序文

まえがき

マニュアルについて

TMS320C54xはTM320ファミリの固定小数点ディジタル シグナル プロセッサ(DSP)です。この本書は、TMS320C54xDSPに関する情報を提供することを目的とした4分冊の1番目になります。また、ハードウェア/ソフトウェアを開発するためのリファレンスとしても活用していただけます。特別な記述が無い限り、'54xはTMS320C54x、TMS320LC45xおよびTMS320VC54xを示します。

マニュアルの使用方法

下の表は本書に含まれる'54x情報をまとめて示しています。

探したい情報	参照すべき章
'54x概要	1章　はじめに
CPUアーキテクチャ	2章　アーキテクチャ概要 4章　中央演算処理ユニット
TDMシリアル ポート	9章　シリアル ポート
アドレッシング モード	5章　データ アドレッシング 6章　プログラム メモリ アドレッシング
ウェイト ステート生成器	2章　アーキテクチャ概要 8章　オンチップ ペリフェラル
オンチップ ペリフェラル	8章　オンチップ ペリフェラル
外部バス	10章　外部バス機能
クロック生成器	2章　アーキテクチャ概要 8章　オンチップ ペリフェラル
シリアル ポート	9章　シリアル ポート
ステータス レジスタ	4章　中央演算処理ユニット 付録A　CPUおよびペリフェラル レジスタ
タイマ	2章　アーキテクチャ概要 8章　オンチップ ペリフェラル

マニュアルの使用方法

探したい情報	参照すべき章
パイプライン レイテンシ	2章　アーキテクチャ概要 7章　パイプライン
バス構成	2章　アーキテクチャ概要
バッファド シリアル ポート	9章　シリアル ポート
パラレルI/O**ポート**	2章　アーキテクチャ概要 8章　オンチップ ペリフェラル
パワーダウン モード	6章　プログラム メモリ アドレッシング
ブートローダ	3章　メモリ
プログラム制御	6章　プログラム メモリ アドレッシング
ホスト ポート インターフェイス	8章　オンチップ ペリフェラル
ホールド モード	10章　外部バス機能
メモリ	2章　アーキテクチャ概要 3章　メモリ
リセット	6章　プログラム メモリ アドレッシング
割り込み	6章　プログラム メモリ アドレッシング

書体とシンボルの定義

本文中では以下の定義にしたがって説明をすすめています。

- プラグラム一覧、プログラム例は下記の例のような特殊文字を使用しています。

```
OUTPUT:
      LDP      #0          ;data page 0
      RPT      #63         ;Output 64 values form a table at 800h
       LMMR    50h,800h    ;in data memory to port 50h.
      RET
```

- 文法に関する記述では、命令を太文字、パラメータをイタリック文字で記述しています。太文字部分は記述通りに、イタリック文字部分は任意のデータを記述して下さい。以下は、命令の文法例です。

  [*label*] **BLDD** *src, dst*

  BLDD命令は、srcとdstで示される2つのパラメータをもっています。BLDD命令を使う場合、一番目のパラメータにはソースとなるメモリのアドレスを、二番目のパラメータにはディスティネーションとなるメモリのアドレスを書きます。コンマとスペースでこの2つのアドレスを区切って下さい。

- [ ]は、オプションのパラメータ入力です。オプションを選択する場合は、選択するオプションを書いて下さい。括弧を書く必要はありません。前述の例の場合、[*label*]と書く代わりに任意のラベル名を書いて下さい。2つ以上のオプション パラメータを選択する場合は、それらをコンマとスペースで区切って下さい。

- { }は選択肢を示します。シンボル | によって、括弧内の項目は分類されます。下記は選択肢の例です。

  ```
  ind: { * | *+ | *• | *0+ | *0• | *BRO+ | *BRO•}
  ```

  これは、七者択一の例です。

  リストが[ ]で囲まれない場合は、選択肢の中から1項目を選ばなくてはいけません。

注記および警告の説明

この本には注記と警告があります。

- **注記ではデバイス使用上有効な方法や推奨される使用方法などを記述します。**

> **注 :**
>
> **注記の記述例。**

- **警告ではユーザのソフトウェアや装置に大きな損害をもたらす可能性のある状態を記述します。**

警告の記述例。

注記や警告での情報はユーザの装置の保護のためのものです。各注記および警告を注意深く読んで下さい。

テキサス インスツルメンツから供給されている関連文書

以下の文書は'54xとそのツールについて説明されているものです。

TMS320C54x DSP Reference Set, Volume 1: CPU and Peripherals TMS320C54x 16ビット固定小数点汎用DSPについて記述されています。アーキテクチャ、内部レジスタ構造、データおよびプログラム アドレッシング、命令、パイプライン、オンチップ ペリフェラルをカバーしており、その他開発サポート情報も含んでいます。

TMS320C54x DSP Reference Set, Volume 2: Mnemonic Instruction Set TMS320C54x DSPのニモニック命令を個別に説明しています。命令セットと動作サイクル数の一覧も含まれています。

TMS320C54x DSP Reference Set, Volume 3: Algebraic Instruction Set TMS320C54x DSPの代数表記命令を個別に説明しています。命令セットと動作サイクル数の一覧も含まれています。

TMS320C54x DSP Reference Set, Volume 4: Applications Guide TMS320C54xのソフトウェアとハードウェアアプリケーションについて記述されています。その他開発の際の有益な情報も含まれています。

TMS320C54x, TMS320LC54x, TMS320VC54x Fixed-Point Digital Signal '54xのデータシートです。全てのパッケージの電気的特性、タイミング、信号説明、ピン配置等が記述されています。

TMS320C54x Assembly Language Tools User's Guide '54xシリーズのアセンブリ言語ツール等に関するユーザーズガイドです。

TMS320C5xx C Source Debugger User's Guide '54xエミュレータ、EVM、シミュレータに関するユーザーズガイドです。

TMS320C54x Evaluation Module Technical Reference 'C54x EVMのユーザーズガイドです。

TMS320C54x Optimizing C Compiler User's Guide '54xシリーズのCコンパイラについてのユーザーズガイドです。

TMS320 Third-Party Support Reference Guide TMS320ファミリの100を超えるサードパーティ(協力会社)をリストアップ、紹介しています。主にハードウェアをサポートしている会社を紹介しています。

TMS320 Software Cooperative Resource Guide TMS320ファミリのサードパーティの内、ソフトウェアを共給している会社を紹介しています。

技術文書

ディジタル信号処理に関する広範囲の関連文書が各社より発行されています。それらは以下の応用分野に分けられます。

- ❑ **汎用**DSP
- ❑ **グラフィック/イメージング**
- ❑ **スピーチ/音声**
- ❑ **制御**
- ❑ **マルチメディア**
- ❑ **情報通信**
- ❑ **自動車**
- ❑ **民生機器**
- ❑ **医療**
- ❑ **開発サポート**

汎用DSP

1) Chassaing, R., Horning, D.W., *"Digital Signal Processing with Fixed and Floating-Point Processors"* , CoED, USA, Volume 1, Number 1, pages 1-4, March 1991.

2) Defatta, David J., Joseph G. Lucas, and William S. Hodgkiss, *Digital Signal Processing: A System Design Approach*, New York: John Wiley, 1988.

3) Erskine, C., and S. Magar, "Architecture and Applications of a Second-Generation Digital Signal Processor," *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, USA, 1985.

4) Essig, D., C. Erskine, E. Caudel, and S. Magar, "A Second-Generation Digital Signal Processor," *IEEE Journal of Solid-State Circuits*, USA, Volume SC-21, Number 1, pages 86-91, February 1986.

5) Frantz, G., K. Lin, J. Reimer, and J. Bradley, "The Texas Instruments TMS320C25 Digital Signal Microcomputer," *IEEE Microelectronics*, USA, Volume 6, Number 6, pages 10-28, December 1986.

6) Gass, W., R. Tarrant, T. Richard, B. Pawate, M. Gammel, P. Rajasekaran, R. Wiggins, and C. Covington, "Multiple Digital Signal Processor Environment for Intelligent Signal Processing," *Proceedings of the IEEE, USA*, Volume 75, Number 9, pages 1246-1259, September 1987.

7) Jackson, Leland B., *Digital Filters and Signal Processing*, Hingham, MA: Kluwer Academic Publishers, 1986.

8) Jones, D.L., and T.W. Parks, *A Digital Signal Processing Laboratory Using the TMS32010*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1987.

9) Lim, Jae, and Alan V. Oppenheim, *Advanced Topics in Signal Processing*, Englewood Cliffs, NJ: Prentice- Hall, Inc., 1988.

10) Lin, K., G. Frantz, and R. Simar, Jr., "The TMS320 Family of Digital Signal Processors," *Proceedings of the IEEE*, USA, Volume 75, Number 9, pages 1143-1159, September 1987.

11) Lovrich, A., Reimer, J., *"An Advanced Audio Signal Processor"* , Digest of Technical Papers for 1991 International Conference on Consumer Electronics, June 1991.

12) Magar, S., D. Essig, E. Caudel, S. Marshall and R. Peters, "An NMOS Digital Signal Processor with Multiprocessing Capability," *Digest of IEEE International Solid-State Circuits Conference*, USA, February 1985.

13) Oppenheim, Alan V., and R.W. Schafer, *Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975 and 1988.

14) Papamichalis, P.E., and C.S. Burrus, "Conversion of Digit-Reversed to Bit-Reversed Order in FFT Algorithms," *Proceedings of ICASSP 89*, USA, pages 984-987, May 1989.

15) Papamichalis, P., and R. Simar, Jr., "The TMS320C30 Floating-Point Digital Signal Processor," *IEEE Micro Magazine*, USA, pages 13-29, December 1988.

16) Papamichalis, P.E., "FFT Implementation on the TMS320C30," *Proceedings of ICASSP 88*, USA, Volume D, page 1399, April 1988.

17) Parks, T.W., and C.S. Burrus, *Digital Filter Design*, New York, NY: John Wiley and Sons, Inc., 1987.

18) Peterson, C., Zervakis, M., Shehadeh, N., *"Adaptive Filter Design and Implementation Using the TMS320C25 Microprocessor"*, Computers in Education Journal, USA, Volume 3, Number 3, pages 12-16, July-September 1993.

19) Prado, J., and R. Alcantara, “A Fast Square-Rooting Algorithm Using a Digital Signal Processor,” *Proceedings of IEEE*, USA, Volume 75, Number 2, pages 262-264, February 1987.

20) Rabiner, L.R. and B. Gold, *Theory and Applications of Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975.

21) Simar, Jr., R., and A. Davis, “The Application of High-Level Languages to Single-Chip Digital Signal Processors,” *Proceedings of ICASSP 88*, USA, Volume D, page 1678, April 1988.

22) Simar, Jr., R., T. Leigh, P. Koeppen, J. Leach, J. Potts, and D. Blalock, “A 40 MFLOPS Digital Signal Processor: the First Supercomputer on a Chip,” *Proceedings of ICASSP 87*, USA, Catalog Number 87CH2396-0, Volume 1, pages 535-538, April 1987.

23) Simar, Jr., R., and J. Reimer, “The TMS320C25: a 100 ns CMOS VLSI Digital Signal Processor,” *1986 Workshop on Applications of Signal Processing to Audio and Acoustics*, September 1986.

24) Texas Instruments, *Digital Signal Processing Applications with the TMS320 Family*, 1986; Englewood Cliffs, NJ: Prentice-Hall, Inc., 1987.

25) Treichler, J.R., C.R. Johnson, Jr., and M.G. Larimore, *A Practical Guide to Adaptive Filter Design*, New York, NY: John Wiley and Sons, Inc., 1987.

グラフィック/イメージング

1) Reimer, J., and A. Lovrich, “Graphics with the TMS32020,” *WESCON/85 Conference Record*, USA, 1985.

スピーチ/音声

1) DellaMorte, J., and P. Papamichalis, “Full-Duplex Real-Time Implementation of the FED-STD-1015 LPC-10e Standard V.52 on the TMS320C25,” *Proceedings of SPEECH TECH 89*, pages 218-221, May 1989.

2) Gray, A.H., and J.D. Markel, *Linear Prediction of Speech*, New York, NY: Springer-Verlag, 1976.

3) Frantz, G.A., and K.S. Lin, “A Low-Cost Speech System Using the TMS320C17,” *Proceedings of SPEECH TECH ’87*, pages 25-29, April 1987.

4) Papamichalis, P., and D. Lively, “Implementation of the DOD Standard LPC-10/52E on the TMS320C25,” *Proceedings of SPEECH TECH ’87*, pages 201-204, April 1987.

5) Papamichalis, Panos, *Practical Approaches to Speech Coding*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1987.

6) Pawate, B.I., and G.R. Doddington, "Implementation of a Hidden Markov Model-Based Layered Grammar Recognizer," *Proceedings of ICASSP 89*, USA, pages 801-804, May 1989.

7) Rabiner, L.R., and R.W. Schafer, *Digital Processing of Speech Signals*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1978.

8) Reimer, J.B. and K.S. Lin, "TMS320 Digital Signal Processors in Speech Applications," *Proceedings of SPEECH TECH '88*, April 1988.

9) Reimer, J.B., M.L. McMahan, and W.W. Anderson, "Speech Recognition for a Low-Cost System Using a DSP," *Digest of Technical Papers for 1987 International Conference on Consumer Electronics*, June 1987.

制御

1) Ahmed, I., "16-Bit DSP Microcontroller Fits Motion Control System Application," *PCIM*, October 1988.

2) Ahmed, I., "Implementation of Self Tuning Regulators with TMS320 Family of Digital Signal Processors," *MOTORCON '88*, pages 248-262, September 1988.

3) Allen, C. and P. Pillay, *"TMS320 Design for Vector and Current Control of AC Motor Drives"* , Electronics Letters, UK, Volume 28, Number 23, pages 2188-2190, November 1992.

4) Panahi, I. and R. Restle, *"DSPs Redefine Motion Control"* , Motion Control Magazine, December 1993.

5) Lovrich, A., G. Troullinos, and R. Chirayil, "An All-Digital Automatic Gain Control," *Proceedings of ICASSP 88*, USA, Volume D, page 1734, April 1988.

6) Ahmed, I., and S. Meshkat, "Using DSPs in Control," *Control Engineering*, February 1988.

7) Meshkat, S., and I. Ahmed, "Using DSPs in AC Induction Motor Drives," *Control Engineering*, February 1988.

8) Matsui, N. and M. Shigyo, *"Brushless DC Motor Control Without Position and Speed Sensors"* , IEEE Transactions on Industry Applications, USA, Volume 28, Number 1, Part 1, pages 120-127, January-February 1992.

9) Hanselman, H., "LQG-Control of a Highly Resonant Disc Drive Head Positioning Actuator," *IEEE Transactions on Industrial Electronics*, USA, Volume 35, Number 1, pages 100-104, February 1988.

10) Bose, B.K., and P.M. Szczesny, “A Microcomputer-Based Control and Simulation of an Advanced IPM Synchronous Machine Drive System for Electric Vehicle Propulsion,” *Proceedings of IECON ’87*, Volume 1, pages 454-463, November 1987.

11) Ahmed, I., and S. Lindquist, “Digital Signal Processors: Simplifying High-Performance Control,” *Machine Design*, September 1987.

マルチメディア

1) Reimer, J., *“DSP-Based Multimedia Solutions Lead Way Enhancing Audio Compression Performance”* , Dr. Dobbs Journal, December 1993.

2) Reimer, J., G. Benbassat, and W. Bonneau Jr., *“Application Processors: Making PC Multimedia Happen”* , Silicon Valley PC Design Conference, July 1991.

情報通信

1) Ahmed, I., and A. Lovrich, “Adaptive Line Enhancer Using the TMS320C25,” *Conference Records of Northcon/86*, USA, 14/3/1-10, September/October 1986.

2) Casale, S., R. Russo, and G. Bellina, “Optimal Architectural Solution Using DSP Processors for the Implementation of an ADPCM Transcoder,” *Proceedings of GLOBECOM ’89*, pages 1267-1273, November 1989.

3) Cole, C., A. Haoui, and P. Winship, “A High-Performance Digital Voice Echo Canceller on a SINGLE TMS32020,” *Proceedings of ICASSP 86*, USA, Catalog Number 86CH2243-4, Volume 1, pages 429-432, April 1986.

4) Cole, C., A. Haoui, and P. Winship, “A High-Performance Digital Voice Echo Canceller on a Single TMS32020,” *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, USA, 1986.

5) Lovrich, A., and J. Reimer, “A Multi-Rate Transcoder,” *Transactions on Consumer Electronics*, USA, November 1989.

6) Lovrich, A. and J. Reimer, *“A Multi-Rate Transcoder”* , Digest of Technical Papers for 1989 International Conference on Consumer Electronics, June 7-9, 1989.

7) Lu, H., D. Hedberg, and B. Fraenkel, “Implementation of High-Speed Voiceband Data Modems Using the TMS320C25,” *Proceedings of ICASSP 87*, USA, Catalog Number 87CH2396-0, Volume 4, pages 1915-1918, April 1987.

8) Mock, P., “Add DTMF Generation and Decoding to DSP- mP Designs,” *Electronic Design*, USA, Volume 30, Number 6, pages 205-213, March 1985.

9) Reimer, J., M. McMahan, and M. Arjmand, “ADPCM on a TMS320 DSP Chip,” *Proceedings of SPEECH TECH 85*, pages 246-249, April 1985.

10) Troullinos, G., and J. Bradley, “Split-Band Modem Implementation Using the TMS32010 Digital Signal Processor,” *Conference Records of Electro/86 and Mini/Micro Northeast*, USA, 14/1/1-21, May 1986.

自動車

1) Lin, K., “Trends of Digital Signal Processing in Automotive,” *International Congress on Transportation Electronic (CONVERGENCE ’88)*, October 1988.

民生機器

1) Frantz, G.A., J.B. Reimer, and R.A. Wotiz, “Julie, The Application of DSP to a Product,” *Speech Tech Magazine*, USA, September 1988.

2) Reimer, J.B., and G.A. Frantz, “Customization of a DSP Integrated Circuit for a Customer Product,” *Transactions on Consumer Electronics*, USA, August 1988.

3) Reimer, J.B., P.E. Nixon, E.B. Boles, and G.A. Frantz, “Audio Customization of a DSP IC,” *Digest of Technical Papers for 1988 International Conference on Consumer Electronics*, June 8-10 1988.

医療

1) Knapp and Townshend, “A Real-Time Digital Signal Processing System for an Auditory Prosthesis,” *Proceedings of ICASSP 88*, USA, Volume A, page 2493, April 1988.

2) Morris, L.R., and P.B. Barszczewski, “Design and Evolution of a Pocket-Sized DSP Speech Processing System for a Cochlear Implant and Other Hearing Prosthesis Applications,” *Proceedings of ICASSP 88*, USA, Volume A, page 2516, April 1988.

開発ツール

1) Mersereau, R., R. Schafer, T. Barnwell, and D. Smith, “A Digital Filter Design Package for PCs and TMS320,” *MIDCON/84 Electronic Show and Convention*, USA, 1984.

2) Simar, Jr., R., and A. Davis, “The Application of High-Level Languages to Single-Chip Digital Signal Processors,” *Proceedings of ICASSP 88*, USA, Volume 3, pages 1678-1681, April 1988.

登録商標

製品に関する問い合わせ先

日本TIプロダクト インフォメーション センター(PIC)
TEL. 0120−81−0026
FAX. 0120−81−0036

World Wide Web とFTPサイト

インターネットのホームページでは、DSP製品をはじめ、あらゆる製品の情報を提供しています。

- ❑ **米国TIウェブ ホームページ**: http://www.ti.com
- ❑ **日本TIウェブ ホームページ**: http://www.tij.co.jp
- ❑ **米国TI FTPサイト**: ftp.ti.com(192.94.94.5)
  - User ID: anonymous
  - Subdirectory: /pub/mirrors

目次

図

図

表

表

例

第1章

1

はじめに

TMS320C54xデバイスは、TMS320製品ファミリーの固定小数点デジタル シグナル プロセッサ(DSP)です。'54xは、遠隔通信などの特有なリアルタイムの組み込みアプリケーションのための製品です。'54x中央処理演算ユニット(CPU)は改良型ハーバード アーキテクチャを持ち、省電力化機能および高度な並列機能を備えています。さらに多彩なアドレッシングモードおよび命令セットにより、システムの全般的な機能を向上させています。

Topic **Page**

1

1.1 TMS320製品ファミリー概要

TMS320製品ファミリーは、固定小数点、浮動小数点、マルチプロセッサ型のディジタル シグナル プロセッサ(DSP)から構成されています。TMS320のアーキテクチャは、特にリアルタイム プロセッシングのために設計されています。次のような特性により、TMS320製品ファミリーは幅広いプロセッシング アプリケーションにとって理想的なデバイスになっています。

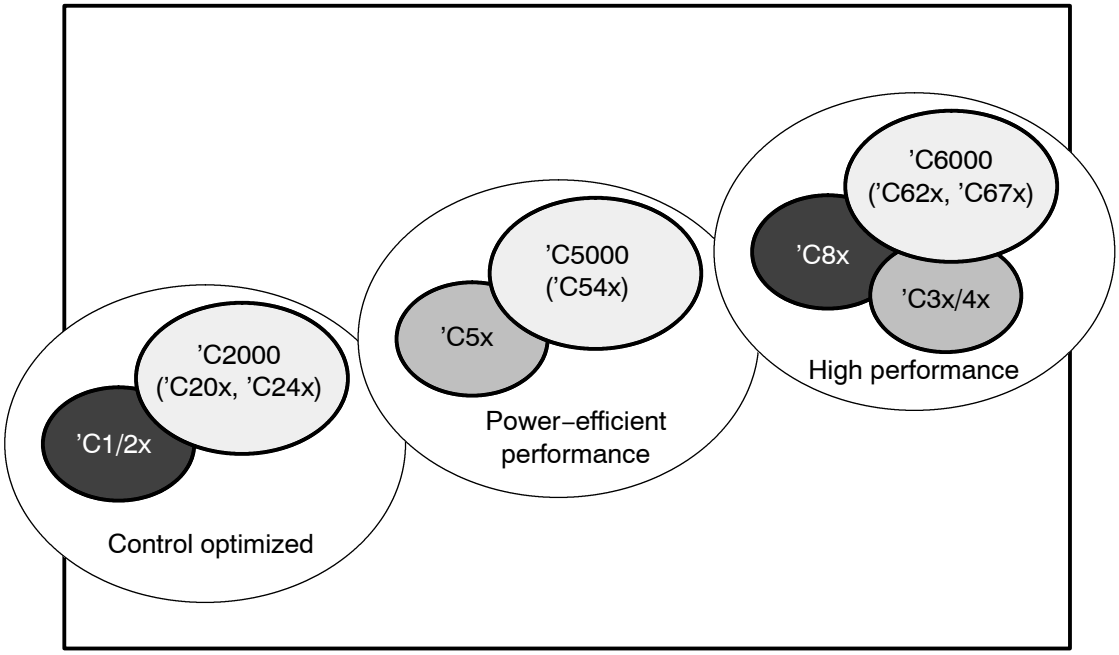
- ❑ 柔軟性の高い命令セット
- ❑ 動作の柔軟性
- ❑ 高速処理
- ❑ 革新的な並列アーキテクチャ
- ❑ 優れたコスト パフォーマンス
- ❑ Cとの親和性のあるアーキテクチャ

1.1.1 TMS320DSPの歴史、発展、優位性

テキサス インスツルメンツは、1982年にTMS32010というTMS320ファミリーの最初の固定小数点DSPを発表しました。その年の末には、TMS32010はElectronic Products誌から”Product of the Year”を受賞しています。現在TMS320ファミリーは、’C1x、’C2x、’C2xx、’C5x、’C54x、’C6xの各固定小数点DSP、’C3x、’C4xの浮動小数点DSP、’C8xのマルチプロセッサDSPという各世代の製品から構成されています。

TMS320ファミリーのそれぞれの世代の各デバイスは同じCPU構造を持ちますが、オンチップ メモリおよびペリフェラル構成は異なります。派生デバイスは、オンチップ メモリおよびペリフェラルの新しい組み合わせを使用して、世界のエレクトロニクス市場における幅広いニーズに対応しています。メモリおよびペリフェラルを単一のチップに統合することにより、TMS320デバイスはシステムのコストを削減して回路基板のスペースを節約します。図1−1は、TMS320ファミリーの性能順位を示しています。

図1−1. TMS320ファミリーの発展



1.1.2 TMS320ファミリーの代表的なアプリケーション

表1−1は、TMS320ファミリーDSPの代表的なアプリケーションを集めたものです。TMS320 DSPは、標準的なマイクロプロセッサ/マイクロコンピュータ デバイスに比べて、音声コーデック、フィルタリングなど従来のシグナル プロセッシングの問題に、より的確に対処できるように設計されています。TMS320 DSPは、多重演算の同時処理が必要な複雑なアプリケーションをサポートします。

表1−1. TMS320 DSPの代表的なアプリケーション

自動車利用	民生利用	制御利用
適応型運転制御 横滑り防止ブレーキ 携帯電話 デジタル ラジオ エンジン制御 ナビゲーションおよび走行位置確認 振動分析 音声命令 衝突防止レーダ	デジタル ラジオ/TV 教育玩具 音楽用シンセサイザ ポケベル パワー ツール レーダ検出 チップ録音留守番電話	ディスク ドライブ制御 エンジン制御 レーザ プリンタ制御 モータ制御 ロボット制御 サーボ制御
汎用利用	**グラフィックス/画像利用**	**産業利用**
適応型フィルタリング 重畳 相関 デジタル フィルタリング 高速フーリエ変換 ヒルベルト変換 波形生成 ウィンドウ生成	3−D回転 アニメーション/デジタル マップ 準同形プロセッシング 画像圧縮/転送 画像改善 パターン認識 ロボット視覚 ワークステーション	数値制御 電力ライン監視 ロボット セキュリティ アクセス
機器利用	**医療利用**	**軍事利用**
デジタル フィルタリング 機能生成 パターン適合 フェーズ ロック ループ 地震プロセッシング スペクトル分析 過度現象解析	診断機器 胎児モニタ 聴覚補助 患者モニタ 補装具 超音波機器	画像処理 ミサイル誘導 ナビゲーション レーダ プロセッシング 高周波モデム 高セキュリティ通信 ソナー プロセッシング
テレコミュニケーション(遠隔通信)利用		**音声利用**
1200bpsから33600bpsのモデム 適応イコライザ ADPCMトランスコーダ 携帯電話 チャネル多重化(マルチプレックス) データ暗号化 デジタル構内交換(PBX) デジタル音声補間(DSI) DTMFエンコード/デコード エコー キャンセレーション	ファックス通信 ライン リピータ パーソナル通信システム(PCS) パーソナル デジタル アシスタンス(PDA) スピーカ フォン 広帯域通信 ビデオ会議 X.25パケット交換	音声識別 音声改善 音声認識 音声合成 音声VOコード テキスト−音声変換 音声メール

1.2 TMS320C54x概要

’54xは、高度な動作柔軟性および処理速度を備えています。先進の改良型ハーバード アーキテクチャ(プログラム メモリ バス×1、データ メモリ バス×3、アドレス バス×4をともなう)、特定用途向けハードウェア ロジック、オンチップ メモリ、オンチップ ペリフェラルをともなうCPU、それに高度に特化された命令セットを組み合わせています。’54x CPUとオンチップ メモリ、ペリフェラルを組み合わせた派生デバイスは、エレクトロニクス市場の特定分野を対象に今後も開発が続きます。

’54xデバイスには、次のような優位性があります。

- ❑ プログラム バス×1、データ バス×3、アドレス バス×4から構成される改良型Harvardアーキテクチャにより、性能および汎用性を向上
- ❑ 高度な並列機能および特定用途向けハードウェア ロジックを備えた先進のCPU設計により、性能が向上
- ❑ 高度に特化された命令セットにより、高速なアルゴリズムを実現、高級言語処理に最適化
- ❑ モジュール アーキテクチャ設計により、派生デバイスの迅速な開発を実現
- ❑ 先進のICプロセッシング技術により、性能を向上しながら省電力化を実現
- ❑ 新しいスタティック設計技法により省電力化と高い放熱効率を実現

1

1.3 TMS320C54xの主な機能

'54xには、次のような主な機能があります。

❑ CPU

- プログラム バス×1、データ バス×3、アドレス バス×4を備えた先進のマルチバス アーキテクチャ
- 40ビット バレル シフタ×1、40ビット アキュムレータ×2を含む40ビット算術論理演算器(ALU)
- 40ビット専用の加算器と組み合わされた17ビット×17ビット並列乗算器で、ノンパイプラインの1サイクル積和演算(MAC)に対応
- ビタビ オペレータの加算/比較選択のための、比較選択ストアユニット(CSSU)
- べき指数エンコーダが40ビット アキュムレータ値のべき指数を1サイクルで演算
- 8個の補助レジスタと2個の補助レジスタ演算器を含んだ2つのアドレス生成器。

❑ メモリ

- 拡張プログラム メモリ(8Mワード)をともなう('548と'549)、192Kワード×16ビット アドレス可能メモリ空間(64Kワード プログラム、64Kワード データ、64KワードI/O)
- オンチップ メモリ構成を以下に示します。(単位:Kワード)

デバイス	プログラム ROM	プログラム/データROM	DARAM†	SARAM‡
'541	20	8	5	0
'542	2	0	10	0
'543	2	0	10	0
'545	32	16	6	0
'546	32	16	6	0
'548	2	0	8	24
'549	0	16	8	24

† デュアル アクセスRAM
‡ シングル アクセスRAM

❑ 命令セット

- 単一命令のリピートおよびブロック リピート実行
- プログラムおよびデータ管理を向上するブロック メモリ移動命令
- 32ビット ロング オペランド命令
- 2または3オペランド同時読み取り命令
- 並列ストアおよび並列ロードをともなう算術命令
- 条件付きストア命令
- 割り込みからの高速復帰

❑ オンチップ ペリフェラル

- ソフトウェアによるプログラマブルなウェイト ステート発生器
- プログラマブルなバンク スイッチング
- 内部オシレータまたは外部クロック ソースを入力とするオンチップのフェーズ ロックド ループ(PLL)クロック発生器。デバイスのオプションにより入力クロックが選択できます。

オプション1	オプション2	オプション3
1.0	1.0	ソフトウェア プログラマブルPLL†
1.5	4.0	
2.0	4.5	
3.0	5.0	

† '545A、'546A、'548、'549は、ソフトウェア プログラマブルPLLがあります。ソフトウェア プログラマブルPLLについては、8−17ページのサブセクション8.4.2、"ソフトウェア プログラマブルPLL"を参照してください。飽和モードは4.1.2章プロセッサ モード ステータス レジスタを参照して下さい。

各デバイスは、オプション リストから選ばれた1つのオプションのクロック モードを提供します。

- 外部データ バス、アドレス バス、制御信号をディセーブルにする外部バス オフ(bus−off)機能
- データ バスのバス ホルダ機能
- プログラマブル タイマ

■ ポート

デバイス	ホストポート インタフェース	シリアル ポート		
		同期	バッファド	時分割多重
’541	0	2	0	0
’542	1	0	1	1
’543	0	0	1	1
’545	1	1	1	0
’546	0	1	1	0
’548	1	0	2	1
’549	1	0	2	1

❑ 電源

■ IDLE1、IDLE2、IDLE3命令によるパワー ダウン モードで消費電力を制御

■ CLKOUT信号をディセーブルにする機能

❑ エミュレーション:オンチップのスキャン ベース エミュレーション ロジックにインタフェースするIEEE標準1149.1バウンダリ、スキャン ロジック

❑ 速度:1サイクル、固定小数点命令の25/20/15/12.5/10−ns† (40MIPS/50MIPS/66MIPS/80MIPS/100MIPS)†実行時間

デバイス	電源	速度	パッケージ
’541	5 V	25 ns	100-pin TQFP
	3.3 V	25 ns/20 ns/15 ns	100-pin TQFP
’542	5 V	25 ns	144-pin TQFP
	3.3 V	25 ns/20 ns	128-pin/144-pin TQFP
’543	3.3 V	25 ns/20 ns	100-pin TQFP
’545	3.3 V	25 ns/20 ns/15 ns	128-pin TQFP
’546	3.3 V	25 ns/20 ns/15 ns	100-pin TQFP
’548	3.3 V	15 ns/12.5 ns	144-pin TQFP/BGA
’549	3.3 V 2.5 V	15 ns/12.5 ns 10 ns	144-pin TQFP/BGA 144-pin TQFP/BGA

第2章

アーキテクチャの概要

2

本章では、’54xのアーキテクチャ構造の概要について説明します。このアーキテクチャには、中央処理演算ユニット(CPU)、メモリ、オンチップ ペリフェラルが含まれます。

’54x DSPは先進の改良型ハーバード アーキテクチャを使用しており、8つのバスによって処理能力を最大限に発揮します。

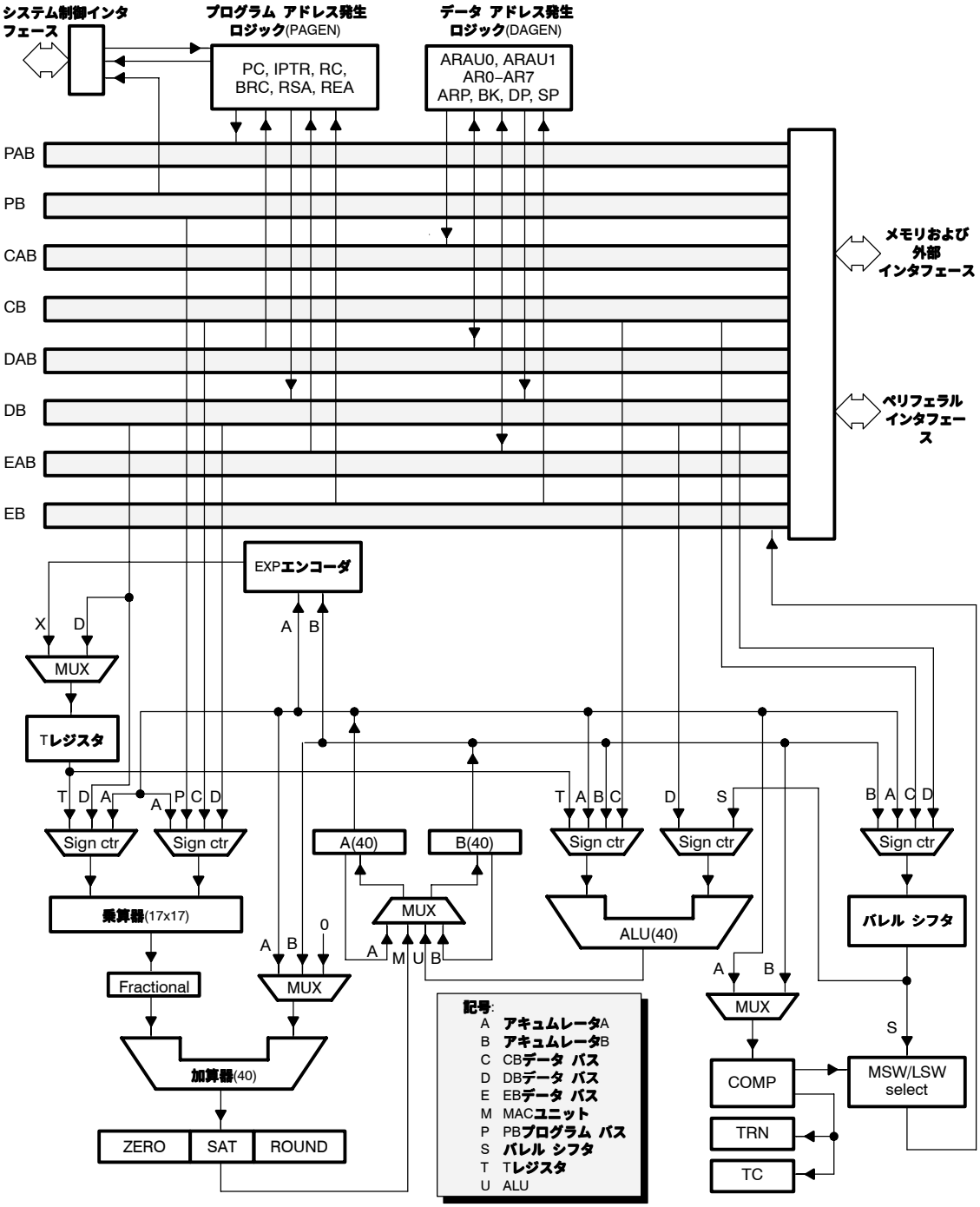
独立したプログラムおよびデータ空間により、プログラム命令およびデータに同時にアクセスでき、高度な並列処理を実現します。たとえば、3個の読み取りおよび１個の書き込みが1サイクルで実行できます。並列ストアをともなう命令および特定用途向け命令は、このアーキテクチャを最大限に利用しています。さらに、データをデータ空間とプログラム空間の間で転送することもできます。このような並列機能は、算術、論理、ビット操作実行セットを強力にサポートして、全て1マシン サイクルで実行できます。また’54xは、割り込み、繰り返し実行、ファンクション コールを管理する制御機能を備えています。

図2−1は、’54x機能のブロック図です。ここには、ペリフェラル ブロックおよびバス構造が示されます。

Topic **Page**

2

図2−1. TMS320C54x内部ハードウェアのブロック図



2.1 バス構造

'54xのアーキテクチャは、8本の主要16ビット バス(プログラム/データ バス×4本、アドレス バス×4本)から構成されています。

- プログラム バス(PB)は、命令コードおよびイミディエイト オペランドをプログラム メモリから転送します。
- 3本のデータ バス(CB、DB、EB)は、CPU、データ アドレス発生ロジック、プログラム アドレス発生ロジック、オンチップ ペリフェラル、データ メモリなど、様々な要素と相互接続します。
  - CB、DBは、データ メモリから読み取るオペランドを転送します。
  - EBは、メモリに書き込まれるデータを転送します。
- 4本のアドレス バス(PAB、CAB、DAB、EAB)は、命令実行に必要なアドレスを転送します。

'54xは、2つの補助レジスタ算術ユニット(ARAU0およびARAU1)を使って、最大2つのデータ メモリ アドレスを1サイクルごとに発生できます。

PBは、プログラム空間に格納されるデータ オペランドを、積和演算のために乗算器および加算器に転送するか、またはデータ移動命令(MVPDおよびREADA)のためにデータ空間に転送することができます。この機能は、デュアル オペランド読み取り機能とともに、FIRS命令など1サイクルの3オペランド命令をサポートします。

'54xには、オンチップ ペリフェラルにアクセスするためのオンチップ双方向バスを持ち、このバスはCPUインタフェースのバス ブリッジによってDBおよびEBに接続しています。このバスを利用するアクセスは、ペリフェラルによりリード/ライトのための複数のサイクルを可能とします。

表2−1は、様々なアクセス タイプで利用するバスをまとめています。

表2−1. 読み書きアクセスのためのバスの利用

	アドレス バス				データ バス			
アクセス タイプ	PAB	CAB	DAB	EAB	PB	CB	DB	EB
プログラム読み取り	√				√			
プログラム書き込み	√							√
データ単一読み取り			√				√	
データ デュアル読み取り		√	√			√	√	
データ長(32ビット)読み取り		√(hw)	√(lw)			√(hw)	√(lw)	
データ単一書き込み				√				√
データ読み書き			√	√			√	√
デュアル読み取り/係数読み取り	√	√	√		√	√	√	
ペリフェラル読み取り			√				√	
ペリフェラル書き込み				√				√

記号: hw = 上位16ビット ワード
lw = 下位16ビット ワード

2.2 内部メモリ構成

’54xメモリの空間は、プログラム、データ、I/Oの３つから構成されます。すべての’54xデバイスは、ランダム アクセス メモリ(RAM)および読み取り専用メモリ(ROM)の双方を備えています。デバイスには、デュアル アクセスRAM(DARAM)およびシングル アクセスRAM(SARAM)の2種類のRAMがあります。表2−2は、’54xの各デバイスごとのROM、DARAM、SARAMの容量を示しています。’54xには、26本のCPUレジスタおよびペリフェラル レジスタがあり、データ メモリ空間にマップされています。’54xのメモリ タイプおよび機能については、以降のサブセクションで説明します。様々なメモリ ブロックを使った構成についての詳細は、第3章メモリを参照してください。

表2−2. TMS320C5xデバイスのプログラムおよびデータ メモリ

メモリ タイプ	’541	’542	’543	’545	’546	’548	’549
ROM:	28K	2K	2K	48K	48K	2K	16K
プログラム	20K	2K	2K	32K	32K	2K	0
プログラム/データ	8K	0	0	16K	16K	0	16K
DARAM†	5K	10K	10K	6K	6K	8K	8K
SARAM†	0	0	0	0	0	24K	24K

† デュアル アクセスRAM(DARAM)およびシングル アクセスRAM(SARAM)をデータ メモリまたはプログラム/データ メモリとして構成できます。

2.2.1 オンチップROM

オンチップROMはプログラム メモリ空間の一部にあり、データメモリ空間の一部とすることもできます。オンチップROMの容量は、各デバイスによって異なります(表2−2参照)。

小容量のROM(2Kワード)を持つデバイスでは、ROMにブート ローダが内蔵され、より高速なオンチップまたは外部RAMでブートするのに役立ちます。ブート ローディングについての詳細は、TMS320C54x DSP Reference Set、Volume4: Application Guidesを参照してください。

大容量のROM(2Kワードを超える)を持つデバイスでは、ROMの一部はデータおよびプログラム空間の双方にマップされている場合があります。高容量のROMはカスタム用マスクROMです。コードまたはデータをROMにプログラムすることができます。

2.2.2 オンチップ デュアル アクセスRAM(DARAM)

DARAMは、いくつかのブロックから構成されています。各DARAMブロックには1マシン サイクルごとに2回アクセスできるので、中央処理演算ユニット(CPU)はDARAMの1ブロックに対して同時に読み書きできます。DARAMは常にデータ空間にマップされ、データ値の保存を優先的に行います。DARAMはプログラム空間にもマップすることができ、プログラム コードの格納にも使用できます。

2.2.3 オンチップ シングル アクセスRAM(SARAM)

SARAMは、いくつかのブロックから構成されます。各ブロックは、1マシン サイクルごとに読み取りまたは書き込みのために1回のアクセスが可能です。SARAMは常にデータ空間にマップされ主データ値の保持をします。SARAMはプログラム空間にもマップすることができ、プログラム コードのストアに使用できます。

2.2.4 オンチップ メモリのセキュリティ

'54xメモリのマスカブル セキュリティ オプションは、オンチップ メモリの内容を保護します。このオプションを指定すると、外部メモリで発行された命令はオンチップのメモリ空間にアクセスできません。

2.2.5 メモリ マップ レジスタ

データ メモリ空間には、CPUのメモリ マップ レジスタおよびオンチップ ペリフェラルがあります。これらのレジスタはデータ ページ0にあり、アクセスが容易です。メモリ マップ アクセスにより、コンテキスト スイッチのためのレジスタの待避、復元や、アキュムレータおよび他のレジスタ間の内容転送を容易に実現できます。

2.3 中央演算ユニット(CPU)

'54xのCPUは、'54xデバイスすべてに共通です。'54xのCPUは、次の項目を含んでいます。

- 40ビット算術論理演算器(ALU)
- 40ビット アキュムレータ×2
- バレル シフタ
- 17×17ビット乗算器
- 40ビット加算器
- 比較選択ストア ユニット(CSSU)
- データ アドレス発生器
- プログラム アドレス発生器

2.3.1 算術論理演算器(ALU)

'54xは、2の補数算術演算を1つの40ビット算術論理演算器(ALU)および2つの40ビット アキュムレータ(アキュムレータA、B)によって実行します。ALUは、ブール演算も実行できます。ALUには次の入力があります。

- 16ビット イミディエイト値
- データ メモリからの16ビット ワード
- Tレジスタの16ビット値
- データ メモリからの2つの16ビット ワード
- データ メモリからの32ビット ワード
- アキュムレータからの40ビット ワード

ALUは、2つの16ビットALUとして機能して、16ビット演算を同時に実行します。ALU機能の詳細については、4−11ページ、セクション4.2算術論理演算器(ALU)を参照してください。

2.3.2 アキュムレータ

アキュムレータAおよびB(2−2ページ図2−1参照)は、ALUまたは乗算器/加算器ブロックからの出力を格納します。これらは、ALUへの第2の入力になります。アキュムレータAは乗算器/加算器への入力になります。各アキュムレータは、3部分から構成されています。

- ガード ビット(ビット39−32)
- 上位ワード(ビット31−16)
- 下位ワード(ビット15−0)

ガード ビットにストアするための命令、上位および下位アキュムレータ ワードをデータ メモリにストアするための命令、データ メモリへまたはデータ メモリから32ビット アキュムレータ ワードを転送するための命令があります。さらに、いずれのアキュムレータも片方のアキュムレータの一時記憶のために使用できます。アキュムレータ機能の詳細については、4−15ページ、セクション4.3アキュムレータAおよびBを参照してください。

2.3.3 バレル シフタ

’54xのバレル シフタには、アキュムレータまたはデータ メモリ(CBまたはDBを利用)からの40ビット入力、およびALUまたはデータ メモリ(EBを利用)に接続される40ビット出力があります。バレル シフタは、0から31ビットの左シフト、および0から16ビットの右シフトを入力データについて実行できます。シフト要求は、命令のシフト カウント フィールド、ステータス レジスタST1のシフト カウント フィールド(ASM)またはTレジスタ(シフト カウント レジスタとして指定時)で指定されます。

バレル シフタおよびべき指数エンコーダは、アキュムレータの値を1サイクルで正規化します。出力の最下位ビット(LSB)は0づめされ、最上位ビット(MSB)はST1の符号拡張モード ビット(SXM)のステータスに応じて0づめか符号拡張でされます。さらにシフト機能は、数値スケーリング、ビット抽出、拡張算術演算、オーバーフロー防止機能を実行できます。シフタ機能と利用についての詳細は、4−19ページ、セクション4.4バレル シフタを参照してください。エンコーダのアキュムレータ正規化機能の詳細については、4−29ページ、セクション4.7べき指数エンコーダを参照してください。

2.3.4 乗算器/加算器ユニット

乗算器/加算器ユニットは、17×17ビットの2の補数乗算と40ビット加算を1命令サイクルで実行します。積和演算ユニットブロックは、乗算器、加算器、符号あり/符号なし入力制御ロジック、小数制御ロジック、ゼロ検出器、2の補数丸め、オーバーフロー/飽和ロジック、16ビット レジスタ(Tレジスタ)といった、複数の要素から構成されます。乗算器には2つの入力があり、1つはTレジスタ、データ メモリ オペランド、またはアキュムレータAから選択され、もう1つはプログラム メモリ、データ メモリ、アキュムレータA、またはイミディエイト値から選択されます。

高速なオンチップ乗算器により、’54xは重畳、相関、フィルタリングなどの演算を効率的に実行します。さらに、乗算器とALUにより積和(MAC)演算およびALU演算を、並列に1命令サイクルで実行します。この機能は、ユークリッド距離の検出、および対称フィルタとLMSフィルタの実現に使用でき、これらは複雑なDSPアルゴリズムで必要となります。積和演算器ユニットの詳細については、4−21ページ、セクション4.5積和演算器ユニットを参照してください。

2.3.5 比較選択格納ユニット(CSSU)

比較選択格納ユニット(CSSU)は、アキュムレータの上位および下位ワード間の大小比較を行い、ステータス レジスタST0およびトランジションレジスタ(TRN)双方のテスト/制御フラグ ビット(TC)がそれぞれの遷移履歴を維持できるようにして、アキュムレータのより大きなワードを選択してデータ メモリに格納します。CSSUは、ビタビ タイプのバタフライ演算を、最適化したオンチップ ハードウェアによって高速実行します。CSSUの詳細については、ページ4−26、セクション4.6、比較選択ストアユニット(CSSU)を参照してください。

2.4 データ アドレッシング

’54xには、7種類の基本的なデータ アドレッシング モードがあります。

- ❑ イミディエイト アドレッシングは、命令中の値を固定値として使用します。
- ❑ 絶対アドレッシングは、命令中の固定アドレスを使用します。
- ❑ アキュムレータ アドレッシングは、アキュムレータAを使ってプログラム メモリのある位置にデータとしてアクセスします。
- ❑ 直接アドレッシングは、命令中の7ビットを使ってアドレスの下位7ビットとします。この7ビットはデータ ページ ポインタ(DP)またはスタック ポインタ(SP)とともに使用されて、実際のメモリ アドレスが割り出されます。
- ❑ 間接アドレッシングは、補助レジスタを使ってメモリにアクセスします。
- ❑ メモリ マップド レジスタ アドレッシングは、現在のDP値またはSP値を変更せずにメモリ マップド レジスタをアクセスします。
- ❑ スタック アドレッシングは、システム スタックへの追加および削除を管理します。

直接、間接、またはメモリ マップド レジスタのアドレッシングを使う命令の実行中、データ アドレス発生ロジック(DAGEN)は、データ メモリ オペランドのアドレスを処理します。データ アドレッシング モードの詳しい説明は、第5章データ アドレッシングを参照してください。

2.5 プログラム メモリ アドレッシング

’54xデバイスではプログラム メモリは通常、プログラム カウンタ(PC)を使ってアドレッシングされます。しかし一部の命令では、プログラム メモリ中に記憶されたデータに絶対アドレッシングを使ってアクセスする場合があります。(絶対アドレッシングについては第5章 データ アドレッシングを参照してください。)

個々の命令をフェッチするのに使用するPCは、プログラム アドレス発生ロジック(PAGEN)によってロードされます。一般的に、PAGENは連続命令がフェッチされるにしたがってPCをインクリメントします。一方でPAGENは、一部の命令または他の演算の結果として不連続値を、PCにロードすることがあります。不連続を生ずる演算には、分岐、サブルーチン コール、リターン、条件付き演算、単一命令リピート、複数命令リピート、リセット、割り込みなどがあります。サブルーチン コールおよび割り込みでは、現在のPCがスタック ポインタ(SP)で示されるスタックに待避されます。呼び出されたファンクションまたは割り込みサービス ルーチンが終了すると、待避されたPC値がリターン命令によってスタックから復帰します。

プログラム アドレス発生におけるハードウェアおよびソフトウェアの役割につていは、第6章プログラム メモリ アドレッシングを参照してください。

2.6 パイプライン処理

命令パイプラインは、命令実行時に発生する各連続的処理から構成されます。’54xパイプラインには、プリフェッチ、フェッチ、デコード、アクセス、リード、実行の6つのステージがあります。各ステージで、独立した処理が発生します。これらの処理は独立しているので、1から6つの命令をどのサイクルでもアクティブにでき、各命令は異なるステージに振り分けられます。一般的に、パイプラインは連続する命令セットで満たされ、それぞれが6ステージのひとつに対応します。分岐、サブルーチン コール、復帰などの際にPCの不連続が発生すると、ひとつ以上のパイプライン ステージが一時的に使用されなくなります。パイプライン処理の詳細については、第7章パイプラインを参照してください。

2.7 オンチップ ペリフェラル

’54xシリーズのデバイスは全て同じCPUを持ちますが、異なるオンチップ ペリフェラルがそれぞれのCPUに接続されています。’54xデバイスには、次のようなオンチップ ペリフェラルのオプションがあります。

- 汎用I/Oピン($\overline{\text{BIO}}$およびXF)
- プログラマブル ウェイト ステート発生器
- プログラマブル バンク スイッチング ロジック
- ホスト ポート インタフェース(HPI)
- ハードウェア タイマ
- クロック発生器
- シリアル ポート
  - 同期シリアル ポート
  - バッファド シリアル ポート
  - 時分割多重(TDM)シリアル ポート

これらペリフェラルの詳細については、第8章オンチップ ペリフェラル、第9章シリアル ポート、第10章外部バス機能を参照してください。

2.7.1 汎用I/Oピン

各’54xデバイスには、$\overline{\text{BIO}}$およびXFの2種類の汎用I/Oピンがあります。$\overline{\text{BIO}}$は入力ピンで、外部デバイスの状態モニタに使用できます。XFはソフトウェア制御による出力で、外部デバイスに信号を出力できます。$\overline{\text{BIO}}$およびXFの詳細については、8−10ページ、セクション8.2 汎用I/Oピンを参照してください。

2.7.2 ソフトウェア プログラマブル ウェイト ステート発生器

ソフトウェア プログラマブル ウェイト ステート発生器は、外部バス サイクルを最大7マシン サイクルまで拡張して、低速なオフチップ メモリおよびI/Oデバイスとのインターフェイスを行います。ソフトウェア ウェイト ステート発生器は、外付け部ハードウェアなしにウェイト ステートを挿入することができます。外部メモリ アクセスでは0から7のウェイト ステートをソフトウェア ウェイト ステート レジスタ(SWWSR)で、プログラムおよびデータ メモリの各32Kワード ブロック、およびI/O空間の64Kワード ブロックについて指定できます。詳細については、10−5ページ、サブセクション10.3.1ウェイト ステート発生器を参照してください。

2.7.3 プログラマブル バンク スイッチング ロジック

プログラマブル バンク スイッチング ロジックは、プログラム メモリまたはデータ メモリ内へのアクセスがメモリ バンク境界をまたぐとき、自動的に1サイクルを挿入します。アクセスがプログラム メモリからデータ メモリにまたぐときも、1サイクルを挿入できます。この余分なサイクルは、他のデバイスがバスを駆動開始する前にメモリ デバイスがバスを解放できるようにして、バスの競合を防ぎます。バンク スイッチングのメモリ サイズは、バンク スイッチング制御レジスタ(BSCR)によって指定します。詳しくは、10−7ページの、サブセクション10.3.2**バンク スイッチング ロジック**を参照してください。

2.7.4 ホスト ポート インターフェイス

ホスト ポート インターフェイス(HPI)は8ビットのパラレル ポートで、ホスト プロセッサへのインターフェイスを実現します。情報は、'54xおよびホスト プロセッサ間で'54xオンチップ メモリを経て交換され、このメモリはホスト プロセッサおよび'54x双方からアクセスできます。表2−3は、HPIを備えた'54xデバイスを示しています。HPI機能の詳細については、8−26ページ、セクション8.5**ホスト ポート インターフェイス**を参照してください。

表2−3. TMS320C54x**デバイスのホスト ポート インターフェイス**

オンチップ ペリフェラル	'541	'542	'543	'545	'546	'548	'549
ホスト ポート インターフェイス	0	1	0	1	0	1	1

2.7.5 ハードウェア タイマ

'54xは、4ビット プリスケーラをともなう16ビットタイマ カウンタ回路を備えています。タイマ カウンタは、CLKOUTサイクルごとに1デクリメントします。カウンタがデクリメントされ0になるたびに、タイマ割り込みが発生します。タイマは、ステータス ビットの指定により停止、再始動、リセット、無効にできます。詳細については、8−12ページ、セクション8.3**タイマ**を参照してください。

2.7.6 クロック発生器

クロック発生器は、内部オシレータおよびフェーズ ロックド ループ(PLL)回路から構成されます。クロック発生器は、内部的には水晶共振子と内蔵発振器の組み合わせにより、または外部のクロック ソース(外部発振器)によって駆動できます。PLL回路は、クロック ソースに特定の係数を掛けて内部CPUクロックを発生できます。クロック発生器の詳細については、ページ8−16、セクション8.4**クロック発生器**を参照してください。

2.8 シリアル ポート

’54xのシリアル ポートはデバイスによって異なりますが、同期、バッファ、時分割多重(TDM)の3タイプがあります。表2−4は、’54xデバイスごとの各タイプのシリアル ポート数を示しています。本セクションの以下のサブセクションでは、3タイプのシリアル ポートについて概要を説明します。これらのポートの詳細については、第9章シリアル ポートを参照してください。

表2−4. TMS320C54xデバイスのシリアル ポート インタフェース

シリアル ポート	’541	’542	’543	’545	’546	’548	’549
同期	2	0	0	1	1	0	0
バッファ	0	1	1	1	1	2	2
TDM	0	1	1	0	0	1	1

2.8.1 同期シリアルI/Oポート

同期シリアル ポートは高速、全二重シリアル ポートで、CODEC、アナログ/デジタル(A/D)コンバータなどのシリアル デバイスや他のシリアル システムとの直接通信を実現します。’54xに複数の同期シリアル ポートがある場合は、それらのポートは同じ機能ですが独立しています。各同期シリアル ポートは、マシン サイクル速度(CLKOUT)の最大4分の1で作動します。同期シリアル ポート送信部および受信部はダブルバッファで構成され、マスカブル外部割り込み信号によって個別に制御されます。データは、バイトまたはワードとしてフレーム化されます。

2.8.2 バッファド シリアル ポート

バッファド シリアル ポート(BSP)は、自動バッファリング ユニットによって拡張された同期シリアル ポートで、CLKOUTの速度で動作します。全二重、ダブルバッファで、フレキシブルなデータ ストリーム長を提供します。自動バッファリング ユニットは、高速転送をサポートして、割り込みのサービス ルーチンオーバヘッドを低減します。通常のシリアル ポートとしても使用できます。

2.8.3 TDMシリアル ポート

時分割多重(TDM)シリアル ポートは、データの時分割多重が可能なように拡張されたシリアル ポートです。TDM動作は通常のシリアル ポートとしても使用できます。TDMシリアル ポートは、一般にマルチプロセッサ アプリケーションで使用されます。

2.9 外部バス インターフェイス

’54xは、最大64Kワードのデータ メモリ、64Kワードのプログラム メモリ(’548と’549では8Mワード)、および最大64Kワードの16ビット パラレルI/Oポートにアクセスできます。外部メモリまたはI/Oポートのいずれかへのアクセスは、外部インタフェースを経て行われます。個別のセレクト信号である、$\overline{DS}$、$\overline{PS}$、$\overline{IS}$は、物理的な空間選択を可能にします。

外部レディ入力信号およびソフトウェアによって発生するウェイト ステートによって、プロセッサは速度の異なるメモリおよびI/Oデバイスにインターフェイスできます。ホールド モードによって、外部デバイスは’54xバスを制御することも可能です。このようにして、外部デバイスはプログラム、データ、I/O空間のリソースにアクセスできます。

ほとんどの’54x命令は外部メモリに、アクセスできます。しかし、I/Oポートへのアクセスには特別な命令であるPORTR、PORTWに限定されています。

’54xの外部デバイスへのインターフェイスに関する詳細は、第10章外部バス機能を参照してください。

2.10 IEEE1149.1スキャニング ロジック

IEEE1149.1スキャニング ロジック回路は、エミュレーションおよびテスト目的にのみ使用します。このロジックは、インターフェイスしているデバイスへ、またはデバイスからのバウンダリ スキャンを実現します。また、このロジックはpin−to−pinの連続性のテスト、および’54xの周辺デバイスの動作テストのために使用できます。
IEEE1149.1スキャニング ロジックは、あらゆるオンチップ リソースへのアクセスが可能な内部のスキャニング ロジック回路にインタフェースされます。
これにより、’54xはIEEE1149.1シリアル スキャン ピンおよびエミュレーション専用ピンを使ってオンボード エミュレーションを実行できます。詳しくは、付録Bの“XDS510エミュレータ使用のための設計について”を参照してください。

第3章

メモリ

3

本章では、’54xのメモリ構成および動作について説明します。通常、’54xデバイスには192K×16ビット ワードの総メモリ空間があります。この空間は、64Kワードのプログラム、64Kワードのデータ、64KワードののI/Oという3つのメモリ セグメントに分割されます。’548および’549など一部のデバイスでは、ページ切り替えなどの機能によりメモリ構造が異なります。’54xアーキテクチャの並列特性およびオンチップRAMのデュアル アクセス機能により、’54xは1命令フェッチ、2オペランド読み取り、1オペランド書き込みの4つの同時メモリ動作をどのマシン サイクルでも実行できます。

オンチップ メモリからの動作には、次の長所があります。

- ❑ ウェイト ステートが不要なので高性能化が可能
- ❑ 外部メモリに比べて低コスト
- ❑ 外部メモリに比べて省電力

外部メモリからの動作による主な長所は、大規模なメモリ空間へのアクセスが可能なことです。

Topic Page

3.1 メモリ空間

’54xのメモリは、プログラム、データ、I/Oという3つの独立した空間から構成されます。これらのどの空間でも、RAM、ROM、EPROM、EEPROM、またはメモリ マップ ペリフェラルをオンチップまたはオフチップのいずれかに置くことができます。これらの3つの空間は全体で、192Kワードのアドレス幅になります(’548と’549を除く)。

プログラム メモリ空間には、実行する命令、実行に使われるテーブルが含まれます。データ メモリ空間は、命令が使用するデータを格納します。I/Oメモリ空間は外部のメモリ マップド ペリフェラルにインタフェースしたり、追加データ記憶空間としても使用できます。

チップのバージョンに応じて、’54xにはデュアル アクセスRAM(DARAM)、シングル アクセスRAM(SARAM)、ROMという複数タイプのオンチップ メモリがあります。RAMは常にデータ空間内にマップされますが、プログラム空間にマップできる場合もあります。ROMはプログラム空間にマップされ、アクティブにできます。一部、データ空間にマップできるものもあります。

’54xの次の3種類の機能により、プログラムまたはデータ空間でオンチップ メモリを柔軟にイネーブルまたはディセーブルできるようになります。

- MP/$\overline{\text{MC}}$ビット。このビットを0に設定した場合は、オンチップROMはプログラム空間にマップされます。このビットを1に設定した場合は、オンチップROMはプログラム空間にマップされず、ディセーブルされます。
- OVLYビット。このビットを1に設定した場合は、オンチップRAMはプログラム空間およびデータ空間にマップされます。このビットが0に設定される場合は、オンチップRAMはデータ空間のみにマップされます。
- DROMビット。このビットを1に設定した場合は、オンチップROMの一部はデータ空間にマップされます。このビットを0に設定した場合は、オンチップROMはデータ空間にマップされず、ディセーブルとなります。DROMは、MP/$\overline{\text{MC}}$の状態には影響されません。

MP/$\overline{\text{MC}}$、OVLY、DROMビットは、プロセッサ モード ステータス レジスタ(PMST)に配置されています。詳しくは、4−2ページ、セクション4.1CPUステータスおよび制御レジスタを参照してください。

図3−1から図3−4は、’54xデバイスのデータおよびプログラム メモリ マップ、およびマップがMP/$\overline{\text{MC}}$、OVLY、DROMビットによってどのように影響を受けるかを示しています。

図3−1. TMS320C541**のメモリ マップ**

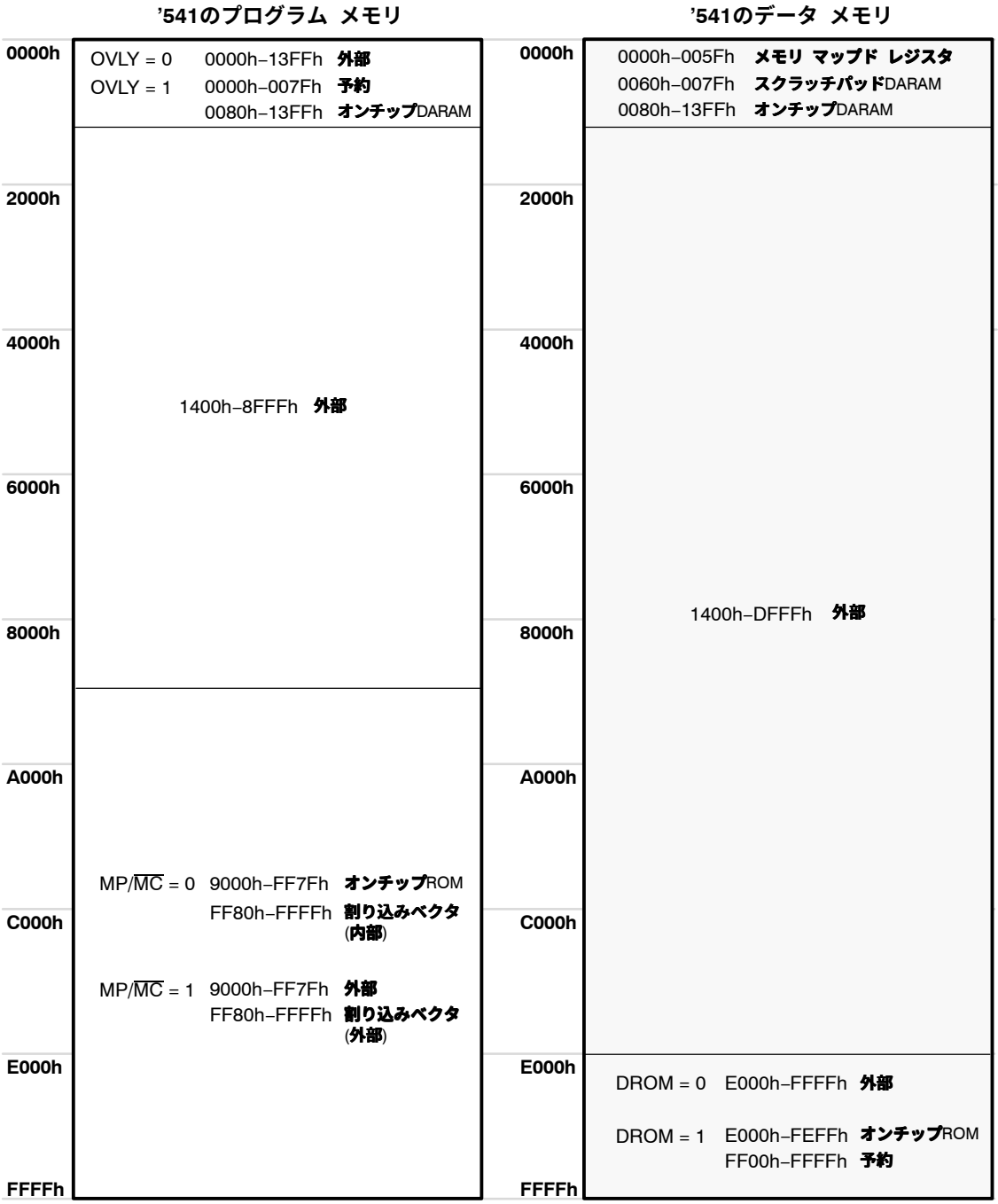


図3−2. TMS320C542**および**TMS320C543**のメモリ マップ**

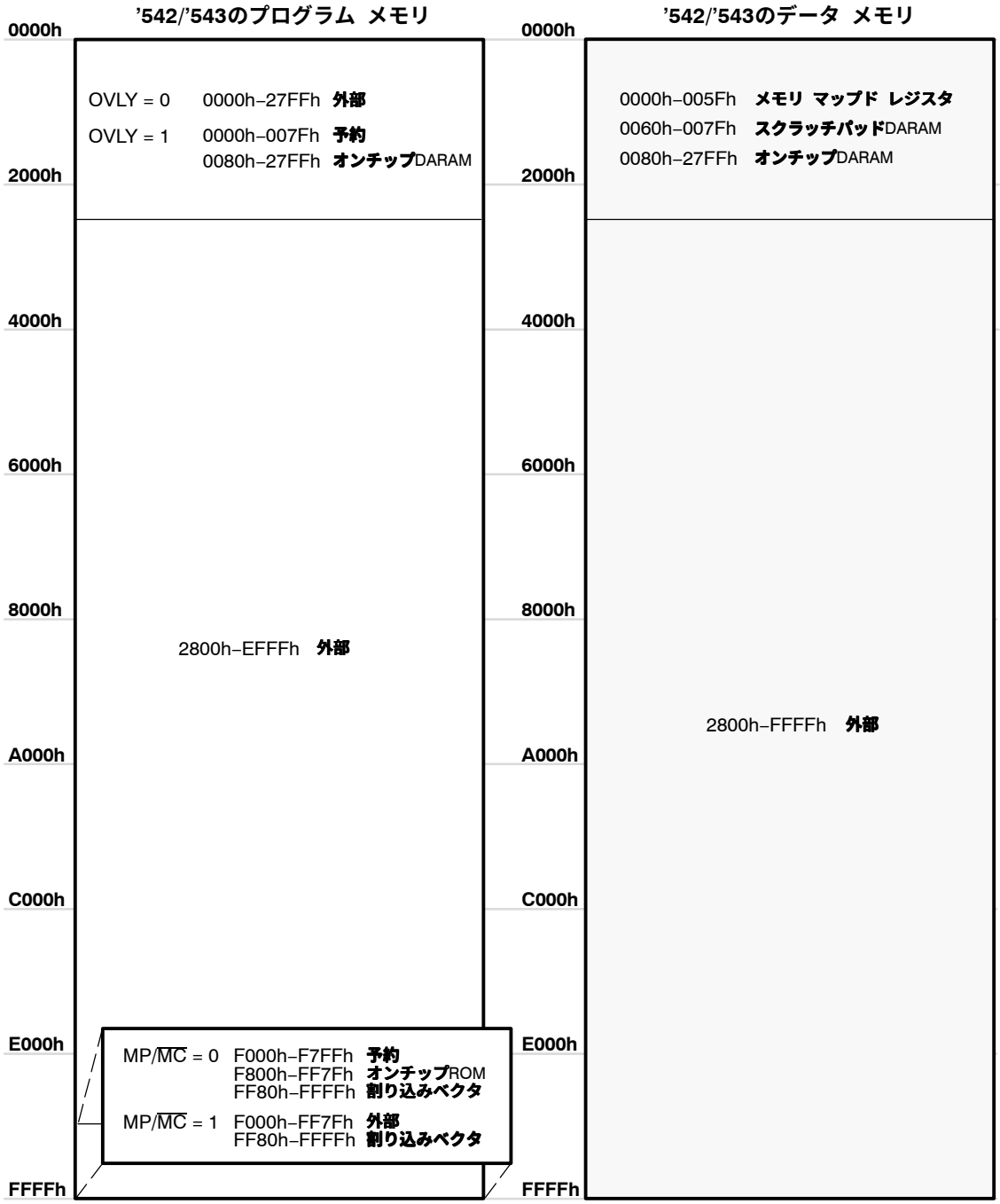


図3−3. TMS320C545**および**TMS320C546**のメモリ マップ**

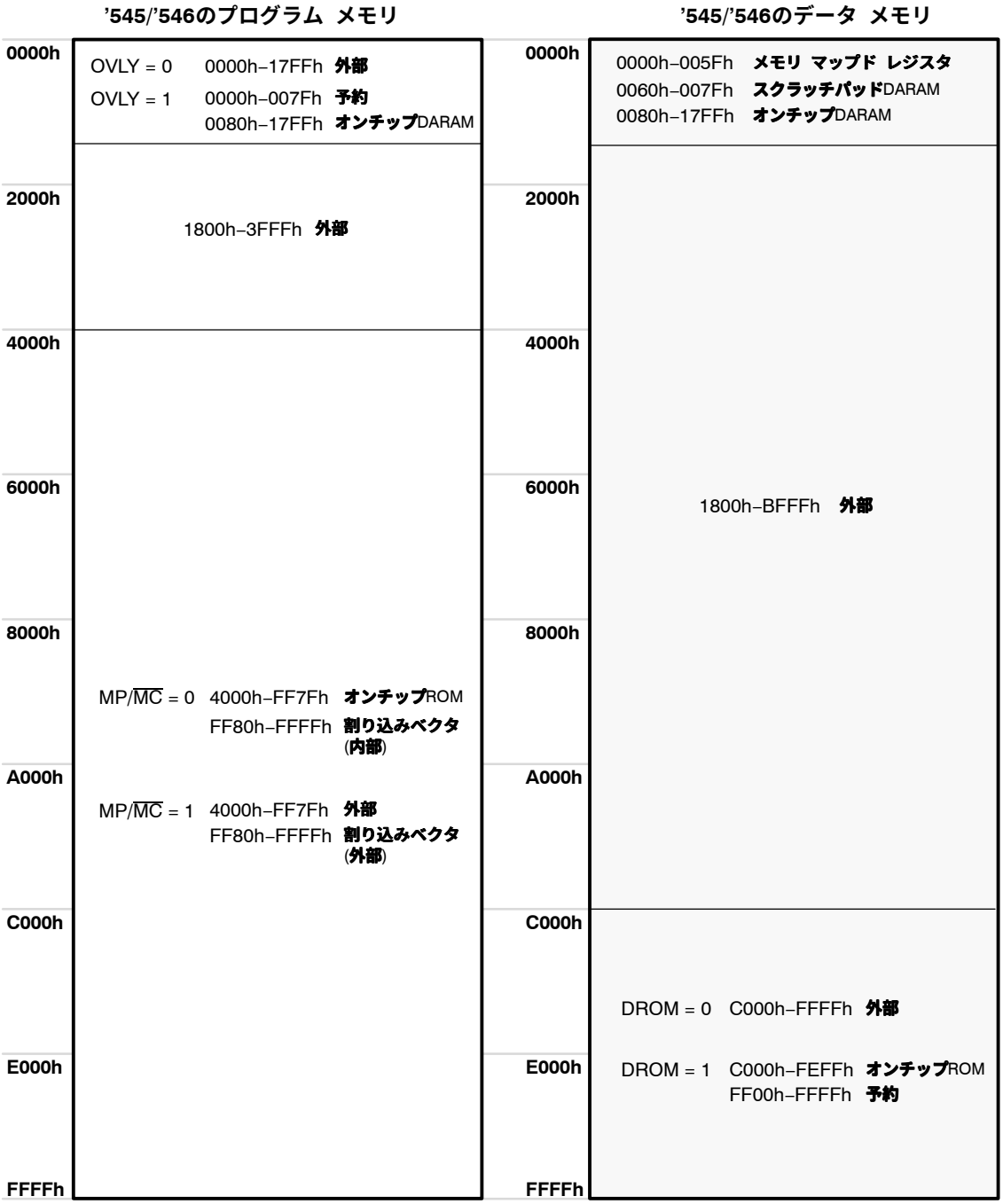


図3−4. TMS320C548**のメモリ マップ**

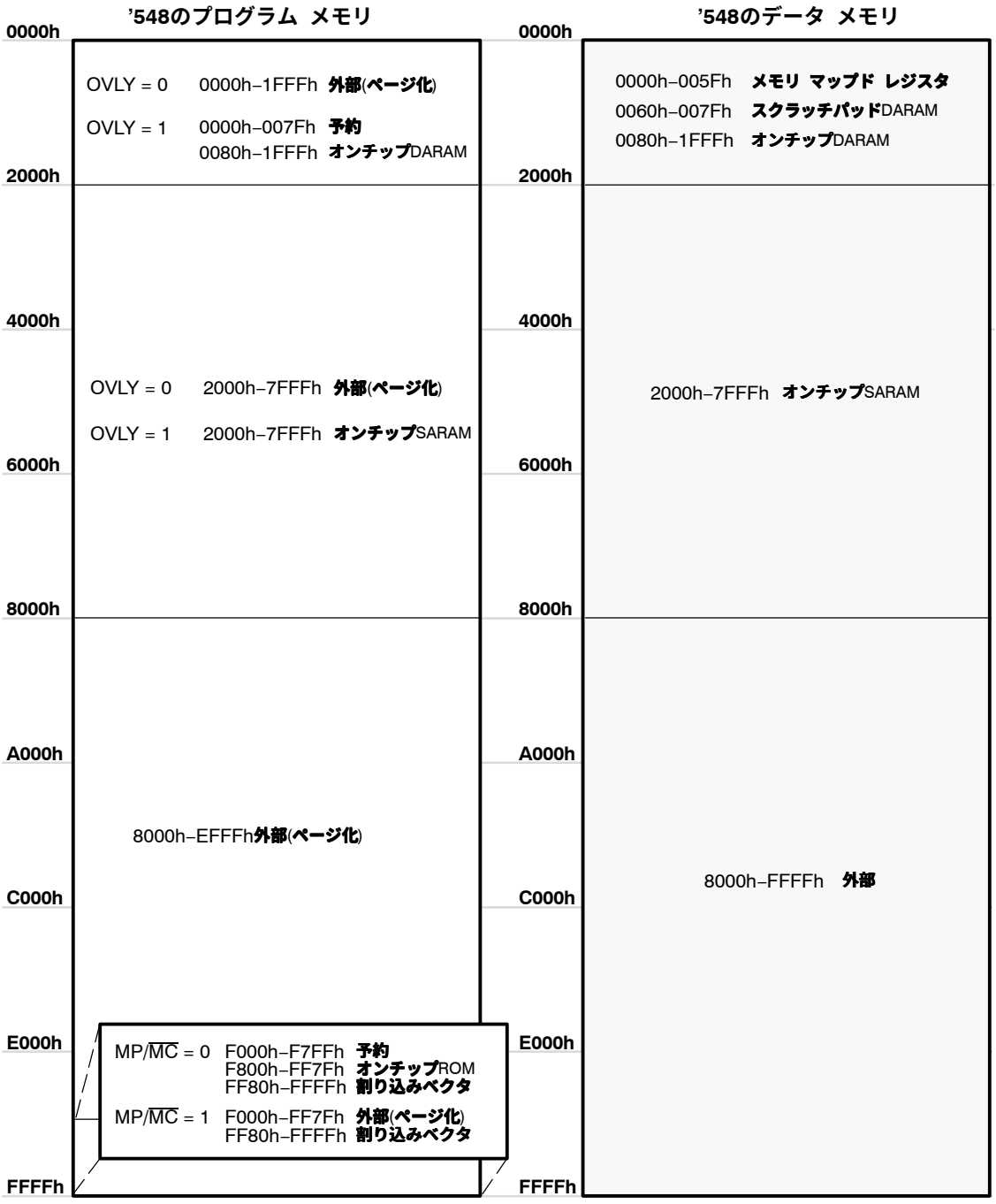
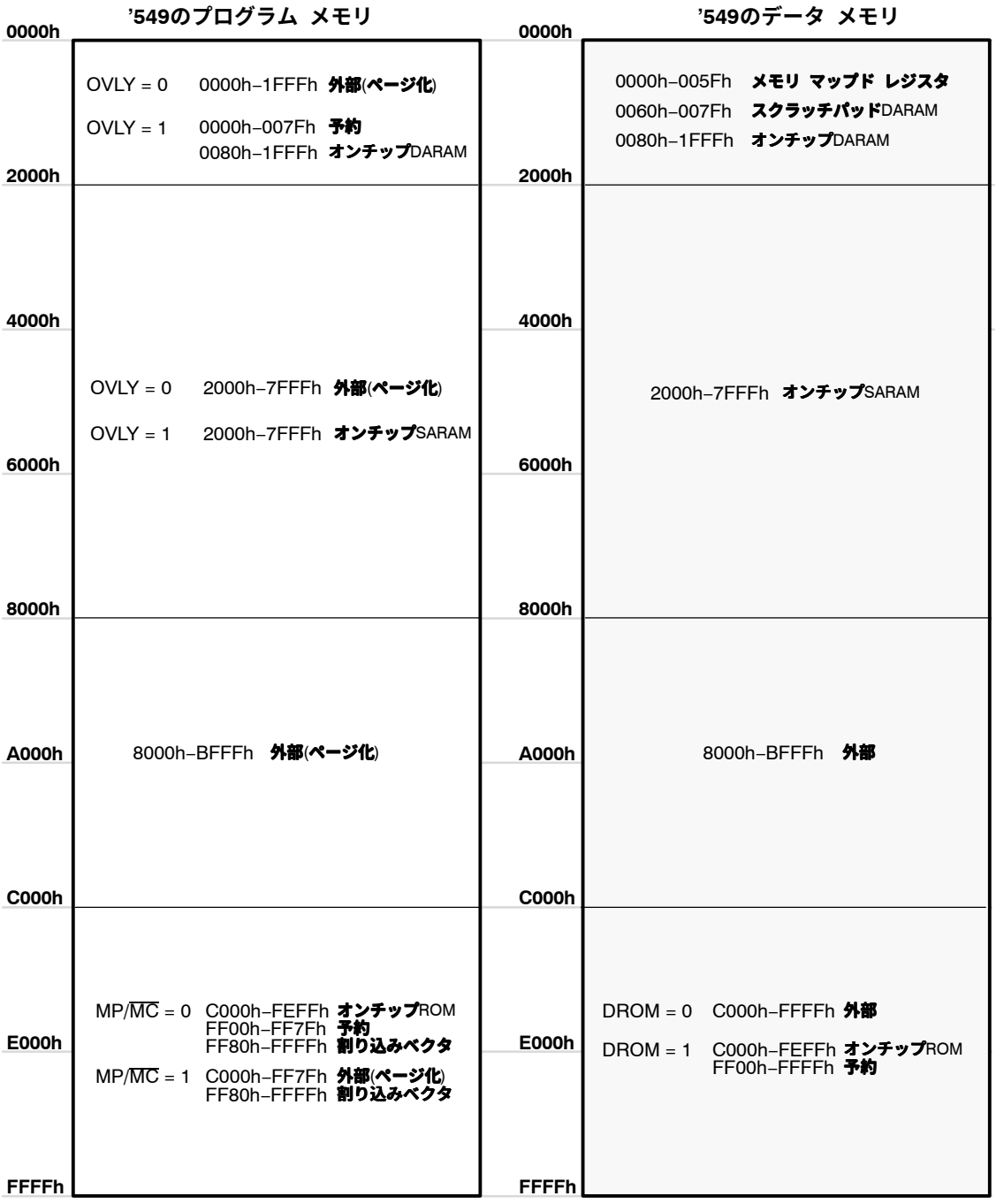


図3−5. TMS320C549**のメモリ マップ**



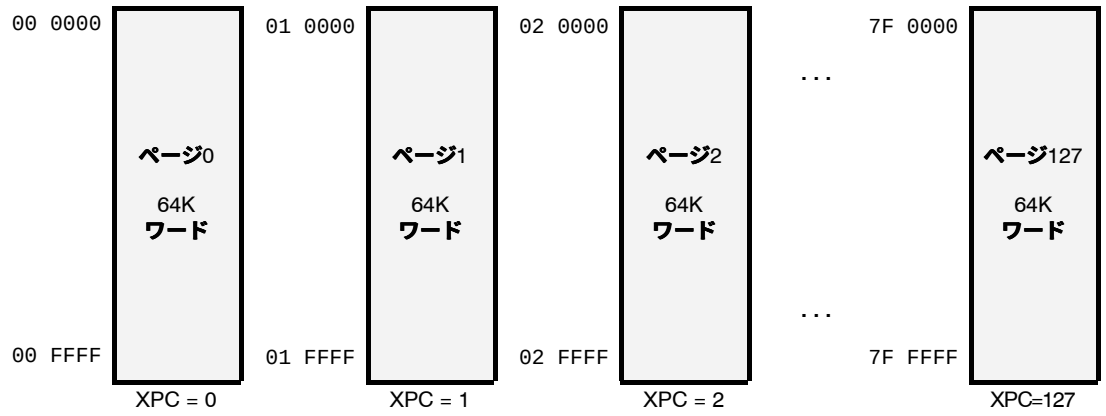
3.1.1 拡張プログラム メモリ(TMS320C548およびTMS320C549)

’548および’549は、プログラム空間にページ毎に分割された拡張メモリ機能を持っており、最大8192Kのプログラム メモリのアクセスが可能になります。この機能を実現するために、’548および’549には複数の追加機能があります。

- ❑ アドレス ラインの追加。16本→23本
- ❑ 追加されたメモリ マップ レジスタ、プログラム カウンタ拡張レジスタ(XPC)
- ❑ 拡張プログラム空間にアクセスするための6つの追加命令

’548のプログラム メモリは、128ページで構成されており、各ページは64Kワードです(図3−6参照)。

図3−6. オンチップRAMがプログラム空間にマップされない場合の拡張プログラム メモリ(OVLY=0)

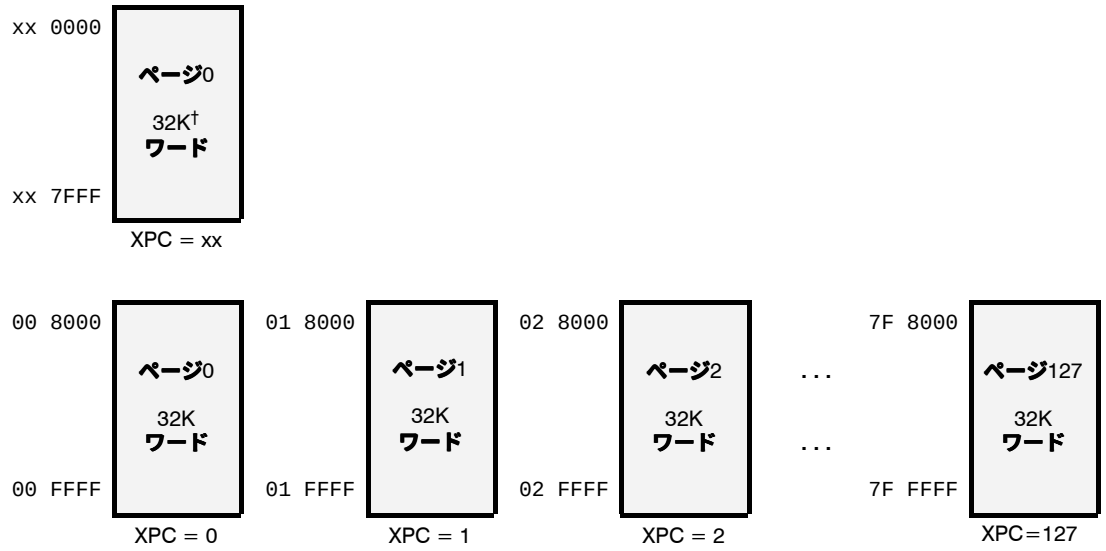


オンチップRAMがプログラム空間でイネーブルのとき、プログラム メモリの各ページは、最大32Kワードの共通ブロックおよび32Kワードの専用ブロックの、2つの部分から構成されます。共通ブロックはすべてのページが共有して、専用ブロックは割り当てられたページによってのみアクセス可能です。図3−7は、共通ブロックおよび専用ブロックを示します。

オンチップROMがイネーブル(MP/$\overline{MC}$=0)のときは、ページ0上でのみイネーブルになります。プログラム メモリの他のページにはマップされません。

XPCレジスタの値によって、ページを指定します。このレジスタは、アドレス001Ehのデータ空間にメモリ マップされます。ハードウェア リセット時には、XPCは0に初期化されます。

図3−7. オンチップRAMがプログラム空間とデータ空間にマップされた場合の拡張プログラム メモリ (OVLY=1)



注： オンチップRAMがプログラム空間でイネーブルのときは、xx0000–xx7FFF領域へのすべてのアクセスはページ番号に関係なくオンチップRAMの00 0000–00 7FFFにマップされます。

† このオンチップ メモリ領域の詳細については、図3−4を参照してください。

ソフトウェアによるページ スイッチングを容易にするために、’548と’549にはXPCに影響する6つの特別な命令があります。

- FB[D] − ファー ブランチ
- FBACC[D] − アキュムレータAまたはアキュムレータBの値によって指定される位置へのファー ブランチ
- FCALA[D] − アキュムレータAまたはアキュムレータBの値によって指定される位置へのファー コール
- FCALL[D] − ファー コール
- FRET[D] − ファー リターン
- FRETE[D] − 割り込みのイネーブルをともなうファー リターン

これらの新しい命令に加えて、’548と’549で拡張アドレスを使用するために、次の2つの’54x命令が拡張されています。

- READA − アキュムレータAがアドレスするプログラム メモリを読み取ってデータ メモリに記憶します。
- WRITA − アキュムレータAがアドレスするプログラム メモリにデータを書き込みます。

その他のすべての命令はXPCを変更せず現在のページの範囲でメモリのみにアクセスします。

3.2 プログラム メモリ

’54xデバイス上の外部プログラム メモリ(’548’549を除く)は、最大64K×16ビット ワードをアドレスします。’54xデバイスには、オンチップROM、デュアル アクセスRAM(DARAM)、シングル アクセスRAM(SARAM)があり、ソフトウェアによってプログラム空間にマップできます。プログラム アドレス発生器(PAGEN)がオンチップ メモリの境界外にアドレスを発生すると、デバイスは自動的に外部アクセスをします。(プログラム アドレス発生の詳細については、第6章プログラム メモリ アドレッシングを参照してください。)表3−1は、各種’54xデバイスのオンチップ プログラム メモリを示しています。

表3−1. TMS320C54xデバイスのオンチップ プログラム メモリ

デバイス	ROM (MP/$\overline{MC}$ = 0)	DARAM (OVLY = 1)	SARAM (OVLY = 1)
‘541	28K	5K	–
‘542	2K	10K	–
‘543	2K	10K	–
‘545	48K	6K	–
‘546	48K	6K	–
‘548	2K	8K	24K
‘549	16K	8K	24K

3.2.1 プログラム メモリ構成の可能性

MP/$\overline{MC}$およびOVLYビットは、どのオンチップ メモリがプログラム空間でイネーブルにされるかを決めます。

リセット時は、MP/$\overline{MC}$ピンのロジック レベルはPMSTレジスタのMP/$\overline{MC}$ビットに転送されます(ページ4−2、セクション4.1CPUステータスおよび制御レジスタ参照)。MP/$\overline{MC}$ビットは、オンチップROMをイネーブルにするかどうかを決めます。MP/$\overline{MC}$=1の場合は、デバイスはマイクロプロセッサ モードとして設定され、オンチップROMはディセーブルされます。MP/$\overline{MC}$=0の場合は、デバイスはマイクロコンピュータ モードとして設定され、オンチップROMはイネーブルになります。MP/$\overline{MC}$ピンは、リセット時にサンプルされますが、PMSTレジスタのMP/$\overline{MC}$ビットを設定またはクリアすることで、ソフトウェアによってオンチップROMをディセーブルまたはイネーブルにできます。

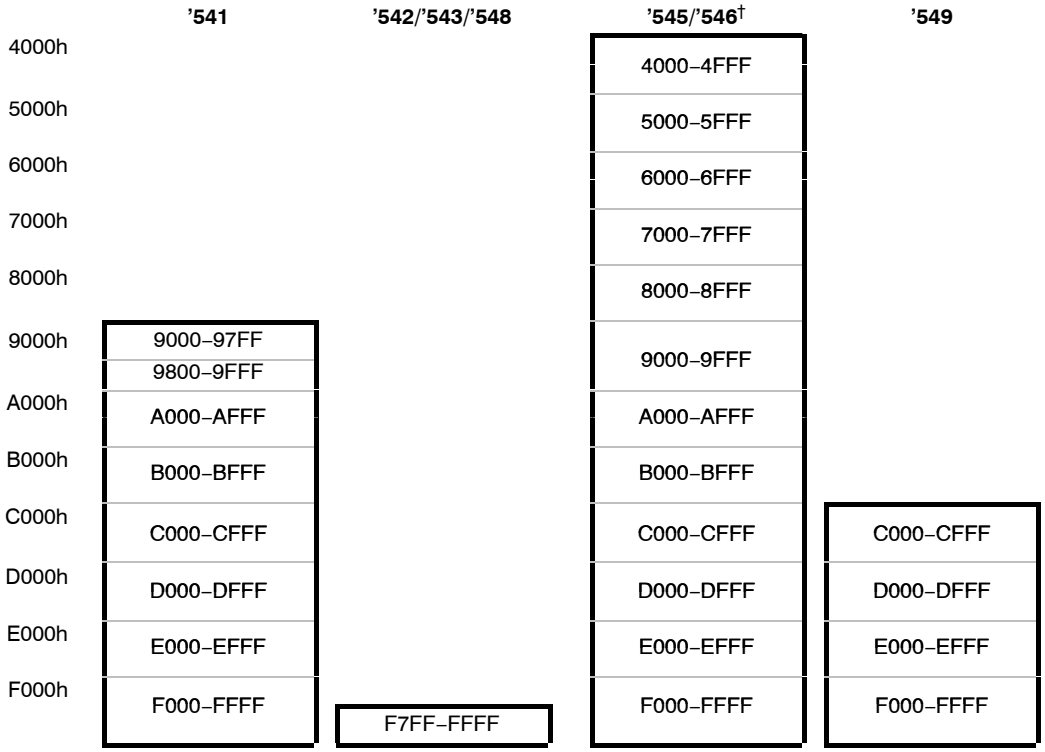
リセット時は、RAM(DARAMおよびSARAM)はプログラム空間でアドレッシング不可能です。PMSTレジスタのOVLYビットを設定することで、RAMをプログラム空間でアドレス可能にできます。OVLY=0の場合は、RAMはデータ空間のみにマップされます。OVLY=1の場合は、RAMはプログラムおよびデータ両空間にマップされます。

各’54xデバイスのプログラム メモリ構成については、図3−1から図3−4まで(3−3ページから3−6ページ)を参照してください。

3.2.2 オンチップROM構成

オンチップROMは、ブロック分割された構成によって性能を向上させています。たとえば、このブロック構成によって、ROMのある1ブロックから命令をフェッチでき、同時に発生する別のROMブロックからのデータ アクセスが可能になります。図3−8は、各’54xデバイスのROMのブロック構成を示しています。図の点線はブロック境界を示します。

図3−8. オンチップROMブロック構成



† 545/546Aバージョンのデバイスでは ROMは8Kブロックで構成されます。

3.2.3 プログラム メモリ アドレス マップおよびオンチップROM内容

デバイスのリセット時に、リセット、割り込み、トラップ ベクタがプログラム空間のアドレスFF80hにマップされます。ただし、デバイスのリセット終了後、これらのベクタはプログラム空間の任意の128ワードの先頭に再度マップできます。この機能により、ベクタ テーブルをブートROM外に移動してROMをメモリ マップから外すことが可能になります。ベクタの再マッピングの詳細については、6−35ページ、サブセクション6.10.9割り込みベクタ アドレスの再マッピングを参照してください。

注 :

オンチップROM中では、128ワードがデバイスのテストのために予約されています。オンチップROMに実装するために書かれたアプリケーション コードは、プログラム空間のアドレスFF00h−FF7Fhの128ワードは予約領域であるため、使用できません。

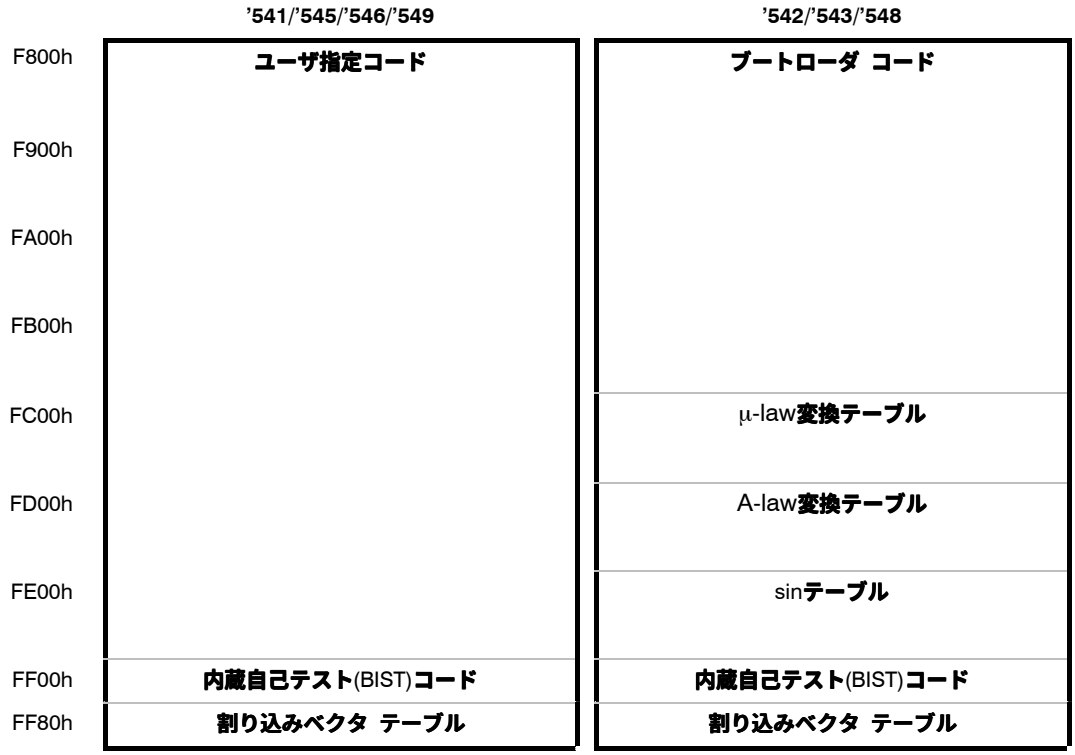
3.2.4 オンチップROMコード内容およびマッピング

'54xデバイスには、大容量(16K、24K、28K、48Kワード)オンチップROMまたは2Kワードのオンチップ ROMがあります。大容量オンチップROMは、ユーザーにより作成されたコードでプログラムでき、2Kワードのオンチップ ROMの内容はテキサス インスツルメンツが指定するものになります。2Kワードのオンチップ ROMの'54xデバイス(F800hからFFFFh)には次の内容が含まれます。

- ❑ シリアル ポート、外部メモリ、I/Oポート、またはホスト ポート インタフェース(もしあれば)からブートするブートローダ プログラム。
- ❑ 256ワード μ−law変換テーブル
- ❑ 256ワードA−law変換テーブル
- ❑ 256ワードsinテーブル
- ❑ 割り込みベクタ テーブル

図3−9は、これらの項目がどの'54xデバイスにあるか、および各項目のアドレスを示しています。コードのアドレス領域、F800h−FFFFhは、MP/$\overline{\text{MC}}$ビットが0の場合、オンチップROMにマップされます。

図3−9. **オンチップ**ROM**プログラム メモリ マップ(上位アドレス)**



3.3 データ メモリ

’54xのデータ メモリには、最大64K×16ビットワードが含まれます。’54xデバイスの多くは、ソフトウェアによってデータ空間にマップできるオンチップROMがあり、さらにデュアルおよびシングル アクセスRAM(DARAM、SARAM)があります。表3−2は、’54xデバイスのオンチップ データ メモリを示しています。

表3−2. TMS320C54xデバイスのオンチップ データ メモリ

デバイス	プログラム/データROM (DROM = 1)	DARAM	SARAM
’541	8K	5K	–
’542	–	10K	–
’543	–	10K	–
’545	16K	6K	–
’546	16K	6K	–
’548	–	8K	24K
’549	16K	8K	24K

RAMおよびデータROMへのアクセス(イネーブル時)は、アドレスが対応するオンチップ メモリの範囲に合うとき行われます。データ アドレス発生ロジック(DAGEN)がオンチップ メモリ外のアドレスを発生すると、デバイスは自動的に外部アクセスをします。(データ アドレス発生の詳細については、第5章データ アドレッシングを参照してください。)

3.3.1 データ メモリ設定

データ メモリは、オンチップおよびオフチップに配置できます。オンチップDARAMは、データ メモリ空間内にマップされます。’54xデバイスによっては、PMSTレジスタのDROMビットを設定することでオンチップROMの一部(そのサイズは表3−2を参照)をデータ空間にマップできます(4−2ページ、セクション4.1CPUステータスおよび制御レジスタを参照)。このオンチップROM部分は、データ空間(DROMビット)およびプログラム空間(MP/$\overline{\text{MC}}$ビット)の両方でイネーブルにできます。これにより、ユーザーはROM領域をデータ空間に置かれたデータROMとして使用できます。リセット時に、’54xはDROMビットを0にクリアします。

データROMは、32ビット ロング ワード オペランドをともなう命令などの、シングル データ メモリ オペランド アドレッシングを使う命令によって1サイクルでアクセスされます。デュアル メモリ オペランド アドレッシングでは、両方のオペランドが同じブロックにある場合は2サイクルが必要になります。オペランドが異なるブロックにある場合は、アクセスには1サイクルが必要になります。ROMブロックのアドレス境界については、3−11ページ、サブセクション3.2.2オンチップROM構成を参照しください。

各'54xデバイスのデータ メモリ構成については、図3−1から図3−4(3−3ページから3−6ページ)を参照してください。

3.3.2 オンチップRAM構成

オンチップRAM構成は、ブロックごとに分割された構成で、性能を向上しています。たとえば、このブロック構成によりDARAMのある1ブロックから2つのオペランドをフェッチして、同じサイクルで別のDARAMブロックに書き込むことができます。3−16ページの図3−10は、各'54xデバイスのRAMブロック構成を示しています。点線はブロック境界を示しています。

3−17ページの図3−11は、あらゆる'54xデバイスにおけるDARAMの最初の1Kの構成を示しています。このデータ メモリ部分には、メモリ マップCPUおよびペリフェラル レジスタ、32ワードのスクラッチパッドDARAM、896ワードのDARAMが含まれます。図のグレーのラインは、直接メモリ アドレッシングで使用されるメモリ ページ境界を示しており(5−7ページ、セクション5.4直接アドレッシング参照)、DPはST0のデータ ページ ポインタを意味します(4−2ページ、セクション4.1CPUステータスおよび制御レジスタ参照)。

図3−10. **オンチップ**RAM**ブロック構成**

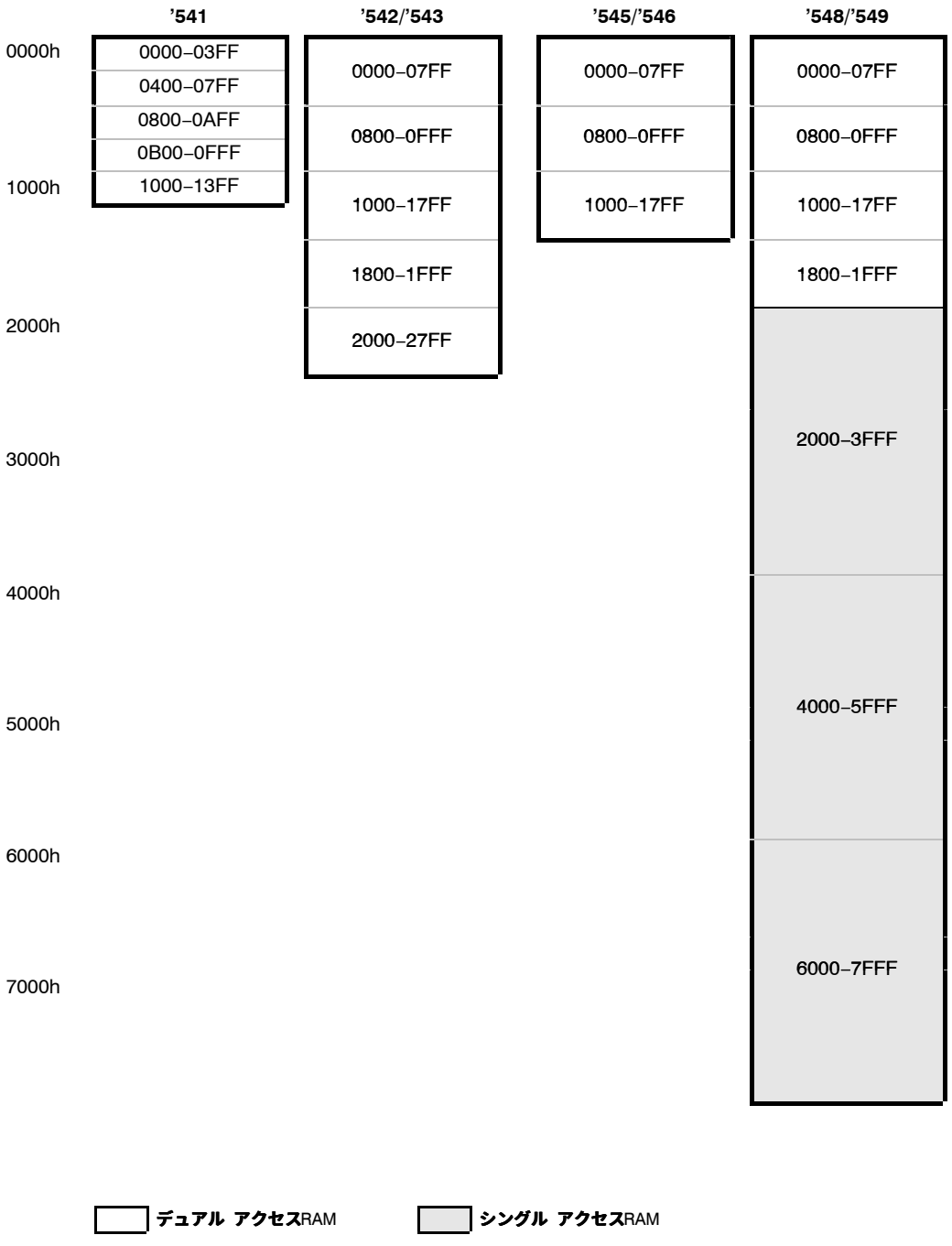
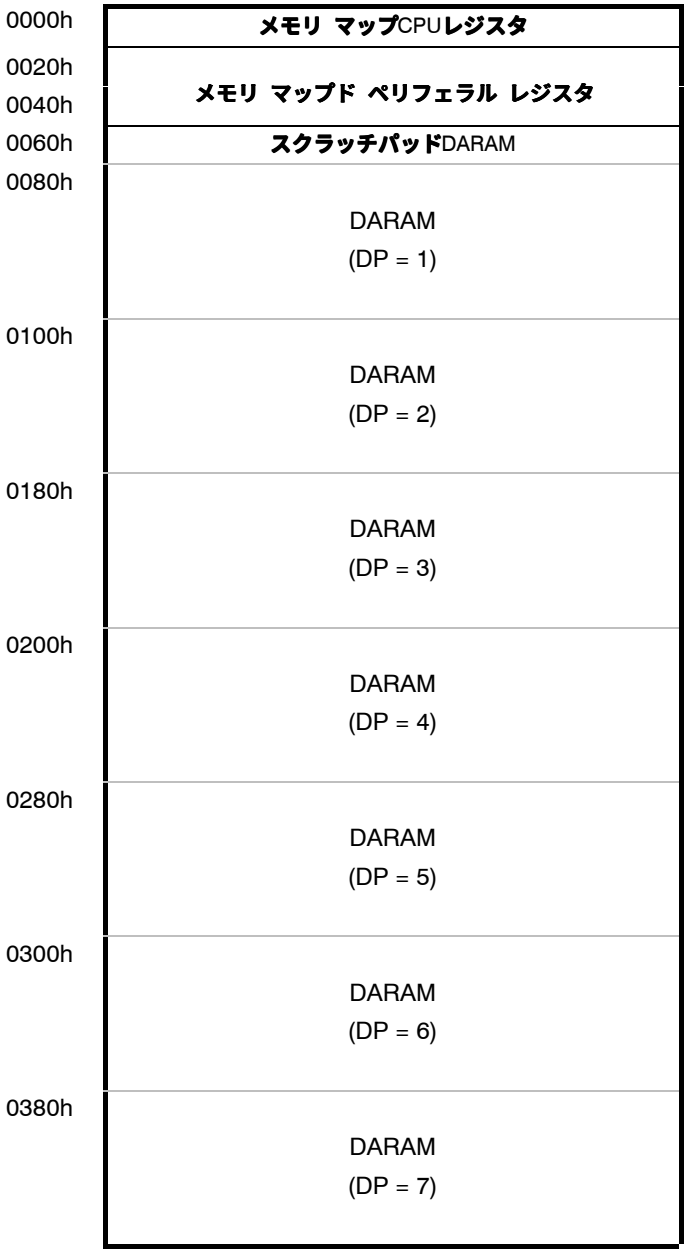


図3−11. **オンチップ データ メモリの下位アドレス**



3.3.3 メモリ マップド レジスタ

64Kワードのデータ メモリ空間はデバイスのメモリ マップ レジスタを含み、これはデータ ページ0(データ アドレス0000h−007Fh、3−17ページ図3−11参照)に配置されます。データ ページ0は次の項目から構成されます。

- CPUレジスタ(トータル26)はウェイト ステートなしでアクセスできます。3−19ページ表3−3を参照してください。
- ペリフェラル レジスタは、ペリフェラル回路の制御およびデータ レジスタとして使用されます。これらのレジスタは、アドレス0020h−005Fに置かれて、TIBUSと呼ばれる専用ペリフェレラル バス構造上に位置付けられます。これらは、アクセス時に最小限2サイクルを必要とします。必要なサイクル数は、'54xデバイスに組み込まれたペリフェラルに応じます。各'54xデバイスのペリフェラルのリストは、8−2ページ、セクション8.1ペリフェラル メモリ マップド レジスタを参照しください。
- スクラッチパッドRAMブロック(データ メモリの60h−7Fh)には、変数記憶のための32ワードのDARAMがあります。大容量RAMブロックの細分化防止に役立ちます。

3.3.4 CPUメモリ マップド レジスタ

表3−3は、CPUメモリ マップド レジスタを示しています。以降のサブセクションには、レジスタについて概説します。

3.3.4.1 割り込みレジスタ(IMR、IFR)

割り込みマスク レジスタ(IMR)は、それぞれの割り込みに個別にマスク(無視)を施します。割り込みフラグ レジスタ(IFR)は、割り込みの現在の状態を示します。割り込みについては、6−26ページ、セクション6.10割り込みを参照してください。

3.3.4.2 ステータス レジスタ(ST0、ST1)

ステータス レジスタST0およびST1には、'54xデバイスの様々な条件およびモードのステータスが含まれます。ST0は、算術演算およびビットマップ操作により影響されるフラグ(OVA、OVB、C、TC)とDPおよびARPフィールドを含んでいます。ST1は、CPUモードおよび命令の状態を表します。詳しくは、4−2ページ、セクション4.1CPUステータスおよび制御レジスタを参照してください。

表3−3. CPUメモリ マップド レジスタ

アドレス	名称	内容
0	IMR	割り込みマスク レジスタ
1	IFR	割り込みフラグ レジスタ
2−5	−	テスト用予約
6	ST0	ステータス レジスタ0
7	ST1	ステータス レジスタ1
8	AL	アキュムレータA下位ワード(ビット15−0)
9	AH	アキュムレータA上位ワード(ビット31−16)
A	AG	アキュムレータAガード ビット(ビット39−32)
B	BL	アキュムレータB下位ワード(ビット15−0)
C	BH	アキュムレータB上位ワード(ビット31−16)
D	BG	アキュムレータBガード ビット(ビット39−32)
E	T	Tレジスタ
F	TRN	トランジション レジスタ
10	AR0	補助レジスタ0
11	AR1	補助レジスタ1
12	AR2	補助レジスタ2
13	AR3	補助レジスタ3
14	AR4	補助レジスタ4
15	AR5	補助レジスタ5
16	AR6	補助レジスタ6
17	AR7	補助レジスタ7
18	SP	スタック ポインタ
19	BK	サーキュラ バッファ サイズ レジスタ
1A	BRC	ブロック リピート カウンタ
1B	RSA	ブロック リピート開始アドレス
1C	REA	ブロック リピート終了アドレス
1D	PMST	プロセッサ モード ステータス レジスタ
1E	XPC	プログラム カウンタ拡張レジスタ('548、549)
1E−1F	−	予約

3.3.4.3 アキュムレータ(A、B)

’54xデバイスには、40ビット アキュムレータのアキュムレータAおよびアキュムレータBがあります。各アキュムレータは、アキュムレータ下位ワード(AL、BL)、アキュムレータ上位ワード(AH、BH)、アキュムレータ ガード ビット(AG、BG)に分割されており、それぞれはメモリにマップされています。これらのアキュムレータ機能については、4−15ページ、セクション4.3アキュムレータAおよびBを参照してください。

3.3.4.4 Tレジスタ

Tレジスタには、各種の利用法があり、以下の例のようにある特定の値を保持します。

- 乗算および積和命令のための被乗数のひとつ。(Tレジスタおよび乗算プロセスの詳細については、4−21ページ、セクション4.5積和演算ユニットを参照してください。)
- シフト処理をともなうADD、LD、SUB命令などのためのダイナミック(実行時にプログラマブル)シフト カウント値。
- BITT命令のためのダイナミック ビット アドレス値。
- DADSTとDSADT命令がビタビ 復号のACS(Add Compare Select)処理のために使用する分岐メトリクス値。

さらに、EXP命令はTレジスタに算出される、べき指数値を記憶してNORM命令がTレジスタ値を使ってその数を正規化します。

3.3.4.5 トランジション レジスタ

16ビット トランジション(TRN) レジスタは、ビタビ アルゴリズムを実行するために、新しいメトリクス パスの選択された状態遷移を保持します。CMPS(大小比較選択およびストア)命令は、アキュムレータ上位ワードおよびアキュムレータ下位ワードの比較を基に、TRNレジスタの内容を更新します。

3.3.4.6 補助レジスタ(AR0–AR7)

8つの16ビット補助レジスタ(AR0−AR7)にはCPUによってアクセスでき、補助レジスタ算術ユニット(ARAU)によって変更されます。補助レジスタは主に、データ空間の16ビット アドレスを生成するために用いられます。又、これらのレジスタは汎用レジスタまたはカウンタとしても使用できます。データ メモリ アドレッシングにおける補助レジスタの役割については、5−10ページ、セクション5.5間接アドレッシングを参照してください。

3.3.4.7 スタック ポインタ レジスタ(SP)

16ビットのスタック ポインタ レジスタ(SP)は、システム スタックのトップアドレスを保持しています。SPは常に、スタックにプッシュされた最後の要素を指しています。このスタックは、割り込み、トラップ、サブルーチン コール、リターン、プッシュ、ポップ命令によって操作されます。スタックのプッシュはスタック ポインタの値をプリデクリメントして、スタックのポップはスタック ポインタの値をポストインクリメントして行われます。

3.3.4.8 サーキュラ バッファ サイズ レジスタ(BK)

ARAUは、16ビットのサーキュラ バッファ サイズ レジスタ(BK)をサーキュラ アドレッシングで使用して、データ ブロック サイズを指定します。BKおよびサーキュラ アドレッシングについては、5−15ページ、サブセクション5.5.3.4サーキュラ アドレスの修飾を参照してください。

3.3.4.9 ブロック リピート レジスタ(BRC、RSA、REA)

16ビット ブロック リピート カウンタ(BRC)レジスタは、ブロック リピート実行時に命令ブロックを何回繰り返すかを指定します。16ビットのブロック リピート開始アドレス(RSA)レジスタは、繰り返される命令ブロックの開始アドレスを保持します。16ビットのブロック リピート終了アドレス(REA)レジスタは、繰り返される命令ブロックの終了アドレスを保持しています。命令ブロックの繰り返しおよびBRC、RSA、REAについては、6−23ページ、セクション6.8命令ブロックの繰り返しを参照してください。

3.3.4.10 プロセッサ モード ステータス レジスタ(PMST)

プロセッサ モード ステータス レジスタ(PMST)は、'54xデバイスのメモリ構成等を制御します。PMSTについては、4−2ページ、セクション4.1CPUステータスおよび制御レジスタを参照してください。

3.3.4.11 プログラム カウンタ拡張レジスタ(XPC、'548と'549のみ)

プログラム カウンタ拡張レジスタ(XPC)は、現在のプログラム メモリ アドレスの上位7ビットを保持しています。拡張メモリについては、3−8ページ、サブセクション3.1.1拡張プログラム メモリを参照してください。

3.4 I/Oメモリ

’54xデバイスには、プログラムおよびデータ メモリ空間の他にI/Oメモリ空間があります。I/Oメモリ空間は、64Kワードのアドレス空間(0000h−FFFFh)で、デバイスの外部にのみ存在します。PORTR、PORTWの2つの命令を使って、この空間にアクセスすることができます。I/O空間のリード タイミングは、個々のI/Oマップ デバイスへのアクセスを容易にするため、プログラム及びデータ空間へのアクセスとは異なります。外部バス機能およびI/Oアクセス制御については、第10章、外部バス機能を参照してください。

3.5 プログラムおよびデータ セキュリティ

’54xデバイスのROMにユーザコードをマスク(プログラム)する場合、2種類のセキュリティ オプション、オンチップROMセキュリティおよびROM/RAMセキュリティが選べます。プログラムのマスクROM時に指定して下さい。表3−4は、セキュリティ機能をまとめたものです。

表3−4. データ セキュリティ

	オンチップ メモリ アクセスへの影響	
セキュリティ	ROM	RAM
ROM	プログラムおよびデータ アクセス両方。 オンチップROMからフェッチされた命令は、プログラムおよびデータ アクセスの際にオンチップROMからリード可能。 オンチップRAMまたは外部メモリからフェッチされた命令は、オンチップROMへのアクセスを禁じられ、プログラムおよびデータ アクセスの際に無効なデータ(0FFFFh)をリード。	セキュリティ オプションによる影響なし。
ROM/RAM	プログラムおよびデータ アクセス両方。 オンチップROMおよびオンチップRAMからフェッチされた命令は、プログラムおよびデータ アクセスの際にオンチップROMからリード可能。 外部メモリからフェッチされた命令は、オンチップROMへのアクセスを禁じられ、プログラムおよびデータ アクセスの際に無効なデータ(0FFFFh)をリード。	プログラムおよびデータ アクセス両方。 オンチップROMおよびオンチップRAMからフェッチされた命令は、プログラムおよびデータ アクセスの際にオンチップRAMからリード可能。 外部メモリからフェッチされた命令は、オンチップRAMへのアクセスを禁じられ、プログラムおよびデータ アクセスの際に無効なデータ(0FFFFh)をリード。

第4章

中央演算ユニット

本章では、’54xデバイスの中央演算処理ユニット(CPU)機能について説明します。CPUは、その並列アーキテクチャ設計により、高速算術演算を1命令サイクルで実行できます。

本章では、次のCPU機能部分について説明します。

- 40ビット算術論理演算器(ALU)
- 2つの40ビット アキュムレータ レジスタ
- −16から31シフト範囲をサポートするバレル シフタ
- 積和演算ブロック
- 16ビットTレジスタ(T)
- 16ビットトランジションレジスタ(TRN)
- 比較選択ストアユニット(CSSU)
- べき指数エンコーダ

CPUレジスタは、メモリ マッピングされ高速退避および復元を可能にします。

Topic **Page**

4.1 CPUステータスおよび制御レジスタ

’54xデバイスには、3つのステータスおよび制御レジスタがあります。

- ステータス レジスタ0(ST0)
- ステータス レジスタ1(ST1)
- プロセッサ モード ステータス レジスタ(PMST)

ST0およびST1には、様々な条件およびモードのステータスが含まれています。PMSTは、メモリ設定ステータスおよび制御情報を含んでいます。これらのレジスタはメモリ マッピングされているので、データ メモリにストアでき、そこからロードもできます。プロセッサのステータスは、サブルーチンおよび割り込みサービス ルーチン(ISR)において退避、復元できます。

4.1.1 ステータス レジスタ(ST0およびST1)

ST0およびST1のそれぞれのビットは、SSBXおよびRSBX命令によってセットまたはクリアできます。たとえば、符号拡張モードは”SSBX 1，SXM”というインストラクションによってセットされ、”RSBX 1，SXM”によってリセットされます。ARP、DP、ASMビット フィールドを、ショート イミディエイト オペランドをともなうLD命令を使ってその設定値をロードすることができます。またASMおよびDPフィールドには、LD命令を使ってデータ メモリの値をロードすることも可能です。

ST0ビットについては図4−1および表4−1、ST1ビットについては図4−2および表4−2を参照してください。

図4−1. ステータス レジスタ0(ST0)

15–13	12	11	10	9	8–0
ARP	TC	C	OVA	OVB	DP

表4−1. ステータス レジスタ0(ST0)ビット概要

ビット	名称	リセット時の値	機能
15 − 13	ARP	0	補助レジスタ ポインタ。この3ビット フィールドは、間接シングル オペランド アドレッシングの互換モードで使用するために補助レジスタを選択します(5−10ページ、セクション5.5間接アドレッシングを参照)。ARPは、DSPが標準モード(CMPT=0)のときは常にゼロに設定されます。

表4−1. ステータス レジスタ0(ST0)ビット概要 (続き)

ビット	名称	リセット時の値	機能
12	TC	1	**テスト/制御フラグ。**TCは、算術論理演算器(ALU)テスト ビット処理の結果を反映します。TCには、BIT、BITF、BITT、CMPM、CMRP、CMPS、SFTC命令の実行結果が反映されます。TCのステータス(セットまたはクリア)により、条件分岐、条件付きサブルーチンコール、条件付き実行、条件付きリターン命令を実行するかどうかが決定されます。 次の条件が真(true)のとき TC = 1: ❑ BITまたはBITTにテストされたビットの内容が1。 ❑ CMPM命令において、データ メモリの値とイミディエイト オペランドの値が等しい場合、CMPR命令において、AR0と他の補助レジスタ(ARx)の比較結果が指定した条件と一致する場合、およびCMPS命令において、アキュムレータの上位16ビットおよび下位16ビットにある2つの2の補数の値を比較した結果、下位側の値が上位側の値以下である場合。 ❑ SFTCによってテストされるアキュムレータのビット31およびビット30が、おのおの異なる値を持つ。
11	C	1	加算の結果がキャリーを発生する場合は、キャリー フラグは1にセットされます。減算の結果がボローを発生する場合は、キャリー フラグは0にクリアされます。それ以外の場合は、加算の後にリセットされ、減算の後にセットされます。ただし、16ビット シフトをともなうADDまたはSUBを除きます。これらの場合、ADDはキャリー フラグのセット、SUBはキャリー フラグのリセットのみが可能ですが、その他の場合は影響を与えることはできません。キャリーおよびボローは、ビット32で指定され、ALUにのみ影響します。シフトおよびローテート命令(ROR、ROL、SFTA、SFTL)およびMIN、MAX、ABS、NEG命令もこのビットに影響を与えます。
10	OVA	0	**アキュムレータAのオーバーフロー フラグ。**OVAは、オーバーフローがALUまたは積和演算器の加算器部のいずれかで発生して結果の格納先がアキュムレータAのとき、1にセットされます。オーバーフローが発生すると、OVAはリセット、または AOVおよびANOV条件を使ったBC[D]、CC[D]、RC[D]、XC命令が実行されるまで、セットされたままになります。RSBX命令もこのビットをクリアできます。
9	OVB	0	**アキュムレータBのオーバーフロー フラグ。**OVBは、オーバーフローがALUまたは積和演算器の加算器部のいずれかで発生して結果の格納先がアキュムレータBのとき、1にセットされます。オーバーフローが発生すると、OVBはリセット、または BOVおよびBNOV条件を使ったBC[D]、CC[D]、RC[D]、XC命令が実行されるまで、セットされたままになります。RSBX命令もこのビットをクリアできます。
8 − 0	DP	0	データ メモリ ページ ポインタ。この9ビット フィールドは、命令ワードのLSB側の7ビットと連結して、シングル データ メモリ オペランド アドレッシングのための16ビットの直接メモリ アドレスを形成します。この処理は、ST1のコンパイラ モード ビットが(CPL)=0のとき実行されます。DPフィールドは、ショート イミディエイト オペランド、またはデータ メモリ オペランドをともなうLD命令によってロードできます。

図4−2. **ステータス レジスタ**1(ST1)

15	14	13	12	11	10	9	8	7	6	5	4−0
BRAF	CPL	XF	HM	INTM	0	OVM	SXM	C16	FRCT	CMPT	ASM

表4−2. **ステータス レジスタ**1(ST1)**ビット概要**

ビット	名称	リセット値	機能
15	BRAF	0	**ブロック リピート アクティブ フラグ。**BRAF**は、ブロック リピートが現在アクティブかどうかを示します。** BRAF = 0 **ブロック リピートはアクティブでありません。**BRAF**は、ブロック リピート カウンタ**(BRC)**が0未満にデクリメントするとクリアされます。** BRAF = 1 **ブロック リピートはアクティブです。**BRAF**は、**RPTB**命令が実行されると自動的にセットされます。**
14	CPL	0	**コンパイラ モード。**CPL**は、直接アドレッシングでどのポインタが使われるかを示します。** CPL = 0 **データ ページ ポインタ**(DP)**を使う直接アドレッシング モードが選択されます。** CPL = 1 **スタック ポインタ**(SP)**を使う**SP**相対直接アドレッシング モードが選択されます。**
13	XF	1	XF**ステータス。**XF**は、汎用出力ピンである外部フラグ**(XF)**ピンのステータスを示します。**SSBX**命令は**XF**をセットして、**RSBX**命令は**XF**をリセットすることができます。**
12	HM	0	**ホールド モード。**HM**は、アクティブ**$\overline{\text{HOLD}}$**信号がアクティブであると認識されたときプロセッサが内部実行を継続するかどうかを示します。** HM = 0 **プロセッサは内部プログラム メモリからの実行を続けますが、外部インタフェースをハイ インピーダンス ステートにします。** HM = 1 **プロセッサは内部実行を停止します。**
11	INTM	1	**割り込みモード。**INTM**はグローバルに割り込みをマスクまたはイネーブルにします。** INTM = 0 **マスクされていない全ての割り込みがイネーブルになります。** INTM = 1 **マスカブル割り込みすべてがディセーブルになります。** SSBX**命令は**INTM**をセットして、**RSBX**命令は**INTM**をリセットします。**INTM**は、リセットによって、またはマスカブル割り込みトラップが受け付けられるとき**(INTR**または外部割り込み)、**1**にセットされます。**INTM**は、**RETE**または**RETF**命令(割り込みからの復帰)が実行されると、**0**にクリアされます。**INTM**は、ノンマスカブル割り込み(**$\overline{\text{RS}}$**および**$\overline{\text{NMI}}$**)には影響を与えません。**INTM**は、メモリ ライトで設定することはできません。**
10		0	**常にゼロがリードされます。**

表4−2. ステータス レジスタ1(ST1)ビット概要

ビット	名称	リセット値	機能
9	OVM	0	**オーバーフロー モード。**OVMは、オーバーフロー発生時に、格納先アキュムレータに何がロードされるかを決めます。 OVM = 0　ALUまたは積和演算器の加算器部分からオーバーフローしたままの結果が通常、格納先アキュムレータにストアされます。 OVM = 1　格納先アキュムレータは、オーバーフロー時に正の最大の値(00 7FFF FFFFh)または負の最大の値(FF 8000 0000h)に設定されます。 SSBXおよびRSBX命令は、OVMをそれぞれセットおよびリセットします。
8	SXM	1	**符号拡張モード。**SXMは、符号拡張が実行されるかどうかを判断します。 SXM = 0　符号拡張はされません。 SXM = 1　データはALUが使用する前に符号拡張されます。 SXMは、特定の命令には影響しません。ADDS、LDU、SUBS命令は、SXMの値にかかわらず符号拡張されません。SSBXおよびRSBX命令は、それぞれSXMをセットおよびリセットします。
7	C16	0	**デュアル16ビット倍精度演算モード。**C16は、ALUの演算モードを決めます。 C16 = 0　ALUは倍精度演算モードを実行します。 C16 = 1　ALUはデュアル16ビット演算モードを実行します。
6	FRCT	0	**小数モード。**FRCTが1のとき、乗算器出力は1ビットで左シフトして、冗長符号ビットを補正します。
5	CMPT	0	**互換モード。**CMPTはARPの互換モードを決めます。 CMPT = 0　ARPは、シングル データ メモリ オペランドをともなう間接アドレッシング モードでは更新されません。ARPは、DSPがこのモードのとき必ず0に設定されなくてはなりません。 CMPT = 1　ARPは、シングル データ メモリ オペランドをともなう間接アドレッシング モードで更新されます。ただし、この命令が補助レジスタ0(AR0)を選択するときは除きます。
4 − 0	ASM	0	**アキュムレータ シフト モード。**5ビットASMフィールドは、−16から15の範囲でシフト値を指定して、2の補数として表されます。 STH、STL、ADD、SUB、LDの様に並列ストアをともなう命令は、このシフト機能を使用します。ASMは、データ メモリからロードでき、またはショート イミディエイト オペランドをともなうLD命令によってロードできます。

4.1.2 プロセッサ モード ステータス レジスタ(PMST)

PMSTレジスタは、STMなどメモリ マップ レジスタ命令によってロードします。PMSTの各ビットについては、図4−3および表4−3を参照してください。

図4−3. プロセッサ モード ステータス レジスタ(PMST)

15−7	6	5	4	3	2	1	0
IPTR	MP/$\overline{MC}$	OVLY	AVIS	DROM	CLKOFF	SMUL†	SST†

† 一部デバイスのみ(付録Dを参照)。その他のデバイスでは予約。

表4−3. プロセッサ モード ステータス レジスタ(PMST)ビット概要

ビット	名称	リセット時の値	機能
15 − 7	IPTR	1FFh	割り込みベクタ ポインタ。9ビットのIPTRフィールドは、128ワードのプログラム ページを指し、そこに割り込みベクタが置かれます。ブート ロードされたアプリケーションのために、割り込みベクタをRAMにリマップすることもできます。リセット時に、これらのビットはすべて1に設定されます。リセット ベクタは、常にプログラム メモリ空間のアドレスFF80hに置かれます。RESET命令は、このフィールドに影響を与えません。
6	MP/$\overline{MC}$	MP/$\overline{MC}$ pin	マイクロプロセッサ/マイクロコンピュータ モード。MP/$\overline{MC}$は、オンチップROMがプログラム メモリ空間でアドレッシング可能になることを、イネーブル/ディセーブルにします。 MP/$\overline{MC}$ = 0 オンチップROMをイネーブルにしてアドレッシング可能にします。 MP/$\overline{MC}$ = 1 オンチップROMは使えません。 MP/$\overline{MC}$は、リセット時のサンプルされ、MP/$\overline{MC}$ピンのロジック レベルに対応する値に設定されます。このピンは、次のリセットまで再びサンプルされません。RESET命令は、このビットに影響を与えません。このビットは、ソフトウェアによってセット、リセットできます。
5	OVLY	0	RAMオーバーレイ。OVLYは、オンチップデータRAMブロックをプログラム空間にマッピング可能にします。 OVLY = 0 オンチップRAMはデータ空間でアドレッシング可能ですが、プログラム空間ではアドレッシングできません。 OVLY = 1 オンチップRAMはプログラム空間およびデータ空間にマッピングされます。ただし、データ ページ0(アドレス0hから7Fh)はプログラム空間にマッピングされません。

† 一部デバイスのみ(付録Dを参照)。その他のデバイスでは予約。

表4−3. プロセッサ モード ステータス レジスタ(PMST)ビット概要 (続き)

ビット	名称	リセット時の値	機能
4	AVIS	0	**アドレス可視モード。**AVISは、内部プログラム アドレスをアドレス ピンでビジブル(可視)にすることをイネーブル/ディセーブルにします。 AVIS = 0 **外部アドレス ラインは、内部プログラム アドレスによって変化しません。制御およびデータ ラインは影響を受けず、アドレス バスは最後にアクセスされたアドレスによってドライブします。** AVIS = 1 **このモードでは、内部プログラム アドレスが'54xのアドレスピンに現れるので、内部プログラム アドレスをトレースすることができます。また、割り込みベクタがオンチップ メモリに置かれているとき、IACKを利用して割り込みベクタをデコードできます。**
3	DROM	0	**データ**ROM**。**DROM**は、**ROM**をデータ空間にマッピング可能にします。** DROM = 0 **オンチップ**ROM**はデータ空間にマッピングされません。** DROM = 1 **オンチップ**ROM**の一部はデータ空間にマッピングされます。詳しくは、第3章メモリを参照してください。**
2	CLKOFF	0	CLOCKOUT**オフ。**CLKOFF**ビットが**1**のときは、**CLKOUT**の出力はディセーブルになりハイレベルに固定されます。**
1	SMUL†	N/A	**乗算器の飽和機能。**SMUL=1**のとき、**MAC**または**MAS**命令のアキュムレーションを実行する前に、乗算結果が飽和されます。**SMUL**ビットは、**OVM=1**および**FRCT=1**のときにのみ適用します。** SMUL**ビットにより、**MAC**および**MAS**処理を**ETSI GSM**仕様(**GSM**仕様6.06、6.10、6.53)で定義されている**MAC**および**MAS**処理に準拠させることができます。その効果としては、小数点演算モードで**8000h x 8000h**の結果が**7FFF FFFFh**に飽和して、その後で**MAC**または**MAS**命令の加算/減算を実行します。このモードでは、**OVM=1**のとき、**MAC**命令は**MPY + ADD**と等しくなります。このモードが設定されず**OVM=1**のときは、加算/減算を実行する前に乗算の結果は飽和されず、**MAC**および**MAS**命令の結果のみが飽和されます。**

† **一部デバイスのみ(付録Dを参照)。その他のデバイスでは予約。**

表4−3. プロセッサ モード ステータス レジスタ(PMST)ビット概要 (続き)

ビット	名称	リセット時の値	機能
0	SST†	N/A	**ストア時の飽和。**SST=1**のとき、アキュムレータのデータがメモリへストアされる前に飽和されます。飽和は、シフト処理の後に実行されます。ストア時の飽和は、**STH、STL、STLM、DST、STIIADD、STIILD、STIIMACR[R]、STIIMAS[R]、STIIMPY、STIISUB**命令の実行時に行われます。ストア時の飽和を使用する場合は、次の手順が実行されます。** 1) 40**ビット データ値が命令に応じてシフト(右または左)されます。シフトは、**SFTA**命令と同じ処理が行われ、**SXM**ビットでイネーブル/ディセーブルされます。** 2) 40**ビット データ値が、**32**ビット値に飽和されます。この飽和は、**SXM**ビットに応じます。** SXM=0**なら、次の**32**ビット値が発生します。** ■ **その値が**FFFF FFFFh**より大きいときは**FFFF FFFFh**。** SXM=1**なら、次の**32**ビット値が発生します。** ■ **その値が**7FFF FFFFh**より大きいときは**7FFF FFFFh**。** ■ **その値が**8000 0000h**より小さいときは**8000 0000h**。** 3) **データが、命令に応じてメモリにストアされます。** 4) **飽和処理の間、アキュムレータの内容は変化しません。**

† **一部デバイスのみ(付録D**を参照**)。その他のデバイスでは予約。**

例4−1. SMUL**ビットの使用法**

```
MAC    *AR1+, A      ;SMUL=1, FRCT=1, OVM=1, SXM=1
```

	Before Instruction		After Instruction
A	FF FFFF FFFFh	A	00 7FFF FFFEh
T	8000h	T	8000h
AR1	100h	AR1	101h
Data Memory		Data Memory	
100h	8000h	100h	8000h

```
MAC    *AR1+, A      ;SMUL=0, FRCT=1, OVM=1, SXM=1
```

	Before Instruction		After Instruction
A	FF FFFF FFFFh	A	00 7FFF FFFFh
T	8000h	T	8000h
AR1	101h	AR1	102h
Data Memory		Data Memory	
100h	8000h	100h	8000h

例4−2. SST**ビットの使用法**

```
STH A, •4, *AR1+ ; SXM=1, SST=1
```

	Before Instruction		After Instruction
A	7F FFFF 0000h	A	7F FFFF 0000h
AR1	100h	AR1	101h
Data Memory		Data Memory	
100h	5555h	100h	7FFFh

```
STL A, •4, *AR1+ ; SXM=1, SST=1
```

	Before Instruction		After Instruction
A	7F FFFF 0000h	A	7F FFFF 0000h
AR1	101h	AR1	102h
Data Memory		Data Memory	
101h	0AAAAh	101h	0FFFFh

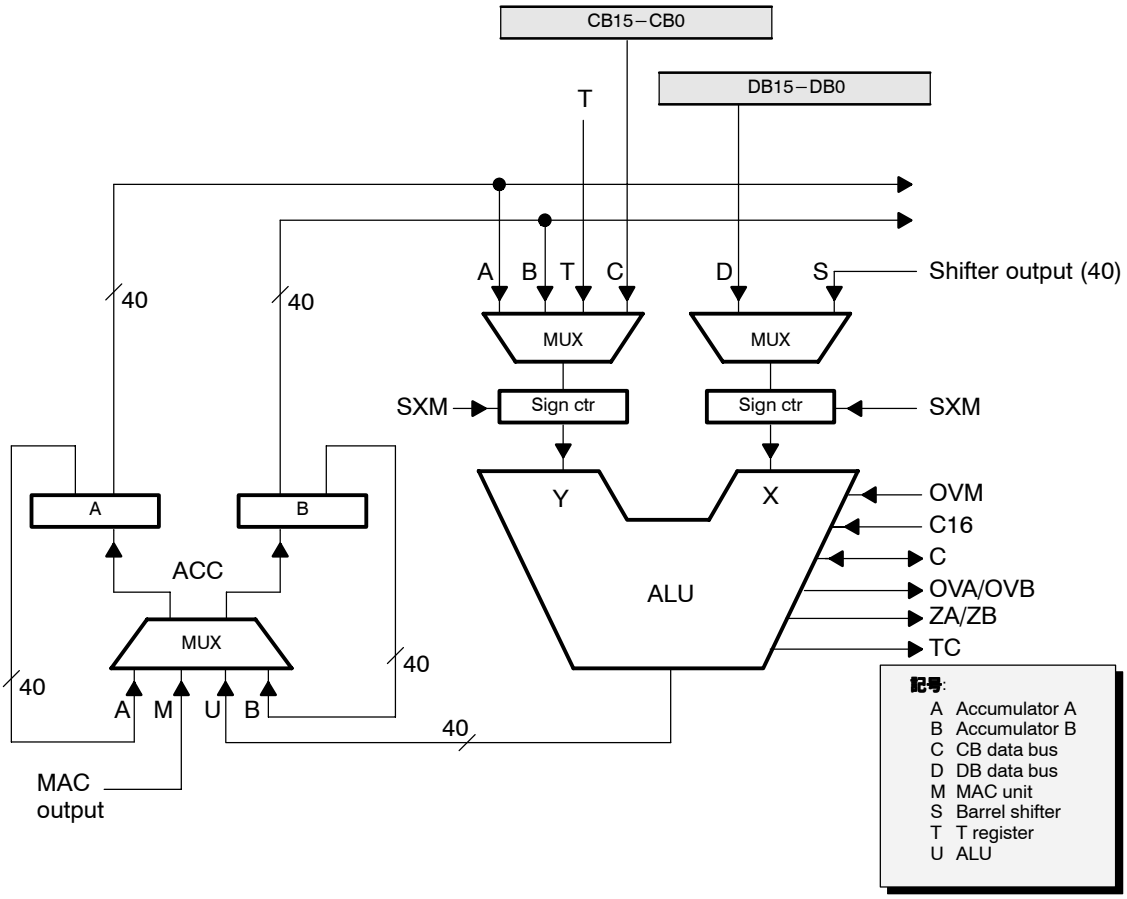
```
DST B, *AR3• ; SXM=0, SST=1
```

	Before Instruction		After Instruction
B	8F FFFF 0000h	B	8F FFFF 0000h
AR3	103h	AR3	101h
Data Memory		Data Memory	
102h	1234h	102h	0FFFFh
103h	5678h	103h	0FFFFh

4.2 算術論理演算ユニット(ALU)

図4−4に示されるように、40ビットALUは幅広い算術および論理機能を実現しており、そのほとんどは1クロック サイクルで実行されます。ALUで処理が実行されると、通常はその結果が格納先アキュムレータに転送されます(アキュムレータAまたはB)。メモリ−to−メモリ処理(ADDM、ANDM、ORM、XORM)を実行する命令は例外になります。

図4−4. ALU機能図



4.2.1 ALU入力

ALU入力は、複数ソースからの複数形式を取ります。

ALUへのX入力ソースは、次の2つの値のうちいずれかになります。

- シフタ出力(32ビットまたは16ビット データ メモリ オペランドまたはシフトされたアキュムレータの値)

❑ データ バスDBからのデータ メモリ オペランド。

ALUへのY入力ソースは、次の3つの値のうちいずれかになります。

❑ アキュムレータAまたはBのいずれかの値。

❑ データバスCBからのデータメモリ オペランド。

❑ Tレジスタの値。

16ビット データ メモリ オペランドがデータ バスCBまたはDBによってリードされるとき、40ビットALU入力が次の2つの方法のうちいずれかで構成されます。

❑ 15から0のビットがデータ メモリ オペランドを含むとき、39から16のビットはゼロ詰めか(SXM=0)または符号拡張されます(SXM=1)。

❑ 31から16のビットがデータ メモリ オペランドを含むとき、15から0のビットはゼロ詰め、さらに39から32のビットがゼロ詰めされるか(SXM=0)または符号拡張されます(SXM=1)。

表4−4は、使用される構文のタイプに応じて、ALU入力がどのようにADD命令のために使用されるかを示しています。ADD命令は1サイクルで実行されますが、2ワードを使うケース4、7、8は2サイクルになります。

表4−4. ADD命令のためのALU入力選択

ケース	命令構文		ワード	A	B	DB	CB	シフト
1	ADD	*AR1, A	1	√				√
2	ADD	*AR3, TS, A	1	√				√
3	ADD	*AR2, 16, B, A	1		√			√
4	ADD	*AR1, 8, B, A	2		√			√
5	ADD	*AR2, 8, B	1		√			√
6	ADD	*AR2, *AR3, A	1			√	√	
7	ADD	#1234h, 6, A, B	2	√				√
8	ADD	#1234h, 16, A, B	2	√				√
9	ADD	A, 12, B	1		√			√
10	ADD	B, ASM, A	1	√				√
11	DADD	*AR2, A, B	1	√				√

4.2.2 オーバーフロー処理

ALU飽和ロジックは、結果を最大値(または最小値)に保持することによって、結果がオーバーフローするのを防ぎます。この機能は、フィルタ演算に役立ちます。このロジックは、ステータス レジスタST1のオーバーフロー モード ビット(OVM)がセットされるとき、イネーブルになります。

結果がオーバーフローするとき、

- ❑ OVM=0なら、アキュムレータにALUの結果が修正なしにロードされます。
- ❑ OVM=1なら、アキュムレータに、オーバーフローの符号により、最も正の32ビット値(00 7FFF FFFFh)または最も負の32ビット値(0FF 8000 0000h)のいずれかがロードされます。
- ❑ ステータス レジスタST0のオーバーフロー フラグ(OVA/OVB)は、格納先アキュムレータに応じてセットされ、次のいずれかが発生するまでセットされたままになります。
  - ■ リセットが実行されたとき。
  - ■ 条件付き命令(分岐、リターン、サブルーチンコール、実行など)がオーバーフロー条件で実行されたとき。
  - ■ オーバーフロー フラグ(OVA/OVB)がクリアされたとき。

注:

OVMの値にかかわらず、SAT命令によってアキュムレータを飽和させることができます。

4.2.3 キャリービット

ALUにはキャリービット(C)があり、ローテート、シフト処理などほとんどの算術ALU命令によって影響を受けます。キャリービットは、拡張精度演算を効率的にサポートします。キャリービットは、アキュムレータのロード、論理演算の実行、またはその他の非算術命令または制御命令の実行による影響を受けません。したがってオーバーフロー管理に使用できます。

2つの条件付きオペランドであるCおよびNCは、キャリービットのステータス(セットまたはクリア)に応じて、分岐、サブルーチンコール、リターン、条件付き実行をイネーブルにします。またRSBXおよびSSBX命令を使って、キャリービットをロードできます。
キャリービットは、ハードウェア リセット時にセットされます。

4.2.4 デュアル16ビット モード

算術演算のために、ALUは特別のデュアル16ビット算術モードで実行でき、このモードは2つの16ビット演算(2つの加算または2つの減算など)を1サイクルで実行します。このモードは、ST1のC16フィールドをセットすることにより選択できます。このモードは、特にビタビ加算比較選択処理に役立ちます(4−26ページ、セクション4.6比較選択ストア ユニットを参照してください)。

4.3 アキュムレータAおよびB

アキュムレータAおよびアキュムレータBは、乗算器/加算機ユニットまたはALUいずれかのための行き先レジスタとして構成できます。さらに、MINおよびMAX命令のために使用されるか、または並列命令LD || MACのために使用され、この場合一方のアキュムレータがデータをロードして他方のアキュムレータが演算を実行します。

図4−5、図4−6に示すように、各アキュムレータは3部分に分割さます。

図4−5. アキュムレータA

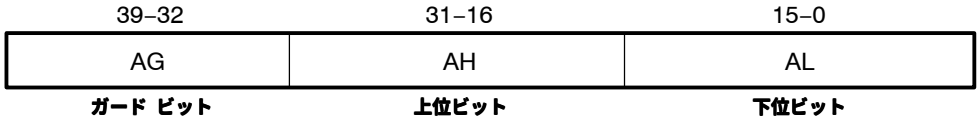
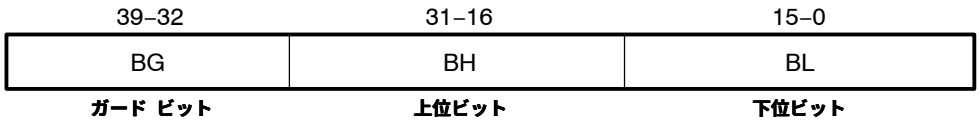


図4−6. アキュムレータB



ガード ビットは、演算のヘッドマージンとして利用されます。ヘッドマージンによりアキュムレーションのような反復演算のオーバーフローを一部防ぐことができます。

AG、BG、AH、BH、AL、BLはメモリ マップ レジスタで、PSHMおよびPOPM命令を使ってコンテキストセーブのためにスタックにプッシュおよびスタックからポップすることができます。これらのレジスタは、メモリ マップ レジスタ(MMR)を使う他の命令が使用することもできます。アキュムレータAおよびアキュムレータBの唯一の違いは、Aのビット32−16を積和演算ユニットの乗算器への入力として使用できることです。

4.3.1 アキュムレータの内容のストア

STH、STL、STLM、SACCD命令を使うか、または並列ストア命令を使うことにより、アキュムレータの内容をデータ メモリにストアすることができます。メモリにアキュムレータの上位16ビットをシフトを伴ってストアするには、STH、SACCDおよび並列記憶命令を使います。右シフト処理では、AGおよびBGからのビットがAHおよびBHにシフトします。左シフト処理では、ALおよびBLからのビットがそれぞれAHおよびBHにシフトします。

メモリにアキュムレータの下位16ビットをシフトを伴ってストアするには、STL命令を使います。右シフト処理では、AHおよびBHからのビットがそれぞれALおよびBLにシフトして、LSBは失われます。

左シフト処理では、ALおよびBLのビットがゼロで満たされます。シフト処理はシフタで実行されるので、アキュムレータの内容は変化しません。例4−3は、シフトをともなうアキュムレータストア演算の結果を示しています。アキュムレータA=0FF4321 1234hを前提とします。

例4−3. シフトをともなうアキュムレータ記憶

```
STH  A,8,TEMP    ; TEMP = 2112h
STH  A,-8,TEMP   ; TEMP = FF43h
STL  A,8,TEMP    ; TEMP = 3400h
STL  A,-8,TEMP   ; TEMP = 2112h
```


4.3.2 アキュムレータ シフトおよびローテート演算

次の命令は、アキュムレータの内容を繰り上げビットによりシフトまたはローテートします。

- SFTA(算術的にシフト)
- SFTL(論理的シフト)
- SFTC(条件シフト)
- ROL(アキュムレータを左にローテート)
- ROL(アキュムレータを右にローテート)
- ROLTC(TCによりアキュムレータを左にローテート)

SFTAおよびSFTLでは、シフト カウントは−16≤SHIFT≤15で定義されます。SFTAは、SXMビットによって影響を受けます。SXM=1でSHIFTが負の値の場合は、SFTAは算術右シフトを実行してアキュムレータの符号を維持します。SXM=0のときは、アキュムレータのMSBはゼロで満たされます。SFTLはSXMビットによって影響を受けません。SFTLはビット31−0でシフト処理を実行して、シフトの方向に応じて0をMSBまたはLSBにシフトします。

SFTCは、ビット31および30が両方とも1または0のとき、1ビット左シフトを実行します。これは、最上位非符号ビットを解消することにより、アキュムレータの32ビットを正規化します。

ROLは、アキュムレータの各ビットを左に1ビットずつローテートして、キャリー ビットの値をアキュムレータのLSBにシフトし、アキュムレータのMSBの値をキャリー ビットにシフト、さらにアキュムレータのガード ビットをクリアします。

RORは、アキュムレータの各ビットを右に1ビットずつローテートして、キャリー ビットの値をアキュムレータのMSBにシフトし、アキュムレータのLSBをキャリー ビットにシフトして、さらにアキュムレータのガード ビットをクリアします。

ROLTC命令(TCによってアキュムレータを左にローテート)は、アキュムレータを左にローテートしてテスト制御(TC)ビットをアキュムレータのLSBにシフトします。

4.3.3 アキュムレータのストア時の飽和(対象デバイスは付録Dを参照)

PMSTのSSTビットは、アキュムレータのデータがそれをメモリにストアされる前に飽和されるかどうかを決めます。飽和は、シフト処理の後に実行されます。ストア時の飽和は、10種類の命令によって実行できます。

- ❑ STH
- ❑ STL
- ❑ STLM
- ❑ DST
- ❑ ST || ADD
- ❑ ST || LD
- ❑ ST || MAC[R]
- ❑ ST || MAS[R]
- ❑ ST || MPY
- ❑ ST || SUB

次の手順により、アキュムレータのストアで飽和を実行します。

1) 命令に応じて、40ビット データ値がシフト(右または左)されます。このシフトは、SFTA命令と同じ処理が行われ、SXMビットでイネーブル/ディセーブルされます。

2) 40ビット値が32ビット値に飽和になります。この飽和は、SXMビットの値に応じます。

   - ■ SXM=0のとき。40ビット値がFFFF FFFFh以上なら、FFFF FFFFhが発生します。
   - ■ SXM=1のとき。40ビット値が7FFF FFFFhより大きいなら、7FFF FFFFhが発生します。40ビット値が8000 0000hより小さいなら、8000 0000hが発生します。

3) 命令に応じて(16ビットLSB、16ビットMSB、または32ビット データ)、データがメモリにストアされます。

アキュムレータは、このプロセスの際に、変化しません。

4.3.4 特定用途命令

各アキュムレータは、並列処理をともなう特定用途命令のある特定の処理に専用のものです。これらには、FIRS命令を使用する対称FIRフィルタ処理、LMS命令を使う適応フィルタ処理、SQDST命令を使うユークリッド距離計算、その他の並列処理があります。

- ❑ FIRSは、対称FIRフィルタの処理をmultiply/accumulate(MAC)を使って加算と並列に実行します。
- ❑ LMSは、MACおよび丸めをともなう並列加算を実行して、FIRフィルタの係数を効率的に更新します。
- ❑ SQDSTは、MACおよび減算を並列に実行して、ユークリッド距離を計算します。

FIRSは、アキュムレータA(32−16)をプログラム メモリ アドレスでアドレスされたプログラム メモリの値で掛けます。その結果をアキュムレータBの値に加えます。同時に、FIRSはメモリ オペランドXmemとYmemを加算して、結果を16ビット左にシフトして、この値をアキュムレータAにロードします。

LMS命令では、アキュムレータBは入力列のたたみ込みの当座の結果およびフィルタ係数をストアします。アキュムレータAはフィルタ係数を更新します。アキュムレータAは、MACの入力としても使用でき、並列処理をともなう命令の1サイクル実行に寄与します。

SQDST命令は、2つのベクタ間の距離の二乗を計算します。アキュムレータA(32−16)が二乗されてその積がアキュムレータBに加えられます。同時にYmemがXmemから引かれて、その差がアキュムレータAに記憶されます。二乗された値は、減算Ymem − Xmemが実行される前のアキュムレータ値です。

4.4 バレル シフタ

バレル シフタは、次のような処理のスケーリングに使用します。

- ALU処理前の、入力データ メモリ オペランドまたはアキュムレータの値のプリスケーリング。
- アキュムレータの値の論理シフトまたは算術シフトを実行。
- アキュムレータの正規化。
- アキュムレータ値をデータ メモリにストアする前のアキュムレータのポストスケーリング。

40ビット シフタ(4−20ページ、図4−7参照)は、次のように接続されます。

- 入力は次の項目に接続されます。
  - 16ビット データ入力オペランドのDB
  - 32ビット データ入力オペランドのDBおよびCB
  - 2つの40ビット アキュムレータのいずれか
- 出力は次の項目に接続されます。
  - ALU入力のひとつ
  - MSW/LSW書き込み選択ユニット経由のEBバス

SXMビットは、データ オペランドの符号あり/符号なし拡張を制御します。SXMビットをセットすると符号拡張がイネーブルされます。LDU、ADDS、SUBSなどの命令は、符号なしメモリ オペランドをともない、命令の実行時に、SXMの値にかかわらず符号拡張を行いません。

シフト カウントは、何ビットシフトするかを決めます。正のシフト値は左シフトに対応し、負の値は右シフトに対応します。シフト カウントは、命令のタイプに応じて複数の方法で2の補数値として指定されます。イミディエイト オペランド、ST1のアキュムレータ シフト モード(AMS)フィールド、またはTを使ってシフト カウントを指定できます。

- 命令のオペランドの中で指定される4または5ビット イミディエイト値は、シフト カウント値を−16から15の範囲で表します。次はその一例です。

```
ADD    A,-4,B     ; Add accumulator A (right-shifted
                  ; 4 bits) to accumulator B
                  ; (one word, one cycle).
SFTL   A,+8       ; Shift (logical) accumulator A eight
                  ; bits left (one word, one cycle)
```

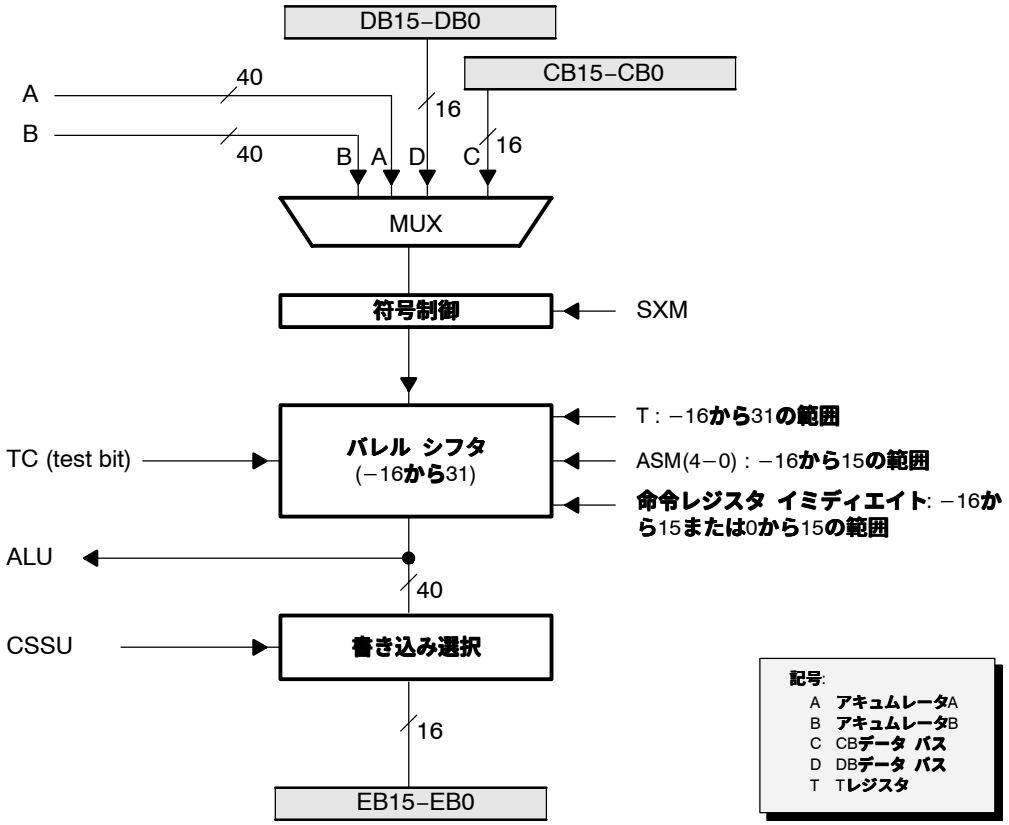
- ASM値は、シフト カウント値を−16から15の範囲で表し、LD命令(イミディエイト オペランドまたはデータ メモリ オペランドをともなう)によってロードできます。次はその一例です。

```
ADD    A, ASM, B    ; Add accumulator A to accumulator B
                    ; with a shift specified by ASM
```

❑ Tの6つのLSBは、シフト カウント値を−16から31の範囲で表します。次はその一例です。

```
NORM   A            ; Normalize accumulator A (T
                    ; contains the exponent value)
```

図4−7. バレル シフタ機能図



4.5 積和演算ユニット

’54xのCPUには、17ビット×17ビットのハードウェア乗算器があり、40ビットの専用加算器と組み合わされています。この積和演算ユニットは、myltiply/accumulate(MAC)機能を1パイプライン フェーズ サイクルで実現します。4−22ページの図4−8には、積和演算ユニットが示されています。

乗算器は、次のような制約をともなう符号付き、符号なし、符号付き/符号なし乗算を実行します。

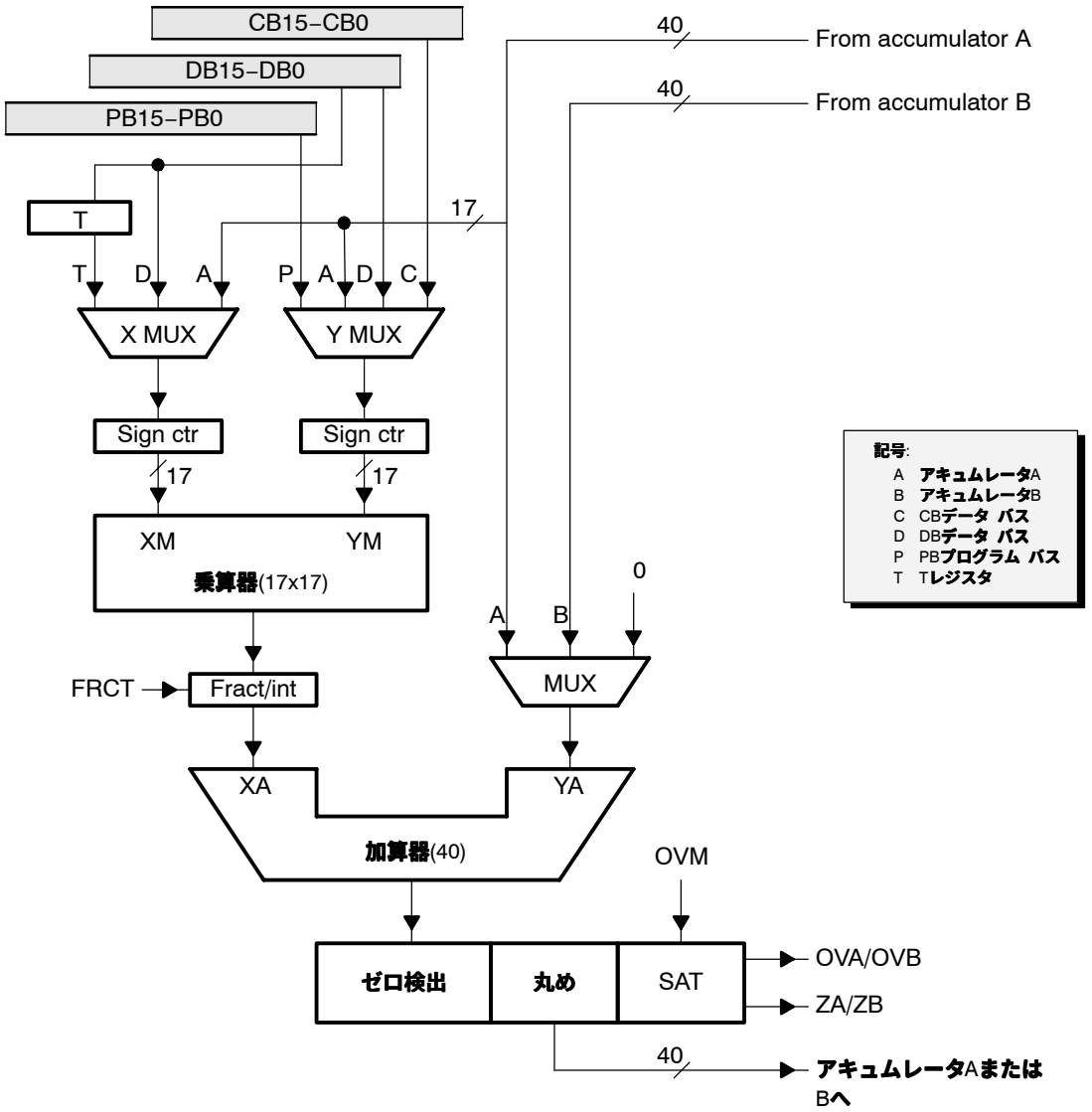
- ❑ 符号付き乗算では、各16ビット メモリ オペランドを、符号拡張をともなう17ビット ワードとして扱います。
- ❑ 符号なし乗算では、各入力オペランドで0をMSB(ビット16)に加えます。
- ❑ 符号付き/符号なし乗算では、オペランドのひとつは符号拡張され、別のオペランドがMSBで0とともに拡張されます(ゼロ フィル)。

乗算器出力を1ビットずつ左にシフトして、小数モードで2つの16ビットの2の補数を掛けて発生する冗長符号ビットを補正できます(小数モードはST1でFRCTビット＝1のとき選択されます)。

積和演算ユニットの加算器には、ゼロ検出器、ラウンダ(2の補数)、オーバーフロー/飽和ロジックがあります。丸めは、2^{15}を結果に加えて行き先アキュムレータの下位16ビットをクリアすることで実現されます。丸めは、接尾語Rがつく命令(一部除く)、一部の乗算、MAC、およびmultiply/substract(MAS)命令で実行されます。LMS命令も丸めを行って、更新される係数での量子化誤差を最小限にします。

加算器の入力は乗算器の出力から、およびアキュムレータのひとつから入力します。このユニットで乗算または積和演算処理が実行されると、その結果は行き先アキュムレータ(AまたはB)に転送されます。

図4−8. **積和演算機能図**



4.5.1 乗算器入力ソース

このサブセクションでは、乗算器入力のソースを示して、乗算器入力が各種命令に合わせてどのように選択されるかを説明します。

乗算器へのXM入力ソースは、次の値のどれもが当てはまります。

- ❑ **Tレジスタ**
- ❑ **データ バスDBからのデータ メモリ オペランド**
- ❑ **アキュムレータAビット**32−16

乗算器へのYM入力ソースは、次の値のどれもが当てはまります。

- ❑ データ バスDBからのデータ メモリ オペランド
- ❑ データ バスCBからのデータ メモリ オペランド
- ❑ プログラム バスPBからのプログラム メモリ オペランド
- ❑ アキュムレータAビット32−16

表4−5は、乗算器入力が、各種命令に合わせてどのように獲得されるかを示しています。実際に使用される乗算器入力の組み合わせには、合計9種類があります。

Tをひとつの入力として使用する命令では、2番目の入力はイミディエイト値として獲得されるか、またはデータ メモリからデータ バス(DB)を経て獲得されるか、またはアキュムレータAから獲得されます。

シングル データ メモリ オペランド アドレッシングを使用する命令では、ひとつのオペランドがDBを経由して乗算器に送られます。2番目のオペランドは、イミディエイト値としてTレジスタから入力されるか、またはPB経由でプログラム メモリから、またはアキュムレータAから入力されます。

デュアル データ メモリ オペランド アドレッシングを使用する命令では、DBおよびCBがデータを乗算器に転送します。

最後の2つのケースは、FIRS命令、SQURおよびSQDST命令で使用されます。
FIRS命令は、入力をPBおよびアキュムレータAから獲得します。SQURおよびSQDSTは、両方の入力をアキュムレータAから獲得します。

表4−5. 各種命令の乗算器入力選択

			Xマルチプレクサ			Yマルチプレクサ			
ケース	入力タイプ		T	DB	A	PB	CB	DB	A
1	MPY	#1234h, A	√					√	
2	MPY[R]	*AR2, A	√					√	
3	MPYA	B	√						√
4	MACP	*AR2, pmad, A		√		√			
5	MPY	*AR2, *AR3, B		√			√		
6	SQUR	*AR2, B		√				√	
7	MPYA	*AR2		√					√
8	FIRS	*AR2, *AR3, pmad			√	√			
9	SQUR	A, B			√				√

Tは、ひとつのオペランドを乗算およびmultiply/accumulate命令のために提供します。もう1つのメモリ オペランドは、シングル データ メモリ オペランドです。Tは、LD||MAC、LD||MAS、ST||MAC、ST||MAS、ST||MPYなどの、並列ロードまたは並列ストアをともなう乗算命令にもオペランドを提供します。Tは、メモリ マップ レジスタ アドレッシング モードをサポートする命令でオペランドとして明記してロードでき、または乗算処理の間にオペランドとして明記せずにロードできます。

ビットA(32−16)は乗算器の入力にもなるので、ある演算結果をメモリにストアしてその結果を乗算器に送り込むことが必要なシーケンスをより高速にすることもできます。特定用途命令の一部(FIRS、SQDST、ABDST、POLYなど)では、アキュムレータAの内容をALUによって処理することができ、それを乗算器にオーバーヘッドなしに入力できます。

4.5.2 Multiply/Accumulate(MAC)命令

MAC命令は乗算器の演算バンド幅を使って、2つのオペランドを同時に処理します。積和演算ユニットによって、複数の算術演算を1サイクルで処理することができます。

デュアル データ メモリ オペランド アドレッシングをともなうMAC、MAS、MACSU命令では、各サイクルでデータを乗算器にCB、DB経由で転送して、1サイクルで乗算、加算できます。これらのオペランドのためのデータ アドレスは、ARAU0およびARAU1(補助レジスタ算術ユニット)によって生成されます。ARAU0およびARAU1について、詳しくは5−11ページ、サブセクション5.5.2 ARAUおよびアドレス生成処理を参照してください。

MACDおよびMACP命令では、データを各サイクルでDBおよびPB経由で乗算器に転送できます。DBは、データをデータ メモリから取り出して、PBは係数をプログラム メモリから取り出します。MACDおよびMACP処理がリピート命令(RPTおよびRPTZ)とともに使われる場合、1サイクルMAC処理をデータおよび係数の連続アクセスをともなって実行します。データ アドレスはARAU0およびプログラム アドレス レジスタ(PAR)によって生成されます。データ メモリ アドレスは間接アドレッシング モードのシングル データ メモリ オペランドに応じて、ARAUによって更新されます。プログラム メモリ アドレスはPAGENによってインクリメントされます。

リピートされるMACD命令は、フィルタリング構造をサポートします(重み付き平均)。積和が実行される間に、サンプル データがメモリにシフトされて次のサンプルのための空きを確保し、最も古いサンプルを破棄します。サーキュラ アドレッシングをともなうMACおよびMACP命令は、フィルタの実現をサポートできます。FIRS命令は、サーキュラ アドレッシングが使用される際に、FIRフィルタのための効率的な対称構造を実装します。

MPYUおよびMACSU命令は、拡張精度算術演算を容易にします。MPYU命令は、符号なし乗算を実行します。Tの符号なしの内容は、アドレスされたデータ メモリの符号なしの内容と掛け合わされ、その結果が指定のアキュムレータに置かれます。MACSU命令は、符号付き/符号なし乗算および加算を実行します。データ メモリのある符号なしの内容は、データ メモリの別の符号付きの内容と掛け合わされて、その結果がアキュムレータに加えられます。この処理により、16ビットより大きなオペランドを16ビット ワードに分割して、個別に処理して32ビットより大きな積を生成できます。

二乗/加算(SQURA)および二乗/減算(SQURS)命令は、同じデータ値を乗算器の両方の入力に渡してその値を二乗します。その結果は、加算器の段階において、アキュムレータに加えられる(SQURA)か、またはアキュムレータから引かれ(SQURS)ます。SQUR命令は、データ メモリの値またはアキュムレータAの内容を二乗します。

4.5.3 乗算におけるMACおよびMAS飽和(対象デバイスは付録Dを参照)

乗算における飽和がセットされる場合(SMUL=1)、OVM=1のときMAC命令はMPY + ADDと同等になります。この場合、乗算のあとにつづく加算(MAC)または減算(MAS)の実行の前に、乗算、8000h×8000hが小数モードで7FFF FFFFhに飽和されます。

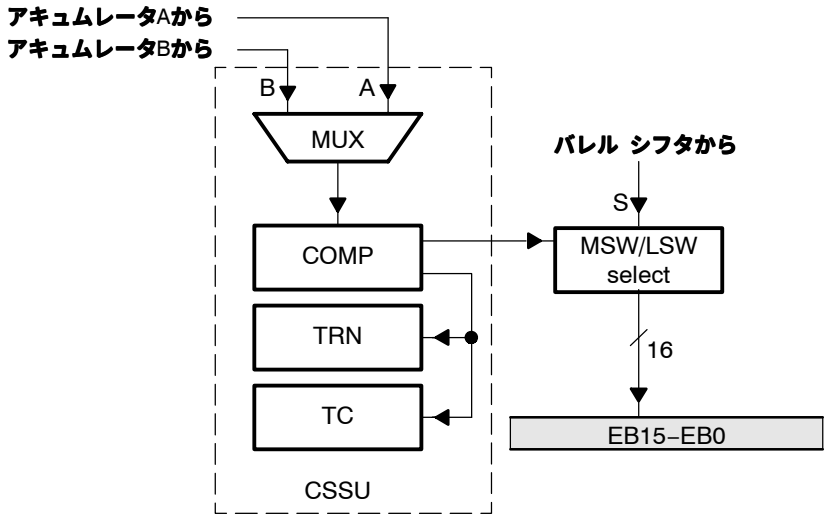
乗算における飽和がセットされない場合(SMUL=0)、MACおよびMASの最終結果のみが飽和されます。

OVM=1およびFRCT=1のとき、PMSTのSMULビットは、アキュムレーションがMACおよびMAS命令で実行される前に乗算結果が飽和されるかどうかを決めます。この機能により、MACおよびMAS処理をETSI GSM仕様(GSM仕様6.06、6.10、6.53)で定義されるMACおよびMASの基本処理に準拠させることができます。

4.6 比較選択ストア ユニット(CSSU)

比較選択ストア ユニット(CSSU)は、ビタビ オペレータの加算/比較/選択(ACS)処理専用の特定用途ハードウェア ユニットです。図4−9はCSSUを示したものです。CSSUは、ALUとともに使用して高速ACS演算を実行します。

図4−9. 比較選択ストア ユニット(CSSU)

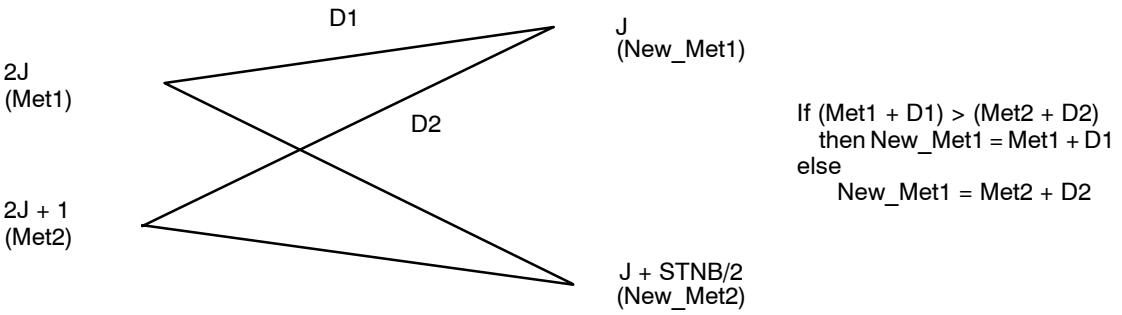


CSSUにより、イコライザやチャネル デコーダで使用される様々なビタビ バタフライ アルゴリズムを'54xでサポートできます。

ビタビ オペレータの加算機能(図4−10)は、ALUによって実行されます。この機能は、デュアル加算機能(Met1 ± D1およびMet2 ± D2)から構成されます。デュアル加算は、ST1のC16ビットをセットすることでALUがデュアル16ビット モードに設定されている場合、1マシン サイクルで完了します。ALUがデュアル16ビット モードに設定されているとき、すべてのロング ワード(32ビット)命令は、デュアル16ビット算術命令になります。

TはALUの入力(デュアル16ビット オペランドとして)に接続され、ローカル記憶として使用してメモリ アクセスを最小限にします。表4−6は、デュアル16ビットALU演算を実行する命令を示しています。

図4−10. **ビタビ オペレータ**



記号: STNB **ステート数**
Met **パス メトリクス**
D **分岐メトリクス**

表4−6. **デュアル**16**ビット モードの**ALU**演算**

命令		機能(デュアル16ビット モード)
DADD	Lmem, src [, dst]	src(31−16) + Lmem(31−16) → dst(39−16) src(15−0) + Lmem(15−0) → dst(15−0)
DADST	Lmem, dst	Lmem(31−16) + T → dst(39−16) Lmem(15−0) − T → dst(15−0)
DRSUB	Lmem, src	Lmem(31−16) − src(31−16) → src(39−16) Lmem(15−0) − src(15−0) → src(15−0)
DSADT	Lmem, dst	Lmem(31−16) − T → dst(39−16) Lmem(15−0) + T → dst(15−0)
DSUB	Lmem, src	src(31−16) − Lmem(31−16) → src(39−16) src(15−0) − Lmem(15−0) → src(15−0)
DSUBT	Lmem, dst	Lmem(31−16) − T → dst(39−16) Lmem(15−0) − T → dst(15−0)

記号:
→ **ストア先**
Lmem **ロング**(32**ビット**)**データ メモリ値**
src **ソース アキュムレータ**(A**または**B)
dst **行き先アキュムレータ**(A**または**B)
x(n−m) x**の第**m**ビットから第**n**ビットまで**

CSSUは、比較選択の演算を、CMPS命令、コンパレータ、16ビット トランジション レジスタ(TRN)によって実現します。この演算により、指定されたアキュムレータの2つの16ビット部分を比較してその決定をTRNのビット0にシフトします。またこの決定は、ST0のTCビットに記憶されます。この決定を基に、対応するアキュムレータの16ビット部分がデータ メモリに記憶されます。例4−4は、CMPS命令が実行する比較選択演算を示しています。

例4−4. CMPS命令の演算

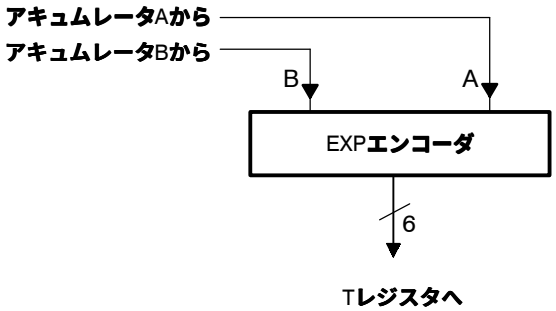
```
CMPS   B,*AR3    ;if (B(31-16)>B(15-0)) then
                 ;B(31-16)->(*AR3); TRN<<1; 0->TRN(0);
                 ;0->TC}
                 ;else B(15-0)->(*AR3); TRN<<1;
                 ;1->TRN(0); 1->TC;
```

TRNは、遷移決定の新たなステートへのパスについての情報を含んでいます。この情報は、バック トラッキング ルーチンに使用されます。このルーチンは、最適なパスを検出してこれによりコードをデコードします。

4.7 べき指数エンコーダ

べき指数エンコーダは、特定用途ハードウェア デバイスで、EXP命令を1サイクルでサポートするための専用のハードウェアです(図4−11参照)。EXP命令により、アキュムレータのべき指数の値をTに−8から31の範囲の2の補数の値としてストアできます。べき指数は、上位の冗長ビットの数−8として定義され、非有効符号ビットを解消するためにアキュムレータに必要なシフト数と一致します。この演算は、アキュムレータの値が32ビットを超える場合、負の値になります。

図4−11. べき指数エンコーダ



EXPおよびNORM命令は、べき指数エンコーダを使ってアキュムレータの内容を効率的に正規化します。NORMは、Tで指定されたビット数によりアキュムレータの値を1サイクルでシフトするのをサポートします。Tの負の値はアキュムレータ内容の右シフトを指定して、アキュムレータの32ビットを超えるあらゆる値を正規化します。例4−5は、アキュムレータAの正規化を示しています。

例4−5. アキュムレータAの正規化

```
;Normalize accumulator A
  EXP   A            ; (the number of leading bits • 8)-> T.
  ST    T, EXPONENT  ; Store the exponent (T) into data
                     ; memory
  NORM  A            ; Normalize accumulator A, (A)<<(T)
```

第5章

データ アドレッシング

’54xには、7種類の基本的なアドレッシング モードがあります。

- ❑ **イミディエイト アドレッシング。命令からの固定値をエンコードします。**
- ❑ **絶対アドレッシング。命令からの固定アドレスをエンコードします。**
- ❑ **アキュムレータ アドレッシング。アキュムレータを使ってプログラム メモリにデータとしてアクセスします。**
- ❑ **直接アドレッシング。命令の7ビットを使ってDPまたはSPに相対するオフセットをエンコードします。オフセットとDPまたはSPで、データ メモリの実際のアドレスを決めます。**
- ❑ **間接アドレッシング。補助レジスタを使ってメモリにアクセスします。**
- ❑ **メモリ マップド レジスタ アドレッシング。現在のDP値または現在のSP値に影響を与えずに、メモリ マップド レジスタをアクセスします。**
- ❑ **スタック アドレッシング。システム スタックに値を追加したり取り除く管理をします。**

Topic **Page**

5.1 イミディエイト アドレッシング

イミディエイト アドレッシングでは、命令構文はオペランドに特定の値を含みます。2種類の値をひとつの命令でエンコードできます。

- ショート イミディエイト値は、3、5、8、または9ビット長の場合があります。
- 16ビット イミディエイト値は、常に16ビット長になります。

イミディエイト値は、1ワードまたは2ワード命令でエンコードされます。3、5、8、9ビット値は、1ワード命令にエンコードされ、16ビット値は2ワード命令にエンコードされます。

ひとつの命令にエンコードされるイミディエイト値の長さは、使用される命令のタイプに応じます。表5−1は'54xの命令を示していますが、これらの命令はイミディエイト値をそれぞれの命令ワードにエンコードできます。

表5−1. イミディエイト アドレッシングが可能な命令

3および5ビット定数	8ビット定数	9ビット定数	16ビット定数	
LD	FRAME	LD	ADD	ORM
	LD		ADDM	RPT
	RPT		AND	RPTZ
			ANDM	ST
			BITF	STM
			CMPM	SUB
			LD	XOR
			MAC	XORM
			OR	

イミディエイト アドレッシングのための構文は、シャープ記号(#)を値または記号の直前につけて、それがイミディエイト値であることを示します。たとえば、アキュムレータAに16進数の80をロードするには、次のように表記します。

```
LD #80h, A
```

図5−1および図5−2はRPT命令を使い、イミディエイト値がイミディエイト アドレッシングを使う命令でどのようにエンコードされるかを示しています。命令の中でオペコードは、上位半分でエンコードされます(1ワード エンコーディングの場合、ビット8−15、2ワード エンコーディングの場合、上位ワードのビット0−15)。定数はその他の空間にあります。

図5−1. ショート イミディエイト アドレッシングをともなうRPT命令

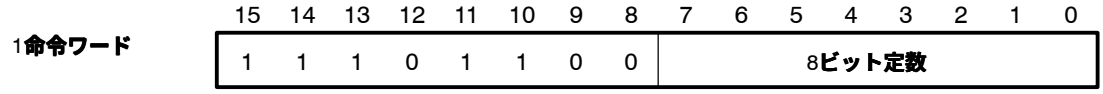
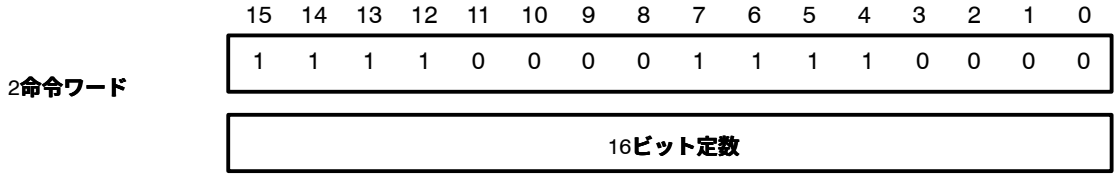


図5−2. 16ビット イミディエイト アドレッシングをともなうRPT命令



5

5.2 絶対アドレッシング

絶対アドレッシングには4つのタイプがあります。

- ❑ dmad**アドレッシング**:
  - MVDK Smem, dmad
  - MVDM dmad, MMR
  - MVKD dmad, Smem
  - MVMD MMR, dmad
- ❑ pmad**アドレッシング**:
  - FIRS Xmem, Ymem, pmad
  - MACD Smem, pmad, src
  - MACP Smem, pmad, src
  - MVDP Smem, pmad
  - MVPD pmad, Smem
- ❑ PA**アドレッシング**:
  - PORTR PA, Smem
  - PORTW Smem, PA
- ❑ *(lk)**アドレッシングは、シングル データ メモリ**(Smem)**オペランドをサポートするすべての命令で使用されます。**

絶対アドレスは、常に16**ビットの長さでエンコードされるので、絶対アドレスをエンコードする命令は常に最小**2**ワード長になります。**

5.2.1 dmadアドレッシング

データ メモリ アドレス(dmad)**アドレッシングは、特定の値を使ってデータ メモリ空間のアドレスを指定します。**

dmad**アドレッシングのための構文は、記号または数値を使ってデータ メモリ空間のアドレスを指定します。たとえば、データ メモリ空間中の**SAMPLE**とラベルのついたアドレスに含まれる値を、**AR5**で指し示されたデータ メモリ空間のメモリ位置にコピーするには、次のように記述します。**

```
MVKD   SAMPLE, *AR5
```

この例では、SAMPLE**が参照するアドレスは**dmad**値になります。**

5.2.2 pmadアドレッシング

プログラム メモリ アドレス(pmad)アドレッシングは、特定の値を使ってプログラム メモリ空間のアドレスを指定します。

pmadアドレッシングのための構文は、記号または数値を使ってプログラム メモリ空間のアドレスを指定します。たとえば、TABLEとラベルの付いたプログラム メモリ位置のワードを、AR7によって指定されるデータ メモリ位置にコピーするには、次のように記述します。

```
MVPD   TABLE, *AR7+
```

この例では、TABLEが参照するアドレスはpmad値になります。

5.2.3 PAアドレッシング

ポート アドレス(PA)アドレッシングは、特定の値を使って外部I/Oポート アドレスを指定します。

PAアドレッシングのための構文は、記号または数値を使ってポート アドレスを指定します。たとえば、ポート アドレスFIFOのI/OポートからAR5で指し示されたデータ メモリ位置に値をコピーするには、次のように記述します。

```
PORTR  FIFO, *AR5
```

この例では、FIFOはポート アドレスを意味します。

5.2.4 *(lk)アドレッシング

***(lk)アドレッシングは、特定の値を使ってデータ メモリ空間のアドレスを指定します。**

***(lk)アドレッシングのための構文は、記号または数値を使ってデータ メモリ空間のアドレスを指定します。たとえば、アキュムレータAをデータ メモリ空間にあるアドレスBUFFERに含まれる値をロードするには、次のように記述します。**

```
LD  *(BUFFER),A
```

***(lk)アドレッシングのための構文により、Smemアドレッシングを使うすべての命令が、DPを変更せずARの初期化も行わずに、データ メモリ空間のあらゆる位置にアクセスできます。この形式の絶対アドレッシングを使用する場合、命令の長さは1ワードずつ拡張されます。たとえば、1ワード命令は2ワード命令になり、2ワード命令は3ワード命令になります。命令に1ワード加えることで、パイプラインのディレイ スロットに影響があります。**

> **注:**
>
> ***(lk)形式の絶対アドレッシングを使用する命令は、単一リピート命令(RPT、RPTZ)とともに使用できません。**

5.3 アキュムレータ アドレッシング

アキュムレータ アドレッシングは、アキュムレータにある値をアドレスとして使用します。このアドレッシング モードは、プログラム メモリにデータとしてアドレッシングするのに使用します。

次の2つの命令により、アキュムレータをアドレスとして使用できます。

- READA Smem
- WRITA Smem

READAは、アキュムレータAで指定されるプログラム メモリから、命令のシングル データ メモリ(Smem)オペランドで指定されるデータ メモリにワードを転送します。

WRITAは、命令のSmemオペランドで指定されるデータ メモリから、アキュムレータAで指定されるプログラム メモリにワードを転送します。

リピート モードでは、アキュムレータAをインクリメントすることができます。

> 注 :
>
> ほとんどの'54xデバイスでは、プログラム メモリはアキュムレータAの下位16ビットによって指定されます。しかし、'548と'549には23本のアドレス ラインがあることから、'548と'549のプログラム メモリはアキュムレータAの下位23ビットによって指定されます。詳しくは、3−8ページ、サブセクション3.1.1拡張プログラム メモリを参照してください。

5.4 直接アドレッシング

直接アドレッシング モードでは、命令はデータ メモリ アドレス(dma)の下位7ビットを含んでいます。この7ビットdmaはアドレス オフセットで、ベース アドレス、データ ページ ポインタ(DP)、またはスタック ポインタ(SP)と組み合わされ、16ビット データ メモリ アドレスを形成します。この形式のアドレッシングを使って、DPまたはSPを変更せずに、128ワード内のどこにでもランダムにアクセスできます。

注:

直接アドレッシングは、オフセット アドレッシングの唯一の方法ではありません。しかし、このモードの長所は、各命令をエンコードしてシングル ワードでアドレッシングできる点にあります。

DPまたはSPのいずれかをdmaオフセットと組み合わせて、実際のアドレスを形成できます。ステータス レジスタST1内にあるコンパイラ モード ビット(CPL)は、アドレスを発生するのに使用する方法を選択します。

- CPL=0の場合、dmaフィールドは9ビットDPフィールドに連結されて16ビット データ メモリ アドレスを形成します。
- CPL=1の場合、dmaフィールドはSPに加えられ(正のオフセット)、16ビット データ メモリ アドレスを形成します。

直接アドレッシングの構文は、記号または数値を使用してオフセット値を指定します。たとえば、メモリ位置SAMPLEの内容をアキュムレータBに加えるには、正しいベース アドレスがDP(CPL=0)またはSP(CPL=1)にある場合、次のように記述します。

```
ADD    SAMPLE, B
```

SAMPLEの下位7ビット アドレスは、命令ワードに記憶されます。

図5−3は、直接アドレッシングを使用する命令のオペコード形式を示しています。表5−2は、直接アドレッシング命令のビットを表しています。図5−4は、16ビット データ アドレスがどのように形成されるかを示しています。

図5−3. 直接アドレッシング命令形式

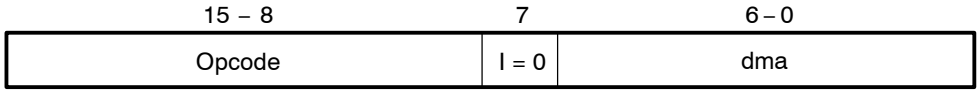
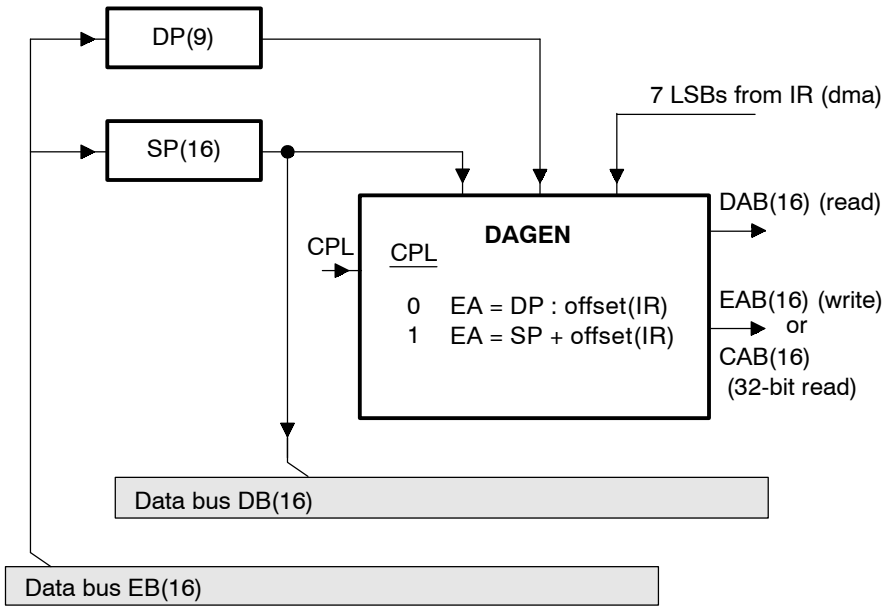


表5−2. 直接アドレッシング命令ビット概要

ビット	名称	機能
15 − 8	Opcode	この8ビット フィールドは、命令の演算コードを含んでいます。
7	I	I=0のとき、命令が使用するアドレッシング モードは直接アドレッシング モードです。
6−0	dma	この7ビット フィールドは、命令のデータ メモリ アドレス オフセットを含んでいます。

図5−4. 直接アドレッシングのブロック図

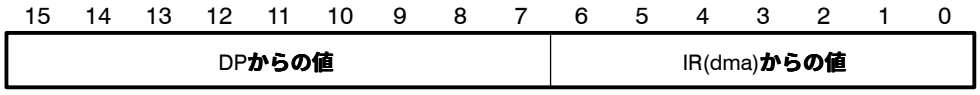


Legend: EA Effective address
IR Instruction register

5.4.1 DPリファレンスド直接アドレッシング

DPリファレンスド直接アドレッシングでは、命令レジスタの7ビットのdmaが9ビットDPに連結されてアドレスを形成します。図5−5は、2つの値がどのようにアドレスを形成するかを示しています。

図5−5. DPリファレンスド直接アドレッシング



DPリファレンスド直接アドレッシングはメモリを512ページに分割しますが、これはDPの領域が0から511(2^9-1)だからです。各ページには128のアドレッシング可能なワードがありますが、これはそのdma領域が0から127(2^7-1)であるためです。

つまり、DPは512通りの128ワード データ メモリ ページの1つを指します。dmaは、そのページ内の特定の場所を指します。ページ1上の位置0にアクセスするのとページ2上の位置0にアクセスすることの違いは、DPの値だけです。

DPはLD命令によってロードされます。

5.4.2 SPリファレンスド直接アドレッシング

SPリファレンスド直接アドレッシングでは、命令レジスタの7ビットdmaが正のオフセットとしてSPに加えられ、有効な16ビット データ メモリ アドレスを形成します。図5−6は、2つの値がどのように組み合わさって、アドレスを形成するかを示しています。

図5−6. SPリファレンスド直接アドレッシング

SPは、メモリのあらゆるアドレスを指します。dmaはページの特定の位置を指して、これによりメモリの連続する128ワード(2^7-1)ブロックにあらゆるベース アドレスからアクセスできるようになります。

SPは、スタックに値を加えたりスタックから取り除いたりします。詳しくは、セクション5.7スタック アドレッシングを参照してください。

5.5 間接アドレッシング

間接アドレッシングでは、補助レジスタに含まれる16ビット アドレスを通して、64Kワードのデータ メモリ空間のあらゆる場所にアクセスできます。'54xには8つの16ビット補助レジスタがあります(AR0−AR7)。間接アドレッシングは、主に固定されたサイズのステップで連続するメモリをアドレッシングする必要があるとき使用されます。

メモリが間接アドレッシングによってアドレッシングされる場合、補助レジスタおよびアドレスはオプションでデクリメント、インクリメント、オフセット、またはインデックスによって修飾できます。特別なモードには、サーキュラおよびビット リバース アドレッシングがあります。サーキュラ バッファ サイズ レジスタ(BK)は、サーキュラ アドレッシングによって使用されます。AR0レジスタは、インデックスおよびビット リバース アドレッシング モードに使用する他、他の補助レジスタのようにメモリを指すためにも使用します。

間接アドレッシングは、シングル16ビット データ オペランドをメモリから1命令でリード/ライトするのに十分な柔軟性があるばかりでなく、2つのデータ メモリ位置に1命令でアクセスするのにも十分対応します。2つのデータ メモリへのアクセスには、独立した2つのメモリからのリード、2つの連続するメモリへのリード/ライト、1つのメモリへのライトおよび1つのメモリからのリードがあります。

5.5.1 シングル オペランド アドレッシング

図5−7は、シングル データ メモリ(Smem)オペランドのための、間接アドレッシング命令形式を示しています。表5−3は、命令のビットを説明しています。

図5−7. シングル データ メモリ オペランドのための間接アドレッシング命令形式

15–8	7	6–3	2–0
Opcode	I = 1	MOD	ARF

表5−3. 間接アドレッシング命令ビット概要—シングル データ メモリ オペランド

ビット	名称	機能
15 − 8	Opcode	この8ビット フィールドには、命令の演算コードが含まれます。
7	I	I=1とのき、命令が使用するアドレッシング モードは間接アドレッシング モードです。
6−3	MOD	この4ビット修飾フィールドは、間接アドレッシングのタイプを定義します。MODフィールドでアドレッシング タイプを指定する16の方法については、ページ5−13、サブセクション5.5.3命令が使用するアドレスの修飾を参照してください。

表5−3. 間接アドレッシング命令ビット概要—シングル データ メモリ オペランド (続き)

ビット	名称	機能
2−0	ARF	この3ビット補助レジスタ フィールドは、アドレッシングで使用する補助レジスタを定義します。ARFは、ステータス レジスタST1の互換性モード ビット(CMPT)に依存します。
		CMPT = 0　標準モード。標準モードでは、ARPの値にかかわらずARFは常に補助レジスタを指定します。ARPは更新されません。ARPは、DSPがこのモードのとき、常にゼロに設定されます。
		CMPT = 1　互換モード。互換モードでは、ARF=0のときARPは補助レジスタを選択します。そうでない場合は、ARFは補助レジスタを選択して、アクセスが完了するとARF値はARPにロードされます。アセンブリ命令の*AR0は、互換モードでARPにより選択された補助レジスタを示します。

注:

場合によっては、2つのデータ オペランドを同時にフェッチできます。この場合、異なる命令形式が必要です。この形式については、5−19ページ、サブセクション5.5.4デュアル オペランド アドレス修飾を参照してください。

5.5.2 ARAUおよびアドレス生成

2つの補助レジスタ算術ユニット(ARAU0およびARAU1)は、補助レジスタの内容を使用します。ARAUは、符号なし、16ビット補助レジスタ算術演算を実行します。この補助レジスタを事前に修飾するアドレッシングもあります。

補助レジスタを次のように処理できます。

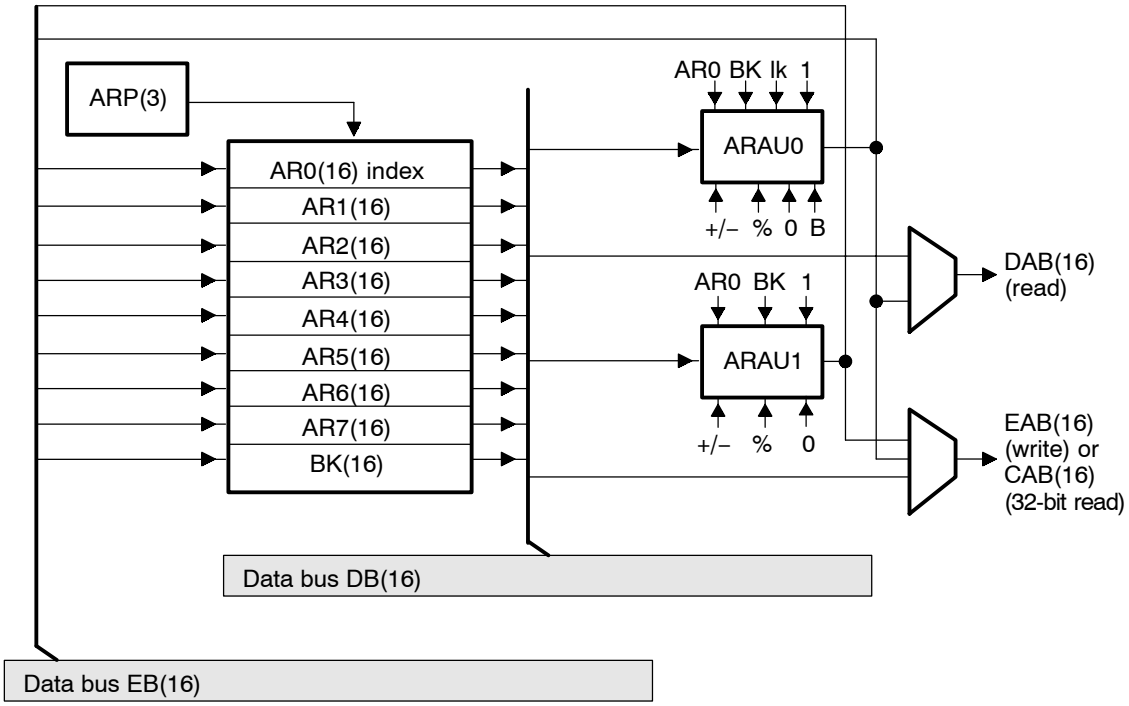
- STM命令を使ってイミディエイト値をロードできます。
- メモリ マップド補助レジスタへの書き込みによって、データ バス経由でロードできます。
- 間接アドレッシングをサポートする命令の間接アドレッシング フィールドによって修飾できます。
- 補助レジスタ修飾(MAR)命令によって修飾できます。
- BANZ[D]命令を使ってループ カウンタとして使用できます。

注 :

一般にSTMまたはMVDKは、補助レジスタをロードするのに使用します。これらの命令双方により、次の命令が新しい値をレジスタで使えるようになります。新たな値をARにロードする他の命令は、パイプライン レイテンシ(遅延)を発生します。パイプラインおよび考えられるパイプラインの衝突については、第7章パイプラインを参照してください。

図5−8はARAUを示しており、これらのARAUは、シングル データ メモリ オペランドを使って間接アドレッシング モードでアドレスを生成するために使用されます。図のように、間接アドレッシングにおけるアドレス生成では、主に補助レジスタ算術ユニット(ARAU0およびARAU1)および補助レジスタ(AR0−AR7)が使用されます。

図5−8. シングル データ メモリ オペランドのための間接アドレッシング ブロック図



5.5.3 シングル オペランド アドレス修飾

命令で使用するアドレスをそれらにアクセスする前後に修飾することができ、またそのままにもできます。アドレスを修飾するには、アドレスを1ずつインクリメントまたはデクリメントするか、16ビット オフセットを加えるか、またはAR0の値で指示します。これら3タイプの処理は、アクセスの前または後のいずれに行う方法、アドレスを変更しない方法とを組み合わせて、全部で16のアドレッシング タイプになります。それぞれが、間接アドレッシングを使う命令のエンコーディングを決める4ビット修飾フィールドのMOD値に割り当てられています。

表5−4には、オペランド構文および各アドレッシング タイプの説明があります。

表5−4. シングル データ メモリ オペランドをともなう間接アドレシング タイプ

MOD フィールド	オペランド構文	ファンクション	説明†
0000 (0)	*ARx	addr = ARx	ARxにはデータ メモリ アドレスを含みます。
0001 (1)	*ARx−	addr = ARx ARx = ARx − 1	アクセス後、ARxのアドレスがデクリメントされます。‡
0010 (2)	*ARx+	addr = ARx ARx = ARx + 1	アクセス後、ARxのアドレスがインクリメントされます。‡
0011 (3)	*+ARx	addr = ARx + 1 ARx = ARx + 1	ARxのアドレスが使用される前にインクリメントされます。‡§#
0100 (4)	*ARx−0B	addr = ARx ARx = B(ARx − AR0)	アクセス後、AR0がARxから逆キャリー伝播を使用して減算されます。
0101 (5)	*ARx−0	addr = ARx ARx = ARx − AR0	アクセス後、AR0がARxから減算されます。
0110 (6)	*ARx+0	addr = ARx ARx = ARx + AR0	アクセス後、AR0がARxに加算えられます。
0111 (7)	*ARx+0B	addr = ARx ARx = B(ARx + AR0)	アクセス後、AR0が逆キャリー伝播によってARxに加算されます。
1000 (8)	*ARx−%	addr = ARx ARx = circ(ARx − 1)	アクセス後、ARxのアドレスがサーキュラ アドレッシングによってデクリメントされます。‡
1001 (9)	*ARx−0%	addr = ARx ARx = circ(ARx − AR0)	アクセス後、AR0がサーキュラ アドレッシングによってARxから減算されます。

† ARxは、指定がない限りデータ メモリ アドレスとして使用されます。
‡ インクリメント/デクリメント値は、16ビットワード アクセスでは1、32ビットワード アクセスでは2になります。
§ このモードは、メモリ マップド レジスタ アドレッシングでは使えません。
¶ このモードの詳細については、5−5ページ、サブセクション5.2.4*(lk)アドレッシングを参照してください。
# このモードは、ライト アクセスのみで使用できます。

表5−4. シングル データ メモリ オペランドをともなう間接アドレシング タイプ(続き)

MOD フィールド	オペランド構文	ファンクション	説明†
1010 (10)	*ARx+%	addr = ARx ARx = circ(ARx + 1)	アクセス後、ARxのアドレスがサーキュラ アドレッシングによってインクリメントされます。‡
1011 (11)	*ARx+0%	addr = ARx ARx = circ(ARx + AR0)	アクセス後、AR0がサーキュラ アドレッシングによってARxに加算されます。
1100 (12)	*ARx(lk)	addr = ARx + lk ARx = ARx	ARxの和および16ビット ロング オフセット(lk)は、データ メモリ アドレスとして使用されます。ARxは更新されません。
1101 (13)	*+ARx(lk)	addr = ARx + lk ARx = ARx + lk	ARxのアドレスは使用される前にインクリメントされ、符号付き16ビット ロング オフセット(lk)に加えられます。次にデータ メモリ アドレスとして使用されます。§
1110 (14)	*+ARx(lk)%	addr = circ(ARx + lk) ARx = circ(ARx + lk)	ARxのアドレスが使用される前にインクリメントされ、サーキュラ アドレッシングによって符号付き16ビット ロング オフセット(lk)に加えらます。次にデータ メモリ アドレスとして使用されます。§
1111 (15)	*(lk)	addr = lk	符号なし16ビット ロング オフセット(lk)は、データ メモリの絶対アドレスとして使用されます(絶対アドレッシング)。§¶

† ARxは、指定がない限りデータ メモリ アドレスとして使用されます。
‡ インクリメント/デクリメント値は、16ビットワード アクセスでは1、32ビットワード アクセスでは2になります。
§ このモードは、メモリ マップド レジスタ アドレッシングでは使えません。
¶ このモードの詳細については、5−5ページ、サブセクション5.2.4*(lk)アドレッシングを参照してください。
# このモードは、ライト アクセスのみで使用できます。

5.5.3.1 インクリメント/デクリメント アドレス修飾(MOD=0、1、2または3)

ARを使用する間、ARをその値のインクリメントまたはデクリメントによって修飾できます。

ARを修飾なし、1ずつポストデクリメント、1ずつポストインクリメント、1ずつプリインクリメントする構文は、表5−4に、それぞれMOD=0、1、2、3で示しています。

プリインクリメント(*+ARx)は、ライト時のオペランドにアクセスする命令でのみサポートされます。

5.5.3.2 オフセット アドレス修飾(MOD=12または13)

オフセット アドレッシングは、間接アドレッシングのひとつのタイプで、そこでは事前に決められたオフセット、ステップ サイズが補助レジスタの内容に加えられます。オフセット アドレッシングには、2つのオプションがあります。両方のケースとも、命令の一部である16ビット ロング オフセットが補助レジスタの値に加えられます。その結果を使って、データ メモリにアドレッシングします。最初のケースでは、補助レジスタは更新されません。2番目のケースでは、補助レジスタが新しいアドレスによって更新されます。

このタイプのアドレッシングは、あるアレイまたは構造体の特定エレメントにアクセスするのに役立ち、特に補助レジスタが更新されないときは有用です。補助レジスタが更新されるときは、このタイプのアドレッシングは特にアレイを固定サイズのステップでアドレッシングするときに便利です。

オフセット アドレッシングを使ってARを更新するかしないかや、ARのオフセット アドレッシングのための構文は、表5−4のMOD 12および13にそれぞれ示されます。

注 :

1) オフセット アドレッシングを使用する命令は、シングル リピート命令を使ってリピートできません。

2) 16ビット ワード オフセット(*+ARx(lk))によるプリモディファイは、追加サイクルが必要です。これは、オフセット用のワードが命令コードに追加され、その結果2ワードまたは3ワードになるからです。

5.5.3.3 インデックス付きアドレス修飾(MOD=5または6)

インデックス付きアドレッシングは間接アドレッシングのタイプのひとつで、そこではAR0の内容が他の補助レジスタARxに加えられるか、または他の補助レジスタARxから引かれます。インデックス付きアドレッシングはオフセット アドレッシングと異なり、インデックスまたはステップ サイズをコード実行中に動的に決めることができます。インデックスがコード実行中に決められるので、ステップ サイズを簡単に変更できます。インデックス付きアドレッシングは、オフセット アドレッシングより優れる点があります。それは、命令に付加ワードが不要なことです。

AR0をARxから引く構文、およびAR0をARxに加える構文は、表5−4のMOD=5および6でそれぞれ示されます。

5.5.3.4 サーキュラ アドレス修飾(MOD=8、9、10、11または14)

畳込み、相関、FIRフィルタなど多くのアルゴリズムは、メモリにサーキュラ バッファを必要とします。これらのアルゴリズムでは、サーキュラ バッファは最も新しいデータを含むスライド ウィンドウです。新しいデータが入ると、バッファの最も古いデータに上書きします。サーキュラ バッファの実現には、サーキュラ アドレッシングが必要です。

サーキュラ バッファ サイズ レジスタ(BK)は、サーキュラ バッファのサイズを指定します。サーキュラ バッファのサイズRは、必ずNビット境界から始まり(サーキュラ バッファのベース アドレスの下位Nビットは必ず0)、Nは最小の整数で2N>Rを満たします。値Rは、必ずBKにロードされます。たとえば、31ワードのサーキュラ バッファは必ず、下位5ビットが0(xxxx xxxx xxx0 0000_2)であるアドレスから始まり、値31が必ずBKにロードされます。もうひとつの例としては、32ワード サーキュラ バッファは必ず、下位6ビットが0(xxxx xxxx xx00 0000_2)であるアドレスから始まり、値32が必ずBKにロードされます。ただし、アプリケーションによってはビット リバース アドレッシングを使って2^Nバッファを2^N境界に置き、サーキュラ アドレッシングを実現することもできます。

サーキュラ バッファの実行ベース アドレス(EFB)は、ユーザ選択可能な補助レジスタ(ARx)の下位Nビットをゼロにすることで決定されます。サーキュラ バッファのバッファ アドレスの終わり(EOB)は、ARxの下位NビットをBKの下位Nビットと入れ替えることで決定されます。サーキュラ バッファのインデックスはARxの下位Nビットで、ステップは補助レジスタに加えられるか、引かれる量です。サーキュラ バッファを使うときは、次の3つの規則に従います。

- ❑ サーキュラ バッファの最初の(最下位の)アドレスを2^N境界に置きます。ここでは2^Nはサーキュラ バッファのサイズより大きくなります。
- ❑ サーキュラ バッファのサイズ以下のステップを使います。
- ❑ 最初にサーキュラ バッファがアドレッシングするとき、補助レジスタはサーキュラ バッファ以外のアドレスをアクセスしてはいけません。

サーキュラ アドレッシングのアルゴリズムは、次のようになります。

```
If 0 ≤ index + step < BK:
   index = index + step.
Else if index + step ≥ BK:
   index = index + step – BK.
Else if index + step < 0:
   index = index + step + BK.
```

サーキュラ アドレッシングをシングル データ メモリまたはデュアル データ メモリ オペランドに使用できます。BKレジスタがゼロのときは、サーキュラ アドレッシングはサーキュラ アドレス修飾なしになります。これは、デュアル オペランドがARx+0と同等のアドレス修飾を実行する必要があるとき役立ちます。

図5−9は、BKレジスタ、補助レジスタ(ARx)、サーキュラ バッファの最下位、サーキュラ バッファの最上位、サーキュラ バッファへのインデックスのそれぞれの関係を示しています。

図5−10は、サーキュラ バッファがどのように実現されるか、およびサーキュラ バッファで発生したアドレスとサーキュラ バッファ内のデータの関係を示しています。

図5−9. サーキュラ アドレッシング ブロック図

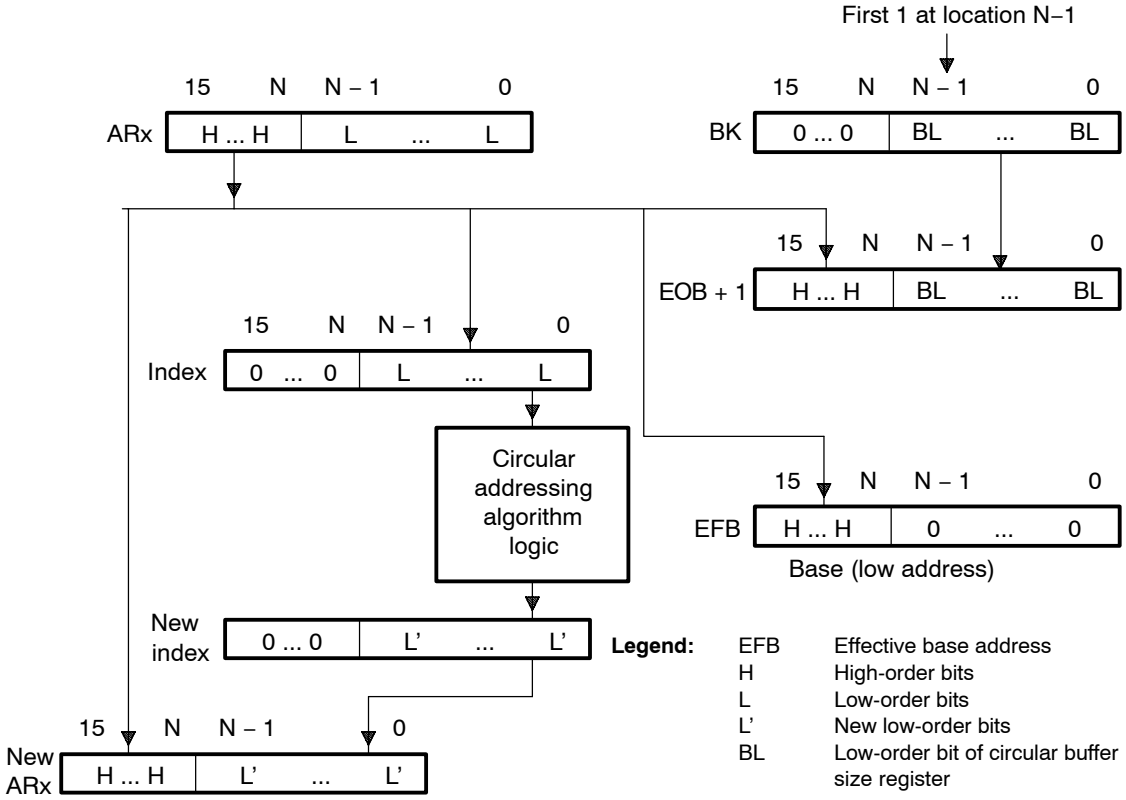
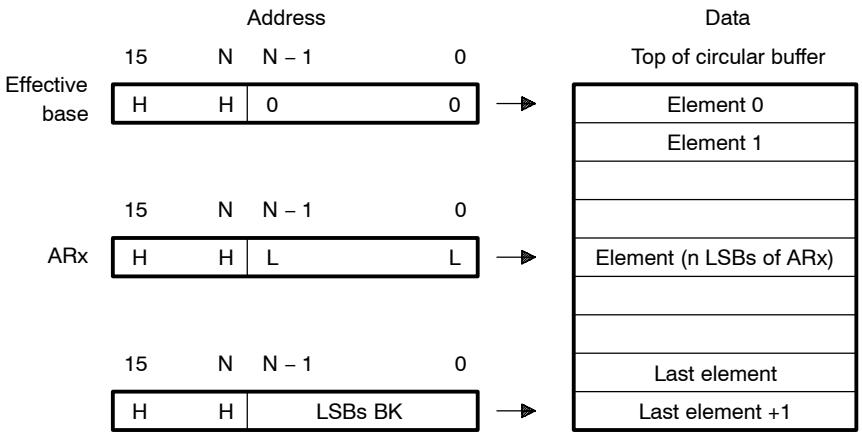


図5−10. サーキュラ バッファ実現



サーキュラ アドレッシングはその特徴として、1ずつのデクリメントまたはインクリメント(MOD=8および10)またはインデックスごとのデクリメントまたはインクリメント(MOD=9および11)を使用します。16ビット ワード オフセット(*+ARx(lk)%)によるプリモディファイは余分なコード ワードを必要とするので、命令コードは2または3ワードになります。最後のワードはオフセットです。間接オフセット アドレッシングを使う命令を、シングル リピート命令を使って繰り返すことはできません。

5タイプあるサーキュラ アドレッシングのそれぞれの構文は、表5−4にそれぞれMOD=8、9、10、11、14で示されます。

5.5.3.5 ビット リバース アドレス修飾(MOD=4または7)

ビット リバース アドレッシングは、様々な基数を使用するFFTアルゴリズムの高速実行およびプログラム メモリの低減に貢献します。このアドレッシング モードでは、AR0はFFTバッファ サイズの半分を指定します。AR0が持つ値は必ず2N −1と等しくなり(Nは整数)、FFTバッファ サイズは2Nとなります。補助レジスタはデータの絶対アドレスを指します。AR0をビット リバース アドレッシングを使って補助レジスタに加える場合、アドレスはビット リバースアドレッシングされ、通常の右から左でなく左から右にキャリーの伝播をします。

2つのビット リバース アドレッシング モードのそれぞれの構文は、表5−4にそれぞれMOD 4および7で示されています。

補助レジスタが8ビット長、AR2がメモリのデータのベース アドレスを表し($0110\ 0000_2$)、AR0に値00001000_2があるとする場合、例5−1は、AR2の逐次的な修飾およびAR2の結果の値を示します。

例5−1. ビット リバース アドレッシングにおける逐次的な補助レジスタ修飾

```
*AR2+0B   ;AR2   =  0110 0000    (0th value)
*AR2+0B   ,AR2   =  0110 1000    (1st value)
*AR2+0B   ;AR2   =  0110 0100    (2nd value)
*AR2+0B   ;AR2   =  0110 1100    (3rd value)
*AR2+0B   ;AR2   =  0110 0010    (4th value)
*AR2+0B   ;AR2   =  0110 1010    (5th value)
*AR2+0B   ;AR2   =  0110 0110    (6th value)
*AR2+0B   ;AR2   =  0110 1110    (7th value)
```

表5−5は、インデックス ステップのビット パターンの関係、およびAR2の4つのLSB(ビット リバース アドレッシングされる)を示しています。

ビット リバース アドレッシング モードのアプリケーションについては、TMS320C54x DSP Rererence Set, Volume4: Application Guidesを参照してください。

表5−5. ビット リバース アドレス

ステップ	ビット パターン	ビット リバース パターン	ビット リバース ステップ
0	0000	0000	0
1	0001	1000	8
2	0010	0100	4
3	0011	1100	12
4	0100	0010	2
5	0101	1010	10
6	0110	0110	6
7	0111	1110	14
8	1000	0001	1
9	1001	1001	9
10	1010	0101	5
11	1011	1101	13
12	1100	0011	3
13	1101	1011	11
14	1110	0111	7
15	1111	1111	15

5.5.4 デュアル オペランド アドレス修飾

デュアル データ メモリ オペランド アドレッシングは、デュアル リードまたは単一のリード パラレル ライト(2つの縦線 || で表される)を同時に実行する命令で使用されます。これらの命令は、すべて1ワード長で間接アドレッシング モードのみで実行します。2つのデータ メモリ オペランドは、XmemおよびYmemで表します。

- ❑ Xmemは、リード オペランドでDバス経由のアクセスをともないます。シフトをともなうSTHおよびSTLなどのストア命令は、オペランドをXmemで指定できます。
- ❑ Ymemは、デュアル リード(Cバス経由アクセス)をともなう命令でリード オペランドとして使用されるか、または並列ストア(Eバス経由アクセス)をともなう命令でライト オペランドとして使用されます。

並列ストアをともなう命令(ST||LDなど)で、ソース オペランドと格納先オペランドが同じメモリ位置を指している場合、ソースからの読み出しの後格納先へのライトが行なわれます。デュアル オペランド命令(ADDなど)が、2つのオペランドに対して同じ補助レジスタを指しており、かつそれぞれ異なるアドレッシング モードが指定されている場合、Xmodフィールドで定義されたモードがアドレッシングに使用されます。

図5−11は、デュアル データ メモリ オペランドのための間接アドレッシング命令のフォーマットを示します。表5−6は、命令のビットについて説明しています。

各補助レジスタをこのモードで選択するためには2ビットのみが使用可能なので、AR2−AR5の4つの補助レジスタのみが使用できます。表5−7は、どのXarまたはYar値がどの補助レジスタを選択するかを示しています。

図5−11. デュアル データ メモリ オペランドのための間接アドレッシング命令フォーマット

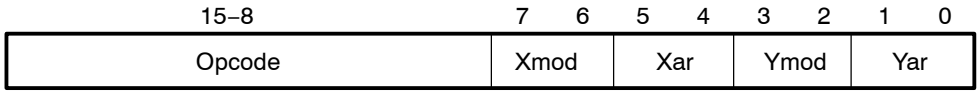


表5−6. 間接アドレッシング命令ビット概要—デュアル データ メモリ オペランド

ビット	名称	機能
15 − 8	Opcode	この8ビット フィールドには命令の演算コードがあります。
7−6	Xmod	この2ビット フィールドは、Xmemオペランドへのアクセスに使用する間接アドレッシング モードのタイプを指定します。
5−4	Xar	2ビットXmem補助レジスタ選択フィールドは、Xmemのアドレスを含む補助レジスタを指定します。
3−2	Ymod	この2ビット フィールドは、Ymemオペランドへのアクセスに使用する間接アドレッシング モードのタイプを指定します。
1−0	Yar	2ビットYmem補助レジスタ選択フィールドは、Ymemのアドレスを含む補助レジスタを指定します。

表5−7. 命令のXarおよびYarフィールドによって選択される補助レジスタ

Xarまたは Yarフィールド	補助レジスタ
00	AR2
01	AR3
10	AR4
11	AR5

図5−12は、デュアル データ メモリ オペランド アドレッシングを使ってどのようにアドレスが生成されるかを示しています。

デュアル データ メモリ オペランド アドレッシングは、4つの補助レジスタ(AR2−AR5)を使用します。これらのレジスタと共に2つのARAUによって、1サイクルで2つのオペランドにアクセスできます。

図5−12. デュアル データ メモリ オペランドの間接アドレッシング ブロック図

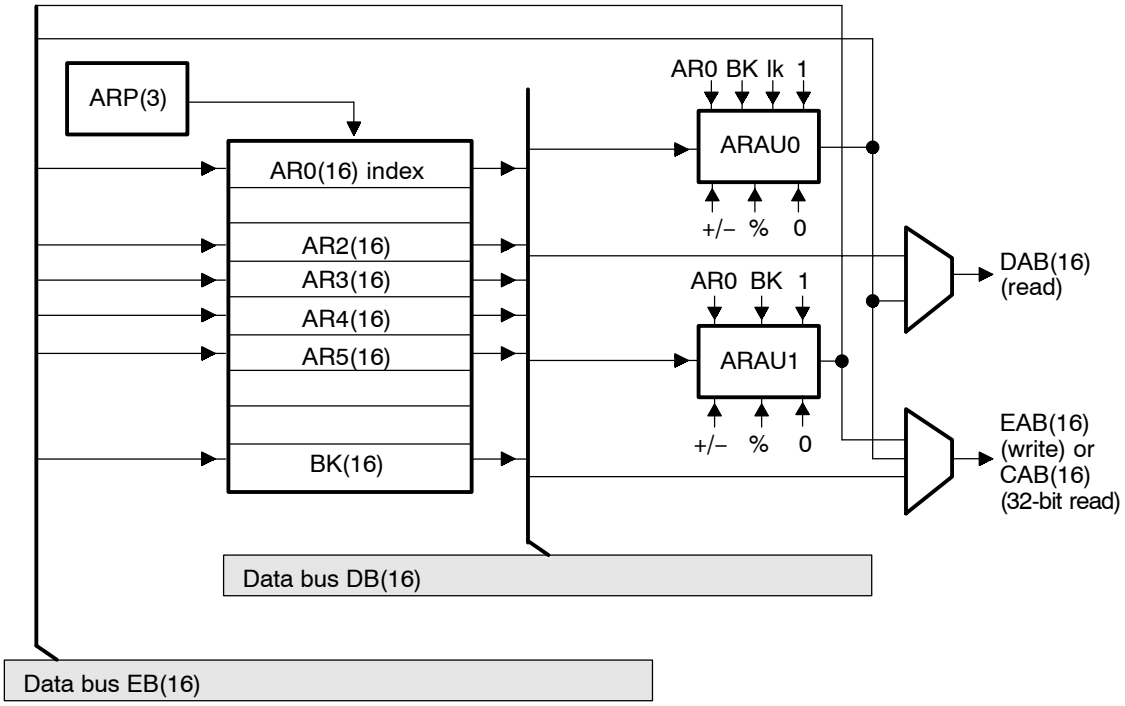


表5−8は、デュアル データ メモリ オペランド アドレッシングのタイプと、修飾フィールド(XmodまたはYmod)の値、アセンブラ構文、各タイプの機能を表しています。

表5−8. デュアル データ メモリ オペランドをともなう間接アドレッシングのタイプ

XmodまたはYmodフィールド	オペランド構文	ファンクション	説明†
00 (0)	*ARx	addr = ARx	ARxはデータ メモリ アドレスです。
01 (1)	*ARx−	addr = ARx ARx = ARx − 1	アクセス後、ARxのアドレスはデクリメントします。
10 (2)	*ARx+	addr = ARx ARx = ARx + 1	アクセス後、ARxのアドレスはインクリメントします。
11 (3)	*ARx+0%	addr = ARx ARx = circ(ARx + AR0)	アクセス後、サーキュラ アドレッシングを使ってAR0がARxに加えられます。‡

† ARxは、指定がなければデータ メモリ アドレスとして使用されます。
‡ サーキュラ バッファのサイズは、サーキュラ バッファ サイズ レジスタ(BK)で指定されます。

それぞれのケースで、補助レジスタの内容はデータ メモリ オペランドとして使用されます。補助レジスタのアドレスを使用した後、ARAUは指定されたアドレス演算を実行します。サーキュラ修飾をディセーブルにすることで、インデックス アドレッシングまたはARx+0と同等の演算を実行できます。BKレジスタを0にクリアすることで、サーキュラ修飾をディセーブルにできます。

> **CAUTION**
>
> デュアル オペランド リードを実行する命令では、Yarフィールドで指定される補助レジスタがメモリ マップド レジスタのひとつにアクセスする場合、リードされる値はレジスタの内容ではありません。

デュアル オペランド間接アドレッシングの例については、TMS320C54x DSPリファレンス セット第4巻:アプリケーション ガイドを参照してください。

5.5.4.1 デュアル オペランド インクリメント/デクリメント アドレス修飾(Xmod or Ymod=0, 1, or2)

ARの値をインクリメントまたはデクリメントして、ARを修飾できます。XmodまたはYmod=0のとき、ARxはデータ メモリ アドレスとして使用されインクリメントまたはデクリメントをともないません。XmodまたはYmod=1のとき、ARxはアクセス実行の後にデクリメントされます。XmodまたはYmod=2のとき、ARxはアクセス実行の後にインクリメントされます。

5.5.4.2 デュアル オペランド インデックス アドレス修飾(Xmod or Ymod=3およびBK=0)

XmodまたはYmod=3でBK=0のとき、AR0はそれぞれのアクセス後にARxに加えられます。そうでない場合は、デュアル オペランド インデックス アドレッシングはページ5−15のサブセクション5.5.3.3の説明のようになります。

5.5.4.3 デュアル オペランド サーキュラ アドレス修飾(Xmod or Ymod=3およびBK≠0)

XmodまたはYmod=3でBK≠0のとき、AR0はそれぞれのアクセス後にサーキュラ アドレッシングを使ってARxに加えられます。そうでない場合は、デュアル オペランドサーキュラ アドレッシングはページ5−15のサブセクション5.5.3.4の説明のようになります。

5.5.4.4 デュアル オペランド フォーマットを使用するシングル オペランド命令

データ メモリ オペランドをひとつだけ使用する命令の中には、デュアル データ メモリ オペランド アドレッシングを使うものがあり、これらは1サイクル実行のためにシングル ワードとなっています。これらの命令では、Xmemのみが使用でき、XmodおよびXarフィールドはオペランドのためのアドレッシング モードを指定します。下の4つのシングル オペランド命令はそれぞれ1サイクルで実行できます。

- BIT Xmem，BITC
- SACCD src，Xmem，cond
- SRCCD Xmem，cond
- STRCD Xmem，cond

オプションのシフトをともなう5つの命令も、このタイプのアドレッシングをサポートしてシングル ワード、1サイクル実行に対応します。

- ADD Xmem，SHFT，src
- LD Xmem，SHFT，dst
- STH src，SHFT，Xmem
- STL src，SHFT，Xmem
- SUBXmem，SHFT，src

5.5.5 TMS320C2x/C2xx/C5x互換(ARP)モード

ARPを間接アドレッシングの中で使用できます。これにより、ARをARPによって指定して、'C2x/C2xx/C5xデバイスからのコード変換を容易にできます。CMPT=1およびARF=0のとき、ARPを使ってどのARをアドレッシングに使用するかを決めます。図5−13は、ARPが補助レジスタをどのようにインデックスするかを示しています。

ARPの使用では、'54xがARPによってポイントされるARを使用するとき'54xは同じ命令でARPを更新しない点が、'54xと'C5xの違いになります。表5−9は、'C2x/C2xx/C5xのアセンブラ構文を'54xと対比して示しています。

図5−13. ARPが補助レジスタをインデックスする方法

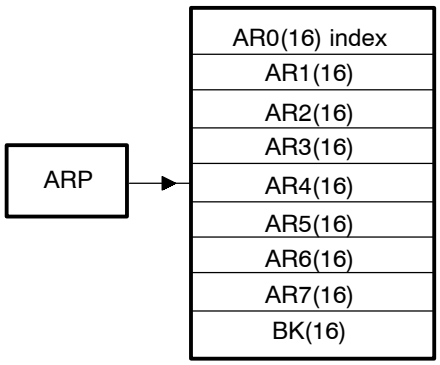


表5−9. TMS320C2x/C2xx/C5xと'54xのアセンブラ構文の対比

'C2x/C2xx/C5xの構文	'54xの構文	'C2x/C2xx/C5xの構文	54xの構文
*	*AR0	*0−	*AR0−0
*−	*AR0−	*0+	*AR0+0
*+	*AR0+	*BR0−	*AR0−0B
		*BR0+	*AR0+0B

図5−14は、ARPモードの間接アドレッシング命令フォーマットを示しています。表5−10は、ARPモード命令のビットについて説明しています。

図5−14. 互換モードの間接アドレッシング命令フォーマット

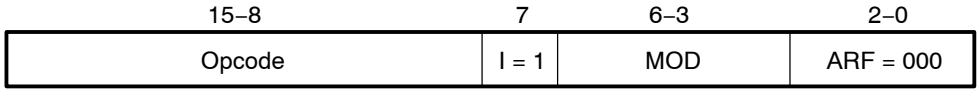


表5−10. 間接アドレッシング命令ビット概要—互換モード

ビット	名称	機能
15 − 8	Opcode	この8ビット フィールドは、命令の演算コードを含んでいます。
7	I	I=1のとき、命令が使用するアドレッシング モードは間接アドレッシング モードになります。
6−3	MOD	この4ビット修飾フィールドは、間接アドレッシングのタイプを指定します。MODフィールドで16通りのアドレッシング タイプを指定する方法については、ページ5−13、サブセクション5.5.3を参照してください。
2−0	ARF	この3ビット補助レジスタ フィールドは、アドレッシングで使用される補助レジスタを指定します。ARFは、ステータス レジスタST1の互換モード ビット(CMPT)によります。
	CMPT = 0	標準モード。このモードでは、ARPの値にかかわりなくARFは常に補助レジスタを指定します。ARPは更新されません。ARPはDSPがこのモードのとき、必ず常に0に設定されます。
	CMPT = 1	互換モード。このモードでは、ARF=0のときARPは補助レジスタを選択します。ARFは補助レジスタを選択して、アクセスが完了するとARF値がARPにロードされます。アセンブリ命令のAR0は、互換モードでARPが選択する補助レジスタを示します。

注:

ARPは常に、DSPが標準モード(CMPT=0)のときは必ず0に設定されます。リセット時に、ARPおよびCMPTの双方は自動的に0に設定されます。

5.6 メモリ マップド レジスタ アドレッシング

メモリ マップド レジスタ アドレッシングは、現在のデータ ページ ポインタ(DP)値または現在のスタック ポインタ(SP)値に影響を与えずに、メモリ マップド レジスタを変更するのに使用します。DPおよびSPはこのモードで変更する必要がないので、レジスタへの書き込みのオーバーヘッドは最小限になります。メモリ マップド レジスタ アドレッシングは、直接および間接アドレッシングの双方で機能します。

図5−15は、メモリ マップド アドレスがどのように生成されるかを示しています。アドレスは、次のように生成されます。

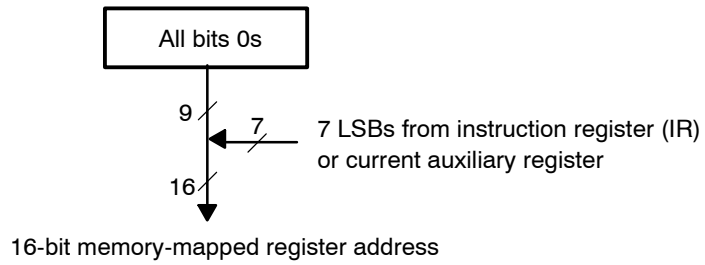
- 直接アドレッシングを使用するとき現在のDPまたはSP値にかかわりなく、データ メモリ アドレスの最上位9ビットを強制的に0にします。
- 間接アドレッシングを使用するとき、現在の補助レジスタ値の下位7ビットを使用します。

注:

間接アドレッシングでは、補助レジスタの上位9ビットが演算の後に強制的に0になります。

たとえば、メモリ マップド レジスタ モードでAR1を使ってメモリ マップド レジスタをポイントする際、AR1にFF25hの値があるなら、AR1の下位7ビットは25hでPRDのアドレスは0025hになるので、AR1はタイマ ピリオド レジスタ(PRD)をポイントします。実行後、AR1の値は0025hのままになります。

図5−15. メモリ マップド レジスタ アドレッシングのブロック図



注:

レジスタの他に、データ ページ0にあるスクラッチパッドRAMを、メモリ マップド レジスタ アドレッシングを使って変更できます。

8種類の命令がメモリ マップド レジスタ アドレッシングを使用できます。

- ❑ LDM MMR, dst
- ❑ MVDM dmda, MMR
- ❑ MVMD MMR, dmad
- ❑ MVMM MMRx, MMRy
- ❑ POPM MMR
- ❑ PSHM MMR
- ❑ STLM src, MMR
- ❑ STM #lk, MMR

注 :

次の間接アドレッシング モードは、メモリ マップド レジスタ アドレッシングでは使用できません。

- ❑ *ARx(lk)
- ❑ *+ARx(lk)
- ❑ *+ARx(lk)%
- ❑ *(lk)

これらのケースでは、アセンブラが警告を出力します。

5.7 スタック アドレッシング

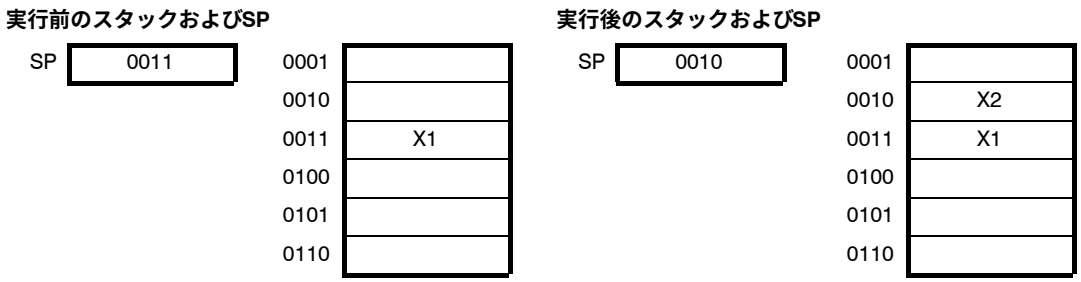
システム スタックは、割り込みおよびサブルーチン実行の間にプログラム カウンタを自動的に退避するために使用します。さらにアプリケーションによって、追加のコンテキストを退避するため、またはデータを渡すためにも使用できます。スタックは、メモリ アドレスの上位側から下位側に向かって使用されます。プロセッサは、16ビットのメモリ マップド レジスタ、スタック ポインタ(SP)を使って、スタックにアドレッシングします。SPは常に、スタックに退避される最新の内容をポイントします。

4つの命令が、スタック アドレッシング モードを使ってスタックにアクセスします。

- PSHDは、データ メモリ値をスタックにプッシュします。
- PSHMは、メモリ マップド レジスタをスタックにプッシュします。
- POPDは、データ メモリ値をスタックからポップします。
- POPMは、メモリ マップド レジスタをスタックからポップします。

プッシュは、SPのアドレスをプリデクリメントして、ポップはSPのアドレスをポストインクリメントします。図5−16は、X2をスタックにプッシュする前と後のスタックおよびSPの例を示しています(PSHD X2)。

図5−16. プッシュ実行前後のスタックおよびスタック ポインタ



他の演算も、スタックおよびスタック ポインタに影響を与えます。割り込みおよびサブルーチン実行の間にスタックを使用して、PCの内容を退避、復元します。サブルーチンが呼び出されるか割り込みが発生すると、プッシュ操作を使ってリターン アドレスが自動的にスタックに退避されます。サブルーチン コールおよび割り込みに使われる命令は、CALA[D]、DALL[D]、CC[D]、INTR、TRAPです。

サブルーチンからリターンすると、ポップ操作をしてリターン アドレスをスタックからPCにロードします。サブルーチンからの復帰のために使用される命令は、RET[D]、RETE[D]、RETEF[D]、RC[D]です。

FRAME命令も、スタックに影響を与えます。この命令はショート イミディエイト オフセットをスタック ポインタに加えます。スタックは、SPリファレンスド直接アドレッシングでも使用されます(5−9ページ、セクション5.4.2 SPリファレンスド直接アドレッシングを参照)。

5.8 データ タイプ

’54xでは、メモリにアクセスするためのデータ タイプは、基本的に16ビットおよび32ビットの2種類があります。ほとんどの命令は、16ビット データにアクセスできます。一方32ビット データにアクセスするには、表5−11に示すような特別な命令を使います。

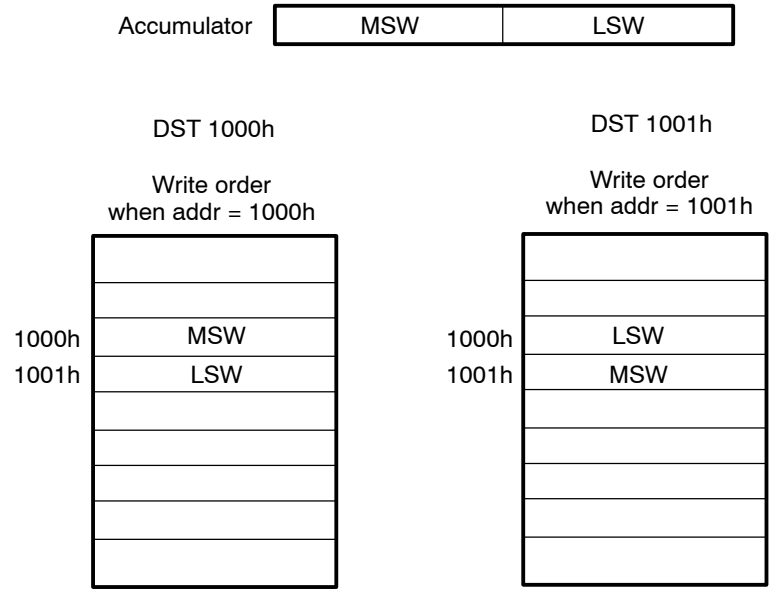
表5−11. 32ビット ワード オペランドをともなう命令

命令	説明
DADD	アキュムレータへの倍精度加算/デュアル16ビット加算
DADST	Tレジスタの内容を加算する倍精度ロード、Tレジスタ加算/減算をともなうデュアル16ビット ロード
DLD	アキュムレータに倍精度ワードをロード
DRSUB	倍精度ワードから倍精度減算/デュアル16ビット減算
DSADT	Tレジスタをともなう倍精度ロード、Tレジスタ減算/加算をともなうデュアル16ビット ロード
DST	倍精度ワードでアキュムレータをストア
DSUB	アキュムレータからの倍精度減算/デュアル16ビット減算
DSUBT	Tレジスタ減算をともなう倍精度ロード/Tレジスタ減算をともなうデュアル16ビット ロード

16ビット オペランド アクセスでは、16ビット ワードがデータ メモリからDバスによってリードされ、Eバスによってデータ メモリにライトされます。32ビット オペランド アクセスでは、C(上位ワードに対応)およびD(下位ワードに対応)バスがリードに使用されます。しかし、ライトにはEバスのみが使用されるので、ライト処理(DST命令)は2サイクルで実行されます。

32ビット アクセスでは、アクセスされる最初のワードは上位ワード(MSW)として扱われ、アクセスされる2番目のワードは下位ワード(LSW)として扱われます。アクセスされる最初のワードが偶数のアドレスにある場合は、2番目のワードは次の(より上位の)アドレスになります。アクセスされる最初のワードが奇数アドレスにある場合は、2番目のワードは前の(より下位の)アドレスになります。図5−17は、この様子を示しています。

図5−17. **メモリのワード順位**



5

第6章

プログラム メモリ アドレッシング

本章では、プログラム メモリ アドレスがどのように発生して、どのアドレスがプログラム カウンタ(PC)にロードされるかを説明します。さらに、PCにロードされる値に影響するプログラム制御についても解説します。

- ❑ 分岐
- ❑ サブルーチン コール
- ❑ リターン
- ❑ 条件付き命令
- ❑ 1命令または命令ブロックのリピート
- ❑ ハードウェア リセット
- ❑ 割り込み

以上の命令等により、PCにロードされるアドレスは不連続となります。セクション7.1パイプライン演算、7.2割り込みおよびパイプラインは、PC中断の不連続動作を理解する上で役立ちます。また、パワーダウン モードはプログラム実行を停止します。

6

Topic Page

6.1 プログラム メモリ アドレス生成

プログラム メモリには、アプリケーションのコード、係数テーブル、イミディエイト オペランドがあります。'54xは、プログラム アドレス バス(PAB)を使って、総計64Kワードのプログラム メモリをアドレスできます。一部の拡張メモリ アドレスをもつデバイスには追加のアドレス ラインが7本あり、64Kワード 128ページの外部プログラム メモリにアクセスできます。プログラム アドレス生成ロジック(PAGEN)は、命令、係数テーブル、16ビット イミディエイト オペランド、プログラム メモリにストアされる他の情報にアクセスするために使用するアドレスを生成したり、このアドレスをPABに送ります。

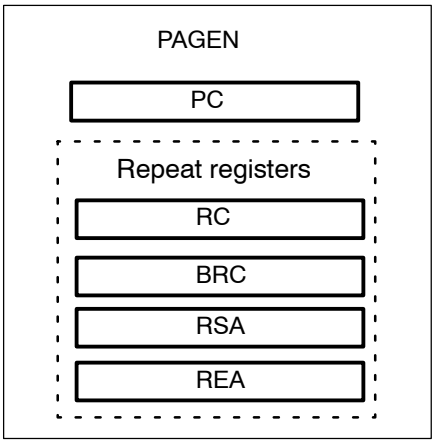
PAGENは、次の5つのレジスタから構成されます(図6−1を参照)。

- プログラム カウンタ(PC)
- リピート カウンタ(RC)
- ブロック リピート カウンタ(BRC)
- ブロック リピート開始アドレス レジスタ(RSA)
- ブロック リピート終了アドレス レジスタ(REA)

一部の拡張メモリ アドレスをもつデバイスでは、更にもうひとつのレジスタを使って拡張メモリにアドレスします。

- プログラム カウンタ拡張レジスタ(XPC)

図6−1. プログラム アドレス生成ロジック(PAGEN)レジスタ



'54xデバイスは、PCの値をPABに送りメモリの適切な位置を読み取ることにより命令をフェッチします。メモリのリードの際、PCは次のフェッチのためにインクリメントされます。プログラム アドレスの不連続が発生すると(たとえば分岐、コール、リターン、割り込み、ブロック リピートなど)、適切なアドレスがPCにロードされます。次に、PAB経由でアドレッシングされた命令が、命令レジスタ(IR)にロードされます。

命令の処理を高度化するために、プログラム アドレス生成ユニットを使ってプログラム メモリからオペランドをフェッチできます。オペランドは、デバイスが係数テーブルをリード/ライトするときや、データをプログラム空間およびデータ空間の間で転送するとき、プログラム メモリからフェッチされます。FIRS、MACD、MACPなどの命令では、プログラム バスを使って2番目の被乗数をフェッチします。

6.2 プログラム カウンタ(PC)

PCは16ビット レジスタで、内部または外部プログラム メモリのアドレスを保持しており、これらのアドレスは、命令がフェッチされるときや16ビット イミディエイト オペランドまたはプログラム メモリの係数テーブルにアクセスするときに使用されます。プログラム メモリをアドレッシングするには、PCのアドレスをPABに送ります。

PCは、いくつかの方法でロードできます。表6−1は、命令の実行に応じて何がPCにロードされるかを示しています。

表6−1. アドレスをPCにロードする

オペレーション	PCにロードされるアドレス
リセット	PCにFF80hがロードされます。
連続実行	PCにPC+1がロードされます。
分岐	PCに分岐命令直後の16ビット イミディエイト値がロードされます。
アキュムレータによる分岐	PCにアキュムレータAまたはBの下位16ビットがロードされます。
ブロック リピート ループ	BRAF=1の時、PC+1がリピート終了アドレス(REA)+1と等しいとき、PCにリピート開始アドレス(RSA)がロードされます。
サブルーチン コール	PC+2がスタックにプッシュされ、PCにコール命令ニーモニック直後の16ビット イミディエイト値がロードされます。リターン命令はスタック最上段をPCにポップして、コール元のコード シーケンスに返ります。
アキュムレータからのサブルーチン コール	PC+1がスタックにプッシュされ、PCにアキュムレータAまたはBの下位16ビット ワードがロードされます。リターン命令はスタック最上段をPCにポップして、コール元のコード シーケンスに返ります。
ハードウェア割り込み、ソフトウェア割り込み、またはトラップ	PCがスタックにプッシュされ、PCに適切なトラップ ベクタのアドレスがロードされます。リターン命令はスタック最上段をPCにポップして、割り込みが発生したコード シーケンスに返ります。

6.2.1 プログラム カウンタ拡張レジスタ(XPC、一部の拡張アドレスをもつデバイスのみ)

XPCは7ビット レジスタで、プログラム メモリの現在のページを選択します。拡張プログラム メモリの詳細については、ページ3−8、サブセクション3.1.1拡張プログラム メモリを参照してください。

XPCは、PCのロードとともにいくつかの方法でロードできます。表6−2は、XPCをロードする処理を示しています。

表6−2. アドレスをXPCにロードする

オペレーション	PCにロードされるアドレス
リセット	PCにFF80hがロードされます。XPCには0hがロードされます。
連続実行	PCにPC+1がロードされます。ただし、XPCは自動的にはインクリメントされません。
ファー ブランチ	PCに、分岐命令直後のイミディエイト値のビット15−0がロードされます。XPCにはイミディエイト値のビット22−16がロードされます。
アキュムレータによるファー ブランチ	PCにアキュムレータAまたはBのビット15−0がロードされます。XPCには、アキュムレータAまたはBのビット22−16がロードされます。
ファー サブルーチン コール	XPCがスタックにプッシュされ、PC+2もスタックにプッシュされ、PCおよびXPCにそれぞれ、コール命令で指定されるイミディエイト値のビット15−0およびビット22−16がロードされます。
アキュムレータのよるファー サブルーチン コール	XPCがスタックにプッシュされ、PC+1もスタックにプッシュされ、PCおよびXPCにそれぞれ、アキュムレータAまたはBのビット15−0およびビット22−16がロードされます。
ファー リターン	リターン命令はスタック最上段をPCにポップして、次の値をXPCにポップしてコール元のコード シーケンスに返ります。

注：

XPCは、表6−2に示される以外の命令ではロードされません。

6.3 分岐

分岐は、制御をプログラム メモリの別の位置に移すことで、命令の連続的な流れを断ち切ります。したがって、分岐は生成されPCにストアされるプログラム アドレスに影響します。'54xには、条件なしおよび条件付きの双方の分岐命令があり、更にそれぞれに対して遅延なしまたは遅延ありのタイプがあります。

6.3.1 条件なし分岐

条件なし分岐は無条件に実行されます。実行中に、PCに、指定された分岐先のプログラム メモリのアドレスがロードされ、コードの新しいセクションの実行がそのアドレスで始まります。PCにロードされるアドレスは、分岐命令の2番目のワードまたはアキュムレータ(アキュムレータAまたはB)の下位16ビットにより指定されます。

分岐命令がパイプラインの実行フェーズに到達した時点で、分岐命令につづく2つの命令ワードはすでにフェッチされています。これら2つの命令ワードが処理されるかは、分岐が「遅延なし」か「遅延あり」かに依存します。

- 遅延なし:分岐命令につづく2つの命令ワードはフラッシュされるので、それらは実行されません。また実行は分岐先のアドレスより続行します。
- 遅延あり:分岐命令に続く2ワード命令ひとつまたは1ワード命令ふたつが実行されます。これにより、パイプラインをフラッシュすることによって発生する余分なサイクルが必要になることを防止します。

> 注:
>
> 遅延ありの分岐命令に続く2ワードは、PCの不連続(分岐、コール、リターン、ソフトウェア割り込みなど)を引き起こす命令であってはなりません。

表6−3は、'54xの条件なし分岐命令、およびこれらの命令(遅延なしおよび遅延あり)の実行に必要なサイクル数を示しています。遅延ありの命令は、対応する遅延なしの命令に比べて2サイクル少ないですが、これは遅延ありの命令がパイプラインをフラッシュしないからです。

表6−3. 条件なし分岐命令

命令	説明	サイクル数 (遅延なし/遅延あり)
B[D]	命令が指定するアドレスをPCにロード。	4/2
BACC[D]	指定されたアキュムレータの下位16ビットが示すアドレスをPCにロード。	6/4

6.3.2 条件付き分岐

条件付き分岐はユーザが指定した条件に合うときのみ実行します。指定できる条件は、ページ6−16、表6−12に示してあります。条件すべてが満たされる場合、PCに分岐アドレスを含む分岐命令の2番目のワードがロードされ、そのアドレスから実行し続けます。

条件がテストされた時点で、条件付き分岐命令に続く2つの命令ワードがすでにフェッチされており、それらはパイプライン上にあります。これらの2つの命令ワードがどのように処理されるかは、分岐が「遅延なし」か「遅延あり」かに依存します。

- 遅延なし:すべての条件が合うと、これらの2つの命令ワードはパイプラインからフラッシュされて実行されず、実行は分岐先のアドレスから続行します。条件が満たされないときは、分岐命令に続く2つの命令ワードは実行されます。
- 遅延あり:分岐命令に続く2ワード命令ひとつまたは1ワード命令ふたつが実行されます。これにより、パイプラインをフラッシュすることによって発生する余分なサイクルが必要になることを防止します。テストされる条件は、遅延分岐に続く命令には影響されません。

> 注:
>
> 遅延命令に続く2ワードは、PCの不連続(分岐、コール、リターン、ソフトウェア割り込みなど)を引き起こす命令であってはなりません。

表6−4は、条件付き分岐命令およびこれらの命令の実行に必要なサイクル数を示しています。条件付き分岐は、前の命令の実行によって決められる条件を使用するので、条件付き分岐命令、BC[D]は、条件なし分岐命令にくらべて1サイクル多くなります。

表6−4. 条件付き分岐命令

命令	説明	サイクル数 (条件適合/不適合) 遅延なし	遅延あり
BC[D]	命令によって指定された条件が満たされれば、命令が指定するアドレスをPCにロード。	5/3	3/3
BANZ[D]	ループ動作に用いられている補助レジスタの値が0でない場合、命令が指定するアドレスをPCにロード(ループに有用)。	4/2	2/2

6.3.3 ファー ブランチ(一部の拡張メモリ アドレスを持つデバイスのみ)

拡張メモリへの分岐が可能になるように、2つのファー ブランチ命令があります。

- FB[D]は、命令が指定する23ビット アドレスに分岐します。
- FBACC[D]は、指定されたアキュムレータで示される23ビット アドレスに分岐します。

表6−5は、ファー ブランチ命令およびこれらの命令の実行に必要なサイクル数を示しています。遅延命令は、対応する遅延なし命令より2サイクル少なくなります。

表6−5. ファー ブランチ命令

命令	説明	サイクル数 (遅延なし/遅延あり)
FB[D]	命令で指定されるアドレスをPCおよびXPCにロード。	4/2
FBACC[D]	指定されたアキュムレータの下位23ビットが示すアドレスをPCおよびXPCにロード。	6/4

6.4 コール

コールは分岐のように、プログラム メモリの他の位置に制御を移すことで、連続的な命令の流れを断ち切ります。しかし分岐とは異なり、この制御の移動は一時的なものです。サブルーチンまたはファンクションがコールされると、コールに続く次の命令のアドレスがスタックに保存されます。このアドレスを使ってコール元のプログラムに返り、実行を再開します。'54xは、条件なしコールおよび条件付きコール命令があり、更にそれぞれに対して遅延なしおよび遅延ありのタイプがあります。

6.4.1 条件なしコール

条件なしコールは無条件に実行します。コールを実行するときは、指定されたプログラム メモリ アドレスをPCにロードして、そのアドレスで始まり、呼び出されたルーチンが実行がされます。PCにロードされるアドレスは、コール命令の2番目のワード、またはアキュムレータ(アキュムレータAまたはB)の下位16ビットです。PCにロードされる前に、リターン アドレスがスタックに保存されます。サブルーチンまたはファンクションが実行された後、リターン命令がリターン アドレスをPCにスタックからロードして、コールに続く命令で実行が再開します。

条件なしコール命令がパイプラインの実行フェーズに到達する時点で、次の2つの命令ワードがすでにフェッチされています。これら2つの命令がどのように処理されるかは、コールが遅延なしか遅延ありかに依存します。

- 遅延なし:2つの命令ワードがパイプラインからフラッシュされて、これらが実行されず、リターン アドレスがスタックにストアされます。さらに実行は呼び出されたファンクションの先頭から実行されます。
- 遅延あり:コール命令に続く2ワード命令ひとつまたは1ワード命令ふたつが実行されます。これにより、パイプラインをフラッシュするこによって発生する余分なサイクルが必要になることを防止します。

> 注:
>
> 遅延命令に続く2ワードは、PCの不連続(分岐、コール、リターン、ソフトウェア割り込みなど)を引き起こす命令であってはなりません。

表6−6は、'54xの条件なしコール命令(遅延なしおよび遅延あり)、およびこれらの命令の実行に必要なサイクル数を示しています。遅延命令は、対応する遅延なし命令より2サイクル少なくなりますが、これは遅延命令はパイプラインをフラッシュしないからです。

表6−6. 条件なしコール命令

命令	説明	サイクル数 (条件適合/不適合)
CALL[D]	リターン アドレスをスタックに置いて、命令が指定するアドレスをPCにロード。	4/2
CALA[D]	リターン アドレスをスタックに置いて、指定のアキュムレータで示されるアドレスをPCにロード。	6/4

6.4.2 条件付きコール

条件付きコールはひとつ以上の条件が満たされたときのみ実行します。設定できる条件を、6−16ページの表6−12に示してあります。すべての条件が満たされると、コールされるファンクションの開始アドレスであるコール命令の2番目のワードがPCにロードされます。コールされるファンクションに分岐する前に、プロセッサはコールの続く命令のアドレスをスタックにストアします。このコールされたファンクションは必ずリターン命令で終了されなければなりません。

条件付きコール命令の条件がテストされた時点で、コール命令に続く2つの命令ワードがすでにパイプラインにフェッチされています。これらの2つの命令ワードがどのように処理されるかは、呼び出しが遅延なしか遅延ありかに依存します。

- 遅延なし:すべての条件が満たされる場合は、これらの2つの命令ワードがパイプラインからフラッシュされるので、これらは実行されません。また、実行はコールされたファンクションの始まりのアドレスから続行されます。条件が満たない場合は、コールに続く2つの命令が実行されます。
- 遅延あり:コール命令に続く2ワード命令ひとつまたは1ワード命令ふたつが、常に実行されます。これにより、パイプラインをフラッシュすることで発生する余分なサイクルが必要になることを防止します。テストされる条件は遅延コールに続く命令には影響されません。条件が満たない場合は、プロセッサはコールに続く2つの命令ワードを実行します。

> 注:
>
> 遅延命令に続く2ワードは、PCの不連続(分岐、コール、リターン、ソフトウェア割り込みなど)を引き起こす命令であってはなりません。

表6−7は、条件付きコール命令およびその命令の実行に必要なサイクル数を示しています。条件が確定するためのウェイト サイクルがあるので、条件付きコールCC[D]は、条件なしコールより1サイクル多く必要になります。

表6−7. 条件付きコール命令

命令	説明	サイクル数 (条件適合/条件不適合) 遅延なし	遅延あり
CC[D]	命令が指定する条件が満たされれば、リターン アドレスをスタックに置いて、命令が指定するアドレスをPCにロード。	5/3	3/3

6.4.3 ファー コール(一部の拡張メモリ アドレスを持つデバイスのみ)

拡張メモリへのコールを可能にするために、2つのファー コール命令があります。

- ❑ FCALL命令は、XPCをスタックにプッシュして、PCをスタックにプッシュして、命令が指定する23ビット アドレスに分岐します。
- ❑ FCALAは、XPCをスタックにプッシュして、PCをスタックにプッシュして、指定されたアキュムレータが示す23ビット アドレスに分岐します。

表6−8は、ファー コール命令(遅延なしおよび遅延あり)およびこれらの命令の実行に必要なサイクル数を示しています。遅延命令は対応する遅延なし命令より2サイクル少なくなります。

表6−8. ファー コール命令

命令	説明	サイクル数 (遅延なし/遅延あり)
FCALL[D]	XPCおよびPCをスタックに置いて、XPCおよびPCに命令が指定するアドレスをロードします。	4/2
FCALA[D]	XPCおよびPCをスタックに置いて、指定されたアキュムレータで示されるアドレスをXPCおよびPCにロードします。	6/4

6.5 リターン

リターン命令により、別のファンクションのコールや割り込みサービス ルーチンのために中断された連続する命令の処理を再開します。コールされたファンクションや割り込みサービス ルーチンが実行を完了すると、コール直後の時点または割り込みが発生したポイントの制御に戻って再開する必要があります。リターン命令は、次に実行される命令のアドレスを持つスタック最上段の値をプログラム カウンタ(PC)にポップすることで、処理再開を実現します。'54xは条件なしおよび条件付きリターン命令があり、更にそれぞれに対して遅延なし、遅延ありのタイプがあります。

一部の拡張メモリ アドレスを持ったデバイスには、条件なしファー リターン命令(遅延なし/遅延あり)も備えています。

6.5.1 条件なしリターン

条件なしリターンは無条件に実行されます。リターンを実行すると、PCにスタックからのリターン アドレスがロードされ、ファンクションを呼び出した命令に続く命令、または割り込みが発生したポイントで実行を再開します。

条件なし復帰命令がパイプラインの実行ステージに達した時点で、次の2つの命令ワードがすでにフェッチされています。これら2つの命令ワードが処理されるかどうかは、復帰が遅延なしか遅延ありかに依存します。

- 遅延なし:2つの命令ワードはパイプラインからフラッシュされるので、それらは実行されず、リターン アドレスがスタックまたはRTNレジスタから取られます。さらに、実行はコール元のファンクションのそのアドレスから実行されます。
- 遅延あり:リターン命令に続く2ワード命令ひとつ、または1ワード命令ふたつが実行されます。これにより、パイプラインをフラッシュすることによって発生する余分なサイクルが必要になることを防止します。リターン アドレスは、スタックまたはRTNレジスタから取られます。

> 注:
>
> 遅延命令に続く2ワードは、PCの不連続(分岐、コール、リターン、ソフトウェア割り込みなど)を引き起こす命令であってはなりません。

表6−9は、'54xの条件なし復帰命令(遅延なしおよび遅延あり)およびこれらの命令の実行に必要なサイクル数を示しています。遅延あり命令は、対応する遅延なし命令より2サイクル少なくなります。

表6−9. 条件なしリターン命令

命令	説明	サイクル数 (遅延なし/遅延あり)
RET[D]	スタック最上段のリターン アドレスをPCにロード。	5/3
RETE[D]	スタック最上段のリターン アドレスをPCにロードして、マスカブル割り込みをイネーブル。	5/3
RETF[D]	RTNレジスタのリターン アドレスをPCにロードして、マスカブル割り込みをイネーブル。	3/1

RETEおよびRETF命令によって、別の割り込みが実行される前に確実にリターンを実行し、割り込みをイネーブルにすることができます。RETF命令を使い、スタックでなくRTNレジスタからPCをロードすることで、より迅速なリターンを実現します。これにより、割り込みルーチンに使われる総サイクル数を減少できます。これは、特に頻繁に使用される短い割り込みルーチンにとって重要です。

注:

RTNレジスタは、リード/ライト不可能なCPU内蔵レジスタです。

6.5.2 条件付きリターン

条件付きリターン(RC)命令を使うことにより、ファンクションまたは割り込みサービス ルーチン(ISR)で可能なリターン パスが複数になります。選ばれるパスは、処理されるデータによります。さらに、条件付きリターン命令を使えば、ファンクションまたはISRの終わりのリターン命令に条件付き分岐する必要がありません。

条件付きリターンは1つ以上の条件が満たされるときのみ実行します。指定できる条件は、6−16ページ、表6−12に示しています。すべての条件が満たされる場合は、プロセッサはリターン アドレスをスタックからPCにロードして、コール元のプログラムの実行を再開します。

条件付きリターンは単一ワード命令ですが、PCの値が不連続に変化するため、条件付き分岐またはコールと同じ実行時間で実行されます。

条件付きリターン命令の条件がテストされた時点では、リターン命令に続く2つの命令ワードがすでにパイプラインにフェッチされています。これら2つの命令ワードがどのように処理されるかは、リターンが遅延なしか遅延ありかに依存します。

- 遅延なし:すべての条件が合う場合は、これらリターン命令につづいて2つの命令ワードはパイプラインからフラッシュされるので、これらは実行されず、コール元のプログラムの実行が続行します。条件が合わない場合は、リターン命令に続く2つの命令を実行します。
- 遅延あり:プロセッサは、リターン命令に続く2つの命令を実行します。これによりパイプラインをフラッシュすることによって発生する余分なサイクルが必要になることを防止します。テストされる条件は、遅延リターン命令に続く命令に影響されません。

注:

遅延命令に続く2ワードは、PCの不連続(分岐、コール、リターン、ソフトウェア割り込みなど)を引き起こす命令であってはなりません。

表6−10は、条件付きリターン命令、およびこれらの命令の実行に必要なサイクル数を示しています。

表6−10. 条件付きリターン命令

命令	説明	サイクル数 (条件に適合/条件に不適合) 遅延なし	遅延あり
RC[D]	命令が指定する条件が合う場合、スタック最上段のリターン アドレスでPCをロード。	5/3	3/3

6.5.3 ファー リターン(一部の拡張メモリ アドレスを持つデバイスのみ)

拡張メモリからの復帰を可能にするため、2つのファー リターン命令があります。

- FRETは、スタックからXPCをロードした後、スタックからPCをロードすることにより、プログラム実行が前の地点から再開します。
- FRETEは、スタックからXPCをロードした後、スタックからPCをロードして、マスカブル割り込みをイネーブルにします。

表6−11は、ファー リターン(遅延なしおよび遅延あり)およびこれらの実行に必要なサイクル数を示しています。遅延命令は、対応する遅延なし命令より2サイクル少なくなります。

表6−11. ファー リターン命令

命令	説明	サイクル数 (遅延なし/遅延あり)
FRET[D]	スタック最上段の値をXPCにロードして、スタックの次の値をPCにロード。	6/4
FRETE[D]	スタック最上段の値をXPCにロードしてから、スタックの次の値をPCにロードして、マスカブル割り込みをイネーブル。	6/4

6.6 条件付き命令

’54xには、ひとつ以上の条件を満たした場合のみ実行する命令があります。表6−12は、これらの命令で使用できる条件、および対応するオペランド記号を示しています。

表6−12. 条件付き命令の条件

条件	説明	オペランド
A = 0	Accumulator A equal to 0	AEQ
B = 0	Accumulator B equal to 0	BEQ
A ≠ 0	Accumulator A not equal to 0	ANEQ
B ≠ 0	Accumulator B not equal to 0	BNEQ
A < 0	Accumulator A less than 0	ALT
B < 0	Accumulator B less than 0	BLT
A ≤ 0	Accumulator A less than or equal to 0	ALEQ
B ≤ 0	Accumulator B less than or equal to 0	BLEQ
A > 0	Accumulator A greater than 0	AGT
B > 0	Accumulator B greater than 0	BGT
A ≥ 0	Accumulator A greater than or equal to 0	AGEQ
B ≥ 0	Accumulator B greater than or equal to 0	BGEQ
AOV = 1	Accumulator A overflow detected	AOV
BOV = 1	Accumulator B overflow detected	BOV
AOV = 0	No accumulator A overflow detected	ANOV
BOV = 0	No accumulator B overflow detected	BNOV
C = 1	ALU carry set to 1	C
C = 0	ALU carry cleared to 0	NC
TC = 1	Test/control flag set to 1	TC
TC = 0	Test/control flag cleared to 0	NTC
$\overline{\text{BIO}}$ low	$\overline{\text{BIO}}$ signal is low	BIO
$\overline{\text{BIO}}$ high	$\overline{\text{BIO}}$ signal is high	NBIO
none	Unconditional operation	UNC

6.6.1 複数条件の使用

複数条件は、条件付き命令のオペランドとして記述できます。複数条件が記述される場合は、すべての条件が満たされた場合に命令を実行します。(表6−13参照)。それぞれの条件の組み合わせは、次のグループ1およびグループ2から選ばれます。

- ❑ グループ1:カテゴリA、カテゴリBから条件をひとつずつ選択できます。同じカテゴリから2つの条件は選択できません。たとえば、EQおよびOVを同時にテストできますが、GTおよびNEQを同時にテストできません。両方の条件は必ず同一のアキュムレータを使用しなければいけません。一つの命令によって二つのアキュムレータをテストできません。たとえば、AGTおよびAOVを同時にテストできますが、AGTおよびBOVを同時にテストできません。

- ❑ グループ2:3つのカテゴリ(A、B、C)それぞれからひとつの条件を選択できます。同じカテゴリから2つの条件は選択できません。たとえば、TC、C、BIOを同時にテストできますが、NTC、C、NCを同時にテストできません。

表6−13. 複数条件命令の条件グループ

Group 1		Group 2		
Category A	Category B	Category A	Category B	Category C
EQ	OV	TC	C	BIO
NEQ	NOV	NTC	NC	NBIO
LT				
LEQ				
GT				
GEQ				

6.6.2 条件付き実行(XC)命令

1または2ワード コードを条件付き分岐命令で分岐する場合、その分岐を1サイクルの条件付き実行命令(XC)と入れ替えることができます。XC命令には、2種類の形式があります。ひとつは1ワードでの条件付き実行命令(XC 1, cond)、もうひとつはひとつの2ワード命令の条件付き実行またはふたつの1ワード命令の条件付き実行です(XC 2, cond)。XCの条件は、条件付き分岐、コール、リターンの条件と同じです(表6−12を参照)。

XC命令が実行される前に、条件を安定させるため2サイクルが必要になります。これにより、XCに続く命令がデコードされる前に、確実に条件の判定が行われます。XCの前のふたつの1ワード命令によってXCの条件が影響を受けない様にしなければいけません。割り込み発生がないときは、これらの命令はXCに影響しませんが、割り込みが発生する場合は、その命令およびXCの間でトラップされるため、XCが実行される前に条件に影響します。パイプラインのレイテンシについては、第7章パイプラインを参照してください。

6.6.3 条件付きストア命令

CPUレジスタによっては、条件付きストア命令を使って条件付きでデータ メモリにストアできるものもあります。これらの命令は、表6−14に示しています。条件付きストア命令に使用される条件は、表6−15に示しています。

条件付きストア命令では、条件にかかわりなくアドレスは修飾され、メモリ オペランドがリードされます。条件が合う場合は、対応するレジスタがデータ メモリにストアされます。条件が合わない場合は、リードされたのと同じメモリ位置にオペランドがライトされるため、そのメモリ位置の値はそのままになります。

条件付きストア命令は1メモリ オペランド命令ですが、これはデュアル メモリ オペランド間接アドレッシング モードを使って命令をひとつの16ビット ワードにします。したがって、これらの命令は1サイクルで実行されます。

条件付きでブロック リピート カウンタ(BRC)をストアすることで、リピート ブロック ループのインデックスをストアできます。

表6−14. 条件付ストア命令

命令	CPUレジスタ
SACCD	Accumulator A or B
STRCD	Temporary register (T)
SRCCD	Block-repeat counter (BRC)

表6−15. 条件付きストア命令の条件

オペランド	条件	説明
AEQ	A = 0	Accumulator A equal to 0
BEQ	B = 0	Accumulator B equal to 0
ANEQ	A ≠ 0	Accumulator A not equal to 0
BNEQ	B ≠ 0	Accumulator B not equal to 0
ALT	A < 0	Accumulator A less than 0
BLT	B < 0	Accumulator B less than 0
ALEQ	A ≤ 0	Accumulator A less than or equal to 0
BLEQ	B ≤ 0	Accumulator B less than or equal to 0
AGT	A > 0	Accumulator A greater than 0
BGT	B > 0	Accumulator B greater than 0
AGEQ	A ≥ 0	Accumulator A greater than or equal to 0
BGEQ	B ≥ 0	Accumulator B greater than or equal to 0

6.7 単一命令をリピートする

’54xにはRPTおよびRPTZの2種類の命令があり、これによって後に続く命令を繰り返すことができます。命令を繰り返す回数は、命令のオペランドから得ることができ、その回数はこのオペランド+1になります。この値は16ビット リピート カウンタ(RC)レジスタにストアされます。RCレジスタにあるこの値はリピート命令(RPTまたはRPTZ)によってのみロードされます。指定された命令の最大実行数は、65 536回です。絶対プログラム アドレスまたは絶対データ アドレスは、シングル リピート機能を使用すると自動的にインクリメントされます。

リピート命令がデコードされると、リピート ループが終了するまですべての割り込み($\overline{\text{NMI}}$も含む、$\overline{\text{RS}}$は含まず)がディセーブルになります。ただし、ST1のHMビットの設定によりRPT/RPTZループの実行の間に$\overline{\text{HOLD}}$信号を受け付けるとこができます。

このリピート機能は、積和演算やブロック転送など一部の命令とともに使用すると、それらの命令の実効速度を高速化できます。リピート命令をともない実行される、これらのマルチサイクルの命令(表6−16参照)は、最初の繰り返しを除き、1サイクル命令で実現できます。

表6−16. リピート時に1サイクルで実行されるマルチサイクル命令

Instruction	Description	# Cycles†
FIRS	Symmetrical FIR filter	3
MACD	Multiply and move result in accumulator with delay	3
MACP	Multiply and move result in accumulator	3
MVDK	Data-to-data move	2
MVDM	Data-to-MMR move	2
MVDP	Data-to-program move	4
MVKD	Data-to-data move	2
MVMD	MMR-to-data move	2
MVPD	Program-to-data move	3
READA	Program-to-data move	5
WRITA	Data-to-program move	5

† Number of cycles when instruction is not repeated

ロング オフセット修飾子または絶対アドレスを使う場合は(*ARn(lk)、*+ARn(lk)、*+ARn(lk)%、*(lk)など)、シングル データ メモリ オペランド命令はリピートできません。表6−17に示した命令は、RPTを使って繰り返せません。

表6−17. **リピート不可の命令**

Instruction	Description
ADDM	Add long constant to data memory
ANDM	AND data memory with long constant
B[D]	Unconditional branch
BACC[D]	Branch to accumulator address
BANZ[D]	Branch on auxiliary register not 0
BC[D]	Conditional branch
CALA[D]	Call to accumulator address
CALL[D]	Unconditional call
CC[D]	Conditional call
CMPR	Compare with auxiliary register
DST	Long word (32-bit) store
FB[D]	Far branch unconditionally
FBACC[D]	Far branch to location specified by accumulator
FCALA[D]	Far call subroutine at location specified by accumulator
FCALL[D]	Far call unconditionally
FRET[D]	Far return
FRETE[D]	Enable interrupts and far return from interrupt
IDLE	Idle instructions
INTR	Interrupt trap
LD ARP	Load auxiliary register pointer (ARP)
LD DP	Load data page pointer (DP)
MVMM	Move memory-mapped register (MMR) to another MMR
ORM	OR data memory with long constant
RC[D]	Conditional return
RESET	Software reset
RET[D]	Unconditional return

表6−17. リピート不可の命令(続き)

Instruction	Description
RETE[D]	Return from interrupt
RETF[D]	Fast return from interrupt
RND	Round accumulator
RPT	Repeat next instruction
RPTB[D]	Block repeat
RPTZ	Repeat next instruction and clear accumulator
RSBX	Reset status register bit
SSBX	Set status register bit
TRAP	Software trap
XC	Conditional execute
XORM	XOR data memory with long constant

6.8 命令ブロックのリピート

リピート ブロック命令は、コードのブロックをN+1回リピートするために使用します。Nはブロック リピート カウンタ レジスタ(BRC)にロードされる値です。シングル リピート命令を実行する場合すべてのマスカブル割り込みがディセーブルになるのとは異なり、ブロック リピート命令の実行では、割り込みが可能です。

この実行に使われる命令は、RPTBおよびRPTBD(遅延命令)です。RPTB命令は4サイクルで実行します。RPTBDによって、RPTBDに続く2ワード命令ひとつまたは1ワード命令ふたつを、パイプラインをフラッシュする代わりに実行できるので、RPTBDは実質的に2サイクルで実行します。RPTBD命令を使うとき、遅延命令はRPTBD命令に続く2ワードには使用できません。

リピート ブロック機能により、オーバーヘッド無しのループが可能になります。オーバーヘッド無しのループは、ST1のブロック リピート アクティブ フラグ(BRAF)および次のメモリ マップド レジスタによって制御されます。

- ❑ BRCは値Nを保持し、Nはブロック リピート回数より1少なくなります。
- ❑ ブロック リピート開始アドレス(RSA)レジスタは、リピートされるコード ブロックの最初の命令のアドレスを保持します。
- ❑ ブロック リピート終了アドレス(REA)レジスタは、リピートされるコード ブロックの最後の命令ワードのアドレスを保持します。

ブロック リピートを実行中に、BRAFは1にセットされます。ブロック リピート機能は、リピート回数が0より大きいときに命令ブロックをリピートします。次の手順でループが開始します。

手順1: BRCに、0から65 535の範囲のループ回数−1の値をロードします。

手順2: 命令は、リピートされる最初の命令のアドレスをロードします。この命令は、RPTBの直後の命令、もしくはRPTBDに続く3番目のワードになります。リピート ブロック(RPTB)または遅延をともなうリピート ブロック(RPTBD)命令は、RPTB命令に続く命令のアドレス、またはRPTBDに続く3番目のワードのアドレスを、自動的にRSAにロードします。

手順3: 命令は、ブロックでリピートされる最後のワードのアドレスを、REAにロードします。これは、命令に与えられたロング イミディエイト オペランドにもなります。REAは、RPTBまたはRPTBD命令の16ビット イミディエイト オペランドでロードされ、同時にBRAFビットがセットされます。このRPTBまたはRPTBD命令の16ビット イミディエイト オペランドはL−1で、Lはリピートされる最後のワードの次のアドレスになります。

ループ実行中はPCが更新されるたびに、REAがPCの値と比較されます。値が等しければ、BRCがデクリメントされます。BRCが0以上の場合は、RSAがPCにロードされループを再スタートします。BRAFが0にリセットされ、プロセッサはループの次の命令から実行を再開します。

リピート ブロック内の最後の命令のデコード フェーズで、BRCがデクリメントされます。このため、ループ内でSRCCD命令を使うときは注意が必要です。ループ カウンタの現在の値(前にデクリメントされたBRC)を保存するには、SRCCD命令は必ずブロックの最後の最低3命令前に置かれなければなりません。

ブロック リピート レジスタはひとつだけなので、複数のブロック リピートはループ外側のコンテキストを保存しなければネストできません。ネスト ループを確立する最も簡単な方法は、最も内側のループだけのためにRPTB[D]命令を使って、それより外のループすべてにBANZ[D]を使います。

6.9 リセット動作

リセット($\overline{RS}$)はノンマスカブルの外部割り込みのひとつで、'54xを既知の状態にするときにいつでも使用できます。電源投入後の正しいシステム動作には、複数クロック サイクルの$\overline{RS}$入力が不可欠です。$\overline{RS}$信号の入力によりデータ、アドレス、制御ラインが適切に設定されます。$\overline{RS}$信号が解除された後、約5クロック サイクルで、プロセッサはFF80hの命令をフェッチしてコードの実行を開始します。リセットの手順については、ページ10−22、セクション10.5起動アクセス シーケンスを参照してください。

$\overline{RS}$信号を受けると、次の動作が発生します。

- ❑ IPTRが1FFhにセットされます。
- ❑ 外部から$\overline{RS}$が解除されます。
- ❑ PMSTのMP/$\overline{MC}$ビットにMP/$\overline{MC}$ピンの値がセットされます。
- ❑ PCがFF80hにセットされます。
- ❑ XPCがクリアされます(一部の拡張アドレスを用いたデバイス)。
- ❑ MP/$\overline{MC}$の状態にかかわらず、FF80hがアドレス バス上にドライブされます。
- ❑ データ バスがハイ インピーダンス状態になります。
- ❑ 制御ラインがインアクティブ状態になります。
- ❑ $\overline{IACK}$信号が生成されます。
- ❑ INTMが1にセットされ、すべてのマスカブル割り込みをディセーブルにします。
- ❑ IFRがクリアされ、割り込みフラグがクリアされます。
- ❑ シングル リピート カウンタ(RC)がクリアされます。
- ❑ ペリフェラルを初期化します。
- ❑ 次のステータス ビットがそれぞれの初期値に設定されます。

■ ARP = 0	■ CLKOFF = 0	■ HM = 0	■ SXM = 1
■ ASM = 0	■ CMPT = 0	■ INTM = 1	■ TC = 1
■ AVIS = 0	■ CPL = 0	■ OVA = 0	■ XF = 1
■ BRAF = 0	■ DP = 0	■ OVB = 0	
■ C = 1	■ DROM = 0	■ OVLY = 0	
■ C16 = 0	■ FRCT = 0	■ OVM = 0	

注 :

1) 残りのステータス ビットは、初期化されません。ユーザ コードでそれらを適切に初期化する必要があります。

2) リセットは、スタック ポインタ(SP)を初期化しません。ユーザ コードでそれを初期化する必要があります。

3) MP/$\overline{MC}$=0の場合は、デバイスはコードの実行をオンチップROMから開始します。そうでない場合は、コードの実行を外部プログラム メモリから始めます。

6.10 割り込み

割り込みは、ハードウェアまたはソフトウェアにより発生させることができ、'54xがメインのプログラムを一時停止して、割り込みサービス ルーチン(ISR)と呼ばれる別のファンクションを呼び出します。一般的に、'54xにデータを与える、または'54xからデータを受ける必要のあるハードウェア デバイスによって割り込みが発生します(ADC、DACその他のプロセッサなど)。また割り込みを使って、特定のイベント発生(タイマのカウント終了など)の信号を発生することもできます。

'54xは、ソフトウェアおよびハードウェア割り込みの双方をサポートします。

- ソフトウェア割り込みは、プログラム命令(INTR、TRAPまたはRESET)によって発生できます。
- ハードウェア割り込みは、外部からの入力信号によって要求され、次の2タイプがあります。
  - 外部ハードウェア割り込みは、外部割り込みピンの信号によりトリガされます。
  - 内部ハードウェア割り込みは、オンチップ ペリフェラルからの信号によりトリガされます。

複数のハードウェア割り込みが同時にトリガされるとき、'54xは優先順位に応じてこれらのサービスを行います。ハードウェア割り込みの優先順位については、ページ6−37、サブセクション6.10.10割り込みテーブルに示される、'54xのデバイスの表を参照してください。優先順位は1の方が優先になります。

'54xの各割り込み(ハードウェアおよびソフトウェアの双方)は、次のいずれかのカテゴリに含まれます。

- マスカブル割り込み。これらはハードウェアまたはソフトウェア割り込みで、ソフトウェアを使ってディセーブル(マスク)またはイネーブル(ノンマスク)にできます。'54xは、最大16のユーザ マスカブル割り込み(SINT15−SINT0)をサポートできます。各デバイスはこれら16の割り込みの一部を使用します。たとえば、'541はこれらの割り込みのうち9種類を使用します(その他は内部でハイ レベルに固定)。これらの割り込みの一部には2つの名前を有するものがあり、ソフトウェアまたはハードウェアによってドリガできます。'541では、これらの割り込みのハードウェア名は次のようになります。
  - $\overline{\text{INT3}}$から$\overline{\text{INT0}}$
  - RINT0、XINT0、RINT1、XINT1(シリアル ポート割り込み)
  - TINT
- ノンマスカブル割り込み。これらの割り込みは、マスク(ディセーブル)することはできません。'54xは常にこれらの割り込みに応答して、メインのプログラムからISRに分岐します。'54xのノンマスカブル割り込みは、ソフトウェア割り込み、および2種類のハードウェア割り込みの$\overline{\text{RS}}$および$\overline{\text{NMI}}$が含まれます($\overline{\text{RS}}$および$\overline{\text{NMI}}$は、ソフトウェアを使っても駆動できます)。

$\overline{RS}$はノンマスカブル割り込みで、どのような'54xの動作モードにも影響を与えます(6−25ページのセクション6.9リセット動作を参照)。$\overline{NMI}$は、ノンマスカブル割り込みです。割り込みは、$\overline{NMI}$が駆動されると全ての割り込みがディセーブルになります。$\overline{NMI}$は$\overline{RS}$とは異なり、'54xのモードに影響を与えません。

'54xは割り込みを3つの段階で処理します。

1) 割り込み要求を受け付けます。メインのプログラムの停止がソフトウェア(プログラム コード)またはハードウェア(ピンまたはオンチップ ペリフェラル)から要求されます。割り込み元がマスカブル割り込みの場合は、割り込みが受け付けられると割り込みフラグ レジスタ(IFR)の対応するビットがセットされます。

2) 割り込みに応答します。'54xは必ず割り込み要求に応答します。割り込みがマスカブルの場合は、'54xが割り込みに応答するために、あらかじめ決められた条件を満足する必要があります。ノンマスカブルのハードウェア割り込みおよびソフトウェア割り込みでは、応答は直ちに行われます。

3) 割り込みサービス ルーチン(ISR)を実行します。割り込みに応答すると、あらかじめ決められたアドレス(ベクタ位置)に置かれた分岐命令を'54xが実行して、ISRを実行します。

6.10.1 割り込みフラグ レジスタ(IFR)

IFRはメモリ マップド レジスタで、アクティブな割り込みを特定できます(図6−2参照)。割り込みは、それがCPUに認識されるまで対応する割り込みフラグをIFRに保持します。次の4つのイベントのいずれもが、割り込みフラグをクリアします。

- '54xがリセットされた時($\overline{RS}$がロー)。
- 割り込みトラップが取られた時。
- IFRの適切なビットに1がライトされた時。
- INTR命令が、適切な割り込み番号を使って実行された時。

どのIFRビットの1も、割り込みの保持を表します。割り込みをクリアするには、割り込みが対応するIFRのビットに1をライトします。IFRの内容をクリアするためには、IFRの現在の内容と同じ値をライトすることでクリアできます。

図6−2. **割り込みフラグ レジスタ**(IFR)**図**

(a) ’541 IFR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	Resvd	INT3	XINT1	RINT1	XINT0	RINT0	TINT	INT2	INT1	INT0

(b) ’542 IFR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	HPINT	INT3	TXINT	TRINT	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(c) ’543 IFR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	Resvd	INT3	TXINT	TRINT	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(d) ’545 IFR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	HPINT	INT3	XINT1	RINT1	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(e) ’546 IFR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	Resvd	INT3	XINT1	RINT1	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(f) ’548 IFR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	BXINT1	BRINT1	HPINT	INT3	TXINT	TRINT	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(g) ’549 IFR

15−14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	BMINT1	BMINT0	BXINT1	BRINT1	HINT	$\overline{\text{INT3}}$	TXNT	TRNT	BXINT0	TRINT0	TINT	$\overline{\text{INT2}}$	$\overline{\text{INT1}}$	$\overline{\text{INT0}}$

6.10.2 割り込みマスク レジスタ(IMR)

図6−3は、’54xが外部および内部割り込みのマスキングのためにメモリ マップド割り込みマスク レジスタ(IMR)をどのように使用するかを示しています。ST1でINTM＝0の場合、IMR中のビットを1にすることで、対応する割り込みをイネーブルにします。$\overline{\text{NMI}}$または$\overline{\text{RS}}$のいずれもIMRには含まれませんが、これはIMRはこれらの割り込みに影響しないからです。IMRへはリード/ライトが可能です。

図6−3. 割り込みマスク レジスタ(IMR)図

(a) ’541 IMR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	Resvd	INT3	XINT1	RINT1	XINT0	RINT0	TINT	INT2	INT1	INT0

(b) ’542 IMR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	HPINT	INT3	TXINT	TRINT	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(c) ’543 IMR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	Resvd	INT3	TXINT	TRINT	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(d) ’545 IMR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	HPINT	INT3	XINT1	RINT1	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(e) ’546 IMR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	Resvd	INT3	XINT1	RINT1	BXINT0	BRINT0	TINT	INT2	INT1	INT0

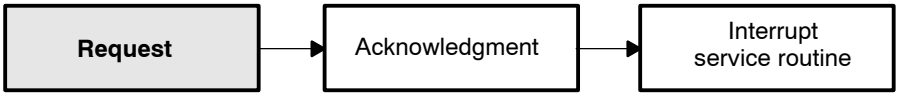
(f) ’548 IMR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	BXINT1	BRINT1	HPINT	INT3	TXINT	TRINT	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(g) ’549 IMR

15−14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	BMINT1	BMINT0	BXINT1	BRINT1	HINT	$\overline{\text{INT3}}$	TXNT	TRNT	BXINT0	TRINT0	TINT	$\overline{\text{INT2}}$	$\overline{\text{INT1}}$	$\overline{\text{INT0}}$

6.10.3 段階1:割り込み要求の受け取り



割り込みは、ハードウェア デバイスまたはソフトウェア命令によって要求されます。割り込み要求が発生すると、対応するフラグがIFRでアクティブになります(6−27ページ、サブセクション6.10.1割り込みフラグ レジスタ(IFR)を参照)。このフラグは、割り込みがプロセッサによって応答されてもされなくても、アクティブになります。このフラグは、対応する割り込みが受け付けられると、自動的にクリアされます。

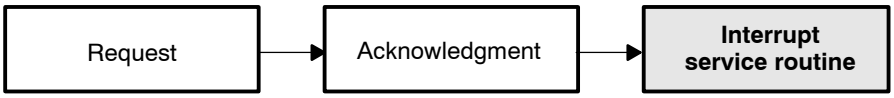
- ハードウェア割り込み要求。外部からのハードウェア割り込みは、外部割り込みポートからの信号によって要求され、内部ハードウェア割り込みはオンチップ ペリフェラルからの信号によって要求されます。たとえば'541では、ハードウェア割り込みは次の項目によって要求されます。
  - ピン$\overline{\text{INT3}}$から$\overline{\text{INT0}}$
  - ピン$\overline{\text{RS}}$(リセット)および$\overline{\text{NMI}}$
  - シリアル ポート割り込み(RINT0およびXINT0、またはRINT1およびXINT1)
  - タイマ割り込み(TINT)

表6−18から表6−24(6−37ページから6−43ページ)は、各'54xデバイスの割り込み元を示しています。

- ソフトウェア割り込み要求。ソフトウェア割り込みは、次のプログラム命令のいずれかによって要求されます。
  - INTR。この命令によって、いかなる割り込みサービス ルーチンも実行できます。割り込みオペランド(K)は、CPUがどの割り込みベクタ位置に分岐するかを示します。表6−18から表6−24(6−37ページから6−43ページ)は、各ベクタ位置の参照に使われるオペランドKを示しています。INTR割り込みが応答されると、ST1の割り込みモード ビット(INTM)が1に設定されマスカブル割り込みをディセーブルにします。
  - TRAP。この命令は、INTMビットを設定することなくINTR命令と同じファンクションを実行します。表6−18から表6−24(6−37ページから6−43ページ)は、各ベクタ位置の参照に使われるオペランドKを示しています。
  - RESET。この命令は、'54xをあらかじめ定められた状態にするためにいつでも使用できるノンマスカブル ソフトウェア リセットを実行します。RESET命令はST0およびST1に影響しますが、PMSTには影響しません。影響されるレジスタおよびビットについては、TMS320C54x DSP Reference Set Volume 2: Mnemonic Instraction Set、またはVolum 3: Algebraic Instruction SetのRESET命令を参照してください。

RESET命令が応答されると、INTMが1に設定されてマスカブル割り込みをディセーブルにします。IPTRおよびペリフェラル レジスタの初期化は、ハードウェア リセットによって行われる初期化とは異なります(6−25ページ、セクション6.9リセット動作を参照)。

6.10.4 段階2:割り込みの応答



ハードウェアまたはソフトウェアによって割り込みが要求された後、CPUはその要求に応答するかどうかを決めます。ソフトウェア割り込みおよびノンマスカブルのハードウェア割り込みは、直ちに応答されます。マスカブル ハードウェア割り込みは、以下の決められた条件が満たされたときのみ応答されます。

- 優先順位が最高位。複数のハードウェア割り込みが同時に要求されると、'54xは優先順位ランクの設定に従って割り込みサービスを行います。ランクは、1が最高順位を示します。表6−18から表6−24(6−37ページから6−43ページ)は、ハードウェア割り込みの優先順位を示しています。

- INTMビットが0。ST1の割り込みモード ビット(INTM)は、すべてのマスカブル割り込みをイネーブルまたはディセーブルにします。
  - INTM=0のとき、IMRでマスクされているすべての割り込みがイネーブルになります。
  - INTM=1のとき、IMRでマスクされているすべての割り込みがディセーブルになります。

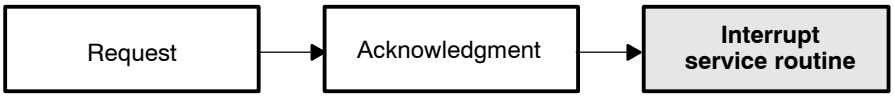
割り込みを受け付けると、INTMビットが自動的に1にセットされます。プログラムがRETE命令(割り込み許可をともなう割り込みからのリターン)を使って割り込みサービスルーチン(ISR)を終了すると、INTMが再びイネーブルになります(クリア)。INTMは、ハードウェア リセット($\overline{RS}$)によって、またはSSBX INTM命令(ディセーブル割り込み)を実行することによって設定できます。INTMは、RSBX INTM命令(イネーブル割り込み)を実行することによりリセットされます。INTMは、IMRまたはIFRを修飾しません。

- IMRマスク ビットが1。各マスカブル割り込みには、それぞれのマスク ビットがIMRにあります。割り込みをイネーブルにするには、マスク ビットを1にセットします。(6−29ページ、サブセクション6.10.2割り込みマスク レジスタ(IMR)を参照。)

CPUは、マスカブル ハードウェア割り込みに応答すると、命令バスにINTR命令を送出します。この命令はPCを適切なアドレスに変更して、ソフトウェア ベクタをフェッチします。CPUはソフトウェア ベクタの最初のワードをフェッチするとき$\overline{IACK}$信号を発生し、この信号は適切な割り込みフラグ ビットをクリアします。

イネーブルの割り込みでは、$\overline{\text{IACK}}$が発生すると、割り込み番号がCLKOUT立ち上がりでアドレス ビットA6−A2によって出力されます。割り込みベクタがオンチップ メモリにありそのアドレスを参照したい場合は、'54xをアドレス可視モード(AVIS=1)で動作することで、割り込み番号をデコードできます。'54xがホールド状態でかつHM=0のとき、$\overline{\text{IACK}}$がアクティブになってもアドレスは出力されません。

6.10.5 段階3:割り込みサービス ルーチン(ISR)の実行



割り込みに応答した後、CPUは次の動作を行います。

1) プログラム カウンタ(PC)の値(リターン アドレス)をデータ メモリのスタックの最上段にストアします。
2) 割り込みベクタのアドレスをPCにロードします。
3) ベクタ アドレスの命令をフェッチします。(その命令が遅延分岐命令の場合は、その後に続く2ワード命令ひとつか、1ワード命令ふたつを、さらにフェッチします。)
4) 対応するISRの先頭に分岐を実行します。(分岐が遅延をともなうときは、その命令に続く命令が分岐の前に実行されます。)
5) ISRがリターン命令で終わるまでISRを実行します。
6) スタック ポインタ(SP)をスタック最上段にし、リターン アドレスをスタックからPCにポップします。
7) メインプログラムを再開します。

どのベクタ アドレスが各割り込みに割り当てられているかは、ページ6−37のサブセクション6.10.10割り込みテーブルの各'54xに対応した表を参照してください。各割り込みベクタは4ワードあり、例えば遅延分岐命令およびふたつの1ワード命令またはひとつの2ワード命令を置くことができます。

6.10.6 割り込みコンテキスト セーブ

割り込みサービス ルーチンが実行されるとき、特定のレジスタをスタック上に保存する必要があります。プログラムがISRから復帰するとき(RC[D]、RETE[D]、またはRETF[D]によって)、ユーザのソフトウェア コードでこれらのレジスタの内容を必ず復元しなければなりません。スタック ポインタがメモリ容量を超えない様に、スタックを管理しなければいけません。このスタックは、サブルーチン コールにも使用できます。'54xはサブルーチン コールをISR内で許可します。CPUレジスタおよびペリフェラルレジスタはメモリ マッピングされているので、PSHMおよびPOPM命令を使ってこれらのレジスタをスタックとの間で転送できます。さらにPSHDおよびPOPD命令は、データ メモリの値をスタックとの間で転送できます。

コンテキスト セーブおよび復元を行う際に、考慮すべき点がいくつかあります。最初の注意点は、スタックを使ってコンテキスト セーブを行うとき、スタックした順番と正反対の順序で復元を実行する必要があることです。もう一つの注意点は、ST1のBRAFビットの復元の前にBRCを復元する必要があることです。この順番に従えない場合は、BRCが復元される前にBRC=0のときBRAFビットビットがクリアされます。

6.10.7 割り込みレイテンシ

'54xは、割り込みを実行する前に、プリフェッチおよびフェッチ ステージの命令を除く、パイプライン中のすべての命令を完了するので、最大割り込みレイテンシはパイプラインの内容によります。割り込みに関連するパイプライン レイテンシの詳細については、7−26ページ、セクション7.2割り込み及びパイプラインを参照してください。低速メモリ アクセスのために、ウェイト ステートによって数サイクル必要な命令およびリピートされる命令は、割り込み処理のために余分なサイクルを必要とします。

シングル リピート命令(RPTおよびRPTZ)では、リピート命令のコンテキストを保護するために、その命令に続く次の命令のすべての実行を完了してから、割り込みを受け付けます。これは、これらの命令がパイプラインで並列に実行されているからです。

ホールド機能は割り込みより優先されるので、割り込みトラップを遅延できます。CPUがホールド状態にあり、割り込みベクタが外部メモリからフェッチされる必要があるときに割り込みが発生した場合、$\overline{\text{HOLDA}}$がインアクティブになるまで(ホールド状態が終わるまで)割り込みは受け付けられません。しかし、プロセッサがコンカレント ホールド モード(HM=0)で割り込みベクタ テーブルが内部メモリにあるとき、$\overline{\text{HOLD}}$にかかわりなくCPUは割り込みを受け付けます。

割り込みは、プログラム シーケンス上で、“RSBX INTM”命令とその次の命令の間では処理されません。割り込みがRSBX INTMのデコード ステージで発生した場合、CPUは保留された割り込みを処理する前に、常にRSBX INTMと次の命令を完了します。これらの命令が完了するのを待つことにより、次の割り込みの処理が行われる前に、リターン(RET)を確実に実行できるようにしています。ISRがRETE命令(イネーブルをともなうISRからのリターン)によって終了する場合、RSBX INTM命令は不要です。RSBX INTM命令と同様に、SSBX INTM命令およびそれに続く命令に割り込みはできません。

注:

リセット($\overline{\text{RS}}$)は、マルチサイクル命令によって遅延されません。$\overline{\text{NMI}}$は、マルチサイクル命令および$\overline{\text{HOLD}}$によって遅延されません。

6.10.8 割り込み処理:要約

割り込みがCPUに渡されると、CPUは次のような処理を行います(6−36ページ図6−5参照)。

- マスカブル割り込みが要求される場合。
  1) IFRの対応するビットがセットされます。
  2) 応答条件(INTM=0およびIMRビット=1)がテストされます。条件が真(true)の場合は、CPUは割り込みに応答して$\overline{\text{IACK}}$シグナルを生成します。そうでない場合は、割り込みを無視してメインのプログラムを続行します。
  3) 割り込みに応答すると、IFRにあるそのフラグ ビットが0にクリアされ、INTMビットが1にセットされます(他のマスカブル割り込みをブロックします)。
  4) PCがスタック上に保存されます。
  5) CPUが割り込みサービス ルーチン(ISR)に分岐してそれを実行します。
  6) ISRが復帰命令によって終了し、その復帰命令はスタックから復帰アドレスをポップします。
  7) CPUはメインのプログラムを続行します。
- ノンマスカブル割り込みが要求される場合。
  1) CPUは直ちに割り込みに応答して、$\overline{\text{IACK}}$シグナルを生成します。
  2) 割り込みが$\overline{\text{RS}}$、$\overline{\text{NMI}}$、またはINTR命令によって要求された場合、INTMビットは1にセットされマスカブル ハードウェア割り込みをブロックします。

3) INTR命令がマスカブル割り込みのひとつを要求した場合は、対応するフラグ ビットが0にクリアされます。
4) PCがスタック上にプッシュされます。
5) CPUがISRに分岐してISRを実行します。
6) ISRは復帰命令により終了し、その復帰命令はスタックの復帰アドレスをポップします。
7) CPUはメインのプログラムを続行します。

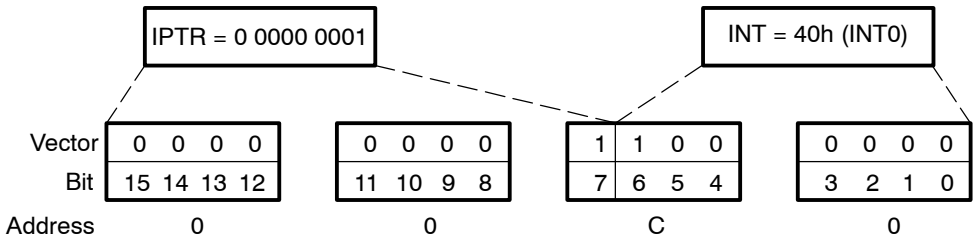
注 :

INTR命令は、割り込みモード ビット(INTM)をセットすることによりマスカブル割り込みをディセーブルにしますが、TRAP命令はINTMの影響を受けません。

6.10.9 割り込みベクタ アドレスの再マッピング

割り込みベクタは、予約領域を除くプログラム メモリの128ワードで区切られたいずれのページの先頭にも再マッピングできます。割り込みベクタ アドレスは、PMSTの割り込みポインタ(IPTR)フィールドを、2ビット左シフトした割り込みベクタ番号(0−31)と連結することで生成します。図6−4の例を見てみると、$\overline{\text{INT0}}$がローにドライブされIPTR=0001hのとき、割り込みベクタは00C0hからフェッチされます。$\overline{\text{INT0}}$の割り込みベクタ番号は、16または10hになります。

図6−4. 割り込みベクタ アドレス生成

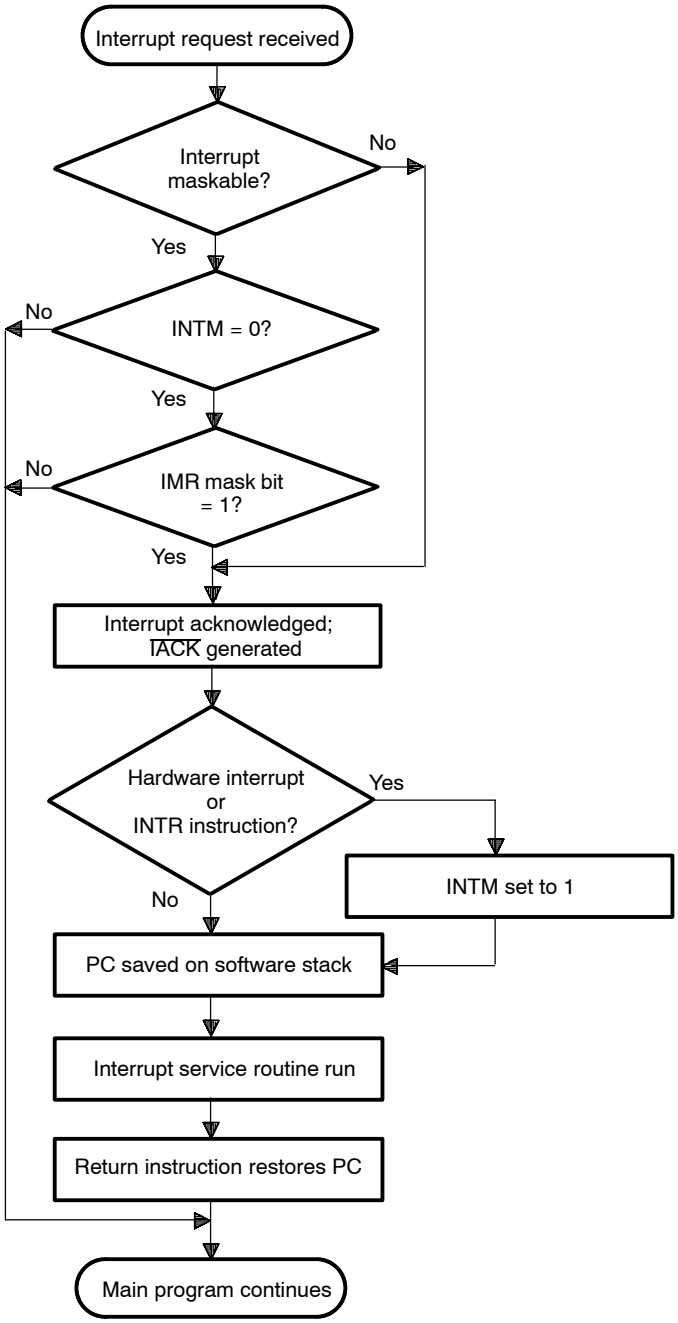


リセット時に、IPTRの全てのビットが1にセットされます(IPTR=1FFh)。この値はベクタをプログラム メモリ領域の511x80hにマッピングします。したがって、ハードウェア リセットのリセット ベクタは常に0FF80hにあります。この割り込みベクタは、1FFh以外の値を、IPTRを設定することで、別の位置にマッピングできます。たとえば、0001hをIPTRに設定することにより、割り込みベクタを0080hの位置にベクタ テーブルの先端がくる様に配置することができます。

注 :

ハードウェア リセット($\overline{\text{RS}}$)ベクタは再マッピングできません。これは、ハードウェア リセットがIPTRに1ffhを設定するからです。したがって、ハードウェア リセットのリセット ベクタは、常にプログラム空間のFF80hの位置に置かれます。

図6−5. 割り込み処理の流れ図



6.10.10 割り込みテーブル

表6−18から表6−24は、各'54xデバイスの割り込みトラップ番号、優先順位、位置を示しています。

表6−18. TMS320C541割り込み位置および優先順位

TRAP/INTR Number (K)	Priority	Name	Location	Function
0	1	$\overline{RS}$/SINTR	0	Reset (hardware and software reset)
1	2	$\overline{NMI}$/SINT16	4	Nonmaskable interrupt
2	–	SINT17	8	Software interrupt #17
3	–	SINT18	C	Software interrupt #18
4	–	SINT19	10	Software interrupt #19
5	–	SINT20	14	Software interrupt #20
6	–	SINT21	18	Software interrupt #21
7	–	SINT22	1C	Software interrupt #22
8	–	SINT23	20	Software interrupt #23
9	–	SINT24	24	Software interrupt #24
10	–	SINT25	28	Software interrupt #25
11	–	SINT26	2C	Software interrupt #26
12	–	SINT27	30	Software interrupt #27
13	–	SINT28	34	Software interrupt #28
14	–	SINT29	38	Software interrupt #29; reserved
15	–	SINT30	3C	Software interrupt #30; reserved
16	3	$\overline{INT0}$/SINT0	40	External user interrupt #0
17	4	$\overline{INT1}$/SINT1	44	External user interrupt #1
18	5	$\overline{INT2}$/SINT2	48	External user interrupt #2
19	6	TINT/SINT3	4C	Internal timer interrupt
20	7	RINT0/SINT4	50	Serial port 0 receive interrupt
21	8	XINT0/SINT5	54	Serial port 0 transmit interrupt
22	9	RINT1/SINT6	58	Serial port 1 receive interrupt
23	10	XINT1/SINT7	5C	Serial port 1 transmit interrupt
24	11	$\overline{INT3}$/SINT8	60	External user interrupt #3
25−31	–		64−7F	Reserved

表6−19. TMS320C542割り込み位置および優先順位

TRAP/INTR Number (K)	Priority	Name	Location	Function
0	1	$\overline{RS}$/SINTR	0	Reset (hardware and software reset)
1	2	$\overline{NMI}$/SINT16	4	Nonmaskable interrupt
2	–	SINT17	8	Software interrupt #17
3	–	SINT18	C	Software interrupt #18
4	–	SINT19	10	Software interrupt #19
5	–	SINT20	14	Software interrupt #20
6	–	SINT21	18	Software interrupt #21
7	–	SINT22	1C	Software interrupt #22
8	–	SINT23	20	Software interrupt #23
9	–	SINT24	24	Software interrupt #24
10	–	SINT25	28	Software interrupt #25
11	–	SINT26	2C	Software interrupt #26
12	–	SINT27	30	Software interrupt #27
13	–	SINT28	34	Software interrupt #28
14	–	SINT29	38	Software interrupt #29, reserved
15	–	SINT30	3C	Software interrupt #30, reserved
16	3	$\overline{INT0}$/SINT0	40	External user interrupt #0
17	4	$\overline{INT1}$/SINT1	44	External user interrupt #1
18	5	$\overline{INT2}$/SINT2	48	External user interrupt #2
19	6	TINT/SINT3	4C	Internal timer interrupt
20	7	BRINT0/SINT4	50	Buffered serial port receive interrupt
21	8	BXINT0/SINT5	54	Buffered serial port transmit interrupt
22	9	TRINT/SINT6	58	TDM serial port receive interrupt
23	10	TXINT/SINT7	5C	TDM serial port transmit interrupt
24	11	$\overline{INT3}$/SINT8	60	External user interrupt #3
25	12	$\overline{HPINT}$/SINT9	64	HPI interrupt
26–31	–		68–7F	Reserved

表6−20. TMS320C543割り込み位置および優先順位

TRAP/INTR Number (K)	Priority	Name	Location	Function
0	1	$\overline{RS}$/SINTR	0	Reset (hardware and software reset)
1	2	$\overline{NMI}$/SINT16	4	Nonmaskable interrupt
2	–	SINT17	8	Software interrupt #17
3	–	SINT18	C	Software interrupt #18
4	–	SINT19	10	Software interrupt #19
5	–	SINT20	14	Software interrupt #20
6	–	SINT21	18	Software interrupt #21
7	–	SINT22	1C	Software interrupt #22
8	–	SINT23	20	Software interrupt #23
9	–	SINT24	24	Software interrupt #24
10	–	SINT25	28	Software interrupt #25
11	–	SINT26	2C	Software interrupt #26
12	–	SINT27	30	Software interrupt #27
13	–	SINT28	34	Software interrupt #28
14	–	SINT29	38	Software interrupt #29, reserved
15	–	SINT30	3C	Software interrupt #30, reserved
16	3	$\overline{INT0}$/SINT0	40	External user interrupt #0
17	4	$\overline{INT1}$/SINT1	44	External user interrupt #1
18	5	$\overline{INT2}$/SINT2	48	External user interrupt #2
19	6	TINT/SINT3	4C	Internal timer interrupt
20	7	BRINT0/SINT4	50	Buffered serial port receive interrupt
21	8	BXINT0/SINT5	54	Buffered serial port transmit interrupt
22	9	TRINT/SINT6	58	TDM serial port receive interrupt
23	10	TXINT/SINT7	5C	TDM serial port transmit interrupt
24	11	$\overline{INT3}$/SINT8	60	External user interrupt #3
25−31	–		64−7F	Reserved

表6−21. TMS320C545割り込み位置および優先順位

TRAP/INTR Number (K)	Priority	Name	Location	Function
0	1	$\overline{RS}$/SINTR	0	Reset (hardware and software reset)
1	2	$\overline{NMI}$/SINT16	4	Nonmaskable interrupt
2	–	SINT17	8	Software interrupt #17
3	–	SINT18	C	Software interrupt #18
4	–	SINT19	10	Software interrupt #19
5	–	SINT20	14	Software interrupt #20
6	–	SINT21	18	Software interrupt #21
7	–	SINT22	1C	Software interrupt #22
8	–	SINT23	20	Software interrupt #23
9	–	SINT24	24	Software interrupt #24
10	–	SINT25	28	Software interrupt #25
11	–	SINT26	2C	Software interrupt #26
12	–	SINT27	30	Software interrupt #27
13	–	SINT28	34	Software interrupt #28
14	–	SINT29	38	Software interrupt #29, reserved
15	–	SINT30	3C	Software interrupt #30, reserved
16	3	$\overline{INT0}$/SINT0	40	External user interrupt #0
17	4	$\overline{INT1}$/SINT1	44	External user interrupt #1
18	5	$\overline{INT2}$/SINT2	48	External user interrupt #2
19	6	TINT/SINT3	4C	Internal timer interrupt
20	7	BRINT0/SINT4	50	Buffered serial port receive interrupt
21	8	BXINT0/SINT5	54	Buffered serial port transmit interrupt
22	9	RINT1/SINT6	58	Serial port receive interrupt
23	10	XINT1/SINT7	5C	Serial port transmit interrupt
24	11	$\overline{INT3}$/SINT8	60	External user interrupt #3
25	12	$\overline{HPINT}$/SINT9	64	HPI interrupt
26−31	–		68−7F	Reserved

表6−22. TMS320C546割り込み位置および優先順位

TRAP/INTR Number (K)	Priority	Name	Location	Function
0	1	$\overline{RS}$/SINTR	0	Reset (hardware and software reset)
1	2	$\overline{NMI}$/SINT16	4	Nonmaskable interrupt
2	–	SINT17	8	Software interrupt #17
3	–	SINT18	C	Software interrupt #18
4	–	SINT19	10	Software interrupt #19
5	–	SINT20	14	Software interrupt #20
6	–	SINT21	18	Software interrupt #21
7	–	SINT22	1C	Software interrupt #22
8	–	SINT23	20	Software interrupt #23
9	–	SINT24	24	Software interrupt #24
10	–	SINT25	28	Software interrupt #25
11	–	SINT26	2C	Software interrupt #26
12	–	SINT27	30	Software interrupt #27
13	–	SINT28	34	Software interrupt #28
14	–	SINT29	38	Software interrupt #29, reserved
15	–	SINT30	3C	Software interrupt #30, reserved
16	3	$\overline{INT0}$/SINT0	40	External user interrupt #0
17	4	$\overline{INT1}$/SINT1	44	External user interrupt #1
18	5	$\overline{INT2}$/SINT2	48	External user interrupt #2
19	6	TINT/SINT3	4C	Internal timer interrupt
20	7	BRINT0/SINT4	50	Buffered serial port receive interrupt
21	8	BXINT0/SINT5	54	Buffered serial port transmit interrupt
22	9	RINT1/SINT6	58	Serial port receive interrupt
23	10	XINT1/SINT7	5C	Serial port transmit interrupt
24	11	$\overline{INT3}$/SINT8	60	External user interrupt #3
25−31	–		64−7F	Reserved

表6−23. TMS320C548割り込み位置および優先順位

TRAP/INTR Number (K)	Priority	Name	Location	Function
0	1	$\overline{RS}$/SINTR	0	Reset (hardware and software reset)
1	2	$\overline{NMI}$/SINT16	4	Nonmaskable interrupt
2	–	SINT17	8	Software interrupt #17
3	–	SINT18	C	Software interrupt #18
4	–	SINT19	10	Software interrupt #19
5	–	SINT20	14	Software interrupt #20
6	–	SINT21	18	Software interrupt #21
7	–	SINT22	1C	Software interrupt #22
8	–	SINT23	20	Software interrupt #23
9	–	SINT24	24	Software interrupt #24
10	–	SINT25	28	Software interrupt #25
11	–	SINT26	2C	Software interrupt #26
12	–	SINT27	30	Software interrupt #27
13	–	SINT28	34	Software interrupt #28
14	–	SINT29	38	Software interrupt #29, reserved
15	–	SINT30	3C	Software interrupt #30, reserved
16	3	$\overline{INT0}$/SINT0	40	External user interrupt #0
17	4	$\overline{INT1}$/SINT1	44	External user interrupt #1
18	5	$\overline{INT2}$/SINT2	48	External user interrupt #2
19	6	TINT/SINT3	4C	Internal timer interrupt
20	7	BRINT0/SINT4	50	Buffered serial port 0 receive interrupt
21	8	BXINT0/SINT5	54	Buffered serial port 0 transmit interrupt
22	9	TRINT/SINT6	58	TDM serial port receive interrupt
23	10	TXINT/SINT7	5C	TDM serial port transmit interrupt
24	11	$\overline{INT3}$/SINT8	60	External user interrupt #3
25	12	$\overline{HPINT}$/SINT9	64	HPI interrupt
26	13	BRINT1/SINT10	68	Buffered serial port 1 receive interrupt
27	14	BXINT1/SINT11	6C	Buffered serial port 1 transmit interrupt
28−31	–		70−7F	Reserved

表6−24. TMS320C549割り込み位置および優先順位

TRAP/INTR Number (K)	Priority	Name	Location	Function
0	1	$\overline{RS}$/SINTR	0	Reset (hardware and software reset)
1	2	$\overline{NMI}$/SINT16	4	Nonmaskable interrupt
2	–	SINT17	8	Software interrupt #17
3	–	SINT18	C	Software interrupt #18
4	–	SINT19	10	Software interrupt #19
5	–	SINT20	14	Software interrupt #20
6	–	SINT21	18	Software interrupt #21
7	–	SINT22	1C	Software interrupt #22
8	–	SINT23	20	Software interrupt #23
9	–	SINT24	24	Software interrupt #24
10	–	SINT25	28	Software interrupt #25
11	–	SINT26	2C	Software interrupt #26
12	–	SINT27	30	Software interrupt #27
13	–	SINT28	34	Software interrupt #28
14	–	SINT29	38	Software interrupt #29, reserved
15	–	SINT30	3C	Software interrupt #30, reserved
16	3	$\overline{INT0}$/SINT0	40	External user interrupt #0
17	4	$\overline{INT1}$/SINT1	44	External user interrupt #1
18	5	$\overline{INT2}$/SINT2	48	External user interrupt #2
19	6	TINT/SINT3	4C	Internal timer interrupt
20	7	BRINT0/SINT4	50	Buffered serial port 0 receive interrupt
21	8	BXINT0/SINT5	54	Buffered serial port 0 transmit interrupt
22	9	TRINT/SINT6	58	TDM serial port receive interrupt
23	10	TXINT/SINT7	5C	TDM serial port transmit interrupt
24	11	$\overline{INT3}$/SINT8	60	External user interrupt #3
25	12	$\overline{HPINT}$/SINT9	64	HPI interrupt
26	13	BRINT1/SINT10	68	Buffered serial port 1 receive interrupt
27	14	BXINT1/SINT11	6C	Buffered serial port 1 transmit interrupt
28	15	BMINT0/SINT12	70	BSP0 buffer misalignment interrupt
29	16	BMINT1/SINT13	74	BSP1 buffer misalignment interrupt
30−31	–		78−7F	Reserved

6.11 パワー ダウン モード

54xには、いくつかのパワー ダウン モードがあります。このモードではプロセッサが休止状態になり、消費電力を通常より抑えながらCPUの内容を保持します。これにより、パワー ダウン モードが終わると処理が引き続き行われます。

パワー ダウン モードにするには、IDLE1、IDLE2またはIDLE3のいずれかの命令を実行するか、HMステータス ビットを1にセットして$\overline{\text{HOLD}}$信号をローにドライブします。パワーダウン動作の概要は、表6−25を参照してください。また、以下のサブセクション6.11.1から6.11.5に説明があります。

表6−25. 4つのパワー ダウン モードの動作

Operation/Feature	IDLE1	IDLE2	IDLE3	HOLD
CPU halted	Yes	Yes	Yes	Yes†
CPU clock stopped	Yes	Yes	Yes	No
Peripheral clock stopped	No	Yes	Yes	No
Phase-locked loop (PLL) stopped	No	No	Yes	No
External address lines put in high-impedance state	No	No	No	Yes
External data lines put in high-impedance state	No	No	No	Yes
External control signals put in high-impedance state	No	No	No	Yes
Power-down terminated by:				
HOLD driven high	No	No	No	Yes
Unmasked internal hardware interrupts	Yes	No	No	No
Unmasked external hardware interrupts	Yes	Yes	Yes	No
$\overline{\text{NMI}}$	Yes	Yes	Yes	No
$\overline{\text{RS}}$	Yes	Yes	Yes	No

† HMビットの設定によっては、外部メモリ アクセスを必要としない限りは、CPUの動作が継続されます。

6.11.1 IDLE1モード

IDLE1モードは、システム クロック以外のすべてのCPUの動作を停止します。システム クロックはペリフェラル モジュールに供給されたままになるので、ペリフェラル モジュールは動作を継続してCLKOUTピンはアクティブのままになります。したがって、シリアル ポートやタイマなどのペリフェラルは、CPUをパワー ダウン状態からウェイク アップさせることができます。

IDLE1モードにするには、IDLE1命令を使います。IDLE1モードを終了するには、ウェイク アップ割り込みを使います。ウェイク アップ割り込みが発生したときINTM=0なら、'54xはIDLE1を終了してISRを実行します。INTM=1なら、'54xはIDLE1モード命令に続く

命令を続行します。すべてのウェイク アップ割り込みは、INTMの値にかかわらず、IMRレジスタの対応するビットをセットして、その割り込みをイネーブルにする必要があります。ただし、ノンマスカブル割り込みである$\overline{RS}$、$\overline{NMI}$は例外です。

6.11.2 IDLE2モード

IDLE2モードは、CPUとオンチップ ペリフェラルを停止します。このモードではオンチップ ペリフェラルが停止するので、オンチップ ペリフェラルを使って割り込みを発生してIDLE1のように'54xをウェイク アップすることはできません。ただし、デバイスの大部分は停止するので消費電力は大幅に低減します。

IDLE2モードにするには、IDLE2命令を使います。IDLE2モードを終了するには、外部割り込みピン($\overline{RS}$、$\overline{NMI}$、$\overline{INTx}$)のいづれかを少なくとも10nsの間ローであるパルスでアクティブにします。ウェイク アップ割り込みが発生したときINTM=0なら、'54xはIDLE2を終了してISRを実行します。INTM=1なら、'54xはIDLE2命令に続く命令を続行します。すべてのウェイクアップ割り込みは、INTMの値にかかわらず、IMRレジスタの対応するビットをセットして、その割り込みをイネーブルにする必要があります。IDLE2が終了したら、すべてのペリフェラルをリセットして下さい(特にこれらのペリフェラルが外部クロックで動作している場合)。

IDLE2モードにおいて$\overline{RS}$がウェイク アップ割り込みの場合、$\overline{RS}$パルスを少なくとも10nsの間ローにすることにより、リセット シーケンスをアクティブにできます。

6.11.3 IDLE3モード

IDLE3モードはIDLE2モードのように機能するとともに、PLLも停止します。IDLE3を使うと、'54xは完全に機能を停止します。このモードは、IDLE2モード以上に消費電力を低減します。さらにIDLE3の状態では、システムが'54xに消費電力を節約するために低速動作することを要求する場合、外部ピンの信号入力によりPLLを再設定できます(ただし、ソフトウェア プログラマブルなPLLを有するデバイスを除く)。

IDLE3モードにするには、IDLE3命令を使います。IDLE3モードを終了するには、外部割り込みピン($\overline{RS}$、$\overline{NMI}$、$\overline{INTX}$)のいづれかを少なくとも10nsの間ローであるパルスでアクティブにします。ウェイク アップ割り込みが発生したときINTM=0なら、IDLE3モードを終了してISRを実行します。INTM=1なら、'54xはIDLE3命令に続く命令を続行します。すべてのウェイクアップ割り込みは、INTMの値にかかわらず、IMRレジスタの対応するビットをセットして、その割り込みをイネーブルにする必要があります。IDLE3が終了したら、すべてのペリフェラルをリセットしてください(特にこれらのペリフェラルが外部クロックで動作している場合)。

IDLE3を終了するには、外部割り込みが少なくとも10nsの間ローであるパルスで、ウェイクアップ割り込みをアクティブにする必要があります。ウェイク アップ シーケンスの間に、'54xは複数の割り込みを受け入れることができます。最優先の割り込みが最初にサービスされます。PLLロックアップ時間の条件については、10−24ページ、サブセクション10.5.2IDLE3、および8.4.2ソフトウェア プログラマブルなPLL(対象デバイスは付録Dを参照)を参照してください。

IDLE3モードにおいて$\overline{RS}$がウェイク アップ割り込みの場合、$\overline{RS}$パルスを少なくとも10nsの間ローにすることによりリセット シーケンスをアクティブにできます。ただし、PLLが安定したシステム クロックを内部ロジックに供給できるようにするには、$\overline{RS}$は50μs間アクティブに保つ必要があります。

6.11.4 ホールド モード

ホールド モードもパワー ダウン モードの一種です。このモードにより、アドレス、データ、制御信号をハイ インピーダンス状態にできます。このモードはHMビットの値に応じて、CPUを停止することもできます。

$\overline{HOLD}$信号により、このパワー ダウン モードに入ります。HMの値に応じて$\overline{HOLD}$動作は異なります。HM=1なら、CPUは動作を停止してアドレス、データ、制御信号がハイ インピーダンスになり、さらに消費電力が低減します。HM=0なら、アドレス、データ、制御信号がハイ インピーダンス状態になりますが、CPUは内部の動作を続行します。システムが外部メモリ アクセスを要求しない場合は、HM=0で$\overline{HOLD}$信号を使用できます。'54xは、命令によって外部アクセスが要求されない限り、通常どおりに動作します。命令が外部アクセスを要求する場合は、'54x は$\overline{HOLD}$が解除されるまで停止します。

このモードは、オンチップ ペリフェラル(タイマ、シリアル ポートなど)の動作を停止しません。オンチップ ペリフェラルは、$\overline{HOLD}$レベルまたはHMビットの条件にかかわらず、動作を続行します。

このモードは、$\overline{HOLD}$信号がインアクティブになると終了します。

6.11.5 他のパワー ダウン機能

'54xには、パワー ダウン処理に影響する機能が他に2種類あります。外部バス オフおよびCLKOUTオフです。

- 外部バス オフによって、'54xは外部インターフェイスの内部クロックをディセーブルにでき、これによりインターフェイスを低消費電力モードにできます。

  外部インターフェイス クロックは、バンク スイッチング制御レジスタ(BSCR)のビット0を1に設定することによりディセーブルになります。リセット時に、このビットは0にクリアされて、外部インターフェイス クロックがイネーブルになります。詳しくは、10−7ページ、サブセクション10.3.2バンク スイッチング ロジックを参照してください。

- CLKOUTオフにより、'54xはソフトウェア命令を使ってCLKOUTをディセーブルにできます。PMSTのCLKOFFビットは、CLKOUTがイネーブルかディセーブルかを判断します。詳しくは、4−6ページ、サブセクション4.1.2プロセッサ モード ステータス レジスタ(PMST)を参照してください。リセット時に、CLKOUTはイネーブルになります。

第7章

パイプライン

本章では、’54xのパイプライン処理を説明するとともに、様々なレジスタをともなう処理のパイプライン レンテンシ(遅延)サイクルを示します。

Topic Page

7

7.1 パイプライン処理

’54xCPUには、6段の命令パイプラインがあります。これら6つのパイプライン ステージは互いに独立しており、これにより命令のオーバーラップ実行が可能です。与えられたサイクルの間に、1から6の異なる命令をアクティブにでき、それぞれが異なるステージで行われます。

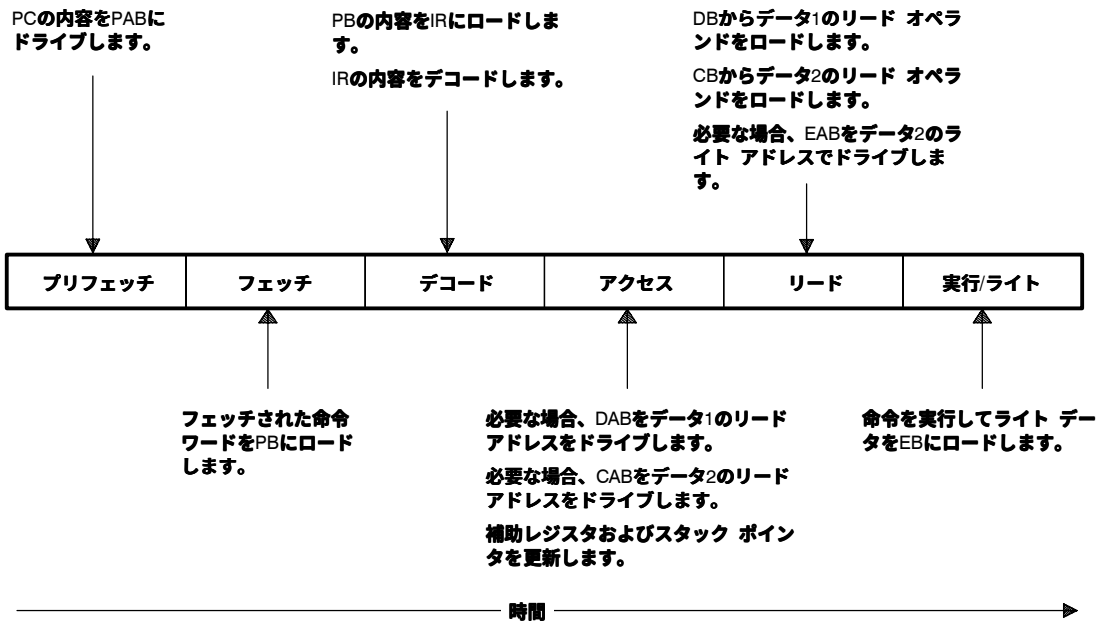
パイプライン構造の6つのスレージおよびファンクションは次の通りです。

- プログラム プリフェッチ。プログラム アドレス バス(PAB)が、次にフェッチされる命令のアドレスでドライブされます。
- プログラム フェッチ。命令ワードがプログラム バス(PB)からフェッチされ、命令レジスタ(IR)にロードされます。これは、このサイクルおよびこの前のサイクルから構成される命令フェッチ シーケンスを完了します。
- デコード。命令レジスタ(IR)の内容をデコードして、データ アドレス発生器(DAGEN)およびCPUにおけるメモリ アクセス処理および制御シーケンスのタイプを判断します。
- アクセス。DAGENがリード オペランドのアドレスをデータ アドレス バス(DAB)に出力します。2番目のオペランドが要求されるときは、もう1つのデータ アドレス バス(CAB)も適切なアドレスでドライブされます。間接アドレッシング モードの補助レジスタおよびスタック ポインタ(SP)も更新されます。アクセス ステージは、2ステージ オペランド リード シーケンスの一番目のステージと考えられます。
- リード。(もしあれば)リード データ オペランドを、データ バスDBおよびCBからリードします。リード ステージは2ステージ リード シーケンスを完了します。同時に、2ステージ オペランド ライト シーケンスが始まります。(もしあれば)ライト オペランドのデータ アドレスを、データ ライト アドレス バス(EAB)にドライブします。メモリ マップド レジスタでは、リード データ オペランドをメモリからリードして、選択されたメモリ マップド レジスタにDBを使ってライトします。
- 実行。データ ライト バス(EB)を使ってデータをライトすることで、オペランド ライト シーケンスを完了します。命令は、この段階で実行されます。

図7−1は、パイプラインの6つのステージおよび各ステージで発生するイベントを示しています。

パイプラインの最初の2ステージ、プリフェッチおよびフェッチは、命令フェッチ シーケンスです。最初のサイクルで、新しい命令のアドレスがドライブされます。次のサイクルでは命令ワードをリードします。マルチワード命令の場合は、複数のこのような命令フェッチ シーケンスが必要になります。

図7−1. **パイプライン ステージ**



パイプラインの3**番目のステージ(デコード)の間に、適正な命令実行のために適切な制御シーケンスを有効にするべく、フェッチされた命令がデコードされます。**

次の2**つのパイプライン ステージ(アクセスおよびリード)は、オペランド リード シーケンスです。命令が要求する場合、**1**つまたは**2**つのオペランドのデータ アドレスがアクセス ステージでドライブされ、そのオペランドを次のリード フェーズでリードします。**

あらゆるライト処理は、パイプラインのリードおよび実行の2**ステージにまたがります。リード ステージでは、ライト オペランドのデータ アドレスが**EAB**にドライブされます。次のサイクルでは、**EB**を使ってオペランドをメモリにライトします。**

各メモリ アクセスは、'54x**パイプラインによって**2**段階で実行されます。最初の段階では、１つのアドレス バスがそのメモリ アドレスでドライブされます。**2**番目の段階では、対応するデータがそのメモリアドレスからリード、またはメモリ アドレスにライトされます。図**7−2**は、様々なメモリ アクセスが'**54x**のパイプラインによってどのように実行されるかを示します。図のすべてのメモリ アクセスは、**1**サイクル、**1**ワード命令によって、オンチップ デュアル アクセス メモリに実行されることを前提にしています。オンチップ デュアル アクセス メモリは、**1**パイプライン サイクルに実際に**2**つのアクセスをサポートします。詳しくは、**7−28**ページ、セクション**7.3**デュアル アクセス メモリおよびパイプラインを参照してください。**

図7−2. **パイプライン メモリ アクセス**

(a) **命令ワード フェッチ**(1**サイクル**)

プリフェッチ	フェッチ	デコード	アクセス	リード	実行/ライト
PABをドライブ	PBからリード				

(b) 1**オペランド リードを実行する命令**(LD *AR1, A**など**:1**サイクル**)

プリフェッチ	フェッチ	デコード	アクセス	リード	実行/ライト
			DABをドライブ	DBからリード	

(c) **デュアル オペランド リードを実行する命令**(MAC *AR2+,AR3+, A**または**DLD *AR2,A**など**:1**サイクル**)

プリフェッチ	フェッチ	デコード	アクセス	リード	実行/ライト
			DABおよび CABをドライブ	DBおよびCB からリード	

(d) 1**オペランド ライトを実行する命令**(STH A *AR1**など**:1**サイクル**)

プリフェッチ	フェッチ	デコード	アクセス	リード	実行/ライト
				EABをドライブ	EBにライト

(e) **ロング オペランド ライトを実行する命令**(DST A *AR1**など**:2**サイクル**)

プリフェッチ	フェッチ	デコード	アクセス	リード	実行
				EABをドライブ	EBにライト

プリフェッチ	フェッチ	デコード	アクセス	リード	実行/ライト
				EABをドライブ	EBにライト

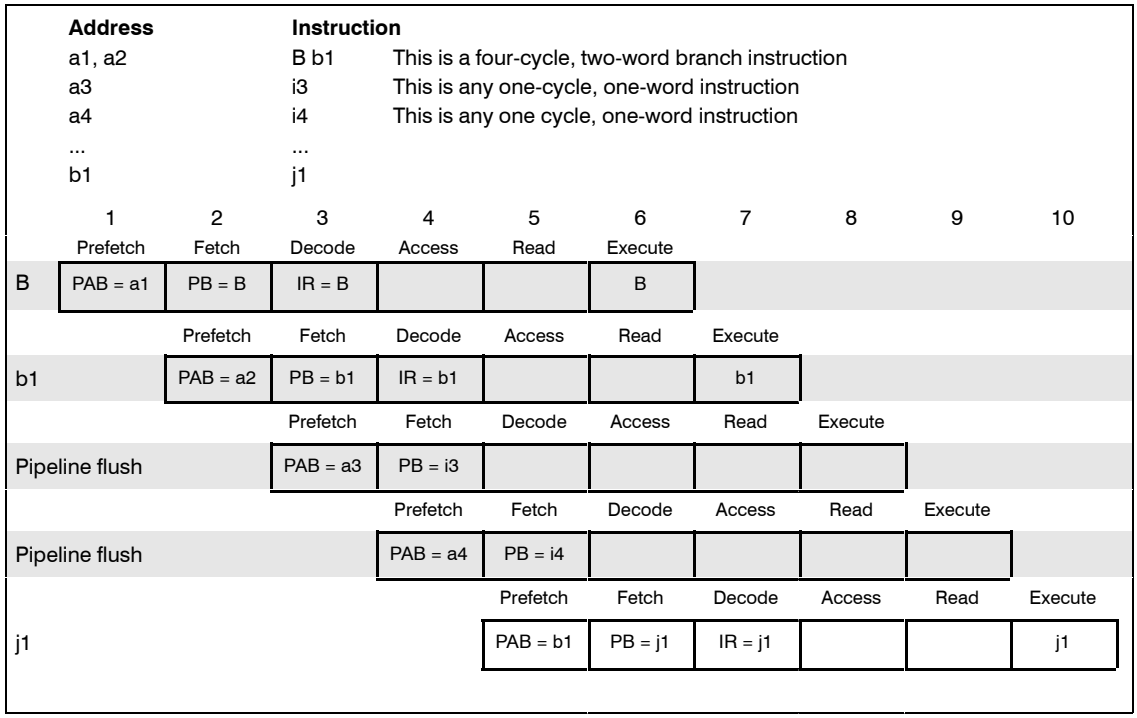
(f) **オペランド リードとオペランド ライトを実行する命令**(ST A *AR2 || LD*AR3,B**など**:1**サイクル**)

プリフェッチ	フェッチ	デコード	アクセス	リード	実行/ライト
			DABをドライブ	DBからリード、 EABをドライブ	EBにライト

以下のサブセクションでは、異なるタイプの命令を実行する間にパイプラインがどのように機能するかを示す例を説明します。特に注記がない限り、例に示されるすべての命令は1サイクル、オンチップ メモリにある1ワード命令です。

これらの例の中では、パイプラインは段差の付いた列で示されており、各列はパイプラインのステージを移行する命令ワードに相当します。例7−1は、サンプルのパイプライン図です。

例7−1. サンプルのパイプライン図



例に示される各列は、左側に命令、オペランド、マルチサイクル命令、またはパイプライン フラッシュでラベル付けされています。上部に並ぶ数字は、各命令サイクルを表します。サイクルによってパイプラインのすべてのステージを示していないのは、不要な表示を避けるためです。

例にある各ボックスは、そのパイプライン ステージに発生する関連動作が示されます。各パイプライン ステージの名称は、ボックスの上に示されます。

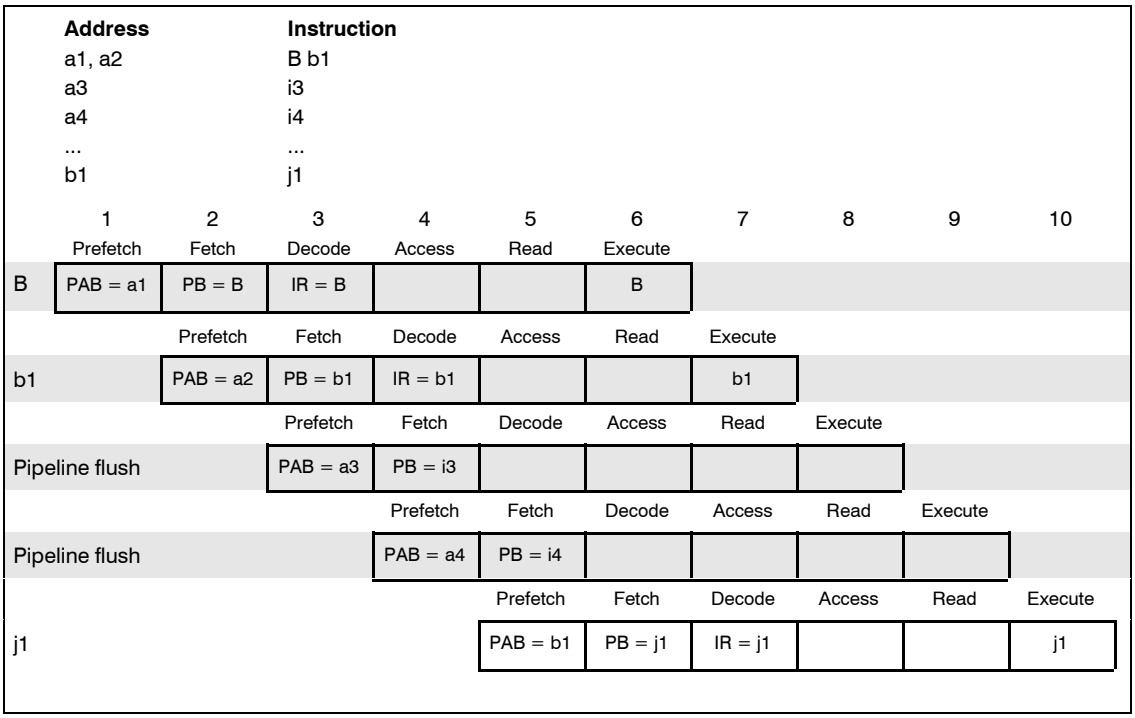
影付きの部分は、示されている分岐命令を完了するために必要な、すべての命令フェッチおよびパイプライン フラッシュを示しています。

7.1.1 パイプラインにおける分岐命令

例7−2および例7−3は、それぞれ分岐(B)命令および遅延分岐(BD)命令が実行される際のパイプラインの行動を示しています。

分岐命令は2つの命令ワードで構成されるので、実行を完了するには最小で2命令サイクルが必要です。ただし、標準的な分岐命令は実際に4サイクルで実行します。これは例7−2に示されます。

例7−2. パイプラインにおける分岐命令



例7−2に示される分岐命令が実行を完了するには、次のイベントが発生します。

サイクル1: 分岐命令のアドレスをPABにドライブします。
サイクル2および3: 分岐命令の2つのワードをフェッチします。

サイクル4および5: **さらに2つの命令、i3およびi4をフェッチします。分岐命令の後の2つの命令i3およびi4は'54xがフェッチしますが、これらはデコードされずに、最終的に無効にされます。分岐命令(左端にb1が示される)の2番目のワードがデコードされた後、PABがこの新たな値によってドライブされます(サイクル5)。**

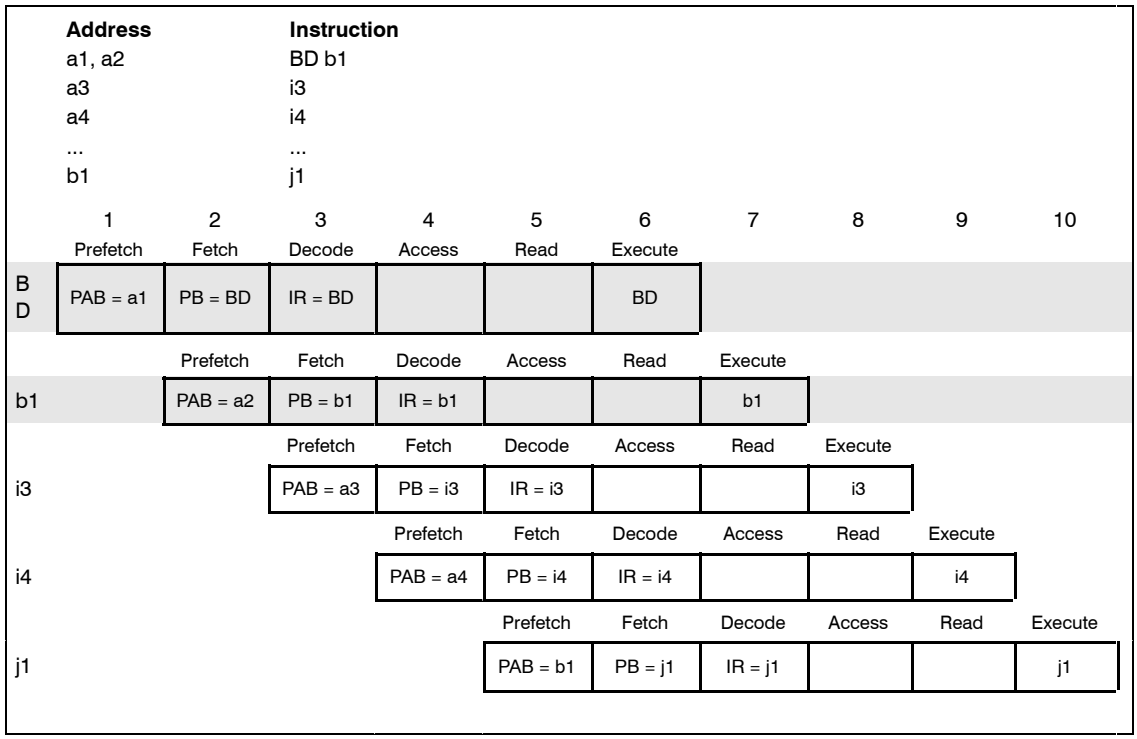
サイクル6および7: **2ワード分岐命令が、サイクル6および7でパイプラインの実行ステージになります。さらにサイクル6で、j1をアドレスb1からフェッチします。**

サイクル8および9: **次の2つの命令i3およびi4への実行は許可されていないので、これらのサイクルは同じ分岐命令でも使われます。これにより、分岐命令は実行に4サイクルを必要とします。**

サイクル10: **j1が実行を完了します。**

例7−3は、遅延分岐命令でのパイプラインの行動を示しています。

例7−3. パイプラインにおける遅延分岐命令



この場合、パイプラインは通常の分岐命令の場合と同じように作動します。ただし、分岐に続く2つの命令i3およびi4は、それぞれの実行を完了できます。したがって、サイクル6および7のみが遅延分岐命令によって使われ、遅延分岐が2サイクル命令になります。

7.1.2 パイプラインにおけるコール命令

標準的なコール命令は、実行に4サイクルかかります。標準的なコールは2ワード命令で2サイクルのみが必要に見えますが、実際には2サイクルでパイプラインをフラッシュするので、実行には4サイクルが必要です。

例7−4は、コール命令を実行する間のパイプラインの動作を示しています。

例7−4. パイプラインにおけるコール命令

	Address	Instruction
	a1, a2	CALL b1
	a3	i3
	a4	i4
	...	...
	b1	j1

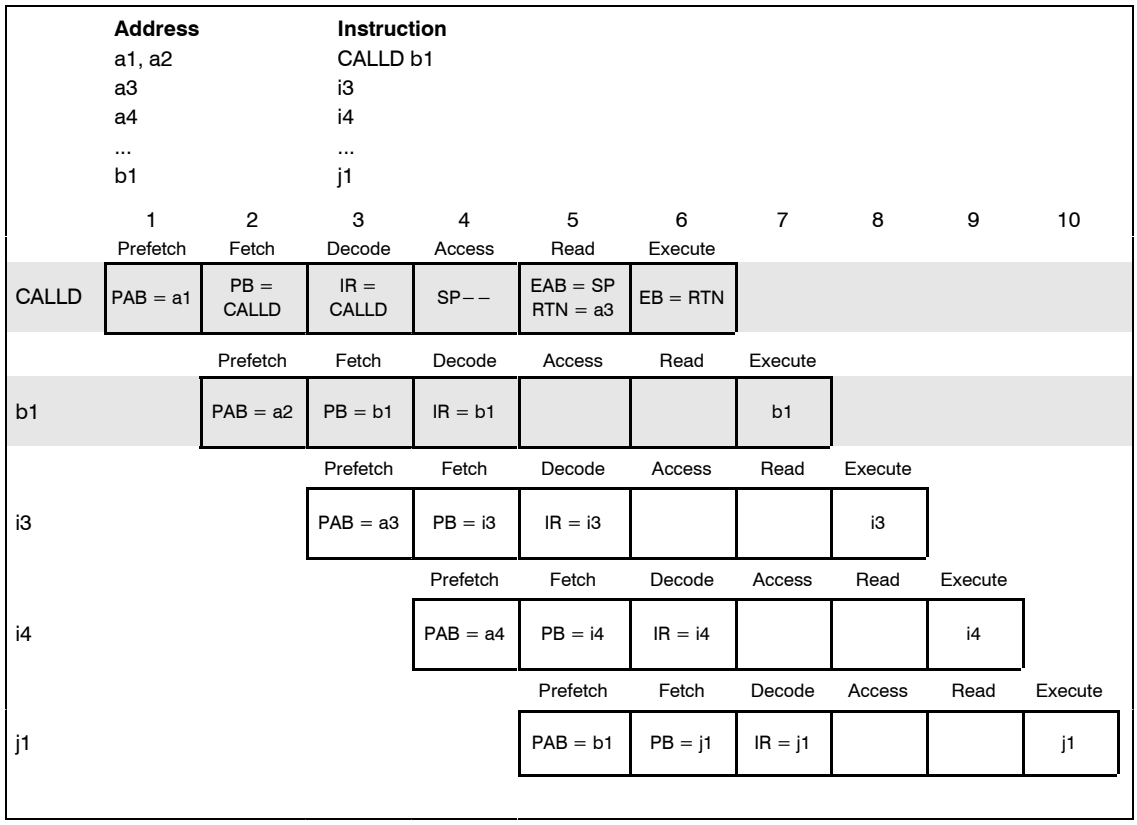
	1	2	3	4	5	6	7	8	9	10
	Pre-fetch	Fetch	Decode	Access	Read	Execute				
CALL	PAB = a1	PB = CALL	IR = CALL	SP− −	EAB = SP RTN = a3	EB = RTN				
		Prefetch	Fetch	Decode	Access	Read	Execute			
b1		PAB = a2	PB = b1	IR = b1			b1			
			Prefetch	Fetch	Decode	Access	Read	Execute		
Pipeline flush			PAB = a3	PB = i3						
				Prefetch	Fetch	Decode	Access	Read	Execute	
Pipeline flush				PAB = a4	PB = i4					
					Prefetch	Fetch	Decode	Access	Read	Execute
j1					PAB = b1	PB = j1	IR = j1			j1

例7−4では、次のイベントが発生します。

サイクル1:	コール命令のアドレスをPABにドライブします。
サイクル2および3:	コール命令の2つのワードをフェッチします。
サイクル4:	リターン アドレスをスタックするために、SPがデクリメント(SP−−で示される)します。命令i3がフェッチされますが、これはデコード ステージを処理されません。
サイクル5:	ライト アドレス バス(EAB)にSPの内容をドライブし、内部レジスタであるリターン レジスタ(RTN)にリターン アドレスa3がロードされます。コール命令(b1)の2番目のワードがデコードされた後、PABがこの新たな値によってサイクル5でドライブされます(j1の列に表示)。
サイクル6および7:	サイクル6で、EBを使ってRTNの内容をスタックにライトします。アドレスb1の命令j1をサイクル6でフェッチします。サイクル6および7では、2ワードのコール命令がパイプラインの実行ステージになります。
サイクル8および9:	次の2つの命令(i3およびi4)はそれぞれの実行を許されていないので、これらのサイクルは呼び出し命令で費やされます。
サイクル10:	j1が実行を完了します。

例7−5は、遅延コール命令のパイプラインの動作を示します。

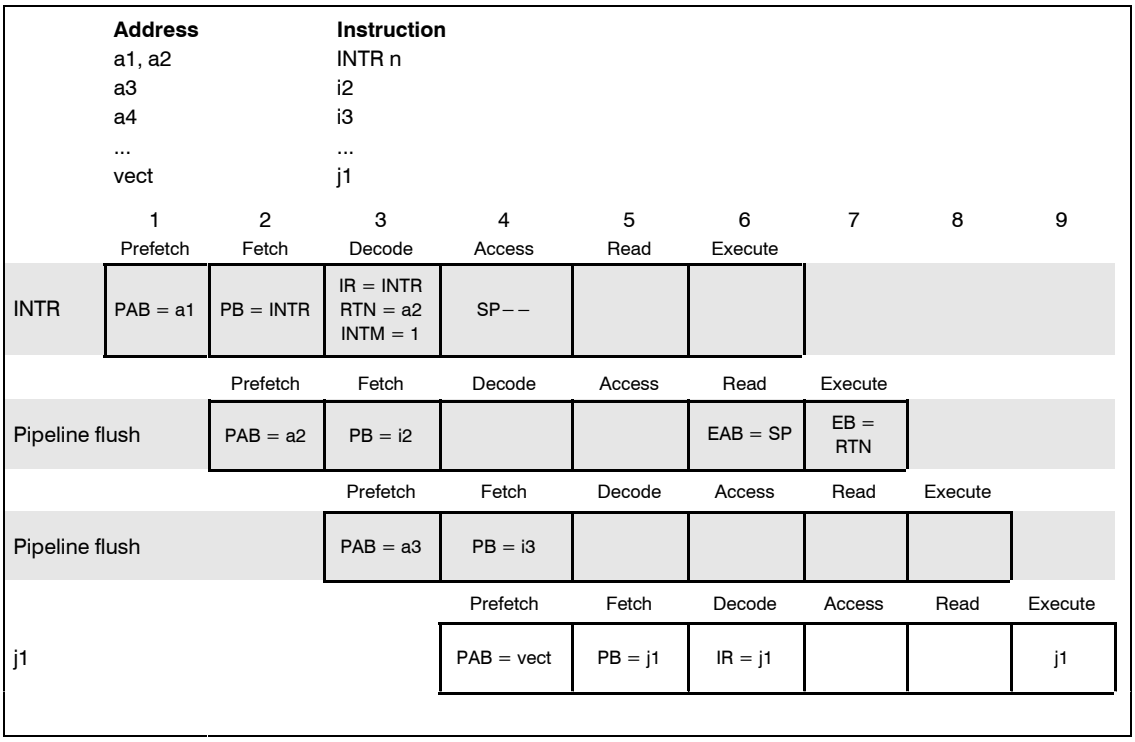
例7−5. パイプラインにおける遅延コール命令



この場合、パイプラインは通常のコール命令と同様の動作になりますが、次の2つの命令i3およびi4はそれぞれの実行を完了できます。したがって、サイクル6およびサイクル7のみが遅延コール命令に使われるので、この命令は2サイクルになります。

INTR命令は、CALL命令のように作動します。ただし、INTRは1ワード命令なのでベクタ テーブル アドレスを計算してそれを1サイクル前にプリフェッチできます。例7−6に示すように、INTRは実行に3サイクルのみかかります。

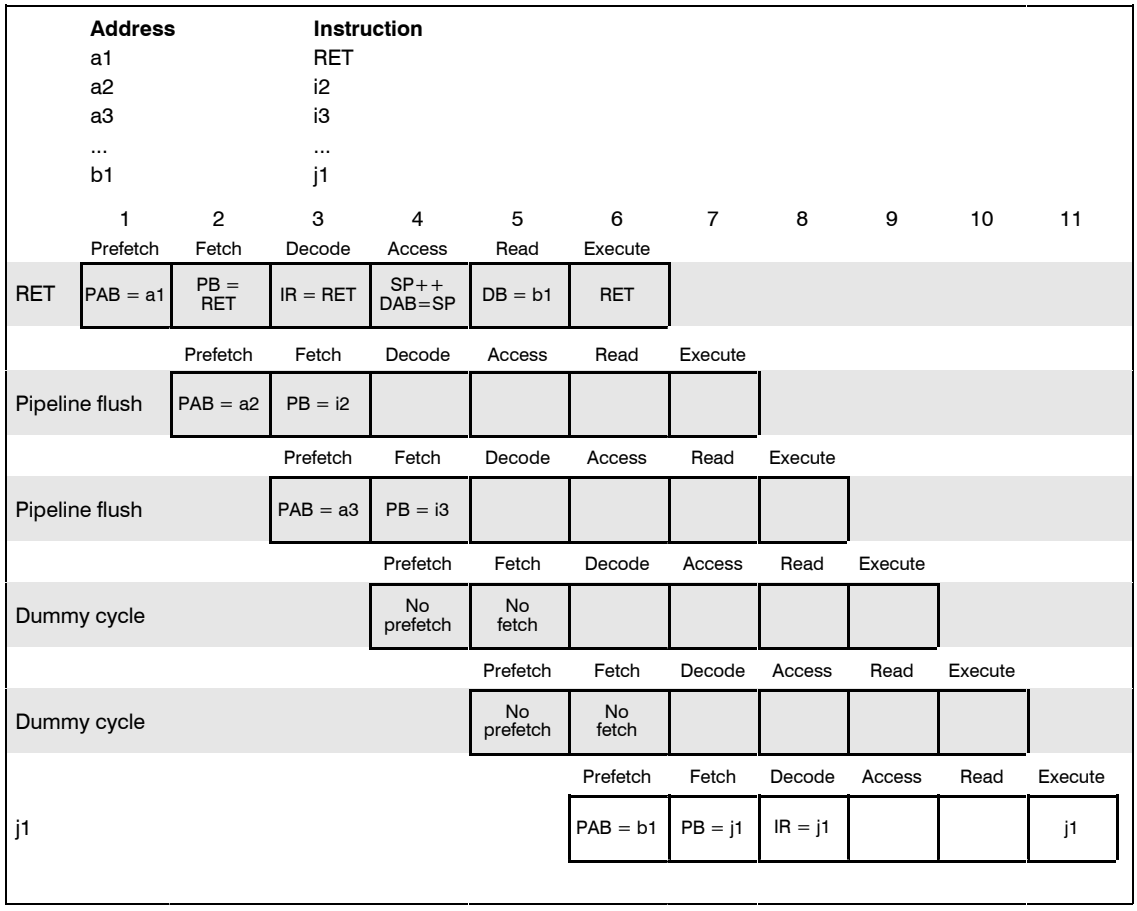
例7−6. パイプラインにおけるINTR命令



7.1.3 パイプラインにおけるリターン命令

リターンは1ワード命令なので、最小1サイクルで実行を完了するように見えます。実際には、標準的なリターン命令は実行に5サイクルかかります。例7−7は、リターン命令を実行する際のパイプラインの動作を示しています。

例7−7. パイプラインにおける復帰命令



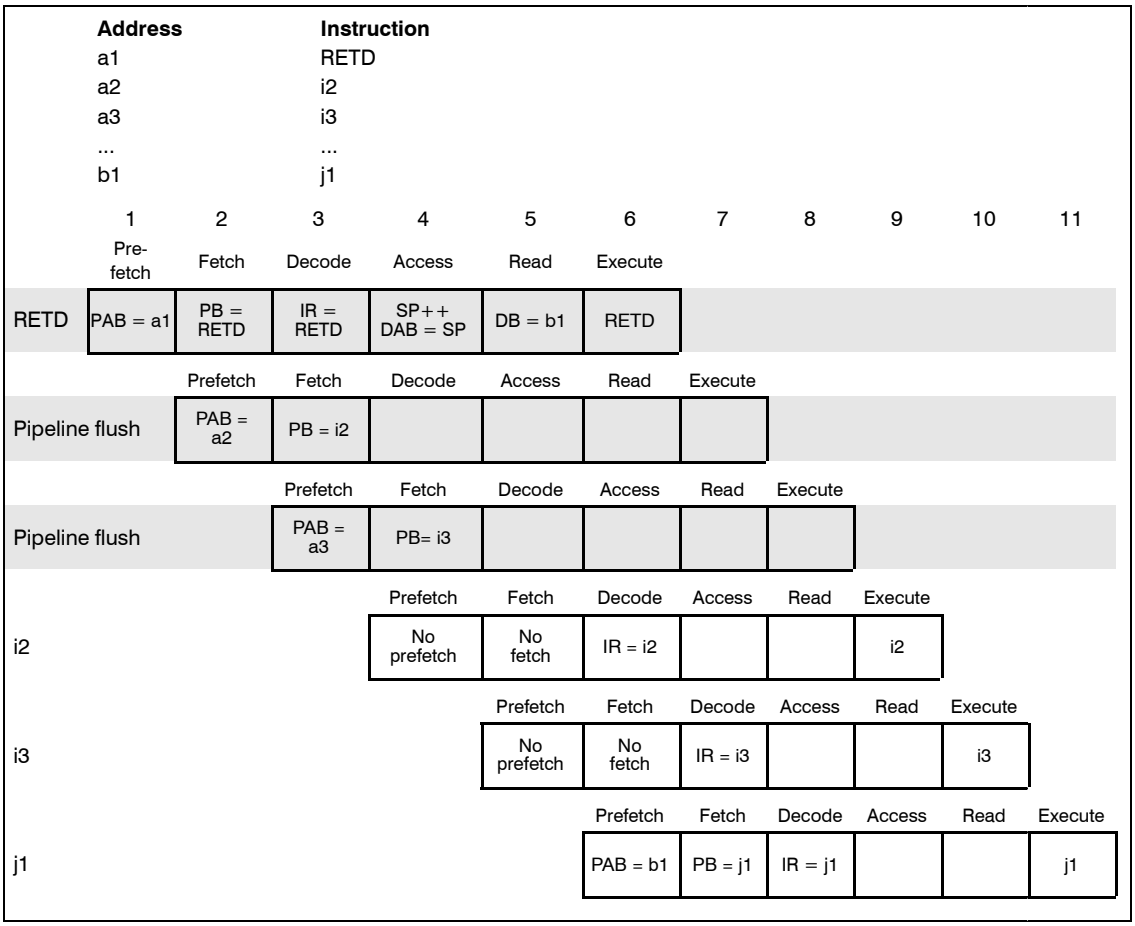
例7－7では、次のイベントが発生します。

サイクル1:	リターン命令のアドレスをPABにドライブします。
サイクル2:	リターン命令オペコードをフェッチします。
サイクル3および4:	さらに2つの命令i2、i3をフェッチします。これらの命令はフェッチされますが、デコード ステージを通過できず処分されます。サイクル4では、SPをインクリメントして(SP++で示される)SPの内容をDABにドライブすることにより、スタックからリターンアドレスをリードします。
サイクル5:	DBを使って、スタックの最上位をリードします。
サイクル6:	リターン命令が、パイプラインの実行ステージに入ります。スタックからフェッチされたアドレスを、PAB上にドライブします。これにより次の命令j1をリターン アドレスでフェッチできます。
サイクル7および8:	続く命令i3、i4のそれぞれの実行が認められないので、これらのサイクルはリターン命令が費します。
サイクル9および10:	サイクル4および5では命令はフェッチされないので、サイクル9および10はダミー サイクルになります。
サイクル11:	j1が実行を完了します。

例7−8は、遅延リターン命令が実行される間のパイプラインの動作を示しています。

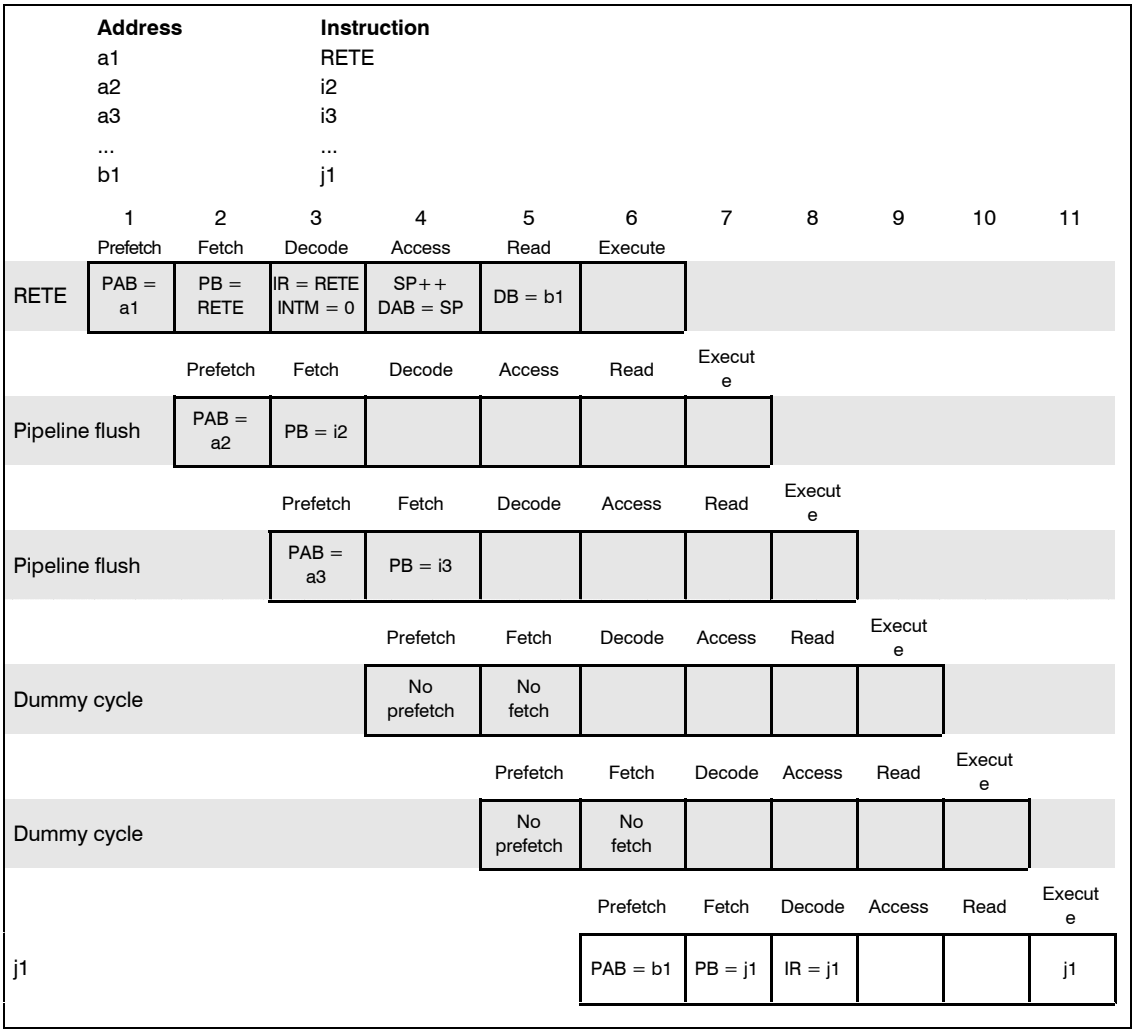
遅延リターン命令では、'54xのパイプラインは通常のリターン命令の場合と同様に作動します。ただし、続く2つの命令i3およびi4がそれぞれの実行を完了できるので、サイクル6、7、8のみが遅延リターン命令によって使われ、これにより命令が3サイクル命令になります(例7−8参照)。

例7−8. パイプラインにおける遅延リターン命令



例7−9および例7−10は、それぞれ割り込みのイネーブルをともなうリターン(RETE)命令、および割り込みのイネーブルをともなう遅延リターン(RETED)命令でのパイプライン動作を示しています。これらの命令でのパイプライン動作は、それぞれが標準のリターン命令および遅延リターン命令と似通っており、これらの命令は実行に数サイクルかかります。相違点としては、これら2つの命令の場合、パイプラインのデコード ステージでINTMビットをリセットすることで全てのマスカブル割り込みをイネーブルにします。

例7−9. パイプラインにおける割り込みのイネーブルをともなうリターン命令



例7−10. **パイプラインにおける割り込みのイネーブルをともなう遅延リターン命令**

	Address	**Instruction**
	a1	RETED
	a2	i2
	a3	i3
	...	...
	b1	j1

	1	2	3	4	5	6	7	8	9	10	11
	Prefetch	Fetch	Decode	Access	Read	Execute					
RETED	PAB = a1	PB = RETED	IR = RETED INTM = 0	SP++ DAB = SP	DB = b1						
		Prefetch	Fetch	Decode	Access	Read	Execute				
Pipeline flush		PAB = a2	PB = i2								
			Prefetch	Fetch	Decode	Access	Read	Execute			
Pipeline flush			PAB = a3	PB = i3							
				Prefetch	Fetch	Decode	Access	Read	Execute		
i2				No prefetch	No fetch	IR = i2			i2		
					Prefetch	Fetch	De-code	Access	Read	Execute	
i3					No prefetch	No fetch	IR = i3			i3	
						Prefetch	Fetch	De-code	Access	Read	Execute
j1						PAB = b1	PB = j1	IR = j1			j1

例7−11および例7−12は、それぞれ高速復帰(RETF)命令および高速遅延復帰(RETFD)命令でのパイプラインの動作を示しています。RETF命令はRETE命令と異なり、スタックからリターン アドレスをリードしません。代わりに、RTNレジスタからリードします。これにより、この命令がRETE命令より2サイクル早くリターン アドレスをPABにドライブできます。例に示されるように、RETF命令は実行に3サイクルかかります。遅延をともなうRETFD命令の場合は、1サイクルで実行します。

例7−11. パイプラインにおける復帰fast命令

	Address	Instruction
	a1	RETF
	a2	i2
	a3	i3
	...	...
	b1	j1

	1	2	3	4	5	6	7	8	9
	Prefetch	Fetch	Decode	Access	Read	Execute			
RETF	PAB = a1	PB = RETF	IR = RETF INTM = 0	SP++ Read RTN					
		Prefetch	Fetch	Decode	Access	Read	Execute		
Pipeline flush		PAB = a2	PB = i2						
			Prefetch	Fetch	Decode	Access	Read	Execute	
Pipeline flush			PAB = a3	PB = i3					
				Prefetch	Fetch	Decode	Access	Read	Execute
j1				PAB = b1	PB = j1	IR = j1			j1

例7−12. **パイプラインにおける高速遅延復帰命令**

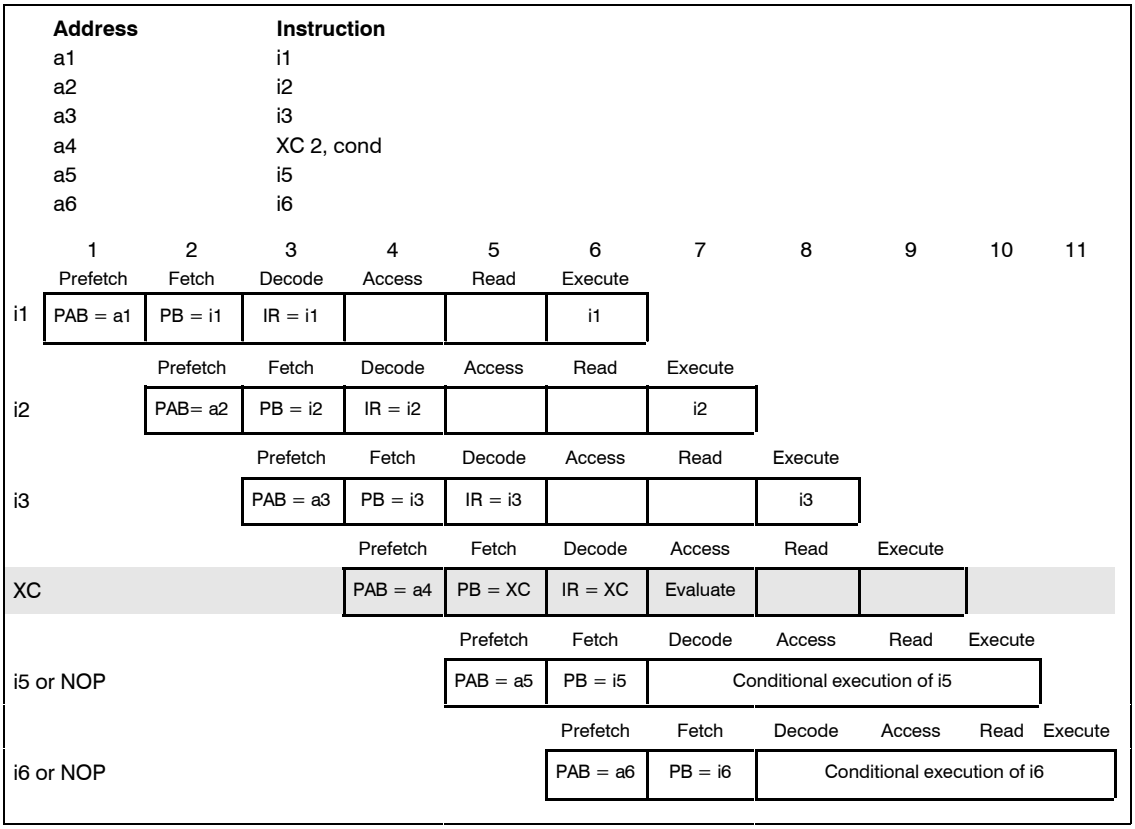
	Address	**Instruction**
	a1	RETFD
	a2	i2
	a3	i3
	...	...
	b1	j1

	1	2	3	4	5	6	7	8	9
	Prefetch	Fetch	Decode	Access	Read	Execute			
RETFD	PAB = a1	PB = RETFD	IR = RETFD INTM = 0	SP++ Read RTN					
		Prefetch	Fetch	Decode	Access	Read	Execute		
i2		PAB = a2	PB = i2	IR = i2			i2		
			Prefetch	Fetch	Decode	Access	Read	Execute	
i3			PAB = a3	PB = i3	IR = i3			i3	
				Prefetch	Fetch	Decode	Access	Read	Execute
j1				PAB = b1	PB = j1	IR = j1			j1

7.1.4 パイプラインにおける条件付き実行命令

XCは1ワード命令で、実行を完了するには少なくとも1サイクルかかります。例7−13は、XCを実行する際のパイプラインの動作を示しています。

例7−13. パイプラインにおけるXC命令



例7−13では、次のイベントが発生します。

サイクル4: XC命令のアドレスをPABにドライブします。

サイクル5: XC命令オペコードをフェッチします。

サイクル7: サイクル7でXC命令がパイプラインのアクセス ステージに進むとき、XC命令で指定される全ての条件が評価されます。テストされた条件が真(true)の場合、次の2つの命令i5およびi6がデコードされ実行可能になります。ただしテストされた条件が偽(false)の場合はi5およびi6はデコードされません。

XCを1サイクルで実行するために、CPUはパイプラインのアクセス ステージでテスト条件を評価します。つまり、XC命令の直前のふたつの1ワード命令(またはひとつの2ワード命令)は、条件がテストされる前では、その実行が完了していません。条件コードは命令の実行ステージによってのみ影響されるので、これら2つの命令はXCの動作には影響しません。

7.1.5 パイプラインにおける条件つきコールおよび条件付き分岐

コール命令は2つの命令ワードから構成されるので、少なくとも実行を完了するには2サイクルかかるように見えます。標準的な条件付きコール命令は、実際には条件によってコールされる場合は5サイクルで実行します。コールされない場合は3サイクルで実行します。

例7－14は、条件付きコール命令(CC)を実行する間のパイプラインの動作を示しています。

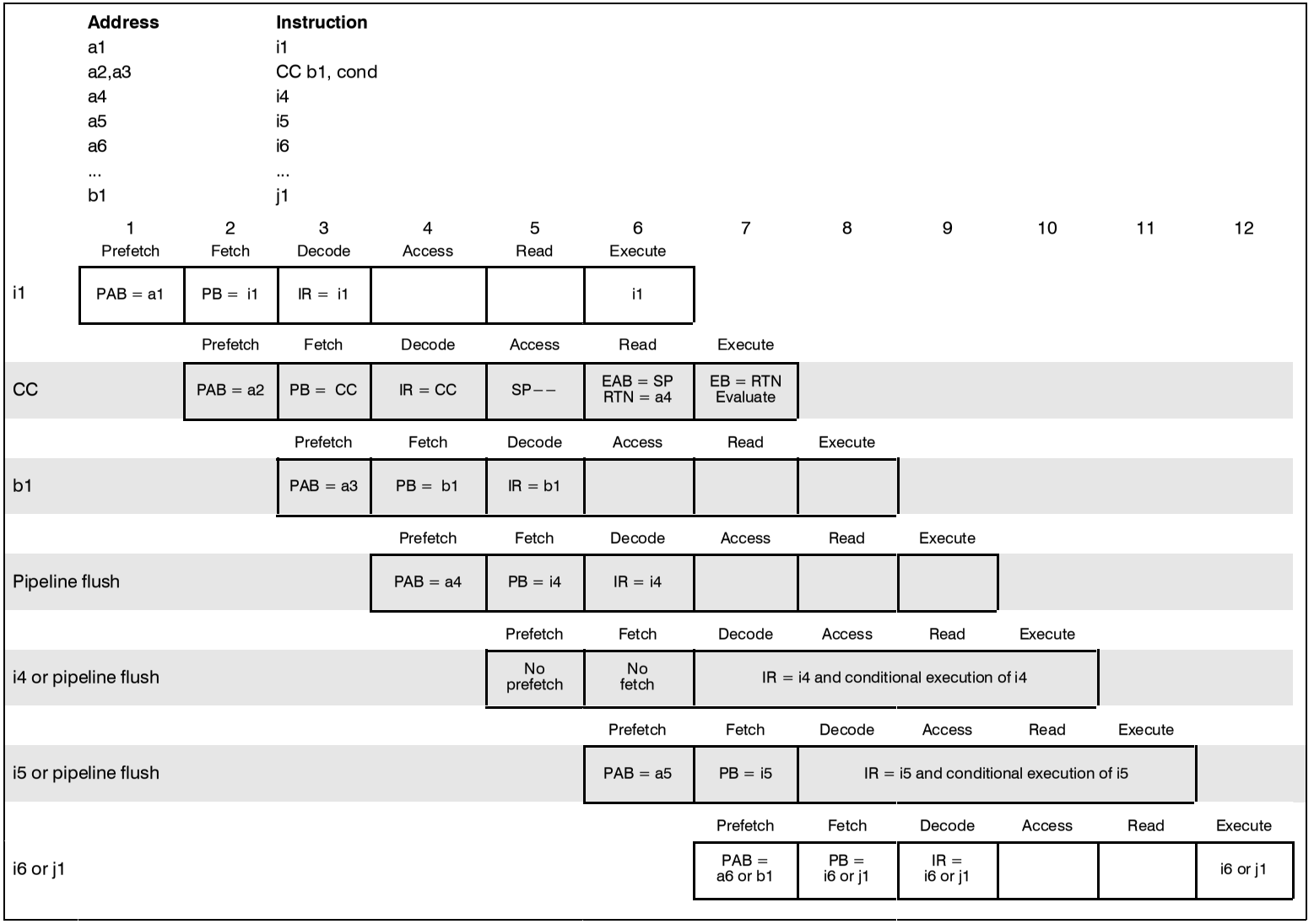
条件付きコール命令は、パイプラインの動作の面では条件なしコール命令と似ています。唯一の例外は、条件付きコール命令のテスト条件が、パイプラインの実行ステージで評価されることです。例7－14に示されるように、テスト条件がサイクル7で評価されるとき、前の命令i1は完全に実行されます。さらに、CCの後に続く2つの命令i4、i5もフェッチされます。テスト条件が偽(false)と評価される場合は、これら2つの命令がパイプラインを通過します。そうでない場合、これらは処分されます。サイクル7における命令フェッチは、評価された条件に依存します。条件が真(true)の場合は、呼び出しアドレス(b1)をPABにドライブします。そうでない場合は、次のインクリメント アドレス(a6)をPABのドライブします。

評価された条件が真(true)の場合は、i4およびi5はサイクル10および11で実行されません。この場合、CC命令は5サイクル命令になります。一方、評価された条件が偽(false)の場合は、i4およびi5が実行され、CCは3サイクル命令になります。

例7－15は、遅延条件付きコール(CCD)命令を実行する間のパイプラインの動作を示しています。

パイプラインの動作は、CC命令の場合と同様になります。ただし、続く2つの命令i3およびi4は、テストされた条件が真かどうかにかかわらず、それぞれの実行を完了できます。CCD命令はサイクル7、8、9を使用して3サイクル命令になります。

例7−14. **パイプラインにおけるCC命令**



例7−15. **パイプラインにおける**CCD**命令**

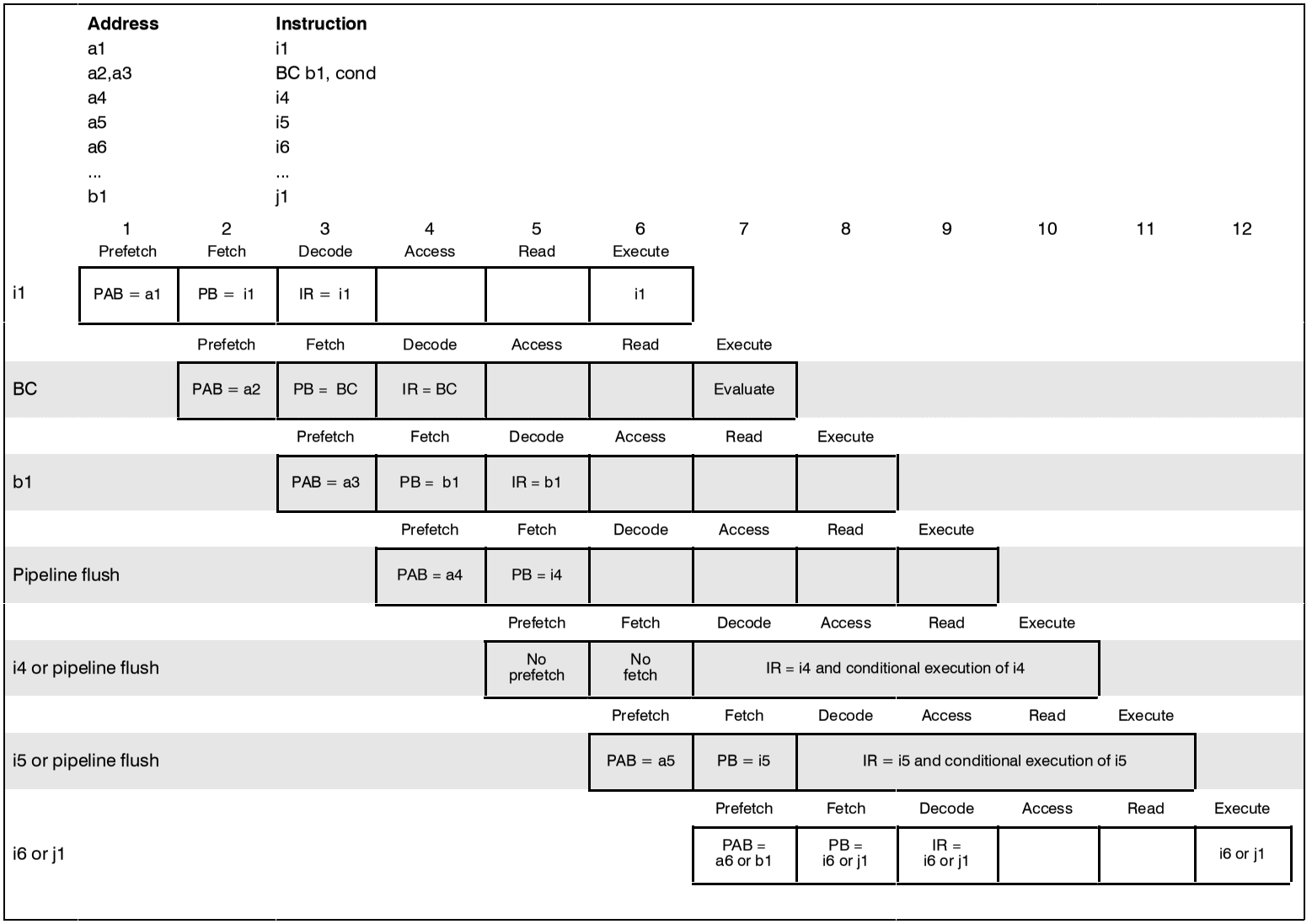
Address	Instruction
a1	i1
a2, a3	CCD b1, cond
a4	i4
a5	i5
a6	i6
:	:
b1	j1

	1	2	3	4	5	6	7	8	9	10	11	12
	Prefetch	Fetch	Decode	Access	Read	Execute						
i1	PAB = a1	PB = i1	IR = i1			i1						
		Prefetch	Fetch	Decode	Access	Read	Execute					
CCD		PAB = a2	PB = CCD	IR = CCD	SP− −	EAB = SP RTN = a4	EB = RTN Evaluate					
			Prefetch	Fetch	Decode	Access	Read	Execute				
b1			PAB = a3	PB = b1	IR = b1							
				Prefetch	Fetch	Decode	Access	Read	Execute			
Pipeline flush				PAB = a4	PB = i4	IR = i4						
					Prefetch	Fetch	Decode	Access	Read	Execute		
i4					No prefetch	No fetch	IR = i4			i4		
						Prefetch	Fetch	Decode	Access	Read	Execute	
i5						PAB = a5	PB = i5	IR = i5			i5	
							Prefetch	Fetch	Decode	Access	Read	Execute
i6 or j1							PAB = a6 or b1	PB = i6 or j1	IR = i6 or j1			i6 or j1

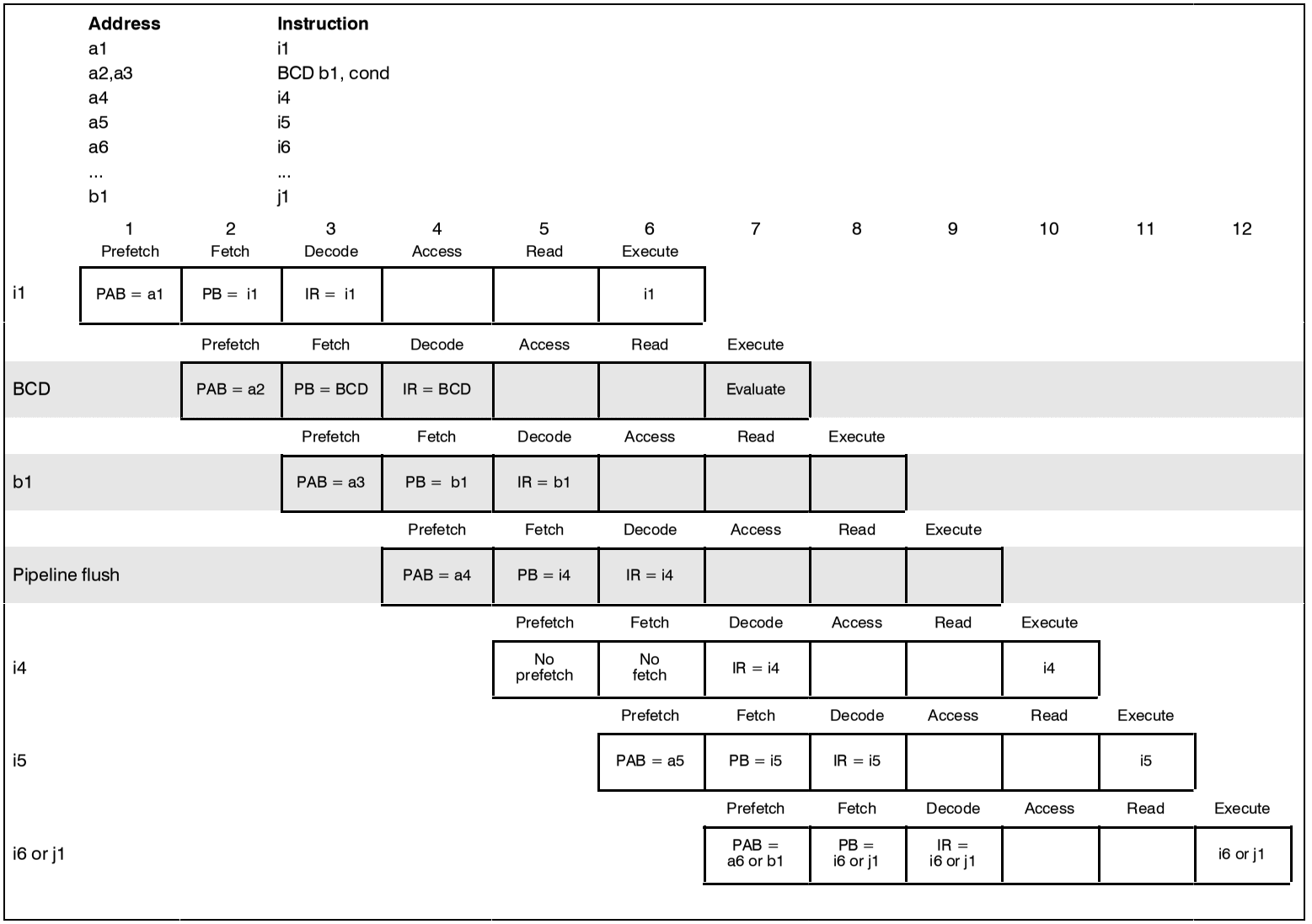
例7－16および例7－17は、条件付き分岐(BC)命令、および遅延条件付き分岐(BCD)命令を実行する間のパイプラインの動作を示しています。

パイプラインにおける条件付き分岐(BC)および遅延条件付き分岐(BCD)命令の動作は、それぞれCCおよびCCDのパイプラインにおける動作に似ています。ただしBCおよびBCDの場合は、リターン アドレスはスタックにライトされません。例7－16に示されるように、分岐を取るか取らないかによって、BC命令は実行に3サイクルまたは5サイクルかかります。BCD命令は、実行に3サイクルかかります。

例7−16. **パイプラインにおけるBC命令**



例7−17. **パイプラインにおける**BCD**命令**



7.2 割り込みおよびパイプライン

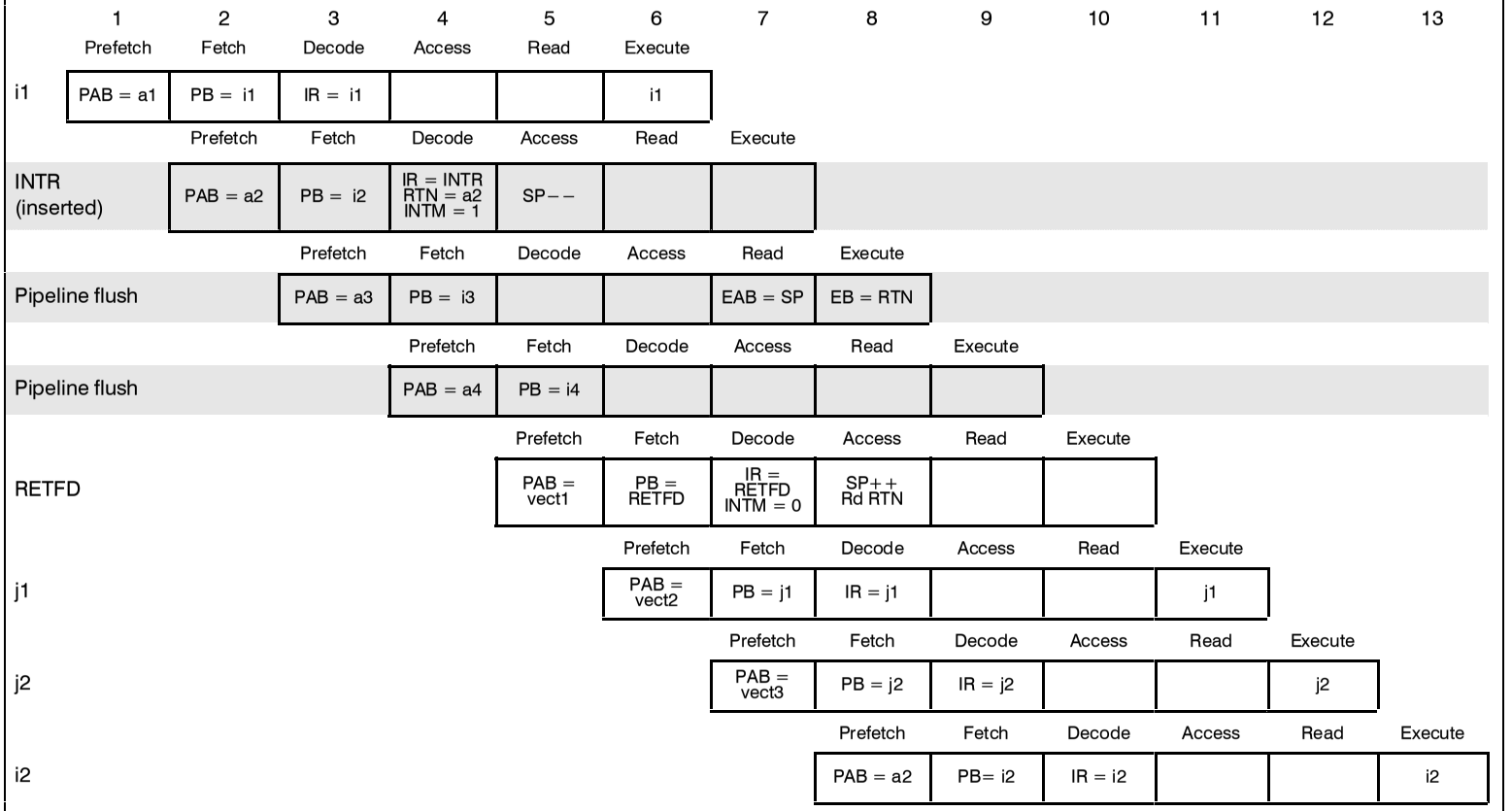
例7−18は、割り込みが取られる場合のパイプラインの動作を示しています。

例7−18に示されるように、サイクル3の最後に割り込みサービスが行われると、次のサイクル4でINTR命令を自動的にパイプラインのデコード ステージに挿入されます。命令i2はデコードされず、INTR命令がパイプラインのそのデコード ステージに挿入されます。次の3サイクル間に、すでにデコードされた命令を実行します。サイクル7、8、9はINTR命令が使用します。割り込みサービス ルーチン(ISR)の最初の命令RETFDをサイクル10で実行します。サイクル11、12はふたつの1ワード命令で使われ、これらの命令はRETFD命令の2つの遅延スロットを構成します。続くサイクルでは、命令i2を実行してISRからの復帰を完了します。

図に示されるように、割り込みオーバーヘッド(ISRへの分岐に必要なサイクル数)は3サイクルのみです。割り込みからの復帰には1サイクルのみかかりますが、これはRETFDが1サイクル命令だからです。割り込みベクタ テーブルの各割り込みには4ワードしか予約されていないので、ISRが4命令ワードをこえる命令ワード数を要求する場合は、ISRは別の場所に置かれる必要があります。この場合、割り込みベクタ テーブルの内容は分岐タイプの命令になります。この結果、割り込みオーバーヘッドが多少大きくなります。

例7−18. パイプラインによる割り込み応答

```
Address     Instruction
a1          i1          ; Interrupt taken after executing this instruction
a2          i2
a3          i3
a4          i4
:           :
vect1       RETFD       ; First instruction in the interrupt vector table
vect2       j1
vect3       j2
```



7.3 デュアル アクセス メモリおよびパイプライン

’54xにはオンチップ メモリがあり、単一サイクルで2回のアクセスをサポートします。このデュアル アクセス メモリは、複数の独立したメモリ ブロックから構成されます。異なるブロックへの同時アクセスも、コンフリクトなしにサポートします。パイプラインの中のひとつの命令がブロックにアクセスする間に、別の命令が同じサイクルのパイプライン ステージでコンフリクトなしに異なるブロックへアクセスできます。さらに、各メモリ ブロックは1サイクルで2つのアクセスをサポートします。それぞれ異なるパイプライン ステージにある2つの命令が、同じブロックに同時にアクセスできます。ただし、2つの同時アクセスが同じブロックで実行される場合は、コンフリクトが発生する場合もあります。’54x CPUはこれらのコンフリクトを自動的に解決します。詳細は、以下のセクションで説明します。表7−1は、各’54xデバイスのブロック サイズおよびブロック数を示しています。DARAMの構成については、3−15ページ、サブセクション3.3.2を参照してください。

表7−1. DARAMブロック

デバイス	ブロック サイズ†	ブロック数
’541	1Kワード	5
’542	2Kワード	5
’543	2Kワード	5
’545	2Kワード	3
’546	2Kワード	3
’548	2Kワード	4

† 下位側の最初のブロックは、メモリ マップド レジスタ、予約領域の分およびスクラッチ パッドRAMのために、多少小さくなります。

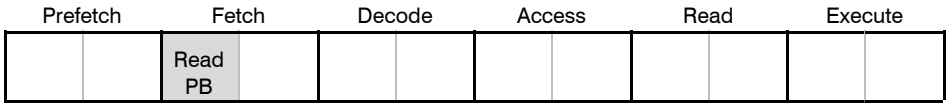
各デュアル アクセス メモリ ブロックは、最初のハーフ サイクルで1つのアクセスを実行し次のハーフ サイクルで別のアクセスを実行することで、2つのメモリ サイクルを1サイクルでサポートします。表7−2は、各ハーフ サイクルで実行されるアクセスを示し、図7−3は異なるタイプのアクセスがどのように実行されるかを示しています。図の簡略化のためにアドレス バスのドライブは省略されています。

表7−2. DARAMブロックへのアクセス

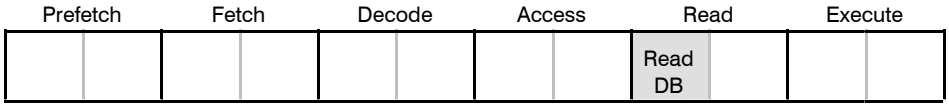
アクセスのタイプ ...	そのアクセスが実行されるサイクル...
PAB/PBを使う命令フェッチ	First half-cycle
DAB/DBを使う最初のデータ オペランドリード	First half-cycle
CAB/CBを使う2番目のデータ オペランドリード	Second half-cycle
EAB/EBを使うデータ オペランドライト	Second half-cycle

図7−3. **デュアル アクセス メモリへのハーフ サイクル アクセス**

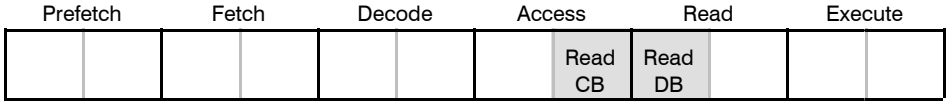
(a) **命令ワード フェッチ**



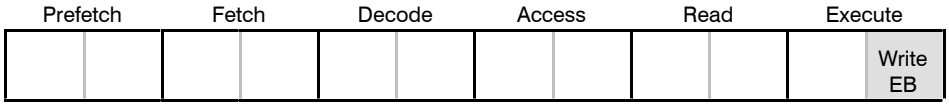
(b) 1**オペランド リードを実行する命令**



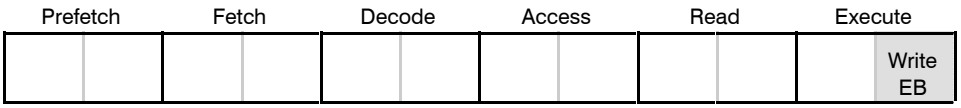
(c) **デュアル オペランド リードを実行する命令**

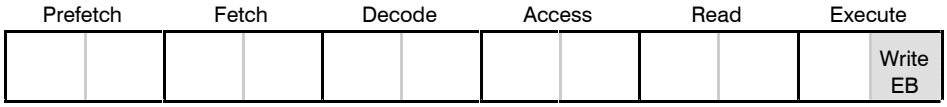


(d) 1**オペランド ライトを実行する命令**

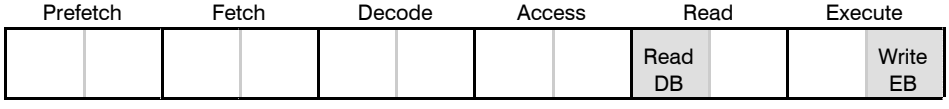


(e) **デュアル オペランド ライトを実行する命令**(2**サイクル**)





(f) **オペランド リードおよびオペランド ライトを実行する命令**

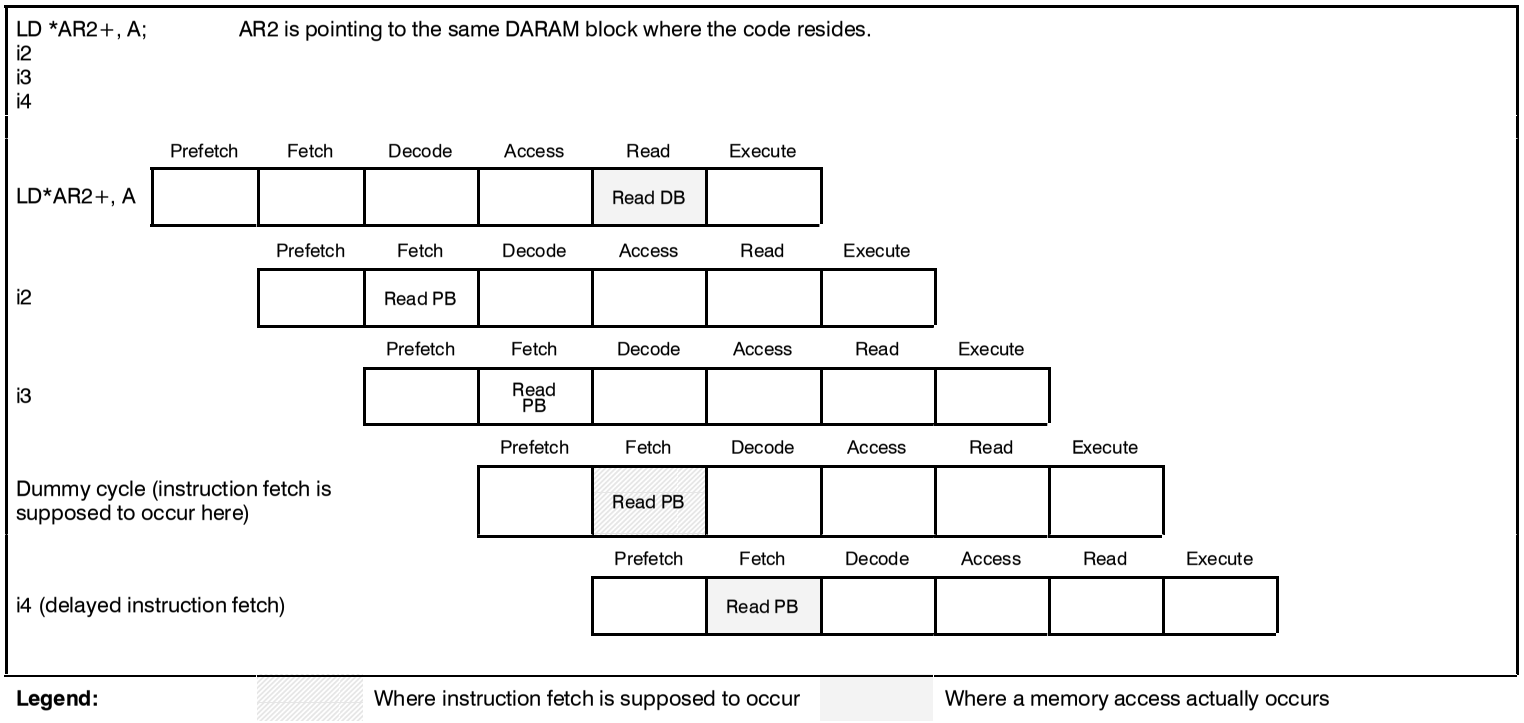


2タイプのアクセスをスケジューリングして、各ハーフ サイクルで1アクセスのみを実行することから、コンフリクトが発生する場合があります。このようなコンフリクトは、アクセス順を並べ替えるかまたはアクセスを1サイクル遅らせることにより、CPUが自動的に解決します。次のサブセクションは、このようなメモリ アクセス コンフリクトの解決について説明します。このようなコンフリクトは、すべてのアクセスが同じデュアル アクセス メモリ ブロックで実行されるときのみに発生することを念頭に置いてください。

7.3.1 命令フェッチとオペランド リード間のコンフリクトの解決

デュアル アクセス メモリ ブロックがプログラムおよびデータ空間の双方にマップされる場合、命令フェッチとデータ オペランド リード アクセスが同じメモリ ブロックで実行される時、これらがコンフリクトを起こします。'54xは、命令フェッチを1サイクル遅らせることによりこのコンフリクトを解決します(例7−19参照)。図では、命令i2およびi3が、自分自身が存在するデュアル アクセス メモリ ブロックにはアクセスしないことを前提とします。

例7−19. **命令フェッチとオペランド リード**



7.3.2 オペランド ライトとデュアル オペランド リード間のコンフリクトの解決

1オペランド ライト命令の後にライト アクセスを実行しない命令が続き、この命令の後にデュアルオペランド リード命令が続く場合は、別のコンフリクトが発生します。例7−20はこのようなケースを示しており、ここではAR3およびAR5が同じデュアル アクセス メモリ ブロックを示しています。

オペランド ライト アクセス(EBバス)およびデュアル オペランド リード命令の2つ目のデータ リード アクセス(CBバス)の間には、コンフリクトが起こります。このコンフリクトは、ライト アクセスを1サイクル遅らせることにより、自動的に解決します。これらの命令の実際の実行時間が増すことはありません。これは、遅らされたライト アクセスが2番目の命令の実行ステージで実行されるからです。

いずれかのリード アクセス(DBまたはCBを経由)のアドレスと、ライト アクセスのアドレスが同じ場合は、CPUは実際のメモリ位置のリードをバイパスします。すなわち、CPUは内部バスから直接データをリードします。これによりパイプラインは、ライトと同じメモリ アドレスにリード アクセスをするパイプライン ステージよりも後で、ライト アクセスを実行することができます。

例7−20. **オペランド ライトとデュアル オペランド リードのコンフリクト**

```
STL A, *AR3+
LD #0, A
ADD *AR4+, *AR5+, A          ; AR3 and AR5 both point to the same dual-access memory block.
```

STL A, *AR3+
(write access delayed until next cycle)

Prefetch	Fetch	Decode	Access	Read	Execute
					Write EB

LD #0, A (write access of previous instruction occurs here)

Prefetch	Fetch	Decode	Access	Read	Execute
					Write EB

ADD *AR4+, AR5, A

Prefetch	Fetch	Decode	Access	Read	Execute
			Read CB	Read DB	

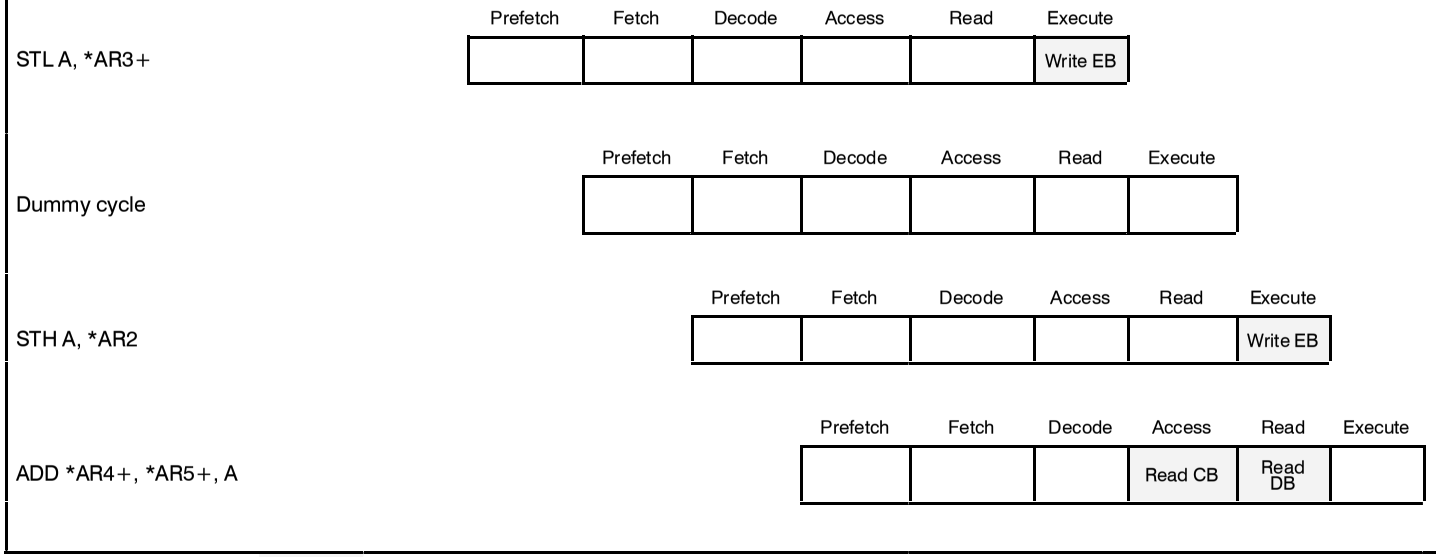
Legend: Where an access is supposed to occur Where an access actually occurs

7.3.3 オペランド ライト　オペランド ライト　デュアル オペランド リードのコンフリクトの解決

上記の2番目の命令がオペランド ライトタイプの命令の場合は、最初の命令が要求するライト アクセスは次のサイクルに移行できません。CPUは、ダミー サイクルを最初の命令の後に挿入することで、コンフリクトを解決します。例7−21これを示しており、ここではAR3およびAR5が同じデュアル アクセス メモリ ブロックを示します。

例7−21. **オペランド ライトおよびオペランド リードのコンフリクト**

```
STL A, *AR3+
STH A, *AR2
ADD *AR4+, *AR5+, A          ; AR3 and AR5 both point to the same dual-access memory block
```



Legend: Where read or write occurs

7.4 シングル アクセス メモリおよびパイプライン

’54xはオンチップのシングル アクセス メモリを備えており、各メモリ ブロックへ1サイクルに1回のアクセスをサポートします。’54xデバイスには、以下のような2種類の異なるシングル アクセス メモリがあります。

- ❑ シングル アクセス リード/ライトメモリ(SARAM)
- ❑ シングル アクセス リード オンリーメモリ(ROMまたはDROM)

両タイプのシングル アクセス メモリは、パイプライン上でのアクセスの面では同じように動作します。ただし、ROMおよびDROMにはライトできません。これらのメモリ ブロックは連続したアドレスにマッピングされています。メモリ ブロックの詳細については、3−11ページ、サブセクション3.2.2オンチップROM構成、および3−15ページ、サブセクション3.3.2オンチップRAM構成を参照してください。

異なるメモリ ブロックにアクセスされる限り、シングル アクセス メモリへの同時アクセスは、コンフリクトなしでサポートされます。パイプラインのあるステージのひとつの命令がひとつのメモリ ブロックにアクセスする間に、別の命令が異なるメモリ ブロックに同じサイクルでコンフリクトなしにアクセスできます。

コンフリクトは、2つの同時アクセスが同じメモリ ブロックで実行される場合に起こります。このようなコンフリクトの場合、ひとつのアクセスのみがそのサイクルで実行され、2番目のアクセスは次のサイクルまで遅らされます。この結果、1サイクルのパイプライン レイテンシが発生します。

シングル アクセス メモリが原因のパイプラインのコンフリクトは、いくつかの異なる状況で発生します。

- ❑ デュアル オペランド命令。多くの命令には2つのメモリ オペランドがあり、データをリード/ライトします。双方のオペランドが同じシングル アクセス メモリ ブロックをポイントする場合は、パイプラインのコンフリクトが発生します。CPUは、自動的に命令の実行を1サイクル遅らせてコンフリクトを解決します。たとえば、次のようになります。

```
MAC *AR2+, *AR3+%,A,B ; This instruction will take two
                      ; cycles if both operands are in
                      ; same SARAM or DROM block.
```

- ❑ 32ビット オペランド命令。32ビット メモリ オペランドをリードする命令は、命令のオペランドがシングル アクセス メモリにある場合も、実行にかかるのは1サイクルのみです。シングル アクセス メモリ ブロックは、32ビット リードが1サイクルで完了できるように設計されています。32ビット オペランドをライトする命令は、実行に2サイクルかかります。

```
DLD *AR2, A   ; This instruction only takes 1 cycle even
              ; if the operand is in single•access memory.
```

- **リード/ライト コンフリクト。シングル アクセス メモリ ブロックにライトを実行する命令に、同じシングル アクセス メモリ ブロックからリードを行う命令が続く場合は、コンフリクトが発生します。これは、両方の命令が同じメモリ ブロックに同時にアクセスしようとするからです。この場合、リード アクセスは自動的に1サイクル遅れます。たとえば、次のようになります。**

```
STL A, *AR1+  ; AR1 and AR3 points at the same SARAM
              ; block.
LD  *AR3, B   ; This instruction takes 1 additional
              ; cycle due to a memory access conflict.
```

一方、リード オペランドおよびライト オペランドがあるデュアル オペランド命令は、このようなコンフリクトを発生しません。これは、これら2つのアクセスが2つの異なるパイプライン ステージで行われるからです。たとえば、次のようになります。

```
ST A, *AR2+     ; This instruction does not take any
||ADD *AR3+, B  ; extra cycles, even if AR2/AR3 point
                ; at the same single access memory
                ; block.
```

- **命令データ コンフリクト。**SRAMまたはROMがプログラムおよびデータ空間の双方にマップされるときに起こるコンフリクトがあります。この場合、命令がメモリ ブロックからフェッチされ、データ アクセス(リード/ライトのいずれか)が同じメモリ ブロックで実行されるときは、命令フェッチは1サイクル遅れます。たとえば、次のようになります。

```
LD  *AR1+, A  ; This read data access delays a
              ; subsequent instruction fetch.
STH A, *AR2   ; This write data access delays a
              ; subsequent instruction fetch
```

この状況では、前に説明したケースに比べて大幅に高いパイプライン レイテンシを引き起こします。これは、メモリ ブロックへのリード/ライトいずれかのアクセスがあるたびに、パイプラインは1サイクル分停止します。通常は、ことなるシングル アクセス メモリ ブロックを、それぞれデータまたはプログラム ストアのために確保して、プログラムがストアされるブロックにデータ アクセスが行われることを回避するよう、お勧めします。

7.5 パイプライン レイテンシ

’54xのパイプラインでは、複数の命令がCPUリソースに同時にアクセスできます。CPUリソースは限られているので、ひとつのCPUリソースに複数のパイプライン ステージがアクセスするとコンフリクトが発生します。このようなコンフリクトの一部は、アクセスを遅らせることでCPUが自動的に解決します。その他のコンフリクトは防止されないので、プログラマが解決しなければなりません。

通常、保護されていないコンフリクトは、命令を並べ替えること、またはNOP命令を挿入することで解決します。さらにこのようなコンフリクトは、パイプラインのコンフリクトを発生しない命令だけを使うこと、または特定のレジスタにアクセスする前に必要な遅延を考慮することにより、防止できます。

7.5.1 メモリ マップド レジスタへのアクセスのための命令

保護されていないパイプラインのコンフリクトは、次のメモリ マップド レジスタのいずれかにアクセスするとき発生する場合があります。

- ❑ 補助レジスタ(AR0−AR7)
- ❑ ブロック サイズ レジスタ(BK)
- ❑ スタック ポインタ
- ❑ テンポラリ レジスタ
- ❑ プロセッサ モード ステータス レジスタ(PMST)
- ❑ ステータス レジスタ(ST0またはST1)
- ❑ ブロック リピート カウンタ レジスタ(BRC)
- ❑ メモリ マップド アキュムレータ レジスタ

ただし命令によっては、適切なレイテンシ サイクルを考慮すればパイプラインのコンフリクトなしにこれらのレジスタにアクセスできるものもあります。表7−3はそのような命令を示しています。

表7−3は、プログラマが3列目に示される命令だけを使って2列目のファンクションを実行する場合に有効です。それ以外の場合、個別の命令レイテンシについては以下のサブセクションを参照してください。この表は、パイプライン レイテンシのクイック リファレンスとしても活用できます。表には、考えられる全てのレイテンシが網羅されてはおらず、レイテンシについての詳しい説明はありません。

表7−3. メモリ マップド レジスタにアクセスするための命令

Cat†	Function	Instruction(s)	Latency	Additional Restrictions
1	Writing to ARx/BK without using an accumulator	STM MVDK MVMM MVMD	ARx update: None BK update: The next word must not use circular addressing	None
2	Writing to ARx/BK using an accumulator	STLM STH STL Store type‡	The next 2 words (ARx) or 3 words (BK) must not use the same register.	The next instruction must not write to any ARx, BK, or SP using STM, MVDK, or MVMD.
3	Popping ARx/BK from stack	POPM	The next 1 word (ARx) or 2 words (BK) must not use the same register.	Do not precede a category 3 instruction with any category 2 or 5 instruction that writes to any ARx, BK, or SP.
4	Writing to SP without using an accumulator	STM MVDK MVMM MVMD	None if CPL = 0 The next 1 word must not use SP if CPL = 1.	None
5	Writing to SP using an accumulator	STLM STH STL Store type‡	The next 2 (if CPL = 0) or 3 (if CPL = 1) words must not use SP.	The next instruction must not write to ARx, BK, or SP using STM, MVDK, or MVMD.
6	Writing to T without using an accumulator	STM MVDK LD Smem,T LD Smem,T || ST	None	None
7	Writing to T using an accumulator	STLM STH STL	The next word must not use T.	None
8	Writing to BRC without using an accumulator	STM MVDK	None	None
9	Writing to BRC using an accumulator	STLM STH STL Store type‡	The next instruction must not be a RPTB[D].	None
10	Writing to ARP	LD #k, ARP	None	None

† Category

‡ Any other store-type instruction. See **表**7−5 on page 7−42 for a list of store-type instructions.

表7−3. メモリ マップド レジスタにアクセスするための命令(続き)

Cat†	Function	Instruction(s)	Latency	Additional Restrictions
11	Writing to DP	LD #k, DP LD Smem, DP	None	None
12	Writing to CPL	RSBX SSBX	The next 3 words must not use direct addressing mode.	None
13	Writing to SXM	RSBX SSBX	The next word must not be affected by SXM status.	None
14	Writing to ASM	LD #k, ASM LD Smem, ASM	None	None
15	Writing to BRAF	RSBX SSBX	The next 5 words must not contain the last instruction word in the RPTB loop.	None
16	Writing BRC to memory	SRCCD	The next 2 words must not contain the last instruction word in the RPTB loop.	None
17	Writing to OVLY, MP/$\overline{MC}$, or IPTR	ANDM ORM XORM	The next 6 cycles must not include an instruction fetch from the on-chip memory's address range.	An external-bus cycle may cause additional latency.
18	Writing to DROM bit	ANDM ORM XORM	The next 3 words must not access the DROM's address range.	An external-bus cycle may cause additional latency.
19	Calculating an exponent	EXP	The next instruction must not use T.	None

† Category

‡ Any other store-type instruction. See 表7−5 on page 7−42 for a list of store-type instructions.

表7−3. メモリ マップド レジスタにアクセスするための命令(続き)

Cat†	Function	Instruction(s)	Latency	Additional Restrictions
20	Stack manipulation in compiler mode (CPL = 1)	FRAME POPM/POPD PSHM/PSHD	The next instruction must not use direct addressing mode (CPL = 1).	None
21	Reading AG, AH, AL, BG, BH, or BL as memory-mapped registers	Any instruction that can read from memory	The previous instruction must not modify accumulator A or accumulator B.	None

† Category
‡ Any other store-type instruction. See **表**7−5 on page 7−42 for a list of store-type instructions.

7.5.2 ARx、BK、SPの更新—コンフリクトの解決

表7−4は、パイプラインのリード ステージでデータアドレス生成ロジック(DAGEN)レジスタを更新する'54xの命令を示しています。DAGEN**レジスタは、補助レジスタ(ARx)、ブロック サイズ レジスタ(BK)、スタック ポインタ(SP)です。**

これらのレジスタにライトを行うすべてのその他の命令は、それぞれのライトを実行ステージで行い、ストア タイプの命令になります。それらは表7−5に示されます。

表7−4. リード ステージでDAGEN**レジスタにアクセスする命令**

Instruction Type	Instructions	
Constant initialization	STM	#lk, MMR
	ST	#lk, Smem†,‡
Move type 1	MVDD	Xmem, Ymem†
	POPM	MMR
	POPD	Smem†,‡
	DELAY	Smem†,‡
Move type 2	MVDK	Smem, dmad†
	MVMD	MMR, dmad†

† This operand must be pointing to one of the DAGEN registers.
‡ DP must be 0 to access DAGEN registers.

表7−5. ストア タイプ命令

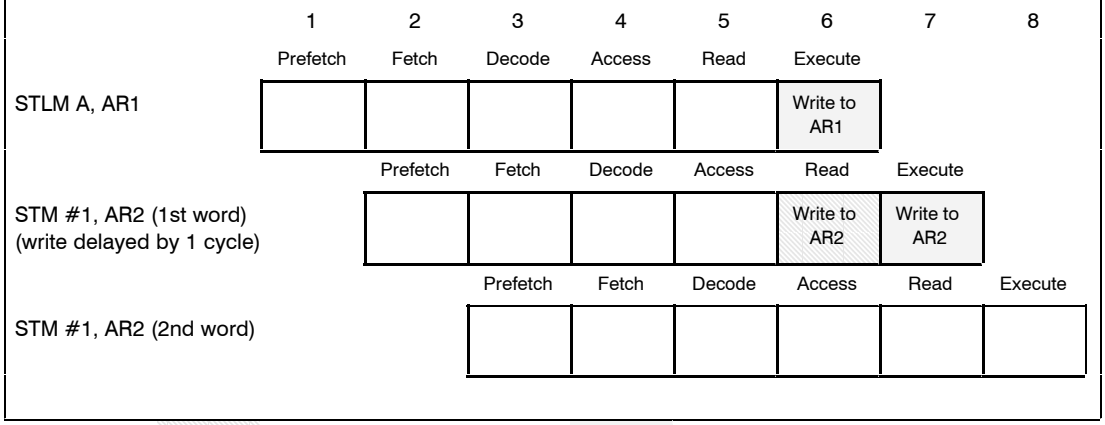
Instruction		Instruction		Instruction	
MVKD	dmad, Smem	STL	src, SHFT, Xmem	SRCCD	Xmem, cond
MVDM	dmad, MMR	STL	src, SHIFT, Smem	STRCD	Xmem, cond
MVPD	dmad, Smem	ST || ADD		CMPS	src, Smem
STH	src, Smem	ST || LD		ST	T, Smem
STH	src, ASM, Smem	ST || LT		ST	TRN, Smem
STH	src, SHFT, Xmem	ST || MAC[R]		ADDM	Smem, #lk
STH	src, SHIFT, Smem	ST || MAS[R]		ANDM	Smem, #lk
STLM	src, MMR	ST || MPY		ORM	#lk, Smem
STL	src, Smem	ST || SUB		XORM	Smem, #lk
STL	src, ASM, Smem	SACCD	src, Xmem, cond		

ストア タイプ命令の直後に、リード ステージでARx、BK、SPを更新する命令が続く場合は、コンフリクトが起こる可能性があります。これは、双方の命令がDAGENレジスタにアクセスしようとするからです。DAGENレジスタ セットに対するライトはCPUの1つのサイクルで1回のみ実行でき、CPUはリード ステージのアクセスを1サイクル遅らせます。このアクセスは、2番目の命令がパイプラインの実行ステージにあるとき実行します。通常、これによってその命令の実行時間は影響されません。例7−22、例7−23、例7−24はこのコンフリクトを示しています。

例7−22. **複数**ARx**の更新におけるコンフリクトの解決**

(a) **実行ステージで**AR1**を更新、リード ステージで**AR2**を更新**

```
STLM   A, AR1    ; This instruction updates AR1 in the Execute stage.
STM    #1, AR2   ; This instruction tries to update AR2 in its read stage,
                 ; creating a conflict with the previous instruction.
                 ; The AR2 update is delayed by one cycle.
```

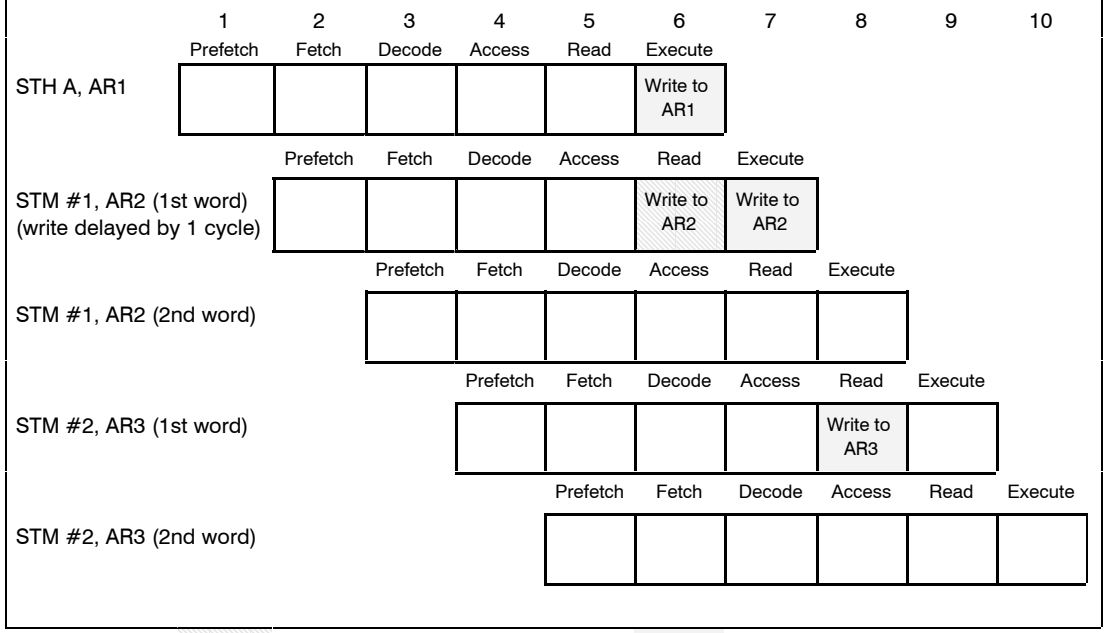


Legend: Where a write conflict occurs　Where the write actually occurs

例7−22. **複数の**ARx**更新時のコンフリクトの解決(続き**)

(b) **実行ステージで**AR1**を更新、リード ステージで**AR2**を更新、リード ステージで**AR3**を更新**

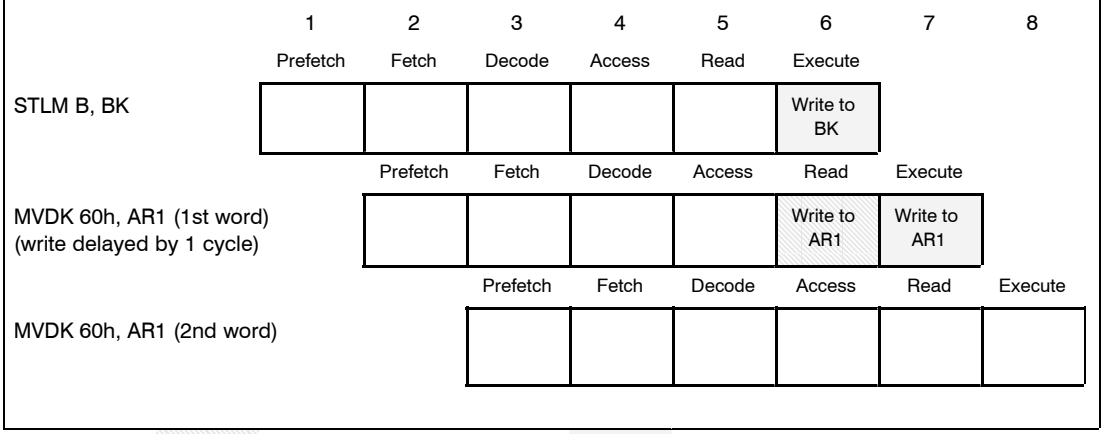
```
    STH    A, AR1     ; This instruction updates AR1 in the execute stage.
    ;
    STM    #1, AR2    ; This instruction tries to update AR2 in its read stage,
                      ; causing a conflict. The update is delayed by one cycle.
    ;
    STM    #2, AR3    ; This instruction updates AR3 in its read stage. It
                      : creates no conflict since the previous instruction was
                      ; a two-word instruction.
```



Legend: Where a write conflict occurs Where the write actually occurs

例7−23. ARx**および**BK**更新時のコンフリクトの解決**

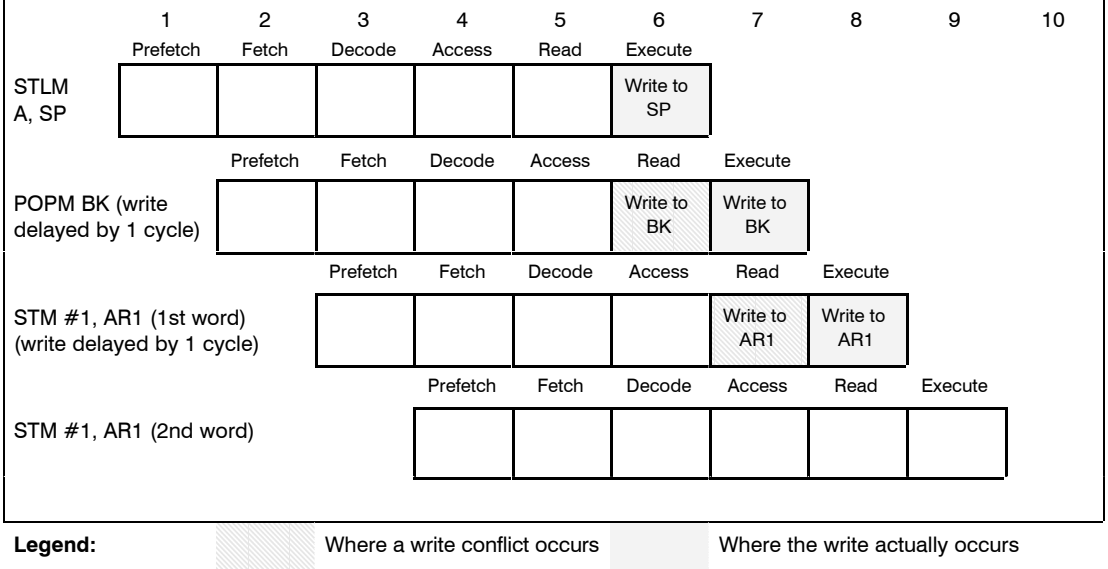
```
STLM   B, BK     ; This instruction updates BK in the execute stage.
MVDK   60h, AR1  ; This instruction tries to update AR1 in it read
                 ; stage. The CPU delays this update by one cycle to
                 ; resolve the conflict.
```



Legend: Where a write conflict occurs　Where the write actually occurs

例7−24. SP、BK、ARx**更新時のコンフリクトの解決**

```
    STLM    A, SP       ; This instruction updates SP in the execute stage.
    ;
    POPM    BK          ; This instruction tries to update BK in its read stage.
                        ; The CPU delays this update by one cycle.
    ;
    STM     #1, AR1     ; This instruction tries to update AR1 in its read stage,
                        ; conflicting with the previous instruction.
                        ; The update is delayed by one cycle.
```



これらのコンフリクトは、’54x CPU**が自動的に解決します。通常、これにより命令実行の動作には影響ありません。ただし、**CPU**による解決が保護されていないパイプライン コンフリクトを招くケースがひとつあります。これについては、**7.5.3DAGEN**レジスタ アクセス コンフリクトの判断規則を参照してください。**

7.5.3 DAGENレジスタ アクセス コンフリクトの判断規則

一部の命令は、リード ステージでDAGENレジスタを更新します。この結果、前の命令がその実行ステージでDAGENレジスタを更新しようとするとコンフリクトが発生する場合があります。このコンフリクトはリード ステージの更新を1サイクル遅らせることでCPUが自動的に解決します。この遅延は、リード ステージでDAGENレジスタにライトを行う命令と、それに続く命令の間で余分なレイテンシ サイクルを発生する原因にもなります。

以下は、このようなコンフリクトがいつ発生するかを判断する条件です。

- 最初の命令が次の2タイプのいずれか。
  - DAGENレジスタのいずれかにアクセスして新しい値をロードするストア タイプ命令(7−42ページの表7−5参照)。
  - 前の命令とのDAGENレジスタ コンフリクトするMove Type1命令(表7−4参照)。
- 2番目の命令が、BK、SP、またはいずれかのARxにライトを行う、定数イニシャライズ タイプ命令、 Move Type1命令、またはMove Type 2命令(表7−4参照)。その命令はロング オフセット　修飾子を使用しません(例7−27参照)。
- 3番目の命令が、間接アドレッシング モードで、2番目の命令と同じレジスタを使用する。

7.5.4 ARxおよびBKのレイテンシ

補助レジスタまたはBKへのアクセス時に次の条件が両方とも適合する場合、保護されていないパイプラインのコンフリクトが発生する可能性があります。

- 命令が補助レジスタまたはBKにライトを行う。
- 次の命令が、間接アドレッシング モードでアドレス ポインタまたはインデックスとして同じ補助レジスタを使うか、BKをサーキュラ アドレッシング モードで使う。この命令は、BKまたは同じARxをリードするMVMMまたはCMPRの場合もあります。

このコンフリクトが発生するのは、最初の命令がARxまたはBKをパイプラインのリードまたは実行ステージで更新して、次の命令がパイプラインのアクセス ステージのときBKまたは同じARxを使うからです。この結果、前の命令がまだレジスタの内容を更新していないときに、2番目の命令の誤ったARxまたはBKリードが起こります。

特定の命令(表7−6参照)では、ARxの更新時にレイテンシがありません。可能な限りこのような命令を使って、パイプラインのコンフリクトを防ぎます。

表7−6. ARx更新のためのパイプライン保護命令

To do this:	Use this instruction:	
Write an immediate value to ARx	STM	#lk, MMR†
Copy a memory location to ARx	MVDK	Smem, MMR†
Copy the contents of an ARx to another ARx	MVMM	MMR, MMR

† これらの命令でも発生するコンフリクトについては表7−7を参照。

STMおよびMVDKは、2つの理由から次の命令とはコンフリクトしません。

- ❑ これらは2ワード命令。
- ❑ これらは、最初の命令ワードがパイプラインのリード ステージで、ARxを更新する。

表7−7は、ARxを更新して使う命令間のレイテンシを示しています。表の2番目および3番目の命令は、同じ補助レジスタまたはBKにアクセスするとき、必ずレイテンシを発生します。表に示されない命令には、レイテンシはありません。

表7−8は、BKを更新して使用する命令間のレイテンシを示しています。

注：

レイテンシに合わせるために必要に応じて命令を並べ替えたりNOPを挿入するのは、プログラマの責任で行ってください。

表7−7. ARx アクセスのレイテンシ

(a) 3番目の命令カテゴリに基ずくレイテンシ

2番目の命令§		3番目の命令 カテゴリI	3番目の命令 カテゴリII
STM ST	#lk, auxreg #lk, auxreg	0†	0
MVDK MVMD	Smem, auxreg MMR, auxreg	0†	0
MVKD MVDM MVPD	dmad, auxreg dmad, auxreg pmad, auxreg	1	0
POPM POPD DELAY LTD MVDD	auxreg auxreg auxreg auxreg Xmem, auxreg	1†	0†
Store-type instructions (see 表7−5)		2	1

(b) 3番目の命令のカテゴリ

カテゴリI			カテゴリII		
MVMM	auxreg, MMR				
CMPR	CC, auxreg				
MVKD	dmad, auxind	With a long-offset modifier	MVKD	dmad, auxind	Without a long-offset modifier
MVDM	dmad, auxind		MVDM	dmad, auxind	
MVPD	pmad, auxind		MVPD	pmad, auxind	
MACP	auxind, pmad, src		MACP	auxind, pmad, src	
MACD	auxind, pmad, src		MACD	auxind, pmad, src	
ADD	auxind, shift, src, dst	With an extended shift¶ and a long-offset modifier	ADD	auxind, shift, src, dst	With an extended shift¶ and without a long-offset modifier
LD	auxind, shift, dst		LD	auxind, shift, dst	
STH	src, shift, auxind		STH	src, shift, auxind	
STL	src, shift, auxind		STL	src, shift, auxind	
SUB	auxind, shift, src, dst		SUB	auxind, shift, src, dst	
BANZ[D]	auxind	With or without a long-offset modifier	FIRS		With one operand using indirect addressing mode with or without a long-offset modifier
Instructions not listed here that use ARx in indirect addressing mode.					

† 最初の命令がDAGENレジスタのコンフリクト基準に適合する場合、レイテンシ サイクルをもう1つ加えます。詳しくは、サブセクション7.5.3DAGENレジスタ アクセス コンフリクトの判断規則を参照してください。

注： 1) 2つのカテゴリのいずれにも当てはまらない命令は、レイテンシがありません。

2) 最初の命令は、'54x命令のいずれも可能です。

§ 格納先オペランドauxregは、直接または間接アドレッシング モードのどちらでも必ずAR0−AR7をポイントします。オペランドauxindは、必ず間接アドレッシング モードを使用します。

¶ 値を−16から5の間でシフトします。

表7−8. BKアクセスのレイテンシ

(a) 3番目の命令のカテゴリに基ずくレイテンシ

2番目の命令§		3番目の命令 カテゴリI	3番目の命令 カテゴリII
STM ST	#lk, bkreg #lk, bkreg	1†	0†
MVDK MVMD	Smem, bkreg MMR, bkreg	1†	0†
MVKD MVDM MVPD	dmad, bkreg dmad, bkreg pmad, bkreg	2	1
POPM POPD DELAY LTD MVDD	bkreg bkreg bkreg bkreg Xmem, bkreg	2†	1†
Store-type instructions (see 表7−5)		3	2

(b) 3番目の命令のカテゴリ

カテゴリI			カテゴリII		
MVKD MVDM MVPD MACP MACD	dmad, circind dmad, circind pmad, circind circind, pmad, src circind, pmad, src	With a long-offset modifier	MVKD MVDM MVPD MACP MACD	dmad, circind dmad, circind pmad, circind circind, pmad, src circind, pmad, src	Without a long-offset modifier
ADD LD STH STL SUB	circind, shift, src, dst circind, shift, dst src, shift, circind src, shift, circind circind, shift, src, dst	With an extended shift¶ and a long-offset modifier	ADD LD STH STL SUB	circind, shift, src, dst circind, shift, dst src, shift, circind src, shift, circind circind, shift, src, dst	With an extended shift¶ and without a long-offset modifier
BANZ[D] circind		With or without a long-offset modifier	FIRS		With one operand using indirect addressing mode with or without a long-offset modifier
Instructions not listed here that use BK in circular addressing mode.					

† 最初の命令がDAGENレジスタのコンフリクト基準に適合する場合、レイテンシ サイクルをもう1つ加えます。詳しくは、サブセクション7.5.3DAGENレジスタ アクセス コンフリクトの判断規則を参照してください。

注： 1) 2つのカテゴリのいずれにも当てはまらない命令は、レイテンシがありません。

2) 最初の命令は、'54x命令のいずれも可能です。

§ 格納先オペランドbkregは、直接または間接アドレッシング モードのどちらでも必ずBKをポイントします。オペランドcircindは、必ずサーキュラ アドレッシング モードを使用します。

¶ 値を−16から15の間でシフトします。

例7−25. **レイテンシなしの**ARx**更新**

(a)

```
ADD    A, B
STM    #100h, AR3    ; This instruction does not conflict
                     ; with the previous instruction.
LD     *AR3+, A      ; No latency is required to use AR3.
```

(b)

```
ADD    A, B          ; This instruction does not create
                     ; a DAGEN conflict.
MVDK   60h, AR7      ; This instruction has zero latency.
STH    B, *AR7+
```

(c)

```
STLM   A, AR1        ; This instruction updates AR1 in
                     ; the execute stage, possibly
                     ; creating a DAGEN conflict.
MVDK   *(200h),AR2   ; However, this instruction uses a
                     ; long offset modifier. Therefore,
                     ; it creates no DAGEN conflict.
MAR    *AR2+         ; No latency is required to use AR2.
```

例7−26. 1**サイクルのレイテンシをともなう**ARx**更新**

(a)

```
ADD    A, B          ; This instruction does not create
                     ; a DAGEN conflict.
POPM   AR3           ; This instruction has a 1-cycle
                     ; latency if AR3 is used in the next
NOP                  ; instruction. The NOP is inserted
LD     *AR3+, A      ; to avoid the conflict.
```

(b)

```
STLM   A, AR1        ; This instruction updates AR1 in
                     ; the execute stage.
POPM   BK            ; This instruction tries to update
                     ; BK in the read stage. The CPU
                     ; delays the update by one cycle.
STM    #1, AR2       ; This instruction tries to update
NOP                  ; AR2 in the read stage. The CPU
                     ; delays this update by one cycle.
LD     *AR2+, B      ; This is why one NOP is required.
```

例7−27. 1**サイクルのレイテンシありまたはなしの**ARx**更新**

(a) 1**サイクルのレイテンシをともなう**ARx**更新**

```
STLM   A, AR1        ; This instruction creates DAGEN
                     ; conflict.
MVDK   60h, AR2      ; The AR2 update is delayed by one
NOP                  ; cycle.
MAR    *AR2+
```

(b) **命令順を変えてレイテンシなしの**ARx**更新**

```
MVDK   60h, AR2      ; This instruction does not require
                     ; any latency.
STLM   A, AR1        ; This instruction is placed after
                     ; MVDK to avoid a DAGEN conflict.
MAR    *AR2+         ; No latency is required now.
```

例7−28. 2**サイクルのレイテンシありまたはなしの**ARx**更新**

(a) 2**サイクルのレイテンシをともなう**ARx**更新**

```
STLM   A, SP         ; This instruction creates a DAGEN
                     ; conflict.
POPM   AR1           ; This instruction has a 2-cycle
NOP                  ; latency due to the DAGEN conflict.
NOP
LD     *AR1+, A      ; Two NOPs avoid this conflict.
```

(b) **命令順を変えてレイテンシなしの**ARx**更新**

```
POPM   AR1           ; This instruction has a 1-cycle
                     ; latency.
STLM   A, SP         ; This instruction is placed after
                     ; the POPM to avoid DAGEN conflicts
                     ; and to eliminate the need for
                     ; NOPs.
LD     *AR1+, A
```

例7−29. 2**サイクルのレイテンシをともなう**ARx**更新**

```
ADD A, B             ; This instruction does not create a
                     ; DAGEN conflict.
STLM   A, AR1        ; This instruction has a 2-cycle
NOP                  ; latency.
NOP
MVMM   AR1, AR2
```

例7−30. 1サイクルのレイテンシをともなうBK更新

```
ADD    A, B          ; This instruction does not create a
STM    #100h, BK     ; DAGEN conflict.
NOP                  ; This instruction needs a 1-cycle
                     ; latency.
ADD    *AR1+%, B     ; This instruction uses BK for
                     ; circular addressing
```

7.5.5 スタック ポインタのレイテンシ

このサブセクションで説明するスタック ポインタ(SP)のレイテンシは、SPを次の2つの方法のいずれかで使用するときに発生します。

- ❑ 直接アドレッシングにおけるオフセットとして(CPL=1のとき)。
- ❑ プッシュ、ポップ、コール、リターン、FRAMEまたはMVMM実行の中。

7.5.5.1 コンパイラ モードでのSPの使用(CPL=1)

次の2つの条件が同時に合うとき、パイプラインのコンフリクトが発生する場合があります。

- ❑ ひとつの命令がSPにライト。
- ❑ 次の命令が、コンパイラ モードで(CPL=1)SPを直接アドレッシングのベース アドレスとして使用。

コンフリクトが発生するのは、2番目の命令が、前の命令がSPを更新する前のパイプラインステージでSPを使おうとするからです。

表7−9は、コンパイラ モードでSPを更新する命令とそれ以降で使用する命令間のレイテンシを示しています。

> **注:**
>
> SPのレイテンシに合わせるために必要に応じて命令を並べ替えたりNOPを挿入するのは、プログラマの責任で行ってください。

表7−9. **コンパイラ モードでの**SP**のレイテンシ**(CPL=1)

(a) 3**番目の命令のカテゴリに基ずくレイテンシ**

		3番目の命令	
2番目の命令		**カテゴリI**	**カテゴリII**
STM ST	#lk, SP #lk, SP	1†	0†
MVDK MVMD	Smem, SP MMR, SP	1†	0†
MVKD MVDM MVPD	dmad, SP dmad, SP pmad, SP	2	1
MVDD POPM POPD	Xmem, spind SP SP	2†	1†
FRAME MVMM POPM POPD PSHM PSHD RETFD	k MMR, SP MMR Smem MMR Smem	1	0
ストア タイプ命令(表7−5**参照)**		3	2

(b) 3**番目の命令のカテゴリ**

カテゴリI	**カテゴリII**
カテゴリII に示されるものを除く、直接アドレッシング モードでSP**を使用するすべての命令**	MVKD dmad, dirmem MVDM dirmem, MMR MVPD pmad, dirmem MACP dirmem, pmad, src MACD dirmem, pmad, src
	ADD dirmem, shift, src, dst LD dirmem, shift, dst STH src, shift, dirmem STL src, shift, dirmem SUB dirmem, shift, src, dst } **拡張シフトをともなう**

記号: SP **直接または間接アドレッシング モードのいずれかでスタック ポインタを指す格納先オペランド**
MMR SP**を除くあらゆるメモリ マップド レジスタ**
spind **間接アドレッシング モードでスタック ポインタを指す行き先オペランド**
dirmem **コンパイラ モード**(CPL=1)**で直接アドレッシング モードを使用するオペランド**

注: 1. **いずれのカテゴリにも合わない命令には、レイテンシはありません。**
2. **最初の命令は、**'54x**のいずれの命令も可能です。**

† **最初の命令が**DAGEN**レジスタのコンフリクト基準に適合する場合、レイテンシ サイクルをもう**1**つ加えます。詳しくは、サブセクション**7.5.3 **DAGEN**レジスタ アクセス コンフリクトの判断規則を参照してください。**

‡ **値を**−16**から**15**の間でシフトします。**

例7−31. **コンパイラ モード**(CPL=1)**でのレイテンシなしの**SP**ロード**

(a)

```
ADD    A, B       ; This instruction does not create a
                  ; DAGEN conflict.
MVDK   60h, SP    ; This SP update requires a zero
                  ; latency according to the above table.
ADD    50h,-3,A,B
```

(b)

```
STLM   A, AR2     ; This instruction does not affect
                  ; pipeline latency.
POPM   AR1        ; This SP update requires a zero
                  ; latency according to the table.
MVKD   100h,1h
```

例7−32. **コンパイラ モード**(CPL=1)**での**1**サイクル レイテンシをともなう**SP**ロード**

(a)

```
STLM   A, AR2     ; This instruction does not affect
                  ; pipeline latency.
MVMM   AR1, SP    ; This SP update requires a one-cycle
                  ; latency since the next instruction
NOP               ; uses SP when CPL = 1.
LD     50h,A
```

(b)

```
ADD    A, B       ; This instruction does not create a
                  ; DAGEN conflict
STM    #100h, SP  ; This SP update requires a one-cycle
                  ; latency since the next instruction
NOP               ; uses SP when CPL = 1.
LD     50h,A
```

(c)

```
ADD    A,B        ; This instruction does not affect
                  ; pipeline latency.
RETFD             ; SP is incremented after popping the
                  ; return address.
NOP
LD     50h, A     ; This instruction cannot be placed in
                  ; the first delay slot since it uses
                  ; direct addressing mode with the new
                  ; SP value.
```

例7−33. **2サイクルのレイテンシありまたはなしのSPロード**

(a) **2サイクルのレイテンシをともなうSPロード**

```
STLM   A, BK      ; This instruction creates a DAGEN
                  ; conflict.
MVDK   60h, SP    ; This SP update requires 2 cycles
NOP               ; of latency according to the above
NOP               ; table.
LD     50h, A
```

(b) **レイテンシなしのSPロード**

```
MVDK   60h, SP    ; This SP update requires 1 cycle
                  ; of latency.
STLM   A, BK      ; This instruction is placed after the
                  ; MVDK instruction to prevent a DAGEN
                  ; conflict.
LD     50h, A     ; No NOPs are required in this case.
```

例7−34. **コンパイラ モード**(CPL=1)**での2サイクル レイテンシをともなうSPロード**

```
ADD    A, B       ; This instruction does not affect
                  ; pipeline latency.
POPM   SP         ; The new value of SP is popped from
NOP               ; the stack. Two NOPs are required to
NOP               ; use the new value of SP.
LD     50h, A
```

例7−35. **コンパイラ モード**(CPL=1、DP=0)**での3サイクル レイテンシをともなうSPロード**

```
STLM   A, AR1     ; This instruction does not affect
                  ; pipeline latency.
STH    A, SP      ; This SP update requires a three-cycle
NOP               ; latency since the next instruction
NOP               ; uses SP when CPL is 1.
NOP
LD     50h,A
```

7.5.5.2 プッシュ、ポップ、コール、リターン、FRAME、MVMM実行でのSPレジスタの使用

次の2つの条件が同時に満たされると、パイプラインのコンフリクトが発生します。

- ひとつの命令がSPレジスタを更新する。
- 次の命令が、プッシュ、ポップ、コール、リターン、FRAMEまたはMVMM実行のためにスタックを使用する。

コンフリクトが発生するのは、2番目の命令がパイプラインのあるステージでSPレジスタを使おうとし、その後のステージで前の一番目の命令がSPレジスタを更新するからです。

表7−10はCPL=0(非コンパイラ モード)の時、SP変更の際にレイテンシが無い命令をリストしています。これらの命令はコンフリクトを避け、どの位置にも使用することが可能です。

表7−10. 非コンパイラ モード(CPL=0)でSPレジスタを更新するパイプライン保護命令

To do this:	Use this instruction:	
Write an immediate value to SP	STM	#lk, SP†
Copy a memory location to SP	MVDK	Smem, SP†
Copy the contents of an ARx or BK to SP	MVMM	MMR, SP
Move SP by a frame	FRAME	k

† これらの命令で考えられるコンフリクトについては表7−11を参照してください。

表7−11には、非コンパイラ モード(CPL=0)でSPレジスタを更新して使用する命令間のレイテンシを示しています。

> 注:
>
> SPレジスタのレイテンシに合わせるために必要に応じて命令を並べ替えたりNOPを挿入するのは、プログラマの責任で行ってください。

表7−11. 非コンパイラ モード(CPL=0)でのSPレジスタのレイテンシ

(a) 3番目の命令の各カテゴリに基ずくレイテンシ

		Third Instruction	
Second Instruction		**Category I**	**Category II**
STM ST	#lk, SP #lk, SP	0†	0
MVDK MVMD	Smem, SP MMR, SP	0†	0
MVKD MVDM MVPD	dmad, SP dmad, SP pmad, SP	1	0
MVDD POPM POPD	Xmem, spind SP SP	1†	0†
Store-type instructions (see 表7−5)		2	1

(b) 3番目の命令のカテゴリ

Category I			**Category II**	
PSHM PSHD POPM PSHM	MMR Smem MMR Smem	Without a long-offset modifier	PSHM MMR PSHD Smem POPM MMR PSHM Smem	With a long-offset modifier
CALL[D] CC[D] FCALL[D] FRET[D] FRETE[D] INTR RC[D] RET[D] RETE[D] RETF[D]	address address k 		CALA[D] address FCALA[D]	
MVMM	SP, MMR			
FRAME TRAP	k n			

記号: SP 直接または間接アドレッシング モードでスタック ポインタを指すオペランド
MMR SPレジスタを除くあらゆるメモリ マップド レジスタ
SPind 間接アドレッシング モードでスタック ポインタを指すオペランド

注: 1. いずれのカテゴリにも合わない命令には、レイテンシはありません。
2. 1番目の命令は、'54xのどんな命令も可能です。

† 1番目の命令がDAGENレジスタのコンフリクト基準に適合する場合、レイテンシ サイクルをもう1つ加えます。詳しくは、サブセクション7.5.3DAGENレジスタ アクセス コンフリクトの判断規則を参照してください。

例7−36. **非コンパイラ モード**(CPL=0)**でのレイテンシなしの**SP**レジスタのロード**

(a)

```
ADD    A, B       ; This instruction does not create
                  ; a DAGEN conflict
STM    #100h, SP  ; This SP update does not require any
                  ; latency according to the above table.
PSHM   AR1
```

(b)

```
STH    A, 60h ; This instruction does not create
                  ; a DAGEN conflict.
MVDK   7Fh, SP    ; This SP update does not require any
                  ; latency according to the above table.
FRAME  10
```

例7−37. **非コンパイラ モード**(CPL=0)**で**1**サイクル レイテンシありまたはなしの**SP**レジスタのロード**

(a) 1**サイクル レイテンシをともなう**SP**レジスタのロード**

```
STLM   A, AR1     ; This instruction causes a DAGEN
                  ; conflict with the next instruction.
MVDK   60h, SP    ; This SP update requires a one-cycle
NOP               ; latency.
PSHM   AR2
```

(b) **レイテンシなしの**SP**レジスタのロード**

```
MVDK   60h, SP    ; This instruction requires no latency.
STLM   A, AR1     ; This instruction was placed after
                  ; MVDK to avoid a DAGEN conflict.
PSHM   AR2
```

例7−38. **非コンパイラ モード**(CPL=0)**で**1**サイクル レイテンシをともなう**SP**レジスタのロード**

```
STLM   A, AR1     ; This instruction does not affect the
                  ; pipeline latency for the next
                  ; instruction.
STLM   B, SP      ; This SP update requires a one-cycle
NOP               ; latency.
CALA   A
```

7.5.6 テンポラリ レジスタのレイテンシ

Tレジスタにアクセスするとき次の2つの条件が同時に満たされると、パイプラインのコンフリクトが発生する場合があります。

- ある命令がTレジスタにライト
- 次の命令がシフトまたはビット テスト実行のためにTレジスタを使う。

コンフリクトが発生するのは、2番目の命令がパイプラインのあるステージでTレジスタを使おうとし、その後のステージで1番目の命令がTレジスタを更新するからです。

表7−12は、Tレジスタの更新でレイテンシのない命令を示しています。これらの命令を可能な限り使用して、コンフリクトを回避します。

表7−12. Tレジスタを更新するパイプライン保護命令

To do this:	Use this instruction:
Write an immediate value to T	STM #lk, T
Copy a memory location to T	MVDK Smem, T
Copy a memory location to T	LD Smem, T
Copy a memory location to T	ST src, Ymem || LD Xmem, T

表7−13は、Tレジスタを更新しながら使用する命令間のレイテンシを示しています。

注:

Tレジスタのレイテンシに合わせるために必要に応じて命令を並べ替えたりNOPを挿入するのは、プログラマの責任で行ってください。

表7−13. 2番目の命令のカテゴリに基ずくTレジスタのレイテンシ

(a) 2番目の命令のカテゴリに基づくレイテンシ

First Instruction		Second Instruction Category I
MVKD MVDM	dmad, T dmad, T	1
POPM POPD DELAY MVDD	T T T Xmem, T_{ind}	1
Store-type instructions (see 表7−5)		1
EXP	src	1

(b) 2番目の命令のカテゴリ

Category I		
LD ADD SUB	Smem, TS, dst Smem, TS, src Smem, TS, src	Without a long-offset modifier
NORM	src, dst	
BITT	Xmem	
DADST	Lmem, dst	
DSADT	Lmem, dst	
DSUBT	Lmem, dst	

記号: T 直接または間接アドレッシング モードでTレジスタを指すオペランド。
T_{ind} 間接アドレッシング モードでTレジスタを指すオペランド。

注: カテゴリIに合わない命令には、レイテンシはありません。

例7−39. **レイテンシをともなわないT レジスタのロード**

(a)

```
    LD     *AR3+, T      ; This T update does not require
    LD     *AR5+,TS,A    ; any latency.
```

(b)

```
    STM    #100h, T      ; This T update does not require
    LD     *AR5+,TS,A    ; any latency.
```

例7−40. 1**サイクル レイテンシをともなうT レジスタのロード**

(a)

```
    POPM   T             ; This instruction requires a one-
    NOP                  ; cycle latency.
    BITT   *AR5+
```

(b)

```
    EXP    A             ; This instruction requires a one-
    NOP                  ; cycle latency.
    NORM   A
```

7.5.7 ステータス レジスタへのアクセスの際のレイテンシ

次のステータス レジスタ フィールドおよびビットはレイテンシにより影響があります。

- ARP (補助レジスタ ポインタ)
- CMPT(互換モード ビット)
- CPL(コンパイラ モード ビット)
- DP(データ ページ ポインタ)
- SXM(符号拡張モード ビット)
- ASM(アキュムレータ シフト モード ビット)
- BRAF(ブロック リピート アクティブ フラグ)

7.5.7.1 ST1とリピート ブロック ループ

命令には、パイプラインの実行ステージでST1にライトを行うものがあります。このような命令の直後に、ST1中のBRAFフラグをセットするRPTB[D]命令が続く場合、不完全なリピート ブロック ループが発生します。これが発生するのは、RPTB[D]がBRAFをパイプラインのアクセス ステージでセットして、その前の命令がST1を1サイクル後に上書きするからです。

このコンフリクトを回避するには、表7−14に示されている命令を使ってRPTB[D]命令の直前にST1にライトを行うことをお勧めします。他のST1にライトを行う命令の直後には、RPTB[D]命令を続けてはいけません。

表7−14. ST1ライトに推奨される命令

To do this	Use this instruction	
Store a value to ST1	STM ST	#k, ST1 #k, ST1
Copy a value from data memory to ST1	MVDK MVMD	Smem, ST1 dmad, ST1
Clear a bit in ST1	RSBX	
Set a bit in ST1	SSBX	
Load ASM with a value	LD LD	#k, ASM Smem, ASM

7.5.7.2 互換モード(CMPT=1)でのARPおよびCMPTビットの更新

次の2つの条件が同時に満たされると、パイプラインのコンフリクトが発生する場合があります。

- ❑ ひとつの命令がARPまたはCMPTを更新する。
- ❑ 次の命令が、間接アドレッシング モードでARPまたはCMPTを使ってアドレス ポインタを更新する。

コンフリクトが発生するのは、2番目の命令がパイプラインのステージでARPまたはCMPTを使って、その後のステージで1番目の命令がARPまたはCMPTを更新するからです。

表7−15に示される命令は、CPUが互換モードのときARPを更新するのにレイテンシがない命令です。可能な限りこの命令を使って、コンフリクトを回避します。

表7−15. 互換モード(CMPT=1)でARPを更新するためのパイプライン保護命令

To do this:	Use this instruction:
Load ARP field of ST0 register	LD #k, ARP

表7−16は、ARPまたはCMPTを更新しながら使用する命令間のレイテンシを示しています。

注 :

1) レイテンシに合わせるために必要に応じて命令を並べ替えたりNOPを挿入するのは、プログラマの責任で行ってください。
2) 互換モード(CMPT=1)では、ARPは間接アドレッシング モードを使う命令で自動的に更新されます。このようなARP更新ではパイプラインのコンフリクトはありません。
3) ARPは、DSPが標準モード(CMPT=0)のとき必ず0にリセットされます。リセット時に、ARPおよびCMPTは自動的に0にリセットされます。

表7−16. 互換モード(CMPT=1)でのARPのレイテンシおよびCMPTビット

(a) 2番目の命令の各カテゴリに基づくレイテンシ

		Second Instruction	
First Instruction		**Category I**	**Category II**
STM ST	#lk, status #lk, status	2	2
MVDK MVMD	Smem, status MMR, status	2	2
MVKD MVDM MVPD	dmad, status dmad, status pmad, status	3	2
MVPD	pmad, status	3	3
POPM POPD DELAY LTD MVDD	status status status status status	3	2
ストア タイプ命令(表7−5参照)		3	2
SSBX RSBX	statbit statbit	3	2

(b) 2番目の命令のカテゴリ

Category I			**Category II**		
MVKD MVDM MVPD MACP MACD	dmad, auxind dmad, auxind dmad, auxind auxind, pmad, src auxind, pmad, src	With a long-offset modifier	MVKD MVDM MVPD MACP MACD	dmad, auxind dmad, auxind pmad, auxind auxind, pmad, src auxind, pmad, src	Without a long-offset modifier
ADD LD STH STL SUB	auxind, shift, src, dst auxind, shift, dst src, shift, auxind arc, shift, auxind auxind, shift, src, dst	With an extended shift† and a long offset modifier	ADD LD STH STL SUB	auxind, shift, src, dst auxind, shift, dst src, shift, auxind src, shift, auxind auxind, shift, src, dst	With an extended shift† and without a long-offset modifier
All other instructions that use ARP or CMPT in indirect addressing mode with or without a long offset modifier.					

記号: status 直接または間接アドレッシング モードそれぞれのARPまたはCMPTを更新するためのST0またはST1を指すオペランド
MMR あらゆるメモリ マップド レジスタ
auxind 間接アドレッシング モードを使うリードまたはライト オペランド
statbit ARPまたはCMPTのビットにライトを行うオペランド

注: いずれのカテゴリにも合わない命令には、レイテンシはありません。
† 値を−16から15の間でシフトします。

例7−41. **互換モード**(CMPT=1)**でのレイテンシなしの**ARP**ロード**

```
    LD     #1h, ARP  ; This ARP load does not require any
                     ; latency.
    LD     *AR0, A
```

例7−42. **互換モード**(CMPT=1)**での**2**サイクル レイテンシをともなう**ARP**ロード**

```
    STLM   A, ST0    ; The ARP field of ST0 is updated here.
    NOP
    NOP
    ADD    *AR0+,•3,B ; The new ARP value is used here
```

例7−43. **互換モード**(CMPT=1)**での**3**サイクル レイテンシをともなう**ARP**ロード**

```
    POPM   ST0       ; The ARP field of ST0 is updated here.
    NOP
    NOP
    NOP
    LD     *AR0+, A  ; The new ARP value is used here.
```


7.5.7.3 直接アドレッシング モード(CPL=0)でのDP更新

次の2つの条件が同時に満たされると、パイプラインのコンフリクトが発生する場合があります。

- ある命令がDPを更新
- 次の命令が、非コンパイラ モード(CPL=0)でDPを直接アドレッシングのベース アドレスとして使用する。

コンフリクトが発生するのは、2番目の命令がDPをパイプラインのあるステージで使用して、その後に前の1番目の命令がDPを更新するからです。

表7−17は、DPへのライト時にレイテンシのない命令を示しています。これらの命令を可能な限り使用してコンフリクトを回避することをお勧めします。

表7−17. **非コンパイラ モード**(CPL=0)**で**DP**を更新するための推奨命令**

To do this:	Use this instruction:	
Load an immediate number to DP	LD	#k, DP
Copy contents of a memory location to DP	LD	Smem, DP

表7−18は、DPを更新しながら使用する命令間のレイテンシを示しています。

注 :

レイテンシに合わせるために必要に応じて命令を並べ替えたりNOP**を挿入するのは、プログラマの責任で行ってください。**

表7−18. **非コンパイラ モード**(CPL=0)**で**DP**のレイテンシ**

(a) 2**番目の命令の各カテゴリに基づくレイテンシ**

		Second Instruction	
First Instruction		**Category I**	**Category II**
STM ST	#lk, status #lk, status	2	2
MVDK MVMD	Smem, status MMR, status	2	2
MVKD MVDM	dmad, status dmad, status	3	2
MVPD	pmad, status	3	3
POPM POPD MVDD	status status status	3	2
Store-type instruction (see **表**7−5)		3	2
SSBX RSBX	ST0, statbit ST0, statbit	3	2

(b) 2**番目の命令のカテゴリ**

Category I	**Category II**		
All instructions that use DP for direct addressing mode except those listed in Category II.	MVKD	dmad, dirmem	
	MVPD	pmad, dirmem	
	MACP	dirmem, pmad, src	
	MACD	dirmem, pmad, src	
	ADD	dirmem, shift, src, dst	With an extended shift†
	LD	dirmem, shift, dst	
	STH	src, shift, dirmem	
	STL	src, shift, dirmem	
	SUB	dirmem, shift, src, dst	

記号: status **直接または間接アドレッシング モードで**DP**を更新するための**ST0**を指すオペランド**
MMR **あらゆるメモリ マップド レジスタ**
statbit ARP**または**CMPT**のビットにライトを行うオペランド**
dirmem CPL=0**のとき直接アドレッシング モードを使うリードまたはライト オペランド**

注: **いずれのカテゴリにも合わない命令には、レイテンシはありません。**
† **値を**−16**から**15**の間でシフトします。**

例7−44. **非コンパイラ モード**(CPL=0)**でのレイテンシなしの**DP**ロード**

(a)

```
LD     #2h, DP   ; This DP load does not require any
                 ; latency.
LD     27h, A
```

(b)

```
LD     60h, DP   ; This DP load does not require any
                 ; latency.
LD     27h, A
```

例7−45. **非コンパイラ モード**(CPL=0)**での**2**サイクル レイテンシをともなう**DP**ロード**

```
STLM   A, ST0    ; The DP field of ST0 is updated here.
NOP
NOP
STH    B,-3,27h  ; The new DP value is used here.
```


例7−46. **非コンパイラ モード**(CPL=0)**での**3**サイクル レイテンシをともなう**DP**ロード**

```
POPM   ST0       ; The DP field of ST0 is updated here.
NOP
NOP
NOP
LD     27h, A    ; The new DP value is used here
```

7.5.7.4 CPLの更新

次の2**つの条件が同時に満たされたとき、パイプラインのコンフリクトが発生する場合があります。**

- **ある命令が**CPL**を変更。**
- **その次の命令が直接アドレッシング モードを使用。**

コンフリクトが発生するのは、パイプラインのあるステージで2**番目の命令が**CPL**を読み取って、その後に前の**1**番目の命令が**CPL**を更新するからです。**

表7−19**は、**CPL**を更新しながら使用する命令間のレイテンシを示しています。**

注:

レイテンシに合わせるために必要に応じて命令を並べ替えたりNOP**を挿入するのは、プログラマの責任で行ってください。**

表7−19. CPLビットのレイテンシ

(a) 2番目の命令の各カテゴリに基づくレイテンシ

First Instruction		Second Instruction: Category I	Second Instruction: Category II
STM ST	#lk, status #lk, status	2	1
MVDK MVMD	Smem, status MMR, status	2	1
MVKD MVDM	dmad, status dmad, status	3	2
MVPD	pmad, status	3	3
POPM POPD MVDD	status status status	3	2
Store-type instruction (see 表7−5)		3	2
SSBX RSBX	CPL CPL	3	2

(b) 2番目の命令のカテゴリ

Category I	Category II		
All instructions that use direct addressing mode except those listed in Category II.	MVKD	dmad, dirmem	
	MVPD	pmad, dirmem	
	MACP	dirmem, pmad, src	
	MACD	dirmem, pmad, src	
	ADD	dirmem, shift, src, dst	With an extended shift†
	LD	dirmem, shift, dst	
	STH	src, shift, dirmem	
	STL	arc, shift, dirmem	
	SUB	dirmem, shift, src, dst	

記号: MMR あらゆるメモリ マップド レジスタ
dirmem CPL=0のとき直接アドレッシング モードを使うリードまたはライト オペランド
status 直接または間接アドレッシング モードでCPLを更新するためのST1を指すオペランド

注: いずれのカテゴリにも合わない命令には、レイテンシはありません。
† 値を−16から15の間でシフトします。

例7−47. 1サイクル レイテンシをともなうCPL更新

```
STM    #k, ST1       ; This instruction modifies the CPL
                     ; bit of ST1.
NOP
MVKD   1000h, 30h    ; Data read from 1000h is written to
                     ; (SP + 30h)
```

例7−48. 2サイクル レイテンシをともなうCPL更新

```
RSBX   CPL           ; CPL is changed from 1 to 0.
NOP                  ; Two NOPs are required, since a LD
NOP                  ; with an extended shift is being
                     ; used to access an operand.
LD     27h, •1, A    ; The operand is read from the
                     ; current data page.
```

例7−49. 3サイクル レイテンシをともなうCPL更新

```
SSBX   CPL           ; CPL is changed from 0 to 1.
NOP                  ; These NOPs can be replaced by
NOP                  ; other instructions that do not use
NOP                  ; direct addressing mode to access
                     ; operands.
LD     27h, A        ; The operand is read at an offset
                     ; of 27h from SP.
```


7.5.7.5 SXMの更新

次の2つの条件が同時に満たされるとき、パイプラインのコンフリクトが発生する場合があります。

- ある命令がSXMを変更。
- 次の命令が、SXMを使って符号拡張を制御。

コンフリクトが発生するのは、2番目の命令がパイプラインのあるステージでSXMを使って、その後に前の1番目の命令がSXMを更新するからです。

表7−20は、SXMを更新して使用する命令間のレイテンシを示しています。

> 注:
>
> レイテンシに合わせるために必要に応じて命令を並べ替えたりNOPを挿入するのは、プログラマの責任で行ってください。

表7−20. SXM**ビットのレイテンシ**

(a) 2**番目の命令の各カテゴリに基づくレイテンシ**

First Instruction	Second Instruction Category I
MVKD dmad, status MVDM dmad, status POPM status POPD status MVDD Xmem, status	1
Store-type instruction (see **表**7−5)	1
SSBX SXM RSBX SXM	1

(b) 2**番目の命令のカテゴリ**

Category I
All instructions affected by the sign-extension mode bit, except those that require an Smem operand with a long offset modifier (for example, LD *+AR1 (100h), A)

記号: status **直接、間接、またはメモリ マップド アドレッシング モードで**SXM**を更新するための**ST1**を指すオペランド**

注: **いずれのカテゴリにも合わない命令には、レイテンシはありません。**

例7−50. **レイテンシをともなわない**SXM**更新**

```
RSBX   SXM    ; This instruction modifies the SXM bit of
              ; ST1.
ADD    *+AR1(100h),A
```

例7−51. 1**サイクル レイテンシをともなう**SXM**更新**

(a)

```
RSBX   SXM    ; This SXM load requires one cycle of
              ; latency.
NOP
LD     *AR5+, A
```

(b)

```
POPM   ST1    ; This instruction modifies the SXM bit of
NOP           ; ST1.
ADD    *AR2+,A
```

(c)

```
STLM   A, ST1 ; This instruction modifies the SXM bit of
              ; ST1.
NOP
SUB    *AR2-,A
```

7.5.7.6 シフト実行に使用するASMフィールド

次の2つの条件が同時に満たされるとき、パイプラインのコンフリクトが発生する場合があります。

- ❑ ある命令がASMを変更
- ❑ 次の命令がASMをシフト制御値として使用。

コンフリクトが発生するのは、2番目の命令がパイプラインのあるステージでASMをリードして、その後前の1番目の命令がASMを更新するからです。

表7−21は、ASMビット フィールドにライトを行うためのレイテンシのない命令を示しています。これらの命令を可能な限り使用して、コンフリクトを回避します。

表7−21. ASMライトのためのパイプライン保護命令

To do this:	Use this instruction:
Load an immediate number to ASM	LD #k, ASM
Copy contents of a memory location to ASM	LD Smem, ASM

表7−22は、ASMにライトを行ってそれを使用する命令間のラテンシーを示しています。

注 :

レイテンシに合わせるために必要に応じて命令を並べ替えたりNOPを挿入するのは、プログラマの責任で行ってください。

表7−22. ASMビット フィールドのレイテンシ

(a) 2番目の命令の各カテゴリに基づくレイテンシ

First Instruction	Second Instruction Category I
MVKD dmad, status MVDM dmad, status POPM status POPD status MVDD Xmem, status	1
Store-type instruction (see 表7−5)	1
SSBX STI, asmbit RSBX STI, asmbit	1

(b) 2番目の命令のカテゴリ

Category I	
STH src, ASM, Smem STL src, ASM, Smem	} Without a long-offset modifier
ST src, Ymem || LD/ADD/SUB/MAC/MAS/MPY	
SACCD src, Smem, cond	

LD src, ASM, dst
ADD src, ASM, dst
SUB src, ASM, dst	

記号: asmbit ST1のASMフィールドのビットにライトを行うオペランド
status 直接、間接、またはメモリ マップド アドレッシング モードでASMを更新するために、ST1を指すオペランド

注: カテゴリに合わない命令には、レイテンシはありません。

例7−52. **レイテンシをともなわない**ASM**更新**

(a)

```
    LD     #6, ASM    ; This instruction loads ASM with no
                      ; latency.
    STH    A,ASM,*AR1+
```

(b)

```
    LD     60h, ASM   ; This instruction loads ASM with no
                      ; latency
    ADD    A,ASM,B
```

(c)

```
    STLM   A, ST1     ; This instruction modifies the ASM
                      ; field of ST1. No latency is needed
                      ; since STL uses a long offset
                      ; modifier.
    STL    A,ASM,*+AR5(100h)
```


例7−53. 1**サイクル レイテンシをともなう**ASM**更新**

```
    POPM   ST1        ; This instruction modifies the ASM
    NOP               ; field of ST1
    SUB    A,ASM,B
```

7.5.8 リピート ブロック ループでのレイテンシ

次のステータス レジスタ フィールドおよびビットはレイテンシによって影響があります。

- ❑ BRC(ブロック リピート カウンタ レジスタ)
- ❑ BRAF(ブロック リピート アクティブ フラグ)

7.5.8.1 ブロック リピート カウンタ(BRC)レジスタの更新

次の2つの条件が同時に満たされるとき、パイプラインのコンフリクトが発生する場合があります。

- ❑ ある命令がBRCレジスタにライト。
- ❑ 次の命令はRPTB[D]。

コンフリクトが発生するのは、2番目の命令がパイプラインのあるステージでBRCをリードし、その後前の1番目の命令がBRCを更新するからです。

BRCを更新するとき、パイプラインのコンフリクトを起こさない命令があります。このような命令を可能な限り使用して、コンフリクトを回避します。

表7−23. RPTBループの前にBRCにライトを行うための推奨命令

To do this	Use this instruction	
Write an immediate value to BRC	STM	#lk, BRC
Copy a memory location to BRC	MVDK	Smem, BRC

表7−24は、BRCおよびRPTB[D]命令を更新する命令間のレイテンシを示しています。

注 :

1) BRCを変更する命令を、RPTB命令の遅延スロットに置かないでください。

2) レイテンシに合わせるために必要に応じて命令を並べ替えたりNOPを挿入するのは、プログラマの責任で行ってください。

表7−24. RPTBループのまえにBRCを更新する場合のレイテンシ

First Instruction	Latency if Second Instruction Is RPTB[D]
MVDK Smem, BRC MVMD MMR, BRC	0
STM #k, BRC ST #k, BRC	0
All other instructions that modify BRC	1

例7−54. 新たなリピート ブロック ループを実行する前のBRCロード

(a)

```
    STM    #lk,BRC       ; There is no latency when BRC is
    RPTB   endloop•1     ; loaded via STM before a new RPTB
    ...                  ; loop.
endloop:
```

(b)

```
    MVDK       count,BRC ; There is no latency when BRC is
    RPTBD      endloop•1 ; loaded using MVDK before a new
    ...                  ; RPTB loop
endloop:
```

(c)

```
    STLM   A,BRC         ; There is a 1 cycle latency when
    NOP                  ; BRC is loaded using an STLM
    RPTB   endloop•1     ; instruction.
    ...
endloop:
```

(d)

```
    POPM   BRC           ; There is a 1 cycle latency when
    NOP                  ; BRC is loaded using a POPM
    RPTBD  endloop•1     ; instruction.
    ...
endloop:
```

リピート ブロック ループでは、ループの最後の命令がパイプラインのデコード ステージにあるとき、BRCがデクリメントされます。ただし、SRCCD命令はBRCの内容をパイプラインの実行ステージでライトします。これにより、SRCCD命令が誤ったBRC値をライトすることがあります。例7−55および例7−56に示すように、パイプラインのコンフリクトは、SRCCD命令をループの最下段から最低3命令ワードより前の位置に置くことにより回避できます。

例7−55. レイテンシなしのSRCCD命令

```
    RPTB   endloop•1
    ...
    SRCCD  *AR3, ALEQ    ; Placing the SRCCD instruction in
                         ; this position ensures that current
                         ; value of BRC will be written
                         ; to memory.
    ADD    *AR1+,A
    SUB    *AR2•,A
    STH    A, *AR1+
endloop:
```

例7−56. 3**サイクルのレイテンシをともなう**SRCCD**命令**

```
    RPTB   endloop•1
    ...
    SRCCD  *AR3, ALEQ   ;This ensures that current value of
    NOP                 ; BRC will be written to memory.
    NOP
    NOP
endloop:
```

RPTBループ中で新しい値をBRCにライトするとき、5から6サイクルのレイテンシが発生します。表7−25には、RPTBループがアクティブでBRCが変更されるときのみあてはまるレイテンシを示しています。詳しくは、例7−57を参照してください。

表7−25. RPTB**ループから**BRC**を更新するためのレイテンシ**

Instruction	Latency
STM #lk, BRC ST #lk, BRC MVDK Smem, BRC MVMD MMR, BRC	The next 5 instruction words must not contain the last instruction in the RPTB loop.
All other instructions that modify BRC	The next 6 instruction words must not contain the last instruction in the RPTB loop.

例7−57. RPTB**ループからの**BRC**の更新**

```
    RPTB   endloop•1
    ...
    XC     2, Condition     ; If Condition is evaluated as
    MVDK   5h, BRC          ; true, write the new count value
                            ; to BRC.
    LD     *AR1+, A         ; These six instructions provide
    ADD    *AR2•, A         ; sufficient latency for the new
    SUB    *AR3+, A         ; BRC value to take effect before
    LD     *AR4+, T         ; the next iteration begins.
    MPYA   A
    STL    A, *AR3+
endloop:
```

7.5.8.2 ブロック リピート アクティブ フラグ(BRAF)を動作しない状態にする

'54xでは、ブロック リピート アクティブ フラグ(BRAF)**を1にセットすることにより、リピート ブロック ループがアクティブであることを示します。**BRAF**は、リピート ブロック ループの最初の命令のデコード ステージでセットされ、最後のループ繰り返しの間にクリアされます**(BBC=0)**。デバイスは**BRAF**を各繰り返しループの最後にテストして、次のプリフェッチがループのトップからするかどうかを判断します。**

BRAFをソフトウェアで動作しない状態にして、リピート ブロックを早めに終了できます。ただし、これはパイプラインで十分早めに行う必要があり、それによりBRAFはループ最上段における命令のプリフェッチ以前にクリアされます。したがって、BRAFをクリアする命令(RSBXなど)はリピート ブロック ループの最低6命令ワード前に置く必要があります。例7−58を参照してください。

例7−58. BRAFの非動作状態

```
    RPTB   endloop•1
    ...
    RSBX   BRAF       ; This ensures that the loop will
    NOP               ; terminate after completing this
    NOP               ; iteration.
    NOP
    NOP               ; These six NOPs may be replaced by
    NOP               ; other instructions.
    NOP
endloop:
```

これら6つのNOPは、他の命令と入れ替えできます。

7.5.9 PMSTレジスタのレイテンシ

PMSTフィールドOVLY、DROM、MP/$\overline{\text{MC}}$、IPTRは、'54xのメモリ空間を設定します。

ある命令がこれらのフィールドのどれかを変更するとき、新しいメモリ空間にアクセスする前に一定のパイプライン サイクル数をパスする必要があります。このレイテンシが必要なのは、PMSTフィールドがパイプラインのリードまたは実行ステージで更新されるとき、前のパイプライン ステージにある命令がそのメモリ空間にアクセスする場合があるからです。OVLY、IPTR、MP/$\overline{\text{MC}}$フィールドによってプログラム メモリを制御する場合、パイプラインのプリフェッチ ステージがこれらのビットを評価します。DROMフィールドによってデータ メモリを制御する場合、アクセスまたはリード ステージがこのビットを評価します。

外部メモリ アクセスが発生する場合、ビット変更に影響される処理のために追加サイクルが必要です。PMSTフィールドのどの変更も、処理中の外部アクセスが完了するまで遅らされます。たとえば、パイプラインの実行ステージの命令がOVLYを変更するとき外部メモリ アクセスが発生する場合、その更新は外部バス サイクルが完了するまで遅らされます。これにより、下記表に示された処理に追加サイクルが必要になります。

表7−26はOVLY、IPTR、MP/$\overline{\text{MC}}$ビット フィールドにライトを行う命令間のレイテンシ、および続いて新しいメモリ空間からフェッチされる命令間のレイテンシを示しています。

注:

レイテンシに合わせるために必要に応じて命令を並べ替えたりNOPを挿入するのは、プログラマの責任で行ってください。

表7−26. OVLY、IPTR、MP/$\overline{MC}$**ビットのレイテンシ**

(a) 2**番目の命令の各カテゴリに基づくレイテンシ**

	Second Instruction			
First Instruction	**Category I**	**Category II**	**Category III**	**Category IV**
STM #lk, pmst ST #lk, pmst MVDK dmad, pmst MVMD MMR, pmst	0	0	1	2
All other instructions that modify OVLY, IPTR, MP/$\overline{MC}$	0	1	2	3

(b) 2**番目の命令のカテゴリ**

Category I	**Category II**	**Category III**	**Category IV**
BACC[D]	BC[D]	B[D]	INTR
CALA[D]	CC[D]	BANZ[D]	RETF[D]
FRET[D]	RC[D]	CALL[D]	TRAP
FRETE[D]	RET[D]	FB[D]	
FBACC[D]	RETE[D]	FCALL[D]	
FCALA[D]			

記号: pmst **直接、間接、またはメモリ マップド アドレッシング モードで、**OVLY**、**IPTR**、**MP/$\overline{MC}$**を変更するために**PMST**を指すオペランド**

注: 1. **命令が**OVLY**、**IPTR**または**MP/$\overline{MC}$**ビット フィールドを変更するとき外部メモリ アクセスが進行中なら、追加レイテンシ サイクルが必要です。**
2. 2**番目の命令は、変更されたプログラム アドレス領域を指す新しい値を**PC**にロードします。**

例7−59. **条件なし分岐**(DP=0)**が続く**OVLY**設定**

```
ORM    #20h, PMST    ; This instruction sets OVLY to 1.
NOP
NOP
B      onchip        ; Branch to on-chip dual-access
                     ; memory.
```

例7−60. **例条件付き分岐が続く**OVLY**設定**

```
MVDK   5h, PMST      ; This instruction sets OVLY to 1.
BC     onchip,AEQ    ; Branch to on-chip dual-access
                     ; memory.
```

例7−61. **リターン**(DP=0)**が続く**OVLY**設定**

```
ANDM #0ffdfh, PMST  ; This instruction sets OVLY to 0.
NOP
RET                 ; Return to off-chip memory.
```

例7−62. **条件なし遅延コールが続く**MP/$\overline{\text{MC}}$**設定**

```
STLM   A, PMST      ;This instruction sets MP/MC to 1.
NOP
NOP
CALLD  offchip      ; Call a routine in external
                    ; program memory after executing
STM    #k, AR1      ; this 2-word instruction.
```

例7−63. **ソフトウェア トラップが続く**IPTR**設定**

```
STM    #k, PMST     ; This instruction relocates
NOP                 ; interrupt vectors by writing to
NOP                 ; the IPTR bit field.
TRAP                ; Fetch a TRAP vector from the
                    ; relocated vector table.
```


表7−27**は、**PMST**レジスタの**DROM**ビットにライトを行う命令間のレイテンシと、これに続いて**DROM**アドレス領域にリード/ライトを行う命令との間のレイテンシを示しています。**

注:

レイテンシに合わせるために必要に応じて命令を並べ替えたりNOP**を挿入するのは、プログラマの責任で行ってください。**

表7−27. DROM**ビットのレイテンシ**

(a) 2**番目の命令のカテゴリに基づくレイテンシ**

First instruction is...		And second instruction is Category I, the latency is...
STM	#lk, drom	2
ST	#lk, drom	
MVDK	dmad, drom	
MVMD	MMR, drom	
All other instructions that modify DROM		3

(b) 2**番目の命令のカテゴリ**

Category I
All instructions that read from or write to the DROM address range

記号: drom **直接、間接、またはメモリ マップド アドレッシング モードで、**DROM**ビットを変更するために** PMST**を指すオペランド**

注: 1. **命令が**DROM**ビット フィールドを変更するとき外部メモリ アクセスが進行中なら、追加のレイテンシ サイクルが必要です。**
2. PMST**レジスタの**DROM**ビットを変更する命令で表に示されないものは、レイテンシがありません。**

例7−64. **リード アクセスが続く**DROM**設定**(DP=0)

```
ORM    #8h, PMST        ; This instruction sets DROM = 1.
NOP
NOP
NOP
LD     *AR3, A          ; Reads from on-chip DROM
```

例7−65. **デュアル リード アクセスが続く**DROM**設定**

```
STM    #k, PMST         ; This instruction sets DROM = 1.
NOP
NOP
MPY    *AR3+,*AR4+, A   ; This instruction reads from
                        ; on-chip DROM.
```

7.5.10 アキュムレータへのメモリ マップド アクセスの時のレイテンシ

アキュムレータAおよびBは、メモリ マップ、直接 間接アドレッシング モードのいずれかを使って、メモリ マップド レジスタとしてアクセスできます。通常、アキュムレータにメモリ マップド レジスタとしてアクセスするのはいい方法ではありません。なぜなら、'54x の命令セットはアキュムレータへの直接アクセスをサポートするからです。アキュムレータへの直接アクセスをサポートする命令には、ADD、SUB、AND、OR、XOR、LD、STH、STL、MAC、MPYAなどがあります。

アキュムレータにメモリ マップド レジスタとしてアクセスしない命令を使って2つのアキュムレータにアクセスするとき、パイプラインのレイテンシは発生しません。希に、メモリ マップド レジスタAG、AH、AL、BG、BH、BLによってアキュムレータにアクセスするとき、メモリ マップ、直接 間接アドレッシング モードのいずれかを使ってオペランドにアクセスする命令のどれでも使える場合があります。このような命令には、POPM ALおよびPSHM AHなどがあります。直接アドレッシング モードによってメモリ マップド レジスタにアクセスするために、DPはゼロにする必要があります。

次の2つの条件が同時に満たされる場合、パイプラインのコンフリクトが発生することがあります。

- ある命令がアキュムレータ(AまたはB)を直接変更する。
- 次の命令が、そのアキュムレータをメモリ マップド レジスタとしてリードする。

コンフリクトが発生するのは、1番目の命令がアキュムレータを更新し、それと同時に次の命令がそのアキュムレータをメモリ マップド レジスタとしてリードするからです。

例7−66. 1サイクル レイテンシをともなうアキュムレータ アクセス

```
ADD    Smem, A       ; A is updated directly by this
                     ; instruction.
NOP                  ; This conflict occurs because the
                     ; next instruction tries to read A
                     ; as a memory-mapped register. A
                     ; one-cycle latency required.
PSHM   AL            ; This instruction reads A as a
                     ; memory•mapped register.
```

例7−67. **コンフリクトなしのアキュムレータ アクセス**

(a)

```
ADD    Smem, A      ; A is updated directly by this
                    ; instruction. No conflict occurs
                    ; because the next instruction reads
                    ; accumulator A directly.
NEG    A            ; This instruction reads A directly.
```

(b)

```
STLM   A, BH        ; BH is written using memory-mapped
                    ; addressing here.
                    ; No conflict occurs because the
                    ; next instruction also accesses the
                    ; same accumulator as a memory-
                    ; mapped register.
PSHM   BH           ; Reads BH as a memory•mapped
                    ; register.
```

(c)

```
STLM   A, BH        ; BH is written using memory-mapped
                    ; addressing here.
                    ; No conflict occurs because the
                    ; next instruction accesses the same
                    ; accumulator directly.
NEG    B            ; This instruction reads B directly.
```

表7−28**は、アキュムレータを直接更新する命令と、同じアキュムレータにメモリ マップド レジスタとしてアクセスする命令との間のレイテンシを示しています。**

注:

レイテンシをさけるために必要に応じて命令を並べ替えたりNOP**を挿入するのは、プログラマの責任で行ってください。**

表7−28. アキュムレータAおよびBをメモリ マップド レジスタとして使用する際のレイテンシ

(a) 2番目の命令の各カテゴリに基づくレイテンシ

First Instruction	Second Instruction Category I	Second Instruction Category II
All 1-word instructions that directly modify A or B without accessing them as memory-mapped registers	1	0
ADD Smem, shift†, src, dst LD Smem, shift†, dst SUB Smem, shift†, src, dst	1	0
All 2-word instructions that directly modify A or B without accessing them as memory-mapped registers	0	0

† −16から15の範囲のシフト値。

(b) 2番目の命令のカテゴリ

Category I	Category II
All instructions that read an accumulator as a memory-mapped register (AG, AH, AL, BG, BH, and BL) without using a long-offset modifier.	All instructions that read an accumulator as memory-mapped register (AG, AH, AL, BG, BH, BL) using a long-offset modifier
	MVKD accum, Smem MVDM accum, MMR
	ADD accum, shift, src, dst LD accum, shift, dst SUB accum, shift, src, dst } With extended shift value of −16 to 15

記号: MMR あらゆるメモリ マップド レジスタ
accum メモリ マップド アドレッシング、直接アドレッシング、間接アドレッシングのいずれかのモードを使い、AG、AH、AL、BG、BHまたはBLを指すソース オペランド。

例7−68. 1サイクル レイテンシをともなうアキュムレータ更新

```
LD     Smem, •1, B      ; B is updated directly by this
                        ; instruction
NOP                     ; Conflict occurs because next
                        ; instruction tries to read B as
                        ; a memory-mapped register. A
                        ; one-cycle latency is required.
PSHM   BH               ; Reads BH as a memory-mapped
                        ; register.
```

例7−69. レイテンシなしのアキュムレータ更新

(a)

```
MAC    #K,A          ; A is updated directly by this
                     ; instruction. No latency is
                     ; required since MAC is a 2•word
                     ; instruction.
PSHM   AL            ; This instruction reads A as a
                     ; memory-mapped register.
```

(b)

```
ADD    Smem,A        ; A is updated directly by this
                     ; instruction. No latency is
                     ; required since the next
                     ; instruction uses a long offset
                     ; modifier.
LD     *(AL),ASM     ; This instruction reads A as a
                     ; memory•mapped register.
```

第8章

オンチップ ペリフェラル

’54xのオンチップ ペリフェラルは、各デバイスによって異なります。本章では、第9章シリアル ポート、第10章外部バス動作とともに、あらゆるオンチップ ペリフェラルについて説明します。デバイスによっては、これらペリフェラルのサブセットのみを備えている場合もあります。

’54xのすべてのデバイスには、汎用I/Oピン、タイマ、クロック生成器、ソフトウェア ウェイト ステート発生器、プログラマブル バンク スイッチング ロジックを備えています。各種タイプのシリアル ポート、ホスト ポート インターフェイス、クロック発生器は、デバイスによって異なるペリフェラルです。シリアル ポートについては、第9章で説明します。ソフトウェア ウェイト ステート発生器およびプログラマブル バンク スイッチング ロジックについては、第10章で説明します。

- ❑ **汎用I/Oピン:XFおよび$\overline{\text{BIO}}$**
- ❑ **タイマ**
- ❑ **クロック発生器**
- ❑ **ホスト ポート インターフェイス**(’542、’545、’548、’549)
- ❑ **同期シリアル ポート**(’541、’545、’546)
- ❑ **バッファド シリアル ポート**(’542、’543、’545、’546、’548、’549)
- ❑ **時分割多重**(TDM)**シリアル ポート**(’542、’543、’548、’549)
- ❑ **ソフトウェア ウェイト ステート発生器**
- ❑ **プログラマブル バンク スイッチング ロジック**

8

Topic **Page**

8.1 ペリフェラルのメモリマップド レジスタ

ペリフェラルは、メモリマップド制御レジスタおよびメモリマップド データ レジスタへアクセスすることにより実行、制御します。これらのレジスタは、ペリフェラルとの間でデータを転送することもできます。制御レジスタのビットをセットしたりクリアすることで、ペリフェラルをイネーブル、ディセーブル、初期化およびダイナミックな再設定ができます。シリアルポートやタイマは、割り込みの入力及び割り込みフラグのポーリングによってCPUとインターフェイスします。ペリフェラルを使用しないときは、内部クロックは遮断され、通常実行モードまたはアイドル モードでペリフェラルの消費電力が節約されます。

ペリフェラル レジスタはデータ ページ0にマップされています。図8−1は、各ペリフェラルのメモリマップド レジスタの構成を示しています。表8−1から表8−6は、'54xの各デバイスごとの個別のペリフェラル メモリマップド レジスタを示しています。ペリフェラル メモリマップド レジスタへのあらゆるアクセスには、2サイクルが必要です。

図8−1. TMS320C54xのペリフェラル メモリマップド レジスタ

	’541	’542	’543	’545	’546	’548/’549
0020h	同期シリアル ポート0	バッファド シリアル ポート	バッファド シリアル ポート	バッファド シリアル ポート	バッファド シリアル ポート	バッファド シリアル ポート0
0024h	タイマ	タイマ	タイマ	タイマ	タイマ	タイマ
0028h	外部バス制御	外部バス制御	外部バス制御	外部バス制御	外部バス制御	外部バス制御
002Ch	予約	ホスト ポート インターフェイス	予約	ホスト ポート インターフェイス	予約	ホスト ポート インターフェイス
0030h	同期シリアル ポート1	TDM シリアル ポート	TDM シリアル ポート	同期シリアル ポート	同期シリアル ポート	TDM シリアル ポート
0034h	予約			予約	予約	
0038h	予約	自動バッファリング ユニット	自動バッファリング ユニット	自動バッファリング ユニット	自動バッファリング ユニット	自動バッファリング ユニット(BSP0)
003Ch	予約	予約	予約	予約	予約	自動バッファリング ユニット(BSP1)
0040h	予約	予約	予約	予約	予約	バッファ シリアル ポート1
0044h	予約	予約	予約	予約	予約	予約
0048h	予約	予約	予約	予約	予約	予約
004Ch	予約	予約	予約	予約	予約	予約
0050h	予約	予約	予約	予約	予約	予約
0054h	予約	予約	予約	予約	予約	予約
0058h	予約	予約	予約	予約/クロック モード†	予約/クロック モード†	クロック モード
005Ch	予約	予約	予約	予約	予約	予約

† 一部デバイスのみ(附録を参照)

表8−1. TMS320C541ペリフェラル メモリ マップド レジスタ

アドレス	名称	説明
20	DRR0	シリアル ポート0データ受信レジスタ
21	DXR0	シリアル ポート0データ送信レジスタ
22	SPC0	シリアル ポート0制御レジスタ
23	−	予約
24	TIM	タイマ レジスタ
25	PRD	タイマ周期レジスタ
26	TCR	タイマ制御レジスタ
27	−	予約
28	SWWSR	ソフトウェア ウェイト ステート レジスタ
29	BSCR	バンク スイッチ制御レジスタ
2A−2F	−	予約
30	DRR1	シリアル ポート1データ受信レジスタ
31	DXR1	シリアル ポート1データ送信レジスタ
32	SPC1	シリアル ポート1制御レジスタ
33−5F	−	予約

表8−2. TMS320C542**ペリフェラル メモリマップド レジスタ**

アドレス	名称	説明
20	BDRR0	**バッファド シリアル ポート データ受信レジスタ**
21	BDXR0	**バッファド シリアル ポート データ送信レジスタ**
22	BSPC0	**バッファド シリアル ポート制御レジスタ**
23	BSPCE0	**バッファド シリアル ポート制御拡張レジスタ**
24	TIM	**タイマ レジスタ**
25	PRD	**タイマ周期レジスタ**
26	TCR	**タイマ制御レジスタ**
27	−	**予約**
28	SWWSR	**ソフトウェア ウェイト ステート レジスタ**
29	BSCR	**バンク スイッチ制御レジスタ**
2A-2B	−	**予約**
2C	HPIC	**ホスト ポート インターフェイス制御レジスタ**
2D-2F	−	**予約**
30	TRCV	TDM**シリアル ポート データ受信レジスタ**
31	TDXR	TDM**シリアル ポート データ送信レジスタ**
32	TSPC	TDM**シリアル ポート制御レジスタ**
33	TCSR	TDM**シリアル ポート チャネル選択レジスタ**
34	TRTA	TDM**シリアル ポート送受信アドレス レジスタ**
35	TRAD	TDM**シリアル ポート受信アドレス レジスタ**
36−37	−	**予約**
38	AXR0	ABU**送信アドレス レジスタ**
39	BKX0	ABU**送信バッファド サイズ レジスタ**
3A	ARR0	ABU**受信アドレス レジスタ**
3B	BKR0	ABU**受信バッファド サイズ レジスタ**
3C−5F	−	**予約**

表8−3. TMS320C543ペリフェラル メモリマップド レジスタ

アドレス	名称	説明
20	BDRR0	バッファド シリアル ポート データ受信レジスタ
21	BDXR0	バッファド シリアル ポート データ送信レジスタ
22	BSPC0	バッファド シリアル ポート制御レジスタ
23	BSPCE0	バッファド シリアル ポート制御拡張レジスタ
24	TIM	タイマ レジスタ
25	PRD	タイマ周期レジスタ
26	TCR	タイマ制御レジスタ
27	−	予約
28	SWWSR	ソフトウェア ウェイト ステート レジスタ
29	BSCR	バンク スイッチ制御レジスタ
2A-2F	−	予約
30	TRCV	TDMシリアル ポート データ受信レジスタ
31	TDXR	TDMシリアル ポート データ送信レジスタ
32	TSPC	TDMシリアル ポート制御レジスタ
33	TCSR	TDMシリアル ポート チャネル選択レジスタ
34	TRTA	TDMシリアル ポート送受信レジスタ
35	TRAD	TDMシリアル ポート受信アドレス レジスタ
36-37	−	予約
38	AXR0	ABU送信アドレス レジスタ
39	BKX0	ABU送信バッファド サイズ レジスタ
3A	ARR0	ABU受信アドレス レジスタ
3B	BKR0	ABU受信バッファド サイズ レジスタ
3C-5F	−	予約

表8−4. TMS320C545**ペリフェラル メモリマップド レジスタ**

アドレス	名称	説明
20	BDRR0	**バッファド シリアル ポート データ受信レジスタ**
21	BDXR0	**バッファド シリアル ポート データ送信レジスタ**
22	BSPC0	**バッファド シリアル ポート制御レジスタ**
23	BSPCE0	**バッファド シリアル ポート制御拡張レジスタ**
24	TIM	**タイマ レジスタ**
25	PRD	**タイマ周期ジスタ**
26	TCR	**タイマ制御レジスタ**
27	−	**予約**
28	SWWSR	**ソフトウェア ウェイト ステート レジスタ**
29	BSCR	**バンク スイッチ制御レジスタ**
2A-2B	−	**予約**
2C	HPIC	**ホスト ポート インターフェイス制御レジスタ**
2D-2F	−	**予約**
30	DRR1	**シリアル ポート データ受信レジスタ**
31	DXR1	**シリアル ポート データ送信レジスタ**
32	SPC1	**シリアル ポート制御レジスタ**
33-37	−	**予約**
38	AXR0	ABU**送信アドレス レジスタ**
39	BKX0	ABU**送信バッファド サイズ レジスタ**
3A	ARR0	ABU**受信アドレス レジスタ**
3B	BKR0	ABU**受信バッファド サイズ レジスタ**
3C-57	−	**予約**
58	CLKMD†	**クロック モード レジスタ**
59-5F	−	**予約**

† **一部デバイスのみ(附録を参照)**

表8−5. TMS320C546ペリフェラル メモリマップド レジスタ

アドレス	名称	説明
20	BDRR0	バッファド シリアル ポート データ受信レジスタ
21	BDXR0	バッファド シリアル ポート データ送信レジスタ
22	BSPC0	バッファド シリアル ポート制御レジスタ
23	BSPCE0	バッファド シリアル ポート制御拡張レジスタ
24	TIM	タイマ レジスタ
25	PRD	タイマ周期レジスタ
26	TCR	タイマ制御レジスタ
27	−	予約
28	SWWSR	ソフトウェア ウェイト ステート レジスタ
29	BSCR	バンク スイッチ制御レジスタ
2A-2F	−	予約
30	DRR1	シリアル ポート データ受信レジスタ
31	DXR1	シリアル ポート データ送信レジスタ
32	SPC1	シリアル ポート制御レジスタ
33-37	−	予約
38	AXR0	ABU送信アドレス レジスタ
39	BKX0	ABU送信バッファド サイズ レジスタ
3A	ARR0	ABU受信アドレス レジスタ
3B	BKR0	ABU受信バッファド サイズ レジスタ
3C-57	−	予約
58	CLKMD†	クロック モード レジスタ
59-5F	−	予約

† 一部デバイスのみ(附録を参照)

表8−6. TMS320C548/TMS320C549**ペリフェラル メモリマップド レジスタ**

アドレス	名称	説明
20	BDRR0	**バッファド シリアル ポート**0**データ受信レジスタ**
21	BDXR0	**バッファド シリアル ポート**0**データ送信レジスタ**
22	BSPC0	**バッファド シリアル ポート**0**制御レジスタ**
23	BSPCE0	**バッファド シリアル ポート**0**制御拡張レジスタ**
24	TIM	**タイマ レジスタ**
25	PRD	**タイマ周期レジスタ**
26	TCR	**タイマ制御レジスタ**
27	−	**予約**
28	SWWSR	**ソフトウェア ウェイト ステート レジスタ**
29	BSCR	**バンク スイッチ制御レジスタ**
2A-2B	−	**予約**
2C	HPIC	**ホスト ポート インターフェイス制御レジスタ**
2D-2F	−	**予約**
30	TRCV	TDM**シリアル ポート データ受信レジスタ**
31	TDXR	TDM**シリアル ポート データ送信レジスタ**
32	TSPC	TDM**シリアル ポート制御レジスタ**
33	TCSR	TDM**シリアル ポート チャネル選択レジスタ**
34	TRTA	TDM**シリアル ポート送受信レジスタ**
35	TRAD	TDM**シリアル ポート受信アドレス レジスタ**
36-37	−	**予約**
38	AXR0	ABU0**送信アドレス レジスタ**
39	BKX0	ABU0**送信バッファド サイズ レジスタ**
3A	ARR0	ABU0**受信アドレス レジスタ**
3B	BKR0	ABU0**受信バッファド サイズ レジスタ**
3C	AXR1	ABU1**送信アドレス レジスタ**
3D	BKX1	ABU1**送信バッファド サイズ レジスタ**
3E	ARR1	ABU1**受信アドレス レジスタ**
3F	BKR1	ABU1**受信バッファド サイズ レジスタ**
40	BDRR1	**バッファド シリアル ポート**1**データ受信レジスタ**
41	BDXR1	**バッファド シリアル ポート**1**データ送信レジスタ**
42	BSPC1	**バッファド シリアル ポート**1**制御レジスタ**
43	BSPCE1	**バッファド シリアル ポート**1**制御拡張レジスタ**
44-57	−	**予約**
58	CLKMD	**クロック モード レジスタ**
59-5F	−	**予約**

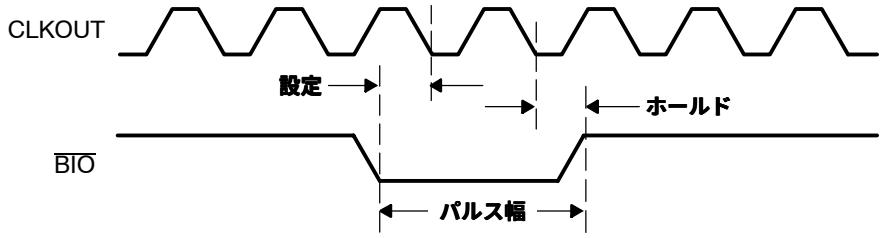
8.2 汎用I/O

’54xには、I/Oメモリ空間と、ソフトウェアで制御される2つの専用ピンからなる汎用I/Oがあります。これらの専用ピンは、分岐制御入力ピン($\overline{\text{BIO}}$)および外部フラグ出力ピン(XF)です。I/Oメモリ空間および外部バスを使用するアクセスの詳細については、3−22ページのセクション3.4I/Oメモリおよび第10章外部バス機能を参照してください。

8.2.1 分岐制御入力ピン($\overline{\text{BIO}}$)

$\overline{\text{BIO}}$を使って、各ペリフェラルの状態をモニタできます。特に、時間的にクリティカルなループ処理に割り込みで中断することができないような時に有用です。$\overline{\text{BIO}}$入力の状態に応じて、分岐を条件付きで実行できます。図8−2に示すタイミング図は、$\overline{\text{BIO}}$の実行を表しています(タイミング仕様については54xのデータ シートを参照してください)。条件付実行命令(XC)は、パイプラインのデコードステージで$\overline{\text{BIO}}$の状態をサンプリングします。他の条件付き命令(分岐、コール、リターン)は、パイプラインのリード ステージで$\overline{\text{BIO}}$をサンプリングします。

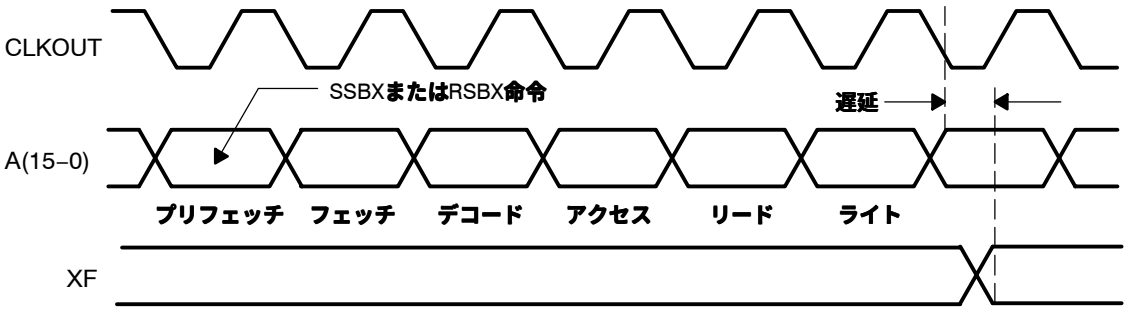
図8−2. BIOタイミング図



8.2.2 外部フラグ出力ピン(XF)

XFを使って、外部デバイスに信号を送ることができます。XFピンはソフトウェアを使って制御します。XFビット(ST1[13:13])をセットすることによりハイになり、XFビットをクリアするとローになります。セット ステータス レジスタ ビット命令(SSBX)およびリセット ステータス レジスタ ビット命令(RSBX)は、それぞれXFをセットおよびクリアします。XFはデバイスのリセット時にhighにセットされます。図8−3は、SSBXまたはRSBX命令がフェッチされるタイミングと、XFピンがセットまたはリセットされるタイミングを表しています(タイミング仕様については'54xのデータ シートを参照してください)。ここに示されるXFタイミングは、1サイクル命令のシーケンスに対応します。実際のタイミングは命令シーケンスのタイプに応じて変わります。

図8−3. 外部フラグ タイミング図



8.3 タイマ

オンチップ タイマはプログラマブルなタイマで、3つのレジスタから構成されており、これらのレジスタを使って周期的に割り込みを発生させることができます。タイマはデバイスのCLKOUTを基本レートとし、最大20ビットのレンジで周期を変更することができます。

8.3.1 タイマ レジスタ

オンチップ タイマは、3つのメモリマップド レジスタ(TIM、PRD、TCR)から構成されます。表8−7は、これらのレジスタを示しています。

表8−7. タイマ レジスタ

アドレス	レジスタ	解説
0024h	TIM	タイマ レジスタ
0025h	PRD	タイマ周期レジスタ
0026h	TCR	タイマ制御レジスタ

- タイマ レジスタ(TIM)。16ビットのメモリマップド タイマ レジスタ(TIM)は周期レジスタ(PRD)の値がロードされ、デクリメントされます。
- タイマ周期レジスタ(PRD)。16ビットのメモリマップド タイマ周期レジスタ(RPD)を使ってタイマ レジスタ(TIM)を再ロードできます。
- タイマ制御レジスタ(TCR)。16ビットのメモリマップド タイマ制御レジスタ(TCR)には、タイマの制御ビットおよびステータス ビットがあります。図8−4にはTCRビット フィールドを表し、表8−8にはその説明があります。

図8−4. タイマ制御レジスタ(TCR)図

15−12	11	10	9−6	5	4	3−0
予約	Soft	Free	PSC	TRB	TSS	TDDR

表8−8. タイマ制御レジスタ(TCR)ビット概要

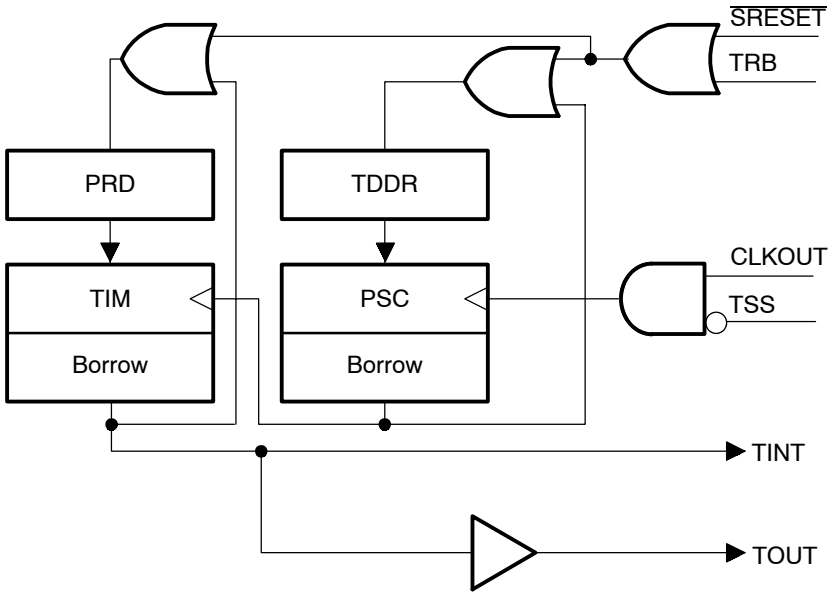
ビット	名称	リセット値	説明
15−12	予約	—	予約;常に0としてリードします。
11	Soft	0	Freeビットと組み合わせて使用することにより、HLLデバッガ使用時にブレークポイントに達したときのタイマの状態を決めます。Freeビットが0のとき、Softビットでタイマ モードを選択できます。 Soft = 0 タイマは直ちに停止します。 Soft = 1 タイマはカウンタがデクリメントされて0になると停止します。
10	Free	0	Softビットと組み合わせて使用することにより、HLLデバッガ使用時にブレークポイントに達したときのタイマの状態を決めます。Freeビットが0のとき、Softビットでタイマ モードを選択できます。 Free = 0 Softビットがタイマ モードを選択します。 Free = 1 Softビットにかかわりなく、タイマはフリー ランします。
9−6	PSC	—	プリスケーラ カウンタ。オンチップ タイマのカウントを指定します。PSCがデクリメントされて0になるかタイマをリセットしたとき、TDDRの値をPSCにロードしてTIMをデクリメントします。
5	TRB	—	タイマ再ロード。オンチップ タイマをリセットします。TRBをセットすると、RPDの値がTIMにロードされ、TDDRの値がPSCにロードされます。TRBは常に0としてリードします。
4	TSS	0	タイマ停止ステータス。オンチップ タイマを停止または起動させます。リセット時は、TSSがクリアされタイマが直ちにタイミングを開始します。 TSS = 0 タイマを起動 TSS = 1 タイマを停止
3−0	TDDR	0000	タイマ分周レジスタ。オンチップ タイマのタイマ分周率を指定します。PSCがデクリメントされて0になると、PSCにTDDRの値がロードされます。

8.3.2 タイマ動作

タイマはオンチップのダウン カウンタで、CPU割り込みを周期的に発生させるのに使用します。タイマは、CLKOUT1サイクルごとに1デクリメントします。カウンタがデクリメントされて0になるたびにタイマ割り込み(TINT)が発生して、ダウン カウンタに周期を再ロードします。割り込みの詳細については、ページ6−26、セクション6.10割り込みを参照してください。

図8−5は、タイマの論理ブロック図を示しています。これは、PRDおよびTIMからなるメイン タイマ ブロック、およびTCR内のTDDRおよびPSCビットからなるプリスケーラ ブロックの2つの基本ブロックにより構成されます。タイマは、デバイスのCLKOUTによりカウントされます。

図8−5. タイマ ブロック図



メインのタイマ ブロックは、ダウン カウンタ レジスタ(TIM)および周期レジスタ(PRD)の2つのレジスタから構成されます。通常の動作では、TIMが0までデクリメントされるときPRDの値をTIMにロードします。デバイスのリセット時(図8−5のSRESET入力)またはタイマ単独リセットのとき(図8−5のTRB入力)、PRDの値をTIMにロードします。TIMはプリスケーラ ブロックによってクロックされます。プリスケーラ ブロックからの出力、1クロックでTIMを1デクリメントします。メイン タイマ ブロックは、タイマ割り込み(TINT)信号をCPUにタイマ出力(TOUT)ピンを外部に出力します。。TOUTパルス幅は、CLKOUTの1周期に等しくなります。

プリスケーラ ブロックは、TIMおよびPRDと同じような動作をします。プリスケーラ カウンタ(PSC)およびタイマ分周レジスタ(TDDR)で構成されます。PSCおよびTDDRの双方とも、タイマ制御レジスタ(TCR)に配置されています。通常の動作では、PSCの値が0になるとTDDRの値をPSCにロードします。デバイスのリセット時、またはタイマ単独リセットのときも、TDDRの値がPSCにロードされます。PSCはデバイスのCLKOUTによってカウントされます。1CLKOUTでPSCは1デクリメントされます。PSCはTCRをリードすることで読み取れますが、PSCに直接ライトすることはできません。

タイマは、TSS入力を使ってタイマへのクロック入力を遮断することで停止できます。タイマが不要なときは、タイマ動作を停止することによりデバイスは低消費電力で実行できます。

タイマ割り込み(TINT)レートは、CLKOUT周波数を2つの個別の係数で割ることで求められます。

$$\text{TINTレート} = \frac{1}{t_{c(C)} \times u \times v} = \frac{1}{t_{c(C)} \times (TDDR + 1) \times (PRD + 1)}$$

この式において、tc(C)はCLKOUTの周期、uはTDDR値+1、vはPRD値+1です。

タイマの現在の値はTIMをリードすることで読み取れます。PSCはTCRをリードすることで読み取れます。2つのレジスタをリードするのに2つの命令が必要です。カウンタが動作している時にこれら2つのリード命令を実行するとそれらの値が異なる場合があります。したがって、正確なタイミング測定が必要なとき、これら2つの値をリードする前にタイマを停止する必要があります。タイマは、TSSビットをセットすることで停止でき、TSSビットをクリアすれば再起動します。

タイマを使って、アナログ インターフェイスなどのペリフェラル回路のサンプル クロックを発生できます。これは、TOUT信号を使ってデバイスにクロックを入力するか、割り込みを利用してレジスタを周期的に読み取ることによって実行できます。

タイマは次の手順で初期化されます。

1) TCRのTSSに1をライトしてタイマを停止します。

2) PRDをロードします。

3) TCRのTDDRを初期化することによりタイマを始動して、TSSを0にリセット、TRBを1にセットしてタイマ周期を再ロードすることによりタイマへのCLKOUTをイネーブルにします。

オプションで、タイマ割り込みを次のようにイネーブルにできます。

1) IFRのTINTに1をライトすることにより、どの待ち状態ののタイマ割り込みもクリアします。

2) IMRのTINTに1をライトすることにより、タイマ割り込みをイネーブルにします。

3) **必要に応じて、**INTM**を**0**にすることにより全てのマスカブル割り込みをイネーブルにします。**

リセット時に、TIM**および**PRD**が**FFFFh**の最大値に設定されます。**TCR**の**TDDR**が**0**にクリアされ、タイマが始動します。**

8.4 クロック発生器

クロック発生器により、システム設計者はクロック ソースを選択できます。クロック発生器を駆動するクロック ソースには次の２つが選択できます。

- ❑ 内部発振回路。水晶発振子と共振回路を'54xのX1およびX2/CLKINピンに接続することにより発振回路を構成し内部にクロックを供給します。
- ❑ 外部クロック。外部クロック ソースは直接X2/CLKINピンに接続され、X1は未接続とします。

'54xデバイスのクロック発生器は、内部発振回路およびフェーズ ロック ループ(PLL)回路から構成されます。現行では、'54xデバイスにはデバイスのタイプによって2種類の異なるPLL回路があります。これらは、ハードウェア コンフィギュラブルなPLL回路、ソフトウェア プログラマブルなPLL回路です(各デバイスがどのPLLを内蔵しているかについては附録Dを参照してください)。C−5ページのセクションC.2部品注文情報を参照してください。

8.4.1 ハードウェア コンフィギュラブルなPLL(附録Dを参照)

PLLは、CPUのマシン サイクル速度より低い外部周波数ソースによって機能します。この機能は、高速スイッチング クロックからの高周波ノイズを低減します。内部の発振器または外部のクロック ソースが、PLLに送られます。外部クロック ソースまたは内部発振器の周波数を逓数Nでかけあわせることにより、内部CPUクロックが生成されます。内部発振回路を使用している場合は、クロック ソースを2分周し内部CPUクロックを生成します。外部クロックを使用している場合は、そのN逓倍が内部CPUクロックとなります。

40MHzC54xデバイスのPLLでは最大動作周波数は40MHzです。PLLには、50μsのロックアップ時間が必要です。ロックアップ時間は、リセット時およびIDLE3パワーダウン モードからの復帰時に必要です。詳しくは、6−44ページ、セクション6.11パワー ダウン モードおよび10−24ページ、サブセクション10.5.2IDLE3ウェイク アップ シーケンスを参照してください。

クロック モードは、CLKMD1、CLKMD2、CLKMD3ピンによって決まります。表8−9は、これらのピンでクロック モードをどのように選択するかを示しています。PLLを使用しない場合は、CPUのクロック周波数は水晶の発振周波数の半分か外部クロック周波数の半分になります。

動作中に、クロック モードはCLKMD1〜3ピンによって再設定できません。IDLE3モードの際は、CLKOUTがハイ レベルにセットされた後にクロック モードを再設定できます。

表8−9. クロック モード

Mode Select Pins			Clock Mode†	
CLKMD1	CLKMD2	CLKMD3	Option 1	Option 2
0	0	0	PLL ¢ 3 with external source	PLL ¢ 5 with external source
1	1	0	PLL ¢ 2 with external source	PLL ¢ 4 with external source
1	0	0	PLL ¢ 3 with internal source	PLL ¢ 5 with internal source
0	1	0	PLL ¢ 1.5 with external source	PLL ¢ 4.5 with external source
0	0	1	Divide-by-2 with external source	Divide-by-2 with external source
1	1	1	Divide-by-2 with internal source	Divide-by-2 with internal source
1	0	1	PLL ¢ 1 with external source	PLL ¢ 1 with external source
0	1	1	Stop mode‡	Stop mode‡

† 各デバイスごとに、オプション1またはオプション2クロック モードに設定されています。

‡ PLLはディセーブルされます。システム クロックは、CPUやペリフェラルには供給されません。stop modeの機能は、IDLE3によるパワー ダウン モードと同じです。ただし、IDLE3命令はクロックを同期的に停止し、割り込みによる復帰が可能なので、徹底した省電力を実現するための用途には、stop modeではなくIDLE3の使用が推奨されます。

8

8.4.2 ソフトウェア プログラマブルなPLL(対象デバイスは附録Dを参照)

ソフトウェア プログラマブルなPLLは高いレベルの柔軟性を備えており、様々なクロック逓倍率を実現するクロック スケーラ、PLLを直接イネーブル/ディセーブルする機能、PLLクロック モードへの切り替えをロックアップ完了まで遅延させるためのPLLロック タイマを含んでいます。

ソフトウェア プログラマブルなPLLを備えたデバイスは、次のいずれかのクロック モードに設定できます。

- ❑ PLLモード。入力クロック(CLKIN)は0.25から15の31通りの逓倍率で逓倍されます。これらの逓倍率は、PLL回路によって得られます。
- ❑ DIV(分周)モード。入力クロック(CLKIN)を2または4分周します。DIVモードを使用している場合は、消費電力を最小限にするため、PLL回路を含むすべてのアナログ部分がディセーブルされます。

クロック モードは、リセット直後にCLKMD1、CLKMD2、CLKMD3の3つの外部ピンの値によって決まります。表8−10は、CLKMDピンに対応するモードを示しています。

表8−10. リセット時のクロック モード設定

CLKMD1	CLKMD2	CLKMD3	CLKMD Reset Value	Clock Mode
0	0	0	0000h	Divide-by-2 with external source
0	0	1	1000h	Divide-by-2 with external source
0	1	0	2000h	Divide-by-2 with external source
1	0	0	4000h	Divide-by-2 with internal source
1	1	0	6000h	Divide-by-2 with external source
1	1	1	7000h	Divide-by-2 with internal source
1	0	1	0007h	PLL ¢ 1 with external source
0	1	1	—	Stop mode

リセットの後、ソフトウェア プログラマブルなPLLを任意の設定にプログラムできます。リセット動作中にCLKMD1〜3を101として、PLL×1 with external sourceオプションを選択した場合、内蔵のPLLロック カウント タイマはインアクティブになります。したがって、PLLのロックアップ時間を計算に入れて、システム側でリセット解除を遅らせる必要があります。

PLLの設定値は、メモリ マップドされた(アドレス58h)クロック モード レジスタ(CLKMD)にロードされます。CLKMDを使ってPLLクロック モジュールの動作を設定します。図8−6はCLKMDビット フィールドを示し、表8−11はその説明です。リセット時、CLKMDはCLKMD1〜3ピンの状態だけで決まる値に初期化されます(表8−10参照)。

図8−6. クロック モード レジスタ(CLKMD)図

15–12	11	10–3	2	1	0
PLLMUL	PLLDIV	PLLCOUNT	PLLON/OFF	PLLNDIV	PLLSTATUS
R/W†	R/W†	R/W†	R/W†	R/W	R

† DIVモード(PLLSTATUSがロー)のとき、PLLMUL、PLLDIV、PLLCOUNT、PLLON/OFFは動作に無関係で、内容は不定となります。

表8−11. クロック モード レジスタ(CLKMD)ビット概要

ビット	名称	ファンクション
15−12	PLLMUL	PLLDIVおよびPLLNDIVとともに周波数逓倍率を設定します(8−21ページ、表8−12参照)。
11	PLLDIV	PLLMULおよびPLLNDIVとともに周波数逓倍率を設定します(8−21ページ、表8−12参照)。
10−3	PLLCOUNT	PLLカウンタの値。PLLの動作開始後、プロセッサへのPLLによるクロック供給を開始するまでの間に、PLLロック タイマにカウントさせる入力クロック サイクル数を(16サイクル単位で)指定します。PLLカウンタは減算カウンタになっており、入力クロックを16分周した信号で駆動されます。したがって、入力クロック16回ごとに、PLLカウンタが1減算されます。PLLCOUNTの詳細については、サブセクションPLLCOUNTプログラマブル ロック タイマの使用法を参照してください。
2	PLLON/OFF	PLLのオン/オフ。PLLNDIVとの組み合わせで、クロック発生器のPLL部をイネーブル/ディセーブルします。PLLON/OFFとPLLNDIVのどちらも、PLLを強制的に動作させる効果があります。すなわち、PLLON/OFFがハイのときは、PLLNDIVの状態と無関係にPLLが動作します。
1	PLLNDIV	クロック発生器がPLLモードで動作するかDIVモードで動作するかを決めます。したがって、PLLMULおよびPLLNDIVとともに周波数逓倍数を設定します。 PLLNDIV = 0 DIVモードを使用 PLLNDIV = 1 PLLモードを使用
0	PLLSTATUS	クロック発生器が動作するモードを示します。 PLLSTATUS = 0 DIVモード PLLSTATUS = 1 PLLモード

PLLON/OFF	PLLNDIV	VCO State
0	0	off
1	0	on
0	1	on
1	1	on

表8−12. PLLNDIV、PLLDIV、PLLMULによるPLL逓倍率

PLLNDIV	PLLDIV	PLLMUL	逓倍数†
0	x	0 − 14	0.5
0	x	15	0.25
1	0	0 − 14	PLLMUL + 1
1	0	15	1
1	1	0 or even	(PLLMUL + 1)/2
1	1	odd	PLLMUL/4

† CLKOUT=CLKIN × 逓倍数

ソフトウエア プログラマブルなPLL使用時のプログラミング注意事項

ソフトウエア プログラマブルなPLLは、スタートアップ時の設定、動作モード、消費電力節約機能により、多くの異なるオプションを提供します。ここでは、プログラミング時の注意事項やいくつかのソフトウエア例を提示することにより、スタートアップ時、異なるクロック モード間での切換え時、およびIDLE1/IDLE2/IDEL3命令実行の前後でのソフトウエア プログラマブルPLLの適切な使用法を示します。

8

PLLCOUNTプログラマブル ロック タイマの使用

ロックアップ期間中は、PLLは'54xのクロックに使用できません。PLLCOUNTプログラマブル ロック タイマによりPLLがロックされるまで、PLLからデバイスへのクロック供給を自動的に遅延する機能が提供されています。

PLLロック タイマはカウンタで、CLKMDレジスタのPLLCOUNTフィールドからロードされるプリセット値から0へデクリメントします。タイマは0から255の任意の値をプリセット可能で、入力クロックは16分周されたCLKINです。従って、結果生じるロックアップ時間は、0から255 × 16CLKINサイクルで設定可能です。

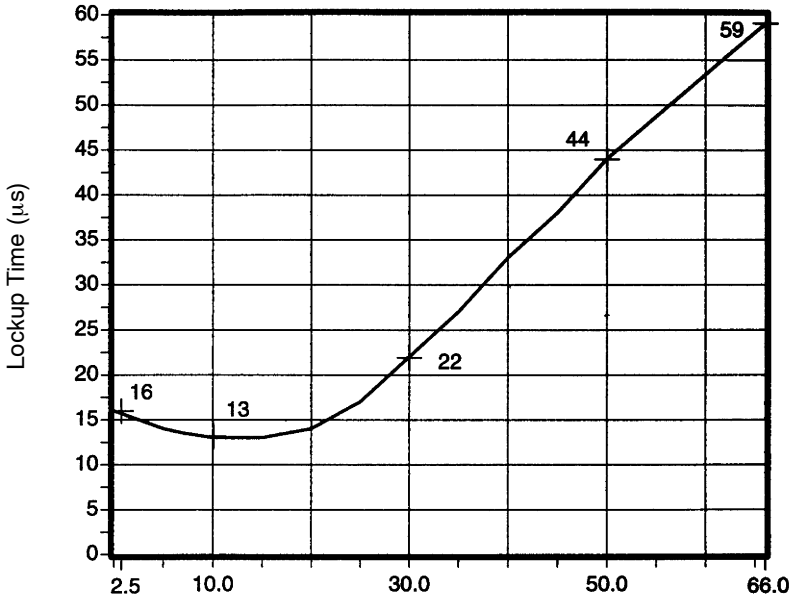
クロック発生器の動作モードがDIVからPLLに切り換わると(8−22ページ、サブセクション、DIVモードからPLLモードへの切換えを参照)、ロック タイマがアクティブになります。ロックアップ期間中、クロック発生器はDIVモードで動作を継続します。PLLロック タイマが0までデクリメントした後、PLLからのクロックが'54xに供給されます。

10進のプリセット値、PLLCOUNTは以下の通りです。

$$PLLCOUNT > \frac{LockupTime}{16 \times T_{CLKIN}}$$

ここで、T_{CLKIN}は入力クロック周期、ロックアップ時間は図8−7に示されたPLLロックアップに要求される時間です。

図8−7. PLL**ロック アップ タイム対**CLKOUT**周波数**



DIVモードからPLLモードへのクロック モード切換え

様々な状況下では、DIV**モードから**PLL**モードへの切換えが要求されます。ただし**DIV**モードから**PLL**モードへの切換え時、**PLL**がロックされていない場合は、確実に適切なクロック信号のみをデバイスへ送るため、モード切換えが発生する前に**PLL**ロックアップ時間遅延に注意する必要があります。従って、動作モードの切換え時に**PLL**がロックされているかどうかを知ることが大切です。**

PLLMUL**や**PLLDIV**値の変更後、**PLL**のターン オフ後**(PLLON/OFF=0)**、入力クロックの損失後、または電源投入時、**PLL**はロックされていません。**PLL**が前もってロックされ、ターンオフされない限り**PLL**はロックされたままです。そして、**PLL**がロックされてから**PLLMUL**および**PLLDIV**の値は変わりません。**

DIV**モードから**PLL**モードへ切換えると**(PLLDIV**を**1**に設定**)**、**PLLCOUNT**プログラマブル ロック タイマをアクティブにし**(PLLCOUNT**がゼロ以外の値で前もってロードされている時**)**、ロックアップ時間遅延の実現の機能を提供します。ロックアップ遅延の実現のため**PLL**を使用せずに、リセット遅延を使用しない限り、**PLLCOUNT**ロック タイマ機能は、前に述べた**PLL**がアンロックされた状況下で使用されます。**

DIVモードからPLLモードへの切換えはCLKMDへのロードで実現します。以下に、PLLがロックされていない時のDIVモードからPLLモードへの切換えの手順を記述します。PLLが既にロックされた状態でこのモード切換えを実行すると、PLLモードからDIVモードへの切換え時と同様です。ただし順序は逆です。この場合、新しいクロック モードが実行される時の遅延は同じです。

PLLがロックされていない状態でDIVモードからPLLモードへ切り換える時、またはロックされていない動作でモード変更が生じる時、PLLMUL、PLLDIV、PLLNDIVビットに、8−21ページ、表8−12に示された希望の周波数逓倍数を選択するよう設定し、PLLCOUNTビットには、要求されたロックアップ時間を選択するよう設定します。PLLMUL、PLLDIV、PLLCOUNT、PLLON/OFFは、DIVモードでは変更のみ可能、という点に注意してください。

PLLNDIVビットがいったん設定されると、PLLCOUNTタイマはプリセット値からデクリメントを開始します。PLLCOUNTタイマが0に達すると、6CLKINサイクル+3.5PLLサイクル(CLKOUT周波数)の後にPLLモードへの切換えが、実行されます。PLLモードへの切換えが完了すると、CLKMDのPLLSTATUSビットは、1となります。PLLロックアップ期間中は、'54xはDIVモードで動作を継続することに注意してください。

DIVモードからPLL × 3モードへの切換え時、CLKIN周波数13MHz、PLLCOUNT=41(10進)で、以下のコードが使用できます。

```
STM   #0010000101001111b, CLKMD
```

PLLモードからDIVモードへのクロック モード切換え

PLLモードからDIVモードへの切換え時、PLLCOUNTによる遅延は発生せず、2つのモード間の切換えは短い遅延の後、行なわれます。

PLLモードからDIVモードへの切換えも、同様にCLKMDへのロードにて実行されます。PLLNDIVビットを0にクリアして、DIVモードを選択し、PLLMULビットは、8−21ページ、表8−12に示された希望の周波数逓倍数を選択するよう設定します。

DIVモードへの切換えは、1111bを除くすべてのPLLMUL値に対して、6CLKINサイクル+3.5PLLサイクル(CLKOUT周波数)で実行されます。1111bのPLLMUL値に対しては、DIVモードへの切換えは12CLKINサイクル+3.5PLLサイクル(CLKOUT周波数)で実行されます。DIVモードへの切換えが完了すると、CLKMDのPLLSTATUSビットは0となります。

例8−1に、PLL × 3モードから2分周モードに切換えるためのコード シーケンスを示します。PLLSTATUSビットで、DIVモードへの切換えをポーリングしており、次にこのポイントでPLLをターン オフするため、STM命令が使用されています。

例8−1. PLL × 3**モードから**2**分周モードへのクロック モード切換え**

```
          STM    #0b, CLKMD    ;switch to DIV mode
TstStatu: LDM    CLKMD, A
          AND    #01b, A       ;poll STATUS bit
          BC     TstStatu, ANEQ
          STM    #0b, CLKMD    ;reset PLLON/OFF when STATUS
                               ;is DIV mode
```

PLL逓倍率の変更

あるPLL**逓倍率から別の逓倍率への切換えが要求されると、クロック発生器は、新しい逓倍率を選択する前に、まず**PLL**モードから**DIV**モードへ切り換える必要があります。ある**PLL**逓倍率から別の逓倍率への直接切換えはできません。**

あるPLL**逓倍率から別の逓倍率へ切り換えるには、以下の手順に従って下さい。**

1) PLLNDIV**ビットを**0**にクリアし、**DIV**モードを選択します。**

2) DIV**モードがイネーブルである**(PLLSTATUS=0)**と示されるまで、**PLLSTATUS**ビットをポーリングします。**

3) 8−21**ページの表**8−12**に示されるように、**PLLMUL**、**PLLDIV**、**PLLNDIV**ビットを希望の周波数逓倍数に設定するよう**CLKMD**を変更します。**

4) PLLCOUNT**ビットを要求されたロックアップ時間に設定します。**

いったんPLLNDIV**ビットが設定されると、**PLLCOUNT**タイマは、プリセット値からデクリメントを開始します。**PLLCOUNT**タイマが**0**に達すると、**6CLKIN**サイクル**+3.5PLL**サイクル**(CLKOUT**周波数**)の後、新しい**PLL**モードが実行されます。**

2分周モードと4分周モード間の直接切換えは不可能であることに注意してください。これら2つのモード間で切り換えするには、クロック発生器を最初に整数のみ(非分数)の逓倍率をPLLモードへ設定し、次に希望の分周を**DIV**モードへ設定し戻す必要があります**(8−22**ページのサブセクション、**DIV**モードから**PLL**モードへの切換えを参照**)。

例8−2**は、**PLL × X**モードから**PLL × 1**モードへのクロック モード切換えに使用可能なコード シーケンスを示します。**

例8−2. PLL × X**モードから**PLL × 1**モードへのクロック モード切換え**

```
          STM    #0b, CLKMD              ;switch to DIV mode
TstStatu: LDM    CLKMD, A
          AND    #01b, A                 ;poll STATUS bit
          BC     TstStatu, ANEQ
          STM#0000001111101111b, CLKMD  ;switch to PLL × 1 mode
```

リセット直後のPLL動作

リセット直後は、クロック モードは、8−19**ページ、表**8−10**に示された**CLKMD1、CLKMD2、CLKMD3**の3つの外部ピンの値にて決定されます。これら動作モードの**2**つを除くすべては、外部ソースの**2**分周です。**2**分周モードから**PLL**モードへの切換えは、**CLKMD**の内容を変更することにより、簡単に実行できます。内部発振器を使用したい場合、内部発振器はソフトウエアのプログラムが不可能なため**CLKMD(1−3)=100**または**CLKMD(1−3)=111**のいずれかをリセット時に(表**8−10**に示された)選択する必要があります。**

以下のコードは、2分周モードからPLL × 3モードへの切換えに使用できます。

```
STM   #0010000101001111b, CLKMD
```

IDLE命令使用時のPLL注意事項

電力消費を低減するためIDLE命令の1つを使用する時は、PLLを適切に管理することが大切です。PLLがディセーブルでDIVモード動作時に、クロック生成器は消費電力が最小となります。従って、消費電力が重要となる場合は、IDEL1、IDLE2、IDLE3命令を実行する前に、PLLモードからDIVモードへ切り換えることが望ましいです。これは、8−23ページ、サブセクション、PLLモードからDIVモードへの切換えで説明されているように実行します。IDEL1/IDEL2/IDLE3からウェイク アップした後、クロック発生器は、8−22ページ、サブセクション、DIVモードからPLLモードへの切換えに説明があるように、PLLモードに再プログラミングできます。

PLLはIDLE状態中停止し、'54xが再スタートし、クロック発生器がPLLモードへ切り換え戻されると、通常の起動時と同様にPLLロックアップ時間が生じることに注意してください。従ってこの場合、ロックアップ時間は、外部的に、あるいはPLLロックアップ タイマを使うことによって対処する必要があります。

例8−3は、PLL × 3モードから2分周モードへクロック発生器を切り換え、PLLをターン オフし、IDLE3へ入るコード シーケンスを示します。IDLE3からのウェイク アップ後、クロック発生器は、ロック タイマ値に対し64(10進)のPLLCOUNTで、単一STM命令を使用して、DIVモードからPLL × 3モードへ切り換えられます。

例8−3. PLL × 3モードから2分周モードへのクロック切り替え、PLLのターン オフ、およびIDLE3への入力

```
          STM    #0b, CLKMD                ;switch to DIV mode
TstStatu: LDM    CLKMD, A
          AND    #01b, A                   ;poll STATUS bit
          BC     TstStatu, ANEQ
          STM    #0b, CLKMD                ;reset PLLON_OFF when STATUS
                                           ;is DIV mode
          IDLE3

   (After IDLE3 wake•up • switch the PLL from DIV mode to PLL × 3 mode)

          STM    #0010001000000111b, CLKMD  ;PLLCOUNT = 64 (decimal)
```

ブートローダ使用時のPLL注意事項

'545Aおよび'546AのROMは、リセット解除後、RAMへプログラムをロード可能なブート ローダ プログラムを持ちます。ソフトウエア プログラマブルなPLLでこのブート ローダを使用する際には、システムを適切に動作させるため、いくつかの重要な注意事項があります。

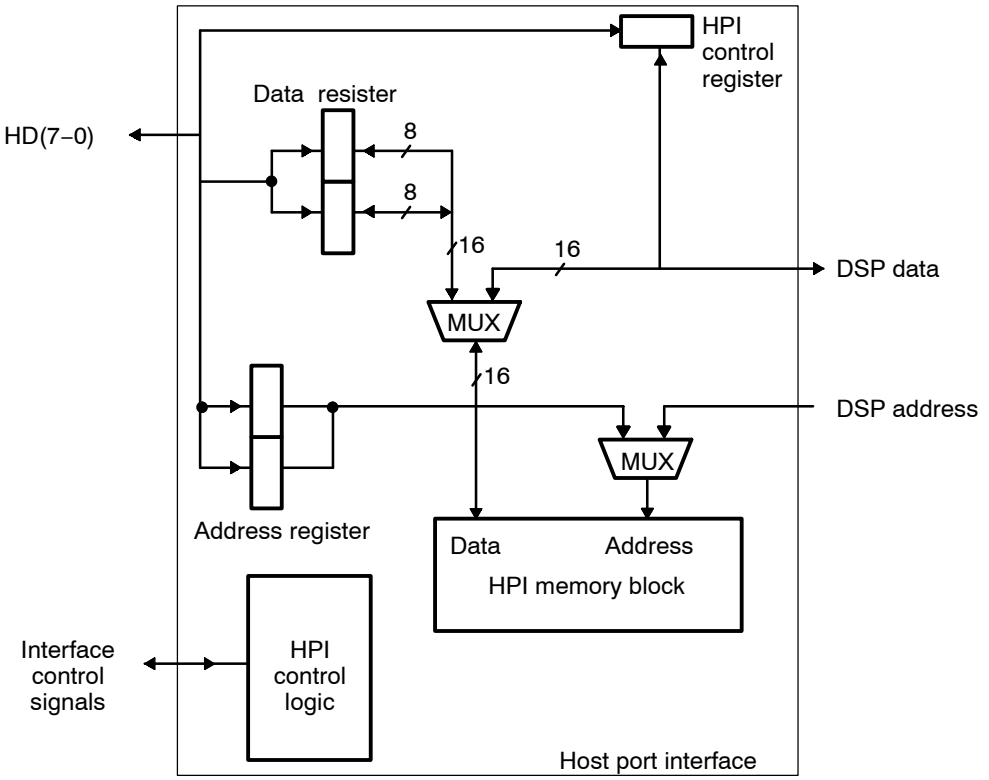
'545Aおよび'546Aでは、互換性を保つため、ブートローダは、クロック モードを、CLKMD(1−3)入力ビットがオプション−2のハードウェア プログラマブルPLL(8−18ページ、表8−9参照)へ与えられた場合と同じモードで、PLLを設定します。いったんブートローダ プログラムが実行を終了し、制御がユーザーのプログラムへ転送されると、PLLは任意の希望する設定に再プログラミングできます。

8.5 ホスト ポート インターフェイス

’54Xシリーズにはホスト ポート インターフェイス(HPI)を備えたものがあります。HPIは8ビットのパラレル ポートで、ホスト デバイスまたはホスト プロセッサを’54xに接続する役割を持ちます。データは’54xとホスト デバイスの間で’54xのオンチップ メモリを通してやり取りされ、このメモリはHPI RAMと呼ばれ、ホストおよび’54xの双方からアクセス可能です。

HPIはホスト デバイスとのインターフェイスを行い、ホスト デバイスがインターフェイスのマスターとなり、ホストによりアクセスを容易にします。ホスト デバイスは、’54xが直接アクセスしない専用アドレスおよびデータ レジスタ、HPI制御レジスタを介して、外部データおよびインターフェイス制御信号を使って、HPIと通信を行います(図8−8参照)。ホスト デバイスおよび’54xの双方が、HPI制御レジスタにアクセスできます。

図8−8. ホスト ポート インターフェイス ブロック図



HPIは、連続する２バイトを組み合わせて16ビット ワードに変換することで、インターフェイス信号を節約した外部インターフェイスで16ビット データを通信可能にします。ホスト デバイスがデータ転送をHPIレジスタを用いて実行するとき、HPI制御ロジックが自動的に’54x内部デュアル アクセスRAMの専用2Kワード ブロックへのアクセスを実行して、その転送を完了します。’54xは、そのメモリ空間でデータにアクセスできるようになります。HPI RAMは、汎用デュアル アクセス データRAMまたはプログラムRAMとしても使用できます。

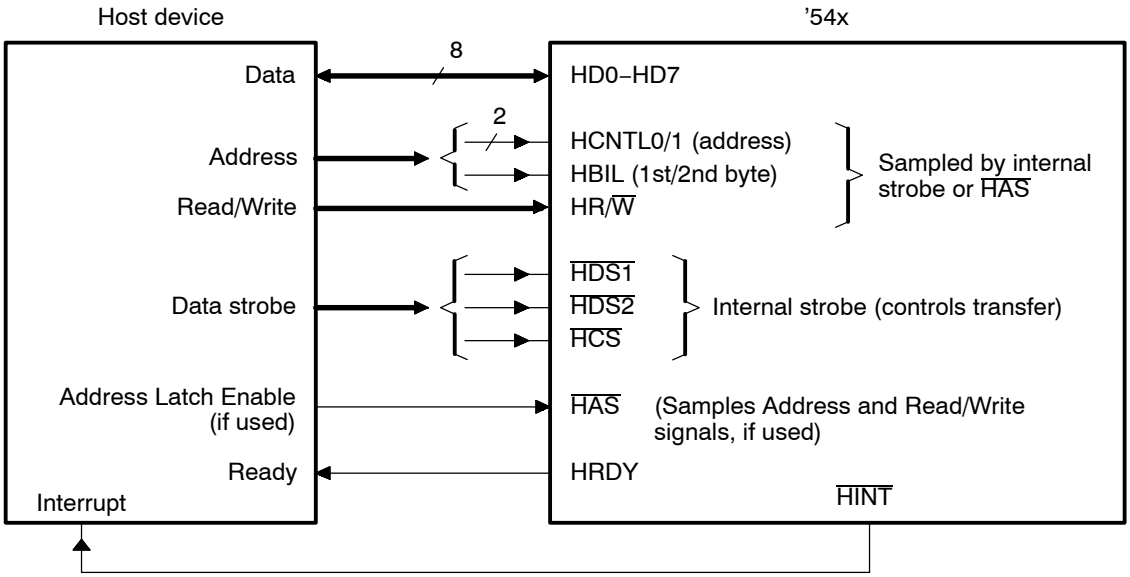
HPIには、共有アクセス モード(SAM)およびホスト専用モード(HOM)の2つの動作モードがあります。共有アクセス モード(通常の動作モード)では、’54xおよびホストの双方がHPI RAMにアクセスできます。このモードでは、’54xとホストのアクセスの衝突が起きた場合、ホストの非同期アクセスは’54xの内部で再同期され、ホストにアクセスの優先権があり54xは1サイクル待たされます。ホスト専用モードでは、’54xがリセットまたはIDLE2とIDLE3ですべての内部および外部クロックが停止しても、ホストはHPI RAMにアクセスできます。

HPIはホストの高速な連続アクセスをサポートします。共有アクセス モードでは、HPIは5CLKOUTサイクルごとに1バイトを転送でき、’54xが40MHz CLKOUTで動作した場合は64Mbpsとなります。HPIはこのように設計されているので、ホストは最大(Fd*n)/5の周波数で動作できます。ここでFdは’54xのCLKOUT周波数で、nはホストの外部アクセスのためのサイクル数になります。したがって、40MHzの’54xでホストの外部アクセス数nの値が4(または3)のとき、ホストはウェイト ステートは必要とせず、最大32(または24)MHzの速度で実行できます。ホスト専用モードでは、HPIは’54xのクロック速度にかかわらず、50nsごとに1バイト(160Mbps相当)の、ホストによるさらに高速な連続アクセスをサポートします(詳しいタイミング情報については、TMS320C54xデータ シートを参照)。

8.5.1 ホスト ポート インターフェイス機能の基本説明

外部HPIインターフェイスは、8ビット データ バスおよびインターフェイスを設定、制御する制御信号から構成されます。このインターフェイスは、各種のホスト デバイスに接続して追加のロジックはほとんど、あるいはまったく必要ありません。図8−9は、HPIおよびホスト デバイス接続の簡略図を示しています。

図8−9. システム ブロック全体図



8ビット データ バス(HD0−HD7)は、データをホストとやり取りします。54xは16ビット ワード構成のため、ホストとのあらゆる転送は必ず2つの連続バイトから構成されます。専用HBILピンは、第1バイトまたは第2バイトのどちらが転送されているかを示します。内部の制御レジスタのBOBビットは、第1バイトまたは第2バイトのどちらを16ビット ワードの上位バイトに置くかを決めます。ホストは、実行中のHPIアクセスの第1バイト/第2バイト(HBIL low/high)のシーケンスを中断できません。もし、このシーケンスが中断されると、データが失われる可能性があります。

2つの制御入力(HCNTL0およびHCNTL1)は、どの内部HPIレジスタにどのタイプのアクセスが行われるかを示します。これらの入力とHBILは、ホスト アドレス バスのピンから直接ドライブすることもできます。HCNTL0/1入力を使用することで、ホストはHPI制御(HPIC)レジスタ、HPIアドレス(HPIA)レジスタ(HPI RAMへのポインタとして働く)、またはデータ レジスタへのアクセスを指定できます。データ レジスタには、オプションの自動アドレス インクリメントによってもアクセスできます。

自動インクリメント機能は連続したHPI RAMへのアクセスを容易にする方法です。自動インクリメント モードでは、データ リードによってHPIAレジスタのポストインクリメントが発生し、データ ライトによってHPIAレジスタのプリインクリメントが発生します。HPICレジスタへのライトによりホストは'54x CPUに割り込みを行い、'54xではHINT出力を使ってのホストへの割り込みが可能です。ホストはHPICレジスタへのライトによりアクノリッジを発生させ、HINTをLowからHighに戻すことも可能です。

表8−13は3種類のレジスタ概要を示しますが、これらはHPIがホスト デバイスおよび'54x CPU間の通信およびこれらの制御のために用います。

表8−13. HPIレジスタの説明

名称	アドレス	説明
HPIA	−	HPIアドレス レジスタ。ホストのみアクセスできます。HPI RAMのアドレスを保持し、そのアドレスにより設定されるHPI RAMへのアクセスが発生します。
HPIC	002Ch	HPI制御レジスタ。ホストまたは'54xがアクセスできます。HPI動作のための制御およびステータス ビットを含んでいます。
HPID	−	HPIデータ レジスタ。ホストのみがアクセスできます。HPI RAMへのアクセスがリードの場合はHPI RAMからリードされたデータを保持し、アクセスがライトの場合はHPIに書かれるデータを保持します。

2つのデータ ストローブ($\overline{\text{HDS1}}$および$\overline{\text{HDS2}}$)、リード/ライト ストローブ(HR/$\overline{\text{W}}$)、アドレス ストローブ($\overline{\text{HAS}}$)により、ロジックの追加がほとんどなくても、HPIは各種のホスト デバイスに接続できます。HPIは基本的に、マルチプレクサ アドレス/データ バス、個別アドレス、データ バス、リード/ライト共通のデータ ストローブ、個別のストローブによってホストにインターフェイスできます。

HPIレディ ピン(HRDY)により、ホストに対するウェイト ステートの挿入が可能になります。ホストはHRDYを入力しアクセスにウェイト サイクルを挿入することができます。このHRDYによるタイミング制御により、HPIの通常動作より高速なインターフェイスを実現できます。HRDYが'54xから出力されるとき、ホストのタイミング条件と合わない場合は、そのシグナルは必要に応じて外部ロジックを使って再同期化して下さい。HRDYは、'54xの動作周波数が変動するとき、あるいはホストが共有アクセス モードの最大アクセス速度より速くアクセスできるときに使用します(ホスト専用モードの最大アクセス速度まで)。両方のケースでHRDYピンを使用することにより、ホスト アクセス速度が'54xクロックより高速でも簡単にアクセスできます。

これらすべての機能が組み合わさり、HPIが各種のホスト デバイスと柔軟、効率的にインターフェイスできます。また、HPIインターフェイスがシンプルであることから、ホストおよび'54x側のインターフェイスからのデータ転送が大幅に簡素化されます。一度インターフェイスを構成すれば、データ転送は最小限のオーバーヘッドおよび最大限の速度で実行できます。

8.5.2 ホスト ポート インターフェイス動作の詳細

このサブセクションでは、各HPI外部インターフェイス ピンの機能について詳しく説明します。また、レジスタおよび制御ビットの機能についても説明します。論理的なインターフェイスのタイミング、初期化およびリード/ライト シーケンスについては、8−37ページのサブセクション8.5.3ホストのHPIへのリード/ライト アクセスを参照してください。

外部HPIインターフェイス信号は、各種ホスト デバイスへの柔軟性のあるインターフェイスを実現します。個別または共有のデータ ストローブを持つデバイス、およびアドレス ラッチ イネーブル(ALE)信号を使用する、しないに関わらず、HPIに簡単に接続できます。

表8−14には、各HPI外部インターフェイス ピンの機能についての詳しい説明があります。

表8−14. HPIシグナル名称および機能

HPIピン	ホスト ピン	ステート†	シグナル ファンクション
$\overline{\text{HAS}}$	アドレス ラッチ イネーブル(ALE)、アドレス ストローブ、使用しない場合はハイ	I	アドレス ストローブ入力。マルチプレクサされたアドレスおよびデータ バスを持ったホストが、$\overline{\text{HAS}}$をALEピンに接続します。次にHBIL、HCNTL0/1、HR/$\overline{\text{W}}$が$\overline{\text{HAS}}$の立ち下がりエッジにラッチされます。$\overline{\text{HAS}}$は使用時に、$\overline{\text{HCS}}$、$\overline{\text{HDS1}}$、$\overline{\text{HDS2}}$のうち最も遅いものより必ず先行してアドレスおよび条件制御信号をラッチします(詳しいHPIタイミング仕様については'54xデータ シートを参照してください)。アドレスとデータ バスが独立したホストは、$\overline{\text{HAS}}$をHighに接続できます。この場合、HBIL、HCNTL0/1、HR/$\overline{\text{W}}$が、$\overline{\text{HDS1}}$、$\overline{\text{HDS2}}$、または$\overline{\text{HCS}}$の最も遅いものの立ち下がりエッジによってラッチされ、$\overline{\text{HAS}}$の役割を果たします。
HBIL	アドレスまたは制御ライン	I	バイト識別入力。転送の第1または第2バイトを識別します(ただし、上位バイトまたは下位バイトは識別しません。これはHPICのBOBビットによって指定されます。本セクション後半を参照してください)。第一バイト転送時、HBILをローとし、第2バイトを転送する時はハイとします。
HCNTL0, HCNTL1	アドレスまたは制御ライン	I	ホスト制御入力。HPIAレジスタへのホスト アクセス、HPIデータ ラッチ(オプションでアドレス インクリメントをともなう)、またはHPICレジスタを選択します。
$\overline{\text{HCS}}$	アドレスまたは制御ライン	I	チップ セレクト。HPIのイネーブル入力の役割を持ち、アクセス時は必ずローですがそれ以外の時にローでもかまいません。$\overline{\text{HCS}}$は通常$\overline{\text{HDS1}}$および$\overline{\text{HDS2}}$に先行しますが、このシグナルは$\overline{\text{HAS}}$が使用されず$\overline{\text{HDS1}}$または$\overline{\text{HDS2}}$がすでにローのとき、HCNTL0/1、HR/$\overline{\text{W}}$、HBILをサンプルします(このサブセクションで以降に詳述します)。8−33ページの図8−10は、$\overline{\text{HCS}}$相当の回路、$\overline{\text{HDS1}}$および$\overline{\text{HDS2}}$入力を示しています。

† I: 入力
O: 出力
Z: ハイ インピーダンス

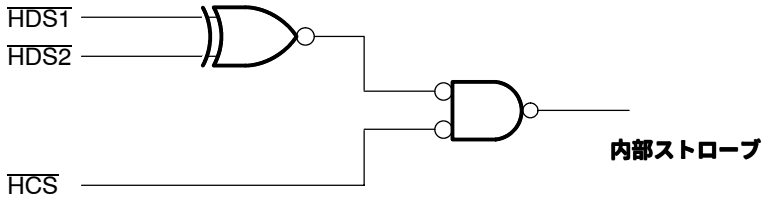
表8−14. HPIシグナル名称および機能(続き)

HPIピン	ホスト ピン	ステート†	シグナル ファンクション
HD0−HD7	データ バス	I/O/Z	パラレル双方向3ステート データ バス。HD7(MSB)からHD0(LSB)は、($\overline{HDSx}$ I $\overline{HCS}$=1)を出力しなしとき、またはEMU1/$\overline{OFF}$がアクティブ(low)のとき、ハイ インピーダンス状態となります。
$\overline{HDS1}$, $\overline{HDS2}$	リード ストローブおよびライト ストローブまたはデータ ストローブ	I	データ ストローブ 入力。ホスト アクセス サイクル時にデータ転送を制御します。また、$\overline{HCS}$がすでにローのとき(通常動作の場合)で、$\overline{HAS}$が使用されない場合に、HBIL、HCNTL0/1、HR/$\overline{W}$をストローブの立ち下がりでサンプルします。独立したリードおよびライト ストローブを持つホストは、個々のストローブを$\overline{HDS1}$と$\overline{HDS2}$に接続します。シングル データ ストローブを持つホストは、そのストローブを$\overline{HDS1}$または$\overline{HDS2}$に接続し、使用されないピンをハイに接続します。$\overline{HDS1}$および$\overline{HDS2}$は内部的にxNORがなされるので、ハイ アクティブのデータ ストローブを持つホストはこれをHDS入力のひとつに接続し、他方のHDS入力をローに固定します。8−33ページの図8−10は、$\overline{HDS1}$、$\overline{HDS2}$、$\overline{HCS}$入力に相当する回路を示しています。HDSの接続にかかわらず、HR/$\overline{W}$は転送方向を決めるのに必要です。
$\overline{HINT}$	ホスト割り込み出力	O/Z	ホスト割り込み出力。HPICのHINTビットによって制御されます。'54xがリセットされるとハイになります。EMU1/$\overline{OFF}$ピンがローのときハイ インピーダンスになります。
HRDY	非同期レディ	O/Z	HPIレディ出力。ハイのとき、HPIが転送実行のための準備状態であることを示します。ローのときは、HPIが前の転送による内部動作を実行中であることを示します。EMU1/$\overline{OFF}$ピンがローのときハイ インピーダンスになります。$\overline{HCS}$はHRDYをイネーブルにしますが、これは、$\overline{HCS}$がハイのときHRDYは常にハイとなることを意味します。
HR/$\overline{W}$	リード/ライト ストローブ、アドレス ライン、またはマルチプレクサ アドレス/データ	I	リード/ライト入力。ホストは必ずHR/$\overline{W}$をハイにドライブしてHPIをリードし、ローでHPIにライトを行います。リード/ライト ストローブを持たないホストは、この信号をホストのアドレス ラインにより制御することができます。

† I: 入力
O: 出力
Z: ハイ インピーダンス

$\overline{\text{HCS}}$入力は主にHPIのイネーブル入力の役割を持ち、$\overline{\text{HDS1}}$および$\overline{\text{HDS2}}$信号はHPIデータ転送を制御しますが、システムに応じてHCS、HDS1、HDS2のいずれかを使用してHPIへのアクセスを制御可能です。$\overline{\text{HCS}}$をHPIデータ転送制御として使用し、HPIアクセス サイクルを制御する場合は、HRDY動作が影響を受けます($\overline{\text{HCS}}$がHRDYをイネーブルにして、HRDYは$\overline{\text{HCS}}$がhighのとき常にhighのため)。これらの入力の等価回路は、図8−10に示されます。この図は、HCNTL0/1、HBIL、HR/W入力をサンプルする($\overline{\text{HAS}}$が使用されないとき)内部ストローブ 信号が、3種類の入力信号すべてから生成されることを示しています。したがって、$\overline{\text{HDS1}}$、$\overline{\text{HDS2}}$、$\overline{\text{HCS}}$のうち最も遅いものが、実際にHCNTL0/1、HBIL、HR/$\overline{\text{W}}$入力のサンプリングを行います。$\overline{\text{HDS1}}$および$\overline{\text{HDS2}}$はXNORされるので、入力信号が全て0の場合はイネーブルの状態となりません。

図8−10. 選択入力ロジック



$\overline{\text{HAS}}$入力を使ってHCNTL0/1、HBIL、HR/$\overline{\text{W}}$をサンプルする場合、これらの信号をアクセス サイクルの早い段階でサンプルできるので、バス ステートをアドレスからデータ情報に切り替える時間を取ることができ、マルチプレックス アドレスおよびデータ バスへのインターフェイスを実現します。このタイプのシステムではALE信号があり、この信号を通常$\overline{\text{HAS}}$に接続します。

2つの制御ピン(HCNTL0およびHCNTL1)は、いずれの内部HPIレジスタ(HPIA、HPID、HPIC)にどのタイプのアクセスが行われるかを切り替えます。これら2つのピンの入力は、HPIアドレス(HPIA)、HPIデータ(HPID)、HPI制御(HPIC)レジスタへのアクセスを選択します。HPIAレジスタはHPIメモリへのポインタとして働き、HPICは転送の制御およびステータス ビットを保持し、HPIDは実際に転送されたデータを保持します。さらに、オプションの自動アドレッシング インクリメントによるデータ転送の設定も、この制御ピンで行われます。表8−15には、HCNTL0/1ビットの機能の説明があります。

表8−15. HPI入力制御シグナル ファンクション選択の説明

HCNTL1	HCNTL0	説明
0	0	ホストは、HPI制御レジスタのHPICを読み書きできます。
0	1	ホストは、HPIデータのHPIDをリード/ライトできます。HPIAは、リード実行のたびに自動的にポストインクリメントされ、ライト実行のたびにプリインクリメントされます。
1	0	ホストはHPIアドレスのHPIAをリード/ライトできます。このレジスタはHPI RAMのアクセス位置をポイントします。
1	1	ホストは、HPIデータのHPIDをリード/ライトできます。このときHPIAは影響を受けません。

’54xでは、HPIメモリは2K x 16ビット ワード ブロックのデュアル アクセスRAMで、データ メモリ空間の1000hから17FFhに置かれます。またオプションで、OVLYビットの設定に応じて、プログラム メモリ空間に置くこともできます。

ホストからは、2Kワード ブロックのHPI RAMのアドレス0から7FFhにアクセスできます。ただし、HPIAの16ビットのうち、上位5ビットは無視されます。たとえば、HPI RAMの1000hにアドレスするにはHPIAは、0000h、0800h、1000h、1800h.....F800hのいずれかの値でも可能です。

HPI自動インクリメント機能により、HPIメモリの連続するワード位置にアドレスの制御なしにアクセスすることができます。自動インクリメント モードでは、データ リードがHPIAのポストインクリメントを行ってデータ ライトがHPIAのプリインクリメントを行います。自動インクリメント モードでは、データ リード時にアドレスをポストインクリメントし、ライト時にプリインクリメントを行います。従って、ホストがHPI RAMの800hを自動インクリメント機能を用いてライトする場合にはHPIAに7FFhをロードする必要があります。HPIAのインクリメントおよびデクリメントは、このレジスタの16ビット全てに影響します。

HPI制御レジスタ ビットおよび機能

HPIの動作を制御する4つのビットには、BOB(第1または第2バイトを上位バイトとして指定)、SMOD(ホスト オンリー アクセス モードまたは共有アクセス モードの選択)、DSPINT(割り込みベクタではHPINTと表記されています)およびHINT(’54xおよびホストへの割り込みを発生するのに使用)があります。これらは、HPI制御レジスタ(HPIC)に配置されています。HPIC ビット ファンクションの詳しい説明については表8−16を参照してください。

表8−16. HPI制御レジスタ(HPIC)ビットの説明

ビット	ホスト アクセス	54xアクセス	説明
BOB	リード/ライト	−	BOB=1のとき、最初に転送されるバイトは下位バイトです。BOB=0のとき、最初に転送されるバイトは上位バイトです。BOBはデータおよびアドレス転送の双方に影響します。ホストのみがこのビットを設定でき、'54xからはアクセスできません。BOBは必ず、最初のデータ アクセスまたはアドレス レジスタのアクセスの前に初期化してください。
SMOD	リード	リード/ライト	SMOD＝1のとき、共有アクセス モード(SAM)となり、'54xはHPI RAMにアクセスできます。SMOD=0のとき、ホスト専用モード(HOM)となり、'54xはHPI RAMにアクセスすることができません。リセット時にSMOD=0になり、リセット解除後にSMOD=1になります。SMODは'54xのみが設定できますが、'54xおよびホストの双方がリード可能です。
DSPINT	ライト	−	ホスト プロセッサから'54xへの割り込み。このビットはホストのみがライトでき、ホストまたは'54xともリードできません。ホストがこのビットに1を書き込むと、'54xに割り込みが発生します。このビットを読み込むと常に0が得られます。ホストがHPICにライトするとき、両方のバイトは必ず同じ値をライトしてください。DSPINTについては、以降の本サブセクションを参照してください。
HINT	リード/ライト	リード/ライト	このビットは'54xからHINTを出力するために用います。$\overline{\text{HINT}}$を使ってホストに割り込みを発生できます。リセット時にHINT=0とし、外部$\overline{\text{HINT}}$をインアクティブ(ハイ)にします。HINTビットは'54xだけがセットでき、ホストだけがクリアできます。'54xは1をHINTビットに書き込み、$\overline{\text{HINT}}$ピンをローにドライブします。HINTビットは、外部$\overline{\text{HINT}}$ピンがインアクティブ(ハイ)のときは0として、$\overline{\text{HINT}}$ピンがアクティブ(ロー)のときは1として、ホストまたは'54xによって読み込まれます。ただし、ホストが割り込みをクリアするには、ホストはHINTビットに1を書き込む必要があります。ホストまたは'54xがHINTビットに0を書き込んでも、特に効果はありません。HINTの機能については、本サブセクションを参照してください。

ホストは常に転送を8ビットで実行して、制御レジスタは通常インターフェイスを初期化するためにアクセスされる最初のレジスタなので、HPICレジスタはホストより上位と下位に同一の値をセットされ、'54x側からはHIPCレジスタの上位は未使用となっています。制御/ステータス ビットは、下位4ビットに配置されます。ホストは前述のように、HCNTL0/1の選択によってHPICにアクセスし、2つの連続するバイトが8ビットHPIデータ バスにアクセスします。ホストがHPICにライトを行うとき、書き込まれる第1および第2バイトは必ず同じ値としてください。'54xは、データ メモリ空間の002ChにHPICレジスタが配置されています。

図8−11から図8−14は、HPICレジスタのビット配置を示しています。図中の0はリード専用ビットを示し、Xは未定義ビットを示しています。また、Xの領域をリードした場合、読み込まれる数値は不定です。ライトの処理では、Xにはどの値でも書き込めます。ホストの書き込みでは、上位/下位のバイトを必ず同じにしてください。ホスト側のビット4−7および12−15、および'54x側のビット4−15は将来の拡張用に予約になっています。

図8−11. HPIC図—ホストのHPICからのリード

15−12	11	10	9	8	7−4	3	2	1	0
X	HINT	0	SMOD	BOB	X	HINT	0	SMOD	BOB

注： X=不定の値をリードします。

図8−12. HPIC図—ホストのHPICへのライト

15−12	11	10	9	8	7−4	3	2	1	0
X	HINT	DSPINT	X	BOB	X	HINT	DSPINT	X	BOB

注： X=あらゆる値をライトします。

8

図8−13. HPIC図—TMS320C54xのHPICからのリード

15−4	3	2	1	0
X	HINT	0	SMOD	0

注： X=不定の値をリードします。

図8−14. HPIC図—TMS320C54xのHPICへのライト

15−4	3	2	1	0
X	HINT	X	SMOD	X

注： X=あらゆる値をライトします。

ホストはHPICを上位と下位の8bitずつリードする為、この2回のリードの間に'54xがライト可能なSMOD、HINTを変更する可能性があります。この場合ホストは第1バイトと第2バイトで異なるデータをリードすることになります。ホストおよび'54xのHPICリード/ライト サイクルについては、表8−17を参照してください。

表8−17. HPICのホスト/TMS320C54xリード/ライト特性

デバイス	リード	ライト
ホスト	2バイト	2バイト(両バイトは必ず等しい)
'54x	16ビット	16ビット

8.5.3 ホストのHPIへのリード ライト アクセス

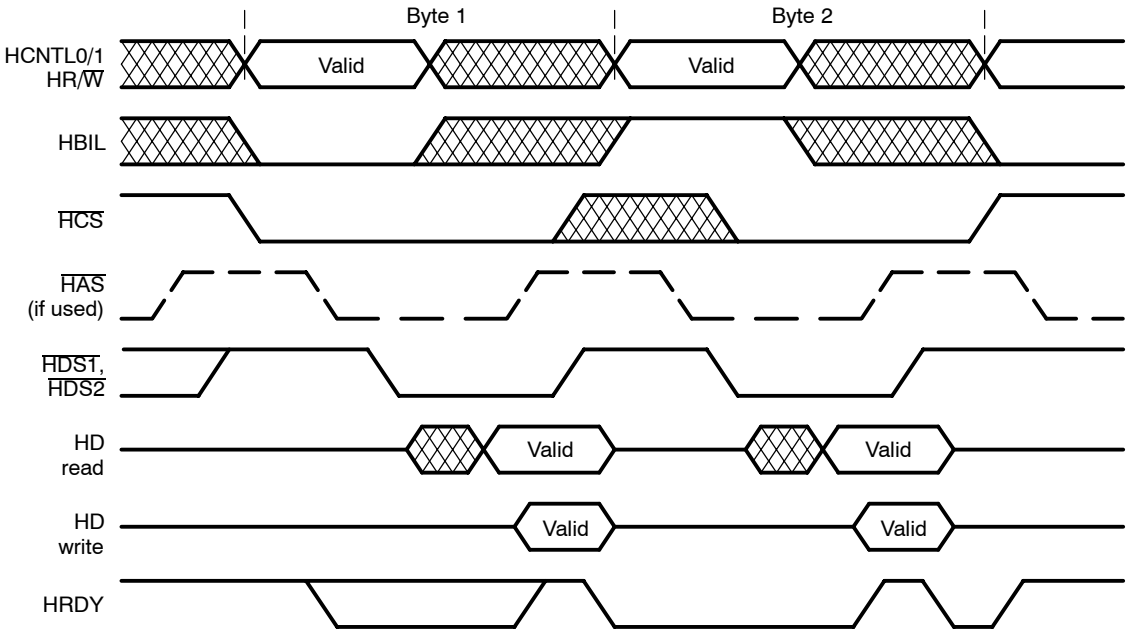
ホストは、サイクルの外部インターフェイス部分を実行することにより、HPIアクセスを開始します。つまり、最初にHPICレジスタを初期化して次にHPIAレジスタを初期化します。そしてHPIDレジスタからデータをリード、またはHPIDレジスタへのデータ ライトを実行します。HPIAまたはHPIDへのライトにより、内部サイクルを始動して任意のデータをHPIDおよび専用内部HPI RAMの間で転送します。HPIアクセスを実行するたびに、HPIDにライトされるデータはホスト アクセス サイクルの後まではHPIメモリにライトされず、HPIDからリードされるデータは前のサイクルからのデータになります。したがってリード時に、取得されるデータは前のアクセスで指定される位置からのデータとなり、カレント アクセスは次のサイクルを始動する役割を持ちます。同様のシーケンスがライト処理でも起こります。HPIDにライトされるデータは、外部サイクルが完了する後までHPI RAMにライトされません。HPIDリード処理がHPIDライト処理の直後に実行される場合は、同じデータ(書き込まれたデータ)がリードされます。

HPIAで使用可能な自動インクリメント機能により、ホストによるHPIへのきわめて効率的な連続的アクセスが実現します。長時間のアクセスの間、'54xはホストのリードおよび前のホスト データのリード/ライトまたはHPIAライト アクセスの間で、アクセスされる位置の内容を変更する場合があります。これは、内部的なHPI処理のプリフェッチ特性によるものです。このような処理が発生する場合、カレント メモリの内容と異なるデータがリードされる可能性があります。したがって、システムでこのようなケースがある場合は、同じアドレスからの2つのリードまたはリード アクセスの前のアドレス ライトが実行され、確実に最新のデータをリードするようにします。

ホストがHPIの外部アクセスを実行するとき、2つの異なるタイプのサイクルが発生する場合があります。それらは、ウェイト ステートが発生(HRDY信号がアクティブ)するものと、ウェイト ステートをともなわないものです。通常、共有アクセス モード(SAM)のときはHRDY信号が使用され、ホスト専用モード(HOM)のときはHRDYはインアクティブでハイのままになります。ただし例外もありますが、それについては以降で説明します。

HRDY信号を利用するアクセスでは、内部で転送が実行されている間は(リードまたはライト)、HRDYはローで別の転送を受け付けないことを示します。内部アクセスが完了して別の外部アクセスを開始できるようになると、HRDYはHPIによってハイでドライブします。これは、サイクル開始に続く固定遅延の後に発生します(HPI外部インターフェイス タイミングの詳しいタイミング情報については、'54xデータ シートを参照してください)。したがって、back−to−backサイクルが実行されないうちは、サイクルの最初のバイトが転送されるときHRDYは通常ハイになります。図8−15のタイミング図は、HRDYを使う外部HPIサイクルを示しています。

図8−15. HPI**タイミング図**



典型的な外部アクセスでは、図8−15**に見られるように、サイクルは**HCNTL0/1、HR/W、HBIL、HCS**を駆動するホストとともに始まり、これらはどのタイプの転送が発生してサイクルがリードかライトになるかを示します。次に、ホストが**HAS**信号を出力して(HAS**信号を使用する場合)データ ストローブのひとつが続きます。**HRDY**がすでにローの場合は、内部サイクルが完了するときハイになり、データ転送が可能になると**HRDY**信号はインアクティブとなります。外部**HPI**サイクルに続いて**HRDY**はローになり約**5CLKOUT**サイクルの間ローを保ちます(HPI**タイミングについては'54x**データ シートを参照してください)。その間、'54x**は内部**HPI RAM アクセスを完了して、HRDY**が再びハイになります。ただし、HCS**がハイのときは**HRDY**は常にハイになります。**

前述のように、SAMアクセスは通常**HRDY**信号を利用します。**HPIC**または**HPIA**リードまたは**HPIC**へのライト時に(DSPINT**または**HINT**のいずれかに**1**をライトする場合を除く)、SAM**のとき**HRDY**ベースのインターフェイス タイミングの例外が発生します。このような場合、HRDY**はハイのままになります。その他の**SAM**アクセスでは、HRDY**はアクティブになります。**

HOMのときホスト アクセス サイクルには、前述のSAMタイミングとは異なるタイミングとなります。HOMではCPUは無関係になり(1つの例外を除き)、短い固定遅延時間の後にアクセスを完了できます。これに対する例外は、HPICのDSPINTまたはHINTビットに1をライトするときに発生します。この場合、ホスト アクセスにはいくらかのCPUクロック サイクルがかかり、SAMタイミングを適用します。HRDYタイミングおよびより高速なサイクル時間の他に、HOMアクセス サイクルが論理的にSAMアクセス サイクルと同じになります。表8−18は条件の概要を示していますが、ここではHRDY信号はホスト アクセスのためにアクティブ(SAMタイミングを適用)になります。HRDYがインアクティブ(HRDYがハイのまま)のときは、HOMタイミングを提供します。HPIタイミング仕様については、'54xデータ シートを参照してください。

表8−18. ウェイト ステート発生条件

	ウェイト ステート発生	
レジスタ	リード	ライト
HPIC	No	DSPINT/HINTへの1 − YES その他すべてのサイクル − No
HPIA	No	HOM − No SAM − Yes
HPID	HOM − No SAM − Yes	HOM − No SAM − Yes

アクセス シーケンスの例

2バイト アクセスによりホストのアクセスが完了します。最初のバイトはHBILロー、2番目のバイトはHBILハイです。この2バイト シーケンスは、ホスト アクセスのタイプにかかわらず(HPIA、HPICまたはデータ アクセス)必ずひと続きになり、ホストは実行中のHPIアクセスの第1バイトおよび第2バイト(HBILロー/ハイ)のシーケンスを中断できません。このシーケンスが中断されると、データが消失します。

データにアクセスする前に、ホストは必ずHPICを初期化して、次にHPIAをセットします(BOBがHPIAアクセスに影響するのでこの順番になります)。HPI RAMのリード では、HPIAのセットの完了後にHPI RAMをリードして、特定アドレスの内容を2つの8ビット データ ラッチ(第1バイト データ ラッチおよび第2バイト データ ラッチ)に送ります。表8−19は、HPI RAM リードのためのBOBおよびHPIAのセットに関連するシーケンスを示しています。この例では、BOBが0にクリアされ、FFFEhのHPI RAMの位置(ここでは1000h)でリードが要求されます。

表8−19. BOBおよびHPIAの初期化

イベント	HD	HR/$\overline{W}$	HCNTL1/0	HBIL	HPIC	HPIA	ラッチ 1	ラッチ 2
ホストがHPICにライト、第1バイト	00	0	00	0	00xx	xxxx	xxxx	xxxx
ホストがHPICにライト、第2バイト	00	0	00	1	0000	xxxx	xxxx	xxxx
ホストがHPIAにライト、第1バイト	10	0	10	0	0000	10xx	xxxx	xxxx
ホストがHPIAにライト、第2バイト	00	0	10	1		1000	xxxx	xxxx
内部HPI RAMリード完了						1000	FF	FE

表8−19に示されるサイクルでは、BOBおよびHPIAが初期化され、HPIAをロードすることにより内部HPI RAM アクセスが始まります。表8−19の最後の行は、内部RAMリードが完了した後のHPIの条件を示しており、HPIAに対する第2バイトのホスト ライトが終了した後に、リードが完了してデータが上位または下位バイトのデータ ラッチにセットされます。ホストが実際にこのデータを取得するには、ホストは次にHPIDリードを実行する必要があります。このHPIDリードのアクセス時に、HBILがローのとき第1バイト データ ラッチの内容がHDピンに現れ、HBILがハイのとき第2バイト データ ラッチの内容がHDピンに現れます。次に、自動インクリメントが選択されている場合はアドレスがインクリメントされ、メモリが再びデータ ラッチにリードされます。表8−20には、このアクセスに関わるシーケンスが示されます。

表8−20. HPIへの自動インクリメントをともなうリード アクセス

イベント	HD	HR/$\overline{W}$	HCNTL1/0	HBIL	HPIC	HPIA	ラッチ 1	ラッチ 2
ホストがデータをリード、第1バイト	FF	1	01	0	0000	1000	FF	FE
ホストがデータをリード、第2バイト	FE	1	01	1	0000	1000	FF	FE
内部HPI RAMリード完了						1001	6A	BC

表8−20に示されるアクセスでは、HPIDリードから取得するデータは、前のサイクルで始まるリードによるデータになり(表8−19に示される)、表8−20のように実行されるアクセスが、この時点では1001hで再びリードを始めます(このアクセスでHCNTL1/0を01に設定することにより自動インクリメントが指定されることによります)。また自動インクリメントが選択されると、インクリメントは各16ビット ワード転送で発生します(各バイトでは発生しません)。したがって、表8−20に示されるように、HPIAは1ずつインクリメントされます。表8−20の最後の行は、2番目の内部RAMリードが完了した後、1001h(6ABCh)位置の内容がリードされ上位および下位バイトのデータ ラッチにセットされることを示しています。

HPIへのライト アクセスの際、HBILピンがローの間に第1バイト データ ラッチがホストからのデータによって上書きされます。HBILピンがハイの間に第2バイト データ ラッチがホストからのデータによって上書きされます。このライト アクセスの終了時に、両方のデータ ラッチのデータが16ビット ワードとしてHPIAレジスタによって指定されるHPI RAMのアドレスに送られます。ここでは自動インクリメントが選択されているので、このアドレスはメモリ ライトの前にインクリメントされています。

表8−21は、HPIライト アクセスを示しています。この例では、内部HPI RAMへのライトが完了したときは、HPI RAMの1002hの位置には1234hがあります。同じアドレスのリードがこのライトに続くと、データ ラッチ(1234h)にライトされたのと同じデータが読み返されます。

表8−21. HPIへの自動インクリメントをともなうライト アクセス

イベント	HD	HR/$\overline{W}$	HCNTL1/0	HBIL	HPIC	HPIA	ラッチ 1	ラッチ 2
ホストがデータをライト、第1バイト	12	0	01	0	0000	1002	12	FE
ホストがデータをライト、第2バイト	34	0	01	1	0000	1002	12	34
内部HPI RAMライト完了						1002	12	34

8

8.5.4 DSPINTおよびHINTファンクションの動作

ホストおよび'54xは、HPICレジスタのビットを使用して相互に割り込みを実行できます。本サブセクションでは、この処理についてを説明します。

ホスト デバイスのDSPINTのよる'54xへの割り込み

'54xへの割り込みは、ホストがHPICのDSPINTに1をライトするときに発生します。この割り込みを使って、'54xをIDLEからウェイク アップできます。ホストおよび'54xは常にこのビットを0としてリードします。'54xはライトできません。ホストがDSPINTに1をライトした後は、再びデバイスによって0をライトする必要がなく、また、このビットに0をライトしても影響はありません。BOBまたはHINTにライト中は、ホストはDSPINTビットに1をライトすると、'54xの予期せぬ(不要な)割り込みが発生します。

'54xでは、ホストから'54xへの割り込みベクトル アドレスはxx64hになります。この割り込みはIMR/IFR[9:9]に位置します。'54xの割り込みベクトルをHPIメモリに再マップできることから、ホストは'54xに、アドレスxx64hにある分岐命令によって'54xに割り込みを行う前にHPIメモリのxx65hアドレスにファンクションの開始アドレスをライトすることによりあらかじめプログラムされたファンクションを実行するように命令できます。割り込みベクタがホスト ポートからアクセス可能なオンチップRAMに再マッピングされる場合は、必ずSAMを使い、ホストはxx00hからxx7F(xx65hを除く)の位置にはライトできません。

ホスト ポート インターフェイスのHINTによるホスト デバイスへの割り込み(’54x)

’54xがHPICのHINTビットに1をライトするとき、$\overline{\text{HINT}}$の出力はロー・レベルにドライブされます。’54xまたはホストは、HINTビットを1としてリードします。$\overline{\text{HINT}}$信号を使って、ホストに割り込みを実行できます。$\overline{\text{HINT}}$の割り込みラインを検出した時に、ホスト デバイスはHINTビットに1をライトすることにより、’54xの割り込みに応答してHINTビットをクリアできます。HINTビットがクリアされると’54xまたはホストが0としてリードし、$\overline{\text{HINT}}$ピンはハイ・レベルにドライブされます。’54xまたはホストが0をライトしても、HINTビットは変化しません。SMODビットにアクセスする際、’54xはホストへの割り込みが必要なければHINTビットに1をライトしません。

8.5.5 HPI RAM アクセス モード(SAM/HOM)の切り替えおよびIDLE2/3使用上の注意点

HPIのホスト専用モード(HOM)においてホストはHPI RAMにアクセスでき、’54xはIDLE2/3(VCO停止)になります。さらに、最小電力消費設定として’54xへの外部クロック入力を停止することもできます。このような条件下で、各アクセスのために外部クロックをウエイク アップすることなく、また’54xオンチップPLLが使用される際はロックアップ時間を待つことなく、ランダム アクセスを実行できます。外部クロックは、’54xをIDLE2/3から戻すときにウエイク アップする必要があります。

’54xがIDLE2/3のとき、ホストはSAMモードのHPI RAMにアクセスできません。これは、このモードでの実行にCPUクロックが必要だからです。したがって、ホストがHPI RAMへのアクセスを必要として’54xがIDLE2/3のとき、IDLE2/3に入る前に’54xはHPIをHOMモードに変更する必要があります。HPIがHOMモードのとき、’54xはHPICにアクセスしてSMODビットを切り替えるか割り込みをホストに送ることができますが、HIP RAMブロックにはアクセスできません。’54xのHPI RAMへのアクセスはHOMでは実行できません。’54xがHPI RAMブロックに再びアクセスするためには、IDLE2/3から戻った後にSAMに変更する必要があります。

HOMモードを選択するには、HPICのSMODビットに0をライトする必要があります。SAMモードを選択するには、SMODに1をライトします。HOMおよびSAMを相互に切り替えるときには、次の2つの規定があります。第1点は、SAMからHOMに変更する命令の直後に続く命令はIDLE2またはIDLE3であってはいけません。この場合、SAMからHOMへの切り替えにおける’54xのパイプライン遅延のために、IDLE2/3がモード切り替えより前に発生して、HPIがSAMのままになるからです。したがって、ホスト アクセスを発生することはできません。

第2点は、HOMからSAMへの切り替え時に、HOMからSAMに切り替える命令の直後に続く命令は、HPI RAMブロックをリードできません。これは、実際にはHPI RAMのリードが発生するときにモードが変更されておらず、HPI RAMからのリードが無視されるからです。HPI RAMへのライトはこの制約には含まれません。HOMからSAMに切り替える命令の直後に続く命令でHPI RAMライトは実行できます。なぜならその実行はパイプラインのかなり後方で発生するからです。

ホスト側では、モード変更に関する特別な注意点はありません。たとえば、'54xをIDLE2/3からウェイク アップさせ、ホストとのソフトウェア ハンドシェイクなしにウェイク アップ時にSAMに切り替えるようにすることも可能です。またHPIのモード変更時に、ホストはアクセスを続行できます。ただし、ホストがHPI RAMにアクセスしながらモードを変更する場合は、実際のモード変更はホストのアクセスが完了するまで行なわれません。この場合、'54xがHPIメモリへアクセスするのも遅れます。

表8−22は、HPIを使用する際に'54xがIDLE2/3状態かIDLE2/3から戻るときのイベント シーケンスを示しています。この過程全てにおいて、HPIはホストにアクセスできます。

表8−22. IDLE2/3になるまたはIDLE2/3から戻るためのシーケンス

ホストまたは他のデバイス	'54x	モード	'54xクロック
	モードをHOMに切り替え	HOM	実行
	NOPを実行	HOM	実行
	IDLE2またはIDLE3命令を実行	HOM	実行
DSPクロックを停止も可能	IDLE2/3モード	HOM	停止または実行
DSPクロックが停止されている場合、これをオンにする†	IDLE2/3モード	HOM	実行
割り込みをDSPに送る	IDLE2/3モード	HOM	実行
	'54xがIDLE2/3からウェイク アップ	HOM	実行
	'54xがSAMにモード切り替え	SAM	実行

† '54xオンチップPLL使用時は、十分なウェイク アップ時間が必要。

8.5.6 リセット時のHPIメモリへのアクセス

ホストはHPIに'54xがリセットの間アクセスできるので、プログラムまたはデータをHPI RAMにダウンロードできます。この機能を使用するとき、ホストが'54xのリセット入力を制御すると有効です。表8−23は、'54xをリセットするため、および'54xのリセット時にプログラムをHPI RAMにダウンロードするためのイベントのシーケンスを示しています。

まず、'54xのリセット ピンをローにドライブする'54xの動作クロックの6クロック分以上前には、ホストはHPIへのアクセスを停止する必要があります。次にホストは'54xのリセット ピンをローにして、'54xの4クロック分以上後にHPIへのアクセスを開始できます。リセットの間はHPIモードは自動的にHOMに設定され、高速なプログラムのダウンロードが可能になります。このとき'54xのクロックを停止することもできますが、リセット ピンが'54xの適切なリセット動作のために立ち下がりおよび立ち上がるときは、クロックが実行する必要があります。

ホストがHPI RAMへのダウンロードを完了すると、ホストはHPIへのアクセスを停止して'54xのリセット ピンをハイにドライブします。リセット信号の立ち上がり後、'54xの動作クロックの20クロック分以上後に、ホストは再びHPIへのアクセスを開始できます。このクロック数は、'54xの内部リセットの遅延によるものです。HPIモードはリセットが終了すると自動的にSAMに設定されます。

'54xがリセットの状態でホストがDSPINTに1をライトする場合は、'54xがリセットから戻ると割り込みは失われます。'54xのウォーム ブートはHPI RAMを使って、最下位のHPIアドレスから実行を開始できます。

表8−23. RESET時のHPI動作

ホスト	'54x	モード	'54x CLK
動作クロックの6サイクル待ち	実行	X	実行
RESETをlowにして4クロック待ち	リセットになる	HOM	実行
'54xクロックを停止可能	リセット中	HOM	停止または実行
HPIメモリにプログラムまたはデータを書き込み	リセット中	HOM	停止または実行
停止の場合DSPクロックをオン†	リセット中	HOM	実行
RESETをhigh	リセット中	HOM	実行
動作クロックの20サイクル待ち	リセットから戻る	SAM	実行
HPIにアクセス可能	実行	SAM	実行

† '54xオンチップPLL使用時は、十分なウェイク アップ時間が必要。

第9章

シリアル ポート

本章では、’54xのコアCPUに接続する3種類のシリアル ポート インターフェイスについて説明します。

- ❑ スタンダード同期シリアル ポート インターフェイス
- ❑ バッファド シリアル ポート インターフェイス
- ❑ 時分割多重シリアル ポート インターフェイス

これらのペリフェラルは、メモリ マップにあるレジスタによって制御されます。シリアル ポートは割り込みによってコアCPUに同期します。

Topic Page

9

9.1 シリアル ポートの概要

'54xデバイスには、3種類の同期シリアル ポート インターフェイスがあります。

- ❑ スタンダード同期シリアル ポート インターフェイス
- ❑ バッファド シリアル ポート(BSP)インターフェイス
- ❑ 時分割多重(TDM)シリアル ポート インターフェイス

表9−1は、各種'54xデバイスが持つシリアル ポートを示しています。

表9−1. TMS320C54xデバイスのシリアル ポート

デバイス	スタンダード 同期シリアル ポート	バッファド シリアル ポート	時分割多重(TDM) シリアル ポート
'541	2	0	0
'542	0	1	1
'543	0	1	1
'545	1	1	0
'546	1	1	0
'548	0	2	1
'549	0	2	1

表9−2は、各種シリアル ポートおよびそれらのモードについての参照先を示しています。

表9−2. シリアル ポート参照先

シリアル ポート	モード	参照セクション . . .
スタンダード	−	セクション9.2スタンダード シリアル ポート インターフェイス(9−3ページ)
バッファド	自動バッファリング	セクション9.3バッファド シリアル ポート(BSP)インタフェース(9−32ページ)
	スタンダード(非バッファ)	セクション9.2スタンダード シリアル ポート インタフェース(9−3ページ)
TDM	TDM	セクション9.4時分割多重(TDM)シリアル ポート インターフェイス(9−55ページ)
	スタンダード(非TDM)	セクション9.2スタンダード シリアル ポート インターフェイス(9−3ページ)

9.2 シリアル ポート インターフェイス

’54x各種デバイスは、様々な柔軟性の高いシリアル ポート インターフェイスを備えています。これらのシリアル ポート インターフェイスは、コーデック、シリアル アナログ/デジタル(A/D)コンバータやその他のシリアル システム等のシリアル デバイスとの通信を全二重(双方向)で行えます。シリアル ポート インターフェイスの信号は、多くの業界標準コーデックおよび他のシリアル デバイスと互換性があります。シリアル ポートは、マルチ プロセシング アプリケーションにおけるプロセッサ間の相互通信にも使用できます(時分割多重(TDM)シリアル ポートは特にマルチ プロセシングに最適化されています)。

’54x各種デバイスには3種類のシリアル ポート インターフェイスがあります。基本のスタンダード シリアル ポート インターフェイスは、’541、’545、’546の各デバイスに実装されています。TDMシリアル ポート インターフェイスは、’542、’543、’548、’549デバイスに実装されています。’542、’543、’545、’546、’548、’549デバイスは、バッファド シリアル ポート(BSP)を持ち、自動バッファ機能を実現しているのでシリアル データ転送の処理にかかるCPUオーバーヘッドを大幅に減少できます。’54x各種デバイスに含まれる機能については、表9−1を参照してください。

BSPは、自動バッファ モードまたはスタンダード モードのいずれかで動作します。スタンダード(非バッファ)モードで動作するときは、BSPは本セクションで説明する基本のスタンダード シリアル ポート(特殊なケースを除く)と同様に機能します。TDMシリアル ポートは、TDMモードまたはスタンダード モードのいずれかで動作します。スタンダード(非TDM)モードの場合は、本セクションで説明する基本のスタンダード シリアル ポートと同様に機能します。

BSPではスタンダード モードにいくつかの改善された機能があります。これらの機能は自動バッファ モードでのBSP動作とともにセクション9.3バッファド シリアル ポート(BSP)インターフェイスで説明しています(9−32ページ)。したがって、’542、’543、’545、’546、’548、’549デバイスを使用するときは、セクション9.3を参照してください。TDMモードでのTDMシリアル ポートの動作については、セクション9.4時分割多重(TDM)シリアル ポート インターフェイスを参照してください(9−55ページ)。BSPおよびTDMシリアル ポートは、リセット時にスタンダード シリアル ポート モードに初期化されます。

すべての'54xのシリアル ポートの送受信の処理はダブルバッファを使用するので、8ビットまたは16ビットのデータ パケットで連続的な通信ストリームが可能になります。連続モードにより、いったん処理が始まると、最大パケット周波数送信時にそれ以降のフレーム同期パルス(FSRまたはFSX)が不要になります。シリアル ポートはスタティック設計なので任意の低クロック周波数で機能します。スタンダード シリアル ポートの最大動作周波数はCLKOUTの1/4(25nsで10Mbits/s、20nsで12.5Mbits/s)の周波数で、内部シリアル ポート クロックを使用したときに実現できます。BSPの最大動作周波数は、CLKOUTです。シリアル ポートではリセット時に、内部シリアル ポート クロックがオフになり、これによりデバイスが低消費電力モードで実行できます。

9.2.1 シリアル ポート インターフェイス レジスタ

シリアル ポートは、3種類のメモリ マップド レジスタ(SPC、DXR、DRR)および別の2種類のレジスタ(RSR、XSR)によって動作します。RSR、XSRは直接アクセスできませんがダブルバッファ機能の実現に必要になります。表9−3はこれらのレジスタを示しています。

表9−3. シリアル ポート レジスタ

アドレス	レジスタ	説明
†	DRR	データ受信レジスタ
†	DXR	データ送信レジスタ
†	SPC	シリアル ポート制御レジスタ
—	RSR	データ受信シフト レジスタ
—	XSR	データ送信シフト レジスタ

† セクション8.1ペリフェラル メモリ マップド レジスタを参照

- データ受信レジスタ(DRR)。16ビットのメモリ マップドのデータ受信レジスタ(DRR)は、RSRから入力されデータ バスに読み出されるシリアル データを保持します。リセット時に、DRRはクリアされます。
- データ送信レジスタ(DXR)。16ビットのメモリ マップドのデータ送信レジスタ(DXR)は、データ バスからXSRにロードするシリアル データを保持します。リセット時に、DXRはクリアされます。
- シリアル ポート制御レジスタ(SPC)。16ビットのメモリ マップドのシリアル ポート制御レジスタ(SPC)は、シリアル ポートのモード制御およびステータス ビットを保持します。

- **データ受信シフト レジスタ**(RSR)。**16ビットのデータ受信シフト レジスタ**(RSR)**は、シリアル データ受信**(DR)**ピンから入力されたデータを保持して、**DRR**に転送を行います。**
- **データ送信シフト レジスタ**(XSR)。**16ビットのデータ送信シフト レジスタ**(XSR)**には、**DXR**からデータが転送され、シリアル データ送信**(DX)**ピンに転送されるデータを保持します。**

通常のシリアル ポート動作の際に、シリアル ポートに送信するデータはプログラム実行によってDXR**にライトされます。その内容は自動的にシリアル ポート ロジックによって読み取られ、転送が始まると送出されます。シリアル ポートに受信されたデータは、シリアル ポート ロジックによって**DRR**に自動的にロードされ、プログラム実行により受信データとして取得できます。**

通常のシリアル ポート動作を行うときは、プログラムはDXR**へのライトおよび**DRR**からのリードの他に、メモリ マップド シリアル ポート レジスタで他の処理を必要とする場合があります。**

SP**では、シリアル ポートがリセット状態か否かに関わらず、**DXR**および**DRR**のリード/ライトが可能ですが、**BSP**では、これらのレジスタへのアクセスが制限されます。**DRR**は**BSP**がリセットのときのみライト可能で、**DRR**のリードおよび**DXR**のライトは自動バッファがディセーブルのときのみ可能です**(9−39**ページ、サブセクション**9.3.2**自動バッファリング ユニット**(ABU)**の動作を参照**)。**DXR**のリードは、常に**(BSP**に関わらず**)**可能です。**

ただし、SP**および**BSP**の両方において、通常の動作中にこれらのレジスタをリード/ライトする場合は注意しなければなりません。前述のように、**DRR**は、データ受信時にシリアル ポート ロジックにより自動的に書き込まれてしまいます。したがって、その書き込みの後に続いて、**DRR**をリードする前にシリアルポートが受信を行うと、**DRR**が再び書き込まれてしまい、その前に書き込まれた値を得られなくなる場合があります。**DXR**にライトするときは、前にライトしたデータがまだ送信されていない場合は、その内容を上書きしてしまい、元のデータが失われる恐れがあるので注意が必要です。**

一方、DXR**および**DRR**は、シリアル ポートで使用しなければ汎用の記憶空間としても利用できます。これらのレジスタを汎用記憶空間として使用する場合は、シリアル ポート転送を不正に発生するので、外部入力ピンを**OFF**にする(プルアップまたはプルダウンのいずれか適切な方法)か、またはシリアル ポートをリセットにして、シリアル ポートの送受信セクションをディセーブルにする必要があります。**

9.2.2 シリアル ポート インターフェイス動作

このサブセクションでは、基本のスタンダード シリアル ポート インターフェイスの動作について説明します。TDMおよびBSPシリアル ポートをスタンダード モードに設定したときも、スタンダード シリアル ポート インターフェイスと同じ動作になります。表9−4は、シリアル ポート動作時に使用されるピンを示しています。図9−1は、デバイス0からデバイス1への単一方向転送のために接続される'54xシリアル ポートのピンを示しています。データ転送のためにシリアル ポート送信部から受信部へ接続するには3つの信号が必要です。転送シリアル データ信号(DX)は実際のデータを送ります。転送フレーム同期信号(FSX)は転送を開始して(パケットの始まりで)、転送クロック信号(CLKX)はビット転送のクロック信号を発生します。受信デバイスの対応するピンは、それぞれDR、FSR、CLKRです。

表9−4. シリアル ポート ピン

ピン	説明
CLKR	受信クロック信号
CLKX	送信クロック信号
DR	受信シリアル データ信号
DX	送信シリアル データ信号
FSR	受信フレーム同期信号
FSX	送信フレーム同期信号

図9−1. 単方向シリアル ポート転送

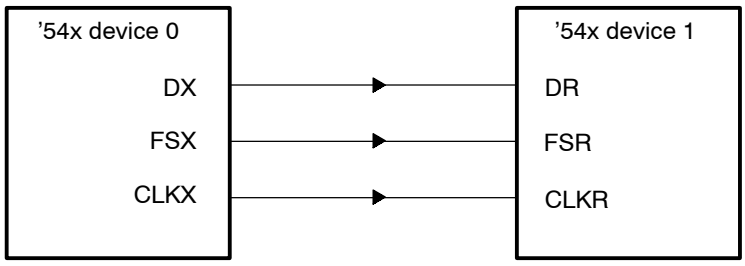


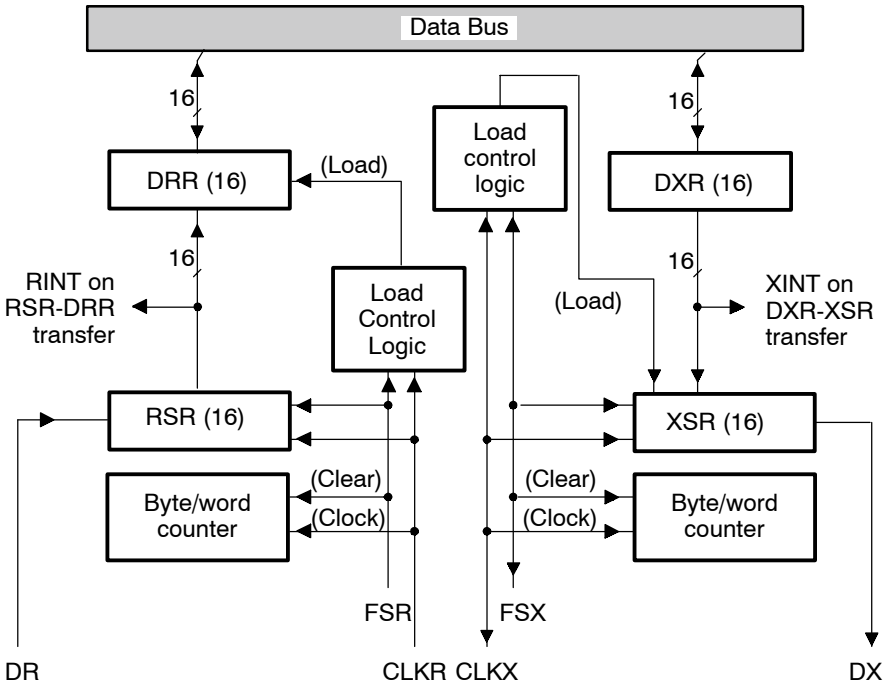
図9−2は、ピンおよびレジスタがシリアル ポートのロジックでどのように設定されるか、およびダブル バッファがどのように実現されているかを示しています。

送信データをDXRにライトし、受信されたデータをDRRからリードします。送信は、DXRにデータをライトすることで始まり、XSRが空になったとき(DXピン上にドライブされる最後のワードがシリアル送信されたとき)、そのデータをXSRにコピーします。XSRはデータをDXピンにシフトさせるので、DXRからXSRへのコピーが完了した後に、DXRへの別のライトが可能になります。

送信の過程で、DXRからXSRへのコピーが完了すると、SPCレジスタの送信レディ(XRDY)ビットが0から1に変更されます。この0から1への変更はシリアル ポート送信割り込み(XINT)を発生し、DXRを再ロードできる状態であることを示します。'54xの割り込みについては、6−26ページ、セクション6.10割り込みを参照してください。

このプロセスは、受信部でも同様です。DRピンからのデータがRSRにシフトされ、DRRにコピーされてそこからリードできます。RSRからDRRへのコピーが完了すると、SPCの受信レディ(RRDY)ビットが0から1へ変化します。この0から1への変更は、シリアル ポート受信割り込み(RINT)を発生します。したがって、シリアル ポートはDXRまたはDRRでデータ転送しているときに、別の送受信を実行できるダブルバッファ構成になっています。転送タイミングは、バースト モードのとき、フレーム同期パルスに同期します(詳しくは、9−17ページ、サブセクション9.2.4バースト モード送受信を参照してください)。

図9−2. シリアル ポート インターフェイスのブロック図



9.2.3 シリアル ポート インターフェイスの設定

SPCには、シリアル ポートの動作を設定する制御ビットがあります。図9−3はSPCビットフィールドを示しており、表9−5にはその説明があります。SPCの7つのビットはリード専用で、残りの9ビットはリード/ライト用です。

図9−3. シリアル ポート制御(SPC)レジスタ図

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Free	Soft	RSRFULL	$\overline{\text{XSREMPTY}}$	XRDY	RRDY	IN1	IN0	$\overline{\text{RRST}}$	$\overline{\text{XRST}}$	TXM	MCM	FSM	FO	DLB	Res
R/W	R/W	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R

注： R=リード、W=ライト

表9−5. シリアル ポート制御(SPC)レジスタのビット概要

ビット	名称	リセット値	ファンクション
15	Free	0	このビットはSoftビットとともに使用して、高級言語デバッガ使用時にブレークポイントに達した時のシリアル ポート クロックの状態を決定します。シリアル ポート クロックの設定については、9−17ページの表9−6を参照してください。 Free = 0 Softビットはエミュレーション モードを選択 Free = 1 Softビットにかかわらずシリアル ポート クロックはフリー ラン
14	Soft	0	このビットはFreeビットとともに使用して、高級言語デバッガ使用時にブレークポイントに達した時のシリアル ポート クロックの状態を決定します。Freeビットが0にクリアされたとき、Softビットではエミュレーション モードを選択できます。シリアル ポート クロックの設定については、9−17ページの表9−6を参照してください。 Soft = 0 シリアル ポート クロックが直ちに停止して、あらゆる転送が中止 Soft = 1 現在の転送が完了した後にクロックが停止
13	RSRFULL	0	受信シフト レジスタ フル(Receive Shift Register Full)。このビットは、受信部がオーバーランしたかどうかを示します。オーバーランは、RSRがフルになり、RSRからDRRへの直前の転送以後DRRがリードされない時に起こります。SPでFSM=1のときは、FSRでフレーム同期パルスが発生すると、RSRFULL=1になります。SPでFSM=0のとき、またBSPのときでは、FSRパルスを待たずに、上記の2つの条件(RSRがフルでDRRがリードされない時)だけでRSRFULL=1になります。 RSRFULL = 0 DRRのリード、受信部のリセット($\overline{\text{RRST}}$ビットを0)、デバイスのリセットの3つのイベントのうち、いずれかが行われるとRSRFULLビットが0にクリアされます。 RSRFULL = 1 ポートをオーバーランとして認識します。RSRFULL=1のとき、受信部が停止してDRRのリードを待ち、DRに送られるデータはなくなります。SPでは、RSRのデータが保存されます。BSPでは、RSRの内容が失われます。

表9−5. シリアル ポート制御(SPC)レジスタのビット概要(続き)

ビット	名称	リセット値	ファンクション
12	$\overline{\text{XSREMP-TY}}$	0	**送信シフト レジスタ エンプティ**(Transmit Shift Register Empty)**。このビットは、送信部がアンダーフローになったかどうかを示します。アンダーフローは、**XSR**が空で最後の**DXR−XSR**転送以降、**DXR**がロードされていないときに発生します。** $\overline{\text{XSREMPTY}}$ = 0 **アンダーフローの発生、送信部のリセット(**$\overline{\text{XRST}}$**ビットを**0**)、デバイスのリセットの**3**つのイベントのうち、いずれかが行われると、**$\overline{\text{XSREMPTY}}$**ビットが**0**にクリアされます。** $\overline{\text{XSREMPTY}}$ = 1 **SPでは、DXRへのライトの結果、**$\overline{\text{XSREMPTY}}$**がインアクティブ(1にセット)になります。BSPでは、DXRがロードされ、FSXパルスが発生した後に**$\overline{\text{XSREMPTY}}$**がインアクティブになります。**
11	XRDY	1	**送信レディ**(Transmit Ready)**。XRDYビットの**0**から**1**への変化は、DXRの内容が**XSR**にコピーされ、DXRに新たなデータ ワードをロードできる状態であることを示します。この時に送信割り込み**(XINT)**が発生します。このビットは、シリアル ポート割り込みを使う代わりに、ソフトウェアでポーリングもできます。SPでは、XRDYはDXRへのライトの結果発生し、BSPではDXRがロードされてFSXパルスが発生した後にのみXDRYが発生します。リセット時またはシリアル ポート送信部のリセット時に(**$\overline{\text{XRST}}$**=0)、XRDYビットは**1**にセットされます。**
10	RRDY	0	**受信レディ**(Receive Ready)**。RRDYビットの**0**から**1**への変化は、RSRの内容が**DRRにコピーされデータがリード可能なことを示します。受信割り込み**(RINT)**はこの変化時に発生します。このビットは、シリアル ポート割り込みを使う代わりに、ソフトウェアでもポーリングできます。リセット時またはシリアル ポート受信部のリセット時に(**$\overline{\text{RRST}}$**=0)、RRDYビットが**0**にクリアされます。**
9	IN1	x	**IN1。このビットにより、CLKXピンをビット入力ピンとして使用できます。**IN1**は、デバイスの**CLKX**ピンの現在のレベルを反映します。**CLKX**のレベルが切り替わるとき、新しい**CLKX**レベルが**SPC**に示される前に**0.5**から**1.5CLKOUT**サイクルの遅延があります。**
8	IN0	x	**IN0。このビットにより、CLKRピンをビット入力として使用できます。**IN0**は、デバイスの**CLKR**ピンの現在のレベルを反映します。**CLKR**のレベルが切り替わるとき、新しい**CLKR**レベルが**SPC**に示される前に**0.5**から**1.5CLKOUT**サイクルの遅延があります。**
7	$\overline{\text{RRST}}$	0	**受信リセット**(Receive Reset)**。この信号は、受信部のリセット、イネーブルを決定します。**0**を**$\overline{\text{RRST}}$**ビットにライトすると、受信部の動作が停止します。** $\overline{\text{RRST}}$ = 0 **シリアル ポート受信部をリセットします。**$\overline{\text{RRST}}$**に**0**をライトすると**RSRFULL**および**RRDY**ビットを**0**にクリアします。** $\overline{\text{RRST}}$ = 1 **シリアル ポート受信部をイネーブルにします。**

表9−5. シリアル ポート制御(SPC)レジスタのビット概要(続き)

ビット	名称	リセット値	ファンクション
6	$\overline{\text{XRST}}$	0	送信部リセット(Transmitter Reset)。この信号を使って、送信部のリセットとイネーブルを決定します。$\overline{\text{XRST}}$ビットに0をライトすると、送信部の動作が停止します。XRDYビットが0のとき、$\overline{\text{XRST}}$に0をライトすることで送信割り込み(XINT)が発生できます。
			$\overline{\text{XRST}}$ = 0　シリアル ポート送信部をリセットします。$\overline{\text{XRST}}$に0をライトすることで、$\overline{\text{XSREMPTY}}$ビットが0にクリアされXRDYビットが1にセットされます。
			$\overline{\text{XRST}}$ = 1　シリアル ポート送信部をイネーブルにします。
5	TXM	0	送信モード。このビットはFSXピンを入力(TXM=0)または出力(TXM=1)として設定します。
			TXM = 0　外部フレーム同期。フレーム同期パルスがFSXピンに供給されるまで、送信部がアイドル状態になります。
			TXM = 1　内部フレーム同期。データがDXRからXSRに転送されデータ転送を開始すると、フレーム同期パルスが内部的に発生します。内部的に発生したフレーム信号は、CLKXと同期します。
4	MCM	0	クロック モード。このビットは、CLKXのクロック ソースを指定します。
			MCM = 0　CLKXをCLKXピンから取ります。
			MCM = 1　CLKXをオンチップ クロック ソースによりドライブします。SPおよびBSPのスタンダード モードでは、このオンチップ クロック ソースはCLKOUTの1/4の周波数になります。BSPには、CLKOUTと別のクロック周波数を発生させるためのオプションがあります。この機能については、9−32ページ、セクション9.3バッファド シリアル ポート(BSP)インターフェイスを参照してください。MCM=1でDLB=1のとき、CLKR信号も内部ソースによって供給されます。
3	FSM	0	フレーム同期モード。このビットは、シリアル ポート動作で初期フレーム同期パルス(FSXおよびFSR)の後にフレーム同期パルスが必要かどうかを指定します。フレーム同期信号については、9−6ページ、サブセクション9.2.2シリアル ポート インターフェイス動作を参照してください。
			FSM = 0　連続モード。フレーム同期パルスは、初期フレーム同期パルスの後には不要ですが無視されず、シリアル転送中に不適切なタイミングのフレーム同期が現れるとエラーを発生します。例外的な条件下のシリアル ポート動作については、9−26ページ、サブセクション9.2.6シリアル ポート インタフェースの例外条件を参照してください。
			FSM = 1　バースト モード。フレーム同期パルスは、各ワードの送受信のためにFSX/FSR上で必要です。

表9−5. シリアル ポート制御(SPC)レジスタのビット概要(続き)

ビット	名称	リセット値	ファンクション
2	FO	0	**フォーマット。このビットは、シリアル ポート送受信部のワード長を指定します。** FO = 0 **データは、16ビット ワードとして送受信されます。** FO = 1 **データは8ビット バイトとして、MSBから転送されます。BSPでは、10ビットおよび12ビットの転送が可能です。この機能については、9−32ページ、セクション9.3バッファド シリアル ポート(BSP)インターフェイスを参照してください。**
1	DLB	0	**ディジタル ループバック モード。このビットを使って、シリアル ポートをデジィタル ループバック モードにできます。** DLB = 0 **ディジタル ループバック モードをディセーブルにします。DR、FSR、CLKR信号はそれぞれのデバイス ピンから取られます。** DLB = 1 **ディジタル ループバック モードをイネーブルにします。DR、FSR信号はマルチプレクサを通してそれぞれDX、FSXに接続されます(9−12ページ図9−4(a)(b)参照)。さらに、MCM=1のときCLKRはCLKXによってドライブされます。DLB=1でMCM=0のとき、CLKRはデバイスのCLKRピンから取られます。この設定により、CLKXおよびCLKRを外部的に結合させて共通の外部クロック ソースによって供給できます。CLKRのロジック図は、9−12ページ、図9−4(c)を参照してください。DLBモードでは、FSXおよびDX信号がデバイス ピンに現れますが、FSRおよびDRは現れないことにも注意してください。TXMビットの指定に応じて、内部および外部のいずれのFSX信号もDLBモードで使用できます。**
0	Res	0	**予約。シリアル ポートでは常に0としてリードされます。このビットは、TDMシリアル ポートで機能します(9−55ページ、セクション9.4時分割多重(TDM)シリアル ポート インターフェイスを参照)。**

予約ビット

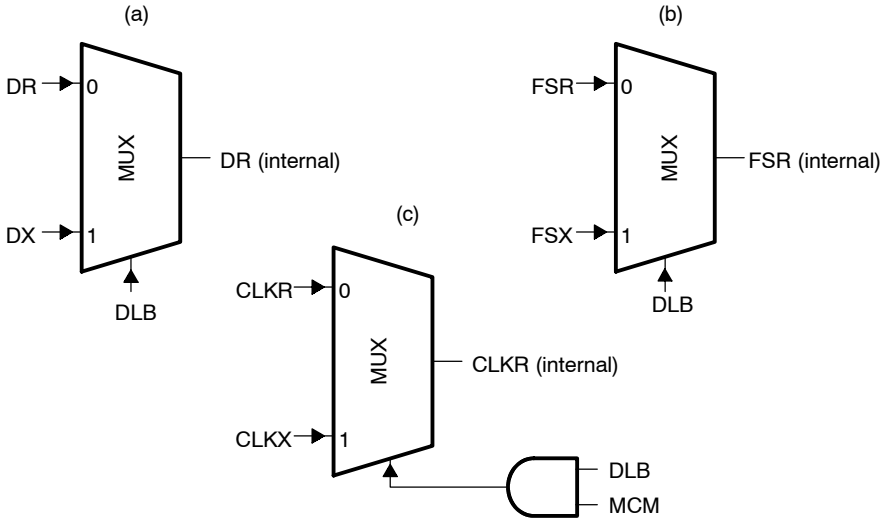
ビット0は予約ビットで常に0としてリードされます。ただしTDMシリアル ポートでは使います(9−55ページ、セクション9.4時分割多重(TDM)シリアル ポート インターフェイスを参照)。

DLBビット

DLB(ビット1)はディジタル ループバック モードを選択でき、単一の'54xデバイスでシリアル ポート コードのテストが可能になります。DLB=1のとき、DR、FSRがマルチプレクサを通してそれぞれDX、FSXに接続されます(図9−4参照)。

ループバック モードのときは、オンチップ シリアル ポート クロック発生が選択されている(MCM=1)とき、CLKRはCLKXによってドライブされますが、MCM=0の場合はCLKRは外部CLKR信号によってドライブされます。これによりディジタル ループバック モードにおいて外部シリアル ポート クロック発生機能を使用できます。DLB=0の場合は、DR、FSR、CLKRがそれぞれのピンから取られ、通常の動作をします。

図9−4. 受信部の信号マルチプレクサ



FOビット

FO(ビット2)は、データの送信を16ビット ワード(FO=0)とするか、8ビット バイト(FO=1)とするかを指定します。後者の場合、DXRにライトされる下位バイトのみが送信され、DRRからリードされるデータの下位バイトが受信されたものになります。16ビット ワード全体を8ビット モードで送信するには、 DXRにライトされる上位8ビットが無視されるので、値を適切にシフトして、DXRへ2回のライトをする必要があります 。同様に、16ビット ワード全体を8ビット モードで受信するには、DRRから2回のリードをし、適切に値をシフトする必要があります。SPでは、8ビット受信ではDRRの上位8ビットは不確定になります。BSPではDRRの未使用ビットは符号拡張されます。さらにBSPでは、10ビットと12ビット ワードの転送を実現して柔軟性を高めています。この機能については、9−32ページ、セクション9.3バッファド シリアル ポート(BSP)インターフェイスを参照してください。

FSMビット

FSM(ビット3)は、連続するシリアル ポート送信にフレーム同期パルスが必要かどうか指定します。FSM=1のとき、FSXがTXMに応じて外部か内部に発生しても、すべての転送にフレーム同期が必要です。通常、送信と送信の間にシリアル ポートがインアクティブになる期間があることから、このモードはバースト モードと呼ばれます。

シリアル ポート送信を行う周期はパケット周波数と呼ばれ、データ パケットは8、10、12、16ビット長が可能です。したがってパケット周波数が上がると、パケットから次のパケットの時間に相当するシリアル ポート クロック サイクル数が、転送されるビットの数と等しくなり、その時パケット周波数が最大になります。連続的に複数の転送を最大速度で送信する場合は、フレーム同期は基本的に冗長になります。実際にバースト モードでFSXが出力として設定され(TXM=1)最大パケット周波数のときにだけ、フレーム同期は冗長になります。FSXが入力(TXM=0)の場合は、送信のためにフレーム同期が必要です。

FSM=0は連続モードを選択しますが、この場合各転送の間、DXRへのライト(送信時)またはDRRからのリード(受信時)が実行される限り、必要なのは初期フレーム同期パルスのみとなります。FSM=0のとき、フレーム同期パルスは不要ですが無視されず、不適切なタイミングのフレーム同期はシリアル転送時にエラーを引き起こすことがあります。バースト モードと連続モードのタイミングについては、サブセクション9.2.4、9.2.5、9.2.6を参照してください。

MCMビット

シリアル ポート クロック ソースはMCM(ビット4)によって設定されます。MCM=0のとき、CLKXは入力として設定されるので、外部クロックを受け入れます。MCM=1のときは、CLKXは出力として設定され内部クロックによってドライブします。スタンダード モードのSPとBSP動作では、このオンチップ クロックの周波数はCLKOUTの1/4になります。BSPには、CLKOUTと異なるクロック周波数を発生するためのオプションがあります。この機能については、9−32ページ、セクション9.3バッファド シリアル ポート(BSP)インターフェイスを参照してください。CLKRピンは、常に入力として設定されます。

TXMビット

送信フレーム同期パルス ソースは、TXM(ビット5)によって設定されます。MCMのように、TXM=1のときFSXは出力として設定され、送信の開始ごとにパルスを生成します。TXM=0のとき、FSXが入力として設定され外部フレーム同期信号を取り込めます。FSRピンは常に入力として設定されます。

$\overline{\text{XRST}}$および$\overline{\text{RRST}}$ビット

シリアル ポート送受信部は、$\overline{\text{XRST}}$(ビット6)および$\overline{\text{RRST}}$(ビット7)によりリセットされます。これらの信号はロー アクティブなので、$\overline{\text{XRST}}$=$\overline{\text{RRST}}$=0ならシリアル ポートはリセット状態になります。シリアル ポートをリセットおよび再設定するには、SPCへの2回のライトが必要です。

- 最初のSPCへのライトは次のようになります。
  - 0を$\overline{\text{XRST}}$および$\overline{\text{RRST}}$ビットにライト
  - 必要な設定を残りのビットにライト
- 2回目のSPCへのライトは次のようになります。
  - 1を$\overline{\text{XRST}}$および$\overline{\text{RRST}}$ビットにライト
  - 必要な設定を残りのビットにライト

2回目のライトによって、シリアル ポートはリセット状態ではなくなります。送受信部は必要に応じて個別にリセットできます。0を$\overline{\text{XRST}}$または$\overline{\text{RRST}}$にライトすると、対応するシリアル ポートのセクションの活動が停止します。これにより切り替えが最小化され、デバイスを低消費電力で動作できます。$\overline{\text{XRST}}$=$\overline{\text{RRST}}$=MCM=0のとき、CLKXはそれ以上出力としてドライブしないので、必要な電力はさらに減らせます。

IDLE2およびIDLE3モードでは、SPの動作は'54xの他の部分と同様に停止します。一方BSPの場合は、外部シリアル ポート クロックを使用中であれば、IDLE2/3が実行された後でも動作は続きます。これにより、IDLE2/3で電力節約を実現でき、必要に応じて不可欠なシリアル ポート機能は保持したままにできます(BSP動作については、9−32ページ、セクション9.3バッファド シリアル ポート(BSP)インターフェイスを参照してください)。

また、SPでは、シリアル ポートをいつでもリセット状態から解除できます。ただし、リセット終了のタイミングに応じて、フレーム同期パルスが遅れる場合があります。BSPの場合、外部フレーム同期による送受信では、設定に最低1CLKOUTサイクルに加えて1/2シリアル ポート クロック サイクルが、スタンダード モードでFSXがアクティブになるより先に必要になります。自動バッファ モードでは、追加の設定が必要です(BSPの初期化タイミング条件については、9−32ページ、セクション9.3バッファド シリアル ポート(BSP)インターフェイスを参照してください)。

IN0およびIN1ビット

IN0(ビット8)とIN1(ビット9)により、CLKRとCLKXピンをビット入力として使用できます。IN0とIN1は、CLKRとCLKXピンの現在の状態を表しています。これらのピンのデータは、SPCレジスタのリードにより取得できます。これは、BIT、BITF、BITT、CMPM命令を使って実現できます。CLKR/CLKX切り替えから新しいCLKR/CLKX値がSPCレジスタで使用可能になるまでに0.5から1.5CLKOUTサイクルの遅延があります。シリアル ポートがリセットのときでも、IN0およびIN1をビット入力として使用でき、DRRとDXRを汎用レジスタとして使用できます。

RRDYおよびXRDYビット

SPCレジスタのビット10−13はリード専用ステータス ビットで、シリアル ポート動作の各種の状態を表します。シリアル ポートのリード/ライトは、RRDY(ビット10)とXRDY(ビット11)をポーリングすること、またはそれらが発生する割り込みを使うことにより、同期できます。RRDYビットの0から1への変化は、RSRの内容がDRRにコピーされ、受信されたデータをリードできることを示します。受信割り込み(RINT)がこの変化の際に発生します。

XRDYビットの0から1への変化は、DXRの内容がXSRにコピーされ、DXRに新しいデータワードをロードできることを示しています。送信割り込み(XINT)はこの変化の際に発生します。ソフトウェアでXRDYおよびRRDYをポーリングすることで、シリアル ポート割り込みの代用(ポーリングまたは割り込みは併用も可能です)または補足をすることができます。外部FSXを用いる時は、SPではDXRロードの結果としてXSRに直接ロードでき、BSPではFSXが現れるまでXSRはロードされません。

$\overline{\text{XSREMPTY}}$ビット

$\overline{\text{XSREMPTY}}$(ビット12)は、送信部がアンダーフローになったかどうかを示します。$\overline{\text{XSREMPTY}}$はロー アクティブなので、$\overline{\text{XSREMPTY}}$=0のときアンダーフローになったことを示します。

次の3つの条件のいずれもが、$\overline{\text{XSREMPTY}}$をアクティブにします($\overline{\text{XSREMPTY}}$=0)。

- 最後のDXRからXSRへの転送の後DXRがロードされておらず、XSRが空き(実際の$\overline{\text{XSREMPTY}}$の変化は最後のビットがXSRからシフトした後に起きます)
- 送信部がリセット($\overline{\text{XRST}}$=0)
- '54xデバイスがリセット($\overline{\text{RS}}$=0)

$\overline{\text{XSREMPTY}}$=0のとき、送信部が停止して、DXのドライブを次のフレーム同期パルスまで停止します(DXピンはハイ インピーダンス状態)。バースト モードではアンダーフローはエラー条件になりませんが、連続モードではなります(エラー条件については9−26ページ、サブセクション9.2.6シリアル ポート インターフェイスの例外条件を参照してください)。

次の条件は、$\overline{\text{XSREMPTY}}$をインアクティブにします($\overline{\text{XSREMPTY}}$=1)。

- SPでのDXRへのライト、またはBSPでDXRのライトに続いてFSXパルスが現れる場合(送信タイミングについては9−17ページ、サブセクション9.2.4バースト モード送受信動作を参照してください)。

RSRFULLビット

RSRFULL(ビット13)は、受信部がオーバーランしたかどうかを示します。RSRFULLはハイ アクティブなのでRSRFULL=1のときRSRがフルです。

バースト モードでは(FSM=1)、次の3項目のすべてが満たされた時、RSRFULLがアクティブになります(RSRFULL=1)。

- 直前のRSRからDRRへの転送以降、DRRをリードしていない
- RSRがフル
- フレーム同期パルスがFSR上に現れる

連続モードでは(FSM=0)、BSPの場合、RSRFULLの設定に最初の2つの条件が必要です。

- 直前のRSRからDRRへの転送以降、DRRをリードしていない
- RSRがフル

したがって連続モードの場合、BSPでは最後のビットを受信した後にRSRFULLが発生します。

RSRFULL=1のとき、受信部が停止してDRRがリードされるのを待ちます。さらにDRに送られるデータは失われます。SPでは、RSRのデータが保存され、BSPではRSRの内容が失われます。

次の3つの条件のいずれかで、RSRFULLがインアクティブになります(RSRFULL=0)。

- DRRをリード
- シリアル ポートをリセット($\overline{\text{RRST}}$=0)
- '54xデバイスをリセット($\overline{\text{RS}}$=0)

SOFTおよびFREEビット

Soft(ビット14)とFree(ビット15)は特別なエミュレーション ビットで、高級言語デバッガで実行中に、ブレークポイントまできた時のシリアル ポート クロックの状態を決めます。Freeビットを1にセットすると、ソフトウェアのブレークポイント上でクロックが動作し続けて(フリー ラン)データがシフトされたままになります。Free=1のとき、Softビットは無効になります。Freeビットがクリアされて0になると、Softビットが有効になります。Softビットがクリアされて0になると、クロックが即停止して送信を中止します。Softビットを1にセットして送信を続行すると、送信は転送が完了するまで続き、その後クロックが停止します。表9−6はこのオプションを示しています。

受信側も同様に機能します。即停止(Soft=Free=0)以外のオプションを選択した場合、受信部は動作し続けてオーバーフローのエラーが発生する可能性があります。これらのビットのデフォルト値は即停止です。

表9−6. シリアル ポート クロック設定

Free	Soft	シリアル ポート クロック設定
0	0	即停止、クロックが停止します(リセット値)。
0	1	送信部が現在のワードの完了後に停止します。受信部は影響を受けません。
1	X	フリー ラン

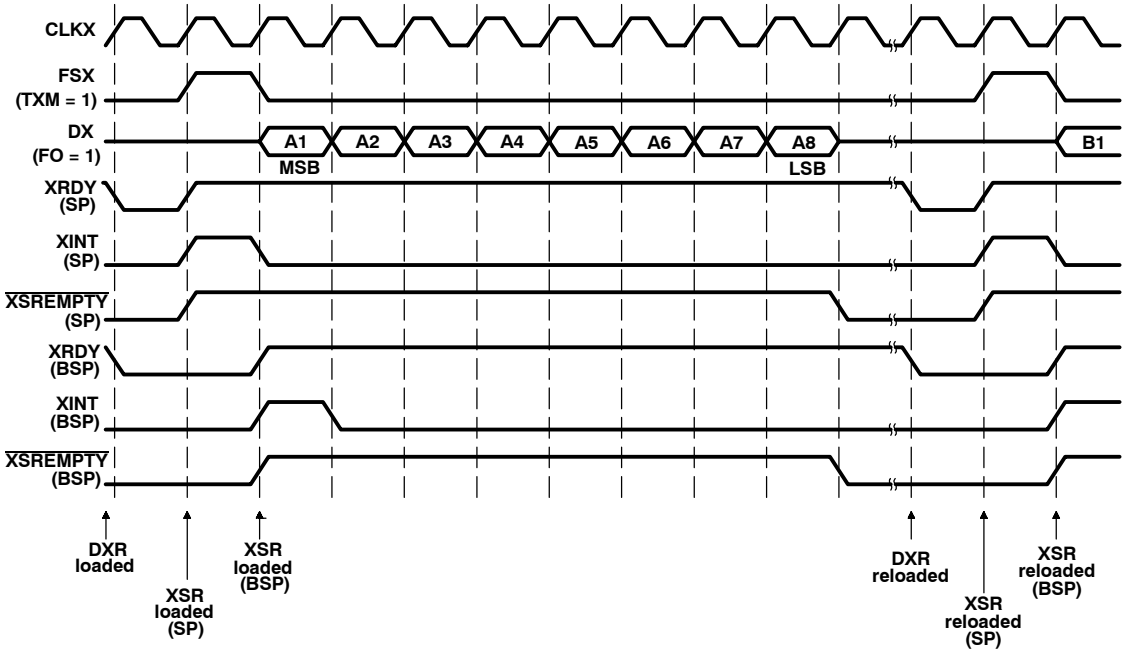
注： X=無効

9

9.2.4 バースト モード送受信の動作

バースト モードの動作では、パケットの送信と送信の間にシリアル ポートがインアクティブになる期間があります。データ パケットは、FSX上のフレーム同期パルスで示されます(図9−5参照)。送信デバイス上では、DXRへのライトよって転送が開始されます。次にDXRの値がXSRに転送され、FSX(TXMに応じて外部または内部で発生)でのフレーム同期パルスによって、XSRの値がシフトされDXピン上にドライブされます。SPでは、DXRがロードされた後のCLKXの2つ目の立ち上がりエッジでDXRからXSRへ転送され、BSPでは、この転送はFSXが外部の場合、FSXが発生するまで実行されません。FSXが内部の場合は、BSPでは、DXRのロードの直後に、DXRからXSRへの転送とFSXが発生します。SPおよびBSPの双方で、XSRがDXRの値によって一度ロードされると、XRDYがハイになり送信割り込み(XINT)を発生して$\overline{\text{XSREMPTY}}$が1にセットされます。

図9−5. バースト モードにおけるシリアル ポート送信の動作



SPとBSPの双方で、DXRからXSRへの転送は、XSRが空で直前のDXRからXSRへの転送以降にDXRがロードされている場合に行われます。古いDXRの内容がXSRに転送されるより前にDXRが再ロードされた場合、前のDXRの内容が上書きされます。DXRの上書きを回避したいなら、DXRはXRDY=1の場合のみにロードするようにしてください。これを確実にするためには、送信割り込みまたはXRDYのポーリングに対する応答に限り、DXRへライトを行うようにします。

フレーム同期が内部(TXM=1、FSXが出力)か、外部(TXM=0、FSXが入力)かによってタイミングが異なります。前者の場合、送信デバイスにおいて、フレーム同期パルスはDXRへのライトの直接的な結果として発生されるのに対し、後者の場合このような直接の効果がないため、タイミングが異なってきます。そのため、送信デバイスはDXRにライトした後、必ず外部のフレーム同期を待ちます。

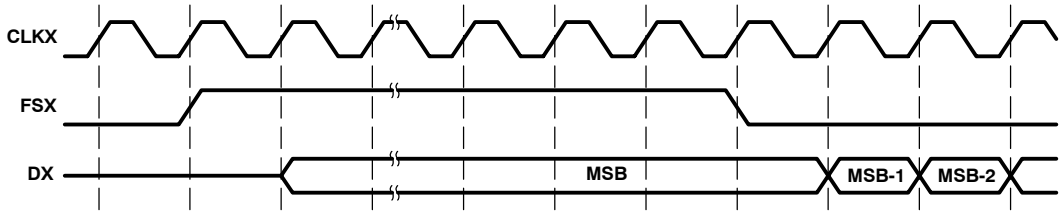
内部のフレーム同期パルスが選択されると(TXM=1)、フレーム同期パルスはDXRのライトに続くCLKXの2つ目の立ち上がりエッジで発生します。外部のフレーム同期の場合は、ここで説明するイベントは適切な時間のフレーム同期パルスの直後に発生します(シリアル ポート インターフェイスのタイミングについてはデータ シートを参照してください)。

FSXがハイになった後CLKXの次の立ち上がりエッジで、最初のデータ ビット(最初にMSB)がDXピン上にドライブされます。このため、フレーム同期パルスを内部で発生する場合(TXM=1)、DXRロード後におよそ2CLKXサイクルの遅延があり、その後にデータがそのライン上にドライブされます。フレーム同期が外部で発生する場合、データ送信はDXRロード後FSXパルスが発生するまで無制限に遅れます(以下このサブセクションで詳述)。フレーム同期の立ち下がりエッジとともに、残りのビットがシフトされます。すべてのビットが転送されるときは、DXはハイ インピーダンス状態になります。

各送信の終わりにおいて、XINTの発生時にDXRが再ロードされなかった場合は、$\overline{\text{XSREMPTY}}$がこの時点でアクティブ(ロー)となり、アンダーフローを示します。外部のフレーム同期を用いた時、$\overline{\text{XSREMPTY}}$がアクティブでフレーム同期パルスが発生する場合は、DXRの古いデータが送信されます。これについては、9−26ページ、サブセクション9.2.6シリアル ポート インターフェイスの例外条件を参照してください。

外部のフレーム同期が現れて、1CLKXサイクル以上ハイが続く場合、転送される最初のデータ ビットは可変的な長さを持つことがあります(図9−6参照)。内部のフレーム同期は、'54xのタイミングによって1CLKXサイクルの長さになります。

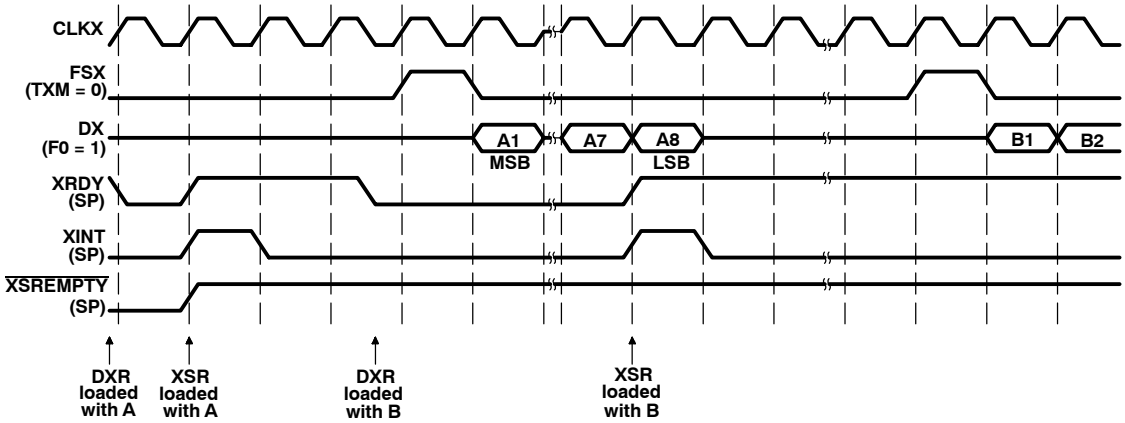
図9−6. 長い FSXパルスをともなうシリアル ポート送信



外部フレーム同期パルスを用いたシリアル ポート送信は、内部のフレーム同期を用いたシリアル ポート送信と同様になります。ただし、実際の転送は外部のフレーム同期が発生するまでは実行されません。しかし、外部フレーム同期が遅れる場合、ダブル バッファが満たされてフレーム同期が現れるまで停止します。

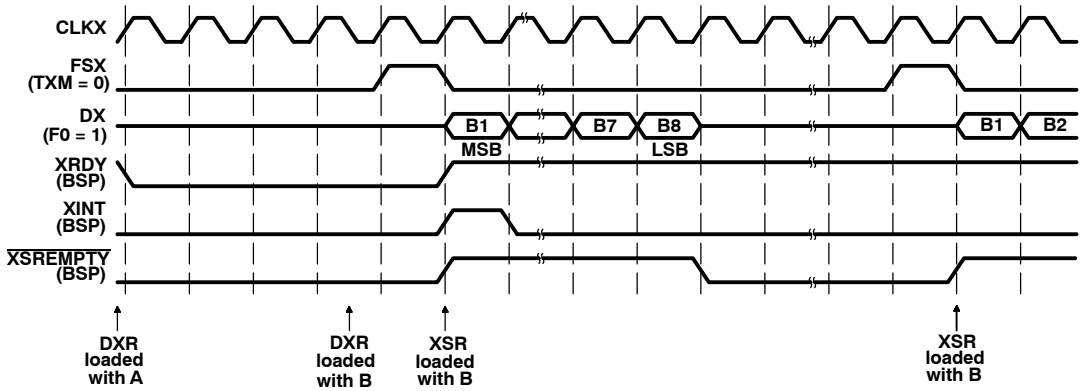
SPでは(図9−7)、フレーム同期が遅れて発生するとき、まず、AがDX上に送信されます。送信後にBがDXRからXSRへコピーされ、XINTが発生し、再び送信部は次のフレーム同期まで停止したままになります。そして、最終的にフレーム同期が発生すると、BがDX上に送信されます。BがDXRにロードされると、BのDXRからXSRへのコピーは、すぐには行われませんが、これはAが送信されておらずXINTが発生しないからです。次の遅延フレーム同期が発生する前に、DXRへライトすると、DXRにおけるBを上書きしてしまいます。

図9−7. **外部フレーム同期モードにおける遅延フレーム同期によるバースト モードのシリアル ポート送信動作**(SP)



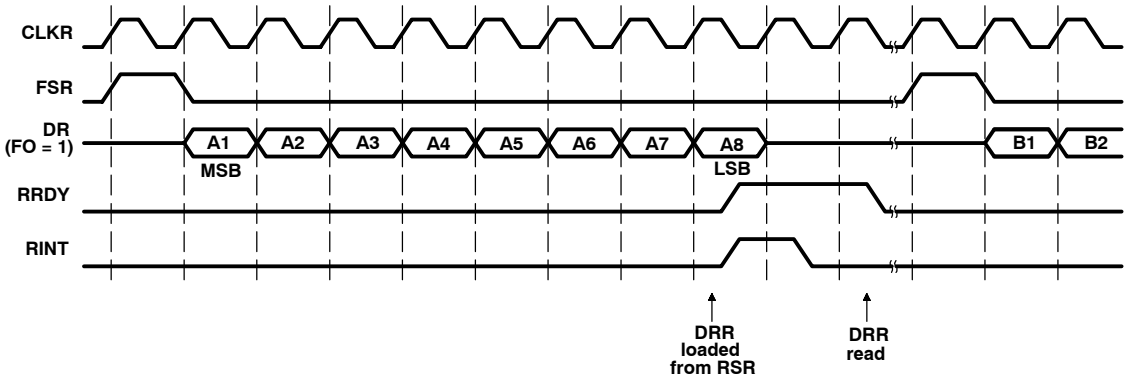
BSP**では(図**9−8)**、遅延フレーム同期が発生するときすでに**DXR**がA**によるロードのすぐ後の**B**の再ロードにより上書きされていると、**B**が**DX**に送信されます。その送信後、送信部は次のフレーム同期まで停止したままになります。フレーム同期が発生すると、**B**は再び**DX**に送信されます。**B**が**DXR**にロードされるとき、**B**の**DXR**から**XSR**へのコピーは直ぐにはされません。これは、**BSP**が送信開始のためにフレーム同期を必要とするからです。次の遅延フレーム同期発生の前に、**DXR**へのライトが起こると**DXR**の**B**が上書きされてしまいます。

図9−8. **外部フレーム同期モードにおける遅延フレーム同期によるバースト モードのシリアル ポート送信動作**(BSP)



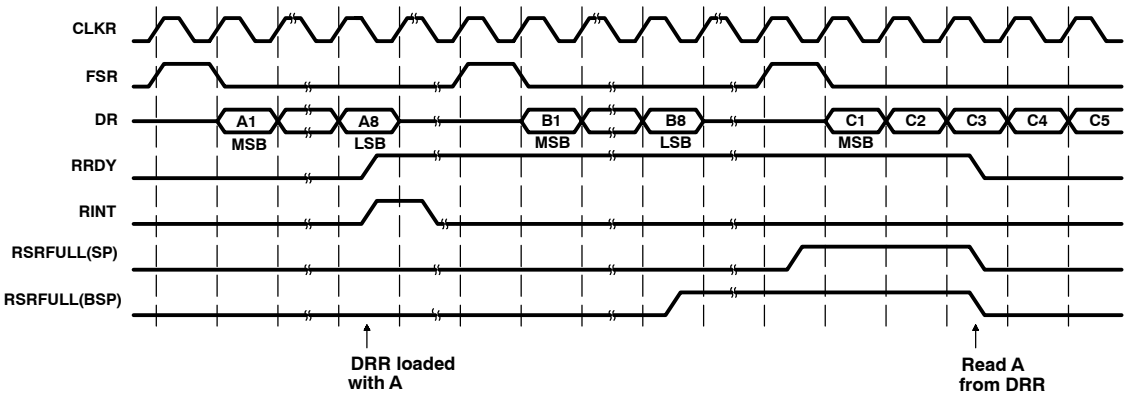
受信動作中に、RSRへのシフトは、フレーム同期がローになった後(図9−9)に、CLKRサイクルの立ち下がりエッジで始まります。そして、最後のデータ ビットが受信される際に、CLKRの立ち下がりエッジでRSRの内容がDRRへ転送され、RRDYがハイとなって受信割り込み(RINT)を発生します。

図9−9. バースト モードのシリアル ポート受信動作



前の受信からのDRRがリードされておらず、別のワードが受信される場合、DRRおよびRSRが両方ともフルなのでそれ以上ビットを受け取るとデータが失われます。この場合、RSRFULLビットがセットされその状態を示します。SPでは次のFSRによってこの状態になり、BSPでは最後のビットが受信される際にRSRFULLがCLKRの立ち下がりエッジでセットされます。図9−10は、SPおよびBSPの双方におけるRSRFULLのタイミングを示しています。

図9−10. バースト モードのシリアル ポート受信オーバーラン



送信アンダーフローとは異なり、オーバーラン(RSRFULL=1)は実際のエラー条件となります。DRRの内容はオーバーランでも保存されますが、そのために他の受信データが失われることがあります。

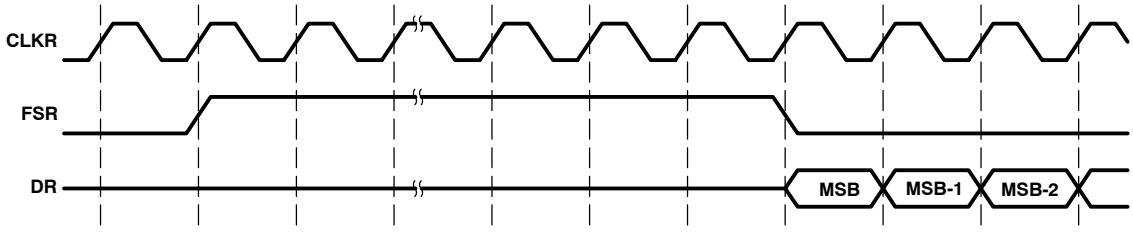
オーバーランは、SPとBSPで異なる方法で扱われます。SPでは、RSRの内容がオーバーラン時に保存されますが、RSRFULLはオーバーフロー受信後次のFSRが発生するまで1にセットされないので、次の入力データは通常RSRFULLがセットされると直ぐに失われ始めます。データ喪失は、RSRFULLをソフトウェアでポーリングし、RSRFULLが1にセットされたら直ちにDRRをリードすることによってのみ避けられます。これが可能なのは、通常CLKR周波数がCLKOUTより低い場合に限られます。というのは、RSRFULLはFSRが現れている間のCLKRの立ち下がりエッジでセットされ、データが次のCLKRの立ち上がりエッジで受信され始めるからです。したがって、データ喪失を避けるためにRSRFULLのポーリングとDRRのリードに利用できる時間は、CLKRサイクルの半分のみになります。

BSPでは、最後に受信された有効ビットでRSRFULLがセットされますが、RSRの内容はDRRには転送されません。したがって、RSRに転送が完了したワードは失われます。次のFSR発生の前にDRRをリード(RSRFULLをクリア)すれば、以降の転送を適切に受信できます。

オーバーラン、および受信中のフレーム同期の発生など様々なシリアル ポートの例外状態については、9−26ページ、サブセクション9.2.6シリアル ポート インターフェイスの例外状態を参照してください。

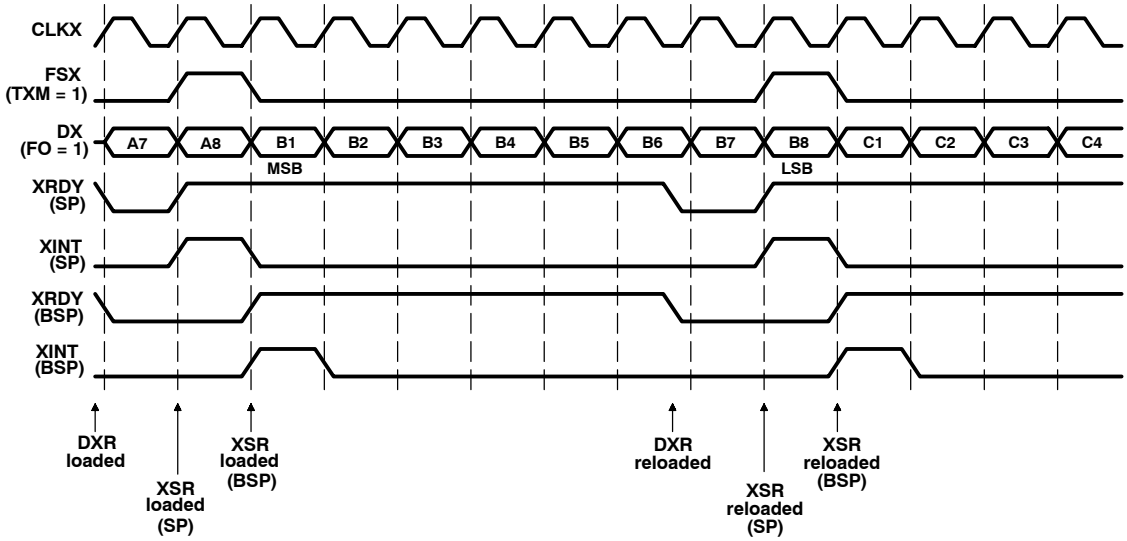
シリアル ポート受信部に、1CLKRサイクルより大幅に長いFSRパルスがある場合、データ受信のタイミングはロングFSXパルスがある場合と同様の影響を受けます。ただし、ロングFSRパルスを用いる場合、すべてのビットの受信は最初からFSRがローになるまで遅れます。図9−11は、ロングFSRパルスによるシリアル ポート受信動作を示しています。

図9−11. ロングFSRパルスをによるシリアル ポート受信



パケット送信周波数が増加すると、連続する転送のデータ パケット間のインアクティブ期間がゼロになります。これは、フレーム同期パルス間の最小期間に対応し、(FOにより8または16CLKX/Rサイクルに相当)、シリアル ポート動作における最大パケット周波数となります。最大パケット周波数での送信タイミングは図9−5を圧縮した図9−12のようにあらわせます。

図9−12. 最大パケット周波数におけるバースト モードのシリアル ポート送信



最大パケット周波数では、連続するパケットのデータ ビットは、ビット間がインアクティブにならずに連続的に転送されます。フレーム同期パルスは、前のパケットに送信された最終ビットと重なります。最大パケット周波数の受信タイミングも、同様になります(図9−13参照)。

図9−13. 最大パケット周波数におけるバースト モードのシリアル ポート受信

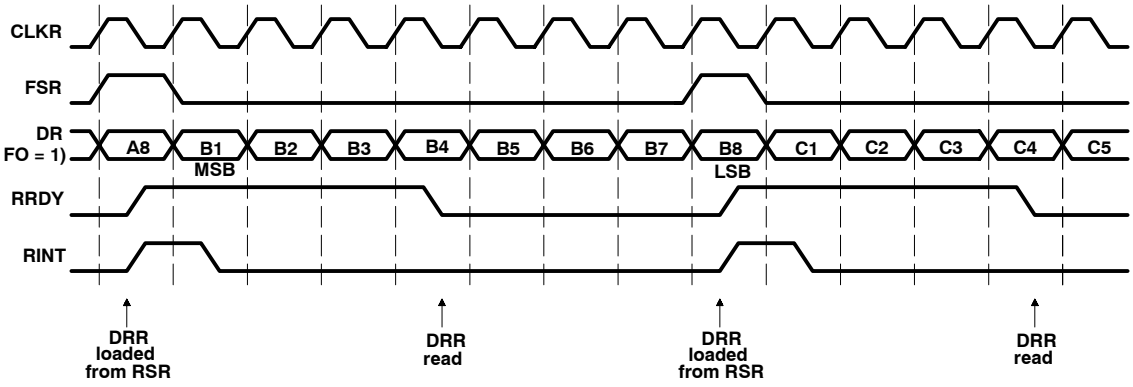


図9−12および図9−13に示すように、バースト モードで複数のデータ パケットを最大パケット周波数で転送する場合、パケットは一定の速度で送信され、シリアル ポート クロックは転送のために十分なタイミングを供給し、データの連続的なストリームを可能にします。このため、フレーム同期パルスは基本的に冗長になります。理論的には、マルチパケット転送を開始するには初期フレーム同期信号が必要です。'54xはこのようなシリアル ポートの動作を、SPCのFSMビットを0にクリアすることで選択できる連続モードとしてサポートします。連続モードのシリアル ポート動作については、サブセクション9.2.5連続モードの送受信動作を参照してください。

9.2.5 連続モードの送受信動作

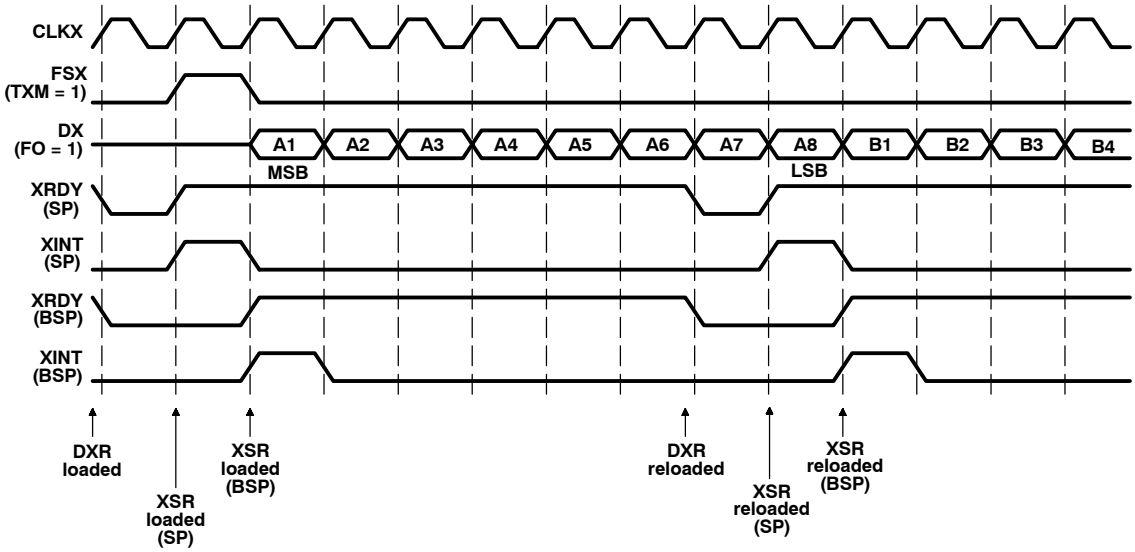
連続モードでは、最大パケット周波数での連続的なパケット転送には、初期パルス後のFSX/FSR上のフレーム同期は必要ありません。連続モードは、FSM=0にセットすることで選択できます。FSM=0のとき、フレーム同期パルスは不要ですが、無視されないので、不適切なタイミングのフレーム同期はシリアル転送でエラーを発生する場合があります。各種エラー状態でのシリアル ポート動作については、9−26ページ、サブセクション9.2.6シリアル ポート インターフェイスの例外条件を参照してください。

連続モードの送信では、ひとつのフレーム同期が最初のDXRのロードに続いて発生し、それ以降は発生しません。DXRが各送信ごとに再ロードされる限り、連続して転送されます。DXRの更新を失敗すると($\overline{\text{XSREMPTY}}$がアクティブになるなど)、バースト モードの時と同様にシリアル ポートは停止します。停止後にDXRが再ロードされる場合、デバイスは連続モードの送信を再開して、内部フレーム同期発生が選択されたものとして、一回のFSX信号を発生します。

連続モードにおける内部と外部のフレーム同期の違いは、バースト モードにおける違いと同様になります(9−17ページ、サブセクション9.2.4バースト モード送受信の動作を参照)。フレーム同期を外部で発生させる場合は(TXM=0)、DXRがロードされてから、フレーム同期パルスが現れることで連続モード送信が始まります。連続モード送信は、シリアル ポートを再設定またはリセットすることでのみ中断(バースト モードになる)できます(9−6ページ、サブセクション9.2.2シリアル ポート インターフェイス動作を参照)。送信時にFSMビットを変更、または送信を停止するだけでは、バースト モードに正しく切り替えられません。

図9−14に示す連続モードの送信タイミングは、図9−12に示したバースト モードの最大パケット周波数での送信と似ています。バースト モードとの主な相違点は、初期フレーム同期パルスの後にフレーム同期パルスがないことです。DXRが送信ごとに更新される限り、このモードは継続します。DXRへの上書きはバースト モードの処理と同様になり、最後に書き込まれたデータが送信されます。XSRの動作はバースト モードと同じようになります。新たな外部FSXパルスにより現在の送信は中止され、データ パケットひとつが失われます。そして新たな連続モード送信が開始されます。この動作については、9−26ページ、シリアル ポート インターフェイスの例外的な状態を参照してください。

図9−14. 連続モードのシリアル ポート送信



連続モードの受信は、送信の動作と同じようになります。FSR上での初期フレーム同期パルス以降は、フレーム同期パルスは必要ありません。このモードは、DRRが各転送でリードされる限り続きます。DRRを次の転送までにリードしない場合は、受信が停止して、RSRFULLがセットされオーバーランします。詳しくは、サブセクション9.2.6シリアル ポート インターフェイスの例外条件を参照してください。

連続モードでのオーバーランの影響は、SPとBSPでは異なります。SPではオーバーランが発生すると、DRRをリードすると、その後に次のワード/バイトの境界で連続モードとして再スタートします。新たなFSRパルスは必要ありません。BSPでは、連続モード受信はDRRをリードしてFSRが発生するまで復活しません。

連続モード受信は、シリアル ポートを再設定およびリセットすることでしか中止できません。受信時にFSMビットを変更または受信を停止するだけでは、バースト モードに正しく切り替えられません。図9−15は、連続モード受信タイミングを示しています。

図9−15. 連続モードのシリアル ポート受信

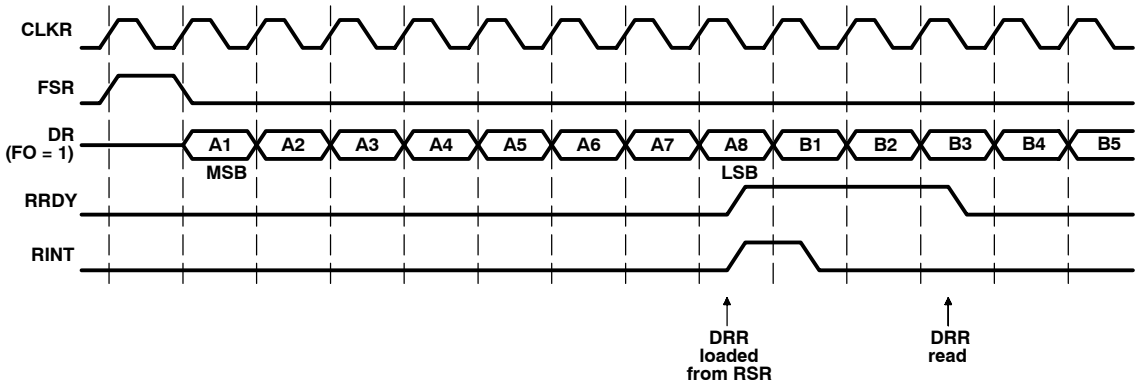


図9−15は、1フレーム同期パルスのみを表しています。それ以外は、図9−13と同様になります。転送時にFSRにパルスが現れると、受信動作が中断して、パケットの一つが失われ、新たな受信サイクルが始まります。これについては、9−6ページ、サブセクション9.2.2シリアル ポート インターフェイス動作、およびサブセクション9.2.6シリアル ポート インターフェイスの例外状態を参照してください。

9.2.6 シリアル ポート インターフェイスの例外状態

例外(またはエラー)状態は、シリアル ポートで発生する予期しないイベントによって生じます。このような状態には、オーバーラン、アンダーフロー、転送時のフレーム同期パルスなどの不適切な動作があります。シリアル ポートがエラー状態のときと、このようなエラーをどのように処理するかについて理解することは、シリアル ポートを効率的に使用する上で重要なことです。エラー状態はバースト モードと連続モードで多少異なるので、それぞれのケースを個別に説明します。

バースト モード

バースト モードでのエラー状態のひとつ(サブセクション9.2.2シリアル ポート インターフェイス動作を参照)は受信オーバーランで、RSRFULLフラグで示されます。このフラグは、デバイスが入力データをリードしない内にさらにデータが送られ続ける場合にセットされます。この状態が発生すると、プロセッサはDRRをリードするまでシリアル ポートの受信を停止します。したがって、それ以降に送られるデータは失われることになります。

SPとBSPでは、オーバーラン時の処理の仕方が異なります。SPでは、RSRの内容はオーバーラン時には保存されますが、オーバーフローの受信後に次のFSRが発生するまでRSRFULLは1にセットされないので、入力データは通常RSRFULLがセットされるとすぐに失われます。RSRFULLがソフトウェアでポーリングされ、RSRFULLが1にセットされた直後にDRRをリードする場合のみに、データ喪失は回避できます。

これが可能なのは、CLKR周波数がCLKOUT周波数より低い場合に限られます。というのは、RSRFULLがFSRが現れている間のCLKRの立ち下がりエッジでセットされ、データが次のCLKRの立ち上がりエッジで受信され始めるからです。したがって、RSRFULLのポーリングとDRRをリードしてデータ喪失を回避するために使用できる時間は、CLKRサイクルの半分のみになります。

BSPの場合は、RSRFULLは受信される最後の有効ビット上でセットされますが、RSRの内容はDRRに転送されないので、RSRの転送が完了したワードは失われます。次のFSRが発生する前にDRRをリードすれば(RSRFULLをクリア)、後に続く転送を適切に受信できます。

もうひとつの受信エラーは、フレーム同期が受信時(つまりデータがDRからRSRにシフトされる)に発生する場合に起こります。この状 態になると、実行中の受信が中断されて新しい受信が始まります。したがって、RSRにロードされていたデータは失われますが、DRRのデータは失われません(RSRからDRRへのコピーは行われません)。バースト モードでの通常の状態およびエラー状態におけるシリアル ポート受信動作については、図9−16がSPの場合を示し、図9−17はBSPの場合を示しています。

図9−16. SP受信部の機能動作(バースト モード)

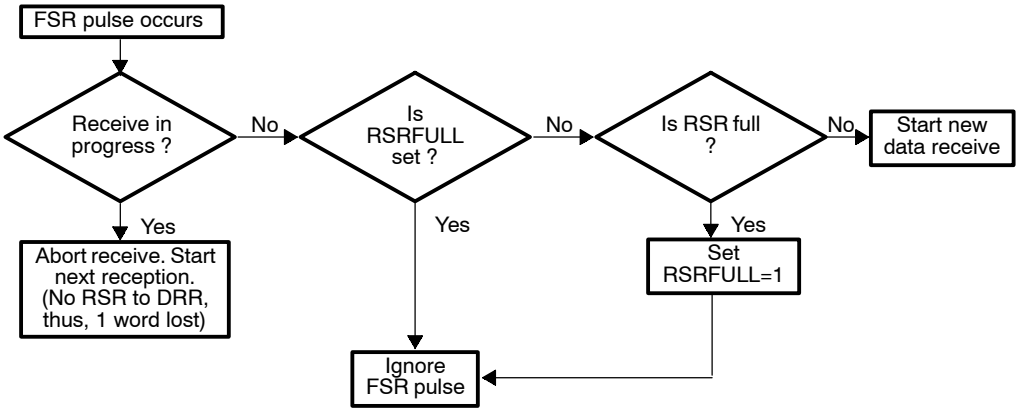
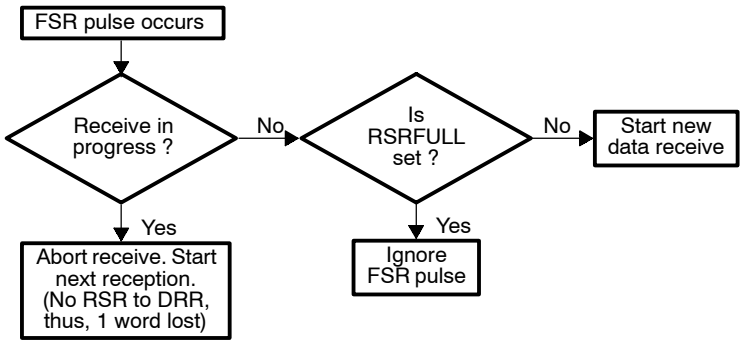
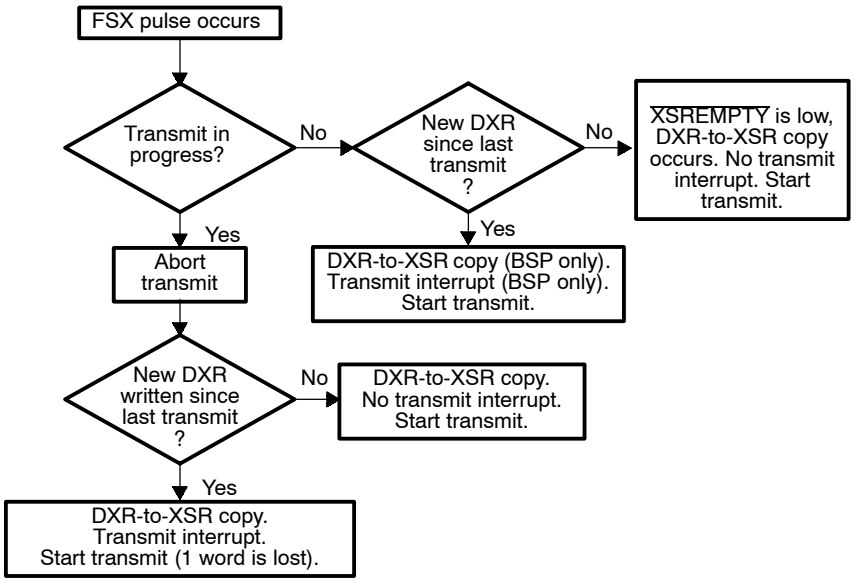


図9−17. BSP受信部の機能動作(バースト モード)



バースト モードにおいて、送信部が例外状態となるには、いくつかの理由があります。アンダーフロー(9−7ページ、サブセクション9.2.3シリアル ポート インターフェイスの構成を参照)はバースト モードで発生する例外状態のひとつですが、アンダーフローは通常エラーとしては考慮されません。送信されるデータにエラーを生じさせる例外状態は、フレーム同期パルスが転送時の不適切なタイミングで発生するときに起こります。フレーム同期パルス発生時に送信中(つまりXSRデータがDX上で駆動)である場合は、送信が中断され、XSRのデータが失われます。そして、フレーム同期パルス発生時にDXRにあるデータは、XSRに転送(DXRからXSRへコピー)され、送信されます。ただしこのケースでは、直前の送信以降にDXRがライトされた場合のみ、XINTが発生します。また、$\overline{\text{XSREMPTY}}$がアクティブでフレーム同期パルスが発生する場合、DXRの古いデータはシフトされます。図9−18は、エラー状態および通常状態におけるシリアル ポート送信動作を要約して示しています。送信中ではない時に、FSXが発生して、直前の送信以降にDXRが再ロードされるとき、BSPでのみ、この時点でDXRからXSRへのコピーおよび送信割り込みが生じます。SPでは、DXRが再ロードされた時点でこれら二つのイベントが生じます。

図9−18. SP/BSP送信部機能動作(バースト モード)

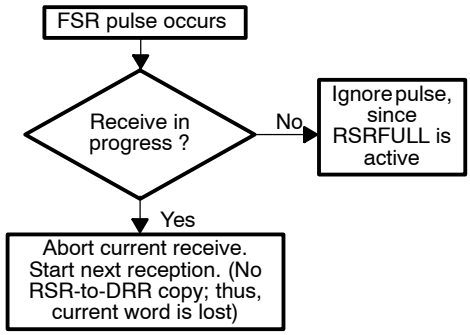


連続モード

連続モードでは、常にデータ転送が行われているので、エラーの条件も増えます。そこで連続モードでは、アンダーフロー($\overline{\text{XSREMPTY}}$=0)もデータが送信されないので、エラーとなります。オーバーラン(RSRFILL=1)もバースト モード時と同様、エラーとなります。連続モードではオーバーランおよびアンダーフローの双方がそれぞれシリアル ポート送受信部を停止させてしまいます(これらの状態については、9−7ページ、サブセクション9.2.3シリアル ポート インターフェイスの設定を参照してください)。幸い、アンダーフローおよびオーバーランのエラーはそれほど危険性はなく、多くの場合はDRRのリードまたはDXRへのライトによって修正できます。

連続モードでのオーバーランにより、SPとBSPはそれぞれ異なる影響を受けます。SPでは、DRRをリードしてRSRFULLをインアクティブにすると、連続モードの動作の復帰には、フレーム同期パルスは不要です。受信部はたとえそれが受信されるデータでなくても、たえず、転送ワードの境界を検出しています。したがって、RSRFULLフラグがDRRからのリードによってインアクティブになると、受信部は正しいビットからのリードを始めます。BSPでは、連続受信を再開するのにFSRパルスを必要とし、これにより受信を再開するとともに適切なビット整合を確立します。図9−19は、連続モードでの受信部機能動作を示しています。

図9−19. SP/BSP受信部機能動作(連続モード)

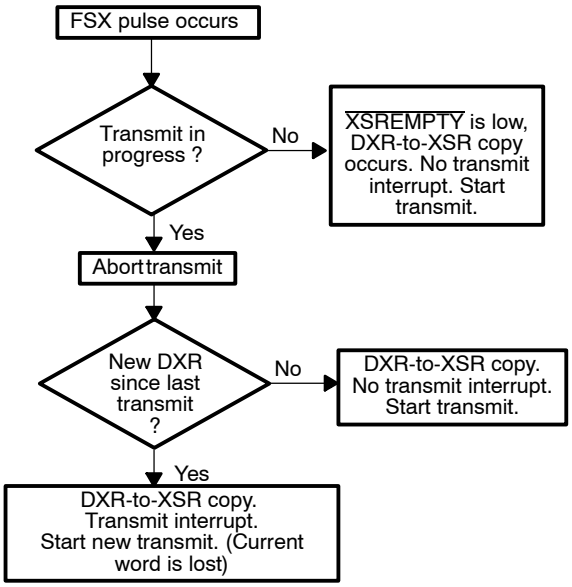


連続モードでの受信中に、フレーム同期パルスが発生すると、受信が中断され、データ パケットがひとつ失われます(フレーム同期パルスがRSRビット カウンタをリセットすることに起因します)。DR上のデータはRSRにシフトされ、再び最初のビットから受信が始まります。DRRのリードによりRSRFULLフラグがインアクティブになり、かつ次のワード境界の前までにフレーム同期が発生してしまう場合にも、受信中断状態となります。

別のエラー原因には、送信中の偶発的なフレーム同期の発生があります。連続モードにおける初期フレーム同期の後、それ以上のフレーム同期は必要なく、送信中に不適切なタイミングでフレーム同期パルスが発生すると、現在の転送(DX上でXSRデータをシリアルにドライブ)が中断され、XSRのデータが失われます。新たな送信サイクルが始まって、それ以降の送信ごとにDXRが更新される限り転送は継続します。図9−20は、連続モードの送信部機能動作を示しています。

連続モードで$\overline{\text{XSREMPTY}}$がアクティブになり、外部フレーム同期が現れる場合、前のDXRデータはバースト モードでの動作と同様に送信されます。

図9−20. SP/BSP送信部機能動作(連続モード)



9.2.7 シリアル ポート インターフェイス動作の例

スタンダード シリアル ポートの適切な動作として、例9−1および例9−2はその手順を示しています。この表は、シリアル ポートとCPU間の通常の入出力を割り込みを使って処理する方法を基本として述べています。これらの例は、'545ペリフェラル設定を参照しています。

例9−1. **シリアル ポート初期化ルーチン**

	動作	説明
1)	0038h(**または**0008h)**を**SPC**にライトすることでシリアル ポートをリセットおよび初期化します。**	**これにより、シリアル ポートの送受信部分をともにリセットにして、シリアル ポートを、内部の**FSX**および**CLKX**信号で動作するようにセットし、さらに各**16**ビット ワードの送受信毎に**FSX/FSR**を必要とするようにセットします。(別のデバイスが**FSX**と**CLKX**を供給している場合は代用できます。)**
2)	00C0h**を**IFR**にライトして待ち状態のシリアル ポート割り込みをクリアします。**	**初期化以前に発生した割り込みを解消します。**
3)	IMR**と**00C0h**のOR**をとることでシリアル ポート割り込みをイネーブルにします。**	**送受信双方の割り込みをイネーブルにします。送受信が互いに同期する場合は、**IMR**を**0080h**または**0040h**で論理和演算するか、同じ割り込みサービス ルーチン(**ISR**)により入出力動作を実行させることで、送信か受信のいずれかをイネーブルにします。**
4)	ST1**の**INTM**ビットをクリアして割り込みをグローバルでイネーブルにします。**	CPU**の応答のために、割り込みは必ずグローバルでイネーブルにされます。**
5)	SPC**に**00F8h(**または**00C8h)**をライトして、シリアル ポートを開始します。**	**これにより、シリアル ポートの送受信部分がリセットから復帰して、**1)**の状態で動作を開始します。**
6)	DXR**に最初のデータ値をライトします。(シリアル ポートが別のプロセッサのシリアル ポートに接続されそのプロセッサが**FSX**を生成するときは、必ず**DXR**に最初のデータ値をライトする前にハンドシェイクしなければなりません。)**	**これにより、**FSX**と**CLKX**を内部で生成してシリアル ポート送信動作を開始するか、外部から最初の**FSX**が現れるまでの、シリアル ポートの送信動作の準備を行います。**

例9−2. **シリアル ポート割り込みサービス ルーチン**

	動作	説明
1)	**スタック上に変更されるコンテキストを保存します。**	**割り込みコードのコンテキスト操作は必ずされなければいけません。**
2)	DRR**のリードと**DXR**へのライトのいずれかまたは両方を実行します。**DRR**からリードしたデータは予め決められたメモリ位置にライトされます。**DXR**にライトされるデータは、予め決められたメモリ位置からリードされます。**	**受信**ISR**で受信されたデータをリードし、新たな送信データを送信**ISR**でライトします。あるいは、**ISR**が送受信の両方に対して必要な場合は、その両方を実行します。**
3)	1)**で保存したコンテキストを復元します。**	**割り込みコードのコンテキスト操作は必ずされなければなりません。**
4)	RETE**によって**ISR**から復帰して、割り込みを再度イネーブルにします。**	CPU**が次の割り込みに応答できるように、割り込みを再度イネーブルにすることが必要です。**

9.3 バッファド シリアル ポート(BSP)インターフェイス

バッファド シリアル ポート(BSP)インターフェイスは、'54xスタンダード シリアル ポートと同様に機能する全二重、ダブル バッファド シリアル ポート インターフェイス、およびオート バッファリング ユニット(ABU)から構成されます(図9−21参照)。BSPのシリアル ポート セクションは、'54xスタンダード シリアル ポートを高機能化したものです。ABUはロジックの追加セクションで、シリアル ポート セクションをCPUに依存せず'54xの内部メモリに直接リード/ライトできるようにします。これにより、シリアル ポート転送でオーバーヘッドを最小限にして、データ転送速度を高速化します。

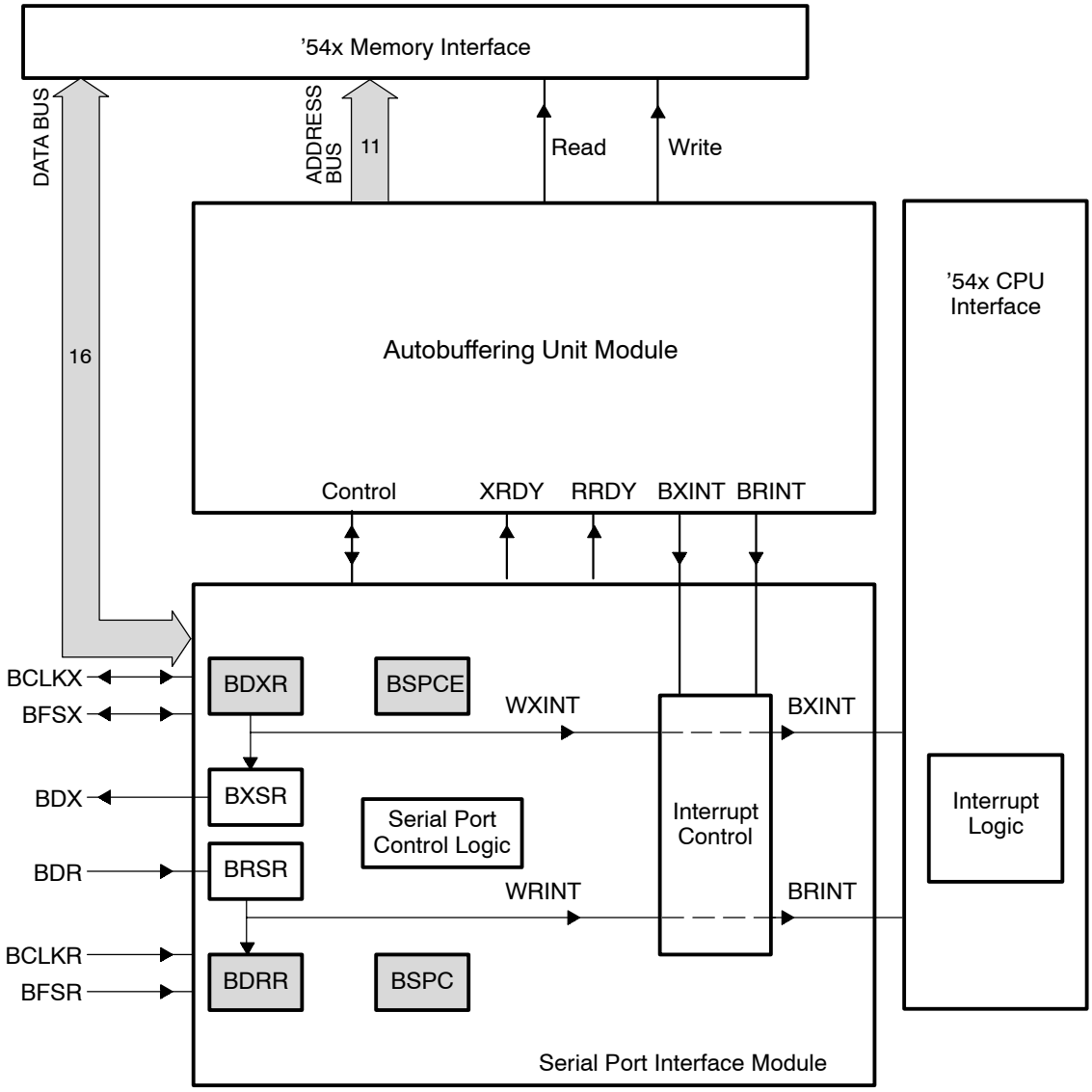
全二重BSPシリアル インターフェイスは、コーデック、A/Dコンバータなどのシリアル インターフェイスを持ったデバイスや、そのデバイスと最小限の外部ハードウェアで通信を実現します。ダブル バッファドBSPにより、8、10、12、16ビット データ パケットの連続的な通信ストリームの転送が可能になります。送受信のために、BSPはフレーム同期パルスおよびプログラマブル周波数シリアル クロックを使用することができます。フレーム同期およびクロック 信号の極性もプログラマブルです。最大動作周波数はCLKOUT(25nsで40M bit/s、30nsで50M bit/s)になります。BSP送信セクションにはパルス コード モジュレーション(PCM)モードが含まれ、PCMラインとの容易なインターフェイスを可能にします。スタンダード(非バッファ)モードにおけるBSP動作については、9−34ページ、サブセクション9.3.1を参照してください。

ABUには独立したサーキュラ アドレッシング レジスタのセットがあり、それぞれアドレス生成ユニットに対応します。送受信バッファのメモリは、'54x内部メモリの特定の2Kワード ブロック空間内にあります。CPUはこのメモリを汎用ストア空間として使用できますが、オート バッファリングを使用できる唯一のメモリ ブロックでもあります。

オート バッファリングを使用することで、ABU内のアドレス生成器を使ってワード転送が自動的にシリアル ポート セクションと'54xの内部メモリ間で直接行われます。2Kブロック内のバッファの長さおよび開始アドレスはプログラマブルで、バッファ エンプティ/フル割り込みをCPUに発生できます。バッファリングは自動ディセーブル機能を使って簡単に停止できます。ABU動作については、9−39ページ、サブセクション9.3.2を参照してください。

送信および受信セクションのために、BSPオート バッファリング機能を個別にイネーブルにできます。オート バッファリングをディセーブル(スタンダード モード)にするとき、シリアル ポート セクションによるデータ転送がスタンダードの'54xシリアル ポートと同様にソフトウェア制御によって行われます。このモードでは、ワードが送信/受信されるごとに発生するWXINT/WRINT割り込みが、送信割り込み(BXINT)/受信割り込み(BRINT)としてCPUに送られます。オート バッファリングがイネーブルのとき、BXINT/BRINT割り込みはバッファの半分が転送されたときにCPUに発生します。

図9−21. BSP**ブロック図**



前述のように、BSP**の動作は多くの点で**'54x**スタンダード シリアル ポートと似ています。**'54x**スタンダード シリアル ポートおよびスタンダード モードの**BSP**の動作については、**9−3**ページ、セクション**9.2**シリアル ポート インターフェイスを参照。スタンダード モードの**BSP**動作は、スタンダード シリアル ポート動作のスーパーセットなので、本セクションは最初にセクション**9.2**シリアル ポート インターフェイスの内容を念頭に置いた上でお読みください。

初期化、低消費電力モードなどBSP**動作システムについては、**9−49**ページ、サブセクション**9.3.3**を参照してください。

9.3.1 スタンダード モードにおけるBSP動作

スタンダード モードでのBSP動作については、9−3ページ、セクション9.2シリアル ポート インターフェイスを参照してください。このサブセクションでは、シリアル ポート動作とスタンダード モードのBSP動作の違いを概説して、BSPが備えているより高度な機能について説明します。BSPの高度な機能は、スタンダード モードおよびオート バッファリング モードの両方で使用できます。ABUについては、9−39ページ、サブセクション9.3.2を参照してください。このセクションでは、セクション9.2シリアル ポート インタフェースの説明が理解されていることを前提として説明します。

BSPは、独自の専用メモリ マップド データ送信、データ受信、およびシリアル ポート制御レジスタ(BDXR、BDRR、BSPC)を使用します。BSPは、高度化された機能の実現およびABU制御のために、別の制御レジスタであるBSP制御拡張レジスタ(BSPCE)も使用します。BDXR、BDRR、BSPCレジスタは、それぞれが対応するシリアル ポートのレジスタと同じように機能します(セクション9.2シリアル ポート インターフェイスを参照)。BSPの送受信シフト レジスタ(BXSR、BRSR)はソフトウェアでは、直接アクセスできません。シリアル ポートが使用されていない場合は、BDXRおよびBDRRレジスタを汎用レジスタとして使用できます。この場合、BFSRは非アクティブの状態にセットして受信動作が始まるのを防ぐべきです。ただし、送信および受信のためにオート バッファリングがイネーブルのときは、BDXRおよびBDRRへのプログラム アクセスは制限されます。ABUがディセーブルのときは、BDRRはリードのみ、BDXRはライトのみが可能です。BDRRへのライトは、BSPがリセットのときのみ可能になります。BDXRはいつでもリードできます。

バッファド シリアル ポート レジスタの概要は、表9−7に示します。ABUはいくつかの追加レジスタを使用しますが、これらについては9−39ページ、サブセクション9.3.2オート バッファリング ユニット(ABU)の動作で説明します。

表9−7. バッファ シリアル ポート レジスタ

アドレス	レジスタ	説明
†	BDRR	16ビットBSPデータ受信レジスタ
†	BDXR	16ビットBSPデータ送信レジスタ
†	BSPC	16ビットBSP制御レジスタ
†	BSPCE	16ビットBSP制御拡張レジスタ
—	BRSR	16ビットBSPデータ受信シフト レジスタ
—	BXSR	16ビットBSPデータ送信シフト レジスタ

† セクション8.1ペリフェラル メモリ マップド レジスタを参照

9.3.1.1 スタンダード モードにおけるシリアル ポートおよびBSP動作の相違

スタンダード モードにおけるシリアル ポートおよびBSP動作の違いについて詳しくは、9−3ページ、セクション9.2のスタンダード モードのシリアル ポート動作を参照してください。それらの相違は、主に境界動作によるものですが、一部のシステムではその相違が顕著です。相違点については、表9−8にまとめて示しています。

表9−8. シリアル ポートとスタンダード モードのBSPとの間の動作の相違

動作	シリアル ポート	BSP
RSRFULLのセット。	RSRがフルでFSRが発生したときRSRFULLがセットされます。ただし連続モードでは、RSRがフルになるとすぐRSRFULLがセットされます。	BRSRがフルになるとすぐRSRFULLがセットされます。
オーバーラン時のRSRのデータ保持。	RSRの内容がオーバーラン時に保持されます。	BRSRの内容がオーバーラン時に保持されません。
オーバーラン後の連続モード受信の再開。	DRRをリードするとすぐに受信が再開します(9−26ページ、サブセクション9.2.6シリアル ポート インターフェイスの例外的な状態を参照)。	BDRRをリードしてBFSRが発生するまで受信は再開しません。
8、10、または12ビット転送時にDRRで符号拡張。	no	yes
XSRロード、$\overline{\text{XSREMPTY}}$クリア、XRDY/XINT発生。	DXRがロードされるとき発生。	BDXRがロードされた後BFSXが発生するときに発生。
DXRおよびDRRへのプログラム アクセス。	DXRおよびDRRは、プログラムの制御下でいつでもリード/ライトできます。ただし、DRRのリード/ライトがシリアル ポートの受信実行時間に近い場合は注意が必要です。この場合DRRのリードによって、前にプログラムがライトしたものを得られない場合があります。またDXRのリライトは、ライトおよびFSXのタイミングにより、前にライトしたデータの喪失を招くことがあります(9−17ページ、サブセクション9.2.4バースト モード送受信動作を参照)。	ABUがディセーブルのとき、BDRRはリードのみ、BDXRはライトのみ可能です。BDRRはBSPがリセット時のみライト可能です。BDXRはいつでもリード可能です。これらのレジスタのリード/ライトについての注意点は、同様にシリアル ポートにも当てはまります。
最大シリアル ポート クロック速度	CLKOUT/4	CLKOUT

表9−8. シリアル ポートとスタンダード モードのBSPとの間の動作の相違(続き)

動作	シリアル ポート	BSP
初期化のタイミング条件	シリアル ポートでは、FSX/FSRに関係せずシリアル ポートはいつでもリセット解除できますが、フレーム同期の最中またはその後に$\overline{\text{XRST}}$/$\overline{\text{RRST}}$がhighになる場合は、フレーム同期は無視されます。	BSPでは、ある条件におけるシリアル ポートのリセット解除は、スタンダード モードではFSXのタイミングより1CLKOUTサイクル前、オート バッファリング モードでは6CLKOUTサイクル前にしなければなりません(9−49ページ、サブセクション9.3.3BSP動作のシステムについてを参照)。
IDLE2/3モードでの動作	no	yes(9−49ページ、サブセクション9.3.3BSP動作のシステムについてを参照)

9.3.1.2 高度化されたBSP機能

BSPの高度化された機能には、プログラマブルな速度のシリアル ポート クロックを発生する機能、クロックおよびフレーム同期信号の極性の正負を選択する機能、シリアル ポートの8ビット、16ビット転送に加えて10ビットおよび12ビット ワードの転送を実行する機能があります。さらに、BSPは特別な指定があるまでフレーム同期信号を無視するように指定する機能も実装しています。またPCMインターフェイスに特化した専用の動作モードも備えています。

BSPCEには、これらの高度化されたBSP機能およびABUを使用するための制御およびステータス ビットが含まれます。BSPCEの下位10ビットは高度化された機能の制御専用であり、上位6ビットはABUの制御に使用されます(これらについては9−39ページ、サブセクション9.3.2オート バッファリング ユニット(ABU)の動作を参照)。図9−22は、BSPCEのビット位置を示し、表9−9はBSPCEビットのファンクションを概説しています。BSPCEのリセット時の値は3です。これにより、スタンダード モードの動作がシリアル ポートと互換性を持つことになります。

図9−22. BSP制御拡張レジスタ(BSPCE)図—シリアル ポート制御ビット

15−10	9	8	7	6	5	4−0
ABU control	PCM	FIG	FE	CLKP	FSP	CLKDV
	R/W	R/W	R/W	R/W	R/W	R/W

注： R=Read(リード)、W=Write(ライト)

表9−9. BSP制御拡張レジスタ(BSPCE)ビット概要—シリアル ポート制御ビット

ビット	名称	リセット値	ファンクション
15−10	ABU制御	—	オート バッファリング ユニット制御のために予約されています(9−39ページ、サブセクション9.3.2自動バッファ ユニット(ABU)の動作を参照)。
9	PCM	0	パルス コード モジュレーション モード。この制御ビットは、シリアル ポートをパルス コード モジュレーション(PCM)モードにします。PCMモードは送信部のみに影響します。BDXRからBXSRへの転送は、PCMビット値によって影響されません。
			PCM = 0 パルス コード モジュレーション モードをディセーブル。
			PCM = 1 パルス コード モジュレーション モードをイネーブル。PCMモードでは、BDXRがその最上位(2^{15})ビットが0にリセットされている場合のみに送信されます。このビットが1にセットされている場合は、BDXRは送信されず、BDXは送信期間の際にハイ インピーダンスになります。
8	FIG	0	フレーム無視。この制御ビットは、外部フレームをともなう送信連続モードおよび受信連続モードのみで作動します。
			FIG = 0 全てのフレーム同期パルスが転送を再始動。
			FIG = 1 転送動作を開始する最初のフレーム パルス以降のフレーム同期パルスを無視。
7	FE	0	フォーマット拡張。FEビットはSPCレジスタのFOとともに(9−7ページ、サブセクション9.2.3シリアル ポート インターフェイスの構成を参照)、ワードの長さを指定します。FO FE=00のときフォーマットは16ビット ワードになり、FO FE=01のときフォーマットは10ビット ワード、FO FE=10のときフォーマットは8ビットになり、FO FE=11のときフォーマットは12ビットになります。8ビット、10ビット、12ビット ワードでは、受信されたワードは右詰めで符号ビットは16ビット ワードになるように拡張されます。送信ワードは必ず右詰めされます。ワード長の構成については、表9−10を参照してください。
6	CLKP	0	クロック極性。この制御ビットは、受信部および送信部のデータのサンプリング方法を指定します。
			CLKP = 0 データがBCLKRの立ち下がりエッジで受信部によってサンプリングされ、BCLKXの立ち上がりエッジで送信部によって送信されます。
			CLKP = 1 データがBCLKRの立ち上がりエッジで受信部によってサンプリングされ、BCLKXの立ち下がりエッジで送信部によって送信されます。

表9−9. BSP制御拡張レジスタ(BSPCE)ビット概要—シリアル ポート制御ビット(続き)

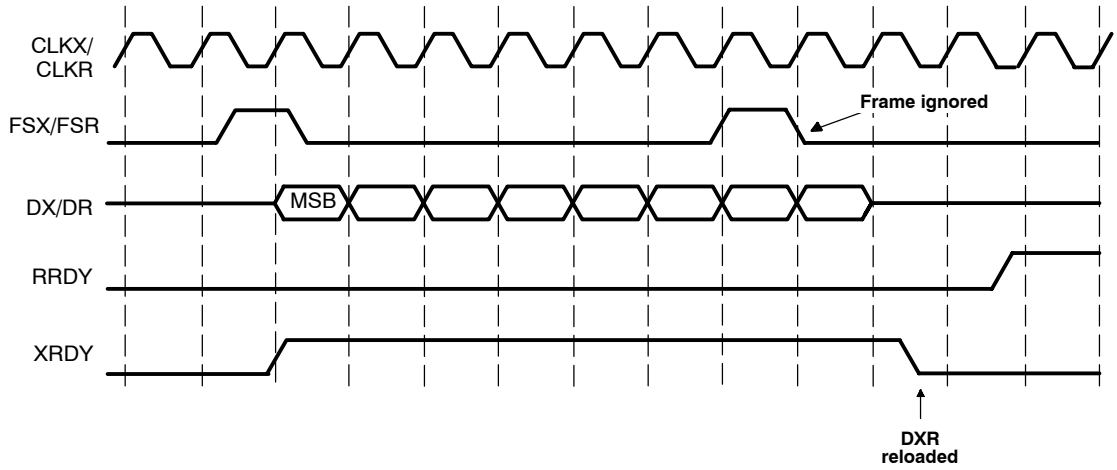
ビット	名称	リセット値	ファンクション
5	FSP	0	フレーム同期極性。この制御ビットは、フレーム同期パルス(BFSXおよびBFSR)がアクティブハイまたはローかを指定します。 FSP = 0 フレーム同期パルス(BFSXおよびBFSR)がアクティブ・ハイ。 FSP = 1 フレーム同期パルス(BFSXおよびBFSR)がアクティブ・ロー。
4−0	CLKDV	00011	内部送信クロック分周値。BSPCのMCMビットが1にセットされる場合、CLKXは内部ソースによりCLKOUTの1/(CLKDV+1)に等しい周波数になり、CLKDVの範囲は0−31です。CLKDVが奇数または0に等しいとき、CLKXのデューティ サイクルは50%です。CLKDVが偶数値(CLKDV=2p)の場合、CLKXハイおよびロー・ステートの長さはCLKPに依存します。CLKPが0のとき、ハイの長さはp+1サイクルになり、ローの長さはpサイクルになります。CLKPが1のとき、ハイの長さはpサイクルで、ローの長さはp+1サイクルになります。

表9−10. バッファド シリアル ポートのワード長構成

FO	FE	バッファド シリアル ポートのワード長構成
0	0	16ビット ワード送信および受信(リセット値)
0	1	10ビット ワード送信および受信
1	0	8ビット ワード送信および受信
1	1	12ビット ワード送信および受信

これらの高度化された機能により、様々な面でシリアル ポート インターフェイスがより高度な柔軟性を備えることになります。特にフレーム無視機能には、転送されるデータ パケットが16ビット フォーマットでない場合にそれを効率的に圧縮するメカニズムを実現する機能があります。この機能は、外部フレーム同期をともなう連続受信および連続送信とともに使用されます。FIG=0のとき、フレーム同期パルスが初期フレーム同期パルスの後に発生する場合に、転送が再度開始されます。FIG=1のときは、このフレーム同期は無視されます。たとえばFIGを1にセットすることにより、フレーム同期パルスが16ビット未満のビット(たとえば8ビットまたは10ビット)ごとに発生する状況でも連続16ビット転送を効率的に実現できます。例えばFIG=0の時、８ビットの転送に８ビット モードを使用するためには、８ビット バイト データのストアに2つの16ビット メモリが必要となります。FIG=1では、１６ビット モードで２つの８ビット バイト データを１６ビット１ワードにまとめて受信できます。FIGを使用する場合は、バッファ サイズの必要条件がオート バッファリング モードおよびスタンダード モードの双方で大幅に改善され、スタンダード モードでシリアル ポートを処理するのに必要なCPUサイクル オーバーヘッドも大幅に改善されます。図9−23は、BSPを16ビット フォーマットで構成しながら8ビット後にフレーム同期を持つ例を示しています。

図9−23. **外部フレームおよび**FIG=1**での送信連続モード**(16**ビット フォーマット**)



9.3.2 自動バッファリング ユニット(ABU)動作

ABU**の機能はスタンダード モードのシリアル ポート動作のスーパーセットなので、このサブセクションでは**9−3**ページのセクション**9.2**シリアル ポート インターフェイス、および** 9−34**ページのサブセクション**9.3.1**スタンダード モードにおける**BSP**動作の内容を前提にしています。また、自動バッファ モード動作のとき、**BSPC**および**BSPCE**のシリアル ポート制御ビットおよびステータス ビットは、スタンダード モードの場合と同じように機能します。**

ABU**は、シリアル ポート上に転送されたデータを**CPU**の干渉を受けない**'54x**内部メモリとの間で移動します。**

ABU**は、アドレス送信レジスタ**(AXR)**、ブロック サイズ送信レジスタ**(BKX)**、アドレス受信レジスタ**(ARR)**、ブロックサイズ受信レジスタ**(BKR)**、**BSPCE**の、**5**種類のメモリ マップド レジスタを利用します。表**9−11**は、これらのレジスタの概要を示しています。**

表9−11. 自動バッファリング ユニット レジスタ

アドレス	レジスタ	説明
†	BSPCE	16ビットBSP制御拡張レジスタ
†	AXR	11ビットBSPアドレス送信レジスタ(ABU)
†	BKX	11ビットBSP送信バッファ サイズ レジスタ(ABU)
†	ARR	11ビットBSPアドレス受信レジスタ(ABU)
†	BKR	11ビットBSP受信バッファ サイズ レジスタ(ABU)

† セクション8.1、ペリフェラル メモリ マップド レジスタを参照

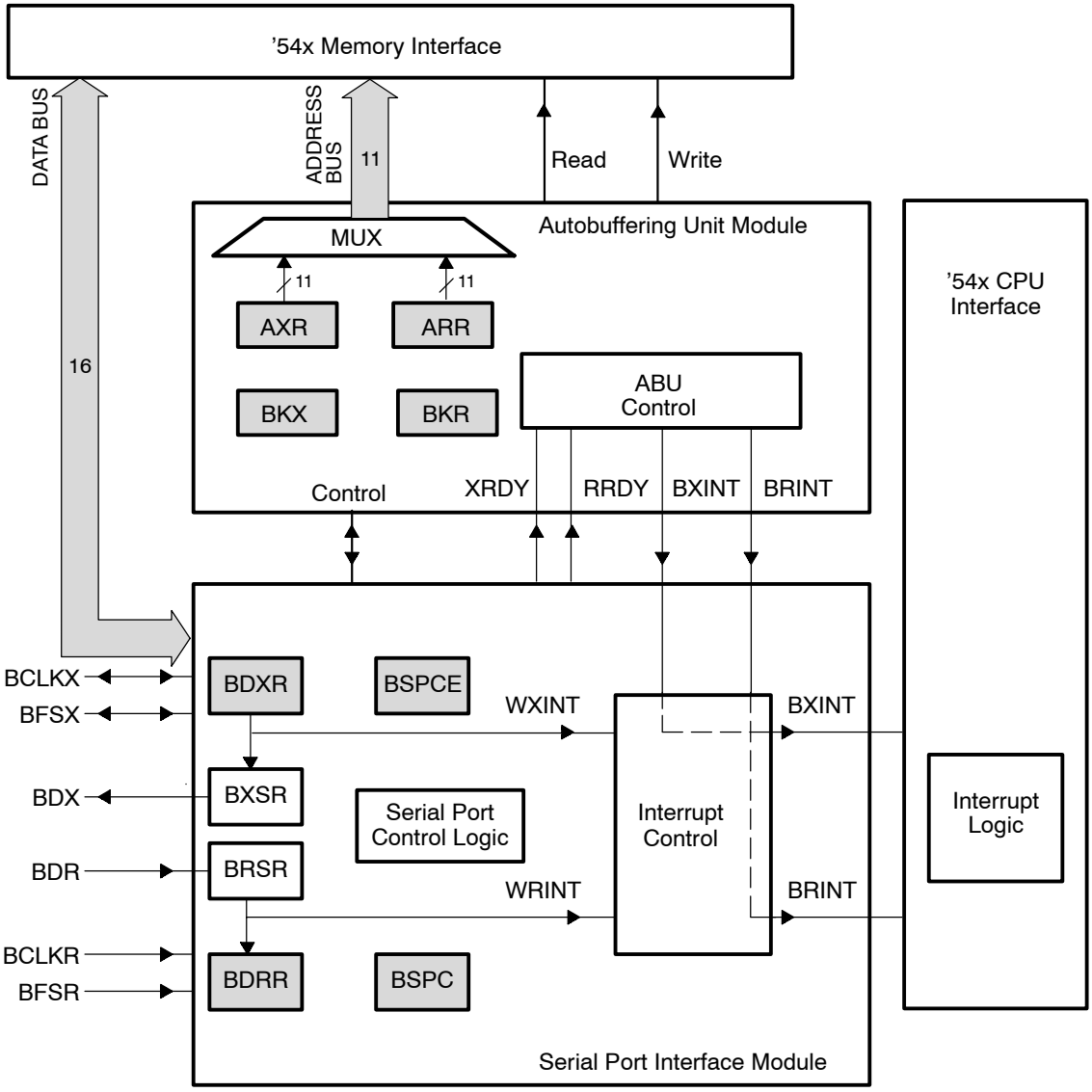
図9−24は、ABUのブロック図です。BSPCEにはABU動作を制御するビットが含まれますが、これについてはこのサブセクションで以降に説明します。AXR、BKX、ARR、BKR、およびこれらに関連するサーキュラ アドレッシング ロジックにより、自動バッファリング モードでワードにアクセスするためのアドレス生成を'54x内部メモリおよびBSPデータ送信レジスタ(BDXR)およびBSPデータ受信レジスタ(BDRR)間で転送できるようになります。アドレスおよびブロック サイズ レジスタ、およびサーキュラ アドレッシングについては、このサブセクションで以降に説明します。

11ビットのメモリ マップドAXR、BKX、ARR、BKRレジスタは、16ビット ワードとして読み取られ、上位5ビットがゼロとして読み取られ、11ビット レジスタの内容が下位11ビットに右ぞろえで位置調整されます。自動バッファリングが使用されない場合は、これらのレジスタを11ビット データの汎用記憶空間として使用できます。

ABUの送信および受信セクションは、個別にイネーブルにできます。いずれかのセクションをイネーブルにすると、ソフトウェアによってそれに対応するシリアル ポート データ レジスタ(BDXRまたはBDRR)へのアクセスは制限されます。BDRRのリードおよびBDXRへのライトは、ABUがディセーブルのときのみ実行できます。BDRRへのライトは、BSPがリセットのときのみ実行できます。BDXRのリードはいつでも実行できます。送信または受信いずれかの自動バッファリングがディセーブルのときは、そのセクションはスタンダード モードで動作して、そのABUは動作に影響しません。

ABUは、送信または受信バッファの半分または全部が完全に満たされるか空の場合に、CPU割り込みを発生する機能を実現します。この割り込みは、スタンダード モードの動作における送信および受信割り込み(自動バッファリング モードでは発生しない)の位置に入ります。このメカニズムには自動ディセーブル機能があり、バッファ中間位置またはバッファ最下位置の境界線を過ぎたときに、この機能を使用して自動バッファリングを自動的に終了できます。これらの機能については、このサブセクションの後半で説明します。

図9−24. ABU**ブロック図**



セクション9.2**シリアル ポート インターフェイスの解説にもあるように、バースト モードまたは連続モードを自動バッファリング機能とともに使用できます。ただし、自動バッファリング モードの特性により、内部フレーム同期をともなうバースト モードを選択する場合、各送信の開始時に**BSP**によって**FSX**が発生し、事実上は連続的な送信になります。**

自動バッファリングに使用される'54xの内部メモリは、2Kワード ブロックのデュアル アクセス メモリから成り立っており、これはデータ、プログラムのいずれかまたは両方として設定できます(他のデュアル アクセス メモリ ブロックと同様)。このメモリは、CPUが汎用記憶空間として使用できますが、自動バッファリングを実現できる唯一のメモリ ブロックでもあります。BSPは各種TMS320デバイス上に実現されるので、ABUメモリの実際のベース アドレスは、あらゆる場合で同じになるとは限りません。使用されている特定のデバイスのメモリ マップは、そのABUメモリの実際のベース アドレスに応じて確認が必要です。

ABUがイネーブルのときも、このメモリの2Kワード ブロックはCPUによってデータおよびプログラム空間内でアクセスできます。2Kワード ブロックがCPUとABUの両方から同時にアクセスされる場合は、これらの間にコンフリクトが発生することが考えられます。コンフリクトが発生する場合は、ABUに優先権がありCPUアクセスは1サイクル分遅れます。それに応じて最悪例では、ABUがメモリ ブロックにアクセスするごと(新たなワードが送信または受信されるたび)にCPUアクセスが1サイクル遅れます。また、オンチップ プログラム メモリをROM保護機能を使って守る場合、ABUメモリの2Kワード ブロックはプログラム メモリにはマップできません。ROM保護機能については、3−22ページ、セクション3.5プログラムおよびデータ セキュリティを参照してください。

ABUが送信および受信の両方のためにイネーブルのとき、シリアル ポート インターフェイスからの送信および受信要求が同時に発生する場合は、送信要求が受信要求より優先的に扱われます。この場合、最初に送信メモリ アクセスが発生して、受信メモリ アクセスはウェイト ステートを発生することで遅れます。送信メモリ アクセスが完了すると、受信メモリ アクセスが実行されます。

9.3.2.1 自動バッファリング制御レジスタ

BSPCEの上位6ビットは、ABU制御レジスタ(ABUC)を構成しています。これらのビットの一部はリード専用で、その他はリード/ライト可能です。図9−25はABUCのビット位置を示し、表9−12はBSPCEの各ABUCビットのファンクションをまとめて示しています。

図9−25. BSP制御拡張レジスタ(BSPCE)図—ABU制御ビット

15	14	13	12	11	10	9−0
HALTR	RH	BRE	HALTX	XH	BXE	Serial Port control
R/W	R	R/W	R/W	R	R/W	

注： R = Read, W = Write

表9−12. BSP制御拡張レジスタ(BSPCE)ビット概要—ABU制御ビット

ビット	名称	リセット値	ファンクション
15	HALTR	0	自動バッファリング受信停止。この制御ビットは、現在のバッファの半分が受信された場合に、自動バッファリング受信を停止するかどうかを決めます。 HALTR = 0 現在のバッファの半分が受信されているとき、自動バッファリングが動作を続行。 HALTR = 1 現在のバッファの半分が受信されているとき、自動バッファリングが停止。この動作が発生すると、BREビットが0にクリアされて、シリアル ポートはスタンダード モードで動作を続行します。
14	RH	0	受信バッファ ハーフ受信。このリード専用ビットは、受信バッファの前後いずれの半分が満たされているかを示します。RINT割り込みが発生したとき(プログラム割り込みとして、またはIFRポーリングによって判別)RHをリードするのは、どの境界線を過ぎたかを識別するのに役立ちます。 RH = 0 バッファの前半分が満たされており、現在は受信によりデータがバッファの後半分に置かれている。 RH = 1 バッファの後半分が満たされており、現在は受信によりデータがバッファの前半分に置かれている。
13	BRE	0	自動バッファリング受信イネーブル。この制御ビットは自動バッファリング受信をイネーブルにします。 BRE = 0 自動バッファリングはディセーブルで、シリアル ポート インターフェイスはスタンダード モードで動作します。 BRE = 1 自動バッファリングはイネーブルで受信に対応します。
12	HALTX	0	自動バッファリング送信停止。この制御ビットは、現在のバッファの半分が送信されたとき、自動バッファリング送信を停止するかどうかを決めます。 HALTX = 0 現在のバッファの半分が送信されても、自動バッファリングは動作を継続します。 HALTX = 1 現在のバッファの半分が送信されたとき、自動バッファリングが停止します。この動作が発生すると、BXEビットが0にクリアされシリアル ポートはスタンダード モードで動作を継続します。

表9−12. BSP制御拡張レジスタ(BSPCE)ビット概要—ABU制御ビット(続き)

ビット	名称	リセット値	ファンクション
11	XH	0	送信バッファ ハーフ送信。このリード専用ビットは、送信バッファの前後いずれの半分が送信されたかを示します。XINT割り込みが発生(プログラム割り込みとして、またはIFRポーリングにより判別)するときXHをリードするのは、どの境界線を通過したかを識別するのに役立ちます。 XH = 0 バッファの前半分が送信されており、現在の送信がバッファの後半分からデータを取り出しています。 XH = 1 バッファの後半分が送信されており、現在の送信がバッファの前半分からデータを取り出しています。
10	BXE	0	自動バッファリング送信イネーブル。この制御ビットは、自動バッファリング送信をイネーブルにします。 BXE = 0 自動バッファリングはディセーブルで、シリアル ポートはスタンダード モードで動作します。 BXE = 1 自動バッファリングはイネーブルで、送信に対応します。
9−0	シリアル ポート制御	—	シリアル ポート インターフェイス制御ビット(9−36ページ、サブセクション 9.3.1.2高度化されたBSP機能を参照)。

9

9.3.2.2 自動バッファリングのプロセス

自動バッファリングのプロセスは、ABUおよびABUメモリの2Kワード ブロックとの間で発生します。シリアル ポート転送が発生するたびに、扱われるデータはABUの制御により自動的に2Kワード ブロックのメモリとの間で転送されます。自動バッファリング モードでシリアル ポート転送が行われる際、各ワードの転送ではスタンダード モード動作のように割り込みが発生しません。これにより、各シリアル ポート転送にCPUが直接関与することを防ぎます。割り込みがCPUに発生するのは、バッファの中間境界線を横切るときのみです。

ABUメモリの2Kワード ブロックでは、開始アドレスおよびバッファ サイズの割り当ては、AXR、ARR、およびBKX、BKRを使ってプログラム可能です。受信および送信バッファは、独立した領域、重なった領域、または同一の領域に置くことができ、必要に応じて特定のバッファから送信しながら同じバッファに受信することが可能です。

自動バッファリングのプロセスは、サーキュラ アドレッシングのメカニズムを利用してABUメモリの2Kワード ブロックの範囲でバッファにアクセスします。このメカニズムは、送信および受信で同じように動作します。各方向で(送信または受信)、2つのレジスタがバッファ サイズおよびバッファにおける現在のアドレスを示しています。これらのレジスタは、送信および受信に対応するブロック サイズおよびアドレス レジスタ(それぞれBKX、BKR、AXR、ARR)です。各ブロック サイズおよびアドレス レジスタのペアは、受信および送信に対応するバッファ アドレスの最上位および最下位を指定します。このサーキュラ アドレッシング メカニズムは、ABUによる2Kワード ブロックへのアクセスにのみ影響します。CPUがこのメモリにアクセスする動作は、メモリ アクセスを実行するアセンブリ言語命令で選択されたアドレッシング モードに必ず従うことになります。

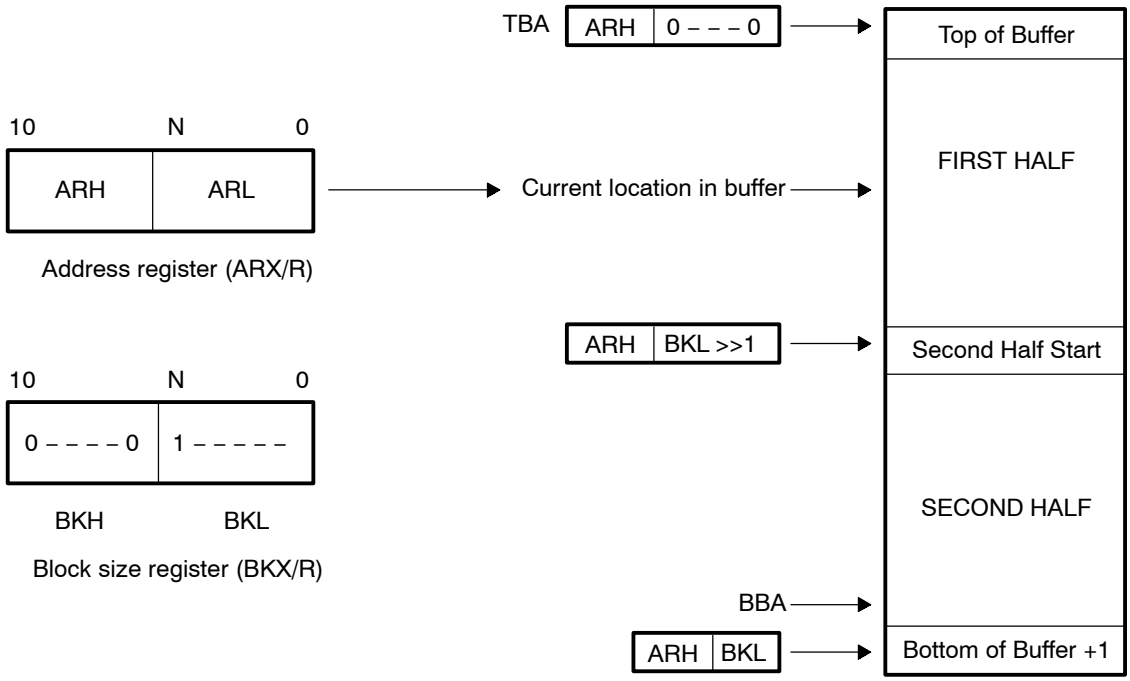
このサーキュラ アドレッシング メカニズムは、指定されたバッファの範囲でABUメモリアクセスを自動的に再循環して、バッファの最下位に到達するたびにバッファの最上位に戻ります。サーキュラ アドレッシングのメカニズムは、任意のバッファ サイズをBKX/Rにロード、および2Kワード ブロック内のバッファの開始アドレであるベース アドレスをARR/AXRをロードすることにより、初期化します(下記に詳述します)。バッファ内の開始アドレスは通常0の場合が多く、バッファの開始を示しますが(バッファ最上位アドレス)、初期開始アドレスは指定のバッファ範囲のどの点でも可能です。

いちど初期化されると、BKX/Rは次の2つの部分から成り立つと考えられます。

- BKX/Rの最上位ビットから0で満たされているすべての上位ビット部分(BKH)。
- それ以外の残りのビット部分(BKL)。その最上位ビットは1で、ビット位置Nとなります。

ビット位置Nは、アドレス レジスタの2つの部分(ARHおよびARL)を定義します。バッファアドレスの最上段(TBA)は、ARHおよび下位N+1個の0ビットの連結によって定義されます。バッファ アドレスの最下段(BBA)は、ARHおよびBKL−1の連結によって定義され、バッファ内の現在のアドレスはARR/AXRの内容によって指定されます。サーキュラー バッファのサイズBKX/Rは必ずNビット(ここではNは2^N>BKX/Rを満たす最小の整数)境界線から始まるか(アドレス レジスタの下位Nビットは0)、または2Kメモリ ブロック内の最下段アドレスから始まります。バッファはその半分にくぎられ、次のような構成になります。最初の半分のアドレス範囲はTBA≧(BKL/2)−1、後半の半分はBKL/2≧(BKL−1)となります。図9−26は、定義されたバッファおよびBKX/RおよびARR/AXRレジスタ間のすべての関係、サーキュラ バッファ アドレスの最下段(BBA)、サーキュラ バッファ アドレスの最上段(TBA)を表しています。

図9−26. サーキュラ アドレッシング レジスタ

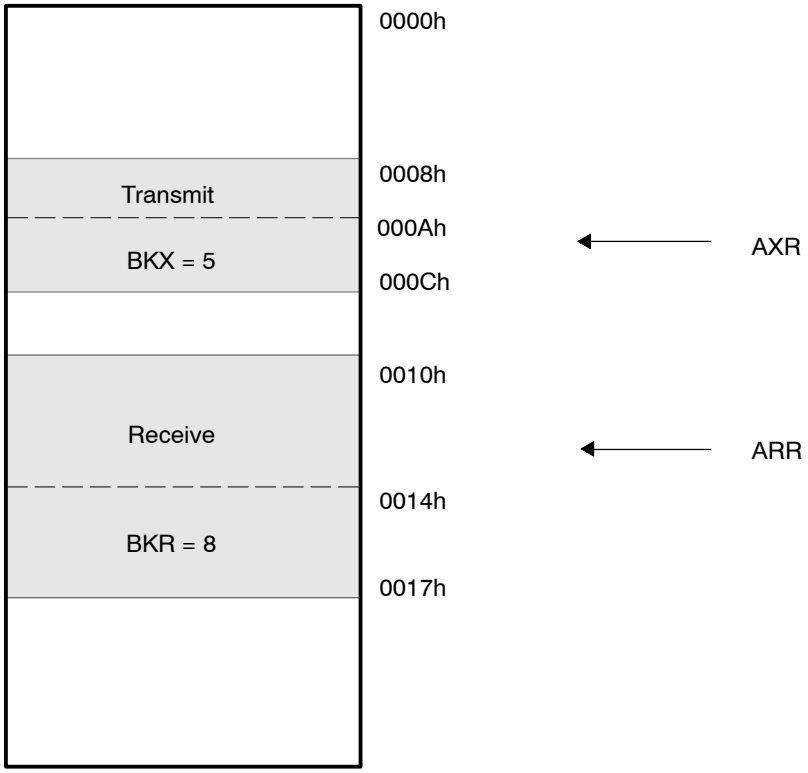


ABUバッファの最小ブロック サイズは2で、最大ブロック サイズは2047となり、1024から2047ワードのあらゆるバッファはABUメモリの2Kブロックのベース アドレスに応じた相対アドレス0x0000から始まります。いずれかのアドレス レジスタ(AXRまたはARR)に、BKX/Rの指定する現在割り当てられてるバッファ サイズの範囲外になる値をロードした場合は、結果として不適切な動作になる場合があります。続くメモリ アクセスは指定された位置から始まりますが、実際にはそれらのメモリ アクセスはそのバッファの範囲外に位置し、ARR/AXRは次に許可されているバッファ開始アドレスに到達するまで、各アクセスごとにインクリメントされます。さらにそれ以降のアクセスは、新しいバッファ開始アドレスでARR/AXRを更新し正しいサーキュラ バッファ アルゴリズムを使って実行されます。不適切にロードされたARR/AXRによって実行されるアクセスは、予期できないメモリ アクセスにより混乱を引き起こすことがあります。

以下の例は、自動バッファリング プロセスの機能的な面を表すものです。送信バッファのサイズが5で(BKX=5)、受信バッファのサイズが8(BKR=8)の場合を考えています(図9−27)。送信バッファは、8の倍数になるあらゆる相対アドレスから始まることができ(アドレス0x0000、0x0008、0x0010、0x0018、.....、0x07F8)、受信バッファは16の倍数になるあらゆる相対アドレスから始まることができます。

この例では、送信バッファは相対アドレス0x0008から始まって、受信バッファは相対アドレス0x0010から始まります。AXRは、0x0008−0x000Cの範囲のあらゆる値を含む可能性があり、ARRは0x0010−0x0017の範囲のあらゆる値を含む可能性があります。この例で、AXRがバッファ サイズ5では許容されていない0x000Dの値によってロードされた場合、メモリ アクセスが実行されAXRはアドレス0x0010に到達するまでインクリメントされます。アドレス0x0010は、バッファ サイズ5で許容される開始アドレスです。ただしこのような処理が行われる場合、AXRは受信バッファと同じベース アドレスで開始する送信バッファを指定することになり、不適切なバッファ動作の原因になるので注意が必要です。

図9−27. 送信バッファおよび受信バッファのマッピング例



自動バッファリングのプロセスは、ワードの受信を知らせるXRDYまたはRRDYがハイになることで、アクティブになります。次に、必要なメモリ アクセスが実行され、指定されたバッファの半分(前半または後半)が処理されていれば、それに続いて割り込みが発生します。BSPCEレジスタのRHおよびXHフラグは、割り込み発生時に前後いずれの半分が処理されたかを示します。

自動ディセーブルを選択する場合(HALTXまたはHALTRビットをセット)、さらに次のバッファ半分(前半または後半)の境界線に遭遇したとき、BSPCEの自動バッファリング イネーブル ビット(BXEまたはBRE)がクリアされるので、自動バッファリングがディセーブルになってそれ以降の要求を発生しません。送信自動バッファリングを停止しても、現在のXSRの内容および最後にロードされるDXRの値の送信は最後まで実行されます。このように、HALTXファンクションを使うと、通常はバッファ境界線の通過と実際に停止する送信の間に遅延が生じます。送信が実際にいつ終了したのかを知る必要がある場合は、最後のビットが送信された後に発生するXRDY=1および$\overline{\text{XSREMPTY}}$=0の状態を見るためのソフトウェアのポーリングが必要になります。

受信部では、HALTRを使用するときは、一番最近のバッファ境界線通過時に自動バッファリングが停止することから、ソフトウェアがその時点で受信割り込みのサービスを開始しない限り、その後の受信が失われる場合があります。これは、BDRRがもはやABUによって自動的にリードされず、またメモリに転送もされないからです。DRRがリードされれないときシリアル ポートがどのように動作するかについては、9−26ページ、サブセクション9.2.6シリアル ポート インターフェイスの例外的な状態を参照してください。

自動バッファリングのプロセスに関連するイベントの段階を次にまとめます。

1) ABUがバッファへのメモリ アクセスを実行。

2) バッファ最下段に到達しないうちは、適切なアドレス レジスタがインクリメント。最下段に到達した場合は、バッファ アドレスの最上段を指すようにアドレス レジスタを変更。

3) バッファの半分またはバッファ最下段の境界線を過ぎた場合、BXINTまたはBRINTを発生してXH/RHを更新。

4) ABUの自動ディセーブル ファンクションが選択されていれば、バッファの半分またはバッファ最下段の境界線を過ぎた場合、ABUを自動ディセーブル。

9.3.2.3 バッファのミスアラインメント(BMINT)割り込み('549のみ)

フレーム同期が発生した時に、ABU送信もしくは受信バッファ ポインタがバッファの先頭アドレスを指していない場合はBMINT割り込みが発生します。この割り込みは、シリアル インターフェイス中に、データに余計なものが入っていたり、クロックやフレーム同期パルスを逃したりといったいくつかの起こりうるエラー状態を検出するのに有効です。これにより、BMINT割り込みは1ワード以上のデータがシリアル インターフェイス中に失われたことを示します。

BMINTは、バッファ ポインタが最初にバッファの先頭アドレスに設定され、データ フレームがバッファ サイズと同じワード数のときのみ、バッファへのミスアラインメントを検出するのに有効です。これらの条件下の時のみ、フレーム同期がバッファの先頭以外のバッファ アドレスで発生した場合にエラー状態となります。上記の条件が満たされている場合、インターフェイスが正常に機能していれば、フレーム同期は、常にバッファ ポインタがバッファ先頭アドレスにきた時に発生されます。

上記の条件を満たさずにBMINTをイネーブルにすると、実際にバッファのミスアラインメントが起きていない時でも割り込みが発生されてしまいます。この場合、プロセッサがBMINT割り込みを無視するように、IMRレジスタでBMINTをマスクしなければなりません。

BMINTは、連続モードで、FIGビットが0にクリアされ、外部シリアル クロックと外部フレーム同期を使った時の自動バッファリング モードで有効になります。

IMRおよびIFRレジスタにおけるBSP0およびBSP1のBMINTのビットはそれぞれビット12、13となり(ビット15がMSBとする)、それらの割り込みベクタの位置はそれぞれ070h、074hとなります。

9.3.3 BSP使用のシステム考察

このサブセクションでは、BSP使用時のシステム レベルの問題について説明します。これには初期化のタイミングの問題、ABUのソフトウェア初期化、パワー ダウン モードの動作などの内容が含まれます。

9.3.3.1 シリアル ポート 初期化のタイミング

'54xデバイスは完全にスタティック(静的)な設計を採用しており、シリアル ポートおよびBSPの双方でシリアル ポート クロックを各転送間で供給すること、また初期化の前に供給することが必ずしも必要ではありません。

そのため、FSX/FSRがCLKX/CLKR開始とともに同時に発生する場合は、適正な動作を保つことができます。ただし、シリアル ポート クロックが前もって供給されているか否かにかかわらず、ポートのリセット復帰後、シリアル ポートの初期化の適切なタイミングは非常に重要で、特に最初のフレーム同期パルスの発生と、ポートのリセットからの復帰のタイミングが重要となります。

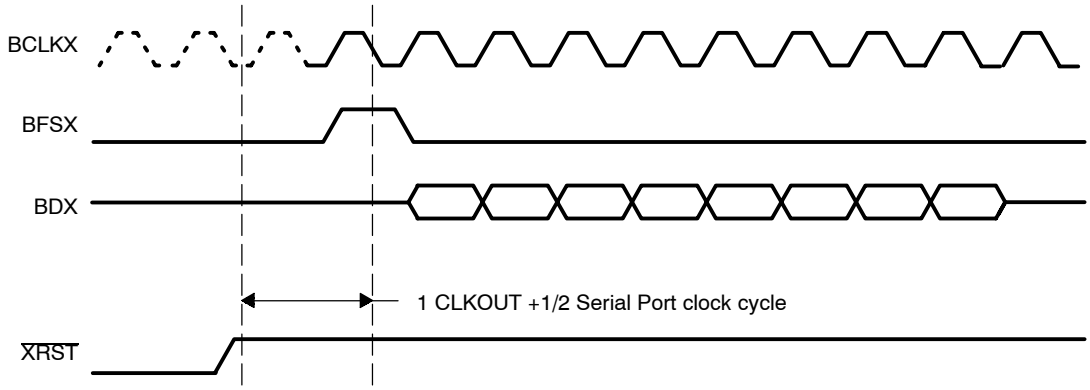
初期化のタイミング条件は、シリアル ポートとBSPでは異なります。シリアル ポートの場合は、FSX/FSRにかかわらずいつでもシリアル ポートがリセットから復帰できますが、フレーム同期の間または後に$\overline{\text{XRST}}$/$\overline{\text{RRST}}$がハイになる場合は、そのフレーム同期は無視されることが考えられます。BSPのスタンダード モードの動作では、受信および外部フレーム同期(TXM=0)をともなう送信の際、適切な動作のためにアクティブなフレーム同期パルスを検出するクロックのエッジより、少なくとも1CLKOUTサイクル+シリアル ポート クロック1/2サイクル(前からクロックが供給されている場合またはそうでない場合も)分だけ先に、BSPが必ずリセットから復帰されなければなりません(図9−28を参照)。

内部クロックおよびフレーム同期をともなう送信動作ではBDXRがロードされるとフレーム同期は自動的に内部で生成($\overline{\text{XRST}}$が解除(1にセット)された後)するので、この条件の対象になりません。

ただし外部シリアル ポート クロックが内部フレーム同期とともに使用される場合は、クロックとの関連で$\overline{\text{XRST}}$を解除するタイミングに応じてフレーム同期生成が遅れる場合があります。

図9−28は、送信部のためのスタンダード モードBSP初期化タイミング条件を示しています。この図は、外部フレーム(TXM=0)およびクロック(MCM=0)、アクティブ ハイ フレーム同期(FSP=0)、およびデータを立ち下がりエッジ時にサンプリング(CLKP=0)するスタンダード モードの動作を示しています。この例では、最初のBCLKXの間にBFSXパルスが生成される場合、フレーム信号は無視され、BDXはハイ インピーダンスになります。

図9−28. スタンダード モードBSP初期化タイミング



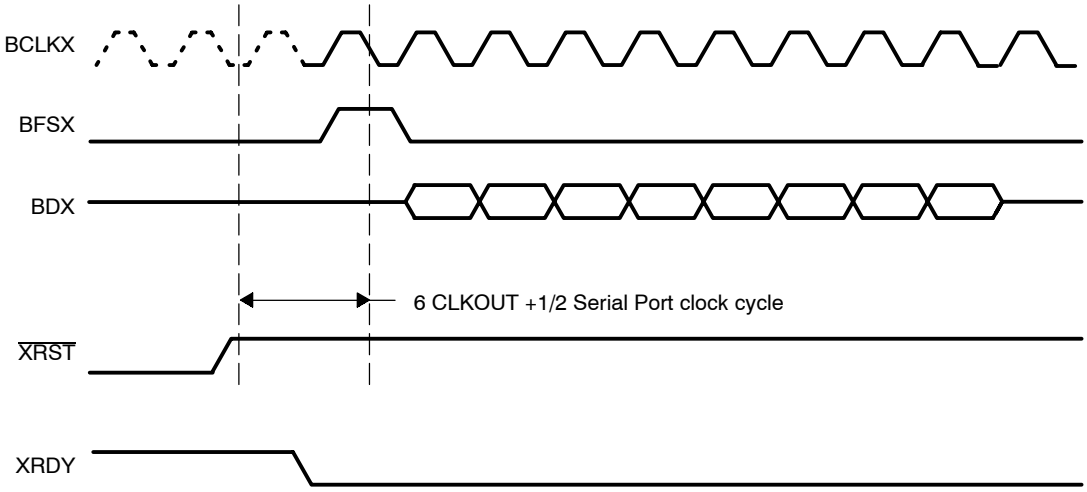
自動バッファリング モードでは、受信、および外部フレーム同期(TXM=0)をともなう送信について、適切な動作のためにアクティブなフレーム同期パルスを検出するクロックのエッジより(前からクロックが供給されている場合またはそうでない場合も)、少なくとも6CLKOUTサイクルと1/2シリアル ポート クロック サイクルの分だけ先に、BSPが必ずリセットから復帰されなければなりません。これは、ABUロジックがアクティブになるための遅延時間によるものです(図9−29を参照)。

内部クロックおよびフレーム同期をともなう送信動作では、$\overline{\text{XRST}}$が解除された後フレーム同期は自動的に内部生成されるので、この条件の対象になりません。

ただし、外部シリアル ポート クロックが内部フレーム同期とともに使用される場合で、$\overline{\text{XRST}}$が解除されるときクロックが供給されていない場合は、クロックとの関連で$\overline{\text{XRST}}$を解除するタイミングに応じてフレーム同期発生が遅れる場合があります。

図9−29は、外部クロックおよびフレーム同期をともなう送信部の、自動バッファリング モードの初期化タイミング条件を示しています。図は、外部フレーム(TXM=0)およびクロック(MCM=0)、アクティブ ハイ フレーム同期(FSP=0)、およびデータを立ち下がりエッジ時にサンプリング(CLKP=0)する、自動バッファリング モードの動作を示しています。

図9−29. 自動バッファリング モードの初期化タイミング



9.3.3.2 初期化例

スタンダード モードでBSP動作を開始または再開するには、シリアル ポートの初期化と同じ手順(9−3ページ、セクション9.2シリアル ポート インタフェース参照)をソフトウェアで実行し、さらにBSPCEレジスタを初期化して必要な応用機能を構成することが必要です。自動バッファリング モードでBSP動作を開始または再開するには、同様の手順を実行する必要があり、さらに自動バッファリング レジスタの初期化が必要になります。

例9−3および例9−4は、バッファド シリアル ポートの適正な動作の手順を示しています。この例では、シリアル ポートとCPU間の通常のI/O処理に割り込みの利用をベースにしています。また、'545のペリフェラル構成をリファレンスとして使用しています。この例は、バッファド シリアル ポートを自動バッファリング モードで動作させるための初期化を示しています。両方の例で、送信および受信の割り込みを使ってABU割り込みのサービスを実行するのを前提にしていますが、割り込みフラグ レジスタ(IFR)のポーリングも使用できます。送信および受信のセクションは、両方を同時に初期化するかまたはシステムの条件に応じて個別にも初期化できます。

例9−3では、バースト モード、外部フレーム同期、外部クロックを選択した上で、送信動作のみのためにシリアル ポートを初期化しています。データ フォーマットは16ビットを選択しており、フレーム同期とクロックの極性はハイが設定されています。BSPCEのABUCセクションでBXEビットをセットすることで送信自動バッファリングをイネーブルにして、HALTXを1にセットされていることで設定されたバッファの半分が送信された段階で送信が停止します。

例9−4では、連続モードを選択した上で受信動作のみのため、シリアル ポートを初期化しています。フレーム同期とクロック極性をローに、データ フォーマットは16ビットに設定しています。フレーム無視を選択しているので、2つの受信バイト データを単一の受信ワードにパックして、必要なメモリを最小化しています。BSPCEのABUCセクションでBREビットをセットすることにより、受信自動バッファリングをイネーブルにしています。

例9−3および例9−4では、使用されている送信および受信割り込みは、BSPが'542、'543、'545、'546、'548、'549の場合で、これらはBSPを含むデバイスです。

例9−3. BSP送信初期化ルーチン

	動作	説明
1)	BSPCに0008hをライトすることでシリアル ポートをリセット、初期化します。	これにより、シリアル ポートの送信および受信部分をリセット状態にして、FSXおよびCLKX信号の外部生成、および各16ビット ワードの送受毎にFSXが必要なように、シリアル ポートを設定します。
2)	IFRに0020hをライトして、待たされているシリアル ポート割り込みをクリアします。	初期化の前に発生している割り込みを消却します。
3)	IMRとともに0020hをOR演算することによりシリアル ポート割り込みをイネーブルにします。	送信割り込みをイネーブルにします。
4)	ST1のINTMビットをクリアして、割り込みをグローバルに(必要に応じて)イネーブルにします。	CPUが応答するように、割り込みをグローバルにイネーブルにする必要があります。
5)	BSPCEに1400hをライトして、ABU送信を初期化します。	これにより、次のFSXが入力されるまで、BSPがバッファの終わりで送信を停止します。
6)	AXRにバッファの開始アドレスをライトします。	ABUに最初のバッファ アドレスを識別させます。
7)	BKXにバッファ サイズをライトします。	ABUにバッファ サイズを識別させます。
8)	BSPCに0048hをライトして、シリアル ポートを開始します。	これにより、シリアル ポートの送信部分をリセットから復帰して、手順1および5で定義された状態で動作を開始します。

例9−4. BSP受信初期化ルーチン

	動作	説明
1)	BSPCに0000hをライトすることでシリアル ポートをリセット、初期化します。	これにより、シリアル ポートの受信部分をリセット状態にして、16ビット ワードの連続受信モードでシリアル ポートが動作するように設定します。
2)	IFRに0010hをライトして、待たされているシリアル ポート割り込みをクリアします。	イニシャライズの前に発生している割り込みを消却します。
3)	IMRとともに0010hをOR演算することにより、シリアル ポート割り込みをイネーブルにします。	受信割り込みをイネーブルにします。
4)	ST1のINTMビットをクリアして、割り込みをグローバルに(必要に応じて)イネーブルにします。	CPUが応答するように、割り込みをグローバルにイネーブルにする必要があります。
5)	BSPCEに2160hをライトして、ABU受信を初期化します。	これにより、BSPが連続的に受信を行い、新たなFSRが受信されても再始動しません。
6)	ARRにバッファの開始アドレスをライトします。	ABUに最初のバッファ アドレスを識別させます。
7)	BKRにバッファ サイズをライトします。	ABUにバッファ サイズを識別させます。
8)	BSPCに0080hをライトして、シリアル ポートを開始します。	これにより、シリアル ポートの送信部分をリセットから復帰して、手順1および5で定義された状態で動作を開始します。

9.3.4 パワーダウン モードのBSP動作

'54xにはいくつかのパワーダウン モードがあり、これによりデバイスの一部またはすべてを休止状態にして通常動作に比べて消費電力を大幅に節減します。パワーダウン モードを呼び出す方法は、IDLE命令の実行、HMステータス ビットを1にセットして$\overline{\text{HOLD}}$入力をロー駆動する方法など、いくつかあります。他のペリフェラル(タイマ、スタンダード シリアル ポートなど)と同様に、BSPは送信割り込み(BXINT)または受信割り込み(BRINT)を使ってCPUをIDLEから復帰させることができます。

IDLEまたはHOLDモードでは、シリアル ポートの場合と同様にBSPの動作は続行します。IDLE2/3の状態にあるときは、シリアル ポートやこのパワーダウン モードで停止する他のオンチップ ペリフェラルとは異なり、BSPは動作が可能です。

スタンダード モードでは、デバイスがIDLE2/3の状態にありながらBSPが外部クロックおよびフレーム同期を使用する場合は、ポートは動作を継続でき、デバイスがIDLE2またはIDLE3命令を実行する前にINTM=0だった場合、送信割り込み(BXINT)または受信割り込み(BRINT)でデバイスをIDLE2/3のモードから復帰させることができます。内部クロックおよびフレーム同期の両方または一方を使用している場合、BSPはCPUが動作復帰するまでIDLE2/3の状態のままになります。

自動バッファリング モードでは、デバイスがIDLE2/3の状態にありBSPが外部クロックおよびフレーム同期を使用する場合、送受信イベントによって、DXR(またはDRR)がメモリ転送を実行するのに必要なサイクルのために内部BSPクロックがオンになります。そして、転送の完了直後に内部BSPクロックが自動的にオフになるので、デバイスはIDLE2/3のままになります。デバイスがIDLE2またはIDLE3命令を実行する前にINTM=0だった場合送受信バッファの半分または全部が空か満たされている場合、デバイスはABU送信割り込み(BXINT)または受信割り込み(BRINT)によってIDLE2/3から立ち上がります。

9.4 時分割多重(TDM)シリアル ポート インターフェイス

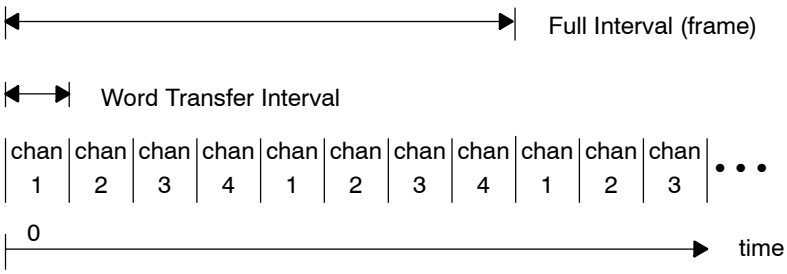
'54xデバイスは時分割多重(TDM)シリアル ポートにより、最大7個の他のデバイスとシリアルに通信できます。したがってTDMポートは、マルチプロセシング アプリケーションのために簡単で効率的なインターフェイスを実現できます。

TDMシリアル ポートは、9−3ページ、セクション9.2で説明したシリアル ポートを拡張したものです。TDMシリアル ポート制御レジスタ(TSPC)のTDMビットによって、ポートをマルチプロセシング モード(TDM=1)またはスタンドアロン モード(TDM=0)に設定できます。スタンドアロン モードの場合は、ポートはセクション9.2で説明されているように動作します。マルチプロセシング モードの場合は、ポートはこのセクションで説明するような動作になります。ポートは、セクション9.2で説明されるように、$\overline{\text{XRST}}$およびRRSTビットによってシャットダウンできるので、消費電力の削減が可能です。

9.4.1 時分割多重の基本動作

時分割多重は、時間間隔(フル インターバル)をいくつかの2次的な時間間隔(サブ インターバル)に分割することで、それぞれのサブ インターバルは事前の設定に応じた通信チャネルを表します。図9−30は、4チャネルのTDM方式を示しています。最初のスロットはchan 1(チャネル1)、2番目のスロットはchan 2(チャネル2)、以下順番にラベルが付けられます。チャネル1は最初の通信期間にアクティブになり、その後4つの通信期間ごとにアクティブになります。残りの3つのチャネルもチャンネル1と同様のタイミングで順に繰り返します。

図9−30. 時分割多重



'54xのTDMポートは、8つのTDMチャネルを使用します。各チャネルに対応してどのデバイス(1つ)が送信して、どのデバイス(1つ以上)が受信するかは、個別に指定できます。これにより、プロセッサ相互の通信の柔軟性が高まっています。

9.4.2 TDMシリアル ポート インターフェイス レジスタ

TDMシリアル ポートは、6つのメモリ マップド レジスタとその他2つのレジスタ(TRSRおよびTXSR)によって動作します。TRSRおよびTXSRはプログラムから直接アクセスできませんが、ダブル バッファ機能の実現のために使用されます。表9−13は、これら8つのレジスタを示しています。

表9−13. TDMシリアル ポート レジスタ

アドレス	レジスタ	説明
†	TRCV	TDMデータ受信レジスタ
†	TDXR	TDMデータ送信レジスタ
†	TSPC	TDMシリアル ポート制御レジスタ
†	TCSR	TDMチャネル選択レジスタ
†	TRTA	TDM受信/送信アドレス レジスタ
†	TRAD	TDM受信アドレス レジスタ
—	TRSR	TDMデータ受信シフト レジスタ
—	TXSR	TDMデータ送信シフト レジスタ

† セクション8.1ペリフェラル メモリ マップド レジスタを参照

- TDMデータ受信レジスタ(TRCV)。このレジスタは16ビットで、入力されるTDMシリアル データを保持します。TRCVにはDRRと同様の機能があります(9−4ページ参照)。
- TDMデータ送信レジスタ(TDXR)。このレジスタは16ビットで、送出されるTDMシリアル データを保持します。TDXRにはDXRと同様の機能があります(9−4ページ参照)。
- TDMシリアル ポート制御レジスタ(TSPC)。このレジスタは16ビットで、TDMシリアル ポート インターフェイスのモード制御と状態を示すビットを含んでいます。TSPCレジスタはSPCレジスタと同一ですが(図9−3参照)、ビット0がTSPCのTDMモード イネーブル制御ビットとして働く点が異なります。このTDMビットによって、ポートをTDMモード(TDM=1)またはスタンドアロン モード(TDM=0)に設定できます。スタンドアロン モードでは、ポートはスタンダード シリアル ポートとして動作します(9−3ページの説明を参照)。
- TDMチャネル選択レジスタ(TCSR)。このレジスタは16ビットで、各'54xデバイスがどのタイム スロットで送信を行うかを指定します。
- TDM受信/送信アドレス レジスタ(TRTA)。このレジスタは16ビットで、下位(RA0−RA7)8ビットにより'54xデバイスの受信アドレスを、上位(TA0−TA7)8ビットにより'54xデバイスの送信アドレスを指定します。
- TDM受信アドレス レジスタ(TRAD)。このレジスタは16ビットで、TDMアドレス ライン(TADD)の状態など様々な情報を表します。

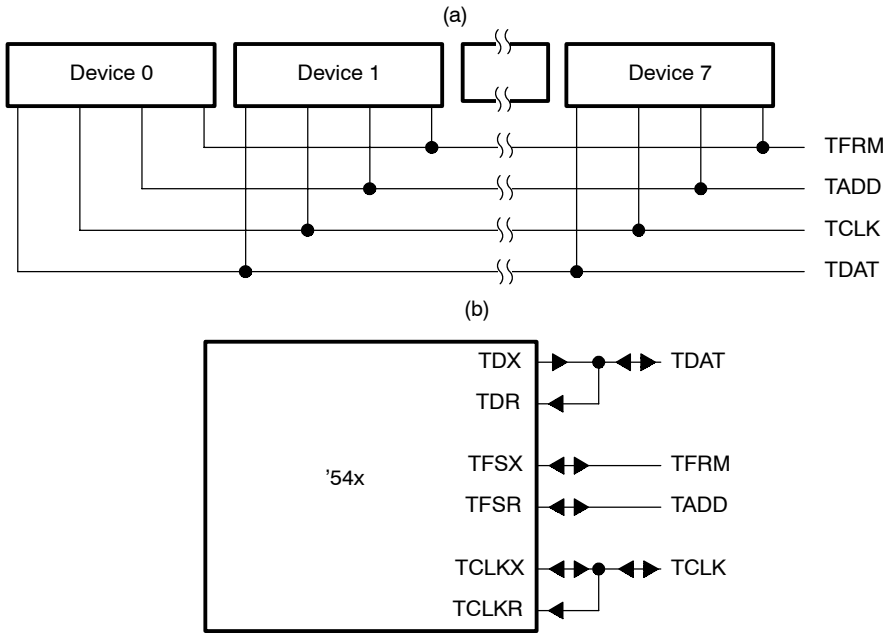
- ❑ TDMデータ受信シフト レジスタ(TRSR)。このレジスタは１６ビットで、入力ピンからTRCVへのデータの受信を制御します。TRSRにはRSRと同様の機能があります(9−4ページ参照)。

- ❑ TDMデータ送信シフト レジスタ(TXSR)。このレジスタは１６ビットで、TDXRから送出されるデータ転送を制御して、データ送信(TDX)ピン上に送信されるデータを保持します。TXSRには、XSRと同じ機能があります(9−4ページ参照)。

9.4.3 TDMシリアル ポート インターフェイスの動作

図9−31(a)は、'54xのTDMポートのアーキテクチャを示しています。最大8つのデバイスを4ワイヤ シリアル バス上に接続できます。この4ワイヤ バスは、従来型シリアル ポートのクロック、フレーム、データのそれぞれのバス(TCLK、TFRM、TDAT)、および追加ワイヤ(TADD)から構成されています。TADDはデバイスのアドレッシング情報の搬送に使われます。TDATおよびTADDは双方向信号で、あるフレーム動作内の異なるタイム スロット間でバス上の異なるデバイスによってドライブされます。

図9−31. TDM4ワイヤ バス



特定のタイム スロットで特定のデバイスによってドライブされるTADDラインは、そのタイム スロットの間にTDMを設定し、どのデバイスが有効なTDM受信を実行すべきかを決めます。これはシリアル ポートの有効なリード動作に似ています(9−3ページ、セクション9.2シリアル ポート インターフェイスを参照)。ただし、対応するTDMレジスタの一部の名前が異なります。TDM受信レジスタはTRCV、TDM受信シフト レジスタはTRSRです。データは、双方向のTDATライン上に送信されます。

図9−31(b)では、デバイスTDXおよびTDRのピンが外部で接続されており、TDATラインを形成します。また、特定スロットのデータおよびアドレス ライン(TDAT、TADD)をドライブできるのは、ひとつのデバイスのみです。その他すべてのデバイスのTDATおよびTADD出力はそのタイム スロットの間はハイ インピーダンスになりますが、これはTDMポート制御レジスタの適切なプログラミングによって実現します(詳細はこのセクションの後半に説明します)。一方、そのスロットでは、すべてのデバイス(そのスロットをドライブするものも含む)がTDATとTADDラインをサンプルして、現在の送信がバス上のいずれかのデバイスによってリードできるかどうかを判断します(詳しくはこのセクション後半を参照してください)。デバイスが応答すべきアドレスを認識した場合、有効なTDMリードが発生して、その値がTRSRからTRCVレジスタに転送されます。そして受信割り込み(TRINT)が発生して、TRCVは有効なデータを受信し、リードできることを示します。

すべてのTDMポート動作はTCLKとTFRM信号によって同期が取られます。これらは、ひとつのデバイスのみ(通常同じデバイス)によって生成され、そのデバイスはTCLKとTFRMソースになります。マスターという表現は、ひとつのデバイスが他方を制御することを意味しますが、ここではあてはまらないためこの言葉は使用しません。また、スロットの争奪を防ぐためにTCSRを設定する必要があります。残りのデバイスはこれらの信号を入力として使用します。図9−31(b)では、TCLKXとTCLKRが外部で結合して、TCLKラインを形成している様子を示しています。また、TFRMとTADDは、それぞれTFSXとTFSRピンから生じます。これにより、TDMシリアル ポートがスタンダード モードで使いやすくなっています。

TDMポート動作は、6つのメモリ マップド レジスタによって制御されます。図9−32は、これらのレジスタのレイアウトを示しています。TRCVとTDXRレジスタは、それぞれDRRとDXRレジスタと同じ機能を持ちます(セクション9.2シリアル ポート インターフェイスを参照)。TSPCレジスタはSPCレジスタと同一ですが、ビット0がTSPCの場合TDMモード イネーブル制御ビットとして働く点が異なります。このビットにより、ポートをTDMモード(TDM=1)またはスタンドアロン モード(TDM=0)に設定できます。スタンドアロン モードでは、ポートはスタンダード シリアル ポートと同様に動作します(セクション9.2を参照)。これらのレジスタにおけるビットの機能について、詳細は9−63ページ、サブセクション9.4.6TDMシリアル ポート インターフェイス動作の例を参照してください。

図9−32. TDMシリアル ポート レジスタ図

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRCV	Receive Data															
TDXR	Transmit Data															
TSPC	Free	Soft	X	X	XRDY	RRDY	IN1	IN0	$\overline{\text{RRST}}$	$\overline{\text{XRST}}$	TXM	MCM	X	0	0	TDM
TCSR	X	X	X	X	X	X	X	X	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
TRTA	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
TRAD	X	X	X2	X1	X0	S2	S1	S0	A7	A6	A5	A4	A3	A2	A1	A0

注： X=Don't care.

TDMモードを選択した場合、TSPCレジスタのDLBとFOビットはハードによって0にセットされ、その結果デジタル ループバック モードへのアクセスが行えず、16ビットの固定ワード長になります(異なるタイプのループバックについては9−63ページ、サブセクション9.4.6を参照してください)。また、TDM=1のとき、FSMの値はポートに影響せず、アンダーフローおよびオーバーラン フラグの状態は不確定です(TDMモードの例外的な処理については9−63ページ、サブセクション9.4.5TDMシリアル ポート インターフェイスの例外的な状態を参照してください)。TDM=1のとき、TSPCレジスタの内容の変更は、現在のフレームのチャネル7が完了したとき有効になります。したがって、TSPCの値は現在のフレームでは変更されません。加えられた変更は、次のフレームで有効になります。

TCLKおよびTFRMのタイミング信号のためのソース デバイスは、それぞれMCMおよびTXMビットによって設定されます。TCLKソース デバイスは、TSPCレジスタのMCMビットを1にセットすることにより指定できます。通常、このデバイスはTDMポート クロック信号のTCLKを供給するものと同じになります。TCLKXピンは、MCM=0のときは入力として、MCM=1のときは出力として設定されます。後者の場合(内部'54xクロック)、MCM=1のデバイスがTDMバス上のすべてのデバイスにクロック(TCLK周波数=CLKOUT周波数の4分の1)を供給します。このクロックは、すべてのデバイスでMCM=0ならば外部ソースによって供給することも可能です。TXM=0のとき、TFRMも外部から供給することができます。ただし外部のTFRMは、適正動作させるためにTCLKに関連したTDM受信タイミングに合わせなければなりません。いかなる場合でも、2つ以上のデバイスでMCMまたはTXMを1にセットしてはなりません。クロックおよびフレーム信号を供給するデバイスがどれかを指定できるのは、通常はシステムの初期化中に1回のみです。

あるデバイスのTDMチャネル選択レジスタ(TCSR)は、どのタイム スロットでそのデバイスが送信するかを指定します。TCSRのビット0−7のいずれか(ひとつ以上)に1をセットすると、対応するタイムスロットで送信部をアクティブに設定できます。システム レベルの重要な制約は、同じタイムスロット間に送信できるのはひとつのデバイスに限られることです。デバイスはバス衝突をチェックしないので、スロットを矛盾しないように割り当てる必要があります。

TSPCの動作と同様に、特定のフレーム間に行うTCSRへのライトは、次のフレームの間でだけ有効です。ただし、特定のデバイスは複数のスロットで送信できます。詳しくは9−61ページ、サブセクション9.4.4TDMモードの送受信動作で、TRTA、TDXR、TCSRの利用とともに説明します。

特定のデバイスのTDM送受信アドレス レジスタ(TRTA)は、重要な2つの情報を指定します。TRTAの下位の半分はデバイスの受信アドレスを指定して、上位は送信アドレスを指定します。受信アドレス(RA7−RA0、図9−32を参照)は8ビットの値で、デバイスはこの値と特定スロットのTADDラインでデバイスがサンプルする8ビットの値を比べて、有効なTDM受信を実行するかどうかを判断します。したがって受信アドレスは、そのデバイスが受信を行う可能性のあるスロットを決定します。受信を行うかどうかはそれらのスロットにあるアドレス次第で、送信デバイスによって指定されます。このプロセスは、スロット毎に各デバイスで発生します。

送信アドレス(TA7−TA0、図9−32参照)は、割り当てられたスロット上での送信動作中に、デバイスがTADDライン上でドライブするアドレスです。送信アドレスは、ドライブされたデータ上でどの受信デバイスが有効なTDM受信を行えるかを決定します。

TADD上の送信アドレスをドライブするのは、一度にひとつのデバイスのみです。各プロセッサは、受信アドレス(RA7−RA0)とTADDライン上でサンプルする値とで論理積演算を行います。この処理によってゼロでない値が生じる場合は、有効なTDM受信がプロセッサ上で実行されます。このように別のデバイスに送信を行うデバイスでは、TRTA(送信アドレス)の上位半分には少なくともひとつのビットが1にセットされなければならず、このビットが受信デバイスのTRTA(受信アドレス)の下位半分の1にセットされたビットと一致します。この方法でTRTAを設定することにより、単一のデバイスでひとつまたは複数のデバイスに送信可能になり、また、どの単一デバイスもひとつまたは複数の送信部から受信できるようになります。このように、送信デバイスはどのデバイスの受信アドレスも変更せずに、受信するデバイスを制御できるようになります。

TDM受信アドレス レジスタ(TRAD)は、TADDラインの状態に関する各種の情報を持っており、TADDラインをポーリングしてその信号の前の値を検証したり、命令サイクルとTDM ポート タイミングの関係を検証したりできます。

ビット13−11(X2−X0)には、そのスロットで有効なデータ受信が実行されたかどうかに関わらず、現在のスロットの数値が保持されます。この値はスロットの始めでラッチされ、スロットの終わりまで保持されます。

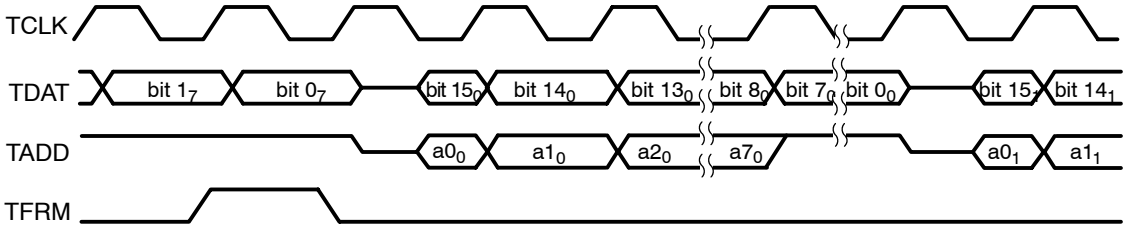
ビット10−8(S02−S0)はデータが受信された最後のスロットの数値+1の値を保持します(例えば、最後の有効データ リードが前のフレームのスロット5で発生した場合、これらのビットには数値6が保持されます)。この値は、最後の有効データ受信が発生したスロットの終わりのTDM受信割り込み(TRINT)の間にラッチされ、次の有効データ受信が起こるスロットの最後まで保持されます。

ビット7−0(A7−A0)には、有効なデータ受信が実行されたかどうかに関わらず、TADDライン上でサンプルされた最後のアドレスが保持されます。この値は各スロットで途中までラッチされ(TADDからシフトされ)、有効な受信が実行されたか否かに関わらず次のスロットの途中まで保持されます。

9.4.4 TDMモードの送受信動作

図9−33は、TDMポート転送のタイミングを示しています。TCLKとTFRM信号は、タイミング ソース デバイスによって生成されます。TCLK周波数は'54xデバイスによって生成させている場合、CLKOUT周波数の4分の1になります。TFRMパルスは128TCLKサイクル毎に発生し、スロット7のビット0(前のフレームの最後のビット)と一致するタイミングになります。TFRMとTCLKの関係により、8つの各タイムスロットに対して16のデータ ビットをTDATラインにドライブできます。これにより、'54xの内部クロックを使用することを前提にすれば、プロセッサが各スロットごとに最大64の命令を実行できます。スロット0およびMSBファーストで始まり、送信部は16のデータ ビットを各スロットにドライブします。各ビットは1TCLKサイクル分だけ持続します。ただし、各スロットの最初のビットは例外で、他のビットの半分しか持続しません。データは、送信デバイスによってTDATラインに送られ、TCLKの立ち上がりエッジで受信デバイスによってTDATラインからサンプルされます(TDMインターフェイス タイミングの詳細はデータシートを参照してください)。

図9−33. シリアル ポート タイミング(TDMモード)



データ転送と同時に、送信デバイスは送信アドレスをTADDラインに各スロットごとにドライブします。この情報はTDAT上の情報とは異なり、1バイトの長さになり、スロットの前半にLSBファーストで送信されます。スロット後半(最後の8TCLK周期)、TADDはハイにドライブされます。TDM受信ロジックは、最初の8TCLK周期についてのみTADDラインをサンプルして、スロット後半では無視します。したがって、送信デバイス('54xでない場合)はスロット後半にTADDをハイまたはローのどちらにドライブしてもかまいません。TDATのように、送信される最初のTADDビットは1/2TCLKサイクルの間だけ持続します。

あるスロットにおいて送信するように設定されたデバイスがTDMバス上にない場合(どのデバイスのTCSRレジスタも、スロットに対応するような1をセットしていない)、そのスロットは空と考えらえれます。空スロットでは、TADDおよびTDATの双方がハイ インピーダンスになります。ただし、この状態では見かけの受信が起こる可能性があります。というのは、TDATとTADDは常にサンプルされ、デバイスのアドレスがTADDライン上のアドレスに一致すると、そのデバイスは有効なTDM受信を実行するからです。見かけのリードを回避するには、1kΩのプルダウン抵抗をTADDラインに接続する必要があります。これにより、TADDラインが空スロット上でローでリードされます。さもなければ、特定の受信アドレスに合うTADDライン上のノイズが擬似的にリードされてしまいます。電力消費の観点から抵抗を用いたくない場合は、0hに等しい送信アドレスを持つ任意のプロセッサが、空のスロットのTDXRにライトを行うことでそれらのスロットをドライブできます。スロットの操作については、このサブセクションの後半で説明します。TDATラインでは、1kΩの抵抗は必要ありません。

空のTDMスロットは、最終的に次のいずれかの場合になります。最も明白なケースは、上述のように、どのデバイスのTCSRもそのスロットで送信するように設定されない場合に発生します。次に考えられるケースは、TDXRが特定フレームの送信スロットの前にロードされなかった場合に発生します。このようなケースは、実際のTCSRの内容が次のTFRMパルス発生まで更新されないことから、TCSR内容が変更される場合にも発生します。したがって、それ以降に変更されたものは次のフレームでのみ効果があります。これは受信アドレスでも同様です(TRTAの下位半分)。一方、送信アドレス(TRTAの上位半分)およびTDXRはTRTAかTDXRをロードする命令が実行されたときに、そのスロットにまだ達していないことを想定して現在のフレーム内で特定のスロットのために変更されることがあり、TDXRがロードされるたびに送信アドレスをロードする必要はなく、TDXRロードが発生して送信が始まるとき、TRTAの現在の送信アドレスがTADD上に送信されます。

現在のスロット番号は、TRADのX2−X0ビットをリードすることで取得できます。これにより、スロットごとのTDMポートの再設定が柔軟に行えるようになり、必要に応じてスロットの共有も可能です。この機能を使用する上で、実行される命令とTDMポートのフレーム/スロットのタイミング関係を理解することが重要になります。

TDMポートをスロットごとに操作する場合は、必要な効果が必要なときに現れるように、変更は迅速に適切なレジスタに加える必要があります。また、TCSRと受信アドレス(TRTAの下位半分)が新たなフレームの開始時のみに有効になること、また、送信アドレス(TRTAの上位半分)とTDXR(送信データ)が新たなスロットの開始時に有効になることも、重要な注意点です。

送信アドレスと受信アドレスは両方ともTRTAレジスタにあり、送信デバイスがどのデバイスが受信するかを指定できるようになっているので、送信アドレスを進行中に変更する場合、受信アドレスを書き換えないように十分注意することが必要です。

9.4.5 TDMシリアル ポート インターフェイスの例外的な状態

ひとつのプロセッサが複数のスロットで送信できるTDMアーキテクチャの特質により、オーバーランとアンダーフローの概念は不確定になっています。したがって、オーバーランとアンダーフローのフラグはTDMモードではアクティブではありません。

受信部では、TRCVをリードせずに有効な受信動作を開始すると(TADDの値とデバイスの受信アドレスにより)、TRCVの現在の値が上書きされ、受信部は停止しません。一方、TDXRが送信の前に更新されていない場合、TADDまたはTDATラインはドライブされず、これらのピンはハイ インピーダンスの状態のままになります。この動作モードは、見かけの送信が発生するのを回避します。

TFRMパルスがフレーム中の不適切な時間に発生すると、スロットおよびビット番号が不確定になりTDMポートは適正動作を続けられません。したがって、TFRMは128TCLKクロック毎に発生させなければなりません。シリアル ポートとは異なり、TDMポートは送信中にフレーム同期パルスによる初期化はできません。不適切なタイミングのTFRMパルスを修正するには、TDMポートをリセットする必要があります。

9.4.6 TDMシリアル ポート インターフェイス動作の例

以下に、TDMシリアル ポート動作の例を示します。ここでは、関連する一部の重要なデバイス レジスタの内容を示し、ポート動作におけるこの設定の効果について説明します。この例では、8つのデバイスがTDMシリアル ポートに接続しています(図9−34参照)。

表9−14は、表記されている受信部に与えられた8つのチャネルのTADD値を示しています。この例では、8つのデバイスによって相互に通信を行う設定を表しています。例では、スロット0の間にデバイス0が他のあらゆるデバイス アドレスに送信を行います。以降のタイム スロットでは、デバイス1から7はそれぞれ他のプロセッサひとつを相手に通信を行います。

図9−34. TDM設定例の図

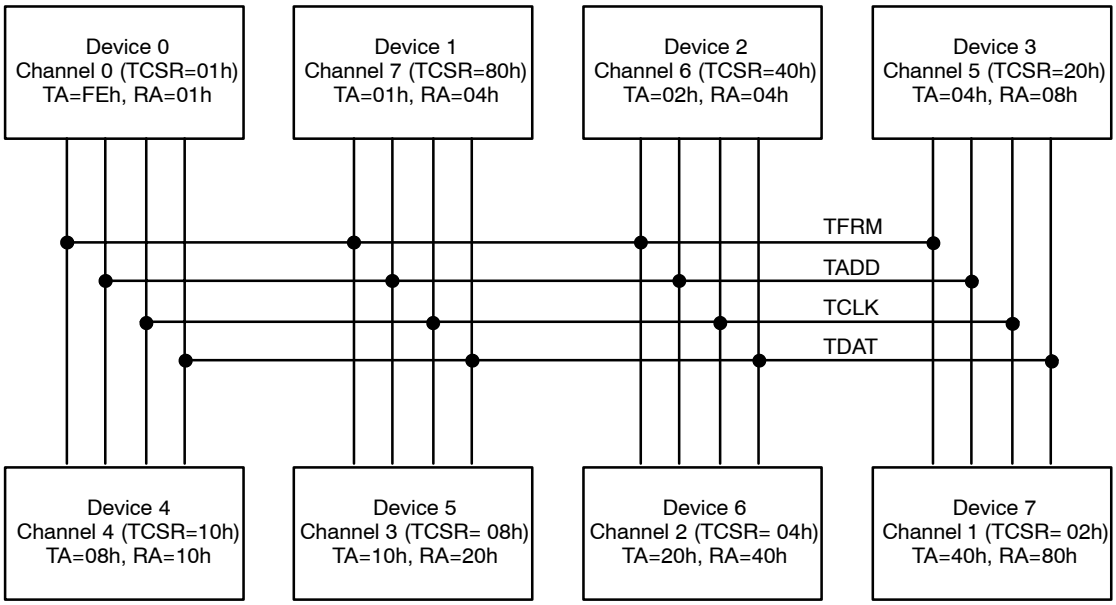


表9−14. プロセッサ相互通信手順

チャネル	TADDデータ	送信デバイス	受信デバイス
0	FEh	0	1−7
1	40h	7	6
2	20h	6	5
3	10h	5	4
4	08h	4	3
5	04h	3	2
6	02h	2	1
7	01h	1	0

表9−15は、表9−14に示された筋書から得られる、各デバイスのTDMシリアル ポート レジスタの内容を示しています。デバイス0は、すべてのチャネルとデバイスに対してクロックとフレーム制御信号を供給します。TCSRとTATRの内容は、与えられたチャネルでどのデバイスが送信して、どのデバイスが受信するかを指定します。

表9−15. TDMレジスタの内容

Device	TSPC	TRTA	TCSR
0	xxF9h	FE01h	xx01h
1	xxC9h	0102h	xx80h
2	xxC9h	0204h	xx40h
3	xxC9h	0408h	xx20h
4	xxC9h	0810h	xx10h
5	xxC9h	1020h	xx08h
6	xxC9h	2040h	xx04h
7	xxC9h	4080h	xx02h

この例では、あるデバイスの送信アドレス(TRTAの上位バイト)が受信デバイスの受信アドレス(TRTAの下位バイト)と一致しています。ただし、送受信アドレスを、完全に一致させる必要はありません。受信部で実現するマッチング処理は、論理積演算です。したがって、フィールドの1ビットが一致すれば受信が発生します。この方式の長所は、送信デバイスが送信アドレスを変えるだけで送信データを受信するデバイスを選択できることです(各デバイスの受信アドレスが固有である限り、受信デバイスの受信アドレスを変更する必要はありません)。例では、デバイス0は、TRTAの上位バイトにライトを行うだけで、他のデバイスとのどの組み合わせにも送信できます。たとえば、送信デバイスが進行中でTRTAを8001hに変更すると、そのデバイスはデバイス7にのみ送信します。

送受信動作はTCLKの立ち上がりエッジで実行されるので、デバイスはそのデバイス自体にも送信できます(TDMインターフェイスのタイミングについては'54xデータシートを参照)。このようなタイプのループバックを可能にするには、標準TDMポート インターフェイス接続を使用する必要があります(図9−31参照)。このとき、デバイス0のTRTAが0101hであれば、デバイス0自体のみに送信します。

例9−5から例9−8は、TDMシリアル ポートの適正動作の手順を表しています。この例では、基本的に割り込みを使ったシリアル ポートとCPU間の通常のI/Oの処理を示します。これらの例は'542のペリフェラル設定を参照しました。

例9−5では、デバイス0からデバイス1へ流れる連続的な値の単方向の送信の進行手順を示しています。デバイス0はスロット0で送信を行い、01hの送信アドレスを持ちます。例9−7は、デバイス1の進行手順を示しています。ここでは01hの受信アドレスを用いています。

例9−5. TDMシリアル ポート送信初期化のルーチン

	動作	説明
1)	TSPCに0039hをライトすることによりTDMシリアル ポートをリセット、初期化します。	これにより、TDMシリアル ポートの送受信部をリセットの状態にして、シリアル ポートをTDMモードで内部TFRMおよびTCLK信号を使って動作するように設定します。
2)	IFRに0080hをライトして、待ち状態のTDMシリアル ポート送信割り込みをクリアします。	初期化の前に発生した可能性のある割り込みを解消します。
3)	IMRと0080hで論理和演算して、TDMシリアル ポート割り込みをイネーブルにします。	送信割り込みをイネーブルにします。
4)	ST1のINTMビットをクリアして、割り込みをグローバルにイネーブルにします。	CPUが応答するように、割り込みをグローバルにイネーブルにする必要があります。
5)	TCSRに0001hをライトします。	これにより、タイムスロット0をデバイスのための送信タイム スロットとして選択します。
6)	TRTAに0100hをライトします。	これにより、アドレス01hで受信するデバイスにデータを送信するように、このデバイスを設定します。さらにこのデバイスは、すべての受信データを無視するようにも設定されます。
7)	TSPCに0049hをライトしてシリアル ポートを開始させます。	これにより、シリアル ポートの送信部分がリセットから復帰して、手順1、5、6で定義された状態で動作を開始します。
8)	ハンドシェイクを実行して、受信デバイスがデータ受信の態勢にあるか検証します。	シングル デバイスのペアでは、これにより$\overline{\text{BIO}}$とXFが使用されます。複数のデバイスでは、TFRMおよびTCLKを発生するデバイスが他のすべてのデバイスに、それぞれのデバイスが応答を返すまでコマンドを転送します。
9)	TDXRに最初のデータ値をライトします(手順8で行われない場合)。	新たなデータがTDXRにライトされない場合TADDとTDATがドライブしないので、この動作によりシリアル ポート送信動作が始まります。

例9−6. TDMシリアル ポート送信割り込みサービス ルーチン

	動作	説明
1)	スタック上で変更されるコンテキストを保存します。	割り込みコードの動作コンテキストを保持します。
2)	メモリの前もって決められた位置から、新たな値をTDXRにライトします。	ISRのために新たな送信データをライトします。
3)	手順1で保存されたコンテキストを復元します。	割り込みコードの動作コンテキストを維持します。
4)	RETEでISRから復帰して、割り込みを再度、イネーブルにします。	CPUが次の割り込みに応答するように、割り込みを再度、イネーブルにする必要があります。

例9−7. TDMシリアル ポート受信初期化のルーチン

	動作	説明
1)	TSPCに0009hをライトすることによりTDMシリアル ポートをリセット、初期化します。	これにより、TDMシリアル ポートの送受信部をリセットして、シリアル ポートをTDMポートで外部TFRMおよびTCLK信号を使って動作するように設定します。
2)	IFRに0040hをライトして、待ち状態のTDMシリアル ポート送信割り込みをクリアします。	初期化の前に発生した可能性のある割り込みを解消します。
3)	IMRと0040hで論理和演算して、TDMシリアル ポート割り込みをイネーブルにします。	受信割り込みをイネーブルにします。
4)	ST1のINTMビットをクリアして、必要であれば割り込みをグローバルにイネーブルにします。	CPUが応答するように、割り込みをグローバルにイネーブルにする必要があります。
5)	TCSRに0000hをライトします。	これにより、どのタイムスロットでも送信しないように、このデバイスを設定します。
6)	TRTAに0001hをライトします。	これにより、どのデバイスにも送信しないように、このデバイスを設定します。またこのデバイスが、アドレス01hに送られるデータを受信するようにも設定します。
7)	ハンドシェイクを実行して、データ送信を受け付けることを送信デバイスに知らせます。	シングル デバイスのペアでは、これに$\overline{\text{BIO}}$とXFが使用されます。複数のデバイスでは、コマンドが転送されるのを待ってから、応答を返します。

例9−8. TDMシリアル ポート受信割り込みサービス ルーチン

	動作	説明
1)	スタック上で変更されるコンテキストを保存します。	割り込みコードの動作コンテキストを維持します。
2)	TDRRをリードしてその値を前もって決められたメモリ位置にライトします。	ISRのために新たな受信データをリードします。
3)	手順1で保存されたコンテキストを復元します。	割り込みコードの動作コンテキストを保持します。
4)	RETEでISRから復帰して、割り込みを再度イネーブルにします。	CPUが次の割り込みに応答するように、割り込みを再度イネーブルにする必要があります。

第10章

外部バス機能

本章では、メモリおよびI/Oアクセスに対する外部バスの機能および制御について説明します。'54xの外部バス機能および制御機能には、ソフトウェア ウェイト ステート、バンク スイッチング ロジック、ホールド ロジックなどが含まれます。'54xは、幅広いシステム インターフェイスをサポートします。

Topic Page

10

10.1 外部バス インターフェイス

’54xの外部インターフェイスは、データ バス、アドレス バス、制御信号から成り、外部メモリとI/Oポートをアクセスします。表10−1は、外部インターフェイスのための主要な信号を示しています。

表10−1. 主な外部インターフェイス信号

信号名称	’541 ’542, ’543, ’545, ’546	’548 ’549	説明
A0–A15†	15−0	22−0	アドレス バス
D0–D15	15−0	15−0	データ バス
$\overline{\text{MSTRB}}$	✓	✓	外部メモリ アクセス ストローブ信号
$\overline{\text{PS}}$	✓	✓	プログラム セレクト信号
$\overline{\text{DS}}$	✓	✓	データ セレクト信号
$\overline{\text{IOSTRB}}$	✓	✓	I/Oアクセス ストローブ信号
IS	✓	✓	I/Oセレクト信号
R/$\overline{\text{W}}$	✓	✓	リード/ライト信号
READY	✓	✓	外部レディ入力信号
$\overline{\text{HOLD}}$	✓	✓	ホールド入力信号
$\overline{\text{HOLDA}}$	✓	✓	$\overline{\text{HOLD}}$応答信号
$\overline{\text{MSC}}$	✓	✓	マイクロステート完了信号
$\overline{\text{IAQ}}$	✓	✓	命令捕捉用信号
$\overline{\text{IACK}}$	✓	✓	割り込み応答信号

† ’548と’549では、アドレス信号は[A0–A22]までです。

パラレル インターフェイスは、$\overline{\text{MSTRB}}$および$\overline{\text{IOSTRB}}$信号によって制御される2つの相互排他的なインターフェイスから成り立ちます。$\overline{\text{MSTRB}}$はメモリ アクセス(プログラムまたはデータ)でアクティブとなり、$\overline{\text{IOSTRB}}$はI/Oポートへのアクセスでアクティブとなります。R/$\overline{\text{W}}$信号はメモリ、I/O共にアクセスの方向を制御します。

外部レディ入力信号(READY)およびソフトウェア生成によるウェイト ステートによって、プロセッサは様々な速度のメモリおよびI/Oデバイスとインターフェイスが可能となります。低速なデバイスと通信する場合、’54xは他のデバイスが動作を完了するまで待ってから、READY信号を受け取って実行を続行します。

2つの外部メモリ バンク間をアクセスが越える時にのみ、インアクティブ ステートが必要になる場合があります。プログラマブルなバンク スイッチング ロジックは、このような状況でインアクティブ ステートの自動挿入を実現します。

ホールド モードにより外部デバイスは'54x外部バスを制御して、'54xの外部プログラム、データ、I/Oメモリ空間のリソースにアクセスできるようになります。ホールド モードには、通常モードおよびコンカレントDMAモードの2つのタイプがあります。

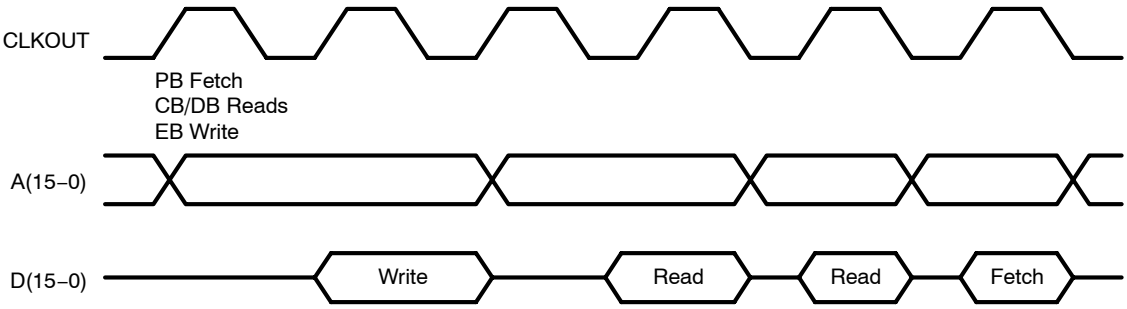
CPUが内部メモリにアドレスするとき、データ バスはハイ インピーダンスの状態になります。ただし、アドレス バスおよびメモリ セレクト信号(プログラム セレクト($\overline{\text{PS}}$)、データ セレクト($\overline{\text{DS}}$)、I/Oセレクト($\overline{\text{IS}}$))は前の状態を維持します。$\overline{\text{MSTRB}}$、$\overline{\text{IOSTRB}}$、R/$\overline{\text{W}}$、$\overline{\text{IAQ}}$、$\overline{\text{MSC}}$信号は、インアクティブのままになります。PMSTレジスタにあるアドレス可視モード(AVIS)ビットが1にセットされると$\overline{\text{IAQ}}$がアクティブとなり、外部アクセスがない時に内部プログラム アドレスはアドレス バス上に出力されます。

10.2 外部バスの優先順位

’54xのCPUには、1本のプログラム バス(PB)、3本のデータ バス(CB、DB、EB)、4本のアドレス バス(PAB、CAB、DAB、EAB)があります。CPUはそのパイプライン構造により、複数のバスに同時にアクセスできます。ただし、外部インターフェイスは1サイクルにつき1アクセスのみとなります。CPUが命令、データ メモリ オペランドのフェッチのために外部メモリまたは外部I/Oデバイスに1サイクルに2回アクセスしようとするとパイプライン コンフリクトが発生します。このパイプライン コンフリクトは、パイプラインのステージによって定義されている、予め決められた優先順位によって自動的に解決されます。

図10−1は、1サイクルの命令のフェッチと外部インターフェイス上のデータ オペランドのリード及びライトによる複数のCPUアクセスを示しています。データ アクセスは、プログラム メモリのフェッチよりも高い優先順位をもちます。プログラム メモリのフェッチは、すべてのCPUデータ アクセスが完了するまでは開始できません。

図10−1. 外部バス インターフェイスの優先順位



プログラムおよびデータが外部メモリにあり、シングル オペランドのライト命令の後にデュアル オペランドのリードまたは32ビット オペランドのリードが続く場合、パイプライン コンフリクトが発生します。次の命令シーケンスは上で述べたパイプライン コンフリクトの例を示しています。

```
ST     T,*AR6        ;Smem write operation
LD     *AR4+,A       ;Xmem and Ymem read operation
||MAC  *AR5+, B
```

パイプライン処理およびコンフリクトの詳細は、第7章パイプラインを参照してください。

10.3 外部バス制御

’54xでは、ウェイト ステート発生器およびバンク スイッチング ロジックの2つのユニットが外部バスを制御します。これらのユニットは、ソフトウェア ウェイト ステート レジスタ(SWWSR)およびバンク スイッチング制御レジスタ(BSCR)によって制御されます。

10.3.1 ウェイト ステート発生器

ソフトウェア プログラマブルのウェイト ステート発生器は、外部バス サイクルに最大7マシン サイクルまでのウェイト ステートを挿入でき、’54xが低速の外部デバイスとインターフェイスする場合に有効です。7サイクルを上回るウェイト ステートが必要なデバイスは、ハードウェアREADY入力によりインターフェイスが可能となります。すべての外部アクセスが0ウェイト ステートに設定される場合は、ウェイト ステート発生器に供給する内部クロックを停止します。内部クロックを供給停止することで、デバイスを低消費電力で動作させることができます。

ソフトウェア プログラマブルのウェイト ステート発生器は、データ空間のアドレス0028hにメモリ マッピングされた16ビットのソフトウェア ウェイト ステート レジスタ(SWWSR)によって制御されます。

このレジスタにより制御可能なプログラムおよびデータ空間は、それぞれ2つの32Kワード ブロックから、I/O空間は64Kワード ブロックひとつから成り立ちます。SWWSRではこれらの各ブロックに対応した3ビットのフィールドが割り当てられています。図10−2はこれらのフィールドを示しており、表10−2でそれを説明しています。表10−3は’548及び’549の場合についての説明です。

SWWSRの3ビット フィールドの値は、対応する空間およびアドレス領域への各アクセスに挿入されるウェイト ステートの数を指定します。ウェイト ステートを加えない最小値は、0(000_2)で、7(111_2)は、ウェイト ステートの最大値となります。

図10−2. ソフトウェア ウェイト ステート レジスタ(SWWSR)図

15	14−12	11−9	8−6	5−3	2−0
Reserved/XPA†	I/O	Data	Data	Program	Program
R	R/W	R/W	R/W	R/W	R/W

† XPAビットは’548と’549のみ

表10−2. ソフトウェア ウェイト ステート レジスタ(SWWSR)ビット概要

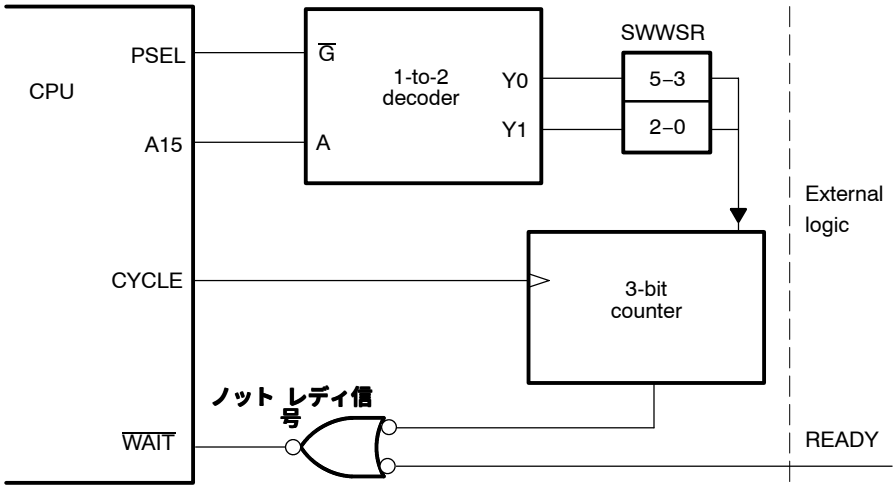
ビット	名称	リセット値	ファンクション
15	予約	0	予約。'548と'549ではこのビットはプログラム フィールドの処理を変更します(表10−3を参照)。
14−12	I/O	1	I/O空間。フィールド値(0−7)はI/O空間0000−FFFFhのウェイト ステート数に対応します。
11−9	データ	1	データ空間。フィールド値(0−7)はデータ空間8000−FFFFhのウェイト ステート数に対応します。
8−6	データ	1	データ空間。フィールド値(0−7)はデータ空間0000−7FFFhのウェイト ステート数に対応します。
5−3	プログラム	1	プログラム空間。フィールド値(0−7)はプログラム空間8000−FFFFhのウェイト ステート数に対応します。
2−0	プログラム	1	プログラム空間。フィールド値(0−7)はプログラム空間0000−7FFFhのウェイト ステート数に対応します。

表10−3. TMS320C548及び549ソフトウェア ウェイト ステート レジスタ(SWWSR)ビット概要

ビット	名称	リセット値	ファンクション
15	XPA	0	拡張プログラム アドレス制御。プログラム フィールドで選択されるアドレス範囲を選択します。
14−12	I/O	1	I/O空間。フィールド値(0−7)はI/O空間0000−FFFFhのウェイト ステート数に対応します。
11−9	データ	1	データ空間。フィールド値(0−7)はデータ空間8000−FFFFhのウェイト ステート数に対応します。
8−6	データ	1	データ空間。フィールド値(0−7)はデータ空間0000−7FFFhのウェイト ステート数に対応します。
5−3	プログラム	1	プログラム空間。フィールド値(0−7)は、次のプログラム空間のウェイト ステート数に対応します。 ❑ XPA=0: xx8000−xxFFFFh ❑ XPA=1: 400000h−7FFFFF
2−0	プログラム	1	プログラム空間。フィールド値(0−7)は、次のプログラム空間のウェイト ステート数に対応します。 ❑ XPA=0: xx0000−xx7FFFh ❑ XPA=1: 000000−3FFFFFh

図10−3は、外部プログラム空間へのウェイト ステート発生器ロジックのブロック図です。外部プログラムのアクセスがデコードされると、SWWSRの適切なフィールドがカウンタにロードされます。そのフィールドが000でない場合、ノット レディ信号がCPUに送られて3bitのウェイト ステート カウンタが起動されます。ノット レディの状態は、カウンタが0にデクリメントされるまで、かつ外部READY入力がハイにセットされるまで維持されます。外部READYおよびウェイト ステートREADYはともに論理和演算の処理がなされ、CPU $\overline{\text{WAIT}}$信号を発生します。READY入力は、CLKOUTの立ち下がりエッジでサンプリングされます。最低でも2サイクルのソフトウェア ウェイト ステートがプログラムされているときのみ、プロセッサはREADYを検出します。

図10−3. ソフトウェア ウェイト ステート発生器のブロック図



リセット時は、SWWSRのすべてのフィールドが外部アクセスのためのウェイト ステート最大値である111_2(SWWSR=7FFFh)に設定されます。これにより、プロセッサの初期化中にCPUは低速な外部メモリとインターフェイスできます。

10.3.2 バンク スイッチング ロジック

'54xは、ターン オフ時間が必要なメモリに対し、プログラマブル バンク スイッチング ロジックにより、外部ウェイト ステートを必要とせずに、外部メモリ バンクを切り替えることができます。バンク スイッチング ロジックは、プログラムまたはデータ空間内のメモリ バンク境界をアクセスがまたぐとき、自動的に1サイクルを挿入します。

バンク スイッチングはバンク スイッチング制御レジスタ(BSCR)によって定義されます。このレジスタはアドレス0029hにメモリ マッピングされています。図10−4は、BSCRおよびそのフィールドを表わしており、表10−4でその内容を説明します。

図10−4. **バンク スイッチング 制御レジスタ**(BSCR)**図**

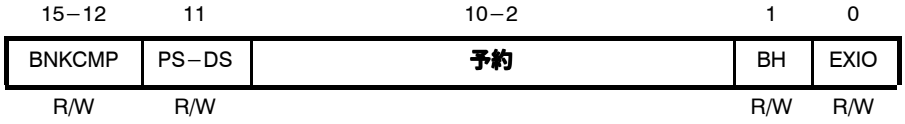


表10−4. **バンク スイッチング 制御レジスタ**(BSCR)**ビット概要**

ビット	名称	リセット値	ファンクション
15−12	BNKCMP	−	**バンク比較。この設定で、外部メモリ バンクのサイズが決まります。**BNKCMP**は、アドレスの上位**4**ビットをマスクするために使用されます。たとえば、**BNKCMP=1111_2**のとき、上位**4**ビット(ビット**12**〜**15**)の比較が行なわれ、結果としてバンク サイズは**4K**ワードになります。**4K**ワードから**64K**ワードまでのサイズが設定可能です。**10−9**ページの表**10−5**に、**BNKCMP**とアドレス範囲の関係を示します。**
11	PS−DS	−	**プログラム リード-データ リード アクセス。プログラム リードに続くデータ リード、またはデータ リードに続くプログラム リードが連続するアクセスの間に追加の**1**サイクルを挿入します。** PS−DS = 0 **この機能による追加サイクルを挿入しません。** PS−DS = 1 **プログラム リードに続くデータ リード、またはデータ リードに続くプログラム リードが連続するアクセスの間に追加の**1**サイクルを挿入します。**
10−2	**予約**	−	**これらのビットは予約されています。**
1	BH	0	**バス ホルダ。バス ホルダ機能を制御します。** BH = 0 **バス ホルダをディセーブルにします。** BH = 1 **バス ホルダをイネーブルにします。データ バス、**D(15−0)**が直前のロジック レベルに保持されます。**
0	EXIO	0	**外部バス インタフェースをオフ。**EXIO**ビットは外部バス オフ機能を制御します。** EXIO = 0 **外部バス オフ機能をディセーブルにします。** EXIO = 1 **外部バス オフ機能をイネーブルします。実行中のバス サイクル完了後に、アドレス バス、データ バス、制御信号がインアクティブになります。**10−9**ページの表**10−6**に、外部バスがディセーブルされた時点での、各信号の状態一覧を示します。また、この状態では**PMST**レジスタの**DROM、MP/$\overline{MC}$、OVLY**各ビット、および**ST1**レジスタの**HM**ビットが、変更できなくなります。**

BNKCMPの値、比較対象となるアドレス ビット、バンク サイズの関係を、表10−5に示します。表に記載されていない値をBNKCMPに設定することはできません。表10−6には、外部バス インターフェイスがディセーブルされた(EXIO=1)時点での、各信号の状態を示します。

表10−5. BNKCMPとバンクサイズの関係

BNKCMP				比較されるMSB	バンク サイズ (16ビット ワード)
Bit 15	Bit 14	Bit 13	Bit 12		
0	0	0	0	None	64K
1	0	0	0	15	32K
1	1	0	0	15−14	16K
1	1	1	0	15−13	8K
1	1	1	1	15−12	4K

表10−6. EXIO=1時の信号状態

信号	状態	信号	状態
A(15−0)	以前の状態	R/$\overline{W}$	ハイ レベル
D(15−0)	ハイ インピーダンス	$\overline{MSC}$	ハイ レベル
$\overline{PS}$, $\overline{DS}$, $\overline{IS}$	ハイ レベル	$\overline{IAQ}$	ハイ レベル
$\overline{MSTRB}$, $\overline{IOSTRB}$	ハイ レベル		

BSCRレジスタのEXIOおよびBHビットは、外部アドレスおよびデータ バスを制御します。これらのビットはリセット後、0に設定されます。外部メモリにまったくアクセスしないか、ごくまれにしかアクセスしない場合、EXIOおよびBHを1にセットすることにより、電力消費を低減できます。

BSCRレジスタのEXIOビットを１にセットすると、CPUはST1レジスタのHMビットを変更できません。またPMSTレジスタのDROM，MP/$\overline{MC}$およびOVLYを変えてメモリ マップを変更することもできません。

’54xは、プログラムおよびデータ空間でのリードやライトの処理に使われた直前のアドレスのMSBs(BNKCMPフィールドで定義される)を保持する内部レジスタを持ちます。現在のリードに使用されているアドレスのMSBが、この内部レジスタに入っているMSBと一致しない場合は、バンク スウィッチング サイクルが挿入され、$\overline{\text{MSTRB}}$(メモリ ストローブ)信号は1CLKOUTサイクルの間、インアクティブとなります。この追加サイクルの間に、アドレス バスは新たなアドレスに切り替わります。内部レジスタの内容が、リードのためのカレント アドレスのMSBsと入れ替わり、現在のリードに使用されているアドレスのMSBが、そのレジスタのビットと一致する場合は、通常のリード サイクルとなります。

同じメモリ バンクから繰り返しリードが実行される場合、追加サイクルは挿入されません。異なるメモリ バンクからリードが実行されるときは、追加サイクルを挿入することでメモリ コンフリクトを回避します。追加サイクルはリード メモリ アクセスの後､別のリード メモリ アクセスが続く場合のみに挿入されます。この機能は、BNKCMPを0にクリアすることでディセーブルになります。

’54xのバンク スイッチング メカニズムは、次の場合に追加の1サイクルを挿入します。

- ❑ プログラム メモリ リードの後、続けて別のメモリ バンクからプログラム メモリまたはデータ メモリのリードを行なう場合。
- ❑ データ メモリ リードの後、続けて別のメモリ バンクからプログラム メモリまたはデータ メモリのリードを行なう場合。
- ❑ プログラム メモリ リードの後、続けて別のページからプログラム メモリ リードを行う場合。(’548、’549のみ)。
- ❑ PS−DSビットが1にセットされているときに、プログラム メモリ リードの後、続けてデータ メモリのリードを行う場合。
- ❑ PS−DSビットが1にセットされているとき、データ メモリのリードの後、続けてプログラム メモリのリードを行う場合。

図10−5は、メモリ バンクが切り替わるときのインアクティブ サイクルの追加を示しています。

図10−5. **メモリ リード間のバンク スイッチング**

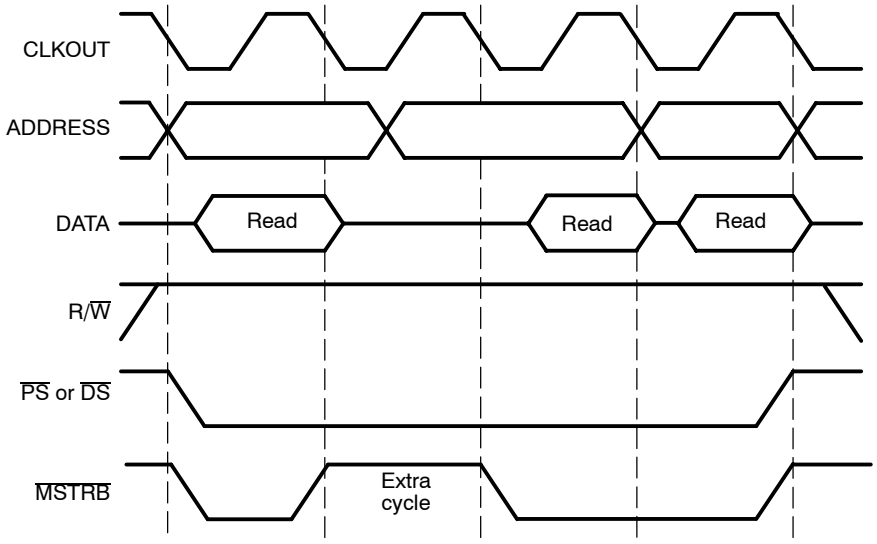
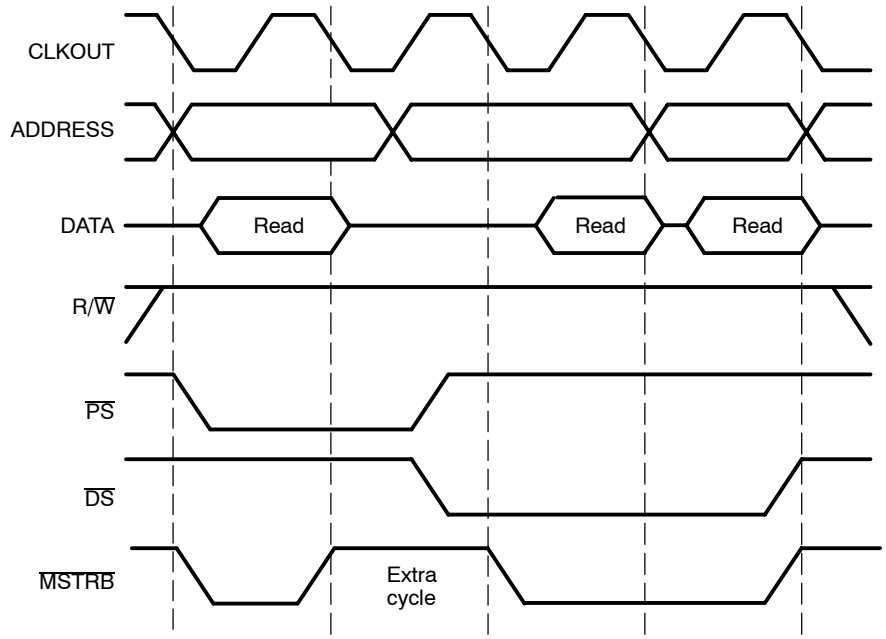


図10−6**は、連続するプログラムのリードとデータのリード間への追加サイクル挿入を示しています。**

図10−6. **プログラム空間およびデータ空間のバンク スイッチング**



10.4 外部バス インターフェイス タイミング

すべての外部バス アクセスは、CLKOUTのサイクルで完了します。1CLKOUTサイクルは、CLKOUTの立ち下がりエッジから次の立ち下がりエッジとして定義します。いくつかの外部バス アクセス、たとえばメモリ ライト、I/Oライト、またはI/Oリードは、ウェイト ステートなしで2サイクルかかります。メモリ リードは1サイクルですが、メモリ ライトにメモリ リードが続く場合、またはその逆の場合は、メモリ リードに半サイクル余分にかかります。下記のサブセクションは、特に説明がない限り0ウェイト ステート アクセスについての説明になります。

10.4.1 メモリ アクセス タイミング

$\overline{\text{MSTRB}}$信号は、リード ライト アクセスのアクティブ部分でローになります。メモリ アクセスのアクティブ部分は、1CLKOUTサイクル以上、維持します。さらに、ライトのアクティブ部分の前後に遷移のためのCLKOUTサイクルが発生します。このサイクルの間は、次のようになります。

- ❑ $\overline{\text{MSTRB}}$はハイ。
- ❑ R/$\overline{\text{W}}$は必要に応じて、CLKOUTの立ち上がりエッジで変化します。
- ❑ 次のケースでは、アドレスがCLKOUTの立ち上がりエッジで変化します。それ以外のケースでは、アドレスはCLKOUTの立ち下がりエッジで変化します。
  - ■ 前のCLKOUTサイクルがメモリ ライトのアクティブ部分だった場合。
  - ■ メモリ リードにメモリライトが続く場合。
  - ■ メモリ リードにI/Oライトが続く場合。
  - ■ メモリ リードにI/Oリードが続く場合。
- ❑ アドレスの変化とともに$\overline{\text{PS}}$、$\overline{\text{DS}}$、$\overline{\text{IS}}$のいずれかが変化します。

図10−7は、ウェイト ステートなしのリード-リード-ライトのシーケンスを示しています。データ リードは可能な限りサイクルの後ろで行われ、アドレスが確定してからのアクセス時間が最大になるように配慮しています。外部ライトは2サイクルかかりますが、外部インターフェイスへのアクセスが進行中でなければ、それは内部的には1サイクルで実行されます。これは、処理のスループットを可能な限り最大レベルに維持するためです。

タイミング図は、次のような内容を示しています。

- ❑ 同一バンクからの連続したリードは、1サイクル アクセス。
- ❑ 連続したリードの間は、$\overline{\text{MSTRB}}$はローを維持します。
- ❑ リードからライトへ遷移する時、$\overline{\text{MSTRB}}$が1サイクルの間、ハイになり、その間にアドレスを確定させ、さらにR/$\overline{\text{W}}$信号が変化します。

図10−7. リード-リード-ライトのメモリ インターフェイス動作

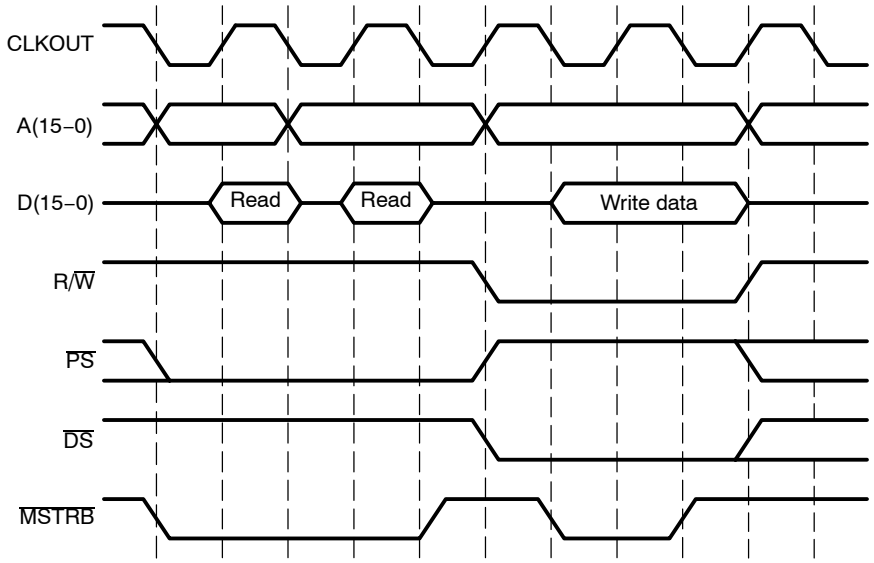
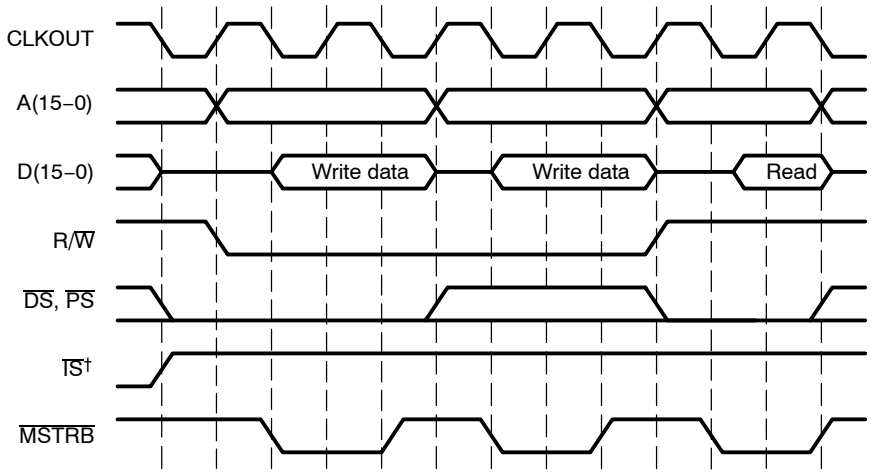


図10−8は、ウェイト ステートなしのライト-ライト-リードのシーケンスを示しています。ライト時のアドレスおよびデータは、$\overline{\text{MSTRB}}$が変化した後、約1.5サイクル確定します。

タイミング図は、次のような内容を示しています。

- $\overline{\text{MSTRB}}$はライト サイクルごとの終わりにハイになりメモリをディセーブルにして、アドレスおよびR/$\overline{\text{W}}$信号を変化させます。
- 各ライトには2サイクルかかります。
- ライトに続くリードは1.5サイクルかかります。

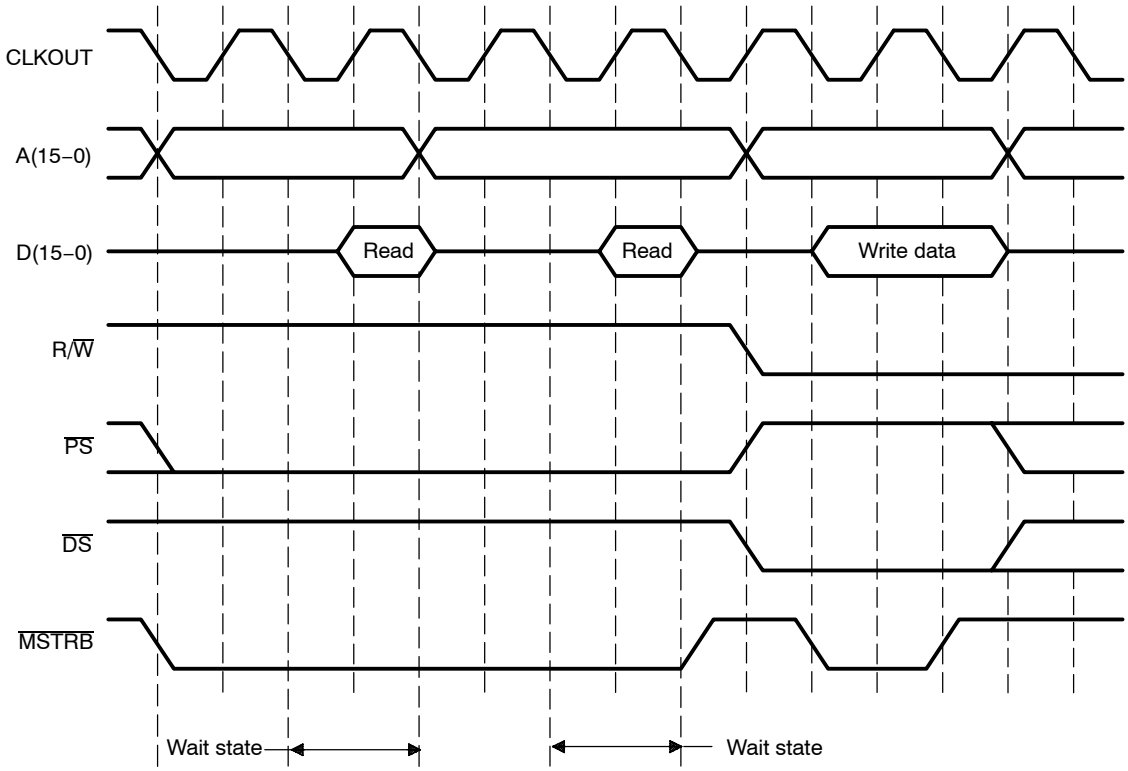
図10−8. ライト-ライト-リードのメモリ インターフェイス動作



† 最初のメモリ ライトの前に、I/Oライトがあったものとします。

図10−9は、1ウェイト ステートをともなうリード-リード-ライトシーケンスを示しています。リードは通常1サイクルですが、ウェイト ステートにより2サイクルになります。

図10−9. リード-リード-ライトのメモリ インタフェース動作(プログラム空間ウェイト ステート)



10.4.2 I/Oアクセス タイミング

I/Oアクセスの、リードとライトはウェイト ステートなしで2サイクルかかります。他の点では、このアクセスのタイミングはメモリ アクセスのタイミングと同じになります。このサイクルの間に、アドレスはCLKOUTの立ち下がりエッジで変わりますが、メモリ アクセスに続くI/Oアクセスの場合は例外となります。$\overline{\text{IOSTRB}}$は、CLKOUTサイクルの立ち上がりエッジから次の立ち上がりエッジまでローになります。

図10−10は、ウェイト ステートなしのリード-ライト-リードシーケンスを示しています。I/Oアクセスでは、リードとライトは2サイクル以上かかります。

タイミング図は、次のような内容を示しています。

- 各I/Oアクセスは2サイクルかかります。
- $\overline{\text{IOSTRB}}$は、各アクセスの終わりにハイになりアドレスを固定にしてR/$\overline{\text{W}}$信号が変化します。

図10−10. リード-ライト-リードのパラレルI/Oインターフェイス動作

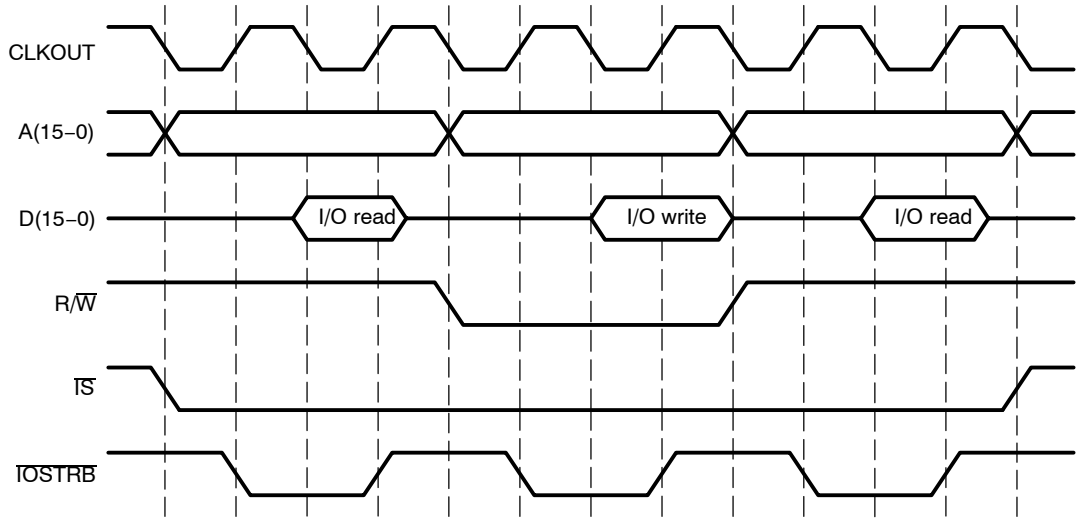
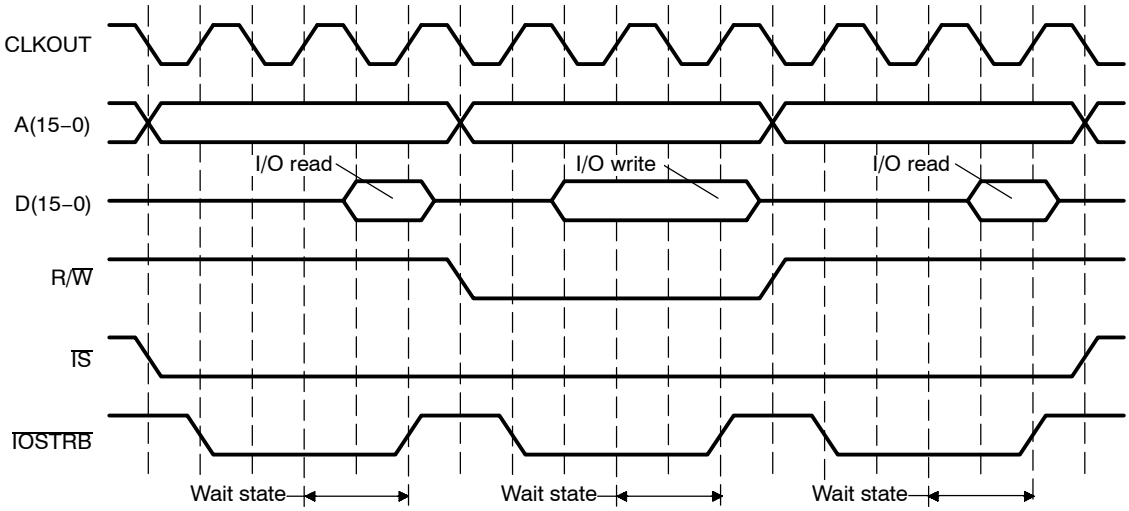


図10−11は1ウェイト ステートのアクセスをともなう同一I/O空間アクセスを示しています。各リードおよびライト アクセスは、追加サイクルによって拡張されます。

図10−11. リード-ライト-リードのパラレルI/O動作(I/O空間ウェイト ステート)



10.4.3 メモリおよびI/Oアクセス タイミング

図10−12から図10−19は、外部インターフェイス バス上における、メモリのリードおよびライトとI/Oのリードおよびライト間の様々なタイミングを示しています。

タイミング図は、次のような内容を示しています。

- メモリのリードまたはライトに続くI/Oのリードまたはライトには、2.5サイクル以上かかります。
- I/Oのリードまたはライトに続くメモリのリードには、2サイクルかかります。
- I/Oのリードまたはライトに続くメモリのライトには、2.5サイクルかかります。

図10−12. **メモリ リードおよび**I/O**ライト**

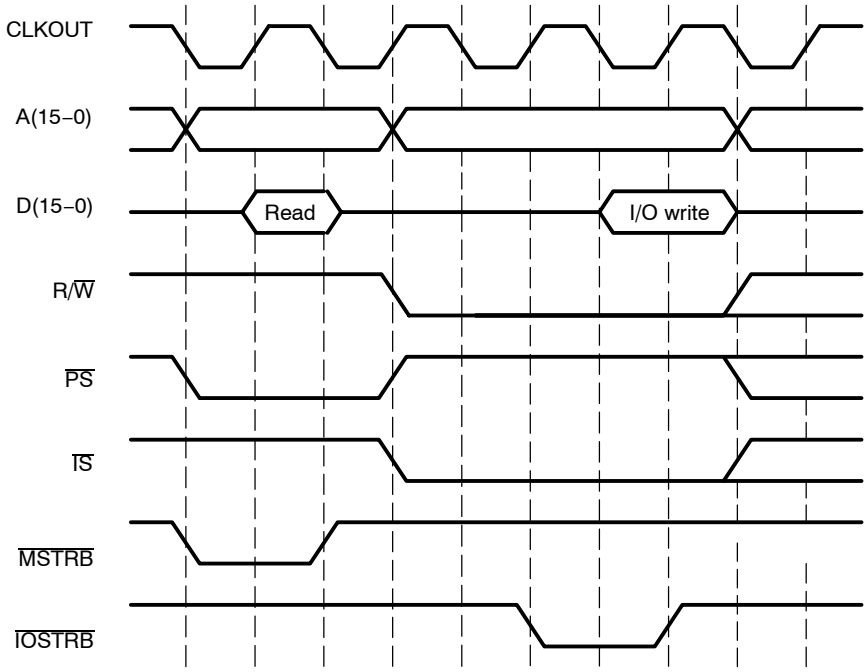


図10−13. **メモリ リードおよび**I/O**リード**

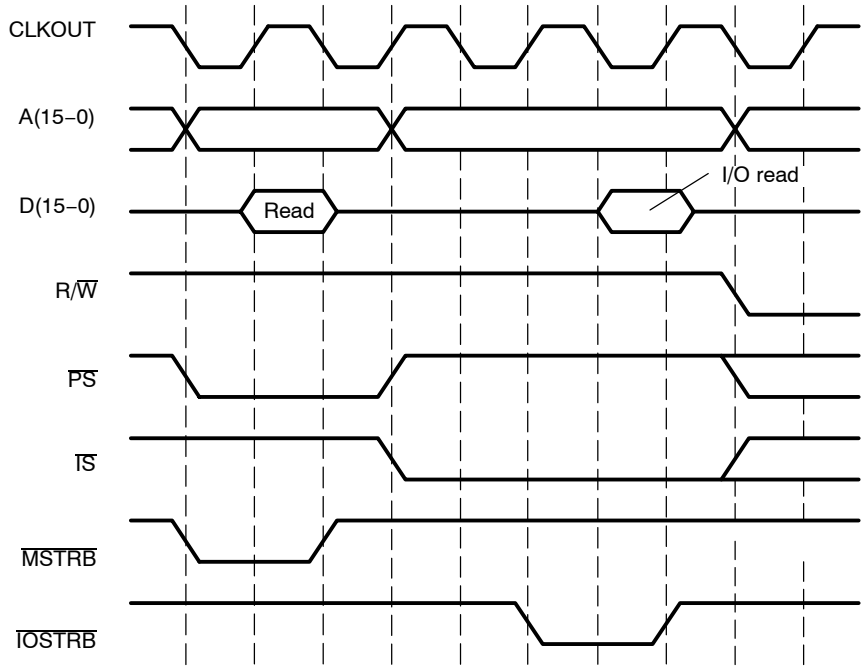


図10−14. **メモリ ライトおよび**I/O**ライト**

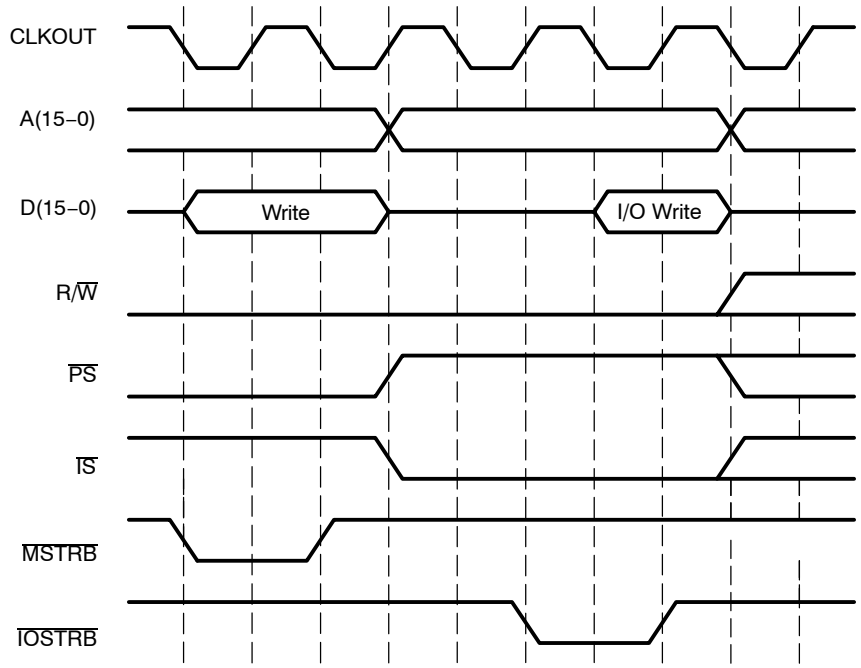
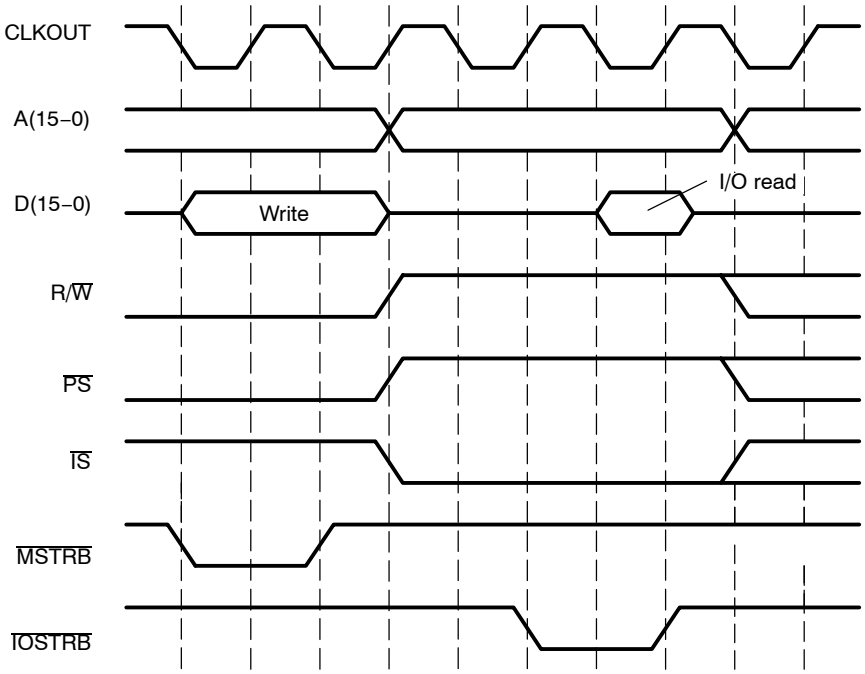


図10−15. **メモリ ライトおよび**I/O**リード**



10

図10−16. I/O**ライトおよびメモリ ライト**

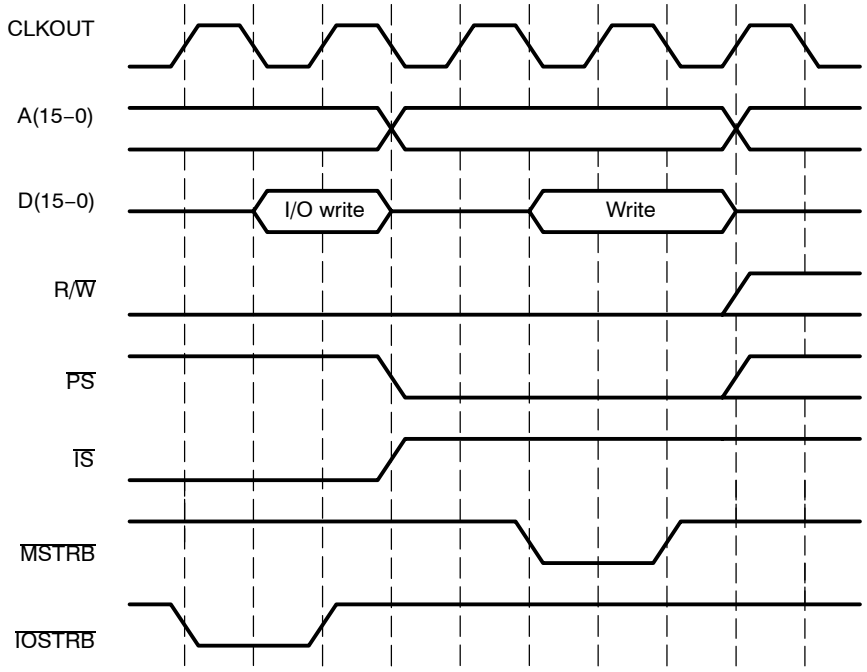


図10−17. I/O**ライトおよびメモリ リード**

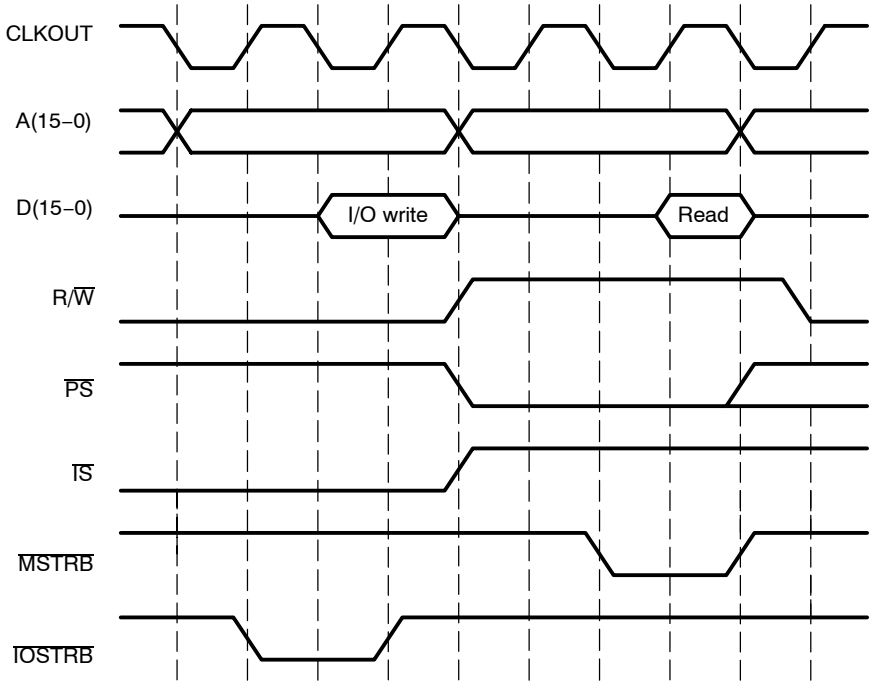


図10−18. I/O**リードおよびメモリ ライト**

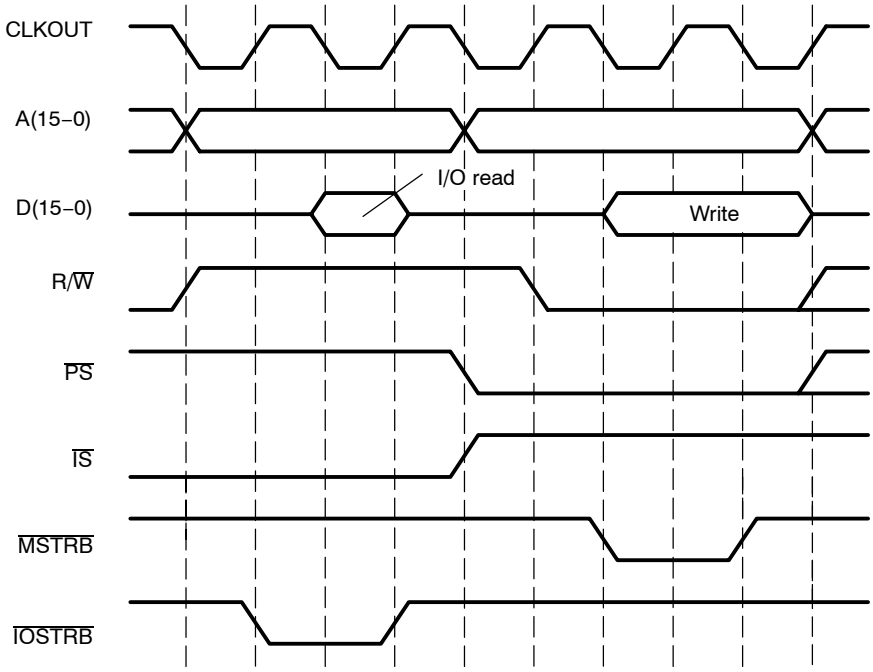
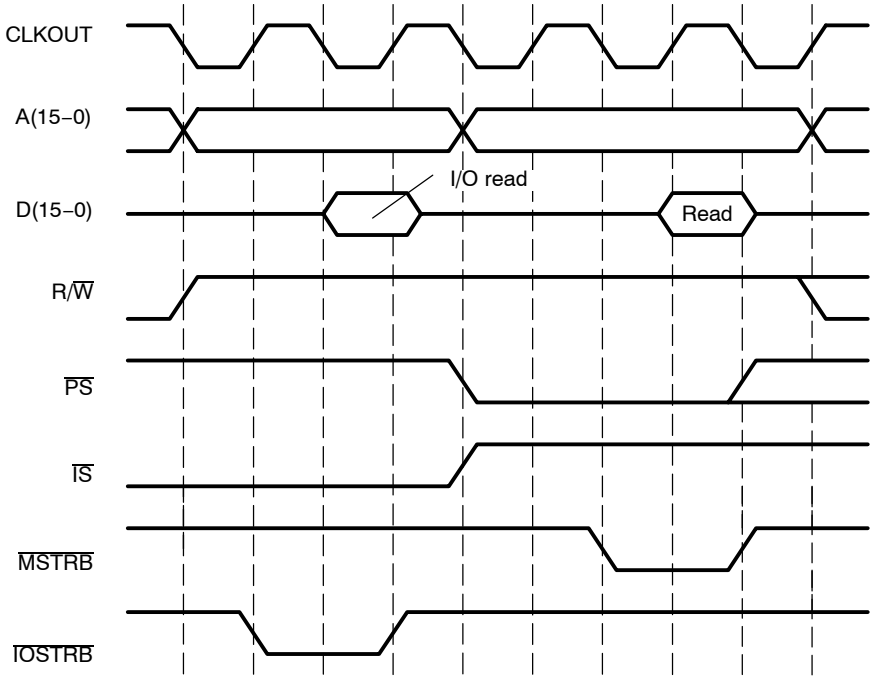


図10−19. I/O**リードおよびメモリ リード**



10

10.5 起動アクセス シーケンス

'54xでは、IDLE1、IDLE2、IDLE3、リセットのいずれかのモードになるとき、またはいずれかを終了するときに、アクティブ状態とインアクティブ状態の間を移行します。

IDLE1またはIDLE2モードになるとき、またはいずれかのモードを終了するときには、CPUおよびオンチップ ペリフェラルに対するクロックはアクティブのままなので、特別な配慮は必要ありません。(6−43ページ、サブセクション6.11パワーダウン モードを参照)しかし、IDLE3またはリセットのモードになるとき、またはいずれかのモードを終了するときには特別な配慮が必要です。

- リセット:ハードウェアの初期化が行われます。
- IDLE3:デバイスは、CPUにもオンチップ ペリフェラルにもクロックが供給されていないインアクティブな状態からアクティブな状態へ移行します。

10.5.1 リセット

図10−20は、外部バスのリセット シーケンスを示しています。適切なリセット動作のために、$\overline{RS}$信号は最低2CLKOUTサイクルの間、ローにする必要があります。しかし、電源投入時およびIDLE3のパワーダウン モード時ではリセット信号を2CLKOUTサイクル以上必要とします。詳しくは、6−44ページ、セクション6.11パワーダウン モードを参照してください。

'54xがリセットを認識すると、CPUは、プログラムの実行を終了してプログラム カウンタを強制的にFF80hにします。アドレス バスは$\overline{RS}$信号がローの間、FF80hにドライブされます。

デバイスは外部バスに対して、次の3つの段階をへて、リセット状態になります。

1) $\overline{RS}$信号をローにした4サイクル後、$\overline{PS}$、$\overline{MSTRB}$、$\overline{IAQ}$がハイにドライブされます。
2) $\overline{RS}$信号をローにした5サイクル後、R/$\overline{W}$がハイにドライブされ、データ バスがハイ インピーダンスの状態になり、アドレス バスはFF80hにドライブされます。
3) デバイスは、リセットの状態になります。

リセットが解除されると、プログラムがプログラム メモリのFF80hから実行されます。図10−20に示すように、MP/$\overline{MC}$信号の状態にかかわらず、命令捕捉用信号($\overline{IAQ}$)および割り込み応答信号($\overline{IACK}$)はアクティブになります。

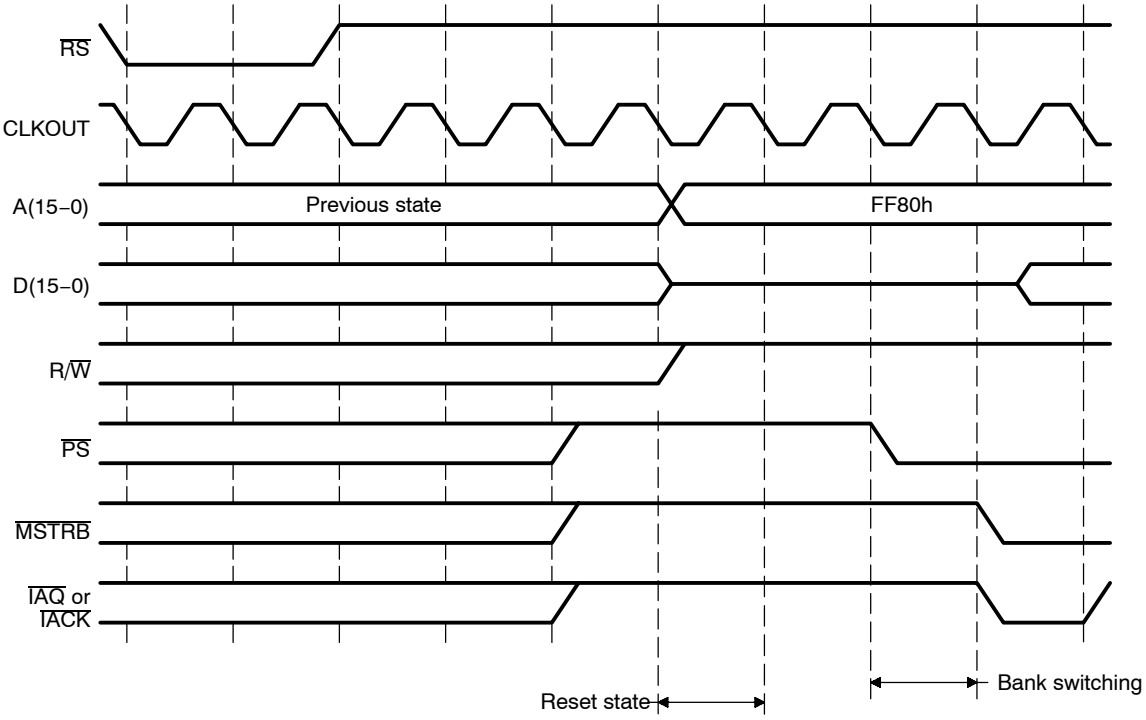
デバイスは外部バスに対して、次の3つの段階をへて、アクティブ状態になります。

1) $\overline{RS}$信号をハイにした5サイクル後、$\overline{PS}$がローにドライブされます。

2) $\overline{RS}$信号をハイにした6サイクル後、$\overline{MSTRB}$および$\overline{IACK}$がローにドライブされます。

3) その半サイクル後に、デバイスはデータ リードの準備ができ、デバイスはアクティブ状態になります。

図10−20. 外部バス リセット シーケンス



注： 1) $\overline{RS}$信号は非同期入力で、クロック サイクルにおける任意の場所で発生させることができます。指定されたタイミングをみたしていれば、上に示されたようなシーケンスとなります。その他の場合、1クロック サイクル分遅延することがあります。

2) リセットの間、データ バスはハイ インピーダンスになり、制御信号はインアクティブとなります。

3) リセット ベクタのフェッチには7サイクルのウェイト ステートが挿入されます。

4) リセット後の最初のアクセス時に、バンク スウィッチング サイクルが挿入されます。

10.5.2 IDLE3

IDLE 3命令の実行によって、IDLE3のパワーダウン モードが起動されます。このパワーダウン モードでは、PLLが完全に停止して電力消費を低減させます。このモードでは、入力クロックを入力しつづけても余計な電力消費はありません。これは、'54x内部のトランスファゲートが入力クロックを内部ロジックから切り離すためです。'54xがIDLE3を終了してプログラムを実行する前にPLLが起動してロック状態である必要があります。このパワーダウン モードは、外部割り込みピン$\overline{\text{INTn}}$、$\overline{\text{NMI}}$、$\overline{\text{RS}}$を所定のシーケンスでアクティブにすることで終了します。

表10−7は、$\overline{\text{INTn}}$および$\overline{\text{NMI}}$信号によるIDLE3のウェイクアップ時間を示しています。これらの時間は、ハードウェア コンフィギュラブルなPLLのために定義されます。ソフトウェア プログラマブルなPLLのための時間については、セクション8.4.2を参照してください。割り込みピンがローになると、内部カウンタは入力クロック サイクルをカウントします。カウンタにロードされる初期値はPLLの逓倍数に応じ、40 MIPS '54xではカウント ダウン時間を50 µsより確実に大きくするように設定されます。

表10−7. PLL乗数によるカウント ダウン時間(40MHz動作時)

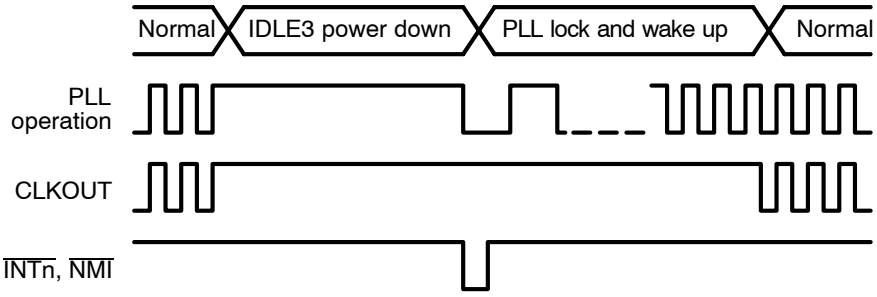
カウンタ開始値	PLL逓倍数	相当するCLKOUTサイクル(N)	40MHz時のカウンタ ダウンタイム(µs)
2048	1	2048	51.2
2048	1.5	3072	76.8
1024	2	2048	51.2
1024	2.5	2560	64
1024	3	3072	76.8
512	4	2048	51.2
512	4.5	2304	57.6
512	5	2560	64

表10−7のカウント ダウン時間は、PLLがロックされたときにCLKOUT周波数が40MHzになるような入力クロック周波数の場合に適用されます。カウンタが0までカウントした後、ロックされたPLLからの出力が内部ロジックに送られます。

外部割り込みのロー パルス(最小期間10ns)によって、'54xがIDLE3からウェイク アップします(図10−21参照)。ロックされたPLLクロックは、Nサイクルの後にCPUに供給されます。その後、'54xがIDLE3から復帰する前に、3サイクルかかります。ただし、'54xは割り込みの同期のために余計に2サイクルを必要としません。これは、IDLE中の割り込みパルスが割り込み同期回路を初期化するからで、'54xのウェイク アップの直後に割り込みを検出することができます。

リセットを使ってIDLE3からウェイク アップする場合は、カウンタは使用しません。PLLからの出力は内部ロジックに直接送り込まれ、CLKOUTピンがドライブされます。PLLおよびCLKOUTが安定するロック アップ時間は50 μs以上です。したがって、この50 μsのロック アップ時間の間にリセット ピンをローに維持する必要があります。このリセット信号の保持により'54xが不安定なクロックを使って処理を開始することがなくなります。

図10−21. IDLE3からのウェイク アップ シーケンス



10.6 ホールド モード

’54xは、オフチップ プログラム、データ、I/O空間への外部プロセッサからのダイレクト メモリ アクセス(DMA)等に対応しています。$\overline{\text{HOLD}}$および$\overline{\text{HOLDA}}$の2つの信号によって、外部デバイスは’54xのバスを制御可能になります。’54xは、外部デバイスから$\overline{\text{HOLD}}$信号を受信したことに応答して、$\overline{\text{HOLDA}}$をローにします。’54xはホールド モードになり、外部アドレス バス、データ バス、制御信号をハイ インピーダンスにします。

ST1レジスタにあるホールド モード(HM)ステータス ビットは、次に示すホールド機能の動作モードを決定します。

- 通常モードでは、$\overline{\text{HOLD}}$信号がローの間はプログラム実行を停止します。
- コンカレントDMAモードでは、プログラムが内部メモリ(ROMまたはRAM)にあれば、その場所から継続して実行できます。

HM＝1のとき、’54xは通常モードで動作します。HM＝0のとき、’54xはコンカレントDMAモードで動作します。このモードでは、プログラムが外部メモリからの場合、または外部アクセスが実行された場合に、’54xはプログラム実行を停止します。しかし、プログラムが内部メモリで外部アクセスが実行されない場合は、’54xは外部的にホールド状態になりますが、内部ではプログラム実行が継続します。このように、外部DMA動作を実行しながら、プログラム実行を続けることができ、効率的に処理を行えます。

’54xがHM＝0でホールド状態にあり、内部的に実行しているプログラム中で、外部アクセス、または外部アドレスへの分岐が必要な場合、プログラム実行は$\overline{\text{HOLD}}$が解除されるまで停止します。また、外部バスを使う必要のある繰り返し命令を、ホールド発生時にHM＝0で実行している場合は、実行中のバス サイクルのあとにホールド状態になります。繰り返し命令をHM＝1の状態で実行しているときにホールドが発生した場合は、内部または外部アクセスのいずれについても、’54xは実行中のバス サイクルを実行した後、ホールド状態となります。リセット時、HMは0にクリアされます。HMはSSBX命令によってセット、またはRSBX命令によってリセットされます。

$\overline{\text{HOLD}}$は、割り込みとしては扱われません。IDLE1を実行中はHMビットの値にかかわらず、ホールドは受け入れられます。IDLE2またはIDLE3の実行中には、HMビットの値にかかわらず、ホールドは受け入れられません。IDLE1を実行中に$\overline{\text{HOLD}}$信号を受信すると、CPUはIDLE1モードを継続しますが、外部バスおよび制御信号はハイ インピーダンスになります。

図10−22は、$\overline{\text{HOLD}}$および$\overline{\text{HOLDA}}$のタイミングを示しています。$\overline{\text{HOLD}}$は、CLKOUTがローになる前にセットアップ時間を満たすと、バスおよび制御信号がハイ インピーダンスになるには、3サイクル以上が必要になります。$\overline{\text{HOLD}}$は外部非同期入力で、ラッチされません。外部デバイスは必ず$\overline{\text{HOLD}}$をローに維持しなければなりません。外部デバイスは、’54xから$\overline{\text{HOLDA}}$信号を受信して、ホールド状態になった事を判定できます。

’54xはマルチサイクル命令の実行中に、ホールド信号を受信すると命令が完了した後、バスはハイ インピーダンスになります。ウェイト ステートによりマルチサイクルとなる命令でも同様です。

$\overline{\text{HOLD}}$が解除された後、プログラム実行を停止した次の命令からプログラム実行が回復します。$\overline{\text{HOLDA}}$は$\overline{\text{HOLD}}$の解除にともない、解除されます(図10−22参照)。セットアップ時間が満たされていれば、’54xはバスおよび制御信号を確定するまで2マシン サイクル(HM=0)または3マシン サイクル(HM=1)必要とします。

10.6.1 ホールド時の割り込み

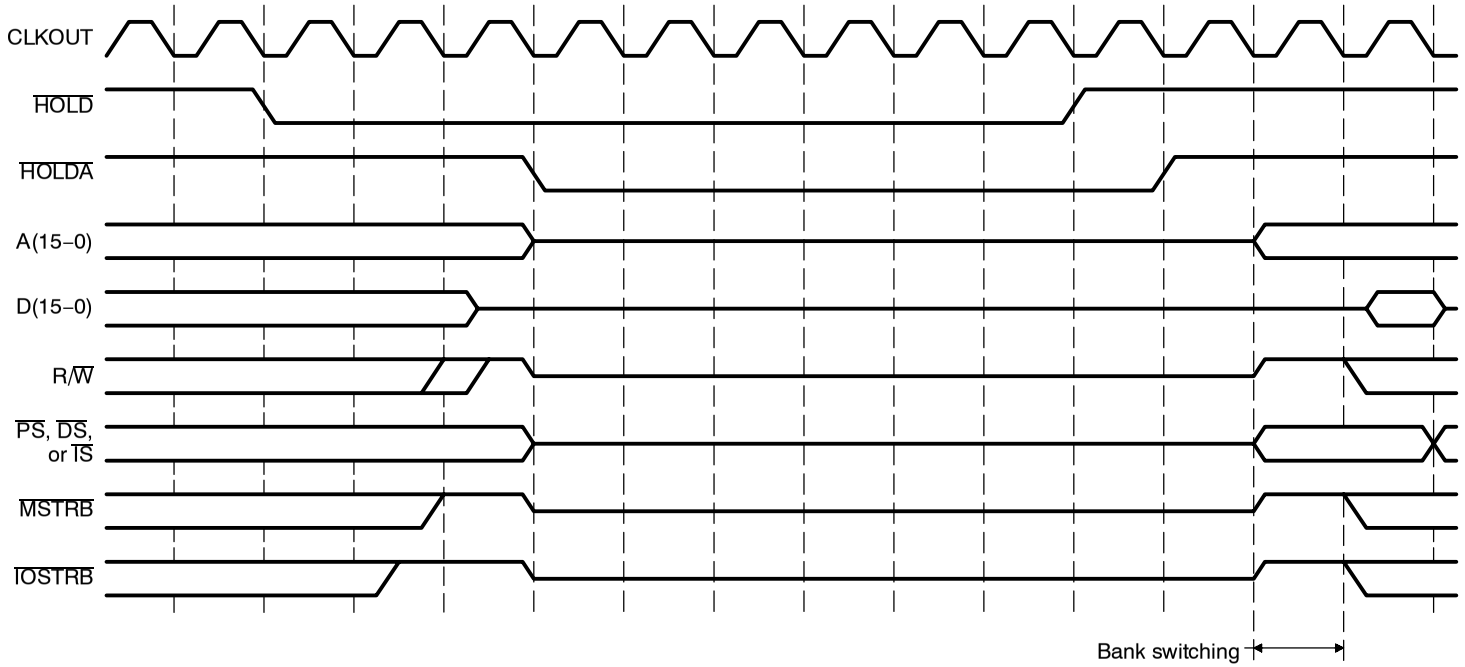
HM=1で$\overline{\text{HOLD}}$がアクティブの間は、すべての割り込みは受け付けられません。この期間に割り込みを受信すると、割り込みはラッチされますが、未処理のまま残ります。したがって、$\overline{\text{HOLD}}$はどの割り込みフラグまたはレジスタにも影響しません。HM=0のとき、割り込みは通常の通り受け付けられます。

10.6.2 ホールドおよびリセット

$\overline{\text{RS}}$信号がアクティブのとき$\overline{\text{HOLD}}$がローになると、内部的には通常のリセット動作を行いますが、すべてのバスおよび制御ラインはハイ インピーダンスとなり、$\overline{\text{HOLDA}}$信号がアクティブになります。(図10−23(a)(b)参照)。しかし、$\overline{\text{HOLD}}$および$\overline{\text{HOLDA}}$がアクティブとなった後、$\overline{\text{RS}}$信号がアクティブになる場合は、’54xはリセットされデータ バス、アドレス バス、制御信号はハイ インピーダンスの状態を維持します。(図10−23(c)(d)参照)。

$\overline{\text{RS}}$信号がアクティブのとき$\overline{\text{HOLD}}$が解除されると、$\overline{\text{HOLDA}}$が応答してホールド モードは解除され、アドレス、データ、制御ピンはリセットの状態に応じてドライブされます(図10−23(a)(d)参照)。しかし、$\overline{\text{HOLD}}$および$\overline{\text{HOLDA}}$がアクティブのとき$\overline{\text{RS}}$信号が解除される場合、デバイスの動作はMP/$\overline{\text{MC}}$ピンの状態に従います。MP/$\overline{\text{MC}}$ピンがハイの場合、ホールド モードが終了すると’54xはリセット ベクタのフェッチを開始します。MP/$\overline{\text{MC}}$がローの場合、’54xはホールド モード終了の前に外部アクセスを必要としない限り、リセット ベクタを内部的にフェッチして処理を続けます。図10−23(b)(c)は、$\overline{\text{HOLD}}$および$\overline{\text{HOLDA}}$がアクティブのとき$\overline{\text{RS}}$信号が解除される例を示しています。

図10−22. $\overline{\text{HOLD}}$と$\overline{\text{HOLDA}}$のHM=0における相互関係



注： 1) このタイミング チャートは、HM=0のときのホールド モードの様子を示しています。HM=1の時は、$\overline{\text{HOLDA}}$がインアクティブになるのに、さらに1サイクルが必要となります

2) ホールド モードを解除した直後の最初のサイクルは、バンク スウィッチング サイクルとなります

図10−23. $\overline{\text{HOLD}}$と$\overline{\text{RS}}$の相互関係

(a)

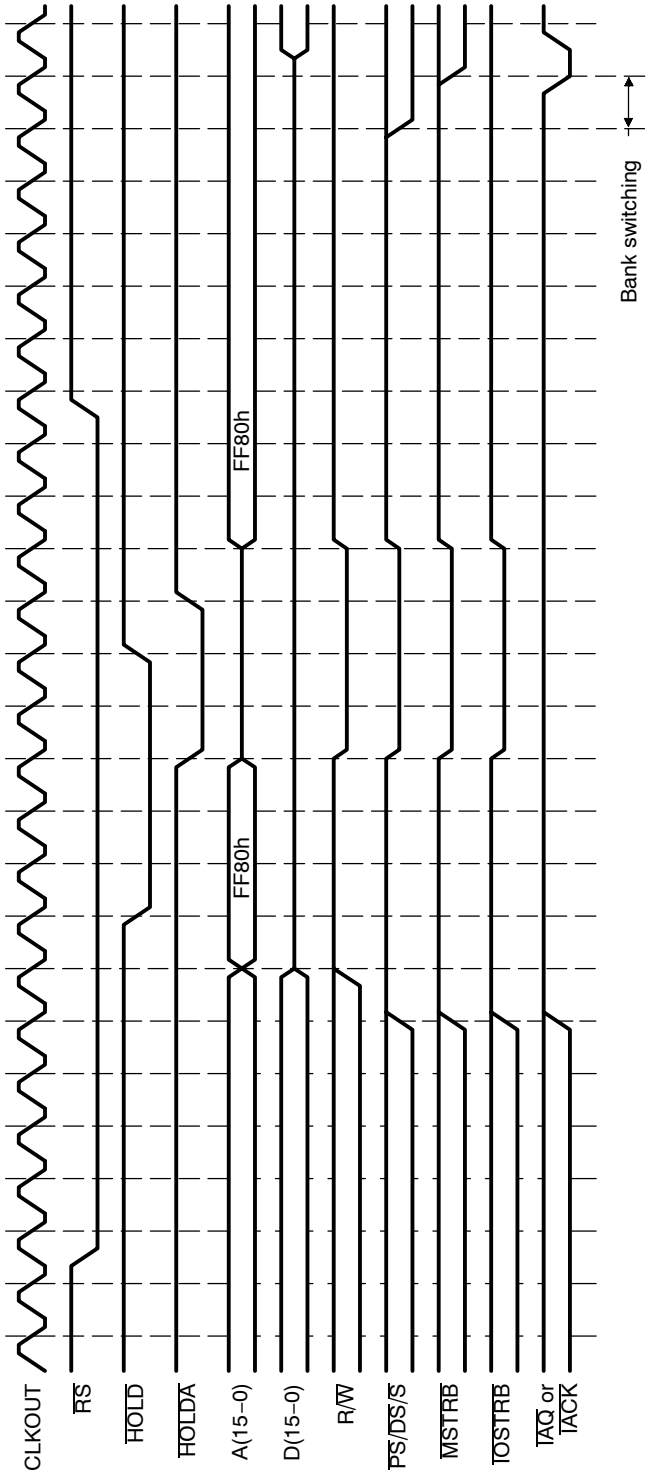


図10−23. $\overline{\text{HOLD}}$とRSの相互関係(続き)

(b)

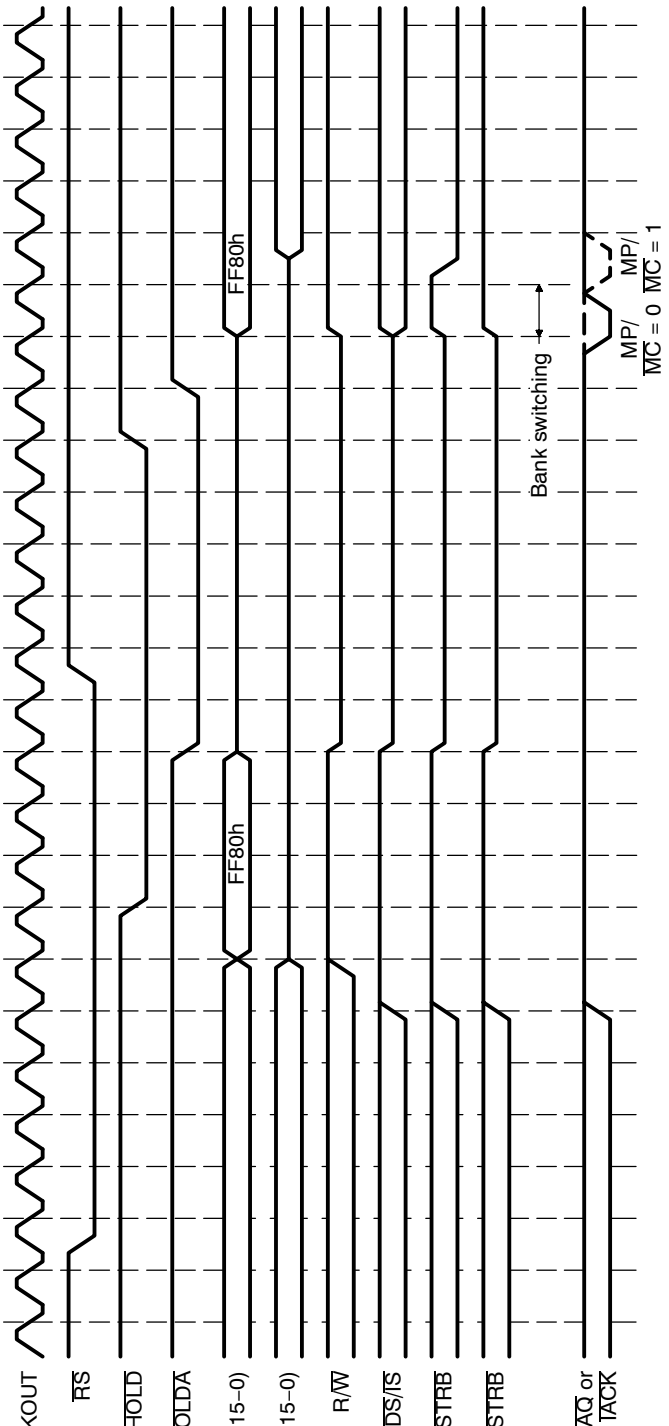
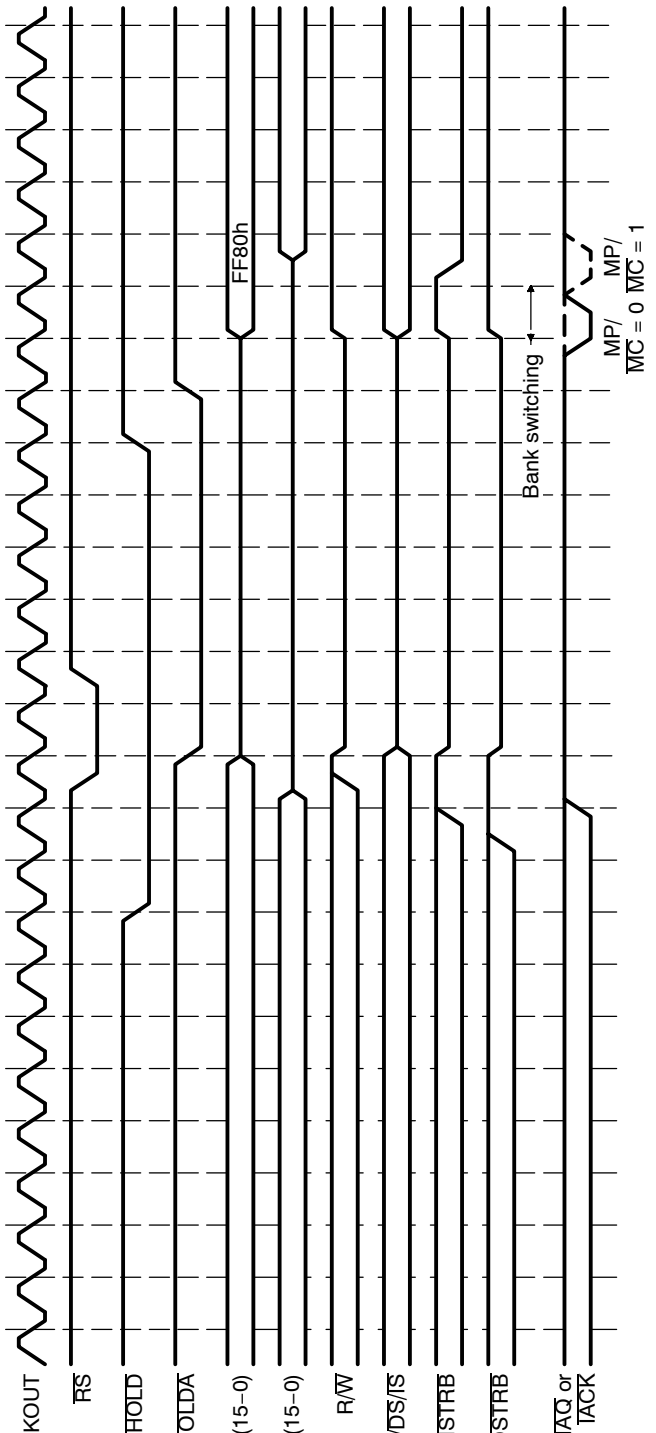


図10−23. $\overline{\text{HOLD}}$と$\overline{\text{RS}}$の相互関係(続き)

(c)



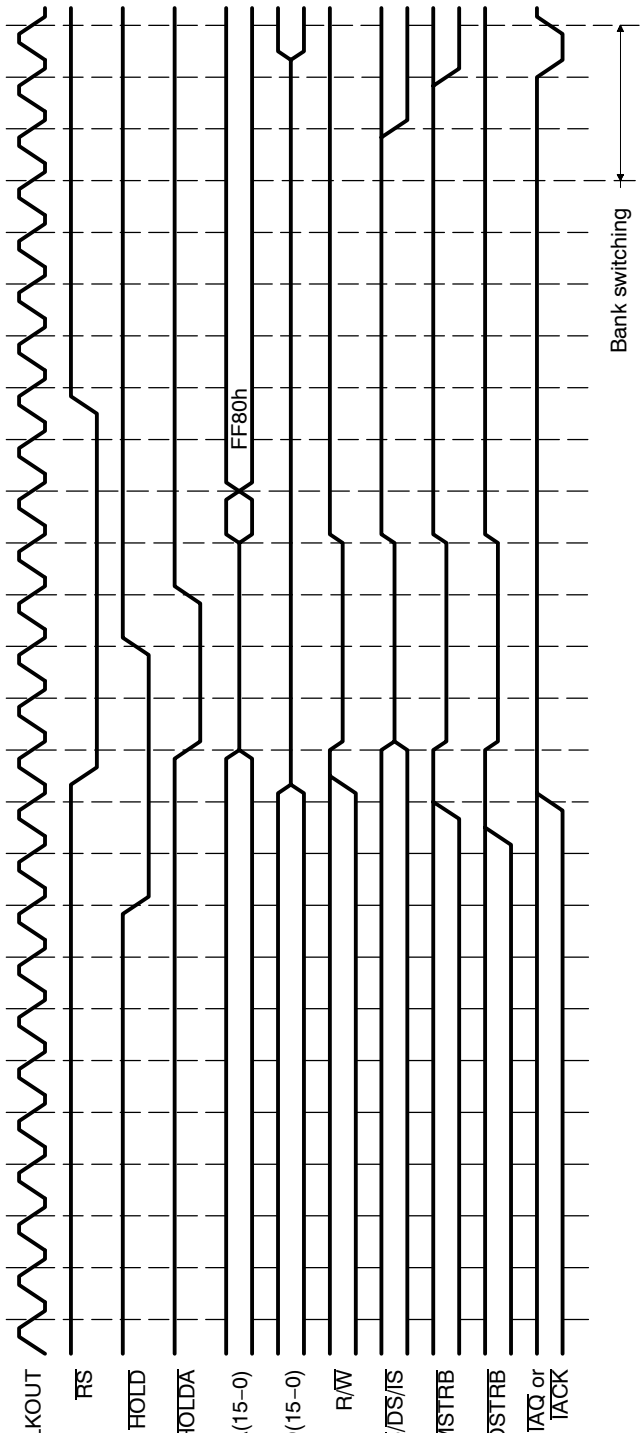


図10−23. HOLDとRSの相互関係(続き)

(d)

付録A

CPU及びペリフェラルのレジスタ群

付録Aは'54x CPU及びペリフェラルのレジスタのビット領域を示します。以下のテーブルにレジスタ領域で使用している用語を記します。

Term	Definition	Term	Definition
ARP	Auxiliary register pointer	OVB	Overflow flag B
ASM	Accumulator shift mode	OVLY	RAM overlay
AVIS	Address visibility mode	OVM	Overflow mode
BH	Bus holder	PCM	Pulse code modulation mode
BNKCMP	Bank compare	PLLCOUNT	PLL counter value
BOB	Byte ordering bit	PLLDIV	PLL divider
BRAF	Block-repeat active flag	PLLMUL	PLL multiplier
BRE	Autobuffering receive enable	PLLNDIV	PLL clock generator select
BRINT1, BRINT0	BSP receive interrupt	PLLON/OFF	PLL on/off
BXE	Autobuffering transmit enable	PLLSTATUS	PLL status
BXINT1, BXINT0	BSP transmit interrupt	PSC	Timer prescaler counter
C	Carry	PS−DS	Program read−data read access
CH0−CH7	TDM channel 0−7	RA0−RA7	TDM receive address
CLKDV	Internal transmit clock division factor	Res, Resvd	Reserved
CLKOFF	CLOCKOUT off	RH	Receive buffer half received
CLKP	Clock polarity	RINT1, RINT0	Serial port receive interrupt
CMPT	Compatibility mode	RRDY	Receive ready
CPL	Compiler mode	$\overline{\text{RRST}}$	Receive reset
C16	Dual 16-bit/Double-precision arithmetic mode	RSRFULL	Receive shift register full

A

Term	Definition	Term	Definition
DLB	Digital loopback mode	SMOD	Shared-access mode
DP	Data page pointer	SMUL	Saturation on multiplication
DROM	Data ROM	SST	Saturation on store
DSPINT	DSP interrupt	SXM	Sign-extension mode
EXIO	External bus interface off	TA0−TA7	TDM transmit address
FE	Format extension	TC	Test/control flag
FIG	Frame ignore	TDDR	Timer divide-down ratio
FO	Format	TDM	TDM mode
FRCT	Fractional mode	TINT	Internal timer interrupt
FSM	Frame sync mode	TRB	Timer reload
FSP	Frame sync polarity	TRINT	TDM receive interrupt
HALTR	Autobuffering receive halt	TSS	Timer stop status
HALTX	Autobuffering transmit halt	TXINT	TDM transmit interrupt
HINT	’54x-to-Host interrupt	TXM	Transmit mode
HM	Hold mode	XF	External flag (XF) status
HPINT	HPI interrupt	XH	Transmit buffer half transmitted
INTM	Interrupt mask	XINT1, XINT0	Serial port transmit interrupt
INT0−INT3	External user interrupt #0−3	XPA	Extended program address control
IPTR	Interrupt vector pointer	XRDY	Transmit ready
MCM	Clock mode	$\overline{\text{XRST}}$	Transmitter reset
MP/$\overline{\text{MC}}$	Microprocessor/microcontroller	$\overline{\text{XSREMPTY}}$	Transmit shift register empty
OVA	Overflow flag A		

図A−1. Bank-Switching Control Register (BSCR) Diagram

15−12	11	10−2	1	0
BNKCMP	PS−DS	予約	BH	EXIO

図A−2. BSP Control Extension Register (BSPCE) Diagram

15	14	13	12	11	10	9	8	7	6	5	4−0
HALTR	RH	BRE	HALTX	XH	BXE	PCM	FIG	FE	CLKP	FSP	CLKDV

図A−3. Clock Mode Register (CLKMD) Diagram (対象デバイスは付録Dを参照)

15−12	11	10−3	2	1	0
PLLMUL	PLLDIV	PLLCOUNT	PLLON/OFF	PLLNDIV	PLLSTATUS

図A−4. HPI Control Register (HPIC) Diagram

15−12	11	10	9	8	7−4	3	2	1	0
X†	HINT	DSPINT‡	SMOD	BOB	X†	HINT	DSPINT‡	SMOD	BOB

† 不定な値が読まれます。自由な値を書き込みできます。
‡ '0'が読まれます。

A

図A−5. Interrupt Flag Register (IFR) Diagram

(a) ’541 IFR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	Resvd	INT3	XINT1	RINT1	XINT0	RINT0	TINT	INT2	INT1	INT0

(b) ’542 IFR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	HPINT	INT3	TXINT	TRINT	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(c) ’543 IFR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	Resvd	INT3	TXINT	TRINT	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(d) ’545 IFR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	HPINT	INT3	XINT1	RINT1	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(e) ’546 IFR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	Resvd	INT3	XINT1	RINT1	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(f) ’548 IFR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	BXINT1	BRINT1	HPINT	INT3	TXINT	TRINT	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(g) ’549 IFR

15−14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	BM INT1	BMINT0	BXINT1	BRINT1	HINT	$\overline{\text{INT3}}$	TXNT	TRNT	BXINT0	TRINT0	TINT	$\overline{\text{INT2}}$	$\overline{\text{INT1}}$	$\overline{\text{INT0}}$

図A−6. Interrupt Mask Register (IMR) Diagram

(a) ’541 IMR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	Resvd	INT3	XINT1	RINT1	XINT0	RINT0	TINT	INT2	INT1	INT0

(b) ’542 IMR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	HPINT	INT3	TXINT	TRINT	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(c) ’543 IMR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	Resvd	INT3	TXINT	TRINT	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(d) ’545 IMR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	HPINT	INT3	XINT1	RINT1	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(e) ’546 IMR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	Resvd	Resvd	Resvd	INT3	XINT1	RINT1	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(f) ’548 IMR

15−12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	BXINT1	BRINT1	HPINT	INT3	TXINT	TRINT	BXINT0	BRINT0	TINT	INT2	INT1	INT0

(g) ’549 IMR

15−14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Resvd	BM INT1	BMINT0	BXINT1	BRINT1	HINT	$\overline{\text{INT3}}$	TXNT	TRNT	BXINT0	TRINT0	TINT	$\overline{\text{INT2}}$	$\overline{\text{INT1}}$	$\overline{\text{INT0}}$

図A−7. Processor Mode Status Register (PMST) Diagram

15−7	6	5	4	3	2	1	0
IPTR	MP/$\overline{\text{MC}}$	OVLY	AVIS	DROM	CLKOFF	SMUL†	SST†

† **対象デバイスのみ。それ以外のデバイスでは予約。**

図A−8. Serial Port Control Register (SPC) Diagram

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Free	Soft	RSRFULL	XSREMPTY	XRDY	RRDY	IN1	IN0	$\overline{\text{RRST}}$	$\overline{\text{XRST}}$	TXM	MCM	FSM	FO	DLB	Res

図A−9. Software Wait-State Register (SWWSR) Diagram

15	14−12	11−9	8−6	5−3	2−0
Reserved/XPA†	I/O	Data	Data	Program	Program

† '548/'549**のみ**XPA**ビット。それ以外のデバイスは予約。**

図A−10. Status Register 0 (ST0) Diagram

15−13	12	11	10	9	8−0
ARP	TC	C	OVA	OVB	DP

図A−11. Status Register 1 (ST1) Diagram

15	14	13	12	11	10	9	8	7	6	5	4−0
BRAF	CPL	XF	HM	INTM	0	OVM	SXM	C16	FRCT	CMPT	ASM

図A−12. TDM Channel Select Register (TCSR) Diagram

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

注： X=Don't care.

図A−13. TDM Receive Address Register (TRAD) Diagram

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X2	X1	X0	S2	S1	S0	A7	A6	A5	A4	A3	A2	A1	A0

注： X=Don't care.

図A−14. TDM Receive/Transmit Address Register (TRTA) Diagram

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0

図A−15. TDM Serial Port Control Register (TSPC) Diagram

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Free	Soft	X	X	XRDY	RRDY	IN1	IN0	RRST	XRST	TXM	MCM	X	0	0	TDM

注： X=Don't care.

図A−16. Timer Control Register (TCR) Diagram

15−12	11	10	9−6	5	4	3−0
Reserved	Soft	Free	PSC	TRB	TSS	TDDR

付録B

XDS510エミュレータ使用の設計について

この付録では、テキサス インスツルメンツXDS510エミュレータを使用しながらIEEE−1149.1に準拠する設計を行うための補足説明、およびXDS510ケーブル(製品番号2617698−0001)についての説明を記載しています。このケーブルは、胴体にJTAG 3/5Vのマークのラベルがついていれば、標準の3Vおよび5Vのターゲットシステム電源入力をサポートします。

本書でJTAGという言葉を使用する場合は、IEEE−1149.1に基づくTIスキャン ベースのエミュレーションを意味します。

IEEE−1149.1の詳細については、IEEEカスタマ サービスまでお問い合わせください。

所在地: IEEEカスタマ サービス
445 Hoes Lane, PO Box 1331
Piscataway, NJ 08855−1331

電話: **米国およびカナダ**;(800)678−IEEE
米国およびカナダ以外;(908)981−1393

FAX: (908)981−9667 Telex: 833233

Topic **Page**

B

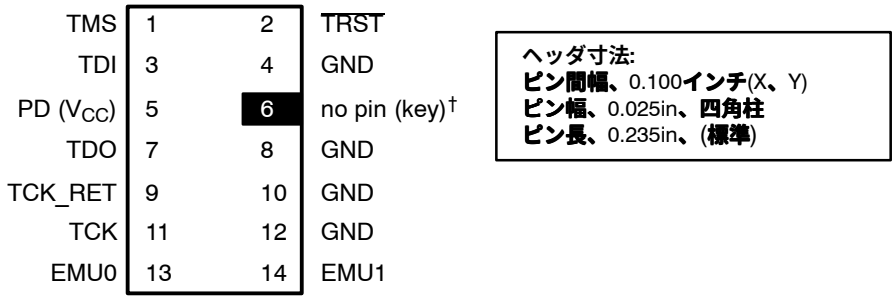
B.1 ターゲット システムのエミュレータ コネクタの設計(14ピン ヘッダ)

JTAGターゲットデバイスは、専用エミュレーションポートによるエミュレーションをサポートしています。このポートは、エミュレータが直接アクセスして、IEEE−1149.1が規定する機能のスーパーセットのエミュレーションを実現します。エミュレータと通信するには、ターゲット システムに図B−1に示す接続がされた14ピン ヘッダ(7ピンを2列)がなければならず、表B−1は、エミュレーション信号について説明しています。

他のヘッダも使用できますが、デュポン コネクタ システムズの、次の型番の、ポストフードなしストレートヘッダを推奨します。

- ❑ 65610−114
- ❑ 65611−114
- ❑ 67996−114
- ❑ 67997−114

図B−1. 14ピン ヘッダ信号およびヘッダ寸法



† ケーブル コネクタの対応するメス位置は逆向き接続にならないように埋めてありますが、第6ピンはケーブルの中でグランドに接地されています。本付録の配線図を参照してください。

表B−1. 14ピン ヘッダ信号の説明

シグナル	説明	エミュレータ ステート†	ターゲット ステート†
EMU0	エミュレーションピン0	I	I/O
EMU1	エミュレーションピン1	I	I/O
GND	グランド		
PD(V_{CC})	ターゲットの検出。エミュレーション ケーブルが接続されており、ターゲット システムの電源が投入されていることを示します。PDはターゲット システムのVccに接続しなければなりません。	I	O
TCK	テスト クロック。TCKは、エミュレーション ケーブル ポッドからの10.368MHzクロック ソースです。この信号を使って、システム テスト クロックをドライブできます。	O	I
TCK_RET	テスト クロック リターン。エミュレータへのテスト クロック入力。TCKをバッファまたはバッファなしで接続できます。	I	O
TDI	テスト データ入力	O	I
TDO	テスト データ出力	I	O
TMS	テスト モード選択	O	I
$\overline{TRST}$‡	テスト リセット	O	I

† I = input、O = output

‡ 内部でプルダウンされている為、$\overline{TRST}$にプルアップ抵抗を接続しないで下さい。
低ノイズの環境では、$\overline{TRST}$はフロートのままにできます。高ノイズの環境では、プルダウン抵抗の追加が必要となることもあります。(抵抗のサイズは電流値によって決めます。)

B

B.2 バス プロトコル

IEEE−1149.1仕様は、テスト アクセス ポート(TAP)バス スレーブ デバイスの条件、および一定のルールを規定しています(下記参照)。

- ❑ TMSおよびTDI入力は、デバイスのTCK信号の立ち上がりエッジでサンプルされる。
- ❑ TDO出力は、デバイスのTCK信号の立ち下がりエッジでクロックと同期される。

これらのデバイスがディジ チェイン接続されている場合は、デバイスのTDOには次のデバイスのTDIシグナルの前に約半TCKサイクルのセットアップ時間があります。このタイミング方式は、TDOおよびTDIの両方が同じTCKエッジからタイミングが取られている場合に発生する競合条件を最小化します。このタイミング方式の短所は、TCK周波数を高く出来ないことです。

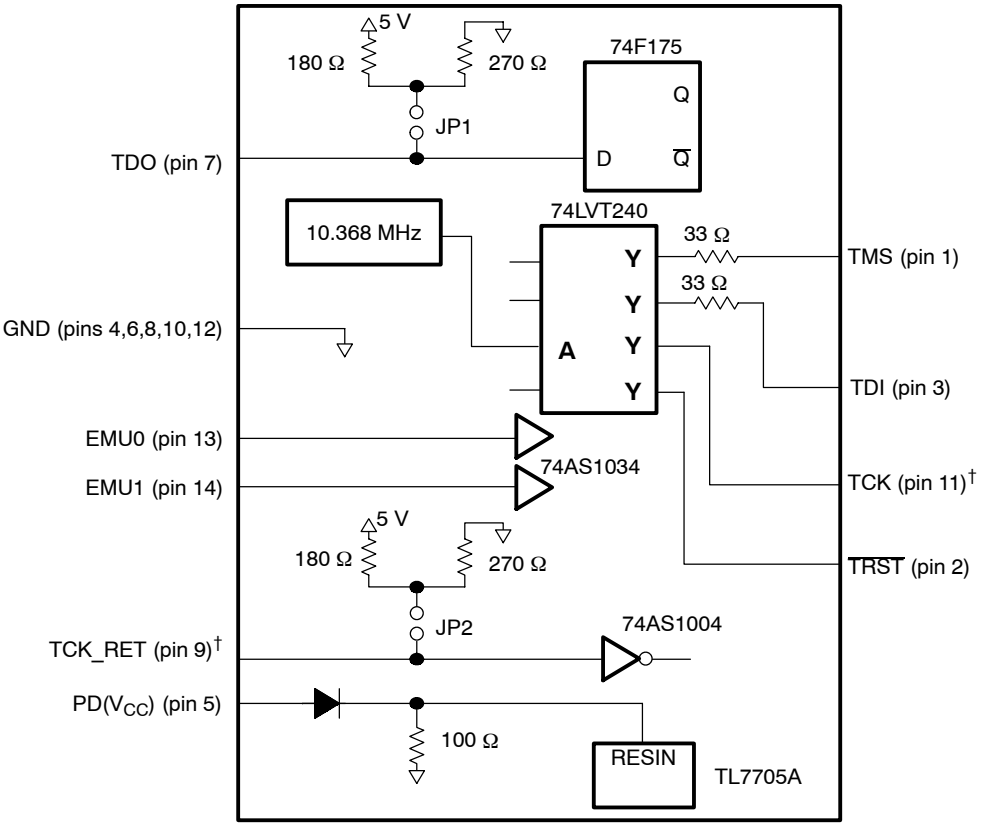
IEEE−1149.1仕様は、バス マスタ(エミュレータ)デバイスのルールは規定していません。代わりに、デバイスがバス マスタにバス スレーブ コンパチブルのタイミングを要求すると規定しています。XDS510は、バス スレーブの規定に合うタイミングを供給しています。

B.3 エミュレータ ケーブル ポッド

図B−2は、エミュレータ ケーブル ポッドの一部を示しています。ポッドの機能には以下のものがあります。

- TDOおよびTCK_RETは、アプリケーションが要求する場合は、ポッド内部で並列にターミネイトできます。デフォルトでは、これらの信号はターミネイトされていません。
- TCKは、74LVT240デバイスによってドライブされます。高電流のドライブ(32mA−I_{OL}/I_{OH})なので、この信号は並列終端することも可能です。TCKがTCK_RETに接続される場合は、ポッドの並列ターミネータを使用できます。
- TMSおよびTDIは、IEEE−1149.1のバス スレーブ デバイス タイミング規定に準じて、TCK_RETの立ち下がりエッジから発生できます。
- TMSおよびTDIは、信号反射を減少させる為直列にターミネイトしています。
- 10.368MHzテスト クロック ソースが供給されます。さらに独自のテスト クロックも供給できる柔軟性があります。

図B−2. エミュレータ ケーブル ポッド インタフェース



† エミュレータ ポッドは、TCK_RETを内部同期のためのクロック ソースとして使用します。TCKは、オプションのターゲット システム テスト クロック ソースとして供給されます。

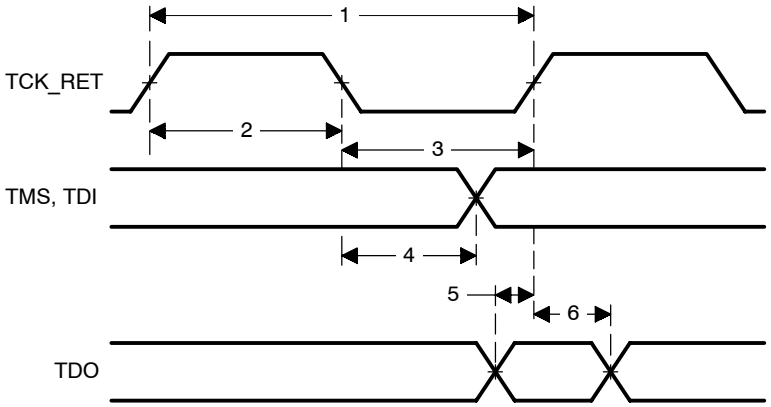
B

B.4 エミュレータ ケーブル ポッド信号タイミング

図B−3は、エミュレータ ケーブル ポッド信号タイミングを示しています。表B−2は、図に示されるタイミング パラメータを定義しています。これらのタイミング パラメータは、エミュレータおよびケーブル ポッドの標準データ シートに指定された値から計算されたものであり、リファレンスでしかありません。テキサス インスツルメンツはこれらのタイミングについてはテストしておらず、又、保証しているものではありません。

エミュレータ ポットは、TCK_RETを内部同期のためのクロック ソースとして使用します。TCKは、オプションのターゲット システム テスト クロック ソースとして供給されます。

図B−3. エミュレータ ケーブル ポッドのタイミング



表B−2. エミュレータ ケーブル ポッドのタイミング パラメータ

番号	パラメータ	説明	最大	最小	単位
1	$t_{c(TCK)}$	TCK_RETの周期	35	200	ns
2	$t_{w(TCKH)}$	TCK_RETのハイ パルス幅	15		ns
3	$t_{w(TCKL)}$	TCK_RETのロー パルス幅	15		ns
4	$t_{d(TMS)}$	TCK_RETがローから、TMS又はTDIが有効になるまでの遅延時間	6	20	ns
5	$t_{su(TDO)}$	TCK_RETがハイになるまでの、TDIセットアップ時間	3		ns
6	$t_{h(TDO)}$	TCK_RETハイからの、TDOホールド時間	12		ns

B.5 エミュレータ タイミング計算

例B−1および例B−2は、各自のシステムのエミュレーション タイミングの計算に役立ちます。実際のターゲット タイミング パラメータについては、エミュレートするデバイスのデータ シートを参照してください。

ここでの例は、次の項目を仮定しています。

$t_{su(TTMS)}$	TCKハイまでのターゲットTMSまたはTDIセットアップ時間	10 ns
$t_{d(TTDO)}$	TCKローからのターゲットTDO遅延時間	15 ns
$t_{d(bufmax)}$	ターゲット バッファ最大遅延時間	10 ns
$t_{d(bufmin)}$	ターゲット バッファ最小遅延時間	1 ns
$t_{bufskew}$	同一パッケージ内の2つのデバイス間のターゲット バッファ スキュー時間 $[t_{d(bufmax)} - t_{d(bufmin)}] \times 0.15$	1.35 ns
$t_{TCKfactor}$	40/60%デューティー サイクル クロックを想定したデューティー サイクル	0.4 (40%)

さらに、ここでの例ではB−6ページ、表B−2からの次の値を使用します。

$t_{d(TMSmax)}$	TCK_RETローからのエミュレータTMSまたはTDI最大遅延時間	20 ns
$t_{su(TDOmin)}$	エミュレータTCK_RETハイまでのTDO最小セットアップ時間	3 ns

エミュレーション設計で考慮されるべき重要なタイミング パスが2種類あります。

- ❑ $t_{pd(TCK\_RET-TMS/TDI)}$(伝播遅延時間)という、TCK_RETからTMSまたはTDIへのパス。
- ❑ $t_{pd(TCK_RET-TDO)}$という、TCK_RETからTDOへのパス。

例の中では、最大のシステム テスト クロック周波数を決定する為に、最悪のパス遅延を計算しています。

B

例B−1. **バッファなしの単一プロセッサ システムのキー タイミング**

$$t_{pd\,(TCK\_RET\text{-}TMS/TDI)} = \frac{\left[t_{d\,(TMSmax)} + t_{su\,(TTMS)}\right]}{t_{TCKfactor}}$$

$$= \frac{(20\ ns + 10\ ns)}{0.4}$$

$$= 75\ ns,\ or\ 13.3\ MHz$$

$$t_{pd\,(TCK\_RET–TDO)} = \frac{\left[t_{d\,(TTDO)} + t_{su\,(TDOmin)}\right]}{t_{TCKfactor}}$$

$$= \frac{(15\ ns + 3\ ns)}{0.4}$$

$$= 45\ ns,\ or\ 22.2\ MHz$$

このケースでは、TCK_RET**から**TMS/TDI**へのパス完了のための時間の方が長いため、(このパスが)制限因子になります。**

例B−2. **バッファ入力および出力をともなうマルチプロセッサ システムのキー タイミング**

$$t_{pd\,(TCK\_RET\text{-}TMS/TDI)} = \frac{\left[t_{d\,(TMSmax)} + t_{su\,(TTMS)} + t_{bufskew}\right]}{t_{TCKfactor}}$$

$$= \frac{(20\ ns + 10\ ns + 1.35\ ns)}{0.4}$$

$$= 78.4\ ns,\ or\ 12.7\ MHz$$

$$t_{pd\,(TCK\_RET–TDO)} = \frac{\left[t_{d\,(TTDO)} + t_{su\,(TDOmin)} + t_{d\,(bufmax)}\right]}{t_{TCKfactor}}$$

$$= \frac{(15\ ns + 3\ ns + 10\ ns)}{0.4}$$

$$= 70\ ns,\ or\ 14.3\ MHz$$

このケースでも、TCK_RET**から**TMS/TDI**へのパス完了のための時間の方が長いため、(このパスが)制限因子になります。**

マルチプロセッサ アプリケーションでは、EMU0**および**EMU1**ラインが、ロジック ロー レベルからロジック ハイ レベルに**10 μs**以下で確実に行けるようにすることが必要で、このパラメータは立ち上がり時間**t_r**と呼ばれます。これは次のように計算されます。**

$$
\begin{aligned}
t_r &= 5(R_{pullup} \times N_{devices} \times C_{load\_per\_device}) \\
&= 5(4.7\ k\Omega \times 16 \times 15\ pF) \\
&= 5(4.7 \times 10^3\ \Omega \times 16 \times 15 = no^{-12}\ F) \\
&= 5(1128 \times 10^{-9}) \\
&= 5.64\ \mu s
\end{aligned}
$$

B

B.6 エミュレータとターゲット システムの接続

エミュレータとJTAGターゲット システムとの間に高品質の信号を供給することは、非常に重要です。エミュレータとターゲットシステムの正しい動作を保証するためには、適切な信号バッファとテスト クロックの供給、及び、適切なマルチプロセッサ相互接続が必要です。

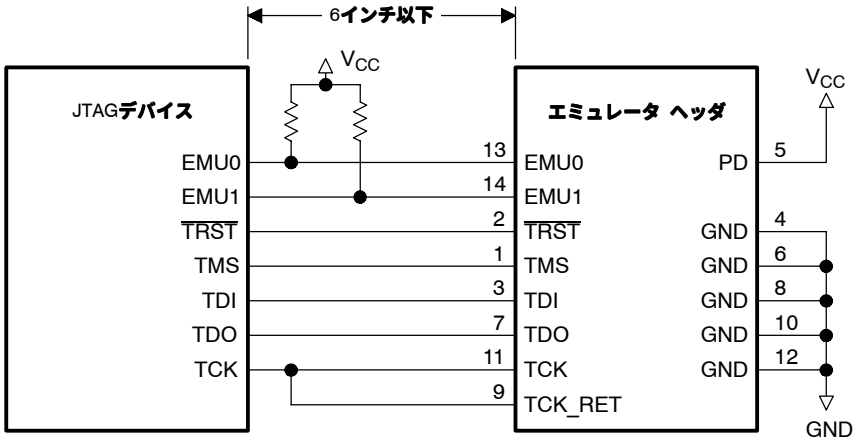
JTAGターゲット デバイスのEMU0およびEMU1ピンに適用する信号は、入力または出力どちらも可能です。これら2つのピンには、入力又は出力のどちらの信号も適用出来ます。一般に、マルチプロセッサ システムでは、これら２つのピンは両方とも入力としても出力としても使用可能で、グローバル システムでの実行、停止を制御します。EMU0およびEMU1信号は、XDS510エミュレータ ヘッダへは入力としてのみ適用されます。

B.6.1 信号のバッファリング

エミュレータ ヘッダとJTAGターゲット デバイスの距離が6インチより長い場合は、エミュレータ信号はをバッファする必要があります。この距離が6インチ未満のときは、バッファは必要ありません。図B−4は、シンプルなバッファなしの例を示しています。

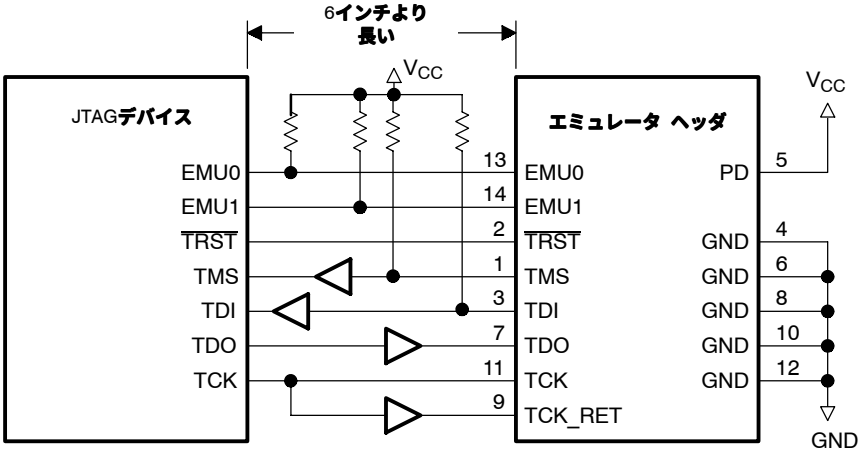
ヘッダとJTAGターゲット デバイスの距離は、6インチを超えてはなりません。EMU0およびEMU1信号の立ち上がり時間を10μs以内にするためには、V_{CC}に接続された、プルアップ抵抗が必要です。また、その抵抗値は、通常4.7kΩが適当です。

図B−4. 信号バッファリングなしのエミュレータ接続



図B−5は、バッファ送信が必要な場合の接続を表しています。エミュレーション ヘッダとプロセッサの間の距離は、6インチより長くなっています。エミュレータ信号のTMS、TDI、TDO、TCK_RETは、同じデバイス パッケージによってバッファされます。

図B−5. 信号バッファリングをともなうエミュレータ接続



EMU0およびEMU1信号の立ち上がり時間を10 μs以内にするためには、V_{CC}に接続された、プルアップ抵抗が必要です。また、その抵抗値は、通常4.7kΩが適当です。

TMSおよびTDIのための入力バッファには、V_{CC}に接続されるプルアップ抵抗を入れて、エミュレータが接続されないときこれらの信号を既知の値にホールドする必要があり、その抵抗値は4.7kΩ以上をお勧めします。

高品質な信号を供給するために(特にプロセッサのTCKおよびエミュレータのTCK_RET信号)、プリント基板の引き回しには特に注意が必要です。また終端抵抗を使ってライン インピーダンスを合わせるのが必要な場合もあります。エミュレータ ポッドには、オプションの内蔵並列ターミネータがTCK_RETおよびTDOにあります。TMSおよびTDIは固定の直列終端があります。

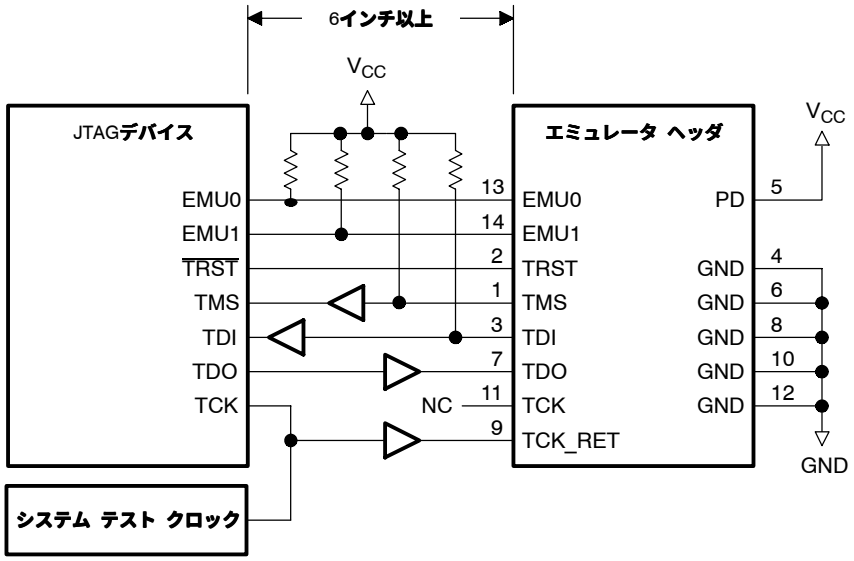
$\overline{\text{TRST}}$は非同期信号なので、必要に応じてバッファリングすることにより、ターゲット デバイスに十分な電流を確保する必要があります。

B

B.6.2 ターゲット システムのクロックの使用法

図B−6は、ターゲット システムで生成したシステム テスト クロックを利用するアプリケーションを示しています。このアプリケーションでは、エミュレータのTCK信号は未接続のままになります。

図B−6. ターゲット システムによるテスト クロック生成



注： TMSおよびTDIラインがバッファされる場合、エミュレータ ケーブルが接続されていなければプルアップ抵抗を使用してバッファ入力を既知のレベルにホールドする必要があります。

ターゲット システムでテスト クロックを発生する場合、2つの利点があります。

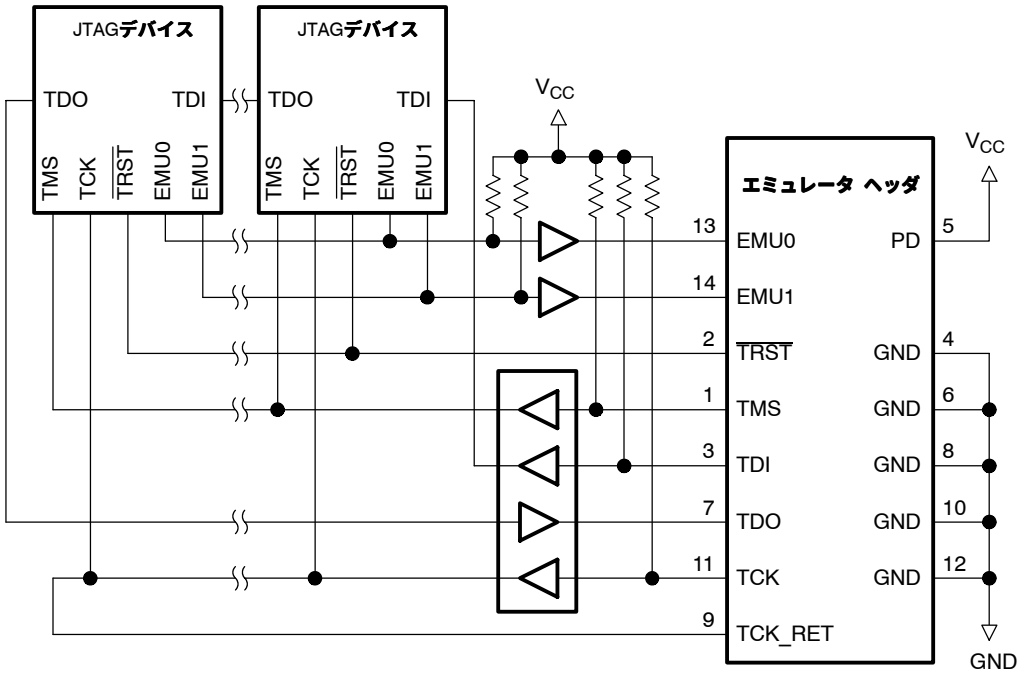
- エミュレータは、10.368MHzテスト クロックのみを供給します。ターゲット システムに各自のテスト クロックを生成させれば、周波数を設定して各自のシステムの条件に合わせることができます。
- 場合によっては、エミュレータが接続されていないときテスト クロックが必要な他のデバイスが、各自のシステムにあることも考えられます。システム テスト クロックは、この目的も果たします。

B.6.3 マルチプロセッサの構成

図B−7は、代表的なディジ チェイン接続のマルチプロセッサ構成を示しており、このような構成はIEEE−1149.1仕様の必要最小条件に満たします。エミュレーション信号をバッファすることにより、プロセッサをエミュレータからアイソレートし、ターゲット システムをドライブするのに十分な信号を供給します。このインタフェースの長所のひとつは、テスト クロックを低速化してタイミングの問題を解消できる点です。マルチプロセッサをサポートするには、次のガイドラインにしたがってください。

- プロセッサのTMS、TDI、TDO、TCK信号は、必ず同じ物理デバイス パッケージによりバッファすること、タイミング スキューの制御が向上します。
- TMS、TDI、TCKのための入力バッファには、V_{CC}に接続されるプルアップ抵抗を入れて、エミュレータが接続されていないときこれらの信号を既知の値にホールドする必要があり、その抵抗値は4.7kΩ以上を推奨します。
- EMU0およびEMU1のバッファはオプションですが、アイソレーションのために、使用すべきです。これらはそれほど重要な信号ではないので、TMS、TDI、TDO、TCKと同じ物理パッケージによってバッファする必要はありません。

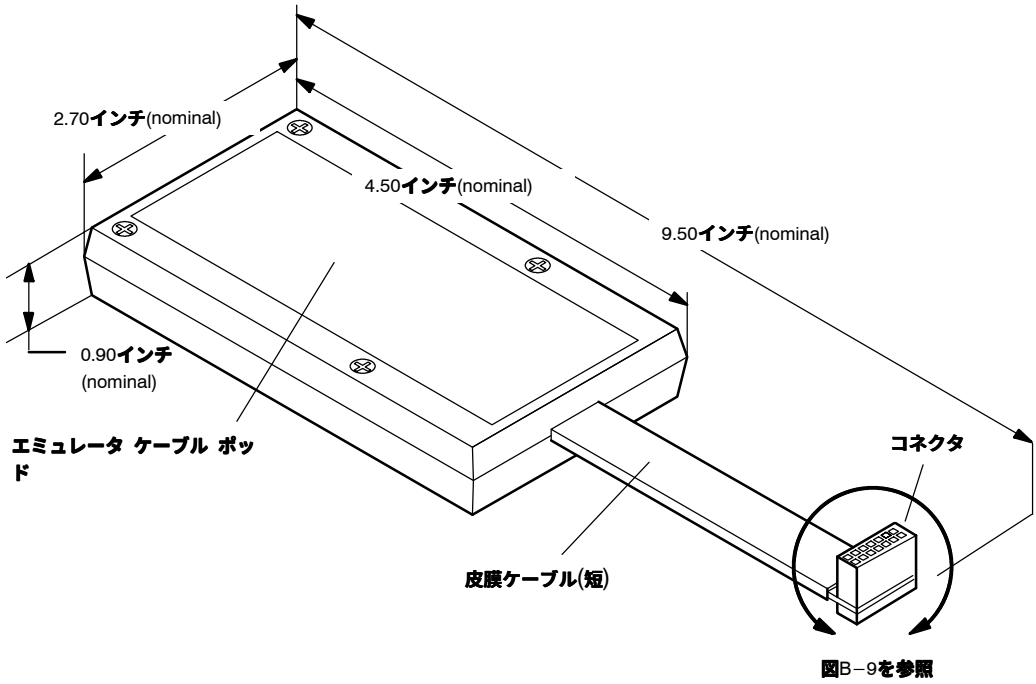
図B−7. マルチプロセッサ接続



B.7 14ピン エミュレータ コネクタの寸法

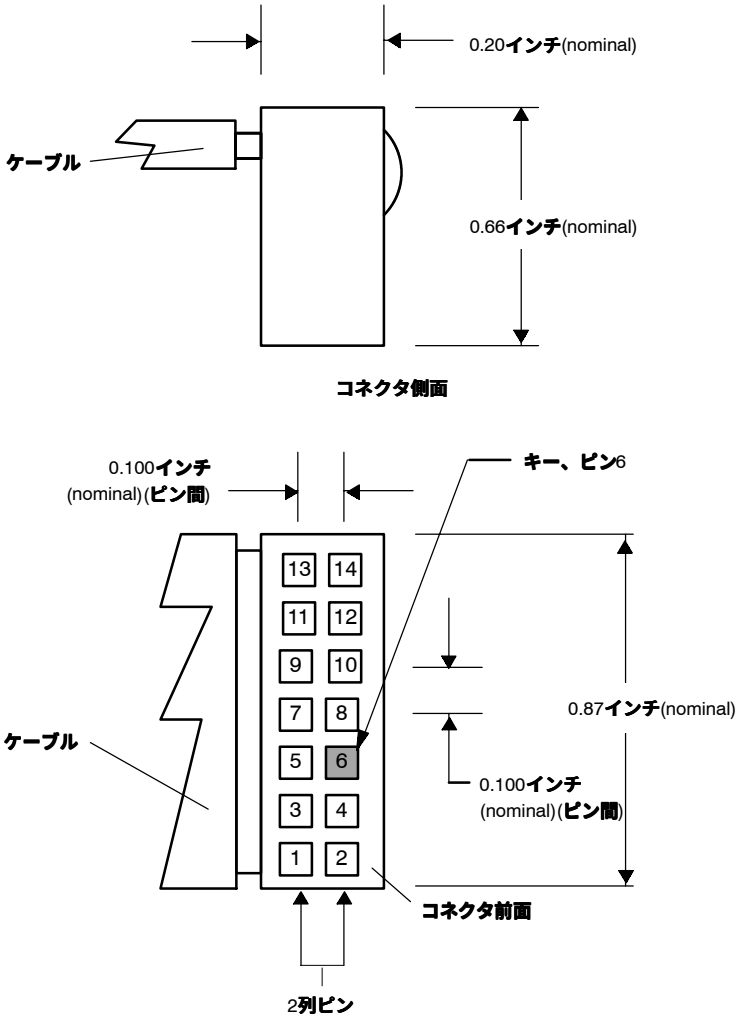
JTAGエミュレータ ターゲット ケーブルは、エミュレータと接続する約90cmの皮膜ケーブルと、アクティブ ケーブル ポット、さらにターゲット システムと接続する短い皮膜ケーブルから構成されます。ケーブル全体の長さは、約115cmです。図B−8および図B−9(B−15ページ)は、ターゲット ケーブル ポッドおよびショート ケーブルの寸法を示しています。ケーブル ポッド ボックスは絶縁プラスチック製で、4本の金属ネジで止めてあります。

図B−8. ポッド/コネクタ寸法



注： 指定がない限りすべての寸法はインチ単位でnominalの寸法です。コネクタ上のピン間はXおよびY平面の双方で0.100インチです。

図B−9. 14**ピン コネクタ寸法**



B

B.8 エミュレーション設計について

このセクションでは、スキャン パス リンカ(SPL)の使用およびアプリケーションについて説明します。SPLは、4つすべての二次のJTAGスキャン パスを同時にメインのスキャン パスに追加できます。また、エミュレーション ピンの使用およびマルチプロセッサの構成についても説明します。

B.8.1 スキャン パス リンカの使用

TIのスキャン パス リンカACT8997を使うことにより、JTAGエミュレーション スキャン パスをより小さな論理接続された4から16個のデバイスのグループに分割できます。Advanced Logic and Bus Interface Logic Data Bookに解説されるように、SPLはJTAGエミュレーション スキャンと互換性があります。SPLは、4つの二次のスキャン パスのあらゆる組み合わせをメインのスキャン パスに加えることができます。

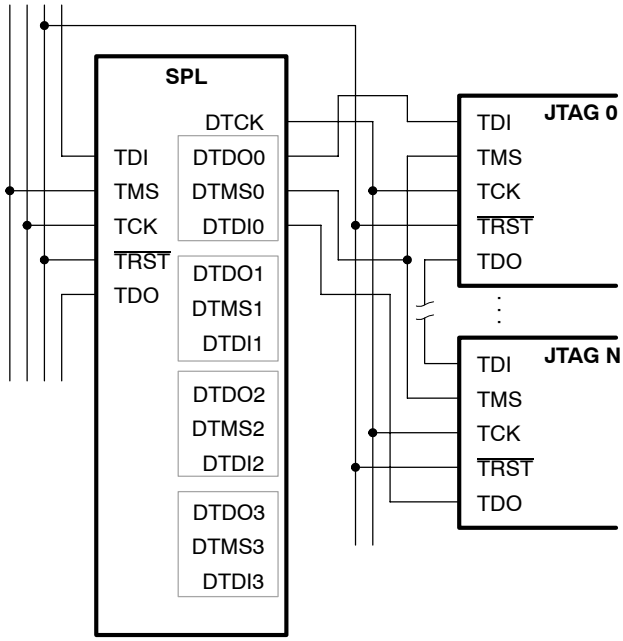
複数の二次のJTAGスキャン パスのシステムには、シングルのスキャン パスに比べて優れたフォルト トレランスおよびアイソレーションがあります。SPLはすべての二次のスキャン パスをメインのスキャン パスに同時に加えられるので、選択されたプロセッサのグループ単位での起動や停止など、グローバルなエミュレーション動作をサポートします。

TIのエミュレータは、SPLのネストをサポートしません(例えば、別のSPLの二次のスキャン パスに接続するSPLなど)。しかし、メインのスキャン パス上に複数のSPLを置くこと可能です。

スキャン パス セレクタは、このエミュレーション システムではサポートされません。TIのACT8999スキャン パス セレクタはSPLに似ていますが、一度にひとつの二次のスキャン パスのみをメインのJTAGスキャン パスに加えることができます。したがって、グローバルなエミュレーション動作はスキャン パス セレクタでは確保されません。

SPLはバックプレーンに挿入できるので、ノンバックプレーン デバイスに必要なジャンパ配線なしに最大4つのデバイス ボードを加えることができます。他のデバイスを接続するのと同じ方法で、SPLをメインのJTAGスキャン パスに加えられます。図B−10は、二次のスキャン パスをSPLに接続する方法を示しています。

図B−10. 二次のJTAGスキャン パスとスキャン パス リンカ(SPL)の接続



メイン スキャン パスからの$\overline{\text{TRST}}$信号は、SPLの二次のスキャン パス上のデバイスなど、すべてのデバイスをドライブします。SPLの二次のスキャン パス上の各ターゲット デバイスのTCK信号は、SPLのDTCK信号によってドライブします。二次のスキャン パス上の各デバイスのTMS信号は、SPLのそれぞれに対応するDTMS信号によってドライブされます。

SPLのDTDO0は、二次のスキャン パス上の最初のデバイスのTDI信号として接続されます。SPLのDTDI0は、二次のスキャン パスの最後のデバイスのTDOに接続されます。各々の二次のスキャン パスの中では、各デバイスのTDIがそのデバイスの前のデバイスのTDOに接続されます。SPLがバックプレーンにある場合は、その二次のJTAGスキャン パスはアドオン ボード上にあり、信号の劣化が問題になる場合は、$\overline{\text{TRST}}$およびDTCK信号の両方をバッファする必要があります。DTMS信号については劣化しにくいですが、同じ理由でその信号をバッファする必要がある場合もあります。

B.8.2 スキャン パス リンカ(SPL)のエミュレーション タイミングの計算

例B−3および例B−4は、ユーザのシステムのSPLの二次のスキャン パスにおけるクリティカルなエミュレーション タイミングを計算するのに役立ちます。実際のターゲットのタイミング パラメータについては、各ターゲット デバイスのデータ シートを参照してください。

ここでの例では、次の項目を仮定しています。

$t_{su(TTMS)}$	TCKハイまでのターゲットTMSまたはTDIセットアップ時間	10 ns
$t_{d(TTDO)}$	TCKローからのターゲットTDO遅延時間	15 ns
$t_{d(bufmax)}$	ターゲット バッファ最大遅延時間	10 ns
$t_{d(bufmin)}$	ターゲット バッファ最小遅延時間	1 ns
$t_{(bufskew)}$	同一パッケージ内の2つのデバイス間のターゲット バッファ スキュー時間 $[t_{d(bufmax)} - t_{d(bufmin)}] \times 0.15$	1.35 ns
$t_{(TCKfactor)}$	40/60%デュティー サイクル クロックを想定したデューティー サイクル	0.4 (40%)

さらに、ここでの例ではSPLデータシートからの次の値を使用します。

$t_{d(DTMSmax)}$	TCKローからのSPL DTMSまたはDTDO最大遅延時間	31 ns
$t_{su(DTDLmin)}$	SPL TCKハイへのDTDI最小セットアップ時間	7 ns
$t_{d(DTCKHmin)}$	TCKハイからのエミュレータTMSまたはTDI最大遅延時間	2 ns
$t_{d(DTCKLmax)}$	TCKローからのエミュレータTMSまたはTDI最小遅延時間	16 ns

エミュレーション設計で考慮されるべき重要なタイミング パスが2種類あります。

- ❑ $t_{pd(TCK-TDMS)}$という、TCKからTDMS/DTDOへのパス。
- ❑ $t_{pd(TCK-DTDI)}$という、TCKからDTDIへのパス。

下記の2つのケースでは、最悪時のパスの遅延を計算して、最大のシステム テスト クロック周波数を決定しています。

例B−3. **バッファなしのシングル プロセッサ システムのキー タイミング**(SPL)

$$t_{pd\,(TCK\text{-}DTMS)} = \frac{\left[t_{d\,(DTMSmax)} + t_{d\,(DTCKHmin)} + t_{su\,(TTMS)}\right]}{t_{TCKfactor}}$$

$$= \frac{(31\text{ ns} + 2\text{ ns} + 10\text{ ns})}{0.4}$$

$$= 107.5\text{ ns, or } 9.3\text{ MHz}$$

$$t_{pd\,(TCK\text{-}DTDI)} = \frac{\left[t_{d\,(TTDO)} + t_{d\,(DTCKLmax)} + t_{su\,(DTDLmin)}\right]}{t_{TCKfactor}}$$

$$= \frac{(15\text{ ns} + 16\text{ ns} + 7\text{ ns})}{0.4}$$

$$= 9.5\text{ ns, or } 10.5\text{ MHz}$$

このケースでは、TCK**から**DTMS/DTDL**へのパスが制限因子になります。**

例B−4. **バッファ入力および出力をともなうシングルまたはマルチプロセッサ システムのキー タイミング**

$$t_{pd\,(TCK\text{-}TDMS)} = \frac{\left[t_{d\,(DTMSmax)} + t_{(DTCKHmin)} + t_{su\,(TTMS)} + t_{(bufskew)}\right]}{t_{TCKfactor}}$$

$$= \frac{(31\text{ ns} + 2\text{ ns} + 10\text{ ns} + 1.35\text{ ns})}{0.4}$$

$$= 110.9\text{ ns, or } 9.0\text{ MHz}$$

$$t_{pd\,(TCK–DTDI)} = \frac{\left[t_{d\,(TTDO)} + t_{d\,(DTCKLmax)} + t_{su\,(DTDLmin)} + t_{d\,(bufskew)}\right]}{t_{TCKfactor}}$$

$$= \frac{(15\text{ ns} + 15\text{ ns} + 7\text{ ns} + 10\text{ ns})}{0.4}$$

$$= 120\text{ ns, or } 8.3\text{ MHz}$$

このケースでは、TCK**から**DTDI**パスが制限因子になります。**

B.8.3 エミュレーション パスの使用法

TIデバイスのEMU0/1ピンは、双方向、3ステート出力ピンです。これらのピンがアクティブではない時，その出力はハイ インピーダンスになります。ピンがアクティブのとき、次の2タイプの出力のうちどちらか一方を出力します。

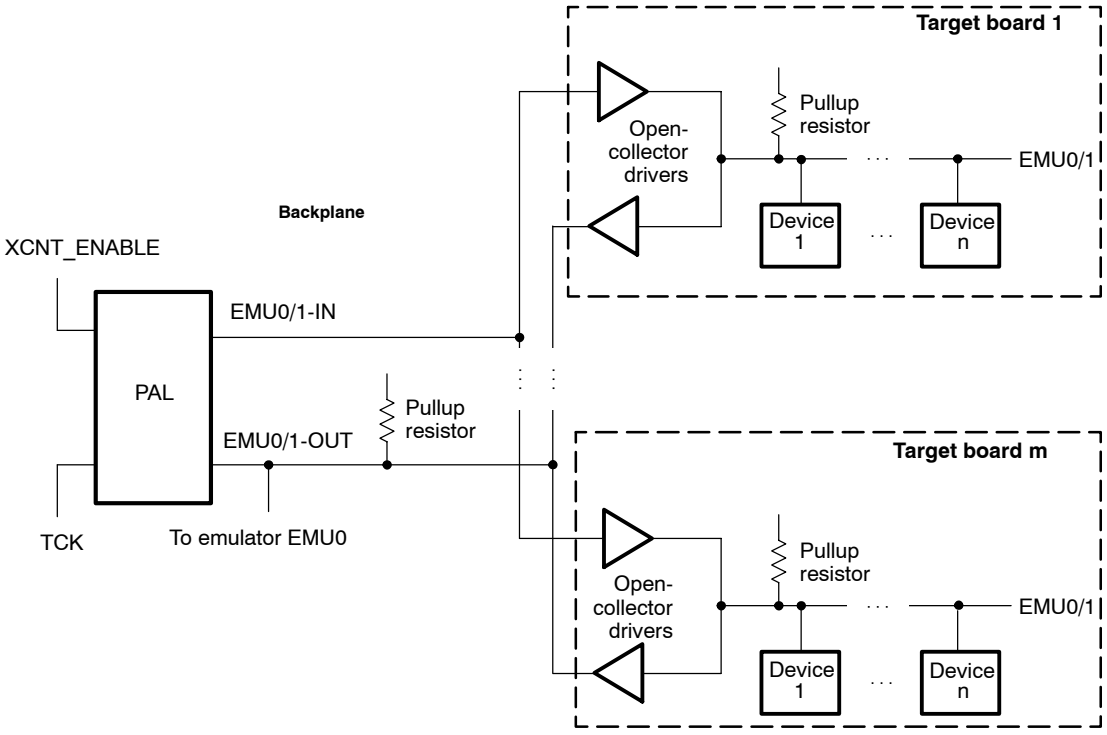
- ❑ **イベント信号。**EMU0/1ピンを、内部のイベントが発生した事を知らせるようにソフトウェアによって設定できます。このモードでは、これらのピンのひとつをローにドライブすることで、デバイスがこのようにイベントを知らせるようになります。この動作を可能にするために、EMU0/1ピンはオープン コレクタ ソースになっています。このIC出力型に従うように、ロジック アナライザなどの外部デバイスをEMU0/1信号に接続できます。このような外部ソースを使用する場合は、オープン コレクタ ソースとして接続することが必要です。

- ❑ **外部カウンタ。**EMU0/1ピンを、外部カウンタをドライブするためのトーテムポール出力として、ソフトウェアによって設定できます。複数のデバイスの出力をトーテムポール動作として構成する場合は、これらのデバイスにダメージを与える場合があります。エミュレーション ソフトウェアは、このような状態を検出して回避しますが、EMU0/1信号で外部ソースを制御する機能はありません。したがってすべての外部ソースは、いずれかのデバイスが外部カウンタを使用する場合は必ずノン アクティブにされなければなりません。

TIデバイスは、それらのEMU0/1ピンがローにドライブされたとき、処理を停止するようにソフトウェアによって設定できます。この機能とイベント信号出力との組み合わせにより、システムレベルでのデバッグにおいて、一つのＴＩデバイスの特定のイベントが発生した時に、他のすべてのＴＩデバイスの動作を停止させることができます。

EMU0/1信号を複数のボードにまたがって配線する場合、これらの信号は通常のエミュレーション信号より複雑なことから、特別の処理が必要です。図B−11はひとつの構成例を示しており、これは、システム内のどのプロセッサも同じシステム内の他の任意のプロセッサを停止できるものです。16個を超えるプロセッサのEMU0/1ピンを、バッファなしに1グループに結合してはなりません。バッファにより、デバッガのRUNB(ベンチマーク実行)コマンドやアナリシス モードの外部カウンタ機能の使用時に必要な信号を供給できます。

図B−11. 25ns未満のタイミング条件に合うEMU0/1構成



注： 1) EMU0/1−INがローである時間は、最短で1TCKサイクル、最長で10 µs未満になります。ソフトウェアは、EMU0/1−OUTピンをハイに設定します。

2) EMU1上のオープン コレクタ ドライバおよびプルアップ抵抗が、立ち上がり/立ち下がり時間を25ns未満にするためには、この図に示されるような修正が考えられます。立ち上がり時間が25nsを超えると、RUNBコマンドを実行した時、またはデバッガのアナリシス メニューから外部カウンタを選択したとき、エミュレータがエッジを誤って検出する場合があります。

図B−11に示される回路および図B−12に示されるタイミングには、下記の7つの要点があります。

- ❑ オープン コレクタ ドライバは、各ボードをアイソレートします。EMU0/1ピンは各ボード上だけで接続されます。
- ❑ ボード エッジでは、EMU0/1信号は分割され入出力の双方を可能にします。これは、一度だけ設定できるラッチとしてオープン コレクタ ドライバが作動するのを防ぐのに必要です。
- ❑ EMU0/1信号は、バックプレーンにバス転送されます。場合によって、プルアップ抵抗が必要です。

B

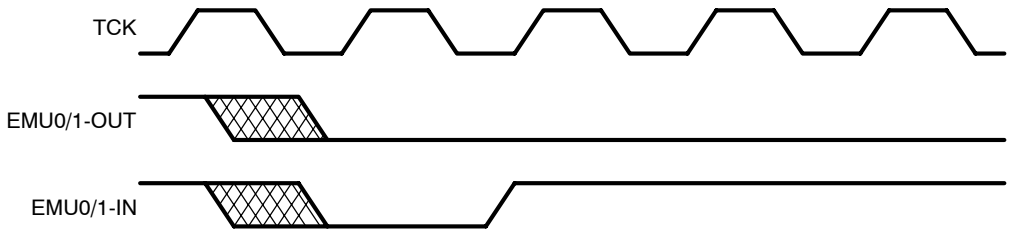
- バス転送されるEMU0/1信号は、プログラマブル ロジック アレイ デバイスPAL®に送り込まれますが、このPALの機能は、EMU0/1−OUT信号がロー レベルとして検出されたとき、EMU0/1−IN信号をロー パルスとして生成することです。このパルスは、デバイスに有効であるように1TCKサイクルより必ず長くなりますが、エミュレーション ソフトウェアがデバイス ピンをクリアした場合のコンフリクトや再トリガを回避するために10μs未満になります。

- デバッガのRUNBコマンドまたはアナリシス モードの外部カウンタを使用する際、ターゲット デバイス上のEMU0/1ピンはトーテムポール出力になります。EMU1ピンは内蔵カウンタのリップル キャリアウトです。EMU0は、プロセッサ停止(processor−halted)信号になります。RUNBまたは外部カウンタを使用する際、すべてのボードへのEMU0/1−IN信号は必ずハイ(ディセーブル)のままにする必要があります。何らかのタイプの外部入力(XCNT_ENABLE)をPAL®に供給して、PAL®がEMU0/1−INをローにドライブできないようにすることが必要です。

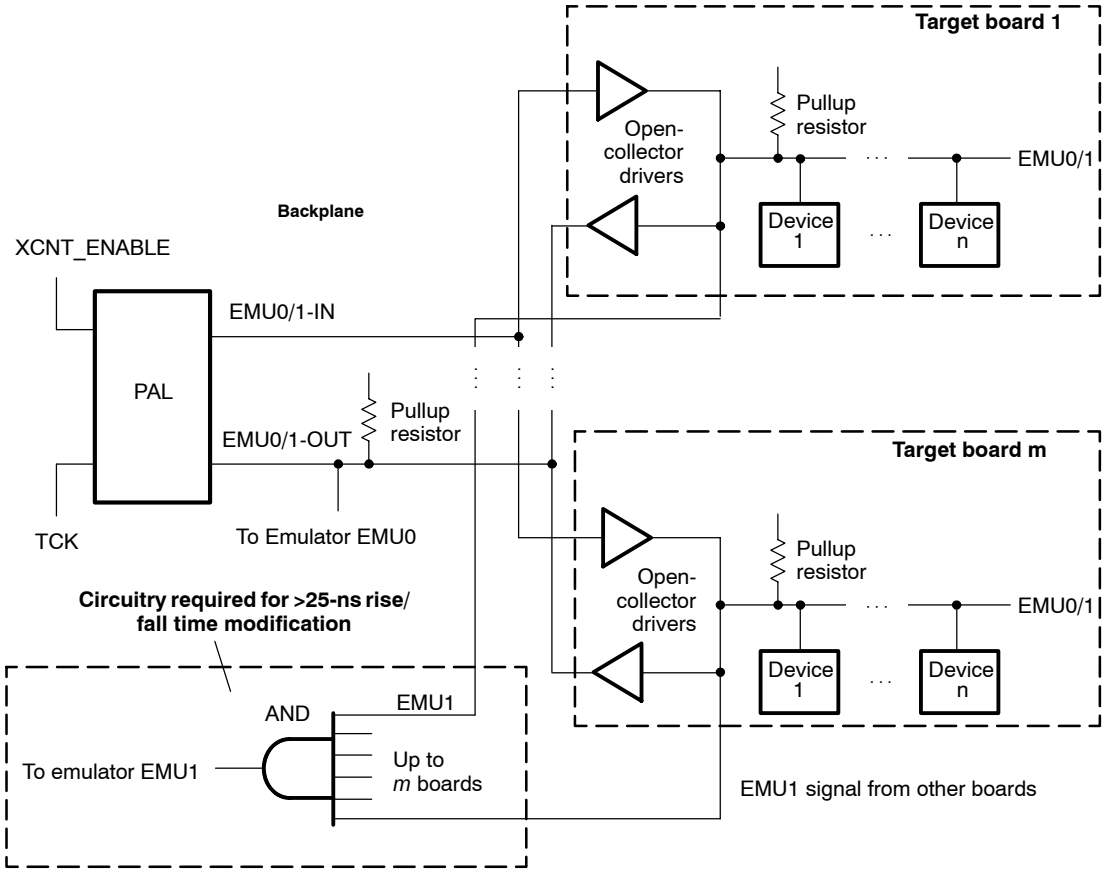
- TIプロセッサ以外のソース(ロジック アナライザなど)を使ってEMU0/1をドライブする場合、そのソースの信号はオープン コレクタ ドライバによって必ずアイソレートして、RUNBおよび外部カウンタを使用する際にノン アクティブにするようにします。

- EMU0/1−OUT信号をエミュレーション ヘッダ、または直接テスト バス コントローラに接続することが必要です。

図B−12. EMU0およびEMU1シグナルの推奨タイミング



図B−13. 25ns**以上のタイミング条件を満たす追加**AND**ゲートを使った**EMU0/1**の構成例**



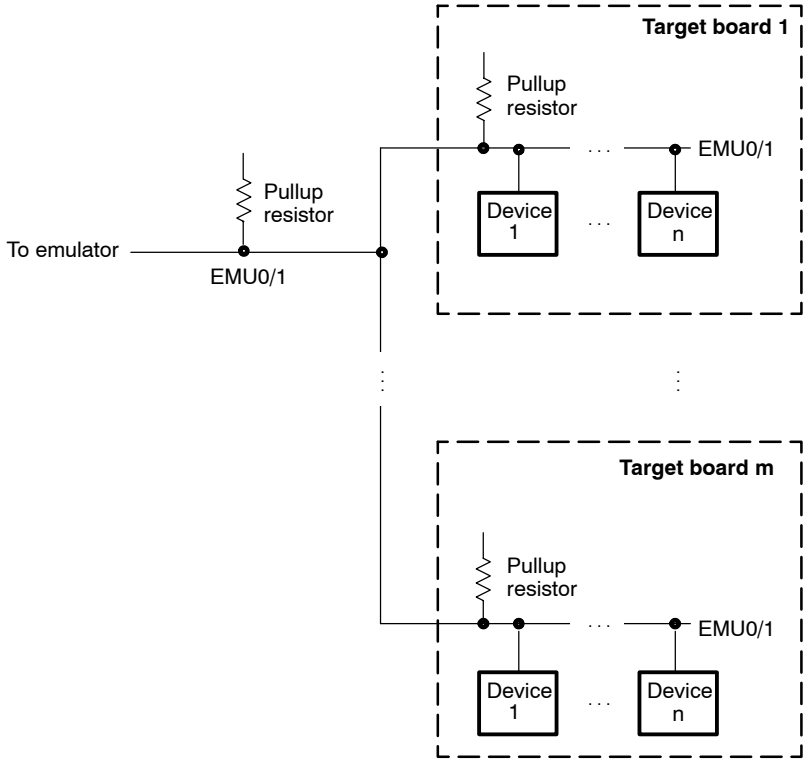
注： 1) EMU0/1−IN**がローである時間は最短で**1TCK**サイクル、最長で**10 µs**未満になります。ソフトウェアは、**EMU0/1−OUT**ピンをハイに設定します。**

2) EMU1**上のオープン コレクタ ドライバおよびプルアップ抵抗が、立ち上がり/立ち下がり時間を**25ns**未満にするためには、この図に示されるような修正が考えられます。立ち上がり時間が**25ns**を超えると、**RUNB**コマンドを実行した際、またはデバッガのアナリシス メニューから外部カウンタを選択したとき、エミュレータがエッジを誤って検出する場合があります。**

B

あるターゲット ボード上のデバイスが他のターゲット ボード上のデバイスをEMU0/1信号を使って停止する必要は限ずしもありません(図B−14参照)。この構成では、グローバルな停止機能が失われます。EMU0/1に16個以上のデバイスの負荷をかけないことが重要です。

図B−14. 広域停止をともなわないEMU0/1構成



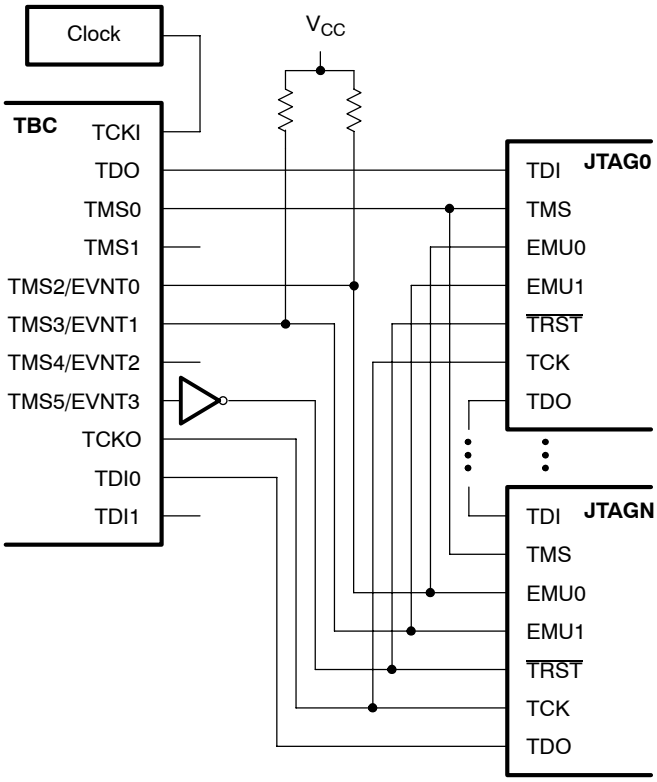
注： EMU1上のオープン コレクタ ドライバおよびプルアップ抵抗は、かならず立ち上がり/下がり時間を25ns未満にする必要があります。立ち上がり時間が25nsを超えると、RUNBコマンドの際、またはデバッガのアナリシス メニューから外部カウンタを選択したとき、エミュレータがエッジを誤って検出する場合があります。この条件を満たせない場合は、個々のボードからのEMU0/1信号はともに論理積演算して(図B−14参照)、エミュレータのためのEMU0/1信号を生成します。

B

B.8.4 診断アプリケーションの実行

内蔵診断機能が必要なシステムについては、エミュレーション スキャン パスをエミュレーション ヘッドの代わりに直接TIのACT8990テスト バス コントローラ(TBC)に接続できます。TBCについては、テキサス インスツルメンツAdvanced Logic and Bus Interface Logic Data Bookを参照してください。図B−15は、n個のデバイスをTBCへ接続するスキャン パス 接続を示しています。

図B−15. n**本の** JTAG**スキャン パスのための**TBC**エミュレーション接続**



図B−15**に示されるシステム設計では、**TBC**のエミュレーション信号は、**TCKI、TDO、TMS0、TMS2/EVNT0、TMS3/EVNT1、TMS5/EVNT3、TCKO、TDI0**が使用され、**TMS1、TMS4/EVNT2、TDI1**は接続されません。ターゲット デバイスの**EMU0**および**EMU1**信号は、プルアップ抵抗を通して**VCC**に接続され、さらにそれぞれ**TBC**の**TMS2/EVNT0、TMS3/EVNT1**ピンに接続します。**TBC**の**TCKI**ピンはクロック発生器に接続します。メインの**JTAG**スキャン パスのための**TCK**信号は、**TBC**の**TCKO**ピンによってドライブされます。

TBCでは、**TMS0**ピンがメインの**JTAG**スキャン パス上の各デバイスの**TMS**ピンをドライブします。**TBC**上の**TDO**は、メインの**JTAG**スキャン パス上の最初のデバイスの**TDI**に接続します。**TBC**上の**TDI0**は、メインの**JTAG**スキャン パス上の最後のデバイスの**TDO**信号に接続されます。メインの**JTAG**スキャン パス内では、デバイスの**TDI**信号はその前のデバイスの**TDO**信号に接続されます。各デバイスのための$\overline{\text{TRST}}$は、**TBC**のソフトウェア制御のための**TMS5/EVNT3**信号を反転するか、またはボード上のロジックによって生成できます。

付録C

開発サポートおよび部品注文情報

この付録Cには、'54xの開発サポート情報、部品型番、サポート ツール注文についての情報を記します。

'54xの各サポート製品については、TMS320 DSP 製品概要を参照してください。100以上のサードパーティが、TI TMS320ファミリーをサポートする製品を提供しています。詳しくは、TMS320 Third−Party Support Reference Guideを参照してください。

価格および製品出荷などの情報については、お近くのTI営業所または正規特約店までお問い合わせください。お問い合わせ先は本書の奥付を参照してください。

Topic Page

C

C.1 開発サポート

このセクションは、テキサス インスツルメンツが提供する開発サポートについて説明します。

C.1.1 開発ツール

TIは、'54xファミリDSPのための一連の開発ツールを提供しており、プロセッサ パフォーマンス評価用ツール、コード生成ツール、アルゴリズム移植開発ツール、ソフトウェアとハードウェア モジュールを統合およびデバッグするツールなどが含まれます。

コード生成ツール

- 最適化ANSI Cコンパイラは、ANSI C言語を高度に最適化されたアセンブリ コードに直接変換します。その後に、コンパイラに付属しているTIアセンブラ/リンカによってコードをアセンブルおよびリンクできます。この製品は、現在PC(DOS、DOS拡張メモリ)、SUN SPARCワークステーションに対応しています。ツールの詳細については、TMS320C54x Optimizing C Compiler User's Guideを参照してください。
- アセンブラ/リンカは、アセンブラ ソースコードを実行可能なオブジェクト コードに変換します。この製品は現在、PC(DOS、DOS拡張メモリ)に対応しています。SPARCワークステーション向け'54xアセンブラは、オプティマイジング'54xコンパイラの一部としてのみ出荷されています。アセンブリ言語ツールについては、TMS320C54x Assembly Language Tools User's Guideを参照してください。

システム統合およびデバッグ ツール

- シミュレータは、(ソフトウェアによって)'54xの動作をシミュレートして、Cおよびアセンブリ ソフトウェア開発において使用します。この製品は現在、PC(DOS、Windows)、SPARCワークステーションに対応しています。デバッガの詳細については、TMS320C54x C Source Debugger User's Guideを参照してください。
- XDS510エミュレータは、'54x上でフルスピードのインサーキット エミュレーションを実行し、すべてのレジスタおよびデバイスの外部、内部メモリにアクセスします。Cおよびアセンブリ ソフトウェア開発に使用でき、マルチプロセッサをデバッグする能力を備えています。この製品は現在、PC(DOS、Windows)、SPARCワークステーションに対応しています。

’C2xx、’C3x、’C4x、’C5x XDS510エミュレータ ボード(またはボックス)は’54xと同じなので、’54x C Source Debugger Softwareを別に購入して’54xのデバッグができます。このソフトウェア製品により、’54xアプリケーションを以前に購入したエミュレータ ボードによってデバッグできます。’C3x XDS510エミュレータに付属のエミュレータ ケーブルは、’54xでは使用できません。代わりに、JTAGエミュレーション用ケーブル(セクションC.2参照)が必要です。’C5x XDS510エミュレータに付属しているエミュレータ ケーブルは’54xにそのまま使用できます。’54xエミュレータの詳細については、TMS320C54x C Source Debugger User’s Guideを参照してください。

- ❑ TMS320C54x評価用ボード(EVT)はDC電源を外部から供給し、DSP自信に内蔵されているブート ローダにより、スタンド アロンでユーザ プログラムを実行させることができます。また、デバッグにはXDS510XL/WSを使用してください。 XDS510XL/WS用ターゲット コネクタが用意されています。このボードは、5ボルト及び3.3ボルト動作のDSPに対応できます。DSPの各パッケージ毎に対応ボードが違います。’54x EVTの詳細については、TMS320ファミリー製品概要を参照してください。

C.1.2 サードパーティ サポート

TMS320ファミリーは、100以上の独立したサードパーティおよびコンサルタント企業による製品、サービスによってサポートされています。サポート製品には様々な種類があり(ソフトウェアおよびハードウェア)、アセンブラ、シミュレータ、DSPユーティリティ パッケージからロジック アナライザ、エミュレータまでが揃っています。サポート サービスには、音声エンコーディング、ベクトル量子化や、ソフトウェア/ハードウェア設計、システム解析などがあります。

サードパーティのサービス、製品、アプリケーション、アルゴリズム開発パッケージなどについては、サードパーティに直接お問い合わせください。連絡先については、TMS320 Third Party Support Reference Guideを参照してください。

C.1.3 技術トレーニング組織(TTO)TMS320ワークショップ

'54x DSP設計ワークショップ:このワークショップは、'54xシリーズのDSPデバイスを設計利用しようと考えている方、またはハードウェア/ソフトウェア設計エンジニアが対象です。コースの具体的な演習により、参加者は'54x開発技術を短期で身につけられます。参加者には、マイクロプロセッサ/アセンブリ言語の経験が必要です。デジタル設計技術およびC言語プログラミング経験があれば、なお可です。

'54xワークショップでは、次の課題について説明します。

- ❑ **'54xアーキテクチャ/命令セット**
- ❑ **PCベースでのソフトウェア シミュレータの使用方法**
- ❑ **'54xアセンブラ/リンカの使用方法**
- ❑ **Cプログラミング環境**
- ❑ **システム アーキテクチャについて**
- ❑ **メモリおよびI/Oインタフェース**
- ❑ **開発サポート**

ワークショップへの参加、料金、登録については、03−3769−8988**までご連絡ください。**

C.1.4 問い合わせ

TMS320におけるデバイスの問題、開発ツール、参考文献、ソフトウェア アップグレード、新製品などについての質問は、日本TIプロダクト インフォメーション センター(Tel:0120−81−0026、Fax:0120−81−0036)**までお問い合わせください。**

C

C.2 部品注文情報

このセクションでは、54xデバイス、開発サポート ハードウェア、ソフトウェア ツールの製品番号について説明します。

C.2.1 デバイスおよび開発ツールの製品記号

製品開発サイクルの段階を示すために、弊社はすべてのTMS320デバイスおよび開発ツールの製品番号に記号を付けています。各TMS320デバイスには、TMX、TMP、TMSの3種類の記号のいずれかが付きます。開発ツールのハードウエアには、TMDXまたはTMDSのいずれかの記号が付きます。これらの記号は、エンジニアリング プロトタイプ(TMX/TMDX)の段階から、評価済み製品デバイスおよびツール(TMS/TMDS)の段階までの製品開発の流れを表します。下記に、記号が示す開発の流れを示します。

デバイス開発の推移:

TMX: 試験的なデバイス。最終的なデバイスの電気的な仕様を満たしているとは限りません。

TMP: デバイスの電気的な仕様を満たした最終シリコンチップ製品ですが、品質、信頼性については検証が完了していません。

TMS: 完全に検証済みの量産デバイスです。

サポート ツール開発の推移:

TMDX: 弊社内部の検証テストを完了していない開発途中の開発ツール製品。

TMDS: この開発サポート製品は、完全に検証された開発ツール製品です。

TMXおよびTMPデバイス、TMDX開発サポート ツールは、「開発途上の製品は内部評価のための製品」という条件付きで出荷されます。

TMSデバイスおよびTMDS開発サポート ツールは、仕様に準じた特性を持ち、デバイスの品質、信頼性ともに検証されています。これらの製品には、テキサス インスツルメンツの標準保証が適用されます。

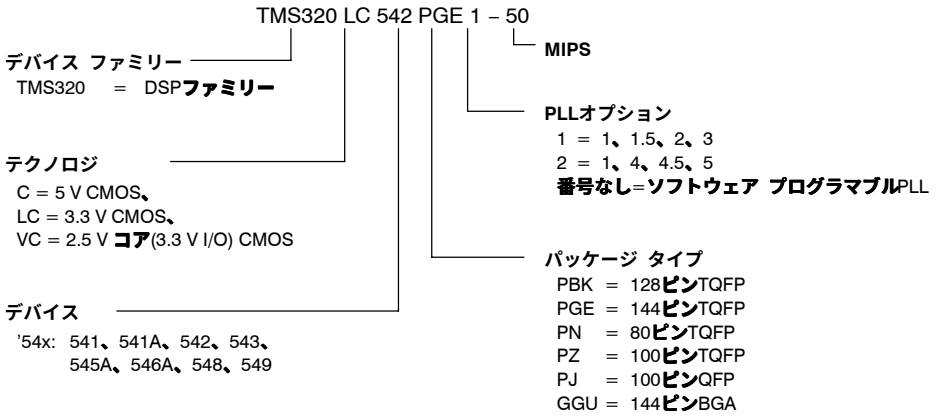
注 :

プロトタイプのデバイス(TMXまたはTMP)は、標準製品デバイスに比べて不良率が高いことが考えられます。テキサス インスツルメンツは、エンド ユースにおける不良率が確定できないので、これらのデバイスをどの量産システムにも使用しないことをお勧めします。検証済みの製品デバイスのみをお使いください。

C.2.2　デバイス名のタイプ型名表記

TIデバイスの型名表記には、デバイス ファミリー名、および記号が表示されています。図C−1は、’54xデバイス ファミリーの製品に共通な型名表記です。

図C−1. TMS320C54xデバイス型名表記



C

C.2.3 開発サポート ツール

表C−1は、’54x用の開発サポート ツール、対応機種、型番を示しています。

表C−1. 開発サポート ツール型番

開発ツール	対応機種	型番
アセンブラ/リンカ	PC (DOS™)	ASM320540-MS
Cコンパイラ/アセンブラ/リンカ	PC (DOS™, Windows™)	CPL320540-MS
Cコンパイラ/アセンブラ/リンカ	SPARC™ (Sun OS™)	CPL320540-SU
XDS−510XL**用Cソース デバッガソフトウェア**	PC (DOS™, Windows™) (XDS510™)	HLL32054X-MS
XDS−510WS**用Cソース デバッガソフトウェア**	SPARC™ (Sun OS™) (XDS510WS™)	HLL32054X-SU
評価用モジュール(EVM)	PC (DOS™, Windows™)	TMDX3260051
シミュレータ(C**言語**)	PC (DOS™, Windows™)	SIM320540MS
シミュレータ(C**言語**)	SPARC™ (Sun OS™)	SIM320540SU
XDS510XL**用エミュレータ ボード**†	PC (DOS™, Windows™)	TMDS00510
XDS510WS**用エミュレータ ボード**‡	SPARC™ (Sun OS™) (SCSI)	TMDS00510WS
3V/5V PC/SPARC JTAG**エミュレーション ケーブル**	XDS510™ / XDS510WS™	TMDS3080002

† XDS510**ボードおよび**JTAG**ケーブルを含みます。** HLL32054X−MS C**ソース デバッガソフトウェアは含まれません。**

‡ XDS510WS**ボックス、**SCSI**ケーブル、電源、**JTAG**ケーブルを含みます。**HLL32054X−SU C**ソース デバッガソフトウェアは含まれません。**

C

各54xデバイスによるサポート機能の差異

この付録は各54xデバイスによってサポートされている機能の差異を表しています。

54xはそれぞれのデバイスによって、次の3つの機能サポートが異なっています。(各機能の詳細に関してはそれぞれの章を参照下さい。)

ソフトウェアPLL (8.4.2**ソフトウェア プログラマブルなPLL、ページ**8−18**を参照**)
飽和モード (4.1.2**プロッセッサ・モード・ステータス・レジスタ**(PMST)、**ページ**4−6**を参照**)
拡張プログラム・アドレス (6**プログラム・メモリ・アドレッシング、ページ**6−1**、および** 3.1.1**拡張 プログラムメモリ、ページ**3−8**参照**)

表D−1は各54xデバイスがその3つの異なる機能の内、どれをサポートしているかを表しています。

表D−1. **各**54x**デバイスによるサポート機能の異差**

	ソフトウェアPLL	飽和モード	拡張プログラム・アドレス
C541	×	×	×
LC541	×	×	×
LC541B	○	○	×
LC542	×	×	×
LC543	×	×	×
LC545A	○	○	×
LC546A	○	○	×
LC548	○	○	○
LC549	○	○	○
VC549	○	○	○

○ーサポートあり、
×ーサポートなし

D

付録E

用語集

A

A: **アキュムレータ**A**を参照。**

ABU: **自動バッファリング ユニットを参照。**

ABUC: ABU**制御レジスタ。自動バッファリング ユニットを制御するレジスタです。**

AG: **アキュムレータ**A **ガード ビット。アキュムレータ**A**のビット**39−32(**ガード ビット**)**から成る**8**ビット レジスタ。**

AH: **アキュムレータ**A**上位ワード。アキュムレータ**A**のビット**31−16**。**

AL: **アキュムレータ**A**下位ワード。アキュムレータ**A**のビット**15−0**。**

ALU: **算術論理演算ユニット。算術演算および論理演算を実行する、**CPU**の一部。**

AR0−AR7: **補助レジスタ**0−7**。**CPU**によってアクセスでき、補助レジスタ算術演算ユニット**(ARAU)**で変更可能な**8**つの**16**ビット レジスタ。主に、データ メモリのアドレッシングに使用されます。**

ARAU: **補助レジスタ算術ユニットを参照。**

ARP: **補助レジスタ ポインタを参照。**

ARR、ARR0、ARR1: 16**ビットの**ABU**アドレス受信レジスタ。自動バッファ ユニットが受信データのストアを開始する行き先アドレスを指定します。**

ASM: **アキュムレータ シフト モード フィールドを参照。**

AVIS: **アドレス可視モード ビットを参照。**

AXR、AXR0、AXR1: ABU**アドレス送信レジスタ。自動バッファ ユニットがデータ送信を開始するソース アドレスを指定する**16**ビット レジスタです。**

B

B: **アキュムレータ**B**を参照。**

BDRR、BDRR0、BDRR1: BSP**データ受信レジスタ。バッファド シリアル ポート経由でデータを受信するのに使用される**16**ビットのレジスタ。**BDRR0**はバッファド シリアル ポート**0**に対応し、**BDRR1**はバッファド シリアル ポート**1**に対応します。

BDXR、BDXR0、BDXR1: BSP**データ送信レジスタ。バッファド シリアル ポート経由でデータを送信するのに使用される**16**ビット レジスタ。**BDXR0**はバッファド シリアル ポート**0**に対応し、**BDXR1**はバッファド シリアル ポート**1**に対応します。

BG: アキュムレータB**ガード ビット。アキュムレータ**B**のビット**39−32**(ガード ビット)より成る**8**ビット レジスタ。**

BH: アキュムレータB**上位ワード。アキュムレータ**B**のビット**31−16**。**

BK: サーキュラ バッファ サイズ レジスタを参照。

BKR、BKR0、BKR1: ABU**受信バッファ サイズ レジスタ。自動バッファリング ユニットのための受信バッファのサイズを設定する**16**ビットのレジスタ。**

BKX、BKX0、BKX1: ABU**送信バッファ サイズ レジスタ。自動バッファリング ユニットのための送信バッファのサイズを設定する**16**ビットのレジスタ。**

BL: アキュムレータB**下位ワード。アキュムレータ**B**のビット**15−0**。**

BRAF: ブロックリピート アクティブ フラグを参照。

BRC: ブロックリピート カウンタを参照。

BRE: 自動バッファ受信部イネーブル ビットを参照。

BRINT、BRINT0、BRINT1: BSP**受信割り込みを参照。**

BRSR: BSP**データ受信シフト レジスタ。**BDR**ピンから受信されたシリアル データを保持する**16**ビット レジスタ。**BDRR**を参照。**

BSCR: バンク切り替え制御レジスタを参照。

BSP: バッファド シリアル ポート。自動バッファリング ユニット(ABU)**を持った拡張型同期シリアル ポート。連続動作中に**CPU**の消費電力を低減できます。**

BSP受信割り込み(BRINT、BRINT0、BRINT1): 割り込みフラグ レジスタ(IFR)**上のビット。**BSP**データ受信シフト レジスタ**(BRSR)**の内容が**BSP**データ受信レジスタ**(BDRR)**にコピーされたことを示します。**BRINT0**はバッファド シリアル ポート**0**に対応し、**BRINT1**はバッファド シリアル ポート**1**に対応します。**

BSP送信割り込み(BXINT、BXINT0、BXINT1): 割り込みフラグ レジスタ(IFR)**上のビット。**BSP**データ送信レジスタ**(BDXR)**の内容が**BSP**データ送信シフト レジスタ**(BXSR)**にコピーされたことを示します。**BXINT0**はバッファド シリアル ポート**0**に対応し、**BXINT1**はバッファド シリアル ポート**1**に対応します。**

BSPC、BSPC0、BSPC1: バッファド シリアル ポート制御レジスタ0および1。バッファド シリアル ポートのステータスおよび制御ビットを含む、16ビット レジスタです。BSPC0**はバッファド シリアル ポート**0**に対応し、**BSPC1**はバッファド シリアル ポート**1**に対応します。**

BSPCE、BSPCE0、BSPCE1: BSP制御拡張レジスタ。バッファド シリアル ポート(BSP)インターフェイスのステータス ビットおよび制御ビットを含む、16ビット レジスタです。BSPCEの下位10ビットはシリアル ポート インターフェイス制御専用で、上位6ビットは自動バッファ ユニット(ABU)制御に使用されます。

BXE: 自動バッファリング送信イネーブル ビットを参照。

BXSR: BSPデータ送信シフト レジスタ。BDXピンから送信されるシリアル データを保持する16ビット レジスタ。BDXRも参照。

C

C: キャリー ビットを参照。

C16: ステータス レジスタ1(ST1)上のビット。ALUがデュアル16ビット モードか倍精度モードのどちらで動作するかを決めます。

CAB: Cアドレス バス。データ メモリへのアクセスに必要なアドレスを転送するバス。

CB: Cバス。データ メモリからリードされるオペランドを転送するのに使用します。

CLKDV: 内部送信クロック ディビジョン ファクタを参照。

CLKP: クロック極性を参照。

CLKOUTオフ ビット(CLKOFF): プロセッサ モード ステータス レジスタ(PMST)上のビット。CLKOUT出力をイネーブルまたはディセーブルにします。

CMPT: 互換モードを参照。

CPL: コンパイラ モードを参照。

CSSU: 比較選択ストア ユニットを参照。

D

DAB: Dアドレス バス。データ メモリへのアクセスに必要なアドレスを転送するバス。

DABアドレス レジスタ: DB経由のデータ メモリ リードに際しDABに出力するアドレスを保持するレジスタ。

DAGEN: データ アドレス生成ロジック(DAGEN)を参照。

DAR: DABアドレス レジスタ。MVDMおよびMVKD命令のイミディエイト値によって表されるデータ メモリ オペランドのアドレスに使用するレジスタ。

DARAM: デュアルアクセスRAM。単一クロック サイクルで2回アクセスできるメモリ。

E

DB: Dバス。データ メモリからリードされるオペランドを転送するために使用するバス。

dma: 直接メモリ アドレスを参照。

DP: データ ページ ポインタを参照。

DRB: 直接データ メモリ アドレス バス。データ メモリのための直接アドレスを転送する16ビット バス。

DROM: データROMを参照。

DRR、DRR0、DRR1: シリアル ポート データ受信レジスタ。同期シリアル ポートを経由してデータを受信することに使う、16ビット レジスタ。DRR0は同期シリアル ポート0に対応し、DRR1は同期シリアル ポート1に対応します。

DSP割り込みビット(DSPINT): HPI制御レジスタ(HPIC)上のビット。ホスト デバイスから'54xへの割り込みをイネーブルかディセーブルにします。

DXR、DXR0、DXR1: シリアル ポート データ送信レジスタ。同期シリアル ポートを経由してデータを送信することに使う、2つの16ビット レジスタ。DXR0は同期シリアル ポート0に対応し、DXR1は同期シリアル ポート1に対応します。

E

EAB: Eアドレス バス。データ メモリへのアクセスのために必要なアドレスを転送するバス。

EABアドレス レジスタ(EAR): EB経由のデータ メモリ リードに際しEABに出力するアドレスを保持するレジスタ。

EB: Eバス。メモリにライトされるデータを転送する為に使用するバス。

F

FE: フォーマット拡張を参照。

FIG: フレーム無視を参照

Freeビット: シリアル ポート制御レジスタ(SPC)、バッファド シリアル ポート制御レジスタ(BSPC)、タイマ制御レジスタ(TCR)、TDMシリアル ポート制御レジスタ(TSPC)上のビット。Softビットとともに使用して、高級言語デバッガでブレークポイント時に、シリアル ポートまたはタイマ クロックの状態を決めます。Softビットも参照。

FSM: フレーム同期モードを参照。

FSP: フレーム同期極性を参照。

H

HALTR: **自動バッファ受信部停止を参照。**

HALTX: **自動バッファ送信部停止を参照。**

HINT: **’54xからホスト プロセッサへの割り込み。**HPI**制御レジスタ**(HPIC)**にあるビットで、’54xからホスト デバイスへの割り込みをイネーブルかディセーブルにします。**

HM: **ホールド モードを参照。**

HPIアドレス レジスタ(HPIA): 16**ビット レジスタ。ホスト ポート インターフェイス**(HPI)**メモリ ブロックのアドレスをストアします。**HPIA**のプリインクリメントまたはポストインクリメントが可能です。**

HPI制御レジスタ(HPIC): 16**ビット レジスタ。ホスト ポート インターフェイス**(HPI)**のステータス ビットおよび制御ビットを含んでいます。**

I

IFR: **割り込みフラグ レジスタを参照。**

IMR: **割り込みマスク レジスタを参照。**

IN0: **入力ビット**0**。シリアル ポート制御レジスタ**(SPC)**、バッファド シリアル ポート制御レジスタ**(BSPC)**、**TDM**シリアル ポート制御レジスタ**(TSPC)**上のビット。このビットにより、**CLKR**ピンを入力として使用できます。**IN0**は、デバイスの**CLKR**ピンの現在のレベルを反映した値となります。**

IN1: **入力ビット**1**。シリアル ポート制御レジスタ**(SPC)**、バッファド シリアル ポート制御レジスタ**(BSPC)**、**TDM**シリアル ポート制御レジスタ**(TSPC)**上のビット。このビットにより、**CLKX**ピンを入力として使用できます。**IN1**は、デバイスの**CLKX**ピンの現在のレベルを反映した値となります。**

IPTR: **割り込みベクタ ポインタ。プロセッサ モード ステータス レジスタ**(PMST)**の**9**ビット フィールド。割り込みベクタが**128**ワード毎のどのページにあるかを指し示します。**

IR: **命令レジスタ。フェッチされた命令を保持するために使用する**16**ビット レジスタ。**

L

LSB: **最下位ビット。ワードの最低順位のビット。**

M

MCM: **クロック モードを参照。**

MSB: **最上位ビット。ワードの最高順位のビット。**

O

OVA: **オーバーフロー フラグ**A**。ステータス レジスタ**0(ST0)**上のビット。アキュムレータ**A**のオーバーフロー状態を示します。**

OVB: **オーバーフロー フラグ**B**。ステータス レジスタ**0(ST0)**上のビット。アキュムレータ**B**のオーバーフロー状態を示します。**

OVLY: RAM**オーバーレイを参照。**

OVM: **オーバーフロー モード ビット。ステータス レジスタ**1(ST1)**上のビット。演算後に**ALU**がオーバーフローをどのように処理するかを指定します。**

P

PAB: **プログラム アドレス バスを参照。**

PAGEN: **プログラム アドレス生成ロジック**(PAGEN)**を参照。**

PAR: **プログラム アドレス レジスタ。**FIRS、MACD、MACP、MVDP、MVPD、READA、WRITA**命令においてプログラム メモリ オペランドのアドレッシングに使用される**16**ビット レジスタ。**PB**経由のメモリ リードに際し**PAB**に出力するアドレスを保持するレジスタ。**

PB: **プログラム データ バスを参照。**

PC: **プログラム カウンタを参照。**

PCM: **パルス コード変調モードを参照。**

pmad: 16**ビット イミディエイトのプログラム メモリ アドレス。**

PMST: **プロセッサ モード ステータス レジスタ。デバイスのメモリ構成を制御する**16**ビットのステータス レジスタ。**ST0、ST1**も参照。**

PRD: **タイマ ピリオド レジスタ。オンチップ タイマの周期を指定する**16**ビット レジスタ。**

R

RAMオーバーレイ(OVLY): プロセッサ モード ステータス レジスタ(PMST)上のビット。オンチップRAMをデータ空間だけでなくプログラム空間にもマッピングするか否かを決めます。

RC: リピート カウンタを参照。

REA: ブロック リピート終了アドレスを参照。

RH: 受信バッファ半受信を参照。

RINT、RINT0、RINT1: シリアル ポート受信割り込みを参照。

RRDY: 受信レディを参照。

RRST: 受信部リセットを参照。

RSA: ブロックリピート開始アドレスを参照。

RSR: データ受信シフト レジスタ。DRピンから受信したシリアル データを保持する16ビット レジスタ。データ受信レジスタ(DRR)も参照。

RSRFULL: 受信シフト レジスタ フル ビットを参照。

RTN: 高速リターン レジスタを参照。

S

SARAM: シングル アクセスRAM:1クロック サイクル間でリードまたはライトが1度だけ行えるメモリ。

SMUL: 乗算飽和を参照。

Softビット: シリアル ポート制御レジスタ(SPC)、バッファド シリアル ポート制御レジスタ(BSPC)、タイマ制御レジスタ(TCR)、TDMシリアル ポート制御レジスタ(TSPC)上のビット。Freeビットとともに使用して、高級言語デバッガでブレークポイント時に、シリアル ポートまたはタイマ クロックの状態を決めます。Freeビットも参照。

SP: スタック ポインタを参照。

SPC、SPC0、SPC1: シリアル ポート制御レジスタ。同期シリアル ポートのステータス ビットおよび制御ビットを含んでいます。SPC0は同期シリアル ポート0に対応し、SPC1は同期シリアル ポート1に対応します。

SST: ストア飽和を参照。

ST0: ’54xのステータス ビットと制御ビットを含む16ビット レジスタ。PMST、ST1を参照。

ST1: ’54xのステータス ビットと制御ビットを含む16ビット レジスタ。PMST、ST0を参照。

SXM: 符号拡張モードを参照。

T

TADD: TDMアドレス。シングル双方向アドレス ライン。4ワイヤ シリアル バス上で、どのデバイスがTDMデータ(TDAT)ラインのデータをリードするかを指定します。

TC: テスト/制御フラグ。ステータス レジスタ0(ST0)上のビット。テスト動作によって影響を受けます。

TCLK: TDMクロック。TDM動作のためのシングル双方向クロック ライン。

TCR: タイマ制御レジスタ。オンチップ タイマのステータス ビットと制御ビットを含む、16ビット メモリマップド レジスタ。

TCSR: TDMチャネル選択レジスタ。8つのタイム スロット(チャネル)中のどのスロットで、4ワイヤ シリアル バス上のデバイスが送信されるかを指定する、16ビット メモリマップド レジスタ。

TDAT: TDMデータ。シングル双方向ライン。ここからすべてのTDMデータが転送されます。

TDM受信割り込み(TRINT): 割り込みフラグ レジスタ(IFR)上のビット。TDMデータ受信シフト レジスタ(TRSR)の内容がTDMデータ受信レジスタ(TRCV)にコピーされたことを示します。

TDM送信割り込み(TINT): 割り込みフラグ レジスタ(IFR)上のビット。TDMデータ送信レジスタ(TDXR)の内容がデータ送信シフト レジスタ(XSR)にコピーされたことを示します。

TDXR: TDMデータ送信レジスタ。TDMシリアル ポートを経由してデータを送信するために使用される16ビット レジスタ。XSRも参照。

TIM: タイマ カウンタ レジスタ。オンチップ タイマの現在のカウントを指定する、16ビット メモリマップド レジスタ。

TINT: タイマ割り込みを参照。

TRAD: TDM受信アドレス レジスタ。TDMシリアル ポートのTADDラインの状態に関する情報を示す、16ビット メモリマップド レジスタ。

TRCV: TDM**データ受信レジスタ。**TDM**シリアル ポートを経由してデータを受信するために使用するレジスタ。**

TRINT: TDM**受信割り込みを参照。**

TRN: **トランジッション レジスタを参照。**

TRSR: TDM**データ受信シフト レジスタ。**TDM**データ**(TDAT)**ラインから受信されたシリアル データを保持する**16**ビット レジスタ。**TRCV**も参照。**

TRTA: TDM**送受信アドレス レジスタ。このレジスタの下位半分は、デバイスの受信アドレスを指定し、上位半分は送信アドレスを指定します。**

TSPC: TDM**シリアル ポート制御レジスタ。**TDM**シリアル ポートのステータス ビットと制御ビットを含む**16**ビット メモリマップド レジスタ。**

TXINT: TDM**送信割り込みを参照。**

TXM: **送信モードを参照。**

X

XF: XF**ステータス フラグ。ステータス レジスタ**1(ST1)**のビット。**XF**ピンの状態を示します。**

XH: **送信バッファ半送信を参照。**

XINT、XINT0、XINT1: **シリアル ポート送信割り込みを参照。**

XPC: **拡張プログラム カウンタ。現在のプログラム メモリ アドレスの上位**7**ビットを示すレジスタ。**

XRDY: **送信レディを参照。**

XRST: **送信部リセットを参照。**

XSR: **データ送信シフト レジスタ。**DX**ピン(**またはTDM=1**のときの**TDX**ピン)から送信されるシリアル データを保持する**16**ビット レジスタ。**TDXR**も参照。**

$\overline{\text{XSREMPTY}}$: **送信シフト レジスタ エンプティを参照。**

Z

ZA: **ゼロ検出**A。**アキュムレータ**A**に**0**が入っていることを示す信号。**

ZB: **ゼロ検出**B。**アキュムレータ**B**に**0**が入っていることを示す信号。**

あ

アキュムレータ: 40**ビット レジスタ。処理の結果をストアしてそれに続く算術論理演算ユニット**(ALU)**の処理の入力を可能にします。**

アキュムレータ シフト モード フィールド(ASM): **ステータス レジスタ**1(ST1)**の**5**ビット フィールド。並列ロードやストアをともなうような命令の実行時に、アキュムレータ値をシフトするためのシフト値**(−16**から**15)**を指定します。**

アドレス: **メモリ内のワードの位置。**

アドレス バス: **アドレスの伝送経路として使用する配線の束。**'54x**には、**4**つの**16**ビット アドレス バスとして**CAB、DAB、EAB、PAB**があります。**

アドレス可視モード(AVIS)ビット: **プロセッサ モード ステータス レジスタ**(PMST)**上のビット。内部のプログラム アドレスがデバイスの外部のアドレス バス ピンに現れるかどうかを決めます。**

アドレッシング モード: **命令がメモリ内の対象の位置を計算する方法。**

アナログ−デジタル(A/D)コンバータ: **アナログ信号をデジタル信号に変換する回路。**

ウェイト ステート: **外部プログラム、データ、**I/O**メモリが外部メモリにリード/ライトを行うとき、**CPU**がそれに応答するまでの待ち時間(周期)。**CPU**は、各ウェイト ステートは**1**サイクル**(1CLKOUT**サイクル**)**単位で挿入できます。**

ウォーム ブート: **前もってロードされているプログラムのエントリ ポイントに制御を移す手順。**

オーバーフロー: **この状態は、算術演算の結果が、その結果を保持するために使用されるレジスタの容量を超えている場合に発生します。**

オーバーフロー フラグ: **算術演算の結果が対応するレジスタの容量を超えたか否かを示すフラグ。**

か

外部割り込み: ピン($\overline{INT0}$–$\overline{INT3}$)によってトリガされるハードウェア割り込み。

加算器: 2つの数値を加算または減算するユニット。

キャリー ビット(C): ステータス レジスタ0(ST0)上のビット。拡張算術演算、アキュムレータ シフトおよびローテイトの動作においてALUが使用します。キャリー ビットは、条件付き命令によってテストできます。

共有アクセス モード(SAM): このモードでは、'54xとホストの双方がHPIメモリにアクセスできます。また非同期ホスト アクセスは内部で同期され、さらに'54xとホストのアクセスが重なった際は、ホストが優先され'54xは1サイクル待ちます。

共有アクセス モード ビット(SMOD): HPI制御レジスタ(HPIC)のビット。共有アクセス モード(SAM)をイネーブルまたはディセーブルにします。**共有アクセス モード(SAM)およびホスト専用モード(HOM)**も参照。

クロック モード ビット(MCM): シリアル ポート制御レジスタ(SPC)、バッファド シリアル ポート制御レジスタ(BSPC)、TDMシリアル ポート制御レジスタ(TSPC)上のビット。CLKXのクロック ソースを指定します。

クロック極性ビット(CLKP): BSP制御拡張レジスタ(BSPCE)上のビット。いつ受信部がデータをサンプリングして、送信部によって送り出すかを示します。

高速リターン レジスタ(RTN): 割り込み(RETF[D])命令から高速で復帰するためのリターン アドレスを保持するために使われる、16ビット レジスタ。

高速フーリエ変換(FFT): フーリエ変換(関数を、時間領域と周波数領域との間で変換させる)を、効率的に処理するための手法。時間から周波数領域への変換を正変換、周波数から時間領域への変換を逆変換と呼びます。**バタフライ**も参照。

コード: タスクを実行するために書かれる命令セット。コンピュータ プログラムまたはプログラムの一部。

コールド ブート: 電源投入時にプログラムメモリに対してプログラムをロードする手順。

互換モード ビット(CMPT):ステータス レジスタ1(ST1)上のビット。シングル間接アドレッシング モードで補助レジスタ ポインタ(ARP)を使って補助レジスタを選択するかどうかを決めます。

コンパイラ モード ビット(CPL): ステータス レジスタ1(ST1)上のビット。直接アドレッシング モードでCPUがデータ メモリ アドレスを生成するのに、データ ページ ポインタとスタック ポインタのどちらを使うかを決めます。

さ

サーキュラ バッファ サイズ レジスタ(BK): 補助レジスタ算術演算ユニット(ARAU)がサーキュラ アドレッシングでデータ ブロック サイズを指定するために使用する16ビット レジスタです。

指数エンコーダ(EXP): アキュムレータの指数値を計算するハードウェア デバイス。

自動バッファ受信部イネーブル ビット(BRE): BSP制御拡張レジスタ(BSPCE)上のビット。自動バッファ受信部をイネーブルかディセーブルにします。

自動バッファ受信部停止 ビット(HALTR): BSP制御拡張レジスタ(BSPCE)上のビット。現行のバッファの半分が受信された時に、自動バッファ受信部をイネーブルかディセーブルにします。

自動バッファ送信部イネーブル ビット(BXE): BSP制御拡張レジスタ(BSPCE)上のビット。自動バッファ送信部をイネーブルかディセーブルにします。

自動バッファ送信部停止ビット(HALTX): BSP制御拡張レジスタ(BSPCE)上のビット。現在のバッファの半分が送信された時に、自動バッファ送信部をイネーブルかディセーブルにします。

自動バッファリング ユニット: 同期シリアル ポートの機能拡張。CPUから独立し、同期シリアル ポートにデータのリード/ライトを行います。

シフタ: ワード内のビットを左右にシフトするハードウェア ユニット。

時分割多重処理: 最大8つの'54xデバイスがシングル シリアル バスを共有するプロセス。各デバイスは、このバスを使って順番に通信を行います。総計8つのタイム スロット(チャネル)を備えています。タイム スロット使用時は、バス上のどのような組み合わせのデバイスでも通信できます。

時分割多重ビット(TDM): TDMシリアル ポートをイネーブルまたはディセーブルにする、TDMシリアル ポート制御レジスタ(TSPC)上のビット。

受信シフト レジスタ フル ビット(RSRFULL): シリアル ポート制御レジスタ(SPC)、バッファド シリアル ポート制御レジスタ(BSPC)上のビット。シリアル ポート受信部がオーバーランしたかどうかを示します。

受信バッファ ハーフ受信ビット(RH): BSP制御拡張レジスタ(BSPCE)上のビット。受信バッファの半分が受信されたことを示します。

受信レディ(RRDY): シリアル ポート制御レジスタ(SPC)、バッファド シリアル ポート制御レジスタ(BSPC)、TDMシリアル ポート制御レジスタ(TSPC)上のビット。0から1に変化することで、データ受信シフト レジスタ(RSR)の内容がデータ受信レジスタ(DRR)にコピーされ、そのデータがリード可能なことを示します。

乗算器: 17ビット × 17ビット乗算器。32ビットの積を生成します。乗算器は乗算を1サイクルで実行し、符号付きまたは符号なしの2の補数を使った算術演算を実行します。

乗算飽和ビット(SMUL): プロセッサ モード ステータス レジスタ(PMST)上のビット。MACまたはMAS命令で関和演算を実行する前に、乗算結果の飽和を行うかどうかを決めます。

小数モード ビット(FRCT): ステータス レジスタ1(ST1)上のビット。乗算器出力の値を1ビット左シフトするか否かを決めます。

シリアル ポート インターフェイス: オンチップの全二重シリアル ポート インターフェイス。コーデックやシリアル アナログ−デジタル(A/D)、デジタル−アナログ(D/A)コンバータなどの必要最小限の外部のハードウエアで、シリアル デバイスとの直接シリアル通信を可能にします。シリアル ポートの状態と制御は、シリアル ポート制御レジスタ(SPC)で指定します。

シリアル ポート受信割り込みビット(RINT、RINT0、RINT1): 割り込みフラグ レジスタ(IFR)上のビット。データ受信シフト レジスタ(RSR)の内容がデータ受信レジスタ(DRR)にコピーされたことを示します。RINT0は同期シリアル ポート0に対応して、RINT1は同期シリアル ポート1に対応します。

シリアル ポート送信割り込みビット(XINT、XINT0、XINT1): 割り込みフラグ レジスタ(IFR)上のビット。データ送信レジスタ(DXR)の内容がデータ送信シフト レジスタ(XSR)にコピーされたことを示します。XINT0は同期シリアル ポート0に対応して、XINT1は同期シリアル ポート1に対応します。

スタック ポインタ(SP): スタックにプッシュされた最後のデータを常にポイントするレジスタ。

スタック: サブルーチンと割り込みサービス ルーチンのためのリターン アドレスや、データを待避するために使用されるメモリ ブロック。

ストア飽和ビット(SST): プロセッサ モード ステータス レジスタ(PMST)上のビット。アキュムレータからのデータに飽和処理を行うかどうか、そのデータがメモリにストアされる前に判断します。

ゼロ検出: ZAおよびZBを参照。

ゼロづめ: 16ビットの値を32ビット フィールドにロードするとき、上位または下位ビットをゼロで充たす方法。

送信シフト レジスタ エンプティ($\overline{\text{XSREMPTY}}$): シリアル ポート制御レジスタ(SPC)、バッファド シリアル ポート制御レジスタ(BSPC)上のビット。シリアル ポート送信部がアンダーフローかどうかを示します。

送信バッファ ハーフ送信ビット(XH): BSP制御拡張レジスタ(BSPCE)上のビット。送信バッファの半分を送信したことを示します。

送信部リセット ビット(XRST): シリアル ポート制御レジスタ(SPC)、バッファド シリアル ポート制御レジスタ(BSPC)、TDMシリアル ポート制御レジスタ(TSPC)上のビット。シリアル ポート送信部をリセットします。

送信モード ビット(TXM): シリアル ポート制御レジスタ(SPC)、バッファド シリアル ポート制御レジスタ(BSPC)、TDMシリアル ポート制御レジスタ(TSPC)上のビット。フレーム同期送信(FSX)パルスのソースを指定します。

送信レディ ビット(XRDY): シリアル ポート制御レジスタ(SPC)、バッファド シリアル ポート制御レジスタ(BSPC)、TDMシリアル ポート制御レジスタ(TSPC)上のビット。0から1に変化することで、データ送信レジスタ(DXR)の内容がデータ送信シフト レジスタ(XSR)にコピーされ、そのデータを新たなデータ ワードとともにロードできる状態にあることを示します。

ソフトウェア ウェイトステート レジスタ(SWWSR): プログラム、データ、I/O空間の外部メモリのために、ウェイト ステートの数を選択する16ビット レジスタ。

ソフトウェア 割り込み(SINT): INTRまたはTRAP命令によって発生する割り込み。

た

タイマ プリスケーラ カウンタ(PSC): タイマ制御レジスタ(TCR)上の４ビット。オンチップ タイマのカウントを示します。

タイマ再ロード ビット ビット(TRB): タイマ制御レジスタ(TCR)上のビット。オンチップ タイマをリセットします。

タイマ停止ステータス(TSS): タイマ制御レジスタ(TCR)のビット。オンチップ タイマを停止および再始動します。

タイマ分割レジスタ(TDDR): タイマ制御レジスタ(TCR)上の4ビット フィールド。オンチップ タイマのタイマ分割率(周期)を指定します。

タイマ割り込み(TINT): 割り込みフラグ レジスタ(IFR)上のビット。タイマ カウンタ レジスタ(TIM)がデクリメントされて0になったことを示します。

直接データ メモリ アドレス バス: データ メモリの直接アドレスを転送する16ビット バス。

直接メモリ アドレス(dma): 直接アドレス命令の下位7ビット。データ ページ ポインタ(DP)と連結して、データ メモリ全体をアドレッシングします。データ ページ ポインタも参照。

データ アドレス バス: データ メモリ アドレスの伝送経路に使用する配線の束。'54xには、データ メモリ アドレスを転送するためにCAB、DAB、EABの3つの16ビット バスがあります。

データ アドレス生成ロジック(DAGEN): データ メモリのリード/ライトのためにアドレスを生成するロジック回路。プログラム アドレス生成ロジック(PAGEN)も参照してください。

データ バス: データの伝送経路として使用する配線の束。'54xにはCB、DB、EBの3つの16ビット データ バスがあります。

データ ページ ポインタ(DP): ステータス レジスタ0(ST0)上の9ビット フィールド。直接アドレッシングのために現在選択されているのは、512(128x16ワード)ページ中のどれかを示します。DPには、データ メモリ アドレスの上位9ビットを割り当て、dmaでは下位の7ビットを割り当てています。dmaも参照してください。

データ メモリ: データをストアし、処理するために使用するメモリ領域。データ メモリではアドレス00h−1FhにはCPUレジスタが割り当てられており、アドレス20h−5Fhにはペリフェラル レジスタが割り当てられています。

データROM(DROM): プロセッサ モード ステータス レジスタ(PMST)上のビット。オンチップROMの一部がデータ空間にマッピングされるかどうかを決めます。

デジタル ループバック モード: 同期シリアル ポートのテスト時の動作モード。DLBで受信ピンと同じデバイス上の送信ピンとを接続して、ポートが適正に動作しているかをテストできます。

デジタル ループバック モード(DLB)ビット: シリアル ポート制御レジスタ(SPC)、バッファド シリアル ポート制御レジスタ(BSPC)、TDMシリアル ポート制御レジスタ(TSPC)上のビット。シリアル ポートをデジタル ループバック モードにします。

デジタル−アナログ(D/A)コンバータ: デジタル信号をアナログ信号に変換する回路。

テンポラリ レジスタ(T): 乗算およびストア命令のオペランド、加算および減算命令のダイナミック シフト カウント、またはビット テスト命令のための動的ビット位置を保持する16ビット レジスタ。

トランジッション レジスタ(TRN): ビタビ アルゴリズムを実行のために新たなメトリクスへのパスのための移行決定を保持する、16ビット レジスタ。

な

内部送信クロック ディビジョン ファクタ(CLKDV): BSP制御拡張レジスタ(BSPCE)上の5ビット フィールド。内部送信クロックのデューティ サイクルを決めます。

ネスト割り込み: 現在の割り込みサービス・ルーチン(ISR)が完了する前に実行する必要がある、優先順位の高い割り込み。実行されているISRで割り込みがマスク・レジスタ(IMR)ビットをセットし、別の割り込みによって実行が中止されるのを防ぐことができます。

ノンマスカブル割り込み: 割り込みマスク レジスタ(IMR)でもマスクされず、ステータス レジスタ1(ST1)によってもディセーブルにならない割り込み。

は

バースト モード: 同期シリアル ポートの動作モード。このモードでは、フレーム同期パルス(FSXおよびFSR)に続いてシングル ワードが送信されます。

ハードウェア割り込み: オンチップ ペリフェラルまたは外部デバイスとを物理的に接続し、それを経由してトリガされる割り込み。

パイプライン: 命令をアセンブリ ラインの形式で実行する手法。

バタフライ演算: Nポイントの高速フーリエ変換(FFT)を処理するためのカーネル ファンクション。Nは2のべき乗になります。入力の組み合わせパターンがバタフライ(蝶)の羽の動きに例えられています。

パルス コード変調モード ビット(PCM): BSP制御拡張レジスタ(BSPCE)上のビット。BSP送信部をイネーブルかディセーブルにします。

バレル シフタ: ワードのビットをシフトさせるユニット。

バンク切り替え制御レジスタ(BSCR): 外部メモリのバンク サイズを指定する16ビット レジスタ。アクセスがメモリ バンク境界を超えるとき、追加サイクルの自動挿入をイネーブル、またはディセーブルにします。

汎用入出力ピン: 外部デバイスからの入力信号、または外部デバイスへの出力信号を供給するために使用するピン。これらのピンは、特定目的に利用されるのでなく、様々な目的の入出力信号を可能にします。これらのピンには、汎用$\overline{\text{BIO}}$入力ピンおよびXF出力ピンが含まれます。

比較選択記憶ユニット(CSSU): ビタビ操作での加算/比較/選択演算専用の特定アプリケーション向けハードウェア ユニット。

ブート: プログラムをプログラム メモリにロードするプロセス。

ブート ローダ: 内蔵されたコード セグメント。リセット後に、コードを外部ソースからプログラム メモリに転送します。

フォーマット ビット(FO): シリアル ポート制御レジスタ(SPC)、バッファド シリアル ポート制御レジスタ(BSPC)、TDMシリアル ポート制御レジスタ(TSPC)上のビット。シリアル ポート送受信部のワード長を指定します。

フォーマット拡張ビット(FE): BSP制御拡張レジスタ(BSPCE)上のビット。フォーマット ビット(FO)とともに使用して、BSPシリアル ポート送受信部のワード長を指定します。

符号拡張: 数値の最上位ビットを符号ビットで満たす処理。

符号拡張モード ビット(SXM): ステータス レジスタ1(ST1)上のビット。CPU演算で符号拡張をイネーブルにします。

符号制御ロジック: データ ビット符号あり/符号なし拡張するために使用する回路。乗算器、ALU、シフタの入力データ フォーマットに合わせて拡張します。

プッシュ: スタックにワードを置く操作。

フレーム同期モード ビット(FSM): シリアル ポート制御レジスタ(SPC)、バッファド シリアル ポート制御レジスタ(BSPC)、TDMシリアル ポート制御レジスタ(TSPC)上のビット。フレーム同期パルス(FSXおよびFSR)がシリアル ポート動作に必要か否かを指定します。

フレーム同期極性ビット(FSP): BSP制御拡張レジスタ(BSPCE)上のビット。フレーム同期パルス(FSXおよびFSR)の状態を決めます。

フレーム無視ビット(FIG): BSP制御拡張レジスタ(BSPCE)上のビット。連続モードにおいての受信、および連続モードで外部のフレームを用いた送信にのみ使用されます。

プログラム アドレス バス(PAB): プログラム メモリの読み書きのためのアドレスを可能にする16ビット バス。

プログラム アドレス発生ロジック(PAGEN): プログラム メモリのリード/ライトのためのアドレス、および2つのデータ オペランドが必要となる命令のデータ メモリのアドレスを生成するロジック回路。この回路は、1マシン サイクル当たりでアドレスを生成できます。データ アドレス生成ロジック(DAGEN)も参照。

プログラム カウンタ(PC): 次に実行される命令の位置を示す16ビット レジスタ。

プログラム コントローラ: 命令のデコード、パイプラインの管理、実行ステータスのストア、条件付き実行のデコードを行う、論理回路。

プログラム データ バス(PB): プログラム メモリから、命令コードおよびイミディエイト オペランドを転送するバス。

プログラム メモリ: プログラムをストアおよび実行するために使用されるメモリ領域。

ブロックリピート アクティブ フラグ ビット(BRAF): ステータス レジスタ1(ST1)上のビット。ブロック リピートが現在アクティブか否かを示します。

ブロック リピート開始アドレス レジスタ(RSA): 繰り返されるコード セグメントの開始アドレスを含む16ビットのメモリ マップド レジスタ。

ブロックリピート カウンタ(BRC): ブロック リピートの実行時にコード ブロックが繰り返される回数を指定する16ビット レジスタ。

ブロックリピート終了アドレス レジスタ(REA): 繰り返されるコード セグメントの終了アドレスを表す16ビットのメモリ マップド レジスタ。

ホールド モード ビット(HM): ステータス レジスタ1(ST1)上のビット。通常モードまたはコンカレント モードでCPUがホールド状態になるか否かを決めます。

補助レジスタ: 8つの16ビット レジスタ(AR7−AR0)。データ空間のアドレスに対するポイントとして使用されます。これらのレジスタは、補助レジスタ算術演算ユニット(ARAU)の処理に使用され、補助レジスタ ポインタ(ARP)によって選択されます。補助レジスタ算術演算ユニットも参照してください。

補助レジスタ ファイル: 8つの16ビット補助レジスタを含むデータ メモリ中の領域。補助レジスタを参照。

補助レジスタ ポインタ ビット(ARP): ステータス レジスタ0(ST0)上の3ビット フィールド。デバイスが'C5x/'C2xx互換モードで動作しているとき、現在選択されている補助レジスタを指すポインタとして使用されます。

補助レジスタ算術演算ユニット: 符号なし、16ビット算術論理演算ユニット(ALU)。補助レジスタを使って間接的にアドレスを計算するために使用します。

ホスト ポート インターフェイス(HPI): 8ビットのパラレル インターフェイス。CPUがホスト プロセッサと通信するために使用します。

ホスト専用モード(HOM): このモードでは、'54xがIDLE2(すべての内部クロックが停止)またはリセット モードのときに、ホストがHPIメモリにアクセスできます。

ポップ: スタックからワードを削除する操作。

ま

マイクロ スタック: プログラム アドレス生成ロジックを使ってデータ空間に連続アドレスを生成するとき、フェッチされる次の命令のアドレスを一時的にストアしておくスタック。

マイクロコンピュータ モード: このモードでは、オンチップROMがイネーブルになり、プログラム アクセスに対してアドレス可能になります。

マイクロプロセッサ モード: このモードでは、オンチップROMがプログラム アクセスに対してディセーブルになります。

マイクロプロセッサ/マイクロコンピュータ ビット(MP/$\overline{\text{MC}}$): プロセッサ モード ステータス レジスタ(PMST)上のビット。プロセッサがマイクロプロセッサあるいはマイクロコンピュータのいずれかのモードで動作していることを示します。マイクロコンピュータ モード、マイクロプロセッサ モードも参照。

マスカブル割り込み: ソフトウェアによってイネーブルまたはディセーブルにできるハードウェア割り込み。

メモリ マップ: '54xによってアクセスされるアドレス可能なメモリ空間のマップ。機能(メモリ、レジスタなど)に応じて分割できます。

メモリマップド レジスタ: '54xのレジスタで、データ メモリ空間のページ0にマッピングされています。

ら

リセット: レジスタおよび制御ビットを事前に決められた値にセットし、指定のアドレスから実行開始するように知らせることにより、CPUを既知の状態にする方法。

リピート カウンタ(RC): シングル命令を実行する回数を指定するために使用される16ビットのレジスタ。

レイテンシ: ある状態を発生させてから、デバイスがそれに反応するまでの時間的なずれ(遅延)。パイプラインにおいては、2つの命令の実行の間に必要な遅延のことで、2番目の命令に適正な値が使用されるように遅延を必要とします。

レシーバ リセット(RRST): シリアル ポート制御レジスタ(SPC)、バッファド シリアル ポート制御レジスタ(BSPC)、TDMシリアル ポート制御レジスタ(TSPC)上のビット。シリアル ポート受信部をリセットします。

レジスタ: 一時的にデータを保持、またはデバイスの状態を制御 指定するために使用される、ビットのグループ。

連続モード: 一同時シリアル・ポートの動作モード。このモードでは、最大周波数において複数のパケットを送信するのに必要なフレーム同期パルス(FSXおよびFSR)はひとつだけです。

わ

割り込み: CPU外部のイベントまたは以前に実行された命令によって発生する状態。現在のプログラムを強制的に停止して、プロセッサにその割り込みに対応するサービス ルーチンを実行させます。

割り込みサービス ルーチン(ISR): ハードウェアまたはソフトウェア割り込みに応答して実行されるコード モジュール。

割り込みフラグ レジスタ(IFR): 16ビット メモリマップド レジスタ。実行されていない割り込みにフラグをセットします。

割り込みマスク レジスタ(IMR): 16ビット メモリマップド レジスタ。外部および内部割り込みをマスクします。

割り込みモード ビット(INTM): ステータス レジスタ1(ST1)上のビット。すべての割り込みをグローバルにマスクまたはイネーブルにします。

索引

記号

数字

A

B

Index

C

D

E

F

H

I

J

M

O

P

R

S

T

X

Z

あ

Index

い

う

え

お

か

Index

き

く

け

こ

さ

Index

し

Index

Index

す

せ

Index

そ

た

ち

Index

Index

Index

へ

ほ

Index

ま

み

め

も

ゆ

り

れ

ろ

わ

Index

Index