お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

RENESAS

ユーザーズ・マニュアル

保守／廃止

μPD98502

ネットワーク・コントローラ

資料番号　S15543JJ1V0UM00（第1版）
発行年月　January 2003 N　CP(K)

保守／廃止

〔メ　モ〕

保守／廃止

目次要約

CMOSデバイスの一般的注意事項

①静電気対策（MOS全般）

注意　MOSデバイス取り扱いの際は静電気防止を心がけてください。

MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には，NECが出荷梱包に使用している導電性のトレーやマガジン・ケース，または導電性の緩衝材，金属ケースなどを利用し，組み立て工程にはアースを施してください。プラスチック板上に放置したり，端子を触ったりしないでください。

また，MOSデバイスを実装したボードについても同様の扱いをしてください。

②未使用入力の処理（CMOS特有）

注意　CMOSデバイスの入力レベルは固定してください。

バイポーラやNMOSのデバイスと異なり，CMOSデバイスの入力に何も接続しない状態で動作させると，ノイズなどに起因する中間レベル入力が生じ，内部で貫通電流が流れて誤動作を引き起こす恐れがあります。プルアップかプルダウンによって入力レベルを固定してください。また，未使用端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してV_{DD}またはGNDに接続することが有効です。

資料中に「未使用端子の処理」について記載のある製品については，その内容を守ってください。

③初期化以前の状態（MOS全般）

注意　電源投入時，MOSデバイスの初期状態は不定です。

分子レベルのイオン注入量等で特性が決定するため，初期状態は製造工程の管理外です。電源投入時の端子の出力状態や入出力設定，レジスタ内容などは保証しておりません。ただし，リセット動作やモード設定で定義している項目については，これらの動作ののちに保証の対象となります。

リセット機能を持つデバイスの電源投入後は，まずリセット動作を実行してください。

保守／廃止

VRはNECエレクトロニクス株式会社の登録商標です。

VR4100, VR4102, VR4111, VR4120A, VR4300, VR4305, VR4310, VR4400, VR5000, VR10000, VRシリーズ，VR4000シリーズ，VR4100シリーズ，EEPROMは，NECエレクトロニクス株式会社の商標です。

Micro Wireは，National Semiconductor社の商標です。

iAPXは，米国Intel Corp.の商標です。

DEC VAXは，米国Digital Equipment Corp.の商標です。

UNIXは，X/Openカンパニーリミテッドがライセンスしている米国ならびに他の国における登録商標です。

イーサネットは，米国Xerox社の商標です。

MIPSは，米国MIPS Technologies, Inc.の米国における登録商標です。

- 本資料に記載されている内容は2002年12月現在のもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。
- 当社は，本資料に記載された当社製品の使用に関連し発生した第三者の特許権，著作権その他の知的財産権の侵害等に関し，一切その責を負いません。当社は，本資料に基づき当社または第三者の特許権，著作権その他の知的財産権を何ら許諾するものではありません。
- 本資料に記載された回路，ソフトウエアおよびこれらに関する情報は，半導体製品の動作例，応用例を説明するものです。お客様の機器の設計において，回路，ソフトウエアおよびこれらに関する情報を使用する場合には，お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し，当社は，一切その責を負いません。
- 当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等安全設計を行ってください。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「特定水準」に分類しております。また，各品質水準は，以下に示す用途に製品が使われることを意図しておりますので，当社製品の品質水準をご確認ください。

  標準水準：コンピュータ，OA機器，通信機器，計測機器，AV機器，家電，工作機械，パーソナル機器，産業用ロボット

  特別水準：輸送機器（自動車，電車，船舶等），交通用信号機器，防災・防犯装置，各種安全装置，生命維持を目的として設計されていない医療機器

  特定水準：航空機器，航空宇宙機器，海底中継機器，原子力制御システム，生命維持のための医療機器，生命維持のための装置またはシステム等

当社製品のデータ・シート，データ・ブック等の資料で特に品質水準の表示がない場合は，標準水準製品であることを表します。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

（注）

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNECエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいう。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E 02.11

〔メ　モ〕

保守／廃止

はじめに

対 象 者　このマニュアルは，μPD98502の機能を理解し，これを用いたアプリケーション・システムを開発するエンジニアを対象としています。

目　　的　このマニュアルは，次の構成に示すμPD98502のハードウエア機能をユーザに理解していただくことを目的としています。

構　　成　このマニュアルは，次の内容で構成しています。

- 概　　説
- VR4120A™ CPU
- システム・コントローラ
- ATMセル・プロセッサ
- イーサネット™・コントローラ
- USBコントローラ
- PCIコントローラ
- UART
- タイマ
- Micro Wire™

読 み 方　このマニュアルの読者には，電気，論理回路，マイクロコンピュータに関する一般知識を必要とします。

一通りμPD98502の機能を理解しようとするとき
→目次に従ってお読みください。

μPD98502の電気的特性を知りたいとき
→別冊のデータ・シートを参照してください。

凡　　例　このマニュアルでは，次の記号を使用しています。

データ表記の重み：左が上位桁，右が下位桁
アクティブ・ロウの表記：XXXX_B（端子，信号名称のあとに_B）
注　　：本文中に付けた注の説明
注意　　：気をつけて読んでいただきたい内容
備考　　：本文の補足説明
数の表記：2進数...xxxxまたはxxxxB
10進数...xxxx
16進数...xxxxH

保守／廃止

関連資料　**このマニュアルは，以下のドキュメントと一緒にお使いください。**

関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

・μPD98502**データ・シート：**S15409J

保守／廃止

目　　次

保守／廃止

保守／廃止

保守／廃止

図の目次（1/6）

図の目次（2/6）

保守／廃止

図の目次（3/6）

保守／廃止

図の目次（4/6）

保守／廃止

図の目次（5/6）

保守／廃止

図の目次（6/6）

保守／廃止

表の目次（1/3）

保守／廃止

表の目次（2/3）

保守／廃止

表の目次（3/3）

保守／廃止

第1章　概　　説

μPD98502は，ADSLルータに最適なIPパケットとATMセル間のプロトコル変換を行う高性能コントローラです。このコントローラは，CPUコアとしてMIPS™ベースの高性能64ビットRISCプロセッサVR4120Aを内蔵しており，さらに，ATMセル・プロセッサ，イーサネット・コントローラ，USBコントローラ，PCIコントローラ，UTOPIA2インタフェース，SDRAMインタフェースを備えています。

1.1　特　　徴

- 高性能なMIPSベースの64ビットRISCプロセッサVR4120Aを搭載
- RTOSとネットワーク・ミドルウエア（M/W）をオン・チップで実行可能
- ブート・プログラム格納用PROM/フラッシュ・メモリ用インタフェース内蔵
- ATMセル・プロセッサ内に32ビットRISCコントローラを搭載
- RISCコントローラ上のソフトウエアによるSAR処理を実行
- CBR/VBR/UBRサービス・クラスに対応
- IEEE802.3, IEEE802.3uおよびIEEE802.3x準拠の10/100 Mbpsイーサネット・コントローラを2チャネル内蔵
- 3.3 V MIIインタフェースにより外部イーサネットPHYデバイスをダイレクトに接続が可能
- USB仕様1.1準拠のUSBフルスピード・ファンクション・コントローラ内蔵
- USB Communication Device Class Specification準拠の動作に対応
- 外部メモリとして64 Mビットと128 MビットのSDRAMをダイレクトに接続が可能
- PCI仕様rev. 2.2準拠の32ビット，33 MHz PCIバス・マスタを内蔵
- ATM Forum af-phy-0039準拠の8ビット，16.5/25/33 MHz UTOPIAレベル2インタフェースを内蔵
- IEEE1149.1準拠のバウンダリ・スキャン機能（JTAG）内蔵
- UARTおよびMicro Wireインタフェース内蔵
- 汎用タイマを2チャネル内蔵
- 先端CMOSテクノロジ採用
- 電源電圧：2.5 V（コア），3.3 V（I/O）
- 500ピンT-BGAパッケージ

1.2　オーダ情報

オーダ名称	パッケージ
μPD98502N7-H6	500ピン・テープBGA（H/Sp付き）（40×40）

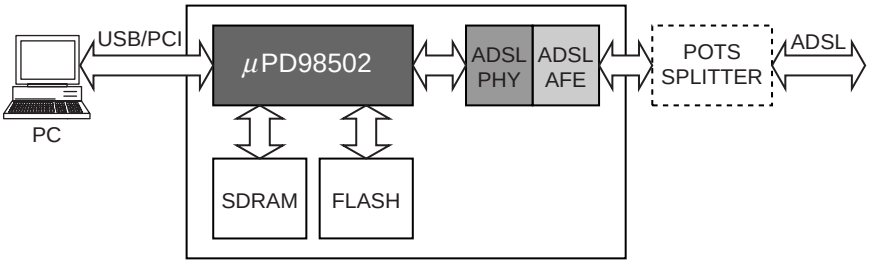
保守／廃止

1.3 システム構成

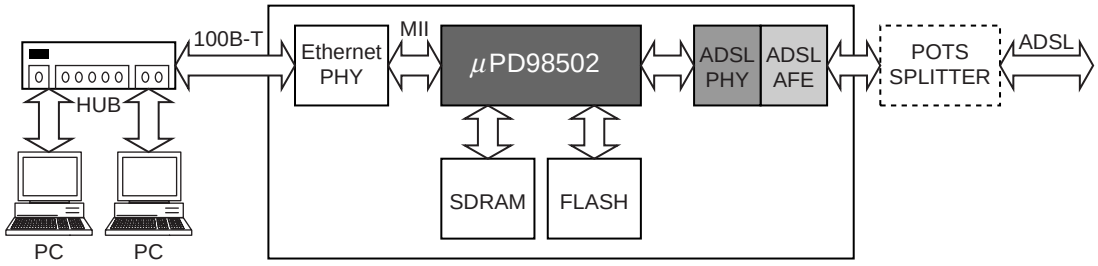
μPD98502は，ADSL/ATMインタフェースとUSB/イーサネット・インタフェース間のブリッジングとルーティングを1つのチップで行います。各ユーザ・インタフェースをそれぞれの機器に応じて使用したシステム構成例を以下に示します。USBとイーサネットの機能は互いに独立して動作します。

図1－1　μPD98502のシステム構成例

（a）ADSLモデム



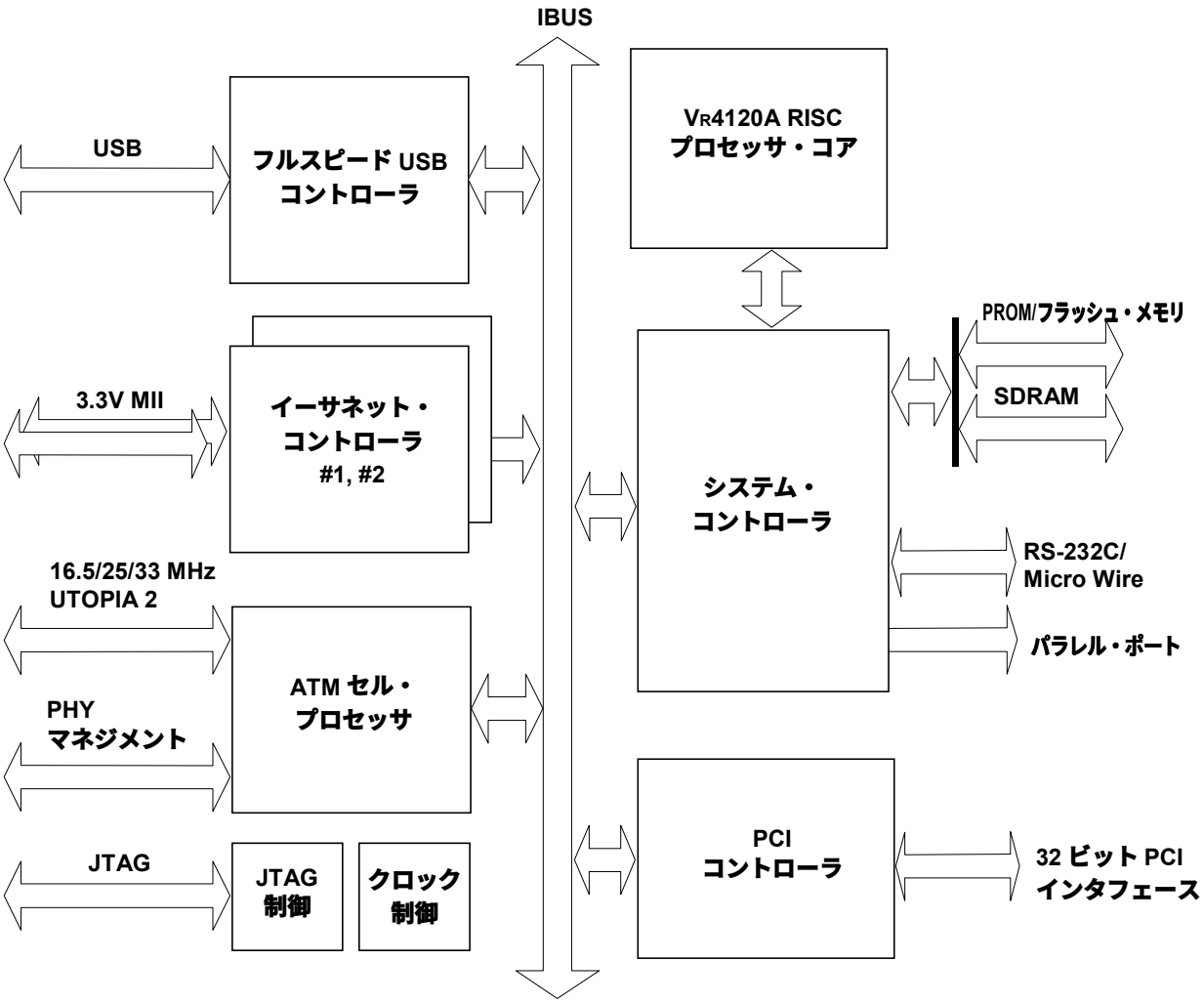
（b）ADSLルータ



保守／廃止

1. 4　ブロック図（概要）

図1－2　μPD98502のブロック図



保守／廃止

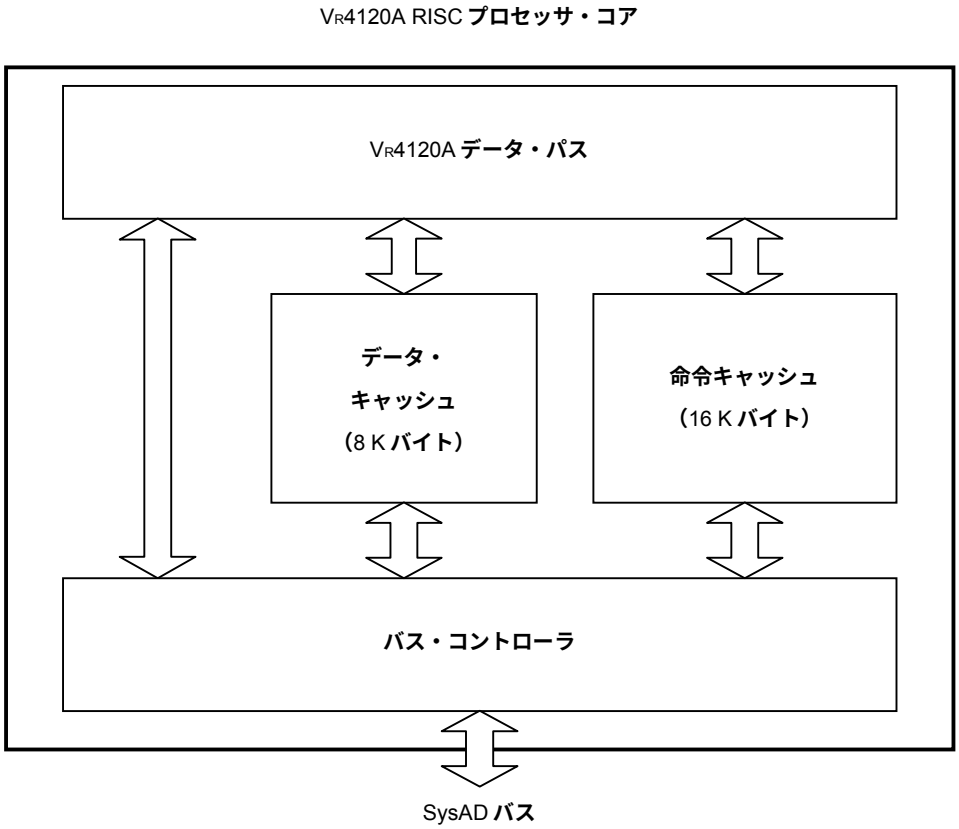
1. 5　ブロック図（詳細）

1. 5. 1　VR4120A RISCプロセッサ・コア

VR4120A RISCプロセッサ・コアの特徴は次のとおりです。

- MIPS/ I /II/III命令セットに対応（FPU, LL, LLD, SC, SCD命令は除く）
- 高速積和演算によりアプリケーションの高速処理を実現
- ラージ・サイズ・キャッシュ・メモリ（命令：16 Kバイト，データ：8 Kバイト）内蔵
- 高速変換緩衝機構（TLB）により最大1 Tバイトの仮想アドレスをサポート
- ビッグ・エンディアンとリトル・エンディアンの切り替え機能を実現

図1－3　VR4120A RISCプロセッサのブロック図

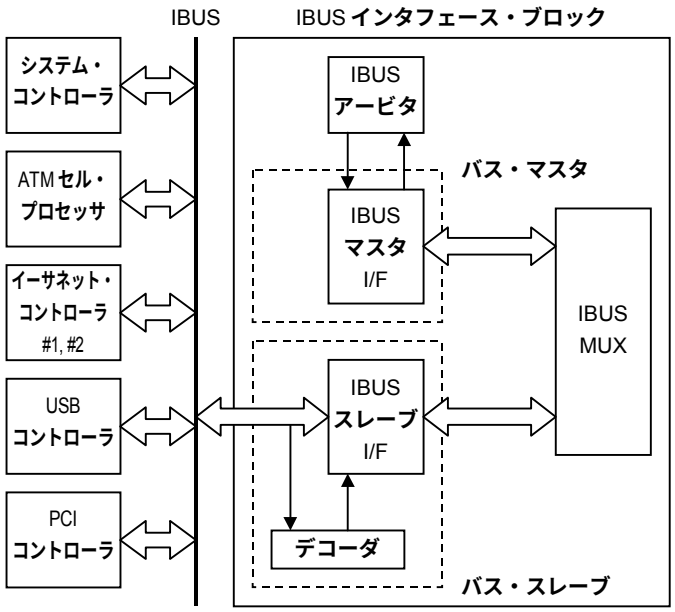


1. 5. 2　IBUS

IBUSは，32ビット，66 MHzの高速内蔵バスで，それぞれのコントロール・ブロックの相互接続を可能にします。IBUSは，以下のバス・プロトコルをサポートします。

- シングル・リード／ライト転送
- バースト・リード／ライト転送
- スレーブ・ロック
- リトライとディスコネクト
- バス・パーキング

図1－4　IBUSのブロック図



（マルチプレクス・バス・アレンジメントを使用）

保守／廃止

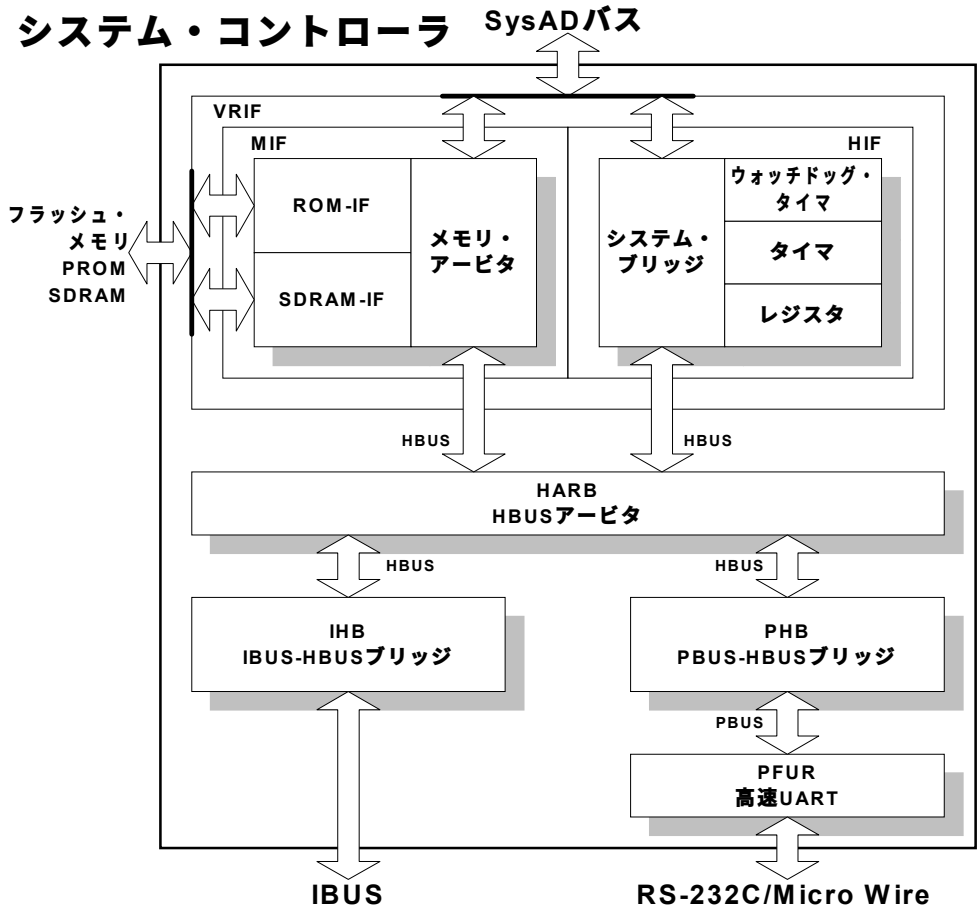
1. 5. 3　システム・コントローラ

システム・コントローラは，μPD98502の内部システム・コントローラです。システム・コントローラは，VR4120Aのシステム・バスSysAD, NECのオリジナル高速内蔵バスIBUSとSDRAM/PROM/フラッシュ・メモリ用のメモリ・バス間のブリッジング機能を提供します。

システム・コントローラの特徴は次のとおりです。

- SysADとメモリ間の4ワード・プリフェッチFIFOバッファを実装
- 各送信，受信からIBUSへの32ビット×64ワードFIFOバッファを実装
- 各送信，受信からHBUSへの32ビット×4ワードFIFOバッファを実装
- SysADバスおよびIBUS（内部バス）とメモリ間のブリッジング機能を提供
- SysADバス上のエンディアン変換機能をサポート
- SDRAM（最大32 Mバイト）とPROM/フラッシュ（最大8 Mバイト）メモリをダイレクトに接続が可能
- 66 MHzまたは100 MHzのいずれかのVR4120Aのバス・サイクルをサポート
- SDRAMのデータ・バスとPROM/フラッシュ・メモリのデータ・バスは兼用
- IBUS上で266 Mバイト／秒（32ビット@66 MHz）をサポート
- NMIとINTを発生
- NS16550準拠のユニバーサル非同期レシーバ／トランスミッタ（UART）をサポート
- 独立動作の2チャネル・タイマをサポート
- ウォッチドッグ・タイマをサポート
- Micro Wireインタフェースをサポート

図1－5　システム・コントローラのブロック図



保守／廃止

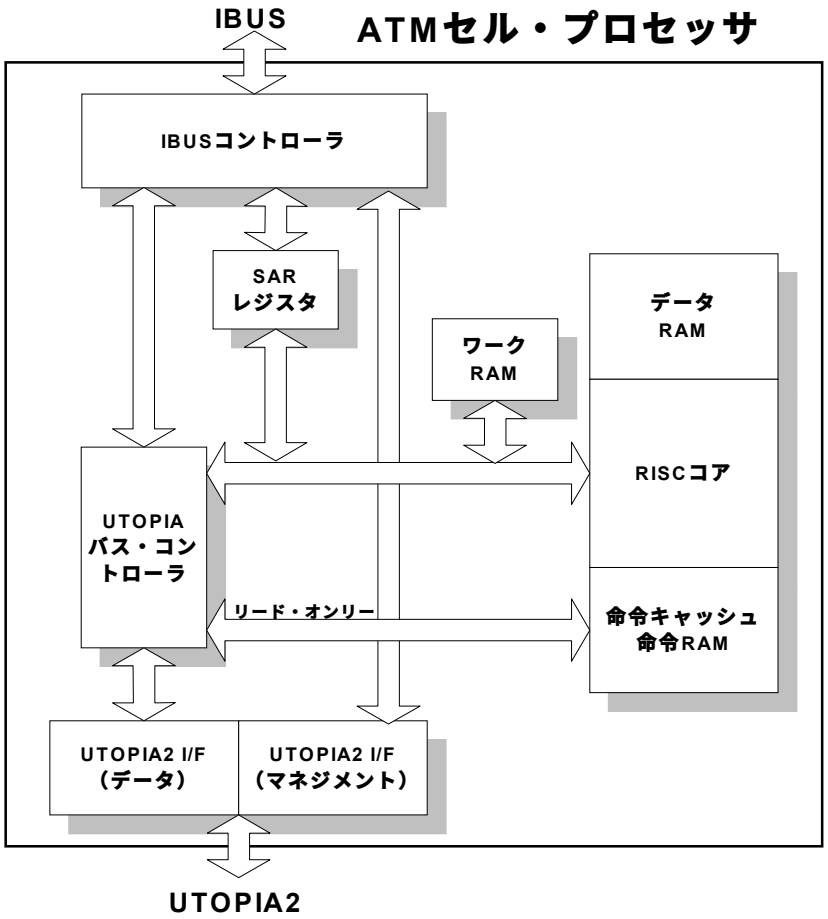
1. 5. 4　ATMセル・プロセッサ

NEC独自の32ビットRISCコントローラを用いて，ATMセル・プロセッサ・ユニットを実現します。ファームウエアによるATMセル処理を行うことで，より柔軟性が高まります。

ATMセル・プロセッサの特徴は次のとおりです。

- 32ビットRISCコントローラ（76 MIPS@66 MHz）によりソフトウエアSAR機能を実現
- 外部メモリから命令キャッシュへのファームウエアのダウンロード
- 最大64 VCをサポート
- UTOPIAレベル2（マネジメント・インタフェースを含む）をPHYレイヤ・インタフェースとしてサポート
- AAL2, AAL5, Rawセル（AAL0），F5 OAMセルの処理をサポート
- 3つのサービス・クラス（CBR, VBR, UBR）をサポート
- アップストリーム，ダウンストリームともに最大各25 Mbpsのセル・スピードをサポート
- VCごとに1セル／秒の精度でATMセル・シェーピングをサポート

図1－6　ATMセル・プロセッサのブロック図



保守／廃止

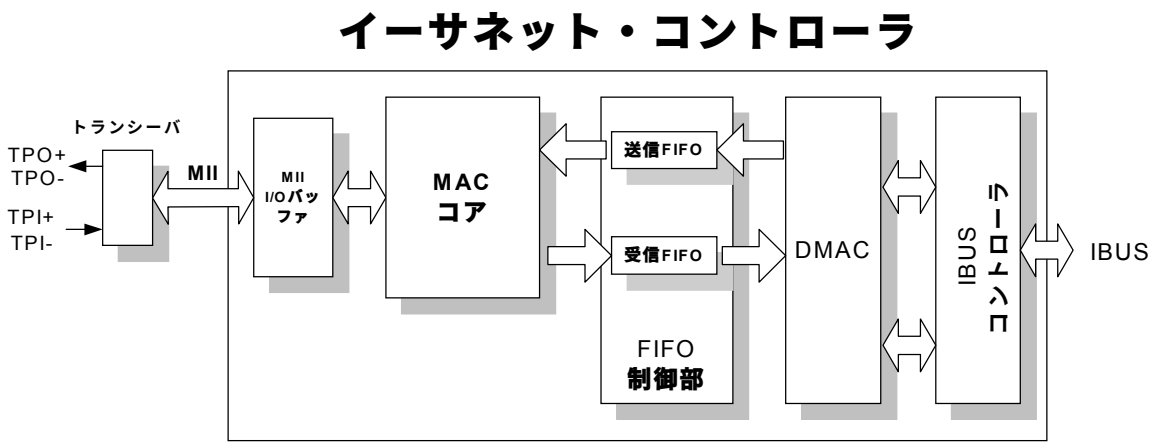
1. 5. 5　イーサネット・コントローラ

イーサネット・コントローラは，2チャネルの10 Mbps/100 Mbpsイーサネット MAC（Media Access Control）機能とMII（Media Independent Interface）機能をサポートします。

イーサネット・コントローラの特徴は，次のとおりです。

- IEEE802.3, IEEE802.3u準拠の10 M/100 Mイーサネット MAC機能をサポート
- IEEE802.3u準拠の3.3 V MIIをサポート
- 100 Mbpsと10 Mbpsの両方の全二重動作をサポート
- 各送信と受信に対して256バイトのFIFOバッファを実装
- ユニキャスト／マルチキャスト／ブロードキャストに対するアドレス・フィルタリング機能をサポート
- ネットワーク管理用のMIBカウンタを実装（MIB II，イーサMIB, IEEE802.3LMEをサポート）
- 各送信と受信に対する個々のDMA転送をサポートするDMAコントローラを実装

図1－7　イーサネット・コントローラのブロック図



保守／廃止

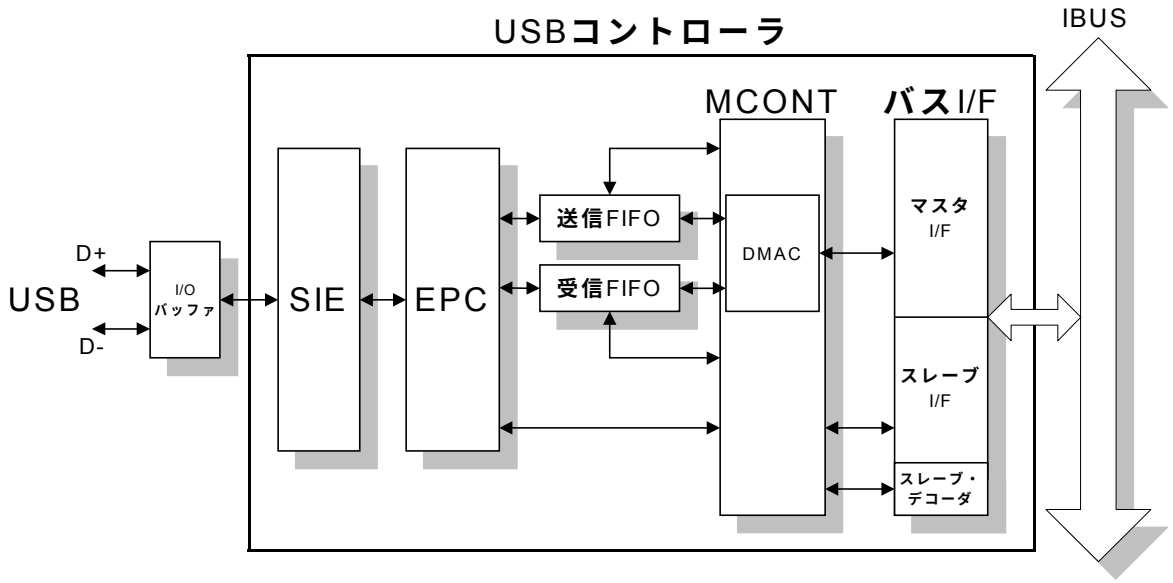
1. 5. 6　USBコントローラ

USBコントローラは，ユニバーサル・シリアル・バスに定義されたフルスピード・ファンクション・デバイス機能を提供します。

USBコントローラの特徴は，次のとおりです。

- ユニバーサル・シリアル・バス仕様rev.1.1に準拠
- VR4120A上で動作するソフトウエアによるデバイス・クラス機能をサポート
- 12 Mbpsフルスピードファンクション・デバイス（ハブ機能はサポートしません）
- サスペンド，リジューム，ウェークアップ・マネジメントの信号を発生可能
- リモート・ウェークアップをサポート
- 7種類のエンドポイント（制御，割り込みI/O，アイソクロナスI/O，バルクI/O）をサポート
- 送信に対する制御転送に使用する64バイトのFIFOバッファを実装
- 送信に対するアイソクロナス転送に使用する128バイトのFIFOバッファを実装
- 送信に対するバルク転送に使用する128バイトのFIFOバッファを実装
- 送信に対する割り込み転送に使用する64バイトのFIFOバッファを実装
- 受信に対する制御／アイソクロナス／バルク／割り込み転送に使用する128バイトの共用FIFOバッファを実装
- ローカルDMAC（DMAコントローラ）ブロックを実装
- USB専用I/Oバッファを介してUSBコネクタをダイレクトに接続が可能

図1－8　USBコントローラのブロック図



保守／廃止

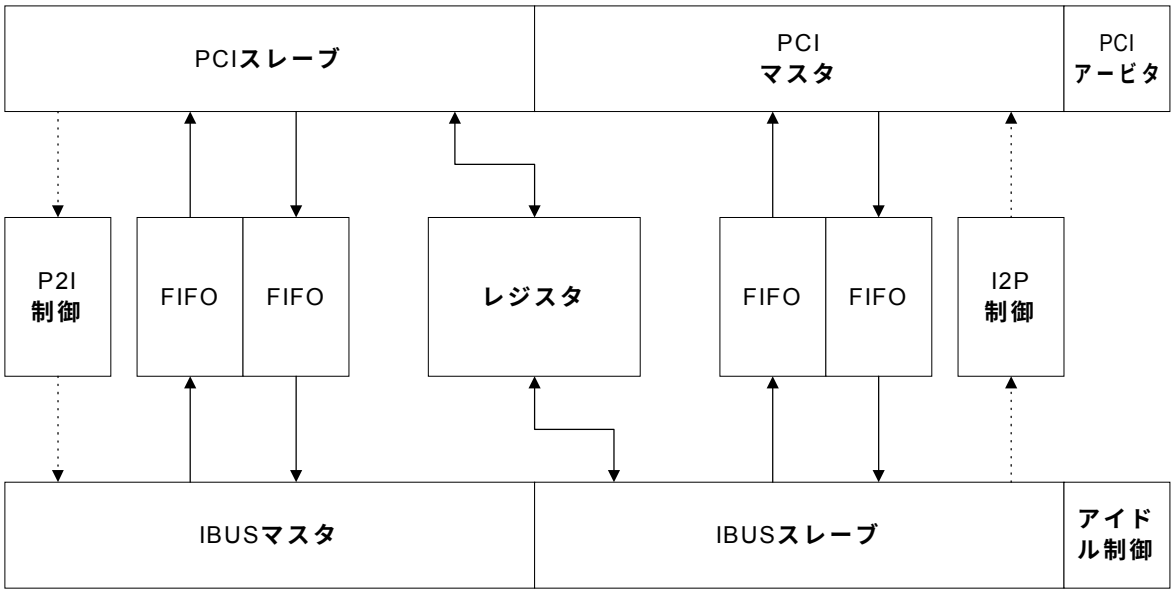
1. 5. 7　PCIコントローラ

PCIコントローラは，PCI SIGで定義されるPCIバス機能を提供します。このブロックは，IBUSとPCIのあいだのブリッジングをします。

PCIコントローラの特徴は，次のとおりです。

- 32ビットPCIインタフェース（最大33 MHz）
- 32ビットIBUSインタフェース（最大33 MHz）
- 33 MHzのPCI周波数
- PCIローカル・バス仕様rev. 2.2に準拠
- PCIバス・パワー・マネジメント・インタフェースrev.1.1に準拠
- 各方向に対し16ワードまでのバーストをサポート
- ホスト・モードで4台までの外部PCIマスタ・デバイスのアービトレーションをサポートするPCIバス・アービタを実装

図1－9　PCIバス・コントローラのブロック図



――――― データ・フロー

・・・・・・・・・・・・・制御

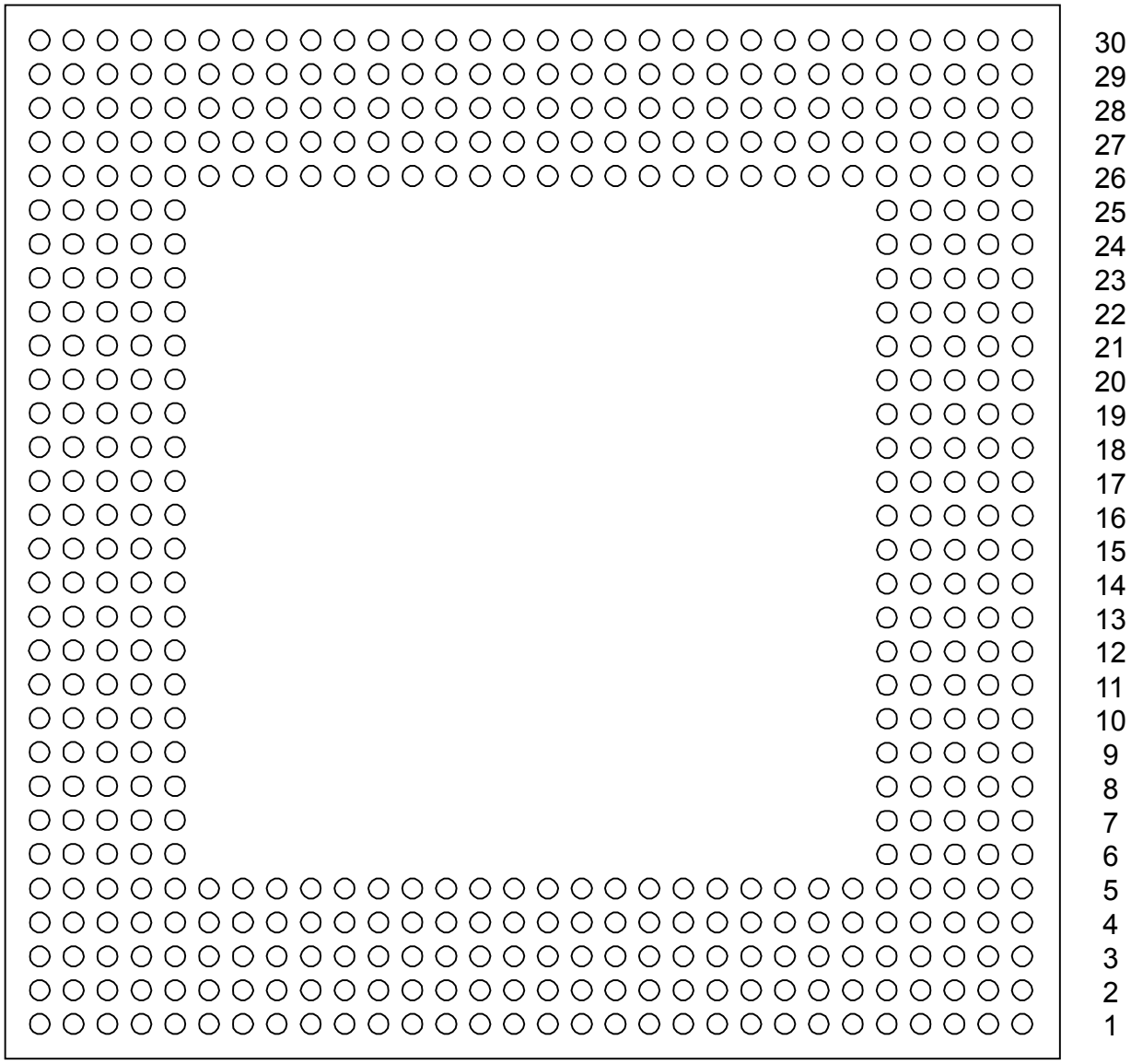
保守／廃止

1. 6　端子配置（Bottom View）

・500ピン・テープBGA（H/Sp付き）（40×40）

μPD98502N7-H6

インデクス・マーク



端子名称

(1/3)

端子番号	端子名	端子番号	端子名	端子番号	端子名	端子番号	端子名	端子番号	端子名
A1	SMA13	B10	URSDO	C19	IC-PDnR	D28	PGTO2_B	F27	GND
A2	SMD0	B11	RMSL1	C20	IC-OPEN	D29	PRQI1_B	F28	GND
A3	SMD4	B12	MWDO	C21	JDI	D30	PAD0	F29	PAD5
A4	SMD7	B13	POM3	C22	GND	E1	GND	F30	PAD6
A5	SMD19	B14	POM5	C23	USBDM	E2	SDRAS_B	G1	SMA8
A6	SMD22	B15	EVDD	C24	IC-OPEN	E3	SMA0	G2	SMA15
A7	SRMCS_B	B16	IC-OPEN	C25	PUDGND	E4	SMA10	G3	SDCLK1
A8	URDSR_B	B17	IC-OPEN	C26	IVDD	E5	GND	G4	EVDD
A9	URDCD_B	B18	IC-PUpR	C27	PMODE	E6	EVDD	G5	SDCAS_B
A10	URDTR_B	B19	IC-OPEN	C28	PGTO3_B	E7	SMD16	G26	PAD1
A11	MWSK	B20	GND	C29	PGTO1_B	E8	GND	G27	PAD3
A12	MWDI	B21	IC-PDn	C30	PRQI0_B	E9	EVDD	G28	EVDD
A13	EXNMI_B	B22	JDO	D1	SDWE_B	E10	GND	G29	PAD7
A14	POM6	B23	GND	D2	SMA1	E11	RMSL0	G30	PAD8
A15	EXINT_B	B24	USBDP	D3	SMA11	E12	GND	H1	SMA4
A16	IVDD	B25	PUDVD	D4	IVDD	E13	POM0	H2	SMA7
A17	IC-PUpR	B26	IC-OPEN	D5	SMD3	E14	GND	H3	SMA9
A18	IC-PDnR	B27	PUMD_B	D6	SMD6	E15	POM7	H4	IVDD
A19	IC-OPEN	B28	PHINT_B	D7	EVDD	E16	GND	H5	GND
A20	IC-OPEN	B29	PRSTO_B	D8	IVDD	E17	GND	H26	GND
A21	IC-PDn	B30	PGTO0_B	D9	URCLK	E18	IC-OPEN	H27	IVDD
A22	JCK	C1	SMA2	D10	IVDD	E19	GND	H28	PCBE0_B
A23	JMS	C2	GND	D11	GND	E20	IC-PDn	H29	PAD9
A24	EVDD	C3	SMA16	D12	IVDD	E21	GND	H30	GND
A25	EVDD	C4	SMD2	D13	POM1	E22	EVDD	J1	SMA18
A26	PUAVD	C5	GND	D14	IVDD	E23	GND	J2	SMA3
A27	GND	C6	SMD18	D15	IC-PDnR	E24	USBCLK	J3	SMA5
A28	IC-OPEN	C7	SMD21	D16	BIG	E25	EVDD	J4	SMA6
A29	GND	C8	SRMOE_B	D17	IVDD	E26	GND	J5	EVDD
A30	PSERI_B	C9	GND	D18	IC-OPEN	E27	PRQI3_B	J26	EVDD
B1	SMA12	C10	URRTS_B	D19	IVDD	E28	PRQI2_B	J27	PAD10
B2	SMA14	C11	EVDD	D20	IC-OPEN	E29	PAD2	J28	PAD11
B3	SMD1	C12	MWCS	D21	IVDD	E30	PAD4	J29	PAD12
B4	SMD5	C13	POM2	D22	JRSTB_B	F1	SMA17	J30	PAD13
B5	SMD17	C14	POM4	D23	IVDD	F2	SDCKE1	K1	SMD31
B6	SMD20	C15	ENDCEN	D24	PUAGND	F3	SDCS_B	K2	SMA20
B7	SMD23	C16	GND	D25	PUSTBY	F4	GND	K3	SMA19
B8	URCTS_B	C17	IC-PDnR	D26	PARBN	F5	EVDD	K4	IVDD
B9	URSDI	C18	IC-OPEN	D27	IVDD	F26	EVDD	K5	GND

保守／廃止

(2/3)

端子番号	端子名	端子番号	端子名	端子番号	端子名	端子番号	端子名	端子番号	端子名
K26	GND	P5	GND	V4	IVDD	AB3	IC-PUp	AF2	MICRS
K27	IVDD	P26	GND	V5	GND	AB4	IC-PDn	AF3	MIMCLK
K28	PAD14	P27	IVDD	V26	PAD26	AB5	EVDD	AF4	MITD3
K29	PAD15	P28	PAD16	V27	PAD25	AB26	EVDD	AF5	GND
K30	EVDD	P29	PAD17	V28	PAD24	AB27	IC-OPEN	AF6	EVDD
L1	SDCLK0	P30	PAD18	V29	GND	AB28	IC-OPEN	AF7	MI2TE
L2	GND	R1	EVDD	V30	PSCLK	AB29	GND	AF8	GND
L3	SDCKE0	R2	SMD12	W1	IC-PUp	AB30	RSTB_B	AF9	EVDD
L4	SMD30	R3	SMD9	W2	IVDD	AC1	IC-PUp	AF10	GND
L5	EVDD	R4	SMD10	W3	IC-PUp	AC2	MIRD3	AF11	UDRD1
L26	PCBE1_B	R5	SMD11	W4	IVDD	AC3	GND	AF12	GND
L27	GND	R26	EVDD	W5	GND	AC4	IVDD	AF13	UDRAD2
L28	PAR	R27	PAD19	W26	GND	AC5	GND	AF14	GND
L29	PSERO_B	R28	PAD20	W27	IVDD	AC26	GND	AF15	UDTAD3
L30	PER_B	R29	GND	W28	PAD28	AC27	IVDD	AF16	UDTD7
M1	EVDD	R30	PAD21	W29	EVDD	AC28	IC-OPEN	AF17	GND
M2	SMD28	T1	SMD8	W30	PAD27	AC29	IC-OPEN	AF18	EVDD
M3	SMD29	T2	GND	Y1	IC-OPEN	AC30	PINT_B	AF19	GND
M4	IVDD	T3	CLKUSL1	Y2	PSDGND	AD1	MIRD2	AF20	UMWR_B
M5	GND	T4	CLKUSL0	Y3	PSAGND	AD2	MIRD1	AF21	GND
M26	GND	T5	EVDD	Y4	PSAVD	AD3	MIRCLK	AF22	EVDD
M27	IVDD	T26	PAD22	Y5	PSDVD	AD4	MIRER	AF23	GND
M28	PSTP_B	T27	IVDD	Y26	PAD31	AD5	MIRDV	AF24	GND
M29	PDSEL_B	T28	GND	Y27	PME_B	AD26	GND	AF25	EVDD
M30	EVDD	T29	PAD23	Y28	PRQO_B	AD27	EVDD	AF26	GND
N1	SMD24	T30	PCBE3_B	Y29	PAD30	AD28	IC-PDnR	AF27	GND
N2	SMD25	U1	CLKSL	Y30	PAD29	AD29	IC-OPEN	AF28	IC-PDnR
N3	GND	U2	GND	AA1	IC-OPEN	AD30	GND	AF29	IC-PDnR
N4	SMD26	U3	IC-PUp	AA2	PSTBY	AE1	MIRD0	AF30	IC-PDnR
N5	SMD27	U4	IVDD	AA3	PSMD_B	AE2	GND	AG1	MIMD
N26	PTRY_B	U5	GND	AA4	IVDD	AE3	MITER	AG2	GND
N27	PIRY_B	U26	GND	AA5	GND	AE4	IVDD	AG3	MITD2
N28	GND	U27	IVDD	AA26	GND	AE5	EVDD	AG4	IVDD
N29	PFRA_B	U28	PIDSEL	AA27	IVDD	AE26	EVDD	AG5	MI2COL
N30	PCBE2_B	U29	GND	AA28	PGTI_B	AE27	IC-PDnR	AG6	IVDD
P1	SMD13	U30	EVDD	AA29	GND	AE28	IC-PDnR	AG7	MI2CRS
P2	SMD14	V1	SCLK	AA30	EVDD	AE29	IC-PDnR	AG8	IVDD
P3	SMD15	V2	GND	AB1	IC-OPEN	AE30	IC-PDnR	AG9	UDRSC
P4	IVDD	V3	IC-PUp	AB2	GND	AF1	MITE	AG10	IVDD

(3/3)

端子番号	端子名	端子番号	端子名	端子番号	端子名	端子番号	端子名	端子番号	端子名
AG11	UDRD0	AH3	MICOL	AH25	EVDD	AJ17	UDTD5	AK9	UDRD5
AG12	IVDD	AH4	MI2RD0	AH26	IVDD	AJ18	GND	AK10	UDRD2
AG13	UDRAD1	AH5	MI2MD	AH27	UMAD8	AJ19	UMRDY_B	AK11	UDRCLK
AG14	IVDD	AH6	MI2TER	AH28	UMAD7	AJ20	GND	AK12	UDRAD3
AG15	UDTAD2	AH7	MI2TD3	AH29	UMAD3	AJ21	EVDD	AK13	UDRAD0
AG16	UDTAD0	AH8	GND	AH30	UMAD1	AJ22	UMD13	AK14	UDTE_B
AG17	IVDD	AH9	UDRCLV	AJ1	MITD0	AJ23	UMD9	AK15	UDTAD4
AG18	UDTD1	AH10	UDRD4	AJ2	MI2MCLK	AJ24	UMD7	AK16	UDTCLK
AG19	IVDD	AH11	UDTCLV	AJ3	MI2RD1	AJ25	UMD4	AK17	UDTD6
AG20	UMMD	AH12	IC-OPEN	AJ4	GND	AJ26	UMD1	AK18	UDTD3
AG21	IVDD	AH13	EVDD	AJ5	MI2RER	AJ27	GND	AK19	UDTD0
AG22	UMD10	AH14	UDTSC	AJ6	GND	AJ28	GND	AK20	UMRST_B
AG23	IVDD	AH15	EVDD	AJ7	MI2TD1	AJ29	UMAD6	AK21	UMRD_B
AG24	UMD2	AH16	UDTAD1	AJ8	UDRE_B	AJ30	UMAD4	AK22	UMD14
AG25	UMAD11	AH17	UDTD4	AJ9	UDRD6	AK1	MI2RD3	AK23	UMD12
AG26	UMAD9	AH18	UDTD2	AJ10	UDRD3	AK2	MI2RD2	AK24	UMD8
AG27	IVDD	AH19	UMINT_B	AJ11	UDRAD4	AK3	MI2RCLK	AK25	UMD6
AG28	UMAD2	AH20	UMSL_B	AJ12	IC-OPEN	AK4	MI2RDV	AK26	UMD3
AG29	UMAD0	AH21	UMD15	AJ13	GND	AK5	MI2TCLK	AK27	UMD0
AG30	IVDD	AH22	UMD11	AJ14	GND	AK6	MI2TD2	AK28	UMAD10
AH1	MITCLK	AH23	GND	AJ15	IVDD	AK7	MI2TD0	AK29	IC-PUp
AH2	MITD1	AH24	UMD5	AJ16	GND	AK8	UDRD7	AK30	UMAD5

特殊端子の名称：

IC-PDn　**：プルダウン**

IC-PDnR **：抵抗付きプルダウン**

IC-PUp　**：プルアップ**

IC-PUpR **：抵抗付きプルアップ**

備考　このドキュメントでは，アクティブ・ロウの端子をXXX_B**と表しています。**

保守／廃止

1.7 端子機能

このセクションでは，I/O欄の記号は次の状態を示します。

I　：入力
O　：出力
I/O　：双方向
I/OZ　：双方向（Hi-Z状態を含む）
I/OD　：双方向（オープン・ドレーン出力）
OZ　：出力（Hi-Z状態を含む）
OD　：出力（オープン・ドレーン）

1.7.1 電　源

端子名	端子番号	I/O	アクティブ・レベル	機　能
GND	A27, A29, B20, B23, C16, C2, C22, C5, C9, D11, E1, E10, E12, E14, E16, E17, E19, E21, E23, E26, E5, E8, F27, F28, F4, H26, H30, H5, K26, K5, L2, L27, M26, M5, N28, N3, P26, P5, R29, T2, T28, U2, U26, U29, U5, V2, V29, V5, W26, W5, AA26, AA29, AA5, AB2, AB29, AC26, AC3, AC5, AD26, AD30, AE2, AF10, AF12, AF14, AF17, AF19, AF21, AF23, AF24, AF26, AF27, AF5, AF8, AG2, AH23, AH8, AJ13, AJ14, AJ16, AJ18, AJ20, AJ27, AJ28, AJ4, AJ6			GND（0 V）
IVDD	A16, C26, D10, D12, D14, D17, D19, D21, D23, D27, D4, D8, H27, H4, K27, K4, M27, M4, P27, P4, T27, U27, U4, V4, W2, W27, W4, AA27, AA4, AC27, AC4, AE4, AG10, AG12, AG14, AG17, AG19, AG21, AG23, AG27, AG30, AG4, AG6, AG8, AH26, AJ15			内部ロジック・コア電源（+2.5 V）
EVDD	A24, A25, B15, C11, D7, E22, E25, E6, E9, F26, F5, G28, G4, J26, J5, K30, L5, M1, M30, R1, R26, T5, U30, W29, AA30, AB26, AB5, AD27, AE26, AE5, AF18, AF22, AF25, AF6, AF9, AH13, AH15, AH25, AJ21			外部（I/O）電源（+3.3 V）

1.7.2 システムPLL電源

端子名	端子番号	I/O	アクティブ・レベル	機　能
PSAGND	Y3			アナログ・グランド（0 V）
PSAVD	Y4			アナログ電源（+2.5 V）
PSDGND	Y2			デジタル・グランド（0 V）
PSDVD	Y5			デジタル電源（+2.5 V）

1. 7. 3　USB PLL電源

端子名	端子番号	I/O	アクティブ・レベル	機　能
PUAGND	D24			アナログ・グランド（0 V）
PUAVD	A26			アナログ電源（＋2.5 V）
PUDGND	C25			デジタル・グランド（0 V）
PUDVD	B25			デジタル電源（＋2.5 V）

1. 7. 4　システム制御インタフェース

端子名	端子番号	I/O	アクティブ・レベル	機　能
SCLK	V1	I		システム・クロック（33 MHz）
CLKSL	U1	I		VR4120AおよびSDRAMの動作クロック・セレクト （L：100 MHz, H：66 MHz）
PSMD_B	AA3	I	L	システムPLLモード制御（L：通常状態，H：スルー）注
PSTBY	AA2	I	H	システムPLLスタンバイ・モード制御 （L：アクティブ，H：スタンバイ）
PUMD_B	B27	I	L	USB PLLモード制御（L：通常状態，H：スルー）注
PUSTBY	D25	I	H	USB PLLスタンバイ・モード制御 （L：アクティブ，H：スタンバイ）
BIG	D16	I	H	VR4120ビッグ・エンディアン・モード
ENDCEN	C15	I		エンディアン・コンバージョン・イネーブル
EXINT_B	A15	I	L	外部割り込み
EXNMI_B	A13	I	L	外部ノンマスカブル割り込み
RSTB_B	AB30	I	L	システム・リセット
RMSL0, RMSL1	E11, B11	I		ROMアクセス・バス幅セレクト （RMSL1/0 = L/L：32ビット，L/H：16ビット，H/L：8ビット）

注　PSMD_BとPUMD_BはGNDに接続してください。

1.7.5 メモリ・インタフェース

端子名	端子番号	I/O	アクティブ・レベル	機　能
SDCLK0, SDCLK1	L1, G3	O		SDRAMクロック
SDCKE0, SDCKE1	L3, F2	O	H	SDRAMクロック・イネーブル
SDCS_B	F3	O	L	チップ・セレクト
SDRAS_B	E2	O	L	ロウ・アドレス・ストローブ
SDCAS_B	G5	O	L	カラム・アドレス・ストローブ
SDWE_B	D1	O	L	書き込みイネーブル
SRMCS_B	A7	O	L	PROM/フラッシュ・メモリ・チップ・セレクト
SRMOE_B	C8	O	L	PROM/フラッシュ・メモリ・アウトプット・イネーブル
SMA0-SMA20	E3, D2, C1, J2, H1, J3, J4, H2, G1, H3, E4, D3, B1, A1, B2, G2, C3, F1, J1, K3, K2	O		メモリ・アドレス
SMD0-SMD31	A2, B3, C4, D5, A3, B4, D6, A4, T1, R3, R4, R5, R2, P1, P2, P3, E7, B5, C6, A5, B6, C7, A6, B7, N1, N2, N4, N5, M2, M3, L4, K1	I/O		メモリ・データ

保守／廃止

1.7.6 PCIインタフェース

(1/2)

端子名	端子番号	I/O	アクティブ・レベル	機 能
PSCLK	V30	I		PCIクロック入力（33 MHz）
PARBN	D26	I	H	PCIアービタ・イネーブル NICモード：機能しない（GNDに接続してください） Hostモード：内部バス・アービタ制御 H…イネーブル L…ディスエーブル
PMODE	C27	I		PCIモード・セレクト（L：ホスト，H：NIC）
PIDSEL	U28	I	H	IDSEL入力 NICモード：IDSEL入力として機能 Hostモード：機能しない（GNDに接続してください）
PDSEL_B	M29	I/OZ	L	デバイス・セレクト（DEVSEL#信号）
PER_B	L30	I/OZ	L	パリティ・エラー（PERR#信号）
PFRA_B	N29	I/OZ	L	サイクル・フレーム（FRAME#信号）
PHINT_B	B28	I	L	PCIホスト割り込み NICモード：機能しない（EV_{DD}に接続してください） Hostモード：PCIホスト時のターゲットからのINT入力として機能
PINT_B	AC30	O	L	INT_A信号 NICモード：INT_A出力として機能 Hostモード：機能しない（オープンにしてください）
PIRY_B	N27	I/OZ	L	イニシエータ・レディ（IRDY#信号）
PME_B	Y27	OD	L	パワー・マネジメント・イベント NICモード：PME出力として機能 Hostモード：機能しない（オープンにしてください）
PRSTO_B	B29	O	L	PCIシステム・リセット・アウト（RST#信号） NICモード：機能しない（オープンにしてください） Hostモード：RST#出力として機能
PSERI_B	A30	I	L	システム・エラー・イン（SERR#信号の入力） NICモード：機能しない（EV_{DD}に接続してください） Hostモード：SERR#入力として機能
PSERO_B	L29	O	L	システム・エラー・アウト（SERR#信号の出力） NICモード：SERR#出力として機能 Hostモード：機能しない（オープンにしてください）
PTRY_B	N26	I/OZ	L	ターゲット・レディ（TRDY#信号）
PSTP_B	M28	I/OZ	L	ターゲットからのストップ・リクエスト（STOP#信号）
PCBE［0：3］_B	H28, L26, N30, T30	I/OZ	L	バス・コマンドとバイト・イネーブル
PRQO_B	Y28	O	L	バス・リクエスト・アウト NICモード：バス・リクエスト出力として機能 Hostモード：機能しない（オープンにしてください）
PRQI［0：3］_B	C30, D29, E28, E27	I	L	バス・リクエスト・イン NICモード：機能しない（EV_{DD}に接続してください） Hostモード：バス・リクエスト入力として機能

(2/2)

端子名	端子番号	I/O	アクティブ・レベル	機　能
PGTI_B	AA28	I	L	バス・グラント・イン NICモード：バス・グラント入力として機能 Hostモード：機能しない（EV_{DD}に接続してください）
PGTO [0:3] _B	B30, C29, D28, C28	O	L	バス・グラント・アウト NICモード：機能しない（オープンにしてください） Hostモード：バス・グラント出力として機能
PAR	L28	I/OZ		アドレス／データのパリティ
PAD0-PAD31	D30, G26, E29, G27, E30, F29, F30, G29, G30, H29, J27, J28, J29, J30, K28, K29, P28, P29, P30, R27, R28, R30, T26, T29, V28, V27, V26, W30, W28, Y30, Y29, Y26	I/OZ		PCIアドレスとデータ

保守／廃止

1. 7. 7　ATMインタフェース

1. 7. 7. 1　UTOPIAマネジメント・インタフェース

端子名	端子番号	I/O	アクティブ・レベル	機　能
UMMD	AG20	O		マネジメント・モード・セレクト
UMINT_B	AH19	I	L	PHYからの割り込み
UMRD_B	AK21	O	L	マネジメント読み込みイネーブル
UMRDY_B	AJ19	I	L	マネジメント・データ・レディ
UMRST_B	AK20	O	L	PHYリセット
UMSL_B	AH20	O	L	PHYセレクト
UMWR_B	AF20	O	L	マネジメント書き込みイネーブル
UMAD0- UMAD11	AG29, AH30, AG28, AH29, AJ30, AK30, AJ29, AH28, AH27, AG26, AK28, AG25	O		PHYアドレス
UMD0- UMD15	AK27, AJ26, AG24, AK26, AJ25, AH24, AK25, AJ24, AK24, AJ23, AG22, AH22, AK23, AJ22, AK22, AH21	I/O		マネジメント・データ

1. 7. 7. 2　UTOPIAデータ・インタフェース

端子名	端子番号	I/O	アクティブ・レベル	機　能
CLKUSL0, CLKUSL1	T4, T3	I		UTOPIAクロック・セレクト (CLKUSL1/0 = L/L：33 MHz, H/L：25 MHz, L/H：16.5 MHz)
UDRCLK	AK11	O		受信クロック
UDRCLV	AH9	I	H	受信セル・アベイラブル
UDRE_B	AJ8	O	L	受信イネーブル
UDRSC	AG9	I	H	受信セル開始
UDRAD0-UDRAD4	AK13, AG13, AF13, AK12, AJ11	O		受信PHYアドレス
UDRD0-UDRD7	AG11, AF11, AK10, AJ10, AH10, AK9, AJ9, AK8	I		受信データ
UDTCLK	AK16	O		送信クロック
UDTCLV	AH11	I	H	送信セル・アベイラブル
UDTE_B	AK14	O	L	送信イネーブル
UDTSC	AH14	O	H	送信セル開始
UDTAD0-UDTAD4	AG16, AH16, AG15, AF15, AK15	O		送信PHYアドレス
UDTD0-UDTD7	AK19, AG18, AH18, AK18, AH17, AJ17, AK17, AF16	O		送信データ

保守／廃止

1.7.8 イーサネット・インタフェース

1.7.8.1 イーサネット・インタフェース（チャネル1）

端子名	端子番号	I/O	アクティブ・レベル	機　能
MIMCLK	AF3	O		MIIマネジメント・クロック
MIMD	AG1	I/O		MIIマネジメント・データ
MICOL	AH3	I		コリジョン
MICRS	AF2	I		キャリア・センス
MIRCLK	AD3	I		受信クロック（2.5/25 MHz）入力
MIRDV	AD5	I		受信データ有効
MIRER	AD4	I		受信エラー
MIRD0-MIRD3	AE1, AD2, AD1, AC2	I		受信データ
MITCLK	AH1	I		送信クロック（2.5/25 MHz）入力
MITE	AF1	O		送信イネーブル
MITER	AE3	O		送信エラー
MITD0-MITD3	AJ1, AH2, AG3, AF4	O		送信データ

1.7.8.2 イーサネット・インタフェース（チャネル2）

端子名	端子番号	I/O	アクティブ・レベル	機　能
MI2MCLK	AJ2	O		MIIマネジメント・クロック
MI2MD	AH5	I/O		MIIマネジメント・データ
MI2COL	AG5	I		コリジョン
MI2CRS	AG7	I		キャリア・センス
MI2RCLK	AK3	I		受信クロック（2.5/25 MHz）入力
MI2RDV	AK4	I		受信データ有効
MI2RER	AJ5	I		受信エラー
MI2RD0-MI2RD3	AH4, AJ3, AK2, AK1	I		受信データ
MI2TCLK	AK5	I		送信クロック（2.5/25 MHz）入力
MI2TE	AF7	O		送信イネーブル
MI2TER	AH6	O		送信エラー
MI2TD0-MI2TD3	AK7, AJ7, AK6, AH7	O		送信データ

1. 7. 9　USBインタフェース

端子名	端子番号	I/O	アクティブ・レベル	機　能
USBCLK	E24	I		外部USBクロック（12 MHz）入力
USBDM	C23	I/O		USBデータ（−）
USBDP	B24	I/O		USBデータ（+）

1. 7. 10　UARTインタフェース

端子名	端子番号	I/O	アクティブ・レベル	機　能
URCLK	D9	I		UART外部クロック（18.432 MHz）入力
URCTS_B	B8	I	L	UART送信クリア
URDCD_B	A9	I	L	UARTデータ・キャリア検出
URDSR_B	A8	I	L	UARTデータ・セット・レディ
URDTR_B	A10	O	L	UARTデータ・ターミナル・レディ
URRTS_B	C10	O	L	UARTデータ送信要求
URSDI	B9	I		UARTシリアル・データ入力
URSDO	B10	O		UARTシリアル・データ出力

1. 7. 11　Micro Wireインタフェース

端子名	端子番号	I/O	アクティブ・レベル	機　能
MWCS	C12	O		Micro Wireチップ・セレクト
MWDI	A12	I		Micro Wireデータ入力
MWDO	B12	O		Micro Wireデータ出力
MWSK	A11	O		Micro Wireサンプリング・クロック出力

1. 7. 12　パラレル・ポート・インタフェース

端子名	端子番号	I/O	アクティブ・レベル	機　能
POM0-POM7	E13, D13, C13, B13, C14, B14, A14, E15	O		パラレル・ポート信号出力

保守／廃止

1. 7. 13　バウンダリ・スキャン・インタフェース

端子名	端子番号	I/O	アクティブ・レベル	機　能
JCK	A22	I		B-SCANクロック
JDI	C21	I		B-SCAN入力データ
JDO	B22	t/s O		B-SCAN出力データ
JMS	A23	I		B-SCANモード・セレクト
JRSTB_B	D22	I	L	B-SCANリセット

1. 7. 14　I.C.-オープン

端子名	端子番号	I/O	アクティブ・レベル	機　能
IC-OPEN	A19, A20, A28, B16, B17, B19, B26, C18, C20, C24, D18, D20, E18, Y1, AA1, AB1, AB27, AB28, AC28, AC29, AD29, AH12, AJ12	O		オープンにしてください。

1. 7. 15　I.C.-プルダウン

端子名	端子番号	I/O	アクティブ・レベル	機　能
IC-PDn	A21, B21, E20, AB4	I		GNDに接続してください。

1. 7. 16　I.C.-抵抗付きプルダウン

端子名	端子番号	I/O	アクティブ・レベル	機　能
IC-PDnR	A18, C17, C19, D15, AD28, AE27, AE28, AE29, AE30, AF28, AF29, AF30	I/O		プルダウン抵抗を介してGNDに接続してください。

1. 7. 17　I.C.-プルアップ

端子名	端子番号	I/O	アクティブ・レベル	機　能
IC-PUp	U3, V3, W1, W3, AB3, AC1, AK29	I		EVDDに接続してください。

保守／廃止

1. 7. 18　I.C.-抵抗付きプルアップ

端子名	端子番号	I/O	アクティブ・レベル	機　能
IC-PUpR	A17, B18	I/O		プルアップ抵抗を介してEVDDに接続してください。

保守／廃止

1. 8 I/Oレジスタ・マップ

ブロック	オフセット	レジスタ長（バイト）	名 称	VR4120Aによるアクセス	説 明
ATM	F000H	4	A_GMR	R/W	ジェネラル・モード・レジスタ
ATM	F004H	4	A_GSR	RC	ジェネラル・ステータス・レジスタ
ATM	F008H	4	A_IMR	R/W	割り込みマスク・レジスタ
ATM	F00CH	4	A_RQU	R	受信キュー・アンダフロー・レジスタ
ATM	F010H	4	A_RQA	R	受信キュー枯渇警告レジスタ
ATM	F014H	-	N/A	-	将来の使用のため予約
ATM	F018H	4	A_VER	R	バージョン・レジスタ
ATM	F01CH	-	N/A	-	将来の使用のため予約
ATM	F020H	4	A_CMR	R/W	コマンド・レジスタ
ATM	F024H	-	N/A	-	将来の使用のため予約
ATM	F028H	4	A_CER	R/W	コマンド拡張レジスタ
ATM	F02CH-F04CH	-	N/A	-	将来の使用のため予約
ATM	F050H	4	A_MSA0	R/W	メールボックス0スタート・アドレス・レジスタ
ATM	F054H	4	A_MSA1	R/W	メールボックス1スタート・アドレス・レジスタ
ATM	F058H	4	A_MSA2	R/W	メールボックス2スタート・アドレス・レジスタ
ATM	F05CH	4	A_MSA3	R/W	メールボックス3スタート・アドレス・レジスタ
ATM	F060H	4	A_MBA0	R/W	メールボックス0ボトム・アドレス・レジスタ
ATM	F064H	4	A_MBA1	R/W	メールボックス1ボトム・アドレス・レジスタ
ATM	F068H	4	A_MBA2	R/W	メールボックス2ボトム・アドレス・レジスタ
ATM	F06CH	4	A_MBA3	R/W	メールボックス3ボトム・アドレス・レジスタ
ATM	F070H	4	A_MTA0	R/W	メールボックス0テール・アドレス・レジスタ
ATM	F074H	4	A_MTA1	R/W	メールボックス1テール・アドレス・レジスタ
ATM	F078H	4	A_MTA2	R/W	メールボックス2テール・アドレス・レジスタ
ATM	F07CH	4	A_MTA3	R/W	メールボックス3テール・アドレス・レジスタ
ATM	F080H	4	A_MWA0	R/W	メールボックス0ライト・アドレス・レジスタ
ATM	F084H	4	A_MWA1	R/W	メールボックス1ライト・アドレス・レジスタ
ATM	F088H	4	A_MWA2	R/W	メールボックス2ライト・アドレス・レジスタ
ATM	F08CH	4	A_MWA3	R/W	メールボックス3ライト・アドレス・レジスタ
ATM	F090H	4	A_RCC	R	有効受信セル・カウンタ
ATM	F094H	4	A_TCC	R	有効送信セル・カウンタ
ATM	F098H	4	A_RUEC	R	受信無効VPI/VCIエラー・セル・カウンタ
ATM	F09CH	4	A_RIDC	R	受信内部破棄セル・カウンタ
ATM	F0A0H-F0BCH	-	N/A	-	将来の使用のため予約
ATM	F0C0H	4	A_T1R	R/W	T1タイム・レジスタ
ATM	F0C4H	-	N/A	-	将来の使用のため予約
ATM	F0C8H	4	A_TSR	R/W	タイム・スタンプ・レジスタ
ATM	F0CCH	-	N/A	-	将来の使用のため予約
ATM	F200H-F2FCH	-	N/A	-	VR4120Aコアからアクセスできません。
ATM	F300H	4	A_IBBAR	R/W	IBUSベース・アドレス・レジスタ
ATM	F304H	4	A_INBAR	R/W	命令ベース・アドレス・レジスタ
ATM	F308H-F31CH	-	N/A	-	将来の使用のため予約

保守／廃止

ブロック	オフセット	レジスタ長（バイト）	名 称	VR4120Aによるアクセス	説 明
ATM	F320H	4	A_UMCMD	R/W	UTOPIAマネジメント・インタフェース・コマンド・レジスタ
ATM	F324H-F3FCH	-	N/A	-	将来の使用のため予約
ATM	F400H-F4FCH	-	N/A	-	VR4120Aコアからアクセスできません。
ATM	F500H-FFFCH	-	N/A	-	将来の使用のため予約
PCI	000H	4	P_PLBA	R/W	PCI下位ベース・アドレス・レジスタ
PCI	004H	-	N/A	-	将来の使用のため予約
PCI	008H	4	P_IBBA	R/W	内部バス・ベース・アドレス・レジスタ
PCI	00CH	-	N/A	-	将来の使用のため予約
PCI	010H	4	P_VERR	R	バージョン・レジスタ
PCI	014H	4	P_PCAR	R/W	PCIコンフィギュレーション・アドレス・レジスタ
PCI	018H	4	P_PCDR	R/W	PCIコンフィギュレーション・データ・レジスタ
PCI	01CH	4	P_IGSR	RC	内部バス・ジェネラル・ステータス・レジスタ
PCI	020H	4	P_IIMR	R/W	内部バス割り込みマスク・レジスタ
PCI	024H	4	P_PGSR	R/W	PCIジェネラル・ステータス・レジスタ
PCI	028H	4	P_PIMR	R/W	PCI割り込みマスク・レジスタ
PCI	02CH	-	N/A	-	将来の使用のため予約
PCI	030H	4	P_HMCR	R/W	ホスト・モード制御レジスタ
PCI	034H-03CH	-	N/A	-	将来の使用のため予約
PCI	040H	4	P_PWCD	R/W	消費電力データ・レジスタ
PCI	044H	4	P_PWDD	R/W	損失電力データ・レジスタ
PCI	048H-04CH	-	N/A	-	将来の使用のため予約
PCI	050H	4	P_BCNT	R/W	ブリッジ制御レジスタ
PCI	054H	4	P_PPCR	R/W	電力制御レジスタ
PCI	058H	4	P_SWRR	-	ソフトウエア・リセット・レジスタ
PCI	05CH	4	P_RTMR	R/W	リトライ・タイマ・レジスタ
PCI	060H-0FCH	-	N/A	-	将来の使用のため予約
PCI	100H-1FCH	4	P_CONFIG	(*)	PCIコンフィギュレーション・スペース ＊いくつかのレジスタはリード／ライト可。ほかのレジスタはリード・オンリー（7. 5. 18参照）。
PCI	200H-FFCH	-	N/A	-	将来の使用のため予約
Ether	00H	4	En_MACC1	R/W	MACコンフィギュレーション・レジスタ1
Ether	04H	4	En_MACC2	R/W	MACコンフィギュレーション・レジスタ2
Ether	08H	4	En_IPGT	R/W	Back-to-Back IPGレジスタ
Ether	0CH	4	En_IPGR	R/W	Non Back-to-Back IPGレジスタ
Ether	10H	4	En_CLRT	R/W	コリジョン・レジスタ
Ether	14H	4	En_LMAX	R/W	最大パケット長レジスタ
Ether	18H-50H	-	N/A	-	将来の使用のため予約
Ether	54H	4	En_LSA2	R/W	ステーション・アドレス・レジスタ2
Ether	58H	4	En_LSA1	R/W	ステーション・アドレス・レジスタ1
Ether	5CH	4	En_PTVR	R	ポーズ・タイマ値リード・レジスタ
Ether	60H	-	N/A	-	将来の使用のため予約
Ether	64H	4	En_VLTP	R/W	VLANタイプ・レジスタ
Ether	80H	4	En_MIIC	R/W	MIIコンフィギュレーション・レジスタ

ブロック	オフセット	レジスタ長（バイト）	名 称	VR4120Aによるアクセス	説 明
Ether	84H-90H	-	N/A	-	将来の使用のため予約
Ether	94H	4	En_MCMD	W	MIIコマンド・レジスタ
Ether	98H	4	En_MADR	R/W	MIIアドレス・レジスタ
Ether	9CH	4	En_MWTD	R/W	MIIライト・データ・レジスタ
Ether	A0H	4	En_MRDD	R	MIIリード・データ・レジスタ
Ether	A4H	4	En_MIND	R	MIIインディケータ・レジスタ
Ether	A8H-C4H	-	N/A	-	将来の使用のため予約
Ether	C8H	4	En_AFR	R/W	アドレス・フィルタリング・レジスタ
Ether	CCH	4	En_HT1	R/W	ハッシュ・テーブル・レジスタ1
Ether	D0H	4	En_HT2	R/W	ハッシュ・テーブル・レジスタ2
Ether	D4H-D8H	-	N/A	-	将来の使用のため予約
Ether	DCH	4	En_CAR1	R/W	キャリー・レジスタ1
Ether	E0H	4	En_CAR2	R/W	キャリー・レジスタ2
Ether	E4H-12CH	-	N/A	-	将来の使用のため予約
Ether	130H	4	En_CAM1	R/W	キャリー・マスク・レジスタ1
Ether	134H	4	En_CAM2	R/W	キャリー・マスク・レジスタ2
Ether	138H-13CH	-	N/A	-	将来の使用のため予約
Ether	140H	4	En_RBYT	R/W	バイト受信カウンタ
Ether	144H	4	En_RPKT	R/W	パケット受信カウンタ
Ether	148H	4	En_RFCS	R/W	CRCエラー受信カウンタ
Ether	14CH	4	En_RMCA	R/W	マルチキャスト・パケット受信カウンタ
Ether	150H	4	En_RBCA	R/W	ブロードキャスト・パケット受信カウンタ
Ether	154H	4	En_RXCF	R/W	コントロール・フレーム・パケット受信カウンタ
Ether	158H	4	En_RXPF	R/W	ポーズ・フレーム受信カウンタ
Ether	15CH	4	En_RXUO	R/W	未定義コントロール・フレーム受信カウンタ
Ether	160H	4	En_RALN	R/W	アラインメント・エラー受信カウンタ
Ether	164H	4	En_RFLR	R/W	データ長不一致受信カウンタ
Ether	168H	4	En_RCDE	R/W	コード・エラー受信カウンタ
Ether	16CH	4	En_RFCR	R/W	False Carrier受信カウンタ
Ether	170H	4	En_RUND	R/W	ショート・パケット受信カウンタ
Ether	174H	4	En_ROVR	R/W	ジャバ・パケット受信カウンタ
Ether	178H	4	En_RFRG	R/W	エラー・ショート・パケット受信カウンタ
Ether	17CH	4	En_RJBR	R/W	エラー・ジャバ・パケット受信カウンタ
Ether	180H	4	En_R64	R/W	64バイト・フレーム受信カウンタ
Ether	184H	4	En_R127	R/W	65-127バイト・フレーム受信カウンタ
Ether	188H	4	En_R255	R/W	128-255バイト・フレーム受信カウンタ
Ether	18CH	4	En_R511	R/W	256-511バイト・フレーム受信カウンタ
Ether	190H	4	En_R1K	R/W	512-1023バイト・フレーム受信カウンタ
Ether	194H	4	En_RMAX	R/W	1023-RMAXバイト・フレーム受信カウンタ
Ether	198H	4	En_RVBT	R/W	有効バイト受信カウンタ
Ether	19CH-1BCH	-	N/A	-	将来の使用のため予約
Ether	1C0H	4	En_TBYT	R/W	バイト送信カウンタ
Ether	1C4H	4	En_TPKT	R/W	パケット送信カウンタ
Ether	1C8H	4	En_TFCS	R/W	CRCエラー送信パケット・カウンタ

保守／廃止

ブロック	オフセット	レジスタ長（バイト）	名 称	VR4120Aによるアクセス	説 明
Ether	1CCH	4	En_TMCA	R/W	マルチキャスト・パケット送信カウンタ
Ether	1D0H	4	En_TBCA	R/W	ブロードキャスト・パケット送信カウンタ
Ether	1D4H	4	En_TUCA	R/W	ユニキャスト・パケット送信カウンタ
Ether	1D8H	4	En_TXPF	R/W	ポーズ・コントロール・フレーム送信カウンタ
Ether	1DCH	4	En_TDFR	R/W	送信遅延カウンタ
Ether	1E0H	4	En_TXDF	R/W	送信過剰遅延カウンタ
Ether	1E4H	4	En_TSCL	R/W	シングル・コリジョン・パケット送信カウンタ
Ether	1E8H	4	En_TMCL	R/W	マルチ・コリジョン・パケット送信カウンタ
Ether	1ECH	4	En_TLCL	R/W	レイト・コリジョン・パケット・カウンタ
Ether	1F0H	4	En_TXCL	R/W	過剰コリジョン・パケット・カウンタ
Ether	1F4H	4	En_TNCL	R/W	トータル・コリジョン回数カウンタ
Ether	1F8H	4	En_TCSE	R/W	キャリア・センス・エラー・カウンタ
Ether	1FCH	4	En_TIME	R/W	MAC内部エラー数カウンタ
Ether	200H	4	En_TXCR	R/W	送信コンフィギュレーション・レジスタ
Ether	204H	4	En_TXFCR	R/W	送信FIFO制御レジスタ
Ether	208H-210H	-	N/A	-	将来の使用のため予約
Ether	214H	4	En_TXDPR	R/W	送信ディスクリプタ・レジスタ
Ether	218H	4	En_RXCR	R/W	受信コンフィギュレーション・レジスタ
Ether	21CH	4	En_RXFCR	R/W	受信FIFO制御レジスタ
Ether	220H-228H	-	N/A	-	将来の使用のため予約
Ether	22CH	4	En_RXDPR	R/W	受信ディスクリプタ・ポインタ・レジスタ
Ether	230H	4	En_RXPDR	R/W	受信プール・ディスクリプタ・レジスタ
Ether	234H	4	En_CCR	R/W	コンフィギュレーション・レジスタ
Ether	238H	4	En_ISR	RC	割り込み処理レジスタ
Ether	23CH	4	En_MSR	R/W	マスク処理レジスタ
Ether	240H-FFCH	-	N/A	-	将来の使用のため予約
SYSCNT	00H	4	S_GMR	R/W	ジェネラル・モード・レジスタ
SYSCNT	04H	4	S_GSR	R	ジェネラル・ステータス・レジスタ
SYSCNT	08H	4	S_ISR	RC	割り込みステータス・レジスタ
SYSCNT	0CH	4	S_IMR	R/W	割り込みマスク・レジスタ
SYSCNT	10H	4	S_NSR	RC	NMIステータス・レジスタ
SYSCNT	14H	4	S_NER	R/W	NMIイネーブル・レジスタ
SYSCNT	18H	4	S_VER	R	バージョン・レジスタ
SYSCNT	1CH	4	S_IOR	R/W	I/Oポート・レジスタ
SYSCNT	20H-2CH	-	N/A	-	将来の使用のため予約
SYSCNT	30H	4	S_WRCR	W	ウォーム・リセット制御レジスタ
SYSCNT	34H	4	S_WRSR	R	ウォーム・リセット・ステータス・レジスタ
SYSCNT	38H	4	S_PWCR	R/W	電力制御レジスタ
SYSCNT	3CH	4	S_PWSR	R	電力ステータス・レジスタ
SYSCNT	40H-48H	-	N/A	-	将来の使用のため予約
SYSCNT	4CH	4	ITCNTR	R/W	IBUSタイムアウト・タイマ制御レジスタ
SYSCNT	50H	4	ITSETR	R/W	IBUSタイムアウト・タイマ設定レジスタ
SYSCNT	54H-7CH	-	N/A	-	将来の使用のため予約
SYSCNT	80H	4	UARTDLL	R/W	UARTディバイザ・ラッチLSBレジスタ［DLAB = 1］

保守／廃止

ブロック	オフセット	レジスタ長（バイト）	名 称	VR4120Aによるアクセス	説 明
SYSCNT	80H	4	UARTRBR	R	UARTレシーバ・バッファ・レジスタ [DLAB = 0，リード]
SYSCNT	80H	4	UARTTHR	W	UARTトランスミッタ・データ保持レジスタ [DLAB = 0，ライト]
SYSCNT	84H	4	UARTDLM	R/W	UARTディバイザ・ラッチMSBレジスタ［DLAB = 1］
SYSCNT	84H	4	UARTIER	R/W	UART割り込みイネーブル・レジスタ［DLAB = 0］
SYSCNT	88H	4	UARTFCR	W	UART FIFO制御レジスタ［ライト］
SYSCNT	88H	4	UARTIIR	R	UART割り込みIDレジスタ［リード］
SYSCNT	8CH	4	UARTLCR	R/W	UARTライン制御レジスタ
SYSCNT	90H	4	UARTMCR	R/W	UARTモデム制御レジスタ
SYSCNT	94H	4	UARTLSR	R/W	UARTライン・ステータス・レジスタ
SYSCNT	98H	4	UARTMSR	R/W	UARTモデム・ステータス・レジスタ
SYSCNT	9CH	4	UARTSCR	R/W	UARTスクラッチ・レジスタ
SYSCNT	A0H	4	DSUCNTR	R/W	ウォッチドッグ・タイマ制御レジスタ
SYSCNT	A4H	4	DSUSETR	R/W	ウォッチドッグ・タイマ・タイム設定レジスタ
SYSCNT	A8H	4	DSUCLRR	W	ウォッチドッグ・タイマ・クリア・レジスタ
SYSCNT	ACH	4	DSUTIMR	R	ウォッチドッグ・タイマ経過時間レジスタ
SYSCNT	B0H	4	TMMR	R/W	タイマ・モード・レジスタ
SYSCNT	B4H	4	TM0CSR	R/W	タイマ・チャネル0カウント設定レジスタ
SYSCNT	B8H	4	TM1CSR	R/W	タイマ・チャネル1カウント設定レジスタ
SYSCNT	BCH	4	TM0CCR	R	タイマ・チャネル0カレント・カウント・レジスタ
SYSCNT	C0H	4	TM1CCR	R	タイマ・チャネル1カレント・カウント・レジスタ
SYSCNT	C4H-CCH	-	N/A	-	将来の使用のため予約
SYSCNT	D0H	4	ECCR	W	EEPROM™コマンド制御レジスタ
SYSCNT	D4H	4	ERDR	R	EEPROMリード・データ・レジスタ
SYSCNT	D8H	4	MACAR1	R	MACアドレス・レジスタ1
SYSCNT	DCH	4	MACAR2	R	MACアドレス・レジスタ2
SYSCNT	E0H	4	MACAR3	R	MACアドレス・レジスタ3
SYSCNT	E4H-FCH	-	N/A	-	将来の使用のため予約
SYSCNT	100H	4	RMMDR	R/W	ROMモード・レジスタ
SYSCNT	104H	4	RMATR	R/W	ROMアクセス・タイミング・レジスタ
SYSCNT	108H	4	SDMDR	R/W	SDRAMモード・レジスタ
SYSCNT	10CH	4	SDTSR	R/W	SDRAMタイプ選択レジスタ
SYSCNT	110H	4	SDPTR	R/W	SDRAMプリチャージ・タイミング・レジスタ
SYSCNT	114H-118H	-	N/A	-	将来の使用のため予約
SYSCNT	11CH	4	SDRMR	R/W	SDRAMリフレッシュ・モード・レジスタ
SYSCNT	120H	4	SDRCR	R	SDRAMリフレッシュ・タイマ・カウント・レジスタ
SYSCNT	124H	4	MBCR	R/W	メモリ・バス制御レジスタ
SYSCNT	128H-FFCH	-	N/A	-	将来の使用のため予約
USB	00H	4	U_GMR	R/W	USBジェネラル・モード・レジスタ
USB	04H	4	U_VER	R	USBフレーム・ナンバ／バージョン・レジスタ
USB	08H-0CH	-	N/A	-	将来の使用のため予約
USB	10H	4	U_GSR1	RC	USBジェネラル・ステータス・レジスタ1
USB	14H	4	U_IMR1	R/W	USB割り込みマスク・レジスタ1

保守／廃止

ブロック	オフセット	レジスタ長（バイト）	名 称	VR4120Aによるアクセス	説 明
USB	18H	4	U_GSR2	RC	USBジェネラル・ステータス・レジスタ2
USB	1CH	4	U_IMR2	R/W	USB割り込みマスク・レジスタ2
USB	20H	4	U_EP0CR	R/W	USB EP0制御レジスタ
USB	24H	4	U_EP1CR	R/W	USB EP1制御レジスタ
USB	28H	4	U_EP2CR	R/W	USB EP2制御レジスタ
USB	2CH	4	U_EP3CR	R/W	USB EP3制御レジスタ
USB	30H	4	U_EP4CR	R/W	USB EP4制御レジスタ
USB	34H	4	U_EP5CR	R/W	USB EP5制御レジスタ
USB	38H	4	U_EP6CR	R/W	USB EP6制御レジスタ
USB	3CH	-	N/A	-	将来の使用のため予約
USB	40H	4	U_CMR	R/W	USBコマンド・レジスタ
USB	44H	4	U_CA	R/W	USBコマンド拡張レジスタ
USB	48H	4	U_TEPSR	R	USB Txエンドポイント・ステータス・レジスタ
USB	4CH	-	N/A	-	将来の使用のため予約
USB	50H	4	U_RP0IR	R/W	USB Rxプール0情報レジスタ
USB	54H	4	U_RP0AR	R	USB Rxプール0アドレス・レジスタ
USB	58H	4	U_RP1IR	R/W	USB Rxプール1情報レジスタ
USB	5CH	4	U_RP1AR	R	USB Rxプール1アドレス・レジスタ
USB	60H	4	U_RP2IR	R/W	USB Rxプール2情報レジスタ
USB	64H	4	U_RP2AR	R	USB Rxプール2アドレス・レジスタ
USB	68H-6CH	-	N/A	-	将来の使用のため予約
USB	70H	4	U_TMSA	R/W	USB Txメールボックス・スタート・アドレス・レジスタ
USB	74H	4	U_TMBA	R/W	USB Txメールボックス・ボトム・アドレス・レジスタ
USB	78H	4	U_TMRA	R/W	USB Txメールボックス・リード・アドレス・レジスタ
USB	7CH	4	U_TMWA	R	USB Txメールボックス・ライト・アドレス・レジスタ
USB	80H	4	U_RMSA	R/W	USB Rxメールボックス・スタート・アドレス・レジスタ
USB	84H	4	U_RMBA	R/W	USB Rxメールボックス・ボトム・アドレス・レジスタ
USB	88H	4	U_RMRA	R/W	USB Rxメールボックス・リード・アドレス・レジスタ
USB	8CH	4	U_RMWA	R	USB Rxメールボックス・ライト・アドレス・レジスタ
USB	90H-FFCH	-	N/A	-	将来の使用のため予約

ベース・アドレス

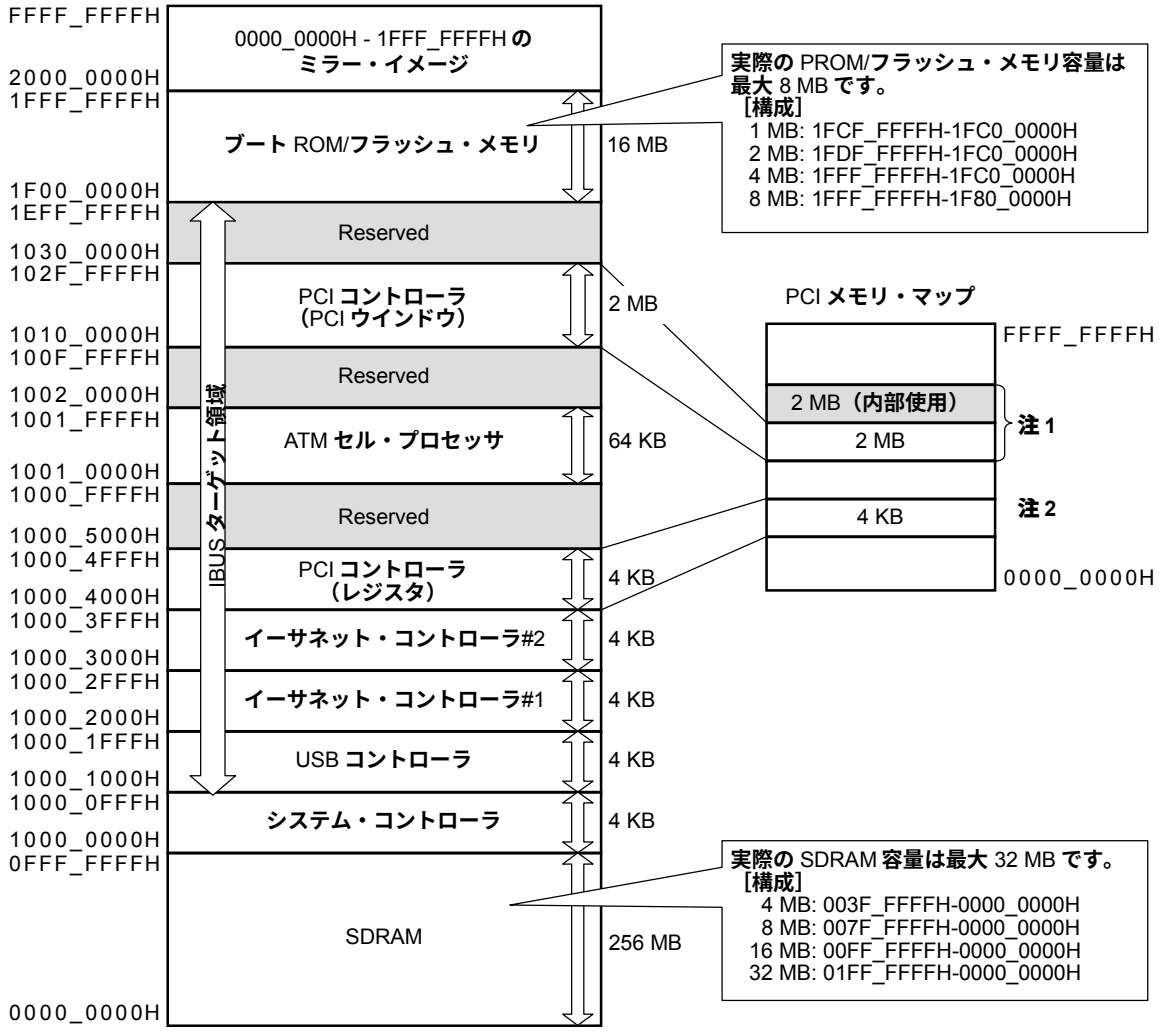
ATM**セル・プロセッサ**（ATM）	－1001_0000H
PCI**コントローラ**（PCI）	－1000_4000H
イーサネット・コントローラ（Ether）#1（n = 1）	－1000_2000H
イーサネット・コントローラ（Ether）#2（n = 2）	－1000_3000H
USB**コントローラ**（USB）	－1000_1000H
システム・コントローラ（SYSCNT）	－1000_0000H

保守／廃止

1.9 メモリ・マップ

VR4120Aコアは32ビット・アドレスを使用するので，プロセッサの物理アドレス空間が4 Gバイトになります。この4 Gバイトの物理アドレス空間をμPD98502では図1－10のように使用しています。

図1－10　メモリ・マップ



注1. **この領域は**P_PLBA，P_IBBA**レジスタで割り当てられている**μPD98502**の**PCI**ウインドウです。**

7. 5. 2　P_PLBA**（**PCI**下位ベース・アドレス・レジスタ）および**7. 5. 3　P_IBBA**（内部ベース・アドレス・レジスタ）を参照してください。**

2. **この領域はレジスタ・メモリ・ベース・アドレス・レジスタで割り当てられている**μPD98502**の**I/O**領域の**PCI**ウインドウです。**

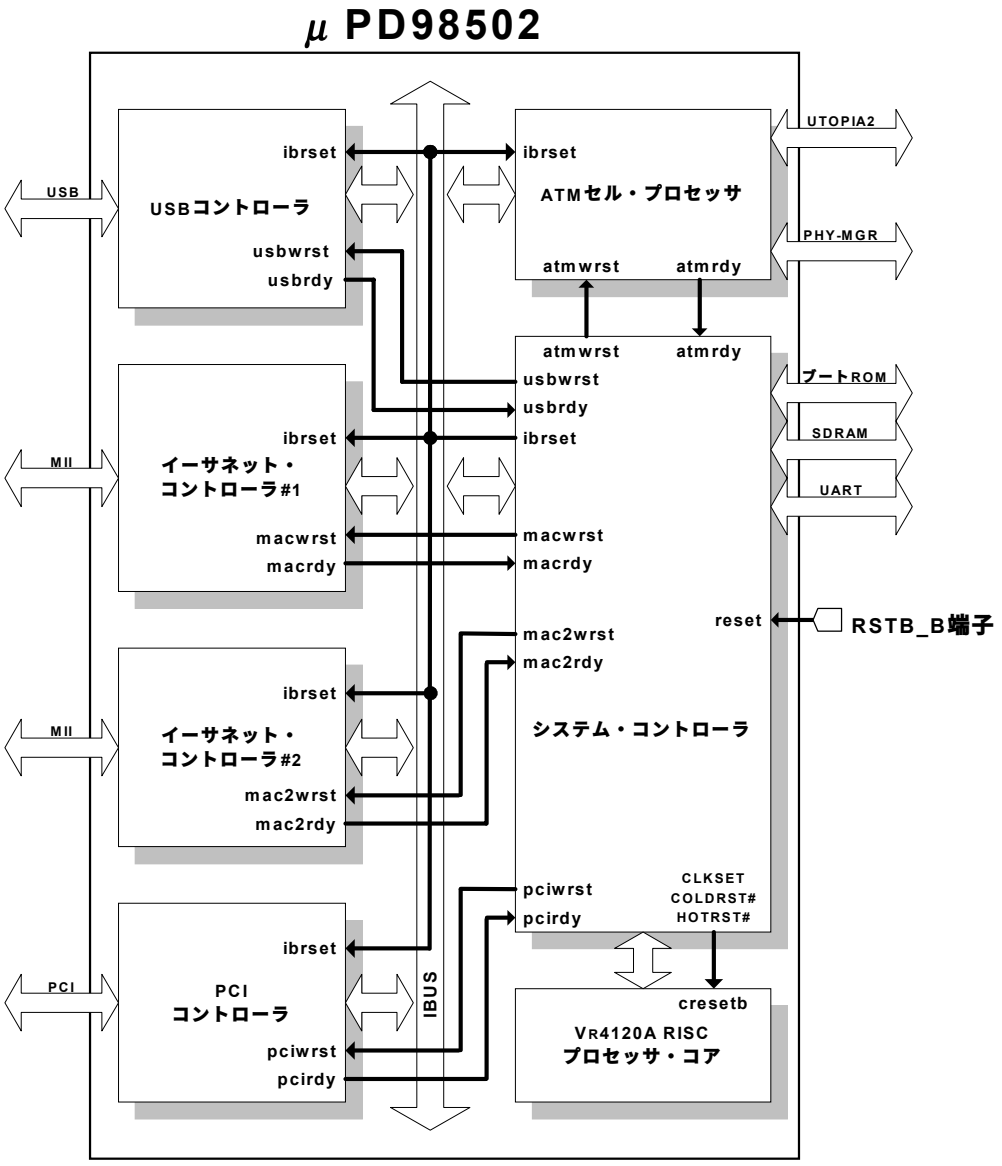
7. 5. 18. 12　**レジスタ・メモリ・ベース・アドレス・レジスタを参照してください。**

保守／廃止

1. 10　リセット・コンフィギュレーション

クロック・コントロール・ユニット（CCU）に対してリセット・ライン（RSTB_B）の立ち下がりエッジが，μPD98502の内部リセットとして機能します。システム・コントローラは，μPD98502に対するチップ・リセットであるRSTB_B端子入力を用いて，IBUSリセット信号（ibrset）を生成します。IBUSクロック（SDCLK）の4パルス分あとに，システム・コントローラはIBUSリセット信号（ibrset）をIBUSクロック（SCLK入力が33 MHzのとき66 MHz）と同期してディアサートします。これによって，IBUSに接続している周辺ブロックがリセットされ，初期状態となります。また，システム・コントローラは，VR4120Aのコールド・リセットを行うため，内部のコールド・リセット信号（COLDRST#）とホット・リセット信号（HOTRST#）を生成します。μPD98502に電力が供給され始めると，システム・コントローラは，RSTB_B信号の立ち下がりエッジで，内部のCLKSET信号，内部コールド・リセット（COLDRST#）信号，内部ホット・リセット（HOTRST#）信号をアサートします。μPD98502では，ホット・リセットのみをアクティブにする仕組みはなく，したがって，ソフト・リセットはサポートしていません。

図1－11　リセット・コンフィギュレーション

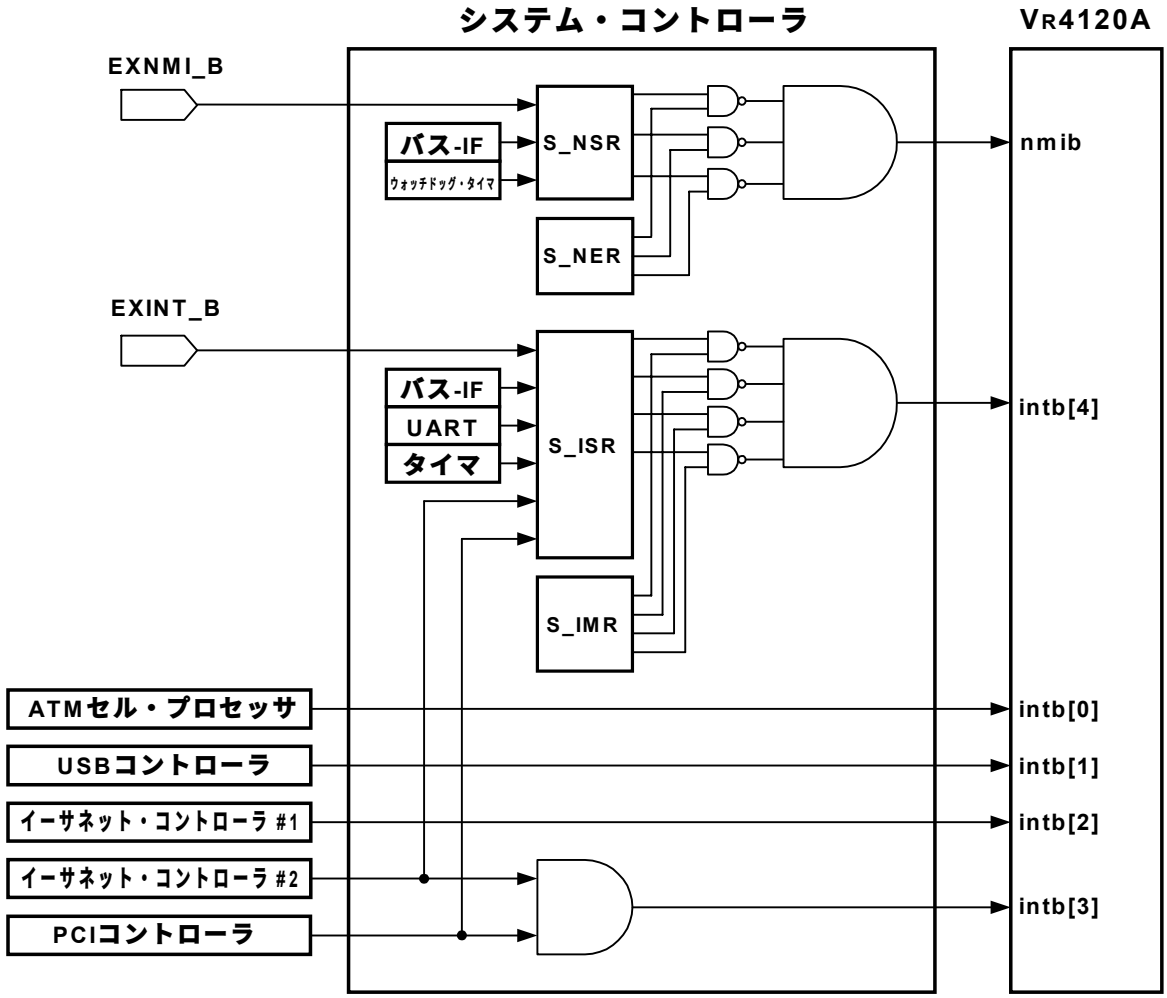


保守／廃止

1. 11　割り込み

システム・コントローラは，VR4120Aに対するマスカブル割り込みとノンマスカブル割り込みの制御をサポートします。

図1－12　割り込み信号の接続

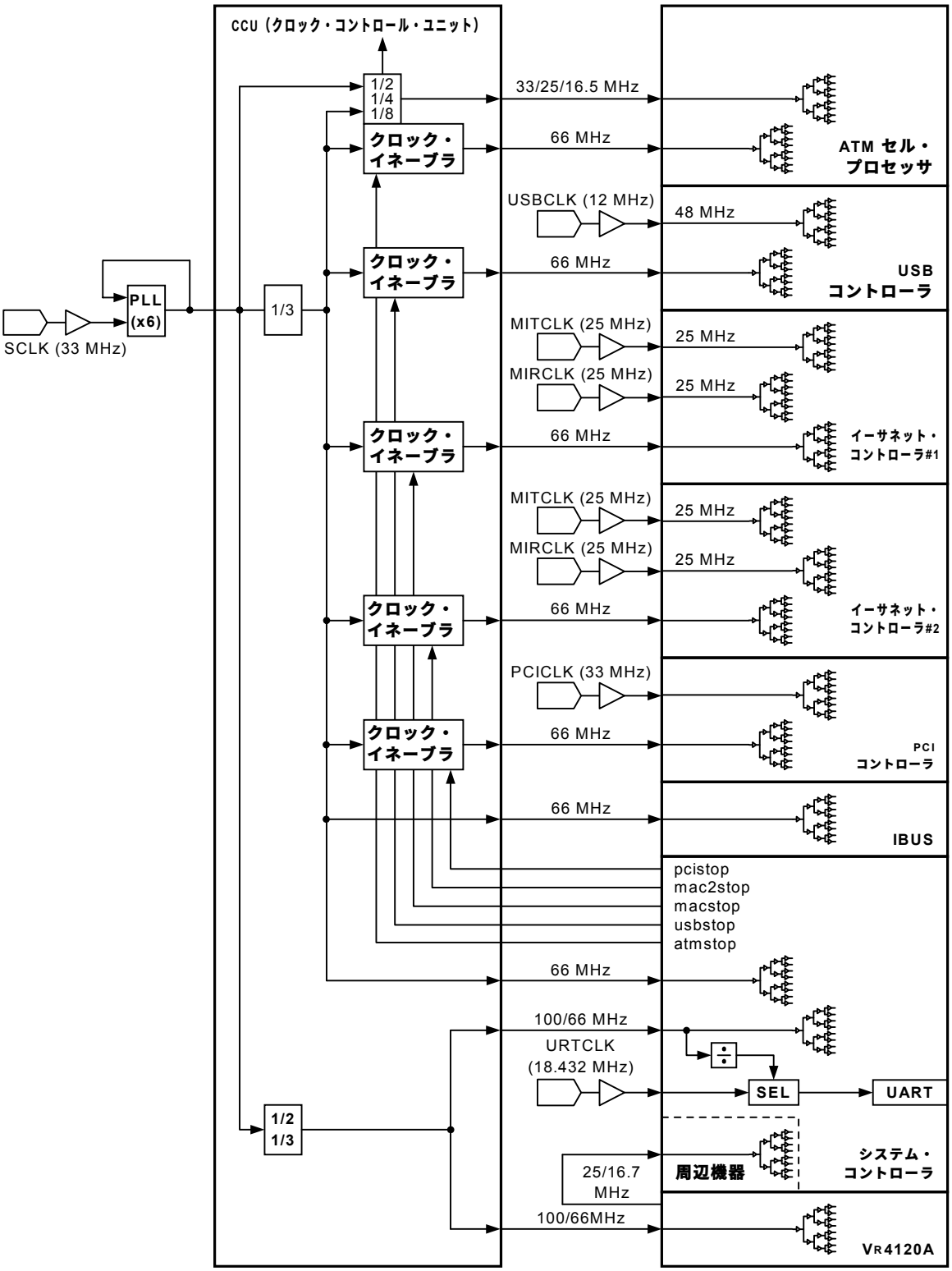


保守／廃止

1. 12　クロック・コントロール・ユニット

μPD98502の内部クロックは，以下の図で示す各クロック信号がクロック・コントロール・ユニット（CCU）で生成されています。

図1－13　クロック・コントロール・ユニットのブロック図



第2章 VR4120A

注意　μPD98502 は，MIPS16 命令に対応していません。

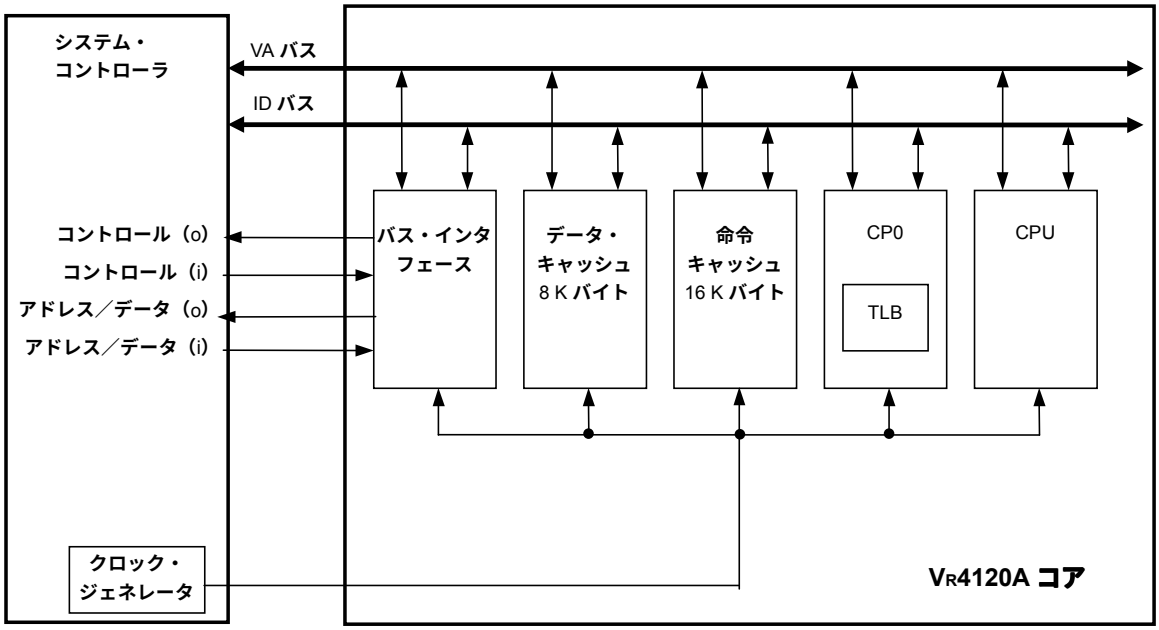
この章では，VR4120A RISC **プロセッサ・コアの動作（**MIPS **命令，パイプラインなど）について説明しています。このマニュアルでは，**VR4120A RISC **プロセッサ・コアを「**VR4120A**」，「**VR4120A **コア」と省略して記述しています。**

2.1 VR4120A の概要

図 2-1**に** VR4120A **コアの内部ブロック図を示します。**

VR4120A **コアは，高性能整数演算ユニットに加えて，**1 **エントリに** 2 **つのページを対応させた** 32 **のエントリを持つフルアソシアティブ形式の高速変換緩衝機構（**TLB**），命令キャッシュとデータ・キャッシュ，およびバス・インタフェースを持っています。**

図 2-1 VR4120A コア内部ブロック図



保守／廃止

2.1.1 内部ブロック構成

2.1.1.1 CPU

整数演算を行うブロックです。64 ビットのレジスタ・ファイル，64 ビットの整数データ・パス，積和演算器を備えています。

2.1.1.2 コプロセッサ 0

メモリ管理ユニット（MMU）と例外処理機能を内蔵しています。MMU はアドレス変換を行い，異なる種類（ユーザ，スーパバイザ，またはカーネル）のメモリ・セグメント間でのアクセスをチェックします。仮想アドレスから物理アドレスへの変換は TLB（高速変換緩衝機構）が行います。

2.1.1.3 命令キャッシュ

ダイレクト・マッピング，仮想インデクス，物理タグ方式で，容量は 16 K バイトです。

2.1.1.4 データ・キャッシュ

ダイレクト・マッピング，仮想インデクス，物理タグ，ライトバック方式で，容量は 8 K バイトです。

2.1.1.5 バス・インタフェース

VR4120A コアと周辺ユニットの 1 つであるシステム・コントローラとのデータのやりとりを制御します。入力用と出力用の 2 つの 32 ビットのマルチプレクスされたアドレス／データ・バス，クロック信号，割り込み要求信号，および各種制御信号があります。

保守／廃止

2.1.2 VR4120A コアのレジスタ

VR4120A コアには，次に示すレジスタがあります。

- 汎用レジスタ（GPR）：64 ビット×32 個
- PC：プログラム・カウンタ（64 ビット）
- HI レジスタ：整数乗除算結果の上位ダブル・ワードを格納（64 ビット）
- LO レジスタ：整数乗除算結果の下位ダブル・ワードを格納（64 ビット）

また，汎用レジスタのうち 2 つに，次の機能を割り当てています。

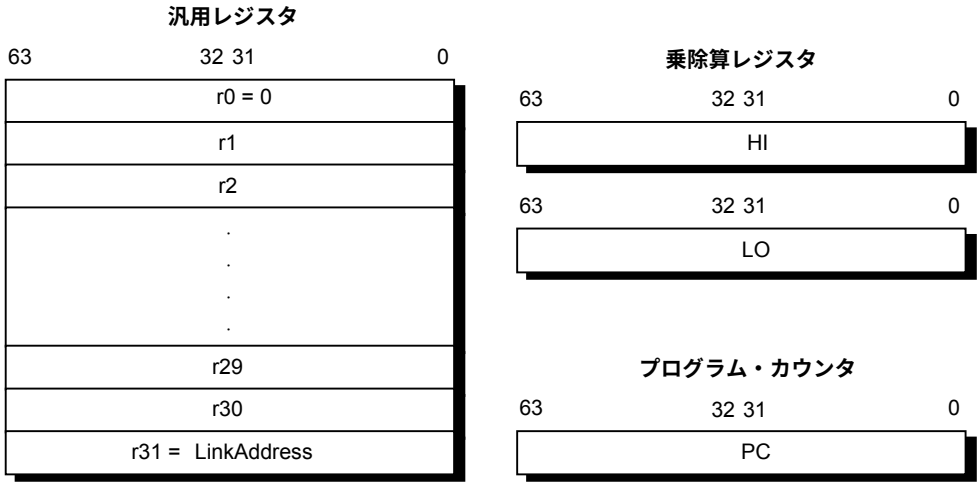
- r0 は，ゼロ固定のため，結果が廃棄される命令の対象レジスタとして使用できます。また，ゼロ値が必要なときにソース・レジスタとしても使用できます。
- r31 は，JAL（Jump and Link）命令などのリンク命令で使用するリンク・レジスタです。ほかの命令でも使用できますが，リンク命令でのレジスタの使用とほかの演算でのデータの使用が重複しないように注意してください。

さらに，例外処理，アドレス管理などを行う CP0（システム制御コプロセッサ）内のレジスタ群があります。

CPU レジスタは 32 ビットか 64 ビットのレジスタとして動作しますが，どちらで動作するかはプロセッサの動作モードで決まります。

図 2-2に CPU レジスタを示します。

図 2-2 VR4120A コアのレジスタ



プログラム・ステータス・ワード（PSW）は存在しません。その機能は，システム制御コプロセッサ（CP0）に組み込まれたステータス・レジスタと原因レジスタという 2 つのレジスタが代行します。CP0 のレジスタについては，**2.1.5 システム制御コプロセッサ（CP0）**を参照してください。

保守／廃止

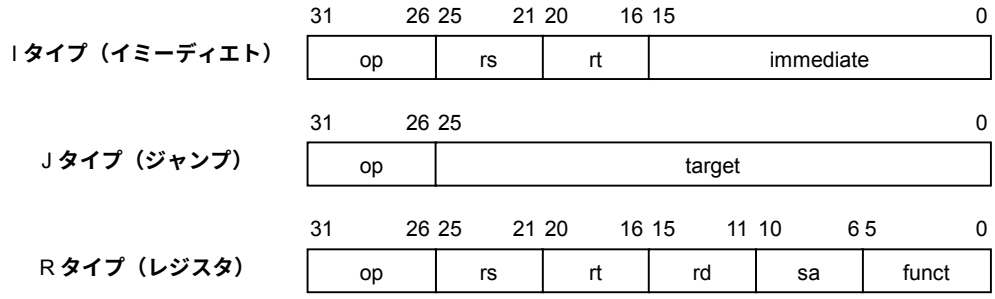
2.1.3 VR4120A コアの命令セット概略

CPU 命令は，32 ビット長（MIPS III）命令のみです。

2.1.3.1 MIPS III 命令

MIPSIII 命令実行時，CPU 命令はすべて 32 ビット長です。図 2-3に示すようにイミーディエト（I タイプ），ジャンプ（J タイプ），レジスタ（R タイプ）の 3 種類の命令形式に分類されます。命令形式の各フィールドについては，2.2　MIPS III 命令セット概要を参照してください。

図 2-3 CPU 命令形式（32 ビット長命令）



命令セットは次の 5 つのグループに分類されます。

(a) ロード／ストア命令は，メモリと汎用レジスタの間のデータ転送を行います。命令形式はすべて I タイプ（イミーディエト）です。アドレシング・モードは，ベース・レジスタに 16 ビットの符号付きオフセットを加える形式だけをサポートしています。

(b) 算術命令は，レジスタの値に対して算術演算，論理演算，シフト操作，乗除算演算を実行します。命令形式は，R タイプ（両オペランドとその結果をレジスタに格納）と I タイプ（オペランドの 1 つは 16 ビットの符号付きイミーディエト値）です。

(c) ジャンプ／ブランチ命令は，プログラムの制御の流れを変更します。ジャンプ命令では，26 ビットのターゲット・アドレスとプログラム・カウンタの上位ビットとを結合して生成される絶対アドレス（J タイプの場合）か，レジスタの指すアドレス（R タイプの場合）のいずれかにジャンプします。ブランチ命令の形式は，I タイプです。プログラム・カウンタ相対の 16 ビット・オフセット・アドレスに分岐します。JAL 命令は戻りアドレスをレジスタ 31 に退避します。

(d) システム制御コプロセッサ（CP0）命令は，プロセッサのメモリ管理や例外処理を行うため，CP0 レジスタに対するオペレーションを実行します。

(e) 特殊命令は，システム・コール例外やブレークポイント例外を発生させたり，比較結果に基づいて汎用例外処理ベクタに分岐させたりします。命令形式は，R タイプと I タイプです。

各命令の動作については，2.2　MIPS III 命令セット概要，付録 A　MIPS III 命令セットを参照してください。

保守／廃止

2.1.4 データ形式とアドレシング

VR4120A コアは，次の 4 種類のデータ形式を使用しています。

- ダブル・ワード（64 ビット）
- ワード（32 ビット）
- ハーフ・ワード（16 ビット）
- バイト（8 ビット）

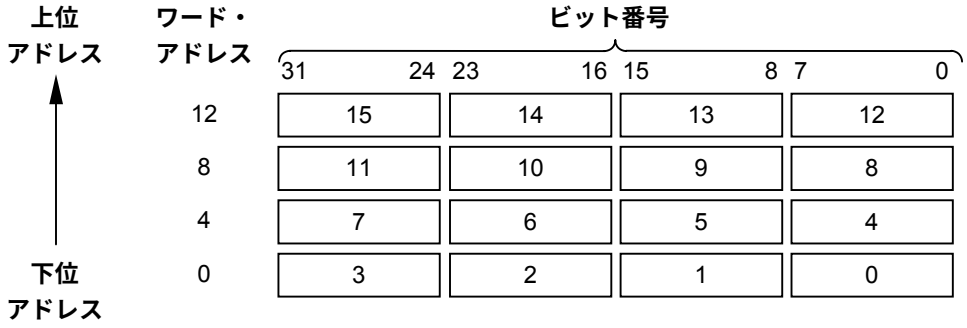
データ形式がダブル・ワード，ワード，ハーフ・ワードの場合，バイトの並び方を，ビッグ・エンディアンかリトル・エンディアンのどちらかに設定できます。ただし，μPD98502 では，リトル・エンディアンだけをサポートしています。

エンディアンは，複数バイトのデータ構造の中のバイト 0 の位置に着目して定義しています。

リトル・エンディアンのシステムの場合，バイト 0 が最下位（最も右の）バイトになります。この並び方は，iAPX™ や DEC VAX™ の採用している方法と互換性があります。図 2-4，図 2-5 に，その構成を示します。

なお，このマニュアル中のビット表示はリトル・エンディアンです。

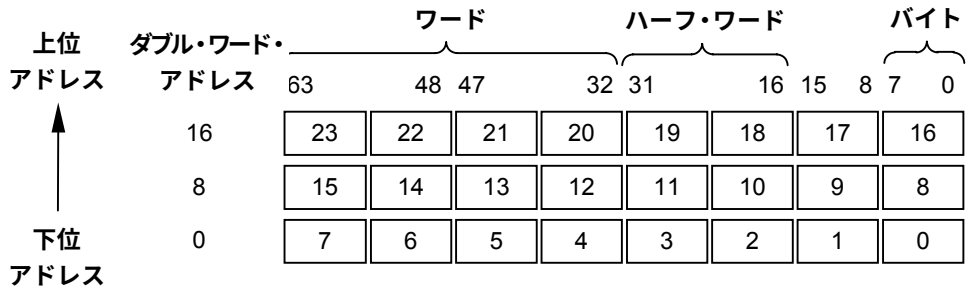
図 2-4 ワード内のバイト・アドレス：リトル・エンディアン



備考 1. 最下位バイトが最下位アドレスです。

2. ワードは最下位バイトのアドレスでアドレス指定されます。

図 2-5 ダブル・ワード内のバイト・アドレス：リトル・エンディアン



備考 1. 最下位バイトが最下位アドレスです。

2. ワードは最下位バイトのアドレスでアドレス指定されます。

VR4120A コアは，ハーフ・ワード，ワード，およびダブル・ワードをアクセスする場合，次に示すような境界に合わせたアドレスを使用します。

- ハーフ・ワード： 偶数バイト境界（0, 2, 4...）
- ワード： 4 バイト境界（0, 4, 8...）
- ダブル・ワード： 8 バイト境界（0, 8, 16...）

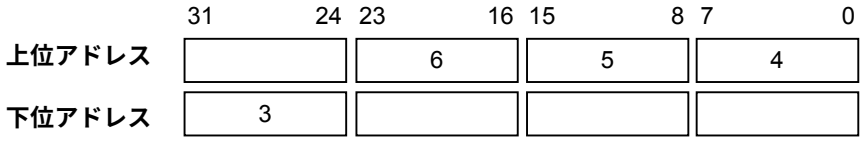
4 バイト境界（ワード）または 8 バイト境界（ダブル・ワード）に位置していないデータをロードおよびストアする場合は，次の特殊命令を使用します。

LWL	LWR	SWL	SWR
LDL	LDR	SDL	SDR

これらの命令は，境界に配列していないデータにアクセスする際に使用され，必ずペアで用います。境界に配列していないデータをアクセスする場合，境界に配列したデータをアクセスするときに必要な命令サイクルより 1 パイプライン・クロック・サイクル余分に必要です。

バイト・アドレス 3 を持つ，位置合わせされていないワードをアクセスしたときの様子を，図 2-6に示します。

図 2-6 位置合わせされていないワードのバイト・アドレス（リトル・エンディアン）



保守／廃止

2.1.5 システム制御コプロセッサ（CP0）

MIPS の ISA では，4 種類のコプロセッサ（CP0-CP3）を定義します。

- CP0 は，仮想アドレスから物理アドレスへの変換，オペレーティング・モード（カーネル，スーパバイザ，ユーザ）の切り替え，および例外の管理を行います。また，キャッシュ・サブシステムを制御し，原因分析，エラーからの復帰を行います。
- CP1 は，浮動小数点命令を実行するために予約されています。
- CP2 は，MIPS が将来定義するときのために予約されています。
- CP3 は，定義されていません。CP3 の命令は将来の拡張用に予約されています。

CP0 レジスタを図 2-7に示します。また，表 2-1に各レジスタの簡単な説明を示します。仮想メモリ・システムに関するレジスタの詳細な説明は**2.4 メモリ管理システム**，例外処理に関するレジスタの説明は**2.5 例外処理**を参照してください。

図 2-7 CP0 レジスタ

レジスタ番号	レジスタ名	レジスタ番号	レジスタ名
0	インデクス注1	16	コンフィグ注1
1	ランダム注1	17	LLAddr 注1
2	エントリ Lo0 注1	18	ウォッチ Lo 注2
3	エントリ Lo1 注1	19	ウォッチ Hi 注2
4	コンテキスト注2	20	X コンテキスト注2
5	ページ・マスク注1	21	RFU
6	ワイアード注1	22	RFU
7	RFU	23	RFU
8	BadVAddr 注1	24	RFU
9	カウント注2	25	RFU
10	エントリ Hi 注1	26	パリティ・エラー注2
11	比較注2	27	キャッシュ・エラー注2
12	ステータス注2	28	タグ Lo 注1
13	原因注2	29	タグ Hi 注1
14	EPC 注2	30	エラーEPC 注2
15	PRId 注1	31	RFU

注 1. メモリ管理に使用

2. 例外処理に使用

備考 RFU：Reserved for Future Use

表 2-1 CP0 レジスタ

レジスタ番号	レジスタ名	説　明
0	インデクス	TLB 配列へのプログラマブル・ポインタ
1	ランダム	TLB 配列への疑似ランダム・ポインタ（読み出し専用）
2	エントリ Lo0	偶数 VPN 用 TLB エントリの後半
3	エントリ Lo1	奇数 VPN 用 TLB エントリの後半
4	コンテキスト	32 ビット・モード時のカーネルの仮想 PTE テーブルへのポインタ
5	ページ・マスク	ページ・サイズの指定
6	ワイアード	ワイアード TLB エントリ数
7	—	RFU（Reserved for future use）
8	BadVAddr	最後にエラーを起こした仮想アドレス表示
9	カウント	タイマ・カウント
10	エントリ Hi	TLB エントリの前半（ASID を含む）
11	比較	タイマ比較値
12	ステータス	動作状況の設定
13	原因	最後に発生した例外の原因表示
14	EPC	例外プログラム・カウンタ
15	PRId	プロセッサ・リビジョン ID
16	コンフィグ	メモリ・システム・モードの設定
17	LLAddr	RFU（Reserved for future use）
18	ウォッチ Lo	メモリ参照トラップ・アドレスの下位ビット
19	ォッチ Hi	メモリ参照トラップ・アドレスの上位ビット
20	X コンテキスト	64 ビット・モード時のカーネルの仮想 PTE テーブルへのポインタ
21 - 25	—	RFU（Reserved for future use）
26	パリティ・エラー注	キャッシュのパリティ・ビット
27	キャッシュ・エラー注	キャッシュのエラーおよび状態レジスタ
28	タグ Lo	キャッシュ・タグ・レジスタ下位
29	タグ Hi	キャッシュ・タグ・レジスタ上位
30	エラーEPC	エラー例外プログラム・カウンタ
31	—	RFU（Reserved for future use）

注　VR4100™ との互換を保つために定義しています。μPD98502 では使用しません。

2.1.6 浮動小数点ユニット（FPU）

μPD98502 では浮動小数点ユニット（FPU）が用意されていません。FPU 命令が実行されると，コプロセッサ使用不可例外が発生します。必要に応じて，例外ハンドラ中のソフトウエアでエミュレートしてください。

保守／廃止

2.1.7 VR4120A コア・メモリ管理システム（MMU）

VR4120A コアは，4 G（ギガ）バイト（32 ビット）の物理アドレス空間を持っています。しかし，これほど大きな物理メモリ空間を搭載したシステムを開発することは非現実的です。そこで，CPU がメモリ空間を論理的に拡張し，大きな仮想アドレス空間に構成したアドレスを実際の物理メモリ・アドレスに変換します。

VR4120A コアは，次に示す 2 つのアドレシング・モードをサポートしています。

- 32 ビット・モード：仮想アドレス空間を，2 G バイトのユーザ・プロセス空間と 2 G バイトのカーネル空間の 2 つに分割します。
- 64 ビット・モード：仮想アドレス空間を，1 T（テラ）バイトのユーザ仮想アドレス空間に拡張します。

これらのアドレス空間の詳細は，**2.4 メモリ管理システム**を参照してください。

2.1.8 高速変換緩衝機構（TLB）

仮想メモリのマッピングは，高速変換緩衝機構（TLB）によって行われます。TLB は，仮想アドレスから物理アドレスへのアドレス変換を行います。TLB はフルアソシアティブ方式で，32 個のエントリを持ち，各エントリには連続する 2 ページの情報がマッピングされます。ページ・サイズは，1 K バイトから 256 K バイトまで 4 のべき乗ごとに可変です。

2.1.8.1 ジョイント TLB（JTLB）

この TLB は命令アドレスとデータ・アドレスの両方を保持します。

仮想アドレスから物理アドレスへ高速アドレス変換を行うために VR4120A コアは 64 個の仮想ページに対応する物理アドレスに変換するフルアソシアティブ方式の TLB を持ちます。また，TLB は 32 組の奇数／偶数エントリで構成され，仮想アドレスとアドレス空間識別子（ASID）を 4 G バイトの物理アドレス空間に割り付けます。

ページ・サイズは，1 エントリごとに，1 K バイトから 256 K バイトまでの間で構成できます。マッピングするページ・サイズが CP0 レジスタにロードされます。このページ・サイズは新しいエントリが登録されるときに TLB に格納されます。これにより OS は各用途に応じてメモリを割り付けられます。たとえば，フレーム・バッファを１個の TLB エントリを使用してメモリにマップできます。

仮想アドレスから物理アドレスへの変換は，プロセッサからの仮想アドレスと，TLB 内の物理アドレスの比較から始まります。アドレスの仮想ページ番号（VPN）とエントリの VPN フィールドが同じ場合，または TLB エントリのグローバル・ビット（G）がセットされている場合，セットされていないときは仮想アドレスの ASID フィールドと TLB エントリの ASID フィールドが同じ場合，一致が起こります。

この一致を TLB ヒットといいます。もし一致しなければ，プロセッサは TLB ミス例外を発生させ，メモリの仮想／物理アドレス・ページ・テーブルから TLB へのリフィルをソフトウエアで行います。

保守／廃止

2.1.9 動作モード

VR4120A コアには，次の 3 つの動作モードがあります。

- ユーザ・モード
- スーパバイザ・モード
- カーネル・モード

この動作モードによって，メモリのアドレス変換（マッピング）方式が異なります。詳細は，**2.4 メモリ管理システム**を参照してください。

2.1.10 キャッシュ

VR4120A コアは，パイプラインの効率を高めるために，それぞれ独立した命令キャッシュとデータ・キャッシュを内蔵しています。どちらのキャッシュも 64 ビットのデータ・バスを持っており，1 クロックでアクセスできます。また命令キャッシュとデータ・キャッシュは並列にアクセスできます。VR4120A コアの命令キャッシュは 16 K バイト，データ・キャッシュは 8 K バイトの容量を持っています。

キャッシュの詳細については，**2.7 キャッシュ・メモリ**を参照してください。

2.1.11 命令パイプライン

VR4120A コアは 6 段の命令パイプラインを持っています。通常の環境であれば，1 サイクルで 1 命令を実行します。詳細は，**2.3 パイプライン**を参照してください。

保守／廃止

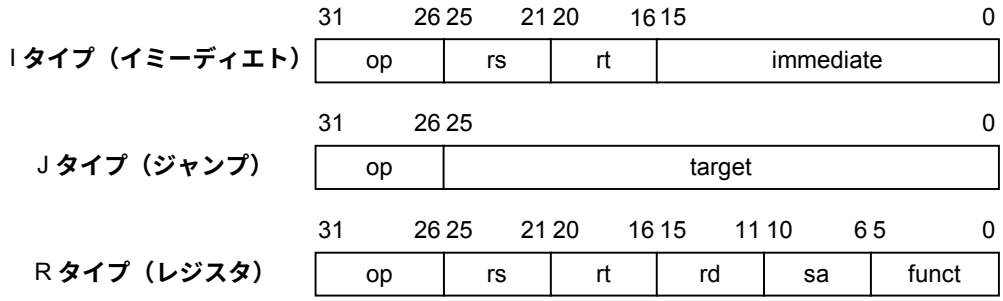
2.2 MIPS III 命令セット概要

この節では，MIPS III ISA の CPU 命令セットを概説します。VR4120A コアの各命令の詳細については，付録 A MIPS III 命令セットを参照してください。

2.2.1 MIPS III ISA 命令形式

MIPS III ISA CPU 命令の命令長はすべて 1 ワード（32 ビット）で，ワード境界に配置されています。命令形式は，図 2-8に示す 3 つの形式があります。命令形式を 3 つに単純化することによって命令のデコードが簡略化されています。複雑で使用頻度の低いオペレーションやアドレシング・モードは，コンパイラで 3 つの形式を組み合わせて実現しています。

図 2-8 MIPS III ISA CPU 命令形式



op	6 ビットの命令コード
rs	5 ビットのソース・レジスタ番号
rt	5 ビットのターゲット（ソース／デスティネーション）・レジスタ番号，または分岐条件
immediate	16 ビットのイミーディエト値，分岐ディスプレースメント，またはアドレス・ディスプレースメント
target	26 ビットの無条件分岐ターゲット・アドレス
rd	5 ビットのデスティネーション・レジスタ番号
sa	5 ビットのシフト量
funct	6 ビットの機能フィールド

2.2.1.1 MIPS ISA の対応

VR4120A コアは複数のプロセッサを同時に動作させる環境をサポートしていません。したがって，MIPS II および MIPS III ISA で定義されている同期サポート命令（ロード・リンク命令，ストア条件付き命令など）は，予約命令例外の原因になります。また，ロード・リンク・ビット（LL ビット）は削除されています。

注意　SYNC 命令は NOP 命令として処理するため，すべてのロード／ストア命令はプログラム順に実行します。

保守／廃止

2.2.2 命令クラス

CPU 命令は 5 つのクラスに分類できます。

2.2.2.1 ロード／ストア命令

ロード／ストア命令は，メモリと汎用レジスタの間でデータの転送を行います。これらの命令はすべて I-タイプです。ロード／ストア命令のアドレシング・モードは，ベース・レジスタに符号付き 16 ビット・イミーディエト・オフセットを加えるモードのみです。

(1) ロード遅延スロットのスケジュール

ロード結果を直後の命令が使用できないロード命令を，遅延付きロード命令といいます。また遅延付きロード命令の直後の命令スロットを，ロード遅延スロットといいます。VR4000 シリーズでは，ロード命令の直後にロード先のレジスタを含んだ命令を記述することは可能ですが，この場合は必要なサイクル分，インタロックを発生します。したがって，どのような命令記述も可能ですが，性能面と VR シリーズ ™ との互換性の面からも，ロード遅延スロットをスケジューリングすることをお勧めします（詳細については**2.3 パイプライン**を参照してください）。

(2) ストア遅延スロット

ストア命令がデータ・キャッシュに書き込みを行っている場合，DC ステージと WB ステージではそのデータ・キャッシュがビジィ状態になります。直後の命令（たとえばロード命令）が DC ステージでデータ・キャッシュにアクセスする必要がある場合，ハードウエアによるインタロックが発生します。したがって，ストア遅延スロットをスケジューリングすることをお勧めします。

表 2-2 ロード／ストア命令の遅延スロット・サイクル数

命令	必要サイクル数（PCycle）
ロード	1
ストア	1

(3) アクセス・タイプの定義

アクセス・タイプとは，プロセッサがロード／ストアするデータの大きさです。

ロード／ストア命令のオペコードは，アクセス・タイプを決定します。表 2-3に，アクセス・タイプとロード／ストアされるデータを示します。ロード／ストア命令で使用されるアドレスは，アクセス・タイプやバイト並び（エンディアン）にかかわらず，最下位のバイト・アドレス（リトル・エンディアンにおいては最下位バイトを指すアドレス）になります。

アクセスされるデータのダブル・ワード内のバイト並びは，表 2-3に示すようにアクセス・タイプとアドレスの下位 3 ビットによって決定します。また，表 2-3に示す組み合わせ以外のアクセス・タイプとアドレスの下位ビットとの組み合わせは禁止されています。これ以外の組み合わせの場合，アドレス・エラー例外が発生します。

表 2-4に ISA の定義するロード／ストア命令，表 2-5に拡張 ISA の命令の一覧を示します。

表 2-3 ロード／ストア命令に関するバイト指定

アクセス・タイプ（値）	下位アドレス・ビット			アクセスされるバイト（リトル・エンディアン）							
	2	1	0	63							0
ダブル・ワード（7）	0	0	0	7	6	5	4	3	2	1	0
7 バイト（6）	0	0	0		6	5	4	3	2	1	0
	0	0	1	7	6	5	4	3	2	1	
6 バイト（5）	0	0	0			5	4	3	2	1	0
	0	1	0	7	6	5	4	3	2		
5 バイト（4）	0	0	0				4	3	2	1	0
	0	1	1	7	6	5	4	3			
ワード（3）	0	0	0					3	2	1	0
	1	0	0	7	6	5	4				
トリプル・バイト（2）	0	0	0						2	1	0
	0	0	1					3	2	1	
	1	0	0		6	5	4				
	1	0	1	7	6	5					
ハーフ・ワード（1）	0	0	0							1	0
	0	1	0					3	2		
	1	0	0			5	4				
	1	1	0	7	6						
バイト（0）	0	0	0								0
	0	0	1							1	
	0	1	0						2		
	0	1	1					3			
	1	0	0				4				
	1	0	1			5					
	1	1	0		6						
	1	1	1	7							

表 2-4 ロード／ストア命令

命 令	形式と説明	op	base	rt	offset
Load Byte	LB rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 アドレス指定されたバイトの内容を符号拡張して，レジスタ rt にロードします。				
Load Byte Unsigned	LBU rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 アドレス指定されたバイトの内容をゼロ拡張して，レジスタ rt にロードします。				
Load Halfword	LH rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 アドレス指定されたハーフ・ワードの内容を符号拡張して，レジスタ rt にロードします。				
Load Halfword Unsigned	LHU rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 アドレス指定されたハーフ・ワードの内容をゼロ拡張して，レジスタ rt にロードします。				
Load Word	LW rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 アドレス指定されたワードの内容を（64 ビット・モード時符号拡張して）レジスタ rt にロードします。				
Load Word Left	LWL rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 アドレス指定されたバイトがワードの最左端になるようにアドレス指定されたワードを左にシフトします。シフトした結果と，レジスタ rt の内容とをマージし（64 ビット・モード時符号拡張して），レジスタ rt にロードします。				
Load Word Right	LWR rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 アドレス指定されたバイトがワードの最右端になるようにアドレス指定されたワードを右にシフトします。シフトした結果と，レジスタ rt の内容とをマージし（64 ビット・モード時符号拡張して），レジスタ rt にロードします。				
Store Byte	SB rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 レジスタ rt の最下位バイトの内容をアドレス指定されたメモリにストアします。				
Store Halfword	SH rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 レジスタ rt の最下位ハーフ・ワードの内容をアドレス指定されたメモリにストアします。				
Store Word	SW rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 レジスタ rt の下位ワードの内容をアドレス指定されたメモリにストアします。				
Store Word Left	SWL rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 ワードの最左端バイトがアドレス指定されたバイトの位置になるように，レジスタ rt の内容を右にシフトします。シフトした結果を，メモリ中のワードの下位部分にストアします。				
Store Word Right	SWR rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 ワードの最右端バイトがアドレス指定されたバイトの位置になるように，レジスタ rt の内容を左にシフトします。シフトした結果を，メモリ中のワードの上位部分にストアします。				

保守／廃止

表 2-5 ロード／ストア命令（拡張 ISA）

命 令	形式と説明	op	base	rt	offset
Load Doubleword	LD rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 アドレス指定されたダブル・ワードの内容をレジスタ rt にロードします。				
Load Doubleword Left	LDL rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 アドレス指定されたバイトがダブル・ワードの最左端になるようにアドレス指定されたダブル・ワードを左にシフトします。シフトした結果とレジスタ rt の内容とをマージし，レジスタ rt にロードします。				
Load Doubleword Right	LDR rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 アドレス指定されたバイトがダブル・ワードの最右端になるようにアドレス指定されたダブル・ワードを右にシフトします。シフトした結果とレジスタ rt の内容とをマージし，レジスタ rt にロードします。				
Load Word Unsigned	LWU rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 アドレス指定されたワードの内容をゼロ拡張して，レジスタ rt にロードします。				
Store Doubleword	SD rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 レジスタ rt の内容をアドレス指定されたメモリにストアします。				
Store Doubleword Left	SDL rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 ダブル・ワードの最左端バイトがアドレス指定されたバイトの位置になるように，レジスタ rt の内容を右にシフトします。シフトした結果を，メモリ中のダブル・ワードの下位部分にストアします。				
Store Doubleword Right	SDR rt，offset（base） 符号拡張した offset をレジスタ base の内容に加算しアドレスを生成します。 ダブル・ワードの最右端バイトがアドレス指定されたバイトの位置になるように，レジスタ rt の内容を左にシフトします。シフトした結果を，メモリ中のダブル・ワードの上位部分にストアします。				

保守／廃止

2.2.2.2 演算命令

演算命令には，レジスタ内の値に対する算術演算，乗除算，論理演算，シフト動作を実行します。これらの命令は，両方のソースがレジスタである R-タイプか，片方のソースがイミーディエトである I-タイプとなります。演算命令は次の 4 つに分類されます。

(1) ALU イミーディエイト命令
(2) 3 オペランド・レジスタ・タイプ命令
(3) シフト命令
(4) 乗除算命令

64 ビット・モードと 32 ビット・モードでデータの互換性が必要である場合，32 ビットのオペランドは，正しく符号拡張されている必要があります。正しく符号拡張されていない値を使用した演算結果の 32 ビットの値は意味を持ちません。

表 2-6 ALU イミーディエト命令

命　令	形式と説明	op	rs	rt	immediate
Add Immediate	ADDI　rt，rs，immediate 16 ビット・イミーディエトを符号拡張し，レジスタ rs と加算します。32 ビットの結果をレジスタ rt に（64 ビット・モード時符号拡張して）格納します。 2 の補数オーバフローが発生すると例外を発生します。				
Add Immediate Unsigned	ADDIU　rt，rs，immediate 16 ビット・イミーディエトを符号拡張し，レジスタ rs と加算します。32 ビットの結果をレジスタ rt に（64 ビット・モード時符号拡張して）格納します。オーバフローが発生しても例外を発生しません。				
Set On Less Than Immediate	SLTI　rt，rs，immediate 16 ビット・イミーディエトを符号拡張し，符号付き整数としてレジスタ rs と比較します。rs がイミーディエトより小さい場合は 1 を，そうでない場合は 0 をレジスタ rt に格納します。				
Set On Less Than Immediate Unsigned	SLTIU　rt，rs，immediate 16 ビット・イミーディエトを符号拡張し，符号なし整数としてレジスタ rs と比較します。rs がイミーディエトより小さい場合は 1 を，そうでない場合は 0 をレジスタ rt に格納します。				
And Immediate	ANDI　rt，rs，immediate 16 ビット・イミーディエトをゼロ拡張してレジスタ rs と AND をとり，結果をレジスタ rt に格納します。				
Or Immediate	ORI　rt，rs，immediate 16 ビット・イミーディエトをゼロ拡張してレジスタ rs と OR をとり，結果をレジスタ rt に格納します。				
Exclusive Or Immediate	XORI　rt，rs，immediate 16 ビット・イミーディエトをゼロ拡張してレジスタ rs と Ex - OR をとり，結果をレジスタ rt に格納します。				
Load Upper Immediate	LUI　rt，immediate 16 ビット・イミーディエトを 16 ビット左にシフトし，ワードの下位 16 ビットを 0 にします。結果を（64 ビット・モード時符号拡張して）レジスタ rt に格納します。				

表 2-7 ALU イミーディエト命令（拡張 ISA）

命 令	形式と説明	op	rs	rt	immediate
Doubleword Add Immediate	DADDI rt，rs，immediate 16 ビット・イミーディエトを 64 ビットに符号拡張し，レジスタ rs と加算します。64 ビットの結果をレジスタ rt に格納します。整数オーバフローが発生すると例外を発生します。				
Doubleword Add Immediate Unsigned	DADDIU rt，rs，immediate 16 ビット・イミーディエトを 64 ビットに符号拡張し，レジスタ rs と加算します。64 ビットの結果をレジスタ rt に格納します。オーバフローが発生しても例外を発生しません。				

表 2-8 3 オペランド・タイプ命令

命 令	形式と説明	op	rs	rt	rd	sa	funct
Add	ADD rd，rs，rt レジスタ rs と rt の内容を加算し，32 ビットの結果を（64 ビット・モード時符号拡張して）レジスタ rd に格納します。 整数オーバフローが発生すると例外を発生します。						
Add Unsigned	ADDU rd，rs，rt レジスタ rs と rt の内容を加算し，32 ビットの結果を（64 ビット・モード時符号拡張して）レジスタ rd に格納します。 整数オーバフローが発生しても例外を発生しません。						
Subtract	SUB rd，rs，rt レジスタ rs からレジスタ rt の内容を減算し，32 ビットの結果を（64 ビット・モード時符号拡張して）レジスタ rd に格納します。 整数オーバフローが発生すると例外を発生します。						
Subtract Unsigned	SUBU rd，rs，rt レジスタ rs からレジスタ rt の内容を減算し，32 ビットの結果を（64 ビット・モード時符号拡張して）レジスタ rd に格納します。 整数オーバフローが発生しても例外を発生しません。						
Set On Less Than	SLT rd，rs，rt レジスタ rs と rt の内容を符号付き整数として比較します。 レジスタ rs が rt より小さい場合は 1 を，そうでない場合は 0 をレジスタ rd に格納します。						
Set On Less Than Unsigned	SLTU rd，rs，rt レジスタ rs と rt の内容を符号なし整数として比較します。 レジスタ rs が rt より小さい場合は 1 を，そうでない場合は 0 をレジスタ rd に格納します。						
And	AND rd，rs，rt レジスタ rs と rt の内容をビット単位で AND をとり，結果をレジスタ rd に格納します。						
Or	OR rd，rs，rt レジスタ rs と rt の内容をビット単位で OR をとり，結果をレジスタ rd に格納します。						
Exclusive Or	XOR rd，rs，rt レジスタ rs と rt の内容をビット単位で Ex - OR をとり，結果をレジスタ rd に格納します。						
Nor	NOR rd，rs，rt レジスタ rs と rt の内容をビット単位で NOR をとり，結果をレジスタ rd に格納します。						

表 2-9 3 オペランド・タイプ命令（拡張 ISA）

命 令	形式と説明 op rs rt rd sa funct
Doubleword Add	DADD rd，rs，rt レジスタ rs と rt の内容を加算し，64 ビットの結果をレジスタ rd に格納します。 整数オーバフローが発生すると例外を発生します。
Doubleword Add Unsigned	DADDU rd，rs，rt レジスタ rs と rt の内容を加算し，64 ビットの結果をレジスタ rd に格納します。 整数オーバフローが発生しても例外を発生しません。
Doubleword Subtract	DSUB rd，rs，rt レジスタ rs からレジスタ rt の内容を減算し，64 ビットの結果をレジスタ rd に格納します。 整数オーバフローが発生すると例外を発生します。
Doubleword Subtract Unsigned	DSUBU rd，rs，rt レジスタ rs からレジスタ rt の内容を減算し，64 ビットの結果をレジスタ rd に格納します。 整数オーバフローが発生しても例外を発生しません。

表 2-10 シフト命令

命 令	形式と説明 op rs rt rd sa funct
Shift Left Logical	SLL rd，rt，sa レジスタ rt の内容を sa ビット左にシフトし，下位ビットに 0 を挿入します。 32 ビットの結果を（64 ビット・モード時符号拡張して）レジスタ rd に格納します。
Shift Right Logical	SRL rd，rt，sa レジスタ rt の内容を sa ビット右にシフトし，上位ビットに 0 を挿入します。 32 ビットの結果を（64 ビット・モード時符号拡張して）レジスタ rd に格納します。
Shift Right Arithmetic	SRA rd，rt，sa レジスタ rt の内容を sa ビット右にシフトし，上位ビットを符号拡張します。 32 ビットの結果を（64 ビット・モード時符号拡張して）レジスタ rd に格納します。
Shift Left Logical Variable	SLLV rd，rt，rs レジスタ rt の内容を左にシフトし，下位ビットに 0 を挿入します。 シフトするビット数はレジスタ rs の下位 5 ビットで指定します。 32 ビットの結果を（64 ビット・モード時符号拡張して）レジスタ rd に格納します。
Shift Right Logical Variable	SRLV rd，rt，rs レジスタ rt の内容を右にシフトし，上位ビットに 0 を挿入します。 シフトするビット数はレジスタ rs の下位 5 ビットで指定します。 32 ビットの結果を（64 ビット・モード時符号拡張して）レジスタ rd に格納します。
Shift Right Arithmetic Variable	SRAV rd，rt，rs レジスタ rt の内容を右にシフトし，上位ビットを符号拡張します。 シフトするビット数はレジスタ rs の下位 5 ビットで指定します。 32 ビットの結果を（64 ビット・モード時符号拡張して）レジスタ rd に格納します。

保守／廃止

表 2-11 シフト命令（拡張 ISA）

命 令	形式と説明	op	rs	rt	rd	sa	funct
Doubleword Shift Left Logical	DSLL rd，rt，sa **レジスタ** rt **の内容を** sa **ビット左にシフトし，下位ビットに** 0 **を挿入します。** 64 **ビットの結果をレジスタ** rd **に格納します。**						
Doubleword Shift Right Logical	DSRL rd，rt，sa **レジスタ** rt **の内容を** sa **ビット右にシフトし，上位ビットに** 0 **を挿入します。** 64 **ビットの結果をレジスタ** rd **に格納します。**						
Doubleword Shift Right Arithmetic	DSRA rd，rt，sa **レジスタ** rt **の内容を** sa **ビット右にシフトし，上位ビットを符号拡張します。** 64 **ビットの結果をレジスタ** rd **に格納します。**						
Doubleword Shift Left Logical Variable	DSLLV rd，rt，rs **レジスタ** rt **の内容を左にシフトし，下位ビットに** 0 **を挿入します。** **シフトするビット数はレジスタ** rs **の下位** 6 **ビットで指定します。** 64 **ビットの結果をレジスタ** rd **に格納します。**						
Doubleword Shift RIght Logical Variable	DSRLV rd，rt，rs **レジスタ** rt **の内容を右にシフトし，上位ビットに** 0 **を挿入します。** **シフトするビット数はレジスタ** rs **の下位** 6 **ビットで指定します。** 64 **ビットの結果をレジスタ** rd **に格納します。**						
Doubleword Shift Right Arithmetic Variable	DSRAV rd，rt，rs **レジスタ** rt **の内容を右にシフトし，上位ビットを符号拡張します。** **シフトするビット数はレジスタ** rs **の下位** 6 **ビットで指定します。** 64 **ビットの結果をレジスタ** rd **に格納します。**						
Doubleword Shift Left Logical＋32	DSLL32 rd，rt，sa **レジスタ** rt **の内容を** 32＋sa **ビット左にシフトし，下位ビットに** 0 **を挿入します。** 64 **ビットの結果をレジスタ** rd **に格納します。**						
Doubleword Shift Right Logical＋32	DSRL32 rd，rt，sa **レジスタ** rt **の内容を** 32＋sa **ビット右にシフトし，上位ビットに** 0 **を挿入します。** 64 **ビットの結果をレジスタ** rd **に格納します。**						
Doubleword Shift Right Arithmetic＋32	DSRA32 rd，rt，sa **レジスタ** rt **の内容を** 32＋sa **ビット右にシフトし，上位ビットを符号拡張します。** 64 **ビットの結果をレジスタ** rd **に格納します。**						

表 2-12 乗除算命令

命 令	形式と説明	op	rs	rt	rd	sa	funct
Multiply	MULT　rs，rt **レジスタ** rs **と** rt **の内容を** 32 **ビット符号付き整数として乗算します。**64 **ビットの結果を，特殊レジスタ** HI **と** LO **に（**64 **ビット・モード時符号拡張して）格納します。**						
Multiply Unsigned	MULTU　rs，rt **レジスタ** rs **と** rt **の内容を** 32 **ビット符号なし整数として乗算します。**64 **ビットの結果を，特殊レジスタ** HI **と** LO **に（**64 **ビット・モード時符号拡張して）格納します。**						
Divide	DIV　rs，rt **レジスタ** rs **をレジスタ** rt **の内容で除算します。オペランドは** 32 **ビット符号付き整数として扱います。**32 **ビットの商を特殊レジスタ** LO **に，**32 **ビットの剰余を特殊レジスタ** HI **に（**64 **ビット・モード時符号拡張して）格納します。**						
Divide Unsigned	DIVU　rs，rt **レジスタ** rs **をレジスタ** rt **の内容で除算します。オペランドは** 32 **ビット符号なし整数として扱います。**32 **ビットの商を特殊レジスタ** LO **に，**32 **ビットの剰余を特殊レジスタ** HI **に（**64 **ビット・モード時符号拡張して）格納します。**						
Move From HI	MFHI　rd **特殊レジスタ** HI **の内容をレジスタ** rd **に転送します。**						
Move From LO	MFLO　rd **特殊レジスタ** LO **の内容をレジスタ** rd **に転送します。**						
Move To HI	MTHI　rs **レジスタ** rs **の内容を特殊レジスタ** HI **に転送します。**						
Move To LO	MTLO　rs **レジスタ** rs **の内容を特殊レジスタ** LO **に転送します。**						

表 2-13 乗除算命令（拡張 ISA）

命 令	形式と説明	op	rs	rt	rd	sa	funct
Doubleword Multiply	DMULT　rs，rt レジスタ rs と rt の内容を符号付き整数として乗算します。 128 ビットの結果を特殊レジスタ HI と LO に格納します。						
Doubleword Multiply Unsigned	DMULTU　rs，rt レジスタ rs と rt の内容を符号なし整数として乗算します。 128 ビットの結果を特殊レジスタ HI と LO に格納します。						
Doubleword Divide	DDIV　rs，rt レジスタ rs をレジスタ rt の内容で除算します。 オペランドは符号付き整数として扱います。 64 ビットの商を特殊レジスタ LO に，64 ビットの剰余を特殊レジスタ HI に格納します。						
Doubleword Divide Unsigned	DDIVU　rs，rt レジスタ rs をレジスタ rt の内容で除算します。 オペランドは符号なし整数として扱います。 64 ビットの商を特殊レジスタ LO に，64 ビットの剰余を特殊レジスタ HI に格納します。						
Multiply and Add Accumulate	MACC{h}{u}{s}　rd，rs，rt レジスタ rs と rt の内容を 32 ビット符号付き整数として乗算し，結果を特殊レジスタ HI と LO をつなげた値と加算します。64 ビットの結果を特殊レジスタ HI と LO に格納します。 h = 0 の場合レジスタ LO に格納されたものと同じデータが，h = 1 の場合レジスタ HI に格納されたものと同じデータが，レジスタ rd にも格納されます。 u を指定した場合，オペランドは符号なしデータとして扱われます。 s を指定した場合，レジスタ rs と rd は 16 ビット値として（32 ビットに符号拡張または 0 拡張して），またレジスタ HI と LO をつなげた値は 32 ビット値として（64 ビットに符号拡張または 0 拡張して）扱われます。また，演算結果に対し u で指定したフォーマットで飽和処理が行われます。						
Doubleword Multiply and Add Accumulate	DMACC{u}{s}　rd，rs，rt レジスタ rs と rt の内容を 32 ビット符号付き整数として乗算し，結果を特殊レジスタ LO の値と加算します。64 ビットの結果を特殊レジスタ LO に格納します。 レジスタ LO に格納されたものと同じデータがレジスタ rd に格納されます。 u を指定した場合，オペランドは符号なしデータとして扱われます。 s を指定した場合，レジスタ rs と rd は 16 ビット値として（32 ビットに符号拡張または 0 拡張して），またレジスタ LO は 32 ビット値として（64 ビットに符号拡張または 0 拡張して）扱われます。また，演算結果に対し u で指定したフォーマットで飽和処理が行われます。						

乗除算命令のあとに MFHI 命令や MFLO 命令を実行すると，インタロックを発生して次の命令を遅らせ，乗除算命令が終了するまで結果の読み出しを禁止します。

表 2-14に乗除算命令とその後の MFHI，MFLO 命令との間のインタロックやストールを解決するのに必要なプロセッサ・サイクル（PCycle）数を示します。

保守／廃止

表 2-14 乗除算命令のストール・サイクル数

命 令	必要サイクル数
MULT	1
MULTU	1
DIV	36
DIVU	36
DMULT	3
DMULTU	3
DDIV	68
DDIVU	68
MACC	0
DMACC	0

2.2.2.3 ジャンプ／ブランチ命令

ジャンプ命令とブランチ命令はプログラムの流れを変更します。すべてのジャンプ命令とブランチ命令は 1 遅延スロットを生じます。ジャンプかブランチ命令の直後の命令（遅延スロット内の命令）は，飛び先の先頭の命令をメモリからフェッチする間に実行します。

JAL 命令や BLTZAL 命令などリンクを伴う命令では，復帰アドレスをレジスタ r31 に保存します。

表 2-15 ジャンプ／ブランチ命令の遅延スロット・サイクル数

命令	必要サイクル数
ブランチ	1
ジャンプ	1

(1) ジャンプ命令の概要

高級言語で記述したサブルーチン・コールは，通常，J 命令か JAL 命令を使用します。J 命令，JAL 命令は J-タイプです。この形式は，26 ビットのターゲット・アドレスを 2 ビット左にシフトし，現在のプログラム・カウンタの上位 4 ビットと組み合わせて 32 ビットまたは 64 ビットの絶対アドレスを生成します。

復帰，ディスパッチ，ページ間のジャンプは，通常，JR 命令か JALR 命令を使用します。どちらも R-タイプで，汎用レジスタの 32 ビットまたは 64 ビットのバイト・アドレスを参照します。

詳細は**付録 A　MIPS III 命令セット**を参照してください。

(2) ブランチ命令の概要

ブランチ命令は，プログラム・カウンタ相対の符号付き 16 ビット・オフセットを持っています。

表 2-16にジャンプ命令，表 2-17にブランチ命令，表 2-18に拡張 ISA のブランチ命令の一覧を示します。

表 2-16 ジャンプ命令

命 令	形式と説明	op	target
Jump	JAL target 26 ビットのターゲット・アドレスを左に 2 ビット分シフトし，PC の上位 4 ビットと結合したアドレスへ，1 命令遅れてジャンプします。		
Jump And Link	J target 26 ビットのターゲット・アドレスを左に 2 ビット分シフトし，PC の上位 4 ビットと結合したアドレスへ，1 命令遅れてジャンプします。遅延スロットに続く命令のアドレスを r31（リンク・レジスタ）に格納します。		

命 令	形式と説明	op	target
Jump And Link Exchange	JALX target 26 ビットのターゲット・アドレスを左に 2 ビット分シフトし，PC の上位 4 ビットと結合したアドレスへ 1 命令遅れてジャンプし，ISA モード・ビットを反転します。遅延スロットに続く命令のアドレスを r31（リンク・レジスタ）に格納します。		

命 令	形式と説明	op	rs	rt	rd	sa	funct
Jump Register	JR rs レジスタ rs のアドレスへ 1 命令遅れてジャンプします。						
Jump And Link Register	JALR rs，rd レジスタ rs のアドレスへ 1 命令遅れてジャンプします。 遅延スロットに続く命令のアドレスをレジスタ rd に格納します。						

表 2-17，表 2-18では，次に示す共通の制限があります。

・ブランチ・アドレス

すべてのブランチ命令によるブランチ・アドレスは，遅延スロットの命令のアドレスに 16 ビットのオフセット（2 ビット左にシフトした符号付き 64 ビット）を加算して算出されます。すべてのブランチ命令は 1 遅延スロットを生じます。

・非分岐時の動作

ブランチ likely 命令で分岐条件が成立しなかった場合は，遅延スロット内の命令は無効となります。ほかのすべてのブランチ命令については，遅延スロット内の命令が無条件に実行されます。

備考 ブランチ先の対象となる命令は，ブランチ命令の EX ステージでフェッチされます。ブランチの対象とターゲット・アドレスの計算は，ブランチ命令の RF ステージのフェーズ 2 と EX ステージのフェーズ 1 で行われます。アーキテクチャで定義されている 1 サイクルのブランチ遅延スロットが必要です。ジャンプ命令も同様に 1 サイクルの遅延スロットが必要です。ブランチ likely 命令で分岐条件が成立しなかった場合，遅延スロット内の命令は無効になります。

表 2-17から表 2-21までの命令形式の記号のうち，特殊なものを次に示します。

REGIMM ：オペコード
sub ：サブオペレーション・コード
CO ：サブオペレーション識別子
BC ：BC サブオペレーション・コード
br ：分岐条件識別子
op ：オペレーション・コード

表 2-17 ブランチ命令

命 令	形式と説明	op	rs	rt	offset
Branch On Equal	BEQ rs，rt，offset レジスタ rs と rt が等しかった場合，分岐アドレスへ分岐します。				
Branch On Not Equal	BNE rs，rt，offset レジスタ rs と rt が等しくない場合，分岐アドレスへ分岐します。				
Branch On Less Than Or Equal To Zero	BLEZ rs，offset レジスタ rs が 0 以下の場合，分岐アドレスへ分岐します。				
Branch On Greater Than Zero	BGTZ rs，offset レジスタ rs が 0 より大きい場合，分岐アドレスへ分岐します。				

命 令	形式と説明	REGIMM	rs	sub	offset
Branch On Less Than Zero	BLTZ rs，offset レジスタ rs が 0 より小さい場合，分岐アドレスへ分岐します。				
Branch On Greater Than Or Equal To Zero	BGEZ rs，offset レジスタ rs が 0 以上の場合，分岐アドレスへ分岐します。				
Branch On Less Than Zero And Link	BLTZAL rs，offset 遅延スロットに続く命令のアドレスをレジスタ r31（リンク・レジスタ）へ格納し，レジスタ rs が 0 より小さい場合，分岐アドレスへ分岐します。				
Branch On Greater Than Or Equal To Zero And Link	BGEZAL rs，offset 遅延スロットに続く命令のアドレスをレジスタ r31（リンク・レジスタ）へ格納し，レジスタ rs が 0 以上の場合，分岐アドレスへ分岐します。				

命 令	形式と説明	COP0	BC	br	offset
Branch On Coprocessor 0 True	BC0T offset 遅延スロットの命令のアドレスに，16 ビットの offset を 2 ビット左にシフトし，32 ビットに符号拡張したものを加え，分岐アドレスを算出します。 コプロセッサ 0 の条件信号が真なら，分岐アドレスへ 1 命令遅れで分岐します。				
Branch On Coprocessor 0 False	BC0F offset 遅延スロットの命令のアドレスに，16 ビットの offset を 2 ビット左にシフトし，32 ビットに符号拡張したものを加え，分岐アドレスを算出します。 コプロセッサ 0 の条件信号が偽なら，分岐アドレスへ 1 命令遅れで分岐します。				

保守／廃止

表 2-18 ブランチ命令（拡張 ISA）

命 令	形式と説明	op	rs	rt	offset
Branch On Equal Likely	BEQL rs，rt，offset **レジスタ** rs **と** rt **が等しい場合，分岐アドレスへ分岐します。分岐条件が成立しなかった場合，分岐遅延スロット内の命令は破棄されます。**				
Branch On Not Equal Likely	BNEL rs，rt，offset **レジスタ** rs **と** rt **が等しくない場合，分岐アドレスへ分岐します。分岐条件が成立しなかった場合，分岐遅延スロット内の命令は破棄されます。**				
Branch On Less Than Or Equal To Zero Likely	BLEZL rs，offset **レジスタ** rs **が** 0 **以下の場合，分岐アドレスへ分岐します。分岐条件が成立しなかった場合，分岐遅延スロット内の命令は破棄されます。**				
Branch On Greater Than Zero Likely	BGTZL rs，offset **レジスタ** rs **が** 0 **より大きい場合，分岐アドレスへ分岐します。分岐条件が成立しなかった場合，分岐遅延スロット内の命令は破棄されます。**				

命 令	形式と説明	REGIMM	rs	sub	offset
Branch On Less Than Zero Likely	BLTZL rs，offset **レジスタ** rs **が** 0 **より小さい場合，分岐アドレスへ分岐します。分岐条件が成立しなかった場合，分岐遅延スロット内の命令は破棄されます。**				
Branch On Greater Than Or Equal To Zero Likely	BGEZL rs，offset **レジスタ** rs **が** 0 **以上の場合，分岐アドレスへ分岐します。分岐条件が成立しなかった場合，分岐遅延スロット内の命令は破棄されます。**				
Branch On Less Than Zero And Link Likely	BLTZALL rs，offset **遅延スロットに続く命令のアドレスをレジスタ** r31**（リンク・レジスタ）に格納します。** **レジスタ** rs **が** 0 **より小さい場合，分岐アドレスへ分岐します。分岐条件が成立しなかった場合，分岐遅延スロット内の命令は破棄されます。**				
Branch On Greater Than Or Equal To Zero And Link Likely	BGEZALL rs，offset **遅延スロットに続く命令のアドレスをレジスタ** r31**（リンク・レジスタ）に格納します。** **レジスタ** rs **が** 0 **以上の場合，分岐アドレスへ分岐します。分岐条件が成立しなかった場合，分岐遅延スロット内の命令は破棄されます。**				

命 令	形式と説明	COP0	BC	br	offset
Branch On Coprocessor 0 True Likely	BC0TL offset **遅延スロットの命令のアドレスに，**16 **ビットの** offset **を** 2 **ビット左にシフトし，符号拡張したものを加え，分岐アドレスを算出します。** **コプロセッサ** 0 **の条件信号が真なら，分岐アドレスへ** 1 **命令遅れで分岐します。** **分岐条件が成立しなかった場合，分岐遅延スロット内の命令は破棄されます。**				
Branch On Coprocessor 0 False Likely	BC0FL offset **遅延スロットの命令のアドレスに，**16 **ビットの** offset **を** 2 **ビット左にシフトし，符号拡張したものを加え，分岐アドレスを算出します。** **コプロセッサ** 0 **の条件信号が偽なら，分岐アドレスへ** 1 **命令遅れで分岐します。** **分岐条件が成立しなかった場合，分岐遅延スロット内の命令は破棄されます。**				

保守／廃止

2.2.2.4 特殊命令

特殊命令は，ソフトウエアによる例外を発生させます。命令形式は R-タイプです（Syscall，Break）。トラップ命令は VR4000 シリーズだけに有効です。その他の命令は，すべての VR シリーズに有効です。

表 2-19 特殊命令

命 令	形式と説明	SPECIAL	rs	rt	rd	sa	funct
Synchronize	SYNC 現在パイプライン内にあるロード／ストア命令を，新たなロード／ストア命令の実行が開始される前に完結させます。						
System Call	SYSCALL システム・コール例外を発生し，例外処理プログラムに制御を移します。						
Breakpoint	BREAK ブレークポイント例外を発生し，例外処理プログラムに制御を移します。						

表 2-20 特殊命令（拡張 ISA）（1/2）

命 令	形式と説明	SPECIAL	rs	rt	rd	sa	funct
Trap If Greater Than Or Equal	TGE　rs，rt レジスタ rs と rt を符号付き整数として比較し，レジスタ rs が rt 以上の場合，例外を発生します。						
Trap If Greater Than Or Equal Unsigned	TGEU　rs，rt レジスタ rs と rt を符号なし整数として比較し，レジスタ rs が rt 以上の場合，例外を発生します。						
Trap If Less Than	TLT　rs，rt レジスタ rs と rt を符号付き整数として比較し，レジスタ rs が rt より小さい場合，例外を発生します。						
Trap If Less Than Unsigned	TLTU　rs，rt レジスタ rs と rt を符号なし整数として比較し，レジスタ rs が rt より小さい場合，例外を発生します。						
Trap If Equal	TEQ　rs，rt レジスタ rs と rt が等しいとき，例外を発生します。						
Trap If Not Equal	TNE　rs，rt レジスタ rs と rt が等しくないとき，例外を発生します。						

表 2-20 特殊命令（拡張 ISA）（2/2）

命 令	形式と説明	REGIMM	rs	sub	immediate
Trap If Greater Than Or Equal Immediate	TGEI rs，immediate レジスタ rs の内容と，16 ビット符号拡張したイミーディエトを符号付き整数として比較し，rs の内容がイミーディエト以上のとき，例外を発生します。				
Trap If Greater Than Or Equal Immediate Unsigned	TGEIU rs，immediate レジスタ rs の内容と，16 ビット 0 拡張したイミーディエトを符号なし整数として比較し，rs の内容がイミーディエト以上のとき，例外を発生します。				
Trap If Less Than Immediate	TLTI rs，immediate レジスタ rs の内容と，16 ビット符号拡張したイミーディエトを符号付き整数として比較し，rs の内容がイミーディエトより小さいとき，例外を発生します。				
Trap If Less Than Immediate Unsigned	TLTIU rs，immediate レジスタ rs の内容と，16 ビット 0 拡張したイミーディエトを符号なし整数として比較し，rs の内容がイミーディエトより小さいとき，例外を発生します。				
Trap If Equal Immediate	TEQI rs，immediate レジスタ rs の内容がイミーディエトと等しいとき，例外を発生します。				
Trap If Not Equal Immediate	TNEI rs，immediate レジスタ rs の内容がイミーディエトと等しくないとき，例外を発生します。				

