参 考 資 料

TEXAS INSTRUMENTS

www.tij.co.jp

JAJA142– 2月 2008年

C6000ペリフェラルズリファレンス・ガイドに関する参考資料(内部メモリー及びDMA)

アプリケーション技術部

アブストラクト

本資料は、日本語版TMS320C6000ペリフェラルズリファレンス・ガイドのうち「第2章 TMS320C6201/C6701 プログラム及びデータ・メモリー」、「第3章 TMS320C6202/C6202 プログラム及びデータ・メモリー」、「第5章 ダイレクト・メモリー・アクセス(DMA)」を抜粋したものです。

本文章について

本資料は、”TMS320C6000 Peripherals Reference Guide”(spru190c)を翻訳したTMS320C6000ペリフェラルズリファレンス・ガイド(spru537)の「第2章 TMS320C6201/C6701 プログラム及びデータ･メモリー」、「第3章 TMS320C6202/C6202 プログラム及びデータ･メモリー」、「第5章 ダイレクト･メモリー･アクセス(DMA)」を抜粋したものです。

元となった英語版文書はのちに改定され、現在はC6000デバイスのペリフェラルの概要のみを述べる資料となっています。ペリフェラルの詳細説明に関しては、ペリフェラル固有のリファレンス･ガイドを用意しています。

本資料は、日本語でのペリフェラル理解の手助けのために、英語版ペリフェラル・リファレンス・ガイドの参考資料として用意しました。ペリフェラルの詳細につきましては、必ず最新の英語版リファレンス･ガイド及びデータシートをご参照ください。

参考文献

1. TMS320C6000 DSP Peripherals Overview Reference Guide (SPRU190)
2. TMS320C620x/C670x DSP Program & Data Memory Controller/DMA Controller Reference Guide (SPRU577)

第2章

TMS320C6201/C6701プログラム及びデータ・メモリー

この章では、プログラム・メモリーの構成、プログラム・メモリーとキャッシュ・モード、そしてDMAを介したプログラム・メモリーへのアクセスについて解説します。

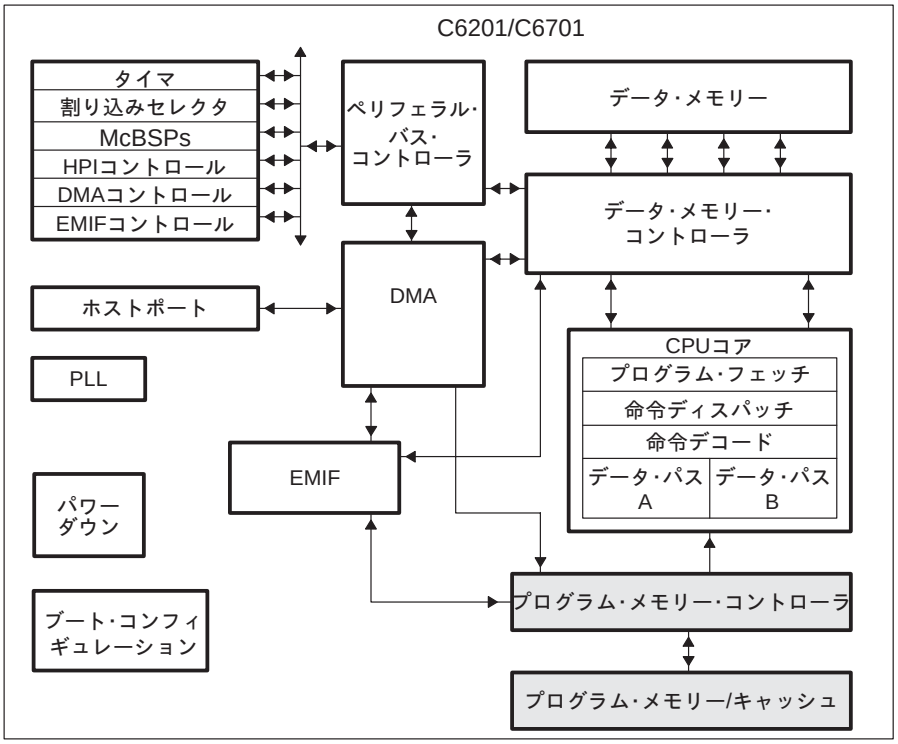
Topic **Page**

2.1 プログラム·メモリー·コントローラ

図2−1に示す、プログラム·メモリー·コントローラは次のような処理を行います。

- ❑ 内部プログラム·メモリーに対するCPU及びDMAからのリクエストを処理し、必要に応じてアービトレーションを実行します。
- ❑ 外部メモリー·インターフェイス(EMIF)を介して、外部メモリーに対するCPUリクエストを処理します。
- ❑ キャッシュとして設定された場合には、内部プログラム·メモリーの管理を行ないます。

図2−1. TMS320C6201/C6701プログラム·メモリー·ブロック図



2.2 内部プログラム･メモリー

内部プログラム･メモリーは、64KバイトのRAMから成り、2Kの256ビット･フェッチ･パケットもしくは16Kの32ビット命令を格納することができます。CPUは、プログラム･メモリー･コントローラを介して、内部プログラム･メモリーに256ビット幅で接続されており、1フェッチ･パケットを1サイクルでフェッチすることができます。

2.2.1 内部プログラム･メモリー･モード

内部プログラム･メモリーについては、CPUコントロール･ステータス･レジスタ(CSR)中のプログラム･キャッシュ･コントロール(PCC)フィールド(bits 7−5)の設定により、表2−1に示す４つのモードを選択することができます。設定可能なモードは次のとおりです。

- **マップド:** 選択されたメモリー･マップに応じて、プログラム･メモリーは以下のいずれかのアドレスに設定されます。
  - 0000 0000h−0000 FFFFh (Map1)
  - 0140 0000h−0140 FFFFh (Map0)

  マップド･モードが選択されると、内部プログラム･メモリーからのプログラムフェッチにより、そのアドレスに置かれたフェッチ･パケットが返されます。その他のモードでは、CPUのこの領域へのアクセスにより、未定義のデータが返されます。マップド･モードは、リセット時の内部プログラム･メモリーのデフォルト設定となっています。(メモリー･マップを選択する方法については、「第10章 **ブート･モード及びコンフィギュレーション**」をご覧ください。)

- **キャッシュ･イネーブル:** キャッシュ･イネーブル･モードでは、任意のアドレスに対する最初のプログラム･フェッチはキャッシュ･ミスを起こします。キャッシュ･ミスが起こると、フェッチ･パケットが外部メモリー･インターフェイス(EMIF)から32ビットずつロードされ、内部キャッシュ･メモリーにストアされます。フェッチ･パケットがロードされている間、CPUは停止します。この場合に必要とされるウェイト･ステートの数は、使用する外部メモリーの種類、メモリーの状態、DMAやCPUのデータ･アクセスなどの他のリクエストとの競合状態によって異なります。一度フェッチしたアドレスに対する２回目以降のリードに対しては、キャッシュがヒットし、ウェイト･ステートの挿入なしに、フェッチ･パケットが内部プログラム･メモリーからCPUに送られます。プログラム･メモリー･モードからキャッシュ･イネーブル･モードへ切り替えると、プログラム･キャッシュの内容がフラッシュされます。このようなモードの変更によってのみ、キャッシュのフラッシュが可能となっています。

- **キャッシュ･フリーズ:** キャッシュ･フリーズの間、キャッシュはその状態を保持し続けます。フリーズ状態にあるキャッシュからのリードは、キャッシュ･ミス時に外部メモリー･インターフェイスからリードされたデータがキャッシュの中にストアされない点を除いては、イネーブル状態にあるキャッシュからのリード動作と違いがありません。キャッシュ･ミスしたアドレスに対するリードは常にキャッシュ･ミスとなり、データは再び外部メモリーからフェッチされます。

キャッシュをフリーズさせることにより、キャッシュの中にある重要なプログラム･データへの上書きを防止することができます。

- **キャッシュ･バイパス:** このモードでは常に外部メモリーのデータがフェッチされます。このデータはキャッシュ･メモリーにはストアされません。キャッシュのフリーズのように、キャッシュ･バイパスの間も、キャッシュはその状態を保持し続けます。このモードは、プログラム･データが外部メモリーからフェッチされることを保証します。

表2−1. 内部メモリー･モードのまとめ

内部プログラム･メモリー･モード	PCCの値	解説
マップド	000	キャッシュ･不使用(リセットにおけるデフォルト状態)
キャッシュ･イネーブル	010	キャッシュ･アクセス、キャッシュ･ミス時に更新
キャッシュ･フリーズ	011	キャッシュ･アクセス、キャッシュ･ミス時に更新なし
キャッシュ･バイパス	100	キャッシュ･アクセスなし、キャッシュ･ミス時に更新なし
	その他	予約

注:

PMEMCの動作モードを変更するときは、PMEMCの正常動作を保証するために以下のようなアセンプリ･ルーチンを使用してください。このルーチンはキャッシュを有効にします。キャッシュ･イネーブル以外の状態にPMEMCの動作モードを変更するには、下のルーチンの4行目を修正してください。ここでは、設定したいPCCの値をB5に入れています。例えば、キャッシュをマップド･モードにするにはB5に0000hを入れることになります。この例で使用されているCPUレジスタは重要な意味を持っていません。A0–A15またはB0–B15のどのレジスタでも使うことができます。

```
   .align   32
   MVC.S2   CSR,B5    ;copy control status register
|| MVK.S1   0xff1f.A5
   AND.L1x  A5,B5,A5  ;clear PCC field of CSR value
|| MVK S2   0x0040,B5 ;set cache enable mask
   OR .L2x  A5,B5,B5  ;set cache enable bit
   MVC.S2   B5,CSR    ;update CSR to enable cache
   NOP4
   NOP
```

2.2.2 キャッシュ･アーキテクチャ

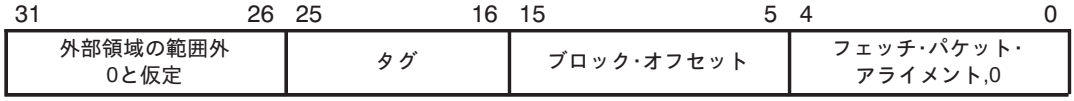
キャッシュのアーキテクチャには、ダイレクト･マップド方式を採用しています。64Kバイトのキャッシュは、2Kのフェッチ･パケット、つまり、2Kのフレームを持っています。キャッシュの幅(フレームのサイズ)は、256ビットです。キャッシュの中のそれぞれのフレームが1つのフェッチ･パケットに対応します。

2.2.2.1 キャッシュにおけるCPUアドレスの使用

図2−2に、CPUからのフェッチ･パケットのアドレスがキャッシュでどのように使用されるかを示します。

- 5ビットのフェッチ･パケット･アラインメント:プログラム･フェッチのリクエストはフェッチ･パケットのバウンダリ(８ワードまたは32バイト)に一致するため、アドレスの下位5ビットは、0として取扱われます。
- 11ビット･タグ･ブロック･オフセット:ダイレクト･マップド方式のため、任意の外部アドレスは、2Kのフレームのいずれかの１つにマップされます。64Kバイトの整数倍分アドレスの違う任意の2つのフェッチ･パケットは、同じフレームにマップされます。CPUアドレスのビット15−5は、フェッチ･パケットが2Kのフレーム中のどのフレームにマップングされるかを決定する11ビットのブロック･オフセットを生成します。
- 10ビット･タグ:キャッシュに関しては、最大64M(0000 0000h−03FF FFFFh)バイトの外部アドレス領域が想定されています。従って、アドレスのビット25からビット16を付け加えることによって、外部メモリー領域にあるフェッチ･パケットの元の位置を決定することができます。キャッシュは、全てのタグを保持するための独立した2K×11タグ･メモリーを持っています。このメモリーの各アドレスには、10ビットのタグに加えてフレームの有効性に関する情報を記録するための有効ビットを持っています。

図2−2. キャッシュ･アドレスのマッピング



2.2.2.2 キャッシュ･フラッシュ

タグ･メモリーの各アドレス毎に配置された専用の有効ビットは、対応するキャッシュ･フレームのデータが有効であるかどうかを示します。キャッシュ･フラッシュ時には、すべての有効ビットがクリアされ、キャッシュのフレームに有効なデータが存在しないことが示されます。キャッシュ･フラッシュは、内部プログラム･メモリーがマップド･モードからキャッシュ･イネーブル･モードに移行する時にのみ実行されます。このようなモードの移行は、CPUコントロール･ステータス･レジスタのPCCフィールドをキャッシュ･イネーブルに設定することで発生させることができます。

2.2.2.3 フレームの置き換え

キャッシュ･ミスは、CPUによってリクエストされたフェッチ･パケットのアドレスのブロック･オフセットに対応するタグが、フェッチ･パケットのアドレスのビット25から16に対応しないとき、あるいは、ブロック･オフセットの位置にある有効ビットがクリアされているときに起こります。キャッシュがイネーブルされている場合には、キャッシュはフェッチ･パケットを対応するフレームにロードし、有効ビットをセットし、タグをリクエストされたアドレスのビット25からビット16にセットし、8つの命令すべてが利用できるようになったのち、フェッチ･パケットをCPUに送ります。

2.3 DMAによるプログラム･メモリーへのアクセス

内部プログラム･メモリーがマップド･モードに設定されている場合には、DMAは、内部プログラム･メモリーに対してリードとライトを行なうことができます。CPUは、対応するDMAチャネルのPRIビットの設定値とは無関係に、内部プログラム･メモリーへのアクセスに関して、DMAよりも優先されます。DMAによるアクセスは、CPUがリクエストを終了するまで延期されます。アービトレーションの後でDMAによるアクセスが行われている間に生成されるCPUからのリクエストが失われることを防止するため、CPUは、DMAによるアクセス毎に1つのウェイト･ステートを挿入します。DMAによる最大のスループットは、1サイクルおきのアクセスとなります。キャッシュ･モードでは、DMAによるライト動作は、プログラム･メモリー･コントローラに無視され、リード動作に対しては未定義のデータが返されます。キャッシュ･モードにおけるDMAのリードとライトの動作について、DMAはリクエストの完了を示す信号を受け取ります。リセット時には、プログラム･メモリー･システムは、マップド･モードに設定され、DMAが内部プログラム･メモリーにコードをブートロードすることが可能となっています。

ブートロードのコードについての詳細は、「第10章 **ブート･モード及びコンフィギュレーション**」をご覧ください。

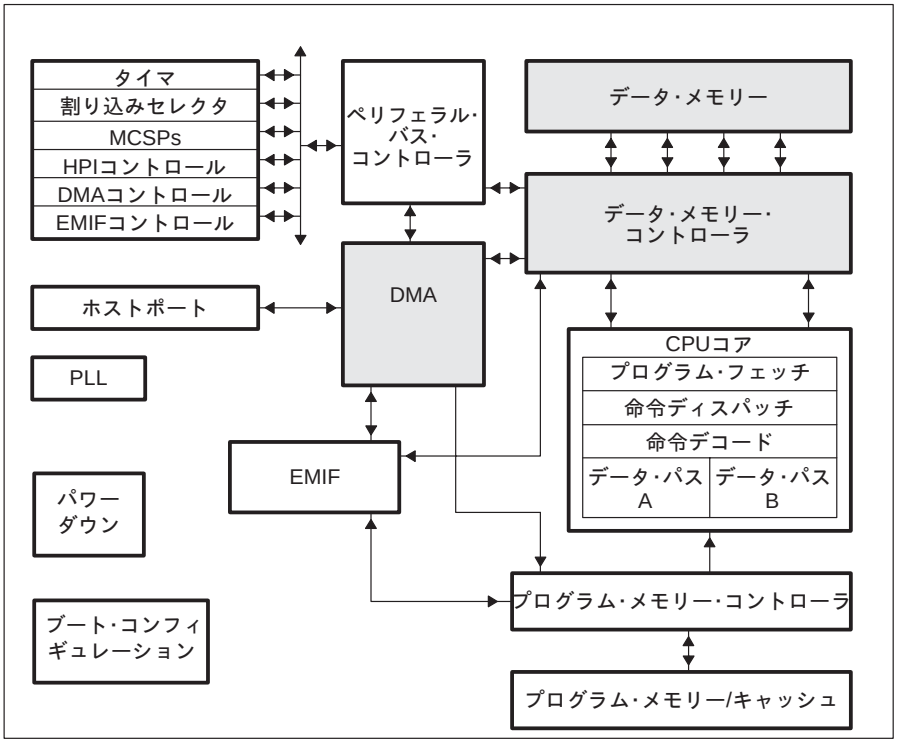
2.4 データ･メモリー･コントローラ

図2−3に示すように、データ･メモリー･コントローラは次のユニットを接続します。

- ❑ CPUとDMAを内部データ･メモリーに接続し、必要に応じてアービトレーションを実行します。
- ❑ CPUを外部メモリー･インターフェイス(EMIF)に接続します。
- ❑ CPUをペリフェラル･バス･コントローラを介してオンチップのペリフェラルに接続します。

ペリフェラル･バス･コントローラは、オンチップのペリフェラルに関し、CPUとDMAの間でのアービトレーションを実行します。

図2−3. TMS320C6000ブロック図



2.5 データ･メモリー･アクセス

データ･メモリー･コントローラは、CPUとDMAからの内部メモリーに対するすべてのデータ･リクエストを処理します。図2−4、図2−5、それに図2−6に、データ･フローの方向と、これらのモジュールの間でのマスタ(リクエスト側)とスレーブ(リソース側)の関係を示します。

- CPUは、次のユニットとの間でデータのリード/ライトを行ないます。
  - 内部データ･メモリー
  - ペリフェラル･バス･コントローラを介したオンチップ･ペリフェラル
  - EMIF
- DMAは、内部データ･メモリーへのリード/ライトをリクエストします。
- CPUは、データ･メモリー･コントローラを介して内部プログラム･メモリーにアクセスできません。

CPUは、2つのアドレス･バス(DA1とDA2)を介してリクエストをデータ･メモリー･コントローラに送ります。ストア･データは、CPUのデータ･ストア･バス(ST1とST2)を介して送信されます。ロード･データは、CPUのデータ･ロード･バス(LD1とLD2)を介して受信されます。CPUのデータ･リクエストは、そのアドレスに基づいて、内部データ･メモリー、ペリフェラル･バス･コントローラを介した内部ペリフェラル領域、または外部メモリー･インターフェイスのいずれかにマッピングされます。また、データ･メモリー･コントローラは、DMAを内部データ･メモリーに接続し、CPU/DMAのアービトレーションを実行します。

2.6 内部データ･メモリーの構成

以下のセクションでは、C6xシリーズDSPのC6201及びC6701のメモリー構成について解説します。

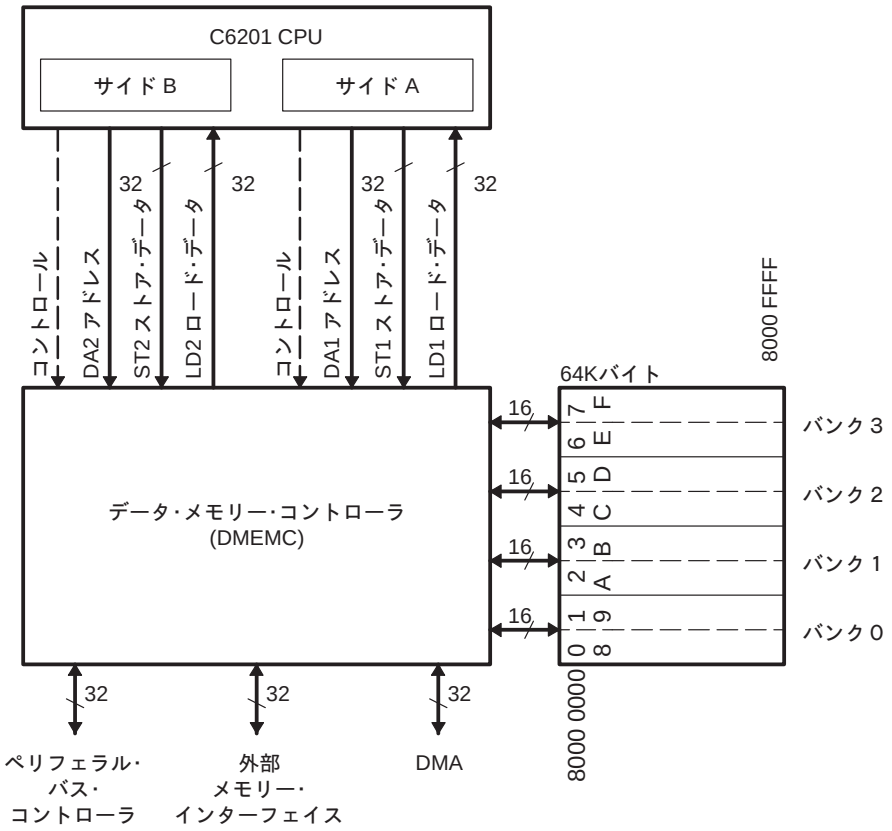
2.6.1 TMS320C6201 リビジョン2

64Kバイトの内部データRAMは、8000 0000hから8000 FFFFhまでのアドレスが割り当てられた64Kバイトの単一ブロックから成っています。このブロックは、16ビット･ハーフ･ワードの8Kのバンク4つから構成されます。CPUとDMAは、異なるバンクに格納されたデータを同時にアクセスすることができます。また、このような構成により、2つのCPUのデータ･ポートA及びBは、リソースのコンフリクトを引き起こすことなく、ブロック内の隣接する16ビットのデータ･エレメントに同時にアクセスすることができます。

表2−2. データ･メモリーの構成(TMS320C6201 リビジョン2)

	バンク0		バンク1		バンク2		バンク3	
先頭アドレス	80000000	80000001	80000002	80000003	80000004	80000005	80000006	80000007
	80000008	80000009	8000000A	8000000B	8000000C	8000000D	8000000E	8000000F
	•	•	•	•	•	•	•	•
	•	•	•	•	•	•	•	•
	•	•	•	•	•	•	•	•
	8000FFF0	8000FFF1	8000FFF2	8000FFF3	8000FFF4	8000FFF5	8000FFF6	8000FFF7
終端アドレス	8000FFF8	8000FFF9	8000FFFA	8000FFFB	8000FFFC	8000FFFD	8000FFFE	8000FFFF

図2−4. データ･メモリー･コントローラとバンクとの接続(TMS320C6201 リビジョン2)



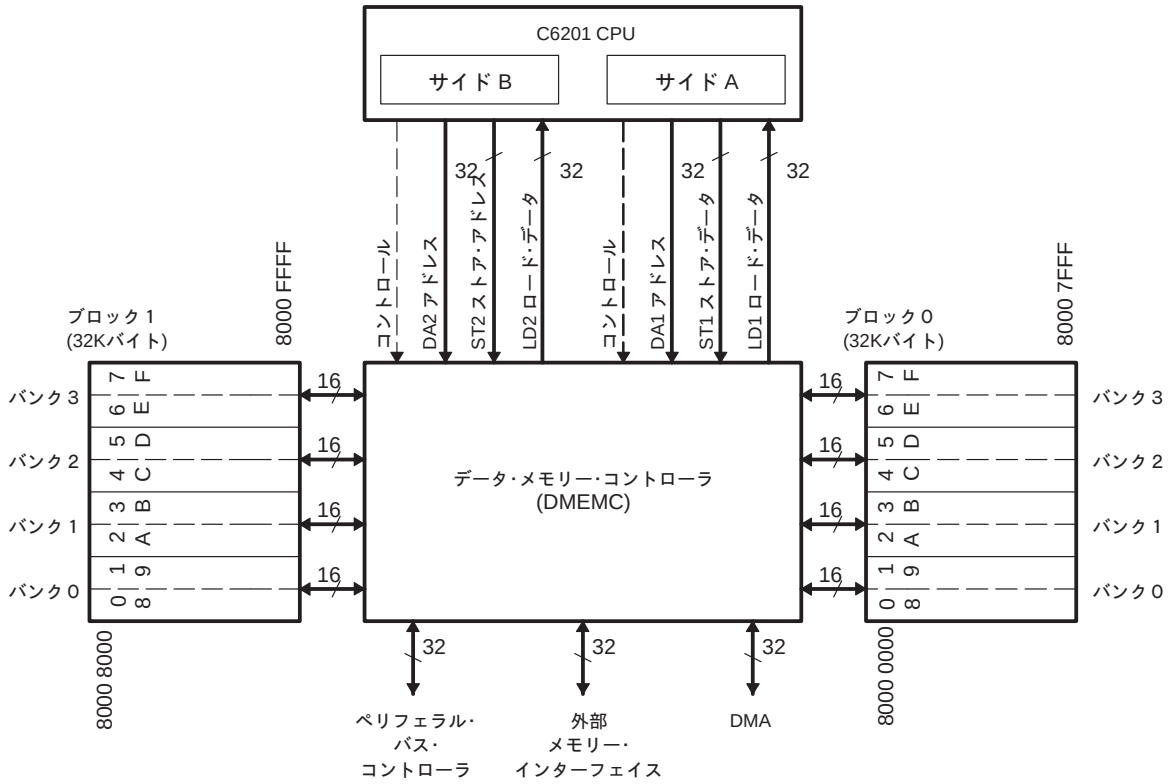
2.6.2 TMS320C6201 リビジョン3

64Kバイトの内部データRAMは、2つのブロックに分かれていて、それぞれが、アドレス8000 00000hから8000 7FFFh及び8000 8000hから8000 FFFFhの2つの32Kバイトの領域に割り当てられています。DMA、それにCPUのサイドAとサイドBは、異なるバンクを使用する限り、コンフリクトを引き起こすことなく、内部メモリーの任意の場所を同時にアクセスすることができます。各ブロックは、いずれも16ビット･ハーフ･ワードの4Kのバンク4つにより構成されています。このため、異なるブロックに対するアクセスが同時に発生する場合、ブロック内でのアドレスを考慮する必要はありません。異なるブロックにアクセスすることによるパフォーマンスの低下もありません。また、CPUとDMAは、同じブロックの中にある異なるバンクのデータを、パフォーマンスの低下を招くことなく、同時にアクセスすることができます。パフォーマンスの低下を防止するためには、同じブロックのデータに対する2つのアクセスが発生する場合に、アドレスの下位ビットに注意を払う必要があります。このような構成は、また、CPUの2つのデータ･ポートAとBが、リソースのコンフリクトを引き起こすことなく、ブロック内の隣接する2つの16ビット･データ･エレメントにアクセスすることを可能としています。

表2−3. データ･メモリーの構成(TMS320C6201 リビジョン3)

	バンク０		バンク１		バンク２		バンク３	
先頭アドレス (ブロック０)	80000000	80000001	80000002	80000003	80000004	80000005	80000006	80000007
	80000008	80000009	8000000A	8000000B	8000000C	8000000D	8000000E	8000000F
	•	•	•	•	•	•	•	•
	•	•	•	•	•	•	•	•
	•	•	•	•	•	•	•	•
	80007FF0	80007FF1	80007FF2	80007FF3	80007FF4	80007FF5	80007FF6	80007FF7
終端アドレス (ブロック０)	80007FF8	80007FF9	80007FFA	80007FFB	80007FFC	80007FFD	80007FFE	80007FFF
先頭アドレス (ブロック１)	80008000	80008001	80008002	80008003	80008004	80008005	80008006	80008007
	80008008	80008009	8000800A	8000800B	8000800C	8000800D	8000800E	8000800F
	•	•	•	•	•	•	•	•
	•	•	•	•	•	•	•	•
	•	•	•	•	•	•	•	•
	8000FFF0	8000FFF1	8000FFF2	8000FFF3	8000FFF4	8000FFF5	8000FFF6	8000FFF7
終端アドレス (ブロック１)	8000FFF8	8000FFF9	8000FFFA	8000FFFB	8000FFFC	8000FFFD	8000FFFE	8000FFFF

図2−5. データ･メモリー･コントローラとバンクとの接続(TMS320C6201 リビジョン3)



2.6.3 TMS320C6701

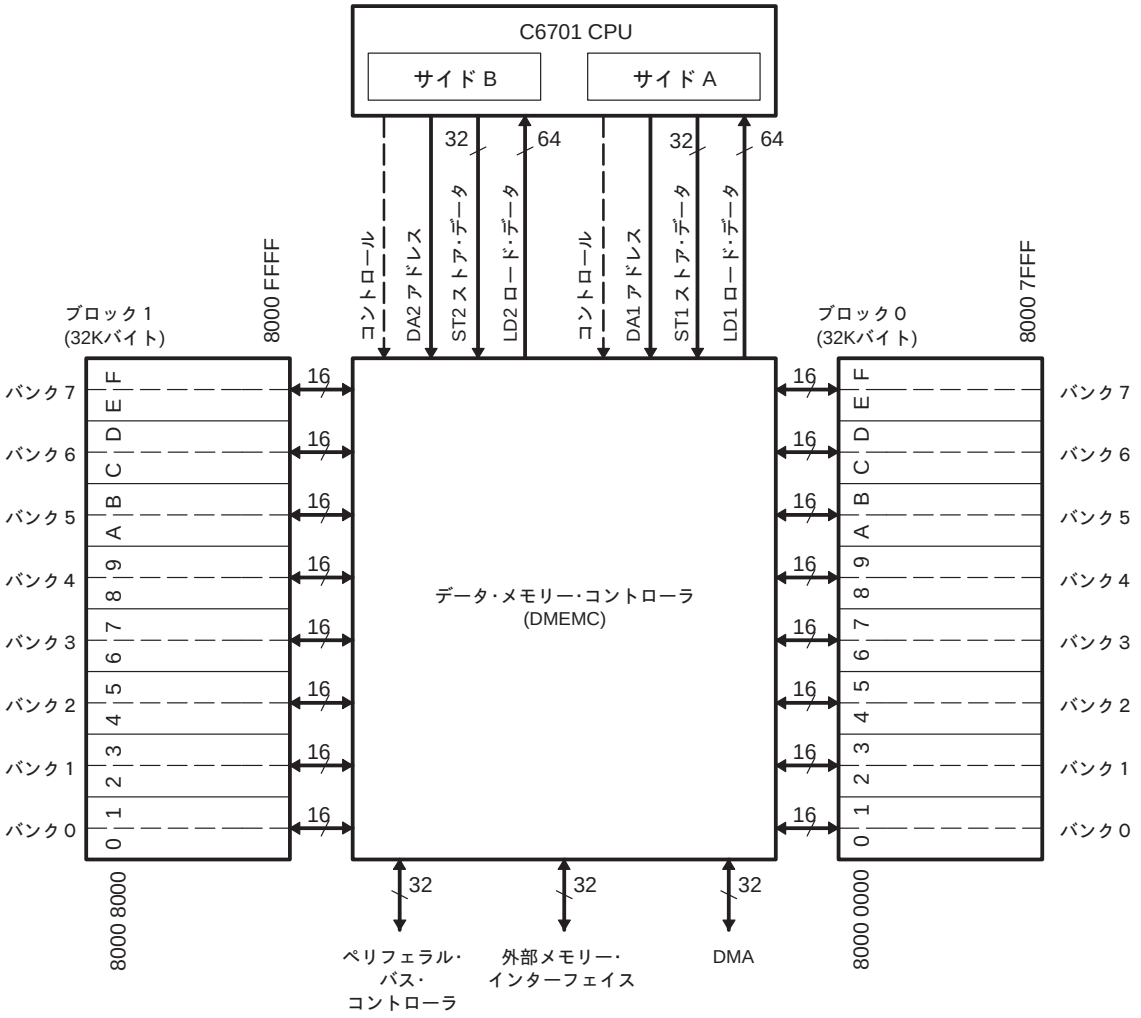
64Kバイトの内部データRAMは、2つのブロックに分かれていて、それぞれが、アドレス8000 00000hから8000 7FFFh及び8000 8000hから8000 FFFFhの2つの32Kバイトの領域に割り当てられています。CPUのサイドAとサイドB、それにDMAは、異なるバンクを使用する限り、コンフリクトを引き起こすことなく、内部データメモリーの任意の場所を同時にアクセスすることができます。このため、異なるブロックに対するアクセスが同時に発生する場合には、ブロック内でのアドレスを考慮する必要はありません。異なるブロックにアクセスすることによるパフォーマンスの低下もありません。同じブロックに対する2つのアクセスが発生する場合にのみ、アドレスに注意すればよいことになります。2つのブロックは、16ビット・ハーフ・ワードの2Kのバンク8つから構成されています。また、CPUとDMAは、同じブロックの中にある異なるバンクのデータを、パフォーマンスの低下を招くことなく、同時にアクセスすることができます。パフォーマンスの低下を防止するためには、同じブロックのデータに対する2つのアクセスが発生する場合に、アドレスの下位ビットに注意を払う必要があります。このような構成は、また、CPUの2つのデータ・ポートAとBが、リソースのコンフリクトを引き起こすことなく、ブロック内の隣接する2つの16ビット・データ・エレメントにアクセスすることを可能としています。

表2−4. データ・メモリーの構成(TMS320C6701)

	バンク0		バンク1		バンク2		バンク3	
先頭アドレス (ブロック0)	80000000	80000001	80000002	80000003	80000004	80000005	80000006	80000007
終端アドレス (ブロック0)	80007FF0	80007FF1	80007FF2	80007FF3	80007FF4	80007FF5	80007FF6	80007FF7
	バンク4		バンク5		バンク6		バンク7	
先頭アドレス (ブロック0)	80000008	80000009	8000000A	8000000B	8000000C	8000000D	8000000E	8000000F
終端アドレス (ブロック0)	80007FF8	80007FF9	80007FFA	80007FFB	80007FFC	80007FFD	80007FFE	80007FFF

	バンク0		バンク1		バンク2		バンク3	
先頭アドレス (ブロック1)	80008000	80008001	80008002	80008003	80008004	80008005	80008006	80008007
終端アドレス (ブロック1)	8000FFF0	8000FFF1	8000FFF2	8000FFF3	8000FFF4	8000FFF5	8000FFF6	8000FFF7
	バンク4		バンク5		バンク6		バンク7	
先頭アドレス (ブロック1)	80008008	80008009	8000800A	8000800B	8000800C	8000800D	8000800E	8000800F
終端アドレス (ブロック1)	8000FFF8	8000FFF9	8000FFFA	8000FFFB	8000FFFC	8000FFFD	8000FFFE	8000FFFF

図2−6. データ･メモリー･コントローラとブロックとの接続(TMS320C6701)



2.6.4 データ･アラインメント

データ･アラインメントについては以下の制約があります。

ダブル･ワード: (C6701のみ)ダブル･ワードは、8バイト(ダブル･ワード)毎のバウンダリに配置され、常にバイト･アドレスの下位3ビットが0となる位置から開始します。ダブル･ワードは、LDDW命令によるロード動作によってのみ使用されます。ストア動作では、ダブル･ワードを使用しません。

ワード: ワードは、4バイト(ワード)毎のバウンダリに配置され、常にバイト･アドレスの下位2ビットが0である位置から開始します。ワード単位でのアクセスには、2つの隣接する16ビット幅のバンクを必要とします。

ハーフ･ワード: ハーフ･ワードは、2バイト(ハーフ･ワード)毎のバウンダリに配置され、常にバイト･アドレスの最下位ビットが0である位置から開始します。ハーフ･ワード単位でのアクセスには、1つの16ビット幅のバンクを必要とします。

バイト: バイトによるアクセスには、アラインメントについての制約はありません。

2.6.5 CPUによる2つの内部メモリーへのアクセス

CPUとDMAは、8ビットのバイト、16ビットのハーフ･ワード、それに32ビットのワードのリード/ライトを行なうことができます。データ･メモリー･コントローラは、16ビット･バンクのそれぞれについて、個別にアービトレーションを実行します。アービトレーションは、16ビット幅のバンクに対して行われますが、各バンクは、バイト幅でのアクセスをサポートするためのバイト･イネーブルを持っています。しかしながら、バイト単位のアクセスの場合でも、そのバイトを含む16ビットの全体を他のアクセス動作により同時にアクセスすることはできなくなります。

複数のリクエスト側が異なるバンクにあるデータにアクセスする限り、パフォーマンスの低下を招くことなく、すべてのアクセスを同時に行なうことができます。また、別々の32Kバイトのメモリー･ブロックに対して2つのメモリー･アクセスが行われる場合には、アドレスにかかわらず、メモリーのコンフリクトは発生しません。同じブロックの中で複数のデータ･アクセスが行われる場合には、これらが異なるバンクに対して行われる限り、同時に複数のメモリー･アクセスが許されるようなメモリー構成となっています。1つのCPUサイクルの中では、内部メモリー中の2つの異なるバンクへの2つのアクセスを、ウェイト･ステートの挿入なしに実行することができます。内部メモリー中の同じバンクに対する2つのアクセスが同時に発生すると、CPUの1クロックにわたり、CPUのパイプライン全体がストール状態となり、2CPUクロックの間に2つのアクセスが実行されます。これらの制約は、アクセス動作がロードであるかストアであるかに関係なく適用されます。

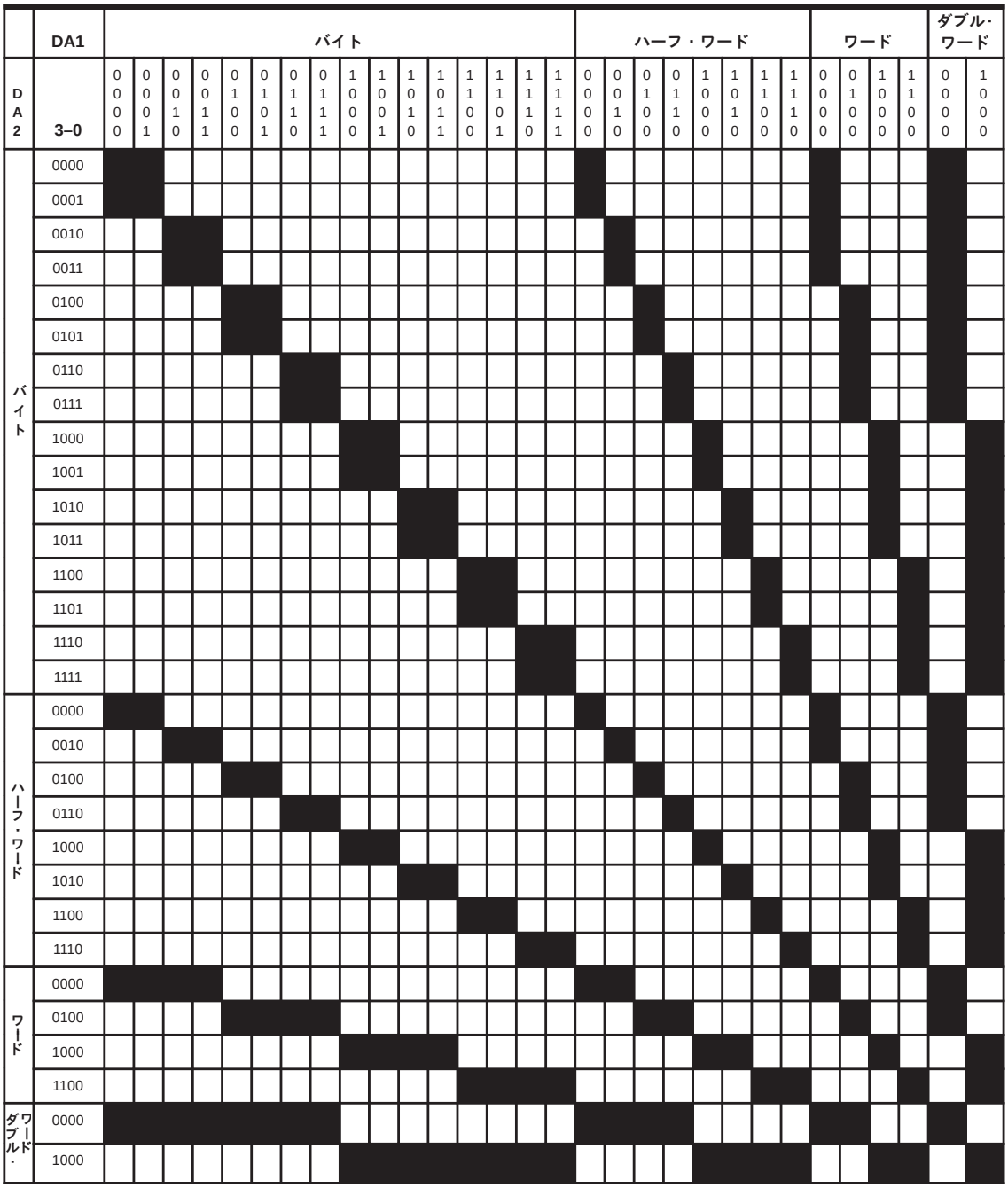
同じ実行パケットの中のロードとストアは、同じCPUサイクルの中で、データ･メモリー･コントローラに渡されます。前後のCPUサイクルにあるロードとストアは、現在のCPUサイクルにおける、内部メモリー･アクセスに対してウェイト･ステートを生成しません。このように、内部データ･メモリー･アクセスにおいてウェイト･ステートが発生するのは、同じ16ビット幅のバンクをアクセスする命令が同じフェッチ･パケットの中にあり、それらの命令の間でコンフリクトが発生した場合に限られます。この状態を、内部メモリー･コンフリクトと呼びます。このとき、データ･メモリー･コントローラは、CPUを1CPUクロック分ストールさせ、アクセス動作をシリアルにし、それぞれのアクセスを別々に行わせます。2つのアクセスの優先順位については、ロード･アクセスがストア･アクセスに先立って行われます。ストアと並列に実行されるロードは、常に、ストア命令に対して優先されます。ロードとストアが同じリソース(例えば、EMIF、ペリフェラル･バス、内部メモリー･ブロック)をアクセスする場合には、ロードは、常に、ストアの前に実行されます。2つのアクセスがどちらもストアである場合には、DA1からのアクセスがDA2からのアクセスに優先されます。2つのアクセスがどちらもロードである場合には、DA2からのアクセスがDA1からのアクセスに優先されます。図2−7に、CPUが2つのデータ･アクセス(DA1とDA2)を実行した場合に、内部メモリー･コンフリクトが発生する条件を示します。

図2−7. 同じブロックへのアクセスによる内部メモリー･コンフリクト(TMS320C6201 リビジョン2及び3)

	DA1	バイト								ハーフ・ワード				ワード	
DA2	2:0	000	001	010	011	100	101	110	111	000	010	100	110	000	100
バイト	000	■	■							■				■	
	001	■	■							■				■	
	010			■	■						■			■	
	011			■	■						■			■	
	100					■	■					■			■
	101					■	■					■			■
	110							■	■				■		■
	111							■	■				■		■
ハーフ・ワード	000	■	■							■				■	
	010			■	■						■			■	
	100					■	■					■			■
	110							■	■				■		■
ワード	000	■	■	■	■					■	■			■	
	100					■	■	■	■			■	■		■

注： 黒く表示した部分において競合が発生します。

図2−8. 同じブロックへのアクセスによる内部メモリー･コンフリクト(TMS320C6701)



注： 黒く表示した部分において競合が発生します。

2.6.6 DMAによる内部メモリーへのアクセス

CPUが一方のブロックの任意の部分をアクセスしている間、DMAは、もう一方のブロックの任意の部分をアクセスすることができます。CPUとDMAが同じブロックをアクセスしていて、アクセス先が同じ16ビット・バンクのときには、CPU/DMAの優先順位の設定に従って、DMAの動作が先か、CPUの動作が先かが決定されます。DMAとCPUのコンフリクトについては、図2−7及び図2−8を参照してください。図中で、一方の軸はDMAのアクセスに対応し、もう一方の軸はCPUの1つのデータ・ポートに対応するとして解析を行ない、次に、CPUのもう1つのデータ・ポートについても同じ解析を行ないます。これらの2つの参照の結果、コンフリクトが生じないのであれば、内部メモリーに関してCPUとDMAのコンフリクトは発生しません。もし、いずれかの参照においてコンフリクトが生ずる場合には、内部メモリーに関してCPUとDMAの間にコンフリクトが発生することになります。この場合には、第5章のダイレクト・メモリー・アクセス(DMA)コントローラで説明するとおり、優先順位の問題はDMAチャネルのPRIビットにより解決されることになります。DMAチャネルの優先順位がCPUより高く設定されている場合(PRI=1)には、DMAによるアクセスが完了するまでCPUによるアクセスは禁止され、CPUは、1クロックサイクルのウェイト・ステートを挿入します。2つのCPUのポートとDMAが同じメモリーブロックをアクセスする場合には、ウェイト・ステートの数は2に増えます。DMAが、CPUがアクセスしようとするブロックに対して、複数の連続したリクエストを持っている場合には、DMAがブロックへのアクセスを完了するまで、CPUのアクセスは禁止されます。これに対し、CPUがより高い優先順位(PRI=0)に設定されている場合には、CPUのデータ・ポートの双方がそのバンクに対するアクセスを終了するまで、DMAのアクセスは禁止されます。このような設定においては、DMAのアクセス・リクエストがウェイト・ステートを発生することはありません。

2.6.7 データ・エンディアン

バイト単位のアドレッシングが可能なマイクロプロセッサに関しては、データの配列の方法として、次の2つの標準があります。

- ❑ リトル・エンディアン、バイトは右から左に並べられ、、最上位のバイトはアドレスの大きい方に置かれます。
- ❑ ビッグ・エンディアン、バイトは左から右に並べられ、、最上位のバイトは小さいアドレスに置かれます。

CPUとDMAでは、どちらのエンディアンもサポートしています。このエンディアンは、デバイスのLENDIANのピンで選択することができます。このピン入力は、デバイスがリセットされたときにだけラッチされます。それゆえ、LENDIANピンはデバイスリセット後は定常となります。つまり、オペレーションの途中で変更できません。LENDIAN＝1とLENDIAN=0は、それぞれ、リトル・エンディアンとビッグ・エンディアンを選択します。この選択は、CPUとDMAの双方に対して有効となります。表2−5は、ビッグ・エンディアンまたはリトル・エンディアンからCPUへのすべてのデータ・ロードについて、メモリー上のデータ・ワードのどのビットが、ロードの実行対象となるレジスタのどのビットにロードされるかを示したものです。ここで、メモリー上のデータは、最初の列のLDW命令を実行した後のレジスタ中のデータと同じものと仮定しています。表2−6と表2−7は、ビッグ・エンディアンま

たはリトル・エンディアンからCPUへのすべてのデータ・ストアについて、レジスタのどのビットが、ストアの実行対象となるメモリー・ワードのどのビットにストアされるかを示したものです。ここで、ソース・レジスタにあるデータは、最初の列のSTW命令を実行した後のメモリーにあるデータと同じものと仮定しています。

表2−5. リトル・エンディアンまたはビッグ・エンディアンのデータ・ロード後のレジスタの内容 (TMS320C6201/TMS320C6701)

命令	アドレス・ビット (1:0)	ビッグ・エンディアン レジスタ結果	リトル・エンディアン レジスタ結果
LDW	00	BA987654h	BA987654h
LDH	00	FFFFBA98h	00007654h
LDHU	00	0000BA98h	00007654h
LDH	10	00007654h	FFFFBA98h
LDHU	10	00007654h	0000BA98h
LDB	00	FFFFFFBAh	00000054h
LDBU	00	000000BAh	00000054h
LDB	01	FFFFFF98h	00000076h
LDBU	01	00000098h	00000076h
LDB	10	00000076h	FFFFFF98h
LDBU	10	00000076h	00000098h
LDB	11	00000054h	FFFFFFBAh
LDBU	11	00000054h	000000BAh

注： データ・メモリー上のアドレス”xxxx xx00”にあるワードの内容はBA987654hです。

表2−6. リトル･エンディアンまたはビッグ･エンディアンのデータ･ロード後のレジスタの内容(C6701のみ)

命令	アドレス･ビット (2:0)	ビッグ･エンディアン レジスタ結果	リトル･エンディアン レジスタ結果
LDDW (C6701のみ)	000	FEDC BA98 7654 3210h	FEDC BA98 7654 3210h
LDW	000	FEDC BA98h	7654 3210h
LDW	100	7654 3210h	FEDC BA98h

注： データ･メモリー上のアドレス”xxxx x000”にあるダブル･ワードの内容はFEDC BA98 7654 3210hです。

表2−7. リトル･エンディアンまたはビッグ･エンディアンのデータ･ストア後のメモリーの内容 (TMS320C6201/C6701)

命令	アドレス･ビット (1:0)	ビッグ･エンディアン メモリー結果	リトル･エンディアン メモリー結果
STW	00	BA98 7654h	BA98 7654h
STH	00	7654 1970h	0112 7654h
STH	10	0112 7654h	7654 1970h
STB	00	5412 1970h	0112 1954h
STB	01	0154 1970h	0112 5470h
STB	10	0112 5470h	0154 1970h
STB	11	0112 1954h	5412 1970h

注： ST命令実行前における、データ･メモリー上のアドレス”xxxx xx00”にあるワードの内容は01121970hです。ソース･レジスタの内容はBA987654hです。

2.7 ペリフェラル･バス

ペリフェラルは、コントロール･レジスタを介したアクセスにより、CPUとDMAによって制御されます。CPUとDMAは、ペリフェラル･データ･バスを介してこれらのレジスタにアクセスします。DMAは、直接にペリフェラル･バス･コントローラにアクセスしますが、CPUは、データ･メモリー･コントローラを介してこれにアクセスします。

2.7.1 バイト及びハーフ･ワード･アクセス

ペリフェラル･バス･コントローラは、ペリフェラル･バスのすべてのアクセスをワード単位のアクセスに変換します。しかし、リード･アクセスでは、CPUとDMAの双方がワード中から正しいビット位置をリードして、バイトやハーフ･ワードのアクセスを行なうことができます。一方、バイトやハーフ･ワードでのライトについては、CPUもDMAも、イネーブルされたバイトに対してのみ正しい値をライトします。表2−8に、正しい値が保証されるものを示します。このようにして、イネーブルされていないバイトに対しては、未定義の結果がライトされることになります。ユーザがイネーブルされていないバイトにライトされる値を気にしないのであれば、問題は生じません。もし、この結果を受容できないのであれば、ペリフェラル･レジスタに対しては、ワード単位でアクセスしなければなりません。

表2−8. リトル･エンディアンまたはビッグ･エンディアンのデータ･ストア後のメモリーの内容

アクセス･タイプ	アドレス･ビット (1:0)	ビッグ･エンディアン メモリー結果	リトル･エンディアン メモリー結果
ワード	00	XXXXXXXX	XXXXXXXX
ハーフ･ワード	00	XXXX????	????XXXX
ハーフ･ワード	10	????XXXX	XXXX????
バイト	00	XX??????	??????XX
バイト	01	??XX????	????XX??
バイト	10	????XX??	??XX????
バイト	11	??????XX	XX??????

注： Xは正しくライトされる位置を示し、?は、ライトの後の値が未定義となる位置を示します。

2.7.2 CPUウェイト･ステート

CPUからペリフェラル･バス･コントローラへの連続ではない 1 回のアクセスにより 6 つのCPUウェイト･ステートが発生します。これらのウェイト･ステートを挿入することにより、パイプライン･レジスタは、信号がチップ上のCPUとペリフェラルの間の経路を伝わる時間を確保し、アービトレーションのための時間を確保することができます。

2.7.3 CPUとDMAの間のアービトレーション

図2−5と図2−6に示すように、ペリフェラル･バス･コントローラは、ペリフェラル･バスについてのCPUとDMAの間のアービトレーションを実行します。内部データ･アクセスの場合と同様に、DMAの中のPRIビットにより、CPUとDMAの優先順位が決定されます。CPUの間でのコンフリクト(データ･メモリーを介するもの)が発生した場合には、リクエスト側のうちより優先順位の低いものは、より高い優先順位を持つものがペリフェラル･バス･コントローラに対するアクセスのすべてを終了するまで、アクセスを禁止されます。ペリフェラル･バスは、単一のリソースとしてアービトレーションの対象とされるため、より低い優先順位を持ったリソースは、リクエスト側のより優先順位の高いものによってアクセスされるペリフェラルだけではなく、すべてのペリフェラルに対するアクセスから排除されます。

第3章

TMS320C6202/C6203プログラム及びデータ・メモリー

3

この章では、TMS320C6202/C6203のプログラム・メモリー・コントローラ及びデータ・メモリー・コントローラについて解説します。キャッシュ動作とブートロード動作などのプログラム・メモリーのモードについてもこの章で述べます。

Topic **Page**

3.1 TMS320C6202/C6203プログラム･メモリー･コントローラ

TMS320C6202/C6203のプログラム･メモリー･コントローラ(PMEMC)はTMS320C6201リビジョン3と同じ機能が利用可能です。PMEMCは、128Kバイトのメモリーとして、もしくはダイレクト･マップド･キャッシュとして動作するブロックを持っています。メモリーまたはキャッシュとしての領域に加えて、マップド･メモリーとして動作するブロックを、C6202では128Kバイト、C6203では256Kバイト持っています。この2つのブロックには、それぞれ独立にアクセスすることができるので、CPUによって一方のブロックからプログラム･フェッチをしているのと同時に、プログラム･メモリーの他方のブロックに干渉なくDMA転送することが可能です。表3−1、表3−2は、現在のTMS320C6000シリーズで利用可能な、内部メモリーとキャッシュの設定の比較です。図3−1に、C6202CPU、PMEMC、それにメモリー･ブロックの接続を、ブロック図で示します。表3−1で示されたアドレスはメモリー･マップ･モード1の動作におけるものです。

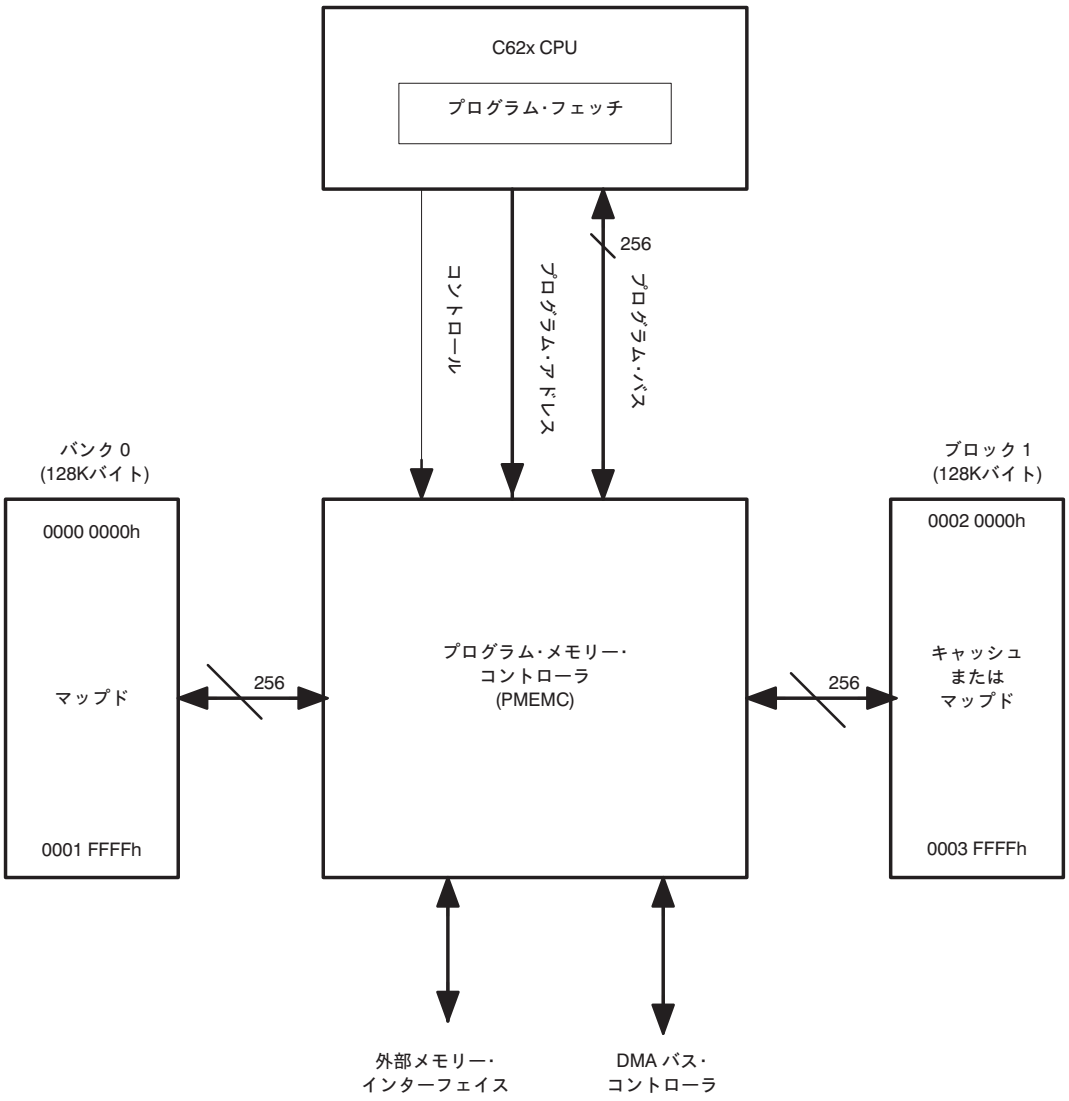
表3−1. TMS320C6201/C6701/C6202/C6203内部メモリー･コンフィギュレーション

デバイス	CPU	内部メモリー･アーキテクチャ	全メモリー(バイト)	プログラム･メモリー(バイト)	データ･メモリー(バイト)
C6201	6200	ハーバード	128K	64K(マップド/キャッシュ)	64K(マップド)
C6701	6700	ハーバード	128K	64K(マップド/キャッシュ)	64K(マップド)
C6202	6200	ハーバード	384K	128K(マップド) 128K(マップド/キャッシュ)	128K(マップド)
C6203	6200	ハーバード	896K	256K(マップド) 128K(マップド/キャッシュ)	512K(マップド)

表3−2. TMS320C6201/C6701/C6202/C6203キャッシュ･アーキテクチャ

キャッシュ空間	サイズ(バイト)	方式	ライン･サイズ(バイト)
C6201 プログラム	64K	ダイレクト･マップド	32
C6701 プログラム	64K	ダイレクト･マップド	32
C6202 プログラム	128K	ダイレクト･マップド	32
C6203 プログラム	128K	ダイレクト･マップド	32

図3−1. TMS320C6202プログラム･メモリー･コントローラ･ブロック図



3.2 マップド･メモリー動作

CPUコントロール･ステータス･レジスタのPCCフィールドで、マップド･モードに設定されている場合には、内部プログラムRAMの全てのブロックが内部プログラム空間にマッピングされます。表3−3、表3−4に、リセット時に選択されたマップ･モードにおける、TMS320C6202/C6203内部RAMのブロックのアドレス空間をそれぞれ示します。

表3−3. マップド･モードにおけるTMS320C6202内部プログラムRAMのアドレス･マッピング

	マップ0	マップ1
ブロック0	0140 0000h–0141 FFFFh	0000 0000h–0001 FFFFh
ブロック1	0142 0000h–0143 FFFFh	0002 0000h–0003 FFFFh

表3−4. マップド･モードにおけるTMS320C6203内部プログラムRAMのアドレス･マッピング

	マップ0	マップ1
ブロック0	0140 0000h–0143 FFFFh	0000 0000h–0003 FFFFh
ブロック1	0144 0000h–0145 FFFFh	0004 0000h–0005 FFFFh

マップド･モードでは、CPUとDMAはRAMの全てのブロックにアクセスすることができます。内部RAMがマッピングされているアドレス空間以外へのアクセスは、すべてEMIFに向けられます。DMAは一度にひとつのブロックにのみアクセスすることができます。CPUとDMAは異なるブロックにアクセスする限り、競合なく内部RAMにアクセス可能です。もし、CPUとDMAが同時に同じブロックにアクセスしようとすると、CPUのそのブロックに対するアクセスが終了するまで、DMAはストールします。CPUのアクセスが終了した後、DMAはそのブロックにアクセスできるようになります。DMAは、単一の転送でブロック間にまたがって転送することができません。ブロック境界をまたがってDMA転送を実行するためには、別にDMA転送を行なう必要があります。

3.3 キャッシュ動作

CPUコントロール･ステータス･レジスタのPCCフィールドで、キャッシュ･モードのいずれかに設定されている場合には、ブロック1はキャッシュとして働き、ブロック0は内部プログラム空間にマッピングされます。表3−5、表3−6に、各マップ･モードにおける、TMS320C6202/C6203で、キャッシュとして使われないRAMのアドレス空間をそれぞれ示します。

表3−5. キャッシュ･モードにおけるTMS320C6202内部プログラムRAMのアドレス･マッピング

	マップ0	マップ1
ブロック0	0140 0000h–0141 FFFFh	0000 0000h–0001 FFFFh

表3−6. キャッシュ･モードにおけるTMS320C6203内部プログラムRAMのアドレス･マッピング

	マップ0	マップ1
ブロック0	0140 0000h–0143 FFFFh	0000 0000h–0003 FFFFh

C6202/C6203でのキャッシュは、C6201のキャッシュと同じように動作します。キャッシュとして使用されているメモリー領域に、CPUやDMAがアクセスした場合には、未定義の結果が返されます。マップド･モード時と同様に、CPUとDMAがブロック0に同時にアクセスした場合には、CPUのアクセスが完了するまでDMAはストール状態になります。キャッシュがフラッシュされている間、DMAはブロック0にストールすることなくアクセスし続けることができます。一方、キャッシュ･フラッシュの間、CPUは停止します。キャッシュを有効にする際には、DMAのブロック1へのアクセスがすべて終了している必要があります。

注：

PMEMCの動作モードを変更するときは、PMEMCの正常動作を保証するために以下のようなアセンプリ･ルーチンを使用してください。このルーチンはキャッシュを有効にします。キャッシュ･イネーブル以外の状態にPMEMCの動作モードを変更するには、下のルーチンの4行目を修正してください。ここでは、設定したいPCCの値をB5に入れています。例えば、キャッシュをマップド･モードにするにはB5に0000hを入れることになります。この例で使用されているCPUレジスタは重要な意味を持っていません。A0–A15またはB0–B15のどのレジスタでも使うことができます。

```
   .allgn    32
   MVC .S2   CSR,B5    ;copy control status register
|| MVK .S1   0xff1f,A5
   AND .L1x  A5,B5,A5  ;clear PCC field of CSR value
|| MVK  S2   0x0040,B5 ;set cache enable mask
   OR  .L2x  A5,B5,B5  ;set cache enable bit
   MVC .S2   B5,CSR    ;update CSR to enable cache
   NOP 4
   NOP
```

3.4 ブートロード動作

C6202/C6203のブートロードはC6201リビジョン3と同様に動作します。ROMブートロードでは、64Kバイトのデータが CE1領域の先頭アドレスからメモリーの0番地に転送されます。HPIブートロードでは、ホストは、内部プログラム領域を含めて、全てのメモリー空間にリード/ライトが可能です。

3.5 TMS320C6202/C6203データ･メモリー･コントローラ

TMS320C6202/C6203のデータ･メモリー･コントローラ(DMEMC)では、TMS320C6201リビジョン3と同じ機能が利用可能です。C6202/C6203のDMEMCは、それぞれ128Kバイト及び512KバイトのRAMを持ちます。このRAMは4つバンクからなる2つのブロックにより構成され、それぞれのバンクは16ビット幅です。C6202/C6203のDMEMCはC6201のDMEMCと同様に動作し、DMAやCPUのサイドA、サイドBが異なるバンクに同時にアクセスする際には、コンフリクトはありません。図3−2に、C6202　CPU、DMEMC、それにメモリー･ブロックの接続をブロック図で示します。また、表3−7、表3−8には、内部データRAMの各ブロックのアドレス空間を示します。

図3−2. TMS320C6202データ･メモリー･コントローラ･ブロック図

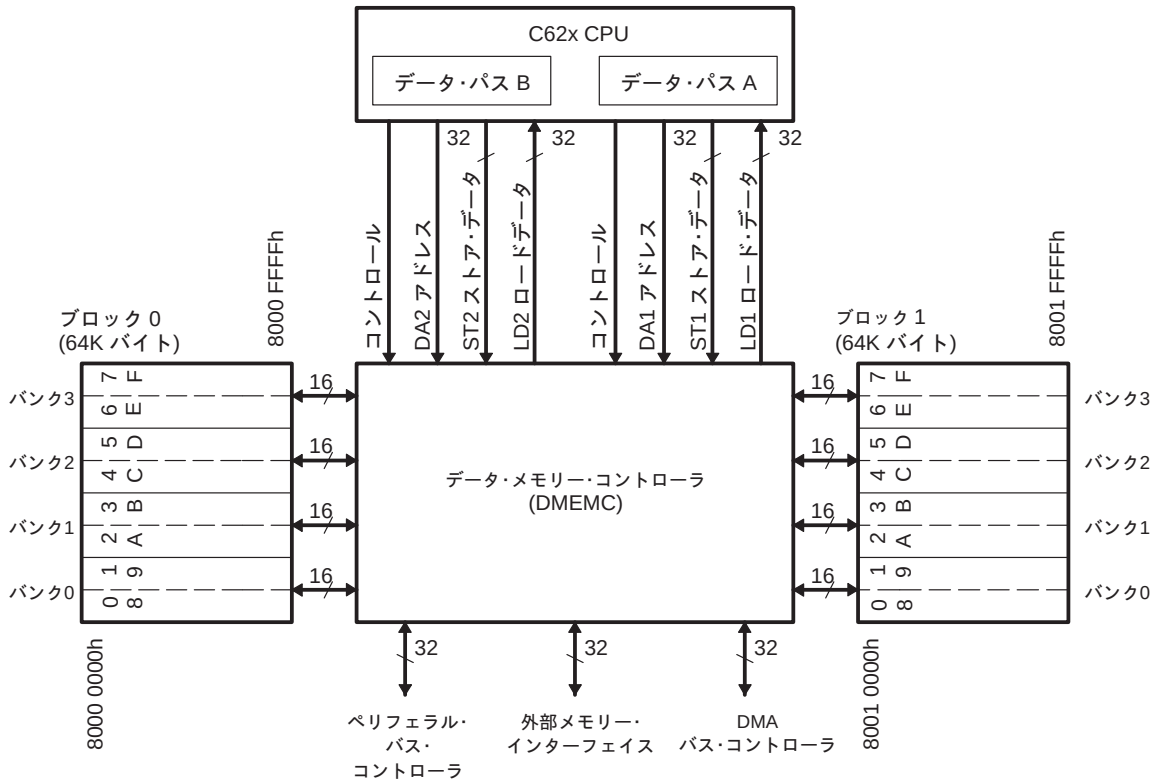


表3−7. TMS320C6202内部データRAM アドレス・マッピング

ブロック0	8000 0000h–8000 FFFFh
ブロック1	8001 0000h–8001 FFFFh

表3−8. TMS320C6203内部データRAM アドレス・マッピング

ブロック0	8000 0000h—8003 FFFFh
ブロック1	8004 0000h—8007 FFFFh

第5章

DMA(ダイレクト･メモリー･アクセス)

この章では、TMS320C6201/C6202/C6203/C6701デバイスにおいて利用できるDMA(Direct Memory Access)のチャネルとレジスタについて解説します。

5

Topic **Page**

5.1 概要

ダイレクト･メモリー･アクセス(DMA)は、メモリー･マップ上の領域間におけるデータの転送をCPUとは独立に実行するものです。DMAにより、内部メモリー、内部ペリフェラル、外部デバイスの間でのデータのやりとりを、CPUの動作のバックグランドで行なうことができます。DMAは、4つの独立したプログラマブルなチャネルを持っていて、4つの異なるデータ転送を行なうことができます。また、5番目(補助用)のチャネルはホストポート･インターフェイス(HPI)用の補助チャネルで、HPIからのリクエストに従いデータ転送を行ないます。DMAの動作を理解する上で、以下のような用語が重要となります。

- リード転送: DMAが、メモリー上のソース位置からデータ･エレメントをリードすることです。
- ライト転送: DMAが、リード転送の間にリードしたデータ･エレメントをメモリー上のデスティネーションにライトすることです。
- エレメント転送: 単一のデータ･エレメントに対するリード転送とライト転送の組み合わせです。
- フレーム転送: DMAの各チャネルは、フレームあたり、それぞれ独立にエレメント数を設定することができます。DMAが、1つのフレーム中のすべてのエレメントを移動すると、フレーム転送が完了します。
- ブロック転送: DMAの各チャネルは、また、ブロックあたり、それぞれに独立に、そのフレーム数をプログラムによって設定可能です。DMAが、転送するようにプログラムされていたすべてのフレームを移動すると、ブロック転送が完了します。
- エレメント転送の送信:スプリット･モードでは、データ･エレメントはソース･アドレスからリードされ、スプリット･デスティネーション･アドレスにライトされます。詳細は5.8節をご覧ください。
- エレメント転送の受信:スプリット･モードでは、データ･エレメントはスプリット･ソース･アドレスからリードされ、デスティネーション･アドレスにライトされます。詳細は5.8節をご覧ください。

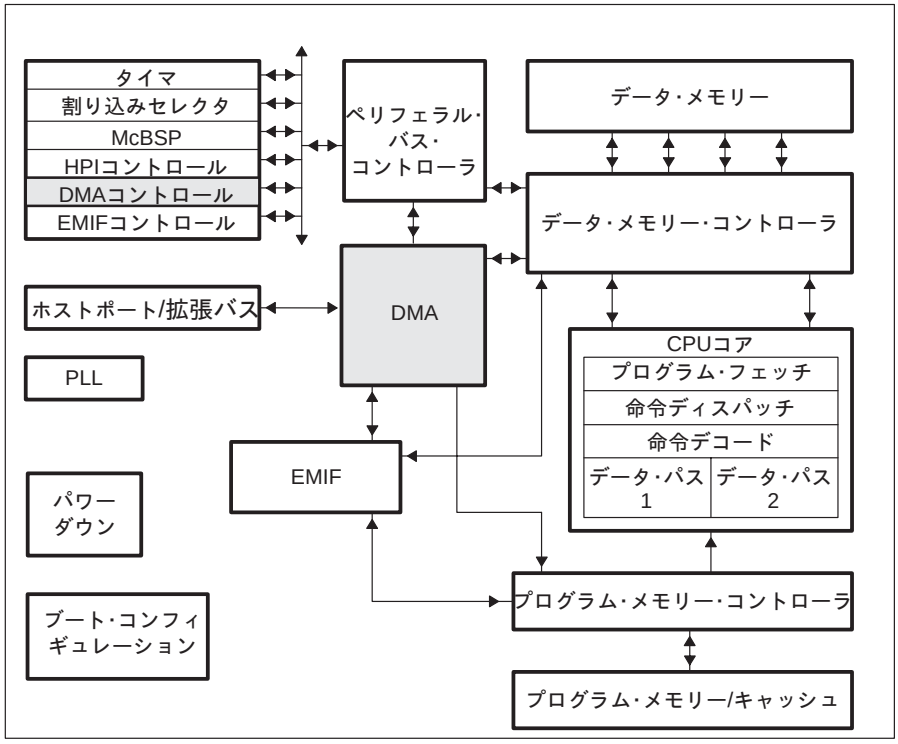
DMAは、以下のような特徴を持っています。

- バックグラウンド動作: DMAは、CPUと独立に動作します。
- 高スループット: エレメントは、CPUのクロック･レートで転送することができます。詳細は、「5.11 DMAの制御構造」をご覧ください。
- 4つのチャネル: 4つの独立したブロック転送を行なうことができます。「5.2 DMAレジスタ」をご覧ください。

- 補助チャンネル: このチャネルによって、ホストポートはCPUのメモリー空間にアクセスすることができます。補助チャネルからのリクエストは、他のチャネルやCPUからのリクエストより優先されます。
- スプリット･チャネル動作: 1つのチャネルでペリフェラルに対するリード/ライト転送を同時に行なうことができます。1つのチャネルで2つのチャネルを使ったのと同じ効果が得られます。5.8節をご覧ください。
- マルチフレーム転送: 各ブロックは複数のフレームから構成することができます。フレーム数はプログラムで設定可能です。「5.5 **転送カウント**」をご覧ください。
- 優先順位が設定可能: 各チャネルのCPUに対する優先度は各チャネル毎に設定できます。
- プログラマブルなアドレス生成: 各チャネルのソースおよびデスティネーション･アドレス･レジスタは、リード/ライトそれぞれの転送に対してさまざまな設定をすることができます。アドレスは、一定に保持、インクリメント/デクリメント、またはインデックスを用いて更新することができます。インデックスは、前回の転送と次の転送で設定を変更することが可能です。詳細は、5.7.1節を参照して下さい。
- 32ビット･アドレス空間: DMAはメモリー･マップ上のすべての空間にアクセスすることができます。
  - 内部データ･メモリー
  - 内部プログラム･メモリー(マップド･モードのときのみ)
  - オンチップ･ペリフェラル
  - EMIFを介した外部メモリー
  - 拡張バスを介した拡張メモリー
- データ幅を選択可能: 各チャネルは独立に転送のデータ幅をバイト、16ビット･ハーフ･ワード、または32ビット･ワードの中から選択することができます。5.7.3節を参照して下さい。
- 自動初期化: ブロック転送完了後に、次のブロック転送のための初期化を自動で行なうことができます。5.4.1節を参照して下さい。
- イベント同期: 各リード、ライト、またはフレーム転送を、選択されたイベントによって開始させることができます。5.6節を参照して下さい。
- 割り込み生成: フレーム転送完了、ブロック転送完了、エラー発生時などに、各DMAチャネルはCPUに対して割り込みをかけることができます。5.10節を参照して下さい。

図5−1に、C6000のブロック図を示します。網掛けの部分が、DMAに関連した部分です。

図5−1. DMAとTMS320C6201/C6202/C6203/C6701のメモリー･マップされたモジュールとの接続



5.2 DMAレジスタ

DMAレジスタにより、DMAの動作を設定します。表5−1と表5−2に、DMAコントロール･レジスタのメモリー空間上でのアドレスを示します。これらのレジスタには、DMAグローバル･データ、カウント･リロード、インデックス、アドレス･レジスタがあり、また、各チャネル毎に独立したコントロール･レジスタがあります。

表5−1. DMAコントロール･レジスタ(アドレス順)

16進によるバイト･アドレス	名前	参照のセクション
0184 0000	DMAチャネル０プライマリ･コントロール	5.2.1
0184 0004	DMAチャネル２プライマリ･コントロール	5.2.1
0184 0008	DMAチャネル０セカンダリ･コントロール	5.10
0184 000C	DMAチャネル２セカンダリ･コントロール	5.10
0184 0010	DMAチャネル０ソース･アドレス	5.7
0184 0014	DMAチャネル２ソース･アドレス	5.7
0184 0018	DMAチャネル０デスティネーション･アドレス	5.7
0184 001C	DMAチャネル２デスティネーション･アドレス	5.7
0184 0020	DMAチャネル０転送カウンタ	5.5
0184 0024	DMAチャネル２転送カウンタ	5.5
0184 0028	DMAグローバル･カウント･リロード･レジスタA	5.5
0184 002C	DMAグローバル･カウント･リロード･レジスタB	5.5
0184 0030	DMAグローバル･インデックス･レジスタ A	5.7.2
0184 0034	DMAグローバル･インデックス･レジスタ B	5.7.2
0184 0038	DMAグローバル･アドレス･レジスタ A	5.8
0184 003C	DMAグローバル･アドレス･レジスタ B	5.8
0184 0040	DMAチャネル１プライマリ･コントロール	5.2.1
0184 0044	DMAチャネル３プライマリ･コントロール	5.2.1
0184 0048	DMAチャネル１セカンダリ･コントロール	5.10
0184 004C	DMAチャネル３セカンダリ･コントロール	5.10
0184 0050	DMAチャネル１ソース･アドレス	5.7
0184 0054	DMAチャネル３ソース･アドレス	5.7
0184 0058	DMAチャネル１デスティネーション･アドレス	5.7
0184 005C	DMAチャネル３デスティネーション･アドレス	5.7
0184 0060	DMAチャネル１転送カウンタ	5.5
0184 0064	DMAチャネル３転送カウンタ	5.5
0184 0068	DMAグローバル･アドレス･レジスタ C	5.8
0184 006C	DMAグローバル･アドレス･レジスタ D	5.8
0184 0070	DMA補助コントロール･レジスタ	5.9.1

表5−2. DMAコントロール･レジスタ(名前順)

名前	16進によるバイト･アドレス	参照のセクション
DMA補助コントロール･レジスタ	0184 0070	5.9.1
DMAチャネル０デスティネーション･アドレス	0184 0018	5.7
DMAチャネル０プライマリ･コントロール	0184 0000	5.2.1
DMAチャネル０セカンダリ･コントロール	0184 0008	5.10
DMAチャネル０ソース･アドレス	0184 0010	5.7
DMAチャネル０転送カウンタ	0184 0020	5.5
DMAチャネル１デスティネーション･アドレス	0184 0058	5.7
DMAチャネル１プライマリ･コントロール	0184 0040	5.2.1
DMAチャネル１セカンダリ･コントロール	0184 0048	5.10
DMAチャネル１ソース･アドレス	0184 0050	5.7
DMAチャネル１転送カウンタ	0184 0060	5.5
DMAチャネル２デスティネーション･アドレス	0184 001C	5.7
DMAチャネル２プライマリ･コントロール	0184 0004	5.2.1
DMAチャネル２セカンダリ･コントロール	0184 000C	5.10
DMAチャネル２ソース･アドレス	01840014	5.7
DMAチャネル２転送カウンタ	0184 0024	5.5
DMAチャネル３デスティネーション･アドレス	0184 005C	5.7
DMAチャネル３プライマリ･コントロール	0184 0044	5.2.1
DMAチャネル３セカンダリ･コントロール	0184 004C	5.10
DMAチャネル３ソース･アドレス	0184 0054	5.7
DMAチャネル３転送カウンタ	0184 0064	5.5
DMAグローバル･アドレス･レジスタ A	0184 0038	5.8
DMAグローバル･アドレス･レジスタ B	0184 003C	5.8
DMAグローバル･アドレス･レジスタ C	0184 0068	5.8
DMAグローバル･アドレス･レジスタ D	0184 006C	5.8
DMAグローバル･カウント･リロード･レジスタ A	0184 0028	5.5
DMAグローバル･カウント･リロード･レジスタ B	0184 002C	5.5
DMAグローバル･インデックス･レジスタ A	0184 0030	5.7.2
DMAグローバル･インデックス･レジスタ B	0184 0034	5.7.2

5.2.1 DMAチャネル･コントロール･レジスタ

DMAチャネル･プライマリ･コントロール･レジスタ(図5−2)とDMAチャネル･セカンダリ･コントロール･レジスタ(図5−3)は、各チャネル毎に割り当てられており、DMAのチャネルを制御するビット･フィールドを持っています。これらのビット･フィールドを表5−4と表5−3に示します。

図5−2. DMAチャネル･プライマリ･コントロール･レジスタ

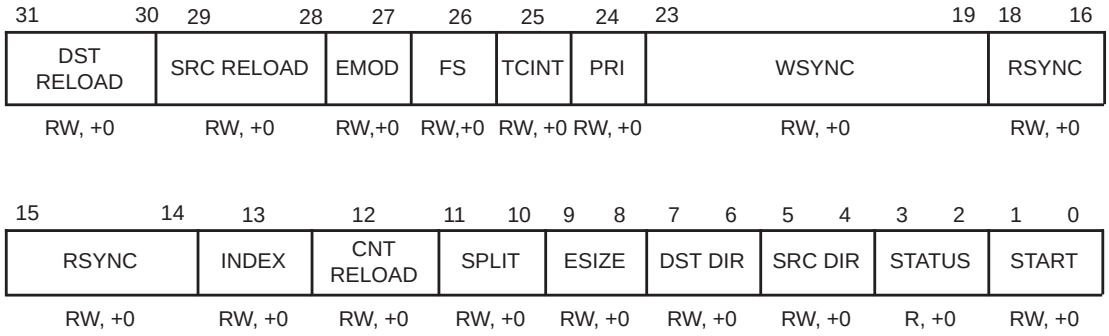


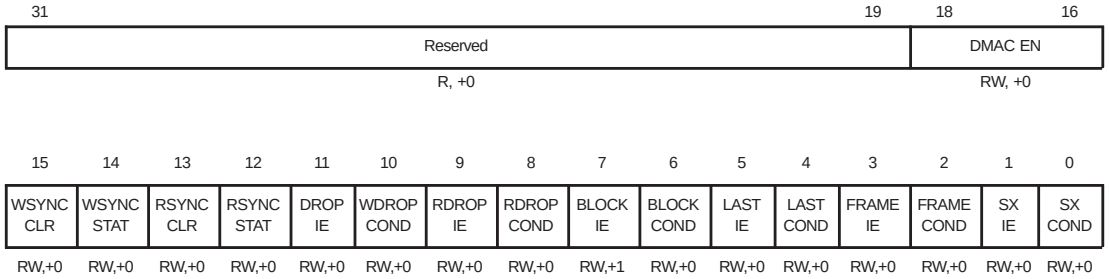
表5−3. DMAチャネル･プライマリ･コントロール･レジスタのフィールド解説

フィールド	解説	セクション
SRC RELOAD, DST RELOAD	自動初期化のためのDMAチャネル･ソース/デスティネーション･アドレスのリロード SRC/DST RELOAD = 00b: 自動初期化の間はリロードなし SRC/DST RELOAD = 01b: DMAグローバル･アドレス･レジスタBをリロードに使用 SRC/DST RELOAD = 10b: DMAグローバル･アドレス･レジスタCをリロードに使用 SRC/DST RELOAD = 11b: DMAグローバル･アドレス･レジスタDをリロードに使用	5.4.1.1
EMOD	エミュレーション･モード EMOD = 0: エミュレーション･ホールトの間、DMAチャネルは動作 EMOD = 1: エミュレーション･ホールトの間、DMAチャネルはポーズ状態	5.13
FS	フレーム同期 FS = 0: 同期なし FS = 1: フレーム全体の同期のためにRSYNCイベントを使用	5.6
TCINT	転送コントローラ割り込み TCINT = 0: 割り込み禁止 TCINT = 1: 割り込み許可	5.10
PRI	優先順位モード:DMAとCPU PRI = 0: CPU優先 PRI = 1: DMA優先	5.9
RSYNC, WSYNC	リード転送/ライト転送同期 (R/W)SYNC = 00000b: 同期なし (R/W)SYNC = その他:同期イベントをセット	5.6

表5−3. DMAチャネル・プライマリ・コントロール・レジスタのフィールド解説 (続き)

フィールド	解説	セクション
INDEX	プログラマブルなインデックスとして使用するDMAグローバル・データ・レジスタを選択 INDEX = 0: DMAグローバル・インデックス・レジスタ A を使用 INDEX = 1: DMAグローバル・インデックス・レジスタ B を使用	5.7.2
CNT RELOAD	自動初期化とマルチフレーム転送のためのDMA転送カウンタのリロード CNT RELOAD = 0: DMAグローバル・カウント・リロード・レジスタ A の値をリロード CNT RELOAD = 1: DMAグローバル・カウント・リロード・レジスタ B の値をリロード	5.4.1.1
SPLIT	スプリット・チャネル・モード SPLIT = 00b: ディスエーブル SPLIT = 01b: イネーブル:DMAグローバル・アドレス・レジスタ A の値をスプリット・アドレスとして使用 SPLIT = 10b: イネーブル:DMAグローバル・アドレス・レジスタ B の値をスプリット・アドレスとして使用 SPLIT = 11b: イネーブル:DMAグローバル・アドレス・レジスタ C の値をスプリット・アドレスとして使用	5.8
ESIZE	エレメント・サイズ ESIZE = 00b: 32ビット ESIZE = 01b: 16ビット ESIZE = 10b: 8ビット ESIZE = 11b: 予約	5.7.3
SRC DIR, DST DIR	エレメント転送後のソース/デスティネーション・アドレス更新 (SRC/DST) DIR = 00b: 更新なし (SRC/DST) DIR = 01b: エレメント・サイズを指定されたバイト数だけインクリメント (SRC/DST) DIR = 10b: エレメント・サイズを指定されたバイト数だけデクリメント (SRC/DST) DIR = 11b: INDEXにより選択されたDMAグローバル・インデックス・レジスタにより調整	5.7.1, 5.7.2
STATUS	STATUS = 00b: ストップ状態 STATUS = 01b: 自動初期化なしで動作中 STATUS = 10b: ポーズ状態 STATUS = 11b: 自動初期化後に動作中	5.4
START	START = 00b: ストップ START = 01b: 自動初期化なしのスタート START = 10b: ポーズ START = 11b: 自動初期化つきスタート	5.4

図5−3. DMAチャネル･セカンダリ･コントロール･レジスタ



5

表5−4. DMAチャネル･セカンダリ･コントロール･レジスタのフィールド解説

フィールド	解説	セクション
SX COND FRAME COND LAST COND BLOCK COND (R/W)DROP COND	DMAの状態、各ビットがそれぞれの状態を表示 COND = 0: 状態検出なし COND = 1: 状態検出あり	5.10
SX IE FRAME IE LAST IE BLOCK IE (R/W)DROP IE	DMAコンディション割り込みイネーブル IE = 0: 対応する状態の検出によるDMAチャネル割り込みを禁止 IE = 1: 対応する状態の検出によるDMAチャネル割り込みを許可	5.10.1
(R/W)SYNC STAT	リード、ライト同期状態 1のライトにより関連する状態をセット STAT = 0: 同期受信なし STAT = 1: 同期受信あり	5.6.1
DMAC EN	DMACピン制御 DMAC EN = 000b: DMACピンをロー･レベルに固定 DMAC EN = 001b: DMACピンをハイ･レベルに固定 DMAC EN = 010b: DMACピンはRSYNC STATの状態を出力 DMAC EN = 011b: DMACピンはWSYNC STATの状態を出力 DMAC EN = 100b: DMACピンはFRAME CONDの状態を出力 DMAC EN = 101b: DMACピンはBLOCK CONDの状態を出力 DMAC EN = その他:予約	5.12
(R/W)SYNC CLR	リード/ライト同期状態をクリア リード値は0、1のライトにより関連する状態をクリア	5.6.1

C6202のDMAチャネル･セカンダリ･コントロール･レジスタは、WSPOL、RSPOL、FSIGという3つの新しいフィールドが増えています。このフィールドは、フレーム同期データ転送のコントロールの追加に使われます。C6202セカンダリ･コントロール･レジスタを図5−4に示します。新しいフィールドはグレイで表示されています。表5−5では、新しいフィールドの可能な設定について解説しています。

図5−4. TMS320C6202セカンダリ･コントロール･レジスタ

31	22	21	20	19	18	16
Reserved		WSPOL	RSPOL	FSIG	DMAC EN	
R, +0		RW, +0	RW, +0	RW, +0	RW, +000	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WSYNC CLR	WSYNC STAT	RSYNC CLR	RSYNC STAT	DROP IE	WDROP COND	RDROP IE	RDROP COND	BLOCK IE	BLOCK COND	LAST IE	LAST COND	FRAME IE	FRAME COND	SX IE	SX COND
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+1	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0

表5−5. 同期コンフィギュレーション･オプション

フィールド	解説	セクション
WSPOL/ RSPOL	同期イベント極性 外部同期イベントの極性を選択 1＝アクティブ･ロー、0=アクティブ･ハイ このフィールドはEXT_INTxが選択されたときのみ有効です。	5.6.3
FSIG	レベル/エッジ判定モード選択 FSIGはフレーム同期なし転送のときには0にセットしなければいけません。 FSIG=1:エッジ判定モード(FS=1またはFS=0) FSIG=0:レベル判定モード(FS=1のときのみ有効) レベル判定モードでは、フレーム転送中に受信する同期信号は、フレーム転送が完了したあとにもセットされていない限り、無視されます。	5.6.3

5.2.2 レジスタ･アクセス･プロトコル

DMA転送の設定のときは、次の手順に従います。

1) プライマリ･コントロール･レジスタで、START=00bに設定
2) セカンダリ･コントロール･レジスタで、(W/R)SYNC_CLR=1に設定
3) ソース/デスティネーション･アドレス･レジスタとカウント･レジスタの設定
4) プライマリ･コントロール･レジスタで、START=01bまたは11bに設定

5.3 メモリー･マップ

DMAでは、「第10章 **ブート･モード及びコンフィギュレーション**」に示すデバイスのメモリー･マップを前提としています。リクエストは、次の6つのリソースのいずれかに送られます。

- ❑ 拡張バス
- ❑ 外部メモリー･インターフェイス
- ❑ 内部プログラム･メモリー(ブロック0,ブロック1)
- ❑ 内部ペリフェラル･バス
- ❑ 内部データ･メモリー

ソース･アドレスは、ブロック転送が実行されている間、これらの領域のいずれか1つを指し示しているものとして取扱われます。このような制約は、デスティネーションのアドレスについてもあてはまります。

5.4 ブロック転送の初期化

DMAの各チャネルは、それぞれ独立にスタートさせることができます。スタートは、CPUによる直接のアクセス、または自動初期化により行なうことができます。また、DMAの各チャネルは、CPUの直接アクセスにより、それぞれ独立にストップさせ、ポーズさせることができます。各DMAチャネルの状態は、DMAチャネル･プライマリ･コントロール･レジスタのSTATUSフィールドをリードすることにより確認できます。

マニュアルによるスタート: 各DMAコントロール･レジスタにそれぞれ適切な値がライトされている場合に、特定のチャネルのDMA動作をスタートさせるには、対応するDMAチャネル･コントロール･レジスタに対して設定値をライトし、START=01bとします。このライト動作は、すでにスタートしているDMAチャネルに対しては影響を与えません。DMA動作がスタートすると、STATUS=01bとなります。

ポーズ: DMAの動作がスタートした後では、DMAチャネルは、START=10bのライトによりポーズさせることができます。ポーズの状態となると、DMAは、すでにリード転送リクエストの完了しているものについて、ライト転送を完了します。また、DMAが、リード同期を受け取っていれば、もう1つのエレメントの転送が許可されます。ポーズ状態となると、ライト転送を完了した後に、STATUS=10bとなります。

ストップ: DMAは、START=00bのライトを行なうことによりストップさせることができます。ストップ時の動作は、ポーズ時の動作と同じです。DMAによる転送が完了すると、自動初期化がイネーブルされていない限り、ライト転送を完了した後に、DMAチャネルはストップ状態に戻り、STATUS=00bとなります。

5.4.1 自動初期化

DMAによるブロック転送完了時に、DMAを自動的に再初期化させることができます。一部のDMAコントロール･レジスタは、選択されたDMAグローバル･データ･レジスタを介して次のブロック転送のためにプリロードすることができます。この機能を使えば、次のブロック転送のために、あらかじめパラメータをセットしておくことができます。

連続動作: 通常の連続動作では、現行のブロック転送の最後のライト転送が完了した直後で、次のブロック転送のリード同期の前に、CPUは次のブロック転送の設定を行なう必要があります。しかし、自動初期化を用いると、現行のブロック転送が始まった後の任意のタイミングで、CPUがリロード･レジスタに次の転送のためのパラメータをロードしておくと、現行のブロック転送の終了後、DMAはその値をリードし、次の転送を開始することができます。

繰り返し動作: この動作は、連続動作のうちの特別なものです。ブロック転送が終了すると、DMAは、直前のブロック転送を繰り返します。この場合には、CPUは、それぞれのブロック転送のためにリロード･レジスタをリロードすることはせず、最初のブロック転送の前にのみ、これらのレジスタのロードを行ないます。

自動初期化のイネーブル: DMAチャネル･コントロール･レジスタにSTART=11bのライトを行なうことにより、自動初期化がイネーブルされます。この場合には、ブロック転送が終了すると、選択されたDMAチャネル･レジスタがリロードされ、DMAチャネルが再びスタートします。ポーズの後に再スタートを行なう場合には、STARTに11bをライトして、自動初期化をイネーブルします。

自動初期化から非自動初期化への切り替え: 特定のチャネルのDMAを終了させるために、自動初期化から非自動初期化へ切り替えることが可能です。モードを切り替えるには、プライマリ･コントロール･レジスタで、START=10bとして動作中のチャネルをポーズした後、START=01bとして再スタートします。
動作中のチャネルがスプリット･モードの場合、自動初期化から非自動初期化への切り替えがフレーム境界で起こらないようにする必要があります。送信ソース側においてフレームnで、受信デスティネーションにおいてフレームn−1のときにチャネルがポーズされた場合、そのチャネルは自動初期化(START=11b)で再スタートし、モードの切り替えが起こる前に再びポーズしなければいけません。これにより、送信と受信の両方のデータ･ストリームが同じ番号のフレームを完了したことを保証できます。

5.4.1.1 DMAチャネル･リロード･レジスタ

自動初期化モードでは、同様の条件でのブロック転送が連続して行われるものと仮定されます。このため、ブロック転送の間に書き換えられるレジスター転送カウンタとアドレス･レジスタについてのみリロードが可能となります。DMAチャネル転送カウンタ、DMAチャネル･ソース･アドレス･レジスタ、およびDMAチャネル･デスティネーション･アドレス･レジスタには、対応するリロードレジスタが存在します。どのリロード･レジスタを使うかは、DMAチャネル･プライマリ･コントロール･レジスタのRELOADフィールド(図5−2)で決定されます。

自動初期化モードにおいて、ソース･アドレス･レジスタやデスティネーション･アドレス･レジスタをリロードしないように設定することもできます。この機能により、ブロック転送の間を通じて、あるレジスタがその値を保持し続けるようにすることができます。これにより、ブロック転送中に値が変わらないものに対して、DMAグローバル･データ･レジスタを割り当てる必要がなくなります。あるチャネルでは、複数のチャネル･レジスタについて同じ値を使っている場合もあります。例えば、スプリット･モードでは、ソース･アドレスとデスティネーション･アドレスが同じである場合があります。同様に、複数のチャネルが同じリロード･レジスタを使用していることもあります。例えば、1つのチャネルが同じ転送カウント･リロード･レジスタを使用している場合が考えられます。

ブロック転送が完了すると、チャネル･レジスタには、対応するリロード･レジスタの値がリロードされます。DMAチャネル転送カウント･レジスタの場合には、リロードはブロック転送全体の終了時ではなく、それぞれのフレーム転送の終了時に発生することに注意してください。DMAチャネル転送カウンタへのリロードは、自動初期化がイネーブルな時だけでなく、マルチ･フレームの転送が設定されている場合にも必要とされます。

5.11.2節で説明するように、DMAは、ライト転送の前にリード転送を行なうことができます。そのために、ブロックとフレームの終わりに必要なリロードを、DMAチャネルのリード(ソース)とライト(デスティネーション)に対して独立に発生させることができます。同様に、5.9節で解説するスプリット･チャネル動作の場合では、対応する送信または受信のエレメント転送が完了した時点で、ソース･アドレスとデスティネーション･アドレスがそれぞれ独立にリロードされます。

DMAチャネル転送カウンタへのリロード値のユーザーによる書き換えは、現行のブロック転送の中で最後から2番めのフレームの転送が完了した後に行なって下さい。これ以外の場合には、新しいリロード値は、現行のブロック転送の中の後続のフレームのバウンダリに影響を与えます。しかしながら、現行のブロック転送と次回のブロック転送でのフレームのサイズが同じであれば、このような制約はありません。DMAチャネル転送カウンタの詳細については、5.5節を参照してください。

注 :自動初期化モード時に、拡張バスでないソース/デスティネーション･アドレスから拡張バスのソース/デスティネーション･アドレスに切り換えることはできません。同様に、拡張バスから拡張バスでないソース/デスティネーション･アドレスへの切り替えもできません。

5.5 転送カウント

DMAチャネル転送カウント･レジスタ(図5−5)は、転送すべきフレーム数とフレームあたりのエレメント数を設定するビットフィールドを持っています。図5−6に、DMAグローバル･カウント･リロード･レジスタを示します。

FRAME COUNT: 符号なし16ビット値をこのフィールドにセットすることにより、ブロック転送中のフレームの総数を設定します。1ブロックあたりのフレーム数は、最大65535となります。このカウンタは、フレーム転送の中での最後のリード転送の完了によりデクリメントされます。最後のフレームが転送されると、カウンタ全体が、DMAチャネル･プライマリ･コントロール･レジスタのCNT RELOADフィールドで選択されるDMAグローバル･カウント･リロード･レジスタよりリロードされます(5.4.1.1節参照)。FRAME COUNTの初期値が0と1の場合には、いずれも単一のフレーム転送になることにご注意ください。

ELEMENT COUNT: 符号なし16ビット値をこのフィールドにセットすることにより、フレームあたりのエレメントの数を設定します。このカウンタは、それぞれのエレメントのリード転送の後でデクリメントされます。1フレームあたりのエレメント数は、最大65535となります。それぞれのフレームで最後のエレメントに達すると、ELEMENT COUNTには、DMAチャネル･プライマリ･コントロール･レジスタのCNT RELOADフィールドで選択されるDMAグローバル･カウント･リロード･レジスタの下位16ビットがリロードされます。このリロード動作は、自動初期化モードの影響を受けません。ブロック転送が始まる前に、カウンタと選択されたDMAグローバル･カウント･リロード･レジスタに同じ16ビットLSBをロードして、最初のフレームとそれ以後のフレームで、フレームあたりのエレメントの数が同じになるようにします。マルチ･フレーム転送では、リロード値は、自動初期化がイネーブルされている時だけでなく、すべての場合に指定する必要があります。エレメントの数が0に初期化された場合には、動作は未定義となります。

図5−5. DMAチャネル･転送カウント･レジスタ

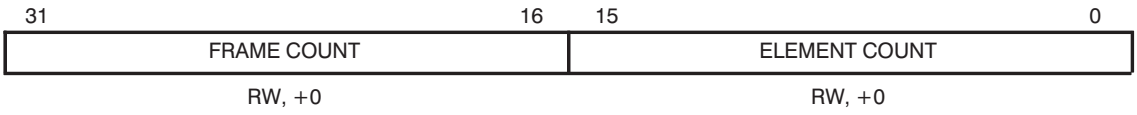
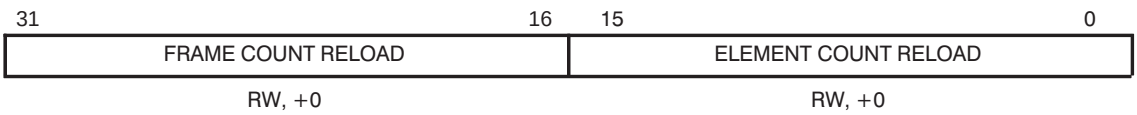


図5−6. 転送カウンタのリロードに使用されるDMAグローバル･カウント･リロード･レジスタ



5.6 同期:DMA転送のトリガ

同期を使用することにより、DMA転送を内部ペリフェラルや外部ピンからの割り込み等のイベントによりトリガすることができます。それぞれのチャネルについて、次の3種類の同期方法をイネーブルすることができます。

- ❑ リード同期: 選択されたイベントの発生によりリード転送を開始します。
- ❑ ライト同期: 選択されたイベントの発生によりライト転送を開始します。
- ❑ フレーム同期: 選択されたイベントの発生によりフレーム転送を開始します。

同期イベントの選択:DMAチャネル・プライマリ・コントロール・レジスタのRSYNCとWSYNCフィールドにより、イベントを選択することができます。このレジスタで、FS=1に設定すると、RSYNCで選択されたイベントが、フレーム同期のイベントになります。このとき、WSYNCは00000bにセットされていなければいけません。チャネルがスプリット・モード(SPLIT≠00b)で動作するよう設定されている場合にはRSYNC及びWSYNCは0ではない値が設定されている必要があります。最大31のイベントを使用することができます。これらのフィールドが00000bに設定されると、同期なしとなります。この場合には、リード、ライト、またはフレーム転送は、そのチャネルへのリソースが利用可能となった時点で発生します。これらのフィールドの値とイベントとの関係を表5−6に示します。ここで、このフィールドが、割り込みセレクタのものと非常によく似ていることにご注意ください。「13.4 **割り込みセレクタの設定**」を参照してください。違いとしては、McBSPが、割り込みではなくDMA同期イベントを生成することです。これ以外のただ1つの違いとしては、DSPINTがエンコードされる位置です。

表5−6. 同期イベント

イベント番号 (バイナリ)	イベントの略号	イベントの解説
00000	なし	同期なし
00001	TINT0	タイマ０割り込み
00010	TINT1	タイマ１割り込み
00011	SD_INT	EMIF SDRAM タイマ割り込み
00100	EXT_INT4	外部割り込みピン４
00101	EXT_INT5	外部割り込みピン５
00110	EXT_INT6	外部割り込みピン６
00111	EXT_INT7	外部割り込みピン７
01000	DMA_INT0	DMAチャネル０割り込み
01001	DMA_INT1	DMAチャネル１割り込み

表5−6. 同期イベント(続き)

イベント番号 (バイナリ)	イベントの略号	イベントの解説
01010	DMA_INT2	DMAチャネル２割り込み
01011	DMA_INT3	DMAチャネル３割り込み
01100	XEVT0	McBSP０送信イベント
01101	REVT0	McBSP０受信イベント
01110	XEVT1	McBSP１送信イベント
01111	REVT1	McBSP１受信イベント
10000	DSPINT	ホスト･ポート、ホストからDSPへの割り込み
10001	XEVT2	McBSP2送信イベント
10010	REVT2	McBSP2受信イベント
その他	予約	

5.6.1 DMAチャネル･イベント･フラグのラッチ

DMAチャネル･セカンダリ･コントロール･レジスタ(表5−4)は、リード同期及びライト同期(RSYNC及びWSYNC)イベントのためのSTAT及びCLRフィールドを持っています。

DMA同期イベントのラッチ: 選択されたイベントのローからハイへの遷移(WSPOL及びRSPOLの設定によってはハイからローへの遷移)は、各DMAチャネルによってラッチされます。この遷移が発生すると、DMAチャネル･セカンダリ･コントロール･レジスタの中の対応するSTATフィールドがセットされます。同期なしの場合には、STATビットのリード値は、常に1であることに注意してください。また、1つのイベントが複数のアクションを引き起こす場合もあることにもご注意ください。

ユーザによるイベントのセットとクリア: ペンディングとなっているイベントをブロック転送をスタートする前にクリアすることにより、DMAチャネルを次のイベントの発生まで待機させることができます。逆に、イベントをブロック転送をスタートする前にセットすることにより、最初のエレメントの転送のために必要なイベントを強制的に発生させることができます。イベントのクリアとセットは、それぞれ対応するCLRまたはSTATのフィールドに1をライトすることにより行ないます。これらのビットへの0のライトは動作に影響を与えません。また、CLRビットのリード値は常に0であり、対応する記憶ビットはありません。セットとクリアのビットが別々にあることにより、他のビットを考慮することなくあるビットのセット/クリアを行なうことができます。ユーザによるイベントの操作は、同時に発生する自動的なイベントのセットやクリアよりも優先されることに注意してください。

5.6.2 イベントの自動クリア

同期イベントによりラッチされたSTATは、イベントに対応するアクションが完了した時にのみ自動的にクリアされます。イベントのクリアは、同期イベント間の時間を最小とするために直ちに行われます。この機能により、イベントの認識のためのスループットの向上が図られます。以下の各種類の同期について、この動作を詳しく説明します。

- ❑ リード同期状態のクリア: DMAが対応するリード転送のリクエストを完了すると、リード同期に対するラッチ状態がクリアされます。
- ❑ ライト同期状態のクリア: DMAが対応するライト転送のリクエストを完了すると、ライト同期に対するラッチ状態がクリアされます。
- ❑ フレーム同期状態のクリア: DMAが新しいフレームの最初のリード転送に対するリクエストを完了すると、フレーム同期がRSYNC STATフィールドをクリアします。

5.6.3 同期コントロール

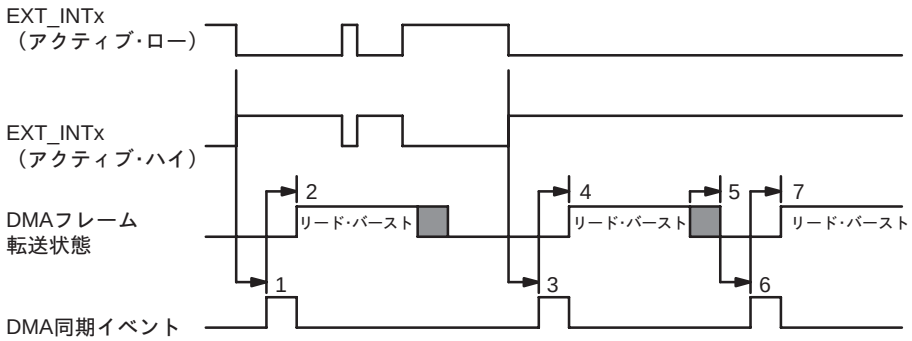
C6202のDMAでは、外部同期イベントを認識する方法についてより柔軟なコントロールが可能です。WSPOL及びRSPOLを1にセットすることで、外部イベントの極性をアクティブ・ローに変えることができます。WSPOLではライト同期転送、RSPOLではリード同期転送の設定を行ないます。

フレーム同期転送では、バースト転送中に外部割り込みを同期イベントとして認識しないよう設定することが可能です(FSIG=1に設定)。チャネルは、バーストの状態を内部でモニターしており、フレーム転送が行なわれていないときのみ、同期イベントとしてラッチします。

図5−7では、目的の同期イベントを発生するまでの流れを示しています。この図では、アクティブ･ハイとアクティブ･ローの両方が示されていますが、以下の説明ではアクティブ･ローの動作を参照してください。

1) バースト転送が行なわれていないときに、EXT_INT4のハイからローへの遷移が同期イベントをトリガします。
2) 同期イベントがフレーム転送をトリガし、DMA同期イベントを制限します。この同期イベントの間は、EXT_INT4の遷移は無視されます。
3) 1)と同じ
4) 2)と同じ
5) 内部でリード･バーストが完了した後、EXT_INT4がまだアクティブかどうかをチェックするために、32CPUクロック分のディレイがあります。
6) バースト転送とディレイの後、EXT_INT4がまだアクティブなので、DMA内で新しいイベントが発生します。
7) 新しいDMA同期イベントが、またバースト転送をトリガします。

図5−7. 同期フラグ



新しい同期モードは、データ･バッファとして動作する外部FIFOとのインターフェイスに有効です。FIFO中の現在のデータ量を示すフラグで同期イベントがトリガされるので、競争状態が起こる可能性があります。DMAがFIFOからリードしようとしていて(同期イベントによって立てられるフラグをクリアする)、直後にFIFOに新しいエレメントがライトされる場合、フラグはリセットされ、現行のバースト転送に続いてすぐに新しいフレームが開始される可能性があります。現行のバースト転送の間、イベントを無視するようDMAを設定することで、この状況を避けることができます。

この設定のもうひとつの特徴は、バースト転送の間、同期イベントがアクティブな場合、次のバースト転送で再びラッチされます。これもFIFOインターフェイスとして強力なものとしています。これは、FLAGのアクティブからインアクティブへの遷移はバーストの間にのみ起こりうることに起因します。例えば、C6202がFIFOからリードする立場のとき、FIFOのハーフ･フル信号(HF)がアクティブからインアクティブに変わるのは、FIFOからデータを読んだときのみになります。フラグがバーストの間アクティブである場合、C6202が次のフレームをリードする準備ができる前にFIFOが新しいデータを用意したことになります。

フレーム転送が完了した後、フラグがまだアクティブであるかを判定するためにDMAは32CPUクロック･サイクル分のウェイトをいれます。このディレイは、外部FIFOにフラグの更新をさせ、DMA内で登録される前に内部レジスタを介してフラグを認識させるためのものです。例として、典型的なFIFOは外部フラグを更新するのに、1から3のFIFOクロック･サイクルを必要とします。XRCLKの出力の分周比によりますが、これは24CPUクロック･サイクルと置き換えることができます(x8モードのとき)

これらの機能は、WSPOL、RSPOL、FSIGが適切に設定されたDMAでのみ使用できます。全てのフィールドが0のままの場合(デフォルト)、C6202 DMAの機能はC6201 DMAと同じです。

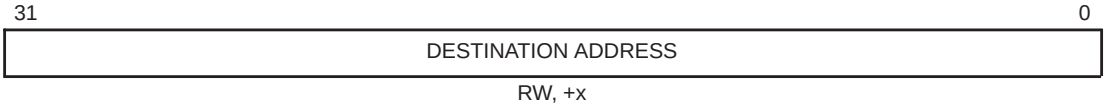
5.7 アドレス生成

各チャネルについて、DMAは、リード転送とライト転送のそれぞれについてアドレスの計算を行ないます。DMAでは、さまざまなデータ構造の生成が可能です。例えば、DMAは、nエレメント毎にインクリメントする配列を自由に移動することができます。DMAの設定により、フレームの中での位置が同じエレメントを効率的に処理し、グループ化することができます。

DMAチャネル･ソース･アドレス･レジスタ(図5−8)とDMAチャネル･デスティネーション･アドレス･レジスタ(図5−9)は、それぞれ次のリード転送とライト転送のためのアドレスを保持します。

図5−8. DMAチャネル･ソースアドレス･レジスタ

図5−9. DMAチャネル･デスティネーション･アドレス･レジスタ



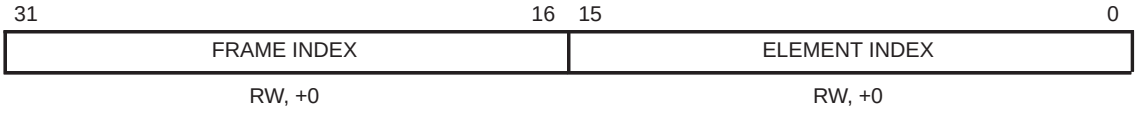
5.7.1 基本アドレス調整

表5−3に示すように、SRC DIRとDST DIRのフィールドの値により、DMAチャネル･ソース/デスティネーション･アドレス･レジスタに対し、それぞれ、エレメント･サイズだけインクリメント/デクリメント、グローバル･インデックス･レジスタの使用、値を更新しない、のいずれかを設定することができます。デフォルトでは、これらの値は00bに設定され、アドレス更新がディスエーブルにされています。インクリメントかデクリメントが選択されると、アドレス調整量がエレメントサイズのバイト数として設定されます。例えば、ソース･アドレスがインクリメントに設定され、16ビット･ハーフ･ワードで転送中の場合、アドレスは、リード転送ごとに2バイトづつインクリメントされます。

5.7.2 グローバル･インデックス･レジスタによるアドレス調整

DMAチャネル･プライマリ･コントロール･レジスタのINDEXフィールドにより、特定のDMAグローバル･インデックス･レジスタ(図5−10)が選択されます。基本アドレス調整の場合と異なり、このモードでは、エレメント転送が現行のフレームの中で最後のものかどうかによって、異なった調整量とすることができます。基準調整量(ELEMENT INDEX)は、選択されたDMAグローバル･インデックス･レジスタの下位16ビットに含まれています。フレームの終端に対する調整量(FRAME INDEX)は、選択されたDMAグローバル･インデックス･レジスタの上位16ビットによって決定されます。これらのフィールドは、符号付き16ビットの値を持っています。このため、インデックスの量は、−32768から32767までとなります。

図5−10. DMAグローバル･インデックス･レジスタ



これらのフィールドは、アドレス調整に次のような影響を与えます。

- ELEMENT INDEX: エレメント転送のうちフレームの最後のものを除き、ELEMENT INDEXは、対応するDMAチャネルのソース/デスティネーション･アドレス･レジスタに対する、リード転送またはライト転送のたびごとの変位を決定します。
- FRAMEINDEX: 各フレーム転送の中の最後のエレメントに対するリード転送またはライト転送に対しては、FRAME INDEXフィールド(ELEMENT INDEXではない)が、アドレスの調整に使用されます。この調整は、単一のフレーム転送、マルチ･フレーム転送のいずれにおいても発生します。

5.7.3 エレメント･サイズ、アラインメント、エンディアン

ユーザは、DMAチャネル･プライマリ･コントロール･レジスタのESIZEフィールドを使用して、DMAによる各転送ごとに、8ビット･バイト、16ビット･ハーフ･ワード、32ビット･ワード毎の転送を設定できます。次のレジスタとフィールドは、正確にアラインメントされた値でロードする必要があります。

- DMAチャネル･ソース･アドレス･レジスタとデスティネーション･アドレス･レジスタ、およびこれらに対応するリロード･レジスタ
- ELEMENT INDEX
- FRAME INDEX

ワード転送の場合には、これらのレジスタには、ワード･アドレスに対応するように、4の倍数を設定しなければなりません。ハーフ･ワードの場合には、2の倍数とする必要があります。もし、ロードされる値が適切にアラインメントされていない場合には、動作は未定義となります。バイト転送の場合には、このような制約はありません。プログラム･メモリーに対するアクセスは、すべて32ビット幅で行なう必要があります。また、32ビットのレジスタの中の特定の8ビットまたは16ビットを使用する場合には、エンディアンを把握しておく必要があります。例えば、リトル･エンディアンでは、00bで終わるアドレスが最下位バイトとなり、ビッグ･エンディアンでは、11bで終わるアドレスが最下位バイトとなります。

5.7.4 アドレスのリロードにフレーム･インデックスを使用

自動初期化が行われる単一フレームのブロック転送においては、リロード･レジスタのかわりにFRAME INDEXを使って次のアドレスの再計算をすることができます。次の例では、1つのフレーム転送により同じデータが 1 バイトおきに10バイト分転送されます。

- ❑ SRC DIR=00b、ソース･アドレスは一定
- ❑ DST DIR=11b、グローバル･インデックス･レジスタを使用
- ❑ ELEMENT INDEX=10b、デスティネーションの増加幅2バイト
- ❑ FRAME INDEX=−(9x2)=−18=FFEEh、デスティネーションを同じ位置からスタートさせるために、位置を−18バイトだけ補正

5.7.5 サイズの大きな単一ブロックを転送する

ELEMENT　COUNTをFRAME　COUNTと組み合わせて使用することにより、サイズが65535以上の単一フレームのブロック転送を効率的に行なうことができます。ここで、エレメント･カウントとフレーム･カウントの積により、大きな実効エレメント･カウントを得ることができます。これは、次の手順で行なう必要があります。

- ❑ アドレスがグローバル･インデックス･レジスタを使って調整されるように設定されている場合には(DIR=11b)、FRAME INDEXと、ELEMENT INDEXの値が一致していなければなりません。これは、ソース･アドレス、デスティネーション･アドレスのいずれについてもあてはまります。アドレスがグローバル･インデックス･レジスタを使って調整されるように設定されていない場合には、デフォルト設定によりエレメントとフレームのバウンダリで同じアドレス調整が行われるため、この制約はあてはまりません。
- ❑ フレーム同期は、ディスエーブル(DMAチャネルのプライマリ･コントロール･レジスタのFS＝0)とする必要があります。これにより、大きなブロックの途中での同期が不要となります。
- ❑ 最初のフレームの中でのエレメントの数をEiとします。後続のフレームにおけるエレメントの数は((F−1)×Er)となります。実効エレメント･カウントは、((F−1)×Er)+Eiとなります。

  ここで、

  F=FRAME COUNTの初期値
  Er=ELEMENT COUNTのリロード値
  Ei=ELEMENT COUNTの初期値

  となります。

  これにより、128K+1のエレメントを転送するには、例えば、F=5,Er=32K,Ei=1と設定できます。

5.7.6 ソート処理

次の手順は、フレーム中での位置が同じエレメントをメモリー上の連続した領域に転送するためのものです。(例えば、第1フレームの第1エレメントの次に第2フレームの第1エレメント、といった場合)

- ❑ ELEMENT INDEXにF×Sをセットする。
- ❑ FRAME INDEXを−(((E−1)×F)−1)×Sにセットする。

ここで、

E = ELEMENT COUNT(フレームあたりのエレメント数)とELEMENT COUNT RELOADの初期値
F = FRAME COUNT(フレームの総数)の初期値
S = バイト数であらわしたエレメント･サイズ

ここで、4つのハーフ･ワード･エレメント(E=4,S=2)からなる3つのフレーム(F=3)を転送する場合について考えてみましょう。これは、ELEMENT INDEX=3×2=6、FRAME INDEX=−(((4−1)×3)−1)×2=FFF0hの場合に相当します。ここで、ソース･アドレスについての更新はないものとし、デスティネーションは、8000 0000hからインクリメントするものとします。表5−7は、データを転送の順に示したものであり、表5−8は、転送が終わった後のメモリー上でのデータの配置を示すものです。

表5−7. DMA転送によるソート例

フレーム	エレメント	アドレス(16進)	調整量
0	0	8000 0000	+6
0	1	8000 0006	+6
0	2	8000 000C	+6
0	3	8000 0012	–16
1	0	8000 0002	+6
1	1	8000 0008	+6
1	2	8000 000E	+6
1	3	8000 0014	–16
2	0	8000 0004	+6
2	1	8000 000A	+6
2	2	8000 0010	+6
2	3	8000 0016	–16

表5−8. ソートによるグループ化

アドレス(16進)	フレーム	エレメント
8000 0000	0	0
8000 0002	1	0
8000 0004	2	0
8000 0006	0	1
8000 0008	1	1
8000 000A	2	1
8000 000C	0	2
8000 000E	1	2
8000 0010	2	2
8000 0012	0	3
8000 0014	1	3
8000 0016	2	3

5.8 スプリット･チャネル動作

スプリット･チャネル動作によって、1つのDMAチャネルをあたかも2つのチャネルのように使用して、特定のアドレスを持った外部または内部のペリフェラルに対する入力(受信)と出力(送信)を同時に扱うことができます。

5.8.1 スプリットDMAの動作

スプリット･チャネル動作は、送信エレメントの転送と、受信エレメントの転送により構成されます。それぞれの転送は、リード転送とライト転送が交互に組み合わされたものです。

- 送信エレメント転送
  - 送信リード転送: データは、DMAチャネルのソース･アドレスからリードされます。次に、ソース･アドレスは、設定に従って調整されます。続いて、転送カウントがデクリメントされます。このイベントは、同期なしです。
  - 送信ライト転送: 送信リード転送からのデータは、スプリット･デスティネーション･アドレスにライトされます。このイベントは、WSYNCのフィールドの設定に従って同期化されています。DMAチャネルでは、ペンディングとなっている受信転送の数を保持しています。
- 受信エレメント転送
  - 受信リード転送: データは、スプリット･ソース･アドレスからリードされます。このイベントは、RSYNCフィールドに従って同期化されています。
  - 受信ライト転送: 受信リード転送からのデータは、デスティネーション･アドレスにライトされます。次に、デスティネーション･アドレスは、設定に従って調整されます。このイベントは、同期なしです。

チャネル毎に1つのエレメント･カウントと1つのフレーム･カウントしか存在しないため、エレメント･カウントとフレーム･カウントは、受信データと送信データの双方について、同じになることに注意する必要があります。スプリット動作を正常に行なうためには、RSYNCとWSYNCのフィールドの双方を0以外の同期イベントに設定しておく必要があります。また、スプリット･モードでは、フレーム同期をディスエーブルの状態としておかなければなりません。

すべての転送は上記のシーケンスにしたがって行われます。しかしながら、送信エレメント転送を開始するために、それ以前のすべての受信エレメント転送が完了している必要はありません。このため、場合によっては、送信ストリームが受信ストリームに先行することがあります。DMAチャネル転送カウンタは、対応する送信転送が完了してからデクリメント(または再初期化)されます。ところが、ソース･アドレス･レジスタの再初期化は、すべての送信エレメントの転送が完了してから行われます。この設定は、受信転送の前に8つ以上の送信転送がなされない限りにおいて有効です。これ以上多くの送信転送が受信転送の前に行われると、送信エレメント転送がストップされ、場合によっては、同期イベントを失うことになります。受信または送信のエレメント転送が、互いに他の転送に先立って7つ以下しか行われない場合には、DMAチャネルは、その情報を内部状態として保持します。

5.8.2 スプリット･アドレスの生成

DMAプライマリ･コントロール･レジスタのSPLITフィールドにより選択されたDMAグローバル･アドレス･レジスタは、以下のスプリット転送においてアクセスされるペリフェラルのアドレスを決定します。

- スプリット･ソース･アドレス: このアドレスは、C6000への入力ストリームのソースとなります。選択されたDMAグローバル･アドレス･レジスタによって、このスプリット･ソース･アドレスが指定されます。
- スプリット･デスティネーション･アドレス: このアドレスは、C6000からの出力ストリームのデスティネーションとなります。スプリット･デスティネーション･アドレスは、スプリット･ソース･アドレスより1ワード･アドレス(4バイト･アドレス)だけ大きいものとして取扱われます。

図5−11. スプリット･アドレスに使用されるDMAグローバル･アドレス･レジスタ

最下位2ビットは、ワード･アドレスにアラインメントするために0に固定されています。3番目のビットが0となっているのは、スプリット･ソース･アドレスは、偶数のワード･バウンダリにあると仮定されているためです。同様にして、スプリット･デスティネーション･アドレスは、奇数のワード･バウンダリにあると仮定されます。これらの関係は、転送の幅に関係なくあてはまります。外部ペリフェラルについては、この規則にしたがって、アドレスのデコードの方法を設計しておく必要があります。

注 :スプリット･モードは拡張バスでは使えません。ソース･アドレス、デスティネーション･アドレス、スプリット･アドレスは、拡張バスI/Oメモリー範囲内にありません。

5.9 リソースのアービトレーションと優先順位の設定

複数のリクエストが生じたときに、競合するどのリクエスタにリソースの制御を与えるかは優先順位により決まります。リクエスタとしては、以下のものがあります。

- ❑ DMAチャネル
- ❑ CPUのプログラム及びデータのアクセス

リソースとしては、以下のものがあります。

- ❑ 内部データ･メモリー
- ❑ 内部プログラム･メモリー
- ❑ ペリフェラル･バスを介してアクセスされる内部ペリフェラル･レジスタ
- ❑ 外部メモリー･インターフェイス(EMIF)を介した外部メモリー
- ❑ 拡張バスを介してアクセスされる拡張メモリー

優先順位は、設定可能です。

- ❑ DMAとCPUの優先順位: 各DMAチャネルは、対応するDMAチャネル･コントロール･レジスタのPRIビットを設定することにより、それぞれ、独立にCPUよりも優先順位を高く設定することができます。DMA補助コントロール･レジスタのAUXPRIフィールドは、補助チャネルについての同様の設定を行なうためのものです。CPUよりも優先順位が高いモードでは、対応するチャネルからのリクエストは、優先順位の高いことを示す信号とともに、対応するリソースに送られます。デフォルトでは、これらのフィールドは0に設定され、CPUよりも優先順位が低いモードになっています。各リソースでは、この信号を、それぞれの手順に従って処理することにより、コンフリクトを回避することができます。この信号の処理手順については、それぞれのリソースのドキュメントを参照してください。
- ❑ DMAチャネル間における優先順位: DMAでは、チャネル0を最高位の優先順位、チャネル3を最下位の優先順位とする固定的な優先順位の処理手順を持っています。補助レジスタについては、これらの階層のどの優先順位にも指定することができます。

5.9.1 DMA補助コントロール･レジスタとチャネル間における優先順位

DMA補助コントロール･レジスタのフィールドによって、補助チャネルは制御されます。このレジスタのフィールドを図5−12に示します。また、それをまとめたものを、表5−9に示します。

図5−12. DMA補助コントロール･レジスタ

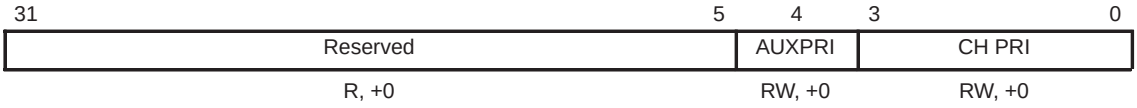


表5−9. DMA補助コントロール･レジスタのフィールド解説

フィールド	解説
CH PRI	DMAチャネル優先順位 CH PRI = 0000b: チャネル優先順位固定モード 補助チャネル優先順位1位 CH PRI = 0001b: チャネル優先順位固定モード 補助チャネル優先順位2位 CH PRI = 0010b: チャネル優先順位固定モード 補助チャネル優先順位3位 CH PRI = 0011b: チャネル優先順位固定モード 補助チャネル優先順位4位 CH PRI = 0100b: チャネル優先順位固定モード 補助チャネル優先順位5位 CH PRI = その他、予約
AUXPRI	補助チャネル優先順位モード AUXPRI = 0: CPU 優先 AUXPRI = 1: DMA 優先

DMAチャネルに指定された優先順位により、2つ以上のチャネルが転送の準備が完了した状態で、最初にリード転送またはライト転送を行なうべきDMAチャネルが決定されます。

補助チャネルの優先順位は、DMA補助コントロール･レジスタのCHI PRIフィールドを設定することにより設定可能です。デフォルトでは、CH PRIは、リセット時に0000bに設定されます。この値により、補助チャネルが最も高い優先順位となり、チャネル0、チャネル1、チャネル2がこれに続き、チャネル3 が最も低い優先順位となります。

チャネル間のアービトレーションは、各CPUクロック･サイクルで、リード転送とライト転送のそれぞれに対して独立に発生します。何らかの同期に対してウェイト状態にあるチャネルでは、DMAの制御をより優先順位の低いチャネルに渡さなければならないことがあります。同期が受け付けられれば、そのチャネルは、制御を渡した優先順位の低いチャネルから制御を取り戻すことができます。このルールは、スプリット･モード転送の送信と受信のそれぞれの部分に個別に適用されます。送信は、受信より高い優先順位を持っています。

複数のDMAチャネルとCPUが同じリソースに関して競合を生じている場合には、DMAチャネル間でのアービトレーションが最初に発生します。次に、最高の優先順位を持ったDMAチャネルとCPUとの間でのアービトレーションが実行されます。通常は、あるチャネルがCPUよりも低い優先順位を持っている場合には、より優先順位の低いチャネルは、すべて、CPUよりも低い優先順位を持つことになります。同様にして、あるチャネルがCPUより高い

優先順位を持っていれば、より高い優先順位を持ったチャネルは、CPUよりも高い優先順位を持つことになります。

このようなDMAとCPUのアービトレーションは、競合しているリソースによって実行されます。詳細については、それぞれのリソースのドキュメントを参照してください。チャネルのPRIフィールドは、そのチャネルがポーズされるかストップされた時にのみ更新されるべきことに注意してください。

5.9.2 チャネルの切り替え

優先順位の高いチャネルは、必要なリード同期を受け取ると、優先順位の低いチャネルからDMAの制御を取り戻します。チャネルの切り替え時には、現行のチャネルには、リクエストされたリードがすべて完了することが許されます。DMAは、優先順位の高いチャネルの中から、DMAの制御を与えるチャネルを決定します。次いで、制御を与えられたチャネルがリード動作を開始します。これと同時に、もとのチャネルからのライト転送が許可されます。

5.10 DMAチャネル状態の判定

重要なイベントやDMAのチャネル動作における潜在的な問題についての情報を知るために、いくつかのフラグが用意されています。このフラグはDMAチャネル･セカンダリ･コントロール･レジスタにあります。

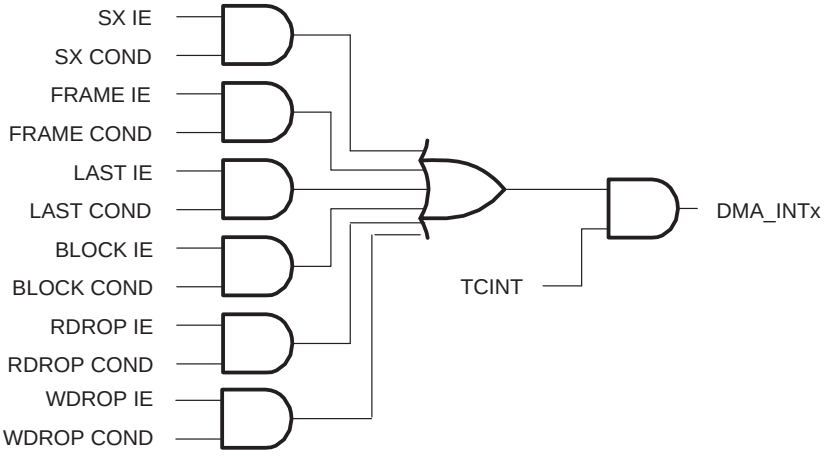
DMAチャネル･セカンダリ･コントロール･レジスタには、割り込みイネーブル･ビット(IE)フィールドが存在し、それらによってDMAチャネルからCPUに割り込みを発生させることができます。CONDビットと対応するIEビットがセットされると、その状態が対応するDMAチャネルからCPUへの割り込み信号に反映されます。DMAチャネルxのプライマリ･コントロール･レジスタのTCINTビットがセットされると、IEビットによってイネーブルされたすべての状態の論理ORがDMA_INT*x*信号となります。これ以外の場合には、DMA_INT*x*は、アクティブとなりません。この論理回路を図5−13に示します。割り込みセレクタにより選択されていると、DMA_INTのローからハイへの遷移により、割り込み状態がCPUにラッチされます。

DMAチャネル･セカンダリ･コントロール･レジスタのSX COND、WDROP COND、RDROP CONDの各ビットは、警告状態として取扱われます。これらの状態がイネーブルされており、アクティブであれば、TCINTビットの内容にかかわらず、対応するDMAチャネルを動作状態からポーズ状態に移行させます。

CONDビットに対応するIEビットがセットされていると、ユーザによる0のライトによってのみ、CONDビットをクリアすることができます。それ以外の場合には、CONDビットは、自動的にクリアされます。ユーザのCONDビットへの1のライトは、無視されます。このため、状態をマニュアル操作によって強制的に変更することはできません。

このレジスタのほとんどの値は、リセットによってクリアされます。ただ1つの例外としては、ブロック転送完了イベント(BLOCK IE)の割り込みイネーブルがあり、これは、リセット時にセットされます。つまりデフォルトでは、ブロック転送完了状態が、CPUに割り込みを生じさせる唯一の状態となっています。他の状態は、対応するIEビットをセットすることによりイネーブルすることができます。

図5−13. 状態に応じたチャネルxへのDMA割り込みの生成



5.10.1 チャネル状態の定義

表5−10に、DMAチャネル･セカンダリ･コントロール･レジスタの各状態フラグを示します。

システム･アプリケーションによっては、これらの状態は、システムのエラーを示します。ラスト･フレーム状態は、自動初期化モード時に、リロード･レジスタの値を変更したいときに使用できます。フレーム･インデックスとエレメント･カウント･リロードは、どのフレームにおいても使用されます。このため、これらの値の変更は、ブロック転送の中の最後から2番めのフレームの転送が完了した後に行なう必要があります。このようなタイミングで変更を行わないと、現行のブロック転送に影響を与えることとなります。

表5−10. DMAチャネル状態の解説

ビット フィールド	イベント	発生条件	状態のクリア条件	
			IEがイネーブル	その他
SX	スプリット送信オーバラン受信	スプリット動作がイネーブルされていて、送信エレメント転送が受信エレメント転送に対して7エレメント以上先行した場合	ユーザによるCONDへの0のライト	
FRAME	フレーム完了	各フレームの最後のライト転送がメモリーにライトされた後	ユーザによるCONDへの0のライト	2CPUクロック後
LAST	ラスト･フレーム	ブロック転送の最後のフレームに移るためのカウンタ調整が完了した後	ユーザによるCONDへの0のライト	2CPUクロック後
WDROP RDROP	リード/ライト同期喪失	最後の同期イベントがクリアされる前に次の同期イベントが発生	ユーザによるCONDへの0のライト	
BLOCK	ブロック転送完了	ブロック転送の最後のライト転送がメモリーにライトされた後	ユーザによるCONDへの0のライト	2CPUクロック後

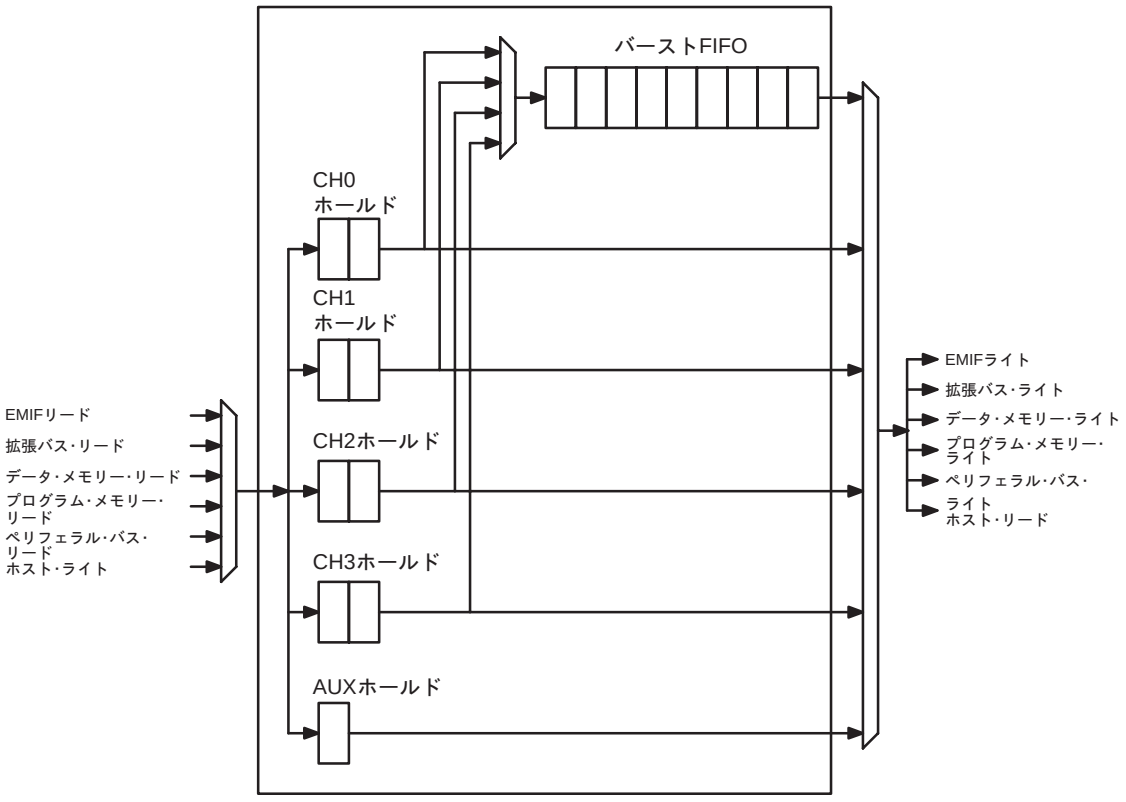
5.11 DMAの制御構造

TMS320C6000プラットフォームのDMAは、ユーザがプログラム可能なチャネルと補助チャネルから成ります。各チャネルは、高速メモリーへのバースト動作が可能で、ペリフェラルに対してスプリット･モードで使用することもできます。C6201/C6701/C6202のDMAは、各チャネルのホールド･レジスタのペアと9層の共有のFIFOによって、この動作を実行します。C6203では少し改良が加えられ、特定の場合に、より高いパフォーマンスを実現します。また、共有FIFOではなく、それぞれのチャネルが専用の9層のFIFOを持っています。

5.11.1 TMS320C6201/C6701/C6202 DMAの構造

図5−14に、データ･バスと内部ホールド･レジスタを含むC6201/C6701/C6202 DMAの内部データ移動パスを示します。

図5−14. TMS320C6201/C6701/C6202 DMAデータ･バス･プロック図



5.11.1.1 リード及びライト･バス

各DMAチャネルは、次の5つのソースとデスティネーションから1つを選択することができます。

1) EMIF
2) 拡張バス
3) 内部プログラム･メモリー
4) 内部データ･メモリー
5) 内部ペリフェラル･バス

各ソースからのリード及びライト･バスは、DMAとインターフェイスされています。

補助チャネルも、リード及びライト･バスを持っています。しかし、補助チャネルはDMA転送に対してアドレス生成することができるので、その命名の方法は異なっています。例えば、補助チャネルからDMAを介してのデータのライトは、補助ライト･バスにより行われます。同じように、補助チャネルからDMAを介してのデータのリードは、補助リード･バスにより行われます。

5.11.1.2 共有FIFO

内部プログラム･メモリー、内部データ･メモリー、外部シンクロナスDRAM(SDRAM)、それにシンクロナス･バーストSRAM(SBSRAM)などの高速メモリーに対するバースト動作を容易に行なうために、9段のDMA FIFOホールド･パスが設けられています。チャネルのホールド･レジスタとこのパスを組み合わせることにより、11段のFIFOを構成することもできます。各時点においてFIFOを制御できるのは1チャネルだけです。チャネルがFIFOの制御を獲得するには、次の条件が満たされる必要があります。

- ❑ チャネルにおいて、リード同期、ライト同期がイネーブルされていないこと。スプリット･モードでは、リードとライトの同期が必要とされるため、FIFOは、スプリット･モードのチャネルでは使用できません。フレーム同期だけがイネーブルされている場合には、FIFOをそのチャネルで使用することができます。
- ❑ チャネルが動作中であること。
- ❑ FIFOが、他のチャネルからのデータを持っていないこと。
- ❑ 上記3つの条件を満たすチャネルのうち、最も優先順位の高いチャネルであること。

3番目の条件により、”ヘッドオブライン”ブロッキングの可能性が最小限に抑えられます。ヘッドオブライン･ブロッキングは、より高い優先順位を持ったDMAチャネルが最初のリクエストを発行する前に、より低い優先順位を持ったチャネルのリクエストが完了するのを待っている時に起こります。より優先順位の高いチャネルが低い優先順位のチャネルにDMAの制御権を要求した場合に、前のチャネルの最後のリクエストのみが完了される必要があります。その後、その優先順位の高いチャネルは、ホールド･レジスタを通じて自分のリクエストを完了します。ホールド･レジスタのスループットは、DMAのスループットよりも低いものとなります。このギャップのため、優先順位の低いチャネルは新しいリード転送は開始しませんが、ライト転送の完了のためのFIFOの掃き出しを許されます。高い優先順位のチャネルがFIFOの制御権を得ていないため、優先順位の低いチャネルが転送データをFIFOから掃き出すことが可能になります。FIFOがクリアされれば、優先順位の高いチャネルは、ストップしていない限り、FIFOの制御権を得ることができます。

DMAのFIFOには、次の2つの目的があります。

- ❑ パフォーマンスの向上
- ❑ アービトレーション･レイテンシの減少

パフォーマンス向上のために、FIFOはライト転送に先立ってリード転送を行ないます。この機能により、エレメント転送の際のソース側及びデスティネーション側のスループットの違いを吸収します。このため、DMAは、エレメント転送のリードとライトを、別々の機会で行なうことができます。リクエストしているDMAチャネルがFIFOを使用していれば、CPUクロック･サイクルのレートでのリードまたはライト･アクセスを維持することができます。しかし、最初のアクセスを実行するまでに、いくらかのレイテンシがあります。リソースとDMAの間の関係により、連続するリクエストのレートと受信リード転送データのレイテンシが左右されます。

DMA FIFOのその他の機能として、特定のリソースに対するペンディングされたリクエストからのリード･データの捕獲があります。例えば、DMAがSDRAMやSBSRAMなどのパイプライン化された外部メモリーから内部データ･メモリーにデータをリードしている場合を想定します。ここで、CPUは、リクエストを発生しているDMAチャネルよりも高い優先順位を持っており、EMIFへのプログラム･フェッチ･リクエストがコンフリクトを起こしているとします。また、CPUが内部メモリーのすべてのバンクにアクセスし、DMAを締め出したとします。この状況で、FIFOがペンディングとなっているDMAのアクセスの完了とプログラム･フェッチを可能とします。DMAのパイプライン化されたリクエスト構造により、DMAはどの時点においても、ペンディングとされたデータ待ち状態のリード転送リクエストを持つことになります。DMAは、FIFOの空白を埋めるためのリクエストを発生した後で、リード転送のリクエストを発生するのを止めることになります。

5.11.1.3 内部ホールド･レジスタ

各チャネルは、専用の内部ホールド･レジスタを持っています。あるDMAのチャネルが内部のFIFOではなくホールド･レジスタを通じてデータを転送しているときには、リード転送のリクエストが連続的に発行されることになります。DMAがスプリット･モードであるかどうかによって、さらに以下の制約が生じます。

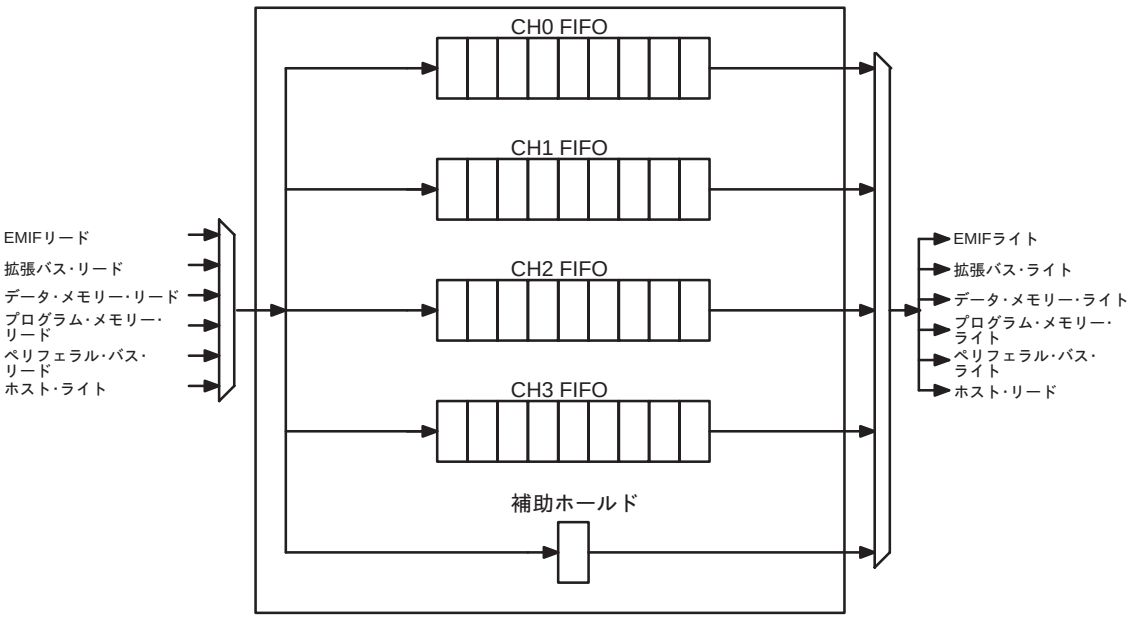
スプリットモードでは、2つのレジスタは、送信用と受信用のデータ･ストリーム･ホールド･レジスタとして別々に機能します。送信と受信のリード転送においては、対応するライト転送リクエストが完了するまで、次のリード転送リクエストは発生しません。

ノン･スプリット･モードでは、データが到達すると、対応するライト転送の完了を待つことなく、次のリード転送が発行されることがあります。しかし、ホールド･レジスタは2つしかないため、リード転送は、ライト転送に対して1つしか先行することができません。

5.11.2 TMS320C6203 DMAの構造

C6203DMAでは、パフォーマンス向上のため、C6201DMAの構造を改良しています。単一のバーストFIFOのアービトレーションを省くことで、スループットのロスなく複数のバースト･チャネルの共存が可能になります。図5−15に、データ･バスと内部ホールド･レジスタを含むC6203の内部データ移動パスを示します。

図5−15. TMS320C6203 DMAデータ･バス･ブロック図



5.11.2.1 リード及びライト･バス

C6201/C6701/C6202 DMAと同様に、各DMAチャネルは、次の5つのソースとデスティネーションから1つを選択することができます。

1) EMIF
2) 拡張バス
3) 内部プログラム･メモリー
4) 内部データ･メモリー
5) 内部ペリフェラル･バス

補助チャネルのリード及びライト･バスは、図5−15に示した通りです。

5.11.2.2 チャネルFIFO

C6203 の各DMAチャネルは、専用の9層のFIFOを持っており、高速メモリーへのバースト動作が可能です。各チャネルが専用のFIFOを持つことで、高速なバースト･チャネル間の切り替えに必要なアービトレーションを減らすことができます。どのチャネルも、単体での動作は、C6201/C6701/C6202 DMAと変わりません。複数のFIFOは、チャネル間の切り替えがあるときに有効です。

専用のFIFOにより、バースト動作しているあるDMAチャネルから他のDMAチャネルへシームレスに切り替えることが可能です。C6201/C6701/C6202DMAは、優先順位の低いチャネルがペンディングされた全てのライト転送を終了するより先に、優先順位が高いチャネルを始めることができるので、より優先順位の高いチャネルが、FIFOのアクセス権を得られない可能性があります。優先順位の高いチャネルのソースが、優先順位の低いチャネルのデスティネーションと同じとき、FIFOはデータを掃き出すことができません。DMAの優先順位は、どちらのチャネルがリソースにアクセスできるか、を決定するために、DMAチャネルの番号を考慮します。優先順位の高いチャネルは共通のソースを持つので、優先順位の低いチャネルはペンディングされたライトとして、そこにアクセスすることができません。データは共有FIFOに残り、結果として、優先順位の高いチャネルはFIFOを使うことができません。

このような場合、優先順位の高いチャネルがFIFOを使用できないので、優先順位の高いチャネルが割り込んでバースと動作をしようとしても、正しく動作しません。代わりに、優先順位の高いチャネルは、そのチャネルのホールド･レジスタを2層のFIFOとして使用しますが、これはバースト動作には十分でありません。連続的なデータのバースト転送ではなく、1度に2エレメントのみ転送されます。

C6203の新しいDMAの構造は、C6201 DMAのこのような限定された特徴を改善しています。各チャネルに専用のFIFOを持たせることで、優先順位の高いチャネルの高速バースト動作を実現するために、優先順位の低いチャネルがペンディングされたデータを掃き出す必要がなくなります。優先順位の高いチャネルが終了するまで、優先順位の低いチャネルは、データをそのチャネルのFIFOに保存しておきます。優先順位の低いチャネルは、そのデスティネーション･リソースへのアクセス権を得ると、転送を再開します。

その他の状況では、C6203 DMAチャネルの動作は、C6201/C6701/C6202 DMAと同じです。

5.11.2.3 スプリット･モード

スプリット･モードでは、C6203のDMAチャネルは、C6201/C6701/C6202のDMAチャネルと同様に動作します。チャネルFIFOの最初の2つのセルのみが使用され、2つのホールド･レジスタとなります。各セルはそれぞれ送信及び受信データ･ストリームのためのホールド･レジスタとして機能します。送信及び受信リード転送の両方において、対応する転送リクエストが完了するまで、次のリード転送リクエストは発生しません。

5.11.3 DMAパフォーマンス

DMAは、リード転送とライト転送のために異なるリソースをアクセスしており、これらのリソースの双方がシングル･サイクルのスループットを持っている場合には、シングル･サイクルのスループットで実行します。例えば、シングル･サイクルの外部SBSRAMから内部データ･メモリーへの非同期のブロック転送が他のチャネルやCPUとの競合なしに行われる場合が該当します。DMAのパフォーマンスは、次の要素によって制限されます。

- ❑ リクエストするリソースのスループットとレイテンシ
- ❑ リード、ライト、またはフレーム同期へのウェイト
- ❑ 優先順位の高いチャネルからの割り込み
- ❑ リソースに対するCPUとのコンフリクト

5.12 DMA動作完了表示ピン

DMA動作完了表示ピン(DMAC0−DMAC3)により、各チャネルから外部ロジックに対してフィードバックを与えることができます。DMAチャネル･セカンダリ･コントロール･レジスタのDMAC ENビットフィールドの指定により、このピンにRSYNC STAT、WSYNC STAT、BLOCK COND、またはFRAME CONDの状態を出力させることができ、また、汎用の出力ピンとしても使用することができます。DMACビットが、RSYNC STATまたはWSYNC STATを反映する場合には、同期イベントの認識により、DMACは、ローからハイに遷移します。イベントのサービスが実行され、状態ビットがクリアされると、DMACは、ハイからローへ遷移します。DMAC信号は、チップから送出される際にCLKOUT1信号と同期されます。これらの信号がアクティブとなる期間は、最低、CLKOUT1の２サイクル分であることが保証されています。

5.13 エミュレーション

エミュレータをデバッグに使用する際には、CPUを実行パケットのバウンダリで停止させて、シングル･ステップ、ベンチマーク、プロファイリング、その他のデバッグの目的に使用することができます。ユーザは、このときに、DMAがポーズするか、動作を継続するかを設定することができます。この設定は、DMAプライマリ･コントロール･レジスタのEMODビットの値を0か1にすることによって選択することができます。ポーズが選択されると、STATUSフィールドに、チャネルのポーズ状態が反映されます。補助チャネルは、エミュレーションによるCPU停止の間も動作を続けます。このエミュレーションは、DMAのシングルステップ転送をおおまかにシミュレートするものです。EMOD=1の設定がされているDMAチャネルでは、シングル･ステップの間に複数の転送を行なうことがありますが、これは、ステップを成功させるために、実行されていない複数の転送を完了させることが必要となる場合があるためです。

ご注意

日本テキサス・インスツルメンツ株式会社（以下TIJといいます）及びTexas Instruments Incorporated（TIJの親会社、以下TIJないしTexas Instruments Incorporatedを総称してTIといいます）は、その製品及びサービスを任意に修正し、改善、改良、その他の変更をし、もしくは製品の製造中止またはサービスの提供を中止する権利を留保します。従いまして、お客様は、発注される前に、関連する最新の情報を取得して頂き、その情報が現在有効かつ完全なものであるかどうかご確認下さい。全ての製品は、お客様とTIJとの間に取引契約が締結されている場合は、当該契約条件に基づき、また当該取引契約が締結されていない場合は、ご注文の受諾の際に提示されるTIJの標準販売契約約款に従って販売されます。

TIは、そのハードウェア製品が、TIの標準保証条件に従い販売時の仕様に対応した性能を有していること、またはお客様とTIJとの間で合意された保証条件に従い合意された仕様に対応した性能を有していることを保証します。検査およびその他の品質管理技法は、TIが当該保証を支援するのに必要とみなす範囲で行なわれております。各デバイスの全てのパラメーターに関する固有の検査は、政府がそれ等の実行を義務づけている場合を除き、必ずしも行なわれておりません。

TIは、製品のアプリケーションに関する支援もしくはお客様の製品の設計について責任を負うことはありません。TI製部品を使用しているお客様の製品及びそのアプリケーションについての責任はお客様にあります。TI製部品を使用したお客様の製品及びアプリケーションについて想定されうる危険を最小のものとするため、適切な設計上および操作上の安全対策は、必ずお客様にてお取り下さい。

TIは、TIの製品もしくはサービスが使用されている組み合せ、機械装置、もしくは方法に関連しているTIの特許権、著作権、回路配置利用権、その他のTIの知的財産権に基づいて何らかのライセンスを許諾するということは明示的にも黙示的にも保証も表明もしておりません。TIが第三者の製品もしくはサービスについて情報を提供することは、TIが当該製品もしくはサービスを使用することについてライセンスを与えるとか、保証もしくは是認するということを意味しません。そのような情報を使用するには第三者の特許その他の知的財産権に基づき当該第三者からライセンスを得なければならない場合もあり、またTIの特許その他の知的財産権に基づきTI からライセンスを得て頂かなければならない場合もあります。

TIのデータ・ブックもしくはデータ・シートの中にある情報を複製することは、その情報に一切の変更を加えること無く、かつその情報と結び付られた全ての保証、条件、制限及び通知と共に複製がなされる限りにおいて許されるものとします。当該情報に変更を加えて複製することは不公正で誤認を生じさせる行為です。TIは、そのような変更された情報や複製については何の義務も責任も負いません。

TIの製品もしくはサービスについてTIにより示された数値、特性、条件その他のパラメーターと異なる、あるいは、それを超えてなされた説明で当該TI製品もしくはサービスを再販売することは、当該TI製品もしくはサービスに対する全ての明示的保証、及び何らかの黙示的保証を無効にし、かつ不公正で誤認を生じさせる行為です。TIは、そのような説明については何の義務も責任もありません。

TIは、TIの製品が、安全でないことが致命的となる用途ないしアプリケーション（例えば、生命維持装置のように、TI製品に不良があった場合に、その不良により相当な確率で死傷等の重篤な事故が発生するようなもの）に使用されることを認めておりません。但し、お客様とTIの双方の権限有る役員が書面でそのような使用について明確に合意した場合は除きます。たとえTIがアプリケーションに関連した情報やサポートを提供したとしても、お客様は、そのようなアプリケーションの安全面及び規制面から見た諸問題を解決するために必要とされる専門的知識及び技術を持ち、かつ、お客様の製品について、またTI製品をそのような安全でないことが致命的となる用途に使用することについて、お客様が全ての法的責任、規制を遵守する責任、及び安全に関する要求事項を満足させる責任を負っていることを認め、かつそのことに同意します。さらに、もし万一、TIの製品がそのような安全でないことが致命的となる用途に使用されたことによって損害が発生し、TIないしその代表者がその損害を賠償した場合は、お客様がTIないしその代表者にその全額の補償をするものとします。

TI製品は、軍事的用途もしくは宇宙航空アプリケーションないし軍事的環境、航空宇宙環境にて使用されるようには設計もされていませんし、使用されることを意図されておりません。但し、当該TI製品が、軍需対応グレード品、若しくは「強化プラスティック」製品としてTIが特別に指定した製品である場合は除きます。TIが軍需対応グレード品として指定した製品のみが軍需品の仕様書に合致いたします。お客様は、TIが軍需対応グレード品として指定していない製品を、軍事的用途もしくは軍事的環境下で使用することは、もっぱらお客様の危険負担においてなされるということ、及び、お客様がもっぱら責任をもって、そのような使用に関して必要とされる全ての法的要求事項及び規制上の要求事項を満足させなければならないことを認め、かつ同意します。

TI製品は、自動車用アプリケーションないし自動車の環境において使用されるようには設計されていませんし、また使用されることを意図されておりません。但し、TIがISO/TS 16949の要求事項を満たしていると特別に指定したTI製品は除きます。お客様は、お客様が当該TI指定品以外のTI製品を自動車用アプリケーションに使用しても、TIは当該要求事項を満たしていなかったことについて、いかなる責任も負わないことを認め、かつ同意します。

弊社半導体製品の取り扱い・保管について

半導体製品は、取り扱い、保管・輸送環境、基板実装条件によっては、お客様での実装前後に破壊/劣化、または故障を起こすことがあります。

弊社半導体製品のお取り扱い、ご使用にあたっては下記の点を遵守して下さい。

1. 静電気
   - 素手で半導体製品単体を触らないこと。どうしても触る必要がある場合は、リストストラップ等で人体からアースをとり、導電性手袋等をして取り扱うこと。
   - 弊社出荷梱包単位（外装から取り出された内装及び個装）又は製品単品で取り扱いを行う場合は、接地された導電性のテーブル上で（導電性マットにアースをとったもの等）、アースをした作業者が行うこと。また、コンテナ等も、導電性のものを使うこと。
   - マウンタやはんだ付け設備等、半導体の実装に関わる全ての装置類は、静電気の帯電を防止する措置を施すこと。
   - 前記のリストストラップ・導電性手袋・テーブル表面及び実装装置類の接地等の静電気帯電防止措置は、常に管理されその機能が確認されていること。
2. 温・湿度環境
   - 温度：0～40℃、相対湿度：40～85%で保管・輸送及び取り扱いを行うこと。（但し、結露しないこと。）
   - 直射日光があたる状態で保管・輸送しないこと。
3. 防湿梱包
   - 防湿梱包品は、開封後は個別推奨保管環境及び期間に従い基板実装すること。
4. 機械的衝撃
   - 梱包品（外装、内装、個装）及び製品単品を落下させたり、衝撃を与えないこと。
5. 熱衝撃
   - はんだ付け時は、最低限260℃以上の高温状態に、10秒以上さらさないこと。（個別推奨条件がある時はそれに従うこと。）
6. 汚染
   - はんだ付け性を損なう、又はアルミ配線腐食の原因となるような汚染物質（硫黄、塩素等ハロゲン）のある環境で保管・輸送しないこと。
   - はんだ付け後は十分にフラックスの洗浄を行うこと。（不純物含有率が一定以下に保証された無洗浄タイプのフラックスは除く。）

以上