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 3. データメモリ

ハイライト

本セクションには以下の主要項目を記載しています。

Note: dsPIC33F/PIC24H ファミリ リファレンス マニュアルの本セクションは、デバイス データシートの内容の補足を目的としています。本セクションの内容は、dsPIC33F/PIC24H デバイスの一部の製品には対応しません。

本書の内容がお客様のご使用になるデバイスに対応しているかどうかは、最新デバイス データシート内の**「メモリ構成」**の冒頭に記載している注意書きでご確認ください。

デバイス データシートとファミリ リファレンス マニュアルの各セクションは、マイクロチップ社のウェブサイト (http://www.microchip.com) からダウンロードできます。

3.1 はじめに

dsPIC33F/PIC24H のデータ幅は 16 ビットです。全ての内部レジスタとデータ領域メモリは、16 ビット幅で構成されます。dsPIC33F/PIC24H は 2 つのデータ領域を備えます。一部のデジタル信号処理 (DSP) 命令を使用すると、2 つのデータ領域へ個別にアクセスできます。これに対しマイクコントローラ (MCU) 命令は、2 つのデータ領域を 1 つの 64K バイト リニア アドレッシング領域としてアクセスします。2 つのデータ領域へのアクセスには、2 つのアドレス生成ユニット (AGU) と、それぞれに個別のデータパスを使用します。

図 3-1 に、データ領域マップの一例を示します。0x0000 ～ 0x07FF のデータメモリ アドレスは、デバイスの特殊機能レジスタ (SFR) 向けに予約されています。SFR は、デバイス上の CPU および周辺モジュール向けの制御ビットとステータスビットを格納します。

RAM はアドレス 0x0800 から始まり、2 つのブロック (X および Y データ領域) に分割されます。データ書き込みの場合、X および Y データ領域へは常に単一のリニアデータ領域としてアクセスします。データ読み出しの場合には、X および Y メモリ領域へ個別にアクセスするか、あるいは単一リニア領域としアクセスする事ができます。MCU クラス命令によるデータ読み出しは、常に X および Y データ領域を結合した単一のデータ領域へアクセスします。2 つのソースオペランドを持つ DSP 命令 (MAC 命令等) は、X および Y データ領域へ個別にアクセスして、2 つのソースオペランドの同時読み出しをサポートします。

MCU 命令は、データ読み書き用アドレスポインタとして任意のワーキング レジスタを使用できます。

DSP クラス命令は、データ読み出し中に Y アドレス領域を他のデータ領域から分離します。Y データ領域からの読み出し用アドレスポインタには W10 と W11 を使用します。この場合、残りのデータ領域を X 領域と呼びます。正確には「X マイナス Y」領域と呼ぶ事ができます。DSP クラス命令は、X データ領域からのデータ読み出し用アドレスポインタとして W8 と W9 を使用します。

図 3-2 に、MCU クラス命令と DSP クラス命令向けのデータメモリの割り当て方法を示します。データ読み出し時のアドレス領域へのアクセス方法は、使用するワーキング レジスタと命令タイプによって異なる事に注意してください。特に MCU クラス命令は、X および Y データメモリを結合した単一のデータ領域へアクセスする事に注意が必要です。MCU 命令は、読み書き用アドレスポインタとして任意のワーキング レジスタを使用できます。2 つのデータオペランドを同時にプリフェッチ可能な DSP 命令は、データメモリを 2 つの領域に分割します。この場合、読み出しアドレスポインタには決められた特定のワーキング レジスタを使用する必要があります。

一部の DSP 命令では、命令の対象ではないアキュムレータをデータメモリに格納できるものがあります。この機能は「アキュムレータ書き戻し」と呼びます。アキュムレータ書き戻しにおいては、X/Y を結合した単一データメモリ領域へのアドレスポインタとして W13 を使用する必要があります。

DSP クラス命令の場合、メモリ読み出し時の X メモリ領域へのアドレスポインタには、常に W8 と W9 を使用する必要があります。W8 または W9 を Y メモリ領域へのポインタとして使用すると、ゼロが返されます。W8 または W9 が未実装のメモリアドレスを指した場合には、アドレス エラートラップが発生します。

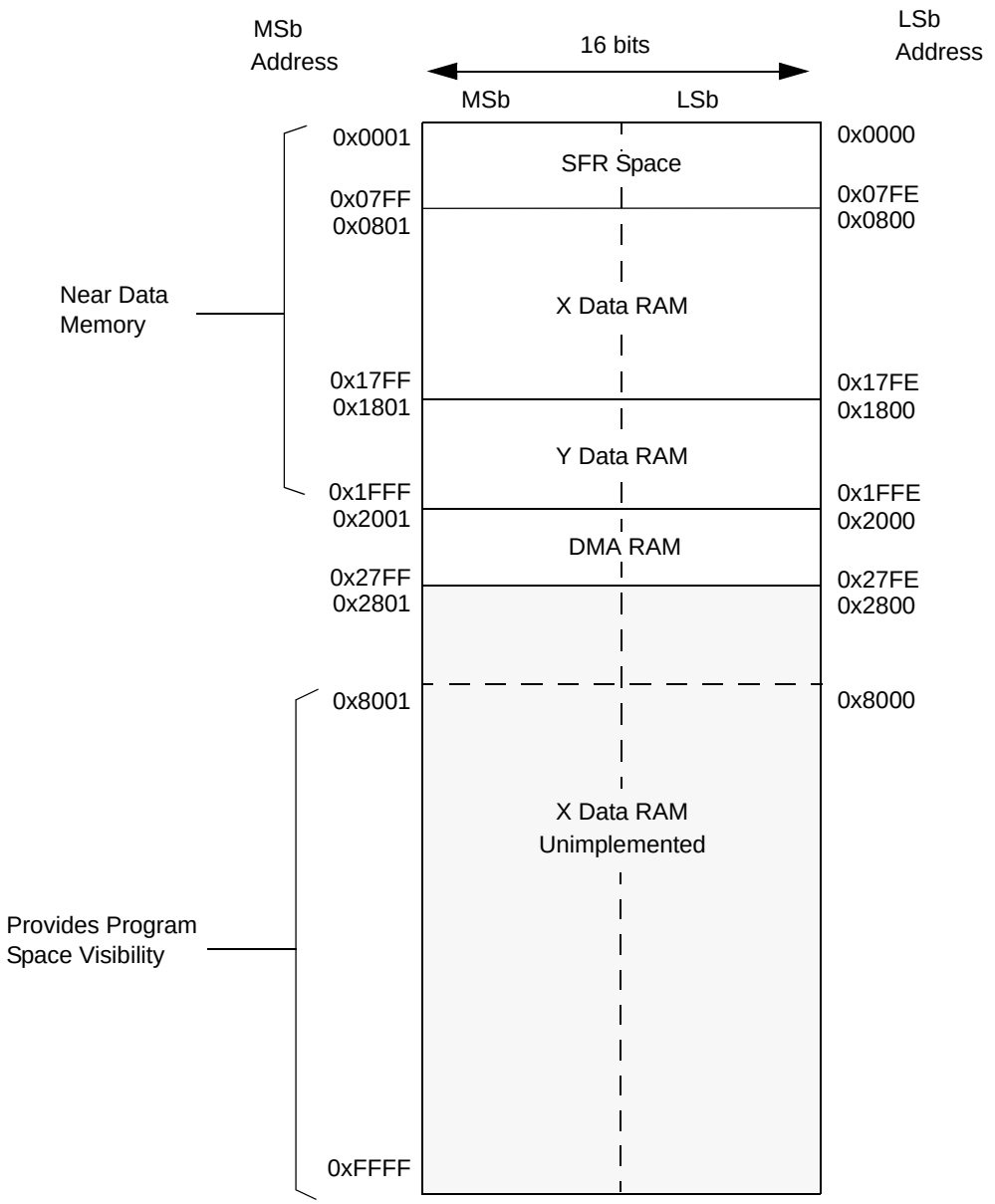
DSP クラス命令の場合、メモリ読み出し時の Y メモリ領域へのアドレスポインタには、常に W10 と W11 を使用する必要があります。W10 または W11 を X メモリ領域へのポインタとして使用すると、ゼロが返されます。W10 または W11 が未実装のメモリアドレスを指した場合には、アドレス エラートラップ発生します。
アドレス エラートラップの詳細は**セクション 6.「割り込み」**(DS70184) を参照してください。