2.2.2.5 システム制御コプロセッサ（CP0）命令

システム制御コプロセッサ（CP0）命令は，プロセッサのメモリ管理および例外処理を行うために，CP0 レジスタに対するオペレーションを実行します。

保守／廃止

表 2-21 システム制御コプロセッサ（CP0）命令

命 令	形式と説明	COP0	sub	rt	rd	0
Move To System Control Coprocessor	MTC0 rt，rd CPU の汎用レジスタ rt のワードの内容を，CP0 の汎用レジスタ rd にロードします。					
Move From System Control Coprocessor	MFC0 rt，rd CP0 の汎用レジスタ rd のワードの内容を，CPU の汎用レジスタ rt にロードします。					
Doubleword Move To Coprocessor 0	DMTC0 rt，rd CPU の汎用レジスタ rt のダブル・ワードの内容を，CP0 の汎用レジスタ rd にロードします。					
Doubleword Move From Coprocessor 0	DMFC0 rt，rd CP0 の汎用レジスタ rd のダブル・ワードの内容を，CPU の汎用レジスタ rt にロードします。					

命 令	形式と説明	COP0	CO	funct
Read Indexed TLB Entry	TLBR エントリ Hi，エントリ Lo0，エントリ Lo1，ページ・マスク・レジスタに，インデクス・レジスタにより指示されている TLB エントリをロードします。			
Write Indexed TLB Entry	TLBWI インデクス・レジスタにより指示されている TLB エントリに，エントリ Hi，エントリ Lo0，エントリ Lo1，ページ・マスク・レジスタの内容をロードします。			
Write Random TLB Entry	TLBWR ランダム・レジスタにより指示されている TLB エントリに，エントリ Hi，エントリ Lo0，エントリ Lo1，ページ・マスク・レジスタの内容をロードします。			
Probe TLB For Matching Entry	TLBP インデクス・レジスタに，エントリ Hi レジスタの内容と一致する TLB エントリのアドレスをロードします。			
Return From Exception	ERET 例外，割り込み，エラー・トラップから復帰します。			

命 令	形式と説明	COP0	CO	funct
STANDBY	STANDBY プロセッサを Fullspeed モードから Standby モードへ移行します。			
SUSPEND	SUSPEND プロセッサを Fullspeed モードから Suspend モードへ移行します。			
HIBERNATE	HIBERNATE プロセッサを Fullspeed モードから Hibernate モードへ移行します。			

命 令	形式と説明	CACHE	base	op	offset
Cache Operation	Cache op，offset（base） 16 ビットの offset を 32 ビットに符号拡張してレジスタ base と加算し，仮想アドレスを生成します。仮想アドレスを TLB を用いて物理アドレスに変換し，そのアドレスに対し 5 ビットのサブオペコードで示されるキャッシュ・オペレーションを行います。				

保守／廃止

2.3 パイプライン

この節では VR4120A コアのパイプラインの基本動作について説明します。

また，遅延スロット（ブランチ，ロード命令），パイプラインのインタロックと例外，CP0 ハザードについて説明します。

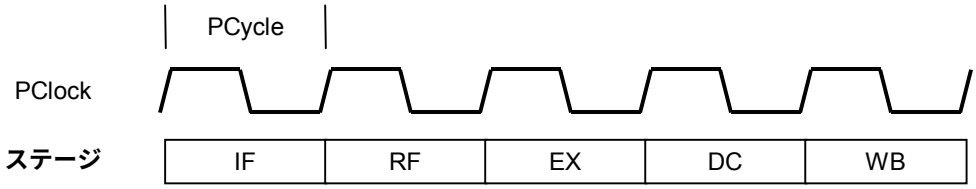
2.3.1 Pipeline stages

パイプラインは PClock（パイプライン・クロック・サイクル）で制御されます。また，この PClock の1サイクルを PCycle といいます。パイプラインの各ステージは 1 PCycle で実行されます（PClock 周波数は VR4120A の動作周波数と同じです）。

2.3.1.1 MIPS III（32 ビット長）命令モード時のパイプライン

VR4120A コアは，MIPS III 命令モード時，5 段パイプラインを使用します。したがって，各命令の実行には少なくとも 5 PCycle かかります。必要なデータがキャッシュになく，メイン・メモリから取り出さなければならない場合などは，それ以上のサイクルが必要となります。パイプラインがスムーズに流れる場合，5 つの命令が同時に実行されていることになります。

図 2-9 パイプライン・ステージ（MIPS III 命令モード）

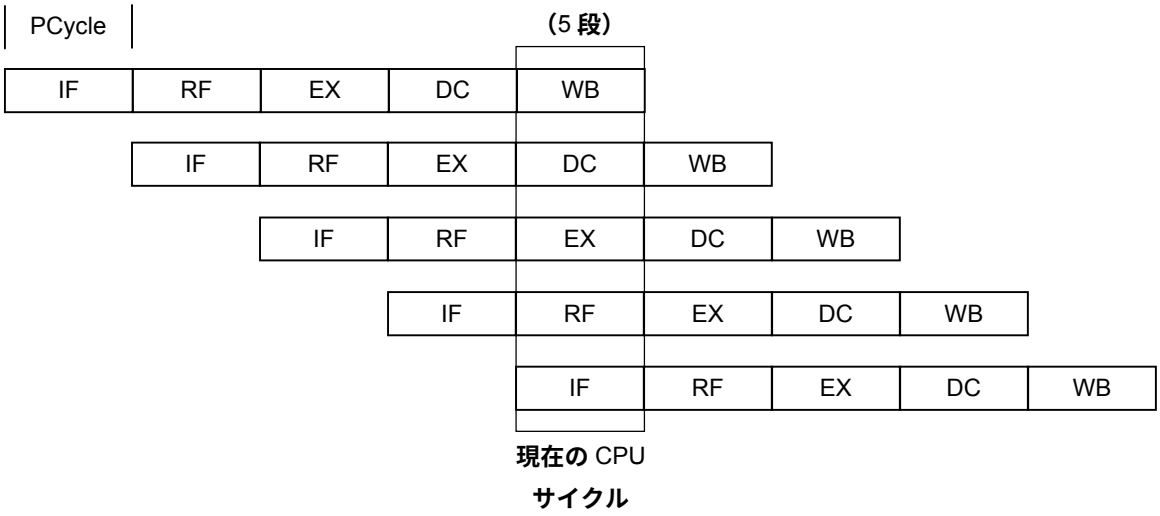


パイプラインの各ステージの名称と意味は次のとおりです。

- IF … 命令キャッシュのフェッチ
- RF … レジスタ・フェッチ
- EX … 実行
- DC … データ・キャッシュのフェッチ
- WB … ライトバック

図 2-10にパイプラインの概要を示します。この図の横の列は各命令の実行プロセス，縦の列は同時に実行している 5 つのプロセスを示します。

図 2-10 パイプライン中の命令実行（MIPS III 命令モード）



2.3.1.2 パイプラインの動作

図 2-11に MIPS III 命令モードの各パイプライン・ステージにおける動作，表 2-22に各部の名称を示します。

図 2-11 パイプラインの動作（MIPS III 命令モード）

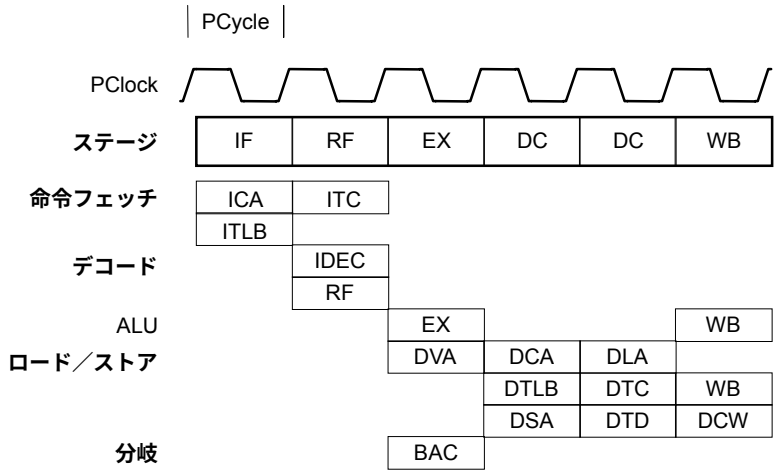


表 2-22 パイプラインにおける各ステージの動作（MIPS III 命令モード）

サイクル	ニモニック	説　明
IF	ITLB	命令アドレス変換
	ICA	命令キャッシュ・アレイのアクセス
RF	IDEC	命令デコード
	RF	レジスタ・オペランドのフェッチ
	ITC	命令タグ・チェック
EX	EX	実行ステージ
	BAC	分岐アドレスの計算
	DVA	データ仮想アドレスの計算
DC	DLA	データ・キャッシュ・ロード・アライン
	DSA	ストア・アライン
	DCA	データ・キャッシュ・アドレスのデコード／アレイのアクセス
	DTLB	データ・アドレス変換
	DTC	データ・タグ・チェック
	DTD	データ・キャッシュへのデータ転送
WB	DCW	データ・キャッシュへのライト
	WB	レジスタ・ファイルへのライトバック

保守／廃止

2.3.2 分岐遅延

VR4120A コアのパイプラインでは，次のとき分岐遅延を生じます。

- ジャンプ命令でターゲット・アドレスを計算したとき
- ブランチ命令で分岐条件が成立し，分岐先比較のための論理動作をしたとき

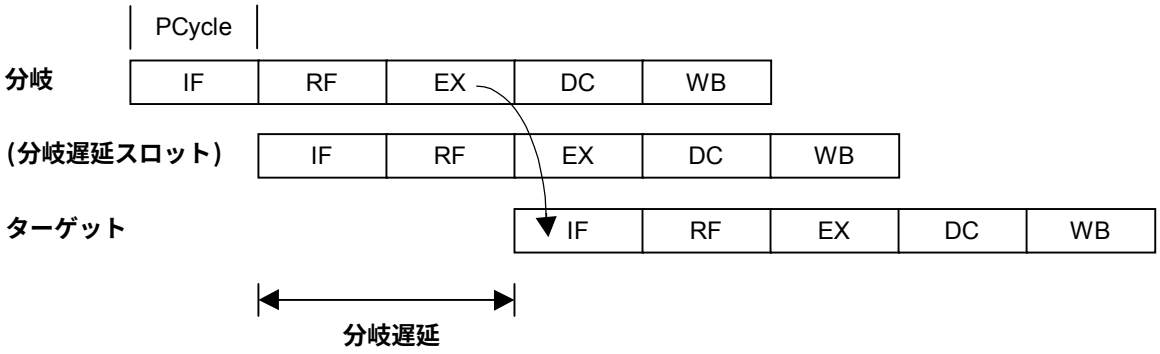
ジャンプ／ブランチ命令の次の命令の位置を分岐遅延スロットと呼びます。

ジャンプ／ブランチ命令の EX ステージで生成された命令アドレスは，次の次に実行される命令の IF ステージまで使用できません。

MIPSIII 命令モード時は，分岐遅延は 2 サイクルです。分岐遅延スロット内の 1 命令は，likely 命令を除き，実行されます。

MIPS III 命令モード時の分岐遅延の様子と分岐遅延スロットの位置について，図 2-12に示します。

図 2-12 分岐遅延（MIPS III 命令モード時）



2.3.3 ロード遅延

ロード命令では，データ・キャッシュからの読み出しとデータのアライン動作のため，DC ステージが 2 サイクル必要となります。この場合，ハードウエアが自動的にインタロックを発生させます。

ロード命令では，ロードした結果をその直後の命令で使用することはできません。この命令を遅延付きロード命令と呼びます。また，この遅延付きロード命令の直後の命令をロード遅延スロットといいます。

VR4120A コアでは，ロード命令の直後の命令がロードしたレジスタの内容を使用できますが，その場合でもハードウエアがインタロックを発生させ，遅延サイクルを追加挿入します。したがって，性能面と VR シリーズのプロセッサ能力の点から見て，ロード遅延スロットのスケジューリングを行うことをお勧めします。

2.3.4 パイプライン動作

パイプラインの動作を，例を用いて説明します。この例では，代表的な命令がどのように実行されているかを示します。例として取り上げた命令は，ADD，JALR，BEQ，TLT，LW，SW の 6 つです。パイプラインに取り込まれる命令別に関連する各ステージで行われる動作を記述しています。

2.3.4.1 Add 命令（ADD rd, rs, rt）

IF ステージ　仮想アドレスの下位 11 ビットを用いて命令キャッシュにアクセスし，キャッシュ・データを読み出します。仮想プログラム・カウンタを 4 つインクリメントし，次の命令をフェッチします。

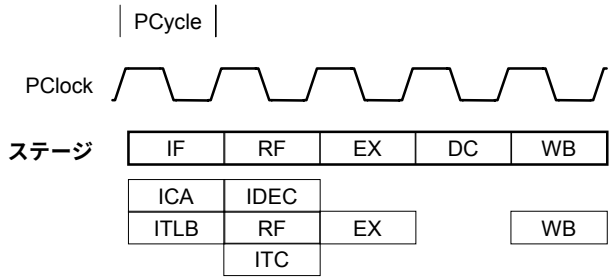
RF ステージ　2 ポートのレジスタ・ファイルの rs 領域と rt 領域をアクセスし，データをレジスタ・ファイルから読み出します。同時に，バイパス・マルチプレクサは，EX ステージの出力，DC ステージの出力，およびレジスタ・ファイルの出力から，オペランド・バイパスの必要性に応じて入力を選択します。

EX ステージ　ALU で演算を行います。

DC ステージ　このステージでは，命令に関係する動作を何も実行しません。EX ステージの出力（ALU）からのデータが，DC ステージの出力ラッチに移動します。

WB ステージ　データをレジスタ・ファイルに書き込みます。

図 2-13 ADD 命令のパイプライン動作（MIPS III 命令モード時）



2.3.4.2 Jump and Link Register 命令（JALR rd,rs）

IF ステージ　ADD 命令の IF ステージと同じ動作です。

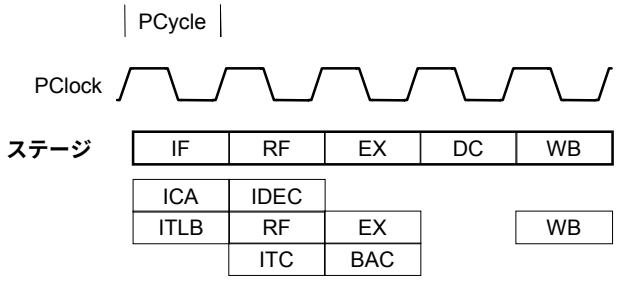
RF ステージ　rs 領域で指定したレジスタをファイルから読み出し，仮想プログラム・カウンタ・ラッチに同期入力します。この値はジャンプ先の命令をフェッチするために使用します。IF ステージでインクリメントした仮想プログラム・カウンタの値を再びインクリメントし，リンク・アドレスである PC+8 を生成します。ここで，PC とは JALR 命令のアドレスのことです。結果として得られる値が戻り先の PC であり，プログラムはジャンプ先からそこに復帰します。この値を，命令アドレス・ユニットのリンク出力ラッチに移動します。

EX ステージ　PC+8 の値をリンク出力ラッチから EX ステージの出力ラッチに移動します。

DC ステージ　PC+8 の値を EX ステージの出力ラッチから DC ステージの出力ラッチに移動します。

WB ステージ　ADD 命令の場合を参照してください。rd に値を指定しなければ，レジスタ 31 をディフォールトとして使用します。rd に値を指定する場合，rs で指定したレジスタとは異なるレジスタを使用してください。もし同じレジスタを使用した場合，実行結果は不定です。

図 2-14 JALR 命令のパイプライン動作（MIPS III 命令モード時）



2.3.4.3 Branch on Equal 命令（BEQ rs,rt,offset）

IF ステージ　ADD 命令の IF ステージと同じ動作です。

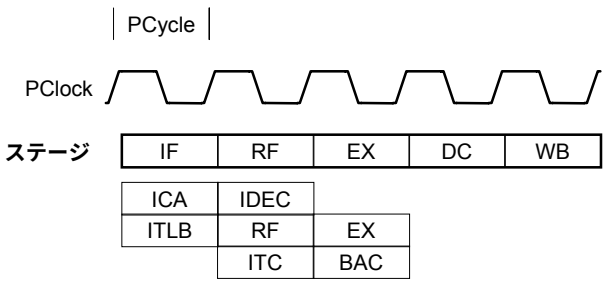
RF ステージ　rs と rt の両領域によってレジスタ・ファイルをアクセスし，これら 2 つのオペランドが同じ値を持っているかどうかをチェックします。両レジスタの値が同じ場合，プログラム・カウンタを PC + target にセットします。ここで，target とは，符号拡張されたオフセットの領域を表します。値が異なる場合，プログラム・カウンタを PC+4 にセットします。

EX ステージ　分岐比較を実行した結果生じる次のプログラム・カウンタの値は，命令フェッチの始めから確定します。

DC ステージ　この命令に関する動作は行いません。

WB ステージ　この命令に関する動作は行いません。

図 2-15 BEQ 命令のパイプライン動作（MIPS III 命令モード時）



2.3.4.4 Trap if Less Than 命令（TLT rs,rt）

IF ステージ　ADD 命令の IF ステージと同じ動作です。

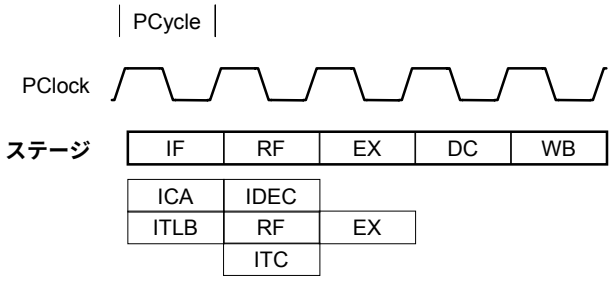
RF ステージ　ADD 命令の RF ステージと同じ動作です。

EX ステージ　ALU 制御部を A－B の演算を実行するようにセットします。オペランドが ALU に入力され，ALU の動作が始まります。演算の結果を ALU 出力ラッチにラッチします。
オペランドの符号ビットと ALU の出力ラッチの符号ビットをチェックし，less than 条件が真であるかどうかを判定します。真ならばトラップ例外を発生します。PC レジスタの値を例外ベクタの値とし，それ以前のパイプライン・ステージに続く命令は無効になります。

DC ステージ　この命令に関する動作は行いません。

WB ステージ　EX ステージで less than 条件が満たされた場合，この命令の PC の値を EPC にストアします。ステータス・レジスタの EXL ビットの内容により，原因レジスタの ExCode 領域と BD ビットを適宜更新します。EX ステージにおいて less than 条件が満たされなかった場合，WB ステージでは何も動作しません。

図 2-16 TLT 命令のパイプライン動作



2.3.4.5 Load word 命令（LW rt, offset（base））

IF ステージ　ADD 命令の IF ステージと同じ動作です。

RF ステージ　ADD 命令の RF ステージと同じ動作です。ただし，rs 領域は base 領域に読み替えてください。

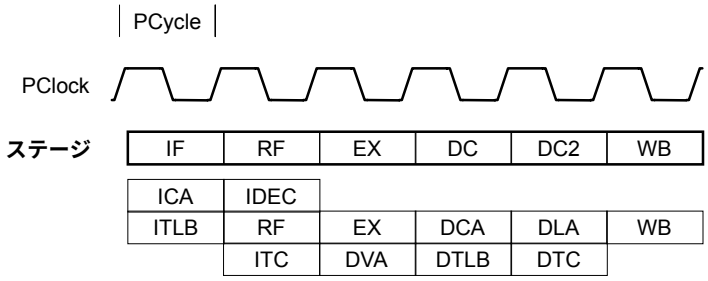
EX ステージ　ADD 命令の EX ステージを参照してください。LW 命令の場合，オペランドをバイパス・マルチプレクサを介して GPR [base]から，および符号拡張したオフセット領域から ALU に入力します。ALU の演算の結果が，オペランドの有効な仮想アドレス（DVA）を表します。

DC ステージ　キャッシュ・アレイからタグとデータを読み出します。キャッシュ・タグ領域を TLB エントリのページ・フレーム番号（PFN）領域と比較します。ロード配列を済ませたあと，DC ステージの出力ラッチに移動します。

DC2 ステージ　データをアラインし，DC2 ステージの出力ラッチに移動します。

WB ステージ　キャッシュ・リード・データを，rt 領域の指定するレジスタ・ファイルに書き込みます。

図 2-17 LW 命令のパイプライン動作（MIPS III 命令モード時）



2.3.4.6 Store word 命令（SW rt, offset（base））

IF ステージ　ADD 命令の IF ステージと同じ動作です。

RF ステージ　LW 命令の RF ステージと同じ動作です。

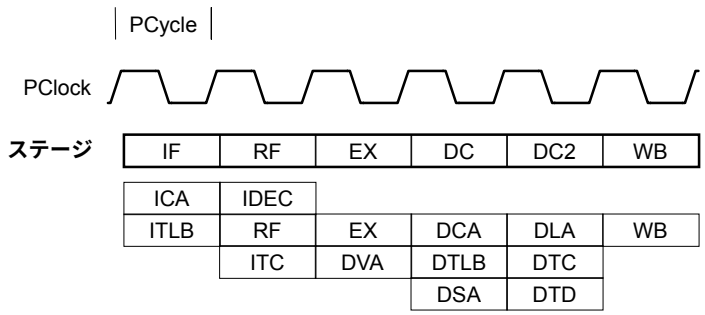
EX ステージ　実効アドレスの計算に関しては LW 命令を参照してください。RF ステージの出力ラッチから GPR[rt]（ストア・データ）がバイパス・マルチプレクサを経由してメイン・シフタに送り込まれ，EX ステージの出力ラッチに移動します。

DC ステージ　キャッシュ・アクセスに関しては，LW 命令を参照してください。ストア・データをアラインします。

DC2 ステージ　キャッシュ・アクセスに関しては，LW 命令を参照してください。

WB ステージ　キャッシュがヒットした場合，ストア・データ出力ラッチの内容をデータ・キャッシュの中の適切なワード位置に書き込みます。
ストア命令はすべて，連続して 2 PCycle の間データ・キャッシュを使用します。もし直後の命令がデータ・キャッシュを使用する必要がある場合，配列を揃えた状態でストア・データを書き込むために 1 PCycle 分だけパイプラインをスリップします。

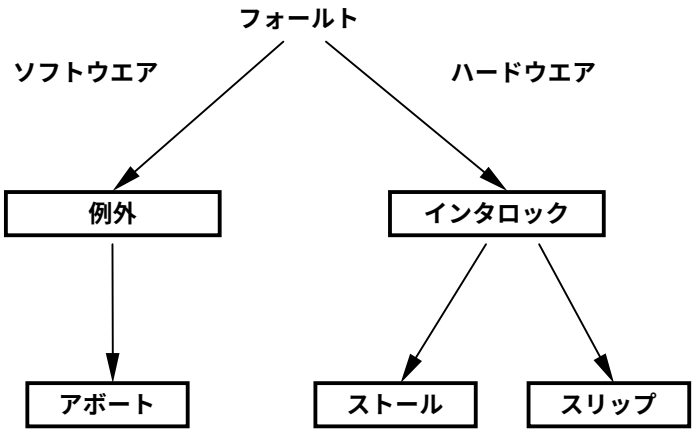
図 2-18　SW 命令のパイプライン動作（MIPS III 命令モード時）



2.3.5 インタロックと例外処理

キャッシュ・ミス，例外の発生，およびデータ依存性を検出したときは，パイプラインの流れを中断します。キャッシュ・ミスのような，ハードウエアで処理するものをインタロックとよびます。一方，ソフトウエアで処理するものを例外と呼びます。そして，図 2-19にあるように，インタロックと例外の条件をまとめてフォールトと呼びます。

図 2-19 インタロック，例外とフォールトの関係



例外とインタロックの条件は，有効な命令すべてに対して各サイクルごとにチェックします。

各例外やインタロックの条件は，特定のパイプライン・ステージに対応していますので，表 2-23に示すように，例外やインタロックが発生したステージの特定の命令まで条件をさかのぼることができます。たとえば，LDI インタロックはレジスタ・フェッチ（RF）ステージで発生しています。

表 2-24から表 2-25に，表 2-23で一覧にしたインタロックと例外についてまとめています。

表 2-23 インタロック，例外の条件とパイプライン・ステージとの相対

ステージ / 状態		IF	RF	EX	DC	WB
インタロック	ストール	－	ITM ICM	－	DTM DCM DCB	－
	スリップ	－	LDI MDI SLI CP0	－	－	－
例外		IAErr	NMI ITLB INTr IBE SYSC BP CUn RSVD	Trap OVF DAErr	Reset DTLB TMod WAT DBE	－

備考　例外の条件は優先順位の高い方から載せています。

表 2-24 パイプライン・インタロック

インタロック	説　明
ITM	命令 TLB ミス
ICM	命令キャッシュ・ミス
LDI	ロード・データ・インタロック
MDI	MD ビジィ・インタロック
SLI	ストア・ロード・インタロック
CP0	コプロセッサ 0 インタロック
DTM	データ TLB ミス
DCM	データ・キャッシュ・ミス
DCB	データ・キャッシュ・ビジィ

表 2-25 パイプライン例外

例 外	説　明
IAErr	命令アドレス・エラー例外
NMI	ノンマスカブル割り込み例外
ITLB	ITLB 例外
INTr	割り込み例外
IBE	命令バス・エラー例外
SYSC	システム・コール例外
BP	ブレークポイント例外
CUn	コプロセッサ使用不可例外
RSVD	予約命令例外
Trap	トラップ例外
OVF	整数オーバフロー例外
DAErr	データ・アドレス・エラー例外
Reset	リセット例外
DTLB	DTLB 例外
DTMod	DTLB 変更例外
WAT	ウォッチ例外
DBE	データ・バス・エラー例外

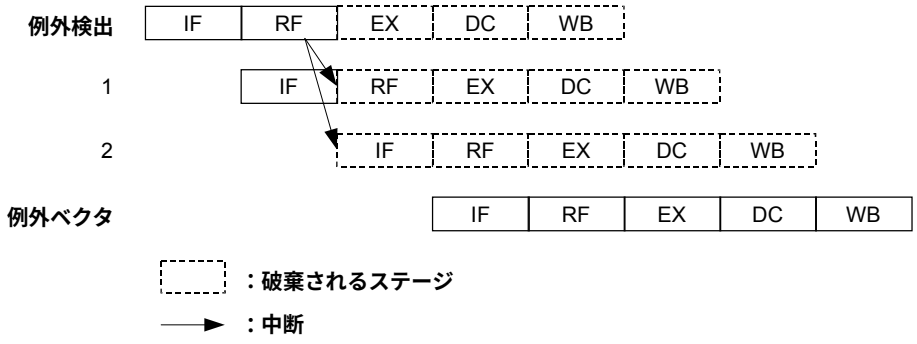
2.3.5.1 例外条件

例外条件が発生すると，関連する命令と，以降のパイプライン中にある命令はすべて中断されます。この命令を参照するストール条件や後続の例外条件は中断されます。中断された命令に対するストールは，何の意味もありません。

命令で例外的な条件が発見されると，VR4120A コアはその命令と以降の命令すべてを破棄します。この命令が WB ステージに達していれば，例外フラグとさまざまな例外情報を CP0 レジスタに書き込み，現在の PC を適切な例外ベクタ・アドレスに変更し，以前のパイプライン・ステージの例外ビットをクリアします。

この実行により，先行するすべての命令は完了し，あとの命令は終了しません。これにより，EPC の値は実行を再開するのに十分なものとなります。これは，例外が実行順に行われるのを確定させます。ある例外のあとに，例外が起こりうる命令があっても，さらに後続のサイクルで例外が起きた場合にはその命令は中断されます。

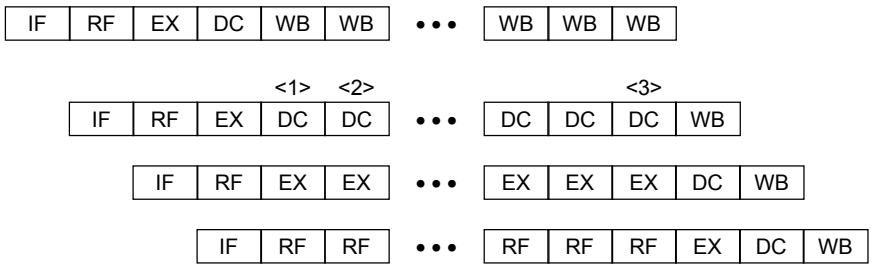
図 2-20 例外検出



2.3.5.2 ストール条件

ストールは，RF ステージのあとに条件が検出された場合にパイプラインを停止するのに使われます。ストールが起こると，プロセッサは条件を解決しパイプラインは続行します。図 2-21はデータ・キャッシュ・ミスによるストール，図 2-22は CACHE 命令によるストールを表します。

図 2-21 データ・キャッシュ・ミス・ストール



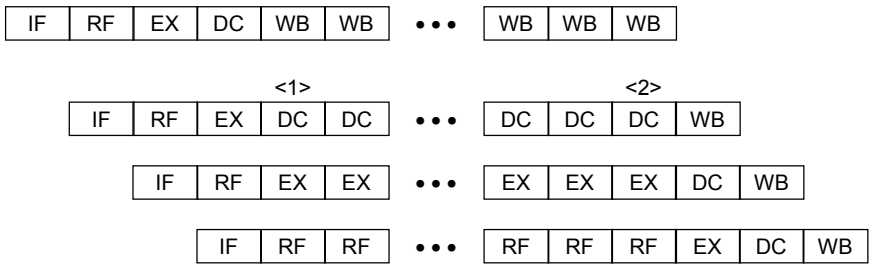
<1> データ・キャッシュ・ミス検出

<2> データ・キャッシュ・ラインをライト・バッファへ移動開始

<3> 最後のワードをキャッシュに取り込み，パイプラインを再開

もしリプレースされるべきキャッシュ・ラインが Dirty（W ビットがセット）なら，データは次のサイクルの内部ライト・バッファに移されます。ライトバック・データはメモリに書き戻されます。データの最後のワードは，<3>でキャッシュに返され，パイプラインは再スタートします。

図 2-22 CACHE 命令ストール



<1> CACHE 命令開始

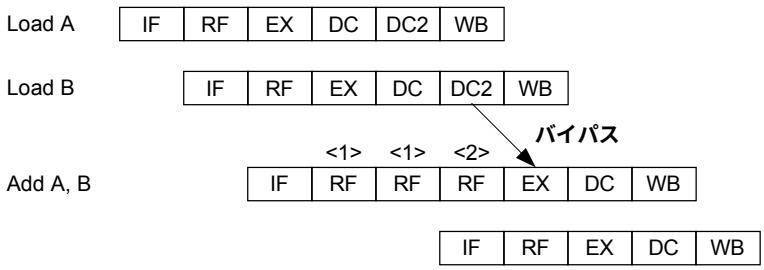
<2> CACHE 命令完了

CACHE 命令がパイプラインの DC ステージに入ったとき，命令が実行されている間パイプラインはストールします。CACHE 命令が完了したとき，命令フェッチが行われ，パイプラインは再開します。

2.3.5.3 スリップ条件

RF ステージと EX ステージの間に，内部ロジックは現在の命令がこのサイクル中に開始可能かどうかを決定します。すべてのソース・オペランドが利用可能で（レジスタ・ファイルから，または内部バイパス・ロジックを経由して），命令の完了に必要なハードウエア資源のすべてが必要なときに利用できる場合，命令は“実行”され，そうでなければ命令は“スリップ”します。スリップした命令は発行されるまで後続のサイクルに後退します。パイプラインの最後（DC ステージと WB ステージ）は，スリップの間，衝突を解決するため正常に進みます。パイプラインのすき間には，NOP が挿入されます。Branch-likely 命令や ERET 命令や例外によって中断された命令は，スリップの原因にはなりません。

図 2-23 ロード・データ・インタロック

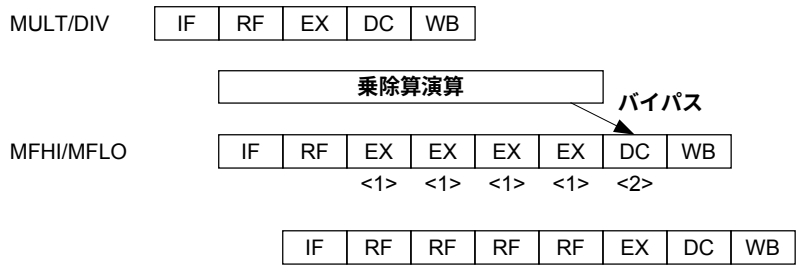


<1> ロード・データ・インタロックの検出

<2> ターゲット・データの取得

ロード・データ・インタロックは図 2-23に示されるように RF ステージで検出され，スリップも RF ステージで起こります。ロード・データ・インタロックはロード命令によってロードしたデータ，または，HI レジスタ，LO レジスタ，CP0 レジスタから移動されるデータを直後の命令で使いたいときに起こります。パイプラインはロードのターゲットがデータ・キャッシュや HI レジスタ，LO レジスタ，CP0 レジスタから読まれたクロックに再開します。DC ステージの最後で返されたデータは，バイパス・マルチプレクサを使って RF ステージの最後に入力されます。

図 2-24 MD ビジィ・インタロック



<1> MD ビジィ・インタロックの検出

<2> ターゲット・データの取得

MD ビジィ・インタロックは図 2-24に示されるように，RF ステージで検出され，パイプラインは RF ステージでスリップします。MD ビジィ・インタロックは乗除算命令が完了する前に MFHI/MFLO 命令によって HI/LO レジスタの内容が必要になる場合に起こります。パイプラインは乗除算の実行後のクロックから再開し始めます。DC ステージの最後に返された HI/LO レジスタからのデータはバイパス・マルチプレクサを使って RF ステージの最後に入力されます。

ストア・ロード・インタロックは EX ステージで検出され，パイプラインは RF ステージでスリップします。ストア・ロード・インタロックはストア命令の直後のロード命令で起きます。パイプラインは１クロック後に再開します。

コプロセッサ 0 インタロックは EX ステージで検出され，パイプラインは RF ステージでスリップします。コプロセッサ・インタロックはコンフィグ・レジスタとステータス・レジスタへの MTC0 命令が検出されたときに起こります。パイプラインは１クロック後に再開します。

2.3.5.4 バイパス

パイプラインの EX ステージ，DC ステージ，WB ステージで生成されたデータと条件はバイパス・データ・パスを通じて EX ステージで利用可能になります。

オペランドをバイパスすると，データや条件が WB ステージの最後にレジスタ・ファイルに書き込まれるのを待たずに，EX ステージ中の命令が実行を続けられます。ただし，バイパス制御部はパイプラインの始めの方の命令が，あとのパイプライン・ステージからのデータや条件を適切なときに確実に利用できるようにしています。

バイパス制御部は，さらに，レジスタ・ファイルから供給されるソースとデスティネーションのレジスタ・アドレスも管理します。

保守／廃止

2.3.6 プログラムの互換性

VR4120A コアは，ほかの VR シリーズとのプログラムの互換性を考慮して設計されています。しかし，アーキテクチャのうえでほかのプロセッサと異なる点がいくつかあるため，ほかのプロセッサ上で実行できるプログラムが必ずしも VR4120A コアで実行できるとはかぎらず，同様に VR4120A コア上で実行できるプログラムがほかのプロセッサで実行できるとはかぎりません。

次に，VR4120A コアとほかの VR シリーズ・プロセッサとの間でプログラムを移植する際に注意すべき点を示します。

- VR4120A コアは浮動小数点ユニット（FPU）を持たないため，FPU 命令はサポートしていません。
- VR4120A コアでは，32 ビット積和演算用命令（DMACC，MACC）が追加されています。この命令は，VR4100 コアや VR4110™ コアでサポートしている 16 ビット積和演算用命令（DMADD16，MADD16）に対し上位互換性があります。
- VR4120A コアでは，パワー・モードをサポートするため，パワー・モード用命令（HIBERNATE, STANDBY, SUSPEND）が追加されています。
- VR4120A コアには，マルチプロセッサ用の同期を行うための LL ビットがありません。このため，LL ビットを操作する命令（LL, LLD, SC, SCD）はサポートしていません。
- VR4120A コアでは，16 ビット長の MIPS16 命令セットが追加されています。ただし，μPD98502 ではこの命令を使用できません
- VR4120A コアの CP0 ハザードの値は，VR4000 と同程度かそれよりも減少しています（詳細は**付録 B CP0 のハザード**参照）。
- VR4120A コアでは，ディバグ用命令が追加されています。ただし，μPD98502 ではこの命令を使用できません（この命令やディバグ機能についての説明も，このマニュアルでは行いません）。

命令の詳細は**付録 A MIPS III 命令セット**，および VR4100，VR4111™，または VR4300™ のユーザーズ・マニュアルを参照してください。

次に，VR シリーズの各製品がサポートする命令の一覧を示します。

保守／廃止

表 2-26 VR シリーズ・サポート命令一覧

製品名 / サポート命令	VR4100 VR4102™	VR4111	VR4120A コア VR4122™	VR4300 VR4305™ VR4310™	VR5000™ VR10000™
MIPS I 命令セット	O	O	O	O	O
MIPS II 命令セット	O	O	O	O	O
MIPS III 命令セット	O	O	O	O	O
LL ビット操作	×	×	×	O	O
MIPS IV 命令セット	×	×	×	×	O
MIPS16 命令セット	×	O	O注	×	×
16 ビット積和演算	O	O	O (32 ビット積和演算を使用)	×	×
32 ビット積和演算	×	×	O	×	×
浮動小数点演算	×	×	×	O	O
パワー・モード遷移	O	O	O	×	×

注　μPD98502 は MIPS16 命令に対応していません。MPS16EN 端子（端子番号 D15）は GND に接続してください。

保守／廃止

2.4 メモリ管理システム

VR4120A コアは，仮想アドレスを物理アドレスに変換する高速変換緩衝機構（TLB）を用いた，メモリ管理ユニット（MMU）を備えています。この章では，TLB のオペレーション，TLB とのソフトウエア・インタフェースとして使用される CP0 レジスタ，仮想アドレスを物理アドレスに変換するメモリ・マッピング方法の詳細について説明します。

2.4.1 高速変換緩衝機構（TLB）

仮想アドレスは，内蔵 TLB を用いて物理アドレスに変換されます注。内蔵 TLB は，32 のエントリを持つフルアソシアティブ・メモリで，1 つのエントリは奇数と偶数のページをペアでマッピングします。これらのページの大きさは，1 K，4 K，16 K，64 K，256 K の 5 種類で，エントリごとに指定できます。仮想アドレスが与えられると各 TLB エントリは，エントリ Hi レジスタに保存されている ASID 領域を付加した仮想アドレスと一致しているかどうか，32 本のエントリに対して同時にチェックを行います。

一致（ヒット）した場合，TLB 内の物理ページ番号とオフセット値から物理アドレスを生成します。

不一致（ミス）の場合，例外が発生し，ソフトウエアは TLB のエントリに，メモリ上にあるページ・テーブルから書き込みます。ソフトウエアは，インデクス・レジスタで選択したエントリかランダム・レジスタが示すランダムなエントリに書き込みます。

一致する TLB エントリが複数ある場合，TLB オペレーションは正常に実行されません。この場合，ステータス・レジスタ内の TLB-Shutdown（TS）ビットが 1 にセットされ，以降，TLB は使用不能となります（TLB アクセスするとヒットするエントリの有無にかかわらず TLB 不一致例外が発生します）。

TS ビットはリセットによってのみクリアできます。

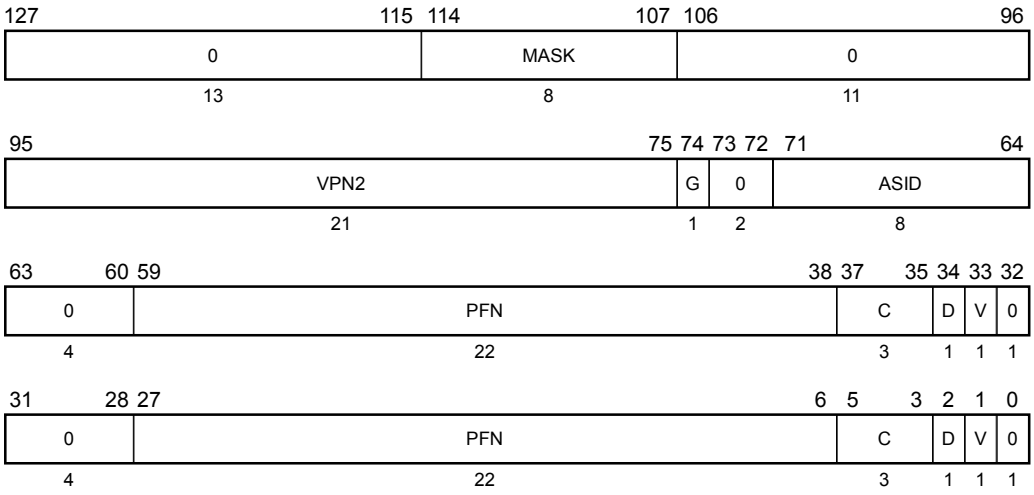
注 ただしアドレス空間によっては，TLB を使わずに仮想アドレスから物理アドレスへ変換する場合があります。たとえば，kseg0 および kseg1 の各空間のアドレスを変換する場合，マッピングという方法は取りません。このような空間では，仮想アドレスからその空間のベース・アドレス分を差し引き，物理アドレスを求めます。

2.4.1.1 TLB エントリの形式

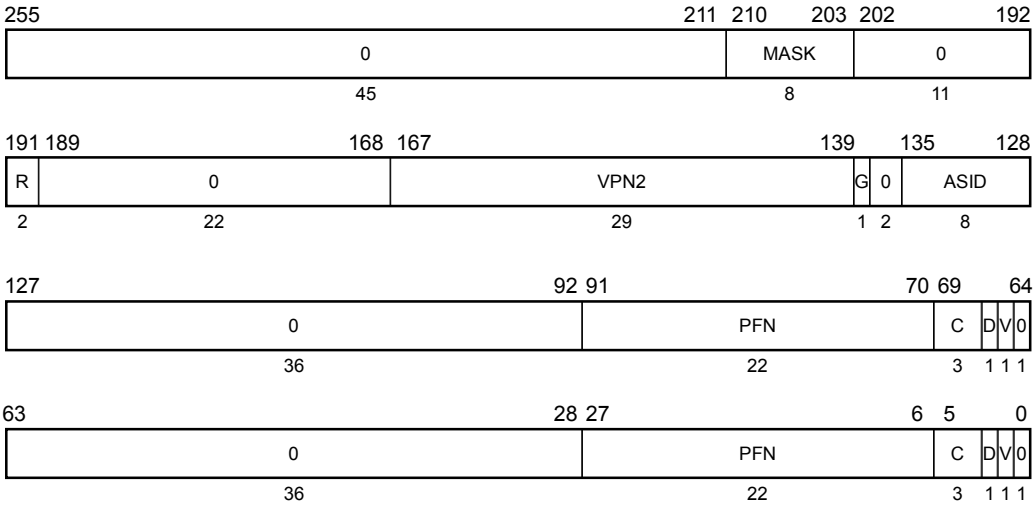
32 ビットおよび 64 ビット・モード時の TLB エントリを図 2-25に示します。各エントリは，エントリ Hi，エントリ Lo0，エントリ Lo1，ページ・マスク・レジスタに対応する領域を持っています。

図 2-25 TLB エントリの形式

(a) 32ビット・モード



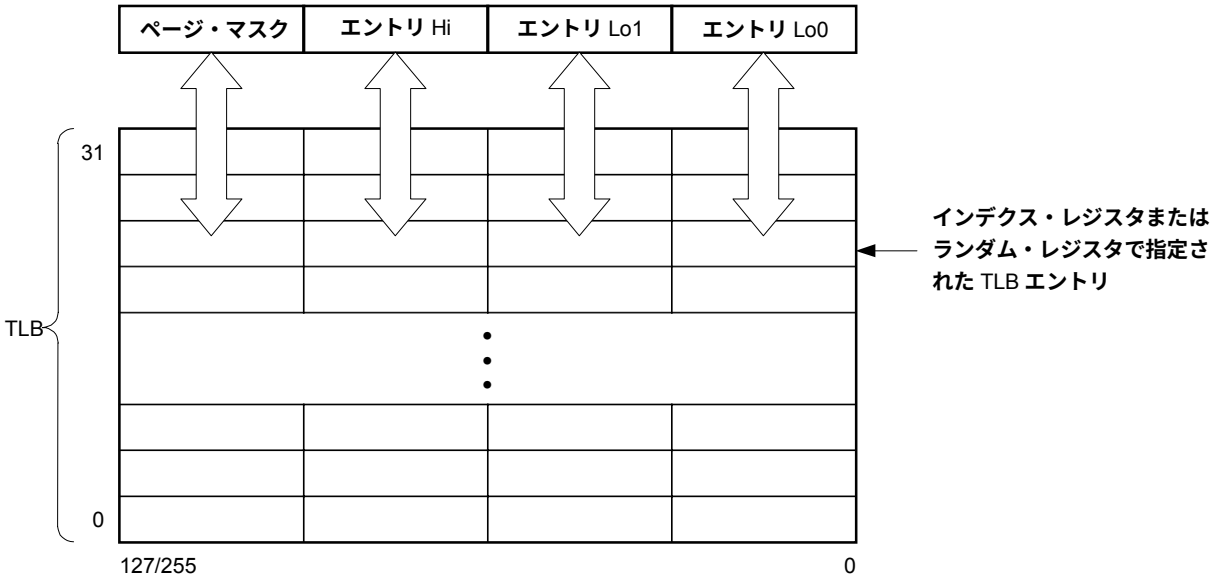
(b) 64ビット・モード



エントリ Hi，エントリ Lo0，エントリ Lo1，ページ・マスク・レジスタの形式は，TLB エントリとほとんど同じです。ただし，TLB の G ビットに対応する位置のビットは，エントリ Hi レジスタでは予約（0）です。また，エントリ Lo レジスタの G ビットに対応する位置のビットは，TLB では予約（0）です。その他の各領域の詳細は，**2.4.6 メモリ管理レジスタ**を参照してください。

各 TLB エントリの内容は，図 2-26に示すように，TLB 操作用の命令により，エントリ Hi，エントリ Lo0，エントリ Lo1，ページ・マスクの各レジスタを通じて読み書きできます。対象となるエントリは，インデクス・レジスタで指定するか，またはランダム・レジスタに示されたものが使用されます。

図 2-26 TLB 操作の概略



2.4.1.2 TLB 命令

次に，TLB 制御のための命令について示します。

（1）TLBP (Translation Lookaside Buffer Probe)

インデクス・レジスタに，エントリ Hi レジスタの内容と一致した TLB エントリ番号がロードされます。TLB エントリが一致しなかった場合，インデクス・レジスタの最上位ビットがセット（1）されます。

（2）TLBR (Translation Lookaside Buffer Read)

インデクス・レジスタの内容が示す TLB エントリの内容をエントリ Hi，エントリ Lo0，エントリ Lo1，ページ・マスク・レジスタに書き込みます。

（3）TLBWI (Translation Lookaside Buffer Write Index)

エントリ Hi，エントリ Lo0，エントリ Lo1，ページ・マスク・レジスタの内容を，インデクス・レジスタの内容が示す TLB エントリに書き込みます。

（4）TLBWR (Translation Lookaside Buffer Write Random)

エントリ Hi，エントリ Lo0，エントリ Lo1，ページ・マスク・レジスタの内容を，ランダム・レジスタの内容が示す TLB エントリに書き込みます。

2.4.1.3 TLB 例外

仮想アドレスと一致する TLB エントリがない場合，TLB 不一致例外が発生します。仮想アドレスと一致する TLB エントリがある場合，アクセス・コントロール・ビット（D, V）が，アクセスが有効でないと示していれば，TLB 変更例外または TLB 無効例外が発生します。また，C 領域が 010 であれば，キャッシュにアクセスせず，検索された物理アドレスによって直接メイン・メモリをアクセスします。

TLB 例外の詳細は**2.5 例外処理**を参照してください。

保守／廃止

2.4.2 仮想アドレスから物理アドレスへの変換

仮想アドレスを物理アドレスに変換するには，まず CPU から送られてくる仮想アドレスを TLB 内のすべてのエントリの仮想アドレスと比較します。はじめに，アドレスの仮想ページ番号（VPN）について，次のいずれかの比較が行われます。

- 32 ビット・モード時：仮想アドレスの上位ビット注と，TLB の各エントリの VPN2（仮想ページ番号を 2 で割ったもの）の内容を比較します。
- 64 ビット・モード時：仮想アドレスの上位ビット注と，TLB の各エントリの R と VPN2（仮想ページ番号を 2 で割ったもの）の内容を比較します。

注 ページ・サイズにより，ビット数が異なります。
ページ・サイズ 256K バイトと 1K バイトの場合を例として示します。

ページ・サイズ / モード	256K バイト	1K バイト
32 ビット・モード	ビット 31-19	ビット 31-11
64 ビット・モード	ビット 63, 62, 39-19	ビット 63, 62, 39-11

この比較で 2 つの領域が同じであるようなエントリがあった場合，次のどちらかに当てはまると，一致が起こります。

- TLB エントリのグローバル・ビット（G）が 1
- 仮想アドレスの ASID 領域が TLB エントリの ASID 領域と同じ

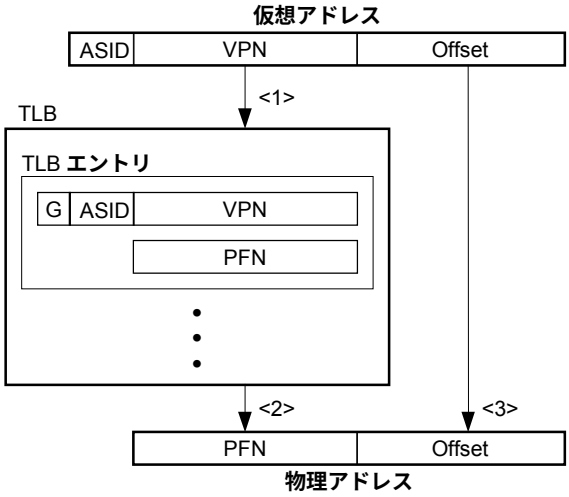
このような一致状態を，TLB ヒットと呼びます。

一致するエントリが TLB 内にある場合，その TLB エントリから物理アドレスとアクセス・コントロール・ビット（C, D, V）が読み出されます。エントリの V ビットは，有効なアドレス変換を行うためにセット（1）しなければなりませんが，一致する TLB エントリの決定とは関連していません。読み出された物理アドレスにはオフセットが付加されます。オフセットは，ページ・フレーム空間内のアドレスを表しています。オフセットの部分は TLB を通過せず，仮想アドレスの下位ビットがそのまま出力されます。

一致が起こらない場合，VR4120A コアが TLB 不一致という例外を発生させ，メモリにある仮想アドレスと物理アドレスが対になっているページ・テーブルを参照し，その内容をソフトウエアで TLB に書き込みます。

図 2-27にアドレス変換の概要を，図 2-28に TLB アドレス変換のフロー・チャートを示します。

図 2-27 仮想アドレスから物理アドレスへの変換

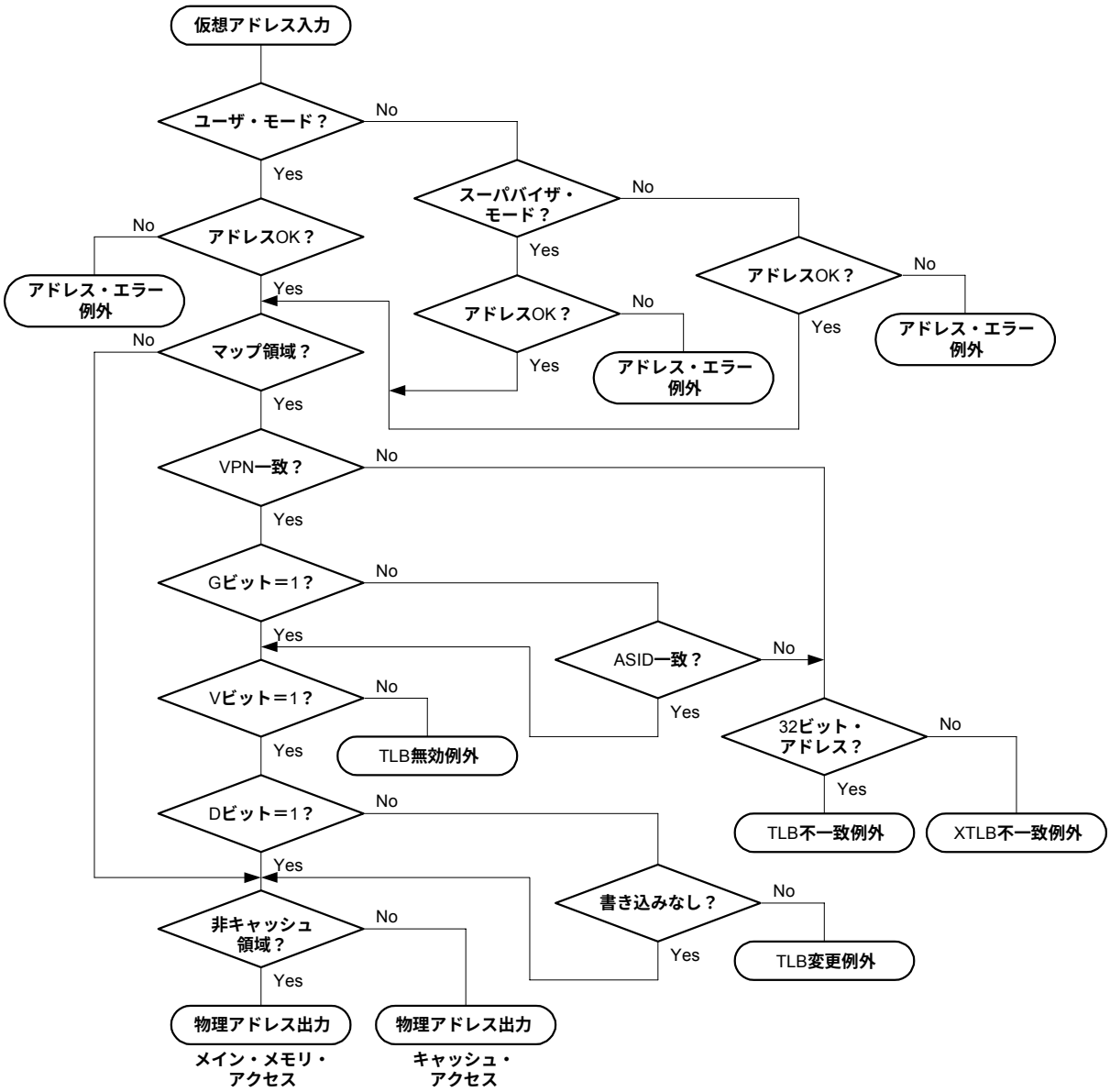


<1>　仮想アドレスのページ番号（VPN，アドレスの上位ビット）が TLB 内の VPN と比較される。

<2>　比較内容が一致すると，物理アドレスの上位ビットを表すページ・フレーム番号（PFN）が TLB から出力される。

<3>　オフセット部分が TLB を介さずに PFN に付加される。

図 2-28 TLB アドレス変換

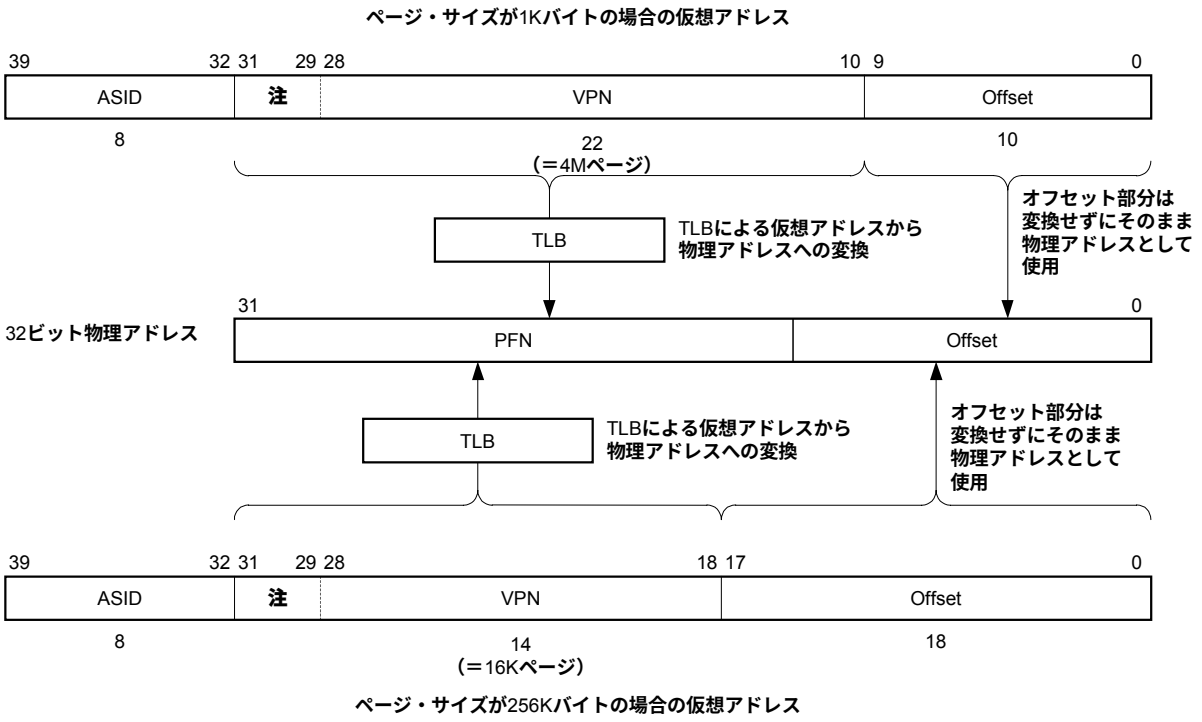


2.4.2.1 32ビット・モード時のアドレス変換

32ビット・モード時の仮想アドレスから物理アドレスへの変換方法を図2-29に示します。5種類のページ・サイズのうち1Kバイト（オフセット10ビット）と256Kバイト（オフセット18ビット）の場合を示しています。

- 図2-29の上部に示しているのが，ページ・サイズ1Kバイト，オフセット10ビットの仮想アドレスです。ASIDを除く残り22ビットが仮想ページ番号（VPN）を表し，4Mエントリのページ・テーブルを選択します。
- 図2-29の下部に示しているのが，ページ・サイズ256Kバイト，オフセット18ビットの仮想アドレスです。ASIDを除く残り14ビットはVPNを表し，16Kエントリのページ・テーブルを選択します。

図2-29 32ビット・モード時の仮想アドレスの変換



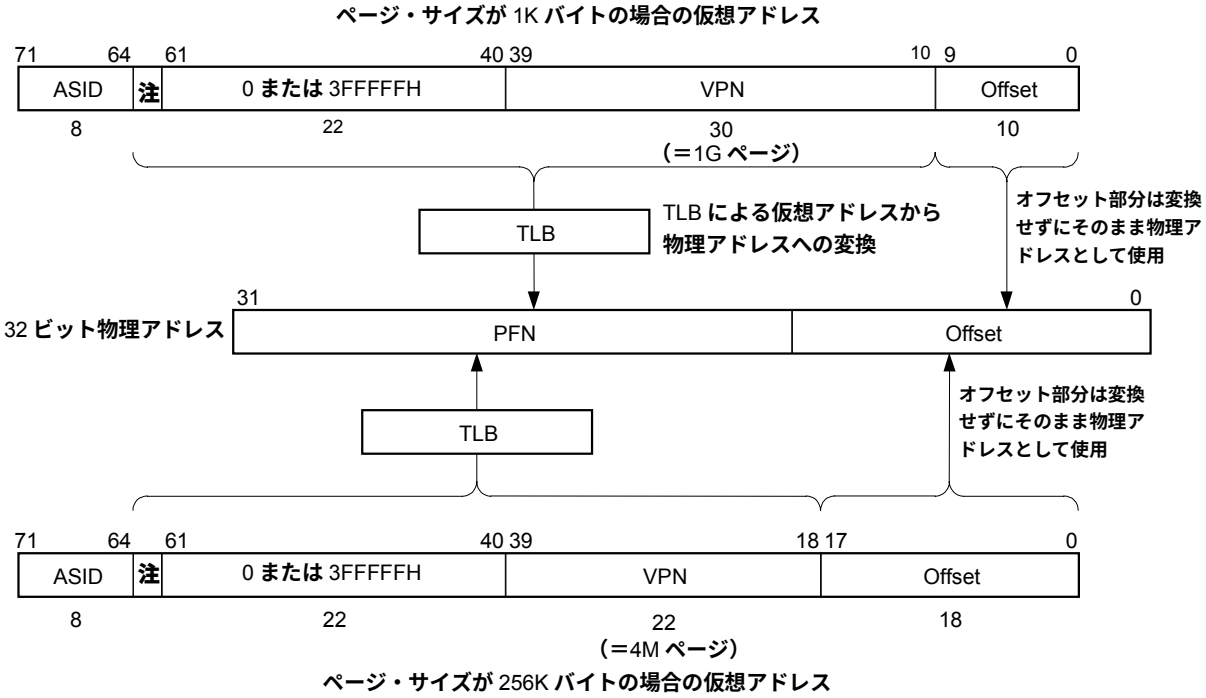
注 仮想アドレスのビット31-29によりユーザ，スーパバイザ，カーネル・アドレス空間が選択されます。

2.4.2.2 64 ビット・モード時のアドレス変換

64 ビット・モード時の仮想アドレスから物理アドレスへの変換方法を図 2-30に示します。5 種類のページ・サイズのうち 1K バイト（オフセット 10 ビット）と 256K バイト（オフセット 18 ビット）の場合を示しています。

- 図 2-30の上部に示しているのが，ページ・サイズ 1K バイト，オフセット 10 ビットの仮想アドレスです。ASID を除く残り 30 ビットが仮想ページ番号（VPN）を表し，1G エントリのページ・テーブルを選択します。
- 図 2-30の下部に示しているのが，ページ・サイズ 256K バイト，オフセット 18 ビットの仮想アドレスです。ASID を除く残り 22 ビットは VPN を表しており，4M エントリのページ・テーブルを選択します。

図 2-30 64 ビット・モード時の仮想アドレスの変換



注 仮想アドレスのビット 63, 62 によりユーザ，スーパバイザ，カーネル・アドレス空間が選択されます。

保守／廃止

2.4.3 仮想アドレス空間

メモリ管理システムは，大きな仮想メモリ空間のアドレスを物理アドレスに変換することによって，アドレス空間を拡張します。

VR4120A コアの物理アドレス空間は 4G バイトで，32 ビットのアドレスを使用します。仮想アドレスは，32 ビット・モードの場合は 32 ビット幅で，最大のユーザ領域は 2G バイト（2^{31} バイト）です。64 ビット・モードの場合，仮想アドレスは 64 ビット幅で，最大のユーザ領域は 1T バイト（2^{40} バイト）です。各モードの TLB エントリの形式は**2.4.1.1 TLB エントリの形式**を参照してください。

仮想アドレスは，アドレス空間 ID（ASID）により拡張されます（図 2-29，図 2-30参照）。ASID により，コンテキスト切り替えの際の TLB フラッシュの回数を減らします。

2.4.3.1 動作モード

VR4120A コアには，32 ビット・モード時と 64 ビット・モード時に機能する，3 つの動作モードがあります。これらの動作モードごとに，アクセスできるアドレス空間が決まっています。

- ユーザ・モード
- スーパバイザ・モード
- カーネル・モード

ユーザ・モードとカーネル・モードは，すべての VR シリーズに共通です。一般的に，カーネル・モードではオペレーティング・システムが実行され，ユーザ・モードではアプリケーション・プログラムが実行されます。VR4000 シリーズは，3 つめのモードとしてスーパバイザ・モードと呼ばれるユーザ・モードとカーネル・モードの中間のモードを持っています。このモードは，セキュリティの高いシステムの構築に使用されます。

例外が発生すると，VR4120A コアはカーネル・モードに入ります。そして，例外復帰命令（ERET）が実行されるまでカーネル・モードのままです。ERET 命令は VR4120A コアを例外が発生する直前のモードに戻します。

カーネル・アドレス空間へのアクセスは，カーネル・モードにおいてのみ許可されます。

スーパバイザ・アドレス空間へのアクセスは，スーパバイザ・モード，またはカーネル・モードにおいて許可されます。

ユーザ・アドレス空間へのアクセスは，すべてのモードで許可されています。

2.4.3.2 ユーザ・モードの仮想アドレシング

ユーザ・モード時，32 ビット・モードでは 2G バイト（2^{31} バイト）の仮想アドレス空間（useg），64 ビット・モードでは 1T バイト（2^{40} バイト）の仮想アドレス空間（xuseg）が使用できます。図 2-29，図 2-30 に示すように，各仮想アドレスは 8 ビットのアドレス空間 ID（ASID）領域によって，最大 256 までのユーザ・プロセスのための，別々の仮想アドレスに拡張されます。各プロセスを ASID によって割り当てることにより，システムはコンテキスト切り替えを行っても，TLB の内容を保持することが可能です。useg, xuseg の参照は TLB を通して行われます。また，キャッシュの使用／不使用は，各ページごとに TLB エントリによって決定されます（TLB エントリの C ビットがキャッシュの使用を決定）。

ユーザ・セグメントはアドレス 0 から始まり，現在有効なユーザ・プロセスが useg（32 ビット・モード時）または xuseg（64 ビット・モード時）のいずれかに常駐します。

VR4120A コアは，ステータス・レジスタ内のビットの値が次のようになっている場合，ユーザ・モードで動作します。

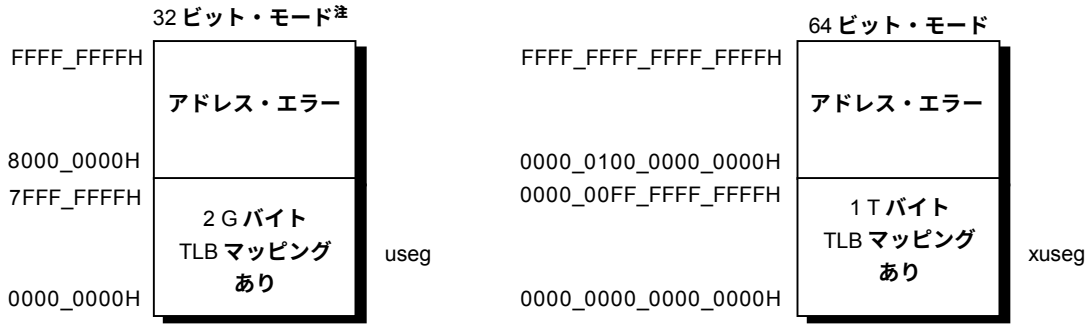
- KSU 領域＝10
- EXL ビット＝0
- ERL ビット＝0

これらのビットのほか，ステータス・レジスタ内の UX ビットがアドレシング・モードを決定します。

- UX ビット＝0：32 ビットの useg 空間を選択
- UX ビット＝1：64 ビットの xuseg 空間を選択

図 2-31にユーザ・モードのマッピングを，表 2-27に各ユーザ・セグメント（useg と xuseg）の特徴を示します。

図 2-31 ユーザ・モード・アドレス空間



注 VR4120A コアは，内部では 64 ビットのアドレスを使用します。カーネル・モード時，VR4120A コアはコンテキスト切り替えを行う前に各レジスタを退避し，初期化します。32 ビット・モード時のアドレスには，32 ビット値のビット 31 をビット 32-63 へ符号拡張した値を使用します。通常は，32 ビット・モードのプログラムは無効なアドレスを生成しません。しかしコンテキスト切り替えが発生し，カーネル・モードに遷移すると，前記の符号拡張された 32 ビットのアドレス以外の値を 64 ビット・レジスタに格納する可能性があります。このような場合にユーザ・モードのプログラムは無効アドレスを生成する可能性があります。

保守／廃止

表 2-27 ユーザ・モードのセグメント

モード	アドレス・ビット値	ステータス・レジスタ・ビット値				セグメント名	アドレス範囲	サイズ
		KSU	EXL	ERL	UX			
32 ビット	A31 = 0	10	0	0	0	useg	0000_0000H ～ 7FFF_FFFFH	2G バイト (2^{31} バイト)
64 ビット	A(63:40) = 0	10	0	0	1	xuseg	0000_0000_0000_0000H ～ 0000_00FF_FFFF_FFFFH	1T バイト (2^{40} バイト)

(1) useg（32 ビット・モード）

ステータス・レジスタの UX ビットが 0 で，仮想アドレスの最上位ビットが 0 の場合，この仮想アドレス空間を useg と呼びます。最上位ビットが 1 のアドレスを参照しようとした場合，アドレス・エラー例外が発生します（**2.5 例外処理**参照）。

TLB 不一致例外が発生した場合，TLB 不一致ベクタが使用されます。

(2) xuseg（64 ビット・モード）

ステータス・レジスタの UX ビットが 1 で，仮想アドレスのビット 63-40 がすべて 0 の場合，この仮想アドレス空間を xuseg と呼び，1T バイト（2^{40} バイト）のユーザ・アドレス空間を使用できます。ビット 63-40 の中に 1 を含むアドレスを参照しようとした場合，アドレス・エラー例外が発生します（**2.5 例外処理**参照）。TLB 不一致例外が発生した場合，XTLB 不一致ベクタが使用されます。

2.4.3.3 スーパバイザ・モードの仮想アドレシング

スーパバイザ・モードは，オペレーティング・システムの実行を階層化するためのモードです。カーネル・モードではオペレーティング・システムのうちのカーネルが実行され，その他の部分はスーパバイザ・モードで実行されます。

すべてのスーパバイザ空間（suseg, sseg, xsuseg, xsseg, csseg）の参照は TLB を通して行われます。また，キャッシュの使用／不使用は，各ページごとに TLB エントリによって決定されます（TLB エントリの C ビットがキャッシュの使用を決定）。

ステータス・レジスタのビットがそれぞれ次のような状態のとき，VR4120A コアはスーパバイザ・モードで動作します。

- KSU 領域＝01
- EXL ビット＝0
- ERL ビット＝0

さらに，ステータス・レジスタの SX ビットによって，スーパバイザ・モードのアドレシング・モードが決まります。

- SX ビット＝0：32 ビットのスーパバイザ空間
- SX ビット＝1：64 ビットのスーパバイザ空間

保守／廃止

図 2-32にスーパバイザ・モードのアドレス・マッピング，表 2-28にスーパバイザ・モードの各セグメントの特徴を示します。

図 2-32 スーパバイザ・モード・アドレス空間

注 VR4120A コアは，内部では 64 ビットのアドレスを使用します。32 ビット・モード時のアドレスには，32 ビット値のビット 31 をビット 32-63 へ符号拡張した値を使用します。通常は，32 ビット・モードのプログラムは無効なアドレスを生成しません。しかし，アドレス計算をするときのベース・レジスタ＋オフセットという演算で，2 の補数オーバフローが発生する可能性があります。このとき算出したアドレスは無効となり，結果は未定義となるので注意してください。次に 2 つの発生要因を示します。

- オフセットのビット 15＝0，ベース・レジスタのビット 31＝0，（ベース・レジスタ＋オフセット）のビット 31＝1 の場合。
- オフセットのビット 15＝1，ベース・レジスタのビット 31＝1，（ベース・レジスタ＋オフセット）のビット 31＝0 の場合。

表 2-28 32 ビットと 64 ビットのスーパバイザ・モードのセグメント

アドレス・ビット値	ステータス・レジスタ・ビット値				セグメント名	アドレス範囲	サイズ
	KSU	EXL	ERL	SX			
32 ビット A31 = 0	01	0	0	0	suseg	0000_0000H ～ 7FFF_FFFFH	2G バイト （2^{31} バイト）
32 ビット A(31:29) = 110	01	0	0	0	sseg	C000_0000H ～ DFFF_FFFFH	512M バイト （2^{29} バイト）
64 ビット A(63:62) = 00	01	0	0	1	xsuseg	0000_0000_0000_0000H ～ 0000_00FF_FFFF_FFFFH	1T バイト （2^{40} バイト）
64 ビット A(63:62) = 01	01	0	0	1	xsseg	4000_0000_0000_0000H ～ 4000_00FF_FFFF_FFFFH	1T バイト （2^{40} バイト）
64 ビット A(63:62) = 11	01	0	0	1	csseg	FFFF_FFFF_C000_0000H ～ FFFF_FFFF_DFFF_FFFFH	512M バイト （2^{29} バイト）

(1) suseg（32 ビット・スーパバイザ・モード，ユーザ空間）

ステータス・レジスタ内の SX ビットが 0 で，仮想アドレスの最上位ビットが 0 のとき，suseg と呼ばれる仮想アドレス空間が選択されます。このセグメントは，2G バイト（2^{31} バイト）の通常のユーザ・アドレス空間です。仮想アドレスは，8 ビットの ASID 領域の内容により別々の仮想アドレスに拡張されます。このマップされた空間の仮想アドレスは 0000_0000H～7FFF_FFFFH です。

(2) sseg（32 ビット・スーパバイザ・モード，スーパバイザ空間）

ステータス・レジスタ内の SX ビットが 0 で，仮想アドレスの上位 3 ビットが 110 のとき，sseg と呼ばれる仮想アドレス空間が選択されます。このセグメントは，512M バイト（2^{29} バイト）の通常のスーパバイザ・アドレス空間です。仮想アドレスは，8 ビットの ASID 領域の内容により別々の仮想アドレスに拡張されます。このマップされた空間の仮想アドレスは C000_0000H～DFFF_FFFFH です。

(3) xsuseg（64 ビット・スーパバイザ・モード，ユーザ空間）

ステータス・レジスタ内の SX ビットが 1 で，仮想アドレスのビット 63, 62 が 00 のとき，xsuseg と呼ばれる仮想アドレス空間が選択されます。このセグメントは，1T バイト（2^{40} バイト）の通常のユーザ・アドレス空間です。仮想アドレスは，8 ビットの ASID 領域の内容により別々の仮想アドレスに拡張されます。このマップされた空間の仮想アドレスは 0000_0000_0000_0000H ～ 0000_00FF_FFFF_FFFFH です。

(4) xsseg（64 ビット・スーパバイザ・モード，通常スーパバイザ空間）

ステータス・レジスタ内の SX ビットが 1 で，仮想アドレスのビット 63, 62 が 01 のとき，xsseg と呼ばれる仮想アドレス空間が選択されます。このセグメントは，1T バイト（2^{40} バイト）の通常のスーパバイザ・アドレス空間です。仮想アドレスは，8 ビットの ASID 領域の内容により別々の仮想アドレスに拡張されます。このマップされた空間の仮想アドレスは 4000_0000_0000_0000H ～ 4000_00FF_FFFF_FFFFH です。

(5) csseg（64 ビット・スーパバイザ・モード，独立スーパバイザ空間）

ステータス・レジスタ内の SX ビットが 1 で，仮想アドレスのビット 63, 62 が 11 のとき，csseg と呼ばれる仮想アドレス空間が選択されます。このセグメントは，512M バイト（2^{29} バイト）の独立したスーパバイザ・アドレス空間です。仮想アドレスは，8 ビットの ASID 領域の内容により別々の仮想アドレスに拡張されます。このマップされた空間の仮想アドレスは FFFF_FFFF_C000_0000H ～ FFFF_FFFF_DFFF_FFFFH です。

2.4.3.4 カーネル・モードの仮想アドレシング

ステータス・レジスタが，次にあげる条件を 1 つ以上満たしている場合，VR4120A コアはカーネル・モードで動作します。

- KSU 領域＝00
- EXL ビット＝1
- ERL ビット＝1

さらに，ステータス・レジスタの KX ビットの値によって，次のようにカーネル・モードのアドレス幅が変わります。

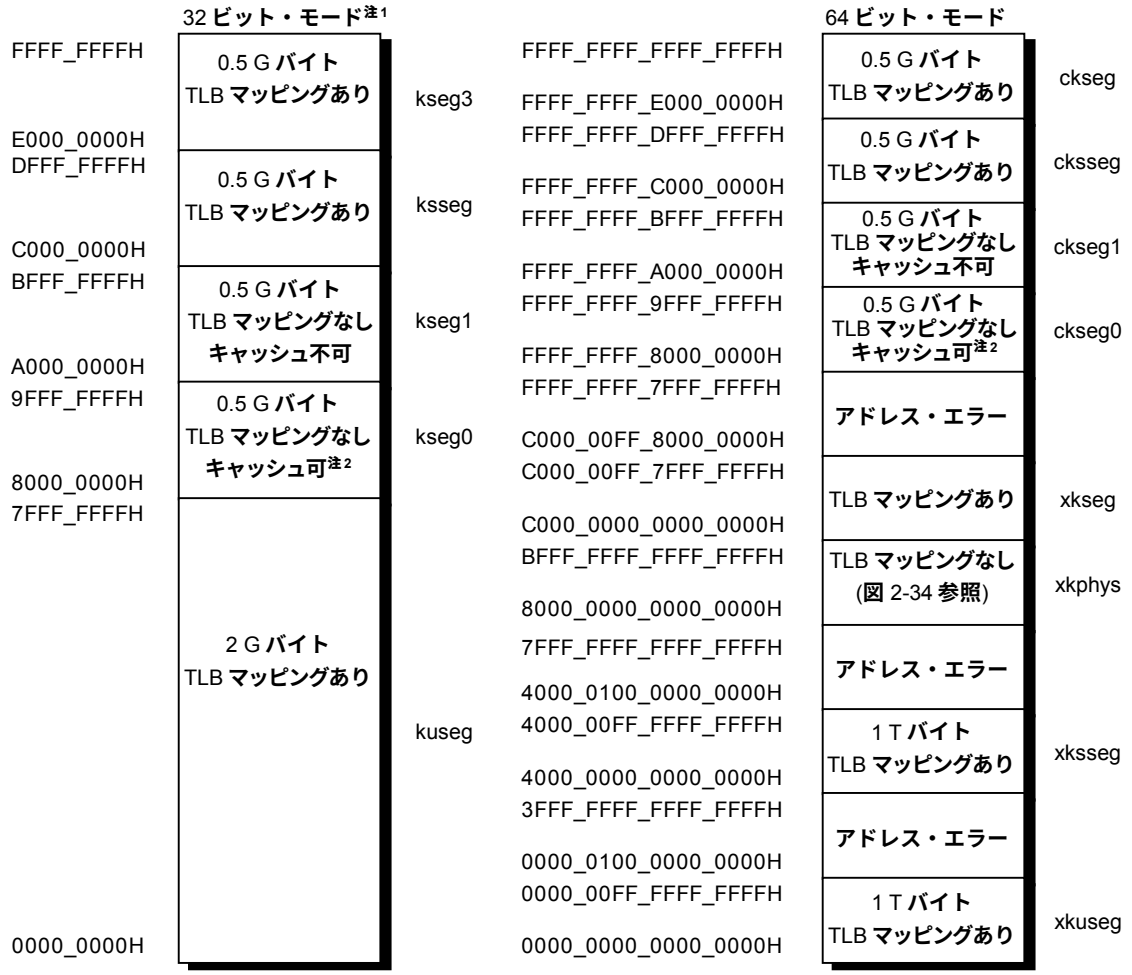
- KX ビット＝0：32 ビットのカーネル空間
- KX ビット＝1：64 ビットのカーネル空間

例外が発生すると，VR4120A コアはカーネル・モードになります。そして，例外復帰（ERET）命令を実行すると ERL ビットと EXL ビットの両方か一方が 0 になってカーネル・モードは終了し，例外の発生した直前のモードに戻ります。

カーネル・モードの仮想アドレス空間は，仮想アドレスの上位ビットの値によって，いくつかの領域に分割されます。その様子を図 2-33に示します。また，表 2-29と表 2-30に，32 ビットと 64 ビットのカーネル・モードの各セグメントの特徴をそれぞれ示します。

保守／廃止

図 2-33 カーネル・モード・アドレス空間



注 1. VR4120A コアは，内部では 64 ビットのアドレスを使用します。32 ビット・モード時のアドレスには，32 ビット値のビット 31 をビット 32-63 へ符号拡張した値を使用します。通常は，32 ビット・モードのプログラムは 64 ビット命令を使用しますが，アドレス計算をするときのベース・レジスタ＋オフセットという演算で，2 の補数オーバフローが発生する可能性があります。このとき算出したアドレスは無効となり，結果は未定義となるので注意してください。次に 2 つの発生要因を示します。

- オフセットのビット 15 = 0，ベース・レジスタのビット 31 = 0，（ベース・レジスタ＋オフセット）のビット 31 = 1 の場合。
- オフセットのビット 15 = 1，ベース・レジスタのビット 31 = 1，（ベース・レジスタ＋オフセット）のビット 31 = 0 の場合。

2. kseg0 と ckseg0 をキャッシュ領域とするかどうかはコンフィグ・レジスタの K0 領域の状態で決まります。

図 2-34 xkphys 領域の詳細

表 2-29　32 ビットのカーネル・モードのセグメント

アドレス・ビット値	ステータス・レジスタ・ビット値				セグメント名	仮想アドレス	物理アドレス	サイズ
	KSU	EXL	ERL	KX				
A31 = 0	KSU = 00 または EXL = 1 または ERL = 1			0	kuseg	0000_0000H ～ 7FFF_FFFFH	TLB マップ	2 G バイト (2^{31} バイト)
A(31:29) = 100				0	kseg0	8000_0000H ～ 9FFF_FFFFH	0000_0000H ～ 1FFF_FFFFH	512 M バイト (2^{29} バイト)
A(31:29) = 101				0	kseg1	A000_0000H ～ BFFF_FFFFH	0000_0000H ～ 1FFF_FFFFH	512 M バイト (2^{29} バイト)
A(31:29) = 110				0	ksseg	C000_0000H ～ DFFF_FFFFH	TLB マップ	512 M バイト (2^{29} バイト)
A(31:29) = 111				0	kseg3	E000_0000H ～ FFFF_FFFFH	TLB マップ	512 M バイト (2^{29} バイト)

(1) kuseg（32 ビット・カーネル・モード，ユーザ空間）

ステータス・レジスタ内の KX ビットが 0 で，仮想アドレスの最上位ビットが 0 のとき，kuseg と呼ばれる仮想アドレス空間が選択されます。この空間は，2G バイト（2^{31} バイト）の通常のユーザ・アドレス空間です。仮想アドレスは，8 ビットの ASID 領域の内容により別々の仮想アドレスに拡張されます。

この空間は TLB を通して参照します。またキャッシュの使用／不使用は，各ページの TLB エントリの C ビットの値で決まります。

ステータス・レジスタの ERL ビットが 1 の場合，ユーザ・アドレス領域は 2G バイト（2^{31} バイト）の非マップ領域（仮想アドレスをそのまま物理アドレスとする），非キャッシュ領域となります。これにより，キャッシュ・エラー例外ハンドラは r0 をベース・レジスタに用いることで非キャッシュで実行できます。

(2) kseg0（32 ビット・カーネル・モード，カーネル空間 0）

ステータス・レジスタ内の KX ビットが 0 で，仮想アドレスの上位 3 ビットが 100 のとき，kseg0 と呼ばれる仮想アドレス空間が選択されます。この空間は，512M バイト（2^{29} バイト）の通常の物理アドレス空間です。

kseg0 の参照は TLB を通さず，仮想アドレスから 8000_0000H を引いて物理アドレスを算出して行います。またキャッシュの使用とコヒーレンシは，コンフィグ・レジスタの K0 領域によって制御されます（**2.5 例外処理**参照）。

(3) kseg1（32 ビット・カーネル・モード，カーネル空間 1）

ステータス・レジスタ内の KX ビットが 0 で，仮想アドレスの上位 3 ビットが 101 のとき，kseg1 と呼ばれる仮想アドレス空間が選択されます。この空間は，512M バイト（2^{29} バイト）の通常の物理アドレス空間です。

kseg1 の参照は，TLB を通さず，仮想アドレスから A000_0000H を引いて物理アドレスを算出して行います。この空間にアクセスするとキャッシュは無効になり，直接メイン・メモリ（または，メモリにマッピングされた I/O デバイス・レジスタ）を選択します。

(4) ksseg（32 ビット・カーネル・モード，スーパバイザ空間）

ステータス・レジスタ内の KX ビットが 0 で，仮想アドレスの上位 3 ビットが 110 のとき，ksseg と呼ばれる仮想アドレス空間が選択されます。この空間は，512M バイト（2^{29} バイト）の通常のスーパバイザ・アドレス空間です。仮想アドレスは，8 ビットの ASID 領域の内容により別々の仮想アドレスに拡張されます。

この空間は TLB を通して参照します。また，キャッシュの使用／不使用は，各ページの TLB エントリの C ビットの値で決まります。

(5) kseg3（32 ビット・カーネル・モード，カーネル空間 3）

ステータス・レジスタ内の KX ビットが 0 で，仮想アドレスの上位 3 ビットが 111 のとき，kseg3 と呼ばれる仮想アドレス空間が選択されます。この空間は，512M バイト（2^{29} バイト）の通常のカーネル・アドレス空間です。仮想アドレスは，8 ビットの ASID 領域の内容により別々の仮想アドレスに拡張されます。

この空間は TLB を通して参照します。また，キャッシュの使用／不使用は，各ページの TLB エントリの C ビットの値で決まります。

表 2-30 64 ビットのカーネル・モードのセグメント

アドレス・ビット値	ステータス・レジスタ・ビット値				セグメント名	仮想アドレス	物理アドレス	サイズ
	KSU	EXL	ERL	KX				
A(63:62) = 00	KSU = 00 または EXL = 1 または ERL = 1			1	xkuseg	0000_0000_0000_0000H ～ 0000_00FF_FFFF_FFFFH	TLB マップ	1 T バイト (2^{40} バイト)
A(63:62) = 01				1	xksseg	4000_0000_0000_0000H ～ 4000_00FF_FFFF_FFFFH	TLB マップ	1 T バイト (2^{40} バイト)
A(63:62) = 10				1	xkphys	8000_0000_0000_0000H ～ BFFF_FFFF_FFFF_FFFFH	0000_0000H ～ FFFF_FFFFH	4 G バイト (2^{32} バイト)
A(63:62) = 11				1	xkseg	C000_0000_0000_0000H ～ C000_00FF_7FFF_FFFFH	TLB マップ	2^{40} バイト ～ 2^{31} バイト
A(63:62) = 11 A(63:31) = 1111…1				1	ckseg0	FFFF_FFFF_8000_0000H ～ FFFF_FFFF_9FFF_FFFFH	0000_0000H ～ 1FFF_FFFFH	512 M バイト (2^{29} バイト)
A(63:62) = 11 A(63:31) = 1111…1				1	ckseg1	FFFF_FFFF_A000_0000H ～ FFFF_FFFF_BFFF_FFFFH	0000_0000H ～ 1FFF_FFFFH	512 M バイト (2^{29} バイト)
A(63:62) = 11 A(63:31) = 1111…1				1	cksseg	FFFF_FFFF_C000_0000H ～ FFFF_FFFF_DFFF_FFFFH	TLB マップ	512 M バイト (2^{29} バイト)
A(63:62) = 11 A(63:31) = 1111…1				1	ckseg3	FFFF_FFFF_E000_0000H ～ FFFF_FFFF_FFFF_FFFFH	TLB マップ	512 M バイト (2^{29} バイト)

(6) xkuseg（64 ビット・カーネル・モード，ユーザ空間）

ステータス・レジスタ内の KX ビットが 1 で，仮想アドレスのビット 63, 62 が 00 のとき，xkuseg と呼ばれる仮想アドレス空間が選択されます。この空間は，1T バイト（2^{40} バイト）の通常のユーザ・アドレス空間です。仮想アドレスは，8 ビットの ASID 領域の内容により別々の仮想アドレスに拡張されます。この空間は TLB を通して参照します。また，キャッシュの使用／不使用は，各ページの TLB エントリの C ビットの値で決まります。

ステータス・レジスタの ERL ビットが 1 の場合，ユーザ・アドレス領域は 2G バイト（2^{31} バイト）の TLB マッピングなし（仮想アドレスをそのまま物理アドレスとする），非キャッシュ領域となります。これにより，キャッシュ・エラー例外ハンドラは r0 をベース・レジスタに用いることで非キャッシュで実行できます。

(7) xksseg（64 ビット・カーネル・モード，通常スーパバイザ空間）

ステータス・レジスタ内の KX ビットが 1 で，仮想アドレスのビット 63, 62 が 01 のとき，xksseg と呼ばれる仮想アドレス空間が選択されます。この空間は，1T バイト（2^{40} バイト）通常のスーパバイザ・アドレス空間です。仮想アドレスは，8 ビットの ASID 領域の内容により別々の仮想アドレスに拡張されます。

この空間は TLB を通して参照します。また，キャッシュの使用／不使用は，各ページの TLB エントリの C ビットの値で決まります。

(8) xkphys (64-bit kernel mode, physical spaces)

ステータス・レジスタ内の KX ビットが 1 で，仮想アドレスのビット 63, 62 が 10 のとき，xkphys と呼ばれる仮想アドレス空間が選択されます。この空間は，参照時に TLB を通さない物理アドレス空間で，キャッシュ領域か非キャッシュ領域のどちらかが選択されます。仮想アドレスのビット 58-32 の中に 1 が含まれている場合，アドレス・エラーが発生します。

キャッシュの使用は，仮想アドレスのビット 61-59 で示します。表 2-31に，8 つのアドレス空間と対応するキャッシュの使用を示します。

表 2-31 キャッシュの使用と xkphys アドレス空間

仮想アドレスのビット 61-59	キャッシュの使用	アドレス
0	使用	8000_0000_0000_0000H ～ 8000_0000_FFFF_FFFFH
1	使用	8800_0000_0000_0000H ～ 8800_0000_FFFF_FFFFH
2	使用しない	9000_0000_0000_0000H ～ 9000_0000_FFFF_FFFFH
3	使用	9800_0000_0000_0000H ～ 9800_0000_FFFF_FFFFH
4	使用	A000_0000_0000_0000H ～ A000_0000_FFFF_FFFFH
5	使用	A800_0000_0000_0000H ～ A800_0000_FFFF_FFFFH
6	使用	B000_0000_0000_0000H ～ B000_0000_FFFF_FFFFH
7	使用	B800_0000_0000_0000H ～ B800_0000_FFFF_FFFFH

保守／廃止

(9) xkseg（64 ビット・カーネル・モード，カーネル空間）

ステータス・レジスタ内の KX ビットが 1 で，仮想アドレスのビット 63, 62 が 11 のとき，xkseg と呼ばれる仮想アドレス空間が選択されます。この空間は，次にあげるアドレス空間のうちどちらかとなります。

- カーネル仮想空間 xkseg（通常のカーネル仮想空間）。仮想アドレスは 8 ビットの ASID 領域の内容によって別々の仮想アドレスに拡張されます。
  この空間は TLB を通して参照します。またキャッシュの使用／不使用は，各ページの TLB エントリの C ビットの値で決まります。
- 32 ビットのカーネル空間と互換性のある 4 つのうち 1 つ。詳細は次に説明します。

(10) 64 ビット・カーネル・モード互換空間（ckseg0, ckseg1, cksseg, ckseg3）

カーネル・モードにおいて次の条件のとき，アドレスの下位 2 バイトによって，互換空間として ckseg0, ckseg1, cksseg, ckseg3（各空間とも 512M バイト）のうちのどれかを選択します。

- ステータス・レジスタ内の KX 領域が 1
- 64 ビットの仮想アドレスのビット 63, 62 が 11
- 仮想アドレスのビット 61-31 の値がすべて 1

(a) ckseg0

この空間は 32 ビット・モード時の kseg0 空間と互換性があり，非マップ領域です。キャッシュの使用とコヒーレンシはコンフィグ・レジスタの K0 領域によって制御します。

(b) ckseg1

この空間は 32 ビット・モード時の kseg1 空間と互換性があり，非マップ，非キャッシュ領域です。

(c) cksseg

この空間は通常のスーパバイザ仮想空間であり，32 ビット・モード時の ksseg 空間と互換性があります。
この空間は TLB を通して参照します。また，キャッシュの使用／不使用は，各ページの TLB エントリの C ビットの値で決まります。

(d) ckseg3

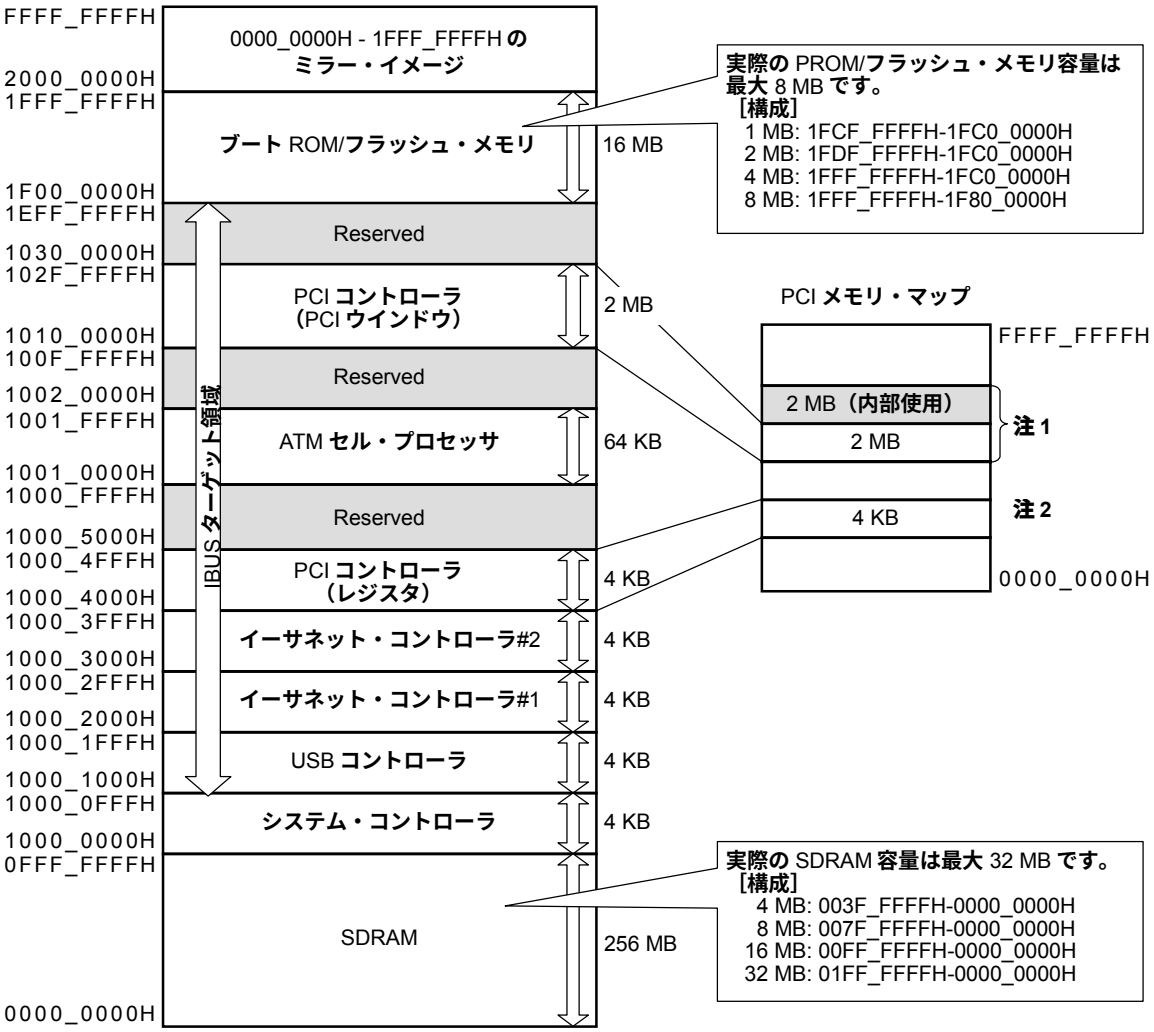
この空間は通常のカーネル仮想空間であり，32 ビット・モード時の kseg3 空間と互換性があります。
この空間は TLB を通して参照します。また，キャッシュの使用／不使用は，各ページの TLB エントリの C ビットの値で決まります。

保守／廃止

2.4.4 物理アドレス空間

VR4120A コアは 32 ビット・アドレスを使用するので，プロセッサの物理アドレス空間が 4 G バイトになります。この 4 G バイトの物理アドレス空間をμPD98502 では図 2-35のように使用しています。

図 2-35 μPD98502 物理アドレス空間



注 1.　この領域は P_PLBA，P_IBBA レジスタで割り当てられているμPD98502 の PCI ウインドウです。
7.5.2 P_PLBA（PCI 下位ベース・アドレス・レジスタ）および 7.5.3 P_IBBA（内部ベース・アドレス・レジスタ）を参照してください。

2.　この領域はレジスタ・メモリ・ベース・アドレス・レジスタで割り当てられているμPD98502 の I/O 領域の PCI ウインドウです。
7.5.18.12 レジスタ・メモリ・ベース・アドレス・レジスタを参照してください。

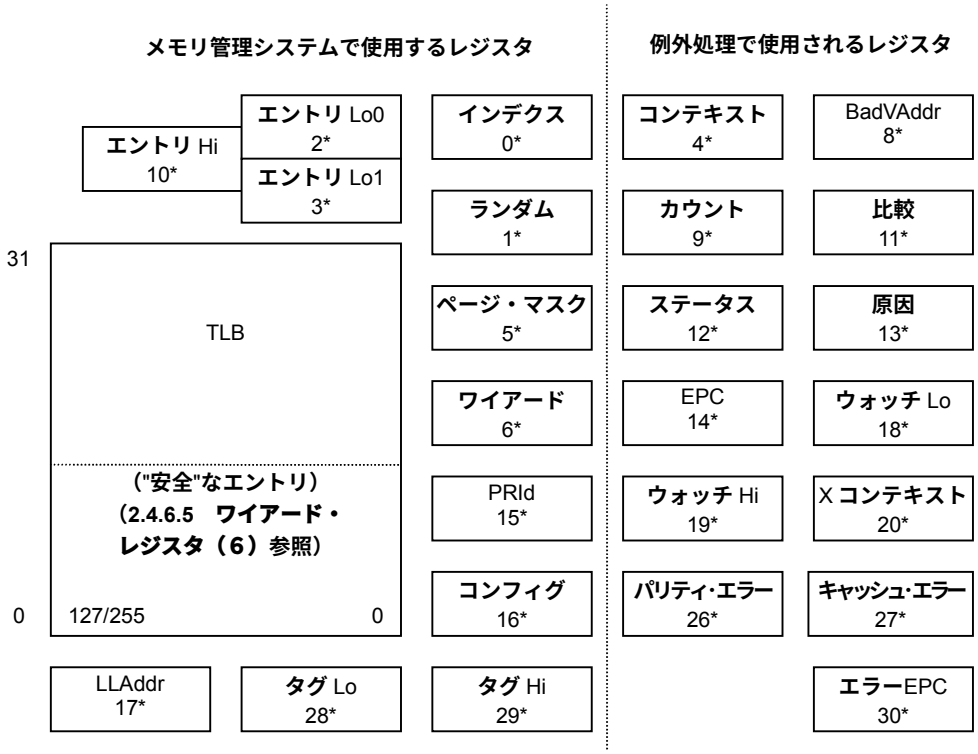
保守／廃止

2.4.5 システム制御コプロセッサ

システム制御コプロセッサ（CP0）は，完全に CPU の一部として動作します。CP0 は，メモリ管理，アドレス変換，例外処理，特権オペレーションをサポートします。CP0 は図 2-36に示すレジスタと 32 エントリの TLB を持っています。この項では，メモリ管理に関連するレジスタについて説明します。

備考　各 CP0 レジスタには，レジスタ番号と呼ばれる番号が付いています。レジスタ番号については，**2.1.5 システム制御コプロセッサ（CP0）**を参照してください。CP0 の機能と，例外処理とレジスタの関係については**2.5　例外処理**を参照してください。

図 2-36 CP0 レジスタと TLB



備考　「*」はレジスタ番号を示します。

注意　**CP0 レジスタにアクセスする際には，使用する命令によっては次の命令実行までの間隔に注意する必要があります。これは，CP0 レジスタの内容が変更されてから CPU の動作に影響が出るまでに時間がかかるためです。これを CP0 ハザードと呼びます。詳細は付録 B　VR4120A コプロセッサ 0 ハザードを参照してください。**

保守／廃止

2.4.6 メモリ管理レジスタ

ここではメモリ管理で使用する CP0 レジスタについて説明します。表 2-32にメモリ管理レジスタの一覧を示します。CP0 のレジスタのうち，この表にないものは例外処理に使用されています（詳細は2.5 例外処理参照）。

表 2-32 CP0 のメモリ管理レジスタ

レジスタ名	レジスタ番号
インデクス・レジスタ	0
ランダム・レジスタ	1
エントリ Lo0 レジスタ	2
エントリ Lo1 レジスタ	3
ページ・マスク・レジスタ	5
ワイアード・レジスタ	6
エントリ Hi レジスタ	10
PRId レジスタ	15
コンフィグ・レジスタ	16
LLAddr レジスタ注	17
タグ Lo レジスタ	28
タグ Hi レジスタ	29

注　VR4000, VR4400™ との互換を保つために定義しています。通常の動作では意味を持ちません。

次に，各レジスタの詳細を説明します。タイトル中のかっこ内の数字はレジスタ番号です。

保守／廃止

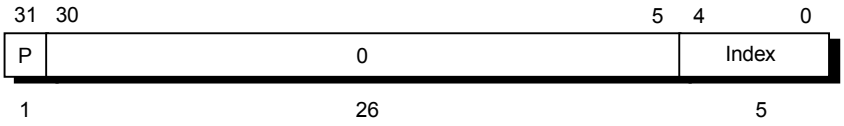
2.4.6.1 インデクス・レジスタ（0）

インデクス・レジスタは，読み書き可能な 32 ビットのレジスタです。そのうち 5 ビットを，TLB エントリのインデクスに使用します。このレジスタの最上位ビットは TLB プローブ（TLBP）命令の成功，失敗を表しています。

このレジスタは，TLB リード（TLBR）命令または TLB ライト・インデクス（TLBWI）命令の対象となる TLB エントリを示しています。

リセット時のインデクス・レジスタの内容は不定となるため，ソフトウエアで初期化してください。

図 2-37 インデクス・レジスタ



P ： プローブの成功／失敗。最後の TLBP 命令が不成功の場合 1 がセットされます。成功の場合 0 にクリアされます。

Index ： TLBR 命令と TLBWI 命令の対象となる TLB エントリへのインデクスを指定します。

0 ： RFU。0 を書き込んでください。読み出すと 0 が返されます。

2.4.6.2 ランダム・レジスタ（1）

ランダム・レジスタは読み出し専用のレジスタで，下位 5 ビットは TLB エントリの参照に使用されます。

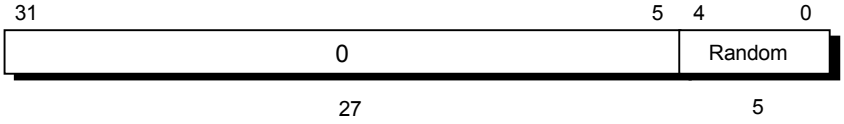
このレジスタは，各命令が実行されるたびにその内容をデクリメントします。このレジスタの値の範囲は次のとおりです。

- 下限はワイアード・レジスタの内容で示します。
- 上限は 31 です。

ランダム・レジスタは，TLBWR 命令の対象となる TLB エントリを示します。また，プロセッサの動作確認のために読み出すこともできます。

ランダム・レジスタは，コールド・リセット時に上限の値にセットされます。また，ワイアード・レジスタに書き込みを行ったときも上限の値にセットされます。図 2-38にランダム・レジスタの形式を示します。

図 2-38 ランダム・レジスタ



Random ： TLB ランダム・インデクス。

0 ： RFU。0 を書き込んでください。読み出すと 0 が返されます。

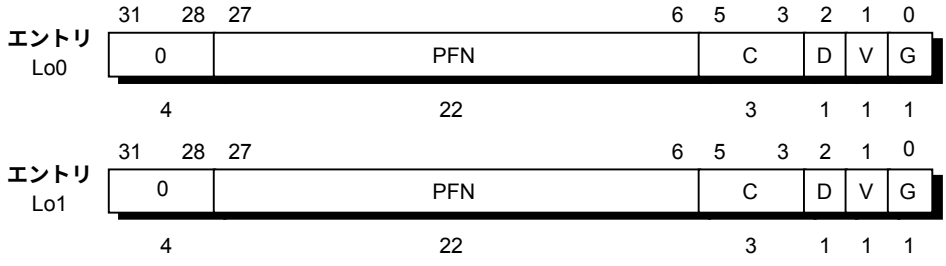
2.4.6.3 エントリ Lo0（2），エントリ Lo1（3）レジスタ

エントリ Lo は，偶数の仮想ページ用のエントリ Lo0 と，奇数の仮想ページ用のエントリ Lo1 の 2 つのレジスタから構成されます。エントリ Lo0 とエントリ Lo1 レジスタは読み書き可能なレジスタで，内蔵 TLB の下位ビットのアクセスに使用します。TLB リード／ライト・オペレーションが実行されると，エントリ Lo0 とエントリ Lo1 は，それぞれ偶数アドレスと奇数アドレスの TLB エントリの下位 32 ビットの内容を保持します。

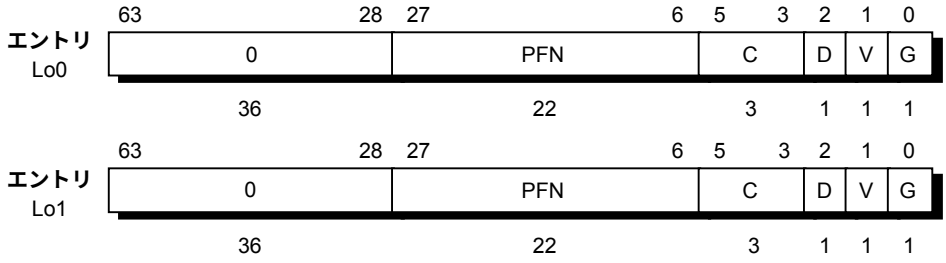
リセット時のこれらのレジスタの内容は不定となるため，ソフトウエアで初期化してください。

図 2-39 エントリ Lo0, Lo1 レジスタ

(a) 32 ビット・モード



(b) 64 ビット・モード



PFN ： ページ・フレーム番号。物理アドレスの上位ビット。

C ： TLB のページ属性を指定します（表 2-33参照）。

D ： Dirty。このビットを 1 にセットするとページは「Dirty」としてマークされ書き込み可能となります。実際このビットは「書き込み保護」ビットとしてデータの変更を防ぐためにソフトウエアが使用します。

V ： Valid。このビットを 1 にセットすると TLB エントリが有効であることを示します。V ビットがセットされないままこのエントリにヒットすると，TLB 無効例外（TLBL または TLBS）が発生します。

G ： Global。エントリ Lo0，エントリ Lo1 の両方のグローバル・ビットがセットされていると，TLB 参照時に ASID は無視されます。

0 ： RFU。0 を書き込んでください。読み出すと 0 が返されます。

ページ参照時にキャッシュを使用するかどうかを，TLB のページ・コヒーレンシ属性（C）ビットで指定します。キャッシュを使用する場合，ページ属性として「キャッシュ使用」か「キャッシュ不使用」のいずれかをアルゴリズムによって選択します。表 2-33に C ビットが選択するページ属性を示します。

保守／廃止

表 2-33 キャッシュ・アルゴリズム

C ビットの値	キャッシュ・アルゴリズム
0	キャッシュ使用
1	キャッシュ使用
2	キャッシュ使用不可
3	キャッシュ使用
4	キャッシュ使用
5	キャッシュ使用
6	キャッシュ使用
7	キャッシュ使用

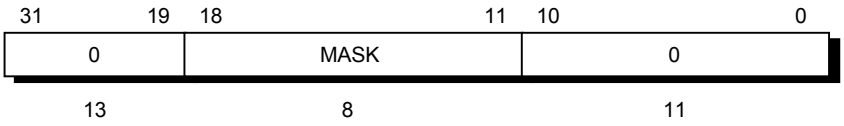
2.4.6.4 ページ・マスク・レジスタ（5）

ページ・マスク・レジスタは，TLB の読み出しや書き込みのための読み書き可能なレジスタです。このレジスタに比較マスク・データを登録すると，表 2-34のように各 TLB エントリのページ・サイズを 1 K バイトから 256 K バイトまで 5 通りに設定できます。

TLB リード／ライト命令は，このレジスタをデスティネーションまたはソースとして使用します。アドレス変換時には，比較対象のビット 18-11 をマスクします。

リセット時のページ・マスク・レジスタの内容は不定となるため，ソフトウエアで初期化してください。

図 2-40 ページ・マスク・レジスタ



MASK： ページ比較マスク。該当するエントリの仮想ページ・サイズを決定します。

0 ： RFU。0 を書き込んでください。読み出すと 0 が返されます。

各ページ・サイズに対するマスク・パターンを表 2-34に示します。マスク・パターンが，これら以外のとき，TLB のオペレーションは不定となります。

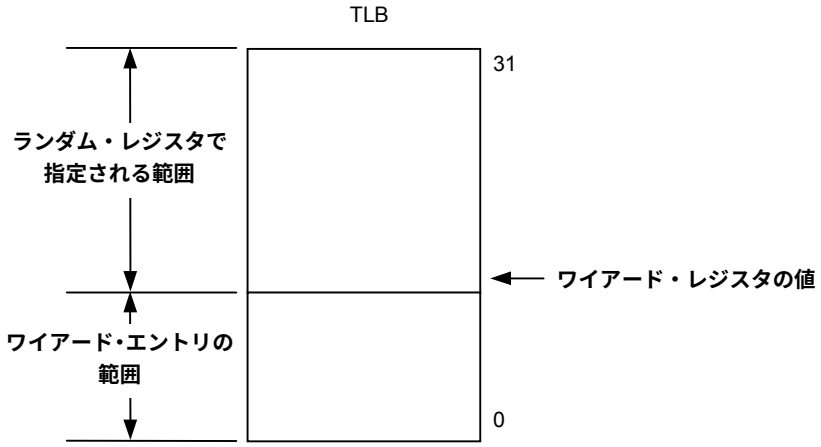
表 2-34 マスク値とページ・サイズ

ページ・サイズ	ビット							
	18	17	16	15	14	13	12	11
1 K バイト	0	0	0	0	0	0	0	0
4 K バイト	0	0	0	0	0	0	1	1
16 K バイト	0	0	0	0	1	1	1	1
64 K バイト	0	0	1	1	1	1	1	1
256 K バイト	1	1	1	1	1	1	1	1

2.4.6.5 ワイアード・レジスタ（6）

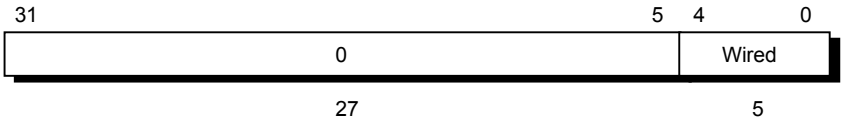
ワイアード・レジスタは読み書き可能なレジスタで，TLB のランダム・エントリの下限を示します（図2-41参照）。ワイアード・エントリは TLBWR 命令では更新できませんが，TLBWI 命令では更新できます。ランダム・エントリはどちらでも更新できます。

図 2-41 ワイアード・レジスタの示す位置



ワイアード・レジスタは，コールド・リセット時，0 にクリアされます。ワイアード・レジスタに書き込みを行うと，ランダム・レジスタは上限値にセットされます（2.4.6.2 **ランダム・レジスタ（1）**参照）。図2-42にワイアード・レジスタの形式を示します。

図 2-42 ワイアード・レジスタ



Wired ： TLB ワイアード境界を指定します。

0 ： RFU。0 を書き込んでください。読み出すと 0 が返されます。

2.4.6.6 エントリ Hi レジスタ（10）

エントリ Hi レジスタは，書き込み可能なレジスタで，内蔵 TLB の上位ビットのアクセスに使用します。

TLB リードやライト・オペレーション実行時，エントリ Hi レジスタは，TLB エントリの上位ビットの内容を保持します。TLB 不一致，TLB 無効，TLB 変更例外が発生した場合，エントリ Hi レジスタは，TLB エントリの上位ビットの内容を保持します。TLB 不一致，TLB 無効，TLB 変更例外が発生した場合，エントリ Hi レジスタには，例外の原因となった仮想アドレスの仮想ページ番号（VPN2）と ASID をセットします。TLB 例外についての詳細は**2.5 例外処理**を参照してください。

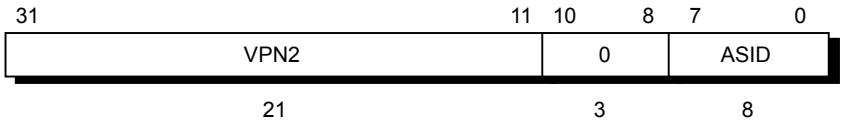
ASID は TLB エントリの ASID 領域の書き込み，読み出しに使用します。またアドレス変換時に，仮想アドレスの ASID として TLB エントリの ASID と照合します。

このレジスタにアクセスするときは，TLBP，TLBWR，TLBWI，TLBR 命令を使用します。

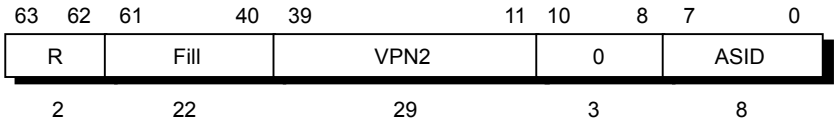
リセット時のエントリ Hi レジスタの内容は不定となるため，ソフトウエアで初期化してください。

図 2-43 エントリ Hi レジスタ

(a) 32 ビット・モード



(b) 64 ビット・モード



VPN2 ： 仮想ページ番号を 2 で割った値（2 ページにマッピングされる）。

ASID ： アドレス空間 ID 領域。8 ビットの ASID 領域は TLB をマルチプロセスで共用できます。各プロセスの仮想アドレスは重なってもかまいません。

R ： 空間種別（00→ユーザ，01→スーパバイザ，11→カーネル）。仮想アドレスのビット 63，62 と一致します。

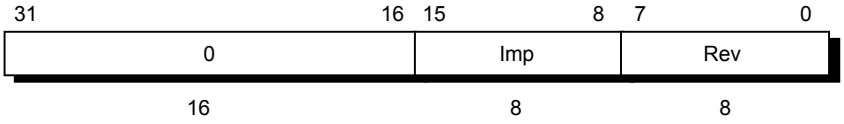
Fill ： RFU。書き込みは無視され，読み出すと 0 が返されます。

0 ： RFU。0 を書き込んでください。読み出すと 0 が返されます。

2.4.6.7 プロセッサ・リビジョン ID（PRId）レジスタ（15）

PRId（プロセッサ・リビジョン ID）レジスタは，32 ビットの読み出し専用のレジスタで，CPU と CP0 の識別子とリビジョン・レベルを示します。図 2-44に PRId レジスタの形式を示します。

図 2-44 PRId レジスタ



Imp ： CPU プロセッサ ID 番号。μPD98502 では 0CH です。

Rev ： CPU プロセッサ・リビジョン番号。

0 ： RFU。0 を書き込んでください。読み出すと 0 が返されます。

プロセッサ・リビジョン番号は，yx 形式の値です。y はビット 7-4 に含まれるメジャー・リビジョン番号で，x はビット 3-0 に含まれるマイナ・リビジョン番号です。

プロセッサ・リビジョン番号は，VR4120A コアの CPU のリビジョンを識別します。しかし，CPU の変更が必ずしも PRId レジスタに反映されているとは限りません。逆にリビジョン番号の変更が，実際の CPU の変更を反映しているとも限りません。したがって，プロセッサ・リビジョン番号領域に依存しないようにプログラムを組んでください。

保守／廃止

2.4.6.8 コンフィグ・レジスタ（16）

このレジスタは，VR4120A コアのさまざまな状態を表示／設定します。

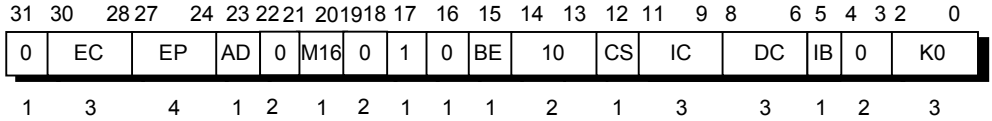
EC 領域と BE 領域は，コールド・リセット時にハードウエアによって設定されます。これらは読み出し専用のビットで，ソフトウエアでアクセスするとそのときの状態をチェックできます。AD，EP，K0 領域は読み書き可能で，ソフトウエアで操作できます。ただし，コールド・リセット直後の値は不定です。

VR4120A コアでは，VR4000 シリーズで使用するオプションの一部しか使用できません。たとえばビット 14 やビット 13 のように，VR4000 シリーズでは変更できても，VR4120A コアでは定数に固定されているものもあります。コンフィグ・レジスタは，キャッシュが使用される前に必ず初期化してください。

図 2-45にコンフィグ・レジスタの形式を示します。

リセット時のコンフィグ・レジスタの内容は不定となるため，ソフトウエアで初期化してください。

図 2-45 コンフィグ・レジスタ（1/2）



EC ： システム・インタフェース・クロック（VTClock）[注]周波数比（読み出し専用）。
0 ～ 6→RFU
7→パイプライン・クロック（PClock）と VTClock の周波数比は 1：1

EP ： ライトバック・データの転送パターンの設定。
0→DD：1 ワード／1 サイクル
その他→RFU：設定禁止

AD ： 高速データ（AD）モードの設定。
0→VR4000 シリーズと互換のデータ・モード
1→RFU：設定禁止

M16 ： MIPS16 ISA モード許可の表示（読み出し専用）。
0→MIPS16 命令実行不可（μPD98502 は MIPS16 モードに対応していないため，必ず 0 が設定されます。）
1→MIPS16 命令実行可能

BE ： BigEndianMem（エンディアンの表示）。
0→リトル・エンディアン
1→ビック・エンディアン

CS ： キャッシュ・サイズ・モードの表示（μPD98502 では 1 に固定）。
0→IC = $2^{(n+12)}$バイト，DC = $2^{(n+12)}$バイト
1→IC = $2^{(n+10)}$バイト，DC = $2^{(n+10)}$バイト

注 VTClock は VR4120A の動作クロック（33 MHz 入力時，CLKSL 端子が“L”なら 100 MHz，“H”なら 66 MHz）。

注意 EP 領域，AD ビットには 0 を必ずセットしてください。それ以外の値をセットした場合，プロセッサの動作は不定です。

図 2-45 コンフィグ・レジスタ（2/2）

IC　：命令キャッシュ・サイズの表示。μPD98502 では $2^{(IC+10)}$バイトとなります。
　4→16 K バイト
　その他→RFU

DC　：データ・キャッシュ・サイズの表示。μPD98502 では $2^{(DC+10)}$バイトとなります。
　3→8 K バイト
　その他→RFU

IB：　リフィル・サイズの設定。μPD98502 は 8 ワード・モードに対応していません。
　0 … 4 ワード（16 バイト）
　1 … 設定禁止（8 ワード（32 バイト））

K0　：kseg0 のコヒーレンシ・アルゴリズムの設定。
　010→キャッシュ使用不可
　その他→キャッシュ使用

1　：読み出すと 1 が返されます。

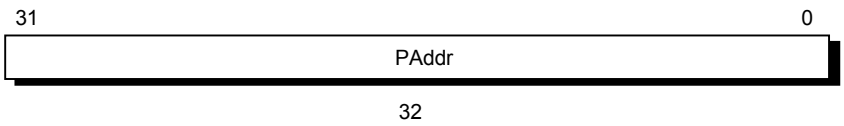
0　：読み出すと 0 が返されます。

2.4.6.9 ロード・リンク・アドレス（LLAddr）レジスタ（17）

LLAddr レジスタは読み書き可能なレジスタで，自己診断目的にのみ使用されます。VR4120A コアでは VR4000，VR4400 との互換を保つために定義されており，通常の動作では意味を持ちません。

リセット時の LLAddr レジスタの内容は不定です。

図 2-46 LLAddr レジスタ



PAddr ：32 ビットの物理アドレス。

保守／廃止

2.4.6.10 キャッシュ・タグ・レジスタ（タグ Lo（28），タグ Hi（29））

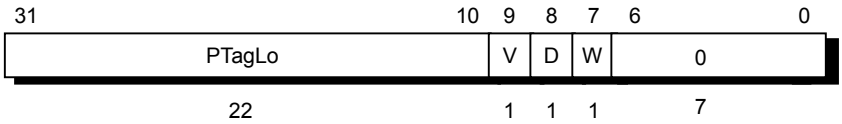
タグ Lo レジスタとタグ Hi レジスタは 32 ビットの読み書き可能なレジスタで，キャッシュの初期化，自己診断，エラー処理のために，キャッシュのキャッシュ・タグを保持します。これらのレジスタは，CACHE 命令か MTC0 命令によって書き込みが行われます。

図 2-47, 図 2-48にこれらのレジスタの形式を示します。

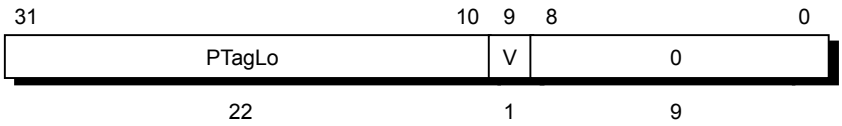
リセット時のこれらのレジスタの内容は不定です。

図 2-47 タグ Lo レジスタ

（a）データ・キャッシュに対して使用する場合



（b）命令キャッシュに対して使用する場合



PTagLo ： 物理アドレスのビット 31-10。

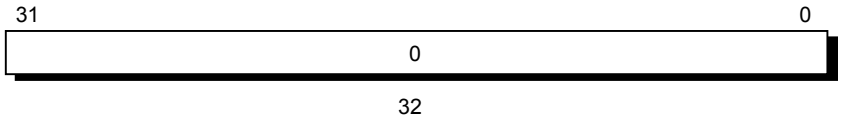
V　　： バリッド・ビット。

D　　： ダーティ・ビット。キャッシュの状態がダーティのときセットされます。ただし，このビットは VR4000 シリーズとの互換を保つために定義されており，読み書き可能ですが，キャッシュ・メモリの状態は表示しません。また，このビットによってキャッシュ・メモリの状態を変更することもできません。

W　　： ライトバック・ビット。キャッシュ・ラインが更新されるとセットされます。

0　　： RFU。0 を書き込んでください。読み出すと 0 が返されます。

図 2-48 タグ Hi レジスタ



0　　： RFU。0 を書き込んでください。読み出すと 0 が返されます。

保守／廃止

2.5 例外処理

この節では，VR4120A コアの例外処理と，そのためのハードウエアについて説明します。

2.5.1 例外処理オペレーション

VR4120A コアがサポートする例外には，TLB ミス，演算オーバフロー，I/O 割り込み，システム・コールなどがあります。例外が発生すると，通常の命令は実行を停止します。プロセッサは現在のモードを抜け，カーネル・モードに移ります（動作モードについては**2.4 メモリ管理システム**を参照してください）。また，プロセッサは割り込みを禁止し，例外ハンドラ（ソフトウエアによる例外処理ルーチンで，特定のアドレスに位置している）に実行を移します。例外ハンドラでは，プログラム・カウンタの内容，現在のオペレーティング・モード（ユーザかスーパバイザ），ステータス，割り込み許可などのプロセッサ状態を保存してください。これらの状態には，例外処理を終了したあとに復帰できます。

例外が発生すると，CPU は例外を処理したあとで実行を再開するアドレスを EPC レジスタにロードします。通常，EPC レジスタには例外を発生した命令のアドレスを再開アドレスとしてロードします。ただし，例外を発生した命令が分岐遅延スロットで実行されていた場合には，分岐遅延スロットの直前のブランチ命令のアドレスを EPC レジスタにロードします。

VR4120A コアはスーパバイザ・モードと，全アドレス空間を対象とする高速 TLB リフィルをサポートしています。また，次のような機能もあります。

- 割り込み許可（IE）
- オペレーティング・モード（ユーザ，スーパバイザ，カーネル）
- 例外レベル（ノーマル／例外を，ステータス・レジスタの EXL ビットで示す）
- エラー・レベル（ノーマル／エラーを，ステータス・レジスタの ERL ビットで示す）

それぞれの設定条件について，次に示します。

(1) 割り込み許可

次の条件がそろったとき割り込みが許可されます。

- IE（割り込み許可ビット） = 1
- EXL ビット = 0，ERL ビット = 0
- ステータス・レジスタ内の対応する IM 領域のビット = 1

(2) オペレーティング・モード

例外レベルとエラー・レベルがノーマル（0）のとき，ステータス・レジスタの KSU ビットによってベースのオペレーティング・モードが指定されます。ステータス・レジスタの EXL ビット，ERL ビットのどちらかが 1 にセットされるとカーネル・モードになります。

(3) 例外／エラー・レベル

例外処理から復帰するときに例外レベルはノーマル（0）にリセットされます（詳細は**付録 A MIPS III 命令セット**を参照してください）。

このほかにも，例外処理中にアドレス，原因，状態の情報を保持するレジスタ群があります。詳細は**2.5.3 例外処理レジスタ**を参照してください。また，例外処理の詳細については**2.5.4 例外の詳細**を参照してください。

2.5.2 例外の正確性

VR4120A コアの例外は論理的に正確に処理されます。例外の原因となった命令と，その後の命令はアボートされ，例外処理が済んだあとに再実行されます。あとに続く命令が破棄されると，それらの命令に連動した例外もすべて破棄されます。例外の発生順序は，検出された順ではなく，命令のフェッチ順です。

例外ハンドラは例外の原因とアドレスを特定することができます。プログラムはデスティネーション・レジスタを書き直すことで，自動的にではありませんが，再スタートできます。これは，ステータスの変化がおきないその他すべての正確な例外の場合も同様です。

2.5.3 例外処理レジスタ

ここでは，例外処理で使用する CP0 レジスタについて説明します。表 2-35に CP0 例外処理レジスタの一覧を示します。例外処理レジスタそれぞれに固有の番号が付いており，その番号をレジスタ番号と呼びます。全 CP0 レジスタのうち表中にないものは，メモリ管理に使用されています（詳細は**2.4 メモリ管理システム**参照）。

例外ハンドラは CP0 レジスタの内容を調べ，発生した例外の原因やそのときの CPU の状態を判断します。表 2-35に記載したレジスタは例外処理時に使用されます。その詳細を以降に記述します。

表 2-35 CP0 の例外処理レジスタ

レジスタ名	レジスタ番号
コンテキスト・レジスタ	4
BadVAddr レジスタ	8
カウント・レジスタ	9
比較レジスタ	11
ステータス・レジスタ	12
原因レジスタ	13
EPC レジスタ	14
ウォッチ Lo レジスタ	18
ウォッチ Hi レジスタ	19
X コンテキスト・レジスタ	20
パリティ・エラー・レジスタ注	26
キャッシュ・エラー・レジスタ注	27
エラーEPC レジスタ	30

注 このレジスタは VR4100 との互換性を保つために定義しています。μPD98502 のハードウエアでは使用しません。

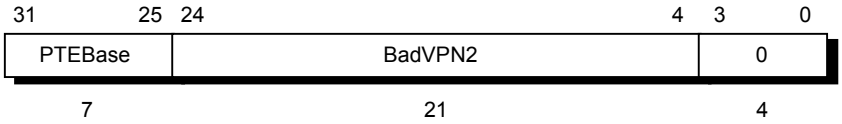
2.5.3.1 コンテキスト・レジスタ（4）

コンテキスト・レジスタは読み書き可能なレジスタで，メモリにあるページ・テーブル・エントリ（PTE）アレイへのポインタを示します。アレイには仮想アドレスから物理アドレスへの変換テーブルをストアします。TLB ミスが発生すると，オペレーティング・システムは，変換できなかったエントリを PTE から TLB にロードします。コンテキスト・レジスタは，TLB エントリをロードする TLB 不一致例外ハンドラで使用されます。

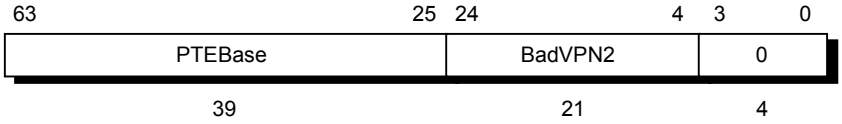
コンテキスト・レジスタは，BadVAddr レジスタといくつか重複する情報を含んでいますが，TLB 例外ハンドラに使いやすい形式になっています。図 2-49にコンテキスト・レジスタの形式を示します。

図 2-49 コンテキスト・レジスタ

（a）32 ビット・モード



（b）64 ビット・モード



PTEBase ：ページ・エントリ・テーブルのベース・アドレス。

BadVPN2 ：変換が無効となった最後の仮想アドレスの仮想ページ番号を 2 で割った値。

0 ：RFU。0 を書き込んでください。読み出すと 0 が返されます。

PTEBase 領域は，現在のユーザ・アドレス空間の PTE テーブルのベース・アドレスへのポインタとしてソフトウエアが使用します。

21 ビットの BadVPN2 領域には，TLB ミスの原因となった仮想アドレスのビット 31-11 の値が保持されます。1 つの TLB エントリは偶数アドレスと奇数アドレスがペアになっているため，ビット 10 は無視されます。このレジスタ形式は，ページ・サイズが 1 K バイトなら 8 バイトの PTE ペア・テーブルを参照するポインタとして，そのまま使用できます。4 K バイト以上のページ・サイズでは，この値をシフトしたりマスクすることによって適切な PTE 参照アドレスを生成できます。

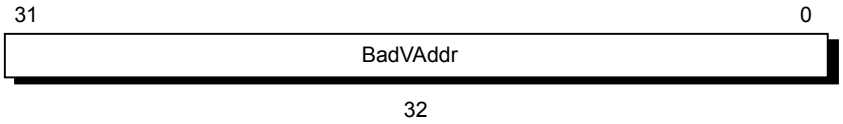
2.5.3.2 BadVAddr レジスタ（8）

BadVAddr（Bad Virtual Address）レジスタは読み出し専用のレジスタで，最後に無効な変換を行った仮想アドレス，またはアドレシング・エラーがあった仮想アドレスが保持されます。図 2-50に，BadVAddr レジスタの形式を示します。

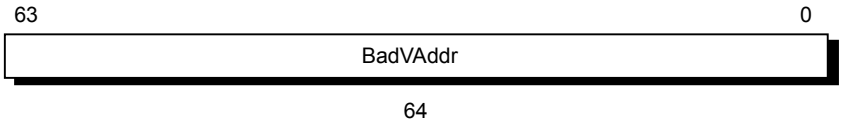
注意　バス・エラー例外が発生してもアドレス・エラー例外ではないため，このレジスタには情報が保存されません。

図 2-50 BadVAddr レジスタ

（a）32 ビット・モード



（b）64 ビット・モード



BadVAddr ：最後にアドレス変換を失敗したか，アドレシング・エラーがあった仮想アドレス。

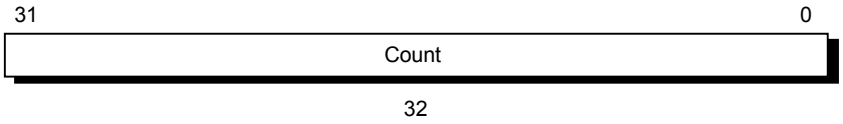
2.5.3.3 カウント・レジスタ（9）

カウント・レジスタは読み書き可能なレジスタで，タイマとして動作します。カウント・レジスタの内容は，命令の実行や中断，先行するほかの命令が確実にパイプラインを通ったかにかかわらず，MasterOut の周波数に同期してインクリメントされます。

このレジスタはフリー・ランニング・タイプです。すべてのビットが 1 になると，次のイベントですべて 0 に戻り，インクリメントを続けます。自己診断やシステムの初期化，あるいはプロセス間の同期を取るために使用されます。

図 2-51にカウント・レジスタの形式を示します。

図 2-51 カウント・レジスタ



Count　：最新のカウント値。比較レジスタの内容と比較します。

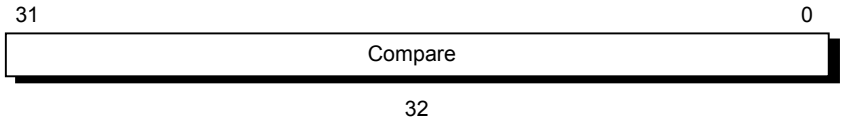
2.5.3.4 比較レジスタ（11）

比較レジスタは，タイマ割り込みを発生するためのレジスタで，値を保持しますが，単独で値を変えることはありません。比較レジスタの値とカウント・レジスタ（2.5.3.3 **カウント・レジスタ（9）**参照）の値が等しくなった場合，原因レジスタの IP7 ビットがセットされます。IP7 ビットがセットされると，ただちに割り込みが発生します。比較レジスタに値を書き込むと，タイマ割り込み要求はクリアされます。

比較レジスタは，自己診断に使用するため読み書きが可能ですが，通常は書き込み用としてのみ使用します。図 2-52に比較レジスタの形式を示します。

リセット時の比較レジスタの内容は不定です。

図 2-52 比較レジスタ



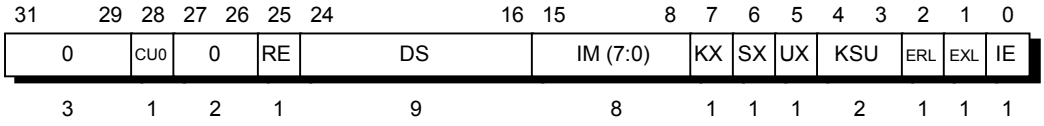
Compare ：カウント・レジスタの内容と比較する値。

保守／廃止

2.5.3.5 ステータス・レジスタ（12）

ステータス・レジスタは読み書き可能なレジスタで，オペレーティング・モード，割り込み許可，プロセッサの自己診断などの状態情報を保持しています。図 2-53に，ステータス・レジスタの形式を示します。

図 2-53 ステータス・レジスタ（1/2）



CU0 ： コプロセッサの使用許可（1→使用可，0→使用不可）。
カーネル・モードでは，CU0 ビットにかかわらず CP0 は常に使用可能です。

RE ： ユーザ・モードにおけるエンディアン反転の許可（0→禁止，1→反転）。

注意 RE ビットは 0 に設定してください。

DS ： 自己診断ステータス領域（詳細は図 2-54参照）。

IM(7:0) ： 割り込みマスク。外部，内部およびソフトウエア割り込みの許可（0→禁止，1→許可）。
8 ビットで構成され，8 つの割り込みを制御します。割り込みは各ビットに次のように割り当てられています。

IM7 ： タイマ割り込みのマスク
IM(6:2) ： 通常割り込み（Int（4:0）[注]）のマスク
IM(1:0) ： ソフトウエア割り込みのマスク

注 Int(4:0) はμPD98502 の内部信号（各周辺ブロックからの割り込み信号）です。μPD98502 の内蔵周辺ユニットとの接続については，**1.11 割り込み**，および**2.8 VR4120A コアの割り込み**を参照してください。

KX ： カーネル・モードでの 64 ビット・アドレシングの許可（0→32 ビット，1→64 ビット）。このビットがセットされていた場合，カーネル・モード・アドレス空間での TLB ミス時，XTLB 不一致例外が発生します。
なお，カーネル・モードでは，64 ビット・オペレーションは常に有効です。

SX ： スーパバイザ・モードでの 64 ビット・アドレシングと 64 ビット・オペレーションの許可（0→32 ビット，1→64 ビット）。
このビットがセットされていた場合，スーパバイザ・モード・アドレス空間での TLB ミス時，XTLB 不一致例外が発生します。

UX ： ユーザ・モードでの 64 ビット・アドレシングと 64 ビット・オペレーションの許可（0→32 ビット，1→64 ビット）。
このビットがセットされていた場合，ユーザ・モード・アドレス空間での TLB ミス時，XTLB 不一致例外が発生します。

図 2-53 ステータス・レジスタ（2/2）

KSU ： オペレーティング・モードの設定／表示（10→ユーザ，01→スーパバイザ，00→カーネル）。
ERL ： エラー・レベルの設定／表示（0→ノーマル，1→エラー）。
EXL ： 例外レベルの設定／表示（0→ノーマル，1→例外）。
IE ： 割り込み許可の設定／表示（0→禁止，1→許可）。
0 ： RFU。0 を書き込んでください。読み出すと 0 が返されます。

図 2-54に，自己診断（DS）領域の詳細を示します。DS 領域のビットはすべて，読み書き可能です。

図 2-54 自己診断ステータス領域

24	23	22	21	20	19	18	17	16
DME	0	BEV	0	SR	0	CH	CE	DE
1	1	1	1	1	1	1	1	1

DME ： ディバッグ・モード・イネーブル（0→禁止，1→許可）
BEV ： TLB 不一致例外ベクタと汎用例外ベクタのベース・アドレスの指定（0→通常，1→ブートストラップ）。
SR ： ソフト・リセットまたは NMI の発生（0→未発生，1→発生）。
CH ： CP0 の条件ビット（0→偽，1→真）。ソフトウエアでだけ読み書き可能で，ハードウエアではアクセスできません。
CE, DE ： VR4100 との互換を保つために定義しています。VR4120A コアのハードウエアでは使用しません。
0 ： RFU。0 を書き込んでください。読み出すと 0 が返されます。

ステータス・レジスタの内容は，コールド・リセット後，自己診断テータス領域の次のビットを除いては不定になります。

- ERL ビット = 1
- BEV ビット = 1
- SR ビット = 0（コールド・リセット時），または 1（ソフト・リセット，または NMI 割り込み時）

ステータス・レジスタの中には，次に説明するように，モードやアクセス状態を設定する領域があります。

(1) 割り込み許可

次の条件を満たしたとき，割り込みは許可されます。

- IE が 1 にセット
- EXL が 0 にクリア
- ERL が 0 にクリア
- IM の該当ビットが 1 にセット

(2) オペレーティング・モード

次に示すステータス・レジスタの設定で，ユーザ，カーネル，スーパバイザ・モードへ移行します。

- KSU が 10，EXL が 0，ERL が 0 の場合，ユーザ・モードです。
- KSU が 01，EXL が 0，ERL が 0 の場合，スーパバイザ・モードです。
- KSU が 00 か，EXL が 1 か，ERL が 1 の場合，カーネル・モードです。

(3) 32 ビット・モードと 64 ビット・モード

VR4120A コアは，ステータス・レジスタ内の次に示すビットにより，ユーザ，スーパバイザ，カーネル・モードごとに 32 ビット・オペレーションか 64 ビット・オペレーションを選択します。64 ビット・モードでは 64 ビットのデータを処理し，64 ビット・アドレスの変換を行います。64 ビット・オペレーションは，モードごとに個別に設定できます。

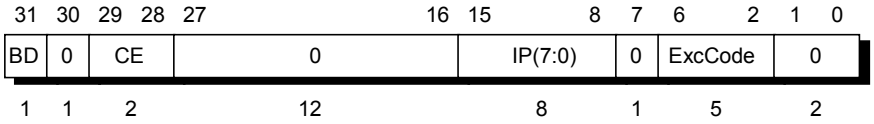
- KX ビットが 1 にセットされている場合，カーネル・モードでは 64 ビットのアドレシングを行います。なおカーネル・モードでは，64 ビット・オペレーションは常に有効です。
- SX ビットが 1 にセットされている場合，スーパバイザ・モードでは 64 ビットのアドレシングを行い，64 ビット・オペレーションが有効になります。
- UX ビットが 1 にセットされている場合，ユーザ・モードでは 64 ビットのアドレシングを行い，64 ビット・オペレーションが有効になります。

保守／廃止

2.5.3.6 原因レジスタ（13）

原因レジスタは，32 ビットの読み書き可能なレジスタで，最後に起きた例外の原因を保持します。例外コード領域の 5 ビットは例外の原因を示します（表 2-36参照）。残りの領域は，特定の例外に対する詳しい情報を保持します。IP1-IP0 ビット以外のすべてのビットは読み出し専用です。IP1-IP0 ビットは，ソフトウエアの割り込みの発生に使用されます。図 2-55に原因レジスタの形式，表 2-36に例外コードの説明を示します。

図 2-55 原因レジスタ



BD ：最後に発生した例外が分岐遅延スロット内で実行されたかどうかを示します（1→遅延スロット内，0→通常）。

CE ：コプロセッサ使用不可例外が発生したコプロセッサ番号を示します。この例外が発生していないときは不定となります。

IP(7:0) ：ペンディングされている割り込みを示します（1→保留中，0→割り込みなし）。
割り込みは各ビットに次のように割り当てられています。

- IP7 ： タイマ割り込み
- IP(6:2) ： 通常割り込み（Int(4:0)注）
- IP(1:0) ： ソフトウエア割り込み。このビットだけは，ソフトウエアで 1 にセットすると割り込み例外を発生します。

注 Int(4:0) はμPD98502 の内部信号（各周辺ブロックからの割り込み信号）です。μPD98502 の内蔵周辺ユニットとの接続については，**1.11 割り込み**，および**2.8 VR4120A コアの割り込み**を参照してください。

ExcCode ：例外コード領域（詳細は表 2-36参照）。

0 ：RFU。0 を書き込んでください。読み出すと 0 が返されます。

保守／廃止

表 2-36 原因レジスタの例外コード領域

例外コード	ニモニック	説　　明
0	Int	割り込み例外
1	Mod	TLB 変更例外
2	TLBL	TLB 不一致例外（ロードまたはフェッチ）
3	TLBS	TLB 不一致例外（ストア）
4	AdEL	アドレス・エラー例外（ロードまたはフェッチ）
5	AdES	アドレス・エラー例外（ストア）
6	IBE	バス・エラー例外（命令フェッチ）
7	DBE	バス・エラー例外（データのロードまたはストア）
8	Sys	システム・コール例外
9	Bp	ブレークポイント例外
10	RI	予約命令例外
11	CpU	コプロセッサ使用不可例外
12	Ov	演算オーバフロー例外
13	Tr	トラップ例外
14-22	ー	RFU
23	WATCH	ウォッチ例外
24-31	ー	RFU

The VR4120A コアには，8 つの割り込み要求，IP7-IP0 があり，次の目的で使用されます。なお，割り込みの詳細については，**2.8　VR4120A コアの割り込み**を参照してください。

（1）IP7

タイマ割り込み要求の有無を示し，カウント・レジスタと比較レジスタの内容が等しくなったとき，セットされます。

（2）IP6 to IP2

IP6-IP2 には，VR4120A コアへの割り込み要求信号の状態が反映されます。

（3）IP1 and IP0

IP1，IP0 は，各ビットを操作することにより，ソフトウエア割り込み要求の設定，およびクリアを行います。

2.5.3.7 例外プログラム・カウンタ（EPC）レジスタ（14）

例外プログラム・カウンタ（EPC）レジスタは読み書き可能なレジスタで，例外処理後のプログラムの再開アドレスを示します。μPD98502 が MIPS16 命令モードに対応していないため，EPC レジスタは次のどちらかを示します。

- 例外の直接の原因となった命令の仮想アドレス
- 直前のブランチやジャンプ命令の仮想アドレス（例外の原因となった命令が分岐遅延スロットにある場合。原因レジスタの BD ビットも同時に 1 にセットされる）

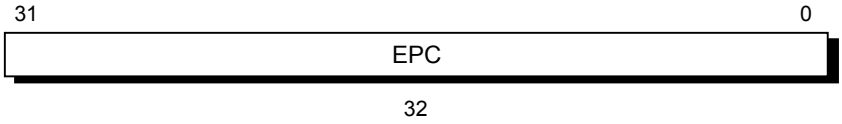
16 ビット命令実行時，EPC レジスタは次のどちらかを示します。

ステータス・レジスタの EXL ビットが 1 にセットされていると EPC レジスタの内容は変化しません。これは EPC レジスタに格納されている例外の原因となった命令のアドレスを，ほかの例外の発生時にプロセッサが上書きすることを防ぐためです。

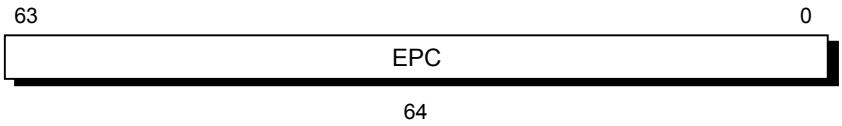
図 2-56に EPC レジスタの形式を示します。

図 2-56 EPC レジスタ

（a）32 ビット・モード



（b）64 ビット・モード



EPC　：例外処理後のプログラムが再開されるアドレス。

保守／廃止

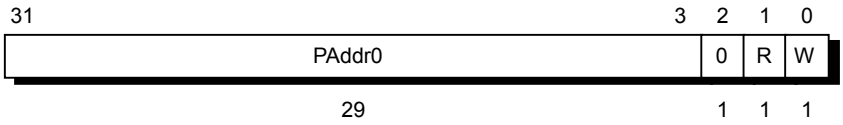
2.5.3.8 ウォッチ Lo レジスタ（18）とウォッチ Hi レジスタ（19）

VR4120A コアは，ウォッチ Lo レジスタとウォッチ Hi レジスタによって示される物理アドレスのデータに対して参照の要求があったことを検知します。ロード／ストア命令を実行すると，ウォッチ例外を発生するディバグ機能を持っています。

図 2-57, 図 2-58に，ウォッチ Lo レジスタとウォッチ Hi レジスタの形式を示します。

リセット時のこれらのレジスタの内容は不定となるため，ソフトウエアで初期化してください。

図 2-57 ウォッチ Lo レジスタ



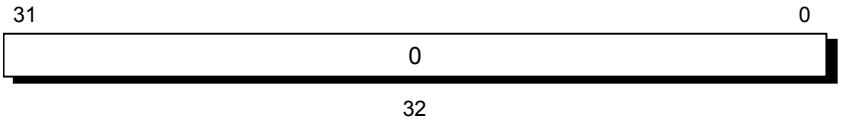
PAddr0 ：物理アドレスのビット 31 - 3 を指定します。

R ：このビットが 1 の場合，ロード命令実行時に例外が発生します。

W ：このビットが 1 の場合，ストア命令実行時に例外が発生します。

0 ：RFU。0 を書き込んでください。読み出すと 0 が返されます。

図 2-58 ウォッチ Hi レジスタ



0 ：RFU。0 を書き込んでください。読み出すと 0 が返されます。

2.5.3.9 X コンテキスト・レジスタ（20）

X コンテキスト・レジスタは読み書き可能なレジスタで，メモリにあるページ・テーブル・エントリ（PTE）アレイのうちの 1 つのエントリを示します。ここで PTE アレイとは，オペレーティング・システムのデータ構造を意味し，仮想アドレスから物理アドレスへの変換テーブルを保存します。TLB ミスが発生すると，オペレーティング・システムは PTE から TLB に変換ミスしたデータをロードして，ソフトウエア処理によってミスに対処します。

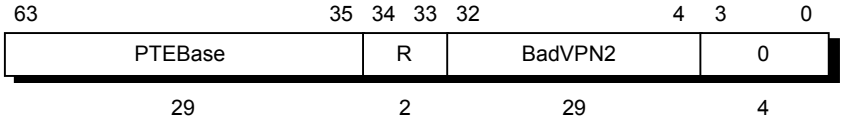
X コンテキスト・レジスタは，64 ビット・アドレシング・モードにおいて，TLB エントリをロードする XTLB 不一致例外ハンドラで使用されます。

X コンテキスト・レジスタは，BadVAddr レジスタといくつか重複する情報を含んでいますが，XTLB 例外ハンドラに使いやすい形式になっています。

このレジスタは，オペレーティング・システムだけが使用し，必要に応じてこのレジスタの PTEBase 領域をセットします。

図 2-59に，X コンテキスト・レジスタの形式を示します。

図 2-59 X コンテキスト・レジスタ



PTEBase：ページ・テーブル・エントリのベース・アドレス。

R　　：空間種別（00→ユーザ，01→スーパバイザ，11→カーネル）。仮想アドレスのビット 63，62 と一致します。

BadVPN2：変換が無効となった最後の仮想アドレスの仮想ページ番号を 2 で割った値（VPN2）。

0　　：RFU。0 を書き込んでください。読み出すと 0 が返されます。

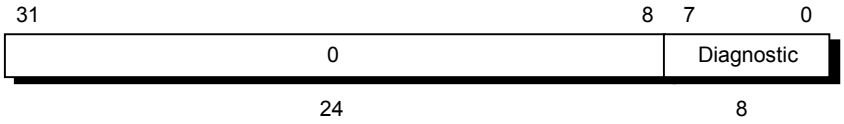
29 ビットの BadVPN2 領域は，TLB 不一致を引き起こした仮想アドレスのビット 39-11 の値を保持します。TLB エントリは奇数ページと偶数ページがペアになっているため，ビット 10 は含みません。このレジスタの形式は，ページ・サイズが 1 K バイトでは，8 バイトの PTE ペア・テーブルを参照するポインタとして，そのまま使用できます。4 K バイト以上のページ・サイズでは，この値をシフトあるいはマスクすることによって適切な PTE 参照アドレスを生成できます。

2.5.3.10 パリティ・エラー（PErr）レジスタ（26）

パリティ・エラー・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 VR4100 とソフトウエア上の互換を保つために定義されています。VR4120A コアではパリティがないためハードウエアでは使用しません。

図 2-60に，パリティ・エラー・レジスタの形式を示します。

図 2-60 パリティ・エラー・レジスタ



Diagnostic ： 8 ビットの自己診断領域

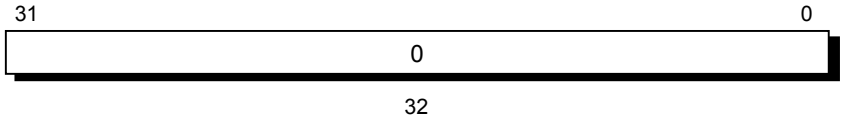
0　　： RFU。0 を書き込んでください。読み出すと 0 が返されます。

2.5.3.11 キャッシュ・エラー・レジスタ（27）

キャッシュ・エラー・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 VR4100 とソフトウエア上の互換を保つために定義しています。VR4120A コアではパリティがないためハードウエアでは使用しません。

図 2-61にキャッシュ・エラー・レジスタの形式を示します。

図 2-61 キャッシュ・エラー・レジスタ



0 ： RFU。0 を書き込んでください。読み出すと 0 が返されます。

2.5.3.12 エラーEPC レジスタ（30）

エラーEPC（例外プログラム・カウンタ）レジスタは，EPC レジスタとほとんど同じです。ただし，このレジスタはコールド・リセット，ソフト・リセット，NMI 例外時のプログラム・カウンタのストアに使用されます。エラーEPC レジスタは読み書き可能なレジスタで，エラー処理終了後，命令の実行が再開する次のどちらかの仮想アドレスを保持しています。

- 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にある場合，直前のブランチ命令やジャンプ命令の仮想アドレス

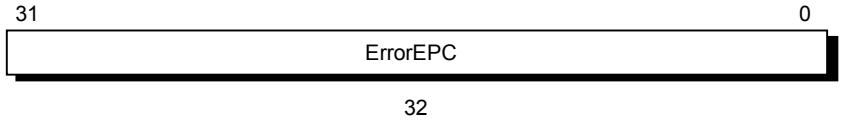
ステータス・レジスタの ERL ビットが 1 にセットされていると，エラーEPC レジスタの内容は変化しません。これにより，エラーEPC レジスタに格納されているエラー例外の原因となった命令のアドレスを，ほかの例外の発生時にプロセッサが上書きするのを防ぎます。

エラーEPC レジスタが分岐遅延スロットを示すことはありません。

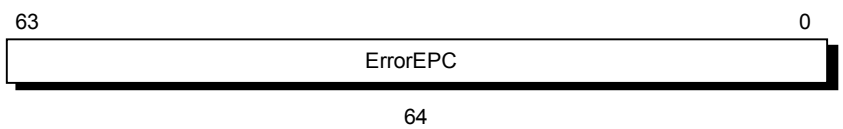
図 2-62にエラーEPC レジスタの形式を示します。

図 2-62 エラーEPC レジスタ

（a）32 ビット・モード



（b）64 ビット・モード



ErrorEPC ： コールド・リセット，ソフト・リセット，NMI の各例外時のプログラム・カウンタを示します。

保守／廃止

2.5.4 例外の詳細

この項では，VR4120A コアの各例外（原因，処理，操作）について説明します 。

2.5.4.1 Exception types

ここでは，次の例外の種類における例外ハンドラの動作の一例を述べます。

- コールド・リセット
- ソフト・リセット（サポートしていません）
- NMI
- その他のプロセッサ例外

ステータス・レジスタの EXL と ERL の両ビットが 0 であれば，ステータス・レジスタの KSU ビットの値によってユーザ・モードか，スーパバイザ・モードか，カーネル・モードのどれかが選択されます。EXL ビットと ERL ビットのどちらかが 1 の場合，プロセッサはカーネル・モードとなります。

プロセッサに例外が発生すると，EXL ビットが 1 にセットされ，システムがカーネル・モードに移行します。情報を退避させたあと，例外ハンドラは通常 EXL ビットを 0 にリセットします。また，退避した情報を復帰させている間にほかの例外によって失われないように，例外ハンドラは EXL ビットを 1 に再設定します。

例外処理を抜けると，EXL ビットは 0 にリセットされます。詳細は**付録 A　MIPS III 命令セットの** ERET 命令を参照してください。

2.5.4.2 例外ベクタ・アドレス

コールド・リセット，ソフト・リセット，NMI の各例外が発生すると，常に次に示すリセット例外ベクタ・アドレスに分岐します。このアドレスは非キャッシュ，非マッピング領域です。

- 32 ビット・モード時 : BFC0_0000H（仮想アドレス）
- 64 ビット・モード時 : FFFF_FFFF_BFC0_0000H（仮想アドレス）

その他の例外が発生したときのベクタ・アドレスは，ベクタ・オフセットにベース・アドレスを加えた値になります。

64 ビット・モードと 32 ビット・モード時の例外ベクタとそのオフセット値を次に示します。

表 2-37　64 ビット・モードの例外ベクタのベース・アドレス

	ベクタ・ベース・アドレス（仮想）	ベクタ・オフセット
コールド・リセット ソフト・リセット NMI	FFFF_FFFF_BFC0_0000H （BEV ビットは自動的に 1 にセットされます）	0000H
TLB 不一致 （EXL = 0）	FFFF_FFFF_8000_0000H（BEV = 0） FFFF_FFFF_BFC0_0200H（BEV = 1）	0000H
XTLB 不一致（EXL = 0）		0080H
その他の例外		0180H

表 2-38 32 ビット・モードの例外ベクタのベース・アドレス

	ベクタ・ベース・アドレス（仮想）	ベクタ・オフセット
コールド・リセット ソフト・リセット NMI	BFC0_0000H （BEV ビットは自動的に 1 にセットされます）	0000H
TLB 不一致 （EXL = 0）	8000_0000H（BEV = 0） BFC0_0200H（BEV = 1）	0000H
XTLB 不一致（EXL = 0）		0080H
その他の例外		0180H

（1）TLB 不一致ベクタの場合

BEV ビットが 0 のとき，この例外用のベクタのベース・アドレス（仮想）は，次のように kseg0（非マッピング）領域になります。

- 32 ビット・モード時 : 8000_0000H
- 64 ビット・モード時 : FFFF_FFFF_8000_0000H

BEV ビットが 1 のとき，この例外用のベクタのベース・アドレス（仮想）は，次のように kseg1（非キャッシュ，非マッピング）領域になります。

- 32 ビット・モード時 : BFC0_0200H
- 64 ビット・モード時 : FFFF_FFFF_BFC0_0200H

この空間は非キャッシュ領域であり非 TLB マッピングの空間ですから，例外ハンドラはキャッシュと TLB を使用しません。

2.5.4.3 例外の優先順位

1 つの命令に対して複数の例外が同時に発生したとき，その中の 1 つだけが選択されます。優先順位は表 7-5 のとおりです。

表 2-39 例外優先順位

優先度	例　外
高い	コールド・リセット
↑	ソフト・リセット（サポートしていません）
	ディバグ（端子）
	ディバグ（命令アドレス・ブレーク）
	NMI
	アドレス・エラー（命令フェッチ）
	TLB/XTLB 不一致（命令フェッチ）
	TLB 無効（命令フェッチ）
	割り込み（NMI 以外）
	バス・エラー（命令フェッチ）
	ディバグ（DBREAK 命令）
	システム・コール
	ブレーク・ポイント
	コプロセッサ使用不可
	予約命令
	トラップ
	整数オーバフロー
	ディバグ（データ・アドレス・ブレーク）
	アドレス・エラー（データ・アクセス）
	TLB/XTLB 不一致（データ・アクセス）
	TLB 無効（データ・アクセス）
	TLB 変更（データ書き込み）
	ウォッチ
	バス・エラー（データ・アクセス）
低い	ディバグ（データ・データ・ブレーク）注

注 データ・アクセス時にアドレスとデータの両方の条件が一致した場合も含みます。

以降，例外処理のうちハードウエアで扱う部分を「処理」，ソフトウエアで扱う部分を「操作」と呼びます。

2.5.4.4 コールド・リセット例外

(1) 原　　因

RSTB_B 端子入力をアクティブにすると，ColdReset_B 信号（内部）をアクティブにしたあとインアクティブにし，コールド・リセット例外が発生します。この例外はマスク不可能です（詳細は**2.6　初期化インタフェース**を参照）。

(2) 処　　理

リセット例外用の特殊割り込みベクタを使用します。

- 32 ビット・モード時 : BFC0_0000H（仮想アドレス）
- 64 ビット・モード時 : FFFF_FFFF_BFC0_0000H（仮想アドレス）

リセット・ベクタは，非マッピング，非キャッシュ領域にあります。このため，リセット例外は TLB の初期化やキャッシュの操作を必要としません。VR4120A コアは，キャッシュや仮想メモリが定義されていない状態でも，命令を読み込み，実行することができます。
この例外が発生すると，次のレジスタを除くすべての VR4120A コアのレジスタの内容は不定となります。

- ステータス・レジスタの ERL ビットが 0 の場合，エラーEPC レジスタには例外発生時のプログラム・カウンタの値がセットされます。
- ステータス・レジスタの SR ビットがクリア（0）されます。
- ステータス・レジスタの ERL, BEV ビットがセット（1）されます。
- ランダム・レジスタに上限値（31）がセットされます。
- ワイアード・レジスタが 0 に初期化されます。
- コンフィグ・レジスタのビット 31-28, 22-3 が初期化されます。

(3) 操　　作

リセット例外時，次の操作を行ってください。

- すべての VR4120A コアのレジスタ，コプロセッサ・レジスタ，TLB，キャッシュ，メモリ・システムの初期化
- 自己診断
- オペレーティング・システムのブートストラップ

保守／廃止

2.5.4.5 ソフト・リセット例外

注意 μPD98502 **ではソフト・リセットはサポートしていません。**

(1) 原　　因

ソフト・リセットは，ウォーム・リセットとも呼ばれます。ColdReset_B **信号（内部）がインアクティブの状態で** HOTRST#**信号（内部）をアクティブにしてからインアクティブにすると，ソフト・リセット例外が発生します（詳細は2.6　初期化インタフェースを参照）。**
この例外が発生すると，すべてのステート・マシンがリセットされ，ステータス・レジスタの SR **ビットがセットされます。また，**VR4120A **コアはリセット・ベクタから実行を開始します。**
この例外はマスク不可能です。

(2) 処　　理

特殊割り込みベクタを使用します（コールド・リセットと同じ）。

- 32 **ビット・モード時** : BFC0_0000H**（仮想アドレス）**
- 64 **ビット・モード時** : FFFF_FFFF_BFC0_0000H**（仮想アドレス）**

このベクタは非マッピング，非キャッシュ領域にあります。このため，ソフト・リセット例外は，TLB **の初期化やキャッシュの操作を必要としません。また，ステータス・レジスタの** SR **ビットが** 1 **にセットされ，コールド・リセット例外と区別されます。**
この例外が発生したとき，次のレジスタを除いて，すべてのレジスタの内容は保存されます。

- **エラー**EPC **レジスタには，例外発生時のプログラム・カウンタの値がセットされます。**
- **ステータス・レジスタの** ERL，SR，BEV **ビットは** 1 **にセットされます。**

ソフト・リセットでは，キャッシュやシステム・インタフェースへのアクセスがアボートされることがあります。そのため，この例外が発生したときのキャッシュとメモリの状態は不定となります。

(3) 操　　作

ソフト・リセット例外時，次の操作を行ってください。

- **自己診断のために現在のプロセッサの状態を保存**
- **コールド・リセット例外と同じようにシステムを再初期化**

2.5.4.6 NMI 例外

(1) 原　　因

ノンマスカブル割り込み（NMI）例外は，NMI 信号（内部）がアクティブになると発生します。この割り込みはマスク不可能です。したがって，この割り込みはステータス・レジスタの EXL，ERL，IE ビットにかかわらず発生します（詳細は2.8　VR4120A コアの割り込みを参照）。

(2) 処　　理

NMI 例外は次に示す特殊割り込みベクタを使用します。

- 32 ビット・モード時 : BFC0_0000H（仮想アドレス）
- 64 ビット・モード時 : FFFF_FFFF_BFC0_0000H（仮想アドレス）

このベクタは非マッピング，非キャッシュ領域にあります。このため，NMI 例外は TLB の初期化やキャッシュの操作を必要としません。また，ステータス・レジスタの SR ビットが 1 にセットされ，コールド・リセット例外と区別されます。
コールド・リセットやソフト・リセットと違い，他の例外と同様に命令と命令の間でのみ発生します。したがって，キャッシュとメモリの状態は保存されます。
この例外が発生したとき，次のレジスタを除くすべてのレジスタの内容は保存されます。

- エラーEPC レジスタには，例外発生時のプログラム・カウンタの値がセットされます。
- ステータス・レジスタの ERL，SR，BEV ビットは 1 にセットされます。

(3) 操　　作

NMI 例外時，次の操作を行ってください。

- 自己診断のために現在のプロセッサの状態を保存
- コールド・リセット例外と同じようにシステムを再初期化

2.5.4.7 アドレス・エラー例外

(1) 原　　因

アドレス・エラー例外は，次のような場合に発生します。この例外はマスク不可能です。

- ワード境界に位置していないワード・データに対し，LW，LWU，SW，CACHE 命令を実行しようとしたとき。
- ハーフ・ワード境界に位置していないハーフ・ワード・データに対し，LH，LHU，SH 命令を実行しようとしたとき。
- ダブル・ワード境界に位置していないダブル・ワード・データに対し，LD，SD 命令を実行しようとしたとき。
- ユーザ，スーパバイザ・モードから，カーネル・アドレス空間を参照しようとしたとき。
- ユーザ・モードから，スーパバイザ空間を参照しようとしたとき。
- 64 ビットのカーネル／ユーザ／スーパバイザ・モードで，それぞれのアドレス空間にはないアドレスを参照しようとしたとき。
- ワード境界に位置していないアドレスに分岐しようとしたとき。

(2) 処　　理

この例外は一般例外ベクタを使用します。この例外によって，原因レジスタに AdEL，または AdES コードがセットされます。例外の原因が命令参照かロード動作の場合は AdEL，ストア動作の場合は AdES がセットされます。

この例外が発生すると，位置合せされていない仮想アドレス，あるいは参照した空間の仮想アドレスを BadVAddr レジスタに保持します。コンテキスト・レジスタおよびエントリ Hi レジスタの VPN 領域，エントリ Lo レジスタの内容は不定です。

EPC レジスタは，例外の原因となった命令を指しています。ただし，その命令が分岐遅延スロットにあった場合，EPC レジスタは直前のジャンプ／ブランチ命令を指し，原因レジスタの BD ビットが 1 にセットされます。

(3) 操　　作

カーネルは，UNIX™ SIGSEGV（セグメント違反）シグナルを現在実行しているプロセスに知らせます。通常，この例外は致命的エラーです。

2.5.4.8 TLB 例外

TLB 例外には，次の 3 種類があります。

- 参照するアドレスと一致する TLB エントリがない場合，TLB 不一致例外が発生します。
- 参照する仮想アドレスと一致する TLB エントリが無効（V ビットが 0）の場合，TLB 無効例外が発生します。
- ストア命令で参照する仮想アドレスと一致する TLB エントリが有効（V ビットが 1）であるが書き込み不可（D ビットが 0）の場合，TLB 変更例外が発生します。

次に，各 TLB 例外について説明します。

(1) TLB 不一致例外（32 ビット空間モード）／XTLB 不一致例外（64 ビット空間モード）

(a) 原　　因

参照するアドレスと一致する TLB エントリがない場合，TLB 不一致が発生します。この例外はマスク不可能です。

(b) 処　　理

この例外用に，32 ビット・モード用と 64 ビット・モード用の 2 つの特別なベクタが用意されています。どちらのベクタを使用するかはステータス・レジスタの UX，SX，KX ビットによって，ユーザ，スーパバイザおよびカーネル・モードが 32 ビット空間と 64 ビット空間のどちらを使用していたかにより判断します。ステータス・レジスタの EXL ビットが 0 の場合はこの 2 つの特別なベクタを参照し，1 の場合は一般例外ベクタを参照します。

この例外により，原因レジスタの ExcCode 領域に TLBL または TLBS コードがセットされます。例外の原因が命令参照かロード動作の場合は TLBL，ストア動作の場合は TLBS がセットされます。

この例外が発生すると，BadVAddr，コンテキスト，X コンテキスト，エントリ Hi の各レジスタは，アドレス変換に失敗した仮想アドレスを保持します。また，エントリ Hi レジスタは，変換ミスを発生したアドレスの ASID を含んでいます。ランダム・レジスタは，置き換える TLB エントリを示します。エントリ Lo レジスタの内容は不定です。

EPC レジスタは，例外の原因となった命令を指しています。ただし，その命令が分岐遅延スロットにあった場合，EPC レジスタは直前のジャンプ／ブランチ命令を指し，原因レジスタの BD ビットが 1 にセットされます。

(c) 操　作

この例外操作では，コンテキストあるいは X コンテキスト・レジスタの内容は，物理ページ・フレームとアクセス制御ビットを含むメモリ・ワードを TLB エントリ・ペアにロードするための仮想アドレスとして使用します。メモリ・ワードは，エントリ Lo0／エントリ Lo1／エントリ Hi レジスタを通して TLB エントリに書き込まれます。

物理ページ・フレームとアクセス制御ビットを仮想アドレスが TLB にないページ上に置くこともできます。これは，TLB 不一致例外ハンドラ内で TLB 不一致例外を起こすことで処理できます。その場合，ステータス・レジスタの EXL ビットが 1 にセットされているため，一般例外ベクタを使用します。

(2) TLB 無効例外

(a) 原　因

参照する仮想アドレスと一致する TLB エントリが無効（V ビットが 0）の場合，TLB 無効例外が発生します。この例外はマスク不可能です。

(b) 処　理

この例外は一般例外ベクタを使用します。この例外により，原因レジスタの ExcCode 領域に TLBL，または TLBS コードがセットされます。例外の原因が命令参照かロード動作の場合は TLBL，ストア動作の場合は TLBS がセットされます。

この例外が発生すると，BadVAddr，コンテキスト，X コンテキスト，エントリ Hi の各レジスタは，アドレス変換に失敗した仮想アドレスを保持します。また，エントリ Hi レジスタは，変換ミスを発生したアドレスの ASID を含んでいます。ランダム・レジスタは，置き換える TLB エントリを示します。エントリ Lo レジスタの内容は不定です。

EPC レジスタは，例外の原因となった命令を指しています。ただし，その命令が分岐遅延スロットにあった場合，EPC レジスタは直前のジャンプ／ブランチ命令を指し，原因レジスタの BD ビットが 1 にセットされます。

(c) 操　作

通常 TLB エントリの V ビットは，次のような場合にクリアします。

- 仮想アドレスが存在しない場合。
- 仮想アドレスは存在するが，メイン・メモリ中にない場合（ページ・フォールト）。
- そのページの参照でトラップが要求される場合（たとえば，参照する内容を保持したい場合）。

この例外の原因を解決後，TLBP（TLB Probe）命令によって例外を起こした TLB エントリの位置を調べ，そこに別のエントリを置き，V ビットを 1 にセットします。

(3) TLB 変更例外

(a) 原　　因

ストア命令で参照する仮想アドレスと一致する TLB エントリが有効（V ビットが 1）であるが，書き込み不可（D ビットが 0）の場合，TLB 変更例外が発生します。この例外は，マスク不可能です。

(b) 処　　理

この例外は一般例外ベクタを使用し，原因レジスタの ExcCode 領域に Mod コードがセットされます。この例外が発生すると，BadVAddr，コンテキスト，X コンテキスト，エントリ Hi の各レジスタはアドレス変換に失敗した仮想アドレスを保持します。また，エントリ Hi レジスタは，変換ミスを発生したアドレスの ASID を含んでいます。エントリ Lo レジスタの内容は不定です。

EPC レジスタは，例外の原因となった命令を指しています。ただし，その命令が分岐遅延スロットにあった場合，EPC レジスタは直前のジャンプ／ブランチ命令を指し，原因レジスタの BD ビットが 1 にセットされます。

(c) 操　　作

カーネルは，変換ミスを発生した仮想アドレス，あるいは仮想ページ番号を使って，対応するアクセス制御ビットを調べます。アクセスしようとしたページに対して書き込みが許可されていない場合，書き込み保護違反となります。

書き込みが許可されている場合，カーネルによって，そのページ・フレームが独自のデータ構造で Dirty（書き込み可）にマークされます。

そして，変更すべき TLB エントリのインデクスを TLBP 命令でインデクス・レジスタに書き込みます。エントリ Lo レジスタに物理ページ・フレームとアクセス制御ビット（D ビットが 1 にセットされた）を含むワードをロードします。そして，エントリ Hi とエントリ Lo レジスタの内容を TLB に書き込みます。

2.5.4.9 バス・エラー例外

(1) 原　　因

バス・タイムアウト，ローカル・バスのパリティ・エラー，物理メモリ・アドレスやアクセス・タイプが無効など，VR4120A コア内でのイベントが原因で発生するバス・エラー例外です。この例外はマスク不可能です。

キャッシュ・ミス・リフィル，非キャッシュ領域の参照，または非バッファによる書き込みが同時に起きた場合のみ，バス・エラーが発生します。この例外は，システム・コントローラによるリード時に不正アクセスを検出した場合に発生する IBUS バス・エラーによる NMI 割り込みとは異なりますので注意してください。

(2) 処　　理

バス・エラー例外は一般例外ベクタを使用します。原因レジスタの ExcCode 領域に IBE，または DBE コードがセットされ，例外を起こした命令が命令参照なのか，ロード命令かストア命令かを示します。

EPC レジスタは，例外の原因となった命令を指しています。ただし，その命令が分岐遅延スロットにあった場合，EPC レジスタは直前のジャンプ／ブランチ命令を指し，原因レジスタの BD ビットが 1 にセットされます。

(3) 操　　作

エラーの発生した物理アドレスは，システム制御コプロセッサ（CP0）レジスタから得られた情報によって計算します。

- 原因レジスタの IBE コードがセットされていた場合（命令フェッチ時），仮想アドレスは EPC レジスタに格納されます。
- DBE コードがセットされていた場合（ロード／ストア時），例外の原因となった命令の仮想アドレスが EPC レジスタに格納されます。

ロード／ストア命令の対象となる仮想アドレスは，この命令を解読することによって得られます。TLBP 命令を使用し，エントリ Lo レジスタを読み込み，物理ページ番号を計算することによって，物理アドレスは得られます。

この例外発生時，カーネルは UNIX SIGBUS（バス・エラー）シグナルを現在のプロセスに知らせます。通常，この例外は致命的エラーとなります。

2.5.4.10 システム・コール例外

(1) 原　　因

SYSCALL 命令が実行されるとシステム・コール例外が発生します。この例外はマスク不可能です。

(2) 処　　理

この例外は一般例外ベクタを使用します。原因レジスタの ExcCode 領域に Sys コードがセットされます。

EPC レジスタは，SYSCALL 命令を指しています。ただし，SYSCALL 命令が分岐遅延スロットにあった場合，EPC レジスタは直前のブランチ命令を指し，原因レジスタの BD ビットが 1 にセットされます。その他の場合はクリアされます。

(3) 操　　作

この例外が発生すると，制御は該当するシステム・ルーチンに移ります。

実行が再開する際，SYSCALL 命令が再び実行されないように，復帰する前に EPC レジスタに 4 を加えます。

SYSCALL 命令が分岐遅延スロットにある場合，ジャンプ／ブランチ命令をデコードして分岐先を求め，実行を再開します。

2.5.4.11 ブレークポイント例外

(1) 原　　因

BREAK 命令が実行されると，ブレークポイント例外が発生します。この例外はマスク不可能です。

(2) 処　　理

この例外は一般例外ベクタを使用し，原因レジスタの ExcCode 領域に Bp コードがセットされます。

EPC レジスタは，例外の原因となった命令を指しています。ただし，その命令が分岐遅延スロットにあった場合，EPC レジスタは直前のジャンプ／ブランチ命令を指し，原因レジスタの BD ビットが 1 にセットされます。その他の場合はクリアされます。

(3) 操　　作

ブレークポイント例外が発生すると，制御は該当するシステム・ルーチンに移ります。BREAK 命令の未使用ビット（ビット 25-6）を用いて，追加情報を渡すことができます。この情報は，EPC レジスタが指す命令の内容をデータとして読み出すことで得られます。

実行が再開する際，BREAK 命令が再び実行されないように，復帰する前に EPC レジスタに 4 を加えます。

BREAK 命令が分岐遅延スロット内にあった場合，ブランチ命令をデコードして分岐先を求め，実行を再開します。

2.5.4.12 コプロセッサ使用不可例外

(1) 原　　因

コプロセッサの命令の実行において，次に示すような 2 つの場合，コプロセッサ使用不可例外が発生します。

- 対応するコプロセッサ・ユニットの使用が許可されていない場合（ステータス・レジスタのビット CU0 = 0）。
- CP0 が使用不可の場合で，ユーザ・モードかスーパバイザ・モードで CP0 命令を実行したとき。

この例外は，マスク不可能です。

(2) 処　　理

この例外は一般例外ベクタを使用し，原因レジスタの ExcCode 領域に CpU コードがセットされます。また，原因レジスタの CE ビットは，4 つのコプロセッサのどれが参照されたかを示します。

EPC レジスタは，例外の原因となった命令を指しています。ただし，その命令が分岐遅延スロットにあった場合，EPC レジスタは直前のジャンプ／ブランチ命令を指し，原因レジスタの BD ビットが 1 にセットされます。

(3) 操　　作

どのコプロセッサを参照しようとしたのかは，原因レジスタの CE ビットによって示され，ハンドラによって次のいずれかの処理を行います。

(a) プロセスが，アクセスの権利を与えられた場合，コプロセッサの使用を許可することで実行を再開できます。

(b) プロセスがコプロセッサのアクセスの権利を与えられたが，コプロセッサが存在しないかエラー状態のままである場合でも，コプロセッサ命令のデコードは可能です。

(c) 原因レジスタの BD ビットが 1 にセットされていた場合，ブランチ命令をデコードしてください。デコード後，コプロセッサ命令をエミュレートし，EPC レジスタの内容をコプロセッサ命令の先へ進めて実行を再開します。

(d) プロセスがコプロセッサのアクセスの権利を与えられなかった場合，カーネルは，UNIX SIGILL/ILL_PRIVIN_FAULT（不当な命令／特権命令のエラー）シグナルを現在のプロセスに知らせます。この場合，この例外は致命的エラーとなります。

2.5.4.13 予約命令例外

(1) 原　　因

次に示す命令を実行しようとすると，予約命令例外が発生します。

- オペコード（ビット 31-26）が未定義の命令
- サブオペコード（ビット 5-0）が未定義の SPECIAL 命令
- サブオペコード（ビット 20-16）が未定義の REGIMM 命令
- ユーザ，スーパバイザ・モードで 32 ビット・モード時の 64 ビット命令

ステータス・レジスタの KX ビットの値にかかわらず，カーネル・モードでの 64 ビット・オペレーションは常に有効です。
この例外はマスク不可能です。

(2) 処　　理

この例外は一般例外ベクタを使用し，原因レジスタの ExcCode 領域に RI コードがセットされます。
EPC レジスタは，例外の原因となった命令を指しています。ただし，その命令が分岐遅延スロットにあった場合，EPC レジスタは直前のジャンプ／ブランチ命令を指し，原因レジスタの BD ビットが 1 にセットされます。

(3) 操　　作

現在定義されている MIPS ISA 命令はすべて実行できます。
この例外発生時のプロセスの実行は，UNIX SIGILL／ILL_RESOP_FAULT（不当な命令／予約オペランドのミス）信号によって操作されます。通常，この例外は致命的エラーとなります。

2.5.4.14 トラップ例外

(1) 原　　因

TGE, TGEU, TLT, TLTU, TEQ, TNE, TGEI, TGEUI, TLTI, TLTUI, TEQI, TNEI 命令において，条件が真になった場合，トラップ例外が発生します。この例外はマスク不可能です。

(2) 処　　理

この例外は一般例外ベクタを使用し，原因レジスタの ExcCode 領域に Tr コードがセットされます。

EPC レジスタは，例外の原因となったトラップ命令を指しています。ただし，その命令が分岐遅延スロットにあった場合，EPC レジスタは直前のブランチ命令を指し，原因レジスタの BD ビットが 1 にセットされます。

(3) 操　　作

トラップ例外発生時，カーネルは UNIX SIGFPE/FPE_INTOVF_TRAP （浮動小数点例外／整数オーバフロー）シグナルを現在のプロセスへ知らせます。通常，この例外は致命的エラーとなります。

2.5.4.15 整数オーバフロー例外

(1) 原　　因

ADD, ADDI, SUB, DADD, DADDI, DSUB 命令のどれかを実行した結果，2 の補数オーバフローが発生した場合，整数オーバフロー例外が発生します。この例外はマスク不可能です。

(2) 処　　理

この例外は一般例外ベクタを使用し，原因レジスタの ExcCode 領域に Ov コードがセットされます。

EPC レジスタは，例外の原因となった命令を指しています。ただし，その命令が分岐遅延スロットにあった場合，EPC レジスタは直前のブランチ命令を指し，原因レジスタの BD ビットが 1 にセットされます。

(3) 操　　作

この例外発生時，カーネルは UNIX SIGFPE/FPE_INTOVF_TRAP シグナル（浮動小数点例外／整数オーバフロー）を現在のプロセスに知らせます。通常，この例外は，現在のプロセスにとって致命的エラーとなります。

保守／廃止

2.5.4.16 ウォッチ例外

(1) 原　　因

ウォッチ Lo/ウォッチ Hi レジスタ内の物理アドレスをロード／ストア命令で参照しようとした場合，ウォッチ例外が発生します。例外を発生させる命令とビットの状態の組み合わせを次に示します

- ウォッチ Lo レジスタの R ビットが 1 の場合 : ロード命令
- ウォッチ Lo レジスタの W ビットが 1 の場合 : ストア命令
- ウォッチ Lo レジスタの R, W ビット両方が 1 の場合 : ロード命令またはストア命令

キャッシュ命令は，この例外を引き起こしません。

ステータス・レジスタの EXL ビットが 1 にセットされている間，ウォッチ例外は保留されます。ウォッチ例外は，ステータス・レジスタの EXL ビットを 1 にセット，またはウォッチ Lo レジスタの R, W ビットを 0 にクリアすることによりマスク可能です。

(2) 処　　理

この例外は一般割り込みベクタを使用し，原因レジスタの ExcCode 領域に WATCH コードがセットされます。

EPC レジスタは，例外の原因となった命令を指しています。ただし，その命令が分岐遅延スロットにあった場合，EPC レジスタは直前のジャンプ／ブランチ命令を指し，原因レジスタの BD ビットが 1 にセットされます。

(3) 操　　作

ウォッチ例外はディバグ用です。通常，この例外ハンドラはディバッガに制御を移します。これにより，ユーザが状況を検証できます。プログラムを進めるためには，ウォッチ例外を一時マスクして，例外を発生した命令を実行し，その後ウォッチ例外を再び許可します。例外を発生した命令は，ディバッガで 1 命令ごとに実行させるか，ブレークポイントを設定することで実行できます。

リセット時ウォッチ Lo／ウォッチ Hi レジスタの内容は不定となりますので，ソフトウエアで初期化（特に R, W ビットは 0 にクリア）してください。初期化しない場合，ウォッチ例外が発生してしまう場合があります。

2.5.4.17 割り込み例外

(1) 原　　因

8 つの割り込み要因[注]のうち，いずれか 1 つがアクティブとなった場合，割り込み例外が発生します。

μPD98502 では，内蔵周辺ユニットからの割り込み要求を ICU でまとめて，4 つの割り込み要因（Int（3:0））と NMI のいずれかを使用して VR4120A コアに通知します。

8 つの割り込みのそれぞれは，ステータス・レジスタの IM 領域の対応するビットをクリアすることによってマスク可能です。また，ステータス・レジスタの IE ビットをクリア，または EXL/ERL ビットをセットすることによって，すべての割り込みをマスクできます。

注　タイマ割り込み：1，通常割り込み：5，ソフトウエア割り込み：2

(2) 処　　理

この例外は一般例外ベクタを使用し，原因レジスタの ExcCode 領域に Int コードがセットされます。

原因レジスタの IP 領域は，現在発生している割り込みを示します。原因レジスタを読み出す前に割り込み要求信号をアクティブ（インアクティブ）にすると，複数のビットを同時にセット（クリア）できます。

EPC レジスタは，例外の原因となった命令を指しています。ただし，その命令が分岐遅延スロットにあった場合，EPC レジスタは直前のジャンプ／ブランチ命令を指し，原因レジスタの BD ビットが 1 にセットされます。

(3) 操　　作

2 つのソフトウエア例外（SW0 と SW1）のどちらかによって割り込みが発生した場合，対応する原因レジスタのビットに 0 を書き込むことによって割り込みは解除されます。

割り込みがハードウエアによって発生した場合，原因となる割り込み要求信号をインアクティブにすることによって割り込みは解除されます。

2.5.5 例外処理のフロー・チャート

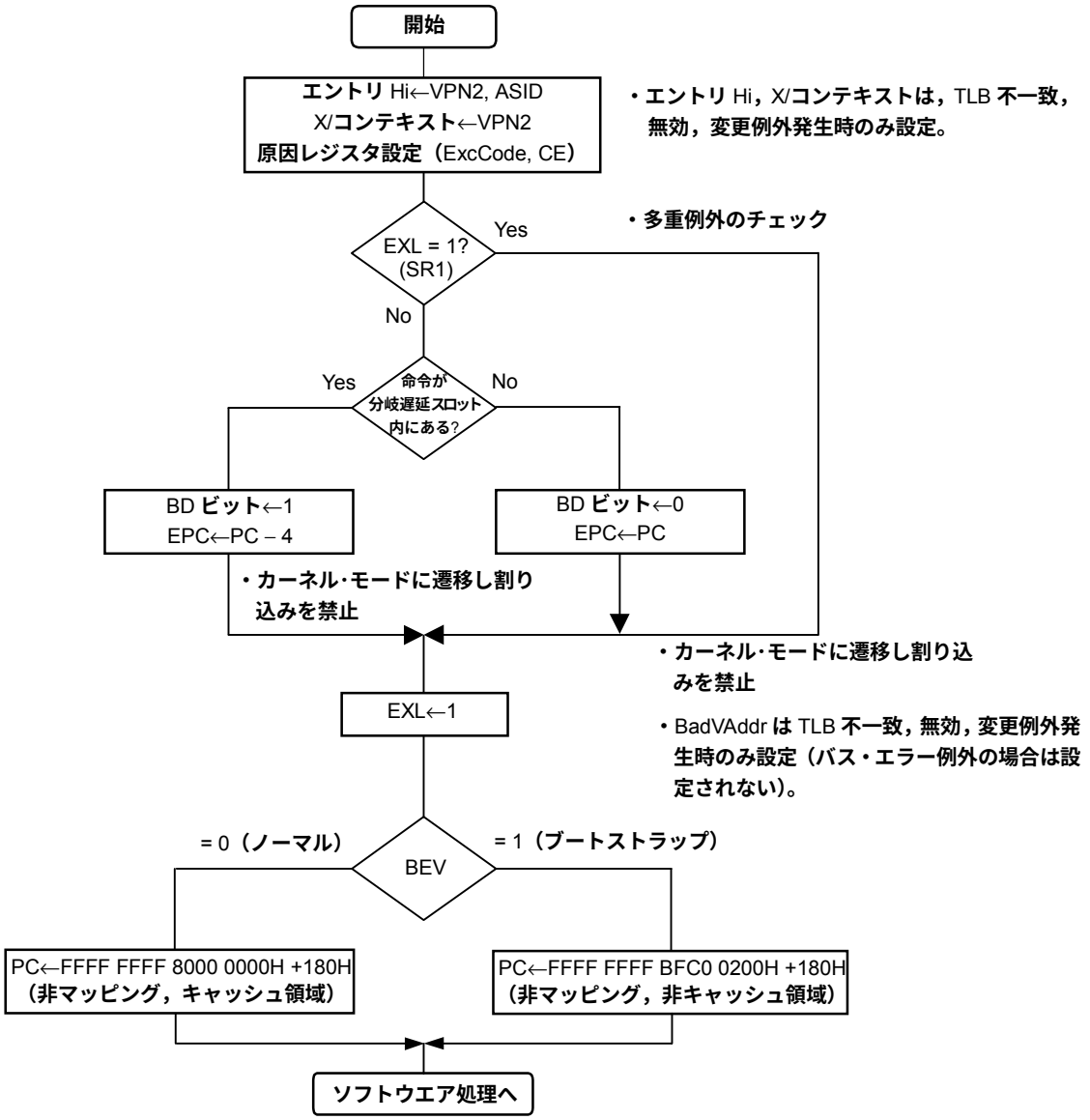
ここでは，次に示す例外の処理の流れや例外ハンドラの操作基準を記述します。

- 一般例外の処理と例外ハンドラ
- TLB/XTLB 不一致例外の処理と例外ハンドラ
- コールド・リセット例外，ソフト・リセット例外，NMI 例外の処理と，各例外ハンドラ

また，ハードウエアが取り扱うことを「処理」，ソフトウエアが取り扱うことを「操作」と呼んでいます。

図 2-63 一般例外の処理（1/2）

(a) コールド・リセット，ソフト・リセット，NMI，TLB/XTLB 不一致例外以外の例外（ハードウエア）



備考　割り込みは IE または IM ビットでマスク可能です。
また，ウォッチ例外は EXL ビット = 1 で保留にされます。

図 2-63　一般例外の処理（2/2）

（b）一般例外の処理（ソフトウエア）

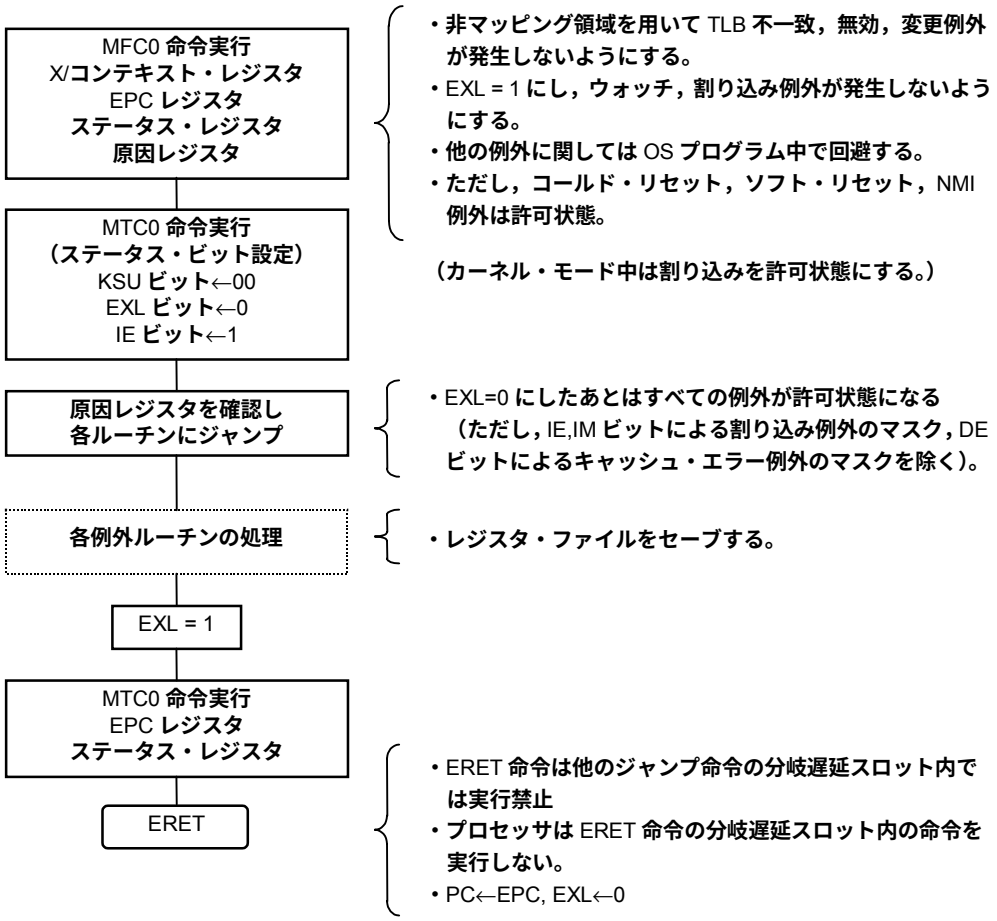


図 2-64　TLB/XTLB 不一致例外の処理（1/2）

（a）TLB/XTLB 不一致例外の例外処理（ハードウエア）

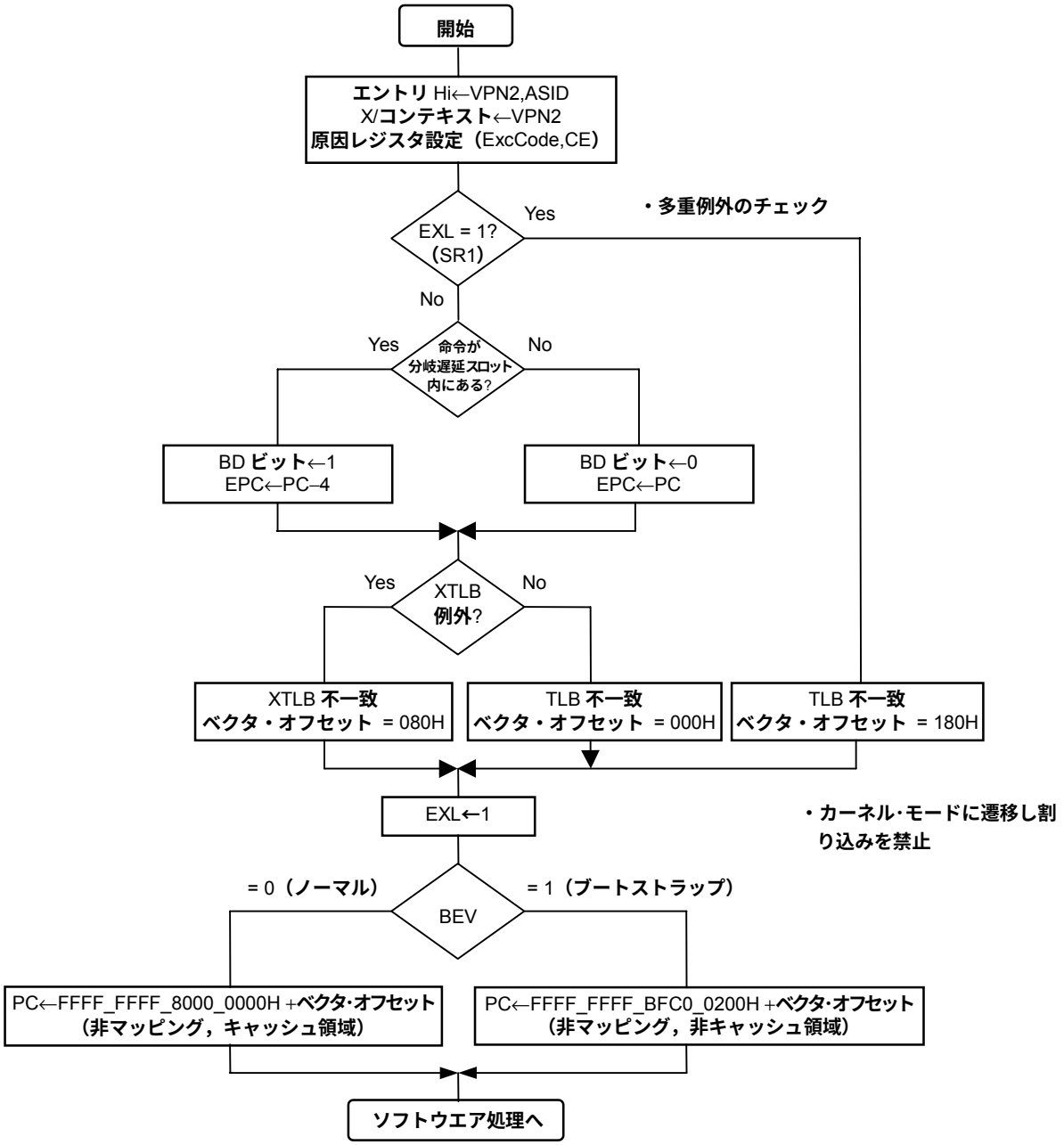


図 2-64　TLB/XTLB 不一致例外の処理（2/2）

（b）TLB/XTLB 不一致例外の処理（ソフトウエア）

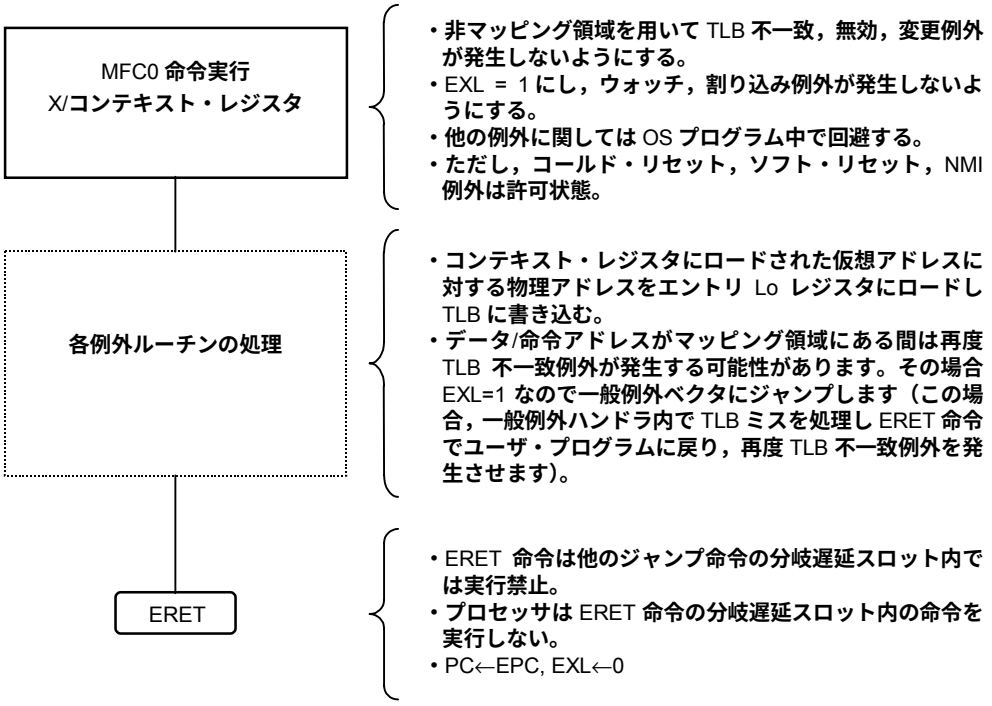


図 2-65 コールド・リセット例外ハンドラの処理

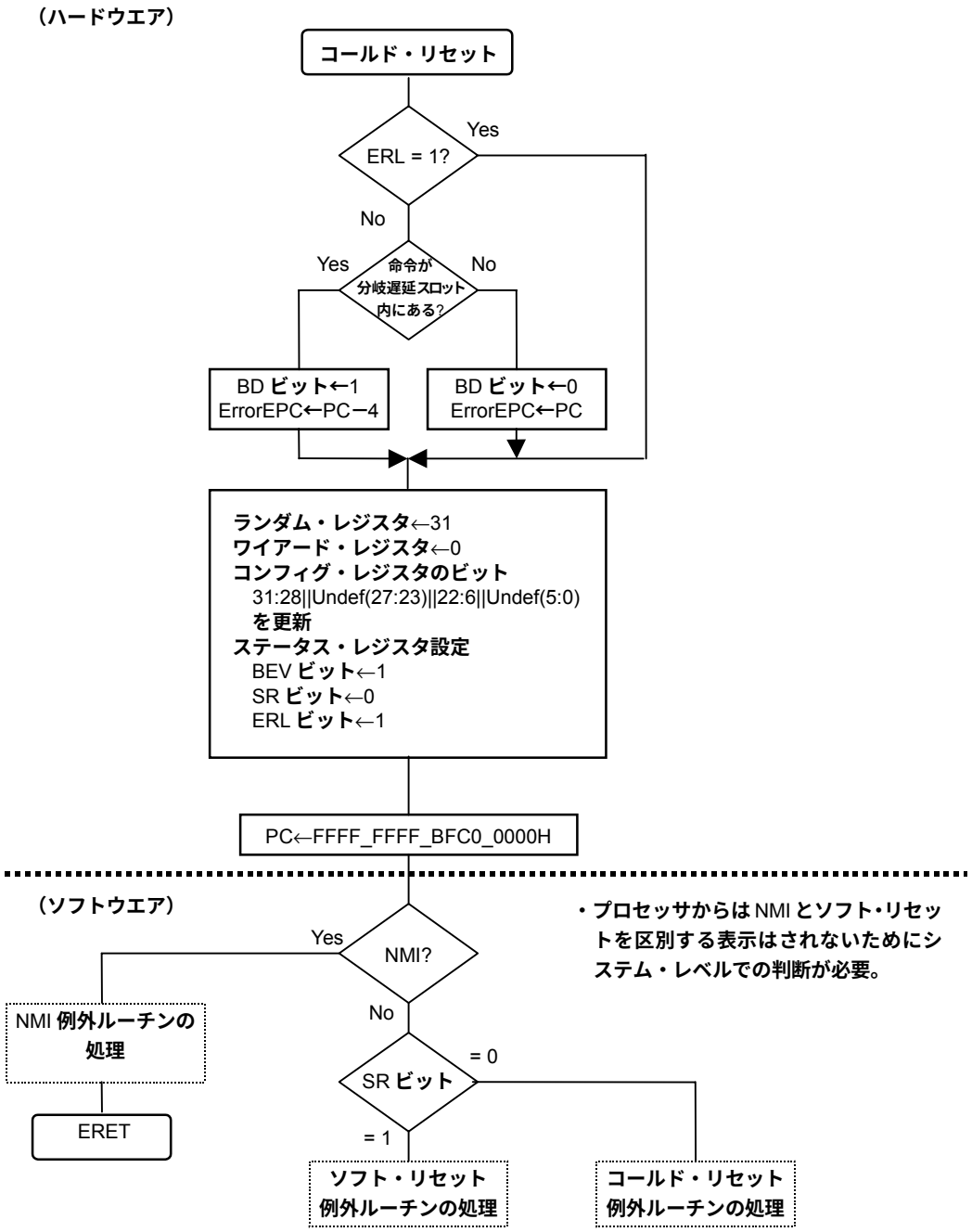
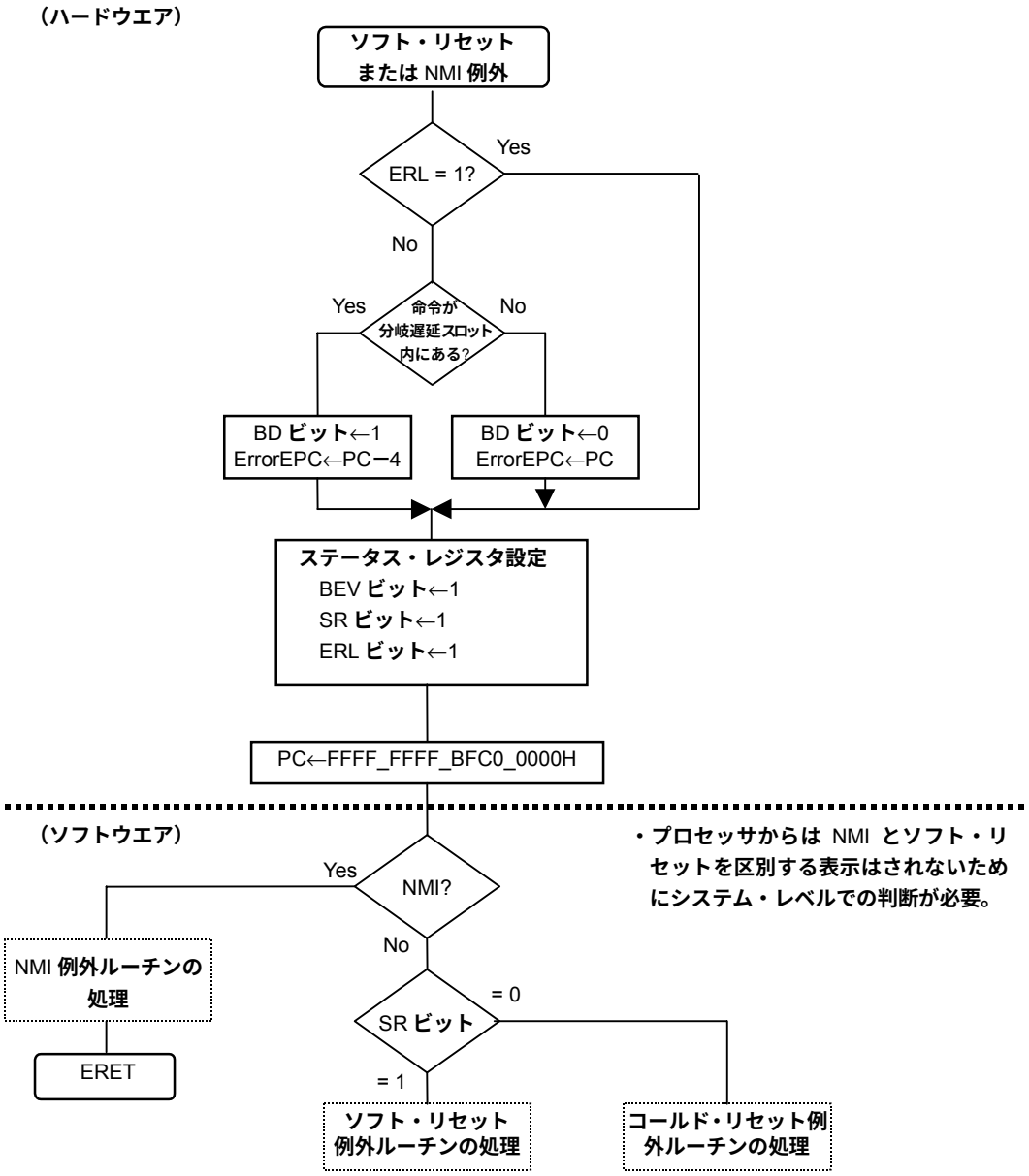


図 2-66 ソフト・リセット/NMI 例外ハンドラの処理



保守／廃止

2.6 初期化インタフェース

ここでは，VR4120A コアがリセットされるシーケンスを示します。リセットの要因やμPD98502 全体のリセットについては，**1.10 リセット概要**を参照してください。

2.6.1 コールド・リセット

μPD98502 では，次の場合に VR4120A コアでコールド・リセット・シーケンスが実行されます。

- ハードウエア・リセット
- ウォッチドッグ・タイマ・シャットダウン
- ソフトウエア・シャットダウン
- HAL Timer シャットダウン

コールド・リセットでは，VR4120A コアが完全に初期化されます。リセット後，次のレジスタ以外は値が不定になります。

- ステータス・レジスタの SR ビットが 0 にクリアされます。
- ステータス・レジスタの ERL，BEV ビットが 1 にセットされます。
- ランダム・レジスタに上限値（31）がセットされます。
- ワイアード・レジスタは 0 に初期化されます。
- コンフィグ・レジスタのビット 31-28 が 0，ビット 22-3 が 04800H に設定されます。その他のビットは不定になります。

2.6.2 ソフト・リセット

μPD98502 では，ソフト・リセットはサポートしていません。

ソフト・リセットでは，出力クロックに影響を与えずに VR4120A コアが初期化されます。つまり，ソフト・リセットは論理的リセットです。ソフト・リセットでは，VR4120A コアはほとんどの状態情報を保持しますが，次にあげるものは保持しません。

- ステータス・レジスタの BEV，SR，ERL ビットが 1 にセットされます。
- カウント・レジスタは 0 に初期化されます。
- 原因レジスタの IP7 ビットが 0 にクリアされます。
- SysAD バス上で発生する割り込みはクリアされます。
- ノンマスカブル割り込み（NMI）はクリアされます。
- コンフィグ・レジスタは初期化されます。

2.6.3 VR4120A コアのモード

VR4120A コアは，いろいろなモードをサポートしており，ユーザが選択できます。VR4120A コアのモードは，ステータス・レジスタとコンフィグ・レジスタに書き込むことで設定します。内蔵周辺回路のモードは，I/O レジスタに書き込むことで設定します。

ここでは，VR4120A コアの動作モードについて説明します。内蔵周辺回路の動作モードについては各ユニットの章を参照してください。

2.6.3.1 電力モード

VR4120A コアは，Fullspeed モード，Standby モード，Suspend モード，Hibernate モードの 4 つのモードをサポートしています。

（1）Fullspeed モード

通常の動作モードです。

ディフォールトの状態では，VR4120A コアは Fullspeed モードで動作します。また，リセット後も Fullspeed モードに戻ります。

（2）Standby モード

STANDBY 命令を実行すると，VR4120A コアを Standby モードにすることができます。Standby モードの動作は Suspend モードの動作と同じです

（3）Suspend モード

SUSPEND 命令を実行すると，VR4120A コアを Suspend モードにすることができます。Suspend モードでは，タイマ用，割り込み制御用を除いた VR4120A コア内部のクロックをハイ・レベルの状態に保持します。

SUSPEND 命令が WB ステージを終了すると，VR4120A コアはシステム・インタフェースがアイドル状態になるまで待機し，システム・バス（SysAD バス）がアイドル状態になるまで待機します。SysAD バスがアイドル状態になると，内蔵のクロック・ジェネレータが内部クロックをハイ・レベルに固定します。割り込み制御用クロック（MOUT）は動作を続けます。

Suspend モードの VR4120A コアは，内部で発生するタイマ割り込みを含めたすべての割り込みにより，Fullspeed モードに戻ります。

（4）Hibernate モード

HIBERNATE 命令を実行すると，VR4120A コアを Hibernate モードにすることができます。Hibernate モードでは，VR4120A コアの割り込み制御用クロック（MOUT）を含むすべての内部クロックをハイ・レベルの状態に保持します。

ただし，Hibernate モード中でもμPD98502 の内蔵クロックジェネレータは CPU クロック（100 MHz/66 MHz）で動作します。

Hibernate モードの VR4120A コアは，コールド・リセットにより Fullspeed モードに復帰できます。

Hibernate モード期間中に電源が供給され続けていれば，コールド・リセット時にクロック・ジェネレータを初期化する必要はありません。

2.6.3.2 特権モード

VR4120A はカーネル／スーパバイザ／ユーザ拡張アドレシングの 3 つのシステム・モードをサポートしています。この項ではこの 3 つのモードについて説明します。

(1) カーネル拡張アドレシング・モード

ステータス・レジスタの KX ビットをセットすると，カーネル・アドレスの TLB 不一致に拡張 TLB 不一致例外ベクタを使用します。カーネル・モードでは，MIPS III オペコードは KX ビットに関係なく常に使用できます。

(2) スーパバイザ拡張アドレシング・モード

ステータス・レジスタの SX ビットをセットすると，スーパバイザ・モードで MIPS III オペコードを使用でき，スーパバイザ・アドレスの TLB 不一致に拡張 TLB 不一致例外ベクタを使用します。

(3) ユーザ拡張アドレシング・モード

ステータス・レジスタの UX ビットをセットすると，ユーザ・モードで MIPS III オペコードを使用でき，ユーザ・アドレスの TLB 不一致に拡張 TLB 不一致例外ベクタを使用します。このビットをクリアすれば，MIPS I，II オペコードと 32 ビット仮想アドレスを使用します。

2.6.3.3 リバース・エンディアン

ステータス・レジスタの RE ビットをセットすると，ユーザ・ソフトウエアから見てエンディアンが逆転します。ただし，VR4120A コアは常にリトル・エンディアンで動作するので，RE ビットは 0（反転禁止）に固定してください。

2.6.3.4 ブートストラップ例外ベクタ（BEV）

ステータス・レジスタの BEV ビットは，診断テストでキャッシュとメイン・メモリ・システムの適正動作の検査中に例外を発生させるときに使用します。

ステータス・レジスタの BEV ビットをセットすると，TLB 不一致例外ベクタを FFFF_FFFF_BFC0_0200H の仮想アドレスに変更し，一般の例外ベクタをアドレス FFFF_FFFF_BFC0_0380H に変更します。

BEV ビットをクリアすると，TLB 不一致例外ベクタは FFFF_FFFF_8000_0000H に，一般の例外ベクタは FFFF_FFFF_8000_0180H になります。

2.6.3.5 キャッシュ・エラー・チェック

VR4120A コアでは，キャッシュ・パリティは存在しないので，ステータス・レジスタの CE ビットは意味を持ちません。

2.6.3.6 パリティ・エラーの禁止

VR4120A コアでは，ステータス・レジスタの DE ビットにかかわらず，キャッシュ・パリティ・エラー例外を発生することはありません。

2.6.3.7 割り込みイネーブル（IE）

ステータス・レジスタの IE ビットをクリアすると，リセットとノンマスカブル割り込み以外のすべての割り込みが受け付けられなくなります。

保守／廃止

2.7 キャッシュ・メモリ

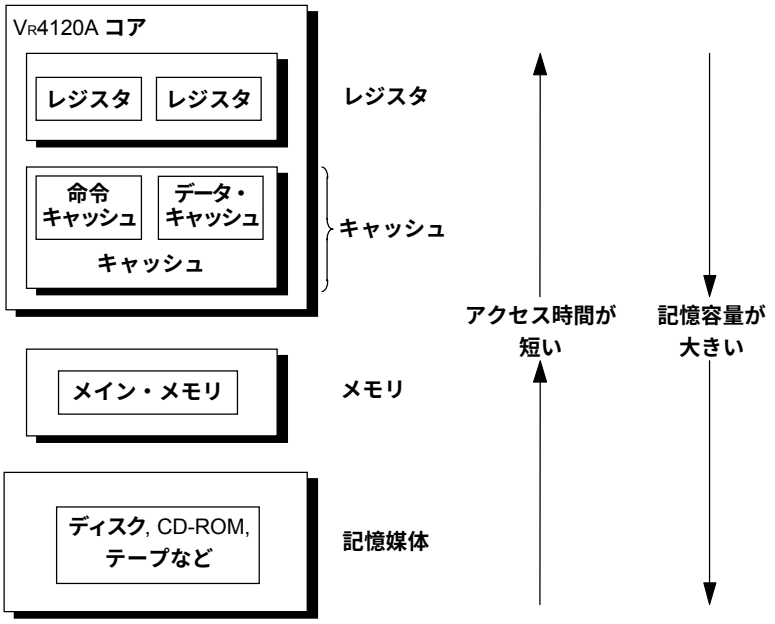
この節では，VR4120A コアのメモリ構成におけるキャッシュ・メモリの位置付けと各キャッシュの構成について説明します。

2.7.1 メモリ構成

VR4120A コアのメモリ構成を図 2-67に示します。論理上のメモリ階層中，キャッシュは CPU とメイン・メモリの中間に位置しており，ユーザ側から見てメモリへのアクセスがより速くなるように設計されています。

図 2-68に示す機能ブロック図では，下に行くほど容量が多くなっています。たとえば，メイン・メモリ（物理メモリ）は，キャッシュより大きな容量を持っています。反対に，下に行くほど（容量が大きいほど）アクセス時間が長くなります。たとえばメイン・メモリへのアクセスの方が，CPU 内に搭載されたレジスタへのアクセスより時間がかかります。

図 2-67 メモリ階層



VR4120A コアは，キャッシュを 2 つ搭載しています。一方は命令キャッシュといい，命令を保持します。他方をデータ・キャッシュといい，データを保持します。両キャッシュとも 1 PClock サイクルで読み出せます。

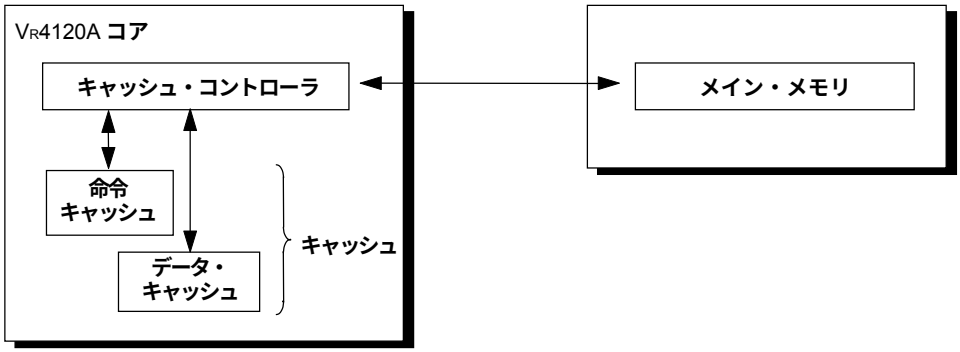
データの書き込みには 2 PCycle 必要ですが，1 PCycle 単位のパイプライン処理が可能なので，1 PCycle ごとに実行できます。最初のサイクルでストア・アドレスを変換し，タグをチェックします。そして次のサイクルでデータ RAM にデータを書き込みます。

保守／廃止

2.7.2 キャッシュの構成

この項では，VR4120A コアに内蔵しているデータ・キャッシュと命令キャッシュの構成について説明します。キャッシュとメモリの構成を図 2-68に示します。

図 2-68 キャッシュ



(1) キャッシュ・ラインのサイズ

キャッシュ・ラインは，メイン・メモリからキャッシュにフェッチできる情報の最小単位であり，単一のタグによって表示されます。

キャッシュ・ラインのサイズは，命令キャッシュ，データ・キャッシュともに 4 ワード（16 バイト）です。

キャッシュ・タグについては，2.7.2.1 **命令キャッシュの構成**，2.7.2.2 **データ・キャッシュ**の構成を参照してください。

(2) キャッシュのサイズ

命令キャッシュの容量は 16 K バイト，データ・キャッシュの容量は 8 K バイトです。

2.7.2.1 命令キャッシュの構成

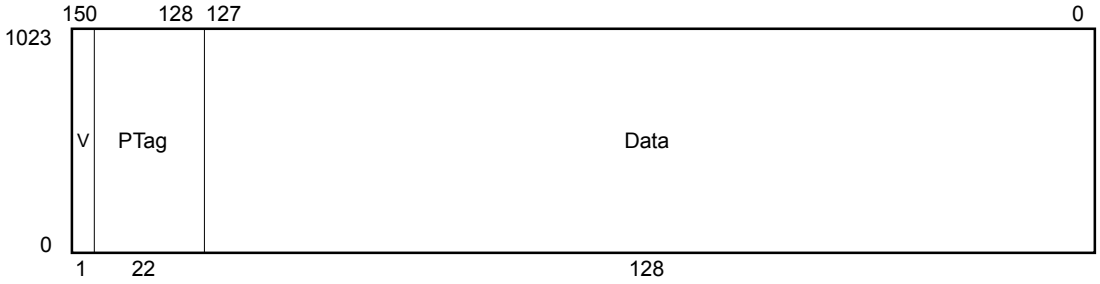
命令キャッシュのデータ（実際は命令ですが，タグと区別するためにデータと呼びます）の各ラインは，23 ビットのタグと連動しています。タグは，22 ビットの物理アドレス，1 ビットのバリッド・ビットで構成しています。

次に命令キャッシュの特徴を示します。

- ダイレクト・マッピング方式
- 仮想アドレスを用いたインデクス
- 物理タグを用いたチェック
- 4 ワード（16 バイト）構成のキャッシュ・ライン

命令キャッシュの 4 ワード（16 バイト）のキャッシュ・ラインの形式を図 2-69に示します。

図 2-69 命令キャッシュのライン形式



PTag　：物理タグ（物理アドレスのビット 31：10）

V　　：バリッド・ビット

Data　：キャッシュ・データ

2.7.2.2 データ・キャッシュの構成

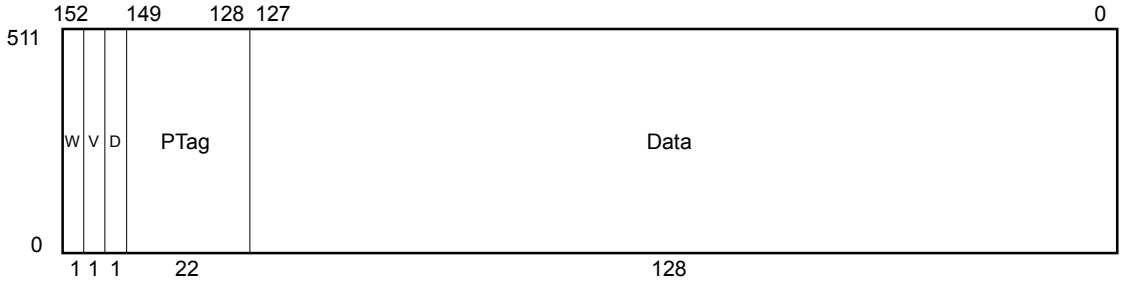
データ・キャッシュのデータの各ラインは，25 ビットのタグと連動しています。タグは，22 ビットの物理アドレス，1 ビットのバリッド・ビット，1ビットのダーティ・ビット，および 1 ビットのライトバック・ビットで構成されています。

次にデータ・キャッシュの特徴を示します。

- ライトバック
- ダイレクト・マッピング方式
- 仮想アドレスを用いたインデクス
- 物理タグを用いたチェック
- 4 ワード（16 バイト）構成のキャッシュ・ライン

データ・キャッシュの 4 ワード（16 バイト）のキャッシュ・ラインの形式を図 2-70に示します。

図 2-70 データ・キャッシュのライン形式



W　　：ライトバック・ビット（キャッシュ・ラインが書き換えられるとセット）

D　　：ダーティ・ビット

V　　：バリッド・ビット

PTag　：物理タグ（物理アドレスのビット 31：10）

Data　：データ・キャッシュのデータ

保守／廃止

2.7.2.3 キャッシュへのアクセス

図 2-71にキャッシュ内への仮想アドレス（VA）のインデクスについて示します。仮想アドレスのうち，命令キャッシュやデータ・キャッシュをインデクスするために使用するビット数は，そのキャッシュ・サイズによって異なります。

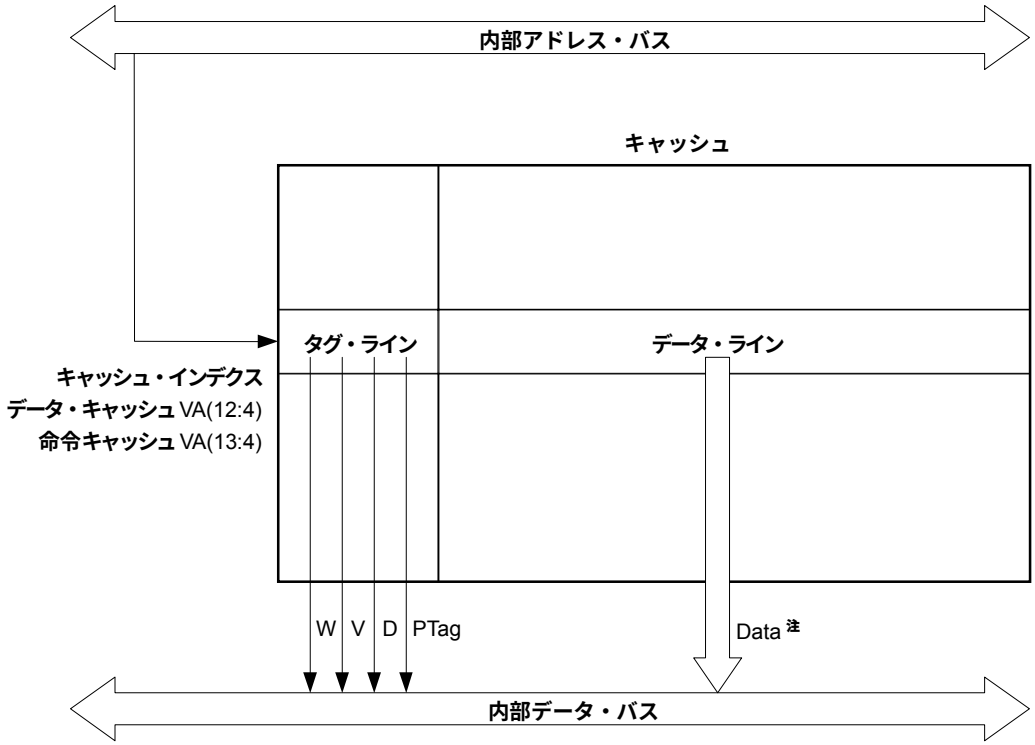
(1) データ・キャッシュ・アドレシング

VA（12：4）を用います。キャッシュ・サイズが 8 K バイトなので，最上位ビットは VA12 となります。またライン・サイズが 4 ワード（16 バイト）なので，最下位ビットは VA4 となります。

(2) 命令キャッシュ・アドレシング

VA（13：4）を用います。キャッシュ・サイズが 16 K バイトなので，最上位ビットは VA13 となります。またライン・サイズが 4 ワード（16 バイト）なので，最下位ビットは VA4 となります。

図 2-71 キャッシュ・データとタグの構成



注 キャッシュからのデータ出力は，命令キャッシュでは 32 ビット単位で，データ・キャッシュでは 64 ビット単位で行われます。

保守／廃止

2.7.3 キャッシュの動作

キャッシュは一時的にデータをストアするものです。キャッシュを使用すればユーザから見たメモリ・アクセスのスピードが向上します。VR4120A コアの CPU はキャッシュに常駐している命令やデータを次の手順でアクセスします。

<1> CPU は，適切なキャッシュから次に使用する命令やデータを，内蔵のキャッシュ・コントローラを介してアクセスします。

<2> キャッシュ・コントローラは，要求された命令やデータがそのキャッシュにあるかどうかをチェックします。

- 存在する場合，CPU にその命令やデータを引き渡します。これをキャッシュ・ヒットと呼びます。
- 存在しなかった場合，キャッシュ・コントローラが要求された命令やデータをメイン・メモリから読み込みます。これをキャッシュ・ミスと呼びます。

<3> 要求された命令やデータが見つかったら，それを CPU に渡します。CPU はそれを受けとり，動作を続行します。

また，同じデータが 2 つの場所（メイン・メモリとキャッシュ）に同時に存在することがあります。このデータは，ライトバック手法によってコヒーレンシ（一貫性）が保たれます。すなわち，データを変更すると，そのキャッシュ・ラインが置き換えられるまでライトバックされません。

2.7.3.1 キャッシュへの書き込み原則

VR4120A コアは，ライトバック方式でデータ・キャッシュを管理しています。ライトバックとは，ストアするデータをメイン・メモリに書き込まず，キャッシュだけに書き込むことです[注]。キャッシュに入ったデータは，しばらく経ってから単独でメイン・メモリに転送されます。VR4120A コアでは，キャッシュ・ミスの場合か，またはライトバック CACHE 命令の実行中に，キャッシュ・ラインが置き換えられるまで，変更されたキャッシュ・ラインはメイン・メモリにライトバックされません。

VR4120A コアがキャッシュ・ラインの内容をメイン・メモリにライトバックするときは，通常キャッシュ・ラインのコピーを保持せず，そのキャッシュ・ラインは Invalid 状態になります。

注 これと逆の働きをするのがライトスルー・キャッシュです。メモリに書き込まれる情報が，同時にキャッシュにも書き込まれます。

2.7.4 キャッシュの状態

(1) キャッシュ・ライン

キャッシュ・ラインには次の 3 **つの状態があります。**

- Dirty ： **キャッシュ・ラインのデータが，メイン・メモリからロードされたあとに変更されている。**
- Clean ： **キャッシュ・ラインのデータが，メイン・メモリからロードされてから変更されていない。**
- Invalid ： **キャッシュ・ラインが有効な情報を持っていない。**
  この状態のキャッシュ・ラインは使用できません。
  ソフト・リセット後のキャッシュ・ラインは状態が不安なため，ソフトウエアですべて Invalid **にセットしてください。**Invalid **以外の状態のキャッシュ・ラインは，有効な情報を持っていると見なされます。**
  なお，コールド・リセットもソフト・リセットも，キャッシュの状態を Invalid **にはしません。キャッシュの無効化はソフトウエアで行います。**

(2) 命令キャッシュ

命令キャッシュは 2 **つのキャッシュ状態をサポートします。**

- Invalid
- Valid

(3) データ・キャッシュ

データ・キャッシュは 3 **つのキャッシュ状態をサポートします。**

- Invalid
- Valid clean
- Valid dirty

Valid **状態のキャッシュ・ラインは，**VR4120A **コアが** CACHE **オペレーションを実行すると変更されることがあります。**CACHE **オペレーションについては，付録** A **MIPS** III **命令セット**を参照してください。

保守／廃止

2.7.5 キャッシュの状態遷移

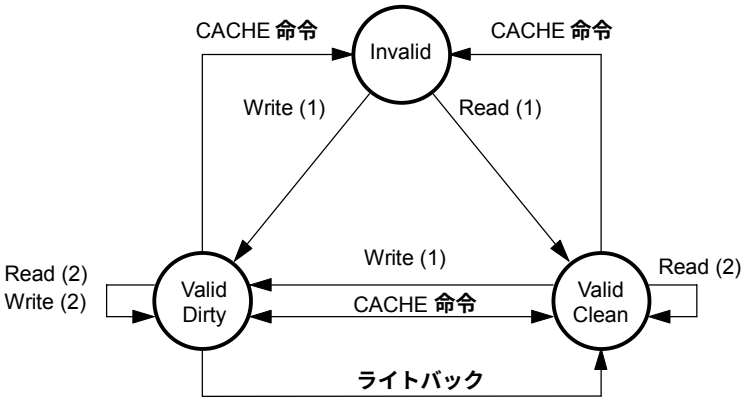
ここでは，データ・キャッシュ・ラインと命令キャッシュ・ラインの状態遷移について説明します。ただし，システムの初期状態はそのシステムに依存するため，ここでは触れていません。

2.7.5.1 データ・キャッシュの状態遷移

図 2-72に示すように，ロード／ストア動作には，さらに小さいリード／ライト動作を 1 つ以上含んでおり，これがキャッシュの状態を遷移させます。

- Read（1）：メイン・メモリからキャッシュへの読み出し動作で，キャッシュの状態が変わる。
- Read（2）：キャッシュから CPU への読み出し動作で，キャッシュの状態は変わらない。
- Write（1）：CPU からキャッシュへの書き込み動作で，キャッシュの状態が変わる。
- Write（2）：CPU からキャッシュへの書き込み動作で，キャッシュの状態は変わらない。

図 2-72 データ・キャッシュの状態遷移

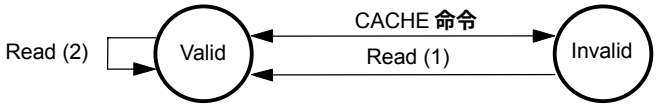


2.7.5.2 命令キャッシュの状態遷移

図 2-73に，命令キャッシュの状態遷移を示します。

- Read（1）：メイン・メモリからキャッシュへの読み出し動作であり，キャッシュの状態が変わる。
- Read（2）：キャッシュから CPU への読み出し動作であり，キャッシュの状態は変わらない。

図 2-73 命令キャッシュの状態遷移



保守／廃止

2.7.6 キャッシュ・データのチェック

図 2-74から図 2-88にさまざまなキャッシュ・アクセスにおけるオペレーションのフローを示します。

図 2-74 命令フェッチ時のチェック・フロー

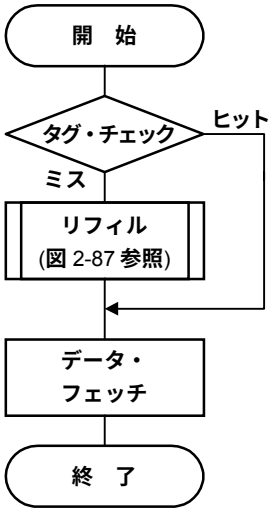


図 2-75 ロード時のチェック・フロー

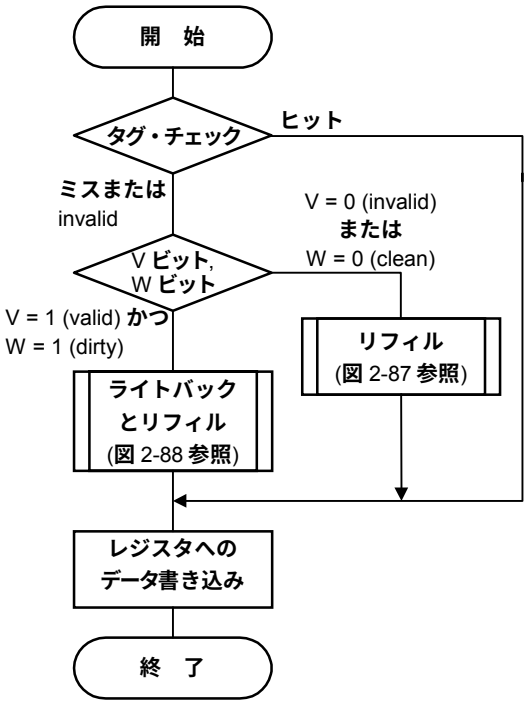


図 2-76 ストア時のチェック・フロー

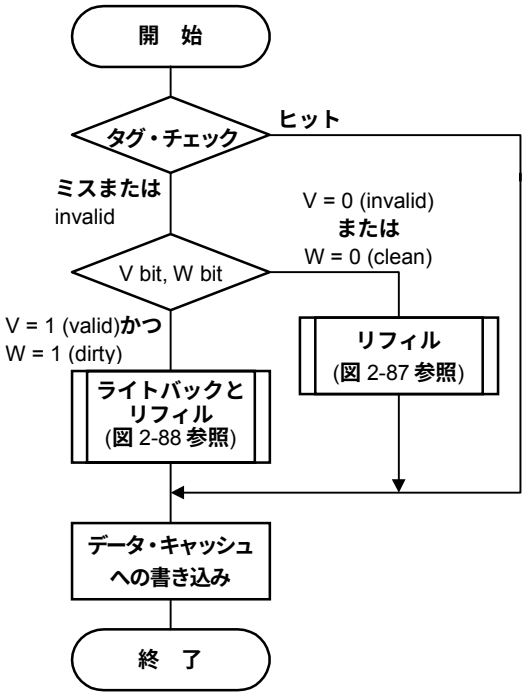


図 2-77 Index_Invalidate オペレーション時のチェック・フロー

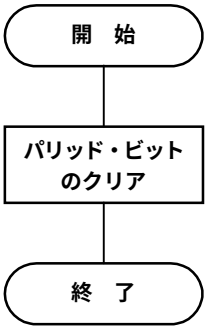


図 2-78 Index_Writeback_Invalidate オペレーション時のチェック・フロー

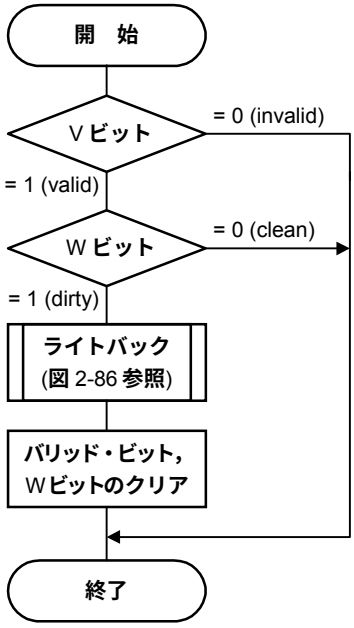


図 2-79 Index_Load_Tag オペレーション時のチェック・フロー

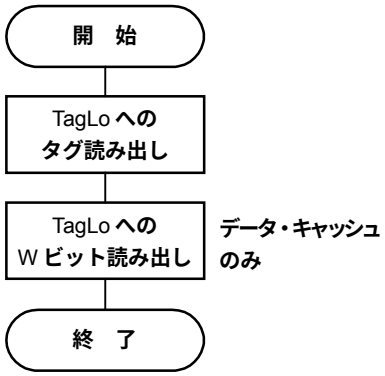


図 2-80 Index_Store_Tag オペレーション時のチェック・フロー

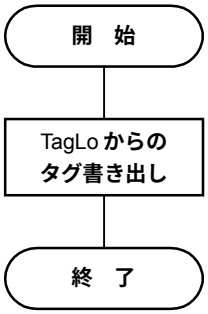


図 2-81　Create_Dirty オペレーション時のチェック・フロー

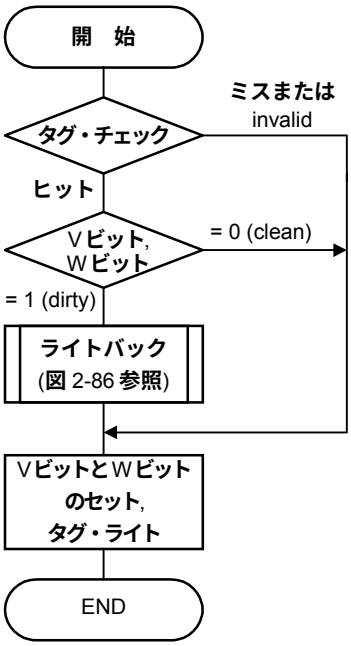


図 2-82　Hit_Invalidate オペレーション時のチェック・フロー

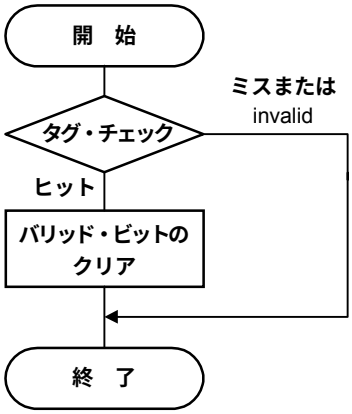


図 2-83 Hit_Writeback_Invalidate オペレーション時のチェック・フロー

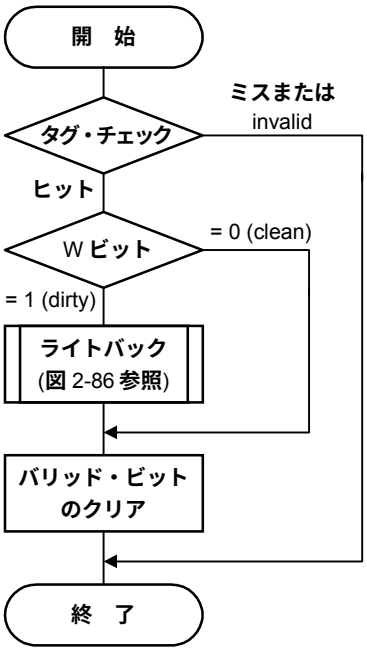


図 2-84 Fill オペレーション時のチェック・フロー

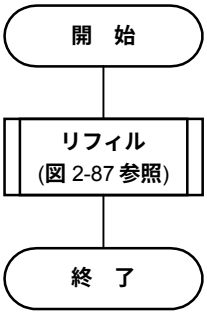
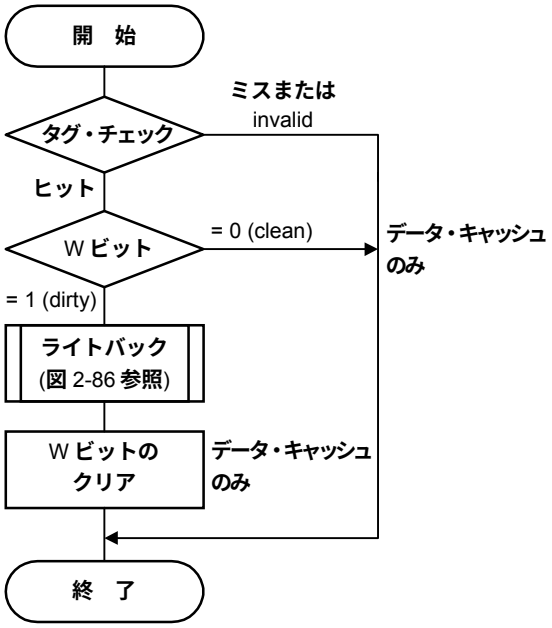


図 2-85 Hit_Writeback オペレーション時のチェック・フロー



保守／廃止

図 2-86 ライトバック・フロー

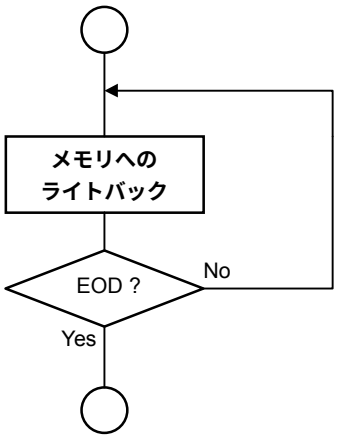


図 2-87 リフィル・フロー

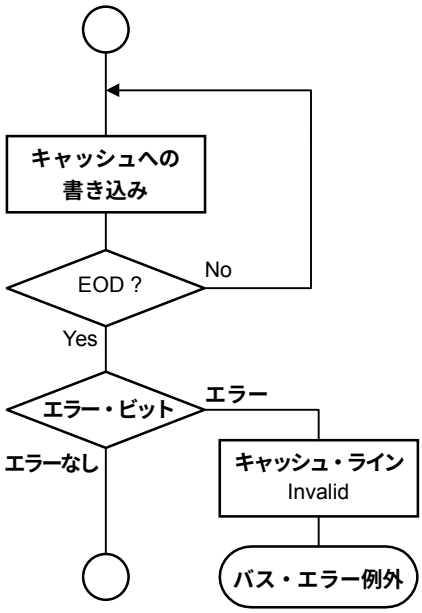
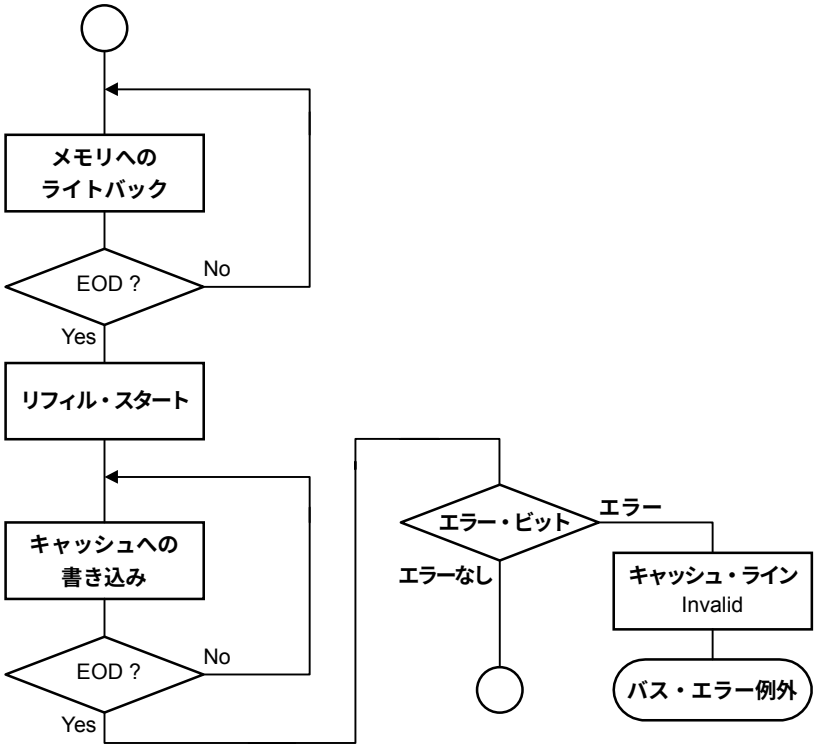


図 2-88 ライトバックとリフィル・フロー



備考　ライトバック手順：

ストア・ミス時にライトバックするとき，データ・タグをチェックし，データをライト・バッファに転送します。

データ・フィールドにエラーが検出された場合，ライトバックの動作は終了せずに，誤ったデータをメイン・メモリに書き込みます。タグ・フィールドにエラーが検出された場合，ライトバックは行われません。

CACHE オペレーションの間は，キャッシュ・データはチェックされない場合があります。

2.7.7 外部エージェントによるキャッシュの操作

μPD98502 では，両キャッシュの状態や内容を外部エージェントから検査したり操作することはできません。

保守／廃止

2.8 VR4120A コアの割り込み

この節では，VR4120A コアの 4 種類の割り込みについて説明します。

（1）ノンマスカブル割り込み（NMI）：1 要因
（2）通常割り込み：5 要因
（3）ソフトウエア割り込み：2 要因
（4）タイマ割り込み：1 要因

2.8.1 ノンマスカブル割り込み（NMI）

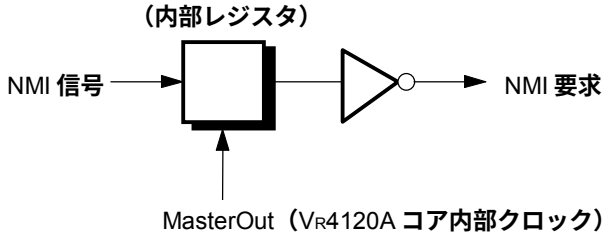
NMI 信号（内部）がアクティブになると NMI 要求が受け付けられ，リセット例外ベクタに分岐します。NMI 信号は，図 2-89に示すように，MasterOut 信号（内部）の立ち上がりに同期して内部レジスタにラッチされます。

NMI がかかるのは，パイプラインの動作中だけです。

NMI はマスクできません。

図 2-89に，NMI 信号の内部処理を示します。NMI 信号を MasterOut の立ち上がりに同期して内部のレジスタにラッチします。ラッチした NMI 信号を反転し，内部に NMI 要求として伝達します。

図 2-89 NMI 信号



2.8.2 通常割り込み

この割り込み要求は，Int（4:0）信号（内部）がアクティブになると受け付けられます。

なお，この割り込み要求は，ステータス・レジスタの IM（6:2），IE, EXL, ERL フィールドでマスクできます。

2.8.3 VR4120A コアのソフトウエア割り込み

この割り込み要求は，原因レジスタの IP（割り込み保留）フィールドのビット 1 かビット 0 を 1 にセットすると受け付けられます。これらのビットには，ソフトウエアで書き込みを行ってください。ハードウエアでのセット／クリアはできません。

また，ソフトウエア割り込み例外発生後，通常ルーチンに戻るまで，または多重割り込みを許可する前に，原因レジスタの IP フィールドの該当ビットを 0 にクリアしてください。

なお，この割り込み要求は，ステータス・レジスタの IM（1:0），IE, EXL, ERL フィールドでマスクできます。

保守／廃止

2.8.4 タイマ割り込み

この割り込み要求は，原因レジスタの IP（割り込み保留）フィールドのビット 7 を使用します。カウント・レジスタの値が比較レジスタの値と等しくなると自動的にセットされ，割り込み要求を受け付けます。

なお，この割り込み要求は，ステータス・レジスタの IM7 ビット，IE, EXL, ERL フィールドでマスクできます。

注意　DSU からのタイマ割り込みとは異なります。

2.8.5 割り込み要求信号の発生

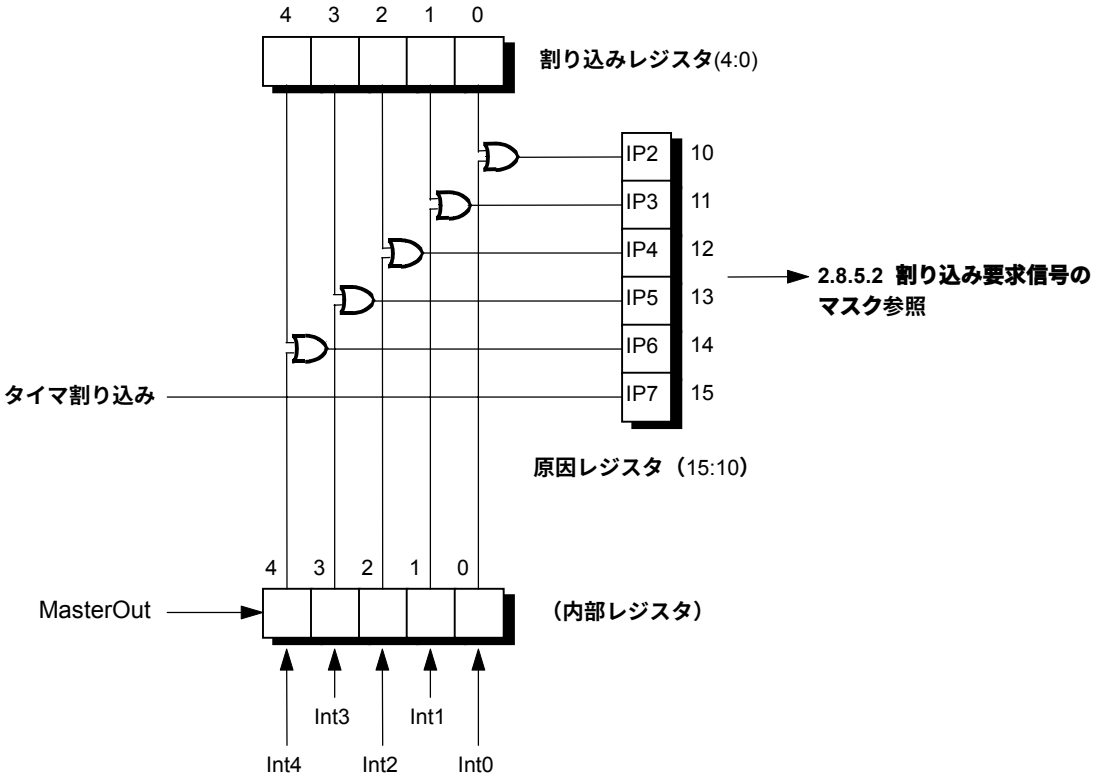
2.8.5.1 ハードウエア割り込みの検知

ハードウエア割り込みの要求を原因レジスタを使って検知する方法を図 2-90に示します。原因レジスタについては2.5.3.6　**原因レジスタ（13）**を参照してください。

- VR4120A コアのタイマ割り込み信号（IP7）は，原因レジスタのビット 15 から直接調べられます。
- Int0 から Int4 までの各信号は，原因レジスタのビット 10 からビット 14 に入力されているので，原因レジスタから直接調べられます。

なお，原因レジスタの IP0 と IP1 は，ソフトウエア割り込みに関係しています（詳細は2.5　**例外処理**を参照してください）。ソフトウエア割り込みはハードウエアでセット／クリアできません。

図 2-90　ハードウエア割り込み要求信号

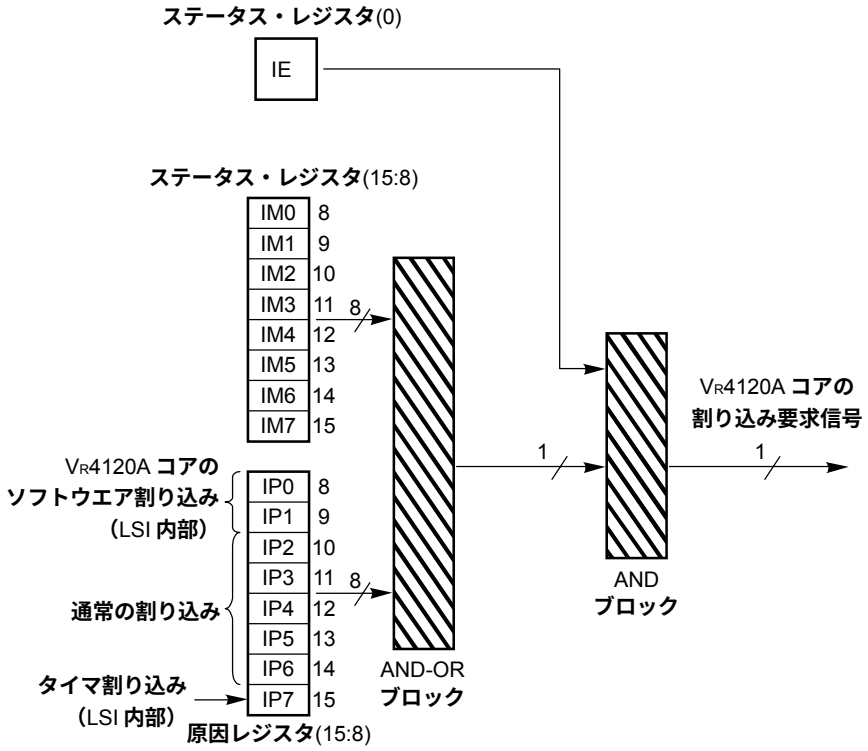


2.8.5.2 割り込み要求信号のマスク

図 2-91に，割り込み要求信号をマスクする方法を示します。原因レジスタ，ステータス・レジスタについては，2.5.3.5 ステータス・レジスタ（12），2.5.3.6 原因レジスタ（13）を参照してください。

- 原因レジスタのビット 15 からビット 8 まで（IP7 から IP0 まで）が，ステータス・レジスタの割り込みマスク・ビット（ビット 15 からビット 8 まで，すなわち IM7 から IM0 まで）と，それぞれ AND-OR をとって接続されており，個別に割り込み要求信号をマスクします。
- ステータス・レジスタのビット 0 は，グローバル割り込み許可（IE）ビットです。このビットの出力と，図 2-91に示す AND-OR 論理ブロックの出力との AND をとって，VR4120A コアの割り込み要求信号を生成しています。さらに，ステータス・レジスタの EXL ビットによって，これらの割り込みを許可します。

図 2-91 割り込み要求信号のマスク



ビット	内　容	設　定
IE	すべての割り込みに対する許可	1：許可 0：禁止
IM(7:0)	割り込みマスク	各ビットそれぞれ　1：許可 0：禁止
IP(7:0)	割り込み要求	各ビットそれぞれ　1：要求あり 0：要求なし

保守／廃止

第3章　システム・コントローラ

3.1 概　　要

このブロックは，μPD98502の内部システム・コントローラです。システム・コントローラは，CPUシステム・バス“SysAD”，NECのオリジナル高速内蔵バス“IBUS”，SDRAM/PROM/フラッシュ・メモリ用のメモリ・バス間のブリッジング機能を提供します。

システム・コントローラの特徴は，次のとおりです。

- SysADバス，IBUS，メモリのあいだでバスのブリッジング機能を提供
- SysADバス上のエンディアン変換機能をサポート
- SDRAMとPROM/フラッシュ・メモリに直接接続可能
- ウォッチドッグ・タイマと独立した2チャネル・タイマをサポート
- NS16550準拠のUARTをサポート

3.1.1 CPUインタフェース

- VR4120AのCPUバス“SysADバス”に直接接続可能
- 66 MHzまたは100 MHzのいずれかのVR4120Aのバス・サイクルをサポート
- データ・レートDのみをサポート
- シーケンシャル・オーダリングのみをサポート
- 4ワード（16バイト）×4エントリ・ライト・コマンド・バッファを内蔵
- リトル・エンディアンまたはビッグ・エンディアン・バイト・オーダをサポート
- SysADバス上で8ワードのバースト・リード／ライトはサポートしない

3.1.2 メモリ・インタフェース

- 66 MHzまたは100 MHzメモリ・バス
- 最大32 Mバイトのベース・メモリ・レンジのSDRAMをサポート
- 最大8 Mバイトのライト・プロテクタブル・ブート・メモリ・レンジのPROM/フラッシュ・メモリをサポート
- プログラマブルSDRAMリフレッシュ・コントローラ内蔵
- 4ワード（16バイト）ライト・データ・バッファ
- 4ワード（16バイト）プリフェッチ・データ・バッファ（メモリからCPU）
- SDRAMのデータ・バスとPROM/フラッシュ・メモリのデータ・バスは兼用
- 可変フラッシュ・メモリ・データ・バス（8, 16, 32ビット）
- プログラマブル・メモリ・バス・アービトレーション
- メモリに対するプログラマブル・アドレス・レンジ
- プログラマブルRAS-CAS遅延（2, 3, 4クロック）
- プログラマブルCASレイテンシ（2, 3クロック）

保守／廃止

3. 1. 3　IBUSインタフェース

- マスタとターゲットとして動作
- 64ワード（256バイト）IBUSスレーブTxFIFO（IBUSはメモリからデータを読み取る）
- 64ワード（256バイト）IBUSスレーブRxFIFO（IBUSはメモリにデータを書き込む）
- 4ワード（16バイト）IBUSマスタTxFIFO（VR4120AはIBUSからデータを読み取る）
- 4ワード（16バイト）×4エントリIBUSマスタRxFIFO（VR4120AはIBUSにデータを書き込む）
- IBUSのストールを検知するためバス・タイマをサポート
- 66 MHzのIBUSクロック・レート
- IBUS上で266 Mバイト／秒（32ビット@66 MHz）のバーストをサポート
- メモリとIBUSスレーブ・インタフェース間のエンディアン変換をサポート
- SysADバスとIBUSマスタ・インタフェース間のエンディアン変換をサポート

3. 1. 4　UART

- ユニバーサル非同期レシーバ／トランスミッタ
- モデム制御機能
- 偶数，あるいは奇数パリティ生成／検出機能
- 優先制御可能な割り込み機能

3. 1. 5　EEPROM

- 165/250 kHzクロック・レート（66 MHzまたは100 MHzのCPUクロック・レートに依存）
- 3.3 V EEPROMのみをサポート（National SemiconductorのNM93C46を推奨）
- シリアルEEPROMに対するMicro Wireインタフェースをサポート
- システム・ブートで2つのMACアドレスをオート・ローディングする機能をサポート

3. 1. 6　タ イ マ

- CPUに対する割り込みを発生するローディング可能な2つの32ビット汎用タイマ

3. 1. 7　割り込みコントローラ

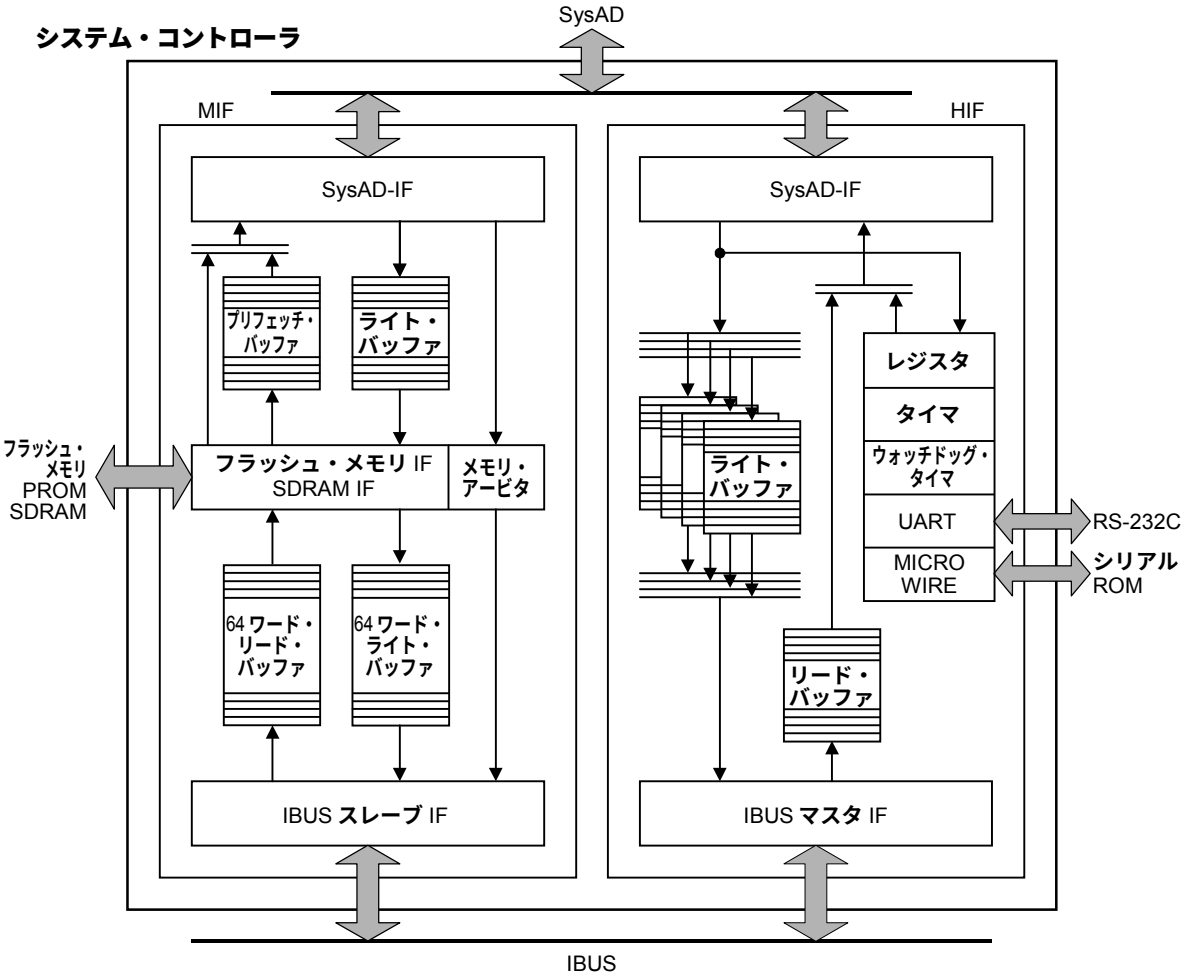
- NMIとINTを発生
- すべての割り込み要因はマスク可能

3. 1. 8　ウォッチドッグ・タイマ

- ウォッチドッグ・タイマは，CPUに対するコールド・リセットを生成

保守／廃止

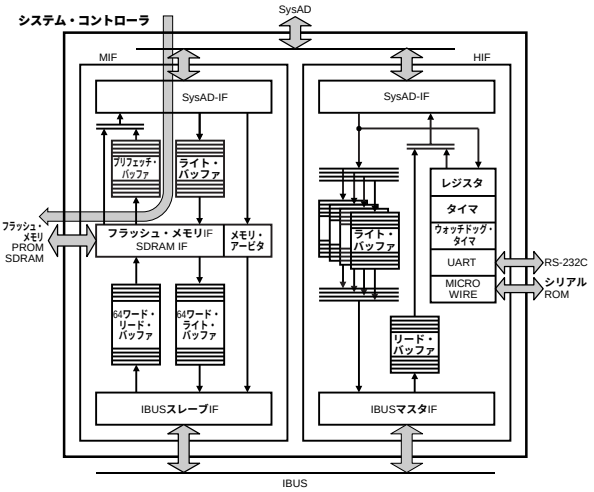
3. 1. 9　システム・ブロック図



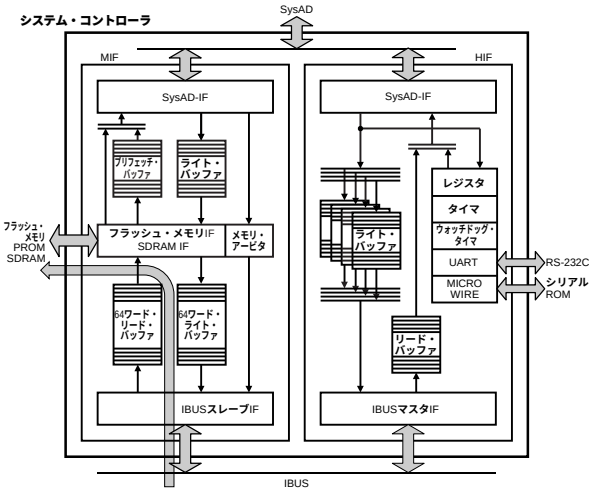
保守／廃止

3. 1. 10　データ・フロー図

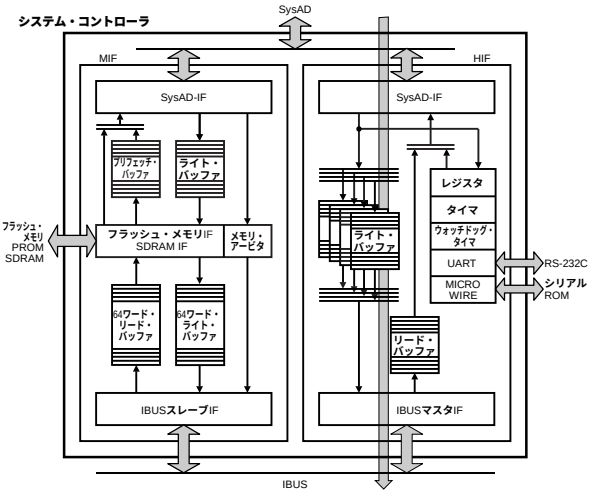
VR4120Aコアから SDRAM



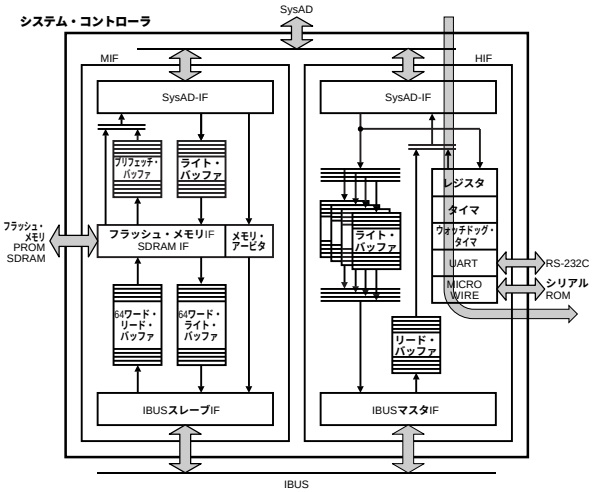
IBUSからSDRAM



VR4120Aコアから IBUS



VR4120Aコアから UART



保守／廃止

3. 2　レジスタ

3. 2. 1　レジスタ・マップ

システム・コントローラのレジスタ・セットを以下の表に示します。セットに対するベース・アドレスは，物理アドレス空間における1000_0000Hです。

アドレス	レジスタ名	R/W	アクセス	説　明
1000_0000H	S_GMR	R/W	W/H/B	ジェネラル・モード・レジスタ
1000_0004H	S_GSR	R	W/H/B	ジェネラル・ステータス・レジスタ
1000_0008H	S_ISR	RC	W/H/B	割り込みステータス・レジスタ
1000_000CH	S_IMR	R/W	W/H/B	割り込みマスク・レジスタ
1000_0010H	S_NSR	RC	W/H/B	NMIステータス・レジスタ
1000_0014H	S_NER	R/W	W/H/B	NMIイネーブル・レジスタ
1000_0018H	S_VER	R	W/H/B	バージョン・レジスタ
1000_001CH	S_IOR	R/W	W/H/B	I/Oポート・レジスタ
1000_0020H： 1000_002CH	N/A	-	-	将来の使用のため予約
1000_0030H	S_WRCR	W	W/H/B	ウォーム・リセット制御レジスタ
1000_0034H	S_WRSR	R	W/H/B	ウォーム・リセット・ステータス・レジスタ
1000_0038H	S_PWCR	R/W	W/H/B	電力制御レジスタ
1000_003CH	S_PWSR	R	W/H/B	電力ステータス・レジスタ
1000_0040H： 1000_0048H	N/A	-	-	将来の使用のため予約
1000_004CH	ITCNTR	R/W	W/H/B	IBUSタイムアウト・タイマ制御レジスタ
1000_0050H	ITSETR	R/W	W/H/B	IBUSタイムアウト・タイマ設定レジスタ
1000_0054H： 1000_007CH	N/A	-	-	将来の使用のため予約
1000_0080H	UARTRBR	R	W/H/B	UARTレシーバ・バッファ・レジスタ [DLAB = 0，リード]
1000_0080H	UARTTHR	W	W/H/B	UARTトランスミッタ・データ保持レジスタ [DLAB = 0，ライト]
1000_0080H	UARTDLL	R/W	W/H/B	UARTディバイザ・ラッチLSBレジスタ [DLAB = 1]
1000_0084H	UARTIER	R/W	W/H/B	UART割り込みイネーブル・レジスタ［DLAB = 0]
1000_0084H	UARTDLM	R/W	W/H/B	UARTディバイザ・ラッチMSBレジスタ [DLAB = 1]
1000_0088H	UARTIIR	R	W/H/B	UART割り込みIDレジスタ［リード]
1000_0088H	UARTFCR	W	W/H/B	UART FIFO制御レジスタ［ライト]
1000_008CH	UARTLCR	R/W	W/H/B	UARTライン制御レジスタ
1000_0090H	UARTMCR	R/W	W/H/B	UARTモデム制御レジスタ
1000_0094H	UARTLSR	R/W	W/H/B	UARTライン・ステータス・レジスタ
1000_0098H	UARTMSR	R/W	W/H/B	UARTモデム・ステータス・レジスタ
1000_009CH	UARTSCR	R/W	W/H/B	UARTスクラッチ・レジスタ
1000_00A0H	DSUCNTR	R/W	W/H/B	ウォッチドッグ・タイマ制御レジスタ
1000_00A4H	DSUSETR	R/W	W/H/B	ウォッチドッグ・タイマ・タイム設定レジスタ
1000_00A8H	DSUCLRR	W	W/H/B	ウォッチドッグ・タイマ・クリア・レジスタ

保守／廃止

アドレス	レジスタ名	R/W	アクセス	説　　明
1000_00ACH	DSUTIMR	R	W/H/B	ウォッチドッグ・タイマ経過時間レジスタ
1000_00B0H	TMMR	R/W	W/H/B	タイマ・モード・レジスタ
1000_00B4H	TM0CSR	R/W	W/H/B	タイマ・チャネル0カウント設定レジスタ
1000_00B8H	TM1CSR	R/W	W/H/B	タイマ・チャネル1カウント設定レジスタ
1000_00BCH	TM0CCR	R	W/H/B	タイマ・チャネル0カレント・カウント・レジスタ
1000_00C0H	TM1CCR	R	W/H/B	タイマ・チャネル1カレント・カウント・レジスタ
1000_00C4H： 1000_00CCH	N/A	-	-	将来の使用のため予約
1000_00D0H	ECCR	W	W/H/B	EEPROMコマンド制御レジスタ
1000_00D4H	ERDR	R	W/H/B	EEPROMリード・データ・レジスタ
1000_00D8H	MACAR1	R	W/H/B	MACアドレス・レジスタ1
1000_00DCH	MACAR2	R	W/H/B	MACアドレス・レジスタ2
1000_00E0H	MACAR3	R	W/H/B	MACアドレス・レジスタ3
1000_00E4H： 1000_00FCH	N/A	-	-	将来の使用のため予約
1000_0100H	RMMDR	R/W	W	ROMモード・レジスタ
1000_0104H	RMATR	R/W	W	ROMアクセス・タイミング・レジスタ
1000_0108H	SDMDR	R/W	W	SDRAMモード・レジスタ
1000_010CH	SDTSR	R/W	W	SDRAMタイプ選択レジスタ
1000_0110H	SDPTR	R/W	W	SDRAMプリチャージ・タイミング・レジスタ
1000_0114H： 1000_0118H	N/A	-	-	将来の使用のため予約
1000_011CH	SDRMR	R/W	W	SDRAMリフレッシュ・モード・レジスタ
1000_0120H	SDRCR	R	W	SDRAMリフレッシュ・タイマ・カウント・レジスタ
1000_0124H	MBCR	R/W	W	メモリ・バス制御レジスタ
1000_0128H： 1000_0FFCH	N/A	-	-	将来の使用のため予約

備考1. R/Wフィールド内で，

“W”は，“ライト可能”を意味します。

“R”は，“リード可能”を意味します。

“RC”は，“リード・クリア”を意味します。

“-”は，“アクセス不可”を意味します。

2. すべての内部レジスタは，32ビット・ワード配列のレジスタです。
3. 内部レジスタに対するバースト・アクセスは，禁止されています。バースト・アクセスが発生すると，NSRのIRERRビットがセットされ，NMIがCPUに対してアサートされます。
4. 予約領域に対するリード・アクセスが発生すると，NSRレジスタのCBERRビットがセットされ，SysCMD［0］にデータ・エラー・ビットが設定されたダミー読み取り応答データが返信されます（一部の非公開レジスタを除きます）。
5. 予約領域に対するライト・アクセスが発生すると，NSRレジスタのCBERRビットがセットされ，書き込みデータが失われます（一部の非公開レジスタを除きます）。

保守／廃止

備考6.　“アクセス”フィールド内で，

“W”は，ワード・アクセスが有効であることを意味します。

“H”は，ハーフ・ワード・アクセスが有効であることを意味します。

“B”は，バイト・アクセスが有効であることを意味します。

7. リード・オンリー・レジスタに対するライト・アクセスはエラーになりませんが，書き込みデータは失われます。

8. CPUはすべての内部レジスタにアクセスできますが，IBUSマスタ・デバイスがそれらのレジスタにアクセスすることはできません。

3. 2. 2　S_GMR（ジェネラル・モード・レジスタ）

ジェネラル・モード・レジスタ“S_GMR”は，リード／ライト可で32ビット・ワード配列のレジスタです。初期化のあと，VR4120AがIAENビット（ビット1）をセットしてIBUSアービタをイネーブルにします。S_GMRには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：10	Reserved	R/W	0	0固定
9	MSWP	R/W	0	MIFブロック・データ・スワップ機能イネーブル： 0 = ディスエーブル 1 = イネーブル
8	HSWP	R/W	0	HIFブロック・データ・スワップ機能イネーブル： 0 = ディスエーブル 1 = イネーブル
7：4	Reserved	R/W	0	0固定
3	UCSEL	R/W	0	UARTソース・クロック選択： 0 = CPUクロックの1/2を使用 1 = 外部クロック（18.432 MHz）を使用
2	MPFD	R/W	0	メモリからCPUへのプリフェッチFIFOディスエーブル： 0 = イネーブル 1 = ディスエーブル
1	IAEN	R/W	0	IBUSアービタ・イネーブル： 0 = ディスエーブル（IBUSアービタはシステム・コントローラを除いて許可しない） 1 = イネーブル
0	CRST	R/W	0	コールド・リセット： 0 = 何もしない 1 = コールド・リセットを実行（ハードウエア・システム・リセットと同じ）

S_GMRの“HSWP”ビットは，SysADインタフェースとIBUSマスタ・インタフェースの間のエンディアン・コンバータのイネーブラであり，IBUSターゲット領域に対してのみ動作します。このコンバータは，アドレス・スワップ・モードの場合のみ有効です。このコンバータは，以下のデータ操作を行います。

保守／廃止

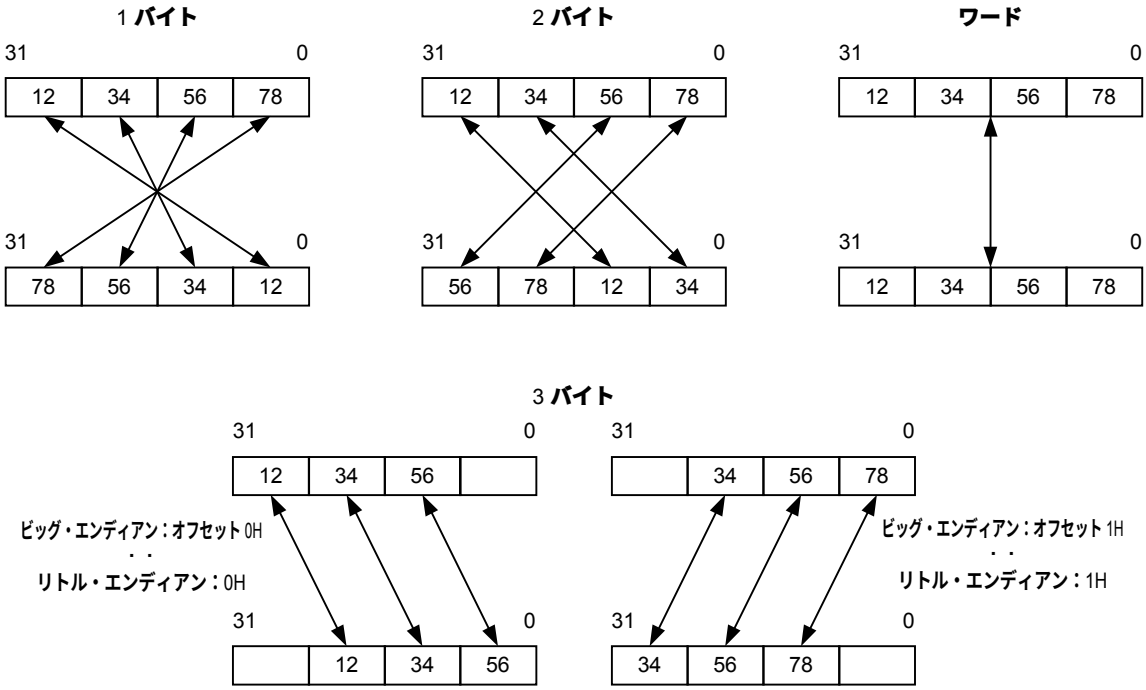
表3－1　データ・スワップ・モードでのエンディアン変換表（マスタ）

GMRの HSWP	データ・サイズ	オフセット・アドレス［1：0］注	データ・フェーズ内の変換前入力データ［31：0］	データ・フェーズ内の変換後出力データ［31：0］	備考
0	任意	任意	[31：0]	[31：0]	-
1	1ワード以上	0			
	1バイト	0, 1, 2, 3	[31：24]　[23：16]　[15：8]　[7：0]	[7：0]　[15：8]　[23：16]　[31：24]	-
	2バイト	0, 2	[31：16]　[15：0]	[15：0]　[31：16]	-
	3バイト	0	[31：8]　[7：0]	[31：24]　[23：0]	-
		1	[31：24]　[23：0]	[31：8]　[7：0]	-

注　このオフセット・アドレス［1：0］は，ビッグ・エンディアン・モード上の表記です。

次の図で，上側はSysADバスの4つのオクテット・データです。また，下側はIBUSマスタ・インタフェースの4つのオクテット・データです。

図3－1　データ・スワップ・モードでのエンディアン変換（マスタ）



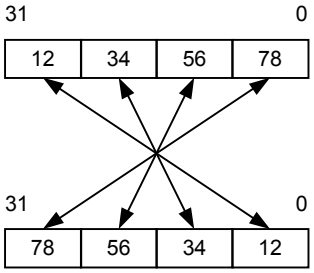
S_GMRレジスタの“MSWP”ビットは，メモリ・インタフェースとIBUSスレーブ・インタフェースの間のエンディアン・コンバータのイネーブラであり，IBUSスレーブ・インタフェースを介したメモリ・アクセスに対してのみ動作します。このコンバータは，アドレス・スワップ・モードの場合のみ有効です。このコンバータは，以下のデータ操作を行います。

表3−2　データ・スワップ・モードでのエンディアン変換表（スレーブ）

GMRのMSWP	データ・フェーズ内の変換前入力データ［31：0］	データ・フェーズ内の変換後出力データ［31：0］	備考
0	［31：0］	［31：0］	-
1	［31：24］　［23：16］　［15：8］　［7：0］	［7：0］　［15：8］　［23：16］　［31：24］	-

次の図で，上側はメモリ・インタフェースの4つのオクテット・データです。また，下側はIBUSスレーブ・インタフェースの4つのオクテット・データです。

図3−2　データ・スワップ・モードでのエンディアン変換（スレーブ）



保守／廃止

3. 2. 3　S_GSR（ジェネラル・ステータス・レジスタ）

ジェネラル・ステータス・レジスタ“S_GSR”は，リード・オンリーで32ビット・ワード配列のレジスタです。S_GSRは，μPD98502の外部端子のステータスを示します。S_GSRには以下のフィールドがあります。

ビット	フィールド	R/W	ディフォールト	説　明
31：3	Reserved	R	0	0固定
2	MIPS16	R	-	リセット後の外部端子“MIPS16”のステータスを反映： このフィールドは，CPUコンフィギュレーション・レジスタのM16ビットと同じ値を示します。 0 = GNDに接続，MIPS16モードがディスエーブルであることを意味します[注]。 1 = VCCに接続，MIPS16モードがイネーブルであることを意味します。
1	CLKSL	R	-	リセット後の外部端子“CLKSL”のステータスを反映： 0 = GNDに接続，CPUが100 MHzで動作することを意味します。 1 = VCCに接続，CPUが66 MHzで動作することを意味します。
0	ENDCEN	R	-	リセット後の外部端子“ENDCEN”のステータスを反映： 0 = GNDに接続，エンディアン・コンバータがディスエーブルであることを意味します。 1 = VCCに接続，エンディアン・コンバータがイネーブルであることを意味します。

注　μPD98502は，MIPS16モードをサポートしません。MIPS16モード端子（D15に配置）は，GNDに接続する必要があります。

保守／廃止

3. 2. 4　S_ISR（割り込みステータス・レジスタ）

割り込みステータス・レジスタ“S_ISR”は，リード・クリアで32ビット・ワード配列のレジスタです。S_ISRは，SysAD/IBUSインタフェース，タイマ，UARTなどからの割り込み状態を示します。S_IMR（割り込みマスク・レジスタ）内の対応するビットがセットされ，割り込みがマスクされていないと，システム・コントローラは割り込み信号を使ってVR4120Aに割り込みます。S_ISR内のビットは，VR4120Aによって読み取られたあとリセットされます。ビットが読み取られる前に同じ割り込み要因が発生すると，ビットは再びセットされます。S_ISRには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：18	Reserved	RC	0	0固定
17	MAC2IS	RC	0	イーサネット・コントローラ#2割り込み： 0 = 保留中のイーサネット・コントローラ#2割り込みなし 1 = イーサネット・コントローラ#2割り込み保留中
16	PCIIS	RC	0	PCI割り込み： 0 = 保留中のPCI割り込みなし 1 = PCI割り込み保留中
15：5	Reserved	RC	0	0固定
4	WUIS	RC	0	ウェークアップ割り込み： 0 = 保留中のウェークアップ・リクエストなし 1 = ウェークアップ・リクエスト保留中
3	EXTIS	RC	0	外部割り込み： 0 = 保留中の外部割り込みなし 1 = 外部割り込み保留中
2	UARTIS	RC	0	UART割り込み： 0 = 保留中のUART割り込みなし 1 = UART割り込み保留中 UART割り込みは，以下の割り込みのいずれかです。 1. UART受信データ・バッファ・フル割り込み 2. UARTトランスミッタ・バッファ・エンプティ割り込み 3. UARTライン・ステータス割り込み 4. UARTモデム・ステータス割り込み
1	TM1IS	RC	0	タイマCH1割り込み： 0 = 保留中のタイマCH1割り込みなし 1 = タイマCH1割り込み保留中
0	TM0IS	RC	0	タイマCH0割り込み： 0 = 保留中のタイマCH0割り込みなし 1 = タイマCH0割り込み保留中

備考1. このレジスタのビットはリード・クリアされますが，新しく割り込みを発生させるためには，割り込みが発生した要因をリセット（TM0ISならばカウンタの再セット，起動など）する必要があります。

2. イーサネット・コントローラ#2割り込みとPCI割り込みは，システム・コントローラでマスクできません。

3. MAC2IS/PCIISビットがセットされたとき，このレジスタをリードするとビットはクリアされますが，VR4120Aに対して割り込み信号（intb［4］）を送り続けます。したがって，割り込みをインアクティブにするため，それぞれの割り込み要因の処理（イーサネット・コントローラ#2であればE2_ISRを読むなど）が必要となります。

保守／廃止

3. 2. 5　S_IMR（割り込みマスク・レジスタ）

割り込みマスク・レジスタ“S_IMR”は，リード／ライト可で32ビット・ワード配列のレジスタです。S_IMRは，各々の対応する要因ごとに割り込みをマスクできます。S_ISR内の対応するビットと同じ位置にあるマスク・ビットは，各要因による割り込みの生成を制御します。このレジスタのビットが0にリセットされると，S_ISR内の対応するビットはマスクされます。1にセットされると，対応するビットはマスクされません。マスク解除にセットされ，S_ISR内のビットがセットされると，システム・コントローラはVR4120Aに対して割り込み信号をアサートします。S_IMRには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　　明
31：5	Reserved	R/W	0	0固定
4	WUIM	R/W	0	ウェークアップ割り込みマスク： 1 = マスク解除 0 = マスク
3	EXTIM	R/W	0	外部割り込みマスク： 1 = マスク解除 0 = マスク
2	UARTIM	R/W	0	UART割り込みマスク： 1 = マスク解除 0 = マスク
1	TM1IM	R/W	0	タイマCH1割り込みマスク： 1 = マスク解除 0 = マスク
0	TM0IM	R/W	0	タイマCH0割り込みマスク： 1 = マスク解除 0 = マスク

備考　イーサネット・コントローラ#2割り込みとPCI割り込みは，システム・コントローラでマスクできません。

保守／廃止

3. 2. 6　S_NSR（NMIステータス・レジスタ）

NMIステータス・レジスタ“S_NSR”は，リード・クリアで32ビット・ワード配列のレジスタです。S_NSRは，SysAD/IBUSインタフェース，外部NMI，メモリ・インタフェースなどからのノンマスカブル割り込み状態を示します。S_NER（NMIイネーブル・レジスタ）内の対応するビットがセットされ，NMIがイネーブルにされていると，システム・コントローラはノンマスカブル割り込み信号をVR4120Aに入力します。S_NSR内のビットは，VR4120Aが読み出したあとリセットされます。ビットを読み出す前に同じ割り込み要因が発生すると，ビットは再びセットされます。S_NSRには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：6	Reserved	RC	0	0固定
5	IRERR	RC	0	不正内部レジスタ・アクセス・エラー： 0 = 該当アクセスなし。 1 = 不正内部レジスタ・アクセスあり。たとえば，バースト・アクセスが行われた場合など。
4	EXTNMI	RC	0	外部NMI： 0 = 外部NMIがアサートされていない。 1 = 外部NMIがアサートされた。
3	MAERR	RC	0	メモリ・アドレス・エラー： メモリ・アドレス・エラーとは，不正メモリ・スペース（予約領域とSDRAM/ROMスペースの範囲外）へのメモリ・アクセスと不正メモリ・アクセス（バイトまたはハーフ・ワードROMアクセスあるいはROMに対するバースト・ライト・アクセス）を含みます。 0 = 該当エラーなし。 1 = メモリ・アクセス中にアドレス・レンジ・エラーが発生。
2	ITERR	RC	0	IBUSタイムアウト・エラー： IBUSタイムアウト・エラーは，IBUSがストールしたときに発生します。 0 = 該当エラーなし。 1 = IBUSタイムアウト・エラー。
1	IBERR	RC	0	IBUSバス・エラー： IBUSバス・エラーは，VR4120AがIBUSターゲット・アドレス・スペース内の予約領域にアクセスしたときに発生します（1. 9　**メモリ・マップ**参照）。 0 = 該当エラーなし。 1 = IBUSマスタ・アクセス中にバス・エラーが発生した。
0	CBERR	RC	0	CPU（VR4120A）バス・エラー： VR4120Aバス・エラーとは，不正バス・コマンド，不正データ配置，不正バス・サイズ，レジスタ・スペース内の予約領域に対する不正アクセスを含みます。 0 = 該当エラーなし。 1 = VR4120Aバス・エラー。

備考1.　このレジスタをクリアするためには，VR4120AはCBERRレジスタを含むバイトをすべて読み込む必要があります。

2.　外部NMIのアサートを検出した場合，次に外部NMIのアサートを検出するためには，いったん外部NMI信号をディアサートしておく必要があります。

保守／廃止

3. 2. 7　S_NER（NMIイネーブル・レジスタ）

NMIイネーブル・レジスタ“S_NER”は，リード／ライト可で32ビット・ワード配列のレジスタです。S_NERは，各々の対応する要因ごとにNMIをイネーブルにします。S_NSR内の対応するビットと同じ位置にあるイネーブル・ビットは，各要因による割り込みの生成を制御します。このレジスタのビットが0にリセットされると，S_NSR内の対応するビットはNMI割り込みをマスクされます。1にセットされると，対応するビットはNMI割り込みが可能になります。イネーブルにセットされ，S_NSR内のビットがセットされると，システム・コントローラはVR4120Aに対してNMI割り込み信号をアサートします。S_NERには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：6	Reserved	R/W	0	0固定
5	IRERRE	R/W	0	不正内部レジスタ・アクセス・エラー・イネーブル： 1 = イネーブル 0 = ディスエーブル
4	EXTNMIE	R/W	0	外部NMIイネーブル： 1 = イネーブル 0 = ディスエーブル
3	MAERRE	R/W	0	メモリ・アドレス・エラー・イネーブル： 1 = イネーブル 0 = ディスエーブル
2	ITERRE	R/W	0	IBUSタイムアウト・エラー・イネーブル： 1 = イネーブル 0 = ディスエーブル
1	IBERRE	R/W	0	IBUSバス・エラー・イネーブル： 1 = イネーブル 0 = ディスエーブル
0	CBERRE	R/W	0	CPUバス・エラー・イネーブル： 1 = イネーブル 0 = ディスエーブル

3. 2. 8　S_VER（バージョン・レジスタ）

バージョン・レジスタ“S_VER”は，リード・オンリーで32ビット・ワード配列のレジスタです。S_VERは，μPD98502のバージョン・ナンバを示します。S_VERには以下のフィールドがあり，リセット時に300Hに初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：16	Reserved	R	0	0固定
15：8	MAJOR	R	03H	大規模な改訂を行った場合にアップデートします。K規格品は03H固定。
7：0	MINOR	R	00H	小規模な改訂を行った場合にアップデートします。K規格品は00H固定。

保守／廃止

3. 2. 9　S_IOR（I/Oポート・レジスタ）

I/Oポート・レジスタ“S_IOR”は，リード／ライト可で32ビット・ワード配列のレジスタです。I/Oポート・レジスタは，外部パラレル・ポート出力の制御に使われます。以下に示すPOM_OUTフィールドの各ビットは，外部I/Oポート（POM［7：0］）に直接接続されます。S_IORには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：8	Reserved	R/W	0	0固定
7	POM_OUT7	R/W	0	POM7端子に対する出力レベルを設定： 0 = POM7をディアサート（ロウ）する。 1 = POM7をアサート（ハイ）する。
6	POM_OUT6	R/W	0	POM6端子に対する出力レベルを設定： 0 = POM6をディアサート（ロウ）する。 1 = POM6をアサート（ハイ）する。
5	POM_OUT5	R/W	0	POM5端子に対する出力レベルを設定： 0 = POM5をディアサート（ロウ）する。 1 = POM5をアサート（ハイ）する。
4	POM_OUT4	R/W	0	POM4端子に対する出力レベルを設定： 0 = POM4をディアサート（ロウ）する。 1 = POM4をアサート（ハイ）する。
3	POM_OUT3	R/W	0	POM3端子に対する出力レベルを設定： 0 = POM3をディアサート（ロウ）する。 1 = POM3をアサート（ハイ）する。
2	POM_OUT2	R/W	0	POM2端子に対する出力レベルを設定： 0 = POM2をディアサート（ロウ）する。 1 = POM2をアサート（ハイ）する。
1	POM_OUT1	R/W	0	POM1端子に対する出力レベルを設定： 0 = POM1をディアサート（ロウ）する。 1 = POM1をアサート（ハイ）する。
0	POM_OUT0	R/W	0	POM0端子に対する出力レベルを設定： 0 = POM0をディアサート（ロウ）する。 1 = POM0をアサート（ハイ）する。

保守／廃止

3. 2. 10　S_WRCR（ウォーム・リセット制御レジスタ）

ウォーム・リセット制御レジスタ“S_WRCR”は，ライト・オンリーで32ビット・ワード配列のレジスタです。S_WRCRは，ウォーム・リセット・リクエストを，USBコントローラ，イーサネット・コントローラ，ATMセル・プロセッサ，UART, PCIコントローラに対して個別に生成します。S_WRCRには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：6	Reserved	W	0	0固定
5	PCIWR	W	0	PCIコントローラに対するウォーム・リセット・リクエスト： 0 = ウォーム・リセットを実行しない。 1 = ウォーム・リセットを実行。
4	UARTWR	W	0	UARTに対するウォーム・リセット・リクエスト： 0 = ウォーム・リセットを実行しない。 1 = ウォーム・リセットを実行。
3	MAC2WR	W	0	イーサネット・コントローラ#2に対するウォーム・リセット・リクエスト： 0 = ウォーム・リセットを実行しない。 1 = ウォーム・リセットを実行。
2	ATMWR	W	0	ATMセル・プロセッサに対するウォーム・リセット・リクエスト： 0 = ウォーム・リセットを実行しない。 1 = ウォーム・リセットを実行。
1	MACWR	W	0	イーサネット・コントローラ#1に対するウォーム・リセット・リクエスト： 0 = ウォーム・リセットを実行しない。 1 = ウォーム・リセットを実行。
0	USBWR	W	0	USBコントローラに対するウォーム・リセット・リクエスト： 0 = ウォーム・リセットを実行しない。 1 = ウォーム・リセットを実行。

備考　リード・アクセスを行うと，このレジスタ内のすべてのフィールドは，0として読み出されます。

保守／廃止

3. 2. 11　S_WRSR（ウォーム・リセット・ステータス・レジスタ）

ウォーム・リセット・ステータス・レジスタ“S_WRSR”は，リード・オンリーで32ビット・ワード配列のレジスタです。S_WRSRは，USBコントローラ，イーサネット・コントローラ，ATMセル・プロセッサ，UART，PCIコントローラからの応答を個別に示します。S_WRSRには以下のフィールドがあります。

ビット	フィールド	R/W	ディフォールト	説　明
31：6	Reserved	R	0	0固定
5	PCIWRST	R	-	PCIコントローラからのウォーム・リセット状態を示す： 0 = PCIコントローラはウォーム・リセットを実行中。 1 = ウォーム・リセット終了。PCIコントローラはレディ状態。
4	UARTWRST	R	-	UARTからのウォーム・リセット状態を示す： 0 = UARTはウォーム・リセットを実行中。 1 = ウォーム・リセット終了。UARTはレディ状態。
3	MAC2WRST	R	-	イーサネット・コントローラ#2からのウォーム・リセット状態を示す： 0 = イーサネット・コントローラ#2はウォーム・リセットを実行中。 1 = ウォーム・リセット終了。MACイーサネット・コントローラ#2はレディ状態。
2	ATMWRST	R	-	ATMセル・プロセッサからのウォーム・リセット状態を示す： 0 = ATMセル・プロセッサはウォーム・リセットを実行中。 1 = ウォーム・リセット終了。ATMセル・プロセッサはレディ状態。
1	MACWRST	R	-	イーサネット・コントローラ#1からのウォーム・リセット状態を示す： 0 = イーサネット・コントローラ#1はウォーム・リセットを実行中。 1 = ウォーム・リセット終了。MACイーサネット・コントローラ#1はレディ状態。
0	USBWRST	R	-	USBコントローラからのウォーム・リセット状態を示す： 0 = USBコントローラはウォーム・リセットを実行中。 1 = ウォーム・リセット終了。USBコントローラはレディ状態。

備考　このレジスタのディフォールト値は，システム・リセット後，およびS_WRCRによる各周辺ブロックのウォーム・リセット後の値を示します。

通常は，リセット処理が高速に行われるため，このレジスタをリードすると1が読み出されます。しかし，ATMセル・プロセッサはファームウエアのダウンロードと実行によってレディ状態となるため，0が読み出される場合もあります。

各ブロックにクロック供給を行わなかった場合も同様で，ディフォールト値の定義が困難なため，ディフォールト値の記載は行いません。

保守／廃止

3. 2. 12　S_PWCR（電力制御レジスタ）

電力制御レジスタ“S_PWCR”は，リード／ライト可で32ビット・ワード配列のレジスタです。S_PWCRは，以下の各IDRQフィールドをセットすることにより，USBコントローラ，イーサネット・コントローラ，ATMセル・プロセッサ，PCIコントローラに対してアイドル状態を保つように要求します。VR4120Aは，これらのブロックをアイドル状態に保つように要求したあと，以下の各STOPフィールドをセットしてサスペンドを行う前に，電力ステータス・レジスタ“S_PWSR”を読み取ることで，それらの応答を確認する必要があります。S_PWCRには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：22	Reserved	R/W	0	0固定
21	PCISTOP	R/W	0	PCIコントローラに対するサスペンド・リクエスト： 0 = PCIコントローラに対するシステム・クロック供給をイネーブルにする。 1 = PCIコントローラに対するシステム・クロック供給をディスエーブルにする。
20	Reserved	R/W	0	0固定
19	MAC2STOP	R/W	0	イーサネット・コントローラ#2に対するサスペンド・リクエスト： 0 = イーサネット・コントローラ#2に対するシステム・クロック供給をイネーブルにする。 1 = イーサネット・コントローラ#2に対するシステム・クロック供給をディスエーブルにする。
18	ATMSTOP	R/W	0	ATMセル・プロセッサに対するサスペンド・リクエスト： 0 = ATMセル・プロセッサに対するシステム・クロック供給をイネーブルにする。 1 = ATMセル・プロセッサに対するシステム・クロック供給をディスエーブルにする。
17	MACSTOP	R/W	0	イーサネット・コントローラ#1に対するサスペンド・リクエスト： 0 = イーサネット・コントローラ#1に対するシステム・クロック供給をイネーブルにする。 1 = イーサネット・コントローラ#1に対するシステム・クロック供給をディスエーブルにする。
16	USBSTOP	R/W	0	USBコントローラに対するサスペンド・リクエスト： 0 = USBコントローラに対するシステム・クロック供給をイネーブルにする。 1 = USBコントローラに対するシステム・クロック供給をディスエーブルにする。
15：6	Reserved	R/W	0	0固定
5	PCIIDRQ	R/W	0	PCIコントローラに対するアイドル・リクエスト： 0 = アイドル状態にしない。 1 = アイドル状態を保つように要求。
4	Reserved	R/W	0	0固定
3	MAC2IDRQ	R/W	0	イーサネット・コントローラ#2に対するアイドル・リクエスト： 0 = アイドル状態にしない。 1 = アイドル状態を保つように要求。

保守／廃止

ビット	フィールド	R/W	ディフォールト	説　明
2	ATMIDRQ	R/W	0	ATMセル・プロセッサに対するアイドル・リクエスト： ATMセル・プロセッサ内のRISCコアが，ATMIDRQフィールドのビットの状態をチェックします。このビットをセットすると，ATMセル・プロセッサがアイドル状態に移行します。移行が完了すると，S_PWSR（電力ステータス・レジスタ）のATMIDLEフィールド（ビット2）がセットされます。ATMセル・プロセッサ内のRISCコアは，電源投入後，VR4120AがA_INBAR（命令ベース・アドレス・レジスタ）を設定してから動作を開始します。RISCコアが正常に動作開始したあとに，S_PWCRレジスタへの設定が有効になります。 0 = アイドル状態にしない。 1 = アイドル状態を保つように要求。
1	MACIDRQ	R/W	0	イーサネット・コントローラ#1に対するアイドル・リクエスト： 0 = アイドル状態にしない。 1 = アイドル状態を保つように要求。
0	USBIDRQ	R/W	0	USBコントローラに対するアイドル・リクエスト： 0 = アイドル状態にしない。 1 = アイドル状態を保つように要求。

備考　このレジスタにアクセスする前に，VR4120AはIBUSターゲットを読み取ることで内部ライト・コマンド・バッファをクリアする必要があります。

保守／廃止

3. 2. 13　S_PWSR（電力ステータス・レジスタ）

電力ステータス・レジスタ“S_PWSR”は，リード・オンリーで32ビット・ワード配列のレジスタです。S_PWSRの各IDLEフィールドは，サスペンドの準備が整った状態であることを示します。S_PWSRの各WKUPフィールドは，ウェークアップ・リクエストを示します。IDLEフィールドのビットが1になると，VR4120Aは，S_PWCRのSTOPフィールドをセットすることで，対応するデバイスのシステム・クロック供給をディスエーブルにできます。S_PWSRのWKUPフィールドのビットが1になると，VR4120Aは，S_PWCRのSTOPフィールドをリセットすることで，対応するデバイスのシステム・クロック供給をイネーブルにする必要があります。S_PWSRには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：14	Reserved	R	0	0固定
13	PCIWKUP	R	0	PCIコントローラからのウェークアップ・リクエストを示す： 0 = 保留中のウェークアップ・リクエストなし。 1 = ウェークアップ・リクエスト保留中。
12	Reserved	R	0	0固定
11	MAC2WKUP	R	0	イーサネット・コントローラ#2からのウェークアップ・リクエストを示す： 0 = 保留中のウェークアップ・リクエストなし。 1 = ウェークアップ・リクエスト保留中。
10	ATMWKUP	R	注	ATMセル・プロセッサからのウェークアップ・リクエストを示す： 0 = 保留中のウェークアップ・リクエストなし。 1 = ウェークアップ・リクエスト保留中。
9	MACWKUP	R	0	イーサネット・コントローラ#1からのウェークアップ・リクエストを示す： 0 = 保留中のウェークアップ・リクエストなし。 1 = ウェークアップ・リクエスト保留中。
8	USBWKUP	R	0	USBコントローラからのウェークアップ・リクエストを示す： 0 = 保留中のウェークアップ・リクエストなし。 1 = ウェークアップ・リクエスト保留中。
7：6	Reserved	R	0	0固定
5	PCIIDLE	R	1	PCIコントローラのアイドル状態を示す： 0 = アイドル状態ではない。PCIコントローラはサスペンドする準備が整っていないことを意味する。 1 = アイドル状態。PCIコントローラはサスペンドする準備が整っていることを意味する。
4	Reserved	R	0	0固定
3	MAC2IDLE	R	0	イーサネット・コントローラ#2のアイドル状態を示す： 0 = アイドル状態ではない。イーサネット・コントローラ#2はサスペンドする準備が整っていないことを意味する。 1 = アイドル状態。イーサネット・コントローラ#2はサスペンドする準備が整っていることを意味する。

注　ATMWKUPフィールドのディフォールト値は，UMINT_B端子の状態に依存します。UTOPIAマネジメント・インタフェースに接続されたPHYデバイスから出力される割り込み信号（UMINT_B端子へ入力される信号）のレベルを反転した状態が読み出されます。

保守／廃止

ビット	フィールド	R/W	ディフォールト	説　　明
2	ATMIDLE	R	0	ATMセル・プロセッサのアイドル状態を示す： 0 = アイドル状態ではない。ATMセル・プロセッサはサスペンドする準備が整っていないことを意味する。 1 = アイドル状態。ATMセル・プロセッサはサスペンドする準備が整っていることを意味する。
1	MACIDLE	R	0	イーサネット・コントローラ#1のアイドル状態を示す： 0 = アイドル状態ではない。イーサネット・コントローラ#1はサスペンドする準備が整っていないことを意味する。 1 = アイドル状態。イーサネット・コントローラ#1はサスペンドする準備が整っていることを意味する。
0	USBIDLE	R	0	USBコントローラのアイドル状態を示す： 0 = アイドル状態ではない。USBコントローラはサスペンドする準備が整っていないことを意味する。 1 = アイドル状態。USBコントローラはサスペンドする準備が整っていることを意味する。

保守／廃止

3. 3　CPUインタフェース

システム・コントローラは，100 MHzまたは66 MHzで動作する32ビットのSysADバスを用いて，VR4120Aへのダイレクト・インタフェースを提供します。

3. 3. 1　概　　要

- VR4120AのCPUバス“ SysADバス ”にダイレクトに接続が可能
- 66 MHzまたは100 MHzいずれかのVR4120Aのバス・サイクルをサポート
- データ・レートDのみをサポート
- シーケンシャル・オーダリングのみをサポート
- 4ワード（16バイト）×4エントリ・ライト・コマンド・バッファを内蔵
- リトル・エンディアンまたはビッグ・エンディアン・バイト・オーダをサポート
- SysADバス上で8ワードのバースト・リード／ライトをサポートしない

3. 3. 2　データ・レート制御

VR4120Aからシステム・コントローラへのデータ・レートは， VR4120Aのコンフィギュレーション・レジスタのEPフィールド（ビット27-24）をセットすることでプログラムできます（2. 4. 6. 8　**コンフィギュレーション・レジスタ**（16）参照）。コントローラは，データ・レートDDのみをサポートします。したがって，VR4120Aのコンフィギュレーション・レジスタのADビットを1にセットしても，このブロックはADモードをサポートしません。

3. 3. 3　バースト・サイズ制御

このブロックからVR4120Aへのバースト・データ・サイズは，SysADバス上のOSysCMD［2：0］信号によって決定します。VR4120Aのコンフィギュレーション・レジスタのIBビット（ビット5）でプログラムできます（2. 4. 6. 8　**コンフィギュレーション・レジスタ**（16）参照）。μPD98502は，4ワードのバースト・モードのみをサポートします。VR4120Aのコンフィギュレーション・レジスタのIBビットを0にセットしてください。

3 .3. 4　アドレス・デコーディング

コントローラは，アドレスをSysADバスにラッチします。次に，アドレスとSysCmd信号をデコードしてトランザクションの種類を決定します。

- 外部ブートPROMまたはフラッシュ・メモリに対して1つのレンジ
- 外部SDRAMに対して1つのレンジ
- システム・コントローラの内部コンフィギュレーション・レジスタに対して1つのレンジ

ブートPROM/フラッシュ・メモリは，そのサイズに応じてマップされます。システム・コントローラの内部レジスタは，ベース・アドレス1000_0000Hに固定され，コンフィギュレーションされる前に，VR4120Aがブート中にアクセスできるようにします。すべてのほかのデコード・レンジはプログラム可能です。

保守／廃止

3. 3. 5　エンディアン変換

VR4120Aのコンフィギュレーション・レジスタのBEビットは，リセット時にバイト・オーダリングを指定します（2. 4. 6. 8　**コンフィギュレーション・レジスタ（16）参照**）。BE = 0はリトル・エンディアン・オーダのコンフィギュレーションを行い，BE = 1はビッグ・エンディアン・オーダのコンフィギュレーションを行います。エンディアン・モードは，“ BIG ”信号で制御します。システム・コントローラのVR4120Aインタフェースは，エンディアン・コンバータを用いてSysADバス上でビッグ・エンディアンとリトル・エンディアンの両方のバイト・オーダリングをサポートします。システム・コントローラ以外のすべてのインタフェースは，リトル・エンディアン・モードでのみ動作します。

VR4120Aがビッグ・エンディアン・モード（外部BIG端子がハイ）で動作するとき，システム・コントローラは外部ENDCEN端子で制御される2つのエンディアン変換方法を提供します。ENDCEN端子がロウの場合，システム・コントローラはSysADバス上でデータ・スワップを行います（データ・スワップ・モードについては，表3－4　**エンディアン・コンバータのエンディアン変換表**参照）。ENDCEN端子がハイの場合，システム・コントローラはSysADバス上でアドレス・スワップを行います（アドレス・スワップ・モードの詳細については，表3－4　**エンディアン・コンバータのエンディアン変換表**に記述してあります）。

表3－3　エンディアン・コンフィギュレーション表

BIG端子	ENDCEN端子	VR4120Aのステータス・レジスタのREフィールド[注]	VR4120Aのエンディアン	システム・コントローラのエンディアン	エンディアン・コンバータ動作
0	0	0	LITTLE	LITTLE	トランスペアレント
0	1	0	LITTLE	LITTLE	トランスペアレント
1	0	0	BIG	LITTLE	データ・スワップ・モード
1	1	0	BIG	LITTLE	アドレス・スワップ・モード

注　2. 5. 3. 5　**ステータス・レジスタ（12）参照**

備考　VR4120Aはリバース・エンディアン・モードをサポートしません。

保守／廃止

表3－4　エンディアン・コンバータのエンディアン変換表

CPUアクセス・タイプ		BIG	ENDCEN	SysAD［1：0］変換前	SysAD［1：0］変換後	注
ブロック	2ワード 4ワード	1	1	00	00	有効
				01	01	無効
				10	10	無効
				11	11	無効
シングル	1バイト	1	1	00	11	有効
				01	10	有効
				10	01	有効
				11	00	有効
シングル	2バイト	1	1	00	10	有効
				01	11	無効
				10	00	有効
				11	01	無効
シングル	3バイト	1	1	00	01	有効
				01	00	有効
				10	11	無効
				11	10	無効
シングル	4バイト	1	1	00	00	有効
				01	01	無効
				10	10	無効
				11	11	無効

CPUアクセス・タイプ	BIG	ENDCEN	SysAD［1：0］変換前	SysAD［1：0］変換後	注
任意	1	0	[31：24]　[23：16] [15：8]　[7：0]	[7：0]　[15：8]　[23：16] [31：24]	バイト・スワップ

保守／廃止

3. 3. 6　I/Oの性能

システム・コントローラを介してVR4120Aからアクセスするときの I/Oの性能を以下の表に示します。

W/R	ターゲット領域	バースト長	アクセス・レイテンシ［VR4120Aクロック］
R	IBUSターゲット	1	24
R	IBUSターゲット	2	24-1
R	IBUSターゲット	4	24-1-1-1
R	内部レジスタ（UARTを除く）	1	7
R	内部レジスタ（UARTを除く）	2	無効
R	内部レジスタ（UARTを除く）	4	無効
R	内部UARTレジスタ	1	14（UARTソース・クロックに依存）
R	内部UARTレジスタ	2	無効
R	内部UARTレジスタ	4	無効
W	IBUSターゲット	1	23
W	IBUSターゲット	2	23
W	IBUSターゲット	4	23-2-1-2
W	内部ライト・コマンドFIFO	1	6-1（ウエイト）
W	内部ライト・コマンドFIFO	2	6-1
W	内部ライト・コマンドFIFO	4	6-1-1-1
W	内部レジスタ（UARTを除く）	1	7
W	内部レジスタ（UARTを除く）	2	無効
W	内部レジスタ（UARTを除く）	4	無効
W	内部UARTレジスタ	1	15（UARTソース・クロックに依存）
W	内部UARTレジスタ	2	無効
W	内部UARTレジスタ	4	無効

備考1. バス周波数：SysAD = 100 MHz, IBUS = 66 MHz

2. IBUSターゲットへアクセスするレイテンシの値は，IBUSバス・アービトレーション・サイクル（約6CPU クロック）を含みません。

保守／廃止

3. 4　メモリ・インタフェース

VR4120Aは，メモリ・スペースをアドレスする通常の方法でコントローラに接続されたメモリにアクセスします。

3. 4. 1　概　　要

- 66 MHzまたは100 MHzメモリ・バス
- 最大32 Mバイトのベース・メモリ・レンジがSDRAMをサポート
- 最大8 Mバイトのライト・プロテクタブル・ブート・メモリ・レンジがPROM/フラッシュ・メモリをサポート
- プログラマブルSDRAMリフレッシュ・コントローラ内蔵
- 4ワード（16バイト）プリフェッチ・データ・バッファ（メモリからVR4120A）
- SDRAMのデータ・バスとPROM/フラッシュ・メモリのデータ・バスは兼用
- プログラマブル・メモリ・バス・アービトレーション
- メモリに対するプログラマブル・アドレス・レンジ
- プログラマブルRAS-CAS遅延（2, 3, 4クロック）
- プログラマブルCASレイテンシ（2, 3クロック）
- IBUSスレーブ・インタフェース上のエンディアン・コンバータ
- SysADバス上で8ワードのバースト・リード／ライトをサポートしない

3. 4. 2　メモリ領域

コントローラは，メモリに直接接続して，以下のアドレス・レンジに対してアドレス，データ，制御信号を管理します。

- 1つのブートPROM/フラッシュ・メモリ・レンジ（プログラマブル）
- 1つのシステム・メモリ・レンジ（プログラマブル）

例として（これらに限られたものではありませんが），以下の種類のメモリ・モジュールが使用できます。

- 各データ・サイズ（8, 16, 32ビット）のフラッシュ・メモリをブートROMとして使用可能
- 各データ・サイズ（8, 16, 32ビット）のPROMをブートROMとして使用可能
- SDRAMをシステム・メモリとして使用可能

ブートROMとして，PROMまたは85 nsのフラッシュ・メモリを使用できます。PROM/フラッシュ・メモリにアクセスする前に，ソフトウエアはこのアドレス・レンジのコンフィギュレーションをする必要があります。SDRAMにアクセスする前に，ソフトウエアはこのアドレス・レンジのコンフィギュレーションをする必要があります。

保守／廃止

3.4.3　メモリ信号接続

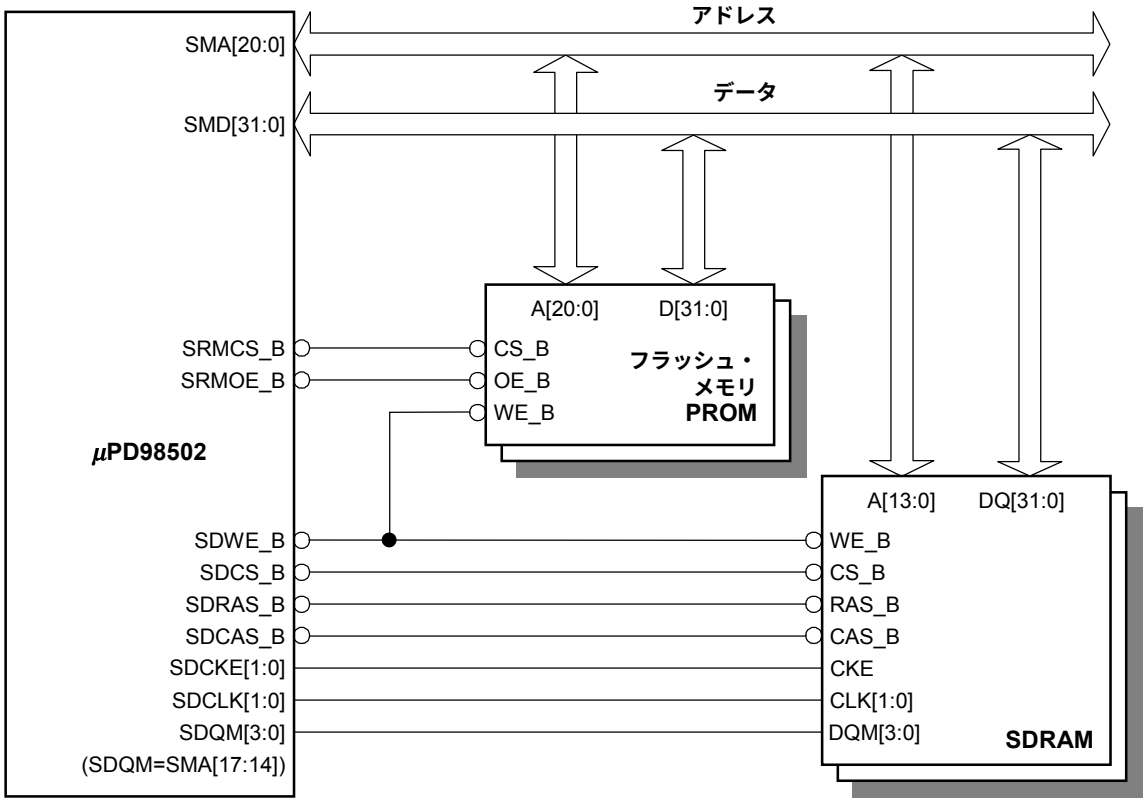


表3－5　外部端子マッピング

外部端子		ROMへのアクセス	SDRAMへのアクセス
名　　称	ビット		
SMA	[13：0]	A［13：0]	A［13：0]
	[17：14]	A［17：14]	SDQM［3：0]
	[20：18]	A［20：18]	---
SMD	[31：0]	D［31：0]	DQ［31：0]
SDCS_B		---	SDCS_B
SDRAS_B		---	SDRAS_B
SDCAS_B		---	SDCAS_B
SDWE_B		SDWE_B	SDWE_B
SDCKE	[1：0]	---	SDCKE［1：0]
SDCLK	[1：0]	---	SDCLK［1：0]
SRMCS_B		SRMCS_B	---
SRMOE_B		SRMOE_B	---
RMSL	[1：0]	---	---

備考　RMSL信号は，ブート・メモリ・データ・バス・サイズを決定します。

保守／廃止

3.4.4　メモリ性能

メモリ・アクセス・レイテンシは，メモリの種類，スピード，プリフェッチ・スキームで決定されます。アクセス・レイテンシのいくつかの例を以下の表に示します。4ワード（16バイト）CPU命令キャッシュ・ライン・フィルの各転送には，66 MHzまたは100 MHzのメモリ・バス・クロックが必要です。“ SysADクロック ”欄の最初の数字は，最初のワードに対するものです。残りの数字は後続のワードに対するものです。最も一般的な組み合わせを示します。

表3－6　メモリ性能の例（CPUからの4ワード・バースト・アクセス）

メモリの種類	インタリーブされたバンク	ページ・ヒット	R/W	プリフェッチ・ヒット	CPUから見たアクセス・レイテンシ［SysADクロック］
SDRAM, 10 ns	なし	あり	R	あり	6-1-1-1
SDRAM, 10 ns	なし	あり	R	なし	14-1-1-1
SDRAM, 10 ns	なし	あり	W	N/A	9-1-1-1
フラッシュ・メモリ，85 ns（32ビット・バス）	なし	なし	R	N/A	19-12-1-12
フラッシュ・メモリ，85 ns（16ビット・バス）	なし	なし	R	N/A	31-24-24-24
フラッシュ・メモリ，85 ns（8ビット・バス）	なし	なし	R	N/A	55-48-48-48
フラッシュ・メモリ，85 ns	なし	なし	W	N/A	18（シングル・アクセスのみ）
PROM	なし	なし	R	N/A	19-12-12-12

備考1. SDRAMコンフィギュレーションはRCD = 3, CL = 2, SDCLK = 100 MHz, FAT = 10

2. バス周波数はSysAD = 100 MHz, IBUS = 66 MHz

3. リード・パフォーマンスは，リード・コマンドがCPUによって発行されるCPUクロックに対する立ち上がりエッジをカウントすることで計算されます。ライト・コマンドが発行されるとCPUはウエイト・ステートのないライト・データを発行しますので，表中の数字はデータがメモリに書き込まれるスピードを表します。数字の合計は，ライト動作が発行されたタイミングと次のCPUメモリ動作が開始できるタイミングの間のサイクル数を表します。

4. フラッシュ・メモリ/ROMに対するバースト・ライト・アクセスは無効です。CPUはシングル・アクセスによってのみフラッシュ・メモリ/ROMにアクセスできます。

保守／廃止

表3－7　メモリ性能の例（IBUSマスタからの4ワード・バースト・アクセス）

メモリの種類	インタリーブされたバンク	ページ・ヒット	R/W	プリフェッチ・ヒット	IBUSから見たアクセス・レイテンシ［IBUSクロック］
SDRAM, 10 ns	なし	あり	R	N/A	18-1-1-1
SDRAM, 10 ns	なし	あり	W	N/A	12-1-1-1
フラッシュ・メモリ，85 ns（32ビット・バス）	なし	なし	R	N/A	45-1-1-1
フラッシュ・メモリ，85 ns（16ビット・バス）	なし	なし	R	N/A	77-1-1-1
フラッシュ・メモリ，85 ns（8ビット・バス）	なし	なし	R	N/A	141-1-1-1
フラッシュ・メモリ，85 ns	なし	なし	W	N/A	無効
PROM	なし	なし	R	N/A	45-1-1-1

備考1. SDRAMコンフィギュレーションはRCD = 3, CL = 2, SDCLK = 100 MHz, FAT = 10

2. バス周波数はSysAD = 100 MHz, IBUS = 66 MHz

3. 上記のアクセス・レイテンシは，IBUSアービトレーション・サイクル（4IBUSクロック）を含みません。

4. フラッシュ・メモリ/ROMに対するいかなるライト・アクセスも禁止されます。IBUSマスタがフラッシュ・メモリ/ROMにライト・アクセスをすると，IBUSバス・エラーが発生します。

3. 4. 5　RMMDR（ROMモード・レジスタ）

ROMモード・レジスタ“RMMDR”は，リード／ライト可で32ビット・ワード配列のレジスタです。RMMDRは，PROM/フラッシュ・メモリ・インタフェースを設定します。RMMDRには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：9	Reserved	R/W	0	0固定
8	WM	R/W	0	ライト・マスク： 0 = マスクされている。フラッシュ・メモリ・データは意図されない書き込みから保護されます。 1 = マスクされません。フラッシュ・メモリ・データは保護されません。
7：2	Reserved	R/W	0	0固定
1：0	FSM	R/W	00	フラッシュ・メモリ/PROMサイズ・モデル： 00 = モード1（4 Mバイト・モード） 01 = モード2（8 Mバイト・モード） 10 = モード3（1 Mバイト・モード） 11 = モード4（2 Mバイト・モード）

備考　初期化時にFSMフィールドに値を設定したあとは，値を変えないでください。

保守／廃止

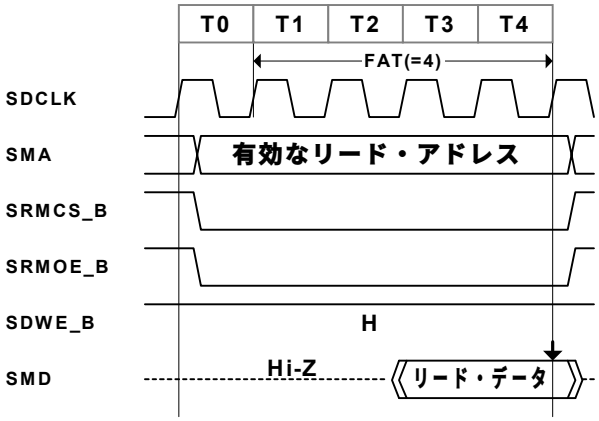
3. 4. 6　RMATR（ROMアクセス・タイミング・レジスタ）

ROMアクセス・タイミング・レジスタ“RMATR”は，リード／ライト可で32ビット・ワード配列のレジスタです。RMATRは，PROM/フラッシュ・メモリ・インタフェースのアクセス・タイムを設定します。RMATRには以下のフィールドがあり，リセット時に0に初期化されます。

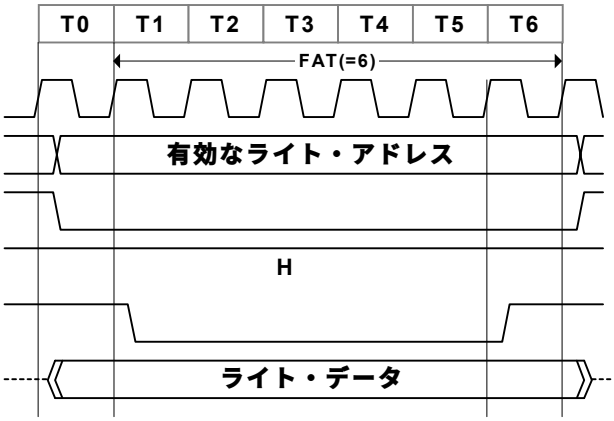
ビット	フィールド	R/W	ディフォールト	説　明		
31：3	Reserved	R/W	0	0固定		
2：0	FAT	R/W	000	通常のROMに対するフラッシュ・メモリ/PROMアクセス・タイミング		
				000 = 18クロック	66 MHz：272.4 ns	100 MHz：180 ns
				001 = 4クロック	66 MHz：60.6 ns	100 MHz：40 ns
				010 = 6クロック	66 MHz：90.9 ns	100 MHz：60 ns
				011 = 8クロック	66 MHz：121.2 ns	100 MHz：80 ns
				100 = 10クロック	66 MHz：151.5 ns	100 MHz：100 ns
				101 = 12クロック	66 MHz：181.8 ns	100 MHz：120 ns
				110 = 14クロック	66 MHz：212.1 ns	100 MHz：140 ns
				111 = 16クロック	66 MHz：242.4 ns	100 MHz：160 ns

備考　ROMアクセス・タイミングは，システム・クロック周波数に依存します。

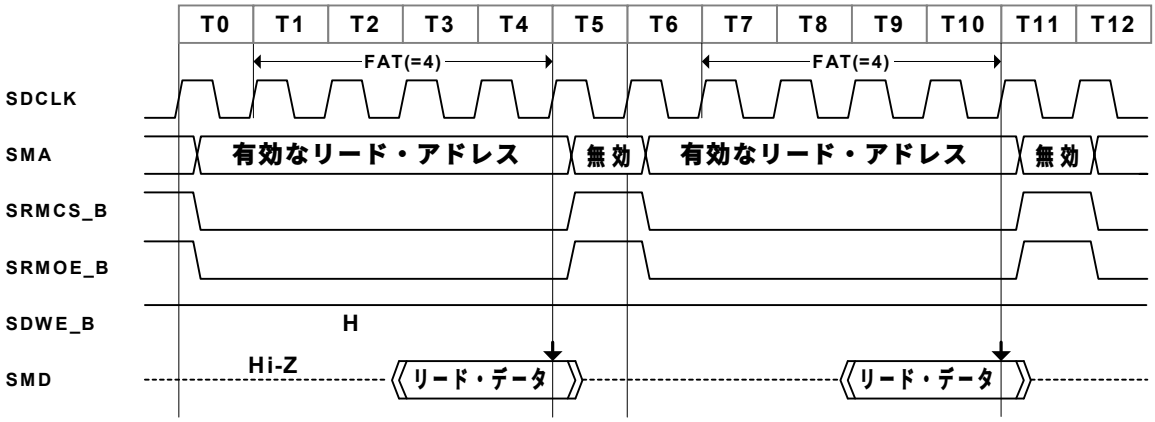
通常のROMリード・サイクル



フラッシュ・メモリ・ライト・サイクル



ROMバースト・リード・サイクル



保守／廃止

3. 4. 7　SDMDR（SDRAMモード・レジスタ）

SDRAMモード・レジスタ“ SDMDR ”は，リード／ライト可で32ビット・ワード配列のレジスタです。SDMDRは，SDRAMインタフェースの各モードを設定します。SDMDRには以下のフィールドがあり，リセット時に330Hに初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明		
31：10	Reserved	R/W	0	0固定		
9：8	RCD	R/W	11	SDRAM RAS-CAS遅延：		
				00 = 設定禁止		
				01 = 2クロック	66 MHz：30.3 ns	100 MHz：20 ns
				10 = 3クロック	66 MHz：45.5 ns	100 MHz：30 ns
				11 = 4クロック	66 MHz：60.6 ns	100 MHz：40 ns
				（ディフォールト）		
7	Reserved	R/W	0	0固定		
6：4	LTMD	R/W	011	SDRAM CASレイテンシ： 000 = 設定禁止 001 = 設定禁止 010 = 2 011 = 3（ディフォールト） 1xx = 設定禁止		
3	Reserved	R/W	0	0固定		
2：0	BL	R/W	000	SDRAMバースト長： 000 = 1（ディフォールト） 001 = 設定禁止 010 = 設定禁止 011 = 設定禁止 1xx = 設定禁止		

備考1. RAS-CAS遅延時間は，システム・クロック周波数に依存します。

2. SDRAMを使用したあとにこのレジスタの値を変えないでください。

3. このレジスタ設定による初期化は，SDRAMを使用する前に行う必要があります。

保守／廃止

3. 4. 8　SDTSR（SDRAMタイプ選択レジスタ）

SDRAMタイプ選択レジスタ“SDTSR”は，リード／ライト可で32ビット・ワード配列のレジスタです。SDTSRは，SDRAMのタイプを設定します。SDTSRには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明	
31：10	Reserved	R/W	0	0固定	
9：8	SDS	R/W	00	合計SDRAMサイズ：	
				00 = 4 Mバイト（ディフォールト）	003F_FFFFH – 0000_0000H
				01 = 8 Mバイト	007F_FFFFH – 0000_0000H
				10 = 16 Mバイト	00FF_FFFFH – 0000_0000H
				11 = 32 Mバイト	01FF_FFFFH – 0000_0000H
7	BTM	R/W	0	バンク数 0 = 1または2バンク（ディフォールト） 1 = 3または4バンク	
6：4	RAB	R/W	000	SDRAMアドレス・ビット（RAS＋CAS）の合計数（バンク選択端子を除く）： 000 = 17ビット（ディフォールト） 001 = 18ビット 010 = 19ビット 011 = 20ビット 100 = 21ビット 101 = 22ビット 110 = 設定禁止 111 = 設定禁止	
3：2	Reserved	R/W	0	0固定	
1：0	CAB	R/W	00	カラム・アドレス・ビット数（バンク選択端子を除く）： 00 = 7ビット（ディフォールト） 01 = 8ビット 10 = 9ビット 11 = 10ビット	

備考　このレジスタの各フィールドで「設定禁止」となっている値は設定しないでください。

保守／廃止

3. 4. 9　SDPTR（SDRAMプリチャージ・タイミング・レジスタ）

SDRAMプリチャージ・タイミング・レジスタ“SDPTR”は，リード／ライト可で32ビット・ワード配列のレジスタです。SDPTRは，SDRAMコントローラのプリチャージ・タイミングを設定します。SDPTRには以下のフィールドがあり，リセット時に142Hに初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明		
31：9	Reserved	R/W	0	0固定		
8	DPL	R/W	1	入力データ → プリチャージ・コマンド・タイミング（t_{DPL}）：		
				0 = 1クロック 1 = 2クロック （ディフォールト）	66 MHz：15.2 ns 66 MHz：30.3 ns	100 MHz：10 ns 100 MHz：20 ns
7	Reserved	R/W	0	0固定		
6：4	APT	R/W	100	アクティブ・コマンド → プリチャージ・コマンド・タイミング（t_{RAS}）：		
				000 = 4クロック 001 = 5クロック 010 = 6クロック 011 = 7クロック 100 = 8クロック （ディフォールト） 101 = 設定禁止 110 = 設定禁止 111 = 設定禁止	66 MHz：60.6 ns 66 MHz：75.7 ns 66 MHz：90.9 ns 66 MHz：106.0 ns 66 MHz：121.2 ns	100 MHz：40 ns 100 MHz：50 ns 100 MHz：60 ns 100 MHz：70 ns 100 MHz：80 ns
3：2	Reserved	R/W	0	0固定		
1：0	PAT	R/W	10	プリチャージ・コマンド → アクティブ・コマンド・タイミング（t_{RP}）：		
				00 = 2クロック 01 = 3クロック 10 = 4クロック （ディフォールト） 11 = 設定禁止	66 MHz：30.3 ns 66 MHz：45.5 ns 66 MHz：60.6 ns	100 MHz：20 ns 100 MHz：30 ns 100 MHz：40 ns

備考　このレジスタの各フィールドで「設定禁止」となっている値は設定しないでください。

保守／廃止

3. 4. 10　SDRMR（SDRAMリフレッシュ・モード・レジスタ）

SDRAMリフレッシュ・モード・レジスタ“SDRMR”は，リード／ライト可で32ビット・ワード配列のレジスタです。SDRMRは，SDRAMリフレッシュ・コントローラを初期化します。SDRMRには以下のフィールドがあり，リセット時に200Hに初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：16	Reserved	R/W	0	0固定
15：0	RCSET	R/W	0200H	SDRAMリフレッシュ・タイマ・カウンタのリロード値。 この値は，システム・クロックによってカウントするカウンタがゼロになるとリフレッシュ・タイマ・カウンタに自動的にリロードされます。リフレッシュ・タイマ・カウンタは，この値からカウント・ダウンします。カウント・サイクルの時間はこのフィールドの値プラス1に相当します。ディフォールト値（200H = 512）は，66 MHzで動作するシステム・クロックに対しては32 msごとに4096回のリフレッシュ・サイクル（たとえば，7.8125 μsごとに1回のリフレッシュ）を必要とするSDRAMチップのリフレッシュ・レートです。これは，非常に用心深い方法ですが確実にブートを行うことができます。あとでリロード値を大きくすることもできます。RCSET［7：0］は0に固定されていますので，100H未満のタイマ値はこのフィールドに設定できません。100H未満の値をこのフィールドに設定すると，ディフォールト値の“200H”が自動的にロードされます。

3. 4. 11　SDRCR（SDRAMリフレッシュ・タイマ・カウント・レジスタ）

SDRAMリフレッシュ・タイマ・カウント・レジスタ“SDRCR”は，リード・オンリーで32ビット・ワード配列のレジスタです。SDRCRは，カウントが終了するごとにSDRAMリフレッシュを行うための16ビット・タイマです。SDRAMリフレッシュ・コントローラは，このフリー・ランニング・タイマのカウント値をSDRMRレジスタのRCSETフィールドから自動的にリロードします。SDRCRには以下のフィールドがあり，リセット時に200Hに初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：16	Reserved	R	0	0固定
15：0	RCC	R	0200H	このフィールドは，SDRAMリフレッシュ・タイマの現在値を表します。

保守／廃止

3. 4. 12　MBCR（メモリ・バス制御レジスタ）

メモリ・バス制御レジスタ“MBCR”は，リード／ライト可で32ビット・ワード配列のレジスタです。MBCRは，VR4120AまたはIBUSがメモリにアクセスする優先順位を選択するために使用します。VR4120Aは，メモリへのリクエストについてIBUSよりも高い優先度を自分自身に割り振ることができます。また，メモリへのリクエストについてVR4120AとIBUSに同じ優先度を割り振ることもできます。MBCRには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：1	Reserved	R/W	0	0固定
0	BPR	R/W	0	メモリ・アクセスの優先度： 0：SysAD（CPU）＞リフレッシュ＞IBUS（ディフォールト） メモリ・アービタは，IBUSからの現在のメモリ・アクセスが完了していなくても，1つのCPUメモリ・トランザクションまたはリフレッシュ動作を許可します。CPUは，IBUSからのバースト・アクセスの現在のワード・カウントが16ワードまたは32ワードに達したときに，メモリ・アクセスを行うことが許可されます。リフレッシュ動作も割り込みIBUSアクセスに対して許可されます。 1：SysAD（CPU）= リフレッシュ = IBUS 3つの動作は発生した順序で実行されますが，CPUは，3つのリクエストが同時に発生しても，メモリ・アクセスが許可されます。

3. 4. 13　ブートROM

システム・コントローラは，最大8 Mバイトのブート・メモリをサポートします。

3. 4. 13. 1　ブートROM構成とアドレス・レンジ

ブートROMとして，PROMまたは85 nsのフラッシュ・メモリを使用することができます。ブートROMのアクセス・タイムは200 ns以下でなければなりません。システム・コントローラは，VR4120Aの物理メモリ空間の1F80_0000Hから1FFF_FFFFHまでの8, 16, 32ビット幅のブートROMをサポートします。ブートROMは，VR4120Aのキャッシュ動作をサポートしません。

表3－8　リセット時のブートROMサイズ構成

RMMDR.FSM	ブートROMサイズ	アドレス・レンジ
00	4 Mバイト	1FC0_0000H-1FFF_FFFFH
01	8 Mバイト	1F80_0000H-1FFF_FFFFH
10	1 Mバイト	1FC0_0000H-1FCF_FFFFH
11	2 Mバイト	1FC0_0000H-1FDF_FFFFH

保守／廃止

コントローラは，1F80_0000Hから1FFF_FFFFHまでのアドレス・レンジ内で，フラッシュ・メモリ/ROMチップ・セレクト（SRMCS_B）をアサートします。ROM/フラッシュ・メモリ・スペースに書き込みを行うときは，コントローラはSRMCS_BとともにSDWE_Bをアサートします。読み出しを行うときは，コントローラはSRMCS_BとともにSRMOE_Bをアサートします。VR4120Aがフラッシュ・メモリ/ROMの定義されたサイズ外のブートROMアドレスをアクセスしようとすると，コントローラはSysCMD［0］上にデータ・エラー・ビットをセットして0を返します。さらに，割り込みがNMIイネーブル・レジスタ“S_NER”でイネーブルにセットされていれば，NMIステータス・レジスタ“S_NSR”を更新し，VR4120Aに対してNMIをアサートします。

表3－9　メモリ・マップとアドレス・バスの関係

メモリ・アドレス［31：0］	出力値SMA［20：0］		
	8ビット	16ビット	32ビット
ブートROM領域（SRMCS_B端子をチップ選択に使用）－8 Mビット			
1FFF_FFFFH	使用不可	1F_FFFFH	1F_FFFFH
1FF0_0000H		18_0000H	1C_0000H
1FEF_FFFFH		17_FFFFH	1B_FFFFH
1FE0_0000H		10_0000H	18_0000H
1FDF_FFFFH	1F_FFFFH	0F_FFFFH	17_FFFFH
1FD0_0000H	10_0000H	08_0000H	14_0000H
1FCF_FFFFH	0F_FFFFH	07_FFFFH	13_FFFFH
1FC0_0000H	00_0000H	00_0000H	10_0000H
1FBF_FFFFH	使用不可	使用不可	0F_FFFFH
1FB0_0000H			0C_0000H
1FAF_FFFFH			0B_FFFFH
1FA0_0000H			08_0000H
1F9F_FFFFH	使用不可		07_FFFFH
1F90_0000H			04_0000H
1F8F_FFFFH			03_FFFFH
1F80_0000H			00_0000H
ROMSEL［1：0］	10B	01B	00B

保守／廃止

3. 4. 13. 2　フラッシュ・メモリ書き込み保護

フラッシュ・メモリはソフトウエアで保護できます。ROMモード・レジスタ“RMMDR”のWMフィールドをプログラムすることによってソフトウエアによる書き込み保護（ライト・プロテクト）ができます。

3. 4. 13. 3　フラッシュ・メモリ動作

フラッシュ・メモリ・インタフェースは，それぞれのフラッシュ・メモリ・データ・バス・サイズに対して3つのモード（8, 16, 32ビット）があります。各バス・サイズによってライト・サイクルを行わせる方法は変わります。

表3－10　使用可能なライト・アクセス・タイプ

	フラッシュ・メモリ・データ・バス・サイズ		
	8ビット	16ビット	32ビット
使用可能 ライト・コマンドを引き起こすアクセス・タイプ	バイト・ライト （asm言語の“sb”）	ハーフ・ワード・ライト （asm言語の“sh”）	ワード・ライト （asm言語の“sw”）

フラッシュ・メモリは，VR4120Aによる以下のライト・サイクル・シーケンスを用いてプログラムできます。以下のコマンドは，システム・コントローラにおける32ビット・フラッシュ・メモリ・データ・バス・モードでのAMD AM29LV800BTフラッシュ・メモリ（バイト・モードを使用）に関する動作例です。

保守／廃止

表3－11　コマンド・シーケンス

(a) プログラム・コマンド・シーケンス（4つのライト・サイクル）

1番目のライト		2番目のライト		3番目のライト		4番目のライト		5番目のライト		6番目のライト	
A	1FC0_2AA8H	A	1FC0_1554H	A	1FC0_2AA8H	A	PA*	A		A	
D	AAAA_AAAAH	D	5555_5555H	D	A0A0_A0A0H	D	PD*	D		D	

(b) チップ・イレース・コマンド・シーケンス（6つのライト・サイクル）

1番目のライト		2番目のライト		3番目のライト		4番目のライト		5番目のライト		6番目のライト	
A	1FC0_2AA8H	A	1FC0_1554H	A	1FC0_2AA8H	A	1FC0_2AA8H	A	1FC0_1554H	A	1FC0_2AA8H
D	AAAA_AAAAH	D	5555_5555H	D	8080_8080H	D	AAAA_AAAAH	D	5555_5555H	D	1010_1010H

(c) セクタ・イレース・コマンド・シーケンス（6つのライト・サイクル）

1番目のライト		2番目のライト		3番目のライト		4番目のライト		5番目のライト		6番目のライト	
A	1FC0_2AA8H	A	1FC0_1554H	A	1FC0_2AA8H	A	1FC0_2AA8H	A	1FC0_1554H	A	EA*
D	AAAA_AAAAH	D	5555_5555H	D	8080_8080H	D	AAAA_AAAAH	D	5555_5555H	D	3030_3030H

備考　A = メモリ・ライト・アドレス

D = メモリ・ライト・データ

PA = プログラムするフラッシュ・メモリ・ロケーションのアドレス

PD = ロケーションPAにプログラムするデータ

EA = 消去するフラッシュ・メモリ・ロケーションのブロック・アドレス

フラッシュ・メモリ・プログラミングの場合，以下のシステム要因を考慮に入れてください。

(1) リード・サイクルは，これらのライト・コマンドに割り込むことはできません。したがって，μPD98502はフラッシュ・メモリからのフェッチでフラッシュ・メモリをプログラムできません。

(2) フラッシュ・メモリに対するこれらのライト・コマンドは，以下のシステム要因によって変化します。

- 製造会社：ライト・シーケンスは各会社で異なります。
- エンディアン・モード（3つのシステム・エンディアン・モードがあります。すなわち，リトル・エンディアン，データ・スワップ・モード付きビッグ・エンディアン，アドレス・スワップ・モード付きビッグ・エンディアン）
- フラッシュ・メモリのフラッシュ・メモリ・データ・バス・サイズ（通常のフラッシュ・メモリは，8ビットと16ビットのバス・モードがあります）
- システム・コントローラのフラッシュ・メモリ・データ・バス・サイズ（8, 16, 32ビット）

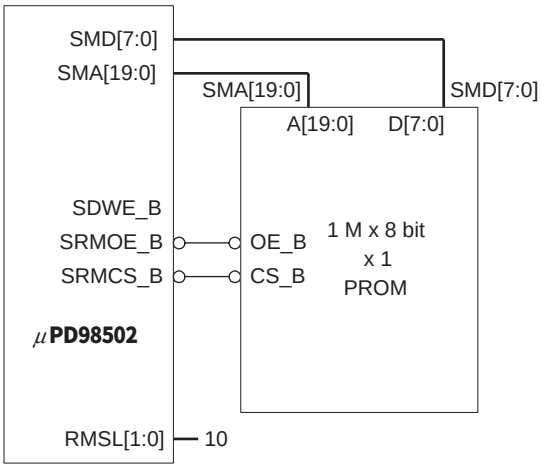
(3) ライト・シーケンスでのSMDとSMA信号出力を同じにしてください。

保守／廃止

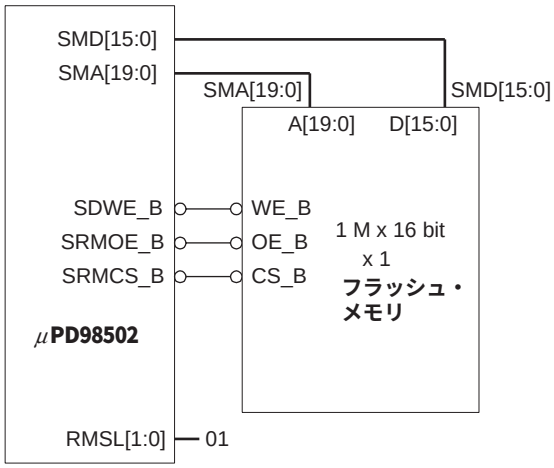
3. 4. 13. 4　ブートROM信号接続

フラッシュ・メモリ/ROM構成

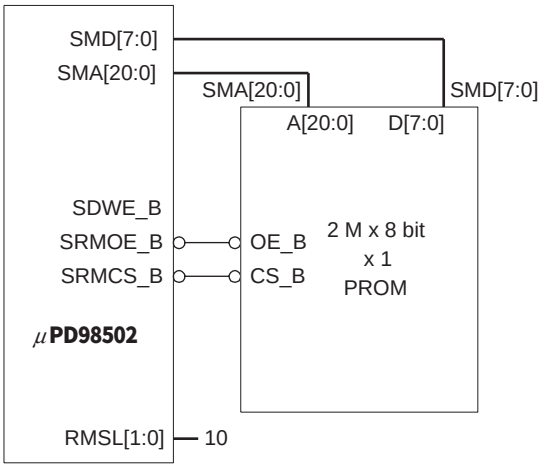
例（1 Mバイト PROM）



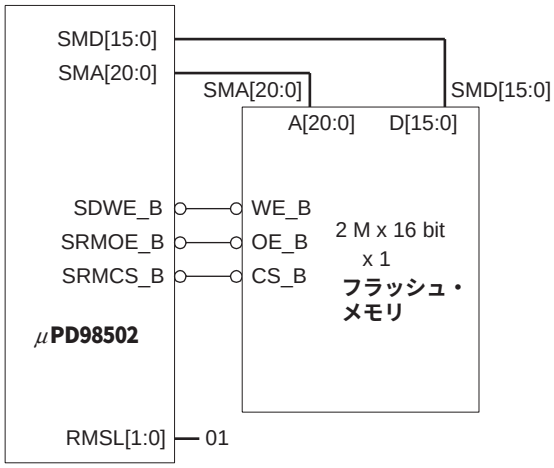
例（2 Mバイト・フラッシュ・メモリ）



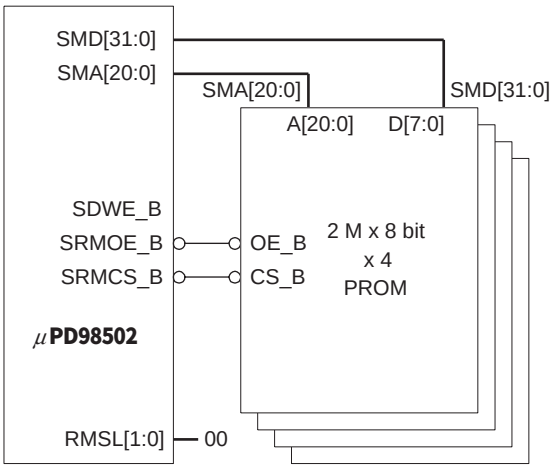
例（2 Mバイト PROM）



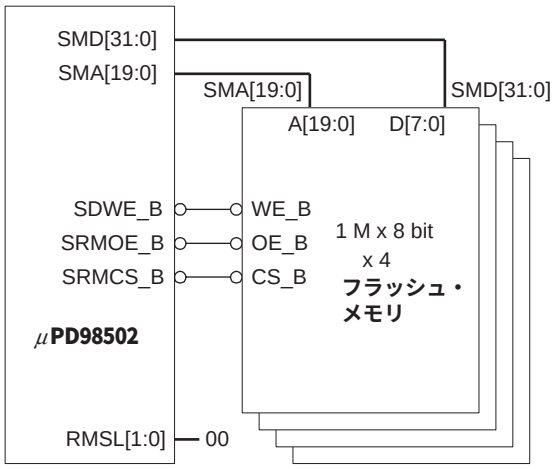
例（4 Mバイト・フラッシュ・メモリ）



例（8 Mバイト PROM）



例（4 Mバイト・フラッシュ・メモリ）



保守／廃止

3. 4. 14　SDRAM

3. 4. 14. 1　SDRAMアドレス・レンジ

システム・メモリの1つとして，SDRAMチップを使用することができます。SDRAMのアクセス・タイムは10 ns以下でなければなりません。システム・コントローラは，VR4120Aの物理メモリ空間の0000_0000Hから01FF_FFFFHにおいて，16 Mビット，64 Mビット，128 MビットのSDRAMをサポートします。SDRAMは，VR4120Aのキャッシュ動作をサポートします。

表3－12　リセット時のSDRAMサイズ構成

SDMDR.SDS	SDRAMサイズ	アドレス・レンジ
00	4 Mバイト	0000_0000H-003F_FFFH
01	8 Mバイト	0000_0000H-007F_FFFH
10	16 Mバイト	0000_0000H-00FF_FFFH
11	32 Mバイト	0000_0000H-01FF_FFFH

3. 4. 14. 2　SDRAMデバイス構成

コントローラは，以下に示す16 Mビット，64 Mビット，128 MビットのSDRAM構成をサポートします。システム・メモリとしてサポートするSDRAM構成を以下の表に示します。

表3－13　サポートするSDRAM構成

メモリ・サイズ	必要なSMAアドレス・ビット	構成 （バンク×ワード×ビット）	数
4 Mバイト	12：0	2×0.5 M×16	2
8 Mバイト	12：0	4×0.5 M×32	1
16 Mバイト	13：0	4×1.0 M×16	2
32 Mバイト	13：0	4×2.0 M×16	2

3. 4. 14. 3　SDRAMのバースト・タイプとバンク

インタリーブとバンクは，SDRAMチップを使ったメモリ設計の情況によって複数の意味を持ちます。これらの意味は以下のとおりです。

- バンク（メモリ・モジュールとSDRAMチップで適用が異なる）：メモリ・モジュールに関して参照されるバンクは，SDRAMチップ内のバンクとは異なります。コントローラはモジュールのバンクを識別する機能はサポートしていません。

- バースト・タイプ（SDRAMチップに適用）：1つのSDRAMチップのバースト・タイプは，インタリーブあるいはシーケンシャルでチップのモード・レジスタにプログラムされます。このバースト・タイプは，データがSDRAMチップに読み書きされるワード・オーダにのみ関連しています。このバースト・タイプは，ある一定のクロック・サイクルで転送されるワード数との関連はありません。μPD98502に接続されるすべてのSDRAMチップのバースト・タイプは，メモリ初期化手順の間にコンフィギュレーションされます。システム・コントローラ内のメモリ・コントローラは，インタリーブ・バースト・モードはサポートせず，シーケンシャル・バースト・モードのみをサポートします。

保守／廃止

3. 4. 14. 4　SDRAMワード・オーダリング

4ワード命令キャッシュ・ライン・フィルに対するSDRAMからのワード・アドレス・オーダを以下の表に示します。このオーダは，SDRAMチップのバースト・タイプで決定されます。これは，メモリの初期化手順の間にプログラムされます。メモリ・コントローラは，すべての接続されたSDRAMチップに対して同じバースト・タイプとワード・オーダをシステム・メモリ・レンジでプログラムします。この表の“シーケンシャル”とは，SDRAMのバースト・タイプを示します。バースト長は，CPUが行うアクセス・タイプにのみ依存します。

表3－14　命令キャッシュ・ライン・フィルに対するSDRAMワード・オーダ

スタート・カラム・アドレスA1, A0	SDRAMチップ・バースト・タイプ	
	シーケンシャル	インタリーブド
00	0-1-2-3	サポートしない
01	1-2-3-0	サポートしない
10	2-3-0-1	サポートしない
11	3-0-1-2	サポートしない

備考　メモリ・コントローラは，インタリーブド・バースト・タイプのSDRAMをサポートしません。すべてのSDRAMは，4ワードのバースト長を用いて，シーケンシャル・バースト・タイプに初期化されていると仮定します。

3. 4. 14. 5　SDRAM信号接続

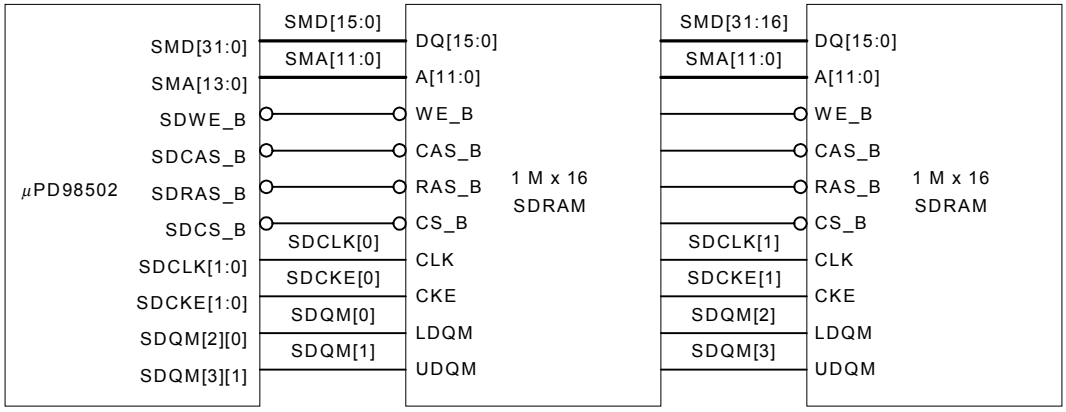
SDRAM信号の接続例を以下の図に示します。SMA［11］は，バンク・セレクト信号です。コマンド・サイクルでは，SMA［11］ロウはバンクAを，SMA［11］ハイはバンクBを選択します。両方のバンクとも同じSDCS_B, SDRAS_B, SDCAS_B, SDWE_B信号を共有します。

システム・メモリの2つのバンクは，マスクされない最上位のアドレス・ビットがバンクの選択を制御して，アドレス・レンジの2つの半分として動作します。

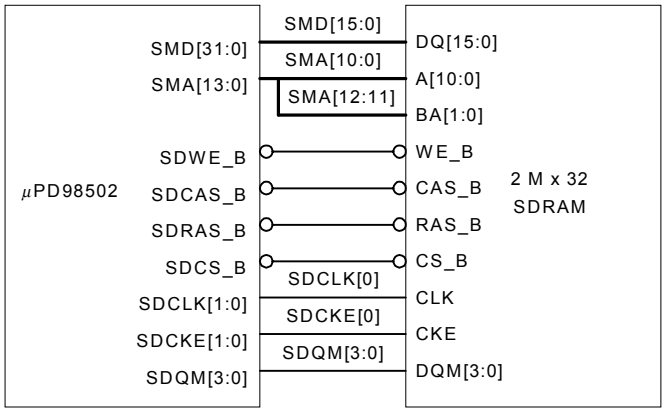
保守／廃止

SDRAM構成

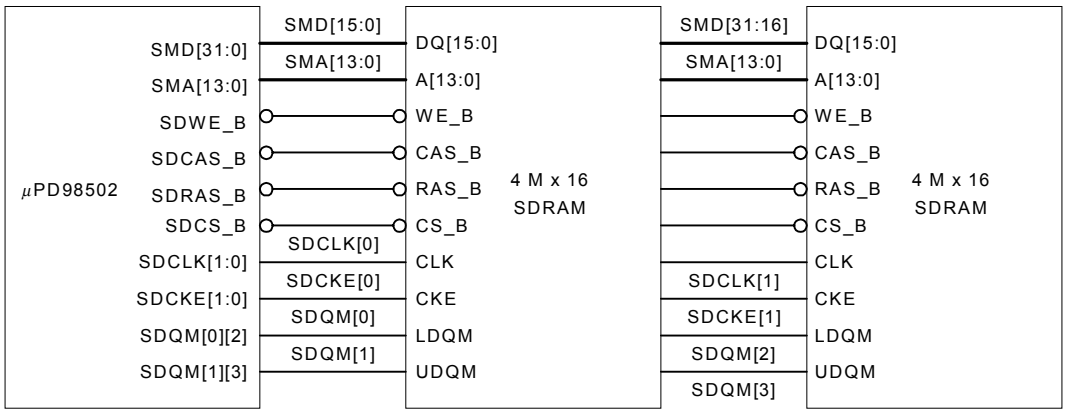
4 Mバイト



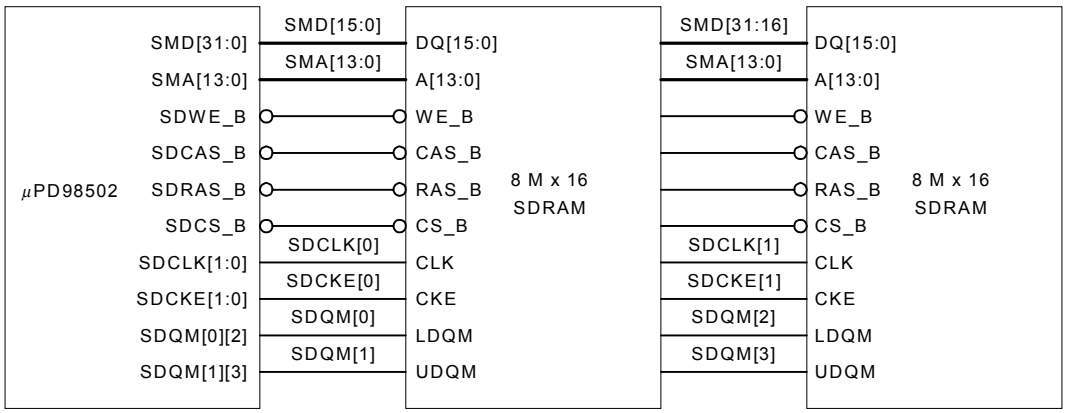
8 Mバイト



16 Mバイト



32 Mバイト



保守／廃止

3. 4. 15　SDRAMリフレッシュ

システム・コントローラは，CAS-Before-RAS（CBR）DRAMリフレッシュをすべてのSDRAMアドレス・レンジに対してサポートします。リフレッシュ・クロックは，システム・クロックから作り出します。そのレートは，SDRAMリフレッシュ・モード・レジスタ“SDRMR”のRCSETフィールドをプログラムすることによって決定します。

リフレッシュ・ロジックは，カウンタが0になるたびにSDRAMへのアクセスを要求します。リフレッシュ・ロジックは，バスを待つあいだに最大15個のリフレッシュ・リクエストを蓄積できます。リフレッシュ・ロジックがバスを持つと，すべての蓄積されたリフレッシュはシステム・メモリに対して実行され，ほかのアクセス（CPUまたはIBUS）は許可されません。リフレッシュ動作は，1つのクロックで交互に配置されます。すなわち，SDRAS_B信号の任意のペアの遷移間に少なくとも1つのバス・クロックが存在します。リフレッシュは，システム・メモリ・プリフェッチFIFOを自動的にクリアします。

3. 4. 16　メモリからCPUへのプリフェッチFIFO

各々のバースト4ワード・リードのあと，メモリ・コントローラは4つの追加ワードをその内部プリフェッチFIFOにプリフェッチします。プロセッサが，引き続き最後のリード・サイクル・アドレスの直後（シーケンシャルで）のアドレスからデータの読み出しをすると，最初の4ワードがプリフェッチFIFOから送られてきます。

メモリ・コントローラは，現在のSysADアドレスを前のアドレスと比較してアクセスがシーケンシャルであることを判断します。プリフェッチされたワードは，システム・メモリ以外のリソースに対するアクセスがシステム・メモリへのアクセス間で行われていても，プリフェッチFIFOに保持されています。

3. 4. 17　CPUからメモリへの書き込みFIFO

メモリ・コントローラには，4ワードのCPUからメモリへの書き込みFIFOがあります。このFIFOは，CPUの最高速度で書き込みを受け付けます。CPUからの1つの書き込みに対し，1つのアドレスが保持され，バッファリングを行います。そのトランザクションは，1ワード，ダブル・ワード，あるいは4ワードのデータ・キャッシュ・ライトバックです。1つのワードがCPUによってFIFOに置かれると，メモリ・コントローラはFIFOの内容をメモリにできるだけ早く書き込もうとします。次のCPUのリードまたはライトがメモリに通知されると，コントローラはレディ信号を無効にし，次のCPUトランザクション（リードまたはライト）をコントローラがそのFIFOを空にするまでストールさせます。次のCPUトランザクション（リードまたはライト）がIBUSターゲットに通知されると，メモリ・コントローラはレディ信号をアサートし，CPUトランザクションを完了させます。

保守／廃止

3. 4. 18　SDRAMメモリ初期化

以下のセクションは，この初期化で用いられるコンフィギュレーション・シーケンスについて記述します。

3. 4. 18. 1　メモリ・コントローラによるパワーオン初期化シーケンス

メモリのコンフィギュレーションを行う以下のシーケンスがリセット後に自動的に行われます。

1. 電源投入後に100 μs待ちます。
2. すべてのバンク・プリチャージを行います。
3. シーケンシャル・オート・リフレッシュ（CBR）を8回行います。

3. 4. 18. 2　ソフトウエアを使ったメモリ初期化シーケンス

SDRAMは，パワーオン初期化後に以下のシーケンスでソフトウエアによる初期化を行う必要があります。

1. SDRAMタイプ選択レジスタ“SDTSR”をプログラムします。
2. SDRAMモード・レジスタ“SDMDR”をプログラムします。
3. 20 μs待ちます。
4. SDRAMリフレッシュ・モード・レジスタ“SDRMR”をプログラムします。

この時点でメモリの使用準備が整います。コントローラ内のすべてのほかのコンフィギュレーション・レジスタは，通常動作を始める前にプログラムされなければなりません。

備考　ソフトウエアは，SDRAM初期化シーケンスのあとに，SDTSRとSDMDRを変更してはいけません。

保守／廃止

3. 5　IBUSインタフェース

3. 5. 1　概　　要

- IBUSマスタとターゲットとして動作
- 64ワード（256バイト）IBUSスレーブTxFIFO（IBUSからのIBUSリード・データ）
- 64ワード（256バイト）IBUSスレーブRxFIFO（IBUSへのIBUSライト・データ）
- 4ワード（16バイト）IBUSマスタTxFIFO（IBUSからのVR4120Aリード・データ）
- 4ワード（16バイト）IBUSマスタRxFIFO（IBUSへのVR4120Aライト・データ）
- IBUSのストールを検知するためバス・タイマをサポート
- 66 MHzのIBUSクロック・レート
- IBUS上で266 Mバイト／秒（32ビット@66 MHz）のバーストをサポート
- SysADバスとIBUSマスタ・インタフェース間のエンディアン変換をサポート
- メモリ・バスとIBUSスレーブ・インタフェース間のエンディアン変換をサポート

3. 5. 2　ITCNTR（IBUSタイムアウト・タイマ制御レジスタ）

IBUSタイムアウト・タイマ制御レジスタ“ITCNTR”は，リード／ライト可で32ビット・ワード配列のレジスタです。ITCNTRは，IBUSタイムアウト・タイマの使用をイネーブルにするために使用します。ITCNTRには以下のフィールドがあり，リセット時に0Hに初期化されます。

ビット	フィールド	R/W	ディフォールト	説　　明
31：1	Reserved	R/W	0	0固定
0	ITWEN	R/W	0	IBUSタイムアウト・タイマ・イネーブル： 1 = イネーブル 0 = ディスエーブル

保守／廃止

3. 5. 3　ITSETR（IBUSタイムアウト・タイマ設定レジスタ）

このレジスタは，ウォッチドッグ・タイマ機能に対するサイクルを設定します。ウォッチドッグ・タイマ・サイクルは，1～2^{32}－1クロックの範囲で1クロック単位で設定できます。DSUCLRRのDSWCLRビットは，指定されたサイクル時間内にソフトウエアによって設定されなければなりません。ITSETRは，32ビット・ワード配列のレジスタです。ディフォールトは8000_0000Hです。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	ITTIME	R/W	8000_ 0000H	IBUSタイムアウト値の設定： タイムアウト = ITTIME値のシステム・クロック時間 （100 MHz：10 ns, 66 MHz：15 ns） 例： ITTIME = 05F5_E100H（100 MHz）または03F9_40AAH（66 MHz） →タイムアウト = 1秒 ITTIME = 0BEB_C200H（100 MHz）または07F2_8154H （66 MHz） →タイムアウト = 2秒 ITTIME = 11E1_A300H（100 MHz）または0BEB_C200H （66 MHz） →タイムアウト = 3秒 ：　：　： ：　：　： ：　：　：

保守／廃止

3. 6　ウォッチドッグ・タイマ

3. 6. 1　概　　要

ウォッチドッグ・タイマは，VR4120Aが暴走（エンドレス・ループ）状態にあることを検知しVR4120Aをリセットします。

3. 6. 2　DSUCNTR（ウォッチドッグ・タイマ制御レジスタ）

このレジスタは，ウォッチドッグ・タイマ機能の使用をイネーブルにするために使用します。DSUCNTRは，32ビット・ワード配列のレジスタです。ディフォールトは0Hです。

ビット	フィールド	R/W	ディフォールト	説　　明
31：1	Reserved	R/W	0	0固定
0	DSWEN	R/W	0	ウォッチドッグ・タイマ機能イネーブル： 1 = イネーブル 0 = ディスエーブル

3. 6. 3　DSUSETR（ウォッチドッグ・タイマ・タイム設定レジスタ）

このレジスタは，ウォッチドッグ・タイマ・サイクルを設定します。ウォッチドッグ・タイマ・サイクルは，1～2^{32}－1クロックの範囲で1クロック単位で設定できます。DSUCLRRのDSWCLRビットは，指定されたサイクル時間内にソフトウエアによって設定されなければなりません。DSUSETRは，32ビット・ワード配列のレジスタです。ディフォールトは8000_0000Hです。

ビット	フィールド	R/W	ディフォールト	説　　明
31：0	DEDTIM	R/W	8000_ 0000H	ウォッチドッグ・タイマ・サイクル設定： ウォッチドッグ・タイマ・サイクル = DEDTIME値のシステム・クロック時間 （100 MHz：10 ns, 66 MHz：15 ns） 例： DEDTIM = 05F5_E100H（100 MHz）または03F9_40AAH（66 MHz） →ウォッチドッグ・タイマ・サイクル = 1秒 DEDTIM = 0BEB_C200H（100 MHz）または07F2_8154H（66 MHz） →ウォッチドッグ・タイマ・サイクル = 2秒 DEDTIM = 11E1_A300H（100 MHz）または0BEB_C200H（66 MHz） →ウォッチドッグ・タイマ・サイクル = 3秒 ：　　：　　：

保守／廃止

3. 6. 4　DSUCLRR（ウォッチドッグ・タイマ・クリア・レジスタ）

このレジスタのDSWCLRビットを“1”にセットすると，ウォッチドッグ・タイマ・カウンタがクリアされます。VR4120Aは，DSUSETRに指定された時間内にビットに“1”が書き込まれないと，自動的にリセットされます。DSUCLRは，32ビット・ワード配列のレジスタです。ディフォールトは0Hです。

ビット	フィールド	R/W	ディフォールト	説　明
31：1	Reserved	W	0	0固定
0	DSWCLR	W	0	ウォッチドッグ・タイマ・カウンタ・クリア。1が書き込まれると0にクリアされます。 1 = クリア 0 = クリアしない

3. 6. 5　DSUTIMR（ウォッチドッグ・タイマ経過時間レジスタ）

このレジスタは，現在のウォッチドッグ・タイマに対する経過時間を示します。DSUTIMRは，リード・オンリーで32ビット・ワード配列のレジスタです。ディフォールトは0Hです。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	CRTTIM	R	0	現在のウォッチドッグ・タイマ値（経過時間） 例： CRTTIM = 05F5_E100H（100 MHz）または 03F9_40AAH（66 MHz）→1秒 CRTTIM = 0BEB_C200H（100 MHz）または 07F2_8154H（66 MHz）→2秒 CRTTIM = 11E1_A300H（100 MHz）または 0BEB_C200H（66 MHz）→3秒 ：　：　：　：　：　：

3. 6. 6　ウォッチドッグ・タイマ・レジスタ設定フロー

ウォッチドッグ・タイマ・レジスタの設定フローを以下に記述します。

1. ウォッチドッグ・タイマのカウント・アップ値を設定します（1から$2^{32}-1$まで）。
   CPUは，この時間内にタイマをクリアしない（DSUCLRRに1が書き込まれない）と，リセットされます。
2. ウォッチドッグ・タイマをイネーブルにします。
3. 上記のステップ1で指定された時間内にタイマをクリアします。通常の使用状態では，ステップ3を繰り返します。現在の経過時間を得るには，DSUTIMRを読み込んでください。
4. シャットダウンのためにウォッチドッグ・タイマをディスエーブルにします。

保守／廃止

3. 7　エンディアン・モードのソフトウエアの問題

3. 7. 1　概　　要

モトローラやIBM 370プロセッサのようなMIPSプロセッサに元々組み込まれているエンディアン・モードは，ビッグ・エンディアンです。しかしながら，（PCI標準を開発した）インテルやVAXプロセッサに元々組み込まれたモードは，リトル・エンディアンです。PCIとの互換性の理由上，ほとんどのPCI周辺チップは元々リトル・エンディアン・モードで動作します。μPD98502は，元々リトル・エンディアンですが，SysADバス上でビッグ・エンディアンとリトル・エンディアン・モードのいずれかをサポートします。リセット時のENDCEN信号の状態が，このエンディアン・モードを決定します。しかし，エンディアンが混在している設計でコントローラを使用する場合，重要な問題点があります。エンディアン問題の最も重要な面は，SysADバスのどのバイト・レーンが特定のアドレスに対してアクティブになるかということです。ビッグ・エンディアン・モードがCPUインタフェースに対して実行されると，コントローラはSysADバス上で出入りするワードやハーフ・ワード内でバイトをスワップします。システム・コントローラ以外のほかのすべてのインタフェースは，リトル・エンディアン・モードで動作します。

以下のセクションは，プログラマの視点からエンディアンの問題を記述しています。エンディアンが混在している設計をどのように実行するか，コードをエンディアンに依存しないものにするにはどうすればよいかについて記述しています。

メモリ内のデータは，ビッグ・エンディアンCPUでさえもリトル・エンディアン・モードで順序づけされています。

すべての内部レジスタとFIFO内のデータは，CPUのエンディアン・モードとは無関係にリトル・エンディアンとみなされます。

ワード内のデータ・アドレスあるいはデータ・バイト・オーダは，“ BIG ”信号がロウのとき，リトル・エンディアンのVR4120Aからすべてのローカル・レジスタやメモリへのアクセスに対して，デバイスの中でスワップされません。

データ・アドレスは，“ BIG ”信号と“ ENDCEN ”信号がハイのとき，ビッグ・エンディアンのVR4120Aからすべてのローカル・レジスタやメモリへのアクセスに対して，デバイスの中でスワップされます。

ワード内のデータ・バイト・オーダは，“ BIG ”信号がハイ，“ ENDCEN ”信号がロウのとき，ビッグ・エンディアンのCPUからすべてのローカル・レジスタやメモリへのアクセスに対して，デバイスの中でスワップされます。

保守／廃止

3.7.2　エンディアン・モード

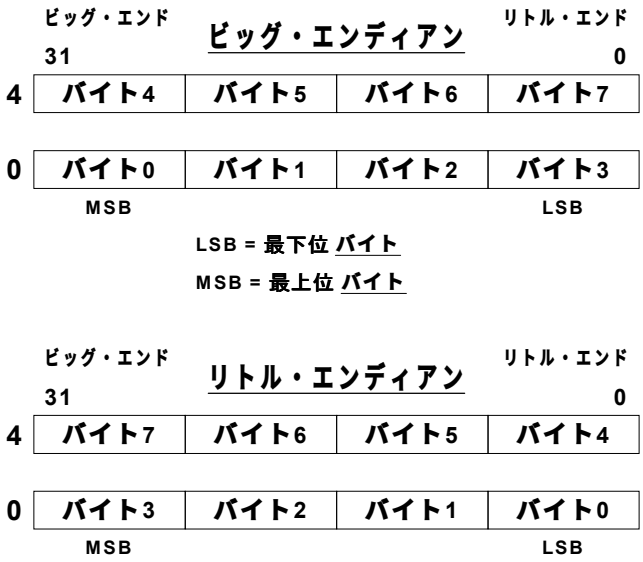
デバイスのエンディアン・モードは，そのワード・アドレシング方法とバイト・オーダを参照します。

ビッグ・エンディアン・デバイスは，ビッグ・エンド（最上位ビット・ナンバ）でデータ項目の番地づけを行います。番地づけされたデータ項目の最上位バイト（MSB）は，最下位アドレスにあります。

リトル・エンディアン・デバイスは，リトル・エンド（最下位ビット・ナンバ）でデータ項目の番地づけを行います。番地づけされたデータ項目の最上位バイト（MSB）は，最上位アドレスにあります。

次の図は，2つのエンディアン・モードのビット／バイト・オーダを示していて，ワード・サイズのデータ内のバイトに適用されます。バイト内のビット・オーダは，両方のモードとも同じです。ビッグ（最上位）ビットは左側に，リトル（最下位）ビットは右側にあります。サブ項目のビット・オーダだけが，2つのエンディアン・モードにわたるとき，より大きな番地づけが可能なデータ項目（ハーフ・ワード，ワード，ダブル・ワード）内で反転されます。より大きなデータ項目内のサブ項目の上位／下位のオーダは，変化しません。たとえば，ワード内の最下位ハーフ・ワード（LSHW）は常に右側に，最上位ハーフ・ワード（MSHW）は左側にあります。

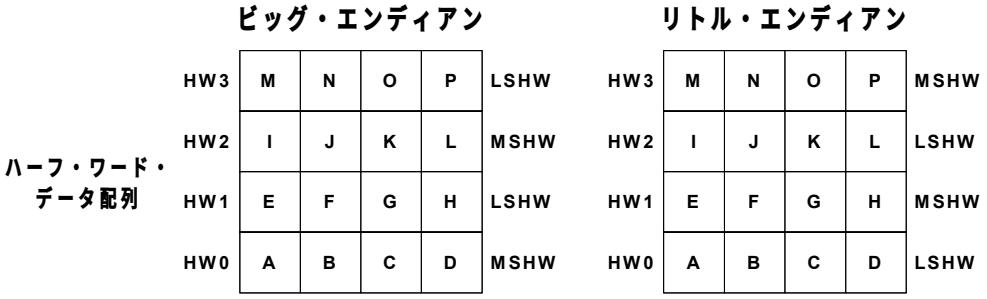
図3−3　エンディアン・モードのビット／バイト・オーダ



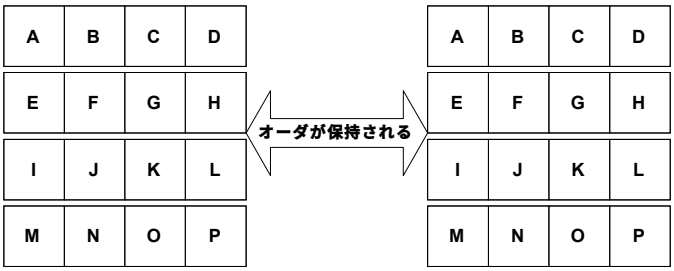
アクセス・タイプがデータ項目タイプと一致していれば，データのサブ項目のスワップは必要ではありません。このようにして，ハーフ・ワード・データから成るデータ配列にハーフ・ワード・アクセスをするときは，バイト・スワップは発生しません。この場合，データ項目のビット・オーダは，2つのエンディアン・モードのあいだで保持されます。ハーフ・ワード・データ配列にシーケンシャルにアクセスするコードは，そのVR4120Aのエンディアン・モードにかかわらず，同じです。コードはエンディアンに依存しません。

保守／廃止

図3－4　ハーフ・ワード・データ配列の例



シーケンシャル・ハーフ・ワード・アクセスを用いたデータ抽出

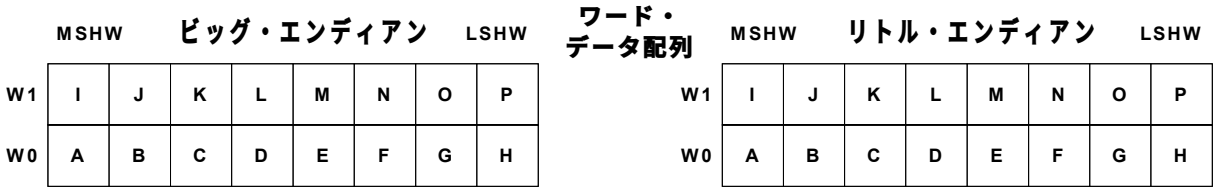


シーケンシャル・ハーフ・ワード・アクセスを用いたデータ抽出

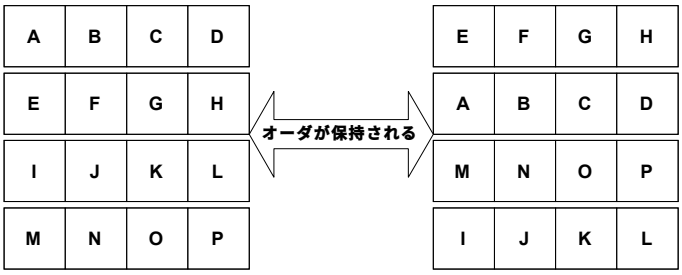
しかし，ワード・データから成るデータ配列にハーフ・ワード・アクセスをするときは，より上位（下位）のハーフ・ワードに対するアクセスは，より下位（上位）のハーフ・ワードに相当するアドレスを必要とします。そのようなコードは，エンディアンに依存しています。スーパー・グループ・アクセス（たとえば，ハーフ・ワード・データ配列から2つのハーフ・ワードを1つのワードとして同時にアクセスすること）は，同じ問題を引き起こします。そのような問題は，32ビット・レジスタにハーフ・ワード・アクセスをした場合にも発生しますが，32ビット・レジスタにワード・アクセスをした場合には問題は起こりません。

保守／廃止

図3－5　ワード・データ配列の例



シーケンシャル・ハーフ・ワード・アクセスを用いたデータ抽出



シーケンシャル・ハーフ・ワード・アクセスを用いたデータ抽出

保守／廃止

第4章　ATMセル・プロセッサ

4.1 概　　要

このセクションでは，ATMセル・プロセッサ・ユニットの機能上の仕様について記述します。

4.1.1 機能の特徴

ATMセル・プロセッサの特徴は，次のとおりです。

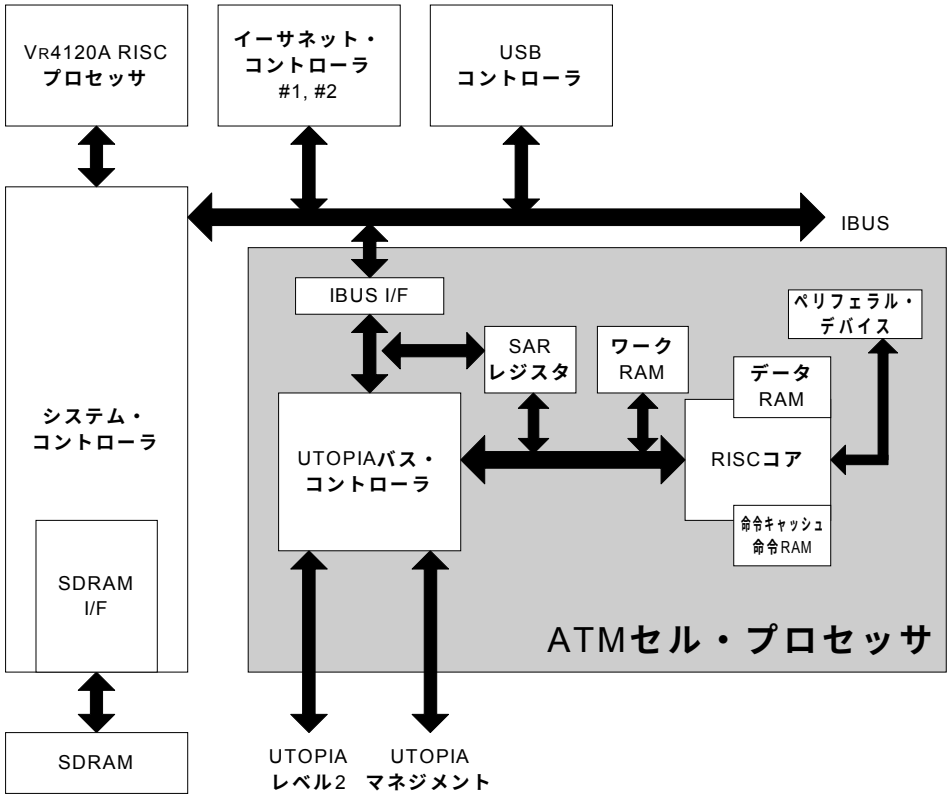
- データ伝送能力
  トータルのデータ伝送能力は50 Mbpsです。ダウンストリーム，アップストリームともそれぞれ25 Mbpsです。
- ATMアダプテーション・レイヤ（AAL）
  AAL-0（Rawセル），AAL-2, AAL-5をサポートします。
- 3つのサービス・クラスのサポート
  CBR, VBR, UBR
- VC数
  最大64のVCをサポートします。
- スケジューリング
  セル・レートのシェーピングは，VCごとに1セル時間の精度で行われます。
- OAM機能のサポート
- スイッチング機能のサポート

ATMセル・プロセッサには，32ビットのマイクロコントローラを内蔵しています。上記のすべての機能は，ハードウエア単独ではなく，ハードウエア上でファームウエアが実行されることにより実現されます。

保守／廃止

4. 1. 2　ATMセル・プロセッサのブロック図

図4－1　ATMセル・プロセッサのブロック図



ATMセル・プロセッサの内部ブロック図を示します。32ビットのRISCコア，ペリフェラル・デバイス（割り込みコントローラ，セル・タイマ，スケジューリング・テーブル，Rxルックアップ・テーブル），DMAコントローラ，ワークRAM，SARレジスタで構成されています。

4. 1. 2. 1　RISCコア

このブロックは，ATMのSAR機能をファームウエアで実現する際のメインとなるRISCマイクロコントローラです。その特徴は，次のとおりです。

- 高性能32ビットRISCマイクロコントローラ，76 MIPS@66 MHz
- 32×32ビット汎用レジスタ
- 32ビットALU, 32ビット・シフタ，16×16乗算加算器
- 1Kバイト・データRAM, 8Kバイト命令RAM, 8Kバイト命令キャッシュ

4. 1. 2. 2　ペリフェラル・デバイス

- 割り込みコントローラ（INTC）と割り込みエッジ検出回路（INTEDGE）
- ATM用の各種ブロック － スケジューリング・テーブル，Rxルックアップ・テーブル，セル・タイマ

保守／廃止

4. 1. 2. 3　UTOPIAバス・コントローラ

このブロックには，いくつかのハードウエア・リソース（DMAコントローラ，FIFO, CRC演算器／チェッカ）があります。その特徴は，次のとおりです。

- スキャッタ／ギャザDMAコレクタ：ファームウエアによる直接のデータ転送制御を介さずに，ディスクリプタ・テーブルによって指定されたデータを転送します。DMAコントローラは，ATMセル・データのそれぞれの送信と受信に使われます。
  また，通常のDMAモードもサポートします。
  さらに，このDMAコントローラは，ワークRAM内のVCテーブルにおけるDMAオペレーションに関する情報も更新します。
- 内部バス・インタフェース（IBUS）
- ATMゼロ・パディング
  ゼロ・パディングは，AAL-5機能で必要とされます。このブロックには，ATMセルに対してゼロ・パディングを行うための回路があります。ソース・アドレスとパディングするバイト数が与えられると，このブロックは指示されたバイト数のゼロ・パディングを挿入します。
- 送信／受信SAR FIFO
  UTOPIAインタフェース制御ブロックには，送信／受信のそれぞれ4セルで構成されるFIFOがあります。TxFIFO内の最後のセルとRxFIFO内の最初のセルは，VR4120AのRISCプロセッサ／RISCコア・メモリ・スペースにマッピングされています。
  UTOPIA2インタフェースは8ビットのバス・インタフェースであり，ATMフォーラム・ドキュメント“ATM-PHY-0039”に定義されているPHYデバイスを制御します。UTOPIAマネジメント・インタフェースもサポートしています。
  特徴は次のとおりです。
  - μPD98502の内蔵通信IPはすべてリトル・エンディアンで扱いますが，ATMではセル・ヘッダの最初の2ワードとペイロードの最後の2ワードをビッグ・エンディアン・バイトで扱う必要があるため，エンディアン変換機能を持っています。
  - ライン・ヘッドのブロッキングを避けるため，デスティネーションPHYデバイスがレディの状態になければ，後続のセルが先行するセルを追い越すこともできます。
  - Rx側では，ヘッダ・パターンの検出とチェックを行い，アイドル・セルとアンアサインド・セルを除去します。
  - 3つの異なる周波数のクロックをTxクロックおよびRxクロックとして供給します。周波数は，CLKUSL［1：0］信号で定義されます。33 MHzクロックがSCLKとして使われた場合は，周波数は以下のようにして決定されます。

    33 MHz　：CLKUSL［1：0］= 00（SCLKと同一周波数）
    16.5 MHz：CLKUSL［1：0］= 01（SCLKの1/2の周波数）
    25 MHz　：CLKUSL［1：0］= 10（SCLKの3/4の周波数）
    出力なし：CLKUSL［1：0］= 11（設定禁止）

・CRC-32/CRC-10演算器／チェッカ

UTOPIAバス・コントローラ・ブロックは，送信／受信パケットそれぞれのためのCRC-32演算器を持っています。CRC-32の値は，パケットごとに計算されます。送信時には，CRC-32の演算結果をトレイラ内のCRC-32フィールドに挿入します。受信時には，エラーが発生していないかどうかチェックするために，CRC-32の演算結果とトレイラ内のCRC-32フィールドの値を比較します。

UTOPIAバス・コントローラ・ブロックは，送信／受信それぞれのCRC-10演算器も持っています。ユーザは，CRC-10をペイロードに含むかどうかを選択できます。CRC-10の値は，セルごとに計算されます。送信時には，CRC-10のモードが選択されていれば，ペイロードの最後の10ビット領域に挿入します。受信時には，CRC-10のモードが選択されていれば，ペイロードの最後の10ビットの値と比較します。

4. 1. 2. 4　その他のブロック

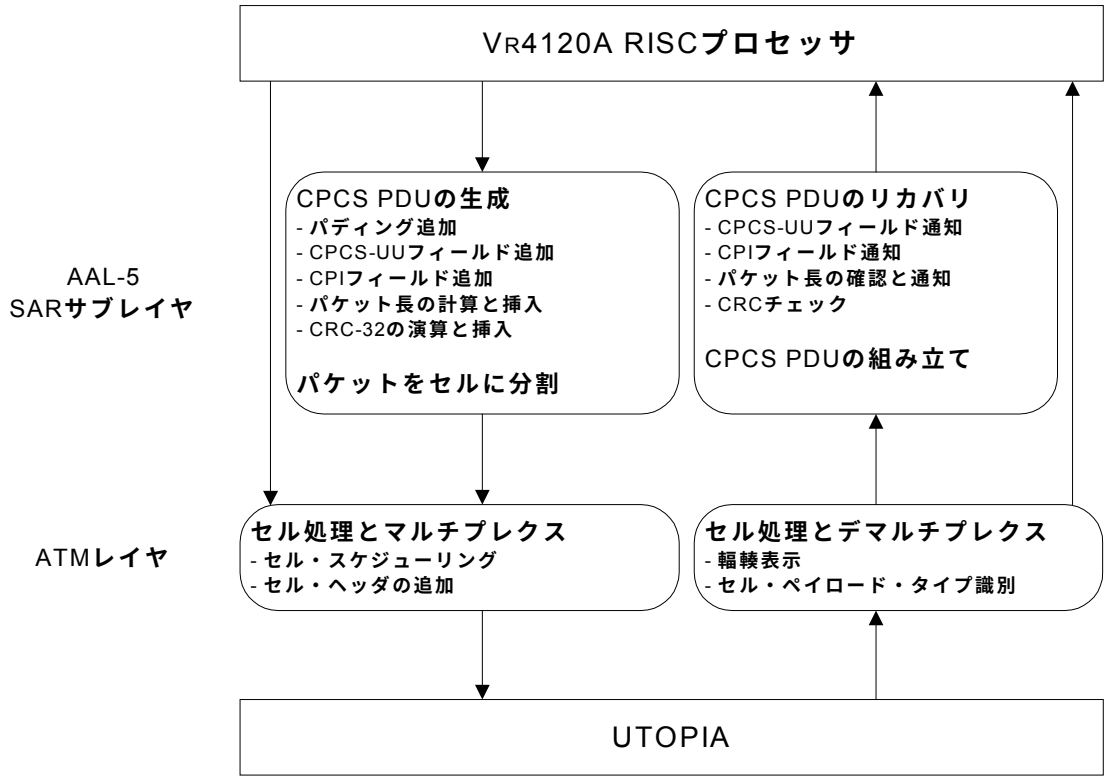
ワークRAMは，16 Kバイトのメモリです。テーブルやプール・ディスクリプタはファームウエアによってこのRAMに配置されています。RISCコアとUTOPIAバス・コントローラ・ブロックの間で共有されます。Indirect_Accessコマンドで，VR4120A RISCプロセッサからのアクセスも可能です。

4. 1. 3　ATMセル処理動作の概要

このセクションでは，ATMセル処理動作の概要だけを記述します。詳細については，4. 7を参照してください。

ATMセル・プロセッサは，AAL-5 SARサブレイヤとATMレイヤ機能をサポートします。このブロックは，LLCのカプセリングを提供します。

図4－2　AAL-5サブレイヤとATMレイヤ

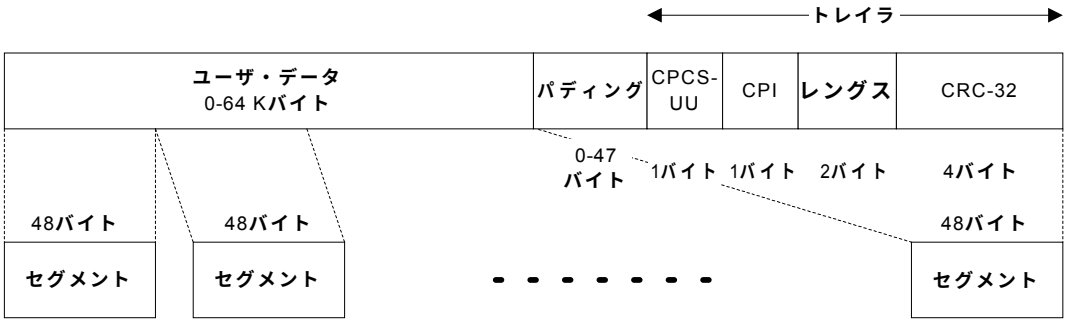


保守／廃止

4. 1. 3. 1　AAL-5 SARサブレイヤ機能

ATMセル・プロセッサがAAL-5モードでセルを送信するときは，ユーザ・データにトレイラを付加して，さらに，全体の長さが48バイトの倍数になるように，パディングを追加して，AAL-5 PDUを生成します。ATMセル・プロセッサがセルを受信すると，CPCS PDUを組み立てるためにSDRAM内にそれらのセルをいったん格納します。ATMセル・プロセッサは，組み立てられたCPCS PDUのトレイラを検証します。エラーが発生すると，ATMセル・プロセッサはその結果を受信表示でVR4120A RISCプロセッサに通知します（4. 8. 3. 3 受信表示参照）。

図4－3　AAL-5サブレイヤとATMレイヤ



(a) パディング・フィールド：パケットの長さを48バイトの倍数に調整するためにユーザ・データとトレイラの間に挿入される0-47バイトのフィールドです。ATMセル・プロセッサはこのフィールドのすべてのビットにゼロを書き込みます。

(b) CPCS-UUフィールド：ユーザ情報を転送するために使用します。ホストがパケット・ディスクリプタに設定した値がこのフィールドに書き込まれます。

(c) CPIフィールド：このフィールドの使い方は現在のATMフォーラムでは決まっていません。現在の仕様では，すべてのビットをゼロにセットすることになっています。ただし，ATMセル・プロセッサでは，CPCS-UUフィールドの場合と同様に，パケット・ディスクリプタにセットされた値をこのフィールドに書き込み，送信することができます。

(d) レングス・フィールド：ユーザ・データ長を2進数で表示します。

(e) CRC-32フィールド：ユーザ・データからレングス・フィールドの終わりまでの範囲に対してのCRC-32演算結果をこのフィールドに設定します。生成多項式は次のとおりです。

$$G(X) = 1+X+X^2+X^4+X^5+X^7+X^8+X^{10}+X^{11}+X^{12}+X^{16}+X^{22}+X^{23}+X^{26}+X^{32}$$

4. 1. 3. 2　ATMレイヤ機能

(1) トラフィック・クラス

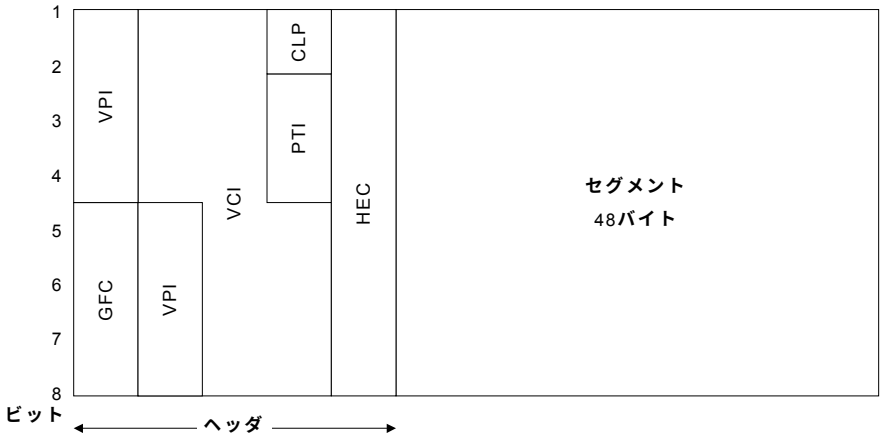
ATMセル・プロセッサは3つのトラフィック・クラスをサポートします。CBR（Constant Bit Rate：一定のビット・レート），VBR（Variable Bit Rate：可変ビット・レート），UBR（Unspecified Bit Rate：非帯域確保のビット・レート）の3つです。

(2) セル生成

ATMセル・プロセッサは，次の図に示すようにセグメントに5バイトのヘッダを追加して，セルを生成します。

保守／廃止

図4－4　ATMセル



ヘッダの各フィールドの機能は次のとおりです。

(a) GFC（General Flow Control：一般フロー制御）フィールド

フロー制御に使用します。送信時には，パケット・ディスクリプタに設定した値をこのフィールドに書き込みます。受信時には，このフィールドは無視されます。

(b) VPI（Virtual Path Identifier：バーチャル・パス識別子）/VCI（Virtual Channel Identifier：バーチャル・チャネル識別子）フィールド

ルーチング・パスを示すルーチング・フィールドです。ATMセル・プロセッサは，送信／受信において合計64通りのVPI/VCIの組み合わせをサポートします。送信時には，Tx VCテーブルに設定した24ビットの値をこれらのフィールドに書き込みます。受信時には，受信ルックアップ・テーブルに登録されたVPI/VCIだけが受信することを許されます。

(c) PTI（Payload Type Indication：ペイロード・タイプ表示）フィールド

セル・ペイロードがユーザ・データであるかマネジメント・データであるかを示す3ビットのフィールドです。輻輳情報も含んでいます。

PTI	用　途
000	ユーザ・データ，輻輳なし，SDUタイプ = 0
001	ユーザ・データ，輻輳なし，SDUタイプ = 1
010	ユーザ・データ，輻輳発生，SDUタイプ = 0
011	ユーザ・データ，輻輳発生，SDUタイプ = 1
100	OAM F5フロー・セル
101	OAM F5フロー・セル
110	将来の使用のため予約
111	将来の使用のため予約

SDUタイプ = 0　：AAL-PDUの最終セルを除くすべてのセグメント

SDUタイプ = 1　：AAL-PDUの最終セル。このセグメントにはトレイラが含まれます。しかし，SDUタイプはファームウエアが自動で設定するため，ユーザが設定する必要はありません。

OAM F5フロー・セル：オペレーション，管理，保守用のセル。

(d) CLP（Cell Loss Priority：セル損失優先度）フィールド

ネットワークの輻輳時に，このセルを優先的に破棄するかどうかを示します。CLPの値が1のときは，セルは優先的に破棄すべきことを示します。ATMセル・プロセッサは，パケット・ディスクリプタのCLPMフィールドに従ってこのフィールドに適切な値を設定します。

(e) HEC（Header Error Control：ヘッダ・エラー制御）フィールド

セル同期，ヘッダ・エラーの検出と訂正のために使用されます。このフィールドは，TCサブレイヤで処理されます。

(3) セル・スケジューリング

ATMセル・プロセッサは，セル・スケジューリングのために，スケジューリング・テーブル，セル・タイマ，Tx VCテーブルを使用します。VR4120Aがパケットを送信する前に，Tx VCテーブルにレート情報を設定します。ATMセル・プロセッサは，レート情報からセル送信のインターバルを計算し，次の送信時間をスケジューリング・テーブルに書き込みます。セル・タイマと当該VCの次の送信時間が同じになると，VCテーブルで指定されたセルが送信されます。

VCがCBRまたはUBRである場合，PCR（Peak Cell Rate：ピーク・セル・レート）だけをスケジューリングに使用します。VCがVBRの場合は，PCR, SCR（Sustained Cell Rate：維持されるセル・レート），MBS（Maximum Burst Size：最大バースト・サイズ）を使用します。

(4) AAL-2サポート/OAMサポート

ATMセル・プロセッサは，AAL-2とOAM F5機能もサポート可能です。詳細については，当社販売部門にお問い合わせください。

(5) Rawセル・サポート

ATMセル・プロセッサは，非AAL-5トラフィックをサポートするために，セルをRawセルとして扱うこともできます。Rawセル・モードとして設定されたVCに，ATMセル・プロセッサは，CRC-32演算をしたりトレイラを追加したりするような，AAL-5に依存するオペレーションを実行しません。受信モードにおいては，ATMセル・プロセッサは，受信したセルをヘッダと11バイトの受信表示とともにSDRAMに格納します。

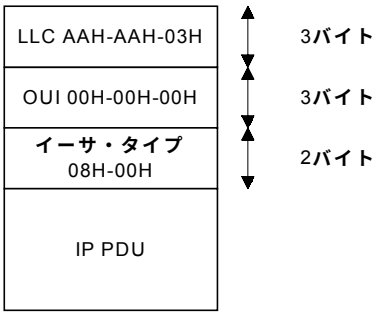
ATMセル・プロセッサには，非AAL-5トラフィックに対してCRC-10挿入と検証機能があります。CRC-10の挿入をイネーブルにすると，ATMセル・プロセッサは，セルごとにCRC-10を計算し，その結果を送信側のペイロードの最後に挿入します。ATMセル・プロセッサは，セルを受信するときには常にCRC-10を検証します。ATMセル・プロセッサがエラーを検出すると，エラー・フラグを受信表示にセットし，VR4120Aに通知します。

保守／廃止

4. 1. 3. 3　LLCカプセリング

LLCカプセリング・モードがVCテーブルにセットされると，ATMセル・プロセッサは，LLCヘッダをIPパケットの上部に追加します。この場合，ATMセル・プロセッサは，CPCS-PDUをIP PDUとして常にカプセリングします。しかし，ATMモードのTx_Readyコマンドを使用すると，ATMセル・プロセッサはカプセリングを行いません。

図4－5　LLCカプセリング



保守／廃止

4.2　メモリ空間

ATMセル・プロセッサ内のRISCコアは32ビットのMPUですが，物理的なメモリ空間は24ビット幅です。

図4－6　VR4120AとRISCコアのメモリ空間

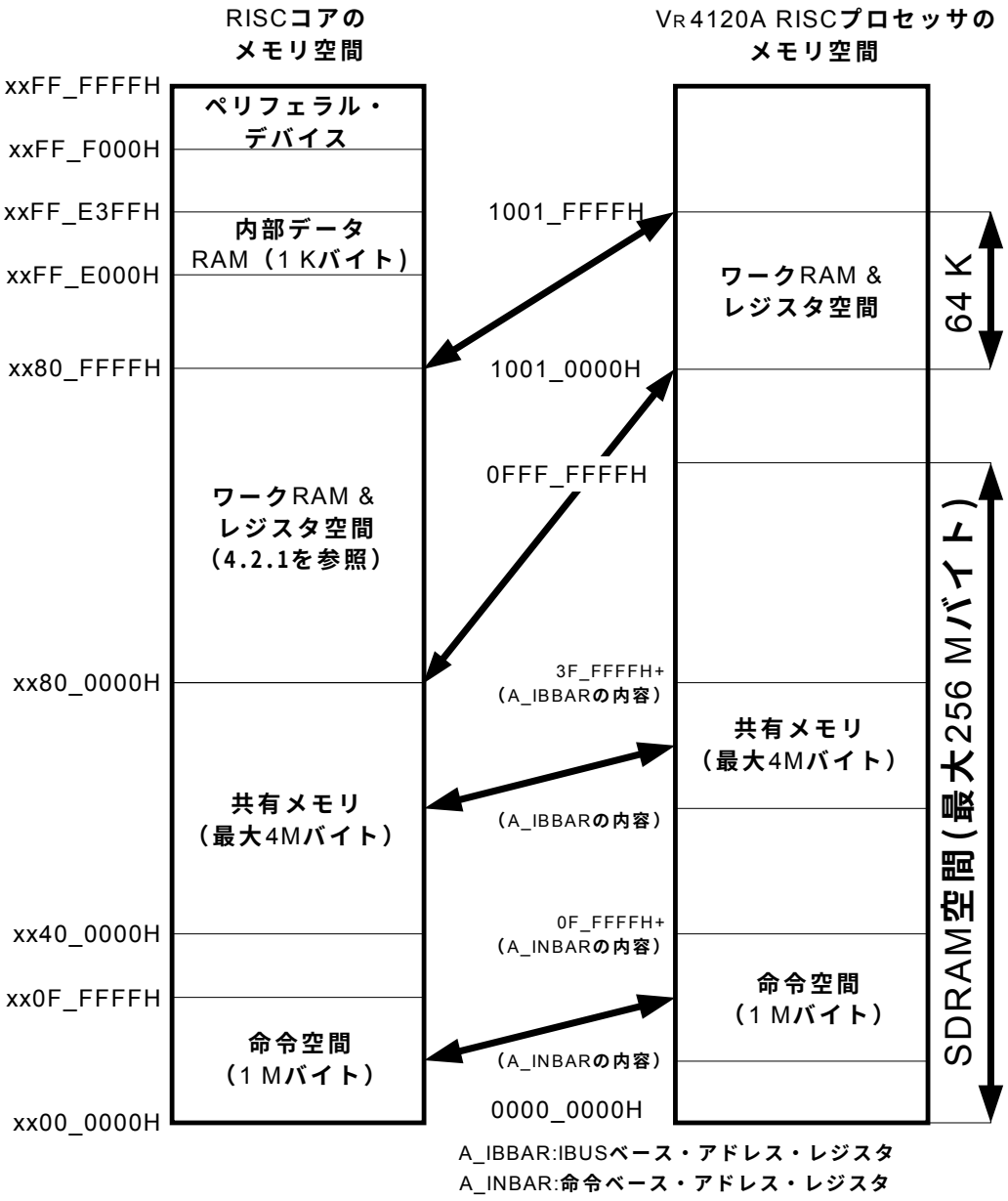


図4－6に構成を示します。命令空間，共有メモリ空間，ワークRAM，内部メモリ空間，ペリフェラル・デバイス空間を含んでいます。

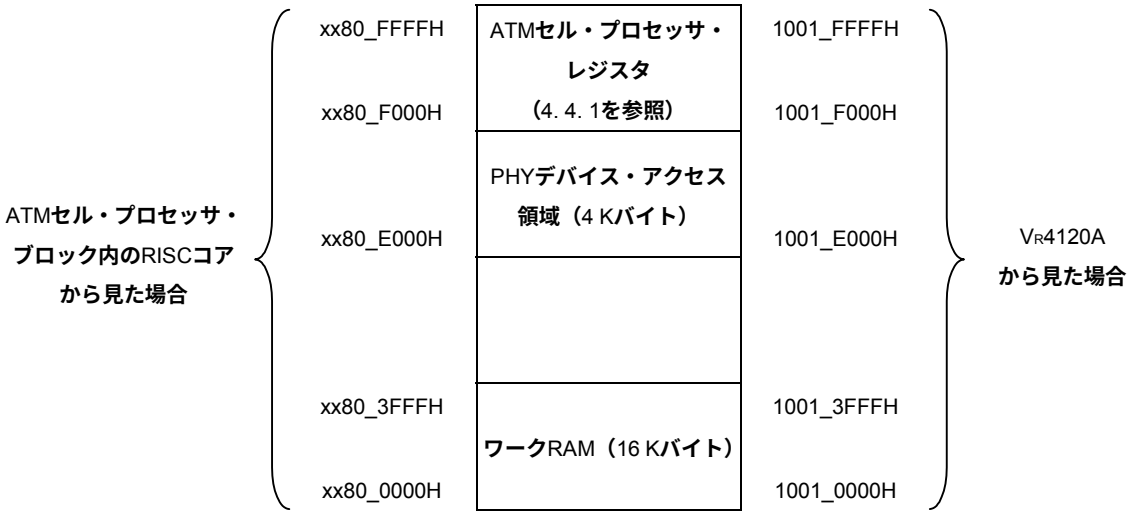
VR4120AとATMセル・プロセッサ内のRISCコアは，外部メモリ空間を共有します。共有メモリは，SDRAMデバイスを共有することによって実現します。VR4120Aのメモリ空間のアドレスは，ソフトウエアで決定され，A_IBBAR（IBUSベース・アドレス・レジスタ）を設定することで，RISCコアに通知されます。その容量は，物理的メモリ容量の合計に依存しますが，4 Mバイトを越えることはありません。

保守／廃止

4. 2. 1　ワークRAMとレジスタ空間

ワークRAMとレジスタ空間を図4－7に示します。ワークRAMの容量は16 Kバイトです。ワークRAMにアクセスするためには，ユーザは“ Indirect_Accessコマンド ”（4. 7. 6　Indirect_Accessコマンドを参照）を使用しなければなりません。レジスタ空間には，A_GMR（ジェネラル・モード・レジスタ），A_GSR（ジェネラル・ステータス・レジスタ），A_CMR（コマンド・レジスタ），A_CER（コマンド拡張レジスタ）やほかのレジスタがマップされます。PHYデバイス・アクセス領域では，UTOPIAマネジメント・インタフェースを介してPHYデバイスにアクセスできます。

図4－7　ワークRAMとレジスタ空間



内部データRAMとペリフェラル・デバイス空間は，ATMセル・プロセッサ内部のRISCコアだけからアクセスでき，ほかのブロックからアクセスできません（ただしRxルックアップ・テーブルは除く）。内部データRAMは，スタックおよびグローバル・スペースとして使用されます。ペリフェラル・デバイス空間には，割り込みコントローラとほかのいくつかの特別なブロックがマップされます。スケジューリング・テーブル，VCルックアップ・テーブル，セル・タイマもペリフェラル・デバイス空間にマップされます。

4. 2. 2　共有メモリ

ATMセル・プロセッサは，セル・バッファおよびパケット・バッファとして使用する4 Mバイト以下のメモリ・スペースをアクセスできます。これは命令メモリとしても使用できます。このメモリは，SDRAM上で実現します。ATMセル・プロセッサ内のRISCコアは，システム・コントローラを介してこのメモリにアクセスできます。VR4120Aから見て，メモリにアクセスする際のベース・アドレスは，A_IBBAR（IBUSベース・アドレス・レジスタ）に設定してください。なお，A_IBBARの設定は，RISCコアの動作開始前にVR4120Aによって行ってください。

保守／廃止

4.3　割り込み

A_GSR（ジェネラル・ステータス・レジスタ）内の該当するビットが，“1”にセットされ，かつA_IMR（割り込みマスク・レジスタ）内の相当するビットが“1”の場合に，割り込みをVR4120Aに発行します。割り込みの要因は，A_GSRを読み込むことで得られます。VR4120AがA_GSRを読み込むと，ビットはリセットされます。割り込みは，A_IMR内の相当する要因のビットをリセット（“0”に設定）することでマスクできます。

PHYデバイスからの割り込み（UMINT_B信号）は，A_GSRのPIビットを介してVR4120Aに通知します。

保守／廃止

4.4　ATMセル処理用のレジスタ

ATMセル・プロセッサ・ブロック内のレジスタは，3つのグループに分類されます。SARレジスタ，DMAレジスタ，FIFO制御レジスタです。これらのレジスタは，VR4120AとATMセル・プロセッサ内のRISCコアの両方からアクセスできます。

4.4.1　レジスタ・マップ

レジスタは，SAR機能制御のために使用します。VR4120Aは，これらのレジスタにデータを書き込んでSAR機能を制御し，これらのレジスタからデータを読み込んでステータス情報を得ます。RISCコア上のファームウエアは，これらのレジスタを読んでVR4120Aからの指示を知り，これらのレジスタに書き込んでATMセル・プロセッサの状態を表示します。

4.4.1.1　直接アドレス・レジスタ

VR4120Aから見て，1001_0000HはATMセル・プロセッサ内のレジスタにアクセスするためのベース・アドレスです。

アドレス	レジスタ名	R/W	アクセス	説　明
1001_F000H	A_GMR	R/W	W	ジェネラル・モード・レジスタ
1001_F004H	A_GSR	RC	W	ジェネラル・ステータス・レジスタ
1001_F008H	A_IMR	R/W	W	割り込みマスク・レジスタ
1001_F00CH	A_RQU	R	W	受信キュー・アンダフロー・レジスタ
1001_F010H	A_RQA	R	W	受信キュー枯渇警告レジスタ
1001_F014H	N/A	-	-	将来の使用のため予約
1001_F018H	A_VER	R	W	バージョン・レジスタ
1001_F01CH	N/A	-	-	将来の使用のため予約
1001_F020H	A_CMR	R/W	W	コマンド・レジスタ
1001_F024H	N/A	-	-	将来の使用のため予約
1001_F028H	A_CER	R/W	W	コマンド拡張レジスタ
1001_F02CH：1001_F04CH	N/A	-	-	将来の使用のため予約
1001_F050H	A_MSA0	R/W	W	メールボックス0スタート・アドレス・レジスタ
1001_F054H	A_MSA1	R/W	W	メールボックス1スタート・アドレス・レジスタ
1001_F058H	A_MSA2	R/W	W	メールボックス2スタート・アドレス・レジスタ
1001_F05CH	A_MSA3	R/W	W	メールボックス3スタート・アドレス・レジスタ
1001_F060H	A_MBA0	R/W	W	メールボックス0ボトム・アドレス・レジスタ
1001_F064H	A_MBA1	R/W	W	メールボックス1ボトム・アドレス・レジスタ
1001_F068H	A_MBA2	R/W	W	メールボックス2ボトム・アドレス・レジスタ
1001_F06CH	A_MBA3	R/W	W	メールボックス3ボトム・アドレス・レジスタ
1001_F070H	A_MTA0	R/W	W	メールボックス0テール・アドレス・レジスタ
1001_F074H	A_MTA1	R/W	W	メールボックス1テール・アドレス・レジスタ
1001_F078H	A_MTA2	R/W	W	メールボックス2テール・アドレス・レジスタ
1001_F07CH	A_MTA3	R/W	W	メールボックス3テール・アドレス・レジスタ
1001_F080H	A_MWA0	R/W	W	メールボックス0ライト・アドレス・レジスタ
1001_F084H	A_MWA1	R/W	W	メールボックス1ライト・アドレス・レジスタ

保守／廃止

アドレス	レジスタ名	R/W	アクセス	説　明
1001_F088H	A_MWA2	R/W	W	メールボックス2ライト・アドレス・レジスタ
1001_F08CH	A_MWA3	R/W	W	メールボックス3ライト・アドレス・レジスタ
1001_F090H	A_RCC	R	W	有効受信セル・カウンタ
1001_F094H	A_TCC	R	W	有効送信セル・カウンタ
1001_F098H	A_RUEC	R	W	受信無効VPI/VCIエラー・セル・カウンタ
1001_F09CH	A_RIDC	R	W	受信内部廃棄セル・カウンタ
1001_F0A0H： 1001_F0BCH	N/A	-	-	将来の使用のため予約
1001_F0C0H	A_T1R	R/W	W	T1タイム・レジスタ
1001_F0C4H	N/A	-	-	将来の使用のため予約
1001_F0C8H	A_TSR	R/W	W	タイム・スタンプ・レジスタ
1001_F0CCH： 1001_F1FCH	N/A	-	-	将来の使用のため予約
1001_F200H： 1001_F2FCH	N/A	-	-	VR4120Aコアからアクセスできません。この領域は内部機能が使用。
1001_F300H	A_IBBAR	R/W	W	IBUSベース・アドレス・レジスタ
1001_F304H	A_INBAR	R/W	W	命令ベース・アドレス・レジスタ
1001_F308H： 1001_F31CH	N/A	-	-	将来の使用のため予約
1001_F320H	A_UMCMD	R/W	W	UTOPIAマネジメント・インタフェース・コマンド・レジスタ
1001_F324H： 1001_F3FCH	N/A	-	-	将来の使用のため予約
1001_F400H： 1001_F4FCH	N/A	-	-	VR4120Aコアからアクセスできません。この領域は内部機能が使用。
1001_F500H： 1001_FFFCH	N/A	-	-	将来の使用のため予約

備考1. “R/W”フィールド内で，
“W”は，“書き込み可能”を意味します。
“R”は，“読み取り可能”を意味します。
“RC”は，“読み取りクリア”を意味します。
“-”は，“アクセス不可能”を意味します。

2. すべての内部レジスタは，32ビット・ワード配列のレジスタです。
3. 内部レジスタに対するバースト・アクセスは，禁止されています。バースト・アクセスすると，NSRのIRERRビットがセットされ，NMIがCPUに対してアサートされます。
4. 予約領域に対してリード・アクセスした場合，NSRレジスタのCBERRビットがセットされ，SysCMD［0］にデータ・エラー・ビットが設定されたダミー読み取り応答データが返信されます。
5. 予約領域に対してライト・アクセスした場合，NSRレジスタのCBERRビットがセットされ，書き込みデータが失われます。
6. “アクセス”フィールド内で，
“W”は，ワード・アクセスが有効であることを意味します。
“H”は，ハーフ・ワード・アクセスが有効であることを意味します。
“B”は，バイト・アクセスが有効であることを意味します。

備考7.　**リード・オンリー・レジスタに対するライト・アクセスはエラーになりませんが，書き込みデータは失われます。**

8.　CPU**はすべての内部レジスタにアクセスできますが，**IBUS**マスタ・デバイスはそれらのレジスタにアクセスできません。**

4. 4. 1. 2　Rx**ルックアップ・テーブル制御レジスタ**

アドレス[注]	レジスタ名	R/W	アクセス	説　明
FFF410H	RXLCTR	R/W	W	Rx**ルックアップ・テーブル制御レジスタ**
FFF600H	RxTBL000	W	H	Rx**ルックアップ・テーブル・エントリ**00**ハーフ・ワード**0
FFF602H	RxTBL001	W	H	Rx**ルックアップ・テーブル・エントリ**00**ハーフ・ワード**1
FFF6FCH	RxTBL3F0	W	H	Rx**ルックアップ・テーブル・エントリ**3F**ハーフ・ワード**0
FFF6FEH	RxTBL3F1	W	H	Rx**ルックアップ・テーブル・エントリ**3F**ハーフ・ワード**1
FFF700H	RxTBC00	R/W	W	Rx**ルックアップ・テーブル制御エントリ**00
FFF77EH	RXTBC3F	R/W	W	Rx**ルックアップ・テーブル制御エントリ**3F

注　これらのアドレスは，Indirect_Access**コマンド（**4. 7. 6　Indirect_Access**コマンドを参照）で指定します。**

4. 4. 2　A_GMR**（ジェネラル・モード・レジスタ）**

A_GMR**は，このブロックのオペレーション・モードを選択するために使用し，**ATM SAR**オペレーションをイネーブル／ディスエーブルにします。リセット後，**VR4120A**はATM**セル・プロセッサを動作させるために必要なすべての初期化を行い，準備が完了したところで通信を開始するために，このレジスタに送受信イネーブルを書き込みます。このレジスタのすべてのビットは，書き込み可能ですが，ビット**31-15, 13-2**は将来の使用のために予約されています。初期値はすべて**0**です。**

ビット	フィールド	R/W	ディフォールト	説　明
31：15	Reserved	R/W	0	**将来の使用のため予約。**0**を書き込んでください。**
14	LP	R/W	0	0 = UTOPIA**インタフェースでループバックを行わない** 1 = UTOPIA**インタフェースでループバックを行う**
13：2	Reserved	R/W	0	**将来の使用のため予約。**0**を書き込んでください。**
1	TE	R/W	0	0 = **送信ディスエーブル** 1 = **送信イネーブル**
0	RE	R/W	0	0 = **受信ディスエーブル** 1 = **受信イネーブル**

保守／廃止

4.4.3　A_GSR（ジェネラル・ステータス・レジスタ）

A_GSRは，割り込みの状態を示します。割り込みを引き起こすイベントが発生すると，RISCコア上のファームウエアは，イベントの種類に対応するA_GSR内のビットをセットします。A_IMR（割り込みマスク・レジスタ）内の対応するビットがセットされ，割り込みがマスクされていないと，割り込みをVR4120Aに対して発行します。A_GSR内のビットは，リード・クリアとなっています。ビットのリード後に同じ種類のイベントが発生すると，ビットは再びセットされます。

初期値はすべて0です。

ビット	フィールド	R/W	ディフォールト	説　明
31	PI	RC	0	0 = PHYレイヤ・デバイスからの割り込み入力がされていない 1 = PHYレイヤ・デバイスからの割り込み入力がされた
30	RQA	RC	0	0 = 受信キュー枯渇警告は発生していない 1 = 受信キュー枯渇警告が発生した
29	RQU	RC	0	0 = 受信キュー・アンダフローは発生していない 1 = 受信キュー・アンダフローが発生した
28：24	Reserved	R	0	将来の使用のため予約
23	SQO	RC	0	0 = スケジューリング・キュー・オーバフローは発生していない 1 = スケジューリング・キュー・オーバフローが発生した
22	Reserved	R	0	将来の使用のため予約
21	FER	RC	0	0 = 致命的エラーは発生していない 1 = 致命的エラーが発生した
20：17	Reserved	R	0	将来の使用のため予約
16	BER	RC	0	0 = 内部バス・エラーは発生していない 1 = 内部バス・エラーが発生した
15：8	RCR［7：0］	RC	0	0 = Rawセルはプールno.［7：0］にない 1 = Rawセルがプールno.［7：0］にある
7：4	MF［3：0］	RC	0	0 = メールボックスno.［3：0］は満杯でない 1 = メールボックスno.［3：0］が満杯である
3：0	MM［3：0］	RC	0	0 = メールボックスno.［3：0］は更新されていない 1 = メールボックスno.［3：0］が更新された

保守／廃止

4.4.4　A_IMR（割り込みマスク・レジスタ）

A_IMRは，各々の対応するイベントに対して割り込みをマスクします。A_GSR内の対応するビットと同じ位置にあるマスク・ビットが，対応する割り込みをマスクできます。このレジスタのビットを0にセットすると，A_GSRの対応するビットがマスクされます。1にセットすると，対応するビットはマスクされません。マスク解除ビットがセットされて，A_GSR内のビットが1にセットされると，割り込みをVR4120Aに対して発行します。

このレジスタのすべてのビットは書き込み可能ですが，ビット28-24, 22, 20-16は将来の使用のために予約されています。

初期値はすべて0です。

ビット	フィールド	R/W	ディフォールト	説　明
31	PI	R/W	0	PHYレイヤ・デバイスから入力される割り込み（UMINT_B信号）に対するマスク・ビット： 0 = マスク 1 = マスク解除
30	RQA	R/W	0	受信キュー枯渇警告に対するマスク・ビット： 0 = マスク 1 = マスク解除
29	RQU	R/W	0	受信キュー・アンダフローに対するマスク・ビット： 0 = マスク 1 = マスク解除
28：24	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
23	SQO	R/W	0	スケジューリング・キュー・オーバフローに対するマスク・ビット： 0 = マスク 1 = マスク解除
22	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
21	FER	R/W	0	致命的エラーに対するマスク・ビット： 0 = マスク 1 = マスク解除
20：16	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
15：8	RCR［7：0］	R/W	0	Rawセル受信に対するマスク・ビット： 1 = マスク 0 = マスク解除
7：4	MF［3：0］	R/W	0	メールボックス満杯に対するマスク・ビット： 1 = マスク 0 = マスク解除
3：0	MM［3：0］	R/W	0	メールボックス・マークに対するマスク・ビット： 1 = マスク 0 = マスク解除

保守／廃止

4.4.5　A_RQU（受信キュー・アンダフロー・レジスタ）

A_RQUは，各プールの状態を示します。プールに空いているバッファがない場合は，対応するビットをセットします。ATMセル・プロセッサは，セルを受信したときにプールの空きを検出し，セルをバッファに転送することを試みます。A_RQUビットのうちの1つがセットされているときは，常にA_GSR内のRQUビット（ビット29）がセットされます。このブロックでは，プール7-0だけで使用します。ビットが1にセットされたときは，対応するプールに空いているバッファはありません。初期値はすべて0です。

ビット	フィールド	R/W	ディフォールト	説　明
31：8	Reserved	R	0	将来の使用のため予約
7：0	A_RQU［7：0］	R	0	0 = プール［7：0］に空いているバッファがある 1 = プール［7：0］に空いているバッファがない

4.4.6　A_RQA（受信キュー枯渇警告レジスタ）

A_RQAは，VR4120Aが設定した“ アラート・レベル ”よりもキューの残りが少ないプールを示します。A_RQAビットのうちの1つがセットされているときは，常にA_GSR内のRQAビット（ビット30）がセットされます。このブロックでは，プール7-0だけで使用します。ビットが“ 1 ”にセットされたときは，対応するプールの残りのキュー数は“ アラート・レベル ”未満です。初期値はすべて0です。

ビット	フィールド	R/W	ディフォールト	説　明
31：8	Reserved	R	0	将来の使用のため予約
7：0	A_RQA［7：0］	R	0	0 = プール［7：0］に“ アラート・レベル ”以上のキューの残りがある 1 = プール［7：0］のキューの残りが“ アラート・レベル ”未満となっている

4.4.7　A_VER（バージョン・レジスタ）

A_VERは，ATMセル・プロセッサ・ブロックのバージョン・ナンバを示します。初期値は0000_0200Hです。

ビット	フィールド	R/W	ディフォールト	説　明
31：16	Reserved	R	0	将来の使用のため予約
15：8	MAJOR	R	02H	大規模な改訂を行った場合にアップデートします。K規格品は02H固定
7：0	MINOR	R	01H	小規模な改訂を行った場合にアップデートします。K規格品は01H固定

保守／廃止

4.4.8　A_CMR（コマンド・レジスタ）

ATMセル・プロセッサは，VR4120Aがコマンドとパラメータを A_CMRとA_CERに書き込むと，それらを受け取ります。ATMセル・プロセッサは，一度に1つのコマンドしか処理できません。ATMセル・プロセッサがVR4120Aからコマンドを受け取ったときは，レジスタ内のビジィ・フラグを自動的にセットし，ビジィであることを表示します。ビジィ・フラグがセットされている間，VR4120Aは新しいコマンドを発行することができません。ビジィ・フラグがセットされているときにVR4120Aが新しいコマンドを書き込むと，そのコマンドは無効となります。初期値は0です。このレジスタの詳細については，4.7　コマンドを参照してください。

ビット	フィールド	R/W	ディフォールト	説　明
31	BSY	R/W	0	ビジィ・フラグ 0 = コマンド発行可 1 = コマンド発行不可（ビジィ状態）
30：0	A_CMR	R/W	0	コマンドとパラメータ（4.7　コマンド参照）

4.4.9　A_CER（コマンド拡張レジスタ）

A_CERは，コマンド拡張レジスタです。初期値は0です。このレジスタの詳細については，4.7　コマンドを参照してください。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_CER	R/W	0	コマンドのパラメータ（4.7　コマンド参照）

4.4.10　A_MSA0-A_MSA3（メールボックス・スタート・アドレス・レジスタ）

A_MSA0-A_MSA3には，受信メールボックス（メールボックス0とメールボックス1）と送信メールボックス（メールボックス2とメールボックス3）のスタート・アドレスをそれぞれ設定します。初期値はすべて0です。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MSA0	R/W	0	メールボックス0のスタート・アドレス

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MSA1	R/W	0	メールボックス1のスタート・アドレス

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MSA2	R/W	0	メールボックス2のスタート・アドレス

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MSA3	R/W	0	メールボックス3のスタート・アドレス

保守／廃止

4. 4. 11　A_MBA0-A_MBA3（メールボックス・ボトム・アドレス・レジスタ）

A_MBA0-A_MBA3には，受信メールボックス（メールボックス0とメールボックス1）と送信メールボックス（メールボックス2とメールボックス3）のボトム・アドレスをそれぞれ設定します。初期値はすべて0です。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MBA0	R/W	0	メールボックス0のボトム・アドレス

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MBA1	R/W	0	メールボックス1のボトム・アドレス

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MBA2	R/W	0	メールボックス2のボトム・アドレス

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MBA3	R/W	0	メールボックス3のボトム・アドレス

4. 4. 12　A_MTA0-A_MTA3（メールボックス・テール・アドレス・レジスタ）

A_MTA0-A_MTA3には，受信メールボックス（メールボックス0とメールボックス1）と送信メールボックス（メールボックス2とメールボックス3）のテール・アドレスをそれぞれ設定します。初期値は0です。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MTA0	R/W	0	メールボックス0のテール・アドレス

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MTA1	R/W	0	メールボックス1のテール・アドレス

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MTA2	R/W	0	メールボックス2のテール・アドレス

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MTA3	R/W	0	メールボックス3のテール・アドレス

保守／廃止

4. 4. 13　A_MWA0-A_MWA3（メールボックス・ライト・アドレス・レジスタ）

A_MWA0-A_MWA3には，受信メールボックス（メールボックス0とメールボックス1）と送信メールボックス（メールボックス2とメールボックス3）のライト・アドレスをそれぞれ設定します。初期値は0です。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MWA0	R/W	0	メールボックス0のライト・アドレス

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MWA1	R/W	0	メールボックス1のライト・アドレス

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MWA2	R/W	0	メールボックス2のライト・アドレス

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_MWA3	R/W	0	メールボックス3のライト・アドレス

4. 4. 14　A_RCC（有効受信セル・カウンタ）

A_RCCは，有効な受信セル数をカウントする32ビット・カウンタです。このカウンタのオーバフローは，割り込みの要因になりません。初期値は0です。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_RCC	R	0	有効受信セル数

4. 4. 15　A_TCC（有効送信セル・カウンタ）

A_TCCは，有効な送信セル数をカウントする32ビット・カウンタです。このカウンタのオーバフローは，割り込みの要因になりません。初期値は0です。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_TCC	R	0	有効送信セル数

4. 4. 16　A_RUEC（受信無効VPI/VCIエラー・セル・カウンタ）

A_RUECは，VPI/VCIエラーを持つ受信セル数をカウントする32ビット・カウンタです。このカウンタのオーバフローは，割り込みの要因になりません。初期値は0です。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_RUEC	R	0	VPI/VCIエラーを持つ受信セル数

保守／廃止

4. 4. 17　A_RIDC（受信内部廃棄セル・カウンタ）

A_RIDCは，ATMセル・プロセッサ内部で廃棄された受信セル数をカウントする32ビット・カウンタです。このカウンタのオーバフローは，割り込みの要因になりません。初期値は0です。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_RIDC	R	0	廃棄されたセルの数

4. 4. 18　A_T1R（T1タイム・レジスタ）

A_T1Rは，ATMセル・プロセッサが1つのパケット全体を受け取るのに費やすユーザ許容時間を設定します。初期値は0000_FFFFHです。時間カウントのソースはIBUSクロック（66 MHz：33 MHz入力時）です。

ビット	フィールド	R/W	ディフォールト	説　明
31	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
30：0	A_T1R	R/W	FFFFH	1つのパケット全体を受け取るのに許容される時間

4. 4. 19　A_TSR（タイム・スタンプ・レジスタ）

A_TSRは，ATMセル・プロセッサがシステム・クロック（IBUSクロック，66 MHz：33 MHz入力時）をカウントする32ビット・カウンタの値を示します。T1タイマ機能の受信スタート時間をタイム・スタンプの開始基準として使用します。初期値は0000_0000Hで，カウント・アップはリセットの直後に開始します。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_TSR	R/W	0	ATMセル・プロセッサのシステム・クロック・カウント

4. 4. 20　A_IBBAR（IBUSベース・アドレス・レジスタ）

A_IBBARは，IBUSを介して外部にアクセスするベース・アドレスを指定します。RISCコアのメモリ空間は，24ビット・アドレスで番地づけされ，VR4120AのRISCプロセッサのメモリ空間は32ビット・アドレスで番地づけされます。したがって，このブロックの内部から外部へのアクセスが要求された場合，アドレスの拡張が必要です。そのため，VR4120Aコアから見たATMセル・プロセッサとの共有アドレスはA_IBBARのアドレスからA_IBBAR＋3F_FFFFHのアドレスまでとなります。初期値は0です。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_IBBAR	R/W	0	IBUSを介して外部にアクセスするベース・アドレス 下位24ビットは0に設定してください。

保守／廃止

4. 4. 21　A_INBAR（命令ベース・アドレス・レジスタ）

A_INBARは，命令を取り出すベース・アドレスを設定します。RISCコアのメモリ空間は，24ビット・アドレスで番地づけされ，VR4120AのRISCプロセッサのメモリ空間は32ビット・アドレスで番地づけされます。したがって，このブロックの内部から外部へのアクセスが要求された場合，アドレスの拡張が必要です。そのため，VR4120Aから見たATMセル・プロセッサとの共有アドレスはA_INBARで指定したアドレスからA_INBAR＋0F_FFFFHのアドレスまでとなります。初期値は0です。

電源投入およびシステム・リセット後には，ATMセル・プロセッサ内のRISCコアは停止しており，A_INBARレジスタを設定することにより動作を開始します。

A_INBARレジスタの設定は，電源投入およびシステム・リセット後に1回のみ行ってください。2回以上行うと，RISCコアがハングアップする可能性があります。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	A_INBAR	R/W	0	命令を取り出すベース・アドレス 下位24ビットは0に設定してください。

保守／廃止

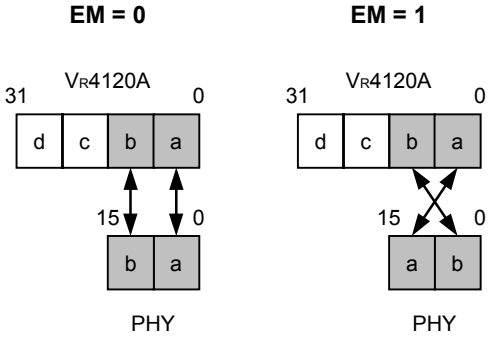
4. 4. 22　A_UMCMD（UTOPIAマネジメント・インタフェース・コマンド・レジスタ）

A_UMCMDは，UTOPIAマネジメント・インタフェースのオペレーション・モードを選択します。リセット後，VR4120AコアはUTOPIAマネジメント・インタフェースのコンフィギュレーションをこのレジスタに設定しなければなりません。

BMビットに0をセットすると，8ビット・モードとUMD［7：0］端子が有効であることを意味します。

BMビットに1をセットすると，16ビット・モードであることを意味します。この場合，ハーフ・ワード・アラインされたアクセスだけが受け付けられます。

EMビットは，16ビット転送モードのときのみ，1にセット可能です。EMビットは，以下に示すようにデータの配置を変えることができます。



A_UMCMDの初期値は0です。

ビット	フィールド	R/W	ディフォールト	説　明
31	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
30	BM	R/W	0	0 = 8ビット転送モード 1 = 16ビット転送モード
29	EM	R/W	0	0 = データをストレートで扱う 1 = データをクロスして扱う
28：3	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
2	PR	R/W	0	0 = UMRST_Bをディアサートする 1 = UMRST_Bをアサートして外部に接続されたPHYデバイスをリセットする
1：0	MSL	R/W	00	00 = UTOPIAマネジメントはモトローラ互換モードで動作する（DS, R/W, DTACKスタイル） 01 = UTOPIAマネジメントはインテル互換モードで動作する（RD, WR, RDYスタイル） 1x = 設定禁止

保守／廃止

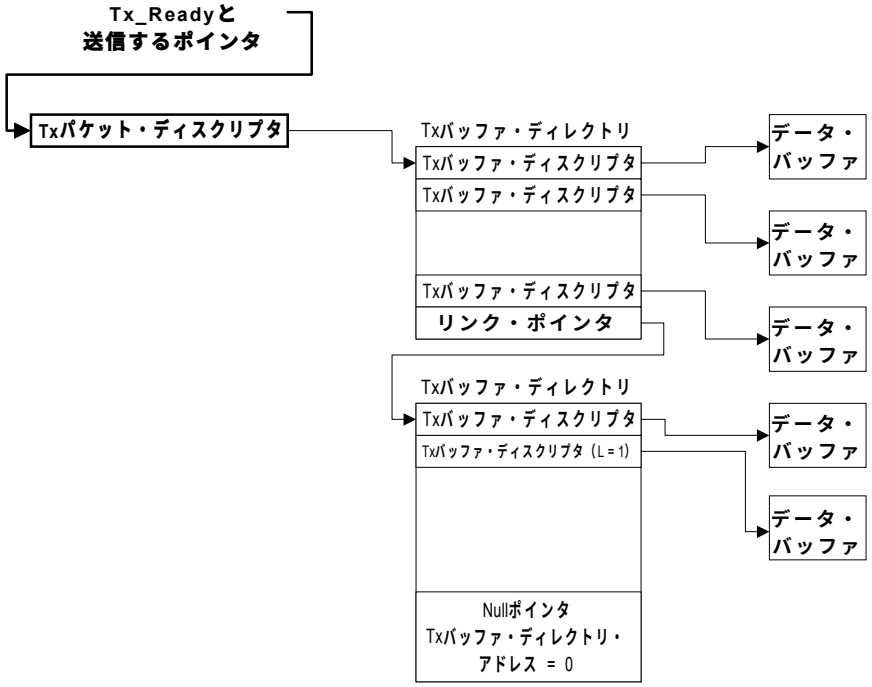
4.5　データ構造

ATMセル・プロセッサは，イーサネット・コントローラやUSBコントローラと同様のTx/Rxバッファ構造となっています。

4.5.1　Txバッファ構造

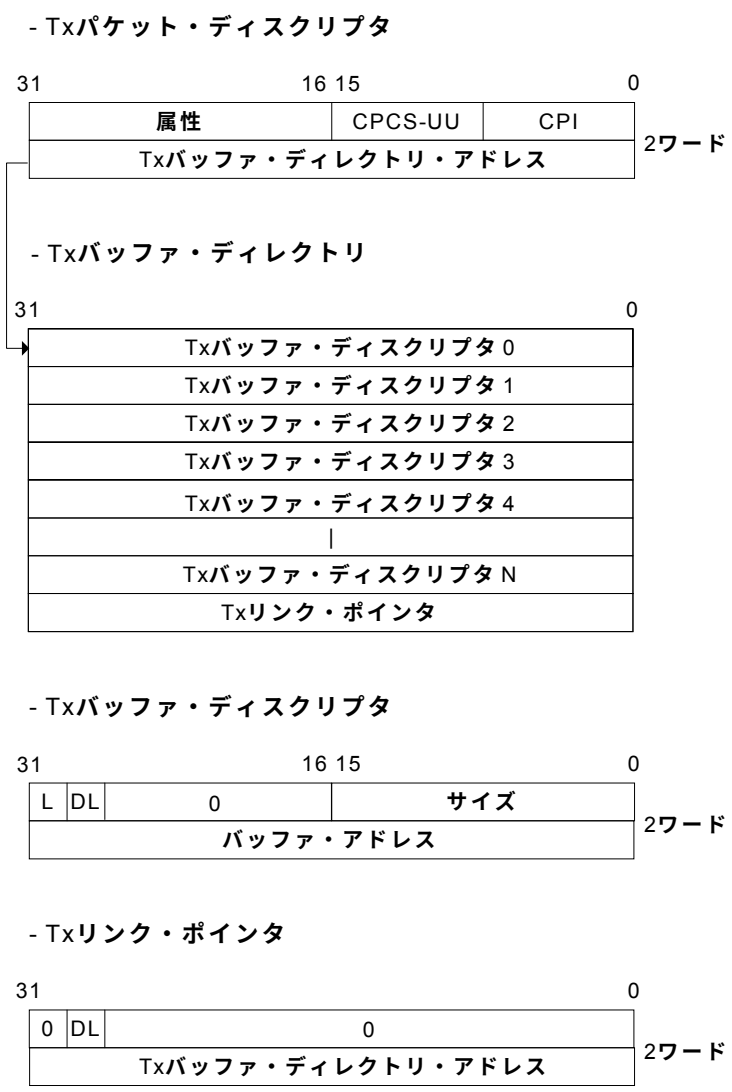
ATMセル・プロセッサが使用するTxバッファ構造を以下の図に示します。パケット・ディスクリプタと，いくつかのバッファ・ディレクトリ，データ・バッファで構成されています。Rxバッファ構造とTxバッファ構造は同じなので，受信パケットを送信するときにバッファ構造を再構成する必要はありません。

図4－8　Txパケット



保守／廃止

図4－9　Txバッファの要素



Txバッファの要素を図4－9に示します。各要素は，シーケンシャル・アドレスの32ビット・ワード・データの組み合わせで構成されています。詳細については以下のセクションに記述します。

保守／廃止

4. 5. 1. 1　パケット・ディスクリプタ

パケット・ディスクリプタは，図4－10に示す2ワードで構成されています。そのアドレスは，ワード配列されています。

図4－10　Txパケット・ディスクリプタ

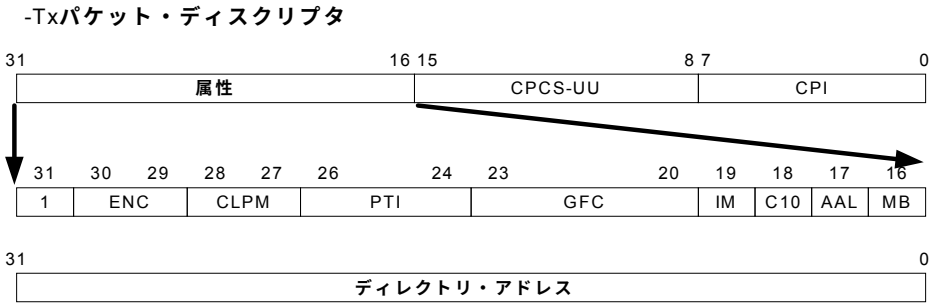


表4－1は，Txパケット属性の一覧です。詳細は，4. 8　オペレーションに記述します。

表4－1　Txパケット属性一覧

フィールド	説　明
ENC	カプセル・モードを指定するためのビットを指定します。
CLPM	Txセル・ヘッダに設定するCLPビットを指定します。
PTI	Txセル・ヘッダに設定するPTIビットを指定します。
GFC	Txセル・ヘッダに設定するGFCビットを指定します。
IM	送信が完了したときの割り込みと表示をディスエーブルにします。
C10	CRC10コードの作成と挿入をイネーブル（1）／ディスエーブル（0）に指定します。
AAL	AALのタイプを指定します。
MB	メールボックス・ナンバを表示します。
CPCS-UU	Txセル・トレイラに設定するCPCS-UUビットを指定します。
CPI	Txセル・トレイラに設定するCPIビットを指定します。

4. 5. 1. 2　Txバッファ・ディレクトリ

Txバッファ・ディレクトリは，複数のバッファ・ディスクリプタ（最大255個）と1つのリンク・ポインタで構成されています。そのアドレスは，ワード配列されています。バッファ・ディレクトリの末尾は，必ずリンク・ポインタです。バッファ・ディスクリプタは，シーケンシャルに先頭から読み取り，使用します。

保守／廃止

4. 5. 1. 3　Txバッファ・ディスクリプタ

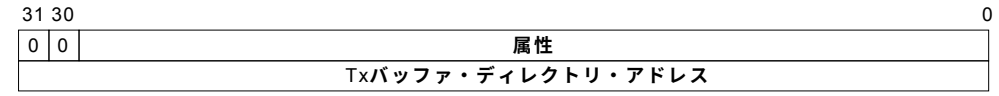
Txバッファ・ディスクリプタとTxリンク・ポインタは両方とも2ワードで構成されます。最初のワードのビット30のDLビットは，これらの2ワードがバッファ・ディスクリプタ（DL = 1）またはリンク・ポインタ（DL = 0）であることを示します。Txバッファ・ディスクリプタでは，ビット31のLビットは，このディスクリプタが示すバッファがパケットの最後の部分を含んでいることを示します。

Txリンク・ポインタを図4－11に示します。ビット31であるLビットは，0に固定されています。リンクするバッファ・ディレクトリがない場合は，リンク・ポインタのディレクトリ・アドレスは，Nullポインタとして0000Hを設定しなければなりません。

図4－11　Txバッファ・ディスクリプタ／リンク・ポインタ

-Txバッファ・ディスクリプタ

-Txリンク・ポインタ



4. 5. 1. 4　データ・バッファ

データ・バッファは，送信する実際のパケット・データが存在するスペースです。バッファのサイズは，1バイトから64Kバイトまでです。そのアドレスは，バイト配列されます。

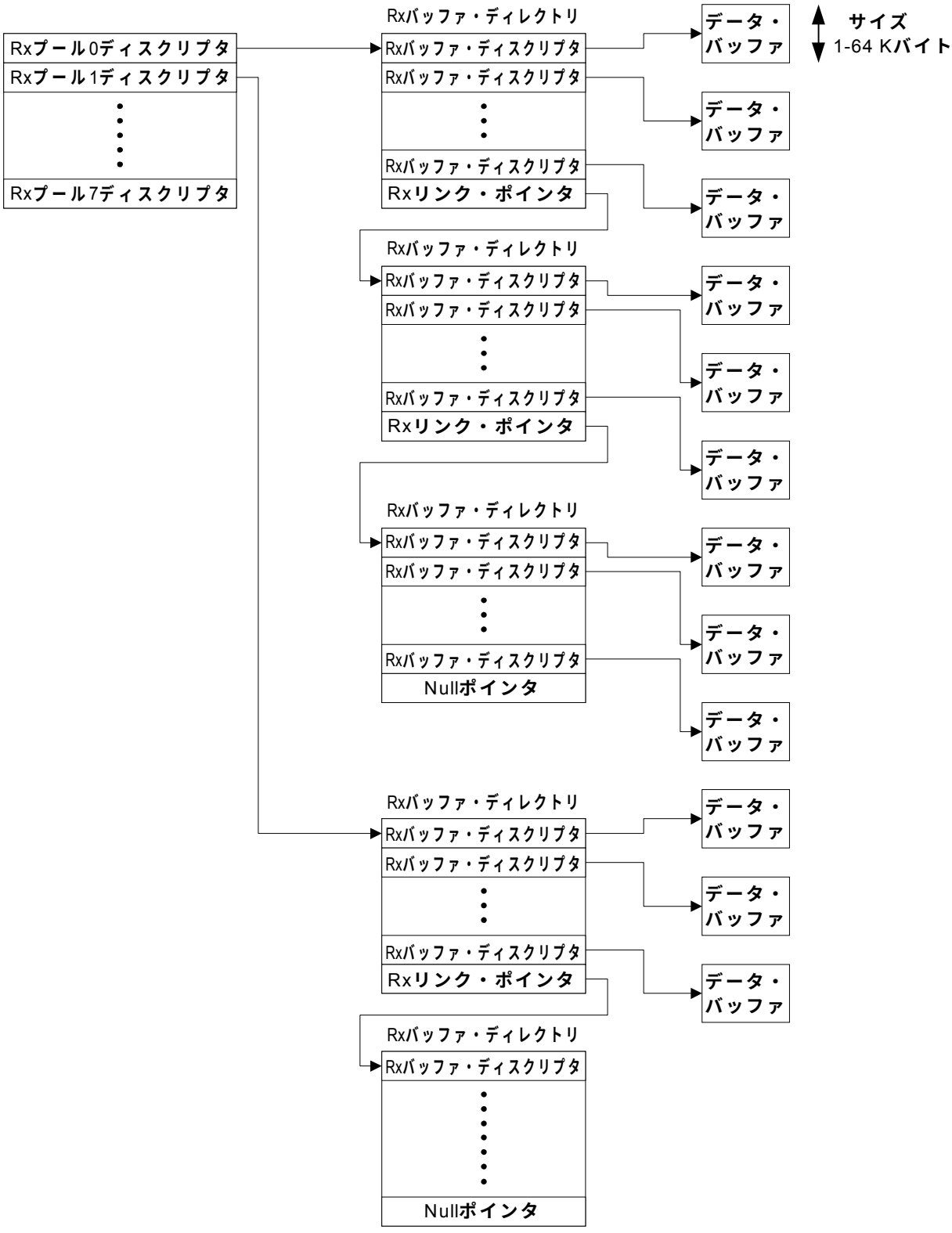
4. 5. 2　Rxプール構造

Rxバッファ構造は，プールとして定義されます。8つのプールをサポートします。プールは，Rxバッファ・ディレクトリのチェーンから構成されます。各Rxバッファ・ディレクトリには，複数のバッファ・ディスクリプタと1つのリンク・ポインタがあります。各バッファ・ディスクリプタは，受信セル・データを格納した受信バッファのアドレスを示します。リンク・ポインタは，次のRxバッファ・ディレクトリに対するアドレスを示します。

VR4120Aは，プールを最大8つまで作成してATMセル・プロセッサに与えます。

保守／廃止

図4－12　Rxプール構造



保守／廃止

図4－13　Rxプール・ディスクリプタ/Rxバッファ・ディレクトリ/Rxバッファ・ディスクリプタ/Rxリンク・ポインタ

Rxバッファの要素を図4－13に示します。各要素は，シーケンシャル・アドレスの32ビット・ワード・データの組み合わせで構成されます。

詳細については以下のセクションに記述します。

保守／廃止

4. 5. 2. 1　Rxプール・ディスクリプタ

プール・ディスクリプタは，図4－14に示す2つのワードで構成されます。そのアドレスは，ワード配列されています。

図4－14　Rxプール・ディスクリプタ

-Rxプール・ディスクリプタ

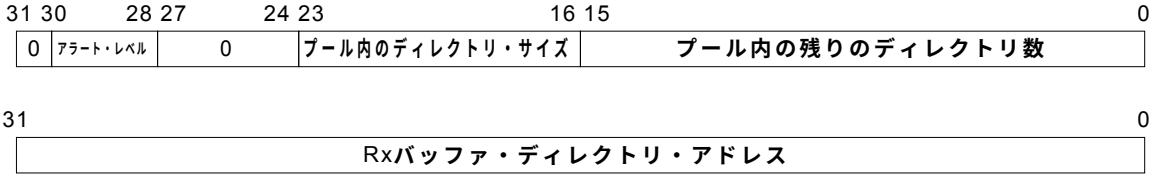


表4－2　Rxプール属性一覧

フィールド	説　明
アラート・レベル	バッファ・ディレクトリの残り数がこの数に4を乗算した数よりも小さい場合は，受信キュー枯渇警告（A_GSRのRQA = 1）割り込みが発行されます。
プール内のディレクトリ・サイズ	プール内のバッファ・ディレクトリにあるRxバッファ・ディスクリプタの数を表示します。
プール内の残りのディレクトリ数	バッファ・ディレクトリの現在の残り数を表示します。
ディレクトリ・アドレス	プール内の最初のバッファ・ディレクトリのアドレスを表示します。

4. 5. 2. 2　Rxバッファ・ディレクトリ

Rxバッファ・ディレクトリは，複数のバッファ・ディスクリプタ（最大255個）と1つのリンク・ポインタで構成されます。1つのプール内の各ディレクトリのバッファ・ディスクリプタの数は同じにしてください。プールごとに設定を変更することができます。

バッファ・ディレクトリのアドレスは，ワード配列されています。バッファ・ディレクトリの末尾は，必ずリンク・ポインタです。バッファ・ディスクリプタは，シーケンシャルに先頭から読み取り，使用します。

4. 5. 2. 3　Rxバッファ・ディスクリプタ

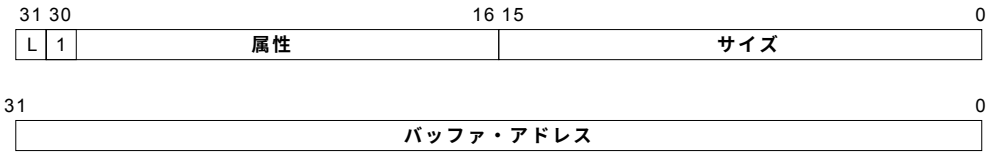
Rxバッファ・ディスクリプタとRxリンク・ポインタは両方とも2ワードで構成されます。最初のワードのビット30のDLビットは，これらの2ワードがバッファ・ディスクリプタ（DL = 1）またはリンク・ポインタ（DL = 0）であることを示します。

Rxバッファ・ディスクリプタとRxリンク・ポインタを図4－16に示します。そのアドレスは，ワード配列されています。Lビットは，このディスクリプタが示すバッファがパケットの最後の部分を含んでいることを示します。

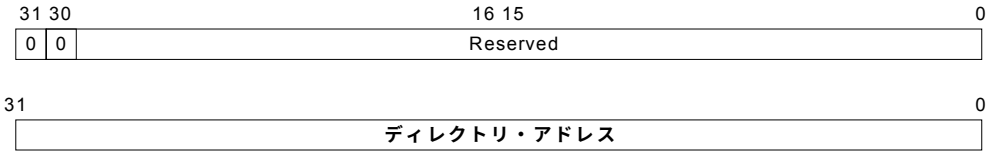
リンクするバッファ・ディレクトリがない場合は，リンク・ポインタのディレクトリ・アドレスはNullポインタとして0000Hを設定しなければなりません。

図4－15　Rxバッファ・ディスクリプタ／リンク・ポインタ

-Rxバッファ・ディスクリプタ



-Rxリンク・ポインタ



4. 5. 2. 4　Rxデータ・バッファ

Rxデータ・バッファは，実際の受信セル・データを格納します。バッファのサイズは，1バイトから64 K バイトまでです。そのアドレスは，バイト配列されます。

保守／廃止

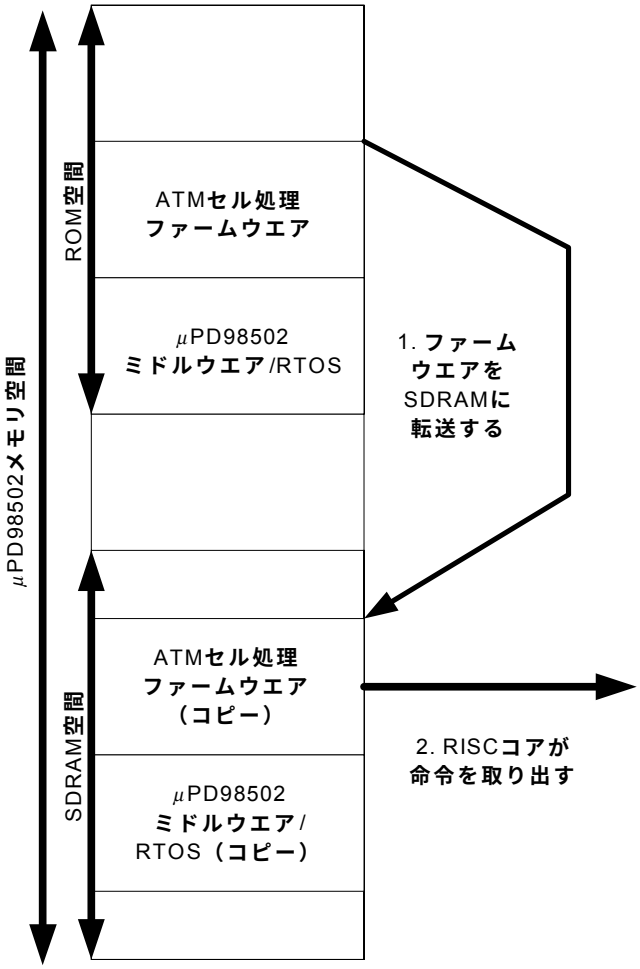
4.6　初 期 化

このATMセル・プロセッサは，当社からお客様に提供するファームウエアによって初期化します。

4.6.1　RISCコアの動作を開始する前に

RISCコアには，1 Mバイトの命令空間，それと内部に8 Kバイトの物理命令RAM，8 Kバイトの命令キャッシュがあります。命令空間は，外部SDRAMのシステム・メモリ空間にマップされます。VR4120Aから見た場合の命令空間のアドレスは，A_INBARの内容を加算することで生成されます。したがってVR4120Aは，初期化中にA_INBARを設定します。VR4120Aは，図4－16に示すように，初期化のときに自身のソフトウエアのほかにあらかじめシステムが割り当てたSDRAMのATMセル・プロセッサ用の命令空間にファームウエアを転送しなければなりません。ファームウエアを転送したあとにVR4120Aは，ファームウエアのベース・アドレスをA_INBARに設定します。同時に共有メモリ空間のベース・アドレスもA_IBBARに設定することが必要な場合があります（現在リリース中のファームウエアでは，共有メモリ空間は使用していないため，設定は必要ありません）。その後，VR4120Aは，ATMセル・プロセッサの動作開始を指示（S_WRCRのATMWRを0にしてリセット解除）します。

図4－16　ファームウエアの転送



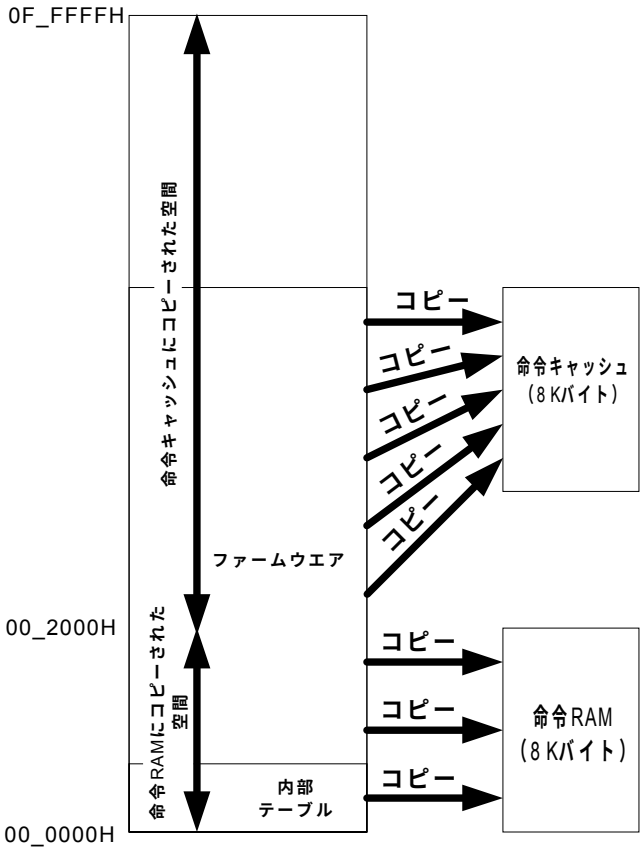
保守／廃止

4. 6. 2　RISCコアのファームウエアを開始したあとで

RISCコアは，アドレスxx00_0000Hから動作を開始します。アドレスxx00_0000Hにある命令を取り出し始めると，内部の特別に設けたハードウエアがRISCコアをいったん停止し，命令のブロックをコピーします。このコピー・オペレーションは，命令キャッシュについても同じ方法で行われます。

RISCコアの命令空間の下位8 Kバイトは，割り込みベクタ・テーブルを含むため，命令RAM上にコピーされます。空間のほかの部分は，8 Kバイトの命令キャッシュを介してアクセスされます。ファームウエアのサイズの合計が16 Kバイトよりも小さい場合は，すべての必要な命令コードがすべてコピーされるためキャッシュ・ミスが発生せず，RISCコアは最大性能で動作します。

図4－17　命令RAMと命令キャッシュ



保守／廃止

4.7　コマンド

このセクションは，AAL-5の動作で使用される基本的なコマンドについて記述します。AAL-2，OAM，セル・スイッチング機能で使用されるほかのコマンドについては，μPD98502アプリケーション・ノート（作成予定）に記述します。

ATMセル・プロセッサは，VR4120Aに以下の基本的なコマンドを提供します。

表4－3　コマンド

コマンド	ブロックの動作
Set_Link_Rate	PHYのリンク・レートを設定します。
Open_Channel	VCテーブル領域をワークRAMに予約します。
Close_Channel	VCテーブル領域を解放します。
Tx_Ready	送信プロセスを開始します。
Add_Buffers	バッファ・ディレクトリを指定したプールに追加します。
Indirect_Access	VR4120AのRISCプロセッサがワークRAMにアクセスすることを許可します。

すべてのコマンドは，VR4120Aにより，コマンド・レジスタ（A_CMR）と，必要な場合はコマンド拡張レジスタ（A_CER）に書き込まれます。コマンド・レジスタには，ビジィ・フラグがあります。ATMセル・プロセッサは，一度に1つのコマンドしか実行できないため，コマンドを受け付けたときにビジィ・フラグをセットします。VR4120Aは，このビジィ・フラグが1である間はほかのコマンドを発行できません。コマンド・オペレーションが終了すると，ATMセル・プロセッサはビジィ・フラグを0にセットします。VR4120Aは，新しいコマンドを発行する前に，コマンド・レジスタのビジィ・フラグを読み込んでビジィ・フラグが0であるかどうかを確認する必要があります。

(1) ATMセル・プロセッサがコマンド実行結果を返すコマンド

ATMセル・プロセッサがOpen_Channel，Close_Channel，Open_IP_Channel，Close_IP_Channel，Tx_Ready，Add_Buffersコマンドを受け取ると，コマンド実行結果をコマンド・レジスタに書き込みます。VR4120AのRISCプロセッサは，これらのコマンドを発行したあとにコマンド実行結果を読み込まなければなりません。また，ビジィ・フラグが1である間は，ATMセル・プロセッサはコマンド処理を終了していないため，VR4120AのRISCプロセッサは，実行結果を読み込むために，コマンド・レジスタのビジィ・フラグが0になるまで待たなければなりません。

(2) コマンド拡張レジスタを使用するコマンド

Indirect_AccessコマンドとAdd_Buffersコマンド，Tx_Readyコマンドは，コマンド・レジスタとともにコマンド拡張レジスタもあわせて使用します。

VR4120Aがこれらのコマンドを書き込むとき，VR4120Aはまず最初にコマンド拡張レジスタから書き込んでください。その次に，コマンド・レジスタに書き込んでください。ATMセル・プロセッサは，コマンド・レジスタに書き込まれるとコマンドの実行を開始します。したがって，コマンド拡張レジスタに先に書き込んでおかないと，コマンド拡張レジスタの情報はATMセル・プロセッサに無視されます。

ATMセル・プロセッサからコマンド実行結果を得るためにコマンド拡張レジスタを使用するとき，VR4120Aはコマンド・レジスタのビジィ・フラグが0になるのを待って，コマンド拡張レジスタを読み込んでください。

保守／廃止

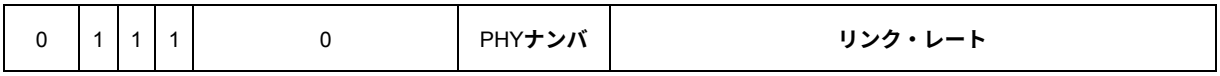
4. 7. 1　Set_Link_Rateコマンド

このコマンドは，ATMのPHYインタフェースのリンク・レートを設定するために使用します。ATMセル・プロセッサを初期化後，パケットを送信する前にこのコマンドを一度発行する必要があります。

図4－18　Set_Link_Rateコマンド

[Set_Link_Rateコマンド]

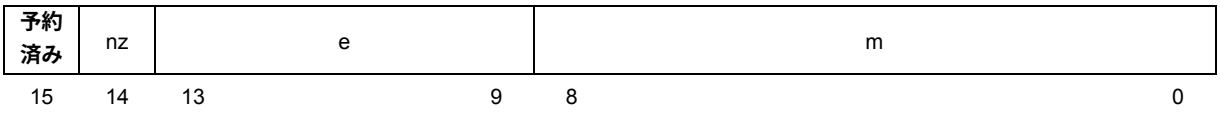
CMR



PHYナンバ　：UTOPIA2インタフェースに接続されたPHYの中の，どのアドレスのPHYに対してリンク・オープンするのかを指定します。

リンク・レート：実際のリンク・レートがこのフィールドに挿入されます。レートのフォーマットは次のとおりです。

2^e（1＋m/512）＊nzセル／秒



4. 7. 2　Open_Channelコマンド

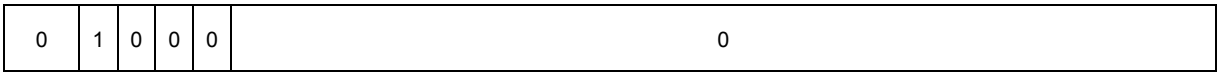
このコマンドは，送信または受信動作に使用する新しいチャネルをオープンするために使用します。チャネルをオープンすると，ATMセル・プロセッサはVCテーブル用の領域をワークRAMに予約し，VCナンバを実行結果として返します。

ATMセル・プロセッサがこのコマンドに返す結果は，以下のフォーマットになっています。

図4－19　Open_Channelコマンドと結果

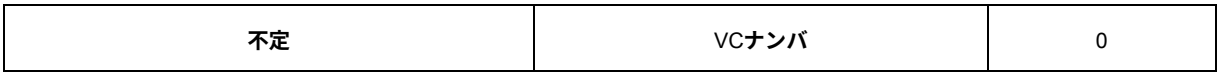
[Open_Channelコマンド]

CMR



[Open_Channelコマンド結果]

CMR



VCナンバ：オープンしたチャネルのVCナンバです。新しいVCがオープンできないときは， VCナンバの表示を0に設定します。

保守／廃止

4. 7. 3　Close_Channelコマンド

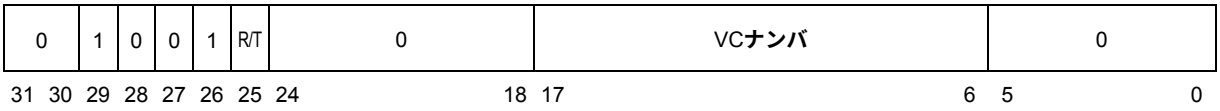
Close_Channelコマンドは，送信または受信チャネルをクローズするために使用します。このコマンドを受け付けると，ATMセル・プロセッサはVCテーブルをVCテーブル・プールに返します。

ATMセル・プロセッサがこのコマンドに返す実行結果は，以下のフォーマットになっています。

図4－20　Close_Channelコマンドと結果

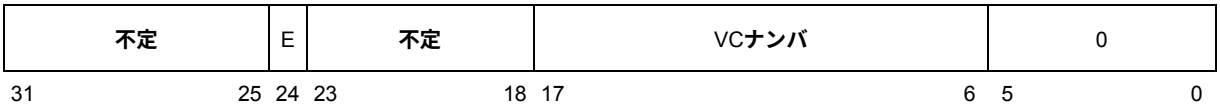
[Close_Channelコマンド]

CMR



[Close_Channelコマンド結果]

CMR



Close_Channelコマンド

R/T　　：クローズするチャネルが送信または受信チャネルのどちらであるかを設定します。
　　1：受信チャネル
　　0：送信チャネル

VCナンバ：VR4120Aがクローズしたいチャネルの VCナンバです。

Close_Channelコマンド結果

E　　：エラー・ビットです。エラー（たとえば，無効なVCナンバ）を検出すると，このビットを1に設定します。VR4120Aがこのコマンドを発行するときは，このビットが0となるように正しくコマンドを設定してください。

VCナンバ：クローズしたチャネルのVCナンバです。チャネルがクローズできない場合は，この領域に0が設定されます。

保守／廃止

4.7.4　Tx_Readyコマンド

Tx_Readyコマンドは，VR4120AがATMセル・プロセッサに対して，送信パケットを指定したチャネルに追加する（新しいパケット・ディスクリプタがシステム・メモリ・キューにセットされている）ことを通知するために使用します。このコマンドを受け取ると，ATMセル・プロセッサはスケジューリングを行うためにスケジューリング・テーブルをアクティブ状態にします。このコマンドでは，エラーを検出すると，EビットをA_CMRに書き込みます。

このコマンドのフォーマットは以下のようになります。

図4－21　Tx_Readyコマンドと結果

[Tx_Readyコマンド]

CMR

0	1	1	0	0	1	0	VCナンバ	0
31 30 29 28 27 26 25 24							18 17	6 5 0

CER

パケット・ディスクリプタ・アドレス
31 0

[Tx_Readyコマンド結果]

CMR

0	1	1	0	0	1	E	0	VCナンバ	0
31 30 29 28 27 26 25 24							23 18 17	6 5	0

Tx_Readyコマンド

VCナンバ　　：送信を開始させるチャネルのVCナンバです。

パケット・ディスクリプタ・アドレス　：パケット・ディスクリプタのアドレスです。

Tx_Readyコマンド結果

E　　：エラー・ビットです。エラー（たとえば，無効なVCナンバ）を検出すると，このビットを1に設定します。VR4120Aがこのコマンドを発行するときは，このビットが0となるように正しくコマンドを設定してください。

VCナンバ：送信を開始させるチャネルのVCナンバです。

保守／廃止

4.7.5　Add_Buffersコマンド

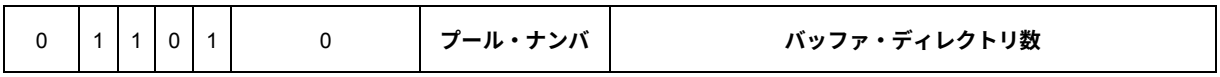
Add_Buffersコマンドは，未使用のバッファ・ディレクトリをプール・ナンバ・フィールドで指定した受信プールに追加するために使用します。

このコマンドでは，ATMセル・プロセッサがエラーを検出すると，EビットをA_CMRに書き込みます。このコマンドのフォーマットは以下のようになります。

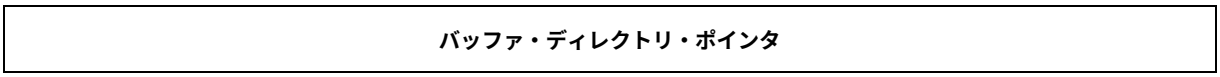
図4－22　Add_Buffersコマンドと結果

[Add_Buffersコマンド]

CMR

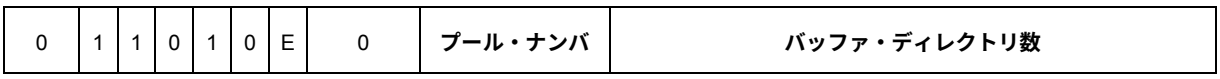


CER



[Add_Buffersコマンド結果]

CMR



Add_Buffersコマンド

プール・ナンバ　：バッファ・ディレクトリを追加したいプール・ナンバを指定します。ATMセル・プロセッサは8つのプール（0-7）しかサポートしないため，ビット19とビット20は0に設定してください。

バッファ・ディレクトリ数　：新しく追加するバッファ・ディレクトリの数です。

バッファ・ディレクトリ・ポインタ　：新しく追加するバッファ・ディレクトリのリスト内の最初のバッファ・ディスクリプタの先頭アドレスです。

Add_Buffersコマンド結果

E　：エラー・ビットです。エラーを検出すると，このビットを1に設定します。VR4120Aがこのコマンドを発行するときは，このビットが0となるように正しくコマンドを設定してください。このコマンドを実行後に起こり得るエラーには，バッファ・ディレクトリの数が65535を越えるというものがあります。

プール・ナンバ　：新しく追加されたバッファ・ディレクトリのプール・ナンバです。

バッファ・ディレクトリ数　：新しく追加されたバッファ・ディレクトリの数です。

保守／廃止

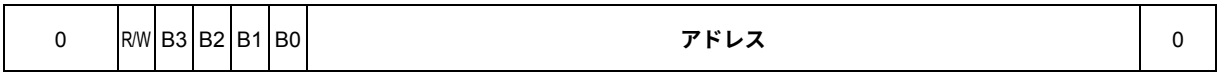
4. 7. 6　Indirect_Accessコマンド

Indirect_Accessコマンドは，VR4120Aからは直接見えないワークRAMにリード／ライト・アクセスをする際に使用します。

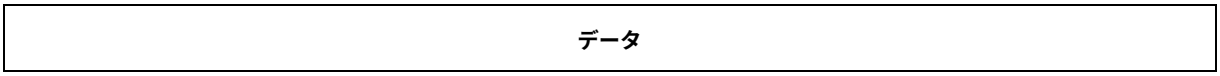
図4－23　Indirect_Accessコマンド

[Indirect_Accessコマンド]

CMR



CER



Indirect_Accessコマンド

R/W　　：ターゲットに対するアクセスがリードまたはライトのどちらであるかを指定します。
　　1：リード
　　0：ライト

B0, B1, B2, B3：ライト・アクセスでバイトを選択するために使用します。

アドレス　　：この領域で指定されたアドレスがワークRAMにあるときは，このアドレスはATMセル・プロセッサからアクセス可能なアドレスに変換されます。そうでなければ，アドレスはそのままで使用されます。アドレスの下位2ビットは“0”でなければなりません。すなわち，アドレスはワード・アラインでなければなりません。

保守／廃止

4. 8　オペレーション

このセクションは，動作の仕様（主にSAR機能）について記述します。

4. 8. 1　ワークRAMの使用

ワークRAMのサイズは，16 Kバイトです。このメモリは，以下の5つの目的に使用します。

(1) テンポラリ・データ

DMAを使用してSDRAMからのデータを一時的に記憶するためなどに使用するバッファです。この領域に一時的に格納されるデータは，送信／受信結果や，IPパケットのセル（送信／受信データ）などです。

(2) フロー・テーブル・プール

フロー・テーブルが格納される領域です。

(3) パケット情報プール

パケット情報が格納される領域です。

(4) フリー・ブロック・プール

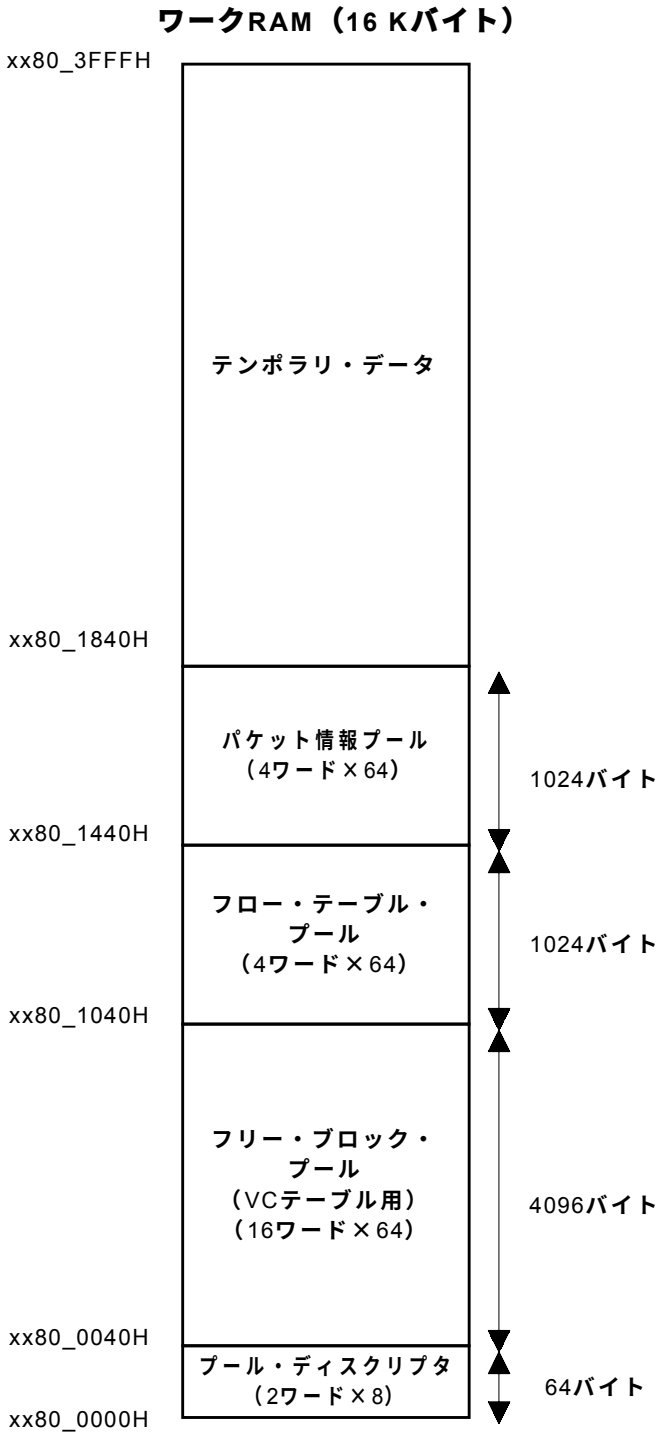
“プール・ディスクリプタ”が格納される領域です。各プール・ディスクリプタは，2ワードで構成されています。

(5) VCテーブル・プール

送信／受信VCテーブルが格納される領域です。各VCテーブルは，1つのブロック（16ワード）を使用します。

保守／廃止

図4－24　ワークRAMの使用



4. 8. 2　送信機能

VR4120Aは，パケット送信前ごとにVC情報をVCテーブルに設定し，VCに該当するパケットを送信するためにVCナンバを指定したTx_Readyコマンドを発行します。

送信データ構造についての詳細は，4. 5. 1　Txバッファ構造を参照してください。

保守／廃止

4. 8. 2. 1　送信手順

(a) 送信データを設定する

パケットを送信する前に，VR4120Aは送信するパケット・データをシステム・メモリに置き，パケット・ディスクリプタを設定します。

(b) 送信チャネルをオープンする

パケット・データを送信するためにVR4120Aが新しいチャネルを必要とする場合は，VR4120AはOpen_Channelコマンドを発行します。VR4120Aがこのコマンドを発行すると，ATMセル・プロセッサは新しいVCテーブル・プール・ブロックをワークRAMに割り当て，コマンド実行結果を報告してその先頭アドレスをVR4120Aに通知します。

(c) 送信VCテーブルを設定する

ワークRAMに割り当てた16ワードのVCテーブル・プール・ブロックを，各VCに送信VCテーブルとして設定します。

(d) Tx_Readyコマンドを発行する→最初のセルの送信の準備をする

VR4120AがTx_Readyコマンドを発行すると，ATMセル・プロセッサはパケット情報（各テーブル類）をワークRAMにセットし，パケット・ディスクリプタを取り出して各エリアに格納します。ATMセル・プロセッサは，送信待ちのパケットがある場合，送信キューを確認します。送信キューが空きでない場合は，ATMセル・プロセッサはパケット情報をキューの最後に追加します。キューに送信待ちのパケットがない場合は，ATMセル・プロセッサは，次の送信時間を“現在の時間＋1セル”でスケジュールします。

(e) セルを送信する

＜1＞ヘッダを作成する

ヘッダ情報としてVCテーブルのWord1から生成して，SAR FIFOに書き込みます。ヘッダのGFCフィールドに“00H”を挿入します。

＜2＞セグメント・データをシステム・メモリからSAR_FIFOに転送する

ATMセル・プロセッサは，送信セグメント（セルの48バイト・ペイロード・データ）をシステム・メモリから読み取り，スキャッタ／ギャザDMAを使用してSAR_FIFOに設定します。セグメントの先頭アドレスは，VCテーブル内の“Buffer Read Address”フィールドで表示されます。セグメントの53番目のバイトが書き込まれると，SAR FIFOは更新されます。

＜3＞CRC-32の値と長さを計算する

セグメントがSDRAMから取り出されるたびに，対象セグメントのCRC-32の値が計算され，送信バイト数もカウントされます。ATMセル・プロセッサは，これらの結果をVCテーブルに書き込みます。

＜4＞VCテーブルを更新する

“Buffer Read Address”フィールドと“Remaining Bytes in Current Buffer”フィールドを更新します。

(f) 末尾のセルを送信する

送信バッファのLフラグが最後のバッファであることを示し，VCテーブルに残っているバイト数を表示するフィールドが40バイト未満であることを示しているときは，そのセルはパケットの末尾のセルです。

＜1＞現在のセルがパケットの末尾のセルで，残りのペイロード・データが40バイト未満の場合は，ゼロ・パディングと8バイト・トレイラが追加されます。残りのペイロード・データが40バイト以上で8バイトのAAL-5トレイラを追加するための十分なスペースがない場合は，ATMセル・プロセッサはゼロ・パディングを追加して48バイトのペイロードを作成するだけで，トレイラとパディングだけを含むセルを次に送信します。

＜2＞AAL-5 PDUの最後のセグメントが読み込まれると，最終的なCRC-32の値とパケット長がAAL-5 PDUのトレイラに挿入され，VCテーブル内の最初のワードの内容がCPCS-UUフィールドとCPIフィールドに挿入されます。このようにして，AAL-5トレイラは完成します。

(g) ATMセル・プロセッサは，後続のパケットがあるかどうかを確認します（Tx VCテーブル内の最後のパケット情報アドレスと最初のパケット情報アドレスを確認します）。後続のパケットがある場合は，(e) と (f) を繰り返します。

(h) 各パケットに対して，ATMセル・プロセッサは，送信結果報告をステータス情報としてメールボックスに格納し，割り込みを発生します。

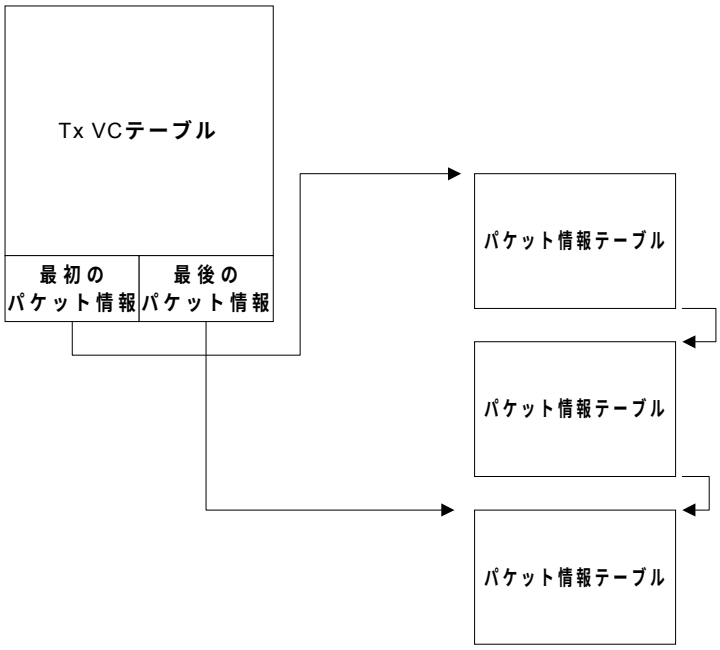
(i) VR4120Aは，メールボックスを読み込み，メールボックスのリード・ポインタを更新します。

保守／廃止

4. 8. 2. 2　送信キュー

Tx_Readyコマンドは，パケットを送信するために発行する必要があります。しかし，VR4120Aは，同じVCに対して次のTx_Readyコマンドを発行する前に送信結果報告を待つ必要はありません。前のパケットに対する送信プロセスを完了する前にVR4120AがTx_Readyコマンドを発行すると，ATMセル・プロセッサはそのVCに対してTxキューを構築します。

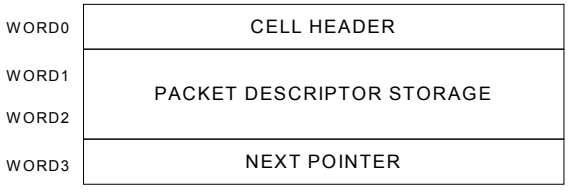
図4－25　送信キューの構造



(1) パケット情報テーブル

各VCごとのパケット情報テーブルの最大数は16です。しかし，すべてのVCに対するパケット情報テーブルの合計数は128のため，ほかのVCのキューの長さによっては，あるVCは16個のパケット情報テーブルを設定できないこともあります。ATMセル・プロセッサがパケット情報テーブルを得られない場合は，ATMセル・プロセッサはTx_Readyコマンドに対して送信結果報告でエラーを返します。

図4－26　パケット情報構造



CELL HEADER　　：セル・ヘッダ

PACKET DESCRIPTOR STORAGE：パケットのパケット・ディスクリプタを一時的に格納するための領域です。

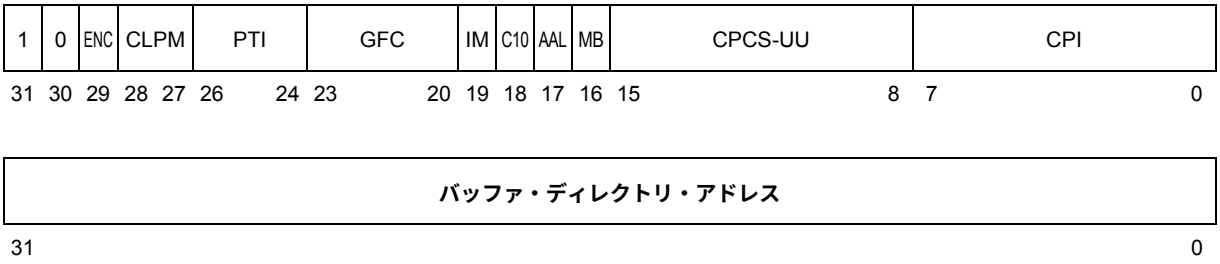
NEXT POINTER　　：次のパケット情報テーブルのアドレスです。

保守／廃止

(2) パケット・ディスクリプタ

図4－27はPACKET DESCRIPTOR STORAGEの内容を示します。

図4－27　送信キュー・パケット・ディスクリプタ



ENC　：カプセリング・モードを示します。
- 1　：LLCカプセリング
- 0　：カプセリングなし

CLPM　：パケット・ヘッダ内のCLPビットを示します。
- 00　：すべてのセル内のCLPビットを“0”として設定します。
- 11　：すべてのセル内のCLPビットを“1”として設定します。
- 01　：末尾のセルを除くすべてのセル内のCLPビットを“1”として設定し，末尾のセルのCLPビットのみを“0”として設定します。
- 10　：未使用。

PTI　：パケット・ヘッダ内のPTIフィールドです。このフィールドは，セルのペイロード・タイプを示すため，このフィールドに従って様々な送信手順を取ります。
- 000　：輻輳なしのユーザ・データ・セルです。
- 001　：輻輳なしのユーザ・データ・セルです。パケット内の末尾のセルです。ユーザがセットすることは，禁止されています。この値がセットされると，000として扱われます。
- 010　：輻輳付きのユーザ・データ・セルです。
- 011　：輻輳付きのユーザ・データ・セルです。パケット内の末尾のセルです。ユーザがセットすることは，禁止されています。この値がセットされると，010として扱われます。
- 100　：セグメント間で情報を交換するために使用されるOAM F5セルです。
- 101　：エンド・ツー・エンド間での情報を交換するために使用されるOAM F5セルです。
- 110　：ユーザがセットすることは，禁止されています。
- 111　：ユーザがセットすることは，禁止されています。

GFC　：パケット・ヘッダ内のGFCフィールドを示します。

IM　：ATMセル・プロセッサが送信結果報告をVR4120Aに発行するかどうかを示します。
- 1　：送信結果報告をVR4120Aに発行しません。
- 0　：送信結果報告をVR4120Aに発行します。

C10　：CRC-10を挿入するかどうかを示します。
- 1　：セルごとにCRC-10を挿入します。
- 0　：CRC-10を挿入しません。

AAL　　：送信パケットがAAL-5パケットであるかどうかを示します。

1　：AAL-5モードです。

0　：Rawセル・モードです。ATMセル・プロセッサは，このパケット内のセルをRawセルとして扱います。

MB　　：このパケットに対する送信結果報告を格納するためのメールボックス・ナンバを示します。

1　：メールボックス・ナンバ3

0　：メールボックス・ナンバ2

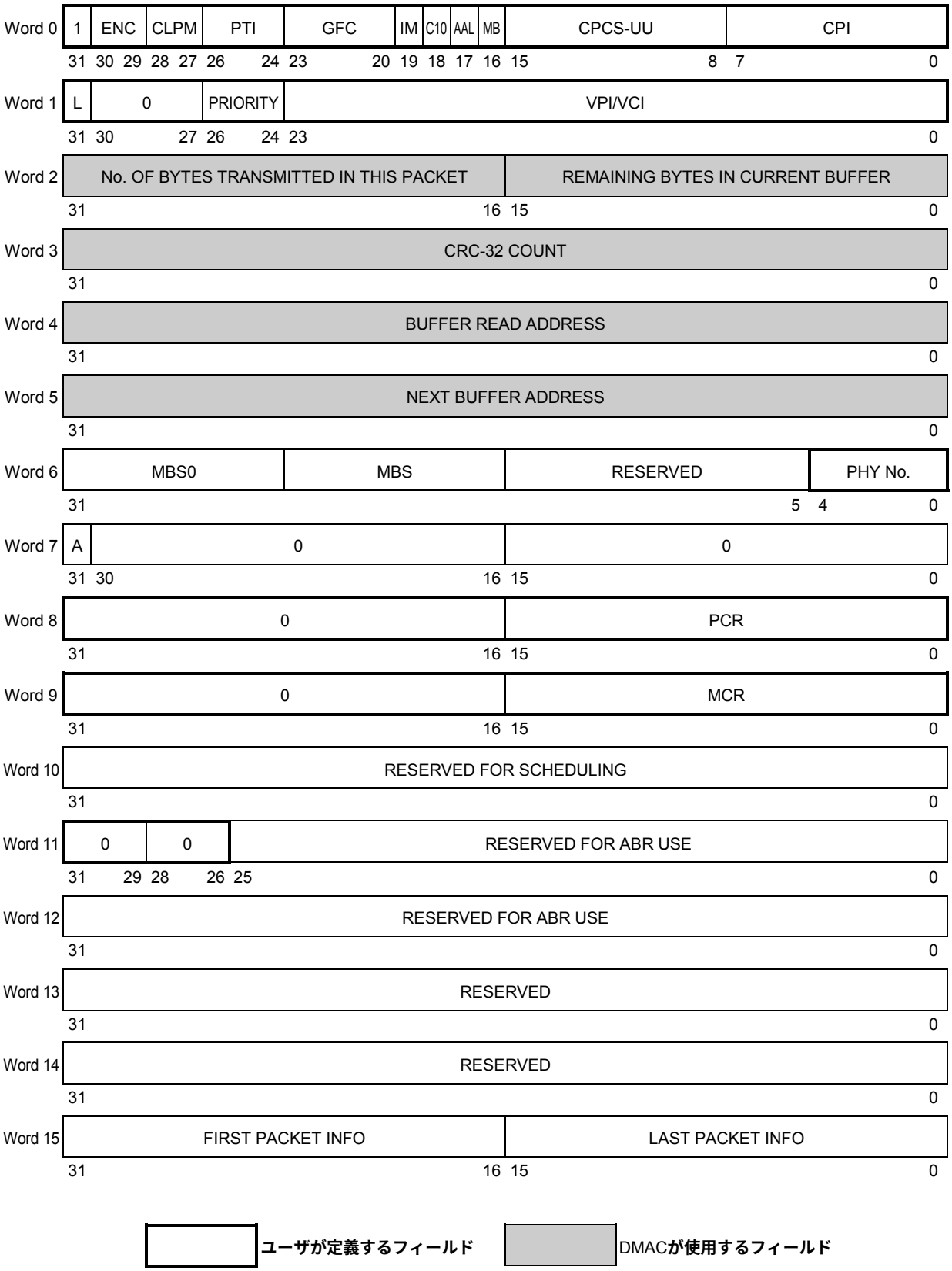
CPCS-UU：ユーザ情報です。AAL-5モードでは，ATMセル・プロセッサは，このフィールドのパターンをトレイラ内のCPCS-UUフィールドに挿入します。Rawセル・モードでは，このフィールドは無視されます。

CPI　　：ユーザ情報です。AAL-5モードでは，ATMセル・プロセッサは，このフィールドのパターンをトレイラ内のCPIフィールドに挿入します。Rawセル・モードでは，このフィールドは無視されます。

保守／廃止

(3) Tx VCテーブル

図4－28　Tx VCテーブル



Word0	：システム・メモリ内のパケット・ディスクリプタのWord0の内容と同じです。初期値はすべて0でなければなりません。ATMセルは，パケット・ディスクリプタ内のWord0をこのフィールドにコピーします。
L	：このビットは，SAR処理のために内部で使用します。初期値は，常に1でなければなりません。
PRIORITY	：送信優先順位を指定します。 111：CBR 110：VBR 010：UBR
VPI/VCI	：セル・ヘッダに含まれるVPI/VCI値です。
No. OF BYTES TRANSMITTED IN THIS PACKET	：これまで送信されたパケット内のバイト数です。初期値は0です。
REMAINING BYTES IN CURRENT BUFFER	：まだ送信されていない現在のバッファ内のバイト数です。初期値は0です。
CRC-32 COUNT	：このパケットの送信されたセルに対して計算されたCRC-32の値です。最終的なCRC-32の値は，トレイラ内のCRC-32フィールドにセットされます。
BUFFER READ ADDRESS	：次の送信時に転送する現在のバッファ内のバイトに対するポインタです。
NEXT BUFFER ADDRESS	：現在のパケット・ディレクトリ内の次のバッファに対するポインタです。
MBS0	：最大バースト・サイズです。PCR（ピーク・セル・レート）をもとに算出した，連続的に送信されるセルの数です。VBRモードでのみ使用します。
MBS	：MBS0をカウントするために内部で使用します。最初にMBS0と同じ値がセットされなければなりません。VBRモードでのみ使用します。
PHY No.	：このVCに対するセルの送信先PHYナンバです。
A	：VCテーブルがアクティブな状態にあるか，アイドルの状態にあるかを示す“アクティブ”ビットです。 1：アクティブ状態 0：アイドル状態
PCR	：ピーク・セル・レートです。
MCR	：最小セル・レートです。VBRモードでは，SCRをこのフィールドにセットしなければなりません。
Word 10	：このフィールドは，スケジューリングのために内部で使用します。
Word 11/Word 12	：これらのフィールドは内部で使用します。
FIRST PACKET INFO	：このVCに対する送信キュー内の最初のパケット情報テーブルの先頭アドレスです。初期値は0でなければなりません。
LAST PACKET INFO	：このVCに対する送信キュー内の最後のパケット情報テーブルの先頭アドレスです。初期値は0でなければなりません。

保守／廃止

4. 8. 2. 3　非AAL-5トラフィックのサポート

(1) OAM F5セル送信

ホストがパケット・ディスクリプタのPTIフィールドにOAM F5セル・パターン（100と101）を設定すると，ATMセル・プロセッサはAAL-5トレイラを追加しません。この場合，ホストがパケット・ディスクリプタの“サイズ”フィールドに48バイトを越えるバイト数を設定しても，ATMセル・プロセッサはデータ・バッファの上から48バイトを読み込むだけでその後のデータを無視します。ホストが“サイズ”フィールドに48バイト未満のバイト数を設定した場合，ATMセル・プロセッサはパディングを追加します。OAM F5のセル送信では，ホストは各OAM F5セルに異なるパケット・ディスクリプタを設定しなければなりません。

OAM F5セル送信では，ホストがパケット・ディスクリプタの“C10ビット”を1にセットすると，ATMセル・プロセッサはCRC-10を挿入します。ATMセル・プロセッサは，46バイトと6ビットに対してCRC-10を計算し，結果をペイロードの最後の10ビットに上書きします。

さらに，ATMセル・プロセッサに対するファームウエアは，順方向監視と逆方向報告機能をサポートします。

(2) Rawセル送信

ホストがOAM F5セル以外の非AAL-5トラフィック・パケットを送信すると，ホストはパケット・ディスクリプタのAALビットを0にし，PTIフィールドをユーザ・データを表す“0xx”に設定します。この場合，ATMセル・プロセッサは，AAL-5トレイラの演算，付加を行いません。

ホストがパケット・ディスクリプタのC10ビットを1に設定すると，ATMセル・プロセッサは，送信する各セルに対してCRC-10の演算，付加を行います。

図4－29　CRC-10付きRawセル

ヘッダ	ペイロード	CRC-10
5バイト	46バイトと6ビット	10ビット

CRC-10の生成多項式は次のとおりです。

$G(X) = 1 + X + X^4 + X^5 + X^9 + X^{10}$

4. 8. 2. 4　送信結果報告

各々の送信されたパケットに対して，ATMセル・プロセッサは，送信結果報告を送信完了状態としてメールボックスに書き込みます。送信に使用されるメールボックスは，メールボックス2とメールボックス3です。具体的には，ATMセル・プロセッサは，パケット内のすべてのデータを読み込むと，送信結果報告を書き込みます。したがって，送信結果報告の発行は，パケットのPMDレイヤへの送信が完了していることを意味するわけではありません。

送信結果報告をメールボックスに格納すると，ATMセル・プロセッサは，A_GSRレジスタの対応するMMビットを1に設定し，マスクされていなければ割り込みを発行します。

ATMセル・プロセッサが送信中にホストに発行する送信結果報告は，次のフォーマットになります。

図4－30　送信結果報告フォーマット

E	VCナンバ	A	パケット・キュー・ポインタ
31	30 ～ 16	15	14 ～ 0

E（エラーID）　：エラーIDはエラーの状態を示します。
　0：エラーあり
　1：エラーなし

VCナンバ　：このVCに対して使用されるVCナンバです。

A（アクティブ）　：0の場合は，パケット・ディスクリプタが無効なのでVCがアイドル状態に入ったことを示します。1の場合は，次のパケット・ディスクリプタが有効なのでVCをアクティブに保持していることを示します。

パケット・キュー・ポインタ：送信されたばかりのパケット・ディスクリプタの先頭アドレスの下位15ビットです。

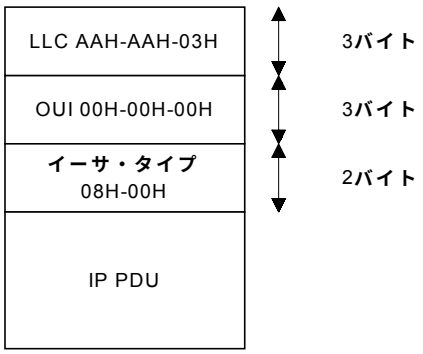
4. 8. 2. 5　スケジューリング

ATMセル・プロセッサは，ATMセル・プロセッサがすべてのアクティブ・チャネルの送信タイミングを設定しているスケジューリング・テーブルを保持しています。送信タイミングは，ATMセル・プロセッサがセルを送信するたびに再計算します。送信タイミングは，Tx_Readyコマンドの前にVR4120Aが設定するライン・レートとレート情報を使用して計算します。レート情報は，Tx VCテーブルに書き込みます。

4. 8. 2. 6　LLCカプセリング

LLCカプセリングがTx VCテーブルに設定されると，ATMセル・プロセッサは，LLCヘッダをIPパケットの先頭部に追加します。ATMセル・プロセッサは，CPCS-PDUをインターネットIP PDUとして常にカプセリングします。

図4－31　LLCカプセリング・フォーマット



保守／廃止

4.8.3　受信機能

受信データ構造についての詳細は，4.5.2　Rxプール構造を参照してください。

4.8.3.1　受信手順

(a) 受信プールを設定する

パケットを受信する前にVR4120Aは，受信セル・データを格納するために受信プールを準備し，ワークRAMのプール・ディスクリプタにプールに関する情報を設定します。

(b) 受信チャネルをオープンする

VR4120Aは新しい接続をオープンするためOpen_Channelコマンドを発行します。Open_Channelコマンドが発行されると，ATMセル・プロセッサは，VCテーブル・ブロックをワークRAM内に割り当て，コマンド実行結果を報告してその先頭アドレスをVR4120Aに通知します。VR4120Aは割り当てたブロックをVCテーブルとして設定します。

(c) 受信VCテーブルを設定する

VR4120Aは，ワークRAMに割り当てた16ワードのブロックを，各VCに受信VCテーブルとして設定します。

(d) Rxルックアップ・テーブルを設定する

受信セルのVPI/VCIをRxルックアップ・テーブルに設定します。

(e) パケットの先頭のセルを受信する

ATMセル・プロセッサがセルを受信すると，ATMセル・プロセッサは，受信セルのVPI/VCIがRxルックアップ・テーブルに登録されているかどうかを確認します。登録されていないと，セルは廃棄されます。登録されていると，ATMセル・プロセッサは，VCナンバをRxルックアップ・テーブルから取得します。それがパケットの先頭のセルである場合，ATMセル・プロセッサは，新しいバッファ・ディレクトリを割り当てます。VCはT1タイマ・リストに追加されます。

(f) セルを受信する

ATMセル・プロセッサは，スキャッタ／ギャザDMAを用いて，受信セル・データをSAR FIFOからシステム・メモリへ転送します。送信の場合と同様に，各受信セル・データに対して計算されたCRC-32の値は，VCテーブルのCRC-32フィールドに格納されます。その後，VCテーブルの“Current Count of Bytes”フィールドと“Remaining words in current buffer”フィールドを，更新します。

(g) 末尾のセルを受信する

＜1＞トレイラ情報のエラーをチェックします。

＜2＞受信結果報告をメールボックスに格納し，割り込みを発行します。

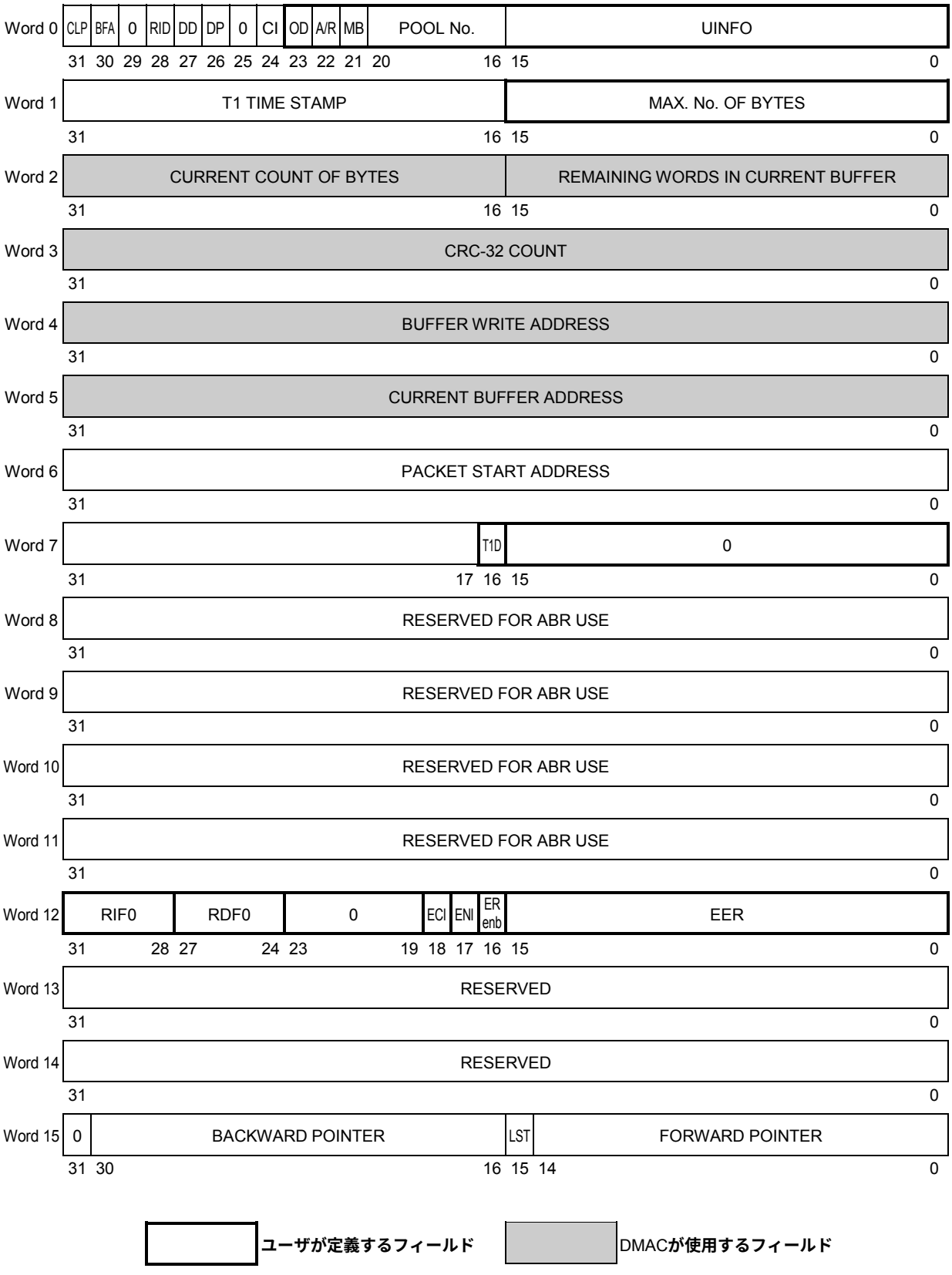
(h) 受信結果報告を読み取る

VR4120Aは受信結果報告を読み取り，メールボックスのリード・ポインタを更新し，受信データをプールから抽出します。

保守／廃止

（1）Rx VCテーブル

図4－32　受信VCテーブル



CLP	：受信中のパケットの少なくとも1つのセルのヘッダのCLPが1の場合は，1にセットします。
BFA	：このVCに割り当てられたフリー・バッファが存在する場合は，1にセットします。
RID	：パケット受信中にエラーが発生すると，1にセットします。エラー発生以降，末尾のセルを含むパケットの後続のセルを廃棄します。
DD	：フリー・バッファが割り当てられていない場合，セルを廃棄し，このフィールドを1にセットします。
DP	：ATMセル・プロセッサが使用していないため，常に0です。
MB	：メールボックス・ナンバです。 0 = メールボックス0 1 = メールボックス1
POOL No.	：プール・ナンバです（ATMセル・プロセッサは，プール・ナンバ0-7をサポートします）。
UINFO	：ユーザ情報です。
T1 TIMESTAMP	：T1タイマの計算に使用される領域です。
CI	：輻輳通知です。
OD	：OAMセルの受信／廃棄表示ビットです。
A/R	：AAL-5/Rawセル受信表示ビットです。 0 = Rawセル 1 = AAL5
MAX No. OF BYTES	：1つのパケット内の最大バイト数です。
REMAINING WORDS IN CURRENT BUFFER	：現在のバッファに残っているフリー領域のワード数です。
CURRENT COUNT OF BYTES	：パケットの受信開始から現在までに受信されたパケットのバイト数です。
CRC-32 COUNT	：CRC-32の値を計算するために使用します。
BUFFER WRITE ADDRESS	：次の格納に使用するために割り当てられたフリー・バッファのアドレスです。
CURRENT BUFFER POINTER	：次に割り当てられるフリー・バッファの開始アドレスです。
PACKET START ADDRESS	：パケットの開始アドレス（最初に割り当てられたバッファの開始アドレス）です。
WORD 6-WORD12	：これらのフィールドは内部で使用します。
BACKWARD POINTER	：T1リンク・リストの中でこのVCの前にリンクされたVCのVCナンバです。
LST	：T1リンク・リストの中でリンクされた最後のVCであれば1にセットします。
FORWARD POINTER	：T1リンク・リストの中でこのVCのあとにリンクされたVCのVCナンバです。

4. 8. 3. 2　非AAL-5トラフィック・サポート

ATMセル・プロセッサは，Rawセルを受信するたびに，53バイトのRawセルと11バイトのステータスを含むRawセル・データを生成し，適切なRxプールに格納します。その後，ATMセル・プロセッサは，A_GSRレジスタの対応するビットをセットし，マスクされていなければ割り込みを発行します。

ATMセル・プロセッサは，Rawセルをパケットではなくセル単位で扱うため，受信結果報告をRxメールボックスに設定しません。ATMセル・プロセッサがRawセルを受信するときは，CRC-10のベリファイ機能は常にイネーブルとなっています。ATMセル・プロセッサがCRC-10エラーを検出すると，Rawセル・データ内にエラー・ビットをセットします。

(1) OAM F5セル

VCテーブルのODビットが1で，受信されたATMセル・ヘッダのPTIフィールドが“1xx”であるときは，ATMセル・プロセッサは，Rawセル・データを生成し，プール0に格納します。ATMセル・プロセッサは，A_GSRレジスタにRCR0 = 1をセットし，マスクされていなければ，VR4120Aに割り込みを発行します。ATMセル・プロセッサは，これらのデータを常にプール0に格納します。ODビットが1にセットされている場合，プール0はRawセル・データ用にセットされなければなりません。

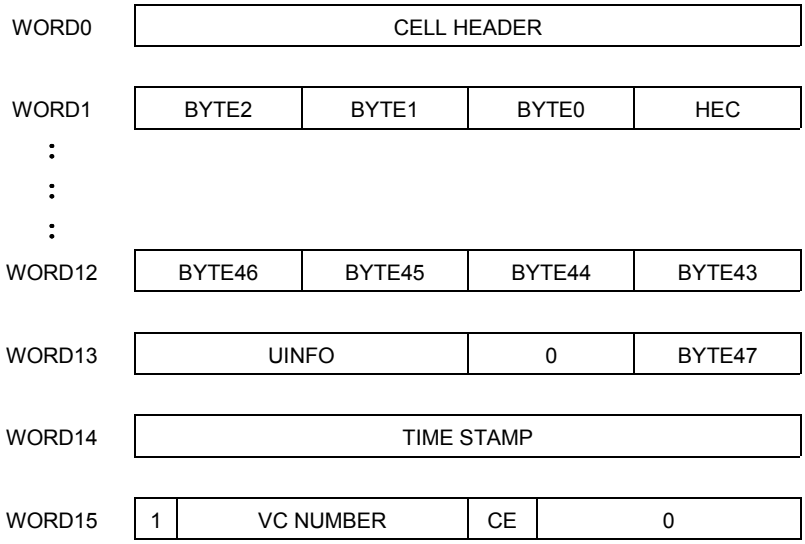
(2) 非AAL-5トラフィック

VCテーブルのA/Rビットが0の場合，ATMセル・プロセッサは，VCに属する受信セルをRawセルとして扱います。Rawセルを受信すると，ATMセル・プロセッサは，プールをRawセル・データに割り当てなければなりません。

(3) Rawセル・データ

リトル・エンディアン・モードのRawセル・データのフォーマットは次のとおりです。

図4－33　Rawセル・データ・フォーマット



CELL HEADER　：HECを除くセルのヘッダです。
HEC　：セルのHECフィールド・パターンです。
BYTE0-BYTE47　：セルのペイロード・データです。
UINFO　：ユーザ情報です。VCテーブルのUINFOフィールドにユーザがセットしたパターンです。
TIME STAMP　：ATMセル・プロセッサがセルを受信したときのA_TSRレジスタの値です。
VC NUMBER　：セルのVCナンバです。
CE　：CRC-10のチェック結果です。
0：エラーなし
1：CRC-10エラー検出

4. 8. 3. 3　受信結果報告

各パケットに対して，ATMセル・プロセッサは，受信結果報告を受信完了状態としてメールボックスに書き込みます。受信に使用されるメールボックスは，メールボックス0とメールボックス1です。具体的には，ATMセル・プロセッサは，以下の場合に受信結果報告を書き込みます。

(1) パケットに属するすべてのセル・データを受信した。
- エラーの検出なし
- CRC-32エラーまたは長さのエラーが検出された

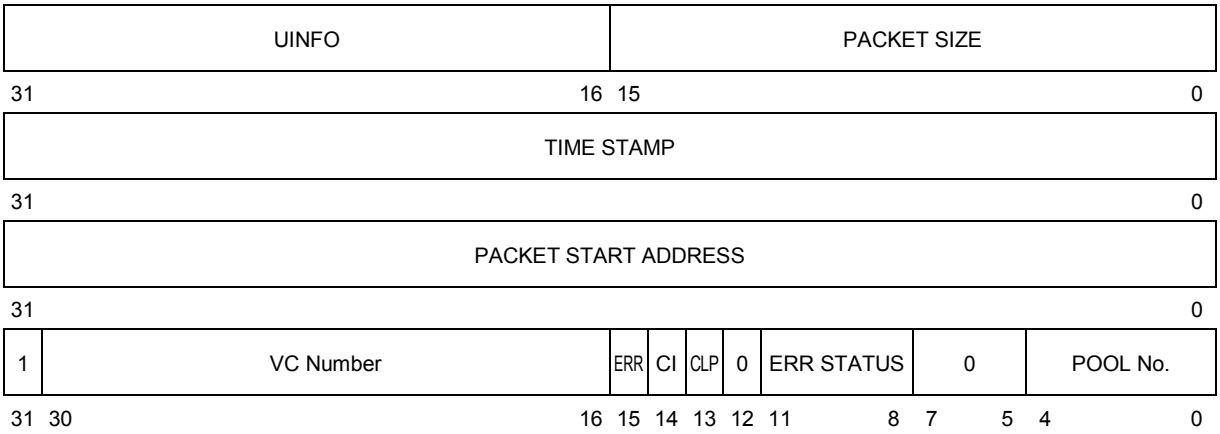
(2) パケットの末尾のセルが受信される前にエラーが検出された。エラーは，CRC-32エラーまたは長さエラー以外である。

受信結果報告は，ATMセル・プロセッサがセルを格納するために使用するバッファの開始アドレスとサイズおよびほかの情報を含んでいます。受信結果報告を読み込むことで，VR4120Aは，受信セルで構成される受信パケットを処理します。

受信結果報告のフォーマットは次のとおりです。

保守／廃止

図4－34　受信結果報告フォーマット



UINFO　：VCテーブルのUINFOフィールドにユーザがセットしたパターンです。

PACKET SIZE　：セル単位の受信パケットのサイズです。

TIME STAMP　：このステータスが発行された時点のA_TSRの値です。

PACKET START ADDRESS　：使用したバッファの開始アドレスです。

CHANNEL NUMBER　：このVCが使用するVCナンバです。

ERR　：1の場合は，パケットを受信中にエラーが発生したことを示します。0の場合は，パケットが正常に受信されたことを示します。

CI　：輻輳を示すPTIフィールドを持つセルがパケットの中に少なくとも1セル以上存在する場合に1をセットします。

CLP　：パケットの少なくとも1つのセルのヘッダ内のCLPフィールドが1であった場合に1をセットします。

ERROR STATUS　：発生したエラーの状態です。

- 0000：エラーなし
- 0001：フリー・バッファ・アンダフロー
- 0011：バイト違反の最大数
- 0100：CRC-32エラー
- 0110：長さエラー
- 0111：T1タイムアウト

POOL NUMBER　：使用したプールのナンバです。

保守／廃止

4. 8. 3. 4　受信エラー

ATMセル・プロセッサは，パケットの受信中とパケットの受信終了後にエラーをチェックします。エラーを検出すると，ATMセル・プロセッサは，メールボックスの受信結果報告を用いて，エラーの種類，開始アドレス，転送されたデータ長をシステム・メモリに報告します。

エラー・ステータスを含む受信結果報告を受け取ると，VR4120Aは，適切な処置をし，エラーを引き起こしたパケットを廃棄します。受信エラーの種類を以下に示します。

(1) フリー・バッファ・アンダフロー

このエラーは，フリー・バッファ内のフリー領域がパケットの受信時に48バイト以下のときに発生します。このエラーが発生すると，RQU割り込みが発生します。廃棄されたセルがパケットの中間セルまたは末尾のセルであるときは，パケットの受信は中断されます。フリー・バッファ・アンダフロー・エラーを通知する受信結果報告が発行され，VCテーブルのRIDビットがセットされます。末尾のセルを含むパケットの残りのセルが廃棄され，RIDビットが参照されます。末尾のセルが受信されたときに，RIDビットはリセットされます。このエラーのために廃棄されたセルがパケットの先頭のセルである場合は，RQU割り込みが発生し，RIDビットがセットされます。しかし，受信結果報告は発行されません。

(2) バイト違反の最大数

このエラーは，受信したセル数がユーザ指定の最大バイト数に達した時点でパケットの末尾のセルが受信されていないときに発生します。次のセルが受信されると，RIDビットがセットされ，受信結果報告が発行されます。末尾のセルを含むパケットの後続のセルは廃棄されます。

(3) CRC-32エラー

このエラーは，パケット内のセル・データがすべて受信された時点でCRC-32の値が受信トレイラ内のCRC-32の値と一致しないときに発生します。トレイラのチェックで受信したパケットにCRC-32エラーと長さエラーの両方が発生した場合は，CRC-32エラーをエラー・ステータスとして受信結果報告が発行されます。

(4) 長さエラー

このエラーは，受信トレイラ内の長さフィールドと計算されたパケット長が以下の条件のいずれかを満たす場合に報告されます。

（受信セル数×48バイト－トレイラ内の長さの値）＞55バイト

（受信セル数×48バイト－トレイラ内の長さの値）＜8バイト

(5) T1タイムアウト

このエラーは，先頭のセルが受信されたあとで，ユーザ指定のA_T1R時間が経過してもパケットの末尾のセルが受信されていないときに発生します。このエラーが発生すると，RIDビットがセットされ，末尾のセルを含むこのエラーを引き起こしたパケットの残りのセルが廃棄されます。

パケットの先頭のセル，中間のセル，末尾のセルで発生するエラーの一覧を以下の表に示します。

表4－4　パケット受信中に発生する受信エラー

エラー	セルの廃棄タイミング	受信結果報告の発行タイミング	エラー発生後のほかのセルの処理
フリー・バッファ・アンダフロー	フリー・バッファのサイズが不十分であるときに受信されたセルが廃棄されたとき	セグメント転送中に転送が不可能になるとき	末尾のセルを含むパケットの後続のセルが廃棄される
セグメント・エラーの最大数	セグメントの最大数に達したあとで受信されたセルが廃棄されたとき	セグメントの最大数に達したあとで次のセルが受信されたとき	末尾のセルを含むパケットの後続のセルが廃棄される
T1エラー	T1時間が経過したあとで受信されたセルが廃棄されたとき	T1時間が経過したとき	末尾のセルを含むパケットの後続のセルが廃棄される

2つ以上のエラーが同時に発生することもあります。しかし，そのうちの1つだけが受信結果として報告されます。エラー報告の優先順位の一覧を以下の表に示します。

表4－5　エラー報告の優先順位

優先順位	エラー
1	アンダフロー
2	MAXエラー
3	CRCエラー
4	長さ
5	T1エラー

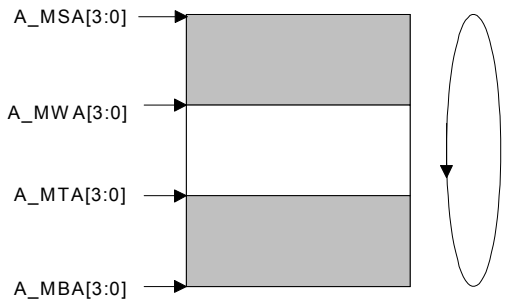
保守／廃止

4.8.4　メールボックス

ATMセル・プロセッサは，システム・メモリ内のメールボックスをリング・バッファとして使用します。メールボックスの構造と定義されたアドレスは，以下のようになります。

メールボックス・スタート・アドレス（A_MSA［3：0］）　：メールボックスのスタート・アドレス
メールボックス・ボトム・アドレス（A_MBA［3：0］）　：メールボックスのボトム・アドレス（メールボックスの最後のアドレスの次のアドレス）
メールボックス・ライト・アドレス（A_MWA［3：0］）　：ライト・ポインタ
メールボックス・テール・アドレス（A_MTA［3：0］）　：ホストが読み取り更新するテール・アドレス

図4－35　メールボックス構造



送信結果報告または受信結果報告を書き込むと，ライト・ポインタ（A_MWA［3：0］）をインクリメントし，A_GSRレジスタの対応するメールボックスの更新を通知するMMビットをセットし，マスクされていなければ割り込みを発行します。ライト・ポインタ（A_MWA［3：0］）を更新するとき，ATMセル・プロセッサは，A_MWA［3：0］がボトム・アドレス（A_MBA［3：0］）に達すると，A_MWA［3：0］をスタート・アドレス（A_MSA［3：0］）にジャンプさせます。送信結果報告または受信結果報告を読み取るために，VR4120Aは，リード・ポインタ（A_MTA［3：0］）を使用します。A_MTA［3：0］は，VR4120Aに管理されています。VR4120Aは，送信結果報告または受信結果報告をメールボックスから読み取るたびに，次の送信結果報告または受信結果報告のアドレスをリード・ポインタ（A_MTA［3：0］）に書き込みます。ライト・ポインタ（A_MWA［3：0］）がリード・ポインタ（A_MTA［3：0］）が示すアドレスと同じアドレスになった場合は，対応するA_GSRレジスタのMFビットをセットしてメールボックスがフル（MFステート）であることを示し，マスクされていなければ割り込みを発行します。メールボックスがMFステートに入ると，ATMセル・プロセッサは，コマンドを実行しません。

保守／廃止

第5章　イーサネット・コントローラ

5.1 概　　要

このセクションは，イーサネット・コントローラ・ブロックについて記述します。このイーサネット・コントローラ・ブロックは，10/100 MbpsイーサネットMAC（Media Access Control），データ送受信FIFO, DMA，内部バス・インタフェースからなります。

μPD98502は，2チャネルのイーサネット・コントローラを内蔵しています。

5.1.1 特　　徴

- IEEE802.3/802.3u/802.3x準拠：
  - 10/100 Mbpsイーサネット MAC
  - Media Independent Interface（MII）
- 10/100 Mbpsの全二重動作
- アドレス・フィルタリング：
  - ユニキャスト
  - マルチキャスト
  - ブロードキャスト
- 管理情報用の統計カウンタ：
  - RMON
  - SNMP MIB
- 大容量の独立した送受信FIFO
- プログラマブル・バースト・サイズ付きのダイレクト・メモリ・アクセス（DMA）
- 内部バス・インタフェース（IBUS）：32ビット@66 MHzに接続

保守／廃止

5. 1. 2　イーサネット・コントローラ・ブロックのブロック図

ブロックのハードウエア構成要素を以下に示します。このブロックのブロック図を図5－1に示します。

IBUSインタフェース：データ転送はこのIBUSインタフェースを介して，VR4120Aとイーサネット・コントローラ間またはメモリとイーサネット・コントローラ間で行います。その転送速度は約2 Gbps（32ビット×66 MHz）です。このブロックでは，IBUSプロトコル・オペレーション（リトライ，ディスコネクトなど）を行います。

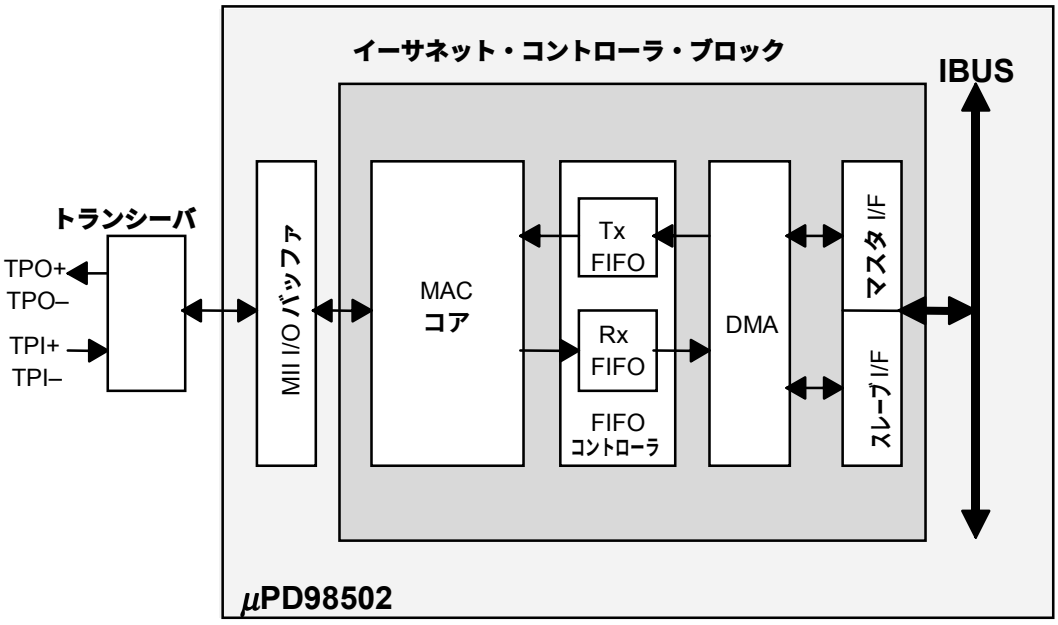
DMAコントローラ　：DMAコントローラには，メモリとFIFO間のデータ転送を処理する全二重送受信用DMAコントローラがあります。このDMAコントローラはバイト配列もサポートします。

FIFOコントローラ　：FIFOコントローラには，送受信用の2つの独立したFIFOがあります。このコントローラは，受信時に自動パケット廃棄（コリジョンまたはアドレス・フィルタリングのあと）および送信時にパケット再送信（コリジョンのあと）をサポートします。

MACコア　：MACコアはIEEE802.3, IEEE802.3u, IEEE802.3xと完全準拠しています。

MII I/Oバッファ　：MII I/OバッファはIEEE802.3uに準拠しており，標準の3.3 VのPHYデバイスに接続可能です。

図5－1　イーサネット・コントローラのブロック図



保守／廃止

5. 2　レジスタ

このブロックのレジスタは，表5－1に示すように4つのカテゴリに分類されます。

VR4120Aは以下のレジスタを制御します。

μPD98502には2チャネルのイーサネット・コントローラがあります。イーサネット・コントローラ#1のベース・アドレスは1000_2000Hで，イーサネット・コントローラ#2のベース・アドレスは1000_3000Hです。

表5－1　イーサネット・コントローラのレジスタのカテゴリ

アドレス	レジスタのカテゴリ
1000_2000H～1000_213FH（イーサネット・コントローラ#1） 1000_3000H～1000_313FH（イーサネット・コントローラ#2）	MAC制御レジスタ
1000_2140H～1000_21FFH（イーサネット・コントローラ#1） 1000_3140H～1000_31FFH（イーサネット・コントローラ#2）	統計カウンタ・レジスタ
1000_2200H～1000_2233H（イーサネット・コントローラ#1） 1000_3200H～1000_3233H（イーサネット・コントローラ#2）	DMA/FIFOマネジメント・レジスタ
1000_2234H～1000_223FH（イーサネット・コントローラ#1） 1000_3234H～1000_323FH（イーサネット・コントローラ#2）	割り込み／コンフィギュレーション・レジスタ

5. 2. 1　レジスタ・マップ

5. 2. 1. 1　MAC制御レジスタ

MAC制御レジスタのマップを表5－2に示します。

表5－2　MAC制御レジスタのマップ

アドレス	レジスタ名	R/W	アクセス	説　明
1000_m000H	En_MACC1	R/W	W	MACコンフィギュレーション・レジスタ1
1000_m004H	En_MACC2	R/W	W	MACコンフィギュレーション・レジスタ2
1000_m008H	En_IPGT	R/W	W	Back-to-Back IPGレジスタ
1000_m00CH	En_IPGR	R/W	W	Non Back-to-Back IPGレジスタ
1000_m010H	En_CLRT	R/W	W	コリジョン・レジスタ
1000_m014H	En_LMAX	R/W	W	最大パケット長レジスタ
1000_m018H： 1000_m050H	N/A	-	-	将来の使用のため予約
1000_m054H	En_LSA2	R/W	W	ステーション・アドレス・レジスタ2
1000_m058H	En_LSA1	R/W	W	ステーション・アドレス・レジスタ1
1000_m05CH	En_PTVR	R	W	ポーズ・タイマ値リード・レジスタ
1000_m060H	N/A	-	-	将来の使用のため予約
1000_m064H	En_VLTP	R/W	W	VLANタイプ・レジスタ
1000_m080H	En_MIIC	R/W	W	MIIコンフィギュレーション・レジスタ
1000_m084H： 1000_m090H	N/A	-	-	将来の使用のため予約
1000_m094H	En_MCMD	W	W	MIIコマンド・レジスタ
1000_m098H	En_MADR	R/W	W	MIIアドレス・レジスタ

保守／廃止

アドレス	レジスタ名	R/W	アクセス	説　　明
1000_m09CH	En_MWTD	R/W	W	MIIライト・データ・レジスタ
1000_m0A0H	En_MRDD	R	W	MIIリード・データ・レジスタ
1000_m0A4H	En_MIND	R	W	MIIインディケータ・レジスタ
1000_m0A8H： 1000_m0C4H	N/A	-	-	将来の使用のため予約
1000_m0C8H	En_AFR	R/W	W	アドレス・フィルタリング・レジスタ
1000_m0CCH	En_HT1	R/W	W	ハッシュ・テーブル・レジスタ1
1000_m0D0H	En_HT2	R/W	W	ハッシュ・テーブル・レジスタ2
1000_m0D4H： 1000_m0D8H	N/A	-	-	将来の使用のため予約
1000_m0DCH	En_CAR1	R/W	W	キャリー・レジスタ1
1000_m0E0H	En_CAR2	R/W	W	キャリー・レジスタ2
1000_m0E4H： 1000_m012CH	N/A	-	-	将来の使用のため予約
1000_m130H	En_CAM1	R/W	W	キャリー・マスク・レジスタ1
1000_m134H	En_CAM2	R/W	W	キャリー・マスク・レジスタ2
1000_m138H： 1000_m13CH	N/A	-	-	将来の使用のため予約

保守／廃止

備考1. **“ アドレス ” フィールドと “ レジスタ名 ” フィールド内でのmとnの値については次のとおりです。**
イーサネット・コントローラ#1：m = 2, n = 1，
イーサネット・コントローラ#2：m = 3, n = 2

2. **“** R/W **” フィールド内で，**
“ W **” は，“ 書き込み可能 ” を意味します。**
“ R **” は，“ 読み取り可能 ” を意味します。**
“ RC **” は，“ 読み取りクリア ” を意味します。**
“ - **” は，“ アクセス不可能 ” を意味します。**

3. **すべての内部レジスタは，**32**ビット・ワード配列のレジスタです。**

4. **内部レジスタに対するバースト・アクセスは，禁止されています。バースト・アクセスすると，**NSR**の**IRERR**ビットがセットされ，**NMI**が**VR4120A**に対してアサートします。**

5. **予約領域に対してリード・アクセスすると，**NSR**レジスタの**CBERR**ビットがセットされ，**SysCMD［0］**にデータ・エラー・ビットが設定されたダミー読み取り応答データが読み出されます（一部の非公開レジスタを除きます）。**

6. **予約領域に対してライト・アクセスすると，**NSR**レジスタの**CBERR**ビットがセットされ，書き込みデータが失われます。**

7. **“ アクセス ” フィールド内で，**
“ W **” は，ワード・アクセスが有効であることを意味します。**
“ H **” は，ハーフ・ワード・アクセスが有効であることを意味します。**
“ B **” は，バイト・アクセスが有効であることを意味します。**

8. **リード・オンリー・レジスタに対するライト・アクセスはエラーになりませんが，書き込みデータは失われます。**

9. **VR4120Aはすべての内部レジスタにアクセスできますが，**IBUS**マスタ・デバイスはそれらのレジスタにアクセスできません。**

保守／廃止

5. 2. 1. 2　統計カウンタ・レジスタ

統計カウンタ・レジスタは，MAC制御ブロックから送受信動作に関する統計情報を取得します。

それぞれのカウンタは32ビットで構成されています。カウンタのオーバフロー状態が発生すると，En_CAR1レジスタまたはEn_CAR2レジスタの対応するビットが1にセットされ，割り込みを発生します。割り込みは，En_CAM1レジスタまたはEn_CAM2レジスタのビットをセットすることでマスクできます。

カウント動作のため，ステータス・ベクタの変化後，すべての統計カウンタの更新を完了するために少し時間がかかります。

表5－3　統計カウンタ・レジスタのマップ

アドレス	レジスタ名	R/W	アクセス	説　明
1000_m140H	En_RBYT	R/W	W	バイト受信カウンタ
1000_m144H	En_RPKT	R/W	W	パケット受信カウンタ
1000_m148H	En_RFCS	R/W	W	CRCエラー受信カウンタ
1000_m14CH	En_RMCA	R/W	W	マルチキャスト・パケット受信カウンタ
1000_m150H	En_RBCA	R/W	W	ブロードキャスト・パケット受信カウンタ
1000_m154H	En_RXCF	R/W	W	コントロール・フレーム・パケット受信カウンタ
1000_m158H	En_RXPF	R/W	W	ポーズ・フレーム受信カウンタ
1000_m15CH	En_RXUO	R/W	W	未定義コントロール・フレーム受信カウンタ
1000_m160H	En_RALN	R/W	W	アラインメント・エラー受信カウンタ
1000_m164H	En_RFLR	R/W	W	データ長不一致受信カウンタ
1000_m168H	En_RCDE	R/W	W	コード・エラー受信カウンタ
1000_m16CH	En_RFCR	R/W	W	False Carrier受信カウンタ
1000_m170H	En_RUND	R/W	W	ショート・パケット受信カウンタ このカウンタは，64バイト長未満で有効なFCSを含むフレームを受信するたびにインクリメントします。
1000_m174H	En_ROVR	R/W	W	ジャバ・パケット受信カウンタ このカウンタは，1518バイト長を越え有効なFCSを含むフレームを受信するたびにインクリメントします。
1000_m178H	En_RFRG	R/W	W	エラー・ショート・パケット受信カウンタ このカウンタは，64バイト長未満で無効なFCSまたはアラインメント・エラーを含むフレームを受信するたびにインクリメントします。
1000_m17CH	En_RJBR	R/W	W	エラー・ジャバ・パケット受信カウンタ このカウンタは，1518バイト長を越え無効なFCSまたはアラインメント・エラーを含むフレームを受信するたびにインクリメントします。
1000_m180H	En_R64	R/W	W	64バイト・フレーム受信カウンタ このカウンタは，64バイト長の正しいまたは不正なフレームを受信するたびにインクリメントします。
1000_m184H	En_R127	R/W	W	65-127バイト・フレーム受信カウンタ このカウンタは，65-127バイト長の正しいまたは不正なフレームを受信するたびにインクリメントします。
1000_m188H	En_R255	R/W	W	128-255バイト・フレーム受信カウンタ このカウンタは，128-255バイト長の正しいまたは不正なフレームを受信するたびにインクリメントします。

保守／廃止

アドレス	レジスタ名	R/W	アクセス	説　明
1000_m18CH	En_R511	R/W	W	256-511バイト・フレーム受信カウンタ このカウンタは，256-511バイト長の正しいまたは不正なフレームを受信するたびにインクリメントします。
1000_m190H	En_R1K	R/W	W	512-1023バイト・フレーム受信カウンタ このカウンタは，512-1023バイト長の正しいまたは不正なフレームを受信するたびにインクリメントします。
1000_m194H	En_RMAX	R/W	W	1023-RMAXバイト・フレーム受信カウンタ このカウンタは，1023-RMAXバイト長の正しいまたは不正なフレームを受信するたびにインクリメントします。
1000_m198H	En_RVBT	R/W	W	有効バイト受信カウンタ このカウンタは，それぞれの受信された有効パケットのバイト・カウントによってインクリメントします。
1000_m19CH： 1000_m1BCH	N/A	-		将来の使用のため予約
1000_m1C0H	En_TBYT	R/W	W	バイト送信カウンタ
1000_m1C4H	En_TPKT	R/W	W	パケット送信カウンタ
1000_m1C8H	En_TFCS	R/W	W	CRCエラー送信パケット・カウンタ
1000_m1CCH	En_TMCA	R/W	W	マルチキャスト・パケット送信カウンタ
1000_m1D0H	En_TBCA	R/W	W	ブロードキャスト・パケット送信カウンタ
1000_m1D4H	En_TUCA	R/W	W	ユニキャスト・パケット送信カウンタ
1000_m1D8H	En_TXPF	R/W	W	ポーズ・コントロール・フレーム送信カウンタ
1000_m1DCH	En_TDFR	R/W	W	送信遅延カウンタ
1000_m1E0H	En_TXDF	R/W	W	送信過剰遅延カウンタ
1000_m1E4H	En_TSCL	R/W	W	シングル・コリジョン・パケット送信カウンタ
1000_m1E8H	En_TMCL	R/W	W	マルチ・コリジョン・パケット送信カウンタ
1000_m1ECH	En_TLCL	R/W	W	レイト・コリジョン・パケット・カウンタ
1000_m1F0H	En_TXCL	R/W	W	過剰コリジョン・パケット・カウンタ
1000_m1F4H	En_TNCL	R/W	W	トータル・コリジョン回数カウンタ
1000_m1F8H	En_TCSE	R/W	W	キャリア・センス・エラー・カウンタ このカウンタは，送信中にキャリア・センス・エラーが発生するフレームを送信するたびにインクリメントします。
1000_m1FCH	En_TIME	R/W	W	MAC内部エラー数カウンタ このカウンタは，送信中に内部MACエラーが発生するフレームを送信するたびにインクリメントします。

備考1. **“ アドレス ” フィールドと “ レジスタ名 ” フィールド内でのmとnの値については次のとおりです。**
イーサネット・コントローラ#1：m = 2, n = 1，
イーサネット・コントローラ#2：m = 3, n = 2

2. **“** R/W **” フィールド内で，**
“ W **” は，“ 書き込み可能 ” を意味します。**
“ R **” は，“ 読み取り可能 ” を意味します。**
“ RC **” は，“ 読み取りクリア ” を意味します。**
“ - **” は，“ アクセス不可能 ” を意味します。**

3. **すべての内部レジスタは，**32**ビット・ワード配列のレジスタです。**

4. **内部レジスタに対するバースト・アクセスは，禁止されています。バースト・アクセスすると，**NSR**の**IRERR**ビットがセットされ，**NMI**がV**R4120A**に対してアサートします。**

5. **予約領域に対してリード・アクセスすると，**NSR**レジスタの**CBERR**ビットがセットされ，**SysCMD［0］**にデータ・エラー・ビットが設定されたダミー読み取り応答データが読み出されます（一部の非公開レジスタを除きます）。**

6. **予約領域に対してライト・アクセスすると，**NSR**レジスタの**CBERR**ビットがセットされ，書き込みデータが失われます。**

7. **“ アクセス ” フィールド内で，**
“ W **” は，ワード・アクセスが有効であることを意味します。**
“ H **” は，ハーフ・ワード・アクセスが有効であることを意味します。**
“ B **” は，バイト・アクセスが有効であることを意味します。**

8. **リード・オンリー・レジスタに対するライト・アクセスはエラーになりませんが，書き込みデータは失われます。**

9. V**R4120A**はすべての内部レジスタにアクセスできますが，**IBUS**マスタ・デバイスはそれらのレジスタにアクセスできません。**

保守／廃止

5. 2. 1. 3　DMA/FIFOマネジメント・レジスタ

これらのレジスタは，このブロックの内部DMACによる送受信データの転送を制御します。

表5－4　DMA/FIFOマネジメント・レジスタのマップ

アドレス	レジスタ名	R/W	アクセス	説　　明
1000_m200H	En_TXCR	R/W	W	送信コンフィギュレーション・レジスタ
1000_m204H	En_TXFCR	R/W	W	送信FIFO制御レジスタ
1000_m208H： 1000_m210H	N/A	-	-	将来の使用のため予約
1000_m214H	En_TXDPR	R/W	W	送信ディスクリプタ・レジスタ
1000_m218H	En_RXCR	R/W	W	受信コンフィギュレーション・レジスタ
1000_m21CH	En_RXFCR	R/W	W	受信FIFO制御レジスタ
1000_m220H： 1000_m228H	N/A	-	-	将来の使用のため予約
1000_m22CH	En_RXDPR	R/W	W	受信ディスクリプタ・ポインタ・レジスタ
1000_m230H	En_RXPDR	R/W	W	受信プール・ディスクリプタ・レジスタ

備考1. “アドレス”フィールドと“レジスタ名”フィールド内でのmとnの値については次のとおりです。
イーサネット・コントローラ#1：m = 2, n = 1，
イーサネット・コントローラ#2：m = 3, n = 2

2. “R/W”フィールド内で，
“W”は，“書き込み可能”を意味します。
“R”は，“読み取り可能”を意味します。
“RC”は，“読み取りクリア”を意味します。
“-”は，“アクセス不可能”を意味します。

3. すべての内部レジスタは，32ビット・ワード配列のレジスタです。

4. 内部レジスタに対するバースト・アクセスは，禁止されています。バースト・アクセスすると，NSRのIRERRビットがセットされ，NMIがVR4120Aに対してアサートします。

5. 予約領域に対してリード・アクセスすると，NSRレジスタのCBERRビットがセットされ，SysCMD［0］にデータ・エラー・ビットが設定されたダミー読み取り応答データが読み出されます（一部の非公開レジスタを除きます）。

6. 予約領域に対してライト・アクセスすると，NSRレジスタのCBERRビットがセットされ，書き込みデータが失われます。

7. “アクセス”フィールド内で，
“W”は，ワード・アクセスが有効であることを意味します。
“H”は，ハーフ・ワード・アクセスが有効であることを意味します。
“B”は，バイト・アクセスが有効であることを意味します。

8. リード・オンリー・レジスタに対するライト・アクセスはエラーになりませんが，書き込みデータは失われます。

9. VR4120Aはすべての内部レジスタにアクセスできますが，IBUSマスタ・デバイスはそれらのレジスタにアクセスできません。

保守／廃止

5. 2. 1. 4　割り込み／コンフィギュレーション・レジスタ

これらのレジスタは，このブロックに対する割り込みの発生とコンフィギュレーションを制御します。

表5－5　割り込み／コンフィギュレーション・レジスタのマップ

アドレス	レジスタ名	R/W	アクセス	説　　明
1000_m234H	En_CCR	R/W	W	コンフィギュレーション・レジスタ
1000_m238H	En_ISR	RC	W	割り込み処理レジスタ
1000_m23CH	En_MSR	R/W	W	マスク処理レジスタ

備考1. “アドレス”フィールドと“レジスタ名”フィールド内でのmとnの値については次のとおりです。
イーサネット・コントローラ#1：m = 2, n = 1，
イーサネット・コントローラ#2：m = 3, n = 2

2. “R/W”フィールド内で，
“W”は，“書き込み可能”を意味します。
“R”は，“読み取り可能”を意味します。
“RC”は，“読み取りクリア”を意味します。
“-”は，“アクセス不可能”を意味します。

3. すべての内部レジスタは，32ビット・ワード配列のレジスタです。

4. 内部レジスタに対するバースト・アクセスは，禁止されています。バースト・アクセスすると，NSRのIRERRビットがセットされ，NMIがVR4120Aに対してアサートします。

5. 予約領域に対してリード・アクセスすると，NSRレジスタのCBERRビットがセットされ，SysCMD［0］にデータ・エラー・ビットが設定されたダミー読み取り応答データが読み出されます（一部の非公開レジスタを除きます）。

6. 予約領域に対してライト・アクセスすると，NSRレジスタのCBERRビットがセットされ，書き込みデータが失われます。

7. “アクセス”フィールド内で，
“W”は，ワード・アクセスが有効であることを意味します。
“H”は，ハーフ・ワード・アクセスが有効であることを意味します。
“B”は，バイト・アクセスが有効であることを意味します。

8. リード・オンリー・レジスタに対するライト・アクセスはエラーになりませんが，書き込みデータは失われます。

9. VR4120Aはすべての内部レジスタにアクセスできますが，IBUSマスタ・デバイスはそれらのレジスタにアクセスできません。

保守／廃止

5. 2. 2　En_MACC1（MACコンフィギュレーション・レジスタ1）

ビット	フィールド	R/W	ディフォールト	説　明
31：12	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
11	TXFC	R/W	0	送信フロー・コントロール・イネーブル： このビットを1にセットすると，ポーズ・コントロール・フレームの送信をイネーブルにします。 この機能はμPD98502では動作しないため，0に設定してください。
10	RXFC	R/W	0	受信フロー・コントロール・イネーブル： このビットを1にセットすると，MAC制御ブロックは，ポーズ・タイマ（En_PTVR）に設定されている時間分，ポーズ動作を実行します。ポーズ・タイマの設定は，このビットの設定とは無関係に，有効なポーズ・コントロール・フレームを受信すると更新されます。 この機能はμPD98502では動作しないため，0に設定してください。
9	SRXEN	R/W	0	受信イネーブル： このビットを1にセットすると，上位システムに対する受信データ・インタフェースの機能が有効になります。
8	PARF	R/W	0	コントロール・パケット・パス： このビットを1にセットすると，MAC制御ブロックは，コントロール・フレームを含むすべての受信パケットを上位CPUにてチェックする必要があります。このビットを0にセットすると，MAC制御ブロックは，コントロール・フレームをチェックします（コントロール・フレームは上位CPUに報告されません）。 PARFビットが1の場合には，RXFCビットの設定にかかわらず，有効なポーズ・コントロール・フレームを受信してもポーズ・タイマの値は更新されません。
7	PUREP	R/W	0	ピュア・プリアンブル： このビットを1にセットすると，プリアンブル中に“0101”以外のデータを許可しません。
6	FLCHT	R/W	0	長さフィールド・チェック： このビットを1にセットすると，MAC制御ブロックは，パケット内の長さフィールド値を実際のパケット長と比較し，その結果をステータス・ベクタに表示します。このビットを0にセットすると，MAC制御ブロックは，フレーム長フィールドをチェックしません。
5	NOBO	R/W	0	バック・オフなし： このビットを1にセットすると，MAC制御ブロックは，常にバック・オフ動作なしにパケットを送信します。
4	Reserved	-	-	将来の使用のため予約。0を書き込んでください。
3	CRCEN	R/W	0	CRC付加イネーブル： このビットを1にセットすると，MAC制御ブロックは，送信パケットの末尾に自動的にCRCを付加します。

保守／廃止

ビット	フィールド	R/W	ディフォールト	説　明
2	PADEN	R/W	0	PAD付加イネーブル： このビットを1にセットすると，CRCを付加する前の入力データ（TPD）長が60バイト未満であるとき，送信パケット長が64オクテット未満とならないように，MAC制御ブロックはPADを付加します。 PADENビットが1であることによりパディング処理が発生した場合には，CRCENビットの設定にかかわらず，パケットの末尾に自動的にCRCを付加します。
1	FULLD	R/W	0	全二重イネーブル： このビットを0にセットすると，半二重動作を行います。 このビットを1にセットすると，全二重動作を行います。
0	HUGEN	R/W	0	ラージ・パケット・イネーブル： このビットを1にセットすると，MACは，En_LMAXレジスタの値を越えるパケットの送受信を可能とします。 このビットを0にセットすると，En_LMAXレジスタの値を越えたパケットの送受信を中断します。

備考　SRXENビットを除く設定ビットを切り替える場合には，レジスタ設定後にソフトウエア・リセットをかけるようにしてください。

保守／廃止

5. 2. 3　En_MACC2（MACコンフィギュレーション・レジスタ2）

ビット	フィールド	R/W	ディフォールト	説　明
31：11	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
10	MCRST	R/W	0	MAC制御ブロック・ソフトウエア・リセット： このビットを1にセットすると，強制的にMAC制御ブロックにソフトウエアによるリセット動作を行うことができます。ソフトウエア・リセットを完了するには，このビットに再度0を書き込む必要があります。
9	RFRST	R/W	0	受信機能ブロック・ソフトウエア・リセット： このビットを1にセットすると，強制的に受信機能ブロックにソフトウエアによるリセット動作を行うことができます。ソフトウエア・リセットを完了するには，このビットに再度0を書き込む必要があります。
8	TFRST	R/W	0	送信機能ブロック・ソフトウエア・リセット： このビットを1にセットすると，強制的に送信機能ブロックにソフトウエアによるリセット動作を行うことができます。ソフトウエア・リセットを完了するには，このビットに再度0を書き込む必要があります。
7	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
6	BPNB	R/W	0	バック・オフなしバック・プレッシャ： このビットを1にセットすると，バック・プレッシャ後の送信にかぎり，バック・オフしません。
5	APD	R/W	0	オートVLANパッド： このビットを1にセットすると，En_VLTPレジスタに登録されたVLANタイプと一致するパケットが送信された場合，VLANパケットとして扱い，PADを付加します。
4	VPD	R/W	0	VLANパッド・モード： このビットを1にセットすると，送信するパケットを必ずVLANパケットとして扱い，PADを付加します。
3：0	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。

備考　リセット・ビットを除く設定ビットを切り替える場合には，レジスタ設定後にソフトウエア・リセットをかけるようにしてください。

5. 2. 4　En_IPGT（Back-to-Back IPGレジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：7	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
6：0	IPGT	R/W	13H	Back-to-Back IPG： このフィールドは，Back-to-Back時のIPGを設定します。 Back-to-Back IPGの計算式は次のとおりです。 IPG =（5＋IPGT）×4ビット・タイム

保守／廃止

5. 2. 5　En_IPGR（Non Back-to-Back IPGレジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：15	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
14：8	IPGR1	R/W	0EH	キャリア・センス時間： このフィールドは，送信動作がNon Back-to-Backモードで行われるときに，キャリア・センス時間を設定します。キャリア・センス時間の計算式は次のとおりです。 キャリア・センス時間 =（2＋IPGR1）×4ビット・タイム このフィールドで定義したキャリア・センス時間は，IPGR2フィールドで定義したNon Back-to-Back IPGに含まれています。
7	Reserved	-	-	将来の使用のため予約。0を書き込んでください。
6：0	IPGR2	R/W	13H	Non Back-to-Back IPG： このフィールドは，Non Back-to-Backモード時の送信動作にIPGを設定します。 Non Back-to-Back IPGの計算式は次のとおりです。 IPG =（5＋IPGR2）×4ビット・タイム

5. 2. 6　En_CLRT（コリジョン・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：14	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
13：8	LCOL	R/W	38H	コリジョン・ウインドウ： このフィールドは，コリジョン・ウインドウ幅を設定します。 コリジョン・ウインドウ幅の計算式は次のとおりです。 コリジョン・ウインドウ幅 =（LCOL＋8）×8ビット・タイム
7：4	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
3：0	RETRY	R/W	0FH	コリジョン発生時最大再送回数： コリジョンが発生した場合の最大再送信回数を設定します。この値以内で再送信が完了しない場合は，送信をアボートします。この値は最大衝突回数を示しています。

5. 2. 7　En_LMAX（最大パケット長レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：16	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
15：0	MAXF	R/W	600H	最大パケット長： このフィールドは，En_MACC1レジスタのHUGENビットが0にセットされているときに，送受信パケットの最大オクテット長を設定します。 受信：現在の受信パケット長がMAXFの値を越えると，MAC制御ブロックは受信を終了します。 送信：現在の送信パケット長がMAXFの値を越えると，MAC制御ブロックは送信をアボートします。

保守／廃止

5. 2. 8　En_LSA2（ステーション・アドレス・レジスタ2）

ビット	フィールド	R/W	ディフォールト	説　明
31：16	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
15：0	LSA2	R/W	0	ステーション・アドレスSA（47：32）： このフィールドは，ステーション・アドレスを設定します。MAC制御ブロックは，ステーション・アドレスSA（47：0）を，ポーズ・コントロール・フレームのソース・アドレスとして使用し，受信パケットのデスティネーション・アドレスとの比較に使用します。5. 3. 6を参照してください。

5. 2. 9　En_LSA1（ステーション・アドレス・レジスタ1）

ビット	フィールド	R/W	ディフォールト	説　明
31：0	LSA1	R/W	0	ステーション・アドレスSA（31：0）： このフィールドは，ステーション・アドレスを設定します。MAC制御ブロックは，ステーション・アドレスSA（47：0）を，ポーズ・コントロール・フレームのソース・アドレスとして使用し，受信パケットのデスティネーション・アドレスとの比較に使用します。5. 3. 6を参照してください。

5. 2. 10　En_PTVR（ポーズ・タイマ値リード・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：16	Reserved	R	0	将来の使用のため予約。
15：0	PTCT	R	0	ポーズ・タイマ・カウンタ： このフィールドは，現在のポーズ・タイマ値を示します。 受信フロー・コントロールがイネーブルである間（En_MACC1レジスタ：RXFCビットが1である間）のみ，このレジスタは有効な値を持ちます。

5. 2. 11　En_VLTP（VLANタイプ・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：16	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
15：0	VLTP	R/W	0	VLANタイプ： このフィールドはTPID値を設定します。 受信：MAC制御ブロックは，このフィールドと受信パケットのTPIDフィールド（ソース・アドレスに続く2オクテット）を比較し，VLANフレームを検出します。 送信：En_MACC2レジスタのAPDビットが1にセットされていて，現在の送信パケットのTPIDフィールドがこのフィールドの値と等しければ，MAC制御ブロックは，VLANフレームとしてパッドを付加します。

保守／廃止

5. 2. 12　En_MIIC（MIIコンフィギュレーション・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：16	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
15	MIRST	R/W	0	MIIマネジメント・インタフェース・ブロック・ソフトウエア・リセット： このビットを1にセットすると，MIIマネジメント・インタフェース・ブロックをソフトウエア・リセットします。ソフトウエア・リセットを解除するには，このビットに0を書き込む必要があります。
14：4	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
3：2	CLKS	R/W	0	周波数範囲選択： このフィールドは，内部クロックの周波数範囲を設定します。 MIIマネジメント・クロック（MIMCLK端子出力）は内部クロック（SCLK端子入力の2逓倍）を分割して生成され，これらのビットによりクロックの分割を設定します。これらのビットの設定は次のとおりです。 00：内部クロックが25 MHz 01：内部クロックが25 MHz以上，33 MHz以下 10：内部クロックが33 MHz以上，50 MHz以下 11：内部クロックが50 MHz以上，66 MHz以下（通常の場合） つまり，SCLK入力の値によって，これらのビットを設定することで，MIIマネジメント・クロック（MIMCLK端子出力）は2.5 MHz以下に設定されます。
1：0	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。

5. 2. 13　En_MCMD（MIIコマンド・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：2	Reserved	W	0	将来の使用のため予約。0を書き込んでください。
1	SCANC	W	0	スキャン・コマンド： このビットを1にセットすると，MAC制御ブロックはスキャン・コマンドを実行します。
0	RSTAT	W	0	MIIマネジメント・リード： このビットを1にセットすると，MAC制御ブロックは，MIIマネジメント・インタフェースによるリード・アクセスを実行します。

保守／廃止

5. 2. 14　En_MADR（MIIアドレス・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：13	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
12：8	FIAD	R/W	0	MII PHYアドレス： このフィールドは，マネジメント・アクセス中に32個のPHYデバイスから1つを選択するためのPHYアドレスを設定します。
7：5	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
4：0	RGAD	R/W	0	MIIレジスタ・アドレス： このフィールドは，マネジメント・アクセス中にPHYデバイス内の16ビット・レジスタからアクセスされるレジスタ・アドレスを設定します。

5. 2. 15　En_MWTD（MIIライト・データ・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：16	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
15：0	CTLD	R/W	0	MIIライト・データ： このフィールドは，MIIマネジメント・インタフェースを介してライト・アクセスする際のMIIマネジメント・ライト・データを設定します。

5. 2. 16　En_MRDD（MIIリード・データ・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：16	Reserved	R	0	将来の使用のため予約。
15：0	PRSD	R	0	MIIリード・データ： このフィールドは，MIIマネジメント・インタフェースを介してリード・アクセスされたMIIマネジメント・リード・データを示します。

保守／廃止

5. 2. 17　En_MIND（MIIインディケータ・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：3	Reserved	R	0	将来の使用のため予約。
2	NVALID	R	0	SCANコマンド開始ステータス： En_MCMDレジスタのSCANCビットが1にセットされると，このビットは1にセットされます。 スキャン・コマンドを使用した最初のリード・アクセスが完了すると，このビットはクリアされます。
1	SCANA	R	0	SCANコマンド・アクティブ： En_MCMDレジスタのSCANCビットが1にセットされている間，このビットは1にセットされます。
0	BUSY	R	0	BUSY： このビットは，MAC制御ブロックが，MIIマネジメント・インタフェースを介して外部PHYデバイスにマネジメント・アクセスを実行していることを示します。このビットは，マネジメント・アクセス中，1にセットされます。

5. 2. 18　En_AFR（アドレス・フィルタリング・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：4	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
3	PRO	R/W	0	プロミスカス・モード： このビットを1にセットすると，すべての受信パケットを受信します。5. 3. 6を参照してください。
2	PRM	R/W	0	マルチキャスト受信： このビットを1にセットすると，すべてのマルチキャスト・パケットを受信します。5. 3. 6を参照してください。
1	AMC	R/W	0	条件付きマルチキャスト受信： このビットを1にセットすると，ハッシュ・テーブルを用い，テーブルに一致したマルチキャスト・パケットを受信します。5. 3. 6を参照してください。
0	ABC	R/W	0	ブロードキャスト受信： このビットを1にセットすると，すべてのブロードキャスト・パケットを受信します。5. 3. 6を参照してください。

5. 2. 19　En_HT1（ハッシュ・テーブル・レジスタ1）

ビット	フィールド	R/W	ディフォールト	説　明
31：0	HT1	R/W	0	ハッシュ・テーブル1： このレジスタは，ハッシュ・テーブルとしてHT2レジスタとともに使用します。条件付きマルチキャスト・パケットを検出するために使用します。このレジスタは，ハッシュ・テーブルの上位32ビットHT（63：32）を設定します。5. 3. 6を参照してください。

保守／廃止

5. 2. 20　En_HT2（ハッシュ・テーブル・レジスタ2）

ビット	フィールド	R/W	ディフォールト	説　　明
31：0	HT2	R/W	0	ハッシュ・テーブル2： このレジスタは，ハッシュ・テーブルとしてHT1レジスタとともに使用します。条件付きマルチキャスト・パケットを検出するために使用します。このレジスタは，ハッシュ・テーブルの下位32ビットHT（31：0）を設定します。5. 3. 6を参照してください。

5. 2. 21　En_CAR1（キャリー・レジスタ1）

このレジスタのビットは，オーバフロー・イベントが統計カウンタで発生したことを示します。各ビットがカウンタに対応しており，統計カウンタにオーバフロー・イベントが発生すると，対応するビットに1がセットされます。このレジスタのいずれかのビットが1にセットされ，キャリー・マスク・レジスタ1（En_CAM1）の対応するマスク・ビットが1であると，En_ISRのCARRYビットがセットされます。

ビット	フィールド	R/W	ディフォールト	説　　明
31：16	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
15	C1VT	R/W	0	En_RVBTカウンタ・キャリー・ビット
14	C1UT	R/W	0	En_TUCAカウンタ・キャリー・ビット
13	C1BT	R/W	0	En_TBCAカウンタ・キャリー・ビット
12	C1MT	R/W	0	En_TMCAカウンタ・キャリー・ビット
11	C1PT	R/W	0	En_TPKTカウンタ・キャリー・ビット
10	C1TB	R/W	0	En_TBYTカウンタ・キャリー・ビット
9	C1MX	R/W	0	En_RMAXカウンタ・キャリー・ビット
8	C11K	R/W	0	En_R1Kカウンタ・キャリー・ビット
7	C1FE	R/W	0	En_R511カウンタ・キャリー・ビット
6	C1TF	R/W	0	En_R255カウンタ・キャリー・ビット
5	C1OT	R/W	0	En_R127カウンタ・キャリー・ビット
4	C1SF	R/W	0	En_R64カウンタ・キャリー・ビット
3	C1BR	R/W	0	En_RBCAカウンタ・キャリー・ビット
2	C1MR	R/W	0	En_RMCAカウンタ・キャリー・ビット
1	C1PR	R/W	0	En_RPKTカウンタ・キャリー・ビット
0	C1RB	R/W	0	En_RBYTカウンタ・キャリー・ビット

保守／廃止

5. 2. 22　En_CAR2（キャリー・レジスタ2）

このレジスタのビットは，オーバフロー・イベントが統計カウンタで発生したことを示します。各ビットがカウンタに対応しており，統計カウンタにオーバフロー・イベントが発生すると，対応するビットに1がセットされます。このレジスタのいずれかのビットが1にセットされ，キャリー・マスク・レジスタ2（En_CAM2）の対応するマスク・ビットが1であると，En_ISRのCARRYビットがセットされます。

ビット	フィールド	R/W	ディフォールト	説　明
31	C2DV	R/W	0	ステータス・ベクタ・オーバラン・ビット
30：23	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
22	C2IM	R/W	0	En_TIMEカウンタ・キャリー・ビット
21	C2CS	R/W	0	En_TCSEカウンタ・キャリー・ビット
20	C2NC	R/W	0	En_TNCLカウンタ・キャリー・ビット
19	C2XC	R/W	0	En_TXCLカウンタ・キャリー・ビット
18	C2LC	R/W	0	En_TLCLカウンタ・キャリー・ビット
17	C2MC	R/W	0	En_TMCLカウンタ・キャリー・ビット
16	C2SC	R/W	0	En_TSCLカウンタ・キャリー・ビット
15	C2XD	R/W	0	En_TXDFカウンタ・キャリー・ビット
14	C2DF	R/W	0	En_TDFRカウンタ・キャリー・ビット
13	C2XF	R/W	0	En_TXPFカウンタ・キャリー・ビット
12	C2TE	R/W	0	En_TFCSカウンタ・キャリー・ビット
11	C2JB	R/W	0	En_RBJRカウンタ・キャリー・ビット
10	C2FG	R/W	0	En_RFRGカウンタ・キャリー・ビット
9	C2OV	R/W	0	En_ROVRカウンタ・キャリー・ビット
8	C2UN	R/W	0	En_RUNDカウンタ・キャリー・ビット
7	C2FC	R/W	0	En_RFCRカウンタ・キャリー・ビット
6	C2CD	R/W	0	En_RCDEカウンタ・キャリー・ビット
5	C2FO	R/W	0	En_RFLRカウンタ・キャリー・ビット
4	C2AL	R/W	0	En_RALNカウンタ・キャリー・ビット
3	C2UO	R/W	0	En_RXUOカウンタ・キャリー・ビット
2	C2PF	R/W	0	En_RXPFカウンタ・キャリー・ビット
1	C2CF	R/W	0	En_RXCFカウンタ・キャリー・ビット
0	C2RE	R/W	0	En_RFCSカウンタ・キャリー・ビット

保守／廃止

5. 2. 23　En_CAM1（キャリー・マスク・レジスタ1）

このレジスタは，En_CAR1レジスタにビットを設定したことにより発生する割り込みをマスクします。
各ビットごとのマスクが可能です。

ビット	フィールド	R/W	ディフォールト	説　明
31：16	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
15	M1VT	R/W	0	En_RVBTカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
14	M1UT	R/W	0	En_TUCAカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
13	M1BT	R/W	0	En_TBCAカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
12	M1MT	R/W	0	En_TMCAカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
11	M1PT	R/W	0	En_TPKTカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
10	M1TB	R/W	0	En_TBYTカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
9	M1MX	R/W	0	En_RMAXカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
8	M11K	R/W	0	En_R1Kカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
7	M1FE	R/W	0	En_R511カウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
6	M1TF	R/W	0	En_R255カウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
5	M1OT	R/W	0	En_R127カウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
4	M1SF	R/W	0	En_R64カウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
3	M1BR	R/W	0	En_RBCAカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除

保守／廃止

ビット	フィールド	R/W	ディフォールト	説　　明
2	M1MR	R/W	0	En_RMCAカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
1	M1PR	R/W	0	En_RPKTカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
0	M1RB	R/W	0	En_RBYTカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除

保守／廃止

5. 2. 24　En_CAM2（キャリー・マスク・レジスタ2）

このレジスタは，En_CAR2レジスタにビットを設定したことにより発生する割り込みをマスクします。
各ビットごとのマスクが可能です。

ビット	フィールド	R/W	ディフォールト	説　明
31	M2DV	R/W	0	ステータス・ベクタ・オーバラン・マスク・ビット 0 = マスク 1 = マスク解除
30：23	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
22	M2IM	R/W	0	En_TIMEカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
21	M2CS	R/W	0	En_TCSEカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
20	M2NC	R/W	0	En_TNCLカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
19	M2XC	R/W	0	En_TXCLカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
18	M2LC	R/W	0	En_TLCLカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
17	M2MC	R/W	0	En_TMCLカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
16	M2SC	R/W	0	En_TSCLカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
15	M2XD	R/W	0	En_TXDFカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
14	M2DF	R/W	0	En_TDFRカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
13	M2XF	R/W	0	En_TXPFカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
12	M2TE	R/W	0	En_TFCSカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
11	M2JB	R/W	0	En_RBJRカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除

保守／廃止

ビット	フィールド	R/W	ディフォールト	説　　明
10	M2FG	R/W	0	En_RFRGカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
9	M2OV	R/W	0	En_ROVRカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
8	M2UN	R/W	0	En_RUNDカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
7	M2FC	R/W	0	En_RFCRカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
6	M2CD	R/W	0	En_RCDEカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
5	M2FO	R/W	0	En_RFLRカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
4	M2AL	R/W	0	En_RALNカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
3	M2UO	R/W	0	En_RXUOカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
2	M2PF	R/W	0	En_RXPFカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
1	M2CF	R/W	0	En_RXCFカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除
0	M2RE	R/W	0	En_RFCSカウンタ・キャリー・マスク・ビット 0 = マスク 1 = マスク解除

保守／廃止

5. 2. 25　En_TXCR（送信コンフィギュレーション・レジスタ）

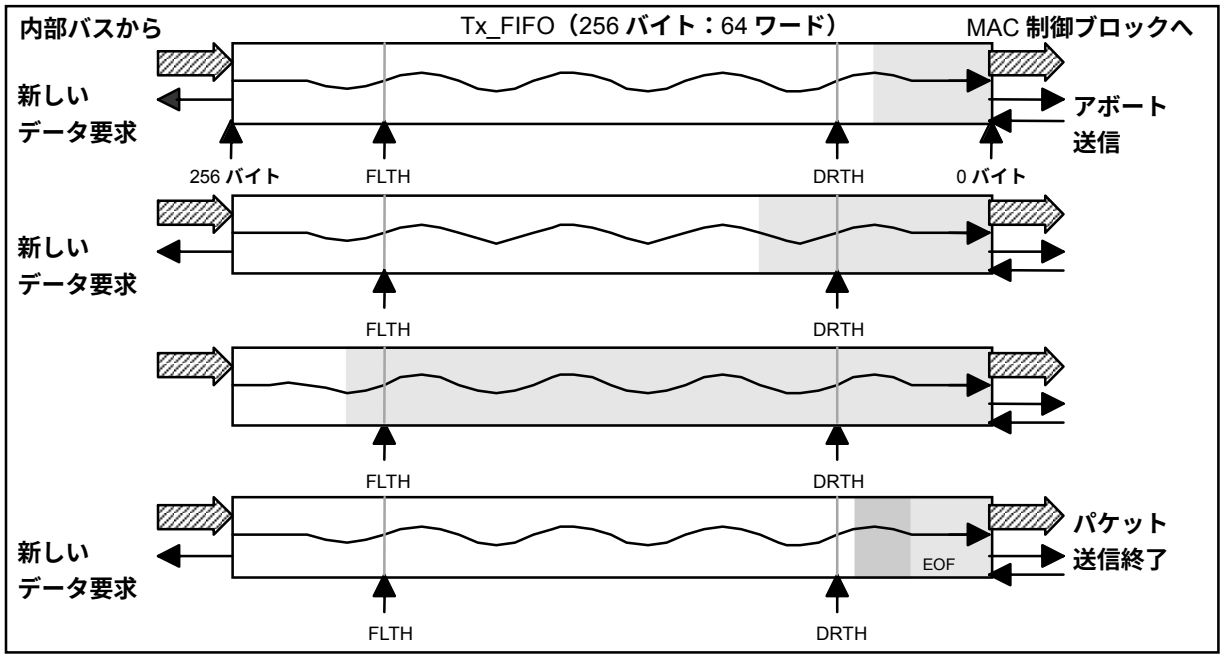
ビット	フィールド	R/W	ディフォールト	説　　明
31	TXE	R/W	0	送信イネーブル： トランスミッタの動作を設定します。 0：ディスエーブル 1：イネーブル
30：19	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
18：16	DTBS [2：0]	R/W	0	送信DMAバースト・サイズ： 000：1ワード（4バイト） 001：2ワード（8バイト） 010：4ワード（16バイト） 011：8ワード（32バイト） 100：16ワード（64バイト） 101：32ワード（128バイト） 110：64ワード（256バイト） 111：将来の使用のため予約
15：1	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
0	AFCE	R/W	0	オート・フロー制御イネーブル： 0：ディスエーブル 1：イネーブル この機能はμPD98502では動作しないため，0に設定してください。

保守／廃止

5. 2. 26　En_TXFCR（送信FIFO制御レジスタ）

ビット	フィールド	R/W	ディフォールト	説　　明
31：16	TPTV	R/W	FFFFH	送信ポーズ・タイマ値： ポーズ・コントロール・フレーム送信の待ち合わせ時間を設定するタイマ値です。
15：10	TX_DRTH	R/W	10H	送信ドレーン・スレッシュホールド・レベル： この値は，Tx-FIFOからMAC制御ブロックへの送信データに対してイネーブルになります。転送データがDMACで完了されずバッファ空きポインタがこのポインタを越えると，このMAC制御ブロックはアボート・パケットを送信します。図5－2を参照してください。これはワード・ポインタです。
9：8	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
7：2	TX_FLTH	R/W	30H	送信フィル・スレッシュホールド・レベル： この値は，このブロックのDMACを介して内部バスからFIFOへの送信データに対してイネーブルになります。図5－2を参照してください。これはワード・ポインタです。
1：0	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。

図5－2　Tx FIFO制御メカニズム



例　DRTHが8H（000100B）の場合，スレッシュホールドは32バイト（8×4），FLTHが20H（100000B）の場合，スレッシュホールドは128バイト（32×4）となります。

保守／廃止

5. 2. 27　En_TXDPR（送信ディスクリプタ・ポインタ）

ビット	フィールド	R/W	ディフォールト	説　　明
31：2	XMTDP	R/W	0	送信ディスクリプタ： 5. 3. 4を参照してください。
1：0	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。

5. 2. 28　En_RXCR（受信コンフィギュレーション・レジスタ）

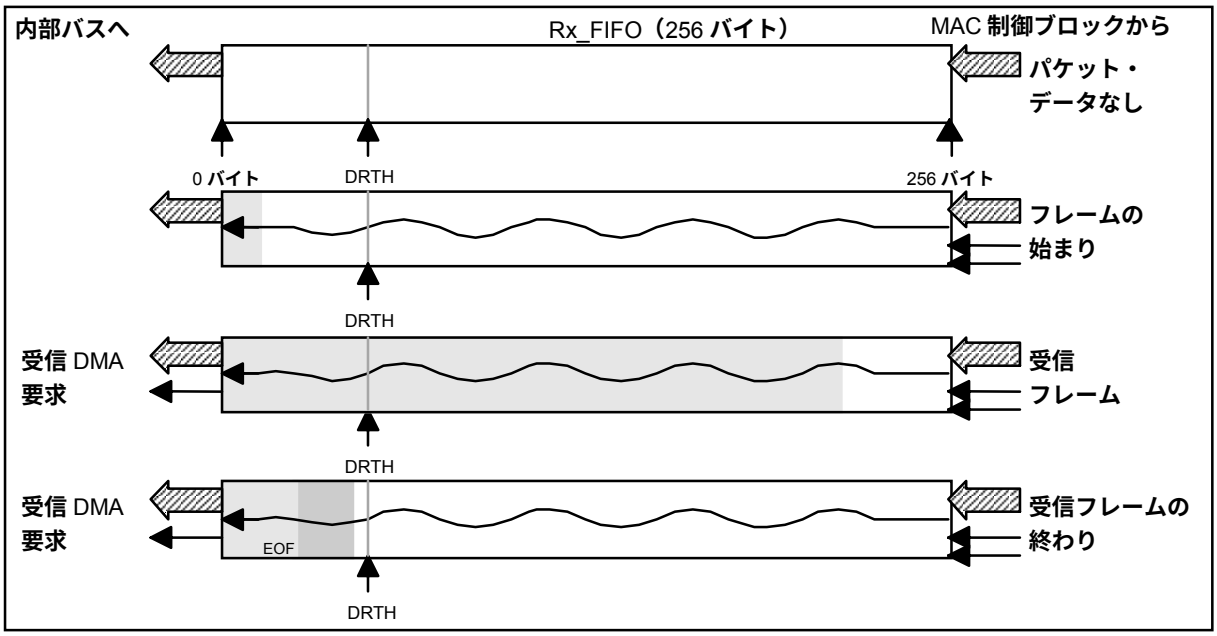
ビット	フィールド	R/W	ディフォールト	説　　明
31	RXE	R/W	0	受信イネーブル： レシーバの動作を設定します。 0：ディスエーブル 1：イネーブル
30：19	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
18：16	DRBS [2：0]	R/W	0	受信DMAバースト・サイズ： 000：1ワード（4バイト） 001：2ワード（8バイト） 010：4ワード（16バイト） 011：8ワード（32バイト） 100：16ワード（64バイト） 101：32ワード（128バイト） 110：64ワード（256バイト） 111：将来の使用のため予約
15：0	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。

保守／廃止

5. 2. 29　En_RXFCR（受信FIFO制御レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：26	UWM [7：2]	R/W	30H	上部ウォーター・マーク： このポインタは，En_TXCRのオート・フロー制御イネーブル・ビットとともに使用します。受信データのフィル・レベルがこのポインタを超えると，送信モジュールがフロー制御フレームを自動的に生成します。これはワード・ポインタです。 μPD98502ではフロー制御は機能しないので，この設定は無効となります。
25：24	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
23：18	LWM [7：2]	R/W	10H	下部ウォーター・マーク： このポインタは，En_TXCRのオート・フロー制御イネーブル・ビットとともに使用します。受信データのフィル・レベルがこのポインタを下回ると，ポーズ・タイマ値が0のポーズ・コントロール・フレームを送信します。これはワード・ポインタです。 μPD98502ではフロー制御は機能しないので，この設定は無効となります。
17：8	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
7：2	RX_DRTH	R/W	10H	受信ドレーン・スレッシュホールド・レベル： この値は，内部DMACを介してFIFOからIBUSへの受信データ転送に対してイネーブルになります。図5－3を参照してください。これはワード・ポインタです。
1：0	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。

図5－3　Rx FIFO制御メカニズム



保守／廃止

5. 2. 30　En_RXDPR（受信ディスクリプタ・ポインタ・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　　明
31：2	RCVDP	R/W	0	受信ディスクリプタ・ポインタ： 5. 3. 5を参照してください。
1：0	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。

5. 2. 31　En_RXPDR（受信プール・ディスクリプタ・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　　明
31	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
30：28	AL［2：0］	R/W	0	アラート・レベル
27：16	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
15：0	RNOD ［15：0］	R/W	0	ディスクリプタの残りの数

5. 2. 32　En_CCR（コンフィギュレーション・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　　明
31：1	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
0	SRT	R/W	0	ソフトウエア・リセット

保守／廃止

5. 2. 33　En_ISR（割り込み処理レジスタ）

各要因の有無（1：ステータスあり，0：ステータスなし）を表します。

ビット	フィールド	R/W	ディフォールト	説　　明
31	BUSERR	RC	0	イーサネット・コントローラ・ブロックがIBUSバス・マスタとなってデータ転送をした際に，存在しないメモリ・エリアに対してデータ転送を行い，転送エラーが発生したことを示す
30：16	Reserved	RC	0	将来の使用のため予約
15	XMTDN	RC	0	送信完了： パケットの送信が完了すると1になります。
14	TBDR	RC	0	送信バッファ・ディスクリプタ・リクエストがNullにある
13	TFLE	RC	0	送信フレーム長超過： En_MACC1のFLCHTビットが1のとき有効となります。送信パケットの長さフィールドが1500バイトを越えた値であったときに1になります。
12	UR	RC	0	アンダラン： 送信アンダランが発生すると1になります。
11	TABR	RC	0	送信中断[注]： 送信中断が発生すると1になります。
10	TCFRI	RC	0	コントロール・フレーム送信： コントロール・フレームを送信すると1になります。
9：8	Reserved	RC	0	将来の使用のため予約
7	RCVDN	RC	0	受信完了： パケットを受信すると1になります。
6	RBDRS	RC	0	受信バッファ・ディスクリプタの残数がアラート・レベル以下である
5	RBDRU	RC	0	受信バッファ・ディスクリプタの残数が0である
4	OF	RC	0	オーバフロー： 受信したパケットでオーバフローが起こると1になります。
3：1	Reserved	RC	0	将来の使用のため予約
0	CARRY	RC	0	キャリー・フラグ： 1つ以上の統計カウンタでオーバフローすると1になります。

注　送信中断とは，パケットの送信中に送信ディスクリプタ（5. 3. 3参照）のLCOL, ECOL, EDFR, TUDR, TGNTのいずれかが1となったことを示します。

保守／廃止

5. 2. 34　En_MSR（マスク処理レジスタ）

各割り込み要因はマスカブルです。En_MSRレジスタは，どの割り込みがイネーブルであるかを示します。

ディフォールト値はすべて0で，これはすべての割り込み要因がディスエーブルであることを示します。

各ビットに1を書き込むとEn_ISRの該当するビットが1となったときにVR4120Aに対して割り込みを発行します。

ビット	フィールド	R/W	ディフォールト	説　明
31	BUSERR	R/W	0	イーサネット・コントローラ・ブロックがバス・マスタ時の転送エラー 0 = マスク 1 = マスク解除
30：16	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
15	XMTDN	R/W	0	送信完了 0 = マスク 1 = マスク解除
14	TBDR	R/W	0	送信バッファ・ディスクリプタ・リクエストがNullにある 0 = マスク 1 = マスク解除
13	TFLE	R/W	0	送信フレーム長超過 0 = マスク 1 = マスク解除
12	UR	R/W	0	アンダラン 0 = マスク 1 = マスク解除
11	TABR	R/W	0	送信中断 0 = マスク 1 = マスク解除
10	TCFRI	R/W	0	コントロール・フレーム送信 0 = マスク 1 = マスク解除
9：8	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
7	RCVDN	R/W	0	受信完了 0 = マスク 1 = マスク解除
6	RBDRS	R/W	0	受信バッファ・ディスクリプタ・リクエストがアラート・レベルにある 0 = マスク 1 = マスク解除
5	RBDRU	R/W	0	受信バッファ・ディスクリプタ・リクエストが0にある 0 = マスク 1 = マスク解除
4	OF	R/W	0	オーバフロー 0 = マスク 1 = マスク解除
3：1	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。

ビット	フィールド	R/W	ディフォールト	説　　明
0	CARRY	R/W	0	キャリー・フラグ。統計カウンタのオーバフローを示す 0 = マスク 1 = マスク解除

保守／廃止

5. 3　オペレーション

5. 3. 1　初 期 化

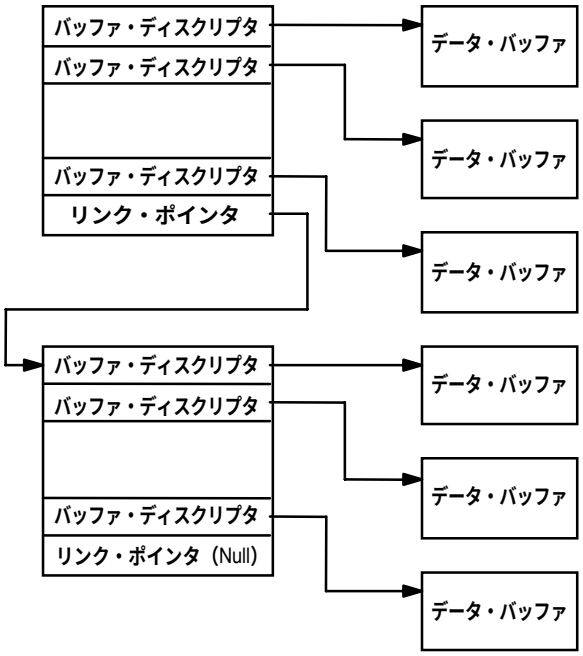
パワーオン・リセットまたはソフトウエア・リセットのあと，VR4120Aは以下のレジスタを設定しなければなりません。

(i) 割り込みマスク・レジスタ
(ii) コンフィギュレーション・レジスタ
(iii) MIIマネジメント・レジスタ
(iv) プール／バッファ・ディスクリプタ・レジスタ

5. 3. 2　イーサネット・コントローラ・ブロックのバッファ構造

イーサネット・コントローラのデータ・バッファ構造を図5－4に示します。

図5－4　イーサネット・ブロックのバッファ構造



保守／廃止

5. 3. 3　バッファ・ディスクリプタのフォーマット

送信ディスクリプタのフォーマットを図5－5に，説明を表5－6に示します。

図5－5　送信ディスクリプタのフォーマット

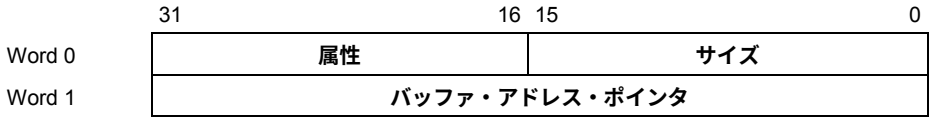


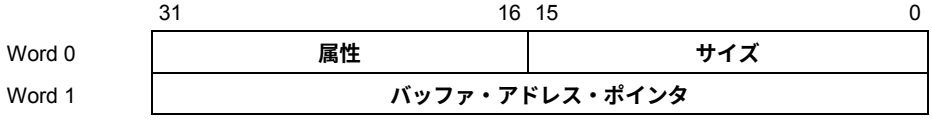
表5－6　送信ディスクリプタの属性

属性／サイズ	ビット名	ステータス
31	L	ラスト・ディスクリプタ
30	D/L	1 = データ・バッファ　0 = リンク・ポインタ
29	OWN	オーナ・ビット 1 = イーサネット・コントローラ　0 = VR4120A イーサネット・コントローラは，それぞれのディスクリプタにデータを転送しはじめたあとに，このビットをセットします。
28	DBRE	データ・バッファ・リード・エラー
27	TUDR	送信アンダラン・エラー
26	CSE	キャリア・センス・ロスト・エラー
25	LCOL	レイト・コリジョン
24	ECOL	過剰コリジョン
23	EDFR	過剰遅延
22：19	-	将来の使用のため予約。0を書き込んでください。
18	TGNT	送信長（En_LMAXのMAXF値）を越えるフレームを送信（En_MACC1：HUGEN = 0のとき）
17	-	将来の使用のため予約。0を書き込んでください。
16	TOK	送信終了
15：0	SIZE	送信バイト・カウント

備考　各ビットに対して，事象ありの場合は1，事象なしの場合は0をセットします（D/Lビット，OWNビットを除く）。

受信ディスクリプタのフォーマットを図5－6に，説明を表5－7に示します。

図5－6　受信ディスクリプタのフォーマット



保守／廃止

表5−7　受信ディスクリプタの属性

属性／サイズ	ビット名	ステータス
31	L	ラスト・ディスクリプタ
30	D/L	1 = データ・バッファ　0 = リンク・ポインタ
29	OWN	オーナ・ビット 1 = イーサネット・コントローラ　0 = VR4120A イーサネット・コントローラは，それぞれのディスクリプタにデータを転送しはじめたあとに，このビットをセットします。
28	DBWE	データ・バッファ・ライト・エラー
27：25	FTYP	フレーム・タイプ［2：0］： 000：ブロードキャスト・フレーム 001：マルチキャスト・フレーム 010：ユニキャスト・フレーム 011：VLANフレーム 100：ポーズ・コントロール・フレーム 101：コントロール・フレーム（ポーズを除く） 11x：将来の使用のため予約
24	OVRN	オーバラン・エラー
23	-	将来の使用のため予約
22	-	将来の使用のため予約
21	RCV	MIRER信号検出
20	FC	キャリア消失検出
19	CRCE	CRCエラー
18	DRNB	ドリブル・ニブル・パケット受信
17	RFLE	受信フレーム長エラー
16	RXOK	受信終了
15：0	SIZE	受信バイト・カウント

備考1. ドリブル・ニブル：ドリブル・ニブルが発生した場合，RXOKとDRNBの両方がセットされます。

2. 各ビットに対して，事象ありの場合は1，事象なしの場合は0をセットします（D/Lビット，OWNビットを除く）。

保守／廃止

5. 3. 4　フレーム送信

トランスミッタは，VR4120Aからの制御をほとんど受けないで自動的に送受信動作するように設計されています。VR4120Aが送信ディスクリプタ・ポインタ・レジスタ（En_TXDPR）と送信イネーブル（TXE）をセットしてトランスミッタの動作をイネーブルにすると，イーサネット・コントローラは最初の送信データ・バッファをバッファ・ディスクリプタから取り出します。

送信FIFOのドレーン・スレッシュホールド・レベル（Tx_DRTH）を越えると，MAC制御ブロックの送信ロジック回路は，プリアンブル・シーケンス，スタート・フレーム・デリミタ，フレーム情報の送信を開始します。ただし，その際，コントローラはラインがビジィ（キャリア・センス（MICRSおよびMI2CRS信号）がアクティブ）であれば，送信を遅らせます。送信する前に，コントローラはキャリア・センス（MICRSおよびMI2CRS信号）がインアクティブになるのを待たなければなりません。キャリア・センス（MICRSおよびMI2CRS信号）がインアクティブになると，コントローラは，キャリア・センス（MICRSおよびMI2CRS信号）がEn_IPGRレジスタのIPGR1ビット時間の間インアクティブであるかどうか確認します。もしそうなら，さらにIPGR2－IPGR1ビット時間（すなわち，IPGは通常96ビット時間）待ったあとで送信を開始します。

フレームの送信中にコリジョンが発生すると，イーサネット・コントローラは，IEEE802.3で規定されたバック・オフ手順に従い，リトライ・リミットしきい値（En_CLRTレジスタのRETRY）に達するまでフレームの再送信を試みます。イーサネット・コントローラは，送信フレームの最初の64バイトを送信FIFOに保持していますので，コリジョンのときシステム・メモリから再度その64バイトを読み込むことはありません。これにより，バス効率とレイテンシが改善します。

イーサネット・コントローラが送信バッファ・ディスクリプタを読み込み，データ・バッファの終わりの“L”ビットは1にセットされていることを示す場合，En_MACC1レジスタのCRCENがイネーブルならば，データの終わりにFCSを付加します。

En_MACC1レジスタのPADENビットがセットされている場合，64バイト以下のショート・フレームは送信ロジックによって自動的にパッドされます。送信フレーム長が1518バイトを越えると，イーサネット・コントローラは割り込みをアサートします。しかし，フレーム自体はそのまま送信されます。

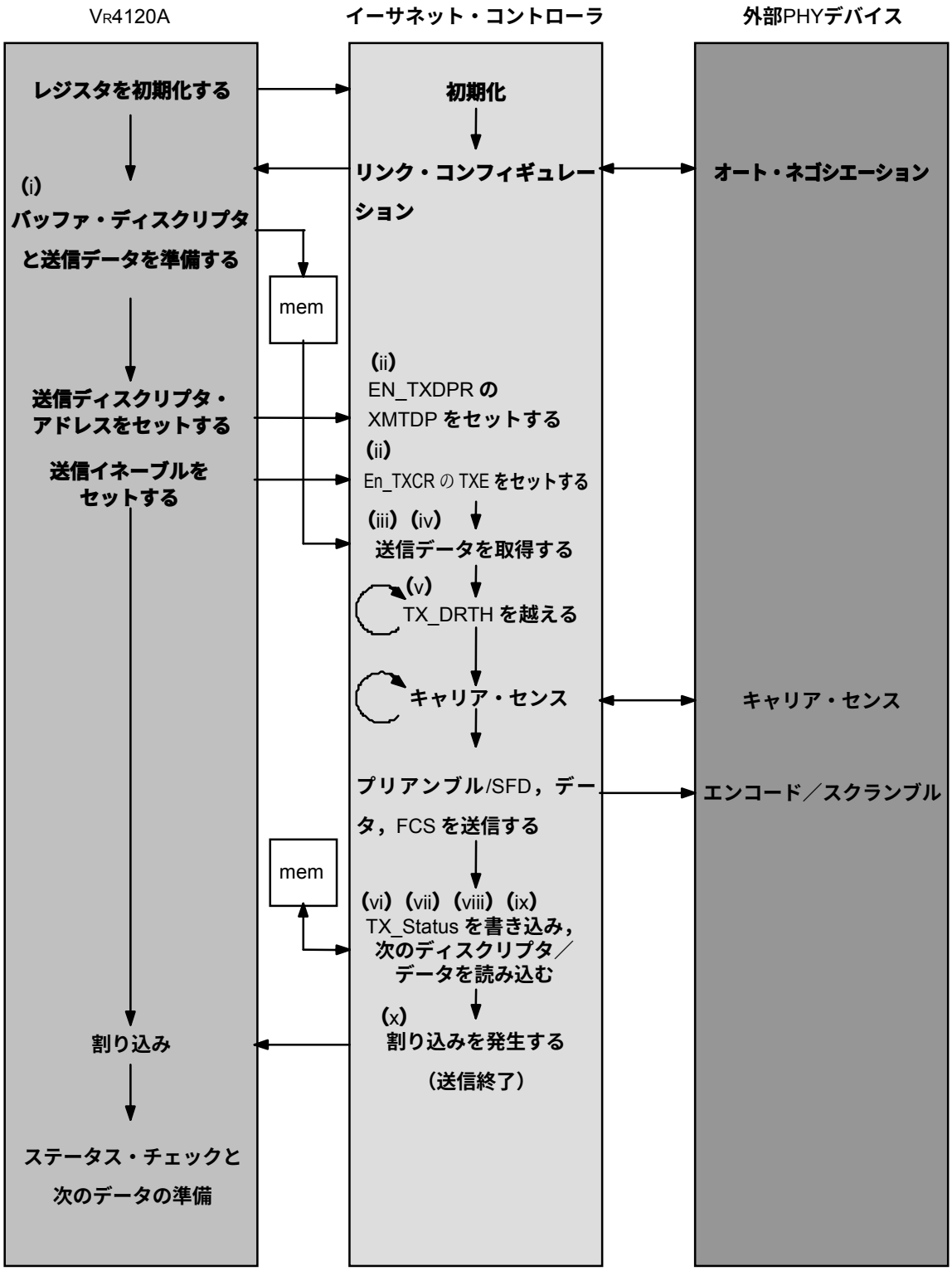
現在のディスクリプタがフレームの終わり（L = 1）であることを示さない場合，イーサネット・コントローラは次のバッファ・ディスクリプタを読み込み，データ・バッファから引き続きデータを読み込みます。イーサネット・コントローラがパケット全体を送信したあと，イーサネット・コントローラは，送信状態を最後のディスクリプタ（L = 1）に書き込み，送信の終わりを示す割り込みを発生（En_MSRのXMTDN = 1であること）します。その後，イーサネット・コントローラは，次の送信バッファ・ディスクリプタを取り出し，次のデータがあれば，上述の手順と同じように送信します。

イーサネット・コントローラがポーズ・コントロール・フレームを受信し，フロー制御がアクティブである場合，イーサネット・コントローラ・トランスミッタは，送信中でない場合，ただちに停止，または，現在のフレームが送信中の場合，コリジョンで停止するまで送信を続けます。ポーズ・タイマがタイムアウトするか，イーサネット・コントローラがゼロ値のポーズ・コントロール・フレームを受け取ると，次のフレームから送信を再び始めます。

送信手順は次のとおりです（**図5－7**）。

保守／廃止

図5－7　送信手順



保守／廃止

送信パケットのオペレーション・フロー

(i) 送信データをデータ・バッファに準備する。
(ii) レジスタ（XMTDP, TXE）を初期化する。
(iii) 送信用バッファ・ディスクリプタをSDRAMから読み込む。
(iv) マスタDMAがバースト・オペレーションを使用してデータ・バッファから送信データを読み込む。
(v) 送信ドレーン・スレッシュホールド（TX_DRTH）を越えるのを待つ。
　キャリアを検知する。
　データを送信する（プリアンブル，SFD，データ）。
(vi) 連続データを読み込む
　現在のバッファ・ディスクリプタが最後のパケット（L = 0）でない場合，引き続きデータを読み込む。
　現在の送信ディスクリプタ・ポインタをインクリメントする。
(vii) 次のバッファ・ディスクリプタを読み込む。
(viii) 連続データをデータ・バッファから再び読み込む。
(ix) 送信ステータスを最後のバッファ・ディスクリプタ（L = 1）に格納する。
(x) 割り込みを発生する。
(xi) もしあれば，次のバッファ・ディスクリプタとデータを読み込む。

備考　アンダランまたは過剰コリジョンが発生したことにより送信が中断すると，新たなディスクリプタ取得を行わないので，バッファ・ディスクリプタのステータスを確認したあとで，XMTDPを再びセットする必要があります。

保守／廃止

5. 3. 5　フレーム受信

レシーバは，VR4120Aからの制御をほとんど受けないで動作するように設計されており，アドレスの認識，CRCチェック，最大フレーム長チェックを行います。

ドライバが，受信ディスクリプタ・ポインタ・レジスタ（En_RXDPR）と受信イネーブル（RXE）をセットしてレシーバをイネーブルにすると，受信フレーム処理をただちに開始します。レシーバは，最初のパケットで有効プリアンブル（PA）／スタート・フレーム・デリミタ（SFD）ヘッダをまずチェックします。PA/SFDが有効なら，それは削除（受信データとはなりません）され，以後，レシーバによってフレームが受信処理されます。有効なPA/SFDが見つからないと，フレームは無視されます。

データのコリジョン・ウインドウ（64バイト）を越えてデータが受信され，アドレス認識が5. 3. 6に示すフレームと一致する場合，イーサネット・コントローラは入力フレームを受信データ・バッファに転送し始めます。フレームがRUNT（コリジョンのため）であったりアドレス認識で不一致となると，受信バッファ転送は行われず，バッファは満杯になりません。このようにしてコリジョン・フレームは，ネットワークに重大な影響を及ぼすレイト・コリジョン以外はメモリには転送されません。

また，受信データがSDRAMに転送されたあとでも，受信後に受信ステータスをディスクリプタに書き込むので，問題にはなりません。

受信フレームがデータ・バッファの長さを越えると，イーサネット・コントローラは，テーブルから次の受信ディスクリプタ・バッファを取り出し，空きがある場合，フレームの残りをこのデータ・バッファに転送し続けます。

ディスクリプタの残り数がアラート・レベル（En_RXPDRのAL［2：0］）の4倍未満ならば，イーサネット・コントローラは，割り込みを発生して新たにディスクリプタを追加するように要求します。

イーサネット・コントローラは，受信中にショートまたはロング・フレームをチェックします。フレームが終了する（キャリア・センスはインアクティブである）と，受信CRCフィールドがチェックされ，データ・バッファに受信結果を書き込みます。イーサネット・フレームの最後のディスクリプタのSIZEフィールドに書き込まれたデータ長は，フレーム全体の長さです。長さが64バイト未満のフレームはDMA転送されないで，データがRx FIFOにあるため，システム・バスの効率には影響せずにハードウエアで廃棄されます。

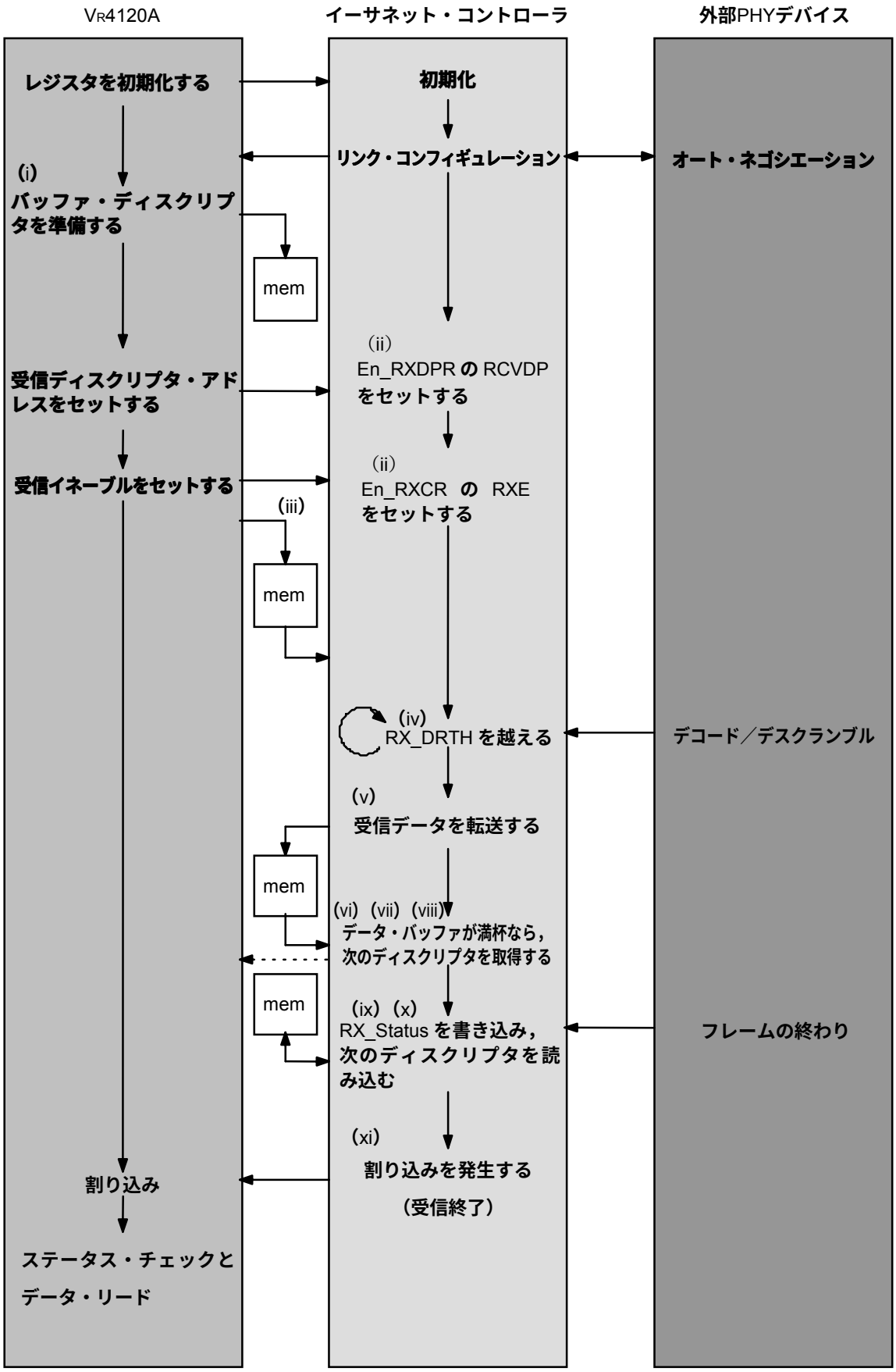
注意　ドレーン・スレッシュホールド・レベルは，16ワードを越える値を推奨します。

フレームが受信完了すると，イーサネット・コントローラは，受信ディスクリプタのLビットを1にセットし，フレーム・ステータス・ビットを受信ディスクリプタに書き込み，OWNビットをセットします。イーサネット・コントローラは，割り込みを発生（En_MSRのRCVDN = 1であること）し，受信されたフレームがメモリ内にあることを示します。引き続き，イーサネット・コントローラは新しいフレームを待ちます。

受信手順は次のとおりです（**図**5－8）。

保守／廃止

図5－8　受信手順



保守／廃止

受信パケットのオペレーション・フロー

(i) 受信バッファ・ディスクリプタを準備する。
(ii) レジスタ（En_RXDPRのRCVDPとEn_RXCRのRXE）をセットする。
(iii) 受信バッファ・ディスクリプタを読み込む。
(iv) 受信ドレーン・スレッシュホールド（RXDRTH）を越えるのを待つ。
(v) マスタDMAバースト・オペレーションを使用して受信データをデータ・バッファに書き込む。
(vi) 現在のデータ・バッファが満杯なら，受信ディスクリプタ・ポインタをインクリメントする。
(vii) 受信ディスクリプタの残り数（En_RXPDRのRNOD）をチェックする。
ディスクリプタの残り数がアラート・レベル（En_RXPDRのAL［2：0］）の4倍未満ならば，割り込みを発生してディスクリプタの追加を要求する。
(viii) 次のバッファ・ディスクリプタを読み込む。
(ix) 受信データを格納する。
(x) 受信ステータスを最後のバッファ・ディスクリプタ（L = 1）に格納する。
(xi) 受信の終わりに割り込みを発生する。
(xii) もし次のフレームの受信が行われ，すでにFIFO内に64バイト以上のデータが蓄積済みであれば，次の受信ディスクリプタを読み込む。

受信バッファ・ディスクリプタを追加する方法

(i) イーサネット・コントローラから割り込み（バッファ・ディスクリプタの残り数がアラート・レベルを下回った）が発生。
(ii) 受信バッファ・ディスクリプタをメモリに追加する。
(iii) バッファ・ディスクリプタの数がアラート・レベルを越えるようにEn_RXPDRにセットする。

保守／廃止

5.3.6　アドレス・フィルタリング

イーサネット・コントローラは，デスティネーション・アドレスを受信パケットごとにチェックできます。デスティネーション・アドレスは，VR4120Aによってセットされる En_AFR レジスタの条件を用いて，フィルタリングされます。ユニキャスト，マルチキャスト，ブロードキャストに対する条件は，独自に設定できます。

(1) ユニキャスト・アドレス・フィルタリング

受信パケットのデスティネーション・アドレスと En_LSA1 レジスタと En_LSA2 レジスタのステーション・アドレスを比較します。双方が一致すると，受信パケットを受け付けます。比較は受信パケットごとに行います。

(2) マルチキャスト・アドレス・フィルタリング

2つのフィルタリング方法をサポートします。1つの方法では，En_AFR レジスタの PRM ビットを1にセットすることで，受信したすべてのマルチキャスト・パケットを受け付けます。

もう1つの方法では，En_AFR レジスタの AMC ビットを1にセットすることで受信したマルチキャスト・パケットは，En_HT1 レジスタと En_HT2 レジスタの値で構成されたハッシュ・テーブルを用いて，フィルタリングされます。この方法では，受信パケットのマルチキャスト・デスティネーション・アドレスに対して次の CRC 計算多項式を用いて CRC 演算を実行します。

$X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$

この CRC 計算結果のビット［28：23］の6ビットがデコードされます。デコードされた結果が En_HT1 レジスタと En_HT2 レジスタにセットされた値と等しいなら，マルチキャスト・デスティネーション・アドレス付きの受信パケットを受け付けます。En_HT1 レジスタと En_HT2 レジスタをセットするためには，受信するマルチキャスト・デスティネーション・アドレスに対する CRC 演算結果を，マルチキャスト・パケットを受信する前に，設定する必要があります。

(3) ブロードキャスト・アドレス・フィルタリング

En_AFR レジスタの ABC ビットを1にセットすると，ブロードキャスト・デスティネーション・アドレス付きのすべての受信パケットを受信します。

(4) プロミスカス・モード

En_AFR レジスタの PRO ビットを1にセットすると，受信パケットをすべて受信します。

フィルタリング手順は次のとおりです。

まず，En_MACC1 レジスタの SRXEN ビットを0にセットします。この場合，受信データ・インタフェースがディスエーブルになります。次に，ステーション・アドレスを En_LSA1 レジスタと En_LSA2 レジスタにセットします。En_AFR レジスタも，ユニキャスト，マルチキャスト，ブロードキャスト受信をイネーブルにするために1にセットします。さらに，En_HT1 レジスタと En_HT2 レジスタは，ハッシュ・テーブルでマルチキャスト・アドレス・フィルタリングを使用する場合にセットする必要があります。これらの手順後，受信データ・インタフェースをイネーブルにするために，SRXEN を1にセットします。

保守／廃止

第6章　USBコントローラ

6.1 概　　要

USBコントローラは，USBを介してデータ通信を行います。USBコントローラの特徴は次のとおりです。

6.1.1 特　　徴

- ユニバーサル・シリアル・バス仕様rev.1.1に準拠
- USBコミュニケーション・デバイス・クラス仕様に準拠した動作をサポート
- フルスピード（12 Mbps）のデータ転送モードをサポート
- コントロール転送用エンドポイント（EP0）に加えて，さらに6つのエンドポイント（EP1-EP6）を内蔵（インタラプト転送，アイソクロノス転送，バルク転送（表6－1を参照））
- コントロール転送のために内蔵64バイトTx FIFOをサポート
- アイソクロノス転送のために内蔵128バイトTx FIFOをサポート
- バルク転送のために内蔵128バイトTx FIFOをサポート
- 割り込み転送のために内蔵64バイトTx FIFOをサポート
- コントロール／アイソクロノス／バルク／割り込み転送のために内蔵128バイト共有Rx FIFOをサポート
- 送受信データを転送するためにDMA機能をサポート
- 制御／ステータス・レジスタをサポート
- ホストPCから発行されるサスペンド／リジューム信号に対応（VR4120Aによる処理が必要）
- リモート・ウェークアップをサポート（VR4120Aによる処理が必要）
- 内部バス（IBUS）マスタとスレーブ・インタフェース・ブロックへのダイレクト接続をサポート

表6－1　各転送モードとエンドポイントとの関係

エンドポイント	転送タイプ	FIFOサイズ
EP0	コントロールIN/OUT転送	64バイト（IN）/128バイト（OUT）注
EP1	アイソクロノスIN転送	128バイト
EP2	アイソクロノスOUT転送	128バイト注
EP3	バルクIN転送	128バイト
EP4	バルクOUT転送	128バイト注
EP5	インタラプトIN転送	64バイト
EP6	インタラプトOUT転送	128バイト注

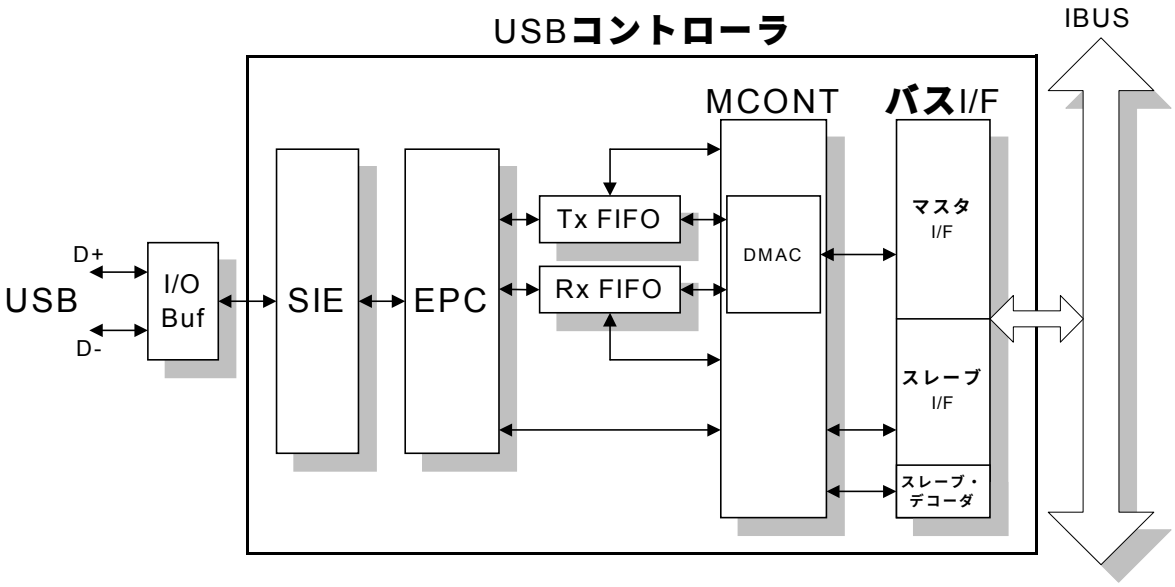
注　これらのFIFOはIN転送/OUT転送双方で共有しています。

保守／廃止

6. 1. 2　内部ブロック図

USBコントローラの内部ブロック図を以下に示します。

図6－1　USBコントローラの内部構成



USBコントローラの構成は，以下のブロックを特徴としています。

SIE（シリアル・インタフェース・エンジン）	：シリアル／パラレル変換，NRZIエンコード／デコード，CRC演算などを行います。
EPC（エンドポイント・コントローラ）	：各エンドポイントに対してデータの送受信を行います。
Tx FIFO（送信FIFO）	：データを送信するためのFIFOです。
Rx FIFO（受信FIFO）	：データを受信するためのFIFOです。
MCONT（メイン・コントローラ）	：送受信を制御するブロックです。
DMAC（DMAコントローラ）	：DMA転送を制御するブロックです。
マスタI/F（マスタ・インタフェース）	：内部バス・インタフェースのマスタ・セクションです。
スレーブI/F（スレーブ・インタフェース）	：内部バス・インタフェースのスレーブ・セクションです。
I/O Buf（I/Oバッファ）	：USBの電気的特性を満たすI/Oバッファです。
IBUS（インターナル・バス）	：μPD98502の内部バスです。

保守／廃止

6. 2　レジスタ

このセクションは，VR4120Aからアクセスできるレジスタのマッピングについて記述します。USBのベース・アドレスは，1000_1000Hです。

6. 2. 1　レジスタ・マップ

アドレス	レジスタ名	R/W	アクセス	説　明
1000_1000H	U_GMR	R/W	W/H/B	USBジェネラル・モード・レジスタ
1000_1004H	U_VER	R	W/H/B	USBフレーム・ナンバ／バージョン・レジスタ
1000_1008H：1000_100CH	N/A	-	-	将来の使用のため予約
1000_1010H	U_GSR1	RC	W	USBジェネラル・ステータス・レジスタ1
1000_1014H	U_IMR1	R/W	W/H/B	USB割り込みマスク・レジスタ1
1000_1018H	U_GSR2	RC	W	USBジェネラル・ステータス・レジスタ2
1000_101CH	U_IMR2	R/W	W/H/B	USB割り込みマスク・レジスタ2
1000_1020H	U_EP0CR	R/W	W/H/B	USB EP0制御レジスタ
1000_1024H	U_EP1CR	R/W	W/H/B	USB EP1制御レジスタ
1000_1028H	U_EP2CR	R/W	W/H/B	USB EP2制御レジスタ
1000_102CH	U_EP3CR	R/W	W/H/B	USB EP3制御レジスタ
1000_1030H	U_EP4CR	R/W	W/H/B	USB EP4制御レジスタ
1000_1034H	U_EP5CR	R/W	W/H/B	USB EP5制御レジスタ
1000_1038H	U_EP6CR	R/W	W/H/B	USB EP6制御レジスタ
1000_103CH	N/A	-	-	将来の使用のため予約
1000_1040H	U_CMR	R/W	W	USBコマンド・レジスタ
1000_1044H	U_CA	R/W	W/H/B	USBコマンド拡張レジスタ
1000_1048H	U_TEPSR	R	W/H/B	USB Txエンドポイント・ステータス・レジスタ
1000_104CH	N/A	-	-	将来の使用のため予約
1000_1050H	U_RP0IR	R/W	W/H/B	USB Rxプール0情報レジスタ
1000_1054H	U_RP0AR	R	W/H/B	USB Rxプール0アドレス・レジスタ
1000_1058H	U_RP1IR	R/W	W/H/B	USB Rxプール1情報レジスタ
1000_105CH	U_RP1AR	R	W/H/B	USB Rxプール1アドレス・レジスタ
1000_1060H	U_RP2IR	R/W	W/H/B	USB Rxプール2情報レジスタ
1000_1064H	U_RP2AR	R	W/H/B	USB Rxプール2アドレス・レジスタ
1000_1068H：1000_106CH	N/A	-	-	将来の使用のため予約
1000_1070H	U_TMSA	R/W	W/H/B	USB Txメールボックス・スタート・アドレス・レジスタ
1000_1074H	U_TMBA	R/W	W/H/B	USB Txメールボックス・ボトム・アドレス・レジスタ
1000_1078H	U_TMRA	R/W	W/H/B	USB Txメールボックス・リード・アドレス・レジスタ
1000_107CH	U_TMWA	R	W/H/B	USB Txメールボックス・ライト・アドレス・レジスタ
1000_1080H	U_RMSA	R/W	W/H/B	USB Rxメールボックス・スタート・アドレス・レジスタ
1000_1084H	U_RMBA	R/W	W/H/B	USB Rxメールボックス・ボトム・アドレス・レジスタ
1000_1088H	U_RMRA	R/W	W/H/B	USB Rxメールボックス・リード・アドレス・レジスタ
1000_108CH	U_RMWA	R	W/H/B	USB Rxメールボックス・ライト・アドレス・レジスタ

保守／廃止

アドレス	レジスタ名	R/W	アクセス	説　　明
1000_1090H： 1000_1FFCH	N/A	-	-	将来の使用のため予約

備考1. “R/W”フィールド内で，
“W”は，“書き込み可能”を意味します。
“R”は，“読み取り可能”を意味します。
“RC”は，“読み取りクリア”を意味します。
“-”は，“アクセス不可能”を意味します。

2. すべての内部レジスタは，32ビット・ワード配列のレジスタです。
3. 内部レジスタに対するバースト・アクセスは，禁止されています。バースト・アクセスすると，NSRのIRERRビットがセットされ，NMIがVR4120Aに対してアサートします。
4. 予約領域に対してリード・アクセスすると，NSRレジスタのCBERRビットがセットされ，SysCMD［0］にデータ・エラー・ビットが設定されたダミー読み取り応答データが読み出されます（一部の非公開レジスタを除きます）。
5. 予約領域に対してライト・アクセスすると，NSRレジスタのCBERRビットがセットされ，書き込みデータが失われます。
6. “アクセス”フィールド内で，
“W”は，ワード・アクセスが有効であることを意味します。
“H”は，ハーフ・ワード・アクセスが有効であることを意味します。
“B”は，バイト・アクセスが有効であることを意味します。
7. リード・オンリー・レジスタに対するライト・アクセスはエラーになりませんが，書き込みデータは失われます。
8. CPUはすべての内部レジスタにアクセスできますが，IBUSマスタ・デバイスはそれらのレジスタにアクセスできません。

保守／廃止

6. 2. 2　U_GMR（USBジェネラル・モード・レジスタ）

このレジスタは，USBコントローラの動作を設定するために使用します。RRビットを除く下位16ビットは，デバイスの初期化中にのみ書き込みできます。これらのビットの値を，送信または受信中に変更すると，USBコントローラは予測できない動作をすることがあります。

ビット	フィールド	R/W	ディフォールト	説　明
31：24	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
23	VT	R/W	0	デバイス・アドレス有効タイミング： このビットを1にセットすると，デバイス・アドレスFAはただちに有効になります。 このビットを0にセットすると，デバイス・アドレスFAは，USBコントローラがエンドポイント0で後続のACKパケットを受け取ったあとに，有効になります。
22：16	FA	R/W	0	デバイス・アドレス： USBホストから割り当てられたμPD98502内のUSBのデバイス・アドレスを格納するレジスタです。これは，USBコンフィギュレーション・プロセスの実行結果としてホストPCによって割り当てられます。 VR4120Aは，割り当てられたアドレスをこのレジスタにセットしなければなりません。
15：8	SOFINTVL	R/W	18H	SOFインターバル： この値は，SOFパケットが許容できるスキューを定義するために使用します。ディフォールト値は18Hです。00Hをセットすると，USBコントローラは2つの連続したSOFパケット間のタイミングを無視します。
7：3	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
2	AU	R/W	0	フレーム・ナンバ自動更新： フレーム・ナンバの自動更新イネーブルです。このビットを0にセットすると，受信SOFパケット中のフレーム・ナンバ・フィールドのデータが，U_VER内のUFNRフィールドに書き込まれます。1にセットした場合は，UFNRをインクリメントします。詳細については，6. 8を参照してください。
1	LE	R/W	0	ループバック・イネーブル： 内部ループバック・モードをイネーブルにするビットです。このビットを1にセットすると，USBコントローラはループバック・モードで動作します。ループバック・モードをセットすると，内部DMAコントローラのテストが可能となります。実際のUSBライン上ではUSBパケットの送受信は行われません。 ループバック・モードの詳細については，6. 9を参照してください。
0	RR	R/W	0	リモート・リジューム： リモート・リジュームを行うときに，VR4120Aはこのビットをセットします。このビットを1にセットすると，USBコントローラはリジューム信号をUSBホストに対して5 msの間送信します。リジューム信号送出が終了すると，このビットは自動的に0にリセットされます。

保守／廃止

6. 2. 3　U_VER（USBフレーム・ナンバ／バージョン・レジスタ）

USBの現在のフレーム・ナンバとUSBコントローラ・ブロックのバージョンを格納するレジスタです。

ビット	フィールド	R/W	ディフォールト	説　明
31：16	UVER	R	0201H	USBバージョン： USBコントローラ・ブロックの改訂ナンバをこのレジスタに格納します。LSIの規格によって異なりますので注意してください。 規格：UVER K：0201H
15：11	Reserved	R	0	将来の使用のため予約
10：0	UFNR	R	0	USBフレーム・ナンバ： U_GMRのAUビットが1の場合，SOFパケット受信ごとに，USBのフレーム・ナンバを格納します。現在のフレーム・ナンバを知ることができます。

保守／廃止

6. 2. 4　U_GSR1（USBジェネラル・ステータス・レジスタ1）

このレジスタは，USBコントローラの状態を示します。

ビット	フィールド	R/W	ディフォールト	説　明
31	GSR2	RC	0	ジェネラル・ステータス・レジスタ2内のいずれかのビットが1にセットされ，かつ，割り込みマスク・レジスタ2の対応するビットが1にセットされている場合，このGSR2ビットが1にセットされます。
30：24	Reserved	R	0	将来の使用のため予約
23	TMF	RC	0	Txメールボックス・フル： 送信メールボックス領域がフルであることを示すビットです。USB Txメールボックス・リード・アドレス（U_TMRA）とUSB Txメールボックス・ライト・アドレス（U_TMWA）が等しくなると，このビットが1にセットされます。VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
22	RMF	RC	0	Rxメールボックス・フル： 受信メールボックス領域がフルであることを示すビットです。USB Rxメールボックス・リード・アドレス（U_RMRA）とUSB Rxメールボックス・ライト・アドレス（U_RMWA）が等しくなると，このビットが1にセットされます。VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
21	RPE2	RC	0	Rxプール2エンプティ： 受信プール2が空であることを示すビットです。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
20	RPE1	RC	0	Rxプール1エンプティ： 受信プール1が空であることを示すビットです。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
19	RPE0	RC	0	Rxプール0エンプティ： 受信プール0が空であることを示すビットです。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
18	RPA2	RC	0	Rxプール2アラート： 受信プール2に残っているバッファ・ディレクトリの数がRxプール2情報レジスタ（U_RP2IR）のALフィールド値の4倍以下になると，このビットが1にセットされます。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
17	RPA1	RC	0	Rxプール1アラート： 受信プール1に残っているバッファ・ディレクトリの数がRxプール1情報レジスタ（U_RP1IR）のALフィールド値の4倍以下になると，このビットが1にセットされます。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。

保守／廃止

ビット	フィールド	R/W	ディフォールト	説　明
16	RPA0	RC	0	Rxプール0アラート： 受信プール0に残っているバッファ・ディレクトリの数がRxプール0情報レジスタ（U_RP0IR）のALフィールド値の4倍以下になると，このビットが1にセットされます。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
15：11	Reserved	R	0	将来の使用のため予約
10	DER	RC	0	DMAエラー： DMA転送中にエラーが発生したことを示すビットです。DMA転送中に内部バスでエラーが発生すると，このビットが1にセットされます。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
9	EP2FO	RC	0	EP2 FIFOエラー： エンドポイント2（アイソクロノスOUT）のFIFOにオーバランが発生したことを示すビットです。エンドポイント2がトランザクションを実行中にFIFOがフルになると，USBコントローラはデータを受信できなくなり後続のデータはすべて廃棄されます。このエラーが発生すると，このビットが1にセットされます。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
8	EP1FU	RC	0	EP1 FIFOエラー： エンドポイント1（アイソクロノスIN）のFIFOにアンダランが発生したことを示すビットです。エンドポイント1がトランザクションを実行中にFIFOが空になると，このビットが1にセットされます。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
7	EP6RF	RC	0	EP6 Rx終了： エンドポイント6（インタラプトOUT）がデータ・セグメントの受信を完了しRx表示を発行したことを示すビットです。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
6	EP5TF	RC	0	EP5 Tx終了： エンドポイント5（インタラプトIN）がデータ・セグメントの送信を完了しTx表示を発行したことを示すビットです。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
5	EP4RF	RC	0	EP4 Rx終了： エンドポイント4（バルクOUT）がデータ・セグメントの受信を完了しRx表示を発行したことを示すビットです。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
4	EP3TF	RC	0	EP3 Tx終了： エンドポイント3（バルクIN）がデータ・セグメントの送信を完了しTx表示を発行したことを示すビットです。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。

保守／廃止

ビット	フィールド	R/W	ディフォールト	説　明
3	EP2RF	RC	0	EP2 Rx終了： エンドポイント2（アイソクロノスOUT）がデータ・セグメントの受信を完了しRx表示を発行したことを示すビットです。 このビットがセットされるタイミングは，Rxモードによって変わります。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
2	EP1TF	RC	0	EP1 Tx終了： エンドポイント1（アイソクロノスIN）がデータ・セグメントの送信を完了しTx表示を発行したことを示すビットです。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
1	EP0RF	RC	0	EP0 Rx終了： エンドポイント0（コントロールOUT）がデータ・セグメントの受信を完了しRx表示を発行したことを示すビットです。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。
0	EP0TF	RC	0	EP0 Tx終了： エンドポイント0（コントロールIN）がデータ・セグメントの送信を完了しTx表示を発行したことを示すビットです。 VR4120Aがこのレジスタを読み込むと，このビットは0にリセットされます。

保守／廃止

6. 2. 5　U_IMR1（USB割り込みマスク・レジスタ1）

このレジスタは，割り込みをマスクするために使用します。

このレジスタのビットを1にセットしている場合に，USBジェネラル・ステータス・レジスタ1（アドレス：10H）の対応するビットが1にセットされると，割り込みを発行します。

ビット	フィールド	R/W	ディフォールト	説　明
31	GSR2	R/W	0	ジェネラル・ステータス・レジスタ2割り込み： 1 = マスク解除 0 = マスク
30：24	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
23	TMF	R/W	0	Txメールボックス・フル： 1 = マスク解除 0 = マスク
22	RMF	R/W	0	Rxメールボックス・フル： 1 = マスク解除 0 = マスク
21	RPE2	R/W	0	Rxプール2エンプティ： 1 = マスク解除 0 = マスク
20	RPE1	R/W	0	Rxプール1エンプティ： 1 = マスク解除 0 = マスク
19	RPE0	R/W	0	Rxプール0エンプティ： 1 = マスク解除 0 = マスク
18	RPA2	R/W	0	Rxプール2アラート： 1 = マスク解除 0 = マスク
17	RPA1	R/W	0	Rxプール1アラート： 1 = マスク解除 0 = マスク
16	RPA0	R/W	0	Rxプール0アラート： 1 = マスク解除 0 = マスク
15：11	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
10	DER	R/W	0	DMAエラー： 1 = マスク解除 0 = マスク
9	EP2FO	R/W	0	EP2 FIFOエラー： 1 = マスク解除 0 = マスク
8	EP1FU	R/W	0	EP1 FIFOエラー： 1 = マスク解除 0 = マスク
7	EP6RF	R/W	0	EP6 Rx終了： 1 = マスク解除 0 = マスク

保守／廃止

ビット	フィールド	R/W	ディフォールト	説　明
6	EP5TF	R/W	0	EP5 Tx終了： 1 = マスク解除 0 = マスク
5	EP4RF	R/W	0	EP4 Rx終了： 1 = マスク解除 0 = マスク
4	EP3TF	R/W	0	EP3 Tx終了： 1 = マスク解除 0 = マスク
3	EP2RF	R/W	0	EP2 Rx終了： 1 = マスク解除 0 = マスク
2	EP1TF	R/W	0	EP1 Tx終了： 1 = マスク解除 0 = マスク
1	EP0RF	R/W	0	EP0 Rx終了： 1 = マスク解除 0 = マスク
0	EP0TF	R/W	0	EP0 Tx終了： 1 = マスク解除 0 = マスク

保守／廃止

6. 2. 6　U_GSR2（USBジェネラル・ステータス・レジスタ2）

このレジスタは，USBコントローラの現在の状態を示します。このレジスタを読み込むと，レジスタのビットはすべてクリアされます。

ビット	フィールド	R/W	ディフォールト	説　明
31：22	Reserved	R	0	将来の使用のため予約
21	FW	RC	0	フレーム・ナンバ書き込み： フレーム・ナンバがUSBフレーム・ナンバ／バージョン・レジスタ（04H）に書き込まれると，このビットは1にセットされます。
20	IFN	RC	0	不正フレーム・ナンバ： USBコントローラが不正フレーム・ナンバかCRC/ビット・スタッフ・エラーを含んだSOFパケットを受信すると，このビットを1にセットします。
19	IEA	RC	0	不正エンドポイント・アクセス： USBコントローラが不正エンドポイント・ナンバを含んだINまたはOUTトークンを受信すると，このビットを1にセットします。
18	URSM	RC	0	USBリジューム： USBコントローラがリジューム信号をホストPCから受信すると，このビットを1にセットします。
17	URST	RC	0	USBリセット： USBコントローラがリセット信号をホストPCから受信すると，このビットを1にセットします。
16	USPD	RC	0	USBサスペンド： USBがサスペンド状態になったことをUSBコントローラが検出すると，このビットを1にセットします。
15：8	Reserved	R	0	将来の使用のため予約
7	EP2OS	RC	0	エンドポイント2のオーバ・サイズ： 受信データ・サイズがEP2の最大パケット・サイズ（1023バイト）を越えると，このビットを1にセットします。
6	EP2ED	RC	0	エンドポイント2で余分なデータを検出： 余分なデータ・パケットがアイソクロノス・エンドポイント（EP2）で検出されると，このビットを1にセットします。U_EP2CRのEP2ENビットが0にセットされた場合，USBコントローラが余分なデータをEP2で検出しても，このビットはセットされません。
5	EP2ND	RC	0	エンドポイント2でアイソクロノス・データが消失： EP2のアイソクロノスOUT転送においてデータが消失したことを示すビットです。U_EP2CRのEP2ENビットが0にセットされている場合，USBコントローラがデータの消失をEP2で検出しても，このビットはセットされません。詳細については6. 6. 8を参照してください。
4	EP1NT	RC	0	エンドポイント1にトークンなし： EP1上において，INトークン・パケットが2つのSOF間（1フレーム期間内）に受信されないときに，このビットを1にセットします。U_EP1CRのEP1ENビットが0にセットされている場合，USBコントローラがEP1上でトークン・パケットが受信されないことを検出しても，このビットはセットされません。

保守／廃止

ビット	フィールド	R/W	ディフォールト	説　明
3	EP1ET	RC	0	エンドポイント1の余分なトークン： EP1上において，2つのSOF間（1フレーム期間内）に2つ以上のINトークン・パケットが受信されたときに，このビットを1にセットします。U_EP1CRのEP1ENビットが0にセットされている場合，USBコントローラが余分なトークン・パケットをEP1上で検出しても，このビットはセットされません。
2	EP1ND	RC	0	エンドポイント1にデータなし： INトークン・パケットが来てもデータがEP1でレディ状態にない（送信するパケットがない）ときに，このビットを1にセットします。 U_EP1CRのEP1ENビットが0にセットされている場合，USBコントローラがEP1でデータがない状態を検出しても，このビットはセットされません。
1	ES	RC	0	余分なSOF： 余分なSOFパケットを検出すると，このビットを1にセットします。詳細については6. 8を参照してください。
0	SL	RC	0	SOFのロス： USBコントローラが1フレーム期間ごとにSOFパケットを受信しないと，このビットを1にセットします。詳細については6. 8を参照してください。

保守／廃止

6. 2. 7　U_IMR2（USB割り込みマスク・レジスタ2）

このレジスタは，割り込みをマスクするために使用します。

このレジスタのビットを1にセットしている場合に，USBジェネラル・ステータス・レジスタ2（アドレス：18H）の対応するビットが1にセットされると，U_GSR1のGSR2ビットが1にセットされます。

ビット	フィールド	R/W	ディフォールト	説　明
31：22	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
21	FW	R/W	0	フレーム・ナンバ書き込み： 1 = マスク解除 0 = マスク
20	IFN	R/W	0	不正フレーム・ナンバ： 1 = マスク解除 0 = マスク
19	IEA	R/W	0	不正エンドポイント・アクセス： 1 = マスク解除 0 = マスク
18	URSM	R/W	0	USBリジューム： 1 = マスク解除 0 = マスク
17	URST	R/W	0	USBリセット： 1 = マスク解除 0 = マスク
16	USPD	R/W	0	USBサスペンド： 1 = マスク解除 0 = マスク
15：8	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
7	EP2OS	R/W	0	エンドポイント2のオーバ・サイズ： 1 = マスク解除 0 = マスク
6	EP2ED	R/W	0	エンドポイント2で余分なデータを検出： 1 = マスク解除 0 = マスク 詳細については6. 6. 7を参照してください。
5	EP2ND	R/W	0	エンドポイント2でアイソクロノス・データが消失： 1 = マスク解除 0 = マスク 詳細については6. 6. 7を参照してください。
4	EP1NT	R/W	0	エンドポイント1にトークンなし： 1 = マスク解除 0 = マスク
3	EP1ET	R/W	0	エンドポイント1の余分なトークン： 1 = マスク解除 0 = マスク
2	EP1ND	R/W	0	エンドポイント1にデータなし： 1 = マスク解除 0 = マスク

保守／廃止

ビット	フィールド	R/W	ディフォールト	説　明
1	ES	R/W	0	余分なSOF： 1 = マスク解除 0 = マスク
0	SL	R/W	0	SOFのロス： 1 = マスク解除 0 = マスク

6. 2. 8　U_EP0CR（USB EP0制御レジスタ）

このレジスタは，エンドポイント0の動作を設定するために使用します。

MAXPフィールドの値を送受信動作中に再度書き込みすると，USBコントローラは予測できない動作となることがあります。したがって，MAXPは初期設定が行われている間のみ書き込んでください。

ビット	フィールド	R/W	ディフォールト	説　明
31	EP0EN	R/W	0	エンドポイント0イネーブル： VR4120Aがこのビットを1にセットすると，エンドポイント0はUSBホストとデータの送受信ができます。
30：21	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
20	ISS	R/W	0	IN送信ストール： VR4120Aがこのビットを1にセットすると，データ・フェーズでSTALLパケットを送信します。 SETUPパケットを受信すると，このビットは0になります。
19	INAK	R/W	0	IN NAK： VR4120Aがこのビットを1にセットすると，データ・フェーズでNAKパケットを送信します。
18	OSS	R/W	0	OUT送信ストール： VR4120Aがこのビットを1にセットすると，ハンドシェーク・フェーズでSTALLハンドシェークを行います。 SETUPパケットを受信すると，このビットは0になります。
17	NHSK0	R/W	0	ハンドシェークなし： VR4120Aがこのビットを1にセットすると，ハンドシェーク・フェーズでハンドシェークを行いません。
16	ONAK	R/W	0	OUT NAK： VR4120Aがこのビットを1にセットすると，ハンドシェーク・フェーズでNAKハンドシェークを行います。
15：7	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
6：0	MAXP0	R/W	0	最大パケット・サイズ： エンドポイント0に対する1つのトランザクションで送受信を行う最大パケット・サイズを指定します。USBトランザクションを開始する前に，VR4120Aは適切な値をこのレジスタにセットしなければなりません。USB1.1の規格では，コントロール転送は8, 16, 32, 64バイトのいずれかを選択することになっています。

備考　コントロール転送のデータ・ステージまたはステータス・ステージでコントロール・エンドポイントによってSTALLハンドシェークを送信すると，SETUP PIDを受信するまで，そのエンドポイントに対するすべてのアクセスにSTALLハンドシェークを返さなければなりません。後続のSETUP PID（USB1.1仕様による）を受信後に，エンドポイントにSTALLハンドシェークを返す必要はありません。

保守／廃止

6. 2. 9　U_EP1CR（USB EP1制御レジスタ）

このレジスタは，エンドポイント1の動作を設定するために使用します。

MAXPフィールドの値を送信動作中に再度書き込みすると，USBコントローラは予測できない動作となることがあります。したがって，MAXPは初期設定が行われている間のみ書き込んでください。

ビット	フィールド	R/W	ディフォールト	説　明
31	EP1EN	R/W	0	エンドポイント1イネーブル： VR4120Aがこのビットを1にセットすると，エンドポイント1はデータを送信できます。
30：20	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
19	TM1	R/W	0	Txモード： 送信モードを設定するためのビットです。 このビットを0にセットすると，送信をSZLPモードで行います。 このビットを1にセットすると，送信をNZLPモードで行います。 送信モードの詳細については，6. 5. 3を参照してください。
18：10	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
9：0	MAXP1	R/W	0	最大パケット・サイズ： エンドポイント1に対する1つのトランザクションで送信を行う最大パケット・サイズを指定します。USBトランザクションを開始する前に，VR4120Aは適切な値をこのレジスタにセットしなければなりません。 USB1.1の規格では，アイソクロノス転送は0-1023バイトの範囲で選択することになっています。

保守／廃止

6. 2. 10　U_EP2CR（USB EP2制御レジスタ）

このレジスタは，エンドポイント2の動作を設定するために使用します。

MAXPフィールドの値を受信動作中に再度書き込みすると，USBコントローラは予測できない動作となることがあります。したがって，MAXPは初期設定が行われている間のみ書き込んでください。

ビット	フィールド	R/W	ディフォールト	説　明
31	EP2EN	R/W	0	エンドポイント2イネーブル： VR4120Aがこのビットを1にセットすると，エンドポイント2はデータを受信できます。
30：21	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
20：19	RM2	R/W	00	Rxモード： 受信モードを設定するためのビットです。 00または01　：通常モード 10　　：アセンブル・モード 11　　：セパレート・モード 受信モードの詳細については，6. 6. 4を参照してください。
18：10	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
9：0	MAXP2	R/W	0	最大パケット・サイズ： エンドポイント2に対する1つのトランザクションで受信する最大パケット・サイズを指定します。USBトランザクションを開始する前に，VR4120Aは適切な値をこのレジスタにセットしなければなりません。USB1.1の規格では，アイソクロノス転送は0-1023バイトの範囲で選択することになっています。

備考　通常モードでは，Rx表示が受信パケットごとに報告されるため，パケットごとにエラー・ステータスが通知されます。その他のモードでは，エラー・ステータスはパケットごとではなく，全受信パケットに対して通知されます。

保守／廃止

6. 2. 11　U_EP3CR（USB EP3制御レジスタ）

このレジスタは，エンドポイント3の動作を設定するために使用します。

MAXPフィールドの値を送信動作中に再度書き込みすると，USBコントローラは予測できない動作となることがあります。したがって，MAXPは初期設定が行われている間のみ書き込んでください。

ビット	フィールド	R/W	ディフォールト	説　明
31	EP3EN	R/W	0	エンドポイント3イネーブル： VR4120Aがこのビットを1にセットすると，エンドポイント3はデータを送信できます。
30：20	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
19	TM3	R/W	0	Txモード： 送信モードを設定するためのビットです。 このビットを0にセットすると，送信をSZLPモードで行います。 このビットを1にセットすると，送信をNZLPモードで行います。 送信モードの詳細については，6. 5. 3を参照してください。
18	SS3	R/W	0	送信ストール： VR4120Aがこのビットを1にセットすると，エンドポイント3からSTALLパケットを送信します。
17	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
16	NAK3	R/W	0	VR4120Aがこのビットを1にセットすると，エンドポイント3からNAKパケットを送信します。
15：7	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
6：0	MAXP3	R/W	0	最大パケット・サイズ エンドポイント3に対する1つのトランザクションで送信する最大パケット・サイズを指定します。USBトランザクションを開始する前に，VR4120Aは適切な値をこのレジスタにセットしなければなりません。 USB1.1の規格では，バルク転送は8, 16, 32, 64バイトのいずれかを選択することになっています。

保守／廃止

6. 2. 12　U_EP4CR（USB EP4制御レジスタ）

このレジスタは，エンドポイント4の動作を設定するために使用します。

MAXPフィールドの値を受信動作中に再度書き込みすると，USBコントローラは予測できない動作となることがあります。したがって，MAXPは初期設定が行われている間のみ書き込んでください。

ビット	フィールド	R/W	ディフォールト	説　明
31	EP4EN	R/W	0	エンドポイント4イネーブル： VR4120Aがこのビットを1にセットすると，エンドポイント4はデータを受信できます。
30：21	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
20：19	RM4	R/W	0	Rxモード： 受信モードを設定するためのビットです。 00または01　：通常モード 10　　：アセンブル・モード 11　　：セパレート・モード 受信モードの詳細については，6. 6. 4を参照してください。
18	SS4	R/W	0	送信ストール： VR4120Aがこのビットを1にセットすると，エンドポイント4でSTALLハンドシェークを行います。
17	NHSK4	R/W	0	ハンドシェークなし： VR4120Aがこのビットを1にセットすると，エンドポイント4でハンドシェークを行いません。
16	NAK4	R/W	0	VR4120Aがこのビットを1にセットすると，エンドポイント4でNAKハンドシェークを行います。
15：7	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
6：0	MAXP4	R/W	0	最大パケット・サイズ： エンドポイント4に対する1つのトランザクションで受信する最大パケット・サイズを指定します。USBトランザクションを開始する前に，VR4120Aは適切な値をこのレジスタにセットしなければなりません。USB1.1の規格では，バルク転送は8, 16, 32, 64バイトのいずれかを選択することになっています。

保守／廃止

6. 2. 13　U_EP5CR（USB EP5制御レジスタ）

このレジスタは，エンドポイント5の動作を設定するために使用します。

MAXPフィールドの値を送信動作中に再度書き込みすると，USBコントローラは予測できない動作となることがあります。したがって，MAXPは初期設定が行われている間のみ書き込んでください。

ビット	フィールド	R/W	ディフォールト	説　明
31	EP5EN	R/W	0	エンドポイント5イネーブル： VR4120Aがこのビットを1にセットすると，エンドポイント5はデータを送信できます。
30：20	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
19	FM	R/W	0	フィードバック・モード： VR4120Aがこのビットを1にセットすると，エンドポイント5はフィードバック・モードで動作します（フィードバック・モードの詳細については，USB仕様1.1を参照してください）。
18	SS5	R/W	0	送信ストール： VR4120Aがこのビットを1にセットすると，エンドポイント5でSTALLハンドシェークを行います。
17	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
16	NAK5	R/W	0	VR4120Aがこのビットを1にセットすると，エンドポイント5でNAKハンドシェークを行います。
15：7	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
6：0	MAXP5	R/W	0	最大パケット・サイズ： エンドポイント5に対する1つのトランザクションで送信する最大パケット・サイズを指定します。USBトランザクションを開始する前に，VR4120Aは適切な値をこのレジスタにセットしなければなりません。USB1.1の規格では，インタラプト転送は0-64バイトの範囲で選択することになっています。

保守／廃止

6. 2. 14　U_EP6CR（USB EP6制御レジスタ）

このレジスタは，エンドポイント6の動作を設定するために使用します。

MAXPフィールドの値を受信動作中に再度書き込みすると，USBコントローラは予測できない動作となることがあります。したがって，MAXPは初期設定が行われている間のみ書き込んでください。

ビット	フィールド	R/W	ディフォールト	説　明
31	EP6EN	R/W	0	エンドポイント6イネーブル： VR4120Aがこのビットを1にセットすると，エンドポイント6はデータを受信できます。
30：19	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
18	SS6	R/W	0	送信ストール： VR4120Aがこのビットを1にセットすると，エンドポイント6でSTALLハンドシェークを行います。
17	NHSK6	R/W	0	ハンドシェークなし： VR4120Aがこのビットを1にセットすると，エンドポイント6でハンドシェークを行いません。
16	NAK6	R/W	0	VR4120Aがこのビットを1にセットすると，エンドポイント6でNAKハンドシェークを行います。
15：7	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
6：0	MAXP6	R/W	0	最大パケット・サイズ： エンドポイント6に対する1つのトランザクションで受信する最大パケット・サイズを指定します。USBトランザクションを開始する前に，VR4120Aは適切な値をこのレジスタにセットしなければなりません。USB1.1の規格では，インタラプト転送は0-64バイトの範囲で選択することになっています。

保守／廃止

6. 2. 15　U_CMR（USBコマンド・レジスタ）

このレジスタは，Txリクエストを発行したりRxバッファ・ディレクトリをプールに追加するために使用します。

VR4120Aは，コマンドをこのレジスタに書き込みます。

Bビット（ビット31）がセットされると，VR4120Aがコマンドをこのレジスタに書き込んでも，そのコマンドは無効となります。

ビット	フィールド	R/W	ディフォールト	説　　明
31	B	R/W	0	ビジィ： 発行したコマンドの処理が終了したかどうかを示すビットです。コマンドの実行が完了していないと，このビットが1にセットされます。コマンドの実行が完了すると，このビットを0にセットします。 VR4120Aが1つのコマンドの直後にさらにコマンドを発行しようとするときは，このビットが0にセットされていることを確認する必要があります。
30：27	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
26：24	コマンド	R/W	000	コマンドの種類を指定するフィールドです。USBコントローラの内部処理は，このフィールドに書き込まれた値によって変わります。 000：エンドポイント0でデータの送信 001：エンドポイント1でデータの送信 010：エンドポイント3でデータの送信 011：エンドポイント5でデータの送信 100：バッファ・ディレクトリをプール0に追加 101：バッファ・ディレクトリをプール1に追加 110：バッファ・ディレクトリをプール2に追加 111：設定禁止
23：16	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
15：0	データ・サイズ/NOD	R/W	0	バッファ・ディレクトリのデータ・サイズ／ナンバ： このフィールドの意味は，コマンド・フィールドに書き込まれた値によって変わります。 コマンド = 0xx：VR4120Aは送信データのサイズをこのフィールドに書き込まなければなりません。 コマンド = 100, 101, 110：プールに追加するバッファ・ディレクトリの数を示します。

保守／廃止

6. 2. 16　U_CA（USBコマンド拡張レジスタ）

このレジスタは，Txリクエストを発行したりRxバッファ・ディレクトリをプールに追加するために使用します。

VR4120Aは，U_CMRを発行する前にTxまたはRxバッファ・ディレクトリのスタート・アドレスをこのレジスタに書き込みます。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	アドレス	R/W	0	このフィールドの意味は，USBコマンド・レジスタのコマンド・フィールドに書き込まれた値によって変わります。 コマンド = 0xx：送信パケットのスタート・アドレス コマンド = 100, 101, 110：受信プールに追加するバッファ・ディレクトリのスタート・アドレス

保守／廃止

6. 2. 17　U_TEPSR（USB Tx**エンドポイント・ステータス・レジスタ**）

このレジスタは，データ送信に使用されているエンドポイントの状態を知るために使用します。

ビット	フィールド	R/W	ディフォールト	説　明
31：26	Reserved	R	0	**将来の使用のため予約**
25：24	EP5TS	R	0	EP5 Tx**ステータス：** **エンドポイント**5**の送信状態を示すレジスタです。** **このレジスタは，リードしてもクリアされません。** 00：**送信するようにスケジュールされたデータはありません（アイドル）。** 01：**送信するようにスケジュールされたデータ・セグメントが**1**つあります。** 10：**送信するようにスケジュールされたデータ・セグメントが**2**つ以上あります（ビジィ）。**
23：18	Reserved	R	0	**将来の使用のため予約**
17：16	EP3TS	R	0	EP3 Tx**ステータス：** **エンドポイント**3**の送信状態を示すレジスタです。** **このレジスタは，リードしてもクリアされません。** 00：**送信するようにスケジュールされたデータはありません（アイドル）。** 01：**送信するようにスケジュールされたデータ・セグメントが**1**つあります。** 10：**送信するようにスケジュールされたデータ・セグメントが**2**つ以上あります（ビジィ）。**
15：10	Reserved	R	0	**将来の使用のため予約**
9：8	EP1TS	R	0	EP1 Tx**ステータス：** **エンドポイント**1**の送信状態を示すレジスタです。** **このレジスタは，リードしてもクリアされません。** 00：**送信するようにスケジュールされたデータはありません（アイドル）。** 01：**送信するようにスケジュールされたデータ・セグメントが**1**つあります。** 10：**送信するようにスケジュールされたデータ・セグメントが**2**つ以上あります（ビジィ）。**
7：2	Reserved	R	0	**将来の使用のため予約**
1：0	EP0TS	R	0	EP0 Tx**ステータス：** **エンドポイント**0**の送信状態を表示するレジスタです。** **このレジスタは，リードしてもクリアされません。** 00：**送信するようにスケジュールされたデータはありません（アイドル）。** 01：**送信するようにスケジュールされたデータ・セグメントが**1**つあります。** 10：**送信するようにスケジュールされたデータ・セグメントが**2**つ以上あります（ビジィ）。**

保守／廃止

6. 2. 18　U_RP0IR（USB Rxプール0情報レジスタ）

このレジスタは，受信プール0の情報を示します。

デバイスが初期化されている間のみ，VR4120Aはこのレジスタに書き込んでください。

ビット	フィールド	R/W	ディフォールト	説　明
31	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
30：28	AL	R/W	000	アラート・レベル： プール0に対する警告レベルを設定します。プール0に残っているバッファ・ディレクトリの数がこのフィールドに設定された値の4倍以下になると，USBコントローラは，USBジェネラル・ステータス・レジスタ1のRPA0ビットを1にセットします。 このフィールドへの設定値Nを書き込むことで，N×4がアラート・レベルとなります（バッファ・ディレクトリの残り数 = 4, 8, 12, ..., 28）。 000をこのフィールドに書き込むと，この機能はディスエーブルとなりVR4120Aには何も通知されません。
27：16	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
15：0	RNOD	R	0	バッファ・ディレクトリの残り数： プール0に残っているバッファ・ディレクトリの数を示します。 VR4120Aは，このフィールドを読み込むことのみできます。 バッファ・ディレクトリの追加には，USBコマンド・レジスタを使用してください。

6. 2. 19　U_RP0AR（USB Rxプール0アドレス・レジスタ）

このレジスタは，現在使用中のバッファ・ディレクトリのスタート・アドレスを示します。

Rxプールの設定方法は，6. 6. 3　**受信プールの設定**を参照してください。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	アドレス	R	0	バッファ・ディレクトリ・アドレス： プール0の最初のバッファ・ディレクトリのスタート・アドレスを示すレジスタです。 VR4120Aは，このレジスタを読み込むことのみできます。 バッファ・ディレクトリの追加には，USBコマンド・レジスタを使用してください。

保守／廃止

6. 2. 20　U_RP1IR（USB Rxプール1情報レジスタ）

このレジスタは，受信プール1の情報を示します。

デバイスが初期化されている間にのみ，VR4120Aはこのレジスタに書き込みます。

ビット	フィールド	R/W	ディフォールト	説　明
31	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
30：28	AL	R/W	000	アラート・レベル： プール1に対する警告レベルを設定します。プール1に残っているバッファ・ディレクトリの数がこのフィールドに設定された値の4倍以下になると，USBコントローラは，USBジェネラル・ステータス・レジスタ1のRPA1ビットを1にセットします。 このフィールドへの設定値Nを書き込むことで，N×4がアラート・レベルとなります（バッファ・ディレクトリの残り数 = 4, 8, 12, ..., 28）。 000をこのフィールドに書き込むと，この機能はディスエーブルとなりVR4120Aには何も通知されません。
27：16	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
15：0	RNOD	R	0	バッファ・ディレクトリの残り数： プール1に残っているバッファ・ディレクトリの数を示します。 VR4120Aは，このフィールドを読み込むことのみできます。 バッファ・ディレクトリの追加には，USBコマンド・レジスタを使用してください。

6. 2. 21　U_RP1AR（USB Rxプール1アドレス・レジスタ）

このレジスタは，現在使用中のバッファ・ディレクトリのスタート・アドレスを示します。

Rxプールの設定方法は，6. 6. 3　**受信プールの設定**を参照してください。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	アドレス	R	0	バッファ・ディレクトリ・アドレス： プール1の最初のバッファ・ディレクトリのスタート・アドレスを示すレジスタです。 VR4120Aは，このレジスタを読み込むことのみできます。 バッファ・ディレクトリの追加には，USBコマンド・レジスタを使用してください。

保守／廃止

6. 2. 22　U_RP2IR（USB Rxプール2情報レジスタ）

このレジスタは，受信プール2の情報を示します。

デバイスが初期化されている間にのみ，VR4120Aはこのレジスタに書き込みます。

ビット	フィールド	R/W	ディフォールト	説　明
31	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
30：28	AL	R/W	000	アラート・レベル： プール2に対する警告レベルを設定します。プール2に残っているバッファ・ディレクトリの数がこのフィールドに設定された値の4倍以下になると，USBコントローラは，USBジェネラル・ステータス・レジスタ1のRPA2ビットを1にセットします。 このフィールドへの設定値Nを書き込むことで，N×4がアラート・レレベルとなります（バッファ・ディレクトリの残り数 = 4, 8, 12, ..., 28）。000をこのフィールドに書き込むと，この機能はディスエーブルとなりVR4120Aには何も通知されません。
27：16	Reserved	R/W	0	将来の使用のため予約。0を書き込んでください。
15：0	RNOD	R	0	バッファ・ディレクトリの残り数： プール2に残っているバッファ・ディレクトリの数を示します。 VR4120Aは，このフィールドを読み込むことのみできます。 バッファ・ディレクトリの追加には，USBコマンド・レジスタを使用してください。

6. 2. 23　U_RP2AR（USB Rxプール2アドレス・レジスタ）

このレジスタは，現在使用中のバッファ・ディレクトリのスタート・アドレスを示します。

Rxプールの設定方法は，6. 6. 3　**受信プールの設定**を参照してください。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	アドレス	R	0	バッファ・ディレクトリ・アドレス： プール2の最初のバッファ・ディレクトリのスタート・アドレスを示すレジスタです。 VR4120Aは，このレジスタを読み込むことのみできます。 バッファ・ディレクトリの追加には，USBコマンド・レジスタを使用してください。

6. 2. 24　U_TMSA（USB Txメールボックス・スタート・アドレス・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：0	アドレス	R/W	0	送信メールボックス領域のスタート・アドレスを示すレジスタです。 VR4120Aは，初期化時にこのフィールドに値をセットしなければなりません。

保守／廃止

6. 2. 25　U_TMBA（USB Txメールボックス・ボトム・アドレス・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：0	アドレス	R/W	0	送信メールボックス領域の終了アドレスを示すレジスタです。VR4120Aは，初期化時にのみこのフィールドに値をセットしなければなりません。

6. 2. 26　U_TMRA（USB Txメールボックス・リード・アドレス・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：0	アドレス	R/W	0	VR4120AがTxメールボックスから次に読み込むアドレスを示すレジスタです。VR4120Aはメールボックスの内容を読み込んだあと，このレジスタの値を変更しなければなりません。

6. 2. 27　U_TMWA（USB Txメールボックス・ライト・アドレス・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：0	アドレス	R	0	USBコントローラがTxメールボックスに対して次にデータを書き込むアドレスを示すレジスタです。

6. 2. 28　U_RMSA（USB Rxメールボックス・スタート・アドレス・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：0	アドレス	R/W	0	受信メールボックス領域のスタート・アドレスを示すレジスタです。VR4120Aは，初期化時にのみこのフィールドに値をセットしなければなりません。

6. 2. 29　U_RMBA（USB Rxメールボックス・ボトム・アドレス・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：0	アドレス	R/W	0	受信メールボックス領域の終了アドレスを示すレジスタです。VR4120Aは，初期化時にのみこのフィールドに値をセットしなければなりません。

6. 2. 30　U_RMRA（USB Rxメールボックス・リード・アドレス・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：0	アドレス	R/W	0	VR4120AがRxメールボックスから次に読み込むアドレスを示すレジスタです。VR4120Aは，メールボックスの内容を読み込んだあと，このレジスタの値を変更しなければなりません。

保守／廃止

6. 2. 31　U_RMWA（USB Rxメールボックス・ライト・アドレス・レジスタ）

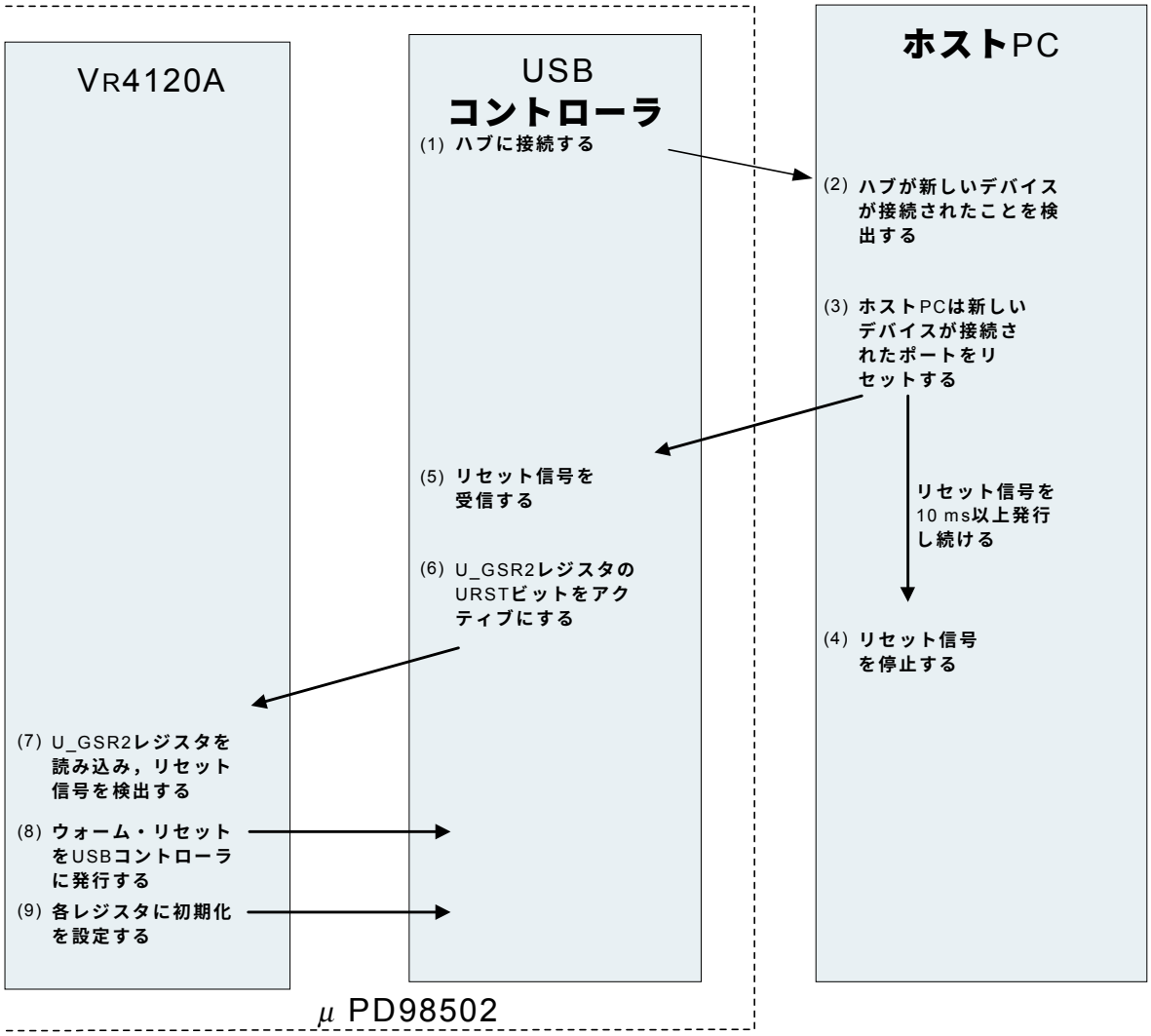
ビット	フィールド	R/W	ディフォールト	説　　明
31：0	アドレス	R	0	USBコントローラがRxメールボックスに対して次にデータを書き込むアドレスを示すレジスタです。

保守／廃止

6. 3　USB接続シーケンス

このセクションは，μPD98502をUSBハブに接続するときのシーケンスについて記述します。

図6－2　USB接続シーケンス



(1) μPD98502のUSBポートをUSBハブに接続します。

(2) 新しいデバイス（μPD98502）がハブに接続されたことをホストPCに通知します。

(3) ホストPCは，デバイスが接続されているポートをリセットするために，リセット信号を発行します。

(4) 10 msが経過すると，ホストPCはリセット信号の発行を停止します。

(5) ホストPCがリセット信号を発行したことをUSBコントローラに通知します。

(6) ホストPCからのリセット信号入力をVR4120Aに通知するため，U_GSR2レジスタのURSTビットがアクティブとなります。

(7) VR4120AはU_GSR2レジスタを読み込み，URSTビットがアクティブであることを検出すると，リセット信号が発行されたことを認識します。

(8) USBコントローラを初期化するために，VR4120Aはウォーム・リセットをUSBコントローラに発行しなければなりません。

(9) リセットが完了すると，USBコントローラは，USBコントローラ内部のレジスタに値を書き込んで初期化を行います。

保守／廃止

6.4 初 期 化

USBコントローラがUSBホストからリセットされたあとに，初期化のためにVR4120Aはいくつかのレジスタをセットしなければなりません。初期化シーケンスは次のとおりです。

(1) 使用したい動作モードをUSBジェネラル・モード・レジスタに設定します。

(2) 受信プールをシステム・メモリに置き，それらのプールが含む情報を以下のレジスタに設定します。

USB Rxプール0情報レジスタ　　：（アドレス：1000_1050H）
USB Rxプール0アドレス・レジスタ：（アドレス：1000_1054H）
USB Rxプール1情報レジスタ　　：（アドレス：1000_1058H）
USB Rxプール1アドレス・レジスタ：（アドレス：1000_105CH）
USB Rxプール2情報レジスタ　　：（アドレス：1000_1060H）
USB Rxプール2アドレス・レジスタ：（アドレス：1000_1064H）

(3) 送信／受信メールボックスをシステム・メモリに置き，それらのメールボックスが含む情報を以下のレジスタに設定します。

USB Txメールボックス・スタート・アドレス・レジスタ：（アドレス：1000_1070H）
USB Txメールボックス・ボトム・アドレス・レジスタ　：（アドレス：1000_1074H）
USB Rxメールボックス・スタート・アドレス・レジスタ：（アドレス：1000_1080H）
USB Rxメールボックス・ボトム・アドレス・レジスタ　：（アドレス：1000_1084H）

Txメールボックス・スタート・アドレス・レジスタを設定すると，レジスタに設定された値はTxメールボックス・リード・アドレス・レジスタ（アドレス：1000_1078H）とTxメールボックス・ライト・アドレス・レジスタ（アドレス：1000_107CH）にコピーされます。同様に，Rxメールボックス・スタート・アドレス・レジスタを設定すると，レジスタに設定された値はRxメールボックス・リード・アドレス・レジスタ（アドレス：1000_1088H）とRxメールボックス・ライト・アドレス・レジスタ（アドレス：1000_108CH）にコピーされます。

(4) USB EP0制御レジスタ，EP1-2制御レジスタ，EP3-4制御レジスタ，EP5-6制御レジスタのMAXPフィールドに値を書き込みます。次に，各エンドポイントをイネーブルとします。これまでの設定によって，データの送信／受信を開始し，ホストPCからのデバイス・コンフィギュレーションに応答できます。

(5) デバイス・コンフィギュレーションは，エンドポイント0を介して開始します。したがって，応答をエンドポイント0に返さなければなりません。この段階で，USBのアドレスが割り当てられます。VR4120Aは，その値をUSBジェネラル・モード・レジスタ（アドレス：1000_1000H）のアドレス・フィールド（FA）に設定しなければなりません。

初期化手順はこれで完了です。

上記に加えて，USB割り込みマスク・レジスタ（アドレス：1000_1014Hまたは1000_101CH）に値を書き込み，割り込みを許可する必要があります。

保守／廃止

6.4.1　受信プールの設定

受信プール設定の詳細については，6.6.3　**受信プールの設定**を参照してください。

6.4.2　送信／受信メールボックスの設定

USBコントローラはデータ・セグメントを送信後，Tx表示をシステム・メモリのメールボックスに書き込むことによりステータスを報告します。初期化時，VR4120Aはメールボックスの設定をしなければなりません。USBコントローラは，メールボックスをリング・バッファとして使用します。このバッファは，送信／受信それぞれに対して4つのレジスタを用いて設定します。

送信メールボックスを設定するレジスタ：

U_TMSA（USB Txメールボックス・スタート・アドレス・レジスタ：1000_1070H）
U_TMBA（USB Txメールボックス・ボトム・アドレス・レジスタ　：1000_1074H）
U_TMRA（USB Txメールボックス・リード・アドレス・レジスタ　：1000_1078H）
U_TMWA（USB Txメールボックス・ライト・アドレス・レジスタ　：1000_107CH）

受信メールボックスを設定するレジスタ：

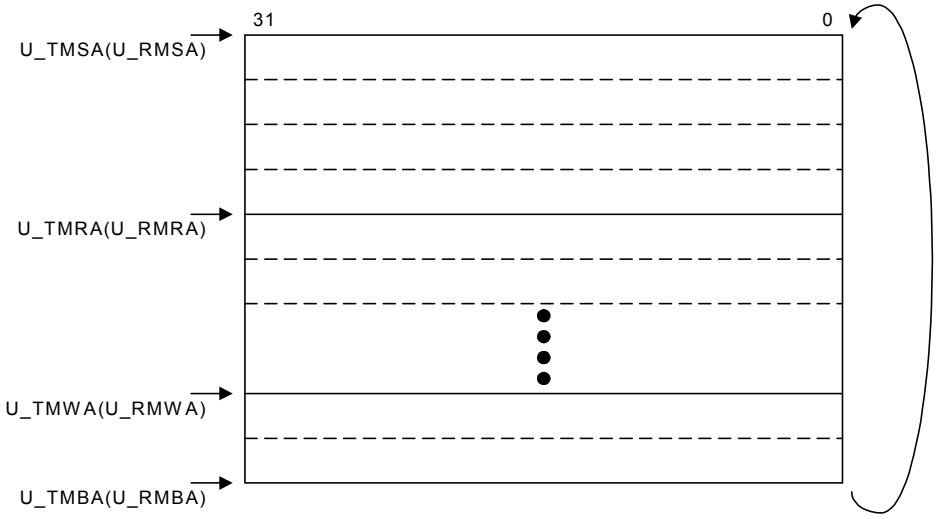
U_RMSA（USB Rxメールボックス・スタート・アドレス・レジスタ：1000_1080H）
U_RMBA（USB Rxメールボックス・ボトム・アドレス・レジスタ　：1000_1084H）
U_RMRA（USB Rxメールボックス・リード・アドレス・レジスタ　：1000_1088H）
U_RMWA（USB Rxメールボックス・ライト・アドレス・レジスタ　：1000_108CH）

備考1. VR4120Aはリセット後最初にU_TMSAに書き込む値を， U_TMRAとU_TMWAにも内部で自動的にコピーします。同様に，VR4120Aがリセット後最初にU_RMSAに書き込む値を，U_RMRAとU_RMWAにも内部で自動的にコピーします。

2. U_TMSAとU_TMBAおよびU_RMSAとU_RMBAに同じ値を設定しないでください。
3. メールボックスの設定に使用するレジスタのうち，VR4120AはU_TMRAとU_RMRAのみを更新してください。ほかのレジスタは，初期化時のみ書き込んでください。ほかのタイミングでは書き込まないでください。
4. 各Rx表示は，2ワードで構成されています。したがって，受信メールボックスのサイズは，2ワードの整数倍でなければなりません。
5. メールボックス領域は，ワード・アラインされていなければなりません。

システム・メモリのメールボックスのコンフィギュレーションを以下の図6－3に示します。

図6－3　メールボックス・コンフィギュレーション



USBコントローラが表示を書き込むと，ライト・ポインタ（U_TMWAまたはU_RMWA）をインクリメントします。USBコントローラが表示を書き込むたびに，対応するエンドポイントの送信／受信終了ビットをセットし，マスクされていなければ割り込みを発行します。

ライト・ポインタが，ボトム・アドレス（U_TMBAまたはU_RMBA）に達すると，スタート・アドレス（U_TMSAまたはU_RMSA）に強制的にジャンプさせます。USBコントローラは，VR4120Aがまだ読み取っていない表示へのオーバライトを防止するためにリード・ポインタ（U_TMRAまたはU_RMRA）を使用します。リード・ポインタ（U_TMRAまたはU_RMRA）は，VR4120Aによって管理されます。VR4120Aが表示をメールボックスから読み取るたびに，次に読み込むアドレスをリード・ポインタ・レジスタ（U_TMRAまたはU_RMRA）に書き込みます。

ライト・ポインタ（U_TMWAまたはU_RMWA）とリード・ポインタ（U_TMRAまたはU_RMRA）の両方が同じアドレスを示すと，USBコントローラは，メールボックスがフルの状態であると考えUSBジェネラル・ステータス・レジスタ1のTMFビット（送信メールボックス・フル）またはRMFビット（受信メールボックス・フル）をセットし，マスクされていなければ割り込みを発行します。

メールボックスのフル状態では，USBコントローラは次の表示の書き込みをしません。VR4120Aは，この状態から復帰するためにフルのメールボックスから表示を読み取り，リード・ポインタ（U_TMRAまたはU_RMRA）を更新しなければなりません。

保守／廃止

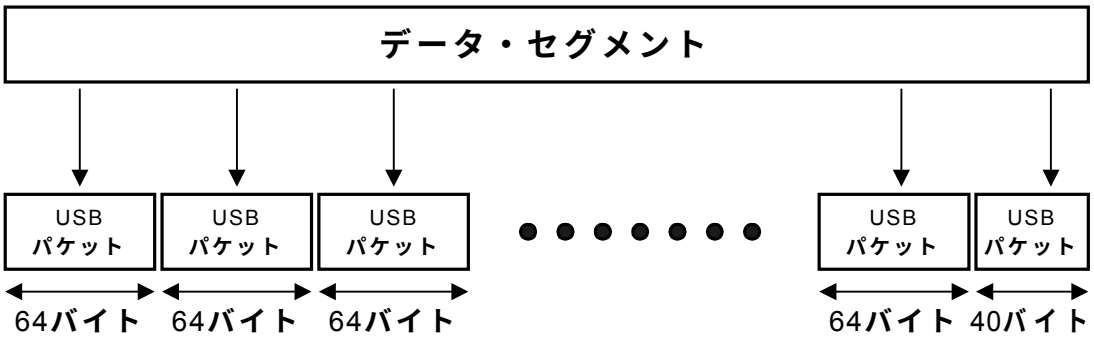
6.5　データ送信機能

このセクションは，USBコントローラのデータ送信機能について記述します。

6.5.1　送信処理の概要

USBコントローラは，システム・メモリのデータ・セグメントをUSBパケットに分割し，ホストPCに送信します。VR4120Aは，USBパケットのサイズをEP0制御レジスタ，EP1-2制御レジスタ，EP3-4制御レジスタ，EP5-6制御レジスタのMAXPフィールドに設定します（下の図では64バイトの値が設定された例を示します）。

図6－4　USBパケットへのデータの分割



データ・セグメントの最後のUSBパケットは，MAXPフィールドに設定された値よりも小さく（上の例では40バイト）なります。その結果，ホストPCはデータ・セグメント間の境界を識別できます。最後のUSBパケット・サイズがMAXPフィールドの値と等しいときは，送信SZLPモードで分割されたデータの最後の項目のあとにゼロ長のUSBパケットを送信します。送信NZLPモードでは，ゼロ長のUSBパケットを送信しません。送信モードの詳細については，6.5.3　**データ送信モード**を参照してください。

すべてのデータ・セグメントが転送されたあとで，USBコントローラは，ステータス情報をTxメールボックスに送信した結果としてTx送信表示をシステム・メモリに書き込みます。

Tx表示の詳細については，6.5.6　Tx**表示**を参照してください。

あるエンドポイントでのデータ・セグメントの送信時間は，対応する送信コマンドの発行時にスケジューリングされます。現在の送信状態は，USB Txエンドポイント・ステータス・レジスタ（アドレス：1000_1048H）の内容を読み込むことで判断できます。

送信コマンドの発行方法の詳細については，6.5.4　**データ送信時のVR4120A処理**を参照してください。

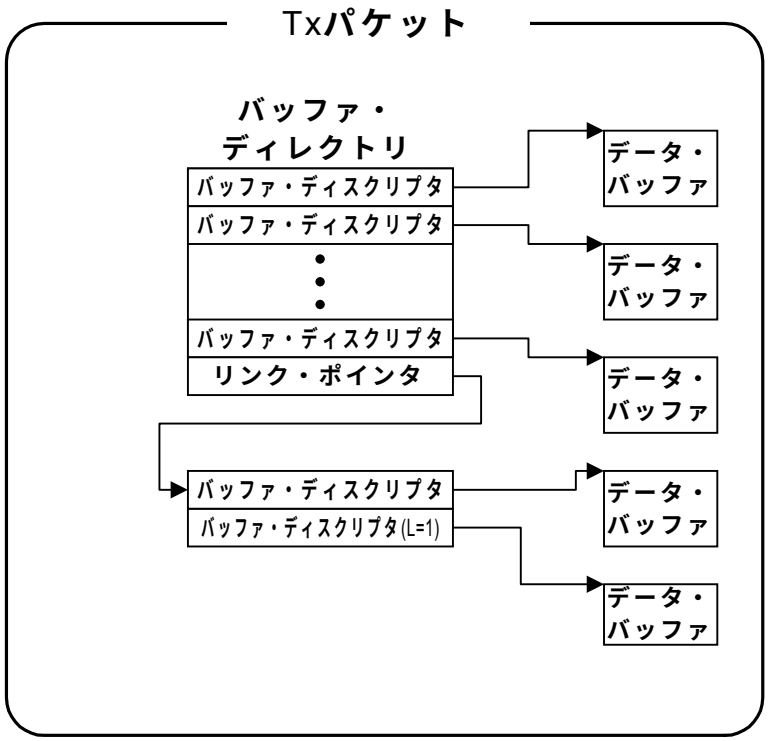
保守／廃止

6. 5. 2　Txバッファ・コンフィギュレーション

VR4120AはTxバッファをシステム・メモリに作成し，データをホストPCに送信するようにUSBコントローラに通知します。

Txバッファのコンフィギュレーションを以下に示します。

図6－5　Txバッファ・コンフィギュレーション



送信パケットは，システム・メモリ上で複数のデータ・バッファに分割されています。これらのデータ・バッファは，バッファ・ディレクトリに束ねておかれます。

Txバッファのバッファ・ディレクトリ，バッファ・ディスクリプタ，リンク・ポインタのフォーマットを以下に示します。

図6－6　送信バッファ・ディレクトリのコンフィギュレーション

-Txバッファ・ディレクトリ

31	0
バッファ・ディスクリプタ 0	
バッファ・ディスクリプタ 1	
バッファ・ディスクリプタ 2	
バッファ・ディスクリプタ 3	
バッファ・ディスクリプタ 4	
|	
バッファ・ディスクリプタ N	
リンク・ポインタ	

-Txバッファ・ディスクリプタ

31	30	29	16	15	0
L	1	Reserved 注		サイズ	
バッファ・アドレス					

-Txリンク・ポインタ

31	30	29	0
0	0	Reserved 注	
ディレクトリ・アドレス			

注　0をライトしてください。

Tx**バッファ・ディレクトリ**　：**バッファ・ディスクリプタとリンク・ポインタから構成されます。1つのTxバッファ・ディレクトリは，最大255個のバッファ・ディスクリプタを収納できます。**

Tx**バッファ・ディスクリプタ**：**Txバッファのアドレス・データを保持します。**

ビット31（Lビット）を1にセットすると，このバッファ・ディスクリプタはパケットの最後のバッファであることを示します。

ビット30は，バッファ・ディスクリプタとリンク・ポインタを区別するために使用します。このビットを1にセットすると，バッファ・ディスクリプタであることを示します。0にセットすると，Txリンク・ポインタであることを示します。

“サイズ”フィールドは，バッファ・サイズを示します。バッファ・サイズとして1-65535バイトが設定できます。“バッファ・アドレス”フィールドはバッファのスタート・アドレスを示します。

Tx**リンク・ポインタ**　：**次のパケット・ディレクトリを示します。**

ビット31は通常0にセットします。

ビット30は，バッファ・ディスクリプタとリンク・ポインタを区別するために使用します。このビットを0にセットすると，リンク・ポインタであることを示します。

“ディレクトリ・アドレス”フィールドは，次のバッファ・ディレクトリのスタート・アドレスを示します。次のバッファ・ディレクトリが存在しない場合，ディレクトリ・アドレスは不定でも問題ありません。

6.5.3　データ送信モード

USB**コントローラは2つの送信モードをサポートします。これらのモードの違いは，データ・セグメントの最後のUSBパケットが送信されたあとでゼロ長のUSBパケットを送信するかどうかだけです。その他すべての面において2つのモードは同じです。送信モードは，**USB EP1**エンドポイント制御レジスタ（アドレス：**1000_1024H**）と**USB EP3**エンドポイント制御レジスタ（アドレス：**1000_102CH**）の**TM**ビット（ビット**19**）を用いて切り替えます。**

注意1. TM**ビットの設定は，エンドポイント**1**とエンドポイント**3**にのみ適用されます。**

2. **エンドポイント**0**とエンドポイント**5**では，**NZLP**モードのみの使用となります。**

(1) SZLP**（送信ゼロ長パケット）モード**

このモードでは，データ・セグメントの最後のUSBパケット・サイズがMAXPフィールドの値に等しいと，データ・セグメントの送信完了後にゼロ長のパケットを送信します。

最後のパケット・サイズがMAXPフィールドの値より小さいと，ショート・パケットを送信し，ゼロ長のパケットは送信しません。

(2) NZLP**（非ゼロ長パケット）モード**

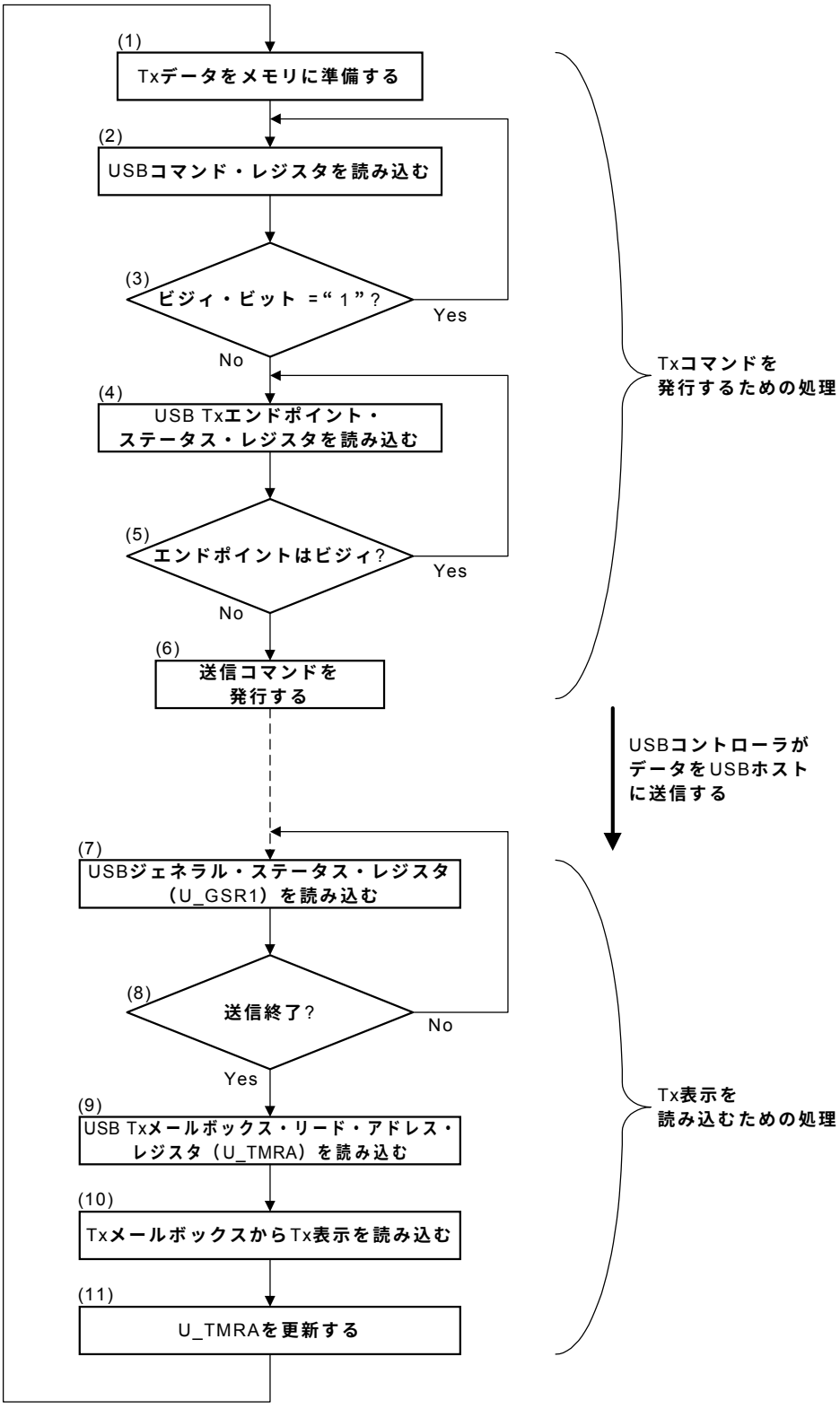
このモードでは，データ・セグメントの最後のUSBパケット・サイズがMAXPフィールドの値に等しくても，そのパケット送信後にゼロ長のパケットを送信しません。

保守／廃止

6. 5. 4　データ送信時のVR4120Aの処理

このセクションは，VR4120Aがデータ送信時に行う処理について記述します。

図6－7　データ送信時のVR4120Aの処理



(1) まず，VR4120Aは送信するデータをシステム・メモリに準備します。

(2) VR4120AはUSBコマンド・レジスタ（U_CMR）を読み込みます。

(3) VR4120AはUSBコマンド・レジスタ（U_CMR）のビジィ・ビット（ビット31：Bビット）がセットされているかどうかをチェックします。ビジィ・ビットがセットされていれば，USBコントローラはまだ前のコマンドを実行していることを示します。この場合，VR4120Aは新しいコマンドを発行できません。

(4) VR4120AはUSB Txエンドポイント・ステータス・レジスタ（U_TEPSR）を読み込みます。

(5) VR4120Aは次に送信を行うエンドポイントがビジィ状態にあるかどうかをチェックします。エンドポイントがビジィである場合は，VR4120AはUSB Txエンドポイント・ステータス・レジスタ（U_TEPSR）の読み込みから処理を繰り返します。

(6) VR4120AはUSBコマンド・レジスタ（U_CMR）に対してTxコマンドを発行します。

(7) VR4120AはUSBジェネラル・ステータス・レジスタ1（U_GSR1）を読み込みます。

(8) VR4120Aは送信が終了したかどうかをチェックします。

(9) VR4120AはUSB Txメールボックス・リード・アドレス・レジスタ（U_TMRA）の内容を読み込みます。

(10) VR4120AはTxメールボックスからTx表示を読み込みます。

(11) VR4120AはUSB Txメールボックス・リード・アドレス・レジスタ（U_TMRA）の内容を更新します。

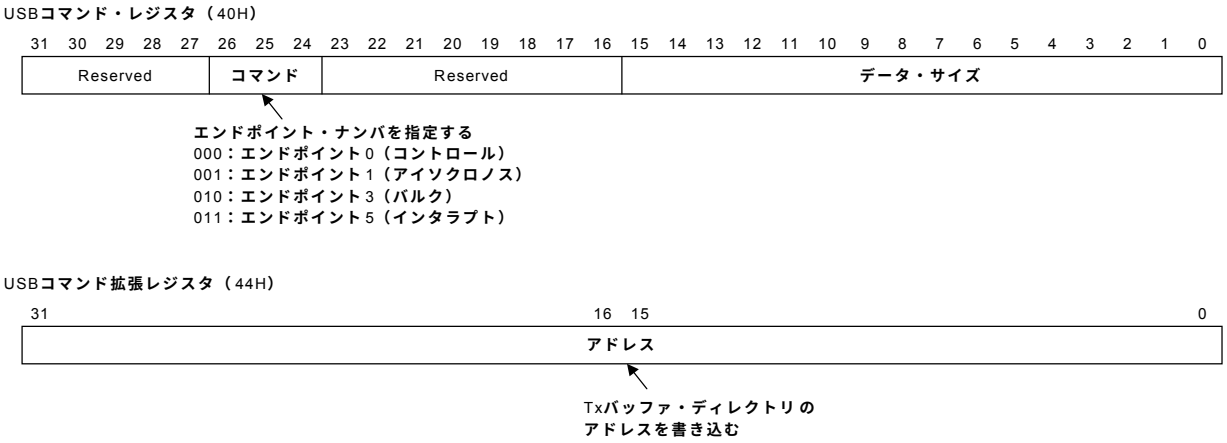
次に示すレジスタに値を書き込むことによって送信コマンドを発行します。書き込みの際，まずUSBコマンド拡張レジスタに書き込み，次にUSBコマンド・レジスタに書き込みます。

USBコントローラは，送信コマンドを発行されると，データ・セグメントの送信をスケジューリングします。

USBコマンド・レジスタのデータ・サイズ・フィールドが“0”であれば，USBコントローラはゼロ長のパケットを送信します。

送信コマンドが設定するデータのサイズと送信するデータ・バッファの合計サイズは，同じ値でなければなりません。

図6－8　送信コマンドの発行

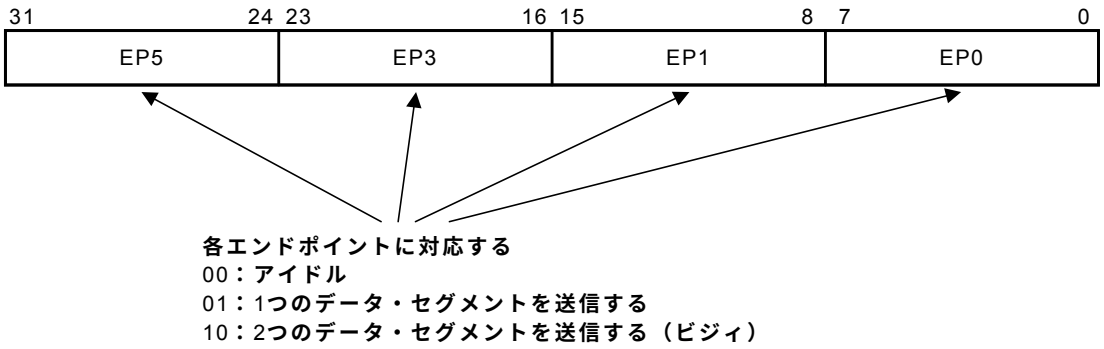


あるエンドポイントにおいて，最大2つのデータ・セグメントの送信をスケジュールすることが可能です。2つのデータ・セグメントをスケジュールすると，VR4120Aが3番目のデータ・セグメントを送信するように送信コマンドをUSBコマンド・レジスタに書き込んでも，そのコマンドは最初にスケジュールされたデータが送信されるまで受け付けられません。

送信がスケジュールされたデータ・セグメントの数は，USB Txエンドポイント・ステータス・レジスタ（U_TEPSR）を読み込むことで判断できます。

図6－9　送信ステータス・レジスタ

USB Txエンドポイント・ステータス・レジスタ
(48H)



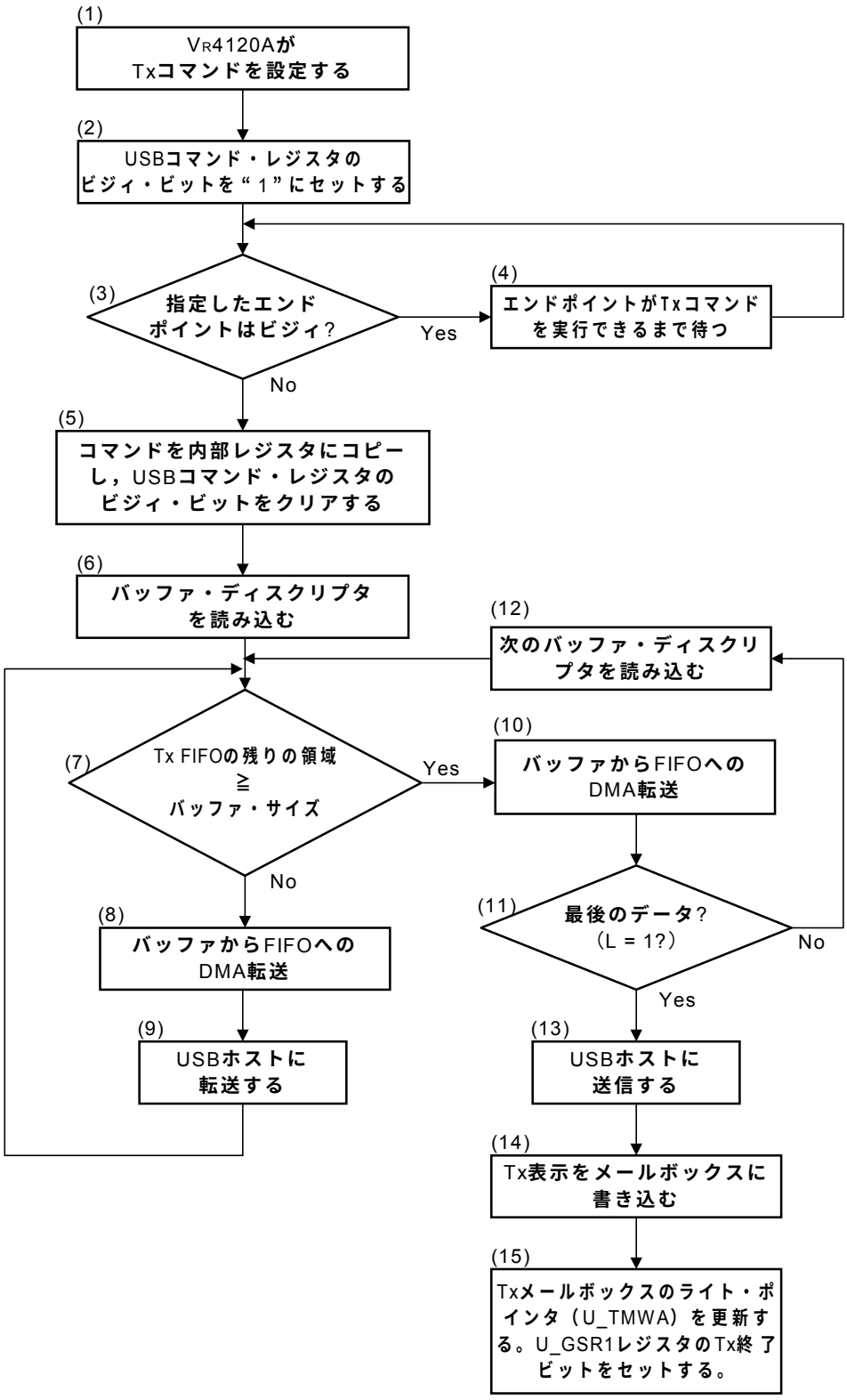
各エンドポイントに対応する
00：アイドル
01：1つのデータ・セグメントを送信する
10：2つのデータ・セグメントを送信する（ビジィ）

保守／廃止

6.5.5　データ送信時のUSBコントローラの処理

このセクションは，USBコントローラがデータ送信時に行う処理について記述します。

図6－10　USBコントローラの送信処理フロー・チャート



（1）から（15）の番号は，USBコントローラが処理を行うべき順序を示しているわけではありません。これらの番号は，以下の説明に対応しています。

（1）USBコントローラは，送信コマンドをVR4120Aから受け取ると送信処理を開始します。

（2）コマンドを書き込むと，USBコントローラはUSBコマンド・レジスタ（U_CMR）のビジィ・ビット（ビット31）を“1”にセットします。

（3）USBコントローラは，送信コマンドで指定したエンドポイントが現在ビジィ状態（2つのデータ・セグメントを送信するようにスケジュールされた）にあるかどうかをチェックします。

（4）送信コマンドで指定したエンドポイントがビジィ状態であると，（1）で書き込まれたコマンドは指定したエンドポイントが新しいTxコマンドを受け付け可となるまで実行されません。

（5）（1）でUSBコマンド・レジスタ（U_CMR）とUSBコマンド拡張レジスタ（U_CA）に書き込まれたコマンドを内部レジスタにコピーし，USBコマンド・レジスタのビジィ・ビットを“0”に戻します。

（6）USBコントローラがU_CAによって指定されたTxバッファ・ディレクトリから最初のバッファ・ディスクリプタを読み込みます。

（7）USBコントローラが，Tx FIFOに残っている領域のサイズを前のステップで読み込まれたバッファ・ディスクリプタのバッファ・サイズと比較します。

（8）上記の（7）でTx FIFOに残っている領域の方が小さい場合，USBコントローラはTx FIFOがフルになるまでバッファからデータをDMAで転送します。

（9）Tx FIFOがフルになると，USBコントローラはデータをUSBホストに転送します。

（10）上記の（7）でTx FIFOに残っている領域の方が大きい場合，USBコントローラはバッファ内のすべてのデータをTx FIFOにDMAで転送します。

（11）USBコントローラは，DMAで転送したデータがホストに送信するデータ・セグメントの最後のデータであるかどうかをチェックします。

（12）DMAで転送したデータが送信する最後のデータでない場合は，バッファが空であることを示します。したがって，USBコントローラは次のバッファ・ディスクリプタを読み込みます。

（13）DMAで転送したデータが送信する最後のデータである場合は，USBコントローラはデータを送信します。

（14）USBコントローラはTx表示をメールボックスに書き込みます。

（15）USBコントローラは，メールボックスのライト・ポインタ（USB Txメールボックス・ライト・アドレス・レジスタ：U_TMWA）を更新します。また，USBジェネラル・ステータス・レジスタ1（U_GSR1）の送信終了ビットをセットし，マスクされていなければ，VR4120Aに割り込みを発行します。

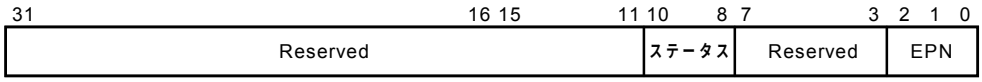
保守／廃止

6.5.6　Tx表示

送信データ・セグメントごとにUSBコントローラはTx表示をTxメールボックスに書き込みます。Tx表示を書き込んだあと，USBコントローラはUSBジェネラル・ステータス・レジスタ1（U_GSR1）の送信終了ビットを1にセットし，マスクされていなければ，VR4120Aに割り込みを発行します。

Tx表示のフォーマットを以下に示します。

図6－11　Tx表示のフォーマット



ステータス：データ送信時の状態を示すフィールドです。

ビット10　：0にセットすると，IBUSエラーが発生していないことを示します。
1にセットすると，IBUSエラーによって処理が異常終了したことを示します。

ビット9　：0にセットすると，データ送信中にバッファのアンダランが発生しなかったことを示します。
1にセットすると，バッファのアンダランが発生したことを示します。
このビットは，データをエンドポイント1に送信するときのみセットします。

ビット8　：0にセットすると，データ送信をSZLPモードで行ったことを示します。
1にセットすると，データ送信をNZLPモードで行ったことを示します。
エンドポイント0とエンドポイント5では，このビットを常時“1”にセットします。

EPN：エンドポイント・ナンバを示すフィールドです。

000：エンドポイント0
010：エンドポイント1
100：エンドポイント3
110：エンドポイント5

保守／廃止

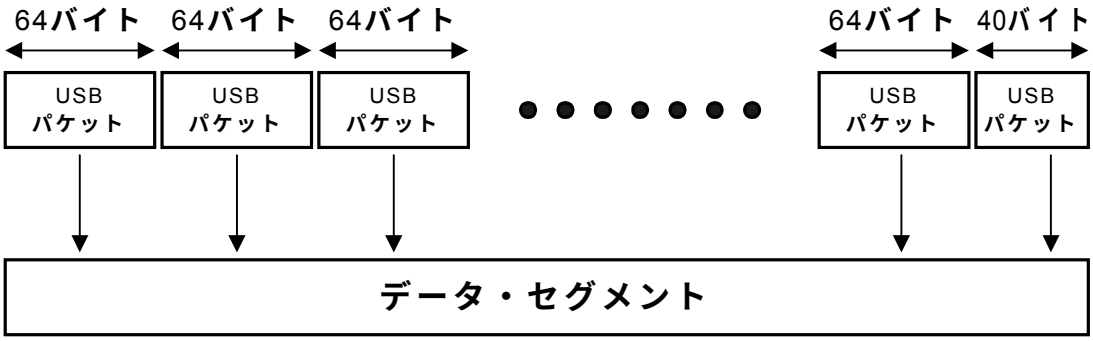
6.6　データ受信機能

このセクションは，USBコントローラのデータ受信機能について記述します。

6.6.1　受信処理の概要

USBコントローラは，USBパケットをUSBホストから受け取り，それらのUSBパケットをシステム・メモリに格納し，1つのデータ・セグメントに組み立てます。VR4120Aは，1つのUSBパケットのサイズをEP0制御レジスタ，EP1-2制御レジスタ，EP3-4制御レジスタ，EP5-6制御レジスタのMAXPフィールドに設定します（以下の図はパケット・サイズを64バイトに設定したときの例です）。

図6－12　USBパケットへのデータの分割



データ・セグメントがMAXPフィールドに設定された値と同じバイト・サイズをもつUSBパケットに分割されると，データ・セグメントの最後のUSBパケットはMAXPフィールドに設定された値よりも小さく（上の例では40バイト）なります。その結果，USBコントローラはデータ・セグメント間の境界を識別できます。最後のUSBバッファ・サイズがMAXPフィールドの値と等しいときは，データ・セグメントの最後のUSBパケットのあとにホストPCからUSBコントローラにゼロ長のUSBパケットが送信されます。

USBからシステム・メモリに受信したデータを置くために，受信USBパケットを格納する領域がシステム・メモリ内に必要となります。この領域をRxバッファと呼びます。この領域はVR4120Aが確保しなければなりません。Rxバッファの詳細については，6.6.2　Rx**バッファ・コンフィギュレーション**を参照してください。

データ・セグメント受信が終了すると，USBコントローラは“Rx表示”をシステム・メモリのメールボックスに書き込みます。Rxモードの詳細については，6.6.4　**データ受信モード**を参照してください。

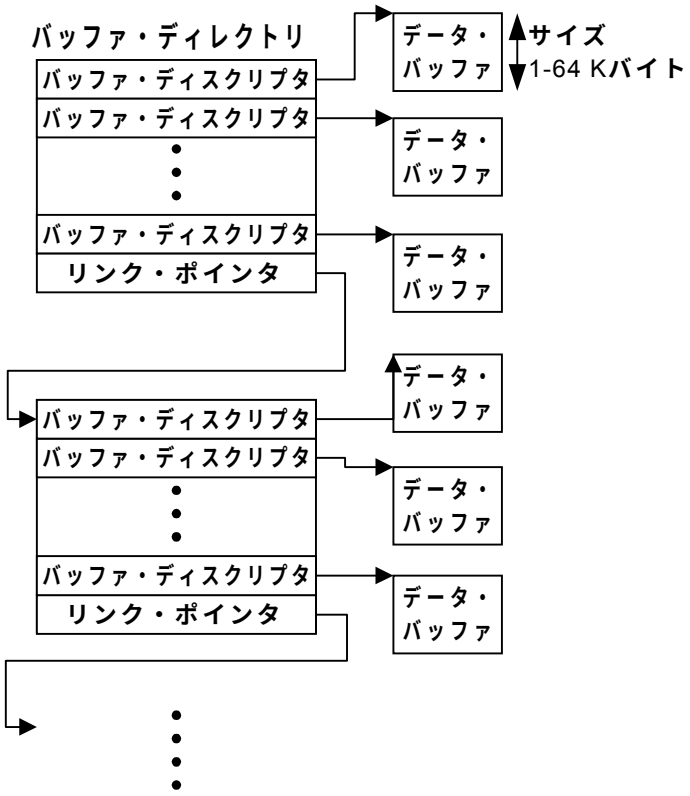
保守／廃止

6. 6. 2　Rxバッファ・コンフィギュレーション

USBから受信したデータは，システム・メモリ内の受信プールに格納します。

USBコントローラは3つの受信プールを使用します。受信プールのコンフィギュレーションを以下に示します。

図6－13　受信バッファ・コンフィギュレーション

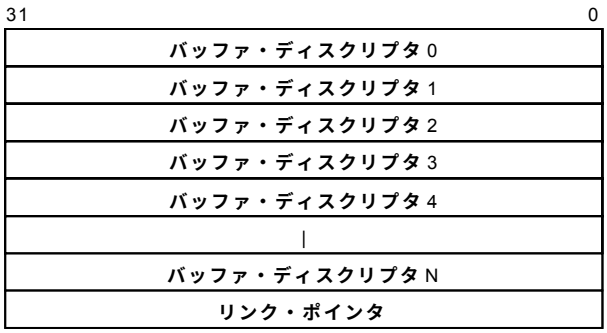


受信バッファは，バッファ・ディレクトリとデータ・バッファから構成されています。受信バッファは，VR4120Aによってシステム・メモリに準備される必要があります。

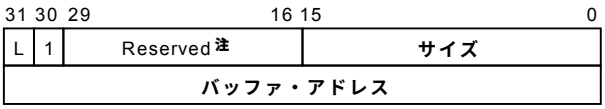
バッファ・ディレクトリ，バッファ・ディスクリプタ，リンク・ポインタのフォーマットを以下に示します。

図6－14　受信ディスクリプタのコンフィギュレーション

-Rxバッファ・ディレクトリ



-Rxバッファ・ディスクリプタ



-Rxリンク・ポインタ

注　0をライトしてください。

Rx**バッファ・ディレクトリ**　：バッファ・ディスクリプタとリンク・ポインタから構成されます。1つのRxバッファ・ディレクトリは，最大255個のバッファ・ディスクリプタを収納できます。

Rx**バッファ・ディスクリプタ**：Rxバッファのアドレスとそのサイズを示しています。

ビット31（Lビット）を1にセットすると，このバッファ・ディスクリプタはプールの最後のバッファであることを示します。

ビット30は，バッファ・ディスクリプタとリンク・ポインタを区別するために使用します。このビットを1にセットすると，バッファ・ディスクリプタであることを示します。0にセットすると，Rxリンク・ポインタであることを示します。

“サイズ”フィールドは，バッファ・サイズを示します。バッファ・サイズとして1-65535バイトが設定できます。“バッファ・アドレス”フィールドはバッファのスタート・アドレスを示します。

保守／廃止

Rxリンク・ポインタ　　：バッファ・ディレクトリの最後を示します。
ビット31は0にセットします。
ビット30は，バッファ・ディスクリプタとリンク・ポインタを区別するために使用します。このビットを0にセットすると，リンク・ポインタであることを示します。
“バッファ・ディレクトリ・アドレス”フィールドは，次のバッファ・ディレクトリのスタート・アドレスを示します。

6.6.3　受信プールの設定

USBコントローラは3つの受信プールを使用します。

プール0　エンドポイント0（コントロール）とエンドポイント6（インタラプト）用
プール1　エンドポイント2（アイソクロノス）用
プール2　エンドポイント4（バルク）用

3つのプールのデータは，対応するレジスタに書き込まれます。

プール0	USB Rxプール0情報レジスタ	（アドレス：1000_1050H）
	USB Rxプール0アドレス・レジスタ	（アドレス：1000_1054H）
プール1	USB Rxプール1情報レジスタ	（アドレス：1000_1058H）
	USB Rxプール1アドレス・レジスタ	（アドレス：1000_105CH）
プール2	USB Rxプール2情報レジスタ	（アドレス：1000_1060H）
	USB Rxプール2アドレス・レジスタ	（アドレス：1000_1064H）

VR4120Aは，これらのレジスタを読み込むことで各プールの現在の状態を知ることができます。

VR4120Aは，上記3つの情報レジスタのアラート・フィールドにのみ直接，値を書き込むことができます。ほかのフィールドは，USBコマンド・レジスタ（U_CMR）とUSBコマンド拡張レジスタ（U_CA）を用いて設定しなければなりません。

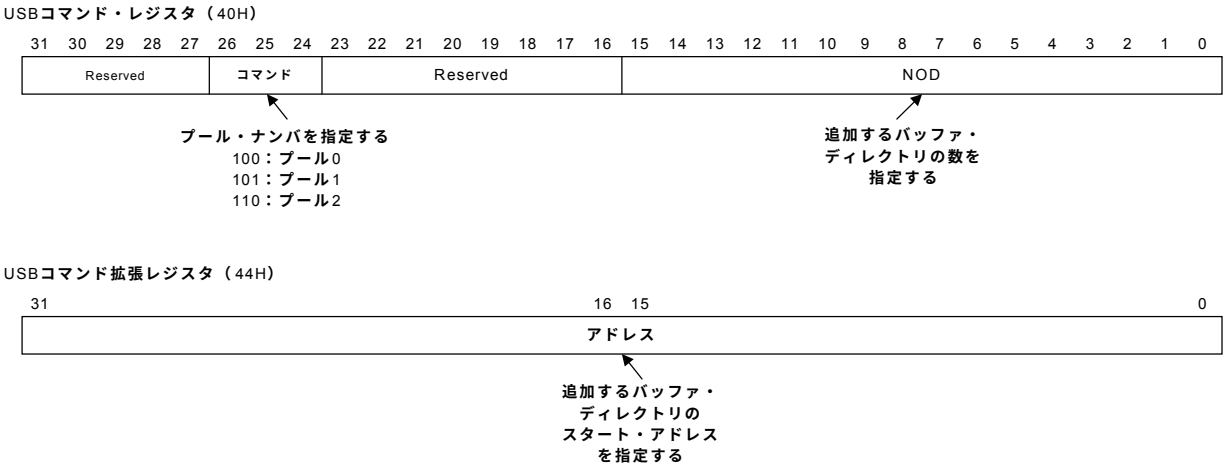
VR4120Aは，USBコマンド・レジスタ（アドレス：1000_1040H）とUSBコマンド拡張レジスタ（アドレス：1000_1044H）を用いて，各プールにバッファ・ディレクトリを追加します。

バッファ・ディレクトリを受信プールに追加するために，VR4120Aは以下の処理を行います。

(1) VR4120Aは，追加するバッファ・ディレクトリをプール，バッファ，システム・メモリに置きます。複数のバッファ・ディレクトリを追加するときは，あらかじめリンクしておきます。
(2) VR4120Aは，追加するバッファ・ディレクトリのスタート・アドレスを，プール内のすでに存在するバッファ・ディレクトリのリストの最後のリンク・ポインタに設定します。
(3) VR4120Aは，追加するバッファ・ディレクトリのスタート・アドレスをUSBコマンド拡張レジスタ（アドレス：1000_1044H）に設定します。
(4) VR4120Aは，追加するバッファ・ディレクトリのプール・ナンバとサイズをUSBコマンド・レジスタ（アドレス：1000_1040H）に設定します。

保守／廃止

図6－15　バッファ・ディレクトリの追加コマンド



USBコントローラの動作は，バッファ・ディレクトリの追加コマンドがUSBコマンド・レジスタに書き込まれたときに，未使用のバッファ・ディレクトリが対応するプールに残っているかどうかで変わります。

(a) 未使用のバッファ・ディレクトリがプールに残っている場合（プール情報レジスタのRNODフィールドが0よりも大きな値に設定されている場合），USBコントローラはコマンドのNODフィールドの数をプール情報レジスタのRNODフィールドに追加します。

(b) プールが空である場合（プール情報レジスタのRNODフィールドが0である場合），USBコントローラはコマンドのNODフィールドに設定した値をプール情報レジスタのRNODフィールドにロードします。さらに，USBコマンド拡張レジスタに書き込まれた値をプール・アドレス・レジスタにロードします。

6.6.4　データ受信モード

USBコントローラは，エンドポイントと受信モードごとに異なる受信処理を行います。

受信モードは，USB EP2制御レジスタ（アドレス：1000_1028H）とUSB EP4制御レジスタ（アドレス：1000_1030H）のRMフィールド（ビット20：19）で決定されます。4種類の受信処理があります。

(1) エンドポイント0，エンドポイント6

(2) エンドポイント2，エンドポイント4通常モード

(3) エンドポイント2，エンドポイント4アセンブル・モード

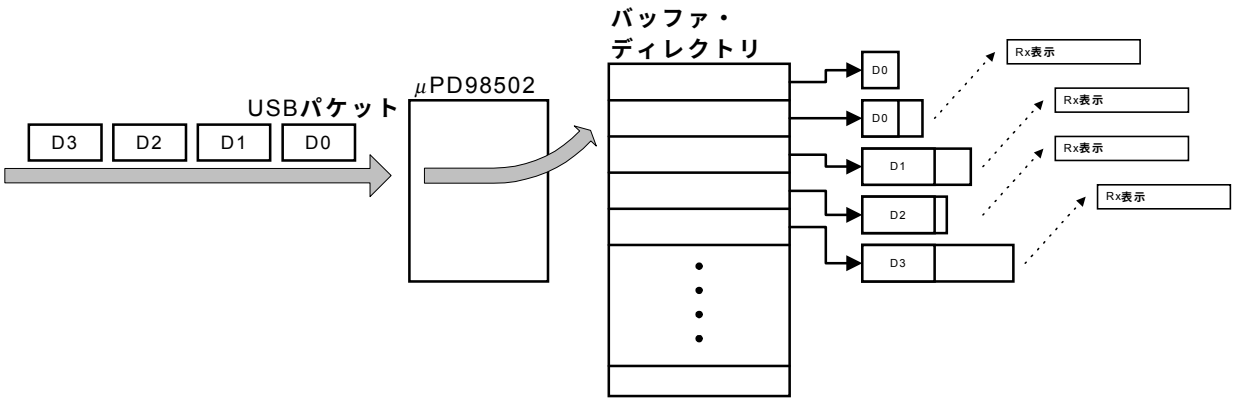
(4) エンドポイント2，エンドポイント4セパレート・モード

各処理について以下に記述します。

(1) エンドポイント0，エンドポイント6における受信

エンドポイント0，エンドポイント6は，受信ごとに受信モードにかかわらず同じ処理を実行します。

図6－16　エンドポイント0，エンドポイント6におけるデータ受信

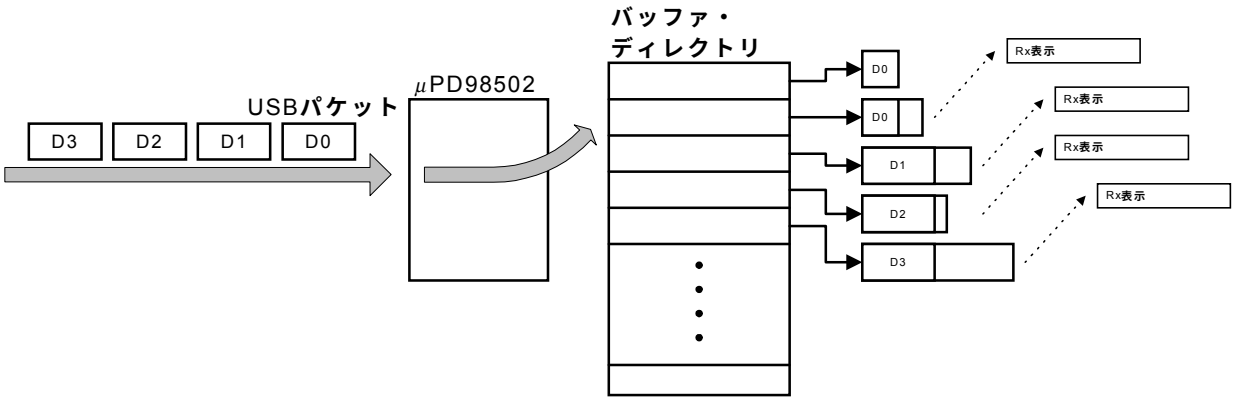


USBコントローラが1つのUSBパケットを受け取ると，データ・バッファに格納しRx表示をメールボックスに書き込みます。USBコントローラは，Rx表示を書き込む前にUSBパケットごとにバッファ・ディスクリプタのサイズ・フィールドとLビット・フィールドを更新します。

(2) エンドポイント2，エンドポイント4，通常モード

エンドポイント2，エンドポイント4受信通常モードにおける処理を以下に記述します。

図6－17　エンドポイント2，エンドポイント4受信通常モード



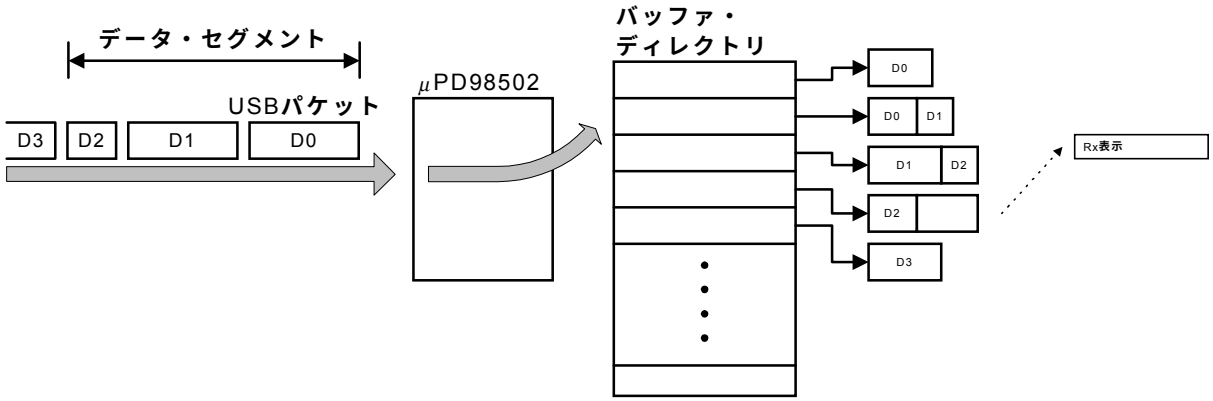
USBコントローラが1つのUSBパケットを受け取ると，データ・バッファに格納しRx表示をメールボックスに書き込みます。USBコントローラは，Rx表示を書き込む前にUSBパケットごとにバッファ・ディスクリプタのサイズ・フィールドとLビット・フィールドを更新します。

保守／廃止

(3) エンドポイント2，エンドポイント4，アセンブル・モード

エンドポイント2，エンドポイント4受信アセンブル・モードにおける処理を以下に記述します。

図6－18　エンドポイント2，エンドポイント4受信アセンブル・モード



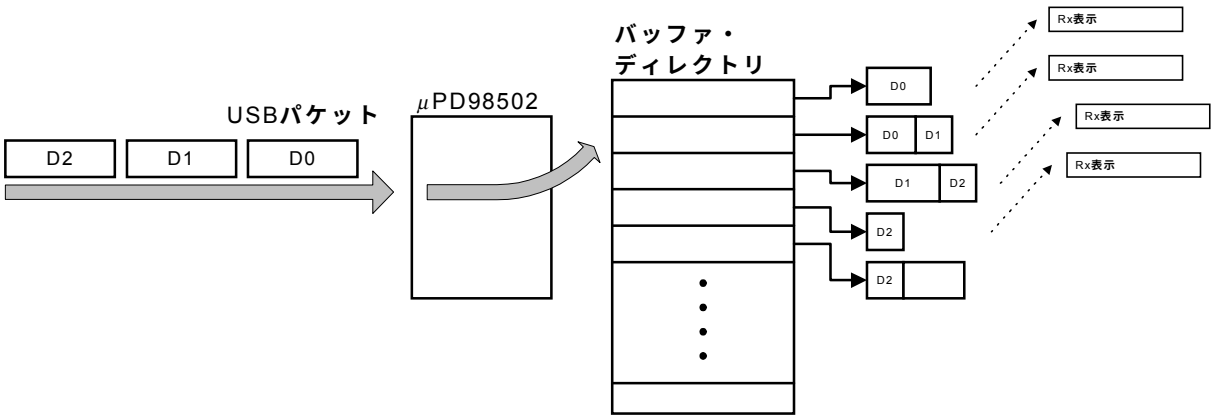
このモードでは，USBコントローラは1つのデータ・セグメントを受け取ったあとにRx表示を報告します。

すなわち，USBコントローラがUSBから受信したショート・パケットまたはゼロ長パケットをシステム・メモリのバッファに書き込んだあとに，USBコントローラは最後のバッファ・ディスクリプタのサイズ・フィールドとLビット・フィールドを更新し，Rx表示を報告します。

(4) エンドポイント2，エンドポイント4，セパレート・モード

エンドポイント2，エンドポイント4受信セパレート・モードにおける処理を以下に記述します。

図6－19　エンドポイント2，エンドポイント4受信セパレート・モード



このモードでは，USBコントローラがUSBパケットを受信し，データをRxバッファに格納したあとに，バッファがフルになるとRx表示を報告します。USBパケットの途中でバッファがフルになっても，Rx表示を報告し，引き続き，USBパケットを次のバッファに格納する処理を続行します。

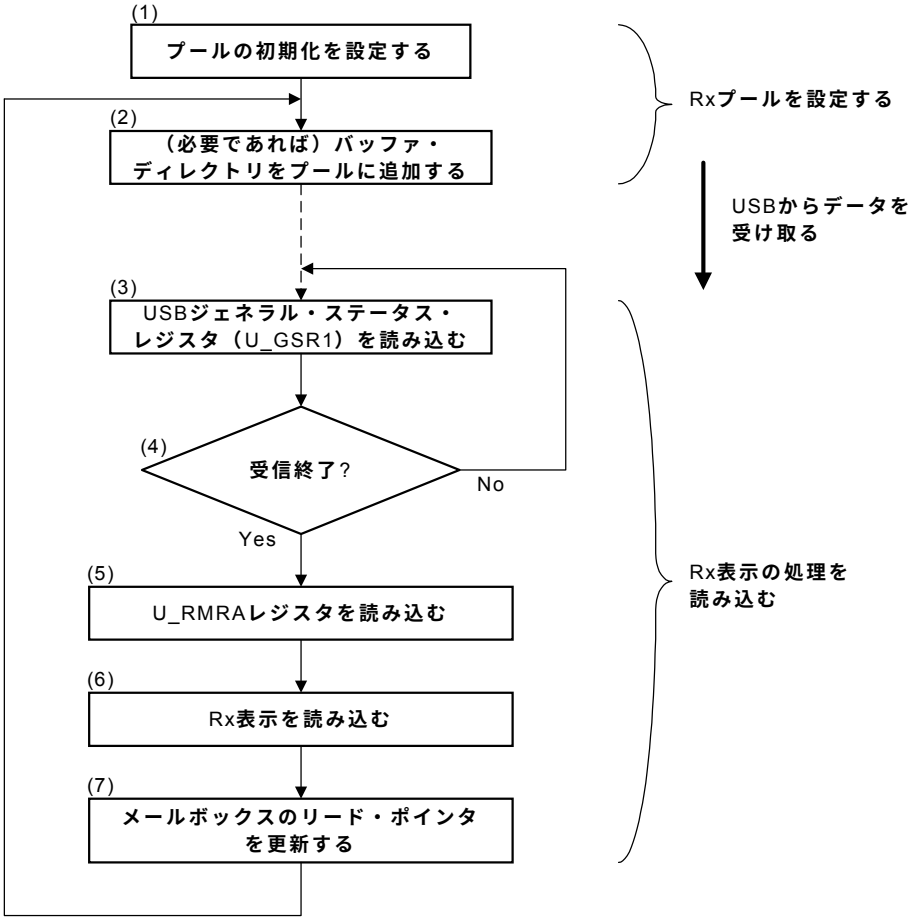
バッファ・ディスクリプタのサイズ・フィールドとLビット・フィールドは更新しません。

保守／廃止

6. 6. 5　VR4120Aの受信処理

このセクションは，VR4120Aがデータ受信中に実行しなければならない処理について記述します。

図6－20　VR4120Aの受信処理



（1）から（7）の番号は，VR4120Aが処理を行うべき順序を示しているわけではありません。これらの番号は，以下の説明に対応しています。

(1) まず初期化の一部として，VR4120Aはプール・コンフィギュレーションを設定しなければなりません。
(2) 必要であれば，受信時にVR4120Aはバッファ・ディレクトリをプールに追加しなければなりません。
(3) VR4120AはUSBジェネラル・ステータス・レジスタ（U_GSR1）を読み込みます。
(4) VR4120Aは受信が終了したかどうかをチェックします。
(5) 受信が終了すると，VR4120Aが次に読み込まなければならないメールボックスのアドレスを決定するために，VR4120AはUSB Rxメールボックス・リード・アドレス・レジスタ（アドレス：1000_1088H）を読み込みます。
(6) 次に，VR4120AはメールボックスからRx表示を読み込みます。
(7) VR4120AはUSB Rxメールボックス・リード・アドレス・レジスタ（U_RMRA）を更新します。

保守／廃止

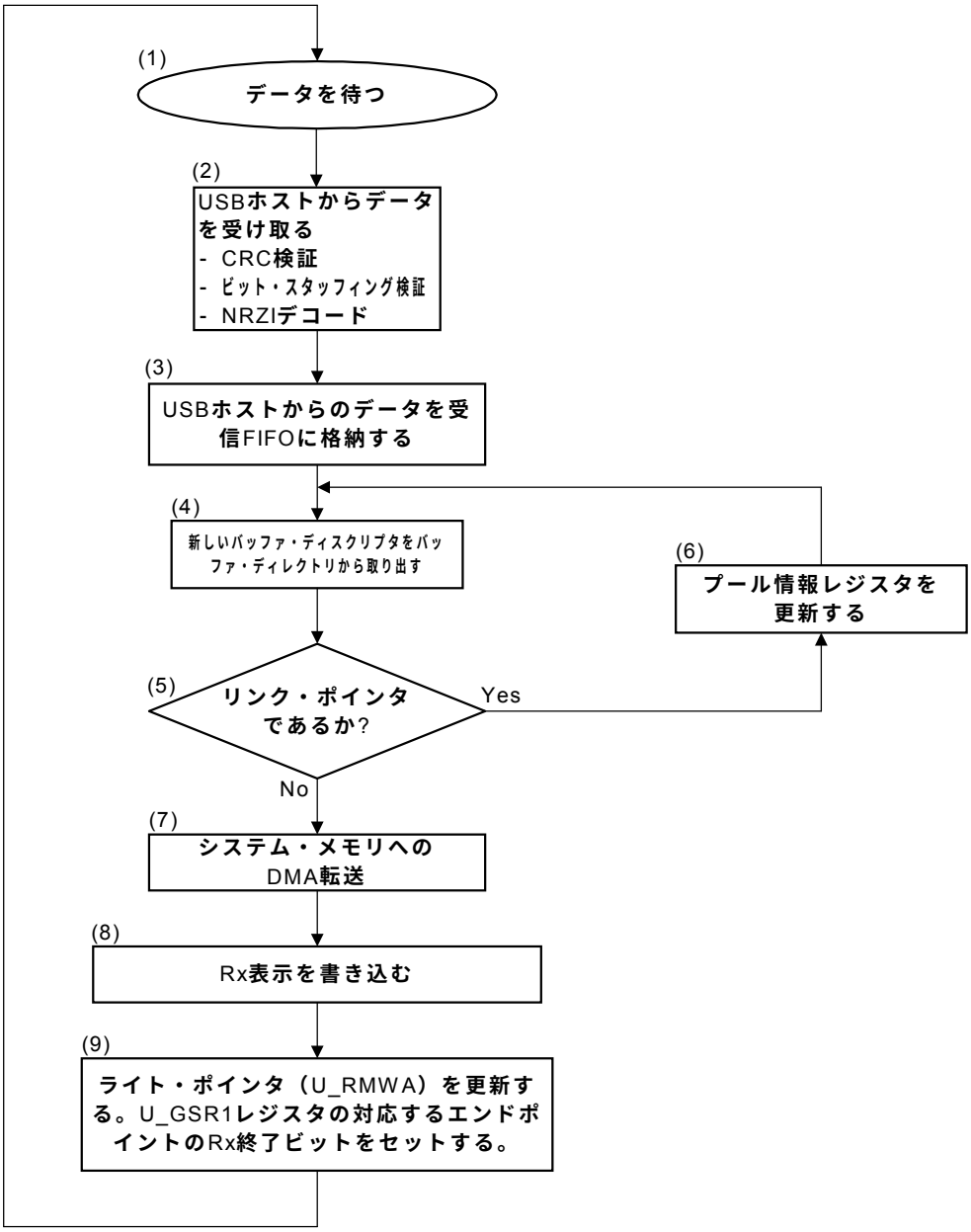
6. 6. 6　USBコントローラの受信処理

このセクションは，USBコントローラがデータ受信時に行う処理について記述します。

6. 6. 6. 1　EP0, EP6およびEP2, EP4の通常モード

USBコントローラがEP0, EP6およびEP2, EP4の通常モードで行う受信動作を以下の図に示します。

図6－21　USBコントローラの受信動作



（1）から（9）の番号は，USBコントローラが処理を行うべき順序を示しているわけではありません。これらの番号は，以下の説明に対応しています。

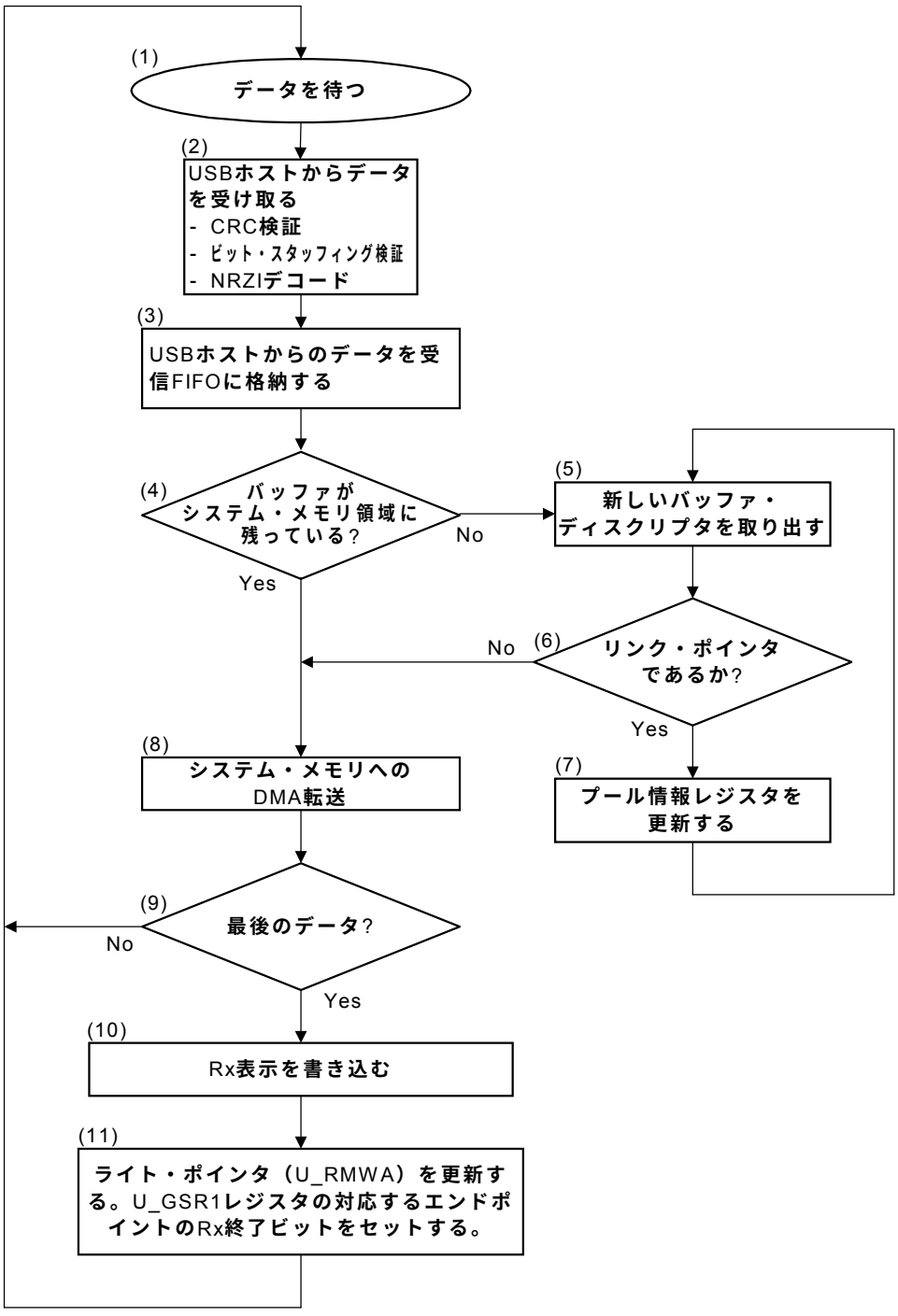
(1) USBコントローラは，USBホストからのデータ（USBパケット）の受信を待つ状態にあります。
(2) USBコントローラは，USBホストからのデータ（USBパケット）を受け取ります。データを受信しながら，NRZIのデコーディング，CRCチェック，ビット・スタッフィング・エラー・チェックを行います。
(3) USBコントローラは，受信したデータをFIFOに格納します。
(4) USBコントローラは，新しいバッファ・ディスクリプタの取り出しを開始します。
(5) USBコントローラは，取り出したバッファ・ディスクリプタがリンク・ポインタであるかどうかをチェックします。
(6) 取り出したバッファ・ディスクリプタがリンク・ポインタである場合は，USBコントローラはプール情報レジスタを更新し，新しいバッファ・ディスクリプタの取り出しを再開します。
(7) USBコントローラは，次にFIFOからシステム・メモリへデータをDMA転送します。
(8) 転送したデータが最後のデータであることをUSBコントローラが検出すると，バッファ・ディスクリプタのサイズ・フィールドとLビット・フィールドを更新し，Rx表示をRxメールボックスに書き込みます。
(9) USBコントローラは，メールボックスのライト・ポインタ（Rxメールボックス・ライト・アドレス・レジスタ，アドレス：1000_108CH）を更新します。また，USBジェネラル・ステータス・レジスタ1（U_GSR1）の受信終了ビットをセットし，マスクされていなければ，VR4120Aに割り込みを発行します。

保守／廃止

6. 6. 6. 2　アセンブル・モード

USBコントローラがアセンブル・モードで行う受信動作を以下の図に示します。

図6－22　USBコントローラの受信動作（アセンブル・モード）



（1）から（11）の番号は，USBコントローラが処理を行うべき順序を示しているわけではありません。これらの番号は，以下の説明に対応しています。

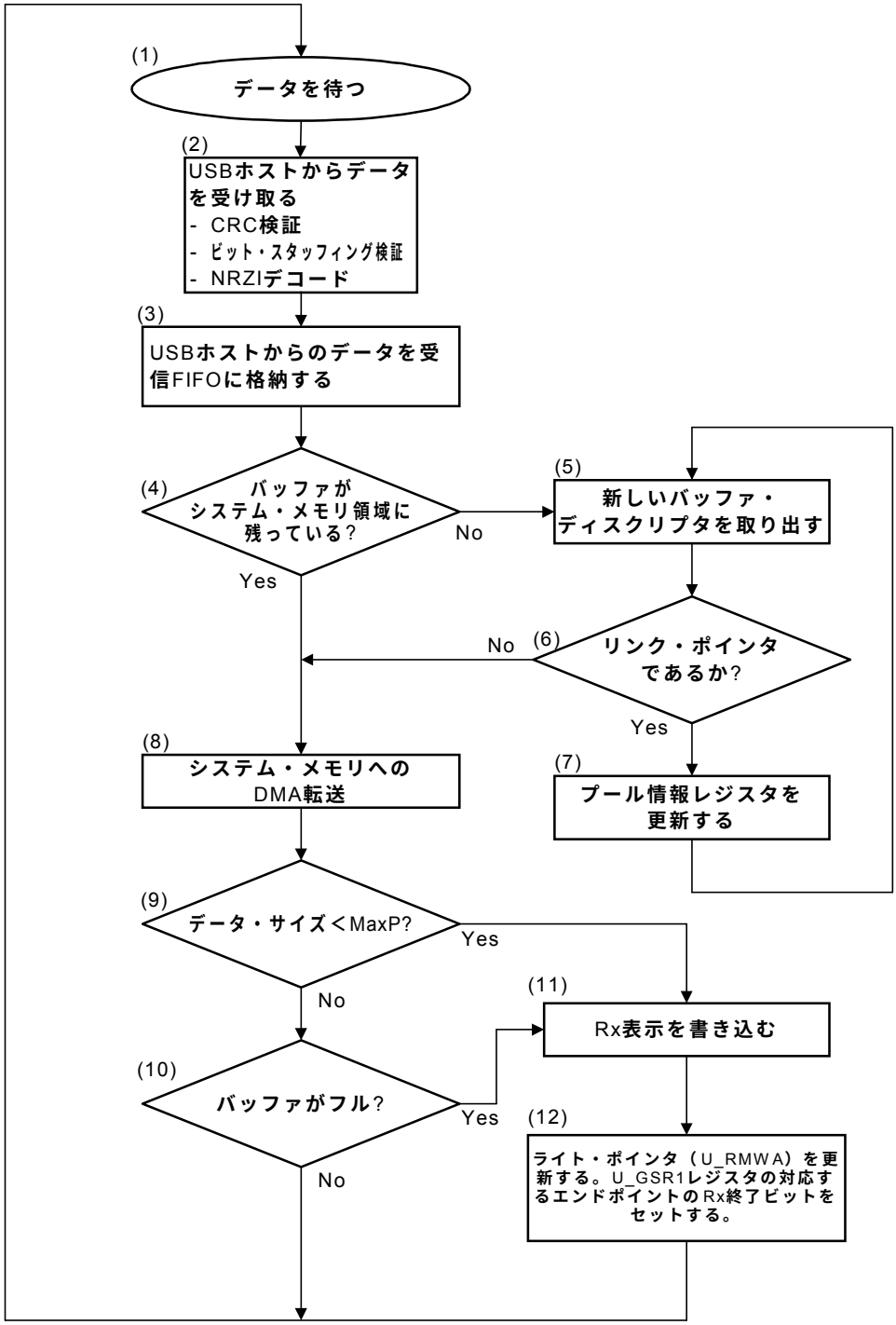
(1) USBコントローラは，USBホストからのデータ（USBパケット）の受信を待つ状態にあります。
(2) USBコントローラは，USBホストからのデータ（USBパケット）を受け取ります。データを受信しながら，NRZIのデコーディング，CRCチェック，ビット・スタッフィング・エラー・チェックを行います。
(3) USBコントローラは，受信したデータをFIFOに格納します。
(4) USBコントローラは，バッファがシステム・メモリ領域に残っているかどうかをチェックします（USBコントローラが新しいバッファ・ディスクリプタを取り出すべきかどうかをチェックします）。
(5) バッファがシステム・メモリ領域に残っていない場合は，USBコントローラは，新しいバッファ・ディスクリプタの取り出しを開始します。
(6) USBコントローラは，取り出したバッファ・ディスクリプタがリンク・ポインタであるかどうかをチェックします。
(7) 取り出したバッファ・ディスクリプタがリンク・ポインタである場合は，USBコントローラはプール情報レジスタを更新し，新しいバッファ・ディスクリプタの取り出しを再開します。
(8) USBコントローラは，次にFIFOからシステム・メモリへデータをDMA転送します。
(9) USBコントローラは，DMA転送したデータがセグメントの最後のデータであるかどうかをチェックします。
(10) 転送したデータが最後のデータであることをUSBコントローラが検出すると，バッファ・ディスクリプタのサイズ・フィールドとLビット・フィールドを更新し，Rx表示を準備したメールボックスに書き込みます。
(11) USBコントローラは，メールボックスのライト・ポインタ（Rxメールボックス・ライト・アドレス・レジスタ，アドレス：1000_108CH）を更新します。また，USBジェネラル・ステータス・レジスタ1（U_GSR1）の受信終了ビットをセットし，マスクされていなければ，VR4120Aに割り込みを発行します。

保守／廃止

6.6.6.3　セパレート・モード

USBコントローラがセパレート・モードで行う受信動作を以下の図に示します。

図6－23　USBコントローラの受信動作（セパレート・モード）



（1）から（12）の番号は，USBコントローラが処理を行うべき順序を示しているわけではありません。これらの番号は，以下の説明に対応しています。

(1) USBコントローラは，USBホストからのデータ（USBパケット）の受信を待つ状態にあります。
(2) USBコントローラは，USBホストからのデータ（USBパケット）を受け取ります。データを受信しながら，NRZIのデコーディング，CRCチェック，ビット・スタッフィング・エラー・チェックを行います。
(3) USBコントローラは，受信したデータをFIFOに格納します。
(4) USBコントローラは，バッファがシステム・メモリ領域に残っているかどうかをチェックします（USBコントローラが新しいバッファ・ディスクリプタを取り出すべきかどうかをチェックします）。
(5) バッファがシステム・メモリ領域に残っていない場合は，USBコントローラは，新しいバッファ・ディスクリプタの取り出しを開始します。
(6) USBコントローラは，取り出したバッファ・ディスクリプタがリンク・ポインタであるかどうかをチェックします。
(7) 取り出したバッファ・ディスクリプタがリンク・ポインタである場合は，USBコントローラはプール情報レジスタを更新し，新しいバッファ・ディスクリプタの取り出しを再開します。
(8) USBコントローラは，次にFIFOからシステム・メモリへデータをDMA転送します。
(9) USBコントローラは，DMA転送したデータが最大パケット・サイズ（MAXPn）よりも小さいかどうかをチェックします。
(10) USBコントローラは，バッファ領域がフルであるかどうかをチェックします。
(11) USBコントローラは，バッファ・ディスクリプタのサイズ・フィールドとLビット・フィールドを更新せず，最初の値のままRx表示を準備したメールボックスに書き込みます。
ここで注意しなければならない点があります。用意したバッファよりも短いショート・パケットを受信すると，バッファの容量が残っている状態でRx表示を報告します。このとき，受信ディスクリプタのサイズ・フィールドとLフィールドが更新されないので，サイズ・フィールドの値と実際に受信したパケットのサイズが一致しない状態が発生します。正確な受信パケットのサイズはRx表示に書き込まれるので，そちらをソフトウエアで参照してください。
(12) USBコントローラは，メールボックスのライト・ポインタ（Rxメールボックス・ライト・アドレス・レジスタ，アドレス：1000_108CH）を更新します。また，USBジェネラル・ステータス・レジスタ1（U_GSR1）の受信終了ビットをセットし，マスクされていなければ，VR4120Aに割り込みを発行します。

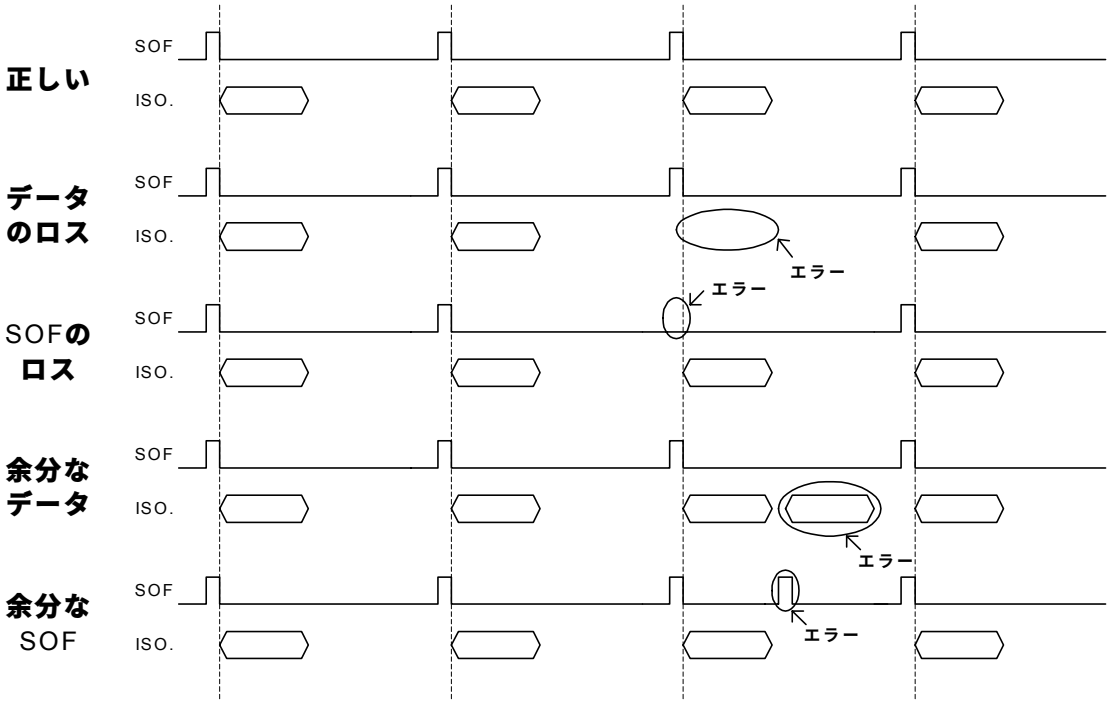
保守／廃止

6. 6. 7　USBでのエラーの検出

USBコントローラには，USB上でのいくつかのエラーを検出する機能があります。

次の図に示すエラーは，アイソクロノス・エンドポイントとSOFパケットの関係を示します。

図6－24　USBタイミング・エラー



(1) “データのロス”エラーを検出すると，USBジェネラル・ステータス・レジスタ2（U_GSR2）のEP2NDビット（ビット5）をセットします。このエラーに対するUSBコントローラのその他のアクションについては，次のセクション6. 6. 8に記述しています。

(2) “SOFのロス”エラーを検出すると，USBジェネラル・ステータス・レジスタ2（U_GSR2）のSLビット（ビット0）をセットします。

この場合，USBコントローラはUSBジェネラル・ステータス・レジスタにエラーを反映させるだけです。

(3) “余分なデータ”エラーを検出すると，USBジェネラル・ステータス・レジスタ2（U_GSR2）のEP2EDビット（ビット6）をセットします。

この場合，USBコントローラはUSBジェネラル・ステータス・レジスタ2（U_GSR2）にエラーを反映させるだけです。

(4) “余分なSOF”エラーを検出すると，USBジェネラル・ステータス・レジスタ2のESビット（ビット1）をセットします。

この場合，USBコントローラはUSBジェネラル・ステータス・レジスタ2（U_GSR2）にエラーを反映させるだけです。

保守／廃止

USBコントローラは，以下に示すその他のエラーを検出できます。

・アイソクロノス・データ・オーバサイズ・エラー：受信したデータ・パケット・サイズがエンドポイント2の最大パケット・サイズを越えると，USBコントローラはUSBジェネラル・ステータス・レジスタ2（U_GSR2）のEP2OSビット（ビット7）をセットします。

・不正エンドポイント・ナンバ：受信したIN/OUTトークン・パケットがVR4120Aからイネーブルにされないエンドポイント・ナンバまたは7を越えるエンドポイント・ナンバを含む場合は，USBコントローラはUSBジェネラル・ステータス・レジスタ2（U_GSR2）のIEAビット（ビット19）をセットします。

・エンドポイント1 Tx FIFOにデータなし：エンドポイント2に対するINトークン・パケットがエンドポイント2に対するTx FIFOがレディでないときに来ると，USBコントローラはUSBにまったくデータを送信せず，USBジェネラル・ステータス・レジスタ2（U_GSR2）のEP1NDビット（ビット2）をセットします。

・エンドポイント1に余分なトークン：エンドポイント1に対する2個以上のINトークン・パケットが2つのSOFの間に来ると，USBコントローラはUSBジェネラル・ステータス・レジスタ2（U_GSR2）のEP1ETビット（ビット3）をセットします。この場合，USBコントローラはデータを1回だけ送信します。

・エンドポイント1にトークンなし：エンドポイント1に対するINトークン・パケットが2つのSOFの間に来ないと，USBコントローラはUSBジェネラル・ステータス・レジスタ2（U_GSR2）のEP1NTビット（ビット4）をセットします。

保守／廃止

6.6.8　アイソクロノス・エンドポイントにおけるRxデータの消失

アイソクロノスRxエンドポイント（EP2）では，1つのデータ・パケットが1フレームごとに来ます。

2つのSOFパケットの間（1フレーム中）に1つもアイソクロノス・データ・パケットが来ないと，アイソクロノス・データは消失したとみなされます。

消失した場合，USBコントローラのアクションはRxモードに応じて変化します。

(a) Rx通常モード

USBコントローラは，USBジェネラル・ステータス・レジスタ2（U_GSR2）のEP2ND（エンドポイント2ノー・データ）ビット（ビット6）をセットします。

USBコントローラは，Rx表示を書き込みません。

USBコントローラは，システム・メモリのデータ・バッファにデータを書き込みません。

(b) Rxアセンブル・モード

USBコントローラは，USBジェネラル・ステータス・レジスタ2（U_GSR2）のEP2ND（エンドポイント2ノー・データ）ビット（ビット6）をセットします。

USBコントローラは，データ・バッファにダミー・データを書き込みます（実際には，USBコントローラは，最大パケット・サイズでデータ・バッファを番地づけするポインタをインクリメントするだけです。DMA転送は発生しません）。

USBコントローラは，受信データがアイソクロノス・エンドポイントで消失したことを示すRx表示を書き込みます。

(c) Rxセパレート・モード

USBコントローラは，USBジェネラル・ステータス・レジスタ2（U_GSR2）のEP2ND（エンドポイント2ノー・データ）ビット（ビット6）をセットします。

USBコントローラは，データ・バッファにダミー・データを書き込みます（実際には，USBコントローラは，最大パケット・サイズでデータ・バッファを番地づけするポインタをインクリメントするだけです。DMA転送は発生しません）。

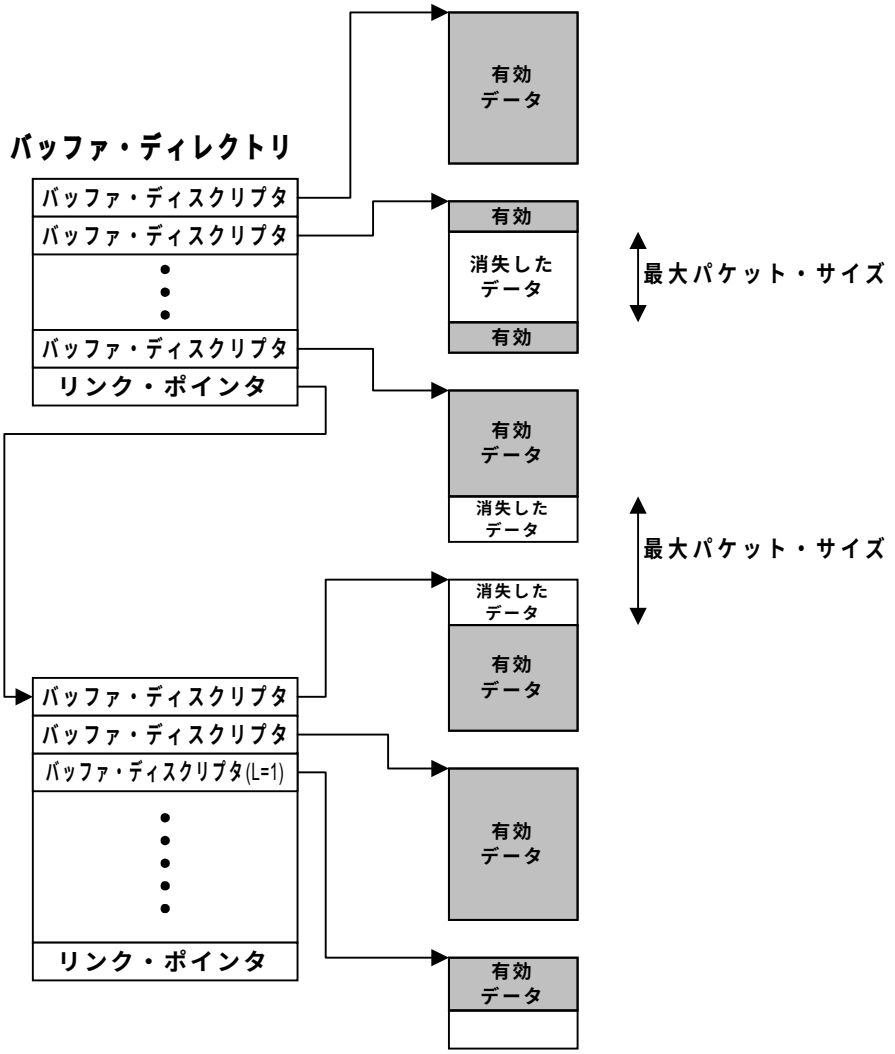
USBコントローラは，受信データがアイソクロノス・エンドポイントで消失したことを示すRx表示を書き込みます。

例

RxアセンブルまたはRxセパレート・モードでUSBコントローラがUSBパケットを受信したときに，アイソクロノスRxエンドポイント（エンドポイント4）でデータが消失すると，データ・バッファは図6－25のようになります。

次の図で灰色になっている領域は有効なデータで満たされています。USBコントローラがデータの消失を検出すると，消失したデータの領域を確保しているデータ・バッファに次のデータを格納します。消失したデータの領域は，アイソクロノスRxエンドポイント（エンドポイント2）の最大パケット・サイズと等しくなります。

図6－25　消失したデータを含むバッファの例



6. 6. 9　Rx FIFOオーバラン

アイソクロノスRxエンドポイント（EP2）でRx FIFOからのデータの読み込みが何らかの理由で遅れると，Rx FIFOオーバランが発生します。

オーバランが発生した場合，USBコントローラのアクションはRxモードに応じて変化します。

（a）Rx通常モード

USBコントローラは，USBジェネラル・ステータス・レジスタ2（U_GSR2）のEP2FO（エンドポイント2ノー・データ）ビット（ビット9）をセットします。

USBコントローラは，EP2 FIFOオーバランが発生したことを示すRx表示を書き込みます。

USBコントローラは，システム・メモリのデータ・バッファにダミー・データを書き込みません。

(b) Rxアセンブル・モード

USBコントローラは，USBジェネラル・ステータス・レジスタ2（U_GSR2）のEP2FO（エンドポイント2ノー・データ）ビット（ビット9）をセットします。

受信データとダミー・データの合計が最大パケット・サイズと等しくなるように，USBコントローラは，データ・バッファにダミー・データを書き込みます（実際には，USBコントローラは，最大パケット・サイズでデータ・バッファを番地づけするポインタをインクリメントするだけです。DMA転送は発生しません）。

USBコントローラが“USBショート・パケット”を受信後，USBコントローラはEP2 FIFOオーバランが発生したことを示すRx表示を書き込みます。

(c) Rxセパレート・モード

USBコントローラは，USBジェネラル・ステータス・レジスタ2（U_GSR2）のEP2FO（エンドポイント2ノー・データ）ビット（ビット9）をセットします。

受信データとダミー・データの合計が最大パケット・サイズと等しくなるように，USBコントローラは，データ・バッファにダミー・データを書き込みます（実際には，USBコントローラは，最大パケット・サイズでデータ・バッファを番地づけするポインタをインクリメントするだけです。DMA転送は発生しません）。

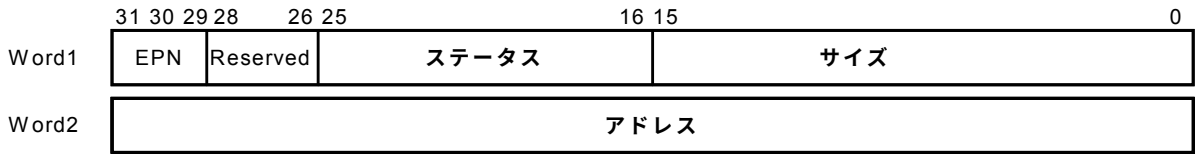
USBコントローラはEP2 FIFOオーバランが発生したことを示すRx表示を書き込みます。

6. 6. 10　Rx表示

受信するデータ・セグメントごとに，USBコントローラは受信メールボックスにRx表示を書き込みます。Rx表示を書き込んだあと，USBコントローラはUSBジェネラル・ステータス・レジスタ1（U_GSR1）の受信終了ビットを1にセットし，マスクされていなければ，割り込みを発行します。

Rx表示のフォーマットを以下に示します。

図6－26　Rx表示のフォーマット



備考　ビット28-26はリザーブ・ビットです。

EPN：エンドポイント・ナンバを示すフィールドです。

001：エンドポイント0

011：エンドポイント2

101：エンドポイント4

111：エンドポイント6

保守／廃止

ステータス： データ受信時の状態を示すフィールドです。

ビット25：　0がセットされると，データの消失がエンドポイント2で発生しなかったことを示します。
1がセットされると，データの消失がエンドポイント2で発生したことを示します。

ビット24：　0がセットされると，内部バス（IBUS）にてエラーが発生しなかったことを示します。
1がセットされると，内部バス（IBUS）にてエラーが発生し異常終了したことを示します。
このビットが1にセットされた場合，アドレス・フィールドは意味を持ちません。

ビット23：　0がセットされると，受信データがUSBコンフィギュレーション・パケット以外であることを示します。
1がセットされると，受信データはUSBコンフィギュレーション・パケットであることを示します。
このビットは，コントロール・エンドポイント（エンドポイント0）からデータを受け取ったときにのみ有効となります。ほかのエンドポイントからデータを受け取った場合は，このビットはセットされません。

ビット22：　0がセットされると，バッファ・オーバランが発生しなかったことを示します。
1がセットされると，バッファ・オーバランが発生したことを示します。
このビットは，エンドポイント1からデータを受け取ったときにのみ有効となります。

ビット21：　リザーブ・ビット。将来の使用のため予約

ビット20：　0がセットされると，CRCエラーが発生しなかったことを示します。
1がセットされると，CRCエラーが発生したことを示します。
アイソクロノス・エンドポイント（エンドポイント2）からデータを受け取ったときに，このビットが1にセットされると，システム・メモリに格納されたデータにCRCエラーがあることを示します。
このビットは，エンドポイント2からデータを受け取ったときにのみ有効となります。
アセンブル・モードでは，このビットは最後に受信したUSBパケットにエラーがある場合のみセットされます。

ビット19：　0がセットされると，ビット・スタッフィング・エラーが発生しなかったことを示します。
1がセットされると，ビット・スタッフィング・エラーが発生したことを示します。
アイソクロノス・エンドポイント（エンドポイント2）からデータを受け取ったときに，このビットが1にセットされると，システム・メモリに格納されたデータにビット・スタッフィング・エラーがあることを示します。
このビットは，エンドポイント2からデータを受け取ったときにのみ有効となります。
アセンブル・モードでは，このビットは最後に受信したUSBパケットにエラーがある場合のみセットされます。

ビット18：　0がセットされると，受信データのサイズが65535バイト以下であることを示します。
1がセットされると，受信データのサイズが65535バイトを越えることを示します。

ビット17-16：　00または01がセットされると，データを通常モードで受信したことを示します。
10がセットされると，データをアセンブル・モードで受信したことを示します。
11がセットされると，データをセパレート・モードで受信したことを示します。
エンドポイント0とエンドポイント6では，このフィールドは00にセットされています。

サイズ：受信データのサイズを示します。

受信データのサイズが65535（FFFFH）バイトを越えると，ビット18が1にセットされ，このフィールドは65535（FFFFH）を表示します。

アドレス：受信データを格納するバッファ・ディスクリプタのスタート・アドレスを示します。

保守／廃止

6.7　パワー・マネジメント

USBコントローラには，ホストPCからサスペンド信号またはリジューム信号を受信したことをVR4120Aに通知する割り込み機能があります。VR4120Aはサスペンドまたはリジュームを受信したときは，適切な処理をしなければなりません。

また，USBコントローラはμPD98502に接続しているポート自体がサスペンド状態（すなわちμPD98502自身がサスペンド状態）にあるときに，サスペンド状態をリジューム状態に切り替えるためのリモート・ウェークアップ信号を発行する機能があります。

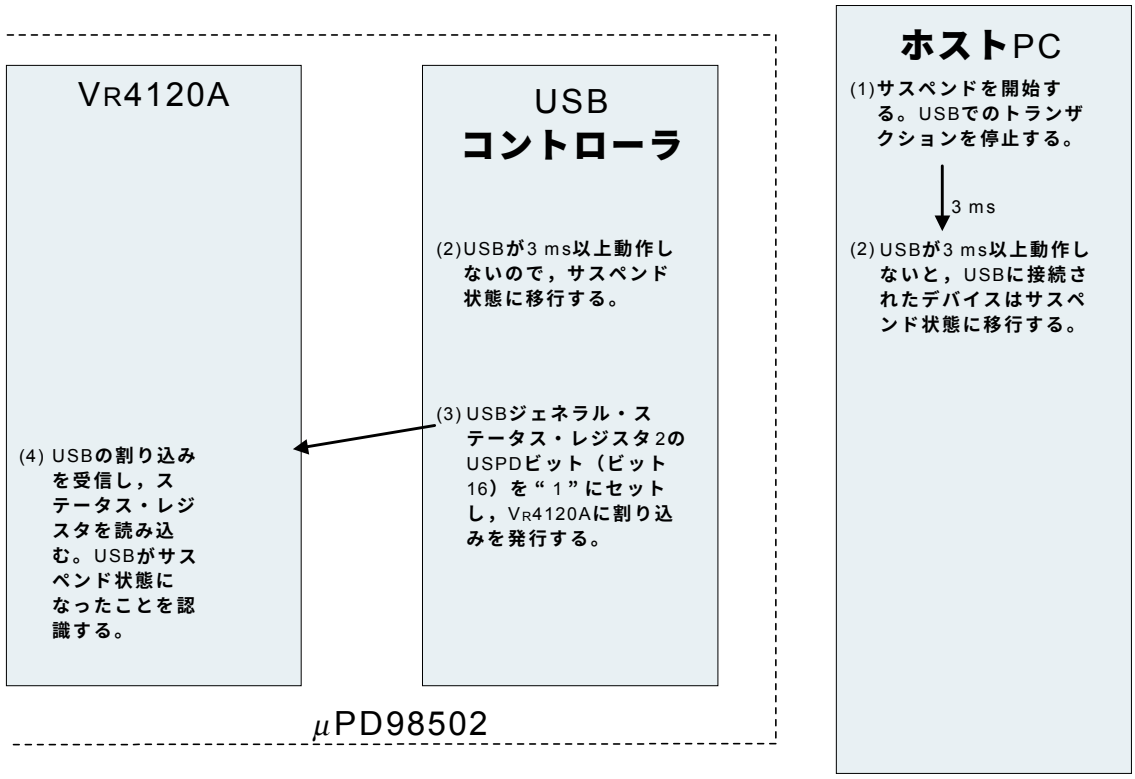
結果として，μPD98502がサスペンド状態にあっても，他のラインからのデータ（ATMを経由して送受信されるADSLデータなど）は廃棄されず，ホストPCに送ることができます。

各シーケンスについて以下のセクションにて記述します。

6.7.1　サスペンド

サスペンド・シーケンスを以下に示します。

図6－27　サスペンド・シーケンス



(1) ホストPCはUSBをサスペンド状態にします。USB上のトラフィックを停止します。

(2) 3 ms後，USB上のトラフィックはありません。したがって，USBに接続されたすべてのデバイスはサスペンド状態に移行します。同様に，USBコントローラもサスペンド状態に入ります。ただし，DMA転送中であった場合は，USBコントローラは，DMA転送が終了するまでサスペンド状態に移行しません。

(3) USBコントローラは，USBジェネラル・ステータス・レジスタ2（U_GSR2）（アドレス：1000_1018H）のUSPDビット（ビット16）を1にセットし，割り込みがマスクされていなければ，VR4120Aに割り込みを発行します。

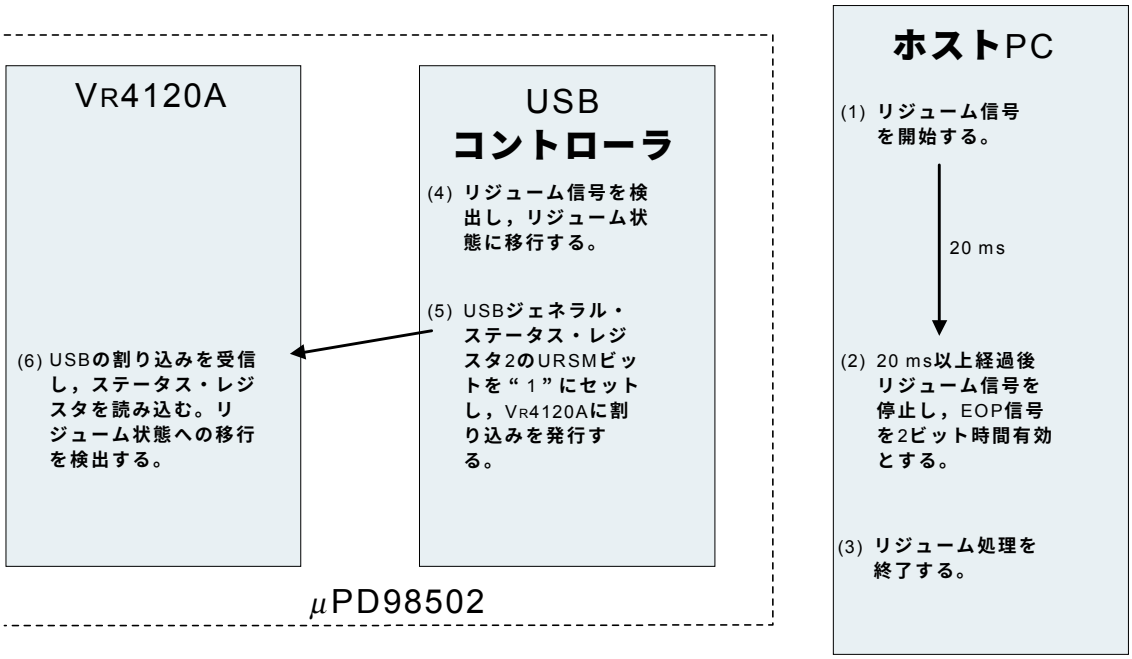
(4) VR4120Aは，USBコントローラからの割り込みを検出し，USBジェネラル・ステータス・レジスタ2（U_GSR2）を読み込み，その結果，USBがサスペンド状態にあることを判断します。

USBコントローラがサスペンド状態の間，VR4120Aは，USBコントローラのUSBジェネラル・モード・レジスタ（U_GMR）とUSB割り込みマスク・レジスタ2（U_IMR2）以外にデータを書き込むことを許されません。これが守られないと，USBコントローラがリジューム状態に入ったあと，その後の動作は予測できないものになる可能性があります。

6.7.2　リジューム

リジューム・シーケンスを以下に示します。

図6－28　リジューム・シーケンス



(1) ホストPCはリジューム信号を開始します。リジューム信号は，USBに接続されているすべてのデバイスに送信されます。

(2) 20 ms以上が経過したあと，ホストPCはリジューム信号を停止し， EOP信号を2ビット時間有効とします。

(3) ホストPCがリジューム処理を終了します。

(4) USBコントローラはリジューム信号を検出します。

(5) USBコントローラは，USBジェネラル・ステータス・レジスタ2（アドレス：1000_1018H）のURSMビット（ビット18）を1にセットし，VR4120Aに割り込みを発行します。クロックが停止すると，VR4120Aは“URSM”ビットの代わりに“usbwakeup_p”信号をチェックしなければなりません。

(6) VR4120Aは，USBコントローラからの割り込みを検出し，その結果，USBがリジューム状態に入ったことを判断します。

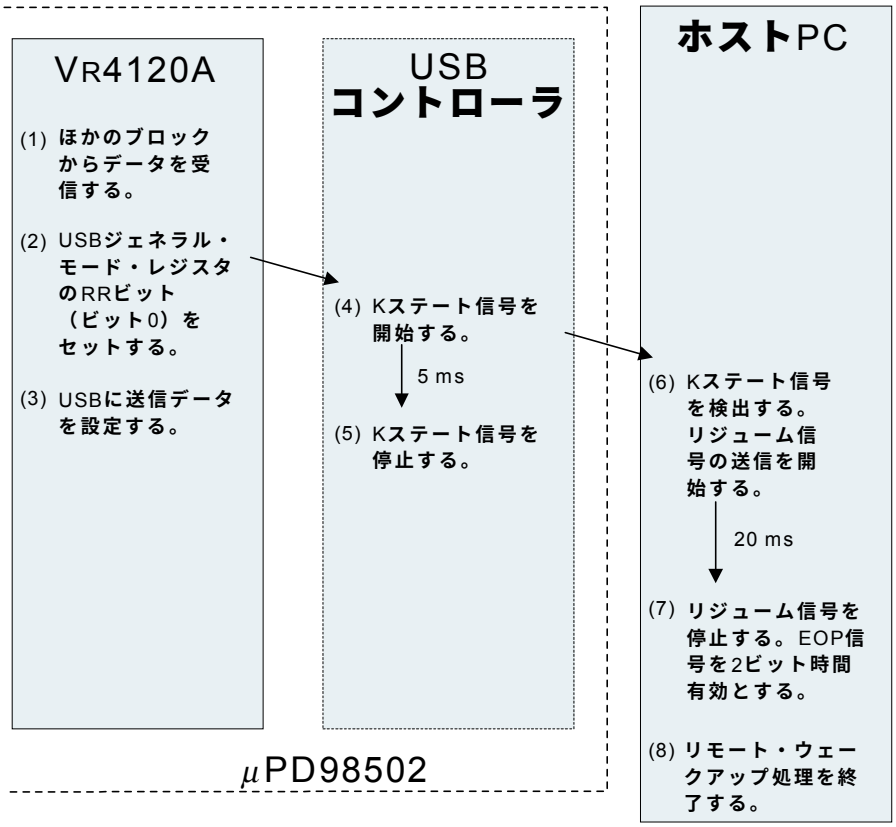
保守／廃止

USBコントローラは，リジューム状態に入ったあと，サスペンド状態に入る直前に行っていた送信／受信処理を継続します。

6. 7. 3　リモート・ウェークアップ

リモート・ウェークアップ・シーケンスを以下に示します。

図6－29　リモート・ウェークアップ・シーケンス



(1) ここでは，USBがサスペンド状態にあると仮定します。ほかのブロックからデータを受信します。

(2) VR4120Aは，サスペンド状態にあるUSBをリジューム状態に切り替えるために，USBジェネラル・モード・レジスタ（U_GMR）のRRビット（ビット0）をセットします。

(3) USBジェネラル・モード・レジスタ（U_GMR）のRRビットがセットされると，USBにKステート信号を送信します。

(4) VR4120Aは，USBに送信データを設定することが継続してできます。具体的にいうと，VR4120Aは送信データをシステム・メモリに準備し，USBコマンド・レジスタ（アドレス：1000_1040H）とUSBコマンド拡張レジスタ（アドレス：1000_1044H）にデータを書き込みます。

(5) USBコントローラは，Kステート信号を5 ms間続け，その後停止します。

(6) ホストPCは，Kステート信号を受信すると，リジューム信号を送信します。このリジューム信号は 20 ms以上継続します。

(7) 20 ms以上が経過すると，ホストPCはリジューム信号を停止し， EOP信号を2ビット時間有効とします。

(8) このシーケンスの結果，リモート・ウェークアップ処理が終了し，トランザクションを再開します。

保守／廃止

6. 8　SOFパケットの受信

USBコントローラはSOFパケットを受信し，フレーム・ナンバが正しくインクリメントしているかどうかをチェックできます。

さらに，USBコントローラはSOFパケットのタイミング・スキューを検出できます。

6. 8. 1　SOFパケットの受信とフレーム・ナンバの更新

USBコントローラはSOFパケットを受信後，USBフレーム・ナンバ／バージョン・レジスタ（アドレス：1000_1004H）のFNフィールドを更新します。FNフィールドを更新後，USBジェネラル・ステータス・レジスタ2（アドレス：1000_1018H）のFWビット（ビット21）を1にセットします。

6. 8. 2　フレーム・ナンバの自動更新

受信したSOFパケットが不正なフレーム・ナンバをもっていると，USBコントローラは以下に示す2つのプロセスのうちの1つをU_GMRのAUビットに対応して実行します。

- USBコントローラは，不正なフレーム・ナンバをFNフィールドに直接反映します。
- USBコントローラは，フレーム・ナンバを自動的にインクリメントし，インクリメントした値をFNフィールドに書き込みます。

FNフィールドを更新するポリシーを以下の表に示します。

ケース	AUビット[注]	FNフィールド	USBジェネラル・ステータス・レジスタ2
正しいSOF	0	受信したフレーム・ナンバをロードするのみ	FWビットを1にセットする
	1	受信したフレーム・ナンバをロードするのみ	FWビットを1にセットする
SOFのロス	0	更新しない	SLビットを1にセットする
	1	現在のFNをインクリメントする	FWビットとSLビットを1にセットする
余分なSOF	0	更新しない	ESビットを1にセットする
	1	更新しない	ESビットを1にセットする
ビット・スタッフ・エラー	0	受信したフレーム・ナンバをロードするのみ	FWビットを1にセットする
	1	現在のFNをインクリメントする	FWビットを1にセットする
CRCエラー	0	受信したフレーム・ナンバをロードするのみ	FWビットを1にセットする
	1	現在のFNをインクリメントする	FWビットを1にセットする
不正なフレーム・ナンバ	0	受信したフレーム・ナンバをロードするのみ	FWビットを1にセットする
	1	現在のFNをインクリメントする	FWビットを1にセットする

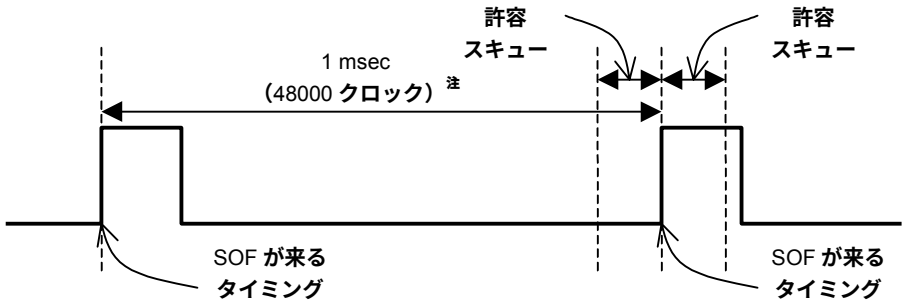
注　AUビットはUSBジェネラル・モード・レジスタ（アドレス：1000_1000H）にあります。

保守／廃止

6. 8. 3　SOF到着時刻のスキューが許容されるかどうかのチェック

許容SOFスキューは，USBジェネラル・モード・レジスタ（アドレス：1000_1000H）のSOFINTVLフィールドで定義できます。

図6－30　SOFに対する許容スキュー



注　USBクロック入力12 MHzを4逓倍したクロックをソースとしています。

SOFINTVLフィールドは，ディフォールトで18H（24クロック）に設定されています。これは，48000クロック（1 msec）の0.05%です。

SOFINTVLの値は，最初のSOFパケットが来る前までに設定されなければなりません。

保守／廃止

6.9　ループバック・モード

USBコントローラには，テストを目的とした内蔵のループバック機能があります。

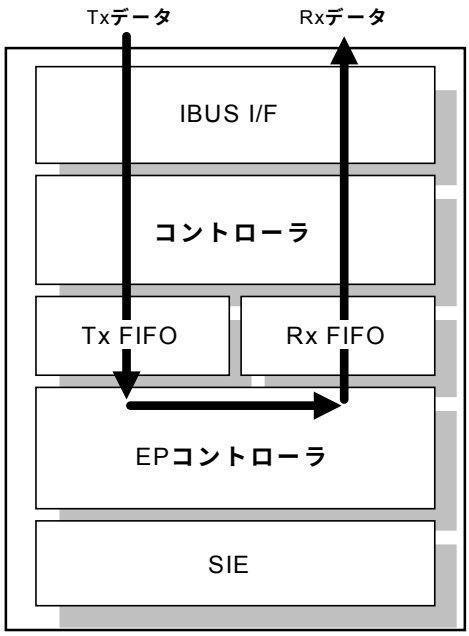
ループバック機能をイネーブルにするには，USBジェネラル・モード・レジスタ（U_GMR）のLEビット（ビット1）を1にセットしてください。

ループバック機能が動作状態に入ると，USBコントローラはデータをシステム・メモリから得てTx FIFOに置きます。データはエンドポイント・コントローラによって返されます。返されたデータは，システム・メモリに返されたあとに，Rx FIFOに書き込まれます。

送信／受信は通常の設定で実行されなければなりません。Tx/Rx表示は通常どおり報告されます。

内部データ・フローを以下に示します。

図6－31　ループバック・モードでのデータ・フロー



図に示すように，ループバック・モードでは，USB上へのデータの送信／受信は行われません。すべてのデータはエンドポイント・コントローラによって返されます。

データの送信／受信に使用するエンドポイントを以下の表に示します。

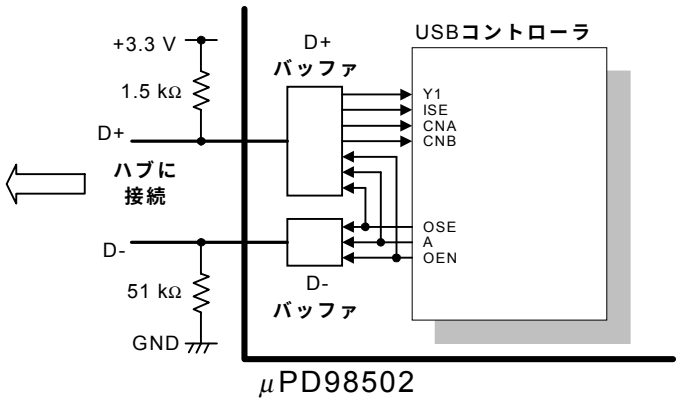
Txエンドポイント	Rxエンドポイント
エンドポイント0（コントロール）	エンドポイント0（コントロール）
エンドポイント1（アイソクロノス）	エンドポイント2（アイソクロノス）
エンドポイント3（バルク）	エンドポイント4（バルク）
エンドポイント5（インタラプト）	エンドポイント6（インタラプト）

保守／廃止

6. 10　接 続 例

USBコントローラは，以下の図6－32に示すように，μPD98502の内部USB I/Oバッファに接続されています。

図6－32　接 続 例



PCBを設計するときは，フルスピード・デバイスの存在を示すために，D+端子と3.3 V電源の間に1.5 kΩのプルアップ抵抗を接続する必要があります。

内蔵されたUSBバッファ上で電流のフローティングを避けるために，D-端子とGNDの間に51 kΩのプルダウン抵抗を接続することを推奨します。

μPD98502のUSB用電源は外部の3.3 V電源と一緒にオン／オフするように回路を設計しなければなりません。μPD98502のUSB用電源がオフで，外部の3.3 V電源がオンだと，μPD98502に接続しているUSBのハブは，新しいデバイスが接続していると認識することはできません。なぜなら，μPD98502のUSB用電源がオフだと応答が返信されてこないからです。

USBの電気的特性の詳細については，USB仕様1. 1を参照してください。

保守／廃止

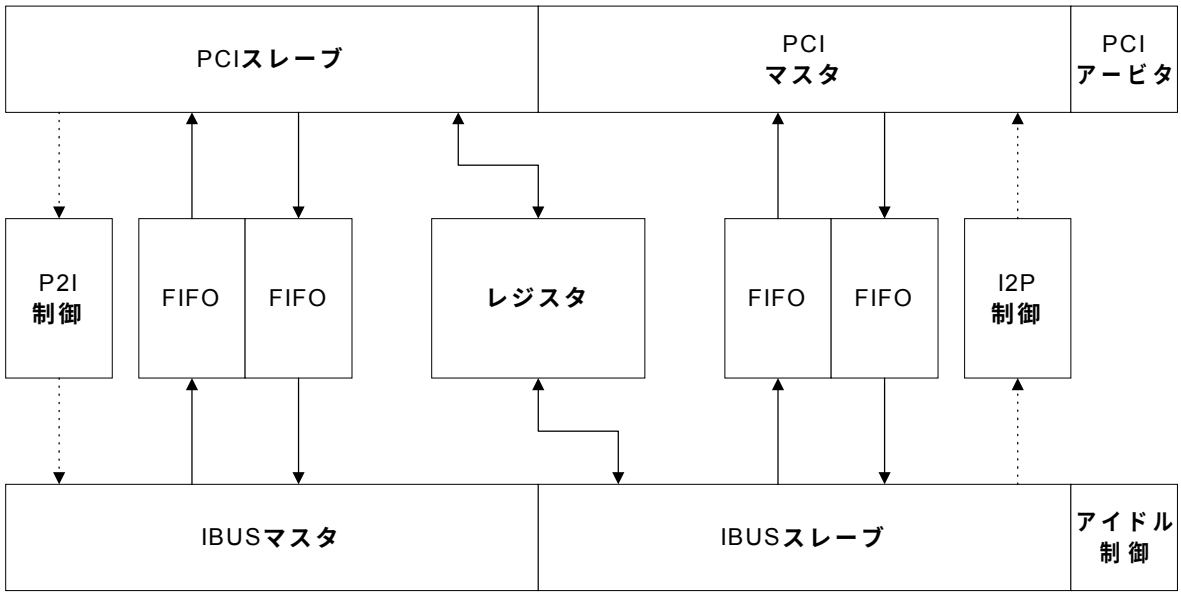
第7章　PCIコントローラ

7.1 概　　要

PCIコントローラは，NICモードとホスト・モードの両方をサポートします。NICモード（PMODE：ハイ）では，PCIコントローラはコンフィギュレーション・サイクルの実行および，アービトレーション機能をイネーブルとすることはできません。ホスト・モード（PMODE：ロウ）では，PCIコントローラはコンフィギュレーション・サイクルを実行でき，アービトレーション機能をイネーブルとする（PARBN：ハイ）ことができます。PCIコントローラの特徴は次のとおりです。

- 32ビットPCIインタフェース
- 33 MHzのPCI周波数能力
- PCIローカル・バス仕様rev.2.2に準拠
- PCIバス・パワー・マネジメント・インタフェース（PPMI）rev.1.1に準拠
- 最大16ワードのバースト・アクセスをサポート
- ポステッド・ライト機能をサポート
- ディレイド／ノン・ディレイド・リード／ライト・サイクルをサポート
- ホスト・モードで最大4台の外部PCIバス・マスタ・デバイスのアービトレーションを行うPCIバス・アービタを内蔵

図7－1　PCIコントローラのブロック図



――――― データ・フロー

・・・・・・・・・・・・・制御

保守／廃止

7.2　バス・ブリッジ機能

7.2.1　内部バスからPCIへのトランザクション

7.2.1.1　ウインドウ・サイズ

PCIコントローラは，内部メモリ・スペースに2 Mバイトのアクセス・ウインドウをもつことができます。VR4120Aは，アクセス・ウインドウを介して外部PCIデバイスにアクセスできます。VR4120Aから見たアクセス・ウインドウは，1010_0000Hから102F_FFFFHまでのメモリ範囲に配置されています。PCIアドレス・スペースのベース・アドレスは，P_PLBAレジスタ（ホスト・モード時）あるいはPCIコンフィギュレーション・スペース内のウインドウ・メモリ・ベース・アドレス・レジスタ（NICモード時）の設定で定義します。

7.2.1.2　PCIマスタ

PCIコントローラは，メモリ・コマンド・サイクルのみを実行できます。I/Oコマンド・サイクル，割り込みアクノリッジ・コマンド・サイクル，スペシャル・サイクル・コマンド・サイクルはサポートしません。PCIコントローラは，ホスト・モードではコンフィギュレーション・サイクルを実行することもできます。

キャッシュ・ライン・サイズ・レジスタが有効（キャッシュ・ライン・サイズが4, 8, 16, 32であることを意味する）である場合，PCIコントローラは，PCI仕様にて規定されたリード・トランザクションに対する3種類のPCIメモリ・コマンドを実行できます。

メモリ・リード　　：キャッシュ・メモリのライン・サイズよりも小さなメモリにアクセスする場合
メモリ・リード・ライン　：1つのキャッシュ・ラインにアクセスする場合
メモリ・リード・マルチプル：複数のキャッシュ・ラインに連続的にアクセスする場合

キャッシュ・ライン・サイズ・レジスタが有効でない場合は，PCIコントローラは常にメモリ・リード・コマンドのみを実行します。

PCIコントローラは，以下に示す条件をすべて満たせば，ライト・トランザクションとしてメモリ・ライト・アンド・インバリデートコマンドを使用します。

- 転送ワード数がキャッシュ・ライン・サイズのちょうど倍数である。
- コンフィギュレーション・レジスタの“メモリ・ライト・アンド・インバリデート”ビットが“1”にセットされている。
- ライト・トランザクションのスタート・アドレスがキャッシュ境界にある。

保守／廃止

7. 2. 1. 3　内部バスからPCIへの書き込み

(1) ポステッド・ライト・トランザクション

P_BCNTレジスタのIPWRDビットが“0”の場合，PCIコントローラは，内部バス側からPCI側へのライト・トランザクションに対して“ポステッド・ライト・トランザクション”を適用します。その動作は次のとおりです。

＜1＞内部バス（IBUS）に接続された内部バス・ブロック[注]は，ライト・トランザクションをPCIコントローラを介してPCIターゲット・デバイスに発行します。

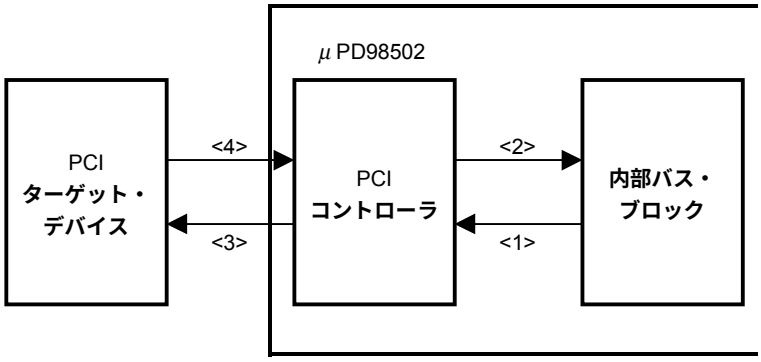
＜2＞PCIコントローラはこのアクセスを受け付け，書き込むデータを内部FIFOに格納します。μPD98502内部でのトランザクションはこの時点で終了します。そして，PCIコントローラは，PCIバスのライト・トランザクションが終了するまでは，内部バス・ブロックが新たに行うPCIターゲット・デバイスへの後続のライト・アクセスに対して“リトライ”を発行します。

＜3＞PCIコントローラは，ライト・トランザクションをPCIターゲット・デバイスに対して発行します。

＜4＞トランザクションの終了後，PCIコントローラは内部バス・ブロックからの新しいライト・トランザクションを再び受け付けることができます。

注　内部バス・ブロックとは，USBコントローラのように内部バス（IBUS）に接続されているブロックを指します。PCIコントローラ以外のすべての内部バス・デバイスは，アクセス・サイクルを外部PCIデバイスに発行できます。VR4120Aもシステム・コントローラを介してPCIターゲット・デバイスに対してアクセス・サイクルを発行できます。

図7－2　内部バスからPCIへのポステッド・ライト・トランザクション



最大バースト・サイズは16ワードです。16ワードを越えるライト・バーストが内部バスで発行されると，PCIコントローラは，16番目のワードで“ディスコネクト”を発行します。アドレス境界を越えてバースト・アクセスした場合も，PCIコントローラは“ディスコネクト”を発行します。

PCIコントローラは，内部バス側からポステッド・ライトを受け付けたあと，PCIバスでターゲット・アボート／マスタ・アボートを検出すると，P_IGSRレジスタのWRTAT/WRMATビットとP_PGSRレジスタのRTABT/RMABTビットをセットし，マスクされていなければ，割り込みをPCIホストとVR4120Aに対して発行します。内部FIFO内のデータは廃棄されます。そして，PCIコントローラは，内部バスの新しいライト・トランザクションを受け付けることができます。

保守／廃止

(2) ノン・ポステッド・ライト・トランザクション

P_BCNTレジスタのIPWRDビットが“1”の場合，PCIコントローラは，内部バス側からPCI側へのライト・トランザクションに対して“ノン・ポステッド・ライト・トランザクション”を適用します。このモードでは，バースト転送は1つのワードごとに分割されます。その動作は次のとおりです。

＜1＞内部バス（IBUS）に接続された内部バス・ブロック[注]は，ライト・トランザクションをPCIコントローラを介して外部PCIターゲット・デバイスに発行します。PCIコントローラは，バースト・データの最初のワードをラッチします。ラッチしたデータをPCIターゲット・デバイスに書き込むまでPCIコントローラは内部バス・ブロックに対してウエイト・サイクルを挿入します。

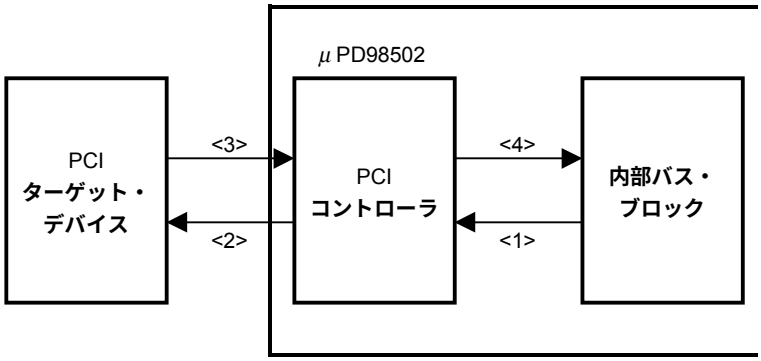
＜2＞PCIコントローラは，ライト・トランザクションを外部PCIターゲット・デバイスに発行します。

＜3＞PCIターゲット・デバイスはアクセスを受け付けます。

＜4＞PCIトランザクションの完了後，PCIコントローラは，バースト転送によるライト・アクセスを続けようとする内部バス・ブロックに対して“ディスコネクト”を発行します。そのため内部バス・ブロックは，できるだけ早くトランザクションを終了しなければなりません。

注　内部バスの特徴はこのマニュアルでは記述していませんが，内部バスはPCIバスと似た構造となっており，内部バスでも“ディスコネクト”機能をサポートしています。

図7－3　内部バスからPCIへのノン・ポステッド・ライト・トランザクション



内部バス・ブロックはさらに多くのワードを転送したい場合，追加のライト・トランザクションを発行しなければなりません。

PCIコントローラは，内部バス側からノン・ポステッド・ライトを受け付けたあと，PCIバスでターゲット・アボート／マスタ・アボートを検出すると，P_IGSRレジスタのWRTAT/WRMATビットとP_PGSRレジスタのRTABT/RMABTビットをセットし，マスクされていなければ，割り込みを外部PCIホスト・デバイスとVR4120Aに発行します。その際，PCIコントローラは，ライト・データを廃棄します。そして，その後，PCIコントローラは，内部バス・ブロックからの新しいライト・トランザクションを再び受け付けることができます。

保守／廃止

7. 2. 1. 4　内部バスからPCIへの読み込み

(1) ディレイド・リード・トランザクション

P_BCNTレジスタのIDRTDビットが“0”の場合，PCIコントローラは，内部バス側からPCI側へのリード・トランザクションに対して“ディレイド・リード・トランザクション”を適用します。その動作は次のとおりです。

＜1＞内部バスに接続する内部バス・ブロックは，リード・トランザクションをPCIコントローラを介して外部PCIターゲット・デバイスに発行します。

＜2＞PCIコントローラは，このアクセスに応答し，“リトライ”を内部バス・ブロックに発行します。一方，PCIコントローラはアドレスとコマンドをラッチし，発行されたアクセスを格納します。次に，PCIコントローラは，PCIバスのラッチされたコマンドに対応するトランザクションが終了するまで“リトライ”をすべてのアクセスに発行します。

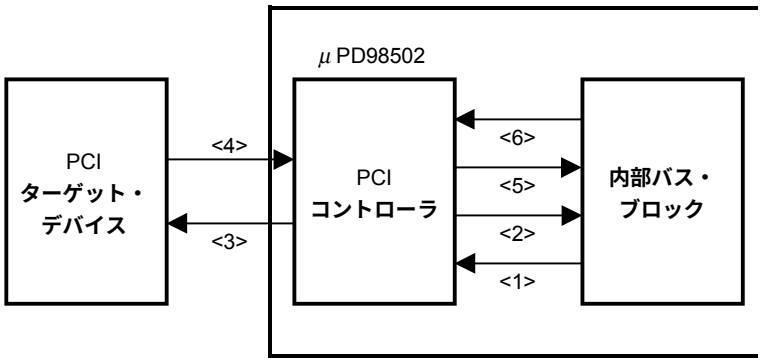
＜3＞PCIコントローラは，リード・トランザクションをPCIターゲット・デバイスに発行します。

＜4＞PCIターゲット・デバイスはアクセスを受け付け，PCIターゲット・デバイスからのリード・データを内部FIFOに格納します。

＜5＞PCIコントローラは，＜1＞で発行されたのと同じアクセスが内部バス・ブロックから来るのを待ちます。PCIコントローラは，それ以外のアクセスに対しては“リトライ”を発行します。

＜6＞＜1＞と同じアクセス（同一アドレスと同一コマンドのアクセス）が来ると，PCIコントローラはこのアクセスを受け付け，内部FIFOのデータを内部バス・ブロックへ転送します。

図7－4　内部バスからPCIへのディレイド・リード・トランザクション



最大バースト・サイズは16ワードです。16ワードを越えるリード・バースト要求が内部バス・ブロックから発行された場合，PCIコントローラは，16番目のワードで“ディスコネクト”を発行します。アドレス境界を越えてバースト・アクセス要求した場合も，PCIコントローラは“ディスコネクト”を発行します。

＜1＞で発行されたのと同じリード・アクセスが2^{15}クロック以内に来ないと，PCIコントローラは，内部FIFO内のデータを廃棄し，P_PGSRレジスタのPFDSCビットをセットし，（マスクされていなければ）割り込みでPCIホストに報告します。その後，PCIコントローラは，内部バス・ブロックからの新しいリード・トランザクションを再び受け付けることができます。

PCIコントローラは，PCI側からディレイド・リードを受け付けたあと，PCIバスでターゲット・アボート／マスタ・アボートを受け取ると，P_IGSRレジスタのRDTAT/RDMATビットとP_PGSRレジスタのRTABT/RMABTビットをセットし，マスクされていなければ，割り込みで外部PCIホスト・デバイス

とVR4120Aに報告します。次に，PCIコントローラは，すべてのビットが“0”であるワードを内部バス・ブロックに転送します。

(2) ノン・ディレイド・リード・トランザクション

P_BCNTレジスタのIDRTDビットが“1”の場合，PCIコントローラは，内部バス側からPCI側へのリード・トランザクションに対して“ノン・ディレイド・リード・トランザクション”を適用します。このモードでは，バースト転送は1ワードごとの転送に分割されます。その動作は次のとおりです。

＜1＞内部バスに接続された内部バス・ブロックは，リード・トランザクションをPCIコントローラを介して外部PCIターゲット・デバイスに発行します。

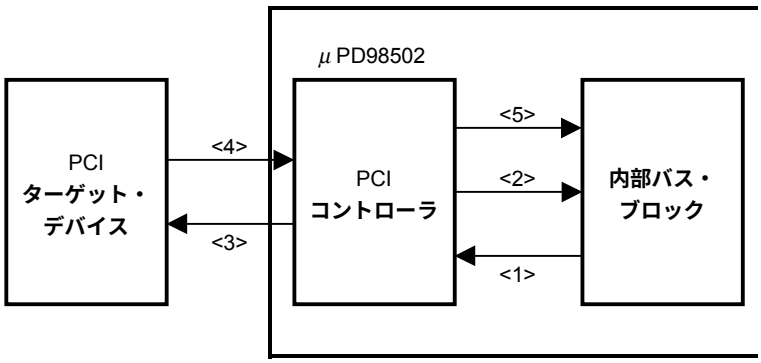
＜2＞PCIコントローラは，内部バスの最初のデータ転送を開始する前にウエイト・サイクルを挿入します。

＜3＞PCIコントローラは，リード・トランザクションを外部PCIターゲット・デバイスに発行します。

＜4＞PCIターゲット・デバイスはこのアクセスを受け付けます。

＜5＞PCIコントローラは，読み込んだ最初の1ワード・データを内部バス・ブロックに転送します。同時に，PCIコントローラは，内部バス・ブロックがバースト・リード・トランザクションを続けようとすると，内部バス・ブロックに“ディスコネクト”を発行します。内部バス・ブロックは，その際，できるだけ早くトランザクションを終了しなければなりません。

図7－5　内部バスからPCIへのノン・ディレイド・リード・トランザクション



PCIコントローラは，外部PCIターゲット・デバイスからノン・ディレイド・リードを受け付けたあと，PCIバスでターゲット・アボート／マスタ・アボートを検出すると，P_IGSRレジスタのRDTAT/RDMATビットとP_PGSRレジスタのRTABT/RMABTビットをセットし，マスクされていなければ，割り込みを外部PCIホスト・デバイスとVR4120Aに発行します。次に，PCIコントローラは，すべてのビットが“0”であるワードを内部バス・ブロックに転送し，内部バス・ブロックがバースト転送要求を発行すると，“ディスコネクト”を発行します。

保守／廃止

7.2.2　PCIから内部バスへのトランザクション

7.2.2.1　ウインドウ・サイズ

PCIコントローラは，2 Mバイトのアドレス・スペースを，PCI側のPCIメモリ・スペースから内部バス側へのアクセス・ウインドウとしてサポートします。ウインドウのベース・アドレスは，NICモードでは外部PCIホスト・デバイスによって，コンフィギュレーション・スペースのウインドウ・メモリ・ベース・アドレス・レジスタに設定されます。ホスト・モードでは，VR4120Aがこのレジスタにベース・アドレスを書き込み設定しなければなりません。

PCIコントローラの内部レジスタが，PCIコントローラのこのアドレス領域を含んでいたとしても，このウインドウを介した読み込み／書き込みはできません。コンフィギュレーション・スペースのレジスタ・メモリ・ベース・アドレス・レジスタに書き込まれたレジスタ用のベース・アドレスは，外部PCIホストからのPCIコントローラの内部レジスタへの読み込み／書き込みに使用します。

発行されたバースト転送がこのアドレス・スペースの境界を越えると，PCIコントローラは境界で“ディスコネクト”を発行します。

7.2.2.2　アクセス・タイプ

(1) PCIターゲット

PCIコントローラが受け付け可能なPCIコマンドを以下に示します。

C/BE#［3：0］	PCIコマンド	PCIターゲット
0000	割り込みアクノリッジ	無視される
0001	スペシャル・サイクル	無視される
0010	I/Oリード	無視される
0011	I/Oライト	無視される
0100	Reserved	-
0101	Reserved	-
0110	メモリ・リード	受け付けられる
0111	メモリ・ライト	受け付けられる
1000	Reserved	-
1001	Reserved	-
1010	コンフィギュレーション・リード	受け付けられる
1011	コンフィギュレーション・ライト	受け付けられる
1100	メモリ・リード・マルチプル	受け付けられる
1101	デュアル・アドレス・サイクル	無視される
1110	メモリ・リード・ライン	受け付けられる
1111	メモリ・ライト・アンド・インバリデート	受け付けられる

(2) 内部バス

P_BCNTレジスタのICMDSビットが“0”の場合，PCIコントローラは，内部バス・ブロックに対してI/Oコマンドを使って，データ転送を行います。ICMDSビットが“1”の場合，PCIコントローラは，内部バス・ブロックに対してメモリ・コマンドを使用します。

保守／廃止

7. 2. 2. 3　PCIから内部バスへの書き込み

(1) ポステッド・ライト・トランザクション

P_BCNTレジスタのPPWRDビットが“0”の場合，PCIコントローラは，内部バス側からPCI側へのライト・トランザクションに対して“ポステッド・ライト・トランザクション”を適用します。その動作は次のとおりです。

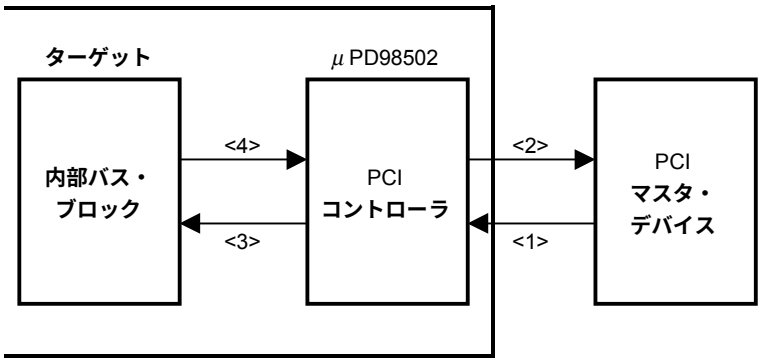
＜1＞PCIマスタ・デバイスは，ライト・トランザクションをターゲットとなる内部バス・ブロックに発行します。

＜2＞PCIコントローラはこのアクセスを受け付け，書き込みデータを内部FIFOに格納します。PCIバスのトランザクションはこの時点で終了します。一方，PCIコントローラは，内部バスのトランザクションが終了するまで，PCIマスタ・デバイスからPCIコントローラへのすべてのPCIライト・トランザクションに対して“リトライ”を返します。

＜3＞PCIコントローラは，ライト・トランザクションを内部バス・ブロックに発行します。

＜4＞内部バス・ブロックはこのアクセスを受け付け，データ転送を行い，その後PCIコントローラは内部バス・ブロックに対するライト・トランザクションを終了します。トランザクションの終了後，PCIコントローラはPCIマスタ・デバイスからの新しいライト・アクセスを再び受け付けることができます。

図7－6　PCIから内部バスへのポステッド・ライト・トランザクション



最大バースト・サイズは16ワードです。16ワードを越えるライト・バーストがPCIマスタ・デバイスで発行されると，PCIコントローラはそれを受け付け，16番目のワードで“ディスコネクト”を発行します。PCIコントローラは，アドレス境界を越えた場合にも“ディスコネクト”を発行します。

PCIコントローラは，PCI側からポステッド・ライトを受け付けたあと，内部バスでバス・エラーを検出すると，P_IGSRレジスタのIWBERビットとP_PGSRレジスタのPWBERビットをセットし，マスクされていなければ，割り込みでPCIホストとVR4120Aに報告します。その際，内部FIFO内のデータは廃棄されます。

保守／廃止

(2) ノン・ポステッド・ライト・トランザクション

P_BCNTレジスタのPPWRDビットが“1”の場合，PCIコントローラは，内部バス側からPCI側へのライト・トランザクションに対して“ノン・ポステッド・ライト・トランザクション”を適用します。このモードでは，バースト転送は1つのワードごとに分割されます。その動作は次のとおりです。

＜1＞PCIマスタ・デバイスは，ライト・トランザクションをPCIコントローラを介して内部バス・ブロックに発行します。

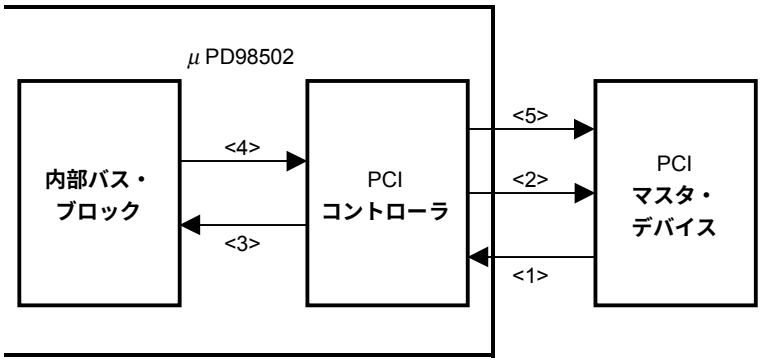
＜2＞PCIコントローラは，DEVSEL_Bをアサートしてこのアクセスに応答し，バースト・データの最初のワードをラッチします。しかし，PCIコントローラは，この時点でTRDY_Bをアサートしません。

＜3＞PCIコントローラは，ライト・トランザクションを内部バス・ブロックに発行します。

＜4＞内部バス・ブロックはこのアクセスを受け付け，PCIコントローラは内部バス・ブロックに対して＜2＞でラッチした最初のワードを書き込みます。

＜5＞PCIコントローラは，最初のデータ・フェーズが終了したことを示すためにTRDY_Bをアサートします。次に，PCIコントローラは，バースト転送を発行したときにPCIマスタ・デバイスに“ディスコネクト”を発行します。PCIマスタ・デバイスは，できるだけ早くトランザクションを終了しなければなりません。

図7－7　PCIから内部バスへのノン・ポステッド・ライト・トランザクション



PCIコントローラは，内部バスでバス・エラーを検出すると，TRDY_Bをアサートして最初のデータ・フェーズを終了し，バースト転送を発行されていればPCIマスタ・デバイスに“ディスコネクト”を発行します。次に，PCIコントローラは，P_IGSRレジスタのIWBERビットとP_PGSRレジスタのPWBERビットをセットし，マスクされていなければ，割り込みを外部PCIホスト・デバイスとVR4120Aに発行します。その際，内部FIFO内のデータは廃棄されます。

保守／廃止

7. 2. 2. 4　PCIから内部バスへの読み込み

(1) ディレイド・リード・トランザクション

P_BCNTレジスタのPDRTDビットが“0”の場合，PCIコントローラは，内部バス側からPCI側へのリード・トランザクションに対して“ディレイド・リード・トランザクション”を適用します。その動作は次のとおりです。

＜1＞PCIマスタ・デバイスは，リード・トランザクションをPCIコントローラを介して内部バス・ブロックに発行します。

＜2＞PCIコントローラは，このアクセスに応答し，“リトライ”をPCIマスタ・デバイスに発行します。その際，PCIコントローラはアドレスとコマンドをラッチし，発行されたアクセスを記憶します。一方，PCIコントローラは，内部バスの＜1＞以前に発行されたトランザクションが終了するまで“リトライ”をすべてのリード・アクセスに発行します。

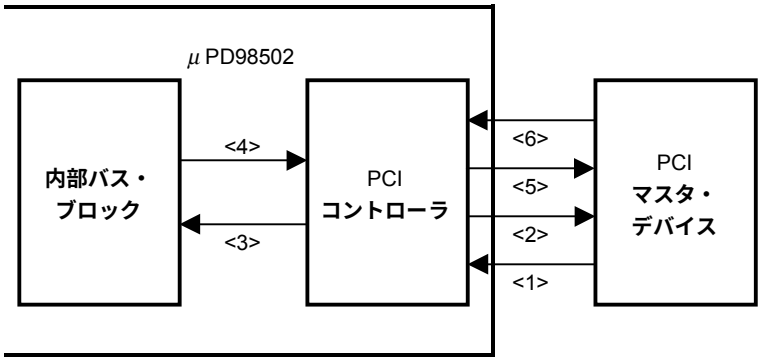
＜3＞PCIコントローラは，リード・トランザクションを内部バス・ブロックに発行します。

＜4＞内部バス・ブロックはこのアクセスを受け付け，PCIコントローラは内部バス・ブロックから内部FIFOにデータを読み込みます。

＜5＞PCIコントローラは，＜1＞で発行されたのと同じアクセスがPCIマスタ・デバイスから来るのを待ちます。PCIコントローラは，ほかのアクセスに対しては“リトライ”を発行します。

＜6＞＜1＞と同じアクセス（アドレスとコマンドが同じアクセス）が来ると，PCIコントローラはこのアクセスを受け付け，内部FIFO内のデータをPCIマスタ・デバイスへ返します。

図7－8　PCIから内部バスへのディレイド・リード・トランザクション



PCIマスタ・デバイスから発行されたリード・トランザクションのバースト・サイズは，トランザクションが終了するまで知ることができないので，PCIコントローラは，キャッシュ・ライン・サイズが有効であるときに，発行されたPCIコマンドに基づいてプリフェッチすべきワード・サイズを決定します。メモリ・リード・コマンドを使用するときは，プリフェッチすべきワード・サイズは1ワードです。メモリ・リード・ライン・コマンドのときは，サイズはキャッシュ・ライン・サイズと同じであり，メモリ・リード・マルチプル・コマンドのときは，プリフェッチすべきサイズは16ワードです。キャッシュ・ライン・サイズが有効でない場合は，プリフェッチすべきサイズは常に16ワードです。PCIマスタ・デバイスがそれ以上さらに多くのワード転送要求を発行すると，PCIコントローラは，プリフェッチしたデータをPCIマスタ・デバイスに転送したあと，“ディスコネクト”してトランザクションを終了します。PCIコントローラは，アドレス境界を越えた場合にも“ディスコネクト”を発行します。

PCIマスタ・デバイスが発行したアクセスと同じアクセスが2^{15}クロック以内に来ないと，PCIコント

ローラは，内部FIFO内のデータを廃棄し，P_IGSRレジスタのIFDSCビットをセットし，マスクされていなければ，VR4120Aに割り込みを発行します。

PCIコントローラは，PCI側からディレイド・リード・トランザクションを受け付けたあと，内部バスでバス・エラーを検出すると，P_IGSRレジスタのIRBERビットとP_PGSRレジスタのPRBERビットをセットし，マスクされていなければ，割り込みを外部PCIホスト・デバイスとVR4120Aに発行します。さらに，PCIコントローラは，PCIマスタ・デバイスからのアクセスに対して“ターゲット・アボート”を発行します。

(2) ノン・ディレイド・リード・トランザクション

P_BCNTレジスタのPDRTDビットが“1”の場合，PCIコントローラは，内部バス側からPCI側へのリード・トランザクションに対して“ノン・ディレイド・リード・トランザクション”を適用します。このモードでは，バースト転送は1ワード転送に分割されます。その動作は次のとおりです。

＜1＞PCIマスタ・デバイスは，リード・トランザクションをPCIコントローラを介して内部バス・ブロックに発行します。

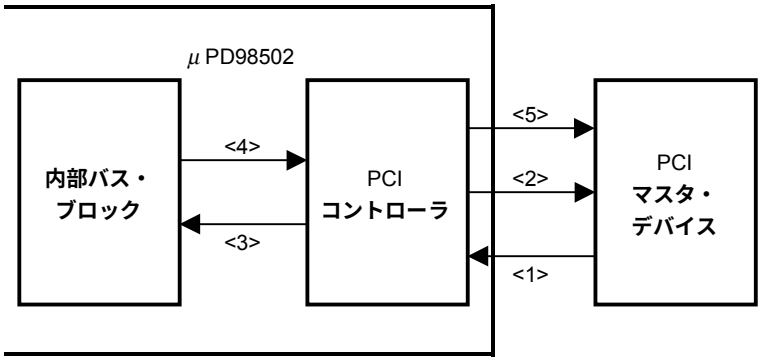
＜2＞PCIコントローラは，DEVSEL_Bをアサートしてこのアクセスに応答しますが，PCIコントローラは，この時点でTRDY_Bをアサートしません。

＜3＞PCIコントローラは，リード・トランザクションを内部バス・ブロックに発行します。

＜4＞内部バス・ブロックはこのアクセスを受け付け，PCIコントローラはそこから1ワードを読み込みます。

＜5＞PCIコントローラは，TRDY_Bをアサートし，その1ワード・データをPCIマスタ・デバイスに転送します。次に，PCIコントローラは，バースト転送を発行したときにPCIマスタ・デバイスに“ディスコネクト”を発行します。PCIマスタ・デバイスは，この際，できるだけ早くトランザクションを終了しなければなりません。

図7－9　PCIから内部バスへのノン・ディレイド・リード・トランザクション



PCIコントローラは，PCI側からノン・ディレイド・リード・トランザクションを受け付けたあと，内部バスでバス・エラーを検出すると，P_IGSRレジスタのIRBERビットとP_PGSRレジスタのPRBERビットをセットし，マスクされていなければ，割り込みで外部PCIホスト・デバイスとVR4120Aに報告します。次に，PCIコントローラは，すべての“0”のワードをPCIマスタ・デバイスに転送し，PCIマスタ・デバイスがバースト転送を発行すると，“ディスコネクト”を発行します。

保守／廃止

7.2.3　異常終了

7.2.3.1　PCIバス

(1) パリティ・エラーの検出

PCIバスからPCIコントローラに対するアクセスが行われ，PCIコントローラがターゲットとしてアドレス・パリティ・エラーを検出すると，PCIコントローラはアクセスを終了するためにターゲット・アボートを発行します。同時に，PCIコントローラは，PCIコンフィギュレーション・スペースの“パリティ・エラー検出”ビット，P_IGSRレジスタのPPERRビット，P_PGSRレジスタのDPERRビットをセットし，マスクされていなければ，割り込みを外部PCIホスト・デバイスとVR4120Aに発行します。PCIコンフィギュレーション・スペース内のコマンド・レジスタにおいて，“パリティ・エラー応答”ビット（ビット6）と“システム・エラー・イネーブル”ビット（ビット8）が“1”にセットされると，PCIコントローラは，PSERO_B信号をアサートし，PCIコンフィギュレーション・スペースのステータス・レジスタにあるシステム・エラー発信ビット，およびP_PGSRレジスタのSSERRビット（ビット7）をセットします（この場合，この不正なアドレス・データは，IBUSに対して送られる，すなわちアドレスが正しいか間違っているかにかかわらずこのアクセス自体は行われるので注意してください）。

PCIバスからPCIコントローラへのアクセスが発行され，PCIコントローラがターゲットとしてデータ・パリティ・エラーを検出すると，PCIコントローラは，コンフィギュレーション・スペースのステータス・レジスタにあるパリティ・エラー検出ビット（ビット15），P_IGSRレジスタのPPERRビット（ビット9），P_PGSRレジスタのDPERRビット（ビット8）をセットします。さらに，PCIコントローラは，マスクされていなければ，割り込みを外部PCIホスト・デバイスとVR4120Aに発行します。ただし，PCIコントローラは現在の転送を終了せず，アクセス自体は続行します。

PCIコントローラからPCIバスへアクセスを発行し，PCIコントローラがPCIマスタ・デバイスとしてデータ・パリティ・エラーを検出すると，PCIコントローラは，コンフィギュレーション・スペースのステータス・レジスタにあるパリティ・エラー検出ビット（ビット15），P_IGSRレジスタのPPERRビット（ビット9），P_PGSRレジスタのDPERRビット（ビット8）をセットします。さらに，PCIコントローラは，マスクされていなければ，割り込みを外部PCIホスト・デバイスとVR4120Aに発行します。コンフィギュレーション・スペースにあるコマンド・レジスタの“パリティ・エラー応答”ビット（ビット6）がセットされていると，PCIコントローラは，PER_B信号をアサートし，コンフィギュレーション・スペースのステータス・レジスタにある“マスタ・データ・パリティ・エラー”ビット（ビット8）をセットします。ただし，PCIコントローラは現在の転送を終了せず続行します。

(2) PCIマスタとしてマスタ・アボートを検出した場合

PCIコントローラがマスタとしてマスタ・アボートを検出すると，コンフィギュレーション・スペースのステータス・レジスタにあるマスタ・アボート受信ビット（ビット13）とP_PGSRレジスタのRMABTビット（ビット6）をセットします。さらに，PCIコントローラは，そのアクセスがリード・トランザクションであったときはP_IGSRレジスタのRDMATビット（ビット1）をセットし，ライト・トランザクションであったときはP_IGSRレジスタのWRMATビット（ビット0）をセットします。次に，PCIコントローラは，マスクされていなければ，割り込みを外部PCIホスト・デバイスとVR4120Aに発行します。PCIコントローラは，現在のアクセスを停止し，続いてPCIコントローラが新しいアクセスを受け付けることができる状態に戻ります。

(3) PCIマスタとしてターゲット・アボートを検出した場合

PCIコントローラがマスタとしてターゲット・アボートを検出すると，コンフィギュレーション・スペースのステータス・レジスタにあるターゲット・アボート受信ビット（ビット12）とP_PGSRレジスタのRTABTビット（ビット5）をセットします。さらに，PCIコントローラは，そのアクセスがリード・トランザクションであったときはP_IGSRレジスタのRDTATビット（ビット3）をセットし，ライト・トランザクションであったときはP_IGSRレジスタのWRTATビット（ビット2）をセットします。次に，PCIコントローラは，マスクされていなければ，割り込みを外部PCIホスト・デバイスとVR4120Aに発行します。PCIコントローラは，現在のアクセスを停止し，続いてPCIコントローラが新しいアクセスを受け付けることができる状態に戻ります。

(4) PCIマスタとしてターゲット側からのディスコネクトを検出した場合

PCIコントローラがマスタとしてターゲット・ディスコネクトを検出すると，PCIコントローラは，現在のアクセスを終了し，データの残りを転送するために同じターゲットへのアクセスを再び発行します。

(5) PCIマスタとしてターゲット・リトライを検出した場合

PCIバスでPCIコントローラがマスタとしてターゲット・リトライを検出すると，PCIコントローラは，現在のアクセスを終了し，再度，データを転送するために同じターゲットへのアクセスを発行します。

P_RTMRレジスタに“0000_0000H”以外の値をセットした場合，PCIコントローラは，同一アクセスに対するターゲット・リトライ回数がP_RTMRレジスタの値を越えると，アクセスを放棄します。この機能を“リトライ・タイマ”と呼びます。P_RTMRレジスタに“0000_0000H”をセットすると，この機能はディスエーブルになります。

7.2.3.2　内部バス

(1) バス・エラー

内部バス・ブロックに対してPCIコントローラがマスタとしてバス・エラーを検出すると，そのアクセスがリード・トランザクションであったときはP_IGSRレジスタのIRBERビット（ビット6）とP_PGSRレジスタのPRBERビット（ビット1）をセットし，ライト・トランザクションであったときはP_IGSRレジスタのIWBERビット（ビット5）とP_PGSRレジスタのPWBERビット（ビット0）をセットします。次に，PCIコントローラは，マスクされていなければ，割り込みを外部PCIホスト・デバイスとVR4120Aに発行します。PCIコントローラは，現在のアクセスを停止し，続いてPCIコントローラが新しいアクセスを受け付けることができる状態に戻ります。

保守／廃止

7.2.4　デッドロックに対する警告

PCIコントローラは，ノン・ディレイド・リード・トランザクションとノン・ポステッド・ライト・トランザクションを各方向で使用できます。これらのトランザクションでは，PCIコントローラは，もう一方のバスでトランザクションを終了するまでバスを開放しません。したがって，ノン・ディレイド・リード・トランザクションとノン・ポステッド・ライト・トランザクションを各バスで使用するようにPCIコントローラを設定し，トランザクションを両方のバスから同時に発行すると，デッドロックが発生します。ノン・ディレイド・リード・トランザクションまたはノン・ポステッド・ライト・トランザクションのいずれかを少なくとも1つの側で使用することを推奨します。

PCIコントローラは，各バスでリードとライトのトランザクションをそれぞれ1つだけ受け付けることができます。PCIコントローラは，1つのトランザクションを受け付けると，同じ種類のトランザクションに対して“リトライ”を発行します。最高優先順位をもつマスタがアクセスを各バスで連続的にPCIコントローラに発行すると，PCIコントローラは，それらのトランザクションを終了できず，デッドロックが発生します。

保守／廃止

7. 3　PCIパワー・マネジメント・インタフェース

PCIコントローラは，PCIデバイスとしてのPCIパワー・マネジメント・インタフェース（PPMI）rev.1.1に準拠したパワー・マネジメントのメカニズムをサポートしています。PCIコントローラは，チップのパワー状態は制御しませんが，VR4120Aから外部PCIホスト・デバイス，またはPCIホスト・デバイスからVR4120Aへのパワー遷移の信号を発行します。PCIホスト・デバイスとVR4120Aはパワー状態の管理を行います。

また，PCIコントローラは，PCIホストとしてのパワー・マネジメントに関するPCIバス制御機能はサポートしていません。

7. 3. 1　パワー状態

PCIコントローラは，PPMI状態としてD0, D1, D3hot, D3coldをサポートします。パワー・マネジメント・イベント（PME）はD0, D1, D3hotから発生します。

PPMI Rev.1.1では，D0を最大パワー状態として，D1をオプションのパワー・マネジメント状態として，D3hotをクロックがサスペンドするパワー・マネジメント状態として，D3coldをクロックがサスペンドし，かつパワー供給も止まったパワー・マネジメント状態として定義しています。

7. 3. 2　パワー・マネジメント・イベント

PCIコントローラは，D0, D1, D3hotからのパワー・マネジメント・イベント（PME）をサポートします。PMEは，デバイスからのパワー状態の遷移を発行するイベントを示します。

PMEは，PME_B端子をアサートすることによって外部PCIホスト・デバイスに報告します。PCIコントローラでは，PME_B信号をアサートすることができます。P_PPCRレジスタのPMERQビット（ビット30）に“1”を書き込むと，PCIコントローラはPME_B信号をアサートします。

7. 3. 3　電　　源

PCIコントローラはD3coldからのPMEをサポートしないため，PCIコントローラは補助電源（Vaux）を必要としません。

PCIコントローラは，システムにおいて，他の電源がないという前提で設計されているため，パワー供給が停止されるD3coldへの遷移は，チップのすべての部分に対するパワー供給が停止されることを意味します。

μPD98502は，PCIコントローラがPCIバス・ホストとしてPCIバス上に接続されたデバイスをリセットするRST#（PRSTO_B信号）を備えています。しかし，PPMI Rev.1.1でPCIリセットのアサートを必要とすると定義されているD3coldからのウェークアップは，チップ全体をリセットするRSTB_B信号を使うため，チップのすべての部分がリセットされます。

保守／廃止

7.3.4　パワー状態の遷移

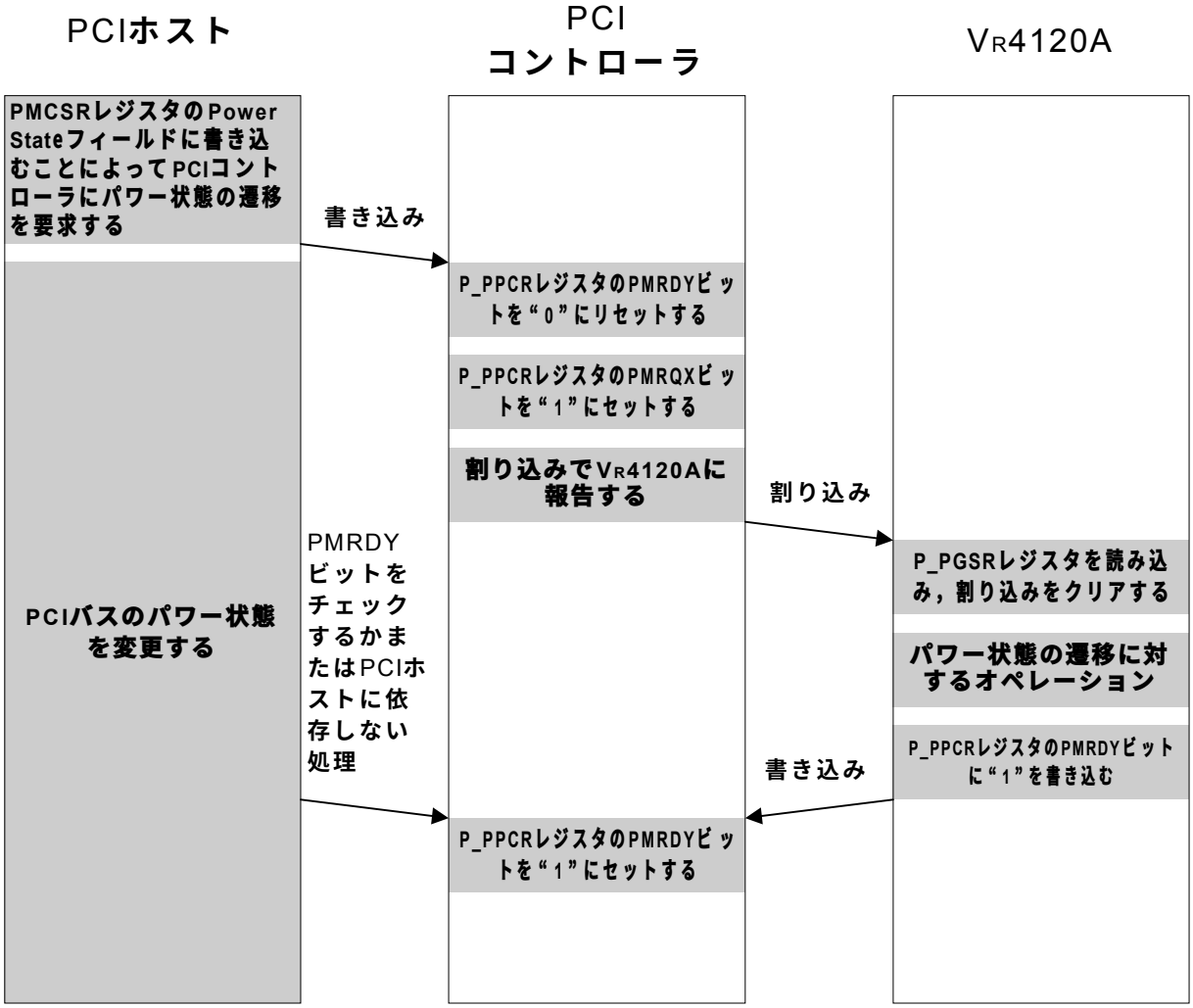
7.3.4.1　PCIホストからの発行による遷移

遷移シーケンスの例を以下に記述します。

1. PCIホストがチップのパワー状態を変更したいときには，状態コードをPCIコンフィギュレーション・スペースのPMCSRレジスタにあるPowerStateフィールドに書き込みます。
2. PCIコントローラは，P_PPCRレジスタのPMRDYビット（ビット31）を“0”にリセットします。
3. PCIコントローラは，P_PPCRレジスタのPMRQXビット（PMRQ0, PMRQ1, PMRQ3）をセットし，マスクされていなければ，割り込みをVR4120Aに発行します。
4. VR4120Aは，P_PGSRレジスタのIPREQビットを読み込むことにより割り込みを認識し，パワー状態の遷移が発行されたことを知ります。
5. 次に，VR4120Aは，P_PPCRレジスタを読み込むことによってどのパワー状態が発行されたかを知ります。
6. VR4120Aは，必要があれば，システムにその状態に応じたオペレーションを実行します。
7. VR4120Aの遷移の準備が整うと，VR4120AはP_PPCRレジスタのPMRDYビットに“1”を書き込みます。
8. 外部PCIホスト・デバイスは，PMRDYビットを読み込むことによってチップの準備が整ったことを知ることができます。PCIホスト・デバイスは，チップの準備が終了するのを待つ必要はなく，パワーとクロックを突然停止してもかまいません。

保守／廃止

図7－10　PCIホストからの発行による遷移シーケンス



保守／廃止

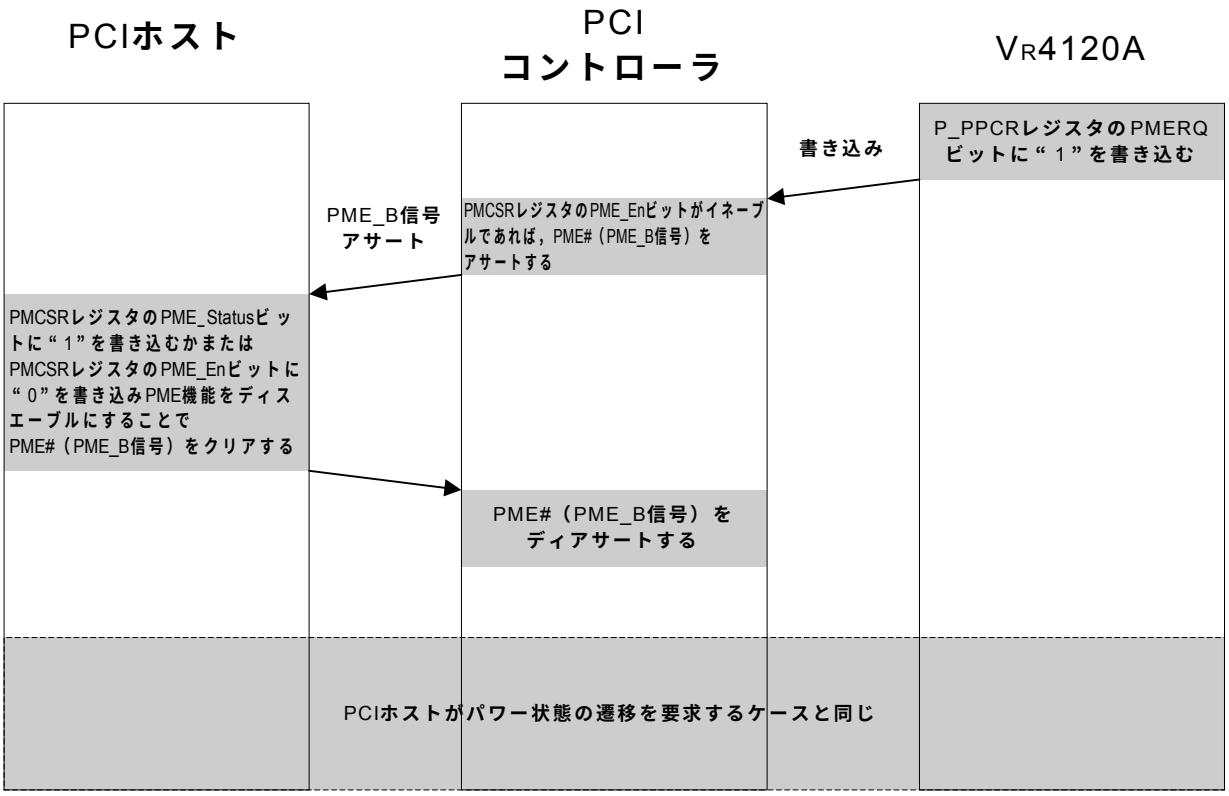
7.3.4.2　パワー・マネジメント・イベントによる遷移

シーケンスを以下に記述します。

1. パワー・マネジメント・イベントが発生すると，VR4120AはP_PPCRレジスタのPMERQビットに“1”を書き込みます。
2. PCIコントローラは，PMCSRレジスタのPME_Enビットがイネーブルであれば，PME_B信号をアサートします。
3. 外部PCIホスト・デバイスは，PME_B信号をディアサートするために，PMCSRレジスタのPME_Statusビットに“1”を書き込むかまたはPMCSRレジスタのPME_Enビットに“0”を書き込みます。
4. PCIコントローラはPME_B信号をディアサートします。

このあと，同じケースの遷移がPCIホストによって発行されます。

図7−11　PME_B信号による遷移シーケンス



保守／廃止

7.4　ホスト・モード機能

このセクションで記述する機能は，PMODEをロウに設定したときに有効になります。

7.4.1　コンフィギュレーション・サイクルの発行

7.4.1.1　コンフィギュレーション・サイクルの発行方法

PCIコントローラは，以下の2つのレジスタをアクセスすることによって，PCIバスに対してコンフィギュレーション・サイクルを発行できます。

PCIコンフィギュレーション・アドレス・レジスタ（P_PCAR）
PCIコンフィギュレーション・データ・レジスタ（P_PCDR）

最初に，コンフィギュレーション・サイクルでアクセスしたいデバイスのアドレス情報を，P_PCARレジスタに設定しなければなりません。そして，次にP_PCDRレジスタにアクセスすると，PCIバスにコンフィギュレーション・サイクルを発行します。

7.4.1.2　PCIコンフィギュレーション・アドレス・レジスタ（P_PCAR）

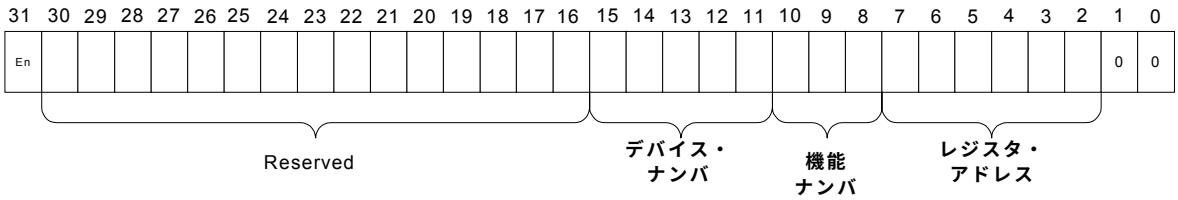
ビット31（コンフィギュレーション・サイクル・イネーブル・ビット）に“1”をセットすると，コンフィギュレーション・サイクルを発行します。このビットを“0”にセットした場合，P_PCDRレジスタにアクセスしてもコンフィギュレーション・サイクルは発行しません。その際，P_PCDRレジスタに対してライト・アクセスを発行するとデータは無視され，リード・アクセスを発行するとすべて“0”のデータが返されます。

PCI仕様では2タイプのコンフィギュレーション・サイクルがあります。PCIブリッジを除くPCIデバイスに対するタイプ0とPCIブリッジに対するタイプ1です。PCIコントローラは，両方のタイプのコンフィギュレーション・サイクルを発行できます。

タイプ0およびタイプ1のコンフィギュレーション・サイクルを発行する際，P_PCARレジスタに設定する内容は図7－12，図7－13のようになります。

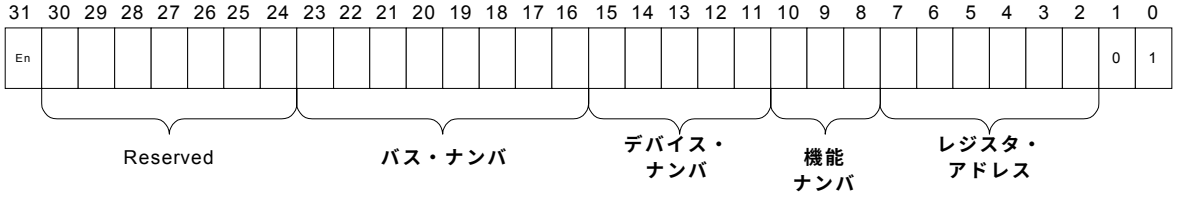
PCIコントローラは，AD［31：0］にこの情報をサイクルのアドレス・フェーズで送信します。

図7－12　タイプ0のコンフィギュレーション・サイクルに対するP_PCARレジスタの内容



保守／廃止

図7－13　タイプ1のコンフィギュレーション・サイクルに対するP_PCARレジスタの内容



PCIコントローラは，個別のIDSEL出力信号を持たないため，タイプ0のトランザクションのアドレス・フェーズでAD［31：16］の各1ビットを“1”にセットし，ほかのすべてのビットを“0”にセットして，“1”にセットされたビットに対応したAD信号をIDSEL信号の代わりに使用します。詳細は次のセクションに記述しています。

7. 4. 1. 3　PCIコンフィギュレーション・データ・レジスタ（P_PCDR）

PCARレジスタのビット31を“1”にセットした場合，PCDRレジスタにアクセスするとコンフィギュレーション・サイクルを発行します。

P_PCDRレジスタに対するリード・アクセスを行うと，PCIバスでコンフィギュレーション・リード・サイクルを発生し，リードしたデータがこのレジスタに反映されます。P_PCDRレジスタに対するライト・アクセスを行うと，PCIバスでコンフィギュレーション・ライト・サイクルを発生し，このレジスタに書き込んだ値がコンフィギュレーション・データとして出力されます。

保守／廃止

7. 4. 1. 4　IDSEL信号

前に述べたように，タイプ0のトランザクションのアドレス・フェーズで，PCIコントローラは，P_PCARレジスタのデバイス・ナンバ・フィールドをデコードして表7－1に示すようにAD［31：16］の中の1ビットを“1”にセットし，ほかのすべてのビットを“0”にセットします。

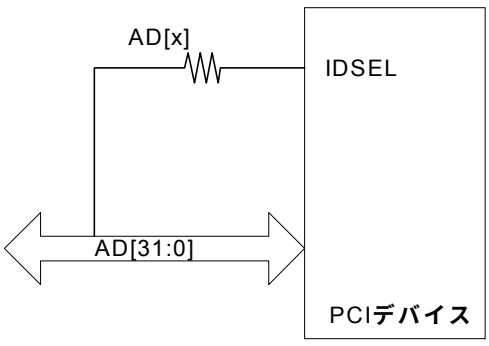
表7－1　デバイス・ナンバ・デコード表

デバイス・ナンバ	AD［31：16］
00000	0000　0000　0000　0001
00001	0000　0000　0000　0010
00010	0000　0000　0000　0100
00011	0000　0000　0000　1000
00100	0000　0000　0001　0000
00101	0000　0000　0010　0000
00110	0000　0000　0100　0000
00111	0000　0000　1000　0000
01000	0000　0001　0000　0000
01001	0000　0010　0000　0000
01010	0000　0100　0000　0000
01011	0000　1000　0000　0000
01100	0001　0000　0000　0000
01101	0010　0000　0000　0000
01110	0100　0000　0000　0000
01111	1000　0000　0000　0000
1xxxx	0000　0000　0000　0000

このAD［31：16］の信号線をIDSEL信号として代替使用できます。しかし，各PCIデバイスのIDSELポートにAD［31：16］の信号線をボード上に直接接続すると，PCIの電気的特性に違反する可能性があります。したがって，これらの信号線を各PCIデバイスのIDSELポートに接続する際は，接続点に抵抗を挿入することを推奨します。

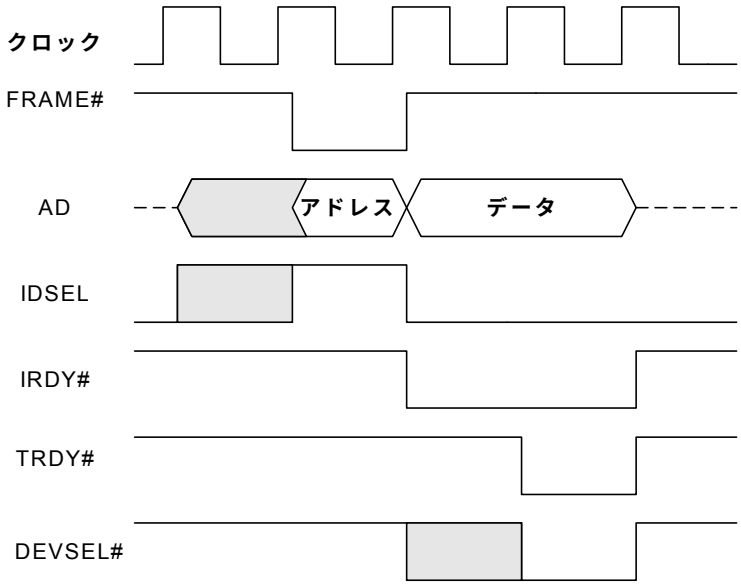
この場合，PCIデバイスへのIDSEL入力が本来有効になるタイミングの前より遅れが生じる可能性があります。PCIコントローラは，この遅れの影響を考慮に入れて，2クロック・アドレス連続ステッピングを常に使用します。

図7－14　AD［31：16］信号線とIDSELポートの接続例



保守／廃止

図7－15　IDSELに対するアドレス・ステッピング



7. 4. 2　PCIバス・アービタ

PCIコントローラには，4つの外部PCIマスタ・デバイスのバス・アービトレーションをサポートするアービタ機能を持っています。このアービタは，ホスト・モード（PMODE：ハイ）でかつPARBEN端子をハイに設定したときにのみイネーブルになります。この内部アービタをディスエーブルにすると，PCIコントローラは，NICモードとホスト・モードの両方でPCIバスを取得する際，外部アービタに対するバス・リクエスト信号としてPRQO_B信号をアサートします。

PCIバスにおいて，PCIマスタ・デバイスが3つ以下でこのアービタを使用しているときは，使用していないリクエスト信号入力端子はプルアップしなければなりません。

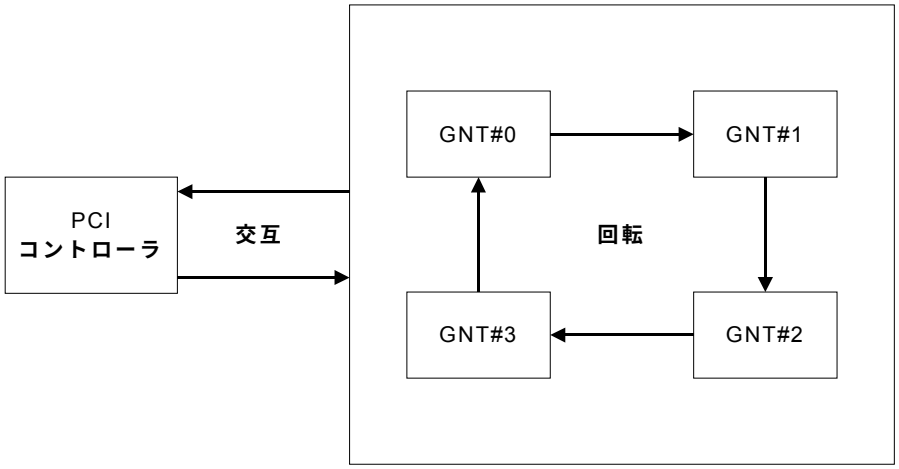
この内部アービタには，アービトレーション・アルゴリズムとして2つのモードがあります。これらのモードは，P_HMCRレジスタのPARBMビットで選択できます。

7. 4. 2. 1　オルタネート・モード

このモードでは，PCIバス上，μPD98502内のPCIコントローラを除く外部PCIマスタ・デバイスは， 1つのグループとして形成されます。優先順位は，トランザクションごとにμPD98502内のPCIコントローラとこのグループとで交互に切り替わります。外部PCIマスタ・デバイスのグループ内では，優先順位はそれらの中で順番に回ります。

オルタネート・モードではすべてのPRQIn_B（nは0, 1, 2, 3）入力信号がハイになったときは，PCIバスの取得を要求するデバイスがないことを意味します。そのときμPD98502内のPCIコントローラがADラインとPCBEn_B（nは0, 1, 2, 3）ラインをアービトレーション・パーキングとしてドライブするように，PCIバスの使用権をPCIコントローラに与えます。

図7－16　オルタネート・モードでのアービトレーション

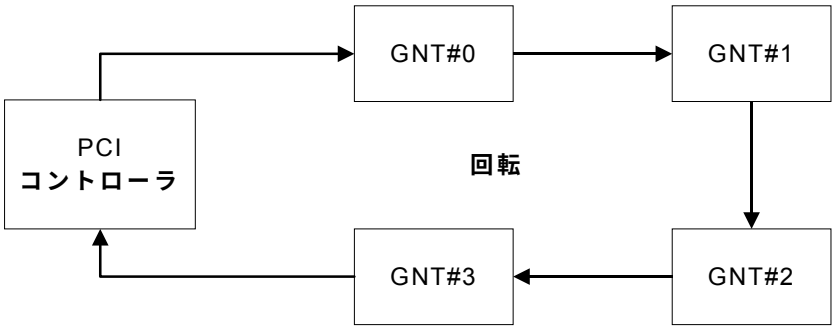


7. 4. 2. 2　ローテート・モード

このモードでは，優先順位はμPD98502内のPCIコントローラを含むすべてのPCIマスタ・デバイス間で順番に回ります。

ローテート・モードではすべてのPRQIn_B（nは0, 1, 2, 3）入力信号がハイになったときは，PCIバスの取得を要求するデバイスがないことを意味します。そのときμPD98502内のPCIコントローラは，PCIバスをアービトレーション・パーキングとして最後に取得したデバイスに対してPCIバスの使用権を与えます。

図7－17　ローテート・モードでのアービトレーション



7. 4. 3　リセット出力

ホスト・モードでは，P_HMCRレジスタのPRSTOビットを“1”にセットすると，PCIコントローラはPCIバスのリセット信号（PRSTO_B）をアサートします。信号をディアサートするためには，VR4120AはPRSTOビットを“0”にセットしなければなりません。

保守／廃止

7.4.4　割り込み入力

PCIコントローラには，PCIターゲット・デバイスから出力割り込み信号（PHINT_B）とSERR#（PSERI_B）を受け取るために，割り込み入力ポート（PHINT_B端子）とSERR#入力ポート（PSERI_B端子）があります。

PCI割り込みとしてPHINT_B端子がアサートされると，P_IGSRレジスタのPINTRビットがセットされ，PCIコントローラは，マスクされていなければ，割り込みをVR4120Aに発行します。PCIコントローラは，割り込みをレベル・トリガで認識します。

SERR#（PSERI_B入力）がアサートされると，P_IGSRレジスタのPSERIビットがセットされ，PCIコントローラは，マスクされていなければ，割り込みをVR4120Aに発行します。PCIコントローラは，SERR#入力をエッジ・トリガで認識します。

PCIコントローラは，割り込み入力とSERR#入力に対してそれぞれ1つのポートしか持たないため，複数のPCIデバイスからの割り込みやSERR#は，たとえばOR回路などで，ボード上で1つの組み合わせにしなければなりません。

保守／廃止

7.5　レジスタ

7.5.1　レジスタ・マップ

アドレス	レジスタ名	R/W		アクセス	説　　明
		内部バス	PCI		
1000_4000H	P_PLBA	R/W	R/W	W/H/B	PCI下位ベース・アドレス・レジスタ
1000_4004H	N/A	-	-	-	将来の使用のため予約
1000_4008H	P_IBBA	R/W	R/W	W/H/B	内部バス・ベース・アドレス・レジスタ
1000_400CH	N/A	-	-	-	将来の使用のため予約
1000_4010H	P_VERR	R	R	W/H/B	バージョン・レジスタ
1000_4014H	P_PCAR	R/W	R	W/H/B	PCIコンフィギュレーション・アドレス・レジスタ
1000_4018H	P_PCDR	R/W	R	W/H/B	PCIコンフィギュレーション・データ・レジスタ
1000_401CH	P_IGSR	RC	R/W	W	内部バス・ジェネラル・ステータス・レジスタ
1000_4020H	P_IIMR	R/W	R/W	W/H/B	内部バス割り込みマスク・レジスタ
1000_4024H	P_PGSR	R/W	RC	W	PCIジェネラル・ステータス・レジスタ
1000_4028H	P_PIMR	R/W	R/W	W/H/B	PCI割り込みマスク・レジスタ
1000_402CH	N/A	-	-	-	将来の使用のため予約
1000_4030H	P_HMCR	R/W	R/W	W/H/B	ホスト・モード制御レジスタ
1000_4034H： 1000_403CH	N/A	-	-	-	将来の使用のため予約
1000_4040H	P_PWCD	R/W	R	W/H/B	消費電力データ・レジスタ
1000_4044H	P_PWDD	R/W	R	W/H/B	損失電力データ・レジスタ
1000_4048H： 1000_404CH	N/A	-	-	-	将来の使用のため予約
1000_4050H	P_BCNT	R/W	R/W	W/H/B	ブリッジ制御レジスタ
1000_4054H	P_PPCR	R/W	R	W/H/B	電力制御レジスタ
1000_4058H	P_SWRR	-	W	W	ソフトウエア・リセット・レジスタ
1000_405CH	P_RTMR	R/W	R/W	W/H/B	リトライ・タイマ・レジスタ
1000_4060H： 1000_40FCH	N/A	-	-	-	将来の使用のため予約
1000_4100H： 1000_41FCH	P_CONFIG	7.5.18 参照	7.5.18 参照	W/H/B	PCIコンフィギュレーション・スペース

備考1.　“R/W”フィールド内で，
“W”は，“書き込み可能”を意味します。
“R”は，“読み取り可能”を意味します。
“RC”は，“読み取りクリア”を意味します。
“-”は，“アクセス不可能”を意味します。

2. すべての内部レジスタは，32ビット・ワード配列のレジスタです。RCタイプの場合，バイト・アクセスするとワード全体がクリアされるので，ワードで必ずアクセスしてください。
3. 内部レジスタに対するバースト・アクセスは，禁止されています。バースト・アクセスが発行されると，NSRのIRERRビットがセットされ，NMIがCPUに対してアサートされます。
4. 予約領域に対するリード・アクセスにより，NSRレジスタのCBERRビットがセットされ，SysCMD［0］にデータ・エラー・ビットが設定されたダミー読み取り応答データが返信されます（一部の非公開レジスタに対しては，返信されません）。

備考5. **予約領域に対するライト・アクセスにより，NSRレジスタのCBERRビットがセットされ，書き込みデータが失われます（一部の非公開レジスタに対しては，セットされません）。**

6. **“アクセス”フィールド内で，**
   “W”は，ワード・アクセスが有効であることを意味します。
   “H”は，ハーフ・ワード・アクセスが有効であることを意味します。
   “B”は，バイト・アクセスが有効であることを意味します。
7. **リード・オンリー・レジスタに対するライト・アクセスはエラーになりませんが，書き込みデータは失われます。**
8. **VR4120Aからはすべての内部レジスタにアクセスできますが，その他のIBUSマスタ・デバイス（イーサネット，USBなど）からはそれらのレジスタにアクセスできません。**

7.5.2　P_PLBA（PCI下位ベース・アドレス・レジスタ）

このレジスタは，PCIコントローラが32ビットのPCIアドレスを発行する際のPCIベース・アドレスを設定します。内部バス（IBUS）側からPCI側へアクセスされると，PCIコントローラは，内部バスのアドレスの上位10ビットをこのレジスタの上位10ビットで置き換え，PCIバスのアドレスとして発行します。

PCIコンフィギュレーション・スペース内にも同一機能のレジスタがありますが，いずれも個別に設定する必要があります。

1.9　**メモリ・マップの注**1を参照してください。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31：0	PLBA	R/W（上位10ビット） R（下位22ビット）	R/W（上位10ビット） R（下位22ビット）	0	PCI下位ベース・アドレス 上位10ビットはリード／ライト可能です。 下位22ビットはリード・オンリーです。

7.5.3　P_IBBA（内部バス・ベース・アドレス・レジスタ）

このレジスタは，内部バスに対するベース・アドレスを設定します。PCI側から内部バス（IBUS）側へアクセスされると，PCIコントローラは，PCIのアドレスの上位10ビットをこのレジスタの上位10ビットで置き換え，内部バス（IBUS）へのアドレスとして発行します。

PCIコンフィギュレーション・スペース内にも同一機能のレジスタがありますが，いずれも個別に設定する必要があります。1.9　**メモリ・マップの注**1を参照してください。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31：0	IBBA	R/W（上位10ビット） R（下位22ビット）	R/W（上位10ビット） R（下位22ビット）	0	内部バス・ベース・アドレス 上位10ビットはリード／ライト可能です。 下位22ビットはリード・オンリーです。

保守／廃止

7.5.4 P_VERR（バージョン・レジスタ）

このレジスタは，PCIコントローラのバージョン・ナンバを示します。上位16ビットはメージャ・バージョンを示し，下位16ビットはマイナ・バージョンを示します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31：0	VERR	R	R	0001_0000H	PCIコントローラのバージョンを示します。 メージャ・バージョン：0001H マイナ・バージョン：0000H

7.5.5 P_PCAR（PCIコンフィギュレーション・アドレス・レジスタ）

このレジスタは，コンフィギュレーション・サイクルの情報を設定するために使用します。コンフィギュレーション・サイクルの発行方法については，7.4.1 **コンフィギュレーション・サイクルの発行**を参照してください。

PCIコントローラは，ホスト・モードでのみコンフィギュレーション・サイクルを発行できます。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31	COCEN	R/W	R	0	コンフィギュレーション・サイクル発行イネーブル このビットを“1”にセットし，続いてPCDRに対してリード／ライトを行うと，コンフィギュレーション・サイクルを発行します。 このビットを“0”にセットすると，PCDRに対してリード／ライトを行ってもコンフィギュレーション・サイクルを発行しません。
30：24	Reserved	-	-	0	“0”固定
23：16	BUSNM	R/W	R	0	バス・ナンバ コンフィギュレーション・サイクルにおいてアドレス・フェーズで出力するバス・ナンバ・フィールドを設定してください。
15：11	DEVNM	R/W	R	0	デバイス・ナンバ コンフィギュレーション・サイクルにおいてアドレス・フェーズで出力するデバイス・ナンバ・フィールドを設定してください。
10：8	FNCNM	R/W	R	0	機能ナンバ コンフィギュレーション・サイクルにおいてアドレス・フェーズで出力する機能ナンバ・フィールドを設定してください。
7：2	REGAC	R/W	R	0	各デバイスに対するオフセット・アドレス コンフィギュレーション・サイクルにおいてアドレス・フェーズで出力するレジスタ・ナンバ・フィールドを設定してください。
1：0	CYCTP	R/W	R	00	コンフィギュレーション・サイクル・タイプの選択 00：コンフィギュレーション・タイプ0 01：コンフィギュレーション・タイプ1 1x：設定禁止

保守／廃止

7. 5. 6　P_PCDR（PCIコンフィギュレーション・データ・レジスタ）

このレジスタは，コンフィギュレーション・サイクルにおけるコンフィギュレーション・データのリード／ライトを行うために使用します。コンフィギュレーション・サイクルの発行方法については，7. 4. 1　**コンフィギュレーション・サイクルの発行**を参照してください。

PCIコントローラは，ホスト・モードでのみコンフィギュレーション・サイクルを発行できます。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31：0	PCDR	R/W	R	0	コンフィギュレーション・サイクルにおけるデータ

保守／廃止

7. 5. 7　P_IGSR（内部バス・ジェネラル・ステータス・レジスタ）

このレジスタは，PCIコントローラが発行した割り込みの発生要因をVR4120Aに示します。割り込みイベントが発生すると，PCIコントローラは，イベントに対応するこのレジスタのビットをセットします。P_IIMRの対応するビットがセットされていた場合，PCIコントローラは，VR4120Aに対して内部割り込み信号（int3：イーサネット・コントローラ#2と共用）をアサートします。VR4120Aからこのレジスタを読み込むと，レジスタのすべてのビットがクリアされます。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31：16	IUINT	RC	R/W	0	システムでユーザが定義可能な割り込み。 PCIバス側からこのフィールドのビットに“1”を書き込むと，VR4120Aに対して割り込みをアサートします。
15：12	Reserved	-	-	0	“0”固定。
11	PINTR	RC	R	0	ホスト・モードでのみ使用。 PCIデバイスから割り込み（PHINT_B端子）がアサートされたことを示します。 “1”は，PCI割り込みが入力されたことを示します。
10	PSERI	RC	R	0	ホスト・モードでのみ使用。 外部PCIデバイスからSERR#（PSERI_B端子）がアサートされたことを示します。 “1”は，SERR#がアサートされたことを示します。
9	PPERR	RC	R	0	PCIパリティ・エラー検出。 “1”は，PCIコントローラがPCIバスでパリティ・エラーを検出したことを示します。
8	PPREQ	RC	R	0	PPMIパワー状態遷移を検出。 “1”は，PCIホストがPCIコントローラのパワー状態の遷移を検出したことを示します。 このビットをセットすると，VR4120Aは，どの状態にPCIコントローラが移行するかを知るためにP_PPCRレジスタをチェックしなければなりません。
7	SRREQ	RC	R	0	ソフトウエア・リセット。 “1”は，PCIホストがP_SWRRレジスタに書き込み，PCIホストがソフトウエア・リセットを要求したことを示します。 実際のリセット動作はVR4120Aが行い，PCIコントローラ自身は，PCIホストからのリセット要求を仲介するための機能のみを提供しています。 VR4120Aは，ソフトウエア・リセットの終了後，P_PGSRレジスタのSWRDNビットを“1”にセットしなければなりません。
6	IRBER	RC	R	0	リード・トランザクション中に内部バス・エラー検出。 “1”は，PCIコントローラが，マスタとしてリード・トランザクション中に内部バスでバス・エラーを検出したことを示します。
5	IWBER	RC	R	0	ライト・トランザクション中に内部バス・エラー検出。 “1”は，PCIコントローラが，マスタとしてライト・トランザクション中に内部バスでバス・エラーを検出したことを示します。

保守／廃止

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
4	IFDSC	RC	R	0	内部バスFIFOデータ廃棄。 “1”は，2^{15}クロック以内に同じトランザクション・アクセス発行が繰り返されなかったので，PCIコントローラがFIFO内のディレイド・リード・トランザクションのデータを廃棄したことを示します。
3	RDTAT	RC	R	0	リード・トランザクション中にPCIターゲット・アボート検出。 “1”は，PCIコントローラが，マスタとしてリード・トランザクション中にPCIバスでターゲット・アボートを検出したことを示します。
2	WRTAT	RC	R	0	ライト・トランザクション中にPCIターゲット・アボート検出。 “1”は，PCIコントローラが，マスタとしてライト・トランザクション中にPCIバスでターゲット・アボートを検出したことを示します。
1	RDMAT	RC	R	0	リード・トランザクション中にPCIマスタ・アボート検出。 “1”は，PCIコントローラが，マスタとしてリード・トランザクション中にPCIバスでマスタ・アボートを検出したことを示します。
0	WRMAT	RC	R	0	ライト・トランザクション中にPCIマスタ・アボート検出。 “1”は，PCIコントローラが，マスタとしてライト・トランザクション中にPCIバスでマスタ・アボートを検出したことを示します。

保守／廃止

7.5.8　P_IIMR（内部バス割り込みマスク・レジスタ）

このレジスタは，P_IGSRの要因それぞれに対応する割り込みをマスクします。P_IGSRの対応するビットと同じ位置にあるマスク・ビットは，イベントが引き起こす割り込みを制御します。このレジスタの該当するビットを“0”にセットすると，P_IGSRの対応するビットによる割り込みをマスクします。このビットを“1”にセットすると，対応するイベントはマスクされません。マスク・ビットを“1”にセットし，IGSRのビットがセットされると，PCIコントローラはVR4120Aに対して割り込み信号をアサートします。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31：16	IUINT	R/W	R/W	0	システムでユーザが定義する割り込みに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
15：12	Reserved	-	-	0	“0H”固定。
11	PINTR	R/W	R/W	0	PINTRに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
10	PSERI	R/W	R/W	0	PSERIに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
9	PPERR	R/W	R/W	0	PPERRに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
8	PPREQ	R/W	R/W	0	PPREQに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
7	SRREQ	R/W	R/W	0	SRREQに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
6	IRBER	R/W	R/W	0	IRBERに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
5	IWBER	R/W	R/W	0	IWBERに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
4	IFDSC	R/W	R/W	0	IFDSCに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
3	RDTAT	R/W	R/W	0	RDTATに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
2	WRTAT	R/W	R/W	0	WRTATに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。

保守／廃止

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
1	RDMAT	R/W	R/W	0	RDMATに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
0	WRMAT	R/W	R/W	0	WRMATに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。

保守／廃止

7.5.9　P_PGSR（PCIジェネラル・ステータス・レジスタ）

このレジスタは，PCIバス側（PCIホスト）への割り込み状態を示します。割り込みを生成するイベントが発生すると，PCIコントローラはこのレジスタの対応するビットを“1”にセットします。P_PIMRレジスタによって割り込みがマスクされていなければ，PCIコントローラはPINT_B信号によってPCIホストへ割り込みを通知します。

このレジスタの上位16ビットは，ユーザが定義可能なフィールドとなっています。IBUS側からこのフィールドのいずれかのビットに“1”を書き込むと，マスクされていなければ，PCIホストへの割り込み信号（PINT_B信号）がアサートされます。したがって，PCIコントローラを用いた独自の割り込み発生イベントをこれらのビットに割り当てることができます。

PCIバス側からこのレジスタをリードすることによって，このレジスタ中の全ビットが“0”にクリアされます。バイト・イネーブルを用いて選択的にビットをクリアすることはできません。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31：16	PUINT	R/W	R	0	ユーザ定義割り込み IBUS側からいずれかのビットに“1”を書き込むと，外部PCIホストへの割り込み（PINT_B信号）がアサートされます。ユーザ独自の割り込み要求に割り当てることができます。
15：11	Reserved	-	-	0	“0”固定。
10	IPREQ	R	R	0	PPMIステート遷移要求 “1”は，外部PCIホスト・デバイスが電力ステートの遷移を要求していることを示します。
9	SWRDN	R/W	R	0	ソフトウエア・リセット終了 “1”は，ソフトウエア・リセットが終了したことを示します。VR4120Aがソフトウエア・リセット処理を終了した直後に“1”にセットしてください。
8	DPERR	R	R	0	PCIパリティ・エラー検出 “1”は，PCIコントローラがPCIバス上のパリティ・エラーを検出したことを示します。
7	SSERR	R	R	0	システム・エラー信号発信 “1”は，PCIコントローラがSERR#（PSERO_B信号）をアサートしたことを示します。
6	RMABT	R	R	0	マスタ・アポート検出 “1”は，PCIコントローラがマスタとしてマスタ・アボートを検出したことを示します。
5	RTABT	R	R	0	ターゲット・アボート検出 “1”は，PCIコントローラがマスタとしてターゲット・アボートを検出したことを示します。
4	STABT	R	R	0	ターゲット・アボート実行 “1”は，PCIコントローラがターゲットとしてターゲット・アボートを実行したことを示します。
3	PFDSC	R	R	0	PCI FIFO廃棄 “1”は，ディレイド・リード・トランザクション時に，PCIバス上で同じ内容のアクセスが2^{15}クロック以内に来なかったため，PCIコントローラがFIFO内部のデータを廃棄したことを示します。

保守／廃止

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
2	RTYTE	R	R	0	リトライ・タイマ完了 “1”は，リトライ・タイマで設定された回数分リトライを繰り返しても転送が完了しなかったため，PCIコントローラがそのリトライを放棄したことを示します。
1	PRBER	R	R	0	リード時のIBUSバス・エラー発生 “1”は，リード・トランザクション時にPCIコントローラがマスタとしてIBUSバス・エラーを検出したことを示します。
0	PWBER	R	R	0	ライト時のIBUSバス・エラー発生 “1”は，ライト・トランザクション時にPCIコントローラがマスタとしてIBUSバス・エラーを検出したことを示します。

保守／廃止

7. 5. 10　P_PIMR（PCI割り込みマスク・レジスタ）

このレジスタは，P_PGSRの各要因それぞれに対応した割り込みをマスクします。P_PGSRの対応するビットと同じ位置にあるマスク・ビットは，その割り込みイベントをマスクできます。このレジスタの該当するビットを“0”にセットすると，P_PGSRの対応するイベントによる割り込み発行がマスクされます。このビットを“1”にセットすると，対応するイベントはマスクされません。マスク・ビットを“1”にセットし，P_PGSRのビットがセットされると，PCIコントローラはPCIホストに対する割り込み信号（PINT_B端子）をアサートします。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31：16	PUINT	R/W	R/W	0	システムでユーザが定義する割り込みに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
15：11	Reserved	-	-	0	“0”固定。
10	IPREQ	R/W	R/W	0	IPREQに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
9	SWRDN	R/W	R/W	0	SWRDNに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
8	DPERR	R/W	R/W	0	DPERRに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
7	SSERR	R/W	R/W	0	SSERRに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
6	RMABT	R/W	R/W	0	RMABTに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
5	RTABT	R/W	R/W	0	RTABTに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
4	STABT	R/W	R/W	0	STABTに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
3	PFDSC	R/W	R/W	0	PFDSCに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
2	RTYTE	R/W	R/W	0	RTYTEに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。
1	PRBER	R/W	R/W	0	PRBERに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。

保守／廃止

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
0	PWBER	R/W	R/W	0	PWBERに対するマスク・ビット。 “0”は，マスクを意味します。 “1”は，マスクされないことを意味します。

7. 5. 11　P_HMCR（ホスト・モード制御レジスタ）

このレジスタは，PCIホスト機能を制御するために使用します。したがって，PCIホスト・モードでのみ有効となります。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31	PRSTO	R/W	R	0	リセット出力。 PCIホスト・デバイスとしてのPCIリセット出力。 PCIコントローラは，このビットが“1”の間にPRSTO_B信号をアサートします。ディアサートする場合は“0”をセットする必要があります。
30：1	Reserved	-	-	0	“0”固定。
0	PARBM	R/W	R/W	0	PCIアービタ・モード。 このビットはアービタ・モードを選択します。 0：オルタネート・モード 1：ローテート・モード

7. 5. 12　P_PWCD（消費電力データ・レジスタ）

このレジスタは，各パワー状態の消費電力データを示すために使用します。

VR4120Aは，このレジスタの値を初期設定時に設定します。データの単位は，0.01ワットです。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31：24	D3CSP	R/W	R	00H	D3消費電力データ。
23：16	Reserved	-	-	00H	“00H”固定。
15：8	D1CSP	R/W	R	00H	D1消費電力データ。
7：0	D0CSP	R/W	R	00H	D0消費電力データ。

保守／廃止

7. 5. 13　P_PWDD（損失電力データ・レジスタ）

このレジスタは，各パワー状態の損失電力データを示すために使用します。

VR4120Aは，このレジスタの値を初期設定時に設定します。データの単位は，0.01ワットです。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31：24	D3DSP	R/W	R	00H	D3損失電力データ。
23：16	Reserved	-	-	00H	“00H”固定。
15：8	D1DSP	R/W	R	00H	D1損失電力データ。
7：0	D0DSP	R/W	R	00H	D0損失電力データ。

7. 5. 14　P_BCNT（ブリッジ制御レジスタ）

このレジスタは，PCIと内部バス（IBUS）間のブリッジ機能を制御するために使用します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31	INITD	R/W	R	0	初期設定完了通知。 VR4120Aがチップの初期設定を完了したあと，このビットを“1”にセットしてください。 PCIコントローラは，このビットが“0”にセットされている間，PCIバスからのアクセスに対して常に“リトライ”を返します。
30：6	Reserved	-	-	0	“0”固定。
5	ICMDS	R/W	R/W	0	内部バス・コマンド選択。 このビットを“1”にセットすると，PCIコントローラは内部バスに対してメモリ・コマンドを使用します。 このビットを“0”にセットすると，PCIコントローラは内部バスに対してI/Oコマンドを使用します。
4	DACEN	R/W	R/W	0	デュアル・アドレス・サイクル・イネーブル。 将来の使用のため予約。μPD98502ではデュアル・アドレス・サイクルはサポートしていません。したがって，このビットは“0”にセットしてください。
3	PDRTD	R/W	R/W	0	PCIのディレイド・リード・トランザクションのディスエーブル。 “1”は，PCIコントローラがディレイド・リード・トランザクションをPCIバスで発行することをディスエーブルにします。 その際，PCIコントローラは，リード・トランザクションをノン・ディレイド・リード・トランザクションとして実行します。

保守／廃止

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
2	PPWRD	R/W	R/W	0	PCIのポステッド・ライト・トランザクションのディスエーブル。 “1”は，PCIコントローラがポステッド・ライト・トランザクションをPCIバスで発行することをディスエーブルにします。 その際，PCIコントローラは，ライト・トランザクションをノン・ポステッド・ライト・トランザクションとして実行します。
1	IDRTD	R/W	R/W	0	内部バスのディレイド・リード・トランザクションのディスエーブル。 “1”は，PCIコントローラがディレイド・リード・トランザクションを内部バスに対して発行することをディスエーブルにします。 その際，PCIコントローラは，リード・トランザクションをノン・ディレイド・リード・トランザクションとして実行します。
0	IPWRD	R/W	R/W	0	内部バスのポステッド・ライト・トランザクションのディスエーブル。 “1”は，PCIコントローラがポステッド・ライト・トランザクションを内部バスに対して発行することをディスエーブルにします。 その際，PCIコントローラは，ライト・トランザクションをノン・ポステッド・ライト・トランザクションとして実行します。

保守／廃止

7. 5. 15　P_PPCR（電力制御レジスタ）

このレジスタは，PPMIのパワー状態を制御するために使用します。

詳細については，7. 6　ソフトウエア処理および7. 3　PCIパワー・マネジメント・インタフェースを参照してください。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31	PMRDY	R/W	R	0	パワー・マネジメント・レディ。 “1”は，パワー状態の遷移を行うことができるレディ状態であることを示します。 PCIホストがコンフィギュレーション・スペースのPMCSRレジスタのPowerStateフィールドにデータを書き込むと，このビットは“0”にセットされます。VR4120Aは，パワー状態の遷移が行われたあとに，このビットに再び“1”をセットし，レディ状態に戻さなければなりません。
30	PMERQ	W	-	0	PME#（PME_B）信号をアサート。 このビットは，内部バス側からのみ書き込み可能です。 PCIコントローラは，“1”がこのビットに書き込まれたときに，PME#（PME_B）信号をアサートします。 PCIホストがPME#（PME_B）信号をディアサートしたいときは，PMCSRレジスタのPME_Statusビットに“1”を書き込むか，PMCSRレジスタのPME_Enビットをディスエーブルにしなければなりません。詳細については，7. 3と7. 6を参照してください。
29：4	Reserved	-	-	0	“0”固定。
3	PMRQ0	R/W	R	0	D0パワー状態への遷移を発行。 “1”は，PCIホストからD0状態への遷移が発行されたことを示します。 VR4120Aは，このビットがセットされたことを認識したあと，クリアするためにこのビットに“1”を書き込まなければなりません。
2	PMRQ1	R/W	R	0	D1パワー状態への遷移を発行。 “1”は，PCIホストからD1状態への遷移が発行されたことを示します。 VR4120Aは，このビットがセットされたことを認識したあと，クリアするためにこのビットに“1”を書き込まなければなりません。
1	Reserved	-	-	0	“0”固定。
0	PMRQ3	R/W	R	0	D3パワー状態への遷移を発行。 “1”は，PCIホストからD3状態への遷移が発行されたことを示します。 VR4120Aは，このビットがセットされたことを認識したあと，クリアするためにこのビットに“1”を書き込まなければなりません。

保守／廃止

7. 5. 16　P_SWRR（ソフトウエア・リセット・レジスタ）

このレジスタは，ソフトウエア・リセットに使用し，PCI側からのみ書き込み可能です。PCIホストはこのレジスタにどのような値を書き込んでもかまいません。このレジスタにライト・アクセスを行うと，P_IGSRレジスタのSRREQビットがセットされ，内部割り込みをVR4120Aへアサートします。リセット動作（たとえば，すべてのブロックへのウォーム・リセットのアサート）自体はVR4120Aが実行しなければなりません。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31：0	SWRR	-	W	-	このレジスタはライト・オンリーです。

7. 5. 17　P_RTMR（リトライ・タイマ・レジスタ）

このレジスタは，リトライの繰り返し回数の制限を設定するために使用します。“0000_0000H”を設定すると，リトライ機能はディスエーブルとなります。詳細については，7. 2. 3. 1（5）PCIマスタとしてターゲット・リトライを検出した場合を参照してください。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31：0	RTMR	R/W	R/W	0000_0000H	リトライの繰り返し回数を設定します。“0000_0000H”設定は，この機能をディスエーブルとします。

保守／廃止

7. 5. 18　P_CONFIG（PCIコンフィギュレーション・スペース）

7. 5. 18. 1　PCIコンフィギュレーション・スペース・マップ

アドレス[注]	31 24	23 16	15 8	7 0
1000_4100H	デバイスID		ベンダID	
1000_4104H	ステータス		コマンド	
1000_4108H	クラス・コード		リビジョンID	
1000_410CH	Reserved	ヘッダ・タイプ	レイテンシ・タイマ	キャッシュ・ライン・サイズ
1000_4110H	ウインドウ・メモリ・ベース・アドレス			
1000_4114H	レジスタ・メモリ・ベース・アドレス			
1000_4118H	Reserved			
1000_411CH	Reserved			
1000_4120H	Reserved			
1000_4124H	Reserved			
1000_4128H	Reserved			
1000_412CH	サブシステムID		サブシステム・ベンダID	
1000_4130H	Reserved			
1000_4134H	Reserved			Cap_Ptr
1000_4138H	Reserved			
1000_413CH	Max_Lat	Min_Gnt	インタラプト・ピン	インタラプト・ライン
1000_4140H	PMC		Next_Item_Ptr	Cap_ID
1000_4144H	PMData	Reserved	PMCSR	
1000_4148H：1000_41FCH	Reserved			

注　このアドレスはVR4120Aから見たアドレスであり，PCI側から見たアドレスは，“ レジスタ・メモリ・ベース・アドレス・レジスタ ”によって割り当てられます。

保守／廃止

アドレス	レジスタ名	サイズ（バイト）	内部バス	PCI
1000_4100H	ベンダID	2	R	R
1000_4102H	デバイスID	2	R	R
1000_4104H	コマンド	2	R/W	R/W
1000_4106H	ステータス	2	R/W	R/W
1000_4108H	リビジョンID	1	R	R
1000_4109H	クラス・コード	3	R	R
1000_410CH	キャッシュ・ ライン・サイズ	1	R/W	R/W
1000_410DH	レイテンシ・ タイマ	1	R/W	R/W
1000_410EH	ヘッダ・タイプ	1	R	R
1000_410FH	Reserved	1	-	-
1000_4110H	ウインドウ・メモリ・ ベース・アドレス	4	R/W	R/W
1000_4114H	レジスタ・メモリ・ベース・アドレス	4	R/W	R/W
1000_4118H	Reserved	4	-	-
1000_411CH	Reserved	4	-	-
1000_4120H	Reserved	4	-	-
1000_4124H	Reserved	4	-	-
1000_4128H	Reserved	4	-	-
1000_412CH	サブシステム・ ベンダID	2	R/W	R
1000_412EH	サブシステムID	2	R/W	R
1000_4130H	Reserved	4	-	-
1000_4134H	Cap_Ptr	1	R	R
1000_4135H	Reserved	3	-	-
1000_4138H	Reserved	4	-	-
1000_413CH	インタラプト・ライン	1	R/W	R/W
1000_413DH	インタラプト・ピン	1	R	R
1000_413EH	Min_Gnt	1	R/W	R
1000_413FH	Max_Lat	1	R/W	R
1000_4140H	Cap_ID	1	R	R
1000_4141H	Next_Item_Ptr	1	R	R
1000_4142H	PMC	2	R/W	R
1000_4144H	PMCSR	2	R/W	R/W
1000_4146H	Reserved	1	-	-
1000_4147H	PMData	1	R	R
1000_4148H： 1000_41FFH	Reserved	8	-	-

コンフィギュレーション・スペースは，VR4120Aからもリード／ライトが可能です。VR4120Aから見たコンフィギュレーション・スペースは，内部メモリ・マップのオフセット・アドレスの1000_4100Hから始まります。

保守／廃止

7. 5. 18. 2　ベンダIDレジスタ

このレジスタは，μPD98502の製造会社を示します。NECのIDは“1033H”です。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
15：0	ベンダID	R	R	1033H	NECのベンダIDを意味します。“1033H”固定。

7. 5. 18. 3　デバイスIDレジスタ

このレジスタは，μPD98502のデバイスIDを示します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
15：0	デバイスID	R	R	00A7H	“00A7H”固定。

保守／廃止

7. 5. 18. 4　コマンド・レジスタ

このレジスタは， PCIサイクルの発生と応答に関して，μPD98502がサポートする機能の設定および表示を行います。このレジスタは，ホスト・モードでのみ有効です。VR4120Aが初期設定時にこのレジスタを設定します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
15：10	Reserved	R	R	0	“0”固定。
9	高速 Back-to-Back イネーブル	R	R	0	μPD98502は高速Back-to-Backトランザクションを実行できないため，“0”固定。
8	システム・エラー・イネーブル	R/W	R/W	0	このビットは，SERR#（システム・エラー）のイネーブル・ビットです。“1”は，SERR#（システム・エラー）をイネーブルにします。
7	ウエイト・サイクル・イネーブル	R	R	0	μPD98502はアドレス／データ・ステッピングをPCIデバイスとして使用しないため，“0”固定。
6	パリティ・エラー応答	R/W	R/W	0	このビットは，パリティ・エラーに対するPCIコントローラの応答を設定します。 このビットを“0”にセットすると，PCIコントローラは，パリティ・エラーを検出しても，通常の動作を継続します。このビットを“1”にセットすると，PCIコントローラは，エラー検出時に，パリティ・エラー検出ビット（ステータス・レジスタのビット15）をセットしますが，PERR#をアサートせず通常動作を続けます。
5	VGA パレット・スヌープ・イネーブル	R	R	0	μPD98502はVGA機能を持たないため，“0”固定。
4	メモリ・ライト・アンド・インバリデートのイネーブル	R/W	R/W	0	このビットは，メモリ・ライト・アンド・インバリデート・コマンドの使用をイネーブルとするビットです。このビットを“1”にセットすると，PCIコントローラは，メモリ・ライト・アンド・インバリデート・コマンドを発行できます。このビットを“0”にセットすると，メモリ・ライトのみが使用され，メモリ・ライト・アンド・インバリデート・コマンドは発行されません。
3	スペシャル・サイクルの認識	R	R	0	PCIコントローラはスペシャル・サイクル動作に応答しないため，“0”固定。
2	バス・マスタ・イネーブル	R/W	R/W	0	PCIバスでマスタとして動作するPCIコントローラを制御します。“0”は，PCIコントローラがマスタとなってPCIアクセスを発行することをディスエーブルにします。“1”は，PCIコントローラがバス・マスタとして動作することを許可します。

保守／廃止

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
1	メモリ・アクセス・イネーブル	R/W	R/W	0	メモリ・スペースのアクセスに対するデバイスの応答を制御します。“0”は，外部PCIコントローラの応答をディスエーブルにします。“1”は，外部PCIコントローラがメモリ・スペースのアクセスに応答することを許可します。
0	I/Oアクセス・イネーブル	R	R	0	PCIコントローラはI/Oアクセス・サイクルを発行しないため，“0”固定。

7. 5. 18. 5 ステータス・レジスタ

このレジスタは，PCIバス関連のイベントの状態を示すために使用します。これらのビットは，PCIバスの状態に関連するイベントが発生するとセットされ，“1”を書き込むと“0”にリセットされます。

ホスト・モードでは，このレジスタのビットは，対応するイベントが発生しても，セットされません。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
15	パリティ・エラー検出	R/W	R/W	0	パリティ・エラーを検出すると，このビットを“1”にセットします。
14	システム・エラー発信	R/W	R/W	0	PSERO_Bをアサートすると，このビットを“1”にセットします。
13	マスタ・アボート受信	R/W	R/W	0	マスタ・デバイスとしてマスタ・アボートのトランザクションを終了すると，このビットを“1”にセットします。
12	ターゲット・アボート受信	R/W	R/W	0	ターゲット・アボートを検出し，マスタ・デバイスとしてそのトランザクションを終了すると，このビットを“1”にセットします。
11	ターゲット・アボート発生	R/W	R/W	0	ターゲット・デバイスとしてターゲット・アボートのトランザクションを終了すると，このビットを“1”にセットします。
10：9	DEVSEL_Bタイミング	R	R	01	μPD98502は中速度でDEVSEL#をアサートするため，“01”固定。
8	マスタ・データ・パリティ・エラー	R/W	R/W	0	このビットは，以下の3つの条件を満たすとセットされます。 (1) バス・エージェントが読み込みのときにPERR#をアサートした，または書き込みのときにPERR#がアサートされるのを検出した。 (2) ビットをセットするエージェントが，エラーを発生したオペレーションに対してバス・マスタとして動作していた。 (3) パリティ・エラー応答ビットが“1”にセットされていた。
7	高速Back-to-Back	R	R	0	高速Back-to-Backトランザクションを受け付けないため，“0”固定。
6	Reserved	R	R	0	“0”固定。
5	66 MHzイネーブル	R	R	0	μPD98502は33 MHzのみで使用可能なため，“0”固定。
4	機能リスト	R	R	1	PCIコントローラは新しい機能としてPPMI機能を持っているため，“1”固定。
3：0	Reserved	R	R	0	“0”固定。

保守／廃止

7. 5. 18. 6　リビジョンIDレジスタ

このレジスタは，μPD98502のリビジョンIDを示します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
7：0	リビジョンID	R	R	01H	μPD98502のリビジョン・ナンバを示す“01H”固定。

7. 5. 18. 7　クラス・コード・レジスタ

このレジスタは，μPD98502のクラス・コードを識別するために使用します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
23：0	クラス・コード	R	R	000302H[注]	μPD98502のクラス・コードを示す“000302H”固定。

注　コンフィギュレーション・スペースのクラス・コード・レジスタを読み込むと，“020300H”（ATMコントローラ）がμPD98502のクラス・コードとして本来返されなければなりませんが，間違った値“000302H”（VGA関連）が返されます。申し訳ありませんが，ホスト・ドライバ側のソフトウエアで対処してください。

7. 5. 18. 8　キャッシュ・ライン・サイズ・レジスタ

このレジスタは，システムのキャッシュ・ライン・サイズをワード（32ビット長）単位で指定します。このレジスタの値は，メモリ・リード，メモリ・リード・ライン，メモリ・リード・マルチプル，メモリ・ライト・アンド・インバリデート・コマンドをメモリ・アクセスに使用するかどうかを判断するためにも使用します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
7：0	キャッシュ・ライン・サイズ	R/W	R/W	0	ワード単位のシステムのキャッシュ・ライン・サイズをセットしてください。

7. 5. 18. 9　レイテンシ・タイマ・レジスタ

このレジスタは，レイテンシ・タイマの値をPCIバス・クロック数の単位で指定します。下位3ビットは，“0”に固定されているため，タイマの精度は8クロックです。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
7：0	レイテンシ・タイマ	R/W	R/W	0	レイテンシ・タイマの値をセットしてください。

保守／廃止

7. 5. 18. 10　ヘッダ・タイプ・レジスタ

このレジスタは，あらかじめヘッダ・レジスタに対して，さらに付加機能およびデバイス能力が複数あるかどうかを示します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
7：0	ヘッダ・タイプ	R	R	0	単機能でかつPCI-PCIブリッジではないため，“00H”固定。

7. 5. 18. 11　ウインドウ・メモリ・ベース・アドレス・レジスタ

このレジスタは，アクセス・ウインドウに対するPCIメモリ・スペースのベース・アドレスを指定します。

P_PLBAレジスタとは別個となるため，NICモードではPCIホストによって別途設定する必要があります。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31：0	ウインドウ・メモリ・ベース・アドレス	R/W（上位10ビット） R（下位22ビット）	R/W（上位10ビット） R（下位22ビット）	0	アクセス・ウインドウに対するメモリ・ベース・アドレス。 このレジスタは上位10ビットのみが書き込み可能です。下位22ビットは，2 Mバイトの領域が32ビットのメモリ・スペースに必要であることを示すために，“0”に固定されています。 1. 9　メモリ・マップの注を参照してください。

7. 5. 18. 12　レジスタ・メモリ・ベース・アドレス・レジスタ

このレジスタは，μPD98502のレジスタに対するPCIメモリ・スペースのベース・アドレスを指定します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
31：0	レジスタ・メモリ・ベース・アドレス	R/W（上位20ビット） R（下位12ビット）	R/W（上位20ビット） R（下位12ビット）	0	I/Oレジスタに対するメモリ・ベース・アドレス。 このレジスタは上位20ビットのみが書き込み可能です。下位12ビットは，4 Kバイトの領域が32ビットのメモリ・スペースに必要であることを示すために，“0”に固定されています。 1. 9　メモリ・マップの注を参照してください。

7. 5. 18. 13　サブシステム・ベンダIDレジスタ

このレジスタは，μPD98502を使用して，ユーザが拡張ボードやPCIデバイスを構成する際に，セットの製造会社を一意に識別するために使用します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
15：0	サブシステム・ベンダID	R/W	R	0	ユーザがこのレジスタにサブシステム・ベンダIDを設定します。

保守／廃止

7. 5. 18. 14　サブシステムIDレジスタ

このレジスタは，μPD98502を使用してセットを開発するユーザが，拡張ボードやPCIデバイスを構成する際に，ユーザ内でサブシステムを一意に識別するために使用します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
15：0	サブシステムID	R/W	R	0	ユーザがこのレジスタにサブシステムIDを設定します。

7. 5. 18. 15　Cap_Ptrレジスタ

μPD98502は，PCI rev.2.2の新しい機能，PCIパワー・マネジメント・インタフェース（PPMI）をサポートしています。このレジスタは，PPMIのデータの位置を示すために使用します。PPMIのデータを示すアドレス・ポインタは，コンフィギュレーション・スペースの“40H”から始まります。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
7：0	ポインタ	R	R	40H	“40H”固定。

7. 5. 18. 16　インタラプト・ライン・レジスタ

インタラプト・ライン・レジスタは，インタラプト・ライン配線情報のやりとりをするために使用します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
7：0	インタラプト・ライン	R/W	R/W	0	このレジスタの値は，μPD98502の割り込み端子を接続している割り込みコントローラの入力ラインを示します。

7. 5. 18. 17　インタラプト・ピン・レジスタ

このレジスタは，μPD98502がどの割り込み端子を使用するかを示します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
7：0	インタラプト・ピン	R	R	01H	“01H”固定。これは，μPD98502がINTA#端子を使用することを意味します。

7. 5. 18. 18　Min_Gntレジスタ

このレジスタは，クロック・レートを33 MHzとしたときにPCIコントローラが要するバースト時間の長さを指定します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
7：0	Min_Gnt	R/W	R	0	PCIコントローラのユーザは，VR4120Aによってこのレジスタにあらかじめ適切な値を設定してください。

保守／廃止

7. 5. 18. 19　Max_Latレジスタ

このレジスタは，デバイスがPCIバスを使用する際に必要とするレイテンシを指定します。

ビット	フィールド	R/W		ディフォールト	説　　明
		内部バス	PCI		
7：0	Max_Lat	R/W	R	0	値はVR4120Aによって設定してください。

7. 5. 18. 20　Cap_IDレジスタ

このレジスタは，新たに備えた機能の種類を示します。“01H”の値は，PCIパワー・マネジメント・インタフェース（PPMI）を意味します。

ビット	フィールド	R/W		ディフォールト	説　　明
		内部バス	PCI		
7：0	Cap_ID	R	R	01H	“01H”固定。これは，PCIパワー・マネジメント・インタフェースを意味します。

7. 5. 18. 21　Next_Item_Ptrレジスタ

このレジスタは，さらに新たに追加された機能の有無，およびそのポインタ値を示します。NULL（00H）は，これ以降新たな機能がないことを意味します。

ビット	フィールド	R/W		ディフォールト	説　　明
		内部バス	PCI		
7：0	Next_Item_Ptr	R	R	0	“00H”固定。

保守／廃止

7. 5. 18. 22　PMCレジスタ

このレジスタは，パワー・マネジメントに関連する機能についての情報を示します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
15：11	PME_Support	R/W	R	0	VR4120Aは，チップがPME#をアサートした際に，サポートするパワー状態のビットをセットします。
10	D2_Support	R	R	0	μPD98502はD2ステートをサポートしないため，“0”固定。
9	D1_Support	R/W	R	0	D1ステートをサポートする場合は，このビットはVR4120Aでセットしてください。
8：6	Aux_Current	R	R	000	μPD98502はD3coldステートからのPME#をサポートしないため，“000”固定。
5	DSI	R	R	0	μPD98502は特別な初期化を必要としないため，“0”固定。
4	Reserved	R	R	0	“0”固定。
3	PMEクロック	R/W	R	0	PME#を生成するためにPCIクロックが必要かどうかを示します。PME#は，66 MHzの内部クロックを使用して生成します。VR4120Aは，このビットに“1”をセットしなければなりません。
2：0	バージョン	R	R	010	PCIコントローラがPPMIのrev.1.1に準拠することを示す“010”固定。

保守／廃止

7. 5. 18. 23 PMCSRレジスタ

このレジスタは，PCI機能のパワー・マネジメント・ステートの管理とPMEのイネーブル／モニタに使用します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
15	PME_Staus	R	R/W	0	このビットは，PCIコントローラがPME_Enビットとは無関係にPME#（PME_B）信号をアサートするときにセットします。 このビットを“1”にセットすると，PCIコントローラは（PME_En = 1であれば）PME#（PME_B）のアサートを停止します。このビットを“0”にセットしても影響はありません。
14：13	Data_Scale	R	R	10	PCIコントローラは“0.01x”をデータ・レジスタの単位として使用しているため，“10”固定。
12：9	Data_Select	R	R/W	0	このフィールドは，どのデータをPMDataレジスタとData_Scaleフィールドを基に報告するかを選択するために使用します。フィールド値の定義は次のとおりです。 “0H” = D0消費電力 “1H” = D1消費電力 “3H” = D3消費電力 “4H” = D0損失電力 “5H” = D1損失電力 “7H” = D3損失電力 その他の値 = すべてゼロ
8	PME_En	R	R/W	0	“1”をセットすると，PME#（PME_B）をアサートする機能はイネーブルになります。“0”をセットすると，PME#（PME_B）をアサートする機能はディスエーブルになります。
7：2	Reserved	R	R	0	“0”固定。
1：0	PowerState	R/W	R/W	0	このフィールドは，現在のパワー状態を決定するためとPCIコントローラを新しいパワー状態に設定するための両方に使用します。フィールド値の定義は次のとおりです。 “00” = D0 “01” = D1 “11” = D3 その他の値をこのフィールドに書き込むと，PCIコントローラは現在のパワー状態のままとなります。

保守／廃止

7. 5. 18. 24　PMDataレジスタ

PMDataレジスタは，消費電力と各パワー状態での熱の放散を報告するために使用します。報告するデータの種類は，PMCSRレジスタのData_Selectフィールドで選択します。

ビット	フィールド	R/W		ディフォールト	説　明
		内部バス	PCI		
7：0	PMData	R	R	00H	このレジスタは，PMCSRレジスタ内のData_Selectフィールドで設定された各状態の消費電力または損失電力の値を示します。その際の表示スケールはData_Scaleフィールドに設定された×0.01倍となります。

保守／廃止

7.6　ソフトウエア処理

7.6.1　NICモード

7.6.1.1　初 期 化

(1) VR4120Aによる初期化

VR4120Aは，PCIコントローラに対する初期設定が完了するまでは，P_BCNTレジスタのINITDビットを“0”にセットし，PCI側からのすべてのアクセスに対して“リトライ”を発行するようにしてください。したがって，チップの初期化は，INITDビットを“1”にセットする前までに行わなければなりません。

VR4120Aに必要な初期化手順の例を以下のシーケンスにて示します。

- **システムの構成に応じて，コンフィギュレーション・スペースのサブシステム・ベンダIDレジスタ，サブシステムIDレジスタ，Min_Gntレジスタ，Max_Latレジスタをセットする。**
- **PME#を発生するためにPCIクロックが必要でなければ，コンフィギュレーション・スペース内のPMCレジスタの“PMEクロック”ビットに“1”をセットする。**
- **P_PLBAレジスタとP_IBBAレジスタにベース・アドレスをセットする。**
- **システムの構成に応じて，P_IIMRレジスタのマスク・ビットをイネーブルにする。**
- **パワーに関するデータをP_PWCDレジスタとP_PWDDレジスタにセットする。**
- **PCIコントローラが内部バス，I/O，またはメモリで使用するコマンドをP_BCNTレジスタのICMDSビットでセットする。**
- **データ転送のモードをP_BCNTレジスタのPDRTDビット，PPWRDビット，IDRTDビット，IPWRDビットで選択する。**
- **チップが動作可能であることを示すために，コンフィギュレーション・スペース内のPMCSRレジスタのPowerStateフィールドに“00”をセットする。**
- **パワー状態の遷移の発行が受け付け可能であることを示すために，P_PPCRレジスタのPMRDYビットに“1”をセットする。**
- **PCIコントローラの初期化が完了したことを示すために，P_BCNTレジスタのINITDビットに“1”をセットする。**

保守／廃止

(2) PCIホストによる初期化

INITDビットがセットされたあと，PCIコントローラはPCI側からのアクセスを受け付けることができます。外部PCIホスト・デバイスは，PCIコントローラがPCIデバイスとして動作できるように，PCIコントローラのコンフィギュレーション・スペースを構成しなければなりません。

外部PCIホスト・デバイスに必要な初期化手順の例を以下のシーケンスに示します。

- μPD98502がまだ初期化中の場合では，PCIホストからのコンフィギュレーション・アクセスにはリトライ応答が返されるので，PCIホスト側では初期化が終わるまで，繰り返しコンフィギュレーション・アクセスを行う。
- コマンド・レジスタの“メモリ・アクセス・イネーブル”ビットに“1”をセットする。
- チップがPCIマスタとしてトランザクションを実行する場合は，コマンド・レジスタの“バス・マスタ・イネーブル”ビットに“1”をセットする。
- システムの構成に応じて，コマンド・レジスタの“メモリ・ライト・アンド・インバリデートのイネーブル”ビットに適切な値をセットする。
- システムの構成に応じて，コマンド・レジスタの“パリティ・エラー応答”ビットに適切な値をセットする。
- システムの構成に応じて，コマンド・レジスタの“システム・エラー・イネーブル”ビットに適切な値をセットする。
- システムのキャッシュ・ライン・サイズを“キャッシュ・ライン・サイズ”レジスタにセットする。
- システムの構成に応じて，“レイテンシ・タイマ”レジスタをセットする。
- ベース・アドレスを“ウインドウ・メモリ・ベース・アドレス”レジスタと“レジスタ・メモリ・ベース・アドレス”レジスタにセットする。
- システムの構成に応じて，PMCSRレジスタのPME_Enビットに適切な値をセットする。

次に，PCIホスト・デバイスは内部レジスタを以下の手順で初期化します。

- システムの構成に応じて，P_IBBAレジスタにベース・アドレスの値をセットする。
- システムの構成に応じて，P_PIMRレジスタの使用する割り込み要因のマスク・ビットをイネーブルまたはディスエーブルに設定する。
- システムの構成に応じて，リトライ・タイマ・レジスタをセットする。

(3) エ ラ ー

7.2.3 **異常終了**に記述したエラーが発生した場合，PCIコントローラは，コンフィギュレーション・スペースのステータス・レジスタ，P_IGSRレジスタ，P_PGSRレジスタの該当するビットをセットし，マスクされていなければ，割り込みをVR4120Aと外部PCIホスト・デバイスに発行します。VR4120Aと外部PCIホスト・デバイスは，これらのエラー状態をどのように処理するかを判断します。PCIコントローラは，現在のトランザクションを停止し，新しいアクセスを受け付けることができる状態に戻します。

(4) ソフトウエア・リセット

PCIホストがソフトウエア・リセットのためにP_SWRRレジスタに適当な値（どのような値でもかまいません）を書き込むと，PCIコントローラは，P_IGSRレジスタのSRREQビットをセットし，マスクされていなければ，割り込みでVR4120Aに報告します。しかし，PCIコントローラは，このシーケンスの中では自分自身をリセットしません。

その際VR4120Aが，チップの各ブロックにウォーム・リセット信号またはコールド・リセット信号をアサートしなければなりません。

(5) パワー状態の遷移

PCIホストがチップのパワー状態を変更するためにデータをコンフィギュレーション・スペース内のPMCSRレジスタのPowerStateフィールドに次に遷移したい状態のコードを書き込むと，PCIコントローラは，P_PPCRレジスタのPMRDYビットをリセットし，P_IGSRレジスタのPPREQビットをセットし，マスクされていなければ，割り込みでVR4120Aに報告します。どの遷移が必要であるかは，PMCSRレジスタのPMRQ0ビット，PMRQ1ビット，PMRQ3ビットで示します。

このパワー状態遷移では，PCIコントローラ自体はチップのパワー状態に関わる処理を行いません。VR4120Aがパワー状態の処理を行います。

VR4120Aがパワー状態の遷移を必要とする場合は，P_PPCRレジスタのPMERQビットに“1”を書き込んでPME#（PME_B）をアサートし，状態遷移要求を発生することができます。

7.6.2 ホスト・モード

ホスト・モードでは，PCIバスのホストはPCIコントローラ自身です。これは，VR4120Aが初期化に責任があることを意味します。

7.6.2.1 初 期 化

ホスト・モードでは，PCIバスのホストはμPD98502のPCIコントローラ自身です。したがって，VR4120AがチップでけでなくPCIバス上のすべてのPCIデバイスの初期化に責任があることを意味します。

PCIコントローラは，P_BCNTレジスタのINITDビットが“1”にセットするまでは，PCI側からのすべてのアクセスに“リトライ”を返します。

コンフィギュレーション・レジスタの初期化の例を以下のシーケンスに示します。

- コンフィギュレーション・スペースのサブシステム・ベンダIDレジスタ，サブシステムIDレジスタ，Min_Gntレジスタ，Max_Latレジスタをセットする。
- コマンド・レジスタの“メモリ・アクセス・イネーブル”ビットに“1”をセットする。
- チップがPCIマスタとしてトランザクションを実行する場合は，コマンド・レジスタの“バス・マスタ・イネーブル”ビットに“1”をセットする。
- システムの構成に応じて，コマンド・レジスタの“メモリ・ライト・アンド・インバリデートのイネーブル”ビットに適切な値をセットする。
- システムの構成に応じて，コマンド・レジスタの“パリティ・エラー応答”ビットに適切な値をセットする。
- システムの構成に応じて，コマンド・レジスタの“システム・エラー・イネーブル”ビットに適切な値をセットする。
- システムのキャッシュ・ライン・サイズを“キャッシュ・ライン・サイズ”レジスタにセットする。
- システムの構成に応じて，“レイテンシ・タイマ”レジスタをセットする。

- ベース・アドレスを“ウインドウ・メモリ・ベース・アドレス”レジスタと“レジスタ・メモリ・ベース・アドレス”レジスタにセットする。

内部レジスタの初期化は以下のように行います。

- P_PLBAレジスタとP_IBBAレジスタにベース・アドレスをセットする。
- システムの構成に応じて，P_IIMRレジスタのマスク・ビットをイネーブルにする。
- システムの構成に応じて，アービタ・モードをP_HMCRレジスタのPARBMビットでセットする。
- PCIコントローラが内部バスに対するアクセスとして，I/O，またはメモリのいずれで使用するかをP_BCNTレジスタのICMDSビットでセットする。
- PCIコントローラはDAC（Dual Address Cycle）で使用できないためP_BCNTレジスタのDACENビットに“0”をセットする。
- システムの構成に応じて，データ転送のモードをP_BCNTレジスタのPDRTDビット，PPWRDビット，IDRTDビット，IPWRDビットで選択する。
- チップが動作可能であることを示すために，コンフィギュレーション・スペース内のPMCSRレジスタのPowerStateフィールドに“00”をセットする。
- PCIコントローラの初期化が完了したことを示すために，P_BCNTレジスタのINITDビットに“1”をセットする。

INITDビットがセットされたあと，PCIコントローラはPCI側からのアクセスを受け付けることができます。

(1) エ ラ ー

7. 2. 3 **異常終了**に記述したエラーが発生した場合，PCIコントローラは，コンフィギュレーション・スペースのステータス・レジスタ，P_IGSRレジスタ，P_PGSRレジスタの該当するビットをセットし，マスクされていなければ，割り込みをVR4120Aに発行します。VR4120Aは，これらのエラー状態をどのように処理するかを判断します。PCIコントローラは，現在のトランザクションを停止し，新しいアクセスを受け付けることができる状態に戻します。

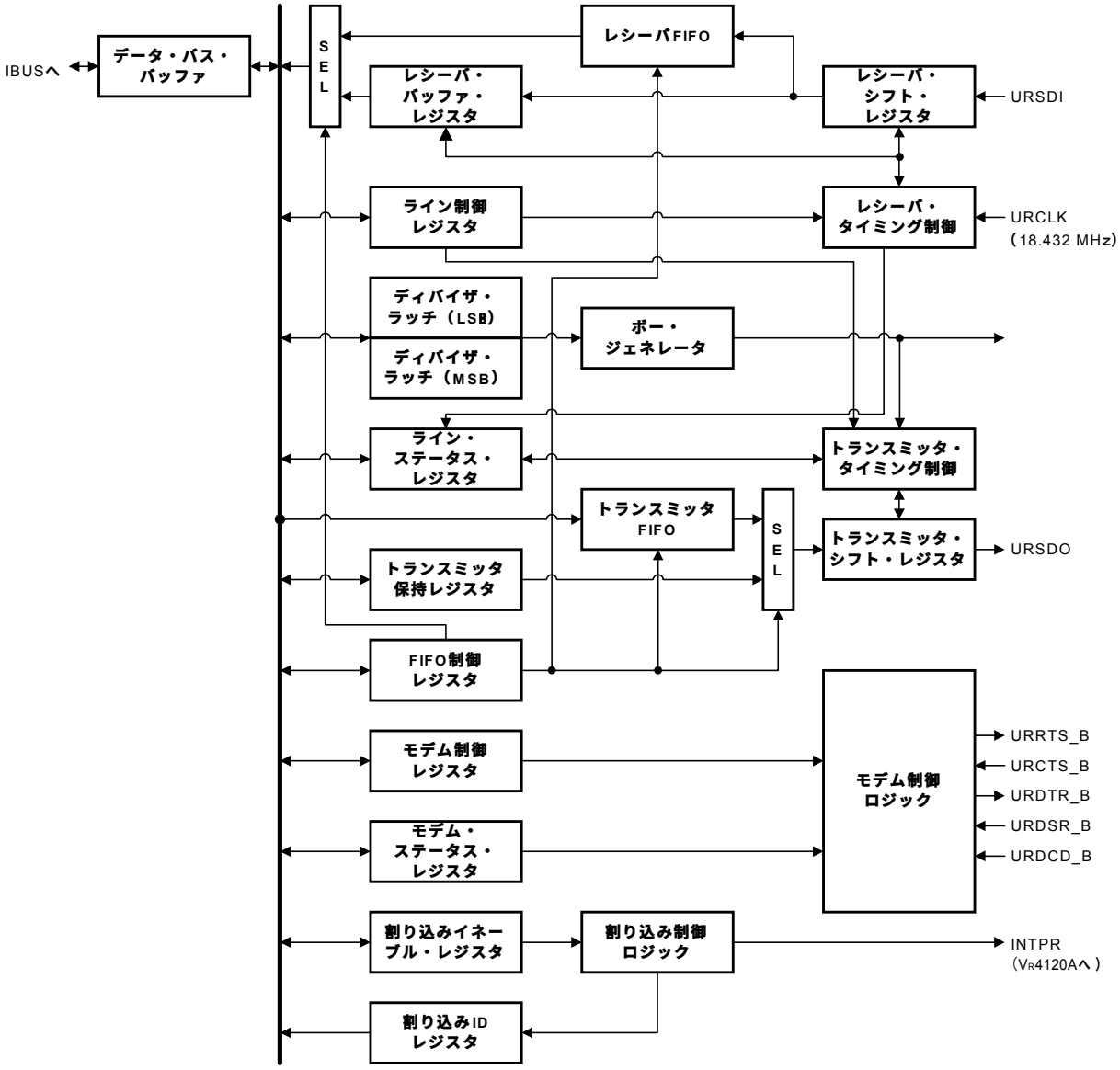
保守／廃止

第8章　UART

8.1 概　　要

UARTは，RS-232C規格に準拠したシリアル・インタフェースであり，それぞれ送信用と受信用に1つのチャネル・インタフェースを備えています。このユニットは，機能的にNational Semiconductor社（米国）製NS16550Dと互換性があります。

8.2 UARTのブロック図



保守／廃止

8.3 レジスタ

このコントローラは，NECエレクトロニクスのNX16550Lメガ・マクロを内部UARTとして使用しています。このUARTは，機能的にNational Semiconductor社のNS16550Dと同じです。

8.3.1 レジスタ・マップ

アドレス	レジスタ名	R/W	アクセス	説　明
1000_0080H	UARTRBR	R	W/H/B	UARTレシーバ・バッファ・レジスタ［DLAB = 0，リード］
1000_0080H	UARTTHR	W	W/H/B	UARTトランスミッタ・データ保持レジスタ［DLAB = 0，ライト］
1000_0080H	UARTDLL	R/W	W/H/B	UARTディバイザ・ラッチLSBレジスタ［DLAB = 1］
1000_0084H	UARTIER	R/W	W/H/B	UART割り込みイネーブル・レジスタ［DLAB = 0］
1000_0084H	UARTDLM	R/W	W/H/B	UARTディバイザ・ラッチMSBレジスタ［DLAB = 1］
1000_0088H	UARTIIR	R	W/H/B	UART割り込みIDレジスタ［リード］
1000_0088H	UARTFCR	W	W/H/B	UART FIFO制御レジスタ［ライト］
1000_008CH	UARTLCR	R/W	W/H/B	UARTライン制御レジスタ
1000_0090H	UARTMCR	R/W	W/H/B	UARTモデム制御レジスタ
1000_0094H	UARTLSR	R/W	W/H/B	UARTライン・ステータス・レジスタ
1000_0098H	UARTMSR	R/W	W/H/B	UARTモデム・ステータス・レジスタ
1000_009CH	UARTSCR	R/W	W/H/B	UARTスクラッチ・レジスタ

備考1. “R/W”フィールド内で，
“W”は，“書き込み可能”を意味します。
“R”は，“読み取り可能”を意味します。
“RC”は，“読み取りクリア”を意味します。
“-”は，“アクセス不可能”を意味します。

2. すべての内部レジスタは，32ビット・ワード配列のレジスタです。
3. 内部レジスタに対するバースト・アクセスは，禁止されています。バースト・アクセスすると，S_NSRレジスタのIRERRビットがセットされ，NMIがVR4120Aに対してアサートします。
4. 予約領域に対してリード・アクセスすると，S_NSRレジスタのCBERRビットがセットされ，SysCMD［0］にデータ・エラー・ビットが設定されたダミー読み取り応答データが返信されます（一部の非公開レジスタを除きます）。
5. 予約領域に対してライト・アクセスすると，S_NSRレジスタのCBERRビットがセットされ，書き込みデータが失われます。
6. “アクセス”フィールド内で，
“W”は，ワード・アクセスが有効であることを意味します。
“H”は，ハーフ・ワード・アクセスが有効であることを意味します。
“B”は，バイト・アクセスが有効であることを意味します。
7. リード・オンリー・レジスタに対するライト・アクセスはエラーになりませんが，書き込みデータは失われます。
8. VR4120Aはすべての内部レジスタにアクセスできますが，IBUSマスタ・デバイス（イーサネット・コントローラ，USBコントローラ，PCIコントローラなど）は，それらのレジスタにアクセスできません。

保守／廃止

8. 3. 2 UARTRBR（UARTレシーバ・データ・バッファ・レジスタ）

このレジスタは，受信データを保持します。UARTLCRのディバイザ・ラッチ・アクセス・ビット（DLAB）がクリアされたときにのみリード・アクセスできます。

ビット	フィールド	R/W	ディフォールト	説　明
31：8	Reserved	R	0	0固定
7：0	UDATA	R	-	UART受信データ（リード・オンリー）

8. 3. 3 UARTTHR（UARTトランスミッタ・データ保持レジスタ）

このレジスタは，送信データを保持します。UARTLCRのディバイザ・ラッチ・アクセス・ビット（DLAB）がクリアされたときにのみライト・アクセスできます。

ビット	フィールド	R/W	ディフォールト	説　明
31：8	Reserved	W	0	0固定
7：0	UDATA	W	-	UART送信データ（ライト・オンリー）

保守／廃止

8. 3. 4 UARTIER（UART割り込みイネーブル・レジスタ）

このレジスタは，UART割り込みとして報告する割り込み要因をイネーブルにするために使用します。UARTLCRのディバイザ・ラッチ・アクセス・ビット（DLAB）がセットされたときにのみアクセスできます。UARTIM（システム・コントローラの割り込みマスク・レジスタ“S_IMR”のビット2）は，このレジスタによってイネーブルになる割り込み要因に対して，さらにマスクを制御します。

ビット	フィールド	R/W	ディフォールト	説　明
31：4	Reserved	R/W	0	0固定
3	ERBMI	R/W	0	UARTモデム・ステータス割り込み： 1 = モデム・ステータス変化割り込みをイネーブルにする。 0 = モデム・ステータス変化割り込みをディスエーブルにする。 モデム・ステータスの変化内容は，UARTMSRに報告されます。
2	ERBLI	R/W	0	UARTライン・ステータス割り込み： 1 = ライン・ステータス・エラー割り込みをイネーブルにする。 0 = ライン・ステータス・エラー割り込みをディスエーブルにする。 ライン・ステータス・エラー割り込み状態は，UARTLSRに報告されます。
1	ERBEI	R/W	0	UARTトランスミッタ・バッファ・エンプティ割り込み： 1 = トランスミッタ・バッファ・エンプティ割り込みをイネーブルにする。 0 = トランスミッタ・バッファ・エンプティ割り込みをディスエーブルにする。 トランスミッタ・バッファ・エンプティ状態は，UARTLSRに報告されます。
0	ERBFI	R/W	0	受信FIFOフル割り込み： 1 = 受信FIFOバッファ・フル割り込みをイネーブルにする。 0 = 受信FIFOバッファ・フル割り込みをディスエーブルにする。 UARTレシーバFIFOの受信データ数がUARTFCRのURFTRで設定した値を越えたことを検出した割り込みに対してイネーブルを設定します。

8. 3. 5 UARTDLL（UARTディバイザ・ラッチLSBレジスタ）

このレジスタは，ボー・レート・ジェネレータにディバイザ（分周レート）の下位バイトを設定するために使用します。このレジスタのデータとUARTDLMレジスタの下位8ビット・データの計16ビット・データが分周比となります。

ビット	フィールド	R/W	ディフォールト	説　明
31：8	Reserved	R/W	0	0固定
7：0	DIVLSB	R/W	-	UARTディバイザ・ラッチ（表8－1参照）： UARTLCRでDLAB = 1のときにのみアクセスできます。

保守／廃止

8. 3. 6 UARTDLM（UARTディバイザ・ラッチMSBレジスタ）

このレジスタは，ボー・レート・ジェネレータにディバイザ（分周レート）の上位バイトを設定するために使用します。このレジスタのデータとUARTDLLレジスタの下位8ビット・データの計16ビット・データが分周比となります。

ビット	フィールド	R/W	ディフォールト	説　明
31：8	Reserved	R/W	0	0固定
7：0	DIVMSB	R/W	-	UARTディバイザ・ラッチ（表8－1参照）： UARTLCRでDLAB = 1のときにのみアクセスできます。

表8－1　ボー・レートとディバイザの関係

ボー・レート [bps]	UARTソース・クロック周波数					
	18.432 MHz（外部クロック使用）		33.000 MHz（CPUクロック = 66 MHz）		50.000 MHz（CPUクロック = 100 MHz）	
	ディバイザ	誤差	ディバイザ	誤差	ディバイザ	誤差
50	23040（5A00H）	0	41250（A122H）	＞1 %	62500（F424H）	0
75	15360（3C00H）	0	27500（6B6CH）	＞1 %	41667（A2C3H）	＞1 %
110	10473（28E9H）	0	18750（493EH）	＞1 %	28409（6EF9H）	＞1 %
134.5	8565（2175H）	0	15335（3BE7H）	＞1 %	23234（5AC2H）	＞1 %
150	7680（1E00H）	0	13750（35B6H）	＞1 %	20833（5161H）	＞1 %
300	3840（F00H）	0	6875（1ADBH）	＞1 %	10417（28B1H）	＞1 %
600	1920（780H）	0	3438（D6EH）	＞1 %	5208（1458H）	＞1 %
1200	920（398H）	0	1719（6B7H）	＞1 %	2604（A2CH）	＞1 %
1800	640（280H）	0	1146（47AH）	＞1 %	1736（6C8H）	＞1 %
2000	573（23DH）	0	1031（407H）	＞1 %	1562（61AH）	＞1 %
2400	480（1E0H）	0	859（35BH）	＞1 %	1302（516H）	＞1 %
3600	320（140H）	0	573（23DH）	＞1 %	868（364H）	＞1 %
4800	240（F0H）	0	430（1AEH）	＞1 %	651（28BH）	＞1 %
7200	160（A0H）	0	286（11EH）	＞1 %	434（1B2H）	＞1 %
9600	120（78H）	0	215（D7H）	＞1 %	326（146H）	＞1 %
19200	60（3CH）	0	107（6BH）	＞1 %	163（A3H）	＞1 %
38400	30（1EH）	0	54（36H）	＞1 %	81（51H）	＞1 %
57600	20（14H）	2.0 %	36（24H）	＞1 %	54（36H）	＞1 %
128000	9（9H）	0	16（10H）	＞1 %	24（18H）	1.69 %
144000	8（8H）	0	14（EH）	2.25 %	22（16H）	1.38 %
192000	6（6H）	0	11（BH）	2.41 %	16（10H）	1.69 %
230400	5（5H）	0	9（9H）	＞1 %	14（EH）	3.22 %
288000	4（4H）	0	7（7H）	2.25 %	11（BH）	1.38 %
384000	3（3H）	0	5（5H）	6.90 %	8（8H）	1.69 %
576000	2（2H）	0	4（4H）	11.7 %	5（5H）	7.83 %
1152000	1（1H）	0	2（2H）	11.7 %	3（3H）	10.6 %

備考 S_GMRレジスタのUCSELビットを1にセットすると，UART外部クロック入力（URCLK）をUARTソース・クロックとして使用します。

UCSELビットを0にセットすると，CPUクロックの1/2をUARTソース・クロックとして使用します。

保守／廃止

8. 3. 7 UARTIIR（UART割り込みIDレジスタ）

このレジスタは，割り込みの優先順位と保留中の割り込みの有無を示します。UART割り込みの優先順位は，最高位から順に，受信ライン状態，受信データ・レディ，キャラクタ・タイムアウト，送信保持レジスタ・エンプティ，モデム状態となっています。ビット3は，FIFOモードでのみ有効で，16540モードでは常に0です。ビット2は，ビット3が1にセットされたときに，1になります。

このレジスタはリード・クリアではありません。ただし，UIID = 001が読み出された場合のみ，次にこのレジスタを読み出すとこの要因がクリアされており，他の割り込み要因がなければビット0のINTPENDLは1となります。

ビット	フィールド	R/W	ディフォールト	説　明
31：8	Reserved	R	0	0固定
7：6	UFIFOEN	R	00	UART FIFOがイネーブル： この2ビットは，UARTFCRのUFIFOEN0ビットをセットし，送信／受信FIFOがイネーブルになると，1にセットされます。
5：4	Reserved	R	00	0固定
3：1	UIID	R	000	保留中の割り込みの優先順位レベル： 011 = 第1優先順位：受信ライン状態 オーバラン・エラー，パリティ，フレーミング・エラー，ブレーク割り込み 010 = 第2優先順位：受信データあり 受信データあり，またはトリガ・レベルに達した 110 = 第3優先順位：キャラクタ・タイムアウト表示 最後の4つのキャラクタ・タイムの間，レシーバFIFOに変更がなく，FIFOが空ではない 001 = 第4優先順位：UARTTHRレジスタのデータが空（エンプティ） 000 = 第5優先順位：モデム状態（CTS_L, DSR_L, DCD_L）
0	INTPENDL	R	1	保留中の割り込み： 0 = UART割り込み保留中 1 = 保留中のUART割り込みなし

保守／廃止

8. 3. 8 UARTFCR（UART FIFO制御レジスタ）

このレジスタは，FIFO（FIFOイネーブル，FIFOクリア）を制御し，受信FIFOトリガ・レベルを設定します。

ビット	フィールド	R/W	ディフォールト	説　明
31：8	Reserved	W	0	0固定
7：6	URFTR	W	00	UART受信FIFOトリガ・レベル： 受信FIFOの受信データ数がこのフィールドで設定したトリガ・レベルに達し，かつUARTIERのERBFIビットがイネーブルの場合，受信バッファ・フル割り込みが発生します。トリガ・レベルとなる受信FIFO内のバイト数は次のとおりです。 00 = 1バイト 01 = 4バイト 10 = 8バイト 11 = 14バイト
5：4	Reserved	W	0	0固定
3	FIFOMD	W	0	16450モード/FIFOモード設定： 1 = FIFOモードに設定する 0 = 16450モードに設定する
2	UTFRST	W	0	UARTトランスミッタFIFOリセット： 1 = 送信FIFOをクリアし，カウンタをリセットする 0 = クリアしない
1	URFRST	W	0	UARTレシーバFIFOリセット： 1 = 受信FIFOをクリアし，カウンタをリセットする 0 = クリアしない
0	UFIFOEN0	W	0	UARTFIFOイネーブル： 1 = 送信／受信FIFOをイネーブルにする 0 = 送信／受信FIFOをディスエーブルにしてクリアする

保守／廃止

8. 3. 9 UARTLCR（UARTライン制御レジスタ）

このレジスタは，非同期通信のフォーマットを指定し，ディバイザ・ラッチ・アクセス・ビットをセットするために使用します。ビット6は，ブレーク信号を送信するために使用します。ビット6 = 1のとき，シリアル出力（URSDO）を強制的にスペーシング（0）状態にします。ビット5の設定は，ビット4, 3の設定に応じて有効になります。

ビット	フィールド	R/W	ディフォールト	説　明
31：8	Reserved	R/W	0	0固定
7	DLAB	R/W	0	ディバイザ・ラッチ・アクセス・ビット： 1 = UARTDLL, UARTDLMの各レジスタをアクセスする前に設定する 0 = UARTRBR, UARTTHR, UARTIERの各レジスタをアクセスする前に設定する
6	USB	R/W	0	送信ブレーク： 1 = URSDO信号出力を強制的にロウ・レベルにする 0 = 通常動作
5	USP	R/W	0	パリティを固定： 1 = パリティ・ビット送信時にURSDO信号出力を強制的に0にする 0 = 通常動作
4	EPS	R/W	0	パリティ選択： 1 = 偶数パリティ 0 = 奇数パリティ
3	PEN	R/W	0	パリティ・イネーブル： 1 = パリティの生成とパリティ・チェックをイネーブルにする 0 = パリティの生成やチェックなし UARTに対して，EPSビット（ビット4）で指定した偶数パリティまたは奇数パリティを生成し，チェックできます。
2	STB	R/W	0	ストップ・ビット数： 1 = ストップ・ビットとして送信するビット数として，キャラクト長が5ビットの場合（WLS = 00）は1.5ビット，キャラクタ長が6～8ビットの場合（WLS = 01, 10, 11）は2ビットを設定する 0 = ストップ・ビットとして送信するビット数として1ビットを設定する。
1：0	WLS	R/W	00	キャラクタ長選択： 11 = 8ビット 10 = 7ビット 01 = 6ビット 00 = 5ビット

保守／廃止

8. 3. 10 UARTMCR（UARTモデム制御レジスタ）

このレジスタは， URDTR_Bモデム制御信号とURRTS_Bモデム制御信号の状態およびループバック・テストの状態を制御します。

ビット	フィールド	R/W	ディフォールト	説　明
31：5	Reserved	-	0	0固定
4	LOOP	R/W	0	ループバック・テスト： 1 = ループバック 0 = 通常動作 当社の出荷時に使用するテスト機能用のビットです。ユーザは必ず0にセットしてください。
3	Reserved	-	0	0固定
2	Reserved	-	0	0固定
1	RTS	R/W	0	送信要求： 1 = URRTS_B信号をディアサートにする。 0 = URRTS_B信号をアサートする。
0	DTR	R/W	0	データ・ターミナル・レディ： 1 = URDTR_B信号をディアサートにする。 0 = URDTR_B信号をアサートする。

保守／廃止

8. 3. 11 UARTLSR（UARTライン・ステータス・レジスタ）

このレジスタは，トランスミッタの現在の状態とレシーバ・ロジックを報告します。

ビット	フィールド	R/W	ディフォールト	説　明
31：8	Reserved	R/W	0	0固定
7	RFERR	R/W	0	レシーバFIFOエラー： 1 = 受信FIFOにあるレシーバ・データの1つ以上でパリティ・エラー，フレーミング・エラー，ブレーク・エラーを検出 0 = エラーなし（16450モードでは常に0となります）
6	TEMT	R/W	1	トランスミッタ・エンプティ： 1 = UARTTHRのデータが空で，かつシフト・レジスタも空である 0 = UARTTHRにデータが存在するか，またはシフト・レジスタにデータが存在する
5	THRE	R/W	1	UARTTHRエンプティ： 1 = UARTTHRが空である 0 = UARTTHRにデータが存在する 送信データは，UARTTHRに格納されます。
4	BI	R/W	0	ブレーク割り込み： 1 = ブレークを受信 0 = ブレークなし
3	FE	R/W	0	受信データのフレーミング・エラー： 1 = 受信データでフレーミング・エラー検出 0 = エラーなし
2	PE	R/W	0	受信データのパリティ・エラー： 1 = 受信データでパリティ・エラー検出 0 = エラーなし
1	OE	R/W	0	受信データのオーバラン・エラー： 1 = 受信データでオーバラン・エラー検出 0 = エラーなし
0	DR	R/W	0	受信データ・レディ： 1 = 受信FIFOおよびUARTRBRに受信データが存在する 0 = 受信FIFOおよびUARTRBRのいずれにも受信データが存在しない 受信データは，UARTRBRに格納されます。

保守／廃止

8. 3. 12 UARTMSR（UARTモデム・ステータス・レジスタ）

このレジスタは，さまざまな制御信号の現在の状態と変更を報告します。

ビット	フィールド	R/W	ディフォールト	説　明
31：8	Reserved	R/W	0	0固定
7	DCD	R/W	0	データ・キャリア検出： 1 = URDCD_B状態がロウ・レベル 0 = URDCD_B状態がハイ・レベル このビットは，URDCD_B入力信号レベルの補数を示します。
6	Reserved	R/W	0	0固定
5	DSR	R/W	0	データ・セット・レディ： 1 = URDSR_B状態がロウ・レベル 0 = URDSR_B状態がハイ・レベル このビットは，URDSR_B入力信号レベルの補数を示します。
4	CTS	R/W	0	送信クリア： 1 = URCTS_B状態がロウ・レベル 0 = URCTS_B状態がハイ・レベル このビットは，URCTS_B入力信号レベルの補数を示します。
3	DDCD	R/W	0	デルタ・データ・キャリア検出： 1 = このレジスタを前回読み込んでから，さらにURDCD_B入力信号の状態に変更があった。 0 = 変更はなかった。
2	Reserved	R/W	0	0固定
1	DDSR	R/W	0	デルタ・データ・セット・レディ： 1 = このレジスタを前回読み込んでから，さらにURDSR_B入力信号の状態に変更があった。 0 = 変更はなかった。
0	DCTS	R/W	0	デルタ・クリア・ツー・センド： 1 = このレジスタを前回読み込んでから，さらにURCTS_B入力信号の状態に変更があった。 0 = 変更はなかった。

保守／廃止

8. 3. 13 UARTSCR（UARTスクラッチ・レジスタ）

このレジスタは，UARTの制御に使用するためのものではなく，プログラミングの際のデータの一時保管に使用します。

ビット	フィールド	R/W	ディフォールト	説　明
31：8	Reserved	R/W	0	0固定
7：0	USCR	R/W	-	UARTスクラッチ・レジスタ： プログラミングの際にデータを一時保管します。

保守／廃止

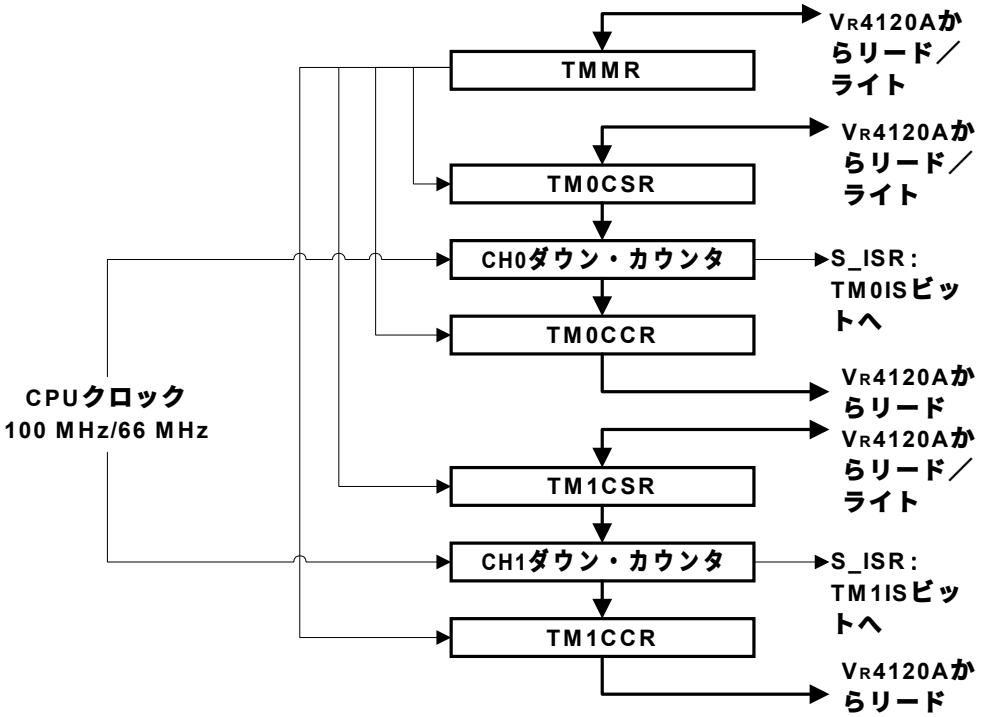
第9章　タ イ マ

9.1 概　　要

μPD98502には2つのタイマがあります。これらのタイマは，システム・クロック（VR4120Aの動作クロック）をカウントします。VR4120Aによるリード／ライトが可能です。タイマがカウント中でもVR4120Aによる読み込みが可能です。2つのタイマは，カウンタ値をダウン・カウントして，0になるとVR4120Aに割り込みを発行します。そして，“タイマ・セット・カウント・レジスタ”値が自動的にリロードされ，再スタートします。割り込みは，マスク／マスク解除が可能です。

システム・コントローラの割り込みステータス・レジスタ“S_ISR”のTM0ISフィールドとTM1ISフィールドは，それぞれのカウンタがタイムアウト（カウント値が0）したことを示します。1にセットされた場合，すでに終了したタイマ・イベントがあることを示します。すべてのタイマはカウント・ダウン方式となっています。“TM0CSR”または“TM1CSR”は，カウンタ値を保持するレジスタです。TM0CCRまたはTM1CCRは，VR4120Aがリード・アクセスした時点での各タイマのカウンタ値を反映するレジスタです。TM0CCRとTM1CCRは独立しており互いへの影響はありません。TM0CSRまたはTM1CSRに値をロードすると，VR4120Aから新たな値がライトされるまでその値を保持します。“TMMR”のタイマCH0/CH1イネーブル・ビットをセットすると，TM0CSRまたはTM1CSRの値が各ダウン・カウンタにロードされ，ダウン・カウントを始めます。タイムアウトによる割り込みはVR4120Aがシステム・コントローラの割り込みステータス・レジスタ“S_ISR”を読み込むと，自動的にクリアされます。

9.2 ブロック図



保守／廃止

9.3 レジスタ

9.3.1 レジスタ・マップ

アドレス	レジスタ名	R/W	アクセス	説　明
1000_00B0H	TMMR	R/W	W/H/B	タイマ・モード・レジスタ
1000_00B4H	TM0CSR	R/W	W/H/B	タイマCH0カウント・セット・レジスタ
1000_00B8H	TM1CSR	R/W	W/H/B	タイマCH1カウント・セット・レジスタ
1000_00BCH	TM0CCR	R	W/H/B	タイマCH0カレント・カウント・レジスタ
1000_00C0H	TM1CCR	R	W/H/B	タイマCH1カレント・カウント・レジスタ

備考1. “R/W”フィールド内で，
“W”は，“書き込み可能”を意味します。
“R”は，“読み取り可能”を意味します。
“RC”は，“読み取りクリア”を意味します。
“-”は，“アクセス不可能”を意味します。

2. すべての内部レジスタは，32ビット・ワード配列のレジスタです。
3. 内部レジスタに対するバースト・アクセスは，禁止されています。バースト・アクセスすると，S_NSRレジスタのIRERRビットがセットされ，NMIがVR4120Aに対してアサートします。
4. 予約領域に対するリード・アクセスにより，S_NSRレジスタのCBERRビットがセットされ，SysCMD［0］にデータ・エラー・ビットが設定されたダミー読み取り応答データが返信されます（一部の非公開レジスタを除きます）。
5. 予約領域に対するライト・アクセスにより，S_NSRレジスタのCBERRビットがセットされ，書き込みデータが失われます。
6. “アクセス”フィールド内で，
“W”は，ワード・アクセスが有効であることを意味します。
“H”は，ハーフ・ワード・アクセスが有効であることを意味します。
“B”は，バイト・アクセスが有効であることを意味します。
7. リード・オンリー・レジスタに対するライト・アクセスはエラーになりませんが，書き込みデータは失われます。
8. VR4120Aはすべての内部レジスタにアクセスできますが，IBUSマスタ・デバイス（イーサネット・コントローラ，USBコントローラ，PCIコントローラなど）は，それらのレジスタにアクセスできません。

保守／廃止

9.3.2 TMMR（タイマ・モード・レジスタ）

タイマ・モード・レジスタ“ TMMR ”は，リード／ライト可能で32ビット・ワード配列のレジスタです。TMMRは，タイマ制御のために使用します。TMMRには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：9	Reserved	R/W	0	0固定
8	TM1EN	R/W	0	タイマCH1イネーブル： 1 = イネーブル，タイマCH1をスタートする （ダウン・カウンタの値が0になったらTM1CSRの設定値を自動的にリロードし，再スタートする） 0 = ディスエーブル，タイマを停止する
7：1	Reserved	R/W	0	0固定
0	TM0EN	R/W	0	タイマCH0イネーブル： 1 = イネーブル，タイマCH0をスタートする （ダウン・カウンタの値が0になったらTM0CSRの設定値を自動的にリロードし，再スタートする） 0 = ディスエーブル，タイマを停止する

9.3.3 TM0CSR（タイマCH0カウント・セット・レジスタ）

タイマCH0カウント・セット・レジスタ “ TM0CSR ” は，リード／ライト可能で32ビット・ワード配列のレジスタです。VR4120Aは，このレジスタにカウンタ値をロードし，カウンタは（TM0CSR－1）の値からカウント・ダウンを始めます。カウントが0000_0000Hに達すると，S_ISRのTM0ISがS_IMRのTM0IMでマスクされていなければ，割り込みステータス・レジスタ“ S_ISR ”を介してVR4120Aに割り込みを発行します。TM0CSRには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	TM0SET	R/W	0	タイマCH0の初期値とリロード値

9.3.4 TM1CSR（タイマCH1カウント・セット・レジスタ）

タイマCH1カウント・セット・レジスタ “ TM1CSR ” は，リード／ライト可能で32ビット・ワード配列のレジスタです。VR4120Aは，このレジスタにカウント値をロードし，カウンタは（TM1CSR－1）の値からカウント・ダウンを始めます。カウントが0000_0000Hに達すると，S_ISRのTM1ISがS_IMRのTM1IMでマスクされていなければ，割り込みステータス・レジスタ“ S_ISR ”を介してVR4120Aに割り込みを発行します。TM1CSRには以下のフィールドがあり，リセット時に0に初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	TM1SET	R/W	0	タイマCH1の初期値とリロード値

保守／廃止

9.3.5 TM0CCR（タイマCH0カレント・カウント・レジスタ）

タイマCH0カレント・カウント・レジスタ“TM0CCR”は，リード・オンリーで32ビット・ワード配列のレジスタです。VR4120Aは，このレジスタから値を読み込んでタイマCH0の現在のカウント値を得ることができます。TM0CCRには以下のフィールドがあり，リセット時にFFFF_FFFFHに初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	TM0CNT	R	FFFF_FFFFH	タイマCH0の現在のカウント値

9.3.6 TM1CCR（タイマCH1カレント・カウント・レジスタ）

タイマCH1カレント・カウント・レジスタ“TM1CCR”は，リード・オンリーで32ビット・ワード配列のレジスタです。VR4120Aは，このレジスタから値を読み込んでタイマCH1の現在のカウント値を得ることができます。TM1CCRには以下のフィールドがあり，リセット時にFFFF_FFFFHに初期化されます。

ビット	フィールド	R/W	ディフォールト	説　明
31：0	TM1CCR	R	FFFF_FFFFH	タイマCH1の現在のカウント値

保守／廃止

第10章　Micro Wire

10.1 概　　要

このインタフェースは，Micro Wireシリアル・インタフェースと互換性があるEEPROMインタフェースです。National Semiconductor社のシリアルEEPROM“NM93C46”に接続することを推奨します。

シリアルEEPROMメモリ領域は，Micro Wireレジスタ（ECCRとERDRレジスタ）を使用してアクセスされます。EEPROMにアクセスするために，VR4120Aは，コマンドをECCRレジスタに書き込みます。このブロックは，このコマンドを受け付けると，EEPROMインタフェースを介してコマンドを実行します。EEPROM内のデータを読み込むために，VR4120Aは，アドレスとREADコマンドをECCRレジスタに設定します。このブロックがデータを読み込んでいる間，ERDRレジスタのMSBビットは1にセットされます。このブロックがデータの読み込みを終了すると，MSBを0にセットし，データをERDRレジスタのREAD_DATAフィールドに格納します。コマンドを発行したあと，VR4120Aは，ERDRレジスタのMSBビットが0にセットされたことを確認し，データを読み込みます。EEPROMにデータの書き込みや削除を行うために，VR4120Aは，あらかじめEWENコマンドを用いて書き込み動作と削除動作をイネーブルにする必要があります。EEPROMが接続されていない場合，これらのレジスタにアクセスしても無効となります。

Micro Wireインタフェースは，自動ロード機能も持っています。この機能によって，ユーザは，ECCRレジスタの制御なしに，EEPROMの12バイト・データ（1Hから6Hまでのハーフ・ワード単位）をMACAR1レジスタからMACAR3レジスタまで読み込むことができます。この自動ロード機能は，システム・リセット後に1回働きます。その際，あらかじめEEPROMに対して表10－1に示す初期データ・フォーマットで書き込み済みである必要があります。

自動ローディングとECCRレジスタによるローディングの両方で，ERDRレジスタのMSBビットは“1”にセットされ，ビジィ状態のフラグが立ちます。EEPROMへのアクセスの終わりで，ERDRレジスタのMSBビットは“0”にセットされ，ユーザはMACAR1レジスタからMACAR3レジスタまたはERDRレジスタの［15：0］フィールドからEEPROM内のデータを読み込むことができます。

保守／廃止

10. 2 オペレーション

10. 2. 1 パワー・アップ・ロードにおけるデータの読み込み

リセット解除後にパワー・アップ・ロード処理が始まります。

EEPROM**アドレス**00H**からの値が**A5A5H**と**A5A5H**以外の場合で処理が異なります：**

1. A5A5H
   システム・コントローラは，EEPROM**のアドレス：**01H-06H**のデータを内部レジスタ（**MACAR1, MACAR2, MACAR3**）にセットします。**
2. A5A5H**以外**
   システム・コントローラは，固定データ“ 0000_0000H **”を内部レジスタ（**MACAR1，MACAR2, MACAR3**）にセットします。**

表10ー1 EEPROM**初期データ**

EEPROMアドレス	データ	格納レジスタ
00H	A5A5H	-
01H	MAC1**アドレス・データ**［15：0］	MACAR1［15：0］
02H	MAC1**アドレス・データ**［31：16］	MACAR1［31：16］
03H	MAC1**アドレス・データ**［47：32］	MACAR2［15：0］
04H	MAC2**アドレス・データ**［15：0］	MACAR2［31：16］
05H	MAC2**アドレス・データ**［31：16］	MACAR3［15：0］
06H	MAC2**アドレス・データ**［47：32］	MACAR3［31：16］

保守／廃止

10. 2. 2　EEPROMへのアクセス

EEPROMへのアクセスは，ECCR（EEPROMコマンド制御レジスタ）にコマンドを書き込むことによって始まります。

EEPROMのコマンド（3ビット）とアドレス（6ビット）をECCRの下位9ビットに書き込んでください。

ライト・コマンドとリード・コマンドによって違いがあります。

1. ライト・コマンド

   データをECCRの上位16ビットに書き込みます。

2. リード・コマンド

   データがERDR（EEPROMリード・データ・レジスタ）の下位16ビットにロードされます。

表10－2　EEPROMコマンド一覧

コマンド	ビット8：6	ビット5：0	オペレーション
READ	110	A5：A0	EEPROMのアドレスA5：A0のデータを読み込む。
EWEN	100	11xxxx	削除／ライト・コマンドをイネーブルにする。 ほかのオペレーションを実行する前に，必ずこのコマンドを実行してください。
ERASE	111	A5：A0	EEPROMのアドレスA5：A0のデータを削除する。
WRITE	101	A5：A0	EEPROMのアドレスA5：A0へデータを書き込む。
ERAL	100	10xxxx	EEPROMのすべてのデータを削除する。
WRAL	100	01xxxx	EEPROMのすべてのアドレスに同じ値（DATAフィールドの内容）を書き込む。
EWDS	100	00xxxx	削除／ライト・コマンドをディスエーブルにする。

保守／廃止

10.3 レジスタ

10.3.1 レジスタ・マップ

オフセット・アドレス	レジスタ名	R/W	アクセス	説　明
1000_00D0H	ECCR	W	W/H/B	EEPROMコマンド制御レジスタ
1000_00D4H	ERDR	R	W/H/B	EEPROMリード・データ・レジスタ
1000_00D8H	MACAR1	R	W/H/B	MACアドレス・レジスタ1
1000_00DCH	MACAR2	R	W/H/B	MACアドレス・レジスタ2
1000_00E0H	MACAR3	R	W/H/B	MACアドレス・レジスタ3

10.3.2 ECCR（EEPROMコマンド制御レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31：16	DATA	W	0	シリアルEEPROMへの書き込みデータ。データの読み込みのときは無効となります。
15：9	Reserved	W	0	0固定
8：6	CMD	W	0	シリアルEEPROMコマンド
5：0	ADDRESS	W	0	シリアルEEPROMアドレス

10.3.3 ERDR（EEPROMリード・データ・レジスタ）

ビット	フィールド	R/W	ディフォールト	説　明
31	B	R	1	Micro Wireブロックのオペレーション・ステータス： 1：ビジィ 0：アイドル
30：16	Reserved	R	0	0固定
15：0	READ DATA	R	0	シリアルEEPROMからの読み込みデータ

10.3.4 MACAR1（MACアドレス・レジスタ1）

ビット	フィールド	R/W	ディフォールト	説　明
31：16	SERIAL EEPROM 02H ADDRESS	R	0	シリアルEEPROMアドレス01H, 02Hに格納されたデータ
15：0	SERIAL EEPROM 01H ADDRESS	R	0	

保守／廃止

10. 3. 5 MACAR2（MACアドレス・レジスタ2）

ビット	フィールド	R/W	ディフォールト	説　明
31：16	SERIAL EEPROM 04H ADDRESS	R	0	シリアルEEPROMアドレス03H, 04Hに格納されたデータ
15：0	SERIAL EEPROM 03H ADDRESS	R	0	

10. 3. 6 MACAR3（MACアドレス・レジスタ3）

ビット	フィールド	R/W	ディフォールト	説　明
31：16	SERIAL EEPROM 06H ADDRESS	R	0	シリアルEEPROMアドレス05H, 06Hに格納されたデータ
15：0	SERIAL EEPROM 05H ADDRESS	R	0	

付録A　MIPSIII命令セット

ここでは，CPU 命令の 32 ビット・モードおよび 64 ビット・モードにおける各機能を，アルファベット順に説明します。

A.1　命令表記法

ここでは，命令形式におけるすべてのサブフィールド（rs，rt，immediateなど）は小文字で示します。分かりやすくするために，特定の命令ではサブフィールドに別名を使用することがあります。たとえば，ロード命令やストア命令では，rsの代わりにbaseを使用します。この別名もサブフィールドを指しているので，常に小文字で記述しています。

すべてのニモニックの実際のコードや機能フィールドのコードは，**A.6 CPU命令オペコード符号**に示します。

命令の説明の中で，オペレーションの項目では高級言語による表現法を使用し，各命令によって実行される動作について記載しています。VR4120Aコアは，32ビット・モードおよび64ビット・モードのどちらでも実行できます。各モードでの動作の違いは，オペレーションの項目で示します。表記中の特殊な記号の意味を，表A-1に示します。

表 A-1　CPU 命令演算表記法

記　号	意　　味
←	代入
‖	ビット・ストリング連結
x^y	ビット・ストリング x を y ビットのストリングで反復します。x は常に1ビットの値です。
$x_{y..z}$	ビット・ストリング x についてビット y から z までの選択。 リトル・エンディアン・ビット表記法が常に使用されます。y が z 未満の場合には，この表現式は空（0の長さ）のビット・ストリングになります。
＋	2の補数加算または浮動小数点加算
－	2の補数減算または浮動小数点減算
＊	2の補数乗算または浮動小数点乗算
div	2の補数整数除算
mod	2の補数剰余
／	浮動小数点除算
＜	2の補数が小さいかの比較
and	ビットごとの論理積
or	ビットごとの論理和
xor	ビットごとの排他的論理和
nor	ビットごとの否定論理和
GPR[x]	汎用レジスタ x。GPR[0]の内容は常に0です。GPR[0]の内容は変更できません。
CPR[z ， x]	コプロセッサ・ユニット z，汎用レジスタ x
CCR[z ， x]	コプロセッサ・ユニット z，制御レジスタ x
COC[z]	コプロセッサ・ユニット z，制御信号
BigEndianMem	リセット時に決定されるエンディアン・モード（0→リトル，1→ビッグ） メモリ・インタフェースのエンディアンと（**表A-2 ロード／ストア命令の共通関数**参照），カーネル・モード，スーパバイザ・モードによるエンディアンを指定します。
ReverseEndian	ロード命令とストア命令のエンディアンを反転させる信号 これはユーザ・モードにのみ使用でき，ステータス・レジスタのREビットをセットすることにより使用します。したがって，この変数は，ユーザ・モードにおいてREビットがセットされたときにのみ1がセットされます。ただし，VR4120Aコアではリバース･エンディアンをサポートしていませんので，この値は常に0となります。
BigEndianCPU	ロード命令，ストア命令によるエンディアン（0→リトル，1→ビッグ） ユーザ・モードにおけるエンディアンは，REビットをセットすることにより反転します。したがって，この変数はBigEndianMem XOR ReverseEndianとして計算されます。
T＋i：	演算間の時間的ステップを示します。時間的ステップ中でのそれぞれのステートメントは，シーケンシャルに実行されるように定義されています（条件分岐やループによって命令の実行順序が変更されることがあります）。 T＋iで記されている演算は，命令実行開始から i 命令サイクル目に実行されます。したがって j で始まる命令は，T＋i：と書かれた演算をi＋jサイクル目に実行します。同時に実行される命令または演算同士の実行順序は定義されていません。

(1) 命令表記例

命令表記の例を次に示します。

例1. GPR[rt] ← immediate ‖ 0^{16}

16個のゼロ・ビットをimmediate（通常16ビット）の下位に連結し，32ビットのビット列をCPUの汎用レジスタrtに代入します。

2. $(immediate_{15})^{16}$ ‖ $immediate_{15..0}$

immediateのビット15（符号ビット）を16ビット分拡張し，その結果をimmediateのビット15からビット0と連結して，32ビットの符号拡張した値を生成します。

A.2　ロード／ストア命令

VR4120Aコアでは，ロード命令の直後の命令が，ロードされたレジスタの内容を使用できます。このような場合，1PCycleだけハードウエアがインタロックします。したがって，機能コードとしては必要ないのですが，性能向上のためにはロード遅延スロットのスケジューリングが必要です。

仮想アドレスと物理メモリの取り扱いを簡略化するために，ロード命令やストア命令の説明で，次に示す関数を使用しています。

表 A-2　ロード／ストア命令の共通関数

関　数	意　味
AddressTranslation	仮想アドレスから物理アドレスを探すためにTLBを使用します。要求どおりの変換内容がTLBになかった場合，この関数がフェイルし，例外が発生します。
LoadMemory	指定した物理アドレスに入っている指定したデータ長の内容を探すためにキャッシュとメイン・メモリを使用します。アドレスの下位3ビットとアクセス・タイプ・フィールドが，データ・ワード内のデータ位置を決定します。指定したデータ長がワード未満の場合，エンディアン・モードとリバース・エンディアン・モードも考慮してデータ位置を決め，その内容をロードします。キャッシュがイネーブルになると，キャッシュにロードされます。
StoreMemory	指定したデータ長の内容を，指定した物理アドレスにストアするために，キャッシュ，ライト・バッファ，およびメイン・メモリを使用します。アドレスの下位3ビットとアクセス・タイプ・フィールドが，データ・ワード内のデータ位置を決定します。指定したデータ長がワード未満の場合，エンディアン・モードとリバース・エンディアン・モードも考慮してデータ位置を決め，その内容をストアします。

アクセス・タイプ・フィールドは，ロード／ストアすべきデータのサイズを示します。アクセス・タイプやバイトの並び順（エンディアン）に関係なく，アドレスは，アクセスされたフィールド内で最も小さいバイト・アドレスを持つバイトを指定します。リトル・エンディアン・オーダのみサポートするため，VR4120AのCPUでは右端のバイトになります。

表 A-3　ロード／ストア命令でのアクセス・タイプの指定

アクセス・タイプ	意　味
DOUBLEWORD	8バイト（64ビット）
SEPTIBYTE	7バイト（56ビット）
SEXTIBYTE	6バイト（48ビット）
QUINTIBYTE	5バイト（40ビット）
WORD	4バイト（32ビット）
TRIPLEBYTE	3バイト（24ビット）
HALFWORD	2バイト（16ビット）
BYTE	1バイト（8ビット）

アクセスしたダブル・ワード内のバイト並びは，アクセス・タイプとアドレスの下位3ビットによって決まります。

A.3　ジャンプ／ブランチ命令

ジャンプ命令とブランチ命令を実行すると，構造的に遅延が生じ，その遅延はちょうど1命令分です。その理由は，ジャンプ／ブランチ命令がメモリからフェッチされている間に，その次に実行する命令（遅延スロットにある命令）が実行されているからです。遅延スロットではジャンプ／ブランチ命令を使用できません。使用した場合エラーは検出されず，オペレーションの結果は不定になります。

命令が遅延スロットにあるときに例外や割り込みが発生し，その命令を終了できなかった場合，ハードウエアで1つ前のジャンプ／ブランチ命令の仮想アドレスをEPCレジスタにセットします。例外や割り込みの処理が終了し，元のプログラムに復帰したとき，ジャンプ／ブランチ命令と，遅延スロットにあった命令の両方を再び実行します。

このように，例外や割り込みの処理のあとにジャンプ／ブランチ命令を再実行するため，ジャンプ・アンド・リンク／ブランチ・アンド・リンク命令ではレジスタ31（リンク・アドレスを保存するレジスタ）をソース・レジスタとして使用しないでください。

命令はワード・アラインされている必要があるので，JR命令やJALR命令では，アドレスの下位2ビット（16ビット・モードでは下位1ビット）が0であるレジスタを使用してください。下位2（1）ビットが0でない場合，ジャンプ先の命令がフェッチされたときにアドレス例外が発生します。

保守／廃止

A.4　システム制御コプロセッサ（CP0）命令

CPUに搭載されているCP0に関わるオペレーションには，制限事項がいくつかあります。MIPSアーキテクチャでは，コプロセッサに対しデータ転送を行うロード／ストア命令や，制御コードをコプロセッサとやり取りする命令は一般的には認められていますが，CP0が例外処理やメモリ管理の責任を負っているため，いくらか保護された状態を与えられています。CP0レジスタに書き込み／読み出しを行うためには，コプロセッサ転送命令が唯一有効な方法です。

CP0命令の中には，TLBエントリに直接リード，ライト，検査するように，またはユーザ・モードか割り込み許可状態に復帰する準備のためオペレーティング・モードを変更するように定義されているものもあります。

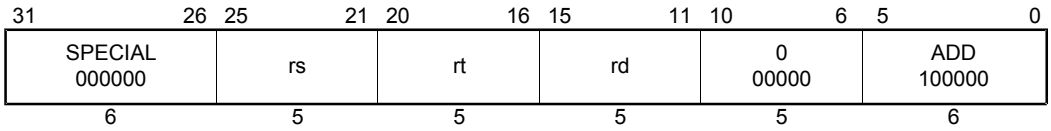
A.5　CPU 命令

ここでは，CPU命令の32ビット・アドレス・モードおよび64ビット・アドレス・モードにおける各機能について詳しく説明します。

それぞれの命令の実行によって起こり得る例外は，各命令の説明の最後に記載します。例外と例外処理の詳細については，**2.5　例外処理**を参照してください。

ADD

Add



命　令　形　式：

ADD　rd，rs，rt

説　　明：

汎用レジスタrs**の内容と汎用レジスタ**rt**の内容を加算し，汎用レジスタ**rd**に格納します。64ビット・モードでは，オペランドは32ビットの値を符号拡張したものでなければなりません。**

ビット30からの桁上がりとビット31からの桁上がりが異なる（2の補数オーバフロー）場合，整数オーバフロー例外が発生します。整数オーバフロー例外が発生すると，デスティネーション・レジスタrd**の内容は変更しません。**

オペレーション：

```
32　T：　GPR[rd] ← GPR[rs]＋GPR[rt]

64　T：　temp ← GPR[rs]＋GPR[rt]
         GPR[rd] ← (temp31)^32 ‖ temp31..0
```

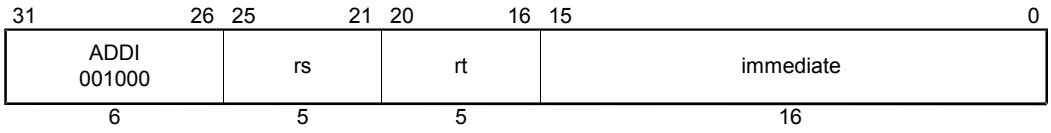
例　　外：

整数オーバフロー例外

保守／廃止

ADDI

Add Immediate



命　令　形　式：

ADDI　rt，rs，immediate

説　　明：

符号拡張した16ビットimmediateと汎用レジスタrsの内容を加算し，汎用レジスタrtに格納します。64ビット・モードでは，オペランドは32ビットの値を符号拡張したものでなければなりません。

ビット30からの桁上がりとビット31からの桁上がりが異なる（2の補数オーバフロー）場合，整数オーバフロー例外が発生します。整数オーバフロー例外が発生すると，デスティネーション・レジスタrtの内容は変更しません。

オペレーション：

```
32  T：  GPR[rt] ← GPR[rs]＋(immediate15)^16 ‖ immediate15..0

64  T：  temp ← GPR[rs]＋(immediate15)^48 ‖ immediate15..0
         GPR[rt] ← (temp31)^32 ‖ temp31..0
```

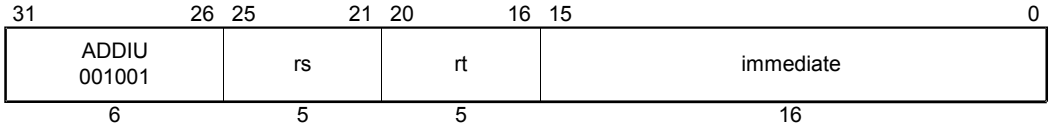
例　　外：

整数オーバフロー例外

保守／廃止

ADDIU

Add Immediate Unsigned



命　令　形　式：

ADDIU　rt，rs，immediate

説　　明：

符号拡張した16ビットimmediateと汎用レジスタrsの内容を加算し，汎用レジスタrtに格納します。どんな状況でも整数オーバフロー例外は発生しません。64ビット・モードでは，オペランドは32ビットの値を符号拡張したものでなければなりません。

ADDI命令との違いは，ADDIU命令が整数オーバフロー例外を発生しないという点だけです。

オペレーション：

```
32　T：　GPR[rt] ← GPR[rs]＋(immediate15)^16 ‖ immediate15..0

64　T：　temp ← GPR[rs]＋(immediate15)^48 ‖ immediate15..0
         GPR[rt] ← (temp31)^32 ‖ temp31..0
```

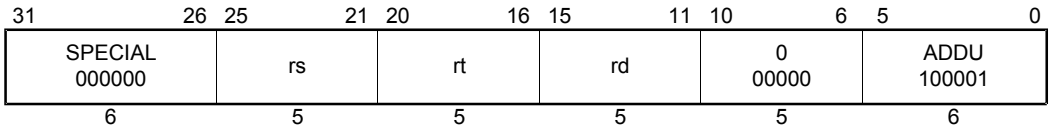
例　　外：

なし

保守／廃止

ADDU

Add Unsigned



命　令　形　式：

ADDU　rd，rs，rt

説　　明：

汎用レジスタrs**の内容と汎用レジスタ**rt**の内容を加算し，汎用レジスタ**rd**に格納します。どんな状況でも整数オーバフロー例外は発生しません。**64**ビット・モードでは，オペランドは**32**ビットの値を符号拡張したものでなければなりません。**

ADD**命令との違いは，**ADDU**命令が整数オーバフロー例外を発生しないという点だけです。**

オペレーション：

```
32  T：  GPR[rd] ← GPR[rs]＋GPR[rt]

64  T：  temp ← GPR[rs]＋GPR[rt]
         GPR[rd] ← (temp31)^32 ‖ temp31..0
```

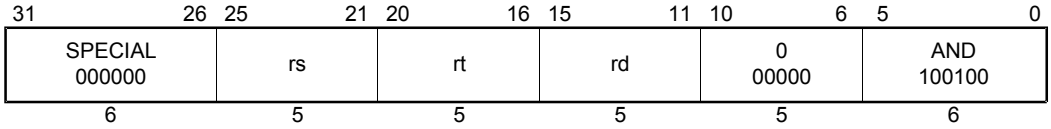
例　　外：

なし

保守／廃止

AND

And



命　令　形　式：

AND　rd，rs，rt

説　　明：

汎用レジスタrsの内容と汎用レジスタrtの内容の，ビットごとの論理積演算を行います。結果は汎用レジスタrdに格納します。

オペレーション：

```
32　T：　GPR[rd] ← GPR[rs] and GPR[rt]

64　T：　GPR[rd] ← GPR[rs] and GPR[rt]
```

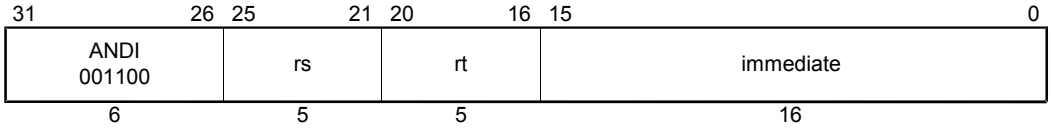
例　　外：

なし

保守／廃止

ANDI

And Immediate



命　令　形　式：

ANDI　rt，rs，immediate

説　　明：

ゼロ拡張した16ビットのimmediateと汎用レジスタrsの内容の，ビットごとの論理積演算を行います。結果は汎用レジスタrtに格納します。

オペレーション：

```
32　T：　GPR[rt] ← 0^16 ‖ （immediate and GPR[rs]15..0）

64　T：　GPR[rt] ← 0^48 ‖ （immediate and GPR[rs]15..0）
```

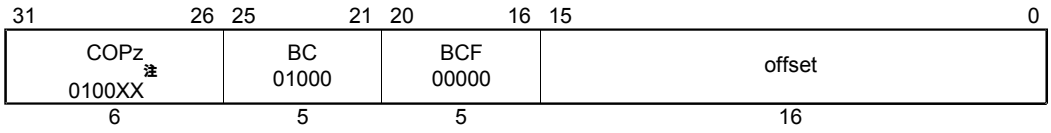
例　　外：

なし

保守／廃止

BC0F

Branch On Coprocessor 0 False



命　令　形　式：

BC0F　offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。直前の命令の実行中にサンプリングされたCP0の条件信号（CpCond）が偽の場合，1命令の遅延付きで分岐アドレスに分岐します。

条件信号は直前の命令の実行中にサンプリングされるので，条件信号を変更するコプロセッサ命令とこの命令の間には，少なくとも1命令は置いてください。

オペレーション：

```
32  T－1：  condition ← not SR18
    T：     target ← (offset15)^14 || offset || 0^2
    T＋1：  if condition then
              PC ← PC＋target
            endif

64  T－1：  condition ← not SR18
    T：     target ← (offset15)^46 || offset || 0^2
    T＋1：  if condition then
              PC ← PC＋target
            endif
```

例　　外：

コプロセッサ使用不可例外

注　次の「オペコード符号表」，またはA.6　CPU命令オペコード符号参照。

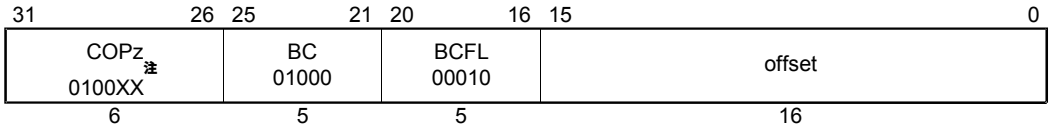
オペコード符号表：

保守／廃止

BC0FL

Branch On Coprocessor 0 False Likely

（1/2）



命　令　形　式：

BC0FL　offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。直前の命令の実行中にサンプリングされたCP0の条件信号（CpCond）が偽の場合，1命令の遅延付きで分岐アドレスに分岐します。

分岐条件が成立しない場合，分岐遅延スロット内の命令は破棄されます。

条件信号は直前の命令の実行中にサンプリングされるので，条件信号を変更するコプロセッサ命令とこの命令の間には，少なくとも1命令は置いてください。

オペレーション：

```
32   T－1：   condition ← not SR18
     T：      target ← (offset15)^14 || offset || 0^2
     T＋1：   if condition then
                PC ← PC＋target
              else
                NullifyCurrentInstruction
              endif

64   T－1：   condition ← not SR18
     T：      target ← (offset15)^46 || offset || 0^2
     T＋1：   if condition then
                PC ← PC＋target
              else
                NullifyCurrentInstruction
              endif
```

例　　外：

コプロセッサ使用不可例外

注　次の「オペコード符号表」，またはA.6　CPU命令オペコード符号参照。

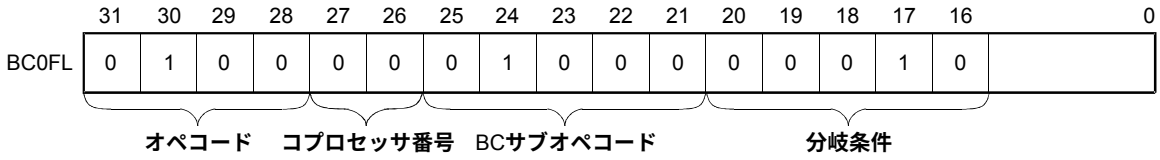
保守／廃止

BC0FL

Branch On Coprocessor 0 False Likely

(2/2)

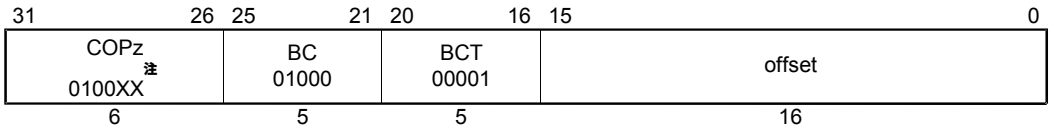
オペコード符号表：



保守／廃止

BC0T

Branch On Coprocessor 0 True



命　令　形　式：

BC0T　offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。直前の命令の実行中にサンプリングされたCP0の条件信号（CpCond）が真の場合，1命令の遅延付きで分岐アドレスに分岐します。

条件信号は直前の命令の実行中にサンプリングされるので，条件信号を変更するコプロセッサ命令とこの命令の間には，少なくとも1命令は置いてください。

オペレーション：

```
32  T－1：  condition ← SR18
    T：     target ← (offset15)^14 || offset || 0^2
    T＋1：  if condition then
              PC ← PC＋target
            endif

64  T－1：  condition ← SR18
    T：     target ← (offset15)^46 || offset || 0^2
    T＋1：  if condition then
              PC ← PC＋target
            endif
```

例　　外：

コプロセッサ使用不可例外

注　次の「オペコード符号表」，またはA.6　CPU命令オペコード符号参照。

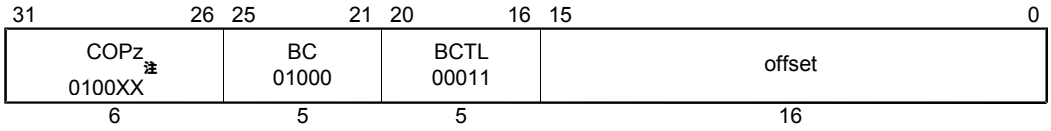
オペコード符号表：

保守／廃止

BC0TL

Branch On Coprocessor 0 True Likely

(1/2)



命　令　形　式：

BC0TL　offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。直前の命令の実行中にサンプリングされたCP0の条件信号（CpCond）が真の場合，1命令の遅延付きで分岐アドレスに分岐します。

分岐条件が成立しない場合，分岐遅延スロット内の命令は破棄されます。

条件信号は直前の命令の実行中にサンプリングされるので，条件信号を変更するコプロセッサ命令とこの命令の間には，少なくとも1命令は置いてください。

オペレーション：

```
32  T－1：   condition ← SR18
    T：      target ← (offset15)^14 || offset || 0^2
    T＋1：   if condition then
               PC ← PC＋target
             else
               NullifyCurrentInstruction
             endif

64  T－1：   condition ← SR18
    T：      target ← (offset15)^46 || offset || 0^2
    T＋1：   if condition then
               PC ← PC＋target
             else
               NullifyCurrentInstruction
             endif
```

例　　外：

コプロセッサ使用不可例外

注　次ページの「オペコード符号表」，またはA.6　CPU命令オペコード符号参照。

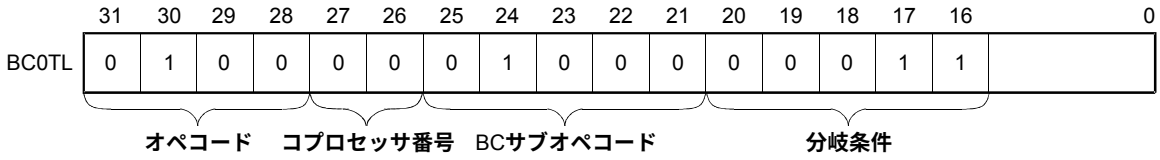
保守／廃止

BC0TL

Branch On Coprocessor 0 True Likely

(2/2)

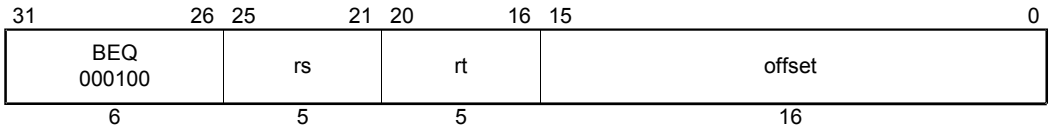
オペコード符号表：



保守／廃止

BEQ

Branch On Equal



命　令　形　式：

BEQ　rs，rt，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。汎用レジスタrsの内容と汎用レジスタrtの内容を比較し，等しい場合は1命令の遅延付きで分岐アドレスに分岐します。

オペレーション：

```
32   T：     target ← (offset15)^14 || offset || 0^2
             condition ← (GPR[rs] = GPR[rt])
        T+1：  if condition then
                 PC ← PC+target
               endif

64   T：     target ← (offset15)^46 || offset || 0^2
             condition ← (GPR[rs] = GPR[rt])
        T+1：  if condition then
                 PC ← PC+target
               endif
```

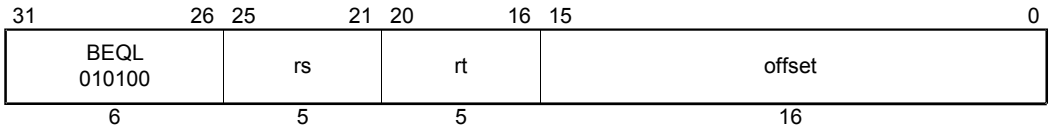
例　　外：

なし

保守／廃止

BEQL

Branch On Equal Likely



命　令　形　式：

BEQL　rs，rt，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。汎用レジスタrsと汎用レジスタrtの内容を比較し，等しい場合は1命令の遅延付きで分岐アドレスに分岐します。

分岐条件が成立しない場合，分岐遅延スロット内の命令を破棄します。

オペレーション：

```
32   T：     target ← (offset15)^14 || offset || 0^2
             condition ← (GPR[rs] = GPR[rt])
        T+1：  if condition then
                 PC ← PC+target
               else
                 NullifyCurrentInstruction
               endif

64   T：     target ← (offset15)^46 || offset || 0^2
             condition ← (GPR[rs] = GPR[rt])
        T+1：  if condition then
                 PC ← PC+target
               else
                 NullifyCurrentInstruction
               endif
```

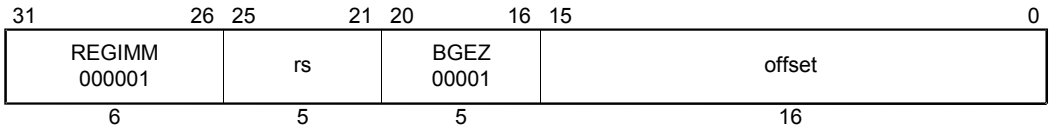
例　　外：

なし

保守／廃止

BGEZ

Branch On Greater Than Or Equal To Zero



命　令　形　式：

BGEZ　rs，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。汎用レジスタrsの内容を0と比較し，0と等しいか0より大きい場合，1命令の遅延付きで分岐アドレスに分岐します。

オペレーション：

```
32   T：     target ← (offset15)^14 || offset || 0^2
             condition ← (GPR[rs]31 = 0)
     T+1：   if condition then
               PC ← PC+target
             endif

64   T：     target ← (offset15)^46 || offset || 0^2
             condition ← (GPR[rs]63 = 0)
     T+1：   if condition then
               PC ← PC+target
             endif
```

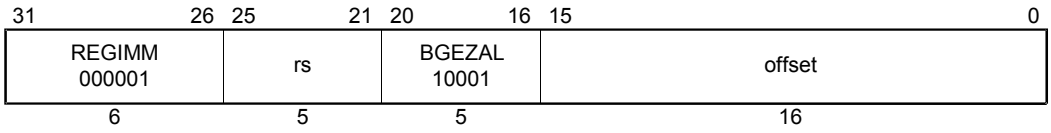
例　　外：

なし

保守／廃止

BGEZAL

Branch On Greater Than Or Equal To Zero And Link



命　令　形　式：

BGEZAL　rs，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。無条件に遅延スロットの次の命令のアドレスがリンク・レジスタr31に格納されます。汎用レジスタrsの内容を0と比較し，0と等しいか0より大きい場合，1命令の遅延付きで分岐アドレスに分岐します。

通常，汎用レジスタrsとして汎用レジスタr31を指定するべきではありません。その理由は，汎用レジスタrsとして汎用レジスタr31を指定すると，リンク・アドレスのストアによってrsの内容が破壊され，再実行できない場合があるからです。しかし，このような命令を実行しても例外とはなりません。

オペレーション：

```
32   T：     target ← (offset15)^14 || offset || 0^2
             condition ← (GPR[rs]31 = 0)
             GPR[31] ← PC+8
     T+1：   if condition then
               PC ← PC+target
             endif

64   T：     target ← (offset15)^46 || offset || 0^2
             condition ← (GPR[rs]63 = 0)
             GPR[31] ← PC+8
     T+1：   if condition then
               PC ← PC+target
             endif
```

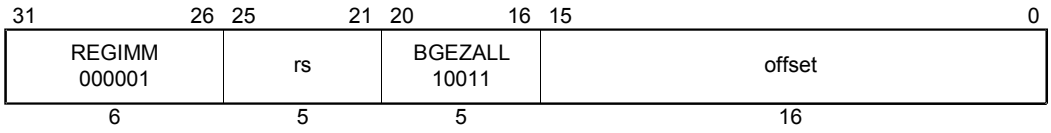
例　　外：

なし

保守／廃止

BGEZALL

Branch On Greater Than Or Equal To Zero And Link Likely



命　令　形　式：

BGEZALL　rs，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。無条件に遅延スロットの次の命令のアドレスがリンク・レジスタr31に格納されます。汎用レジスタrsの内容を0と比較し，0と等しいか0より大きい場合，1命令の遅延付きで分岐アドレスに分岐します。

分岐条件が成立しない場合，分岐遅延スロット内の命令を破棄します。

通常，汎用レジスタrsとして汎用レジスタr31を指定するべきではありません。その理由は，汎用レジスタrsとして汎用レジスタr31を指定すると，リンク・アドレスのストアによってrsの内容が破壊され，再実行できない場合があるからです。しかし，このような命令を実行しても例外とはなりません。

オペレーション：

```
32   T：     target ← (offset15)^14 || offset || 0^2
             condition ← (GPR[rs]31 = 0)
             GPR[31] ← PC+8
     T+1：   if condition then
               PC ← PC+target
             else
               NullifyCurrentInstruction
             endif

64   T：     target ← (offset15)^46 || offset || 0^2
             condition ← (GPR[rs]63 = 0)
             GPR[31] ← PC+8
     T+1：   if condition then
               PC ← PC+target
             else
               NullifyCurrentInstruction
             endif
```

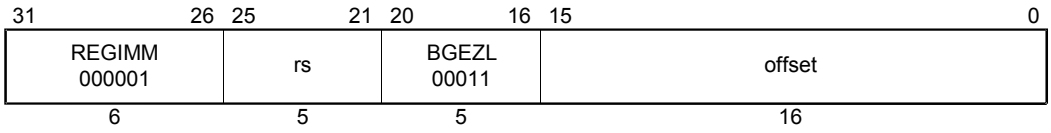
例　　外：

なし

保守／廃止

BGEZL

Branch On Greater Than Or Equal To Zero Likely



命　令　形　式：

BGEZL　rs，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。汎用レジスタrsの内容を0と比較し，0と等しいか0より大きい場合，1命令の遅延付きで分岐アドレスに分岐します。

分岐条件が成立しない場合，分岐遅延スロット内の命令を破棄します。

オペレーション：

```
32   T：      target ← (offset15)^14 || offset || 0^2
              condition ← (GPR[rs]31 = 0)
         T+1：  if condition then
                  PC ← PC+target
                else
                  NullifyCurrentInstruction
                endif

64   T：      target ← (offset15)^46 || offset || 0^2
              condition ← (GPR[rs]63 = 0)
         T+1：  if condition then
                  PC ← PC+target
                else
                  NullifyCurrentInstruction
                endif
```

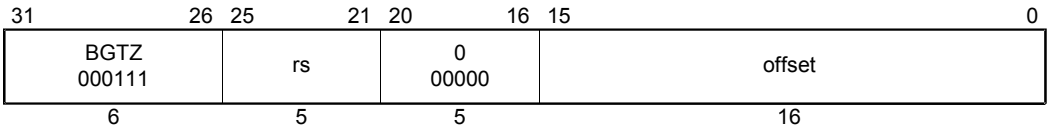
例　　外：

なし

保守／廃止

BGTZ

Branch On Greater Than Zero



命　令　形　式：

BGTZ　rs，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。汎用レジスタrsの内容を0と比較し，0より大きい場合は1命令の遅延付きで分岐アドレスに分岐します。

オペレーション：

```
32   T：     target ← (offset15)^14 || offset || 0^2
             condition ← (GPR[rs]31 = 0) and (GPR[rs] ≠ 0^32)
        T+1：  if condition then
                 PC ← PC+target
               endif

64   T：     target ← (offset15)^46 || offset || 0^2
             condition ← (GPR[rs]63 = 0) and (GPR[rs] ≠ 0^64)
        T+1：  if condition then
                 PC ← PC+target
               endif
```

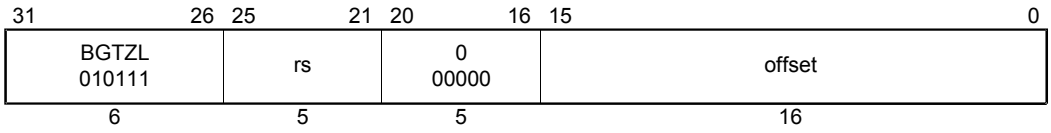
例　　外：

なし

保守／廃止

BGTZL

Branch On Greater Than Zero Likely



命　令　形　式：

BGTZL　rs，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。汎用レジスタrsの内容を0と比較し，0より大きい場合は1命令の遅延付きで分岐アドレスに分岐します。

分岐条件が成立しない場合，分岐遅延スロット内の命令を破棄します。

オペレーション：

```
32   T：      target ← (offset15)^14 || offset || 0^2
              condition ← (GPR[rs]31 = 0) and (GPR[rs] ≠ 0^32)
     T+1：    if condition then
                PC ← PC+target
              else
                NullifyCurrentInstruction
              endif

64   T：      target ← (offset15)^46 || offset || 0^2
              condition ← (GPR[rs]63 = 0) and (GPR[rs] ≠ 0^64)
     T+1：    if condition then
                PC ← PC+target
              else
                NullifyCurrentInstruction
              endif
```

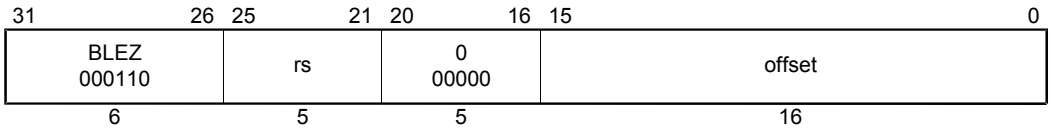
例　　外：

なし

保守／廃止

BLEZ

Branch On Less Than Or Equal To Zero



命　令　形　式：

BLEZ　rs，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。汎用レジスタrsの内容を0と比較し，0と等しいか0より小さい場合，1命令の遅延付きで分岐アドレスに分岐します。

オペレーション：

```
32   T：     target ← (offset15)^14 || offset || 0^2
             condition ← (GPR[rs]31 = 1) or (GPR[rs] = 0^32)
     T+1：   if condition then
               PC ← PC+target
             endif

64   T：     target ← (offset15)^46 || offset || 0^2
             condition ← (GPR[rs]63 = 1) or (GPR[rs] = 0^64)
     T+1：   if condition then
               PC ← PC+target
             endif
```

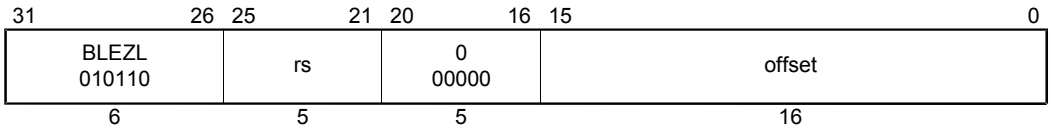
例　　外：

なし

保守／廃止

BLEZL

Branch On Less Than Or Equal To Zero Likely



命　令　形　式：

BLEZL　rs，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。汎用レジスタrsの内容を0と比較し，0と等しいか0より小さい場合，1命令の遅延付きで分岐アドレスに分岐します。

分岐条件が成立しない場合，分岐遅延スロット内の命令を破棄します。

オペレーション：

```
32   T：     target ← (offset15)^14 || offset || 0^2
             condition ← (GPR[rs]31 = 1) or (GPR[rs] = 0^32)
        T+1：  if condition then
                 PC ← PC+target
               else
                 NullifyCurrentInstruction
               endif

64   T：     target ← (offset15)^46 || offset || 0^2
             condition ← (GPR[rs]63 = 1) or (GPR[rs] = 0^64)
        T+1：  if condition then
                 PC ← PC+target
               else
                 NullifyCurrentInstruction
               endif
```

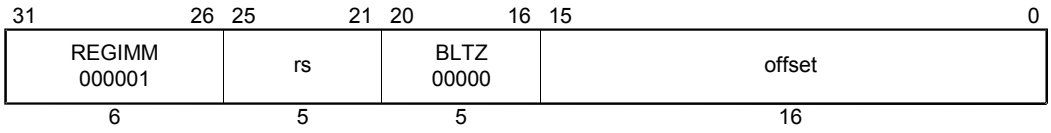
例　　外：

なし

保守／廃止

BLTZ

Branch On Less Than Zero



命　令　形　式：

BLTZ　rs，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。汎用レジスタrsの内容を0と比較し，0より小さい場合は1命令の遅延付きで分岐アドレスに分岐します。

オペレーション：

```
32   T：     target ← (offset15)^14 || offset || 0^2
             condition ← (GPR[rs]31 = 1)
        T+1：  if condition then
                 PC ← PC+target
               endif

64   T：     target ← (offset15)^46 || offset || 0^2
             condition ← (GPR[rs]63 = 1)
        T+1：  if condition then
                 PC ← PC+target
               endif
```

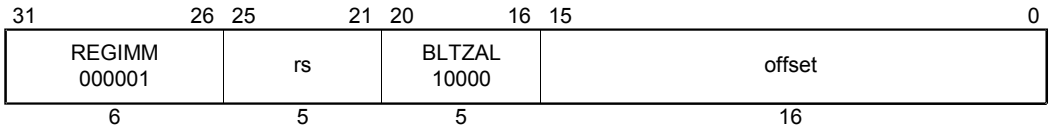
例　　外：

なし

保守／廃止

BLTZAL

Branch On Less Than Zero And Link



命　令　形　式：

BLTZAL　rs，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。無条件に遅延スロットの次の命令のアドレスがリンク・レジスタr31に格納されます。汎用レジスタrsの内容を0と比較し，0より小さい場合は1命令の遅延付きで分岐アドレスに分岐します。

通常，汎用レジスタrsとして汎用レジスタr31を指定するべきではありません。その理由は，汎用レジスタrsとして汎用レジスタr31を指定すると，リンク・アドレスのストアによってrsの内容が破壊され，再実行できない場合があるからです。しかし，この命令を実行しても例外とはなりません。

オペレーション：

```
32  T：     target ← (offset15)^14 || offset || 0^2
            condition ← (GPR[rs]31 = 1)
            GPR[31] ← PC+8
    T+1：   if condition then
              PC ← PC+target
            endif

64  T：     target ← (offset15)^46 || offset || 0^2
            condition ← (GPR[rs]63 = 1)
            GPR[31] ← PC+8
    T+1：   if condition then
              PC ← PC+target
            endif
```

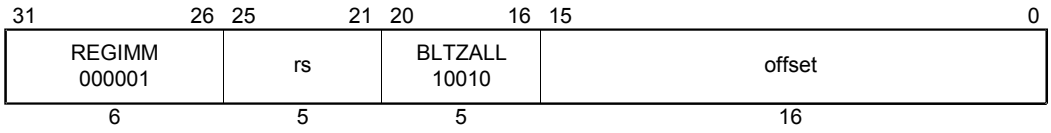
例　　外：

なし

保守／廃止

BLTZALL

Branch On Less Than Zero And Link Likely



命　令　形　式：

BLTZALL　rs，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。無条件に遅延スロットの次の命令がリンク・レジスタr31に格納されます。汎用レジスタrsの内容を0と比較し，0より小さい場合は1命令の遅延付きで分岐アドレスに分岐します。

分岐条件が成立しない場合，分岐遅延スロット内の命令を破棄します。

通常，汎用レジスタrsとして汎用レジスタr31を指定するべきではありません。その理由は，汎用レジスタrsとして汎用レジスタr31を指定すると，リンク・アドレスのストアによってrsの内容が破壊され，再実行できない場合があるからです。しかし，この命令を実行しても例外とはなりません。

オペレーション：

```
32   T：     target ← (offset15)^14 || offset || 0^2
             condition ← (GPR[rs]31 = 1)
             GPR[31] ← PC+8
     T+1：   if condition then
               PC ← PC+target
             else
               NullifyCurrentInstruction
             endif

64   T：     target ← (offset15)^46 || offset || 0^2
             condition ← (GPR[rs]63 = 1)
             GPR[31] ← PC+8
     T+1：   if condition then
               PC ← PC+target
             else
               NullifyCurrentInstruction
             endif
```

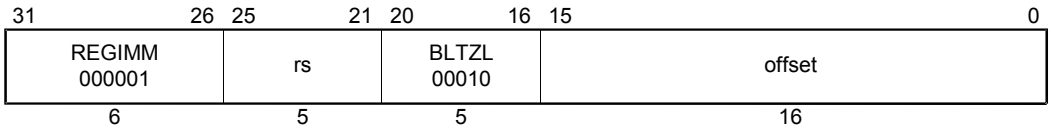
例　　外：

なし

保守／廃止

BLTZL

Branch On Less Than Zero Likely



命　令　形　式：

BLTZL　rs，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。汎用レジスタrsの内容を0と比較し，0より小さい場合は1命令の遅延付きで分岐アドレスに分岐します。

分岐条件が成立しない場合，分岐遅延スロット内の命令を破棄します。

オペレーション：

```
32   T：      target ← (offset15)^14 || offset || 0^2
              condition ← (GPR[rs]31 = 1)
         T+1：  if condition then
                  PC ← PC+target
                else
                  NullifyCurrentInstruction
                endif

64   T：      target ← (offset15)^46 || offset || 0^2
              condition ← (GPR[rs]63 = 1)
         T+1：  if condition then
                  PC ← PC+target
                else
                  NullifyCurrentInstruction
                endif
```

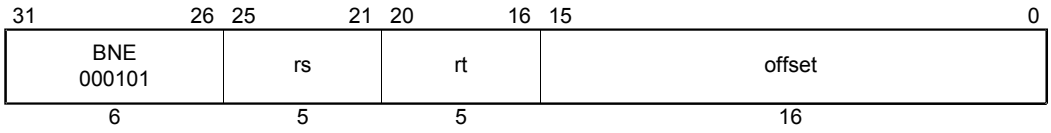
例　　外：

なし

保守／廃止

BNE

Branch On Not Equal



命　令　形　式：

BNE　rs，rt，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。汎用レジスタrsの内容と汎用レジスタrtの内容を比較し，等しくない場合は1命令の遅延付きで分岐アドレスに分岐します。

オペレーション：

```
32   T：     target ← (offset15)^14 || offset || 0^2
             condition ← (GPR[rs] ≠ GPR[rt])
        T+1：  if condition then
                 PC ← PC+target
               endif

64   T：     target ← (offset15)^46 || offset || 0^2
             condition ← (GPR[rs] ≠ GPR[rt])
        T+1：  if condition then
                 PC ← PC+target
               endif
```

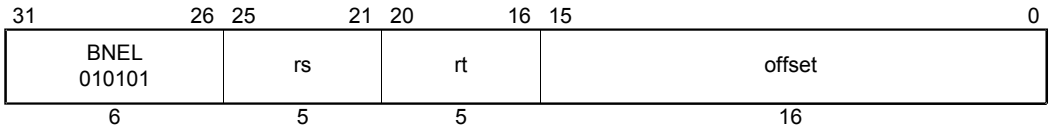
例　　外：

なし

保守／廃止

BNEL

Branch On Not Equal Likely



命　令　形　式：

BNEL　rs，rt，offset

説　　明：

16ビットのoffsetを2ビット左にシフトしてこれを符号拡張したものと，遅延スロット内の命令のアドレスを加算し，分岐アドレスを計算します。汎用レジスタrsと汎用レジスタrtの内容を比較し，等しくない場合は1命令の遅延付きで分岐アドレスに分岐します。

分岐条件が成立しない場合，分岐遅延スロット内の命令を破棄します。

オペレーション：

```
32   T：     target ← (offset15)^14 || offset || 0^2
             condition ← (GPR[rs] ≠ GPR[rt])
        T+1：  if condition then
                 PC ← PC+target
               else
                 NullifyCurrentInstruction
               endif

64   T：     target ← (offset15)^46 || offset || 0^2
             condition ← (GPR[rs] ≠ GPR[rt])
        T+1：  if condition then
                 PC ← PC+target
               else
                 NullifyCurrentInstruction
               endif
```

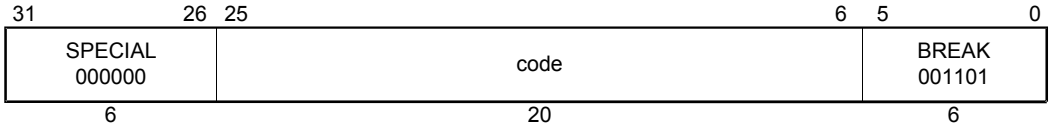
例　　外：

なし

保守／廃止

BREAK

Breakpoint



命　令　形　式：

BREAK

説　　明：

この命令を実行するとブレークポイント例外が発生し，制御を例外ハンドラに渡します。

code領域を使用することによって，例外ハンドラにパラメータを送ることができます。例外ハンドラがこのパラメータを使用する場合には，命令を含むメモリ・ワードの内容をデータとしてロードする必要があります。

オペレーション：

```
32, 64　T：　BreakpointException
```

例　　外：

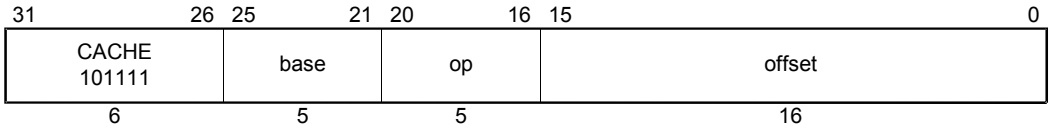
ブレークポイント例外

保守／廃止

CACHE

Cache Operation

（1/4）



命　令　形　式：

CACHE　op，offset（base）

説　　明：

16ビット・オフセットを符号拡張して汎用レジスタbaseの内容に加算し，仮想アドレスを生成します。仮想アドレスはTLBを使用して物理アドレスに変換されます。5ビットのサブオペコードopは，指定されたアドレスに対するキャッシュ・オペレーションの内容を指定します。

ユーザまたはスーパバイザ・モードにおいてステータス・レジスタのCP0許可ビットCU_0がクリアされている場合，CP0は使用不可となり，この命令を実行するとコプロセッサ使用不可例外が発生します。次の一覧表にないキャッシュ・オペレーションとの組み合わせや，V_R4120Aコアにはない二次キャッシュに対する命令実行は不定です。非キャッシュ領域に対するこの命令の実行も不定です。

Indexオペレーションでは，仮想アドレスの一部を使用してキャッシュ・ブロックを指定します。たとえば，容量が$2^{CACHEBITS}$で，タグごとに$2^{LINEBITS}$を持つキャッシュの場合，$vAddr_{CACHEBITS...LINEBITS}$がそのブロックを示します。

Index_Load_Tagオペレーションでは，さらに$vAddr_{LINEBITS..3}$を使用して，パリティの読み込みを行うダブル・ワードを指定します。ステータス・レジスタのCEビットがセットされた場合，Fillキャッシュ・オペレーションでは，パリティ・エラー・レジスタを使用してパリティ値をキャッシュにストアします。

Hitオペレーションでは，通常のデータ参照と同じようにキャッシュをアクセスし，指定された物理アドレスのデータがキャッシュに存在している場合（ヒット）のみ，指定のキャッシュ・オペレーションを実行します。データがキャッシュにない場合（ミス）には，キャッシュ・オペレーションを実行しません。

保守／廃止

CACHE

Cache Operation

(2/4)

キャッシュからのライトバックは，メイン・メモリに対して行います。

ライトバックされるメイン・メモリのアドレスは，TLBを使用して変換した物理アドレスではなく，キャッシュ・タグに入っていたアドレスです。

TLB不一致例外とTLB無効例外は，どのキャッシュ・オペレーションでも発生する可能性があります。非マップ領域のアドレスに対するIndexオペレーション[注]は，TLB例外の発生を回避するために使用します。また，Indexオペレーションは決してTLB変更例外を発生しません。命令コードのビット16とビット17が，次のようにオペレーション対象となるキャッシュを示します。

コード	略号	キャッシュの種類
0	I	命令キャッシュ
1	D	データ・キャッシュ
2	−	予約
3	−	予約

注　ここで物理アドレスがキャッシュをインデクスするために使用されますが，キャッシュ・タグと一致する必要はありません。

この命令のビット20-ビット18は，キャッシュ・オペレーションの内容を指定します。詳細については次ページ以降を参照してください。

保守／廃止

CACHE

Cache Operation

（3/4）

op4..2	キャッシュ	キャッシュ・オペレーション	説　　明
0	I	Index_Invalidate	キャッシュ・ブロックのキャッシュ状態をInvalidにセットします。
0	D	Index_Write_Back _Invalidate	仮想アドレスによって示されるインバリデート・インデクスに対するデータ・キャッシュ・ブロックのキャッシュ状態とWビットを検査します。その状態がInvalidでなくWビットがセットされていれば，このブロックをメイン・メモリへライトバックします。そのアドレスへのライトは，キャッシュ・タグのアドレスを使用します。キャッシュ・ブロックのキャッシュ状態をInvalidにセットします。
1	I, D	Index_Load_Tag	指定されたインデクスに対するキャッシュ・ブロックのタグを読み出し，コプロセッサ0のタグLoレジスタにロードします。
2	I, D	Index_Store_Tag	CP0レジスタのタグLoレジスタの内容を，指定されたインデクスに対するキャッシュ・ブロックのタグに書き込みます。
3	D	Create_Dirty_Exclusive	このオペレーションは，キャッシュ・ブロックに新しいデータをライトする場合，メイン・メモリからのデータ・ロードを必要最低限に抑えるのに使用します。キャッシュ・ブロックに指定されたアドレスが含まれていない，かつ，そのブロックがdirtyである場合，メイン・メモリにライトバックを行います。すべての場合，キャッシュ状態をDirtyにセットします。
4	I, D	Hit_Invalidate	キャッシュ・ブロックが指定されたアドレスを含む場合，そのキャッシュ・ブロックのキャッシュ状態をInvalidにセットします。
5	D	Hit_Write_Back_Invalidate	キャッシュ・ブロックが指定されたアドレスを含む場合，そのブロックがdirtyならばデータをライトバックし，そのキャッシュ・ブロックのキャッシュ状態をInvalidにします。
5	I	Fill	命令キャッシュ・ブロックをメイン・メモリからの命令データで埋めます。
6	D	Hit_Write_Back	キャッシュ・ブロックが指定されたアドレスを含んでおり，Wビットがセットされていればデータをメイン・メモリにライトバックしWビットをクリアします。
6	I	Hit_Write_Back	キャッシュ・ブロックが指定されたアドレスを含んでいたならば，無条件にデータのライトバックを行います。

保守／廃止

CACHE

Cache Operation

オペレーション：

```
32，64　T：　vAddr ← ((offset15)^48 ‖ offset15..0）＋GPR[base]
　　（pAddr，uncached）← AddressTranslation（vAddr，DATA）
　　CacheOp（op，vAddr，pAddr）
```

例　　外：

コプロセッサ使用不可例外

TLB**無効例外**

TLB**不一致例外**

バス・エラー例外

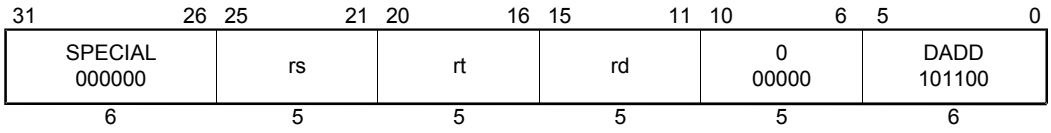
アドレス・エラー例外

キャッシュ・エラー例外

保守／廃止

DADD

Doubleword Add



命　令　形　式：

DADD　rd，rs，rt

説　　明：

汎用レジスタrs**の内容と汎用レジスタ**rt**の内容を加算し，結果を汎用レジスタ**rd**に格納します。ビット**62**からの桁上がりとビット**63**からの桁上がりが異なる（**2**の補数オーバフロー）場合，整数オーバフロー例外が発生します。整数オーバフロー例外が発生した場合，デスティネーション・レジスタ**rd**の内容は変更しません。**

この演算は，64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

オペレーション：

```
64　T：　GPR[rd] ← GPR[rs]＋GPR[rt]
```

例　　外：

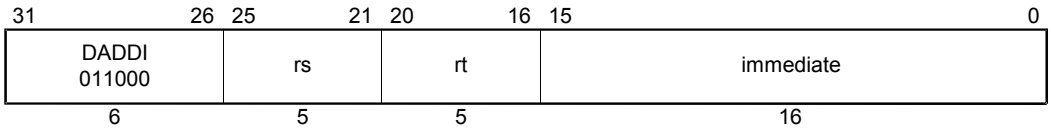
整数オーバフロー例外

予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

DADDI

Doubleword Add Immediate



命　令　形　式：

DADDI　rt，rs，immediate

説　　明：

符号拡張した16ビットimmediateと汎用レジスタrsの内容を加算し，結果を汎用レジスタrtに格納します。ビット62からの桁上がりとビット63からの桁上がりが異なる（2の補数オーバフロー）場合，整数オーバフロー例外が発生します。整数オーバフロー例外が発生した場合，デスティネーション・レジスタrtの内容は変更しません。

この演算は，64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

オペレーション：

```
64　T：　GPR[rt] ← GPR[rs]＋(immediate15)^48 ‖ immediate15..0
```

例　　外：

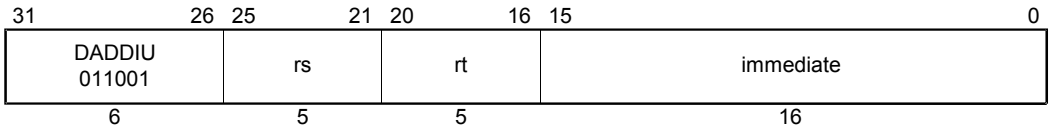
整数オーバフロー例外

予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

DADDIU

Doubleword Add Immediate Unsigned



命　令　形　式：

DADDIU　rt，rs，immediate

説　　明：

符号拡張した16ビットimmediate**と汎用レジスタ**rs**の内容を加算し，結果を汎用レジスタ**rt**に格納します。**

この演算は，64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

DADDI**命令との違いは，**DADDIU**命令は整数オーバフロー例外を発生しないという点だけです。**

オペレーション：

```
64　T：　GPR[rt] ← GPR[rs]＋(immediate15)^48 ‖ immediate15..0
```

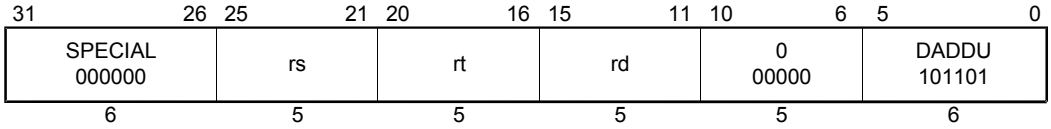
例　　外：

予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

DADDU

Doubleword Add Unsigned



命　令　形　式：

DADDU　rd，rs，rt

説　　明：

汎用レジスタrs**の内容と汎用レジスタ**rt**の内容を加算し，結果を汎用レジスタ**rd**に格納します。**

この演算は，64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

DADD**命令との違いは，**DADDU**命令は整数オーバフロー例外を発生しないという点だけです。**

オペレーション：

```
64　T：　GPR[rd] ← GPR[rs]＋GPR[rt]
```

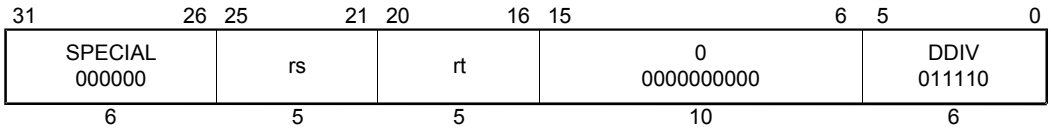
例　　外：

予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

DDIV

Doubleword Divide



命　令　形　式：

DDIV　rs，rt

説　　明：

汎用レジスタrs**の内容を汎用レジスタ**rt**の内容で除算します。両オペランドは，符号付き整数として取り扱われます。整数オーバフロー例外は発生しません。除数が0のときの結果は未定義です。**

通常，この命令はゼロ除算とオーバフローをチェックする命令のあとに実行します。

演算が完了すると，商（ダブル・ワード）を特殊レジスタLO**に格納し，剰余（ダブル・ワード）を特殊レジスタ**HI**に格納します。**

直前の2**つの命令のいずれかが**MFHI**命令または**MFLO**命令の場合，これらの命令の実行結果は未定義になります。正しい結果を得るには，**MFHI**，**MFLO**命令と**DDIV**命令の間に，**2**つ以上のほかの命令を入れてください。**

この演算は，64**ビット・モード時と**32**ビット・カーネル・モード時に定義されています。**32**ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると予約命令例外が発生します。**

オペレーション：

```
64   T－2：   LO ← undefined
              HI ← undefined
     T－1：   LO ← undefined
              HI ← undefined
     T：      LO ← GPR[rs] div GPR[rt]
              HI ← GPR[rs] mod GPR[rt]
```

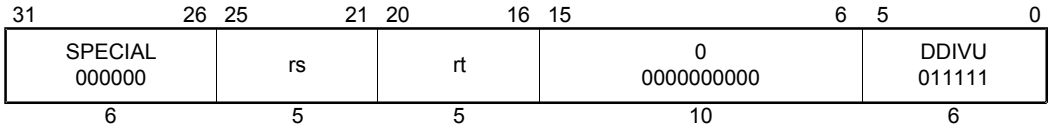
例　　外：

予約命令例外（32**ビット・ユーザ／スーパバイザ・モード時）**

保守／廃止

DDIVU

Doubleword Divide Unsigned



命　令　形　式：

DDIVU　rs，rt

説　　明：

汎用レジスタrs**の内容を汎用レジスタ**rt**の内容で除算します。両オペランドは，符号なし整数として扱われます。整数オーバフロー例外は発生しません。除数が0のときの結果は未定義です。**

通常，この命令はゼロ除算をチェックする命令のあとに実行します。

演算が完了すると，商（ダブル・ワード）を特殊レジスタLO**に格納し，剰余（ダブル・ワード）を特殊レジスタ**HI**に格納します。**

直前の2**つの命令のいずれかが**MFHI**命令または**MFLO**命令の場合，これらの命令の実行結果は未定義になります。正しい結果を得るには，**MFHI**，**MFLO**命令と**DDIVU**命令の間に，**2**つ以上のほかの命令を入れてください。**

この演算は，64**ビット・モード時と**32**ビット・カーネル・モード時に定義されています。**32**ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。**

オペレーション：

```
64   T－2：   LO ← undefined
              HI ← undefined
     T－1：   LO ← undefined
              HI ← undefined
     T：      LO ← (0 || GPR[rs]) div (0 || GPR[rt])
              HI ← (0 || GPR[rs]) mod (0 || GPR[rt])
```

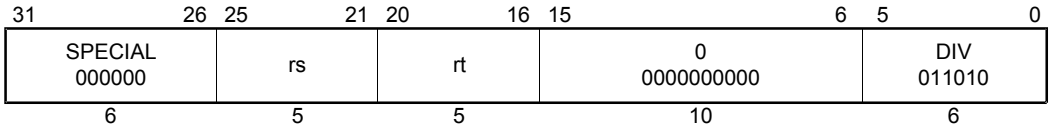
例　　外：

予約命令例外（32**ビット・ユーザ／スーパバイザ・モード時）**

保守／廃止

DIV

Divide



命　令　形　式：

DIV　rs，rt

説　　明：

汎用レジスタrs**の内容を汎用レジスタ**rt**の内容で除算します。両オペランドは，符号付き整数として扱われます。整数オーバフロー例外は発生しません。除数が0のときの結果は未定義です。64ビット・モード時，結果は32ビット値を符号拡張します。**

通常，この命令はゼロ除算とオーバフローをチェックする命令のあとに実行します。

演算が完了すると，商（ダブル・ワード）を特殊レジスタLO**に格納し，剰余（ダブル・ワード）を特殊レジスタ**HI**に格納します。**

直前の2つの命令のいずれかがMFHI**命令または**MFLO**命令の場合，これらの命令の実行結果は未定義になります。正しい結果を得るには，**MFHI**，**MFLO**命令と**DIV**命令の間に，2つ以上のほかの命令を入れてください。**

オペレーション：

```
32  T−2：   LO ← undefined
             HI ← undefined
     T−1：   LO ← undefined
             HI ← undefined
     T：     LO ← GPR[rs] div GPR[rt]
             HI ← GPR[rs] mod GPR[rt]

64  T−2：   LO ← undefined
             HI ← undefined
     T−1：   LO ← undefined
             HI ← undefined
     T：     q ← GPR[rs]31..0  div GPR[rt]31..0
             r ← GPR[rs]31..0  mod GPR[rt]31..0
             LO ← (q31)^32 || q31..0
             HI ← (r31)^32 || r31..0
```

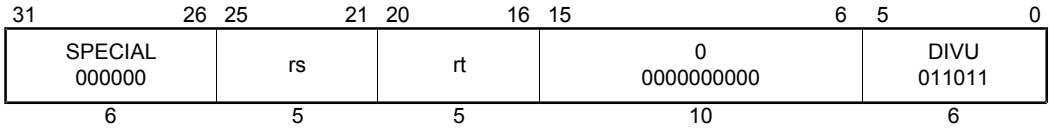
例　　外：

なし

保守／廃止

DIVU

Divide Unsigned



命　令　形　式：

DIVU　rs，rt

説　　明：

汎用レジスタrs**の内容を汎用レジスタ**rt**の内容で除算します。両オペランドは，符号なし整数として扱われます。整数オーバフロー例外は発生しません。除数が0のときの結果は未定義です。64ビット・モード時，結果は32ビット値を符号拡張します。**

通常，この命令はゼロ除算をチェックする命令のあとに実行します。

演算が完了すると，商（ダブル・ワード）を特殊レジスタLO**に格納し，剰余（ダブル・ワード）を特殊レジスタ**HI**に格納します。**

直前の2つの命令のいずれかがMFHI**命令または**MFLO**命令の場合，これら命令の実行結果は未定義になります。正しい結果を得るには，**MFHI，MFLO**命令と**DIVU**命令との間に，2つ以上のほかの命令を入れてください。**

オペレーション：

```
32  T−2：   LO ← undefined
             HI ← undefined
     T−1：   LO ← undefined
             HI ← undefined
     T：     LO ← （0 ‖ GPR[rs]）div（0 ‖ GPR[rt]）
             HI ← （0 ‖ GPR[rs]）mod（0 ‖ GPR[rt]）

64  T−2：   LO ← undefined
             HI ← undefined
     T−1：   LO ← undefined
             HI ← undefined
     T：     q ← （0 ‖ GPR[rs]31..0）div（0 ‖ GPR[rt]31..0）
             r ← （0 ‖ GPR[rs]31..0）mod（0 ‖ GPR[rt]31..0）
             LO ← (q31)^32 ‖ q31..0
             HI ← (r31)^32 ‖ r31..0
```

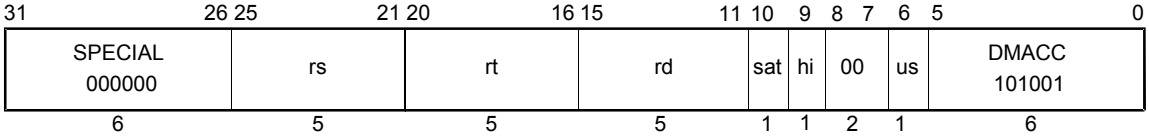
例　　外：

なし

保守／廃止

DMACC

Doubleword Multiply and Accumulate



命　令　形　式：

DMACC　　rd，rs，rt

DMACCU　　rd，rs，rt

DMACCHI　　rd，rs，rt

DMACCHIU　　rd，rs，rt

DMACCS　　rd，rs，rt

DMACCUS　　rd，rs，rt

DMACCHIS　　rd，rs，rt

DMACCHIUS　　rd，rs，rt

説　　明：

DMACC**命令は，オペコードの**sat，hi，us**の各ビットの設定により，次のようにニモニックが異なります。**

ニモニック	sat	hi	us
DMACC	0	0	0
DMACCU	0	0	1
DMACCHI	0	1	0
DMACCHIU	0	1	1
DMACCS	1	0	0
DMACCUS	1	0	1
DMACCHIS	1	1	0
DMACCHIUS	1	1	1

また，飽和処理実行時（sat = 1**）と飽和処理非実行時（**sat = 0**）とで，オペランドの有効ビット数が異なります。**

・飽和処理実行時（sat = 1）：DMACCS, DMACCUS, DMACCHIS, DMACCHIUS命令

汎用レジスタrs**の内容と汎用レジスタ**rt**の内容を乗算します。両オペランドは，**us = 1**の場合，16ビットの符号なしデータとして扱われ，**us = 0**の場合，16ビットの符号付き整数として扱われます。オペランドのビット16以上のビットはソフトウエアで符号／ゼロ拡張されている必要があります。**

乗算の結果は，特殊レジスタLO**の値と加算されます。加算は，**us = 1**の場合，32ビットの符号なしデータとして行われ，**us = 0**の場合，32ビットの符号付き整数として行われます。特殊レジスタ**LO**のビット32以上のビットは符号／ゼロ拡張されている必要があります。**

保守／廃止

DMACC

Doubleword Multiply and Accumulate

加算の結果は，32ビットで飽和処理（下表参照）を行ったあと，特殊レジスタLOにロードされます。hi = 1の場合，特殊レジスタHIにロードされるデータと同じものが汎用レジスタrdへもロードされます。hi = 0の場合，特殊レジスタLOにロードされるデータと同じものが汎用レジスタrdへもロードされます。オーバフロー例外は発生しません。

・飽和処理非実行時（sat = 0）：DMACC, DMACCU, DMACCHI, DMACCHIU命令

汎用レジスタrsの内容と汎用レジスタrtの内容を乗算します。両オペランドは，us = 1の場合，32ビットの符号なしデータとして扱われ，us = 0の場合，32ビットの符号付き整数として扱われます。オペランドのビット32以上のビットはソフトウエアで符号／ゼロ拡張されている必要があります。

乗算の結果は，特殊レジスタLOの値と加算されます。加算は，us = 1の場合，64ビットの符号なしデータとして行われ，us = 0の場合，64ビットの符号付き整数として行われます。

加算の結果は，特殊レジスタLOにロードされます。hi = 1の場合，特殊レジスタHIにロードされるデータと同じものが汎用レジスタrdへもロードされます。hi = 0の場合，特殊レジスタLOにロードされるデータと同じものが汎用レジスタrdへもロードされます。オーバフロー例外は発生しません。

この演算は64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

次に，us，satの設定と飽和処理時の格納値の対応，およびレジスタHIとLOを操作する命令とDMACC命令の間に必要なハザード・サイクルを示します。

飽和処理時の格納値

us	sat	オーバフロー	アンダフロー
0	0	演算結果を そのまま格納	演算結果を そのまま格納
1	0	演算結果を そのまま格納	演算結果を そのまま格納
0	1	0x0000 0000 7FFF FFFF	0xFFFF FFFF 8000 0000
1	1	0xFFFF FFFF FFFF FFFF	なし

ハザード・サイクル数

命　　令	サイクル数
MULT, MULTU	1
DMULT, DMULTU	3
DIV, DIVU	36
DDIV, DDIVU	68
MFHI, MFLO	2
MTHI, MTLO	0
MACC	0
DMACC	0

オペレーション：

```
64, sat=0, hi=0, us=0
        T:  temp1 ← ((GPR[rs]31)32 || GPR[rs]) * ((GPR[rt]31)32 || GPR[rt])
            temp2 ← temp1 + LO
            LO ← temp2
            GPR[rd] ← LO
```

保守／廃止

DMACC

Doubleword Multiply and Accumulate


```
64, sat=0, hi=0, us=1
        T:   temp1 ← (0^32 || GPR[rs]) * (0^32 || GPR[rt])
             temp2 ← temp1 + LO
             LO ← temp2
             GPR[rd] ← LO
64, sat=0, hi=1, us=0
        T:   temp1 ← ((GPR[rs]31)^32 || GPR[rs]) * ((GPR[rt]31)^32 || GPR[rt])
             temp2 ← temp1 + LO
             LO ← temp2
             GPR[rd] ← HI
64, sat=1, hi=1, us=1
        T:   temp1 ← (0^32 || GPR[rs]) * (0^32 || GPR[rt])
             temp2 ← temp1 + LO
             LO ← temp2
             GPR[rd] ← HI
64, sat=1,hi=0, us=0
        T:   temp1 ← ((GPR[rs]31)^32 || GPR[rs]) * ((GPR[rt]31)^32 || GPR[rt])
             temp2 ← saturation(temp1 + LO)
             LO ← temp2
             GPR[rd] ← LO
64, sat=1, hi=0, us=1
        T:   temp1 ← (0^32 || GPR[rs]) * (0^32 || GPR[rt])
             temp2 ← saturation(temp1 + LO)
             LO ← temp2
             GPR[rd] ← LO
64, sat=1, hi=1, us=0
        T:   temp1 ← ((GPR[rs]31)^32 || GPR[rs]) * ((GPR[rt]31)^32 || GPR[rt])
             temp2 ← saturation(temp1 + LO)
             LO ← temp2
             GPR[rd] ← HI
64, sat=1, hi=1, us=1
        T:   temp1 ← (0^32 || GPR[rs]) * (0^32 || GPR[rt])
             temp2 ← saturation(temp1 + LO)
             LO ← temp2
             GPR[rd] ← HI
```

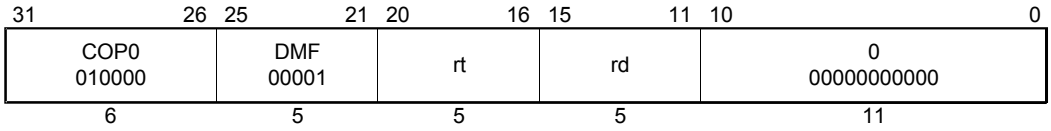
例　　外：

予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

DMFC0

Doubleword Move From System Control Coprocessor



命　令　形　式：

DMFC0　rt，rd

説　　明：

CP0のコプロセッサ・レジスタrdの内容を，汎用レジスタrtにロードします。

この演算は，64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

デスティネーションの64ビット汎用レジスタrtに，ソースのコプロセッサ・レジスタrdの内容を書き込みます。CP0の32ビット・レジスタに対するDMFC0命令の動作は不定です。

オペレーション：

```
64　T ：　　data ← CPR[0，rd]
　　T＋1：　GPR[rt] ← data
```

例　　外：

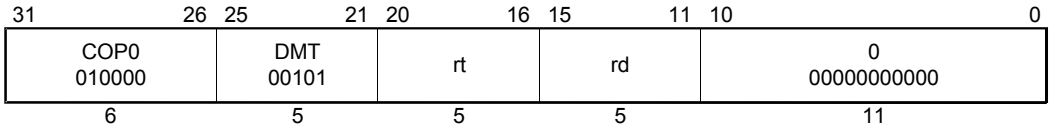
コプロセッサ使用不可例外（CP0が禁止されている場合で64／32ビット・ユーザ／スーパバイザ・モード時）

予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

DMTC0

Doubleword Move To System Control Coprocessor



命　令　形　式：

DMTC0　rt，rd

説　　明：

汎用レジスタrt**の内容を，**CP0**のコプロセッサ・レジスタ**rd**にロードします。**

この演算は，64**ビット・モード時と**32**ビット・カーネル・モード時に定義されています。**32**ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。**

デスティネーションの64**ビット・コプロセッサ・レジスタ**rt**に，ソースの汎用レジスタ**rd**の内容を書き込みます。**CP0**の**32**ビット・レジスタに対する**DMTC0**命令の動作は不定です。**

仮想アドレス変換システムの状態は，この命令の実行により変更されることがあるため，この命令の直前および直後における，ロード命令，ストア命令およびTLB**操作命令は未定義です。**

オペレーション：

```
64    T ：      data ← GPR[rt]
      T+1：     CPR[0，rd] ← data
```

例　　外：

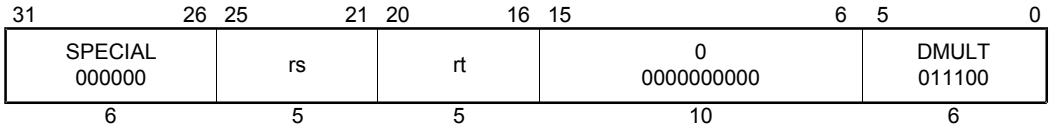
コプロセッサ使用不可例外（CP0**が禁止されている場合で**64／32**ビット・ユーザ／スーパバイザ・モード時）**

予約命令例外（32**ビット・ユーザ／スーパバイザ・モード時）**

保守／廃止

DMULT

Doubleword Multiply



命　令　形　式：

DMULT　rs，rt

説　　明：

汎用レジスタrs**の内容と汎用レジスタ**rt**の内容を乗算します。両オペランドは，符号付き整数として扱われます。整数オーバフロー例外は発生しません。**

演算が完了すると，結果の下位ダブル・ワードを特殊レジスタLO**に格納し，結果の上位ダブル・ワードを特殊レジスタ**HI**に格納します。**

直前の2**つの命令のいずれかが**MFHI**命令または**MFLO**命令の場合，これらの命令の実行結果は未定義になります。正しい結果を得るには，**MFHI**，**MFLO**命令と**DMULT**命令との間に，**2**つ以上のほかの命令を入れてください。**

この演算は，64**ビット・モード時および**32**ビット・カーネル・モード時に定義されています。**32**ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。**

オペレーション：

```
64   T－2：   LO ← undefined
              HI ← undefined
     T－1：   LO ← undefined
              HI ← undefined
     T：      t ← GPR[rs] ＊ GPR[rt]
              LO ← t63..0
              HI ← t127..64
```

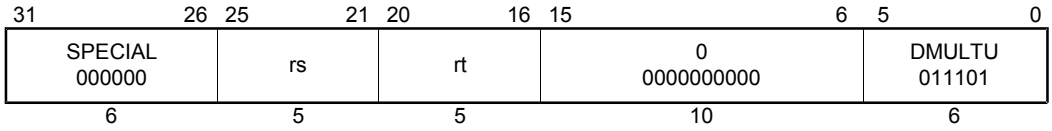
例　　外：

予約命令例外（32**ビット・ユーザ／スーパバイザ・モード時）**

保守／廃止

DMULTU

Doubleword Multiply Unsigned



命　令　形　式：

DMULTU　rs，rt

説　　明：

汎用レジスタrs**の内容と汎用レジスタ**rt**の内容を乗算します。両オペランドは，符号なし整数として扱われます。整数オーバフロー例外は発生しません。**

演算が完了すると，結果の下位ダブル・ワードを特殊レジスタLO**に格納し，結果の上位ダブル・ワードを特殊レジスタ**HI**に格納します。**

直前の2つの命令のいずれかがMFHI**命令または**MFLO**命令の場合，これらの命令の実行結果は未定義になります。正しい結果を得るには，**MFHI**，**MFLO**命令と**DMULTU**命令の間に，2つ以上のほかの命令を入れてください。**

この演算は，64ビット・モード時および32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

オペレーション：

```
64   T−2：   LO ← undefined
              HI ← undefined
     T−1：   LO ← undefined
              HI ← undefined
     T：      t ← （0 ‖ GPR[rs]）＊（0 ‖ GPR[rt]）
              LO ← t63..0
              HI ← t127..64
```

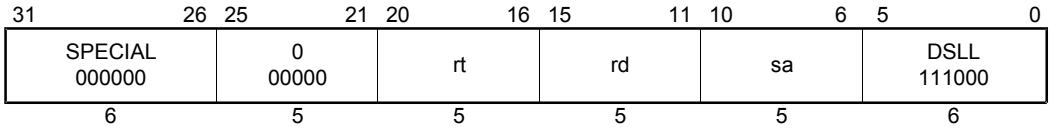
例　　外：

予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

DSLL

Doubleword Shift Left Logical



命　令　形　式：

DSLL　rd，rt，sa

説　　明：

汎用レジスタrt**の内容を，**sa**ビット左にシフトします。下位ビットには**0**を挿入します。結果は，汎用レジスタ**rd**に格納します。**

この演算は，64**ビット・モード時と**32**ビット・カーネル・モード時に定義されています。**32**ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。**

オペレーション：

```
64　T ：　s ← 0 || sa
          GPR[rd] ← GPR[rt] (63−s) ..0 || 0^s
```

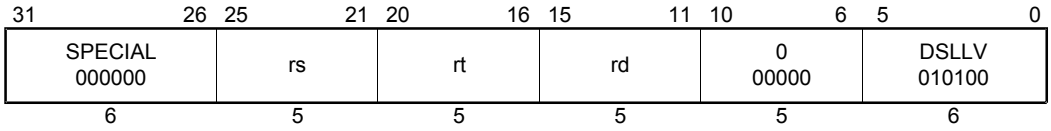
例　　外：

予約命令例外（32**ビット・ユーザ／スーパバイザ・モード時）**

保守／廃止

DSLLV

Doubleword Shift Left Logical Variable



命　令　形　式：

DSLLV　rd，rt，rs

説　　明：

汎用レジスタrtの内容を，汎用レジスタrsの内容の下位6ビットで指定される数だけ，左にシフトします。下位ビットには0を挿入します。結果は，汎用レジスタrdに格納します。

この演算は，64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

オペレーション：

```
64　T：　s ← GPR[rs]5..0
         GPR[rd] ← GPR[rt](63−s)..0 ‖ 0^s
```

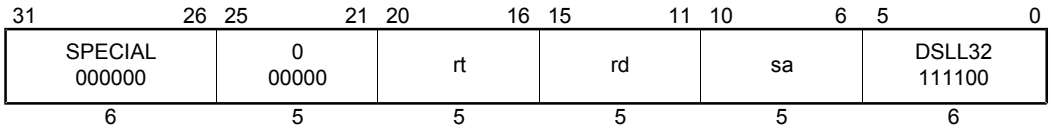
例　　外：

予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

DSLL32

Doubleword Shift Left Logical＋32



命　令　形　式：

DSLL32　rd，rt，sa

説　　明：

汎用レジスタrt**の内容を，**32＋sa**ビット左にシフトします。下位ビットには**0**を挿入します。結果は，汎用レジスタ**rd**に格納します。**

この演算は，64**ビット・モード時と**32**ビット・カーネル・モード時に定義されています。**32**ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。**

オペレーション：

```
64　T ：　s ← 1 || sa
          GPR[rd] ← GPR[rt] (63−s) ..0 || 0^s
```

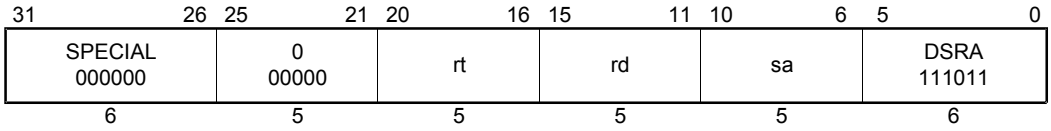
例　　外：

予約命令例外（32**ビット・ユーザ／スーパバイザ・モード時）**

保守／廃止

DSRA

Doubleword Shift Right Arithmetic



命　令　形　式：

DSRA　rd，rt，sa

説　　明：

汎用レジスタrt**の内容を，**sa**ビット右にシフトします。上位ビットは符号拡張します。結果は，汎用レジスタ**rd**に格納します。**

この演算は，64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

オペレーション：

```
64　T ：　s ← 0 || sa
          GPR[rd] ← (GPR[rt]63)^s || GPR[rt]63..s
```

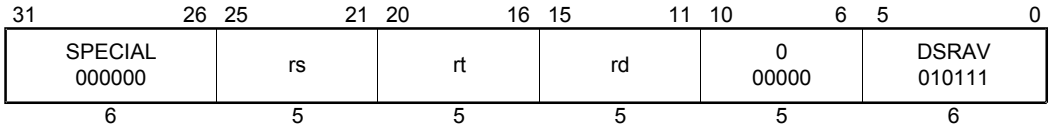
例　　外：

予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

DSRAV

Doubleword Shift Right Arithmetic Variable



命　令　形　式：

DSRAV　rd，rt，rs

説　　明：

汎用レジスタrt**の内容を，汎用レジスタ**rs**の内容の下位6ビットで指定される数だけ，右にシフトします。上位ビットは符号拡張します。結果は，汎用レジスタ**rd**に格納します。**

この演算は，64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

オペレーション：

```
64　T：　s ← GPR[rs]5..0
         GPR[rd] ← (GPR[rt]63)^s ‖ GPR[rt]63..s
```

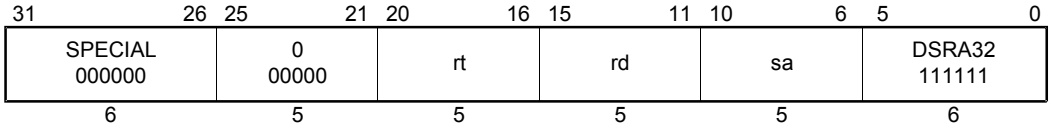
例　　外：

予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

DSRA32

Doubleword Shift Right Arithmetic＋32



命　令　形　式：

DSRA32　rd，rt，sa

説　　明：

汎用レジスタrt**の内容を，**32＋sa**ビット右にシフトします。上位ビットは符号拡張します。結果は，汎用レジスタ**rd**に格納します。**

この演算は，64**ビット・モード時と**32**ビット・カーネル・モード時に定義されています。**32**ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。**

オペレーション：

```
64　T ：　s ← 1 || sa
          GPR[rd] ← (GPR[rt]63)^s || GPR[rt]63..s
```

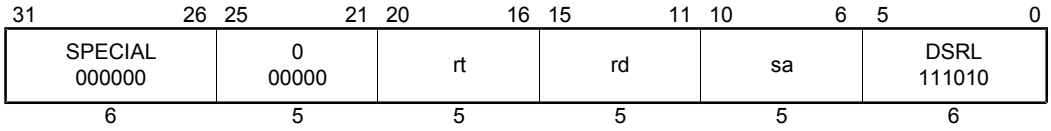
例　　外：

予約命令例外（32**ビット・ユーザ／スーパバイザ・モード時）**

保守／廃止

DSRL

Doubleword Shift Right Logical



命　令　形　式：

DSRL　rd，rt，sa

説　　明：

汎用レジスタrt**の内容を，**sa**ビット右にシフトします。上位ビットには**0**を挿入します。結果は，汎用レジスタ**rd**に格納します。**

この演算は，64**ビット・モード時と**32**ビット・カーネル・モード時に定義されています。**32**ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。**

オペレーション：

```
64　T ：　s ← 0 || sa
          GPR[rd] ← 0^s || GPR[rt]63..s
```

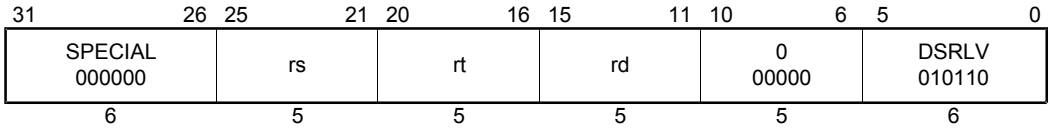
例　　外：

予約命令例外（32**ビット・ユーザ／スーパバイザ・モード時）**

保守／廃止

DSRLV

Doubleword Shift Right Logical Variable



命　令　形　式：

DSRLV　rd，rt，rs

説　　明：

汎用レジスタrt**の内容を，汎用レジスタ**rs**の内容の下位6ビットで指定される数だけ，右にシフトします。上位ビットには0を挿入します。結果は，汎用レジスタ**rd**に格納します。**

この演算は，64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

オペレーション：

```
64　T ：　s ← GPR[rs]5..0
          GPR[rd] ← 0^s || GPR[rt]63..s
```

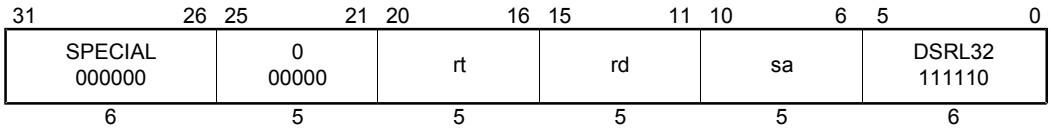
例　　外：

予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

DSRL32

Doubleword Shift Right Logical＋32



命　令　形　式：

DSRL32　rd，rt，sa

説　　明：

汎用レジスタrt**の内容を，**32＋sa**ビット右にシフトします。上位ビットには**0**を挿入します。結果は，汎用レジスタ**rd**に格納します。**

この演算は，64**ビット・モード時と**32**ビット・カーネル・モード時に定義されています。**32**ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。**

オペレーション：

```
64　T ：　s ← 1 || sa
          GPR[rd] ← 0^s || GPR[rt]63..s
```

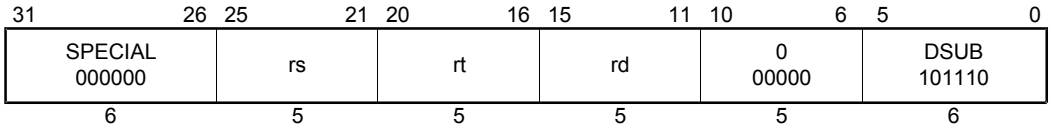
例　　外：

予約命令例外（32**ビット・ユーザ／スーパバイザ・モード時）**

保守／廃止

DSUB

Doubleword Subtract



命　令　形　式：

DSUB　rd，rs，rt

説　　明：

汎用レジスタrs**の内容から汎用レジスタ**rt**の内容を減算し，結果を汎用レジスタ**rd**に格納します。**

ビット62**からの桁上がりとビット**63**からの桁上がりが異なる（**2**の補数オーバフロー）場合，整数オーバフロー例外が発生します。整数オーバフロー例外が発生した場合，デスティネーション・レジスタ**rd**の内容は変更しません。**

この演算は，64**ビット・モード時と**32**ビット・カーネル・モード時に定義されています。**32**ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。**

オペレーション：

```
64　T ：　GPR[rd] ← GPR[rs]－GPR[rt]
```

例　　外：

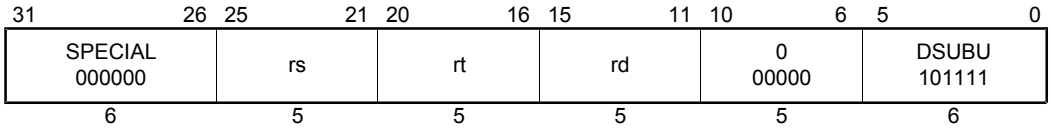
整数オーバフロー例外

予約命令例外（32**ビット・ユーザ／スーパバイザ・モード時）**

保守／廃止

DSUBU

Doubleword Subtract Unsigned



命　令　形　式：

DSUBU　rd，rs，rt

説　　明：

汎用レジスタrs**の内容から汎用レジスタ**rt**の内容を減算し，結果を汎用レジスタ**rd**に格納します。**

DSUB**命令との違いは，**DSUBU**命令は整数オーバフローを発生しないという点だけです。**

この演算は，64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

オペレーション：

```
64　T ：　GPR[rd] ← GPR[rs]－GPR[rt]
```

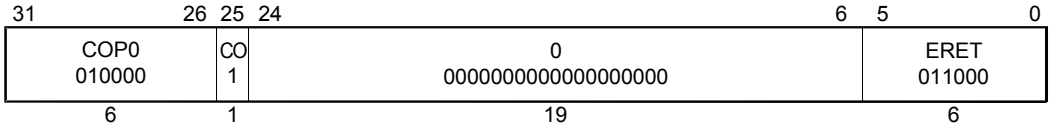
例　　外：

予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

ERET

Return From Exception



命　令　形　式：

ERET

説　　明：

ERET命令は，割り込み，例外およびエラー例外から復帰するための命令です。ブランチ命令やジャンプ命令と違って，ERET命令は直後の命令を実行しません。

ERET命令は，分岐遅延スロット内にあってはなりません。

ステータス・レジスタのERLビットがセット（SR_2 = 1）されている場合，エラーEPCレジスタの内容をPCにロードし，ERLビットをクリアします。そうでない場合（SR_2 = 0）はEPCからPCをロードし，ステータス・レジスタのEXLビットをクリア（SR_1 = 0）します。

MIPS16命令実行可能な場合，EPCレジスタまたはエラーEPCレジスタの最下位ビットを0にクリアした値をPCにロードし，最下位ビットの内容をISAモード・ビット（内部）に反映させます（μPD98502ではMIPS16モードをサポートしていません）。

オペレーション：

```
32, 64    T：    if SR2 = 1 then
                   if MIPS16EN = 1 then
                     PC ← ErrorEPC63..1 ‖ 0
                     ISA MODE ← ErrorEPC0
                   else
                     PC ← ErrorEPC
                   endif
                   SR ← SR31..3 ‖ 0 ‖ SR1..0
                 else
                   if MIPS16EN = 1 then
                     PC ← EPC63..1 ‖ 0
                     ISA MODE ← EPC0
                   else
                     PC ← EPC
                   endif
                   SR ← SR31..2 ‖ 0 ‖ SR0
                 endif
```

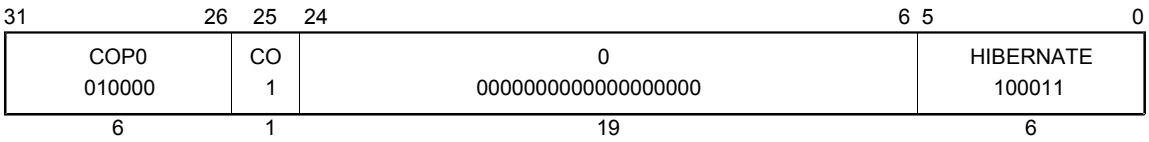
例　　外：

コプロセッサ使用不可例外

保守／廃止

HIBERNATE

Hibernate



命　令　形　式：

HIBERNATE

説　　明：

VR4120Aコアを Fullspeed モードから Hibernate モードに移行します。

命令の実行が WB ステージまで進むと，SysAD バスがアイドル状態になるのを待って CPU コアが生成するすべてのクロックをハイ・レベルに固定し，パイプラインの動作を中断します。

Hibernate モードから Fullspeed モードに移行するには，コールド・リセットを実行します。

オペレーション：

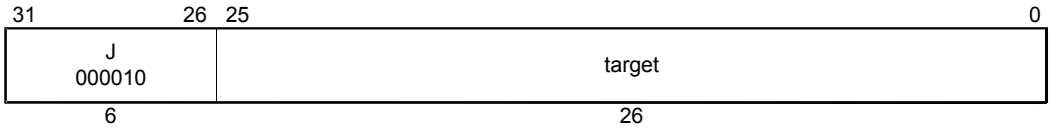
```
32, 64    T：
          T＋1：  Hibernate Operation  ()
```

例　　外：

コプロセッサ使用不可例外

J

Jump



命　令　形　式：

J　target

説　　明：

26ビットのtargetを2ビット左にシフトし，遅延スロットのアドレスの上位4ビットと結合してターゲット・アドレスを計算します。1命令の遅延付きで，無条件にターゲット・アドレスへジャンプします。

オペレーション：

```
32   T：    temp ← target
          T+1： PC ← PC31..28 || temp || 0^2

64   T：    temp ← target
          T+1： PC ← PC63..28 || temp || 0^2
```

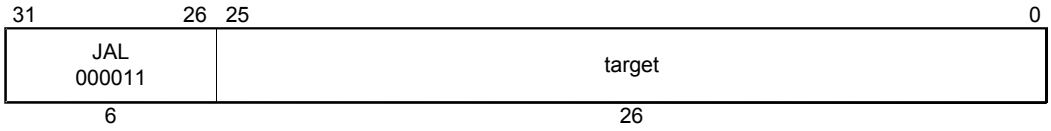
例　　外：

なし

保守／廃止

JAL

Jump And Link



命　令　形　式：

JAL　target

説　　明：

26ビットのtargetを2ビット左にシフトし，遅延スロットのアドレスの上位4ビットと結合してターゲット・アドレスを計算します。1命令の遅延付きで，無条件にターゲット・アドレスへジャンプします。遅延スロットの直後の命令のアドレスを，リンク・レジスタ（r31）に格納します。MIPS16命令実行可能な場合は，リンク・レジスタ（r31）のビット0はジャンプ実行前のISAモード・ビット（内部）の値を示します（μPD98502ではMIPS16モードをサポートしていません）。

オペレーション：

```
32 T：    temp ← target
          if MIPS16EN = 1 then
             GPR[31] ← (PC＋8)31..1 ‖ ISA MODE
          else
             GPR[31] ← PC＋8
          endif
   T＋1：  PC ← PC31..28 ‖ temp ‖ 0^2

64 T：    temp ← target
          if MIPS16EN = 1 then
             GPR[31] ← (PC＋8)63..1 ‖ ISA MODE
          else
             GPR[31] ← PC＋8
          endif
   T＋1：  PC ← PC63..28 ‖ temp ‖ 0^2
```

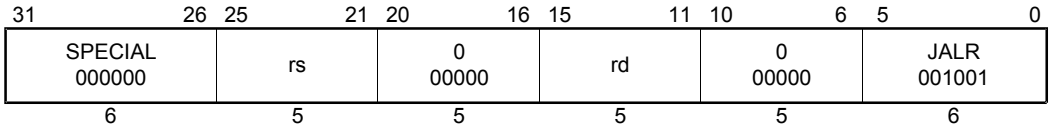
例　　外：

なし

保守／廃止

JALR

Jump And Link Register



命　令　形　式：

JALR　rs

JALR　rd，rs

説　　明：

1命令の遅延付きで，汎用レジスタrsの内容のアドレスに無条件でジャンプします。

MIPS16命令実行可能な場合，1命令の遅延付きで，汎用レジスタrsの最下位ビットを0にクリアした値のアドレスに無条件でジャンプし，汎用レジスタrsの最下位ビットの内容をISAモード・ビット（内部）にセットします。

遅延スロットの直後の命令のアドレスが汎用レジスタrdに格納されます。rdを省略した場合，既定値（31）となります。MIPS16命令実行可能な場合は，rdのビット0はジャンプ実行前のISAモード・ビット（内部）の値を示します（μPD98502ではMIPS16モードをサポートしていません）。

レジスタ番号rsとrdは，この命令が再実行されたとき同じ結果になるとは限らないので，等しくすべきではありません。等しいと，リンク・アドレスのストアによってrsの内容が破壊されるからです。しかし，このような命令を実行しても例外は発生せず，実行結果は未定義になります。

32ビット長命令は，ワード境界に位置合わせされていなければならないので，MIPS16命令実行可能な場合，JALR命令は下位2ビットが0であるターゲット・レジスタ（rs）を指定しなければなりません。下位2ビットが0でない場合，ジャンプ先の命令をフェッチしたときにアドレス例外が発生します。

オペレーション：

```
32, 64   T：   temp ← GPR[rs]
               if MIPS16EN = 1 then
                   GPR[rd] ← (PC+8)63..1 || ISA MODE
               else
                   GPR[rd] ← PC+8
               endif
         T+1：if MIPS16EN = 1 then
                   PC ← temp63..1 || 0
                   ISA MODE ← temp0
               else
                   PC ← temp
               endif
```

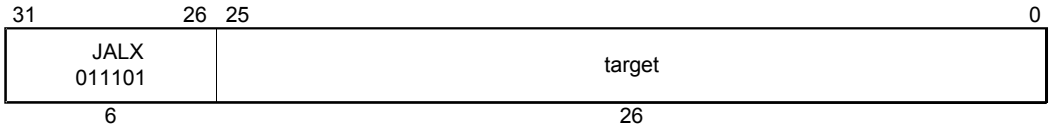
例　　外：

なし

保守／廃止

JALX

Jump And Link Exchange



命　令　形　式：

JALX　target

説　　明：

MIPS16命令実行可能な場合，26ビットのtargetを2ビット左にシフトし，遅延スロットのアドレスの上位4ビットと結合してターゲット・アドレスを計算します。1命令の遅延付きで，無条件にターゲット・アドレスへジャンプします。遅延スロットの直後の命令のアドレスを，リンク・レジスタ（r31）に格納します。1命令の遅延付きでISAモード・ビット（内部）を反転します。リンク・レジスタ(r31)のビット0は，ジャンプ実行前のISAモード・ビット（内部）の値を示します（μPD98502ではMIPS16モードをサポートしていません）。

オペレーション：

```
32 T：    temp ← target
          GPR[31] ← (PC＋8)31..1 ‖ ISA MODE
   T＋1：  PC ← PC31..28 ‖ temp ‖ 0^2
          ISA MODE toggle

64 T：    temp ← target
          GPR[31] ← (PC＋8)63..1 ‖ ISA MODE
   T＋1：  PC ← PC63..28 ‖ temp ‖ 0^2
          ISA MODE toggle
```

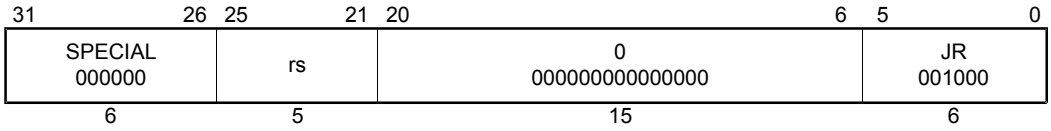
例　　外：

予約命令例外（MIPS16命令実行不可時）

保守／廃止

JR

Jump Register



命　令　形　式：

JR　rs

説　　明：

1命令の遅延付きで，汎用レジスタrsの内容のアドレスに無条件でジャンプします。

MIPS16命令実行可能な場合，1命令の遅延付きで，汎用レジスタrsの最下位ビットを0にクリアした値のアドレスに無条件でジャンプし，汎用レジスタrsの最下位ビットの内容をISAモード・ビット（内部）にセットします（μPD98502ではMIPS16モードをサポートしていません）。

32ビット長命令は，ワード境界に位置合わせされていなければならないので，MIPS16命令実行可能な場合，JR命令は下位2ビットが0であるターゲット・レジスタ（rs）を指定しなければなりません。下位2ビットが0でない場合，ジャンプ先の命令をフェッチしたときにアドレス例外が発生します。

オペレーション：

```
32, 64    T：     temp ← GPR[rs]
          T＋1：if MIPS16EN = 1 then
                     PC ← temp63..1 ‖ 0
                     ISA MODE ← temp0
                else
                     PC ← temp
                endif
```

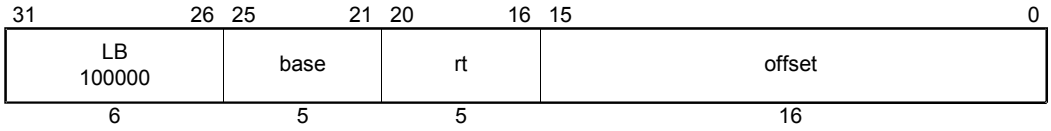
例　　外：

なし

保守／廃止

LB

Load Byte



命　令　形　式：

LB　rt，offset（base）

説　　明：

16ビットのoffsetを符号拡張して汎用レジスタbaseの内容に加算し，仮想アドレスを生成します。アドレス指定したメモリ位置のバイトの内容を符号拡張し，汎用レジスタrtにロードします。

オペレーション：

```
32　T：　vAddr ← ((offset15)^16 ‖ offset15..0)＋GPR[base]
　　（pAddr，uncached）← AddressTranslation（vAddr，DATA）
　　pAddr ← pAddrPSISE−1..3 ‖ （pAddr2..0 xor ReverseEndian^3）
　　mem ← LoadMemory（uncached，BYTE，pAddr，vAddr，DATA）
　　byte ← vAddr2..0 xor BigEndianCPU^3
　　GPR[rt]← (mem7+8∗byte)^24 ‖ mem7+8∗byte..8∗byte

64　T：　vAddr ← ((offset15)^48 ‖ offset15..0)＋GPR[base]
　　（pAddr，uncached）← AddressTranslation（vAddr，DATA）
　　pAddr ← pAddrPSISE−1..3 ‖ （pAddr2..0 xor ReverseEndian^3）
　　mem ← LoadMemory（uncached，BYTE，pAddr，vAddr，DATA）
　　byte ← vAddr2..0 xor BigEndianCPU^3
　　GPR[rt]← (mem7+8∗byte)^56 ‖ mem7+8∗byte..8∗byte
```

例　　外：

TLB不一致例外

TLB無効例外

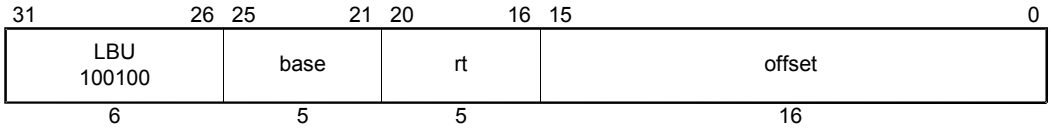
バス・エラー例外

アドレス・エラー例外

保守／廃止

LBU

Load Byte Unsigned



命　令　形　式：

LBU　rt，offset（base）

説　　明：

16ビットのoffsetを符号拡張して汎用レジスタbaseの内容に加算し，仮想アドレスを生成します。アドレス指定したメモリ位置のバイトの内容をゼロ拡張し，汎用レジスタrtにロードします。

オペレーション：

```
32   T:  vAddr ← ((offset15)^16 || offset15..0) + GPR[base]
         (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
         pAddr ← pAddrPSISE-1..3 || (pAddr2..0 xor ReverseEndian^3)
         mem ← LoadMemory (uncached, BYTE, pAddr, vAddr, DATA)
         byte ← vAddr2..0 xor BigEndianCPU^3
         GPR[rt] ← 0^24 || mem7+8*byte..8*byte

64   T:  vAddr ← ((offset15)^48 || offset15..0) + GPR[base]
         (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
         pAddr ← pAddrPSISE-1..3 || (pAddr2..0 xor ReverseEndian^3)
         mem ← LoadMemory (uncached, BYTE, pAddr, vAddr, DATA)
         byte ← vAddr2..0 xor BigEndianCPU^3
         GPR[rt] ← 0^56 || mem7+8*byte..8*byte
```

例　　外：

TLB不一致例外

TLB無効例外

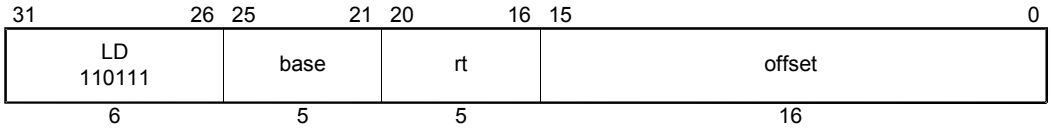
バス・エラー例外

アドレス・エラー例外

保守／廃止

LD

Load Doubleword



命　令　形　式：

LD　rt，offset（base）

説　　明：

16ビットのoffsetを符号拡張して汎用レジスタbaseの内容に加算し，仮想アドレスを生成します。アドレス指定したメモリ位置の64ビット・ダブル・ワードの内容を，汎用レジスタrtにロードします。

アドレスの下位3ビットが0でない場合，アドレス・エラー例外が発生します。

この演算は，64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モード時にこの命令を実行すると，予約命令例外が発生します。

オペレーション：

```
64　T：　vAddr ← ((offset15)^48 || offset15..0)+GPR[base]
         （pAddr，uncached）← AddressTranslation（vAddr，DATA）
         data ← LoadMemory（uncached，DOUBLEWORD，pAddr，vAddr，DATA）
         GPR[rt] ← data
```

例　　外：

TLB不一致例外

TLB無効例外

バス・エラー例外

アドレス・エラー例外

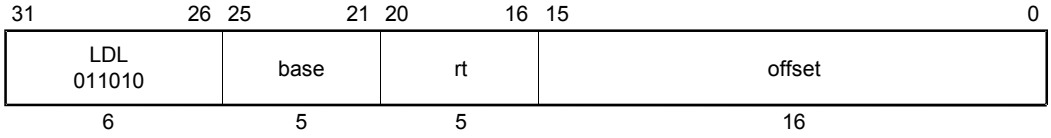
予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

LDL

Load Doubleword Left

（1/3）



命　令　形　式：

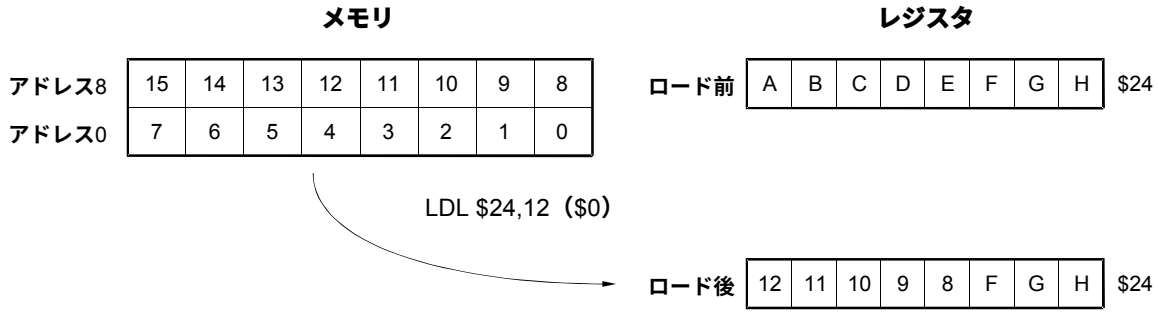
LDL　rt，offset（base）

説　　明：

この命令は，ダブル・ワード境界にないメモリ中のダブル・ワード・データを汎用レジスタrt**にロードするときに，**LDR**命令と組み合わせて使用します。**LDL**命令がデータの上位部分を，**LDR**命令がデータの下位部分をレジスタにロードします。**

16ビットのoffset**を符号拡張して汎用レジスタ**base**の内容に加算し，任意のバイトを指定できる仮想アドレスを生成します。アドレス指定したバイトを最上位バイトとするメモリ中のダブル・ワード・データについて，ターゲット・アドレスと同じダブル・ワード境界にあるデータのみをロードし，汎用レジスタ**rt**の上位部分に格納します。汎用レジスタ**rt**の残りの部分は変化しません。指定したアドレスによっては，ロードされるバイト数は**1 - 8**バイトの範囲で変わります。**

別の言い方をすれば，まずアドレス指定したバイトを汎用レジスタrt**の最上位バイトに格納し，同じダブル・ワード境界に引き続く下位のバイト・データがあれば，これを汎用レジスタ**rt**の次のバイトに格納する動作を繰り返します。残りの下位バイトは変化しません。**



保守／廃止

LDL

Load Doubleword Left

レジスタrtはバイパスされるので，同じレジスタrtをターゲットとする直前のロード命令と，続くLDL，LDR**命令との間に**NOP**命令は必要ありません。**

指定アドレスがダブル・ワード境界に位置していないことによるアドレス・エラー例外は発生しません。

この命令は，64**ビット・モード時と**32**ビット・カーネル・モード時に定義されます。**32**ビット・ユーザ／スーパバイザ・モード時にこの命令を実行すると，予約命令例外が発生します。**

オペレーション：

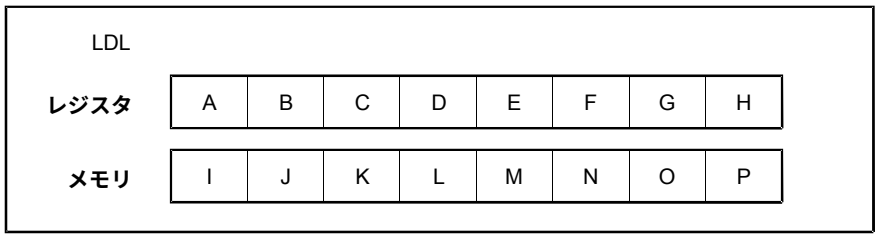
```
64   T：   vAddr ← ((offset15)^48 || offset15..0)＋GPR[base]
           （pAddr，uncached）← AddressTranslation（vAddr，DATA）
           pAddr ← pAddrPSIZE−1..3 ||（pAddr2..0 xor ReverseEndian^3）
           if BigEndianMem = 0 then
             pAddr ← pAddrPSIZE−1..3 || 0^3
           endif
           byte ← vAddr2..0 xor BigEndianCPU^3
           mem ← LoadMemory（uncached, byte, pAddr, vAddr, DATA）
           GPR[rt] ← mem7+8∗byte..0 || GPR[rt]55−8∗byte..0
```

保守／廃止

LDL

Load Doubleword Left

LDL命令に与えるアドレスとその結果（レジスタの各バイト）の関係を次に示します。



vAddr2..0	ビッグ・エンディアンCPU＝0				ビッグ・エンディアンCPU＝1			
	デスティネーション	タイプ	オフセット		デスティネーション	タイプ	オフセット	
			LEM	BEM			LEM	BEM
0	**PBCDEFGH**	0	0	7	**IJKLMNOP**	7	0	0
1	**OPCDEFGH**	1	0	6	**JKLMNOPH**	6	0	1
2	**NOPDEFGH**	2	0	5	**KLMNOPGH**	5	0	2
3	**MNOPEFGH**	3	0	4	**LMNOPFGH**	4	0	3
4	**LMNOPFGH**	4	0	3	**MNOPEFGH**	3	0	4
5	**KLMNOPGH**	5	0	2	**NOPDEFGH**	2	0	5
6	**JKLMNOPH**	6	0	1	**OPCDEFGH**	1	0	6
7	**IJKLMNOP**	7	0	0	**PBCDEFGH**	0	0	7

備考　タイプ　　：メモリに出力されるアクセス・タイプ（表 2-3　ロード／ストア命令に関するバイト指定参照）

オフセット：メモリに出力される pAddr2..0

LEM：**リトル・エンディアン・メモリ**（BigEndianMem = 0）

BEM：**ビッグ・エンディアン・メモリ**（BigEndianMem = 1）

例　　外：

TLB不一致例外

TLB無効例外

バス・エラー例外

アドレス・エラー例外

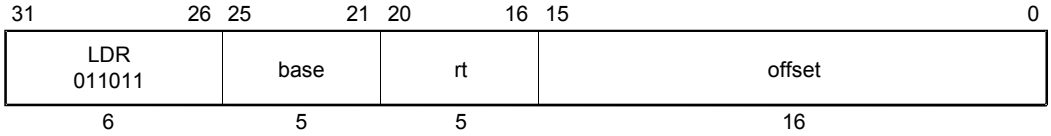
予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

LDR

Load Doubleword Right

（1/3）



命　令　形　式：

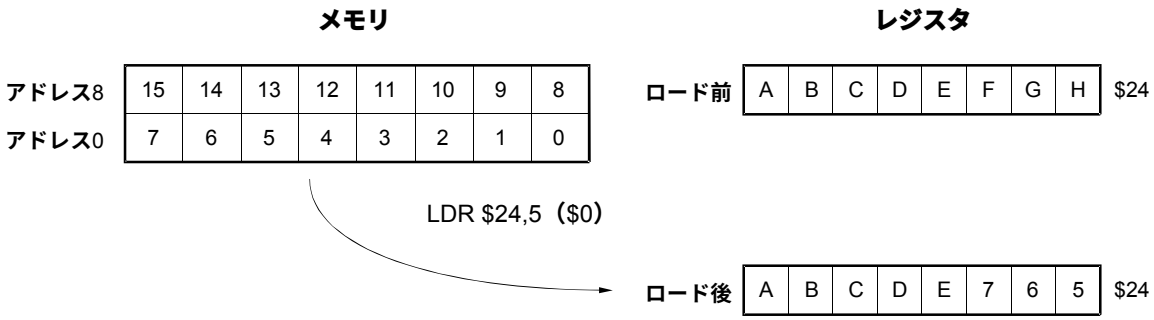
LDR　rt，offset（base）

説　　明：

この命令は，ダブル・ワード境界にないメモリ中のダブル・ワード・データを汎用レジスタrt**にロードするときに，**LDL**命令と組み合わせて使用します。**LDL**命令がデータの上位部分を，**LDR**命令がデータの下位部分をレジスタにロードします。**

16ビットのoffset**を符号拡張して汎用レジスタ**base**の内容に加算し，任意のバイトを指定できる仮想アドレスを生成します。アドレス指定したバイトを最下位バイトとするメモリ中のダブル・ワード・データについて，ターゲット・アドレスと同じダブル・ワード境界にあるデータのみをロードし，汎用レジスタ**rt**の下位部分に格納します。汎用レジスタ**rt**の残りの部分は変化しません。指定したアドレスによっては，ロードされるバイト数は**1 - 8**バイトの範囲で変わります。**

別の言い方をすれば，まずアドレス指定したバイトを汎用レジスタrt**の最下位バイトに格納し，同じダブル・ワード境界に引き続く上位のバイト・データがあれば，これを汎用レジスタ**rt**の次のバイトに格納する動作を繰り返します。残りの上位バイトは変化しません。**



保守／廃止

LDR

Load Doubleword Right

レジスタrtはバイパスされるので，同じレジスタrtをターゲットとする直前のロード命令と，続くLDL，LDR**命令との間に**NOP**命令は必要ありません。**

指定アドレスがダブル・ワード境界に位置していないことによるアドレス・エラー例外は発生しません。

この命令は，64**ビット・モード時と**32**ビット・カーネル・モード時に定義されます。**32**ビット・ユーザ／スーパバイザ・モード時にこの命令を実行すると，予約命令例外が発生します。**

オペレーション：

```
64   T： vAddr ← ((offset15)^48 ‖ offset15..0)＋GPR[base]
          （pAddr，uncached）← AddressTranslation（vAddr，DATA）
          pAddr ← pAddrPSIZE－1..3 ‖ （pAddr2..0 xor ReverseEndian^3）
          if BigEndianMem = 1 then
            pAddr ← pAddrPSIZE－1..3 ‖ 0^3
          endif
          byte ← vAddr2..0 xor BigEndianCPU^3
          mem ← LoadMemory（uncached, DOUBLEWORD - byte, pAddr, vAddr, DATA）
          GPR[rt] ← GPR[rt]63..64－8∗byte ‖ mem63..8∗byte
```

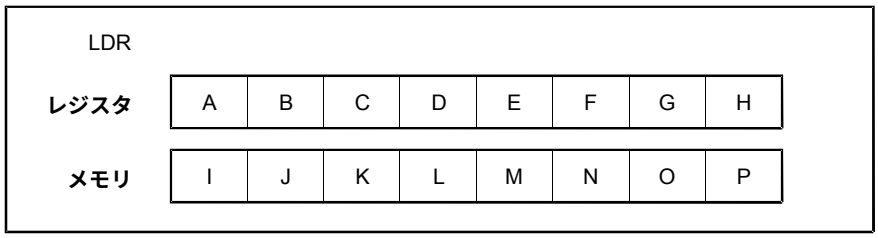
保守／廃止

LDR

Load Doubleword Right

（3/3）

LDR命令に与えるアドレスとその結果（レジスタの各バイト）の関係を次に示します。



vAddr2..0	ビッグ・エンディアンCPU＝0				ビッグ・エンディアンCPU＝1			
	デスティネーション	タイプ	オフセット		デスティネーション	タイプ	オフセット	
			LEM	BEM			LEM	BEM
0	**I J K L M N O P**	7	0	0	**A B C D E F G I**	0	7	0
1	**A I J K L M N O**	6	1	0	**A B C D E F I J**	1	6	0
2	**A B I J K L M N**	5	2	0	**A B C D E I J K**	2	5	0
3	**A B C I J K L M**	4	3	0	**A B C D I J K L**	3	4	0
4	**A B C D I J K L**	3	4	0	**A B C I J K L M**	4	3	0
5	**A B C D E I J K**	2	5	0	**A B I J K L M N**	5	2	0
6	**A B C D E F I J**	1	6	0	**A I J K L M N O**	6	1	0
7	**A B C D E F G I**	0	7	0	**I J K L M N O P**	7	0	0

備考　タイプ　　：メモリに出力されるアクセス・タイプ（表 2-3　ロード／ストア命令に関するバイト指定参照）

オフセット：メモリに出力される pAddr2..0

LEM：**リトル・エンディアン・メモリ**（BigEndianMem = 0）

BEM：**ビッグ・エンディアン・メモリ**（BigEndianMem = 1）

例　　外：

TLB**不一致例外**

TLB**無効例外**

バス・エラー例外

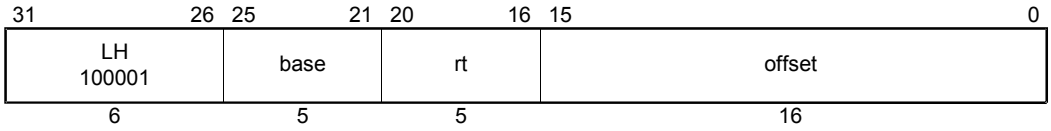
アドレス・エラー例外

予約命令例外（32**ビット・ユーザ／スーパバイザ・モード時）**

保守／廃止

LH

Load Halfword



命　令　形　式：

LH　rt，offset（base）

説　　明：

16ビットのoffsetを符号拡張して汎用レジスタbaseの内容に加算し，仮想アドレスを生成します。アドレス指定したメモリ位置のハーフ・ワードの内容を符号拡張し，汎用レジスタrtにロードします。

アドレスの最下位ビットが0でない場合，アドレス・エラー例外が発生します。

オペレーション：

```
32　T：　vAddr ← ((offset15)^16 || offset15..0)＋GPR[base]
          （pAddr，uncached）← AddressTranslation（vAddr，DATA）
          pAddr ← pAddrPSIZE−1..3 ||　(pAddr2..0 xor (ReverseEndian^2 || 0 ))
          mem ← LoadMemory（uncached，HALFWORD，pAddr，vAddr，DATA）
          byte ← vAddr2..0 xor（BigEndianCPU^2 || 0）
          GPR[rt] ← (mem15+8∗byte)^16 || mem15+8∗byte..8∗byte

64　T：　vAddr ← ((offset15)^48 || offset15..0)＋GPR[base]
          （pAddr，uncached）← AddressTranslation（vAddr，DATA）
          pAddr ← pAddrPSIZE−1..3 ||　(pAddr2..0 xor (ReverseEndian^2 || 0 ))
          mem ← LoadMemory（uncached，HALFWORD，pAddr，vAddr，DATA）
          byte ← vAddr2..0 xor（BigEndianCPU^2 || 0）
          GPR[rt] ← (mem15+8∗byte)^48 || mem15+8∗byte..8∗byte
```

例　　外：

TLB不一致例外

TLB無効例外

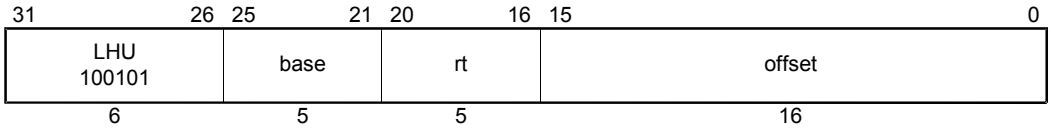
バス・エラー例外

アドレス・エラー例外

保守／廃止

LHU

Load Halfword Unsigned



命　令　形　式：

LHU　rt，offset（base）

説　　明：

16ビットのoffsetを符号拡張して汎用レジスタbaseの内容に加算し，仮想アドレスを生成します。アドレス指定したメモリ位置のハーフ・ワードの内容をゼロ拡張し，汎用レジスタrtにロードします。

アドレスの最下位ビットが0でない場合，アドレス・エラー例外が発生します。

オペレーション：

```
32　T：　vAddr ← ((offset15)^16 ‖ offset15..0)＋GPR[base]
          （pAddr，uncached）← AddressTranslation（vAddr，DATA）
          pAddr ← pAddrPSIZE−1..3 ‖ (pAddr2..0 xor (ReverseEndian2 ‖ 0))
          mem ←LoadMemory（uncached，HALFWORD，pAddr，vAddr，DATA）
          byte ← vAddr2..0 xor（BigEndianCPU^2 ‖ 0）
          GPR[rt]← 0^16 ‖ mem15+8∗byte..8∗byte

64　T：　vAddr ← ((offset15)^48 ‖ offset15..0)＋GPR[base]
          （pAddr，uncached）← AddressTranslation（vAddr，DATA）
          pAddr ← pAddrPSIZE−1..3 ‖ (pAddr2..0 xor (ReverseEndian^2 ‖ 0))
          mem ← LoadMemory（uncached，HALFWORD，pAddr，vAddr，DATA）
          byte ← vAddr2..0 xor（BigEndianCPU^2 ‖ 0）
          GPR[rt] ← 0^48 ‖ mem15+8∗byte..8∗byte
```

例　　外：

TLB不一致例外

TLB無効例外

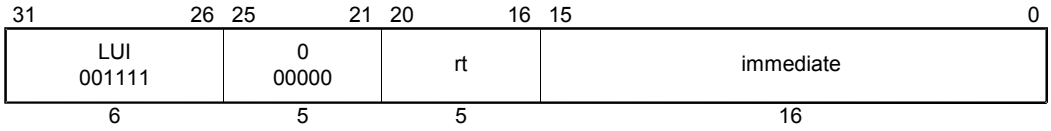
バス・エラー例外

アドレス・エラー例外

保守／廃止

LUI

Load Upper Immediate



命　令　形　式：

LUI　rt，immediate

説　　明：

16ビットのimmediateを16ビット左にシフトして，16ビットの0と結合します。結果は汎用レジスタrtに格納します。64ビット・モード時は，さらに64ビットに符号拡張します。

オペレーション：

```
32   T：   GPR[rt] ← immediate ‖ 0^16

64   T：   GPR[rt] ← (immediate15)^32 ‖ immediate ‖ 0^16
```

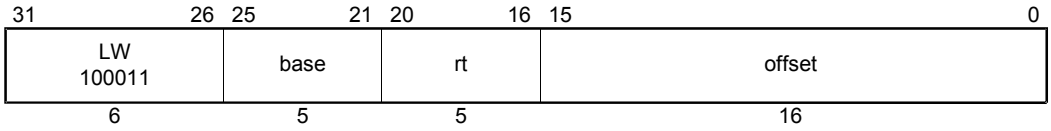
例　　外：

なし

保守／廃止

LW

Load Word



命　令　形　式：

LW　rt，offset（base）

説　　明：

16ビットのoffsetを符号拡張して汎用レジスタbaseの内容に加算し，仮想アドレスを生成します。アドレス指定したメモリ位置のワードの内容を，汎用レジスタrtにロードします。64ビット・モード時は，さらに64ビットに符号拡張します。

アドレスの下位2ビットのいずれかが0でない場合，アドレス・エラー例外が発生します。

オペレーション：

```
32　T：　vAddr ← ((offset15)^16 || offset15..0)+GPR[base]
　　（pAddr，uncached）← AddressTranslation（vAddr，DATA）
　　pAddr ← pAddrPSIZE−1..3 ||（pAddr2..0 xor (ReverseEndian || 0^2)）
　　mem ← LoadMemory（uncached，WORD，pAddr，vAddr，DATA）
　　byte ← vAddr2..0 xor（BigEndianCPU || 0^2）
　　GPR[rt] ← mem31+8∗byte..8∗byte

64　T：　vAddr ← ((offset15)^48 || offset15..0)+GPR[base]
　　（pAddr，uncached）← AddressTranslation（vAddr，DATA）
　　pAddr ← pAddrPSIZE−1..3 ||（pAddr2..0 xor (ReverseEndian || 0^2)）
　　mem ← LoadMemory（uncached，WORD，pAddr，vAddr，DATA）
　　byte ← vAddr2..0 xor（BigEndianCPU || 0^2）
　　GPR[rt] ← (mem31+8∗byte)^32 || mem31+8∗byte..8∗byte
```

例　　外：

TLB不一致例外

TLB無効例外

バス・エラー例外

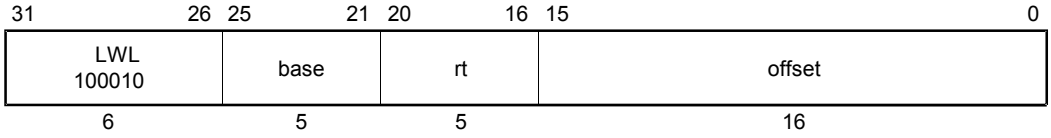
アドレス・エラー例外

保守／廃止

LWL

Load Word Left

（1/3）



命　令　形　式：

LWL　rt，offset（base）

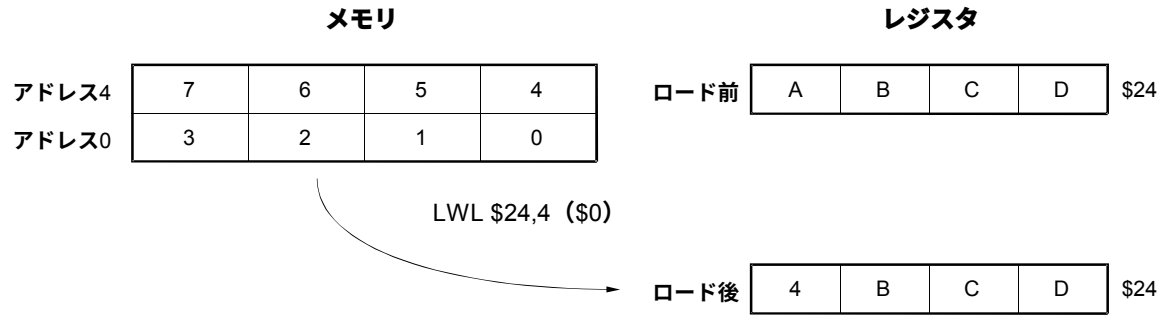
説　　明：

この命令は，ワード境界にないメモリ中のワード・データを汎用レジスタrt**にロードするときに，**LWR**命令と組み合わせて使用します。**LWL**命令がデータの上位部分を，**LWR**命令がデータの下位部分をレジスタにロードします。**

16ビットのoffset**を符号拡張して汎用レジスタ**base**の内容に加算し，任意のバイトを指定できる仮想アドレスを生成します。アドレス指定したバイトを最上位バイトとするメモリ中のワード･データについて,ターゲット・アドレスと同じワード境界にあるデータのみをロードし，汎用レジスタ**rt**の上位部分に格納します。汎用レジスタ**rt**の残りの部分は変化しません。指定したアドレスによっては，ロードされるバイト数は**1-4**バイトの範囲で変わります。**

別の言い方をすれば，まずアドレス指定したバイトを汎用レジスタrt**の最上位バイトに格納し，同じワード境界に引き続く下位のバイト・データがあれば，これを汎用レジスタ**rt**の次のバイトに格納する動作を繰り返します。**

残りの下位バイトは変化しません。



保守／廃止

LWL

Load Word Left

レジスタrtはバイパスされるので，同じレジスタrtをターゲットとする直前のロード命令と，続くLWL，LWR**命令との間に**NOP**命令は必要ありません。**

指定アドレスがワード境界に位置していないことによるアドレス・エラー例外は発生しません。

オペレーション：

```
32  T：  vAddr ← ((offset15)^16 || offset15..0)+GPR[base]
         (pAddr，uncached) ← AddressTranslation (vAddr，DATA)
         pAddr ← pAddrPSIZE−1..3 || (pAddr2..0 xor ReverseEndian^3)
         if BigEndianMem = 0 then
           pAddr ← pAddrPSIZE−1..2 || 0^2
         endif
         byte ← vAddr1..0 xor BigEndianCPU^2
         word ← vAddr2 xor BigEndianCPU
         mem ← LoadMemory (uncached，byte，pAddr，vAddr，DATA)
         temp ← mem32∗word+8∗byte+7..32∗word || GPR[rt]23−8∗byte..0
         GPR[rt] ← temp

64  T：  vAddr ← ((offset15)^48 || offset15..0)+GPR[base]
         (pAddr，uncached) ← AddressTranslation (vAddr，DATA)
         pAddr ← pAddrPSIZE−1..3 || (pAddr2..0 xor ReverseEndian^3)
         if BigEndianMem = 0 then
           pAddr ← pAddrPSIZE−1..2 || 0^2
         endif
         byte ← vAddr1..0 xor BigEndianCPU^2
         word ← vAddr2 xor BigEndianCPU
         mem ← LoadMemory (uncached，0 || byte，pAddr，vAddr，DATA)
         temp ← mem32∗word+8∗byte+7..32∗word || GPR[rt]23−8∗byte..0
         GPR[rt] ← (temp31)^32 || temp
```

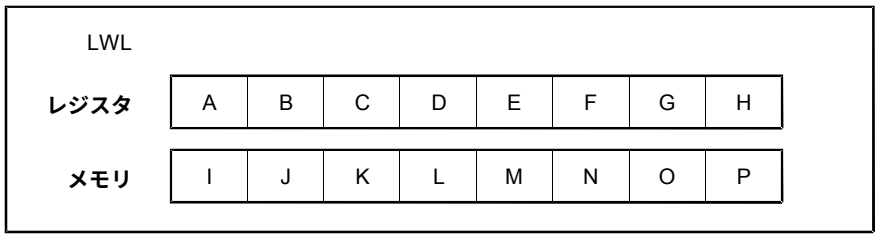
保守／廃止

LWL

Load Word Left

（3/3）

LWL命令に与えるアドレスとその結果（レジスタの各ワード）の関係を次に示します。



vAddr2..0	ビッグ・エンディアンCPU＝0				ビッグ・エンディアンCPU＝1			
	デスティネーション	タイプ	オフセット		デスティネーション	タイプ	オフセット	
			LEM	BEM			LEM	BEM
0	**S S S S P F G H**	0	0	7	**S S S S I J K L**	3	4	0
1	**S S S S O P G H**	1	0	6	**S S S S J K L H**	2	4	1
2	**S S S S N O P H**	2	0	5	**S S S S K L G H**	1	4	2
3	**S S S S M N O P**	3	0	4	**S S S S L F G H**	0	4	3
4	**S S S S L F G H**	0	4	3	**S S S S M N O P**	3	0	4
5	**S S S S K L G H**	1	4	2	**S S S S N O P H**	2	0	5
6	**S S S S J K L H**	2	4	1	**S S S S O P G H**	1	0	6
7	**S S S S I J K L**	3	4	0	**S S S S P F G H**	0	0	7

備考　**タイプ**　　**：メモリに出力されるアクセス・タイプ（表 2-3　ロード／ストア命令に関するバイト指定参照）**

オフセット：メモリに出力される pAddr2..0

LEM：**リトル・エンディアン・メモリ**（BigEndianMem = 0）

BEM：**ビッグ・エンディアン・メモリ**（BigEndianMem = 1）

S　　**：デスティネーションのビット** 31 **の符号拡張**

例　　外：

TLB**不一致例外**

TLB**無効例外**

バス・エラー例外

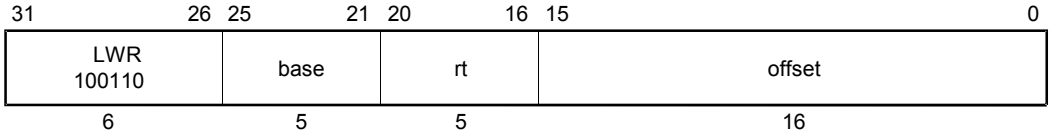
アドレス・エラー例外

保守／廃止

LWR

Load Word Right

（1/3）



命　令　形　式：

LWR　rt，offset（base）

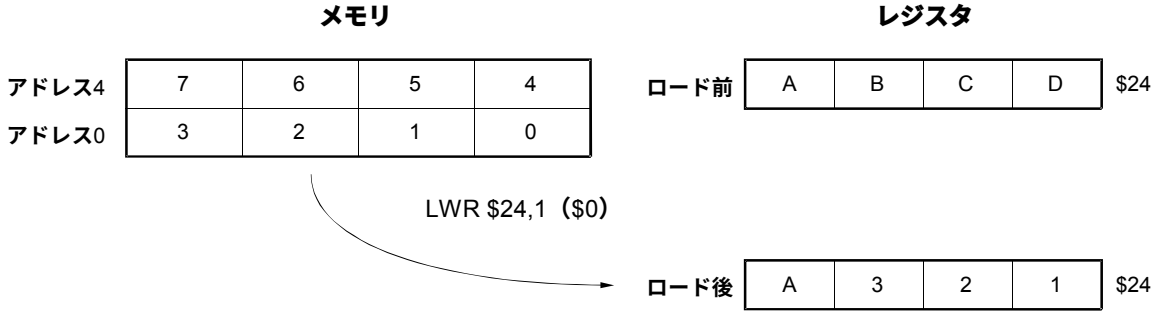
説　　明：

この命令は，ワード境界にないメモリ中のワード・データを汎用レジスタrt**にロードするときに，**LWL**命令と組み合わせて使用します。**LWL**命令がデータの上位部分を，**LWR**命令がデータの下位部分をレジスタにロードします。**

16ビットのoffset**を符号拡張して汎用レジスタ**base**の内容に加算し，任意のバイトを指定できる仮想アドレスを生成します。アドレス指定したバイトを最下位バイトとするメモリ中のワード･データについて,ターゲット・アドレスと同じワード境界にあるデータのみをロードし，汎用レジスタ**rt**の下位部分に格納します。汎用レジスタ**rt**の残りの部分は変化しません。指定したアドレスによっては，ロードされるバイト数は**1-4**バイトの範囲で変わります。**

別の言い方をすれば，まずアドレス指定したバイトを汎用レジスタrt**の最下位バイトに格納し，同じワード境界に引き続く上位のバイト・データがあれば，これを汎用レジスタ**rt**の次のバイトに格納する動作を繰り返します。**

残りの上位バイトは変化しません。



保守／廃止

LWR

Load Word Right

レジスタrt**はバイパスされるので，同じレジスタ**rt**をターゲットとする直前のロード命令と，続く**LWL，LWR**命令との間にNOP命令は必要ありません。**

指定アドレスがワード境界に位置していないことによるアドレス・エラー例外は発生しません。

オペレーション：

```
32  T：  vAddr ← ((offset15)^16 ‖ offset15..0)＋GPR[base]
         （pAddr，uncached）← AddressTranslation（vAddr，DATA）
         pAddr ← pAddrPSIZE−1..3 ‖ （pAddr2..0 xor ReverseEndian^3）
         if BigEndianMem = 1 then
           pAddr ← pAddrPSIZE−1..3 ‖ 0^3
         endif
         byte ← vAddr1..0 xor BigEndianCPU^2
         word ← vAddr2 xor BigEndianCPU
         mem ← LoadMemory（uncached，0 ‖ byte，pAddr，vAddr，DATA）
         temp ← GPR[rt]31..32−8∗byte ‖ mem31+32∗word ..32∗word+ 8∗byte
         GPR[rt] ← temp

64  T：  vAddr ← ((offset15)^48 ‖ offset15..0)＋GPR[base]
         （pAddr，uncached）← AddressTranslation（vAddr，DATA）
         pAddr ← pAddrPSIZE−1..3 ‖ （pAddr2..0 xor ReverseEndian^3）
         if BigEndianMem = 1 then
           pAddr ← pAddrPSIZE−1..3 ‖ 0^3
         endif
         byte ← vAddr1..0 xor BigEndianCPU^2
         word ← vAddr2 xor BigEndianCPU
         mem ← LoadMemory（uncached，WORD - byte，pAddr，vAddr，DATA）
         temp ← GPR[rt]31..32−8∗byte ‖ mem31+32∗word ..32∗word+ 8∗byte
         GPR[rt] ← (temp31)^32 ‖ temp
```

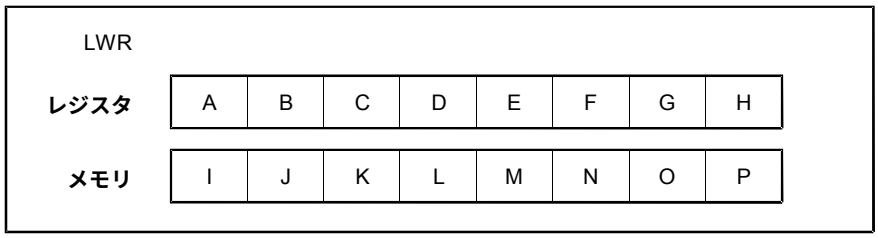
保守／廃止

LWR

Load Word Right

（3/3）

LWR命令に与えるアドレスとその結果（レジスタの各ワード）の関係を次に示します。



vAddr2..0	ビッグ・エンディアンCPU＝0				ビッグ・エンディアンCPU＝1			
	デスティネーション	タイプ	オフセット		デスティネーション	タイプ	オフセット	
			LEM	BEM			LEM	BEM
0	S S S S M N O P	3	0	4	S S S S E F G I	0	7	0
1	S S S S E M N O	2	1	4	S S S S E F I J	1	6	0
2	S S S S E F M N	1	2	4	S S S S E I J K	2	5	0
3	S S S S E F G M	0	3	4	S S S S I J K L	3	4	0
4	S S S S I J K L	3	4	0	S S S S E F G M	0	3	4
5	S S S S E I J K	2	5	0	S S S S E F M N	1	2	4
6	S S S S E F I J	1	6	0	S S S S E M N O	2	1	4
7	S S S S E F G I	0	7	0	S S S S M N O P	3	0	4

備考　タイプ　　：メモリに出力されるアクセス・タイプ（**表 2-3　ロード／ストア命令に関するバイト指定**参照）

オフセット：メモリに出力される pAddr2..0

LEM：リトル・エンディアン・メモリ（BigEndianMem = 0）

BEM：ビッグ・エンディアン・メモリ（BigEndianMem = 1）

S　　：デスティネーションのビット 31 の符号拡張

例　　外：

TLB不一致例外

TLB無効例外

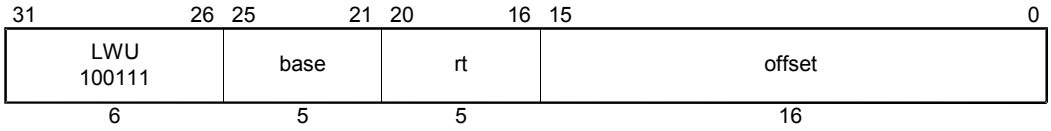
バス・エラー例外

アドレス・エラー例外

保守／廃止

LWU

Load Word Unsigned



命　令　形　式：

LWU　rt，offset（base）

説　　明：

16ビットのoffsetを符号拡張して汎用レジスタbaseの内容に加算し，仮想アドレスを生成します。アドレス指定したメモリ位置のワードの内容を，汎用レジスタrtにロードします。64ビット・モード時は，ロードしたワードをゼロ拡張します。

アドレスの下位2ビットのいずれかが0でない場合，アドレス・エラー例外が発生します。

この演算は，64ビット・モード時および32ビット・カーネル・モード時に定義されます。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

オペレーション：

```
32 T:  vAddr ← ((offset15)^16 || offset15..0) + GPR[base]
       (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
       pAddr ← pAddrPSIZE-1..3 || (pAddr2..0 xor (ReverseEndian || 0^2))
       mem ← LoadMemory (uncached, WORD, pAddr, vAddr, DATA)
       byte ← vAddr2..0 xor (BigEndianCPU || 0^2)
       GPR[rt] ← 0^32 || mem31+8*byte..8*byte

64 T:  vAddr ← ((offset15)^48 || offset15..0) + GPR[base]
       (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
       pAddr ← pAddrPSIZE-1..3 || (pAddr2..0 xor (ReverseEndian || 0^2))
       mem ← LoadMemory (uncached, WORD, pAddr, vAddr, DATA)
       byte ← vAddr2..0 xor (BigEndianCPU || 0^2)
       GPR[rt] ← 0^32 || mem31+8*byte..8*byte
```

例　　外：

TLB不一致例外

TLB無効例外

バス・エラー例外

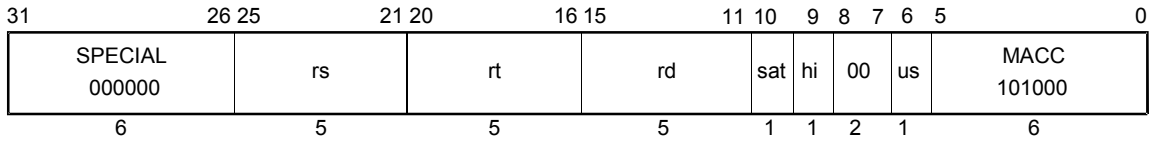
アドレス・エラー例外

予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

MACC

Multiply and Accumulate



命　令　形　式：

MACC	rd，rs，rt
MACCU	rd，rs，rt
MACCHI	rd，rs，rt
MACCHIU	rd，rs，rt
MACCS	rd，rs，rt
MACCUS	rd，rs，rt
MACCHIS	rd，rs，rt
MACCHIUS	rd，rs，rt

説　　明：

MACC**命令は，オペコードの**sat，hi，us**の各ビットの設定により，次のようにニモニックが異なります。**

ニモニック	sat	hi	us
MACC	0	0	0
MACCU	0	0	1
MACCHI	0	1	0
MACCHIU	0	1	1
MACCS	1	0	0
MACCUS	1	0	1
MACCHIS	1	1	0
MACCHIUS	1	1	1

また，飽和処理実行時（sat = 1**）と飽和処理非実行時（**sat = 0**）とで，オペランドの有効ビット数が異なります。**

・飽和処理実行時（sat = 1）：MACCS, MACCUS, MACCHIS, MACCHIUS命令

汎用レジスタrs**の内容と汎用レジスタ**rt**の内容を乗算します。両オペランドは，**us = 1**の場合，16ビットの符号なしデータとして扱われ，**us = 0**の場合，16ビットの符号付き整数として扱われます。オペランドのビット16以上のビットはソフトウエアで符号／ゼロ拡張されている必要があります。**

乗算の結果は，特殊レジスタHI**と**LO**をつなげた64ビットの値（ただし，下位32ビットのみ有効）と加算されます。加算は，**us = 1**の場合，32ビットの符号なしデータとして行われ，**us = 0**の場合，32ビットの符号付き整数として行われます。特殊レジスタ**HI**と**LO**をつなげた値のビット32以上のビットは符号／ゼロ拡張されている必要があります。**

保守／廃止

MACC

Multiply and Accumulate

加算の結果は，32ビットで飽和処理（下表参照）を行ったあと，特殊レジスタHI, LOにロードされます。hi = 1の場合，特殊レジスタHIにロードされるデータと同じものが汎用レジスタrdへもロードされます。hi = 0の場合，特殊レジスタLOにロードされるデータと同じものが汎用レジスタrdへもロードされます。オーバフロー例外は発生しません。

・飽和処理非実行時（sat = 0）：MACC, MACCU, MACCHI, MACCHIU命令

汎用レジスタrsの内容と汎用レジスタrtの内容を乗算します。両オペランドは，us = 1の場合，32ビットの符号なしデータとして扱われ，us = 0の場合，32ビットの符号付き整数として扱われます。オペランドのビット32以上のビットはソフトウエアで符号／ゼロ拡張されている必要があります。乗算の結果は，特殊レジスタHIとLOをつなげた64ビットの値と加算されます。加算は，us = 1の場合，64ビットの符号なしデータとして行われ，us = 0の場合，64ビットの符号付き整数として行われます。

加算の結果は，64ビットの下位ワードが特殊レジスタLOにロードされ，上位ワードが特殊レジスタHIにロードされます。hi = 1の場合，特殊レジスタHIにロードされるデータと同じものが汎用レジスタrdへもロードされます。hi = 0の場合，特殊レジスタLOにロードされるデータと同じものが汎用レジスタrdへもロードされます。オーバフロー例外は発生しません。

次に，us，satの設定と飽和処理時の格納値の対応，およびレジスタHIとLOを操作する命令とMACC命令の間に必要なハザード・サイクルを示します。

飽和処理時の格納値

us	sat	オーバフロー	アンダフロー
0	0	演算結果を そのまま格納	演算結果を そのまま格納
1	0	演算結果を そのまま格納	演算結果を そのまま格納
0	1	0x0000 0000 7FFF FFFF	0xFFFF FFFF 8000 0000
1	1	0xFFFF FFFF FFFF FFFF	なし

ハザード・サイクル数

命　　令	サイクル数
MULT, MULTU	1
DMULT, DMULTU	3
DIV, DIVU	36
DDIV, DDIVU	68
MFHI, MFLO	2
MTHI, MTLO	0
MACC	0
DMACC	0

保守／廃止

MACC

Multiply and Accumulate

オペレーション：

```
32, sat=0, hi=0, us=0（MACC命令）
        T:    temp1 ← GPR[rs] * GPR[rt]
              temp2 ← temp1 + (HI || LO)
              LO ← temp2[63..32]
              HI ← temp2[31..0]
              GPR[rd] ← LO
32, sat=0, hi=0, us=1（MACCU命令）
        T:    temp1 ← (0 || GPR[rs]) * (0 || GPR[rt])
              temp2 ← temp1 + ((0 || HI) || (0 || LO))
              LO ← temp2[63..32]
              HI ← temp2[31..0]
              GPR[rd] ← LO
32, sat=0, hi=1, us=0（MACCHI命令）
        T:    temp1 ← GPR[rs] * GPR[rt]
              temp2 ← temp1 + (HI || LO)
              LO ← temp2[63..32]
              HI ← temp2[31..0]
              GPR[rd] ← HI
32, sat=0, hi=1, us=1（MACCHIU命令）
        T:    temp1 ← (0 || GPR[rs]) * (0 || GPR[rt])
              temp2 ← temp1 + ((0 || HI) || (0 || LO))
              LO ← temp2[63..32]
              HI ← temp2[31..0]
              GPR[rd] ← HI
32, sat=1, hi=0, us=0（MACCS命令）
        T:    temp1 ← GPR[rs] * GPR[rt]
              temp2 ← saturation(temp1 + (HI || LO))
              LO ← temp2[63..32]
              HI ← temp2[31..0]
              GPR[rd] ← LO
32, sat=1, hi=0, us=1（MACCUS命令）
        T:    temp1 ← (0 || GPR[rs]) * (0 || GPR[rt])
              temp2 ← saturation(temp1 + ((0 || HI) || (0 || LO)))
              LO ← temp2[63..32]
              HI ← temp2[31..0]
              GPR[rd] ← LO
```

保守／廃止

MACC

Multiply and Accumulate

（4/5）

32, sat=1, hi=1, us=0（MACCHIS命令）

T: temp1 ← GPR[rs] ＊ GPR[rt]
temp2 ← saturation(temp1 + (HI || LO))
LO ← $temp2_{63..32}$
HI ← $temp2_{31..0}$
GPR[rd] ← HI

32, sat=1, hi=1, us=1（MACCHIUS命令）

T: temp1 ← (0 || GPR[rs]) ＊ (0 || GPR[rt])
temp2 ← saturation(temp1 + ((0 || HI) || (0 || LO)))
LO ← $temp2_{63..32}$
HI ← $temp2_{31..0}$
GPR[rd] ← HI

64, sat=0, hi=0, us=0（MACC命令）

T: temp1 ← $((GPR[rs]_{31})^{32}$ || GPR[rs]) ＊ $((GPR[rt]_{31})^{32}$ || GPR[rt])
temp2 ← temp1 + ($HI_{31..0}$ || $LO_{31..0}$)
LO ← $((temp2_{63})^{32}$ || $temp2_{63..32}$)
HI ← $((temp2_{31})^{32}$ || $temp2_{31..0}$)
GPR[rd] ← LO

64, sat=0, hi=0, us=1（MACCU命令）

T: temp1 ← $(0^{32}$ || GPR[rs]) ＊ $(0^{32}$ || GPR[rt])
temp2 ← temp1 + ($HI_{31..0}$ || $LO_{31..0}$)
LO ← $((temp2_{63})^{32}$ || $temp2_{63..32}$)
HI ← $((temp2_{31})^{32}$ || $temp2_{31..0}$)
GPR[rd] ← LO

64, sat=0, hi=1, us=0（MACCHI命令）

T: temp1 ← $((GPR[rs]_{31})^{32}$ || GPR[rs]) ＊ $((GPR[rt]_{31})^{32}$ || GPR[rt])
temp2 ← temp1 + ($HI_{31..0}$ || $LO_{31..0}$)
LO ← $((temp2_{63})^{32}$ || $temp2_{63..32}$)
HI ← $((temp2_{31})^{32}$ || $temp2_{31..0}$)
GPR[rd] ← HI

64, sat=0, hi=1, us=1（MACCHIU命令）

T: temp1 ← $(0^{32}$ || GPR[rs]) ＊ $(0^{32}$ || GPR[rt])
temp2 ← temp1 + ($HI_{31..0}$ || $LO_{31..0}$)
LO ← $((temp2_{63})^{32}$ || $temp2_{63..32}$)
HI ← $((temp2_{31})^{32}$ || $temp2_{31..0}$)
GPR[rd] ← HI

保守／廃止

MACC

Multiply and Accumulate

64, sat=1, hi=0, us=0（MACCS命令）

T: temp1 ← $((GPR[rs]_{31})^{32} \| GPR[rs]) * ((GPR[rt]_{31})^{32} \| GPR[rt])$

temp2 ← saturation(temp1 + $(HI_{31..0} \| LO_{31..0})$)

LO ← $((temp2_{63})^{32} \| temp2_{63..32})$

HI ← $((temp2_{31})^{32} \| temp2_{31..0})$

GPR[rd] ← LO

64, sat=1, hi=0, us=1（MACCUS命令）

T: temp1 ← $(0^{32} \| GPR[rs]) * (0^{32} \| GPR[rt])$

temp2 ← saturation(temp1 + $(HI_{31..0} \| LO_{31..0})$)

LO ← $((temp2_{63})^{32} \| temp2_{63..32})$

HI ← $((temp2_{31})^{32} \| temp2_{31..0})$

GPR[rd] ← LO

64, sat=1, hi=1, us=0（MACCHIS命令）

T: temp1 ← $((GPR[rs]_{31})^{32} \| GPR[rs]) * ((GPR[rt]_{31})^{32} \| GPR[rt])$

temp2 ← saturation(temp1 + $(HI_{31..0} \| LO_{31..0})$)

LO ← $((temp2_{63})^{32} \| temp2_{63..32})$

HI ← $((temp2_{31})^{32} \| temp2_{31..0})$

GPR[rd] ← HI

64, sat=1, hi=1, us=1（MACCHIUS命令）

T: temp1 ← $(0^{32} \| GPR[rs]) * (0^{32} \| GPR[rt])$

temp2 ← saturation(temp1 + $(HI_{31..0} \| LO_{31..0})$)

LO ← $((temp2_{63})^{32} \| temp2_{63..32})$

HI ← $((temp2_{31})^{32} \| temp2_{31..0})$

GPR[rd] ← HI

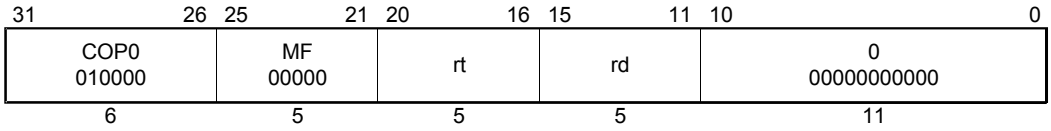
例　　外：

なし

保守／廃止

MFC0

Move From System Control Coprocessor



命　令　形　式：

MFC0　rt，rd

説　　明：

CP0の汎用レジスタrdの内容を，汎用レジスタrtにロードします。

オペレーション：

```
32  T：    data ← CPR[0，rd]
       T＋1： GPR[rt] ← data

64  T：    data ← CPR[0，rd]
       T＋1： GPR[rt] ← (data31)^32 ‖ data31..0
```

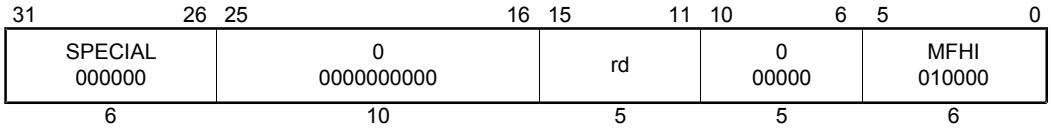
例　　外：

コプロセッサ使用不可例外（CP0が禁止されている場合で64／32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

MFHI

Move From HI



命　令　形　式：

MFHI　rd

説　　明：

特殊レジスタHI**の内容を汎用レジスタ**rd**にロードします。**

割り込みが発生した場合に正しい動作を保証するために，MFHI**命令に続く2つの命令には，**HI**レジスタを変更するような命令（**MULT, MULTU, DIV, DIVU, MTHI, DMULT, DMULTU, DDIV, DDIVU**）を使用しないでください。**

オペレーション：

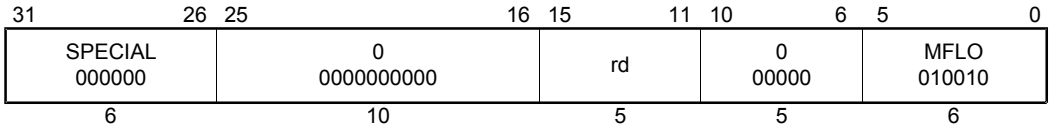
```
32，64　T：　GPR[rd] ← HI
```

例　　外：

なし

MFLO

Move From LO



命　令　形　式：

MFLO　rd

説　　明：

特殊レジスタLO**の内容を汎用レジスタ**rd**にロードします。**

割り込みが発生した場合に正しい動作を保証するために，MFLO**命令に続く2つの命令には，**LO**レジスタを変更する命令（**MULT, MULTU, DIV, DIVU, MTLO, DMULT, DMULTU, DDIV, DDIVU**）を使用しないでください。**

オペレーション：

```
32，64　T：　GPR[rd] ← LO
```

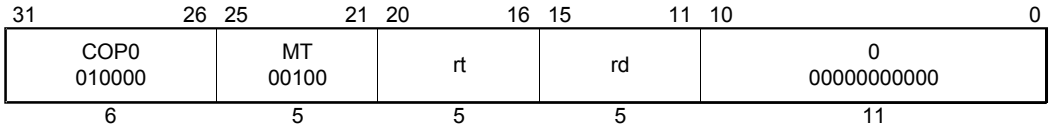
例　　外：

なし

保守／廃止

MTC0

Move To System Control Coprocessor



命　令　形　式：

MTC0　rt，rd

説　　明：

汎用レジスタrt**の内容を，**CP0**の汎用レジスタ**rd**にロードします。**

TLB**の内容はこの命令の実行により変更されることがあるので，この命令の直前および直後のロード命令とストア命令，および**TLB**操作命令の動作は未定義になります。**

この命令で操作するレジスタをその前後の命令で使用する場合は，付録B　VR4120Aコプロセッサ0ハザードを参照して，適切な位置に命令を配置してください。

オペレーション：

```
32，64　T：　　data ← GPR[rt]
　　T＋1：　CPR[0，rd] ← data
```

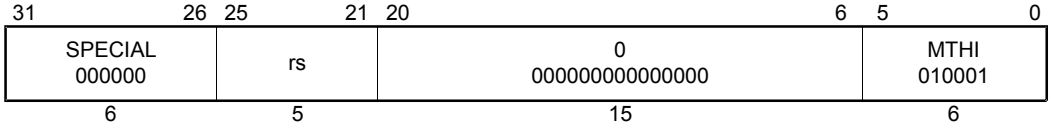
例　　外：

コプロセッサ使用不可例外（CP0**が禁止されている場合で**64/32**ビット・ユーザ／スーパバイザ・モード時）**

保守／廃止

MTHI

Move To HI



命　令　形　式：

MTHI　rs

説　　明：

汎用レジスタrs**の内容を特殊レジスタ**HI**にロードします。**

MULT，MULTU，DIV，DIVU**命令に続いて**MTHI**命令を実行した場合は正常に動作します。しかし，そのあと** MTHI**命令に続いて**MFLO，MFHI，MTLO，MTHI**命令を実行した場合，特殊レジスタ**LO**の内容は不定です。**

オペレーション：

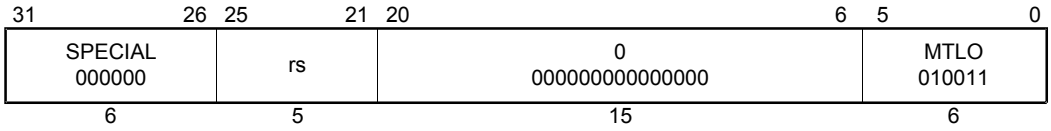
```
32，64  T－2：    HI ← undefined
        T－1：    HI ← undefined
        T：       HI ← GPR[rs]
```

例　　外：

なし

MTLO

Move To LO



命　令　形　式：

MTLO　rs

説　　明：

汎用レジスタrs**の内容を特殊レジスタ**LO**にロードします。**

MULT，MULTU，DIV，DIVU**命令に続いて**MTLO**命令を実行した場合は正常に動作します。しかし，そのあと**MTLO**命令に続いて**MFLO，MFHI，MTLO，MTHI**命令を実行した場合，特殊レジスタ**HI**の内容は不定です。**

オペレーション：

```
32，64  T－2：    LO ← undefined
        T－1：    LO ← undefined
        T：       LO ← GPR[rs]
```

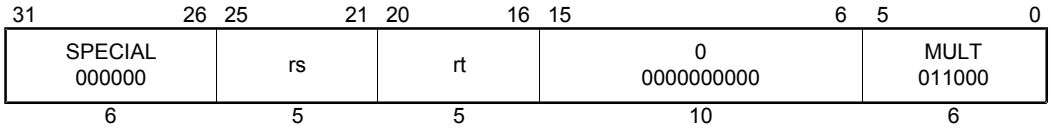
例　　外：

なし

保守／廃止

MULT

Multiply



命　令　形　式：

MULT　rs，rt

説　　明：

汎用レジスタrs**の内容と汎用レジスタ**rt**の内容を乗算します。両オペランドを**32**ビットの符号付き整数として扱います。整数オーバフロー例外は発生しません。**

64ビット・モードでは，オペランドは32ビット値を符号拡張した値でなければなりません。

結果は，ダブル・ワードの下位ワードを特殊レジスタLO**にロードし，上位ワードを特殊レジスタ**HI**にロードします。64ビット・モードでは，それぞれの結果を符号拡張して格納します。**

直前の2つの命令のいずれかがMFHI**命令または**MFLO**命令の場合，これらの転送命令の実行結果は未定義になります。正しい結果を得るには，**MFHI**，**MFLO**命令と**MULT**命令の間に，2つ以上のほかの命令を入れてください。**

オペレーション：

```
32   T－2：   LO ← undefined
              HI ← undefined
         T－1：   LO ← undefined
              HI ← undefined
         T：      t ← GPR[rs] ＊ GPR[rt]
              LO ← t31..0
              HI ← t63..32

64   T－2：   LO ← undefined
              HI ← undefined
         T－1：   LO ← undefined
              HI ← undefined
         T：      t ← GPR[rs]31..0 ＊ GPR[rt]31..0
              LO ← (t31)32 ‖ t31..0
              HI ← (t63)32 ‖ t63..32
```

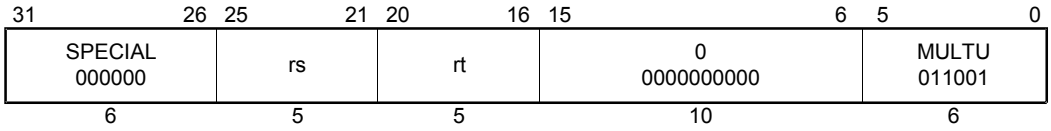
例　　外：

なし

保守／廃止

MULTU

Multiply Unsigned



命　令　形　式：

MULTU　rs，rt

説　　明：

汎用レジスタrs**の内容と汎用レジスタ**rt**の内容を乗算します。両オペランドを**32**ビットの符号なしの値として扱います。整数オーバフロー例外は発生しません。**

64**ビット・モードでは，オペランドは**32**ビット値を符号拡張した値でなければなりません。**

結果は，ダブル・ワードの下位ワードを特殊レジスタLO**にロードし，上位ワードを特殊レジスタ**HI**にロードします。**64**ビット・モードでは，それぞれの結果を符号拡張して格納します。**

直前の2**つの命令のいずれかが**MFHI**命令または**MFLO**命令の場合，これらの転送命令の実行結果は未定義になります。正しい結果を得るには，**MFHI**，**MFLO**命令と**MULTU**命令の間に，**2**つ以上のほかの命令を入れてください。**

オペレーション：

```
32   T－2：   LO ← undefined
              HI ← undefined
     T－1：   LO ← undefined
              HI ← undefined
     T：      t ← (0 || GPR[rs]) * (0 || GPR[rt])
              LO ← t31..0
              HI ← t63..32

64   T－2：   LO ← undefined
              HI ← undefined
     T－1：   LO ← undefined
              HI ← undefined
     T：      t ← (0 || GPR[rs]31..0) * (0 || GPR[rt]31..0)
              LO ← (t31)^32 || t31..0
              HI ← (t63)^32 || t63..32
```

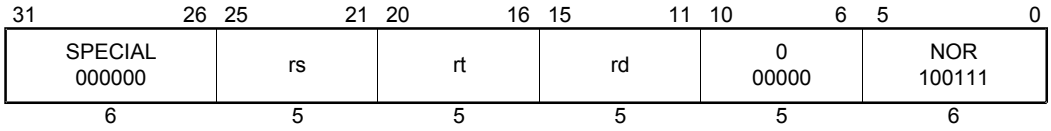
例　　外：

なし

保守／廃止

NOR

Nor



命　令　形　式：

NOR　rd，rs，rt

説　　明：

汎用レジスタrsの内容と汎用レジスタrtの内容の，ビットごとの否定論理和演算を行います。結果は汎用レジスタrdに格納します。

オペレーション：

```
32，64　T：　GPR[rd] ← GPR[rs] nor GPR[rt]
```

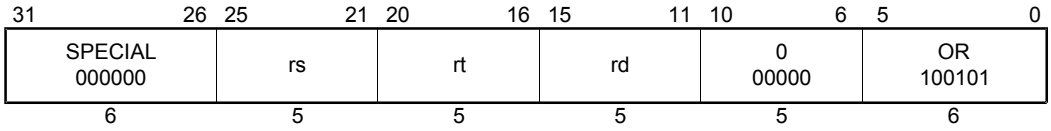
例　　外：

なし

保守／廃止

OR

Or



命　令　形　式：

OR　rd，rs，rt

説　　明：

汎用レジスタrsの内容と汎用レジスタrtの内容の，ビットごとの論理和演算を行います。結果は汎用レジスタrdに格納します。

オペレーション：

```
32，64　T：　GPR[rd] ← GPR[rs] or GPR[rt]
```

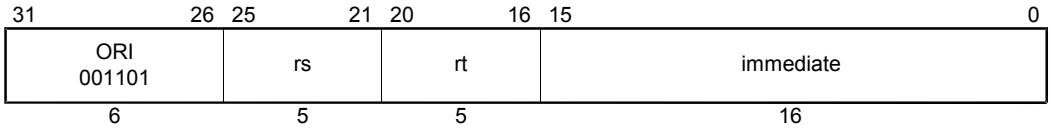
例　　外：

なし

保守／廃止

ORI

Or Immediate



命　令　形　式：

ORI　rt，rs，immediate

説　　明：

ゼロ拡張した16ビットのimmediateと汎用レジスタrsの内容の，ビットごとの論理和演算を行います。結果は汎用レジスタrtに格納します。

オペレーション：

```
32　T：　GPR[rt] ← GPR[rs] 31..16 ‖ （immediate or GPR[rs]15..0）

64　T：　GPR[rt] ← GPR[rs] 63..16 ‖ （immediate or GPR[rs]15..0）
```

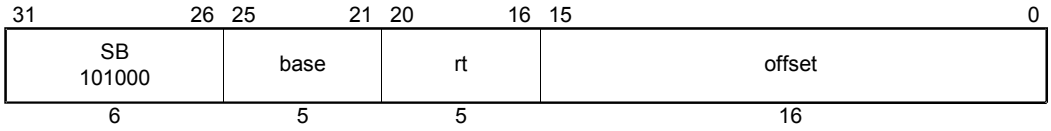
例　　外：

なし

保守／廃止

SB

Store Byte



命　令　形　式：

SB　rt，offset（base）

説　　明：

16ビットのoffsetを符号拡張して汎用レジスタbaseの内容に加算し，仮想アドレスを生成します。レジスタrtの最下位バイトを，アドレス指定したメモリにストアします。

オペレーション：

```
32　T：　vAddr ← ((offset15)^16 ‖ offset15..0 )＋GPR[base]
　　（pAddr，uncached）← AddressTranslation（vAddr，DATA）
　　pAddr ← pAddrPSIZE−1..3 ‖ （pAddr2..0 xor ReverseEndian^3）
　　byte ← vAddr2..0 xor BigEndianCPU^3
　　data ← GPR[rt]63−8∗byte..0 ‖ 0^8∗byte
　　StoreMemory（uncached，BYTE，data，pAddr，vAddr，DATA）

64　T：　vAddr ← ((offset15)^48 ‖ offset15..0)＋GPR[base]
　　（pAddr，uncached）← AddressTranslation（vAddr，DATA）
　　pAddr ← pAddrPSIZE−1..3 ‖ （pAddr2..0 xor ReverseEndian^3）
　　byte ← vAddr2..0 xor BigEndianCPU^3
　　data ← GPR[rt]63−8∗byte..0 ‖ 0^8∗byte
　　StoreMemory（uncached，BYTE，data，pAddr，vAddr，DATA）
```

例　　外：

TLB不一致例外

TLB無効例外

TLB変更例外

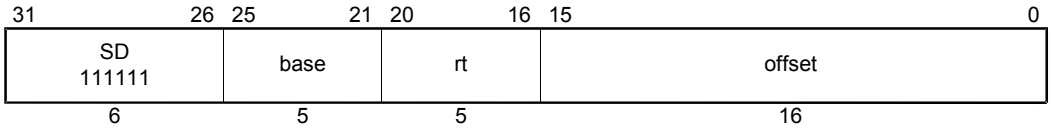
バス・エラー例外

アドレス・エラー例外

保守／廃止

SD

Store Doubleword



命　令　形　式：

SD　rt，offset（base）

説　　明：

16ビットのoffsetを符号拡張して汎用レジスタbaseの内容に加算し，仮想アドレスを生成します。汎用レジスタrtの内容を，アドレス指定したメモリにストアします。

アドレスの下位3ビットが0でない場合，アドレス・エラー例外が発生します。

この演算は，64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モード時にこの命令を実行すると，予約命令例外が発生します。

オペレーション：

```
64　T：　vAddr ← ((offset15)^48 || offset15..0)＋GPR[base]
         （pAddr，uncached）← AddressTranslation（vAddr，DATA）
         data ← GPR[rt]
         StoreMemory（uncached，DOUBLEWORD，data，pAddr，vAddr，DATA）
```

例　　外：

TLB不一致例外

TLB無効例外

TLB変更例外

バス・エラー例外

アドレス・エラー例外

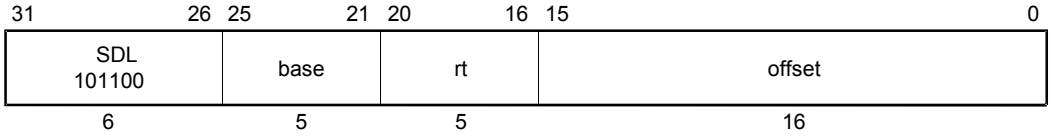
予約命令例外（32ビット・ユーザ／スーパバイザ・モード時）

保守／廃止

SDL

Store Doubleword Left

（1/3）



命　令　形　式：

SDL　rt，offset（base）

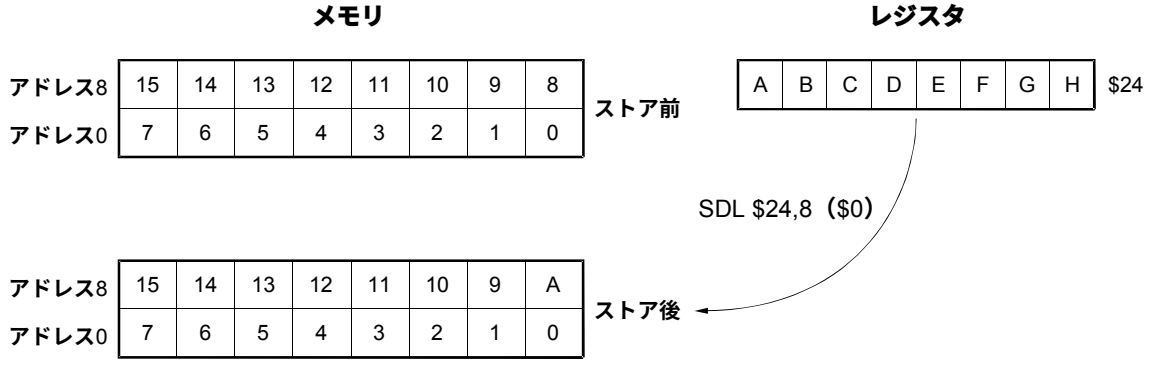
説　　明：

この命令は，レジスタ中のダブル・ワード・データを，ダブル・ワード境界にないメモリ中のダブル・ワードにストアするときに，SDR命令と組み合わせて使用します。SDL命令がデータの上位部分を，SDR命令がデータの下位部分をメモリにストアします。

16ビットのoffsetを符号拡張して汎用レジスタbaseの内容と加算し，仮想アドレスを生成します。アドレス指定したバイトを最上位バイトとするメモリ中のダブル・ワード・データについて，ターゲット・アドレスと同じダブル・ワード境界にあるメモリに，汎用レジスタrtの上位部分をストアします。

指定したアドレスによっては，ストアされるバイト数は1-8バイトの範囲で変わります。

つまり，まず汎用レジスタrtの最上位バイトをアドレス指定したメモリ内バイトにストアし，同じダブル・ワード境界に引き続く下位のバイト・データがあれば，これをメモリの次のバイトにストアする動作を繰り返します。



保守／廃止

SDL

Store Doubleword Left

指定アドレスがダブル・ワード境界に位置していないことによるアドレス・エラー例外は発生しません。

この命令は，64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

オペレーション：

```
64　T：　vAddr ← ((offset15)^48 || offset15..0)+GPR[base]
          (pAddr，uncached) ← AddressTranslation (vAddr，DATA)
          pAddr ← pAddrPSIZE−1..3 ||  (pAddr2..0 xor ReverseEndian^3)
          if BigEndianMem = 0 then
            pAddr ← pAddrPSIZE−1..3  || 0^3
          endif
          byte ← vAddr2..0 xor BigEndianCPU^3
          data ← 0^56−8∗byte || GPR[rt]63..56−8∗byte
          StoreMemory (uncached，byte，data，pAddr，vAddr，DATA)
```

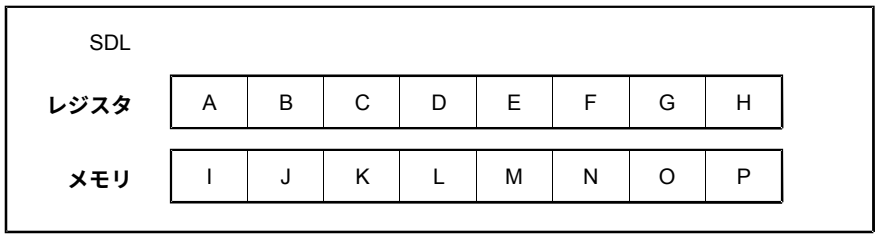
保守／廃止

SDL

Store Doubleword Left

（3/3）

SDL命令に与えるレジスタの内容とその結果（メモリのダブル・ワードの各バイト）の関係を次に示します。



vAddr2..0	ビッグ・エンディアンCPU＝0				ビッグ・エンディアンCPU＝1			
	デスティネーション	タイプ	オフセット		デスティネーション	タイプ	オフセット	
			LEM	BEM			LEM	BEM
0	I J K L M N O A	0	0	7	A B C D E F G H	7	0	0
1	I J K L M N A B	1	0	6	I A B C D E F G	6	0	1
2	I J K L M A B C	2	0	5	I J A B C D E F	5	0	2
3	I J K L A B C D	3	0	4	I J K A B C D E	4	0	3
4	I J K A B C D E	4	0	3	I J K L A B C D	3	0	4
5	I J A B C D E F	5	0	2	I J K L M A B C	2	0	5
6	I A B C D E F G	6	0	1	I J K L M N A B	1	0	6
7	A B C D E F G H	7	0	0	I J K L M N O A	0	0	7

備考　タイプ　　：メモリに出力されるアクセス・タイプ（表 2-3　ロード／ストア命令に関するバイト指定参照）

オフセット：メモリに出力される pAddr2..0

LEM：**リトル・エンディアン・メモリ**（BigEndianMem = 0）

BEM：**ビッグ・エンディアン・メモリ**（BigEndianMem = 1）

例　　外：

TLB**不一致例外**

TLB**無効例外**

TLB**変更例外**

バス・エラー例外

アドレス・エラー例外

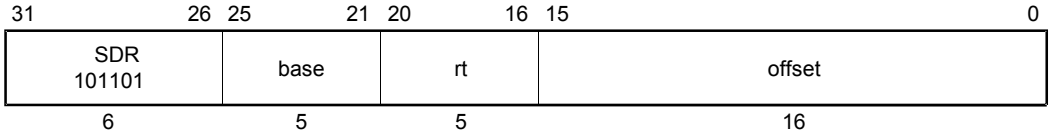
予約命令例外（32**ビット・ユーザ／スーパバイザ・モード時**）

保守／廃止

SDR

Store Doubleword Right

（1/3）



命　令　形　式：

SDR　rt，offset（base）

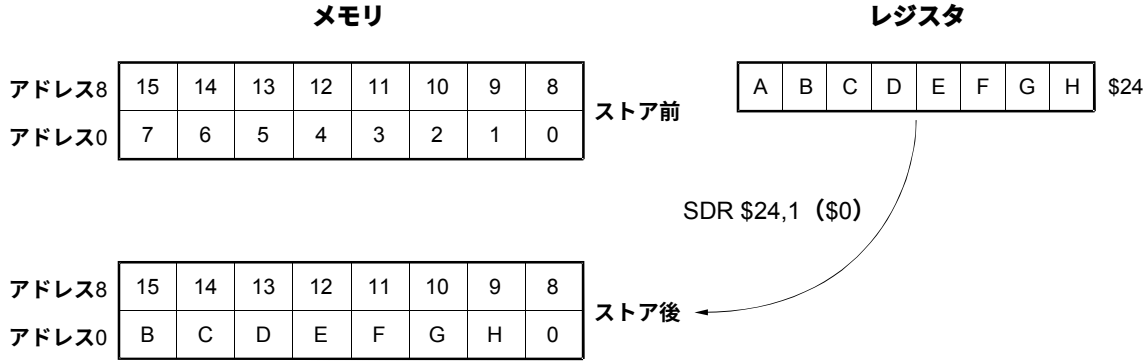
説　　明：

この命令は，レジスタ中のダブル・ワード・データを，ダブル・ワード境界にないメモリ中のダブル・ワードにストアするときに，SDL命令と組み合わせて使用します。SDL命令がデータの上位部分を，SDR命令がデータの下位部分をメモリにストアします。

16ビットのoffsetを符号拡張して汎用レジスタbaseの内容と加算し，仮想アドレスを生成します。アドレス指定したバイトを最下位バイトとするメモリ中のダブル・ワード・データについて，ターゲット・アドレスと同じダブル・ワード境界にあるメモリに，汎用レジスタrtの下位部分をストアします。

指定したアドレスによっては，ストアされるバイト数は1-8バイトの範囲で変わります。

つまり，まず汎用レジスタrtの最下位バイトをアドレス指定したメモリ内バイトにストアし，同じダブル・ワード境界に引き続く上位のバイトがあれば，これをメモリの次のバイトにストアする動作を繰り返します。



保守／廃止

SDR

Store Doubleword Right

指定アドレスがダブル・ワード境界に位置していないことによるアドレス・エラー例外は発生しません。

この演算は，64ビット・モード時と32ビット・カーネル・モード時に定義されています。32ビット・ユーザ／スーパバイザ・モードでこの命令を実行すると，予約命令例外が発生します。

オペレーション：

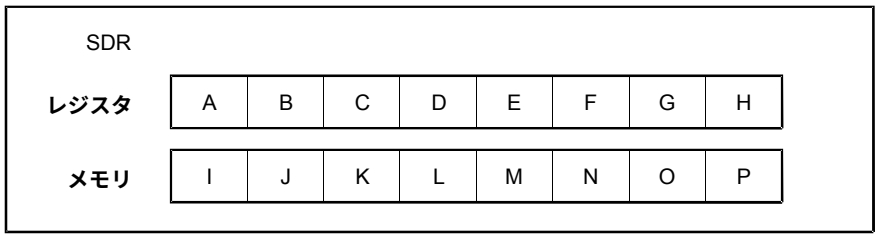
```
64　T：　vAddr ← ((offset15)^48 || offset15..0) + GPR[base]
         (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
         pAddr ← pAddrPSIZE−1..3 || (pAddr2..0 xor ReverseEndian^3)
         if BigEndianMem = 0 then
           pAddr ← pAddrPSIZE−1..3 || 0^3
         endif
         byte ← vAddr2..0 xor BigEndianCPU^3
         data ← GPR[rt]63−8∗byte || 0^8∗byte
         StoreMemory (uncached, DOUBLEWORD-byte, data, pAddr, vAddr, DATA)
```

保守／廃止

SDR

Store Doubleword Right

SDR命令に与えるレジスタの内容とその結果（メモリのダブル・ワードの各バイト）の関係を次に示します。



vAddr2..0	ビッグ・エンディアンCPU＝0				ビッグ・エンディアンCPU＝1			
	デスティネーション	タイプ	オフセット		デスティネーション	タイプ	オフセット	
			LEM	BEM			LEM	BEM
0	**A B C D E F G H**	7	0	0	**H J K L M N O P**	0	7	0
1	**B C D E F G H P**	6	1	0	**G H K L M N O P**	1	6	0
2	**C D E F G H O P**	5	2	0	**F G H L M N O P**	2	5	0
3	**D E F G H N O P**	4	3	0	**E F G H M N O P**	3	4	0
4	**E F G H M N O P**	3	4	0	**D E F G H N O P**	4	3	0
5	**F G H L M N O P**	2	5	0	**C D E F G H O P**	5	2	0
6	**G H K L M N O P**	1	6	0	**B C D E F G H P**	6	1	0
7	**H J K L M N O P**	0	7	0	**A B C D E F G H**	7	0	0

備考　タイプ　　：メモリに出力されるアクセス・タイプ（表 2-3　ロード／ストア命令に関するバイト指定参照）

オフセット：メモリに出力される pAddr2..0

LEM：**リトル・エンディアン・メモリ**（BigEndianMem = 0）

BEM：**ビッグ・エンディアン・メモリ**（BigEndianMem = 1）

例　　外：

TLB**不一致例外**

TLB**無効例外**

TLB**変更例外**

バス・エラー例外

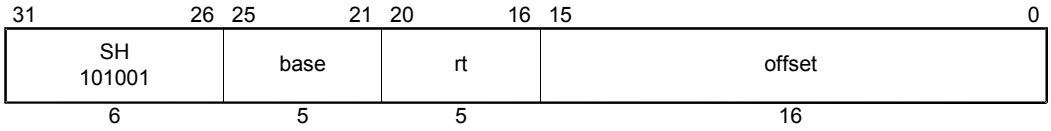
アドレス・エラー例外

予約命令例外（32**ビット・ユーザ／スーパバイザ・モード時）**

保守／廃止

SH

Store Halfword



命　令　形　式：

SH　rt，offset（base）

説　　明：

16ビットのoffsetを符号拡張して汎用レジスタbaseの内容に加算し，仮想アドレスを生成します。レジスタrtの最下位ハーフ・ワードを，アドレス指定したメモリにストアします。

アドレスの最下位ビットが0でない場合，アドレス・エラー例外が発生します。

オペレーション：

```
32　T：　vAddr ← ((offset15)^16 ‖ offset15..0)＋GPR[base]
　　（pAddr，uncached）← AddressTranslation（vAddr，DATA）
　　pAddr ← pAddrPSIZE−1..3 ‖ （pAddr2..0 xor（ReverseEndian^2 ‖ 0））
　　byte ← vAddr2..0 xor（BigEndianCPU^2 ‖ 0）
　　data ← GPR[rt]63−8∗byte..0 ‖ 0^8∗byte
　　StoreMemory（uncached，HALFWORD，data，pAddr，vAddr，DATA）

64　T：　vAddr ← ((offset15)^48 ‖ offset15..0)＋GPR[base]
　　（pAddr，uncached）← AddressTranslation（vAddr，DATA）
　　pAddr ← pAddrPSIZE−1..3 ‖ （pAddr2..0 xor（ReverseEndian^2 ‖ 0））
　　byte ← vAddr2..0 xor（BigEndianCPU^2 ‖ 0）
　　data ← GPR[rt]63−8∗byte..0 ‖ 0^8∗byte
　　StoreMemory（uncached，HALFWORD，data，pAddr，vAddr，DATA）
```

例　　外：

TLB不一致例外

TLB無効例外

TLB変更例外

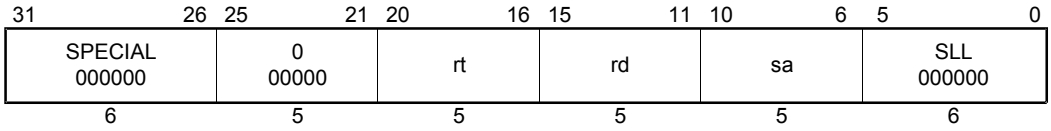
バス・エラー例外

アドレス・エラー例外

保守／廃止

SLL

Shift Left Logical



命　令　形　式：

SLL　rd，rt，sa

説　　明：

汎用レジスタrtの内容をsaで指定するビット数分左にシフトします。下位ビットには0を挿入します。結果は汎用レジスタrdに格納します。64ビット・モード時は，シフトされた32ビット値を符号拡張した値を結果として格納します。シフト値を0にすると，64ビット値の下位32ビットを符号拡張します。この命令で，32ビット値を符号拡張した64ビット値を生成できます。

オペレーション：

```
32  T：   GPR[rd] ← GPR[rt]31−sa..0 ‖ 0^sa

64  T：   s ← 0 ‖ sa
          temp ← GPR[rt]31−s..0 ‖ 0^s
          GPR[rd] ← (temp31)^32 ‖ temp
```

例　　外：

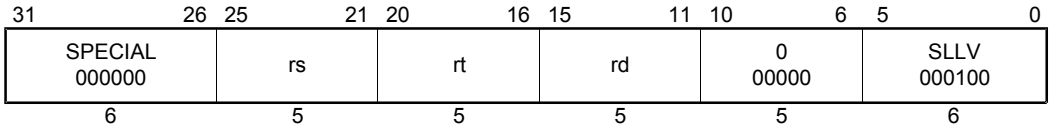
なし

注意　この命令のシフト値が0のとき，アセンブラがNOP命令として扱うことがあります。符号拡張の目的でこの命令を使う場合，アセンブラの仕様をチェックしてください。

保守／廃止

SLLV

Shift Left Logical Variable



命　令　形　式：

SLLV　rd，rt，rs

説　　明：

汎用レジスタrtの内容を，汎用レジスタrsの内容の下位5ビットで指定されるビット数分，左にシフトします。下位ビットには0を挿入します。結果は，汎用レジスタrdに格納します。64ビット・モード時は，シフトされた32ビット値を符号拡張した値を結果として格納します。シフト値を0にすると，64ビット値の下位32ビットを符号拡張します。この命令で，32ビット値を符号拡張した64ビット値を生成できます。

オペレーション：

```
32  T:  s ← GPR[rs]4..0
        GPR[rd] ← GPR[rt] (31− s) ..0 ‖ 0^s

64  T:  s ← 0 ‖ GPR[rs]4..0
        temp ← GPR[rt] (31− s) ..0 ‖ 0^s
        GPR[rd] ← (temp31)^32 ‖ temp
```

例　　外：

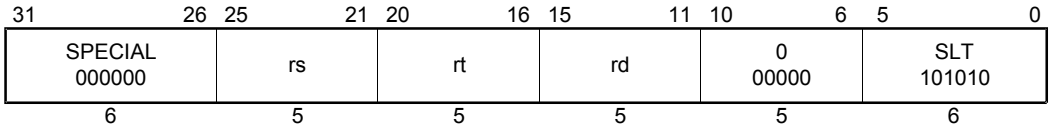
なし

注意　この命令のシフト値が0のとき，アセンブラがNOP命令として扱うことがあります。符号拡張の目的でこの命令を使う場合，アセンブラの仕様をチェックしてください。

保守／廃止

SLT

Set On Less Than



命　令　形　式：

SLT　rd，rs，rt

説　　明：

汎用レジスタrtの内容を，汎用レジスタrsの内容から減算します。これらのレジスタの内容を符号付き整数として扱い，汎用レジスタrsの内容が汎用レジスタrtの内容より小さい場合は1，それ以外の場合は0を汎用レジスタrdに格納します。

整数オーバフロー例外は発生しません。減算がオーバフローしても比較は有効です。

オペレーション：

```
32　T：　if　GPR[rs] ＜ GPR[rt] then
             GPR[rd] ← 0^31 ‖ 1
         else
             GPR[rd] ← 0^32
         endif

64　T：　if　GPR[rs] ＜ GPR[rt] then
             GPR[rd] ← 0^63 ‖ 1
         else
             GPR[rd] ← 0^64
         endif
```

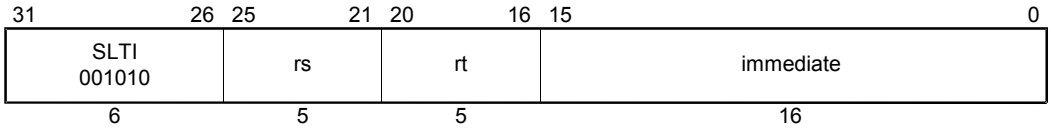
例　　外：

なし

保守／廃止

SLTI

Set On Less Than Immediate



命　令　形　式：

SLTI　rt，rs，immediate

説　　明：

16ビットのimmediateを符号拡張して，汎用レジスタrsの内容から減算します。これらの値を両方とも符号付き整数として扱い，rsの内容が符号拡張したimmediateより小さい場合は1，それ以外の場合は0を汎用レジスタrtに格納します。

整数オーバフロー例外は発生しません。減算がオーバフローしても比較は有効です。

オペレーション：

```
32  T:  if   GPR[rs] < (immediate15)^16 || immediate15..0 then
            GPR[rt] ← 0^31 || 1
        else
            GPR[rt] ← 0^32
        endif

64  T:  if   GPR[rs] <(immediate15)^48 || immediate15..0 then
            GPR[rt] ← 0^63 || 1
        else
            GPR[rt] ← 0^64
        endif
```

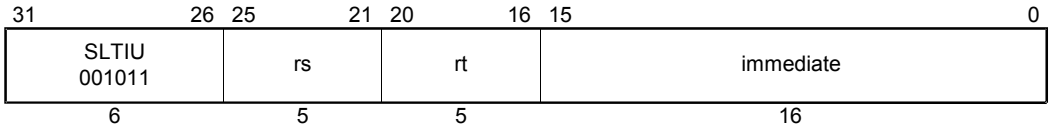
例　　外：

なし

保守／廃止

SLTIU

Set On Less Than Immediate Unsigned



命　令　形　式：

SLTIU　rt，rs，immediate

説　　明：

16ビットのimmediateを符号拡張して，汎用レジスタrsの内容から減算します。これらの値を両方とも符号なし整数として扱い，rsの内容が符号拡張したimmediateより小さい場合は1，それ以外の場合は0を汎用レジスタrtに格納します。

整数オーバフロー例外は発生しません。減算がオーバフローしても比較は有効です。

オペレーション：

```
32  T:   if  (0 || GPR[rs] ) < (0 || (immediate15)^16  || immediate15..0) then
             GPR[rt] ← 0^31 || 1
         else
             GPR[rt] ← 0^32
         endif

64  T:   if  (0 || GPR[rs] ) < (0  || (immediate15)^48 || immediate15..0) then
             GPR[rt] ← 0^63 || 1
         else
             GPR[rt] ← 0^64
         endif
```

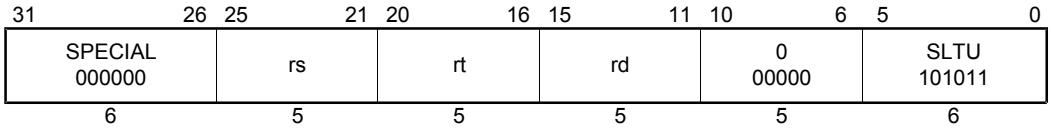
例　　外：

なし

保守／廃止

SLTU

Set On Less Than Unsigned



命　令　形　式：

SLTU　rd，rs，rt

説　　明：

汎用レジスタrtの内容を，汎用レジスタrsの内容から減算します。これらの値を両方とも符号なし整数として扱い，汎用レジスタrsの内容が汎用レジスタrtの内容より小さい場合は1，それ以外の場合は0を汎用レジスタrdに格納します。

整数オーバフロー例外は発生しません。減算がオーバフローしても比較は有効です。

オペレーション：

```
32  T:  if  (0 || GPR[rs] ) < 0 || GPR[rt] then
            GPR[rd] ← 0^31 || 1
        else
            GPR[rd] ← 0^32
        endif

64  T:  if  (0 || GPR[rs] ) < 0 || GPR[rt] then
            GPR[rd] ← 0^63 || 1
        else
            GPR[rd] ← 0^64
        endif
```

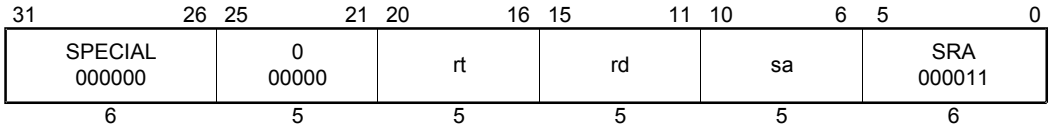
例　　外：

なし

保守／廃止

SRA

Shift Right Arithmetic



命　令　形　式：

SRA　rd，rt，sa

説　　明：

汎用レジスタrt**の内容を，**sa**で指定されるビット数分右にシフトします。上位ビットには符号ビットを挿入します。結果は汎用レジスタ**rd**に格納します。64ビット・モード時は，32ビット値を符号拡張した値を結果として格納します。**

オペレーション：

```
32　T：　GPR[rd] ← (GPR[rt]31)^sa ‖ GPR[rt]31..sa

64　T：　s ← 0 ‖ sa
         temp ← (GPR[rt]31)^s ‖ GPR[rt]31..s
         GPR[rd] ← (temp31)^32 ‖ temp
```

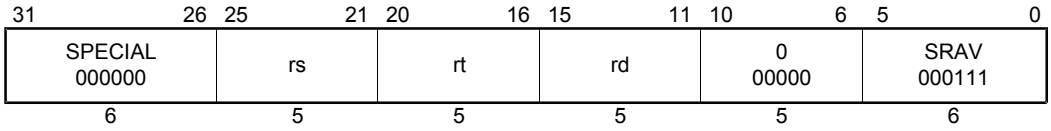
例　　外：

なし

保守／廃止

SRAV

Shift Right Arithmetic Variable



命　令　形　式：

SRAV　rd，rt，rs

説　　明：

汎用レジスタrt**の内容を，汎用レジスタ**rs**の内容の下位5ビットで指定されるビット数分，右にシフトします。上位ビットは符号拡張します。結果は汎用レジスタ**rd**に格納します。64ビット・モード時は，32ビット値を符号拡張した値を結果として格納します。**

オペレーション：

```
32　T：　s ← GPR[rs]4..0
         GPR[rd] ← (GPR[rt]31)^s ‖ GPR[rt]31..s

64　T：　s ← GPR[rs]4..0
         temp ← (GPR[rt]31)^s ‖ GPR[rt]31..s
         GPR[rd] ← (temp31)^32 ‖ temp
```

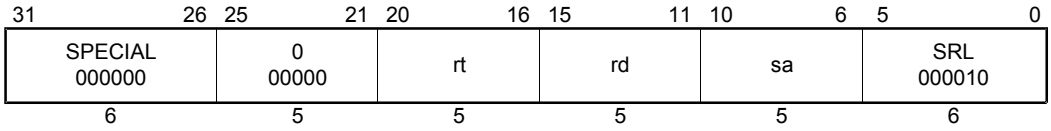
例　　外：

なし

保守／廃止

SRL

Shift Right Logical



命　令　形　式：

SRL　rd，rt，sa

説　　明：

汎用レジスタrt**の内容を，**sa**で指定されるビット数分右にシフトします。上位ビットには0を挿入します。結果は汎用レジスタ**rd**に格納します。64ビット・モード時は，32ビット値を符号拡張した値を結果として格納します。**

オペレーション：

```
32　T：　GPR[rd] ← 0^sa || GPR[rt]31..sa

64　T：　s ← 0 || sa
         temp ← 0^s || GPR[rt]31..s
         GPR[rd] ← (temp31)^32 || temp
```

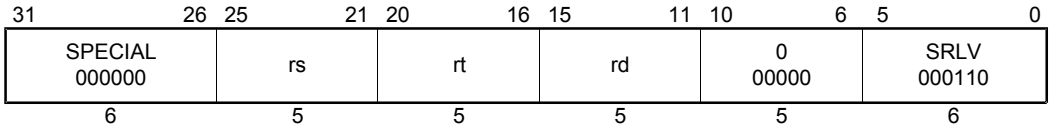
例　　外：

なし

保守／廃止

SRLV

Shift Right Logical Variable



命　令　形　式：

SRLV　rd，rt，rs

説　　明：

汎用レジスタrt**の内容を，汎用レジスタ**rs**の内容の下位5ビットで指定されるビット数分，右にシフトします。上位ビットには0を挿入します。結果は汎用レジスタ**rd**に格納します。64ビット・モード時は，32ビット値を符号拡張した値を結果として格納します。**

オペレーション：

```
32　T：　s ← GPR[rs]4..0
         GPR[rd] ← 0^s ‖ GPR[rt]31..s

64　T：　s ← GPR[rs]4..0
         temp ← 0^s ‖ GPR[rt]31..s
         GPR[rd] ← (temp31)^32 ‖ temp
```

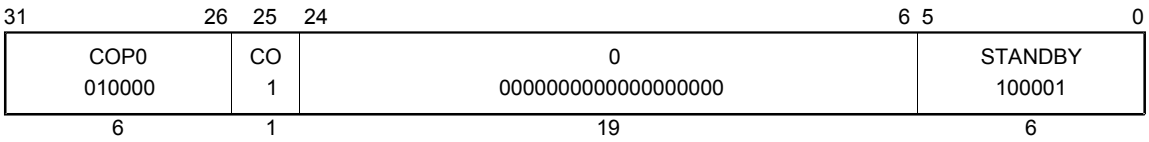
例　　外：

なし

保守／廃止

STANDBY

Standby



命　令　形　式：

STANDBY

説　　明：

VR4120Aコアを Fullspeed モードから Standby モードに移行します。

命令の実行が WB ステージまで進むと，SysAD バスがアイドル状態になるのを待って内部クロックをハイ・レベルに固定し，パイプラインの動作を中断します。

Standby モードでは，PLL，タイマ／割り込み関連のクロック，内部バス・クロック（TClock, MasterOut）は通常どおり動作します。

Standby モードから Fullspeed モードに移行するには，外部割り込み，内部タイマ割り込み，NMI を発生させるか，ソフト・リセット，コールド・リセットを実行します。

オペレーション：

```
32, 64    T：
          T＋1：  Standby Operation（）
```

例　　外：

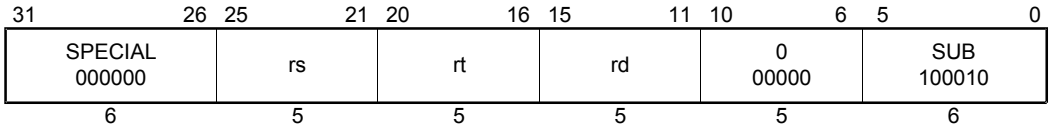
コプロセッサ使用不可例外

備考　モード遷移時の周辺ユニット動作の詳細については，2.6.3.1 **パワー・モード**を参照してください。

保守／廃止

SUB

Subtract



命　令　形　式：

SUB　rd，rs，rt

説　　明：

汎用レジスタrs**の内容から汎用レジスタ**rt**の内容を減算し，結果を汎用レジスタ**rd**に格納します。64ビット・モードでは，32ビットの値を符号拡張したものを結果として格納します。**

ビット30**からの桁上がりとビット**31**からの桁上がりが異なる（2の補数オーバフロー）場合，整数オーバフロー例外が発生します。整数オーバフロー例外が発生すると，デスティネーション・レジスタ**rd**は変更しません。**

オペレーション：

```
32　T：　GPR[rd] ← GPR[rs]－GPR[rt]

64　T：　temp ← GPR[rs]－GPR[rt]
         GPR[rd] ← (temp31)^32 ‖ temp31..0
```

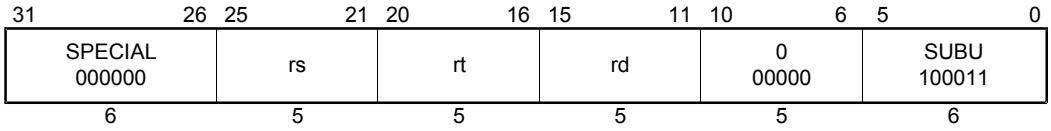
例　　外：

整数オーバフロー例外

保守／廃止

SUBU

Subtract Unsigned



命　令　形　式：

SUBU　rd，rs，rt

説　　明：

汎用レジスタrs**の内容から汎用レジスタ**rt**の内容を減算し，結果を汎用レジスタ**rd**に格納します。64ビット・モードでは，32ビット値を符号拡張したものを結果として格納します。**

SUB**命令との違いは，**SUBU**命令は整数オーバフロー例外を発生しないという点だけです。**

オペレーション：

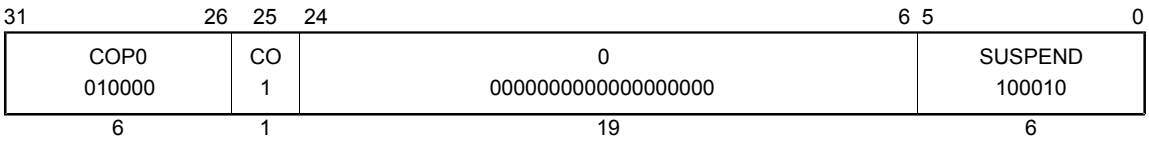
```
32　T：　GPR[rd] ← GPR[rs]－GPR[rt]

64　T：　temp ← GPR[rs]－GPR[rt]
         GPR[rd] ← (temp31)^32 ‖ temp31..0
```

例　　外：

なし

SUSPEND

Suspend



命　令　形　式：

SUSPEND

説　　明：

VR4120Aコアを Fullspeed モードから Suspend モードに移行します。

命令の実行が WB ステージまで進むと，SysAD バスがアイドル状態になるのを待って内部クロック（TClock を含む）をハイ・レベルに固定し，パイプラインと外部バス・インタフェースの動作を中断します。

Suspend モードでは，PLL，タイマ／割り込み関連のクロック，MasterOut（内部）は通常どおり動作します。

Suspend モードから Fullspeed モードに移行するには，外部割り込み，内部タイマ割り込み，NMI を発生させるか，ソフト・リセット，コールド・リセットを実行します。

オペレーション：

```
32, 64    T：
          T＋1：  Suspend Operation（）
```

例　　外：

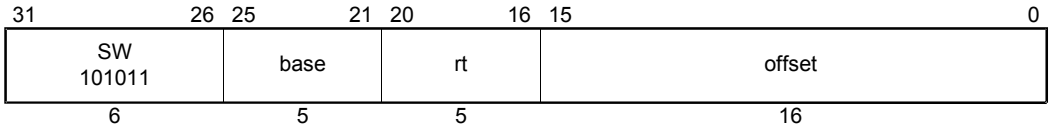
コプロセッサ使用不可例外

備考　モード遷移時の周辺ユニット動作の詳細については，**2.6.3.1　パワー・モード**を参照してください。

保守／廃止

SW

Store Word



命　令　形　式：

SW　rt，offset（base）

説　　明：

16ビットのoffsetを符号拡張して汎用レジスタbaseの内容に加算し，仮想アドレスを生成します。汎用レジスタrtの内容を，アドレス指定したメモリにストアします。アドレスの下位2ビットのいずれかが0でない場合，アドレス・エラー例外が発生します。

オペレーション：

```
32　T：　vAddr ← ((offset15)^16 ‖ offset15..0)＋GPR[base]
　　（pAddr，uncached）← AddressTranslation（vAddr，DATA）
　　pAddr ← pAddrPSIZE−1..3 ‖ (pAddr2..0 xor (ReverseEndian ‖ 0^2))
　　byte ← vAddr2..0 xor（BigEndianCPU ‖ 0^2）
　　data ← GPR[rt]63−8∗byte..0 ‖ 0^8∗byte
　　StoreMemory（uncached，WORD，data，pAddr，vAddr，DATA）

64　T：　vAddr ← ((offset15)^48 ‖ offset15..0)＋GPR[base]
　　（pAddr，uncached）← AddressTranslation（vAddr，DATA）
　　pAddr ← pAddrPSIZE−1..3 ‖ (pAddr2..0 xor (ReverseEndian ‖ 0^2))
　　byte ← vAddr2..0 xor（BigEndianCPU ‖ 0^2）
　　data ← GPR[rt]63−8∗byte..0 ‖ 0^8∗byte
　　StoreMemory（uncached，WORD，data，pAddr，vAddr，DATA）
```

例　　外：

TLB不一致例外

TLB無効例外

TLB変更例外

バス・エラー例外

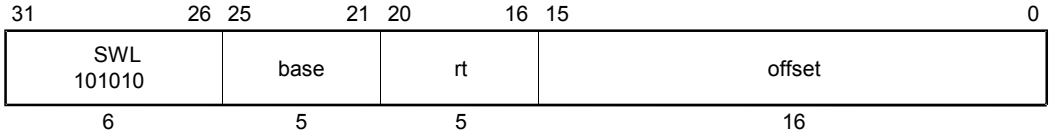
アドレス・エラー例外

保守／廃止

SWL

Store Word Left

（1/3）



命　令　形　式：

SWL　rt，offset（base）

説　　明：

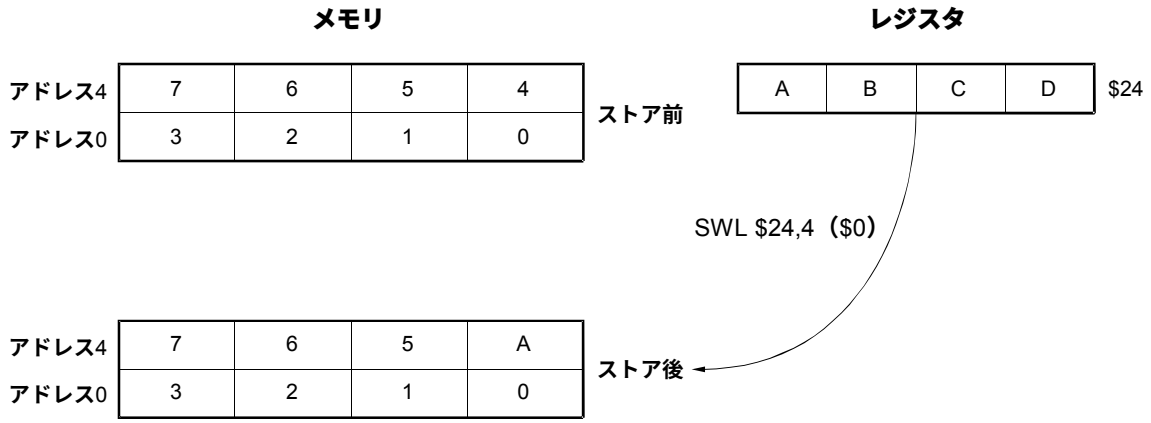
この命令は，レジスタ中のワード・データを，ワード境界にないメモリ中のワードにストアするときに，SWR**命令と組み合わせて使用します。**SWL**命令がデータの上位部分を，**SWR**命令がデータの下位部分をメモリにストアします。**

16ビットのoffset**を符号拡張して汎用レジスタ**base**の内容と加算し，仮想アドレスを生成します。アドレス指定されたバイトを最上位バイトとするメモリ中のワード・データについて，ターゲット・アドレスと同じワード境界にあるメモリに，汎用レジスタ**rt**の上位部分をストアします。**

指定したアドレスによっては，ストアされるバイト数は1-4**バイトの範囲で変わります。**

つまり，まず汎用レジスタrt**の最上位をアドレス指定したメモリ内バイトにストアし，同じワード境界に引き続く下位のバイトがあれば，これをメモリの次のバイトにストアする動作を繰り返します。**

指定アドレスがワード境界に位置していないことによるアドレス例外は発生しません。



保守／廃止

SWL

Store Word Left

（2/3）

オペレーション：

```
32　T：　vAddr ←((offset15)^16 ‖ offset15..0)＋GPR[base]
         （pAddr，uncached）← AddressTranslation（vAddr，DATA）
         pAddr ← pAddrPSIZE−1..3 ‖ （pAddr2..0 xor ReverseEndian^3）
         if BigEndianMem = 0 then
            pAddr ← pAddrPSIZE−1..2 ‖ 0^2
         endif
         byte ← vAddr1..0 xor BigEndianCPU^2
         if（vAddr2 xor BigEndianCPU）= 0 then
            data ← 0^32 ‖ 0^(24−8∗byte) ‖ GPR[rt]31..24−8∗byte
         else
            data ← 0^(24−8∗byte) ‖ GPR[rt]31..24−8∗byte ‖ 0^32
         endif
         StoreMemory（uncached，byte，data，pAddr，vAddr，DATA）

64　T：　vAddr ← ((offset15)^48 ‖ offset15..0)＋GPR[base]
         （pAddr，uncached）← AddressTranslation（vAddr，DATA）
         pAddr ← pAddrPSIZE−1..3 ‖ （pAddr2..0 xor ReverseEndian^3）
         if BigEndianMem = 0 then
            pAddr ← pAddrPSIZE−1..2 ‖ 0^2
         endif
         byte ← vAddr1..0 xor BigEndianCPU^2
         if（vAddr2 xor BigEndianCPU）= 0 then
            data ← 0^32 ‖ 0^(24−8∗byte) ‖ GPR[rt]31..24−8∗byte
         else
            data ← 0^(24−8∗byte) ‖ GPR[rt]31..24−8∗byte ‖ 0^32
         endif
         StoreMemory（uncached，byte，data，pAddr，vAddr，DATA）
```

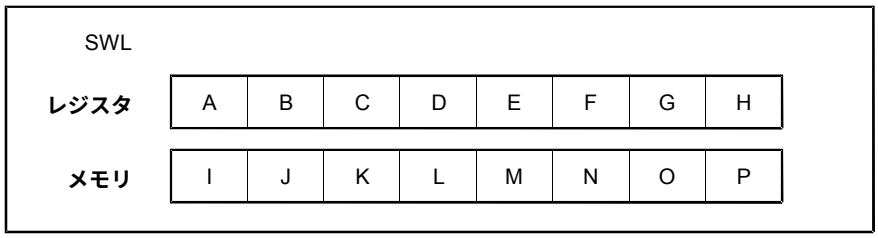
保守／廃止

SWL

Store Word Left

（3/3）

SWL命令に与えるレジスタの内容とその結果（メモリのワードの各バイト）の関係を次に示します。



vAddr2..0	ビッグ・エンディアンCPU＝0				ビッグ・エンディアンCPU＝1			
	デスティネーション	タイプ	オフセット		デスティネーション	タイプ	オフセット	
			LEM	BEM			LEM	BEM
0	I J K L M N O E	0	0	7	E F G H M N O P	3	4	0
1	I J K L M N E F	1	0	6	I E F G M N O P	2	4	1
2	I J K L M E F G	2	0	5	I J E F M N O P	1	4	2
3	I J K L E F G H	3	0	4	I J K E M N O P	0	4	3
4	I J K E M N O P	0	4	3	I J K L E F G H	3	0	4
5	I J E F M N O P	1	4	2	I J K L M E F G	2	0	5
6	I E F G M N O P	2	4	1	I J K L M N E F	1	0	6
7	E F G H M N O P	3	4	0	I J K L M N O E	0	0	7

備考　タイプ　　：メモリに出力されるアクセス・タイプ（表 2-3　ロード／ストア命令に関するバイト指定参照）

オフセット：メモリに出力される pAddr2..0

LEM：**リトル・エンディアン・メモリ**（BigEndianMem = 0）

BEM：**ビッグ・エンディアン・メモリ**（BigEndianMem = 1）

例　　外：

TLB**不一致例外**

TLB**無効例外**

TLB**変更例外**

バス・エラー例外

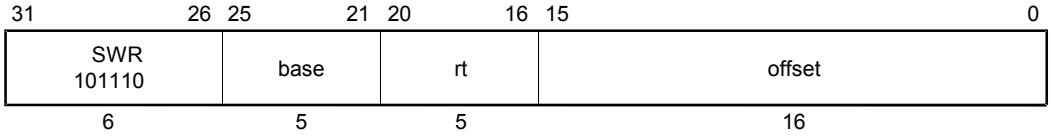
アドレス・エラー例外

保守／廃止

SWR

Store Word Right

（1/3）



命　令　形　式：

SWR　rt，offset（base）

説　　明：

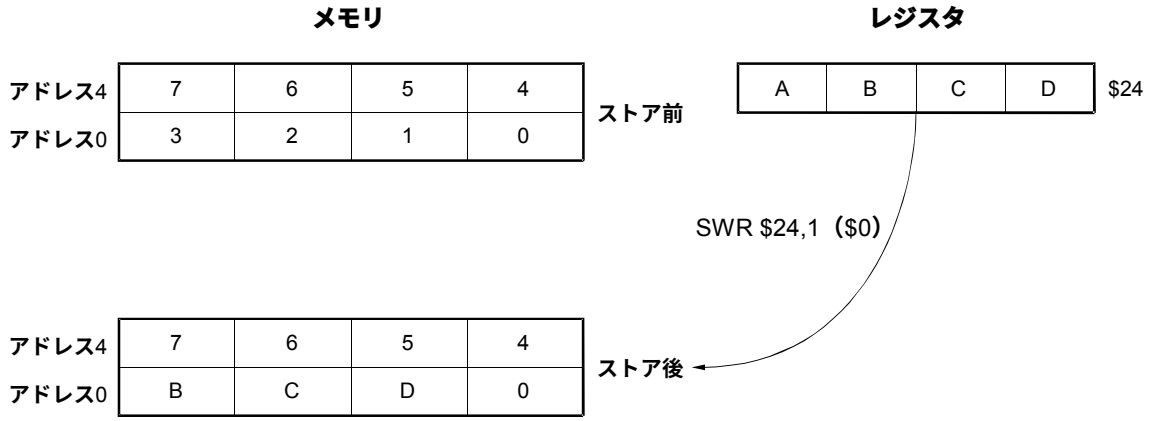
この命令は，レジスタ中のワード・データを，ワード境界にないメモリ中のワードにストアするときに，SWL**命令と組み合わせて使用します。**SWL**命令がデータの上位部分を，**SWR**命令がデータの下位部分をメモリにストアします。**

16ビットのoffset**を符号拡張して汎用レジスタ**base**の内容と加算し，仮想アドレスを生成します。アドレス指定されたバイトを最下位バイトとするメモリ中のワード・データについて，ターゲット・アドレスと同じワード境界にあるメモリに，汎用レジスタ**rt**の下位部分をストアします。**

指定したアドレスによっては，ストアされるバイト数は1-4**バイトの範囲で変わります。**

つまり，まず汎用レジスタrt**の最下位バイトをアドレス指定したメモリ内バイトにストアし，同じワード境界に引き続く上位のバイトがあれば，これをメモリの次のバイトにストアする動作を繰り返します。**

指定アドレスがワード境界に位置していないことによるアドレス例外は発生しません。



保守／廃止

SWR

Store Word Right

（2/3）

オペレーション：

```
32　T：　vAddr ←((offset15))^16 ‖ offset15..0)＋GPR[base]
         （pAddr，uncached）← AddressTranslation（vAddr，DATA）
         pAddr ← pAddrPSIZE−1..3 ‖ （pAddr2..0 xor ReverseEndian^3）
         if BigEndianMem = 1 then
             pAddr ← pAddrPSIZE−1..2 ‖ 0^2
         endif
         byte ← vAddr1..0 xor BigEndianCPU^2
         if（vAddr2 xor BigEndianCPU）= 0 then
             data ← 0^32 ‖ GPR[rt]31−8∗byte..0 ‖ 0^8∗byte
         else
             data ← GPR[rt]31−8∗byte ‖ 0^8∗byte ‖ 0^32
         endif
         StoreMemory（uncached，WORD - byte，data，pAddr，vAddr，DATA）

64　T：　vAddr ← ((offset15))^48 ‖ offset15..0)＋GPR[base]
         （pAddr，uncached）← AddressTranslation（vAddr，DATA）
         pAddr ← pAddrPSIZE−1..3 ‖ （pAddr2..0 xor ReverseEndian^3）
         if BigEndianMem = 1 then
             pAddr ← pAddrPSIZE−1..2 ‖ 0^2
         endif
         byte ← vAddr1..0 xor BigEndianCPU^2
         if（vAddr2 xor BigEndianCPU）= 0 then
             data ← 0^32 ‖ GPR[rt]31−8∗byte..0 ‖ 0^8∗byte
         else
             data ← GPR[rt]31−8∗byte ‖ 0^8∗byte ‖ 0^32
         endif
         StoreMemory（uncached，WORD - byte，data，pAddr，vAddr，DATA）
```

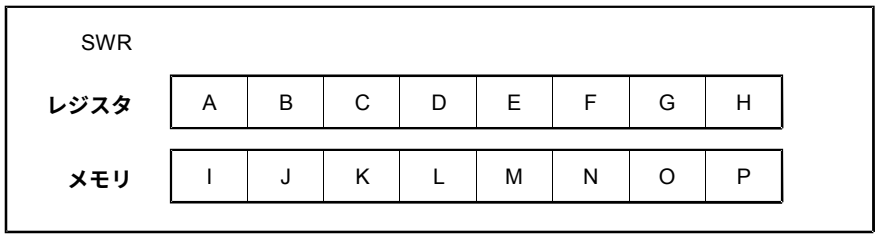
保守／廃止

SWR

Store Word Right

（3/3）

SWR命令に与えるレジスタの内容とその結果（メモリのワードの各バイト）の関係を次に示します。



vAddr2..0	デスティネーション	タイプ	オフセット（LEM）
0	**I J K L E F G H**	3	0
1	**I J K L F G H P**	2	1
2	**I J K L G H O P**	1	2
3	**I J K L H N O P**	0	3
4	**E F G H M N O P**	3	4
5	**F G H L M N O P**	2	5
6	**G H K L M N O P**	1	6
7	**H J K L M N O P**	0	7

備考　LEM　　：リトル・エンディアン・メモリ（BigEndianMem = 0）

タイプ　　：メモリに出力されるアクセス・タイプ（**表 2-3　ロード／ストア命令に関するバイト指定**参照）

オフセット：メモリに出力される pAddr$_{2..0}$

例　　外：

TLB**不一致例外**

TLB**無効例外**

TLB**変更例外**

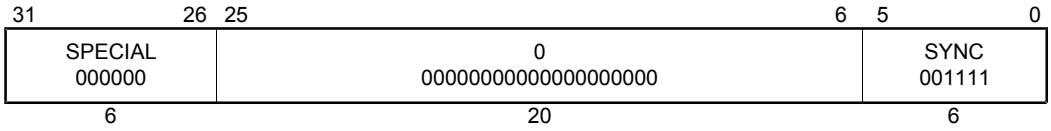
バス・エラー例外

アドレス・エラー例外

保守／廃止

SYNC

Synchronize



命　令　形　式：

SYNC

説　　明：

SYNC命令は，VR4120AコアのNOP命令として実行されます。オペレーションはVR4000™に準拠したコードと互換性があります。

なお，この命令は主にVR4000, VR4400™とのソフトウエア互換性を保つために定義しています。

オペレーション：

```
32，64　T：　SyncOperation（）
```

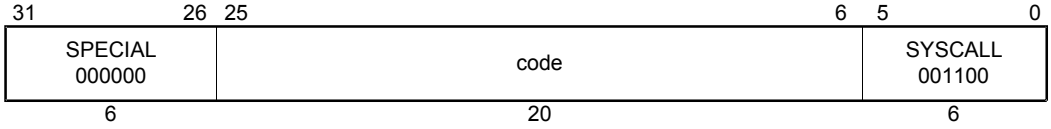
例　　外：

なし

保守／廃止

SYSCALL

System Call



命　令　形　式：

SYSCALL

説　　明：

この命令を実行するとシステム・コール例外が発生し，無条件に制御を例外ハンドラに渡します。

code**領域を使用することによって例外ハンドラにパラメータを送ることができます。例外ハンドラがこのパラメータを使用する場合には，命令を含むメモリ・ワードの内容をデータとしてロードする必要があります。**

オペレーション：

```
32，64　T：　SystemCallException
```

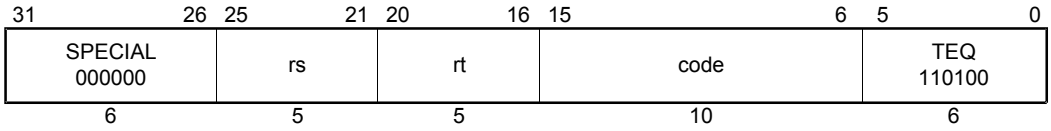
例　　外：

システム・コール例外

保守／廃止

TEQ

Trap If Equal



命　令　形　式：

TEQ　rs，rt

説　　明：

汎用レジスタrtの内容を，汎用レジスタrsの内容と比較します。汎用レジスタrsの内容が汎用レジスタrtの内容と等しい場合，トラップ例外が発生します。

code領域を使用することによって，例外ハンドラにパラメータを送ることができます。例外ハンドラがこのパラメータを使用する場合には，命令を含むメモリ・ワードの内容をデータとしてロードする必要があります。

オペレーション：

```
32，64　T：　if GPR[rs] = GPR[rt] then
               TrapException
             endif
```

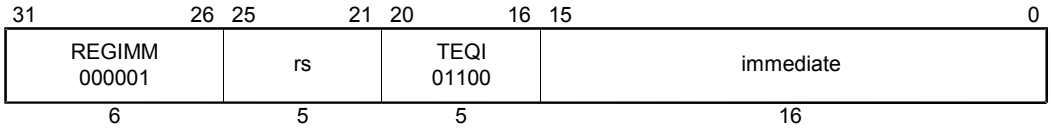
例　　外：

トラップ例外

保守／廃止

TEQI

Trap If Equal Immediate



命　令　形　式：

TEQI　rs，immediate

説　　明：

16ビットのimmediateを符号拡張して，汎用レジスタrsの内容と比較します。汎用レジスタrsの内容が符号拡張したimmediateと等しい場合，トラップ例外が発生します。

オペレーション：

```
32　T：　if GPR[rs] = (immediate15)^16 || immediate15..0 then
            TrapException
         endif

64　T：　if GPR[rs] = (immediate15)^48 || immediate15..0 then
            TrapException
         endif
```

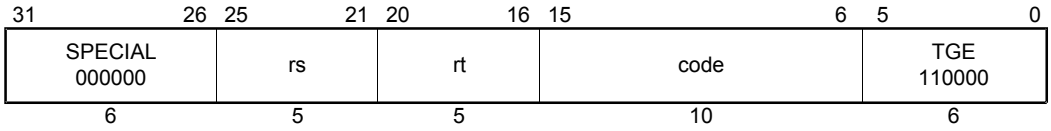
例　　外：

トラップ例外

保守／廃止

TGE

Trap If Greater Than Or Equal



命　令　形　式：

TGE　rs，rt

説　　明：

汎用レジスタrtの内容を，汎用レジスタrsの内容と比較します。両方のレジスタの内容を符号付き整数として扱い，レジスタrsの内容が汎用レジスタrtの内容以上の場合，トラップ例外が発生します。

code領域を使用することによって，例外ハンドラにパラメータを送ることができます。例外ハンドラがこのパラメータを使用する場合には，命令を含むメモリ・ワードの内容をデータとしてロードする必要があります。

オペレーション：

```
32，64　T：　if GPR[rs] ≧ GPR[rt] then
               TrapException
             endif
```

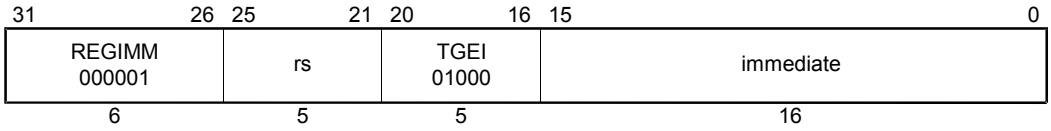
例　　外：

トラップ例外

保守／廃止

TGEI

Trap If Greater Than Or Equal Immediate



命　令　形　式：

TGEI　rs，immediate

説　　明：

16ビットのimmediateを符号拡張し，汎用レジスタrsの内容と比較します。両方の値を符号付き整数として扱い，汎用レジスタrsの内容が符号拡張したimmediate以上の場合，トラップ例外が発生します。

オペレーション：

```
32　T：　if GPR[rs] ≧ (immediate15)^16 || immediate15..0 then
            TrapException
         endif

64　T：　if GPR[rs] ≧ (immediate15)^48 || immediate15..0 then
            TrapException
         endif
```

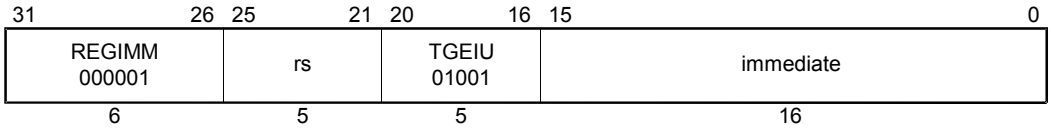
例　　外：

トラップ例外

保守／廃止

TGEIU

Trap If Greater Than Or Equal Immediate Unsigned



命　令　形　式：

TGEIU　rs，immediate

説　　明：

16ビットのimmediateを符号拡張し，汎用レジスタrsの内容と比較します。両方の値を符号なし整数として扱い，汎用レジスタrsの内容が符号拡張したimmediate以上の場合，トラップ例外が発生します。

オペレーション：

```
32　T：　if (0 || GPR[rs] ) ≧ (0 || (immediate15)^16 || immediate15..0) then
            TrapException
         endif

64　T：　if (0 || GPR[rs] ) ≧ (0 || (immediate15)^48 || immediate15..0) then
            TrapException
         endif
```

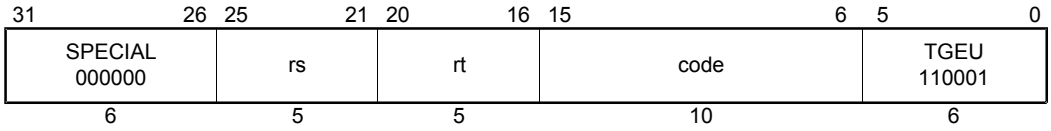
例　　外：

トラップ例外

保守／廃止

TGEU

Trap If Greater Than Or Equal Unsigned



命　令　形　式：

TGEU　rs，rt

説　　明：

汎用レジスタrtの内容を，汎用レジスタrsの内容と比較します。両方の値を符号なし整数として扱い，汎用レジスタrsの内容が汎用レジスタrtの内容以上の場合，トラップ例外が発生します。

code領域を使用することによって，例外ハンドラにパラメータを送ることができます。例外ハンドラがこのパラメータを使用する場合には，命令を含むメモリ・ワードの内容をデータとしてロードする必要があります。

オペレーション：

```
32，64　T：　if（0 ‖ GPR[rs] ）≧（0 ‖ GPR[rt] ）then
                TrapException
             endif
```

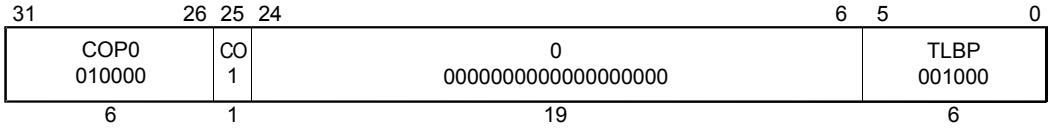
例　　外：

トラップ例外

保守／廃止

TLBP

Probe TLB For Matching Entry



命　令　形　式：

TLBP

説　　明：

エントリHi**レジスタの内容と一致する**TLB**エントリを検索し，一致した**TLB**エントリの番号をインデクス・レジスタにセットします。一致する**TLB**エントリがなかった場合，インデクス・レジスタの最上位ビットをセットします。**

TLBP命令の直後の命令と関連するメモリ参照のオペレーションや，複数のTLB**エントリが一致した場合の動作は不定です。**

オペレーション：

```
32  T:  Index ← 1 || 0^25 || Undefined^6
        for i in 0..TLBEntries−1
            if (TLB[i]95..77 = EntryHi31..13) and (TLB[i]76 or
             (TLB[i]71..64 = EntryHi7..0)) then
               Index ← 0^26 || i5..0
            endif
        endfor

64  T:  Index ← 1 || 0^25 || Undefined^6
        for i in 0..TLBEntries−1
            if (TLB[i]167..141 and not (0^15 || TLB[i]216..205))
            = (EntryHi39..13 and not (0^15 || TLB[i]216..205))and
             (TLB[i]140 or (TLB[i]135..128 = EntryHi7..0)) then
               Index ← 0^26 || i5..0
            endif
        endfor
```

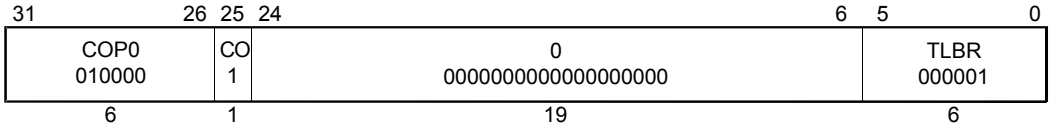
例　　外：

コプロセッサ使用不可例外

保守／廃止

TLBR

Read Indexed TLB Entry



命　令　形　式：

TLBR

説　　明：

エントリHi**とエントリ**Lo**レジスタに，インデクス・レジスタによって指定された**TLB**エントリの内容をロードします。**Gビット（ASID**一致の制御）を**TLB**から読み出し，エントリ**Lo0**とエントリ**Lo1**レジスタに書き込みます。**

インデクス・レジスタの内容がVR4120A**コアの**TLB**エントリの数より大きい場合，オペレーションは無効です。**

オペレーション：

```
32  T:  PageMask ← TLB[Index5..0]127..96
        EntryHi ← TLB[Index5..0]95..64 and not TLB[Index5..0]127..96
        EntryLo1 ← TLB[Index5..0]63..33 || TLB[Index5..0]76
        EntryLo0 ← TLB[Index5..0]31..1 || TLB[Index5..0]76

64  T:  PageMask ← TLB[Index5..0]255..192
        EntryHi ← TLB[Index5..0]191..128 and not TLB[Index5..0]255..192
        EntryLo1 ← TLB[Index5..0]127..65 || TLB[Index5..0]140
        EntryLo0 ← TLB[Index5..0]63..1 || TLB[Index5..0]140
```

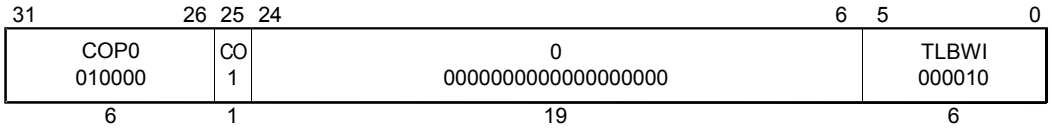
例　　外：

コプロセッサ使用不可例外

保守／廃止

TLBWI

Write Indexed TLB Entry



命　令　形　式：

TLBWI

説　　明：

エントリHi**とエントリ**Lo**レジスタの内容を，インデクス・レジスタによって指定された**TLB**エントリにロードします。このとき，**TLB**の**Gビットには，エントリ**Lo0**とエントリ**Lo1**レジスタ内の**Gビットの論理積をとった結果を書き込みます。

インデクス・レジスタの内容がVR4120A**コアの**TLB**エントリの数より大きい場合，オペレーションは無効です。**

オペレーション：

```
32, 64　T：　TLB[Index5..0] ←
                PageMask ‖　（EntryHi and not PageMask）　‖ EntryLo1 ‖ EntryLo0
```

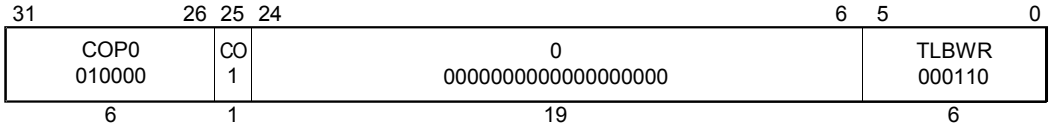
例　　外：

コプロセッサ使用不可例外

保守／廃止

TLBWR

Write Random TLB Entry



命　令　形　式：

TLBWR

説　　明：

エントリHi**とエントリ**Lo**レジスタの内容を，ランダム・レジスタによって指定された**TLB**エントリにロードします。このとき，**TLB**のG**ビットには，エントリ**Lo0**とエントリ**Lo1**レジスタ内のG**ビットの論理積をとった結果を書き込みます。

オペレーション：

```
32, 64　T：　TLB[Random5..0] ←
                PageMask ‖ （EntryHi and not PageMask） ‖ EntryLo1 ‖ EntryLo0
```

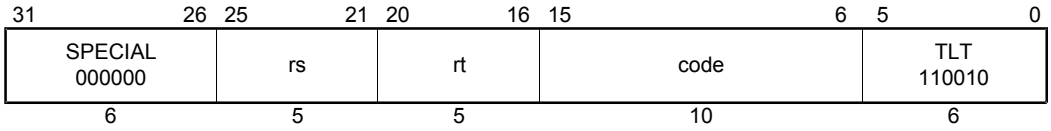
例　　外：

コプロセッサ使用不可例外

保守／廃止

TLT

Trap If Less Than



命　令　形　式：

TLT　rs，rt

説　　明：

汎用レジスタrtの内容を，汎用レジスタrsの内容と比較します。両方の値を符号付き整数として扱い，汎用レジスタrsの内容が汎用レジスタrtの内容より小さい場合，トラップ例外が発生します。

code領域を使用することによって，例外ハンドラにパラメータを送ることができます。例外ハンドラがこのパラメータを使用する場合には，命令を含むメモリ・ワードの内容をデータとしてロードする必要があります。

オペレーション：

```
32，64　T：　if GPR[rs] ＜ GPR[rt] then
               TrapException
             endif
```

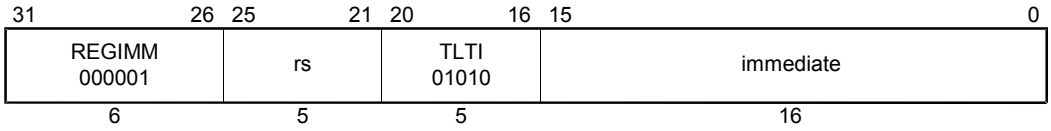
例　　外：

トラップ例外

保守／廃止

TLTI

Trap If Less Than Immediate



命　令　形　式：

TLTI　rs，immediate

説　　明：

16ビットのimmediateを符号拡張し，汎用レジスタrsの内容と比較します。両方の値を符号付き整数として扱い，汎用レジスタrsの内容が符号拡張したimmediateより小さい場合，トラップ例外が発生します。

オペレーション：

```
32　T：　if　GPR[rs] ＜ (immediate15)^16 ‖ immediate15..0 then
           TrapException
         endif

64　T：　if　GPR[rs] ＜ (immediate15)^48 ‖ immediate15..0 then
           TrapException
         endif
```

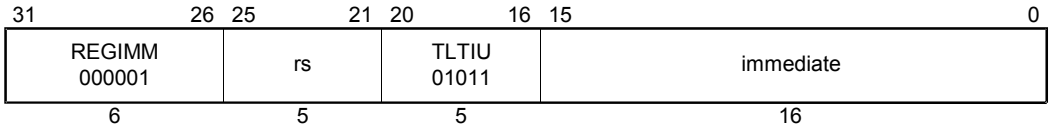
例　　外：

トラップ例外

保守／廃止

TLTIU

Trap If Less Than Immediate Unsigned



命　令　形　式：

TLTIU　rs，immediate

説　　明：

16ビットのimmediateを符号拡張し，汎用レジスタrsの内容と比較します。両方の値を符号なし整数として扱い，汎用レジスタrsの内容が符号拡張したimmediateより小さい場合，トラップ例外が発生します。

オペレーション：

```
32　T：　if　(0 || GPR[rs] ) < (0 || (immediate15)^16 || immediate15..0) then
            TrapException
         endif

64　T：　if　(0 || GPR[rs] ) < (0 || (immediate15)^48 || immediate15..0) then
            TrapException
         endif
```

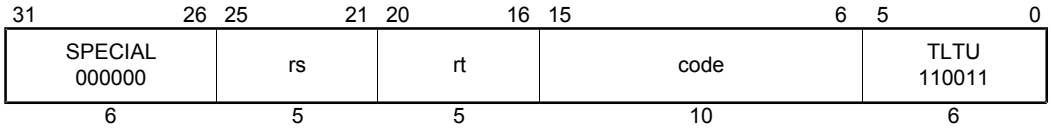
例　　外：

トラップ例外

保守／廃止

TLTU

Trap If Less Than Unsigned



命　令　形　式：

TLTU　rs，rt

説　　明：

汎用レジスタrtの内容を，汎用レジスタrsの内容と比較します。両方の値を符号なし整数として扱い，汎用レジスタrsの内容が汎用レジスタrtの内容より小さい場合，トラップ例外が発生します。

code領域を使用することによって，例外ハンドラにパラメータを送ることができます。例外ハンドラがこのパラメータを使用する場合には，命令を含むメモリ・ワードの内容をデータとしてロードする必要があります。

オペレーション：

```
32，64　T：　if（0 ‖ GPR[rs] ）＜（0 ‖ GPR[rt] ）then
               TrapException
             endif
```

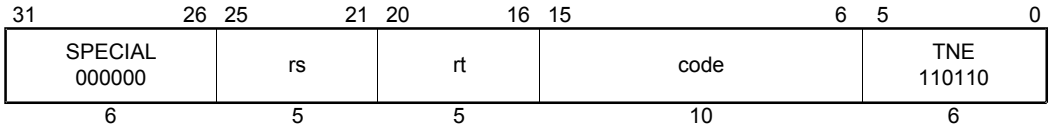
例　　外：

トラップ例外

保守／廃止

TNE

Trap If Not Equal



命　令　形　式：

TNE　rs，rt

説　　明：

汎用レジスタrtの内容を，汎用レジスタrsの内容と比較します。汎用レジスタrsの内容が汎用レジスタrtの内容と等しくない場合，トラップ例外が発生します。

code領域を使用することによって，例外ハンドラにパラメータを送ることができます。例外ハンドラがこのパラメータを使用する場合には，命令を含むメモリ・ワードの内容をデータとしてロードする必要があります。

オペレーション：

```
32，64　T：　if GPR[rs] ≠ GPR[rt] then
               TrapException
             endif
```

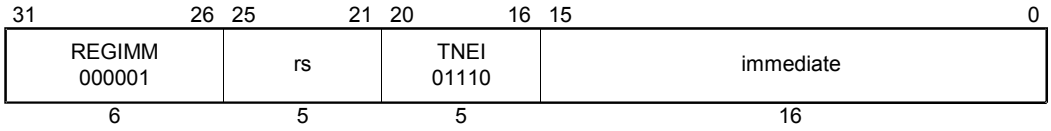
例　　外：

トラップ例外

保守／廃止

TNEI

Trap If Not Equal Immediate



命　令　形　式：

TNEI　rs，immediate

説　　明：

16ビットのimmediateを符号拡張し，汎用レジスタrsの内容と比較します。汎用レジスタrsの内容が符号拡張されたimmediateと等しくない場合，トラップ例外が発生します。

オペレーション：

```
32　T：　if　GPR[rs] ≠ (immediate15)16 ‖ immediate15..0 then
           TrapException
         endif

64　T：　if　GPR[rs] ≠ (immediate15)48 ‖ immediate15..0 then
           TrapException
         endif
```

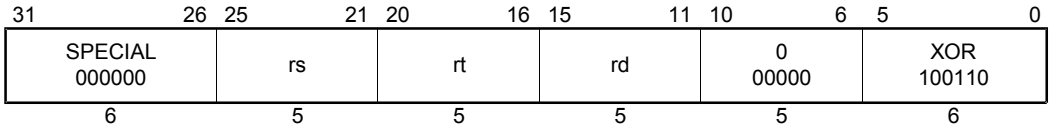
例　　外：

トラップ例外

保守／廃止

XOR

Exclusive Or



命　令　形　式：

XOR　rd，rs，rt

説　　明：

汎用レジスタrs**の内容と汎用レジスタ**rt**の内容の，ビットごとの排他的論理和演算を行います。結果は汎用レジスタ**rd**に格納します。**

オペレーション：

```
32，64　T：　GPR[rd] ← GPR[rs] xor GPR[rt]
```

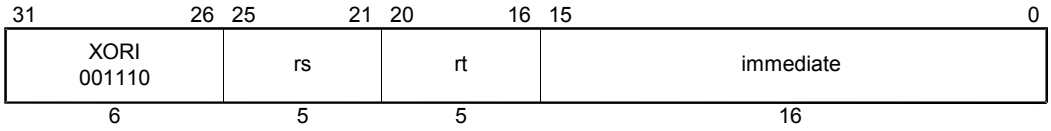
例　　外：

なし

保守／廃止

XORI

Exclusive Or Immediate



命　令　形　式：

XORI　rt，rs，immediate

説　　明：

ゼロ拡張した16ビットのimmediateと汎用レジスタrsの内容の，ビットごとの排他的論理和演算を行います。結果は汎用レジスタrtに格納します。

オペレーション：

```
32　T：　GPR[rt] ← GPR[rs] xor（0^16 ‖ immediate）

64　T：　GPR[rt] ← GPR[rs] xor（0^48 ‖ immediate）
```

例　　外：

なし

A.6　CPU 命令オペコード符号

図A-1に，VR4120Aコアの命令オペコード（ISAおよび拡張ISA）の符号表を示します。

図 A-1　CPU 命令オペコード符号表（1/2）

Opcode

31.. 29 \ 28.. 26	0	1	2	3	4	5	6	7
0	SPECIAL	REGIMM	J	JAL	BEQ	BNE	BLEZ	BGTZ
1	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
2	COP0	π	π	$*$	BEQL	BNEL	BLEZL	BGTZL
3	DADDI ε	DADDIU ε	LDL ε	LDR ε	$*$	JALX θ	$*$	$*$
4	LB	LH	LWL	LW	LBU	LHU	LWR	LWU ε
5	SB	SH	SWL	SW	SDL ε	SDR ε	SWR	CACHE δ
6	$*$	π	π	$*$	$*$	π	π	LD ε
7	$*$	π	π	$*$	$*$	π	π	SD ε

SPECIAL function

5.. 3 \ 2.. 0	0	1	2	3	4	5	6	7
0	SLL	$*$	SRL	SRA	SLLV	$*$	SRLV	SRAV
1	JR	JALR	$*$	$*$	SYSCALL	BREAK	$*$	SYNC
2	MFHI	MTHI	MFLO	MTLO	DSLLV ε	$*$	DSRLV ε	DSRAV ε
3	MULT	MULTU	DIV	DIVU	DMULT ε	DMULTU ε	DDIV ε	DDIVU ε
4	ADD	ADDU	SUB	SUBU	AND	OR	XOR	NOR
5	MACC	DMACC	SLT	SLTU	DADD ε	DADDU ε	DSUB ε	DSUBU ε
6	TGE	TGEU	TLT	TLTU	TEQ	$*$	TNE	$*$
7	DSLL ε	$*$	DSRL ε	DSRA ε	DSLL32 ε	$*$	DSRL32 ε	DSRA32 ε

REGIMM rt

20.. 19 \ 18.. 16	0	1	2	3	4	5	6	7
0	BLTZ	BGEZ	BLTZL	BGEZL	$*$	$*$	$*$	$*$
1	TGEI	TGEIU	TLTI	TLTIU	TEQI	$*$	TNEI	$*$
2	BLTZAL	BGEZAL	BLTZALL	BGEZALL	$*$	$*$	$*$	$*$
3	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$

COP0 rs

25.. 24 \ 23.. 21	0	1	2	3	4	5	6	7
0	MF	DMF ε	γ	γ	MT	DMT ε	γ	γ
1	BC	γ	γ	γ	γ	γ	γ	γ
2	CO							
3								

保守／廃止

図A-1　CPU命令オペコード符号表（2/2）

COP0 rt

20..19 \ 18..16	0	1	2	3	4	5	6	7
0	BCF	BCT	BCFL	BCTL	γ	γ	γ	γ
1	γ	γ	γ	γ	γ	γ	γ	γ
2	γ	γ	γ	γ	γ	γ	γ	γ
3	γ	γ	γ	γ	γ	γ	γ	γ

CP0 function

5..3 \ 2..0	0	1	2	3	4	5	6	7
0	φ	TLBR	TLBWI	φ	φ	φ	TLBWR	φ
1	TLBP	φ	φ	φ	φ	φ	φ	φ
2	ξ	φ	φ	φ	φ	φ	φ	φ
3	ERET χ	φ	φ	φ	φ	φ	φ	φ
4	φ	STANDBY	SUSPEND	HIBERNATE	φ	φ	φ	φ
5	φ	φ	φ	φ	φ	φ	φ	φ
6	φ	φ	φ	φ	φ	φ	φ	φ
7	φ	φ	φ	φ	φ	φ	φ	φ

備考　**図中の記号の意味は次のとおりです。**

＊：アスタリスクのついたオペレーション・コードを現在の VR4120A **コアで実行すると，予約命令例外が発生します。このコードは将来の拡張用に予約されています。**

γ **：ガンマのついたオペレーション・コードは，予約命令例外を発生します。このコードは将来の拡張用に予約されています。**

δ **：デルタのついたオペレーション・コードは** CP0 **が許可された** VR4400 **シリーズのみに有効です。他のプロセッサでは予約命令例外が発生します。**

φ **：ファイのついたオペレーション・コードは無効ですが，**VR4120A **コアでは予約命令例外を発生しません。**

ξ **：クシーのついたオペレーション・コードは，**VR4120A **コアでは予約命令例外が発生します。**

χ **：カイのついたオペレーション・コードは，**VR4000 **シリーズのみ有効です。**

ε **：イプシロンのついたオペレーション・コードは，**64 **ビット・モードと** 32 **ビット・カーネル・モード時に有効です。**32 **ビット・ユーザ／スーパバイザ・モード時は，予約命令例外が発生します。**

π **：パイのついたオペレーション・コードは無効ですが，**VR4120A **コアではコプロセッサ使用不可例外を発生します。**

θ **：シータのついたオペレーション・コードは，**MIPS16 **命令実行許可時に有効です。**MIPS16 **命令の実行が許可されていない場合は，予約命令例外が発生します（**μPD98502 **では** MIPS16 **モードをサポートしていません）。**

付録B　VR4120Aコプロセッサ0ハザード

VR4120A コアは，各命令間で内部リソースの競合が生じた場合（デスティネーション・レジスタの内容を次の命令でソースとして使う場合など），パイプラインのインタロックを発生し，内部リソースの競合を回避します。このため，NOP などの命令を間に入れる必要はありません。

しかし，CP0 レジスタ，TLB に関してはインタロックを発生しません。このため，CP0 レジスタ，TLB を扱うプログラムを作成するときは，内部リソースの競合を考慮してください。CP0 ハザードとは，このような内部リソースの競合を回避するために命令間に挿入する NOP 命令，または競合に無関係な命令の数を規定するものです。この節では，この CP0 ハザードについて説明します。

VR4120A コアの CP0 ハザードの値は VR4000 と同程度か，それよりも減少しています。表 B－1 に VR4120A コアの CP0 ハザードを示します。これらのハザードに準拠したプログラムならば，そのまま VR4000 シリーズ上で実行できます。

表のソース欄にある CP0 レジスタかビットのデータが確定すると，そのデータをソースとして使用できます。

デスティネーション欄にある CP0 レジスタかビットにデータが格納されると，そのデータがデスティネーションになります。

CP0 レジスタ，TLB に関する命令間の NOP 命令，または競合に無関係な命令の数は，この表から次の式で計算できます。プログラム上で必要な命令数を管理してください。

（命令 A のデスティネーションのハザード数）－（（命令 B のソースのハザード数）＋1）

例　MTC0 命令と，そのあとに実行する MFC0 命令の間に必要な命令数

（5）－（3＋1）＝1 命令

表B－1　CP0ハザード

オペレーション	ソース		デスティネーション	
	名　　称	ハザード数	名　　称	ハザード数
MTC0	－		CPU の汎用レジスタ	5
MFC0	CPU の汎用レジスタ	3	－	
TLBR	インデクス，TLB	2	ページ・マスク，エントリ Hi，エントリ Lo0，エントリ Lo1	5
TLBWI TLBWR	インデクスまたはランダム ページ・マスク，エントリ Hi， エントリ Lo0，エントリ Lo1	2	TLB	5
TLBP	ページ・マスク，エントリ Hi	2	インデクス	6
ERET	EPC またはエラーEPC, TLB	2	ステータス［EXL］，［ERL］	4
	ステータス	2	－	
CACHE Index_Load_Tag	－		タグ Lo，タグ Hi，パリティ・エラー	5
CACHE Index_Store_Tag	タグ Lo，タグ Hi，パリティ・エラー	3	－	
CACHE Hit_ops.	キャッシュ・ライン	3	キャッシュ・ライン	5
コプロセッサ使用可テスト	ステータス［CU］，［KSU］，［EXL］，［ERL］	2	－	
命令フェッチ	エントリ Hi［ASID］ ステータス［KSU］，［EXL］，［ERL］，［RE］， コンフィグ［K0］	2	－	
	TLB	2	－	
命令フェッチ例外	－		EPC，ステータス	4
			原因，BadVAddr，コンテキスト，X コンテキスト	5
割り込み	原因［IP］，ステータス［IM］ ステータス［IE］，［EXL］ ステータス［ERL］	2	－	
ロード／ストア	エントリ Hi［ASID］ ステータス［KSU］，［EXL］，［ERL］，［RE］， コンフィグ［K0］，TLB	3	－	
	コンフィグ［AD］，［EP］	3		
	ウォッチ Hi，ウォッチ Lo	3		
ロード／ストア例外	－		EPC，ステータス，原因，BadVAddr，コンテキスト，X コンテキスト	5

備考　［ ］内はレジスタ内のビット名または領域名を示します。

保守／廃止

注意 1. MTC0 命令でコンフィグ・レジスタの K0 ビットを非キャッシュ・モードに変更した場合，MTC0 命令の 2 命令後の命令フェッチから非キャッシュ領域に移行します。

2. MTC0 命令のあとに MFC0 命令を続けて実行しないでください。

3. KSU ビットを変更し EXL ビットと ERL ビットをセットする，ステータス・レジスタへの MTC0 命令のあとに続く 5 つの命令は，カーネル・モードではなく新しいモードで実行されることがあります。このことを避けるためには，EXL ビットを最初にセットし，KSU ビットがカーネルにセットされるのを待ち，そのあとに KSU ビットを変更します。

4. ストア先として同じキャッシュ・ラインを使用するストア命令と CACHE 命令の間に，ロードでも CACHE でもない命令が 2 つ必要です。

次に各命令実行時で，CP0 ハザードを考慮する必要がある状態を示します。

（1）MTC0

デスティネーション：MTC0 命令のデスティネーション・レジスタ（CP0）への書き込み完了

（2）MFC0

ソース：MFC0 命令のソース・レジスタ（CP0）の確定

（3）TLBR

ソース：TLBR 命令実行前の TLB 状態とインデクス・レジスタの確定

デスティネーション：TLBR 命令のデスティネーション・レジスタ（CP0）への書き込み完了

（4）TLBWI, TLBWR

ソース：これらの命令のソース・レジスタと TLB エントリ指定に使用するレジスタの確定

デスティネーション：これらの命令による TLB への書き込み完了

（5）TLBP

ソース：TLBP 命令実行前のページ・マスク・レジスタとエントリ Hi レジスタの確定

デスティネーション：TLBP 命令実行結果のインデクス・レジスタへの書き込み完了

（6）ERET

ソース：ERET 命令実行に必要な情報を保持するレジスタの確定

デスティネーション：ERET 命令実行による VR4120A コア状態の移行完了

（7）CACHE Index_Load_Tag

デスティネーション：この命令の実行結果の各レジスタへの書き込み完了

（8）CACHE Index_Store_Tag

ソース：この命令の実行に必要な情報を保持するレジスタの確定

(9) コプロセッサ使用可テスト

ソース：ソース欄にある CP0 レジスタのビット値で設定したモードの確定

例 1. ステータス・レジスタの CU0 ビットの内容変更後，ユーザ・モードで CP0 レジスタにアクセスする場合か，CP0 のリソースを使用する命令（TLB 命令，CACHE 命令，ブランチ命令など）を実行する場合

2. ステータス・レジスタの KSU, EXL, ERL ビットの内容変更後，その動作モードで CP0 レジスタにアクセスする場合

(10) 命令フェッチ

ソース：命令フェッチに必要な動作モード，TLB などの確定

例 1. ステータス・レジスタの KSU, EXL, ERL ビットの内容変更後，ユーザ・モードからカーネル・モードに変更して命令をフェッチする場合

2. TLB を書き換えて，その TLB エントリを使用して命令をフェッチする場合

(11) 命令フェッチ例外

デスティネーション：命令フェッチによる例外が発生した場合で，例外に関する情報を保持する各レジスタへの書き込み完了

(12) 割り込み

ソース：割り込み要因発生時に例外発生条件を判断する各レジスタの確定

(13) ロード／ストア

ソース：ロード／ストア命令のアドレス生成に関する動作モードの確定，TLB エントリの確定，コンフィグ・レジスタの K0 ビットで設定するキャッシュ・モードの確定，ウォッチ例外の発生条件を設定するレジスタの確定

例　ユーザ・モードからカーネル・モードに変更後のカーネル領域でのロード／ストア命令

(14) ロード／ストア例外

デスティネーション：ロード／ストア動作によって例外が発生した場合で，例外に関する情報を保持する各レジスタへの書き込み完了

表 B－2 に計算例を示します。

保守／廃止

表B－2　CP0ハザードと挿入命令数の計算例

デスティネーション	ソース	競合する内部リソース	挿入命令数	計算式
TLBWR/TLBWI	TLBP	TLB エントリ	2	5－（2＋1）
TLBWR/TLBWI	新しく書き換えられた TLB を使用するロード／ストア	TLB エントリ	1	5－（3＋1）
TLBWR/TLBWI	新しく書き換えられた TLB を使用する命令フェッチ	TLB エントリ	2	5－（2＋1）
ステータス・レジスタ［CU］に対する MTC0	CU のセットを必要とするコプロセッサ命令	ステータス・レジスタ［CU］	2	5－（2＋1）
TLBR	エントリ Hi レジスタに対する MFC0	エントリ Hi レジスタ	1	5－（3＋1）
エントリ Lo0 レジスタに対する MTC0	TLBWR/TLBWI	エントリ Lo0 レジスタ	2	5－（2＋1）
TLBP	インデクス・レジスタに対する MFC0	インデクス・レジスタ	2	6－（3＋1）
エントリ Hi レジスタに対する MTC0	TLBP	エントリ Hi レジスタ	2	5－（2＋1）
EPC レジスタに対する MTC0	ERET	EPC レジスタ	2	5－（2＋1）
ステータス・レジスタに対する MTC0	ERET	ステータス・レジスタ	2	5－（2＋1）
ステータス・レジスタ［IE］に対する MTC0 注	割り込みの要因となる命令	ステータス・レジスタ［IE］	2	5－（2＋1）

注　命令実行順序が例外によって変わった場合，IE ビットの値が確定するまでの最小ハザード数と，保留中で許可された割り込み要求が発生するまでの最大ハザード数が同じになることがあります。

備考　［ ］内はレジスタ内のビット名または領域名を示します。

保守／廃止

【発　行】

NECエレクトロニクス株式会社

〒211-8668　神奈川県川崎市中原区下沼部1753

電話（代表）：044(435)5111

【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。

URL(アドレス)　http://www.necel.co.jp/

【営業関係お問い合わせ先】

下記のページに最新版のお問い合わせ先が記載されています。

URL(アドレス)　http://www.necel.com/ja/contact/contact_j.html

【技術的なお問い合わせ先】

半導体テクニカルホットライン

（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−435−9494
FAX　：044−435−9608
E-mail　：info@lsi.nec.co.jp

【資料請求先】

NECエレクトロニクス特約店または上記ホームページ記載の営業関係お問い合わせ先へお申し付けください。