
Tel: 408-735-9110

カナダ - トロント
Tel: 905-673-0699
Fax: 905-673-6509

アジア / 太平洋

アジア太平洋支社
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

オーストラリア - シドニー
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

中国 - 北京
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

中国 - 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 - 重慶
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 - 杭州
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

中国 - 香港 SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

中国 - 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 - 青島
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 - 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 - 瀋陽
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 - 深圳
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

中国 - 武漢
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 - 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

中国 - 厦門
Tel: 86-592-2388138
Fax: 86-592-2388130

中国 - 珠海
Tel: 86-756-3210040
Fax: 86-756-3210049

アジア / 太平洋

インド - バンガロール
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

インド - ニューデリー
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

インド - プネ
Tel: 91-20-3019-1500

日本 - 大阪
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

日本 - 東京
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

韓国 - 大邱
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韓国 - ソウル
Tel: 82-2-554-7200
Fax: 82-2-558-5932 または
82-2-558-5934

マレーシア - クアラルンプール
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

マレーシア - ペナン
Tel: 60-4-227-8870
Fax: 60-4-227-4068

フィリピン - マニラ
Tel: 63-2-634-9065
Fax: 63-2-634-9069

シンガポール
Tel: 65-6334-8870
Fax: 65-6334-8850

台湾 - 新竹
Tel: 886-3-5778-366
Fax: 886-3-5770-955

台湾 - 高雄
Tel: 886-7-213-7830

台湾 - 台北
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

タイ - バンコク
Tel: 66-2-694-1351
Fax: 66-2-694-1350

ヨーロッパ

オーストリア - ヴェルス
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

デンマーク - コペンハーゲン
Tel: 45-4450-2828
Fax: 45-4485-2829

フランス - パリ
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

ドイツ - デュッセルドルフ
Tel: 49-2129-3766400

ドイツ - ミュンヘン
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

ドイツ - プフォルツハイム
Tel: 49-7231-424750

イタリア - ミラノ
Tel: 39-0331-742611
Fax: 39-0331-466781

イタリア - ヴェニス
Tel: 39-049-7625286

オランダ - ドリューネン
Tel: 31-416-690399
Fax: 31-416-690340

ポーランド - ワルシャワ
Tel: 48-22-3325737

スペイン - マドリッド
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

スウェーデン - ストックホルム
Tel: 46-8-5090-4654

イギリス - ウォーキンガム
Tel: 44-118-921-5800
Fax: 44-118-921-5820

10/28/13