Note:	データメモリ マップと、X および Y データ領域間の境界位置は、デバイスによって異なります。詳細は各 dsPIC33F/PIC24H のデバイス データシートを参照してください。

一部の dsPIC33F/PIC24H は、DMA とデュアルポート SRAM メモリ (DPSRAM) を内蔵しています。CPU コントローラと DMA コントローラは、CPU ストール等の影響を受けずに DSPRAM 内のアドレスに対して読み書きできます。これにより、リアルタイム性能が最大限に高められます。詳細は**セクション 22.「ダイレクト メモリアクセス (DMA)」**(DS70182) を参照してください。

Note:	DMA RAM の実装の有無とサイズはデバイスによって異なります。詳細は各 dsPIC33F/PIC24H のデバイス データシートを参照してください。

図 3-1: データメモリ マップの例



Note 1: X および Y データ領域のサイズは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。上図のデータ領域境界の位置は参考例です。

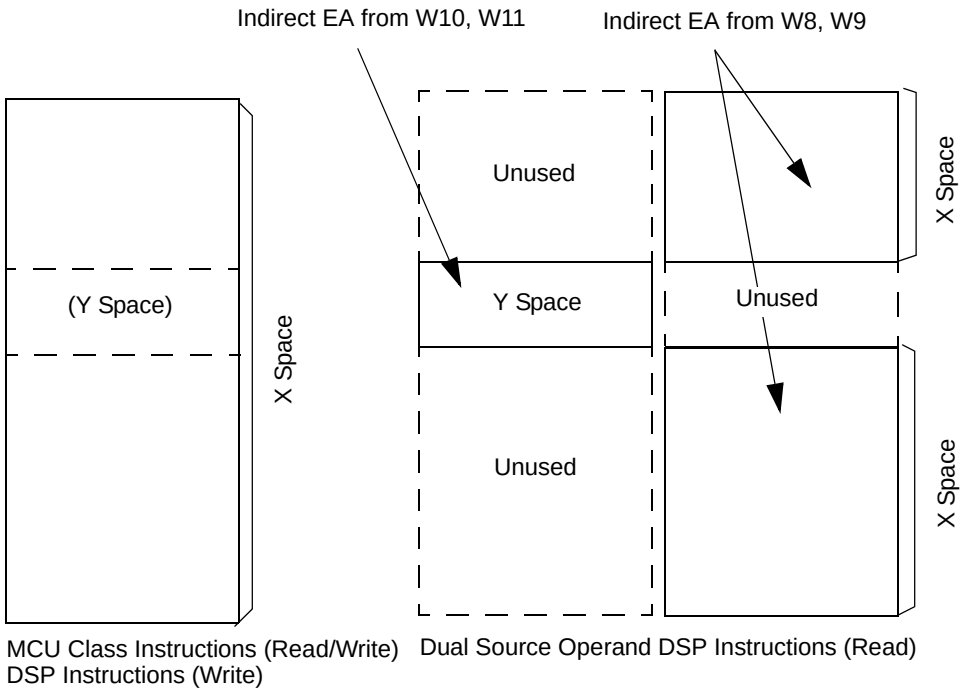
2: 一部のデバイスだけが DMA RAM を実装しています。詳細は各デバイスのデータシートを参照してください。

3: ニアデータ メモリへは、13 ビットアドレスをオペコード内へエンコードするファイル レジスタ命令を使用して直接アクセスできます。

4: 全てのデータメモリへは、ワーキング レジスタを介して間接的にアクセスするか、あるいは MOV 命令を用いて直接アクセスできます。

5: データメモリ マップの上位側半分は、PSV (Program Space Visibility) 向けに、プログラムメモリ領域の 1 つのセグメント内へ割り当てる事ができます。

図 3-2: MCU 命令と DSP 命令のデータ領域



Note: DSP 命令のデータ書き込みでは、データメモリ全体を 1 つの結合領域と見なします。アキュムレータ書き戻しを実行する DSP 命令は、W13 を結合データ領域への書き込み用アドレスポインタとして使用します。

3.1.1 ニアデータメモリ

データメモリ領域内の 0x0000 ～ 0x1FFF には、ニアデータメモリと呼ばれる 8K バイトのアドレス領域が確保されます。全てのファイルレジスタ命令では、13 ビット絶対アドレス フィールドを介してニアデータメモリを直接アドレッシングできます。

ニアデータ領域が格納するメモリ領域は、各 dsPIC33F/PIC24H 製品のデータメモリ実装サイズによって異なります。ニアデータ領域は、少なくとも全ての SFR と X データメモリの一部を格納します。データモリ サイズが小さいデバイスでは、ニアデータ領域内に全ての X メモリ領域と、Y メモリ領域の一部または全部を格納する事ができます。詳細は図 3-1 を参照してください。

Note: MOV 命令を使用すると、64K データ領域の全体を直接アドレッシングできます。詳細は「*16 ビット MCU および DSC プログラマ リファレンス マニュアル*」(DS70157) を参照してください。

3.2 制御レジスタ

各レジスタの固有機能を以下に要約します。

- **MODCON: モジュロおよびビット反転アドレッシング制御レジスタ (1)**
  - X AGU および Y AGU 向けモジュロ アドレッシングの有効化 / 無効化
  - ビット反転アドレッシング用レジスタの選択
  - モジュロ アドレッシング用レジスタの選択
- **XMODSRT: X AGU モジュロ アドレッシング開始レジスタ**
  - X RAGU および X WAGU 向けモジュロ アドレッシング開始アドレスの選択
- **XMODEND: X AGU モジュロ アドレッシング終了レジスタ**
  - X RAGU および X WAGU 向けモジュロ アドレッシング終了アドレスの選択
- **YMODSRT: Y AGU モジュロ アドレッシング開始レジスタ**
  - YGAU 向けモジュロ アドレッシング開始アドレスビットの選択
- **YMODEND: Y AGU モジュロ アドレッシング終了レジスタ**
  - YGAU 向けモジュロ アドレッシング終了アドレスビットの選択
- **XBREV: X 書き込み AGU ビット反転アドレッシング制御レジスタ**
  - X AGU 向けビット反転アドレッシングの有効化 / 無効化
  - X AGU ビット反転修飾子向けのバッファ選択

3.3 制御レジスタの説明

以下に記載する各レジスタは、モジュロおよびビット反転アドレッシングを制御します。

レジスタ 3-1: MODCON: モジュロおよびビット反転アドレッシング制御レジスタ (1)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
XMODEN	YMODEN	—	—	BWM<3:0>			
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
YWM<3:0>				XWM<3:0>			
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 での値	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 **XMODEN:** X RAGU および X WAGU モジュロ アドレッシングのイネーブルビット
1 = X AGU モジュロ アドレッシング有効
0 = X AGU モジュロ アドレッシング無効

bit 14 **YMODEN:** Y AGU モジュロ アドレッシングのイネーブルビット
1 = Y AGU モジュロ アドレッシング有効
0 = Y AGU モジュロ アドレッシング無効

bit 13-12 **未実装 :**「0」として読み出し

bit 11-8 **BWM<3:0>:** ビット反転アドレッシングの X WAGU レジスタ選択ビット
1111 = ビット反転アドレッシング無効
1110 = ビット反転アドレッシング用に W14 を選択
1101 = ビット反転アドレッシング用に W13 を選択
•
•
•
0000 = ビット反転アドレッシング用に W0 を選択

bit 7-4 **YWM<3:0>:** モジュロ アドレッシングの Y AGU ワーキング レジスタ選択ビット
1111 = モジュロ アドレッシング無効
1010 = モジュロ アドレッシング用に W10 を選択
1011 = モジュロ アドレッシング用に W11 を選択
YWM<3:0> 制御ビットのその他の設定は全て予約済みです。使用しないでください。

bit 3-0 **XWM<3:0>:** モジュロ アドレッシングの X RAGU および X WAGU ワーキング レジスタ選択ビット
1111 = モジュロ アドレッシング無効
1110 = モジュロ アドレッシング用に W14 を選択
•
•
•
0000 = モジュロ アドレッシング用に W0 を選択

Note 1: MODCON レジスタへの書き込みの直後に、ワーキング レジスタを用いた間接読み出しを実行する命令を使用しないでください。予期せぬ結果が生じる可能性があります。一部の命令は、暗黙的に間接読み出しを実行します。これには右記の命令が含まれます : POP、RETURN、RETFIE、RETLW、ULNK

レジスタ 3-2: XMODSRT: X AGU モジュロ アドレッシング開始レジスタ

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
XS<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-0
XS<7:1>							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 での値	1 = ビットをセット	0 = ビットをクリア	x = ビットは未知

bit 15-1 **XS<15:1>:** X RAGU および X WAGU モジュロ アドレッシング開始アドレスビット

bit 0 **未実装 :**「0」として読み出し

レジスタ 3-3: XMODEND: X AGU モジュロ アドレッシング終了レジスタ

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
XE<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-1
XE<7:1>							1
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 での値	1 = ビットをセット	0 = ビットをクリア	x = ビットは未知

bit 15-1 **XE<15:1>:** X RAGU および X WAGU モジュロ アドレッシング終了アドレスビット

bit 0 **未実装 :**「1」として読み出し

レジスタ 3-4: YMODSRT: Y AGU モジュロ アドレッシング開始レジスタ

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
YS<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-0
YS<7:1>							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 での値	1 = ビットをセット	0 = ビットをクリア	x = ビットは未知

bit 15-1 **YS<15:1>:** Y AGU モジュロ アドレッシング開始アドレスビット

bit 0 **未実装 :**「0」として読み出し

レジスタ 3-5: YMODEND: Y AGU モジュロ アドレッシング終了レジスタ

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
YE<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-1
YE<7:1>							1
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 での値	1 = ビットをセット	0 = ビットをクリア	x = ビットは未知

bit 15-1 **YE<15:1>:** Y AGU モジュロ アドレッシング終了アドレスビット

bit 0 **未実装 :**「1」として読み出し

レジスタ 3-6: XBREV: X 書き込み AGU ビット反転アドレッシング制御レジスタ

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BREN	XB<14:8>						
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
XB<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 での値	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 **BREN:** ビット反転アドレッシング (X AGU) イネーブルビット
1 = ビット反転アドレッシング有効
0 = ビット反転アドレッシング無効

bit 14-0 **XB<14:0>:** X AGU ビット反転修飾子ビット
0x4000 = 32768 ワードバッファ
0x2000 = 16384 ワードバッファ
0x1000 = 8192 ワードバッファ
0x0800 = 4096 ワードバッファ
0x0400 = 2048 ワードバッファ
0x0200 = 1024 ワードバッファ
0x0100 = 512 ワードバッファ
0x0080 = 256 ワードバッファ
0x0040 = 128 ワードバッファ
0x0020 = 64 ワードバッファ
0x0010 = 32 ワードバッファ
0x0008 = 16 ワードバッファ
0x0004 = 8 ワードバッファ
0x0002 = 4 ワードバッファ
0x0001 = 2 ワードバッファ

3.4 データ領域アドレス生成ユニット (AGU)

dsPIC33F/PIC24H は、データメモリのアドレス生成用に X AGU と Y AGU を備えています。X および Y AGU は、それぞれ 64K バイトレンジ内で実効アドレス (EA) を生成可能です。ただし、物理メモリの実装範囲外の実効アドレスは何も効果を持ちません (データ読み出しではゼロを返し、書き込みでは効果なし)。アドレス エラートラップの詳細は**セクション 6.「割り込み」**(DS70184) を参照してください。

3.4.1 X アドレス生成ユニット

X AGU は、全ての命令が使用し、全てのアドレッシング モードをサポートします。X AGU は読み出し AGU (X RAGU) と書き込み AGU (X WAGU) により構成されます。これらは命令サイクルの各種フェイズにおいて、それぞれ読み出しバスと書き込みバス上で個別に動作します。X 読み出しデータバスは、データ領域を X および Y アドレス領域を結合した単一データ領域として見なす全ての命令に対して、データのリターンパスとして機能します。DSP クラスの 2 オペランド読み出し命令に対しては、このデータパスは X アドレス領域データパスとして機能します。X 書き込みデータバスは、全ての命令に対して、X/Y 結合データ領域への書き込みパスとしてのみ機能します。

X RAGU は、1 つ前の命令サイクル実行中に、プリフェッチされた命令の情報に基づいて実効アドレスの計算を開始します。X RAGU が計算した実効アドレスは、命令サイクル開始時にアドレスバスへ提示されます。

X WAGU は、命令サイクル開始時に実効アドレスの計算を開始します。この実効アドレスは、その命令の書き込みフェイズ時にアドレスバスへ提示されます。

X RAGU と X WAGU は、どちらもモジュロ アドレッシングをサポートします。X WAGU だけがビット反転アドレッシングもサポートします。

3.4.2 Y アドレス生成ユニット

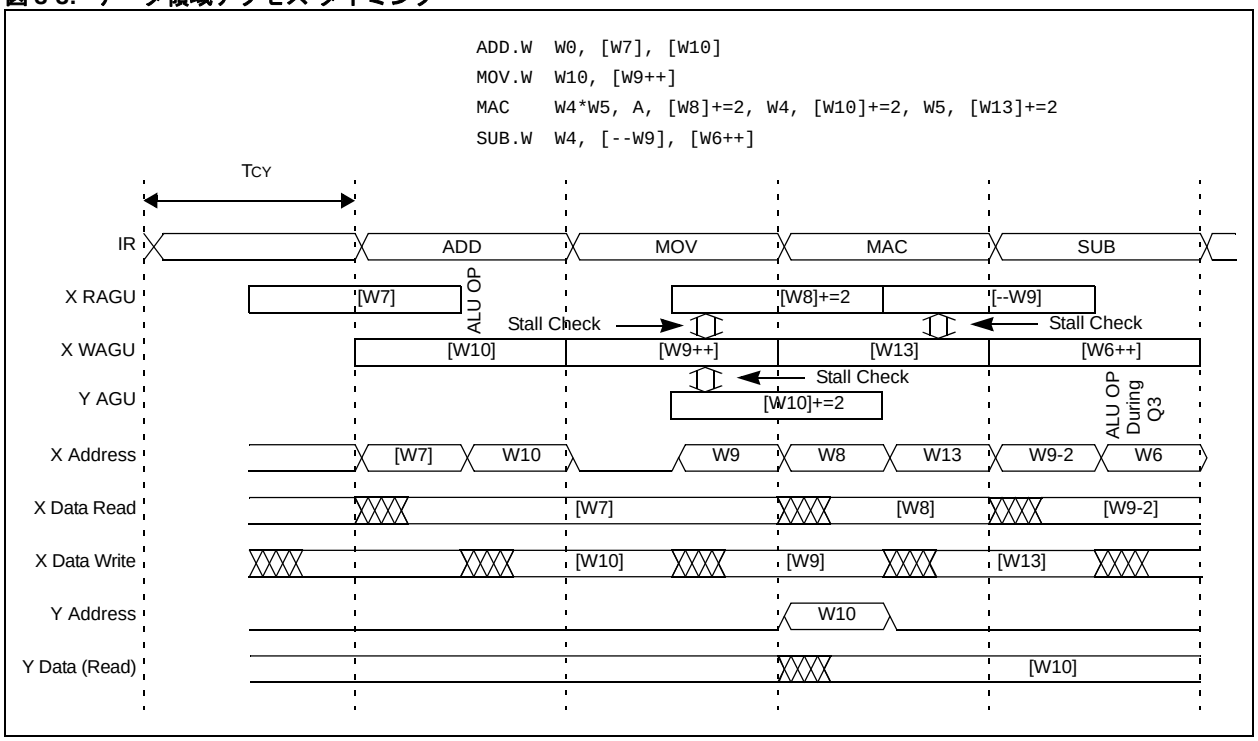
Y データメモリ領域は、データ読み出しをサポートする 1 つの AGU を備えます。データ書き込みには Y メモリバスを使用しません。Y AGU と Y モリバスの機能は、DSP クラス命令の同時データ読み出しをサポートします。

Y AGU のタイミングは X RAGU のタイミングと同じです。従って命令サイクルの開始前に、プリフェッチされた命令の情報に基づいて実効アドレスの計算を開始します。この実効アドレスは、命令サイクルの開始時にアドレスバスへ提示されます。

Y AGU は、Y AGU を使用する DSP クラス命令に対して、モジュロ アドレッシングおよび後置修飾アドレッシング モードをサポートします。

Note:	Y AGU はデータ書き込みをサポートしません。全てのデータ書き込みは、X WAGU 経由で X/Y 結合データ領域に対して行います。Y AGU は、2 ソースオペランド DSP 命令のデータ読み出し時にのみ使用します。

図 3-3: データ領域アクセス タイミング



3.4.3 アドレス生成ユニットと DSP クラス命令

DSP クラス命令は、Y AGU と Y メモリ データパスを X RAGU と連携させて使用します。これにより、2 つの同時データ読み出しパスが得られます。例えば MAC 命令は、次の乗算に使用する 2 つのオペランドを同時にプリフェッチできます。図 3-3 を参照してください。

DSP クラス命令は、ワーキング レジスタポインタ W8 と W9 を用いて、常に Y データ領域から独立して X RAGU により X データ領域を動作させます。また、ワーキング レジスタポインタ W10 と W11 を用いて、常に X データ領域から独立して Y AGU により Y データ領域を動作させます。DSP クラス命令によるデータ書き込みは、X バスを介して常に X/Y 結合データ領域内で行います。このため、実効アドレスとは無関係に任意のアドレスへ書き込む事ができます。

Y AGU は、DSP クラス命令に関連する後置修飾アドレッシング モードのみをサポートします。アドレッシング モードの詳細は「*16 ビット MCU および DSC プログラマ リファレンス マニュアル*」(DS70157) を参照してください。Y AGU は、自動リングバッファ向けのモジュロ アドレッシングもサポートします。その他の MCU クラス命令は、全て X AGU を介して Y データ アドレス領域へアクセスできます。この場合、Y データ領域は結合リニア領域の一部と見なされます。

3.4.4 データ配置

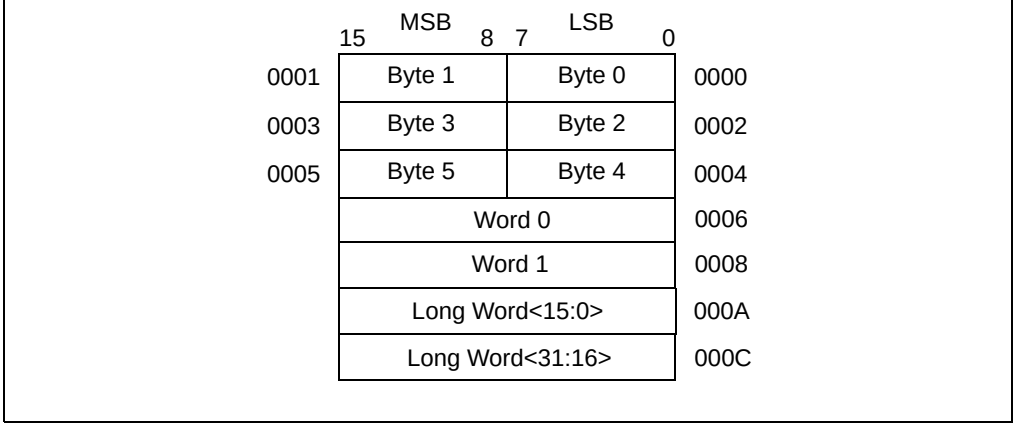
命令セット アーキテクチャ(ISA) は、X メモリ AGU を介してデータへアクセスする全ての MCU 命令に対して、ワードおよびバイト動作の両方をサポートします。ワード動作では、16 ビット データアドレスの LSb を無視します。ワードデータはリトルエンディアン形式で配置します。この場合、最下位バイト (LSB) を偶数アドレス (LSb = 0)、最上位バイト (MSB) を奇数アドレス (LSb = 1) に配置します。図 3-4 を参照してください。

バイト動作では、データアドレスの LSB がアクセス先のバイトを選択します。アドレスされたバイトは、内部データバスの下位 8 ビットに配置されます。

全ての実効アドレス計算は、バイトアクセスかワードアクセスかに応じて自動的に調整されます。例えば、1 回のワード操作でアドレスを 2 つインクリメントし、これによりアドレスポインタをポストインクリメントします。

Note: 全てのワードアクセスは、偶数アドレス (LSb = 0) へ配置する必要があります。これに従わないワードデータ フェッチはサポートされません。従って、バイト動作とワード動作を混用する場合や、8 ビット PIC®MCU の既存コードから移行する場合には、注意が必要です。不正な配置のワード読み書きを試みると、アドレス エラートラップが発生します。不正配置の読み出しは実行されますが、書き込みは実行されません。その後のトラップ発生により、システムはアドレスフォルト実行前のマシンステートを分析する事ができます。

図 3-4: データ配置



3.5 モジュロ アドレッシング

モジュロまたは巡回アドレッシングは、ハードウェアを用いてリングデータ バッファをサポートするための自動的な手段を提供します。一般的に多くの DSP アルゴリズムはループ内のコードを高速に実行しますが、この自動的手段は、そのようなコード実行時のソフトウェアによるデータアドレス境界チェックを不要にする事を目的とします。

モジュロバッファへのポインタには、W15 以外の全てのワーキング レジスタを選択できます。モジュロ ハードウェアは、選択したワーキング レジスタが保持するアドレス上で境界チェックを実行し、必要に応じてポインタ値をバッファ境界位置へ自動的に調整します。

dsPIC33F/PIC24H のモジュロ アドレッシングは、データ領域でもプログラム領域でも動作可能です (両領域のデータポインタのメカニズムは基本的に同じため)。X および Y データ領域では、それぞれ 1 つのリングバッファを使用できます (X 領域のリングバッファはプログラム領域へのポインタも提供)。

モジュロデータのバッファ長は 32K ワード以下で選択できます。モジュロバッファのロジックは、ワードサイズまたはバイトサイズ データの使用をサポートします。ただし、モジュロロジックは、アドレス境界チェックを必ずワードアドレス境界位置で行います。従ってバイト モジュロ バッファの長さは偶数でなければなりません。加えて、バイトサイズ モジュロバッファには Y AGU を使用できません (Y メモリ データバス経由のバイトアセスはサポートしないため)。

3.5.1 モジュロ開始 / 終了アドレスの選択

モジュロバッファの開始および終了アドレスの指定には、下記 4 つのアドレスレジスタを使用できます。

- **XMODSRT: X AGU モジュロ アドレッシング開始レジスタ**
- **XMODEND: X AGU モジュロ アドレッシング終了レジスタ**
- **YMODSRT: Y AGU モジュロ アドレッシング開始レジスタ**
- **YMODEND: Y AGU モジュロ アドレッシング終了レジスタ**

モジュロバッファの開始アドレスは、偶数バイトのアドレス境界上に置く必要があります。従って、XMODSRT および YMODSRT レジスタの LSB は「0」である事が必要です。モジュロバッファの終了アドレスは、奇数バイトのアドレス境上に置く必要があります。従って、XMODEND および YMODEND レジスタの LSB は「1」である事が必要です。各モジュロバッファ向けに選択した開始および終了アドレスの処理方法は、インクリメント バッファまたはデクリメント バッファのどちらを使用するかによって異なります。インクリメント バッファの場合、ワーキング レジスタポインタをバッファアドレス範囲内でインクリメントし、これがインクリメント バッファの終了アドレスに達すると、ワーキング レジスタポインタをリセットします (バッファ開始アドレスを指します)。デクリメント バッファの場合、ワーキング レジスタポインタをバッファアドレス範囲内でデクリメントし、これがデクリメント バッファの開始アドレスに達すると、ワーキング レジスタポインタをリセットします(バッファ終了アドレスを指します)。

Note:	ユーザは、アプリケーション要件に応じて、インクリメントまたはデクリメント モジュロバッファのどちらかを選択する必要があります。この選択に応じて、上記のようにアドレスの処理方法が異なります。

3.5.1.1 モジュロ開始アドレス

データバッファ開始アドレスは任意ですが、インクリメント モジュロバッファでは、バイナリ「0」境界の値である必要があります。デクリメント モジュロバッファのモジュロ開始アドレスは、選択した終了アドレスとバッファ長から計算で求まります。

例えば、インクリメント バッファのバッファ長として 50 ワード (100 バイト) を選択した場合、バッファ開始バイトアドレスの下位 7 バイトは全てゼロでなければなりません。従って開始アドレスの有効範囲は 0xNN00 ～ 0xNN80 となります (N は任意の 16 進数)。

3.5.1.2 モジュロ終了アドレス

データバッファ終了アドレスは任意ですが、デクリメント バッファではバイナリ「1」境界の値である必要があります。インクリメント バッファのモジュロ終了アドレスは、選択したバッファ開始アドレスとバッファ長から計算で求まります。

例えば、デクリメント モジュロバッファのバッファサイズ (モジュラス値) として 50 ワード (100 バイト) を選択した場合、バッファ終了バイトアドレスの下位 7 ビットは全て 1 でなければなりません。従って終了アドレスの有効範囲は 0xNNFF 〜 0xNN7F となります (N は任意の 16 進数)。

Note:	必要なモジュロバッファ長が「2 のべき乗」である場合、モジュロ開始および終了アドレスは、インクリメント バッファとデクリメント バッファ両方の要求を満たすように選択できます。

3.5.1.3 モジュロアドレスの計算

インクリメント モジュロバッファの終了アドレスは、選択した開始アドレスとバッファ長 (共にバイト値) から計算する必要があります。式 3-1 に、終了アドレスの計算式を示します。

式 3-1: インクリメント バッファのモジュロ終了アドレス

終了アドレス = 開始アドレス + バッファ長 – 1

デクリメント モジュロバッファの開始アドレスは、選択した終了アドレスとバッファ長 (共にバイト値) から計算する要があります。式 3-2 に、開始アドレスの計算式を示します。

式 3-2: デクリメント バッファのモジュロ開始アドレス

開始アドレス = 終了アドレス - バッファ長 + 1

3.5.1.4 モジュロ アドレッシング SFR に関連するデータ依存性

モジュロおよびビット反転アドレッシング制御レジスタ (MODCON) への書き込みの直後に、ワーキング レジスタを用いた間接読み出しを行ってはなりません。例 3-1 にコード 例を示します。

Note 1:	POP 命令を用いてスタック先頭位置 (TOS) の内容を MODCON レジスタへポップした場合にも、MODCONレジスタへの書き込みが発生します。MODCONレジスタへの書き込み直後に、間接読み出しを行う命令を実行する事はできません。
2:	一部の命令は、暗黙的に間接読み出しを実行します。これには右記の命令が含まれます : POP RETURN、RETFIE、RETLW、ULNK

例 3-1: 不適正な MODCON 初期化

```
MOV #0x8FF4, w0    ;Initialize MODCON
MOV w0, MODCON
MOV [w1], w2       ;Incorrect EA generated here
```

この初期化問題に対処するには、MODCON レジスタ初期化直後の命令内で間接読み出し以外のアドレッシング モードを使用します。例 3-2 に、MODCON レジスタ初期化直後に NOP 命令を追加するだけの簡単な対処法を示します。

例 3-2: 適正な MODCON 初期化

```
MOV #0x8FF4, w0    ;Initialize MODCON
MOV w0, MODCON
NOP                ;See Note below
MOV [w1], w2       ;Correct EA generated here
```

Note: NOP 命令のかわりに、モジュロバッファ アクセス向けのワーキング レジスタを用いた間接読み出しを行わない別の命令を使用する事もできます。

下記モジュロアドレス SFR への書き込みにおいても、直後に実行する間接読み出しに対して以下に記載する条件が適用されます。

- XMODSRT
- XMODEND
- YMODSRT
- YMODEND

すなわち、モジュロ アドレッシングが MODCON レジスタで既に有効になっている場合、X または Y モジュロ アドレス SFR への書き込み直後に、X または Y データ領域からのモジュロ バッファ アクセス向けに指定したワーキング レジスタを用いて間接読み出しを行ってはなりません。例 3-3 に、X データ領域に関連するモジュロ SFR の初期化において予期せぬ結果が生じる可能性のあるサンプルコードを示します。Y データ領域での初期化についても同様です。

例 3-3: 不適正なモジュロ アドレッシングのセットアップ

```
MOV #0x8FF4,w0      ;Modulo addressing enabled
MOV w0, MODCON      ;in X-data space using w4
                    ;for buffer access
MOV #0x1200,w4      ;XMODSRT is initialized
MOV w4, XMODSRT
MOV #0x12FF,w0      ;XMODEND is initialized
MOV w0, XMODEND
MOV [w4++], w5      ;Incorrect EA generated
```

この問題に対処するには、モジュロアドレス SFR の初期化直後に NOP を実行するか、あるいはモジュロバッファ アクセス向けに指定したワーキング レジスタを用いる間接読み出し以外の動作を実行します (例 3-4 参照)。あるいは、モジュロ開始および終了アドレス SFR を初期化した後に、MODCON レジスタでモジュロ アドレッシングを有効にする方法もあります。

例 3-4: 適正なモジュロ アドレッシングのセットアップ

```
MOV #0x8FF4,w0      ;Modulo addressing enabled
MOV w0, MODCON      ;in X-data space using w4
                    ;for buffer access
MOV #0x1200,w4      ;XMODSRT is initialized
MOV w4, XMODSRT
MOV #0x12FF,w0      ;XMODEND is initialized
MOV w0, XMODEND
NOP                 ;See Note below
MOV [w4++], w5      ;Correct EA generated here
```

Note: NOP 命令のかわりに、モジュロバッファ アクセス向けのワーキング レジスタを用いた間接読み出しを行わない別の命令を使用する事もできます。

3.5.2 ワーキング アドレスレジスタの選択

モジュロ アドレッシング適用先の X アドレス領域ポインタ ワーキング レジスタは、MODCON レジスタの XWM ビット (MODCON<3:0>) に保存されます。XMODSRT、XMODEND、XWM レジスタの選択は、X RAGU と X WAGU の間で共有されます。X データ領域のモジュロ アドレッシングは、XWM ビットが 15 以外の値に設定され、かつ XMODEN ビット (MODCON<15>) がセットされた時に有効になります。W15 はモジュロ アドレッシング用のポインタとして使用できません (W15 はソフトウェア スタックポインタ専用です)。

モジュロ アドレッシング適用先の Y アドレス領域ポインタ ワーキング レジスタは、MODCON レジスタの YWM ビット (MODCON<7:4>) に保存されます。Y データ領のモジュロ アドレッシングは、YWM ビットが 15 以外の値に設定され、かつ YMODEN ビット (MODCON<14>) がセットされた時に有効になります。

Note: MODCON レジスタへの書き込みの直後に、ワーキング レジスタを用いた間接読み出しを行う命令を実行してはなりません。予期せぬ結果が生じる可能性があります。一部の命令は、暗黙的に間接読み出しを実行します。これには右記の命令が含まれます : POP、RETURN、RETFIE、RETLW、ULNK

3.5.3 モジュロ アドレッシングの適用性

モジュロ アドレッシングは、選択したワーキング レジスタに関連する実効アドレス計算へ適用できます。アドレス境界チェックは、インクリメント バッファの上位側アドレス境界以上のアドレス、またはデクリメント バッファの下位側アドレス境界以下のアドレスを探すという事に注意しください。これによりアドレス変更は、境界を飛び越えても正しく調整されます。モジュロ ハードウェアによるワーキング レジスタ ポインタの自動調整は、一方向にだけ行われます。すなわち、インクリメント バッファのワーキング レジスタ ポインタがデクリメントされた場合や、その逆の合には、モジュロ ハードウェアはワーキング レジスタ ポインタを正しく調整する事ができません。例外として、バッファ長が 2 の偶数乗である場合には、インクリメントとデクリメントの両モジュロバッファの境界要件に適合するように開始および終了アドレスを選択できます。

新しい実効アドレスは、モジュロバッファ境界を超えても修正可能です (最大でバッファ長まで超過可能)。これは、インデックス レジスタ ([Wb + Wn]) およびリテラル オフセット ([Wn + lit10]) アドレッシング モードを使用する場合に重要です。加えて、インデックス レジスタおよびリテラル オフセット アドレッシング モードは、ワーキング レジスタ内のホールド値を変更しません。前置修飾または後置修飾アドレッシング モード ([Wn++],[Wn--]、[++Wn]、[--Wn]) による間接アクセスだけがワーキング アドレスの値を変更します。

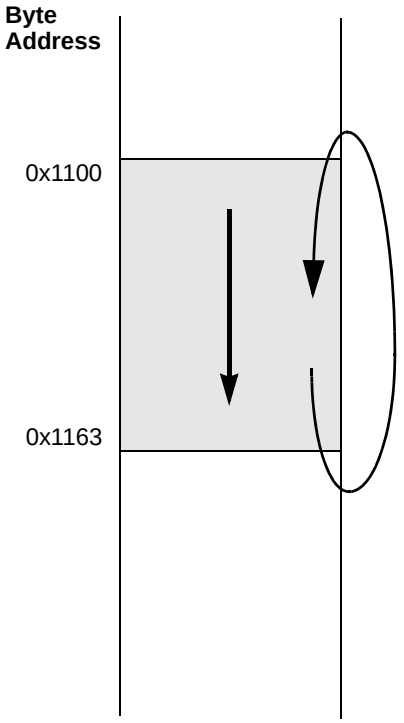
3.5.4 インクリメント モジュロバッファのモジュロ アドレッシング初期化

以下に、X AGU または Y AGU を用いたインクリメント リングバッファのセットアップ手順を記載します。

1. 16 ビット データワードでバッファ長を決定します。この値を 2 倍にしてバッファのバイト長を求めます。
2. 上記のバッファ長に基づいて、バイナリ「0」境界に位置するバッファ開始アドレスを選択します。バイトアドレス範囲を求めるには、バッファのワード長を 2 倍する必要がある事に注意してください。例えば、長さ 100 ワード (200 バイト) のバッファは 0xXX00 を開始アドレスとして使用できます。
3. ステップ 1 で選択したバッファ長と、ステップ 2 で選択したバッファ開始アドレスからバッファ終了アドレスを計算します (式 3-1 を使用)。
4. ステップ 2 で選択したバッファ開始アドレスを XMODSRT または YMODSRT レジスタへ読み込みます。
5. ステップ 3 で計算したバッファ終了アドレスを XMODEND または YMODEND レジスタへ読み込みます。
6. リングバッファへのアクセスに使用するワーキング レジスタを選択するために、XWM ビット (MODCON<3:0>) または YWM ビット (MODCON<7:4>) へ書き込みます。
7. XMODEN ビット (MODCON<15>) または YMODEN ビット (MODCON<14>) をセットして、リングバッファを有効にします。
8. バッファを指すポインタ値を上記で選択したワーキング レジスタへ読み込みます。

プリ / ポスト インクリメントによる間接アクセスが実行されると、ワーキング レジスタ アドレスは自動的にバッファの終了位置へ調整されます (図 3-5 参照)。

図 3-5: インクリメント バッファ モジュロ アドレッシングの動作例




```
MOV    #0x1100,W0
MOV    W0,XMODSRT      ;set modulo start address
MOV    #0x1163,W0
MOV    W0,XMODEND      ;set modulo end address
MOV    #0x8001,W0
MOV    W0,MODCON       ;enable W1, X AGU for modulo
MOV    #0x0000,W0      ;W0 holds buffer fill value
MOV    #0x1100,W1      ;point W1 to buffer
REPEAT #49,            ;fill the 50 buffer locations
MOV    W0,[W1++]       ;fill the next location

;W1 = 0x1100 when DO loop completes
```

Start Address = 0x1100
End Address = 0x1163
Length = 50 Words

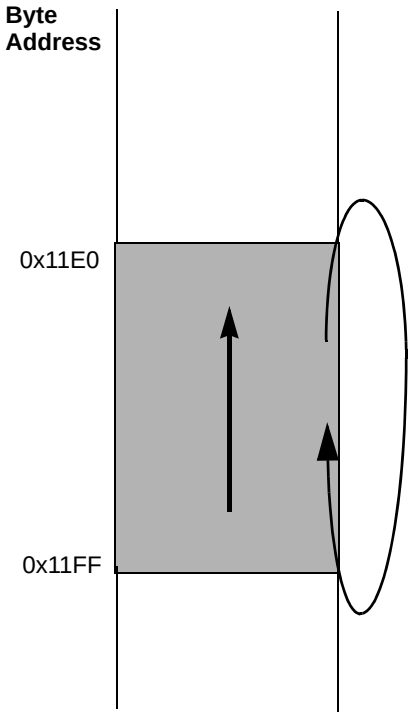
3.5.5 デクリメント モジュロバッファのモジュロ アドレッシング初期化

以下に、X AGU または Y AGU を用いたデクリメント リングバッファのセットアップ手順を記載します。

1. 16 ビット データワードでバッファ長を決定します。この値を 2 倍にしてバッファのバイト長を求めます。
2. 上記のバッファ長に基づいて、バイナリ 1's 境界に位置するバッファ終了アドレスを選択します。バイトアドレス範囲を求めるには、バッファのワード長を 2 倍する必要がある事に注意してください。例えば、長さ 128 ワード (256 バイト) のバッファは 0xXXFF を終了アドレスとして使用できます。
3. ステップ 1 で選択したバッファ長と、ステップ 2 で選択したバッファ終了アドレスからバッファ開始アドレスを計算します (式 3-2 を使用)。
4. ステップ 3 で計算したバッファ開始アドレスを XMODSRT または YMODSRT レジスタへ読み込みます。
5. ステップ 2 で選択したバッファ終了アドレスを XMODEND または YMODEND レジスタへ読み込みます。
6. リングバッファへアクセスするワーキング　レジスタを選択するために、XWM ビット (MODCON<3:0>) または YWM ビット (MODCON<7:4>) へ書き込みます。
7. XMODEN ビット (MODCON<15>) または YMODEN ビット (MODCON<14>) をセットして、リングバッファを有効にします。
8. バッファを指すポインタ値を上記で選択したワーキング レジスタへ読み込みます。

プリ / ポスト デクリメントによる間接アクセスが実行されると、ワーキング レジスタ アドレスは自動的にバッファの終了位置へ調整されます (図 3-6 参照)。

図 3-6: デクリメント バッファ モジュロ アドレッシングの動作例




```
MOV     #0x11E0,W0
MOV     W0,XMODSRT    ;set modulo start address
MOV     #0x11FF,W0
MOV     W0,XMODEND    ;set modulo end address
MOV     #0x8001,W0
MOV     W0,MODCON     ;enable W1, X AGU for modulo
MOV     #0x000F,W0    ;W0 holds buffer fill value
MOV     #0x11FE,W1    ;point W1 to buffer
DO      #15,FILL      ;fill the 16 buffer locations
MOV     W0,[W1--]     ;fill the next location

FILL:
DEC     W0,W0         ;decrement the fill value

; W1 = 0x11FE when DO loop completes
```

Start Address = 0x11E0
End Address = 0x11FF
Length = 16 Words

3.6 ビット反転アドレッシング

3.6.1 ビット反転アドレッシングについて

ビット反転アドレッシングは、基数 2 の FFT アルゴリズムのデータ レコーディングを単純化します。ビット反転アドレッシングは X WAGU を介してのみサポートされます。ビット反転アドレッシングは、バイナリ値の中央を基準にして各ビットを対称に入れ換える事によって、アドレス ポインタの鏡像を生成します (図 3-7 参照)。表 3-1 に、例として 4 ビットアドレス フィールドのビット反転結果を示します。

図 3-7: ビット反転アドレッシングの例

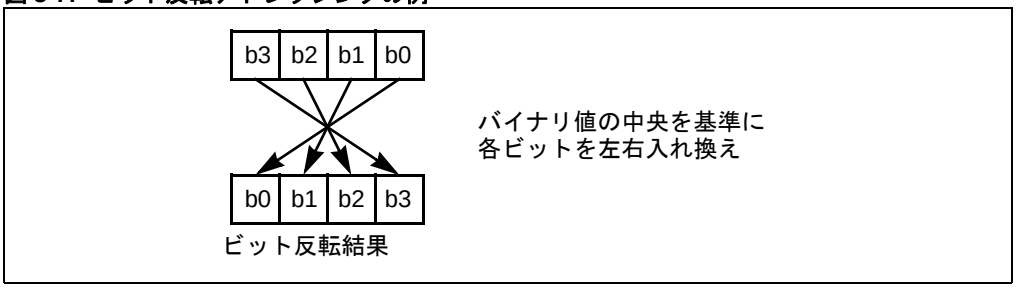


表 3-1: ビット反転アドレッシングの例 (4 ビット フィールド)

反転前 アドレス					反転後 アドレス				
A3	A2	A1	A0	Decimal	A3	A2	A1	A0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	8
0	0	1	0	2	0	1	0	0	4
0	0	1	1	3	1	1	0	0	12
0	1	0	0	4	0	0	1	0	2
0	1	0	1	5	1	0	1	0	10
0	1	1	0	6	0	1	1	0	6
0	1	1	1	7	1	1	1	0	14
1	0	0	0	8	0	0	0	1	1
1	0	0	1	9	1	0	0	1	9
1	0	1	0	10	0	1	0	1	5
1	0	1	1	11	1	1	0	1	13
1	1	0	0	12	0	0	1	1	3
1	1	0	1	13	1	0	1	1	11
1	1	1	0	14	0	1	1	1	7
1	1	1	1	15	1	1	1	1	15

3.6.2 ビット反転アドレッシングの動作

X WAGU のみがビット反転アドレッシングをサポートします。また、MODCON および X 書き込み AGU ビット反転アドレッシング制御 (XBREV) 特殊機能レジスタがビット反転アドレッシングを制御します。ビット反転アドレッシングは下記のように呼び出します。

1. BWM (MODCON<11:8>) 制御ビットを用いて、ビット反転アドレッシングをいずれか 1 つのワーキング レジスタへ割り当てます。
2. BREN (XBREV<15>) 制御ビットをセットしてビット反転アドレッシングを有効にします。
3. XB (XBREV<14:0>) 制御ビットで X AGU ビット反転修飾子を設定します。

ビット反転アドレッシングを有効にすると、プリ / ポスト インクリメント アドレッシング モードによるレジスタ間接を使用する場合 ([Wn++]、[++Wn]) にのみ、ビット反転アドレッシング ハードウェアがビット反転アドレスを生成します。また、ビット反転アドレスはワードモード命令に対してのみ生成されます。その他のアドレッシング モードやバイトモードの命令に対しては、ビット反転アドレッシングは機能しません (非反転の通常アドレスを生成)。

Note:	MODCON レジスタへの書き込みの直後に、ワーキング レジスタを用いて間接読み出しを行う命令を実行してはなりません。予期せぬ結果が生じる可能性があります。一部の命令は、暗黙的に間接読み出しを実行します。これには右記の命令が含まれます : POP、RETURN、RETFIE、RETLW、ULNK

3.6.2.1 モジュロ アドレッシングとビット反転アドレッシング

モジュロ アドレッシングとビット反転アドレッシングは、同一ワーキング レジスタを用いて同時に有効にできます。ただしこの場合、データ書き込みではビット反転アドレッシングが常に優先します。例として、下記のセットアップは、同一ワーキング レジスタをモジュロ アドレッシングビット反転アドレッシングの両方へ割り当てます。

- X モジュロ アドレッシング有効 (XMODEN = 1)
- ビット反転アドレッシング有効 (BREN = 1)
- W1 をモジュロ アドレッシングへ割り当て (XWM<3:0> = 0001)
- W1 をビット反転アドレッシングへ割り当て (BWM<3:0> = 0001)

W1 をポインタとして用いるデータ読み出しでは、モジュロアドレス境界チェックが発生します。W1 をデスティネーション ポインタとして用いるデータ書き込みでは、ビット反転ハードウェアが W1 をデータ レコーディング向けに修正します。

3.6.2.2 XBREV に関連するデータ依存性

BREN ビット (XBREV<15>) がセットされてビット反転アドレッシングが既に有効になっている場合、XBREV レジスタへの書き込みの直後に、ビット反転アドレス ポインタとして指定したワーキング レジスタを用いて間接読み出しを行ってはなりません。

3.6.3 ビット反転修飾子の値

XBREV レジスタへ読み込まれる値は、ビット反転データバッファのサイズを間接的に定義する定数です。表 3-2 に、一般的なビット反転バッファで使用する XB 修飾子の値を示します。

表 3-2: ビット反転アドレス 修飾子の値

バッファサイズ (ワード)	XB ビット反転アドレス修飾子の値
32768	0x4000
16384	0x2000
8192	0x1000
4096	0x0800
2048	0x0400
1024	0x0200
512	0x0100
256	0x0080
128	0x0040
64	0x0020
32	0x0010
16	0x0008
8	0x0004
4	0x0002
2	0x0001

Note: 上に示したビット反転修飾子の値だけが、有効なビット反転アドレス シーケンスを生成します。

ビット反転ハードウェアは、ワーキング レジスタの内容と XB 飾子定数の間で「反転キャリー」加算を行う事によって、ワーキング レジスタ アドレスを変更します。反転キャリー加算は、通常とは逆に左から右へ向かってビットを加算します。いずれかのビット位置でキャリーアウトが発生すると、そのキャリーアウト ビットは右隣のビットへ加算されます。
例 3-5 に、XB 修飾子値として 0x0008 を使用した場合の反転キャリー加算と、その結果のワーキング レジスタ値を示します。XB 修飾子を左へ 1 ビット シフトしてワードアドレス値を生成する事に注意してください。

例 3-5: XB アドレス計算

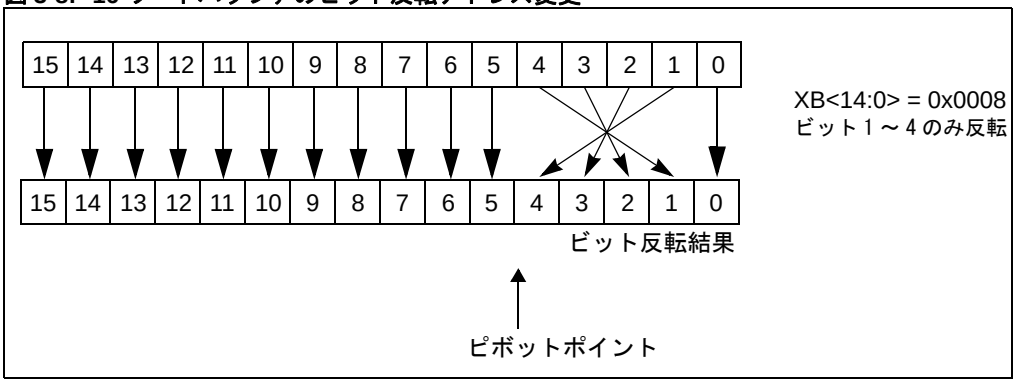
```
0000 0000 0000 0000     Wn points to word 0
           +1 0000      Wn = Wn + XB
-------------------

0000 0000 0001 0000     Wn points to word 8
           +1 0000      Wn = Wn + XB
-------------------

0000 0000 0000 1000     Wn points to word 4
           +1 0000      Wn = Wn + XB
-------------------

0000 0000 0001 1000     Wn points to word 12
           +1 0000      Wn = Wn + XB
-------------------

0000 0000 0000 0100     Wn points to word 2
           +1 0000      Wn = Wn + XB
-------------------

0000 0000 0001 0100     Wn points to word 10
```

XB<14:0> = 0x0008 の場合、ビット反転バッファのサイズは 16 ワードです。ワーキング レジスタのビット 1 ～ 4 はビット反転アドレス訂正の結果に依存しますが、ビット 5 ～ 5(ピボットポイントの外側) はビット反転ハードウエアによる修正は行われません。

ビット反転ハードウェアはワードアドレス上でのみ動作するため、ビット 0 も変更しません。

XB 修飾子は、ビット反転アドレス変更のピボットポイントを制御します。ピボットポイントの外側のビットは、ビット反転アドレッシングの影響を受けません。

図 3-8: 16 ワードバッファのビット反転アドレス変更



3.6.4 ビット反転アドレッシングのサンプルコード

例 3-6 のコードは、連続する 16 個のデータワードを読み出して、そのデータをビット反転順に新たな位置へ書き込みます。W0 は読み出しアドレスポインタ、W1 はビット反転変更を受ける書き込みアドレスポインタです。

例 3-6: ビット反転アドレッシングのサンプルコード

```
; Set XB for 16-word buffer, enable bit reverse addressing
    MOV     #0x8008,W0
    MOV     W0,XBREV
; Setup MODCON to use W1 for bit reverse addressing
    MOV     #0x01FF,W0
    MOV     W0,MODCON
; W0 points to input data buffer
    MOV     #Input_Buf,W0
; W1 points to bit reversed data
    MOV     #Bit_Rev_Buf,W1
; Re-order the data from Input_Buf into Bit_Rev_Buf
    REPEAT  #15
    MOV     [W0++],[W1++]
```

3.7 DMA RAM

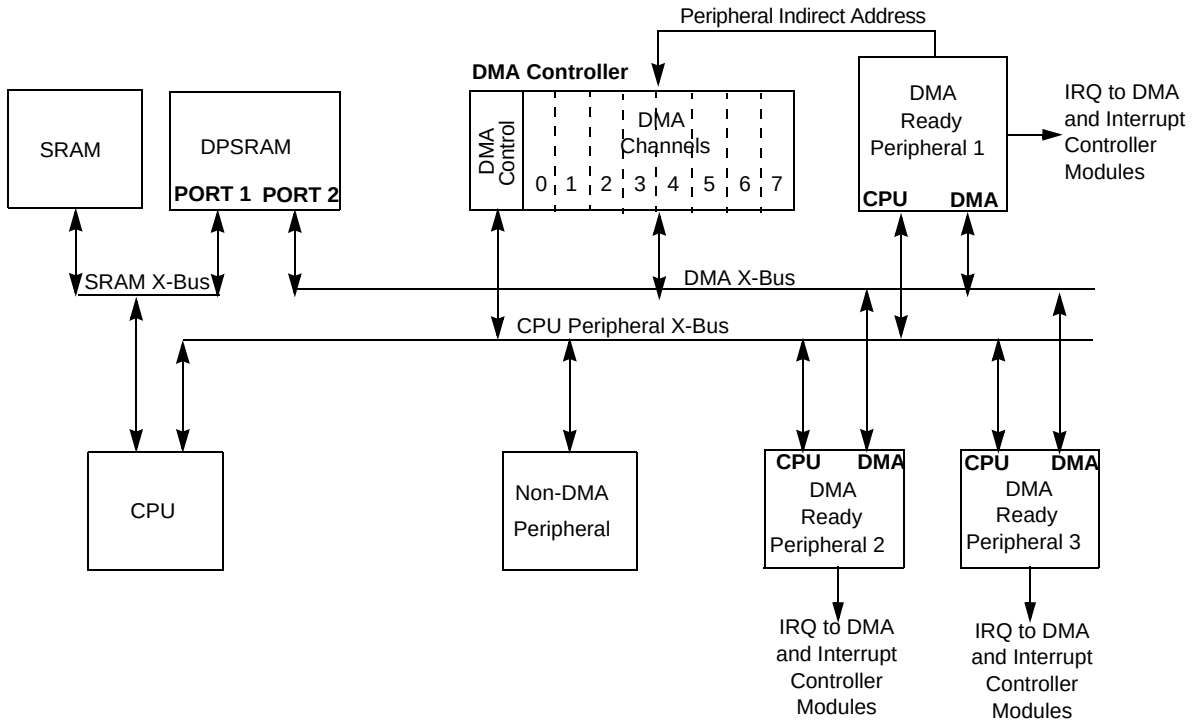
一部の dsPIC33F/PIC24H は、DMA とデュアルポート SRAM メモリ (DPSRAM) を内蔵しています。CPU コントローラと DMA コントローラは、CPU ストール等に影響されずに DSPRAM 内のアドレスに対して読み書きを行えます。これにより、リアルタイム性能が最大限に高められます。

Note:	DMA RAM の実装の有無とサイズはデバイスによって異なります。詳細は各 dsPIC33F/PIC24H のデバイス データシートを参照してください。

図 3-9 は、dsPIC33F/PIC24H 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 X バスを介して通常の SRAM と通信します。これに加えて CPU は、別の周辺 X バスを介して周辺モジュールとも通信します。これらの X バスは共に X データ領域内に備わっています。

各 DMA チャンネルは、DMA 専用バスを介して、DPSRAM の PORT 2 および各 DMA 対応周辺モジュールの DMA ポートと通信します。詳細は**セクション 22.「ダイレクトメモリ アクセス (DMA)」**(DS70182) を参照してください。

図 3-9: DMA コントローラのブロック図



Note: 図を見やすくするために､CPU および DMA アドレスバスは表示していません。

3.8 レジスタマップ

表 3-3 に、データメモリ制御レジスタに対するビット機能の割り当てを示します。

表 3-3: データメモリ レジスタの割り当て

ファイル名	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
MODCON	XMODEN	YMODEN	—	—	BWM<3:0>				YWM<3:0>				XWM<3:0>				0000
XMODSRT	XS<15:1>															0	xxxx
XMODEND	XE<15:1>															1	xxxx
YMODSRT	YS<15:1>															0	xxxx
YMODEND	YE<15:1>															1	xxxx
XBREV	BREN	XB<14:0>															xxxx

凡例： x = リセット時に不明の値、— = 未実装、「0」として読み出し、リセット値は 16 進数で表記

3.9 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/PIC24H デバイス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。本モジュールに関連する最新のアプリケーション ノートは以下の通りです。

タイトル	アプリケーション ノート番号

現在、関連するアプリケーション ノートはありません。

Note: dsPIC33F/PIC24H デバイスファミリ向けのその他のアプリケーション ノートとサンプルコードは、マイクロチップ社のウェブサイト (www.microchip.com) でご覧になれます。

3.10 改訂履歴

リビジョン A (2007 年 3 月)

本書の初版

リビジョン B (2007 年 4 月)

本書全体の小規模な更新

リビジョン C (2010 年 1 月)

このリビジョンでの変更内容は次の通りです。

- ファミリ リファレンス マニュアルのタイトルを dsPIC33F から dsPIC33F/PIC24H へ変更
- 本書内の「dsPIC33F」を全て「dsPIC33F/PIC24H」へ変更
- 図
  - 図 3-3 内のコード行の変更

```
MAC W4*W5, A, W4, [W8]+=2, W5, [W10]+=2, [W13]+=2
```

    を下記へ変更

```
MAC W4*W5, A, [W8]+=2, W4, [W10]+=2, W5, [W13]+=2
```

  - 図 3-5 内のコード行の変更

```
DO #49, FILL        ;fill the 50 buffer locations
FILL:
MOV    W0,[W1++]    ;fill the next location
```

    を下記へ変更

```
REPEAT #49,         ;fill the 50 buffer locations
MOV    W0,[W1++]    ;fill the next location
```

- Notes
  - **3.1.1「ニアデータメモリ」**の網掛けボックス内の注釈で、「dsPIC30F/33F」を「16 ビット MCU および DSC」へ変更
  - **3.5.1.2「モジュロ終了アドレス」**の網掛けボックス内の注釈で、「2 の複数乗」を「2 のべき乗」へ変更
- セクション
  -「制御レジスタ」セクションを追加 (**3.2「制御レジスタ」**参照)
  -「レジスタ」セクションを **3.3「制御レジスタの説明」**として再編成
  -「レジスタマップ」セクションを追加 (**3.8「レジスタマップ」**参照)
- **3.4.3「アドレス生成ユニットと DSP クラス命令」**内で、「dsPIC30F/33F」を「16 ビット MCU および DSC」へ変更
- 上記に加えて、表現および体裁の変更等、本書全体の細部を修正

ISBN: 978-1-60932-555-8

NOTES: