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 36. プログラマブル巡回冗長検査 (CRC)

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。

36.1 はじめに

dsPIC33F のプログラマブル巡回冗長検査 (CRC) モジュールは、ソフトウェアによるコンフィグレーションが可能な CRC チェックサム ジェネレータです。チェックサムとは、複数バイトを格納した個々のメッセージまたはデータブロックに関連付けられる固有の数値です。通信用のデータパケットにせよメモリに保存されたデータブロックにせよ、チェックサム等の情報は、それらのデータを処理する前の検証に役立ちます。

最も単純な方法として、メッセージ内のデータバイトを全て加算する事によってチェックサムを計算できます。しかし、バイト単位で反転したり並び順を入れ換えたりしてメッセージが改変された場合、このような方法では誤りを検出できません。また、この方法ではメッセージの任意位置にヌルバイトが追加された場合も検出できません。

CRC はこのような単純な方法に比べると複雑ですが、信頼性の高い誤り検出アルゴリズムを提供します。CRC アルゴリズムはメッセージをバイナリ ビットストリームとして扱い、これを決められたバイナリ値で除算した時の余りをチェックサムとみなします。通常の除算と同様に、CRC 計算も反復処理です。ただし、これらの演算は mod2 によるモジュロ演算で行われるという点で異なります。例えば、除算は XOR 演算 (すなわちキャリーなしの減算) に置き換えられます。CRC アルゴリズムは、全ての計算に多項式の項を使用します。除数、被除数、余りのバイナリ値の各桁は、多項式の各項の係数に割り当てられます。例えば、値 19h (11001) は式 36-1 のように表現されます。

式 36-1:

$$(1 \bullet x^4) + (1 \bullet x^3) + (0 \bullet x^2) + (0 \bullet x^1) + (1 \bullet x^0)$$

従って

$$x^4 + x^3 + x^0$$

CRC 計算では、まず適当な除数を選択します。この除数は生成多項式と呼ばれます。生成多項式によって CRC の誤り検出能力が決まるため、アプリケーションに合わせて適切な長さと項を持つ生成多項式を選択する必要があります。多くのアプリケーションで広く用いられている標準的な多項式もいくつか存在します。しかし本書では、特定の多項式の誤り検出能力については説明しません。

CRC 演算は反復処理であるため、ソフトウェアで処理するとかなりの CPU 処理能力を消費します。dsPIC33F が内蔵する CRC ハードウェア モジュールは、ソフトウェアによるコンフィグレーションが可能であり、最小限のソフトウェア オーバーヘッドで高速に CRC チェックサムを計算できます。

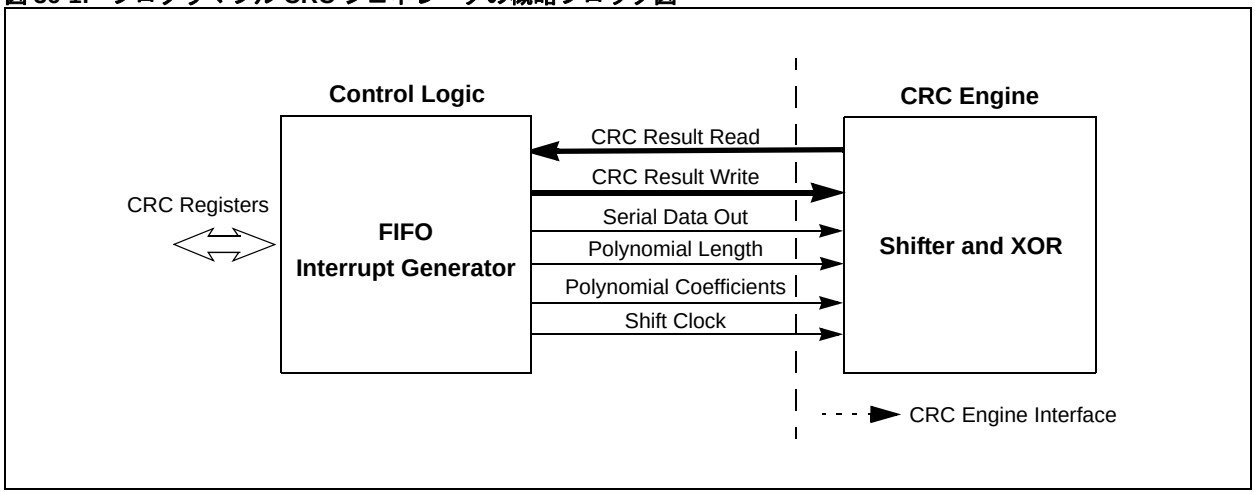
プログラマブル CRC モジュールの主な特長を以下に挙げます。

- ビット長を設定可能な CRC 生成多項式 (最大 16 ビット長)
- プログラマブルな CRC 生成多項式
- 割り込み出力
- 8 段 x 16 ビットまたは 16 段 x 8 ビットのデータ入力用 FIFO

36.2 モジュールの概要

dsPIC33F のプログラマブル CRC モジュールは、2 つの論理ブロック (制御ロジックと CRC エンジン) で構成されます。制御ロジックはレジスタ インターフェイス、FIFO、割り込みジェネレータ、CRC エンジン インターフェイスを備えます。CRC エンジンは CRC カリキュレータを備え、これは XOR 機能を備えたシリアルシフタを使用して実装されます。CRC モジュールの概略ブロック図を図 36-1 に示します。

図 36-1: プログラマブル CRC ジェネレータの概略ブロック図



36.3 CRC レジスタ

以下では、CRC モジュールに関連するレジスタについて説明します。これらのレジスタは特殊機能レジスタ (SFR) として dsPIC33F デバイスのデータ RAM 空間に割り当てられます。

- **CRCCON: CRC 制御レジスタ**
- **CRCXOR: CRC XOR レジスタ**
- **CRCDAT: CRC データ入力レジスタ**
- **CRCWDAT: CRC シフト書き込みレジスタ**

CRCCON レジスタ (レジスタ 36-1 参照) は、CRC モジュール用レジスタの中で主要となる制御 / ステータスレジスタです。CRCXOR レジスタ (レジスタ 36-2 参照) では、多項式に使用する項を選択する事によって生成多項式を定義します。CRCDAT および CRCWDAT レジスタは、それぞれデータ入力用および結果出力用のバッファです。

レジスタ 36-1: CRCCON: CRC 制御レジスタ

U-0	U-0	R/W-0	R-0	R-0	R-0	R-0	R-0
—	—	CSIDL	VWORD<4:0>				
bit 15							bit 8

R-0	R-1	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CRCFUL	CRCMPT	—	CRCGO	PLEN<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 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 15-14 **未実装 :**「0」として読み出し

bit 13 **CSIDL:** アイドルモード時 CRC 停止ビット
1 = デバイスがアイドルモードに移行した時にモジュールの動作を停止する
0 = アイドルモード中もモジュールの動作を継続する

bit 12-8 **VWORD<4:0>:** FIFO ポインタ値ビット
FIFO 内の有効なワード数またはバイト数を示します。PLEN<3:0> が 8 以上であれば最大数は 8、PLEN<3:0> が 7 以下であれば最大数は 16 です。

bit 7 **CRCFUL:** FIFO フルビット
1 = FIFO はフルである
0 = FIFO はフルではない

bit 6 **CRCMPT:** FIFO エンプティビット
1 = FIFO はエンプティである
0 = FIFO はエンプティではない

bit 5 **未実装 :**「0」として読み出し

bit 4 **CRCGO:** CRC 開始ビット
1 = CRC シリアルシフトを開始する
0 = FIFO がエンプティになった後に CRC シリアルシフトを停止する

bit 3-0 **PLEN<3:0>:** 多項式長ビット
生成多項式の長さ = PLEN<3:0> 値 + 1

レジスタ 36-2: CRCXOR: CRC XOR 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
X<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	U-0
X<7:1>							—
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 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 15-1 **X<15:1>:** 多項式項 n の XOR イネーブルビット
1 = n 次項 (x^n 項) を多項式に含める (XOR 演算を有効にする)
0 = x^n 項を多項式に含めない

bit 0 **未実装 :**「0」として読み出し

36.4 CRC エンジン

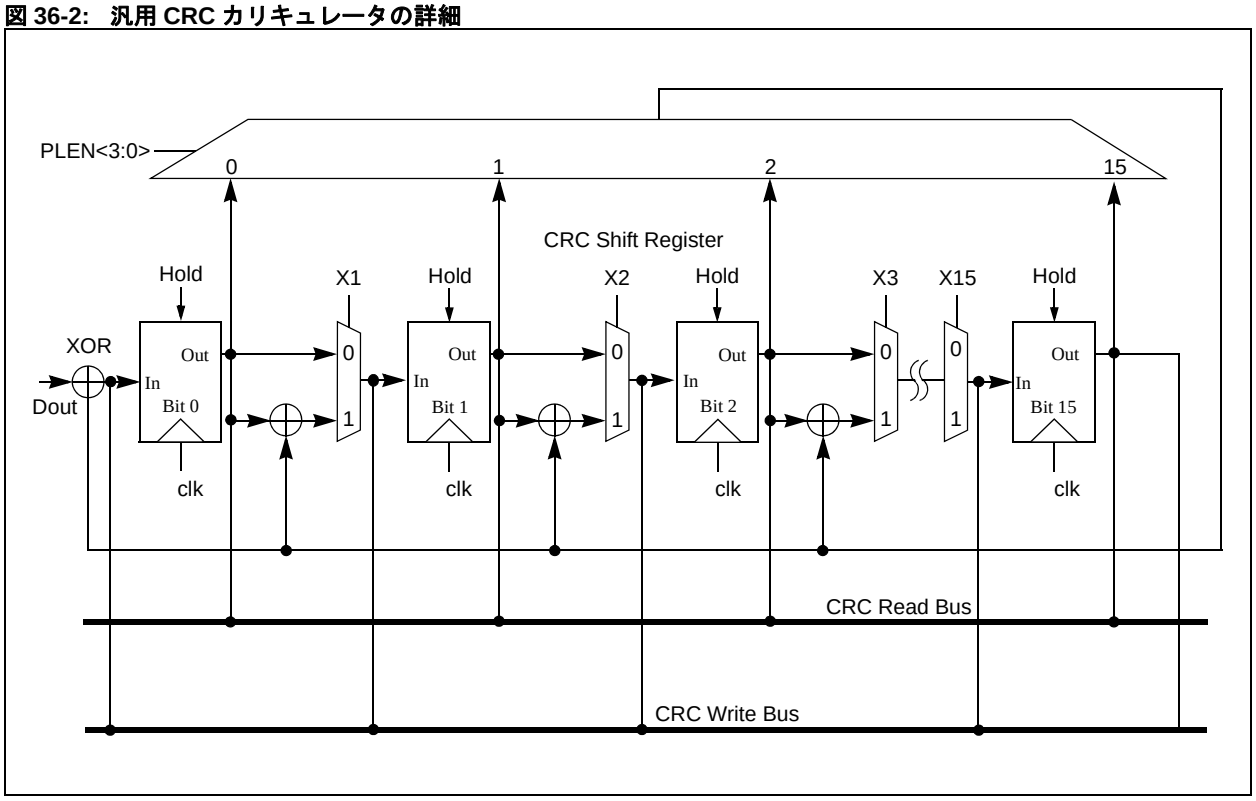
36.4.1 CRC エンジンの概要

CRC エンジンは、マルチプレクサの設定によってフィードフォワード / フィードバック位置を設定可能なシリアルシフト式 CRC カリキュレータです。汎用 CRC カリキュレータの接続図を図 36-2 に示します。

CRC アルゴリズムは、バイナリ除算のかわりに XOR 論理演算を使用する事により、計算処理を単純化します。生成多項式の係数の設定には CRCXOR<15:1> ビットを使用します。いずれかのビットに「1」を書き込むと、多項式の対応する項の XOR 演算が有効化されます。多項式の長さの設定には、CRC 制御レジスタ (CRCCON<3:0>) の多項式長 (PLEN<3:0>) ビットを使用します。PLEN<3:0> の値が示す多項式の長さに基づいてマルチプレクサを切り換え、フィードバックを始めるタップ位置を指定します。

CRC 計算の結果は、CRC 読み出しバスを介してホールドレジスタを読み出す事によって取得できます。CRC 書き込みバスを介する CRC シフトレジスタへの直接書き込みパスも用意されています。CPU は CRC シフトレジスタ書き込み (CRCWDAT) レジスタを介してこのパスにアクセスします。

図 36-2: 汎用 CRC カリキュレータの詳細



36.4.2 CRC エンジンのソフトウェア コンフィグレーション

CRC エンジンは、必要とされる生成多項式に基づいてソフトウェアで正しく設定する必要があります。生成多項式はビット数 n の 16 進数で表現されます。この数の最上位ビット (MSb) は x^n 項、最下位ビット (LSb) は x^1 項の係数に対応します。最上位ビット (MSb) は常に「1」であるとみなされます。x^0 項の係数は常に「1」であるとみなされるため、省略されています。これら 2 つの項以外の x^{n-1} ～ x^1 項の係数を CRCXOR レジスタで設定する必要があります。

ここでは式 36-2 に示す CRC 多項式を例として取り上げます。

式 36-2:

$$x^{16} + x^{12} + x^5 + 1$$

多項式の長さは多項式の最高次数によって表されます。従って式36-2 の多項式の長さは 16 ビットです。各項の係数は「0」または「1」に設定されます。

この多項式を CRC ジェネレータにプログラミングするには、PLEN ビット (CRCCON<3:0>) と CRCXOR<15:1> を表 36-1 のように設定する必要があります。

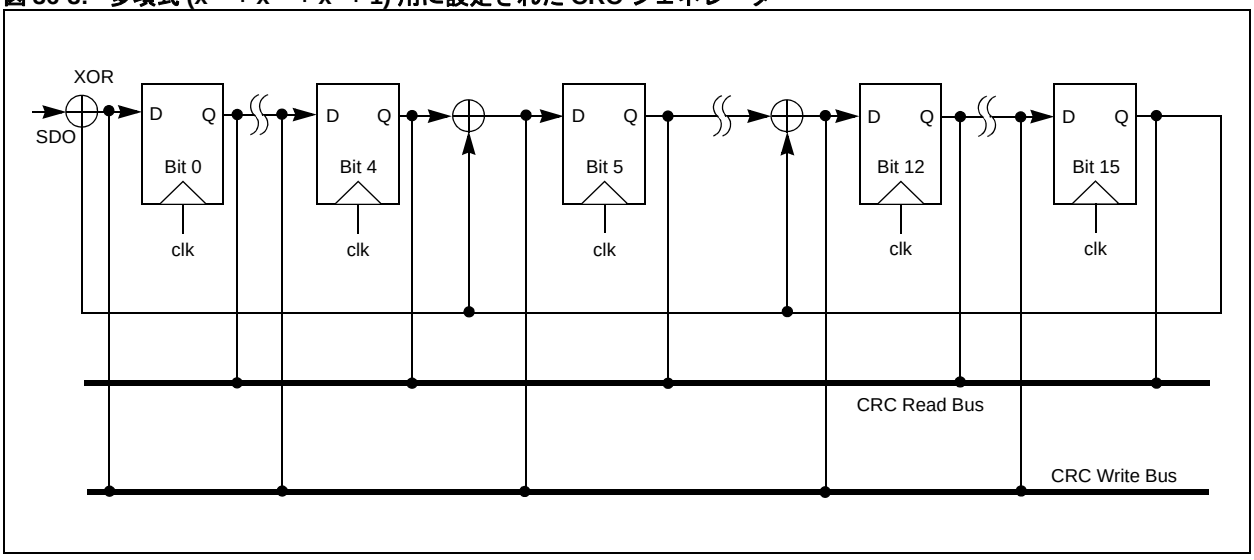
表 36-1: CRC のセットアップ例

レジスタ名	ビット名	ビット値
CRCCON	PLEN<3:0>	0Fh
CRCXOR	X<15:1>	1020h

この多項式の長さは 16 (PLEN<3:0> + 1) です。上記の多項式を設定するために、表 36-1 では X<15:1> の第 12 ビットと第 5 ビットが「1」にセットされています (値 1020h)。Bit 0 は常に XOR 演算されます。また、16 ビット多項式の場合、第 16 ビットも必ず XOR 演算されます。このため bit 0 と bit 16 に対応する CRCXOR ビットは存在しません。

上記の多項式用に設定された CRC ジェネレータの接続図を図 36-3 に示します。

図 36-3: 多項式 ($x^{16} + x^{12} + x^5 + 1$) 用に設定された CRC ジェネレータ



Note: CRCXOR レジスタの x^0 項に対応するビットの値は無視され、この項の係数は常に「1」であるとみなされます。従って、CRXOR レジスタの最下位ビット (LSb) が「0」であっても「1」であっても (例 : 1020h であっても 1021h であっても)、CRC 計算の結果には影響しません。

36.5 制御ロジック

36.5.1 FIFO

FIFO は物理的には 8 段 x 16 ビット幅のストレージ エレメントです。FIFO 関連のロジックは、VWORD (VWORD<4:0> または CRCCON<12:8>) と呼ばれる 5 ビットカウンタを備えます。VWORD<4:0> ビットの値は FIFO に新たに書き込まれたデータエレメントの数を示します。

PLEN<3:0> が 8 以上の場合、FIFO は 8 段 x 16 ビット幅のアレイとして機能します。それ以外の場合、FIFO は 16 段 x 8 ビット幅のアレイとして機能します。CRC 計算を開始する前に、CPU は CRC データ (CRCDAT) レジスタを介して FIFO にデータを書き込む必要があります。データは、必ず CRCDAT レジスタに書き込む必要があります。CRCDAT レジスタを読み出す事はできません。読み出しても常にゼロが返されます。

FIFO への書き込みにおける最小データ単位は 1 バイトです。PLEN<3:0> が 7 以下である場合、VWORD の値は、FIFO に 1 バイトを書き込むたびに 1 つインクリメントし、1 ワードを書き込むたびに 2 つインクリメントします

PLEN<3:0> が 8 以上である場合、VWORD の値は、FIFO に 1 ワードを書き込むたびに 1 つインクリメントします。この場合、CRCDAT レジスタに 1 バイトだけを書き込んでも VWORD の値はインクリメントしません。VWORD の値は、CRCDAT レジスタに 2 バイト (1 ワード) が書き込まれた時にだけインクリメントします。

VWORD レジスタの値が 8 (8 段 x 16 ビット FIFO の場合) または 16 (16 段 x 8 ビット FIFO の場合) に達すると、FIFO フル (CRCFUL) ビットがセットされ、FIFO がフルである事が示されます。アプリケーションは、新たな値を CRCDAT レジスタに書き込む際に、FIFO がフルではない事を確認する必要があります。

データのブロックを処理する場合、正しい CRC チェックサム結果を生成するために、アプリケーションはデータブロックの最終ワードを CRC レジスタから確実にシフトアウトさせる必要があります。このために、データブロックの最終ワードを CRCDAT レジスタに書き込んだ後に、0x0000 を CRCDAT レジスタに書き込む必要があります。加えて、アプリケーションは、このワードを CRCDAT レジスタに書き込む際に、FIFO がフルではない (CRCFUL = 0) 事を確認する必要があります。

36.5.2 CRC エンジン インターフェイス

36.5.2.1 FIFO から CRC カリキュレータへのインターフェイス

FIFO から CRC カリキュレータへのシリアルシフトを開始するには、CRC 開始 (CRCGO) ビットをセット (CRCCON<4> = 1) する必要があります。CRCGO ビットが「1」かつ VWORD の値が 1 以上の場合にのみ、シリアルシフタは MSb を先頭に CRC エンジンへのデータシフトを開始します。シフト開始前に CRCFUL ビットがセットされていた場合、このビットは VWORD が 1 つデクリメントした時にクリアされます。FIFO の 1 段を CRC カリキュレータへ完全にシフトし終えると、VWORD が 1 つデクリメントします。シリアルシフタは VWORD がゼロになるまでシフトを続け、VWORD がゼロになると FIFO エンプティ (CRCMPT) ビットをセットして FIFO がエンプティである事を示します。

CRC シフトクロック周波数は、dsPIC33F 命令クロック周波数の 2 倍です。従ってこのハードウェア シフト処理はソフトウェア シフタよりも高速です。アプリケーションは、シフト実行中に FIFO に書き込む事ができます。CRC エンジンへ連続的にデータを転送するために、FIFO に十分な数のワードまたはバイトを事前に書き込んでおく事を推奨します。このように FIFO を準備した後に、アプリケーションは CRCGO ビットを「1」にセットして CRC シフトを開始できます。シフト開始後に、VWORD または CRCFUL ビットを監視する必要があります。CRCFUL ビットがクリア状態または VWORD の値が 8 または 16 未満であれば、FIFO に追加のワードを書き込む事ができます。CRCDAT レジスタに書き込んだ後、少なくとも 1 命令サイクルが経過しないと、VWORD の値を読み出す事はできません。

FIFOに書き込み済みのワードを全てシフトアウトしてFIFOをエンプティにするには、CRCGO ビットを「1」にセットし、CRCMPT ビットがセットされるまで CRC シフタを動作させる必要があります。CRC シフタが動作を開始した後に CRCGO ビットをクリアしても、データのシフトは停止しません。CRCGO ビットを「0」にクリアした場合、FIFO がエンプティになった後に CRC シフタが停止します。その後、CRCGO ビットを再セットするまで、FIFO にデー

タを書き込んでもそのデータはシフトアウトされません。CRCGO ビットをセットしたままにすると、CRC シフタは停止する事なくシフトアウトし続けます。この場合、FIFO にデータを書き込むと即座にそのデータが処理されます。

Note:	PLEN<3:0> が 8 以上の場合、アプリケーション ソフトウェアで CRCGO ビットをセットする前に、FIFO に整数個のワード (偶数個のバイト) を書き込む必要があります。FIFO に奇数個のバイトを書き込んだ後に CRCGO ビットをセットすると、最後の 1 バイトはシフトアウトされず、CRCMPT ビットは常時「0」となって FIFO がエンプティではない事を示し続けます。

36.5.2.2 FIFO からシフトされるデータのビット数

シフトされるデータのビット数は、選択した多項式の長さによって決まります。例えば PLEN<3:0> = 5 の場合、生成多項式の長さ (すなわち 1 データのサイズ) は 6 ビット (PLEN<3:0> + 1)です。CPUはバイト単位で書き込む事しかできませんが、FIFOは1バイトの一部だけをシフトアウトします。この例の場合、第6ビット(このデータのMSb)を先頭に1バイト中の 6 ビットだけをシフトアウトします。各バイトの上位 2 ビットはドントケアビットです。従って、CRC計算の実行に要するシフトクロック サイクル数は、[(PLEN<3:0> + 1) • VWORD] です。同様に、12ビット多項式を選択した場合、1ワード中の第12ビット(このデータのMSb)を先頭に12ビットだけをシフトアウトします。従って各ワードの上位4ビットは無視されます。

Note:	n ビット多項式を使用する CRC 計算には整数個の n ビットデータを使用します。例えば 16 ビット多項式の場合、CRC 計算には整数個のワードを使用します。

36.5.2.3 CRC 結果

CPU が CRCWDAT レジスタを読み出すと、CRC 結果が CRC 読み出しバスを介してシフトレジスタから直接読み出されます。CRC 結果は、全てのデータが処理されて CRCIF (IFS4<3>) ビットがセットされた後に読み出す必要があります。CRC FIFO がエンプティになる前に次のデータが CRC モジュールに供給されない場合、データ処理シーケンスの途中で CRCIF フラグがセットされる可能性がある事に注意が必要です。

CRC 書き込みバスを介する CRC シフトレジスタへの直接書き込みパスも用意されています。CPU は、CRCWDAT レジスタを介して、このパスにアクセスします。これにより、シフト処理を開始する前に CRCWDAT レジスタに必要な値を書き込む事ができます。

Note:	CPU が CRCWDAT レジスタを介してシフトレジスタに直接書き込む時、CRCGO ビットは「0」である事が必要です。

36.5.3 割り込み動作

FIFO から CRC エンジンへのデータのシリアルシフトは、CRCGO ビットがセットされ、かつ VWORD<4:0> ビットがゼロより大きい時に開始されます。シフト処理中に CRCMPT ビットが「0」(非エンプティ) から「1」(エンプティ) に変化するか、VWORD<4:0> ビットがゼロに変化すると、CRCIF 割り込みフラグがセットされます。CRCIE ビットがセット (CRC 割り込みが有効化) されていれば、CRCIF ビットがセットされた時に割り込みが生成されます。この場合、CRC 割り込みルーチンから CRCWDAT レジスタを読み出す事ができます。結果は、全てのデータワードが処理された後に読み出す必要があります。次の CRC 動作要求に備えて CRCGO (CRCCON<4>) ビットをクリアする事ができます。

CRC モジュールに関連する割り込みレジスタの詳細は、**36.9「レジスタマップ」**に記載した表 36-2 を参照してください。割り込みと割り込み優先度の設定に関する詳細は、**セクション 32.「割り込み (パート III)」**(DS70214) を参照してください。

Note:	CRCFUL ビットがセットされている時に新たなデータを CRCDAT レジスタに書き込むと、VWORD ポインタが「0」に戻ってロールオーバーします。しかしこの条件では CRC 割り込みフラグ (CRCIF) はセットされません。CRCFUL ビットはリセットされ、FIFO に書き込まれていたデータは失われ、CRCDAT に新たに書き込んだデータが FIFO の先頭位置に書き込まれます。FIFO の残りの位置はエンプティとなり、それらの位置には新たなデータを書き込む事ができます。

36.5.4 CRC モジュールの動作例

以下では、長さ 16 (PLEN<3:0> = 0xF) の CRC 生成多項式 (式 36-3) を使用する CRC モジュールのコンフィグレーションを例として取り上げます。CRCXOR レジスタは、この多項式に対応する値 (0x0800E) を格納します。PLEN の値に従って 16 ビットが CRC シフトレジスタを通してシフトされるため、FIFO は 16 ビット x 8 段 FIFO として機能します。

式 36-3: 生成多項式

$$x^{16} + x^{15} + x^3 + x^2 + x^1 + 1$$

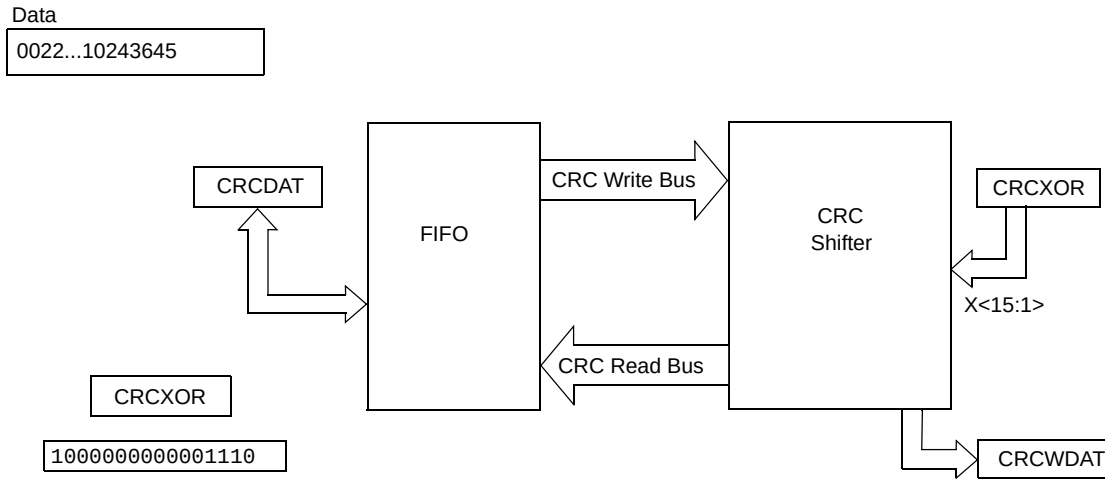
以下では、プロセスごとに図を示しながら CRC モジュールの動作を解説します。

1. CRC モジュールは 16 ビット CRC 生成多項式用に設定されています。CRCMPT ビットの値が「1」であるため、CRC FIFO はエンプティです。

図 36-4: CRC モジュール (PLEN = 0xF)

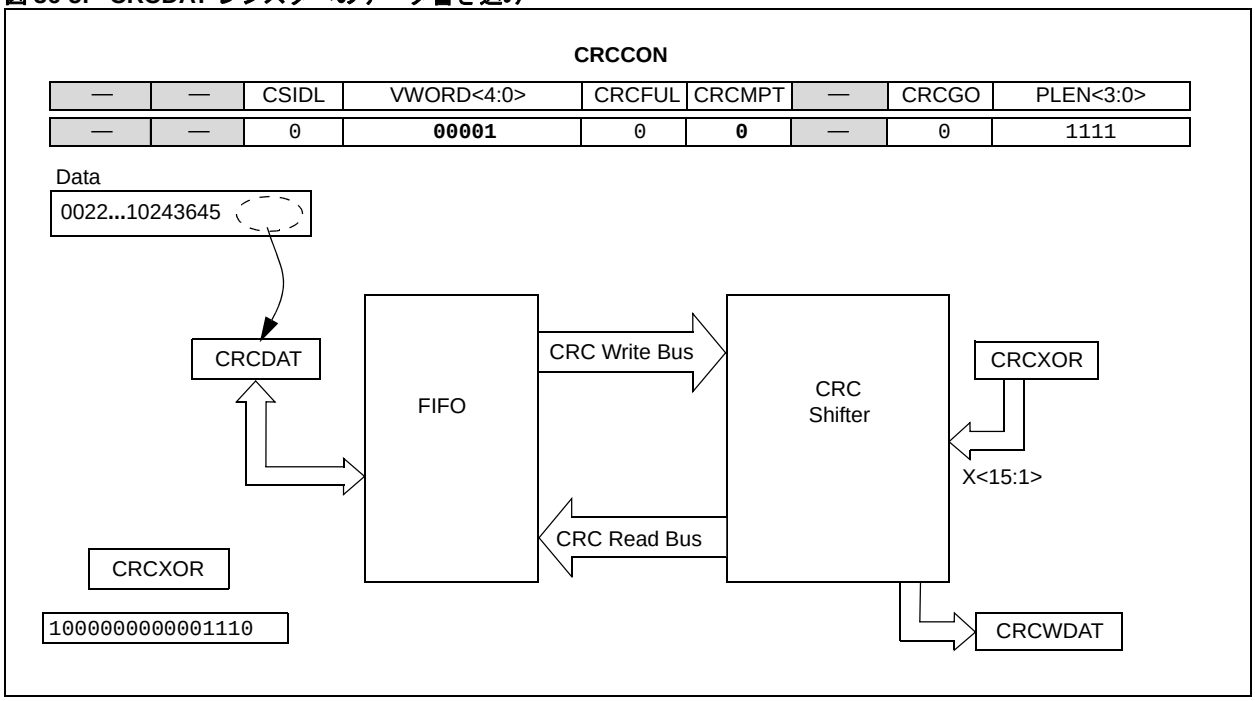
CRCCON

—	—	CSIDL	VWORD<4:0>	CRCFUL	CRCMPT	—	CRCGO	PLEN<3:0>
—	—	0	00000	0	**1**	—	0	**1111**



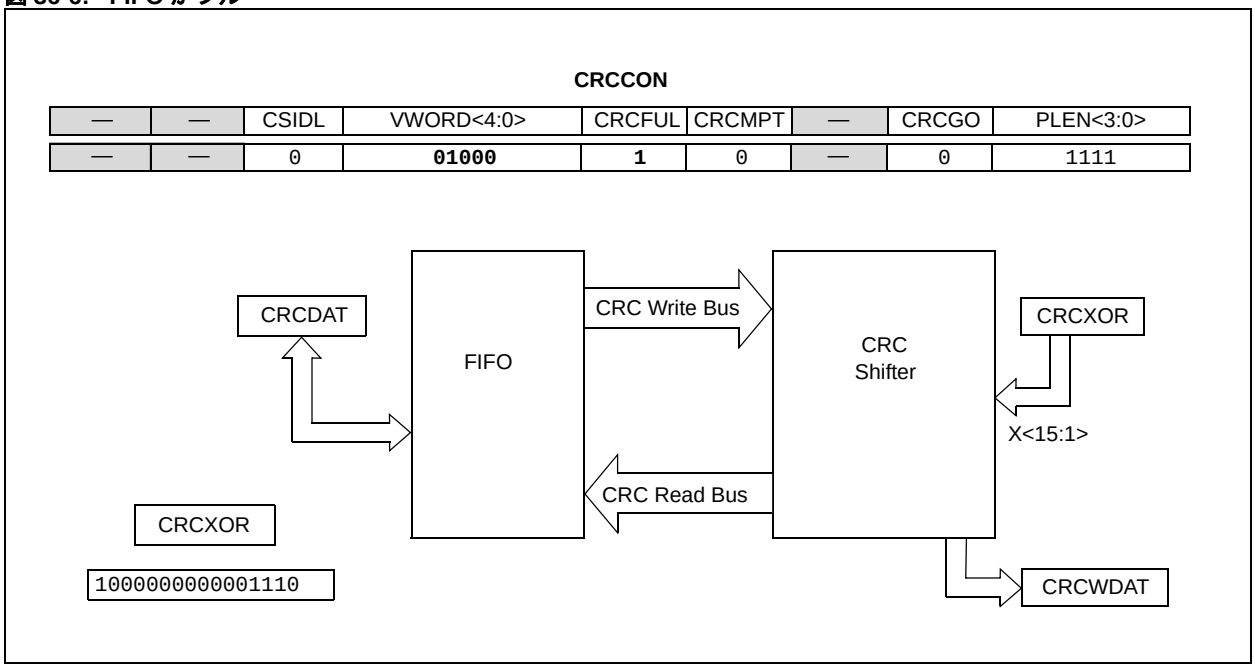
2. アプリケーションが CRCDAT レジスタへのデータ書き込みを開始します。VWORD ビットがインクリメントし、CRCDAT レジスタに書き込まれたワードの数を示します。CRCMPT ビットがクリアされ、CRC FIFO はエンプティではない事を示します。

図 36-5: CRCDAT レジスタへのデータ書き込み



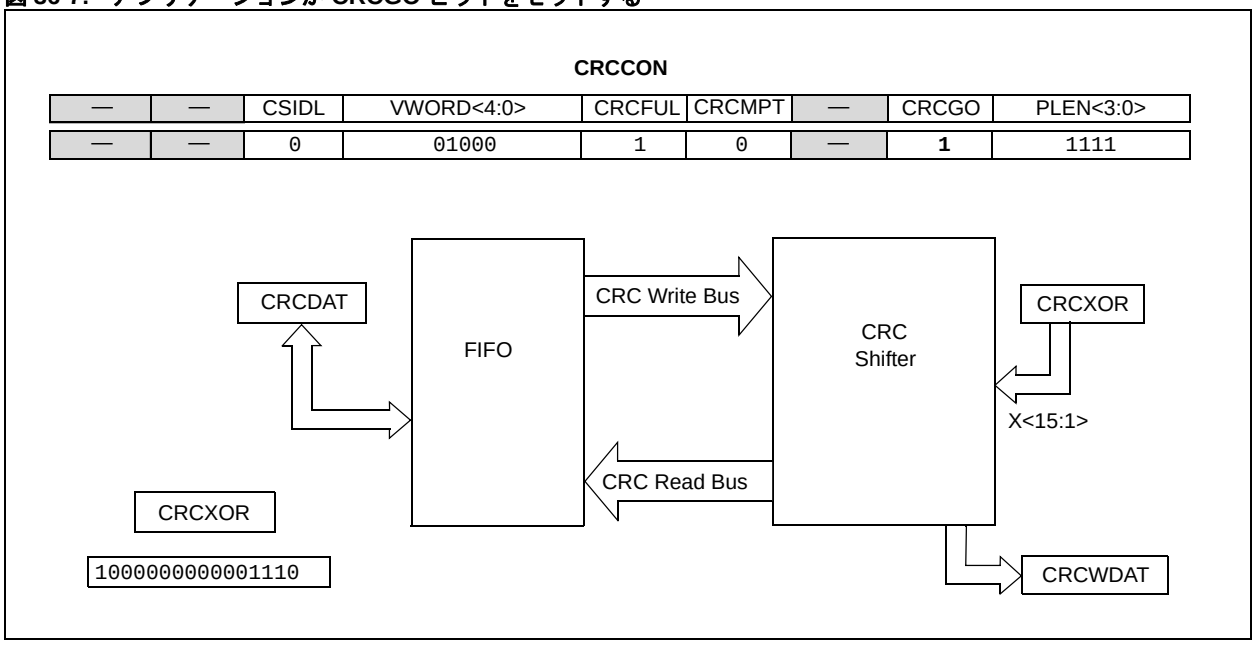
3. 8 ワードが FIFO に書き込まれた時に、VWORD が 0x8 として読み出され、CRCFUL ビットがセットされます。この時点で、CRC モジュールは 8 ワードのデータブロックを処理できる状態となりました。

図 36-6: FIFO がフル



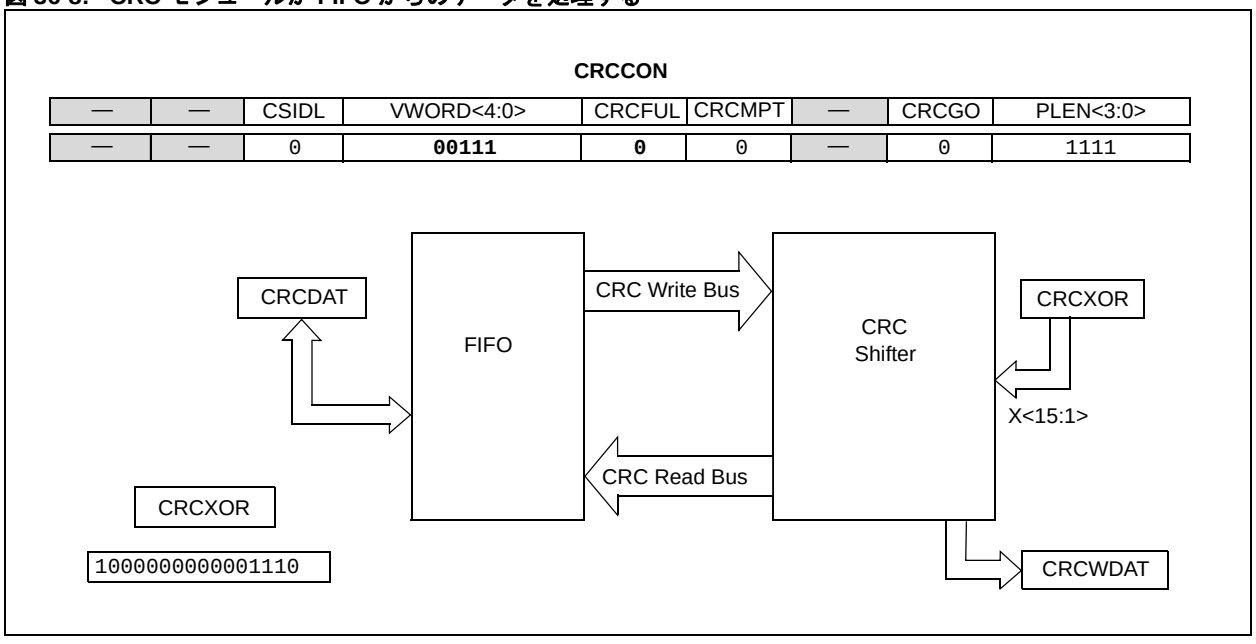
4. アプリケーションが CRCGO ビットをセットし、CRC シフトレジスタを通してワードのシフトを開始するように CRC モジュールに指示します。

図 36-7: アプリケーションが CRCGO ビットをセットする



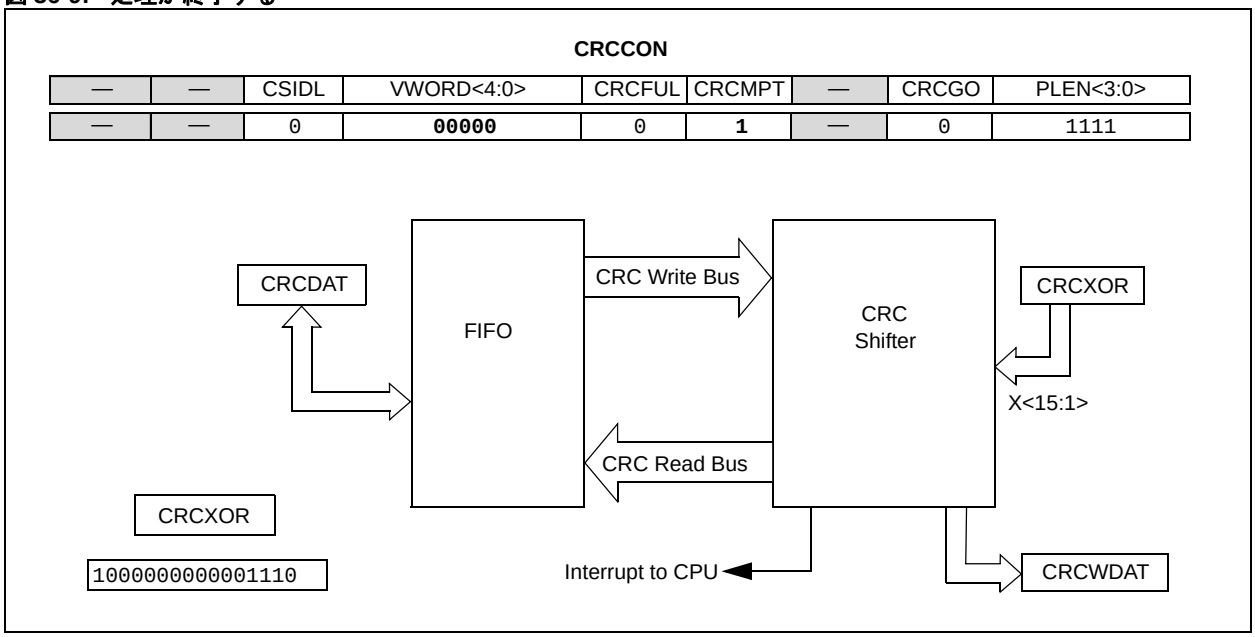
5. CRC シフトレジスタが FIFO からのワードのシフトを開始します。処理されたワード数に応じて VWORD ビットがデクリメントします。CRC モジュールが FIFO から CRC シフトレジスタへのデータ転送を開始すると、CRCFUL ビットがクリアされます。

図 36-8: CRC モジュールが FIFO からのデータを処理する



6. CRC モジュールが 8 ワードのデータ処理を完了し、割り込みを生成します。次いで CRC チェックサム結果を CRCWDAT レジスタに保存します。最後に CRCMPT ビットが CRC FIFO がエンプティである事を示します。

図 36-9: 処理が終了する



36.6 プログラマブル CRC モジュールの利点

CRC アルゴリズムは簡単にソフトウェアに組み込めます。しかしシフト、ビット検査、XOR 演算等の基本要件を実装するには、かなりの CPU 処理能力が必要です。さらに、CRC 計算は対話型処理であるため、データ転送命令による追加のソフトウェア オーバーヘッドがデバイスの MIPS 要件にとって重荷となります。

dsPIC33F の CRC エンジンは、CPU に負荷をかけずに CRC チェックサムを計算します。これはソフトウェアで実装するよりも大幅に高速です。CRC シフトクロックは dsPIC33F の命令クロックより 2 倍高速であるため、計算には 1 ビットあたり 1/2 命令サイクルしか要しません。例えば、128 ビット (16 ビット x8) の長さを持つメッセージの CRC チェックサムを計算する場合、CRC ハードウェア エンジンは 64 命令サイクルしか必要としません。同じ計算をソフトウェアで実行すると、最適化されたコードを使用したとしても 1000 命令サイクル以上が必要です。

36.7 CRC モジュールの適用

CRC は、複数のバイトまたはワードを格納したメッセージのデジタル通信に、信頼性の高い誤り検査アルゴリズムを提供します。計算後にチェックサムを付加したメッセージが送信されます。受信者は、受信したメッセージのチェックサムを計算する事によってデータの完全性を検証します。

36.7.1 CRC 計算のバリエーション

dsPIC33F の CRC モジュールは、MSb を先頭にデータをシフトアウトします。この方式は XMODEM プロトコルでも採用されているように広く一般的に用いられる方式です。しかし、例えば CCITT プロトコルの CRC 計算のように LSb を先頭にシフトアウトする場合もあります。この方式では、メッセージ多項式を dsPIC33F の CRC ハードウェア モジュールに供給する前にソフトウェアでビットを反転する必要があるため、ソフトウェア オーバーヘッドがかなり増加します。本書では個々の CRC 計算バリエーションについて説明しませんが、dsPIC33F のプログラマブル CRC モジュールを使用する事により、最小限のソフトウェア オーバーヘッドで各種の CRC バリエーションを実装できます。

多項式の長さと項の選択はアプリケーションによって異なります。各種の標準的な実装では、多項式の長さとして通常 5、7、8、10、12、16 を使用します。dsPIC33F の CRC モジュールには、各種の多項式長さと計算式を設定できます。n ビットの多項式を選択した場合、通常は n 個のゼロをメッセージ ストリームに付加しますが、この処理にもバリエーションが存在します。次のセクションに記載した CRC 計算の推奨手順では、n ビット多項式のメッセージ ストリームに n 個のゼロを付加します。ユーザは、ゼロまたはゼロ以外の任意の値を選択してメッセージ ストリームに付加できます。アプリケーションの要求に応じて、ユーザはどのような値でも自由に付加する事ができます。

36.7.2 8 ビット多項式

8 ビット多項式を使用する CRC 計算の推奨手順は以下の通りです。

1. PLEN<3:0> ビット (CRCCON<3:0>) に 07h を書き込む
2. CRCXOR に値 (例 : 31h) を書き込む
3. CRCWDAT に値を書き込む :
   - 0000h (新たな計算を開始する場合)、または、
   - それまでに計算した中間結果 (メッセージ ストリームの一部の結果)
   - CRCIF (IFS4<3>) ビットをクリアし、CRCIE (IEC4<3>) ビットをセットして CRC 割り込みを有効化する
4. CRCFUL ビットがセットされておらず、かつメッセージ ストリームの全部のデータバイトをまだ FIFO に書き込んでいない場合、1 データバイトを CRCDAT レジスタに書き込む
5. CRCFUL ビットがセットされておらず、かつメッセージ ストリームの全部のデータバイトを既に FIFO に書き込んだ場合、1 バイト (値 00h) を CRCDAT レジスタ書き込み、CRC を使用するアプリケーションでソフトウェア フラグ (例 : FINAL_CALCULATION) をセットする
6. CRCFUL ビットまたはソフトウェア フラグ (FINAL_CALCULATION) がセットされている場合、CRCGO ビットをセットして CRC を開始する

7. FINAL_CALCULATION フラグがセットされている場合、CRC 割り込みで CRCWDAT レジスタを読み出し、CRCGO ビットをクリアする
8. 中間結果である場合 (CRC 計算を実行したが FINAL_CALCULATION フラグがセットされていない場合)、その中間結果を次の計算処理に引き渡す

36.7.3 5 ビットまたは 7 ビット多項式

5 ビットまたは 7 ビット多項式の場合、CRC モジュールはバイトの下位 5 ビットまたは下位 7 ビットに基づいてチェックサムを計算します。5 ビットデータの場合、バイトの下位 5 ビットに 5 ビットデータを格納し、上位 3 ビットに 0 を書き込む事ができます。7 ビットデータの場合、バイトの下位 7 ビットに 7 ビットデータを格納し、最上位ビットに 0 を書き込む事ができます。詳細は **36.5.2.1「FIFO から CRC カリキュレータへのインターフェイス」**を参照してください。

上記のようにメッセージ ストリームからバイトデータを形成した後に、**36.7.2「8 ビット多項式」**に記載した手順を適用する事ができます。多項式長 (PLEN<3:0>) には、5 ビット多項式の場合 04h、7 ビット多項式の場合 06h を設定します。CRCXOR レジスタで適当な 5 ビットまたは 7 ビット多項式を設定する事ができます。

36.7.4 16 ビット多項式

16 ビット多項式を使用する CRC 計算の推奨手順は以下の通りです。

1. PLEN<3:0> ビット (CRCCON<3:0>) に 0Fh を書き込む
2. CRCXOR に値 (例 : 8005h) を書き込む
3. CRCWDAT に値を書き込む :
   - 0000h (新たな計算を開始する場合)、または、
   - それまでに計算した中間結果 (メッセージ ストリームの一部の結果)
4. CRCFUL ビットがセットされておらず、かつメッセージ ストリームの全部のデータワードをまだ FIFO に書き込んでいない場合、1 データワードを CRCDAT レジスタに書き込む
5. CRCFUL ビットがセットされておらず、かつメッセージ ストリームの全部のデータワードを既に FIFO に書き込んだ場合、1 ワード (値 0000h) を CRCDAT レジスタに書き込み、CRC を使用するアプリケーションでソフトウェア フラグ(例: FINAL_CALCULATION) をセットする
6. CRCFUL ビットまたはソフトウェア フラグ (FINAL_CALCULATION) がセットされている場合、CRCGO ビットをセットして CRC を開始する
7. CRCMPT がセットされた時に CRCGO ビットをクリアし、CRCWDAT レジスタから結果バイトを読み出す
8. 中間結果である場合 (CRC 計算を実行したが FINAL_CALCULATION フラグがセットされていない場合)、その中間結果を次の計算処理に引き渡す

Note:	多項式の長さが 16 ビットである場合、CRC モジュールは FIFO に整数個の 16 ビットデータが格納されているものとみなします。

16 ビット多項式では単純にワード書き込みできます。しかし 16 ビット多項式にバイト書き込み動作を使用するアプリケーションも存在します (例 : UART 送受信)。そのようなアプリケーションでは、バイトが奇数個であればダミーバイトを追加する必要があります。一方、メッセージ ストリームが偶数個のバイトを格納している場合、ダミーバイトを追加すべきではありません。このような場合、上記の 16 ビット多項式用の手順を下記のように修正する必要があります。

1. PLEN<3:0> ビット (CRCCON<3:0>) に 0Fh を書き込む
2. CRCXOR に値 (例 : 8005h) を書き込む
3. CRCWDAT に値を書き込む :
   - 0000h (新たな計算を開始する場合)、または、
   - それまでに計算した中間結果 (メッセージ ストリームの一部の結果)
4. CRCFUL ビットがセットされておらず、かつメッセージ ストリームの全部のデータバイトをまだ FIFO に書き込んでいない場合、1 データバイトを CRCDAT レジスタに書き込み、カウンタをインクリメントして FIFO に書き込んだバイト数を正しく示す

5. CRCFUL ビットがセットされておらず、かつメッセージ ストリームの全部のデータバイトを既に FIFO に書き込んだ結果 FIFO 内のデータバイト数が奇数であった場合、ダミーバイト (値 00h) を CRCDAT レジスタに書き込み、ソフトウェア アプリケーションでソフトウェア フラグ (例 : FINAL_CALCULATION) をセットする
6. CRCFUL ビットがセットされておらず、かつメッセージ ストリームの全部のデータバイトを既に FIFO に書き込んだ結果 FIFO 内のデータバイト数が偶数であった場合、ソフトウェア フラグ (MESSAGE_OVER) をセットする
7. CRCFUL ビットがセットされておらず、かつ MESSAGE_OVER フラグがセットされている場合、1 ワード (値 0000h) を CRCDAT に書き込み、ソフトウェア フラグ (例 : FINAL_CALCULATION) をセットする。
8. CRCFUL ビットまたは FINAL_CALCULATION フラグがセットされている場合、CRCGO ビットをセットして CRC を開始する
9. CRCMPT がセットされた時に CRCGO ビットをクリアし、CRCWDAT レジスタから結果バイトを読み出す
10. 中間結果である場合 (CRC 計算を実行したが FINAL_CALCULATION フラグがセットされていない場合)、その中間結果を次の計算処理に引き渡す

36.7.4.1 サンプルコード

例 36-1 に、16 ビット多項式を使用して 8 ワードのデータブロックを処理するように CRC モジュールを設定するアプリケーション コードの例を示します。このアプリケーション コードは、CRCIF フラグをポーリングするか、あるいは CRC ISR を作成および使用して、CRC モジュールの処理完了を検出します。

このサンプルコードでは、FIFO よりも大きなサイズのデータブロックを処理します。FIFO がフルではない時にデータを FIFO に書き込みます。FIFO がフルになると CRCGO ビットを一度セットしてからクリアします。CRCGO ビットをクリアする事により、FIFO を再びフルにする事ができます。これにより、無用な割り込みの生成を回避できます。このプロセスは、CRC シフトレジスタが FIFO からのデータワードを処理中である時も実行されます。データブロックの最終ワードを FIFO に書き込んだ後に値 0x0000 を書き込む事によって、最終ワードを CRC シフトレジスタからシフトアウトさせます。フラグ (dataDone) は、全てのデータワードの処理が完了した事を示します。このフラグを ISR 内でチェックする事により、入力データシーケンス全体の最終的な CRC 結果を取得します。

例 36-1: サンプルコード

```
int src[20];
int i;

CRCCONbits.PLEN = 0x0F;               /* 16 bit CRC*/
CRCXOR= 0x1020;                              /* CRC-CCITT (XMODEM) */
IFS4bits.CRCIF = 0;                          /* Clear the CRC Interrupt Flag */
IEC4bits.CRCIE = 1;                          /* Enable the CRC Interrupt */

for(i = 0; i < 20; i ++)
{
 CRCDAT=Src[i];                      /* Load the CRC FIFO */
 if(CRCCONbits.CRCFUL == 1)
 {
   CRCCONbits.CRCGO = 1;             /* If FIFO is full then start the */
   CRCCONbits.CRCGO = 0;             /* CRC serial register */
 }
 while(CRCCONbits.CRCFUL == 1);      /* Wait till FIFO is not full before */
                                                        /*writing next data */
}
CRCCONbits.CRCGO = 1;                /*Enable the serial shift register */
CRCDAT = 0x0000;                     /* and write 0x0000 to shift out the */
dataDone = 1;                        /* last result.Set the dataDone flag */
while(1);
}

void __attribute__((__interrupt__,no_auto_psv)) _CRCInterrupt(void)
{

    IFS4bits.CRCIF = 0;
    if(dataDone == 1)
    {

            CRCCONbits.CRCGO = 0;     /* Stop the CRC serial register */
            CRCResult = CRCWDAT;      /* read the result */
    }
}
```

36.7.5 10 ビットまたは 12 ビット多項式

10 ビットまたは 12 ビット多項式の場合、CRC モジュールはワードの下位 10 ビットまたは下位 12 ビットを使用してチェックサムを計算します。10 ビットデータの場合、ワードの上位 6 ビットにゼロを書き込む事ができます。12 ビットデータの場合、ワードの上位 4 ビットにゼロを書き込む事ができます。詳細は **36.5.2.1「FIFO から CRC カリキュレータへのインターフェイス」**を参照してください。

上記のように 10 ビットまたは 12 ビットデータと「ドントケア」ビットでワードデータを形成した後に、**36.7.4「16 ビット多項式」**に記載した手順を適用できます。PLEN<3:0> ビットには、10 ビット多項式の場合 09h、12 ビット多項式の場合 0Bh を設定します。設定した長さに応じた適当な生成多項式を CRCXOR レジスタで設定する事ができます。

36.8 省電力モード時の動作

36.8.1 スリープモード時

CRC モジュールの動作中にデバイスがスリープモードに移行すると、モジュールはクロック動作が再開するまで現在の状態で一時停止します。

36.8.2 アイドルモード時

アイドルモード時も CRC モジュールを完全に動作させるには、アイドルモードに移行する前に CSIDL ビットをクリアする必要があります。

CSIDL = 1 の場合、アイドルモード時のモジュールの挙動はスリープモード時と同じです。モジュール クロックが利用できなくても、保留中の割り込みイベントは継承されます。

36.9 レジスタマップ

dsPIC33F プログラマブル巡回冗長検査 (CRC) モジュールに関連する特殊機能レジスタ (SFR) の要約を表 36-2 に示します。

表 36-2: プログラマブル CRC モジュール関連の特殊機能レジスタ (1)

SFR 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
CRCCON	—	—	CSIDL	VWORD<4:0>					CRCFUL	CRCMPT	—	CRCGO	PLEN<3:0>				0040
CRCXOR	X<15:1>															—	0000
CRCDAT	CRC データ入力レジスタ																0000
CRCWDAT	CRC シフト書き込みレジスタ																0000

凡例 : — = 未実装、「0」として読み出し . 網掛けしたビットはプログラマブル CRC モジュールの動作には使用しません。

Note 1: 各メモリマップの詳細はデバイスのデータシートを参照してください。

36.10 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 製品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。プログラマブル巡回冗長検査 (CRC) 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。	N/A

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。

36.11 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 (2009 年 9 月)

このリビジョンでの変更内容は以下の通りです。

- 例 :
  - サンプルコード (例 36-1) を更新
- レジスタ :
  - **CRCCON:CRC 制御レジスタ** (レジスタ 36-1 参照) の bit 4 の説明を更新
- セクション :
  - **36.5.2.3「CRC 結果」**を更新
  - **36.5.2.1「FIFO から CRC カリキュレータへのインターフェイス」**を更新
  - **36.5.3「割り込み動作」**を更新
  - **36.7.2「8 ビット多項式」**内の手順を更新
  - **36.7.4.1「サンプルコード」**を更新
- 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: 978-1-61341-041-7

NOTE: