注意 : この日本語版文書は参考資料としてご利用ください。最新情報は必ずオリジナルの英語版をご参照願います。

セクション 38. ダイレクト メモリアクセス (DMA) (パート III)

ハイライト

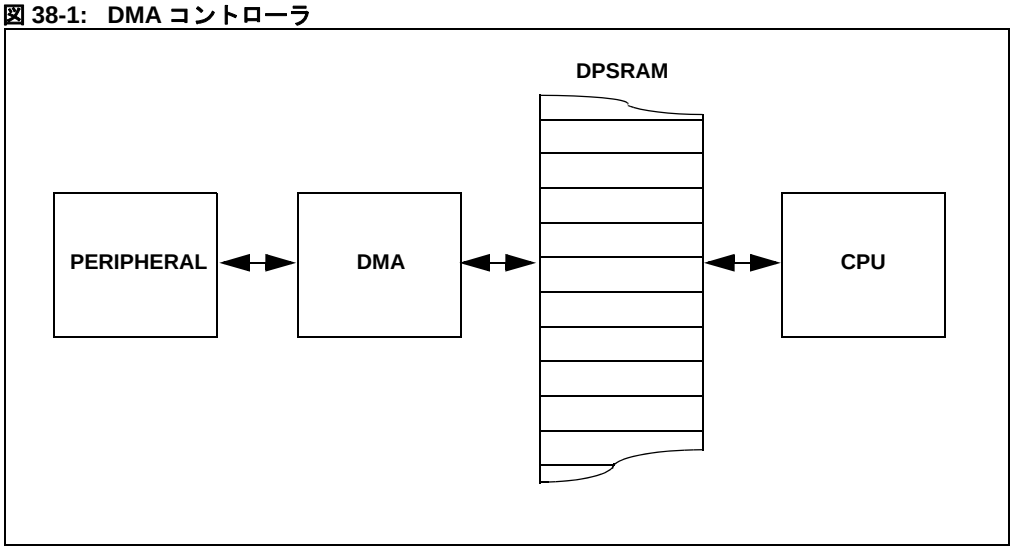
本セクションには下記の主要項目を記載しています。

38.1 はじめに

ダイレクト メモリ アクセス (DMA) コントローラは、マイクロチップ社の高性能 16 ビット デジタル シグナル コントローラ (DSC) ファミリにおける重要なサブシステムです。このサブシステムにより、CPU 時間を消費せずメモリと周辺モジュール間でデータ転送を行えます。dsPIC33F DMA コントローラは、決定論的機能とシステム レイテンシが重視される、高性能、リアルタイム、組み込みアプリケーション向けに最適化されています。

DMA コントローラは、周辺モジュール用データ レジスタとデータ領域 SRAM 間でデータを転送します。dsPIC33F DMA サブシステムはデュアルポート SRAM メモリ (DPSRAM) とレジスタ構造を採用し、CPU に負荷をかけずに独立したアドレスバスとデータバスで DMA を動作させる事が可能です。このアーキテクチャにより、優先度の高い DMA 転送が要求された際に CPU を停止させる事になるサイクル スチールが不要になります。CPU と DMA コントローラは共に、CPU ストール等の影響を受けずにデータ空間内のアドレスに対して読み書きできます。このため、リアルタイム性能を最大限に高められます。一方、DMA 動作と、メモリと周辺モジュール間でのデータ転送は CPU 処理の影響を受けません。例えば、実行時自己プログラミング (RTSP) 実行中は、RTSP が終了するまで CPU は一切の命令を実行しません。しかし、この状態でもメモリと周辺モジュール間のデータ転送に影響はありません。

図 38-1: DMA コントローラ



DMA コントローラは独立した 8 チャンネルをサポートします。各チャンネルは、選択された周辺モジュールとの単方向データ転送用に設定できます。DMA コントローラがサポートする周辺モジュールには下記が含まれます。

- 拡張コントローラ エリア ネットワーク (ECAN™) テクノロジ
- データコンバータ インターフェイス (DCI)
- 10/12 ビット A/D コンバータ (ADC)
- シリアル ペリフェラル インターフェイス (SPI)
- UART (Universal Asynchronous Receiver Transmitter)
- 入力キャプチャ
- 出力コンペア
- D/A コンバータ (DAC)
- パラレル マスタポート (PMP)

さらに、DMA 転送は、タイマまたは外部割り込みを使用して開始できます。

各 DMA チャンネルは単方向です。周辺モジュールとの間で読み出しと書き込みの双方向の転送を行う場合、2 つの DMA チャンネルを割り当てる必要があります。複数のチャンネルがデータ転送要求を受け取った場合、チャンネル番号順の固定的な優先度に従って、転送を実行するチャンネルと保留されるチャンネルが決まります。各 DMA チャンネルは、最大 1024 個のデータ要素を含むデータブロックを移動し、移動完了時に CPU に対して割り込みを生成して、そのデータブロックが処理可能である事を知らせます。

DMA コントローラは下記の機能を備えます。

- 8 x DMA チャンネル
- ポストインクリメント アドレッシング モードを使用するレジスタ間接
- ポストインクリメント アドレッシング モードを使用しないレジスタ間接
- 周辺モジュール間接アドレッシング モード (周辺モジュール側で転送先アドレスを生成)
- ハーフブロックまたはフルブロック転送完了時の CPU 割り込み
- バイト転送またはワード転送
- 固定優先度に従うチャンネル間調停
- 手動 (ソフトウェア) または自動 (周辺モジュール DMA 要求) による転送開始
- ワンショットまたは自動再送ブロック転送モード
- ピンポンモード(2つのDPSRAM 開始アドレスをブロック転送ごとに交互に自動切り換え)
- 各チャンネルに対する DMA 要求は、サポートされる任意の割り込み要因から選択可能
- デバッグサポート機能

38.2 DMA レジスタ

各 DMA チャンネルには、それぞれ 6 個のステータスおよび制御レジスタが割り当てられています。

- DMAxCON: DMA チャンネル x 制御レジスタ

  このレジスタでは、DMA チャンネル x の有効化 / 無効化、データ転送サイズ / データ転送方向 / ブロック割り込み方法の設定、DAMA チャンネル アドレッシング モード / 動作モード /NULL データ書き込みモードの選択を行います。

- DMAxREQ: DMA チャンネル x IRQ 選択レジスタ

  このレジスタでは、DMA チャンネル x に周辺モジュール IRQ を割り当てる事によって、その DMA チャンネルを特定の DMA 対応周辺モジュールに関連付けます。

- DMAxSTA: DMA チャンネル x DPSRAM 開始アドレス オフセット レジスタ A

  このレジスタでは、DMA チャンネル x と DPSRAM 間で転送するデータブロックの、DMA DPSRAM ベースアドレスからのプライマリ開始アドレスオフセットを指定します。このレジスタを読み出すと、直前の DPSRAM 転送のアドレスオフセット値が返されます。チャンネル x が有効 (アクティブ) な時にこのレジスタが書き込まれると、予測不能な挙動が生じる可能性があるため、書き込みを回避する必要があります。

- DMAxSTB: DMA チャンネル x DPSRAM 開始アドレス オフセット レジスタ B

  このレジスタでは、DMA チャンネル x と DPSRAM 間で転送するデータブロックの、DMA DPSRAM ベースアドレスからのセカンダリ開始アドレスオフセットを指定します。このレジスタを読み出すと、直前の DPSRAM 転送のアドレスオフセット値が返されます。チャンネル x が有効 (アクティブ) な時にこのレジスタが書き込まれると、予測不能な挙動が生じる可能性があるため、書き込みを回避する必要があります。

- DMAxPAD: DMA チャンネル x 周辺モジュール アドレス レジスタ

  この読み書き可能レジスタは、周辺モジュール データレジスタの静的アドレスを格納します。対応する DMA チャンネルが有効 (アクティブ) な時にこのレジスタが書き込まれると、予測不能な挙動が生じる可能性があるため、書き込みを回避する必要があります。

- DMAxCNT: DMA チャンネル x 転送カウントレジスタ

  このレジスタは転送数を格納します。そのチャンネルが DMAxCNT + 1 個の DMA 要求を処理すると、データブロック転送が完了したとみなされます。DMAxCNT の値が「0」の場合、1 個のデータ要素が転送されます。DMAxCNT レジスタの値は、データ転送サイズ (DMAxCON レジスタの SIZE ビット) とは無関係です。対応する DMA チャンネルが有効 (アクティブ) な時にこのレジスタが書き込まれると、予測不能な挙動が生じる可能性があるため、書き込みを回避する必要があります。

上記の各 DMA チャンネル用のレジスタ以外に、DMA コントローラ用に 3 個の DMA ステータス レジスタが割り当てられています。

- DSADR: 最新の DMA DPSRAM アドレスレジスタ

  全ての DMA チャンネルは、この 16 ビット読み出し専用ステータス レジスタを共有します。このレジスタは、直前の DPSRAM アクセス (読み / 書きのいずれか) のアドレスを格納します。このレジスタはリセット時にクリアされます。従って、リセット後に、一度も DMA を実行せずにこのレジスタを読み出すと 「0x0000」が返されます。このレジスタにはいつでもアクセスできますが、デバッグの支援がこのレジスタの本来の目的です。

- DMACS0: DMA コントローラ ステータス レジスタ 0

  この 16 ビット読み出し専用ステータス レジスタは、DPSRAM と周辺モジュールの書き込みコリジョンフラグ (それぞれ XWCOLx と PWCOLx) を格納します。詳細は **38.10「データ書き込みコリジョン」**を参照してください。

- DMACS1: DMA コントローラ ステータス レジスタ 1

  この 16 ビット読み出し専用ステータス レジスタは、直前にアクティブであった DMA チャンネルと、各 DMA チャンネルのピンポンモードのステータス (どちらの DPSRAM 開始アドレス オフセット レジスタ (DMAxSTA または DMAxSTB) が選択されているか) を示します。

レジスタ 38 - 1: DMAxCON: DMA チャンネル x 制御レジスタ

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0
CHEN	SIZE	DIR	HALF	NULLW	—	—	—
bit 15							bit 8

U-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
—	—	AMODE<1:0>		—	—	MODE<1:0>	
bit 7							bit 0

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

bit 15 **CHEN:** チャンネル イネーブルビット
1 = このチャンネルを有効にする
0 = このチャンネルを無効にする

bit 14 **SIZE:** データ転送サイズビット
1 = バイト
0 = ワード

bit 13 **DIR:** 転送方向ビット (ソース / デスティネーション バス選択)
1 = DPSRAM アドレスから読み出して周辺モジュール アドレスに書き込む
0 = 周辺モジュールアドレスから読み出して DPSRAM アドレスに書き込む

bit 12 **HALF:** ブロック転送割り込み選択ビット
1 = ハーフブロック (データの半分) を移動した時点で割り込みを生成する
0 = フルブロック (全てのデータ) を移動した時点で割り込みを生成する

bit 11 **NULLW:** NULL データ周辺モジュール書き込みモード選択ビット
1 = DPSRAM 書き込み時に周辺モジュールに NULL データを書き込む (DIR ビットも要クリア)
0 = 通常動作

bit 10-6 **未実装 :** 「0」として読み出し

bit 5-4 **AMODE<1:0>:** DMA チャンネル アドレッシング モード選択ビット
11 = 予約済み
10 = 周辺モジュール間接アドレッシング モード
01 = ポストインクリメント アドレッシング モードを使用しないレジスタ間接
00 = ポストインクリメント アドレッシング モードを使用するレジスタ間接

bit 3-2 **未実装 :** 「0」として読み出し

bit 1-0 **MODE<1:0>:** DMA チャンネル動作モード選択ビット
11 = ワンショット (ピンポンモード有効) (各 DMA RAM バッファとの間で 1 ブロックを転送)
10 = 連続 (ピンポンモード有効)
01 = ワンショット (ピンポンモード無効)
00 = 連続 (ピンポンモード無効)

レジスタ 38 - 2: DMAxREQ: DMA チャンネル x IRQ 選択レジスタ

R/S-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
FORCE(1)	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	IRQSEL<6:0>						
bit 7							bit 0

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

bit 15 **FORCE:** 強制 DMA 転送ビット (1)
1 = 1 回の DMA 転送を実行する (手動モード)
0 = DMA 要求によって自動的に DMA 転送を開始する

bit 14-7 **未実装 :**「0」として読み出し

bit 6-0 **IRQSEL<6:0>:** DMA 周辺モジュール IRQ 番号選択ビット
0000000 = INT0 – 外部割り込み 0
0000001 = IC1 – 入力キャプチャ 1
0000010 = OC1 – 出力コンペア 1
0000101 = IC2 – 入力キャプチャ 2
0000110 = OC2 – 出力コンペア 2
0000111 = TMR2 – Timer 2
0001000 = TMR3 – Timer 3
0001010 = SPI1 – 転送完了
0001011 = UART1RX – UART1 受信
0001100 = UART1TX – UART1 送信
0001101 = ADC1 – ADC1 変換完了
0011110 = UART2RX – UART2 受信
0011111 = UART2TX – UART2 送信
0100001 = SPI2 – 転送完了
0100010 = ECAN1 – RX データレディ
0101101 = PMP – PMP マスタデータ転送
0111100 = DCI – CODEC 転送完了
1000110 = ECAN1 – TX データ要求
1001110 = DAC1 – DAC1 右チャンネルデータ出力
1001111 = DAC1 – DAC1 左チャンネルデータ出力

Note 1: ユーザは FORCE ビットをクリアできません。FORCE ビットは、強制 DMA 転送が完了した時にハードウェアでクリアされます。

レジスタ 38 - 3: DMAxSTA: DMA チャンネル x DPSRAM 開始アドレス オフセット レジスタ A

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STA<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STA<7:0>							
bit 7							bit 0

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

bit 15-0 **STA<15:0>:** プライマリDPSRAM開始アドレス オフセットビット(ソースまたはデスティネーション)

レジスタ 38 - 4: DMAxSTB: DMA チャンネル x DPSRAM 開始アドレス オフセット レジスタ B

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STB<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STB<7:0>							
bit 7							bit 0

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

bit 15-0 **STB<15:0>:** セカンダリDPSRAM開始アドレス オフセットビット(ソースまたはデスティネーション)

レジスタ 38 - 5: DMAxPAD: DMA チャンネル x 周辺モジュール アドレス レジスタ

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PAD<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PAD<7:0>							
bit 7							bit 0

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

bit 15-0 **PAD<15:0>:** 周辺モジュール アドレス レジスタビット

レジスタ 38 - 6: DMAxCNT: DMA チャンネル x 転送カウントレジスタ

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	—	CNT<9:8>	
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNT<7:0>							
bit 7							bit 0

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

bit 15-10 予約済み

bit 9-0 **CNT<9:0>:** DMA 転送カウント レジスタビット

レジスタ 38 - 7: DSADR: 最新の DMA DPSRAM アドレスレジスタ

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DSADR<15:8>							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DSADR<7:0>							
bit 7							bit 0

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

bit 15-0 **DSADR<15:0>:** DMA が直前にアクセスした DMA DPSRAM アドレスを示すビット

レジスタ 38 - 8: DMACS0: DMA コントローラ ステータス レジスタ 0

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
PWCOL7	PWCOL6	PWCOL5	PWCOL4	PWCOL3	PWCOL2	PWCOL1	PWCOL0
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
XWCOL7	XWCOL6	XWCOL5	XWCOL4	XWCOL3	XWCOL2	XWCOL1	XWCOL0
bit 7							bit 0

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

bit 15 **PWCOL7:** チャンネル 7 周辺モジュール書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

bit 14 **PWCOL6:** チャンネル 6 周辺モジュール書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

bit 13 **PWCOL5:** チャンネル 5 周辺モジュール書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

bit 12 **PWCOL4:** チャンネル 4 周辺モジュール書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

bit 11 **PWCOL3:** チャンネル 3 周辺モジュール書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

bit 10 **PWCOL2:** チャンネル 2 周辺モジュール書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

bit 9 **PWCOL1:** チャンネル 1 周辺モジュール書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

bit 8 **PWCOL0:** チャンネル 0 周辺モジュール書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

bit 7 **XWCOL7:** チャンネル 7 DPSRAM 書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

bit 6 **XWCOL6:** チャンネル 6 DPSRAM 書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

bit 5 **XWCOL5:** チャンネル 5 DPSRAM 書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

bit 4 **XWCOL4:** チャンネル 4 DPSRAM 書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

bit 3 **XWCOL3:** チャンネル 3 DPSRAM 書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

レジスタ 38 - 8: DMACS0: DMA コントローラ ステータス レジスタ 0 (続き)

bit 2 **XWCOL2:** チャンネル 2 DPSRAM 書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

bit 1 **XWCOL1:** チャンネル 1 DPSRAM 書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

bit 0 **XWCOL0:** チャンネル 0 DPSRAM 書き込みコリジョン フラグビット
1 = 書き込みコリジョンを検出した
0 = 書き込みコリジョンを検出していない

レジスタ 38 - 9: DMACS1: DMA コントローラ ステータス レジスタ 1

U-0	U-0	U-0	U-0	R-1	R-1	R-1	R-1
—	—	—	—	LSTCH<3:0>			
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
PPST7	PPST6	PPST5	PPST4	PPST3	PPST2	PPST1	PPST0
bit 7							bit 0

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

bit 15-12 **未実装 :**「0」として読み出し

bit 11-8 **LSTCH<3:0>:** 直前アクティブ DMA チャンネル ビット
1111 =システムリセット後に DMA 転送は発生していない
1110-1000 = 予約済み
0111 =直前のデータ転送はチャンネル 7 で発生した
0110 =直前のデータ転送はチャンネル 6 で発生した
0101 =直前のデータ転送はチャンネル 5 で発生した
0100 =直前のデータ転送はチャンネル 4 で発生した
0011 =直前のデータ転送はチャンネル 3 で発生した
0010 =直前のデータ転送はチャンネル 2 で発生した
0001 =直前のデータ転送はチャンネル 1 で発生した
0000 =直前のデータ転送はチャンネル 0 で発生した
リセット時に「1111」に設定されます。このビットフィールドにはいつでもアクセス可能ですが、デバッグ支援を本来の用途とします。

bit 7 **PPST7:** チャンネル 7 ピンポンモード ステータスフラグ
1 = DMA7STB レジスタが選択されている
0 = DMA7STA レジスタが選択されている

bit 6 **PPST6:** チャンネル 6 ピンポンモード ステータスフラグ
1 = DMA6STB レジスタが選択されている
0 = DMA6STA レジスタが選択されている

bit 5 **PPST5:** チャンネル 5 ピンポンモード ステータスフラグ
1 = DMA5STB レジスタが選択されている
0 = DMA5STA レジスタが選択されている

bit 4 **PPST4:** チャンネル 4 ピンポンモード ステータスフラグ
1 = DMA4STB レジスタが選択されている
0 = DMA4STA レジスタが選択されている

bit 3 **PPST3:** チャンネル 3 ピンポンモード ステータスフラグ
1 = DMA3STB レジスタが選択されている
0 = DMA3STA レジスタが選択されている

bit 2 **PPST2:** チャンネル 2 ピンポンモード ステータスフラグ
1 = DMA2STB レジスタが選択されている
0 = DMA2STA レジスタが選択されている

bit 1 **PPST1:** チャンネル 1 ピンポンモード ステータスフラグ
1 = DMA1STB レジスタが選択されている
0 = DMA1STA レジスタが選択されている

bit 0 **PPST0:** チャンネル 0 ピンポンモード ステータスフラグ
1 = DMA0STB レジスタが選択されている
0 = DMA0STA レジスタが選択されている

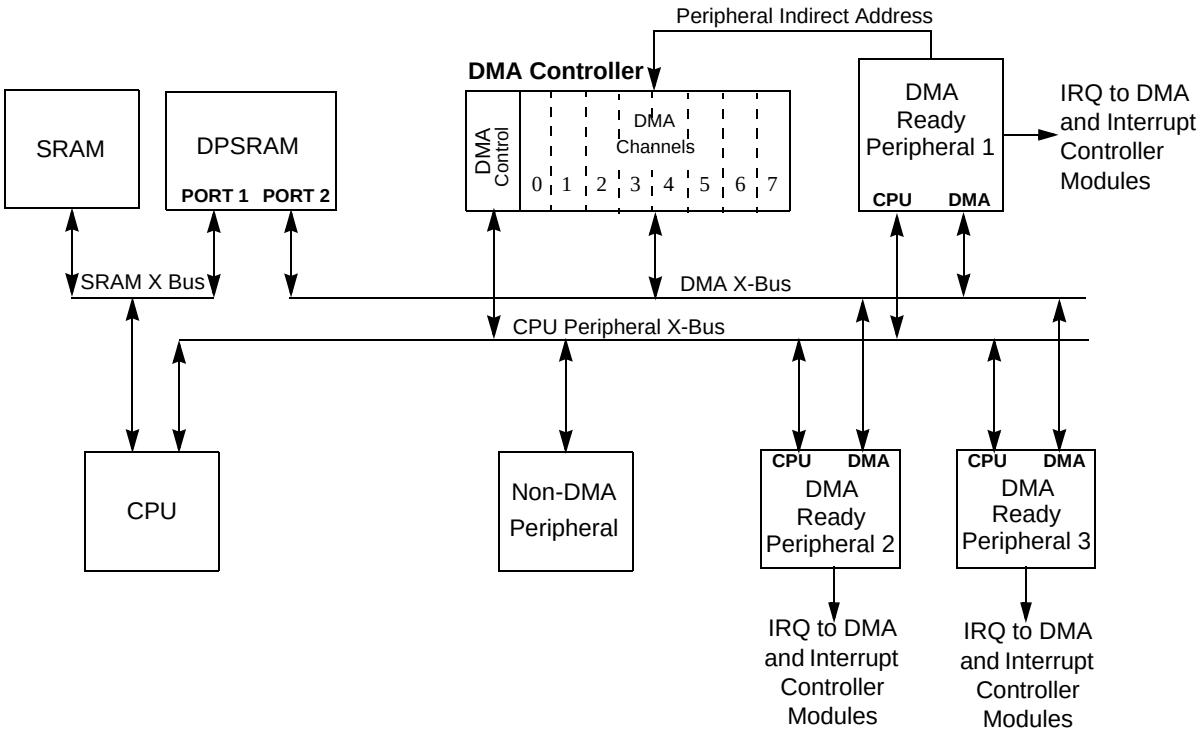
Note: このビットは読み出し専用です。

38.3 DMA のブロック図

図 38-2 は、dsPIC33F の内部アーキテクチャに DMA がどのように組み込まれているのかを示すブロック図です。CPU は、デュアルポート SRAM (DPSRAM) ブロックのポート 1 との通信に、通常の SRAM との通信に使用するのと同じ SRAM 用 X バスを使用します。周辺モジュールとの通信には、これとは別に X データ空間内に存在する周辺モジュール用 X バスを使用します。

各 DMA チャンネルは、DPSRAM の PORT 2 および各 DMA 対応周辺モジュールの DMA ポートとの通信に、専用の DMA X バスを使用します。

図 38-2: DMA コントローラのブロック図



Note: 見やすくするために、図には CPU と DMA のアドレスバスを表示していません。

他のアーキテクチャとは異なり、dsPIC33F CPU は、読み書きアクセスを単一 CPU バスサイクル内で実行できます。同様に DMA も、専用バスを使用して、単一バスサイクル内でバイトまたはワードの転送を完了できます。従って、全ての DMA 転送は実行中に割り込まれません。すなわち、あるチャンネルで開始された転送は、他チャンネルの動作に関係なく、そのサイクル中に完了します。

ユーザ アプリケーションは、任意の DMA 対応周辺モジュール割り込みを DMA 要求として指定できます。DMA 要求とは、DMA に対する IRQ の事です。当然ですが、ある DMA チャンネルを特定の割り込み (DMA 要求) に応答するように設定した場合､対応する CPU 割り込みを無効化する必要があります。これを無効化しないと、CPU 割り込みも要求されます。

ソフトウェアから手動で各 DMA チャンネルをトリガする事もできます。DMAxCON レジスタの FORCE ビットをセットすると、手動 DMA 要求が生成されます。この DMA 要求は、割り込みによる DMA 要求と同様に調停されます (**38.8「DMA チャンネルの調停と オーバーラン」**)。

38.4 DMA データ転送

図 38-3 に、周辺モジュールとデュアルポート SRAM 間のテータ転送を示します。

A. この例では、DMA チャンネル 5 を DMA 対応周辺モジュール 1 に関連付けています。
B. 周辺モジュールは、データ転送の準備を完了した時に DMA 要求を発行します。この DMA 要求は、同時に存在する他の DMA 要求との間で調停されます。調停時にこのチャンネルが最高優先度である場合、次のサイクル中に転送が完了します。その時点では優先度が他より低い場合、この DMA 要求は最高優先度となるまで保留されます。
C. この DMA チャンネルは、ユーザ アプリケーションがそのチャンネルに対して定義した周辺モジュール アドレスからデータを読み出します。
D. この DMA チャンネルは、読み出したデータを指定された DPSRAM アドレスに書き込みます。

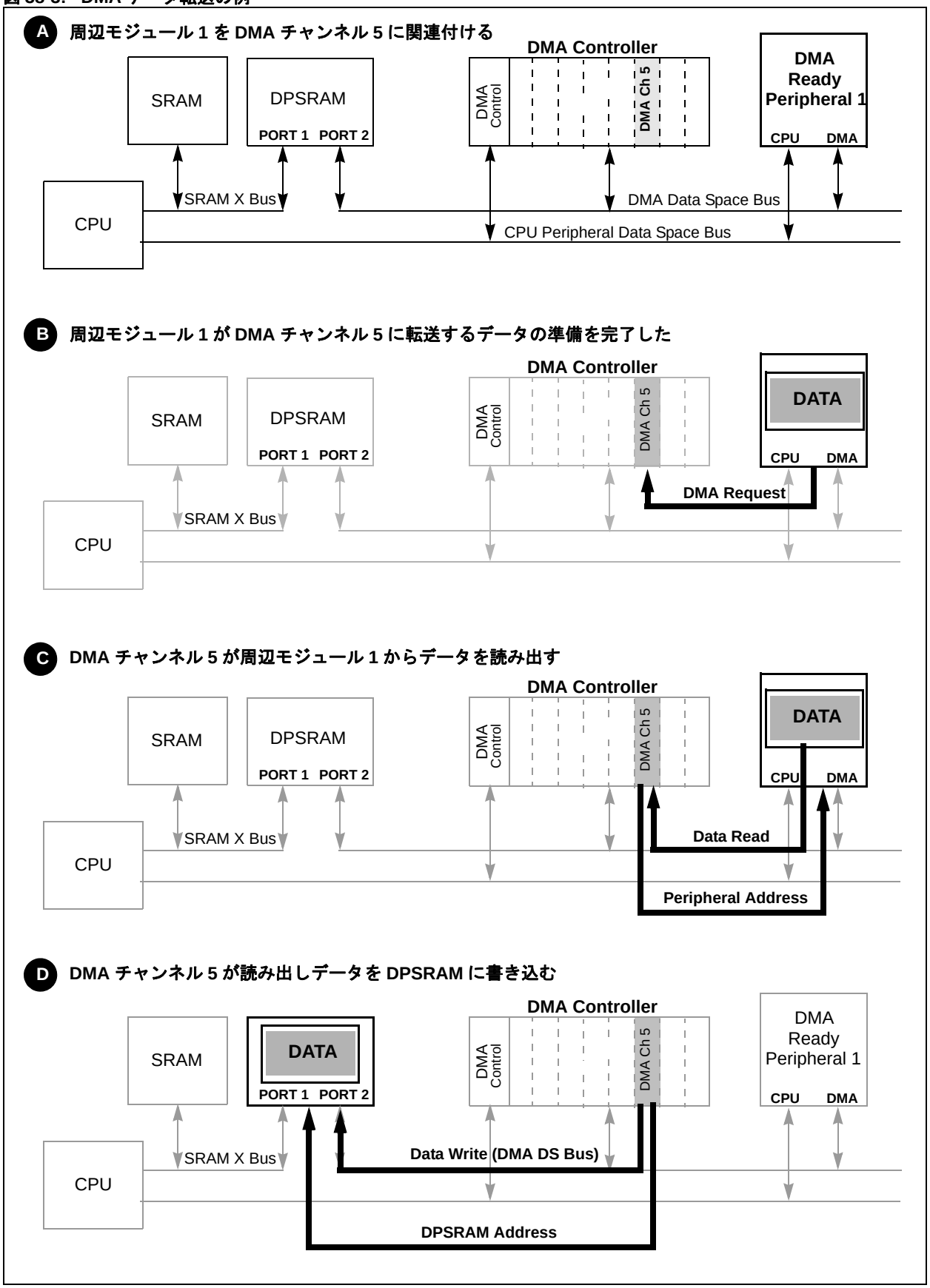
この図は、レジスタ間接モードでの動作を示しています。この場合、DPSRAM アドレスは、DMA ステータス レジスタ (DMAxSTA または DMAxSTB) によって、DMA チャンネル側で指定されます。周辺モジュール間接モードの場合、DPSRAM アドレスは DMA チャンネル側ではなく周辺モジュール側から指定されます。詳細は **38.6.6「周辺モジュール間接アドレッシング モード」**を参照してください。

DMA 読み書き転送動作は、単一命令サイクル内で割り込まれる事なく完了します。この処理中、データ転送が完了するまで DMA 要求はその DMA チャンネルにラッチされます。

この間、DMA チャンネルは転送カウンタレジスタ (DMA5CNT) を監視し、転送カウンタ値がユーザ アプリケーションによって定義されたリミット値に達すると、データ転送が完了したとみなして CPU 割り込みを生成し、受信したデータを処理するよう CPU に促します。

スループットを最大限に高めるために、DMA コントローラはデータ転送サイクル中に保留中または後続の DMA 要求の調停を続けます。

図 38-3: DMA データ転送の例



38.5 DMA のセットアップ

DMA データ転送を正しく実行するために、DMA チャンネルと周辺モジュールを下記に従って適切に設定する必要があります。

- DMA チャンネルを周辺モジュールに関連付ける (**38.5.1「DMA チャンネルと周辺モジュールの関連付け」**参照)
- 周辺モジュールを正しく設定する (**38.5.2「周辺モジュール コンフィグレーションのセットアップ」**参照)
- DPSRAM データ開始アドレスを初期化する (**38.5.3「メモリアドレスの初期化」**参照)
- DMA 転送カウンタを初期化する (**38.5.4「DMA 転送カウントのセットアップ」**参照)
- アドレッシング モードと動作モードを適切に選択する (**38.6「DMA 動作モード」**参照)

38.5.1 DMA チャンネルと周辺モジュールの関連付け

DMA チャンネルには、読み出し元または書き込み先の周辺モジュール アドレスと、転送を開始する方法を設定する必要があります。これらは、それぞれ DMA チャンネル x 周辺モジュール アドレスレジスタ (DMAxPAD) と DMA コントローラ チャンネル x IRQ 選択レジスタ (DMAxREQ) で設定します。

周辺モジュールと DMA チャンネルを関連付けるために、これらのレジスタには表 38-1 に示す値を書き込む必要があります。

表 38-1: 周辺モジュールと DMA チャンネルの関連付け

関連付ける DMA 対応周辺モジュール	DMAxREQ レジスタ IRQSEL<6:0> ビット	周辺モジュールから読み出す場合の DMAxPAD レジスタ値	周辺モジュールに書き込む場合の DMAxPAD レジスタ値
INT0 – 外部割り込み 0	0000000	—	—
IC1 – 入力キャプチャ 1	0000001	0x0140 (IC1BUF)	—
IC2 – 入力キャプチャ 2	0000101	0x0144 (IC2BUF)	—
OC1 – 出力コンペア 1 データ	0000010	—	0x0182 (OC1R)
OC1 – 出力コンペア 1 セカンダリデータ	0000010	—	0x0180 (OC1RS)
OC2 – 出力コンペア 2 データ	0000110	—	0x0188 (OC2R)
OC2 – 出力コンペア 2 セカンダリデータ	0000110	—	0x0186 (OC2RS)
TMR2 – Timer2	0000111	—	—
TMR3 – Timer3	0001000	—	—
SPI1 – 転送完了	0001010	0x0248 (SPI1BUF)	0x0248 (SPI1BUF)
SPI2 – 転送完了	0100001	0x0268 (SPI2BUF)	0x0268 (SPI2BUF)
UART1RX – UART1 受信	0001011	0x0226 (U1RXREG)	—
UART1TX – UART1 送信	0001100	—	0x0224 (U1TXREG)
UART2RX – UART2 受信	0011110	0x0236 (U2RXREG)	—
UART2TX – UART2 送信	0011111	—	0x0234 (U2TXREG)
ECAN1 – RX データレディ	0100010	0x0440 (C1RXD)	—
ECAN1 – TX データ要求	1000110	—	0x0442 (C1TXD)
DCI – CODEC 転送完了	0111100	0x0290 (RXBUF0)	0x0298 (TXBUF0)
ADC1 – ADC1 変換完了	0001101	0x0300 (ADC1BUF0)	—
PMP – PMP マスタデータ転送	0101101	0x0608 (PMDIN1)	0x0608 (PMDIN1)
DAC1 – DAC1 右データ転送	1001110	—	0x03F6 (DAC1RDAT)
DAC1 – DAC1 左データ転送	1001111	—	0x03F8 (DAC1LDAT)

2 つの DMA チャンネルに対して同じ周辺モジュールを DMA 要求源として割り当てた場合、両方のチャンネルが同時に DMA 要求を受け取ります。ただし、優先度が高い方のチャンネルが先に転送を実行し、他方のチャンネルは保留されます。このような状況は、1 つの DMA 要求を使用して周辺モジュール (SPI 等) との間で双方向にデータを転送する場合に発生します。この場合、2 つの DMA チャンネルを使用し、1 チャンネルを周辺モジュールからの読み出し用に割り当て、もう 1 チャンネルを周辺モジュールへのデータ書き込み用に割り当てます。両チャンネルは同一の DMA 要求を使用します。

DMAxPAD レジスタが表 38-1 以外の値に初期化された場合、その周辺モジュール アドレスへの DMA チャンネル書き込みは無視されます。そのアドレスからの DMA チャンネル読み出しの結果は「0」です。

38.5.2 周辺モジュール コンフィグレーションのセットアップ

DMA セットアップの第 2 段階として、DMA 対応周辺モジュールを DMA 動作用に適切に設定する必要があります。表 38-2 に、各 DMA 対応周辺モジュールの設定要件の概要を示します。

表 38-2: DMA 対応周辺モジュールの設定における注意事項

DMA 対応周辺モジュール	設定時の注意事項
ECAN™ モジュール	ECAN バッファは DMA RAM 内に割り当てられます。ユーザは、DMA RAM 内のCANバッファ領域とFIFOの合計サイズを、ECAN FIFO制御(C1FCTRL) レジスタの DMA バッファサイズ ビット (DMABS<2:0>) で定義する必要があります。例 38- 9 のサンプルコードを参照してください。
データコンバータ インターフェイス (DCI)	DCI は、1 データワードをバッファリングするたびに割り込みを生成するように設定する必要があります。このため、DCI 制御 2 (DCICON2) レジスタのバッファ長制御ビット (BLEN<1:0>) を「00」に設定する必要があります。双方向のデータ転送 (Rx と Tx) をサポートするために、2 つの DMA チャンネルに同じ DCI 割り込みを DMA 要求として割り当てる必要があります。DCI モジュールがマスタとして動作してデータを受信するだけの場合でも、2 つの DMA チャンネルを使用して、片方のチャンネルでダミーデータを送信する必要があります。例 38- 11 のサンプルコードを参照してください。
10/12 ビット A/D コンバータ (ADC)	ADC を周辺モジュール間接アドレッシング モードで使用する場合、ADCx 制御 2 (ADCxCON2) レジスタの DMA アドレス インクリメント レートビット (SMPI<3:0>) と DCx 制御 4 (ADCxCON4) レジスタのアナログ入力ごとの DMA バッファサイズ ビット (DMABL<2:0>) を適切に設定する必要があります。さらに、ADC アドレスを生成するために、DCx 制御 1 (ADxCON1) レジスタの DMA バッファ構築モードビット (ADDMABM) を適切に設定する必要があります。詳細は **38.6.6.1「ADC の DMA アドレス生成サポート」**を参照してください。例 38- 5 と例 38- 7 のサンプルコードを参照してください。
シリアル ペリフェラル インターフェイス (SPI)	SPI モジュールがマスタとして動作してデータを受信するだけの場合でも、2 つの DMA チャンネルを割り当てて片方のチャンネルでダミーデータを送信する必要があります。別の方法として、NULL データ書き込みモードで 1 つの DMA チャンネルだけを使用する事もできます。詳細は **38.6.11「NULL データ書き込みモード」**を参照してください。例 38- 12 のサンプルコードを参照してください。
UART	UART は、1 キャラクタを受信または送信するたびに割り込みを生成するように設定する必要があります。UART レシーバは 1 キャラクタを受信するたびに Rx 割り込みを生成する必要があります。このため、ステータス / 制御レジスタ (UxSTA) の受信割り込みモード選択ビット (URXISEL<1:0>) を「00」または「01」に設定する必要があります。 UART トランシーバは 1 キャラクタを送信するたびに Tx 割り込みを生成する必要があります。このため、ステータス / 制御 (UxSTA) レジスタの送信割り込みモード選択ビット (UTXISEL0 と UTXISEL1) を「0」に設定する必要があります。例 38- 10 のサンプルコードを参照してください。
入力キャプチャ	入力キャプチャ モジュールは、キャプチャ イベントが発生するたびに割り込みを生成する必要があります。このため、入力キャプチャ制御 (ICxCON) レジスタの割り込みあたりキャプチャ回数ビット (ICI<1:0>) を「00」に設定する必要があります。例 38- 4 のサンプルコードを参照してください。

表 38-2: DMA 対応周辺モジュールの設定における注意事項 (続き)

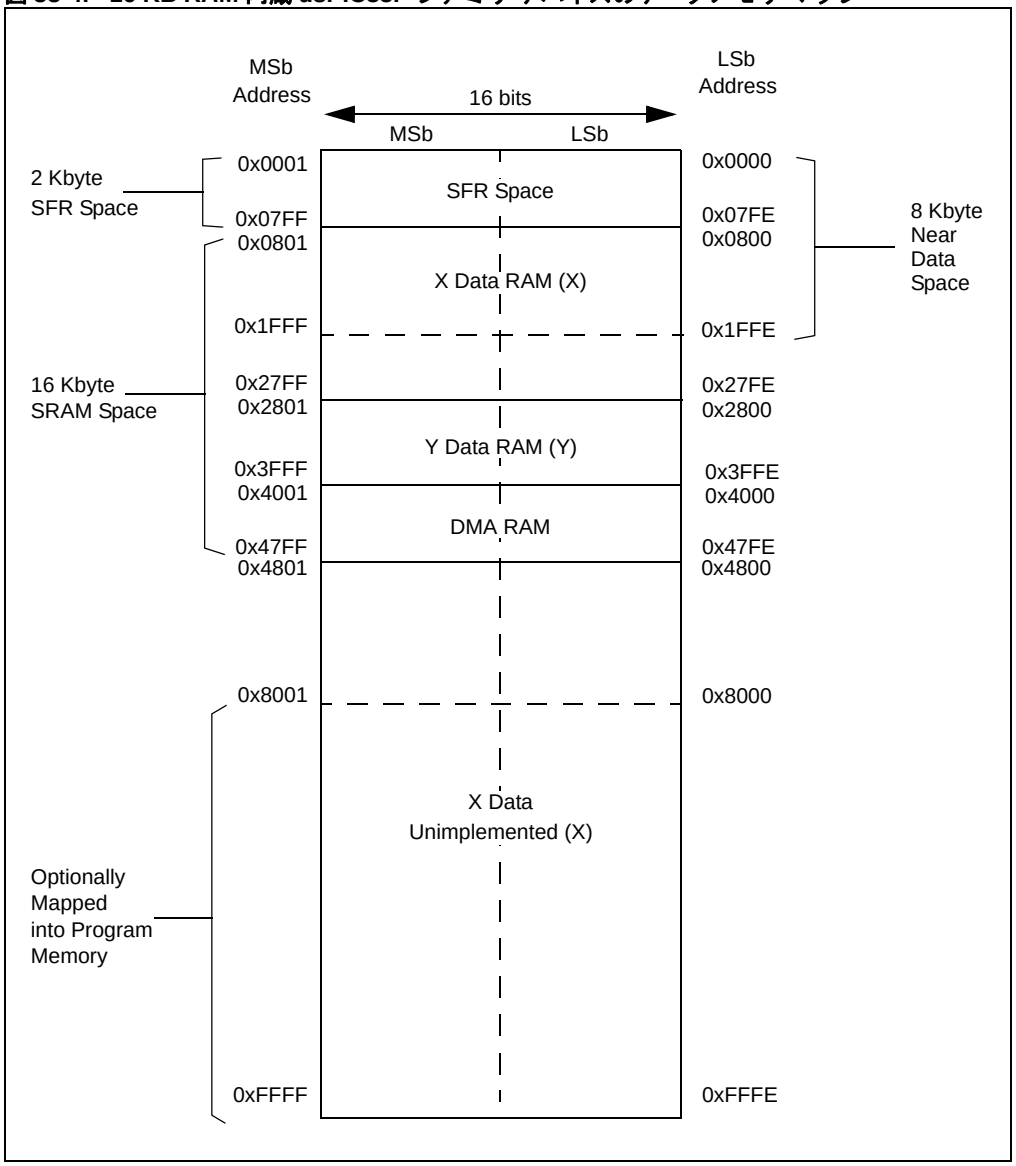
DMA 対応周辺モジュール	設定時の注意事項
出力コンペア	出力コンペア モジュールを DMA で動作させるために特別な設定は不要です。ただし、通常は DMA 要求の生成にタイマを使用するため、タイマを適正に設定する必要があります。例 38- 3 のサンプルコードを参照してください。
外部割り込みとタイマ	DMA 要求として選択できるのは、外部割り込み 0 と Timer2 および Timer3 だけです。これらの周辺モジュール自体は DMA 転送をサポートしませんが、DMA 対応周辺モジュールの DMA 転送をトリガするために使用できます。例えば、Timer2 は PWM モードで出力コンペア周辺モジュールの DMA トランザクションをトリガできます。例 38- 3 のサンプルコードを参照してください。
パラレルマスタ ポート (PMP)	PMP モジュールはマスタとして設定する必要があります。このため、パラレルポート モード (PMMODE) レジスタのパラレルポート モード選択ビット (MODE<1:0>) を「10」または「11」に設定する必要があります。また、データ転送のたびに割り込みを生成するために、PMMODE レジスタの割り込み要求モードビット (IRQM<1:0>) を「01」に設定する必要があります。詳細は dsPIC33F ファミリ リファレンス マニュアルの**セクション 35.「パラレルマスタ ポート (PMP)」**(DS70299) を参照してください。
D/A コンバータ (DAC)	DAC モジュールは、DAC FIFO がエンプティの時に割り込みを生成するように設定する必要があります。このため、DAC1ステータス/制御(DAC1STAT) レジスタの右チャンネル割り込みタイプ (RITYPE) ビットと左チャンネル割り込みタイプ (LITYPE) ビットの両方またはいずれかを「1」に設定する必要があります。詳細は dsPIC33F ファミリ リファレンス マニュアルの**セクション 33.「デジタル / アナログ コンバータ (DAC)」**(DS70298) を参照してください。

DMA 対応周辺モジュールでエラー条件が発生すると、ステータスフラグがセットされて割り込みが生成されます (ユーザ アプリケーションが対応する割り込みを有効化している場合のみ)。CPU が周辺モジュールを使用している時、データ割り込みハンドラでエラーフラグをチェックし、必要に応じて適切に対処する必要があります。これに対し、DMA チャンネルが周辺モジュールを使用している時、DMA はデータ転送要求にのみ応答する事ができ、その後発生するエラー条件を認識しません。従って、DMA 対応周辺モジュール内のエラー条件に関連する割り込みを全て有効化し、発生したそれらの割り込みに対しては、ユーザ定義割り込みサービスルーチン (ISR) で処理する必要があります。

38.5.3 メモリアドレスの初期化

DMA セットアップの第 3 段階として、特定メモリ領域に DMA アクセス用のメモリバッファを割り当てる必要があります。このメモリバッファのアドレスとサイズは、使用する dsPIC33F デバイスによって異なります (詳細は各デバイスのデータシートを参照してください)。図 38-4 に、16 KB の RAM を内蔵した dsPIC33F デバイスに 2 KB の DMA メモリが割り当てられている状態を示します。

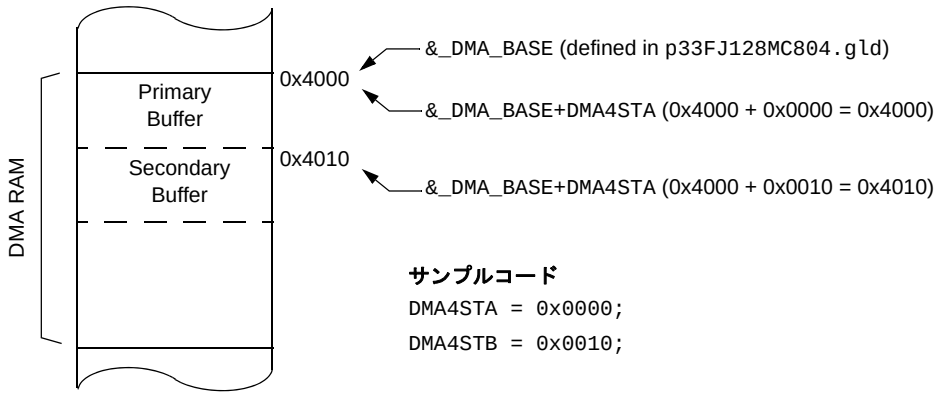
図 38-4: 16 KB RAM 内蔵 dsPIC33F ファミリ デバイスのデータメモリ マップ



DMA を正しく実行するには、読み出し元または書き込み先の DPSRAM アドレスを、DMA メモリの開始アドレスからのオフセットとして指定する必要があります。このオフセット情報は、DMA チャンネル x DPSRAM 開始アドレス オフセット A (DMAxSTA) レジスタと DMA チャンネル x DPSRAM 開始アドレス オフセット B (DMAxSTB) レジスタで設定します。

図 38-5 に、例として dsPIC33FJ128MC804 デバイスにおいて、チャンネル 4 のプライマリおよびセカンダリ バッファがそれぞれ 0x4000 と 0x4010 に配置されている状態を示します。

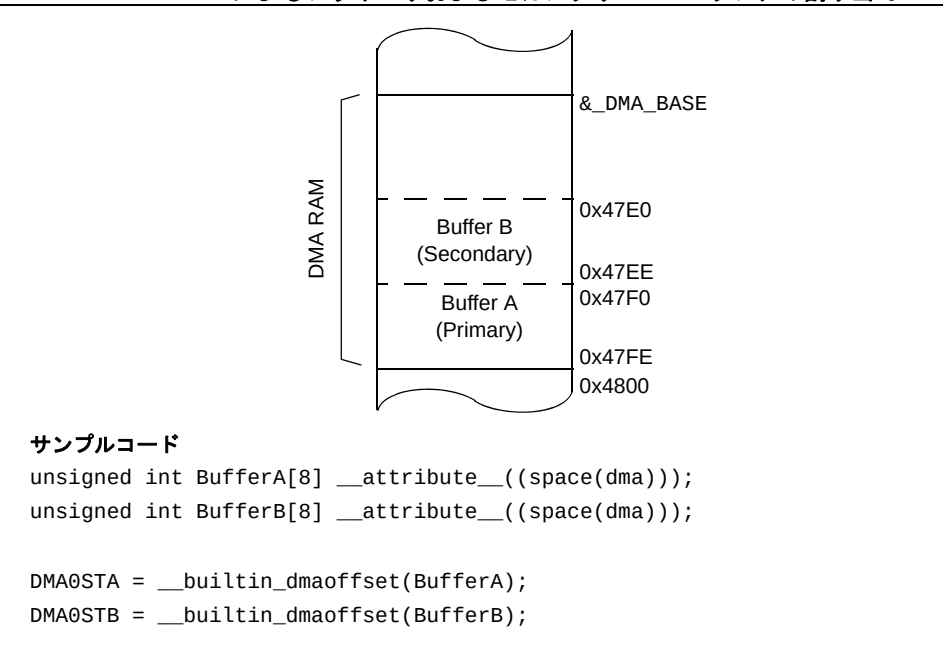
図 38-5: DMA メモリ内のプライマリおよびセカンダリ バッファの割り当て



このアドレス情報をアプリケーションにハードコードするには、使用するデバイスのメモリレイアウトを熟知する必要があります。また、DMA 転送完了後これらのバッファにアクセスするには、ポインタ計算が必要です。従って、これらの設定を異なるデバイスに簡単に移植する事はできません。

このため MPLAB® C30 コンパイラは、DMA バッファの初期化とアクセスを単純化するために、ビルトイン C プリミティブを提供します。例えば図 38-6 に記載したコードは、DMA メモリ内に 2 つのバッファを配置し、DMA チャンネルのアクセス先アドレスをそれらの位置に初期化します。

図 38-6: MPLAB® IDE によるプライマリおよびセカンダリ DMA バッファの割り当て



Note: MPLAB LINK30 リンカは、DMA メモリ空間の末尾からプライマリ バッファとセカンダリ バッファを逆順に配置します。

DMAxSTA (および / または DMAxSTB) レジスタが適正な値に初期化されなかったために、DMA チャンネルが DMA RAM 空間の外側の RAM アドレスに対して読み書きを行う結果となる場合、そのアドレスに対する DMA 書き込みは無視されます。そのメモリアドレスからの DMA チャンネル読み出し結果は「0」です。

38.5.4 DMA 転送カウントのセットアップ

DMA セットアップの第 4 段階では、データブロック転送を完了するまでに処理する必要のある DMA 要求の数 (N + 1) を各 DMA チャンネルに設定する必要があります。値「N」は、DMA チャンネル x 転送カウント (DMAxCNT) レジスタで指定します。DMAxCNT の値が「0」の場合、1 つのデータ要素を転送します。

DMAxCNT レジスタの値は、データ転送サイズ (バイトまたはワード ; DMAxCON レジスタの SIZE ビットで指定) とは無関係です。

DMAxCNT レジスタが適正な値に初期化されなかったために、DMA チャンネルが DMA RAM 空間の外側のRAMアドレスに対して読み書きを行う結果となる場合、そのメモリアドレスに対する DMA 書き込みは無視されます。そのメモリアドレスからの DMA チャンネル読み出し結果は「0」です。

38.5.5 動作モードのセットアップ

DMA セットアップの第 5 段階として、DMA チャンネル x 制御 (DMAxCON) レジスタで各 DMA チャンネルの動作モードを指定します。詳細は **38.6「DMA 動作モード」**を参照してください。

38.6 DMA 動作モード

DMA チャンネルは以下の動作モードをサポートします。

- ワードサイズまたはバイトサイズのデータ転送
- 転送方向 (周辺モジュールから DPSRAM へ、または、DPSRAM から周辺モジュールへ)
- CPU への割り込みタイミング (フルブロック (全データ) 転送時、またはハーフブロック (半分のデータ) 転送時)
- ポストインクリメントあり / なしの DPSRAM アドレッシング
- 周辺モジュール間接アドレッシング
- ワンショットまたは連続ブロック転送
- 2 つの開始アドレス オフセット (DMAxSTA と DMAxSTB) の自動的な交互切り換え (ピンポンモード)
- NULL データ書き込みモード

さらに、DMA 転送を 1 回だけ実行する手動モードも使用できます。

38.6.1 ワードサイズまたはバイトサイズのデータ転送

各 DMA チャンネルは、ワードサイズまたはバイトサイズのデータ転送用に設定できます。ワードデータの転送は、偶数アドレスに対してのみ可能です。バイトデータの転送は、偶数アドレスでも奇数アドレスでも可能です。

SIZE ビット (DMAxCON<14>) がクリアされている場合、ワードサイズのデータが転送されます。ポストインクリメント アドレッシング モードを使用するレジスタ間接が有効化されている場合、1 ワードを転送するたびにアドレスが 2 ずつポストインクリメントされます (**38.6.4「ポストインクリメント アドレッシング モードを使用するレジスタ間接」**参照)。

SIZE ビット (DMAxCON<14>) がセットされている場合、バイトサイズのデータが転送されます。ポストインクリメント アドレッシング モードを使用するレジスタ間接が有効化されている場合、1 バイトを転送するたびにアドレスが 1 ずつポストインクリメントされます。

38.6.2 転送方向

各 DMA チャンネルには転送方向 (周辺モジュール→ DPSRAM、または、DPSRAM →周辺モジュール) を設定できます。

DMAxCON レジスタの転送方向 (DIR) ビットがクリアされている場合、データは周辺モジュール (DMAxPAD が指定する周辺モジュールアドレス) から読み出され、DMAxSTA または DMAxSTB が指定する DPSRAM DMA メモリアドレス オフセットに書き込まれます。

DIR ビットがセットされている場合、データは DMAxSTA または DMAxSTB が指定する DPSRAM DMA メモリアドレス オフセットから読み出され、周辺モジュール (DMAxPAD が指定する周辺モジュール アドレス) に書き込まれます。

設定された各チャンネルは単方向データパスとして機能します。従って、周辺モジュールがデータ読み出しと書き込みの両方向に DMA コントローラを使用する場合、読み出し用に 1 チャンネルと書き込み用に 1 チャンネルを割り当てる必要があります。

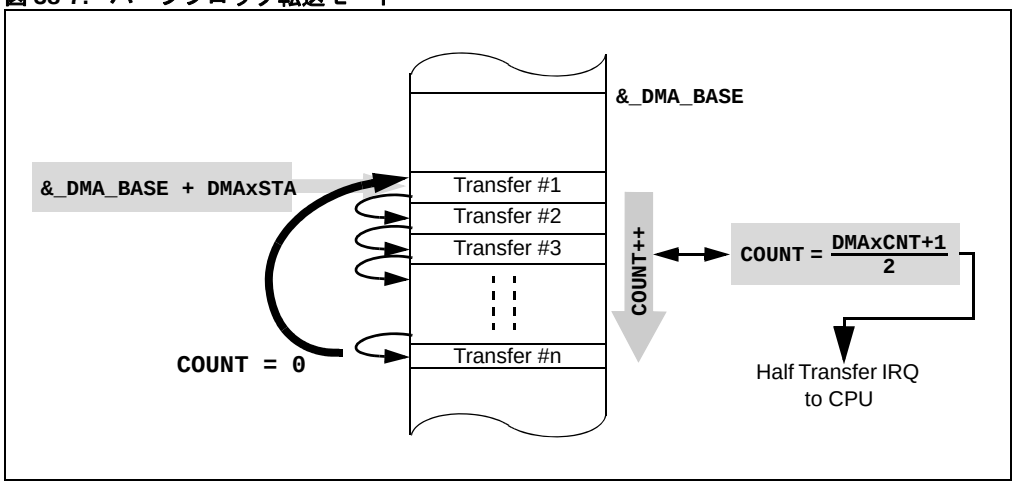
38.6.3 フルブロックまたはハーフブロック転送割り込み

各 DMA チャンネルは、ブロックデータの全部を転送した時または半分を転送した時に割り込みコントローラに対して割り込みを生成します。どちらのタイミングで割り込むかは、DMA チャンネル x 制御 (DMAxCON) レジスタの HALF ビットで選択できます。

- HALF = 0: フルブロック (全てのデータ) を転送した時に割り込みを生成する
- HALF = 1: ハーフブロック (半分のデータ) を転送した時に割り込みを生成する

DMA 連続モードを使用する場合、CPU は DMA 転送速度以上の速度で受信または送信データを処理する必要があります。ハーフブロック転送割り込みを使用してデータの半分を転送した時に割り込みを生成する事により、この要求を軽減できます。例えば、DMA コントローラが ADC を連続的に読み出している場合、ハーフブロック転送割り込みを使用すると、CPU はバッファが完全にフルになる前にバッファを処理できます。CPU によるバッファ処理が DMA によるバッファ書き込みを追い越さない限り、この方法によって CPU の応答時間に対する要求を緩和できます。図 38-7 に、このプロセスを示します。

図 38-7: ハーフブロック転送モード



HALF ビットがセットされている場合、DMA は常にバッファ A および / またはバッファ B の前半が転送された時にのみ割り込みを生成します。バッファ A および / または B の全データ転送完了時には割り込みは生成されません。つまり、DMA が (DMAxCNT + 1)/2 個のデータを転送した時にのみ割り込みが生成されます。(DMAxCNT + 1) が奇数である場合、割り込みは (DMAxCNT + 2)/2 個のデータ転送後に生成されます。

例えばDMA3をワンショット/ピンポンバッファ モード(MODE<1:0> = 11)に設定し、DMA3CNT = 7 とした場合、DMA3 割り込みは 2 回生成されます (バッファ A から 4 個のエレメント転送後と、バッファ B から 4 個のエレメント転送後)。詳細は **38.6.7「ワンショット モード」**と **38.6.9「ピンポンモード」**を参照してください。

DMA チャンネルはハーフブロック転送後またはフルブロック転送後のいずれかだけで割り込みを生成しますが、各 DMA 割り込み中にユーザ アプリケーションで HALF ビットの値をトグルする事により、ハーフブロック転送後とフルブロック転送後の両方で割り込みを生成できます。例として、DMA チャンネルの HALF ビットが「1」(ハーフブロック転送後に割り込みを生成) にセットされている場合、割り込みサービス中にユーザ アプリケーションが HALF ビットを「0」にリセットすると、その DMA チャンネルはフルブロック転送後にも割り込みを生成します。

これらの割り込みを有効化するために、割り込みコントローラ モジュールで割り込みイネーブル制御 (IECx) レジスタの対応する DMA 割り込みイネーブルビット (DMAxIE) をセットしておく必要があります (表 38-3 参照)。

表 38-3: DMA 割り込みを有効化 / 無効化するための割り込みコントローラの設定

DMA チャンネル	割り込みコントローラ レジスタ名 < ビット番号 >	対応する レジスタビット名	C 構造体 アクセスコード
0	IEC0<4>	DMA0IE	IEC0bits.DMA0IE
1	IEC0<14>	DMA1IE	IEC1bits.DMA1IE
2	IEC1<8>	DMA2IE	IEC1bits.DMA2IE
3	IEC2<4>	DMA3IE	IEC2bits.DMA3IE
4	IEC2<14>	DMA4IE	IEC2bits.DMA4IE
5	IEC3<13>	DMA5IE	IEC3bits.DMA5IE
6	IEC4<4>	DMA6IE	IEC4bits.DMA6IE
7	IEC4<5>	DAM7IE	IEC4bits.DMA7IE

例 38- 1 に、DMA チャンネル 0 割り込みを有効化するサンプルコードを示します。

例 38- 1: DMA チャンネル 0 割り込みを有効化するコード

```
IEC0bits.DMA0IE = 1;
```

DMA チャンネルの転送割り込みが発生すると、割り込みコントローラ内の対応するステータスフラグがセットされ、これにより ISR がトリガされます。転送完了 ISR の再実行を避けるために、ユーザ アプリケーションはこのステータスフラグをクリアする必要があります。

表 38-4 に、割り込みコントローラ モジュール内の割り込みフラグステータス (IFSx) レジスタと、対応するビット名 (DMAxIF) の一覧を示します。この表には、フラグをクリアする C 構造体アクセスコードも記載しています。

表 38-4: DMA 割り込みステータスフラグをクリアするための割り込みコントローラの設定

DMA チャンネル	割り込みコントローラ レジスタ名 < ビット番号 >	対応する レジスタビット名	C 構造体 アクセスコード
0	IFS0<4>	DMA0IF	IFS0bits.DMA0IE
1	IFS0<14>	DMA1IF	IFS0bits.DMA1IE
2	IFS1<8>	DMA2IF	IFS1bits.DMA2IE
3	IFS2<4>	DMA3IF	IFS2bits.DMA3IE
4	IFS2<14>	DMA4IF	IFS2bits.DMA4IE
5	IFS3<13>	DMA5IF	IFS3bits.DMA5IE
6	IFS4<4>	DMA6IF	IFS4bits.DMA6IE
7	IFS4<5>	DMA7IF	IFS4bits.DMA7IE

例として、DMA チャンネル 0 割り込みが有効化されている場合、DMA チャンネル 0 の転送が完了すると、対応する割り込みが割り込みコンローラに対して発行されます。ステータスフラグをクリアして ISR の再実行を回避するために、下記のコードを DMA チャンネル 0 の ISR に含める必要があります。

例 38-2: DMA チャンネル 0 割り込みをクリアするコード

```
void __attribute__((interrupt, no_auto_psv)) _DMA0Interrupt(void)
{
  ...

IFS0bits.DMA0IF = 0;
}
```

38.6.4 ポストインクリメント アドレッシング モードを使用するレジスタ間接

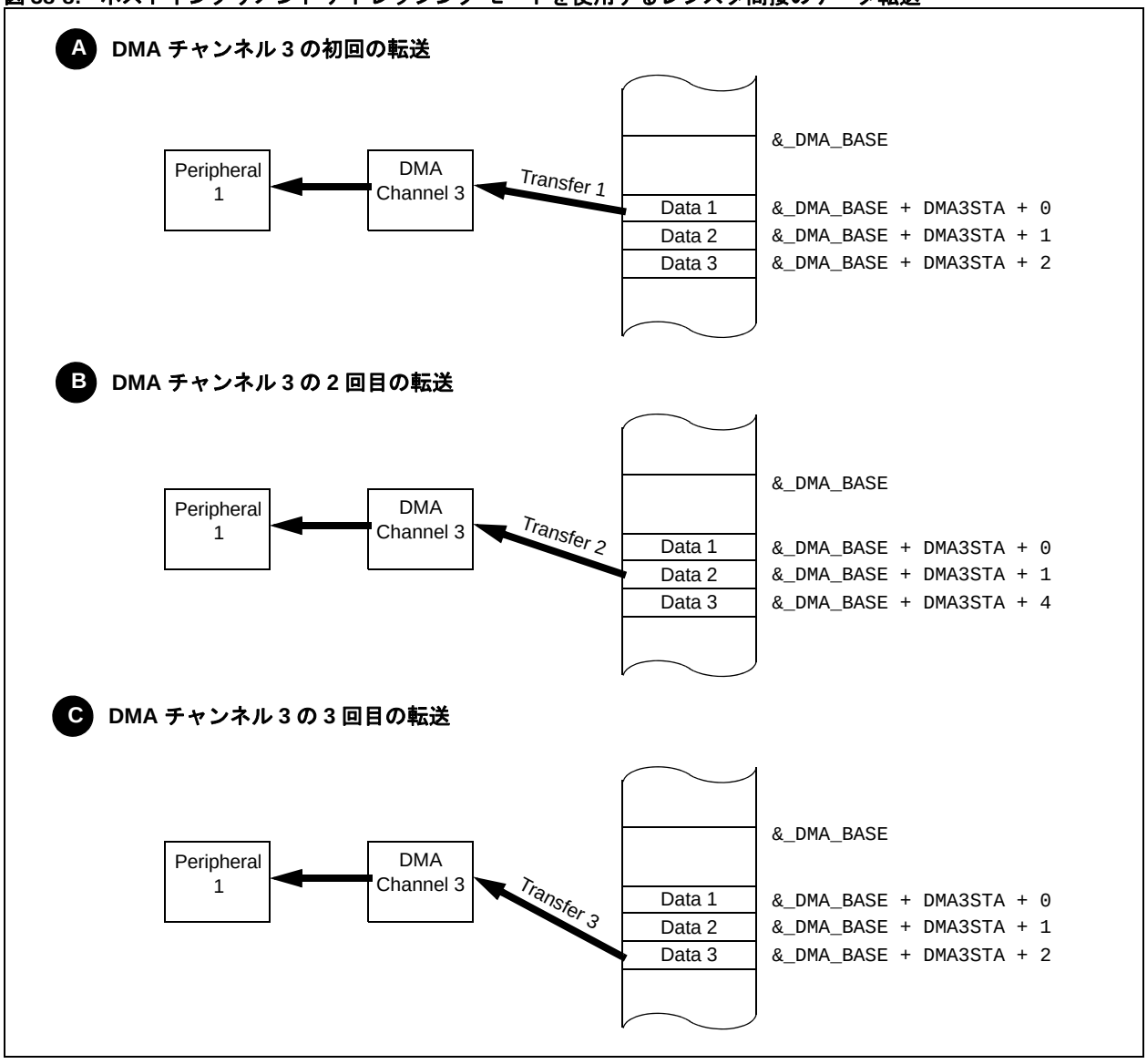
ポストインクリメント アドレッシング モードを使用するレジスタ間接では、毎回の転送後に DPSRAM アドレスをインクリメントする事によってデータ ブロックを移動します。

DMA コントローラのリセット時に、DMA チャンネルはこの既定値モードに設定されます。DMA チャンネル制御 (DMAxCON) レジスタのアドレッシング モード選択ビット (AMODE<1:0>) を「00」に設定すると、このモードが選択されます。このモードでは、DPSRAM 開始アドレス オフセット (DMAxSTA または DMAxSTB) レジスタによって DPSRAM バッファの開始アドレスが指定されます。

ユーザ アプリケーションは、DPSRAM 開始アドレス オフセット レジスタを読み出す事によって、直前の DPSRAM 転送アドレス オフセットを検出します。ただし、DMA コントローラはこのレジスタの内容を変更しません。

図 38-8 に、このモードでのデータ転送を示します。

図 38-8: ポストインクリメント アドレッシング モードを使用するレジスタ間接のデータ転送



例 38- 3: ポストインクリメント アドレッシング モードを使用するレジスタ間接による出力コンペア データの DMA 転送

出力コンペア 1 モジュールを PWM モードに設定する :

```
OC1CON = 0;                // Reset OC module
OC1R = 0x60;               // Initialize PWM Duty Cycle
OC1RS = 0x60;              // Initialize PWM Duty Cycle Buffer

OC1CONbits.OCM = 6;        // Configure OC for the PWM mode
```

DMA チャンネル 3 を、Timer2 を DMA 要求源とするポストインクリメント モードに設定する :

```
unsigned int BufferA[32] __attribute__((space(dma)));
/* Insert code here to initialize BufferA with desired Duty Cycle values */

DMA3CONbits.AMODE = 0;     // Configure DMA for Register indirect mode
                           // with post-increment
DMA3CONbits.MODE = 0;      // Configure DMA for Continuous mode
DMA3CONbits.DIR   = 1;     // RAM-to-Peripheral data transfers
DMA3PAD = (volatile unsigned int)&OC1RS;// Point DMA to OC1RS
DMA3CNT = 31;              // 32 DMA request
DMA3REQ = 7;               // Select Timer2 as DMA Request source

DMA3STA = __builtin_dmaoffset(BufferA);

IFS2bits.DMA3IF = 0;       // Clear the DMA interrupt flag bit
IEC2bits.DMA3IE = 1;       // Set the DMA interrupt enable bit

DMA3CONbits.CHEN = 1;      // Enable DMA
```

Timer2 を出力コンペア PWM モードに設定する :

```
PR2 = 0xBF;                // Initialize PWM period
T2CONbits.TON = 1;         // Start timer 2
```

DMA チャンネル 3 割り込みハンドラを設定する :

```
void __attribute__((interrupt, no_auto_psv)) _DMA3Interrupt(void)
{
 /* Update BufferA with new Duty Cycle values if desired here*/

 IFS2bits.DMA3IF = 0;      //Clear the DMA3 Interrupt Flag
}
```

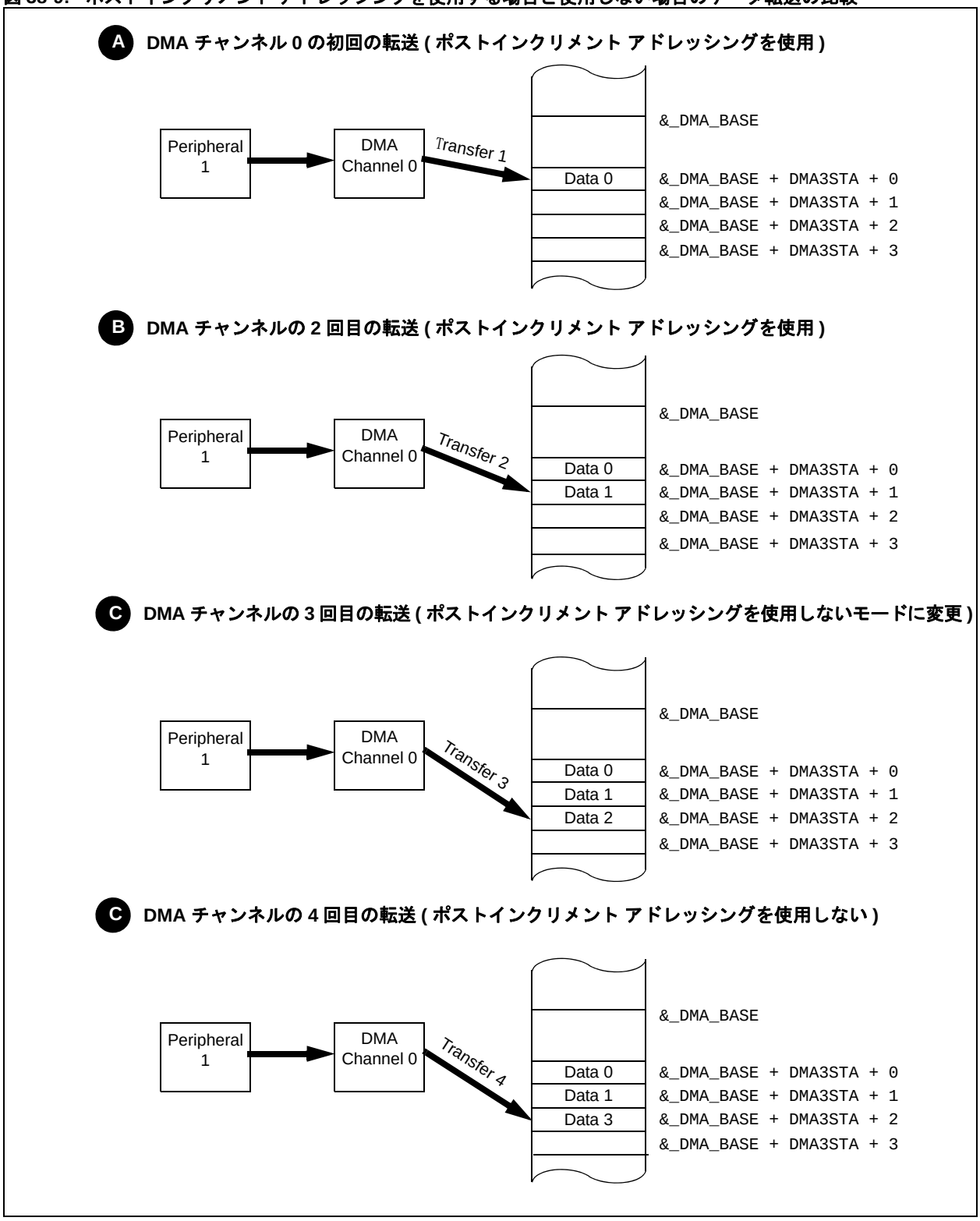
38.6.5 ポストインクリメント アドレッシングモードを使用しないレジスタ間接

ポストインクリメント アドレッシング モードを使用しないレジスタ間接では、毎回の転送後にデータバッファの開始アドレスをインクリメントせずにデータブロックを移動します。このモードでは、DPSRAM 開始アドレス オフセット (DMAxSTA または DMAxSTB) レジスタによって DPSRAM バッファの開始アドレスからのオフセットが指定されます。DMA データ転送の実行中に、DPSRAM アドレスは次のアドレスへインクリメントされません。従って、次回の DMA データ転送も同じ DPSRAM アドレスに対して開始されます。

DMAチャンネル制御(DMAxCON)レジスタのアドレッシング モード選択ビット(AMODE<1:0>)を「01」に設定すると、このモードが選択されます。

DMA チャンネルがアクティブな状態 (DMA 転送を何回か実行した後等) でアドレッシングモードをポストインクリメント アドレッシング モードを使用しないレジスタ間接に変更した場合、DMA DPSRAM アドレスは現在の DPSRAM バッファ位置を指します (つまり、DMAxSTA または DMAxSTB の値は、現在の DPSRAM バッファ位置とは異なる可能性があります)。図 38-9 に、周辺モジュールから DMA DPSRAM へのデータ転送で、ポストインクリメント アドレッシングを使用する場合と使用しない場合の動作の比較を示します。

図 38-9: ポストインクリメント アドレッシングを使用する場合と使用しない場合のデータ転送の比較



例 38- 4: ポストインクリメント アドレッシング モードを使用しないレジスタ間接による入力キャプチャ データの DMA 転送

入力キャプチャ 1 を DMA 動作用に設定する :

```
IC1CON = 0;              // Reset IC module
IC1CONbits.ICTMR = 1;    // Select Timer2 contents for capture
IC1CONbits.ICM   = 2;    // Capture every falling edge
IC1CONbits.ICI   = 0;    // Generate DMA request on every capture event
```

Timer2 を入力キャプチャ モジュール用に設定する :

```
PR2 = 0xBF;              // Initialize count value
T2CONbits.TON = 1;       // Start timer
```

DMA チャンネル 0 を「ポストインクリメント アドレッシングを使用しない」モードに設定する :

```
unsigned int CaptureValue __attribute__((space(dma)));

DMA0CONbits.AMODE = 1;   // Configure DMA for Register indirect
                         // without post-increment
DMA0CONbits.MODE = 0;    // Configure DMA for Continuous mode
DMA0PAD = (volatile unsigned int)&IC1BUF;// Point DMA to IC1BUF
DMA0CNT = 0;             // Interrupt after each transfer
DMA0REQ = 1;             // Select Input Capture module as DMA Request source

DMA3STA = __builtin_dmaoffset(&CaptureValue);

IFS0bits.DMA0IF = 0;     // Clear the DMA interrupt flag bit
IEC0bits.DMA0IE = 1;     // Set the DMA interrupt enable bit

DMA0CONbits.CHEN = 1;    // Enable DMA
```

DMA チャンネル 0 割り込みハンドラを設定する :

```
void __attribute__((interrupt, no_auto_psv)) _DMA3Interrupt(void)
{
 /* Process CaptureValue variable here*/

 IFS0bits.DMA0IF = 0;    //Clear the DMA3 Interrupt Flag
}
```

38.6.6 周辺モジュール間接アドレッシング モード

周辺モジュール間接アドレッシング モードは、DMA チャンネルではなく周辺モジュールが DPSRAM アドレスの可変部を制御する特殊なアドレッシング モードです。このモードでは、周辺モジュールが DPSRAM アドレスの下位ビット (LSb) を生成し、DMA チャンネルは固定されたバッファ ベースアドレスを提供します。ただし、このモードにおいても、DMA チャンネルは引き続きデータ転送の調整 / 転送数のカウント / 対応する CPU 割り込みの生成を行います。

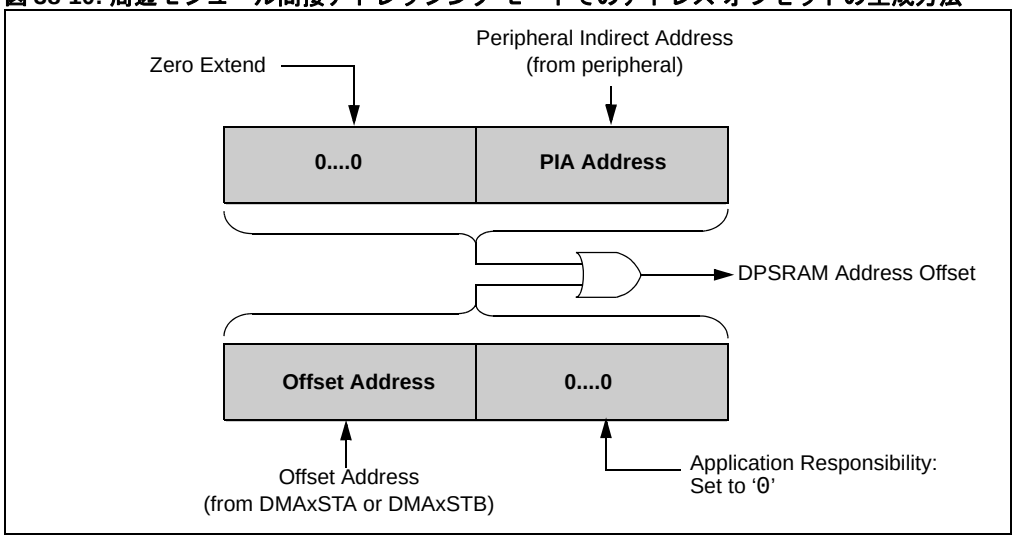
周辺モジュール間接アドレッシング モードでも、周辺モジュール側の要求に応じて、どちらの方向にもデータを転送できます。従って各 DMA チャンネルは、周辺モジュールの読み出しまたは書き込みのいずれか用に適正に設定する必要があります。

DMAチャンネル制御(DMAxCON)レジスタのアドレッシングモード選択ビット(AMODE<1:0>)を「1x」に設定すると、周辺モジュール間接アドレッシング モードが選択されます。

周辺モジュール間接アドレッシング モードでは、このモードをサポートする周辺モジュール側の要求に合わせて DMA 動作を設定できます。すなわち、DPSRAM 内のアクセス先アドレスシーケンスは、周辺モジュール側で定義されます。これにより、例えば ADC からの読み出しデータを複数バッファに並び換えて格納する事により、後の CPU 処理負荷を軽減できます。

周辺モジュールが周辺モジュール間接アドレッシング モードをサポートしている場合、その周辺モジュールからの DMA 要求割り込みが発生すると、周辺モジュールから DMA チャンネルにアドレスが渡されます。この要求に応答する DMA チャンネルも周辺モジュール間接アドレッシング用に有効化されている場合、バッファのベースアドレスとゼロ拡張した周辺モジュール間接アドレスの論理和 (OR) によって、実際の DPSRAM オフセットアドレスが生成されます (図 38-10 参照)。

図 38-10: 周辺モジュール間接アドレッシング モードでのアドレス オフセットの生成方法



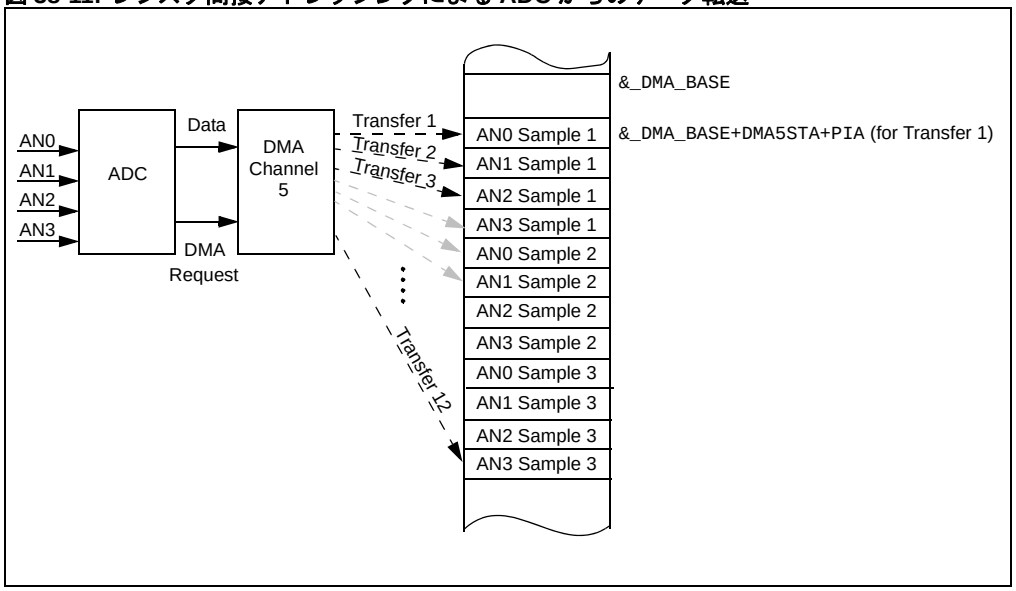
周辺モジュールが生成するアドレスの下位ビット数は、周辺モジュールによって決まります。アプリケーション プログラムは、DPSRAM 内バッファのベースアドレスを選択する必要があります。このベースアドレスの下位ビット (周辺モジュールが生成する下位ビットに対応するビット) はゼロである事が必要です。他のモードと同様に、DPSRAM 開始アドレス オフセット レジスタを読み出すと、直前の DPSRAM 転送アドレスオフセットが返されますが、これには上記アドレス オフセット計算が反映されます。DMA チャンネルが周辺モジュール間接アドレッシング用に設定されていない場合、周辺モジュールからのアドレスは無視され、データ転送は通常モードで行われます。

周辺モジュール間接アドレッシング モードは、他の全ての動作モードと互換性を持ち、現在のところ ADC および ECAN モジュールでサポートされます。

38.6.6.1 ADC の DMA アドレス生成サポート

周辺モジュール間接アドレッシング モードでは、周辺モジュールがアドレッシング シーケンスを定義します。これにより、アドレッシング シーケンスを周辺モジュールの機能により適合させる事ができます。例として、入力 1/2/3 を順次連続的 (例 : 0, 1, 2, 3, 0, 1, 2, ,,,,) に変換するように ADC を設定し、ポストインクリメント アドレッシング モードを使用するレジスタ間接に設定された DMA チャンネルを割り当てた場合、DMA 転送は ADC データを 1 つのシーケンシャル バッファに格納します (図 38-11 参照)。例 38- 5 に、このコンフィグレーション用のサンプルコードを示します。

図 38-11: レジスタ間接アドレッシングによる ADC からのデータ転送



例 38- 5: レジスタ間接アドレッシングによる ADC からのデータ転送

ADC1 をチャンネル 0 ～ 1 でサンプリングするように設定する :

```
AD1CON1bits.FORM = 3;     // Data Output Format:Signed Fraction (Q15 format)
AD1CON1bits.SSRC = 2;     // Sample Clock Source:GP Timer starts conversion
AD1CON1bits.ASAM = 1;     // Sampling begins immediately after conversion
AD1CON1bits.AD12B = 0;    // 10-bit ADC operation
AD1CON1bits.SIMSAM = 0;   // Samples individual channels sequentially

AD1CON2bits.BUFM = 0;
AD1CON2bits.CSCNA = 1;    // Scan CH0+ Input Selections during Sample A bit
AD1CON2bits.CHPS = 0;     // Converts CH0

AD1CON3bits.ADRC = 0;     // ADC Clock is derived from Systems Clock
AD1CON3bits.ADCS = 63;    // ADC Conversion Clock

//AD1CHS0:A/D Input Select Register
AD1CHS0bits.CH0SA = 0;    // MUXA +ve input selection (AIN0) for CH0
AD1CHS0bits.CH0NA = 0;    // MUXA -ve input selection (Vref-) for CH0

//AD1CHS123:A/D Input Select Register
AD1CHS123bits.CH123SA = 0;// MUXA +ve input selection (AIN0) for CH1
AD1CHS123bits.CH123NA = 0;// MUXA -ve input selection (Vref-) for CH1

//AD1CSSH/AD1CSSL:A/D Input Scan Selection Register
AD1CSSH = 0x0000;
AD1CSSL = 0x000F;         // Scan AIN0, AIN1, AIN2, AIN3 inputs
```

Timer3 を ADC1 変換のトリガ用に設定する :

```
TMR3 = 0x0000;
PR3 = 4999;               // Trigger ADC1 every 125usec @ 40 MIPS
IFS0bits.T3IF = 0;        // Clear Timer 3 interrupt
IEC0bits.T3IE = 0;        // Disable Timer 3 interrupt

T3CONbits.TON = 1;        //Start Timer 3
```

DMA チャンネル 5 をポストインクリメント アドレッシング モードを使用するレジスタ間接に設定する :

```
unsigned int BufferA[32] __attribute__((space(dma)));
unsigned int BufferB[32] __attribute__((space(dma)));

DMA5CONbits.AMODE = 0;    // Configure DMA for Register indirect mode
                          // with post-increment
DMA5CONbits.MODE = 2;     // Configure DMA for Continuous Ping-Pong mode
DMA5PAD = (volatile unsigned int)&ADC1BUF0;// Point DMA to ADC1BUF0
DMA5CNT = 31;             // 32 DMA request
DMA5REQ = 13;             // Select ADC1 as DMA Request source

DMA5STA = __builtin_dmaoffset(BufferA);
DMA5STB = __builtin_dmaoffset(BufferB);

IFS3bits.DMA5IF = 0;      //Clear the DMA interrupt flag bit
IEC3bits.DMA5IE = 1;      //Set the DMA interrupt enable bit

DMA5CONbits.CHEN=1;       // Enable DMA
```

例 38-5: レジスタ間接アドレッシングによる ADC からのデータ転送 (続き)

DMA チャンネル 5 の割り込みハンドラを設定する :

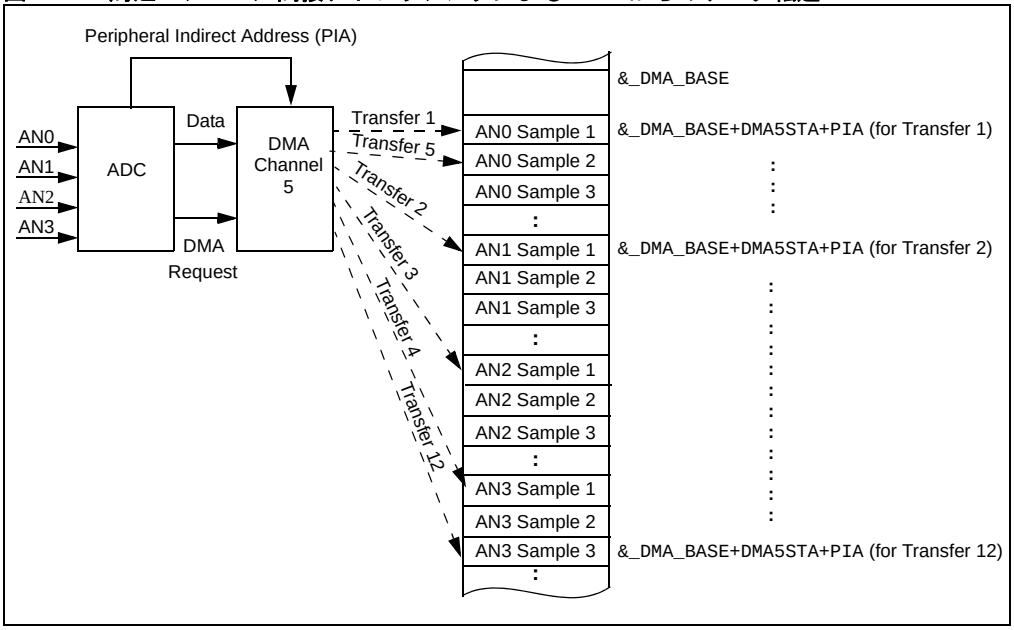
```
unsigned int DmaBuffer = 0;

void __attribute__((interrupt, no_auto_psv)) _DMA5Interrupt(void)
{
 // Switch between Primary and Secondary Ping-Pong buffers
if(DmaBuffer == 0)
 {
   ProcessADCSamples(BufferA);
 }
 else
 {
   ProcessADCSamples(BufferB);
 }

 DmaBuffer ^= 1;

 IFS3bits.DMA5IF = 0;    //Clear the DMA5 Interrupt Flag
}
```

ADC1 を DMA 動作用に設定する :

```
AD1CON1bits.ADDMABM = 0;  // Don't Care:ADC address generation is
                          // ignored by DMA
AD1CON2bits.SMPI    = 3;  // Don't Care
AD1CON4bits.DMABL   = 3;  // Don't Care

IFS0bits.AD1IF      = 0;  // Clear the A/D interrupt flag bit
IEC0bits.AD1IE      = 0;  // Do Not Enable A/D interrupt
AD1CON1bits.ADON    = 1;  // Turn on the A/D converter
```

通常のアルゴリズムは、データを ADC チャンネルの変換順に 1 つのバッファに格納するため、転送後にデータの並べ換えまたはインデックス処理による不要データ位置の読み飛ばしが必要です。これらの方法では、追加の処理コードが必要であり、実行時間が増加します。ADC の周辺モジュール間接アドレッシング モードは、ADC チャンネル別のバッファにデータを格納する特殊なアドレッシング モードを定義します。DMA チャンネルを周辺モジュール間接アドレッシング モードに設定した場合、前述の例とは異なり、DMA 転送された ADC データはチャンネル別のバッファに格納されます (図 38-12 参照)。

図 38-12: 周辺モジュール間接アドレッシングによる ADC からのデータ転送



この種の ADC アドレッシングを行う場合、ADCx 制御 1 (ADxCON1) レジスタの DMA バッファ構築モード (ADDMABM) ビットをクリアする必要があります。このビットがセットされている場合、ADC は変換順にアドレスを生成します (ポストインクリメント アドレッシング モードを使用するレジスタ間接と同じ)。

前述のように、周辺モジュール間接アドレッシングでは、DPSRAM 開始アドレス オフセット レジスタ (DMAxSTA と DMAxSTB) をユーザ アプリケーションで初期化する際に、周辺モジュール間接アドレスに対応する下位ビットのビット数に特に注意が必要です。ADC の場合、このビット数は ADC バッファのサイズと個数によって決まります。

ADC バッファの個数は、ADCx 制御 2 (ADxCON2) レジスタの DMA アドレス インクリメント レートビット (SMPI<3:0>) の値を使用して初期化されます。各 ADC バッファのサイズは、ADCx 制御 4 (ADCxCON4) レジスタのアナログ入力ごとの DMA バッファ位置ビット (DMABL<2:0>) の値を使用して初期化されます。例として SMPI<3:0> と DMABL<2:0> を 3 に初期化した場合、8 ワード ($2^{DMABL<2:0>}$) の ADC バッファが 4 個 (SMPI<3:0> + 1) 配置されます (計 32 ワード = 64 バイト)。この場合、DMAxSTA と DMAxSTB に書き込まれるアドレス オフセットの下位 6 ビット ($2^{6\ bits}$ = 64 バイト) をゼロに設定する必要があります。

MPLAB C30 コンパイラを使用して DMAxSTA および DMAxSTAB レジスタを初期化する場合、データ属性で適切なデータ配置を指定する必要があります。このような場合、例 38- 6 に示すサンプルコードを使用すると、DMAxSTA および DMAxSTB レジスタを適切に初期化できます。

例 38- 6: MPLAB® C30 による DMA バッファの配置

```
int  BufferA[4][8] __attribute__((space(dma),aligned(64)));
int  BufferB[4][8] __attribute__((space(dma),aligned(64)));

DMA0STA = __builtin_dmaoffset(&BufferA[0][0]);
DMA0STB = __builtin_dmaoffset(&BufferB[0][0]);
```

例 38- 7 に、このコンフィグレーション用のサンプルコードを示します。

例 38- 7: 周辺モジュール間接アドレッシングによる ADC データの DMA 転送

ADC1 をチャンネル 0 ～ 1 でサンプリングするように設定する :

```
AD1CON1bits.FORM = 3;     // Data Output Format:Signed Fraction (Q15 format)
AD1CON1bits.SSRC = 2;     // Sample Clock Source:GP Timer starts conversion
AD1CON1bits.ASAM = 1;     // Sampling begins immediately after conversion
AD1CON1bits.AD12B = 0;    // 10-bit ADC operation
AD1CON1bits.SIMSAM = 0;  // Samples multiple channels sequentially

AD1CON2bits.BUFM = 0;
AD1CON2bits.CSCNA = 1;    // Scan CH0+ Input Selections during Sample A bit
AD1CON2bits.CHPS = 0;     // Converts CH0

AD1CON3bits.ADRC = 0;     // ADC Clock is derived from Systems Clock
AD1CON3bits.ADCS = 63;    // ADC Conversion Clock

//AD1CHS0:A/D Input Select Register
AD1CHS0bits.CH0SA = 0;    // MUXA +ve input selection (AIN0) for CH0
AD1CHS0bits.CH0NA = 0;    // MUXA -ve input selection (Vref-) for CH0

//AD1CHS123:A/D Input Select Register
AD1CHS123bits.CH123SA = 0;// MUXA +ve input selection (AIN0) for CH1
AD1CHS123bits.CH123NA = 0;// MUXA -ve input selection (Vref-) for CH1

//AD1CSSH/AD1CSSL:A/D Input Scan Selection Register
AD1CSSH = 0x0000;
AD1CSSL = 0x000F;         // Scan AIN0, AIN1, AIN2, AIN3 inputs
```

Timer3 を ADC1 の変換トリガ用に設定する :

```
TMR3 = 0x0000;
PR3 = 4999;// Trigger ADC1 every 125usec
IFS0bits.T3IF = 0;        // Clear Timer 3 interrupt
IEC0bits.T3IE = 0;        // Disable Timer 3 interrupt

T3CONbits.TON = 1;        //Start Timer 3
```

DMA チャンネル 5 を周辺モジュール間接アドレッシング モードに設定する :

```
struct
{
 unsigned int Adc1Ch0[8];
 unsigned int Adc1Ch1[8];
 unsigned int Adc1Ch2[8];
 unsigned int Adc1Ch3[8];
} BufferA __attribute__((space(dma)));

struct
{
 unsigned int Adc1Ch0[8];
 unsigned int Adc1Ch1[8];
 unsigned int Adc1Ch2[8];
 unsigned int Adc1Ch3[8];
} BufferB __attribute__((space(dma)));

DMA5CONbits.AMODE = 2;    // Configure DMA for Peripheral indirect mode
DMA5CONbits.MODE = 2;     // Configure DMA for Continuous Ping-Pong mode
DMA5PAD = (volatile unsigned int)&ADC1BUF0;// Point DMA to ADC1BUF0
DMA5CNT = 31;             // 32 DMA request (4 buffers, each with 8 words)
DMA5REQ = 13;             // Select ADC1 as DMA Request source

DMA5STA = __builtin_dmaoffset(&BufferA);
DMA5STB = __builtin_dmaoffset(&BufferB);

IFS3bits.DMA5IF = 0;      //Clear the DMA interrupt flag bit
IEC3bits.DMA5IE = 1;      //Set the DMA interrupt enable bit

DMA5CONbits.CHEN=1;       // Enable DMA
```

例 38-7: 周辺モジュール間接アドレッシングによる ADC データの DMA 転送 (続き)

DMA チャンネル 5 割り込みハンドラを設定する :

```
unsigned int DmaBuffer = 0;

void __attribute__((interrupt, no_auto_psv)) _DMA5Interrupt(void)
{
 // Switch between Primary and Secondary Ping-Pong buffers
if(DmaBuffer == 0)
 {
   ProcessADCSamples(BufferA.Adc1Ch0);
   ProcessADCSamples(BufferA.Adc1Ch1);
   ProcessADCSamples(BufferA.Adc1Ch2);
   ProcessADCSamples(BufferA.Adc1Ch3);
 }
 else
 {
   ProcessADCSamples(BufferB.Adc1Ch0);
   ProcessADCSamples(BufferB.Adc1Ch1);
   ProcessADCSamples(BufferB.Adc1Ch2);
   ProcessADCSamples(BufferB.Adc1Ch3);
 }

 DmaBuffer ^= 1;

 IFS3bits.DMA5IF = 0;    //Clear the DMA5 Interrupt Flag
}
```

ADC1 を DMA 動作用に設定する :

```
AD1CON1bits.ADDMABM = 0; // DMA buffers are built in scatter/gather mode
AD1CON2bits.SMPI    = 3; // 4 ADC buffers
AD1CON4bits.DMABL   = 3; // Each buffer contains 8 words

IFS0bits.AD1IF      = 0; // Clear the A/D interrupt flag bit
IEC0bits.AD1IE      = 0; // Do Not Enable A/D interrupt
AD1CON1bits.ADON    = 1; // Turn on the A/D converter
```

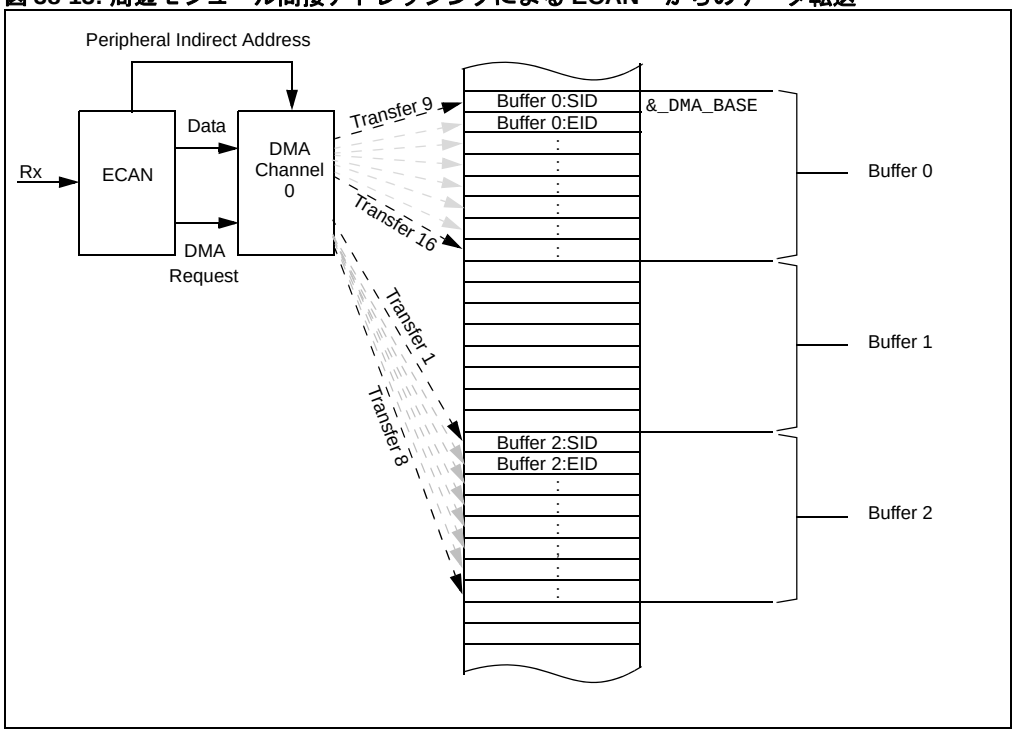
38.6.6.2 ECAN の DMA アドレス生成サポート

ECAN モジュールでも、周辺モジュール間接アドレッシング モードを使用する事により、特殊なアドレッシング機能を定義できます。dsPIC33F が CAN バス経由のメッセージをフィルタ処理して受信する場合、メッセージは下記の 2 つに分類できます。

- 処理する必要のある受信メッセージ
- 処理せずに他の CAN ノードへ転送する必要のある受信メッセージ

処理が必要な受信メッセージは、ユーザ アプリケーションで処理するために、8 ワードずつのバッファに再構成する必要があります。この場合、DMA RAM 内に複数の ECAN バッファを配置し、ECAN 周辺モジュール側で受信 (または送信) データの RAM アドレスを生成する事により、処理を単純化できます (図 38-13 参照)。この例では、受信データを最初にバッファ 2 に格納し、次にバッファ 0 に格納しています。ECAN モジュールは、DMA RAM 内にデータを適切に格納するために、格納先アドレス (周辺モジュール間接アドレス) を生成します。

図 38-13: 周辺モジュール間接アドレッシングによる ECAN™ からのデータ転送



DMA が周辺モジュール間接アドレッシング モードで動作する場合、前述のように、DPSRAM 開始アドレス オフセット レジスタ (DMAxSTA と DMAxSTB) をユーザ プリケーションで初期化する際に、周辺モジュール間接アドレスに対応する下位ビットのビット数に特に注意が必要です。ECAN モジュールの場合、このビット数は、ECAN FIFO 制御レジスタ (CiFCTRL) の DMA バッファサイズ ビット (DMABS<2:0>) によって定義される ECAN バッファ数によって決まります。

例えば、ECAN モジュールが 12 個のバッファを確保 (DMABS<2:0> ビットを「3」に設定) している場合、12 個 x 8 ワード (計 96 ワード = 192 バイト) のバッファが配置されます。この場合、DMAxSTA および DMAxSTB レジスタに書き込まれるアドレス オフセットの下位 8 ビット ($2^{8\ bits}$ = 256 バイト) を「0」に設定する必要があります。MPLAB C30 コンパイラを使用して DMAxSTA レジスタを初期化する場合、データ属性で適切なデータ配置を指定する必要があります。このような場合、例 38- 8 のサンプルコードを使用すると、DMAxSTA レジスタを適切に初期化できます。

例 38- 8: MPLAB® C30 による DMA バッファの配置

```
int  BufferA[12][8] __attribute__((space(dma),aligned(256)));

DMA0STA = __builtin_dmaoffset(&BufferA[0][0]);
```

例 38- 9 に、このコンフィグレーション用のサンプルコードを示します。

受信メッセージを処理する必要がない場合もあります。例えば、車載アプリケーションでは、受信メッセージを CPU で処理せずに他のノードへ単純に転送する場合があります。この場合、受信バッファをメモリ内で並び換える必要はなく、転送可能となり次第、他のノードへ転送できます。

このようなデータ転送は、ポストインクリメント アドレッシング モードを使用するレジスタ間接で実行できます。図 38-14 に、このプロセスを示します。

例 38- 9: 周辺モジュール間接アドレッシングによる ECAN™ データの DMA 転送

2 つのフィルタを使用して ECAN1 を設定する :

```
/* Initialize ECAN clock first.See ECAN section for example code */


C1CTRL1bits.WIN = 1;      // Enable filter window
C1FEN1bits.FLTEN0 = 1;    // Filter 0 is enabled
C1FEN1bits.FLTEN1 = 1;    // Filter 1 is enabled
C1BUFPNT1bits.F0BP = 0;   // Filter 0 points to Buffer0
C1BUFPNT1bits.F1BP = 2;   // Filter 1 points to Buffer2

C1RXF0SID = 0xFFEA;       // Filter 0 configuration
C1RXF0EID = 0xFFFF;

C1RXF1SID = 0xFFEB;       // Filter 1 configuration
C1RXF1EID = 0xFFFF;

C1FMSKSEL1bits.F0MSK = 0; // Mask 0 used for both filters
C1FMSKSEL1bits.F1MSK = 0; // Mask 0 used for both filters
C1RXM0SID = 0xFFEB;
C1RXM0EID = 0xFFFF;

C1FCTRLbits.DMABS = 3;    // 12 buffers in DMA RAM
C1FCTRLbits.FSA   = 3;    // FIFO starts from TX/RX Buffer 3

C1CTRL1bits.WIN     = 0;
C1TR01CONbits.TXEN0 = 0;  // Buffer 0 is a receive buffer
C1TR23CONbits.TXEN2 = 0;  // Buffer 2 is a receive buffer

C1TR01CONbits.TX0PRI = 0b11;//High Priority
C1TR01CONbits.TX1PRI = 0b10;//Intermediate High Priority

C1CTRL1bits.REQOP = 0;// Enable Normal Operation Mode
```

DMA チャンネル 0 を周辺モジュール間接アドレッシング モードに設定する :

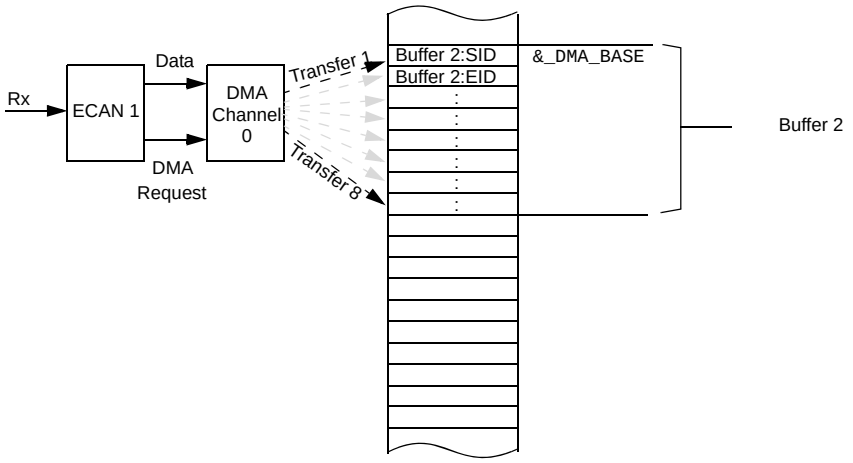
```
unsigned int Ecan1Rx[12][8] __attribute__((space(dma)));// 12 buffers, 8
words each

DMA0CONbits.AMODE = 2;    // Continuous mode, single buffer
DMA0CONbits.MODE = 0;     // Peripheral Indirect Addressing

DMA0PAD = (volatile unsigned int) &C1RXD;// Point to ECAN1 Rx register
DMA0STA = __builtin_dmaoffset(Ecan1Rx);// Point DMA to ECAN1 buffers

DMA0CNT = 7;              // 8 DMA request (1 buffer, each with 8 words)
DMA0REQ = 0x22;           // Select ECAN1 Rx as DMA Request source

IEC0bits.DMA0IE = 1;      // Enable DMA Channel 0 interrupt
DMA0CONbits.CHEN = 1;     // Enable DMA Channel 0
```

DMA 割り込みハンドラを設定する :

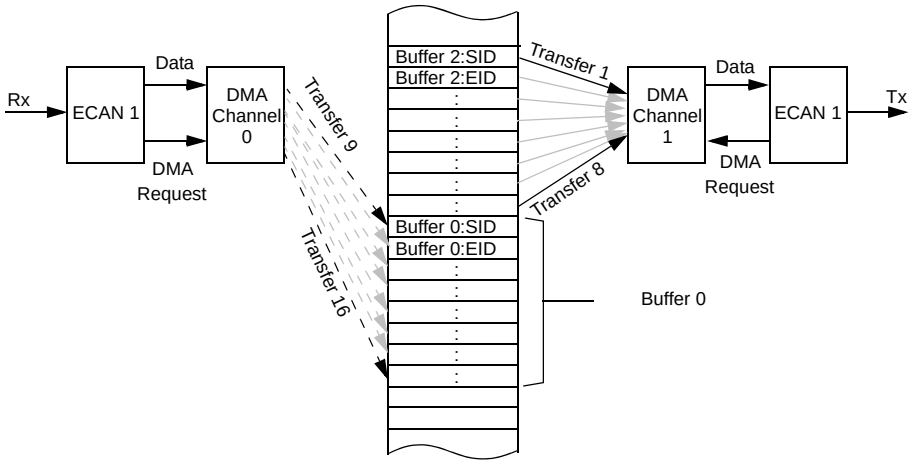
```
void __attribute__((interrupt, no_auto_psv)) _DMA0Interrupt(void)
{
   ProcessData(Ecan1Rx[C1VECbits.ICODE]);// Process received buffer;

   IFS0bits.DMA0IF = 0;  // Clear the DMA0 Interrupt Flag;
}
```

図 38-14: レジスタ間接アドレッシングによる ECAN™ からのデータ転送

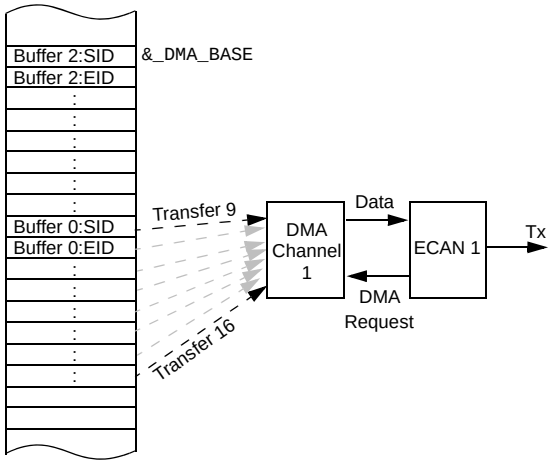
A バッファ2 に受信



B バッファ0 に受信、バッファ2 を送信



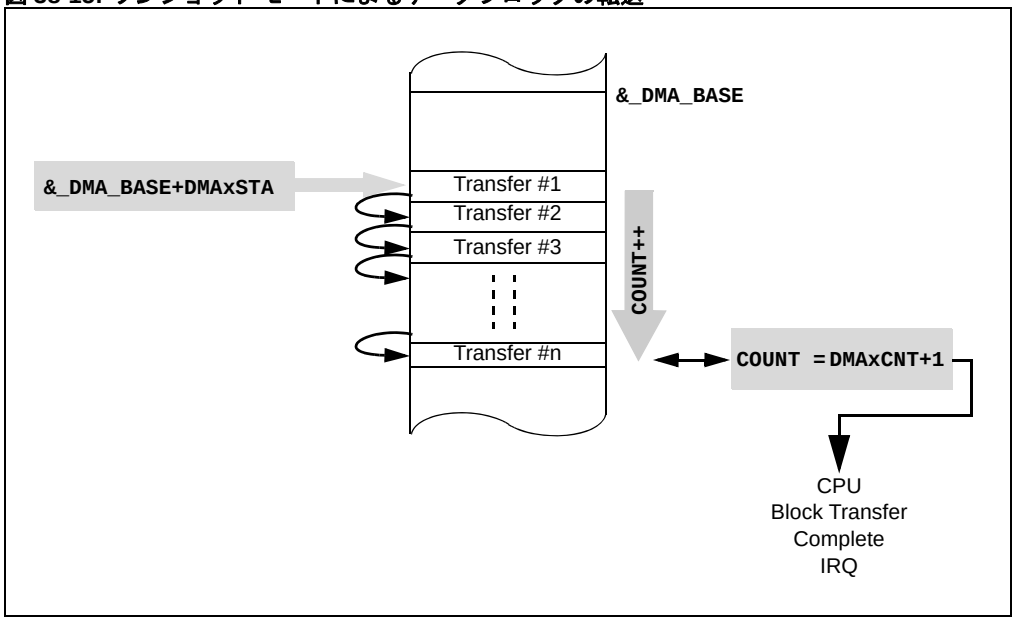
C バッファ0 を送信



38.6.7 ワンショット モード

データ転送を繰り返す必要がない場合、アプリケーション プログラムはワンショット モードを使用できます。DMA チャンネル制御 (DMAxCON) レジスタの動作モード選択ビット (MODE<1:0>) を「x1」に設定すると、ワンショット モードが選択されます。このモードでは、データブロック (ブロック長は DMAxCNT で定義) が完全に転送されてブロック終端が検出された時点で、そのチャンネルが自動的に無効化されます (ハードウェアが DMA チャンネル制御 (DMAxCON) レジスタの CHEN ビットをクリアします)。図 38-15 にワンショット モードの動作を示します。

図 38-15: ワンショット モードによるデータブロックの転送



DMA チャンネル制御 (DMAxCON) レジスタの HALF ビットがセットされている場合、データブロックの半分 (ハーフブロック) が転送された時点で DMAxIF ビットがセットされ、DMA 割り込みが生成されます (その割り込みがアプリケーション プログラムで有効化されている場合のみ)。この時点では、チャンネル無効化されません。ブロック全体 (フルブロック) の転送が完了した時、割り込みフラグはセットされず、チャンネルは自動的に無効化されます。DMA チャンネルをハーフブロック転送またはフルブロック転送割り込みに設定する方法は、**38.6.3「フルブロックまたはハーフブロック転送割り込み」**を参照してください。

DMAxCON レジスタの CHEN ビットを「1」にセットしてそのチャンネルを再度有効化した場合、DPSRAM 開始アドレス オフセット (DMAxSTA、DMAxSTB) レジスタに格納されている開始アドレスからブロック転送が開始されます。例 38- 10 に、ワンショット動作用のサンプルコードを示します。

例 38- 10: ワンショット モードによる UART データの DMA 転送

UART を Rx および Tx 用に設定する :

```
#define FCY      40000000
#define BAUDRATE 9600
#define BRGVAL   ((FCY/BAUDRATE)/16)-1

U2MODEbits.STSEL = 0;  // 1-stop bit
U2MODEbits.PDSEL = 0;  // No Parity, 8-data bits
U2MODEbits.ABAUD = 0;  // Autobaud Disabled

U2BRG = BRGVAL;// BAUD Rate Setting for 9600

U2STAbits.UTXISEL0 = 0;// Interrupt after one Tx character is transmitted
U2STAbits.UTXISEL1 = 0;
U2STAbits.URXISEL = 0; // Interrupt after one RX character is received

U2MODEbits.UARTEN   = 1;// Enable UART
U2STAbits.UTXEN     = 1;// Enable UART Tx
```

例 38-10: ワンショット モードによる UART データの DMA 転送 (続き)

DMA チャンネル 0 をワンショット / 単一バッファモードの送信用に設定する :

```
unsigned int BufferA[8] __attribute__((space(dma)));
unsigned int BufferB[8] __attribute__((space(dma)));


DMA0CON = 0x2001;         // One-Shot, Post-Increment, RAM-to-Peripheral
DMA0CNT = 7;              // 8 DMA requests
DMA0REQ = 0x001F;         // Select UART2 Transmitter

DMA0PAD = (volatile unsigned int) &U2TXREG;
DMA0STA = __builtin_dmaoffset(BufferA);

IFS0bits.DMA0IF = 0;      // Clear DMA Interrupt Flag
IEC0bits.DMA0IE = 1;      // Enable DMA interrupt
```

DMA チャンネル 1 を連続 / ピンポンモードの受信用に設定する :

```
DMA1CON = 0x0002;         // Continuous, Ping-Pong, Post-Inc., Periph-RAM
DMA1CNT = 7;              // 8 DMA requests
DMA1REQ = 0x001E;         // Select UART2 Receiver

DMA1PAD = (volatile unsigned int) &U2RXREG;
DMA1STA = __builtin_dmaoffset(BufferA);
DMA1STB = __builtin_dmaoffset(BufferB);


IFS0bits.DMA1IF = 0;      // Clear DMA interrupt
IEC0bits.DMA1IE = 1;      // Enable DMA interrupt
DMA1CONbits.CHEN = 1;     // Enable DMA Channel
```

DMA 割り込みハンドラを設定する :

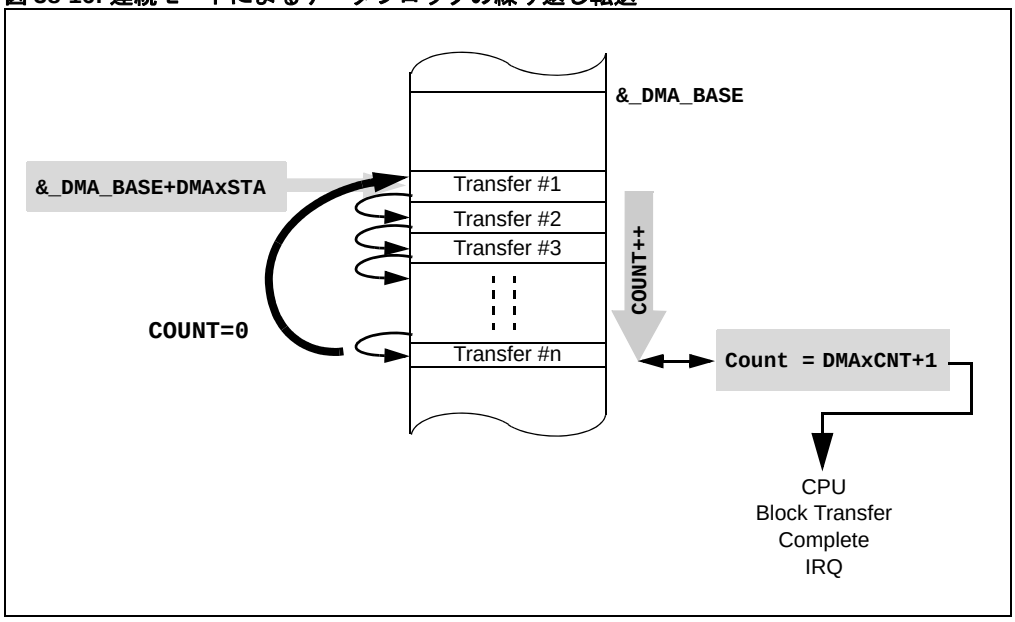
```
void __attribute__((interrupt, no_auto_psv)) _DMA0Interrupt(void)
{
   IFS0bits.DMA0IF = 0;  // Clear the DMA0 Interrupt Flag;
}


void __attribute__((interrupt, no_auto_psv)) _DMA1Interrupt(void)
{
   static unsigned int BufferCount = 0; // Keep record of which buffer
                                        // contains Rx Data

   if(BufferCount == 0)
   {
 DMA0STA = __builtin_dmaoffset(BufferA);// Point DMA 0 to data
                                        // to be transmitted
   }
   else
 {
 DMA0STA = __builtin_dmaoffset(BufferB); // Point DMA 0 to data
                                        // to be transmitted
   }

   DMA0CONbits.CHEN = 1;  // Enable DMA0 Channel
   DMA0REQbits.FORCE = 1; // Manual mode:Kick-start the 1st transfer

   BufferCount ^= 1;
   IFS0bits.DMA1IF = 0;  // Clear the DMA1 Interrupt Flag
}
```

38.6.8 連続モード

プログラムが動作する間、転送を繰り返し実行する必要がある場合、アプリケーション プログラムは連続モードを使用できます。

DMA チャンネル制御 (DMAxCON) レジスタの動作モード選択ビット (MODE<1:0>) を「x0」に設定すると、連続モードが選択されます。このモードでは、データ ブロック (ブロック長は DMAxCNT で定義) が完全に転送されてブロック終端が検出されてもチャンネルは無効化されません。ブロック最後のデータを転送中に、DMA DPSRAM アドレスはプライマリ DPSRAM 開始アドレス オフセット A(DMAxSTA) レジスタの値にリセットされます。図 38-16 に連続モードの動作を示します。

図 38-16: 連続モードによるデータ ブロックの繰り返し転送



DMA チャンネル制御 (DMAxCON) レジスタの HALF ビットがセットされている場合、データ ブロックの半分 (ハーフブロック) が転送された時点で DMAxIF ビットがセットされ、DMA 割り込みが生成されます (その割り込みがアプリケーション プログラムで有効化されている場合のみ)。この時、チャンネルは無効化されません。ブロック全体 (フルブロック) の転送が完了した時、割り込みフラグはセットされず、チャンネルは無効化されません。DMA チャンネルをハーフブロック転送またはフルブロック転送割り込みに設定する方法は、**38.6.3「フルブロックまたはハーフブロック転送割り込み」**を参照してください。

38.6.9 ピンポンモード

ピンポンモードでは 2 つのバッファを使用し、DMA チャンネルが一方のバッファを使用している間に、CPU は他方のバッファを処理できます。この結果、CPU は、DMA が一方のバッファでブロック転送を完了するまでの時間を使用して、他方のバッファのデータを処理できます。当然ですが、この転送モードは、所定サイズのバッファリングに他のモードの 2 倍の DPSRAM 領域を使用します。

全ての DMA 動作モードでは、DMA チャンネルを有効化した時に、DMA チャンネル x DPSRAM プライマリ開始アドレス オフセット A (DMAxSTA) レジスタが既定値とて選択され、これにより初期の DSPRAM 実効アドレスが生成されます。1 ブロックの転送が完了して DMA チャンネルが再初期化されるたびに、バッファ開始アドレスは同じ DMAxSTA レジスタから読み出されます。

これに対しピンポンモードでは、バッファ開始アドレスが下記の 2 つのレジスタから読み出されます。

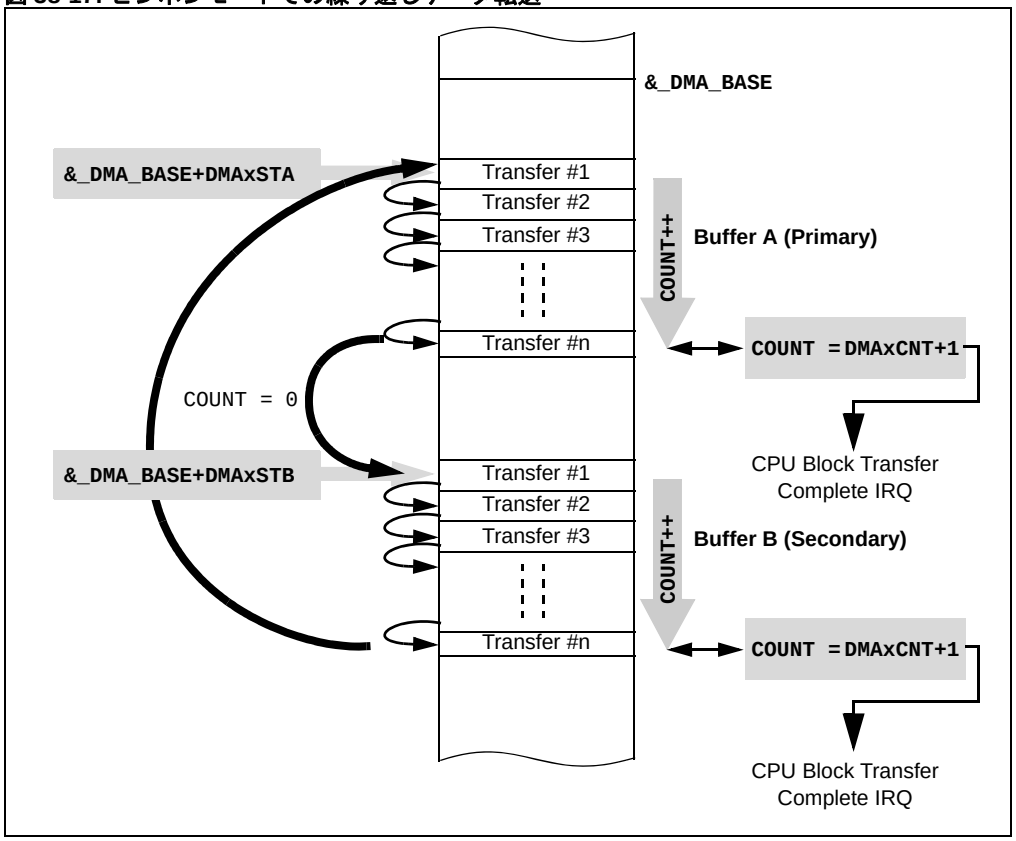
- プライマリ : DMA チャンネル x DPSRAM 開始アドレス オフセット A (DMAxSTA) レジスタ
- セカンダリ : DMA チャンネル x DPSRAM 開始アドレス オフセット B (DMAxSTB) レジスタ

DMA は、プライマリ バッファとセカンダリ バッファをブロック転送ごとに交互に使用します。1 ブロックの転送が完了して DMA チャンネルが再初期化されるたびに、バッファ開始アドレスは他方のレジスタから読み出されます。

DMA チャンネル制御 (DMAxCON) レジスタの動作モード選択ビット (MODE<1:0>) を「1x」に設定すると、ピンポンモードが選択されます。

連続モードの DMA でピンポンモードを有効化した場合、DMA はプライマリ バッファを使用して転送した後、再初期化時にセカンダリ バッファを選択して次の転送を実行します。以降のブロック転送では、プライマリ バッファとセカンダリ バッファを交互に使用します。各バッファの転送後に割り込みが生成されます (割り込みがアプリケーション プログラムで有効化されている場合のみ)。図 38-17 に、ピンポンモード有効時の連続モード動作を示します。例 38- 11 に、DCI モジュールを使用したピンポンモード動作用のサンプルコードを示します。

図 38-17: ピンポンモードでの繰り返しデータ転送



例 38- 11: 連続ピンポンモードによる DCI データの DMA 転送

DCI を Rx および Tx 用に設定する :

```
#define FCY 40000000
#define FS 48000
#define FCSCK64*FS
#define BCGVAL(FCY/(2*FS))-1

DCICON1bits.CSCKD = 0;// Serial Bit Clock (CSCK pin) is output
DCICON1bits.CSCKE = 0;// Data sampled on falling edge of CSCK
DCICON1bits.COFSD = 0;// Frame Sync Signal is output
DCICON1bits.UNFM = 0;// Transmit '0's on a transmit underflow
DCICON1bits.CSDOM = 0;// CSDO pin drives '0's during disabled TX time slots
DCICON1bits.DJST = 0;// TX/RX starts 1 serial clock cycle after frame sync
pulse
DCICON1bits.COFSM = 1;// Frame Sync Signal set up for I2S mode

DCICON2bits.BLEN = 0;// One data word will be buffered between interrupts
DCICON2bits.COFSG = 1;// Data frame has 2 words:LEFT & RIGHT samples
DCICON2bits.WS = 15;// Data word size is 16 bits

DCICON3 = BCG_VAL;// Set up CSCK Bit Clock Frequency

TSCONbits.TSE0 = 1;    // Transmit on Time Slot 0
TSCONbits.TSE1 = 1;    // Transmit on Time Slot 1
TSCONbits.TSE0 = 1;    // Transmit on Time Slot 0
RSCONbits.RSE1 = 1;    // Receive on Time Slot 1
```

DMA チャンネル 0 を連続ピンポンモードで送信用に設定する :

```
unsigned int TxBufferA[16] __attribute__((space(dma)));
unsigned int TxBufferB[16] __attribute__((space(dma)));

DMA0CON = 0x2002;  // Ping-Pong, Continous, Post-Increment, RAM-to-Peripheral
DMA0CNT = 15;      // 15 DMA requests
DMA0REQ = 0x003C;  // Select DCI as DMA Request source

DMA0PAD = (volatile unsigned int) &TXBUF0;
DMA0STA = __builtin_dmaoffset(TxBufferA);
DMA0STB = __builtin_dmaoffset(TxBufferB);

IFS0bits.DMA0IF = 0;// Clear DMA Interrupt Flag
IEC0bits.DMA0IE = 1;// Enable DMA interrupt
DMA0CONbits.CHEN = 1;// Enable DMA Channel
```

DMA チャンネル 1 を連続ピンポンモードで受信用に設定する :

```
unsigned int RxBufferA[16] __attribute__((space(dma)));
unsigned int RxBufferB[16] __attribute__((space(dma)));


DMA1CON = 0x0002;  // Continuous, Ping-Pong, Post-Inc., Periph-RAM
DMA1CNT = 15;      // 16 DMA requests
DMA1REQ = 0x003C;  // Select DCI as DMA Request source

DMA1PAD = (volatile unsigned int) &RXBUF0;
DMA1STA = __builtin_dmaoffset(RxBufferA);
DMA1STB = __builtin_dmaoffset(RxBufferB);

IFS0bits.DMA1IF = 0;// Clear DMA interrupt
IEC0bits.DMA1IE = 1;// Enable DMA interrupt
```

例 38-11: 連続ピンポンモードによる DCI データの DMA 転送 (続き)

DMA 割り込みハンドラを設定する :

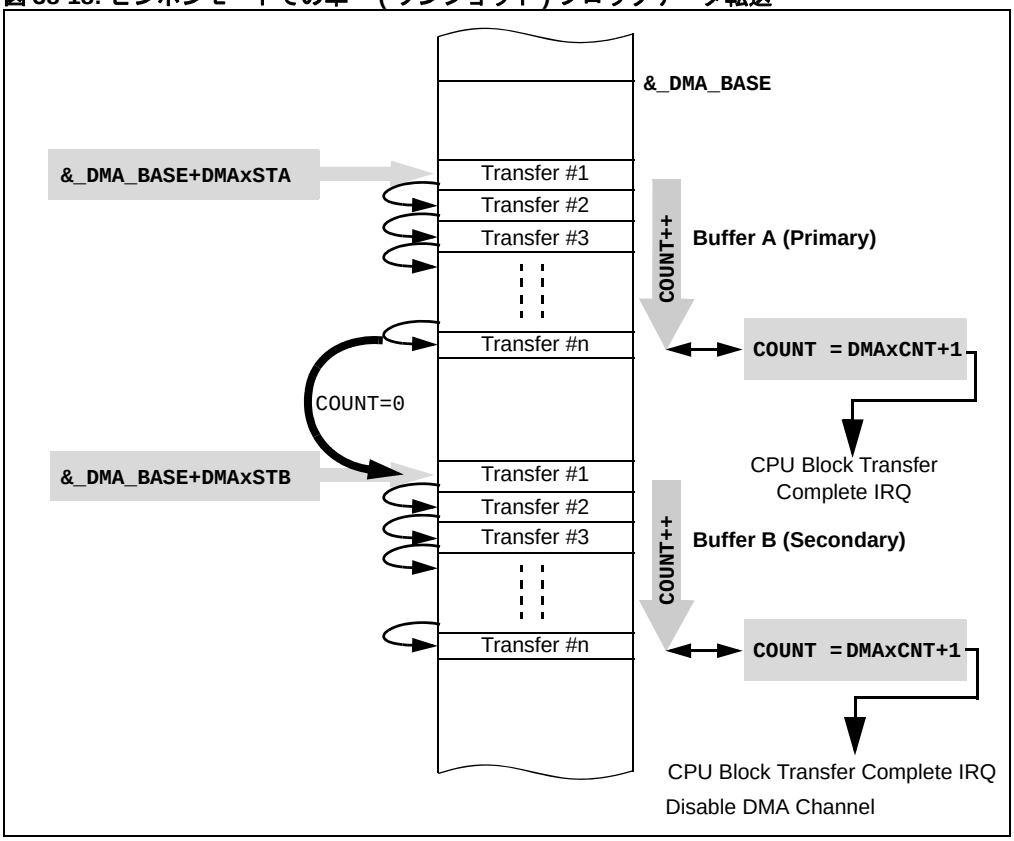
```
void __attribute__((interrupt, no_auto_psv)) _DMA0Interrupt(void)
{
   static unsigned int TxBufferCount = 0;// Keep record of which buffer
                                  // has Rx Data

   if(BufferCount == 0)
   {
 /* Notify application that TxBufferA has been transmitted */
   }
   else
   {
 /* Notify application that TxBufferB has been transmitted */
   }

   BufferCount ^= 1;
   IFS0bits.DMA0IF = 0;           // Clear the DMA0 Interrupt Flag;
}


void __attribute__((interrupt, no_auto_psv)) _DMA1Interrupt(void)
{
   static unsigned int RxBufferCount = 0;// Keep record of which buffer
                                  // has Rx Data

   if(BufferCount == 0)
   {
 /* Notify application that RxBufferA has been received */
   }
   else
   {
 /* Notify application that RxBufferB has been received */   }

   BufferCount ^= 1;
   IFS0bits.DMA1IF = 0;           // Clear the DMA1 Interrupt Flag
}
```

DCI を有効化する :

```
/* Force First two words to fill-in Tx buffer/shift register */
DMA0REQbits.FORCE = 1;
while(DMA0REQbits.FORCE == 1);

DMA0REQbits.FORCE = 1;
while(DMA0REQbits.FORCE == 1);

DCICON1bits.DCIEN = 1;            // Enable DCI
```

ワンショット モードの DMA でピンポンモードを有効化すると、DMA はプライマリ バッファを使用して転送した後、再初期化時にセカンダリ バッファを選択して次の転送を実行します。その後 DMA チャンネルは無効化されるため、以降の転送は発生しません。図 38-18 に、ピンポンモード有効時のワンショット データ転送の動作を示します。

図 38-18: ピンポンモードでの単一 (ワンショット) ブロックデータ転送



38.6.10 手動転送モード

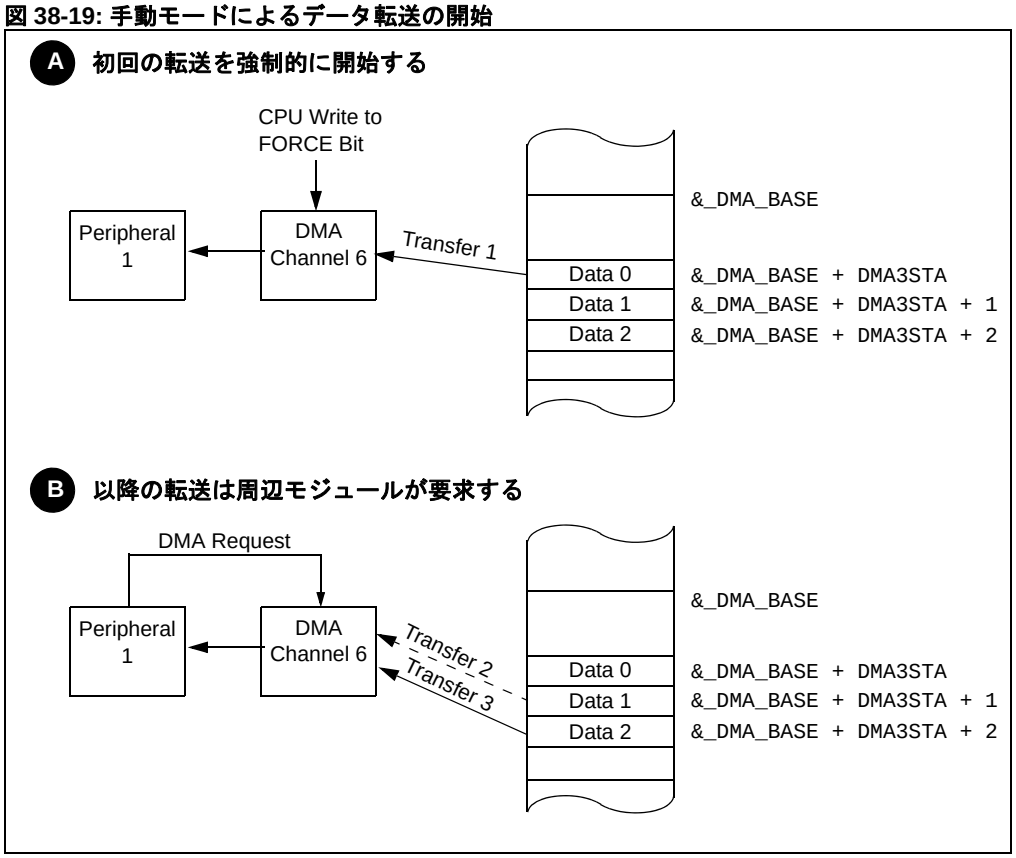
DMA コントローラを使用して周辺モジュールから DPSRAM にデータを送信する場合、DMA 転送は DMA チャンネルと周辺モジュールの初期化後自動的に開始されます。周辺モジュールは、送信データの準備を完了した時に DMA 要求を生成します。加えて、DPSRAM バッファから周辺モジュールへもデータを送信する必要がある場合、同じ DMA 要求を使用して別の 1 チャンネル (DPSRAM からデータを読みして周辺モジュールに書き込むためのチャンネル) も有効化できます。

逆に DPSRAM バッファから周辺モジュールへの単方向のデータ送信だけが必要な場合、処理を開始するには、手動で最初のデータを周辺モジュールに送信する必要があります (**38.7「DMA 転送の開始」**参照)。このプロセスはソフトウェアで開始できますが、より便利な方法として、DMA チャンネル レジスタ内の 1 ビットをセットするだけで、そのチャンネルの DMA 要求を簡単に模倣できます (手動 DMA 要求)。DMA チャンネルは、模倣された DMA 要求を通常の要求と同様に処理し、最初のデータ要素を転送してシーケンスを開始します。周辺モジュールは、次のデータの受信準備を完了した時に通常の DMA 要求を送信し、これを受けた DMA は次のデータ要素を送信します。このプロセスを図 38-19 に示します。

手動 DMA 要求は、DMA チャンネル x IRQ 選択 (DMAxREQ) レジスタの FORCE ビットをセットする事により生成できます。1 度セットした FORCE ビットをユーザ アプリケーションでクリアする事はできません。FORCE ビットは、手動 DMA 転送が完了した時にハードウェアによってクリアされます。FORCE ビットをセットするタイミングによっては、下記の特殊な条件が適用されます。

- DMA 転送実行中に FORCE ビットをセットした場合、何も効果を持たず無視されます。
- チャンネル x の設定時に FORCE ビットをセットした場合 (DMA チャンネルを設定するためのレジスタへの書き込みで同時に FORCE ビットもセットした場合)、予測不能な挙動が生じます。従って、このような操作を避ける必要があります。
- DMA チャンネルの周辺モジュール割り込み要求が保留されている時、そのチャンネルの FORCE ビットのセットを試みても無視され、割り込みによる要求が優先されます。しかしこの場合、DMA コントローラ ステータス 0 (DMACS0) レジスタの DMA RAM 書き込みコリジョン フラグビット (XWCOLx) と周辺モジュール書き込みコリジョン フラグビット (PWCOLx) の両方がセットされてエラー条件が発生します。詳細は **38.10「データ書き込みコリジョン」**を参照してください。

図 38-19: 手動モードによるデータ転送の開始



38.6.11 NULL データ書き込みモード

NULL データ書き込みモードは、SPI のようにデータを一切送信せずにシーケンシャルなデータ受信だけを必要とするアプリケーションで非常に効果的に使用できます。

基本的に SPI は単純なシフトレジスタであり、1 クロック周期あたり 1 ビットのデータを出し入れします。受信データだけが必要であるにも関わらず SPI をマスタモード (SPI がクロックの供給源) に設定すると、特殊な状況が発生します。すなわち、SPI データクロックを起動して外部データを受信するために、SPI データレジスタに何らかのデータを書き込む必要があります。

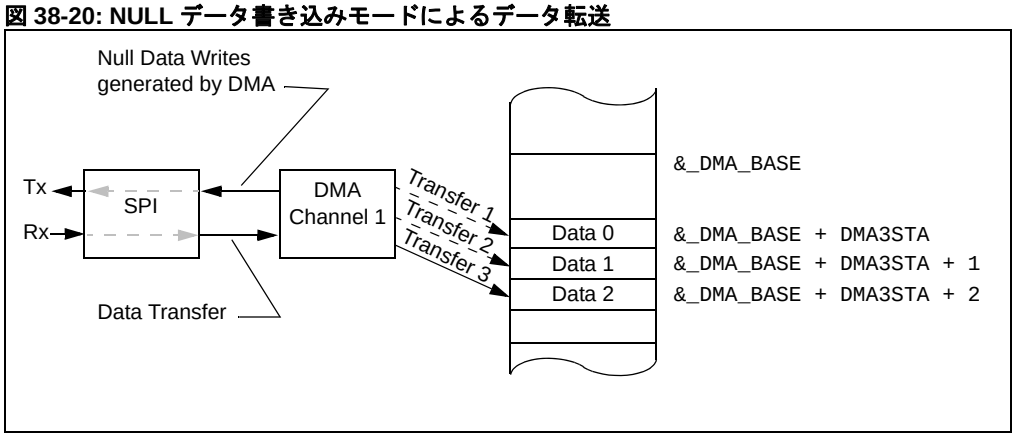
このため、2 つの DMA チャンネルを割り当てて、1 チャンネルをデータ受信用に使用し、もう 1 チャンネルを NULL またはゼロデータを SPI に送信するためだけに使用する事もできます。しかし、DMA NULL データ書き込みモードを使用すると、より簡単に NULL データを送信できます。このモードでは、周辺モジュール データ読み出し用に設定された DMA チャンネルがデータ要素を受信して移動するたびに、SPI データレジスタに NULL 値が自動的に書き込まれます。

DMA チャンネル x 制御 (DMAxCON) レジスタの NULL データ周辺モジュール書き込みモード選択ビット (NULLW) がセットされ、かつ DMA チャンネルが周辺モジュールからの読み出し用に設定されている場合、DMA チャンネルは 周辺モジュールデータの読み出しサイクル中に NULL データ (全ゼロデータ) を周辺モジュール アドレスに書き込みます。この書き込みは周辺モジュールバスを使用し、DPSRAM バスによる DPSRAM へのデータ書き込みと同時に発生します。図 38-20 に NULL データ書き込みモードの動作を示します。

通常このモードでは、周辺モジュール DMA 要求への応答時 (データを受信して移動可能となった時) にのみ NULL データ書き込みが発生します。最初のワードの受信を開始するために、CPU から周辺モジュールへの初期書き込みが必要です。その後、DMA が後続の全ての周辺モジュールへの NULL データ書き込みを処理します。つまり、CPU による NULL 書き込みによって SPI (マスタ) のデータ送受信が開始され、その結果、新たに受信したデータを転送するために DMA 要求が生成されます。

強制 (手動)DMA 転送を使用してこのプロセスを開始する事もできます。しかしこの方法では、冗長な周辺モジュール読み出し (データ無効) と関連 DPSRAM ポインタの調整が必要である事に注意が必要です。

図 38-20: NULL データ書き込みモードによるデータ転送



例 38- 12: NULL データ書き込みモードによる SPI データの DMA 転送

SPI をマスタモードに設定する :

```
SPI1CON1bits.MODE16 = 1; //Communication is word-wide (16 bits)
SPI1CON1bits.MSTEN = 1;  //Master Mode Enabled
SPI1STATbits.SPIEN = 1;  //Enable SPI Module
```

DMA チャンネル 1 を NULL データ書き込みモードに設定する :

```
unsigned int BufferA[16] __attribute__((space(dma)));
unsigned int BufferB[16] __attribute__((space(dma)));

DMA1CON = 0x0802;         // Null Write, Continuous, Ping-Pong,
                          // Post-Increment, Periph-to-RAM
DMA1CNT = 15;             // Transfer 16 words at a time
DMA1REQ = 0x000A;         // Select SPI1 as DMA request source

DMA1STA = __builtin_dmaoffset(BufferA);
DMA1STB = __builtin_dmaoffset(BufferB);
DMA1PAD = (volatile unsigned int) &SPI1BUF;

IFS0bits.DMA1IF = 0;
IEC0bits.DMA1IE = 1;      // Enable DMA interrupt
DMA1CONbits.CHEN = 1;     // Enable DMA Channel

DMA1REQbits.FORCE = 1;    // Force First word after Enabling SPI
```

DMA 割り込みハンドラを設定する :

```
void __attribute__((interrupt, no_auto_psv)) _DMA1Interrupt(void)
{
   static unsigned int BufferCount = 0;// Keep record of which buffer
                                       // contains Rx Data

   if(BufferCount == 0)
   {
 ProcessRxData(BufferA);          // Process received SPI data in
                                  // DMA RAM Primary buffer
   }
   else
   {
 ProcessRxData(BufferB);          // Process received SPI data in
                                  // DMA RAM Secondary buffer
   }

   BufferCount ^= 1;
   IFS0bits.DMA1IF = 0;           // Clear the DMA1 Interrupt Flag
}
```

38.7 DMA 転送の開始

DMA 転送を開始する前に、DMAxCON レジスタの CHEN ビットを「1」にセットして、そのDMA チャンネルを有効化する必要があります。DMA チャンネルが既に有効化されている場合、そのチャンネルをまず無効化 (CHEN = 0) してから再度有効化 (CHEN = 1) する事によって、そのチャンネルを再初期化できます。この操作により、DMA 転送カウントはゼロにリセットされ、DMA バッファにはプライマリバッファが選択されます。

DMA チャンネルと周辺モジュールが正しく初期化されていれば、周辺モジュールがデータ転送準備を完了して DMA 要求を生成した時点で、DMA 転送が開始されます。ただし、一部の周辺モジュールは、一定の条件が成立するまで DMA 要求を生成しません (従って DMA 転送は開始されません)。このような場合、DMA 転送を開始するには、以下で説明する各種の DMA モードとプロセスの組み合わせを適用する必要があります。

38.7.1 シリアル ペリフェラル インターフェイス (SPI) との DMA 転送の開始

SPI 周辺モジュールとの DMA 転送の開始方法は、SPI データの転送方向とスレーブ / マスタのどちらのモードで転送するかによって異なります。

- **マスタモード /Tx のみ**

  このコンフィグレーションでは、SPI データの先頭ブロックが送信されるまで DMA 要求は生成されません。DMA 転送を開始するには、ユーザ アプリケーションで DMA 手動転送モードを使用して最初にデータを送信するか、あるいは DMA を使用せずに最初に SPI バッファ (SPIxBUF) にデータを書き込む必要があります。
- **マスタモード /Rx のみ**

  このコンフィグレーションでは、SPI データの先頭ブロックが受信されるまで DMA 要求は生成されません。しかしマスタモードでは SPI が先に送信するまでデータは受信されないため、ユーザ アプリケーションで DMA NULL データ書き込みモードを使用して DMA 手動転送モードを開始する必要があります。
- **マスタモード /Rx と Tx**

  このコンフィグレーションでは、SPI データの先頭ブロックが受信されるまで DMA 要求は生成されません。しかしマスタモードでは SPI が先に送信するまでデータは受信されないため、ユーザ アプリケーションで DMA 手動転送モードを使用して最初にデータを送信するか、あるいは DMA を使用せずに最初に SPI バッファ (SPIxBUF) にデータを書き込む必要があります。
- **スレーブモード /Tx のみ**

  このコンフィグレーションでは、SPI データの先頭ブロックが受信されるまで DMA 要求は生成されません。DMA 転送を開始するには、ユーザ アプリケーションで DMA 手動転送モードを使用して最初にデータを送信するか、あるいは DMA を使用せずに最初に SPI バッファ (SPIxBUF) にデータを書き込む必要があります。
- **スレーブモード /Rx のみ**

  このコンフィグレーションでは、最初の SPI データが到着すると即座に DMA 要求が生成されます。従って、DMA 転送を開始するための特別な手順は不要です。
- **スレーブモード /Rx と Tx**

  このコンフィグレーションでは、先頭の SPI データブロックが受信されるまで DMA 要求は生成されません。DMA 転送を開始するには、ユーザ アプリケーションで DMA 手動転送モードを使用して最初にデータを送信するか、あるいは DMA を使用せずに最初に SPI バッファ (SPIxBUF) にデータを書き込む必要があります。

38.7.2 データコンバータ インターフェイス (DCI) との DMA 転送の開始

他のシリアル ペリフェラルとは異なり、DCI は有効化されると即座に転送を開始します (DCI がマスタである場合)。DCI は、接続先の外部コーデックに向けてデータの同期フレームを常時供給します。DCI を有効化する前に、下記の操作が必要です。

- **38.5.2「周辺モジュール コンフィグレーションのセットアップ」**の説明に従って DCI を設定する
- ステレオ コーデックに接続する場合、最初の 2 個のデータの転送を開始するために、下記のように DMA 手動転送モードを使用します。
  - DMAxREQ レジスタの FORCE ビットをセットして DCI の左チャンネル サンプルを転送する
  - 再度 FORCE ビットをセットして DCI の右チャンネル サンプルを転送する

上記を実行した後に、DCI 周辺モジュールを有効化します (例 38- 11 参照)。

38.7.3 UART との DMA 転送の開始

UART レシーバは、データを受信すると即座に DMA 要求を生成します。DMA 転送を開始するためにユーザ アプリケーションによる特別な手順は不要です。UART とトランスミッタを有効化すると、即座に UART トランスミッタが DMA 要求を生成します。従って、UART とトランスミッタを有効化する前に、DMA チャンネルとバッファを初期化および有効化しておく必要があります。

UART は、**38.5.2「周辺モジュール コンフィグレーションのセットアップ」**(表 38-2) の説明に従って設定する必要があります。

別の手順として、DMA チャンネルを有効化する前に、UART と UART トランスミッタを有効化する事もできます。この場合、UART トランスミッタの DMA 要求は無効となるため、DMA 転送を開始するには、ユーザ アプリケーションで DMAxREQ レジスタの FORCE ビットをセットして DMA 要求を生成する必要があります。

38.8 DMA チャンネルの調停と オーバーラン

各 DMA チャンネルの優先度は、チャンネル番号順に固定されています (チャンネル 0 の優先度が最高、チャンネル 7 の優先度が最低)。DMA 転送要求は、その要求源に割り当てられた DMA チャンネルによってラッチされます。DMA コントローラはアービトレータとして動作します。実行中または保留中の転送が他に存在しない場合、コントローラは要求中の DMA チャンネルに対してバスリソースの使用を承認します。DMA コントローラは、バスリソースを使用中の DMA チャンネルが動作を完了するまで、そのリソースを他の DMA チャンネルに一切使用させません。

複数の DMA 要求を同時に受け取った場合、または複数の DMA 要求が保留中である場合、DMA コントローラの優先度ロジックは、その中で最も優先度の高い DMA チャンネルにリソースの使用を承認します。他の全ての DMA 要求は、最優先の DMA 転送が完了するまで保留されます。DMA 転送の実行中に新たな DMA 要求が届いた場合、その要求も既に保留中の DMA 要求に加えた上で優先度が再評価されます。これにより、常に最優先要求が実行中の DMA 転送完了後に処理されます。

このように、DMA チャンネルは優先度に基づいて処理されるため、DMA 要求は即座に処理されず保留される可能性があります。優先度の低い要求は、より高優先度のチャンネルが全て処理されるまで保留されます。DMA コントローラが先に受け取った DMA 要求をクリアする前に別の割り込みを受け取り、その割り込みが保留中の割り込みと同タイプであった場合、データ オーバーランが発生します。

データ オーバーランは、DMA が先のデータを移動し終える前に、周辺モジュール データバッファに新たなデータが届いた状態として定義されます。一部の DMA 対応周辺モジュールは、データ オーバーランを検出して CPU 割り込みを生成できます (その周辺モジュール エラー割り込みが有効化されている場合のみ)。表 38-5 を参照してください。

表 38-5: DMA 対応周辺モジュールによるオーバーラン処理

DMA 対応周辺モジュール	データ オーバーラン処理
シリアル ペリフェラル インターフェイス (SPI)	DMA チャンネルがまだ移動し終えていないデータは、後続の受信データによって上書きされません。後続の受信データは破棄され、SPI ステータス (SPIxSTAT) レジスタの SPI 受信オーバーフロー(SPIROV) ビットがセットされます。この時、割り込みコントローラで割り込みイネーブル制御 (IECx) レジスタの SPI エラー割り込みイネーブル (SPIxEIE) ビットがセットされていれば、SPIx フォルト割り込みが生成されます。
UART	DMA チャンネルがまだ移動し終えていないデータは、後続の受信データによって上書きされません。後続の受信データは破棄され、UART ステータス (UxSTA) レジスタのオーバーフロー エラー (OERR) ビットがセットされます。この時、割り込みコントローラで割り込みイネーブル制御 (IECx) レジスタの UART エラー割り込みイネーブル (UxEIE) ビットがセットされていれば、UARTx エラー割り込みが生成されます。
データコンバータ インターフェイス (DCI)	DMA チャンネルがまだ移動し終えていないデータは、後続の受信データによって上書きされ、DCI ステータス (DCISTAT) レジスタの受信オーバーフロー (ROV) ビットがセットされます。この時、割り込みコントローラで割り込みイネーブル制御 (IECx) レジスタの DCI エラー割り込みイネーブル (DCIEIE) ビットがセットされていれば、DCI フォルト割り込みが生成されます。
10/12 ビット アナログ / デジタル コンバータ (ADC)	DMA チャンネルがまだ移動し終えていないデータは、後続の受信データによって上書きされます。ADC はオーバーラン条件を検出しません。
その他の DMA 対応 周辺モジュール	データ オーバーランは発生しません。

DMA コントローラが周辺モジュールから DPSRAM へデータを移動中である場合にのみ、データ オーバーランを検出できます。バッファ エンプティ割り込み等に基づく DPSRAM から周辺モジュールへの DMA データ転送は常に実行されます。結果として生じる DPSRAM データ オーバーランは、ソフトウェアを使用して検出する必要があります。重複した DMA 要求は無視され、保留中の要求はそのまま保留されます。通常と同様に、DMA チャンネルは転送が完了した時に DMA 要求をクリアします。この際に CPU が介入しないと、その転送データが最新 (オーバーラン) データとなり、先のデータは失われます。

ユーザ アプリケーションは、データ ソースの性質に応じて複数の方法でオーバーラン エラーを処理できます。DMAC のデータ リカバリとそのデータ ソース / シンクとの同期再確立は、アプリケーションに大きく依存するタスクです。ストリーミングデータ (DCI 周辺モジュール経由のコーデックからのデータ等) の場合、アプリケーションは失われたデータを無視できます。問題の原因を修正した後 (可能な場合)、DMA 割り込みハンドラはデータが再度正しくバッファされるように DMAC と DCI の同期再確立を試みます。ユーザ アプリケーションは、さらにオーバーランが発生するのを防ぐため迅速に対応する必要があります。

周辺モジュール オーバーラン割り込みに移行するまでに、保留状態の DMA 要求は、失われたデータが移動されるはずであったアドレスにオーバーラン データ値の移動を完了しています。そのデータを正しいアドレスに移動し、欠損データ スロットに NULL データ値を挿入する事ができます。その後、チャンネルの DPSRAM アドレスを適切に調整できます。エラーのあったチャンネルに対する後続の DMA 要求は、修正済み DPSRAM アドレスに対して通常通り転送を開始します。データ損失を許容できないアプリケーションの場合、周辺モジュール オーバーラン割り込みによってデータが失われる前に現在のブロック転送を中止し、DMA チャンネルを再初期化し、データの再送を要求する必要があります。

38.9 デバッグのサポート

デバッグ中の DMA 動作の監視を容易にするために、DMA コントローラは各種のステータス レジスタを備えています。これらのレジスタは、直前に転送を実行した DMA チャンネル (DMACS1 レジスタの LSTCH<3:0> ビット) と、そのチャンネルがアクセスしていた DPSRAM アドレス オフセット (DSADR レジスタの DSADR<15:0>) および使用していたバッファ (DMACS1 レジスタの PPSTx ビット) に関する情報を提供します。

38.10 データ書き込みコリジョン

CPU と DMA チャンネルは、同時に任意の DPSRAM または DMA 対応周辺モジュール データ レジスタを読み出すか、あるいは一方が読み出し中に他方が書き込む事ができます。唯一の制約条件として、CPU と DMA チャンネルの両方が同時に同一アドレスに書き込む事はできません。通常、このような状況は発生しませんが、何らかの原因によって同時書き込みが発生した場合、フラグがセットされて DMA フォルトトラップが開始します。CPU 書き込みを優先させる事もできますが、これは主に予測可能な動作を行うためであり、それ以外にほとんど実用性はありません。

同一バスサイクル中に同一アドレスに対して DMA チャンネルによる書き込みと CPU による読み出し (あるいは CPU による書き込みと DMA チャンネルによる読み出し) を実行できます。しかしこの場合、そのバスサイクル中に書き込まれた値ではなく、以前の値が読み出されます。また、このような状況は正常動作とみなされ、特別な対応動作は発生しません。

CPU と DMA チャンネルが同時に同一 DPSRAM アドレスに書き込みアクセスした場合、DMA コントローラ ステータス 0 (DMACS0) レジスタの XWCOLx ビットがセットされます。CPU と DMA チャンネルが同時に同一周辺モジュール アドレスに書き込みアクセスした場合、DMA コントローラ ステータス 0 (DMACS0) レジスタの PWCOLx ビットがセットされます。全てのコリジョン ステータスフラグの論理和 (OR) から共通の DMAC フォルトトラップが生成されます。XWCOLx および PWCOLx フラグは、ユーザ アプリケーションが割り込みコントローラ (INTCON1) レジスタの DMAC エラー ステータスビット (DMACERR) をクリアした時に、自動的にクリアされます。

XWCOLx または PWCOLx がクリアされるまで、書き込みコリジョン エラーが発生したチャンネルに対する後続の DMA 要求は無視されます。

書き込みコリジョン条件では、XWCOLx または PWCOLx のいずれか一方だけがセットされます。両方のフラグがセットされた状態は、稀に発生する手動トリガイベント エラーのステータス表示用として予約されています (このエラー専用のステータス ビットが存在しないため ; **38.6.10「手動転送モード」**参照)。

例 38- 13 に、DMA チャンネル 0 で DPSRAM から周辺モジュール (UART) にデータを転送し、DMA チャンネル 1 で周辺モジュール (ADC) から DPSRAM にデータを転送する場合の DMA コントローラ のトラップ処理用サンプルコードを示します。

例 38- 13: DMA コントローラのトラップ処理

```
void __attribute__((interrupt, no_auto_psv)) _DMACError(void)
{
 static unsigned int ErrorLocation;

 // Peripheral Write Collision Error Location
 if(DMACS0 & 0x0100)
 {
    ErrorLocation = DMA0STA;
 }

 // DMA RAM Write Collision Error Location
 if(DMACS0 & 0x0002)
 {
    ErrorLocation = DMA1STA;
 }

 DMACS0 = 0;                      //Clear Write Collision Flag
 INTCON1bits.DMACERR = 0;         //Clear Trap Flag
}
```

38.11 省電力モード時の動作

38.11.1 スリープモード

スリープモード時に DMA は無効化されます。スリープモードに移行する前に、全ての DMA チャンネルの未完了ブロック転送を完了させるか、あるいは無効化する事を推奨します。ただしこれは必須の措置ではありません。

38.11.2 アイドルモード

DMA はシステム内の第 2 のバスマスタであるため、CPU がアイドルモードに移行しても、データ転送を続ける事ができます。周辺モジュールが DMA チャンネルを使用し、かつその周辺モジュールがアイドルモード時にも動作するように設定されている場合、その周辺モジュールと DPSRAM 間で双方向にデータを転送できます。ブロック転送完了時に、DMA チャンネルは割り込みを生成して CPU をウェイクアップします (その割り込みが有効化されている場合のみ)。ウェイクアップした CPU は割り込みサービスハンドラを実行します。

各周辺モジュールは、アイドル時停止制御ビットを備えています。この制御ビットがセットされている場合、その周辺モジュールは、CPU がアイドルモードである間無効化されます。周辺モジュールとのデータ転送 (方向問わず) に DMAC を使用する場合、周辺モジュールのアイドル時停止機能を有効化すると、実質的にその周辺モジュールに割り当てられている DMA チャンネルも無効化されます。

38.12 レジスタマップ

表 38-6 に DMA コントローラ関連のレジスタマップを示します。

表 38-6: DMA 関連のレジスタマップ

レジスタ名	アドレス	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	全リセット
DMA0CON	0380	CHEN	SIZE	DIR	HALF	NULLW	—	—	—	—	—	AMODE<1:0>		—	—	MODE<1:0>		0000
DMA0REQ	0382	FORCE	—	—	—	—	—	—	—	—	IRQSEL<6:0>							0000
DMA0STA	0384	STA<15:0>																0000
DMA0STB	0386	STB<15:0>																0000
DMA0PAD	0388	PAD<15:0>																0000
DMA0CNT	038A	—	—	—	—	—	—	CNT<9:0>										0000
DMA1CON	038C	CHEN	SIZE	DIR	HALF	NULLW	—	—	—	—	—	AMODE<1:0>		—	—	MODE<1:0>		0000
DMA1REQ	038E	FORCE	—	—	—	—	—	—	—	—	IRQSEL<6:0>							0000
DMA1STA	0390	STA<15:0>																0000
DMA1STB	0392	STB<15:0>																0000
DMA1PAD	0394	PAD<15:0>																0000
DMA1CNT	0396	—	—	—	—	—	—	CNT<9:0>										0000
DMA2CON	0398	CHEN	SIZE	DIR	HALF	NULLW	—	—	—	—	—	AMODE<1:0>		—	—	MODE<1:0>		0000
DMA2REQ	039A	FORCE	—	—	—	—	—	—	—	—	IRQSEL<6:0>							0000
DMA2STA	039C	STA<15:0>																0000
DMA2STB	039E	STB<15:0>																0000
DMA2PAD	03A0	PAD<15:0>																0000
DMA2CNT	03A2	—	—	—	—	—	—	CNT<9:0>										0000
DMA3CON	03A4	CHEN	SIZE	DIR	HALF	NULLW	—	—	—	—	—	AMODE<1:0>		—	—	MODE<1:0>		0000
DMA3REQ	03A6	FORCE	—	—	—	—	—	—	—	—	IRQSEL<6:0>							0000
DMA3STA	03A8	STA<15:0>																0000
DMA3STB	03AA	STB<15:0>																0000
DMA3PAD	03AC	PAD<15:0>																0000
DMA3CNT	03AE	—	—	—	—	—	—	CNT<9:0>										0000
DMA4CON	03B0	CHEN	SIZE	DIR	HALF	NULLW	—	—	—	—	—	AMODE<1:0>		—	—	MODE<1:0>		0000
DMA4REQ	03B2	FORCE	—	—	—	—	—	—	—	—	IRQSEL<6:0>							0000
DMA4STA	03B4	STA<15:0>																0000
DMA4STB	03B6	STB<15:0>																0000
DMA4PAD	03B8	PAD<15:0>																0000
DMA4CNT	03BA	—	—	—	—	—	—	CNT<9:0>										0000
DMA5CON	03BC	CHEN	SIZE	DIR	HALF	NULLW	—	—	—	—	—	AMODE<1:0>		—	—	MODE<1:0>		0000
DMA5REQ	03BE	FORCE	—	—	—	—	—	—	—	—	IRQSEL<6:0>							0000
DMA5STA	03C0	STA<15:0>																0000
DMA5STB	03C2	STB<15:0>																0000
DMA5PAD	03C4	PAD<15:0>																0000

凡例： — = 未実装、「0」として読み出し . リセット値は 16 進数で表記

表 38-6: DMA 関連のレジスタマップ (続き)

レジスタ名	アドレス	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	全リセット
DMA5CNT	03C6	—	—	—	—	—	—	CNT<9:0>										0000
DMA6CON	03C8	CHEN	SIZE	DIR	HALF	NULLW	—	—	—	—	—	AMODE<1:0>		—	—	MODE<1:0>		0000
DMA6REQ	03CA	FORCE	—	—	—	—	—	—	—	—	IRQSEL<6:0>							0000
DMA6STA	03CC	STA<15:0>																0000
DMA6STB	03CE	STB<15:0>																0000
DMA6PAD	03D0	PAD<15:0>																0000
DMA6CNT	03D2	—	—	—	—	—	—	CNT<9:0>										0000
DMA7CON	03D4	CHEN	SIZE	DIR	HALF	NULLW	—	—	—	—	—	AMODE<1:0>		—	—	MODE<1:0>		0000
DMA7REQ	03D6	FORCE	—	—	—	—	—	—	—	—	IRQSEL<6:0>							0000
DMA7STA	03D8	STA<15:0>																0000
DMA7STB	03DA	STB<15:0>																0000
DMA7PAD	03DC	PAD<15:0>																0000
DMA7CNT	03DE	—	—	—	—	—	—	CNT<9:0>										0000
DMACS0	03E0	PWCOL7	PWCOL6	PWCOL5	PWCOL4	PWCOL3	PWCOL2	PWCOL1	PWCOL0	XWCOL7	XWCOL6	XWCOL5	XWCOL4	XWCOL3	XWCOL2	XWCOL1	XWCOL0	0000
DMACS1	03E2	—	—	—	—	LSTCH<3:0>				PPST7	PPST6	PPST5	PPST4	PPST3	PPST2	PPST1	PPST0	0000
DSADR	03E4	DSADR<15:0>																0000
INTCON1	0080	NSTDIS	—	—	—	—	—	—	—	—	—	DMACERR	—	—	—	—	—	0000
IFS0	0084	—	DMA1IF	—	—	—	—	—	—	—	—	—	DMA0IF	—	—	—	—	0000
IFS1	0086	—	—	—	—	—	—	—	DMA2IF	—	—	—	—	—	—	—	—	0000
IFS2	0088	—	DMA4IF	—	—	—	—	—	—	—	—	—	DMA3IF	—	—	—	—	0000
IFS3	008A	—	—	DMA5IF	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
IFS4	008C	—	—	—	—	—	—	—	—	—	—	DMA7IF	DMA6IF	—	—	—	—	0000
IEC0	0094	—	DMA1IE	—	—	—	—	—	—	—	—	—	DMA0IE	—	—	—	—	0000
IEC1	0096	—	—	—	—	—	—	—	DMA2IE	—	—	—	—	—	—	—	—	0000
IEC2	0098	—	DMA4IE	—	—	—	—	—	—	—	—	—	DMA3IE	—	—	—	—	0000
IEC3	009A	—	—	DMA5IE	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
IEC4	009C	—	—	—	—	—	—	—	—	—	—	DMA7IE	DMA6IE	—	—	—	—	0000
IPC1	00A6	—	—	—	—	—	—	—	—	—	—	—	—	—	DMA0IP<2:0>			4444
IPC3	00AA	—	—	—	—	—	DMA1IP<2:0>			—	—	—	—	—	—	—	—	4444
IPC6	00B0	—	—	—	—	—	—	—	—	—	—	—	—	—	DMA2IP<2:0>			4444
IPC9	00B6	—	—	—	—	—	—	—	—	—	—	—	—	—	DMA3IP<2:0>			4444
IPC11	00BA	—	—	—	—	—	DMA4IP<2:0>			—	—	—	—	—	—	—	—	4444
IPC15	00C2	—	—	—	—	—	—	—	—	—	DMA5IP<2:0>			—	—	—	—	4444
IPC17	00C6	—	—	—	—	—	—	—	—	—	DMA7IP<2:0>			—	DMA6IP<2:0>			4444

凡例 : — = 未実装、「0」として読み出し . リセット値は 16 進数で表記

38.13 関連アプリケーション ノート

本セクションに関連するアプリケーション ノートの一覧を下に記載します。一部のアプリケーション ノートは dsPIC33F ファミリ向けではありません。ただし概念は共通しており、変更が必要であったり制限事項が存在するものの利用が可能です。ダイレクト メモリアクセス (DMA) (パート III) モジュールに関連する最新のアプリケーション ノートは下記の通りです。

タイトル	アプリケーション ノート番号

現在、関連するアプリケーション ノートはありません。

Note: dsPIC33F ファミリ関連のアプリケーション ノートとサンプルコードはマイクロチップ社のウェブサイト (www.microchip.com) でご覧頂けます。

38.14 改訂履歴

リビジョン A (2007 年 10 月)

本書の初版

リビジョン B (2011 年 6 月)

周辺モジュールと DMA チャンネルの関連付けテーブル (表 38-1 参照) の「周辺モジュールに書き込む場合の DMAxPAD レジスタ値」列 /「PMP - PMP マスタデータ転送」行の値を 0x608 に変更

ISBN: 978-1-61341-043-1

NOTE: