



---

**dsPIC30F ファミリー**  
リファレンスマニュアル  
高性能デジタルシグナルコントローラー

Microchip デバイスのコード保護機能に関する以下の点に留意ください。

- Microchip の製品は各製品独自の Microchip データーシートにある仕様を満たしています。
- 各製品ファミリーは、通常の状態且つ所定の方法で利用いただければ市場にある類似製品の中で最も安全なファミリーの一つと Microchip は信じております。
- 不正かつ非合法な方法を使ったコード保護機能の侵害があります。弊社の理解ではこうした手法は、Microchip データーシートにある動作仕様書以外の方法で Microchip 製品を使用することになります。こうした手法を使用した人は、ほとんどの場合、知的財産権の侵害となります。
- Microchip はコードの統合性に関心をお持ちの顧客とは協働させていただきます。
- Microchip または他のセミコンダクターメーカーがコードの安全性を保証したものではありません。コード保護は製品保護が「破られない」ということを保証するものではありません。

コード保護は常に進化します。Microchip は、当社製品のコード保護機能を継続的に改善することをお約束いたします。

この文献は読者が内容をお読みになる時の参考のために日本語に翻訳されています。翻訳に誤りが存在する場合、**Microchip Technology Inc.** 及び全ての子会社、関連会社、役員、従業員、代理業者は一切の責任を負いかねます。オリジナル文献を必ずご参照することを強くお勧めします。

本書に書かれているデバイスアプリケーション等に関する内容は、参考情報に過ぎません。ご利用のアプリケーションが仕様を満たしているかどうかについては、お客様の責任において確認をお願いします。これらの情報の正確さ、またはこれの情報の使用に関し、Microchip はいかなる表明と保証を行ふものではなく、また、一切の責任を負うものではありません。Microchip の明示的な書面による承認なしに、生命維持装置に Microchip の製品を使用することは認められていません。知的財産権に基づく、ライセンスを暗示的に与えたものではありません。

## 商標

Microchip の名称とロゴ、Microchip のロゴ、Accuron、dsPIC、KEELOQ、microID、MPLAB、PIC、PICmicro、PICSTART、PRO MATE、PowerSmart、rfPIC、SmartShunt は米国及び他の国々において、Microchip Technology Inc. の登録商標です。

AmpLab, FilterLab, Migratable Memory, MXDEV, MXLAB, PICMASTER, SEEVAL, SmartSensor and The Embedded Control Solutions Company は、米国において Microchip Technology Inc. の登録商標です。

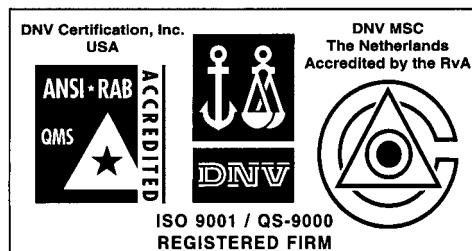
Analog-for-the-Digital Age, Application Maestro, dsPICDEM、dsPICDEM.net、dsPICworks、ECAN、ECONOMONITOR、FanSense、FlexROM、fuzzyLAB、In-Circuit Serial Programming、ICSP、ICEPIC、Linear Active Thermistor、MPASM、MLIB、MPLINK、MPSIM、PICkit、PICDEM、PICDEM.net、PICLAB、PICtail、PowerCal、PowerInfo、PowerMate、PowerTool、Real ICE、rfLAB、rfPICDEM、Select Mode、Smart Serial、SmartTel、Total Endurance、UNI/O、WiperLock 及び Zena は、米国及び他の国々において、Microchip Technology Inc. の登録商標とです。

SQTP は米国において Microchip Technology Inc. のサービスマークです。

本書に記載された上記以外の商標は、それぞれ所有会社の登録商標です。

著作権。© 2006 Microchip Technology Inc.、無断複写・転載を禁じます。

 再生紙を使用。



Microchip は、QS-9000 を受けました。本社、アリゾナ州チャンドラーとテンペとカリフォルニア州マウンテンビューにあるデザイン及びウェハ施設に対する 2002 年品質システム認証です。弊社の品質システムプロセスと手続きは、QS-9000 PICmicro™ 8-bit MCUs、KEELOQ コードホッピングデバイス、シリアル EEPROMs、マイクロペリフェラル、非揮発性メモリーとアナログ製品を対象としています。更に、開発システムの設計及び製造に関する Microchip の品質システムは、ISO9001 の認証を受けています。

# 目次

---



---

	<u>PAGE</u>
<b>第 1 章 . はじめに</b>	<b>1-1</b>
はじめに .....	1-2
マニュアルの目的 .....	1-2
デバイスの構造 .....	1-3
開発サポート .....	1-4
スタイルおよび記号規定 .....	1-4
関連文書 .....	1-6
改訂履歴 .....	1-7
<b>第 2 章 . CPU</b>	<b>2-1</b>
はじめに .....	2-2
プログラマ用モデル .....	2-4
ソフトウェアスタックポインタ .....	2-8
CPU レジスタの説明 .....	2-11
論理演算ユニット (ALU) .....	2-17
DSP エンジン .....	2-18
除算サポート .....	2-27
命令フローの種類 .....	2-27
ループ構造 .....	2-30
アドレスレジスタの依存関係 .....	2-35
レジスタマップ .....	2-38
関連するアプリケーションノート .....	2-40
改訂履歴 .....	2-41
<b>第 3 章 . データメモリ</b>	<b>3-1</b>
はじめに .....	3-2
データ空間アドレス生成ユニット (AGUs) .....	3-5
モジュロアドレッシング .....	3-7
ビット反転アドレッシング .....	3-14
コントロールレジスタ記述 .....	3-18
関連するアプリケーションノート .....	3-23
改訂履歴 .....	3-24
<b>第 4 章 . プログラムメモリ</b>	<b>4-1</b>
プログラムメモリアドレスマップ .....	4-2
プログラムカウンタ (PC) .....	4-4
プログラムメモリからのデータアクセス .....	4-4
データ空間からのプログラム空間の可視化 .....	4-8
プログラムメモリへの書き込み .....	4-10
関連するアプリケーションノート .....	4-11
改訂履歴 .....	4-12
<b>第 5 章 . FLASH と EEPROM のプログラミング</b>	<b>5-1</b>
はじめに .....	5-2
テーブル命令動作 .....	5-2
コントロールレジスタ .....	5-5
実行時セルフプログラミング (RTSP) .....	5-9
データ EEPROM プログラミング .....	5-14
設計の秘訣 .....	5-20
関連するアプリケーションノート .....	5-21
改訂履歴 .....	5-22

---



---

# 目次

	<u>PAGE</u>
<b>第 6 章 . 割り込み</b>	<b>6-1</b>
序章 .....	6-2
マスクできないトラップ .....	6-6
割り込み処理タイミング .....	6-11
割り込み用制御、ステータスレジスタ .....	6-14
割り込み設定手順 .....	6-42
設計の秘訣 .....	6-44
関連するアプリケーションノート .....	6-45
改訂履歴 .....	6-46
<b>第 7 章 . 発振器</b>	<b>7-1</b>
はじめに .....	7-2
CPU クロックの仕組み .....	7-4
発振器のコンフィギュレーション .....	7-5
発振器コントロールレジスタ (OSCCON) .....	7-6
主発振器 .....	7-8
水晶発振子 / セラミック発振子 .....	7-9
水晶、クロックモード C1、C2 および Rs の最適値を決定する .....	7-11
外部クロック入力 .....	7-12
外部 RC 発振器 .....	7-13
フェーズロックループ (PLL) .....	7-17
低電力 32 kHz 水晶発振器 .....	7-18
発振器スタートアップタイム (OST) .....	7-18
内蔵高速 RC 発振器 (FRC) .....	7-18
内蔵低電力 RC 発振器 (LPRC) .....	7-19
フェールセーフクロックモニター (FSCM) .....	7-19
プログラマブル発振器ポストスケーラ .....	7-20
クロック切替え動作 .....	7-21
設計の秘訣 .....	7-25
関連するアプリケーションノート .....	7-26
改訂履歴 .....	7-27
<b>第 8 章 . リセット</b>	<b>8-1</b>
はじめに .....	8-2
リセット時のクロックソースの選択 .....	8-5
POR : パワーオンリセット .....	8-5
外部リセット (EXTR) .....	8-7
ソフトウェアリセット命令 (SWR) .....	8-7
ウォッチドッグタイムアウト命令 (WDTR) .....	8-7
ブラウンアウトリセット (BOR) .....	8-8
RCON ステータスピットの使用 .....	8-10
デバイスリセット時間 .....	8-11
デバイススタートアップタイムチャート .....	8-13
特殊機能レジスラリセットステート .....	8-16
設計の秘訣 .....	8-17
関連するアプリケーションノート .....	8-18
改訂履歴 .....	8-19

## 目次

---



---

	<u>PAGE</u>
<b>第 9 章 . 低電圧検出 (LVD)</b>	<b>9-1</b>
はじめに .....	9-2
LVD 動作 .....	9-5
設計の秘訣 .....	9-6
関連するアプリケーションノート .....	9-7
改訂履歴 .....	9-8
<b>第 10 章 . ウオッチドッグタイマーおよび省電力モード</b>	<b>10-1</b>
序章 .....	10-2
省電力モード .....	10-2
SLEEP モード .....	10-2
IDLE モード .....	10-4
省電力命令との同時割り込み .....	10-5
ウォッチドッグタイマ .....	10-5
設計の秘訣 .....	10-8
関連するアプリケーションノート .....	10-9
改訂履歴 .....	10-10
<b>第 11 章 . I/O ポート</b>	<b>11-1</b>
序章 .....	11-2
入出力ポート制御レジスタ .....	11-3
周辺装置マルチプレクサ .....	11-4
ポートの詳細説明 .....	11-5
状態変化通知 (CN) ピン .....	11-5
SLEEP および IDLE モードでの CN 動作 .....	11-6
関連するアプリケーションノート .....	11-9
改訂履歴 .....	11-10
<b>第 12 章 . タイマー</b>	<b>12-1</b>
序章 .....	12-2
タイマーのバリエーション .....	12-3
制御レジスタ .....	12-6
動作モード .....	12-9
タイマープリスケーラ .....	12-14
タイマー割り込み .....	12-14
16 ビットタイマーモジュールレジスタの読み込みおよび書き込み .....	12-15
低電力 32 kHz クリスタル発振器入力 .....	12-15
32 ビットタイマー構成 .....	12-16
32 ビットタイマーモード動作 .....	12-18
32 ビットタイマーへの読み込みおよび書き込み .....	12-21
省電力状態でのタイマー動作 .....	12-21
タイマーモジュール使用周辺装置 .....	12-22
設計の秘訣 .....	12-24
関連するアプリケーションノート .....	12-25
改訂履歴 .....	12-26

# 目次

---



---

	<u>PAGE</u>
<b>第 13 章. 入力キャプチャ</b>	<b>13-1</b>
序章 .....	13-2
入力キャプチャレジスタ .....	13-3
タイマー選択 .....	13-4
入力キャプチャイベントモード .....	13-4
キャプチャバッファオペレーション .....	13-8
入力キャプチャ割り込み .....	13-9
UART オートポートサポート .....	13-9
省電力状態における入力キャプチャのオペレーション .....	13-10
入出力ピン制御 .....	13-10
入力キャプチャモジュールと関連する特別関数レジスタ .....	13-11
製品情報 .....	13-12
関連するアプリケーションノート .....	13-13
改訂履歴 .....	13-14
<b>第 14 章. 出力比較モジュール</b>	<b>14-1</b>
序章 .....	14-2
出力比較レジスタ .....	14-3
動作モード .....	14-4
省電力状態での出力比較動作 .....	14-23
入出力ピン制御 .....	14-23
設計の秘訣 .....	14-26
関連するアプリケーションノート .....	14-27
改訂履歴 .....	14-28
<b>第 15 章. モーター制御 PWM</b>	<b>15-1</b>
序章 .....	15-2
制御レジスタ .....	15-4
PWM タイムベース .....	15-16
PWM デューティ比較ユニット .....	15-20
相補 PWM 出力モード .....	15-24
デッドタイム制御 .....	15-25
独立 PWM 出力モード .....	15-28
PWM 出力オーバーライド .....	15-29
PWM 出力極性制御 .....	15-32
PWM フォールトピン .....	15-32
PWM アップデートロックアウト .....	15-35
PWM 特殊イベントトリガー .....	15-35
デバイス省電力モードにおける動作 .....	15-36
デバイスエミュレーション用特別機能 .....	15-37
関連するアプリケーションノート .....	15-40
改訂履歴 41 .....	15-40

## 目次

---

	<u>PAGE</u>
<b>第 16 章. 直交エンコーダインターフェース (QEI)</b>	<b>16-1</b>
序章 .....	16-2
制御およびステータスレジスタ .....	16-4
プログラム可能デジタルノイズフィルタ .....	16-8
直交デコーダ .....	16-9
16 ビットアップ/ダウンポジションカウンタ .....	16-11
16 ビットのタイマー/カウンタの代替として QEI を使用 .....	16-16
直交エンコーダインターフェース割り込み .....	16-17
入出力ピン制御 .....	16-18
省電力モード時の QEI の動作 .....	16-19
リセットの効果 .....	16-19
設計の秘訣 .....	16-21
関連するアプリケーションノート .....	16-22
改訂履歴 .....	16-23
<b>第 17 章. 10 ビット A/D コンバータ</b>	<b>17-1</b>
序章 .....	17-2
制御レジスタ .....	17-4
A/D 結果バッファ .....	17-4
A/D 用語と変換シーケンス .....	17-11
A/D モジュール構成 .....	17-13
電圧リファレンスソースの選択 .....	17-13
A/D 変換クロックの選択 .....	17-14
サンプリング用アナログ入力の選択 .....	17-15
モジュールの有効化 .....	17-17
サンプル/変換シーケンスの指定 .....	17-17
サンプリングの開始方法 .....	17-18
サンプリング停止と変換開始の方法 .....	17-19
サンプリング/変換動作の制御 .....	17-30
変換結果をバッファに書き込む方法の指定 .....	17-31
変換シーケンスの例 .....	17-32
A/D サンプリング要件 .....	17-46
A/D 結果バッファの読み込み .....	17-49
変換関数 .....	17-50
A/D 精度 / 誤差 .....	17-50
接続の条件 .....	17-50
初期化 .....	17-51
SLEEP モードおよび IDLE モード時の動作 .....	17-52
リセットの影響 .....	17-52
10 ビット A/D コンバータに関連する特殊関数レジスタ .....	17-53
設計の秘訣 .....	17-54
関連するアプリケーションノート .....	17-55
改訂履歴 .....	17-56

## 目次

---

	<u>PAGE</u>
<b>第 18 章 . 12- ビット A/D コンバータ</b>	<b>18-1</b>
序章 .....	18-2
制御レジスタ .....	18-4
A/D 結果バッファ .....	18-4
A/D 専門用語、変換シーケンス .....	18-10
A/D モジュール構成 .....	18-11
電圧リファレンスソース選択 .....	18-11
A/D 変換クロックの選択 .....	18-12
サンプリングのアナログ入力選択 .....	18-13
モジュールの有効化 .....	18-15
サンプリングの開始方法 .....	18-15
サンプリングの停止および変換の開始方法 .....	18-15
サンプル・変換動作の制御 .....	18-20
変換結果をバッファに書き込む方法の指定 .....	18-21
変換シーケンス例 .....	18-22
A/D サンプリング要件 .....	18-27
A/D 結果バッファの読み込み .....	18-30
変換関数 .....	18-31
A/D の精度と誤差 .....	18-31
接続条件 .....	18-31
初期化 .....	18-32
SLEEP モードおよび IDLE モード時の動作 .....	18-33
RESET の影響 .....	18-33
12 ビット A/D コンバータに付随する特別機能レジスタ .....	18-34
設計の秘訣 .....	18-35
関連するアプリケーションノート .....	18-36
改訂履歴 .....	18-37
<b>第 19 章 . UART</b>	<b>19-1</b>
序章 .....	19-2
制御レジスタ .....	19-3
UART ポーレートレジスタ (BRG) .....	19-8
UART 構成 .....	19-10
UART トランスマッタ .....	19-11
UART レシーバ .....	19-14
9 ビット通信における UART の使用 .....	19-18
ブレーク文字の受信 .....	19-19
初期設定 .....	19-20
UART のその他の機能 .....	19-21
CPU SLEEP および IDLE モードの際の UART 動作 .....	19-21
UART モジュール関連レジスタ .....	19-22
設計の秘訣 .....	19-23
関連するアプリケーションノート .....	19-24
改訂履歴 .....	19-25

## 目次

---



---

	<u>PAGE</u>
<b>第 20 章 . シリアル周辺装置インターフェース (SPI)</b>	<b>20-1</b>
序章 .....	20-2
ステータスおよびコントロールレジスタ .....	20-4
動作モード .....	20-7
SPI マスター モードクロック周波数 .....	20-19
省電力モードでの動作 .....	20-20
SPI モジュールに関連の特殊機能レジスタ .....	20-22
関連するアプリケーションノート .....	20-23
改訂履歴 .....	20-24
<b>第 21 章 . I<sup>2</sup>C™</b>	<b>21-1</b>
概要 .....	21-2
I <sup>2</sup> C バスの特長 .....	21-4
ステータスおよびコントロールレジスタ .....	21-7
I <sup>2</sup> C 動作の有効化 .....	21-13
シングルマスター環境でマスターとして通信 .....	21-15
マルチマスター環境でマスターとして通信 .....	21-29
スレーブとして通信 .....	21-32
I <sup>2</sup> C バス接続についての検討事項 .....	21-47
PWRSAV 命令実行中のモジュール動作 .....	21-49
RESET の影響 .....	21-49
設計の秘訣 .....	21-50
関連するアプリケーションノート .....	21-51
改訂履歴 .....	21-52
<b>第 22 章 . データ変換器インターフェース (DCI)</b>	<b>22-1</b>
序章 .....	22-2
制御レジスタの説明 .....	22-2
コーデックインターフェースの基礎と技術 .....	22-8
DCI 動作 .....	22-10
DCI モジュールを使用する .....	22-17
省電力モードでの動作 .....	22-28
DCI に関連するレジスタ .....	22-28
設計の秘訣 .....	22-30
関連するアプリケーションノート .....	22-31
改訂履歴 .....	22-32

# 目次

---



---

	<u>PAGE</u>
<b>第 23 章 . CAN モジュール</b>	<b>23-1</b>
序章 .....	23-2
CAN モジュールの制御レジスタ .....	23-2
CAN モジュールの特徴 .....	23-28
CAN モジュールの実装 .....	23-29
CAN モジュールの動作モード .....	23-38
メッセージ受信 .....	23-41
送信 .....	23-51
エラー検出 .....	23-60
CAN ポーレート .....	23-62
割り込み .....	23-66
タイムスタンプ .....	23-67
CAN モジュール I/O .....	23-67
CPU パワー節約モードでの動作 .....	23-68
CAN プロトコルの概要 .....	23-70
関連するアプリケーションノート .....	23-74
改訂履歴 .....	23-75
<b>第 24 章 . デバイスコンフィギュレーション</b>	<b>24-1</b>
序章 .....	24-2
デバイスコンフィギュレーションレジスタ .....	24-2
コンフィギュレーションビットの説明 .....	24-7
デバイス識別レジスタ .....	24-8
関連するアプリケーションノート .....	24-9
改訂履歴 .....	24-10
<b>第 25 章 . 開発ツールのサポート</b>	<b>25-1</b>
序章 .....	25-2
Microchip ハードウェアおよび言語ツール .....	25-2
カードパーティハードウェア/ソフトウェアツールおよびアプリケーションライブラリ .....	25-6
dsPIC30F ハードウェア開発ボード .....	25-11
関連するアプリケーションノート .....	25-15
改訂履歴 .....	25-16
<b>第 26 章 . 付録</b>	<b>26-1</b>
I <sup>2</sup> C <sup>TM</sup> 概要 .....	26-2
CAN 概要 .....	26-12
コーデックプロトコル概要 .....	26-26



## 第1章. はじめに

### ハイライト

この章は、以下の項目を含んでいます。

1.1	はじめに.....	1-2
1.2	マニュアルの目的.....	1-2
1.3	デバイスの構造.....	1-3
1.4	開発サポート.....	1-4
1.5	スタイルおよび記号規定.....	1-4
1.6	関連文書.....	1-6
1.7	改訂履歴.....	1-7

## 1.1 はじめに

Microchip は、マイクロコントローラおよびアナログ半導体の主要なプロバイダです。同社は組み込みコントロール市場のニーズを満たす製品を主に提供しています。以下の製品の主要なプロバイダです。

- 8- ビット汎用マイクロコントローラ (PICmicro® MCUs)
- dsPIC30F 16- ビットマイクロコントローラ
- 特殊及び標準の不揮発性メモリデバイス
- セキュリティデバイス (KEELOQ®)
- アプリケーション特定標準製品

Microchip 社の提供する興味深い製品すべてを一覧できる Microchip プロダクトラインカードをぜひ入手してください。本資料は、現地オフィスまたは Microchip ウェブサイト ([www.microchip.com](http://www.microchip.com)) からダウンロードして入手いただけます。

## 1.2 マニュアルの目的

命令ワードおよびデータバスのサイズによって PICmicro および dsPIC30F デバイスはグループ分けされます。

現在、デバイスのファミリーは以下の通りです。

1. ベースライン :12- ビット 命令ワード長、8 - ビット データバス
2. ミッドレンジ :14- ビット 命令ワード長、8 - ビット データバス
3. ハイエンド : 16- ビット 命令ワード長、8 - ビット データバス
4. 拡張 : 16- ビット 命令ワード長、8 - ビット データバス
5. dsPIC30F: 24- ビット 命令ワード長、16- ビット データバス

本書は、dsPIC30F 16- ビット MCU ファミリーデバイスについて説明します。

dsPIC30F MCU のファミリーのアーキテクチャおよび周辺モジュールの動作について説明しますが、各デバイスの詳細は説明しません。ユーザーは各デバイス専用のデータシートを参照する必要があります。データシートの情報は以下を含んでいます。

- デバイスマモリマップ
- デバイスピンアウトとパッケージの詳細
- デバイスの電気的仕様書
- デバイスに含まれる周辺モジュールの一覧

本書では、全体に渡ってコード例が記載されます。これらの例は、ほとんどのデバイスに有効ですが、ファミリー全般で使用できるもの以外にも、あるデバイス専用として書き込む必要があるものもあります。レジスタファイルのマッピングの際には、コードは、デバイスの種類により若干修正する必要があります。

### 1.3 デバイスの構造

dsPIC30F デバイスの各部分は以下の 3 つのグループのうちの 1 つに分類されます。

1. CPU コア
2. システム統合
3. 周辺装置

#### 1.3.1 CPU コア

CPU コアには、デバイスの動作に必要な基本的な機能が含まれています。CPU コアに関連するマニュアルは以下の内容を含みます。

1. CPU
2. データメモリ
3. プログラムメモリ
4. DSP エンジン
5. 割り込み

#### 1.3.2 システム統合

システム統合機能で以下が可能になります。

- システム費用の節約
- システム信頼性の向上
- 設計柔軟性の向上

本書の以下の章では、dsPIC30F システム統合機能について説明します。

1. 発振器
2. リセット
3. 低電圧の検出
4. ウオッチドッグタイマーおよび省電力モード
5. FLASH 及び EEPROM プログラミング
6. デバイスコンフィギュレーション

#### 1.3.3 周辺装置

dsPIC30F には、デバイスを外部にインターフェースできる多くの周辺装置があります。本書では、以下のような周辺装置について説明します。

1. I/O ポート
2. タイマー
3. インプットキャプチャモジュール
4. アウトプット比較モジュール
5. 直交エンコーダインターフェース (QEI)
6. 10 - ビット A/D コンバータ
7. 12 - ビット A/D コンバータ
8. UART モジュール
9. SPI モジュール
10. I<sup>2</sup>C<sup>TM</sup> モジュール
11. データコンバーターインターフェース (DCI) モジュール
12. CAN モジュール

#### 1.3.4 メモリ技術

書き込みの際、すべての dsPIC30F デバイスは FLASH プログラムメモリ技術を使用します。FLASH プログラムメモリは電気的に消去またはプログラム可能です。

## 1.4 開発サポート

Microchip は、ユーザーがアプリケーションコードを効果的に開発およびデバッグできる様々な開発ツールを提供します。 Microchip の開発ツールは 4 つのカテゴリに分類されます：

1. コード生成
2. ハードウェア／ソフトウェアデバッグ
3. デバイスプログラマ
4. 製品評価ボード

第 25 章、「開発ツールのサポート」では、各 Microchip の開発ツールの全体説明があります。新しいツールの開発にともない、最新製品の簡単な説明及びユーザーガイドを Microchip ウェブサイト ([www.microchip.com](http://www.microchip.com)) 又は Microchip の現地オフィスから入手できます。

Microchip は、開発サイクルを短縮するための下記のような参照ツールも提供します。

- アプリケーションノート
- 参照設計
- Microchip ウェブサイト
- フィールドアプリケーションサポートおよび現地オフィス
- 企業サポートライン

Microchip のウェブサイトは役に立つ参考情報を含む他のサイトのリストも提供しています。

## 1.5 スタイルおよび記号規定

本文書の全体にわたって、所定のスタイル及びフォントフォーマットの規定が使用されています。ほとんどのフォーマット規定は、強調テキストを意味します。 MCU 業界には多くの記号及び非規定ワード定義／省略があります。表 1-1 は、本文書に適用される多くの規定の説明を示します。巻末にある用語集は、文書全体にわたって使用される追加の用語とその略語の定義を提供しています。

## 1.5.1 文書規定

表 1-1 は、このマニュアルで使用されるいくつかの記号と条件を定義します。

表 1-1: 文書規定

記号又は用語	説明
セット	ビット / レジスタを論理 ‘1’ の値にする。
クリア	ビット / レジスタを論理 ‘0’ の値にする。
リセット	1) レジスタ / ビットをデフォルト状態にする。 2) デバイスのリセットが発生した後にデバイスが自らの初期値を決定するために使用する条件。いくつかのビットは ‘0’ になり、(割込み可能ビット等)、他は ‘1’ となる (I/O データ方向ビット等)。
0xnn or nnh	数字 ‘nn’ を 16 進法とする。これらの規定はコード例に使用される。例：0x13F 又は 13Fh。
B'bbbbbbbb'	数字 ‘bbbbbbbb’ を 2 進法とする。本規定はテキストおよび図と表に使用される。例：B'10100000'。
R-M-W	<b>Read-Modify-Write</b> 。レジスタ又はポートを読み込んで値を変更し、その値をレジスタまたはポートへ書き込む動作を指す。この動作は、1つの命令 (ビットセット、BSET 等) 又は命令のシーケンスから発生する。
: (colon)	レジスタ / ビット / ピンの範囲又は連結の指定に使用される。1つの例は、32-ビットタイマー値になる 2 個の 16-ビットレジスタの連結である TMR3:TMR2 です。 連結順番 (左一右) は、一般的な位置関係 (MSb to LSb、上位から下位へ) を指定する。
< >	特定レジスタのビット位置を指定する。 1つの例は、レジスタ及び関連付けられたビット又はビット位置を指定する PTCON<PTMOD1:PTMOD0> (or PTMOD<1:0>) である。
MSb, MSbit, LSb, LSbit	フィールドの最下位ビット又は最上位のビットを示す。
MSByte, MSWord, LSByte, LSWord	ビットフィールド中の最下位または最上位のバイトまたはワードを示す。
Courier Font	コード例、2 進数及びテキスト中の命令を表す。
Times Font	式及び変数に使用される。
<b>Times, Bold Font, Italic</b>	グラフィック / 式 / 例から呼び出された項目の説明テキストに使用される。
注	注は、発生しがちな問題を回避するため、または、デバイスファミリーメンバー間の特定の動作の差異に対し注意を喚起するために強調したい情報を表す。表 (この表のように) の下部にある場合、常に網掛けのボックス (以下のように) 内に表示される。
<b>注：</b> これは網掛けのボックスにある注です。	

## 1.5.2 電気的仕様書

本マニュアルには、電気的特性及びそれらのパラメータ番号への参照が含まれます。表 1-2 は、dsPIC30F デバイスのためのパラメータ番号付けを示します。パラメータ番号は各データシートと一致する特性及び条件のセットを示しますが、実パラメータはデバイスにより異なる可能性があります。本マニュアルは、デバイスのファミリーを説明し、パラメータ値を指定しません。デバイスの実パラメータ値については、ユーザーがデバイスデータシートの章「電気的特性」を参照する必要があります。

表 1-2: 電気的仕様書のパラメータ番号規定

パラメータ番号フォーマット	コメント
DXXX	DC 特性
AXXX	アナログ周辺モジュール DC 特性
XXX	タイミング (AC) 特性
PDXXX	デバイスプログラミング DC 特性
PXXX	デバイスプログラミング (AC) 特性

凡例: XXX は番号を示します。

## 1.6 関連文書

Microchip および他の情報源から、dsPIC30F MCUs を使用した開発に役立つ追加の文書を提供します。この一覧に含まれる文書のほとんどは最もよく使用される文書ですが、それ以外の文書もあります。最新の技術文書は、Microchip ウェブサイト ([www.microchip.com](http://www.microchip.com)) でチェックしてください。

### 1.6.1 Microchip 文書

dsPIC30F MCU を使用した開発の際に参考すべき以下の dsPIC30F 文書が Microchip から入手できます。多くの文書は、dsPIC30F MCU を使ったプログラミングおよび設計の実例を挙げることでアプリケーション開発に有益な情報を提供します。

#### 1. dsPIC30F プログラマリーファレンスマニュアル (DS70030)

dsPIC30F プログラマリーファレンスマニュアルは、dsPIC30F プログラマモデルおよび命令セットについて情報を提供します。本書では各命令および構文例の説明を提供します。

#### 2. dsPIC30F 高性能 16 - ビット ディジタル信号コントローラファミリ概要 (DS70043)

本書はデバイスピニ配置、メモリサイズおよび利用可能な周辺モジュールを含む dsPIC30F ファミリーのバリエーションの概要を提供します。

#### 3. dsPIC30F データシート (DS70082 と DS70083)

データシートにはピン配置およびパッケージの詳細、電気的仕様書およびメモリマップ等のデバイスの特定の情報を含んでいます。

### 1.6.2 第三者の文書

世界中のサードパーティから利用可能な文書が提供されています。Microchip はこれらの文書の技術的な精度は検証していませんが、Microchip dsPIC30F デバイスの運用を理解するために役に立つ可能性はあります。dsPIC30F と関連するサードパーティの文書は、Microchip ウェブサイトを参照してください。

## 1.7 改訂履歴

### A版

これは本ドキュメントの初版です。

### B版

マニュアルの本章に対する技術内容や編集改訂版はありませんが、マニュアル全体を通してB版を反映するために、この章は更新されています。

注意：



## 第2章 . CPU

### ハイライト

この章は、以下の項目を含んでいます。

2.1	はじめに.....	2-2
2.2	プログラマ用モデル.....	2-4
2.3	ソフトウェアスタックポインタ.....	2-8
2.4	CPU レジスタの説明 .....	2-11
2.5	論理演算ユニット (ALU) .....	2-17
2.6	DSP エンジン .....	2-18
2.7	除算サポート.....	2-27
2.8	命令フローの種類.....	2-27
2.9	ループ構造.....	2-30
2.10	アドレスレジスタの依存関係.....	2-35
2.11	レジスタマップ.....	2-38
2.12	関連するアプリケーションノート.....	2-40
2.13	改訂履歴.....	2-41

## 2.1 はじめに

dsPIC30F CPU モジュールは、16 ビット長のデータバスに修正されたハーバードアーキテクチャと、機能強化された命令セットを有していて、DSP 機能を強力にサポートします。CPU は 24 ビット命令語と可変長の **opcode** フィールドを有しています。プログラムカウンタの長さは 24 ビットで、これによりユーザは 4M ワード × 24 ビット長のプログラムメモリ空間を利用できます。1 命令サイクルの先読み機構によりスループットを維持するとともに、予測実行を提供します。プログラムフローを変更する命令、ダブルワード移動命令 (MOVE.D)、テーブル処理命令以外のすべての命令が 1 サイクルで実行されます。DO と REPEAT 命令によりオーバーヘッド無しのループプログラムが構成できます。しかも任意の時点でループを終了させることができます。

プログラマ用モデルでは、dsPIC30F デバイスには 16 個の 16 ビット幅の作業用レジスタがあります。各作業用レジスタはデータ、アドレスおよびアドレスオフセットレジスタとして使用されます。16 番目の作業用レジスタ (W15) は割り込みとコールのためのソフトウェアスタックポインタとして使用されます。

dsPIC30F 命令セットには、MCU クラス命令と DSP クラス命令の 2 つのクラスがあります。両方の命令クラスがシームレスに 1 つのアーキテクチャに統合されており、共通のユニットで実行されます。命令セットにいくつかのアドレッシングモードが含まれており、C コンパイラの効率が最高になるように設計されました。

データ空間は 32K ワードまたは 64K バイトとしてアクセスすることができます、X と Y データメモリの 2 つのブロックに分割されています。各メモリブロックには独立のアドレス発生ユニット (AGU) があります。MCU クラスの命令は X メモリ AGU で動作し、全データメモリ空間を 1 つのリニアなデータ空間としてアクセスします。一部の DSP 命令は、二重オペランド読み込みを行うため、X と Y の両方の AGU を使います。これによりデータアドレス空間は 2 つの空間に分けられます。X と Y のデータ空間の境界はデバイス毎に異なっています。

データメモリの上位 32k バイトは、16k ワード境界単位で、任意の位置のプログラムメモリ空間としてマッピングすることができるようになっています。そのマッピング位置の割り当ては、8 ビットの PSVPAG レジスタ (Program Space Visibility Page) で行います。データ空間としてマッピングしたプログラムでは、その空間のプログラムメモリは、データメモリとしてアクセスされます。さらに、RAM がプログラムメモリバスに接続されるため、プログラムメモリ空間が拡張された内部 RAM のように扱われます。

オーバーヘッドなしの循環バッファ (モジュロアドレッシング) が X と Y アドレス空間でサポートされます。モジュロアドレッシングにより、DSP アルゴリズムを実行するとき必要となるバッファ境界チェックのオーバーヘッドを無くすことができます。さらに、X AGU 循環アドレッシングはどの MCU クラス命令でも使うことができます。基数 2 FFT アルゴリズム用の入出力データ並べ替えを容易にするために X AGU はビットリバースアドレッシングもサポートしています。

CPU は命令内 (オペランドなし)、リラティブ、リテラル、メモリダイレクト、レジスタダイレクトおよびレジスタインダイレクトのアドレッシングモードをサポートします。機能の要件により各命令が所定のアドレッシングモードグループに分類されます。各命令は 6 つのアドレッシングモードをサポートします。

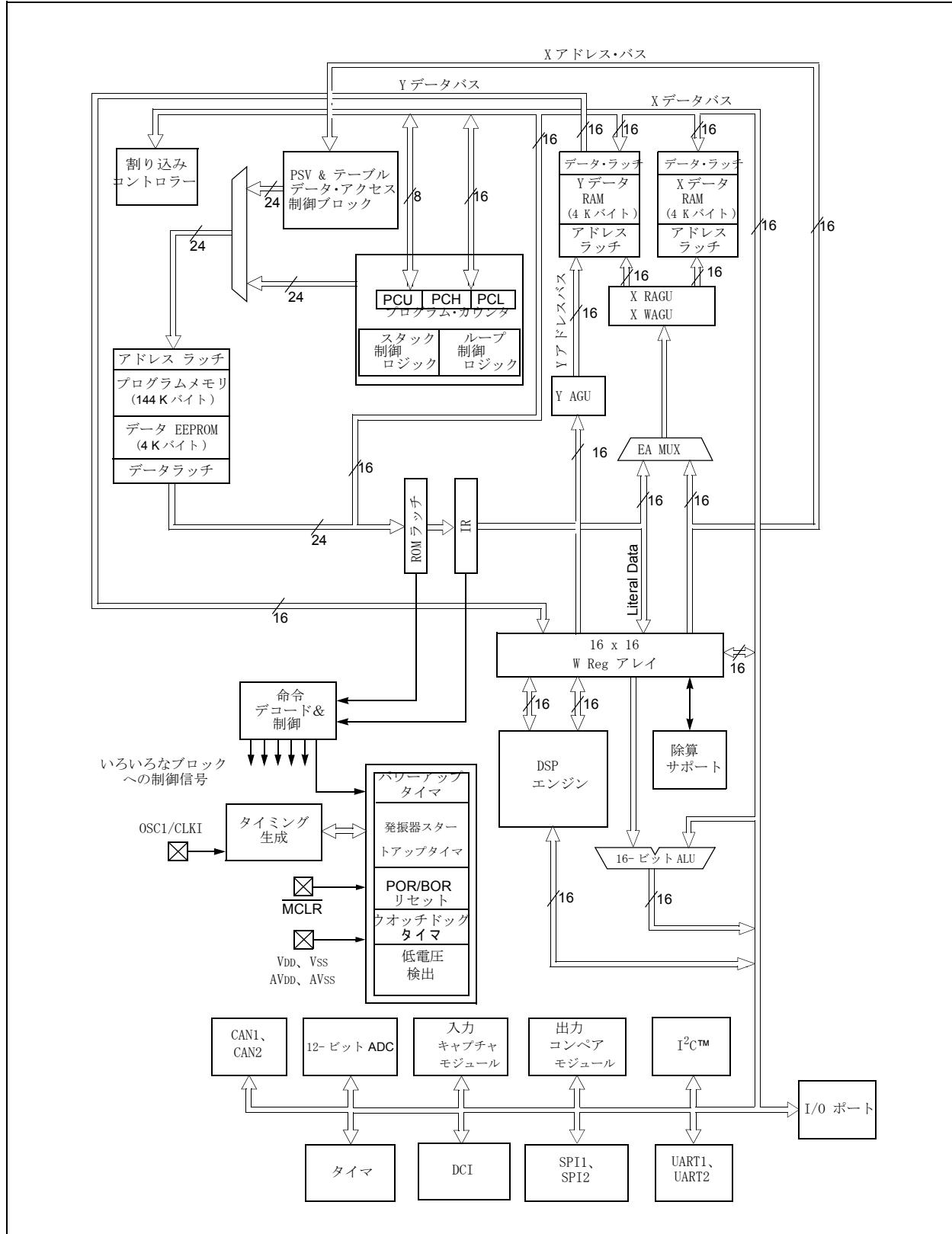
dsPIC30F の大部分の命令は、1 サイクルの間に、データ・メモリ読み込み (またはプログラムデータ)、作業用レジスタ (データ) 読み込み、データメモリ書き込みおよびプログラム (命令) メモリ読み込みが実行されます。結果として、3 つのオペランドを持つ命令がサポートされ、 $A+B=C$  の動作が 1 サイクルで実行できることになります。

DSP エンジンは、高速 17 ビット × 17 ビット乗算器、40 ビット ALU、2 つの 40 ビット飽和付アキュムレータおよび 40 ビット双方向のバレルシフタを搭載しています。バレルシフタにより、40 ビットの値を最大 15 ビット右、または最大 16 ビット左へのシフトを 1 サイクルで実行することができます。DSP 命令は他のすべての命令とシームレスに実行されて、最適なリアルタイムパフォーマンスが出るように設計されています。MAC 命令やその他の関連命令では、2 つのレジスタの乗算のために必要な 2 つのオペランドを、メモリから同時に取り出すことができます。そのためこれらの命令に対してはデータ空間が分割され、他の命令に対してはリニアなひとつの空間とすることが必要とされます。各アドレス空間ごとに作業用レジスタを割り付けることで、この問題を制限のない柔軟な方法で実現しています。

dsPIC30F は、ベクタ方式の例外処理機構をもつていて、8 個のノンマスカブルなトラップと、54 個の割り込みで構成されています。各割り込みソースが 7 つの優先レベルの 1 つに割り当てられます。

図 2-1 は CPU のブロック図を示します。

図 2-1: dsPIC30F CPU コア・ブロック図



## 2.2 プログラマ用モデル

プログラマ用モデル図 2-2 は dsPIC30F 用のプログラマ用モデルを示しています。プログラマ用モデル内の全てのレジスタがメモリにマップされており、命令によって直接的に操作できます。表 2-1 が各レジスタの説明です。

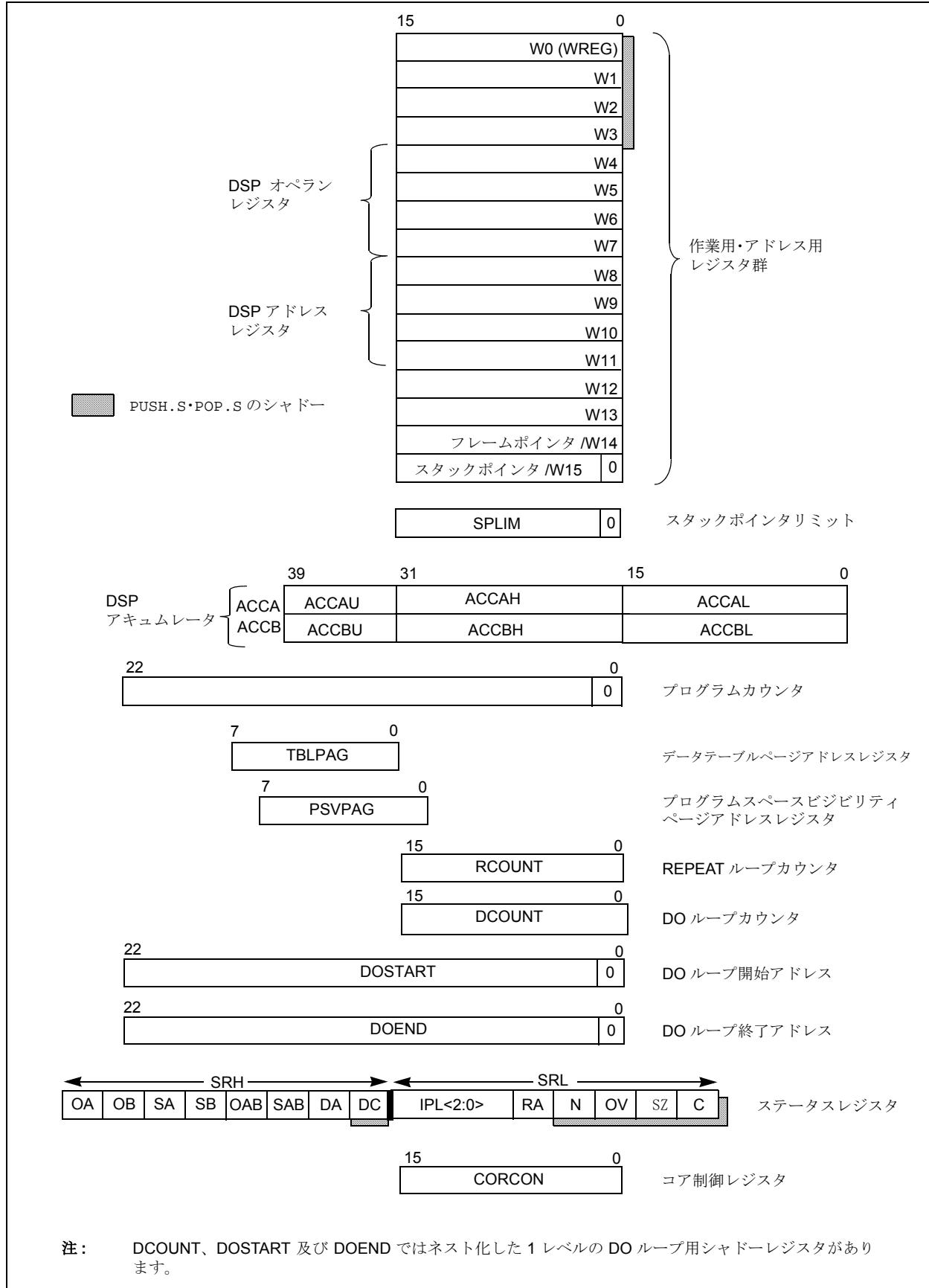
表 2-1: プログラマ用モデルのレジスタの説明

レジスタ名	説明
WO から W15	作業用レジスタアレイ
ACCA、ACCB	40 ビット DSP アキュムレータ
PC	23 ビットプログラムカウンタ
SR	ALU と DSP エンジンステータスレジスタ
SPLIM	スタックポインター制限値レジスタ
TBLPAG	テーブルメモリページアドレスレジスタ
PSVPAG	プログラムスペースビジビリティページアドレスレジスタ
RCOUNT	REPEAT ループカウントレジスタ
DCOUNT	DO ループカウントレジスタ
DOSTART	DO ループスタートアドレスレジスタ
DOEND	DO ループエンドアドレスレジスタ
CORCON	DSP エンジンと DO ループ制御ビットを含む

プログラマ用モデルに含まれているレジスタの他に dsPIC30F にはモジュロアドレッシング、ビットリバースアドレッシングおよび割り込みのための制御レジスタが含まれています。これらのレジスタについては後述します。

2-38 ページにおいて表 2-8 に示したようにプログラマ用モデルに含まれている全レジスタがメモリマップされています。

図 2-2: プログラマ用モデル



## 2.2.1 作業用レジスタアレイ

16 個の作業用レジスタ (W) はデータ、アドレス、またアドレスオフセットレジスタとして機能します。W レジスタの機能はアクセスする命令のアドレッシングモードで決定されます。

dsPIC30F 命令セットはレジスタおよびファイルレジスタ命令の 2 つの種類に分類されます。レジスタ命令は各 W レジスタをデータ値あるいはアドレスオフセット値として使用できます。例えば、以下のように設定します。

```
MOV    W0,W1      ; move contents of W0 to W1
MOV    W0,[W1]     ; move W0 to address contained in W1
ADD    W0,[W4],W5 ; add contents of W0 to contents pointed
                  ; to by W4. Place result in W5.
```

### 2.2.1.1 W0 及びファイルレジスタ命令

W0 はファイルレジスタ命令で使用される唯一の作業用レジスタであるため、特別な作業用レジスタです。ファイルレジスタ命令は命令 **opcode** または W0 で指定されたメモリアドレスに対して実行されます。W1-W15 はファイルレジスタ命令の対象レジスタとして指定できません。

ファイルレジスタ命令は W レジスタを 1 つだけしか持たない現行の PICmicro デバイスとの下位互換性があります。アンセンブラー記述の場合、「WREG」ラベルはファイルレジスタ命令の W0 を示すために使われます。例えば、以下のように設定します。

```
MOV    WREG,0x0100 ; move contents of W0 to address 0x0100
ADD    0x0100,WREG ; add W0 to address 0x0100, store in W0
```

**注：** アドレッシングモード及び命令構文の説明については、dsPIC30F プログラマリ ファレンスマニュアル (DS70032) を参照してください。

### 2.2.1.2 W レジスタメモリマッピング

W レジスタはメモリマップされているため、ファイルレジスタ命令で以下のようにして W レジスタにアクセスできます。

```
MOV    0x0004, W10 ; equivalent to MOV W2, W10
```

ここで 0x0004 は W2 のメモリアドレスです。

又、W レジスタをアドレスポインタとオペランド先の両方として使用する命令も実行できます。例えば、以下のように設定します。

```
MOV    W1,[W2++]
```

ここで：

```
W1 = 0x1234
W2 = 0x0004      ; [W2] addresses W2
```

上記の例では、W2 の内容は 0x0004 とします。W2 はアドレスポインタとして使用されるため、メモリ内の 0x0004 番地を指します。メモリでは、W2 はこの同じアドレスにマップされています。このような例はまれですが、実行時まで検出不可能です。dsPIC30F ではデータ書き込みが優先実行されるので、上例の結果は W2 = 0x1234 となります。

### 2.2.1.3 W レジスタ及びバイトモード命令

W レジスタアレイのみを対象とするバイト命令は対象レジスタの最下位バイトに影響します。作業用レジスタはメモリマップされるため、最下位バイトと最上位バイトはバイト単位のメモリアクセスで操作可能です。

## 2.2.2 シャドーレジスタ

プログラマ用モデルの複数のレジスタには図 2-2 に示すように関連シャドーレジスタが存在します。どのシャドーレジスタも直接アクセスすることはできません。シャドーレジスタには、PUSH.S と POP.S 命令で利用するレジスタおよび DO 命令で利用するレジスタの 2 種類があります。

### 2.2.2.1 PUSH.S と POP.S のシャドーレジスタ

PUSH.S と POP.S の命令は、関数呼び出し時または割り込みサービスルーチン(ISR)の実行中の、高速コンテキスト保存／復元のために便利に使えます。PUSH.S の命令は以下のレジスタ値をそれぞれのシャドーレジスタに転送します。

- W0...W3
- SR (N, OV, Z, C, DC ビットのみ)

POP.S 命令はシャドーレジスタからの値をそれぞれのレジスタ位置に復元します。PUSH.S と POP.S の命令を使用するコード例を以下に示します。

MyFunction:

```

PUSH.S           ; Save W registers, MCU status
MOV   #0x03,W0    ; load a literal value into W0
ADD   RAM100      ; add W0 to contents of RAM100
BTSC  SR,#Z       ; is the result 0?
BSET  Flags,#IsZero; Yes, set a flag
POP.S           ; Restore W regs, MCU status
RETURN

```

PUSH.S 命令はシャドーレジスタに保存されている内容に上書きします。シャドーレジスタの深度は一段階のみなので、シャドーレジスタを複数のソフトタスクで使用する場合は注意が必要です。

シャドーレジスタを使用するタスクの実行中には、それよりも優先順位の高いシャドーレジスタを使用するタスクが割り込まないことを確認する必要があります。優先順位の高いタスクが割り込むと、優先順位の低いタスクで保存されたシャドーレジスタの内容は上書きされてしまいます。

### 2.2.2.2 DO ループシャドーレジスタ

DO 命令を実行すると、以下のレジスタは自動的にシャドーレジスタに格納されます。

- DOSTART
- DOEND
- DCOUNT

DO シャドーレジスタの深度は一段階であり、2 つのループを自動的にネスト化できます。詳細は、セクション 2.9.2.2 「DO ループネスティング」を参照してください。

### 2.2.3 初期化されていない W レジスタ RESET

すべての RESET の場合、W レジスタアレイ (W15 以外) はクリアされ、書き込むまで初期化されていないものと見なされます。初期化されていないレジスタをアドレスポインタとして使用しようとするとデバイスはリセットされます。

W レジスタを初期化するためにはワード書き込みをする必要があります。バイト書き込みは初期化検出ロジックに影響しません。

## 2.3 ソフトウェアスタックポインタ

W15 は専用のソフトスタックポインタとして使用され、例外処理、サブルーチン CALL 命令と RETURN 命令で自動的に変更されます。W15 は他の W レジスタと同様のいずれの命令によっても参照可能です。これにより、スタックポインタの読み取り、書き込み、および操作が容易になります（例：スタックフレームの作成）。

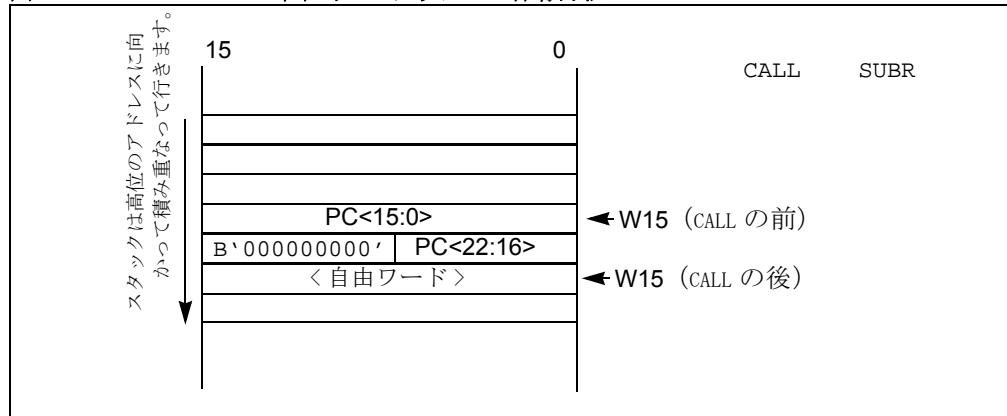
**注：** 誤ったスタックアクセスを防ぐために、W15 <0> はハードウェアによって '0' にセットされています。

W15 はすべての RESET 時 0x0800 に初期化されます。このアドレス初期化により、スタックポインタ (SP) がすべての dsPIC30F デバイスの有効な RAM を示すことが保証され、ユーザソフトでスタックポインタを初期化する前に、例外トラップが発生してしまうことを避ることができます。ユーザーはこの後の初期化の際、データ空間内の希望の位置に SP をプログラムし直すことができます。

スタックポインタは、常に初めの利用可能な空きワードを指しており、下位アドレスから上位アドレスに向かって昇順で格納します。図 2-3 に示すように、SP を減算してからスタックポップ（読み出し）を実行し、スタックプッシュ（書き込み）を実行してから SP を加算します。

PC がスタックにプッシュされるときには、PC<15:0> は初めの利用可能なスタックワードにプッシュされ、PC<22:16> は二番目の利用可能なスタック位置にプッシュされます。CALL 命令時の PC プッシュについては、PC の最上位バイトは図 2-3 のようにプッシュの前にゼロ拡張されます。例外トラップ処理の時は、PC の最上位バイトには CPU ステータスレジスタ、SR の下位の 8 ビットが連結されます。したがって、SRL の内容は割り込み処理の際に自動的に保存されます。

図 2-3: CALL 命令時のスタックへの保存方法



### 2.3.1 ソフトウェアスタック例

ソフトウェアスタックは PUSH と POP 命令を使って操作されます。PUSH および POP 命令は、W15 レジスタが移動先ポインタである MOV 命令に相当します。例えば、WO レジスタの内容は以下の操作によりスタック上にプッシュできます。

PUSH W0

この構文は以下のものに相当します。

MOV W0,[W15++]

スタックの一番上の内容を W0 には戻すには、次の操作を行います。

POP W0

この構文は以下のものに相当します。

MOV [--W15],W0

図 2-4 から図 2-7 までは、ソフトウェアスタックの使用法を示しています。図 2-4 はデバイス初期化の際のソフトウェアスタックを示します。W15 は 0x0800 にあらかじめ初期化されており、さらにこの例では 0x5A5A 及び 0x3636 がそれぞれ W0 および W1 に書き込まれていたものと想定します。図 2-5 で、スタックが初めてプッシュされ、WO 内の値がスタックにコピーされます。W15 は自動更新され、次のスタック位置を示します (0x0802)。図 2-6 では、W1 内容がスタック上にプッシュされます。図 2-7において、スタックはポップされ、前に W1 からプッシュされたスタックの 1 番上の値は、W3 に書き込まれます。

図 2-4: デバイスリセット時のスタックポインタ

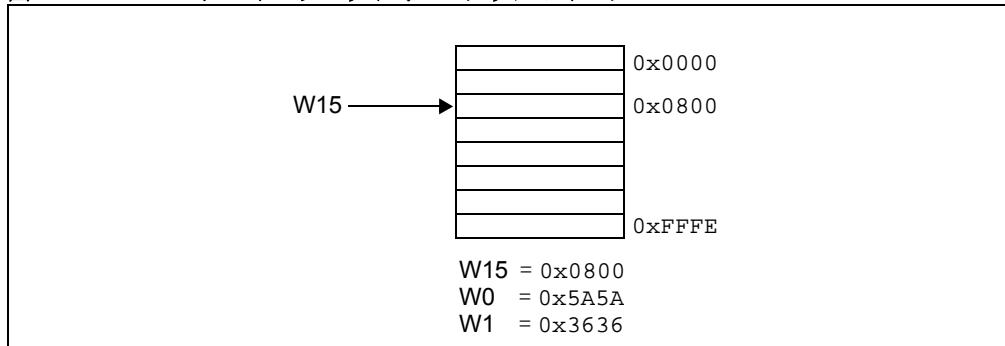


図 2-5: 1 番目 PUSH 命令後のスタックポインタ

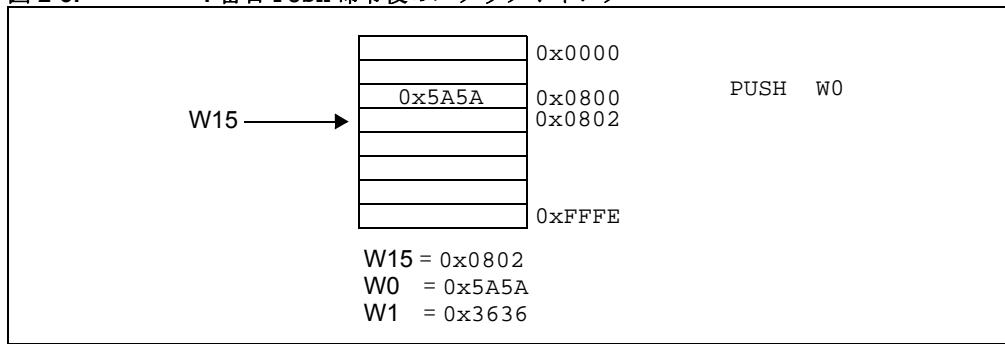


図 2-6: 1 度目の PUSH 命令後のスタックポインタ

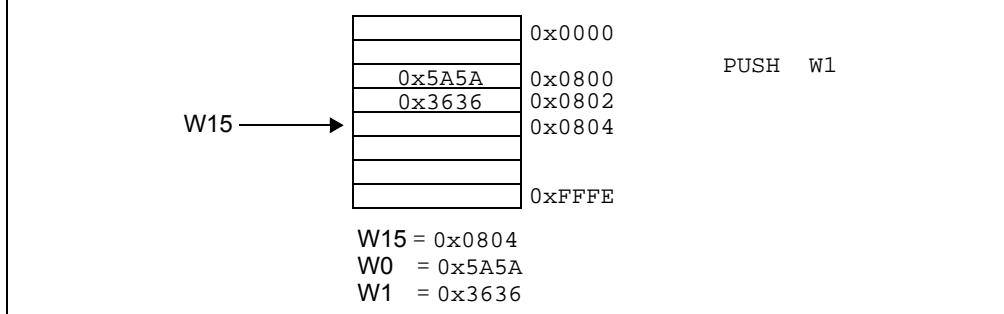


図 2-7: スタック POP 命令後のポインタ



### 2.3.2 W14 ソフトウェアスタックフレームポインタ

フレームとは、ひとつのサブルーチンが使用するスタック内のメモリのユーザー定義セクションのことです。W14 は LNK (リンク) 及び ULNK (アンリンク) 命令を用いてスタックフレームポインタとして使用できる特別な作業用レジスタです。W14 は、フレームポインタとして使用しない場合は通常の作業用レジスタとして使用できます。

スタックフレームポインタとして W14 を使用するソフトウェア例については、『dsPIC30F プログラマリファレンスマニュアル (DS70030)』を参照してください。

### 2.3.3 スタックポインタオーバーフロー

SPLIM レジスタ (Stack Limit Register) はスタックの上限を指定するレジスタでリセットにより 0x0000 に設定されます。SPLIM は 16 ビットレジスタですが、全スタック操作がワード単位で行われるため、SPLIM<0> は ‘0’ に固定されます。

スタックオーバーフローチェックは SPLIM へのワード書き込みが行われてから有効となり、以後、無効化するにはデバイスリセットが必要となります。W15 レジスタを実行元あるいは実行先として生成されたアドレスは、常に SPLIM レジスタと比較されています。もし生成されたアドレスが SPLIM 内容より大きい場合、スタックエラートラップが発生します。

スタックオーバーフローチェックが有効化された場合、W15 レジスタで生成されたアドレスが、データ空間の終わりのアドレス 0xFFFF を超えたときにもスタックエラートラップが発生します。

**注:** W15 を使用した間接読み取り操作に続いてスタックポインタリミットレジスタ、SPLIM への書き込みを行わないでください。

スタックエラートラップの詳細については、第 6 章「割り込み」を参照してください。

### 2.3.4 スタックポインタアンダーフロー

スタックは RESET の時 0x0800 に初期化されます。スタックポインタアドレスが 0x0800 より小さくなると、スタックエラートラップが発生します。

**注:** データ空間中、0x0000 から 0x07FF の間のロケーションは、一般的にコアおよび周辺モジュール制御用の特殊機能レジスタ (SFR) 用として確保されています。

## 2.4 CPU レジスタの説明

### 2.4.1 SR: CPU ステータスレジスタ

dsPIC30F CPU は 16 ビットステータスレジスタ (SR) を有し、その下位バイトは下位ステータスレジスタ (SRL) と称されます。SR の上位バイトは SRH と称されます。SR の詳細な説明はレジスタ 2-1 項にあります。

SRL には全 MCU の ALU 実行結果のステータスフラグに加え、CPU 割り込み優先ステータスビット、IPL<2:0> と REPEAT ループアクティブステータスピット、RA (SR<4>) も含まれています。例外トラップ処理の場合の SRL は、PC の最上位バイトと連結したワードとして構成され、その後でスタックに保存されます。

SRH には DSP 加算器・減算器ステータスピット、DO ループアクティブビット、DA (SR<9>) およびデジットキャリービット、DC (SR<8>) が含まれています。

下記のものを除いて SR ビットは読み取り・書き込み可能です。

1. DA ビット (SR<8>) : DA は読み取り専用ビット。
2. RA ビット (SR<4>) : RA は読み取り専用ビット。
3. OA、OB (SR<15:14>) 及び OAB (SR<11>) ビット : これらのビットは読み取り専用で DSP エンジンハードウェアだけが書き込みができます。
4. SA、SB (SR<13:12>) 及び SAB (SR<10>) ビット : これらのビットは読み取りとクリア専用で DSP エンジンハードウェアだけがセットできます。一度セットされると、次の DSP 操作の結果にかかわらず、ユーザーがクリアするまでビットはセットされた状態のままでです。

**注:** SAB をクリアすると、SA 及び SB ビットもクリアされます。

**注:** 各命令が影響を及ぼす SR ビットの説明は dsPIC30F プログラマリファレンスマニュアル (DS70030) に記載されています。

### 2.4.2 CORCON: コア制御レジスタ

CORCON レジスタには DSP 乗算器と DO ループハードウェアの操作を制御するビットが含まれています。CORCON レジスタには IPL3 ステータスピットも含まれていますが、これは IPL<2:0>(<SR<7:5>) と連結して CPU 割り込み優先レベルを形成します。

## レジスタ 2-1: SR: CPU ステータスレジスタ

上位バイト:

R-0	R-0	R/C-0	R/C-0	R-0	R/C-0	R-0	R/W-0
OA	OB	SA	SB	OAB	SAB	DA	DC
ビット 15				ビット 8			

下位バイト: (SRL)

R/W-0 <sup>(2)</sup>	R/W-0 <sup>(2)</sup>	R/W-0 <sup>(2)</sup>	R-0	R/W-0	R/W-0	R/W-0	R/W-0
IPL<2:0>	RA	N	OV	Z	C		
ビット 7				ビット 0			

ビット 15 **OA:** アキュムレータ A オーバーフローステータスピット

1 = アキュムレータ A はオーバーフローした。

0 = アキュムレータ A はオーバーフローしなかった。

ビット 14 **OB:** アキュムレータ B オーバーフローステータスピット

1 = アキュムレータ B はオーバーフローした。

0 = アキュムレータ B はオーバーフローしなかった。

ビット 13 **SA:** アキュムレータ A 飽和ステータスピット

1 = アキュムレータ A は飽和したまたは飽和している。

0 = アキュムレータ A は飽和しなかった。

注: このビットは読み取り又は消去可能(セット不可)。

ビット 12 **SB:** アキュムレータ B 飽和ステータスピット

1 = アキュムレータ B は飽和したまたは飽和している。

0 = アキュムレータ B は飽和しなかった。

注: このビットは読み取り又は消去可能(セット不可)。

ビット 11 **OAB:** OA || OB 結合アキュムレータオーバーフローステータスピット

1 = アキュムレータ A または B はオーバーフローした。

0 = アキュムレータ A または B はオーバーフローしなかった。

ビット 10 **SAB:** SA || SB 結合アキュムレータステータスピット

1 = アキュムレータ A または B は飽和しているまたは飽和した。

0 = アキュムレータ A または B は飽和しなかった。

注: このビットは読み取り又は消去可能(セット不可)。このビットをクリアすると、SA と SB もクリアされます。

ビット 9 **DA:** DO ループアクティブビット

1 = DO ループ進行中。

0 = DO ループは進行中でない。

ビット 8 **DC:** MCU ALU ハーフキャリーまたはボロービット

1 = 第 4 下位ビットよりキャリーアウト(バイトサイズデータ用)または結果の第 8 下位ビット(ワードサイズデータ用)が発生した。

0 = 第 4 下位ビットよりキャリーアウトなし(バイトサイズデータ用)または結果の第 8 下位ビット(ワードサイズデータ用)の発生なし。

## レジスタ 2-1: SR: CPU ステータスレジスタ (続き)

ビット 7-5 IPL<2:0> : CPU 割り込み優先順位ステータスピット<sup>(1)</sup>

- 111 = CPU 割り込み優先順位は 7 (15). ユーザ割り込み無効にした。  
 110 = CPU 割り込み優先順位は 6 (14)  
 101 = CPU 割り込み優先順位は 5 (13)  
 100 = CPU 割り込み優先順位は 4 (12)  
 011 = CPU 割り込み優先順位は 3 (11)  
 010 = CPU 割り込み優先順位は 2 (10)  
 001 = CPU 割り込み優先順位は 1 (9)  
 000 = CPU 割り込み優先順位は 0 (8)

注 1: IPL<2:0> ビットは IPL<3> ビット (C0RC0N<3>) に連結して CPU 割り込み優先順位を構成します。カッコ内の値は IPL<3> = I の場合の IPL を示します。IPL<3> = I の場合、ユーザー割り込みは無効になります。

2: NSTDIS = I (INTCON1<15>) の場合、IPL<2:0> ステータスピットは読み込み専用です。

## ビット 4 RA: REPEAT ループアクティブビット

- 1 = REPEAT ループ進行中  
 0 = REPEAT ループは進行中でない。

## ビット 3 N: MCU ALU 負ビット

- 1 = 結果は負。  
 0 = 結果は負でなかった (ゼロ又は正)。

## ビット 2 OV: MCU ALU オーバーフロー ビット

- このビットは符号付き算術演算 (2 の補数) 用に使用されます。符号ビットのが変わる大きさのオーバーフローが発生したことを表す。  
 1 = 符号付き算術演算でオーバーフロー発生 (この算術演算内)。  
 0 = オーバーフロー発生無し。

## ビット 1 Z: MCU ALU ゼロビット

- 1 = Z ビットがセットされる演算が実行された。  
 0 = Z ビットがクリアされる演算が実行された。 (すなわち、非ゼロ結果)。

## ビット 0 C: MCU ALU キャリー・ボロー ビット

- 1 = 発生した結果の最上位のビットよりキャリーアウト。  
 0 = 発生した結果の最上位のビットよりキャリーアウト無し。

凡例 :

R = 読み込み可能なビット	W = 書き込み可能なビット	U = 実施されないビット、'0' として読み込む
C = ビットのみクリア	S = ビットのみセット	-n = POR への値
'1' = ビットをセット	'0' = ビットをクリア	x = ビットは不明

## レジスタ 2-2: CORCON: コア制御レジスタ

上位バイト:							
U-0	U-0	U-0	R/W-0	R/W-0	R-0	R-0	R-0
—	—	—	US	EDT	DL<2:0>		
ビット 15							ビット 8

下位バイト:								
R/W-0	R/W-0	R/W-1	R/W-0	R/C-0	R/W-0	R/W-0	R/W-0	R/W-0
SATA	SATB	SATDW	ACCSAT	IPL3	PSV	RND	IF	
ビット 7							ビット 0	

ビット 未使用: ‘0’ と読み取り

15-13

ビット **US:** DSP 乗算符号無し・符号有り制御ビット

12 1 = DSP エンジン乗算符号無し。  
0 = DSP エンジン乗算符号有り。

ビット **EDT:** 高速 DO ループ 終了制御ビット

11 1 = 現在ループ繰り返しの終わりに実行中 DO ループを終了します。  
0 = 無効。

注: このビットはいつも ‘0’ と読み取られます。

ビット **DL<2:0>:** DO ループネスティングレベルステータスビット

10-8 111 = 7 DO ループアクティブ

•

•

001 = 1 DO ループアクティブ

000 = 0 DO ループアクティブ

ビット 7 **SATA:** AccA 飽和有効ビット

1 = アキュムレータ A 飽和は有効。  
0 = アキュムレータ A 飽和は無効。

ビット 6 **SATB:** AccB 飽和有効ビット

1 = アキュムレータ B 飽和は有効。  
0 = アキュムレータ B 飽和は無効。

ビット 5 **SATDW:** DSP エンジンによるデータスペース書き込み飽和有効ビット

1 = データスペース書き込み飽和は有効。  
0 = データスペース書き込み飽和は無効。

ビット 4 **ACCSAT:** アキュムレータ飽和モード選択ビット

1 = 9.31 飽和 (超飽和)。  
0 = 1.31 飽和 (正常飽和)。

ビット 3 **IPL3:** CPU 割り込み優先レベルステータスビット 3

1 = CPU 割り込み優先レベルは 7 以上。  
0 = CPU 割り込み優先レベルは 7 以下。

注: IPL3 ビットは IPL<2:0> ビット (SR<7:5>) と連結して CPU 割り込み優先レベルを形成します。

## レジスタ 2-2: CORCON: コア制御レジスタ (続き)

ビット 2 **PSV:** プログラムメモリをデータメモリ空間に割り当てる制御

- 1 = データ空間にプログラムメモリを割り当てる
- 0 = データ空間には割り当てない

ビット 1 **RND:** 丸めモード選択ビット

- 1 = バイアス(従来)丸め有効。
- 0 = アンバイアス(収束)丸め有効。

ビット 0 **IF:** 整数または小数乗算器モード選択ビット

- 1 = DSP 乗算整数モード。
- 0 = DSP 乗算小数モード。

凡例:

<b>R</b> = 書き込み可能	<b>W</b> = 読み込み可能	<b>U</b> = 未使用ビット、'0' として読み込	<b>C</b> = ビットクリアできる
なビット	なビット	む	
-n = POR 後の値	'1' = ビットセッ	'0' = ビットをクリア	x = ビットは不定
ト			

## 2.4.3 他の dsPIC30F CPU 制御レジスタ

以下にリスト化されたレジスタは、dsPIC30F CPU コアに関連するレジスタです。詳細は本マニュアルの指定セクションを参照してください。

### 2.4.3.1 TBLPAG: テーブルページレジスタ

TBLPAG レジスタは、テーブルの読み取りと書き込み操作時、プログラムメモリアドレスの上位 8 ビットを保持します。テーブル命令は、プログラムメモリスペースとデータメモリスペース間のデータ転送に使用します。詳細は 第4章 .「プログラムメモリ」を参照してください。

### 2.4.3.2 PSVPAG: プログラム空間可視性ページレジスタ

プログラムメモリ空間をデータメモリ空間に割り当てるため、ユーザーはプログラムメモリ空間の 32 キロバイトセクションをデータアドレス空間の上位 32 キロバイトにマップできます。この特徴のため、データメモリをアクセスする dsPIC30F 命令を使って定数データを直接アクセスできます。PSVPAG レジスタは、データアドレス空間にマップするプログラムメモリ空間の 32 キロバイト領域を選択します。PSVPAG レジスタの詳細情報は、第4章 .「プログラムメモリ」を参照してください。

### 2.4.3.3 MODCON: モジュロ制御レジスタ

MODCON レジスタはモジュロアドレス（サーキュラバッファ）を有効化し構成するのに使用されます。モジュロアドレスの詳細は、第3章 .「データメモリ」を参照してください。

### 2.4.3.4 XMODSRT, XMODEEND: X モジュロの開始と終了アドレスレジスタ

XMODSRT および XMODEEND レジスタは、X データメモリアドレス空間で実行されるモジュロ（サーキュラ）バッファのための開始と終了アドレスを保持します。モジュロアドレスの詳細は、第3章 .「データメモリ」を参照してください。

### 2.4.3.5 YMDSRT, YMODEEND: Y モジュロの開始と終了アドレスレジスタ

YMODSRT および YMODEEND レジスタは、Y データメモリアドレス空間で実行されるモジュロ（サーキュラ）バッファのための開始と終了アドレスを保持します。モジュロアドレスの詳細は、第3章 .「データメモリ」を参照してください。

### 2.4.3.6 XBREV: X モジュロビットリバースレジスタ

XBREV レジスタは、ビットリバースアドレスのためのバッファサイズを設定するのに使用されます。ビットリバースアドレスの詳細は、第3章 .「データメモリ」を参照してください。

### 2.4.3.7 DISICNT: 割込みカウントレジスタの無効化

DISICNT レジスタは、指定されたサイクル時間だけ優先順位 1-6 の割り込みを無効にするために DISI 命令によって使われます。詳細は、第6章 .「割り込み」を参照してください。

## 2.5 論理演算ユニット (ALU)

dsPIC30F の ALU は 16 ビット幅であり、加算、減算、1 ビットシフトおよび論理演算が可能です。特に指定のない限り、算術演算は 2 の補数です。算術演算によって、ALU は SR レジスタのキャリー (C)、ゼロ (Z)、負 (N)、オーバーフロー (OV) およびディジットキャリー (DC) ステータスピットの値を書き替えます。C および DC ステータスピットは減算操作のときにはそれぞれボロウビットとデジットボロウビットとして機能します。

ALU は使用される命令モードによって 8 ビットまたは 16 ビット操作を実行します。命令のアドレッシングモードに応じて、ALU 操作のためのデータは W レジスタアレイあるいはデータメモリから取得されます。同様に、ALU からの出力データは W レジスタアレイまたはデータメモリに書き込まれます。

各命令、アドレッシングモードおよび 8 ビット /16 ビット命令モードに影響される SR ビットの詳細は、dsPIC30F プログラマリファレンスマニュアル (DS70030) を参照してください。

- 注 1:** バイト操作は 16 ビット ALU を使用して 8 ビット以上の結果を生じます。ただし、PICmicro デバイスとの下位互換性を維持するように、すべてのバイト操作からの ALU 結果はバイト（すなわち、最上位バイトは変更されない）として書き戻され、結果の下位バイトの状態のみに基づいて SR レジスタが更新されます。
- 2:** バイトモードだけで実行されたすべてのレジスタ命令は W レジスタの下位バイトのみに影響します。いずれの W レジスタの最上位バイトもファイルレジスタ命令を使ってメモリアドレスで指定することでアクセスすることができます。

### 2.5.1 バイトのワードへの変換

dsPIC30F には 8 ビットおよび 16 ビット ALU 操作を組み合わせる時に有効な 2 つの命令を持っています。符号拡張 (SE) 命令は W レジスタまたはデータメモリからバイト値を取り出し、符号つきワードデータとして W レジスタに格納します。ゼロ拡張 (ZE) 命令は W レジスタまたはデータメモリにあるワード値の上位 8 ビットをクリアして結果を W レジスタに保存します。

## 2.6 DSP エンジン

DSP エンジンは W レジスタアレイからデータを取得するハードウェアのブロックですが、専用の結果用レジスタも持っています。MCU ALU を使うのと同じ命令デコーダによって DSP エンジンが 駆動されます。また、すべてのオペランド有効アドレス (EAs) が W レジスタアレイ用に生成されます。MCU 命令フローとの同時実行は不可能ですが、MCU ALU と DSP エンジンの両方のリソースは全命令で共有されて使われます。

DSP エンジンは以下のコンポーネントで構成されています。

- 高速 17 ビット × 17 ビット乗算器
- バ렐シフタ
- 40 ビット加算器／減算器
- 2 つのアキュムレータレジスタ
- 選択可能なモードをもつ丸めロジック
- 選択可能なモードをもつた飽和ロジック

以下のいずれかのソースから DSP にデータが入力されます。

1. 二重ソースオペランド DSP 命令用 W アレイ（レジスタ W4、W5、W6 または W7）から直接入力する。W4、W5、W6、および W7 レジスタ用のデータ値は X および Y メモリデータバスを通じてプリフェッチされる。
2. 他のすべての DSP 命令のときは、X メモリデータバスから入力する。

以下のいずれかの送り先に DSP エンジンからのデータ出力が書き込まれます。

1. 実行中の DSP 命令で定義された対象アキュムレータ。
2. データメモリアドレス空間内の X データメモリ。

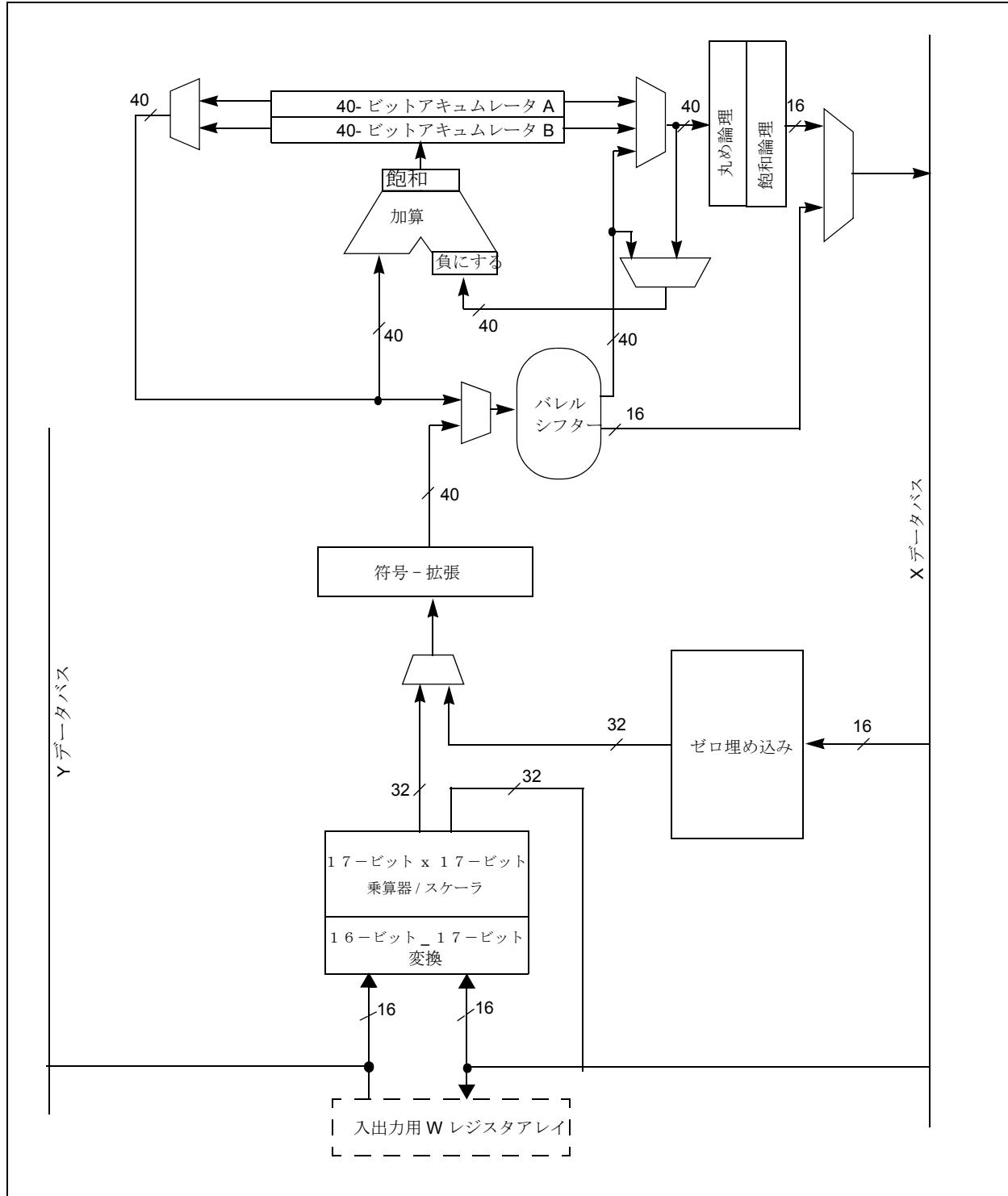
DSP エンジンは追加データの必要がないアキュムレータ操作には固有のアキュムレータを使うことができます。

MCU シフトと乗算命令は命令実行に DSP エンジンハードウェアを使用します。その実行に必要なデータの読み取りと書き込みのために X メモリデータを使用します。

図 2-8 は DSP エンジンのブロック図表です。

**注：** 本セクションに関するコード例と命令構文の詳細は、**dsPIC30F プログラマリファレンスマニュアル (DS70030)** を参照してください。

図 2-8: DSP エンジンブロック図



## 2.6.1 データアキュムレータ

ACCA と ACCB の 2 つの 40 ビットデータアキュムレータがあり、表 2-3 に示された DSP 命令の実行結果格納用として使われます。各アキュムレータは 3 つのレジスタにメモリマップされ、そのレジスタ名の ‘x’ は特定のアキュムレータを示します。

- ACCxL: ACCx<15:0>
- ACCxH: ACCx<31:16>
- ACCxU: ACCx<39:32>

アキュムレータを使用する固定小数演算では、小数点はビット 31 の右に位置します。各アキュムレータに保存できる固定小数値の範囲は -256.0 から (256.0 - 2<sup>-31</sup>) までです。アキュムレータを使用する整数操作では、小数点はビット 0 の右に位置します。各アキュムレータに保存できる整数値の範囲は -549,755,813,888 から 549,755,813,887 までです。

## 2.6.2 乗算器

dsPIC30F は MCU ALU および DSP エンジン両方により共有される乗算機を持っています。乗算器では符号付きおよび符号無しオペレーションの両方が可能で、1.31 固定小数 (Q.31) または 32 ビット整数のいずれかをサポートできます。

乗算器は 16 ビットインプットデータを取り込んでそのデータを 17 ビットに変換します。乗算器に入力される符号付きオペランドは符号拡張され、符号無し入力オペランドはゼロ拡張されます。17 ビット変換論理はユーザーには制限を与えず、乗算器で混合符号および符号無し/符号無し乗算することができます。

IF 制御ビット (CORCON<0>) は表 2-3 記載の命令の整数/固定小数オペレーションを決定します。IF ビットは表 2-4 記載の MCU 乗算命令に影響しません。表 2-4 記載の MCU 乗算命令は常に整数オペレーションです。乗算器は固定小数オペレーション用に結果を 1 ビット左ヘシフトします。このため結果の最下位ビットは常に 0 となります。乗算器は RESET で DSP オペレーションを固定小数モードに初期設定します。

これらの各モードでのハードウェアのデータ表現は以下の通りです：

- 整数データは本質的に符号付き 2 の補数値として表示され、一方最上位ビットは符号ビットとして定義されます。一般的に、N ビット 2 の補数整数の範囲は  $-2^{N-1}$  から  $2^{N-1} - 1$  です。
- 固定小数データは 2 の補数の固定小数として表示され、最上位ビットは符号ビットとして定義されます。小数点は符号ビット (Q.X フォーマット) のすぐ後にあるものとします。この小数点位置の N ビット 2 の補数の固定小数の範囲は -1.0 から  $(1 - 2^{1-N})$  です。

図 2-9 および図 2-10 は整数および固定小数モードのデータを乗算器ハードウェアがどのように解釈するかを示しています。整数および固定小数の両方のモードのデータ範囲は表 2-2 に記載されています。

図 2-9: 0x4001 の整数および固定小数表示

0x4001 の異なる表現

整数 :

0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
$-2^{15}$	$2^{14}$	$2^{13}$	$2^{12}$	...												$2^0$

$$0x4001 = 2^{14} + 2^0 = 16385$$

1.15 固定小数 :

0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
$-2^0$	$\bullet$	$2^{-1}$	$2^{-2}$	$2^{-3}$	...											$2^{-15}$

暗黙の小数点位置

$$0x4001 = 2^{-1} + 2^{-15} = 0.500030518$$

図 2-10: 0xC002 の整数および固定小数表示

0xC002 の異なる表現

整数 :

1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
$-2^{15}$	$2^{14}$	$2^{13}$	$2^{12}$	...												$2^0$

$$0xC002 = -2^{15} + 2^{14} + 2^0 = -32768 + 16384 + 2 = -16382$$

1.15 固定小数 :

1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
$-2^0$	$\bullet$	$2^{-1}$	$2^{-2}$	$2^{-3}$	...											$2^{-15}$

暗黙の小数点位置

$$0xC002 = -2^0 + 2^{-1} + 2^{-14} = -1 + 0.5 + 0.000061035 = -0.499938965$$

表 2-2: dsPIC30F データ範囲

レジスタ サイズ	整数範囲	固定小数範囲	固定小数解像度
16- ビット	-32768 to 32767	-1.0 to (1.0 - $2^{-15}$ ) (Q.15 書式)	$3.052 \times 10^{-5}$
32- ビット	-2,147,483,648 to 2,147,483,647	-1.0 to (1.0 - $2^{-31}$ ) (Q.31 書式)	$4.657 \times 10^{-10}$
40- ビット	-549,755,813,888 to 549,755,813,887	-256.0 to (256.0 - $2^{-31}$ ) (Q.31 8 ガードビットの書式)	$4.657 \times 10^{-10}$

## 2.6.2.3 DSP 乗算命令

アキュムレータを利用する DSP 命令は表 2-3 にまとめられます。

表 2-3: アキュムレータを利用する DSP 命令

DSP 命令	説明	代数方程式
MAC	乗算してアキュムレータに加算するまたは二乗してアキュムレータに加算する	$a = a + b^*c$ $a = a + b^2$
MSC	乗算してアキュムレータから減算する	$a = a - b^*c$
MPY	乗算する	$a = b^*c$
MPY.N	乗算して結果を負にする	$a = -b^*c$
ED	部分ユークリッド距離	$a = (b - c)^2$
EDAC	アキュムレータに部分ユークリッド距離を加算する	$a = a + (b - c)^2$

注： アキュムレータを利用する DSP 命令は小数 (1.15) のモードまたは整数モードで実行可能です。

US 制御ビット (CORCON<12>) は、DSP 乗算命令が符号有り（デフォルト）か符号無しかを決定します。US ビットは、符号有りまたは符号無しの実行が可能な MCU 乗算命令に影響しません。US ビットをセットすると、表 2-3 に記述された命令用の入力オペランドは乗算データの 17 番目のビットに常にゼロ拡張される符号無し値と見なされます。

## 2.6.2.2 MCU 乗算命令

本乗算器は、MCU 乗算命令のサポートにも使用されます。この命令は、表 2-4 に示される、整数 16 ビット符号有り、符号無しおよび混合符号乗算操作を含みます。MUL 命令で実行されたすべての乗算操作は整数結果になります。MUL 命令はバイトまたはワードのサイズのオペランドを使用するよう指示することができます。W アレイにある指定されたレジスタに対して、バイト入力オペランドは 16 ビット結果を生じ、ワード入力オペランドは 32 ビット結果を生じます。

表 2-4: アキュムレータを利用する MCU 命令

MCU 命令	説明
MUL/MUL.UU	2 つの符号無し整数を乗算する
MUL.SS	2 つの符号有り整数を乗算する
MUL.SU/MUL.US	符号有り整数と符号無し整数を乗算する

注 1: 乗算器を利用する MCU 命令は整数モードでのみ動作します。

2: MCU 乗算の結果は 32 ビットになり、2 つの W レジスタに保存されます。

### 2.6.3 固定小数データアキュムレータ加算器・減算器

データアキュムレータは、40ビット幅の加算器／減算器と結合されており、(符号付きの場合)乗算器の結果に対して自動的に符号を拡張付加する機能を持っています。2つのアキュムレータのいずれか(AまたはB)を選択して、演算前に加算するか、後で加算するかを指定することができます。ADD(アキュムレータ)およびLAC命令に対しては、加算するデータまたはロードするデータを、加算する前にパレルシフタを使って桁合わせすることができます。

40ビット加算器／減算器は、いずれかのオペランド入力を無効化することで、オペランドを変更せずに結果の符号を変更することができます。この機能は乗算および減算(MSC)、または乗算および負にする(MPY.N)時に実行されます。

40ビット加算／減算には、アキュミュレータ飽和制御ブロックが追加されていて、飽和制御を行うように指定されていると動作するようになっています。

#### 2.6.3.1 アキュムレータステータスビット

飽和およびオーバーフローをサポートするために6つのステータスレジスタビットがあります。それらはCPUステータスレジスタSRにあり、以下の通りです：

表2-5: アキュムレータオーバーフローおよび飽和ステータスピット

ステータスピット	ロケーション	説明
OA	SR<15>	アキュムレータAがガードビットにオーバーフローした(ACCA<39:32>)
OB	SR<14>	アキュムレータBがガードビットにオーバーフローした(ACCB<39:32>)
SA	SR<13>	ACCAが飽和した(ビット31オーバーフローおよび飽和)またはACCAがガードビットにオーバーフローして飽和した(ビット39オーバーフローおよび飽和)
SB	SR<12>	ACCBが飽和した(ビット31オーバーフローおよび飽和)またはACCBがガードビットにオーバーフローして飽和した(ビット39オーバーフローおよび飽和)
OAB	SR<11>	OAとOBのOR
SAB	SR<10>	SAとSBのOR SABを消去すればSAおよびSBも消去される。

OAおよびOBビットは読み出し専用で、データがアキュムレータの加算／減算論理を通るたびに変更されます。1にセットされている時は、最も最近のオペレーションがアキュムレータガードビットにオーバーフローしたことを示します(ビット32から39)。このタイプのオーバーフローは壊滅的ではなく、ガードビットがアキュムレータデータを保存します。OABステータスピットはOAおよびOBの論理和の値です。

OAおよびOBビットが1にセットされた場合、算術エラートラップを生成することができます。トラップは対応するオーバーフロートラップフラグ有効ビットOVATEをセットすれば有効になります:OVBTE(INTCON1<10:9>)。トラップイベントにより、ユーザーは希望する場合に迅速な修正措置を取ることができます。

SAおよびSBビットはデータがアキュムレータの飽和論理を通過するごとにセットされます。いつたんセットされると、これらのビットはユーザーが消去するまでセットされたままとなります。SABステータスピットはSAおよびSBの論理和の値を示します。SABが消去されればSAおよびSBビットも消去されます。これらのビットがセットされている場合、アキュムレータがその最大範囲をオーバーフローし(32ビット飽和ではビット31、40ビット飽和ではビット39)、飽和していることを示します(飽和制御が有効な場合)。

飽和制御が有効ではない場合、SAおよびSBビットは壊滅的オーバーフローが発生したことを示します(アキュムレータの符号が壊れています)。COVTE(INTCON1<8>)ビットがセットされると、飽和が無効化されたときSAおよびSBビットにより算術エラートラップを生成します。

**注:** 算術警告トラップに関して詳しくは第6章「割り込み」を参照してください。

**注:** ユーザーは、SA, SBおよびSABステータスピットはアキュムレータ飽和制御が有効であるか否かによって異なる意味を有することを忘れてはいけません。アキュムレータ飽和モードはCORCONレジスタによって制御されます。

## 2.6.3.2 飽和およびオーバーフローモード

デバイスは3つの飽和およびオーバーフローモードをサポートします。

### 1. アキュムレータ 39 ビット飽和：

このモードでは、アキュミュレータは、正の最大値 9.31(0x7FFFFFFF) から負の最大値 9.31 (0x8000000000) までの範囲の値を扱うことができます。SA または SB ビットがセットされると、ユーザーが消去するまでセットされたままです。この飽和モードはアキュムレータのダイナミック・レンジを拡張するのに役立ちます。

この飽和モードの設定には、ACCSAT(CORCON<4>) ビットをセットする必要があります。加えて、アキュムレータ飽和が有効になるように SATA および／または SATB(CORCON<7 および／または 6>) ビットをセットする必要があります。

### 2. アキュムレータ 31 ビット飽和：

このモードでは、アキュミュレータは正の最大値 1.31 (0x007FFFFFFF) から負の最大値 1.31(0xFF80000000) までの範囲の値を扱うことができます。SA または SB ビットがセットされると、ユーザーが消去するまでセットされたままとなります。この飽和モードが有効の場合、アキュムレータの符号拡張子以外にガードビット 32～39 は使用されません。従って、SR の OA、OB、OAB ビットはセットされません。

このオーバーフローと飽和モードを設定するには、ACCSAT (CORCON<4>) ビットを 0 クリアする必要があります。加えて、アキュムレータ飽和が有効になるように SATA および／または SATB (CORCON<7 および／または 6>) ビットをセットする必要があります。

### 3. アキュムレータの壊滅的オーバーフロー：

SATA および／または SATB (CORCON<7 および／または 6>) ビットがセットされていない場合、アキュムレータでの飽和制御は実行されず、アキュムレータはビット 39 までオーバーフローすることができます（符号を破壊します）。COVTE ビット (INTCON1<8>) がセットされている場合、壊滅的オーバーフローが発生した場合すると算術演算エラートラップを発生します。

アキュムレータ飽和およびオーバーフロー検出は 40 ビット DSP ALU を使って 2 つのアキュムレータのうちの 1 つを変更する DSP 命令の実行のときにのみ発生することに注意してください。MCU クラス命令を使ってアキュムレータがメモリにマップされたレジスタとしてアクセスされた場合には飽和およびオーバーフロー検出は行われません。さらに、表 2-5 に示したアキュムレータステータスピットも変更されません。ただし、MCU ステータスピット (Z, N, C, OV, DC) はアキュムレータにアクセスする MCU 命令に基づいて変更されます。

**注：** 算術演算エラートラップに関して詳しくは第 6 章、「割り込み」を参照してください。

## 2.6.3.3 データメモリ書き込み飽和

加算 / 減算による飽和に加えて、データメモリへの書き込みはソースのアキュムレータのコンテンツに影響を及ぼすことなく飽和を検出することができます。この機能により、途中の計算段階でアキュムレータの動作中の内容を犠牲にすることなくデータを制限することができます。データメモリ書き込み飽和は SATDW 制御ビット (CORCON<5>) をセットすれば有効になります。データメモリ書き込み飽和はデバイス RESET で有効になります。

データメモリ書き込み飽和機能は SAC および SAC.R 命令と共に機能します。これらの命令の実行中はアキュムレータの値は決して変更されません。ハードウェアは以下のステップによって飽和書き込み結果を検出します：

- 読み出しデータは命令に指定されている算術シフト値に基づいて縮小されます。
- 縮小時にはデータは丸められます (SAC.R のみ)。
- 縮小 / 丸め値はガードビットを使って 16 ビット値で飽和させます。0x007FFF より大きいデータの場合には、メモリに書き込まれるデータは正の 1.15 値の最大値である 0x7FFF で飽和とします。0xFF8000 以下のデータの場合には、メモリに書き込まれるデータは、負の 1.15 値の最大値である 0x8000 で飽和とし、いわゆるクリッピングされた値となります。

### 2.6.3.4 アキュムレータ ‘書き戻し’

MAC および MSC 命令は、実行中の命令の対象ではないアキュムレータの丸めたものをデータメモリに書き込むことができます。書き込みは X-bus を使って、X と Y アドレス空間を結合した範囲に対して実行されます。このアキュムレータ書き戻し機能は FFT および LMS アルゴリズムの時に便利に使えます。

アキュムレータ書き戻しハードウェアは以下のアドレスモードをサポートしています：

1. W13、レジスタダイレクト：  
非対象のアキュムレータの丸めたものを 1.15 形式の値として W13 に書き込みます。
2. [W13]+=2、後置インクリメント付きレジスタ間接：  
非対象のアキュムレータの丸めたものを 1.15 形式の値として指定した W13 で指定されたアドレスに書き込みます。その後 W13 は +2 されます。

### 2.6.4 丸め論理

丸め論理は、アキュムレータ書き込み(格納)のときに従来型(偏向あり)または収束型(偏向なし)丸め機能を実行することができます。どちらの丸めモードか RND (CORCON<1>) ビットの状態で決定されます。そしてデータメモリに書き込むとき、飽和論理回路を通過して、16 ビットの 1.15 形式のデータとなります。命令で丸めが指示されていない場合、切り捨てによる 1.15 形式のデータが格納されます。

図 2-11 は 2 つの丸めモードを示しています。従来型丸めはアキュムレータのビット 15 を取得してそれをゼロ拡張し、ガードまたはオーバーフロービット(ビット 16 から 31)を除いて MSWord に追加します。アキュムレータの LSWord が 0x8000 と 0xFFFF の間の場合(0x8000 を含む)、MSWord は増大し、アキュムレータの LSWord が 0x0000 と 0x7FFF の間の場合、MSWord は増大しません。このアルゴリズムの結果は、一連の任意丸めオペレーションの間に値がわずかに正に偏向するということになります。

LSWord が 0x8000 のとき以外は収束型(不偏)丸めは従来型丸めと同じように機能します。この場合、MSWord の LS ビット(アキュムレータのビット 16)をチェックして、それが ‘1’ の場合、MSWord を +1 し、‘0’ の場合、MSWord は変更しません。ビット 16 が本質的にランダムであるとすれば、この方式により蓄積する丸め偏向はすべて除去されます。

SAC および SAC.R 命令は対象のアキュムレータのコンテンツの切り捨て(SAC)または丸め(SAC.R)た結果を X-bus を通してデータメモリに格納します(飽和制御への入力となる。セクション 2.6.3.3「データメモリ書き込み飽和」を参照してください)。

命令の MAC クラスに関しては、アキュムレータ書き出しによるデータは常に丸められることに注意してください。

図 2-11: 従来型および収束型丸めモード

従来型(偏向あり)	収束型(偏向なし)																
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">16 15</td><td style="text-align: center;">0</td></tr> <tr> <td>MSWord</td><td>1XXXX XXXXX XXXXX XXXXX</td></tr> </table> <p>Round Up (add 1 to MSWord) when: LSWord &gt;= 0x8000</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">16 15</td><td style="text-align: center;">0</td></tr> <tr> <td>MSWord</td><td>0XXXX XXXXX XXXXX XXXXX</td></tr> </table> <p>Round Down (add nothing) when: LSWord &lt; 0x8000</p>	16 15	0	MSWord	1XXXX XXXXX XXXXX XXXXX	16 15	0	MSWord	0XXXX XXXXX XXXXX XXXXX	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">16 15</td><td style="text-align: center;">0</td></tr> <tr> <td>MSWord</td><td>11000 00000 00000 0000</td></tr> </table> <p>Round Up (add 1 to MSWord) when: 1. LSWord = 0x8000 and bit 16 = 1 2. LSWord &gt; 0x8000</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">16 15</td><td style="text-align: center;">0</td></tr> <tr> <td>MSWord</td><td>01000 00000 00000 0000</td></tr> </table> <p>Round Down (add nothing) when: 1. LSWord = 0x8000 and bit 16 = 0 2. LSWord &lt; 0x8000</p>	16 15	0	MSWord	11000 00000 00000 0000	16 15	0	MSWord	01000 00000 00000 0000
16 15	0																
MSWord	1XXXX XXXXX XXXXX XXXXX																
16 15	0																
MSWord	0XXXX XXXXX XXXXX XXXXX																
16 15	0																
MSWord	11000 00000 00000 0000																
16 15	0																
MSWord	01000 00000 00000 0000																

## 2.6.5 バレルシフター

バレルシフターは単一サイクルで 16 ビット算術右シフト、あるいは 16 ビット左シフトまで実行することができます。マルチビット シフトの DSP 命令または MCU 命令はバレルシフターを使用することができます。

シフターには、シフトオペレーションの規模（ビット数）と方向の両方を決定するために符号付きバイナリ値を必要とします：

- 正の値はオペランドを右にシフトします
- 負の値はオペランドを左にシフトします
- ‘0’ の値はオペランドを変更しません

バレルシフターはアキュムレータの幅に対応するために 40 ビット幅です。DSP シフトオペレーションの結果は 40 ビットの出力で、MCU シフトオペレーションの結果は 16 ビットです。

バレルシフターを使用する命令の要約を表 2-6 に示します。

表 2-6: DSP エンジンバレルシフターを使用する命令

命令	説明
ASR	データ メモリの算術マルチビット右シフト
LSR	データ メモリの論理マルチビット右シフト
SL	データ メモリのマルチビット左シフト
SAC	DSP アキュムレータに任意シフトした結果を格納する
SFTAC	DSP アキュムレータをシフトする

## 2.6.6 DSP エンジンモード選択

前のサブセクションで述べた DSP エンジンの様々な操作機能が CPU コア設定レジスタ (CORCON) で選択できます。以下の機能があります：

- 固定小数または整数乗算オペレーション。
- 従来型または収束型丸め。
- ACCA の自動飽和制御 on/off。
- ACCB の自動飽和制御 on/off。
- データメモリの書き込みの自動飽和制御 on/off。
- アキュムレータ飽和モード選択。

## 2.6.7 DSP エンジントラップイベント

DSP エンジンの例外の処理のために生成することができる様々な算術エラートラップは割り込み制御レジスタ (INTCON1) を通して選択します。これらは以下の通りです：

- OVATE (INTCON1<10>) を使用して ACCA オーバーフローのトラップの有効化。
- OVBTE (INTCON1<9>) を使用して ACCB オーバーフローのトラップの有効化。
- COVTE (INTCON1<8>) を使用して壊滅的 ACCA および／または ACCB オーバーフローのトラップの有効化。

算術エラートラップは、ユーザーが SFTAC 命令を使用して最大許容範囲 (+/- 16 ビット) を超える値のシフトをしようとした場合にも生成されます。このトラップソースは無効化できません。命令の実行は完了しますが、シフトの結果は対象のアキュムレータには書き込まれません。

INTCON1 レジスタおよび算術エラートラップに関して詳しくは第 6 章 .「割り込み」を参照してください。

## 2.7 除算サポート

dsPIC30F は次の種類の割り算操作をサポートします。

- DIVF: 16/16 符号付き固定小数除算。
- DIV.SD: 32/16 符号付き除算。
- DIV.UD: 32/16 符号無しの除算。
- DIV.SW: 16/16 符号付き除算。
- DIV.UW: 16/16 符号無しの除算。

すべての除算結果の商は W0 に、そして余りは W1 に配置されます。16 ビット除数はいずれの W レジスタにも配置できます。16 ビット被除数の場合はいずれの W レジスタにも配置でき、32 ビット被除数は W レジスタの隣接ペアに配置する必要があります。

すべての除算命令は反復演算で、REPEAT ループ内で 18 回実行する必要があります。ユーザーが REPEAT 命令を記述する必要があります。完全な除算を実行するためには 19 命令サイクルが必要です。

除算実行ループは、他の REPEAT ループと同様、中断可能です。ループ反復毎に、すべてのデータがそれぞれのデータレジスタに返されるので、各 W レジスタを確実に ISR に保存する必要があります。ループ実行中に W レジスタにある中間値は、ハードウェアにとって重要ですが、ユーザーには無意味なデータです。有益な結果を得るために除算命令を REPEAT ループで 18 回実行する必要があります。

除算命令のプログラミング例と詳細情報については、“dsPIC30F プログラマリファレンスマニュアル”(DS70030) を参照してください。

## 2.8 命令フローの種類

dsPIC30F アーキテクチャのほとんどの命令は、プログラムメモリの 1 ワードで構成され、1 サイクルで実行されます。命令プリフェッチャニズムによって 1 サイクル (1Tcy) 実行が実現されています。ただし、2 つまたは 3 つの命令サイクルを必要とする命令もあります。その結果、dsPIC® アーキテクチャには 7 つの異なるタイプの命令フローがあります。そのフローは以下のとおりです。

### 1. 1 命令ワード、1 命令 サイクル：

図 2-12 の通り、これらの命令を実行するには命令サイクルが 1 つ必要です。ほとんどの命令は 1 ワード、1 サイクル命令です。

図 2-12: 命令フロー - 1- ワード、1 - サイクル

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOV #0x55AA,W0	フェッチ 1	実行 1				
2. MOV W0,PORTA		フェッチ 2	実行 2			
3. MOV W0,PORTB			フェッチ 3	実行 3		

### 2. 1 命令ワード、2 命令サイクル：

これらの命令には、プリフェッチャラッシュはありません。このタイプの命令は MOV.D 命令(ロードおよび格納ダブルワード)だけです。これらの命令を完了するには図 2-13 のとおり、2 つのサイクルが必要です。

図 2-13: 命令フロー - 1- ワード、2 - サイクル (MOV.D オペレーション)

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOV #0x1234,W0	フェッチ 1	実行 1				
2. MOV.D [W0++],W1		フェッチ 2	実行 2 R/W サイクル			
3. MOV #0x00AA,W1			フェッチ 3	実行 2 R/W サイクル		
4. MOV #0x00CC,W0				No Fetch	実行 3	
					フェッチ 4	実行 4

### 3. 1 命令ワード、2 または 3 命令サイクルプログラムフロー変更:

これらの命令には、相対呼び出しおよび分歧命令、スキップ命令が含まれます。命令が PC を変更すると(それを増分する場合以外は)、先読みしたプログラムメモリのデータは破棄しなければなりません。このため、命令を実行するためには図 2-14 の通り 2 つの有効なサイクルが必要です。

図 2-14: 命令フロー - 1- ワード、2 - サイクル(プログラムフロー変更)

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOV.B #0x55,W0	フェッチ 1	実行 1				
2. BTSC PORTA,#3		フェッチ 2	実行 2 Skip Taken			
3. ADD.B PORTA	(executed as FNOP)		フェッチ 3	強制 NOP		
4. BRA SUB_1				フェッチ 4	実行 4	
5. ADD.B PORTB	(executed as FNOP)				フェッチ 5	強制 NOP
6. SUB_1: Instruction @ address SUB_1						フェッチ SUB_1

2 ワード命令がスキップされると 3 つのサイクルが実行されます。この場合、先読みデータは破棄され、2 ワード命令の 2 番目のワードがフェッチされます。図 2-15 の通り、命令の 2 番目のデータは NOP として実行されます。

図 2-15: 命令フロー - 1- ワード、3- サイクル(2- ワード命令スキップ)

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. BTSC SR,#Z	フェッチ 1	実行 1, Skip Taken				
2. GOTO LABEL (GOTO 2nd word)		フェッチ 2	強制 NOP			
			Fetch 2nd word of GOTO	2nd word executed as a NOP		
3. BCLR PORTB,#3				フェッチ 3	実行 3	
4. MOV W0,W1					フェッチ 4	実行 4

### 4. 1 命令ワード、3 命令 サイクル (RETFIE, RETURN, RETLW):

サブルーチン呼び出しありは割り込みサービスルーチンから戻るために使用される RETFIE, RETURN および RETLW 命令は、命令を実行するためには図 2-16 の通り 3 つの有効なサイクルが必要です。

図 2-16: 命令フロー - 1- ワード、3 - サイクル (RETURN, RETFIE, RETLW)

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOV #0x55AA,W0	フェッチ 1	実行 1				
2. RETURN		フェッチ 2	実行 2			
3. (instruction in old program flow)			フェッチ 3	実行 2		
4. MOV W0, W3 (instruction in new program flow)				No Fetch	実行 2	
5. MOV W3, W5					フェッチ 4	実行 4
						フェッチ 5

### 5. テーブル読み取り・書き込み命令:

これらの命令はフェッチを一時停止して読み取りまたは書き込みサイクルをプログラムメモリに挿入します。テーブルオペレーション実行時にフェッチした命令は1サイクルの間保存され、図 2-17 に示されたテーブルオペレーションの直後のサイクルで実行されます。

図 2-17: 命令パイプラインフロー - テーブルオペレーション

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOV #0x1234,W0	フェッチ 1	実行 1				
2. TBLRDL.w [W0++],W1		フェッチ 2	実行 2			
3. MOV #0x00AA,W1			フェッチ 3	PM データ 読み込みサイ クル		
4. MOV #0x00CC,W0				バス読み込み	実行 3	
					フェッチ 4	実行 4

### 6. 2命令ワード、2命令サイクル:

これらの命令では、命令のあとにデータを含みます。その結果、図 2-18 に示されているように 2 命令サイクルとなります。CPU が 2 つのワード命令の一番目のワードを先にフェッチしないで 2 番目のワードをフェッチした場合、2 番目のワードは NOP として実行されるようにエンコードされています。これは 2 ワード命令がスキップ命令でスキップされた際に重要です。(図 2-15 参照)

図 2-18: 命令パイプラインフロー - 2- ワード、2 - サイクル

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOV #0xAA55,W0	フェッチ 1	実行 1				
2. GOTO LABEL		フェッチ 2L	更新 PC			
			フェッチ 2H	強制 NOP		
3. LABEL: MOV W0,W2				フェッチ 3	実行 3	
4. BSET PORTA, #3					フェッチ 4	実行 4

### 7. アドレスレジスタの依存関係:

ある命令では X データメモリ読み取りと書き込み操作間のデータアドレス依存のためストールすることになってしまいます。このリソース競合の解決のため増設サイクルが挿入されています。セクション 2.10 「アドレスレジスタの依存関係」を参照してください。

図 2-19: 命令パイプラインフロー - 1- ワード、1- サイクル(命令中止も含む)

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOV W0,W1	フェッチ 1	実行 1				
2. MOV [W1],[W4]		フェッチ 2	実行 1			
			中止	実行 2		
3. MOV W2,W1				フェッチ 3	実行 3	

## 2.9 ループ構造

dsPIC30F は無条件の自動プログラムループ制御を提供するため REPEAT と DO 両方の命令をサポートします。REPEAT 命令は単一命令プログラムループを実行するため使用され、DO 命令は複数の命令プログラムループを実行するため使用されます。両方の命令は CPU ステータスレジスタ SR 内のコントロールビットを使用し、CPU 操作を一時的に変更します。

### 2.9.1 REPEAT ループコンストラクト

REPEAT 命令はそれに続く命令を反復実行させます。命令に含まれているリテラル値または W レジスタ値の 1 つを用いて、反復カウント値を指定できます。W レジスタオプションを使用すると、ループ回数がソフトウェア変数になります。

REPEAT ループにある命令は 1 回以上実行されます。REPEAT ループ用反復回数は 14 ビットリテラル値 + 1§ または Wn + 1 です。

二種類の REPEAT 命令の構文は以下のとおりです。

```
REPEAT #lit14      ; RCOUNT <- lit14  
(Valid target Instruction)
```

または

```
REPEAT Wn          ; RCOUNT <- Wn  
(Valid target Instruction)
```

#### 2.9.1.1 REPEAT 操作

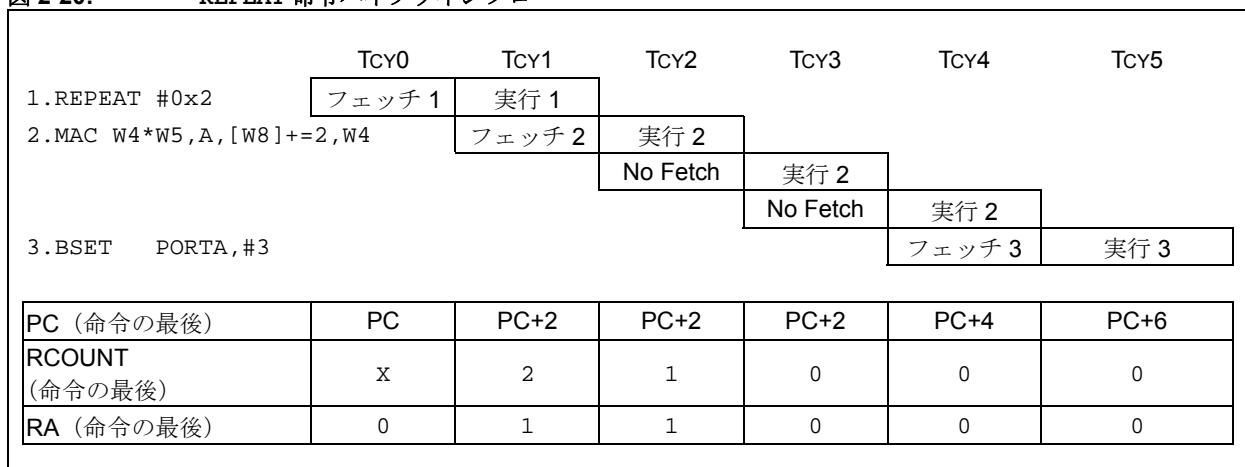
REPEAT 操作用ループ回数はメモリマップされた 14 ビット RCOUNT レジスタに保存されています。RCOUNT は REPEAT 命令で初期化されます。RCOUNT 値がゼロでなければ、REPEAT 命令は Repeat Active、または RA(SR<4>) ステータスピットを ‘1’ にセットします。

RA は読み取り専用で、ソフトウェアによって修正できません。反復ループカウンタの値が 0 より大きい間は、PC は増分されません。つまり PC 増分は RCOUNT=0 まで禁止されています。反復ループの命令フロー例は、図 2-20 をご覧ください。

ループカウント値が 0 のときは、REPEAT は NOP の効果を持ち、RA (SR<4>) ビットはセットされません。REPEAT ループは本来開始前に無効化されるため、対象命令が次の命令をプリフェッチする間に一度だけ実行されます。(すなわち、通常実行フロー)

**注:** REPEAT 命令の直後の命令（すなわち、対象命令）は常に少なくとも 1 回実行されます。この命令はいつも 14 ビットリテラルまたは W レジスタオペランドに指定されている値より 1 回多く実行されます。

図 2-20: REPEAT 命令パイプラインフロー



### 2.9.1.2 REPEAT ループを中の割り込み

REPEAT ループはいつでも割り込むことができます。

**RA** 状態はスタック上に保存されるため、例外プロセシングの時ユーザーは、ネスト化した中断（回数は問いません）からさらに別のループを実行し続けることができます。SRL をスタックした後、ISR 実行中に正常なループで通常命令が実行ができるようにするため RA ステータスビットはクリアされます。

**注：** REPEAT ループが中断され、ISR が進行中である場合、ISR 内で他の REPEAT 命令を実行する前に、RCOUNT (REPEAT カウントレジスタ) をスタックへ待避する必要があります。

**注：** REPEAT が ISR 内で使用された場合、RETFIE を実行する前に RCOUNT をスタックから復旧 (unstack) する必要があります。

RETFIE を使用して ISR から REPEAT ループへ復帰するのに特別な処理は必要ありません。割り込みは RETFIE 命令の第 3 サイクルの間に繰り返し命令をあらかじめフェッチします。SRL レジスタを POP すれば待避していた RA ビットが復旧し、設定していれば、中断された REPEAT ループが再開されます。

**注：** 繰り返し命令 (REPEAT ループ内の対象の命令) が PSV を使用して PS データにアクセスしている場合、例外から復帰後最初に実行されるときには 2 命令サイクルが必要です。ループの最初の繰り返しと同様、1 命令サイクルの間には時間制限のために最初の命令は PS にあるデータにはアクセスできません。

#### 2.9.1.2.1 REPEAT ループの早期終了

割り込まれた REPEAT ループは、ISR 処理内でソフトウェアにより RCOUNT レジスタを消去すれば通常よりも早く終了させることができます。

#### 2.9.1.3 REPEAT 命令に関する制限

以下の場合を除いて、いずれの命令も即座に REPEAT 命令直下に続くことができます：

1. プログラムフロー制御命令(分岐、比較およびスキップ、サブルーチン呼び出し、復帰等)。
2. 別の REPEAT または DO 命令。
3. DISI, ULINK, LNK, PWRSAV, RESET。
4. MOV.D 命令。

**注：** REPEAT ループ内で実行可能な命令および／または命令アドレスモードもありますが、反復してもほとんど意味がないものがあります。

## 2.9.2 DO ループ構造

DO 命令はオーバーヘッド無しで指定回数だけ繰り返す命令グループを構成することができます。エンドアドレスの命令を含む一連の命令が繰り返されます。DO 命令の繰り返しカウント値は 14 ビットリテラル値または命令内で宣言されている W レジスタの内容で指定できます。2 種類の DO 命令の構文は以下の通りです：

```
DO      #lit14,LOOP_END      ; DCOUNT <-- lit14
Instruction1
Instruction2
:
:
LOOP_END: Instruction n

DO      Wn,LOOP_END      ; DCOUNT <-- Wn<13:0>
Instruction1
Instruction2
:
:
LOOP_END: Instruction n
```

DO ループ構造には以下の特徴があります：

- W レジスタをループカウントの指定に使用できます。これにより、実行中にループ回数を指定することができます。
- 繰り返す命令は順番に並んでいなくてもよい(例. 分岐、サブルーチン呼び出し等も可能です。)。
- ループエンドアドレスは開始アドレスよりも大きい値である必要はありません。

### 2.9.2.1 DO ループ レジスタおよびオペレーション

DO ループが実行する反復回数は (14 ビットリテラル値 +1) または (Wn 値 + 1) です。W レジスタを使用して反復回数を指定する場合、ループカウントを指定するために W レジスタの上位 2 ビットは使用しません。ループ内の命令は常に最低 1 度は実行されるため、DO ループのオペレーションは C プログラム言語の ‘do-while’ 構造と類似しています。

dsPIC30F は DO ループに関連する 3 つのレジスタをしています：DOSTART、DOEND および DCOUNT です。これらのレジスタはメモリにマップされていて DO 命令が実行されるとハードウェアによって自動的にロードされます。DOSTART は DO ループの開始アドレス指定用で、DOEND は DO ループの終了アドレス指定用です。DCOUNT レジスタはループが実行する反復の回数指定用です。DOSTART および DOEND は 22 ビットレジスタで、PC 値となります。これらのレジスタの MS ビットおよび LS ビットは ‘0’ に固定されています。詳しくは図 2-2 を参照してください。PC<0> は常に強制的に ‘0’ であるため、LS ビットはこれらのレジスタには格納されません。

DA ステータスビット (SR<9>) は、単一の DO ループ ( またはネスト化した DO ループ ) がアクティブであることを示します。DO 命令が実行されれば DA ビットがセットされ、その後の各命令サイクルにおける PC アドレスと DOEND レジスタの比較を可能にします。PC が DOEND の値と一致すると、DCOUNT はデクリメントされます。DCOUNT レジスタが 0 でない場合、PC には DO ループの反復を再び開始するため DOSTART レジスタの値がロードされます。

DO ループは DCOUNT = 0 で終了します。もし他に継続中のネスト化した DO ループがない場合、DA ビットがクリアされます。

**注：** DO ループ構造内の命令グループは常に最低 1 度は実行されます。DO ループは常にリテラルまたは W レジスタオペランドに指定された値よりも 1 回多く実行されます。

### 2.9.2.2 DO ループネスティング

DOSTART、DOEND および DCOUNT レジスタには、DO ループハードウェアが 1 つのレベルの自動ネスティングをサポートするようにそれぞれに対応するするシャドーレジスタあります。DOSTART、DOEND および DCOUNT レジスタはユーザーがアクセス可能で、必要な場合に手動で保存して追加のネスティングを可能にすることができます。

DO レベルビット、DL<2:0> (CORCON<10:8>) は現在実行されている DO ループのネスティングレベルを示します。最初の DO 命令が実行されると、DL<2:0> はレベル 1 の DO ループが実行中であることを示す B ‘001’ に設定されます。DA (SR<9>) も設定されます。最初の DO ループ内で別な DO 命令が実行されると、DOSTART、DOEND および DCOUNT レジスタは新しいループ値に更新される前にシャドーレジスタに移動されます。DL<2:0> ビットは 2 番目のネスト化した DO ループが実行中であることを示す B ‘010’ に設定されます。DA (SR<9>) ビットも設定されたままです。

アプリケーションで 1 つ以上のレベルの DO ループネスティングが要求されていない場合、特別な注意は必要ありません。万一ユーザーが 1 つ以上のレベルの DO ループネスティングを必要とした場合、次の DO 命令を実行する前に手動で DOSTART、DOEND および DCOUNT レジスタを保存することで可能になります。DL<2:0> が B ‘010’ 以上の場合にはこれらのレジスタを常に保存する必要があります。

DOSTART、DOEND および DCOUNT レジスタは DL<2:0> = B ‘010’ の時に、DO ループが終了したときには、シャドーレジスタから自動的に復元されます。

**注：** DL<2:0> (CORCON<10:8>) ビットは一体化されて（論理 OR）、DA(SR<9>) ビットを形成します。ネスト化した DO ループが実行されている場合、一番外側のループに関連するループカウントが消滅すると DA ビットがクリアされます。

### 2.9.2.3 DO ループの割り込み

DO ループはいつでも割り込むことができます。ISR の間にほかの DO ループが実行される場合、ユーザーは DL<2:0> ステータスピットをチェックし、必要に応じて DOSTART、DOEND および DCOUNT レジスタを保存する必要があります。

単一レベルの DO ループのみが以下で実行されることが保証できる場合、特別な処理は必要ありません：

- バックグラウンドおよびいずれか 1 つの ISR ハンドラ（割り込みネスティングが有効の場合）または
- バックグラウンドおよびすべての ISR（割り込みネスティングが無効の場合）

あるいは、バックグラウンドまたは以下のいずれかで 2 つ以下の（ネスト化した）DO ループを実行できます

- 1 つの ISR ハンドラ（割り込みネスティングが有効の場合）または
- いずれかの ISR（割り込みネスティングが無効の場合）

いずれのトラップハンドラ内でも DO ループは使用されないことを前提とします。

RETFILE 命令を使用して ISR から DO ループへ復帰する場合特別な処理は必要ありません。

### 2.9.2.4 DO ループの早期終了

DO ループを早期に終了する方法は 2 つあります：

1. EDT(CORCON<11>) ビットは、すべてのループを完了する前に DO ループを終了する方法をユーザーに提供します。EDT ビットに ‘1’ を書き込むとループは強制的に反復を途中で中断し、終了します。ループの最後または最後から 2 番目の命令の間に EDT がセットされると、もう 1 回ループが実行されます。EDT の読み出しへ常に ‘0’ です；それらをクリアしても無効です。EDT ビットのセット後、ユーザーは任意的に DO ループから分岐することができます。
2. 他にも、コードは最後の命令以外からならいつでもループから分岐することができます。最後の命令はフロー制御命令は使えません。DA ビットは DO ループハードウェアを有効にしますが、命令プレフェッチの間に最後から 2 番目の命令のアドレスに遭遇しない限り効果はありません。この早期終了方法は、DO ループの終了方法として推奨されません。

**注：** EDT を使用せずに DO ループから抜け出ることは推奨されません。ハードウェアは引き続き DOEND アドレスをチェックし続けるためです。

## 2.9.2.5 DO ループ制限

DO ループには以下の制限があります。

- ループ内の最後の命令選択
- ループ長（最初の命令からのオフセット）
- DOEND レジスタの読み取り

ループ終了テストが最後から 2 番目の命令で行われますので、すべての DO ループは少なくとも 2 つの命令を含む必要があります。1 つの命令ループのときには REPEAT 命令を使用します。

特別な機能レジスタである DOEND は、DO 命令または DOEND SFR へのファイルレジスタ書き込み操作の直後の命令においてはユーザーソフトウェアで読み取れません。

### 2.9.2.5.1 最後の命令制限

DO ループで実行される最後の命令には制限があります。以下の命令を DO ループにおける最後の命令とすることはできません。

1. フロー制御命令（例えば、ブランチ、比較とスキップ、GOTO, CALL, RCALL, TRAP のいずれも）。
2. RETURN、RETFIE と RETLW は DO ループの最後の命令として正確に機能しますが、ループを完了するために確実にループに戻る必要があります。
3. 他の REPEAT または DO 命令。
4. REPEAT ループ内の対象命令。すなわち、最後から 2 番目の命令も REPEAT にできません。
5. プログラムメモリの 2 ワードを占める命令。

### 2.9.2.5.2 ループ長制限

ループ長は DO ループで最初の命令から最後の命令の符号付きのオフセットとして定義されます。ループ長とループ内の最初の命令のアドレスをあわせて、ループの最後の命令のアドレスが形成されます。以下は、避けるべきループ長値の数例です。

#### 1. ループ長 = -2

実行はループの最初の命令（例えば `#PC` とする）で開始し、ループエンドアドレス（例の場合「PC-4」）がプリフェッчされるまで続きます。これは DO 命令の最初のワードなので、DO 命令をもう一度実行し、DCOUNT を再度初期化し、「PC」をプリフェッチします。この処理はループエンドアドレス「PC - 4」がプリフェッチされ続ける限り永久に続きます。`n` のこの値は無限ループを作成する可能性があります（Watchdog タイマーリセットがない条件）。

```
end_loop: DO #33, end_loop ;DO is a two-word instruction
           NOP          ;2nd word of DO executes as a NOP
           ADD W2,W3,W4  ;First instruction in DO loop([PC])
```

#### 2. ループ長 = -1

実行はループの最初の命令（例えば `#PC` とする）で開始し、ループエンドアドレス（この場合「PC-2」）がプリフェッчされるまで続きます。ループエンドアドレスは DO 命令の二番目のワード単語なので、このループは NOP として実行しますが「PC」のプリフェッチも行います。従ってループは再度実行されます。この処理はループエンドアドレス「PC-2」がプリフェッチされ続ける限り続き、ループは終了しません。DCOUNT レジスタの値がゼロに達し、後続の減算命令でボローが発生するとループは終了します。ただしそのような場合、ループから出るときの最初の命令は再び最初のループ命令となります。

```
DO #33, end_loop ;DO is a two-word instruction
end_loop: NOP      ;2nd word of DO executes as a NOP
           ADD W2,W3,W4  ;First instruction in DO loop([PC])
```

### 3. ループ長 = 0

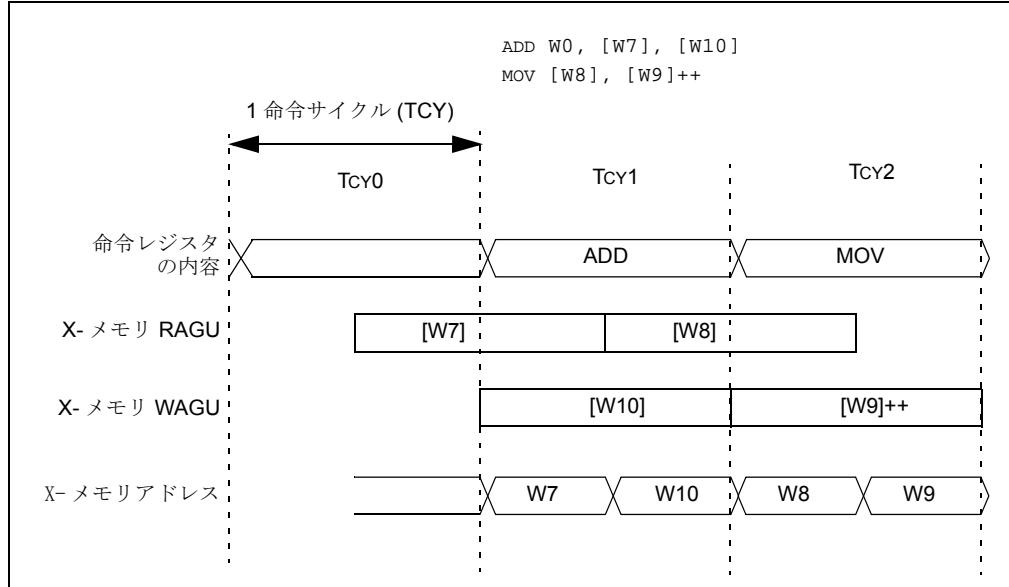
実行はループの最初の命令で開始し(例.[PC])、ループエンドアドレス([PC])がプレフェッチされた時点で終了します。もしループが継続する場合、このプレフェッチにより DO ループハードウェアは次のフェッチ(再度 [PC])のために DOEND アドレス([PC])を PC にロードします。ループの最初の反復の後、ループの最初の命令がループカウントがアンダーフローしてループが終了するまで繰り返し実行されます。アンダーフローが発生すると、ループから出るときの最初の命令は [PC] の後の命令となります。

```
DO #33, end_loop ;DO is a two-word instruction
NOP           ;2nd word of DO executes as a NOP
end_loop: ADD W2,W3,W4      ;First instruction in DO loop([PC])
```

## 2.10 アドレスレジスタの依存関係

dsPIC30F アーキテクチャはほとんどの MCU クラス命令用のデータメモリ読み取り(元)とデータメモリ書き込み(先)をサポートします。AGU が連結するデータメモリ読み取りまたは書き込み命令実行に必要な有効アドレス(EA)計算を完了するには、それぞれ 1 命令サイクルずつかかります。このタイミングのため、図 2-21 で示されるように、各命令用のデータメモリ読み書き操作が部分的に重複してしまいます。この重複のため、「書き込み後読み取り」(RAW) データ依存が命令境界をまたいで発生する可能性があります。RAW データ依存は dsPIC30FCPU によって実行時に検出および処理されます。

図 2-21: データメモリアクセスタイミング



### 2.10.1 「書き込み後読み取り」依存ルール

現在の命令で W レジスタが書き込み操作先として使用され、プリフェッチした次の命令で読み取られる W レジスタが同じ場合、以下のルールが適用されます。

1. 書き込み先(現在の命令)が Wn の内容を変更しない場合、ストールは発生しません。  
または
2. 読み取り元(プリフェッチした命令)が Wn を使用して EA を計算しない場合、ストールは発生しません。

各命令サイクルの間に、dsPIC30F ハードウェアは RAW データ依存が発生しようとしているか自動的に確認します。上記の条件が満たされない場合、CPU はプリフェッチした命令を実行する前に一命令サイクル分自動的に遅らせます。命令のストールにより、次の(プリフェッチした)命令が書き込みデータを使用する前に、W レジスタ書き込みが行われる時間が与えられます。

表 2-7: 呼び出し後書き込み依存状態のサマリー

$W_n$ を使用するアドレスシングルモードの送り先	$W_n$ を使用するアドレスシングルモードの元	ステータス	例 ( $W_n = W2$ )
直接	直接	適用される	ADD.w W0, W1, W2 MOV.w W2, W3
直接	間接	ストール	ADD.w W0, W1, W2 MOV.w [W2], W3
直接	Indirect with modification	ストール	ADD.w W0, W1, W2 MOV.w [W2++], W3
間接	直接	適用される	ADD.w W0, W1, [W2] MOV.w W2, W3
間接	間接	適用される	ADD.w W0, W1, [W2] MOV.w [W2], W3
間接	修正付き間接	適用される	ADD.w W0, W1, [W2] MOV.w [W2++], W3
修正付き間接	直接	適用される	ADD.w W0, W1, [W2++] MOV.w W2, W3
間接	間接	ストール	ADD.w W0, W1, [W2] MOV.w [W2], W3 ; W2=0x0004 (mapped W2)
間接	修正付き間接	ストール	ADD.w W0, W1, [W2] MOV.w [W2++], W3 ; W2=0x0004 (mapped W2)
修正付き間接	間接	ストール	ADD.w W0, W1, [W2++] MOV.w [W2], W3
修正付き間接	修正付き間接	ストール	ADD.w W0, W1, [W2++] MOV.w [W2++], W3

### 2.10.2 命令ストールサイクル

命令ストールは基本的には、次の読み出しオペレーションの前に先の書き込みが完了する時間をとるために命令の読み出しフェーズの前に付加された 1 命令サイクルの待ち時間です。割り込み待ち時間の目的においては、ストールサイクルは検出された場所の命令に従う命令と関連していることに注意する必要があります（例：ストールサイクルは常に命令実行サイクルに先行します）。

RAW データ依存が検出されると、dsPIC30F は命令ストールを開始します。命令ストールの間に、以下のイベントが発生します。

1. 進行中の書き込み操作（先行命令による）は正常に完了します。
2. データ空間は命令ストール後まで指示されません。
3. PC 増分は命令ストール後まで禁止されます。
4. 更なる命令フェッチは命令ストール後まで禁止されます。

### 2.10.2.1 命令ストールサイクルと割り込み

割り込みが命令ストールを導く2つの隣接命令と同時発生する場合、発生しうる結果は以下の2つのうちいいずれかです。

1. 割り込みが最初の命令と同時に発生した場合。このような状況では、最初の命令は完了し、第2の命令はISRが完了した後に実行されます。この場合、例外処理の与える時間で最初の命令の書き込みフェーズが完了するため、ストールサイクルは第2の命令から消去されます。
2. 割り込みが第2の命令と同時に発生した場合。このような状況では、第2の命令と追加のストールサイクルが、ISRの前に実行されます。この場合、第2の命令に関連するストールサイクルは正常に実行されます。しかし、ストールサイクルは例外処理タイミングに効果的に吸収されます。例外処理は、通常2命令サイクルが割り込み発生の前に挿入されるためです。

### 2.10.2.2 命令ストールサイクルとフロー変更命令

CALLとRCALL命令はW15を使用してスタックに書き込みを行うため、次の命令のソース読み取りがW15を使用する場合、それに先立って命令ストールが強制挿入されます。

RETFIEとRETURN命令は読み取り操作のみ行うので、次の命令に先立って命令ストールが強制挿入されることはありません。しかし、RETLW命令は最後のサイクル中にWレジスタに書き込みを行うので、ストールが強制挿入されます。

GOTOとブランチ命令は書き込み操作を行わないため、命令ストールが挿入されることはありません。

### 2.10.2.3 命令ストールとDOとREPEATループ

命令ストールサイクルの追加以外、RAWデータ依存はDO操作やREPEAT操作に影響しません。

REPEATループ内でプリフェッヂした命令は、ループを完了するか例外が発生するまで変更されません。レジスタ依存チェックは複数の命令の境界で行われますが、dsPIC30FはREPEATループの際、同一命令の操作元と操作先を比較することで効果的に行われます。

DOループの最後の命令は、ループスタートアドレスにある命令か、次の命令（ループ外部の）をプリフェッヂします。命令ストールはループでの最後の命令とプリフェッヂした内容に基いて決定されます。

### 2.10.2.4 命令ストールとプログラムスペースビジビリティ (PSV)

プログラムスペース(PS)がPSVCORCON<2>ビットの有効化によってデータ空間にマップされ、XスペースEAがビジュアルプログラムスペースウィンドウの範囲内に入る場合、読み取りまたは書き込みサイクルはプログラム空間内のアドレスにリダイレクトされます。プログラム空間からデータにアクセスするには2つの命令サイクルを要します。

PSVアドレス空間で操作する命令は、他の命令と同様、RAWデータ依存とそれに続いて発生する命令ストールに従います。

以下のコードセグメントの場合を考えて下さい。

```
ADD W0,[W1],[W2++] ; PSV = 1, W1=0x8000, PSVPAG=0xAA
MOV [W2],[W3]
```

この命令シーケンスを実行するには4つ命令サイクルが必要です。W1経由のPSVアクセスのため、1つの命令サイクルが追加されます。さらに、W2によるRAWデータ依存を解決するためにもう一つの命令ストールサイクルが挿入されます。

## 2.11 レジスタマップ

dsPIC30F CPU コア関連のレジスタの概要は表 2-8 を参照してください。

表 2-8: dsPIC30F コアレジスタマップ

名	Addr	ビット 15	ビット 14	ビット 13	ビット 12	ビット 11	ビット 10	ビット 9	ビット 8	ビット 7	ビット 6	ビット 5	ビット 4	ビット 3	ビット 2	ビット 1	ビット 0	リセット状態
W0	0000																0000 0000 0000 0000	
W1	0002																0000 0000 0000 0000	
W2	0004																0000 0000 0000 0000	
W3	0006																0000 0000 0000 0000	
W4	0008																0000 0000 0000 0000	
W5	000A																0000 0000 0000 0000	
W6	000C																0000 0000 0000 0000	
W7	000E																0000 0000 0000 0000	
W8	0010																0000 0000 0000 0000	
W9	0012																0000 0000 0000 0000	
W10	0014																0000 0000 0000 0000	
W11	0016																0000 0000 0000 0000	
W12	0018																0000 0000 0000 0000	
W13	001A																0000 0000 0000 0000	
W14	001C																0000 0000 0000 0000	
W15	001E																0000 0000 0000 0000	
SPLIM	0020																0000 0000 0000 0000	
ACCAL	0022																0000 0000 0000 0000	
ACCAH	0024																0000 0000 0000 0000	
ACCAU	0026																0000 0000 0000 0000	
ACCBL	0028																0000 0000 0000 0000	
ACCBH	002A																0000 0000 0000 0000	
ACCBU	002C																0000 0000 0000 0000	
PCL	002E																0000 0000 0000 0000	
PCH	0030	—	—	—	—	—	—	—	—	—	—	—	—	—	PCH	0	0000 0000 0000 0000	
TBLPAG	0032	—	—	—	—	—	—	—	—	—	—	—	—	—	TBLPAG		0000 0000 0000 0000	
PSVPAG	0034	—	—	—	—	—	—	—	—	—	—	—	—	—	PSVPAG		0000 0000 0000 0000	
RCOUNT	0036																xxxx xxxx xxxx xxxx	
DCOUNT	0038																xxxx xxxx xxxx xxxx	
DOSTARTL	003A															0	xxxx xxxx xxxx xxxx0	
DOSTARTH	003C	—	—	—	—	—	—	—	—	—	—	—	—	—	DOSTARTH		0000 0000 00xx xxxx	
DOENDL	003E															0	xxxx xxxx xxxx xxxx0	
DOENDH	0040	—	—	—	—	—	—	—	—	—	—	—	—	—	DOENDH		0000 0000 00xx xxxx	

表 2-8: dsPIC30F コアレジスタマップ (続き)

名	Addr	ビット 15	ビット 14	ビット 13	ビット 12	ビット 11	ビット 10	ビット 9	ビット 8	ビット 7	ビット 6	ビット 5	ビット 4	ビット 3	ビット 2	ビット 1	ビット 0	リセット状態
SR	0042	OA	OB	SA	SB	OAB	SAB	DA	DC	IPL2	IPL1	IPL0	RA	N	OV	Z	C	0000 0000 0000 0000 0000
CORCON	0044	—	—	—	—	US	EDT	DL<1:0>	SATA	SATB	SATDW	ACCSAT	IPL3	PSV	RND	IF	0000 0000 0010 0000	
MODCON	0046	XMODEN	YMODEN	—	—	BWM<3:0>		YWM<3:0>		XWM<3:0>							0000 0000 0000 0000	
XMODSRT	0048							XMODSRT<15:0>								0	xxxx xxxx xxxx xxxx xx0	
XMODEND	004A							XMODEND<15:0>								1	xxxx xxxx xxxx xxxx xx1	
YMODSRT	004C							YMODSRT<15:0>								0	xxxx xxxx xxxx xxxx xx0	
YMODEND	004E							YMODEND<15:0>								1	xxxx xxxx xxxx xxxx xx1	
XBREV	0050	BREN						XBREV<14:0>									xxxx xxxx xxxx xxxx	
DISICNT	0052	—	—					DISICNT<13:0>									0000 0000 0000 0000	
予約	0054 - 007E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000	

凡例: x = 不定

注: 個々のコアレジスタマップの詳細はデバイスデータシートを参照してください。

## 2.12 関連するアプリケーションノート

この項では、マニュアルのこの章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正して使用可能です。制限がある場合もあります。現状、dsPIC30F CPU モジュールに関連する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 #
現在のところ、関連するアプリケーションノートはありません。	

**注：** dsPIC30F ファミリのデバイスに関しての、その他のアプリケーションノートやコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 2.13 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

この版は、dsPIC30FCPU モジュールに関する追加の技術情報を含みます。

注意：



# MICROCHIP

## 第3章. データメモリ

### ハイライト

この章は、以下の項目を含んでいます。

3.1	はじめに.....	3-2
3.2	データ空間アドレス生成ユニット (AGUs) .....	3-5
3.3	モジュロアドレッシング.....	3-7
3.4	ビット反転アドレッシング.....	3-14
3.5	コントロールレジスタ記述.....	3-18
3.6	関連するアプリケーションノート.....	3-23
3.7	改訂履歴.....	3-24

## 3.1 はじめに

dsPIC30F のデータ幅は 16 ビットです。すべての内部レジスタとデータメモリは 16 ビット幅で構成されます。dsPIC30F は 2 つのデータ空間を持つことを特徴としています。そのデータ空間は、(いくつかの DSP 命令に関しては) 別々にアクセスされるか、または (MCU 命令に関しては) 1 つの 64K バイトのリニアなアドレス範囲として、一緒にアクセスされます。データ空間は、2 つのアドレス生成ユニット (AGUs) と別々のデータバスを使用することでアクセスされます。

データ空間メモリマップの例を図 3-1 に示します。

0x0000 と 0x07FF の間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 (SFRs) 用にリザーブされています。SFRs には、CPU やデバイス内蔵周辺のためのコントロールビットやステータスピットも含みます。

RAM はアドレス 0x0800 から開始し、2 つのブロック、X と Y データ空間に分割されています。データを書き込む時には、X と Y データ空間は、常に 1 つのリニアなデータ空間としてアクセスされます。データを読み出す時には、X と Y のメモリ空間は独立してアクセスされるか、または 1 つのリニアな空間としてアクセスされます。MCU クラス命令のデータ読み出しへは、X と Y データ空間を、常に 1 つの結合されたデータ空間としてアクセスします。MAC 命令等の 2 つのソースオペランドを持つ DSP では、2 つのソースオペランドを同時に読むことをサポートするために、X と Y データ空間を別々にアクセスします。

MCU 命令では、データ読み出しほりは書き込み動作用のアドレスポインタとして、どの W レジスタも使用することができます。

データ読み出しの間は、DSP クラス命令は、Y アドレス空間を総合データ空間から切り離します。W10 と W11 は、Y データ空間からの読み出し用アドレスポインタとして使用されます。残りのデータ空間は、X 空間として参照されますが、より詳細には “X マイナス Y” 空間として記述されます。W8 と W9 は、DSP クラス命令において、X データ空間からのデータ読み出し用アドレスポインタとして使用されます。

図 3-2 には、MCU クラスと DSP クラス両方の命令用として、データメモリマップがどのように機能するかを示しています。アドレス空間がデータ読み出しにあたりどのようにアクセスされるかを決定するのは、W レジスタ番号と命令タイプであることに注意してください。特に、MCU 命令は X と Y メモリを 1 つに結合されたデータ空間として扱います。MCU は読み出し、書き込み用のアドレスポインタとして、どの W レジスタも使用できます。2 つのデータオペランドを同時にプリフェッチする DSP 命令は、データメモリを 2 つの空間に分けます。この場合には、読み出しあдресスポインタとして特定の W レジスタが使用される必要があります。

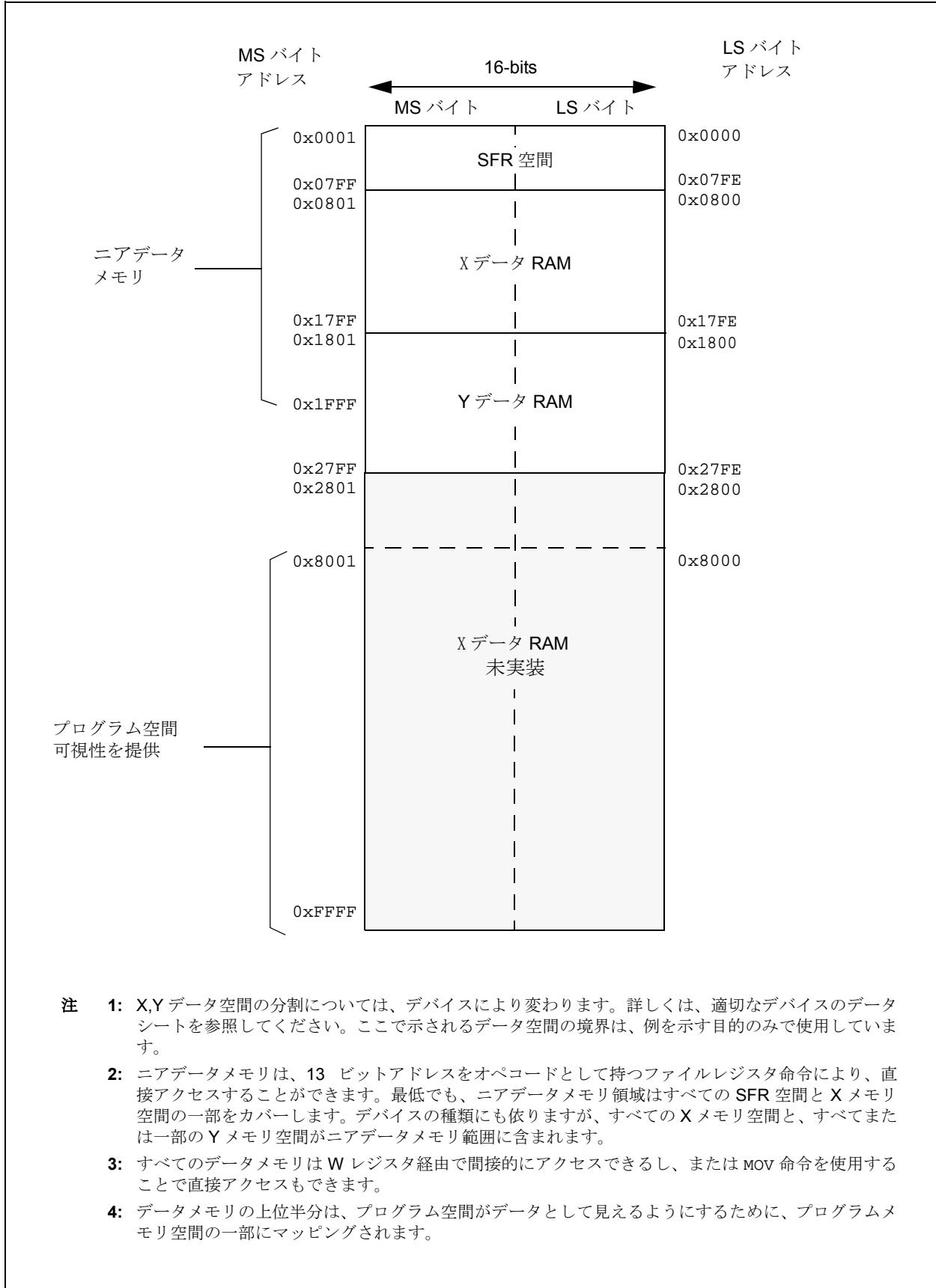
いくつかの DSP 命令は、命令では対象とされないアキュムレータをデータメモリに格納できるものがあります。この機能は、“アキュムレータ書き戻し (accumulator write back)” とよばれます。アキュムレータ書き戻しの場合にはひとつに結合されたデータメモリ空間へのアドレスポインタとしては、W13 を使用する必要があります。

DSP クラス命令では、すべてのメモリ読み出し用として、W8 と W9 が、実装されたされた X メモリ空間を指し示す必要があります。W8 または W9 が Y メモリ空間を指している場合、戻り値としてゼロが返されます。W8 または W9 が実装されていないメモリ空間を指し示す場合、アドレスエラートラップが発生します。

DSP クラス命令では、すべてのメモリ読み出し用として、W10 と W11 が、実装されたされた Y メモリ空間を指し示す必要があります。W10 または W11 が実装された X メモリ空間を指している場合、戻り値としてゼロが返されます。W10 または W11 が実装されていないメモリ空間を指し示す場合、アドレスエラートラップが発生します。アドレスエラートラップに関しての追加情報については、第 6 章、「割り込み」を参照してください。

**注：** データメモリマップと、X,Y データ空間の分割については、デバイスにより変わります。詳しくは、特定の dsPIC30F デバイスのデータシートを参照してください。

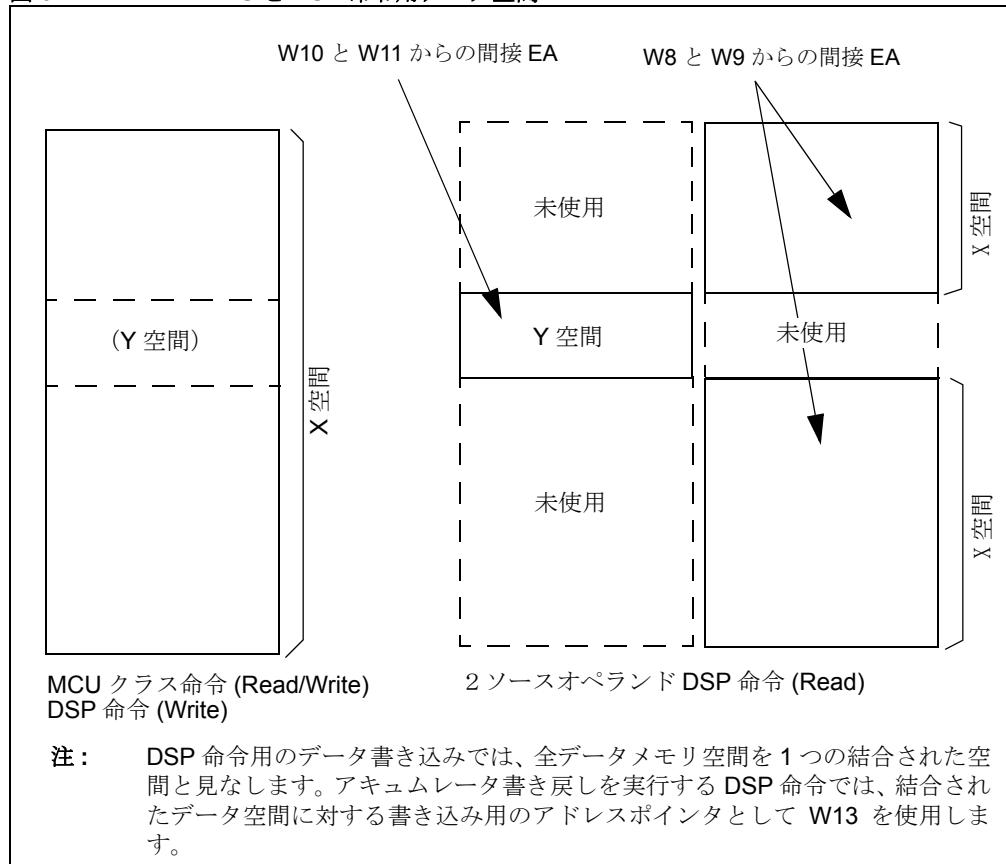
図 3-1: 例データメモリマップ



**注**

- 1: X,Y データ空間の分割については、デバイスにより変わります。詳しくは、適切なデバイスのデータシートを参照してください。ここで示されるデータ空間の境界は、例を示す目的のみで使用しています。
- 2: ニアデータメモリは、13 ビットアドレスをオペコードとして持つファイルレジスタ命令により、直接アクセスすることができます。最低でも、ニアデータメモリ領域はすべての SFR 空間と X メモリ空間の一部をカバーします。デバイスの種類にも依りますが、すべての X メモリ空間と、すべてまたは一部の Y メモリ空間がニアデータメモリ範囲に含まれます。
- 3: すべてのデータメモリは W レジスタ経由で間接的にアクセスできるし、または MOV 命令を使用することで直接アクセスもできます。
- 4: データメモリの上位半分は、プログラム空間がデータとして見えるようにするために、プログラムメモリ空間の一部にマッピングされます。

図 3-2: MCU と DSP 命令用データ空間



### 3.1.1 ニアデータメモリ

ニアデータメモリと呼ばれる、8-K バイトアドレス空間は、0x0000 から 0x1FFF の間のデータメモリ空間内に予約されています。ニアデータメモリは、すべてのファイルレジスタ命令内の 13-ビット絶対アドレッセフィールドにより、直接アドレスできます。

ニアデータ領域に含まれるメモリ範囲は、それぞれの dsPIC30F デバイスに実装されたデータメモリの量に依存します。最低でも、ニアデータ領域は、すべての SFRs と一部の X データメモリを含みます。データメモリの量が少ないデバイスの場合は、ニアデータ領域は、すべての X メモリと、一部またはすべての Y データメモリ空間を含む場合があります。詳しくは図 3-1 を参照してください。

**注:** 全 64K データ空間は、MOV 命令を使用することで直接アクセスすることができます。詳しくは、dsPIC30F プログラマリファレンスマニュアル (DS70030) を参照してください。

## 3.2 データ空間アドレス生成ユニット (AGUs)

dsPIC30F は、データメモリアドレスを生成するために、X AGU と Y AGU を内蔵しています。X と Y 両方の AGUs は 64-K バイト領域内で実行アドレス (EA) を生成します。ただし、実装された物理メモリの範囲外にある EAs は、データ読み込みの場合はゼロを返し、そのような位置へのデータ書き込みについては影響を与えません。さらに、アドレスエラートラップが発生します。アドレスエラートラップについての詳細情報は、[第6章.「割り込み」](#) を参照して下さい。

### 3.2.1 X アドレス生成ユニット

X AGU は、すべての命令で用いられ、すべてのアドレッシングモードをサポートします。X AGU は読み出し用 AGU (X RAGU) と書き込み用 AGU (X WAGU) から構成され、命令サイクルの別々のフェーズで、分かれている読み出しバスと書き込みバスに対して独立に動作します。X 読み出しバスは、X と Y アドレス空間を結合したひとつのデータ空間として扱うすべての命令に対して、戻りデータ読み出し経路になります。それはまた、2 つのオペランドを持つ読み出し命令 (DSP 命令クラス) に対する X アドレス空間のデータ読み出し経路にもなります。X 書き込みデータバスは、すべての命令に対して、結合された X と Y アドレス空間に対する唯一の書き込み経路です。

X RAGU は、直前にプリフェッチされた命令から引き出される情報を使用して、前の命令サイクルの間に実行アドレスの計算を始めます。X RAGU EA は、命令サイクルの開始時点でのアドレスバスに現れます。

X WAGU は、命令サイクルの開始時点で実行アドレスの計算を開始します。EA は、命令の書き込みフェーズの間でアドレスバスに現れます。

X RAGU と X WAGU はともに、モジュロアドレッシングをサポートします。

ビット反転アドレッシングは X WAGU のみでサポートされます。

### 3.2.2 Y アドレス生成ユニット

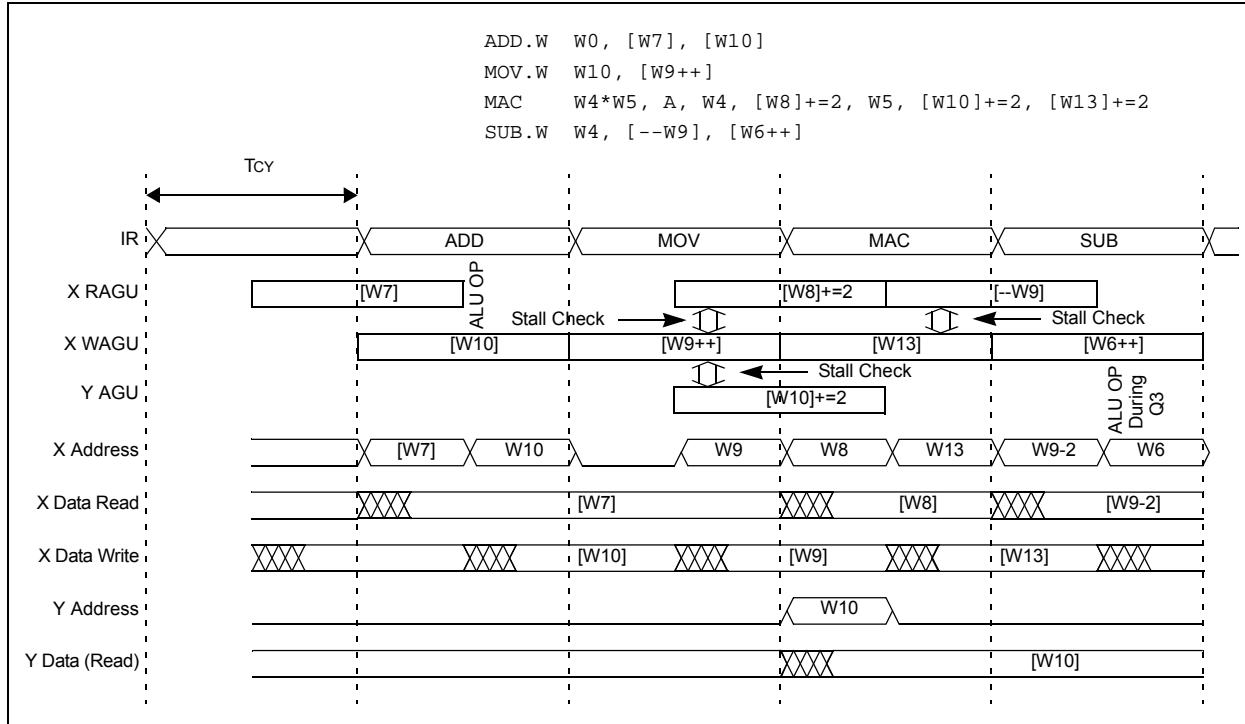
Y データメモリ空間は、Y データメモリ空間からのデータ読み出しをサポートする、1 つの AGU を持っています。Y メモリバスはデータ書き込みには使用されません。Y AGU と Y メモリバスの機能は、DSP クラス命令用として並列データ読み出しをサポートすることです。

Y AGU タイミングは X RAGU のタイミングと同じであり、直前にプリフェッチされた命令から引き出される情報を使用して、命令サイクルの前に実行アドレスの計算を始めます。EA は、命令サイクルの開始時点でのアドレスバスに現れます。

Y AGU は、DSP クラス命令用として、モジュロアドレッシングとポストモディフィケーションアドレッシングをサポートします。

**注:** Y AGU はデータ書き込みはサポートしません。すべてのデータ書き込みは X WAGU 経由で、結合された X、Y データ空間に対して発生します。Y AGU は、2 つのソースオペランドを持つ DSP 命令に対するデータ読み出しの時のみに使用されます。

図 3-3: データ空間アクセスタイミング



### 3.2.3 アドレス生成ユニットと DSP クラス命令

Y AGU と Y メモリデータ経路は、DSP クラス命令により、X RAGU と一緒に使用され、2つの並列データ読み出し経路を提供します。例えば、MAC 命令は、次の乗算に使用される 2つのオペランドを同時にプリフェッチします。

DSP クラス命令は、2つの W レジスタポインタ、W8 と W9 を専用化し、X RAGU を通して、Y データ空間とは独立に X データ空間を常に動作させるようにします。さらに 2つの W レジスタポインタ、W10 と W11 を専用化し、YAGU を通して、X データ空間とは独立に Y データ空間を常に動作させるようにします。DSP クラス命令により実行されるデータ書き込みは、結合された X,Y データ空間内で発生し、書き込みは X-バスを経由して発生します。その結果、EA が示す場所とは関係ないどんなアドレスにも書き込みできます。

YAGU は、DSP クラス命令と関連して、ポストモディフィケーション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について詳しくは、dsPIC30F プログラマリファレンスマニュアルを参照してください。Y AGU はまた、自動巡回バッファ用として、モジュロアドレッシングをサポートします。その他すべての (MCU) クラス命令については、X AGU 経由で複合リニア空間の一部とみなされる Y データアドレス空間をアクセスできます。

### 3.2.4 データ配置

ISA は、X メモリ AGU 経由でデータをアクセスするすべての MCU 命令に対して、ワード・バイト両方の動作をサポートします。ワード動作では、16-ビットデータアドレスの LSB は無視されます。ワードデータはリトルエンディアンフォーマットで配置され、偶数アドレス (LSB=0) には LSByte が、奇数アドレス (LSB = 1) には MSByte が配置されます。

バイト動作では、データアドレスの LSB はアクセスされるべきバイトを選択するために使用されます。このアドレスされたバイトは内部データバスの下位 8 ビット上に配置されます。

すべての実効アドレスの計算は、バイトまたはワードアクセスのどちらが実行されるかに依存して自動的に調整されます。例えば、アドレスポインタを後置増分するワード動作においては、アドレスは 2 つ増分されます。

**注:** すべてのワードアクセスでは、偶数アドレス (LSB = 0) に合わせなければなりません。配置を誤ったワードデータをフェッチする機能はサポートされていません。したがって、バイト・ワードの動作を混在させる場合または現存する PICmicro コードから翻訳する際には注意が必要です。誤配置のワードの読み書きを行うと、アドレスエラートラップが発生します。誤配置の読み出し動作は完了しますが、誤配置の書き込みは行われません。トラップが発生し、システムがアドレス FAULT の実行以前にマシン状態を検査するようにします。

図 3-4: データ配列

	MS バイト 15 8 7 0	LS バイト 0	
0001	バイト 1	バイト 0	0000
0003	バイト 3	バイト 2	0002
0005	バイト 5	バイト 4	0004
	ワード 0		0006
	ワード 1		0008
	ロング・ワード <15:0>		000A
	ロング・ワード <31:16>		000C

### 3.3 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により、ハードウェアを使用した巡回データバッファをサポートする自動化手段を提供します。この目的は、典型的な多くの DSP アルゴリズムで使うきつい高速なループコードを実行する際に、アドレス境界のチェックをソフトウェアで行うことなくすることです。

W15 以外の W レジスタは、モジュロバッファのポインタとして選択できます。モジュロハードウェアは、選択された W レジスタにあるアドレスに関して境界条件をチェックし、必要であれば、バッファ境界におけるポインタ値を自動的に調整します。

dsPIC30F のモジュロアドレッシングは、データまたはプログラム空間内で動作します（データポインタ機構は基本的に両方とも同じだからです。）。1 つの巡回バッファは、X データ空間（これはまたプログラム空間のポインタを提供します。）および Y データ空間内でサポートされます。

モジュロデータバッファ長は最大 32K ワードのサイズです。モジュロバッファ論理は、ワードまたはバイトサイズのデータを使用するバッファをサポートします。ただし、モジュロ論理はワードアドレス境界ではアドレス境界チェックのみを実行しますので、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。さらに、バイトアクセスが Y メモリデータバス経由ではサポートされていませんので、バイトサイズのモジュロバッファは、Y AGU を使用して実装することはできません。

## 3.3.1 モジュロの開始アドレスと終了アドレスの選択

モジュロバッファの開始・終了アドレスを指定するために、4つのアドレスレジスタが使用できます。

- XMODSRT: X AGU モジュロ開始アドレスレジスタ
- XMODEND: X AGU モジュロ終了アドレスレジスタ
- YMDSRT: Y AGU モジュロ開始アドレスレジスタ
- YMDEND: Y AGU モジュロ終了アドレスレジスタ

モジュロバッファの開始アドレスは偶数バイトアドレス境界に配置する必要があります。XMODSRT と YMDSRT レジスタの LSB は、正しいモジュロ開始アドレスを保証するために ‘0’ に固定される必要があります。モジュロバッファの終了アドレスは奇数バイトアドレス境界に配置される必要があります。XMODEND と YMDEND レジスタの LSB は、正しいモジュロ終了アドレスを保証するために ‘1’ に固定される必要があります。

それぞれのモジュロバッファ用に選択された開始アドレスと終了アドレスは、実装されるのが増分バッファであるか減分バッファであるかにより、ある制限を持ちます。増分バッファの場合は、W レジスタポインタは、バッファアドレス範囲内で増分されます。増分バッファの終了アドレスに達すると、W レジスタポインタはリセットされバッファの開始点になります。減分バッファの場合は、W レジスタポインタは、バッファアドレス範囲内で減分されます。減分バッファの開始アドレスに達すると、W レジスタポインタはリセットされバッファの終了点になります。

**注:** ユーザーは、増分または減分モジュロバッファのどちらがアプリケーションで必要とされるかを決定する必要があります。増分または減分モジュロバッファのうちのどちらが実装されるかにより、アドレス制約が存在することになります。

### 3.3.1.1 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、増分モジュロバッファでは、2のバイナリ「ゼロ」境界の値でなければなりません。モジュロ開始アドレスは、減分モジュロバッファではどんな値も取ることができ、選ばれたバッファの終了アドレスとバッファ長を使用して計算されます。

例えば、増分バッファのバッファ長として 50 ワード(100 バイト)が選ばれると、バッファ開始バイトアドレスは、7つの LS ゼロを含む必要があります。有効な開始アドレスは、従って、0xNN00 と 0xNN80 となります。ここで ‘N’ は 16 進数の値です。

### 3.3.1.2 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、減分バッファではバイナリ「1」境界の値でなければなりません。モジュロ終了アドレスは、増分バッファではどんな値もとることができ、選ばれたバッファの開始アドレスとバッファ長を使用して計算されます。

例えば、バッファサイズ(モジュロ値)として 50 ワード(100 バイト)が選ばれると、減分モジュロバッファのバッファ終了バイトアドレスは、7つの LS 1 を含む必要があります。有効な終了アドレスは、従って、0xNNFF と 0xNN7F となります。ここで ‘N’ は 16 進数の値です。

**注:** 必要なモジュロバッファ長が 2 の偶数乗である場合、モジュロ開始と終了アドレスは、増分と減分バッファの要求を満たすように選ばれる必要があります。

### 3.3.1.3 モジュロアドレス計算

増分モジュロバッファの終了アドレスは、選択された開始アドレスと、選択されたバッファ長（バイト長）から計算されます。式 3-1 が、終了アドレスを計算するために使用されます。

**式 3-1:** 増分バッファのモジュロ終了アドレス

$$\text{End Address} = \text{Start Address} + \text{Buffer Length} - 1$$

減分モジュロバッファの開始アドレスは、選択された終了アドレスとバッファ長から、式 3-2 に示すように、計算されます。

**式 3-2:** 減分バッファのモジュロ開始アドレス

$$\text{Start Address} = \text{End Address} - \text{Buffer Length} + 1$$

### 3.3.1.4 モジュロアドレッシング SFRs に関するデータ依存性

モジュロアドレッシングコントロールレジスタ MODCON への書き込み動作直後には、W レジスタを使った間接アドレッシング読み出し動作を続けてはいけません。例 3-1 に示されるようなコードセグメントは、予期せぬ結果をもたらします。

- 注**
- 1:** POP 命令を使用して、スタックのトップ (TOS) にある内容を MODCON に動かすことでも、MODCON への書き込みを実行できます。MODCON への書き込み直後の命令として、間接読み出し動作を実行することはできません。
  - 2:** いくつかの命令は、暗黙のうちに、間接読み出し動作を行うものがあることに注意が必要です。それらは、POP, RETURN, RETFIE, RETLW および ULNK です。

**例 3-1:** 正しくない MODCON の初期化

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

初期化に関するこの問題に対処するためには、MODCON の初期化の直後に来る命令内で、間接読み出し以外のアドレッシングモードを使用してください。この問題の簡単な対策は、例 3-2 に示されるように、MODCON の初期化の後に NOP を追加することで達成されます。

**例 3-2:** 正しい MODCON の初期化

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

下記のモジュロアドレス SFRs への書き込み直後に実行される間接読み出し動作に関しては、さらに条件があります。

- XMDSRT
- XMODEND
- YMDSRT
- YMODEND

モジュロアドレッシングが MODCON すでに有効になっていると、X (または Y) モジュロアドレス SFRs への書き込み直後には、X-データ空間 (または Y-データ空間) からのモジュロバッファアクセスで指定される W レジスタを使用した間接読み出し動作を続けてはなりません。例 3-3 内のコードセグメントは、X-データ空間と関連してモジュロ SFRs を初期化することが、いかにして予期しない結果をもたらすかを示しています。同様な例は Y-データ空間の初期化でも示されます。

### 例 3-3: 正しくない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      ;XMDSRT is initialized
MOV w4, XMDSRT
MOV #0x12FF, w0      ;XMODEND is initialized
MOV w0, XMODEND
MOV [w4++], w5       ;Incorrect EA generated
```

この問題を回避するには、モジュロアドレス SFRs を初期化した後で、NOP 命令を挿入するか、またはモジュロバッファアクセス用に指定される W レジスタを使用する間接読み出し以外の命令を挿入してください。これは、例 3-4 で示されています。別の方法としては、モジュロ開始・終了アドレス SFRs を初期化した後に MODCON 内のモジュロアドレッシングを有効にすることです。

### 例 3-4: 正しいモジュロアドレッシングセットアップ

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

注：代わりに、モジュロバッファアドレス用に指定される W レジスタを使用して、間接読み出し命令の動作をしない他の命令を実行するという方法もあります。

### 3.3.2 Wアドレスレジスタ選択

モジュロアドレッシングが適用される X アドレスポインタ W レジスタ (XWM) は、MODCON<3:0> に格納されます (レジスタ 3-1 参照)。XMODSRT, XMODEND, および XWM レジスタ選択は、X RAGU と X WAGU の間で共有されます。モジュロアドレッシングは、XWM が 15 以外の値にセットされ、XMODEN ビットがセット (M0DC0N<15>) されると、X- データ空間に對して有効になります。W15 は、専用のソフトスタックポインタですので、モジュロアドレッシング用のポインタとしては使用できません。

モジュロアドレッシングが適用される Y アドレスポインタ W レジスタ (YWM) は、MODCON<7:4> に格納されます (レジスタ 3-2 参照)。モジュロアドレッシングは、YWM が 15 以外の値にセットされ、YMODEN ビットがセット (M0DC0N<14>) されると、Y- データ空間に對して有効になります。

**注:** MODCON レジスタへの書き込みの後に、W レジスタを使用した間接読み出し動作を実行する命令を続けてはいけません。予期せぬ結果が起こるかもしれません。いくつかの命令は、暗示的に間接読み出しを実行します。それらは、POP, RETURN, RETFIE, RETLW および ULNK です。

### 3.3.3 モジュロアドレッシング適用性

モジュロアドレッシングは、選択された W レジスタと関連して実効アドレス (EA) 計算に適用されます。アドレス境界テストは、増分バッファ用には、上位アドレス境界と同じかより大きいアドレスを、減分バッファ用には、下位アドレス境界と同じかより小さい値を探すことであると認識することが重要です。したがって、アドレス変更により境界を飛び越えたりしても、また正しい値に戻ります。モジュロハードウェアによる W レジスタポインタの自動調整は單方向のみであることに注意してください。すなわち、増分バッファ用の W レジスタポインタが減分するとき、または逆に減分バッファのときポインタが増分するときにも、W レジスタポインタはモジュロハードウェアによっては正確には調整されません。このルールの例外は、バッファ長が 2 の偶数乗であり、開始・終了アドレスが、増分・減分モジュロバッファの両方の境界要求を満たすように選択された時です。

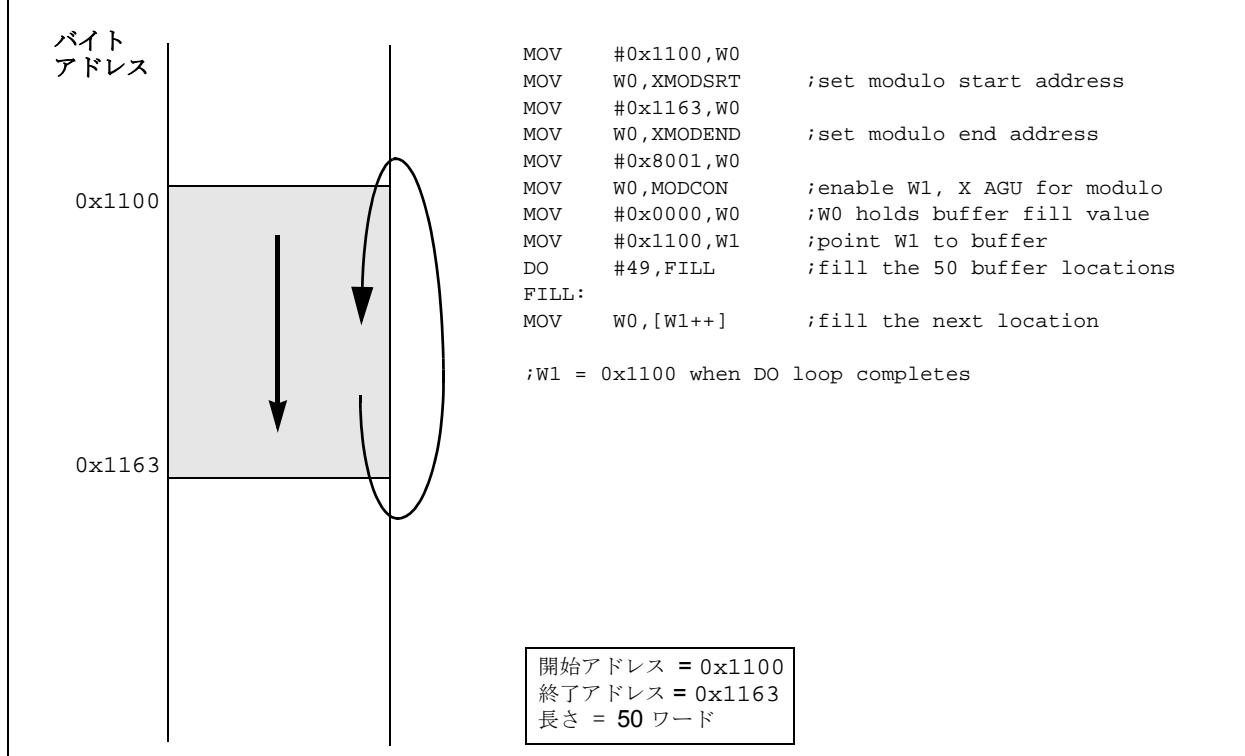
新しい EA は、バッファ長までモジュロバッファ境界を越えることができ、しかも正しく訂正されます。このことは、レジスタインデックスアドレッシングモード ([Wb + Wn]) とリテラルオフセット ([Wn + lit10]) アドレッシングモードを使う時には、覚えておくべき重要なことです。レジスタインデックスアドレッシングモードとリテラルオフセットアドレッシングモードは、W レジスタ内の値を変更することはないということを、ユーザーは覚えておくべきです。プリ・ポスト修飾を持つ、間接アドレッシングモード ([Wn++], [Wn--], [+Wn], [-Wn]) のみが W レジスタアドレスの値を修正します。

### 3.3.4 増分モジュロバッファ用のモジュロアドレッシングの初期化

以下のステップは、増分循環バッファ用のセットアップ手順を示しています。そのステップは X AGU または Y AGU が使用されても同様です。

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 倍し、バイト単位でのバッファ長を計算します。
2. 望ましいバッファ長に基づいたバイナリ ‘ゼロ’ 境界にある開始アドレスを選択します。バイトアドレス範囲を得るにはワード単位のバッファ長を 2 倍しなければならない点に注意して下さい。例えば、100 ワード(200 バイト)の長さを持つバッファは、開始アドレスとして 0xXX00 を使用します。
3. ステップ 1 で選択されたバッファ長とステップ 2 で選択されたバッファ開始アドレスを使用してバッファ終了アドレスを計算します。バッファ終了アドレスは式 3-1 を使用して計算されます。
4. XMDSRT (YMODSRT) レジスタに、ステップ 2 で選択されたバッファ開始アドレスを設定します。
5. XMODEEND (YMODEEND) レジスタに、ステップ 3 で計算されたバッファ終了アドレスを設定します。
6. MODCON レジスタ内の XWM<3:0> (YWM<3:0>) ビットに書き込み、循環バッファをアクセスするために使用される W レジスタを選択します。
7. MODCON レジスタ内の XMODEN (YMODEN) をセットし、循環バッファを有効にします。
8. 選択された W レジスタにバッファを指すアドレスを設定します。
9. W レジスタアドレスは、プリ / ポスト増分で間接アクセスが実行される時に、バッファの終了時点で自動的に調整されます(図 3-5 参照)。

図 3-5: 増分バッファモジュロアドレッシング動作の例

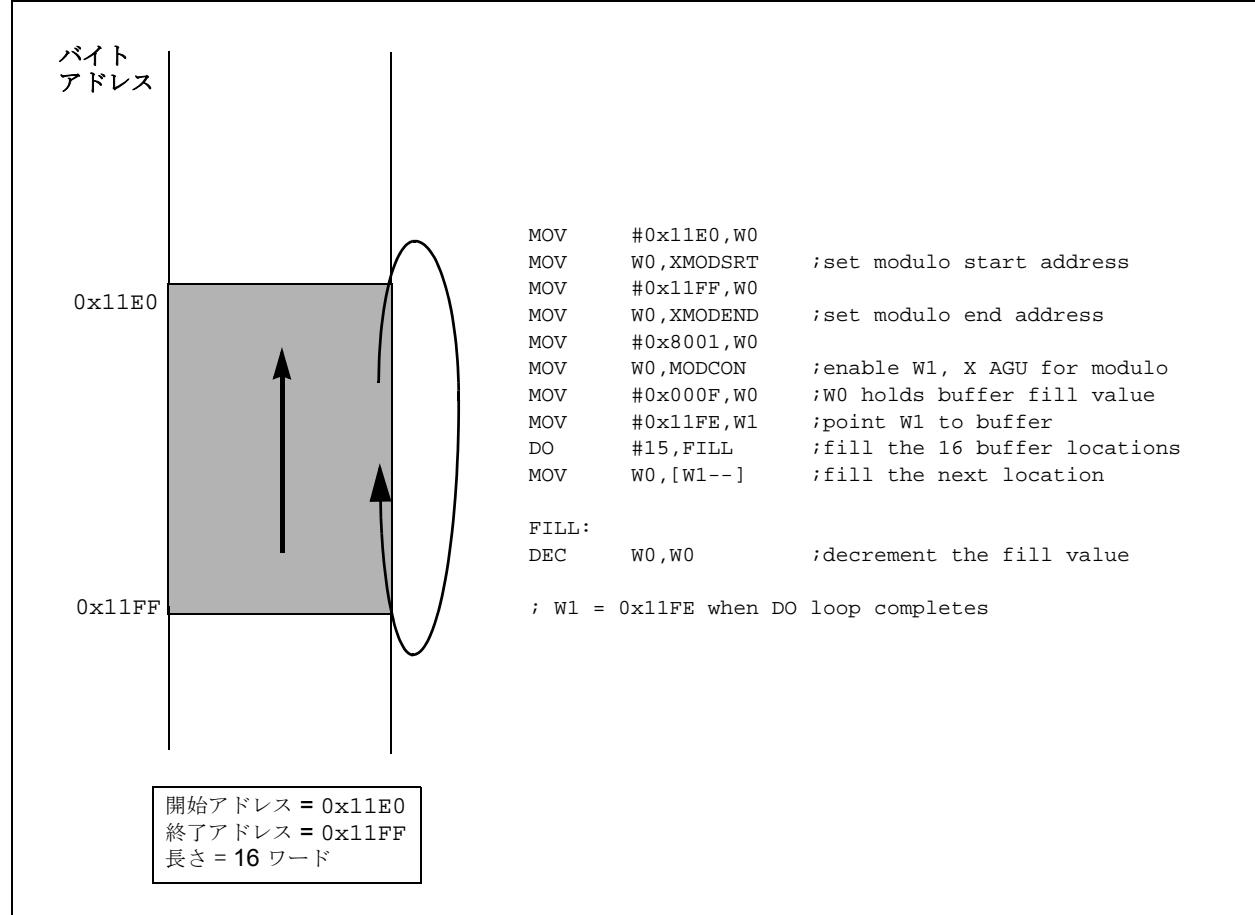


### 3.3.5 減分モジュロバッファ用のモジュロアドレッシングの初期化

以下のステップは、減分巡回バッファ用のセットアップ手順を示しています。そのステップは X AGU または Y AGU が使用されても同様です。

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 倍し、バイト単位でのバッファ長を計算します。
2. 望ましいバッファ長に基づいたバイナリ ‘ones’ 境界にある終了アドレスを選択します。バイトアドレス範囲を得るにはワード単位のバッファ長を 2 倍しなければならない点に注意して下さい。例えば、128 ワード (256 バイト) の長さを持つバッファは、終了アドレスとして 0xFFFF を使用します。
3. ステップ 1 で選択されたバッファ長とステップ 2 で選択されたバッファ終了アドレスを使用してバッファ開始アドレスを計算します。バッファ開始アドレスは式 3-2 を使用して計算されます。
4. XMDSRT (YMODSRT) レジスタに、ステップ 3 で選択されたバッファ開始アドレスを設定します。
5. XMODEEND (YMODEND) レジスタに、ステップ 2 で計算されたバッファ終了アドレスを設定します。
6. MODCON レジスタ内の XWM<3:0> (YWM<3:0>) ビットに書き込み、巡回バッファをアクセスするために使用される W レジスタを選択します。
7. MODCON レジスタ内の XMODEN (YMODEN) をセットし、巡回バッファを有効にします。
8. 選択された W レジスタにバッファを指すアドレスを設定します。
9. W レジスタアドレスは、プリ / ポスト増分で間接アクセスが実行される時に、バッファの終了時点で自動的に調整されます(図 3-6 参照)。

図 3-6: 減分バッファモジュロアドレッシング動作の例



### 3.4 ビット反転アドレッシング

#### 3.4.1 ビット反転アドレッシングへの序章

ビット反転アドレッシングは、radix-2 の FFT アルゴリズム用のデータ並び替えを簡単にします。これは、X WAGU によってのみサポートされます。ビット反転アドレッシングは、図 3-7 に示すように、バイナリ値の中央値の周りでビット位置を交換することによりアドレスポインタの‘ミラーイメージ’を効果的に生成することで達成されます。表 3-1 は、ビットアドレスフィールドのビット反転シーケンスの例を示しています。

図 3-7: ビット反転アドレッシングの例

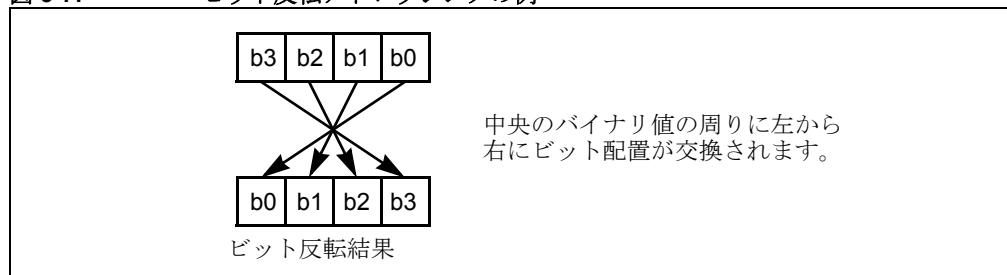


表 3-1: ビット反転アドレスシーケンス (16 エントリ)

通常 アドレス					ビット反転 アドレス				
A3	A2	A1	A0	小数	A3	A2	A1	A0	小数
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

### 3.4.2 ビット反転アドレッシング動作

ビット反転アドレッシングは、X WAGU によってのみサポートされ、MODCON と XBREV の SFR によってコントロールされます。ビット反転アドレッシングは以下のようにして起動されます。

1. ビット反転アドレッシングは、BWM コントロールビット (MODCON<11:8>) を使用して W レジスタの 1 つに割り当てられます。
2. ビット反転アドレッシングは BREN コントロールビット (XBREV<15>) をセットすることで有効になります。
3. X AGU のビット反転修飾子は、XB コントロールビット (XBREV<14:0>) 経由で設定されます。

有効になると、ビット反転アドレッシング用ハードウエアは、プリ / ポスト増分アドレッシングモードを持ったレジスタ間接モード ([Wn++], [++Wn]) が使用される場合のみ、ビット反転アドレスを生成します。さらに、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またはバイトモードに対しては機能しません（通常アドレスが生成されます）。

**注：** MODCON レジスタへの書き込みの後に、W レジスタを使用した間接読み出し動作を実行するような命令を行ってはなりません。予期せぬ結果が発生するかもしれません。いくつかの命令は、暗示的に間接読み出しを実行します。それらは、POP, RETURN, RETFIE, RETLW および ULNK です。

#### 3.4.2.1 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、同じ W レジスタを使用することで、同時に有効にすることができますが、ビット反転アドレッシング動作は、有効の時には、データ書き込みより先に実行されます。例えば、以下のようない定手順で、同じ W レジスタをモジュロとビット反転アドレッシングに設定します。

- ・ X モジュロアドレッシングを有効にします。 (XMODEN = 1)
- ・ ビット反転アドレッシングを有効にします。 (BREN = 1)
- ・ W1 をモジュロアドレッシングに設定します。 (XWM<3:0> = 0001)
- ・ W1 をビット反転アドレッシングに設定します。 (BWM<3:0> = 0001)

W1 をポインタとして使用するデータ読み出しのときに、モジュロアドレス境界チェックが動作します。W1 を目的アドレスポインタとして使用するデータ書き込みのときに、ビット反転ハードウエアがデータ並び替えを実行して W1 の値を訂正します。

#### 3.4.2.2 XBREV と関連したデータ依存性

ビット反転アドレッシングが BREN (XBREV<15>) ビットがセットされて、すでに有効になっていたら、XBREV レジスタへの書き込みの後に、ビット反転アドレスポインタとして指定された W レジスタを使用した間接読み出し動作を行ってはなりません。

## 3.4.3 ビット反転修飾子の値

**XBREV** レジスタに設定された値は、ビット反転データバッファのサイズを間接的に規定する定数となります。共通ビット反転バッファと一緒に用いられる **XB** 修飾子の値は、表 3-2 のように纏められます。

表 3-2: ビット反転アドレス変更子値

バッファサイズ(ワード)	<b>XB</b> ビット反転アドレス修飾子値
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

注： 表に示されたビット反転修飾子の値のみが、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、W の内容と **XB** 修飾子定数に反転キャリーを追加することにより、W レジスタアドレスを修正します。反転キャリー追加は右から左ではなく、左から右へのビット追加で実施されます。ビット位置内でキャリーアウトが発生すると、キャリーアウトビットは右隣のビットに追加されます。例 3-5 では、**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 W レジスタの値を示しています。

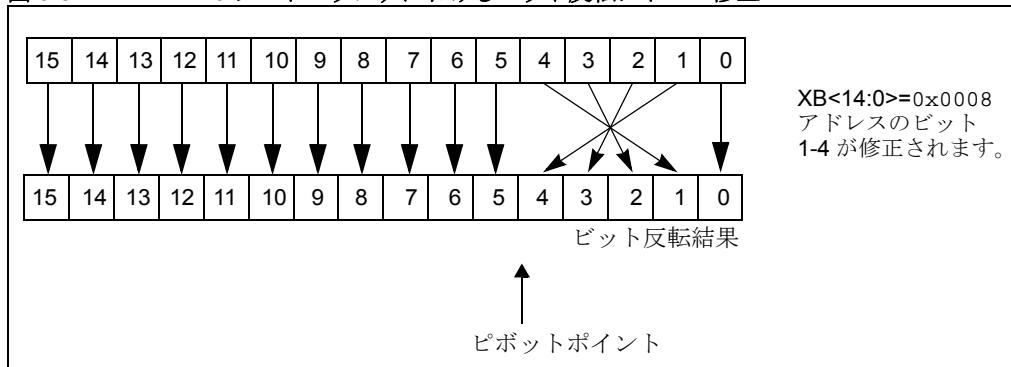
例 3-5: XB アドレス計算

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 の時、ビット反転バッファサイズは 16 ワードです。W 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 5-15( ピボットポイントより外 ) はビット反転ハードウェ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についてのみ動作しますので、ビット 0 は修正されません。

XB 修飾子は、ビット反転アドレス修正用の ‘ピボットポイント’ をコントロールします。ピボットポイント外のビットは、ビット反転アドレス訂正に依存しません。

図 3-8: 16 ワードバッファにおけるビット反転アドレス修正



## 3.4.4 ビット反転アドレッシングコードの例

以下のコード例では、一連の 16 データワードを読み込み、ビット反転の順に新しい位置にデータを書き込みます。W0 は読み込みアドレスポインタで、W1 は、ビット反転修正に依存する書き込みアドレスポインタです。

```
; 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++]
```

## 3.5 コントロールレジスタ記述

以下のレジスタはモジュロとビット反転アドレッシングをコントロールするのに用いられるレジスタです。

- MODCON: モジュロアドレッシングコントロールレジスタ
- XMDSRT: X AGU モジュロ開始アドレスレジスタ
- XMODEND: X AGU モジュロ終了アドレスレジスタ
- YMDSRT: Y AGU モジュロ開始アドレスレジスタ
- YMODEND: Y AGU モジュロ終了アドレスレジスタ
- XBREV: X AGU ビット反転アドレッシングコントロールレジスタ

それぞれのレジスタの詳細説明については、以下のページに示されます。

## レジスタ 3-1: MODCON: モジュロとピット反転アドレッシングコントロールレジスタ

上位バイト:									
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>					
ビット 15									

下位バイト:							
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>	
						ビット 0	

ビット XMODEN: X RAGU と X WAGU モジュラスアドレッシング有効ビット

15 1 = X AGU モジュラスアドレッシングビットが有効  
0 = X AGU モジュラスアドレッシングビットが無効

ビット YMODEN: Y AGU モジュラスアドレッシング有効ビット

14 1 = Y AGU モジュラスアドレッシングビットが有効  
0 = Y AGU モジュラスアドレッシングビットが無効

ビット 未実装: '0' が読み出されます。

13-12

ビット BWM<3:0>: XWAGU レジスタのビット反転アドレッシング選択ビット

11-8 1111 = ビット反転レジスタアドレッシング無効  
1110 = W14 をビット反転アドレッシング用として選択  
1101 = W13 をビット反転アドレッシング用として選択  
•  
•  
0000 = W0 をビット反転アドレッシング用として選択

ビット YWM<3:0>: Y AGU W レジスタのモジュロアドレッシング選択ビット

7-4 1111 = モジュロアドレッシング無効  
1110 = W14 をモジュロアドレッシング用として選択  
1101 = W13 をビット反転アドレッシング用として選択  
•  
•  
0000 = W0 をモジュロアドレッシング用として選択

ビット XWM<3:0>: X RAGU と X WAGU W レジスタのモジュロアドレッシング選択ビット

3-0 1111 = モジュロアドレッシング無効  
1110 = W14 をモジュロアドレッシング用として選択  
•  
•  
0000 = W0 をモジュロアドレッシング用として選択

注: MODCON レジスタへの書き込みの後に、W レジスタを使用した間接読み出し動作を実行するような命令を行ってはなりません。予期せぬ結果が発生するかもしれません。いくつかの命令は、暗示的に間接読み出しを実行します。それらは、POP, RETURN, RETFIE, RETLW および ULINK です。

凡例:

R = 読み出しができるビット

W = 書き込みができるビット

U = 未実装ビット、読み出し時 '0'

-n = POR での値

'1' = ビットセット

'0' = ビットクリア

x = ビット不定

# dsPIC30F ファミリーリファレンスマニュアル

## 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

上位バイト :							
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>							
ビット 15							ビット 8

下位バイト :							
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>							
ビット 7							ビット 0

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

ビット 0 未実装: ‘0’ が読み出されます。

凡例 :

R = 読み出しできるビット      W = 書き込みできる      U = 未実装ビット、読み出し時 ‘0’

ビット

-n = POR での値      ‘1’ = ビットセット      ‘0’ = ビットクリア      x = ビット不定

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

上位バイト :							
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>							
ビット 15							ビット 8

下位バイト :							
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>							
ビット 7							ビット 0

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

ビット 0 未実装: ‘1’ が読み出されます。

凡例 :

R = 読み出しできるビット      W = 書き込みできる      U = 未実装ビット、読み出し時 ‘0’

ビット

-n = POR での値      ‘1’ = ビットセット      ‘0’ = ビットクリア      x = ビット不定

## レジスタ 3-4: YM0DSRT: Y AGU モジュロアドレッシング開始レジスタ

上位バイト:	R/W-0						
YS<15:8>							
ビット 15							

## 下位バイト:

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>							
ビット 7							

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

ビット 0 未実装: ‘0’ が読み出されます。

## 凡例:

R = 読み出しできるビット	W = 書き込みできるビット	U = 未実装ビット、読み出し時 ‘0’
-n = POR での値	‘1’ = ビットセット	‘0’ = ビットクリア

x = ビット不定

## レジスタ 3-5: YM0DEND: Y AGU モジュロアドレッシング終了レジスタ

上位バイト:	R/W-0						
YE<15:8>							
ビット 15							

## 下位バイト:

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>							
ビット 7							

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

ビット 0 未実装: ‘1’ が読み出されます。

## 凡例:

R = 読み出しできるビット	W = 書き込みできるビット	U = 未実装ビット、読み出し時 ‘0’
-n = POR での値	‘1’ = ビットセット	‘0’ = ビットクリア

x = ビット不定

## レジスタ 3-6: XBREV: X Write AGU ビット反転アドレッシングコントロールレジスタ

上位バイト :							
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>						
ビット 15							ビット 8

下位バイト :							
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>						
ビット 7							ビット 0

ビット 15 **BREN:** ビット反転アドレッシング (XAGU) 有効ビット

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

ビット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

凡例 :

R = 読み出しできるビット    W = 書き込みできる    U = 未実装ビット、読み出し時 ‘0’  
ビット

-n = POR での値    ‘1’ = ビットセット    ‘0’ = ビットクリア    x = ビット不定

## 3.6 関連するアプリケーションノート

この章では、マニュアルのこの章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリ用に書かれているわけではありませんが、その概念は適切であり、修正して使用できるし、制限がある場合もあります。現状、データメモリモジュールに関連するアプリケーションノートは以下の通りです。

題目	アプリケーションノート #
現在のところ、関連するアプリケーションノートはありません。	

**注：** dsPIC30F ファミリのデバイスに関しての、他のアプリケーションノートやコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 3.7 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

この版は、dsPIC30F データメモリモジュール用の追加技術情報を含みます。



# MICROCHIP

## 第4章. プログラムメモリ

### ハイライト

この章は、以下の項目を含んでいます。

4.1	プログラムメモリアドレスマップ.....	4-2
4.2	プログラムカウンタ (PC) .....	4-4
4.3	プログラムメモリからのデータアクセス.....	4-4
4.4	データ空間からのプログラム空間の可視化.....	4-8
4.5	プログラムメモリへの書き込み.....	4-10
4.6	関連するアプリケーションノート.....	4-11
4.7	改訂履歴.....	4-12

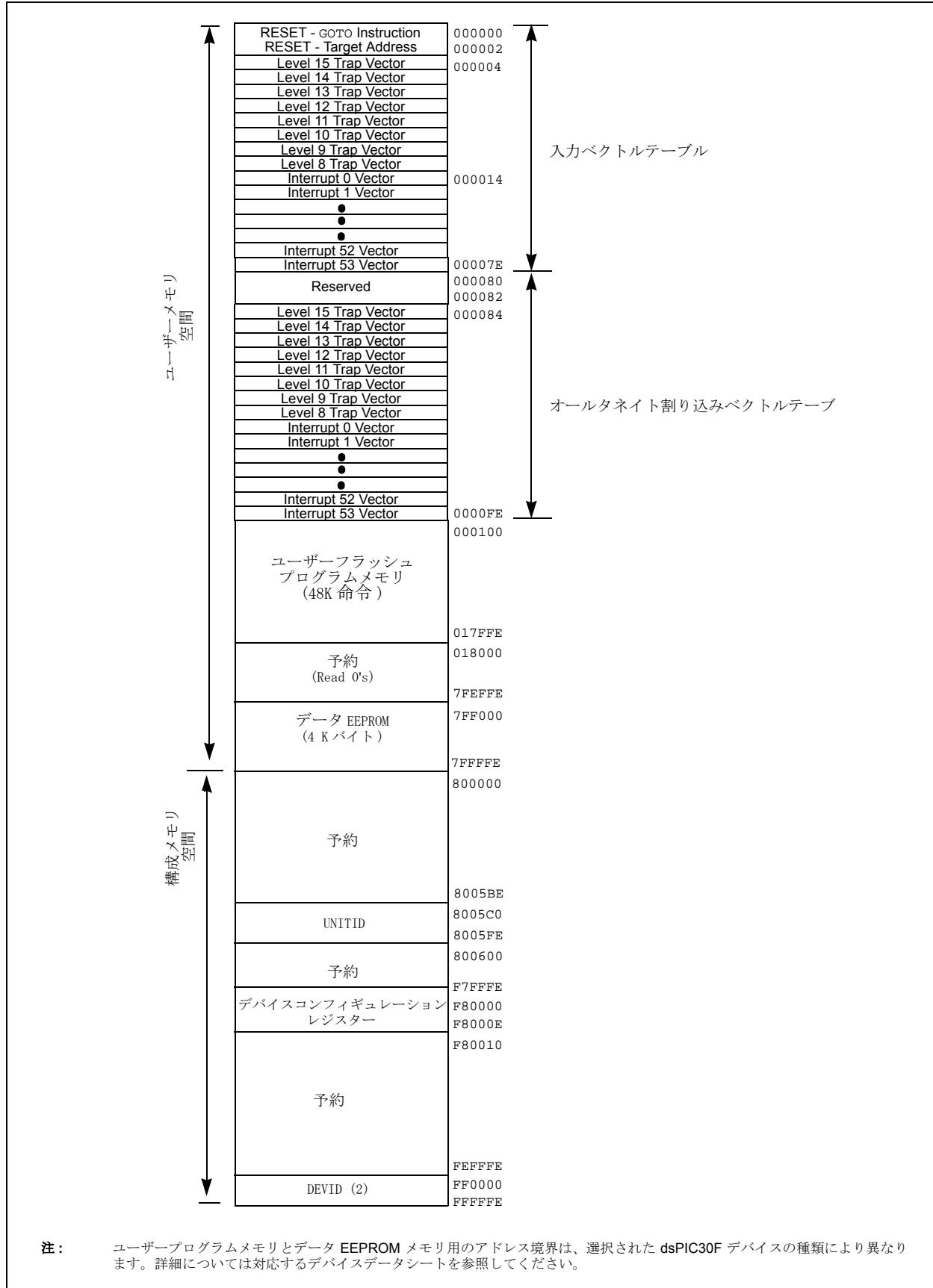
## 4.1 プログラムメモリアドレスマップ

dsPIC30F デバイスは **4M x 24** ビットのプログラムメモリアドレス空間を持っており、それは図 4-1 に示されます。プログラム空間をアクセスするために 3 つの方法があります。

1. 23- ビット PC による。
2. テーブルリード (TBLRD) とテーブルライト (TBLWT) 命令による。
3. プログラムメモリの 32-K バイトセグメントをデータメモリアドレス空間にマッピングすることによる。

プログラムメモリマップはユーザープログラム空間とユーザーコンフィギュレーション空間に分けられます。ユーザープログラム空間には RESET ベクトル、割り込みベクトルテーブル、プログラムメモリおよび EEPROM メモリを含みます。ユーザーコンフィギュレーション空間は、デバイスオプション設定用の不揮発コンフィギュレーションビットとデバイス ID 部を含みます。

図4-1: プログラム空間メモリマップの例



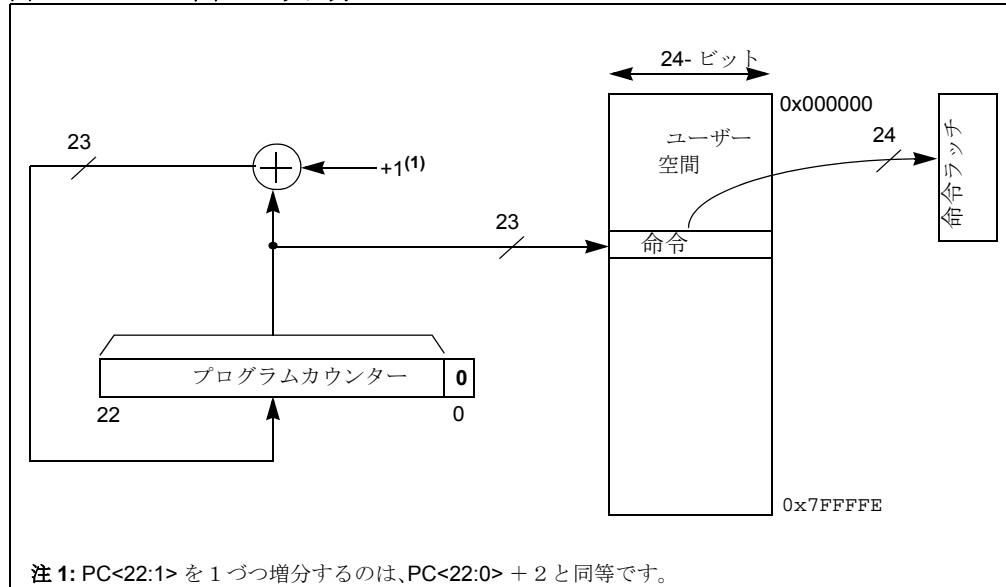
## 4.2 プログラムカウンタ (PC)

PC は、データ空間アドレスとの整合性を与えるために、LSb を ‘0’ に設定し 2 単位で増分します。連続する命令ワードは、PC<22:1> により 4M プログラムメモリ空間内でアドレスされます。それぞれの命令は 24 ビット幅です。

プログラムカウンタの LSB(PC<0>) は、プログラムメモリをプログラム空間可視化してデータ空間としてアクセスした場合と、テーブル命令でアクセスした場合に使われるバイト選択ビットとして予約されています。PC による命令フェッチのときには、バイト選択ビットは不要です。従って、PC<0> は常に ‘0’ にセットされます。

命令フェッチの例を図 4-2 に示します。PC<22:1> を 1 づつ増分するのは、PC<22:0> に 2 を加えることと同等である点に注意してください。

図 4-2: 命令フェッチ例



注 1: PC<22:1> を 1 づつ増分するのは、PC<22:0> + 2 と同等です。

## 4.3 プログラムメモリからのデータアクセス

プログラムメモリとデータメモリ空間の間のデータ転送に使用される方法として 2 つの方法があります。それは、特殊テーブル命令によるものと、データ空間の上位半分に 32K バイトのプログラム空間ページをマッピング変更することによるものの 2 つです。TBLRDL と TBLWTL 命令は、データ空間を経由することなく、プログラム空間内のアドレスの LSWORD を直接読み書きする方法を与え、これはあるアプリケーションでは好都合な方法です。TBLRDH と TBLWTH 命令は、プログラムワードの上位 8 ビットをデータとしてアクセスする、唯一の方法です。

### 4.3.1 テーブル命令の纏め

テーブル命令セットは、プログラム空間とデータ空間の間で、バイトまたはワードサイズのデータを動かすために使用されます。テーブル読み出し命令は、プログラムメモリ空間からデータメモリ空間への読み出しに使用されます。テーブル書き込み命令により、データメモリをプログラムメモリ空間へ書き込むことができます。

**注：** テーブル命令を使用した詳細なコードの例は、**第5章. 「FLASHとEEPROMのプログラミング」**に書かれています。

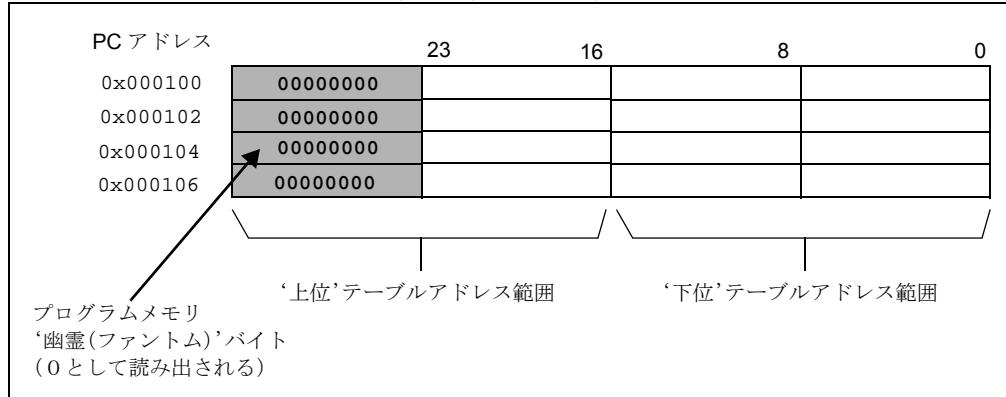
利用できる4つのテーブル命令は以下となります。

- TBLRDL: テーブル読み出し下位
- TBLWTL: テーブル書き込み下位
- TBLRDH: テーブル読み出し上位
- TBLWTH: テーブル書き込み上位

テーブル命令では、プログラムメモリは、隣り合って存在する2つの16ビットワード幅のアドレス空間として見なされ、図4-3に示すように、それらは同じアドレス範囲を持ちます。これにより、プログラム空間は、バイトまたは、配列されたワード単位で、16ビット幅、64Kバイトページ（すなわち、データ空間と同じ）としてアクセスできます。

TBLRDLとTBLWTLは、プログラムメモリの下位データワードをアクセスし、TBLRDHとTBLWTHは上位ワードをアクセスします。プログラムメモリは24ビット幅ですので、この後者の空間の上位のバイトは、アクセスは可能ですが、存在しません。従って、幽霊（ファンтом）バイトと呼ばれます。

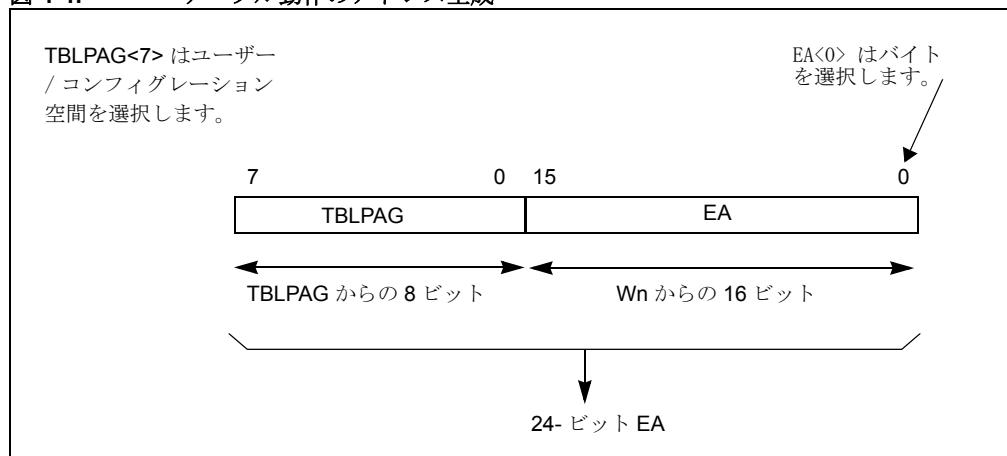
図4-3: テーブル動作用の上位・下位アドレス範囲



## 4.3.2 テーブルアドレス生成

すべてのテーブル命令では、W レジスタアドレス値は、8 ビットのデータテーブルページレジスタ (TBLPAG) と連結し、図 4-4 に示すように、23 ビットの実行プログラム空間アドレス、プラス、バイト選択ビットを構成します。W レジスタから与えられるプログラム空間アドレスは 15 ビットありますので、プログラムメモリ内のデータテーブルページサイズは、従って、32k ワードです。

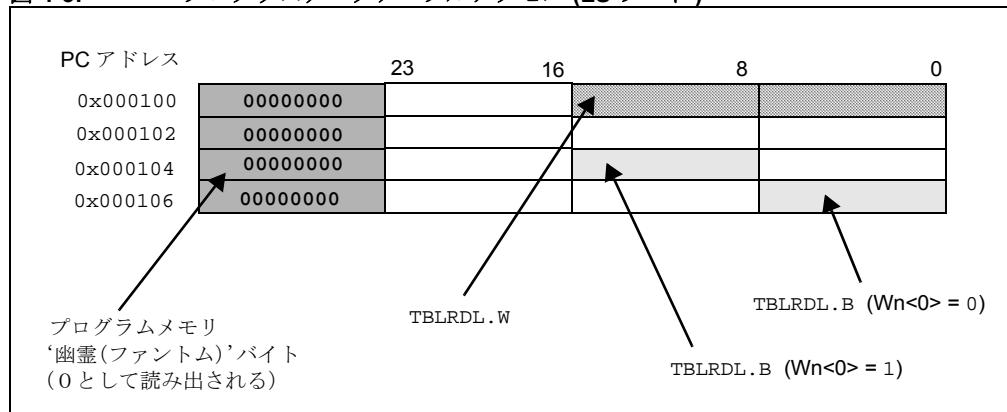
図 4-4: テーブル動作のアドレス生成



## 4.3.3 プログラムメモリ下位ワードアクセス

TBLRDL と TBLWTL 命令は、プログラムメモリデータの下位 16 ビットをアクセスするために使用されます。W レジスタアドレスの LS ビットは、ワード幅テーブルアクセス用としては無視されます。バイト幅アクセスでは、W レジスタアドレスの LS ビットは、どのバイトが読み出されるかを決定します。図 4-5 に、TBLRDL と TBLWTL 命令でアクセスされるプログラムメモリデータ領域を示します。

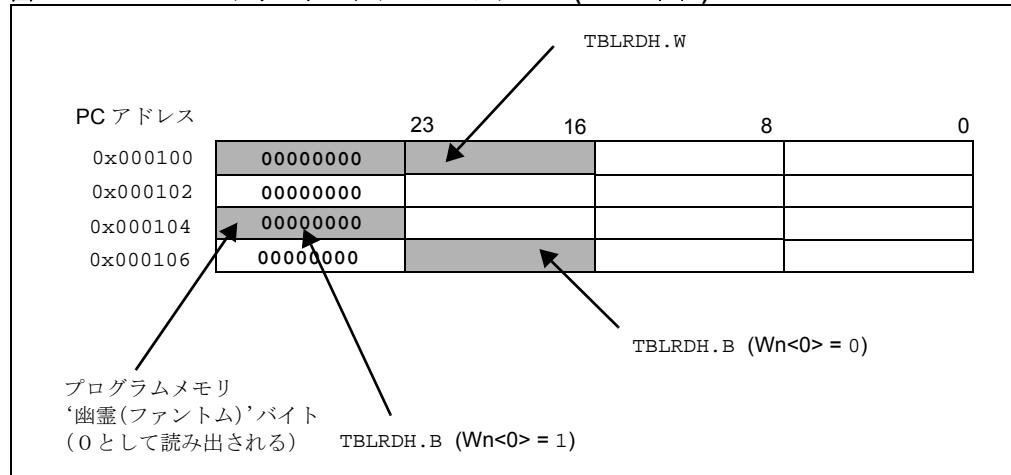
図 4-5: プログラムデータテーブルアクセス (LS ワード)



### 4.3.4 プログラムメモリ上位ワードアクセス

TBLRDH と TBLWTH 命令は、プログラムメモリデータの上位 8 ビットをアクセスするために使用されます。これらの命令はまた、ワードまたはバイトアクセスモードの直交性をサポートしますが、図 4-6 に示すように、プログラムメモリデータの上位バイトは、常に ‘0’ を返します。

図 4-6: プログラムデータテーブルアクセス (MS バイト)



### 4.3.5 プログラムメモリ内のデータストレージ

ほとんどのアプリケーションでは、上位バイト ( $P<23:16>$ ) はデータ用としては使用されないことを前提としており、その結果プログラムメモリはデータストレージとしては 16 ビット幅のように見えます。デバイスが、格納されたデータが偶然に実行されることから防ぐため、プログラムデータの上位バイトは NOP かまたは、無効なオペコードとしてプログラムされることを推奨します。TBLRDH と TBLWTH 命令は、元々は、配列プログラム / 検証目的のために、および圧縮されたデータストレージを要求するようなアプリケーションのために、準備されたのです。

## 4.4 データ空間からのプログラム空間の可視化

dsPIC30F データメモリアドレス空間の上位 32K バイトは、オプションとして 16K ワードプログラム空間ページにマッピングすることができます。この動作モードはプログラム空間可視性 (PSV) と呼ばれ、特別な命令を使用することなく、X データ空間から、格納された定数データに直接アクセスをすることができます。（すなわち、TBLRD、TBLWT 命令）

### 4.4.1 PSV 構成

プログラム空間可視性は、PSV ビット (C0RC0N<2>) をセットすることで有効になります。CORCON レジスタの説明は第 2 章、「CPU」で述べられています。

PSV が有効になると、データメモリマップの上位半分のアドレス内にあるそれぞれのデータ空間アドレスは、プログラムアドレスに直接マッピングされます（図 4-7 参照）。PSV ウィンドウにより 24 ビットプログラムワードの下位 16 ビットをアクセスできます。プログラムメモリデータの上位 8 ビットは、装置の堅牢性を維持するために、無効命令または NOP になるよう強制的にプログラムする必要があります。テーブル命令はプログラムメモリワードの上位 8 ビットを読み出す唯一の方法を提供することに注意してください。

図 4-8 は PSV アドレスがどのように生成されるかを示しています。PSV アドレスの下位 15 ビットは実効アドレスを含む W レジスタにより与えられます。W レジスタの最上位ビットはアドレスを形成するためには使用されません。その代わり、最上位ビットは、プログラム空間からの PSV アクセスを実行するか、データメモリ空間からの通常アクセスを実行するかを決定します。0x8000 かまたはそれより大きな W レジスタ実効アドレスが使用されると、PSV が有効な時は、データアクセスはプログラムメモリ空間に行われます。W レジスタ実効アドレスが 0x8000 より小さい時は、すべてのアクセスは、データメモリに行われます。

残りのアドレスビットは、図 4-8 に示すように PSVPAG レジスタ (PSVPAG<7:0>) により与えられます。PSVPAG ビットは、W レジスタの下位 15 ビットと結合され、23 ビットのプログラムメモリアドレスを形成するため実効アドレスを保持します。PSV はプログラムメモリ空間内の値をアクセスするときのみに使用されます。ユーザーコンフィギュレーション内の値をアクセスするためには、テーブル命令を使う必要があります。

W レジスタ値の LS ビットはバイト選択ビットとして使用され、PSV を使用したバイトまたはワードモードでの動作を指定します。

### 4.4.2 X と Y のデータ空間への PSV マッピング

Y データ空間は、ほとんどの dsPIC30F 系ではデータ空間の上位の外側に配置されているため、PSV エリアは X データ空間にマッピングされます。X と Y のマッピングは、アルゴリズム内でどのようにして PSV が使用されるかに影響を与えます。

例として、PSV マッピングは、有限インパルスレスポンス (FIR) フィルタアルゴリズムにおいて、係数データを格納するのに使用されます。FIR フィルタは、定数フィルタ係数を含むデータバッファの要素と、過去の入力データを含むデータバッファの値を乗算します。FIR アルゴリズムは、REPEAT ループ内で MAC 命令を使用して実行されます。MAC 命令の繰り返しの度に 1 つの過去の入力データと 1 つの係数をプリフェッチし、次の繰り返しの中で乗算します。プリフェッチされる値の 1 つは X データメモリ空間内に配置され、別の 1 つは Y データメモリ空間内に配置される必要があります。

FIR フィルタアルゴリズムにおいて、PSV マッピングの要求を満たすためには、過去の入力データを Y メモリ空間に、フィルタ係数は X メモリ空間に配置する必要があります。

図 4-7: プログラム空間可視性動作

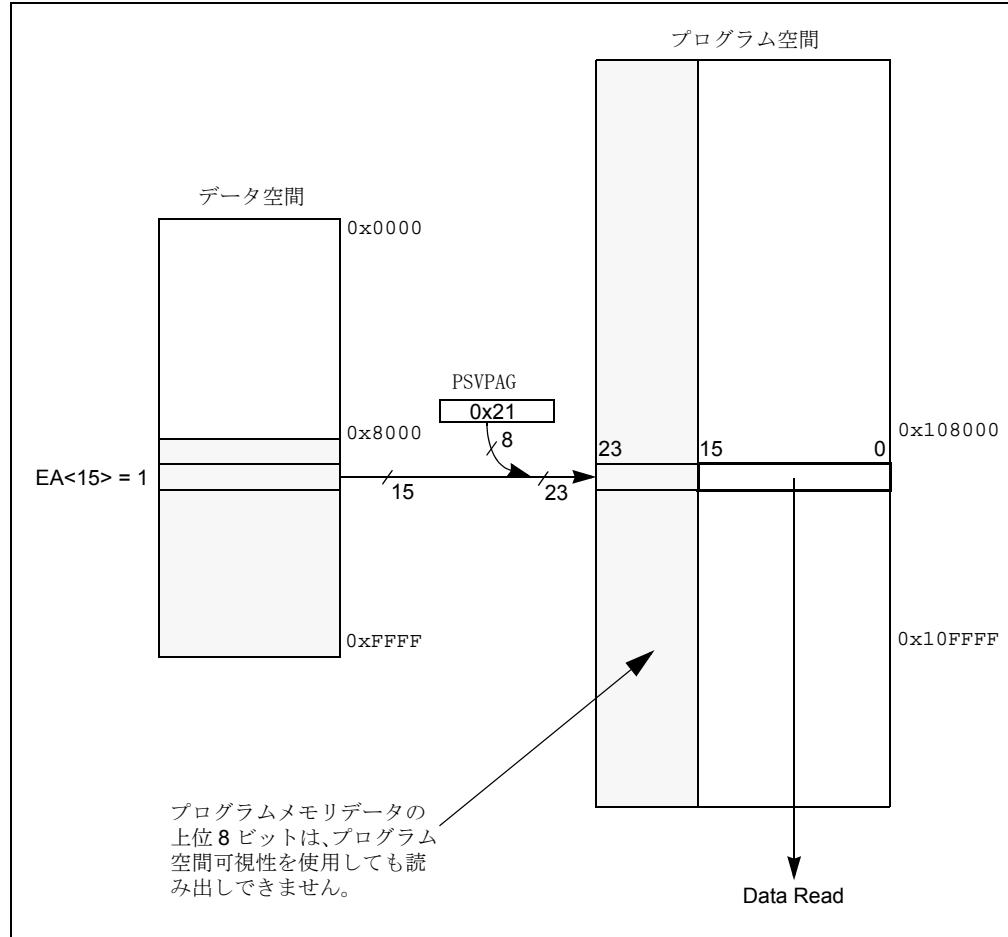
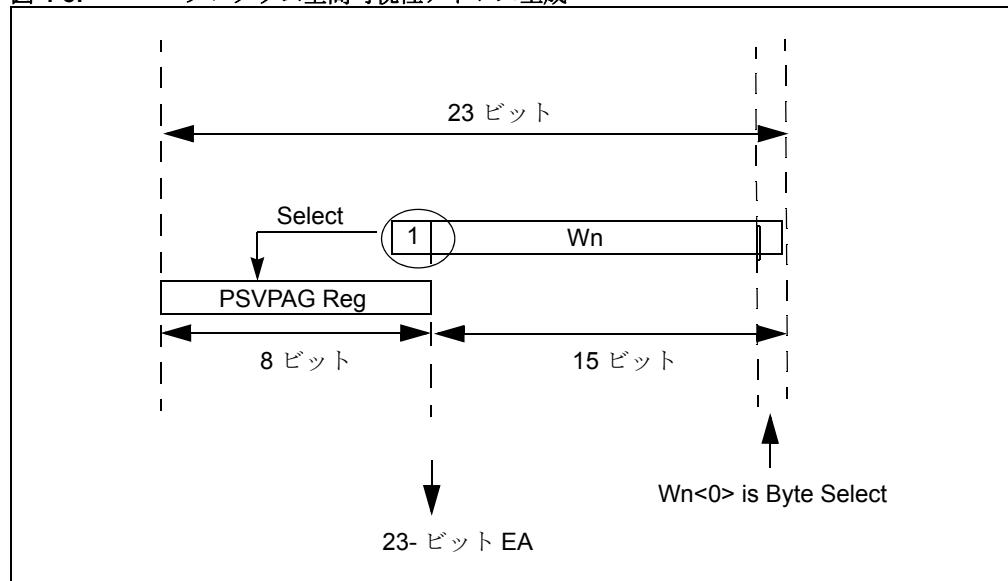


図 4-8: プログラム空間可視性アドレス生成



## 4.4.3 PSV タイミング

PSV を使用する命令は、実行するために 1 つ余計な命令サイクルを必要とします。この追加サイクルはプログラムメモリバス上の PSV データをフェッチするために使われます。

### 4.4.3.1 REPEAT ループ内で PSV を使用する

REPEAT ループ内で PSV を使用するほとんどの命令では、PSV メモリアクセスのために必要な余分な命令サイクルを無くすことができます。ただし、ループ内での命令の最初の繰り返しでは 2 サイクル必要です。REPEAT ループ内での、それに続く命令繰り返しは、1 サイクルで実行します。

### 4.4.3.2 PSV と命令のストール

PSV を使用した命令のストールについて詳しくは、[第 2 章.「CPU」](#) を参照してください。

## 4.5 プログラムメモリへの書き込み

dsPIC30F ファミリのデバイスは、ユーザーコードを実行するための FLASH メモリを内蔵しています。このメモリにユーザーが書き込む方法としては、2 つの方法があります。

1. 実行時自己プログラミング (RTSP)
2. インサーキットシリアルプログラミング™ (ICSP™)

RTSP は TBLWT 命令を使用して実行されます。ICSP は SPI インターフェースと統合ブートローダソフトウェアを使用して実行されます。RTSP に関して詳しくは、[第 5 章.「FLASH と EEPROM のプログラミング」](#) を参照してください。ICSP の仕様書は、Microchip Technology のウェブサイト ([www.microchip.com](http://www.microchip.com)) からダウンロードできます。

## 4.6 関連するアプリケーションノート

この章では、マニュアルのこの章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリ用に書かれているわけではありませんが、その概念は適切であり、修正して使用でき、制限がある場合もあります。現状、データメモリモジュールに関連するアプリケーションノートは以下の通りです。

題目	アプリケーションノート #
現在のところ、関連するアプリケーションノートはありません。	

**注：** dsPIC30F ファミリのデバイスに関する、その他のアプリケーションノートやコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 4.7 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

マニュアルの本章に対する技術内容や編集改訂版はありませんが、マニュアル全体を通して B 版を反映するために、この章は更新されています。



**MICROCHIP**

## 第5章 . FLASH と EEPROM のプログラミング

### ハイライト

この章は、以下の項目を含んでいます。

5.1	はじめに.....	5-2
5.2	テーブル命令動作.....	5-2
5.3	コントロールレジスタ.....	5-5
5.4	実行時セルフプログラミング (RTSP).....	5-9
5.5	データ EEPROM プログラミング .....	5-14
5.6	設計の秘訣.....	5-20
5.7	関連するアプリケーションノート.....	5-21
5.8	改訂履歴.....	5-22

## 5.1 はじめに

この章では、FLASH プログラムメモリと EEPROM メモリのプログラム技術について述べます。dsPIC30F ファミリーのデバイスは、ユーザーコードの実行用として内部プログラム FLASH メモリを内蔵しています。ユーザーがこのメモリにプログラムする方法として 2 つのやり方があります。

1. 実行時セルフプログラミング (RTSP)
2. インサーチットシリアルプログラミング™(ICSP™)

RTSP は、ユーザーのソフトウェアで実行されます。ICSP はデバイスへのシリアルデータ接続を使用して実行され、RTSP よりも高速でプログラミングできます。RTSP 技術はこの章で述べます。ICSP プロトコルは dsPIC30F プログラミング仕様書で述べられており、その仕様書は Microchip のウェブサイトからダウンロードできます。

データ EEPROM はプログラムメモリ空間にマッピングされます。EEPROM は 16 ビット幅のメモリで構成され、そのメモリサイズは最大 2K ワード(4K バイト)です。EEPROM の量はデバイスにより異なります。詳しくは、そのデバイスのデータシートを参照してください。

データ EEPROM 用に使用されるプログラミング技術は、FLASH プログラムメモリの RTSP に似ています。FLASH とデータ EEPROM プログラミング動作の主な違いは、プログラム / 消去サイクル時にプログラムしたり消去したりするデータの量です。

## 5.2 テーブル命令動作

テーブル命令は、dsPIC30F デバイスのプログラムメモリ空間とデータメモリ空間の間のデータ転送手段を提供します。テーブル命令は、FLASH プログラムメモリとデータ EEPROM のプログラミングの時に使用されるので、ここで概略をまとめます。4 つの基本的なテーブル命令があります。

- TBLRDL: テーブル読み出し下位
- TBLRDH: テーブル読み出し上位
- TBLWTL: テーブル書き込み下位
- TBLWTH: テーブル書き込み上位

TBLRDL と TBLWTL 命令は、プログラムメモリ空間のビット <15:0> に対しての読み書きをするために使用され、TBLRDL と TBLWTL はワードもしくはバイトモードでプログラムメモリをアクセスできます。

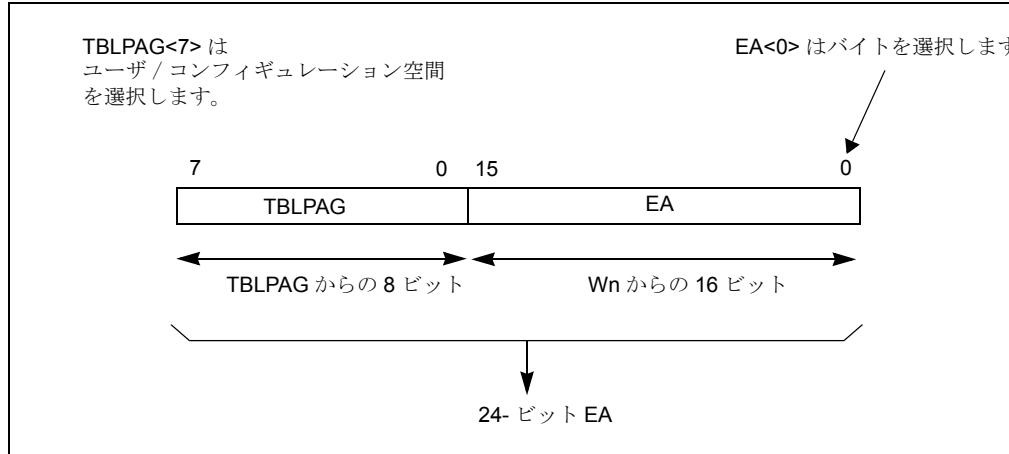
TBLRDH と TBLWTH 命令は、プログラムメモリ空間のビット <23:16> に対しての読み書きをするために使用され、TBLRDH と TBLWTH はワードもしくはバイトモードでプログラムメモリをアクセスできます。プログラムメモリは 24 ビット幅しかありませんので、TBLRDH と TBLWTH 命令は、プログラムメモリの、存在しない上位バイトをアドレッシングすることができます。このバイトは ‘ファンタムバイト (phantom byte)’ と呼ばれます。ファンタムバイトを読んでも 0x00 が戻るだけであり、ファンタムバイトに書き込んでも影響はありません。

24 ビットのプログラムメモリは、並列した 2 つの 16 ビットの空間と見なされ、それぞれの空間は同じアドレス範囲を共有している点に、常に注意してください。従って、TBLRDL と TBLWTL 命令は ‘下位’ のプログラムメモリ空間 (PM<15:0>) をアクセスし、TBLRDH と TBLWTH 命令は ‘上位’ のプログラムメモリ空間 (PM<31:16>) をアクセスします。PM<31:24> への、どんな読み書きも、ファンタム (未実装) バイトへのアクセスになります。どのテーブル命令も、バイトモードで使用される時には、テーブルアドレスの LS ビットはバイト選択ビットとして使用されます。LS ビットは、プログラムメモリ空間の上位もしくは下位のどちらのバイトをアクセスするかを決定します。

図 5-1 にテーブル命令を使用して、プログラムメモリがどのようにアドレッシングされるかを示します。24 ビットのプログラムメモリアドレスは、テーブル命令で規定される、TBLPAG レジスタのビット <7:0> と W レジスタの実効アドレス (EA) で構成されます。24 ビットのプログラムカウンタを、図 5-1 に参考に示します。EA の上位 23 ビットはプログラムメモリ位置の選択に使用されます。バイトモードのテーブル命令では、W レジスタの EA の LS ビットは、16 ビットプログラムメモリワードのどのバイトがアドレスされるかを選択するために使用されます。‘1’ はビット <15:8>, ‘0’ はビット <7:0> を選択します。W レジスタの EA の LS ビットは、ワードモードのテーブル命令では無視されます。

プログラムメモリアドレスに加えて、テーブル命令は W レジスタの指定も行います。その W レジスタは、書き込まれるべきプログラムメモリデータのソースとなったり（もしくは、メモリ位置を指し示す W レジスタポインタ）、もしくはプログラムメモリ読み出し用の読み出し先となります。バイトモードでのテーブル書き込み動作では、ソースワーキングレジスタのビット <15:8> は無視されます。

図 5-1: テーブル命令のアドレッシング



## 5.2.1 テーブル読み出し命令を使用する

テーブル読み出しを行うには 2 つのステップが必要です。第一に、TBLPAG レジスタと W レジスタの 1 つを使用してアドレスポインタを設定します。それから、アドレス位置にあるプログラムメモリの内容を読み出します。

### 5.2.1.1 ワードモードでの読み出し

以下のコード例で、ワードモードでテーブル命令を使用してプログラムメモリのワードを読み出す方法を示します。

```
; Setup the address pointer to program space
MOV      #tblpage(PROG_ADDR),W0      ; get table page value
MOV      W0,TBLPAG                  ; load TBLPAG register
MOV      #tbloffset(PROG_ADDR),W0     ; load address LS word
; Read the program memory location
TBLRDH  [W0],W3                   ; Read high word to W3
TBLRDL  [W0],W4                   ; Read low word to W4
```

### 5.2.1.2 バイトモードでの読み出し

```
; Setup the address pointer to program space
MOV      #tblpage(PROG_ADDR),W0      ; get table page value
MOV      W0,TBLPAG                  ; load TBLPAG register
MOV      #tbloffset(PROG_ADDR),W0     ; load address LS word
; Read the program memory location
TBLRDH.B [W0],W3                   ; Read high byte to W3
TBLRDL.B [W0++],W4                 ; Read low byte to W4
TBLRDL.B [W0++],W5                 ; Read middle byte to W5
```

上記のコード例では、下位バイトの読み出し時に、ポスト増分オペレータにより、ワーキングレジスタ内のアドレスが 1 増分されます。これにより、3 番目の書き込み命令で中間バイト(middle byte)にアクセスするために EA<0> は ‘1’ に設定されます。最後のポスト増分により、W0 は偶数アドレスにもどり、次のプログラムメモリ位置を指します。

**注:** tblpage() と tbloffset() ディレクティブは、dsPIC30F 用の Microchip アセンブラーで提供されます。これらのディレクティブは、プログラムメモリアドレス値からテーブル命令用の TBLPAG と W レジスタの適切な値を選択します。詳しくは Microchip ソフトウェアツールを参照してください。

## 5.2.2 テーブル書き込み命令を使用する

テーブル書き込み命令の影響は、デバイスのプログラムメモアドレス空間に存在するメモリテクノロジのタイプに依存します。プログラムメモアドレス空間は、揮発性もしくは不揮発性プログラムメモリ、不揮発性データメモリ、および外部バスインターフェース (EBI) を含む場合があります。例えば、テーブル書き込み命令が EBI アドレス領域で発生したら、書き込みデータは、EBI データライン上に置かれます。

### 5.2.2.1 テーブル書き込み保持ラッチ

テーブル書き込み命令は、不揮発性プログラムメモリおよびデータメモリに直接書き込みを行うのではなく、書き込みデータを格納する保持ラッチに書き込みデータを転送します。保持ラッチはメモリ空間上にマッピングされておらず、テーブル命令を使用してのみアクセスされます。すべての保持ラッチにデータが転送されたら、特別な命令シーケンスを実行することにより、実際のメモリプログラム動作が開始されます。

保持ラッチの数は、プログラムされるメモリロックサイズの最大値を決定し、不揮発性メモリのタイプとデバイスの種類により変わります。例えば、保持ラッチの数は、当該デバイスの、プログラムメモリ、データ EEPROM メモリおよびデバイス構成レジスタにより異なります。

一般に、プログラムメモリは行とパネルに区切られます。1 つのパネルはそれ独自の、テーブル書き込み保持ラッチを一式持っています。これにより、一度に複数のメモリパネルにプログラム出来て、当該デバイスの全体のプログラミング時間を低減できます。1 つのメモリパネルには、メモリの 1 つの行を、一度にプログラムするのに十分な保持ラッチが在ります。メモリ制御論理は、テーブル書き込み命令で使用されるアドレスの値を基にして、どの書き込みラッチに転送するかを、自動的に決定します。

詳しくは、個別のデバイスデータシートを参照してください。

### 5.2.2.2 ワードモード書き込み

下記のシーケンスは、ワードモードで、1 つのプログラムメモリラッチの位置に書き込みを行う時に使用されます。

```
; プログラム空間へのアドレスポインタを設定します。
MOV      #tblpage(PROG_ADDR),W0      ; get table page value
MOV      W0,TBLPAG                  ; load TBLPAG register
MOV      #tbloffset(PROG_ADDR),W0    ; load address LS word
; 書き込みデータを W レジスタに転送します
MOV      #PROG_LOW_WORD,W2
MOV      #PROG_HI_BYTE,W3
; ラッチに転送するため、テーブル書き込みを実行します。
TBLWTL  W2,[W0]
TBLWTH  W3,[W0++]
```

この例では、W3 の上位バイトの内容は、ファントムバイト位置に書き込まれるので、関係ありません。W0 は、2 番目の TBLWTH 命令の後で 2だけポスト増分され、次のプログラムメモリ位置に書き込む準備をします。

## 5.2.2.3 バイトモード書き込み

バイトモードで、1つのプログラムメモリラッチの位置に書き込みを行う時には、下記のコードシケンスが使用されます。

```
; プログラム空間へのアドレスポインタを設定します。
MOV      #tblpage(PROG_ADDR),W0          ; get table page value
MOV      W0,TBLPAG                      ; load TBLPAG register
MOV      #tbloffset(PROG_ADDR),W0         ; load address LS word
; 書き込みデータをワーキングレジスタに転送します
MOV      #LOW_BYTE,W2
MOV      #MID_BYTE,W3
MOV      #HIGH_BYTE,W4
; ラッチに書き込みます。
TBLWTH.B W4,[W0]                      ; write high byte
TBLWTL.B W2,[W0++]                     ; write low byte
TBLWTL.B W3,[W0++]                     ; write middle byte
```

上記コード例では、下位バイトに書き込みを行う際のポスト増分により、W0 のアドレスが 1だけ増分されます。これにより、3番目の書き込み命令の中間バイトにアクセスするため EA<0> = 1 が設定されます。最後のポスト増分により、次のプログラムメモリ位置を指すために、W0 は偶数アドレスに設定されます。

## 5.3 コントロールレジスタ

FLASH とデータ EEPROM プログラミング動作は、以下の不揮発性メモリ (NVM) コントロールレジスタを使うことで制御されます。

- NVMCON: 不揮発性メモリコントロールレジスタ
- NVMKEY: 不揮発性メモリキーレジスタ
- NVMADR: 不揮発性メモリアドレスレジスタ

### 5.3.1 NVMCON レジスタ

NVMCON レジスタは FLASH や EEPROM のプログラム / 消去用の主なコントロールレジスタです。このレジスタは、FLASH もしくは EEPROM の選択、消去もしくはプログラム動作を行うかどうかの選択を行い、プログラムもしくは消去サイクルを開始するのに使用されます。

NVMCON レジスタの内容はレジスタ 5-1 に示されます。NVMCON の下位バイトは、実行すべき NVM 動作のタイプを構成します。簡単にするために、種々のプログラムや消去の動作用の NVMCON 設定値のサマリーを表 5-1 に示します。

表 5-1: NVMCON レジスタ値

RTSP プログラムと消去動作用の NVMCON レジスタ値			
メモリタイプ	動作	データサイズ	NVMCON 値
FLASH PM	消去	1 行 (32 命令語)	0x4041
	プログラム	4 命令	0x4001
Data EEPROM	消去	1 データワード	0x4044
		16 データワード	0x4045
	プログラム	1 データワード	0x4004
		16 データワード	0x4005
コンフィギュレーションレジスタ	書き込み(1)	1 コンフィギュレーションレジスタ	0x4008

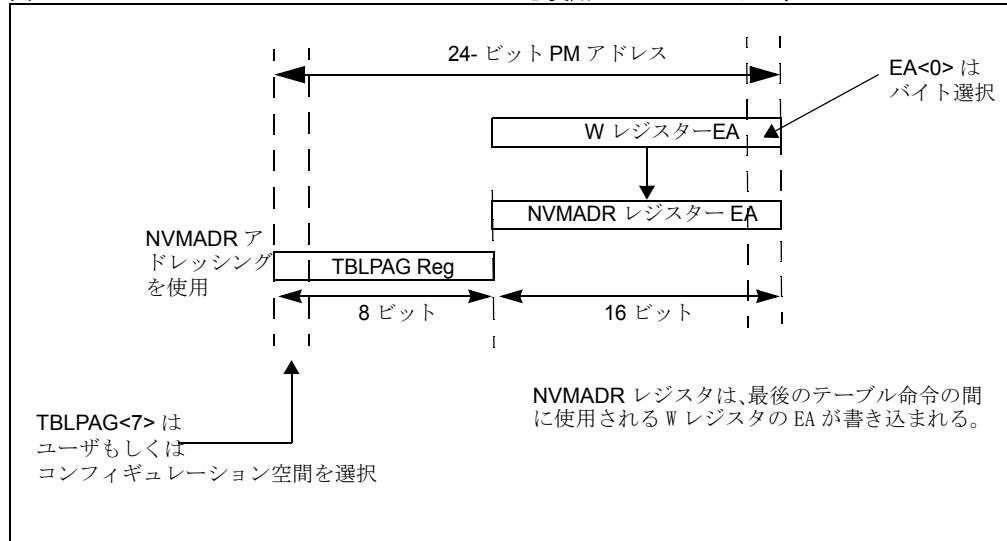
注 1: デバイスコンフィギュレーションレジスタは、消去サイクルを実行することなく、新しい値に書き換えられます。

## 5.3.2 NVMADR レジスタ

NVMADR レジスタは、プログラミングや消去動作用の実効アドレス (EA) の下位 16 ビットを保持するために使用されます。NVMADR レジスタは、最後に実行されたテーブル命令の EA<15:0>を取り込み、書き込み / 消去すべき FLASH もしくは EEPROM メモリの行を選択します。図 5-2 に、プログラミングや消去時にプログラムメモリ EA がどのように構成されるかを示します。

NVMADR レジスタはテーブル命令で自動的に転送されますが、ユーザーは、プログラミング動作が始まる前にその内容を直接修正することもできます。消去時にはテーブル書き込み命令は必要ないため、NVMADR への書き込みが、消去動作より前に必要となります。

図 5-2: TBLPAG と NVMADR レジスタを使用した NVM アドレッシング



## 5.3.3 NVMKEY レジスタ

NVMKEY は、FLASH もしくは EEPROM メモリの、偶発的な誤書き込みや誤消去を防止するために使用される、書き込み専用のレジスタです。プログラミングもしくは消去のロック解除シーケンスを開始するには、以下のステップを示されたとおりの順番で実行する必要があります。

1. NVMKEY に 0x55 を書き込む。
2. NVMKEY に 0xAA を書き込み。
3. 2つの NOP 命令を実行する。

このロック解除シーケンスの後に、1 命令サイクル分だけ、NVMCOM レジスタへの書き込みが許されます。ほとんどの場合、ユーザーは、NVMCOM レジスタ内の WR ビットをセットするだけで、プログラムもしくは消去サイクルを開始できます。割り込みは、ロック解除シーケンスの間は無効にしなければなりません。以下のコード例は、ロック解除シーケンスの実行方法を示しています。

```

PUSH      SR      ; 有効になっていれば、割り込みを無効にします。
MOV       #0x00E0,W0
IOR       SR

MOV       #0x55,W0
MOV       #0xAA,W0
MOV       W0,NVMKEY
MOV       W0,NVMKEY ; NOP は不要です。
BSET     NVMCON,#WR ; プログラム / 消去サイクルの開始
NOP
NOP
POP      SR      ; 割り込みを再度有効にします。

```

さらなるプログラミングの例については [セクション 5.4.2 「FLASH プログラミング動作」](#) を参照してください。

## 第5章 . FLASHとEEPROMのプログラミング

レジスタ 5-1: NVMCON: 不揮発性メモリコントロールレジスタ

上位バイト:							
R/S-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
WR	WREN	WRERR	-	-	-	-	-
ビット 15							ビット 8

下位バイト:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PROGOP<7:0>							
ビット 7							ビット 0

ビット WR: 書き込み(プログラムもしくは消去)制御ビット

15 1 = データ EEPROM もしくは FLASH の消去もしくは書き込み動作を起動します。  
(WR ビットはセットはできますが、ソフトウェアではクリアできません)。  
0 = 書き込みサイクルが終了した。

ビット WREN: 書き込み(消去もしくはプログラム)有効ビット

14 1 = 消去もしくはプログラム動作を有効にします。  
0 = 動作禁止。

ビット WRERR: FLASH エラーフラグビット

13 1 = 書き込み動作の異常終了(プログラム動作中の MCLR もしくは WDT リセット)  
0 = 書き込み動作が問題なく終了したことを示します。

ビット Reserved: これらの位置には 0 を書き込みます。

12-8

ビット PROGOP<7:0>: プログラム動作コマンドバイトのビットです。

7-0 消去動作:

0x41 = プログラム FLASH の 1 パネルから 1 行 (32 命令ワード) を消去します。  
0x44 = データ FLASH から 1 データワードを消去します。  
0x45 = データ FLASH から 1 行 (16 データワード) を消去します。

プログラミング動作:

0x01 = FLASH プログラムメモリに、8 分の 1 行 (4 命令ワード) をプログラムします。  
0x04 = データ EEPROM に 1 データワードをプログラムします。  
0x05 = データ EEPROM に 1 行 (16 データワード) をプログラムします。  
0x08 = デバイス構成レジスタに 1 データワードをプログラムします。

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' が読み出されます

S = 設定可能ビット      -n = POR の値      '1' = ビットがセットされます

'0' = ビットはクリアされま  
す      x = ビットは不定

## レジスタ 5-2: NVMADR: 不揮発性メモリアドレスレジスタ

上位バイト :							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
NVMADR<15:8>							
ビット 15	ビット 8						

下位バイト :							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
NVMADR<7:0>							
ビット 7	ビット 0						

ビット 15-0 **NVMADR<15:0>:** NV メモリ書き込みアドレスビット

プログラムもしくはデータ FLASH メモリ内のプログラムすべき位置を選択します。

このレジスタはユーザーにより読んだり書いたりできます。このレジスタは、ユーザーにより書き込まれるまでに実行された最後のテーブル書き込み命令の EA<15:0> のアドレスを含みます。

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' が読み出されます。
-n = POR の値	'1' = ビットがセット	'0' = ビットがクリア
	されます	x = ビットは不定です

## レジスタ 5-3: NVMKEY: 不揮発性メモリキーレジスタ

上位バイト :							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
ビット 15	ビット 8						

下位バイト :							
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
NVMKEY<7:0>							
ビット 7	ビット 0						

ビット 15-8 未実装 : '0' が読み出されます。

ビット 7-0 **NVMKEY<7:0>:** キーレジスタ（書き込み専用）ビット

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' が読み出されます。
-n = POR の値	'1' = ビットがセット	'0' = ビットがクリア
	されます	x = ビットは不定です

## 5.4 実行時セルフプログラミング (RTSP)

RTSPによりユーザーコードでFLASHプログラムメモリの内容を修正することができます。RTSPは、TBLRD(テーブル読み出し)とTBLWT(テーブル書き込み)命令、およびNVMコントロールレジスタを使用して行うことができます。RTSPを使用して、ユーザーはプログラムメモリの32命令(96バイト)を一度に消去でき、プログラムメモリデータの4命令(12バイト)を一度にプログラムできます。

### 5.4.1 RTSP動作

dsPIC30FのFLASHプログラムメモリは行とパネルとで構成されます。1つの行は32命令すなわち96バイトで構成されます。パネルのサイズはdsPIC30Fデバイスの種類に依存して変わります。詳しくはデバイスデータシートを参照してください。典型的には、1つのパネルは128行つまり4Kx24命令から構成されます。RTSPを使用して、ユーザーは1つの行(32命令)を一度に消去でき、4命令を一度にプログラムできます。

プログラムメモリのそれぞれのパネルには、プログラミングデータの4つの命令を保持する書き込みラッチがあります。これらのラッチはメモリマッピングされていません。ユーザーが書き込みラッチにアクセスする唯一の方法は、テーブル書き込み命令を使用することです。実際のプログラミング動作の前に、書き込みデータは、テーブル書き込み命令でパネル書き込みラッチに転送する必要があります。パネルにプログラムされるべきデータは、書き込みラッチに、命令1、命令2等のように、順次転送されます。転送される命令語は、常に4つのアドレス境界を持つ偶数グループ(すなわち、命令3, 4, 5, 6, を転送することは禁止)からでなければなりません。この要求を別の言い方をすると、4つの命令のプログラムメモリ開始アドレスは、下位3ビットが0に等しくなければなりません。すべての4つの書き込みラッチが、ラッチに保持されている古いデータに上書きされるために、プログラミング動作の度に全部書き込まれなければなりません。

RTSPプログラミング用の基本シーケンスは、テーブルポインタを設定し、それから書き込みラッチに転送するため、一連のTBLWT命令を実行します。プログラミングは、NVMCONレジスタ内の特別のビットを設定することで実行されます。4つの命令を転送するには、4つのTBLWTLと4つのTBLWTH命令が必要です。プログラムメモリの1つの消去された行を完全に再プログラムするには、4つのTBLWTLと4つのTBLWTHの8つのサイクルが必要です。複数の、連続しないプログラムメモリ領域をプログラムする場合は、それぞれの領域と次の複数書き込みラッチに書き込まれるように、その都度テーブルポインタを変更しなければなりません。

FLASHプログラムメモリへのすべてテーブル書き込み命令は1命令あたり2つの命令サイクルだけで完了します。なぜならテーブルラッチのみが書き込まれるからです。実際のプログラミング動作はNVMCONレジスタを使用することで起動されます。1行あたり、全部で8つのプログラミングパス(それぞれが4つの命令語を書き込みます)が必要です。128行のパネルには128の消去サイクルと1024のプログラミングサイクルが必要です。

### 5.4.2 FLASHプログラミング動作

プログラム/消去動作は、RTSPモードで内蔵FLASHプログラムメモリをプログラミングしたり消去したりするために必要です。プログラムもしくは消去動作は、デバイスにより自動的に時間が決まりますが通常は2msecの時間です。WRビット(NVMCON<15>)をセットすることで、動作が開始され、WRビットは、動作が終了すると自動的にクリアされます。

CPUはプログラミング動作が終了するまでストール(待つ)します。CPUはこの時間の間は、どの命令も実行しませんし、割り込みにも応答しません。プログラミングサイクル中に割り込みが発生した場合は、プログラミングサイクルが完了するまで待たれます。

## 5.4.2.1 FLASH プログラムメモリのプログラミングアルゴリズム

ユーザーはプログラム FLASH メモリを行(32 命令語)単位で消去できます。ユーザーは FLASH を4命令語のブロック毎にプログラムできます。一般的な手順は以下の通りです。

1. プログラム FLASH の1行(32命令語)を読み出し、データ“イメージ”としてデータ RAM に格納します。RAM イメージは偶数の 32 ワードプログラムメモリアドレス境界から読み出さなければなりません。
2. RAM データイメージを新しいプログラムメモリデータで更新します。
3. プログラム FLASH 行を消去します。
  - NVMCON レジスタを FLASH プログラムメモリの1行を消去するように設定します。
  - 消去すべき行のアドレスを NVMADR に書き込みます。
  - 割り込みを無効にします。
  - キーシーケンスを NVMKEY に書き込み、消去を有効にします。
  - WR ビットを設定します。これにより消去サイクルが開始されます。
  - CPU は消去サイクルの期間は待ち状態になります。
  - 消去サイクルが終了すると WR ビットはクリアされます。
  - 割り込みを再度有効にします。
4. RAM からの4つの命令語を FLASH プログラムメモリ書き込みラッチに書き込みます。
5. プログラム FLASH に4つの命令語をプログラムします。
  - NVMCON レジスタを FLASH プログラムメモリの1行をプログラムするように設定します。
  - 割り込みを無効にします。
  - キーシーケンスを NVMKEY に書き込み、プログラムサイクルを有効にします。
  - WR ビットを設定します。これによりプログラムサイクルが開始されます。
  - CPU はプログラムサイクルの期間は待ち状態になります。
  - プログラムサイクルが終了すると WR ビットはハードウェアによりクリアされます。
  - 割り込みを再度有効にします。
6. 行のプログラミングを終了させるためにステップ(4-5)をさらに7回繰り返します。
7. ステップ1から6を、必要分だけ繰り返し、必要な量の FLASH プログラムメモリのプログラムを行います。

**注:** ユーザーは、RTSP を使用して修正することのできるプログラムメモリの最小単位が 32 命令語分であることに注意してください。従ってその場所のイメージが、消去サイクルが起動される前に汎用 RAM に格納されていることが重要です。消去サイクルは、プログラミングが実行される前に、以前書き込まれた位置に対して実行する必要があります。

## 5.4.2.2 プログラムメモリの1行を消去する

以下のコードシーケンスは、プログラムメモリの1行(32命令)を消去するために使用されます。NVMCONレジスタはプログラムメモリの1行を消去するように構成します。TBLPAGとNVMADRレジスタには消去すべき行のアドレスが転送されます。プログラムメモリは、偶数行の領域で消去する必要があります。従って、NVMADRに書き込まれる値の下位6ビットは、行が消去される時には影響ありません。

消去動作は、WRコントロールビット(NVMCON<15>)をセットする前にNVMKEYレジスタに、特殊なロック解除もしくはキーシーケンスをNVMKEYレジスタに書き込むことにより起動されます。ロック解除シーケンスは、割り込みなく、まさに示された順序で実行する必要があります。従って、シーケンスの書き込み前に、割り込みを無効にしなければなりません。

CPUが動作を再開する時点で、2つのNOP命令を挿入しなければなりません。最後に(必要であれば)割り込みを有効にします。

```
; Flash プログラムメモリの1行を消去するために NVMCON を設定します。
    MOV      #0x4041, W0
    MOV      W0, NVMCON
; 消去すべき行へのアドレスポインタを設定します。
    MOV      #tblpage(PROG_ADDR), W0
    MOV      W0, TBLPAG
    MOV      #tbloffset(PROG_ADDR), W0
    MOV      W0, NVMADR
; 有効なら、割り込みを無効にします
    PUSH     SR
    MOV      #0x00E0, W0
    IOR      SR
; キーシーケンスを書き込みます
    MOV      #0x55, W0
    MOV      W0, NVMKEY
    MOV      #0xAA, W0
    MOV      W0, NVMKEY
; 消去動作を開始します。
    BSET    NVMCON, #WR
; (必須) 消去サイクルの後で、2つの NOP 命令を挿入します
    NOP
    NOP
; 必要なら、割り込みを再度有効にします。
    POP      SR
```

## 5.4.2.3 書き込みラッチへの転送

以下の命令シーケンスは、書き込みラッチに 96 ビット (4 命令語) を転送するために使用されます。テーブルポインタにより選択される書き込みラッチに転送するには 4 つの TBLWTL と 4 つの TBLWTH 命令が必要です。

TBLPAG レジスタにはプログラムメモリアドレスの上位 8 ビットが転送されます。FLASH プログラム動作としては、NVMADR レジスタへの書き込みは必要ありません。それぞれのテーブル書き込みが実行されると、プログラムメモリアドレスの下位 16 ビットは、自動的に NVMADR レジスタに取り込まれます。プログラムメモリは ‘偶数’ の 4 命令語アドレス領域でプログラムする必要があります。実際に、プログラム動作中は、NVMADR レジスタに取り込まれた値のうち下位 3 ビットは使用されません。

4 つの書き込みラッチのグループは、順番に書き込まれる必要はありません。テーブル書き込みアドレスの下位 3 ビットがどのラッチに書き込まれるかを決定します。但し、すべての 4 つのラッチが、旧データを上書きするために毎回のプログラミングサイクルで全部書き込まなければなりません。

**注：** 以下のコード例は、次の例で参照される ‘Load\_Write\_Latch’ コードです。

; 書き込まれるべき最初のプログラムメモリ位置へのポインタを設定します。

```
MOV    #tblpage(PROG_ADDR), W0  
MOV    W0, TBLPAG  
MOV    #tbloffset(PROG_ADDR), W0
```

; ラッチを書き込むために、TBLWT 命令を実行します。

; W0 は、TBLWTH 命令内で増加され、次の命令位置を指し示します。

```
MOV    #LOW_WORD_0, W2  
MOV    #HIGH_BYTE_0, W3  
TBLWTL W2, [W0]  
TBLWTH W3, [W0++]      ; 0th_program_word  
MOV    #LOW_WORD_1, W2  
MOV    #HIGH_BYTE_1, W3  
TBLWTL W2, [W0]  
TBLWTH W3, [W0++]      ; 1st_program_word  
MOV    #LOW_WORD_2, W2  
MOV    #HIGH_BYTE_2, W3  
TBLWTL W2, [W0]  
TBLWTH W3, [W0++]      ; 2nd_program_word  
MOV    #LOW_WORD_3, W2  
MOV    #HIGH_BYTE_3, W3  
TBLWTL W2, [W0]  
TBLWTH W3, [W0++]      ; 3rd_program_word
```

## 5.4.2.4 1行プログラミング例

```
1行のプログラミングコードの例は以下の通りです。
; プログラムメモリの複数ワードを書き込むために NVMCON を設定します。
MOV      #0x4001,W0
MOV      W0,NVMCON

; 全行（32命令）をプログラムするために、以下のコード部分を8回繰り返します。
; 4つのプログラムメモリ書き込みラッチに転送します。
CALL     Load_Write_Latch(1)
; 有効になっていれば、割り込みを無効にします。
PUSH    SR
MOV     #0x00E0,W0
IOR     SR
; キーシーケンスを書き込みます。
MOV     #0x55,W0
MOV     W0,NVMKEY
MOV     #0xAA,W0
MOV     W0,NVMKEY
; プログラミングシーケンスを開始します。
BSET   NVMCON,#WR
; プログラミングの後に2つの NOP を挿入します。
NOP
NOP
; 必要であれば、割り込みを再度有効にします。
POP    SR
```

注 1: セクション 5.4.2.3 「書き込みラッチへの転送」をご覧下さい。

## 5.4.3 デバイスコンフィギュレーションレジスタへの書き込み

RTSP はデバイスコンフィギュレーションレジスタへの書き込みにも使用されます。RTSP は、消去サイクルを最初に実行することなく、それぞれのコンフィギュレーションレジスタを個別に再書き込みすることができます。コンフィギュレーションレジスタは、システムクロックソース、PLL 通倍比や WDT の有効等、デバイスの重要な動作のパラメータを制御するので、書き込む時には注意が必要です。

デバイスコンフィギュレーションレジスタをプログラムする手順は FLASH プログラムメモリの場合の手順と似ていますが、TBLWTL 命令だけが必要な点が異なります。これは、それぞれのデバイスコンフィギュレーションレジスタでは上位 8 ビットが未使用であるからです。さらに、テーブル書き込みアドレスのビット 23 は、コンフィギュレーションレジスタをアクセスするためにセットされなければなりません。デバイスコンフィギュレーションレジスタの完全な説明については、第 24 章、「デバイスコンフィギュレーション」とデバイスのデータシートを参照してください。

### 5.4.3.1 コンフィギュレーションレジスタへの書き込みアルゴリズム

1. TBLWTL 命令を使用してテーブル書き込みラッチに新しいコンフィギュレーション値を書き込みます。
2. コンフィギュレーションレジスタ書き込みのために、NVMCON (NVMCON = 0x4008) を設定します。
3. 割り込みが有効になつていれば無効にします。
4. キーシーケンスを NVMKEY に書き込みます。
5. WR をセットする (NVMCON<15>) ことにより書き込みシーケンスを開始します。
6. 書き込みが終了すると CPU の実行が再開されます。
7. 必要であれば、割り込みを有効にします。

## 5.4.3.2 コンフィギュレーションレジスタ書き込みコード例

以下のコードシーケンスは、デバイスコンフィギュレーションレジスタを変更するために使用されます。

```
; 書き込むべき位置にポインタを設定します。
MOV      #tblpage(CONFIG_ADDR), W0
MOV      W0, TBLPAG
MOV      #tbloffset(CONFIG_ADDR), W0
; コンフィギュレーションレジスタに書き込むべき新しいデータを取得します。
MOV      #ConfigValue, W1
; 書き込みラッチに転送するためにテーブル書き込みを実行します。
TBLWTL  W1, [W0]
; コンフィギュレーションレジスタ書き込みのために、NVMCON を設定します。
MOV      #0x4008, W0
MOV      W0, NVMCON
; 有効になつていれば割り込みを無効にします。
PUSH    SR
MOV      #0x00E0, W0
IOR      SR
; キーシーケンスを書き込みます。
MOV      #0x55, W0
MOV      W0, NVMKEY
MOV      #0xAA, W0
MOV      W0, NVMKEY
; プログラミングシーケンスを開始します。
BSET    NVMCON, #WR
; プログラミングの後に 2 つの NOP を挿入します。
NOP
NOP
; 必要であれば、割り込みを再度有効にします。
POP    SR
```

## 5.5 データ EEPROM プログラミング

EEPROM ブロックは、プログラムメモリと同様に、テーブル読み出し・書き込み動作を使用することでアクセスできます。メモリが 16 ビット幅しかありませんので、EEPROM 動作には TBLWTH と TBLRDH 命令は必要ありません。データ EEPROM 用のプログラムと消去手順は、FLASH プログラムメモリに使用される手順と同様ですが、高速データアクセスに最適化されている点が異なります。以下のプログラミング動作がデータ EEPROM で実行されます。

- 1 ワード消去
- 1 行 (16 ワード) 消去
- 1 ワードプログラム
- 1 行 (16 ワード) プログラム

データ EEPROM は通常動作時(すべての V<sub>DD</sub>動作範囲)に読んだり書いたりできます。FLASH プログラムメモリとは異なり、EEPROM のプログラムもしくは消去動作中でも通常のプログラム実行は停止しません。

EEPROM の消去とプログラム動作は、NVMCON と NVMKEY レジスタを使用して実行されます。プログラミングソフトウェアで動作が完了するのを待つ必要があります。ソフトウェアは、以下の 3 つの方法のうち 1 つの方法で、EEPROM の消去もしくはプログラミングが完了したことを検出できます。

- ソフトウェアで WR ビット (NVMCON<15>) をポーリングします。WR ビットは動作が完了するとクリアされます。
- ソフトウェアで NVMIF ビット (IFS0<12>) をポーリングします。NVMIF ビットは動作が完了するとセットされます。
- NVM 割り込みを有効にします。CPU は動作が完了すると割り込みがかけられます。さらなるプログラミング動作は ISR で取り扱われます。

**注：** プログラミングもしくは消去動作中に EEPROM の読み出しを行うと、予期せぬ結果が発生することがあります。

## 5.5.1 EEPROMの1ワードプログラムアルゴリズム

1. 1 EEPROMワードの消去。
  - 1 EEPROMワードを消去するためにNVMCONレジスタを設定します。
  - 消去すべきワードのアドレスをTBLPAG, NVMADRレジスタに書き込みます。
  - NVMIFFのステータスピットをクリアし、NVM割り込み（オプション）を有効にします。
  - キーシーケンスをNVMKEYに書き込みます。
  - WRビットをセットします。これにより消去サイクルが開始されます。
  - WRビットをポーリングするかNVM割り込みを待ちます。
2. データEEPROM書き込みラッチにデータワードを書き込みます。
3. データワードをEEPROMにプログラムします。
  - 1 EEPROMワードをプログラムするためにNVMCONレジスタを設定します。
  - NVMIFFのステータスピットをクリアし、NVM割り込み（オプション）を有効にします。
  - キーシーケンスをNVMKEYに書き込みます。
  - WRビットをセットします。これによりプログラムサイクルが開始されます。
  - WRビットをポーリングするかNVM割り込みを待ちます。

## 5.5.2 EEPROMの行プログラミングアルゴリズム

複数のワードをEEPROMにプログラムする場合は、1度に16ワード（1行）を消去したりプログラムしたりすることで早くできます。EEPROMの16ワードをプログラムする手順は以下の通りです。

1. データEEPROMの1行（16ワード）を読み出し、データ“イメージ”としてデータRAMに格納します。変更すべきEEPROMのセクションは、偶数の16ワードアドレス境界になければなりません。
2. データイメージを新しいデータに更新します。
3. EEPROM行を消去します。
  - EEPROMの1行を消去するためにNVMCONを設定します。
  - NVMIFFのステータスピットをクリアし、NVM割り込み（オプション）を有効にします。
  - キーシーケンスをNVMKEYに書き込みます。
  - WRビットをセットします。これにより消去サイクルが開始されます。
  - WRビットをポーリングするかNVM割り込みを待ちます。
4. データEEPROM書き込みラッチに16のデータワードを書き込みます。
5. 1行をEEPROMにプログラムします。
  - EEPROMの1行をプログラムするためにNVMCONを設定します。
  - NVMIFFのステータスピットをクリアし、NVM割り込み（オプション）を有効にします。
  - キーシーケンスをNVMKEYに書き込みます。
  - WRビットをセットします。これによりプログラムサイクルが開始されます。
  - WRビットをポーリングするかNVM割り込みを待ちます。

## 5.5.3 データ EEPROM メモリの 1 ワードの消去

TBLPG と NVMADR レジスタに、消去すべきデータ EEPROM アドレスを転送します。EEPROM の 1 ワードがアクセスされるので、NVMADR の最下位ビットは消去動作中は影響を与えません。NVMCON レジスタは EEPROM メモリセットの 1 ワードを消去するように構成する必要があります。

WR 制御ビット (NVMCON<15>) をセットすることで、消去が起動されます。WR 制御ビットをセットする前に、特殊なロック解除もしくはキー シーケンスを NVMKEY レジスタに書き込む必要があります。ロック解除 シーケンスは、割り込みなく、まさに示された順序で実行する必要があります。従って、シーケンスの書き込み前に、割り込みを無効にしなければなりません。

```
; 消去すべき EEPROM 位置にポインタを設定します。
MOV      #tblpage(EE_ADDR),W0
MOV      W0,TBLPG
MOV      #tbloffset(EE_ADDR),W0
MOV      W0,NVMADR
; データ EEPROM の 1 ワードを消去するために NVMCON を設定します。
MOV      #0x4044,W0
MOV      W0,NVMCON
; KEY シーケンスが書き込まれる間に割り込みを無効にします。
PUSH    SR
MOV      #0x00E0,W0
IOR      SR
; KEY シーケンスを書き込みます。
MOV      #0x55,W0
MOV      W0,NVMKEY
MOV      #0xAA,W0
MOV      W0,NVMKEY
; 消去サイクルを開始します。
BSET    NVMCON,#WR
; 割り込みを再度有効にします。
POP      SR
```

## 5.5.4 データ EEPROMメモリに1ワードを書き込む

ユーザーが、プログラムされるべき EEPROM は消去済みと仮定して、1つの書き込みラッチに書き込むためにテーブル書き込み命令を使用します。TBLPAGE レジスタには、EEPROM アドレスの上位 8 ビットが転送されます。EEPROM アドレスの下位 16 ビットは、テーブル書き込み命令が実行される時に NVMADR レジスタに自動的に取り込まれます。NVMADR レジスタの最下位ビットは、プログラミング動作には影響を与えません。NVMCON レジスタは、データ EEPROM の 1 ワードをプログラムするように構成します。

WR 制御ビット (NVMCON<15>) をセットすることで、プログラミングが起動されます。WR 制御ビットをセットする前に、特殊なロック解除もしくはキーシーケンスを NVMKEY レジスタに書き込む必要があります。ロック解除シーケンスは、割り込みなく、まさに示された順序で実行する必要があります。従って、シーケンスの書き込み前に、割り込みを無効にしなければなりません。

```
; データ EEPROMへのポインタを設定します。
MOV      #tblpage(EE_ADDR),W0
MOV      W0,TBLPAGE
MOV      #tbloffset(EE_ADDR),W0
; 保持ラッチにデータ値を書き込みます。
MOV      EE_DATA,W1
TBLWTL  W1,[W0]
; NVMADR / TBLWTL 命令から書き込みアドレスを取り込みます。
; データ EEPROM に1ワードをプログラミングするために NVMCON を設定します。
MOV      #0x4004,W0
MOV      W0,NVMCON
; KEY シーケンスが書き込まれる間に割り込みを無効にします
PUSH    SR
MOV      #0x00E0,W0
IOR     SR
; キーシーケンスを書き込みます
MOV      #0x55,W0
MOV      W0,NVMKEY
MOV      #0xAA,W0
MOV      W0,NVMKEY
; 書き込みサイクルを開始します
BSET   NVMCON,#WR
; 必要なら再度割り込みを有効にします。
POP    SR
```

## 5.5.5 データ EEPROM の 1 行を消去する

NVMCON レジスタを EEPROM メモリの 1 行を消去するように構成します。TABPAG と NVMADR レジスタは、消去すべき行を指示する必要があります。データ EEPROM は偶数アドレス領域で消去される必要があります。従って、NVMADR の下位 5 ビットは、消去される行に影響を与えません。

WR 制御ビット (NVMCON<15>) をセットすることで、消去が起動されます。WR 制御ビットをセットする前に、特殊なロック解除もしくはキー シーケンスを NVMKEY レジスタに書き込む必要があります。ロック解除 シーケンスは、割り込みなく、まさに示された順序で実行される必要があります。従って、シーケンスの書き込み前に、割り込みを無効にしなければなりません。

```
; 消去すべき EEPROM の行を示すポインタを設定します。
MOV      #tblpage(EE_ADDR),W0
MOV      W0,TBLPAG
MOV      #tbloffset(EE_ADDR),W0
MOV      W0,NVMADR
; EEPROM の 1 行を消去するために NVMCON を設定します。
MOV      #0x4045,W0
MOV      W0,NVMCON
; KEY シーケンスが書き込まれる間に割り込みを無効にします。
PUSH    SR
MOV      #0x00E0,W0
IOR     SR
; KEY シーケンスを書き込みます。
MOV      #0x55,W0
MOV      W0,NVMKEY
MOV      #0xAA,W0
MOV      W0,NVMKEY
; 消去動作を開始します。
BSET    NVMCON,#WR
; 必要であれば割り込みを再度有効にします。
POP     SR
```

## 5.5.6 データ EEPROM メモリの1行を書き込む

データ EEPROM の1行を書き込むためには、プログラミングシーケンスを起動する前に、すべての16個の書き込みラッチに書き込まなければなりません。TBLPAGレジスタに、EEPROMアドレスの上位8ビットが転送されます。EEPROMアドレスの下位16ビットは、それぞれのテーブル書き込みが実行される時にNVMADRレジスタに自動的に取り込まれます。データEEPROMの行へプログラミングするときには、偶数アドレス領域で行われますので、NVMADRの下位5ビットは、プログラムされるべき行には影響を与えません。

WR制御ビット(NVMCON<15>)をセットすることで、プログラム動作が起動されます。WR制御ビットをセットする前に、特殊なロック解除もしくはキー シーケンスをNVMKEYレジスタに書き込む必要があります。ロック解除シーケンスは、割り込みなく、まさに示された順序で実行される必要があります。従って、シーケンスの書き込み前に、割り込みを無効にしなければなりません。

```
; プログラムすべき EEPROM の行へのポインタを設定します。
MOV      #tblpage(EE_ADDR),W0
MOV      W0,TBLPAG
MOV      #tbloffset(EE_ADDR),W0
; プログラミングラッチにデータを書き込みます
MOV      data_ptr,W1      ; W1 をデータのポインタとして使用。
TBLWTL [W1++], [W0++]    ; 1番目のデータ書き込み
TBLWTL [W1++], [W0++]    ; 2番目のデータ書き込み
TBLWTL [W1++], [W0++]    ; 3番目のデータ書き込み
TBLWTL [W1++], [W0++]    ; 4番目のデータ書き込み
TBLWTL [W1++], [W0++]    ; 5番目のデータ書き込み
TBLWTL [W1++], [W0++]    ; 6番目のデータ書き込み
TBLWTL [W1++], [W0++]    ; 7番目のデータ書き込み
TBLWTL [W1++], [W0++]    ; 8番目のデータ書き込み
TBLWTL [W1++], [W0++]    ; 9番目のデータ書き込み
TBLWTL [W1++], [W0++]    ; 10番目のデータ書き込み
TBLWTL [W1++], [W0++]   ; 11番目のデータ書き込み
TBLWTL [W1++], [W0++]   ; 12番目のデータ書き込み
TBLWTL [W1++], [W0++]   ; 13番目のデータ書き込み
TBLWTL [W1++], [W0++]   ; 14番目のデータ書き込み
TBLWTL [W1++], [W0++]   ; 15番目のデータ書き込み
TBLWTL [W1++], [W0++]   ; 16番目のデータ書き込み
; NVMADRは最後のテーブルアクセスアドレスを取り込みます。
; EEPROM の1行を書き込むためにNVMCONを設定します。
MOV      #0x4005,W0
MOV      W0,NVMCON
; KEY シーケンスが書き込まれる間に割り込みを無効にします。
PUSH    SR
MOV      #0x00E0,W0
IOR     SR
; KEY シーケンスを書き込みます。
MOV      #0x55,W0
MOV      W0,NVMKEY
MOV      #0xAA,W0
MOV      W0,NVMKEY
; プログラミング動作を開始します。
BSET    NVMCON,#WR
; 必要であれば、割り込みを再度有効にします。
POP     SR
```

**注：** このコードセグメントでは、例をわかり易くするために、16個のテーブル書き込み命令が使用されています。REPEATループでテーブル書き込み命令を使用することで、コードセグメントを簡単にできます。

## 5.5.7 データ EEPROM メモリを読み出す

TBLRD 命令は現在のプログラムワードアドレスのワードを読み出します。この例では W0 をデータ FLASH へのポインタとして使用しています。結果は W4 レジスタに反映されます。

```
; EEPROM メモリへのポインタを設定します。  
MOV      #tblpage(EE_ADDR),W0  
MOV      W0,TBLPAG  
MOV      #tbloffset(EE_ADDR),W0  
; EEPROM データを読み出します。  
TBLRDL [W0],W4
```

**注：** プログラム空間可視化 (PSV) はプログラムメモリアドレス空間内の位置を読み出すためにも使用できます。PSVについての詳しい情報は第4章。「プログラムメモリ」を参照してください。

## 5.6 設計の秘訣

**質問 1 :** デバイスを正常にプログラム、消去できません。私のコードは正しいように見えます。原因は何でしょうか？

**回答：**キーシーケンスが割り込み無しで実行されることを確実にするために、プログラムもしくは消去を起動する時は割り込みを禁止する必要があります。使用中の CPU の優先度をレベル 7 に上げることにより、割り込みを禁止することができます。この章でのコード例では、スタックの現状の SR レジスタの値を保存し、それから SR と 0x00E0 の論理和を取り  $IPL<2:0> = 111$  することにより割り込みを禁止しています。優先度 7 の割り込みが有効でない場合は、キーシーケンスが実行されている間は、別の方として DISI 命令により一時的に割り込みを禁止できます。

**質問 2 :** テーブル命令を使用しないで、簡単にデータ EEPROM を読み出す方法は何ですか？

**回答：**データ EEPROM はプログラムメモリ空間にマッピングされています。PSV はデータメモリ空間に EEPROM 領域をマッピングするために使用できます。PSVに関するより詳しい情報については第4章。「プログラムメモリ」を参照してください。

## 5.7 関連するアプリケーションノート

この章では、マニュアルのこの章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正して使用できるし、制限がある場合もあります。現状、FLASH および EEPROM プログラミングモジュールに関連するアプリケーションノートは以下の通りです。

題目	アプリケーションノート #
現在のところ、関連するアプリケーションノートはありません。	

**注：** dsPIC30F ファミリーのデバイスに関しての、他のアプリケーションノートやコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 5.8 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

この版は、dsPIC30F の FLASH および EEPROM プログラミングモジュールに関する技術内容の変更を含んでいます。



## 第6章. 割り込み

### ハイライト

この章は、以下の項目を含んでいます。

6.1	序章	.....	6-2
6.2	マスクできないトラップ	.....	6-6
6.3	割り込み処理タイミング	.....	6-11
6.4	割り込み用制御、ステータスレジスタ	.....	6-14
6.5	割り込み設定手順	.....	6-42
6.6	設計の秘訣	.....	6-44
6.7	関連するアプリケーションノート	.....	6-45
6.8	改訂履歴	.....	6-46

## 6.1 序章

dsPIC30F の割り込み制御モジュールは多くの周辺割り込み要求信号を、dsPIC30F CPU への 1 つの割り込み要求信号に減らし、以下のような特徴を持っています。

- 最大 8 個のプロセッサ例外とソフトウェアトラップを持ちます。
- 7 つの選択可能な優先度を持ちます。
- 最大 62 ベクトルまでの割り込みベクトル表 (IVT) を持ちはます。
- 各々の割り込みもしくは例外ソース用の一意のベクトルを持ちます。
- 指定ユーザー優先度内での固定優先度を持ちます。
- デバッグサポート用に代替の割り込みベクトル表 (AIVT) を持ちはます。
- 割り込みのエントリと戻りの遅れ時間は一定です。

### 6.1.1 割り込みベクトル表

割り込みベクトル表 (IVT) を図 6-1 に示します。IVT はプログラムメモリに存在し、開始位置は 0x000004 です。IVT は、8 つのマスクできないトラップベクトルと最大 54 までの割り込みソースから構成される 62 のベクトルを持ちます。一般的に、それぞれの割り込みソースはそれぞれのベクトルを持ちます。それぞれの割り込みベクトルは 24 ビット幅のアドレスを持ちます。それぞれの割り込みベクトル位置へプログラムされる値は、それに連動する割り込みサービスルーチン (ISR) の開始アドレスです。

### 6.1.2 代替のベクトル表

代替のベクトル表 (AIVT) は図 6-1 に示すように IVT の後に位置します。AIVT へのアクセスは、ALTIIVT 制御レジスタ (INTCON2<15>) により与えられます。ALTIIVT ビットがセットされると、すべての割り込みと例外処理は、デフォルトのベクトルではなく代替のベクトルを使用します。代替のベクトルはデフォルトベクトルと同じように構成されます。

AIVT は、再プログラムされるべき割り込みベクトルを必要とせずに、アプリケーションとサポート環境の間をスイッチする方法を与えることによりエミュレーションとデバッグ作業をサポートします。この特徴により、実行用と評価用と異なるソフトウェアアルゴリズムとして、アプリケーションをスイッチできます。AIVT が不要であれば、IVT 内で使用される同じアドレスを AIVT にプログラムします。

### 6.1.3 リセットシーケンス

デバイスリセットは、割り込みコントローラがリセットプロセス内には関わっていませんので、真の例外ではありません。dsPIC30F デバイスは PC をゼロにするリセットに反応してレジスタをクリアします。プロセッサは、その後、アドレス 0x000000 からプログラムの実行を開始します。ユーザーは、RESET アドレスに GOTO 命令を記述して、適切なスタートアップルーチンを起動するようにします。

**注：** IVT と AIVT 内の未実装もしくは未使用のベクトル位置には、RESET 命令を含むデフォルトの割り込みハンドラルーチンのアドレスをプログラムする必要があります。

図 6-1: 割り込みベクトル表

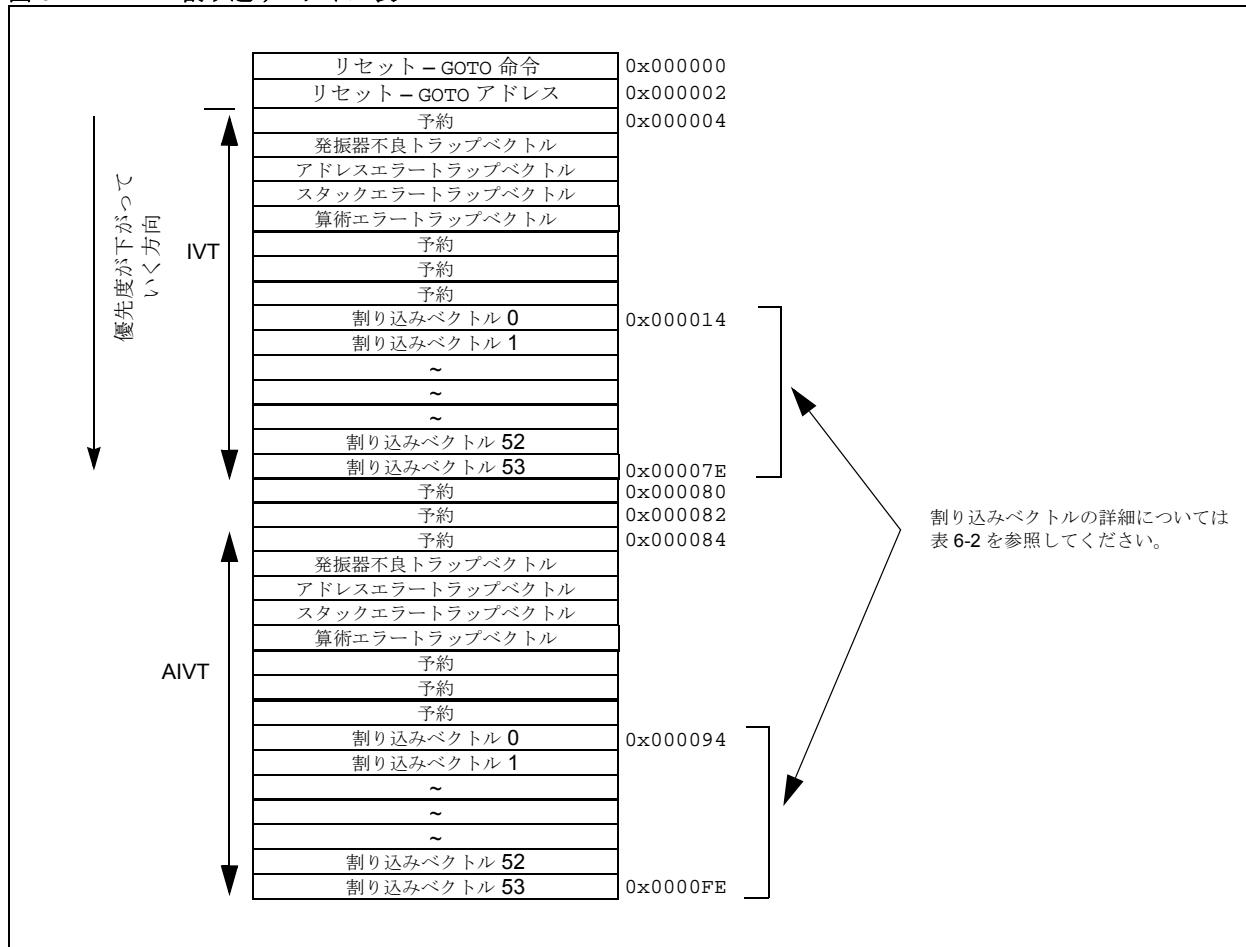


表 6-1: トラップベクトルの詳細

ベクトル番号	IVT アドレス	AIvt アドレス	トラップソース
0	0x000004	0x000084	予約
1	0x000006	0x000086	発振器不良
2	0x000008	0x000088	アドレスエラー
3	0x00000A	0x00008A	スタックエラー
4	0x00000C	0x00008C	算術エラー
5	0x00000E	0x00008E	予約
6	0x000010	0x000090	予約
7	0x000012	0x000092	予約

表 6-2: 割り込みベクトルの詳細

ベクトル番号	IVT アドレス	AIVT アドレス	割り込みソース
8	0x000014	0x000094	INT0 – 外部割り込み 0
9	0x000016	0x000096	IC1 – 入力キャプチャ 1
10	0x000018	0x000098	OC1 – 出力コンペア 1
11	0x00001A	0x00009A	T1 – タイマー 1
12	0x00001C	0x00009C	IC2 – 入力キャプチャ 2
13	0x00001E	0x00009E	OC2 – 出力コンペア 2
14	0x000020	0x0000A0	T2 – タイマー 2
15	0x000022	0x0000A2	T3 – タイマー 3
16	0x000024	0x0000A4	SPI1
17	0x000026	0x0000A6	U1RX – UART1 受信
18	0x000028	0x0000A8	U1TX – UART1 送信
19	0x00002A	0x0000AA	ADC – ADC 変換完了
20	0x00002C	0x0000AC	NVM – NVM 書き込み完了
21	0x00002E	0x0000AE	I <sup>2</sup> C スレーブ動作 – メッセージ検出
22	0x000030	0x0000B0	I <sup>2</sup> C マスター動作 – メッセージイベント完了
23	0x000032	0x0000B2	ポート変化割り込み
24	0x000034	0x0000B4	INT1 – 外部割り込み 1
25	0x000036	0x0000B6	IC7 – 入力キャプチャ 7
26	0x000038	0x0000B8	IC8 – 入力キャプチャ 8
27	0x00003A	0x0000BA	OC3 – 出力コンペア 3
28	0x00003C	0x0000BC	OC4 – 出力コンペア 4
29	0x00003E	0x0000BE	T4 – タイマー 4
30	0x000040	0x0000C0	T5 – タイマー 5
31	0x000042	0x0000C2	INT2 – 外部割り込み 2
32	0x000044	0x0000C4	U2RX – UART2 受信
33	0x000046	0x0000C6	U2TX – UART2 送信
34	0x000048	0x0000C8	SPI2
35	0x00004A	0x0000CA	CAN1
36	0x00004C	0x0000CC	IC3 – 入力キャプチャ 3
37	0x00004E	0x0000CE	IC4 – 入力キャプチャ 4
38	0x000050	0x0000D0	IC5 – 入力キャプチャ 5
39	0x000052	0x0000D2	IC6 – 入力キャプチャ 6
40	0x000054	0x0000D4	OC5 – 出力コンペア 5
41	0x000056	0x0000D6	OC6 – 出力コンペア 6
42	0x000058	0x0000D8	OC7 – 出力コンペア 7
43	0x00005A	0x0000DA	OC8 – 出力コンペア 8
44	0x00005C	0x0000DC	INT3 – 外部割り込み 3
45	0x00005E	0x0000DE	INT4 – 外部割り込み 4
46	0x000060	0x0000E0	CAN2
47	0x000062	0x0000E2	PWM – PWM 周期一致
48	0x000064	0x0000E4	QE1 – 位置カウンタ比較
49	0x000066	0x0000E6	DCI – Codec 転送完了
50	0x000068	0x0000E8	LVD – 低電圧検出
51	0x00006A	0x0000EA	FLTA – MCPWM 不良 A
52	0x00006C	0x0000EC	FLTB – MCPWM 不良 B
53-61	0x00006E-0x00007E	0x00006E-0x00007E	予約

### 6.1.4 CPU プライオリティーステータス

CPU は 0 ~ 15 の 16 の優先度のうち 1 つを持ちます。例外処理を起動するには、割り込みもしくはトラップソースは、現状の CPU 優先度より高い優先度を持たねばなりません。周辺や外部の割り込みソースはレベル 0 ~ 7 でプログラムされ、レベル 8 ~ 15 の CPU 優先度は、トラップソース用に予約されています。トラップは、ハードウェアやソフトウェアの問題を検出する目的を持つ、マスク不可の割り込みソースです（セクション 6.2 「マスクできないトラップ」参照）。それぞれのトラップソースの優先度は固定され、1 つのトラップのみが各優先度に割り当てられます。優先度 0 にプログラムされる割り込みソースの優先度は CPU より大きくならないので、効果的に無効になることに注意してください。

現状 CPU の優先度は以下の 4 つのステータスピットにより示されます。

- SR<7:5> に位置する IPL<2:0> ステータスピット
- CORCON<3> に位置する IPL3 ステータスピット

IPL<2:0> ステータスピットは読み書き可能です。従って、これらのビットを変更して、与えられた優先度以下の条件で、割り込みのすべてのソースを無効にできます。例えば、IPL<2:0> = 3 の場合、優先度 0、1、2 もしくは 3 にプログラムされたソースからは、CPU に割り込むことができません。

トラップイベントは、他のどのユーザー割り込みソースよりも高い優先度を持っています。IPL3 ビットがセットされると、トラップイベントが開始されます。IPL3 ビットは、ユーザーによってクリアできますが、セットすることはできません。いくつかのアプリケーションでは、トラップが発生し、元々のトラップの発生原因になった命令以外の命令に分岐する場合には、IPL3 をクリアしたほうがよい場合があります。

すべてのユーザー割り込みソースは、IPL<2:0> = 111 に設定することで禁止されます。

**注：** IPL<2:0> ビットは、割り込みネスティングが無効になると、読み込みのみとなります。詳しくは、セクション 6.2.4.2 「割り込みのネスティング」を参照してください。

### 6.1.5 割り込み優先度

それぞれの周辺割り込みソースは、7 つの優先度のいずれかに割り当てられます。個々の割り込みに対して、ユーザーが割り当て可能な割り込みの優先度制御ビットは、IPCx レジスタ内のそれぞれのニブルの下位 3 ビット内にあります。それぞれのニブルのビット 3 は使用されず、「0」として読み込まれます。これらのビットは特定の割り込みに割り当てられた優先度を決定します。使用できる優先度は最低レベルとしての「1」から最高レベルとしての 7 まであります。割り込みソースに対応する IPC ビットがすべてクリアされると、割り込みソースは割り込み禁止になります。

1 つ以上の割り込み要求ソースが特定の優先度に割り当てられるので、ユーザーが割り当てたレベル内で優先度の衝突を解決するための方法が準備されています。それぞれの割り込みのソースは IVT 内の位置の順序に基づく優先度（自然優先度）を持ちます。表 6-2 に IVT 内の割り込みソースの位置を示します。小さい番号を与えた割り込みベクトルがより高い自然優先度を持ち、大きい番号を与えた割り込みベクトルがより低い自然優先度を持ちます。ある割り込みの優先度は、まず、IPCx レジスタ内でユーザーがそのソースに割り当てた優先度により決定され、それから IVT 内の自然優先度により決定されます。

自然優先度は、ユーザーが割り当てた同じ優先度を持つ割り込みが同時に発生した場合の競合を解決する場合にのみ使用されます。優先度の競合が解決し、例外処理が始まったら、CPU は、ユーザーが割り当てた優先度の高いソースの場合のみ、さらに割り込みがかけられます。同じユーザー優先度を持つが、高い自然順序に基づく優先度を持つ割り込みは、例外処理が始まった後では待ち状態になりますが、現状の割り込み処理が完了するまで、待ったままでです。

ユーザーが、7 つの優先度の 1 つに割り込みを割り当てる能力は、ユーザーが、低い自然優先度を持つ割り込みに、全体として非常に高い優先度を与えられることを意味します。例えば、PLVD（プログラマブル低電圧検出）には、優先度 7 が与えられ、INT0（外部割り込み 0）は優先度 1 に割り当てられるとすると INT0 には非常に低い優先度を与えることになります。.

**注：** 周辺装置と、IVT 内で利用可能な割り込みソースは、特定の dsPIC30F デバイスにより変わります。このドキュメント内で示される割り込みのソースは、dsPIC30F デバイス内の割り込みソースに関する包括的なリストを意味します。詳しくは、それぞれのデバイスのデータシートを参照してください。

## 6.2 マスクできないトラップ

トラップは、マスクできない、ネスティングできる割り込みと見なされ、固定の優先度構造に割り付けられています。トラップは、デバッグやアプリケーション内の動作時に、誤った動作を訂正する方法をユーザーに提供することを目的としています。ユーザーが、トラップエラー状態のイベントが発生しても訂正処理をするつもりが無い場合は、これらのベクトルには、デバイスのリセットを行うソフトウェアルーチンのアドレスを設定する必要があります。この設定を行なわない場合には、トラップベクトルにトラップ状態を修正するサービスルーチンのアドレスをプログラムしなくてはなりません。

dsPIC30F はマスクできないトラップの 5 つのソースを持ちます。

- 発振器不良トラップ
- スタックエラートラップ
- アドレスエラートラップ
- 算術エラートラップ

これらのトラップ状態の多くは、それらが発生したときのみに検出可能であることに注意してください。その結果、トラップを発生する命令は、例外処理が始まる前に完了することになります。従ってユーザーは、このトラップを発生させた命令の実行結果を修正しなければなりません。

それぞれのトラップソースは、IVT 内の位置により決定される固定の優先度を持っています。発振器不良トラップは最高の優先度を持っており、算術エラートラップは最低の優先度を持っています（図 6-1 参照）。さらにトラップソースは、「ハード」トラップと、「ソフト」トラップの、2 つの明確なカテゴリに分類されます。

### 6.2.1 ソフトトラップ

算術エラートラップ（優先度 11）とスタックエラートラップ（優先度 12）は「ソフト」トラップソースとして分類されます。ソフトトラップは IVT 内の位置により割り当てられる優先度のマスクできない割り込みソースのように扱われます。ソフトトラップは割り込みと同様に処理され、例外処理の前に、サンプリングされて認識されるまでに 2 サイクル必要です。通常の割り込みと同じ扱いですから、ソフトトラップに応答する前に追加の命令を実行できます。

#### 6.2.1.1 スタックエラートラップ（ソフトトラップ、レベル 12）

スタックは、RESET の間に 0x0800 に初期化されます。スタックエラートラップはスタックポインタアドレスが 0x0800 より小さくなった場合に発生します。

スタックポインタに連携する Stack Limit レジスタ (SPLIM) があり、このレジスタはリセットで初期化されません。SPLIM へのワード書き込みが発生するまで、スタックオーバーフローチェックは有効になりません。

ポインタのソースもしくは対象として、W15 を用いて生成されるすべての実効アドレス (EA) は、SPLIM 内の値と比較されます。EA が SPLIM レジスタの内容より大きい場合は、スタックエラートラップが発生します。EA 計算結果が、データ空間の最後 (0xFFFF) を越えた場合にもスタックエラートラップが発生します。

スタックエラーは、STKERR ステータスピット (INTCON1<2>) をポーリングすることによりソフトウェアで検出できます。トラップサービスルーチンに再度入らないようにするために、RETFIE 命令によりトラップから戻る前に、STKERR ステータスフラグはソフトウェアでクリアされねばなりません。

### 6.2.1.2 算術エラートラップ（ソフトトラップ、レベル11）

以下のイベントのいずれかにより算術エラートラップが発生します。

- アキュムレータ A のオーバーフロー
- アキュムレータ B のオーバーフロー
- アキュムレータの壊滅的オーバーフロー
- ゼロによる割り算
- +/-16 ビットを越えるシフトアキュムレータ (SFTAC) 動作

INTCON1 レジスタ内には 3 つの有効ビットがあり、それらは 3 種類のアキュムレータオーバーフロートラップを有効にします。OVATE 制御ビット (INTCON1<10>) は、アキュムレータ A のオーバーフローイベント用のトラップを有効にするために使用されます。OVBTE 制御ビット (INTCON1<9>) は、アキュムレータ B のオーバーフローイベント用のトラップを有効にするために使用されます。COVTE 制御ビット (INTCON1<8>) は、どちらかのアキュムレータの壊滅的オーバーフロー用のトラップを有効にするために使用されます。

アキュムレータ A もしくはアキュムレータ B オーバーフローイベントはビット 31 からのキャリーアウトとして定義されます。アキュムレータに対して 31 ビット飽和モードが有効になっている場合は、アキュムレータオーバーフローは発生しないことに注意してください。壊滅的アキュムレータオーバーフローは、どちらかのアキュームレータのビット 39 からのキャリーアウトとして定義されます。アキュームレータ飽和 (31- ビットもしくは 39- ビット) が有効の場合は、壊滅的オーバーフローは発生しません。

ゼロによる割り算トラップは無効にできません。ゼロによる割り算チェックは、割り算命令を実行する REPEAT ループの最初の繰り返しの間に実行されます。

アキュームレータシフトトラップは無効にできません。SFTAC 命令は、アキュームレータをリテラル値もしくはいずれかの W レジスタの値分だけアキュームレータをシフトする場合に使用されます。シフト値が +/-16 ビットを越えた場合、算術トラップが発生します。SFTAC 命令は実行されますが、シフトの結果はターゲットのアキュームレータには書き込まれません。

算術エラートラップは、MATHERR ステータスビット (INTCON1<4>) をポーリングすることによりソフトウェアで検出できます。トラップサービスルーチンに再度入らないようにするために、RETFIE 命令によりトラップから戻る前に、MATHERR ステータスビットはソフトウェアでクリアされねばなりません。MATHERR ステータスビットをクリアする前に、トラップを発生させるすべての状態もクリアする必要があります。トラップがアキュームレータのオーバーフローによる場合、OA と OB ステータスビット (SR<15:14>) をクリアする必要があります。OA と OB ステータスビットは読み込み専用ですので、ハードウェアの OA もしくは OB ステータスビットをクリアするためには、ユーザーソフトウェアによりオーバーフローしたアキュームレータに ('0' を加える等の) ダミー動作を行う必要があります。

### 6.2.2 ハードトラップ

ハードトラップにはレベル 13 から 15 までの優先度の例外を含みます。スタックエラー (レベル 13) と発振器エラー (レベル 14) トラップがこのカテゴリに入ります。

ソフトトラップと同様に、ハードトラップは、マスクできない割り込みのソースと見なされます。ハードトラップとソフトトラップの違いは、トラップを発生させる命令が完了した後に、ハードトラップは CPU のコード実行を停止させることです。通常のプログラム実行フローは、トラップに応答があり処理が実行されるまで、再開されません。

#### 6.2.2.1 トラップの優先度とハードトラップの衝突

低い優先度のトラップを実行中に高い優先度のトラップが発生した場合は、低い優先度のトラップは中断され、高い優先度のトラップへの応答と処理が実行されます。低い優先度のトラップは、高い優先度のトラップの処理が完了するまで中断状態を維持します。

ハードトラップが発生したら、どのようなタイプのコード実行より先にアクノレッジされねばなりません。高い優先度のトラップが中断中、応答中もしくは処理中に低い優先度のハードトラップが発生した場合は、ハードトラップの衝突が発生します。この衝突は、低い優先度のトラップが、高い優先度のトラップの処理が完了するまで認識されないことにより発生します。

ハードトラップの衝突状態になるとデバイスは自動的にリセットされます。リセットが発生すると、TRAPR ステータスビット (RCON<15>) がセットされ、ソフトウェアで状態が検出できます。

## 6.2.2.2 発振器不良トラップ（ハードトラップ、レベル 14）

発振器不良トラップイベントは、以下のどれかの理由により発生します。

- フェイル・セーフクロックモニター（FSCM）が有効で、システムクロックソースが停止したことを検出した。
- PLL を使用した通常動作中に PLL のロックが外れたことを検出した。
- FSCM が有効で、パワーオンリセット（POR）で PLL をロックできなかった。

発振器不良トラップイベントは、OSCFAIL ステータスビット（INTCON1<1>）、もしくは CF ステータスビット（OSCCON<3>）をポーリングすることによりソフトウェアで検出できます。トラップサービスルーチンに再度入らないようにするために、RETFIE 命令によりトラップから戻る前に、OSCFAIL ステータスフラグはソフトウェアでクリアされねばなりません。

FSCM について詳しくは、**第 7 章.「発振器」と第 24 章.「デバイスコンフィギュレーション」**を参照してください。

## 6.2.2.3 アドレスエラートラップ（ハードトラップ、レベル 13）

以下にアドレスエラートラップを発生させる動作について説明します。

1. 誤った配置のデータワードのフェッチが行われた時。この状態は、命令が、実効アドレスの最下位ビットが 1 にセットされた状態でワードアクセスを行った時に発生します。dsPIC30F の CPU は、すべてのワードアクセスにおいては、偶数領域に配置されている必要があります。
2. 最下位ビットが 1 にセットされた実効アドレスで間接アドレッシングモードを用いて、ビット操作命令を実行した時。
3. 未実装のデータアドレス空間からデータフェッチを行おうとした時。

データ空間への書き込みは、アドレスエラートラップが発生したときは禁止され、データが破壊されないようにします。

データアドレスエラーは、ADDRERR ステータスビット（INTCON1<3>）をポーリングすることによりソフトウェアで検出できます。トラップサービスルーチンに再度入らないようにするために、RETFIE 命令によりトラップから戻る前に、ADDRERR ステータスフラグはソフトウェアでクリアされねばなりません。

**注：** MAC クラス命令では、データ空間は X と Y 空間に分割されています。これらの命令では、X 空間の未実装アクセスには Y 空間も含まれ、Y 空間の未実装アクセスには X 空間も含まれます。

## 6.2.3 割り込み命令を禁止にする

DISI（割り込み禁止）命令は、最大 16384 命令サイクルまで割り込みを禁止できます。この命令は、タイムクリティカルなコードセグメントを実行しなければならない時に役に立ちます。

DISI 命令は、優先度 1～6 の割り込みを禁止にするだけです。優先度 7 の割り込みとすべてのトラップイベントは、DISI 命令が実行中でも、CPU への割り込みができます。

DISI 命令は、DISICNT レジスタと一緒に動作します。DISICNT レジスタがゼロでない時には、優先度 1-6 の割り込みは禁止になります。DISICNT レジスタは、後に続く命令サイクル毎に減分されます。DISICNT レジスタが ‘0’ までカウントダウンされると、優先度 1～6 の割り込みは再び有効になります。DISI 命令サイクルには、PSV アクセス、命令ストール等によるすべてのサイクルを含みます。

DISICNT レジスタは読み書き可能です。ユーザーは、DISICNT レジスタをクリアすることにより、前の DISI 命令の影響を早く終了させることができます。DISICNT に書き込むか値を追加することで、割り込みが禁止である時間の総量を増加させることができます。

DISICNT レジスタがゼロの場合は、非ゼロの値をレジスタに書き込むだけでは割り込みを禁止することはできないことに注意してください。割り込みは、DISI 命令を用いて、最初に禁止にしなければなりません。DISI 命令が実行され、DISICNT が非ゼロの値を持つと、DISICNT の内容を変更することで割り込み禁止時間を延長できます。

**注：** DISICNT レジスタをソフトウェアで変更することは推奨できません。

DISI ステータスビット（INTCON2<14>）は、DISI 命令の結果として割り込みが禁止になっている時はいつでもセットされています。

**注：** CPU 優先度 7 にソースが設定されていない場合、DISI 命令は、すべての割り込みソースを即時禁止するために使用できます。

## 6.2.4 割り込み動作

すべての割り込みイベントフラグは、命令サイクル毎にサンプリングされます。待ち合わせ中の割り込み要求 (IRQ) は、IFSx レジスタ内で、フラグビット = '1' により表示されます。IRQ は、割り込みイネーブル (IECx) レジスタ内の対応するビットがセットされている場合に、割り込みを発生させます。IRQ がサンプリングされた命令サイクルの残りの時間で、待ち合わせ中のすべての割り込み要求の優先度が評価されます。

CPU が IRQ を認識しても命令は中止されません。IRQ がサンプルされた際に実行中の命令は、ISR が実行される前に完了します。

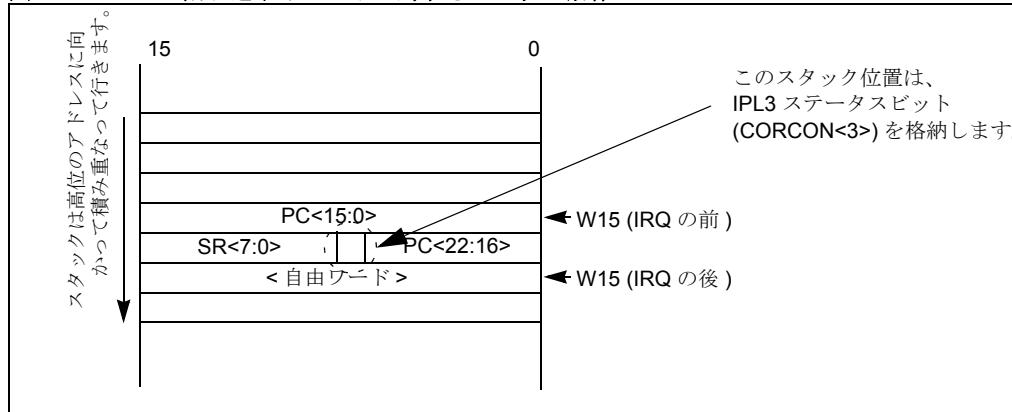
現プロセッサの優先度 (IPL<2:0> ステータスピット (SR<7:5>) で表示されます) より高い優先度がユーザーにより割り当てられた待ち合わせ中の IRQ が存在する場合は、割り込みはプロセッサに認識されます。すると、プロセッサは、以下の情報をソフトウェアスタックに保存します。

- 現 PC の値
- プロセッサステータスレジスタ (SRL) の下位バイト
- IPL3 ステータスピット (CORCON<3>)

スタックに保存されたこれらの 3 つの値は、PC 戻りアドレス値、MCU ステータスピット、自動保存される現プロセッサの優先度になります。

上記情報がスタックに保存された後に、CPU は、認識した割り込みの優先度を IPL<2:0> ビット位置に書き込みます。この動作により、RETFIE 命令を用いて割り込みサービスルーチン (ISR) が終了するまで、この優先度以下もしくは同じ値もすべての割り込みを禁止にします。

図 6-2: 割り込みイベントに対するスタック動作



### 6.2.4.1 割り込みからの復帰

RETFIE (割り込みからの復帰) 命令は、PC 戻りアドレス、IPL3 ステータスピットおよび SRL レジスタをスタックから戻し、プロセッサを、割り込みシーケンスが発生する以前の状態と優先度に戻します。

### 6.2.4.2 割り込みのネスティング

デフォルトでは、割り込みはネスティングできる状態です。実行中のどの ISR も、より高いユーザー割り当て優先度を持つ他の割り込みソースにより割り込みがかけられます。割り込みのネスティングは、NSTDIS 制御ビット (INTCON1<15>) をセットすることで、禁止できます。NSTDIS 制御ビットがセットされると、実行中のすべての割り込みは、IPL<2:0> = 111 に設定することにより CPU 優先度をレベル 7 にします。これにより、RETFIE 命令が実行されるまで、すべてのその他の割り込みソースをマスクできます。割り込みネスティングが禁止になると、同時待ち合わせ割り込み間の衝突を解決する場合以外は、ユーザー割り当ての優先度は無効になります。

IPL<2:0> ビットは、割り込みネスティングが禁止の時は読み込みのみ可能になります。これにより、ユーザーソフトウェアが IPL<2:0> を小さい値に設定すること (これは割り込みネスティングを再度有効にしますが) を防止します。

## 6.2.5 スリープとアイドルからの起動

IECx レジスタ内の対応する制御ビットを用いて、個別に許可された割り込みソースは、プロセッサをスリープもしくはアイドルモードから起動できます。割り込みステータスフラグがセットされ、IEC 制御レジスタ内の対応するビットで割り込みソースが有効になると、起動信号が dsPIC30F CPU に送られます。デバイスがスリープもしくはアイドルモードから起動すると、以下の 2 つうちの 1 つのアクションが発生します。

1. ソースの割り込み優先度が現 CPU の優先度よりも高い場合、プロセッサは割り込みを処理し、割り込みソースの ISR へと分岐します。
2. ソースのユーザー割り当て優先度が、現 CPU の優先度よりも低い場合、プロセッサは単に実行を継続し、CPU をスリープもしくはアイドルモードに以前セットした PWRSAV 命令の直後の命令から開始します。

**注：** CPU 優先度 0 に割り当てられたユーザー割り込みソースは、スリープもしくはアイドルモードから CPU を起動できます。ただし、これらのソースは割り当てられた優先度により、割り込み禁止のままであります。起動イベントでは、プロセッサは常に実行を継続し、CPU をスリープもしくはアイドルモードに以前セットした PWRSAV 命令の直後の命令から開始します。

## 6.2.6 A/D 変換器外部変換要求

INT0 外部割り込みピンは、外部変換要求信号として、A/D 変換器と共有されます。INT0 割り込みソースはプログラム可能なエッジ極性を持ちますが、A/D 変換器の外部変換要求のエッジ極性指定にも利用可能です。

## 6.2.7 外部割り込みサポート

dsPIC30F は最大 5 つの外部割り込みピンソース (INT0-INT4) をサポートします。それぞれの外部割り込みピンは割り込みイベントの検出用として、エッジ検出回路を持っています。INTCON2 レジスタは 5 つの制御ビット (INT0EP-INT4EP) を持ち、それらはエッジ検出回路の極性を選択します。それぞれの外部割り込みピンは、立ち上がりエッジもしくは立ち下がりエッジのイベントで、CPU に割り込みをかけるようにプログラムできます。詳しくは、**レジスタ 6-4** を参照してください。

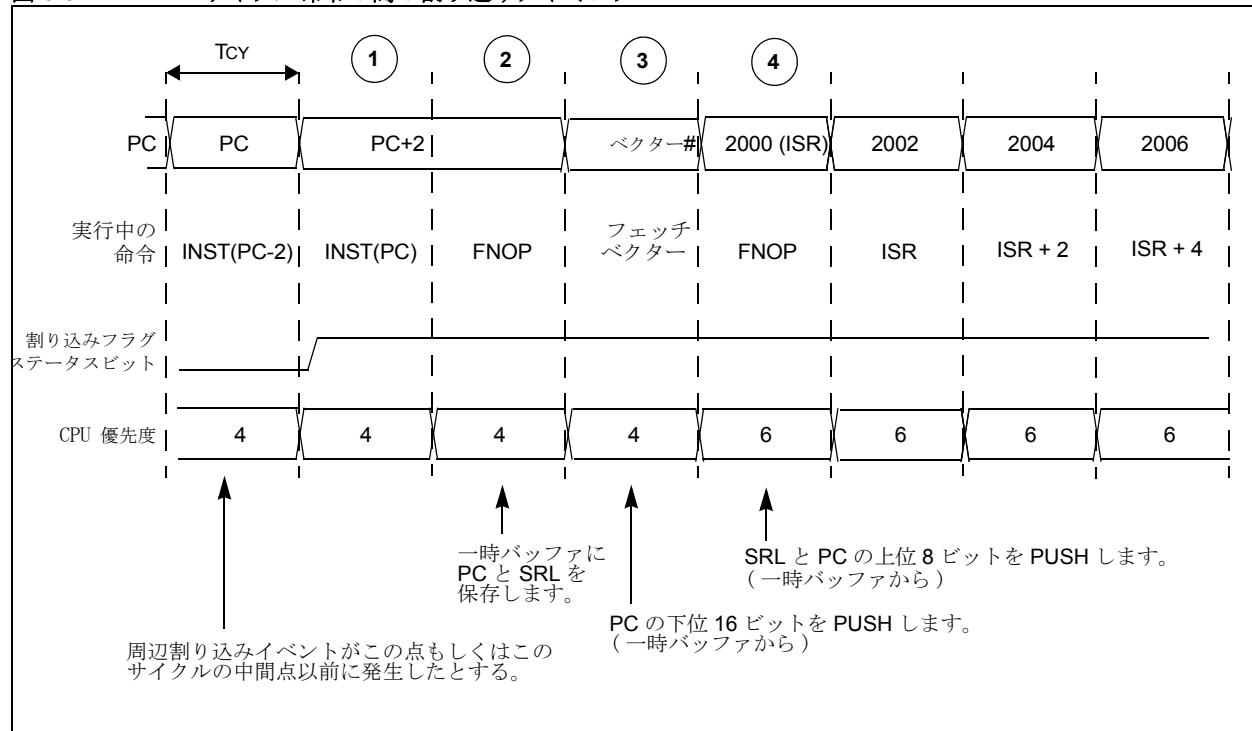
### 6.3 割り込み処理タイミング

#### 6.3.1 1サイクル命令の割り込み遅れ時間

割り込みタイミング図 6-3に、1サイクル命令間に周辺割り込みが割り込む時のイベントのシーケンスを示します。この割り込みには 6 つの命令サイクルが必要です。図内では参照のために各サイクルに番号が付けられています。

割り込みフラグステータスビットは、周辺割り込みが発生した後の命令サイクルの間にセットされます。この命令サイクルの間に現行命令が完了します。割り込みイベント後の 2 つ目の命令サイクルで、PC と SRL レジスタの内容が一時バッファレジスタに保存されます。2 サイクル命令の間に実行されたシーケンスとの一貫性を保つために、割り込み処理の 2 番目のサイクルが、NOP として実行されます（セクション 6.3.2 「2 サイクル命令の割り込み遅れ時間」参照）。3 番目のサイクルでは、PC には割り込みソース用のベクトルアドレスが転送され、ISR の開始アドレスがフェッチされます。4 番目のサイクルでは、PC に ISR アドレスが転送されます。4 番目のサイクルは、ISR 内の最初の命令がフェッチされる間、NOP として実行されます。

図 6-3: 1サイクル命令の間の割り込みタイミング



### 6.3.2 2サイクル命令の割り込み遅れ時間

2サイクル命令の時の割り込み遅れ時間は1サイクル命令の時と同じです。割り込み処理の最初と2番目のサイクルにより2サイクル命令の実行が完了します。図6-5に、2サイクル命令の実行以前の命令サイクル内で周辺割り込みイベントが発生する場合のケースを示しています。

図6-5では、周辺割り込みが、2サイクル命令の最初のサイクルと一致する時のタイミングを示しています。この場合、割り込み処理は、1サイクル命令として完了します（セクション6.3.1「1サイクル命令の割り込み遅れ時間」参照）。

図6-4: 2サイクル命令の間の割り込みタイミング

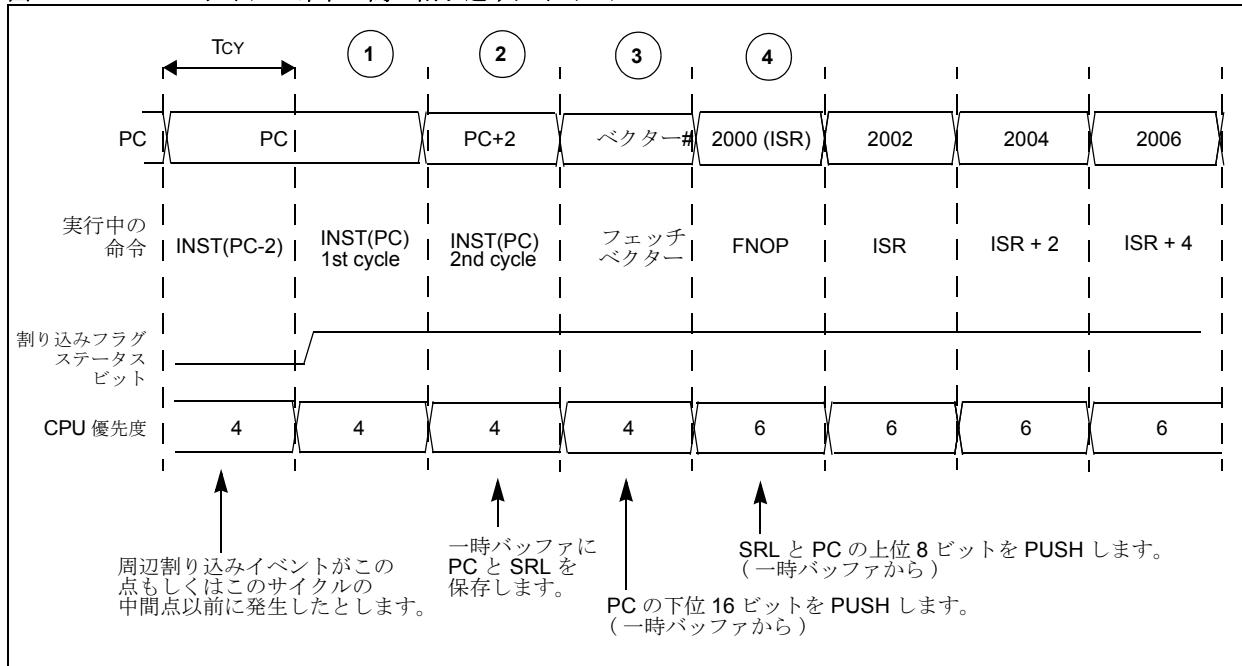
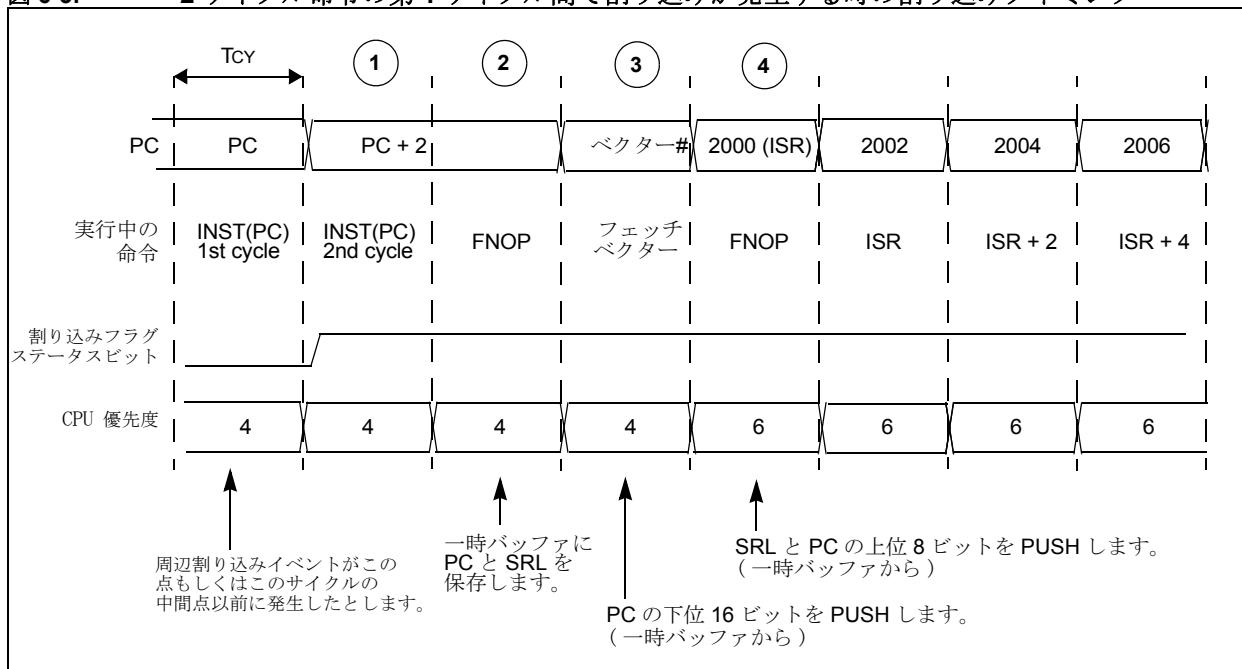


図6-5: 2サイクル命令の第1サイクル間で割り込みが発生する時の割り込みタイミング

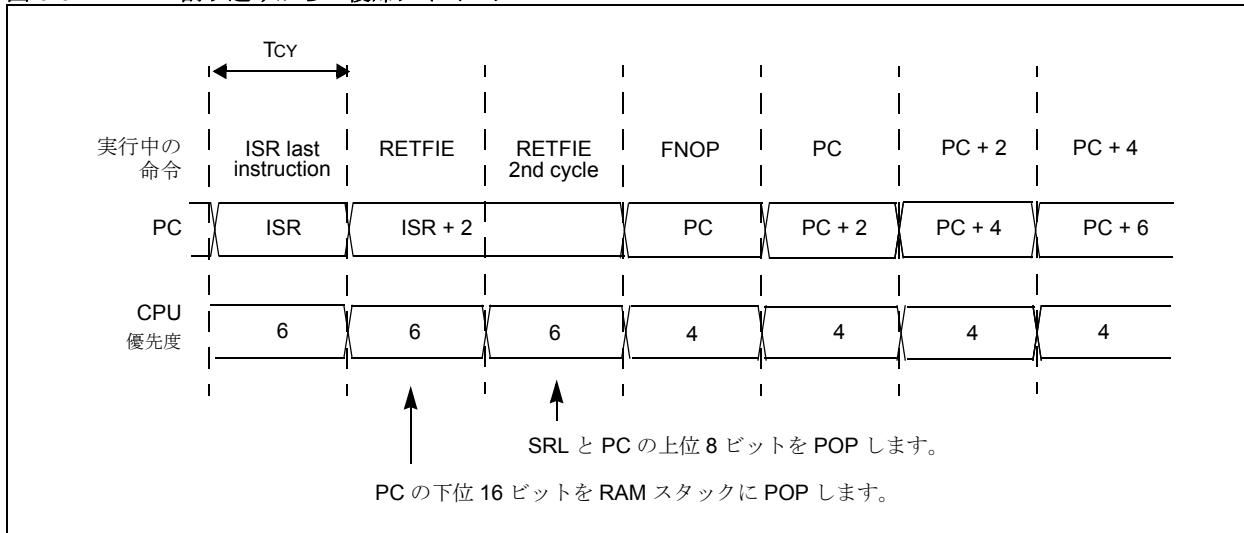


### 6.3.3 割り込みからの復帰

“割り込みからの復帰”命令、RETFIE、は割り込みもしくはトラップルーチンから抜け出ます。

RETFIE命令の最初のサイクルで、PCの上位ビットとSRLレジスタがスタックから引き出されます。スタックされたPCの下位16ビットの値は、2番目のサイクルで引き出されます。3番目のサイクルは、更新されたプログラムカウンタによりアドレスシングされた命令のフェッチに使用されます。このサイクルは、NOPとして実行されます。

図 6-6: 割り込みからの復帰タイミング



### 6.3.4 割り込み遅れ時間の特殊な状態

dsPIC30Fは、周辺割り込みソースが発生したときには、現命令を完了させます。割り込みの遅れ時間は、1サイクル命令および2サイクル命令で同じです。ただし、割り込みが発生するタイミングにより、割り込みの遅れ時間が1サイクルだけ増加する条件があります。固定遅れ時間がアプリケーションにとって重要である場合は、ユーザーはこのような条件が発生するのを避ける必要があります。具体的には、以下のようないくつかの条件です。

- プログラムメモリ空間の値をアクセスするためにPSVを使用してMOV.D命令が実行される。
- 2サイクル命令に対して、命令ストール（停止）サイクルが付加される。
- PSVアクセスを実行する1サイクル命令に対して、命令ストール（停止）サイクルが付加される。
- ビットテストとスキップ命令(BTSC, BTSS)がプログラムメモリ空間内の値にアクセスするするためにPSVを使用する。

## 6.4 割り込み用制御、ステータスレジスタ

以下のレジスタは、割り込みコントローラと連携しています。

- **INTCON1、INTCON2 レジスタ**

グローバル割り込み制御機能は、この 2 つのレジスタで制御されます。INTCON1 は、プロセッサトラップソース用の制御ステータスフラグと、割り込みネスティング禁止(NSTDIS) ビットを含みます。INTCON2 レジスタは外部割り込み要求信号の動作と代替のベクトル表の使用についての制御を行います。

- **IFSx: 割り込みフラグステータスレジスタ**

すべての割り込み要求フラグは IFSx レジスタ内に保持され、「x」はレジスタ番号を示します。それぞれの割り込みソースは、ステータスピットを持ち、それぞれの周辺もしくは外部信号によりセットされ、ソフトウェアでクリアされます。

- **IECx: 割り込み許可制御レジスタ**

すべての割り込み許可制御フラグは IECx レジスタ内に保持され、「x」はレジスタ番号を示します。これらの制御ビットは、周辺もしくは外部信号からの割り込みを、個別に許可にするために用いられます。

- **IPCx: 割り込み優先度制御レジスタ**

それぞれのユーザー割り込みソースは 8 つの優先度の 1 つに割り当てることができます。IPC レジスタは、1 つ 1 つの割り込みソースに割り込み優先度を設定するために用いられます。

- **SR: CPU ステータスレジスタ**

SR は割り込みコントローラハードウェア用として使われるだけではなく、現 CPU の優先度を示す **IPL<2:0>** ステータスピット (**SR<7:5>**) を含んでいます。ユーザーは、IPL ピットに書き込むことにより、現 CPU の優先度を変更することができます。

- **CORCON: コア制御レジスタ**

CORCON は割り込みコントローラハードウェア用として使われるだけではなく、現 CPU の優先度を示す **IPL3** ステータスピットを含んでいます。IPL3 は、トラップイベントがユーザーソフトウェアでマスクされないようにするため、読み込み専用のビットです。

それぞれのレジスタは、以下のページで詳しく説明されます。

**注：** 割り込みソースの総数とタイプはデバイスの種類に依存します。詳しくはデバイスのデータシートを参照してください。

### 6.4.1 制御レジスタへの割り込みの割り当て

割り込みソースは、表 6-2 にリストアップされる割り込みソースと同じ順序で IFSx, IECx および IPCx に割り当てられます。例えば、INT0 (外部割り込み 0) は、ベクトル番号 ‘0’ の自然順優先度を持つものとして示されます。従って、INT0IF ステータスピットは IFS0<0> に存在します。INT0 割り込みは、許可ビットとして IEC0 レジスタのビット 0 を使用し、IPC0<2:0> ビットは割り込み優先度を INT0 割り込みとして割り当てます。

## レジスタ 6-1: SR: ステータスレジスタ (CPU 内)

上位バイト:							
R-0	R-0	R/C-0	R/C-0	R-0	R/C-0	R-0	R-0
OA	OB	SA	SB	OAB	SAB	DA	DC
ビット 15				ビット 8			

下位バイト:							
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
IPL<2:0>		RA	N	OV	Z		C
ビット 7				ビット 0			

ビット 7 IPL&lt;2:0&gt;: CPU 割り込み優先度ステータスピット

-5 111 = CPU 割り込み優先度が 7(15) です。ユーザー割り込みは無効です。

110 = CPU 割り込み優先度が 6(14) です。

101 = CPU 割り込み優先度が 5(13) です。

100 = CPU 割り込み優先度が 4(12) です。

011 = CPU 割り込み優先度が 3(11) です。

010 = CPU 割り込み優先度が 2(10) です。

001 = CPU 割り込み優先度が 1(9) です。

000 = CPU 割り込み優先度が 0(8) です。

注 1: CPU 割り込み優先度を形成するために、IPL&lt;2:0&gt; ビットは IPL&lt;3&gt; ビット (CORCON&lt;3&gt;) と連結されます。括弧内の値は IPL&lt;3&gt; = 1 の IPL を表示します。

2: IPL&lt;2:0&gt; ステータスピットは、NSTDIS = 1 (INTCON1&lt;15&gt;) の時は読み込み専用です。

凡例:

R = 読み込み可能 ビット	W = 書き込み可能 ビット	U = 未実装、'0' が読み込まれます	C = クリア可能なビット
-n = POR での値	'1' = ビットが セットされます	'0' = ビットはクリアされます	x = ビットは不定です

## レジスタ 6-2: CORCON: コア制御レジスタ

上位バイト:							
U-0	U-0	U-0	R/W-0	R/W-0	R-0	R-0	R-0
—	—	—	US	EDT	DL<1:0>		
ビット 15				ビット 8			

下位バイト:							
R/W-0	R/W-0	R/W-1	R/W-0	R/C-0	R/W-0	R/W-0	R/W-0
SATA	SATB	SATDW	ACCSAT	IPL3	PSV	RND	IF
ビット 7				ビット 0			

ビット 3 IPL3: CPU 割り込み優先度ステータスピット 3

1 = CPU 割り込み優先度が 7 より高い

0 = CPU 割り込み優先度が 7 かそれ以下

注 1: CPU 割り込み優先度を形成するために、IPL3 ビットは IPL&lt;2:0&gt; ビット (SR&lt;7:5&gt;) と連結されます。

凡例:

R = 読み込み可能 ビット	W = 書き込み可能 ビット	U = 未実装、'0' が読み込まれます	C = クリア可能なビット
-n = POR での値	'1' = ビットが セットされます	'0' = ビットはクリアされます	x = ビットは不定です

## レジスタ 6-3: INTCON1: 割り込み制御レジスタ 1

上位バイト :							
R/W-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
NSTDIS	—	—	—	—	OVATE	OVBTE	COVTE
ビット 15							ビット 8

下位バイト :							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
—	—	—	MATHERR	ADDRERR	STKERR	OSCF FAIL	—
ビット 7							ビット 0

ビット 15 **NSTDIS:** 割り込みネスティング制御ビット

1 = 割り込みネスティング禁止。  
0 = 割り込みネスティング許可。

ビット 14-11 未実装: ‘0’ が読み込まれます。

ビット 10 **OVATE:** アキュームレータ A オーバーフロー トラップ有効ビット  
1 = キュームレータ A のオーバーフローをトラップします。  
0 = トラップ禁止。

ビット 9 **OVBTE:** アキュームレータ B オーバーフロー トラップ有効ビット  
1 = アキュームレータ B のオーバーフローをトラップします。  
0 = トラップ禁止。

ビット 8 **COVTE:** 破滅的オーバーフロー トラップ有効ビット  
1 = アキュームレータ A もしくは B の破滅的オーバーフローをトラップします。  
0 = トラップ禁止。

ビット 7 未実装: ‘0’ が読み込まれます。

-5

ビット 4 **MATHERR:** 算術エラーステータスビット  
1 = オーバーフロー トラップが発生します。  
0 = オーバーフロー トラップが発生しません。

ビット 3 **ADDRERR:** アドレスエラートラップステータスビット  
1 = アドレスエラー トラップが発生します。  
0 = アドレスエラー トラップが発生しません。

ビット 2 **STKERR:** スタックエラートラップステータスビット  
1 = スタックエラー トラップが発生します。  
0 = スタックエラー トラップが発生しません。

ビット 1 **OSCF FAIL:** 発振器不良トラップステータスビット  
1 = 発振器不良トラップスが発生します。  
0 = 発振器不良トラップスが発生しません。

ビット 0 未実装: ‘0’ が読み込まれます。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’ が読み込まれます

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です

## レジスタ 6-4: INTCON2: 割り込み制御レジスタ 2

上位バイト:								
R/W-0	R-0	U-0						
ALTIVT	DISI	—	—	—	—	—	—	—
ビット 15								ビット 8

下位バイト:								
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP	—
ビット 7								ビット 0

ビット 15 **ALTIVT:** 代替割り込みベクトル表有効ビット

1 = 代替ベクトル表を使用します。

0 = 標準（デフォルト）ベクトル表を使用します。

ビット 14 **DISI:** DISI 命令ステータスビット

1 = DISI 命令が動作中です。

0 = DISI は動作中ではありません。

ビット 13-5 未実装: '0' が読み込まれます。

13-5

ビット 4 **INT4EP:** 外部割り込み #4 エッジ検出極性選択ビット

1 = 立ち下がりエッジで割り込み発生。

0 = 立ち上がりエッジで割り込み発生。

ビット 3 **INT3EP:** 外部割り込み #3 エッジ検出極性選択ビット

1 = 立ち下がりエッジで割り込み発生。

0 = 立ち上がりエッジで割り込み発生。

ビット 2 **INT2EP:** 外部割り込み #2 エッジ検出極性選択ビット

1 = 立ち下がりエッジで割り込み発生。

0 = 立ち上がりエッジで割り込み発生。

ビット 1 **INT1EP:** 外部割り込み #1 エッジ検出極性選択ビット

1 = 立ち下がりエッジで割り込み発生。

0 = 立ち上がりエッジで割り込み発生。

ビット 0 **INT0EP:** 外部割り込み #0 エッジ検出極性選択ビット

1 = 立ち下がりエッジで割り込み発生。

0 = 立ち上がりエッジで割り込み発生。

凡例:

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、'0' が読み込まれます

ト

-n = POR での値

'1' = ビットがセット '0' = ビットはクリア x = ビットは不定です

されます

## レジスタ 6-5: IFS0: 割り込みフラグステータスレジスタ 0

上位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNIF	BCLIF	I2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF
ビット 15							ビット 8

下位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0IF
ビット 7							ビット 0

- ビット 15 **CNIF:** 入力変化通知フラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 14 **BCLIF:** I<sup>2</sup>C バス衝突フラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 13 **I2CIF:** I<sup>2</sup>C バス転送完了割り込みフラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 12 **NVMIF:** 不揮発性メモリ書き込み完了割り込みフラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 11 **ADIF:** A/D 変換完了割り込みフラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 10 **U1TXIF:** UART1 送信割り込みフラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 9 **U1RXIF:** UART1 受信割り込みフラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 8 **SPI1IF:** SPI1 割り込みフラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 7 **T3IF:** タイマー 3 割り込みフラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 6 **T2IF:** タイマー 2 割り込みフラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 5 **OC2IF:** 出力比較チャネル 2 割り込みフラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 4 **IC2IF:** 入力キャプチャ 2 割り込みフラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 3 **T1IF:** タイマー 1 割り込みフラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 2 **OC1IF:** 出力比較チャネル 1 割り込みフラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

## レジスタ 6-5: IFS0: 割り込みフラグステータスレジスタ 0 (続き)

ビット 1 **IC1IF:** 入力キャプチャ 1 割り込みフラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

ビット 0 **INT0IF:** 外部割り込み 0 フラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、'0' が読み込まれます

-n = POR での値

'1' = ビットがセット '0' = ビットはクリア x = ビットは不定であります

## レジスタ 6-6: IFS1: 割り込みフラグステータスレジスタ 1

上位バイト :

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IC6IF	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF
ビット 15				ビット 8			

下位バイト :

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF
ビット 7				ビット 0			

- ビット 15 **IC6IF:** 入力キャプチャチャネル 6 入力フラグステータスビット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 14 **IC5IF:** 入力キャプチャチャネル 5 入力フラグステータスビット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 13 **IC4IF:** 入力キャプチャチャネル 4 入力フラグステータスビット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 12 **IC3IF:** 入力キャプチャチャネル 3 入力フラグステータスビット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 11 **C1IF:** CAN1(結合)割り込みフラグステータスビット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 10 **SPI2IF:** SPI2 割り込みフラグステータスビット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 9 **U2TXIF:** UART2 送信割り込みフラグステータスビット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 8 **U2RXIF:** UART2 受信割り込みフラグステータスビット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 7 **INT2IF:** 外部割り込み 2 フラグステータスビット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 6 **T5IF:** タイマー 5 割り込みフラグステータスビット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 5 **T4IF:** タイマー 4 割り込みフラグステータスビット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 4 **OC4IF:** 出力比較チャネル 4 割り込みフラグステータスビット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 3 **OC3IF:** 出力比較チャネル 3 割り込みフラグステータスビット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。
- ビット 2 **IC8IF:** 入力キャプチャチャネル 8 割り込みフラグステータスビット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

## レジスタ 6-6: IFS1: 割り込みフラグステータスレジスタ 1 (続き)

ビット 1 **IC7IF:** 入力キャプチャチャネル 7 割り込みフラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

ビット 0 **INT1IF:** 外部割り込み 1 フラグステータスピット  
1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、'0' が読み込まれます

-n = POR での値

'1' = ビットがセット '0' = ビットはクリア x = ビットは不定であります

## レジスタ 6-7: IFS2: 割り込みフラグステータスレジスタ 2

上位バイト：							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FLTBIF	FLTAIF	LVDIF	DCIIF	QEIIIF
ビット 15							ビット 8

下位バイト：							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF
ビット 7							ビット 0

ビット 15-13 未実装：‘0’が読み込まれます。

ビット 12 **FLTBIF: FAULT B** 入力割り込みフラグステータスビット

1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

ビット 11 **FLTAIF: FAULT A** 入力割り込みフラグステータスビット

1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

ビット 10 **LVDIF:** プログラマブル低電圧検出割り込みフラグステータスビット

1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

ビット 9 **DCIIF:** データ変換インターフェース割り込みフラグステータスビット

1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

ビット 8 **QEIIIF:** 直交符号インターフェース割り込みフラグステータスビット

1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

ビット 7 **PWMIF:** モーター制御パルス幅変調割り込みフラグステータスビット

1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

ビット 6 **C2IF:** CAN2(結合)割り込みフラグステータスビット

1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

ビット 5 **INT4IF:** 外部割り込み 4 フラグステータスビット

1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

ビット 4 **INT3IF:** 外部割り込み 3 フラグステータスビット

1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

ビット 3 **OC8IF:** 出力比較チャネル 8 割り込みフラグステータスビット

1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

ビット 2 **OC7IF:** 出力比較チャネル 7 割り込みフラグステータスビット

1 = 割り込み要求が発生しています。  
0 = 割り込み要求は発生していません。

## レジスタ 6-7: IFS2: 割り込みフラグステータスレジスタ 2 (続き)

ビット 1 **OC6IF:** 出力比較チャネル 6 割り込みフラグステータスピット

1 = 割り込み要求が発生しています。

0 = 割り込み要求は発生していません。

ビット 0 **OC5IF:** 出力比較チャネル 5 割り込みフラグステータスピット

1 = 割り込み要求が発生しています。

0 = 割り込み要求は発生していません。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、'0' が読み込まれます

-n = POR での値

'1' = ビットがセット '0' = ビットはクリア X = ビットは不定です  
されます

## レジスタ 6-8: IEC0: 割り込み許可制御レジスタ 0

上位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNIE	BCLIE	I2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE
ビット 15							ビット 8

下位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE
ビット 7							ビット 0

ビット 15 **CNIE:** 入力変化通知割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 14 **BCLIE:** I<sup>2</sup>C バス衝突割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 13 **I2CIE:** I<sup>2</sup>C 転送完了割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 12 **NVMIE:** 不揮発性メモリ書き込み完了割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 11 **ADIE:** A/D 変換完了割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 10 **U1TXIE:** UART1 送信割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 9 **U1RXIE:** UART1 受信割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 8 **SPI1IE:** SPI1 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 7 **T3IE:** タイマー 3 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 6 **T2IE:** タイマー 2 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 5 **OC2IE:** 出力比較チャネル 2 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 4 **IC2IE:** 入力キャプチャチャネル 2 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 3 **T1IE:** タイマー 1 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 2 **OC1IE:** 出力比較チャネル 1 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

## レジスタ 6-8: IEC0: 割り込み許可制御レジスタ 0 (続き)

ビット 1 **IC1IE:** 入力キャプチャチャネル 1 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 0 **INT0IE:** 外部割り込み 0 許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、'0' が読み込まれます

-n = POR での値

'1' = ビットがセット '0' = ビットはクリア x = ビットは不定です  
されます

## レジスタ 6-9: IEC1: 割り込み許可制御レジスタ 1

上位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IC6IE	IC5IE	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE
ビット 15							ビット 8

下位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE
ビット 7							ビット 0

ビット 15 **IC6IE:** 入力キャプチャチャネル 6 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 14 **IC5IE:** 入力キャプチャチャネル 5 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 13 **IC4IE:** 入力キャプチャチャネル 4 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 12 **IC3IE:** 入力キャプチャチャネル 3 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 11 **C1IE:** CAN1(結合)割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 10 **SPI2IE:** SPI2 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 9 **U2TXIE:** UART2 送信割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 8 **U2RXIE:** UART2 受信割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 7 **INT2IE:** 外部割り込み 2 許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 6 **T5IE:** タイマー 5 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 5 **T4IE:** タイマー 4 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 4 **OC4IE:** 出力比較チャネル 4 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 3 **OC3IE:** 出力比較チャネル 3 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 2 **IC8IE:** 入力キャプチャチャネル 8 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

## レジスタ 6-9: IEC1: 割り込み許可制御レジスタ 1 (続き)

ビット 1 **IC7IE:** 入力キャプチャチャネル 7 割り込み許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

ビット 0 **INT1IE:** 外部割り込み 1 許可ビット

1 = 割り込み許可。

0 = 割り込み禁止。

凡例 :

R = 読み込み可能ビット

-n = POR での値

W = 書き込み可能ビット  
U = 未実装、'0' が読み込まれます

'1' = ビットがセット '0' = ビットはクリア X = ビットは不定です  
されます

## レジスタ 6-10: IEC2: 割り込み許可制御レジスタ 2

上位バイト :							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FLTBIE	FLTAIE	LVDIE	DCIIE	QEIIIE
ビット 15							ビット 8

下位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWMIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE
ビット 7							ビット 0

ビット 15-13 未実装: ‘0’ が読み込まれます。

ビット 12 **FLTBIE:** FAULT B 入力割り込み許可ビット

1 = 割り込み許可。  
0 = 割り込み禁止。

ビット 11 **FLTAIE:** FAULTA 入力割り込み許可ビット

1 = 割り込み許可。  
0 = 割り込み禁止。

ビット 10 **LVDIE:** プログラマブル低電圧割り込み許可ビット

1 = 割り込み許可。  
0 = 割り込み禁止。

ビット 9 **DCIIE:** データ変換インターフェース割り込み許可ビット

1 = 割り込み許可。  
0 = 割り込み禁止。

ビット 8 **QEIIIE:** 直交符号インターフェース割り込み許可ビット

1 = 割り込み許可。  
0 = 割り込み禁止。

ビット 7 **PWMIE:** モーター制御パルス幅変調割り込み許可ビット

1 = 割り込み許可。  
0 = 割り込み禁止。

ビット 6 **C2IE:** CAN2( 結合 ) 割り込み許可ビット

1 = 割り込み許可。  
0 = 割り込み禁止。

ビット 5 **INT4IE:** 外部割り込み 4 許可ビット

1 = 割り込み許可。  
0 = 割り込み禁止。

ビット 4 **INT3IE:** 外部割り込み 3 許可ビット

1 = 割り込み許可。  
0 = 割り込み禁止。

ビット 3 **OC8IE:** 出力比較チャネル 8 割り込み許可ビット

1 = 割り込み許可。  
0 = 割り込み禁止。

ビット 2 **OC7IE:** 出力比較チャネル 7 割り込み許可ビット

1 = 割り込み許可。  
0 = 割り込み禁止。

## レジスタ 6-10: IEC2: 割り込み許可制御レジスタ 2 (続き)

ビット 1 **OC6IE:** 出力比較チャネル 6 割り込み許可ビット

1 = 割り込み許可。  
0 = 割り込み禁止。

ビット 0 **OC5IE:** 出力比較チャネル 5 割り込み許可ビット

1 = 割り込み許可。  
0 = 割り込み禁止。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、'0' が読み込まれます

-n = POR での値

'1' = ビットがセット '0' = ビットはクリア X = ビットは不定です  
されます

## レジスタ 6-11: IPC0: 割り込み優先度制御レジスタ 0

上位バイト :

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	T1IP<2:0>	—	—	—	OC1IP<2:0>	—	—

ビット 15

ビット 8

下位バイト :

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	IC1IP<2:0>	—	—	—	INT0IP<2:0>	—	—

ビット 7

ビット 0

ビット 15 未実装 : ‘0’ が読み込まれます。

ビット 14-12 T1IP<2:0>: タイマー 1 割り込み優先度ビット

111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 11 未実装 : ‘0’ が読み込まれます。

ビット 10-8 OC1IP<2:0>: 出力比較チャネル 1 割り込み優先度ビット

111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 7 未実装 : ‘0’ が読み込まれます。

ビット 6 IC1IP<2:0>: 入力キャプチャチャネル 1 割り込み優先度ビット

-4 111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 3 未実装 : ‘0’ が読み込まれます。

ビット 2 INT0IP<2:0>: 外部割り込み 0 優先度ビット

-0 111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’ が読み込まれます

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です

## レジスタ 6-12: IPC1: 割り込み優先度制御レジスタ 1

上位バイト:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—		T3IP<2:0>		—		T2IP<2:0>	
ビット 15						ビット 8	

下位バイト:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—		OC2IP<2:0>		—		IC2IP<2:0>	
ビット 7						ビット 0	

ビット 15 未実装: ‘0’ が読み込まれます。

ビット 14-12 **T3IP<2:0>**: タイマー 3 割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 11 未実装: ‘0’ が読み込まれます。

ビット 10-8 **T2IP<2:0>**: タイマー 2 割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 7 未実装: ‘0’ が読み込まれます。

ビット 6 **OC2IP<2:0>**: 出力比較チャネル 2 割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 3 未実装: ‘0’ が読み込まれます。

ビット 2 **IC2IP<2:0>**: 入力キャプチャチャネル 2 割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

凡例:

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’ が読み込まれます

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です

されます

## レジスタ 6-13: IPC2: 割り込み優先度制御レジスタ 2

上位バイト :

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	ADIP<2:0>		—	U1TXIP<2:0>			
ビット 15							ビット 8

下位バイト :

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	U1RXIP<2:0>		—	SPI1IP<2:0>			
ビット 7							ビット 0

ビット 15 未実装 : ‘0’ が読み込まれます。

ビット 14-12 ADIP<2:0>: A/D 変換完了割り込み優先度ビット

111 = 割り込みが優先度 7 です。 (最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 11 未実装 : ‘0’ が読み込まれます。

ビット 10-8 U1TXIP<0>: UART1 送信割り込み優先度ビット

111 = 割り込みが優先度 7 です。 (最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 7 未実装 : ‘0’ が読み込まれます。

ビット 6 U1RXIP<2:0>: UART1 受信割り込み優先度ビット

-4 111 = 割り込みが優先度 7 です。 (最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 3 未実装 : ‘0’ が読み込まれます。

ビット 2 SPI1IP<2:0>: SPI1 割り込み優先度ビット

-0 111 = 割り込みが優先度 7 です。 (最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’ が読み込まれます

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です  
されます されます

## レジスタ 6-14: IPC3: 割り込み優先度制御レジスタ 3

上位バイト:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—		CNIP<2:0>		—		BCLIP<2:0>	
ビット 15						ビット 8	

下位バイト:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—		I2CIP<2:0>		—		NVMIP<2:0>	
ビット 7						ビット 0	

ビット 15 未実装: ‘0’ が読み込まれます。

ビット 14-12 CNIP<2:0>: 入力変化通知割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

- 
- 
- 

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 11 未実装: ‘0’ が読み込まれます。

ビット 10-8 BCLIP<2:0>: I<sup>2</sup>C バス衝突割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

- 
- 
- 

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 7 未実装: ‘0’ が読み込まれます。

ビット 6 I2CIP<2:0>: I<sup>2</sup>C 送信完了割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

- 
- 
- 

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 3 未実装: ‘0’ が読み込まれます。

ビット 2 NVMIP<2:0>: 不揮発性メモリ書き込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

- 
- 
- 

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

凡例:

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’ が読み込まれます

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です

されます

## レジスタ 6-15: IPC4: 割り込み優先度制御レジスタ 4

上位バイト :

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	OC3IP<2:0>	—	—	—	IC8IP<2:0>	—	—

ビット 15

ビット 8

下位バイト :

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	IC7IP<2:0>	—	—	—	INT1IP<2:0>	—	—

ビット 7

ビット 0

ビット 15 未実装 : ‘0’ が読み込まれます。

ビット 14-12 OC3IP<2:0>: 出力比較チャネル 3 割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 11 未実装 : ‘0’ が読み込まれます。

ビット 10-8 IC8IP<2:0>: 入力キャプチャチャネル 8 割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 7 未実装 : ‘0’ が読み込まれます。

ビット 6 IC7IP<2:0>: 入力キャプチャチャネル 7 割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 3 未実装 : ‘0’ が読み込まれます。

ビット 2 INT1IP<2:0>: 外部割り込み 1 優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’ が読み込まれます

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です

## レジスタ 6-16: IPC5: 割り込み優先度制御レジスタ 5

上位バイト:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—		INT2IP<2:0>		—		T5IP<2:0>	
ビット 15						ビット 8	

下位バイト:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—		T4IP<2:0>		—		OC4IP<2:0>	
ビット 7						ビット 0	

ビット 15 未実装: ‘0’ が読み込まれます。

ビット 14-12 INT2IP<2:0>: 外部割り込み 2 優先度ビット  
111 = 割り込みが優先度 7 です。 (最高の優先度割り込み)

- 
- 
- 

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 11 未実装: ‘0’ が読み込まれます。

ビット 10-8 T5IP<2:0>: タイマー 5 割り込み優先度ビット  
111 = 割り込みが優先度 7 です。 (最高の優先度割り込み)

- 
- 
- 

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 7 未実装: ‘0’ が読み込まれます。

ビット 6 T4IP<2:0>: タイマー 4 割り込み優先度ビット  
111 = 割り込みが優先度 7 です。 (最高の優先度割り込み)

- 
- 
- 

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 3 未実装: ‘0’ が読み込まれます。

ビット 2 OC4IP<2:0>: 出力比較チャネル 4 割り込み優先度ビット  
111 = 割り込みが優先度 7 です。 (最高の優先度割り込み)

- 
- 
- 

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

凡例:

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’ が読み込まれます

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です  
されます

## レジスタ 6-17: IPC6: 割り込み優先度制御レジスタ 6

上位バイト :

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	C1IP<2:0>	—	—	SPI2IP<2:0>	—	—	—

ビット 15

ビット 8

下位バイト :

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	U2TXIP<2:0>	—	—	U2RXIP<2:0>	—	—	—

ビット 7

ビット 0

ビット 15 未実装 : ‘0’ が読み込まれます。

ビット 14-12 C1IP<2:0>: CAN1( 結合 ) 割り込み優先度ビット

111 = 割り込みが優先度 7 です。 ( 最高の優先度割り込み )

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 11 未実装 : ‘0’ が読み込まれます。

ビット 10-8 SPI2IP<2:0>: SPI2 割り込み優先度ビット

111 = 割り込みが優先度 7 です。 ( 最高の優先度割り込み )

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 7 未実装 : ‘0’ が読み込まれます。

ビット 6 U2TXIP<2:0>: UART2 送信割り込み優先度ビット

-4 111 = 割り込みが優先度 7 です。 ( 最高の優先度割り込み )

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 3 未実装 : ‘0’ が読み込まれます。

ビット 2 U2RXIP<2:0>: UART2 受信割り込み優先度ビット

-0 111 = 割り込みが優先度 7 です。 ( 最高の優先度割り込み )

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’ が読み込まれます

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です

## レジスタ 6-18: IPC7: 割り込み優先度制御レジスタ 7

上位バイト:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—		IC6IP<2:0>		—		IC5IP<2:0>	
ビット 15						ビット 8	

下位バイト:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—		IC4IP<2:0>		—		IC3IP<2:0>	
ビット 7						ビット 0	

ビット 15 未実装: ‘0’が読み込まれます。

ビット 14-12 **IC6IP<2:0>**: 入力キャプチャチャネル 6 割り込みビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 11 未実装: ‘0’が読み込まれます。

ビット 10-8 **IC5IP<2:0>**: 入力キャプチャチャネル 5 割り込みビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 7 未実装: ‘0’が読み込まれます。

ビット 6 **IC4IP<2:0>**: 入力キャプチャチャネル 4 割り込みビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 3 未実装: ‘0’が読み込まれます。

ビット 2 **IC3IP<2:0>**: 入力キャプチャチャネル 3 割り込みビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

凡例:

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’が読み込まれます

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です

されます

## レジスタ 6-19: IPC8: 割り込み優先度制御レジスタ 8

上位バイト :

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	OC8IP<2:0>	—	—	—	OC7IP<2:0>	—	—

ビット 15

ビット 8

下位バイト :

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	OC6IP<2:0>	—	—	—	OC5IP<2:0>	—	—

ビット 7

ビット 0

ビット 15 未実装 : ‘0’ が読み込まれます。

ビット 14-12 OC8IP<2:0>: 出力比較チャネル 8 割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 11 未実装 : ‘0’ が読み込まれます。

ビット 10-8 OC7IP<2:0>: 出力比較チャネル 7 割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 7 未実装 : ‘0’ が読み込まれます。

ビット 6 OC6IP<2:0>: 出力比較チャネル 6 割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 3 未実装 : ‘0’ が読み込まれます。

ビット 2 OC5IP<2:0>: 出力比較チャネル 5 割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’ が読み込まれます

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です

## レジスタ 6-20: IPC9: 割り込み優先度制御レジスタ 9

上位バイト:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—		PWMIP<2:0>		—		C2IP<2:0>	
ビット 15						ビット 8	

下位バイト:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—		INT4IP<2:0>		—		INT3IP<2:0>	
ビット 7						ビット 0	

ビット 15 未実装: ‘0’が読み込まれます。

ビット 14-12 **PWMIP<2:0>**: モーター制御パルス幅変調割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

- 
- 
- 

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 11 未実装: ‘0’が読み込まれます。

ビット 10-8 **C2IP<2:0>**: CAN2(結合)割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

- 
- 
- 

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 7 未実装: ‘0’が読み込まれます。

ビット 6 **INT4IP<2:0>**: 外部割り込み 4 優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

- 
- 
- 

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 3 未実装: ‘0’が読み込まれます。

ビット 2 **INT3IP<2:0>**: 外部割り込み 3 優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

- 
- 
- 

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

凡例:

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’が読み込まれます

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です

されます

## レジスタ 6-21: IPC10: 割り込み優先度制御レジスタ 10

上位バイト :

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	FLTAIP<2:0>	—	—	—	LVDIP<2:0>	—	—

ビット 15

ビット 8

下位バイト :

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	DCIIP<2:0>	—	—	—	QEIIIP<2:0>	—	—

ビット 7

ビット 0

ビット 15 未実装 : ‘0’ が読み込まれます。

ビット 14-12 **FLTAIP<2:0>:** FAULT A 入力割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 11 未実装 : ‘0’ が読み込まれます。

ビット 10-8 **LVDIP<2:0>:** プログラマブル低電圧検出割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 7 未実装 : ‘0’ が読み込まれます。

ビット 6 **DCIIP<2:0>:** データ変換インターフェース割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

ビット 3 未実装 : ‘0’ が読み込まれます。

ビット 2 **QEIIIP<2:0>:** 直交符号インターフェース割り込み優先度ビット  
111 = 割り込みが優先度 7 です。(最高の優先度割り込み)

•  
•  
•

001 = 割り込みが優先度 1 です。

000 = 割り込みソースが無効です。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’ が読み込まれます

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です

## レジスタ 6-22: IPC11: 割り込み優先度制御レジスタ 11

上位バイト:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
ビット 15							ビット 8

下位バイト:							
U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R/W-0
—	—	—	—	—		FLTBIP<2:0>	
ビット 7							ビット 0

ビット 15-3 未実装: ‘0’が読み込まれます。

- ビット 2 **FLTBIP<2:0>**: FAULT B 入力割り込み優先度ビット  
 -0 111 = 割り込みが優先度 7 です。(最高の優先度割り込み)  
 •  
 •  
 •  
 001 = 割り込みが優先度 1 です。  
 000 = 割り込みソースが無効です。

凡例:

R = 読み込み可能ビット	W = 書き込み可能ビット	U = 未実装、‘0’が読み込まれます
-n = POR での値	‘1’ = ビットがセット	‘0’ = ビットはクリア
	されます	x = ビットは不定です

## 6.5 割り込み設定手順

### 6.5.1 初期化

以下の手順で割り込みのソースを構成します。

1. ネスティングされた割り込みが必要でない場合、**NSTDIS** 制御ビット (**INTCON1<15>**) をセットします。
2. 適切な **IPCx** 制御レジスタ内の制御ビットに書き込むことで、割り込みソース用の、ユーザー割り当て優先度を選択します。優先度は、割り込みソースの特定のアプリケーションとタイプに依存します。複数割り込みレベルが必要でない場合は、すべての有効になつた割り込みソース用の **IPCx** レジスタの制御ビットは、同一の非ゼロの値にプログラムします。

**注：** デバイスの **RESET** 時は、**IPC** レジスタは初期化され、すべてのユーザー割り込みソースは、優先度 **4** に割り当てられます。

3. 関連する **IFSx** ステータスレジスタ内にある、周辺装置に関連する割り込みステータスビットをクリアします。
4. 適切な **IECx** 制御レジスタ内のソースに関連する割り込み許可制御ビットをセットし、割り込みソースを許可にします。

### 6.5.2 割り込みサービスルーチン

**ISR** を宣言し、正しいベクトルアドレスに **IVT** をイニシャライズするために用いられる方法は、プログラミング言語（すなわち、C もしくはアセンブラー）と、アプリケーションを開発するために用いられる言語開発ツールに依存します。一般的に、ユーザーは、**ISR** が取り扱う割り込みのソース用の、適切な **IFSx** レジスタ内にある割り込みフラグをクリアしなければなりません。そうでなければ、**ISR** は、ルーチンから抜け出た直後に再びルーチンに入ることになります。**ISR** がアセンブル言語でコーディングされている場合は、保存された PC 値、**SRL** 値および元の CPU 優先度をスタックから戻すために、**RETFIE** 命令を用いて、**ISR** を終了させる必要があります。

### 6.5.3 トラップサービスルーチン

トラップサービスルーチン (**TSR**) は、**TSR** に再び入ることを防ぐために **INTCON1** レジスタ内の適切なステータスフラグがクリアされる必要がある場合を除いて、**ISR** と同様にコーディングします。

### 6.5.4 割り込みの禁止

すべてのユーザー割り込みは、以下の手順により禁止にできます。

1. **PUSH** 命令を用いて、ソフトウェアスタックに現状の **SR** 値を入れます。
2. **SRL** の値と **0xE0** との論理和を取ることにより、CPU を優先度 **7** に設定します。

ユーザー割り込みを許可するために、**POP** 命令を用いて、元の **SR** 値を再格納します。

優先度 **7** もしくはそれ以下の値を持ったユーザー割り込みのみが禁止になることに注意してください。トラップソース（レベル **8**- レベル **15**）は禁止になりません。

**DISI** 命令により、固定時間内で、優先度 **1** ~ **6** の割り込みを容易に禁止にできます。レベル **7** の割り込みソースは **DISI** 命令では禁止できません。

表 6-3: 特殊関数レジスタ割り込みコントローラに関する

SFR名	ADR	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	RESET状態
INTCON1	0080	NSTDIS	—	—	—	—	—	OVATE	OVBTE	COVTE	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000 0000 0000 0000
INTCON2	0082	ALTIIV	—	—	—	—	—	—	—	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP	0000 0000 0000 0000
IFS0	0084	CNIF	BCLIF	I2CF	NVMIF	ADIF	U1TXIF	U1RXIF	SP11F	T3IF	T2IF	OC2IF	T1IF	OC1IF	IC1IF	INT0	0000 0000 0000 0000	
IFS1	0086	IC6IF	IC5IF	IC4IF	IC3IF	SP12IF	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000	
IFS2	0088	—	—	—	—	FLTBIF	FLTAIF	LVDIF	DC1IF	PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF	0000 0000 0000 0000
IEC0	008C	CNIE	BCLIE	I2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SP11IE	T3IE	T2IE	OC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000	
IEC1	008E	IC6IE	IC5IE	IC4IE	IC3IE	SP12IE	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000	
IEC2	0090	—	—	—	—	FLTBIE	FLTAIE	LVDIE	DC1IE	PWMIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE	0000 0000 0000 0000
IPC0	0094	—	T1IP<2:0>	—	—	OC1IP<2:0>	—	—	—	IC1IP<2:0>	—	—	INT0IP<2:0>	—	0100 0100 0100 0100	0100 0100 0100 0100	0100 0100 0100 0100	
IPC1	0096	—	T3IP<2:0>	—	—	T2IP<2:0>	—	—	—	OC2IP<2:0>	—	—	IC2IP<2:0>	—	0100 0100 0100 0100	0100 0100 0100 0100	0100 0100 0100 0100	
IPC2	0098	—	ADIP<2:0>	—	—	U1TXIP<2:0>	—	—	—	U1RXIP<2:0>	—	—	SP1IP<2:0>	—	0100 0100 0100 0100	0100 0100 0100 0100	0100 0100 0100 0100	
IPC3	009A	—	CNIP<2:0>	—	—	BC1IP<2:0>	—	—	—	I2CIP<2:0>	—	—	NVMIP<2:0>	—	0100 0100 0100 0100	0100 0100 0100 0100	0100 0100 0100 0100	
IPC4	009C	—	OC3IP<2:0>	—	—	IC8IP<2:0>	—	—	—	IC7IP<2:0>	—	—	INT1IP<2:0>	—	0100 0100 0100 0100	0100 0100 0100 0100	0100 0100 0100 0100	
IPC5	009E	—	INT2IP<2:0>	—	—	T5IP<2:0>	—	—	—	T4IP<2:0>	—	—	OC4IP<2:0>	—	0100 0100 0100 0100	0100 0100 0100 0100	0100 0100 0100 0100	
IPC6	00A0	—	C1IP<2:0>	—	—	SP12IP<2:0>	—	—	—	U2TXIP<2:0>	—	—	U2RXIP<2:0>	—	0100 0100 0100 0100	0100 0100 0100 0100	0100 0100 0100 0100	
IPC7	00A2	—	IC6IP<2:0>	—	—	IC5IP<2:0>	—	—	—	IC4IP<2:0>	—	—	IC3IP<2:0>	—	0100 0100 0100 0100	0100 0100 0100 0100	0100 0100 0100 0100	
IPC8	00A4	—	OC8IP<2:0>	—	—	OC7IP<2:0>	—	—	—	OC6IP<2:0>	—	—	OC5IP<2:0>	—	0100 0100 0100 0100	0100 0100 0100 0100	0100 0100 0100 0100	
IPC9	00A6	—	PWMIP<2:0>	—	—	C2IP<2:0>	—	—	—	INT4IP<2:0>	—	—	INT3IP<2:0>	—	0100 0100 0100 0100	0100 0100 0100 0100	0100 0100 0100 0100	
IPC10	00A8	—	FLTAIP<2:0>	—	—	—	—	—	—	LVDIP<2:0>	—	—	DC1IP<2:0>	—	QEIIP<2:0>	—	0100 0100 0100 0100	
IPC11	00AA	—	—	—	—	—	—	—	—	—	—	—	—	—	FLTBIIP<2:0>	—	0000 0000 0000 0000	

注：すべての割り込みソースと関連する制御ビットは特定のデバイスでは利用できません。詳しくは、デバイスのデータシートを参照してください。

## 6.6 設計の秘訣

**質問 1:** 2つの割り込みソースが同時に発生し同じユーザー割り当て優先度を持つ場合は、どうなりますか？

回答：一番高い自然優先度を持った割り込みソースが優先されます。自然順優先度は、そのソースの割り込みベクトル表 (IVT) のアドレスで決定されます。より小さい IVT アドレスを持つ割り込みソースがより高い自然順優先度を持ちます。

**質問 2:** すべての割り込みソースとトラップを禁止にするために *DISI* 命令を使用することはできますか？

回答：*DISI* 命令はトラップや優先度 7 の割り込みソースを禁止にすることはできません。ただし、優先度 7 の割り込みがユーザー-application 内で有効にされない場合、*DISI* 命令は、すべての割り込みソースを禁止にする簡便な方法として使用されます。

## 6.7 関連するアプリケーションノート

この章では、マニュアルのこの章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正して使用可能です。制限がある場合もあります。現状、割り込みモジュールに関連する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 #

現在のところ、関連するアプリケーションノートはありません。

**注：** dsPIC30F ファミリーのデバイスに関しての、その他のアプリケーションノートやコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 6.8 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

この版は、dsPIC30F の割り込みモジュールに関する追加の技術情報を含みます。



## 第7章 . 発振器

## ハイライト

この章は、以下の項目を含んでいます。

7.1	はじめに .....	7-2
7.2	CPU クロックの仕組み .....	7-4
7.3	発振器のコンフィギュレーション .....	7-5
7.4	発振器コントロールレジスタ (OSCCON) .....	7-6
7.5	主発振器 .....	7-8
7.6	水晶発振子 / セラミック発振子 .....	7-9
7.7	水晶、クロックモード C1、C2 および Rs の最適値を決定する .....	7-11
7.8	外部クロック入力 .....	7-12
7.9	外部 RC 発振器 .....	7-13
7.10	フェーズロックループ (PLL) .....	7-17
7.11	低電力 32 kHz 水晶発振器 .....	7-18
7.12	発振器スタートアップタイマ (OST) .....	7-18
7.13	内蔵高速 RC 発振器 (FRC) .....	7-18
7.14	内蔵低電力 RC 発振器 (LPRC) .....	7-19
7.15	フェールセーフクロックモニター (FSCM) .....	7-19
7.16	プログラマブル発振器ポストスケーラ .....	7-20
7.17	クロック切替え動作 .....	7-21
7.18	設計の秘訣 .....	7-25
7.19	関連するアプリケーションノート .....	7-26
7.20	改訂履歴 .....	7-27

## 7.1 はじめに

この章では dsPIC30F の発振器システムとその動作について述べます。dsPIC30F 発振器システムは以下のようなモジュールと特徴を持っています。

- クロックソースとして、種々の外部、内蔵発振器のオプション対応可能
- 内蔵発振周波数を上げるための PLL 内蔵
- 種々のクロックソースを切替えてクロック供給可能
- システムの電力低減のためのプログラマブルクロックポストスケーラ内蔵
- クロック不良の検出とフェールセーフ対策可能なフェールセーフクロックモニター (FSCM) 内蔵
- クロック制御用レジスタ (OSCCON) を内蔵
- 主発振器選択用の不揮発のコンフィギュレーションビット内蔵

発振器システムの簡単な図を図 7-1 に示します。

### 7.1.1 発振器システムの特徴概要

#### 発振器ソース：

- 複数のクロックモードを持つ主発振器
- 副発振器（低電力 32KHz 水晶発振器）
- FRC 発振器：高速内蔵 RC (8 MHz)
- LPRC 発振器：低電力内蔵 RC (512KHz)

#### PLL クロック倍増器

- XT もしくは EC クロックモードで、主発振器で動作します。
- 4MHz-10MHz の入力周波数範囲を持ちます。
- 4 倍ゲインモード ( $F_{OUT} = 16 \text{ MHz}-40 \text{ MHz}$ )
- 8 倍ゲインモード ( $F_{OUT} = 32 \text{ MHz}-80 \text{ MHz}$ )
- 16 倍ゲインモード ( $F_{OUT} = 64 \text{ MHz}-120 \text{ MHz}$ )
- PLLVCO はロック表示とロック外れトラップのオプションを持ちます。

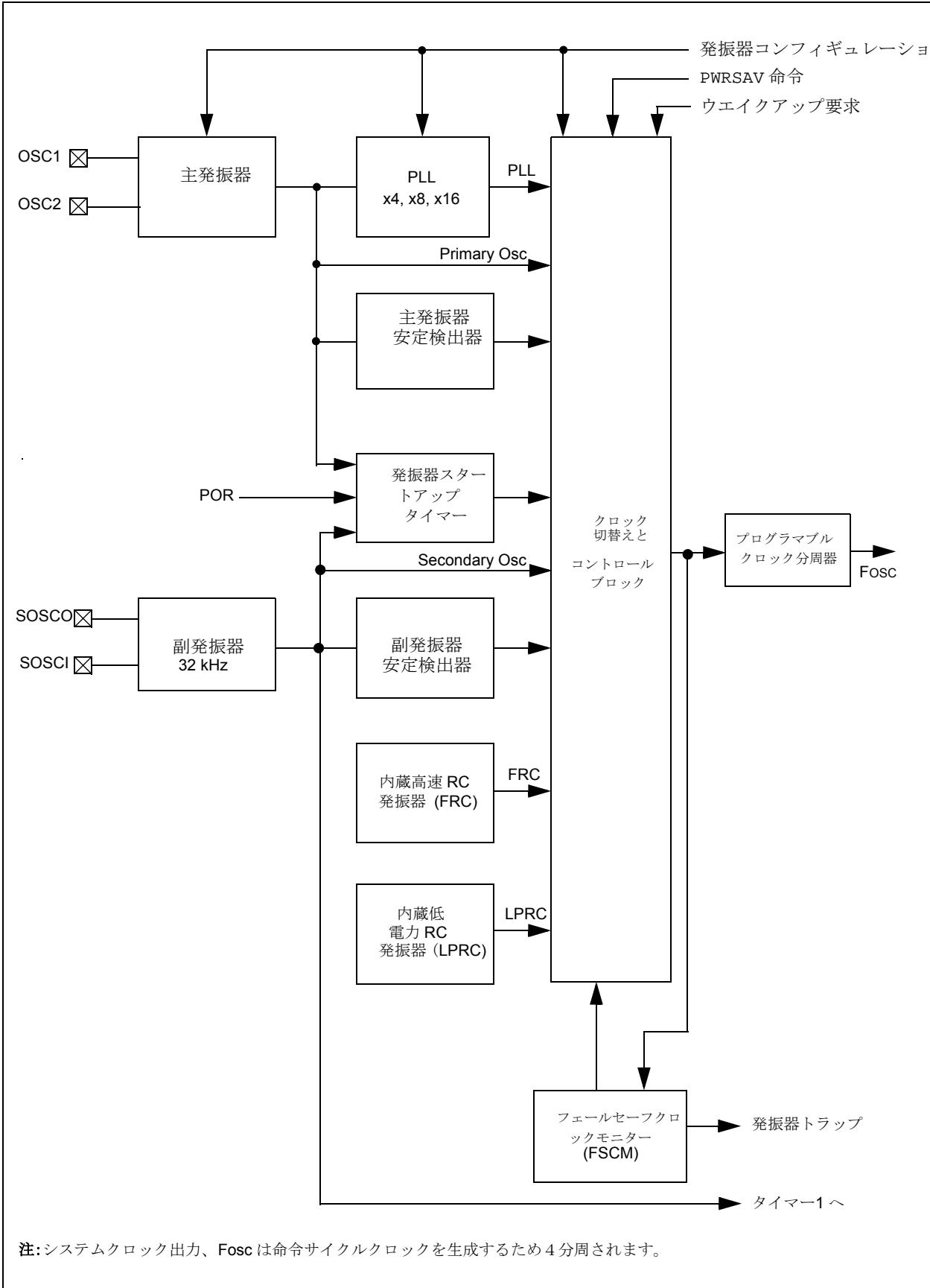
#### クロックスケーリングオプション：

- デバイスロック用の汎用ポストスケーラーを持ちます。

#### フェールセーフクロックモニター (FSCM) :

- クロック不良を検出し、内蔵 FRC 発振器へと切替えます。

図 7-1: 発振器システムブロック図



注: システムクロック出力、Fosc は命令サイクルクロックを生成するため 4 分周されます。

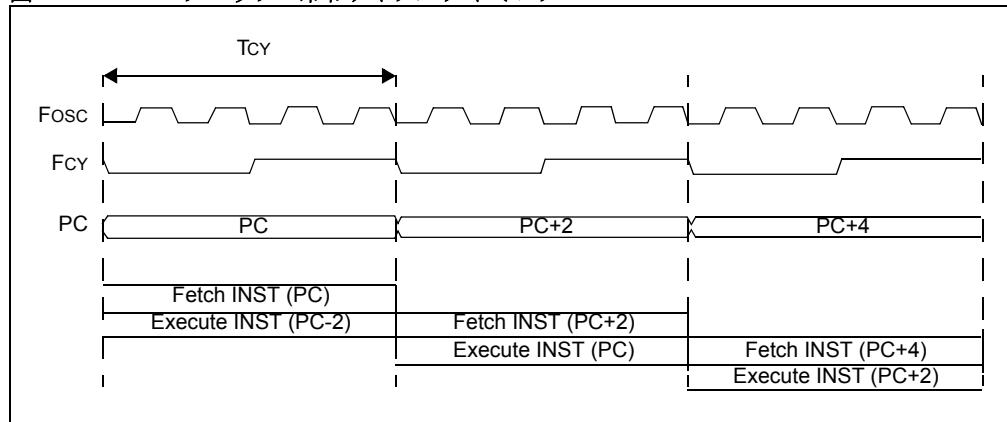
## 7.2 CPU クロックの仕組み

図 7-1 をご参照ください。システムクロックソースは 4 つのソースの 1 つから供給されます。これらのソースは、主発振器、副発振器、内蔵高速 RC 発振器 (FRC)、もしくは、低電力 RC 発振器 (LPRC) です。主発振器は内部 PLL 回路を使うこともできます。選択されたクロックソースの周波数は、プログラマブルクロック分周器で分周することも可能です。プログラマブルクロック分周器からの出力はシステムクロックソース Fosc になります。

システムクロックソースは、内部命令サイクル FCY を生成するために 4 分周されます。この文書では、命令サイクルは Fosc/4 とも記述されます。図 7-2 のタイミング図にシステムクロックソースと命令実行との関係を示します。

内部命令サイクルクロック Fosc/4 は、主発振器のある動作モードでは、OSC2 I/O ピンに出力されます（セクション 7.3 「発振器のコンフィギュレーション」参照）。

図 7-2: クロック / 命令サイクルタイミング



### 7.3 発振器のコンフィギュレーション

デバイスのパワーオンリセット時に用いられる発振ソース（および動作モード）は、不揮発コンフィギュレーションビットを用いて選択できます。発振器コンフィギュレーションビットは、FOSC コンフィギュレーションレジスタ内に配置されています（詳しくは第24章、「デバイスコンフィギュレーション」参照）。

**FOS<1:0>** コンフィギュレーションビット (Fosc<9:8>) はパワーオンリセット時に用いられる発振器ソースを選択します。主発振器がデフォルトの選択（未プログラムのとき）です。副発振器もしくは内蔵発振器の1つが、これらのコンフィギュレーションビット位置をプログラムすることで選択できます。

**FPR<3:0>** コンフィギュレーションビット (Fosc<3:0>) は主発振器の動作モードを選択します。13の動作モードのうちの1つが、以下の表7-1に示されるように選択されます。

表7-1: クロック選択用コンフィギュレーションビット値

発振器モード	発振ソース	FOS<1:0>		FPR<3:0>			OSC2ピン その他の機能
ECとPLL 16x	主発振器	1	1	1	1	1	I/O (注4)
ECとPLL 8x	主発振器	1	1	1	1	0	I/O
ECとPLL 4x	主発振器	1	1	1	1	0	I/O
ECIO	主発振器	1	1	1	1	0	I/O
EC	主発振器	1	1	1	0	1	Fosc/4
予約	主発振器	1	1	1	0	1	なし
ERC	主発振器	1	1	1	0	0	Fosc/4
ERClO	主発振器	1	1	1	0	0	I/O
XTとPLL 16x	主発振器	1	1	0	1	1	(注3)
XTとPLL 8x	主発振器	1	1	0	1	1	(注3)
XTとPLL 4x	主発振器	1	1	0	1	0	(注3)
XT	主発振器	1	1	0	1	0	(注3)
HS	主発振器	1	1	0	0	1	(注3)
XTL	主発振器	1	1	0	0	0	(注3)
LP	副発振器	0	0	—	—	—	(注1, 2)
FRC	内蔵	0	1	—	—	—	(注1, 2)
LPRC	内蔵	1	0	—	—	—	(注1, 2)

- 注 1: OSC2 ピン機能は、主発振器モード選択 (FPR<3:0> コンフィギュレーションビット) により決定されます。  
 2: OSC1 ピンは、副発振器もしくは内部クロックソースが常に選択されても、I/O ピンとして使用できない点に注意してください。  
 3: これらの発振モードでは、水晶発振子は OSC1 と OSC2 ピンの間に接続されます。  
 4: これは、未プログラムのときの（消去された）デバイス用のデフォルト発振モードです。未プログラムのときのコンフィギュレーションビットは、「1」の値をとります。

#### 7.3.1 クロック切替モードコンフィギュレーションビット

**FCKSM<1:0>** コンフィギュレーションビット (Fosc<15:14>) は、デバイスクロック切替とフェールセーフクロックモニター (FSCM) を有効 / 無効にするために使用されます。これらのビットがプログラムされないと（デフォルト）、クロック切替と FSCM は無効になります。

## 7.4 発振器コントロールレジスタ (OSCCON)

OSCCON コントロールレジスタは、クロックの切替えとクロックソースのステータス情報を提供します。

COSC<1:0> ステータスピット (OSCCON<13:12>) は、デバイスが動作している発振ソースを表示する読み出し専用のビットです。COSC<1:0> は、パワーオンリセット時に FOS<1:0> コンフィギュレーションビット値にセットされ、クロック切替え動作の最後で、新しい発振ソースに変化します。

NOSC<1:0> ステータスピット (OSCCON<9:8>) はクロック切替え動作用の新しいクロックソースを選択するコントロールビットです。NOSC<1:0> ビットは、パワーオンリセット時もしくはブラウンアウトリセット時に FOS<1:0> コンフィギュレーションビットにセットされ、クロック切替え動作中にユーザーソフトウェアで変更されます。

POST<1:0> コントロールビット (OSCCON<8:7>) は、システムクロックの分周比をコントロールします。

LOCK ステータスピット (OSCCON<5>) は PLL 回路のステータスを表示する読み出し専用ビットです。

CF ステータスピット (OSCCON<3>) はクロック不良を表示する読み書き可能なステータスピットです。

LPOSCEN コントロールビット (OSCCON<1>) は、32KHz 低電力水晶発振器を有効もしくは無効にするために用いられます。

OSWEN コントロールビット (OSCCON<0>) はクロック切替え動作を起動するために用いられます。OSWEN ビットは、クロック切替えがうまくいった後で自動的にクリアされます。

**注：** OSCCON レジスタは、デバイスクロック切替え機構を制御するので、書き込み保護がかけられています。ユーザーソフトウェアは、レジスタに書き込むために、あるコードシーケンスを実行する必要があります。詳しくは、セクション [7.4.1 「OSCCON への書き込みに対する保護」](#) を参照してください。

### 7.4.1 OSCCON への書き込みに対する保護

OSCCON レジスタへの書き込みは、意図的に難しくなっています。このレジスタはクロック切替えやクロックのスケーリングを制御しているためです。

OSCCON の下位バイトを書き込むには、以下のコードシーケンスを、間に他の命令を挟むことなく、実行する必要があります。

OSCCONL に 0x46 をバイト書き込み。

OSCCONL に 0x57 をバイト書き込み。

このシーケンスの後、OSCCONL へのバイト書き込みが、1 命令サイクルの間のみ許可されます。希望の値を書き込むには、ビット操作命令を使用します。

OSCCON の上位バイトを書き込むには、以下のコードシーケンスを、間に他の命令を挟むことなく、実行する必要があります。

OSCCONH に 0x78 をバイト書き込み

OSCCONH に 0x9A をバイト書き込み

このシーケンスの後、OSCCONH へのバイト書き込みが、1 命令サイクルの間のみ許可されます。希望の値を書き込むには、ビット操作命令を使用します。

## レジスタ 7-1: OSCCON: 発振器コントロールレジスタ

上位バイト:							
U-0	U-0	R-y	R-y	U-0	U-0	R/W-y	R/W-y
—	—	COSC<1:0>	—	—	—	NOSC<1:0>	—
ビット 15							ビット 8

下位バイト:							
R/W-0	R/W-0	R-0	U-0	R/W-0	U-0	R/W-0	R/W-0
POST<1:0>	LOCK	—	CF	—	LPOSCEN	OSWEN	—
ビット 7							ビット 0

ビット 15-1 未実装: '0' が読み出されます。

4

ビット 13-1 **COSC<1:0>**: 現発振ソースステータスビット

2

11 = 主発振器

10 = 内蔵 LPRC 発振器

01 = 内蔵 FRC 発振器

00 = 低電力 32KHz 水晶発振器 (タイマー 1)

ビット 11-1 未実装: '0' が読み出されます。

0

ビット 9-8 **NOSC<1:0>**: 新しい発振器グループの選択ビット

11 = 主発振器

10 = 内蔵 LPRC 発振器

01 = 内蔵 FRC 発振器

00 = 低電力 32KHz 水晶発振器 (タイマー 1)

ビット 7-6 **POST<1:0>**: 発振器ポストスケーラー選択ビット

11 = 発振器ポストスケーラーがクロックを 64 分周します。

10 = 発振器ポストスケーラーがクロックを 16 分周します。

01 = 発振器ポストスケーラーがクロックを 4 分周します。

00 = 発振器ポストスケーラーはクロックを変更しません。

ビット 5 **LOCK**: PLL ロックステータスビット

1 = PPL がロック状態であることを示します。

0 = PLL がロック外れ (もしくは無効) であることを示します。

ビット 4 未実装: '0' が読み出されます。

ビット 3 **CF**: クロック不良ステータスビット

1 = FSCM がクロック不良を検出しました。

0 = FSCM はクロック不良を未検出です。

ビット 2 未実装: '0' が読み出されます。

ビット 1 **LPOSCEN**: 32 kHz LP 発振器有効ビット

1 = LP 発振器は有効です。

0 = LP 発振器は無効です。

ビット 0 **OSWEN**: 発振器切替え有効ビット

1 = NOSC&lt;1:0&gt; ビットで規定される選択肢への発振器切替えを要求します。

0 = 発振器切替えは完了しました。

凡例:

R = 読み出し可能ビット

W = 書き込み可能

U = 未実装ビット、'0' が読み出されます。

ビット

-n = POR の値

'1' = ビットがセット '0' = ビットがクリア x = ビットは不定です  
されますy = POR もしくは BOR のコンフィギュレーション  
ビットから設定される値

## 7.5 主発振器

主発振器は、dsPIC30F デバイスファミリの OSC1 と OSC2 のピン間で利用可能です。主発振器は表 7-2 に纏められているように、13 の動作モードを持ちます。一般的に、主発振器は、外部クロック入力、外部 RC ネットワーク、もしくは外部水晶用で構成されます。主発振器の動作モードの詳細については、次のセクションに記述されています。

FPR<3:0> コンフィギュレーションビット (Fosc<3:0>) は主発振器の動作モードを選択します。dsPIC30F は、OSCCON レジスタ (OSCCON<13:12>) 内の主発振器選択制御ビットが ‘11b’ にセットされた時はいつでも、主発振器から動作を始めます。

表 7-2: 主発振器動作モード

発振器モード	説明	OSC2 ピンのその他の機能
EC	外部クロック入力 (0-40 MHz)	Fosc/4
ECIO	外部クロック入力 (0-40 MHz)、OSC2 ピンは I/O	I/O
EC と PLL 4x	外部クロック入力 (4-10 MHz)、OSC2 ピンは I/O, 4x PLL が有効	I/O
EC と PLL 8x	外部クロック入力 (4-10 MHz)、OSC2 ピンは I/O, 8x PLL が有効	I/O
EC と PLL16x	外部クロック入力 (4-7.5 MHz)、OSC2 ピンは I/O, 16x PLL が有効	I/O
ERC	外部 RC 発振器、OSC2 ピンは Fosc/4 出力	Fosc/4
ERCIO	外部 RC 発振器、OSC2 ピンは I/O	I/O
XT	4 MHz-10 MHz 水晶	(注 1)
XT と PLL 4x	4 MHz-10 MHz 水晶, 4x PLL が有効	(注 1)
XT と PLL 8x	4 MHz-10 MHz 水晶, 8x PLL が有効	(注 1)
XT と PLL 16x	4 MHz-7.5 MHz 水晶, 16x PLL が有効	(注 1)
XTL	200 kHz-4 MHz 水晶	(注 1)
HS	10MHz-25 MHz 水晶	(注 1)

注 1: これらの発振モードでは、外部水晶発振子は OSC1 と OSC2 に接続されます。

### 7.5.1 発振器モード選択ガイドライン

XT、XTL と HS モードの間の主な違いは、発振回路内の内蔵インバーターのゲインの違いで、これにより異なる周波数範囲をカバーできます。一般的には、仕様に合う最小値のゲインを持つ発振器オプションを使用します。これにより、DC 電流 (IDD) を低くできます。それぞれの発振モードの周波数範囲は、周波数カットオフの推奨値ですが、異なるゲインモードを選択する場合は、十分な検証が必要です（電圧、温度、抵抗や容量や内蔵発振回路といったデバイスの変動を含めた検証）。

発振器のフィードバック回路はすべての EC や ECIO モードでは無効です。OSC1 ピンはハイインピーダンス入力で CMOS ドライバでドライブ可能です。

ERC と ERCIO モードによりデバイス発振器を低価格で構成できます。（外付けの抵抗と容量が必要なだけです）これらのモードはまた、発振周波数の範囲を最大にできます。

主発振器は外部クロック入力はもしくは外部 RC ネットワーク用に構成されている場合、OSC2 ピンは発振機能をサポートするためには不要です。これらのモードのときには、OSC2 ピンは追加のデバイス I/O ピンもしくはクロック出力ピンとして使用できます。OSC2 ピンがクロック出力ピンとして使用される場合は、出力周波数は Fosc/4 です。

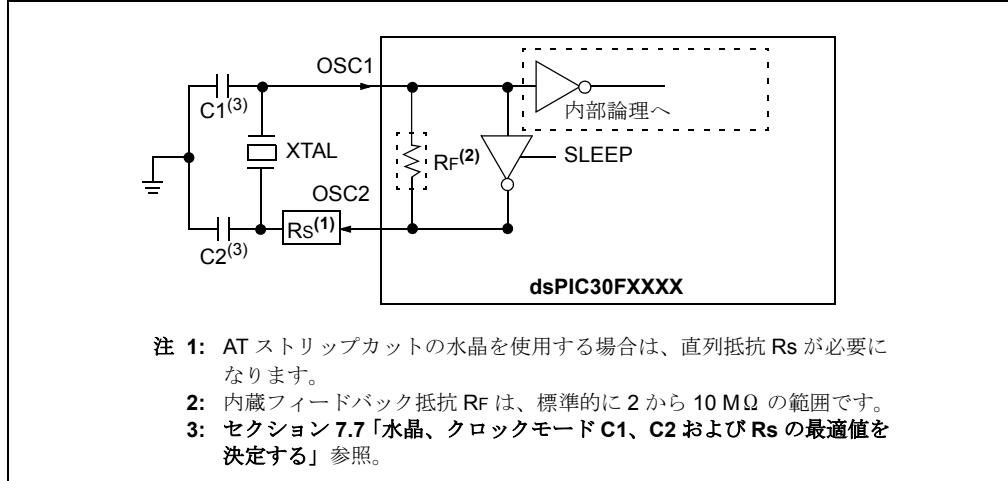
XTL モードは低電力低周波数モードです。このモードの発振器は、3 つの水晶発振モードの中でも最も電力が少なくてすみます。XT モードは中電力中速周波数モードで、HS モードは水晶発振子を用いた中で、最高速の発振周波数を提供できます。

PLL 回路を使用する EC と XT モードでは最高速のデバイス動作周波数を提供できます。発振回路は、発振周波数を倍するために PLL を有効にするので、これらのモードの中でも最も電流を消費します。

## 7.6 水晶発振子 / セラミック発振子

XT、XTL および HS モードにおいて、水晶もしくはセラミック発振子は、発振を確立させるために、OSC1 と OSC2 ピンに接続されます。(図 7-3) dsPIC30F 発振の設計のために、並列カットの水晶を使用することが必要です。直列カットの水晶を使用すると、水晶製造者の仕様から外れた周波数になります。

図 7-3: 水晶もしくはセラミック発振子動作 (XT、XTL もしくは HS 発振モード)



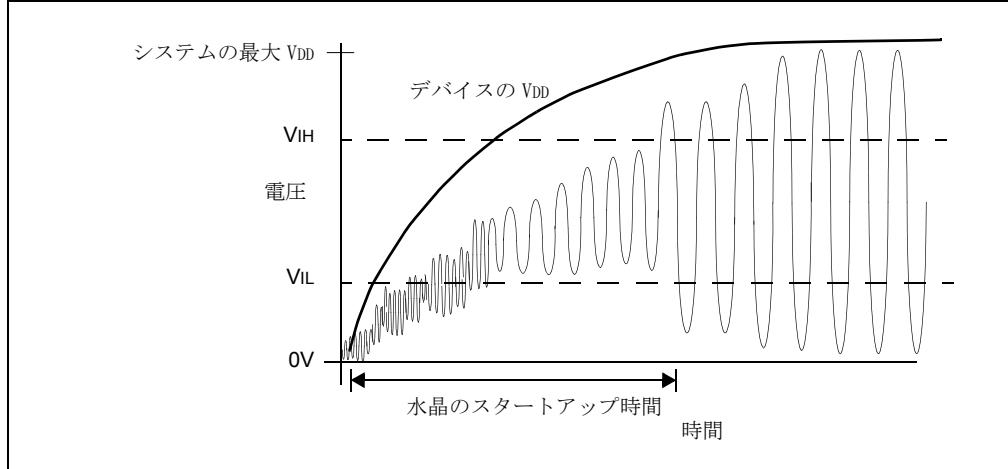
### 7.6.1 発振子 / 共振子のスタートアップ

デバイス電圧が  $V_{SS}$  から増加するにつれて、発振子はその発振を開始します。発振子が発振を開始するのに必要な時間は多くの要素に依存します。それらは以下を含みます。

- ・水晶 / 共振子セラミックの周波数
- ・使用する容量 (図 7-3 の  $C1$  と  $C2$ )
- ・デバイス  $V_{DD}$  の立ち上がり時間
- ・システムの温度
- ・使用される場合は、直列抵抗の値とそのタイプ (図 7-3 の  $Rs$ )
- ・デバイスの発振モード選択 (内蔵発振器のインバータのゲインを選択します)
- ・水晶の品質
- ・発振回路のレイアウト
- ・システムノイズ

図 7-4 は典型的な発振子 / 共振子のスタートアップの波形プロットを示します。

図 7-4: 発振子のスタートアップ特性



## 7.6.2 発振回路の調整

Microchip のデバイスは広い動作範囲（周波数、電圧および温度において、デバイスと注文された版に依存します）を持ち、外付けデバイス（水晶、容量等）は品質も製造者も広範囲になりますので、デバイス選択がアプリケーションの要求を確実に満たすようにするために、動作の検証を行う必要があります。

これらの外付けデバイスの選択と配置に依存する多くの要素があります。それらは以下の項目を含みます。

- アンプのゲイン
- 希望の周波数
- 水晶の共振周波数
- 動作温度
- 供給電源範囲
- スタートアップ時間
- 安定性
- 水晶の寿命
- 消費電力
- 回路の簡素化
- 標準デバイスの使用
- デバイス点数

## 7.6.3 スリープモードからの発振スタートアップ

発振器をスタートアップするのに一番難しい時間は、スリープモードから起動する時です。これは、負荷容量が両方とも部分的にある値までチャージされ、起動時の位相差が最小であるからです。従って、安定的な発振を得るためにには、より時間が必要です。低電圧、高温かつ低周波数クロックモードの場合も、ループゲインに制約が加わり、スタートアップに影響を与えることにも注意してください。以下の要素もそれぞれスタートアップ時間を増加させます。

- 低周波数の設計（低ゲインクロックモード）
- 静寂な環境（電池駆動デバイスのような場合）
- シールドされた箱の中での動作（ノイズの多いRF環境から離れている場合）
- 低電圧
- 高温
- スリープモードからの起動

発振器に“キックスタート”を与えるので、ノイズは実際のところ発振器のスタートアップ時間を見減らせるのに役立ちます。

## 7.7 水晶、クロックモード C1、C2 および $R_s$ の最適値を決定する

デバイスを選択するのに一番良い方法は、知識に基づいて行うよりも、多くの試行、測定およびテストを実行することです。

水晶は、通常並列共振周波数のみにより選択されますが、その他のパラメータ、温度もしくは周波数許容値といった値も設計のためには重要です。アプリケーションノート AN588『PICマイクロコントローラ発振器設計ガイド』は、水晶発振やその注文情報についてより詳しく学ぶにはすぐれた参考文献です。

dsPIC30F 内蔵発振回路は並列発振回路で、並列共振水晶を選択することが必要です。負荷容量は通常  $22\text{pF}$  から  $33\text{pF}$  の範囲で規定されます。この範囲の負荷容量を付けると、水晶は要求周波数に最も近い値で発振します。他の利点を達成するためには、後述するように、これらの値を変更する必要もあるかもしれません。

クロックモードは、初めは水晶発振子の求められる周波数に基づいて選択されます。XT, XTL と HS モードの間の主な違いは、発振回路内の内蔵インバーターのゲインの違いで、これにより異なる周波数範囲をカバーできます。一般的には、仕様に合う最小値のゲインを持つ発振器オプションを使用します。これにより、DC 電流 ( $\text{IDD}$ ) を低くできます。それぞれの発振モードの周波数範囲は、周波数カットオフの推奨値ですが、異なるゲインモードを選択する場合は、十分な検証が必要です（電圧、温度、抵抗や容量や内蔵発振回路といったデバイスの変動を含めた検証）。

C1 と C2 (図 7-3 参照) は、初めは、水晶製造者やデバイスのデータシートに掲載された表で示される負荷容量に基づいて選択されるべきです。水晶製造者や電源やその他すでに述べられた要素により、発振回路は、工場での性能評価過程で使用されたものとは異なる回路になりますので、デバイスのデータシートで与えられる値のみが、開始点として使用されます。

理想的には、回路が動作すべき最高温度と最低電圧で発振するように、容量が選択されます。高温と低  $V_{DD}$  はともにループゲインに制約を与え、回路がこの極端な環境で動作する場合、設計者は、その他の温度と電源の組合せで正しい動作をすることを確認できます。出力正弦波は最高ゲインの環境（最高  $V_{DD}$  と最低温度）でもクリップされではありませんし、正弦波出力振幅は最低ゲインの環境（最低  $V_{DD}$  と最高温度）でも、デバイスのデータシートに記載されているクロックの論理入力仕様より十分大きくなればなりません。

スタートアップを改善する方法は、C1 より大きな値の C2 を使用することです。これによりスタートアップ時に水晶により大きな位相差を発生させ、発振スタートアップを加速します。

適切な周波数応答用に水晶に負荷を与える以外にも、これらの容量は、ループゲインが増加した場合に、それを下げる効果を持ちます。C2 は回路全体のゲインに影響を与えるように選択できます。C2 を大きくすると、水晶がオーバードライブされている場合は、ゲインを下げることができます ( $R_s$  に関する議論も参照)。容量値が高すぎると水晶に過電流が流れます、従って、C1 と C2 は過度に大きくできません。残念ながら水晶で消費される電力を測定することは困難ですが、推奨値からそれほど離れていないければ、これを心配する必要はありません。

他の外付けデバイスが満足できる状態に選択された後でも、水晶がオーバードライブされている場合は、直列抵抗  $R_s$  を回路に追加します。これは、OSC2 ピン（ドライブされるピンですが）をオシロスコープで見ることで決定されます。OSC1 にプローブを接続するとピンは過負荷になり、性能に悪影響を与えます。スコープのプローブは、それ自身が持つ容量を回路に追加することになりますので、これは設計された回路（すなわち、回路が C2 が  $22\text{pF}$  で一番良く動作し、スコープのプローブが  $10\text{pF}$  であれば、 $33\text{pF}$  容量が実際にかけられていることになる）で考慮する必要があります。出力信号はクリップされたり平坦になったりしてはいけません。水晶をオーバードライブすることは、回路がより高次の高調波にジャンプするか、水晶にダメージを与えることになります。

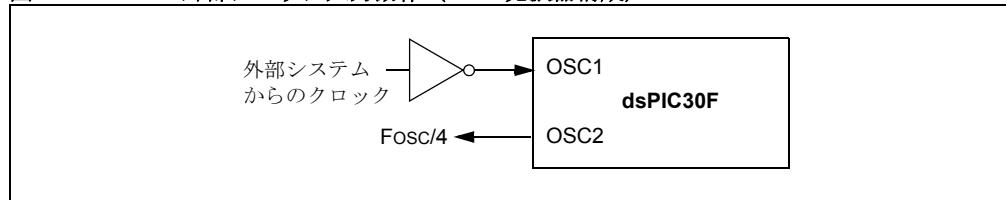
OSC2 信号は、クロック入力ピンの入力最小値と最大値（5V VDD の場合 4V から 5V のピーク TO ピークの値が通常好ましい）に振れるきれいな正弦波でなければなりません。これを設定するための簡易な方法としては、設計回路が動作する予定の最低温度と最大 VDD で回路を再度テストし、出力を観測することです。これによりクロック出力の最大振幅が得られるはずです。もし、VDD や VSS の近傍でクリッピングや正弦波の歪が発生したら、増加した負荷容量のために、水晶を通して過電流が流れているか、製造者が規定した負荷から遠く離れているせいかもしれません。水晶電流を調整するには、水晶インバーター出力ピンと C2 の間にトリマーポテンショニメーターを付加し、正弦波がきれいになるまで調整します。水晶は、低温・高 VDD の極値では、最もドライブ電流が多くなります。トリマーポテンショニメーターは、オーバードライブを防ぐためにこれらの制限範囲内となるように調整する必要があります。ここで、標準値に近い直列抵抗  $R_s$  がトリマーポテンショニメーターの代わりに使用できます。 $R_s$  が高すぎる、おそらく  $20\text{k}\Omega$  以上の場合は、出力から入力があまりにも隔絶されることになり、クロックがノイズの影響を受けやすくなります。水晶がオーバードライブするのを防ぐためにこの高い値が必要な場合は、C2 を増加し補正するか、発振動作モードを変えてみてください。 $R_s$  が 10k 位かもしくはそれ以下で、負荷容量が製造者推奨値からそれほど遠くない値での組合せを試してみてください。

## 7.8 外部クロック入力

主発振モードのうちの 2 つは外部クロックを使用します。それらのモードは EC と ECIO です。

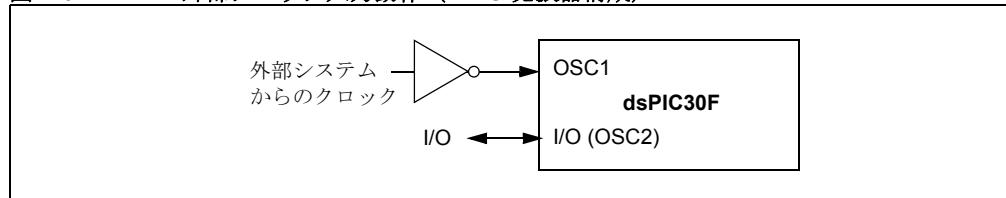
EC モード（図 7-5）では、OSC1 ピンを CMOS ドライバーでドライブします。このモードでは、OSC1 ピンはハイインピーダンスで OSC2 ピンはクロック出力 ( $F_{osc}/4$ ) です。この出力クロックは、テスティングもしくは同期用として使用できます。

図 7-5: 外部クロック入力動作 (EC 発振器構成)



ECIO モード（図 7-6）では、OSC1 ピンを COMS ドライバーでドライブします。このモードでは、OSC1 ピンはハイインピーダンスで OSC2 ピンは汎用 I/O ピンになります。OSC1 と OSC2 の間のフィードバックデバイスは、電流を節約するために切断されます。

図 7-6: 外部クロック入力動作 (ECIO 発振器構成)



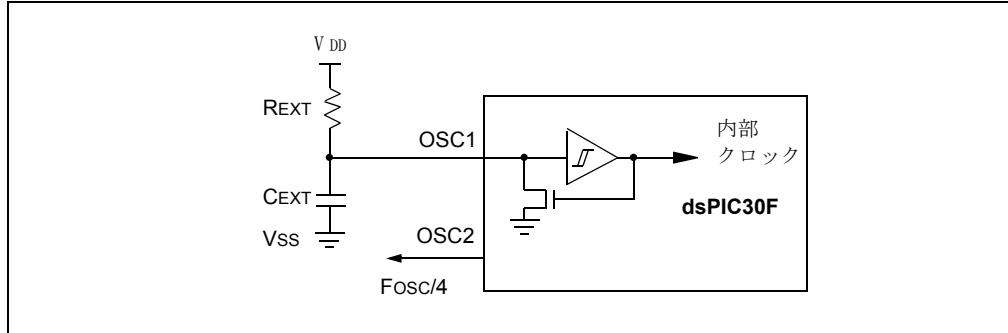
## 7.9 外部 RC 発振器

タイミングにそれほど敏感でないアプリケーションでは、主発振に ERC と ERCIO モードを使用するとさらにコストを節約できます。RC 発振周波数は以下の関数になります。

- 電源電圧
- 外部抵抗 ( $R_{EXT}$ ) 値
- 外部容量 ( $C_{EXT}$ ) 値
- 動作温度

これに加えて、発振周波数は通常のプロセスパラメータ変動によりユニット毎に変わります。さらに、特に低  $C_{EXT}$  の場合、パッケージタイプ間のリードフレーム容量の差により発振周波数が影響されます。ユーザーは、使用される外部  $R_{EXT}$  と  $C_{EXT}$  のデバイスの許容値による変動も考慮する必要があります。図 7-7 に RC の組合せをどのように接続するかを示します。 $2.2\text{k}\Omega$  以下の  $R_{EXT}$  の値では、発振動作は不安定になるか全く停止してしまうかもしれません。大変高い  $R_{EXT}$  の値（例えば  $1\text{M}\Omega$ ）では、発振器はノイズや湿度やリーク電流に敏感になります。従って、 $R_{EXT}$  値としては  $3\text{k}\Omega$  から  $100\text{k}\Omega$  の間の値を使用することをお奨めします。

図 7-7: ERC 発振器モード



発振器は外部容量が無くても ( $C_{EXT} = 0\text{ pF}$ ) 動作しますが、ノイズや安定性の理由で  $20\text{pF}$  以上の値を使うべきです。外部容量が無いか小さい場合は、PCB のトレース容量やパッケージのリードフレーム容量のような外部容量の変化により、発振周波数が大幅に変化します。

4 分周された発振周波数が OSC2/CLKO ピンに現れ、テストやその他ロジックとの同期のために使用できます。

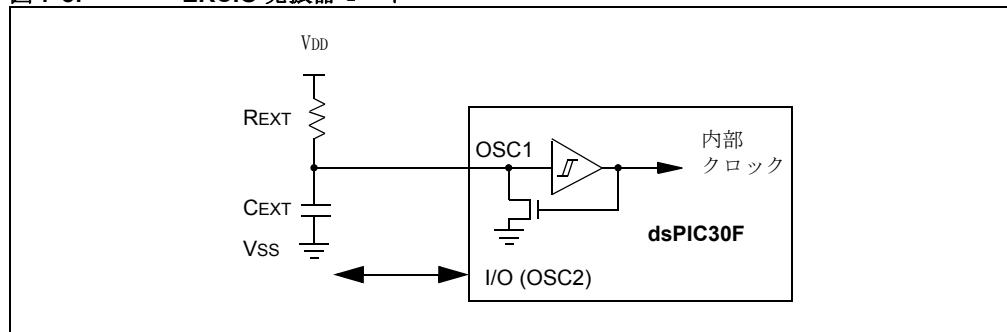
**注:** 発振器が ERC もしくは ERCIO モードの時には、外部クロックソースを OSC1 ピンに接続してはいけません。

## 7.9.1 I/O イネーブルの付いた外部 RC 発振

ERCIO 発振モードは ERC 発振モードと全く同じ方法で機能します。唯一の違いは OSC2 ピンが I/O ピンとなることです。

RC モードでは、ユーザーは、使用される外部 REXT と CEXT のデバイスの許容値による変動、プロセス変動、電圧および温度も考慮する必要があります。図 7-8 に I/O ピンと RC の組合せをどのように接続するかを示します。

図 7-8: ERCIO 発振器モード



## 7.9.2 外部 RC のスタートアップ

RC 発振器に関するスタートアップ遅延はありません。VDD が供給されると直ぐ発振を開始します。

**注:** デバイスがコードを実行する前に、ユーザーは VDD が仕様以内であることを確認する必要があります。

## 7.9.3 RC 動作周波数

以下のグラフは、RC デバイスの値に対するデバイス電圧の関数として、外部 RC 発振周波数を示しています。

**注:** 以下のグラフは、RC デバイスの選択用として概略のガイドラインとしてのみ使用して下さい。実際の周波数は、システム温度やデバイスにより変わります。RC 発振特性データの詳細については特定デバイスのデータシートを参照してください。

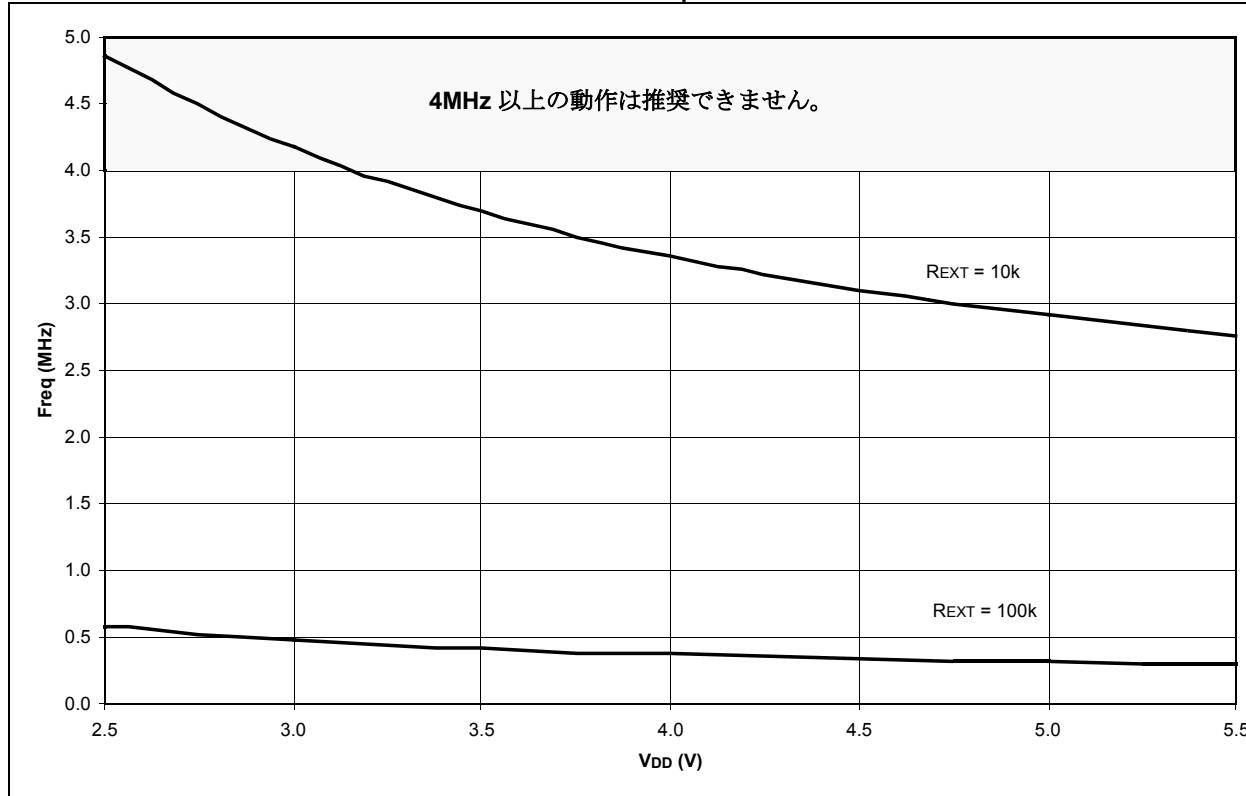
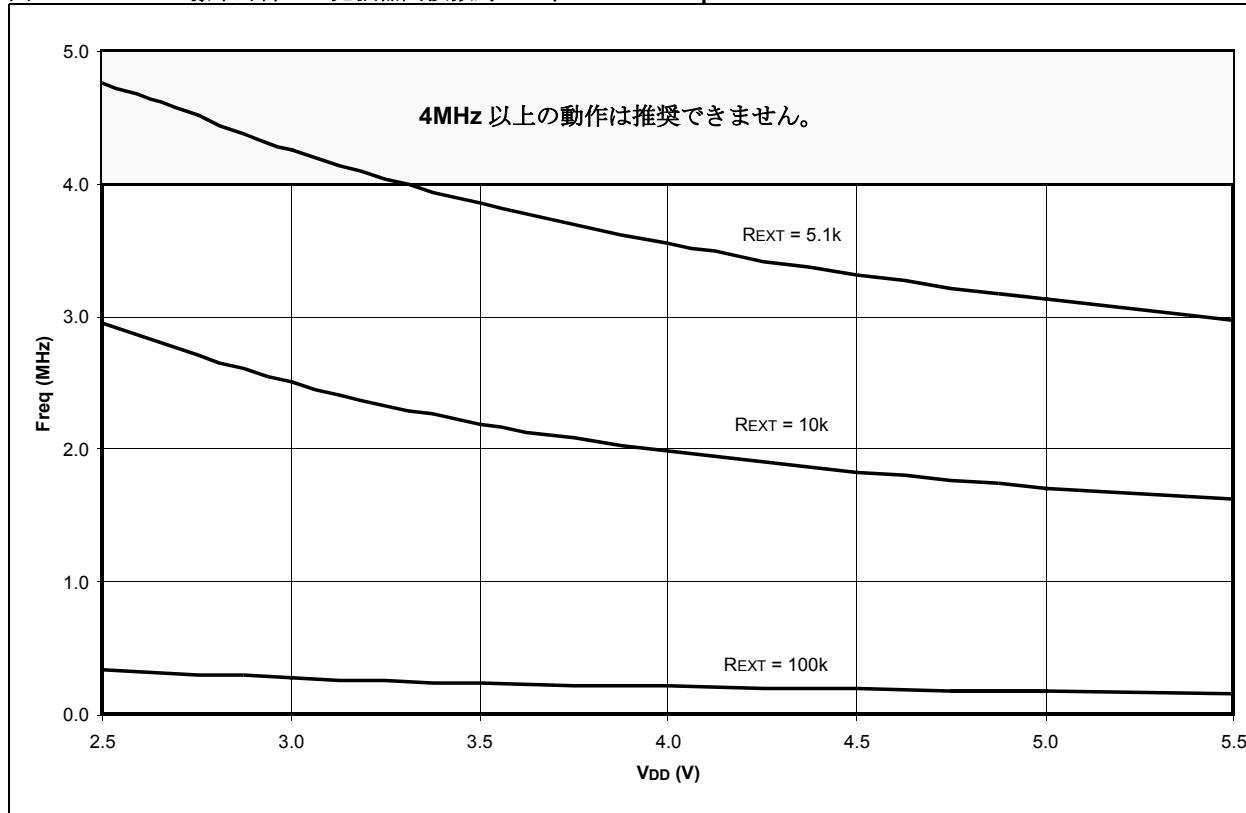
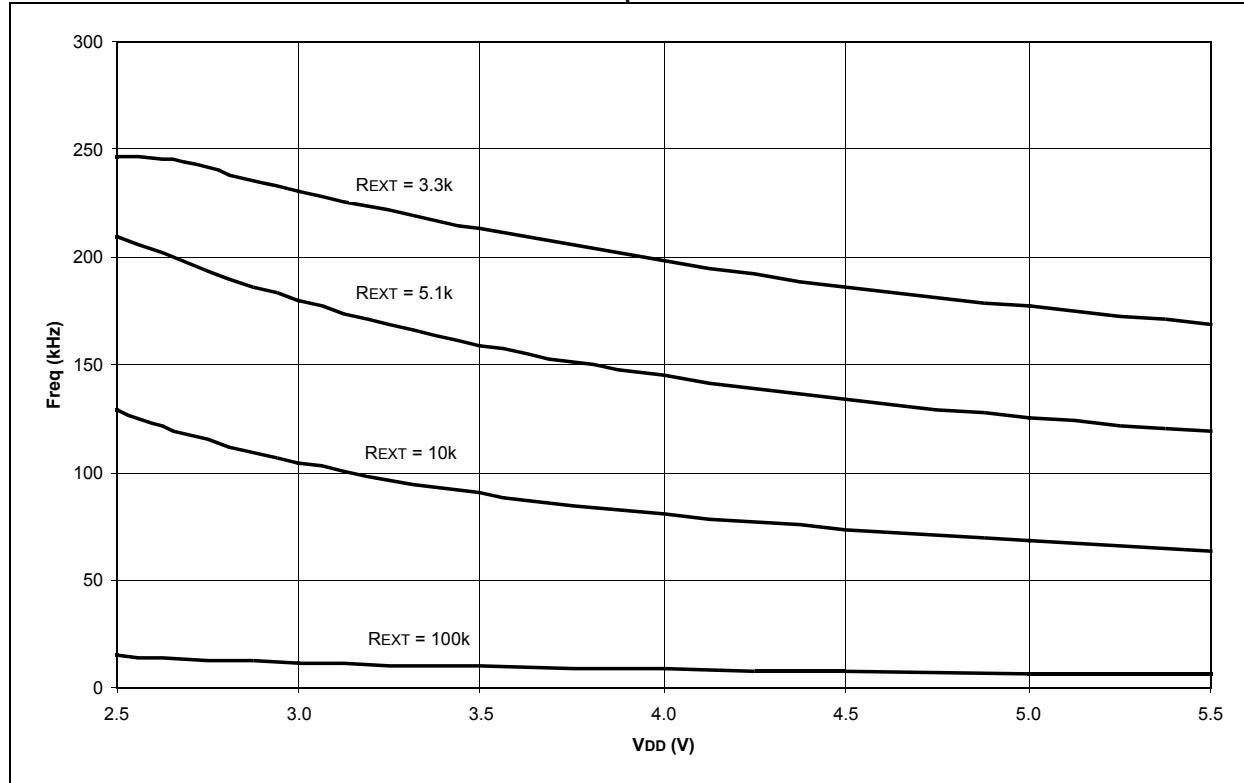
図 7-9: 標準外部 RC 発振器周波数対 VDD、 $C_{EXT} = 20 \text{ pF}$ 図 7-10: 標準外部 RC 発振器周波数対 VDD、 $C_{EXT} = 100 \text{ pF}$ 

図 7-11: 標準外部発振周波数対  $V_{DD}$ 、 $C_{EXT} = 300 \text{ pF}$



## 7.10 フェーズロックループ (PLL)

PLL は FPR<3:0> 発振コンフィギュレーションビットを使用することにより、4 倍、8 倍、もしくは 16 倍の動作モードで使用できます。それぞれの動作モードの入力と出力周波数範囲は表 7-3 に示されています。

**注：** いくつかの PLL では、dsPIC30F デバイスの最大動作周波数を超えた出力周波数範囲を持つものがあります。詳細については特定デバイスのデータシートの「電気的特性」を参照してください。

表 7-3: PLL 周波数範囲

F <sub>IN</sub>	PLL 乗倍器	F <sub>OUT</sub>
4 MHz-10 MHz	x4	16 MHz-40 MHz
4 MHz-10 MHz	x8	32 MHz-80 MHz
4 MHz-7.5 MHz	x16	64 MHz-120 MHz

### 7.10.1 PLL ロックステータス

PLL 回路は、PLL が位相ロック状態に入ったことを検出できます。また、PLL のロックが外れたことも検出できます。PLL がロックするための遅延時間は T<sub>LOCK</sub> で示されます。T<sub>LOCK</sub> 値は通常 20  $\mu$ s です。詳細については特定デバイスのデータシートの「電気的特性」を参照してください。

LOCK ビットは読み出し専用ビット (OSCCON<5>) で、PLL の LOCK 状態を反映します。LOCK ビットはパワーオンリセットでクリアされます。

#### 7.10.1.1 クロック切替え期間中の PLL ロック喪失

(パワーオンリセットも含めて) クロック切替え動作で目標クロックソースとして PLL が選択された場合は、LOCK ビットがクリアされます。LOCK ビットは位相ロックが確立した時にセットされます。PLL のロックができなかった場合は、クロック切替え回路は、システムクロック用出力として PLL には切替えず、その代わり旧クロックソースで動作を継続します。

#### 7.10.1.2 パワーオンリセット期間中の PLL ロック喪失

パワーオンリセット(POR)でPLLのロックができず、フェールセーフクロックモニター(FSCM)が有効の場合、FRC 発振器がデバイスのクロックソースとなり、クロック不良トラップが発生します。

#### 7.10.1.3 通常デバイス動作期間中の PLL ロック喪失

通常動作中に、4 入力クロックサイクル分 PLL のロックが外れた場合は、LOCK ビットはクリアされ、PLL ロック外れを示します。さらに、クロック不良トラップが生成されます。この場合、プロセッサは PLL クロックソースを用いて動作実行を継続します。必要であれば、トラップサービスルーチンでユーザーは別のクロック源に切替え可能です。

**注：** 発振器不良トラップに関する詳細については第6章、「割り込み」を参照ください。

**注：** 通常デバイス動作中に PLL ロックが外れた場合は、クロック不良トラップが生成されますが、システムクロックソースは変化しません。ロック外れを検出するために、FSCM を有効にする必要はありません。

## 7.11 低電力 32 kHz 水晶発振器

LP もしくは副発振器は、32kHz の水晶で低電力動作を行うように特別に設計されています。LP 発振器は SOSCO と SOSCI デバイスピンに存在し、低電力動作のために第 2 水晶クロックソースとして動作します。LP 発振器はリアルタイムクロックのアプリケーション用としてタイマー 1 をドライブします。

### 7.11.1 LP 発振器の有効化

以下のコントロールビットが LP 発振器の動作に影響を与えます。

1. OSCCON レジスタ内の COSC<1:0> ビット (OSCCON<13:12>)。
2. OSCCON レジスタ内の LPOSCEN ビット (OSCCON<1>)。

LP 発振器が有効になると、SOSCO と SOSCI I/O ピンは発振器でコントロールされ、I/O 機能用としては使用できません。

#### 7.11.1.1 LP 発振器の連続動作

LPOSCEN コントロールビット (OSCCON<1>) がセットされると、LP 発振器は常に有効になります。LP 発振器が継続して動作させるようにしている理由が 2 つあります。第 1 に、LP 発振器を常に ON にさせておくことで、低電力動作用の 32kHz システムクロックに高速に切替えることができます。発振器が水晶タイプのソースである場合は、高速メイン発振器へ戻ることは、発振機のスタートアップ時間を必要とします（セクション 7.12「発振器スタートアップタイマ（OST）」参照）。第 2 に、リアルタイムクロックとして、タイマー 1 を使用する場合は発振器は常に ON でなければなりません。

#### 7.11.1.2 LP 発振器の断続的な動作

LPOSCEN コントロールビット (OSCCON<1>) がクリアされると、LP 発振器は、現行デバイスクロックソースとして選択された時 (COSC<1:0> = 00) のみ動作します。LP 発振器が、現行デバイスソースのとき、デバイスがスリープモードに入ったら停止します。

### 7.11.2 タイマー 1 を使った LP 発振器動作

LP 発振器はリアルタイムクロックのアプリケーションにおいて、タイマー 1 用のクロックソースとして使用されます。詳しくは第 12 章、「タイマー」を参照してください。

## 7.12 発振器スタートアップタイマ（OST）

水晶発振子（もしくはセラミック発振子）が発振を開始し安定することを確実にするため、発振器スタートアップタイマーが備えられています。これは単純な 10 ビットのカウンタで、発振器のクロックをシステムのその他に開放する前に 1024Tosc サイクルを計測します。タイムアウト期間は Tost として示されます。発振信号の振幅は、OST がサイクルをカウント開始する前に、発振器のピン用として VIL と VIH 閾値に達する必要があります（図 7-4 参照）。

Tost 時間は、発振器が再スタートする時はいつでも含まれます（すなわち、POR, BOR とスリープモードからの起動）。発振器のスタートアップタイマーは、LP 発振器と主発振器の XT、XTL および HS モードに適用されます。

## 7.13 内蔵高速 RC 発振器（FRC）

FRC 発振器は高速（8MHz 最大）の内蔵 RC 発振器です。この発振器は外部水晶、セラミック発振子もしくは RC 回路を使用せずにそれなりのデバイス動作スピードを供給する目的です。

COSC<1:0> = 01 の時はいつでも、dsPIC30F の動作は、FRC 発振器から始まります。

## 7.14 内蔵低電力 RC 発振器 (LPRC)

LPRC 発振器はウォッチドッグタイマー (WDT) の一デバイスであり、公称 512KHz で発振します。LPRC 発振器はパワーアップタイマー (PWRT) 回路、WDT およびクロックモニター回路用のクロックソースです。これはまた、消費電力が重要でタイミング精度が不要なアプリケーション用の低周波数のクロックソースのオプションとして用いられることもあります。

**注：** LPRC 発振器の発振周波数はデバイス電圧と動作温度により変化します。詳細については特定デバイスのデータシートの「電気的特性」を参照してください。

### 7.14.1 LPRC 発振器を有効にする

LPRC 発振器は、PWRT 用のクロックソースであるため、パワーオンリセット時には常に有効になります。PWRT が無効になった後、以下の条件が当てはまる場合 LPRC 発振器は ON を保ちます。

- フェールセーフクロックモニターが有効な場合
- WDT が有効な場合
- LPRC が、システムクロックとして選択されている場合 ( $COSC<1:0> = 10$ )

上記のどれも当てはまらない場合は、LPRC は PWRT が無効になった後に停止します。

## 7.15 フェールセーフクロックモニター (FSCM)

フェールセーフクロックモニター (FSCM) は、発振器不良が発生しても、デバイスが動作を継続できるようにします。FSCM 機能は、FOSC デバイスコンフィギュレーションレジスタ内の FCKSM (クロック切替えとモニター) ビットをプログラムすることで有効になります。詳しくは第 24 章、「デバイスコンフィギュレーション」を参照してください。FSCM 機能が有効の時は、LPRC 内蔵発振器は（スリープモード時を除き）常に動作します。

発振器不良が発生したら、FSCM は発振器不良トラップを発生させ、システムクロックを FRC 発振器に切替えます。ユーザーは、ここで発振器を再スタートさせるか、コントロールして停止を行うかどうかを行います。

FSCM モジュールは FRC 発振器への切替えが行われた場合には、以下の動作を行います。

1.  $COSC<1:0>$  ビットに ‘01’ を転送します。
2. 発振器不良を表示するために CF ビットがセットされます。
3. 待機中のクロック切替えをキャンセルするために OSWEN コントロールビットがクリアされます。

**注：** 発振器不良に関する詳細については、第 6 章、「割り込み」を参照してください。

### 7.15.1 FSCM 遅延

POR、BOR もしくはスリープモードからの起動時には、FSCM がクロックソースのモニターを開始する前に、通常 100  $\mu$ s の遅延 (TFSCM) が挿入されます。FSCM 遅延の目的は、パワーアップタイマー (PWRT) が未使用の場合に、発振器および/もしくは PLL を安定にするための時間を確保するためです。FSCM 遅延は、内蔵システムリセット信号 SYSRST が解除された後に生成されます。FSCM 遅延タイミングの情報については、第 8 章、「リセット」を参照してください。

FSCM 遅延 TFSCM は、FSCM が有効で、システムクロックとして以下のデバイスクロックソースが選択されている時に有効となります。

- EC+PLL
- XT+PLL
- XT
- HS
- XTL
- LP

**注：** TFSCM の規格値についてはデバイスのデータシートの「電気的特性」を参照してください。

## 7.15.2 FSCM とゆっくりとした発振器のスタートアップ

選択されたデバイスの発振器のスタートアップが、POR、BOR、もしくはスリープモードより遅い時は、発振器がスタートする前に FSCM 遅延を無効にできます。この場合、FSCM はクロック不良トラップを起動します。これが発生すると、COSC<1:0> ビット (OSCCON<13:12>) が FRC 発振器選択となります。これにより、起動させようとしていた元の発振器を効果的に停止できます。ユーザーはこの状況を検出したら、トラップサービスルーチン内でクロック切替えを起動し、希望の発振器に戻します。

## 7.15.3 FSCM と WDT

クロック不良が発生しても、WDT は影響を受けず、LPRC クロックで動作を継続します。

## 7.16 プログラマブル発振器ポストスケーラ

ポストスケーラにより、CPU と周辺に供給されるクロックの周波数を低くすることで、消費電力の低減をはかることができます。ポストスケーラの値は、POST<1:0> コントロールビット (OSCCON<7:6>) によりいつでも変更可能です。

きれいなクロック遷移を確実にするために、クロック変更が発生する前に遅延を入れます。クロックポストスケーラは、64 分周出力の立下りエッジが発生するまで、クロック選択マルチブレクサーを変更しません。実際、POST<1:0> コントロールビットがいつ変更されるかにより、最大 64 サイクルの遅延が発生します。図 7-13 に、3 つの異なるポストスケーラの変化に対しての動作を示します。

図 7-12: プログラマブル発振器ポストスケーラー

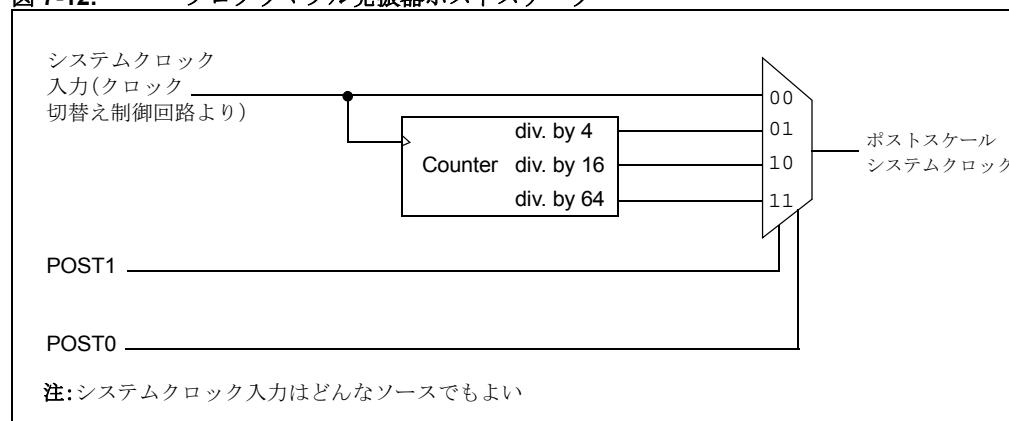
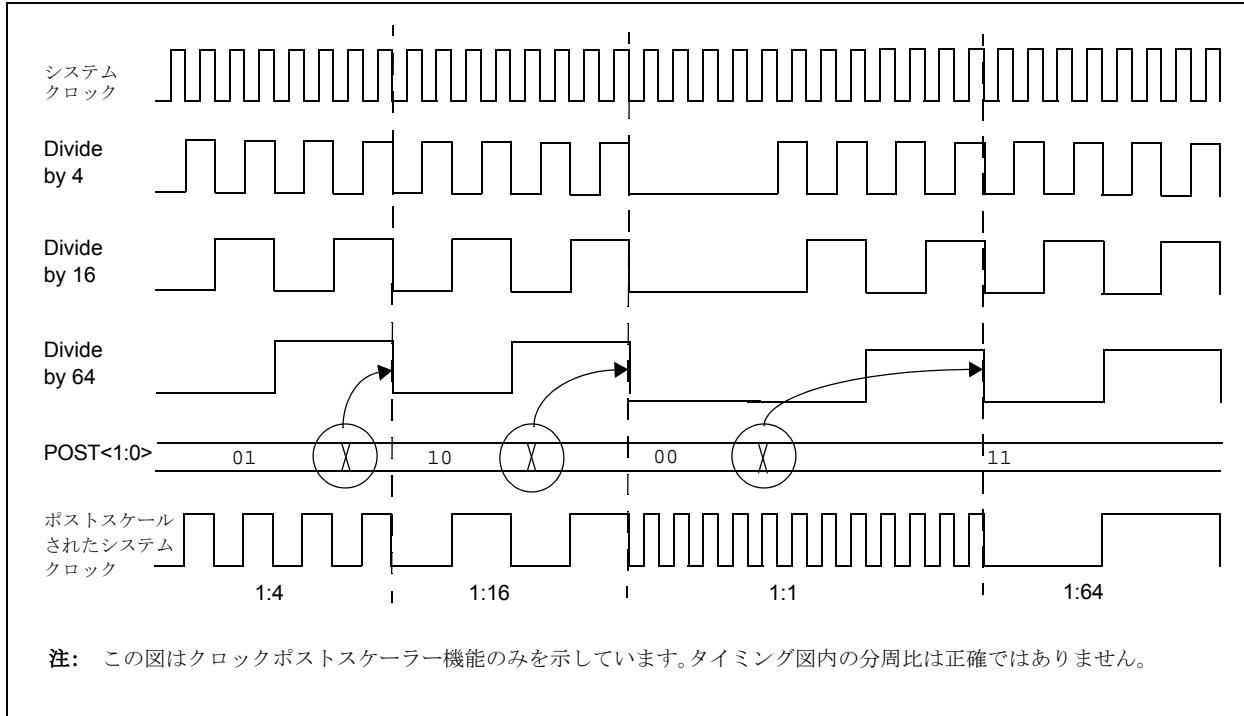


図 7-13: コンフィギュレーションポストスケーラー更新タイミング



## 7.17 クロック切替え動作

デバイス動作中のクロック切替えに使用できるクロックソースは以下から選択できます。

- OSC1/OSC2 ピンの主発振器
- SOSCO/SOSCI ピンの低消費電力 32kHz 水晶発振器（副発振器）
- 内蔵高速 RC (FRC) 発振器
- 内蔵低消費電力 RC (LPRC) 発振器

**注:** 主発振器は複数の動作モード (EC、RC、XT、等) を持っています。主発振器の動作モードは、FOSC デバイスコンフィギュレーションレジスタ内 FPR<3:0> コンフィギュレーションビットにより決定されます（詳細については第 24 章「デバイスコンフィギュレーション」を参照してください）。

### 7.17.1 クロック切替え有効

クロック切替えを有効にするには、FOSC コンフィギュレーションレジスタ内 FCKSM1 コンフィギュレーションビットを ‘0’ にプログラムする必要があります（詳細については第 24 章「デバイスコンフィギュレーション」を参照してください）。

FCKSM1 コンフィギュレーションレジスタビットが ‘1’ (未プログラム時) のときは、クロック切替え機能は無効です。フェールセーフクロックモニター機能も無効です。これがデフォルトの設定です。NOSC<1:0> コントロールビット (OSCCON<9:8>) は、クロック切替えが無効の場合は、クロック選択をコントロールしません。ただし、COSC<1:0> ビット (OSCCON<13:12>) は、Fosc コンフィギュレーションレジスタ内 FPR<3:0> と FOS<1:0> コンフィギュレーションビットにより選択されるクロックソースについて反映します。OSWEN コントロールビット (OSCCON<0>) は、クロック切替えが無効の場合は、影響を与えません。常に ‘0’ を保持します。

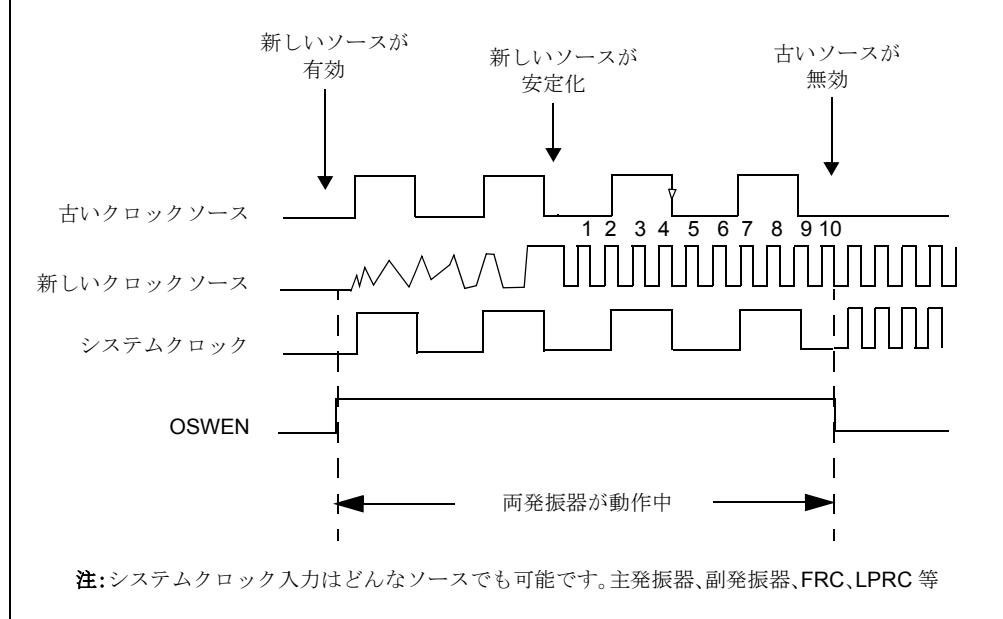
### 7.17.2 発振器切替えシーケンス

デバイスのクロックソースを切替えるには、ハードウェアとソフトウェアにより以下の手順を行います。

1. 現状の発振器ソースを決定するために、必要であれば、COSC<1:0> ステータスピット (OSCCON<13:12>) を読み出します。
2. OSCCON レジスタの上位バイトを書き込むために解除シーケンスを実行します。
3. 新しい発振器ソース用に、NOSC<1:0> コントロールビット (OSCCON<9:8>) に適切な値を書き込みます。
4. OSCCON レジスタの下位バイトを書き込むために解除シーケンスを実行します。
5. OSWEN ビット (OSCCON<0>) をセットします。これにより、発振器切替えを起動します。
6. クロック切替えハードウェアはCOSC<1:0>ステータスピットとNOSC<1:0>コントロールレジスタの新しい値を比較します。それらが同じであればクロック切替えは冗長な動作になります。この場合、OSWEN ビットは自動的にクリアされクロック切替えは中断されます。
7. 有効なクロック切替えが起動されると、LOCK (OSCCON<5>) と CF (OSCCON<3>) ステータスピットがクリアされます。
8. 新しい発振器が動作していないければ、ハードウェアにより開始されます。水晶発振器を発振開始する場合は、ハードウェアは発振器スタートアップタイマ (OST) が完了するまで待ちます。新しいソースが PLL を使用する場合は、ハードウェアは PLL ロックが検出される (LOCK = 1) まで待ちます。
9. ハードウェアは新しいクロックソースからの 10 クロックサイクルを待ち、それからクロックを切替えます。
10. ハードウェアは OSWEN ビットをクリアし、クロック遷移がうまくいったことを表示します。さらに、NOSC<1:0> ビット値が COSC<1:0> ステータスピットに転送されます。
11. クロック切替えが完了します。元のクロックソースはこの時停止しますが、以下の例外があります。
  - WDT もしくは FSCM が有効の時は LPRC 発振器は停止しません。
  - LPOSCEN = 1 (OSCCON<1>) の時は LP 発振器は停止しません。

**注：** クロック切替えシーケンス中でもプロセッサはコードの実行を継続します。この時には、タイミングに敏感なコードは実行すべきではありません。

図 7-14: クロック遷移タイミング図



### 7.17.3 クロック切替え時の秘訣

- 目的とするクロックソースが水晶発振器の場合、クロック切替え時間は発振器スタートアップ時間 (OST) に支配されます。
- 新しいクロックソースが発振開始しないか、存在しないときには、クロック切替えハーデウェアは、おこるべき 10 同期サイクルを単純に待ちます。OSWEN ビット (OSCCON<0>) がセットされたままでありますので、ユーザーはこの状態を検出できます。
- 新しいクロックソースが PLL を使用する場合は、ロックが成立するまでクロック切替えは発生しません。LOCK ビットがクリアされ OSWEN ビットがセットされるので、ユーザーは PLL ロックができなかったことを検出できます。
- ユーザーは、クロック切替え実行中には、POST<1:0> コントロールビット (OSCCON<7:6>) の設定を考慮した方がよいかもしれません。ポストスケーラ比が 1:1 より大きい場合には、LP 発振器のような低周波数のクロックソースへの切替のときには、デバイス動作は大幅に遅くなります。

### 7.17.4 クロック切替えの中断

クロック切替えが終了しない場合、OSWEN ビットをクリアすることによりクロック切替え論理はリセットされます。OSWEN ビット (OSCCON<0>) をクリアすることにより、以下の状態になります。

- クロック切替えが中断します。
- 適用中であれば、OST を停止しリセットします。
- 適用中であれば、PLL を停止します。

クロック切替え手順は、いつでも中断できます。

### 7.17.5 クロック切替え中のスリープモードへの遷移

クロック切替え動作中にデバイスがスリープモードに入る場合、クロック切替え動作は中断されます。プロセッサは元のクロックを選択したままで、OSWEN ビットはクリアされます。通常はこのあと PWRSAV 命令を実行します。

### 7.17.6 クロック切替え用の推奨コードシーケンス

発振ソースを変更するためには、以下の手順を実行する必要があります。

- OSCCON レジスタがロックされていない時や書き込みシーケンス中は、割り込みを無効にします。
- OSCCON の上位バイト用に解除シーケンスを実行します。
- NOSC<1:0> コントロールビットに新しい発振ソースを書き込みます。
- OSCCON の下位バイト用に解除シーケンスを実行します。
- OSWEN ビットをセットします。
- クロックに敏感でないコードの実行を継続します（オプション）。
- 発振器および/もしくは PLL がスタートアップできるように適切な量のソフトウェア遅延（サイクルカウント）を起動します。
- OSWEN が '0' になったか確認します。'0' になったら成功です。
- OSWEN が依然としてセットされていたら、失敗の理由を特定するために LOCK ビットを確認します。

## 7.17.7 クロック切替えコードの例

### 7.17.7.1 クロック切替えの開始

以下のコードは、OSCCONレジスタを解除し、クロック切替え動作を開始させる方法を示しています。

```
;W0に新しい発振器選択を設定  
;OSCCONH(上位バイト)の解除シーケンス  
    MOV    #OSCCONH, w1  
    MOV    #0x78, w2  
    MOV    #0x9A, w3  
    MOV.b w2, [w1]  
    MOV.b w3, [w1]  
  
;新しい発振器選択を設定  
    MOV.b WREG,      OSCCONH  
  
;OSCCONL(下位バイト)の解除シーケンス  
    MOV    #OSCCONL, w1  
    MOV.b #0x01, w0  
    MOV    #0x46, w2  
    MOV    #0x57, w3  
    MOV.b w2, [w1]  
    MOV.b w3, [w1]  
  
;発振器切替え動作の開始  
    MOV.b w0, [w1]
```

### 7.17.7.2 クロック切替えの中断

以下のコードシーケンスは、クロック切替えがうまくいかなかった時に中断するために使用します。

```
MOV.B  #0x46,w0          ; 最初の解除コードを w0 に設定。  
MOV.B  #0x57,w1          ; 次の解除コードを w1 に設定。  
MOV.B  W0,OSCCONL        ; 最初の解除コードの書き込み  
MOV.B  W1,OSCCONL        ; 次の解除コードの書き込み  
BCLR.B OSCCON,#OSWEN     ; クロック切替えの中断
```

## 7.18 設計の秘訣

**質問 1:** 電源投入後、オシロスコープで OSC2 ピンを観測したところ、クロックが表示されていません。どうしてでしょう？

回答：

1. ウェイクアップソース (WDT、MCLR もしくは割り込み) がない状態でスリープモードに入っている場合。ウェイクアップソースを準備しないで、デバイスをスリープにするコードではないことを確認します。可能であれば、MCLR にローパルスを入れて起動をかけてみます。MCLR をローにしたまま電源を投入することにより、水晶発振器にスタートアップ時間を確保できますが、プログラムカウンタは MCLR ピンがハイになるまで進みません。
2. 希望の周波数に対して、間違ったクロックモードが選択されている場合。書き込みがなされていないデバイスの場合、デフォルトの発振器は EC+16 倍 PLL です。ほとんどのデバイスは、デフォルトモードで選択されたクロックであり、水晶発振器や共振子では発振開始しません。クロックモードが正しくプログラムされているか確認してください。
3. 適切な電源投入シーケンスが行われていない場合。電源投入前に I/O ピンを通して CMOS デバイスに電源が供給された場合、不都合が発生します（ラッチアップや不正電源投入等）。BOR 成立状態だったり、スタートアップ時に電源ラインにノイズが乗っていたり、VDD 立上げ時間が遅かったりする場合も問題が発生します。I/O に何も接続せずにデバイスの電源を立上げたり、よく知られた、良質で、立上げ時間の早い電源を使って電源供給を行ってください。BOR や電源立上げシーケンスに関する配慮については、デバイスのデータシートの電源立ち上げ情報を参照してください。
4. 水晶に付けられた C1 と C2 容量が適切に接続されていないか、正しい値ではない場合。すべての接続が正しいことを確認してください。これらのデバイスに対して、デバイスデータシートの値を使えば、通常は水晶が動作しますが、ユーザーのデザインで最適な値でないかもしれません。

**質問 2:** デバイスは動作を開始しましたが、水晶の共振周波数よりもはるかに高い周波数で発振しています。

回答：この発振回路のゲインが高すぎます。C2 (より高くする必要があるかもしれない)、Rs (必要かもしれない) およびクロックモード (間違った選択をしているかもしれない) の選択を確認するため、セクション 7.6 「水晶発振子 / セラミック発振子」を参照してください。これは特に、共通のよく使われる 32.768kHz のように低い周波数の水晶の場合に頻繁に発生します。

**質問 3:** 回路はうまく動作していますが、周波数が少しづれています。これを調整するにはどうすれば良いでしょうか？

回答：C1 の値を変更することで発振周波数を変えられます。直列共振水晶を使用している場合、同じ周波数値の並列共振水晶とは違う周波数で共振します。並列共振水晶を使用していることを確認して下さい。

**質問 4:** ボードはうまく動作していますが、時々急に停止したり時間が欠けたりします。

回答：時間欠けを調査するためのソフトウェアチェック以外に、発振器出力の振幅が発振器入力をトリガーするのに十分高くないかもしれません。C1 と C2 の値を確認し、希望の発振モードに対してデバイスコンフィギュレーションビットが正しいことを確認してください。

**質問 5:** オシロスコープのプローブを発振器のピンに当てたところ、期待したものが観測されません。何故でしょうか？

回答：オシロスコープのプローブが容量を持つことに注意してください。発振回路にプローブを接続することは、発振器の特性を変えることになります。低容量の (アクティブ) プローブの使用を考えてください。

## 7.19 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正して使用可能です。現状、発振モジュールに関連する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 #
PICmicro® マイクロコントローラ発振器デザインガイド	AN588
PICmicro® マイクロコントローラを用いた低電圧デザイン	AN606
水晶発振の基礎と rfPIC™ と PICmicro® デバイス用と水晶発振の基礎と水晶の選択	AN826

**注：** dsPIC30F ファミリーのデバイスに関しての、その他のアプリケーションノートやコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 7.20 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

この版は、dsPIC30F の発振器モジュールに関する技術情報の変更を含んでいます。

注：



## 第8章. リセット

### ハイライト

この章は以下の項目を含んでいます。

8.1	はじめに.....	8-2
8.2	リセット時のクロックソースの選択.....	8-5
8.3	POR : パワーオンリセット .....	8-5
8.4	外部リセット (EXTR) .....	8-7
8.5	ソフトウェアリセット命令 (SWR) .....	8-7
8.6	ウォッチドッグタイマタイムアウト命令 (WDTR) .....	8-7
8.7	ブランハウトリセット (BOR) .....	8-8
8.8	RCON ステータスビットの使用 .....	8-10
8.9	デバイスリセット時間.....	8-11
8.10	デバイススタートアップタイムチャート.....	8-13
8.11	特殊機能レジスタリセットステート.....	8-16
8.12	設計の秘訣.....	8-17
8.13	関連するアプリケーションノート.....	8-18
8.14	改訂履歴.....	8-19

## 8.1 はじめに

リセットモジュールはすべてのリセット要因を結合し、デバイスマスターリセット信号 SYSRST をコントロールします。以下のリストはデバイスリセット要因を示します。

- POR : パワーオンリセット
- EXTR : ピンリセット (MCLR)
- SWR : RESET 命令
- WDTR : ウオッチドッグタイマーリセット
- BOR : ブラウンアウトリセット
- TRAPR : トランプ衝突リセット
- IOPR : 不當命令コードリセット
- UWR : 未初期化 W レジスタリセット

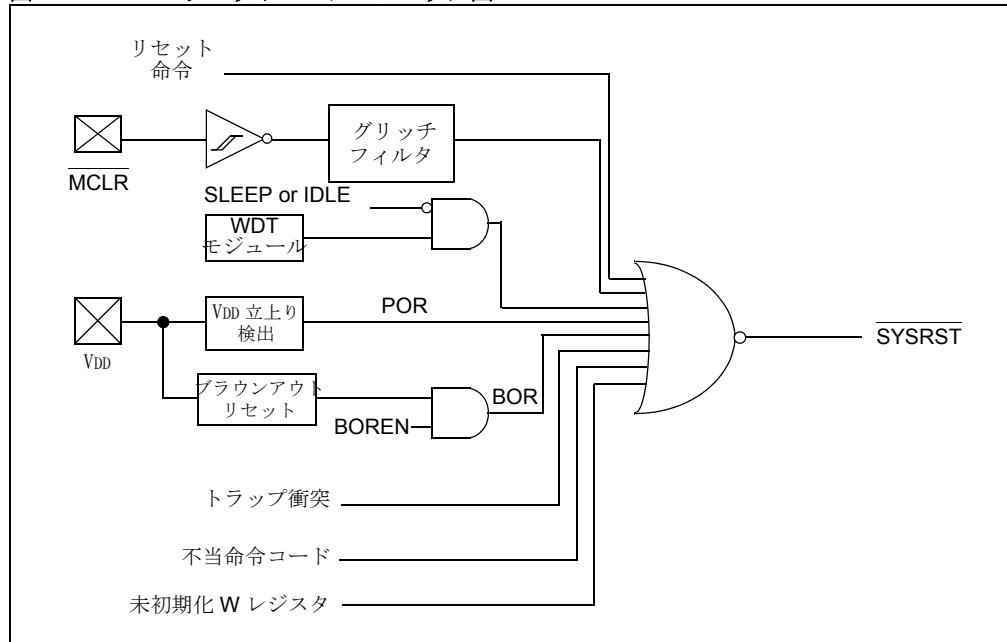
図 8-1 にリセットモジュールの簡略ブロック図を示します。リセットを発生するすべての要因が SYSRST 信号を活性化します。CPU と周辺に関連する多くのレジスタは強制的に「リセット状態」になります。ほとんどのレジスタはリセットにより影響を受けず、その状態は POR 時には不定になり、すべての他のリセットによっても変化しません。

**注:** レジスタのリセット状態についてはこのマニュアルの特定の周辺機器もしくは CPU に関する章を参照してください。

すべての種類のデバイスリセットは、リセットの種類を表示するために RCON レジスタ内の対応するステータスピットをセットします (レジスタ 8-1 をご覧下さい)。POR は、POR と BOR ビット (RCON<2:1>) 以外のすべてのビットをクリアし、POR と BOR ビット (RCON<2:1>) をセットします。ユーザーはコード実行中いつでもすべてのビットをセットしたりクリアしたりできます。RCON ビットはステータスピットとしてのみ機能します。ソフトウェアで特定リセットビットをセットしても、デバイスリセットを発生させることはありません。

RCON レジスタはまた、低電圧検出モジュール、ウォッチドッグタイマーおよびデバイス省電力ステータスに関連したビットを持っています。これらのビットの機能は本マニュアルの別の章で説明します。

図 8-1: リセットシステムブロック図



## レジスタ 8-1: 不當命令コード RCON: リセットコントロールレジスタ

上位バイト:									
R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1		
TRAPR	IOPUWR	BGST	LVDEN	LVDL<3:0>					
ビット 15									

下位バイト:									
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1		
EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR		
ビット 7									

- ビット 15 **TRAPR:** トランプリセットフラグビット  
1 = トランプリセットが発生した。  
0 = トランプリセットは発生していない。
- ビット 14 **IOPUWR:** 不當命令コードもしくは未初期化 W アクセスリセットフラグビット  
1 = 不當命令コード検出、不当アドレスモードもしくはアドレスポインタとして未初期化 W レジスタが使用されるとリセットを発生する。  
0 = 不當命令コードもしくは未初期化 W リセットは発生していない。
- ビット 13 **BGST:** バンドギャップ安定ビット  
1 = バンドギャップは安定している。  
0 = バンドギャップは不安定で、LVD 割込みを無効にする必要がある。
- ビット 12 **LVDEN:** 低電圧検出電力有効ビット  
1 = LVD を有効にし、LVD 回路の電源を立ち上げます。  
0 = LVD を無効にし、LVD 回路の電源を切れます。
- ビット 11-8 **LVDL<3:0>:** 低電圧検出制限ビット。  
詳しくは第9章.「低電圧検出 (LVD)」を参照してください。
- ビット 7 **EXTR:** 外部 RESET ピン (MCLR) ビット  
1 = マスタークリア (ピン) リセットが発生した。  
0 = マスタークリア (ピン) リセットは発生していない。
- ビット 6 **SWR:** ソフトウェア RESET (命令) フラグビット  
1 = RESET 命令が実行された。  
0 = RESET 命令は実行されていない。
- ビット 5 **SWDTEN:** WDT のソフトウェアによる有効/無効ビット  
1 = WDT をオンにする。  
0 = WDT をオフにする。  
注: FWDTEN コンフィギュレーションビットが '1' (未プログラム) の場合、SWDTEN ビットの設定にかかわらず、WDT は常に有効です。
- ビット 4 **WDTO:** ウオッチドッグタイマータイムアウトフラグビット  
1 = WDT タイムアウトが発生した。  
0 = WDT タイムアウトは発生していない。
- ビット 3 **SLEEP:** スリープからの起動フラグビット  
1 = デバイスはスリープモードだった。  
0 = デバイスはスリープモードではなかった。
- ビット 2 **IDLE:** アイドルからの起動フラグビット  
1 = デバイスはアイドルモードだった。  
0 = デバイスはアイドルモードではなかった。

## レジスタ 8-1: RCON: リセットコントロールレジスタ (続き)

### ビット 1 **BOR:** ブラウンアウトリセットフラグビット

1 = ブラウンアウトリセットが発生した。BOR はパワーオンリセット後にもセットされることに注意してください。

0 = ブラウンアウトリセットは発生していない。

### ビット 0 **POR:** パワーオンリセットフラグビット

1 = パワーオンリセットが発生した。

0 = パワーオンリセットは発生していない。

**注:** すべてのリセットステータスビットは、ソフトウェアでセットもしくはクリアされます。これらのビットの 1 つをソフトウェアでセットしてもデバイスリセットは発生しません。

凡例：

R = 読出し可能ビット

W = 書込み可能ビット U = 未定ビット、'0' が読み出されます

-n = POR の値

'1' = ビットがセット '0' = ビットはクリア x = ビットは不定  
されます されます

## 8.2 リセット時のクロックソースの選択

クロック切替えが有効な場合、デバイスリセット時のシステムクロックソースは、表 8-1 に示されるように選択されます。クロック切替えが無効の場合は、システムクロックソースは常に発振器コンフィギュレーションヒューズに従って選択されます。詳しくは第7章.「発振器」を参照してください。

表 8-1: 発振器の選択対リセットのタイプ (クロック切替えが有効の場合)

リセットタイプ	ベースとなるクロックソースの選択
POR	発振器コンフィギュレーションヒューズ
BOR	発振器コンフィギュレーションヒューズ
EXTR	COSC コントロールビット (OSCCON<13:12>)
WDTR	COSC コントロールビット (OSCCON<13:12>)
SWR	COSC コントロールビット (OSCCON<13:12>)

## 8.3 POR : パワーオンリセット

パワーオンリセット (POR) に関連して 2 つの閾値電圧があります。最初の電圧はデバイス閾値電圧  $V_{POR}$  です。デバイス閾値電圧はデバイスの論理回路が動作を開始する電圧です。POR イベントに関連する 2 番目の電圧は、通常は 1.85V の POR 回路閾値電圧です。

$V_{DD}$  の上昇が検出されるとパワーオンイベントは内部パワーオンリセットパルスを生成します。リセットパルスは  $V_{POR}$  で生成されます。デバイスの供給電源特性は POR パルスを生成するために、開始電圧と立ち上がり速度の要求仕様を満たす必要があります。 $V_{POR}$  と  $V_{DD}$  の立ち上がり速度の仕様に関する詳細は、デバイスデータシートの「電気的特性」を参照してください。

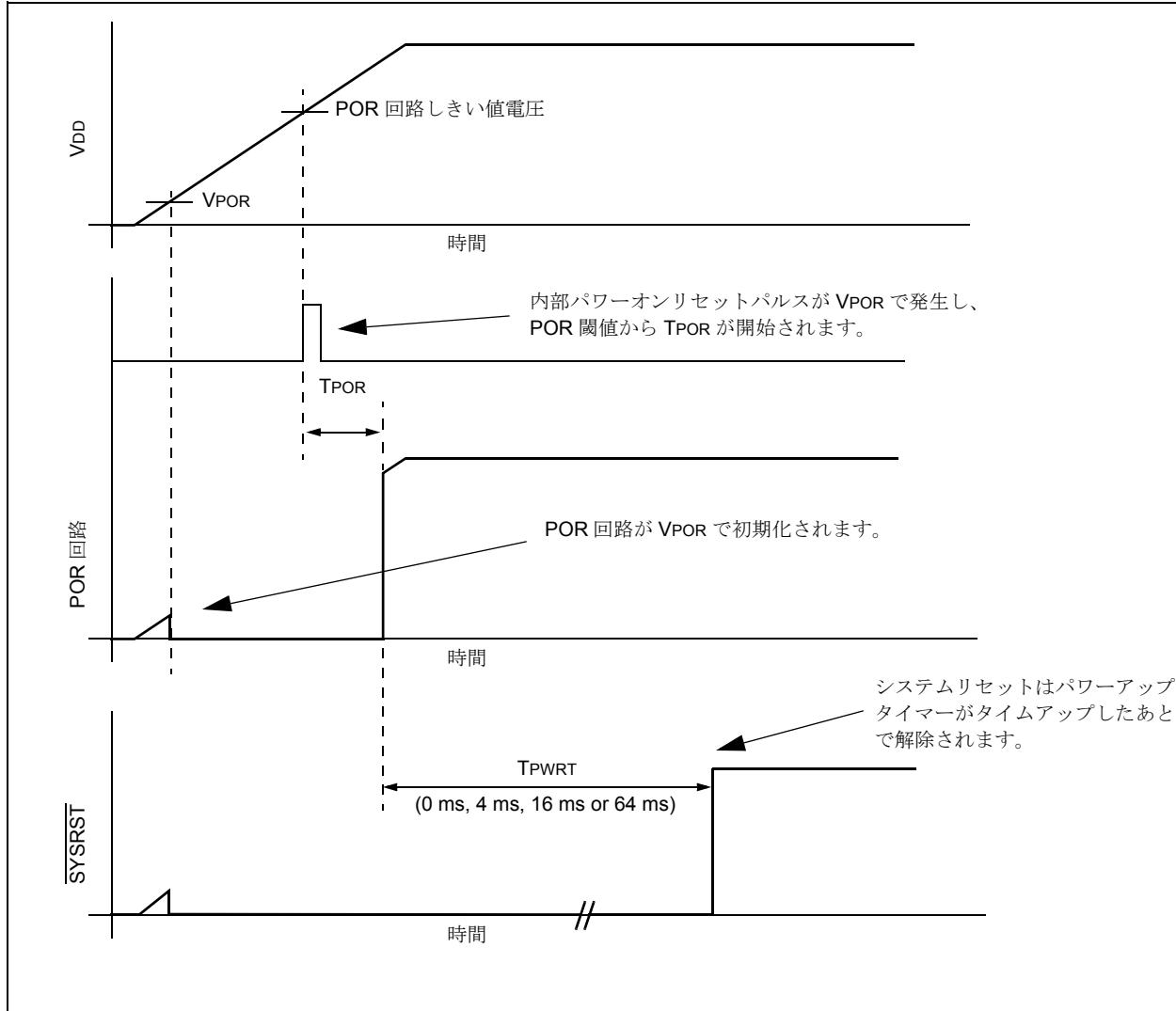
POR パルスは POR タイマーをリセットし、デバイスをリセット状態にします。POR はまた発振器コンフィギュレーションビットにより、デバイスクロックソースを決定します

パワーオンリセットパルスが生成された後、POR 回路は小さな遅延  $TPOR$  (通常は 10 $\mu$ s) を挿入し、内部デバイスのバイアス回路を安定化させます。さらに、ユーザーが選択したパワーアップタイムアウト時間 ( $TPWRT$ ) が適用されます。 $TPWRT$  パラメーターはデバイスコンフィギュレーションビットで指定された、0 ms (遅延なし)、4 ms、16 ms もしくは 64 ms の値をとります。デバイスパワーオンリセットにおける総遅延時間は  $TPOR + TPWRT$  です。これらの遅延が過ぎると命令サイクルクロックの次の最初のエッジで  $SYSRST$  が解除され、PC がリセットペクトルにジャンプします。

$SYSRST$  信号のタイミングを図 8-2 に示します。 $V_{DD}$  がしきい値電圧  $V_T$  より下がるとパワーオンリセットが初期化されます。POR 遅延時間は、 $V_{DD}$  が POR 回路しきい値電圧を過ぎると挿入されます。最後に PWRT 遅延時間  $TPWRT$  は、 $SYSRST$  が解除される前に挿入されます。

パワーオンイベントは POR と BOR ステータスビット (RCON<1:0>) をセットします。

図 8-2: V<sub>DD</sub> 立ち上がり時の POR モジュールタイミング図



**注:** デバイスがリセット状態を抜けたとき（通常動作に入る）には、デバイス動作パラメータ（電圧、周波数、温度等々）が動作範囲内になっているようにしなければなりません。そうでなければデバイスは正常には動作しません。ユーザーは電源が最初に供給された時間と SYSRST が無効になった時間の間の遅延時間を、すべての動作パラメータが仕様内になるように十分長く取る必要があります。

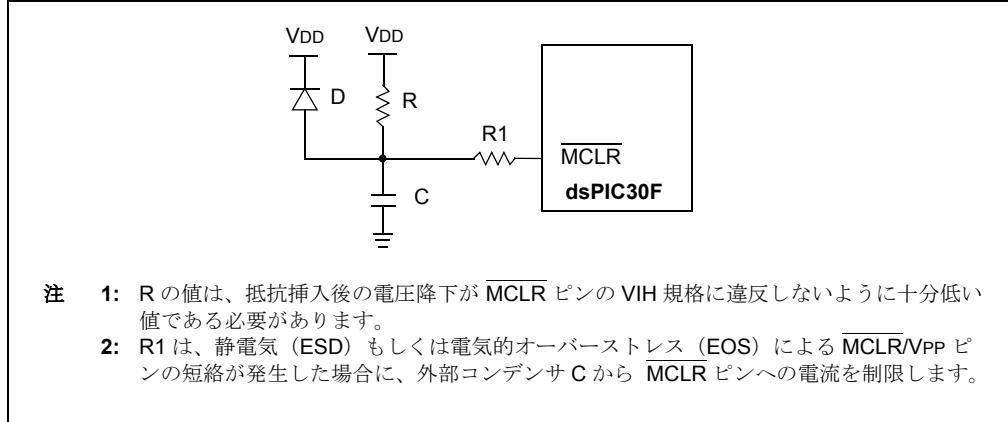
### 8.3.1 POR 回路を使用する

POR 回路を有効に利用するために、MCLR ピンを直接 VDD に接続します。これによりパワー オンリセット遅延を生成するために通常必要な外部 RC 部品をなくすことができます。この場合には VDD の最小立ち上がり時間が必要になります。詳しくは特定デバイスのデータシートの「電気的特性」の章を参照してください。

アプリケーションによっては MCLR ピンと VDD の間に抵抗が必要になるかもしれません。この抵抗は MCLR ピンをノイズの多い電源供給ラインから切り離すために用いられます。またデバイスがアプリケーション回路内に実装されている状態で、デバイスプログラミング電圧 VPP を MCLR ピンに接続する必要がある場合は抵抗が必要になります。VPP はほとんどのデバイスでは 13 ボルトです。

図 8-3 に電源立ち上がりが遅い場合に考えられる POR 回路を示します。デバイス VDD が有効な動作領域に入る前にデバイスがリセット状態を抜ける場合にのみ、外部パワーオンリセット回路が必要になります。ダイオード D は VDD パワーダウン時に、コンデンサ C を急速に放電するのに役に立ちます。

図 8-3: 外部パワーオンリセット回路 (VDD 立上がり時間が遅い場合)



### 8.3.2 パワーアップタイマー (PWRT)

PWRT はデバイス POR もしくは BOR (ブラウンアウトリセット) で SYSRST が解除される前にオプションの遅延時間 (TPWRT) を供給します。PWRT 時間遅延は POR 遅延時間 (TPOR) に追加されて供給されます。PWRT 時間遅延は通常 0 ms、4 ms、16 ms もしくは 64 ms です。(図 8-2 を参照してください。)

PWRT 遅延時間は、FBORPOR デバイスコンフィギュレーションレジスタ内の FPWRT<1:0> コンフィギュレーションヒューズを用いて選択されます。詳しくは第 24 章、「デバイスコンフィギュレーション」を参照してください。

### 8.4 外部リセット (EXTR)

MCLR ピンがローにドライブされている時はいつでも、MCLR の入力パルスがある最小幅より長い場合、デバイスは非同期的に SYSRST を発生します。(詳細については特定デバイスのデータシートの「電気的特性」を参照してください。)MCLR ピンが解除されると、SYSRST は次の命令サイクルで解除され、リセットベクトルの読み込みが始まります。プロセッサは EXTR が発生する前に使用されていた既存クロックソースを保持します。EXTR ステータスビット (RCON<7>) が、MCLR リセットを表示するためにセットされます。

### 8.5 ソフトウェアリセット命令 (SWR)

RESET 命令が実行される時はいつでも、デバイスは SYSRST を発生し、デバイスを特別なリセット状態に置きます。このリセット状態はクロックの再初期化を行いません。RESET 命令前の実行クロックソースは保持されます。SYSRST は次の命令サイクルで解除され、リセットベクトル読み込みが始まります。

### 8.6 ウオッチドッグタイマタイムアウト命令 (WDTR)

ウォッチドッグタイマタイムアウトが発生した時はいつでも、デバイスは非同期的に SYSRST を発生します。クロックソースは変化しません。スリープモードもしくはアイドルモード中の WDT タイムアウトはプロセッサをウェイクアップしますが、プロセッサのリセットは行わない点に注意してください。詳しくは第 10 章、「ウォッチドッグタイマーおよび省電力モード」を参照してください。

## 8.7 ブラウンアウトリセット (BOR)

BOR (ブラウンアウトリセット) モジュールは内蔵基準電圧回路をベースにしています。BOR モジュールの主目的は、ブラウンアウト状態が発生した時にデバイスリセットを生成することです。ブラウンアウト状態は、一般的に AC メイン上のグリッチ (すなわち、電源伝送線の不具合により AC サイクルの波形の一部がなくなること) もしくは高負荷がかかった時に、過電流による電圧低下により引き起こされます。

BOR モジュールは、以下の電圧トリップポイントの中の 1 つを選択します。

- $V_{BOR} = 2.0V$
- $V_{BOR} = 2.7V$
- $V_{BOR} = 4.2V$
- $V_{BOR} = 4.5V$

**注:** ここで示される BOR 電圧トリップポイントは、デザインガイド目的のみで示される標準値です。BOR 電圧の制限仕様については特定のデバイスデータシートの「電気的仕様」を参照してください。

BOR では、デバイスはデバイスコンフィギュレーションビットの値 ( $FPR<3:0>$ ,  $FOS<1:0>$ ) に基づいてシステムクロックソースを選択します。PWRT タイムアウト (TPWRT) は、有効であれば SYSRST が解除される前に適用されます。

水晶発振ソースが選択されている場合は、ブラウンアウトリセットは発振器スタートアップタイマー (OST) を起動します。システムクロックは OST がタイムアップするまで保持されます。システムクロックソースが PLL でドライブされている場合は、クロックは LOCK ビット ( $OSCCON<5>$ ) がセットされるまで保持されます。

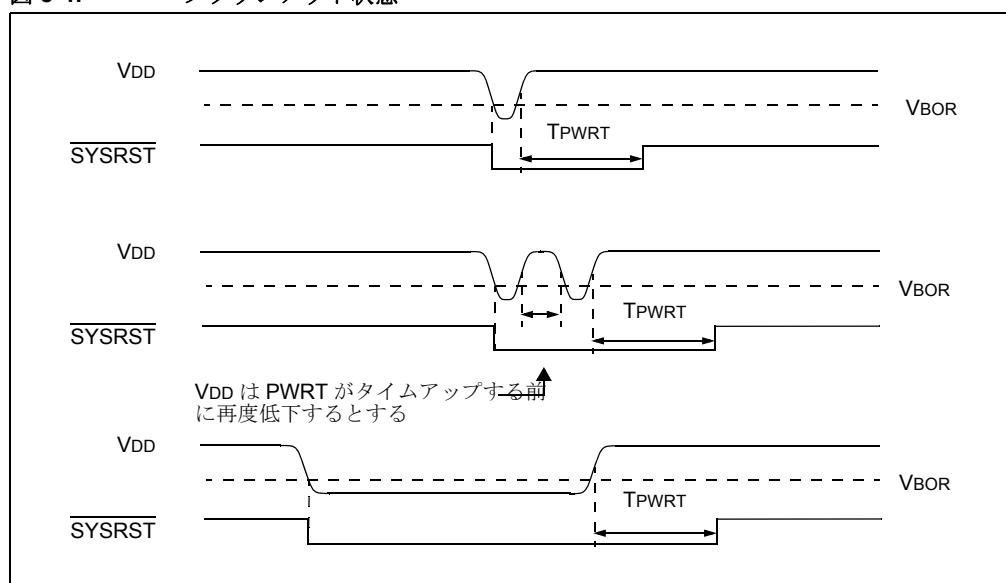
BOR ステータスピット (RCON<1>) が、BOR が発生したことを示すためにセットされます。

BOR 回路は有効であればスリープモードもしくはアイドルモード中でも動作を継続し、 $VDD$  が BOR 閾値電圧以下になった場合にデバイスをリセットします。

BOR の電気的仕様については適切なデバイスのデータシートの「電気的仕様」の章を参照してください。

典型的なブラウンアウトのシナリオを図 8-4 に示します。図に示されるように、 $VDD$  が  $V_{BOR}$  トリップポイント以上になるときは、いつでも PWRT 遅延(もし有効であれば)が起動されます。

図 8-4: ブラウンアウト状態



### 8.7.1 BOR コンフィギュレーション

BOR モジュールは、デバイスコンフィギュレーションヒューズ経由ビットで有効 / 無効にしたり構成を変えたりできます。

BOR モジュールはデフォルトでは有効であり（消費電力を低減するために）、BOREN デバイスコンフィギュレーションヒューズビットを ‘0’ (FBORPOR<7>) にプログラムすることで無効にできます。BOREN コンフィギュレーションヒューズビットは FBORPOR デバイスコンフィギュレーションレジスタ内に配置されています。BOR 電圧トリップポイント (VBOR) は、BORV<1:0> コンフィギュレーションヒューズビット (FBOR<5:4>) を用いることで選択されます。詳しくは第 24 章、「デバイスコンフィギュレーション」を参照してください。

### 8.7.2 BOR 動作の電流消費

BOR 回路は、低電圧検出モジュールのようなその他の周辺デバイスと共有する内蔵基準電圧回路に依存しています。内蔵基準電圧は、関連する周辺回路の 1 つが有効である時はいつでも動作しています。このため BOR を無効にしたときに期待される電流消費の変化は観測されません。

### 8.7.3 不当命令コードリセット

デバイスリセットは、プログラムメモリから取り込まれた不当な命令コードをデバイスが実行しようとした時に発生します。不当命令コードリセット機能は、デバイスが定数データを格納するために使用されるプログラムメモリセクションを実行することを防止します。不当命令コードリセットを効果的に使うためには、データ値を格納するときに、プログラムメモリの下位 16 ビットのみを使用します。そして上位 8 ビットには、不当命令コード値である 0x3F をプログラムします。

不当命令コード値の結果デバイスリセットが発生したら、IOPUWR ステータスピット (RCON<14>) がセットされます。

### 8.7.4 未初期化 W レジスタリセット

W レジスタアレイ (W15 を除く) はすべてのリセットでクリアされ、何か書き込まれるまで未初期化とみなされます。未初期化のレジスタをアドレスポインタとして使おうとすると、デバイスをリセットします。さらに IOPUWR ステータスピット (RCON<14>) がセットされます。

### 8.7.5 トランプ衝突リセット

同時に複数のハードウェアトランプソースが発生した時には、いつでもデバイスリセットが発生します。TRAPR ステータスピット (RCON<15>) がセットされます。トランプ衝突リセットに関する詳細については第 6 章、「割り込み」を参照してください。

## 8.8 RCON ステータスピットの使用

ユーザーはリセットの理由を特定するために、どんなデバイスリセットが発生した後でも RCON レジスタを読み出すことができます。

**注：** RCON レジスタ内のステータスピットは、デバイスリセット後、次の RCON レジスタ値が意味あるようにするため、読み出された後にはクリアする必要があります。

表 8-2 にリセットフラグビット動作のサマリーを示します。

表 8-2: リセットフラグビット動作

フラグビット	設定要因	クリア要因
<b>TRAPR</b> (RCON<15>)	トラップ衝突イベント	POR
<b>IOPWR</b> (RCON<14>)	不当命令コードもしくは未初期化 W レジスタのアクセス	POR
<b>EXTR</b> (RCON<7>)	MCLR リセット	POR
<b>SWR</b> (RCON<6>)	RESET 命令	POR
<b>WDTO</b> (RCON<4>)	WDT タイムアウト	PWRSAV 命令、POR
<b>SLEEP</b> (RCON<3>)	PWRSAV #SLEEP 命令	POR
<b>IDLE</b> (RCON<2>)	PWRSAV #IDLE 命令	POR
<b>BOR</b> (RCON<1>)	POR, BOR	
<b>POR</b> (RCON<0>)	POR	

**注：** すべてのリセットフラグビットはユーザーソフトウェアでセットもしくはクリアできます。

## 8.9 デバイスリセット時間

種々のデバイスリセットのリセット時間を表 8-3 にまとめます。システムリセット信号 **SYSRST** は、POR 遅延時間と PWRT 遅延時間がタイムアップした後で解除されます。

デバイスが実際にコードの実行をはじめる時間は、システム発振器の遅延(発振器スタートアップ時間 (**OST**) と PLL ロック時間も含みます) にも依存します。**OST** と PLL ロック時間は、適用される **SYSRST** 遅延時間と並行して発生します。

**FSCM** 遅延は **SYSRST** 信号が解除された後で、**FSCM** がシステムクロックソースのモニターを開始する時間を決定します。

表 8-3: 種々のデバイスリセットに対するリセット遅延時間

リセットタイプ	クロックソース	<b>SYSRST</b> 遅延	システムクロック遅延	<b>FSCM</b> 遅延	注
POR	EC, EXTRC, FRC, LPRC	TPOR + TPWRT	—	—	1, 2
	EC + PLL	TPOR + TPWRT	TLOCK	TFSCM	1, 2, 4, 5
	XT, HS, XTL, LP	TPOR + TPWRT	TOST	TFSCM	1, 2, 3, 5
	XT + PLL	TPOR + TPWRT	TOST + TLOCK	TFSCM	1, 2, 3, 4, 5
BOR	EC, EXTRC, FRC, LPRC	TPWRT	—	—	2
	EC + PLL	TPWRT	TLOCK	TFSCM	1, 2, 4, 5
	XT, HS, XTL, LP	TPWRT	TOST	TFSCM	1, 2, 3, 5
	XT + PLL	TPWRT	TOST + TLOCK	TFSCM	1, 2, 3, 4, 5
MCLR	Any Clock	—	—	—	
WDT	Any Clock	—	—	—	
ソフトウェア	Any clock	—	—	—	
不当命令コード	Any Clock	—	—	—	
未初期化 W	Any Clock	—	—	—	
トラップ衝突	Any Clock	—	—	—	

注 1: TPOR = パワーオンリセット遅延 (通常 10μs)

2: TPWRT = FPWRT<1:0> コンフィギュレーションビットで決定される追加の ‘パワーアップ’ 遅延。この遅延は通常 0 ms、4 ms、16 ms もしくは 64 ms です。

3: TOST = 発振器スタートアップタイマー。発振器のクロックをシステムに供給する前に 10 ビットカウンタが 1024 発振器周期を計測します。

4: TLOCK = PLL ロック時間 (通常 20μs)

5: TFSCM = フェールセーフクロックモニター遅延 (通常 100μs)

## 8.9.1 POR と長い発振器スタートアップ時間

発振器スタートアップ回路とそれに関連する遅延タイマーは、パワーアップ時に発生するデバイスリセット遅延とは関連しません。いくつかの水晶回路（特に低周波数水晶）は比較的長いスタートアップ時間を持ちます。従って **SYSRST** 解除後に以下の 1 つもしくはそれ以上の状況が考えられます。

- 発振器回路が発振を開始しない。
- (水晶発振器が使用されている場合) 発振スタートアップタイマーがタイムアップしない。
- (PLL が使用されている場合) PLL がロックを確立しない。

デバイスは有効なクロックソースがシステムに供給されるまでコードの実行を開始しません。従ってリセット遅延時間を見る必要がある時には、発振器や PLL スタートアップ遅延も考慮する必要があります。

## 8.9.2 フェール・セーフクロックモニター (FSCM) とデバイスリセット

FSCM が有効な場合、**SYSRST** が解除された時にシステムクロックソースのモニターを開始します。この時有効なクロックソースがない場合、デバイスは自動的に **FRC** 発振器に切り替わり、ユーザーはトラップサービスルーチンで所定の水晶発振器に切り替えることができます。

### 8.9.2.1 水晶と PLL クロックソースに対する FSCM 遅延

システムクロックソースが水晶発振器および / もしくは PLL で供給される時、POR と PWRT 遅延時間の後で、小さな遅延 **T<sub>FSCM</sub>** が自動的に挿入されます。FSCM はこの遅延がタイムアップしないとシステムクロックソースのモニターを開始しません。FSCM 遅延時間は通常 100 $\mu$ s で、発振器および / もしくは PLL を安定化するために追加時間を与えます。ほとんどの場合、FSCM 遅延により PWRT が無効の場合に、デバイスリセット時点で発振器不良トラップが発生することを防ぎます。

### 8.10 デバイススタートアップタイムチャート

図8-5から図8-8は、いくつかの動作シナリオにおけるデバイスリセットに関連した遅延をグラフで示しています。

図8-5はシステムクロックとして水晶発振器とPLLが使用され、PWRTが無効の場合の遅延タイムチャートを示します。内部パワーオンリセットパルスがVPORしきい値で発生します。内部リセットパルスの後で小さなPOR遅延が発生します。(POR遅延は、デバイス動作が開始される前に必ず挿入されます。)

FSCMはもし有効であれば、FSCM遅延がなくなった時にシステムクロックのモニターを開始します。図8-5は、フェールセーフクロックモニター(FSCM)が有効になる前に、発振器とPLLの遅延がなくなる場合を示します。ただし、これらの遅延がFSCMが有効になるまでならないこともあります。この場合は、FSCMはクロック不良を検出し、クロック不良トラップを発生します。FSCM遅延が発振器とPLLが安定するための十分な時間でない場合は、PWRTを使うことで、デバイス動作が開始したり、FSCMがシステムクロックのモニターを開始する前にさらなる遅延時間を与えることができます。

図8-5: デバイスリセット遅延、水晶+PLLクロックソース、PWRT無効時

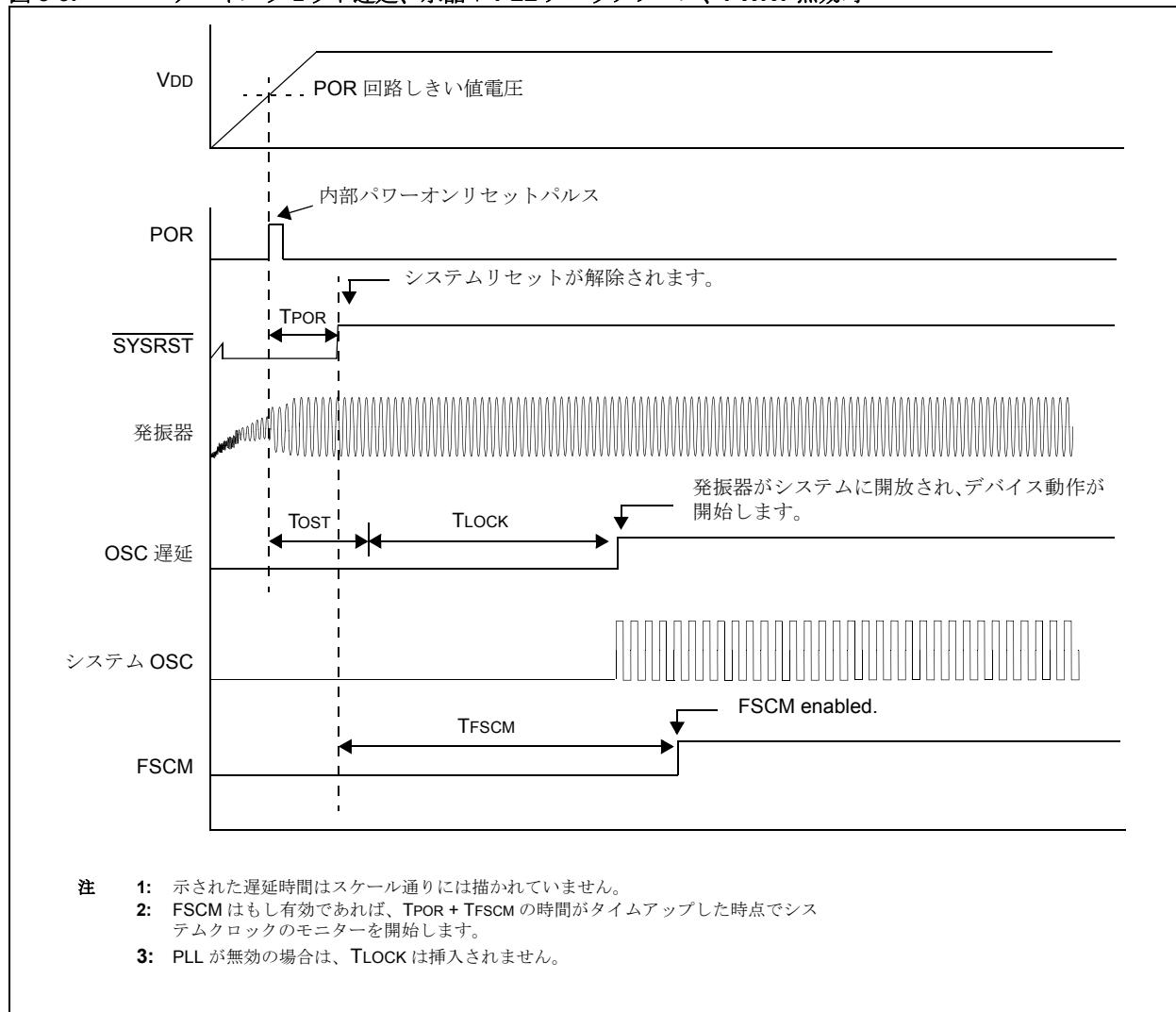
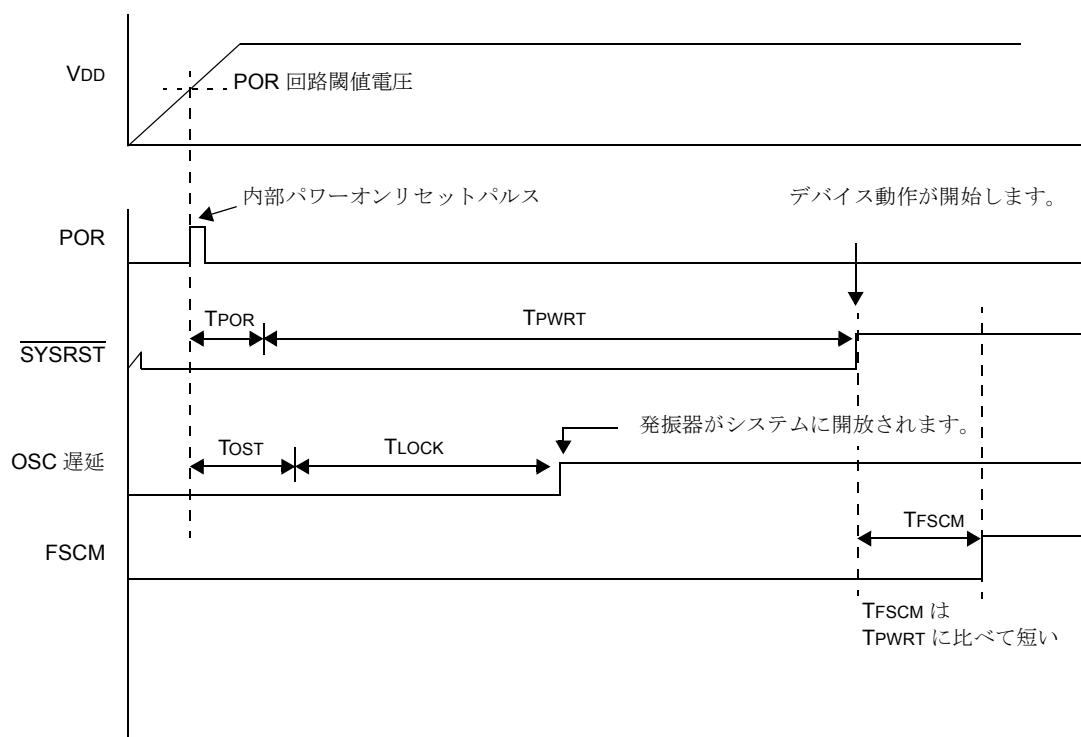


図 8-6 に示されるリセットタイムチャートは、SYSRST が解除される前に、遅延時間の総量を増加させるために PWRT を有効にした点を除けば、図 8-5 に示されるリセット時間と同様です。FSCM はもし有効であれば、TFSCM がタイムアップした後にシステムクロックのモニターを開始します。ほとんどの場合 TFSCM に追加された PWRT 遅延時間で、システムクロックソースを安定化するために十分な時間となります。

図 8-6: デバイスリセット遅延、水晶 + PLL クロックソース、PWRT 有効時



- 注**
- 1: 示された遅延時間はスケール通りには描かれていません。
  - 2: FSCM はもし有効であれば、TPOR + TPWRT + TFSCM の時間がタイムアップした時点でのシステムクロックのモニターを開始します
  - 3: PLL が無効の場合は、TLOCK は挿入されません。

図8-7のリセットタイムチャートはシステムクロックとしてEC+PLLクロックソースが使用され、PWRTが有効な場合の例を示します。この例は、発振スタートアップ時間遅延TOSTが発生しない点を除けば図8-6で示されるものと同様です。

図8-7: デバイスリセット遅延、EC+PLLクロックソース、PWRTが有効の場合

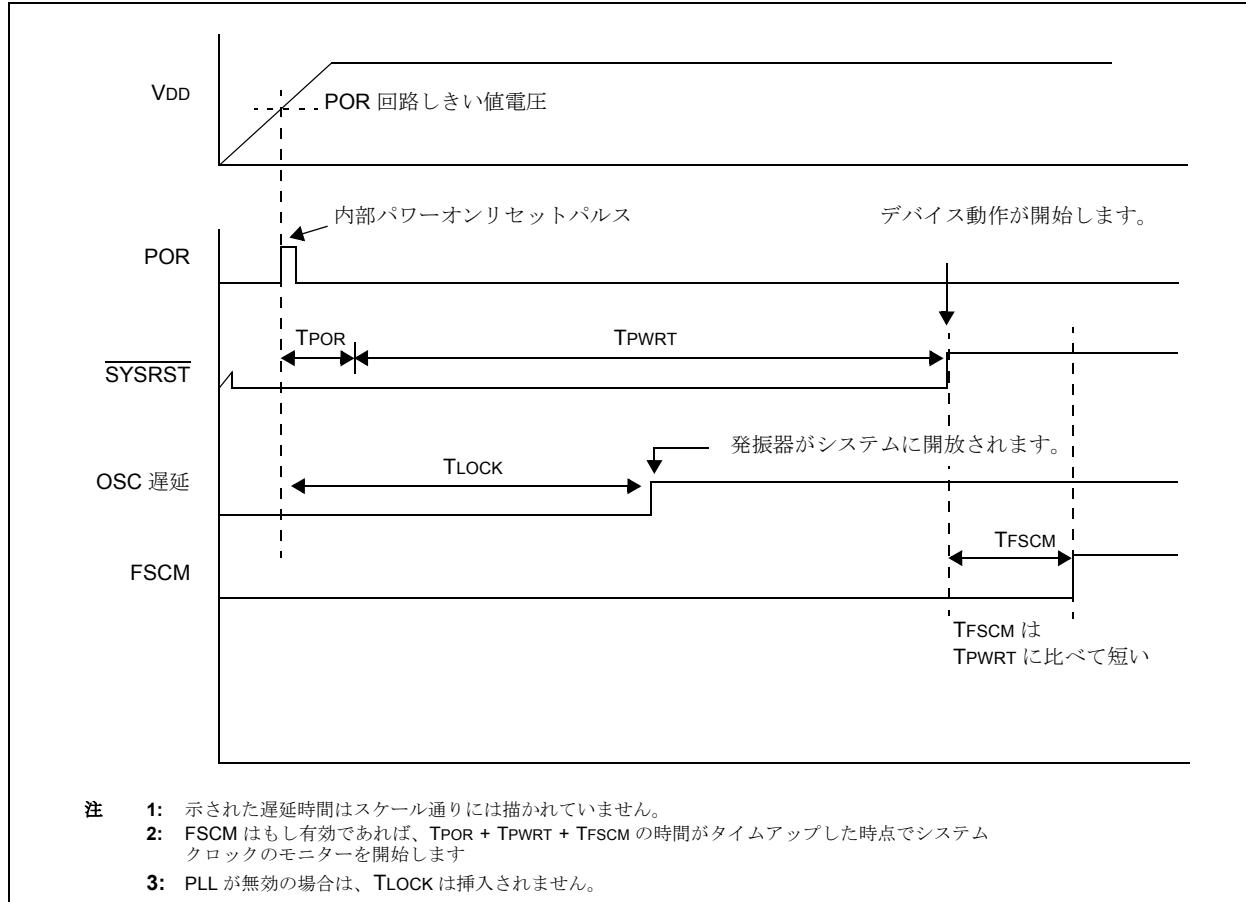
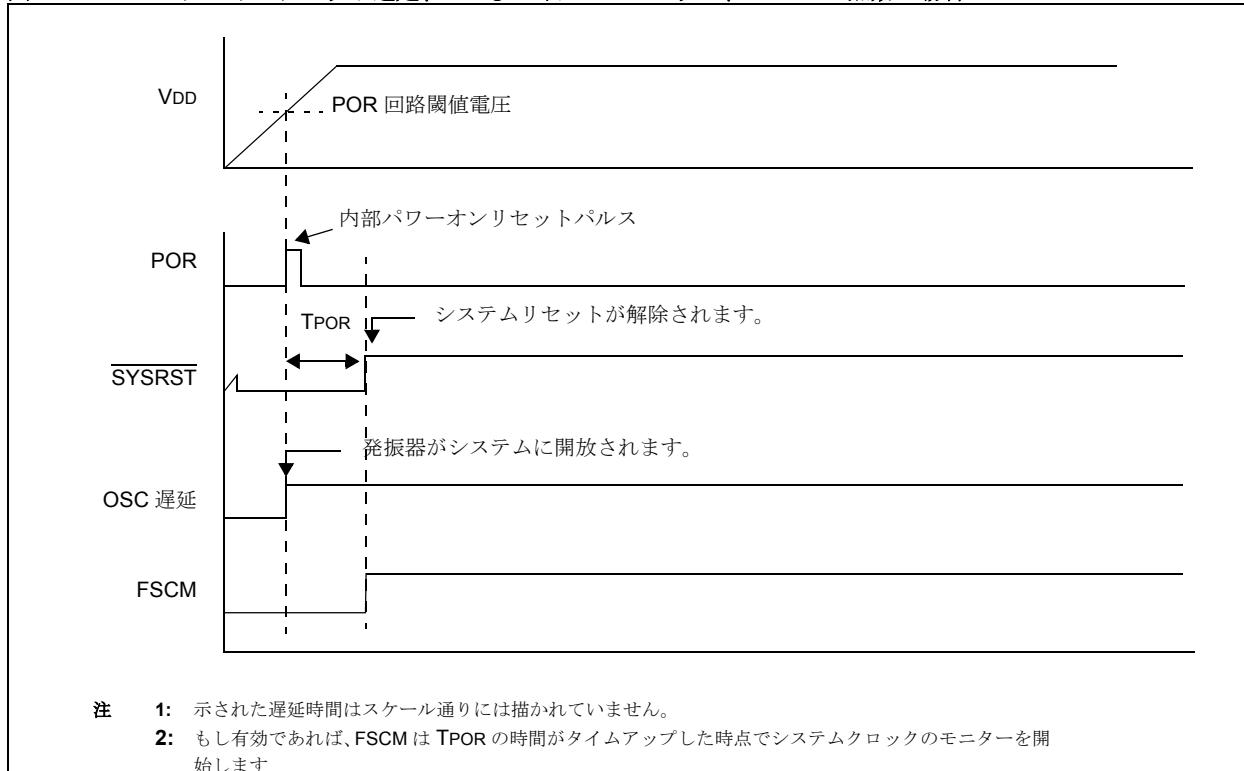


図 8-8 に示されるリセットタイムチャートは、PLL のない EC もしくは RC システムクロックが選択され、PWRT が無効の場合の例を示しています。この構成では最小リセット遅延となることに注意してください。POR 遅延だけがデバイス動作が始まる前に発生する唯一の遅延になります。FSCM が有効の場合でも FSCM 遅延は発生しません。なぜなら、システムクロックソースが水晶発振器もしくは PLL からドライブされていないからです。

図 8-8: デバイスリセット遅延、EC もしくは RC クロック、PWRT が無効の場合



## 8.11 特殊機能レジスタリセットステート

dsPIC30F CPU と周辺に関連するほとんどの特殊機能レジスタ (SFRs) は、デバイスリセット時には特定の値にリセットされます。SFRs は周辺もしくは CPU ごとにグコンフィギュレーションループ分けをされ、リセット値はこのマニュアルのそれぞれの章で規定される値になります。

それぞれの SFR のリセット値はリセットの種類に依存しませんが、2 つのレジスタは例外です。リセットコントロールレジスタ RCON のリセット値は、デバイスリセットの種類に依存します。発振器コントロールレジスタ OSCCON のリセット値は、リセットの種類と FOSC デバイスコンフィギュレーションレジスタ (表 8-1 参照) 内の発振器コンフィギュレーションビットのプログラム値に依存します。

## 8.12 設計の秘訣

## 質問 1: RCON レジスタはどのように使用するのですか?

回答: リセット後の初期化コードで RCON の中身をチェックし、リセットのソースを確認すべきです。あるアプリケーションでは、この情報はリセットを発生させる問題を訂正するために適切な動作をする際に使用されます。RCON レジスタ内のすべてのリセットステータスビットは、次のデバイスリセット後に RCON の値が意味のある結果であることを確実にするために、読み出した後にクリアする必要があります。

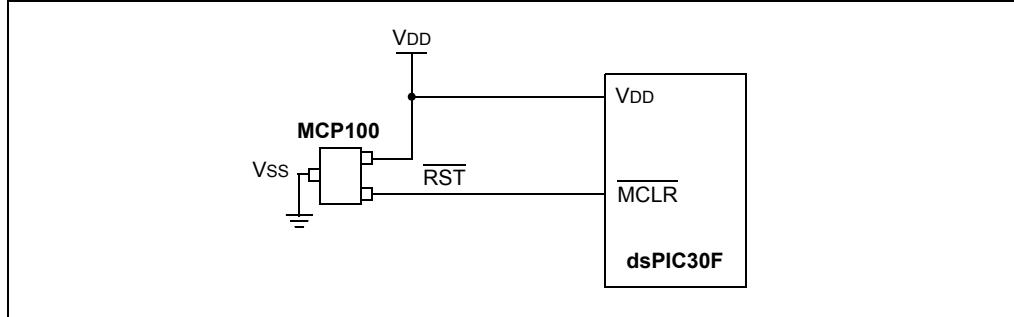
## 質問 2: 電池動作アプリケーションにおいて、BOR はどのように使用するのですか?

回答: BOR の特徴は、電池電圧低下検出として動作するようには設計されていませんので（電流を節約するために）電池動作システムでは無効にする必要があります。低電圧検出周辺モジュールが、電池が寿命電圧に達した時を検出するために用いられます。

## 質問 3: BOR モジュールは、私のアプリケーションで必要なプログラマブルトリップポイントを持っていません。これの対策はどのようにすればよいですか?

回答: デバイスのプログラマブル BOR トリップポイントレベルが、アプリケーションにとっては所望のレベルではないようないくつかのアプリケーションがあります。図 8-9 に、MCP100 システムスーパーバイザーを用いた、外付けブラウンアウト保護用の回路を示します。

図 8-9: MCP100 を用いた外付けブラウンアウト保護回路



## 質問 4: 16 ビットアドレスを持った W レジスタを初期化しましたが、レジスタはアドレスとして使用しようとすると、デバイスがあきらかにリセットしてしまいます。

回答: すべてのデータアドレスは 16 ビット値ですので、W レジスタ未初期化検出回路は、ワード単位でロードした場合にのみ、レジスタが正常に初期化されたかどうかを認識します。従って、バイトモードで 2 バイトを正常に W レジスタに書き込みできたとしても、正常に動作せず、結果的に、W レジスタがアドレスポインタとして使われた時点で、デバイスリセットが発生します。

## 8.13 関連するアプリケーションノート

この章では、マニュアルのこの章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは特に dsPIC30F 製品ファミリー用に書かれたものではありませんが、その概念は適切であり、修正して使用でき、制限がある場合もあります。現状、リセットモジュールに関連する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 #
パワーアップトラブルシューティング	AN607
パワーアップでの考慮事項	AN522

**注:** dsPIC30F ファミリーのデバイスについてのその他のアプリケーションノートやコードの例については、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 8.14 改訂履歴

### A版

これは本ドキュメントの初版です。

### B版

マニュアルの本章に対する技術内容や編集改訂版はありませんが、マニュアル全体を通してB版を反映するためにこの章は更新されています。

注：



**MICROCHIP**

---

---

## 第9章. 低電圧検出 (LVD)

---

---

### ハイライト

この章は、以下の項目を含んでいます。

9.1	はじめに.....	9-2
9.2	LVD 動作.....	9-5
9.3	設計の秘訣.....	9-6
9.4	関連するアプリケーションノート.....	9-7
9.5	改訂履歴.....	9-8

9

低電圧検出  
(LVD)

## 9.1 はじめに

LVD モジュールは電池動作アプリケーションに適用可能です。電池がエネルギーを流すにつれて、電池の電圧はゆっくりと低下します。電池のソース抵抗もエネルギーが無くなるにつれて増加していきます。LVD モジュールは電池の電圧（従って、デバイスの  $V_{DD}$ ）がしきい値以下に低下した時（これが当該アプリケーションにおける電池の寿命に近いと考えられます）を検出するために使用されます。これにより、当該アプリケーションの動作をうまく停止することができます。

LVD モジュールは、比較対象として内部基準電圧を使用します。しきい値電圧  $V_{LVD}$  は、実行中にプログラム可能です。

図 9-1 に考えられる電池電圧の曲線を示します。時間が経つとデバイス電圧は低下していきます。デバイス電圧が  $V_{LVD}$  に等しくなると、LVD 回路が割り込みを生成します。これは時刻  $T_A$  で発生します。アプリケーションソフトは、デバイス電圧が有効な最低動作電圧になる前に、システムを停止しなければなりません。電圧点  $V_B$  は有効動作電圧最低値です。この時刻が  $T_B$  とすると、停止のための総時間は  $T_B - T_A$  になります。

図 9-1: 典型的な低電圧検出応用

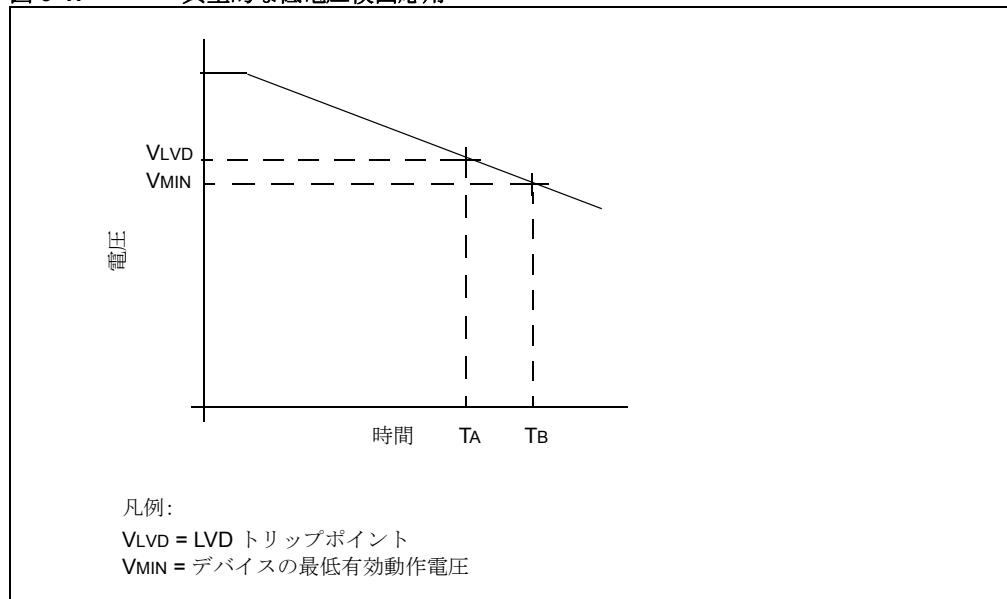
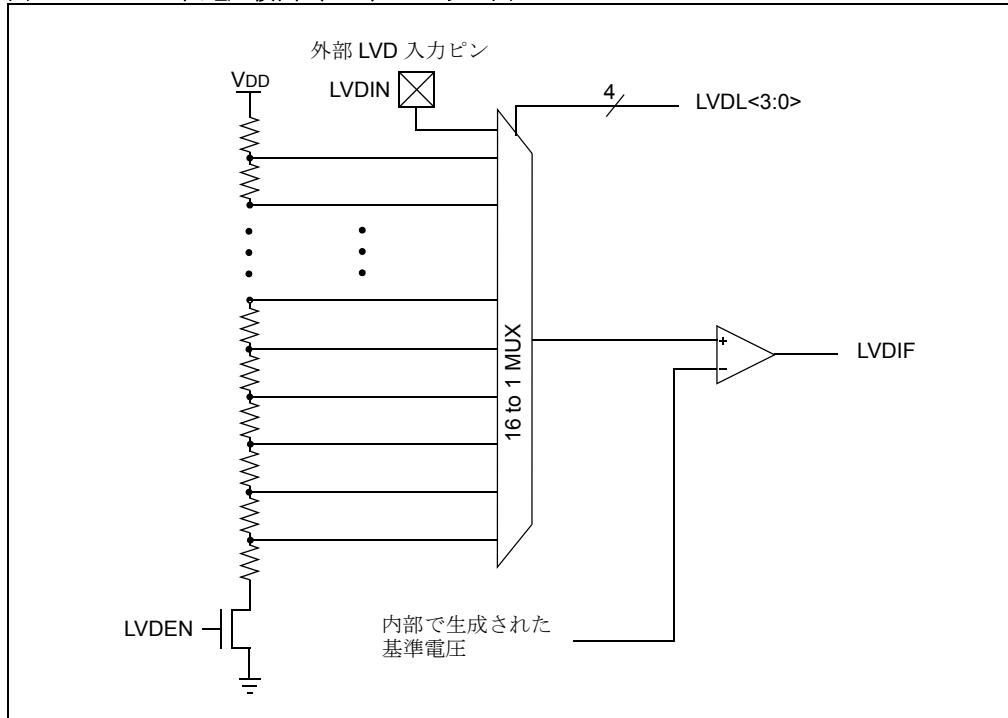


図 9-2 に LVD モジュールのブロック図を示します。比較器が設定ポイントとして内部で生成された基準電圧を使用します。デバイス電圧の選択タップからの出力電圧が基準電圧より低くなると、LVDIF ビット (IFS2<10>) がセットされます。

抵抗分割器内のそれぞれのノードは「トリップポイント」電圧となります。この 16 個の電圧のどの 1 つの値にもプログラム可能です。

図 9-2: 低電圧検出 (LVD) ブロック図



### 9.1.1 LVD 制御ビット

LVD モジュール制御ビットは RCON レジスタ内に配置されています。

LVDEN ビット (RCON<12>) は低電圧検出モジュールを有効にします。LVD モジュールは、 $LVDEN = 1$  の時、有効になります。消費電力が重要な場合は、電力節約を最大にするために、LVDEN ビットをクリアできます。

#### 9.1.1.1 LVD トリップポイントの選択

LVDL<3:0> ビット (RCON<11:8>) は LVD トリップポイントを選択します。VDD に接続された内部電圧分圧器から選択される 15 のトリップポイントオプションがあります。アプリケーションによって、どのトリップポイントオプションも適切でない場合は、LVD トリップ電圧を外部から LVDIN ピンに入力することができます。(ピン配置については、特定のデータシートを参照してください)。外部からの LVD 入力に対するトリップポイントの標準値は 1.24V です。LVD 外部入力オプションでは、希望する VDD で LVD 割り込みを発生させる外部電圧分圧回路の値をユーザーが選択する必要があります。

### 9.1.2 内部電圧基準

LVD は内部のバンドギャップ電圧基準回路を使っており、安定するまで標準的な時間を必要とします。詳しくは特定データシートの「電気的仕様」を参照してください。BGST ステータスピット (RCON<13>) は、バンドギャップ電圧基準が安定したことを表します。ユーザーは LVD モジュールが有効になったあと、ソフトウェアで BGST ステータスピットをポーリングする必要があります。安定化時間の終わりで、LVDIF ビット (IFS2<10>) をクリアする必要があります。セクション 9.2 「LVD 動作」内の LVD モジュールセットアップ手順を参照してください。

バンドギャップ電圧基準回路はデバイスのほかの周辺でも使用されますので、LVD モジュールが有効になる前に有効にされ（安定化され）ていることもあります。

## レジスタ 9-1: RCON : リセット制御レジスタ

上位バイト :

R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1			
TRAPR	IOPUWR	<b>BGST</b>	<b>LVDEN</b>	<b>LVDL&lt;3:0&gt;</b>						
ビット 15				ビット 8						

下位バイト :

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1			
EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR			
ビット 7				ビット 0						

ビット 13 **BGST:** バンドギャップ安定ビット

1 = バンドギャップは安定です。

0 = バンドギャップは不安定で、LVD 割り込みは禁止中です。

ビット 12 **LVDEN:** 低電圧検出有効ビット

1 = LVD を有効にし、LVD 回路に電力を供給開始します。

0 = LVD を無効にし、LVD 回路の電力を停止します。

ビット 11-8 **LVDL<3:0>:** 低電圧検出限界制御ビット

1111 = LVD の入力は LVDIN ピン (標準 1.24V しきい値)

1110 = 4.6V

1101 = 4.3V

1100 = 4.1V

1011 = 3.9V

1010 = 3.7V

1001 = 3.6V

1000 = 3.4V

0111 = 3.1V

0110 = 2.9V

0101 = 2.8V (リセット時のデフォルト値)

0100 = 2.6V

0011 = 2.5V

0010 = 2.3V

0001 = 2.1V

0000 = 1.9V

注： ここに示される電圧閾値はデザインのガイド用にのみ提供されます。詳しくはデバイスのデータシートの“電気的仕様”を参照してください。

凡例：

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未定ビット、'0' が読み込まれます

-n = POR の値

'1' = ビットがセット '0' = ビットはクリア x = ビットは不定

されます

注： RCON レジスタやその他のビットの説明については、第8章、「リセット」を参照してください。

## 9.2 LVD 動作

LVD モジュールは、デバイスが、デバイス電圧の状態をモニターできるので、アプリケーションを頑強にする機能を追加できます。デバイス電圧が有効動作電圧の下限に近い電圧領域になると、デバイスは、「正常な」停止を確実にするために変数などを保存します。

**注:** システムデザインは、デバイスが有効動作範囲を抜け出る前か、もしくはブラウンアウトリセットに入る前に、アプリケーションソフトウェアが変数などを保存するのに十分な時間を与えられることを確実にする必要があります。

デバイスの電源によっては、供給電圧が比較的ゆっくりと減少する場合もあるかもしれません。これは、LVD モジュールが常に動作している必要がないことを意味します。電流必要量を減少させるためには、LVD 回路は電圧がチェックされる短い期間のみ有効にすればよいのです。チェックの後は、LVD モジュールは無効になります。

### 9.2.1 LVD 初期化手順

以下の手順は、LVD モジュールを有効にするために必要です。

- 外部 LVD 入力ピン (LVDIN) を使用する場合、当該ピンに複合化されたすべてのその他の周辺回路は無効であり、TRISx レジスタ内の適切なビットをセットして、そのピンが入力になるように構成されていることを確認します。
- LVDL 制御ビット (RCON<11:8>) に適当な値を書き込み、LVD しきい値電圧を選択します。
- LVDIE ビット (IEC2<10>) をクリアして、LVD 割り込みが禁止であることを確認します。
- LVDEN ビット (RCON<12>) を設定して、LVD モジュールを有効にします。
- 必要であれば、BGST ステータスピット (RCON<13>) をポーリングすることにより、内部電圧基準が安定になるまで待ちます（セクション 9.1.2 「内部電圧基準」を参照してください）。
- 割り込みが許可される前に LVDIF ビット (IFS2<10>) がクリアされていることを確認します。LVDIF がセットされている場合は、デバイス V<sub>DD</sub> が選択された LVD しきい値電圧より低くなっています。
- LVDIP<2:0> 制御ビット (IPC10<10:8>) に書き込むことにより、LVD 割り込み優先度レベルをセットします。
- LVDIE 制御ビットをセットすることにより、LVD 割り込みを有効にします。

一旦 V<sub>DD</sub> がプログラムされた LVD しきい値以下に下がると、LVDIF ビットがセットされます。LVD モジュールが CPU に割り込みをかけると、ISR では以下の 2 つのうちの 1 つが実行できます。

- LVDIE 制御ビットがクリアされ、更なる LVD モジュール割り込みが禁止になり、適切な停止手順を実行します。  
もしくは
- LVDL 制御ビットを用いて LVD 電圧しきい値を減少させ、LDVIF ステータスピットをクリアします。このやり方は、徐々に下がっていく電圧を追跡するために使用されます。

### 9.2.2 LVD 動作の電流消費

LVD 回路は、ブラウンアウトリセット (BOR) モジュールのようなその他の周辺デバイスと共有する内部電圧基準回路に依存しています。内部電圧基準は、関連する周辺回路の 1 つが有効の場合はいつもでも動作します。このため、LVD モジュールを無効にしても電流消費の期待した低減は観測できません。

### 9.2.3 SLEEP と IDLE モードでの動作

LVD 回路は有効になると、SLEEP や IDLE モードでも動作を継続します。デバイス電圧がトリップポイントを過ぎると、LVDIF ビットがセットされます。

SLEEP もしくは IDLE モードから抜け出るために以下的手順を実行します。

- LVDIE ビット (IEC2<10>) がセットされている場合は、デバイスは SLEEP もしくは IDLE モードから起き上がります。
- LVD 割込み用の割り当てられた優先度が、現 CPU の優先度より小さいか同じ場合は、デバイスは起き上がり、SLEEP もしくは IDLE モードを起動した PWRSAV 命令に引き続く命令からコードの実行を継続します。
- LVD 割込み用の割り当てられた優先度が現 CPU の優先度より大きい場合は、デバイスは起き上がり、CPU 例外処理が開始されます。コード実行は、LVDISR の最初の命令から継続します。

## 9.3 設計の秘訣

**質問 1:** LVD 回路はランダムな割り込みを発生するように思えるのですが？

**回答 :** LVD を有効にする前に内部電圧基準が安定であることを確認します。これは、LVD モジュールが有効になった後で BGST ステータスピット (RCON<13>) をポーリングすることで実行できます。この時間遅延の後、LVDIF ビットをクリアし、それから LVDIE ビットをセットします。

**質問 2:** モジュールの電流消費を低減するにはどうしたらよいですか？

**回答 :** 低電圧検出はデバイス電圧をモニターするために使用されます。電源は通常電池でゆっくりと電圧が低下していきます。これは、LVD 回路がほとんどの時間で無効にでき、時々デバイスの電圧チェックをするためにのみ有効にできることを意味します。

**質問 3:** 電池駆動応用の場合 BOR 回路を有効にすべきですか？

**回答 :** BOR 回路は AC ライン電圧により引き起こされる、電源供給の変動による不適切な動作からデバイスを守ることを目的としています。BOR は一般に電池応用には必要でなく、電流消費低減のためには無効にできます。

## 9.4 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正して使用できますが制限がある場合もあります。現状、低電圧検出モジュールに関連するアプリケーションノートは以下の通りです。

題目	アプリケーションノート #
現在のところ、関連するアプリケーションノートはありません。	

**注：** dsPIC30F ファミリーのデバイスに関しての、他のアプリケーションノートやコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 9.5 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

本章に対する技術内容や編集改訂版はありませんが、マニュアル全体を通して B 版を反映するために、この章は更新されています。



**MICROCHIP**

## 第 10 章. ウオッチドッグタイマーおよび省電力モード

### ハイライト

この章は、以下の項目を含んでいます。

10.1 序章 .....	10-2
10.2 省電力モード .....	10-2
10.3 SLEEP モード .....	10-2
10.4 IDLE モード .....	10-4
10.5 省電力命令との同時割り込み .....	10-5
10.6 ウオッチドッグタイマ .....	10-5
10.7 設計の秘訣 .....	10-8
10.8 関連するアプリケーションノート .....	10-9
10.9 改訂履歴 .....	10-10

10

WDT オンチドッグタイマー  
省電力モード

## 10.1 序章

この章では、dsPIC30F デバイスファミリーのウォッチドッグタイマー (WDT) および省電力モードについて説明します。dsPIC デバイスには PWRSAV 命令の実行によって入ることができます。2 つの電力節減モードがあります。

- SLEEP モード : CPU、システムクロックソース、およびシステムクロックソースで動作する他の周辺装置をも無効化します。これはデバイスで最も低電力のモードです。
- IDLE モード : CPU が無効化されますが、システムクロックソースは動作し続けます。周辺機器は動作し続けますが、任意で無効化できます。

WDT を有効化すると、内部 LPRC クロックソースで動作し、ソフトウェアから WDT がクリアされなくなると、デバイスをリセットすることによってシステムソフトウェア故障の検知に使用できます。様々な WDT タイムアウト時間が WDT ポストスケーラを使用して選択できます。WDT はまた、SLEEP または IDLE モードからデバイスをウェイクアップするのに使用できます。

## 10.2 省電力モード

dsPIC30F デバイスファミリーには、SLEEP モードと IDLE モードの 2 つの省電力モードあり、PWRSAV 命令の実行により入ることができます。

PWRSAV 命令のアセンブリ構文は以下の通りです。

```
PWRSAV #SLEEP_MODE      ; Put the device into SLEEP mode  
PWRSAV #IDLE_MODE       ; Put the device into IDLE mode
```

**注：** SLEEP\_MODE および IDLE\_MODE は選択されたデバイスのファイルを含むアセンブリで定義される定数です。

省電力モードは有効化された割り込み、WDT タイムアウト、またはデバイスリセットの結果により終了できます。デバイスがこれら 2 つの動作モードのうち 1 つを終了する場合、「ウェイクアップ」といいます。省電力モードの特性は次の項にて説明します。

## 10.3 SLEEP モード

SLEEP モードの特性は以下の通りです。

- システムクロックソースはシャットダウンされます。オンチップ発振器が使用される場合はオフになります。
- デバイスの電流消費は電流をソースしている I/O ピンがない場合は最小値です。
- フェールセーフクロックモニター (FSCM) はシステムクロックソースが無効化されているため、SLEEP モード中は作動しません。
- LPRC クロックは、WDT が有効化されている場合、SLEEP モードでも作動し続けます。
- 低電圧検出回路が有効化されている場合、SLEEP モードでも動作を維持します。
- BOR 回路が有効化されている場合、SLEEP モードでも動作を維持します。
- WDT が有効化されている場合、SLEEP モードに入る前に自動的にクリアされます。
- 周辺装置の中には SLEEP モードでも動作し続けるものがあります。これらの周辺装置は入力信号の変化を検出する I/O ピン、または外部クロック入力を使用する周辺装置を含みます。システムクロックソースで作動している周辺装置はすべて SLEEP モードでは無効化されます。

以下のイベントのうち 1 つが発生するとプロセッサは SLEEP を終了または SLEEP からウェイクアップします。

- それぞれ有効化された割り込み要因
- すべての要因のデバイスリセット
- WDT タイムアウト

### 10.3.1 SLEEP からウェイクアップ時のクロック選択

SLEEP モードに入った場合にアクティブだった同じクロックソースをプロセッサは再起動します。

### 10.3.2 SLEEP からウェイクアップ時の遅延

SLEEP モードからウェイクアップする際の遅延は、運動して起動される発振器起動遅延を含めて表 10-1 に示されています。全ての場合、POR 遅延時間 ( $TPOR = 10 \mu\text{s}$  公称) が内部システム RESET 信号である SYSRST がリリースされる前に内部デバイス回路が安定できるように適用されます。

表 10-1: SLEEP モードを終了する際の遅延タイム

クロックソース	SYSRST 遅延	発振器遅延	FSCM 遅延	注
EC, EXTRC	TPOR	-	-	1
EC + PLL	TPOR	TLOCK	TFSCM	1, 3, 4
XT + PLL	TPOR	TOST + TLOCK	TFSCM	1, 2, 3, 4
XT, HS, XTL	TPOR	TOST	TFSCM	1, 2, 4
LP (SLEEP 中 OFF)	TPOR	TOST	TFSCM	1, 2, 4
LP (SLEEP 中 ON)	TPOR	-	-	1
FRC, LPRC	TPOR	-	-	1

注 1:  $TPOR = \text{パワーオンリセット遅延} (10 \mu\text{s} \text{ 標準})$ 。

2:  $TLOCK = \text{発振器起動タイマー}。システムに発振器クロックをリリースする前に 10 \text{ ビットカウンタで } 1024 \text{ クロックカウントをします。}$

3:  $TLOCK = \text{PLL ロックタイム} (20 \mu\text{s} \text{ 標準})$ 。

4:  $TFSCM = \text{フェールセーフクロックモニタ遅延} (100 \mu\text{s} \text{ 標準})$ 。

注: TPOR, TFSCM、および TLOCK 仕様値については dsPIC30F デバイスデータシートの「電気的仕様」の章を参照してください。

### 10.3.3 クリスタル発振器または PLL で SLEEP モードからウェイクアップ

システムクロックソースがクリスタル発振器または PLL (あるいはその両方) から供給されている場合、システムクロックソースがデバイスに対して使用可能になる前に発振器起動タイマー (OST) または PLL のロック時間 (あるいはその両方) を適用する必要があります。このルールの例外はシステムクロックソースが LP 発振器であり、SLEEP モード中にも動作している場合には発振器遅延は不要であることです。様々な遅延が適用されるにも関わらず、クリスタル発振器 (および PLL) は POR 遅延の後でも安定発振していないことがあります。

### 10.3.4 FSCM 遅延および SLEEP モード

以下の条件が成立する場合、SLEEP モードからウェイクアップする時には POR 遅延が終了した後に、さらに標準で  $100 \mu\text{s}$  の TFSCM が適用されます。

- 発振器が SLEEP モード中にシャットダウンされた場合。
- システムクロックがクリスタル発振器ソースまたは PLL (あるいはその両方) から供給されている場合。

FSCM 遅延はほとんどの場合、デバイス実行再開の前に OST が終了し、PLL が安定する時間を提供します。FSCM が有効化された場合、FSCM 遅延が終了した後にシステムクロックソースのモニターを開始します。

### 10.3.5 発振器のスロー起動

通常パワーアップ遅延が終了する前に OST および PLL ロックタイムが終了する可能性はありません。

従って FSCM が有効化された場合、デバイスはこの条件をクロック故障として検出し、クロック故障トラップが発生します。デバイスは FRC 発振器へと切り替わり、ユーザーはクロック故障トラップサービスルーチン内でクリスタル発振器ソースを再度有効化できます。

FSCM を有効にしない場合、デバイスは単純にクロックが安定するまで、実行コードを開始しません。ユーザーからは、デバイスは発振器クロックが開始するまでは SLEEP 状態にあるように見えます。

## 10.3.6 割り込み時 SLEEP からのウェイクアップ

IECx レジスタの対応する IE 制御ビットによって有効化された割り込みのどのソースも SLEEP モードからプロセッサをウェイクアップできます。デバイスが SLEEP モードからウェイクアップする場合、2 つのうち 1 つの動作が発生する可能性があります。

- ・割り込み要因に割り当てられた優先順位が現行の CPU 優先順位よりも低いまたは等しい場合、デバイスはウェイクアップし、SLEEP モードを開始した PWRSAV 命令の次の命令からコード実行を継続します。
- ・割り込み要因に割り当てられた優先度が現行の CPU 優先順位よりも高い場合、デバイスはウェイクアップし CPU 例外処理プロセスが開始されます。コード実行は ISR の最初の命令から継続します。

SLEEP ステータスピット (RCON<3>) はウェイクアップと設定されます。

## 10.3.7 RESET 時 SLEEP からのウェイクアップ

すべてのデバイス RESET は SLEEP モードからプロセッサをウェイクアップします。プロセッサをウェイクアップする RESET のどのソース (POR 以外) でもデバイスが以前に SLEEP モードであったことを示すため SLEEP ステータスピット (RCON<3>) をセットします。

パワーオンリセットでは、SLEEP ビットはクリアされます。

## 10.3.8 ウオッチドッグタイムアウト時 SLEEP からのウェイクアップ

ウォッチドッグタイマ (WDT) が有効になっているとき、デバイスが SLEEP 中に WDT のタイマアップが発生するとデバイスはウェイクアップします。SLEEP および WDTO ステータスピット (RCON<3>, RCON<4>) は両方とも、WDT 終了によりデバイスがウェイクアップしたことを見たためにセットされます。このイベントはデバイスをリセットしないことに注意してください。SLEEP モードを開始した PWRSAV 命令に続く命令から動作は継続します。

## 10.4 IDLE モード

デバイスが IDLE モードになる場合、以下のイベントが発生します。

- ・CPU が命令実行を停止する。
- ・WDT が自動的にクリア。
- ・システムクロックソースはアクティブのままで周辺モジュールはデフォルトでシステムクロックソースによる正常動作を継続します。周辺モジュールは任意で自身の「stop-in-idle」制御ビットを使用して IDLE モードのときは停止させることができます。(詳細は周辺モジュールの説明を参照してください)。
- ・WDT または FSCM が有効化された場合、LPRC もまたアクティブのままでです。

プロセッサは以下のイベントにて IDLE モードからウェイクアップします。

- ・それぞれ有効化された割り込み要因
- ・すべてのデバイス RESET
- ・WDT タイムアウト

IDLE からウェイクアップする場合、クロックは CPU へ再度供給され、PWRSAV 命令に続く命令または ISR の最初の命令から命令実行がすぐに始まります。

## 10.4.1 割り込み時 IDLE からのウェイクアップ

IECx レジスタの対応する IE 制御ビットにより有効化され、現行の CPU 優先度を超える割り込み要因であればすべて、IDLE モードからプロセッサをウェイクアップできます。デバイスが IDLE モードからウェイクアップする場合、2 つのオプションのうち 1 つが発生します。

- ・割り込み要因に割り当てられた優先度が現行の CPU 優先度よりも低いまたは等しい場合、デバイスはウェイクアップし、IDLE モードを開始する PWRSAV に続く命令からコード実行を継続します。
- ・割り込み要因に割り当てられた優先度が現行 CPU 優先度よりも高い場合、デバイスはウェイクアップし、CPU 例外処理プロセスが開始されます。コード実行は ISR の最初の命令から継続します。

IDLE ステータスピット (RCON<2>) がウェイクアップ時にセットされます。

#### 10.4.2 RESET 時 IDLE からのウェイクアップ

POR 以外のどの RESET でも、IDLE モードから CPU をウェイクアップすることができます。POR 以外のどのデバイス RESET でも、デバイスが以前 IDLE モードだったことを示すため、IDLE ステータスピットをセット (RCON<2>) します。電源 ON リセットの場合、IDLE ビットはクリアされます。

#### 10.4.3 WDT タイムアウト時 IDLE からのウェイクアップ

WDT が有効化された場合、プロセッサは WDT タイムアウト時に IDLE モードからウェイクアップし、IDLE モードを開始した PWRSAV 命令の次の命令からコード実行を継続します。WDT タイムアウトはこの場合デバイスをリセットしません。WDTO および IDLE ステータスピットは (RCON<4>, RCON<2>) 両方ともセットされます。

#### 10.4.4 IDLE モードからウェイクアップ時のタイム遅延

SLEEP モードからウェイクアップするのとは異なり、IDLE モードからウェイクアップする際の遅延はありません。システムクロックは IDLE モード中に動作しており、従って、ウェイクアップ時起動時間は全く必要ありません。

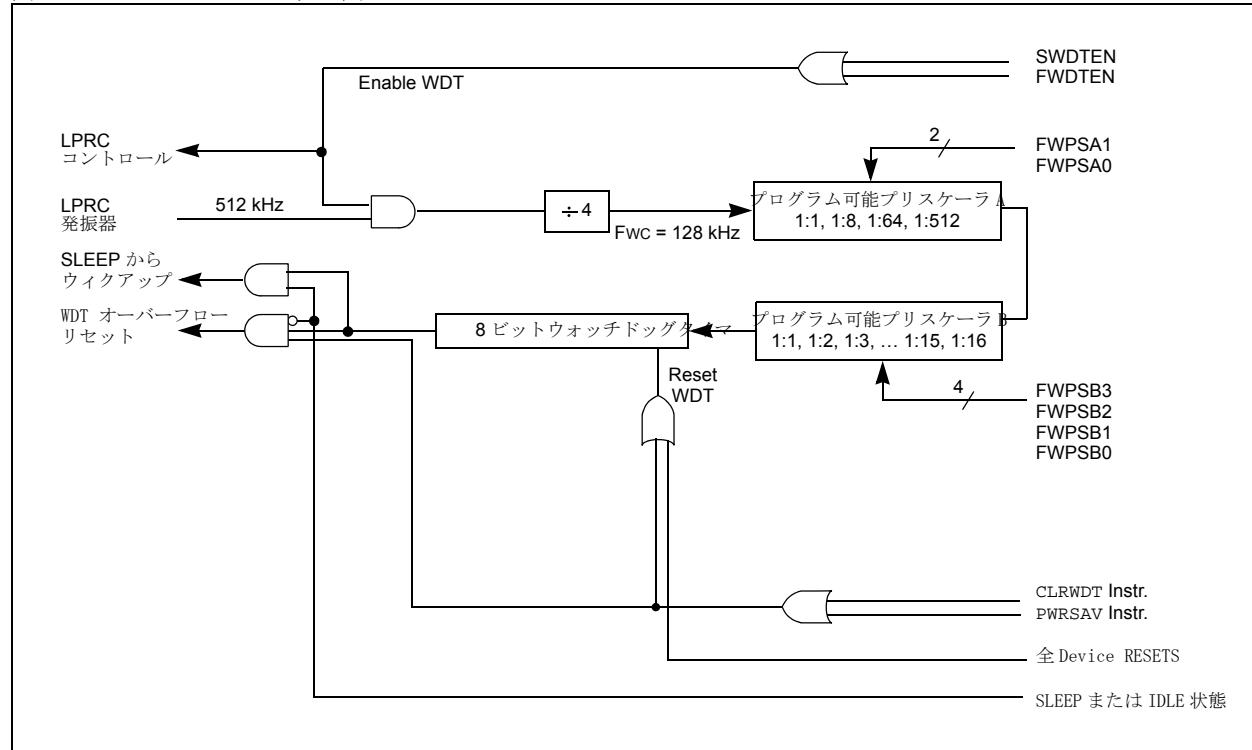
### 10.5 省電力命令との同時割り込み

PWRSAV 命令の実行と同時に割り込みは SLEEP または IDLE モードへの移行が完了するまで保留されます。その後でデバイスは SLEEP または IDLE モードからウェイクアップします。

### 10.6 ウオッチドッグタイマ

ウォッチドッグタイマ (WDT) の主要機能はソフトウェア異常が発生した時にプロセッサをリセットすることです。WDT はフリーランのタイマーであり、外部部品を全く必要としない内部 LPRC 発振器上で動作します。従って、WDT タイマーはシステムクロックソース（例：クリスタル発振器）が機能しなくても動作し続けます。WDT のブロック図は図 10-1 にて表示されています。

図 10-1: WDT ブロック図



## 10.6.1 WDT の有効化と無効化

FWDT デバイスコンフィギュレーションレジスタの FWDTEN コンフィギュレーションビットによって WDT が有効化または無効化されます。FWDT コンフィギュレーション値はデバイスプログラミング中に書き込まれます。FWDTEN コンフィギュレーションビットがセットされると、WDT は有効化されます。これがイレーズされたデバイスのデフォルト値です。FWDT デバイスコンフィギュレーションレジスタについての詳細は 第24章.「デバイスコンフィギュレーション」を参照してください。

### 10.6.1.1 ソフトウェア制御の WDT

FWDTEN デバイスコンフィギュレーションビットがセットされた場合、WDT は常に有効化されます。ただし、FWDTEN コンフィギュレーションビットが ‘0’ にプログラムされた場合、ユーザーソフトウェアで WDT を任意に制御できます。

SWDTEN 制御ビット (RCON<5>) をセットすると WDT はソフトウェア上で有効化されます。SWDTEN 制御ビットはデバイス RESET によってクリアされます。ユーザーはソフトウェア WDT オプションを使用してクリティカルコードセグメント用に WDT を有効化でき、省電力を最大化するために非クリティカルセグメント中に WDT を無効化できます。

## 10.6.2 WDT 操作

WDT は有効化されるとオーバーフローまたは「タイムアウト」するまでカウントアップします。WDT タイムアウトは SLEEP または IDLE モード中以外は強制的にデバイスリセットします。WDT タイムアウトリセットを抑制するには、ユーザーは CLRWDT 命令を使用して、ウォッチドッグタイマーを定期的にクリアする必要があります。更に、CLRWDT 命令は WDT プリスケーラもクリアします。

WDT が SLEEP または IDLE モード中にタイムアウトした場合、デバイスはウェイクアップし、PWRSAV 命令が実行されたところからコード実行を継続します。

どちらの場合でも、WDTO ビット (RCON<4>) はデバイス RESET またはウェイクアップイベントが WDT タイムアウトによって発生したと示すようセットされます。WDT が CPU を SLEEP または IDLE モードからウェイクアップする場合、SLEEP ステータスピット (RCON<3>) または IDLE ステータスピット (RCON<2>) もまたデバイスが省電力モードに入っていたことを示すようセットされます。

## 10.6.3 WDT タイマ時間の選択

WDT クロックソースは標準周波数 512 kHz の内部 LPRC 発振器です。LPRC クロックはをさらに 4 分周して 128 kHz のクロックを WDT に提供します。WDT のカウンタは 8 ビット幅なので、WDT の標準タイムアウト ( $T_{WDT}$ ) は 2 ミリ秒です。

### 10.6.3.1 WDT プリスケーラ

WDT には多種のタイムアウト時間を可能にするため、プリスケーラ A とプリスケーラ B という、2 つのクロックプリスケーラがあります。プリスケーラ A は 1:1、1:8、1:64 または 1:512 の分周比が設定可能です。プリスケーラ B は 1:1 から 1:16 までのどの分割比でも設定可能です。従って 2ms から 16 秒（標準）間を範囲とするタイムアウト時間がプリスケーラを使用して実現可能です。

プリスケーラの設定は FWDT デバイスコンフィギュレーションレジスタにある FWPSA<1:0> (プリスケーラ A) および FWPSB<3:0> (プリスケーラ B) コンフィギュレーションビットを使用して選択します。FWPSA<1:0> および FWPSB<3:0> 値はデバイスプログラミング中に書き込まれます。WDT プリスケーラ コンフィギュレーションビットについて詳しくは 第24章.「デバイスコンフィギュレーション」を参照してください。

WDT のタイムアウト時間は以下で計算できます。

式 10-1: WDT タイムアウト時間

$$\text{WDT ピリオド} = 2 \text{ ms} \times \text{プリスケール A} \times \text{プリスケール B}$$

**注：** WDT タイムアウト時間は直接 LPRC 発振器の周波数に関係しています。LPRC 発振器の周波数はデバイス動作電圧と温度により変動します。LPRC クロック周波数の仕様について詳しくは dsPIC30F デバイスデータシートを参照してください。

表 10-2 は様々なプリスケーラ値ごとのタイムアウト時間を示します。

表 10-2: WDT タイムアウト時間 vs. プリスケール A とプリスケール B 設定

プリスケーラ B 値	プリスケーラ A 値			
	1	8	64	512
1	2	16	128	1024
2	4	32	256	2048
3	6	48	384	3072
4	8	64	512	4096
5	10	80	640	5120
6	12	96	768	6144
7	14	112	896	7168
8	16	128	1024	8192
9	18	144	1152	9216
10	20	160	1280	10240
11	22	176	1408	11264
12	24	192	1536	12288
13	26	208	1664	13312
14	28	224	1792	14336
15	30	240	1920	15360
16	32	256	2048	16384

注： 全ての時間値はミリ秒単位。

#### 10.6.4 ウオッチドッグタイマーのリセット

WDT および全てのプリスケーラは下記のときリセットされます。

- すべてのデバイス RESET
- PWRSAV 命令が実行された場合（つまり SLEEP または IDLE モードに入る場合）
- 正常実行中の CLRWDT 命令実行時

#### 10.6.5 SLEEP モードおよび IDLE モードでの WDT 動作

WDT が有効化されている場合、SLEEP または IDLE モード中も動作し続けます。WDT タイムアウトが発生した場合、デバイスはそのデバイスをウェイクアップし、コード実行は PWRSAV 命令が実行されたところから継続します。

WDT は定期的に SLEEP モードからウェイクアップさせては、システムの状態をチェックし必要な動作をさせるということができるでの、低電力システムに有用です。SWDTEN ビットはこの点に関してとても有用です。WDT が通常操作中に無効化されている場合、(FWDTEN =0)、SWDTEN ビット (RCON<5>) が SLEEP モードに入る直前に WDT を ON にするために使用可能です。

## 10.7 設計の秘訣

**質問 1:** メインソフトウェアループに CLRWDT 命令を挿入したのにもかかわらず、デバイスリセットするのですが。

**回答:** CLRWDT 命令を含むソフトウェアループが WDT の最小仕様（標準値ではない）を満たしていることを確認してください。更に、割り込み処理時間が含まれていないかを確認してください。

**質問 2:** SLEEP または IDLE モードに入る前にソフトウェアは何をしたらよいでしょうか。

**回答:** デバイスをウェイクアップするようにする割り込み要因の IE ビットをかならずセットしてください。更に、割り込みの要因がデバイスのウェイクアップ機能をもっているか確認してください。デバイスが SLEEP モードの場合機能しない割り込み要因もあります。

デバイスが IDLE モードになっている場合、各デバイス周辺装置の「stop-in-idle」制御ビットが正常に設定されていることを確認してください。この制御ビットは周辺装置が IDLE モードでも継続して動作するかどうかを決定します。詳細は本マニュアルの各周辺装置セクションを参照してください。

**質問 3:** どうすればどの周辺装置がデバイスを SLEEP または IDLE モードからウェイクアップさせたのかを分かりますか。

**回答:** ウェイクアップ要因として有効化された各割り込み要因の IF ビットをポールすることで区別できます。

## 10.8 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正し、制限を設けて（必要な場合）使用可能です。現状、ウォッチドッグタイマーおよび省電力モードに関連する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 #
PICmicro® マイクロコントローラを使用した省電力設計	AN606

**注：** dsPIC30F ファミリーデバイスの他のアプリケーションノートおよびコードの例に関しては Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧ください。

## 10.9 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

マニュアルの本章に対する技術内容や編集改訂版はありませんが、マニュアル全体を通して B 版を反映するために、この章は更新されています。

## 第 11 章 . I/O ポート

### ハイライト

この章は、以下の項目を含んでいます。

11.1 序章 .....	11-2
11.2 入出力ポート制御レジスタ .....	11-3
11.3 周辺装置マルチプレクサ .....	11-4
11.4 ポートの詳細説明 .....	11-5
11.5 状態変化通知 (CN) ピン .....	11-5
11.6 SLEEP および IDLE モードでの CN 動作 .....	11-6
11.7 関連するアプリケーションノート .....	11-9
11.8 改訂履歴 .....	11-10

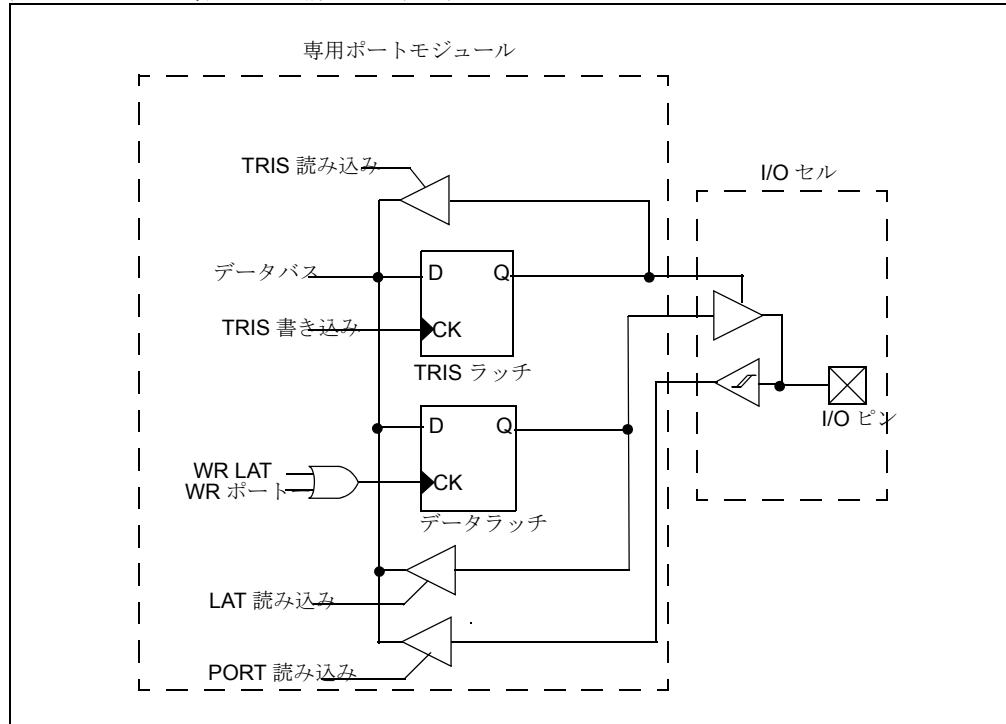
## 11.1 序章

この章では、dsPIC30F ファミリーデバイスの入出力ポートの情報を提供します。デバイスピ  
ンの全て（V<sub>DD</sub>、V<sub>SS</sub>、MCLR、および OSC1/CLK1 を除く）は周辺装置および汎用入出力ポー  
トとの間で共有されます。

汎用入出力ポートは dsPIC30F が他のデバイスのモニタおよび制御することを可能にします。  
ほとんどの入出力ピンは代替機能で多重使用されます。多重使用はデバイスの周辺装置機能に  
左右されます。一般的に、I/O ピンを周辺機器用に使っている場合、そのピンは汎用入出力ピ  
ンとして使うことはできません。

図 11-1 では標準入出力ポートのブロック図を示しています。このブロック図は入出力ピン上で  
多重使用される可能性がある周辺装置機能を考慮していません。

図 11-1: 専用ポート構造ブロック図



## 11.2 入出力ポート制御レジスタ

全ての入出力ポートにはポート操作に直接対応するレジスタが3つあります。ここで「x」は特定の入出力ポートを区別する文字です。

- TRISx: データ方向レジスタ
- PORTx: 入出力ポートレジスタ
- LATx: 入出力ラッチレジスタ

デバイス上の各入出力ピンが TRIS、PORT、および LAT レジスタの各ビットに対応します。

**注:** ポートおよび使用可能入出力ピンの総数はデバイスの種類に依ります。あるデバイスでは、ポート制御レジスタの一部のビットが実装されていないことがあります。詳細については個別のデバイスデータシートを参照してください。

### 11.2.1 TRIS レジスタ

TRISx レジスタ制御ビットは入出力ポートの各ピンが入力か出力かを決定します。入出力ピンの TRIS ビットが「1」の場合、そのピンは入力用に設定されます。入出力ピンの TRIS ビットが「0」の場合、そのピンは出力用に設定されます。「1」は(入力)の「I」、「0」は(出力)の「O」とすると簡単に覚えられるでしょう。全てのポートピンはリセット後には入力として定義されます。

### 11.2.2 PORT レジスタ

入出力ピンのデータは PORTx レジスタを介してアクセスされます。PORTx レジスタへの書き込みはポートデータラッチへ書き込みする一方で、PORTx レジスタの読み込みは入出力ピンの値を読み込みます。

BSET および BCLR 命令のような多くの命令は、「読み込み一変更一書き込み」操作です。従って、ポートへの書き込みは、まずポートピンが読み出され、この値が変更され、それからポートデータラッチへ書き込まれるということを意味します。ポートの一部の入出力ピンが入力として設定されている場合、「読み込み一変更一書き込み」コマンドを PORTx レジスタ上で使用するには注意が必要です。入力として設定された入出力ピンが後に出力に変更される場合、入出力ピン上に予想外の値が出力されることがあります。この現象は、「読み込み一変更一書き込み」命令が入力ピンの現在値を読み込み、その値をポートデータラッチへ書き込むために起ります。

### 11.2.3 LAT レジスタ

入出力ピンに対応した LATx レジスタは「読み込み一変更一書き込み」命令で発生する可能性がある問題を排除します。LATx レジスタの読み込みは、入出力ピンの値の代わりに、ポート出力ラッチで保持された値を戻します。入出力ポートと関連している LAT レジスタ上で「読み込み一変更一書き込み」操作は、入力ピン値をポートラッチへ書き込む可能性を回避します。LATx レジスタへの書き込みは PORTx レジスタへの書き込みと同じ結果となります。

PORT と LAT レジスタの違いは以下のようにまとめられます。

- PORTx レジスタへの書き込みはデータ値をポートラッチへと書き込む。
- LATx レジスタへの書き込みはデータ値をポートラッチへと書き込む。
- PORTx レジスタの読み込みは入出力ピン上のデータ値を読み出す。
- LATx レジスタの読み込みはポートラッチで保持されているデータ値を読み出す。

あるデバイスで使われないビットに関連するデータおよび制御レジスタは無効状態となります。これはつまり、対応する LATx と TRISx レジスタ、そしてポートピンは 0 として読み込まれます。

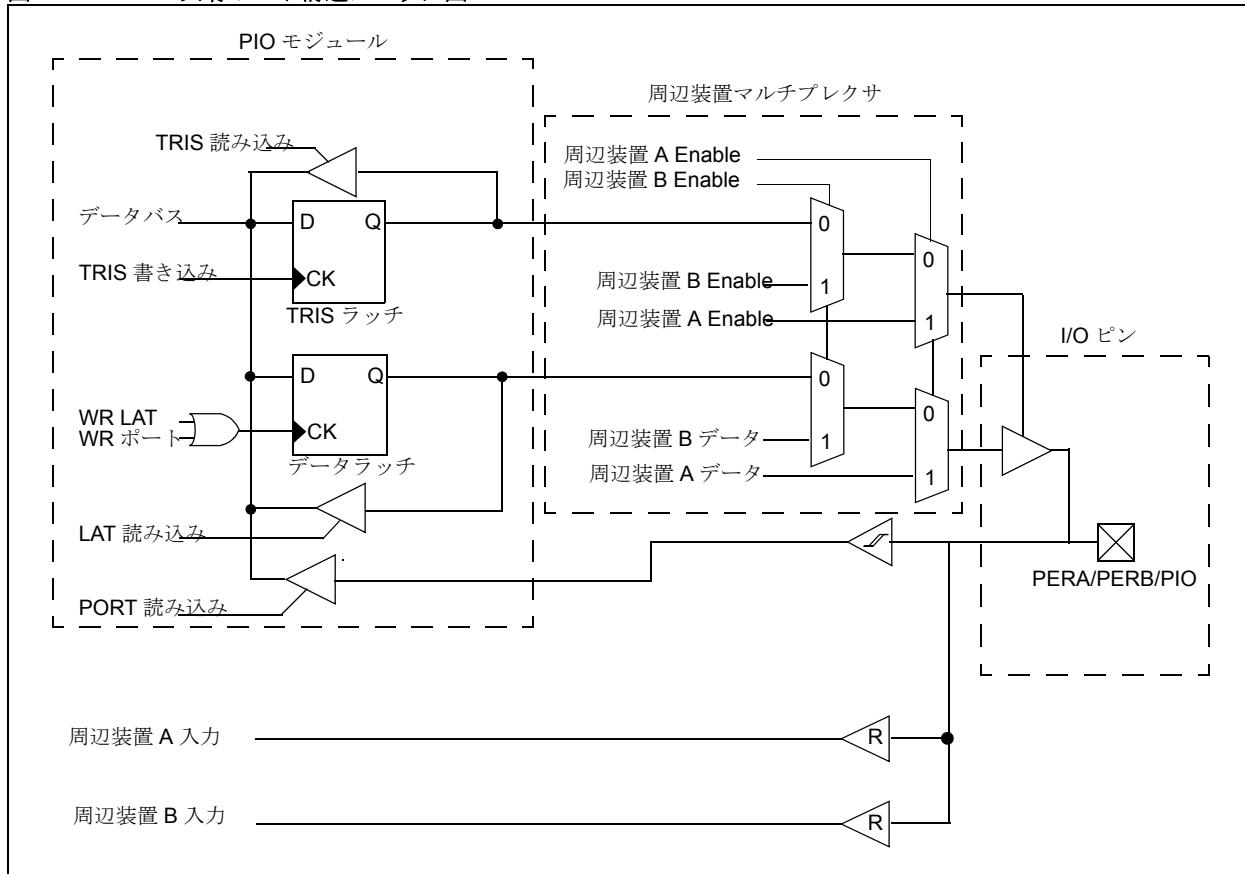
## 11.3 周辺装置マルチプレクサ

周辺装置が有効化される場合、周辺で使用されるピンは、いずれも汎用入出力ピンとしては使えなくなります。入出力ピンは入力データパスを介して読み出せることもありますが、入出力ポートの出力ドライバは無効化されます。

別の周辺装置とピンを共有する入出力ポートは常にその周辺装置に対しての部分集合となります。周辺装置の出力バッファデータおよび制御信号はペアのマルチプレクサへと渡されます。マルチプレクサは周辺装置、または関連ポートのどちらが出力データの所有権および入出力ピンの制御信号をもつのか選択します。図 11-2 では、ポートがどのようにして他の周辺装置と共有されるのか、入出力ピンがどこへ接続されるのかを説明しています。

**注:** PORTB ピンをデジタル入出力用に使用するためには、A/D モジュールが OFF になっていても、ADPCFG レジスタ上の対応ビットを ‘1’ に設定する必要があります。

図 11-2: 共有ポート構造ブロック図



### 11.3.1 複数周辺装置との入出力多重使用

ある dsPIC30F デバイスにとって、特に入出力ピン数の少ないものは、複数周辺装置機能が各入出力ピンに多重化されることがあります。図 11-2 では、同じ入出力ピンに対して多重化された 2 つの周辺装置の例を説明しています。

入出力ピンの名称はそのピンに関連する各機能の優先順位を定義します。図 11-2 で示されている概念的入出力ピンには、「PERA/PERB/PIO」と名付けられている「周辺装置 A」と「周辺装置 B」という 2 つの多重化された周辺装置があります。

入出力ピンの名称はユーザーがピンに関連した機能の優先順位を簡単に見分けられるようつづられています。図 11-2 の例では、周辺装置 A がピンの制御での優先度が一番高くなっています。周辺装置 A と周辺装置 B が同時に有効化される場合、周辺装置 A がその入出力ピンを制御します。

### 11.3.1.1 ソフトウェア入力ピン制御

I/O ピンに関する機能で入力だけしか使わず出力ドライバを制御しないものがあります。そのような周辺装置の例は入力キャプチャモジュールです。その入力キャプチャに対応する入出力ピンを TRIS 制御ビットを使って出力に設定することで、ユーザーは入力キャプチャの状態に応じて PORT レジスタを通じて手動で影響を与えることが可能です。この動作は、外部信号が入力ピンに接続されていないときに、テスト目的で特に有用です。

図 11-2 を参照すると、周辺装置マルチプレクサの構成で、出力 PORT レジスタを使用して、周辺装置入力ピンがソフトウェア上で操作可能かどうかを決定します。この図で示されている概念的な周辺装置は周辺装置機能が有効化された場合、出力 PORT データを入出力ピンから切断します。

一般的に、以下の周辺装置は入力ピンが出力 PORT レジスタを介して手動で制御できるようになります。

- 外部割り込みピン
- タイマクロック入力ピン
- 入力キャプチャピン
- PWM FAULT ピン

ほとんどのシリアル通信周辺装置は、いったん有効化されると入出力ピンを完全に制御します。従って、周辺装置の入力ピンは対応する出力 PORT レジスタを通じて影響を与えられません。このような周辺装置は以下の通りです。

- SPI
- I<sup>2</sup>C<sup>TM</sup>
- DCI
- UART
- CAN

### 11.4 ポートの詳細説明

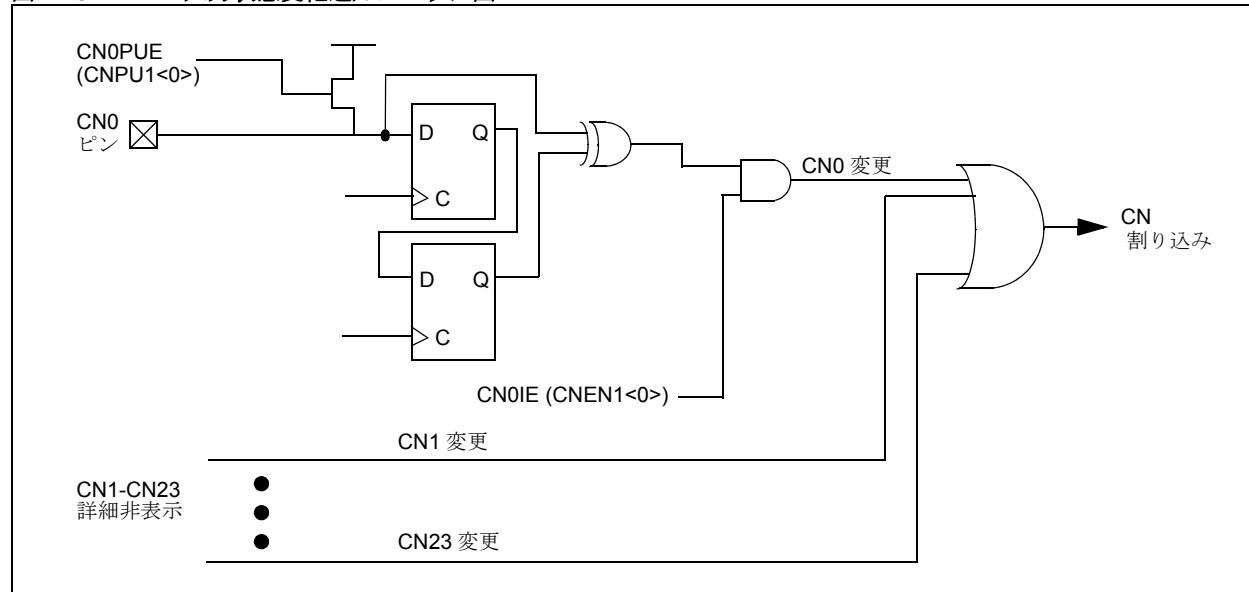
使用可能な入出力ポートおよび周辺装置マルチプレクシングの詳細についてはデバイスデータシートを参照してください。

### 11.5 状態変化通知 (CN) ピン

状態変化通知 (CN) ピンによって dsPIC30F デバイスは選択した入力ピン状態の変化に応じてプロセッサに対し割り込み要求を生成します。24 の入力ピンまで CN 割り込み生成に選択 (有効化) できます。使用可能な CN 入力の総数は選択した dsPIC30F デバイスによります。詳細はデバイスデータシートを参照してください。

図 11-3 は CN ハードウェアの基本機能を示しています。

図 11-3: 入力状態変化通知ブロック図



## 11.5.1 CN 制御レジスタ

CN モジュールには 4 つの関連制御レジスタがあります。CNEN1 および CNEN2 レジスタは、「x」が CN 入力ピンの数を意味する CNxIE 制御ビットを含みます。CN 入力ピンが CPU に割り込むために CNxIE ビットをセットしてください。

CNPU1 および CNPU2 レジスタは CNxPUE 制御ビットを含みます。各 CN ピンは CNxPUE 制御ピンでオンオフできるプルアップ抵抗を持っています。プルアップ抵抗はピンへの電流源として動作し、押しボタンまたはキー・パッド・デバイスが接続されている場合、外部抵抗を不要とすることができます。CN プルアップ・デバイスの電流仕様について詳しくはデバイスデータシートの「電気的仕様」の章を参照してください。

## 11.5.2 CN 構成および操作

CN ピンは以下のように設定します。

1. CN ピンが TRISx レジスタの対応ビットをセットすることでデジタル入力として確実に設定されるようにする。
2. CNEN1 および CNEN2 レジスタの適切なビットをセットすることで、選択した CN ピンの割り込みを有効化する。
3. CNPU1 および CNPU2 レジスタの適切なビットをセットすることで選択した CN ピン用に弱プルアップをオンにする（必要な場合）。
4. CNIF (IFS0<15>) 割り込みフラグをクリアする。
5. CNIP<2:0> 制御ビット (IPC3<14:12>) を使用して、CN 割り込みの希望する割り込み優先順位を選択する。
6. CNIE (IEC0<15>) 制御ビットを使用して、CN 割り込みを有効化する。

CN 割り込みが発生した場合、ユーザーは CN ピンに対応する PORT レジスタを読み出す必要があります。これにより不一致条件が解消され、次のピン変化を検出するために CN ロジックが設定されます。現行 PORT 値は、変化のあったピンを判断するため最終 CN 割り込み時に取得された PORT 読み込み値と比較されます。

CN ピンには最小入力パルス幅仕様があります。詳しくはデバイスデータシートの「電気的仕様」の章を参照してください。

## 11.6 SLEEP および IDLE モードでの CN 動作

CN モジュールは SLEEP または IDLE モード中も継続して動作します。有効化された CN ピンのうち 1 つが状態を変更した場合、CNIF(IFS0<15>) ステータスピットがセットされます。CNIE ビット (IEC0<15>) がセットされていると、デバイスは SLEEP または IDLE モードからウェイクアップし、実行を再開します。

CN 割り込みの割り当てられた優先度が現行の CPU 優先度に等しいまたは低い場合、デバイス実行は SLEEP または IDLE 命令のすぐ次の命令から継続します。

CN 割り込みの割り当てられた優先度が現行の CPU 優先度よりも高い場合、デバイス実行は CN 割り込みベクトルアドレスから継続します。

## レジスタ 11-1: CNEN1: 入力状態変化通知割り込み有効化レジスタ 1

上位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN15IE	CN14IE	CN13IE	CN12IE	CN11IE	CN10IE	CN9IE	CN8IE
ビット 15							ビット 8

下位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN7IE	CN6IE	CN5IE	CN4IE	CN3IE	CN2IE	CN1IE	CN0IE
ビット 7							ビット 0

ビット 15-0 **CNxIE:** 入力状態変化通知割り込み有効化ビット

1 = 入力状態変化時の割り込み有効化

0 = 入力状態変化時の割り込み無効化

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、'0' が読み込まれます

-n = POR での値

'1' = ビットがセッ '0' = ビットはクリ x = ビットは不定です  
トされます アされます

## レジスタ 11-2: CNEN2: 入力状態変化通知割り込み有効化レジスタ 2

上位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	—	—	—
ビット 15							ビット 8

下位バイト :

R/W-0 R/W-0 R/W-0 R/W-0 R/W-0 R/W-0 R/W-0 R/W-0

CN23IE CN22IE CN21IE CN20IE CN19IE CN18IE CN17IE CN16IE

ビット 7

ビット 0

ビット 15-8 未実装: '0' で読み込まれます

15-8

**CNxIE:** 入力状態変化通知割り込み有効化ビット

1 = 入力状態変化時の割り込み有効化

0 = 入力状態変化時の割り込み無効化

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、'0' が読み込まれます

-n = POR での値

'1' = ビットがセッ '0' = ビットはクリ x = ビットは不定です  
トされます アされます

## レジスタ 11-3: CNPU1: プルアップ有効レジスタ 1

上位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN15PUE	CN14PUE	CN13PUE	CN12PUE	CN11PUE	CN10PUE	CN9PUE	CN8PUE
ビット 15							ビット 8

下位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN7PUE	CN6PUE	CN5PUE	CN4PUE	CN3PUE	CN2PUE	CN1PUE	CN0PUE
ビット 7							ビット 0

ビット **CNxPUE:** プルアップ有効化ビット

- 15-0  
1 = プルアップ有効化  
0 = プルアップ無効化

凡例 :

R = 読み込み可能ビット	W = 書き込み可能ビット	U = 未実装、'0' が読み込まれます
-n = POR での値	'1' = ビットがセッ	'0' = ビットはクリ
	トされます	x = ビットは不定です

## レジスタ 11-4: CNPU2: プルアップ有効レジスタ 2

上位バイト :							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
ビット 15							ビット 8

下位バイト :

| R/W-0   |
|---------|---------|---------|---------|---------|---------|---------|---------|
| CN23PUE | CN22PUE | CN21PUE | CN20PUE | CN19PUE | CN18PUE | CN17PUE | CN16PUE |
| ビット 7   |         |         |         |         |         |         | ビット 0   |

ビット **未実装:** '0' で読み込まれます

15-8

ビット **CNxPUE:** プルアップ有効化ビット

- 1 = プルアップ有効化  
0 = プルアップ無効化

凡例 :

R = 読み込み可能ビット	W = 書き込み可能ビット	U = 未実装、'0' が読み込まれます
-n = POR での値	'1' = ビットがセッ	'0' = ビットはクリ
	トされます	x = ビットは不定です

## 11.7 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正し、制限を設けて（必要な場合）使用可能です。現状、入出力ポートに関する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 #
ウェイクアップキーストロークの実装	AN552

**注：** dsPIC30F ファミリーデバイスに関しての、その他のアプリケーションノートやコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 11.8 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

この版は、dsPIC30F の入出力ポートに関する追加の技術情報を含みます。



**MICROCHIP**

## 第 12 章 . タイマー

### ハイライト

この章は以下の主要な項目を含んでいます。

12.1 序章 .....	12-2
12.2 タイマーのバリエーション .....	12-3
12.3 制御レジスタ .....	12-6
12.4 動作モード .....	12-9
12.5 タイマープリスケーラ .....	12-14
12.6 タイマー割り込み .....	12-14
12.7 16 ビットタイマーモジュールレジスタの読み込みおよび書き込み .....	12-15
12.8 低電力 32 kHz クリスタル発振器入力 .....	12-15
12.9 32 ビットタイマー構成 .....	12-16
12.10 32 ビットタイマーモード動作 .....	12-18
12.11 32 ビットタイマーへの読み込みおよび書き込み .....	12-21
12.12 省電力状態でのタイマー動作 .....	12-21
12.13 タイマーモジュール使用周辺装置 .....	12-22
12.14 設計の秘訣 .....	12-24
12.15 関連するアプリケーションノート .....	12-25
12.16 改訂履歴 .....	12-26

12

タイマー

## 12.1 序章

dsPIC30Fデバイスファミリーはバリエーションに応じて幾つかの16ビットタイマーを提供します。これらのタイマーはタイマー1、タイマー2、タイマー3、... 等として指定されます。

各タイマーモジュールは以下の読み込み可能/書き込み可能レジスタで構成される16ビットタイマー/カウンタです。

- TMRx: 16ビットタイマーカウンタレジスタ
- PRx: タイマーと関連する16ビット周期レジスタ
- TxCON: タイマーと関連する16ビット制御レジスタ

更に、各タイマーモジュールは割り込み制御用に関連ビットをもっています。

- 割り込み許可制御ビット (TxIE)
- 割り込みフラグステータスビット (TxIF)
- 割り込み優先順位制御ビット (TxIP<2:0>)

ある例外を除いて、全ての16ビットタイマーは同じ機能回路をもっています。16ビットタイマーは機能の違いを明らかにするため3つのタイプに分類されます。

- タイプAタイマ
- タイプBタイマ
- タイプCタイマ

ある16ビットタイマーは32ビットタイマーを形成するために組み合わせが可能です。

この章では周辺装置デバイスに関する専用タイマーについては説明しません。例えば、これにはモーターコントロールPWMモジュールおよび直交エンコーダインターフェース(QEI:直交符号化インターフェース)モジュールに関するタイマーを含みます。

## 12.2 タイマーのバリエーション

dsPIC30F デバイスで使用可能な全ての 16 ビットタイマーはある例外を除いて機能的に一致しています。16 ビットタイマーは 3 つの機能タイプに分類されます、タイプ A タイマー、タイプ B タイマー、そしてタイプ C タイマーです。

**注：** 利用可能なタイマーおよび各タイプについて詳しくはデバイスデータシートを参照してください。

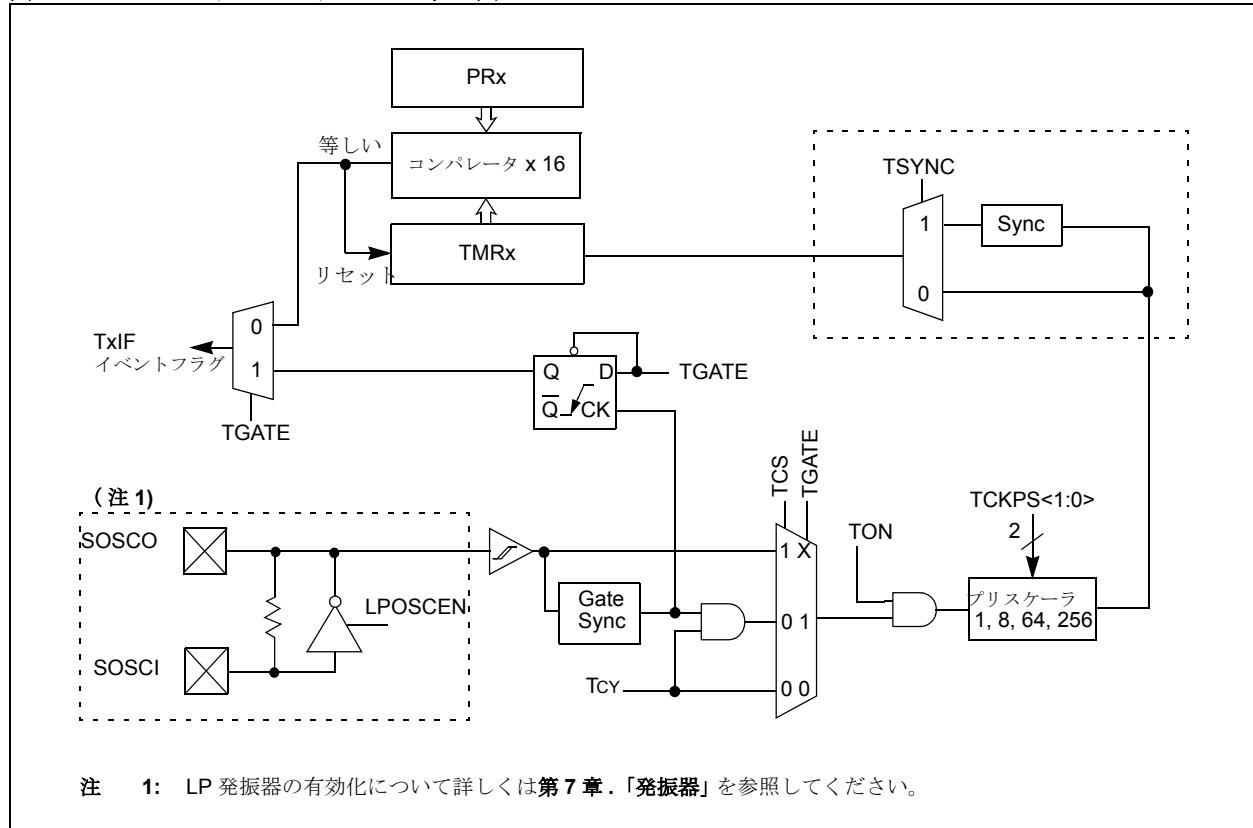
### 12.2.1 タイプ A タイマー

少なくとも 1 つのタイプ A タイマーがほとんどの dsPIC30F デバイスで使用可能です。ほとんどの dsPIC30F デバイスで、タイマー 1 がタイプ A タイマーです。タイプ A タイマーは他のタイプに対して以下の独自の機能をもっています。

- 低電力 32 kHz 発振器デバイスで動作可能。
- 外部クロックソースから非同期モードで動作可能

特にタイプ A タイマーの独自の機能はリアルタイムクロック (RTC) アプリケーション用に使用可能ことです。タイプ A タイマーのブロック図は図 12-1 にて表示されています。

図 12-1： タイプ A タイマーブロック図



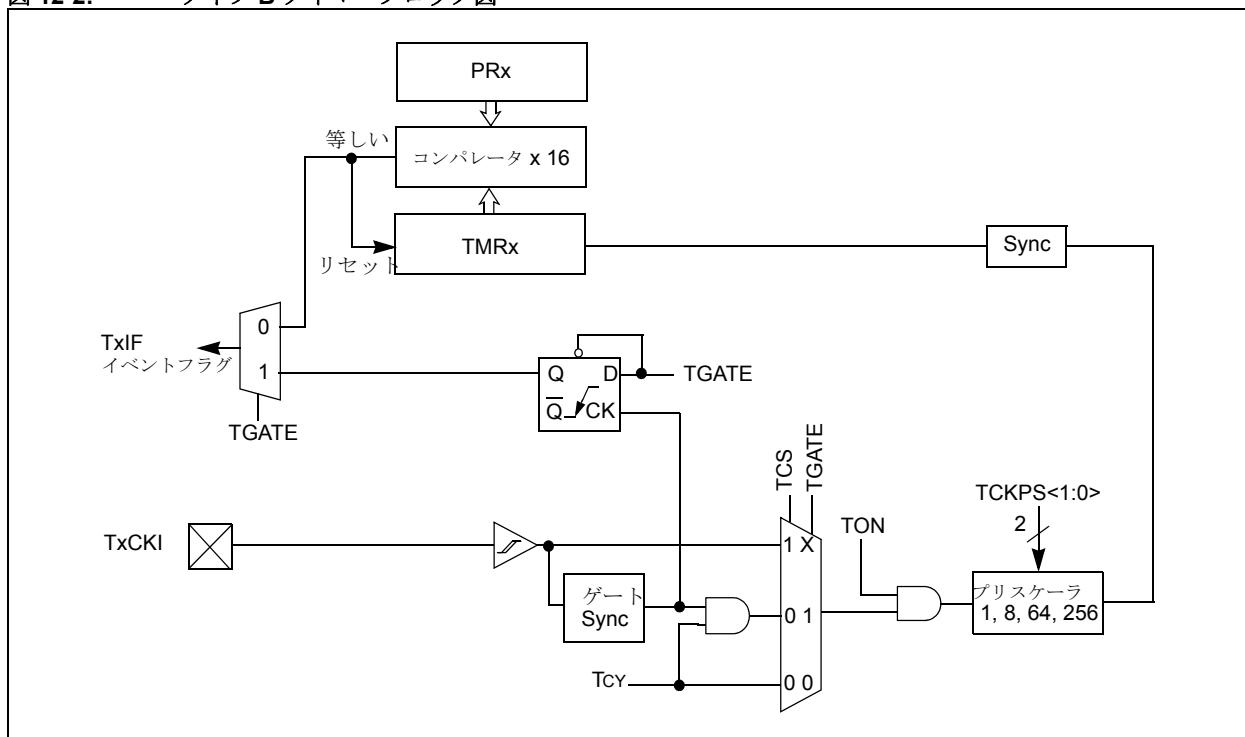
## 12.2.2 タイプ B タイマー

タイマー 2 およびタイマー 4 が存在する場合、ほとんどの dsPIC30F デバイスにてタイプ B タイマーとなります。タイプ B タイマーは他のタイマーに対して以下の独自の機能をもっています。

- ・ タイプ B タイマーは 32 ビットタイマーを形成するためタイプ C タイマーに連結することができます。タイプ B タイマーの TxCON レジスタは 32 ビットタイマー機能を有効化するため T32 制御ビットをもっています。
- ・ タイプ B タイマーのクロック同期はプリスケールロジックの後実行されます。

タイプ B タイマーのブロック図は図 12-2 に表示されています。

図 12-2: タイプ B タイマーブロック図



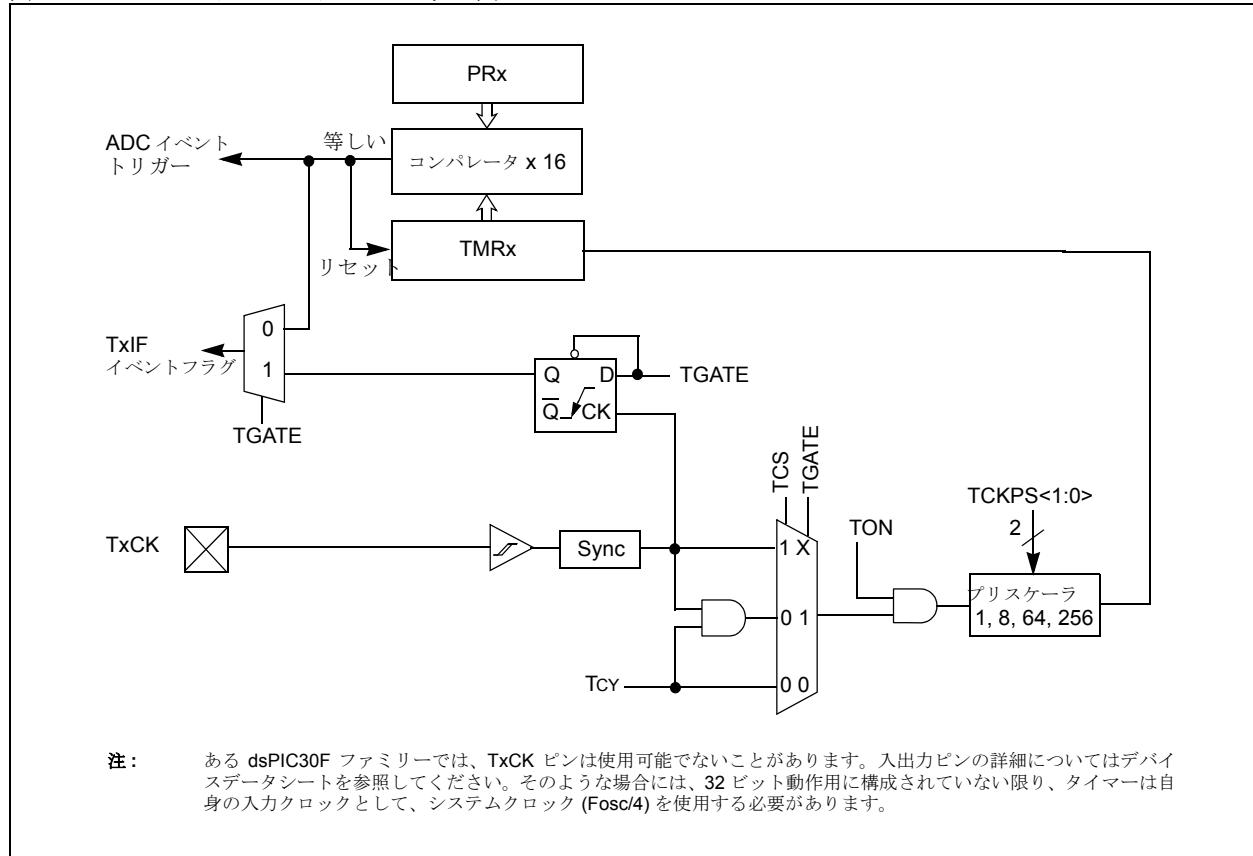
### 12.2.3 タイプ C タイマー

タイマー 3 およびタイマー 5 はほとんどの dsPIC30F デバイスでタイプ C タイマーです。タイプ C タイマーは他のタイプのタイマーに対して以下の独自の機能をもっています。

- ・ タイプ C タイマーは 32 ビットタイマーを形成するため、タイプ B タイマーと連結が可能です。
- ・ あるデバイスでは、少なくとも 1 つのタイプ C タイマーが A/D 変換をトリガーする機能をもっています。

タイプ C タイマーのブロック図は図 12-3 にて表示されています。

図 12-3: タイプ C タイマー ブロック図



## 12.3 制御レジスタ

レジスタ 12-1: TxCON: タイプ A タイマ制御レジスタ

上位バイト:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
ビット 15							ビット 8

下位バイト:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>	—	—	TSYNC	TCS	—
ビット 7							ビット 0

ビット 15 **TON:** タイマー ON 制御ビット

1 = タイマー開始  
0 = タイマー停止

ビット 14 未実装: ‘0’ で読み込まれます

ビット 13 **TSIDL:** IDLE モードでの停止ビット  
1 = デバイスが IDLE モードに入るとタイマー動作を停止  
0 = IDLE モードでタイマー動作の継続

ビット 12-7 未実装: ‘0’ で読み込まれます

ビット 6 **TGATE:** ゲート制御タイム累算有効化ビット  
1 = ゲートタイム累算有効化  
0 = ゲートタイム累算無効化  
(TCS は TGATE = 1 の場合 ‘0’ に設定する必要有り。 TCS = 1 の場合 ‘0’ で読み出す)

ビット 5-4 **TCKPS<1:0>:** 入力クロックプリスケール選択ビット  
11 = 1:256 プリスケール値  
10 = 1:64 プリスケール値  
01 = 1:8 プリスケール値  
00 = 1:1 プリスケール値

ビット 3 未実装: ‘0’ で読み込まれます

ビット 2 **TSYNC:** 外部クロック入力同期化選択ビット  
TCS = 1 の場合:  
1 = 外部クロック入力同期化  
0 = 外部クロック入力同期化なし  
TCS = 0 の場合:

このビットは無視されます。‘0’ で読み込まれます。タイマー 1 は TCS = 0 の場合内部クロックを使用。

ビット 1 **TCS:** クロックソース選択ビット  
1 = ピン TxCK からの外部クロック  
0 = 内部クロック (Fosc/4)

ビット 0 未実装: ‘0’ で読み込まれます

凡例:

R = 読み込み可能ビット      W = 書き込み可能ビット      U = 未実装、‘0’ が読み込まれます  
ト

-n = POR での値      ‘1’ = ビットがセット      ‘0’ = ビットはクリア      x = ビットは不定です  
されます      されます

## レジスタ 12-2: TxCON: タイプ B タイマ制御レジスタ

上位バイト:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
ビット 15	ビット 8						

下位バイト:							
U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>	T32	—	TCS	—	—
ビット 7	ビット 0						

ビット 15 **TON:** タイマー ON ビットT32 の場合 = 1 (32 ビットタイマーモード) :

1 = 32 ビット TMRx:TMRy タイマーペアの開始

0 = 32 ビット TMRx:TMRy タイマーペアの停止

T32 の場合 = 0 (16 ビットタイマーモード) :

1 = 16 ビットタイマーの開始

0 = 16 ビットタイマーの停止

ビット 14 未実装: '0' で読み込まれます

ビット 13 **TSIDL:** IDLE モードでの停止ビット

1 = IDLE モードにデバイスが入るとタイマー動作を停止

0 = IDLE モードでタイマー動作の継続

ビット 12-7 未実装: '0' で読み込まれます

ビット 6 **TGATE:** ゲート制御タイム累算有効化ビット

1 = ゲートタイム累算有効化

0 = ゲートタイム累算無効化

(TCS は TGATE = 1 の場合論理「0」に設定する必要有り)

ビット 5-4 **TCKPS<1:0>:** 入力クロックプリスケール選択ビット

11 = 1:256 プリスケール値

10 = 1:64 プリスケール値

01 = 1:8 プリスケール値

00 = 1:1 プリスケール値

ビット 3 **T32:** 32 ビットタイマーモード選択ビット

1 = TMRx+TMRy の 32 ビットタイマーモード

0 = TMRx、TMRy 独立の 16 ビットタイマーモード

ビット 2 未実装: '0' で読み込まれます

ビット 1 **TCS:** クロックソース選択ビット

1 = ピン TxCK からの外部クロック

0 = 内部クロック (Fosc/4)

ビット 0 未実装: '0' で読み込まれます

凡例:

R = 読み込み可能ビット

W = 書き込み可能

U = 未実装、'0' が読み込まれます

ビット

-n = POR での値

'1' = ビットがセット

'0' = ビットはクリア

x = ビットは不定です

されます

されます

## レジスタ 12-3: TxCON: タイプ C タイマー制御レジスタ

上位バイト :

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
ビット 15							ビット 8

下位バイト :

U-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		—	—	TCS	—
ビット 7							ビット 0

ビット 15 **TON:** タイマー ON ビット  
1 = 16 ビット TMRx の開始  
0 = 16 ビット TMRx の停止

ビット 14 未実装: ‘0’ で読み込まれます

ビット 13 **TSIDL:** IDLE モードでの停止ビット  
1 = IDLE モードにデバイスが入るとモジュール動作を停止  
0 = IDLE モードでモジュール動作の継続

ビット 12-7 未実装: ‘0’ で読み込まれます

ビット 6 **TGATE:** ゲートタイム累算有効化ビット  
1 = ゲートタイム累算有効化  
0 = ゲートタイム累算無効化 (TCS = 1 の場合「0」で読み出す)  
(TCS は TGATE = 1 の場合論理「0」に設定する必要有り)

ビット 5-4 **TCKPS<1:0>:** 入力クロックプリスケール選択ビット  
11 = 1:256 プリスケール値  
10 = 1:64 プリスケール値  
01 = 1:8 プリスケール値  
00 = 1:1 プリスケール値

ビット 3-2 未実装: ‘0’ で読み込まれます

ビット 1 **TCS:** クロックソース選択ビット  
1 = ピン TxCK からの外部クロック  
0 = 内部クロック (Fosc/4)

ビット 0 未実装: ‘0’ で読み込まれます

凡例 :

R = 読み込み可能ビット

W = 書き込み可能

U = 未実装、‘0’ が読み込まれます

ビット

-n = POR での値

‘1’ = ビットがセット

‘0’ = ビットはクリア x = ビットは不定です

されます

## 12.4 動作モード

各タイマーモジュールは以下のモードのうちの 1 つで動作可能です。

- 同期タイマーとして
- 同期カウンタとして
- ゲートタイマーとして
- 非同期カウンタとして (タイプ A タイマーのみ)

タイマーモードは以下のビットで決定されます。

- TCS (TxCON<1>): タイマークロックソース制御ビット
- TSYNC (T1CON<2>): タイマー同期制御ビット (タイプ A タイマーのみ)
- TGATE (TxCON<6>): タイマーゲート制御ビット

各タイマーモジュールは TON 制御ビット (TxCON <15>) を使用して有効化または無効化されます。

**注:** タイプ A タイマーのみ外部非同期クロックモードをサポートします。

### 12.4.1 タイマーモード

タイマーの全てのタイプはタイマーモードの機能をもっています。タイマーモードでは、タイマーへの入力クロックが内部システムクロック ( $F_{osc}/4$ ) から提供されます。有効化された場合、タイマーは 1:1 プリスケーラ設定の場合、命令サイクルごとに +1 します。タイマーモードは TCS 制御ビット (TxCON<1>) をクリアすることで選択されます。同期モード制御ビット、TSYNC (T1CON<2>) はシステムクロックソースがタイマークロックの生成に使用されるため、影響がありません。

#### 例 12-1: システムクロック使用の 16 ビットタイマー初期化コード

```

; The following code example will enable Timer1 interrupts,
; load the Timer1 Period register and start Timer1.

; When a Timer1 period match interrupt occurs, the interrupt
; service routine must clear the Timer1 interrupt status flag
; in software.

CLR  T1CON           ; Stops the Timer1 and reset control reg.
CLR  TMR1            ; Clear contents of the timer register
MOV  #0xFFFF, w0      ; Load the Period register
MOV  w0, PR1          ; with the value 0xFFFF

BSET  IPC0, #T1IP0   ; Setup Timer1 interrupt for
BCLR  IPC0, #T1IP1   ; desired priority level
BCLR  IPC0, #T1IP2   ; (this example assigns level 1 priority)
BCLR  IFS0, #T1IF    ; Clear the Timer1 interrupt status flag
BSET  IEC0, #T1IE    ; Enable Timer1 interrupts
BSET  T1CON, #TON    ; Start Timer1 with prescaler settings
                     ; at 1:1 and clock source set to
                     ; the internal instruction cycle

; Example code for Timer1 ISR

__T1Interrupt:
BCLR  IFS0, #T1IF    ; Reset Timer1 interrupt flag
                     ; User code goes here.
RETFIE              ; Return from ISR

```

## 12.4.2 外部クロック入力使用同期カウンタモード

TCS 制御ビット (TxCON<1>) がセットされた場合、タイマー用のクロックソースは外部から提供され、選択されたタイマーは TxCK ピンからのクロック入力の立ち上がりエッジごとに増分します。

外部クロック同期化はタイプ A タイマーで有効化される必要があります。これは TSYNC 制御ビット (TxCON<2>) をセットすることで設定できます。タイプ B およびタイプ C タイマーでは、外部クロック入力は常にシステム命令サイクルクロックである T<sub>CY</sub> に同期化されます。

タイマーが同期カウンタモードで動作する場合、外部クロックの High と Low 区間の最小パルス幅制限があります。なぜならデバイス命令サイクルと外部クロックソースの同期化は命令サイクル内に 2 回別々に外部クロック信号をサンプリングすることで実現されるからです。

同期化回路が SLEEP モード中に遮断されるため、同期化された外部クロックソースで動作しているタイマーは SLEEP モードでは動作しません。

**注:** 外部入力クロックは、同期カウンタモードでタイマー x が使用された場合、ある High と Low 区間の最小パルス幅制限を満たす必要があります。詳細についてはデータシートの「電気的仕様」の章を参照してください。

例 12-2: 外部クロック入力使用の 16 ビット同期カウンタモードの初期化コード

```

; The following code example will enable Timer1 interrupts, load the
; Timer1 Period register and start Timer1 using an external clock
; and a 1:8 prescaler setting.

; When a Timer1 period match interrupt occurs, the interrupt service
; routine must clear the Timer1 interrupt status flag in software.

CLR    T1CON           ; Stops the Timer1 and reset control reg.
CLR    TMR1            ; Clear contents of the timer register
MOV    #0x8CFF, w0      ; Load the Period register
MOV    w0, PR1          ; with the value 0x8CFF

BSET   IPC0, #T1IP0     ; Setup Timer1 interrupt for
BCLR   IPC0, #T1IP1     ; desired priority level
BCLR   IPC0, #T1IP2     ; (this example assigns level 1 priority)
BCLR   IFS0, #T1IF      ; Clear the Timer1 interrupt status flag
BSET   IEC0, #T1IE      ; Enable Timer1 interrupts
MOV    #0x8016, W0       ; Start Timer1 with prescaler settings at
                        ; 1:8 and clock source set to the external
                        ; clock in the synchronous mode

MOV    w0, T1CON         ; Example code for Timer1 ISR

__T1Interrupt:
BCLR   IFS0, #T1IF      ; Reset Timer1 interrupt flag
                        ; User code goes here.
RETFIE                         ; Return from ISR

```

### 12.4.3 外部クロック入力使用タイプ A タイマー非同期カウンタモード

タイプ A タイマーには TxCK ピンに接続された外部クロックソースを使用して、非同期 カウンタモードで動作できる機能があります。TSYNC 制御ビット (TxCON<2>) がクリアされた場合、外部クロック入力はデバイスシステムクロックソースとは同期をしません。タイマーは内部デバイスクロックと非同期に増分し続けます。

非同期動作タイマーは以下のアプリケーションにて効果的です。

- ・タイマーは SLEEP モード中に動作可能であり、周期レジスタと一致したときにウェイクアップさせる周期的な割り込みを生成可能です。
- ・タイマーはリアルタイムクロックアプリケーション用に低電力 32 kHz 発振器をクロック源にすることが可能です。

**注**

- 1:** タイプ A タイマーのみ非同期カウンタモードをサポートします。
- 2:** 外部入力クロックは、タイマー x が非同期カウンタモードで使用された場合、ある High と Low 区間の最小パルス幅制限を満たす必要があります。詳細についてはデバイスデータシートにある「電気的仕様」の章を参照してください。
- 3:** 非同期モードでタイマー 1, を読み出す場合、予想外の結果が発生する可能性があります。

#### 例 12-3: 外部クロック入力使用の 16 ビット非同期カウンタモード初期化コード

```

; The following code example will enable Timer1 interrupts, load the
; Timer1 Period register and start Timer1 using an asynchronous
; external clock and a 1:8 prescaler setting.

; When a Timer1 period match interrupt occurs, the interrupt service
; routine must clear the Timer1 interrupt status flag in software.

CLR    T1CON           ; Stops the Timer1 and reset control reg.
CLR    TMR1            ; Clear contents of the timer register
MOV    #0x7FFF, w0      ; Load the Period register
MOV    w0, PR1          ; with the value 0x7FFF

BSET   IPC0, #T1IP0     ; Setup Timer1 interrupt for
BCLR   IPC0, #T1IP1     ; desired priority level
BCLR   IPC0, #T1IP2     ; (this example assigns level 1 priority)
BCLR   IFS0, #T1IF      ; Clear the Timer1 interrupt status flag
BSET   IEC0, #T1IE      ; Enable Timer1 interrupts
MOV    #0x8012, w0      ; Start Timer1 with prescaler settings at
                       ; 1:8 and clock source set to the external
MOV    w0, T1CON         ; clock in the asynchronous mode

; Example code for Timer1 ISR

_T1Interrupt:
BCLR   IFS0, #T1IF      ; Reset Timer1 interrupt flag
                       ; User code goes here.
RETFIE                         ; Return from ISR

```

## 12.4.4 高速外部クロックソースによるタイマー動作

アプリケーションによって、外部クロックが比較的高い周波数の場合、これをカウントするために、タイマーのうち 1 つを使用するのが望ましい場合があります。このような場合、タイマーのクロック同期化論理がタイマープリスケーラの後に配置されているため（図 12-1 および図 12-2 を参照）、外部クロックソースをカウントするにはタイプ A およびタイプ B タイマーが最適な選択です。これにより、プリスケーラにより要求される最小のパルス幅が確保されれば、高い周波数の外部クロック周波数が使用可能になります。タイマープリスケーラ率が 1:1 以外がタイプ A またはタイプ B タイマーに選択された場合、外部クロック入力用の High と Low 区間の最小パルス幅制限は選択されたプリスケーラ率により低減されます。

タイプ A タイマーはプリスケーラのタイミング要求事項が不要な非同期クロックモードで動作可能な特徴を有しています。

全ての場合で、外部クロック信号が超えられない High と Low 区間の最小パルス幅制限があります。これらの最小限時間は入出力ピンの要求事項を満たすための必須事項です。

タイマーに関連する外部クロックタイミング仕様について詳しくはデバイスデータシートを参照してください。

## 12.4.5 ゲートタイム累算モード

ゲートタイム累算モードは TxCK ピンの入力の High 継続時間に基づいて増分するよう内部タイマーレジスタを許可します。ゲートタイム累算モードでは、タイマークロックソースは内部システムクロックから得られます。TxCK ピン状態が High の場合、タイマーレジスタは周期レジスタと一致が発生するまで、または TxCK ピン状態が Low へと変化するまでカウントします。High から Low へのピンの状態遷移は TxIF 割り込みフラグをセットします。エッジの発生により、TxCK ピンの信号の立ち下りエッジの後、1 または 2 つの命令サイクル後に割り込みフラグが確立します。

TGATE 制御ビット (TxCON<6>) はゲートタイム累算モードを有効化するためセットされる必要があります。タイマーは有効化し、TON (TxCON<15>) = 1、そしてタイマークロックソースは内部クロック TCS (TxCON<1>) = 0 に設定します。

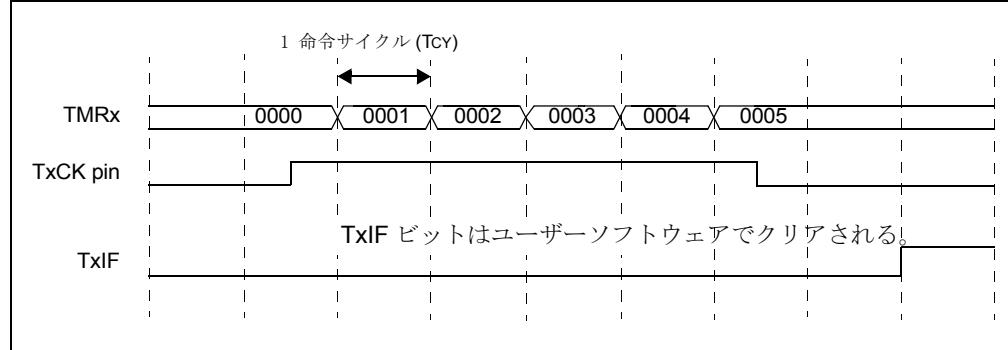
ゲート動作は TxCK ピンに入力された信号の立ち上がりエッジ時で開始し、TxCK ピンに入力された信号の立ち下りエッジ時で停止します。各タイマーは外部ゲート信号が High の間増分します。

ゲート信号の立ち下りエッジはカウント動作を停止しますが、タイマーをリセットはしません。ユーザーは次の立ち上がりエッジゲート入力の際にゼロから開始したい場合はタイマーをリセットする必要があります。

**注：** タイマーはゲートタイム累算モードでタイマー周期レジスター一致が発生した場合、CPU に割り込みません。

タイマーカウントの分解能は直接タイマークロック周期に関係しています。1:1 のタイマープリスケーラに対して、タイマークロック周期は 1 つの命令サイクルです。1:256 のタイマープリスケーラに対して、タイマークロック周期は命令サイクルの 256 倍です。タイマークロック分解能はゲート信号のパルス幅に依存します。ゲート幅パルスの要求事項の詳細はデバイスデータシートにある「電気的仕様」の章を参照してください。

図 12-4: ゲートタイム累算モード動作



例 12-4: 16 ビットゲートタイム累算モードの初期化コード

```

; The following code example will enable Timer2 interrupts, load the
; Timer2 Period register and start Timer2 using an internal clock
; and an external gate signal. On the falling edge of the gate
; signal a Timer2 interrupt occurs. The interrupt service
; routine must clear the Timer2 interrupt status flag in software .

CLR    T2CON          ; Stops the Timer2 and reset control reg.
CLR    TMR2            ; Clear contents of the timer register
MOV    #0xFFFF, w0      ; Load the Period register with
MOV    w0, PR2          ; the value 0xFFFF

BSET   IPC1, #T2IP0    ; Setup Timer2 interrupt for
BCLR   IPC1, #T2IP1    ; desired priority level
BCLR   IPC1, #T2IP2    ; (this example assigns level 1 priority)
BCLR   IFS0, #T2IF     ; Clear the Timer2 interrupt status flag
BSET   IEC0, #T2IE     ; Enable Timer2 interrupts
BSET   T2CON, #TGATE   ; Set up Timer2 for operation in Gated
                       ; Time Accumulation mode
BSET   T2CON, #TON     ; Start Timer2

; Example code for Timer2 ISR

__T2Interrupt:
    BCLR   IFS0, #T2IF    ; Reset Timer2 interrupt flag
                           ; User code goes here.
    RETFIE                  ; Return from ISR

```

## 12.5 タイマープリスケーラ

入力クロック ( $F_{osc}/4$  または外部クロック) と全ての 16 ビットタイマーの間には 1:1、1:8、1:64 および 1:256 のプリスケールオプションがあります。クロックプリスケーラは TCKPS<1:0> 制御ビット (TxCON<5:4>) を使用して選択されます。プリスケーラカウンタは以下のどの項目が発生する場合でもクリアされます。

- TMRx レジスタへの書き込み
- TON (TxCON<15>) を「0」へクリア
- デバイス RESET

**注:** TMRx レジスタは TxCON への書き込みではクリアされません。

## 12.6 タイマ割り込み

16 ビットタイマーは動作モードにより、周期レジスタとの一致の割り込み生成または外部ゲート信号の立ち下りエッジの割り込み生成機能があります。

TxIF ビットは以下の条件のうち 1 つが真の場合セットされます。

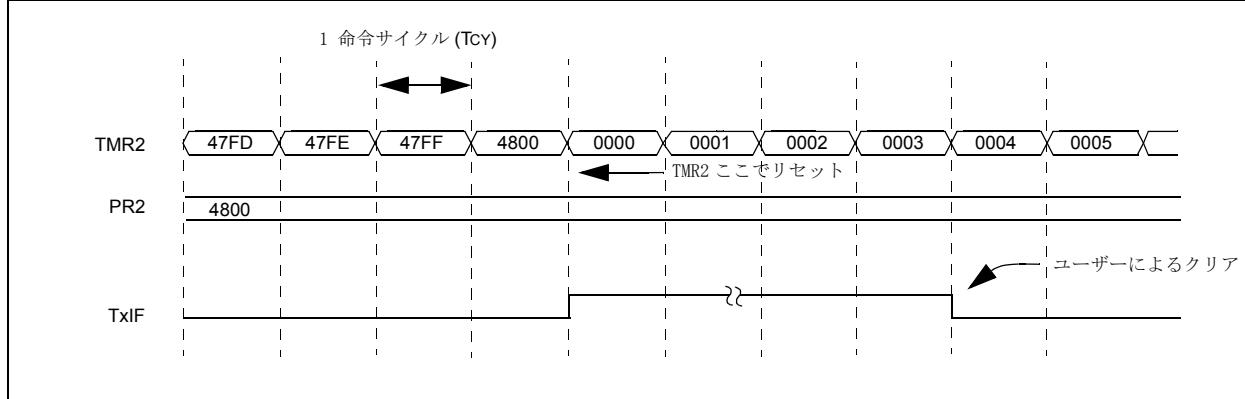
- タイマーカウントが各周期レジスタに一致し、かつタイマーモジュールがゲートタイム累算モードで動作していない。
- 「ゲート」信号の立ち下りエッジがゲートタイム累算モード時に検出される。

TxIF ビットはソフトウェアでクリアされる必要があります。

タイマーは各タイマ割り込み有効ビット TxIE を介して割り込みの要因として有効化されます。更に、割り込み優先度ビット (TxIP<2:0>) はタイマーが割り込み要因となるためにゼロ以外の値で書き込まれる必要があります。詳しくは第 6 章、「割り込み」を参照してください。

**注:** 周期レジスタが 0x0000 でロードされ、タイマーが有効化された場合特例が発生します。この構成ではタイマ割り込みは生成されません。

図 12-5: タイマ周期レジスター一致用割り込みタイミング



## 12.7 16 ビットタイマーモジュールレジスタの読み込みおよび書き込み

- 全てのタイマーモジュール **SFR** はバイト (8 ビット) としてまたはワード (16 ビット) として書き込みが可能です。
- 全てのタイマーモジュール **SFR** はワード (16 ビット) としてのみ読み込み可能です。

### 12.7.1 16 ビットタイマーへの書き込み

タイマーおよびその各周期レジスタはモジュール動作中に書き込みが可能です。ユーザーはバイト書き込みが行われる場合以下の項目に注意が必要です。

- タイマーが増分している、およびタイマーの下位バイトが書き込まれる場合、タイマーの上位バイトは影響を受けません。**0xFF** がタイマーの下位バイトへ書き込まれる場合、この書き込みの後の次のタイマーカウントクロックは下位バイトを **0x00** へ替えタイマーの上位バイトへと桁上げを生成します。
- タイマーが増分している、およびタイマーの上位バイトが書き込まれる場合、タイマーの下位バイトは影響を受けません。書き込みが発生する際にタイマーの下位バイトが **0xFF** の場合、次のタイマーカウントクロックはタイマーワードから桁上げを生成し、この桁上げはタイマーの上位バイトを増分させます。

**TMRx** レジスタが命令で (ワードまたはバイト) 書き込まれる場合、**TMRx** レジスタ増分はマスクされ、命令サイクル中は発生しません。

非同期クロックソースで動作しているタイマーへの書き込みはリアルタイムアプリケーションでは避ける必要があります。詳細は [セクション 12.4.1「タイマーモード」](#) を参照してください。

### 12.7.2 16 ビットタイマーからの読み込み

タイマーの読み込みおよび関連 **SFR** の全てはワードの読み込みである必要があります (16 ビット)。バイトでの読み込みは効果がありません ('0' が返されます)。

タイマーおよび各周期レジスタはモジュールが動作している間読み込みが可能です。同じ命令サイクル中に **TMRx** レジスタの読み込みによってタイマーの増分を防ぐことはできません。

## 12.8 低電力 32 kHz クリスタル発振器入力

各デバイスの、リアルタイムクロック (RTC) アプリケーションで、低電力 32 kHz クリスタル発振器がタイプ A タイマーモジュールで使用可能です。

- LP 発振器は LP 発振器が有効化され、タイマーが外部クロックソースを使用するよう設定された場合、タイマーのクロックソースとなります。
- LP 発振器は OSCCON レジスタの LPOSCEN 制御ビットをセットすることで有効化されます。
- 32 kHz クリスタルは SOSCO/SOSCI デバイスピンへ接続します。

詳細は [第 7 章. 「発振器」](#) を参照してください。

## 12.9 32 ビットタイマー構成

32 ビットタイマーモジュールはタイプ B およびタイプ C の 16 ビットタイマーモジュールを組み合わせることで構成可能です。タイプ C タイマーは結合したタイマーの MSWord、そしてタイプ B タイマーは LSWord です。

32 ビット動作が構成された場合、タイプ B タイマーの制御ビットが 32 ビットタイマーの動作を制御します。タイプ C タイマーの TxCON レジスタ制御ビットには効果がありません。

割り込み制御に関しては、結合した 32 ビットタイマーはタイプ C タイマーの割り込み有効化、割り込みフラグ、および割り込み優先順位制御ビットを使用します。タイプ B タイマーの割り込み制御およびステータスビットは 32 ビットタイマー動作中には使用しません。

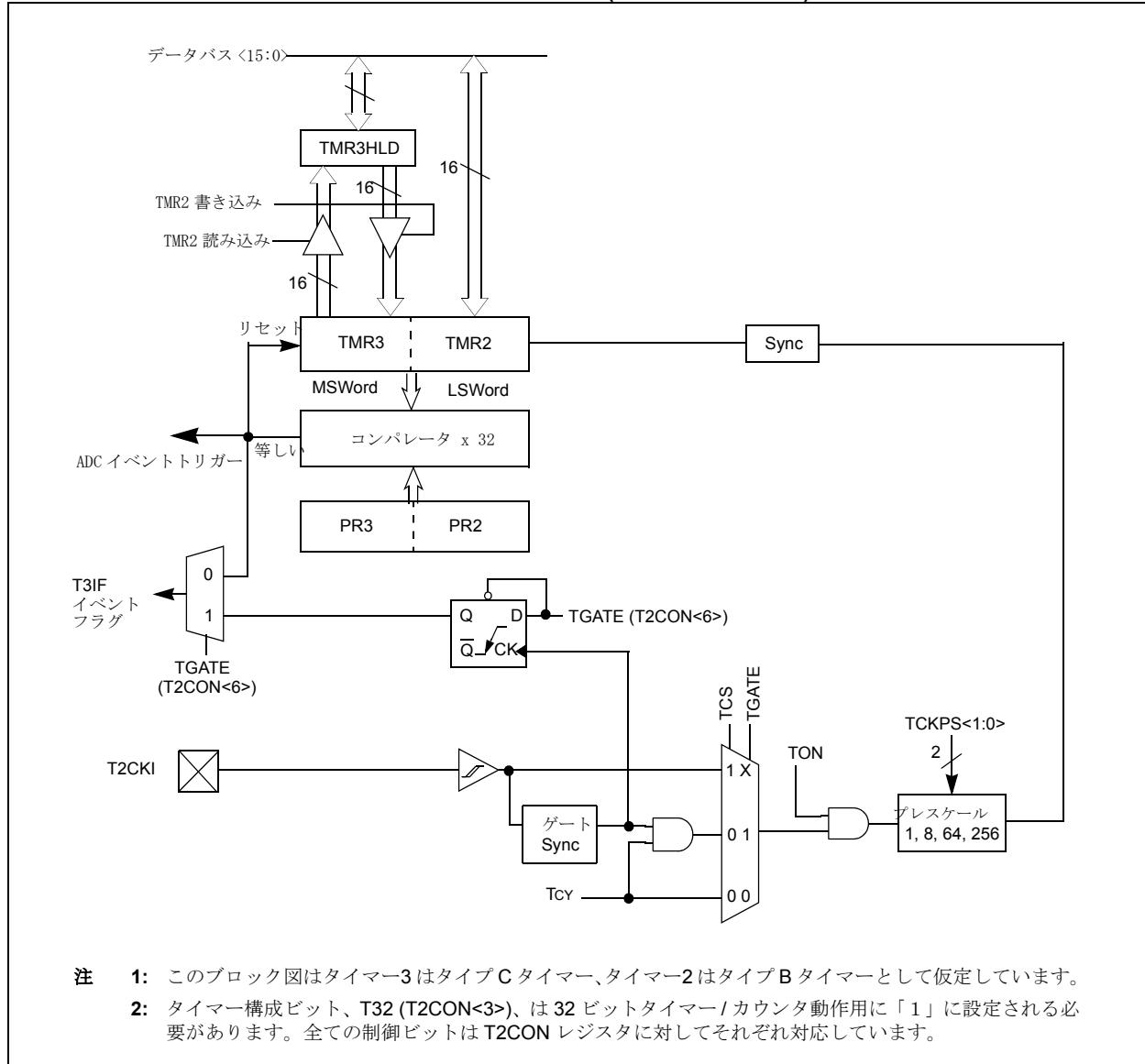
**注：** 結合可能なタイプ B およびタイプ C タイマーの詳細についてはデバイスデータシートを参照してください。

以下の構成設定はタイマー3 がタイプ C タイマーおよびタイマー2 はタイプ B タイマーと仮定しています。

- TON (T2CON<15>) = 1
- T32 (T2CON<3>) = 1
- TCKPS<1:0> (T2CON<5:4>) はタイマー 2 用のプリスケールモードの設定に使用 ( タイプ B タイマー )。
- TMR3:TMR2 レジスタペアはタイマーモジュールの 32 ビット値となります。したがって、TMR2 ( タイプ B タイマー ) レジスタが 32 ビットタイマー値の下位ワードで、TMR3 ( タイプ C タイマー ) レジスタは上位ワードとなります。
- PR3:PR2 レジスタペアは TMR3:TMR2 タイマー値との比較に使用される 32 ビット周期タイマーとなります。
- T3IE (IEC0<7>) この 32 ビットタイマー割り込みの有効化に使用。
- T3IF (IFS0<7>) はタイマー割り込み用ステータスフラグとして使用。
- T3IP<2:0> (IPC1<14:12>) は 32 ビットタイマーの割り込み優先度を設定。
- T3CON<15:0> は「無関係」ビット。

図 12-6 のブロック図はタイマー2 およびタイマー3 を 32 ビットタイマーモジュールとして使用した例を示します。

図 12-6: タイプB-タイプCタイマーペアブロック図(32ビットタイマー)



## 12.10 32 ビットタイマーモード動作

## 12.10.1 タイマーモード

例 12-5 では 32 ビットタイマーのタイマーモードでの構成方法を示しています。この例ではタイマー2はタイプBタイマーおよびタイマー3はタイプCタイマーとして仮定しています。32ビットタイマー動作に関しては、T32 制御ビットは T2CON レジスタ（タイプBタイマー）に設定される必要があります。タイマー2およびタイマー3が32ビットタイマーに設定される場合、T3CON 制御ビットは無視されます。T2CON 制御ビットのみがセットアップおよび制御に必要とされます。タイマー2クロックおよびゲート入力が32ビットタイマーモジュールに使用されますが、T3IF フラグを使用して割り込みが生成されます。タイマー2は32ビットタイマーの LSWord、そしてタイマー3は32ビットタイマーの MSWord です。TMR3 は TMR2 からのオーバーフロー（キャリーアウト）によって増分されます。32ビットタイマーは PR2 および PR3 によって形成された32ビット周期レジスタへにプレロードされた一致値まで増分します。一致したらまた最初からカウントを続けます。最長の32ビットタイマーカウントとするには、0xFFFFFFF 値を PR3:PR2 にロードします。有効にされた場合、割り込みが周期一致で生成されます。

例 12-5: 命令サイクルを入力クロックとして使用 32 ビットタイマー 初期化コード

```

; The following code example will enable Timer3 interrupts, load the
; Timer3:Timer2 Period Register and start the 32-bit timer module
; consisting of Timer3 and Timer2.

; When a 32-bit period match interrupt occurs, the user must clear
; the Timer3 interrupt status flag in software.

CLR    T2CON           ; Stops any 16/32-bit Timer2 operation
CLR    T3CON           ; Stops any 16-bit Timer3 operation
CLR    TMR3            ; Clear contents of the Timer3 timer register
CLR    TMR2            ; Clear contents of the Timer2 timer register
MOV    #0xFFFF, w0      ; Load the Period Register 3
MOV    w0, PR3          ; with the value 0xFFFF
MOV    w0, PR2          ; Load the Period Register2 with value 0xFFFF

BSET   IPC1, #T3IP0     ; Setup Timer3 interrupt for
BCLR   IPC1, #T3IP1     ; desired priority level
BCLR   IPC1, #T3IP2     ; (this example assigns level 1 priority)
BCLR   IFS0, #T3IF      ; Clear the Timer3 interrupt status flag
BSET   IEC0, #T3IE      ; Enable Timer3 interrupts
BSET   T2CON, #T32       ; Enable 32-bit Timer operation
BSET   T2CON, #TON       ; Start 32-bit timer with prescaler
                         ; settings at 1:1 and clock source set to
                         ; the internal instruction cycle

; Example code for Timer3 ISR

__T3Interrupt:
    BCLR   IFS0, #T3IF      ; Reset Timer3 interrupt flag
                           ; User code goes here.
    RETFIE                    ; Return from ISR

```

### 12.10.2 同期カウンタモード

32 ビットタイマーは同期カウンタモードでは 16 ビットタイマーと同様に動作します。例 12-6 では同期カウンタモードでの 32 ビットタイマーの構成の仕方を示しています。この例ではタイマー 2 はタイプ B タイマー、およびタイマー 3 はタイプ C タイマーとして想定しています。

**例 12-6:** 外部クロック入力使用 32 ビット同期カウンタモードの初期化コード

```

; The following code example will enable Timer2 interrupts, load
; the Timer3:Timer2 Period register and start the 32-bit timer
; module consisting of Timer3 and Timer2.

; When a 32-bit period match interrupt occurs, the user must clear
; the Timer3 interrupt status flag in the software.

CLR    T2CON           ; Stops any 16/32-bit Timer2 operation
CLR    T3CON           ; Stops any 16-bit Timer3 operation
CLR    TMR3            ; Clear contents of the Timer3 timer register
CLR    TMR2            ; Clear contents of the Timer2 timer register
MOV    #0xFFFF, w0      ; Load the Period Register3
MOV    w0, PR3          ; with the value 0xFFFF
MOV    w0, PR2          ; Load the Period Register2 with value 0xFFFF

BSET   IPC1, #T3IP0    ; Setup Timer3 interrupt for
BCLR   IPC1, #T3IP1    ; desired priority level
BCLR   IPC1, #T3IP2    ; (this example assigns level 1 priority)
BCLR   IFS0, #T3IF     ; Clear the Timer3 interrupt status flag
BSET   IEC0, #T3IE     ; Enable Timer3 interrupts
MOV    #0x801A, w0      ; Enable 32-bit Timer operation and start
MOV    w0, T2CON         ; 32-bit timer with prescaler settings at
                        ; 1:8 and clock source set to external clock

; Example code for Timer3 ISR

_T3Interrupt:
BCLR   IFS0, #T3IF     ; Reset Timer3 interrupt flag
                        ; User code goes here.
RETFIE                         ; Return from ISR

```

## 12.10.3 非同期カウンタモード

タイプBおよびタイプCタイマーは非同期外部クロックモードをサポートしません。したがって32ビット非同期カウンタモードはサポートされていません。

## 12.10.4 ゲートタイム累算モード

32ビットタイマーはゲートタイム累算モードで16ビットタイマーのように動作します。例12-7は、32ビットタイマーのゲートタイム累算モードでの構成の仕方を示しています。この例ではタイマー2がタイプBタイマー、およびタイマー3はタイプCタイマーであると仮定します。

例12-7: 32ビットのゲートタイム累算モードの初期化コード

```
; The following code example will enable Timer2 interrupts, load the
; Timer3:Timer2 Period register and start the 32-bit timer module
; consisting of Timer3 and Timer2. When a 32-bit period match occurs
; the timer will simply roll over and continue counting.

; However, when at the falling edge of the Gate signal on T2CK
; an interrupt is generated, if enabled. The user must clear the
; Timer3 interrupt status flag in the software.

CLR    T2CON           ; Stops any 16/32-bit Timer2 operation
CLR    T3CON           ; Stops any 16-bit Timer3 operation
CLR    TMR3            ; Clear contents of the Timer3 register
CLR    TMR2            ; Clear contents of the Timer2 register
MOV    #0xFFFF, w0      ; Load the Period Register3
MOV    w0, PR3          ; with the value 0xFFFF
MOV    w0, PR2          ; Load the Period Register2 with value 0xFFFF

BSET   IPC1,    #T3IP0  ; Setup Timer3 interrupt for
BCLR   IPC1,    #T3IP1  ; desired priority level
BCLR   IPC1,    #T3IP2  ; (this example assigns level 1 priority)
BCLR   IFS0,    #T3IF   ; Clear the Timer3 interrupt status flag
BSET   IEC0,    #T3IE   ; Enable Timer3 interrupts
MOV    #0x804C, w0      ; Enable 32-bit Timer operation and
MOV    w0, T2CON         ; Start 32-bit timer in gated time
                           ; accumulation mode.

; Example code for Timer3 ISR

__T3Interrupt:
BCLR   IFS0,    #T3IF   ; Reset Timer3 interrupt flag
                           ; User code goes here.
RETFIE
```

## 12.11 32 ビットタイマーへの読み込みおよび書き込み

32 ビット読み込み/書き込み動作のとき 32 ビットタイマーの LSWord および MSWord の間で同期化するようにするために、追加の制御論理とレジスタが使用されます（図 12-6 参照）。各タイプ C タイマーは TMRxHLD と呼ばれ、タイマーレジスタペアへ読み込みまたは書き込みする場合に使用されます。TMRxHLD レジスタは各タイマーが 32 ビット動作用に構成される場合にのみ使用されます。

TMR3:TMR2 が 32 ビットタイマーペアを形成すると仮定した場合、ユーザーは最初に TMR2 レジスタからタイマー値の LSWord を読み込みます。LSWord の読み込みのとき、自動的に TMR3 のコンテンツを TMR3HLD レジスタへ転送します。それからユーザーはタイマー値の MSWord を得るために TMR3HLD を読み込みます。これは以下の例で示されています。

### 例 12-8: 32 ビットタイマーからの読み込み

```
; The following code segment reads the 32-bit timer formed by the
; Timer3-Timer2 pair into the registers W1(MS Word) and W0(LS Word).

MOV TMR2, W0      ;Transfer the LSW into W1
MOV TMR3HLD, W1   ;Transfer the MSW from the holding register to W0
```

TMR3:TMR2 レジスタペアへ値を書き込むためには、ユーザーは最初に TMR3HLD レジスタへ MSWord を書き込みます。タイマー値の LSWord が TMR2 へ書き込まれるとき同時に、TMR3HLD の内容が自動的に TMR3 レジスタへ転送されます。

## 12.12 省電力状態でのタイマー動作

### 12.12.1 SLEEP モードでのタイマー動作

デバイスが SLEEP モードに入る場合、システムクロックは無効化されます。タイマーモジュールが内部クロックソース ( $F_{osc}/4$ ) で動作している場合、それも無効化されます。

タイプ A タイマーは、外部クロックソースから非同期で動作可能なので、他のタイマーモジュールとは異なります。この違いのため、タイプ A タイマーモジュールは SLEEP モード中も動作を継続可能です。SLEEP モードで動作するには、タイプ A タイマーは以下のように設定される必要があります。

- Timer1 モジュールは有効化される、 $TON = 1(T1CON<15>)$  かつ
- Timer1 クロックソースは外部として選択される、 $TCS = 1(T1CON<1> = 1)$  さらに
- TSYNC ビット ( $T1CON<2>$ ) が論理「0」として設定される（非同期カウンタモード有効化）。

**注:** 非同期カウンタ動作はタイマー 1 モジュールでのみサポートされています。

上記の条件が全て満たされた場合、タイマー 1 はデバイスが SLEEP モード中にも周期一致のカウント及び検出を継続します。タイマーおよび周期レジスタの間で一致が発生した場合、TxIF ビットがセットされ、任意でデバイスを SLEEP からウェイクアップするよう割り込みが生成されます。詳しくは、第 10 章。「ウォッチドッグタイマーおよび省電力モード」を参照してください。

### 12.12.2 IDLE モードでのタイマー動作

デバイスが IDLE モードに入る場合、システムクロックソースは機能を継続しますが、CPU はコード実行を停止します。タイマーモジュールは指定すれば IDLE モードで動作を継続します。

TSIDL ビット ( $TxCON<13>$ ) はタイマーモジュールが IDLE モードで停止するか、または正常に動作を継続するかどうか選択します。 $TSIDL = 0$  の場合、モジュールは IDLE モードで動作を継続します。 $TSIDL = 1$  の場合、モジュールは IDLE モードで停止します。

## 12.13 タイマーモジュール使用周辺装置

### 12.13.1 入力キャプチャ / 出力比較用タイマー

入力キャプチャおよび出力比較周辺装置は 2 つあるタイマーモジュールの内 1 つをタイマーとして選択できます。詳細は第 13 章 . 「入力キャプチャ」、第 14 章 . 「出力比較モジュール」、およびデバイスデータシートを参照してください。

### 12.13.2 A/D 特別イベントトリガー

各デバイスの、1 つのタイプ C タイマーが 16 および 32 ビットモードのいずれでも周期一致で特別 A/D 変換トリガー信号を生成する機能をもっています。タイマーモジュールは A/D サンプリング論理に対して変換開始信号を提供します。

- T32 = 0、16 ビットタイマーレジスタ (TMRx) と各 16 ビット周期レジスタ (PRx) 間で一致が発生した場合、A/D 特別イベント信号が発生する。
- T32 = 1、32 ビットタイマー (TMRx:TMRy) および 32 ビットの各連結した周期レジスタ (PRx:PRy) の間で一致が発生した場合 A/D 特別イベントトリガー信号が生成されます。

特別イベントトリガー信号はタイマーによって常に生成されます。トリガーソースは A/D コンバータ制御レジスタで選択される必要があります。追加情報については第 17 章 . 「10 ビット A/D コンバータ」、第 18 章 . 「12- ビット A/D コンバータ」、およびデバイスデータシートを参照してください。

### 12.13.3 外部割込みピンとしてのタイマー

各タイマーの外部クロック入力ピンは追加割り込みピンとして使用可能です。割り込みを提供するためには、タイマー周期レジスタである PRx はゼロ以外の値で書き込み、TMRx レジスタを周期レジスタに書き込まれた値よりも 1 以上小さい値に初期化します。タイマーは 1:1 のクロックプリスケーラに構成される必要があります。割り込みは外部クロック信号の次の立ち上がりエッジが検出される際に生成されます。

### 12.13.4 入出力ピン制御

タイマーモジュールが外部クロックまたはゲート動作作用に有効化され、構成される場合、ユーザーは入出力ピン方向が入力用に設定されているようにする必要があります。タイマーモジュールを有効化してもピンの方向性の設定とはなりません。

表 12-1: 特別機能タイマーモジュール関連レジスタ

SFR名	アドレス	ピット 15	ピット 14	ピット 13	ピット 12	ピット 11	ピット 10	ピット 9	ピット 8	ピット 7	ピット 6	ピット 5	ピット 4	ピット 3	ピット 2	ピット 1	ピット 0	全リセットの値
TMR1	0100																0000 0000 0000 0000	
PR1	0102																1111 1111 1111 1111	
T1CON	0104	TON	—	TSIDL	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	TSYNC	TCS	—	0000 0000 0000 0000	
TMR2	0106																0000 0000 0000 0000	
TMR3HLD	0108																0000 0000 0000 0000	
TMR3	010A																0000 0000 0000 0000	
PR2	010C																1111 1111 1111 1111	
PR3	010E																1111 1111 1111 1111	
T2CON	0110	TON	—	TSIDL	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T32	—	TCS	—	0000 0000 0000 0000	
T3CON	0112	TON	—	TSIDL	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	—	TCS	—	0000 0000 0000 0000	
TMR4	0114																0000 0000 0000 0000	
TMR5HLD	0116																0000 0000 0000 0000	
TMR5	0118																0000 0000 0000 0000	
PR4	011A																1111 1111 1111 1111	
PR5	011C																1111 1111 1111 1111	
T4CON	011E	TON	—	TSIDL	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T32	—	TCS	—	0000 0000 0000 0000	
T5CON	0120	TON	—	TSIDL	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	—	TCS	—	0000 0000 0000 0000	
IFS0	0084	CNIF	BCLIF	I2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT01F 0000 0000 0000 0000	
IFS1	0086	IC61F	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF 0000 0000 0000 0000	
IEC0	008C	CNIE	BCLIE	IC2IE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE 0000 0000 0000 0000	
IEC1	008E	IC6IE	IC5IE	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE 0000 0000 0000 0000	
IPC0	0094	—		T1IP<2:0>		—		0C1IP<2:0>		—		IC1IP<2:0>					INT0IP<2:0> 0100 0100 0100 0100	
IPC1	0096	—		T3IP<2:0>		—		T2IP<2:0>		—		0C2IP<2:0>					IC2IP<2:0> 0100 0100 0100 0100	
IPC5	009E	—		INT2IP<2:0>		—		T5IP<2:0>		—		T4IP<2:0>					0C4IP<2:0> 0100 0100 0100 0100	

注: メモリマップの詳細についてはデバイスデータシートを参照してください。

## 12.14 設計の秘訣

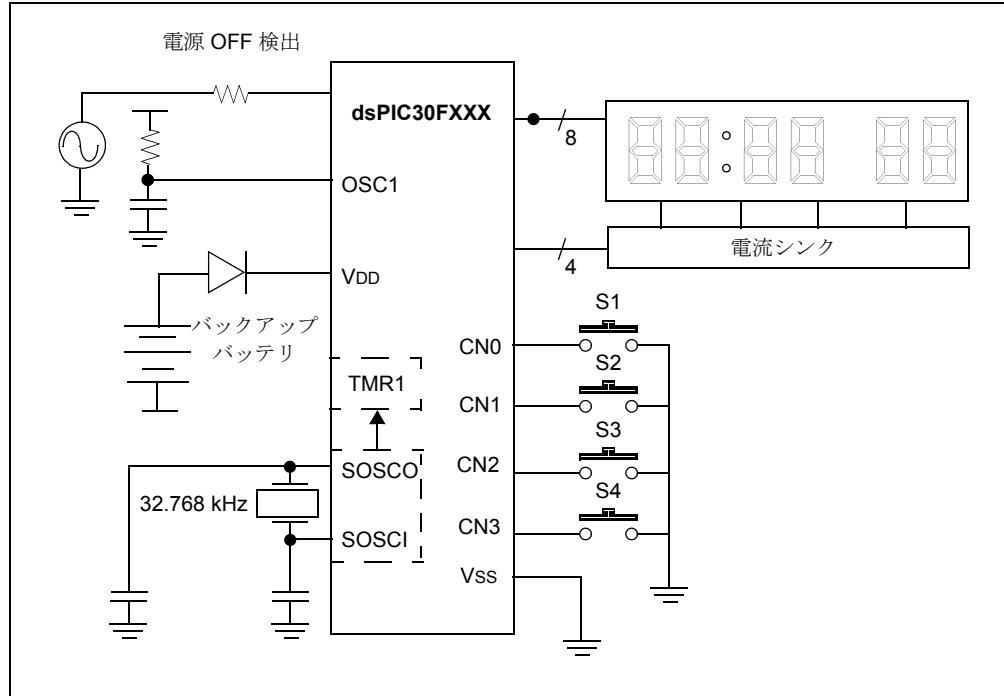
**質問 1:** タイマーモジュールは **SLEEP** モードからデバイスをウェイクアップさせることができますか？

**答え：** 可能です。ただし、タイマー 1 のみがデバイスを **SLEEP** モードからウェイクアップさせる機能をもっています。これはタイマー 1 が TMR1 レジスタに外部の非同期クロックソースによるカウントアップを許可しているからです。TMR1 レジスタが PR1 レジスタと等しくなった場合、タイマー 1 割り込みが T1IE 制御ビットを使用して有効化されると、デバイスは **SLEEP** モードからウェイクアップします。詳細は、**セクション 12. 12.1¢SLEEP モードでのタイマー動作** 参照してください。

### 12.14.1 アプリケーション例

図 12-7 のアプリケーション例は、タイマー 1（タイプ A タイマー）が外部 32.768 kHz 発振器からドライブされています。外部 32.768 kHz 発振器は通常リアルタイムの動作が必要なアプリケーションで使用されますが、最低可能消費電力を実現するのにも適しています。タイマー 1 発振器によってタイマーがカウントアップを継続している間デバイスを **SLEEP** モードにセット可能になります。タイマー 1 がオーバーフローしたとき、割り込みによってデバイスがウェイクアップし、それにより適切なレジスタが更新可能になります。

図 12-7: タイマー 1 アプリケーション



この例では、リアルタイムクロックのタイマーとして 32.768 kHz クリスタルが使用されています。クロックが 1 秒間隔での更新が必要な場合、タイマー 1 が希望周期で一致する値を周期レジスタ PR1 へロードする必要があります。1 秒周期のタイマー 1 一致イベントの場合、PR1 レジスタを 0x8000 値でロードしてください。

**注：** タイマー 1 クロックソースはシステムクロックに対して非同期なので、正確にリアルタイムを維持するためには、TMR1 レジスタには決して書き込まないください。TMR1 レジスタへの書き込みはリアルタイムカウンタ値を破壊し、結果不正確なタイマーとなります。

## 12.15 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正し、制限を設けて（必要な場合）使用可能です。現状、タイマーモジュールに関連する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 #
非同期クロックモードでのタイマー 1 使用	AN580
PIC16C924 を備えた他のクロック	AN649

注： dsPIC30F ファミリーのデバイスに関しての、その他のアプリケーションノートやコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 12.16 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

この版は、dsPIC30F のタイマーモジュールに関する技術情報の変更を含みます。



**MICROCHIP**

---

---

## 第 13 章. 入力キャプチャ

---

---

### ハイライト

この章は、以下の項目を含んでいます。

13.1 序章 .....	13-2
13.2 入力キャプチャレジスタ .....	13-3
13.3 タイマー選択 .....	13-4
13.4 入力キャプチャイベントモード .....	13-4
13.5 キャプチャバッファオペレーション .....	13-8
13.6 入力キャプチャ割り込み .....	13-9
13.7 UART オートボーサポート .....	13-9
13.8 省電力状態における入力キャプチャのオペレーション .....	13-10
13.9 入出力ピン制御 .....	13-10
13.10 入力キャプチャモジュールと関連する特別関数レジスタ .....	13-11
13.11 製品情報 .....	13-12
13.12 関連するアプリケーションノート .....	13-13
13.13 改訂履歴 .....	13-14

**13**

入力キャプチャ

## 13.1 序章

この章では、入力キャプチャモジュールおよび関連する動作モードについて説明します。入力キャプチャモジュールは、入力ピンにイベントが発生した場合、選択可能な 2 種類のタイムベースのうち 1 つがタイマー値をキャプチャするために用いられます。入力キャプチャ機能は、度数(時間)を必要とするアプリケーションやパルス測定に非常に有用です。図 13-1 では、簡単な入力キャプチャモジュールのロックダイアグラムが記述されています。

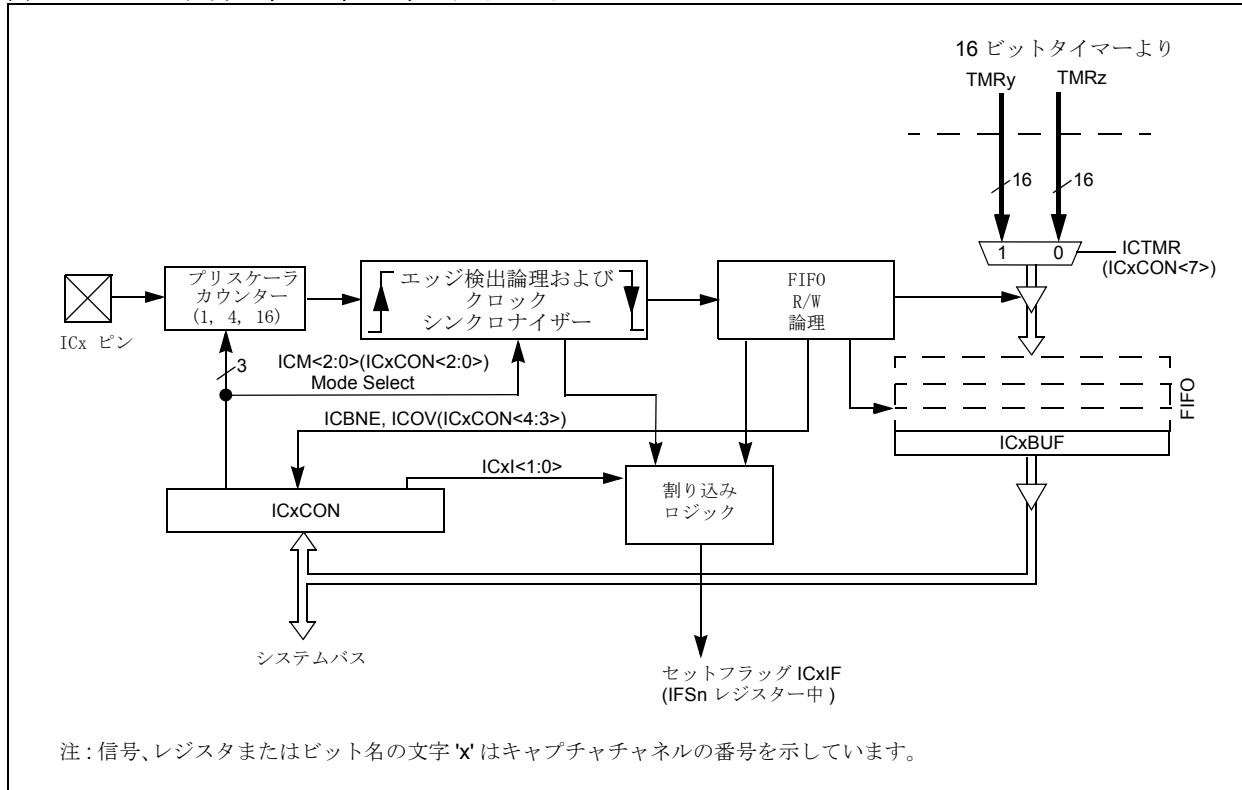
個別のデバイスで利用可能なチャネルの数について詳しくは、個別のデバイス用データシートを参照してください。全ての入力キャプチャチャネルの機能は同一です。本章では、ピンネームまたはレジスタ名の 'x' という文字は、特定の入力キャプチャチャネルを指しています。

入力キャプチャモジュールには、複数の動作モードがあり、ICxCON レジスタにより動作モードが選択されます。動作モードには以下のようないわゆるがあります。

- ICx ピン入力により生じる立ち下がりエッジごとにタイマー値をキャプチャ
- ICx ピン入力により生じる立ち上がりエッジごとにタイマー値をキャプチャ
- ICx ピン入力により生じる立ち上がりエッジ 4 回ごとにタイマー値をキャプチャ
- ICx ピン入力により生じる立ち上がりエッジ 16 回ごとにタイマー値をキャプチャ
- ICx ピンに入力により生じる立ち下がり・立ち下がりエッジごとにタイマー値をキャプチャ

入力キャプチャモジュールには、4 段階の FIFO バッファが用意されています。CPU 割り込みを発生させるために必要なキャプチャイベント数はユーザーが選択できます。

図 13-1: 入力キャプチャロックダイアグラム



注：信号、レジスタまたはビット名の文字 'x' はキャプチャチャネルの番号を示しています。

## 13.2 入力キャプチャレジスタ

dsPIC30F デバイスで利用可能な各キャプチャチャネルには以下のようなレジスタがあり、「x」はキャプチャチャネルの番号を示しています。

- IC<sub>x</sub>CON: 入力キャプチャ制御レジスタ
- IC<sub>x</sub>BUF: 入力キャプチャバッファレジスタ

レジスタ 13-1: IC<sub>x</sub>CON: 入力キャプチャ x 制御レジスタ

上位バイト:							
U-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
—	—	ICSDL	—	—	—	—	—
ビット 15							ビット 8

下位バイト:

R/W-0	R/W-0	R/W-0	R-0, HC	R-0, HC	R/W-0	R/W-0	R/W-0
ICTMR	ICI<1:0>	ICOV	ICBNE	ICM<2:0>			
ビット 7							ビット 0

ビット 15- 未実装: ‘0’ として読み出し

14

ビット 13 ICSDL: IDLE 制御で、入力キャプチャモジュール停止ビット

1 = CPU IDLE モードで入力キャプチャモジュールが停止。  
0 = 入力キャプチャモジュールが CPU IDLE モードで動作を継続。

ビット 12 未実装: ‘0’ として読み出し

-8

ビット 7 ICTMR: 入力キャプチャタイマー選択ビット

1 = TMR2 の内容がキャプチャイベント発生時にキャプチャされます。  
0 = TMR3 の内容がキャプチャイベント発生時にキャプチャされます。

注: タイマー選択は異なる場合があります。詳しくはデバイスのデータシートを参照。

ビット 6- ICI<1:0>: 割り込みを発生させるキャプチャ回数の選択

5  
11 = キャプチャイベント 4 回ごとに割り込み  
10 = キャプチャイベント 3 回ごとに割り込み  
01 = キャプチャイベント 2 回ごとに割り込み  
00 = キャプチャイベントごとに割り込み

ビット 4 ICcov: 入力キャプチャオーバーフロー状態フラグ (読み出し専用) ビット

1 = 入力キャプチャオーバーフロー発生  
0 = 入力キャプチャオーバーフロー発生なし

ビット 3 ICBNE: 入力キャプチャバッファの状態が空 (読み出し専用) ビット

1 = 入力キャプチャバッファが空ではなく、最低 1 個の読み出し可能キャプチャ値がある  
0 = 入力キャプチャバッファが空

ビット 2- ICM<2:0>: 入力キャプチャモード選択ビット

0  
111 = デバイスが SLEEP または IDLE モードのとき、入力キャプチャが割り込みピンとしてのみ機能。(立ち上がりエッジ検出のみで、その他全ての制御ビットは実行不可能)  
110 = 未使用 (モジュール無効)  
101 = 立ち上がりエッジ 16 回ごとにキャプチャ  
100 = 立ち上がりエッジ 4 回ごとにキャプチャ  
011 = 立ち上がりエッジ毎回につきキャプチャ  
010 = 立ち下がりエッジ毎回につきキャプチャ  
001 = 全てのエッジ (立ち上がりまたは立ち下がり) 発生ごとにキャプチャ (ICI<1:0> は本モードでは割り込み発生を制御できません)。  
000 = 入力キャプチャモジュール停止。

凡例:

HC = ハードウェアから消去

HS = ハードウェアに

セット

R = 読み出し可能ビット

W = 書き込み可能ビット U = 未使用ビット、‘0’ として読み込む

-n = POR の値

‘1’ = ビットセット

‘0’ = ビットをクリア x = ビットは不定

## 13.3 タイマー選択

dsPIC30F の各デバイスには、1つ以上の入力キャプチャチャネルが用意されているものもあります。各チャネルには、タイムベース用に二つの 16 ビットタイマーのうち 1つを選択できます。選択可能な個々のタイマーについてはデバイスのデータシートを参照してください。

タイマーリソースの選択は、ICTMR 制御ビット ( $ICxC0N<7>$ ) によって行います。タイマーの設定は、内部のクロックソース ( $Fosc/4$ )、または TxCK ピンに入力される同期させた外部クロックで動作させるようにします。

## 13.4 入力キャプチャイベントモード

入力キャプチャモジュールは、イベントが  $ICx$  ピンで発生した際に、選択されたタイムベースレジスタの 16 ビット値をキャプチャします。キャプチャ可能なイベントは、以下の 3 種類に分類できます。

1. 単純なキャプチャイベントモード
  - $ICx$  ピンで入力時に立ち上がりエッジが発生するごとにタイマー値をキャプチャ。
  - $ICx$  ピンで入力時に立ち下がりエッジが発生するごとにタイマー値をキャプチャ。
2. エッジ（立ち上がり・立ち下がり）発生ごとにタイマー値をキャプチャ。
3. プリスケーラキャプチャイベントモード
  - $ICx$  ピンで入力信号の立ち上がりエッジ 4 回ごとにタイマー値をキャプチャ。
  - $ICx$  ピンで入力信号の立ち上がりエッジ 16 回ごとにタイマー値をキャプチャ。

これらの入力キャプチャモードは適切な入力キャプチャモードのビットを  $ICM<2:0>$  ( $ICxCON<2:0>$ ) に設定することで構成されます。

### 13.4.1 単純なキャプチャイベント

キャプチャモジュールは、 $ICx$  ピンへの入力の選択されたエッジ（立ち上がりエッジか、立ち下がりエッジか）をモードにより決定により、タイマーカウント数（TMR2 または TMR3）をキャプチャすることができます。モードは  $ICM<2:0>$  ( $ICxCON<2:0>$ ) ビットをそれぞれ ‘010’ または ‘011’ に設定することで特定されます。これらのモードでは、プリスケーラカウンタは用いられません。単純なキャプチャイベントを示す簡単なタイミング図については、図 13-3 および図 13-2 を参照してください。

入力キャプチャロジックは、キャプチャピン信号の立ち上がりまたは立ち下がりエッジを検出し、内部クロックに同期させます。立ち上がりまたは立ち下がりエッジが発生した場合、キャプチャモジュールロジックは現在のタイムベース値をキャプチャバッファに書き込み、割り込み発生ロジックを起動します。発生したキャプチャイベントの数が、 $ICI<1:0>$  制御ビットで特定される数と一致した場合、それぞれのキャプチャチャネル割り込み状態フラグである  $ICxF$  が、キャプチャバッファが書き込みイベント完了後 2 命令サイクル経過後にセットされます。

キャプチャタイムベースが命令サイクル毎にカウントアップする場合、キャプチャカウント値は  $ICx$  ピンでイベントが発生してから 1 または 2 命令サイクル後の数値になります。この時間の遅れは、命令サイクルクロックのどこで実際の  $ICx$  エッジイベントが検出されるかと、入力キャプチャロジックによる遅れと関係があります。キャプチャタイムベースに対する入力クロックをプリスケールした場合、キャプチャ値の遅れを取り除くことができます。詳しくは図 13-3 および図 13-2 を参照してください。

入力キャプチャピンは、H レベルの時間と L レベルの時間それぞれに関して最低限の仕様があります。詳しくはデバイスデータシートの「電気的特性」の項目を参照してください。

図 13-2: 単純キャプチャイベントのタイミング図、タイムベースプリスケーラ = 1:1

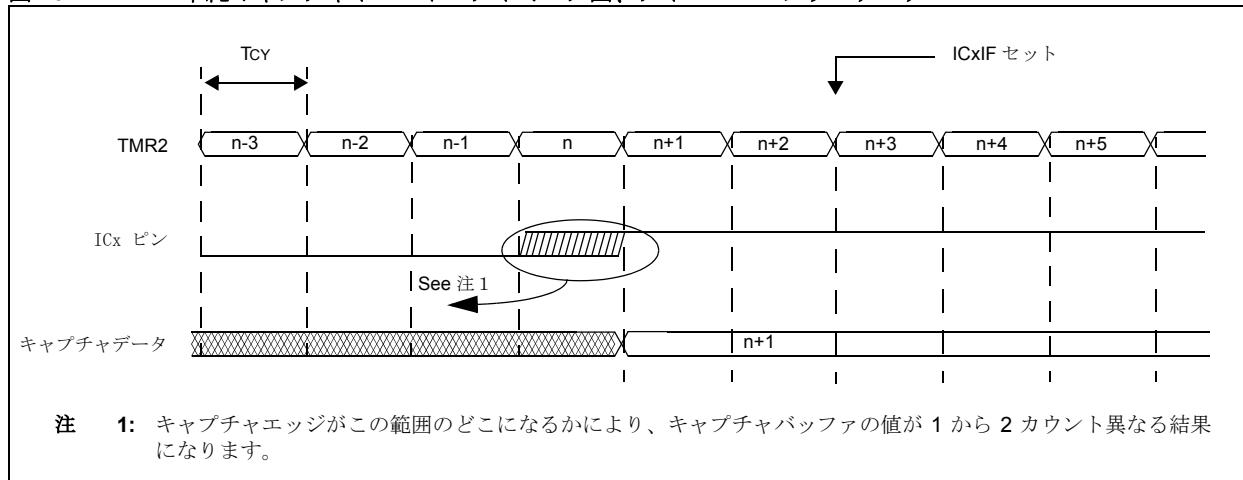
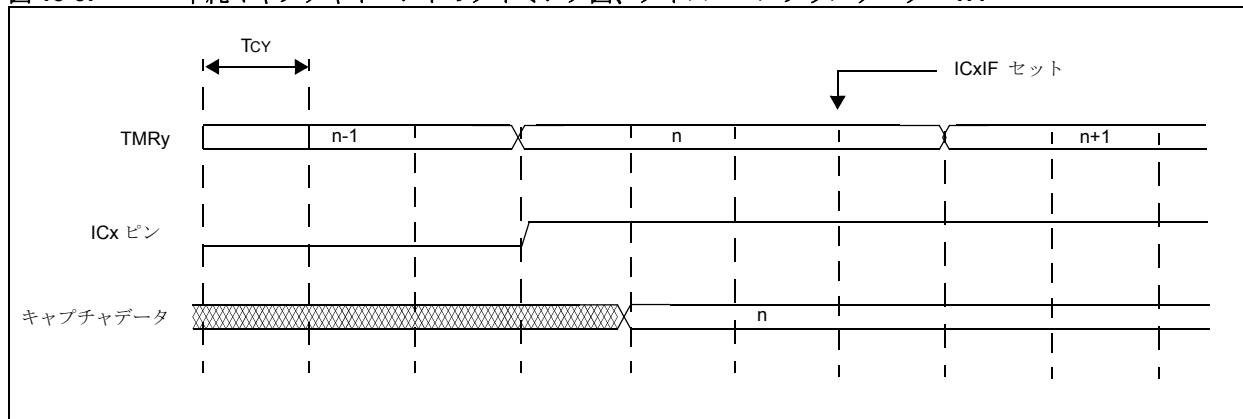


図 13-3: 単純キャプチャイベントのタイミング図、タイムベースプリスケーラ = 1:4



## 13.4.2 プリスケーラキャプチャイベント

キャプチャモジュールには、プリスケールキャプチャモードが 2 種類用意されています。プリスケールモードは  $ICM<2:0>$  ( $ICxCON<2:0>$ ) ビットをそれぞれ ‘100’ または ‘101’ に設定することで選択されます。これらのモードでは、キャプチャイベントが発生するまでに、立ち上がりエッジイベントを 4 回または 16 回カウントします。

キャプチャプリスケーラカウンタは、キャプチャピンに対して有効な立ち上がりエッジが発生するごとに増分されます。ピンの入力の立ち上がりエッジは、事实上カウンタのクロックの役割を果たしています。プリスケーラカウンタが 4 カウントまたは 16 カウントを示したとき(選択モードにより異なる)、カウンタは  $\neq$  有効な  $\&$  キャプチャイベント信号を出力します。この信号は命令サイクルクロックに同期しています。同期したキャプチャイベント信号により、キャプチャバッファ書き込みイベントが発生し、割り込み発生ロジックが起動されます。各キャプチャチャネルの割り込みステータスフラグである  $ICxIF$  が、キャプチャバッファ書き込みイベントから 2 命令サイクル後にセットされます。

キャプチャタイムベースが命令サイクル毎にカウントアップする場合、キャプチャされるカウント値は、キャプチャイベントから 1、2 命令サイクル後の数値になります。

入力キャプチャピンは、H レベルの時間と L レベルの時間それぞれに関して最低限の仕様があります。詳しくはデバイスデータシートの「電気的特性」の項目を参照してください。

あるプリスケール設定から別の設定に切り替えたとき、割り込みが発生する可能性があります。また、プリスケーラカウンタが元に戻らず、プリスケーラのカウントがゼロでない状態でキャプチャが始まることもあります。例 13-1 では、キャプチャプリスケール設定を切り替える場合に推奨される方法が示されています。

プリスケーラカウンタは以下の場合元に戻ります。

- キャプチャチャネルが停止されたとき (すなわち、 $ICM<2:0> = ‘000’$  の時)
- デバイスのリセット時

以下の場合はプリスケーラカウンタは元に戻りません。

- ユーザーがあるアクティブなキャプチャモードから、別のキャプチャモードに切り替えたとき。

## 例 13-1: プリスケールされたキャプチャコードの例

```

; 以下のコード例は、キャプチャイベント 2 回ごとに割り込みが発生し、立ち上がり
; エッジ 4 回ごとにキャプチャが発生し、タイマー 2 がタイマーベースとして選択さ
; れるように入力キャプチャ 1 モジュールをセットするものです。このコード例では
; 予測のつかない割り込みを回避するために ICxCON をクリアしています。

BSET    IPC0, #IC1IP0      ; 希望の優先順位で割り込みが発生するように
BCLR    IPC0, #IC1IP1      ; 入力キャプチャ 1 をセット
BCLR    IPC0, #IC1IP2      ; (この例では優先順位がレベル 1)
BCLR    IFS0, #IC1IF       ; IC1 割り込みステータスフラグを消去
BSET    IEC0, #IC1IE       ; IC1 割り込みを有効化

CLR     IC1CON             ; 入力キャプチャ 1 モジュールをオフ
MOV     #0x00A2, w0         ; 作業レジスタに、新しいプリスケーラモード
MOV     w0, IC1CON          ; をロードし、IC1CON に書き込む

MOV     #IC1BUF, w0         ; キャプチャデータフェッチポインタを作成
MOV     #TEMP_BUFF, w1       ; データ保存ポインタを作成
                           ; TEMP_BUFF がすでに定義されているとみなす

; 次のコードは、割り込みが発生したときのキャプチャバッファ読み出し方法を示しています。
; W0 にはキャプチャバッファのアドレスが含まれています。

; 入力キャプチャ 1 の ISR コード例

_IC1 interrupt:
BCLR    IFS0, #IC1IF       ; 対応割り込みフラグをリセット
MOV     [w0++], [w1++]      ; 最初のキャプチャを読み出し保存
MOV     [w0], [w1]           ; 二回目のキャプチャを読み出し保存
                           ; 残りのユーザーコードをここに記述
RETFIE                         ; ISR から復帰

```

**注:** 新しいモードに切り替える前に、ユーザーがキャプチャモジュールを停止させる(すなわち ICM<2:0> (ICxCON<2:0>) 消去)ことが強く推奨されます。ユーザーが別のキャプチャモードに切り替えた場合、プリスケーラカウンタは元に戻りません。そのため、プリスケーラカウンタがゼロに戻らない(モード切替時点)ために、最初のキャプチャイベントと割り込みが発生する可能性があります。

### 13.4.3 エッジ検出モード

キャプチャモジュールは、**ICx** ピンに対し送られた入力信号の全ての立ち上がり、立ち下がりエッジのタイムベースカウント数をキャプチャできます。**ICM<2:0>** (**ICxCON<2:0>**) ビットを‘001’に設定することで、エッジ検出モードを選択できます。エッジ検出モードでは、キャプチャプリスケーラカウンタは用いられません。簡単なタイミング図については図 13-4 を参照してください。

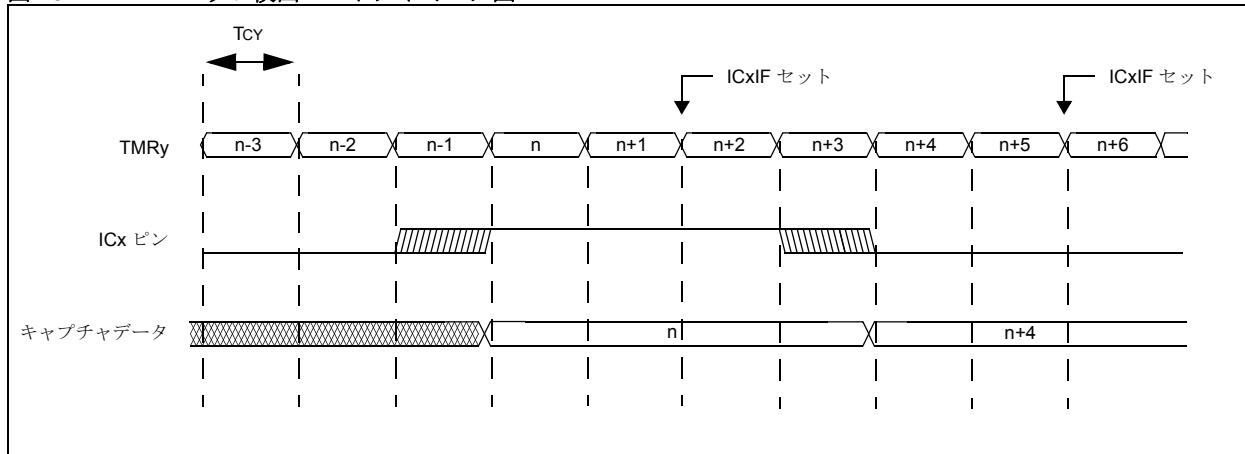
入力キャプチャモジュールがエッジ検出モードに設定された場合、モジュールは以下の役割を果たします。

- 立ち上がり、立ち下がりエッジが発生するたびに入力キャプチャ割り込みフラグ (**ICxF**) をセットします。
- キャプチャ割り込みモードビットである **IC1<1:0>** (**ICxCON<6:5>**) はエッジ検出モードでは用いられません。キャプチャイベントごとに割り込みが発生します。
- キャプチャオーバーフローの **ICOV** (**ICxCON<4>**) ビットは発生しません。

単純キャプチャイベントモードと同様に、入力キャプチャロジックによりキャプチャピン信号の立ち上がりエッジおよび立ち下がりエッジが検出され、内部クロックに同期されます。立ち上がりまたは立ち下がりエッジが発生した場合、キャプチャモジュールロジックにより現在のタイマーカウントがキャプチャバッファに書き込まれ、割り込み発生ロジックが起動されます。各キャプチャチャネルは割り込みステータスフラグである **ICxF** を、キャプチャバッファ書き込みイベントが発生してから 2 命令サイクル後にセットします。

キャプチャされたタイマーのカウント数は、**ICx** ピンでエッジが発生してから、1 または 2 **TCY** (命令サイクル) 多く表示されます (図 13-4 参照)。

図 13-4: エッジ検出モードタイミング図



### 13.5 キャプチャバッファオペレーション

各キャプチャチャネルには 4 層の FIFO バッファがあります。**ICxBUF** レジスタは、メモリにマップされているためユーザーにとって可視的なバッファレジスタです。

入力キャプチャモジュールがリセットされ、**ICM<2:0> = 000** (**ICxCON<2:0>**) となったとき、入力キャプチャロジックは以下のようになります。

- オーバーフロー条件フラグクリア (すなわち **ICxOV** (**ICxCON<4>**) を‘0’にする)
- キャプチャバッファを空の状態にリセット (すなわち **ICBNE** (**ICxCON<3>**) を‘0’にする)

FIFO バッファが以下の条件で読み出された場合結果が不定となります。

- まず入力キャプチャモジュールが無効化され、少し後に再び有効化された場合。
- バッファが空の時に FIFO が読み出された場合。
- デバイスのリセット後。

FIFO バッファのステータスを提供するステータスフラグは 2 種類あります。

- ICBNE** (**ICxCON<3>**) : 入力キャプチャバッファが空でない。
- ICOV** (**ICxCON<4>**) : 入力キャプチャのオーバーフロー

### 13.5.1 入力キャプチャバッファが空でない (ICBNE)

**ICBNE** 読み出し専用ステータスビット (**ICxC0N<3>**) は、最初の入力キャプチャイベント時にセットされ、全てのキャプチャイベントがキャプチャバッファから読み出されるまで設定は維持されます。例えば、3つのキャプチャイベントが発生した場合、**ICBNE(ICxC0N<3>)** フラグが消去されるには、キャプチャバッファが3回読み出される必要があります。キャプチャイベントが4回なら、**ICBNE(ICxC0N<3>)** フラグ消去には4回読み出される必要があります。キャプチャバッファを読み出すごとに、残ったデータはトップロケーションに移されます。**ICBNE** はキャプチャバッファの状態を反映するものであるため、**ICBNE** ステータスビットは全てのデバイス **RESET** 時には消去されます。

### 13.5.2 入力キャプチャオーバーフロー (ICOV)

**ICOV** 読み出し専用ビット (**ICxC0N<4>**) は、キャプチャバッファがオーバーフローした場合にセットされます。4回キャプチャイベントが発生しバッファが満杯のときに、バッファ読み出し前に5回目のキャプチャイベントが発生した場合には、オーバーラン条件が発生し、**ICOV (ICxC0N<4>)** がロジック ‘1’ にセットされ、各キャプチャイベントの割り込みが発生しなくなります。さらに、5回目のキャプチャイベントは記録されず、後のキャプチャイベントにより現在のバッファの内容が変化しなくなります。

オーバーラン条件を消去するためには、キャプチャバッファを4回読み出す必要があります。4回目に読み出されると、**ICOV (ICxC0N<4>)** のステータスフラグは消去され、キャプチャチャネルは通常の動作を再開します。

以下の方法でオーバーフロー条件を消去できます。

- **ICM<2:0>(ICxCON<2:0>)** = 000 にセットします。
- **(ICxCON<3>)** = 0 になるまでキャプチャバッファを読み出します。
- 全てのデバイス **RESET**

#### 13.5.2.1 ICOV および割り込み専用モード

入力キャプチャモジュールは、外部割り込みピンとして機能するよう設定することもできます。このモードに設定するためには、**ICI<1:0> (ICxCON<6:5>)** ビットが ‘00’ にセットされる必要があります。割り込みはバッファの読み出しと関係なく発生します。

### 13.6 入力キャプチャ割り込み

入力キャプチャモジュールは、選択された数のキャプチャイベントに基づき、割り込みを発生させる能力があります。キャプチャイベントとは、タイムベース値をキャプチャバッファに書き込むことです。設定は制御ビット **ICI<1:0> (ICxCON<6:5>)** により行われます。

**ICI<1:0> = ‘00’** の場合を除き、バッファオーバーフロー条件が除去されるまで割り込みは発生しません（セクション 13.5.2 「入力キャプチャオーバーフロー (ICOV)」 参照）。リセットまたは読み出しオペレーションによりキャプチャバッファが空になると、割り込みカウントはリセットされます。これにより割り込みカウントが FIFO 入力ステータスに再同期化されます。

#### 13.6.1 割り込み制御ビット

各入力キャプチャチャネルには割り込みフラグステータスビット (**ICxF**)、割り込み有効化ビット (**ICxE**) および割り込み優先順位制御ビット (**ICxIP<2:0>**) があります。周辺装置の割り込みについて詳しくは、第 6 章、「割り込み」を参照してください。

### 13.7 UART オートボーサポート

UART がオートボーモードに設定され、**ABAUD = 1 (UxMODE<5>)** である時に、入力キャプチャモジュールは **UART** モジュールにより使われます。ABAUD 制御ビットが設定されると、**UART RX** ピンは内部で割り当てられた入力キャプチャモジュールに接続されます。キャプチャモジュールと関連付けられた入出力ピンは接続が切断されます。ボーレートは、**NULL** という文字が受信された時に、**START** ビットの幅の測定により決定されます。オートボー機能を利用するためにはキャプチャモジュールはエッジ検出モード（全ての立ち上がり・立ち下がりエッジをキャプチャ）に設定する必要があります。各 **UART** の入力キャプチャモジュールの割当では、選択された **dsPIC30F** デバイスのタイプにより異なります。オートボーサポートについて詳しくは、デバイスデータシートを参照してください。

## 13.8 省電力状態における入力キャプチャのオペレーション

### 13.8.1 スリープモードにおける入力キャプチャオペレーション

デバイスがスリープモードに入った場合、システムクロックは無効化されます。スリープモードでは、入力キャプチャモジュールは外部割り込み要因としてのみ機能できます。制御ビット  $ICM<2:0> = '111'$  にセットすることで、スリープモードが有効になります。スリープモードの時は、キャプチャピンの立ち上がりエッジによりデバイスがスリープモードからウェイクアップします。各モジュールの割り込みビットが有効化され、モジュールの優先順位が必要な優先順位に達していると、割り込みが発生します。

キャプチャモジュールが、 $ICM<2:0> = '111'$  以外に設定され、dsPIC30F がスリープモードに入った場合は、ピンによる外部からの立ち上がり、または立ち下がりの刺激ではスリープモードからウェイクアップしません。

### 13.8.2 IDLE モードにおける入力キャプチャオペレーション

デバイスが IDLE モードに入ったとき、システムクロックソースは機能を維持し、CPU はコード実行を停止します。入力キャプチャモジュールが、IDLE モード下で停止するか、動作を保つかどうかは ICSIDL ビット ( $ICxCON<13>$ ) により選択されます。

$ICSIDL = 0$  ( $ICxCON<13>$ ) となった場合、モジュールは IDLE モードでも動作を維持します。制御ビットの  $ICM<2:0>$  ( $ICxCON<2:0>$ ) により、キャプチャプリスケールが 4:1 および 16:1 に設定された場合も、入力キャプチャモジュールは完全に機能を維持します。ただし、モジュールが IDLE モードでも動作を維持するためには選択されたタイマーが IDLE モードでも有効であることが必要です。

入力キャプチャモードが  $ICM<2:0> = '111'$  に設定された場合、入力キャプチャピンは外部割り込みピンとしてしか機能しません。このモードの場合、キャプチャピンの立ち上がりエッジにより IDLE モードからデバイスがウェイクアップします。キャプチャタイムベースは有効化する必要はありません。各モジュール割り込み有効化ビットが設定されており、ユーザーの決定した優先順位が現在の CPU の優先順位レベルより上であった場合は、割り込みが発生します。

$ICSIDL = 1$  ( $ICxCON<13>$ ) に設定された場合、モジュールは IDLE モードで停止します。モジュールは IDLE モードで停止した場合、スリープモードの場合と同様の機能を果たします。（セクション 13.8.1「スリープモードにおける入力キャプチャオペレーション」参照）

### 13.8.3 スリープ・IDLE モードからのデバイスウェイクアップ

デバイスが IDLE モードか SLEEP モードのとき、もし有効にされていれば、入力キャプチャイベントによりウェイクアップさせるか割り込みを発生させることができます。

タイマーが有効化されているかどうかに関係なく、以下の条件を満たす場合キャプチャイベント発生時に入力キャプチャモジュールはデバイスを SLEEP あるいは IDLE モードからウェイクアップさせます。

- インプットキャプチャモードビットが  $ICM<2:0> = '111'$  ( $ICxCON<2:0>$ ) に設定され、さらに
- 割り込み可能ビット ( $ICxIE$ ) がセットされている場合。

これと同じウェイクアップ機能により、以下の条件を満たす場合 CPU に割り込みが発生します。

- 対応する割り込みが有効化されており ( $ICxCE = 1$ )、必要な優先順位を満たしていること。

このウェイクアップ機能は、外部ピン割り込みを増やすには大変有用です。このモードで入力キャプチャモジュールが用いられたとき、以下の条件が当てはまります。

- キャプチャプリスケーラカウンタはこのモードでは用いられない。
- $ICI<1:0>$  ( $ICxCON<6:5>$ ) ビットが実行されない。

## 13.9 入出力ピン制御

キャプチャモジュールが有効化されたとき、ユーザーは、関連する TRIS ビットを設定し、入出力ピン制御が入力モードに設定されていることを確かめる必要があります。キャプチャモジュールが有効化されたとき、ピン制御は設定されていません。さらに、入力ピンに多重使用されている全ての周辺装置を無効化する必要があります。

### 13.10 入力キャプチャモジュールと関連する特別関数レジスタ

表 13-1: 例メモリ入力マップキャプチャモジュール

SFR Name	Addr.	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	RESET State
IFS0	0084	CNIF	BCLIF	I2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0F	0000 0000 0000 0000
IFS1	0086	IC6IF	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IEC0	008C	CNIE	BCLIE	I2CIE	IR12	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	IC6IE	EI30	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IPC0	0094	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>			0100 0100 0100 0100
IPC1	0096	—	T31P<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>			0100 0100 0100 0100
IPC4	009C	—	OC3IP<2:0>			—	IC8IP<2:0>			—	IC7IP<2:0>			—	INT1IP<2:0>			0100 0100 0100 0100
IPC7	00A2	—	IC6IP<2:0>			—	IC5IP<2:0>			—	IC4IP<2:0>			—	IC3IP<2:0>			0100 0100 0100 0100
IC1BUF	0140	入力 1 キャプチャレジスタ															uuuu uuuu uuuu uuuu	
IC1CON	0142	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBN	ICM<2:0>	—	—	0000 0000 0000 0000	
IC2BUF	0144	入力 2 キャプチャレジスタ															uuuu uuuu uuuu uuuu	
IC2CON	0146	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBN	ICM<2:0>	—	—	0000 0000 0000 0000	
IC3BUF	0148	入力 3 キャプチャレジスタ															uuuu uuuu uuuu uuuu	
IC3CON	014A	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBN	ICM<2:0>	—	—	0000 0000 0000 0000	
IC4BUF	014C	入力 4 キャプチャレジスタ															uuuu uuuu uuuu uuuu	
IC4CON	014E	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBN	ICM<2:0>	—	—	0000 0000 0000 0000	
IC5BUF	0150	入力 5 キャプチャレジスタ															uuuu uuuu uuuu uuuu	
IC5CON	0152	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBN	ICM<2:0>	—	—	0000 0000 0000 0000	
IC6BUF	0154	入力 6 キャプチャレジスタ															uuuu uuuu uuuu uuuu	
IC6CON	0156	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBN	ICM<2:0>	—	—	0000 0000 0000 0000	
IC7BUF	0158	入力 7 キャプチャレジスタ															uuuu uuuu uuuu uuuu	
IC7CON	015A	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBN	ICM<2:0>	—	—	0000 0000 0000 0000	
IC8BUF	015C	入力 8 キャプチャレジスタ															uuuu uuuu uuuu uuuu	
IC8CON	015E	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBN	ICM<2:0>	—	—	0000 0000 0000 0000	

脚注: u = 初期化なし

注: 個別のメモリマップの詳細についてはデバイスデータシートを参照。

## 13.11 製品情報

**質問 1:** 入力キャプチャモジュールは、デバイスをスリープ状態からウェイクアップさせるために用いることができますか。

**回答:** できます。入力キャプチャモジュールが  $\text{ICM}<2.0> = '111'$  に設定され、各チャネル割り込み有効化ビットが定義されている時は、( $\text{ICxIE}=1$ )、キャプチャピンのクロックの立ち上がりにより、デバイスは SLEEP モードからウェイクアップします（セクション 13.8「省電力状態における入力キャプチャのオペレーション」参照）。

## 13.12 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正して使用可能です。制限がある場合もあります。現状、入力キャプチャモジュールに関連する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 #
CCP モジュールを使う 超音波距離計を作る	AN594 AN597

**注：** dsPIC30F ファミリーのデバイスに関しての、他のアプリケーションノート  
やコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧  
下さい。

## 13.13 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

13 章に関しては、技術的な内容および編集上の改訂は行われていません。ただし、マニュアル全体を通して行われた B 版の内容を反映させるよう、この章の内容は更新されています。



**MICROCHIP**

## 第 14 章. 出力比較モジュール

### ハイライト

この章は、以下の項目を含んでいます。

14.1 序章 .....	14-2
14.2 出力比較レジスタ .....	14-3
14.3 動作モード .....	14-4
14.4 省電力状態での出力比較動作 .....	14-23
14.5 入出力ピン制御 .....	14-23
14.6 設計の秘訣 .....	14-26
14.7 関連するアプリケーションノート .....	14-27
14.8 改訂履歴 .....	14-28

14

出力比較  
モジュール

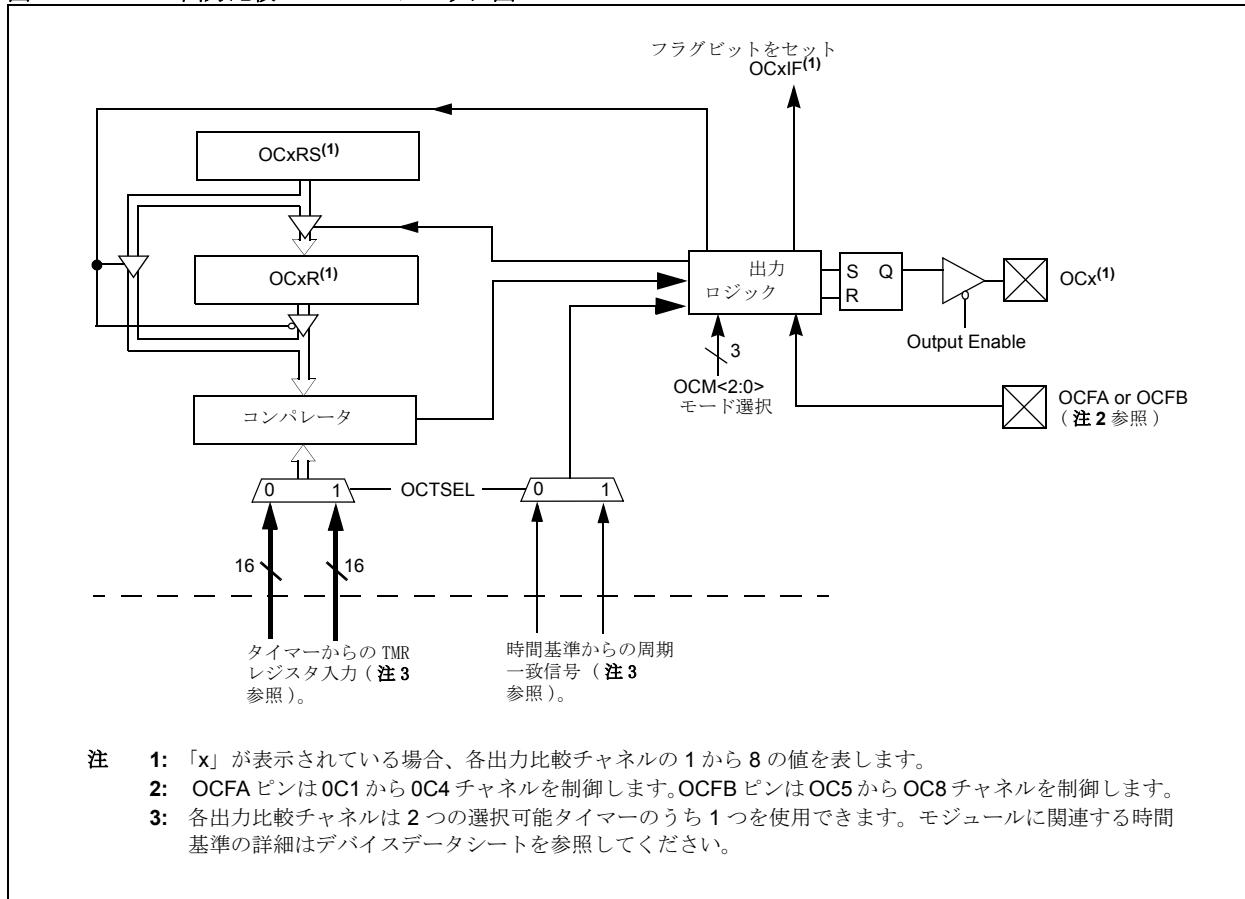
## 14.1 序章

出力比較モジュールは（選択された動作モードにより）選択したタイマーの値を1つまたは2つの比較レジスタの値で比較する機能を持っています。更には、比較一致イベントでは、单一出力パルスまたは連続パルス出力を生成する機能を持っています。ほとんどのdsPIC周辺モジュールのように、比較一致イベント割り込みを生成する機能もまた装備しています。

dsPIC30Fデバイスは専用OC1、OC2、OC3等の最大8つの出力比較チャネルをもつことが可能です。特定のデバイスで使用可能なチャネル数の詳細はそのデバイスデータシートを参照してください。全ての出力比較チャネルは機能的に同一です。この章では、ピン、レジスタ、またはビット名にある「x」が特定の出力比較チャネルを意味します。

各出力比較チャネルは2つの選択可能なタイマーのうちの1つを使用できます。タイマーはOCTSELビット(OCxCON<3>)を使用して選択されます。各出力比較チャネル数と使用可能な特定のタイマーの詳細はデバイスデータシートを参照してください。

図 14-1: 出力比較モジュールブロック図



## 14.2 出力比較レジスタ

各出力比較チャネルには以下のレジスタがあります。

- OCxCON : チャネル用の制御レジスタ
- OCxR : 出力比較チャネル用のデータレジスタ
- OCxRS : 出力比較チャネル用の二次データレジスタ

8つの比較チャネルの制御レジスタは OC1C0N から OC8C0N の名前が付けられています。8つ全ての制御レジスタは同一のビット定義をもっています。それらは以下の共通レジスタ定義によって示されています。OCxCON の「x」は出力比較チャネル番号を意味します。

レジスタ 14-1: OCxCON: 出力比較 x 制御レジスタ

上位バイト:							
U-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
—	—	OCSIDL	—	—	—	—	—
ビット 15							ビット 8

下位バイト:							
U-0	U-0	U-0	R-0, HC	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OCFLT	OCTSEL	OCM<2:0>		
ビット 7							ビット 0

ビット 15-14 未実装: ‘0’ が読み込まれます。

ビット 13 OCSIDL: IDLE モード制御のとき出力比較停止条件ビット

- 1 = 出力比較 x は CPU IDLE モードで停止  
0 = 出力比較 x は CPU IDLE モードでも動作継続

ビット 12-5 未実装: ‘0’ が読み込まれます。

ビット 4 OCFLT: PWM FAULT 条件ステータスビット

- 1 = PWM FAULT 条件の発生 (HW でのみクリア)  
0 = PWM FAULT 条件の発生なし  
(このビットは OCM<2:0> = 111 の場合のみ使用)

ビット 3 OCTSEL: 出力比較タイマー選択ビット

- 1 = タイマー 3 が比較 x のクロックタイマー  
0 = タイマー 2 が比較 x のクロックタイマー

注: 出力比較モジュールに対して使用可能な特定のタイマーの詳細はデバイスデータシートを参照してください。

ビット 2-0 OCM<2:0>: 出力比較モード選択ビット

- 111 = OCx での PWM モード、FAULT ピン有効  
110 = OCx での PWM モード、FAULT ピン無効  
101 = OCx ピンを Low で初期化、連続的な出力パルスを OCx ピンにて生成  
100 = OCx ピンを Low で初期化、シングル出力パルスを OCx ピンにて生成  
011 = 比較イベントは OCx ピンをトグルする  
010 = OCx ピンを High で初期化、比較イベントは OCx ピンを Low にセット  
001 = OCx ピンを Low で初期化、比較イベントが OCx ピンを High にセット  
000 = 出力比較チャネルは停止

凡例:

HC = ハードウェアでクリア  
されます

R = 読み込み可能ビット

W = 書き込み可能

U = 未実装、‘0’ が読み込まれます

ビット

-n = POR での値

‘1’ = ビットがセット

‘0’ = ビットはクリア

x = ビットは不定です

されます

## 14.3 動作モード

各出力比較モジュールには以下の動作モードがあります。

- シングル比較一致モード
- デュアル比較一致モード
  - シングル出力パルス生成
  - 連続出力パルス生成
- シングルパルス幅変調モード
  - フォールトプロテクション入力付き
  - フォールトプロテクション入力なし

**注 1:** ユーザーは新しいモードへ切り替える前に出力比較モジュールを OFF にする(つまり OCM<2:0> (OCxCON<2:0>) をクリア) ことをお勧めします。

**2:** この章では、選択されたタイマーソースに関する SFR に対する参照は ‘y’ 接尾辞で示されています。例えば、TyCON は選択されたタイマーソースのタイマーレジスタ、PRy は選択されたタイマーソースの周期レジスタです。

### 14.3.1 シングル比較一致モード

制御ビット OCM<2:0> (OCxCON<2:0>) が ‘001’、‘010’ または ‘011’、に設定された場合、選択された出力比較チャネルはシングル出力比較一致モードの3つのうち1つに構成されます。

シングル比較モードでは、OCxR レジスタには値がロードされており、選択されたタイマーレジスタの TMRy と比較されます。比較一致イベントでは、以下のイベントの内1つが発生します。

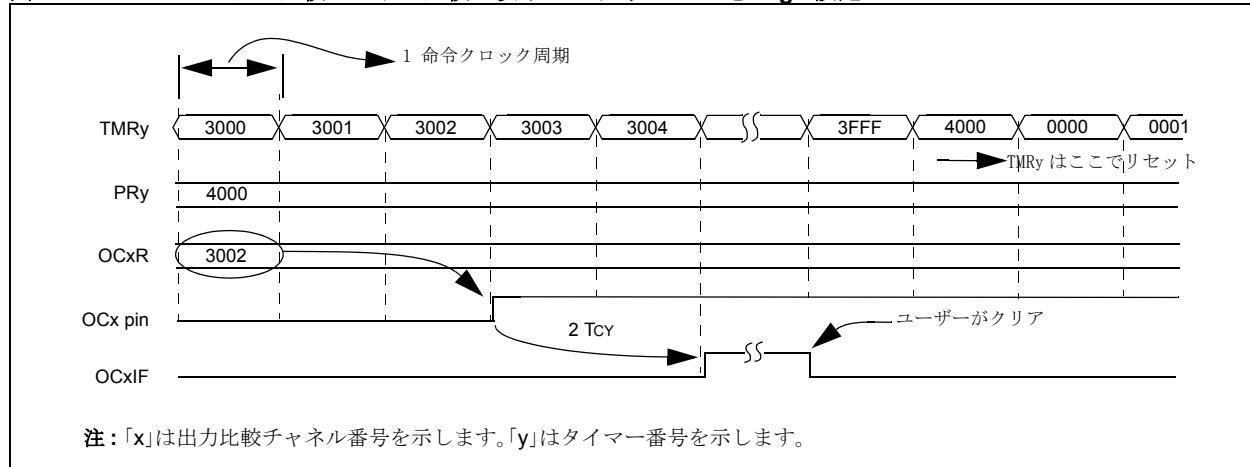
- 比較一致で OCx ピンを High にセット、ピンの初期状態は Low。割り込みがシングル比較一致イベント時に生成される。
- 比較一致で OCx ピンを Low にセット、ピンの初期状態は High。割り込みがシングル比較一致イベント時に生成される。
- 比較一致で OCx ピンをトグルする。トグルイベントは継続し、割り込みがトグルイベントごとに生成される。

## 14.3.1.1 出力を High にする比較モード

このモードに出力比較モジュールを構成するには、制御ビット OCM<2:0> = ‘001’ を設定します。更に、比較用タイマーも有効化します。いったんこの比較モードが有効化されると、出力ピン OCx は最初 Low にドライブされ、TMRy および OCxR レジスタの間で一致が発生するまで Low のままであります。図 14-2 を参照すると、いくつか注意する必要のある重要なタイミングイベントがあります。

- OCx ピンは、比較用タイマーおよび OCxR レジスタとの間で発生する比較一致後 1 命令クロック後に High にドライブされます。OCx ピンはモード変更が行われるか、またはモジュールが無効化されるまで High のままであり続けます。
- 比較用タイマーは対応する周期レジスタに含まれる値までカウントアップします。そして次の命令クロックで 0x0000 へとリセットされます。
- 各チャネル割り込みフラグ OCxIF は OCx ピンが High にドライブされた後、2 命令クロック後にセットされます。

図 14-2: シングル比較モード：比較一致イベント時の OCx を High 設定

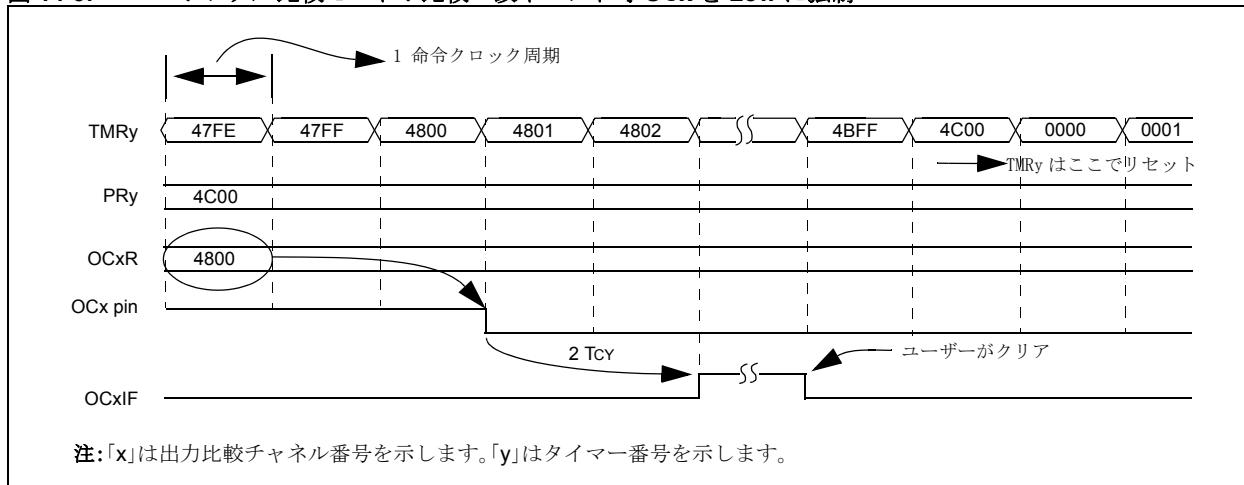


## 14.3.1.2 出力を Low にする比較モード

このモードに出力比較モジュールを構成するには、制御ビット  $OCM<2:0> = '010'$  を設定します。更に比較用タイマーも有効化される必要があります。いったんこの比較モードが有効化されると、出力ピン  $OCx$  は最初 High にドライブされ、タイマーおよび  $OCxR$  レジスタの間で一致が発生するまで High のままであります。図 14-3 を参照すると、幾つか注意する重要なタイミングイベントがあります。

- $OCx$  ピンは、比較用タイマーおよび  $OCxR$  レジスタとの間で発生する比較一致後 1 命令クロック後に Low にドライブされます。 $OCx$  ピンはモード変更が行われる、またはモジュールが無効化されるまで Low のままであります。
- 比較用タイマーは対応する周期レジスタに含まれる値までカウントアップします。そして次の命令クロックで  $0x0000$  へリセットされます。
- 各チャネル割り込みフラグ  $OCxIF$  は  $OCx$  ピンが Low にドライブされた後、2 命令クロック後にセットされます。

図 14-3: シングル比較モード：比較一致イベント時  $OCx$  を Low に強制



## 14.3.1.3 トグル出力のシングル比較モード

このモードに出力比較モジュールを構成するには、制御ビット OCM<2:0> = ‘011’を設定します。更にタイマー2またはタイマー3も有効化される必要があります。いったんこの比較モードが有効化されると、出力ピン OCx は最初 Low にドライブされ、タイマーおよび OCxR レジスタの間で発生する一致イベントごとにトグルします。図 14-4 および図 14-5 を参照すると、いくつか注意する必要がある重要なタイミングイベントがあります。

- OCx ピンは、比較用タイマーおよび OCxR レジスタとの間で発生する比較一致後 1 命令クロック後にトグルされます。OCx ピンは次のトグルイベントか、モード変更が行われるか、またはモジュールが無効化されるまでこの新しい状態のままであります。
- 比較用タイマーは対応する周期レジスタの内容までカウントアップします。そして次の命令クロックで 0x0000 へリセットされます。
- 各チャネル割り込みフラグ OCxIF は OCx ピンがトグルされた後、2 命令クロック後にセットされます。

**注:** 内部 OCx ピン出力論理はデバイスリセットの際論理「0」へ設定されます。しかしながら、トグルモードの動作 OCx ピン状態はユーザーのソフトウェアによって設定されます。例 14-1 では、トグル動作モードでの要求する初期 OCx ピン状態を定義するコード例を示しています。

図 14-4: シングル比較モード：比較一致イベント時のトグル出力 (PR2 &gt; OCxR)

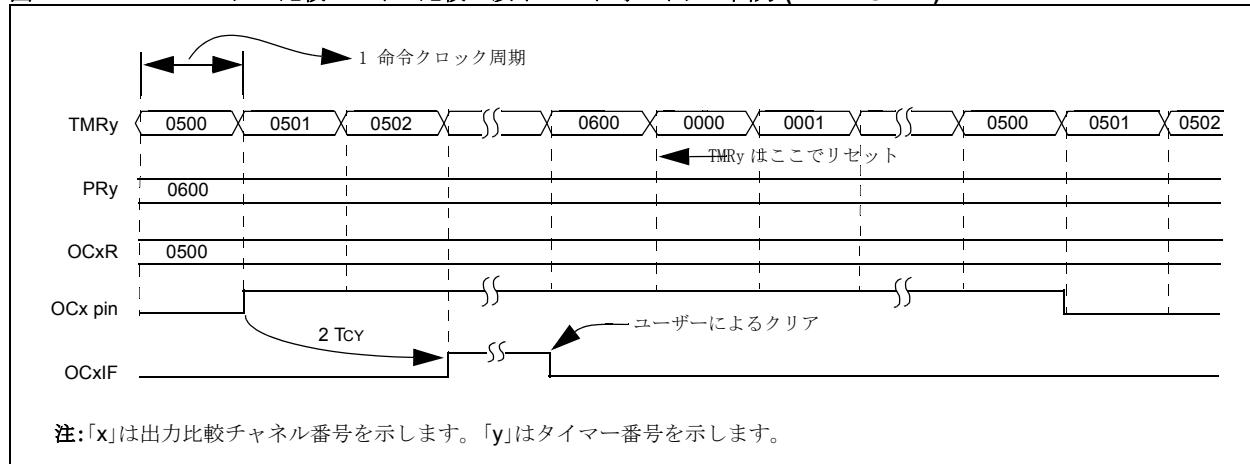
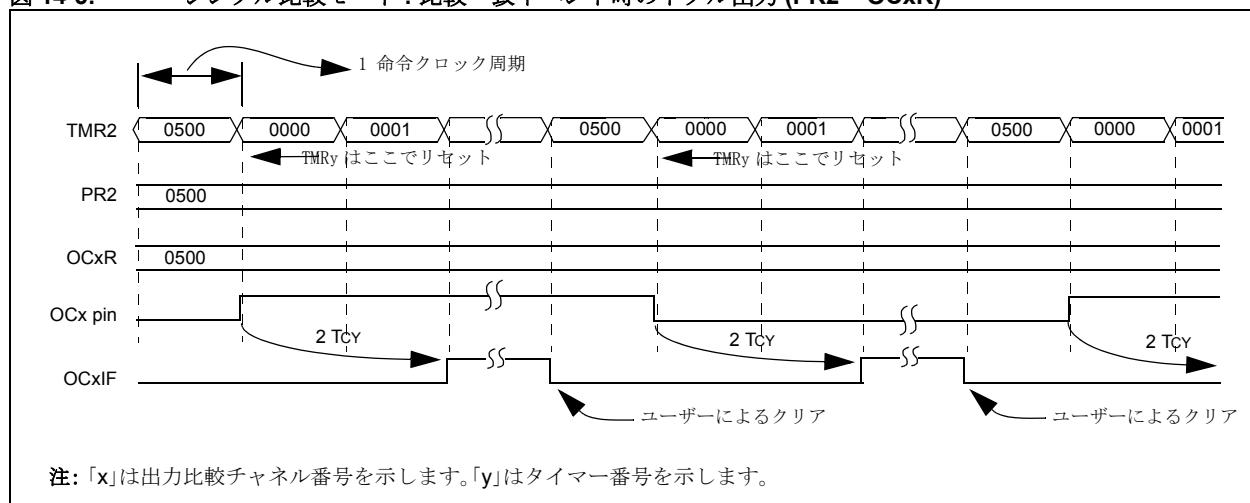


図 14-5: シングル比較モード：比較一致イベント時のトグル出力 (PR2 = OCxR)



## 例 14-1: 比較モードトグルモードピン状態設定

```
; The following code example illustrates how to define the initial
; OC1 pin state for the output compare toggle mode of operation.

; Toggle mode with initial OC1 pin state set low

MOV    0x0001, w0      ; load setup value into w0
MOV    w0, OC1CON      ; enable module for OC1 pin low, toggle high
BSET   OC1CON, #1      ; set module to toggle mode with initial pin
                      ; state low

; Toggle mode with initial OC1 pin state set high

MOV    0x0002, w0      ; load setup value into w0
MOV    w0, OC1CON      ; enable module for OC1 pin high, toggle low
BSET   OC1CON, #0      ; set module to toggle mode with initial pin
                      ; state high
```

例 14-2 ではシングル比較モードトグルイベントの構成および割り込みサービスのコード例を示します。

## 例 14-2: 比較モードトグル設定および割り込みサービス

```
; The following code example will set the Output Compare 1 module
; for interrupts on the toggle event and select Timer 2 as the clock
; source for the compare time-base. It is assumed in that Timer 2
; and Period Register 2 are properly configured. Timer 2 will
; be enabled here.

CLR    OC1CON          ; Turn off Output Compare 1 Module.
MOV    #0x0003, w0      ; Load the working register with the new
MOV    w0, OC1CON        ; compare mode and write to OC1CON
MOV    #0x0500, w0      ; Initialize Compare Register 1
MOV    w0, OC1R          ; with 0x0500
BSET   IPC0, #OC1IP0    ; Setup Output Compare 1 interrupt for
BCLR   IPC0, #OC1IP1    ; desired priority level
BCLR   IPC0, #OC1IP2    ; (this example assigns level 1 priority)
BCLR   IFS0, #OC1IF     ; Clear Output Compare 1 interrupt flag
BSET   IEC0, #OC1IE     ; Enable Output Compare 1 interrupts
BSET   T2CON, #TON      ; Start Timer2 with assumed settings

; Example code for Output Compare 1 ISR:

__OC1Interrupt:
    BCLR   IFS0, #OC1IF    ; Reset respective interrupt flag
                          ; Remaining user code here
    RETFIE
```

### 14.3.2 デュアル比較一致モード

制御ビット OCM<2:0> = ‘100’ または ‘101’ (OCxCON<2:0>) の場合、選択された出力比較チャネルは以下の 2 つのデュアル比較一致モードのうち 1 つで構成されます。

- シングル出力パルスモード
- 連続出力パルスモード

デュアル比較モードでは、比較一致イベント用にモジュールは OCxR および OCxRS レジスタの両方を使用します。OCxR レジスタはカウントアップしているタイマー TMRy と比較され、比較一致イベントの際パルスの OCx ピンを立ち上げます。OCxRS レジスタは同じカウントアップしているタイマー TMRy と比較され、比較一致イベントの際 OCx ピンを立ち下げます。

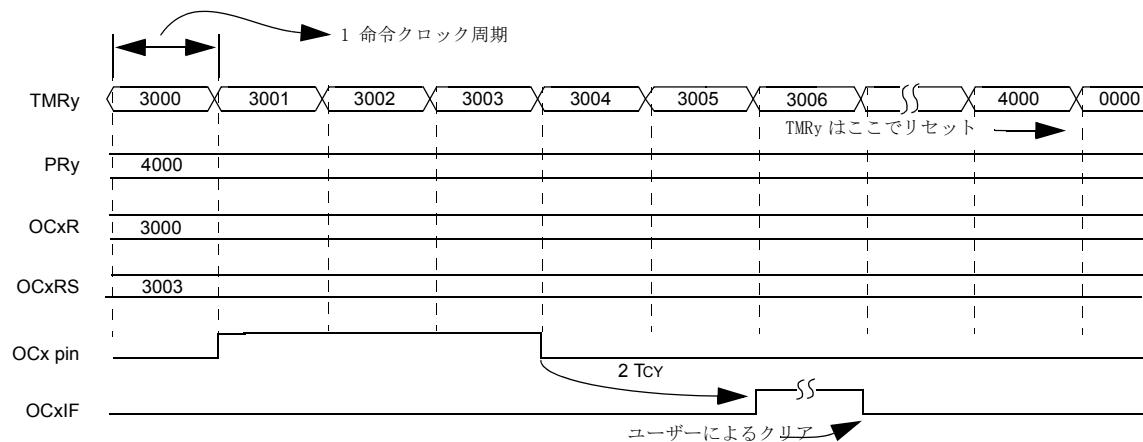
#### 14.3.2.1 デュアル比較モード：シングル出力パルス

出力比較モジュールをシングル出力パルスモード用に構成するには、制御ビット OCM<2:0> = ‘100’ を設定します。更に比較用タイマーを有効にする必要があります。いったんこのモードが有効化されると、出力ピン OCx は Low にドライブされ、タイマーと OCxR レジスタの間で一致が発生するまで Low のままであります。図 14-6 および図 14-7 を参照すると、幾つか注意する重要なタイミングイベントがあります。

- OCx ピンは、比較用タイマーおよび OCxR レジスタとの間で発生する比較一致後 1 命令クロック後に High にドライブされます。OCx ピンは次の一致イベントがタイマーおよび OCxRS レジスタの間で発生するまで High のままであり続けます。この時、イベントで OCx ピンは Low にドライブされます。OCx ピンはモード変更が行われるまでまたはモードが無効化されるまで Low のままであります。
- 比較用タイマーは対応する周期レジスタに含まれる値までカウントアップします。そして次の命令クロックで 0x0000 へリセットされます。
- タイマー周期レジスタの内容が OCxRS レジスタの内容よりも小さい場合、パルスの立ち下がりエッジは生成されません。OCx ピンは OCxRS <= PRy、モード変更、またはリセット条件が発生するまで High のままでいます。
- 各チャネル割り込みフラグ OCxIF は OCx ピンが Low にドライブされた後（シングルパルスの立ち下がりエッジ）、2 命令クロック後にセットされます。

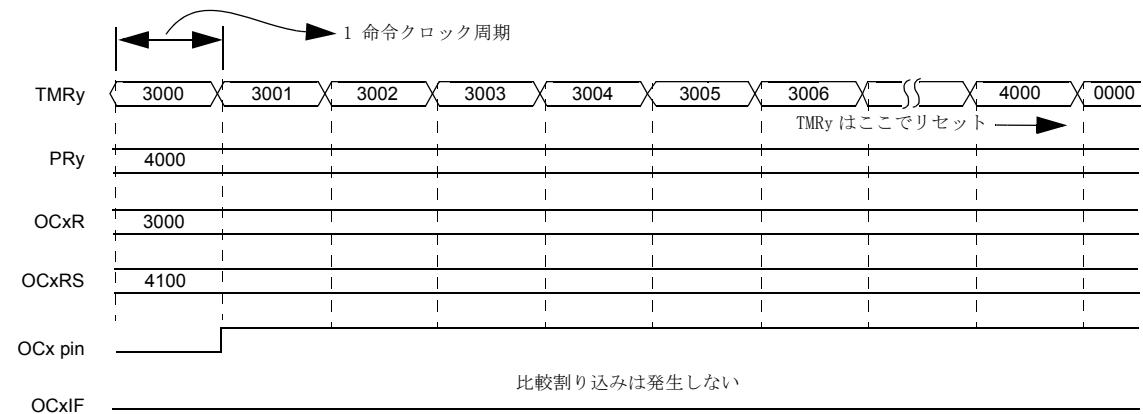
図 14-6 はシングル出力パルスを生成する汎用デュアル比較モードを示します。図 14-7 は OCxRS > PRy の別のタイミング例を示します。この例では、比較用タイマーは 0x4100 までカウントをする前にリセットされるので、パルスの立ち下がりエッジは生成されません。

図 14-6: デュアル比較モード



注 1: 「x」は出力比較チャネル番号を示します。「y」はタイマーパン号を示します。  
2: OCxR = 比較レジスタ、OCxRS = 二次比較レジスタ。

図 14-7: デュアル比較モード：シングル出力パルス (OCxRS > PR2)



注 1: 「x」は出力比較チャネル番号を示します。「y」はタイマーパン号を示します。  
2: OCxR = 比較レジスタ、OCxRS = 二次比較レジスタ。

### 14.3.2.2 シングル出力パルス生成の設定

制御ビット OCM<2:0> (OCxCON<2:0>) が ‘100’ に設定された場合、選択された出力比較チャネルは OCx ピンを Low 状態へ初期化し、シングル出力パルスを生成します。

シングル出力パルスを生成するために、以下のステップが必要です（これらのステップはタイマーソースが最初 OFF 正していると想定していますが、これはモジュール動作の要件ではありません）。

1. 命令クロックサイクルタイムを決定します。外部クロックからタイマーソースへの周波数（使用されている場合）およびタイマーピリオドスケーラ設定を考慮します。
2. TMRy 開始値 (0x0000) からの出力パルスの立ち上がりエッジまでの時間を算出します。
3. 希望するパルス幅およびパルスの立ち上がりエッジからの時間に基づいてパルスの立ち下がりエッジまでの時間を算出します。
4. 上記ステップ 2 および 3 にて算出した値を比較レジスタ OCxR および二次比較レジスタ OCxRS へそれぞれ書き込みます。
5. タイマーピリオドレジスタ PRy を OCxRS、二次比較レジスタに等しいまたはそれよりも大きい値に設定します。
6. OCM<2:0> = ‘100’ および OCTSEL (OCxCON<3>) ビットを希望するタイマー選択となるように設定します。OCx ピン状態はこれで Low にドライブされます。
7. TON (TyCON<15>) ビットを ‘1’ に設定します、これで比較用タイマーのカウントを開始します。
8. TMRy および OCxR 間の最初の一一致の際に、OCx ピンは High にドライブされます。
9. カウントアップしているタイマー TMRy が二次比較レジスタ OCxRS に一致した場合、OCx ピンは 2 回目のエッジ変化 (High から Low) がドライブされます。OCx ピンにはこれ以上パルスはドライブされず、Low のままでとどまります。二次比較一致イベントの結果として、OCxIF 割り込みフラグビットがセットされます、これにより OCxIE ビットがセットされて、有効化されている場合は割り込みが発生します。周辺装置の割り込みについて詳細は第 6 章 . 「割り込み」を参照してください。
10. 別のシングルパルス出力を開始するには、必要であれば、タイマーおよび比較レジスタ設定を変更してから、OCM<2:0> (OCxCON<2:0>) ビットを ‘100’ へ設定する書き込み実行します。タイマーの無効化および再有効化、そして TMRy レジスタのクリアは必須ではありませんが既知のイベント時間境界からパルスを定義する方が有利です。

出力モジュールは出力パルスの立ち下りエッジの後無効化される必要はありません。別のパルスが OCxCON レジスタの値を再度書き込むことで開始されます。

例 14-3 ではシングル出力パルスイベントの構成用コード例を示しています。

### 例 14-3: シングル出力パルス設定および割り込みサービス

```
; The following code example will set the Output Compare 1 module
; for interrupts on the single pulse event and select Timer 2
; as the clock source for the compare time base. It is assumed
; that Timer 2 and Period Register 2 are properly initialized.
; Timer 2 will be enabled here.

CLR    OC1CON           ; Turn off Output Compare 1 Module.
MOV    #0x0004, w0        ; Load the working register with the new
MOV    W0, OC1CON         ; compare mode and write to OC1CON
MOV    #0x3000, w0        ; Initialize Compare Register 1
MOV    W0, OC1R           ; with 0x3000
MOV    #0x3003, w0        ; Initialize Secondary Compare Register 1
MOV    W0, OC1RS          ; with 0x3003
BSET   IPC0, #OC1IP0      ; Setup Output Compare 1 interrupt for
BCLR   IPC0, #OC1IP1      ; desired priority level
BCLR   IPC0, #OC1IP2      ; (this example assigns level 1 priority)
BCLR   IFS0, #OC1IF       ; Clear Output Compare 1 interrupt flag
BSET   IEC0, #OC1IE       ; Enable Output Compare 1 interrupts

BSET   T2CON, #TON        ; Start Timer2 with assumed settings

; Example code for Output Compare 1 ISR:

__OC1Interrupt:
    BCLR   IFS0, #OC1IF      ; Reset respective interrupt flag
                            ; Remaining user code here
    RETFIE                      ; Return from ISR
```

## 14.3.2.3 シングル出力パルスを生成するデュアル比較モードの特例

OCxR、OCxRS、および PRy 値の関係により、出力比較モジュールには幾つか説明が必要な独自な条件があります。これらの特例は表 14-1 にて結果的に生じるモジュールの動作と共に説明されています。

表 14-1: シングル出力パルスを生成するデュアル比較モードの特例

SFR 論理関係	特別条件	操作	OCx での出力
PRy >= OCxRS および OCxRS > OCxR	OCxR = 0 が TMRy = 0 を初期化	TMRy の 0x0000 から PRy までのカウントの最初の反復にて、OCx ピンは Low のままでとどまり、パルスは生成されません。TMRy がゼロへリセットした後（周期一致により）、OCx ピンは OCxR との一致により High になります。次の TMRy から OCxRS の一致の際には、OCx ピンは Low になります。そのままとどまります。OCxIF ビットは二回目の比較の結果としてセットされます。考慮が必要な 2 つの選択可能な初期条件があります。 a] TMRy = 0 に初期化し OCxR >= 1 に設定する b] TMRy = PRy (PRy > 0) に初期化し OCxR = 0 に設定する。	設定された、PRy レジスタの値によってパルスは遅延される
PRy >= OCxR および OCxR >= OCxRS	OCxR >= 1 および PRy >= 1	TMRy は OCxR までカウントし、比較一致イベントの際（つまり TMRy = OCxR）、OCx ピンは High 状態へとドライブされます。TMRy はそれからカウントを継続し、最終的にはピリオド一致の際（つまり PRy = TMRy）にリセットします。タイマーは 0x0000 からリスタートし、OCxRS までカウントします。それから比較一致イベント（つまり TMRy = OCxRS）の際に OCx ピンは Low 状態へとドライブされます。OCxIF ビットは二度目の比較の結果としてセットされます。	パルス
OCxRS > PRy および PRy >= OCxR	なし	OCx ピンでは、立ち上がりエッジのみが生成されます。OCxIF はセットされません。	立ち上がりエッジ /High への遷移
OCxR = OCxRS = PRy = 0x0000	なし	出力パルスはタイマーの一一致の際 2 命令クロック間遅延してから OCx ピンに出力されます。OCxIF ビットは二回目の比較の結果としてセットされます。	遅延パルス
OCxR > PRy	なし	サポートされていないモードです。タイマーは一致条件以前にリセットします。	Low 状態のまま

注 1: ここで考慮された全ての場合にて TMRy レジスタは 0x0000 へ初期化されていると想定します。

2: OCxR = 比較レジスタ、OCxRS = 二次比較レジスタ、TMRy = Timery カウント、PRy = Timery 周期レジスタ。

## 14.3.2.4 デュアル比較モード：連続出力パルス

このモードに出力比較モジュールを構成するには、制御ビット OCM<2:0> = ‘101’を設定します。更に比較用タイマーを選択して有効にする必要があります。いったんこのモードが有効化されると、出力ピン OCx は最初 Low にドライブされ、比較用タイマーと OCxR レジスタの間で一致が発生するまで Low のままでいます。図 14-8 および図 14.3.2.5 を参照すると、幾つか注意が必要なタイミングイベントがあります。

- OCx ピンは、比較用タイマーと OCxR レジスタとの間で発生する比較一致後 1 命令クロックで High にドライブされます。OCx ピンは次の一致イベントがタイマーと OCxRS レジスタの間で発生するまで High のままです、一致した時ピンは Low にドライブされます。Low から High、High から Low というパルス生成シーケンスはこの後ユーザーが何もしなくとも継続して OCx ピン出力されます。
- モード変更が行われる、またはモジュールが無効化されるまで連続するパルスが OCx ピンに出力されます。
- 比較用タイマーは対応する周期レジスタの内容までカウントアップします。そして次の命令クロックで 0x0000 へリセットします。
- 比較用タイマー周期レジスタ値が OCxRS レジスタ値よりも小さい場合、立ち下りエッジは発生しません。OCx ピンは OCxRS <= PR2、モード変更、デバイスがリセットされるまで High のままでいます。
- 各チャネル割り込みフラグ OCxIF は OCx ピンが Low にドライブされた後（シングルパルスの立ち下りエッジ）、2 命令クロック後にセットされます。

図 14-8 では、連続する出力パルスを生成している一般デュアル比較モードを示しています。図 14-9 では、OCxRS > PRy である別のタイミング例を示しています。この例では、タイマーが OCxRS の内容までカウントされる前にリセットされるので、パルスの立ち下りエッジは生成されません。

図 14-8: デュアル比較モード：連続出力パルス (PR2 =&lt; OCxRS)

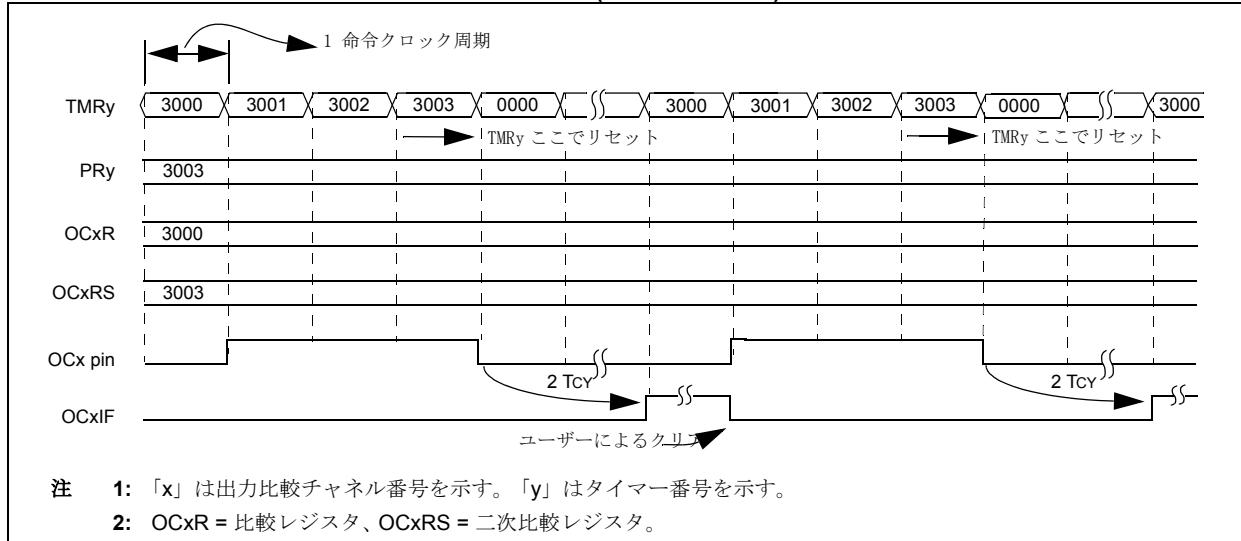
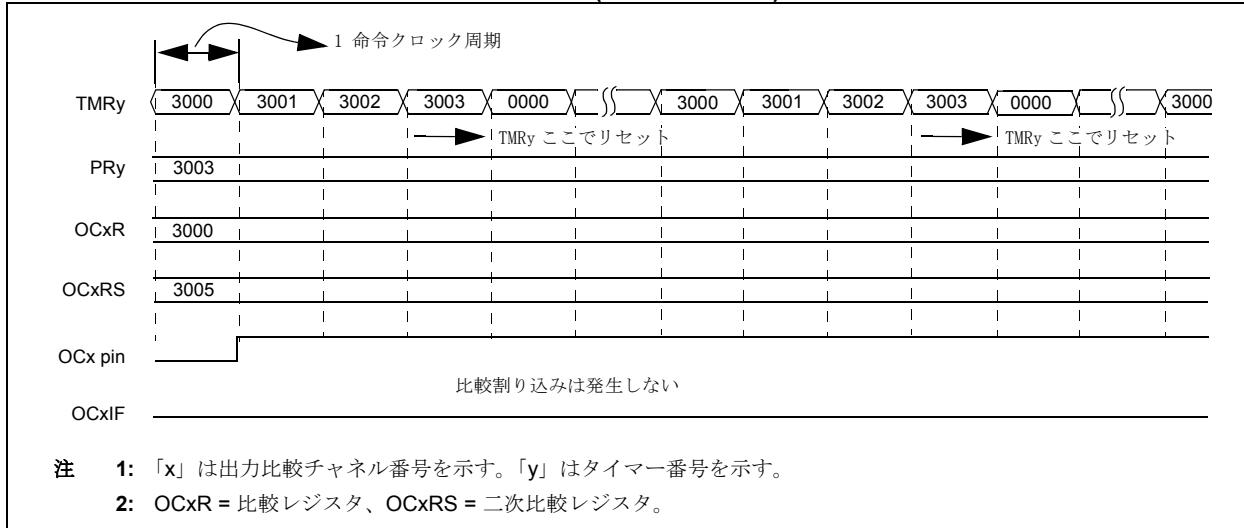


図 14-9: デュアル比較モード: 連続出力パルス (PR2 &lt;&gt; OCxRS)



#### 14.3.2.5 連続出力パルス生成の設定手順

制御ビット OCM<2:0> (OCxCON<2:0>) が ‘101’ に設定された場合、選択された出力比較チャネルは OCx ピンを Low 状態へ初期化し比較一致イベントごとに、出力パルスを生成します。

ユーザーが出力パルスの連続するストリームを生成するようにモジュールを構成するには、以下のステップが必要です(これらのステップはタイマーソースが最初 OFF されていると想定していますが、これはモジュール動作の要件ではありません)。

1. 命令クロックサイクル時間を決定します。外部クロックからタイマーソースへの周波数(使用されている場合)およびタイマープリスケーラ設定を考慮します。
2. TMRy 開始値(0x0000)からの出力パルスの立ち上がりエッジまでの時間を算出します。
3. 希望するパルス幅およびパルスの立ち上がりエッジからの時間に基づいてパルスの立ち下がりエッジまでの時間を算出します。
4. 上記ステップ 2 および 3 にて算出した値を比較レジスタ OCxR および二次比較レジスタ OCxRS へそれぞれ書き込みます。
5. タイマーピリオドレジスタ PRy を OCxRS、二次比較レジスタに等しいまたはそれよりも大きい値に設定します。
6. OCM<2:0> = ‘101’ および OCTSEL (OCxCON<3>) ビットを希望するタイマー選択となるように設定します。OCx ピン状態はこれで Low にドライブされます。
7. TON (TyCON<15>) ビットを ‘1’ に設定することで比較用タイマーを開始します。
8. TMRy および OCxR 間の最初の一一致の際に、OCx ピンは High にドライブされます。
9. 比較用タイマー TMRy が二次比較レジスタ OCxRS に一致した場合、OCx ピンは 2 回目のエッジ変化 (High から Low) が output されます。
10. 二次比較一致イベントの結果として、OCxIF 割り込みフラグビットがセットされます。
11. 比較用タイマーと対応する周期レジスタの値が一致する場合、TMRy レジスタは 0x0000 へとリセットし、カウントを再起動します。
12. ステップ 8 から 11 までは繰り返され、パルスの連続ストリームが永久に生成されます。OCxIF フラグが OCxRS-TMRy 比較一致イベントの際にセットされます。

例 14-4 では、連続出力パルスイベント用の構成のコード例を示しています。

## 例 14-4: 連続出力パルス設定および割り込みサービス

```
; The following code example will set the Output Compare 1 module
; for interrupts on the continuous pulse event and select Timer 2
; as the clock source for the compare time-base. It is assumed
; that Timer 2 and Period Register 2 are properly configured.
; Timer 2 will be enabled here.

CLR    OC1CON           ; Turn off Output Compare 1 Module.
MOV    #0x0005, W0        ; Load the working register with the new
MOV    W0, OC1CON         ; compare mode and write to OC1CON
MOV    #0x3000, W0        ; Initialize Compare Register 1
MOV    W0, OC1R           ; with 0x3000
MOV    #0x3003, W0        ; Initialize Secondary Compare Register 1
MOV    W0, OC1RS          ; with 0x3003
BSET   IPC0, #OC1IP0      ; Setup Output Compare 1 interrupt for
BCLR   IPC0, #OC1IP1      ; desired priority level
BCLR   IPC0, #OC1IP2      ; (this example assigns level 1 priority)
BCLR   IFS0, #OC1IF       ; Clear Output Compare 1 interrupt flag
BSET   IEC0, #OC1IE       ; Enable Output Compare 1 interrupts

BSET   T2CON, #TON        ; Start Timer2 with assumed settings

; Example code for Output Compare 1 ISR:

__OC1Interrupt:
    BCLR   IFS0, #OC1IF      ; Reset respective interrupt flag
                            ; Remaining user code here
    RETFIE                      ; Return from ISR
```

## 14.3.2.6 連続出力パルスを生成するデュアル比較モードの特例

OCxR、OCxRS、および PRy 値の関係により、出力比較モジュールは想定した結果を生まないことがあります。これらの特例は表 14-2 にて結果的に生じるモジュールの動作と共に説明しています。

表 14-2: 連続出力パルスを生成するデュアル比較モードの特例

SFR 論理関係	特別条件	操作	OCx での出力
PRy >= OCxRS および OCxRS > OCxR	OCxR = 0 が TMRy = 0 を初期化	TMRy の 0x0000 から PRy までのカウントの最初の反復にて、OCx ピンは Low のままでとどまり、パルスは生成されません。TMRy がゼロへリセットした後（周期一致の際）、OCx ピンは High になります。次の TMRy から OCxRS の一致の際には、OCx ピンは Low になります。OCxR = 0 および PRy = OCxRS の場合、ピンは 1 クロックサイクル間は Low のままで、それから次の TMRy から OCxRS の一致まで High にドライブされます。OCxIF ビットは二回目の比較の結果としてセットされます。考慮が必要な別の 2 つの選択可能な初期条件があります。 a] TMRy = 0 に初期化し OCxR >= 1 に設定する b] TMRy = PRy (PRy > 0) に初期化し OCxR = 0 に設定する。	設定された、PRy レジスタの値によって最初だけ遅延された連続パルス
PRy >= OCxR および OCxR >= OCxRS	OCxR >= 1 および PRy >= 1	TMRy は OCxR までカウントし、比較一致イベントの際（つまり TMRy = OCxR）、OCx ピンは High 状態へとドライブされます。TMRy はそれからカウントを継続し、最終的にはビリオド一致の際（つまり PRy = TMRy）にリセットします。タイマーは 0x0000 からリスタートし、OCxRS までカウントします。それから比較一致イベント（つまり TMRy = OCxR）の際に OCx ピンは Low 状態へとドライブされます。OCxIF ビットは二度目の比較の結果としてセット定されます。	連続パルス
OCxRS > PRy および PRy >= OCxR	なし	OCxRS レジスタの内容が周期レジスタ内容 (PRy) より少ないか等しい値へと変更されるまで OCx ピンにて 1 つの遷移のみ生成されます。OCxIF はそれまでセットされません。	立ち上がりエッジ /High への遷移
OCxR = OCxRS = PRy = 0x0000	なし	連続する出力パルスが OCx ピンに出力されます。最初のパルスはタイマーおよび周期レジスタの一致の際に 2 つの命令クロック周期間ディレーされます。OCxIF ビットは二度目の比較の結果としてセットされます。	最初のパルスは遅延される。連続パルスが生成される。
OCxR > PRy	なし	サポートされていないモードです。タイマーは一致条件以前にリセットします。	Low 状態のまま

注 1: ここで考慮された全ての場合にて TMRy レジスタは 0x0000 へ初期化されていると想定します。

2: OCxR = 比較レジスタ、OCxRS = 二次比較レジスタ、TMRy = Timery カウント、PRy = Timery 周期レジスタ。

### 14.3.3 パルス幅変調モード

制御ビット  $OCM<2:0>$  ( $OCxCON<2:0>$ ) が ‘110’ または ‘111’ に設定された場合、選択された出力比較チャネルは PWM (パルス幅変調) 動作モードに構成されます。

以下の 2 つの PWM モードが可能です。

- PWM フォールトプロテクション入力なし
- PWM フォールトプロテクション入力付き

OCFA または OCFB FAULT 入力ピンは 2 つ目の PWM モード用に使用されます。このモードでは、 $OCFx$  ピン入力に論理「0」が入力されると非同期に選択した PWM チャネルの運転を停止します (セクション 14.3.3.1 「PWM フォールトプロテクション入力ピン」参照)。

PWM モードでは、 $OCxR$  レジスタは読み取り専用スレーブのデューティレジスタであり、 $OCxRS$  はユーザーによって PWM デューティの更新用に書き込まれるバッファレジスタです。タイマーと周期ドレジスタの一一致イベント (PWM ピリオドの終端) ごとに、デューティレジスタ  $OCxR$  には  $OCxRS$  の内容がロードされます。TyIF 割り込みフラグは各 PWM ピリオド境界でセットされます。

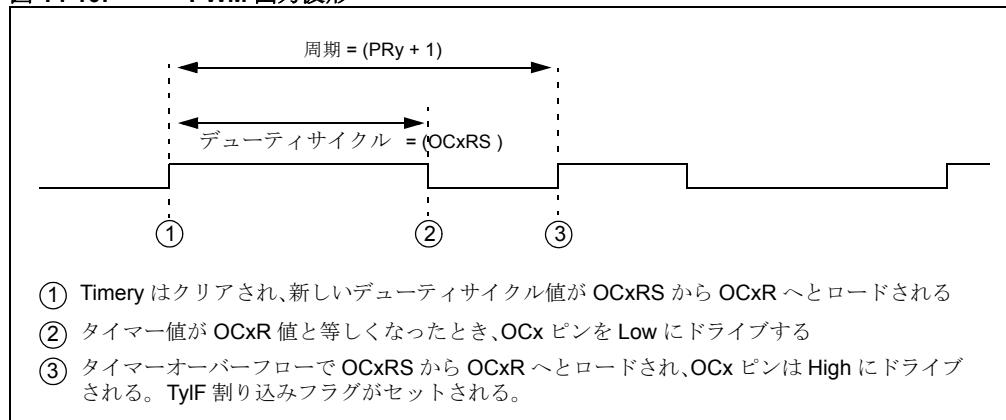
以下のステップは出力比較モジュールを PWM 動作用に構成する場合に使用します。

1. 選択した周期レジスタ ( $PRy$ ) へ書き込むことで PWM ピリオドを設定します。
2.  $OCxRS$  レジスタへ書き込むことで PWM デューティを設定します。
3. 最初のデューティサイクルで  $OCxR$  レジスタに書き込みます。
4. 必要な場合、タイマーおよび出力比較モジュール用に割り込みを有効化します。出力比較割り込みは PWM FAULT ピンを使うときには必要です。
5. 出力比較モードビット  $OCM<2:0>$  ( $OCxCON<2:0>$ ) へ書き込むことで 2 つの PWM 動作モードのうち 1 つの出力比較モジュールを構成します。
6.  $TMRy$  プリスケール値を設定し、 $TON$  ( $TxCON<15>$ ) = 1 を設定することでタイマーを有効化します。

**注:**  $OCxR$  レジスタは出力比較モジュールが最初に有効化される前に初期化する必要があります。 $OCxR$  レジスタはモジュールが PWM モードで動作する場合、読み取り専用デューティサイクルレジスタになります。 $OCxR$  に保持された値は、最初の PWM ピリオドの PWM デューティ値となります。デューティサイクルバッファレジスタ  $OCxRS$  の内容はタイマー周期一致が発生するまで  $OCxR$  へは転送されません。

PWM 出力波形の例は図 14-10 にて示されています。

図 14-10: PWM 出力波形



### 14.3.3.1 PWM フォールトプロテクション入力ピン

出力比較モードビット  $OCM<2:0>$  ( $OCxCON<2:0>$ ) が ‘111’ に設定された場合、選択された出力比較チャネルは PWM 動作モード用に構成されます。セクション 14.3.3 「パルス幅変調モード」で説明される全ての機能に入力 FAULT プロテクション付きを適用します。

FAULT プロテクションは OCFA および OCFB ピン経由で提供されます。OCFB ピンが出力比較チャネルの 5 から 8 までに関連している一方で、OCFA ピンは出力比較チャネルの 1 から 4 に関連しています。

論理「0」が OCFA/OCFB ピンで検出された場合、選択された PWM 出力ピンはハイインピーデンス状態にされます。ユーザーは FAULT 条件が発生した場合、必要な状態を提供するために PWM ピンをプルダウンかプルアップ抵抗のいずれかを選択することができます。PWM 出力の停止は即時のもので、デバイスのクロックには関係していません。この状態は以下の状態まで続きます。

- 外部 FAULT 条件が取り除かれる、および
- PWM モードが適切なモードビット  $OCM<2:0>$  ( $OCxCON<2:0>$ ) で再度有効化される。

FAULT 条件の結果、各割り込みフラグ  $OCxIF$  ビットがセットされ、割り込みが有効化されている場合、割り込みを発生します。FAULT 条件の検出の際、 $OCFLT$  ビット ( $OCxFaultCon<4>$ ) が High にセットされます(論理「1」)。このビットは読み取り専用ビットで、外部 FAULT 条件が取り除かれ、かつ適切なモードビット  $OCM<2:0>$  ( $OCxCON<2:0>$ ) がセットされて、PWM モードが再度有効化されると、クリアされます。

**注:** 外部 FAULT ピンは使用が有効化されると、デバイスが SLEEP または IDLE モードにある間も、 $OCx$  出力ピンを操作し続けます。

### 14.3.3.2 PWM 周期

PWM 周期は PRy、Timery 周期レジスタへの設定により指定されます。PWM 周期は以下の式を使用して算出可能です。

式 14-1: PWM 周期の算出

$$\text{PWM 周期} = [(PRy) + 1] \cdot TCY \cdot (\text{TMRY プリスケール値})$$

$$\text{PWM 周波数} = 1 / [\text{PWM 周期}]$$

**注:** N の PRy 値は N+1 タイムベースサイクルの PWM 周期を生成します。例: PRy レジスタに書き込まれた 7 の値は 8 つのタイムベースサイクルを含む周期を生成します。

### 14.3.3.3 PWM デューティサイクル

PWM デューティサイクルは OCxRS レジスタへ設定することで指定されます。OCxRS レジスタはいつでも書き込みが可能ですが、一致が PRy および TMRy 間で発生するまで（つまり、周期の完了）デューティサイクル値は OCxR へラッ奇されません。これは、PWM デューティサイクルにダブルバッファを提供しグリッチの少ない PWM 動作を実現します。PWM モードでは、OCxR は読み取り専用レジスタです。

PWM デューティには以下のような重要境界パラメータがあります。

- デューティレジスタ OCxR が 0x0000 でロードされた場合、OCx ピンは Low のままであります (0% デューティサイクル)。
- OCxR が PRy (タイマー周期レジスタ) よりも大きい場合、ピンは High のままであります (100% デューティサイクル)。
- OCxR が PRy に等しい場合、OCx ピンは 1 タイムベースカウント値だけ Low、他の全てのカウント値では High となります。

PWM モードタイミングの詳細については図 14-11 を参照してください。表 14-3 および表 14-4 ではそれぞれ 10 および 30 MIPS でのデバイス動作時の PWM 周波数および分解能の例を示しています。

式 14-2: 最大 PWM 分解能用算出

$$\text{最大 PWM 分解能(ビット)} = \frac{\log_{10}\left(\frac{\text{FOSC}}{\text{FPWM}}\right)}{\log_{10}(2)} \text{ ビット}$$

例 14-5: PWM 周期とデューティサイクル分解能の計算方法

希望 PWM 周波数は 52.08 kHz,  
FOSC = 10 MHz with x4 PLL (40 MHz device clock rate) (TCY = 4/FOSC)  
タイマー 2 プリスケール設定 : 1:1

$$\begin{aligned} 1/52.08 \text{ kHz} &= (\text{PR2}+1) \cdot \text{TCY} \cdot (\text{タイマー 2 プリスケール値}) \\ 19.20 \mu\text{s} &= (\text{PR2}+1) \cdot 0.1\mu\text{s} \cdot (1) \\ \text{PR2} &= 191 \end{aligned}$$

52.08 kHz 周波数および 40 MHz デバイスクロックで使用できるデューティの最大分解能を見つける。

$$\begin{aligned} 1/52.08 \text{ kHz} &= 2^{\text{PWM RESOLUTION}} \cdot 1/40 \text{ MHz} \cdot 1 \\ 19.20 \mu\text{s} &= 2^{\text{PWM RESOLUTION}} \cdot 25 \text{ ns} \cdot 1 \\ 768 &= 2^{\text{PWM RESOLUTION}} \\ \log_{10}(768) &= (\text{PWM 分解能}) \cdot \log_{10}(2) \\ \text{PWM 分解能} &= 9.5 \text{ ビット} \end{aligned}$$

図 14-11: PWM 出力タイミング

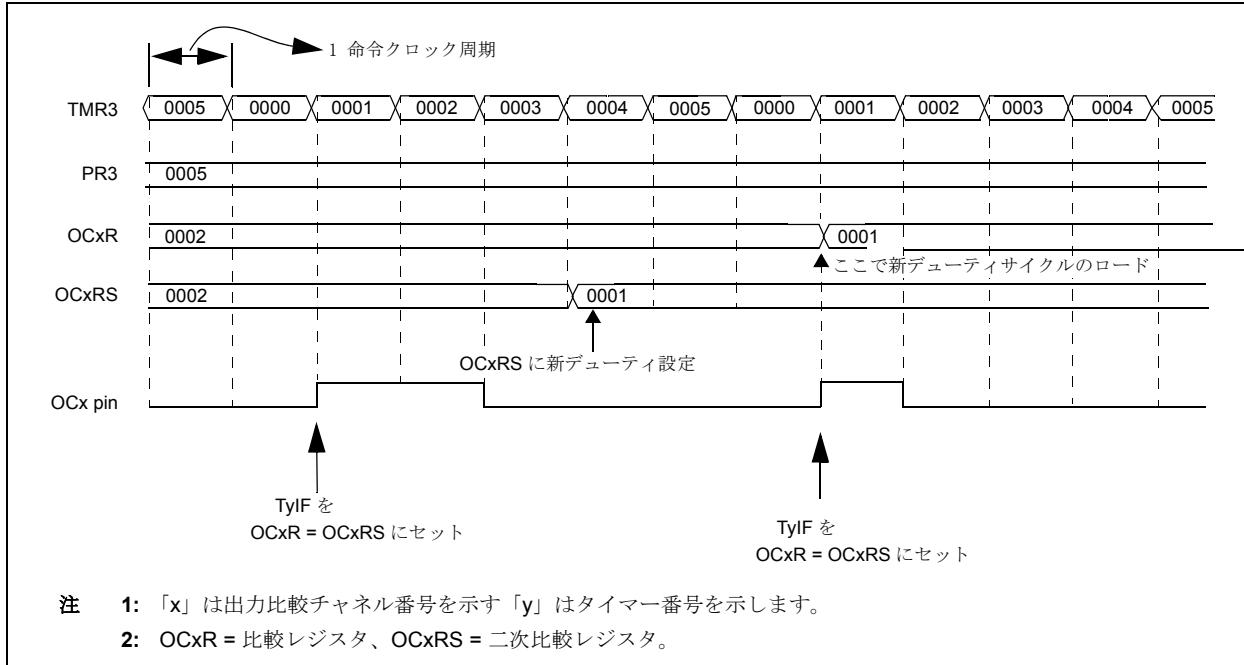


表 14-3: 10 MIPS (Fosc = 40 MHz) での PWM 周波数および分解能例

PWM 周波数	19 Hz	153 Hz	305 Hz	2.44 kHz	9.77 kHz	78.1 kHz	313 kHz
タイマーブリスケーラ率	8	1	1	1	1	1	1
周期レジスタ値	0xFFFF	0xFFFF	0x7FFF	0x0FFF	0x03FF	0x007F	0x001F
分解能 (ビット)	16	16	15	12	10	7	5

表 14-4: 30 MIPS (Fosc = 120 MHz) での PWM 周波数および分解能例

PWM 周波数	57 Hz	458 Hz	916 Hz	7.32 kHz	29.3 kHz	234 kHz	938 kHz
タイマーブリスケーラ率	8	1	1	1	1	1	1
周期レジスタ値	0xFFFF	0xFFFF	0x7FFF	0x0FFF	0x03FF	0x007F	0x001F
分解能 (ビット)	16	16	15	12	10	7	5

例 14-6 では、PWM 動作モードの構成および割り込みサービスコードを示しています。

## 例 14-6: PWM モードパルス設定および割り込みサービス

```
; The following code example will set the Output Compare 1 module
; for PWM mode w/o FAULT pin enabled, a 50% duty cycle and a
; PWM frequency of 52.08 kHz at Fosc = 40 MHz. Timer2 is selected as
; the clock for the PWM time base and Timer2 interrupts
; are enabled.

CLR    OC1CON           ; Turn off Output Compare 1 Module.

MOV    #0x0060, w0        ; Initialize Duty Cycle to 0x0060
MOV    w0, OC1RS          ; Write duty cycle buffer register
MOV    w0, OC1R           ; Write OC1R to initial duty cycle value

MOV    #0x0006, w0        ; Load the working register with the new
MOV    w0, OC1CON          ; compare mode and write to OC1CON
MOV    #0x00BF w0          ; Initialize PR2 with 0x00BF
MOV    w0, PR2             ; ;

BSET   IPC0, #T2IP0       ; Setup Timer 2 interrupt for
BCLR   IPC0, #T2IP1       ; desired priority level
BCLR   IPC0, #T2IP2       ; (this example assigns level 1 priority)
BCLR   IFS0, #T21IF        ; Clear Timer 2 interrupt flag
BSET   IEC0, #T21IE        ; Enable Timer 2 interrupts
BSET   T2CON, #TON         ; Start Timer2 with assumed settings

; Example code for Timer 2 ISR:

__T2Interrupt:
    BCLR   IFS0, #T21IF      ; Reset respective interrupt flag
                                ; Remaining user code here
    RETFIE                      ; Return from ISR
```

## 14.4 省電力状態での出力比較動作

### 14.4.1 SLEEP モードでの出力比較動作

デバイスが SLEEP モードに入る場合、システムクロックは無効化されます。SLEEP の間は、出力比較チャネルはそのピンを SLEEP に入る前にドライブされたのと同じアクティブ状態でドライブします。モジュールはそれからこの状態で停止します。

例えば、そのピンが High であり、CPU が SLEEP 状態に入った場合、ピンは High のままでいます。同様に、そのピンが Low で、CPU が SLEEP 状態に入った場合、ピンは Low のままでいます。いずれでもウェイクアップした場合には、出力比較モジュールは動作を再開します。

### 14.4.2 IDLE モードでの出力比較動作

デバイスが IDLE モードに入る場合、システムクロックは有効なままで CPU はコード実行を停止します。OCSIDL ビット (OCxCON<13>) はキャプチャモジュールが IDLE モードで停止するのかまたは IDLE モードで動作を継続するのかどうか選択します。

- OCSIDL = 1 の場合、モジュールは IDLE モードで動作を停止します。モジュールは IDLE モードで停止を選択された場合 (OCSIDL = 1)、SLEEP モードと同じ手順を実行します。
- OCSIDL = 0 の場合、モジュールは選択されたタイマーが IDLE モードで動作するよう設定されている場合のみ IDLE モードで動作を継続します。出力比較チャネルは OCSIDL ビットが論理「0」の場合 CPU が IDLE モードでも動作します。更に、タイマーの対応する TxSIDL ビットを 0 にすることで有効化される必要があります。

**注:** 外部 FAULT ピン使用が有効化されている場合、デバイスが SLEEP または IDLE モードにいる限り、関連 OCx 出力ピンを制御し続けます。

## 14.5 入出力ピン制御

出力比較モジュールが有効化された場合、I/O ピン方向は比較モジュールによって制御されます。比較モジュールが無効化された場合、対応する LAT および TRIS 制御ビットへ I/O ピン制御を戻します。

フォールトプロテクション入力モードの PWM が有効化された場合、OCFx FAULT ピンは各 TRIS SFR ビットを設定することで、入力に設定しなければなりません。この特別な PWM モードに設定しても OCFx FAULT ピンは入力として自動設定されません。

**表 14-5: 出力比較モジュール 1-8 の関連ピン**

ピン名称	ピンタイプ	バッファタイプ	説明
OC1	O	—	出力比較 /PWM チャネル 1
OC2	O	—	出力比較 /PWM チャネル 2
OC3	O	—	出力比較 /PWM チャネル 3
OC4	O	—	出力比較 /PWM チャネル 4
OC5	O	—	出力比較 /PWM チャネル 5
OC6	O	—	出力比較 /PWM チャネル 6
OC7	O	—	出力比較 /PWM チャネル 7
OC8	O	—	出力比較 /PWM チャネル 8
OCFA	I	ST	PWM フォールトプロテクション A 入力 (チャネル 1-4 用)
OCFB	I	ST	PWM フォールトプロテクション B 入力 (チャネル 5-8 用)

凡例: ST = CMOS レベルのショミットトリガー入力)、I = 入力、O = 出力

表 14-6: 出力比較モジュールに関連したレジスタマップ例

SFR 名称	Addr.	ビット 15	ビット 14	ビット 13	ビット 12	ビット 11	ビット 10	ビット 9	ビット 8	ビット 7	ビット 6	ビット 5	ビット 4	ビット 3	ビット 2	ビット 1	ビット 0	リセット状態				
TMR2	0106	タイマー 2 レジスタ																0000 0000 0000 0000				
TMR3	010A	タイマー 3 レジスタ																0000 0000 0000 0000				
PR2	010C	周期レジスタ 2																1111 1111 1111 1111				
PR3	010E	周期レジスタ 3																1111 1111 1111 1111				
T2CON	0110	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T32	—	TCS	—	0000 0000 0000 0000				
T3CON	0112	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	ム	—	TCS	—	0000 0000 0000 0000				
OC1RS	0180	出力比較 1 二次レジスタ																uuuu uuuu uuuu uuuu				
OC1R	0182	出力比較 1 ジスタ																uuuu uuuu uuuu uuuu				
OC1CON	0184	—	—	OCSIDL	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>				0000 0000 0000 0000				
OC2RS	0186	出力比較 2 二次レジスタ																uuuu uuuu uuuu uuuu				
OC2R	0188	出力比較 2 レジスタ																uuuu uuuu uuuu uuuu				
OC2CON	018A	—	—	OCSIDL	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>				0000 0000 0000 0000				
OC3RS	018C	出力比較 3 二次レジスタ																uuuu uuuu uuuu uuuu				
OC3R	018E	出力比較 3 レジスタ																uuuu uuuu uuuu uuuu				
OC3CON	0190	—	—	OCSIDL	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>				0000 0000 0000 0000				
OC4RS	0192	出力比較 4 二次レジスタ																uuuu uuuu uuuu uuuu				
OC4R	0194	出力比較 4 レジスタ																uuuu uuuu uuuu uuuu				
OC4CON	0196	—	—	OCSIDL	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>				0000 0000 0000 0000				
OC5RS	0198	出力比較 5 二次レジスタ																uuuu uuuu uuuu uuuu				
OC5R	019A	出力比較 5 レジスタ																uuuu uuuu uuuu uuuu				
OC5CON	019C	—	—	OCSIDL	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>				0000 0000 0000 0000				
OC6RS	019E	出力比較 6 二次レジスタ																uuuu uuuu uuuu uuuu				
OC6R	01A0	出力比較 6 レジスタ																uuuu uuuu uuuu uuuu				
OC6CON	01A2	—	—	OCSIDL	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>				0000 0000 0000 0000				
OC7RS	01A4	出力比較 7 二次レジスタ																uuuu uuuu uuuu uuuu				
OC7R	01A6	出力比較 7 レジスタ																uuuu uuuu uuuu uuuu				
OC7CON	01A8	—	—	OCSIDL	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>				0000 0000 0000 0000				
OC8RS	01AA	出力比較 8 二次レジスタ																uuuu uuuu uuuu uuuu				
OC8R	01AC	出力比較 8 レジスタ																uuuu uuuu uuuu uuuu				
OC8CON	01AE	—	—	OCSIDL	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>				0000 0000 0000 0000				

凡例: u = 未初期化

注: レジスタマップはデバイスの出力比較モジュール数に左右されます。詳細はデバイスデータシートを参照してください。

表 14-6: 出力比較モジュールに関連したレジスタマップ例（続き）

SFR 名称	Addr.	ピット 15	ピット 14	ピット 13	ピット 12	ピット 11	ピット 10	ピット 9	ピット 8	ピット 7	ピット 6	ピット 5	ピット 4	ピット 3	ピット 2	ピット 1	ピット 0	リセット状態
IFS0	0084	CNIF	BCLIF	I2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0	0000 0000 0000 0000
IFS1	0086	IC6IF	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IFS2	0088	—	—	—	FLTBIF	FLTAIF	LVDIF	DCIIF	QEIIIF	PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF	0000 0000 0000 0000
IEC0	008C	CNIE	BCLIE	I2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	IC6IE	IC5IE	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IEC2	0090	—	—	—	FLTBIE	FLTAIE	LVDIE	DCIIE	QEIIIE	PWMIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE	0000 0000 0000 0000
IPC0	0094	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>			0100 0100 0100 0100
IPC1	0096	—	T3IP<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>			0100 0100 0100 0100
IPC4	009C	—	OC3IP<2:0>			—	IC8IP<2:0>			—	IC7IP<2:0>			—	INT1IP<2:0>			0100 0100 0100 0100
IPC5	009E	—	INT2IP<2:0>			—	T5IP<2:0>			—	T4IP<2:0>			—	OC4IP<2:0>			0100 0100 0100 0100
IPC8	00A4	—	OC8IP<2:0>			—	OC7IP<2:0>			—	OC6IP<2:0>			—	OC5IP<2:0>			0100 0100 0100 0100

凡例: u = 未初期化

注: レジスタマップはデバイスの出力比較モジュール数に左右されます。詳細はデバイスデータシートを参照してください。

## 14.6 設計の秘訣

**質問 1:** 出力比較ピンが **OCSIDL** ビットがセットされてないにもかかわらず機能停止しますが、なぜでしょうか。

回答: これは関連するタイマーソースの **TSIDL** ビット (**TxCON<13>**) がセットされている場合に最も発生する可能性があります。したがって、**PWRSAV** 命令が実行された場合実際 **IDLE** モードになっているのはタイマーです。

**質問 2:** 出力比較モジュールを32ビットモードに構成されているタイマーとの使用は可能ですか。

回答: いいえ。**T32** ビット (**TxCON<3>**) はタイマーが出力比較モジュールと共に使用された場合クリアされる必要があります。

## 14.7 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正して使用可能です。制限がある場合もあります。現状、出力比較モードに関連する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 #

現在のところ、関連するアプリケーションノートはありません。

**注：** dsPIC30F ファミリーのデバイスに関しての、他のアプリケーションノート  
やコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧  
下さい。

## 14.8 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

マニュアルの本章に対する技術内容や編集改訂版はありませんが、マニュアル全体を通して B 版を反映するために、この章は更新されています。



**MICROCHIP**

## 第 15 章. モーター制御 PWM

### ハイライト

この章には以下の項目が含まれています。

15.1 序章 .....	15-2
15.2 制御レジスタ .....	15-4
15.3 PWM タイムベース .....	15-16
15.4 PWM デューティ比較ユニット .....	15-20
15.5 相補 PWM 出力モード .....	15-24
15.6 デッドタイム制御 .....	15-25
15.7 独立 PWM 出力モード .....	15-28
15.8 PWM 出力オーバーライド .....	15-29
15.9 PWM 出力極性制御 .....	15-32
15.10 PWM フォールトピン .....	15-32
15.11 PWM アップデートロックアウト .....	15-35
15.12 PWM 特殊イベントトリガー .....	15-35
15.13 デバイス省電力モードにおける動作 .....	15-36
15.14 デバイスエミュレーション用特別機能 .....	15-37
15.15 関連するアプリケーションノート .....	15-40
15.16 改訂履歴 .....	15-41

## 15.1 序章

モーター制御用 PWM (MCPWM モジュール) を用いることにより、複数の同期したパルス幅変調出力を簡単に行うことが可能となります。特に次の電力制御や駆動アプリケーションがサポートされています。

- 3 相 AC 誘導電動機
- スイッチトリラクタンス (SR) モーター
- ブラシレス直流 (BLDC) モーター
- 無停電電源装置 (UPS)

PWM モジュールには以下の機能があります。

- TCY/2 の分解能のタイムベースを提供
- 各 PWM ジェネレータ用出力ピン 2 本
- 各出力ペアは相補でも独立でも動作可能
- 相補モード用ハードウエアデッドタイム生成
- デバイス設定ビットにより設定可能な出力ピン極性
- 複数の出力モード
  - エッジ整列モード
  - 中央整列モード
  - ダブルアップデートを伴う中央整列モード
  - シングルイベントモード
- PWM 出力ピン用マニュアル優先レジスタ
- 機能設定変更ができるハードウエアフォールト入力ピン
- A/D コンバータ同期用特殊イベントトリガー
- PWM に対応する各出力ピンは個別に有効化可能。

### 15.1.1 MCPWM モジュールの変形

選択される dsPIC30F デバイスによって、MCPWM モジュールには 2 つのバージョンがあります。64 ピン以上のデバイスには多くの場合 8 出力モジュールが用意されています。64 ピン以下のデバイスには 6 出力の MCPWM が用意されています。dsPIC30F デバイスの中には、2 種類以上の MCPWM モジュールが用いられていることもあります。

詳しくは個別のデバイスデータシートを参照してください。

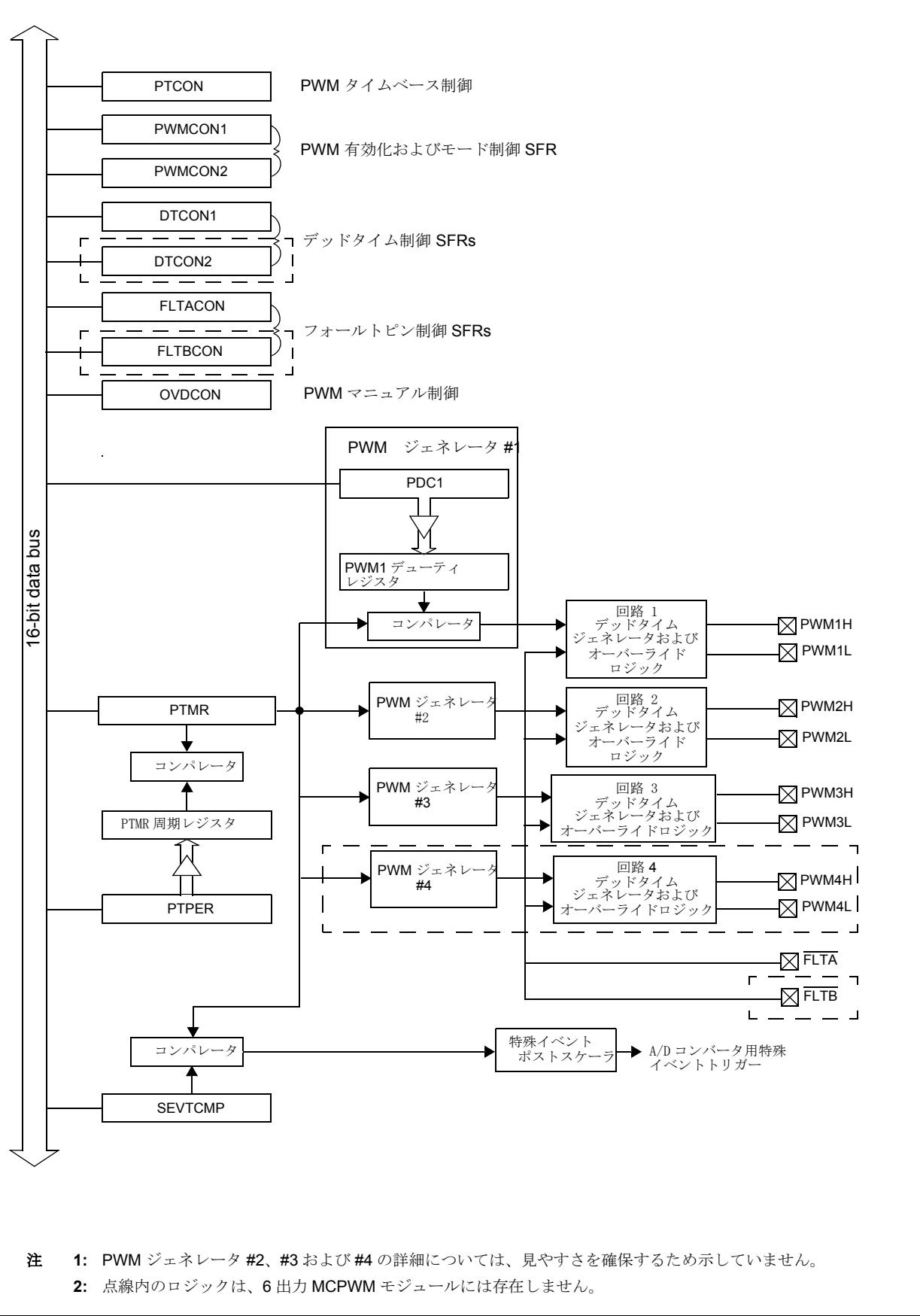
表 15-1: 機能要約 : 6 出力 MCPWM vs 8- 出力 MCPWM

機能	6- 出力 MCPWM モジュール	8- 出力 MCPWM モジュール
入出力ピン	6	8
PWM ジェネレータ	3	4
フォールト入力ピン	1	2
無電圧デッドタイムジェネレータ	1	2

6 出力 MCPWM モジュールは単相または 3 相のパワーアプリケーションにおいて有用であり、一方 8MCPWM は 4 相のモーターアプリケーションをサポート可能です。表 15-1 は 6 出力および 8 出力の MCPWM モジュールの機能を要約したものです。どちらのモジュールも、複数の単相負荷をサポートします。8 出力 MCPWM はまた、フォールトピン 2 本とプログラマブルなデッドタイム 2 つをサポートするため、アプリケーションの柔軟性を高めます。これらの特徴に関しては後の章で詳細に取り上げます。

簡略化した MCPWM の回路図を図 15-1 に記しています。

図 15-1: MCPWM ブロック図



## 15.2 制御レジスタ

以下のレジスタにより MCPWM モジュールの動作が制御されています。

- PTCON: PWM タイムベース制御レジスタ
- PTMR: PWM タイムベースレジスタ
- PTPER: PWM タイムベース周期レジスタ
- SEVTCMP: PWM 特殊イベント比較レジスタ
- PWMCON1: PWM 制御レジスタ #1
- PWMCON2: PWM 制御レジスタ #2
- DTCON1: デッドタイム制御レジスタ #1
- DTCON2: デッドタイム制御レジスタ #2
- FLTACON: フォールト A 制御レジスタ
- FLTBCON: フォールト B 制御レジスタ
- PDC1: PWM デューティレジスタ #1
- PDC2: PWM デューティレジスタ #2
- PDC3: PWM デューティレジスタ #3
- PDC4: PWM デューティレジスタ #4

さらに、最初のリセット状態および入出力ピンの極性を設定する、MCPWM モジュールと関連したデバイスコンフィギュレーションビットが 3 種類あります。これらのコンフィギュレーションビットは FBORPOR デバイスコンフィギュレーションレジスタに位置しています。詳しくはセクション第 24 章 **¶ デバイスコンフィギュレーション** を参照してください。

## レジスタ 15-1: PTCON: PWM タイムベース制御レジスタ

高位バイト:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
PTEN	—	PTSIDL	—	—	—	—	—
ビット 15				ビット 8			

低位バイト:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>	
ビット 7				ビット 0			

- ビット 15 **PTEN:** PWM タイムベースのタイマー有効化ビット  
1 = PWM タイムベース「オン」  
0 = PWM タイムベース「オフ」
- ビット 14 未実装: ‘0’ として読み出し
- ビット 13 **PTSIDL:** IDLE モードにおける PWM タイムベースの停止ビット  
1 = CPU IDLE モードにおいて PWM タイムベース停止  
0 = CPU IDLE モードにおいて PWM タイムベース作動
- ビット 12-8 未実装: ‘0’ として読み出し
- ビット 7-4 **PTOPS<3:0>:** PWM タイムベース出力ポストスケール選択ビット  
1111 = ポストスケール比 1:16  
•  
•  
0001 = ポストスケール比 1:2  
0000 = ポストスケール比 1:1
- ビット 3-2 **PTCKPS<1:0>:** PWM タイムベース入力クロックプリスケール選択ビット  
11 = PWM タイムベース入力クロック周期が 64 TCY (1:64 プリスケール比)  
10 = PWM タイムベース入力クロック周期が 16 TCY (1:16 プリスケール比)  
01 = PWM タイムベース入力クロック周期が 4 TCY (1:4 プリスケール比)  
00 = PWM タイムベース入力クロック周期が TCY (1:1 プリスケール比)
- ビット 1-0 **PTMOD<1:0>:** PWM タイムベースモード選択ビット  
11 = PWM タイムベースは連続的なアップダウンモードで作動し、二重に PWM の更新があった場合は割込み発生  
10 = PWM タイムベースは連続的なアップダウンカウンタモードで作動  
01 = PWM タイムベースはシングルイベントモードで作動  
00 = PWM タイムベースはフリーランモードで作動

## 注釈:

R = 読出し可能ビット	W = 書込み可能ビット	U = 実行せず, ‘0’ として読み出し
-n = POR 値	‘1’ = ビットがセット済	‘0’ = ビット消去
		x = ビット不定

# dsPIC30F ファミリーリファレンスマニュアル

## レジスタ 15-2: PTMR: PWM タイムベースカウンタ

高位バイト :							
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTDIR	PTMR <14:8>						
ビット 15	ビット 8						

低位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTMR <7:0>							
ビット 7	ビット 0						

ビット 15 **PTDIR: PWM タイムベースカウント方向ステータスビット** (読み取り専用)

1 = PWM タイムベースがカウント減少中

0 = PWM タイムベースがカウント上昇中

ビット 14-0 **PTMR <4:0>: PWM タイムベースレジスタカウント値**

注釈 :

R = 読出し可能ビット

W = 書込み可能ビット

U = 実行せず, ‘0’ として読み出

-n = POR 値

‘1’ = ビットがセット済

‘0’ = ビット消去

x = ビット不定

## レジスタ 15-3: PTPER: PWM タイムベース周期レジスタ

高位バイト :							
U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	PTPER <14:8>						
ビット 15	ビット 8						

低位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTPER <7:0>							
ビット 7	ビット 0						

ビット 15 **未実装: ‘0’ として読み出**

15

ビット 14-0 **PTPER<14:0>: PWM タイムベース周期値**

注釈 :

R = 読出し可能ビット

W = 書込み可能ビット

U = 実行せず, ‘0’ として読み出

-n = POR 値

‘1’ = ビットがセット済

‘0’ = ビット消去

x = ビット不定

## レジスタ 15-4: SEVTCMP: 特殊イベント比較レジスタ

高位バイト:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SEVTDIR	SEVTCMP <14:8>						
ビット 15	ビット 8						

低位バイト:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SEVTCMP <7:0>							
ビット 7	ビット 0						

ビット 15 **SEVTDIR:** 特殊イベントトリガータイムベース方向ビット<sup>(1)</sup>

1 = PWM タイムベースのカウント値が減少中に特殊イベントトリガーが作動する。

0 = PWM タイムベースのカウント値が上昇中に特殊イベントトリガーが作動する。

ビット 14-0 **SEVTCMP <14:0>:** 特殊イベント比較値ビット<sup>(2)</sup>

注 1: SEVTDIR が PTDIR(PTMR&lt;15&gt;) と比較され、特殊イベントトリガーを作動させる。

2: SEVTCMP&lt;14:0&gt; が PTMR&lt;14:0&gt; と比較され、特殊イベントトリガーを作動させる。

注釈:

R = 読出し可能ビット

W = 書出し可能ビット

U = 実行せず、'0' として読み出

-n = POR 値

'1' = ビットがセット済

'0' = ビット消去

x = ビット不定

## レジスタ 15-5: PWMCON1: PWM 制御レジスタ 1

高位バイト:							
U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	PMOD4	PMOD3	PMOD2	PMOD1
ビット 15	ビット 8						

低位バイト:

R/W-1							
PEN4H	PEN3H	PEN2H	PEN1H	PEN4L	PEN3L	PEN2L	PEN1L
ビット 7	ビット 0						

ビット 15-12 未実装: '0' として読み出

ビット 11-8 **PMOD4:PMOD1:** PWM 入出力ペアモードビット

1 = PWM 入出力ピンペアは独立出力モード

0 = PWM 入出力ピンペアは相補出力モード

ビット 7-4 **PEN4H-PEN1H:** PWMxH 入出力有効化ビット<sup>(1)</sup>

1 = PWMxH ピンは有効化され PWM 出力に用いられます

0 = PWMxH ピン無効化。入出力ピンは汎用入出力ピンとして用いられます。

ビット 3-0 **PEN4L-PEN1L:** PWMxL 入出力有効化ビット<sup>(1)</sup>

1 = PWMxL ピンは有効化され PWM 出力に用いられます。

0 = PWMxL ピン無効化。入出力ピンは汎用入出力ピンとして用いられます。

注 1: PENxH および PENxL ビットのリセット条件は、FBORPOR デバイスコンフィギュレーション レジスタの PWM/PIN デバイスコンフィギュレーションビットの値により決められます。

注釈:

R = 読出し可能ビット

W = 書込み可能ビット

U = 実行せず、'0' として読み出

-n = POR 值

'1' = ビットがセット済

'0' = ビット消去

x = ビット不定

## レジスタ 15-6: PWMCON2: PWM 制御レジスタ 2

高位バイト :									
U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0		
—	—	—	—	SEVOPS<3:0>					
ビット 15						ビット 8			

低位バイト :							
U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	OSYNC	UDIS
ビット 7						ビット 0	

ビット 15-12 未実装: ‘0’ として読出し

ビット 11-8 **SEVOPS<3:0>:** PWM 特殊イベントトリガー出力ポストスケール選択ビット  
1111 = 1:16 ポストスケール

•  
•

0001 = 1:2 ポストスケール

0000 = 1:1 ポストスケール

ビット 7-2 未実装: ‘0’ として読出し

ビット 1 **OSYNC:** 出力オーバーライド同期化ビット  
1 = OVDCON レジスタによる出力オーバーライドが PWM タイムベースと同期化  
0 = OVDCON レジスタによる出力オーバーライドが次の TCY 境界で発生

ビット 0 **UDIS:** PWM 更新無効化ビット

1 = デューティおよび周期バッファレジスタからの更新無効化。

0 = デューティおよび周期バッファレジスタからの更新有効。

注釈 :

R = 読出し可能ビット W = 書込み可能ビット U = 実行せず, ‘0’ として読出し

-n = POR 値 ‘1’ = ビットがセット済 ‘0’ = ビット消去 x = ビット不定

## レジスタ 15-7: DTCO1: デッドタイムコントロールレジスタ 1

高位バイト：							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DTBPS<1:0>				DTB<5:0>			
ビット 15	ビット 8						

低位バイト：							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DTAPS<1:0>				DTA<5:0>			
ビット 7	ビット 0						

ビット 15-14 DTBPS<1:0>: デッドタイムユニット B プリスケール選択ビット

11 = デッドタイムユニット B のクロック周期が 8 TCY

10 = デッドタイムユニット B のクロック周期が 4 TCY

01 = デッドタイムユニット B のクロック周期が 2 TCY

00 = デッドタイムユニット B のクロック周期が TCY

ビット 13-8 DTB<5:0>: デッドタイムユニット B 用符号なしの 6 ビットデッドタイム値

ビット 7-6 DTAPS<1:0>: デッドタイムユニット A プリスケール選択ビット

11 = デッドタイムユニット A のクロック周期が 8 TCY

10 = デッドタイムユニット A のクロック周期が 4 TCY

01 = デッドタイムユニット A のクロック周期が 2 TCY

00 = デッドタイムユニット A のクロック周期が TCY

ビット 5-0 DTA<5:0>: デッドタイムユニット A 用符号なしの 6 ビットデッドタイム値

注釈：

R = 読出し可能ビット

W = 書込み可能ビット

U = 実行せず, ‘0’ として読み出し

-n = POR 値

‘1’ = ビットがセット済

‘0’ = ビット消去

x = ビット不定

## レジスタ 15-8: DTC0N2: デッドタイムコントロールレジスタ 2

高位バイト：

U-0							
—	—	—	—	—	—	—	—

ビット 15

ビット 8

低位バイト：

R/W-0							
DTS4A	DTS4I	DTS3A	DTS3I	DTS2A	DTS2I	DTS1A	DTS1I

ビット 7

ビット 0

ビット 15-8 未実装：‘0’として読出し

ビット 7 **DTS4A:** PWM4 用デッドタイム選択ビット有効化  
1 = デッドタイムがユニット B から送られる。  
0 = デッドタイムがユニット A から送られる。

ビット 6 **DTS4I:** PWM4 用デッドタイム選択ビット無効化  
1 = デッドタイムがユニット B から送られる。  
0 = デッドタイムがユニット A から送られる。

ビット 5 **DTS3A:** PWM3 用デッドタイム選択ビット有効化  
1 = デッドタイムがユニット B から送られる。  
0 = デッドタイムがユニット A から送られる。

ビット 4 **DTS3I:** PWM3 用デッドタイム選択ビット無効化  
1 = デッドタイムがユニット B から送られる。  
0 = デッドタイムがユニット A から送られる。

ビット 3 **DTS2A:** PWM2 用デッドタイム選択ビット有効化  
1 = デッドタイムがユニット B から送られる。  
0 = デッドタイムがユニット A から送られる。

ビット 2 **DTS2I:** PWM2 用デッドタイム選択ビット無効化  
1 = デッドタイムがユニット B から送られる。  
0 = デッドタイムがユニット A から送られる。

ビット 1 **DTS1A:** PWM1 用デッドタイム選択ビット有効化  
1 = デッドタイムがユニット B から送られる。  
0 = デッドタイムがユニット A から送られる。

ビット 0 **DTS1I:** PWM1 用デッドタイム選択ビット無効化  
1 = デッドタイムがユニット B から送られる。  
0 = デッドタイムがユニット A から送られる。

注釈：

R = 読出し可能ビット

W = 書込み可能ビット

U = 実行せず，‘0’として読出し

-n = POR 値

‘1’ = ビットがセット済

‘0’ = ビット消去

x = ビット不定

## レジスタ 15-9: FLTACON: フォールト A 制御レジスタ

高位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
FAOV4H	FAOV4L	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L
ビット 15							ビット 8

低位バイト :							
R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
FLTAM	—	—	—	FAEN4	FAEN3	FAEN2	FAEN1
ビット 7							ビット 0

ビット 15-8 **FAOV4H-FAOV1L:** フォールト入力時の PWM オーバーライド値ビット

1 = 外部からのフォールト入力イベントにより、PWM 出力ピンをアクティブにする。

0 = 外部からのフォールト入力イベントにより、PWM 出力ピンをインアクティブにする。

ビット 7 **FLTAM:** フォールト A モードビット

1 = フォールト A 入力ピンはサイクルバイサイクルモードで機能。

0 = フォールト A 入力ピンは全ての制御ピンを FLTACON<15:8> でプログラムされた状態にラッチします。

ビット 6-4 未実装: ‘0’ として読み出し

6-4

ビット 3 **FAEN4:** フォールト入力 A 有効化ビット

1 = PWM4H/PWM4L ピンの対はフォールト入力 A により制御されます。

0 = PWM4H/PWM4L ピンの対はフォールト入力 A により制御されません。

ビット 2 **FAEN3:** フォールト入力 A 有効化ビット

1 = PWM3H/PWM3L ピンの対はフォールト入力 A により制御されます。

0 = PWM3H/PWM3L ピンの対はフォールト入力 A により制御されません。

ビット 1 **FAEN2:** フォールト入力 A 有効化ビット

1 = PWM2H/PWM2L ピンの対はフォールト入力 A により制御されます。

0 = PWM2H/PWM2L ピンの対はフォールト入力 A により制御されません。

ビット 0 **FAEN1:** フォールト入力 A 有効化ビット

1 = PWM1H/PWM1L ピンの対はフォールト入力 A により制御されます。

0 = PWM1H/PWM1L ピンの対はフォールト入力 A により制御されません。

注釈 :

R = 読出し可能ビット

W = 書込み可能ビット

U = 実行せず, ‘0’ として読み出し

-n = POR 値

‘1’ = ビットがセット済

‘0’ = ビット消去

x = ビット不定

## レジスタ 15-10: FLTBCON: フォールト B 制御レジスタ

高位バイト :

| R/W-0  |
|--------|--------|--------|--------|--------|--------|--------|--------|
| FBOV4H | FBOV4L | FBOV3H | FBOV3L | FBOV2H | FBOV2L | FBOV1H | FBOV1L |
| ビット 15 |        |        |        | ビット 8  |        |        |        |

低位バイト :

R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
FLTBM	—	—	—	FBEN4	FBEN3	FBEN2	FBEN1
ビット 7				ビット 0			

ビット 15-8 **FBOV4H:FBOV1L:** フォールト入力時 PWM オーバーライド値ビット

1 = 外部からのフォールト入力イベントにより、PWM 出力ピンをアクティブにする。

0 = 外部からのフォールト入力イベントにより、PWM 出力ピンをインアクティブにする。

ビット 7 **FLTBM:** フォールト B モードビット

1 = フォールト B 入力ピンはサイクルバイサイクルモードで機能。

0 = フォールト B 入力ピンは全ての制御ピンを **FLTACON<15:8>** でプログラムされた状態にラッチ。

ビット 6-4 未実装: ‘0’ として読出し

ビット 3 **FAEN4:** フォールト入力 B 有効化ビット <sup>(1)</sup>

1 = PWM4H/PWM4L ピンの対はフォールト入力 B により制御されます。

0 = PWM4H/PWM4L ピンの対はフォールト入力 B により制御されません。

ビット 2 **FAEN3:** フォールト入力 B 有効化ビット <sup>(1)</sup>

1 = PWM3H/PWM3L ピンの対はフォールト入力 B により制御されます。

0 = PWM3H/PWM3L ピンの対はフォールト入力 B により制御されません。

ビット 1 **FAEN2:** フォールト入力 B 有効化ビット <sup>(1)</sup>

1 = PWM2H/PWM2L ピンの対はフォールト入力 B により制御されます。

0 = PWM2H/PWM2L ピンの対はフォールト入力 B により制御されません。

ビット 0 **FAEN1:** フォールト入力 B 有効化ビット <sup>(1)</sup>

1 = PWM1H/PWM1L ピンの対はフォールト入力 B により制御されます。

0 = PWM1H/PWM1L ピンの対はフォールト入力 B により制御されません。

注 1: 有効化された場合、フォールトピン A はフォールトピン B に優先します。

注釈 :

R = 読出し可能ビット W = 書込み可能ビット U = 実行せず, ‘0’ として読出し

-n = POR 値 ‘1’ = ビットがセット済 ‘0’ = ビット消去 x = ビット不定

## レジスタ 15-11: OVDCON: オーバーライド制御レジスタ

高位バイト:							
R/W-1							
POVD4H	POVD4L	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L
ビット 15							ビット 8

低位バイト:							
R/W-0							
POUT4H	POUT4L	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L
ビット 7							ビット 0

## ビット 15-8: POVD4H-POVD1L: PWM 出力オーバーライドビット

1 = PWMxx 入出力ピンによる出力は PWM ジェネレータにより制御されます。

0 = PWMxx 入出力ピンによる出力は対応する POUTxx ビットの値により制御されます。

## ビット 7-0: POUT4H-POUT1L: PWM マニュアル出力ビット

1 = PWMxx 入出力ピンは、対応する POVDxx ビットがクリアされた場合にアクティブになります。

0 = PWMxx 入出力ピンは、対応する POVDxx ビットがクリアされた場合にインアクティブになります。

## 注釈:

R = 読出し可能ビット

W = 書込み可能ビット

U = 実行せず, '0' として読み出し

-n = POR 値

'1' = ビットがセット済

'0' = ビット消去

x = ビット不定

## レジスタ 15-12: PDC1: PWM デューティサイクルレジスタ 1

高位バイト:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM デューティ #1 ビット 15-8							
ビット 15							ビット 8

## 低位バイト:

R/W-0 R/W-0 R/W-0 R/W-0 R/W-0 R/W-0 R/W-0 R/W-0

PWM デューティ #1 ビット 7-0

ビット 7

ビット 0

## ビット 15-0: PDC1&lt;15:0&gt;: PWM デューティ #1 数値

15-0

## 注釈:

R = 読出し可能ビット

W = 書込み可能ビット

U = 実行せず, '0' として読み出し

-n = POR 値

'1' = ビットがセット済

'0' = ビット消去

x = ビット不定

# dsPIC30F ファミリーリファレンスマニュアル

## レジスタ 15-13: PDC2: PWM デューティレジスタ 2

高位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM デューティ #2 ビット 15-8							
ビット 15							ビット 8

低位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM デューティ #2 ビット 7-0							
ビット 7							ビット 0

ビット 15-0 PDC2<15:0>: PWM デューティ #2 数値

注釈 :							
R = 読出し可能ビット	W = 書込み可能ビット	U = 実行せず, ‘0’ として読み出し					
-n = POR 値	‘1’ = ビットがセット済	‘0’ = ビット消去	x = ビット不定				

## レジスタ 15-14: PDC3: PWM デューティレジスタ 3

高位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM デューティ #3 ビット 15-8							
ビット 15							ビット 8

低位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM デューティ #3 ビット 7-0							
ビット 7							ビット 0

ビット 15-0 PDC3<15:0>: PWM デューティ #3 数値

注釈 :							
R = 読出し可能ビット	W = 書込み可能ビット	U = 実行せず, ‘0’ として読み出し					
-n = POR 値	‘1’ = ビットがセット済	‘0’ = ビット消去	x = ビット不定				

## レジスタ 15-15: PDC4: PWM デューティレジスタ 4

高位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM デューティ #4 ビット 15-8							
ビット 15							ビット 8

低位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM デューティ #4 ビット 7-0							
ビット 7							ビット 0

ビット 15-0 PDC4<15:0>: PWM デューティ #4 値

注釈 :							
R = 読出し可能ビット	W = 書込み可能ビット	U = 実行せず, '0' として読み出	-n = POR 値	'1' = ビットがセット済	'0' = ビット消去	x = ビット不定	

## レジスタ 15-16: FBORPOR: BOR および POR デバイスコンフィギュレーションレジスタ

高位バイト :							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—

ビット 23 ビット 16

中位バイト :							
U-0	U-0	U-0	U-0	U-0	R/P	R/P	R/P
—	—	—	—	—	PWMPIN	HPOL	LPOL

ビット 15 ビット 8

低位バイト :							
R/P	U-0	R/P	R/P	U-0	U-0	R/P	R/P
BOREN	—	BORV<1:0>	—	—	—	FPWRT<1:0>	—

ビット 7 ビット 0

ビット 10 PWMPIN: MPWM ドライバ初期化ビット

1 = リセット時のピンの状態は入出力ポートにより制御 (PWMCN1<7:0> = 0x00)

0 = リセット時のピンの状態は PWM モジュールにより制御 (PWMCN1<7:0> = 0xFF)

ビット 9 HPOL: MCPWM ハイサイドドライバ (PWMMxH) 極性ビット

1 = PWMMxH ピンの出力信号がアクティブで High の極性を有する。

0 = PWMMxH ピンの出力信号がアクティブで Low の極性を有する。

ビット 8 LPOL: MCPWM ローサイドドライバ (PWMMxL) 極性ビット

1 = PWMMxL ピンの出力信号がアクティブで High の極性を有する。

0 = PWMMxL ピンの出力信号がアクティブで Low の極性を有する。

注: このレジスタにある他のコンフィギュレーションビットについて、詳しくはセクション 24 章「デバイスコンフィギュレーション」を参照してください。

注释 :

R = 読出し可能ビット W = 書込み可能ビット U = 実行せず, '0' として読み出

-n = POR 値 '1' = ビットがセット済 '0' = ビット消去 x = ビット不定

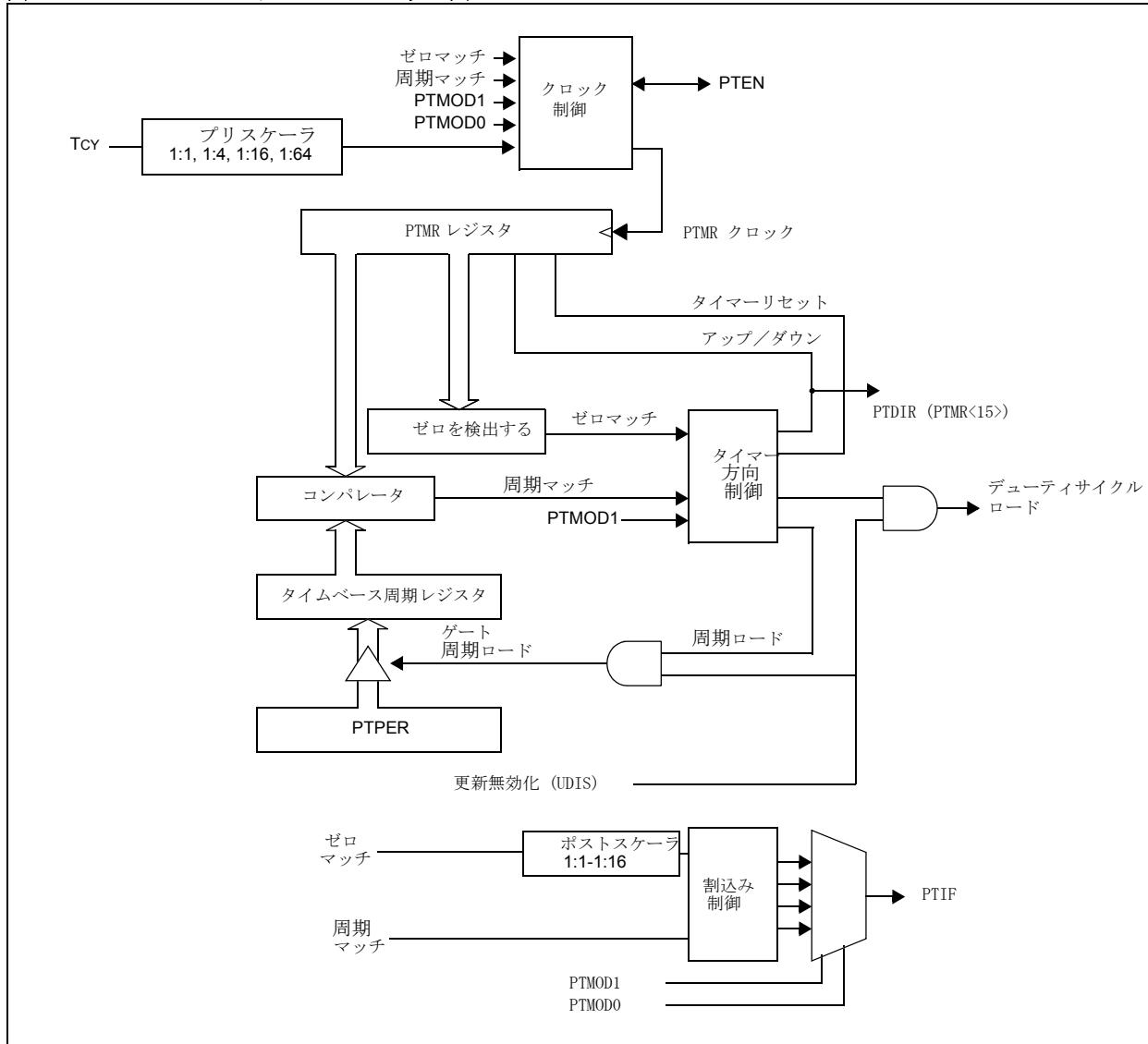
P = プログラム可能設定ビット

## 15.3 PWM タイムベース

PWM タイムベースには 15 ビットタイマーが備えられ、プリスケーラとポストスケーラが用意されています（図 15-2 参照）。タイムベースの 15 ビットは PTMR を通してアクセス可能です。PTMR<15> は読み取り専用ステータスピットであり、PTDIR は PWM タイムベースの現在のカウント方向を示しています。PTDIR ステータスピットがクリアされると PTMR は上方にカウントされ、PTDIR がセットされると PTMR は下方にカウントされます。

タイムベースは、PTEN ビット (PTCON<15>) をセット / クリアすることにより、有効化 / 無効化されます。ソフトウェアで PTEN がクリアされた場合、PTMR はクリアされません。

図 15-2: PWM タイムベースブロック図



PWM タイムベースは 4 つの異なる実行モードに設定可能です。

1. フリーランモード
2. シングルイベントモード
3. 連続的アップ / ダウンカウントモード
4. 二重アップデートが発生した場合に割込みが発生する連続的アップ / ダウンカウントモード

4 つのモードは PTMOD<1:0> 制御 (PTCON<1:0>) により選択されます。

**注:** PWM タイムベースのモードにより、モジュールで発生する PWM 信号の種類が決まります（詳しくは項 15.4.2、項 15.4.3、項 15.4.4 を参照してください）。

### 15.3.1 フリーランモード

フリーランモードでは、PTPER レジスタと一致するまでタイムベースは上方にカウントされます。PTMR レジスタはその後に入力クロックエッジが発生するとリセットされます。PTEN ビットがセットされている限りタイムベースは上方にカウントされ続けます。

### 15.3.2 シングルイベントモード

シングルイベントモードでは、PTEN ビットがセットされると PWM タイムベースは上方にカウントされ始めます。PTMR 値が PTPER レジスタと一致すると、PTMR レジスタは次の入力クロックエッジでリセットされ、PTEN ビットはハードウェアによりクリアされ、タイムベースが停止します。

### 15.3.3 アップ / ダウンカウントモード

アップ / ダウンカウントモードでは、PWM タイムベースは PTPER レジスタの値が一致するまで上方にカウントされます。その後に入力クロックエッジが発生するとタイマーは下方にカウントを始め、「0」に達するまで下方にカウントを続けます。PTDIR ビット PTMR<15> は読み出し専用で、カウント方向を示します。タイマーが下方にカウントされているとき PTDIR ビットがセットされます。

### 15.3.4 PWM タイムベースプリスケーラ

PTMR 入力クロック (Tcy) のプリスケーラ比の選択肢には 1:1、1:4、1:16 または 1:64 があり、PTCKPS<1:0>(PTCON<3:2>) という制御ビットにより選択されます。以下のいずれかが発生した場合、プリスケーラのカウンタはクリアされます。

- PTMR レジスタへの書き込み
- PTCON レジスタへの書き込み
- デバイスのリセット

PTCON が書き込まれたときに PTMR レジスタはクリアされません。

### 15.3.5 PWM タイムベースポストスケーラ

一致した PTMR の出力は、任意に 4 ビットポストスケーラ (1:1 から 1:16 までのスケーリングを含む) によりポストスケールされ、割込みを発生させます。PWM デューティサイクルが毎 PWM サイクルごとに更新される必要がない場合、ポストスケーラは有用です。

以下のいずれかが発生した場合、ポストスケーラのカウンタはクリアされます。

- PTMR レジスタへの書き込み
- PTCON レジスタへの書き込み
- デバイスのリセット

PTCON が書き込まれたときに PTMR レジスタはクリアされません。

## 15.3.6 PWM タイムベース割込み

PWM タイムベースにより発生する割込み信号は、モード選択ビットである PTMOD<1:0> (PTCON<1:0>) およびタイムベースポストスケーラビットの PTOPS<3:0>(PTCON<7:4>) により決められます。

- フリーランモード

PWM タイムベースがフリーランモードにあるとき (PTMOD<1:0> = 00)、PTPER レジスタと一致することで PTMR レジスタが ‘0’ にリセットされた場合に割込みが発生します。割込みイベントの頻度を下げるために、タイマーがフリーランモードに設定されているときに、ポストスケーラ選択ビットが用いられます。

- シングルイベントモード

PWM タイムベースがシングルイベントモードのとき (PTMOD<1:0> = 01)、PTPER レジスタと一致することで PTMR レジスタが ‘0’ にリセットされた場合に割込みが発生します。この場合 PTEN ビット (PTCON<15>) もクリアされ、PTMR のカウントは中止されます。ポストスケーラ選択ビットは、タイマーがこのモードに設定されている場合は影響を及ぼしません。

- アップ/ダウンカウントモード

アップ/ダウンカウントモードの場合 (PTMOD<1:0> = 10)、PTMR レジスタの値が 0 になり、PWM タイムベースが上方にカウントを始めるごとに割込みイベントが発生します。タイマーがこのモードに設定されている場合、割込みイベントの頻度を下げるためにポストスケーラ選択ビットが用いられます。

- ダブルアップデートのあるアップ/ダウンカウントモード

ダブルアップデートモードの場合 (PTMOD<1:0> = 11)、PTMR がゼロになり、周期の一致が発生するごとに割込みが発生します。ポストスケーラ選択ビットは、タイマーがこのモードに設定されている場合は影響を及ぼしません。

ダブルアップデートモードの場合、PWM デューティサイクルは周期ごとに 2 回更新されるため制御ループ帯域幅は 2 倍になります。PWM 信号の立上がりエッジおよび立下がりエッジ全てがダブルアップデートモードにより制御されます。

## 15.3.7 PWM 周期

PTPER レジスタにより PTMR のカウント周期がセットされます。ユーザーは PTPER<14:0> に 15 ビットの値を書き込む必要があります。PTMR<14:0> の値が PTPER<14:0> の値と一致すると、タイムベースは ‘0’ にリセットされるか、または次にクロック入力エッジが発生した場合、タイムベースのカウント方向が逆転します。どちらの動作が生じるかはタイムベースの動作モードによって決まります。

タイムベース周期は二重バッファ構成になっていて、グリッチ発生なく PWM 信号の周期の変更が可能となります。PTPER レジスタは実際のタイムベース周期レジスタへのバッファレジスタとして用いられ、ユーザーがアクセスすることはできません。PTPER レジスタの内容は、次の場合一実際のタイムベース周期レジスタにロードされます。

- フリーランモードおよびシングルイベントモード : PTPER レジスタと一致し、PTMR レジスタがゼロにリセットされた場合。
- アップ/ダウンカウントモード : PTMR レジスタが 0 になった場合。

PWM タイムベースが無効化された場合 (PTEN = 0)、PTPER レジスタの値は自動的にタイムベース周期レジスタにロードされます。

図 15-3 および図 15-4 は、PTPER レジスタの内容がタイムベース周期レジスタにロードされた場合のタイミングを示しています。

図 15-3: フリーランカウントモードにおける PWM 周期バッファ更新

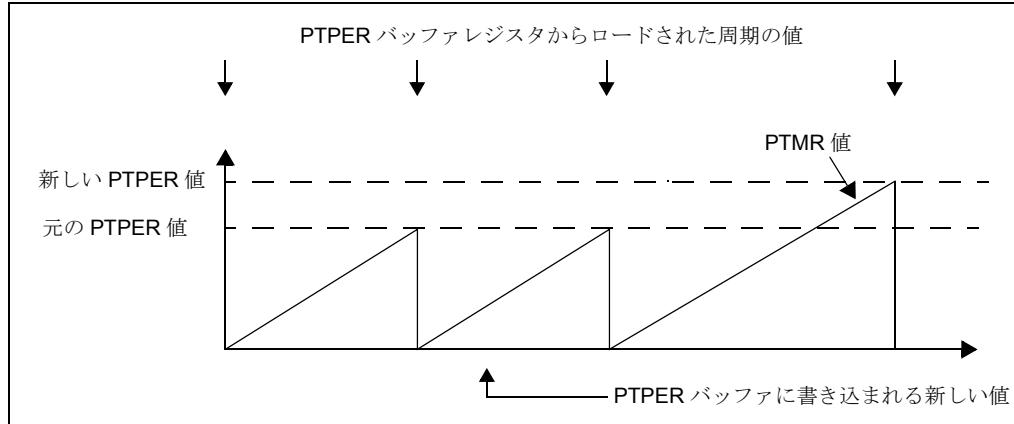
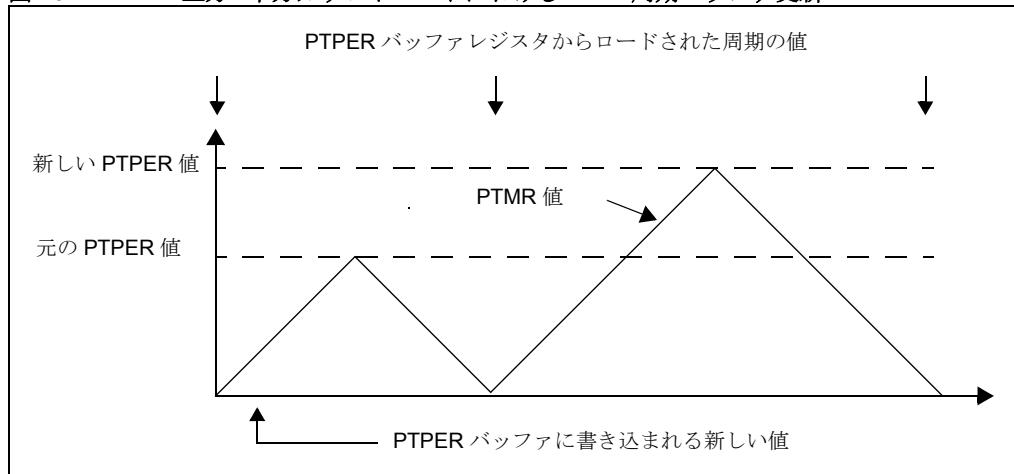


図 15-4: 上方 / 下方カウントモードにおける PWM 周期バッファ更新



PWM 周期は次の公式から求められます。

式 15-1: PWM 周期の算出

$$\text{PTPER} = \frac{\text{FCY}}{\text{FPWM} \cdot (\text{PTMR プリスケーラ}) - 1}$$

例 :

FCY = 20 MHz  
FPWM = 20,000 Hz  
PTMR プリスケーラ = 1:1

$$\begin{aligned}\text{PTPER} &= \frac{20,000,000}{20,000 \cdot 1} - 1 \\ &= 1000 - 1 \\ &= 999\end{aligned}$$

注： PWM タイムベースがアップカウントまたはダウンカウントモードのいずれかに設定された場合、PWM 周期は式 15-1 で求められる数値の 2 倍となります。

## 15.4 PWM デューティ比較ユニット

MCPWM モジュールには 4 つの PWM ジェネレータがあります。PWM ジェネレータのデューティ値を特定するために用いられる 16 ビット特殊関数レジスタが 4 つあります。

- PDC1
- PDC2
- PDC3
- PDC4

今後 PDCx とは、4 つの PWM デューティレジスタのうちいずれか 1 つを示すものとします。

### 15.4.1 PWM デューティ分解能

特定のデバイスクロックの最大分解能（単位ビット）および PWM の周波数は、次の式により求められます。

式 15-2: PWM 分解能

$$Resolution = \frac{\log\left(\frac{2T_{PWM}}{TCY}\right)}{\log(2)}$$

選択された実行速度と PTPER 値ごとの PWM の分解能および周波数は表 15-2 に示されています。表 15-2 の PWM 周波数は、エッジ整列された（フリーラン PTMR）PWM モードの場合の値です。中央整列モード（上昇 / 下降 PTMR モード）の場合は、PWM の周波数は表 15-2 で示される数値の半分となります。

表 15-2: PWM の周波数および分解能の例、プリスケーラ 1:1

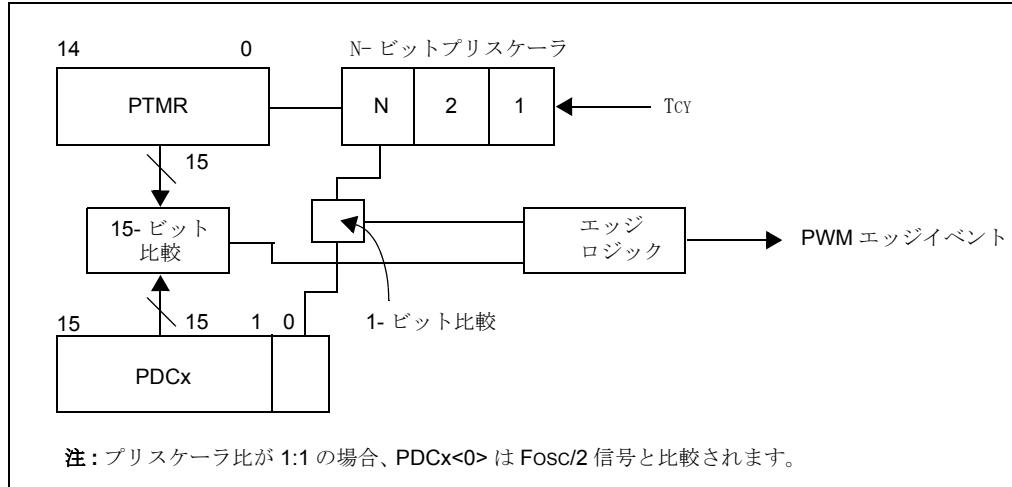
TCY (FCY)	PTPER 値	PWM 分解能	PWM 周波数
33 ns (30 MHz)	0x7FFF	16 ビット	915 Hz
33 ns (30 MHz)	0x3FF	11 ビット	29.3 KHz
50 ns (20 MHz)	0x7FFF	16 ビット	610 Hz
50 ns (20 MHz)	0x1FF	10 ビット	39.1 KHz
100 ns (10 MHz)	0x7FFF	16 ビット	305 Hz
100 ns (10 MHz)	0xFF	9 ビット	39.1 KHz
200 ns (5 MHz)	0x7FFF	16 ビット	153 Hz
200 ns (5 MHz)	0x7F	8 ビット	39.1 KHz

注： 中央整列演算の場合、PWM 周波数は示されている数値の半分となります。

MCPWM モジュールにより、解像度  $TCY/2$  で PWM 信号エッジを発生させることができます。PTMR はプリスケーラが 1:1 の場合、TCY 毎にカウントアップします。エッジ解像度が  $TCY/2$  に達するためには、 $PDCx<15:1>$  が  $PTMR<14:0>$  と比較され、デューティの一一致が決められます。 $PDCx<0>$  により、PWM 信号エッジを TCY で発生するか、 $TCY/2$  の境界で発生するかが決まります。1:4, 1:16 または 1:64 のプリスケーラが PWM タイムベースとともに用いられた場合、 $PDCx<0>$  がプリスケーラカウンタクロックの Msbit と比較され、PWM エッジを発生するかどうか決まります。

注： MCPWM は  $TCY/2$  エッジ分解能で PWM 信号を発します。

図 15-5: デューティ比較ロジック

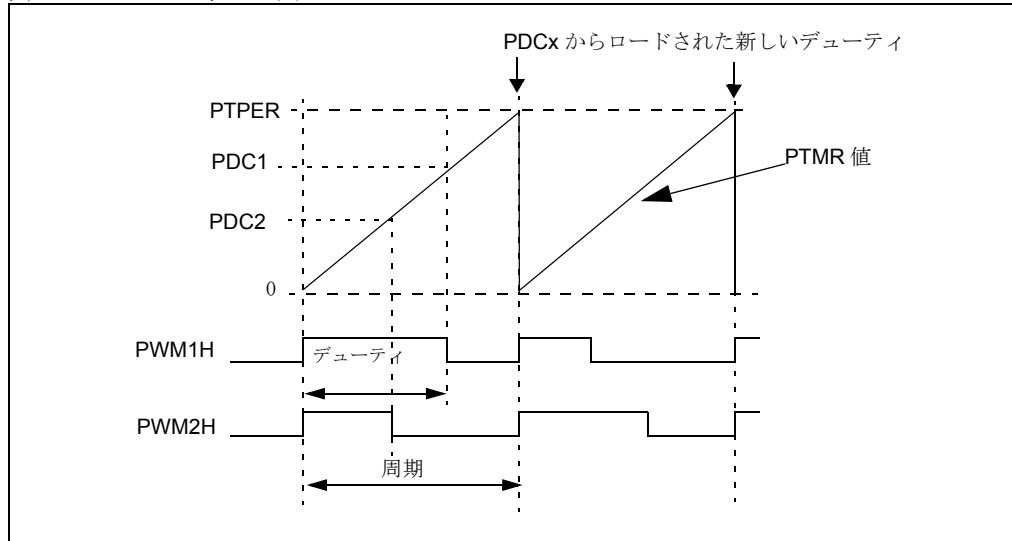


#### 15.4.2 エッジ整列 PWM

エッジ整列 PWM 信号は、PWM タイムベースがフリーランモードで動作しているときに、モジュールにより出力されます。PWM チャンネルにより出力される PWM 信号の周期は、PTPER にロードされる値により決まり、デューティは適切なPDCxレジスタにより決まります(図 15-6 参照)。デューティゼロでなければ、どの PWM ジェネレータの出力も、PWM 周期の初めにアクティブにされます (PTMR = 0)。各 PWM 出力は、PTMR 値が PWM ジェネレータのデューティ値と一致した場合にインアクティブとなります。

PDCx レジスタの値がゼロの場合、対応する PWM ピンの出力は PWM 周期全体にわたってインアクティブになります。さらに PWM ピンの出力は、PDCx レジスタの値が PTPER レジスタの値より大きい場合には PWM 周期全体にわたってアクティブとなります。

図 15-6: エッジ整列 PWM

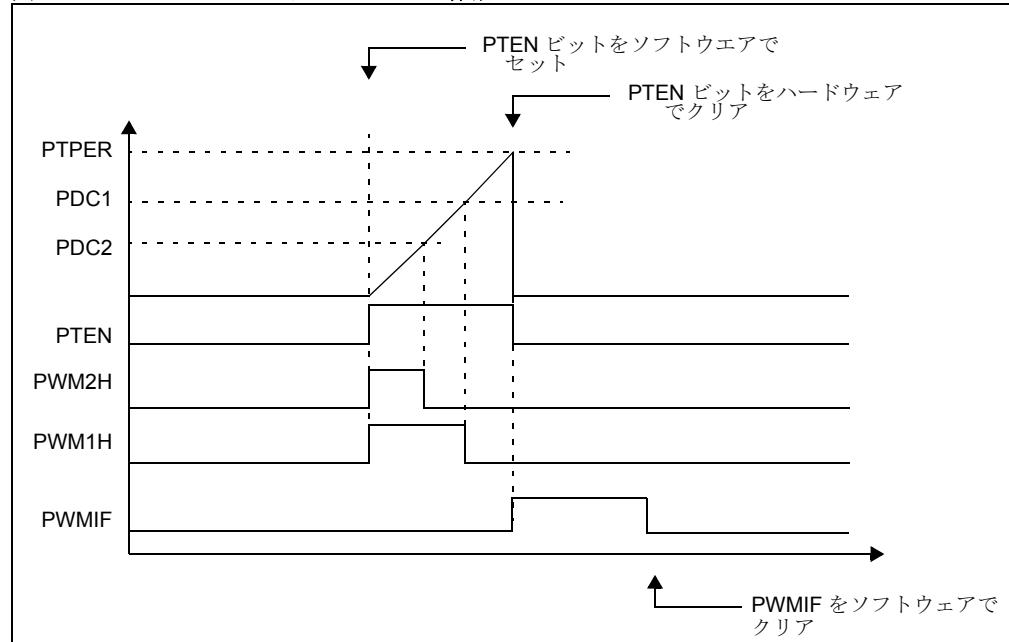


### 15.4.3 シングルイベント PWM 動作

PWM タイムベースがシングルイベントモードに設定されている場合 ( $\text{PTMOD}_{1:0} = 01$ )、PWM モジュールによりシングルパルス出力が出力されます。この動作モードは、ある種のブランシモータを動かすために有用です。特にこのモードは高速スイッチドリラクタンス(SR)モーター制御に有用です。シングルイベントモードでは、エッジ整列出力のみが発生します。

シングルイベントモードでは、**PTEN** ビットがセットされた場合に PWM 入出力ピンがアクティブとなります。デューティレジスタとの一致が発生した場合、PWM 入出力ピンはインアクティブとなります。**PTPER** レジスタとの一致が発生した場合、**PTMR** レジスタはクリアされ、アクティブな PWM 入出力ピンは全てインアクティブとされ、**PTEN** ビットはクリアされ、割込みが発生します。PWM モジュールの動作は、**PTEN** が再びソフトウェアにセットされるまで停止します。

図 15-7: シングルイベント PWM 作動



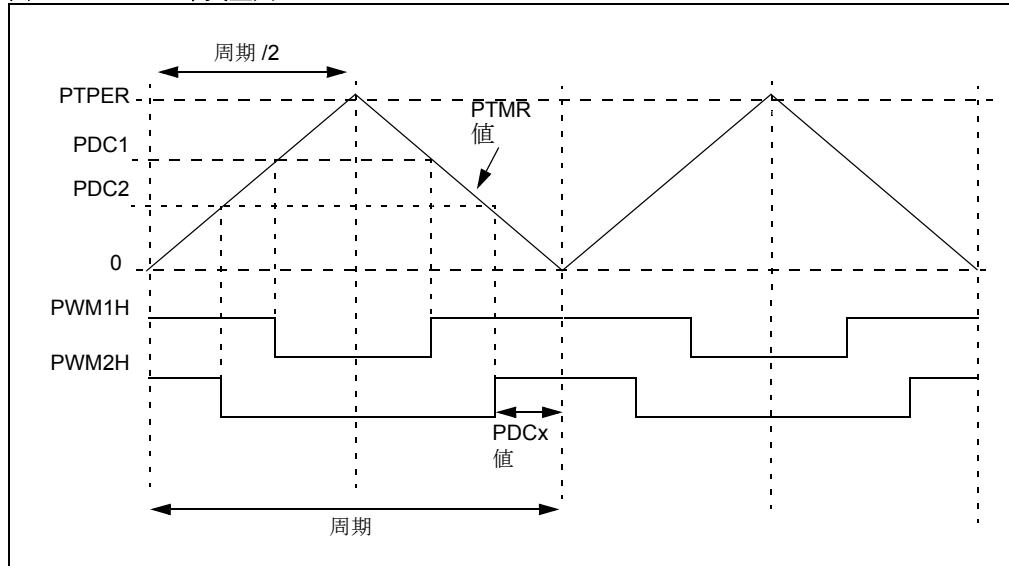
#### 15.4.4 中央整列 PWM

PWM タイムベースがアップカウントモードまたはダウンカウントモードのいずれかに設定されている場合( $PTMOD<1:0> = 1x$ )、PWM モジュールにより中央整列 PWM 信号が出力されます。

PWM 比較出力は、デューティレジスタの値が PTMR の値と一致し、PWM タイムベースが下方にカウントされているとき( $PTDIR = 1$ )にアクティブとなります。PWM 比較出力は、PWM タイムベースが下方にカウント中で( $PTDIR = 0$ )、PTMR レジスタの値がデューティ値と一致した場合にインアクティブとなります。

あるデューティレジスタの値がゼロである場合、対応する PWM ピンの出力は PWM 周期全体にわたってインアクティブとなります。さらにデューティサイクルの値が PTPER レジスタの値よりも大きくなると、PWM ピンの出力は PWM 周期全体にわたってアクティブとなります。

図 15-8: 中央整列 PWM



#### 15.4.5 デューティレジスタバッファリング

PDC1 から PDC4 の 4 つの PWM デューティレジスタはバッファされ、PWM 出力をグリッチなしでアップデートが可能となります。ジェネレータごとにユーザーがアクセス可能な PDCx レジスタと、実際の比較値を保持しているメモリーマップされていないデューティレジスタがあります。PWM デューティレジスタは、PWM 出力信号にグリッチが出ないように、PWM 周期内のある特定のタイミングで PDCx レジスタからロードされて更新されます。

PWM タイムベースがフリーランモードまたはシングルイベントモードで作動中の場合( $PTMOD<1:0> = 0x$ )、PWM デューティは PTPER レジスタとの一致が発生し、PTMR が ‘0’ にリセットされるときに毎回アップデートされます。

**注:** PWM タイムベースが無効化されている時は( $PTEN = 0$ )、PDCx レジスタに書き込みがあると直ちにデューティがアップデートされます。これにより PWM 信号発生が有効化される前にデューティの変更をさせることができます。

PWM タイムベースがアップ / ダウンカウントモードで動作中は( $PTMOD<1:0> = 10$ )、PTMR レジスタの値がゼロになり、PWM タイムベースが上方にカウントを開始した時にデューティがアップデートされます。図 15-9 は、PWM タイムベースがこのモードに設定されている場合にデューティのアップデートが発生するタイミングを示したものです。

PWM タイムベースがダブルアップデートを伴うアップ / ダウンカウントモードに設定されている場合( $PTMOD<1:0> = 11$ )、PTMR レジスタの値がゼロになり、PTMR レジスタの値が PTPER レジスタの値と一致した場合にデューティがアップデートされます。図 15-10 は、PWM タイムベースがこのモードに設定されている場合にデューティのアップデートが発生するタイミングを示したものです。

図 15-9: 上方 / 下方カウントモードにおけるデューティアップデートのタイミング

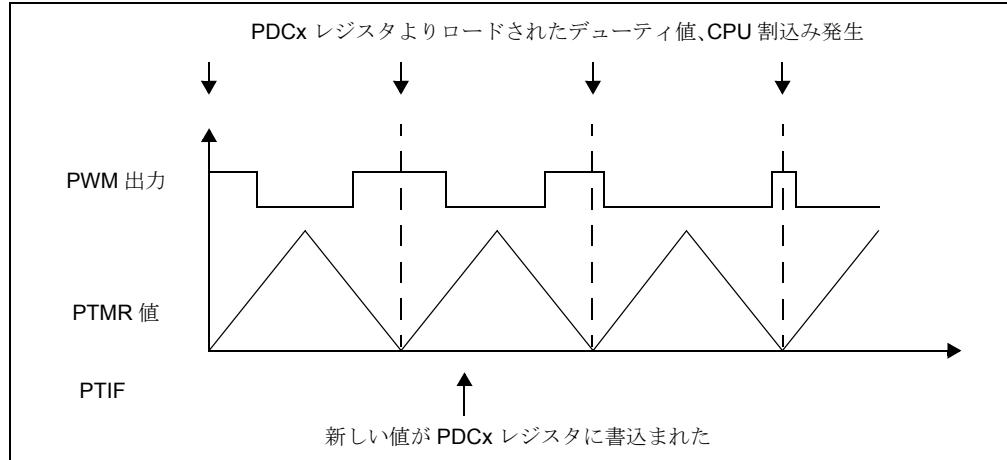
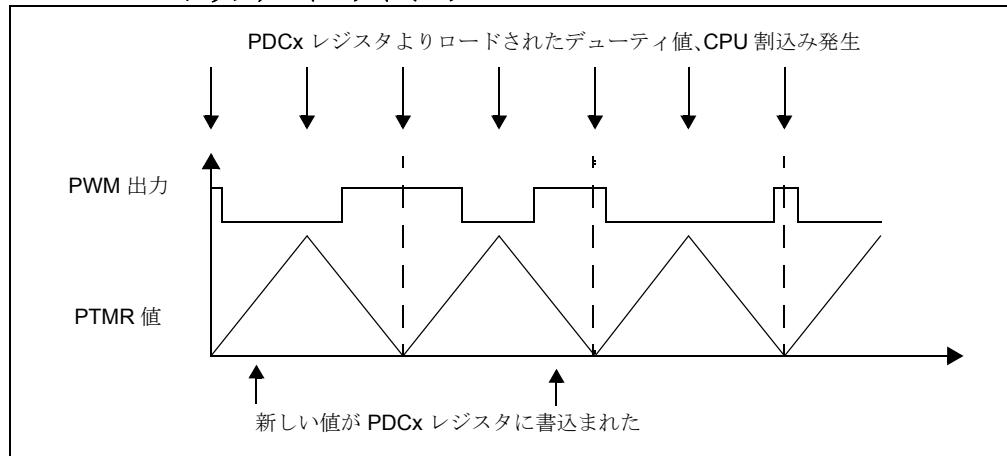


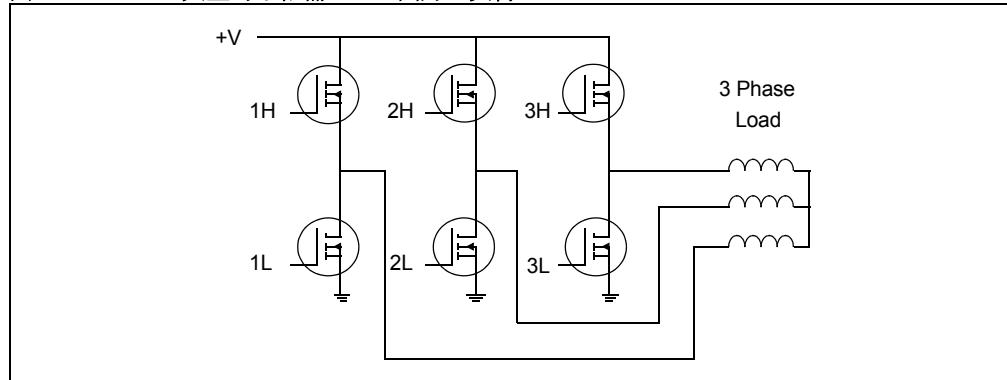
図 15-10: ダブルアップデートを伴うアップ / ダウンカウントモードにおけるデューティアップデートのタイミング



## 15.5 相補 PWM 出力モード

相補 PWM 出力モードは、図 15-11 に示されたものと類似するインバータ負荷を駆動させるために用いられます。このインバータのトポロジーは、ACIM および BLDC アプリケーションに典型的なものです。相補 PWM 出力モードでは、PWM 出力装置のペアは同時にアクティブにすることはできません。各 PWM チャンネルおよび出力ピンのペアの内部の構成は図 15-12 のようになっています。デバイス切替え時には、短時間両方の出力がインアクティブになるよう任意にデッドタイムを入れることができます（セクション 15.6 「デッドタイム制御」参照）。

図 15-11: 典型的な相補 PWM 出力の負荷



相補モードは PWMCON1 の対応する PMODx ビットをクリアすることで、各 PWM 入出力ピンのペア毎に選択できます。PWM 入出力ピンは、デバイスリセット時にデフォルトで相補モードにセットされるように設定されています。

図 15-12: PWM チャンネルブロック図、相補モード



## 15.6 デッドタイム制御

PWM 入出力ピンの対が相補出力モードで動作中には、デッドタイムの発生は自動的に有効化されます。パワー出力デバイスは瞬時に切替えができないため、相補ペアにおける PWM 出力のターンオフイベントと他のトランジスタのターンオンイベントの間に多少時間が必要となります。

6 出力 PWM モジュールは、プログラム可能なデッドタイムが 1 組あります。8 出力 PWM モジュールには、異なる 2 組のデッドタイムをプログラムすることができます。この 2 組のデッドタイムは、ユーザーの柔軟性が高まるように、以下の 2 方法のいずれかで用いることができます。

- PWM 出力信号は、ハイサイドトランジスタおよびローサイトトランジスタの異なるターンオフ時間間に最適化することができます。最初のデッドタイムは、相補ペアの下側にあるトランジスタのターンオフイベントと上側のトランジスタのターンオンイベントの間となります。2 回目のデッドタイムは、上側のトランジスタのターンオフイベントと下側のトランジスタのターンオンイベントの間となります。
- 2 組のデッドタイムは、個別の PWM 入出力ピンのペアに割り振ることができます。この動作モードでは、PWM モジュールで異なるトランジスタや負荷の組合せを、各相補 PWM 出力ピンペアにより駆動させることができます。

### 15.6.1 デッドタイムジェネレータ

PWM モジュールの各相補出力ペアには 6 ビットのダウンカウンタが用意されていて、デッドタイムの挿入に用いられています。図 15-13 で示したように、各デッドタイムユニットには立上がりエッジおよび立下がりエッジの検出器があり、デューティ比較出力部と結合されています。

PWM エッジイベントが検出されると、取り得る 2 つのデッドタイムのうち 1 つがタイマーに読み込まれます。エッジが立上がりか立下がりかによって、相補出力の遷移の 1 つが、タイマーのカウントがゼロになるまで延期されます。1 対の PWM 出力に対するデッドタイム挿入を示すタイミングに関する図表が図 15-14 に示されています。立上がりエッジイベントと立下がりエッジイベントに対し、2 つの異なるデッドタイムを用いることが、図表ではわかりやすいように誇張されて示されています。

図 15-13: 一対の出力ピンのデッドタイムユニット回路図

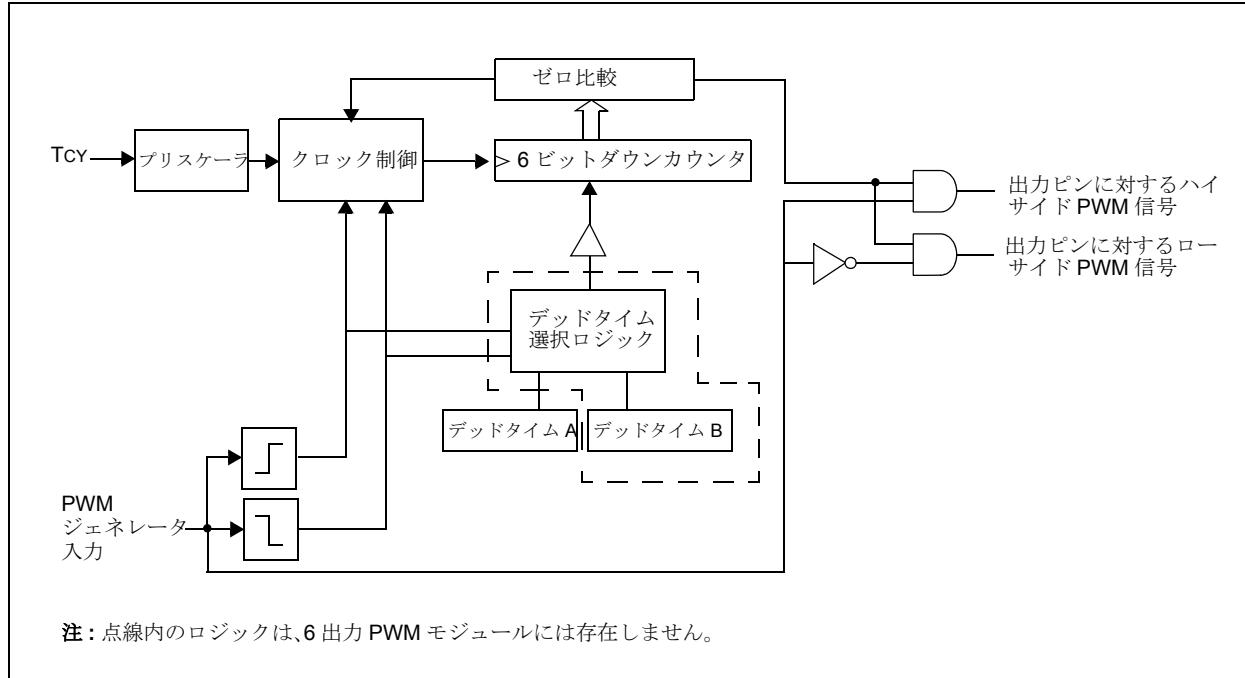
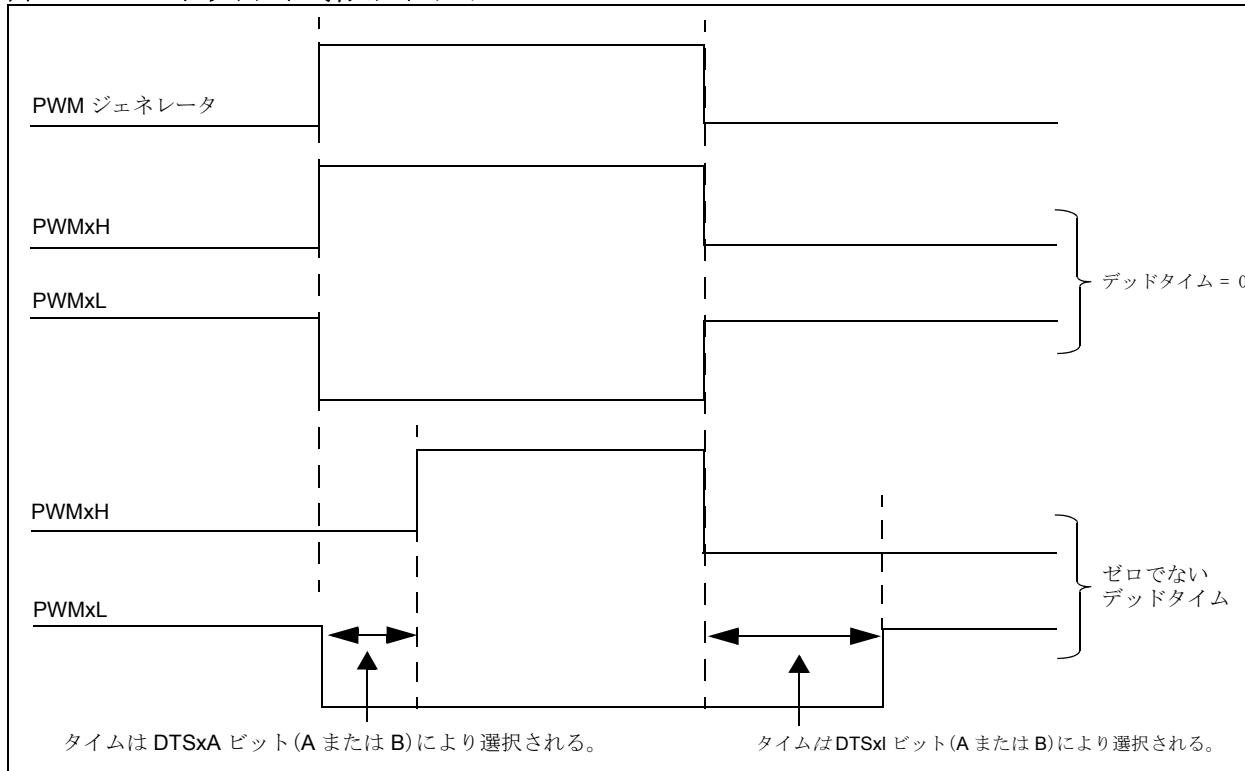


図 15-14: デッドタイム挿入タイミング



### 15.6.2 デッドタイム割当て

**注:** デッドタイム割当てロジックは、8 出力 PWM モジュールを含む dsPIC にのみ適用可能です。6 出力 PWM モジュールではデッドタイム A しか使えません。

DTCON2 レジスタには制御ビットが含まれ、プログラム可能な 2 組のデッドタイムを相補出力それぞれに割り当てる役割を果たします。各相補出力には 2 つのデッドタイム割当て制御ビットがあります。例えば DTS1A および DTS1I 制御ビットにより、PWM1H/PWM1L 相補出力ペアに用いられるデッドタイムが選択されます。デッドタイム選択制御に用いられるペアビットは、それぞれ「アクティブ時デッドタイム選択」制御ビット、「インアクティブ時デッドタイム選択」制御ビットと呼ばれています。ペアビットの機能は以下の通りです。

- DTSxA 制御ビットにより、ハイサイド出力がアクティブとなる前に挿入されるデッドタイムが選択されます。
- DTSxI 制御ビットにより、ローサイド PWM アクティブがアクティブとなる前に挿入されるデッドタイムが選択されます。

表 15-3 は、各デッドタイム選択制御ビットの機能を要約して示したものです。

表 15-3: デッドタイム選択ビット

ビット	関数
DTS1A	PWM1H がアクティブにされる前に挿入される PWM1H/PWM1L デッドタイムが選択されます。
DTS1I	PWM1L がアクティブにされる前に挿入される PWM1H/PWM1L デッドタイムが選択されます。
DTS2A	PWM2H がアクティブにされる前に挿入される PWM1H/PWM1L デッドタイムが選択されます。
DTS2I	PWM2L がアクティブにされる前に挿入される PWM1H/PWM1L デッドタイムが選択されます。
DTS3A	PWM3H がアクティブにされる前に挿入される PWM1H/PWM1L デッドタイムが選択されます。
DTS3I	PWM3L がアクティブにされる前に挿入される PWM1H/PWM1L デッドタイムが選択されます。
DTS4A	PWM4H がアクティブにされる前に挿入される PWM1H/PWM1L デッドタイムが選択されます。
DTS4I	PWM4L がアクティブにされる前に挿入される PWM1H/PWM1L デッドタイムが選択されます。

### 15.6.3 デットタイム範囲

デッドタイム A とデッドタイム B は、入力クロックプリスケーラ値と 6 ビットの符号なしデットタイムカウント値を選択することにより設定されます。

デバイスの動作周波数に基づき、4 種類の入力クロックプリスケーラ選択が可能となり、適切なデットタイムの範囲を決めることができます。クロックプリスケーラのオプションは、2 組のデットタイム値それぞれ別々に選択することができます。デッドタイムクロックプリスケーラは、DTCON1 SFR において DTAPS<1:0> および DTBPS<1:0> 制御ビットを用いて選択されます。以下のクロックプリスケーラオプションが、各デッドタイム値に対し選択することができます。

- TCY
- 2 TCY
- 4 TCY
- 8 TCY

式 15-3: デッドタイムの算出

$$DT = \frac{\text{デッドタイム}}{\text{プリスケール値} \cdot TCY}$$

注: DT(デッドタイム)は、DTA<5:0> または DTB<5:0> のレジスタ値です。

表 15-4 は、選択される入力クロックプリスケーラおよびデバイス動作周波数の関数としてのデッドタイム範囲の例を示すものです。

表 15-4: デッドタイム範囲の例

T <sub>CY</sub> (F <sub>CY</sub> )	プリスケーラ選択	解像度	デッドタイム範囲
33 ns (30 MHz)	4 TCY	130 ns	130 $\mu$ s-9 usec
50 ns (20 MHz)	4 TCY	200 ns	200 $\mu$ s-12 usec
100 ns (10 MHz)	2 TCY	200 ns	200 $\mu$ s-12 usec

#### 15.6.4 デッドタイムのひずみ

小さな PWM デューティサイクルの場合、能動的 PWM 時間にに対するデッドタイムの比が大きくなってしまうこともあります。極端な場合、デューティデッドタイムがプログラムされたデューティより小さいか等しい時は、PWM パルスが発生しないこともあります。こうした場合には挿入されたデッドタイムにより、PWM モジュールにより生成した波形にひずみが生じます。PWM デューティをデッドタイムと比べて最低 3 倍の大きさに保つことにより、ユーザーはデッドタイムのひずみを最小限にすることができます。デッドタイムのひずみは、この他クローズループ電流制御などの手法を用いて修正することもできます。

同様の現象は、デューティサイクルが 100% 近いときにも発生します。PWM 信号の最低インアクティブ時間がデッドタイムと比べて最低 3 倍は長くなるように、アプリケーションに用いられる最大デューティサイクルを決める必要があります。

#### 15.7 独立 PWM 出力モード

独立 PWM 出力モードは、図 15-15 に示されるような負荷を駆動させるためには有用です。PWM 出力ペアは、PWMCON1 レジスタにある対応する PMOD ビットがセットされているとき、独立出力モードとなります。独立モードではデッドタイムジェネレータは無効化されており、出力ピンの対のピンの状態について制限はありません。

図 15-15: 非対称インバータ

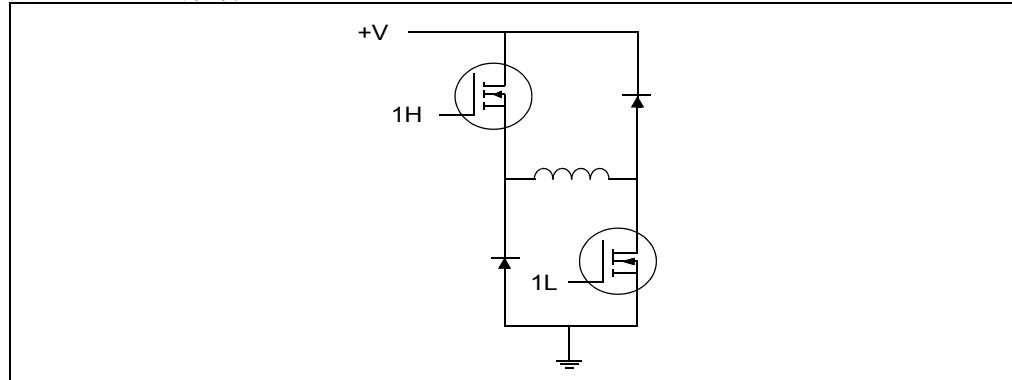
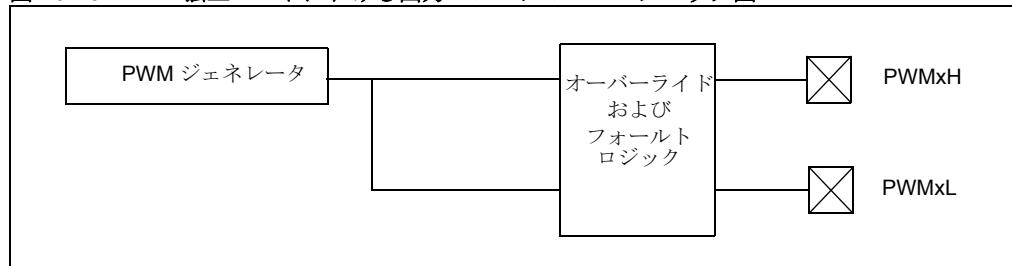


図 15-16: 独立モードにおける出力ピンペアの PWM ブロック図



## 15.8 PWM 出力オーバーライド

PWM 出力オーバーライドビットは、デューティサイクル比較ユニットと関係なく、ユーザーが手動で PWM 入出力ピンを特定のロジック状態に切り替えることを可能とするものです。PWM オーバーライドは、様々な整流子モーターを制御するにあたり有用です。

PWM 出力オーバーライド機能と関連する全ての制御ビットは、OVDCON レジスタに含まれています。OVDCON レジスタの上位半分には 8 ビットの POVDxx が含まれ、どの PWM 入出力ピンがオーバーライドされるか決定する役割を持っています。OVDCON レジスタの下位半分には 8 ビットの POUTxx が含まれ、POVDxx ビットによりオーバーライドされた場合の PWM 入出力ピンの状態を決定します。

POVD ビットはアクティブを Low にする制御ビットです。POVD ビットがセットされた場合、対応する POUTxx ビットは PWM 出力に影響を及ぼしません。POVD ビットのうち 1 つがクリアされた場合、対応する PWM 入出力ピンの出力は POUT ビットの状態により決められます。POUT ビットがセットされたとき、PWM ピンはアクティブの状態になります。POUT ビットがクリアされた場合、PWM ピンはインアクティブの状態になります。

### 15.8.1 相補出力モードの場合のオーバーライド制御

PWM モジュールにより、一対の PWM 入出力ピンが相補モードで動作中 ( $PMODx = 0$ )、ある種のオーバーライドが発生しなくなります。ペア出力の両方を同時にアクティブにすることはできません。各出力装置ペア中のハイサイドピンが常に優先されます。

注： PWM チャンネルが手動でオーバーライドされた場合も、デッドタイム挿入は実行されます。

### 15.8.2 オーバーライド同期化

OSYNC ビットがセットされた場合 ( $PWMC0N2<1>$ )、OVDCON レジスタを通して実行された全ての出力オーバーライドは、PWM タイムベースと同期化されます。同期化された出力オーバーライドは以下の場合一に発生します。

- エッジ配列モードの場合、PTMR がゼロの時
- 中央配列モードの場合、PTMR がゼロのまたは
- PTMR 値が PTPER と一致した場合

オーバーライド同期化機能が有効化された場合、PWM 出力ピンに予期しない細いパルスが出力されるのを避けることができます。

### 15.8.3 出力オーバーライドの例

図 15-17 は PWM 出力オーバーライド機能を用いた場合の波形の例を示したものです。BLDC モーターの 6 段階の転流シーケンスが図中に示されています。モーターは図 15-11 に示されている三相のインバータを用いて駆動されています。適切なローター位置が検出されると、PWM 出力は次の転流状態に切り替えられます。この例では、PWM 出力が特定のロジック状態になっています。図 15-17 の信号を発生させるために用いられた OVDCON レジスタの値は、表 15-5 に示されています。

PWM デューティレジスタは、OVDCON レジスタと共に用いられることがあります。デューティレジスタは負荷に送られる電流を制御し、OVDCON レジスタは転流を制御します。その例が図 15-18 に示されています。図 15-18 の信号を発生させるために用いられる OVDCON レジスタ値は表 15-6 に示されています。

表 15-5: PWM 出力オーバーライドの例 #1

状態	OVDCON<15:8>	OVDCON<7:0>
1	00000000b	00100100b
2	00000000b	00100001b
3	00000000b	00001001b
4	00000000b	00011000b
5	00000000b	00010010b
6	00000000b	00000110b

図 15-17: PWM 出力オーバーライドの例 #1

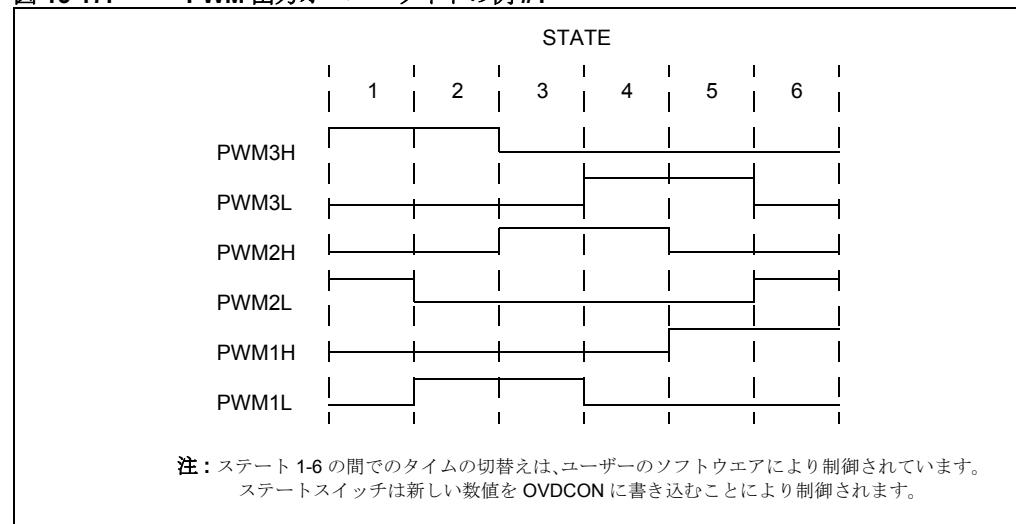
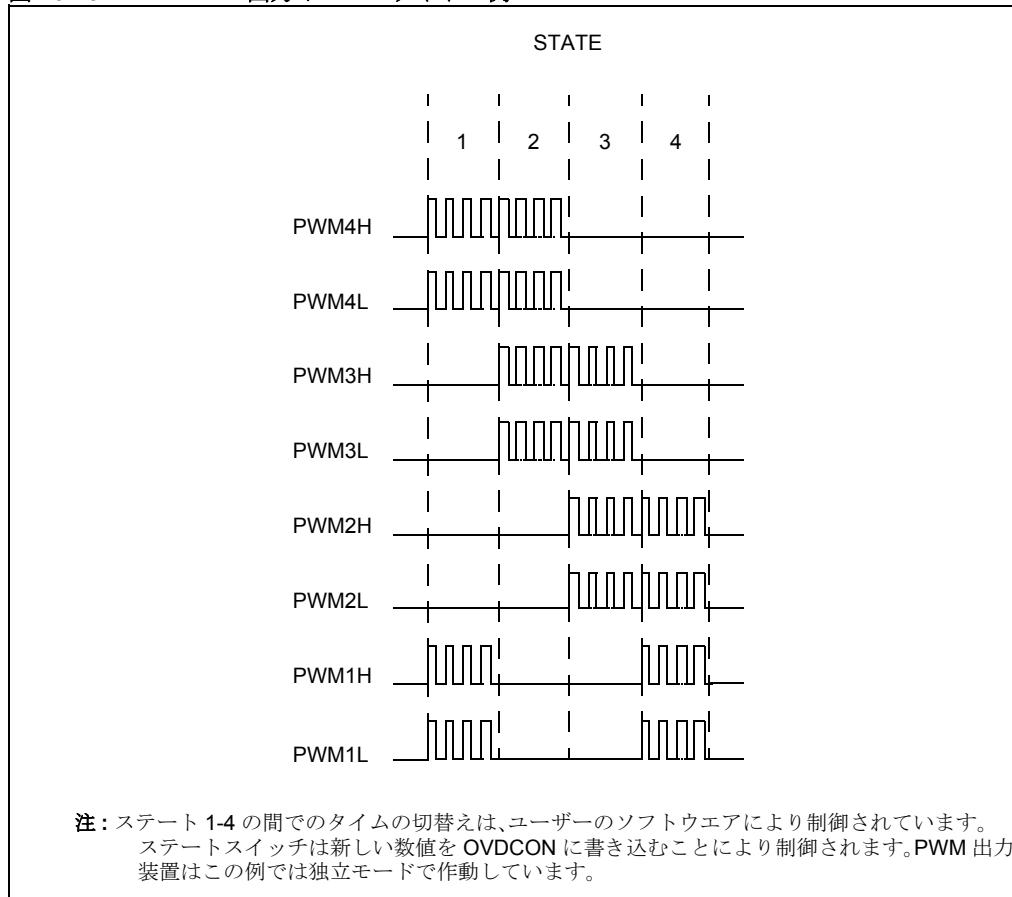


表 15-6: PWM 出力オーバーライドの例 #2

状態	OVDCON<15:8>	OVDCON<7:0>
1	11000011b	00000000b
2	11110000b	00000000b
3	00111100b	00000000b
4	00001111b	00000000b

図 15-18: PWM 出力オーバーライドの例 #2



## 15.9 PWM 出力極性制御

PWMC0N1 における PEN<sub>xx</sub> 制御ビットにより、モジュールにより用いられる各 PWM 出力ピンが有効化されます。ピンが PWM 出力用に有効化されると、ピンを制御する PORT および TRIS レジスタが無効化されます。

PEN<sub>xx</sub> 制御ビットの他に、FBORPOR デバイスコンフィギュレーションレジスタには、PWM 出力ピン制御を行う 3 つのデバイスコンフィギュレーションビットがあります。

- HPOL コンフィギュレーションビット
- LPOL コンフィギュレーションビット
- PWMPIN コンフィギュレーションビット

これら 3 つのコンフィギュレーションビットは、PWMC0N1 にある PWM 有効化ビット (PEN<sub>xx</sub>) と連動して作用します。コンフィギュレーションビットにより、デバイスリセット発生後に PWM ピンが正しい状態に保たれます。

### 15.9.1 出力極性制御

PWM 入出力ピンの極性は、FBORPOR デバイスコンフィギュレーションレジスタ中の HPOL および LPOL コンフィギュレーションビットがデバイスプログラムする中で設定されます。HPOL 設定ビットにより、ハイサイド PWM 出力の PWM1H から PWM4H までの出力極性が設定されます。LPOL 設定ビットにより、ローサイド PWM 出力である PWM1L から PWM 4L までの出力極性が設定されます。

極性設定ビットが ‘1’ にプログラムされた場合、対応する PWM 入出力ピンはアクティブ時 High の極性を有するようになります。極性設定ビットが ‘0’ にプログラムされた場合、対応する PWM ピンはアクティブ時 Low の極性を有するようになります。

### 15.9.2 PWM 出力ピンリセットステート

PWMPIN コンフィギュレーションビットにより、デバイスリセット時の PWM 出力ピンの動作が決定され、また PWM モジュールに制御されるデバイスに接続された外部のプルアップ / プルダウンレジスタを省くためにも用いることができます。

PWMPIN コンフィギュレーションビットが ‘1’ にプログラムされている場合、PEN<sub>xx</sub> 制御ビットはデバイスリセット時にクリアされます。その結果、全ての PWM 出力はトライステートの汎用入出力ピンとなり、対応する PORT および TRIS レジスタにより制御されます。

PWMPIN コンフィギュレーションビットが ‘0’ にプログラムされている場合、PEN<sub>xx</sub> 制御ビットがデバイスリセット時にセットされます。これによりデバイスリセット時に全ての PWM ピンが PWM 出力可能となり、HPOL および LPOL 設定ビットにより定義されたインアクティブ時の状態になります。

## 15.10 PWM フォールトピン

フォールトピンは FLTA および FLTB の 2 種類があり、PWM モジュールと関連しています。有効化された場合、これらピンは任意で各 PWM 入出力ピンを定義された状態にすることができます。このような動作はソフトウェアがなくても起きるので、異常が発生したときにすばやく対処することができます。

フォールトピンは dsPIC デバイスのタイプにより、その他多様な機能を持っていることがあります。フォールト入力として用いられた場合、各フォールトピンは対応する PORT レジスタを通して読み出し可能です。FLTA および FLTB ピンは負論理入力として機能するため、外部のプルアップレジスタを通してワイヤード OR された数多くのソースを同一の入力につなげることができます。PWM モジュールと共に用いられない場合は、これらのピンは汎用入出力ピン、あるいは負論理の割込みピンとして用いることが可能です。各フォールトピンには割込みベクター、割込みフラグビット、割込み有効化ビットおよびこのビットと関連する割込み優先ビットを備えています。

FLTA ピンの機能は、FLTACON レジスタにより制御され、FLTB ピンの機能は FLTBCON レジスタにより制御されています。

### 15.10.1 フォールトピン有効化ビット

FLTACON および FLTBCON レジスタには FxEN1 から FxEN4 までそれぞれ 4 つの制御ビットがあり、この制御ビットにより、ある一対の PWM 入出力ピンがフォールト入力ピンにより制御されるか決定されます。ある一対の PWM 入出力ピンを有効化し、フォールトオーバーライドを可能にするためには、FLTACON または FLTBCON レジスタに対応するビットをセットする必要があります。

FLTACON または FLTBCON の全てのビットがクリアされた場合、フォールト入力ピンは PWM モジュールに影響を与えず、ピンは汎用の割込みピンまたは入出力ピンとして用いることができます。

### 15.10.2 フォールト状態

FLTACON および FLTBCON 特殊機能レジスタにはそれぞれ 8 ビットが割り当てられ、フォールト入力ピンが有効化された場合の各 PWM 入出力ピンの状態を決定します。これらのビットがクリアされた場合、PWM 入出力ピンはインアクティブの状態となります。ビットがセットされると PWM 入出力ピンはアクティブとなります。アクティブとインアクティブの状態は、各 PWM 入出力ピンごとに設定されている極性により決まります (HPOL および LPOL デバイスコンフィギュレーションビットにより設定)。

PWM モジュール入出力動作が相補モードで行われ、両方のピンがフォールト状態でアクティブの状態となるようにプログラムされた場合は特別な場合となります。相補モードではハイサイドピンが常に優先されるので、入出力ピン両方が同時にアクティブの状態になることはできません。

### 15.10.3 フォールト入力モード

各フォールト入力ピンには 2 つの動作モードがあります。

- **ラッチモード**: フォールトピンが Low の場合、PWM 出力は FLTxCON レジスタで定義された状態になります。PWM 出力はフォールトピンが High になり、かつ対応する割込みフラグ (FLTxIF) がソフトウェアでクリアされるまでこの状態にとどまります。これら両方の動作が発生した場合、次の PWM 周期または半周期の始めに PWM 出力は通常の動作に戻ります。フォールト状態終了前に割込みフラグがクリアされた場合、フォールト状態が終了するまで PWM モジュールは出力を復帰させません。
- **サイクルバイサイクルモード**: フォールト入力ピンが Low の場合、PWM 出力はフォールトピンが Low を維持する限り、定義されたフォールト状態にとどまります。フォールトピンが High になった後は、PWM 出力は次の PWM 周期（中央整列モードの場合は半周期の境界）の開始時に元の動作に戻ります。

各フォールト入力ピンの動作モードは、FLTAM および FLTBM 制御ビット (FLTACON<7> および FLTBCON<7>) を用いて選択されます。

#### 15.10.3.1 フォールト状態へのエントリ

フォールトピンが有効化され、Low になった場合、PDCx および OVDCON レジスタの値に関係なく PWM ピンは直ちにプログラムされたフォールト状態になります。フォールト動作は、他の全ての PWM 制御レジスタに優先されます。

#### 15.10.3.2 フォールト状態の終了

フォールト状態をクリアするためには、外部の回路によりフォールト入力ピンが High となり、フォールト割込みフラグ（ラッチモードのみ）をクリアする必要があります。フォールトピンの状態がクリアされた後、次の PWM 周期または半周期の境界時に PWM モジュールは PWM 出力信号を元に戻します。エッジ整列 PWM 出力の場合、PTMR = 0 になると、PWM 出力が元に戻ります。中央整列 PWM 出力の場合、PTMR = 0 または PTMR = PTPER のどちらが先であっても、いずれかのイベントが発生した場合に PWM 出力は元に戻ります。

PWM タイムベースが無効化されている場合 (PTEN = 0) はこの規則の例外です。PWM タイムベースが無効化されている場合、フォールト状態がクリアされると直ちに PWM モジュールにより PWM 出力信号が元に戻されます。

### 15.10.4 フォールトピン優先順位

両方のフォールト入力ピンが、ある一対の PWM ピンを制御するように割り当てられた場合、FLTA 入力ピンにプログラムされたフォールト状態が、FLTB 入力ピンに優先します。

フォールト A 状態がクリアされた場合、2 つの動作のうち 1 つが発生します。FLTB 入力が依然としてフォールト状態の場合は、PWM 出力は次の周期または半周期の境界で、FLTBCON レジスタでプログラムされている状態に戻ります。FLTB 入力がフォールト状態でない場合は、次の周期または半周期の境界で PWM 出力は通常の動作に戻ります。

**注：** FLTA ピンがラッチモードにプログラムされている場合、PWM 出力はフォールト A 割込みフラグがクリアされ、FLTA ピンがフォールトでなくなるまで、PWM 出力はフォールト B 状態や通常の動作に復帰しません。

### 15.10.5 フォールトピンソフトウェア制御

各フォールトピンは、ソフトウェア中で手動で制御することができます。各フォールト入力は PORT 入出力ピンと共有されているため、PORT ピンは対応する TRIS ビットをクリアすることで出力用に設定が可能です。ピンの PORT ビットがクリアされると、フォールト入力が有効となります。

**注:** ソフトウェアでのフォールト入力を制御するにあたって、ユーザーは注意を払うことが必要です。フォールトピン用の TRIS ビットがクリアされた場合、フォールト入力は外部から行うことはできません。

### 15.10.6 フォールトタイミングの例

図 15-19: フォールトタイミングの例、サイクルバイサイクルモード

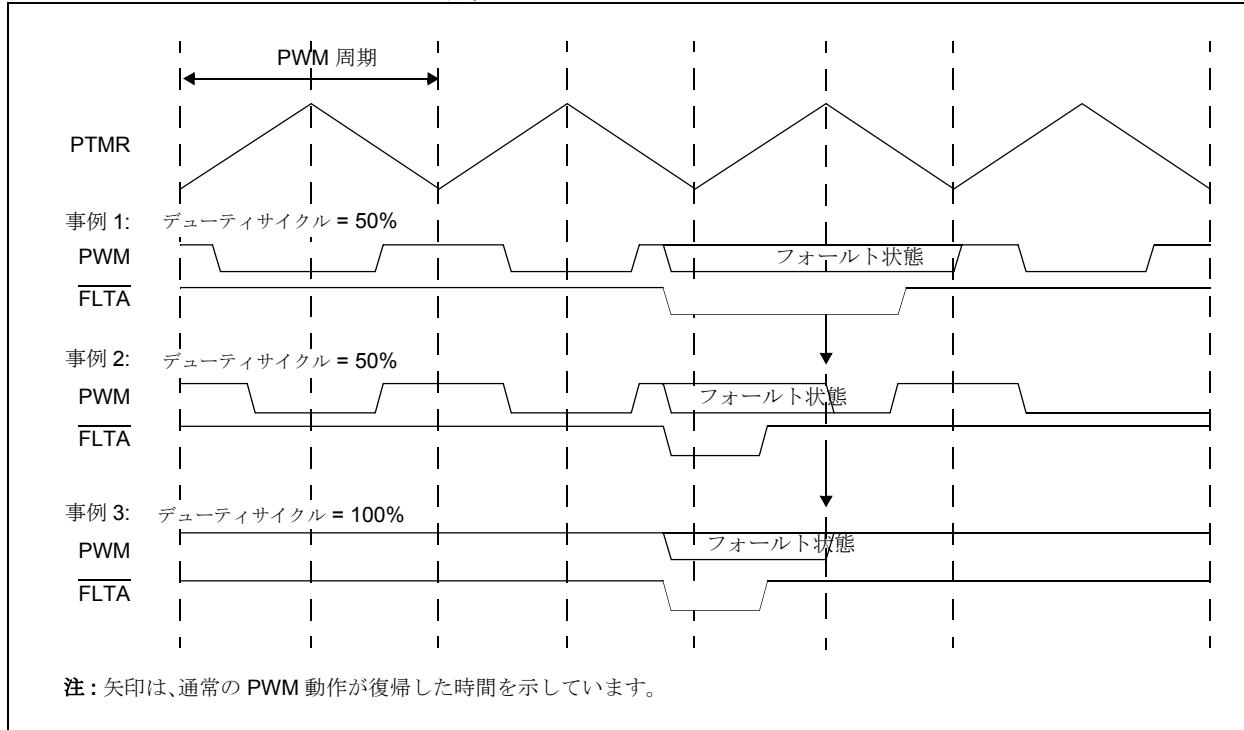


図 15-20: フォールトタイミングの例、ラッチモード

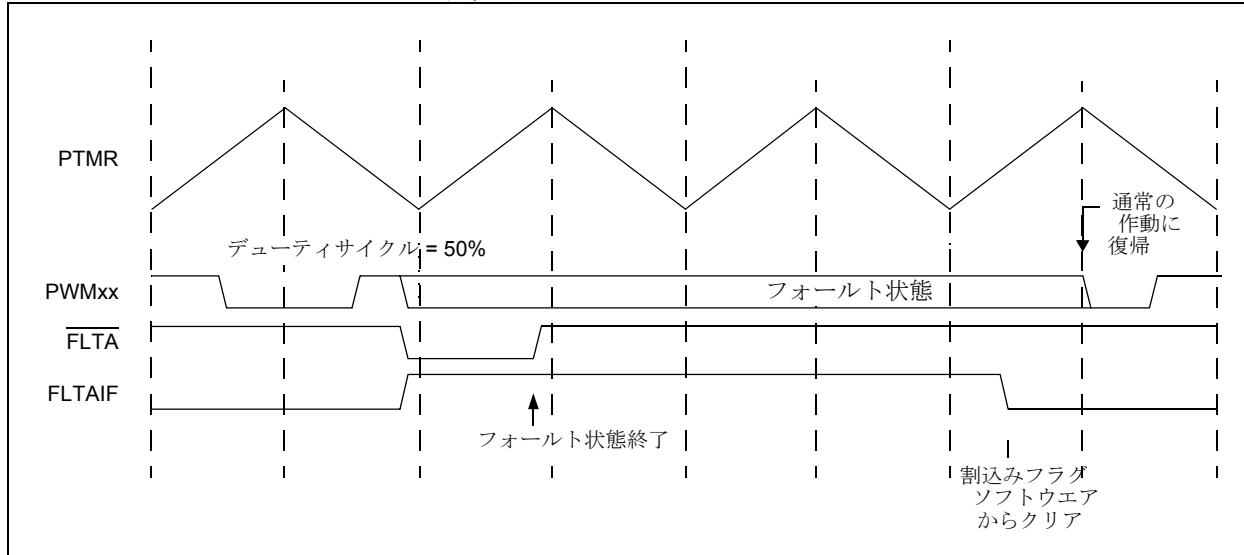
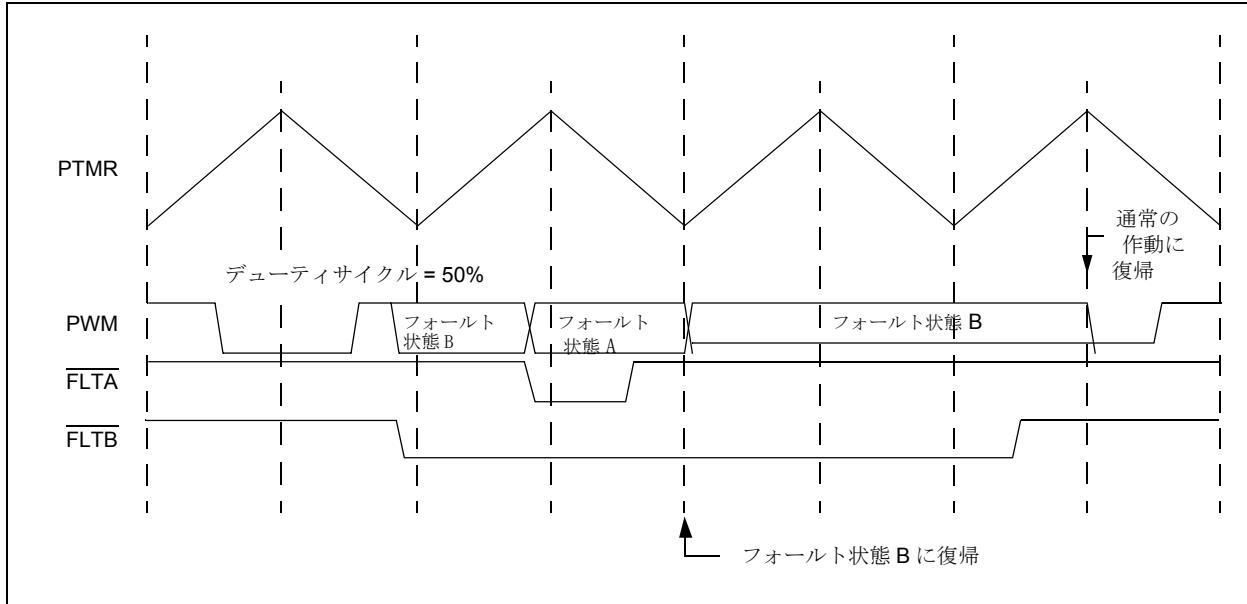


図15-21: フォールトタイミング、サイクルバイサイクルモード、優先動作の例



### 15.11 PWM アップデートロックアウト

アプリケーションによっては、全てのデューティサイクルと周期レジスタが、新しい値が効力を有する前に書き込まれることが重要となる場合があります。アップデート無効化機能により、ユーザーは新しいデューティサイクルおよび期間値がいつモジュールにより用いられるか特定することができます。PWM アップデートロックアウト機能は、UDIS 制御ビットをセットする (PWMCON2<0>) ことにより有効化されます。

UDIS ビットは PDC1-PDC4 まで全てのデューティサイクルレジスタ、および PWM タイムベースピリオドバッファ PTPER に影響を及ぼします。アップデートロックアウトを実行するためには、ユーザーは次の手順を踏む必要があります。

- UDIS ビットをセットする。
- 可能であれば全てのデューティサイクルレジスタと PTPER を書き込む。
- UDIS ビットをクリアし、アップデートを再び有効化する。

### 15.12 PWM 特殊イベントトリガー

PWM モジュールには特殊イベントトリガーがあり、A/D 変換の PWM タイムベースとの同期化を可能としています。A/D のサンプリングおよび変換開始の時間は、PWM 周期内の任意の時間に発生するようにプログラムすることができます。特殊イベントトリガーにより、ユーザーは A/D 変換の結果が得られる時間と、デューティ値が更新される時間の間の遅れを最小限にすることができます。

PWM 特殊イベントトリガーには SFR、SEVTCMP が 1 つ、またポストスケーラ制御ビット (SEVOPS<3:0>) が 4 つあり、イベントトリガーの動作を制御しています。特殊イベントトリガーが発生する PTMR 値は、SEVTCMP レジスタに書き込みます。

PWM タイムベースがアップ / ダウンカウントモードにある場合、特殊イベントトリガーのカウント相を特定するためには別の制御ビットが必要となります。カウント相は SEVTCMP の MSb にある SEVTDIR 制御ビットを用いて選択されます。SEVTDIR ビットがクリアされた場合、特殊イベントトリガーが PWM タイムベースのアップカウントサイクル中で発生します。SEVTDIR ビットがセットされた場合、特殊イベントトリガーは PWM タイムベースのダウンカウントサイクル中で発生します。SEVTDIR 制御ビットは、PWM タイムベースがアップまたはダウンカウントモードに設定されていない限り影響を及ぼしません。

## 15.12.1 特殊イベントトリガー有効化

PWM モジュールにより常に特殊イベントトリガー信号が発生します。この信号は A/D モジュールにより任意で用いられることがあります。特殊イベントトリガーの使用に関する情報について、詳しくは 第 17 章、「**10 ビット A/D コンバータ**」を参照してください。

## 15.12.2 特殊イベントトリガーポストスケーラ

PWM 特殊イベントトリガーはポストスケーラを備えており、1:1 から 1:16 までのポストスケール比を取ることができます。同期 A/D 変換を PWM サイクルごとに実行する必要のない場合は、ポストスケーラが有用です。ポストスケーラは、PWMC0N2 SFR に SEVOPS<3:0> 制御ビットを書き込むことで設定されます。

以下の場合、特殊イベント出力ポストスケーラはクリアされます。

- SEVTCMP レジスタへの書込み時。
- デバイスリセット時。

## 15.13 デバイス省電力モードにおける動作

### 15.13.1 スリープモードにおける PWM の動作

デバイスがスリープモードに入った場合、システムクロックは無効化されます。PWM タイムベースのクロックはシステムクロックソース (Tcy) を使っているため、PWM タイムベースのクロックも無効化されます。この時すべての有効化された PWM 出力は、スリープ直前の状態で凍結されます。

パワー・アプリケーションにおいて負荷を制御するために PWM モジュールが用いられた場合、PWM モジュールの出力を、PWRSAV 命令実行前に「安全な」状態に保つことはユーザーの責任です。アプリケーションによっては PWM 出力が特定の出力状態で凍結した場合、負荷に過大な電流が流れることもあります。例えば OVDCON レジスタは下のコード例で示されているように、手動で PWM 出力ピンをオフにするために用いられます。

; このコード例は全ての PWM ピンを PWRSAV 命令実行前にインакティブの状態にするためのも； のです。

```
CLR      OVDCON      ; 全ての PWM 出力を無効化
PWRSAV  #0          ; デバイスをスリープモードにする
SET.B   OVDCONH     ; デバイス立上がり時に POVD ビットをセット
```

フォールト A およびフォールト B 入力ピンが有効化され、FLTCON レジスタを通して PWM ピンを制御できる場合、デバイスがスリープモードに入っても通常通り機能を維持します。デバイスがスリープモードの間にフォールトピンのうち 1 つが Low になった場合、PWM 出力は FLTCON レジスタでプログラムされたフォールト状態になります。

フォールト出力ピンにより、CPU をスリープモードからウェイクアップさせることができます。フォールト割込み有効化ビットがセットされた場合 (FLTIE = 1)、フォールトピンが Low になったときデバイスはスリープモードからウェイクアップします。フォールトピン割込み優先順位が現在の CPU の優先順位より大きい場合は、ウェイクアップ時にプログラム実行がフォールトピン割込みベクターの場所から開始されます。その他の場合は、PWRSAV 命令の次の命令からプログラムが実行されます。

### 15.13.2 IDLE モードにおける PWM の動作

デバイスが IDLE モードに入ったとき、システムクロックソースは機能を維持し、CPU は命令実行を停止します。PWM モジュールは、IDLE モードでは任意に動作を続けることが可能です。PTSIDL ビット (PTCON<13>) により、IDLE モードで PWM モジュールが停止するか通常通り動作を続けるか選択されます。

PTSIDL = 0 となった場合、デバイスが IDLE モードとなってもモジュールは通常通り動作を続けます。有効化された場合、PWM タイムベースの割込みが、デバイスを IDLE モードからウェイクアップさせるために用いられます。PWM タイムベース割込み有効化ビットがセットされた場合 (PTIE = 1)、PWM タイムベース割込み発生時にデバイスは IDLE モードからウェイクアップします。PWM タイムベース割込みの優先順位が現在の CPU の優先順位より高い場合、ウェイクアップ後に PWM 割込みベクターのメモリアドレスでプログラムの実行が開始されます。その他の場合、PWRSAV 命令の次の命令から実行が続けられます。

PTSIDL = 1 の場合、モジュールは IDLE モードで停止します。PWM モジュールが IDLE モードで停止するようにプログラムされている場合は、PWM 出力とフォールト入力ピンの動作は SLEEP モードにおける動作と同様です (セクション 15.13.1 「スリープモードにおける PWM の動作」の説明を参照してください)。

### 15.14 デバイスエミュレーション用特別機能

PWM モジュールは、デバッグ環境をサポートする特別な機能を備えています。有効化された全ての PWM ピンは、メモリ内容チェックのためにハードウエアエミュレータやデバッガデバイスが停止した場合、トライステートすることができます。ユーザーはデバイス実行が停止した場合に、PWM 出力が正しい状態で行えるように、プルアップレジスタまたはプルダウンレジスタを備え付ける必要があります。

デバイスリセット時の PWM 出力ピンの機能および出力ピンの極性は、3つのデバイスコンフィギュレーションビットにより決定されます (詳しくはセクション 15.9 「PWM 出力極性制御」を参照してください)。ハードウエアデバッガあるいはエミュレーションツールは、これらのコンフィギュレーションビット値を変更する手段となります。詳しくはツールのユーザーマニュアルを参照してください。

表 15-7: 8 出力 PWM モジュールと関連したレジスタ

名称	ADR	ピット 15	ピット 14	ピット 13	ピット 12	ピット 11	ピット 10	ピット 9	ピット 8	ピット 7	ピット 6	ピット 5	ピット 4	ピット 3	ピット 2	ピット 1	ピット 0	リセット時の値		
INTCON1	0080	NSTDIS	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000			
INTCON2	0082	ALTIVT	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000			
IFS2	0088	—	—	—	FLTBIF	FLTAIF	—	—	—	PWMIF	—	—	—	—	—	—	0000 0000 0000 0000			
IEC2	0090	—	—	—	FLTBIE	FLTAIE	—	—	—	PWMIE	—	—	—	—	—	—	0000 0000 0000 0000			
IPC9	00A6	—	PWMIP<2:0>			—	—	—	—	—	—	—	—	—	—	—	0100 0100 0100 0100			
IPC10	00A8	—	FLTAIP<2:0>			—	—	—	—	—	—	—	—	—	—	—	0100 0100 0100 0100			
IPC11	00AA	—	—	—	—	—	—	—	—	—	—	—	—	—	FLTBIP<2:0>		0000 0000 0000 0000			
PTCON	01C0	PTEN	—	PTSIDL	—	—	—	—	—	PTOPS<3:0>			PTCKPS<1:0>	PTMOD<1:0>	—	—	0000 0000 0000 0000			
PTMR	01C2	PTDIR	PWM タイムベースレジスタ												—	—	0000 0000 0000 0000			
PTPER	01C4	—	PWM タイムベース周期レジスタ												—	—	0111 1111 1111 1111			
SEVTCMP	01C6	SEVTDIR	PWM 特殊イベント比較レジスタ												—	—	0000 0000 0000 0000			
PWMCON1	01C8	—	—	—	—	PMOD4	PMOD3	PMOD2	PMOD1	PEN4H	PEN3H	PEN2H	PEN1H	PEN4L	PEN3L	PEN2L	PEN1L	0000 0000 0000 0000		
PWMCON2	01CA	—	—	—	—	SEVOPS<3:0>			—	—	—	—	—	—	OSYNC	UDIS	0000 0000 0000 0000			
DTCN1	01CC	DTBPS<1:0>			デッドタイム B 値レジスタ			DTAPS<1:0>			デッドタイム A 値レジスタ			—	—	—	0000 0000 0000 0000			
DTCN2	01CE	—	—	—	—	—	—	—	—	DTS4A	DTS4I	DTS3A	DTS3I	DTS2A	DTS2I	DTS1A	DTS1I	0000 0000 0000 0000		
FLTACON	01D0	FAOV4H	FAOV4L	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L	FLTAM	—	—	—	FAEN4	FAEN3	FAEN2	FAEN1	0000 00-0 0000 0000		
FLTBCON	01D2	FBOV4H	FBOV4L	FBOV3H	FBOV3L	FBOV2H	FBOV2L	FBOV1H	FBOV1L	FLTBM	—	—	—	FBEN4	FBEN3	FBEN2	FBEN1	0000 0000 0000 0000		
OVDCON	01D4	POVD4H	POVD4L	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L	POUT4H	POUT4L	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L	1111 1111 00-0 0000		
PDC1	01D6	PWM デューティ #1 レジスタ												—	—	—	0000 0000 0000 0000			
PDC2	01D8	PWM デューティ #2 レジスタ												—	—	—	0000 0000 0000 0000			
PDC3	01DA	PWM デューティ #3 レジスタ												—	—	—	0000 0000 0000 0000			
PDC4	01DC	PWM デューティ #4 レジスタ												—	—	—	0000 0000 0000 0000			

注 1: PENnx 制御ビットのリセット状態は、PWMPIN デバイスコンフィギュレーションビットの状態により決定されます。

2: 6 出力 MCPWM モジュールの場合、陰影をつけたレジスタとビットの記憶位置は実装されていません。

表 15-8: 6出力 PWM モジュールと関連するレジスタ

名称	ADR	ビット 15	ビット 14	ビット 13	ビット 12	ビット 11	ビット 10	ビット 9	ビット 8	ビット 7	ビット 6	ビット 5	ビット 4	ビット 3	ビット 2	ビット 1	ビット 0	リセット時の値						
INTCON1	0080	NSTDIS	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000							
INTCON2	0082	ALTIVT	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000							
IFS2	0088	—	—	—	—	FLATIF	—	—	—	PWMIF	—	—	—	—	—	—	0000 0000 0000 0000							
IEC2	0090	—	—	—	—	FLTAIE	—	—	—	PWMIE	—	—	—	—	—	—	0000 0000 0000 0000							
IPC9	00A6	—	PWMIP<2:0>			—	—	—	—	—	—	—	—	—	—	—	0100 0100 0100 0100							
IPC10	00A8	—	FLTAIP<2:0>			—	—	—	—	—	—	—	—	—	—	—	0100 0100 0100 0100							
PTCON	01C0	PTEN	—	PTSIDL	—	—	—	—	—	PTOPS<3:0>			PTCKPS<1:0>		PTMOD<1:0>		0000 0000 0000 0000							
PTMR	01C2	PTDIR	PWM タイムベースレジスタ														0000 0000 0000 0000							
PTPER	01C4	—	PWM タイムベース周期レジスタ														0111 1111 1111 1111							
SEVTCMP	01C6	SEVTDIR	PWM 特殊イベント比較レジスタ														0000 0000 0000 0000							
PWMCON1	01C8	—	—	—	—	PMOD3	PMOD2	PMOD1	—	PEN3H	PEN2H	PEN1H	—	PEN3L	PEN2L	PEN1L	0000 0000 0000 0000							
PWMCON2	01CA	—	—	—	—	—	—	—	—	—	—	—	—	OSYNC	UDIS	0000 0000 0000 0000								
DTCON1	01CC	—	—	—	—	—	—	—	DTAPS<1:0>	デッドタイム A 値レジスタ							0000 0000 0000 0000							
予約	01CE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—							
FLTACON	01D0	—	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L	FLTAM	—	—	—	FAEN4	FAEN3	FAEN2	FAEN1	0000 00-0 0000 0000							
予約	01D2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—							
OVDCON	01D4	—	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L	—	—	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L	1111 1111 00-0 0000							
PDC1	01D6	PWM デューティ #1 レジスタ															0000 0000 0000 0000							
PDC2	01D8	PWM デューティ #2 レジスタ															0000 0000 0000 0000							
PDC3	01DA	PWM デューティ #3 レジスタ															0000 0000 0000 0000							

注 1: PENxx 制御ビットのリセット状態は、PWMPIN デバイスコンフィギュレーションビットの状態により決定されます。

2: 6出力 MCPWM モジュールの場合、陰影をつけたレジスタとビットの記憶位置は実装されていません。

## 15.15 関連するアプリケーションノート

このセクションではこの章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正の必要な場合や制限がある場合もありますが、使用可能です。現在 MCPWM モジュールと関連する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 #
PIC18CXXX/PIC16CXXX サーボモーター	AN696

**注:** dsPIC30F ファミリーのデバイスに関するその他のアプリケーションノートやコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 15.16 改訂履歴

### 改訂 A

これは本ドキュメントの初版です。

### 改訂 B

dsPIC30F の MCPWM モジュールに関してより多くの情報が含まれています。

注意：



**MICROCHIP**

## 第 16 章. 直交エンコーダインターフェース (QEI)

### ハイライト

この章には以下の項目が含まれています。

16.1 序章 .....	16-2
16.2 制御およびステータスレジスタ .....	16-4
16.3 プログラム可能デジタルノイズフィルタ .....	16-8
16.4 直交デコーダ .....	16-9
16.5 16 ビットアップ/ダウンポジションカウンタ .....	16-11
16.6 16 ビットのタイマー/カウンタの代替として QEI を使用 .....	16-16
16.7 直交エンコーダインターフェース割り込み .....	16-17
16.8 入出力ピン制御 .....	16-18
16.9 省電力モード時の QEI の動作 .....	16-19
16.10 リセットの効果 .....	16-19
16.11 設計の秘訣 .....	16-21
16.12 関連するアプリケーションノート .....	16-22
16.13 改訂履歴 .....	16-23

## 16.1 序章

### 16.1.1 機能概要

直交エンコーダ（インクリメンタルエンコーダやオプティカルエンコーダとしても知られています）は、回転機構において位置や速度の検出に用いられています。直交エンコーダにより、スイッチドリラクタンスマーター（SR）や交流誘導電動機（ACIM）など多くのモーター制御アプリケーションのクローズループ制御が有効となります。

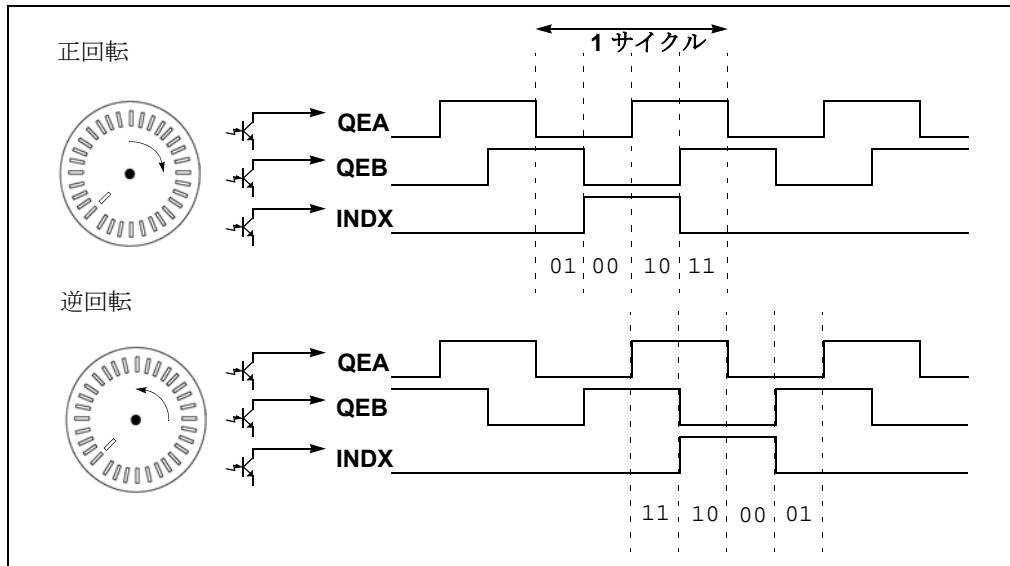
典型的なインクリメンタルエンコーダには、モーターのシャフトに装着されたスリット穴の空いた歯車と、歯車内のスリットを感知するフォトカプラモジュールが含まれています。通常、位相 A、位相 B およびインデックスの 3 つの出力により、距離や方向などモーターのシャフトの動作に関する情報が output されます。

位相 A (QEA) および位相 B (QEB) の 2 つのチャネルの間には独特の関係があります。もし位相 A が位相 B に先行する場合、(モーター) の方向は正回転とされます。位相 A が位相 B よりも遅れる場合、(モーター) の方向は逆回転とされます。インデックスパルスと呼ばれる別のチャネルは、1 回転ごとに作動し絶対的な位置を知るための参考として用いられます。これら 3 つの信号を比較するタイミング図については、図 16-1 を参照してください。

エンコーダにより発せられる直交信号には、4 つの特有の状態があります。図 16-1 では 1 カウントサイクル別にこれらの状態が示されています。回転方向が反対になると、信号の状態の順番が逆になることに注意してください。

直交デコーダにより位相信号およびインデックスパルスがキャプチャされ、情報がポジションパルスのカウント値に変換されます。一般に、シャフトがある方向に回転する際にカウント値がカウントアップする場合、シャフトの回転方向が逆になるとカウント値はカウントダウンします。

図 16-1: 直交エンコーダインターフェース信号

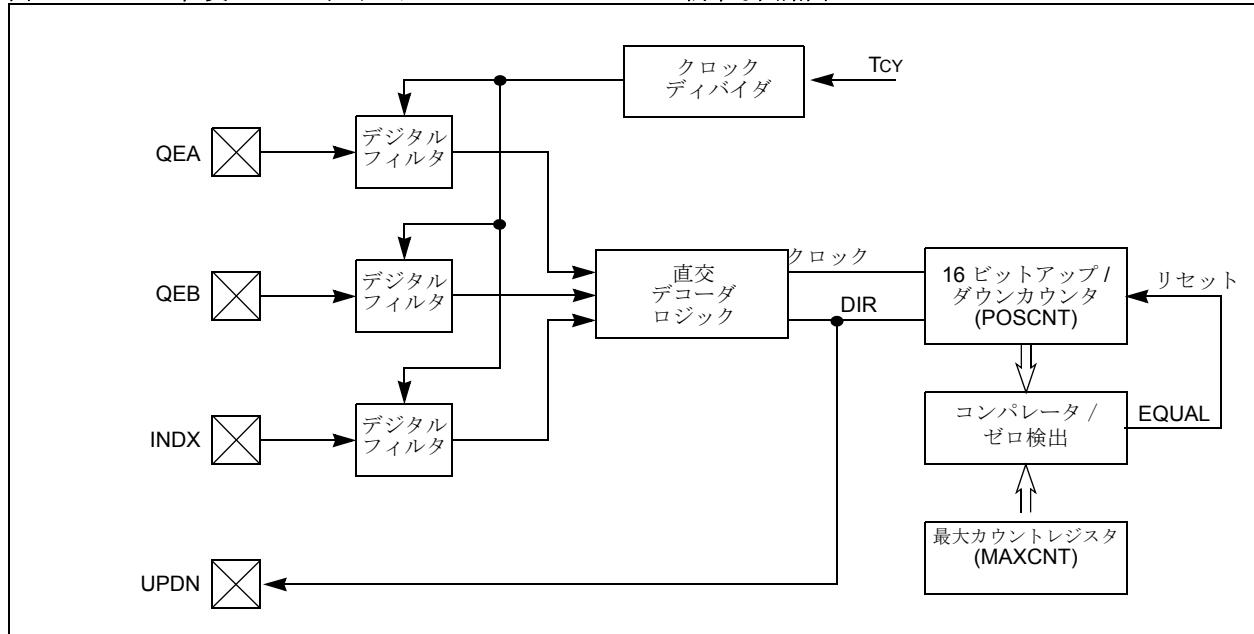


直交エンコーダインターフェース (QEI) によりインクリメンタルエンコーダ用のインターフェースが提供されます。QEI は位相 A および位相 B の信号を解釈する直交デコーダロジックと、カウント数を数え上げるアップダウンカウンタから成り立っています。入力時のデジタルグリッチフィルタが入力信号を整形します。図 16-2 は簡単な QEI モジュールのブロックを示しています。

QEI モジュールには以下のものが含まれます。

- 2 つの位相信号およびインデックスパルス用の入力
- プログラム可能な入力用デジタルノイズフィルタ
- カウンタパルスおよびカウント方向を決定する直交デコーダ
- 16 ビットのアップ / ダウンポジションカウンタ
- カウント方向ステータス
- X2 および X4 のカウント解像度
- 2 モードのポジションカウンタリセット
- 汎用 16 ビットタイマー / カウンタモード
- QEI またはカウントイベントにより生じる割り込み

図 16-2: 直交エンコーダインターフェースモジュールの簡単な回路図



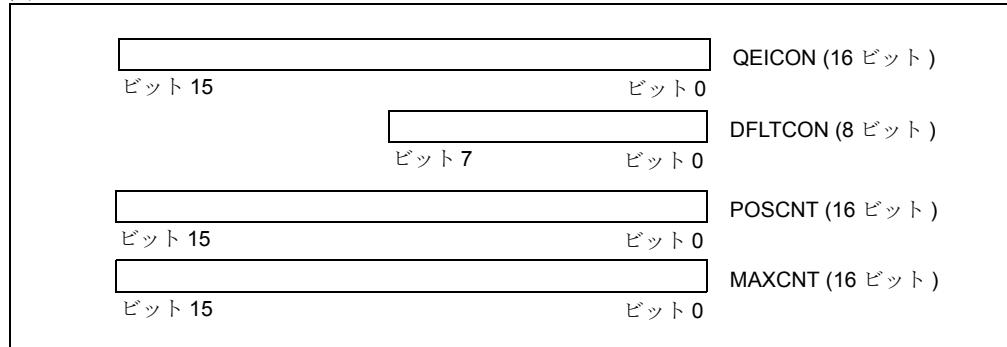
## 16.2 制御およびステータスレジスタ

QEI モジュールには、ユーザーがアクセス可能なレジスタが 4 つあります。レジスタはバイトモードもワードモードもいずれの場合もアクセス可能です。レジスタは図 16-3 に示され、以下がその内容です。

- 制御 / ステータスレジスタ (**QEICON**) - このレジスタにより、QEI の動作制御とステータスフラグによるモジュールの状態表示を行います。
- デジタルフラグ制御レジスタ (**DFLTCON**) - このレジスタにより、デジタル入力フィルタの動作の制御が可能となります。
- ポジションカウントレジスタ (**POSCNT**) - 16 ビットポジションカウンタの読み込みおよび書き込みが可能です。
- 最大カウントレジスタ (**MAXCNT**) - MAXCNT レジスタの数値は、一部の動作において POSCNT カウンタと比較されます。

**注:** POSCNT レジスタは、バイトアクセスができますが、レジスタをバイトモードで読み出すと続く読み込みで数値が一部だけ更新されるという結果が生じることもあります。ワードモード読み込み / 書き込みを用いるか、バイト動作中にカウンタがカウントを続けていないことを確認してください。

図 16-3: QEI プログラマモデル



レジスタ 16-1 および レジスタ 16-2 により、QEI モジュールの制御とデジタルフィルタ制御レジスタである QEICON および DFLTCON が定義されます。

## レジスタ 16-1: QEICON: QEI 制御レジスタ

高位バイト:							
R/W-0	U-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
CNTERR	—	QEISIDL	INDEX	UPDN	QEIM<2:0>		
ビット 15	ビット 8						

低位バイト:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SWPAB	PCDOUT	TQGATE	TQCKPS<1:0>		POSRES	TQCS	UDSRC
ビット 7	ビット 0						

- ビット 15 **CNTERR:** カウントエラーステータスフラグビット  
 1 = ポジションカウントエラーが発生  
 0 = ポジションカウントエラーの発生なし  
 (CNTERR フラグは、QEIM<2:0> が ‘110’ または ‘100’ の場合のみ適用)
- ビット 14 未実装: ‘0’ として読み込み
- ビット 13 **QEISIDL:** IDLE モード停止ビット  
 1 = デバイスが IDLE 時にモジュールの動作停止  
 0 = IDLE モードでもモジュールの動作継続
- ビット 12 **INDEX:** インデックスピン状態のステータスピット (読み込み専用)  
 1 = インデックスピンが High  
 0 = インデックスピンが Low
- ビット 11 **UPDN:** ポジションカウンタ方向ステータスピット  
 1 = ポジションカウンタの方向が正 (+)  
 0 = ポジションカウンタの方向が負 (-)  
 (QEIM<2:0> = ‘1XX’ の場合読み取り専用)  
 (QEIM<2:0> = ‘001’ の場合、読み込み / 書き込みビット)
- ビット 10-8 **QEIM<2:0>:** 直交エンコーダインターフェースモード選択ビット  
 111 = 直交エンコーダインターフェース有効化 (x4 モード)、MAXCNT と一致によりリセット  
 110 = 直交エンコーダインターフェース有効化 (x4 モード)、インデックスパルスによりポジションカウンタリセット  
 101 = 直交エンコーダインターフェース有効化 (x2 モード) MAXCNT と一致によりリセット  
 100 = 直交エンコーダインターフェース有効化 (x2 モード)、インデックスパルスによりポジションカウンタリセット  
 011 = 未使用 (モジュール無効化)  
 010 = 未使用 (モジュール無効化)  
 001 = 16 ビットタイマースタート  
 000 = 直交エンコーダインターフェース / タイマーオフ
- ビット 7 **SWPAB:** 位相 A および位相 B 入力入れ替え選択ビット  
 1 = 位相 A および位相 B が入れ替える  
 0 = 位相 A および位相 B は入れ替えない
- ビット 6 **PCDOUT:** ポジションカウンタ方向状態出力有効化ビット  
 1 = ポジションカウンタ方向状態出力有効化 (QEI ロジックにより入出力ピンの状態制御)  
 0 = ポジションカウンタ方向状態出力無効化 (通常の入出力ピンの動作)
- ビット 5 **TQGATE:** タイマーゲート制御タイマ有効化ビット  
 1 = タイマーゲート制御タイマ有効化  
 0 = タイマーゲートタイマ無効化
- ビット 4-3 **TQCKPS<1:0>:** タイマー入力クロックプリスケール選択ビット  
 11 = 1:256 プリスケーラ値  
 10 = 1:64 プリスケーラ値  
 01 = 1:8 プリスケーラ値  
 00 = 1:1 プリスケーラ値  
 (16 ビットタイマーモード時のみプリスケーラが使用される)

## レジスタ 16-1: QEICON: QEI 制御レジスタ (続き)

ビット 2 **POSRES:** ポジションカウンタリセット有効化ビット

1 = インデックスパルスによりポジションカウンタがリセット  
0 = インデックスパルスによりポジションカウンタがリセットされない  
(QEIM<2:0> = が 100 または 110 の場合にのみ有効)

ビット 1 **TQCS:** タイムクロックソース選択ビット

1 = ピン QEA による外部クロック (立ち上がりエッジ)  
0 = 内部クロック (TCY)

ビット 0 **UDSRC:** ポジションカウンタ方向選択制御ビット

1 = QEB ピンの状態によりポジションカウンタの方向決定  
0 = 制御 / ステータスピットである UPDN (QEICON<11>) によりタイマカウンタ (POSCNT) の方向が決定  
注: QEI モードに設定されている場合は、本ビットは無効

凡例:

R = 読み込み可能ビット W = 書き込み可能ビット U = 未実装、'0' が読み込まれます

-n = POR での値 '1' = ビットがセットされます '0' = ビットはクリアされます x = ビットは不定です

## レジスタ 16-2: DFLTCON: デジタルフィルタ制御レジスタ

高位バイト:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	CEID
ビット 15							ビット 8

低位バイト:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
QEOUT	QECK<2:0>				INDOUT	INDCK<2:0>	
ビット 7							ビット 0

ビット 15-9 未実装: ‘0’ として読み込み

ビット 8 **CEID:** カウントエラー割り込み無効化ビット  
1 = ポジションカウントエラーによる割り込み無効化  
0 = ポジションカウントエラーによる割り込み有効化

ビット 7 **QEOUT:** QEA/QEB デジタルフィルタ有効化ビット  
1 = デジタルフィルタ有効化  
0 = デジタルフィルタ無効化 (ピン通常動作)

ビット 6-4 **QECK<2:0>:** QEA/QEB デジタルフィルタクロック分割選択ビット  
111 = クロック分割 1:256  
110 = クロック分割 1:128  
101 = クロック分割 1:64  
100 = クロック分割 1:32  
011 = クロック分割 1:16  
010 = クロック分割 1:4  
001 = クロック分割 1:2  
000 = クロック分割 1:1

ビット 3 **INDOUT:** インデックスチャネルデジタルフィルタ有効化ビット  
1 = デジタルフィルタ有効化  
0 = デジタルフィルタ無効化 (ピン通常動作)

ビット 2-0 **INDCK<2:0>:** インデックスチャネルデジタルフィルタクロック分割選択ビット  
111 = クロック分割 1:256  
110 = クロック分割 1:128  
101 = クロック分割 1:64  
100 = クロック分割 1:32  
011 = クロック分割 1:16  
010 = クロック分割 1:4  
001 = クロック分割 1:2  
000 = クロック分割 1:1

## 凡例:

R = 読み込み可能ビット

W = 書き込み可能  
ビット

U = 未実装、‘0’ が読み込まれます

-n = POR での値

'1' = ビットがセット  
されます '0' = ビットはクリア  
されます x = ビットは不定です  
されます

### 16.3 プログラム可能デジタルノイズフィルタ

デジタルノイズフィルタの部分は、インデックス信号および直交信号にのって来るノイズをはねのける役割を果たしています。シュミットトリガーによる入力、そして3クロックサイクル遅延フィルタが合わさり、低レベルのノイズや、モーターシステムアプリケーションなどのノイズを発生しやすいアプリケーションでよく発生する、大きくて短いスパイクノイズを防ぐことができます。

フィルタにより、連続3フィルタサイクルの間安定した値が入力されない限り、フィルタ出力信号が変化しないようにすることができます。

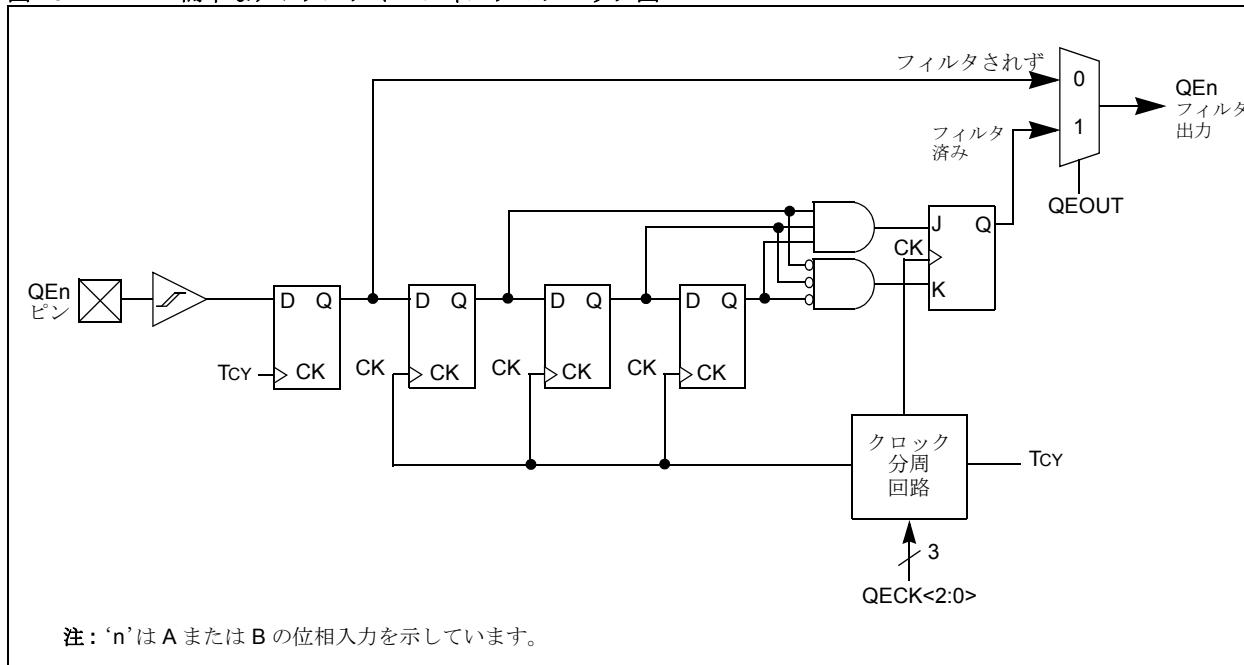
フィルタクロックの速度により、フィルタのローパス帯域が決定されます。フィルタクロックが遅い場合、フィルタクロックが高速の場合と比較してより低い周波数のノイズを防ぐことができます。フィルタクロックは、プログラム可能な分周器により分周されたデバイス  $f_{CY}$  クロックです。

**QEOUT** ビット (DFLTCON<7>) を設定することにより、QEA チャネルおよび QEB チャネルのフィルタが有効化されます。QECK<2:0> ビット (DFLTCON<6:4>) により、QEA および QEB チャンネルに用いられるフィルタクロックの分周比が特定されます。

**INDOUT** ビット (DFLTCON<3>) を設定することにより、インデックスチャネルのフィルタが有効化されます。INDCK<2:0> ビット (DFLTCON<2:0>) により、インデックスチャネル用のフィルタクロック分周比が特定されます。リセット時には、全チャネル用のフィルタが無効化されます。

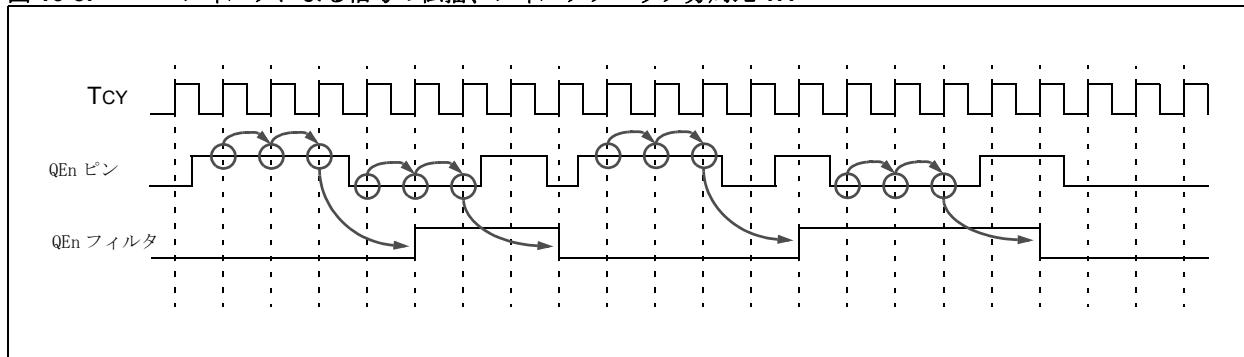
図 16-4 は、デジタルノイズフィルタの簡単なブロック図を示したものです。

図 16-4: 簡単なデジタルノイズフィルタのブロック図



注: 'n' は A または B の位相入力を示しています。

図 16-5: フィルタによる信号の伝播、フィルタクロック分周比 1:1



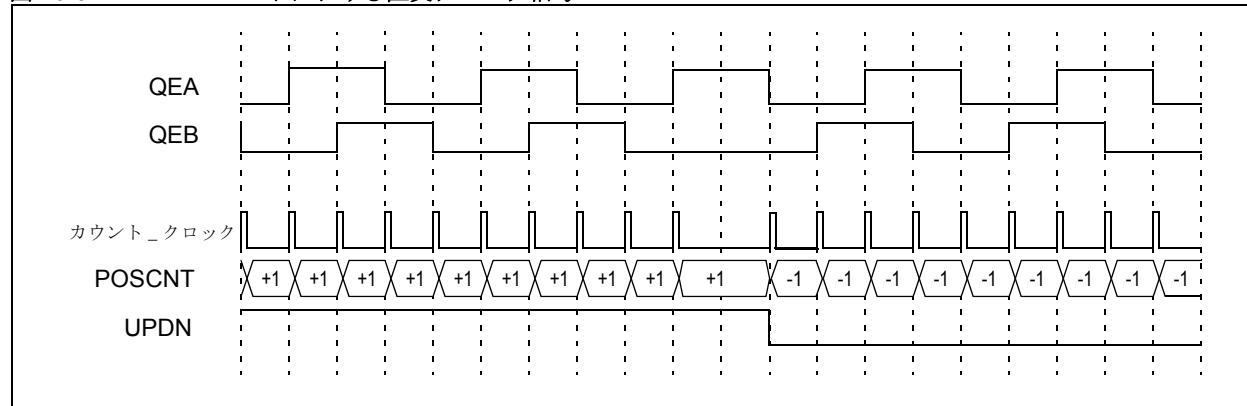
#### 16.4 直交デコーダ

QEIM2 = 1 (QEICON<10>) にセットされた時、位置測定モードが選択されます。

QEIM1 = 1 (QEICON<9>) の場合、「x4」測定モードが選択され、QEL ロジックにより位相 A および位相 B 入力信号の両エッジでポジションカウンタがカウントします。

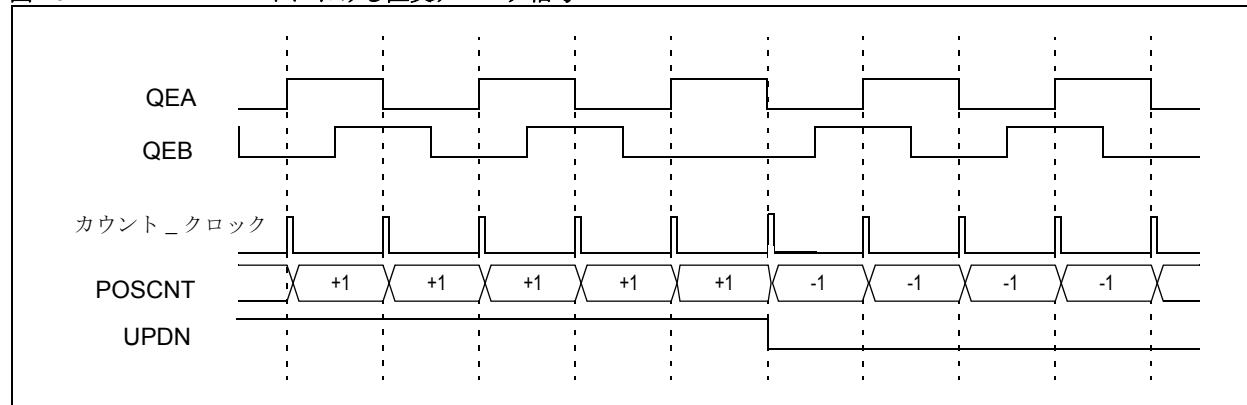
「x4」測定モードにより、より細かい解像度データ（より多くのポジションカウント）により、エンコーダの位置が決定されます。

図 16-6: 4X モードにおける直交デコーダ信号



QEIM1 = 0 の場合、「x2」測定モードが選択され、ポジションカウンタインクリメント用クロックとして QEI ロジックにより位相 A の立ち上がりエッジおよび立ち下がりエッジのみが使われます。位相 A の全ての立ち上がりエッジおよび立ち下がりエッジにより、ポジションカウンタがカウントアップ / ダウンします。x4 測定モードと全く同じく、位相 B 信号がカウンタの方向の決定のために依然として用いられます。

図 16-7: 2X モードにおける直交デコーダ信号



#### 16.4.1 先行 / 遅延テストに関する説明

先行 / 遅延テストは直交デコーダロジックにより行われ、QEA および QEB 信号の位相関係により、POSCNT レジスタのカウントアップ / ダウンが決められます。表 16-1 は先行 / 遅延テストについて明確にしたものです。

表 16-1: 先行 / 遅延テストに関する記述

現在のトランジション	前のトランジション	条件	動作	
QEA↑	QEB↓	QEA が QEB チャネルより遅延	UPDN セット	POSCNT 増分
	QEB↑	QEA が QEB チャネルより遅延	UPDN クリア	POSCNT 減分
	QEA↓	方向転換	UPDN トグル切替	POSCNT 増分または減分
QEA↓	QEB↓	QEA が QEB チャネルより遅延	UPDN クリア	POSCNT 減分
	QEB↑	QEA が QEB チャネルより先行	UPDN セット	POSCNT 増分
	QEA↑	方向転換	UPDN トグル切替	POSCNT 増分または減分
QEB↑	QEA↓	QEA が QEB チャネルより遅延	UPDN クリア	POSCNT 減分
	QEA↑	QEA が QEB チャネルより先行	UPDN セット	POSCNT 増分
	QEB↓	方向転換	UPDN トグル切替	POSCNT 増分または減分
QEB↓	QEA↓	QEA が QEB チャネルより先行	UPDN セット	POSCNT 増分
	QEA↑	QEA が QEB チャネルより遅延	UPDN クリア	POSCNT 減分
	QEB↑	方向転換	UPDN トグル切替	POSCNT 増分または減分

#### 16.4.2 カウント方向ステータス

前項目で説明した通り、QEI ロジックにより位相 A と位相 B の時間の関係に基づき UPDN 信号が生成されます。UPDN 信号は出入力ピンに出力されることもあります。

PCDOUT ビット (QEICON<6>) をセットし、ピンと関連する適切な TRIS ビットをクリアすることにより、UPDN 信号が出力ピンを駆動します。

出力ピンの他に、この内部 UPDN 信号の状態が SFR ビットの QEICON<11> に読み込み専用ビットとして伝えられ、UPDN として提供されます。

#### 16.4.3 エンコーダのカウント方向

直交カウントの方向は SWPAB ビット (QEICON<7>) により決められます。SWPAB = 0 の場合、位相 A は直交カウンタの A 入力に接続され、位相 B 入力は直交カウンタの B 入力に接続されます。そのため、位相 A 信号が位相 B 信号より先行すれば、エッジ発生ごとに直交カウンタは増分されます。このこと（A 信号が B 信号より先行）は、動作の方向が正回転になることとして定義されます。SWPAB ビット (QEICON<7>) をロジック 1 にセットすることで、位相 A の入力が直交カウンタの B 入力に接続され、位相 B 信号が直交カウンタの A 入力に接続されます。そのため、位相 A 信号がデバイスピンで位相 B に先行している場合、直交カウンタへの位相 A の入力は位相 B の入力より遅延していることになり、反対方向への回転として認識され、カウンタは直交パルスごとに減分します。

#### 16.4.4 直交率

位置制御システムのモーター回転数は様々です。モーター回転数と直交エンコーダのスリット数により、QEA および QEB 入力信号の周波数が決まります。直交エンコーダ信号は、カウントパルスが直交信号エッジ発生毎に生成されるようデコードされます。これにより、角位置測定の解像度がエンコーダスリット数の最大 4 倍になります。例えば、4096 スリットのエンコーダを用いるモーター回転数 6,000RPM のモーターの直交カウント速度は  $((6000/60)^*(4096*4)) = 1.6384 \text{ MHz}$  となります。同様に、8192 スリットのエンコーダを使用するモーター回転数 10,000RPM のモーターの直交カウント速度は  $((10000/60)^*(8192*4)) = 5.46 \text{ MHz}$  となります。

QEI は直交周波数が最大  $F_{CY}/3$  まで許容できます。例えば、 $F_{CY} = 30 \text{ MHz}$  の場合、QEA および QEB 信号の最大周波数は  $10\text{MHz}$  です。詳しくはデバイスデータシートの「電気的特性」の項目を参照してください。

#### 16.5 16 ビットアップ/ダウンポジションカウンタ

16 ビットアップ/ダウンカウンタは、直交デコーダロジックにより生成される全てのカウントパルス時に上方または下方にカウントします。そのとき、カウンタは積分回路として機能し、そのカウント値は位置と比例します。カウントの方向は直交デコーダにより決定されます。

ユーザーソフトウェアにより、POSCNT レジスタを読み取ることでカウンタの内容を調べることができます。ユーザーソフトウェアにより、POSCNT レジスタに書き込んでカウントを初期化することもできます。

QEIM ビットの変更はポジションカウンタレジスタの内容に影響を与えません。

##### 16.5.1 ポジションカウンタの使い方

システムは、複数の方法のうちいずれかを用いてポジションカウンタのデータを使用することができます。システムによっては、ポジションカウントは一貫して累算され、システムの位置全体を示す絶対値となります。典型的な例をとると、直交エンコーダがプリンタの印字ヘッドを制御するモーターに添えられていると考えてください。動作中、システムは印字ヘッドを最も左の位置に動かし、POSCNT レジスタをリセットすることで初期化されます。印字ヘッドが右に動くと、直交エンコーダは POSCNT レジスタでカウント数を累算するようになります。印字ヘッドが左に動くと、累算カウント数は減少します。印字ヘッドが最も右の位置に達すると、最大のポジションカウントに達するはずです。最大カウント数が  $2^{16}$  未満の場合 QEI モジュールにより動作の全範囲がエンコードされます。

しかし、最大カウント数が  $2^{16}$  以上の場合は、ユーザーソフトウェアによりさらに正確なカウント数を把握する必要があります。一般に、このためにはモジュールを、最大カウント数に達するとリセットされるモードにセットします。QEIMO = 1 により有効化されるモードでは、MAXCNT レジスタがポジションカウントリセットに用いられます。カウンタが増分中にあらかじめ決められた最大カウント数に達した場合、または減分中にゼロに達した場合は、カウントはリセットされ、割り込みが生成され、ユーザーソフトウェアにより、ポジションカウンタの上位ビットとなるソフトウェアカウンタを増分・減分させることができます。最大カウントは 0xFFFF であり、QEI カウンタやソフトウェアカウンタの全範囲を有効化せたり、エンコーダの 1 回転のカウント数などより小さくても重要性のある値を有効化させます。

他のシステムでは、ポジションカウントは周期的な場合もあります。ポジションカウントはインデックスパルスにより決められる回転数の範囲で、歯車の位置を参照するためにのみ用いられます。例えば、スクリューロッドにより動作するツールのプラットフォームでは、スクリューロッドに取り付けられた直交エンコーダが用いられます。この動作で例えば、スクリューが目指す位置に達するまでに 5.5 回転が必要なことがあります。ユーザーソフトウェアにより、完全に回転が生じる 5 回のインデックスパルスが検出され、残った半回転を測定するためにポジションカウントが用いられます。この方法では、インデックスパルスによりポジションカウンタがリセットされて回転ごとにカウンタが初期化され、回転ごとに割り込みが生成されます。QEIMO = 0 に設定されるとこのモードが有効化されます。

直交エンコーダインターフェース (QEI)

### 16.5.2 ポジションカウンタリセットに MAXCNT 使用

QEIMO ビットが ‘1’ である場合、ポジションカウンタはポジションカウントがあらかじめ定められた高い値および低い値と一致した場合にリセットされます。インデックスパルスのリセットのメカニズムは使用されません。

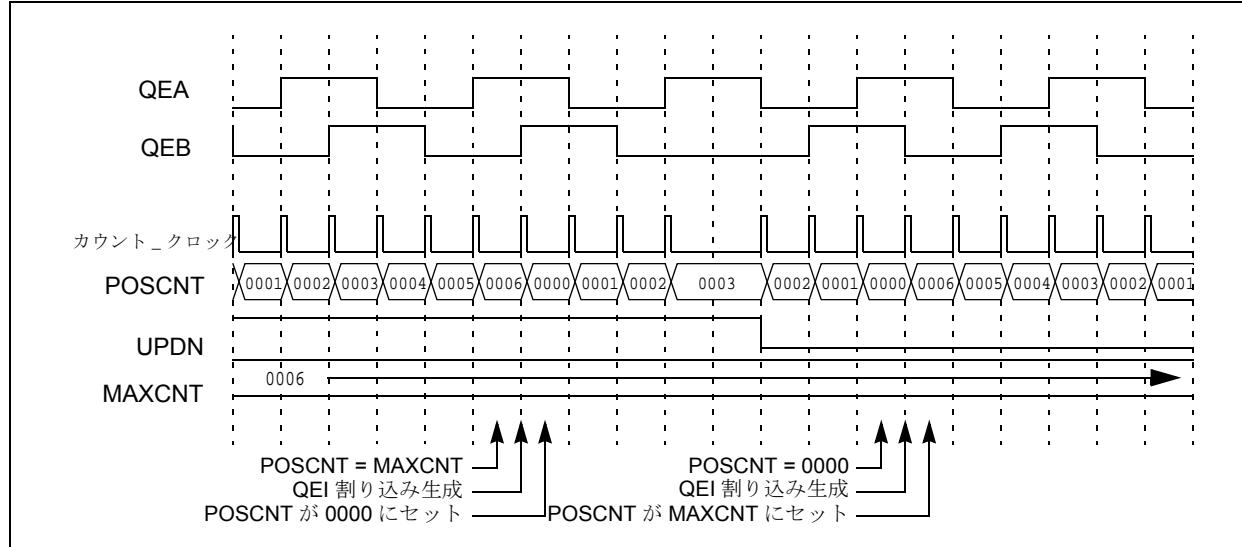
このモードでは、以下のようにポジションカウンタリセットのメカニズムが作動します。(関連するタイミングの詳細については図 16-8 を参照してください)。

- カウンタが前方に移動、すなわち QEA が QEB に先行し、POSCNT レジスタの値が MAXCNT レジスタの値と一致したときは、POSCNT は **POSCNT** を増分させる直交パルスエッジが次に発生したときにゼロにリセットされます。このロールオーバーイベント発生時に割り込みイベントが生成されます。
- エンコーダが後方に移動、すなわち QEB が QEA に先行し、POSCNT レジスタの値が ‘0’ にカウントダウンされるときは、POSCNT には **POSCNT** を減分させる直交パルスエッジが次に発生したときに MAXCNT レジスタの値が読み込まれます。このアンダーフローイベント発生時に割り込みイベントが生成されます。

MAXCNT を位置の境界として使用する場合、ポジションカウンタはエンコーダカウントの  $2X$  または  $4X$  のいずれかでカウントすることに留意してください。標準的な回転エンコーダの場合、MAXCNT に書き込む適切な値は、 $4x$  位置モードでは  $4N-1$  であり、 $2x$  位置モードの場合  $2N-1$  となります。ここで  $N$  とはエンコーダの一回転あたりのカウント数を示しています。

システムのレンジが  $2^{16}$  を超える絶対的な位置に関する情報に関しては、0xFFFF 値を MAXCNT レジスタにロードすることも適切です。モジュールによりポジションカウンタのロールオーバーまたはアンダーフロー時に割り込みが生成されます。

図 16-8: ロールオーバー / ロールアンダーリセット - アップ / ダウンポジションカウンタ

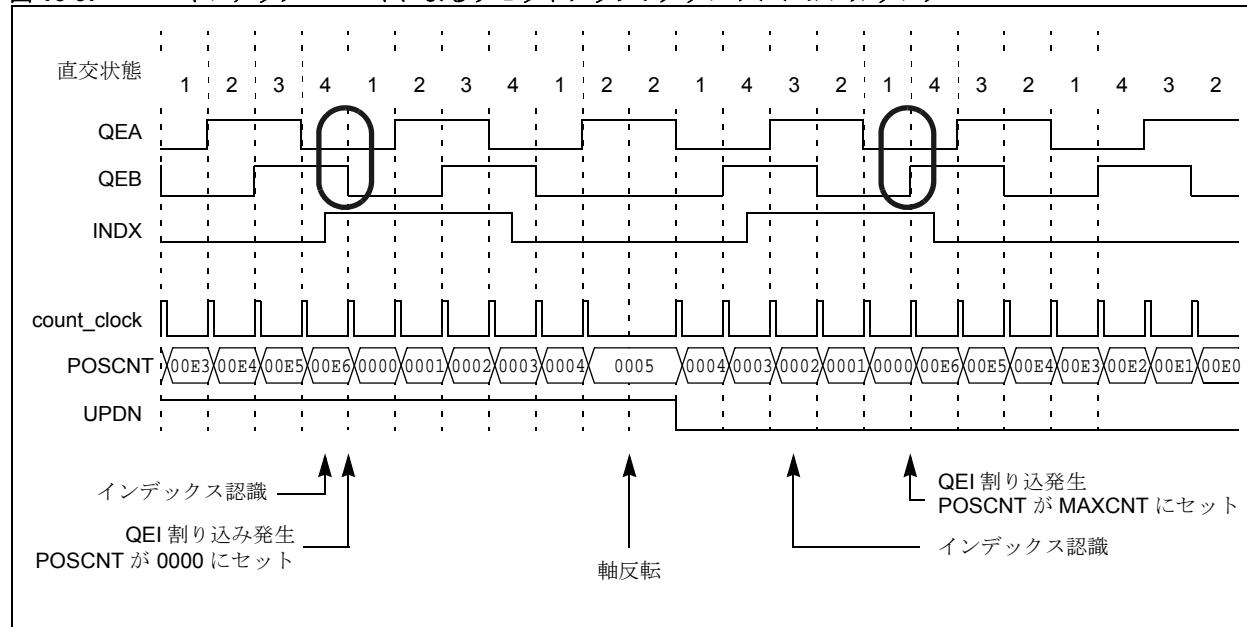


### 16.5.3 ポジションカウンタリセットにインデックス使用

QEIM<0> = 0 の場合、ポジションカウンタのリセットにインデックスパルスが用いられます。このモードでは、以下のようにしてポジションカウンタリセットのメカニズムが作動します。(関連するタイミングの詳細については図 16-9 を参照してください)。

- ・ポジションカウントは、インデックスピンでインデックスパルスを受信する度にリセットされます。
- ・エンコーダが正回転のとき、すなわち QEA が QEB に先行する場合、POSCNT は ‘0’ にリセットされます。
- ・エンコーダが逆回転のとき、すなわち QEB が QEA に先行する場合、MAXCNT レジスタの値は POSCNT に読み込まれます。

図 16-9: インデックスモードによるリセットアップ/ダウンポジションカウンタ



#### 16.5.3.1 インデックスパルス検出基準

様々なメーカーのインクリメンタルエンコーダでは、インデックスパルスに異なるタイミングが用いられています。インデックスパルスは 4 つの直交状態のいずれかに合わせられ、1 サイクル (4 直交状態)、半サイクル (2 直交状態) あるいは 4 分の 1 サイクル (1 直交状態) のパルス幅を持つことがあります。1 サイクルまたは半サイクルの幅のインデックスパルスは通常「ゲートなし」と呼ばれ、幅 4 分の 1 サイクルのインデックスパルスは通常「ゲートあり」と呼ばれています。

発生するインデックスパルスの種類に関係なく、QEI は歯車の方向が反対になってもカウントの対称性を維持します。このことは、歯車が前方あるいは後方に回転する際に、同じ直交状態のトランジションでインデックスパルスがポジションカウンタをリセットさせる必要があることを意味しています。

例えば、図 16-9 では、図に示されているように直交状態が 4 から 1 に変わる際に、最初のインデックスパルスが認識され、このパルスにより POSCNT がリセットされます。QEI によりこのトランジションの状態がラッチされます。後にインデックスパルスが検出されると、リセットするためにその状態トランジションが用いられます。

歯車が反転すると、インデックスパルスが再び発生しますが、ポジションカウンタのリセットは、同じく図に示されているように直交状態が 1 から 4 に変化するまで発生しません。

**注:** QEI インデックスロジックにより、回転方向に関係なく POSCNT レジスタが常にインデックスパルスに対して同じ位置で調整されます。モジュール初期化後最初のインデックスパルス発生時に位置が選択されます。

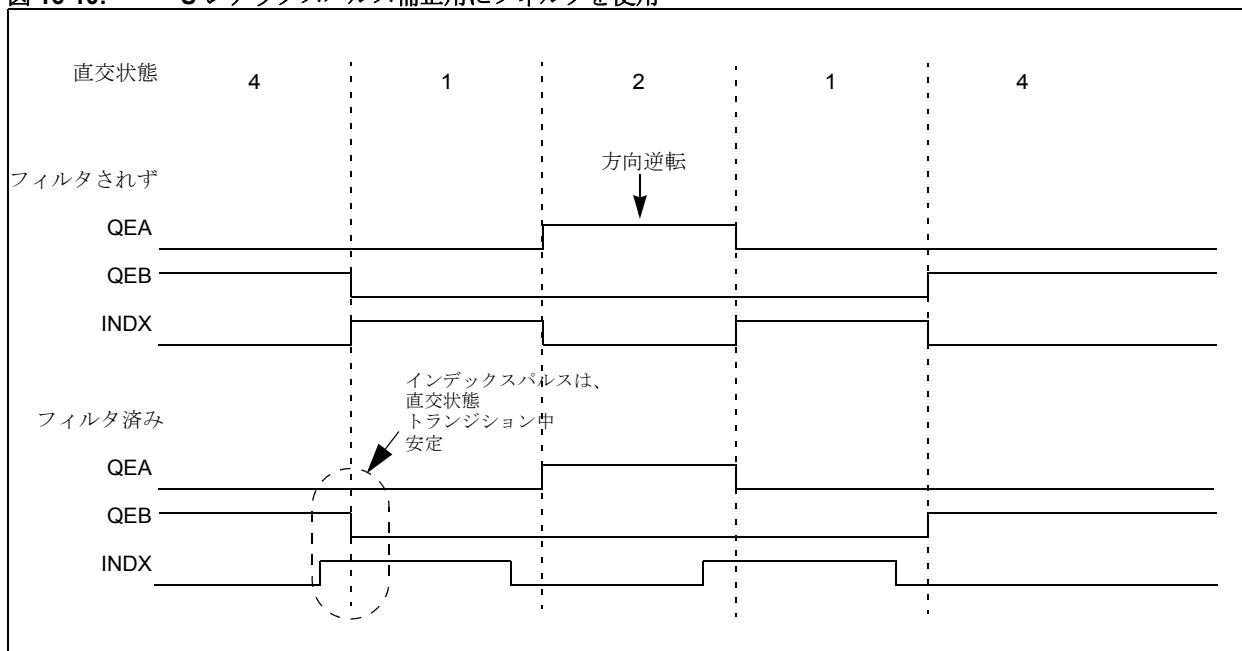
### 16.5.3.2 インデックスパルスのデスキー (補正)

インデックス信号エッジと QEA および QEB 信号との時間の関係は、インデックス信号の正しいサンプリングにあたって重要です。インデックスパルスのエッジタイミングは、多くの場合 QEA 信号および QEB 信号のエッジと関連しています。

直交信号によりカウンタにカウントパルスが伝えられるため、QEI は直交エッジと関連したインデックスパルスを確かに検出できる必要があります。ユーザーは、インデックスパルスが直交信号の状態トランジションの間安定しているかを確認する必要があります。そうしないとパルスが検出されない可能性があります。

ユーザーがフィルタクロックの分周比を変えることで、デジタルフィルタブロックでスキュー調整を解決することができます。例えば、インデックスパルス用フィルタクロックの周波数の 1/2 のフィルタクロックを直交信号用に用いることにより、インデックスパルスと関連する直交信号を遅らせることができ、図 16-10 で示されているようにインデックスパルスの検出を確かにを行うことができます。

図 16-10: U インデックスパルス補正用にフィルタを使用



### 16.5.3.3 インデックスパルスステータス

インデックスビット (QEICON<12>) により、インデックスピンにおけるロジック状態のステータスが決められます。このステータスピットは、原点復帰シーケンスが必要な位置制御システムにおいて大変有用です。このシステムは参照位置を求めようとなります。インデックスビットには、デジタルフィルタが有効化された場合フィルタによる処理後のインデックスピンのステータスが示されます。

#### 16.5.3.4 エラーチェックにインデックスピンと MAXCNT を使用

カウンタがインデックスパルスモードでリセットされて動作する場合、QEI により POSCNT レジスタの境界条件を検出することができます。これは、インクリメンタルエンコーダシステムにおいてシステムエラーを検出するために用いられる場合もあります。

例えば、歯車エンコーダが 100 スリットからなっていると考えてください。X4 測定モードで使用され、インデックスパルス時にリセットされる場合、カウンタがカウントできる範囲は 0 から 399 となり (0x018E)、その後にリセットされます。POSCNT レジスタが 0xFFFF または 0x0190 の値に達すると、何らかのシステムエラーが発生したことになります。

POSCNT レジスタの内容は、上方カウント中であれば MAXCNT + 1 と比較され、下方カウント中であれば 0xFFFF と比較されます。QEI がこれらの値を検出すると、CNTERR ビット (QEICON<15>) をセットし、任意で QEI 割り込みを生成させることによりポジションカウントエラー条件が発生します。

CEID 制御ビット (DFLTCOM<8>) がクリアされた場合（デフォルト）、ポジションカウントエラーが検出されると QEI 割り込みが生成されます。CEID 制御ビットがセットされると、割り込みは発生しません。

ポジションカウンタは、ポジションカウントのエラーを検出後エンコーダのエッジをカウントしつづけます。CNTERR がユーザーによりクリアされるまで、続けて発生するポジションカウントエラーイベントで割り込みは発生しません。

#### 16.5.3.5 ポジションカウンタリセット有効化

ポジションカウンタリセット有効化ビットである POSRES (QEICON<2>) により、インデックスパルスが検出された場合ポジションカウンタのリセットが有効化されます。このビットは QEI モジュールが QEIM<2:0> = ‘100’ または ‘110’ で定義されるモードに設定された場合のみ実行されます。

POSRES ビットがロジック ‘1’ にセットされた場合、この項目で説明されているようにインデックスパルスが検出された場合にポジションカウンタはリセットされます。

POSRES ビットがロジック ‘0’ にセットされた場合、インデックスパルスが検出されてもポジションカウンタはリセットされません。ポジションカウンタは上方または下方にカウントを続け、ロールオーバーまたはアンダーフローが発生した場合にリセットされます。QEI はインデックスパルスが検出されると割り込みを生成しつづけます。

直交エンコーダインターフェース (QEI)

## 16.6 16 ビットのタイマー / カウンタの代替として QEI を使用

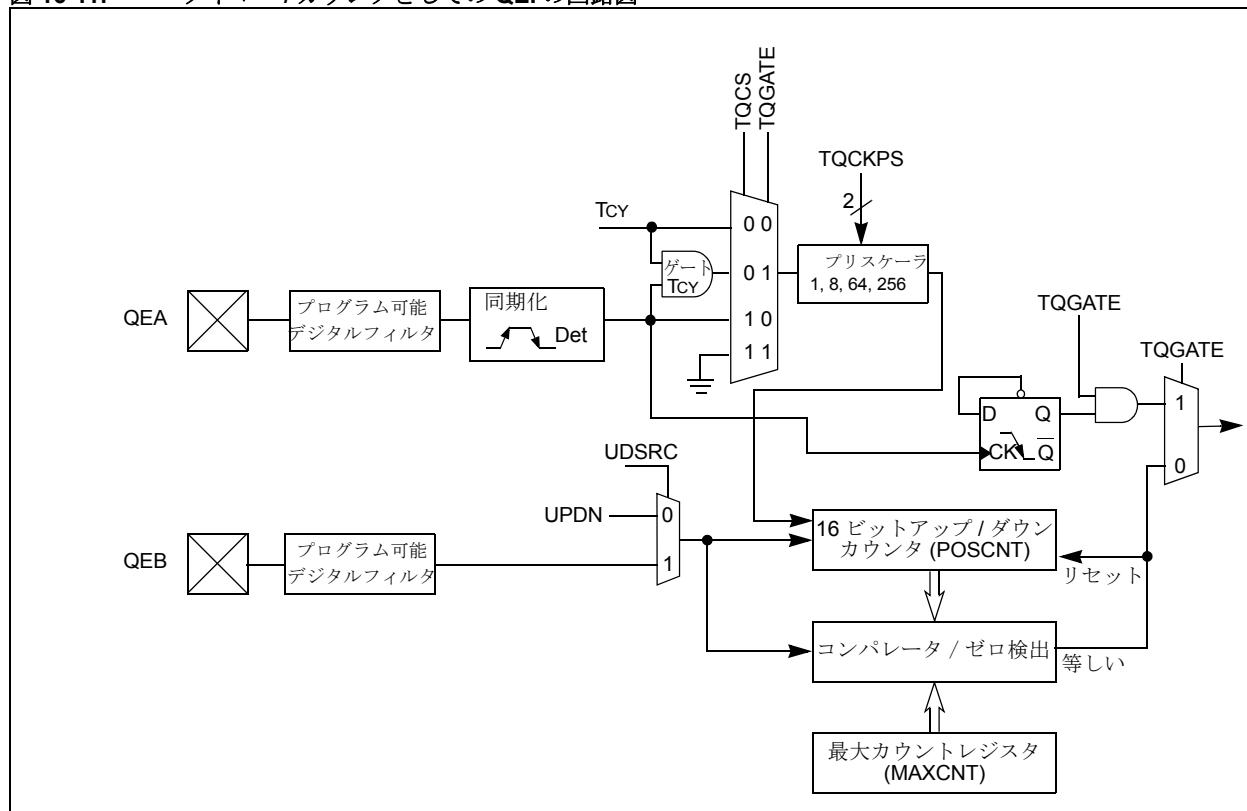
QEI モジュールが QEIM<2:0> = 001 に設定された場合、QEI 機能は無効化され、QEI モジュールは 16 ビットのタイマー / カウンタとして設定されます。補助タイマーのセットアップおよび制御は QEICON レジスタを通してなされます。

QEI タイマーの機能は他の dsPIC30F 内蔵タイマと類似しています。タイマーについての一般的な説明は第 12 章、「タイマー」を参照してください。

タイマーに設定された場合、POSCNT レジスタは汎用タイマーの TMRn レジスタと類似したタイマーレジスタとしての役割を果たします。MAXCNT レジスタは汎用タイマーの PRn レジスタと類似した周期レジスタとしての役割を果たします。タイマーレジスタと周期レジスタの一致が発生した場合、QEIF フラグがセットされます。

**注：** オペレーションモードを QEI からタイマーへ、あるいはタイマーから QEI に変更しても、タイマー / ポジションカウントレジスタの内容に影響は及ぼません。

図 16-11：タイマー / カウンタとしての QEI の回路図



### 16.6.1 アップ / ダウンタイマーの動作

QEI タイマーは増分も減分もできます。これは他の大部分のタイマーにはない独特の機能です。タイマーが上方カウントするように設定されている場合、タイマー (POSCNT) は、カウント値が周期レジスタ (MAXCNT) と一致するまで増分します。一致するとタイマーはゼロにリセットされ、再び増分を始めます。

タイマーが下方カウントするように設定されている場合、タイマー (POSCNT) はカウント値が周期レジスタ (MAXCNT) と一致するまで減分します。タイマーはゼロにリセットされ、再び減分を始めます。

タイマーが下方カウントするように設定されている場合、正しく動作させるためには幾つかの一般的な動作ガイドラインに従う必要があります。

1. MAXCNT レジスタは周期一致レジスタとして動作しますが、カウンタが減分中なので、目指す一致値は 2 の補数となります。例えば、0x1000 クロックをカウントするには、周期レジスタには 0xF000 を書き込む必要があります。

2. 一致する条件で、タイマーはゼロにリセットされます。

入出力ピンまたは SFR 制御ビットによりカウント方向の制御が決定されます。

制御ビット UDSRC (QEICON<0>) により、タイマーのカウント方向が決定されます。

UDSRC = 1 にセットされている場合、タイマーのカウント方向は QEB ピンにより制御されます。QEB ピンが ‘1’ の場合、カウント方向は増分です。QEB ピンが ‘0’ の場合、カウント方向は減分です。

UDSRC = 0 の場合、タイマーのカウント方向は UPDN ビット (QEICON<11>) により制御されます。UPDN = 1 の場合、タイマーは増分します。UPDN = 0 の場合、タイマーは減分します。

### 16.6.2 タイマー外部クロック

TQCS ビット (QEICON<1>) により、内部クロックと外部クロックのどちらが用いられるか選択されます。QEI タイマーは、TQCS がセットされているとき QEA ピンを外部クロック入力に使用することができます。QEI タイマーは外部非同期カウンタモードをサポートしています。外部のクロックソースを用いる場合、クロックは自動的に内部命令サイクル (TCY) と同期化されます。

### 16.6.3 タイマーゲートの作動

TQGATE ビット (QEICON<5>) がセットされ、TQCS がクリアされた場合、QEA ピンはタイマーゲートとして機能します。

TQCS および TQGATE が同時にセットされた場合、タイマーは増分せず、割り込みを生成しません。

## 16.7 直交エンコーダインターフェース割り込み

QEI のモードによって、以下のイベント発生時に QEI により割り込みが生成されます。

- QEIM<2:0> = ‘111’ および ‘101’ に設定され、マッチモードで動作する場合、ポジションカウンタがロールオーバーまたはアンダーフローとなった場合に割り込みが発生します。
- QEIM<2:0> = ‘110’ および ‘100’ に設定され、インデックスモードで動作する場合、インデックスパルス検出時、また CNTERR ビットがセットされた場合も任意に割り込みが発生します。
- QEIM<2:0> = ‘001’ に設定され、タイマーまたはカウンタとして動作している場合、周期一致イベントまたは TQGATE=1 の場合にタイマーゲート立ち下がりエッジイベント発生時に割り込みが発生します。

QEI 割り込みイベントが発生した場合、QEIFF ビット (IFS2<8>) がセットされ、有効化されている場合割り込みが生成されます。QEIFF ビットはソフトウェアでクリアする必要があります。

QEI 割り込みの有効化は、各有効化ビット QEIE (IEC2<8>) を通して行われます。

## 16.8 入出力ピン制御

QEI モジュールを有効化することにより、関連する入出力ピンが QEI の制御におかれ、ポートなど優先順位の低い入出力機能が入出力ピンに影響を与えることを防ぎます。

QEIM<2:0> およびその他の制御ビットにより決定されるモードにより、入出力ピンは、表 16-2 や表 16-3 で示されるように異なる機能を持つことがあります。

表 16-2: 直交エンコーダモジュールピン、入出力に関する記述

ピンの名称	ピンの種類	バッファの種類	詳細
QEA		ST	直交エンコーダ位相 A 入力、あるいは 補助的タイマー外部クロック入力、あるいは 補助的タイマー外部ゲート入力
		ST	
		ST	
QEB		ST	直交エンコーダ位相 A 入力、あるいは 補助的タイマーアップ / ダウン選択入力
		ST	
INDX		ST	直交エンコーダインデックスパルス入力
UPDN	O		ポジションアップ / ダウンカウンタ方向ステータス、 QEI モード

注: | = 入力、O = 出力、ST = シュミットトリガー

表 16-3: モジュール入出力モード関数

QEIM<2:0>	PCDOUT	UDSRC	TQGATE	TQCS	QEA ピン	QEB ピン	INDX ピン	UPDN ピン
000,010,011 モジュールオフ	N/A	N/A	N/A	N/A				
001 タイマーモード	N/A	0	0	0				
		1	0	0		入力 (UPDN)		
		0	1	0	入力 (TQGATE) ポート無効化されず			
		1	1	0	入力 (TQGATE) ポート無効化されず	入力 (UPDN)		
		0	N/A	1	入力 (TQCKI) ポート無効化されず			
101,111 QEI がカウント によりリセット	0	N/A	N/A	N/A	入力 (QEA)	入力 (QEB)		
	1	N/A	N/A	N/A	入力 (QEA)	入力 (QEB)		出力 (UPDN)
100,110 QEI がインデック スによりリセット	0	N/A	N/A	N/A	入力 (QEA)	入力 (QEB)	入力 (INDX)	
	1	N/A	N/A	N/A	入力 (QEA)	入力 (QEB)	入力 (INDX)	出力 (UPDN)

注: 空欄は QEI で使われていないことを表す。このときは汎用入出力ピンとして使われる。

## 16.9 省電力モード時の QEI の動作

### 16.9.1 デバイスがスリープモードに入った場合

デバイスがスリープモードになると、QEI は全ての動作を停止します。POSCNT はその時点の値で停止します。QEI は QEA、QEB、INDX あるいは UPDN ピンのアクティブな信号には反応しません。

QEICON レジスタは変化しません。

QEI が QEIM<2:0> = ‘001’ となりタイマー / カウンタとして設定され、TQCS = 1 でクロックが外部から提供されていた場合も、スリープ状態でモジュールは全ての動作を停止します。

モジュールがウェイクアップすると、直交デコーダにより QEA 信号または QEB 信号の次のトランジションが受け入れられ、そのトランジションがスリープモードに入る直前のトランジションと比較され、次の動作が決定されます。

### 16.9.2 デバイスが IDLE モードに入った場合

QEISIDL ビット (QEICON<13>) 次第でモジュールは IDLE モードでは省電力状態になります。

QEICSIDL = 1 の場合、モジュールは省電力モードに入り、スリープモードに入った場合と類似の動作を行います。

QEICSIDL = 0 の場合、モジュールは省電力モードになりません。モジュールはデバイスが IDLE モードでも通常どおり動作を続けます。

## 16.10 リセットの効果

リセットによりモジュールのレジスタが最初のリセット状態に戻されます。QEI モジュール関連レジスタの全ての初期化およびリセット条件についてはレジスタ 16-1を参照してください。

直交デコーダおよび POSCNT カウンタは初期状態にリセットされます。

表 16-4: QEI と関連する特別関数レジスタ

名称	ピット 15	ピット 14	ピット 13	ピット 12	ピット 11	ピット 10	ピット 9	ピット 8	ピット 7	ピット 6	ピット 5	ピット 4	ピット 3	ピット 2	ピット 1	ピット 0	全リセット時の数値
QEICON	CNTERR	不使用	QEISIDL	INDX	UPDN	QEIM2	QEIM1	QEIMO	SWPAB	PCDOUT	TQGATE	TQCKPS1	TQCKPS0	POSRES	TQCS	UDSRC	0000 0000 0000 0000
DFLTCON	—	—	—	—	—	—	—	—	QEOUT	QECK2	QECK1	QECKO	INDOUT	INDCK2	INDCK1	INDCK0	----- ----- -----
POSCNT	ポジションカウンタレジスタ															0000 0000 0000 0000	
MAXCNT	最大カウントレジスタ															1111 1111 1111 1111	
INTCON1	NSTDIS	—	—	—	—	OVATE	OVBTE	COVTE	—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000 0000 0000 0000
INTCON2	ALTIWT	—	—	—	—	—	—	—	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP	0000 0000 0000 0000
IFS2	—	—	—	FLTBIF	FLTAIF	LVDIF	DCIIF	QEIIIF	PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF	0000 0000 0000 0000
IEC2	—	—	—	FLTBIE	FLTAIE	LVDIE	DCIIE	QEIIIE	PWMIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE	0000 0000 0000 0000
IPC10	—	FLTAIP<2:0>			—	LVDIP<2:0>			—	DCIIP<2:0>			—	QEIIIP<2:0>			0100 0100 0100 0100

## 16.11 設計の秘訣

直交エンコーダインターフェース (QEI)

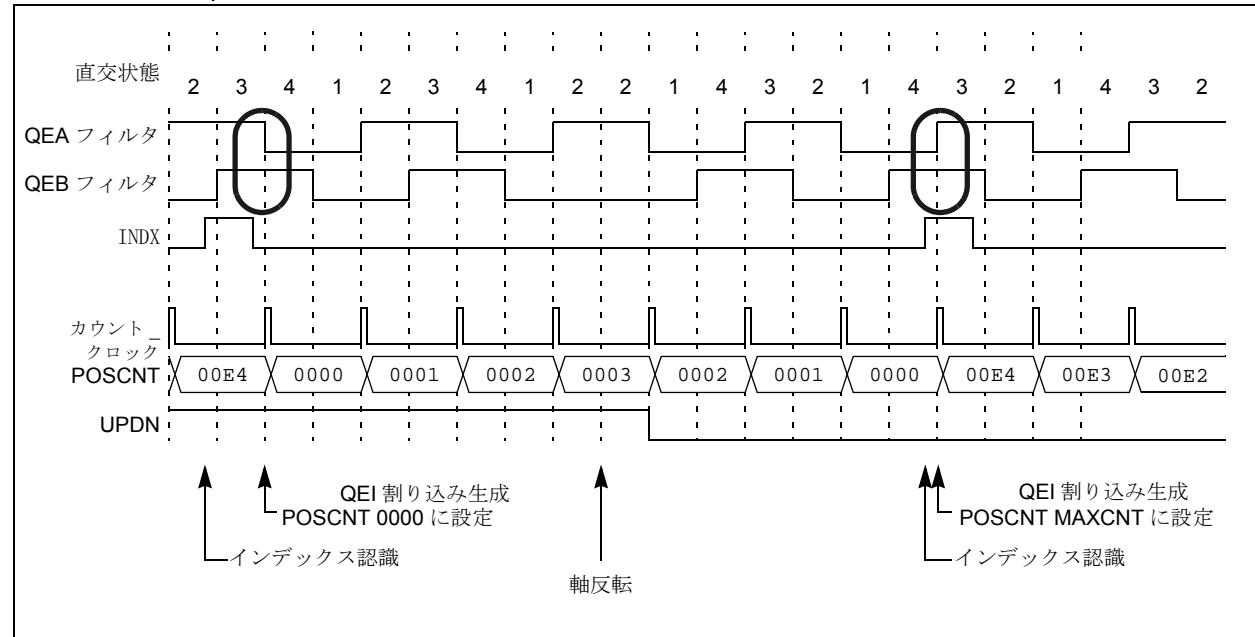
## 質問1: 直交信号のスピードはどれくらいですか。

回答: 回答は直交信号のフィルタ変数の設定により異なります。QEIでは、フィルタが使用されていない場合直交信号の周波数が  $F_{CY}/3$  未満であること、フィルタが使用されている場合周波数が  $F_{CY}/6$  未満であることが必要となります。

質問2: エンコーダのインデックスパルスが  $90^\circ$  度ですが、カウンタが正しくリセットされません。

回答: カウントクロックの生成方法、インデックスパルスにどの直交状態のトランジションが用いられているかによって、 $1/4$  サイクルインデックスパルスは必要なトランジションの前に認識されないことがあります。この状態を解決するためには、インデックスパルスよりもフィルタブリスケーラに分周比の大きいフィルタを直交クロックに用いてください。このことにより直交クロックに少しの遅れを発生させ、インデックスパルスを適切に検出することが可能となります。

図 16-12: インデックスモードにおけるリセット (インデックスパルス  $90^\circ$ ) – アップ/ダウンポジションカウンタ



## 16.12 関連するアプリケーションノート

このセクションでは、本章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正したり制限を設けて（必要な場合）使用可能です。現在、直交エンコーダインターフェース (QEI) と関連する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 #
ブラッシュモーターのサーボ制御	AN532
PIC18CXXX/PIC16CXXX DC サーボモーター	AN696

**注：** dsPIC30F ファミリーのデバイスに関する他のアプリケーションノートやコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 16.13 改訂履歴

### 改訂 A

これは本ドキュメントの初版です。

### 改訂 B

本版では、dsPIC30F の直交エンコーダインターフェース (QEI) モジュールについての詳細情報が追加されています。

直交エンコーダインターフェース (QEI)

注：



## 第 17 章 . 10 ビット A/D コンバータ

## ハイライト

このセクションには以下の項目が含まれています。

17

10 ビット A/D  
コンバータ

17.1 序章 .....	17-2
17.2 制御レジスタ .....	17-4
17.3 A/D 結果バッファ .....	17-4
17.4 A/D 用語と変換シーケンス .....	17-11
17.5 A/D モジュール構成 .....	17-13
17.6 電圧リファレンスソースの選択 .....	17-13
17.7 A/D 変換クロックの選択 .....	17-14
17.8 サンプリング用アナログ入力の選択 .....	17-15
17.9 モジュールの有効化 .....	17-17
17.10 サンプル / 変換シーケンスの指定 .....	17-17
17.11 サンプリングの開始方法 .....	17-18
17.12 サンプリング停止と変換開始の方法 .....	17-19
17.13 サンプリング / 変換動作の制御 .....	17-30
17.14 変換結果をバッファに書き込む方法の指定 .....	17-31
17.15 変換シーケンスの例 .....	17-32
17.16 A/D サンプリング要件 .....	17-46
17.17 A/D 結果バッファの読み込み .....	17-49
17.18 変換閾数 .....	17-50
17.19 A/D 精度 / 誤差 .....	17-50
17.20 接続の条件 .....	17-50
17.21 初期化 .....	17-51
17.22 SLEEP モードおよび IDLE モード時の動作 .....	17-52
17.23 リセットの影響 .....	17-52
17.24 10 ビット A/D コンバータに関する特殊関数レジスタ .....	17-53
17.25 設計の秘訣 .....	17-54
17.26 関連するアプリケーションノート .....	17-55
17.27 改訂履歴 .....	17-56

## 17.1 序章

以下は dsPIC30F の 10 ビット A/D コンバータの主な特徴です。

- 逐次近似 (SAR) 変換
- 最速 500ksps の変換スピード
- 最高 16 のアナログ入力ピン
- 外部電圧リファレンス入力ピン
- 4 つの単極性差動 S/H 増幅器
- 最高 4 アナログ入力ピンまでの同時サンプリング
- 自動チャンネルスキャニモード
- 選択可能な変換トリガーソース
- 16 ワードの変換結果バッファ
- 選択可能なバッファ書き込みモード
- 4 種類の結果フォーマットが選択可能
- CPU SLEEP モードと IDLE モードでも動作

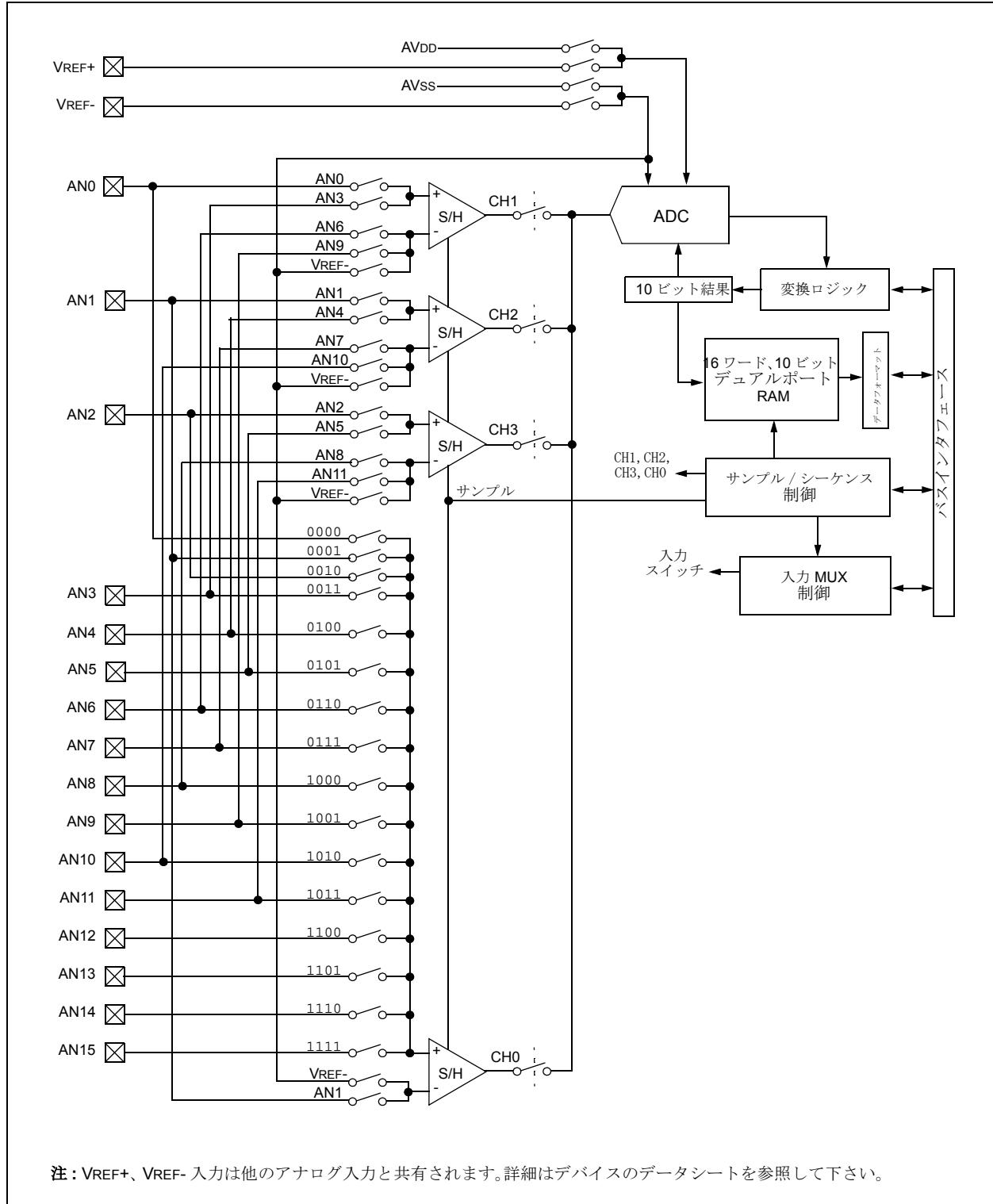
図 17-1 は 10 ビット A/D のブロック図です。10 ビット A/D コンバータには AN0-AN15 と呼ばれるアナログ入力ピンが最大 16 あります。さらに、外部電圧リファレンス接続のためのアナログ入力ピンが 2 つあります。これら電圧リファレンス入力は他のアナログ入力ピンと共有されます。実際のアナログ入力ピンの数と外部電圧リファレンス入力構成は特定の dsPIC30F デバイスによります。詳細はデバイスのデータシートを参照してください。

アナログ入力はマルチプレクサ経由で、CH0-CH3 と呼ばれる 4 つの S/H 増幅器に接続されます。S/H 増幅器の、1 つ、2 つ、または 4 つが入力データ取得のために有効化できます。アナログ入力マルチプレクサは、変換中 2 セットのアナログ入力間で切り替えられます。特定の入力ピンを用いて、すべてのチャネル間の単極性差動変換が可能です。(図 17-1 参照)。

アナログ入力スキャニモードは CHO S/H 増幅器のために有効化することができます。制御レジスタが、スキャニシークエンスにどのアナログ入力を含めるか指定します。

10 ビット A/D は 16 ワードの結果バッファに格納されます。10 ビット結果のそれぞれは、バッファから読み出されるとき、4 つの 16 ビット出力フォーマットのうちの 1 つに変換されます。

図17-1: 10ビット高速A/Dブロック図



## 17.2 制御レジスタ

A/D モジュールには制御とステータスの、以下の 6 つのレジスタがあります。

- ADCON1: A/D 制御レジスタ 1
- ADCON2: A/D 制御レジスタ 2
- ADCON3: A/D 制御レジスタ 3
- ADCHS: A/D 入力チャネル選択レジスタ
- ADPCFG: A/D ポート構成レジスタ
- ADCSSL: A/D 入力スキャン選択レジスタ

ADCON1、ADCON2 および ADCON3 レジスタは、A/D モジュールの動作を制御します。ADCHS レジスタは、S/H 増幅器に接続される入力ピンを選択します。ADPCFG レジスタは、アナログ入力ピンをアナログ入力あるいはデジタル入出力として構成します。ADCSSL レジスタは、連続してスキャンされる入力を選択します。

## 17.3 A/D 結果バッファ

モジュールには A/D 結果を格納する ADCBUF と呼ばれる 16 ワードのデュアルポート RAM があります。その 16 のバッファ区別は、ADCBUFO、ADCBUF1、ADCBUF2....ADCBUFE、ADCBUFF となります。

**注：** A/D 結果バッファは読み込みのみのバッファです。

## レジスタ 17-1: ADCON1: A/D 制御レジスタ 1

上位バイト :							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
ADON	—	ADSLDL	—	—	—	FORM<1:0>	
ビット 15							ビット 8

下位バイト :							
R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/C-0
				HC, HS	HC, HS		
SSRC<2:0>							ビット 0
ビット 7							

ビット 15 **ADON:** A/D 動作モードビット1 = A/D コンバータモジュールを動作中にする  
0 = A/D コンバータはオフになります。

ビット 14 未実装: ‘0’ が読み込まれます。

ビット 13 **ADSLDL:** IDLE モードでの停止ビット1 = デバイスが IDLE モードに入ったら、モジュール動作を中止します。  
0=0 = IDLE モードでのモジュール動作を継続します。

ビット 12-10 未実装: ‘0’ が読み込まれます。

ビット 9-8 **FORM<1:0>:** データ出力フォーマットビット  
11 = 符号付き固定小数 (DOUT = sddd dddd dd00 0000)  
10 = 固定小数 (DOUT = dddd dddd dd00 0000)  
01 = 符号付き整数 (DOUT = ssss ssss dddd dddd)  
00 = 整数 (DOUT = 0000 00dd dddd dddd)ビット 7-5 **SSRC<2:0>:** 変換トリガーソース選択ビット111 = 内部カウンターによりサンプリングが終わり次第変換を開始します。(自動変換)  
110 = 予約済み  
101 = 予約済み  
100 = 予約済み  
011 = モーター制御 PWM インターバルによりサンプリングが終わり、変換を開始します。  
010 = 汎用タイマー 3 比較によりサンプリングが終了し、変換を開始します。  
001 = INTO ピン上のアクティブ遷移によりサンプリングが終了し、変換を開始します。  
000 = SAMP ビットのクリアにより、サンプリングを終了し、変換を開始します。

ビット 4 未実装: ‘0’ が読み込まれます。

ビット 3 **SIMSAM:** 同時サンプル選択ビット (CHPS = 01 あるいは 1x の時のみ )1 = CH0、CH1、CH2、CH3 を同時にサンプルする (CHPS = 1x の時 )  
あるいは

CH0 と CH1 を同時にサンプルする (CHPS = 01 の時 )

0 = 多重チャネルを順にサンプルする

ビット 2 **ASAM:** A/D サンプル自動スタートビット1 = 最後の変換が完了後、ただちに、サンプリングを開始。SAMP ビットも自動セット。  
0 = SAMP ビットがセットされた時にサンプリングを開始。

## レジスタ 17-1: ADCON1: A/D 制御レジスタ 1 (続き)

### ビット 1 **SAMP:** A/D サンプル有効化ビット

1 = 少なくとも A/D サンプル / ホールド増幅器の 1 つがサンプリング中

0 = A/D サンプル / ホールド増幅器がホールド中

ASAM = 0 の時、このビットに ‘1’ を書き込むと、サンプリングが開始になります。

SSRC = 000 の時、このビットに ‘0’ を書き込むと、サンプリングが終了し、変換を開始します。

### ビット 0 **DONE:** A/D 変換ステータスビット (Rev. B シリコン以降)

1 = A/D 変換完了

0 = A/D 変換未完了

ソフトウェアによるクリアあるいは、新しい変換でクリア。

このビットのクリアは、進行中のどんな動作にも影響を与えません。

凡例：

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未定ビット、‘0’ が読み込まれます  
ト

HC = ハードウェアがクリア  
されます

HS = ハードウェアが C = ソフトウェアでクリア可能  
セットされます

-n = POR の値

‘1’ = ビットがセット ‘0’ = ビットがクリア x = ビットは不定  
されます されます

## レジスタ 17-2: ADCON2: A/D 制御レジスタ 2

上位バイト :							
R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
VCFG<2:0>	reserved	—	CSCNA	CHPS<1:0>			
ビット 15					ビット 8		

下位バイト :							
R-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUFS	—	SMPI<3:0>	BUFM	ALTS			
ビット 7					ビット 0		

ビット  
15-13**VCFG<2:0>**: 電圧リファレンス構成ビット

	A/D VREFH	A/D VREFL
000	AVDD	AVss
001	External VREF+ pin	AVss
010	AVDD	External VREF- pin
011	External VREF+ pin	External VREF- pin
1XX	AVDD	AVss

ビット 12 予約済み: ユーザーはここに ‘0’ を書き込んだほうがよい。

ビット 11 未実装: ‘0’ が読み込まれます。

ビット 10 CSCNA: MUX A 用 S/H 入力 + CH0 用スキャン入力選択 制御ビット

1 = スキャン入力

0 = 入力をスキャンしません。

ビット 9-8 CHPS&lt;1:0&gt;: 使用チャネル制御ビット

1x = CH0、CH1、CH2 および CH3 を変換

01 = CH0 および CH1 を変換

00 = CH0 を変換

SIMSAM ビット (ADCON1&lt;3&gt;) = 0 の時、複数チャネルが同時にサンプリングされます。

SIMSAM ビット (ADCON1&lt;3&gt;) = 1 の時、CHPS&lt;1:0&gt; のように複数チャネルがサンプリングされます。

ビット 7 BUFS: バッファステータスビット

BUFM = 1 (ADRES は 2 x 8- ワードバッファに分割される ) の時のみ有効。

1 = A/D は現在 バッファ 0x8-0xF を実行中、ユーザーは 0x0-0x7 にあるデータにアクセスします。

0 = A/D は現在バッファ 0x0-0x7 を実行中、ユーザーは 0x8-0xF にあるデータにアクセスします。

ビット 6 未実装: ‘0’ が読み込まれます

ビット 5-2 SMP&lt;3:0&gt;: 割り込み毎のサンプル / 変換回数選択ビット

1111 = 16 回目のサンプル / 変換シーケンス毎、変換完了時に割り込みます。

1110 = 15 回目のサンプル / 変換シーケンス毎、変換完了時に割り込みます。

.....

0001 = 2 回目のサンプル / 変換シーケンス毎、変換完了時に割り込みます。

0000 = サンプル / 変換シーケンスの都度、変換完了時に割り込みます。

ビット 1 BUFM: バッファモード選択ビット

1 = バッファは、ADCBUF(15...8)、ADCBUF(7...0) の 2 つの 8- ワードバッファとして構成されます。

0 = バッファは、ADCBUF(15...0) の 16 ワードバッファ 1 つとして構成されます。

ビット 0 ALTS: 交差入力サンプルモード選択ビット

1 = 最初のサンプルには MUX A を使用し、その後のサンプルには、MUX B および MUXA 入力を交差に使用します。

0 = 常に MUX A 側を使用します。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未定ビット、‘0’ が読み込まれます  
ト

-n = POR の値

'1' = ビットがセット '0' = ビットがクリア x = ビットは不定  
されま

## レジスタ 17-3: ADCON3: A/D 制御レジスタ 3

上位バイト :								
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
—	—	—	SAMC<4:0>					
ビット 15						ビット 8		

下位バイト :							
R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADRC	—	ADCS<5:0>					
ビット 7						ビット 0	

ビット 15-13 未実装: ‘0’ が読み込まれます。

ビット 12-8 **SAMC<4:0>**: 自動サンプルタイムビット  
 11111 = 31 TAD  
 .....  
 00001 = 1 TAD  
 00000 = 0 TAD (S/H 増幅器を 1 つ以上用いた連続変換を行う時のみ可能)

ビット 7 **ADRC**: A/D 変換クロックソースビット  
 1 = A/D 内部 RC クロック  
 0 = システムクロック

ビット 6 未実装: ‘0’ が読み込まれます。

ビット 5-0 **ADCS<5:0>**: A/D 変換クロック選択ビット  
 111111 = TCY/2 • (ADCS<5:0> + 1) = 32 • TCY  
 .....  
 000001 = TCY/2 • (ADCS<5:0> + 1) = TCY  
 000000 = TCY/2 • (ADCS<5:0> + 1) = TCY/2

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未定ビット、‘0’ が読み込まれます

ト

-n = POR の値

‘1’ = ビットがセット ‘0’ = ビットがクリア x = ビットは不定

されます されます

## レジスタ 17-4: ADCHS: A/D 入力選択レジスタ

上位バイト :									
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		
CH123NB<1:0>	CH123SB	CH0NB		CH0SB<3:0>					
ビット 15				ビット 8					

下位バイト :									
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		
CH123NA<1:0>	CH123SA	CH0NA		CH0SA<3:0>					
ビット 7				ビット 0					

ビット 15-14 **CH123NB<1:0>: MUX B マルチプレクサ用チャネル 1、2、3 負の入力選択設定ビット**  
ビット 6-7 (ノート参照)と同じ定義

ビット 13 **CH123SB: MUX B マルチプレクサ用チャネル 1、2、3 正の入力選択設定ビット**  
ビット 5 (ノート参照)と同じ定義

ビット 12 **CH0NB: MUX B マルチプレクサ用チャネル 0 負の入力選択設定ビット**  
ビット 4 (ノート参照)と同じ定義

ビット 11-8 **CH0SB<3:0>: MUX B マルチプレクサ用チャネル 0 正の入力選択設定ビット**  
ビット 3-0 (ノート参照)と同じ定義

ビット 7-6 **CH123NA<1:0>: MUX A マルチプレクサ用チャネル 1、2、3 負の入力選択設定ビット**  
11 = CH1 負の入力は AN9、CH2 負の入力は AN10、CH3 負の入力は AN11  
10 = CH1 負の入力は AN6、CH2 負の入力は AN7、CH3 負の入力は AN8  
0x = CH1、CH2、CH3 負の入力は VREF-

ビット 5 **CH123SA: MUX A マルチプレクサ用チャネル 1、2、3 正の入力選択設定ビット**  
1 = CH1 正の入力は AN3、CH2 正の入力は AN4、CH3 正の入力は AN5  
0 = CH1 正の入力は AN0、CH2 正の入力は AN1、CH3 正の入力は AN2

ビット 4 **CH0NA: MUX A マルチプレクサ用チャネル 0 負の入力選択設定ビット**  
1 = チャネル 0 負の入力は AN1  
0 = チャネル 0 負の入力は VREF-

ビット 3-0 **CH0SA<3:0>: MUX A マルチプレクサ用チャネル 0 正の入力選択設定ビット**  
1111 = チャネル 0 正の入力は AN15  
1110 = チャネル 0 正の入力は AN14  
1101 = チャネル 0 正の入力は AN13  
||  
||  
||

0001 = チャネル 0 正の入力は AN1  
0000 = チャネル 0 正の入力は AN0

**注:** アナログ入力マルチプレクサは、MUXA および MUX B という 2 つの入力設定構成をサポートします。ADCHS<15:8> は MUX B の設定を決定し、ADCHS<7:0> は MUX A の設定を決定します。いずれの制御ビットも同一の機能をします。

## 凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未定ビット、'0' が読み込まれます

-n = POR の値

'1' = ビットがセット '0' = ビットがクリア x = ビットは不定されます

## レジスタ 17-5: ADPCFG: A/D ポート構成レジスタ

上位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
ビット 15							ビット 8

下位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
ビット 7							ビット 0

ビット 15-0 **PCFG<15:0>**: アナログ入力ピン構成制御ビット  
 1 = アナログ入力ピンはデジタルモード、ポート読み込み入力有効化、A/D 入力マルチプレクサの入力は AVSS に接続  
 0 = アナログ入力ピンはアナログモード、ポート読み込み入力無効化、A/D はピン電圧をサンプリングします。

凡例 :

R = 読み込み可能ビット      W = 書き込み可能ビット      U = 未定ビット、'0' が読み込まれます  
 ト

-n = POR の値      '1' = ビットがセット      '0' = ビットがクリア      x = ビットは不定されます

## レジスタ 17-6: ADCSSL: A/D 入力スキャン選択レジスタ

上位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8
ビット 15							ビット 8

下位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0
ビット 7							ビット 0

ビット 15-0 **CSSL<15:0>**: A/D 入力ピンスキャン選択ビット  
 1 = 入力スキャン用の ANx を選択  
 0 = 入力スキャン用の ANx をスキップします。

凡例 :

R = 読み込み可能ビット      W = 書き込み可能ビット      U = 未定ビット、'0' が読み込まれます  
 ト

-n = POR の値      '1' = ビットがセット      '0' = ビットがクリア      x = ビットは不定されます

## 17.4 A/D 用語と変換シーケンス

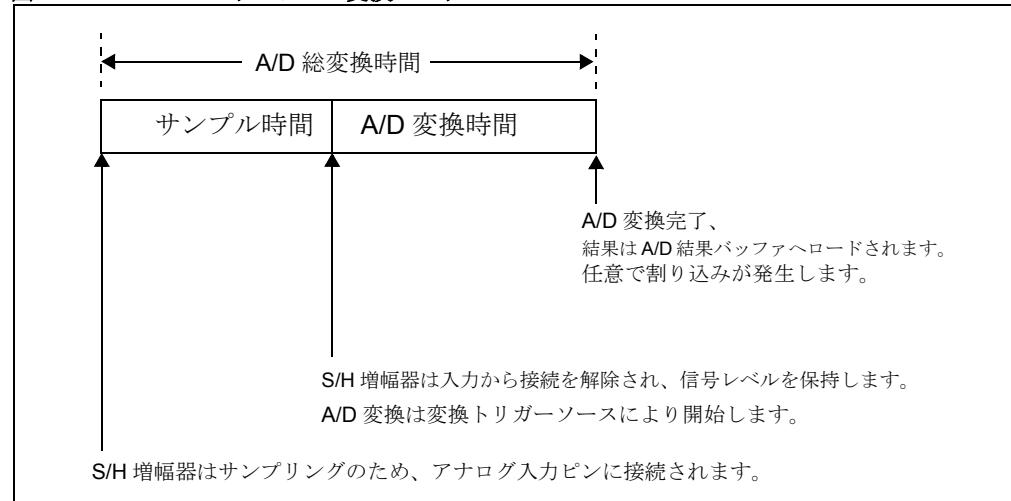
図 17-2 は基本的な変換シーケンスと用語です。アナログ入力ピン電圧のサンプリングは、サンプルホールドアンプ (S/H 増幅器) によって行われます。S/H 増幅器は S/H チャネルとも呼びます。10-ビット A/D コンバータは CH0 から CH3 と呼ばれる S/H チャネル 4 つからなります。アナログ入力マルチプレクサを通してアナログ入力ピンに接続されます。アナログ入力マルチプレクサは、ADCHS レジスタで制御されます。同一の機能をするマルチプレクサ制御ビットが 2 セット ADCHS レジスタにあります。この 2 セットの制御ビットは、MUX A および MUX B という 2 つの異なったアナログ入力マルチプレクサ構成のプログラムを可能にします。A/D コンバータは変換の合間に、MUX A および MUX B の構成を切り替えることができます。A/D コンバータはまた、一連のアナログ入力を連続スキャンすることができます。

サンプル時間は、A/D モジュールの S/H 増幅器がアナログ入力ピンに接続されている時間です。サンプル時間は、SAMP ビット (ADCON1<1>) の設定により手動で、あるいは A/D コンバータハードウェアにより自動的にスタートします。サンプル時間の終了は、ユーザーのソフトウェアで SAMP 制御ビットのクリアにより手動で、あるいは変換トリガーソースにより自動的に行われます。

変換時間は A/D コンバータが、S/H 増幅器のもつ電圧を変換するのに要する時間です。A/D はサンプル時間の最後にアナログ入力ピンから接続を解除されます。A/D コンバータは、1 ビット分の変換に 1 A/D クロックサイクル ( $T_{AD}$ ) を必要とし、さらにもう 1 サイクル必要とします。従って変換を完成するには合わせて 12 TAD サイクルが必要です。変換が終了すると、結果は 16 A/D 結果レジスタのいずれか (ADCBUFO...ADCBUFF) にロードされ、S/H は入力ピンに再接続されさらに CPU 割り込みを発生します。

サンプル時間と A/D 変換時間の総計が、総変換時間になります。S/H 増幅器が A/D 変換の精度保証するため、最低サンプル時間があります (セクション 17.16 「A/D サンプリング要件」を参照)。さらに、A/D コンバータのための多様な入力クロックオプションがあります。ユーザーは最低 TAD 仕様に反しない入力クロックオプションを選択する必要があります。

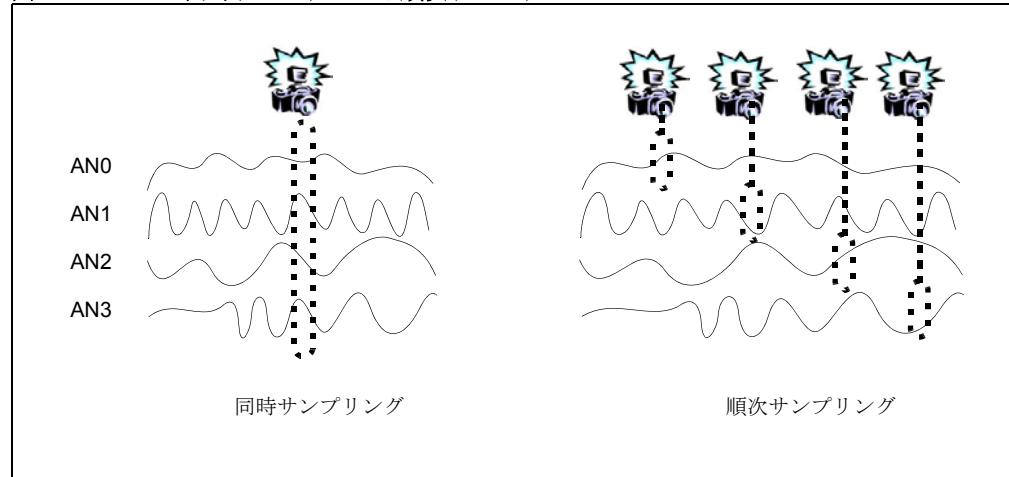
図 17-2: A/D サンプル / 変換シーケンス



10-ビット A/D コンバータは、サンプル / 変換シーケンスとして多数のオプションを可能にします。サンプル / 変換シーケンスは、図 17-3 のようにとても簡単です。図 17-3 では 1 つの S/H 増幅器のみを使用しています。より入念なサンプル / 変換シーケンスにより複数の S/H 増幅器を使用した多重変換が実行できます。10-ビット A/D コンバータは、サンプル / 変換シーケンスで 2 変換を実行するため 2S/H 増幅器、あるいは 4 変換に 4S/H 増幅器を使用します。サンプル / 変換シーケンスで使用される S/H 増幅器、あるいはサンプル毎のチャネルの数は、CHPS 制御ビットで決定されます。

複数の S/H チャネルを使用するサンプル・変換シーケンスは、SIMSAM ビット (ADCON1<3>) の制御に従い、同時サンプリングまたは順次サンプリングが可能です。複数の信号の同時サンプリングにより、すべての入力で全く同時に、アナログ入力のスナップショットが起こります。順次サンプリングは、各アナログ入力のスナップショットを、各々の入力で変換が開始される直前に作成します。従ってこの場合の複数の入力のサンプリングは相関はありません。

図 17-3: 同時サンプリングと順次サンプリング



サンプリングの開始時間は **SAMP** 制御ビットをセットすることによりソフトウェアで制御できます。サンプリング時間の開始は、またハードウェアによっても自動的に制御されます。A/D コンバータが自動サンプルモードで動作する時、S/H 増幅器はサンプル・変換シーケンスでの変換の終了時にアナログ入力ピンに再接続されます。自動サンプル機能は **ASAM** 制御ビット (ADCON1<2>) によって制御されます。

変換トリガーソースにより、サンプリング時間が終了し、A/D 変換またはサンプル / 変換シーケンスが開始されます。変換トリガーソースは **SSRC** 制御ビットにより選択されます。変換トリガーは多様なハードウェアソースから取得できます。あるいは、**SAMP** 制御ビットをクリアすることによりソフトウェア内で手動で制御できます。変換トリガーソースの 1 つは自動変換です。自動変換の間隔は、カウンタと A/D クロックにより設定されます。自動サンプルモードおよび自動変換トリガーと一緒に使用することで、ソフトウェア割り込みのないエンドレスな自動変換が可能になります。

**SMPI** 制御ビット **ADCON2<5:2>** の値の決定に従って、各サンプル / 変換シーケンスの終了時、または複数のサンプル / 変換シーケンスの終了時に割り込みが生成されます。次の割り込みが起こるまでのサンプル / 変換シーケンスの数は、1 から 16 の間で変化します。**SMPI** 値を選択した場合、A/D 変換バッファで 16 の結果が保持されることに注意します。割り込み間の変換結果の合計数は、サンプル当たりのチャネル数と **SMPI** 値の積です。割り込み間の変換の合計回数が、バッファ長を超えてはいけません。

## 17.5 A/D モジュール構成

A/D 変換を動作させるには、以下の手順を実行します。

1. A/D モジュールを設定します。
  - アナログ入力としてポートピンを選択します。ADPCFG<15:0>
  - アナログ入力の電圧範囲に合う電圧リファレンスソースを選択します。ADCON2<15:13>
  - 望ましいデータ変換速度に合うアナログ変換クロックを選択します。ADCON3<5:0>
  - 使用する S/H チャネル数を決定します。ADCON2<9:8> と ADPCFG<15:0>
  - サンプリング方法を決定します。ADCON1<3> と ADCSSL<15:0>
  - 入力をどのように S/H チャネルに割り当てるかを決定します。ADCHS<15:0>
  - 適切なサンプル / 変換シーケンスを選択します。ADCON1<7:0> と ADCON3<12:8>
  - 変換結果をバッファにどのフォーマットで格納するかを選択します。ADCON1<9:8>
  - 割り込み当たりの変換回数を選択します。ADCON2<5:9>
  - A/D モジュールをオンにします。ADCON1<15>
2. A/D 割り込みを構成します。(必要に応じて)
  - ADIF ビットをクリアします。
  - A/D 割り込み優先順位を選択します。

各構成手順の選択肢については、以下に後述します。

## 17.6 電圧リファレンスソースの選択

A/D 変換のための電圧リファレンスは、VCFG<2 : 0> 制御ビット (ADCON2<15 : 13>) を使用して選択されます。高压側電圧リファレンス (VREFH) および低压側電圧リファレンス (VREFL) は、内部 AVDD および AVSS 電圧範囲か、VREF+ および VREF- 入力ピンになります。

外部電圧リファレンスピンは、少ピンデバイスの場合には AN0 入力と AN1 入力で共有される場合があります。A/D コンバータは、VREF+ および VREF- 入力ピンで共有される場合でも、このピンの電圧の変換を実行します。

外部リファレンスピンに適用される電圧は、特定の仕様に合わせる必要があります。詳しくはデバイスデータシートの「電気的仕様」セクションを参照してください。

## 17.7 A/D 変換クロックの選択

A/D 変換には変換が完了できる最大変換レートがあります。アナログモジュールクロックである TAD は、変換のタイミングを制御します。A/D 変換には 12 クロック周期 (12 TAD) が必要です。A/D クロックは、デバイス命令クロックまたは内部 RC クロックソースから導かれます。

A/D 変換クロックの周期は、6 ビットカウンタを使用してソフトウェアで選択できます。TAD には 64 のオプションが可能であり、ADCS<5:0> ビット (ADCON3<5:0>) により特定されます。式 17-1 から、ADCS 制御ビット、およびデバイス命令クロック周期 TCY の関数として TAD 値が求められます。

式 17-1: A/D 変換クロック周期

$TAD = \frac{TCY(ADCS + 1)}{2}$ $ADCS = \frac{2TAD}{TCY} - 1$
--

正しい A/D 変換のために、167ns の最低 TAD 時間を維持できるように A/D 変換クロック (TAD) が選択される必要があります。表 17-1 に、デバイス操作周波数および選択された A/D クロックソースから導かれた最終的な時間 TAD を示しています。

A/D コンバータには、変換を実行するために使用できる専用の内蔵 RC クロックソースがあります。内蔵 RC クロックソースは、dsPIC30F が SLEEP モードの間に A/D 変換を実行する場合に使用されなければなりません。内蔵 RC オシレータは ADRC ビット (ADCON3<7>) の設定により選択されます。ADRC ビットが設定される場合、ADCS<5:0> ビットは A/D 操作に関して無効となります。

表 17-1: 標準的な TAD とデバイス操作周波数

AD クロックインターバル (TAD)/TCONV							
AD クロックソース選択			デバイス TCY/ デバイス FcY				
クロック	ADRC	ADCS<5:0>	33.33 nsec/ 30 MHz	40 nsec/ 25 MHz	80 nsec/ 12.5 MHz	160 nsec/ 6.25 MHz	1000 nsec/ 1 MHz
TCY 2	0	000000	(2)	(2)	(2)	(2)	500 ns/ 6.0 μs
TCY	0	000001	(2)	(2)	(2)	(2)	1.0 μs/ 12 μs
2 TCY	0	000011	(2)	(2)	(2)	320 ns/ 3.86 μs	2.0 μs <sup>(3)</sup> / 24 μs
4 TCY	0	000111	(2)	(2)	320 ns/ 3.86 μs	640 ns <sup>(3)</sup> / 7.68 μs	4.0 μs <sup>(3)</sup> / 48 μs
8 TCY	0	001111	266.64 ns/ 3.2 μs	320 ns/ 3.86 μs	640 ns <sup>(3)</sup> / 7.68 μs	1.28 μs <sup>(3)</sup> / 15.36 μs	8.0 μs <sup>(3)</sup> / 96
16 TCY	0	011111	533.28 ns <sup>(3)</sup> / 6.4 μs	640 ns <sup>(3)</sup> / 7.68 μs	1.28 μs <sup>(3)</sup> / 15.36 μs	2.56 μs <sup>(3)</sup> / 30.72 μs	16.0 μs <sup>(3)</sup> / 192
32 TCY	0	111111	1066.56 ns <sup>(3)</sup> / 12.8 μs	1280 ns <sup>(3)</sup> / 15.36 μs	2.56 μs <sup>(3)</sup> / 30.72 μs	5.12 μs <sup>(3)</sup> / 61.44 μs	32.0 μs <sup>(3)</sup> / 384
RC	1	xxxxxx	200-400 ns/ 3.9 μs <sup>(1,4)</sup>	200-400 ns/ 3.9 μs <sup>(1)</sup>			

- 注 1: RC ソースの通常の TAD 時間は、は VDD > 3.0V の場合 300 ns になります。  
 2: これらの値は TAD 時間に最低必要な 167 ns に違反しています (仕様についてはデバイスのデータシートを参照)。  
 3: 変換時間を高速化するには、別のクロックソースを選択することをお薦めします。  
 4: A/D では、RC クロックソースで Fosc > 20 MHz の場合には完全な精度を得ることはできません。

## 17.8 サンプリング用アナログ入力の選択

サンプル / ホールド増幅器は、どのアナログ入力がサンプルされるかを選択するために、非反転および反転入力双方にアナログマルチプレクサ（図 17-1 を参照）が付いています。サンプル / 変換シーケンスがいったん設定されると、ADCHS ビットにより各サンプルのためにどのアナログ入力が選択されるかが決定されます。

さらに、選択された入力は、交互サンプルに基づいて変化したり、反復されるサンプルシーケンスで変化する場合があります。

**注：** デバイスごとにアナログ入力の数は異なります。アナログ入力の利用については、デバイスデータシートで確認してください。

### 17.8.1 アナログポートピンの構成

ADPCFG レジスタは、アナログ入力として使用されているデバイスピンの入力条件を指定します。

対応する PCFGn ビット ( $ADPCFG<n>$ ) がクリアされると、ピンはアナログ入力として構成されます。ADPCFG レジスタは、RESET 時にはクリアされ、これにより、デフォルトで RESET 時には A/D 入力ピンとして構成されます。

アナログ入力に構成された場合、対応するポートの I/O デジタル入力バッファが無効になるため、電流を消費しません。

ADPCFG レジスタと TRISB レジスタは、A/D ポートピンの動作を制御します。

アナログ入力とするポートピンは、対応する TRIS ビットがセットされ、入力を指定している必要があります。TRIS レジスタをクリアして、A/D 入力に関連する I/O ピンが出力として構成された場合、このポートのデジタル出力レベル ( $V_{OH}$  または  $V_{OL}$ ) が A/D 変換されます。デバイスが RESET されると、TRIS ビットはすべてセットにされます。

対応する PCFGn ビット ( $ADPCFG<n>$ ) がセットされた場合、ピンはデジタル I/O として構成されます。この構成において、アナログマルチプレクサへの入力は  $AV_{ss}$  に接続されます。

**注**

- 1:** A/D ポートレジスタ読み込み時に、アナログ入力として構成されているどのピンも ‘0’ として読み込まれます。
- 2:** デジタル入力として定義されたピン (AN15:AN0 ピンを含む) にアナログ信号を入力すると、入力バッファがデバイスの仕様に準拠しない電流を消費する可能性があります。

### 17.8.2 チャネル 0 入力選択

チャネル 0 は、アナログ入力の選択に関しては、4 つの S/H チャネルで最も柔軟性があります。

ユーザーはチャネルの正入力への入力として、最大 16 のアナログ入力のうちのいずれかを選択できます。CH0SA<3:0> ビット ( $ADCHS<3:0>$ ) は、通常は、チャネル 0 の正入力に対してアナログ入力を選択します。

ユーザーは、チャネルの負入力として VREF- または AN1 を選択できます。CHONA ビット ( $ADCHS<4>$ ) は、通常はチャネル 0 の負入力に対してアナログ入力を選択します。

## 17.8.2.1 交互チャネル 0 入力選択の指定

ALTS ビット (ADCON2<0>) を指定すると、モジュールは連続したサンプルの間に選択された 2 組の入力を交互に切り替えます。

CH0SA<3:0>、CHONA、CHXSA、および CHXNA<1:0> により指定された入力は、まとめて MUX A 入力と呼ばれます。CH0SB<3:0>、CHONB、CHXSB、および CHXNB<1:0> により指定される入力は、まとめて MUX B 入力と呼ばれます。ALTS ビットが ‘1’ の時、モジュールは MUX A の入力をサンプルした次のサンプルでは MUX B の入力をサンプルして交互に切り替えます。

チャネル 0 に関して、ALTS ビットが ‘0’ の場合、CH0SA<3:0> と CHONA により指定された MUX A 入力のみがサンプリングに選択されます。

ALTS ビットが ‘1’ の場合、チャネル 0 の最初のサンプル・変換シーケンスでは、CH0SA<3:0> と CHONA により指定された入力がサンプリングに選択されます。チャネル 0 の次のサンプル変換シーケンスでは、CH0SB<3:0> と CHONB により指定された入力がサンプリングに選択されます。以降のサンプル変換シーケンスに対して、このパターンが繰り返されます。

複数のチャネル (CHPS = 01 または 1x) と同時サンプリング (SIMSAM = 1) が指定されている場合、サンプル時間ごとにすべてのチャネルがサンプリングされるため、交互入力によりすべてのサンプルが変換されます。複数のチャネル (CHPS = 01 または 1x) と順次サンプリング (SIMSAM = 0) が指定されている場合、交互入力により特定チャネルのそれぞれのサンプルのみが変換されます。

## 17.8.2.2 チャネル 0 の複数入力のスキヤニング

チャネル 0 には入力の選択ベクトルに応じて自動スキヤンする能力があります。CSCNA ビット (ADCON2<10>) により、選択した数のアナログ入力すべてを CH0 チャネル入力としてスキヤンできます。CSCNA が設定されると、CH0SA<3:0> ビットは無視されます。

ADCSSL レジスタは、スキヤンする入力を指定します。ADCSSL レジスタの各ビットは、アナログ入力に対応しています。ビット 0 は AN0 に、ビット 1 は AN1 に、のように対応します。ADCSSL レジスタの特定のビットが ‘1’ の場合、対応する入力はスキヤンシーケンスの一部です。入力は常に、割り込みが発生した後、選択された最初のチャネルから始まり、低い番号の入力から高い番号の入力にスキヤンされます。

**注：** スキヤン対象の入力に選択した数が、各割り込み毎に採取されるサンプル数よりも大きい場合、高位の番号の入力はサンプリングされません。

ADCSSL は、チャネルの正入力の入力を指定するにすぎません。CHONA ビットは、スキヤンの間も、チャネルの負入力の入力を選択します。

ALTS ビットが ‘1’ ならば、スキヤンは MUX A 入力選択にのみに適用されます。CH0SB<3:0> の指定に従って MUX B の入力を選択した場合には、代替チャネル 0 入力が選択されます。この方法で入力選択がプログラミングされた場合、チャネル 0 入力は、ADCSSL レジスタで指定される一連のスキヤン入力と、CHOSB ビットで指定される固定入力を交互に切り替えます。

### 17.8.3 チャネル 1、2、3 の入力選択

チャネル 1、2、3 はアナログ入力ピンのサブセットをサンプリングできます。チャネル 1、2、3 は 3 つの入力から構成される 2 つのグループのいずれかを選択します。

CHXSA ビット (ADCHS<5>) は、チャネル 1、2、3 の正入力のソースを選択します。

CHXSA をクリアすると、AN0、AN1、AN2 がチャネル 1、2、3 の正入力のアナログソースとして選択されます。CHXSA をセットした場合、AN3、AN4、AN5 がアナログソースとして選択されます。

CHXNA<1:0> ビット (ADCHS<7:6>) は、チャネル 1、2、3 の負入力のソースを選択します。

CHXNA = 0x をプログラミングした場合、VREF- がチャネル 1、2、3 の負入力のアナログソースとして選択されます。CHXNA = 10 をプログラミングした場合、AN6、AN7、AN8 がチャネル 1、2、3 の負入力のアナログソースとして選択されます。CHXNA = 11 のプログラミングの場合、AN9、AN10、AN11 がアナログソースとして選択されます。

#### 17.8.3.1 チャネル 1、2、3 の交互入力選択の指定

チャネル 0 入力の場合と同様に、ALTS ビット (ADCON2<0>) の設定により、モジュールはチャネル 1、2、3 の連続的なサンプリングでは、選択された 2 組の入力を交互に使用します。

CHXSA と CHXNA<1:0> により指定される MUX A 入力は、常に ALTS = 0 の場合の入力として選択されます。

MUX A 入力は、ALTS = 1 の場合、CHXSB と CHXNB<1:0> で指定される MUX B 入力に交互に切り替わります。

### 17.9 モジュールの有効化

ADON ビット (ADCON1<15>) が ‘1’ の時、モジュールはアクティブモードにあり、十分に電力が供給され、動作中です。

ADON が ‘0’ の時、モジュールは無効です。回路のデジタルおよびアナログ部分は最大省電力のためにオフにされます。

Off モードからアクティブモードに戻るためには、アナログステージが安定するのを待つ必要があります。安定化時間については、デバイスデータシートの電気的仕様セクションを参照してください。

**注:** ADCON3 と ADCSSL レジスタ以外に、SSRC<2:0>、SIMSAM、ASAM、CHPS<1:0>、SMPI<3:0>、BUFM、および ALTS ビットも ADON = 1 の間は書き込まないでください。書き込み結果が確定されなくなります。

### 17.10 サンプル / 変換シーケンスの指定

10 ビット A/D モジュールには、4 つのサンプル / ホールド増幅器と 1 つの A/D コンバータがあります。このモジュールは入力 1、2、または 4 のサンプリングと、A/D 変換をサンプル / 変換シーケンスごとに実行します。

#### 17.10.1 サンプル / ホールドチャネルの数

CHPS<1:0> 制御ビット (ADCON2<9:8>) は、サンプル / 変換シーケンスの間に A/D モジュールが使用する S/H 増幅器の数を選択する場合に使用します。以下の 3 つのオプションから選択できます。

- CH0 のみ
- CH0 と CH1
- CH0、CH1、CH2、CH3

CHPS 制御ビットは、SIMSAM (同時サンプル) 制御ビット (ADCON1<3>) と連動します。

### 17.10.2 同時サンプリングの有効化

アプリケーションの中には、全く同じ時間に複数の信号のサンプリングを要求するものがあります。SIMSAM 制御ビット (ADCON1<3>) は、CHPS 制御ビットと連動し、表 17-2 に示すような複数のチャネルのサンプル / 変換シーケンスを制御します。SIMSAM 制御ビットは、**CHPS<1:0> = 00** の場合、モジュール動作に影響しません。**CHPS** 制御ビットにより複数の S/H 増幅器が有効になり、SIMSAM ビットが **0** の場合、**2** または **4** 周期のサンプリング期間に、選択した **2** つまたは **4** つのチャネルがサンプリングされ、順次変換されます。SIMSAM ビットが '**1**' の場合、**1** 回のサンプル期間に、選択した **2** つまたは **4** つのチャネルが同時にサンプリングされます。この後、チャネルは順次変換されます。

表 17-2: サンプル / 変換制御オプション

CHPS<1:0>	SIMSAM	サンプル / 変換シーケンス	# サンプル / 変換サイクル数	例
00	x	サンプル CH0、変換 CH0	1	図 17-4、 図 17-5、 図 17-6、 図 17-7、 図 17-10、 図 17-11、 図 17-14、 図 17-15
01	0	サンプル CH0、変換 CH0 サンプル CH1、変換 CH1	2	
1x	0	サンプル CH0、変換 CH0 サンプル CH1、変換 CH1 サンプル CH2、変換 CH2 サンプル CH3、変換 CH3	4	図 17-9、 図 17-13、 図 17-20
01	1	サンプル CH0、CH1 同時 変換 CH0 変換 CH1	1	図 17-18
1x	1	サンプル CH0、CH1、CH2、CH3 同時 変換 CH0 変換 CH1 変換 CH2 変換 CH3	1	図 17-8 図 17-12、 図 17-16、 図 17-17、 図 17-9、

### 17.11 サンプリングの開始方法

#### 17.11.1 手動

SAMP ビット (ADCON1<1>) をセットすると、A/D のサンプリングが開始されます。複数のオプションのいずれかを使用すると、サンプリングを終了し、変換を実行できます。SAMP ビットが再度設定されるまで、サンプリングは再開しません。例として、図 17-4 を参照してください。

#### 17.11.2 自動

ASAM ビット (ADCON1<2>) をセットすると、チャネルで変換が行われていなければ、そのチャネルのサンプリングが自動的に開始されます。サンプリングを終了し、変換を実行する場合は、複数のオプションのいずれかを使用します。SIMSAM ビットで順次サンプリングが指定されている場合、チャネルの変換の終了後に、そのチャネルでサンプリングが再開します。SIMSAM ビットで同時サンプリングが指定されている場合、すべてのチャネルの変換が終了した後に、チャネルでサンプリングが再開します。例として、図 17-5 を参照してください。

## 17.12 サンプリング停止と変換開始の方法

変換トリガーソースはサンプリングを終了し、変換の特定のシーケンスを開始します。SSRC<2:0> ビット (ADCON1<7:5>) は、変換トリガーのソースを選択します。

**注：** 利用可能な変換トリガーソースは dsPIC30F デバイスの種類により変化します。利用可能な変換トリガーソースに関してはデバイスデータシートを参照してください。

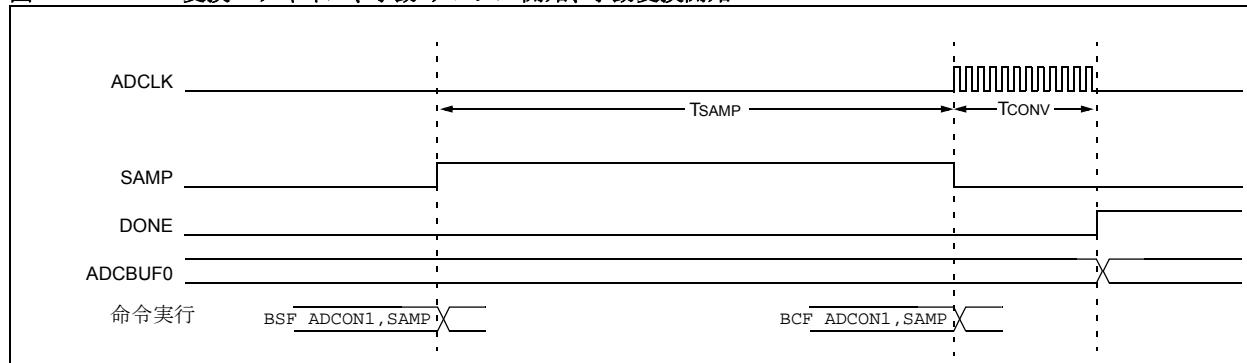
**注：** A/D モジュールが有効に設定されている場合、SSRC 選択ビットは変更できません。変換トリガーソースを変更する場合、まず ADON ビット (ADCON1<15>) をクリアして A/D モジュールを無効にする必要があります。

### 17.12.1 手動

SSRC<2:0> = 000 の時、変換トリガーはソフトウェアに制御されています。SAMP ビット (ADCON1<1>) をクリアすることにより、変換シーケンスが開始されます。

図 17-4 の例では、SAMP ビットのセットによりサンプリングが開始され、SAMP ビットのクリアにより、サンプリングが終了して変換が開始されます。ユーザー側のソフトウェアは、入力信号の十分なサンプリング時間を確保するよう SAMP ビットのセットとクリアの時間を調整する必要があります。コードの例は例 17-1 を参照してください。

図 17-4： 変換 1 チャネル、手動サンプル開始、手動変換開始



例 17-1: 1 チャネル変換、手動サンプル開始、手動変換開始コード

```

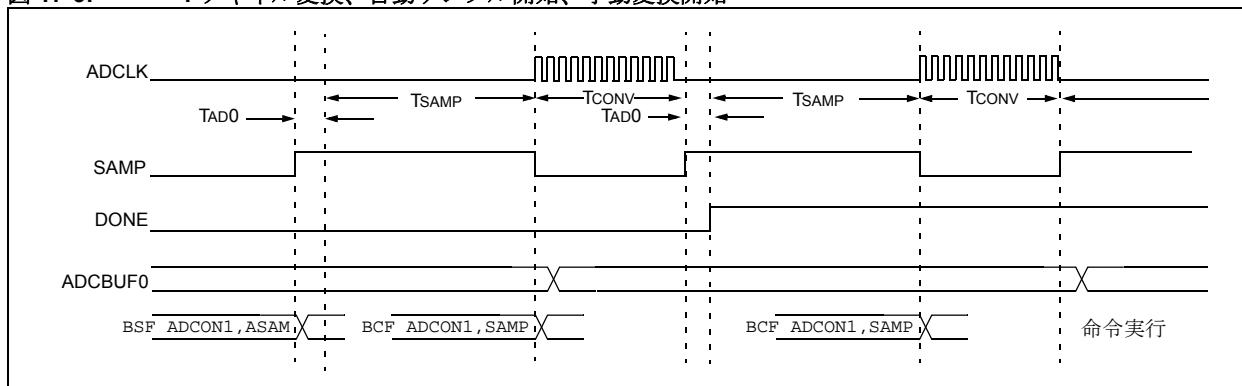
ADPCFG = 0xFFFF;
ADCON1 = 0x0000;                                // all PORTB = Digital; RB2 = analog
ADCHS  = 0x0002;                                // SAMP bit = 0 ends sampling ...
ADCSSL = 0;                                     // and starts converting
ADCON3 = 0x0002;                                // Connect RB2/AN2 as CH0 input ..
ADCON2 = 0;                                     // in this example RB2/AN2 is the input
                                                // Manual Sample, Tad = internal 2 Tcy

ADCON1bits.ADON = 1;                            // turn ADC ON
while (1)                                         // repeat continuously
{
    ADCON1bits.SAMP = 1;                          // start sampling ...
    DelayNmSec(100);                           // for 100 mS
    ADCON1bits.SAMP = 0;                          // start Converting
    while (!ADCON1bits.DONE);                   // conversion done?
    ADCValue = ADCBUF0;                         // yes then get ADC value
}
                                                // repeat

```

図 17-5 は、ASAM ビットの設定により自動サンプリングが開始され、SAMP ビットのクリアによりサンプリングが終了され変換を開始する例です。変換完了後、モジュールは自動的にサンプリング状態に戻ります。SAMP ビットは自動的にサンプルインターバルの開始時点でセットされます。ユーザーソフトウェアにより、SAMP ビットのクリアの間の時間にはサンプリング時間と同じく変換時間も含まれることを理解しながら、入力信号の十分なサンプリング時間を確保するように SAMP ビットのクリアの時間を調整する必要があります。コードの例は例 17-2 を参照してください。

図 17-5: 1 チャネル変換、自動サンプル開始、手動変換開始



例 17-2: 1 チャネル変換、自動サンプル開始、手動変換開始コード例

```

ADPCFG = 0xFF7F;           // all PORTB = Digital but RB7 = analog
ADCON1 = 0x0004;           // ASAM bit = 1 implies sampling ..
                           // starts immediately after last
                           // conversion is done
ADCHS  = 0x0007;           // Connect RB7/AN7 as CH0 input ..
                           // in this example RB7/AN7 is the input
ADCSSL = 0;
ADCON3 = 0x0002;           // Sample time manual, Tad = internal 2 Tcy
ADCON2 = 0;

ADCON1bits.ADON = 1;        // turn ADC ON
while (1)                  // repeat continuously
{
    DelayNmSec(100);        // sample for 100 mS
    ADCON1bits.SAMP = 0;     // start Converting
    while (!ADCON1bits.DONE); // conversion done?
    ADCValue = ADCBUF0;      // yes then get ADC value
}
                           // repeat

```

### 17.12.2 クロック変換トリガー

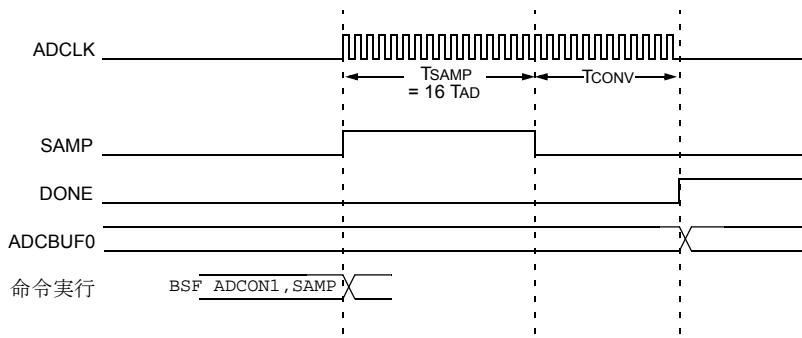
$\text{SSRC}_{<2:0>} = 111$  の時、変換トリガーは A/D クロック制御下に置かれます。SAMC ビット ( $\text{ADCON3}_{<12:8>}$ ) により、サンプリングの開始から変換の開始までのクロックサイクル  $T_{\text{AD}}$  の数が選択されます。このトリガーオプションは、複数チャネルで変換レートが最速になります。サンプリングの開始後、モジュールにより SAMC ビットで指定されたクロック数だけカウントされます。

等式 17-2: クロック変換トリガー時間

$$T_{\text{SMP}} = \text{SAMC}_{<4:0>} * T_{\text{AD}}$$

1 つの S/H チャネルのみ、または同時サンプリングを使用する場合、1 クロックサイクル以上に SAMC を設定する必要があります。複数の S/H チャネルと順次サンプリングを使用する場合、ゼロクロックサイクルで SAMC を設定すると、最も速い変換速度が得られます。コード例については、例 17-3 を参照してください。

図 17-6: 1 チャネル変換、手動サンプル開始、変換に基づく TAD 開始



例 17-3: 1 チャネル変換、手動サンプル開始、  
TAD ベースの変換開始コード

```

ADPCFG = 0xFFFF;           // all PORTB = Digital; RB12 = analog
ADCON1 = 0x00E0;           // SSRC bit = 111 implies internal
                           // counter ends sampling and starts
                           // converting.
ADCHS  = 0x000C;           // Connect RB12/AN12 as CH0 input ..
                           // in this example RB12/AN12 is the input
ADCSSL = 0;                // Sample time = 31Tad, Tad = internal 2 Tcy
ADCON3 = 0x1F02;
ADCON2 = 0;

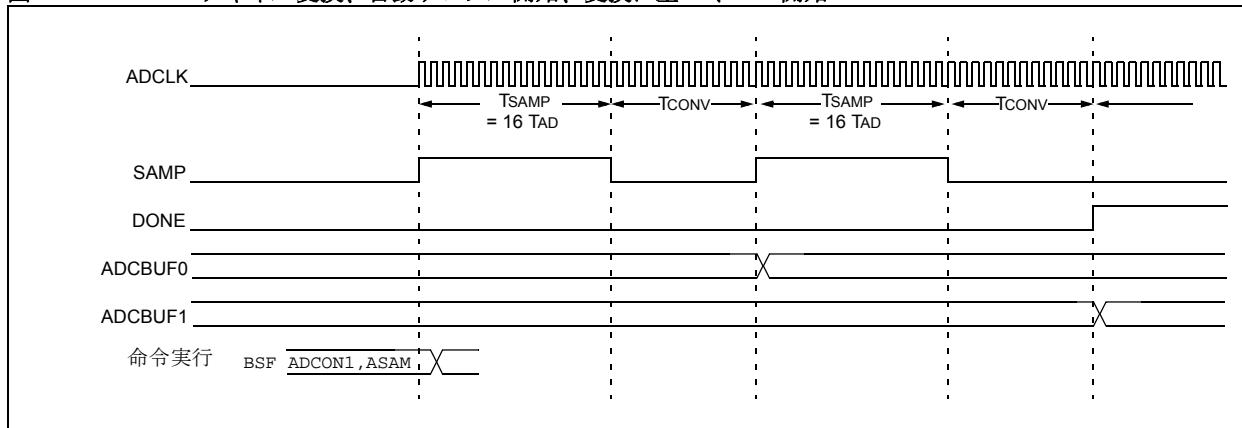
ADCON1bits.ADON = 1;       // turn ADC ON
while (1)                  // repeat continuously
{
    ADCON1bits.SAMP = 1;     // start sampling then ...
                           // after 31Tad go to conversion
    while (!ADCON1bits.DONE); // conversion done?
    ADCValue = ADCBUF0;      // yes then get ADC value
}                           // repeat                                // repeat

```

## 17.12.2.1 自動サンプル変換シーケンス

図 17-7 に示されるように、自動変換の変換トリガーモード (SSRC = 111) を自動サンプル開始モード (ASAM = 1) と組み合わせて使用することにより、A/D モードがユーザーまたは他のデバイスリソースによる割り込みが全くないサンプル / 変換シーケンスにできます。この「クロックされた」モードにより、モジュール初期化後の継続データ収集が可能になります。コード例については、例 17-4 を参照してください。

図 17-7: 1 チャネル変換、自動サンプル開始、変換に基づく TAD 開始

例 17-4: 1 チャネル変換、自動サンプル開始、  
TAD ベースの変換開始コード

```

ADPCFG = 0xFFFF;           // all PORTB = Digital; RB2 = analog
ADCON1 = 0x00E0;           // SSRC bit = 111 implies internal
                           // counter ends sampling and starts
                           // converting.
ADCHS  = 0x0002;           // Connect RB2/AN2 as CH0 input ..
                           // in this example RB2/AN2 is the input
ADCSSL = 0;                // Sample time = 15Tad, Tad = internal Tcy/2
ADCON3 = 0x0F00;           // Interrupt after every 2 samples
                           // ADCON2 = 0x0004;

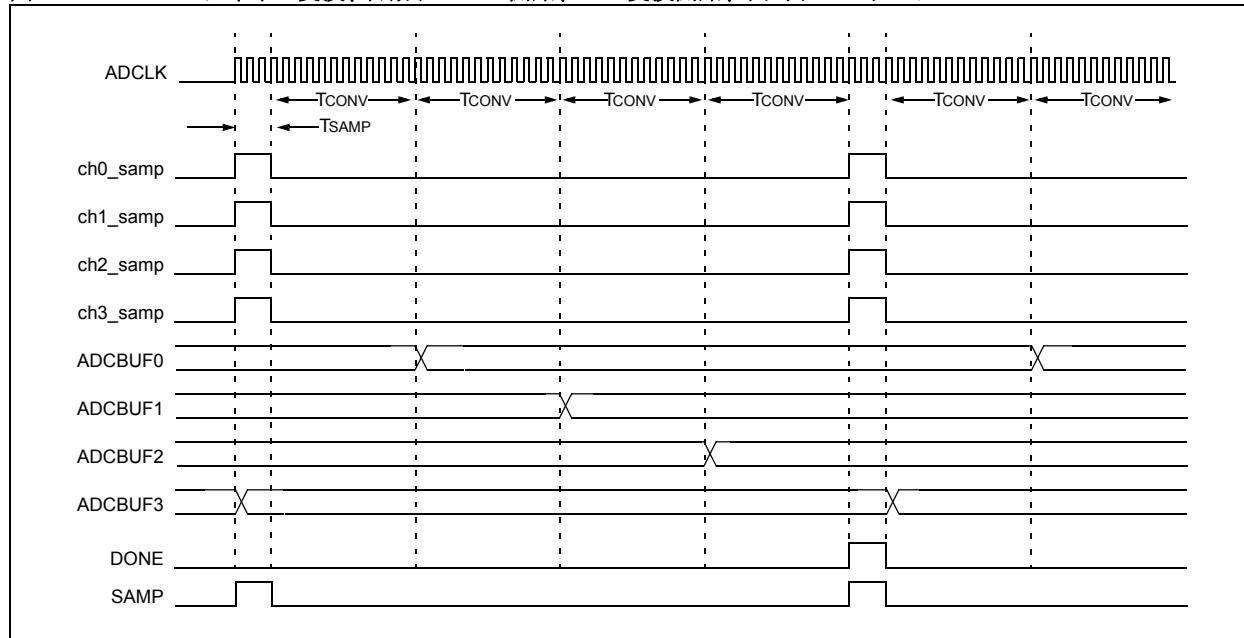
ADCON1bits.ADON = 1;        // turn ADC ON
while (1)                  // repeat continuously
{
    ADCValue = 0;            // clear value
    ADC16Ptr = &ADCBUF0;      // initialize ADCBUF pointer
    IFS0bits.ADIF = 0;        // clear ADC interrupt flag
    ADCON1bits.ASAM = 1;      // auto start sampling
                           // for 31Tad then go to conversion
    while (!IFS0bits.ADIF);   // conversion done?
    ADCON1bits.ASAM = 0;      // yes then stop sample/convert
    for (count = 0; count < 2; count++) // average the 2 ADC value
        ADCValue = ADCValue + *ADC16Ptr++;
    ADCValue = ADCValue >> 1;
}
                           // repeat

```

## 17.12.2.2 複数チャネルの同時サンプリング

図 17-8 に示されるように、同時サンプリングを使用する場合、SAMC の値によりサンプリング継続時間が指定されます。例では SAMC がサンプル時間 3TAD を指定しています。自動サンプル開始がアクティブになっているため、最後の変換が終了した後に全チャネルでサンプリングを開始し、3 A/D クロックの間続きます。コード例については、例 17-5 を参照してください。

図 17-8: 4 チャネル変換、自動サンプル開始、TAD 変換開始、同時サンプリング



17

10 ビット A/D  
コンバータ例 17-5: 4 チャネル変換、自動サンプル開始、  
TAD 変換開始、同時サンプリングコード

```

ADPCFG = 0xFF78;           // RBO,RB1,RB2 & RB7 = analog
ADCON1 = 0x00EC;           // SIMSAM bit = 1 implies ...
                           // simultaneous sampling
                           // ASAM = 1 for auto sample after convert
                           // SSRC = 111 for 3Tad sample time
                           // Connect AN7 as CH0 input

ADCHS  = 0x0007;           // 

ADCSSL = 0;                // Auto Sampling 3 Tad, Tad = internal 2 Tcy
ADCON3 = 0x0302;           // CHPS = 1x implies simultaneous ...
ADCON2 = 0x030C;           // sample CH0 to CH3
                           // SMPI = 0011 for interrupt after 4 converts

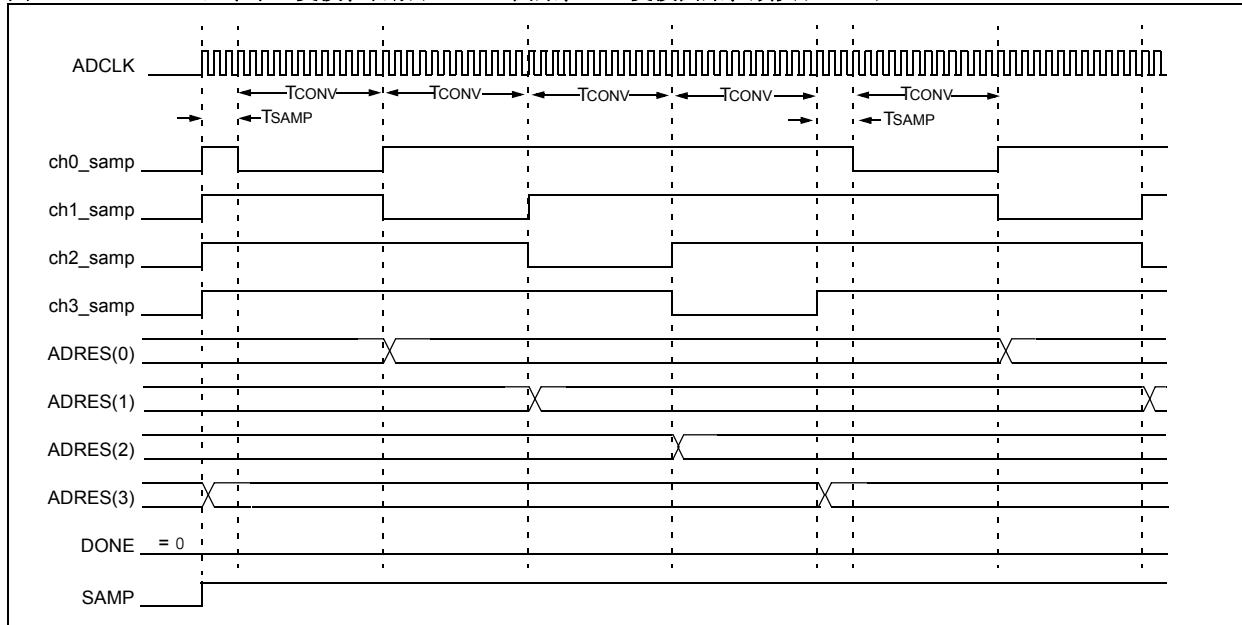
ADCON1bits.ADON = 1;       // turn ADC ON
while (1)                  // repeat continuously
{
    ADC16Ptr = &ADCBUF0;   // initialize ADCBUF pointer
    OutDataPtr = &OutData[0]; // point to first TXbuffer value
    IFS0bits.ADIF = 0;     // clear interrupt
    while (IFS0bits.ADIF); // conversion done?
    for (count = 0; count < 4; count++) // save the ADC values
    {
        ADCValue = *ADC16Ptr++;
        LoadADC(ADCValue);
    }
}
                           // repeat

```

## 17.12.2.3 複数チャネルの順次サンプリング

図 17-9 に示されるように、順次サンプリングを使用する場合、それぞれの変換時間の前にサンプル時間があります。例では、各チャネルの継続時間として 3 TAD クロックが追加されています。

図 17-9: 4 チャネル変換、自動サンプル開始、TAD 変換開始、順次サンプリング



## 17.12.2.4 クロックされた変換トリガーおよび自動サンプリングを使用するサンプル時間の留意事項

異なるサンプル・変換シーケンスにより、アナログ信号を獲得する S/H チャネルのために異なるサンプリング時間が提供されます。ユーザーはサンプリング時間がセクション 17.16 「A/D サンプリング要件」で概要を説明しているサンプリング要件以上であることを確認する必要があります。

モジュールが自動サンプリングおよびクロック変換トリガー使用に設定された場合、サンプリングインターバルは、SAMC ビットに設定されたサンプルインターバルにより決定されます。

SIMSAM ビットが同時サンプリング指定している場合、またはアクティブなチャネルが 1 つのみの場合、サンプル時間は SAMC ビットで指定される期間です。

等式 17-3: 利用可能なサンプリング時間、同時サンプリング

$$TSMP = SAMC<4:0> * TAD$$

SIMSAM ビットが順次サンプリングを指定する場合、すべてのチャネルの変換に費やされる合計インターバルは、チャネル数にサンプリング時間と変換時間の合計を掛けた数値になります。個々のチャネルのサンプル時間は、合計インターバルからそのチャネルの変換時間を引いたものです。

等式 17-4: 利用可能なサンプリング時間、同時サンプリング

$$\begin{aligned} TSEQ &= \text{Channels per Sample (CH/S)} * \\ &\quad ((SAMC<4:0> * TAD) + \text{Conversion Time (TCONV)}) \\ TSMP &= (TSEQ - TCONV) \end{aligned}$$

注 1: CHPS<1:0> ビットで指定される CH/S。  
2: TSEQ はサンプル / 変換シーケンスの合計時間。

### 17.12.3 イベントトリガー変換開始

サンプリングの終了と A/D 変換開始は、他のタイムイベントと同期することが望ましいことがあります。A/D モジュールは、変換トリガーイベントとして、3 つのソースのいずれかを使用します。

#### 17.12.3.1 外部 INT ピントリガー

$SSRC<2:0> = 001$  の時、A/D 変換は INT0 ピンのアクティブな遷移でトリガれます。INT0 ピンは、立ち上がり入力または立ち下り入力のいずれかがアクティブとして設定されます。

#### 17.12.3.2 汎用タイマー比較一致トリガー

A/D は  $SSRC<2:0> = 010$  設定によりこのトリガーモードに構成されます。32 ビットタイマー TMR3/TMR2 と 32 ビットの結合周期レジスタ PR3/PR2 との間で比較一致が発生すると、特別な ADC トリガーイベント信号がタイマー 3 により生成されます。TMR5/TMR4 タイマーペアにはこの特徴はありません。詳しくは第 12 章 . 「タイマー」を参照してください。

#### 17.12.3.3 モーター制御 PWM トリガー

PWM モジュールには A/D 変換が PWM 時間ベースに同期化ようにするイベントトリガーがあります。 $SSRC<2:0> = 011$  の時、A/D サンプリングおよび変換トリガを PWM 周期内の任意の位置で出力することができます。ユーザーは特別イベントトリガーにより、A/D 変換結果獲得時の時間とデューティアップデート時の間の遅延時間を最小にすることができます。詳細は第 15 章 . 「モーター制御 PWM」を参照してください。

#### 17.12.3.4 内部イベントまたは外部イベントとの A/D 変換動作の同期化

外部イベントトリガーパルスによりサンプリングが終了し、変換 ( $SSRC = 001, 10, 011$ ) が開始するモードを自動サンプリング (ASAM = 1) と併用することで、A/D はサンプル変換イベントをトリガーパルスソースに同期させることができます。例えば、 $SSRC = 010$ 、ASAM = 1 の図 17-11 では、A/D は常にサンプリングを終了させて、タイマー比較トリガーイベントとともに変換を開始します。A/D のサンプル変換レートは、タイマー比較イベント周期に依存することになります。コード例については、例 17-6 を参照してください。

図 17-10: 1 チャネル変換、手動サンプル開始、変換トリガーに基づく変換開始

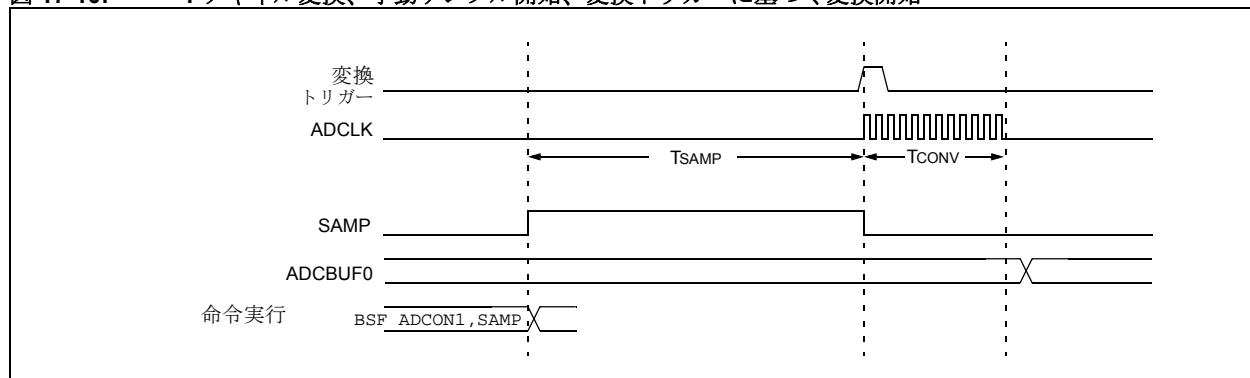
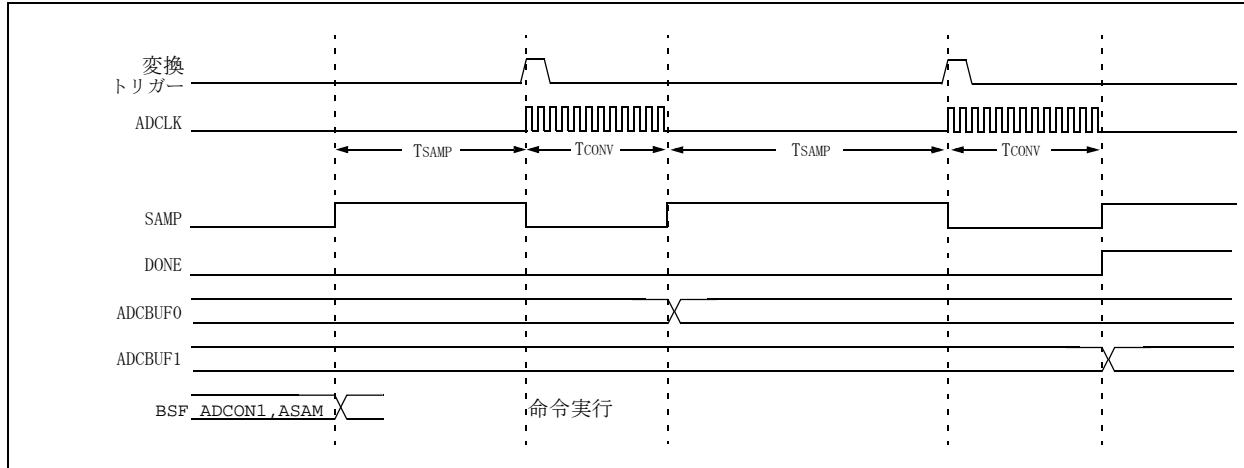


図 17-11: 1 チャネル変換、自動サンプル開始、変換トリガーに基づく変換開始



例 17-6: 1 チャネル変換、自動サンプル開始、変換トリガーに基づく変換開始コード

```

ADPCFG = 0xFFFF;           // all PORTB = Digital; RB2 analog
ADCON1 = 0x0040;           // SSRC bit = 010 implies GP TMR3
                           // compare ends sampling and starts
                           // converting.
ADCHS  = 0x0002;           // Connect RB2/AN2 as CH0 input ..
                           // in this example RB2/AN2 is the input
ADCSSL = 0;                // Sample time is TMR3, Tad = internal Tcy/2
ADCON3 = 0x0000;           // Interrupt after 2 conversions
                           // Internal Tcy/2 = 125 mSecs
ADCON2 = 0x0004;

// set TMR3 to time out every 125 mSecs
TMR3  = 0x0000;
PR3   = 0x3FFF;
T3CON = 0x8010;

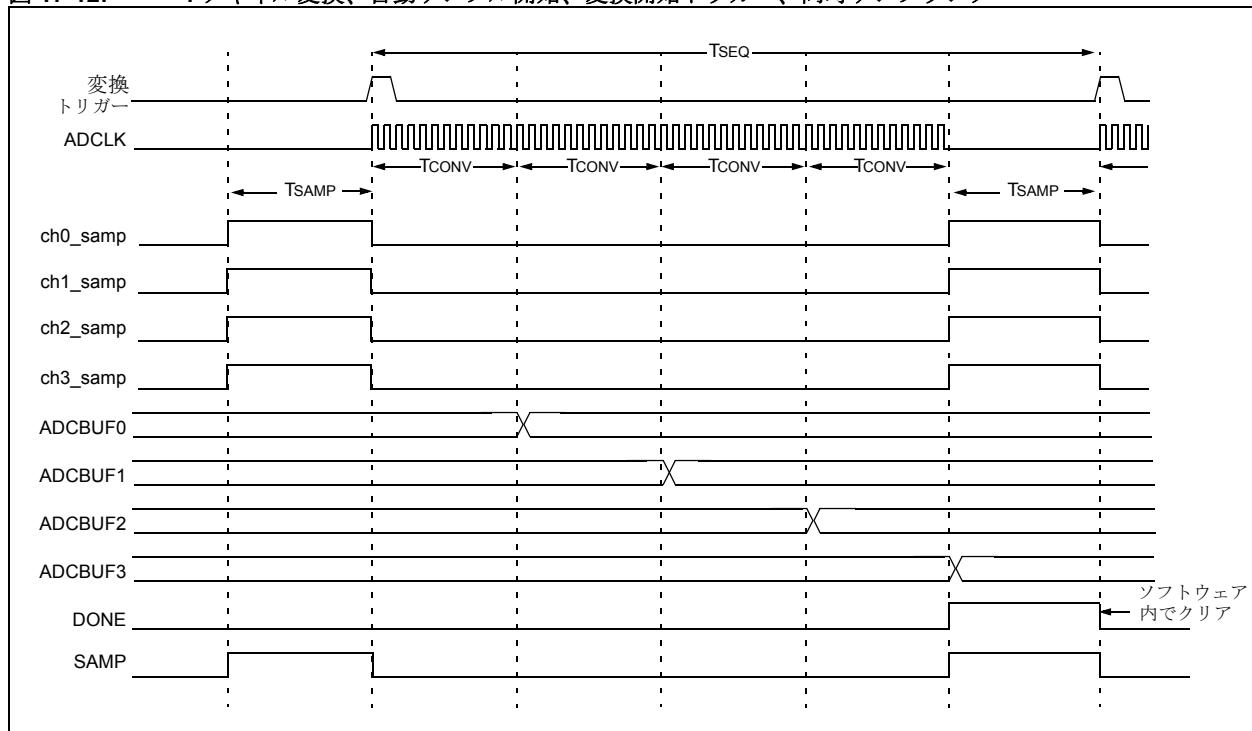
ADCON1bits.ADON = 1;        // turn ADC ON
ADCON1bits.ASAM = 1;        // start auto sampling every 125 mSecs
while (1)                  // repeat continuously
{
    while (!IFS0bits.ADIF);  // conversion done?
    ADCValue = ADCBUF0;      // yes then get first ADC value
    IFS0bits.ADIF = 0;        // clear ADIF
}
                           // repeat

```

## 17.12.3.5 複数チャネルの同時サンプリング

図17-12に示されるように、同時サンプリングを使用する場合、ASAMビットの設定後、または最後の変換の終了時にすべてのチャネルでサンプリングが開始されます。変換トリガーが起ると、サンプリングが停止し、変換が開始されます。

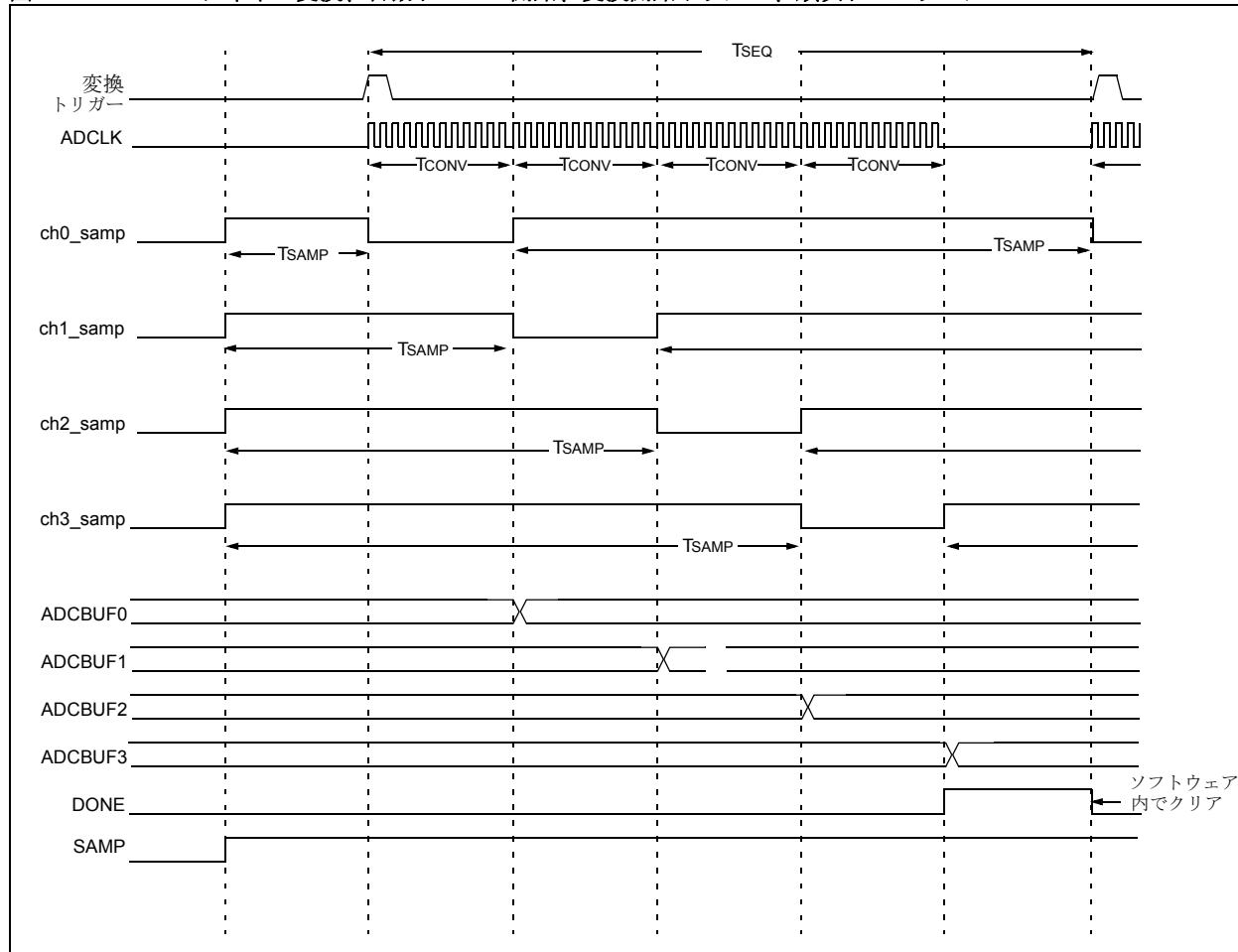
図17-12: 4チャネル変換、自動サンプル開始、変換開始トリガー、同時サンプリング



## 17.12.3.6 複数チャネルの順次サンプリング

図 17-13 に示されるように、順次サンプリングを使用する場合、特定のチャネルのサンプリングはそのチャネルの変換直前に停止し、変換の停止後にサンプリングが再開します。

図 17-13: 4 チャネル変換、自動サンプル開始、変換開始トリガー、順次サンプリング



## 17.12.3.7 自動サンプリング / 変換シーケンスのサンプル時間の留意事項

サンプル・変換シーケンスが異なると、S/H チャネルがアナログ信号を獲得するために利用できるサンプル時間が異なります。サンプリング時間がセクション 17.16 「A/D サンプリング要件」で概要を説明しているサンプリング要件を上回っていることを確認する必要があります。

モジュールが自動サンプリング用に設定され、外部トリガーパルスが変換トリガーとして使用されている場合、サンプリングインターバルはトリガーパルスインターバルの一部を構成します。

SIMSAM ビットが同時サンプリングを指定する場合、サンプリング時間はトリガーパルス期間から、指定された変換に必要な時間を引いた時間になります。

## 等式 17-5: 利用可能なサンプリング時間、同時サンプリング

$$\begin{aligned} \text{TSMP} &= \text{Trigger Pulse Interval (TSEQ)} - \\ &\quad \text{Channels per Sample (CH/S)} * \text{Conversion Time (TCONV)} \\ \text{TSMP} &= \text{TSEQ} - (\text{CH/S} * \text{TCONV}) \end{aligned}$$

注 1: CHPS<1:0> ビットで指定される CH/S。  
2: TSEQ はトリガーパルスインターバル時間。

SIMSAM ビットが順次サンプリングを指定する場合、サンプリング時間はトリガーパルス期間から、1 つの変換のみに必要な時間を引いた時間になります。

## 等式 17-6: 利用可能なサンプリング時間、連続サンプリング

$$\begin{aligned} \text{TSMP} &= \text{Trigger Pulse Interval (TSEQ)} - \\ &\quad \text{Conversion Time (TCONV)} \\ \text{TSMP} &= \text{TSEQ} - \text{TCONV} \end{aligned}$$

注: TSEQ はトリガーパルスインターバル時間。

## 17.13 サンプリング / 変換動作の制御

アプリケーションソフトウェアは、SAMP ビットと DONE ビットをポーリングすることで A/D 変換動作を追跡したり、あるいは変換の終了時にモジュールが CPU に割り込むようにすることができます。またアプリケーションソフトウェアは、必要に応じて、A/D 動作を中断することもできます。

### 17.13.1 サンプル / 変換ステータスのモニタリング

SAMP (ADCON1<1>) ビットおよび DONE (ADCON1<0>) ビットは、それぞれサンプリングの状態と A/D の変換状態を示します。一般に、SAMP ビットがクリアされサンプリングの終了が示される場合、DONE ビットが自動的に設定され、変換開始を示します。SAMP と DONE の両方が ‘0’ ならば、A/D は非アクティブ状態です。一部の動作モードでは、SAMP ビットでサンプリングの呼び出しおよび終了を行うこともできます。

### 17.13.2 A/D 割り込みの生成

SMPI<3:0> ビットは、割り込みの生成を制御します。割り込みはサンプリングの開始の後に、ある回数のサンプル・変換シーケンス後で起こり、以降同じ回数ごとのサンプルでも発生します。割り込みは、変換やバッファメモリ内のデータ書き込みではなく、サンプルを基準に指定されます。

SIMSAM ビットが順次サンプリングを指定する場合、CHPS ビットにより指定されるチャネル数に関係なく、モジュールは各変換とバッファ内のデータ書き込みごとにサンプリングを 1 回行います。したがって、SMPI ビットにより特定される値は、16 を最大数とするバッファのデータ書き込みの数に等しくなります。

SIMSAM ビットが同時サンプリングを指定する場合、バッファ内のデータサンプルの数は CHPS ビットに関連付けられます。チャネル / サンプル時間にサンプル数を掛けた数が、バッファ内のデータ書き込み数になります。オーバーランによりバッファ内のデータが損失するのを防ぐために、必要なバッファサイズをサンプル当たりのチャネル数で割った数に SMPI ビットを設定する必要があります。

A/D 割り込みの無効化は SMPI ビットでは行われません。割り込みを無効化するためには、ADIE アナログモジュール割り込み有効化ビットをクリアします。

### 17.13.3 サンプリングの中断

手動サンプリングモード中に SAMP をクリアすることにより、サンプリングが終了しますが、SSRC = 000 ならば変換が開始される可能性もあります。

自動サンプリングモードの間に ASAM ビットをクリアしても、実行中のサンプル・変換シーケンスは終了しませんが、サンプリングはその後の変換の後に自動的に再開しません。

### 17.13.4 変換の中断

変換中に ADON ビットをクリアすることにより、現在の変換が中断されます。A/D 結果レジスタペアは、A/D 変換の部分的な終了結果では更新されません。つまり、対応している ADCBUF バッファ位置には、最後に実行された変換の値（またはバッファに最後に書き込まれた値）が引き続き格納されます。

## 17.14 変換結果をバッファに書き込む方法の指定

変換が完了すると、モジュールにより A/D 結果バッファに変換の結果が書き込まれます。このバッファは、10 ビットワード 16 個から構成される RAM アレイです。バッファには ADCBUF0...ADCBUFF と名付けられた SFR 空間内の、16 のアドレス位置によってアクセスされます。

ユーザー側のソフトウェアは、A/D 変換結果が生成されるたびに、これを読み込むこともできますが、この方法では CPU 時間が大量に消費されます。従って一般的に、コードを簡単にするために、モジュールがバッファに結果を充填し、バッファが一杯になったときに割り込みを生成するようにします。

### 17.14.1 割り込みごとの変換の数

**SMPI<3:0>** ビット (ADCON2<5:2>) は、CPU の割り込みが起こる前に発生する A/D 変換の回数を選択します。割り込みあたり 1 サンプルから 16 サンプルまで指定できます。各割り込みの後に、A/D コンバータモジュールは常にバッファの先頭から変換結果の書き込みを開始します。例えば、**SMPI<3:0> = 0000** ならば、変換結果は常に ADCBUFO に書き込まれます。この例では、他のバッファ位置は使用されません。

### 17.14.2 バッファサイズによる制限

CHPS ビットと **SMPI** ビットを組み合わせてプログラミングする場合、BUFM ビット (ADCON2<1>) が ‘0’ の場合、割り込みあたり 16 回を超える変換を指定する組み合わせ、または BUFM ビット (ADCON2<1>) が ‘0’ の場合、割り込みあたり 8 回の変換を指定する組み合わせをプログラミングできません。BUFM ビット機能については、以下で説明します。

### 17.14.3 バッファファイルモード

BUFM ビット (ADCON2<1>) が ‘1’ の時、16 ワードの結果バッファ (ADRES) は 2 つの 8 ワードのグループに分けられます。各割り込みイベントの後に、8 ワードバッファは交代で変換結果を受け取ります。BUFM の設定後に使用される最初の 8 ワードバッファは、ADCBUF の下位アドレスに置かれます。BUFM が ‘0’ の時、すべての変換シーケンスで 16 ワードの完全なバッファが使用されます。

BUFM 機能を使用するかどうかの決定は、割り込み後にバッファ内容を移動するのに利用できる時間の長さにより異なり、これはアプリケーションにより決定されます。1 つのチャネルをサンプルし変換するのにかかる時間内に、プロセッサが素早くすべてのバッファをアンロードできる場合、BUFM ビットが ‘0’ とでき、割り込みごとに最大 16 の変換が実行されます。プロセッサは、最初のバッファ位置が上書きされる前に 1 回分のサンプリング / 変換時間だけが与えられます。

1 回のサンプルと変換の時間内にプロセッサがバッファをアンロードできない場合、BUFM ビットを ‘1’ にします。例えば、**SMPI<3:0> = 0111** ならば、8 つの変換がバッファの半分にロードされ、その後割り込みが発生します。次の 8 つの変換はバッファの残りの半分にロードされます。したがって、プロセッサは割り込みから次の割り込みまでの全ての時間を使って、バッファから 8 つの変換を確実に移動することができます。

### 17.14.4 バッファファイルステータス

変換結果バッファが BUFM 制御ビットを使って分割される場合、BUFS ステータスビット (ADCON2<7>) により、どちらの半分が A/D コンバータにより充填中であるかが示されます。BUFS = 01 ならば、A/D コンバータは ADCBUF0-ADCBUF7 を充填中であり、ユーザーソフトウェアが ADCBUF8-ADCBUFF から変換値を読み込みます。BUFS = 1 ならば、上記の逆になります、ユーザーソフトウェアは ADCBUF0-ADCBUF7 から変換値を読み込みます。

## 17.15 変換シーケンスの例

以下の構成例には、サンプリングおよびバッファ構成の異なる A/D動作が示されています。各例では、ASAM ビットの設定により自動サンプリングが開始されます。変換トリガーによりサンプリングが終了し、変換が開始されます。

### 17.15.1 例：单一チャネル複数回のサンプリングおよび変換

図 17-11 と表 17-3 では A/D の基本的構成が説明されています。この場合、1 つの A/D 入力である AN0 が 1 つのサンプル・ホールドチャネル CHO によりサンプリングされ、変換されます。結果は ADCBUF バッファに格納されます。このプロセスは、バッファが一杯になって、モジュールが割り込みを生成するまで 16 回繰り返されます。それから全プロセスが繰り返されます。

CHPS ビットは、サンプル / ホールド CHO のみがアクティブになるように指定します。ALTS をクリアすることで、MUX A 入力のみがアクティブになります。CHOSA ビットと CHONA ビットは、サンプル / ホールドチャネルへの入力として指定されます (AN0-VREF-)。他の入力選択ビットはすべて使用されません。

図 17-14: 単一チャネルの変換 16 回 / 割り込み

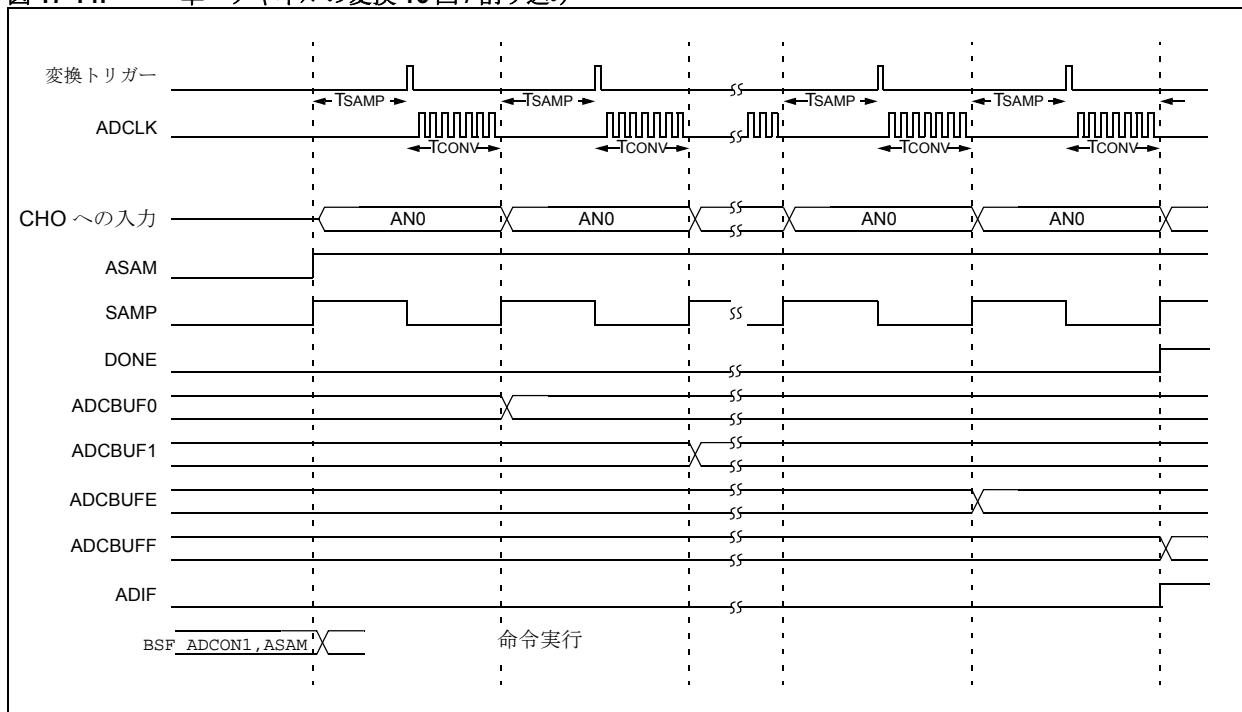


表 17-3: 単一チャネルの変換 16 回 / 割り込み

制御ビット シーケンス選択	
SMPI<2:0> = 1111	16 回目のサンプルに割り込み
CHPS<1:0> = 00	サンプルチャネル CHO
SIMSAM = n/a	单一チャネルのサンプルは非該当
BUFM = 0	シングル 16 ワード結果バッファ
ALTS = 0	常に MUX A 入力選択を使用します。
MUX A 入力選択	
CH0SA<3:0> = 0000	CHO+ 入力には ANO を選択
CH0NA = 0	CHO- 入力には VREF- を選択
CSCNA = 0	入力スキャンなし
CSSL<15:0> = n/a	スキャン入力選択未使用
CH123SA = n/a	チャネル CH1、CH2、CH3 + 未使用入力
CH123NA<1:0> = n/a	チャネル CH1、CH2、CH3 - 未使用入力
MUX B 入力選択	
CH0SB<3:0> = n/a	チャネル CHO+ 入力未使用
CH0NB = n/a	チャネル CHO- 入力未使用
CH123SB = n/a	チャネル CH1、CH2、CH3 + 未使用入力
CH123NB<1:0> = n/a	チャネル CH1、CH2、CH3 - 未使用入力

動作シーケンス	
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0x0
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0x1
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0x2
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0x3
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0x4
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0x5
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0x6
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0x7
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0x8
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0x9
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0xA
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0xB
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0xC
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0xD
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0xE
サンプル MUX A 入力 : AN0 -> CH0	変換 CH0、書き込みバッファ 0xF
	割り込み
	繰り返し

バッファ  
アドレス  
ADCBUF0  
ADCBUF1  
ADCBUF2  
ADCBUF3  
ADCBUF4  
ADCBUF5  
ADCBUF6  
ADCBUF7  
ADCBUF8  
ADCBUF9  
ADCBUFA  
ADCBUFB  
ADCBUFC  
ADCBUD  
ADCBUE  
ADCBUFF

バッファ @  
1回目割り込み  
AN0 サンプル 1  
AN0 サンプル 2  
AN0 サンプル 3  
AN0 サンプル 4  
AN0 サンプル 5  
AN0 サンプル 6  
AN0 サンプル 7  
AN0 サンプル 8  
AN0 サンプル 9  
AN0 サンプル 10  
AN0 サンプル 11  
AN0 サンプル 12  
AN0 サンプル 13  
AN0 サンプル 14  
AN0 サンプル 15  
AN0 サンプル 16

バッファ @  
2回目割り込み  
AN0 サンプル 17  
AN0 サンプル 18  
AN0 サンプル 19  
AN0 サンプル 20  
AN0 サンプル 21  
AN0 サンプル 22  
AN0 サンプル 23  
AN0 サンプル 24  
AN0 サンプル 25  
AN0 サンプル 26  
AN0 サンプル 27  
AN0 サンプル 28  
AN0 サンプル 29  
AN0 サンプル 30  
AN0 サンプル 31  
AN0 サンプル 32

• • •

### 17.15.2 例：全アナログ入力を通じてのスキャン中の A/D 変換の例

図 17-15 と表 17-4 に、すべての利用可能なアナログ入力チャネルが、1 つのサンプル / ホールドチャネル CHO でサンプリングされ、変換される典型的なセットアップを示しています。設定された CSCNA ビットは、CHO 正入力への A/D 入力のスキャンを指定します。他の条件はサブセクション 17.15.1 とほぼ同じです。

最初に、ANO 入力が CHO によりサンプリングされ変換されます。結果は ADCBUF バッファに格納されます。次に AN1 入力がサンプリングされ変換されます。この入力のスキャンのプロセスは、バッファが一杯になってモジュールが割り込みを生成するまで 16 回繰り返されます。次に全プロセスが繰り返されます。

図 17-15: 16 回の入力におけるスキャン / 割り込み

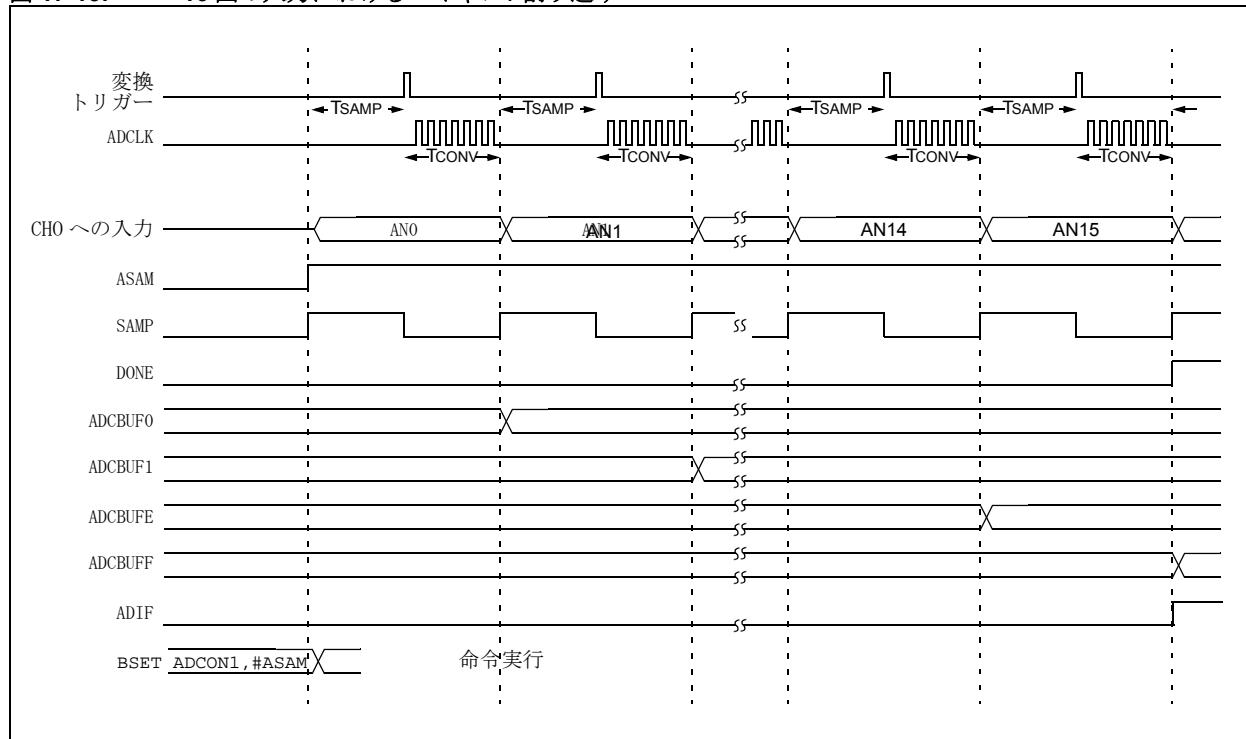


表 17-4: 16 回の入力におけるスキャン / 割り込み

制御ビット シーケンス選択	動作シーケンス
SMPI<2:0> = 1111 16 回目のサンプルで割り込み	サンプル MUX A 入力 : AN0 -> CH0 変換 CH0、書き込みバッファ 0x0
CHPS<1:0> = 00 サンプルチャネル CH0	サンプル MUX A 入力 : AN1 -> CH0 変換 CH0、書き込みバッファ 0x1
SIMSAM = n/a 单一チャネルサンプルは適用なし	サンプル MUX A 入力 : AN2 -> CH0 変換 CH0、書き込みバッファ 0x2
BUFM = 0 单一 16 ワード結果バッファ	サンプル MUX A 入力 : AN3 -> CH0 変換 CH0、書き込みバッファ 0x3
ALTS = 0 常に MUX A 入力選択を使用	サンプル MUX A 入力 : AN4 -> CH0 変換 CH0、書き込みバッファ 0x4
<b>MUX A 入力選択</b>	サンプル MUX A 入力 : AN5 -> CH0 変換 CH0、書き込みバッファ 0x5
CH0SA<3:0> = n/a CSCNA で上書き	サンプル MUX A 入力 : AN6 -> CH0 変換 CH0、書き込みバッファ 0x6
CH0NA = 0 CHO 入力に VREF- 選択	サンプル MUX A 入力 : AN7 -> CH0 変換 CH0、書き込みバッファ 0x7
CSCNA = 1 スキャン CH0+ 入力	サンプル MUX A 入力 : AN8 -> CH0 変換 CH0、書き込みバッファ 0x8
CSSL<15:0> = 1111 1111 1111 1111 全スキャン入力選択	サンプル MUX A 入力 : AN9 -> CH0 変換 CH0、書き込みバッファ 0x9
CH123SA = n/a チャネル CH1、CH2、CH3 + 未使用入力	サンプル MUX A 入力 : AN10 -> CH0 変換 CH0、書き込みバッファ 0xA
CH123NA<1:0> = n/a チャネル CH1、CH2、CH3 - 未使用入力	サンプル MUX A 入力 : AN11 -> CH0 変換 CH0、書き込みバッファ 0xB
<b>MUX B 入力選択</b>	サンプル MUX A 入力 : AN12 -> CH0 変換 CH0、書き込みバッファ 0xC
CH0SB<3:0> = n/a チャネル CH0+ 未使用入力	サンプル MUX A 入力 : AN13 -> CH0 変換 CH0、書き込みバッファ 0xD
CH0NB = n/a チャネル CH0- 未使用入力	サンプル MUX A 入力 : AN14 -> CH0 変換 CH0、書き込みバッファ 0xE
CH123SB = n/a チャネル CH1、CH2、CH3 + 未使用入力	サンプル MUX A 入力 : AN15 -> CH0 変換 CH0、書き込みバッファ 0xF
CH123NB<1:0> = n/a チャネル CH1、CH2、CH3 - 未使用入力	割り込み 繰り返し

バッファ  
アドレスADCBUF0  
ADCBUF1  
ADCBUF2  
ADCBUF3  
ADCBUF4  
ADCBUF5  
ADCBUF6  
ADCBUF7  
ADCBUF8  
ADCBUF9  
ADCBUFA  
ADCBUFB  
ADCBUFC  
ADCBUDF  
ADCBUFE  
ADCBUFFバッファ @  
1回目割り込み

AN0 サンプル 1
AN1 サンプル 2
AN2 サンプル 3
AN3 サンプル 4
AN4 サンプル 5
AN5 サンプル 6
AN6 サンプル 7
AN7 サンプル 8
AN8 サンプル 9
AN9 サンプル 10
AN10 サンプル 11
AN11 サンプル 12
AN12 サンプル 13
AN13 サンプル 14
AN14 サンプル 15
AN15 サンプル 16

バッファ @  
2回目割り込み

AN0 サンプル 17
AN1 サンプル 18
AN2 サンプル 19
AN3 サンプル 20
AN4 サンプル 21
AN5 サンプル 22
AN6 サンプル 23
AN7 サンプル 24
AN8 サンプル 25
AN9 サンプル 26
AN10 サンプル 27
AN11 サンプル 28
AN12 サンプル 29
AN13 サンプル 30
AN14 サンプル 31
AN15 サンプル 32

### 17.15.3 例：他の 4 つの入力スキャン中に 3 つの入力を頻繁にサンプリングする

図 17-16 および表 17-5 は A/D コンバータがサンプル / ホールドチャネル CH1、CH2、CH3 を使用して 3 つの入力を頻繁にサンプリングするよう設定されていることを示しています。また、サンプル / ホールドチャネル CH0 を使用した他の 4 つの入力のスキャンの頻度はサンプリングの頻度より低くなっています。この例では、MUX A 入力が使用されており、4 つのチャネル全てが同時にサンプリングされます。CHO では 4 つの異なる入力 (AN4、AN5、AN6、AN7) がスキャンされ、ANO、AN1、AN2 がそれぞれ CH1、CH2、CH3 の入力になります。そのため、16 のサンプルの全てのセット毎に、ANO、AN1、AN2 は 4 回、AN4、AN5、AN6、AN7 は 1 回のみサンプリングされます。

図 17-16: 3 つの入力を 4 回変換、4 つの入力を 1 回変換 / 割り込み

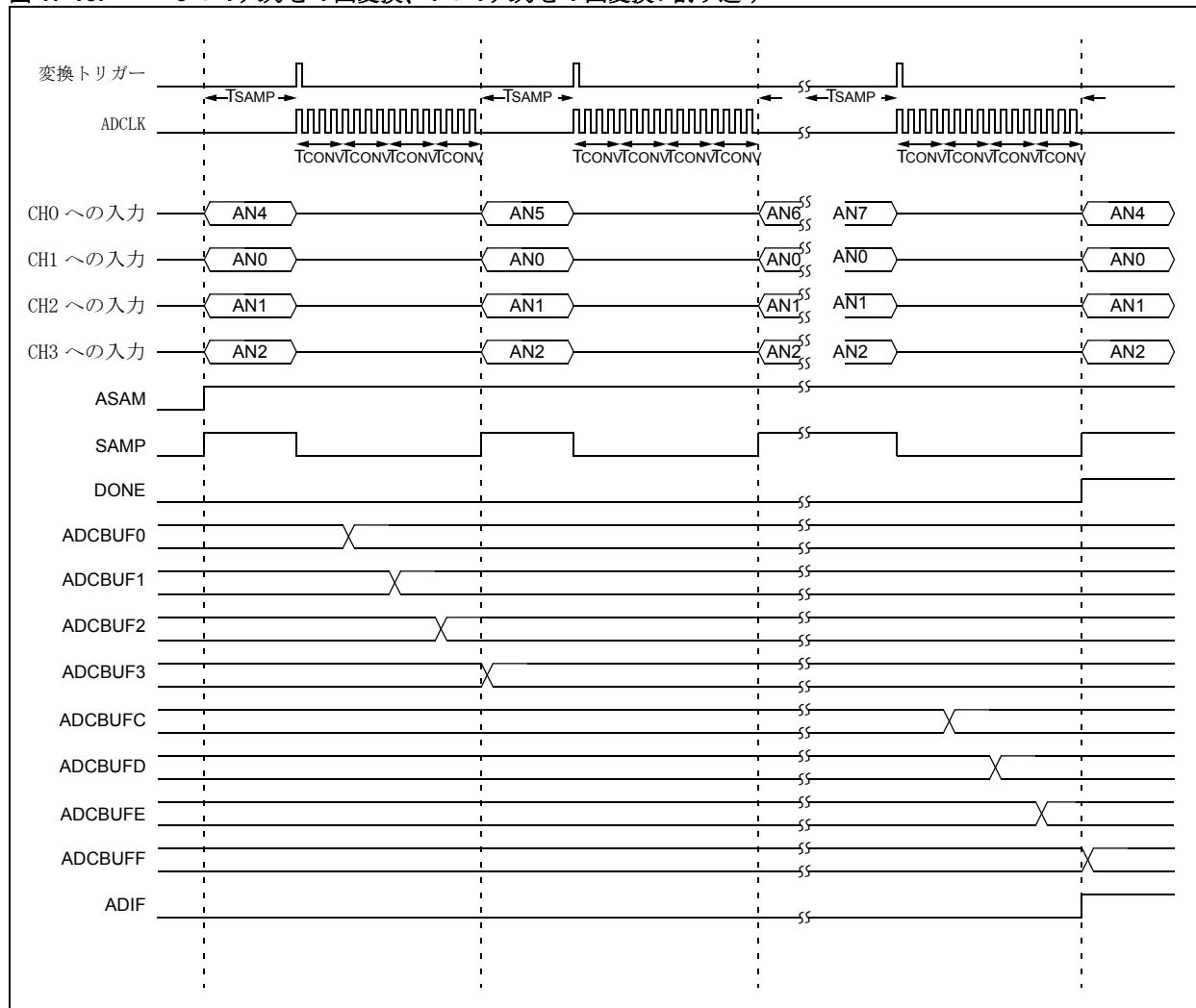


表 17-5: 3 つの入力を 4 回変換、4 つの入力を 1 回変換 / 割り込み

制御ビット シーケンス選択	動作シーケンス
SMPI<2:0> = 1111 16 回目のサンプルで割り込み	サンプル MUX A 入力: AN4 -> CH0、AN0 -> CH1、AN1 -> CH2、AN2 -> CH3 変換 CH0、書き込みバッファ 0x0 変換 CH1、書き込みバッファ 0x1 変換 CH2、書き込みバッファ 0x2 変換 CH3、書き込みバッファ 0x3
CHPS<1:0> = 1x チャネル CHO、CH1、CH2、CH3 をサンプリング	
SIMSAM = 1 全てのチャネルを同時にサンプリング	
BUFM = 0 单一 16 ワード結果バッファ	サンプル MUX A 入力: AN5 -> CH0、AN0 -> CH1、AN1 -> CH2、AN2 -> CH3 変換 CH0、書き込みバッファ 0x4 変換 CH1、書き込みバッファ 0x5 変換 CH2、書き込みバッファ 0x6 変換 CH3、書き込みバッファ 0x7
ALTS = 0 常に MUX A 入力選択を使用	
MUX A 入力選択	
CH0SA<3:0> = n/a CSCNA で上書き	サンプル MUX A 入力: AN6 -> CH0、AN0 -> CH1、AN1 -> CH2、AN2 -> CH3 変換 CH0、書き込みバッファ 0x8 変換 CH1、書き込みバッファ 0x9 変換 CH2、書き込みバッファ 0xA 変換 CH3、書き込みバッファ 0xB
CH0NA = 0 CHO 入力に VREF- 選択	
CSCNA = 1 スキャン CH0+ 入力	
CSSL<15:0> = 0000 0000 1111 0000 AN4、AN5、AN6、AN7 をスキャン	サンプル MUX A 入力: AN7 -> CH0、AN0 -> CH1、AN1 -> CH2、AN2 -> CH3 変換 CH0、書き込みバッファ 0xC 変換 CH1、書き込みバッファ 0xD 変換 CH2、書き込みバッファ 0xE 変換 CH3、書き込みバッファ 0xF
CH123SA = 0 CH1+ = AN0、CH2+ = AN1、CH3+ = AN2	割り込み
CH123NA<1:0> = 0x CH1-、CH2-、CH3- = VREF-	繰り返し
MUX B 入力選択	
CH0SB<3:0> = n/a チャネル CH0+ 未使用入力	
CH0NB = n/a チャネル CH0- 未使用入力	
CH123SB = n/a チャネル CH1、CH2、CH3 + 未使用入力	
CH123NB<1:0> = n/a チャネル CH1、CH2、CH3 - 未使用入力	

バッファ  
アドレス  
ADCBUF0  
ADCBUF1  
ADCBUF2  
ADCBUF3  
ADCBUF4  
ADCBUF5  
ADCBUF6  
ADCBUF7  
ADCBUF8  
ADCBUF9  
ADCBUFA  
ADCBUFB  
ADCBUFC  
ADCBUDF  
ADCBUFE  
ADCBUFF

バッファ @  
1 番目の割り込み  
AN4 サンプル 1  
AN0 サンプル 1  
AN1 サンプル 1  
AN2 サンプル 1  
AN5 サンプル 2  
AN0 サンプル 2  
AN1 サンプル 2  
AN2 サンプル 2  
AN6 サンプル 3  
AN0 サンプル 3  
AN1 サンプル 3  
AN2 サンプル 3  
AN7 サンプル 4  
AN0 サンプル 4  
AN1 サンプル 4  
AN2 サンプル 4

バッファ @  
2 番目の割り込み  
AN4 サンプル 5  
AN0 サンプル 5  
AN1 サンプル 5  
AN2 サンプル 5  
AN5 サンプル 6  
AN0 サンプル 6  
AN1 サンプル 6  
AN2 サンプル 6  
AN6 サンプル 7  
AN0 サンプル 7  
AN1 サンプル 7  
AN2 サンプル 7  
AN7 サンプル 8  
AN0 サンプル 8  
AN1 サンプル 8  
AN2 サンプル 8

## 17.15.4 例：二重 8 ワードバッファの使用

図 17-17 および表 17-6 は二重 8 ワードバッファを使った交互バッファ充填を示しています。BUFM ビットを設定すると 8 ワードバッファが有効化されます。BUFM 設定は他の操作パラメータに影響を与えません。最初に、変換シーケンスが開始され、バッファが ADCBUFO (バッファロケーション 0x0) で充填されます。最初の割り込みが発生すると、バッファが ADCBUF8 (バッファロケーション 0x8) で充填され始めます。BUFS ステータスピットが割り込み毎に交互にセットとクリアを繰り返します。この例では、4 つのチャネル全てが同時にサンプリングされ、各サンプリング毎に割り込みが発生しています。

図 17-17: 4 つの入力を 1 回変換 / 二重 8 ワードバッファを使用して割り込み

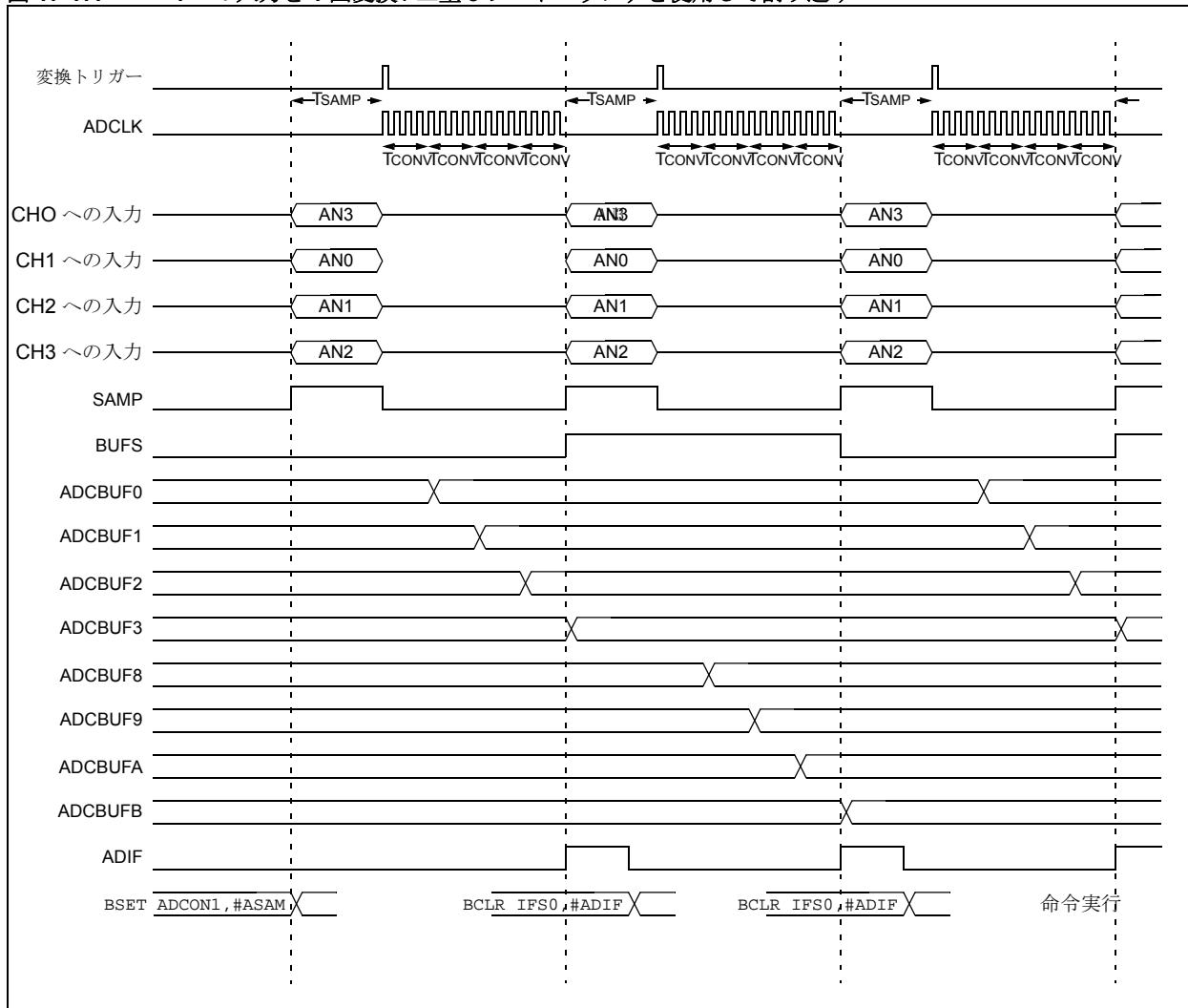


表 17-6: 4つの入力を1回変換 / 二重8ワードバッファを使用して割り込み

制御ビット	動作シーケンス
シーケンス選択	
SMPI<2:0> = 0000 各サンプルで割り込み	サンプル MUX A 入力: AN3 -> CH0、AN0 -> CH1、AN1 -> CH2、AN2 -> CH3
CHPS<1:0> = 1x チャネル CH1、CH2、CH3、CH0 をサンプリング	変換 CH0、書き込みバッファ 0x0
SIMSAM = 1 全てのチャネルを同時にサンプリング	変換 CH1、書き込みバッファ 0x1
BUFM = 1 二重 8 ワード結果バッファ	変換 CH2、書き込みバッファ 0x2
ALTS = 0 常に MUX A 入力選択を使用	変換 CH3、書き込みバッファ 0x3
<b>MUX A 入力選択</b>	割り込み；バッファ変更
CH0SA<3:0> = 0011 CH0+ 入力のために AN3 を選択	サンプル MUX A 入力: AN3 -> CH0、AN0 -> CH1、AN1 -> CH2、AN2 -> CH3
CH0NA = 0 CHO 入力に VREF- 選択	変換 CH0、書き込みバッファ 0x8
CSCNA = 0 入力スキャンなし	変換 CH1、書き込みバッファ 0x9
CSSL<15:0> = n/a スキャン入力選択未使用	変換 CH2、書き込みバッファ 0xA
CH123SA = 0 CH1+ = AN0、CH2+ = AN1、CH3+ = AN2	変換 CH3、書き込みバッファ 0xB
CH123NA<1:0> = 0x CH1-、CH2-、CH3- = VREF-	割り込み；バッファ変更
<b>MUX B 入力選択</b>	繰り返し
CH0SB<3:0> = n/a チャネル CH0+ 未使用入力	
CH0NB = n/a チャネル CH0- 未使用入力	
CH123SB = n/a チャネル CH1、CH2、CH3 + 未使用入力	
CH123NB<1:0> = n/a チャネル CH1、CH2、CH3 - 未使用入力	

17

## 10ビットA/Dコンバータ

バッファ アドレス	バッファ @ 1回目割り込み	バッファ @ 2回目割り込み
ADCBUF0	AN3 サンプル 1	
ADCBUF1	AN0 サンプル 1	
ADCBUF2	AN1 サンプル 1	
ADCBUF3	AN2 サンプル 1	
ADCBUF4		
ADCBUF5		
ADCBUF6		
ADCBUF7		
ADCBUF8		AN3 サンプル 2
ADCBUF9		AN0 サンプル 2
ADCBUFA		AN1 サンプル 2
ADCBUFB		AN2 サンプル 2
ADCBUFC		
ADCBUFD		
ADCBUFE		
ADCBUFF		

### 17.15.5 例：交互 MUX A、MUX B 入力選択の使用

図 17-18 および表 17-7 は MUX A および MUX B に割り当てられた入力の交互サンプリングを示しています。この例では、2つのチャネルで同時にサンプリングができます。ALTS ビットをセットすると、交互入力選択が有効化されます。最初のサンプリングでは、CHOSA、CHONA、CHXSA および CHXNA ビットで指定された MUX A 入力を使用します。2回目のサンプリングでは、CHOSB、CHONB、CHXSB および CHXNB ビットで指定された MUX B 入力を使用します。この例では、MUX B 入力仕様で 2つのアナログ入力 (AN3-AN9) がサンプル / ホールドの差動ソースとして使用されています。

また、二重 8 ワードバッファの使用も示されています。4 番目のサンプリング毎に割り込みが発生し、結果として、各割り込み毎にバッファに 8 ワードが充填されます。

交互入力選択を行わずに 4 つのサンプル / ホールドチャネルを使用すると、2 チャネルを交互入力で使用したこの例と同じ回数の変換が生じます。ただし、CH1、CH2、CH3 チャネルではアナログ入力の選択肢がより限られているため、この例の方法を使用した方が 4 つのチャネルを使用するよりも入力選択の幅がより広くなります。

図 17-18: 交互入力選択を使用して 2 セットの入力を 2 つ変換する

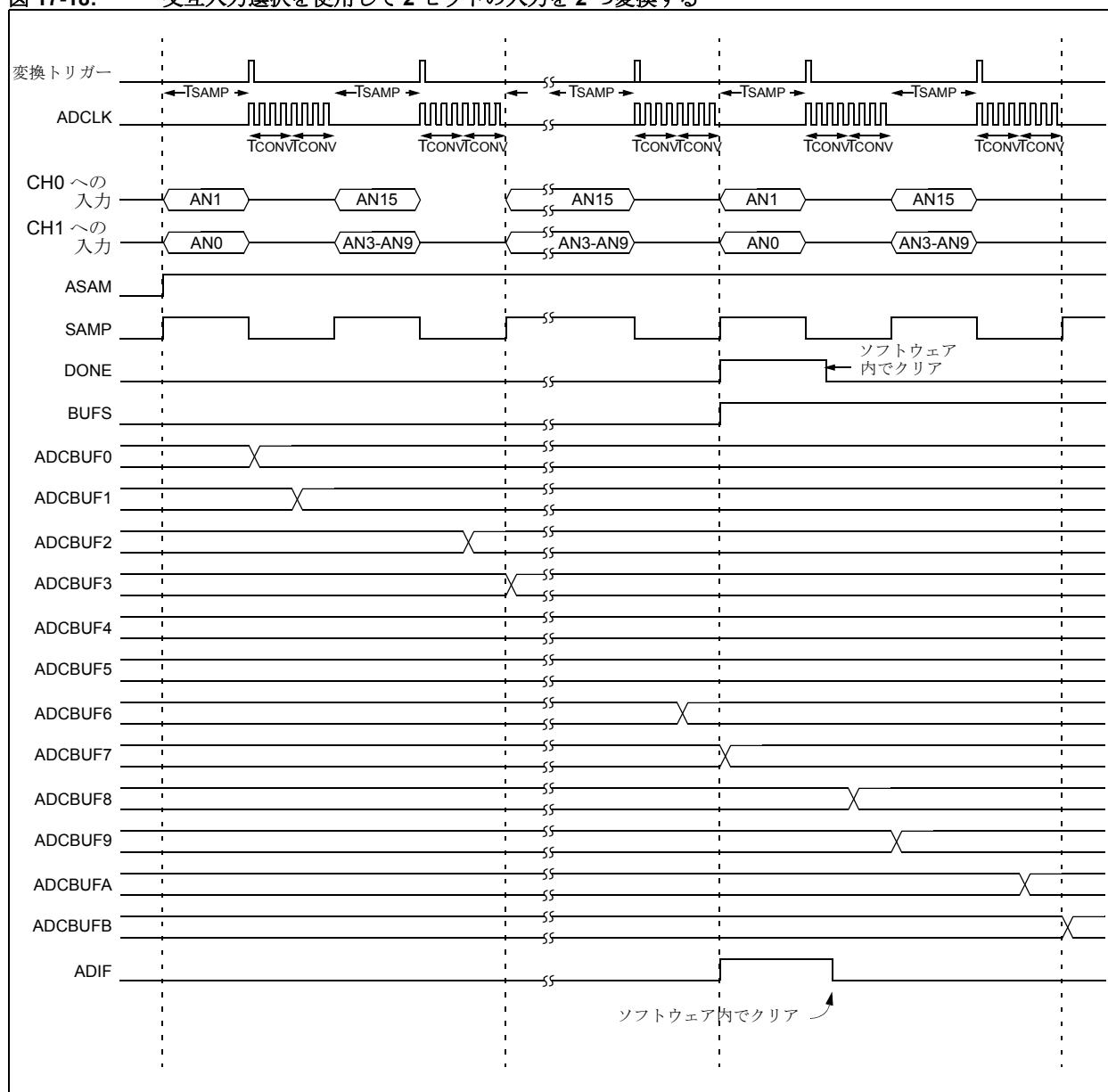


表 17-7: 交互入力選択を使用して 2 セットの入力を 2 つ変換

制御ビット シーケンス選択	
SMPI<2:0> = 0011	4回目のサンプルで割り込み
CHPS<1:0> = 01	サンプルチャネル CHO、CH1
SIMSAM = 1	全てのチャネルを同時にサンプリング
BUFM = 1	二重8ワード結果バッファ
ALTS = 1	交互MUXA/B入力選択
MUX A 入力選択	
CH0SA<3:0> = 0001	CH0+ 入力には AN1 を選択
CH0NA = 0	CH0- 入力に VREF- 選択
CSCNA = 0	入力スキヤンなし
CSSL<15:0> = n/a	スキヤン入力選択未使用
CH123SA = 0	CH1+ = AN0、CH2+ = AN1、CH3+ = AN2
CH123NA<1:0> = 0x	CH1-、CH2-、CH3- = VREF-
MUX B 入力選択	
CH0SB<3:0> = 1111	CH0+ 入力には AN15 を選択
CH0NB = 0	CH0- 入力に VREF- 択
CH123SB = 1	CH1+ = AN3、CH2+ = AN4、CH3+ = AN5
CH123NB<1:0> = 11	CH1- = AN9、CH2- = AN10、CH3- = AN11

動作シーケンス	
サンプル MUX A 入力 : AN1 -> CH0, AN0 -> CH1	変換 CH0、書き込みバッファ 0x0
	変換 CH1、書き込みバッファ 0x1
サンプル MUX B 入力 : AN15 -> CH0, (AN3-AN9) -> CH1	変換 CH0、書き込みバッファ 0x2
	変換 CH1、書き込みバッファ 0x3
サンプル MUX A 入力 : AN1 -> CH0, AN0 -> CH1	変換 CH0、書き込みバッファ 0x4
	変換 CH1、書き込みバッファ 0x5
サンプル MUX B 入力 : AN15 -> CH0, (AN3-AN9) -> CH1	変換 CH0、書き込みバッファ 0x6
	変換 CH1、書き込みバッファ 0x7
割り込み ; バッファ変更	
サンプル MUX A 入力 : AN1 -> CH0, AN0 -> CH1	変換 CH0、書き込みバッファ 0x8
	変換 CH1、書き込みバッファ 0x9
サンプル MUX B 入力 : AN15 -> CH0, (AN3-AN9) -> CH1	変換 CH0、書き込みバッファ 0xA
	変換 CH1、書き込みバッファ 0xB
サンプル MUX A 入力 : AN1 -> CH0, AN0 -> CH1	変換 CH0、書き込みバッファ 0xC
	変換 CH1、書き込みバッファ 0xD
サンプル MUX B 入力 : AN15 -> CH0, (AN3-AN9) -> CH1	変換 CH0、書き込みバッファ 0xE
	変換 CH1、書き込みバッファ 0xF
割り込み ; バッファ変更	
繰り返し	

バッファ  
アドレス

ADCBUF0  
ADCBUF1  
ADCBUF2  
ADCBUF3  
ADCBUF4  
ADCBUF5  
ADCBUF6  
ADCBUF7  
ADCBUF8  
ADCBUF9  
ADCBUFA  
ADCBUBF  
ADCBUFC  
ADCBUDF  
ADCBUFE  
ADCBUEE

バッファ @  
1回目割り込み

AN1 サンプル 1  
AN0 サンプル 1  
AN15 サンプル 2  
N3-AN9) サンプル 2  
AN1 サンプル 3  
AN0 サンプル 3  
AN15 サンプル 4  
N3-AN9) サンプル 4

バッファ @  
2回目割り込み

AN1 サンプル 5  
AN0 サンプル 5  
AN15 サンプル 6  
(AN3-AN9) サンプル 6  
AN1 サンプル 7  
AN0 サンプル 7  
AN15 サンプル 8  
(AN3-AN9) サンプル 8

• • •

### 17.15.6 例：同時サンプリングを使用して 8 つの入力をサンプリング

サブセクション 17.15.6 およびサブセクション 17.15.7 は類似したセットアップを示しています。サブセクション 17.15.6 は **SIMSAM = 1** の同時サンプリング、サブセクション 17.15.7 では、**SIMSAM = 0** での順次サンプリングを示しています。両方の例で交互入力を使用し、サンプル / ホールドへの差動入力を指定しています。

図 17-19 および表 17-8 は同時サンプリングを示しています。1 つ以上のチャネルを使用し、同時サンプリングを選択した場合、モジュールは全てのチャネルをサンプリングし必要な変換を順次実行します。この例では **ASAM** がセットされており、変換完了後にサンプリングが開始されます。

図 17-19: 同時サンプリングを使用して 8 つの入力をサンプリング

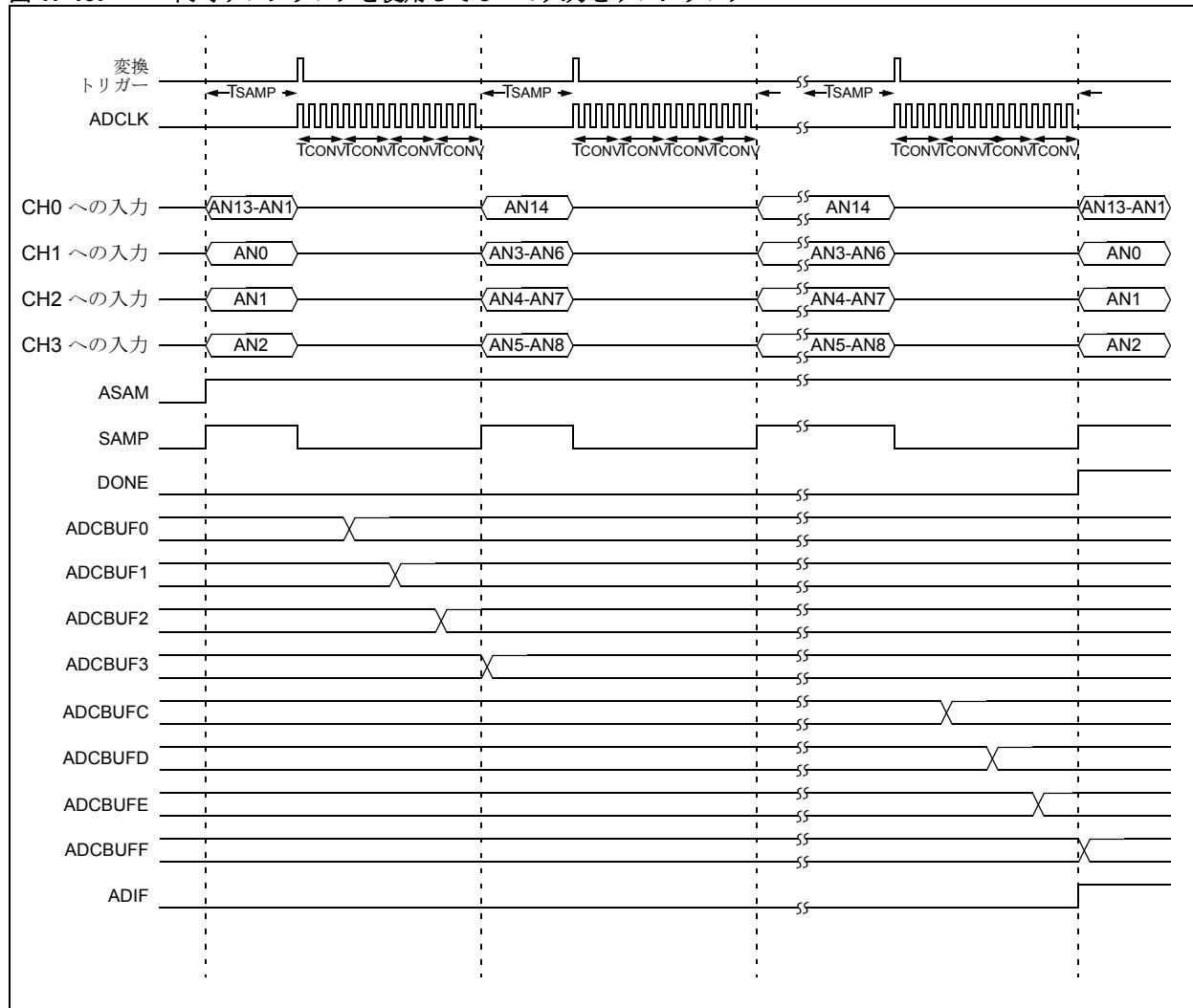


表 17-8: 同時サンプリングを使用して 8 つの入力をサンプリング

制御ビット シーケンス選択		動作シーケンス
SMPI<2:0> = 0011 4 回目のサンプルで割り込み		サンプル MUX A 入力: (AN13-AN1) -> CH0、AN0 -> CH1、AN1 -> CH2、AN2 -> CH3 変換 CH0、書き込みバッファ 0x0
CHPS<1:0> = 1x チャネル CHO、CH1、CH2、CH3 をサンプリング		変換 CH1、書き込みバッファ 0x1 変換 CH2、書き込みバッファ 0x2 変換 CH3、書き込みバッファ 0x3
SIMSAM = 1 全てのチャネルを同時にサンプリング		サンプル MUX B 入力: AN14 -> CH0、 (AN3-AN6) -> CH1、(AN4-AN7) -> CH2、(AN5-AN8) -> CH3 変換 CH0、書き込みバッファ 0x4 変換 CH1、書き込みバッファ 0x5 変換 CH2、書き込みバッファ 0x6 変換 CH3、書き込みバッファ 0x7
BUFM = 0 单一 16 ワード結果バッファ		サンプル MUX A 入力: (AN13-AN1) -> CH0、AN0 -> CH1、AN1 -> CH2、AN2 -> CH3 変換 CH0、書き込みバッファ 0x8 変換 CH1、書き込みバッファ 0x9 変換 CH2、書き込みバッファ 0xA 変換 CH3、書き込みバッファ 0xB
ALTS = 1 交互 MUX A/MUX B 入力選択		サンプル MUX B 入力: AN14 -> CH0、 (AN3-AN6) -> CH1、(AN4-AN7) -> CH2、(AN5-AN8) -> CH3 変換 CH0、書き込みバッファ 0xC 変換 CH1、書き込みバッファ 0xD 変換 CH2、書き込みバッファ 0xE 変換 CH3、書き込みバッファ 0xF
<b>MUX A 入力選択</b>		割り込み 繰り返し
CH0SA<3:0> = 1101 CH0+ 入力のために AN13 を選択		
CH0NA = 1 CH0- 入力に AN1 を選択		
CSCNA = 0 入力スキャンなし		
CSSL<15:0> = n/a スキャン入力選択未使用		
CH123SA = 0 CH1+ = AN0、CH2+ = AN1、CH3+ = AN2		
CH123NA<1:0> = 0x CH1-、CH2-、CH3- = VREF-		
<b>MUX B 入力選択</b>		
CH0SB<3:0> = 1110 CH0+ 入力には AN14 を選択		
CH0NB = 0 CH0- 入力に VREF- 選択		
CH123SB = 1 CH1+ = AN3、CH2+ = AN4、CH3+ = AN5		
CH123NB<1:0> = 10 CH1- = AN6、CH2- = AN7、CH3- = AN8		

バッファ  
アドレス  
ADCBUF0  
ADCBUF1  
ADCBUF2  
ADCBUF3  
ADCBUF4  
ADCBUF5  
ADCBUF6  
ADCBUF7  
ADCBUF8  
ADCBUF9  
ADCBUFA  
ADCBUFB  
ADCBUFC  
ADCBUD  
ADCBUE  
ADCBUFF

バッファ @  
1 回目割り込み

(AN13-AN1) サンプル 1
AN0 サンプル 1
AN1 サンプル 1
AN2 サンプル 1
AN14 サンプル 2
(AN3-AN6) サンプル 2
(AN4-AN7) サンプル 2
(AN5-AN8) サンプル 2
(AN13-AN1) サンプル 3
AN0 サンプル 3
AN1 サンプル 3
AN2 サンプルサンプル 3
AN14 サンプル 4
(AN3-AN6) サンプル 4
(AN4-AN7) サンプル 4
(AN5-AN8) サンプル 4

バッファ @  
2 回目割り込み

(AN13-AN1) サンプル 5
AN0 サンプル 5
AN1 サンプル 5
AN2 サンプル 5
AN14 サンプル 6
(AN3-AN6) サンプル 6
(AN4-AN7) サンプル 6
(AN5-AN8) サンプル 6
(AN13-AN1) サンプル 7
AN0 サンプル 7
AN1 サンプル 7
AN2 サンプル 7
AN14 サンプル 8
(AN3-AN6) サンプル 8
(AN4-AN7) サンプル 8
(AN5-AN8) サンプル 8

## 17.15.7 例：順次サンプリングを使用して 8 つの入力をサンプリングする

図 17-20 および表 17-9 は順次サンプリングを示しています。1つ以上のチャネルを使用し、順次サンプリングを選択した場合、モジュールは可能な限り早いタイミングでチャネルをサンプリングし必要な変換を順次実行します。この例では ASAM がセットされており、チャネル変換完了後にそのチャネルのサンプリングが開始されます。

ASAM がクリアされると、変換完了後にはサンプリングは開始されず、SAMP ビット選択時に実行されます。

1 つ以上のチャネルを使用する場合、順次サンプリングにはより長いサンプル時間がかかります。これは、別の変換を実行している間にそのチャネルが使われている可能性があるためです。

図 17-20: 順次サンプリングを使用して 8 つの入力をサンプリング

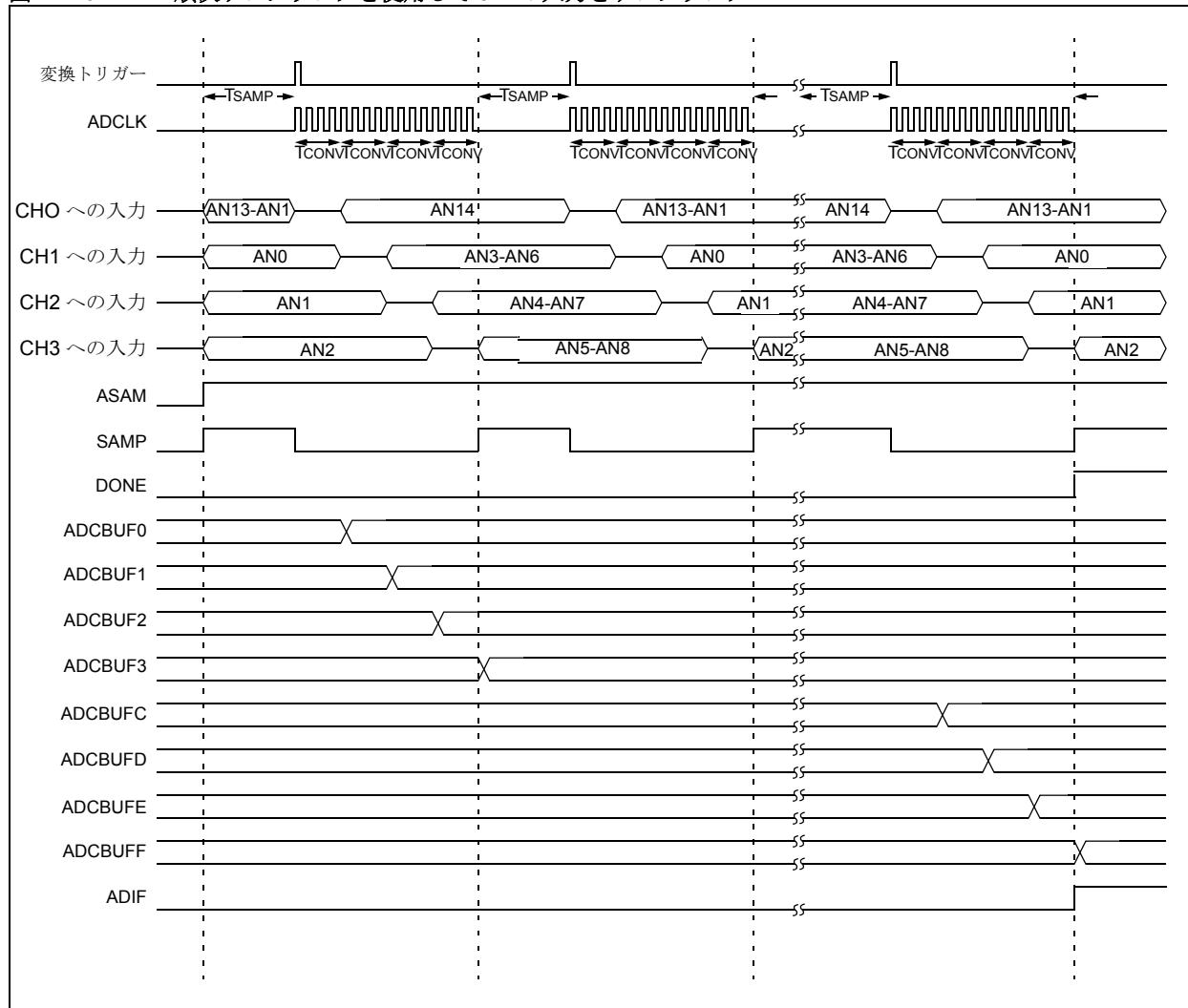


表 17-9: 順次サンプリングを使用して 8 つの入力をサンプリング

制御ビット シーケンス選択		動作シーケンス
SMPI<2:0> = 1111	16 回目のサンプルで割り込み	サンプル : (AN13-AN1) -> CH0 変換 CH0、書き込みバッファ 0x0
CHPS<1:0> = 1x	チャネル CHO、CH1、CH2、CH3 をサンプリング	サンプル : AN0 -> CH1 変換 CH1、書き込みバッファ 0x1
SIMSAM = 0	全てのチャネルを順次にサンプリング	サンプル : AN1 -> CH2 変換 CH2、書き込みバッファ 0x2
BUFM = 0	单一 16 ワード結果バッファ	サンプル : AN2 -> CH3 変換 CH3、書き込みバッファ 0x3
ALTS = 1	交互 MUX A/B 入力選択	サンプル : AN14 -> CH0 変換 CH0、書き込みバッファ 0x4
<b>MUX A 入力選択</b>		サンプル : (AN3-AN6) -> CH1 変換 CH1、書き込みバッファ 0x5
CH0SA<3:0> = 1101	CH0+ 入力には AN13 を選択	サンプル : (AN4-AN7) -> CH2 変換 CH2、書き込みバッファ 0x6
CH0NA = 1	CH0- 入力に AN1 選択	サンプル : (AN5-AN8) -> CH3 変換 CH3、書き込みバッファ 0x7
CSCNA = 0	入力スキャンなし	サンプル : (AN13-AN1) -> CH0 変換 CH0、書き込みバッファ 0x8
CSSL<15:0> = n/a	スキャン入力選択未使用	サンプル : AN0 -> CH1 変換 CH1、書き込みバッファ 0x9
CH123SA = 0	CH1+ = AN0、CH2+ = AN1、CH3+ = AN2	サンプル : AN1 -> CH2 変換 CH2、書き込みバッファ 0xA
CH123NA<1:0> = 0x	CH1-、CH2-、CH3- = VREF-	サンプル : AN2 -> CH3 変換 CH3、書き込みバッファ 0xB
<b>MUX B 入力選択</b>		サンプル : AN14 -> CH0 変換 CH0、書き込みバッファ 0xC
CH0SB<3:0> = 1110	CH0+ 入力には AN14 を選択	サンプル : (AN3-AN6) -> CH1 変換 CH1、書き込みバッファ 0xD
CH0NB = 0	CH0- 入力に VREF- 選択	サンプル : (AN4-AN7) -> CH2 変換 CH2、書き込みバッファ 0xE
CH123SB = 1	CH1+ = AN3、CH2+ = AN4、CH3+ = AN5	サンプル : (AN5-AN8) -> CH3 変換 CH3、書き込みバッファ 0xF
CH123NB<1:0> = 10	CH1-=AN6、CH2-=AN7、CH3- = AN8	割り込み 繰り返し

バッファ  
アドレス  
ADCBUF0  
ADCBUF1  
ADCBUF2  
ADCBUF3  
ADCBUF4  
ADCBUF5  
ADCBUF6  
ADCBUF7  
ADCBUF8  
ADCBUF9  
ADCBUFA  
ADCBUFB  
ADCBUFC  
ADCBUD  
ADCBUE  
ADCBUFF

バッファ @  
1 回目割り込み

(AN13-AN1) サンプル 1  
AN0 サンプル 2  
AN1 サンプル 3  
AN2 サンプル 4  
AN14 サンプル 5  
(AN3-AN6) サンプル 6  
(AN4-AN7) サンプル 7  
(AN5-AN8) サンプル 8  
(AN13-AN1) サンプル 9  
AN0 サンプル 10  
AN1 サンプル 11  
AN2 サンプル 12  
AN14 サンプル 13  
(AN3-AN6) サンプル 14  
(AN4-AN7) サンプル 15  
(AN5-AN8) サンプル 16

バッファ @  
2 回目割り込み

(AN13-AN1) サンプル 17  
AN0 サンプル 18  
AN1 サンプル 19  
AN2 サンプル 20  
AN14 サンプル 21  
(AN3-AN6) サンプル 22  
(AN4-AN7) サンプル 23  
(AN5-AN8) サンプル 24  
(AN13-AN1) サンプル 25  
AN0 サンプル 26  
AN1 サンプル 27  
AN2 サンプル 28  
AN14 サンプル 29  
(AN3-AN6) サンプル 30  
(AN4-AN7) サンプル 31  
(AN5-AN8) サンプル 32

• • •

## 17.16 A/D サンプリング要件

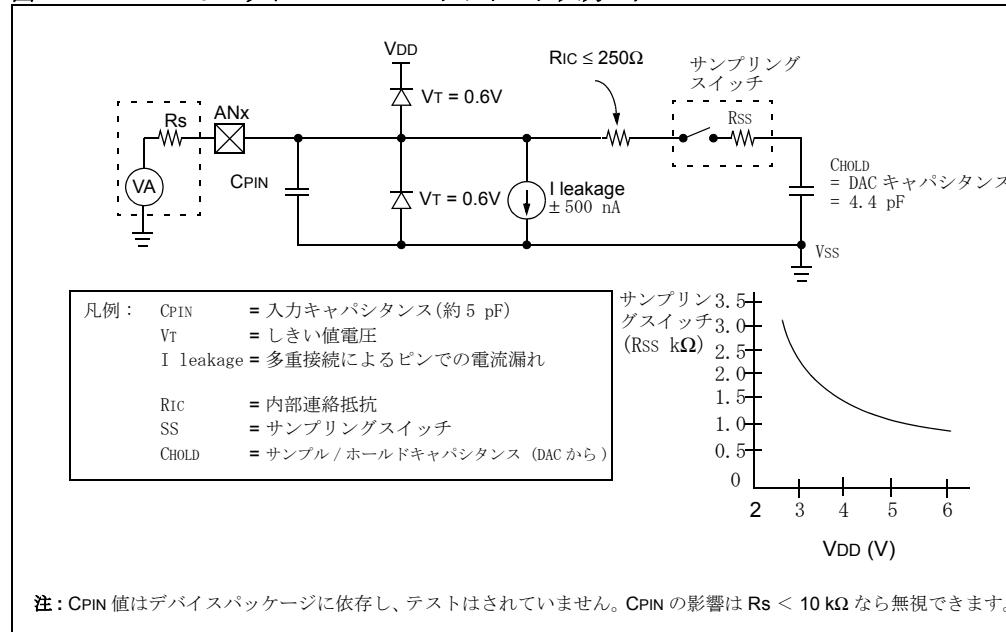
A/D の合計サンプリング時間は、式 17-7 に示されるように、アンプのセトリングタイムとホールドキャパシタの充電時間と温度の関数です。

10 ビット A/D コンバータのアナログ入力モデルは図 17-21 に示されています。A/D コンバータの仕様精度を出すためには、コンデンサ (CHOLD) をアナログ入力ピンの電圧レベルまで十分充電する必要があります。ソースインピーダンス ( $Rs$ ) および内部サンプリングスイッチ ( $Rss$ ) インピーダンスはコンデンサ  $CHOLD$  の充電にかかる時間に直接影響を及ぼします。さらに、サンプリングスイッチ ( $Rss$ ) インピーダンスは、図 17-21 に示されるように、デバイス電圧 ( $VDD$ ) とともに変化します。そこで、アナログソースのインピーダンス ( $Rs$ ) とサンプリングスイッチインピーダンス ( $Rss$ ) の和が、選択されたサンプル時間内にホールドキャパシタを完全に充電するために十分小さいものである必要があります。A/D コンバータの精度に対するピンの電流漏れの影響を最小にするために、最大の推奨ソースインピーダンス  $Rs$  は  $10 \text{ k}\Omega$  です（充電時間計算結果にかかわらず）。アナログ入力チャネルが選択（変更）されたのちに、このサンプリング機能は変換開始の際には完了している必要があります。内部ホールドキャパシタは各サンプル動作に先立ち放電された状態になります。

サンプリングコンデンサ充電時間を計算するには、式 17-8 が使用できます。サンプリングコンデンサは各変換の後に放電されるので、この式では、アナログ入力が  $OV$  から上昇して LSB の ‘ $n$ ’ のレベルまで上がって、出力エラーが  $1/2LSB$  以下になるまでの時間と仮定しています（すなわち 10 ビット A/D のための 2048 段階です）。 $1/2LSB$  エラーは A/D が特定の分解能で許容できる最大限度のエラーです。A/D 変換用  $CHOLD$  は  $4.4 \text{ pF}$  です。

合計サンプリング時間を計算するには、アンプのセトリングタイム  $TAMP$  を  $0.5 \mu\text{sec}$  と仮定します。温度係数は式 17-9 で計算します。温度が  $25^\circ\text{C}$  未満の場合、温度係数は 0（ゼロ）になります。

図 17-21: 10 ビット A/D コンバータアナログ入力モデル



## 式 17-7: サンプリング時間

$$TSMP = \text{アンプセトリングタイム (TAMP)} + \\ \text{ホールドキャパシタ充電時間 (TC)} + \\ \text{温度係数 (TCOFF)}$$

$$TSMP = TAMP + TC + TCOFF$$

注: TAMP は 0.5 μs です。

## 式 17-8: A/D ホールドキャパシタ充電時間

$$TC = -CHOLD (RIC + RSS + RS) \ln(1/2n) \text{ secs}$$

## 式 17-9: A/D ホールドキャパシタ充電時間

$$TCOFF = (Temp - 25^\circ\text{C})(0.05 \mu\text{s}/^\circ\text{C})$$

注: TCOFF は 25°C 未満のすべての温度に対して '0' です。

例 17-7 では、特定の温度で、入力抵抗 RS が関連するとしたときの、最速のサンプリングタイムを計算しています。この計算は以下のシステム仮定に基づいています。

1. CHOLD = 4.4 pF
2. RIC = 250Ω
3. RS = 1Ω (オペアンプでドライブされているとする)
4. VI = 1023 LSB (フルスケール入力電圧); n = 1024
5. VDD = 5V → RSS = 1.2 kΩ
6. Temp (公称) = 25°C

## 例 17-7: 最速のケースサンプル時間計算

$$TC = -CHOLD (RIC + RSS + RS) \ln(1/2n) \\ = -2.5 \text{ pF} (250\Omega + 1.2 \text{ k}\Omega + 1\Omega) \ln(1/2048) \\ = 0.049 \mu\text{s}$$

$$TSMP = TAMP + TC + TCOFF \\ = 0.5 \mu\text{s} + 0.049 \mu\text{s} + [(25^\circ\text{C} - 25^\circ\text{C})(0.05 \mu\text{s}/^\circ\text{C})] \\ = 0.55 \mu\text{s}$$

例 17-8 には最悪のケースサンプル時間の計算が示されています。この計算は以下のシステム仮定に基づいています。

1. CHOLD = 4.4 pF
2. RS = 10 kΩ (内部抵抗のあるセンサーでドライブしているものとする)
3. 10-bit A/D, LSB = 1024
4. VI = 1023 LSB (フルスケール入力電圧) ; n = 1024
5. VDD = 5V → RSS = 1.2 kΩ
6. Temp (最大) = 25°C

## 例 17-8: 最悪のサンプル時間計算、最大ソースインピーダンス

$$\begin{aligned}TC &= -CHOLD(RIC + RSS + RS) \ln(1/2n) \\&= -4.4 \text{ pF} (250\Omega + 1.2 \text{ k}\Omega + 10 \text{ k}\Omega) \ln(1/2048) \\&= 0.38 \mu\text{s}\end{aligned}$$

$$\begin{aligned}TSMP &= TAMP + TC + TCOFF \\&= 0.5 \mu\text{s} + 0.38 \mu\text{s} + [(25^\circ\text{C} - 25^\circ\text{C})(0.05 \mu\text{s}/^\circ\text{C})] \\&= 0.88 \mu\text{s}\end{aligned}$$

例 17-9 では、例 17-7 と同じ条件を使った（温度以外で）サンプル時間に対する高温の影響が示されています。他の基準と比べて、温度係数はサンプル時間に対して最も大きな影響を与えます。この計算は以下のシステム仮定に基づいています。

1. CHOLD = 4.4 pF
2. RIC = 250Ω
3. RS = 1Ω (オペアンプでドライブされているとする)
4. VI = 1023 LSB (フルスケール入力電圧) ; n = 1024
5. VDD = 5V → RSS = 5 kΩ
6. Temp (公称) = 85°C

## 例 17-9: 最悪のサンプル時間計算、高温

$$\begin{aligned}TC &= -CHOLD(RIC + RSS + RS) \ln(1/2n) \\&= -4.4 \text{ pF} (250\Omega + 1.2 \text{ k}\Omega + 1\Omega) \ln(1/2048) \\&= 0.049 \mu\text{s}\end{aligned}$$

$$\begin{aligned}TSMP &= TAMP + TC + TCOFF \\&= 0.5 \mu\text{s} + 0.049 \mu\text{s} + [(85^\circ\text{C} - 25^\circ\text{C})(0.05 \mu\text{s}/^\circ\text{C})] \\&= 0.5 \mu\text{s} + 0.049 \mu\text{s} + 3 \mu\text{s} \\&= 3.55 \mu\text{s}\end{aligned}$$

### 17.17 A/D 結果バッファの読み込み

RAM は 10 ビット幅です、ただしバッファからの読み込みが実行されると、データは自動的に選択可能なフォーマットの 1 つにフォーマットされます。FORM<1:0> ビット (ADCON1<9:8>) によりフォーマットが選択されます。フォーマットを実行するハードウェアが、どのフォーマットの場合でもデータバス上に 16 ビットの結果を出力します。図 17-22 では、FORM<1:0> 制御ビットで選択されるデータ出力フォーマットが示されています。

図 17-22: A/D 出力データフォーマット

RAM コンテンツ :	<table border="1"><tr><td>d09</td><td>d08</td><td>d07</td><td>d06</td><td>d05</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td></tr></table>	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00						
d09	d08	d07	d06	d05	d04	d03	d02	d01	d00								
バスに読み込み :																	
整数	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>d09</td><td>d08</td><td>d07</td><td>d06</td><td>d05</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td></tr></table>	0	0	0	0	0	0	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00
0	0	0	0	0	0	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00		
符号付整数	<table border="1"><tr><td>d09</td><td>d09</td><td>d09</td><td>d09</td><td>d09</td><td>d09</td><td>d09</td><td>d08</td><td>d07</td><td>d06</td><td>d05</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td></tr></table>	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00						
d09	d09	d09	d09	d09	d09	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00		
固定小数 (1.15)	<table border="1"><tr><td>d09</td><td>d08</td><td>d07</td><td>d06</td><td>d05</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0
d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0		
符号付き固定小数 (1.15)	<table border="1"><tr><td>d09</td><td>d08</td><td>d07</td><td>d06</td><td>d05</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0
d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0		

図 17-23: 種々の結果コードの表現

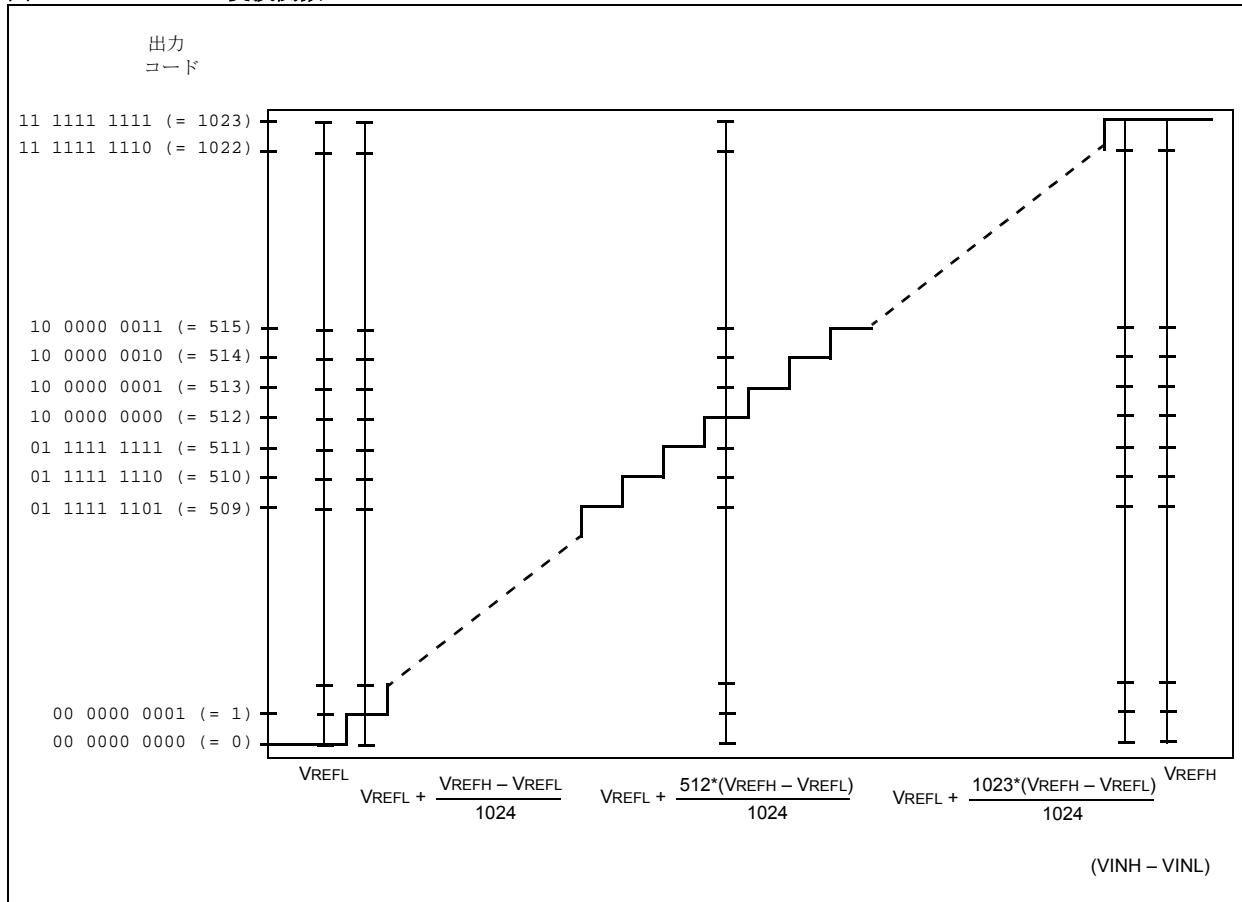
VIN/VREF	10 ビット 出力コード	16 ビット整数フォー マット	16 ビット符号付整数 フォーマット	16 ビット小数フォー マット	16 ビット符号付小数フォー マット
1023/1024	11 1111 1111	0000 0011 1111 1111 = 1023	0000 0001 1111 1111 = 511	1111 1111 1100 0000 = 0.999	0111 1111 1100 0000 = 0.499
1022/1024	11 1111 1110	0000 0011 1111 1110 = 1022	0000 0001 1111 1110 = 5 10	1111 1111 1000 0000 = 0.998	0111 1111 1000 0000 = 0.498
***					
513/1024	10 0000 0001	0000 0010 0000 0001 = 513	0000 0000 0000 0001 = 1	1000 0000 0100 0000 = 0.501	0 000 0000 0100 0000 = 0.001
512/1024	10 0000 0000	0000 0010 0000 0000 = 512	0000 0000 0000 0000 = 0	1000 0000 0000 0000 = 0.500	0000 0000 0000 0000 = 0.000
511/1024	01 1111 1111	0000 0001 1111 1111 = 511	1111 1111 1111 1111 = -1	0111 1111 1100 0000 = .499	1111 1111 1100 0000 = -0.001
***					
1/1024	00 0000 0001	0000 0000 0000 0001 = 1	1111 1110 0000 0001 = -511	0000 0000 0100 0000 = 0.001	1000 0000 0100 0000 = -0.499
0/1024	00 0000 0000	0000 0000 0000 0000 = 0	1111 1110 0000 0000 = -512	0000 0000 0000 0000 = 0.000	1000 0000 0000 0000 = -0.500

## 17.18 変換関数

A/D コンバータの理想的な変換関数は図 17-24 に示されています。入力電圧 ( $V_{INH} - V_{INL}$ ) の差分がリファレンス ( $V_{REFH} - V_{REFL}$ ) と比較されます。

- 入力電力が  $(V_{REFH} - V_{REFL}/2048)$  または  $0.5 \text{ LSb}$  の時、最初のコード遷移が発生します。
- 00 0000 0001 コードは  $(V_{REFH} - V_{REFL}/1024)$  または  $1.0 \text{ LSb}$  を中心としています。
- 10 0000 0000 コードは  $(512*(V_{REFH} - V_{REFL})/1024)$  を中心としています。
- $(1*(V_{REFH} - V_{REFL})/2048)$  未満の入力電圧は 00 0000 0000 として変換されます。
- $(2045*(V_{REFH} - V_{REFL})/2048)$  を超える入力は 11 1111 1111 として変換されます。

図 17-24: A/D 変換関数



## 17.19 A/D 精度 / 誤差

A/D の精度を論じる文書のリストについては、セクション 17.26「関連するアプリケーションノート」を参照してください。

## 17.20 接続の条件

アナログ入力は ESD プロテクションを採用しているので、 $V_{DD}$  および  $V_{SS}$  に対してダイオードがあります。このことにより、アナログ入力は  $V_{DD}$  と  $V_{SS}$  の間である必要があります。入力電圧がこの範囲より（どちらの方向へも） $0.3\text{V}$  以上超えると、ダイオードの 1 つが順方向にバイアスされ電流が流れます。入力電流仕様を上回るとデバイスがダメージを受ける可能性があります。

入力信号のアンチエイリアス処理のために外部 RC フィルタが時折追加されます。この場合の R コンポーネントは、サンプリング時間要件が満たされるように選択される必要があります。さらに、アナログ入力ピンに接続される全てのコンポーネント（コンデンサ、ゼナーダイオードなど）は、高抵抗経由で接続されている場合でも、非常に少ない漏れ電流のものを使う必要があります。

## 17.21 初期化

A/D モジュールの単純初期化コード例は例 17-10 に示されています。

この特別な構成において、AN0-AN15 の 16 のアナログ入力ピン全てがアナログ入力として設定されています。IDLE モードでの動作は無効であり、出力データは符号なし固定小数フォーマットとし、AVDD および AVSS を VREFH および VREFL として使用します。変換(変換トリガー)の開始と同じくサンプリングの開始はソフトウェア内で手動で実行されます。CHO S/H 増幅器を変換に使用します。入力のスキャニングは無効で各サンプル・変換シーケンス(1 変換結果)ごとに割り込みが発生します。A/D 変換クロックは Tcy/2 です。

各変換が完了した後に、サンプリングは SAMP ビット (ADCON1<1>) の設定により手動で開始されるので、自動サンプル時間ビット、SAMC<4:0> (ADCON3<12:8>) は無視されます。さらに、変換の開始(すなわちサンプリングの終了)もまた手動でトリガされているので、SAMP ビットを新しいサンプルが変換される必要があるたびにクリアする必要があります。

例 17-10: A/D 初期化コード例

```

CLR      ADPCFG          ; A/D ポート設定、
                  ; 入力ピンは全てアナログです。
MOV      #0x2208,W0       ;
MOV      W0,ADCON1        ; サンプルクロックソース
                  ; および変換トリガーモード設定。
                  ; 符号無し固定小数フォーマット、
                  ; 手動変換トリガー、
                  ; サンプリングの手動開始、
                  ; 同時サンプリング、
                  ; IDLE モードでは動作しない。
CLR      ADCON2           ; A/D 電圧リファレンス
                  ; およびバッファ充填モード設定。
                  ; AVDD および AVSS から VREF、
                  ; 入力はスキャンされない、
                  ; 1 S/H チャネル使用、
                  ; サンプリング毎に割り込み
CLR      ADCON3           ; A/D 変換クロック設定
CLR      ADCHS            ; 入力チャネル設定、
                  ; CH0+ 入力は AN0。
                  ; CHO- 入力は VREFL (AVSS)
CLR      ADCSSL            ; インプットはスキャンされない。
BCLR    IFS0,#ADIF        ; A/D 変換割り込みをクリア
                  ; 必要な場合は、ここで A/D 割り込み優先順位ビット (ADIP<2:0>) を設定
                  ; (デフォルトの優先順位レベルは 4)
BSET    IEC0,#ADIE        ; A/D 変換割り込みを有効化
BSET    ADCON1,#ADON        ; A/D をオンにする
BSET    ADCON1,#SAMP        ; 入力サンプリングを開始
CALL    DELAY             ; 変換開始前にサンプリングが正常に終了
                  ; していることを確認。
BCLR    ADCON1,#SAMP        ; A/D サンプリングを終了し、変換開始
                  ; 変換シーケンスが終了すると、ハードウェア
                  ; により DONE ビットがセットされる。
                  ; ADIF ビットが設定される。
:
```

## 17.22 SLEEP モードおよび IDLE モード時の動作

CPU、バス、周辺機器のデジタルアクティビティが最小化されるので、SLEEP モードと IDLE モードは変換ノイズを最小化するためには有効です。

### 17.22.1 RC A/D クロックなしの CPU SLEEP モード

デバイスが SLEEP モードに入った場合、モジュールに対するすべてのクロックソースは停止され、論理 ‘0’ にとどまります。

変換の途中で SLEEP が発生すると、A/D は内蔵 RC クロックジェネレータからクロック供給されることなく変換が中断されます。SLEEP モードからウェイクアップしたときも一部完了済み変換が継続開始されることもありません。

レジスタ内容は、SLEEP モードに入ったり出たりするデバイスの影響を受けません。

### 17.22.2 RC A/D クロック付きの CPU SLEEP モード

A/D クロックソースが内蔵 A/D RC オシレータ (ADRC = 1) に設定されると、A/D モジュールは SLEEP モード中にも動作可能です。この場合には、変換の際のデジタルスイッチノイズが除去されます。変換が完了すると、DONE ビットがセットされ、変換結果は A/D 結果バッファである ADCBUF にロードされます。

A/D 割り込みが有効 (ADIE = 1) ならば、A/D 割り込み発生時にデバイスは SLEEP からウェイクアップします。その後のプログラム実行はそのときの A/D 割り込みが電流 CPU 優先順位より高い場合、A/D 割り込みサービスルーチンから開始されます。割り込み優先順位が低い場合には、デバイスを SLEEP モードにする PWRSAV 命令の次の命令から実行が続行されます。

A/D 割り込みが有効でなければ、ADON ビットがセットのままになっていても、この時 A/D モジュールはオフになります。

A/D モジュール動作におけるデジタルノイズの影響を最小にするために、ユーザーは A/D 変換が SLEEP モードでも実行できる変換トリガーソースを選ぶ必要があります。自動変換トリガーオプションは、SLEEP (SSRC<2:0> = 111) 中のサンプリングおよび変換に使用できます。自動変換オプションを使用するために、ADON ビットは PWRSAV 命令に先立つ命令に設定される必要があります。

**注：** A/D モジュールを SLEEP モードで動作させるには、A/D クロックソースを RC(ADRC = 1) に設定する必要があります。

### 17.22.3 CPU が IDLE モード中の A/D 操作

A/D では、ADSIDL ビット (ADCON1<13>) により、モジュールが IDLE 中の動作を停止するか IDLE 中も動作を継続するかが選択されます。ADSIDL = 0 ならば、デバイスが IDLE モードに入ってもモジュールの通常動作が継続されます。A/D 割り込みが有効 (ADIE = 1) ならば、A/D 割り込み発生時にデバイスは IDLE モードからウェイクアップします。A/D 割り込みがそのときの CPU 優先順位より高い場合には、プログラム実行は A/D 割り込みサービスルーチンから開始されます。優先順位が低い場合には、デバイスを IDLE モードにする PWRSAV 命令の次の命令から実行が続行されます。

ADSIDL = 1 ならば、モジュールは IDLE 中は動作を停止します。変換中にデバイスが IDLE モードに入れば、変換は中断されます。IDLE モードからウェイクアップしても、一部完了済みの変換動作が継続開始されることはありません。

## 17.23 リセットの影響

デバイス RESET によりすべてのレジスタが強制的に RESET 状態にされます。これにより A/D にモジュールは強制的にオフになり、進行中のすべての変換は中断されます。アナログ入力と複合されたすべてのピンはアナログ入力として構成されます。対応する TRIS ビットがセットされます。

ADCBUF レジスタの値はパワーオンリセットの間初期化されません。ADCBUFO から ADCBUFF には未知のデータが含まれます。

## 17.24 10 ビット A/D コンバータに関する特殊関数レジスタ

以下の表はアドレス、フォーマットを含む dsPIC30F 10 ビット A/D 特殊関数レジスタの一覧です。未実装レジスタまたはレジスタ内のビット（あるいはその両方）はすべてゼロとして読み込まれます。

表 17-10: ADC レジスタマップ

ファイル名	ADR	ビット 15	ビット 14	ビット 13	ビット 12	ビット 11	ビット 10	ビット 9	ビット 8	ビット 7	ビット 6	ビット 5	ビット 4	ビット 3	ビット 2	ビット 1	ビット 0	RESET ステータス												
INTCON1	0080	NSTDIS	—	—	—	—	OVATE	OVBTE	COVTE	—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000 0000 0000 0000												
INTCON2	0082	ALTIVT	—	—	—	—	—	—	—	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INTOEP	0000 0000 0000 0000												
IFS0	0084	CNIF	BCLIF	I2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INTO	0000 0000 0000 0000												
IEC0	008C	CNIE	BCLIE	I2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INTOIE	0000 0000 0000 0000												
IPC2	0098	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>			0100 0100 0100 0100												
ADCBUF0	0280	ADC データバッファ 0																uuuu uuuu uuuu uuuu												
ADCBUF1	0282	ADC データバッファ 1																uuuu uuuu uuuu uuuu												
ADCBUF2	0284	ADC データバッファ 2																uuuu uuuu uuuu uuuu												
ADCBUF3	0286	ADC データバッファ 3																uuuu uuuu uuuu uuuu												
ADCBUF4	0288	ADC データバッファ 4																uuuu uuuu uuuu uuuu												
ADCBUF5	028A	ADC データバッファ 5																uuuu uuuu uuuu uuuu												
ADCBUF6	028C	ADC データバッファ 6																uuuu uuuu uuuu uuuu												
ADCBUF7	028E	ADC データバッファ 7																uuuu uuuu uuuu uuuu												
ADCBUF8	0290	ADC データバッファ 8																uuuu uuuu uuuu uuuu												
ADCBUF9	0292	ADC データバッファ 9																uuuu uuuu uuuu uuuu												
ADCBUFA	0294	ADC データバッファ 10																uuuu uuuu uuuu uuuu												
ADCBUFB	0296	ADC データバッファ 11																uuuu uuuu uuuu uuuu												
ADCBUFC	0298	ADC データバッファ 12																uuuu uuuu uuuu uuuu												
ADCBUFD	029A	ADC データバッファ 13																uuuu uuuu uuuu uuuu												
ADCBUFE	029C	ADC データバッファ 14																uuuu uuuu uuuu uuuu												
ADCBUFF	029E	ADC データバッファ 15																uuuu uuuu uuuu uuuu												
ADCON1	02A0	ADON	ADFRZ	ADSLD	—	—	—	FORM[1:0]		SSRC[2:0]		—	SIMSAM	ASAM	SAMP	CONV	0000 0000 0000 0000													
ADCON2	02A2	VCFG[2:0]			OFFCAL	—	CSCNA	CHPS[1:0]		BUFS	—	SMP1[3:0]			BUFM	ALTS	0000 0000 0000 0000													
ADCON3	02A4	—	—	—	SAMC[4:0]					ADRC	—	ADCS[5:0]						0000 0000 0000 0000												
ADCHS	02A6	CHXNB[1:0]		CHXSB	CHONB	CHOSB[3:0]			CHXNA[1:0]		CHXSA	CHONA	CHOSA[3:0]				0000 0000 0000 0000													
ADPCFG	02A8	PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8	PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000 0000 0000 0000												
ADCSSL	02AA	ADC 入力スキャン選択レジスタ																0000 0000 0000 0000												

凡例: u = 未知

注: すべての割り込みソースおよびその関連制御ビットは特定のデバイスでは利用できない可能性があります。詳細についてはデバイスデータシートを参照してください。

## 17.25 設計の秘訣

**質問 1:** A/D コンバータのシステムパフォーマンスはどうすれば最適化できますか？

**答え :**

1. タイミング仕様をすべて満たしていることをご確認ください。モジュールのオフ、オンライン切り替え時には、サンプルを取得する前に最低限の遅延が必要となります。入力チャネル変更時にも、同じく待機のための最小限の遅延が発生し、そして最終的に各ビット変換のために選択された時間である  $T_{AD}$  が存在します。この遅延は ADCON3 で選択しますが、電気的仕様で指定された範囲内の値を選択するべきです。 $T_{AD}$  が短過ぎる場合、その結果は変換終了前に十分に変換されない可能性があり、 $T_{AD}$  が長過ぎる場合、サンプリングコンデンサの電圧は変換が完了される前に無くなる可能性があります。このタイミング特性はデバイスデータシートの「電気的仕様」セクションで説明されます。
2. しばしば、アナログシグナルのソースインピーダンスは高く ( $10\text{ k}\Omega$  を超える)、そのためソースから引き出された電流は、サンプルコンデンサを十分充電できず、精度に影響を及ぼす可能性があります。入力シグナルが変わるのが遅い場合は、 $0.1\text{ }\mu\text{F}$  コンデンサをアナログ入力に接続してみてください。このコンデンサにより、サンプルされているアナログ電圧に充電され、 $4.4\text{ pF}$  内蔵ホールドキャッシュを充電するために十分な瞬間電流が供給されます。
3. A/D 変換を開始する前にデバイスを SLEEP モードにしてください。RC クロックソース選択が SLEEP モードでの変換に必要です。CPU 及びその他の周辺機器からのデジタルノイズが最小化されるので、このテクニックにより精度が向上します。

**質問 2:** A/D に関する良い参考文献をご存知ですか？

**答え :** A/D 変換を理解するための良い参考文献は Prentice Hall (ISBN 0-13-03-2848-0) 発行の『Analog-Digital Conversion Handbook』第 3 版です。

**質問 3:** チャネル・サンプルとサンプル / 割り込みのコンビネーションがバッファのサイズを超えています。バッファに何が起こるのでしょうか？

**答え :** この構成は推奨されません。このバッファには未知の結果が含まれるでしょう。

## 17.26 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありません。ただし、その概念は適切であり、修正や可能な制限をして使用できる可能性もあります。10 ビット A/D コンバータモジュールに関連する現在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 #
アナログ / デジタル (A/D) コンバータの使い方	AN546
ディスプレイ、キーボード付きの 4 チャネルデジタル電圧メーター	AN557
A/D コンバータ性能仕様の理解	AN693

注： デバイスの dsPIC30F ファミリー関しての、追加のアプリケーションノートやコード例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 17.27 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

dsPIC30F 10 ビット A/D コンバータモジュールの編集上の変更および技術情報の変更を反映するため改訂されました。



MICROCHIP®

## 第 18 章 . 12- ビット A/D コンバータ

### ハイライト

この章には以下の項目が含まれます。

18.1 序章 .....	18-2
18.2 制御レジスタ .....	18-4
18.3 A/D 結果バッファ .....	18-4
18.4 A/D 専門用語、変換シーケンス .....	18-10
18.5 A/D モジュール構成 .....	18-11
18.6 電圧リファレンスソース選択 .....	18-11
18.7 A/D 変換クロックの選択 .....	18-12
18.8 サンプリングのアナログ入力選択 .....	18-13
18.9 モジュールの有効化 .....	18-15
18.10 サンプリングの開始方法 .....	18-15
18.11 サンプリングの停止および変換の開始方法 .....	18-15
18.12 サンプル・変換動作の制御 .....	18-20
18.13 変換結果をバッファに書き込む方法の指定 .....	18-21
18.14 変換シーケンス例 .....	18-22
18.15 A/D サンプリング要件 .....	18-27
18.16 A/D 結果バッファの読み込み .....	18-30
18.17 変換閾数 .....	18-31
18.18 A/D の精度と誤差 .....	18-31
18.19 接続条件 .....	18-31
18.20 初期化 .....	18-32
18.21 SLEEP モードおよび IDLE モード時の動作 .....	18-33
18.22 RESET の影響 .....	18-33
18.23 12 ビット A/D コンバータに付随する特別機能レジスタ .....	18-34
18.24 設計の秘訣 .....	18-35
18.25 関連するアプリケーションノート .....	18-36
18.26 改訂履歴 .....	18-37

18

12-  
ビット  
A/D  
コンバータ

## 18.1 序章

dsPIC30F 12 ビット A/D コンバータには以下の主要な特徴があります。

- 逐次近似 (SAR) 変換
- 最大 100ksps の変換スピード
- 最大 16 のアナログ入力ピン
- 外部電圧リファレンス入力ピン
- 単極差動 S/H 増幅器
- 自動チャネルスキヤンモード
- 選択可能変換トリガーソース
- 16 ワード変換結果バッファ
- 選択可能バッファ書き込みモード
- 4 種類の結果フォーマットが選択可能
- CPU が SLEEP モードおよび IDLE モードの時も動作

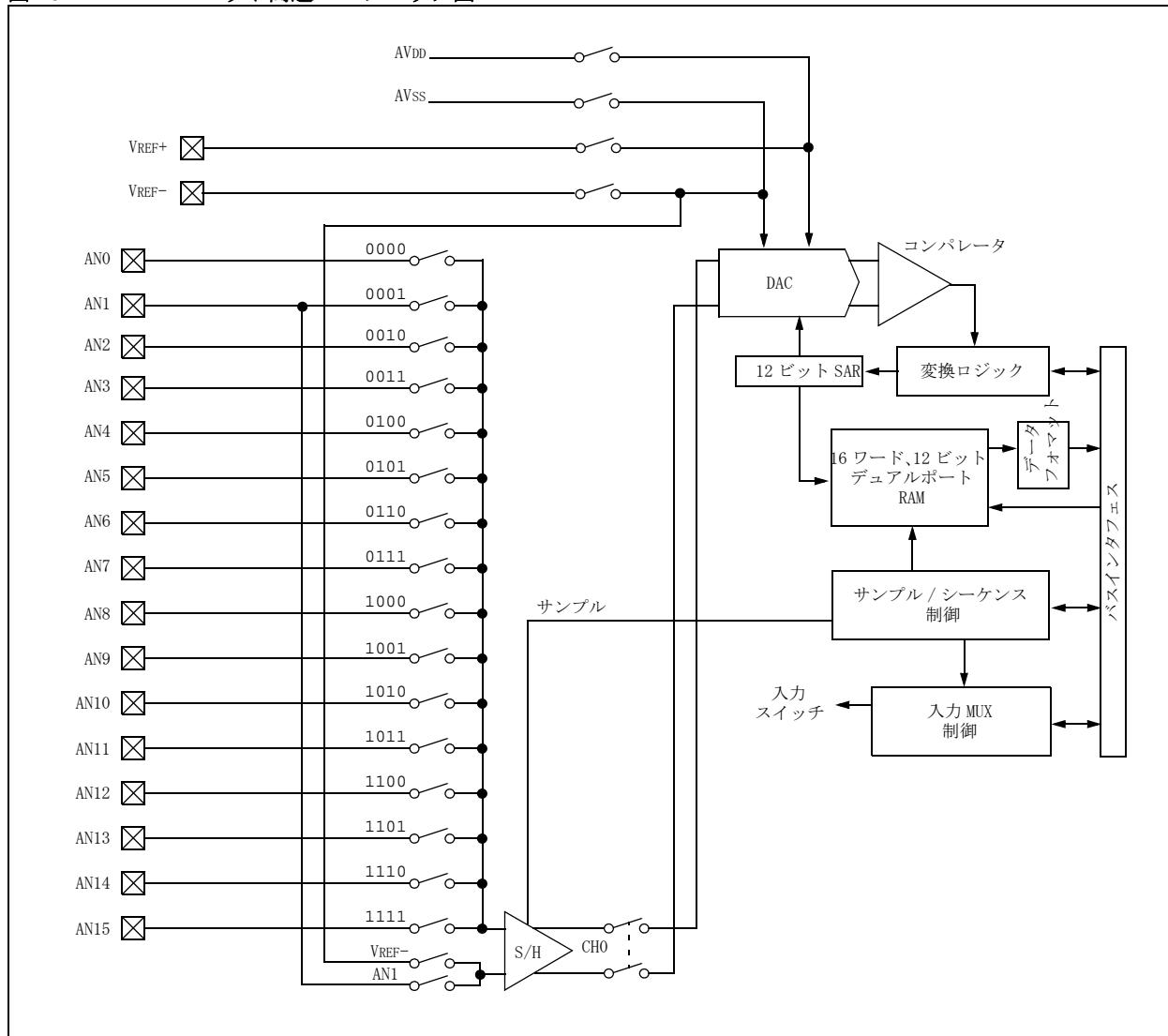
12 ビット A/D のブロック図は図 18-1 に示されます。12 ビット A/D コンバータには、AN0-AN15 と呼称される最大 16 アナログ入力ピンまでが可能です。また、外部電圧リファレンス接続のために 2 つのアナログ入力ピンがあります。この電圧リファレンス入力は他のアナログ入力ピンと共有されます。アナログ入力ピンと外部電圧リファレンス入力構成の実際の数は特定の dsPIC30F デバイスによります。詳細は dsPIC30F デバイスデータシート (DS70082 および DS70083) をご参照ください。

アナログ入力はマルチプレクサを経由して CHO 固定の S/H 増幅器に接続されます。アナログ入力マルチプレクサは変換の間、2 つのアナログ入力の間で切替えが可能です。単極差動変換には特定の入力ピン間が使用できます (図 18-1 をご参照ください)。

アナログ入力スキヤンモードが CHO S/H 増幅器でできます。制御レジスタにより、どのアナログ入力チャネルがスキヤニングシーケンスに含まれるかが決定されます。

12 ビット A/D は 16 ワード結果バッファに格納されます。12 ビットの各結果は、バッファから読み出されるとき、それぞれ 4 つの 16 ビットの出力フォーマットのうちの 1 つに変換されます。

図 18-1: 12 ビット高速 A/D ブロック図



## 18.2 制御レジスタ

A/D モジュールには 6 つの制御ステータスレジスタがあります。このレジスタは以下のもので  
す。

- ADCON1 : A/D 制御レジスタ 1
- ADCON2 : A/D 制御レジスタ 2
- ADCON3 : A/D 制御レジスタ 3
- ADCHS : A/D 入力制御選択レジスタ
- ADPCFG : A/D ポート構成レジスタ
- ADCSSL : A/D 入力スキャン選択レジスタ

ADCON1、ADCON2、ADCON3 レジスタにより A/D モジュールの動作が制御されます。ADCHS レジスタにより S/H 増幅器に接続される入力ピンが選択されます。ADPCFG レジスタにより、アナログ入力またはデジタル I/O としてアナログ入力ピンが構成されます。ADCSSL レジスタにより順次スキャンされるように入力が選択されます。

## 18.3 A/D 結果バッファ

このモジュールには、A/D 結果をバッファするための ADCBUF と呼ばれる 16 ワードデュアルポート RAM が含まれます。16 バッファの位置は ADCBUFO、ADCBUF1、ADCBUF2 ~ ADCBUFE、ADCBFF として参照されます。

**注：** A/D 結果バッファは読み取り専用バッファです。

## レジスタ 18-1: ADCON1A/D制御レジスタ1

上位バイト:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
ADON	—	ADSLD	—	—	—	FORM<1:0>	
ビット 15							ビット 8

下位バイト:							
R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/C-0
					HC, HS	HC, HS	
SSRC<2:0>							ビット 0
ビット 7							

ビット 15 **ADON:** A/D動作モードビット

- 1 = A/Dコンバータモジュールを動作中にする。  
0 = A/Dコンバータはオフです。

ビット 14 未実装: '0'が読み込まれます。

ビット 13 **ADSLD:** IDLEモードビットでの停止

- 1 = デバイスが IDLE モードに入った時動作停止  
0 = IDLE モードでも動作継続

ビット 12-10 未実装: '0'が読み込まれます。

ビット 9-8

- FORM<1:0>:** データ出力フォーマットビット  
11 = 符号付固定小数 (DOUT = sddd dddd dddd 0000)  
10 = 固定小数 (DOUT = dddd dddd dddd 0000)  
01 = 符号付整数 (DOUT = ssss sddd dddd dddd)  
00 = 整数 (DOUT = 0000 dddd dddd dddd)

ビット 7-5

- SSRC<2:0>:** 変換トリガーソース選択ビット

- 111 = 内部カウンタによりサンプリングが終了され、変換（自動変換）が開始されます。  
110 = 予約済み  
101 = 予約済み  
100 = 予約済み  
011 = モーター制御 PWM インターバルによりサンプリングが終了され、変換が開始されます。  
010 = 汎用タイマー 3 比較一致によりサンプリングが終了され、変換が開始されます。  
001 = INTO ピン上のアクティブな遷移によりサンプリングが終了され、変換が開始されます。  
000 = SAMP ビットをクリアすることによりサンプリングが終了され、変換が開始されます。

ビット 4-3 未実装: '0'が読み込まれます。

ビット 2

- ASAM:** A/Dサンプル自動開始ビット

- 1 = 最後の変換が完了するとすぐにサンプリングが開始されます。SAMP ビットは自動で設定されます。  
0 = SAMP ビットが設定されるとサンプリングが開始されます。

ビット 1

- SAMP:** A/Dサンプル開始ビット。

- 1 = 少なくとも 1 つの A/Dサンプル / ホールド増幅器がサンプリング中です。  
0 = A/Dサンプル / ホールド増幅器がサンプリング中です。  
ASAM=0 の時、このビットへの '1' の書き込みによりサンプリングが開始されます。  
SSRC=000 の時、このビットへの '0' の書き込みによりサンプリングが終了され、変換が開始されます。

ビット 0

- DONE:** A/D変換ステータスビット

- 1 = A/D変換は完了しました。  
0 = A/D変換は完了していません。  
このビットをクリアすることにより、実行中動作はどれも影響を受けません。  
ソフトウェアまたは新しい変換の開始によりクリアされます。

凡例:

R = 読出し可能ビット

W = 書込み可能ビット

C = ソフトウェアによりクリア可能

HC = ハードウェアクリア

HS = ハードウェア設定

U = 未実装ビット、「0」として読み出

-n = POR での数値

'1' = ビットがセットさ

'0' = ビットはクリアされ x = ビットは不定

れます

ます

## レジスタ 18-2: ADCON2A/D 制御レジスタ 2

上位バイト：							
R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	U-0	U-0
VCFG<2:0>	—	—	CSCNA	—	—	—	—
ビット 15				ビット 8			

下位バイト：							
R-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUFS	—	—	SMPI<3:0>				BUFM ALTS
ビット 7				ビット 0			

ビット 15-13 **VCFG<2:0>**: 電圧リファレンス構成ビット

	A/D VREFH	A/D VREFL
000	AVDD	AVss
001	外部 VREF+ ピン	AVss
010	AVDD	外部 VREF- ピン
011	外部 VREF+ ピン	外部 VREF- ピン
1xx	AVDD	AVss

ビット 12 **予約済み** : ユーザーはこの位置に ‘0’ と書き込んで下さい。

ビット 11 **未実装** : ‘0’ が読み込まれます。

ビット 10 **CSCNA**: MUX A 入力マルチプレクサ用 CH0+ 入力スキャン制御ビット  
1 = 入力をスキャンする  
0 = 入力をスキャンしない

ビット 9-8 **未実装** : ‘0’ が読み込まれます。

ビット 7 **BUFS**: バッファ格納ステータスピット

BUFM = 1 (ADRES が 2 x8- ワードバッファに分割されている) の時のみ有効  
1 = A/D は現在バッファ 0x8-0xF に格納しています。ユーザーは 0x0-0x7 でデータにアクセス。  
0 = A/D は現在バッファ 0x0-0x7 に格納しています。ユーザーは 0x8-0xF でデータにアクセス。

ビット 6 **未実装** : ‘0’ が読み込まれます。

ビット 5-2 **SMPI<3:0>**: 割込みごとのサンプル・変換シーケンス数指定ビット  
1111 = 16 番目のサンプル・変換シーケンスごとに割込み  
1110 = 15 番目のサンプル・変換シーケンスごとに割込み

.....  
0001 = 2 番目のサンプル・変換シーケンスごとに割込み  
0000 = 各サンプル・変換シーケンスごとに割込み

ビット 1 **BUFM**: バッファモード選択ビット

1 = 2 つの 8 ワードバッファ ADCBUF (15..8)、ADCBUF (7..0) として構成する  
0 = 1 つの 16 ワードバッファ ADCBUF (15..0) として構成する

ビット 0 **ALTS**: 交互入力サンプルモード選択ビット

1 = 第1番目のサンプルのために MUX A 入力マルチプレクサを使用し、次にその後は MUX B と MUX A の間で交互に入れ換える。  
0 = 常に MUX A 入力マルチプレクサを使用する。

凡例 :

R = 読み込み可能ビット      W = 書き込み可能ビット      U = 未実装、‘0’ が読み込まれます

-n = POR での数値      ‘1’ = ビットがセット      ‘0’ = ビットはクリア      x = ビットは不定です

## レジスタ 18-3: ADCON3: A/D 制御レジスタ 3

上位バイト :										
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0			
—	—	—		SAMC<4:0>						
ビット 15							ビット 8			

下位バイト :									
R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		
ADRC	—		ADCS<5:0>						
ビット 7							ビット 0		

ビット 15-13 未実装: ‘0’が読み込まれます。

ビット 12-8 **SAMC<4:0>**: 自動サンプル時間指定ビット  
11111 = 31 TAD  
.....  
00001 = 1 TAD  
00000 = 0 TAD

ビット 7 **ADRC**: A/D 変換クロック指定ビット  
1 = A/D 内蔵 RC クロック  
0 = システムクロック

ビット 6 未実装: ‘0’が読み込まれます。  
ビット 5-0 **ADCS<5:0>**: A/D 変換クロック選択ビット  
111111 = TCY/2 • (ADCS<5:0> + 1) = 32 • TCY  
.....  
000001 = TCY/2 • (ADCS<5:0> + 1) = TCY  
000000 = TCY/2 • (ADCS<5:0> + 1) = TCY/2

## 凡例 :

R = 読出し可能ビット

W = 書込み可能ビット U = 未実装、‘0’が読み込まれます

-n = POR での数値

'1' = ビットがセット '0' = ビットはクリア x = ビットは不定されます

## レジスタ 18-4: ADCHS: A/D 入力選択レジスタ

上位バイト :							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CH0NB	CH0SB<3:0>			
ビット 15							ビット 8

下位バイト :							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CH0NA	CH0SA<3:0>			
ビット 7							ビット 0

ビット 15-13 未実装: ‘0’ が読み込まれます。

ビット 12 **CH0NB:** ビット <4> と同じ定義の MUX B 用チャネル 0 負入力選択 (注 1 をご参照ください)。

ビット 11-8 **CH0SB<3:0>:** ビット <3 : 0> と同じ定義の MUX B 用チャネル 0 正入力選択 (注 1 をご参照ください)。

ビット 7-5 未実装: ‘0’ が読み込まれます。

ビット 4 **CH0NA:** MUX A マルチプレクサ用チャネル 0 負入力選択

1 = チャネル 0 負入力は AN 1 です。

0 = チャネル 0 負入力は VREF- です。

ビット 3-0 **CH0SA<3:0>:** MUX A マルチプレクサ用チャネル 0 正入力選択

1111 = チャンネル 0 正インプットは AN15 です。

1110 = チャネル 0 正入力は AN14 です。

1101 = チャンネル 0 正インプットは AN13 です。

.....

0001 = チャネル 0 正入力は AN1 です。

0000 = チャネル 0 正入力は AN0 です。

**注:** アナログ入力マルチプレクサは MUX A および MUX B と示される 2 つの入力設定構成をサポートします。ADCHS<15 : 8> は MUX B の設定を決定し、ADCHS<7 : 0> は MUX A の設定を決定します。制御ビットの両設定ともあらゆる点で等しく機能します。

凡例 :

R = 読み込み可能ビット

W = 書込み可能ビット U = 未実装、‘0’ が読み込まれます

-n = POR での数値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定されます

## レジスタ 18-5: ADPCFG: A/D ポート構成レジスタ

上位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
ビット 15							ビット 8

## 下位バイト :

| R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| PCFG7 | PCFG6 | PCFG5 | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 |
| ビット 7 |       |       |       |       |       |       | ビット 0 |

ビット 15-0

**PCFG<15:0>**: アナログ入力ピン構成制御ビット

1 = デジタルモードポート読出し可能、A/D のマルチプレクサの入力は AVss に接続される。

0 = アナログモードポート読出し不可、A/D が電圧をサンプルする。

## 凡例 :

R = 読み込み可能ビット

W = 書込み可能ビット U = 未実装、'0' が読み込まれます

-n = POR での数値

'1' = ビットがセット '0' = ビットはクリア x = ビットは不定です

されます

されます

## レジスタ 18-6: ADCSSL: A/D 入力スキャン選択レジスタ

上位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8
ビット 15							ビット 8

## 下位バイト :

| R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| CSSL7 | CSSL6 | CSSL5 | CSSL4 | CSSL3 | CSSL2 | CSSL1 | CSSL0 |
| ビット 7 |       |       |       |       |       |       | ビット 0 |

ビット 15-0

**CSSL<15:0>**: A/D 入力ピンスキャン選択ビット

1 = ANx を入力スキャンする。

0 = ANx のスキャンをスキップする。

## 凡例 :

R = 読み込み可能ビット

W = 書込み可能ビット U = 未実装、'0' が読み込まれます

-n = POR での数値

'1' = ビットがセット '0' = ビットはクリア x = ビットは不定です

されます

されます

## 18.4 A/D 専門用語、変換シーケンス

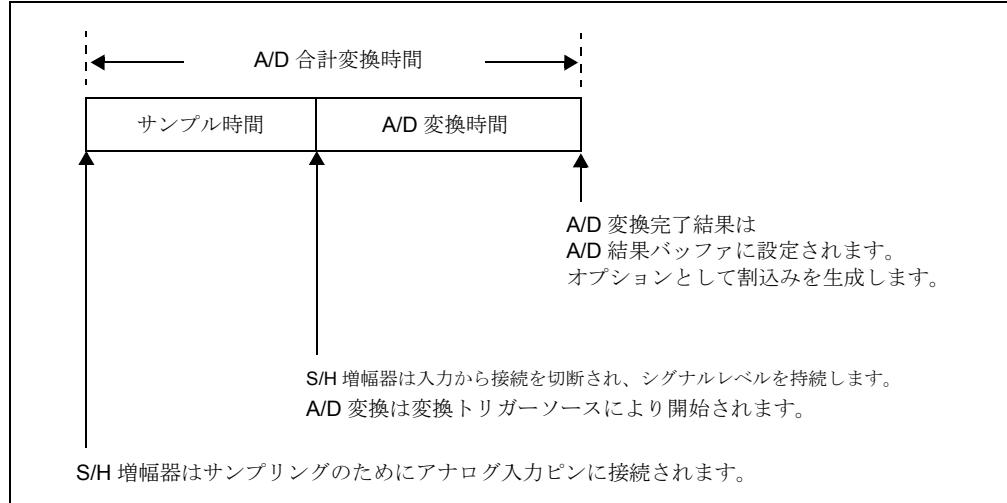
図 18-2 には基本的な変換シーケンスおよび使用用語が示されています。アナログ入力ピン電圧のサンプリングは、S/H 増幅器により実行されます。S/H 増幅器はまた S/H チャネルとも呼ばれます。12 ビット A/D コンバータには CHO という S/H チャネルが 1 つあります。S/H チャネルはアナログ入力マルチプレクサを経由して、アナログ入力ピンに接続されています。アナログ入力マルチプレクサは、ADCHS レジスタにより制御されています。ADCHS レジスタにはあらゆる点で等しく機能するマルチプレクサ制御ビットが 2 組あります。この 2 組の制御ビットにより 2 つの異なるアナログ入力マルチプレクサ構成がプログラムでき、この入力マルチプレクサ構成は MUX A および MUX B と呼ばれます。A/D コンバータにより、オプションとして変換の間で MUX A と MUX B の間を切り替えることができます。またオプションとして、A/D コンバータは、一連のアナログ入力のスキャンができます。

サンプル時間とは A/D モジュールの S/H 増幅器がアナログ入力ピンに接続される時間です。サンプル時間は SAMP ビット (ADCON1<1>) を設定することによって手動で開始されるか、A/D コンバータハードウェアにより自動的に開始できます。サンプル時間は、ユーザーソフトウェアで SAMP ビットをクリアすることにより手動で終了されるか、または変換トリガーソースにより自動的に終了されます。

変換時間とは、A/D コンバータが S/H 増幅器によって保持された電圧を変換するために要する時間です。A/D はサンプル時間終了時にアナログ入力ピンからの接続が切られます。A/D コンバータは 1 ビットあたりの変換に 1A/D クロックサイクル (TAD) が必要で、さらにもう 1 クロック必要とします。完全な変換を実行するためには、全部で 14 の TAD サイクルが必要です。変換時間が完了すると、その結果は 16 ワードの A/D 結果レジタ (ADCBUFO ~ ADCBUFF) の 1 つに格納され、S/H 増幅器は入力ピンに再接続ができる、CPU 割込みが生成されます。

サンプル時間と A/D 変換時間の合計により、合計変換時間ができます。S/H 増幅器が A/D 変換にとって望ましい正確さを出すための最低サンプル時間があります (セクション 18.15 「A/D サンプリング要件」をご参照ください)。さらに、A/D コンバータのための入力クロックオプションが複数あります。ユーザーは最低 TAD 仕様を妨害しない入力クロックオプションを選択する必要があります。

図 18-2: A/D サンプル・変換シーケンス



サンプリングの開始時間は **SAMP** 制御ビットを設定することにより、ソフトウェアで制御できます。サンプリング時間の開始はまたハードウェアによっても自動的に制御されます。A/D コンバータが自動サンプルモードで動作する時、**S/H** 増幅器はサンプル・変換シーケンスでの変換の終了時にアナログ入力ピンに再接続されます。自動サンプル機能は **ASAM** 制御ビットによって制御されます。

変換トリガーソースによりサンプリング時間が終了し、A/D 変換またはサンプル・変換シーケンスが開始されます。変換トリガーソースは **SSRC** 制御ビットにより選択されます。変換トリガーは多様なハードウェアから提供されます。または **SAMP** 制御ビットをクリアすることにより、ソフトウェアで手動で制御できます。変換トリガーソースの 1 つは自動変換です。自動変換の間の時間はカウンタと A/D クロックにより設定されます。自動サンプルモードおよび自動変換トリガーと一緒に使うと、ソフトウェア割込みのないエンドレス自動変換を実現できます。

割り込みが各サンプル・変換シーケンス終了後か、または **SMPI** 制御ビットで設定された回数の複数サンプル・変換シーケンスの終了時に生成されます。割り込み毎のサンプル・変換シーケンスの数は 1 から 16 まで選択できます。

## 18.5 A/D モジュール構成

A/D 変換を動作させるには、以下のステップに従ってください。

### 1. A/D モジュールを構成

- アナログ入力の入力電圧範囲を希望の範囲にするため、電圧リファレンスソースを選択します。
- プロセッサクロックで望ましいデータ変換レートにするため、アナログ変換クロックを選択します。
- サンプリングをどのように発生させるかを決定します。
- 入力をどのように S/H チャネルに割り当てるかを決定します。
- 変換結果をバッファでどのように表現するかを選択します。
- 割込みレートを選択します。
- A/D モジュールをオンにします。

### 2. A/D 割込みを構成（必要に応じて）

- ADIF ビットをクリアします。
- A/D 割込み優先順位を選択します。

各構成の手順のオプションは後の項で説明します。

**注：** ADCON3 および ADCSSL レジスタだけでなく、SSRC<2:0>、SIMSAM、ASAM、CHPS<1:0>、SMPI<3:0>、BUFM および ALTS ビットも、ADON = 1 の間は書き込まないで下さい。この結果が不確定となります。

## 18.6 電圧リファレンスソース選択

A/D 変換のための電圧リファレンス選択には、VCFG<2:0> 制御ビット (ADCON2<15:13>) が使われます。高位側電圧リファレンス (VREFH) および低位側電圧リファレンス (VREFL) は、内蔵 AVDD および AVss 電圧または VREF+ および VREF- 入力ピンとすることができます。

外部電圧リファレンスピンは、少ピンカウントデバイスのときは AN0 および AN1 入力と共有されます。A/D コンバータは、VREF+ および VREF- 入力ピンと共有される場合、さらにこのピンを変換入力として使えます。

外部リファレンスピンに適用される電圧は、特定の仕様に合わせる必要があります。詳しくは「電気的仕様」の章のデバイスデータシートをご参照ください。

## 18.7 A/D 変換クロックの選択

A/D 変換には変換が完了できる最高レートがあります。アナログモジュールクロックである TAD は変換のタイミングを制御します。A/D 変換には 14 クロック周期 (14 TAD) が必要です。A/D クロックはデバイス命令クロックから導かれます。

A/D 変換クロックの周期は 6 ビットカウンタを使ってソフトウェアで選択できます。ADCS<5:0> ビット (ADCON3<5:0>) により 64 種類の TAD が選択できます。式 18-1 により ADCS 制御ビットとデバイス命令クロック周期 TCY の関数として TAD が求められます。

式 18-1: A/D 変換クロック周期

$TAD = \frac{TCY(ADCS + 1)}{2}$ $ADCS = \frac{2TAD}{TCY} - 1$
---

正しい A/D 変換のために、714ns の最低 TAD 時間が確保されるように A/D 変換クロック (TAD) を選択する必要があります。表 18-1 には、デバイス動作周波数および選択した A/D クロックソースの結果の TAD 時間が示されます。

A/D コンバータには変換を実行するために使用できる専用の内蔵 RC クロックソースがあります。内蔵 RC クロックソースは、dsPIC30F が SLEEP モードの間に A/D 変換を動作させる場合に使用されなければなりません。内蔵 RC オシレータは ADRC ビット (ADCON3<7>) の設定により選択されます。ADRC ビットが設定される場合、ADCS<5:0> ビットは A/D 操作に關して無効となります。

表 18-1: 標準的な TAD 対デバイス動作周波数

最低 TAD = 714 ns、最低 TCONV = 10 μs							
AD クロックソース選択			デバイス TCY / デバイス MIPs				
クロック	ADRC	ADCS<5:0>	25 nsec/ 40 MIPs	40 nsec/ 25 MIPs	80 nsec/ 12.5 MIPs	160 nsec/ 6.25 MIPs	1000 nsec/ 1 MIPs
2 TQ	0	000000	12.5 ns <sup>(2)</sup> / .175 μs	20 ns <sup>(2)</sup> / 0.28 μs	40 ns <sup>(2)</sup> / 0.56 μs	80 ns <sup>(2)</sup> / 1.12 μs	500 ns/ 7 μs
4 TQ	0	000001	25 ns <sup>(2)</sup> / 0.35 μs	40 ns <sup>(2)</sup> / 0.56 μs	80 ns <sup>(2)</sup> / 1.12 μs	160 ns <sup>(2)</sup> / 2.24 μs	1.0 μs/ 14 μs
8 TQ	0	000011	50 ns <sup>(2)</sup> / 0.7 μs	80 ns <sup>(2)</sup> / 1.12 μs	160 ns <sup>(2)</sup> / 2.24 μs	320 ns <sup>(2)</sup> / 4.48 μs	2.0 μs <sup>(3)</sup> / 28 μs
16 TQ	0	000111	100 ns <sup>(2)</sup> / 1.4 μs	160 ns <sup>(2)</sup> / 2.24 μs	320 ns <sup>(2)</sup> / 4.48 μs	640 ns/ 8.96 μs	4.0 μs <sup>(3)</sup> / 56 μs
32 TQ	0	001111	200 ns <sup>(2)</sup> / 2.8 μs	320 ns <sup>(2)</sup> / 4.48 μs	640 ns/ 8.96 μs	1.28 μs <sup>(3)</sup> / 17.92 μs	8.0 μs <sup>(3)</sup> / 112 μs
64 TQ	0	011111	400 ns/ 5.6 μs	640 ns/ 8.96 μs	1.28 μs <sup>(3)</sup> / 17.92 μs	2.56 μs <sup>(3)</sup> / 35.84 μs	16.0 μs <sup>(3)</sup> / 224 μs
128 TQ	0	111111	800 ns <sup>(3)</sup> / 11.2 μs	1.28 μs <sup>(3)</sup> / 17.92 μs	2.56 μs <sup>(3)</sup> / 35.84 μs	5.12 μs <sup>(3)</sup> / 71.68 μs	32.0 μs <sup>(3)</sup> / 448 μs
RC	1	xxxxxx	1.2-1.8 μs/ 19.5 μs <sup>(1,4)</sup>				

注 1: RC ソースには VDD > 3.0V のとき 1.5 ns の標準 TAD 時間があります。

2: この値は最低必要 TAD 時間が不足するので使えません。

3: より速い変換時間を得るために別のクロックソースを選択することをお勧めします。

4: RC クロックソースが Fosc > 20 MHz のとき A/D の精度は保証されません。

## 18.8 サンプリングのアナログ入力選択

サンプル・ホールド増幅器にはどのアナログ入力がサンプルされるかを選択するために、非反転および反転入力双方上にアナログマルチプレクサ（図 18-1 を参照）があります。サンプル・変換シーケンスがいったん指定されると、ADCHS ビットにより各サンプルのためにどのアナログ入力が選択されるかが決定されます。

さらに選択された入力は、サンプル毎に交互に入れ替わるか、またはサンプル毎に繰り返しシーケンス順にすることができます。

**注：** アナログ入力ピン数は入力デバイス毎に異なります。デバイスデータシートでアナログ入力ピン数などを確認してください。

### 18.8.1 アナログポートピンの構成

ADPCFG レジスタはアナログ入力として使用できるデバイスピンの入力条件を特定します。

対応している PCFGn ビット (ADPCFG<n>) がクリアされた場合、ピンはアナログ入力として構成されます。ADPCFG レジスタは、RESET でクリアされますので、アナログ入力ピンは RESET 時のデフォルトが A/D 入力ピンとなります。

アナログ入力として構成された場合、関連ポート I/O デジタル入力バッファは無効になり電力を消費しません。

ADPCFG レジスタおよび TRISB レジスタにより、A/D ポートピンの動作が制御されます。

アナログ入力として使うポートピンは、TRIS レジスタを 1 にセットして入力モードにする必要があります。A/D 入力として使う I/O ピンの TRIS ビットをクリアして出力モードとした場合、このピンはアナログモード (ADPCFG<n> = 0) にあり、このポートデジタル出力レベル (V<sub>OH</sub> または V<sub>OL</sub>) が変換されます。デバイスが RESET されると、TRIS ビットはすべてセットされます。

対応している PCFGn ビット (ADPCFG<n>) がセットされた場合、ピンはデジタル I/O として構成されます。この構成において、アナログマルチプレクサの入力は AV<sub>ss</sub> に接続されます。

**注 1:** アナログ入力として構成されたピンを、ポートとして読み込んだ場合は、常に ‘0’ として読み込まれます。  
**2:** デジタル入力として設定された (AN15:AN0 ピンを含む) ピンにアナログレベルの電圧が加わると、デバイス仕様をはずれた電力が入力バッファで消費されることがあります。

## 18.8.2 チャネル 0 入力選択

ユーザーはチャネルの正入力として、最大 16 のアナログ入力のうちの 1 つからどれでも選択できます。CH0SA<3 : 0> ビット (ADCHS<3 : 0>) により、チャネル 0 の正入力のためのアナログ入力が選択されます。

ユーザーはチャネルの負入力として、VREF- または AN1 を選択できます。CHONA ビット (ADCHS<4>) により、チャネル 0 の負入力のためのアナログ入力が選択されます。

### 18.8.2.1 交互チャネル 0 入力選択の特定

ALTS ビット (ADCON2<0>) により、一連のサンプル毎に選択された 2 つの入力の設定の間で交互に入れ替える動作とすることができます。

CH0SA<3 : 0>、CHONA、CHXSA および CHXNA<1 : 0> により特定された入力は、まとめて MUX A 入力と呼ばれます。CH0SB<3 : 0>、CHONB、CHXSB および CHXNB<1 : 0> により特定された入力は、まとめて MUX B 入力と呼ばれます。ALTS ビットが 1 の時、モジュールはサンプル毎に MUX A 入力と MUX B 入力の間を交互に入れ替えます。

チャネル 0 に関して、ALTS ビットが ‘0’ の場合、CH0SA<3 : 0> および CHONA により特定された入力だけがサンプリングのために選択されます。

ALTS ビットが ‘1’ のときは、チャネル 0 の入力は、最初のサンプリングでは、CH0SA<3 : 0> と CHONA で選択された入力となります。次のサンプリングでは、CH0SB<3:1> と CHONB で選択された入力となります。その後のサンプル変換シーケンス毎に、このパターンが繰り返されます。

### 18.8.2.2 いくつかの入力の連続スキャニング

チャネル 0 には選択されたベクトルに応じて自動スキャンする能力があります。CSCNA ビット (ADCON2<10>) により、CH0 チャネル入力を選択した入力チャネルの自動スキャンモードにします。CSCNA がセットされると、CH0SA<3 : 0> は無視されます。

ADCSSL レジスタによりスキャンされるべき入力が特定されます。ADCSSL レジスタにおける各ビットはアナログ入力に対応しています。ビット 0 は AN0 に対応し、ビット 1 は AN1 というように対応します。ADCSSL レジスタの特定のビットが ‘1’ の場合、対応する入力はスキャンシーケンスに含まれます。このスキャンは常に、割込み発生後、最初に選択されたチャネルから始まり、下位から上位へとビット番号順にスキャンされます。

**注：** 選択されたスキャン済入力の数が割込みごとに取得するサンプル数より多い場合、上位ビット番号の入力はサンプリングされません。

ADCSSL により、チャネルの正入力の入力を特定するだけです。CHONA ビットは依然としてスキャン中のチャネルの負入力の入力を選択します。

ALTS ビットが 1 ならば、このスキャニングは MUX A 入力選択にのみに適用されます。CH0SB<3 : 0> により特定された MUX B は交互入力の場合継続して使われます。この方法で入力選択がプログラムされた場合、この入力は ADCSSL レジスタにより特定されたスキャニング入力と CHOSB ビットにより特定された固定入力の間を交互に入れ替わります。

## 18.9 モジュールの有効化

ADONビット(ADCON1<15>)が‘1’の時、モジュールはアクティブモードとなり、十分に電力が供給されて動作状態です。

ADONが‘0’の時、モジュールは無効です。回路のデジタルおよびアナログ部分は最大省電力のためにオフにされます。

Offモードからアクティブモードに戻るとき、ユーザーはアナログステージが安定するのを待つ必要があります。安定化時間については、デバイスデータシートの“電気特性”セクションをご参照ください。

## 18.10 サンプリングの開始方法

### 18.10.1 手動

SAMPビット(ADCON1<1>)を設定することで、A/Dによりサンプリングが開始されます。いくつかのオプションが、サンプリングを終了して変換を完了するために使用できます。SAMPビットがもう一度設定されるまでサンプリングは再開しません。一例として図18-3をご参照ください。

### 18.10.2 自動

ASAMビット(ADCON1<2>)を設定することで、変換がそのチャネルでアクティブになっていない時はいつでも、A/Dにより自動的にチャネルのサンプリングが開始されます。いくつかのオプションが、サンプリングを終了して変換を完了するために使用できます。そのチャネルの変換の完了後にチャネルのサンプリングが再開されます。一例として図18-4をご参照ください。

## 18.11 サンプリングの停止および変換の開始方法

変換トリガーソースによりサンプリングが終了され、変換の選択済みシーケンスが開始されます。SSRC<2:0>ビット(ADCON1<7:5>)により、変換トリガーのソースが選択されます。

**注:** 利用可能な変換トリガーソースはdsPIC30Fデバイスの種類により異なります。利用可能な変換トリガーソースに関しては専用のデバイスデータシートをご参照ください。

**注:** A/Dモジュールが有効の時、SSRC選択ビットは変更できません。変換トリガーソースを変えたい場合、ADONビット(ADCON1<15>)をクリアすることにより最初にA/Dモジュールを無効にしなければなりません。

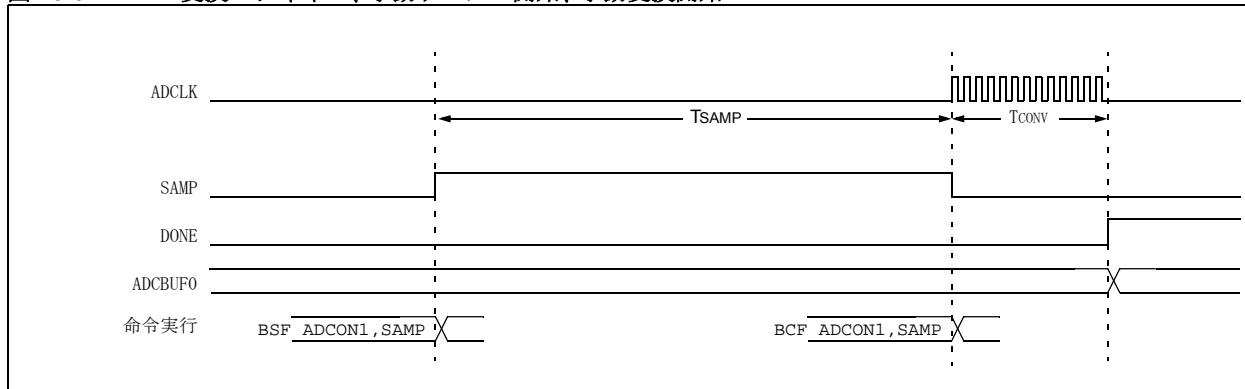
### 18.11.1 手動

$\text{SSRC}_{2:0} = 000$  の時、変換トリガーはソフトウェアの制御となります。SAMP ビット ( $\text{ADCON1}_{1:0}$ ) をクリアすることにより、変換シーケンスが開始されます。

図 18-3 は SAMP ビットをセットするとサンプリングが開始され、SAMP ビットをクリアするとサンプリングが終了して変換が開始される一例です。ユーザーソフトウェアにより、入力シグナルの十分なサンプリング時間を確保するように、SAMP ビットの設定とクリアの時間を調整する必要があります。

図 18-4 は ASAM ビットのセットにより自動サンプリングを開始し、サンプリングが終了し変換が自動開始されると SAMP ビットがクリアされる場合の例です。変換完了後、モジュールは自動的にサンプリング状態に戻ります。SAMP ビットは自動的にサンプルインターバルの開始時にセットされます。ユーザーソフトウェアにより、SAMP ビットのクリアの間の時間にはサンプリング時間と同じく変換時間も含まれていることを理解しながら、入力シグナルの十分なサンプリング時間を確保するように SAMP ビットのクリアのタイミングを調整する必要があります。

図 18-3: 変換 1 チャネル、手動サンプル開始、手動変換開始



例 18-1: チャネル 1 変換、手動サンプル開始、手動変換開始コード例

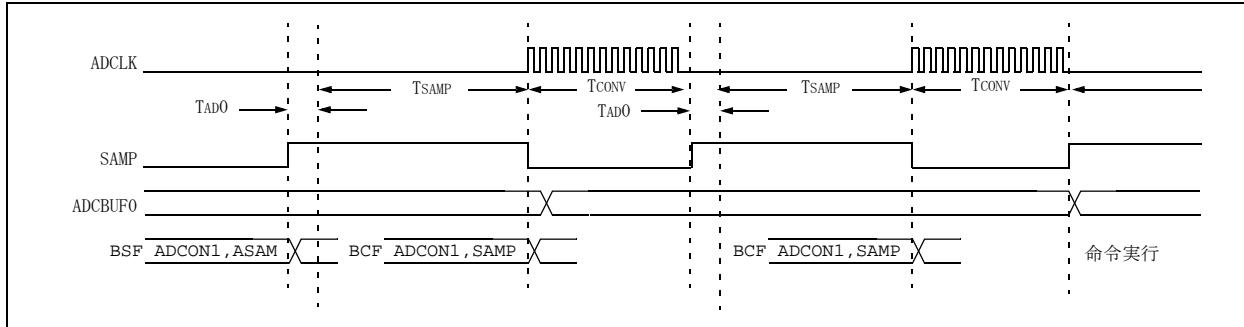
```

ADPCFG = 0xFFFF;           // all PORTB = Digital; RB2 = analog
ADCON1 = 0x0000;           // SAMP bit = 0 ends sampling ...
                           // and starts converting
ADCHS  = 0x0002;           // Connect RB2/AN2 as CH0 input ...
                           // in this example RB2/AN2 is the input
ADCSSL = 0;
ADCON3 = 0x0002;           // Manual Sample, Tad = internal 2 Tcy
ADCON2 = 0;

ADCON1bits.ADON = 1;       // turn ADC ON
while (1)                  // repeat continuously
{
    ADCON1bits.SAMP = 1;    // start sampling ...
    DelayNmSec(100);       // for 100 mS
    ADCON1bits.SAMP = 0;    // start Converting
    while (!ADCON1bits.DONE); // conversion done?
    ADCValue = ADCBUF0;    // yes then get ADC value
}
                           // repeat

```

図 18-4: 変換 1 チャネル、自動サンプル開始、マニュアル変換開始



## 18.11.2 クロックによる変換トリガ

SSRC<2:0> = 111 の時、変換トリガーは A/D クロックの制御のもとにあります。SAMC ビット (ADCON3<12 : 8>) により、サンプリングの開始と変換の開始の間に挿入される TAD クロックサイクルの数が選択されます。サンプリングの開始後、モジュールにより SAMC ビットで特定された TAD クロックの回数がカウントされます。

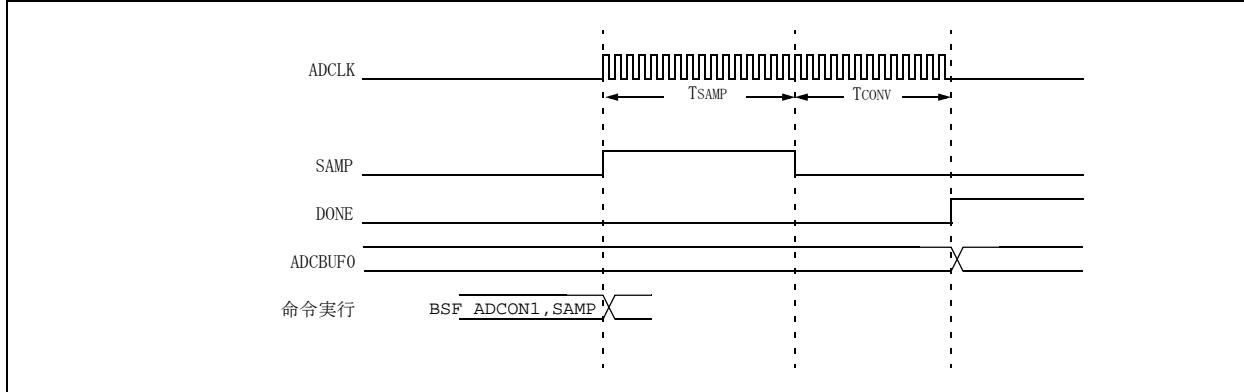
等式 18-2: 計測された変換トリガー時間

$$TSMP = SAMC<4:0> * TAD$$

サンプリング要件を満たすように、SAMC は少なくとも 1 クロック以上に設定する必要があります。

図 18-5 では、ユーザーソフトウェアによりサンプリングを開始する場合に、クロックによる変換トリガを使う方法が示されています。

図 18-5: 1 チャネル変換、手動サンプル開始、TAD クロックによる変換開始



## 例 18-2: 1 チャネル変換、手動サンプル開始、変換に基づく TAD 開始コード例

```

ADPCFG = 0xFFFF;           // all PORTB = Digital; RB12 = analog
ADCON1 = 0x00E0;           // SSRC bit = 111 implies internal
                           // counter ends sampling and starts
                           // converting.
ADCHS = 0x000C;            // Connect RB12/AN12 as CH0 input ..
                           // in this example RB12/AN12 is the input
ADCSSL = 0;
ADCON3 = 0x1F02;           // Sample time = 31Tad, Tad = internal 2
TCY
ADCON2 = 0;

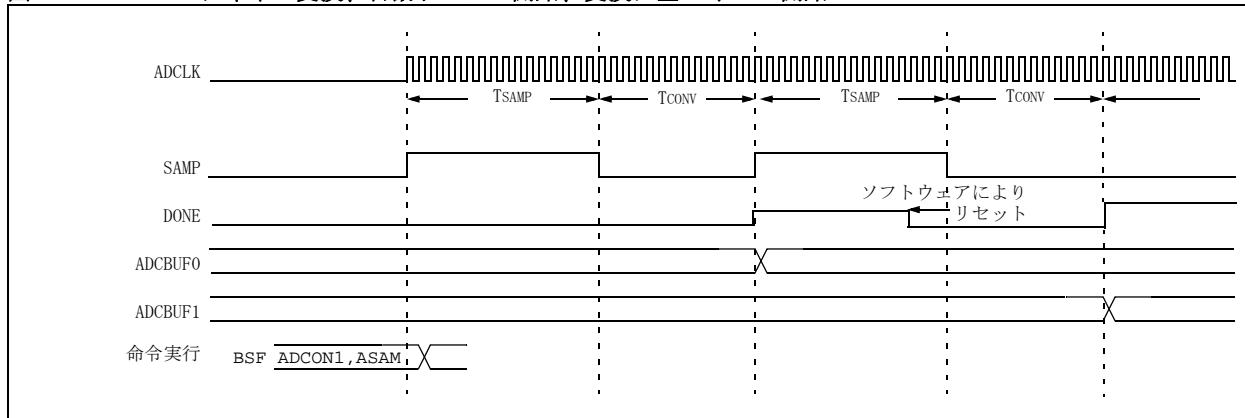
ADCON1bits.ADON = 1;       // turn ADC ON
while (1)                  // repeat continuously
{
    ADCON1bits.SAMP = 1;     // start sampling then ...
                           // after 31Tad go to conversion
    while (!ADCON1bits.DONE); // conversion done?
    ADCValue = ADCBUF0;      // yes then get ADC value
}                           // repeat// repeat

```

### 18.11.2.1 フリーランサンプル変換シーケンス

図 18-6 に示されるように、自動変換の変換トリガーモード (SSRC = 111) を自動サンプル開始モード (ASAM = 1) と組み合わせて使用することにより、A/D モジュールをユーザーまたは他のデバイスリソースによる割込みがまったく必要なサンプル・変換シーケンスとすることができます。この「クロックされた」モードにより、モジュール初期化後の継続データ収集が可能になります。

### 図 18-6: 1 チャネル変換、自動サンプル開始、変換に基づく TAD 開始



### 18.11.2.2 クロックされた変換トリガーおよび自動サンプリング使用を考慮に入れたサンプル時間

ユーザーはサンプリング時間がセクション 18.15 「A/D サンプリング要件」で、概要を説明しているサンプリング要件を越えていることを確認する必要があります。

モジュールが自動サンプリングに設定された場合、クロックされた変換トリガーを使用すると、サンプリングインターバルは SAMC ビットにより特定されます。

### 18.11.3 イベントトリガー変換開始

サンプリングの終了と変換の開始を他のイベントトリガと同期化させることができます。変換トリガーイベントとして、3 つのソースのうちの 1 つが A/D により使用できます。

### 18.11.3.1 外部 INT ピントリガー

SSRC<2:0> = 001 の時、A/D 変換は INTO 上のアクティブな遷移でトリガれます。INTO は立ち上がり入力または両方として設定される必要があります。

### 18.11.3.2 汎用タイマー比較一致トリガー

A/D は SSRC<2:0> = 010 設定により、このトリガー モードに構成されます。32 ビットタイマー TMR3/TMR2 と 32 ビットの結合周期レジスタ PR3/PR2 との間で一致が発生すると、特別な ADC トリガーイベントシグナルがタイマー 3 により生成されます。TMR5/TMR4 タイマーペアにはこの特徴はありません。詳細は第 12 章 . 「タイマー」をご参照ください。

### 18.11.3.3 モーター制御 PWM トリガー

PWM モジュールには A/D 変換を PWM 時間ベースに同期化できるイベントトリガーが 1 つあります。SSRC<2:0> = 011 の時、A/D サンプリングおよび変換時間を、PWM 周期内のどのポイントにもユーザーが設定できます。ユーザーは特別イベントトリガーにより、A/D 変換結果獲得時の時間と負荷デューティアップデート時の間の遅延時間を最小にすることができます。詳細は第 15 章 . 「モーター制御 PWM」をご参照ください。

### 18.11.3.4 内蔵イベントまたは外部イベントへの A/D 動作の同期化

外部イベントトリガーパルスによりサンプリングが終了し、変換が開始されるモード (SSRC = 001, 010, 011) は、トリガーパルスソースに A/D のサンプル変換イベントが同期化されるように、自動サンプリング (ASAM = 1) と組み合わせて使用できます。例えば SSRC = 010, ASAM = 1 の図 18-8 では、A/D は常にサンプリングを終了し、タイマー比較トリガーエベントと同期化して変換を開始します。A/D 変換レートはタイマ比較一致イベントの周期によって決まることになります。

図 18-7: 手動サンプル開始、変換トリガの場合

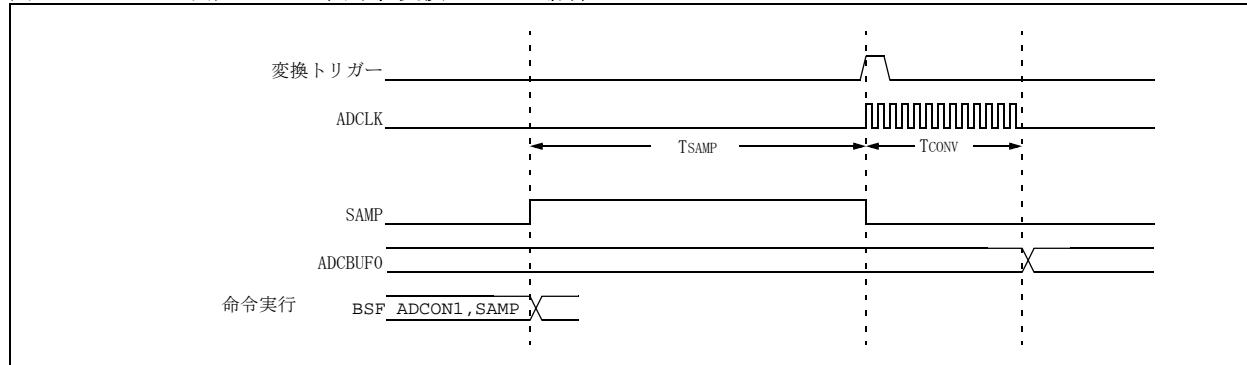
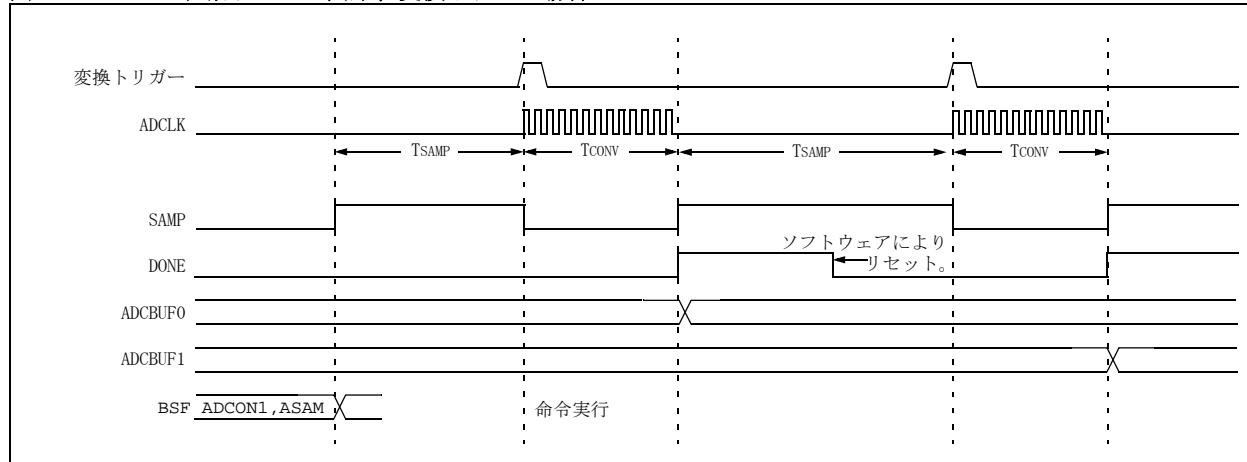


図 18-8: 自動サンプル開始、変換トリガの場合



## 18.11.3.5 自動サンプリング・変換シーケンスのサンプル時間に関する留意事項

サンプル・変換シーケンスが異なると、アナログ信号を入力するための S/H チャネルのサンプリング時間も異なってきます。ユーザーはサンプリング時間がセクション 18.15 「A/D サンプリング要件」で概要を説明している、サンプリング要件より大きくなるようにする必要があります。

モジュールが自動サンプリング用に設定されていること、外部トリガーパルスが変換トリガーとして使用されていることを前提とするので、サンプリングインターバルはトリガーパルス周期の一部です。

サンプリング時間はトリガーパルス周期で、完全な変換に必要な時間よりも短い時間です。

### 等式 18-3: 利用可能なサンプリング時間、連続サンプリング

$$TSMP = \text{Trigger Pulse Interval (TSEQ)} - \text{Conversion Time (TCONV)}$$

$$TSMP = TSEQ - TCONV$$

注: TSEQ はトリガーパルスインターバル時間です。

## 18.12 サンプル・変換動作の制御

アプリケーションソフトウェアにより、A/D 変換動作を続けるために、SAMP および CONV ビットの状態をポーリングするか、または変換完了時にモジュールによる CPU への割込みが可能です。必要に応じて、アプリケーションソフトウェアによっても A/D 変換動作を中断することができます。

### 18.12.1 サンプル・変換ステータスのモニタリング

SAMP (ADCON1<1>) および CONV (ADCON1<0>) により、サンプリングの状態と A/D の変換状態がそれぞれ示されます。一般に、SAMP ビットがクリアされてサンプリングが終了した場合、CONV ビットが自動的にセットされて変換が開始されます。SAMP と CONV の両方が ‘0’ ならば、A/D は非アクティブ状態です。いくつかの動作モードで、SAMP ビットでサンプリングを開始したり終了したりでき、CONV ビットで変換を終了することもできます。

### 18.12.2 A/D 割込みの生成

SMPI<3 : 0> により割込みの生成が制御されます。割り込みを一定回数のサンプリング / 変換シーケンスの後発生させることができ、以降同じ回数ごとに割り込みを発生させることができます。

SMPI に設定される値は、最大 16 までのバッファのデータサンプルの数に対応します。

A/D 割込みの無効化は SMPI ビットでは行われません。割込みを無効化するためには、ADIE アナログモジュール割込み有効化ビットをクリアしてください。

### 18.12.3 サンプリングの中断

手動サンプリングモード中に SAMP をクリアすることによりサンプリングが終了されますが、SSRC = 000 ならば変換が開始されます。

自動サンプリングモード中に ASAM ビットをクリアすることによって、継続中のサンプル・変換シーケンスは終了させられません。ただし、その変換後にサンプリングが自動的に再開されることはありません。

### 18.12.4 変換の中断

変換中に ADON ビットをクリアすることにより、現在の変換が中断されます。A/D 結果レジスタペアは、部分的に完了された A/D 変換サンプルではアップデートされません。つまり、対応している ADCBUF バッファ位置には、最後に完了された変換データ（またはバッファに書き込まれた最新の値）が継続維持されます。

### 18.13 変換結果をバッファに書き込む方法の指定

変換が完了すると、モジュールにより A/D 結果バッファに変換の結果が書き込まれます。このバッファは 16 の 12 ビットワードの RAM アレイです。バッファは ADCBUFO...ADCBUFF と名付けられた SFR スペースの中の 16 のアドレス位置によってアクセスされます。

ユーザーソフトウェアにより、毎回の A/D 変換毎に結果を読み出すことができますが、こうすると多くの CPU 時間を消費してしまいます。一般的にコードを簡単にするために、モジュールによりバッファが結果で満たされていき、バッファが一杯になったとき割込みが生成されます。

#### 18.13.1 割込みごとの変換の数

**SMPI<3 : 0>** ビット (ADCON2<5 : 2>) により、CPU が割り込まれる前に何回の A/D 変換が実行されるかを指定できます。1 つの割込みごとに 1 サンプルから 16 サンプルまで設定可能です。各割込みの後に、A/D コンバータモジュールにより、常にバッファの始めから変換結果の書き込みが開始されます。例えば、**SMPI<3 : 0>= 0000** ならば、変換結果は常に ADCBUFO に書き込まれます。この例では、他のバッファ位置は一切使用されません。

#### 18.13.2 バッファサイズによる制限

**BUFM** ビット (ADCON2<1>) が ‘1’ の場合、ユーザーは割込みごとに 8 を越える値に **SMPI** を設定できません。BUFM ビット機能については次に説明します。

#### 18.13.3 バッファファイルモード

BUFM ビット (ADCON2<1>) が ‘1’ の時、16 ワードの結果バッファ (ADRES) は 2 つの 8 ワードのグループに分けられます。各割込みイベント毎に、交互に 8 ビットバッファに書き込まれます。BUFM が設定されたあとに使用される最初の 8 ワードバッファは、ADCBUF の下位アドレスとなります。BUFM が ‘0’ の時、16 ワードバッファ全体がすべての変換シーケンスのために使用されます。

BUFM 機能を使用するという決定はアプリケーションにより決定され、割込み後のバッファ情報を動かすためにどれぐらいの時間かかるかによります。1 つのチャネルをサンプルして変換するのにかかる時間内に、プロセッサが全バッファを解放できるなら、BUFM は ‘0’ とすることができ、割込みごとに最大 16 の変換を実行できます。プロセッサには最初のバッファ位置が上書きされる前に、1 個のサンプル変換時間しか与えられません。

サンプル・変換時間内にプロセッサによりバッファが解放できない場合は、BUFM を ‘1’ とします。例えば **SMPI<3 : 0> = 0111** ならば 8 つの変換データをバッファの 1/2 の領域に格納した後、割り込みが発生します。次の 8 つの変換データはバッファの残りの 1/2 に格納されます。これで、プロセッサは、バッファの 8 個のデータを取り出すために、割り込み間の全時間を使うことができます。

#### 18.13.4 バッファファイルステータス

変換結果バッファが BUFM 制御ビットを使って分けられる時、BUFS ステータスビット (ADCON2<7>) により、A/D コンバータが現在、半分のバッファのどちらを使っているかを表します。BUFS=0 ならば、A/D コンバータにより ADCBUFO-ADCBUF7 が使われており、ユーザーソフトウェアにより ADCBUF8-ADCBUFF から変換値を読み出します。BUFS=1 ならば状況は逆で、ユーザーソフトウェアにより ADCBUFO-ADCBUF7 から変換値を読み出します。

## 18.14 変換シーケンス例

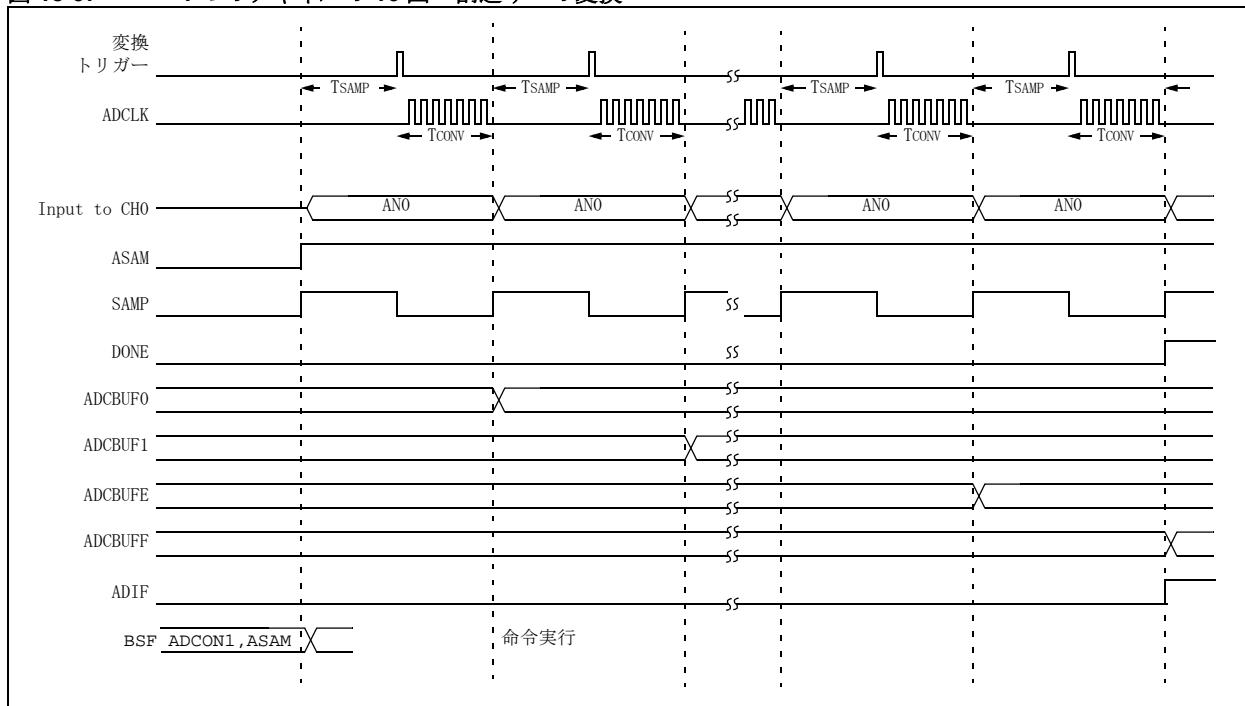
以下の構成サンプルには、異なるサンプリングおよびバッファ構成における A/D 動作が示されています。各事例では、ASAM ビット設定により自動サンプリングが開始されます。変換トリガーによりサンプリングが終了され、変換が開始されます。

### 18.14.1 例：シングルチャネル複数回のサンプリングおよび変換

図 18-9 および表 18-2 では A/D の基本的構成が説明されています。この場合、1 つの A/D 入力である AN0 がサンプリングされ、変換されます。結果は ADCBUF バッファに格納されます。このプロセスは、バッファが一杯になりモジュールにより割込みが生成されるまで 16 回繰り返されます。それから全プロセスが繰り返されます。

ALTS をクリアすることで、MUX A 入力のみがアクティブになります。CHOSA ビットおよび CHONA ビットはサンプル・ホールドチャネルへの入力として特定されます (ANO-VREF-)。他の入力選択ビットはすべて使用されません。

図 18-9: 1 つのチャネルの 16 回・割込みへの変換



## 例 18-3: シングルチャネル複数回のサンプリングおよび変換のコード例

```

ADPCFG = 0xFFFFB;           // all PORTB = Digital; RB2 = analog
ADCON1 = 0x00E0;            // SSRC bit = 111 implies internal
                            // counter ends sampling and starts
                            // converting.
ADCHS = 0x0002;             // Connect RB2/AN2 as CH0 input ..
                            // in this example RB2/AN2 is the input
ADCSSL = 0;                 // Sample time = 15Tad, Tad = internal Tcy/2
ADCON3 = 0x0F00;             // Interrupt after every 16 samples
ADCON2 = 0x003C;

ADCON1bits.ADON = 1;         // turn ADC ON
while (1)                   // repeat continuously
{
    ADCValue = 0;             // clear value
    ADC16Ptr = &ADCBUF0;        // initialize ADCBUF pointer
    IFS0bits.ADIF = 0;         // clear ADC interrupt flag
    ADCON1bits.ASAM = 1;        // auto start sampling
                            // for 31Tad then go to conversion
    while (!IFS0bits.ADIF);    // conversion done?
    ADCON1bits.ASAM = 0;        // yes then stop sample/convert
    for (count = 0; count < 16; count++) // average the 16 ADC value
        ADCValue = ADCValue + *ADC16Ptr++;
    ADCValue = ADCValue >> 4;   // repeat
}

```

# dsPIC30F ファミリーリファレンスマニュアル

表 18-2: 1つのチャネルの 16 回／割込みの変換

制御ビット シーケンス選択		動作シーケンス
SMPI<2:0> = 1111	16 回目のサンプルで割り込み	サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ 0x0 書込み
BUFM = 0	单一 16 ワード結果バッファ	サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ 0x1 書込み
ALTS = 0	常に MUX A 入力選択を使用	サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ 0x2 書込み
<b>MUX A 入力選択</b>		サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ 0x3 書込み
CH0SA<3:0> = 0000	CH0+ 入力には AN0 を選択	サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ 0x4 書込み
CH0NA = 0	CH0- 入力には VREF- を選択	サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ 0x5 書込み
CSCNA = 0	入力スキヤンなし	サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ 0x6 書込み
CSSL<15:0> = n/a	スキヤン入力選択未使用	サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ 0x7 書込み
<b>MUX B 入力選択</b>		サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ 0x8 書込み
CH0SB<3:0> = n/a	チャネル CH0+ 入力未使用	サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ 0x9 書込み
CH0NB = n/a	チャネル CH0- 入力未使用	サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ OxA 書込み
		サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ OxB 書込み
		サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ OxC 書込み
		サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ OxD 書込み
		サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ OxE 書込み
		サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ OxF 書込み
		割込み
		繰返し

バッファ アドレス	バッファ @ 1回目割込み
ADCBUF0	AN0 サンプル 1
ADCBUF1	AN0 サンプル 2
ADCBUF2	AN0 サンプル 3
ADCBUF3	AN0 サンプル 4
ADCBUF4	AN0 サンプル 5
ADCBUF5	AN0 サンプル 6
ADCBUF6	AN0 サンプル 7
ADCBUF7	AN0 サンプル 8
ADCBUF8	AN0 サンプル 9
ADCBUF9	AN0 サンプル 10
ADCBUFA	AN0 サンプル 11
ADCBUFB	AN0 サンプル 12
ADCBUFC	AN0 サンプル 13
ADCBUFD	AN0 サンプル 14
ADCBUFE	AN0 サンプル 15
ADCBUFF	AN0 サンプル 16

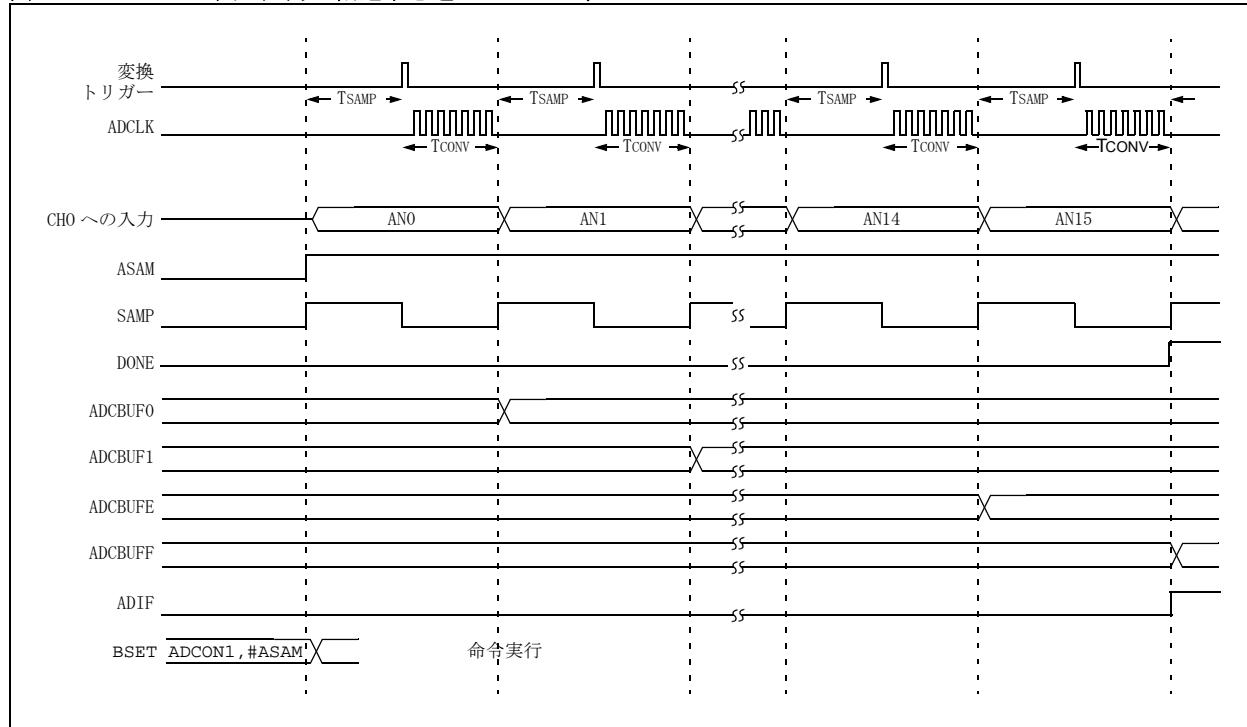
バッファ @ 2回目割込み
AN0 サンプル 17
AN0 サンプル 18
AN0 サンプル 19
AN0 サンプル 20
AN0 サンプル 21
AN0 サンプル 22
AN0 サンプル 23
AN0 サンプル 24
AN0 サンプル 25
AN0 サンプル 26
AN0 サンプル 27
AN0 サンプル 28
AN0 サンプル 29
AN0 サンプル 30
AN0 サンプル 31
AN0 サンプル 32

## 18.14.2 例すべてのアナログ入力スキャンモードの A/D 変換

図 18-10 および表 18-3 には、すべての利用可能なアナログ入力チャネルがサンプリングされて変換される典型的なセットアップが説明されています。CSCNA をセットすることにより、CHO 正入力に対する A/D 入力のスキャニングモードが設定されます。他の条件はサブセクション 18.14.1 と類似しています。

最初に、ANO 入力が CHO によりサンプルされて変換されます。結果は ADCBUF バッファに格納されます。次に AN1 入力がサンプルされて変換されます。この入力のスキャニングのプロセスは、バッファが一杯になり、モジュールにより割込みが生成されるまで 16 回繰り返されます。それから全プロセスが繰り返されます。

図 18-10: 16 回の入力・割込みを通じてのスキャニング



# dsPIC30F ファミリーリファレンスマニュアル

表 18-3: 16 回の入力・割込みを通じてのスキャンニング

制御ビット シーケンス選択		動作シーケンス
SMPI<2:0> = 1111	16 回目のサンプルで割り込み	サンプル MUX A 入力 : AN0 -> CH0 変換 CH0, バッファ 0x0 書込み
BUFM = 0	单一 16 ワード結果バッファ	サンプル MUX A 入力 : AN1 -> CH0 変換 CH0, バッファ 0x1 書込み
ALTS = 0	常に MUX A 入力選択を使用	サンプル MUX A 入力 : AN2 -> CH0 変換 CH0, バッファ 0x2 書込み
<b>MUX A 入力選択</b>		サンプル MUX A 入力 : AN3 -> CH0 変換 CH0, バッファ 0x3 書込み
CH0SA<3:0> = n/a	CSCNA によってオーバライドされた	サンプル MUX A 入力 : AN4 -> CH0 変換 CH0, バッファ 0x4 書込み
CH0NA = 0	CH0- 入力には VREF- を選択	サンプル MUX A 入力 : AN5 -> CH0 変換 CH0, バッファ 0x5 書込み
CSCNA = 1	スキャン CH0+ Inputs	サンプル MUX A 入力 : AN6 -> CH0 変換 CH0, バッファ 0x6 書込み
CSSL<15:0> = 1111 1111 1111 1111	全ての入力をスキャン	サンプル MUX A 入力 : AN7 -> CH0 変換 CH0, バッファ 0x7 書込み
<b>MUX B 入力選択</b>		サンプル MUX A 入力 : AN8 -> CH0 変換 CH0, バッファ 0x8 書込み
CH0SB<3:0> = n/a	チャネル CH0+ 入力未使用	サンプル MUX A 入力 : AN9 -> CH0 変換 CH0, バッファ 0x9 書込み
CH0NB = n/a	チャネル CH0- 入力未使用	サンプル MUX A 入力 : AN10 -> CH0 変換 CH0, バッファ 0xA 書込み
		サンプル MUX A 入力 : AN11 -> CH0 変換 CH0, バッファ 0xB 書込み
		サンプル MUX A 入力 : AN12 -> CH0 変換 CH0, バッファ 0xC 書込み
		サンプル MUX A 入力 : AN13 -> CH0 変換 CH0, バッファ 0xD 書込み
		サンプル MUX A 入力 : AN14 -> CH0 変換 CH0, バッファ 0xE 書込み
		サンプル MUX A 入力 : AN15 -> CH0 変換 CH0, バッファ 0xF 書込み
		割込み
		繰返し

バッファ アドレス	バッファ @ 1回目割込み
ADCBUF0	AN0 サンプル 1
ADCBUF1	AN1 サンプル 2
ADCBUF2	AN2 サンプル 3
ADCBUF3	AN3 サンプル 4
ADCBUF4	AN4 サンプル 5
ADCBUF5	AN5 サンプル 6
ADCBUF6	AN6 サンプル 7
ADCBUF7	AN7 サンプル 8
ADCBUF8	AN8 サンプル 9
ADCBUF9	AN9 サンプル 10
ADCBUFA	AN10 サンプル 11
ADCBUFB	AN11 サンプル 12
ADCBUFC	AN12 サンプル 13
ADCBUFD	AN13 サンプル 14
ADCBUFE	AN14 サンプル 15
ADCBUFF	AN15 サンプル 16

バッファ @ 2回目割込み
AN0 サンプル 17
AN1 サンプル 18
AN2 サンプル 19
AN3 サンプル 20
AN4 サンプル 21
AN5 サンプル 22
AN6 サンプル 23
AN7 サンプル 24
AN8 サンプル 25
AN9 サンプル 26
AN10 サンプル 27
AN11 サンプル 28
AN12 サンプル 29
AN13 サンプル 30
AN14 サンプル 31
AN15 サンプル 32

• • •

### 18.14.3 例：デュアル8ワードバッファの使用

デュアルバッファ使用例については、第17章「10ビットA/Dコンバータ」のサブセクション17.15.4をご参照ください。

### 18.14.4 例：交互MUX A、MUX B入力選択の使用

MUX AおよびMUX B選択の使用例については、第17章「10ビットA/Dコンバータ」のサブセクション17.15.5をご参照ください。

## 18.15 A/Dサンプリング要件

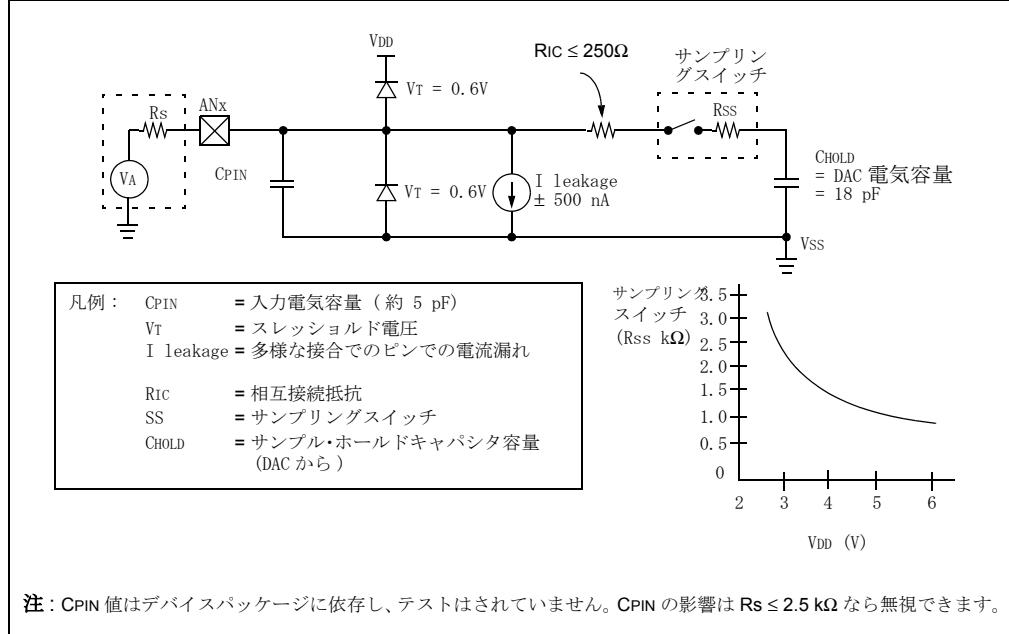
A/Dのためのトータルサンプリング時間は、式18-4に示されるように、アンプのセトリング時間とホールドキャパシタの充電時間と温度の関数となります。

12ビットA/Dコンバータのアナログ入力モデルは図18-11に示されています。仕様の精度を保証する変換のために、コンデンサ(**C<sub>HOLD</sub>**)がアナログ入力ピンの電圧レベルまで十分充電が必要です。インピーダンス(**R<sub>S</sub>**)および内蔵サンプリングスイッチインピーダンス(**R<sub>SS</sub>**)が組み合わさって、コンデンサ**C<sub>HOLD</sub>**を充電するために必要な時間に直接影響します。さらに、サンプリングスイッチインピーダンス(**R<sub>SS</sub>**)は、図18-11に示されるように、デバイス電圧(**V<sub>DD</sub>**)とともに変化します。そこで、組み合わせられたアナログソースインピーダンス(**R<sub>S</sub>, R<sub>IC</sub>**)とサンプリングスイッチインピーダンス(**R<sub>SS</sub>**)は、選択されたサンプル時間内にホールドコンデンサが完全に充電できるように、十分小さいものである必要があります。A/Dコンバータの正確さに影響するピンの電流漏れを最小にするために、最大の推奨インピーダンス(**R<sub>S</sub>**)は2.5 kΩです(充電時間計算結果には依りません)。アナログ入力チャネルが選択(変更)されたのちに、このサンプリング機能は変換開始に先立って完了される必要があります。内部ホールドコンデンサは、各サンプル操作に先立ち放電された状態になります。

サンプリングコンデンサの充電時間を計算するには、式18-5が使用できます。サンプリングコンデンサは各変換の後に放電されるので、この式によりアナログ入力がOVからアナログ入力電圧との誤差がLSBのn倍のレベルまで進むと仮定され、また出力が1/2 LSBエラー内に入ると仮定されています(すなわち12ビットA/Dには8192段階必要です)。1/2 LSBエラーはA/Dが特定の解像度で許容される最大限度のエラーです。A/Dのための**C<sub>HOLD</sub>**は18 pFです。

トータルのサンプリング時間を計算するために、増幅器セトリング時間、T<sub>AMP</sub>は0.5 μsecと仮定されています。温度計数は式18-6により求められます。温度条件による影響は25°C未満の温度で0です。

図18-11: 12ビットA/Dコンバータアナログ入力モデル



## 式 18-4: サンプル時間

$$\begin{aligned} TSMP &= \text{増幅器セトリング時間 (TAMP)} + \\ &\quad \text{ホールドコンデンサ充電時間 (Tc)} + \\ &\quad \text{温度計数 (TCOFF)} \\ TSMP &= TAMP + TC + TCOFF \end{aligned}$$

注: TAMP is 0.5  $\mu$ s.

## 式 18-5: A/D ホールドコンデンサ充電時間

$$Tc = -CHOLD (RIC + RSS + RS) \ln(1/2n) \text{ secs}$$

## 式 18-6: 温度係数による時間

$$TCOFF = (Temp - 25^\circ C)(0.05 \mu s / ^\circ C)$$

注: TCOFF は  $25^\circ C$  未満のすべての温度に対して 0 です。

例 18-4 では、特定温度での最短のサンプル時間を  $R_s$  の関数として計算しています。この計算は以下のシステム仮定に基づいています。

1. CHOLD = 18 pF
2. RIC = 250 $\Omega$
3. RS = 1 $\Omega$  (Op Amp でドライブされているとする)
4. VI = 4095 LSB (最大スケール入力電圧); n = 4096
5. VDD = 5V  $\rightarrow$  RSS = 1.2 k $\Omega$
6. 温度 (公称) =  $25^\circ C$

## 例 18-4: 最良の場合のサンプル時間計算

$$\begin{aligned} TC &= -CHOLD (RIC + RSS + RS) \ln(1/2n) \\ &= -18 \text{ pF} (250\Omega + 1.2 \text{ k}\Omega + 1\Omega) \ln(1/8192) \\ &= 0.24 \mu s \end{aligned}$$

$$\begin{aligned} TSMP &= TAMP + TC + TCOFF \\ &= 0.5 \mu s + 0.24 \mu s + [(25^\circ C - 25^\circ C)(0.05 \mu s / ^\circ C)] \\ &= 0.74 \mu s \end{aligned}$$

例 18-5 には最悪のケースのサンプル時間の計算が示されています。この計算は以下のシステム仮定に基づいています。

1. CHOLD = 18 pF
2. RS = 2.5 kΩ (抵抗センサーによってドライブされているとする。)
3. VI = 4095 LSB (最大スケール入力電圧) n = 4096
4. VDD = 5V → RSS = 1.2 kΩ
5. 温度 (公称) = 25°C

#### 例 18-5: 最悪の場合のサンプル時間計算、最大ソースインピーダンス

$$\begin{aligned} TC &= -CHOLD (RIC + RSS + RS) \ln(1/2n) \\ &= -18 \text{ pF} (250\Omega + 1.2 \text{ k}\Omega + 2.5 \text{ k}\Omega) \ln(1/8192) \\ &= 0.641 \mu\text{s} \end{aligned}$$

$$\begin{aligned} TSMP &= TAMP + TC + TCOFF \\ &= 0.5 \mu\text{s} + 1.66 \mu\text{s} + [(25^\circ\text{C} - 25^\circ\text{C})(0.05 \mu\text{s}/^\circ\text{C})] \\ &= 1.14 \mu\text{s} \end{aligned}$$

例 18-6 では、例 18-4 同じ条件を使った（温度以外で）サンプル時間に対する高温の影響が示されています。他の規準と比べて、温度計数はサンプル時間に対して一番大きな影響を与えます。この計算は以下のシステム仮定に基づいています。

1. CHOLD = 18 pF
2. RIC = 250Ω
3. RS = 1Ω (Op Amp によりドライブされます)
4. VI = 4095 LSB (全スケール入力電圧) ; n = 4096
5. VDD = 5V → RSS = 1.2 kΩ
6. 温度 = 85°C

#### 例 18-6: サンプル時間計算の最悪ケース、高温

$$\begin{aligned} TC &= -CHOLD (RIC + RSS + RS) \ln(1/2n) \\ &= -18 \text{ pF} (250\Omega + 1.2 \text{ k}\Omega + 1\Omega) \ln(1/8192) \\ &= 0.24 \mu\text{s} \end{aligned}$$

$$\begin{aligned} TSMP &= TAMP + TC + TCOFF \\ &= 0.5 \mu\text{s} + 0.24 \mu\text{s} + [(85^\circ\text{C} - 25^\circ\text{C})(0.05 \mu\text{s}/^\circ\text{C})] \\ &= 0.5 \mu\text{s} + 0.24 \mu\text{s} + 3 \mu\text{s} \\ &= 3.74 \mu\text{s} \end{aligned}$$

## 18.16 A/D 結果バッファの読み込み

RAM は 12 ビット幅です、ただしバッファからの読み込みが実行されると、データは自動的に選択可能フォーマットの 1 つにフォーマットされます。フォーム  $\text{**1:0>}**$  ビット ( $\text{ADCON1<9:8>}$ ) によりフォーマットが選択されます。フォーマットをしているハードウェアにより、すべてのデータフォーマットでデータバス上に 16 ビットの結果がもたらされます。図 18-12 では、 $\text{FORM<1:0>}$  制御ビットで選択できるデータ出力フォーマットが示されています。

図 18-12: A/D 出力データフォーマット

RAM コンテンツ:	<table border="1"><tr><td>d11</td><td>d10</td><td>d09</td><td>d08</td><td>d07</td><td>d06</td><td>d05</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td></tr></table>	d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00				
d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00						
バスへの読み込み:																	
整数	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>d11</td><td>d10</td><td>d09</td><td>d08</td><td>d07</td><td>d06</td><td>d05</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td></tr></table>	0	0	0	0	d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00
0	0	0	0	d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00		
符号付整数	<table border="1"><tr><td>d11</td><td>d11</td><td>d11</td><td>d11</td><td>d11</td><td>d10</td><td>d09</td><td>d08</td><td>d07</td><td>d06</td><td>d05</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td></tr></table>	d11	d11	d11	d11	d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00
d11	d11	d11	d11	d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00		
固定小数	<table border="1"><tr><td>d11</td><td>d10</td><td>d09</td><td>d08</td><td>d07</td><td>d06</td><td>d05</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0
d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0		
符号付固定小数 (1.15)	<table border="1"><tr><td><math>\overline{d11}</math></td><td>d10</td><td>d09</td><td>d08</td><td>d07</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td><td>d01</td><td>d00</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	$\overline{d11}$	d10	d09	d08	d07	d04	d03	d02	d01	d00	d01	d00	0	0	0	0
$\overline{d11}$	d10	d09	d08	d07	d04	d03	d02	d01	d00	d01	d00	0	0	0	0		

表 18-4: 種々の結果コードの表現方法

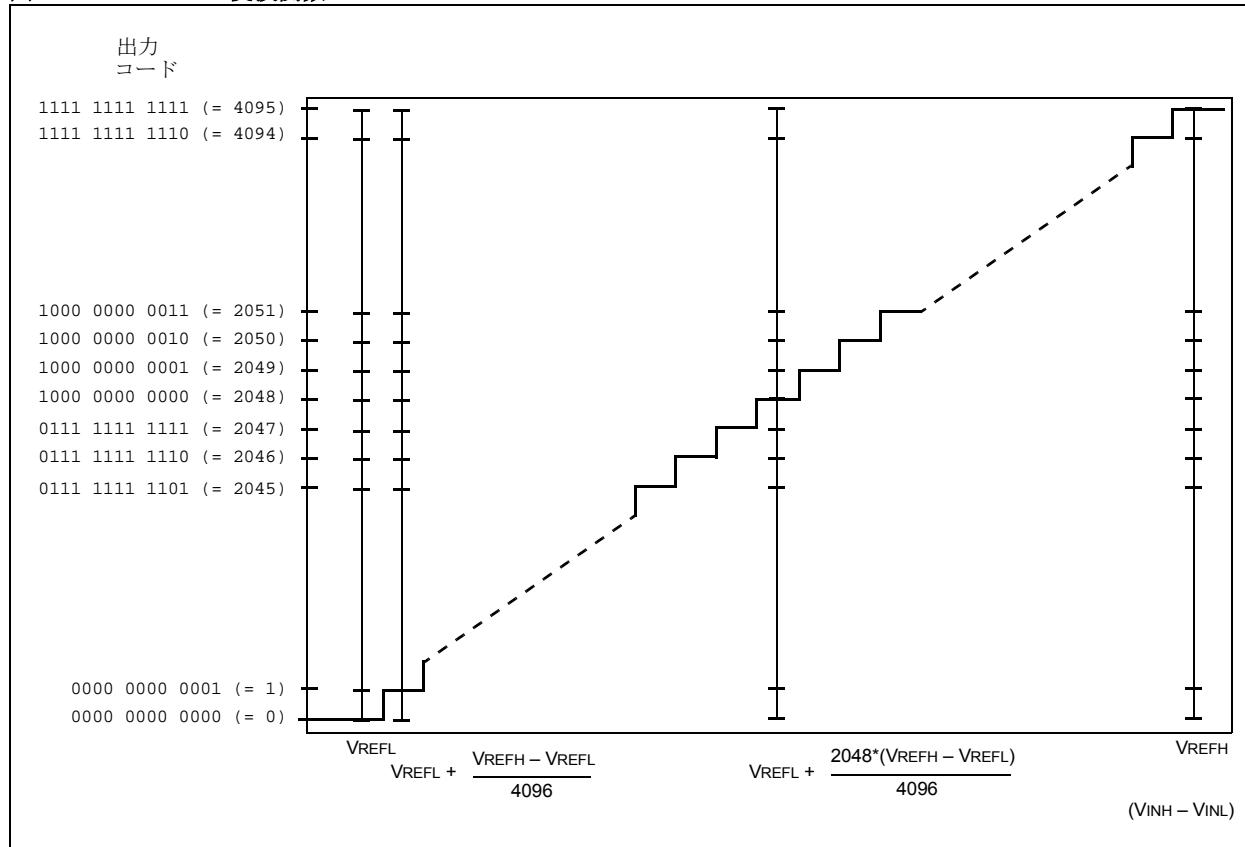
VIN/VREF	12- ビット 出力コード	16 ビット符号付 整数フォーマット	16 ビット符号付 整数フォーマット	16 ビット符号なし 固定小数フォーマット	16 ビット符号付 固定小数フォーマット
4095/4096	1111 1111 1111	0000 1111 1111 1111 = 4095	0000 0111 1111 1111 = 2047	1111 1111 1111 0000 = 0.9998	0111 1111 1111 0000 = 0.9995
4094/4096	1111 1111 1110	0000 1111 1111 1110 = 4094	0000 0111 1111 1110 = 2046	1111 1111 1110 0000 = 0.9995	0111 1111 1110 0000 = 0.9990
•••					
2049/4096	1000 0000 0001	0000 1000 0000 0001 = 2049	0000 0000 0000 0001 = 1	1000 0000 0001 0000 = 0.5002	0000 0000 0001 0000 = 0.0005
2048/4096	1000 0000 0000	0000 1000 0000 0000 = 2048	0000 0000 0000 0000 = 0	1000 0000 0000 0000 = 0.500	0000 0000 0000 0000 = 0.000
2047/4096	0111 1111 1111	0000 0111 1111 1111 = 2047	1111 1111 1111 1111 = -1	0111 1111 1111 0000 = 0.4998	1111 1111 1111 0000 = -0.0005
•••					
1/4096	0000 0000 0001	0000 0000 0000 0001 = 1	1111 1000 0000 0001 = -2047	0000 0000 0001 0000 = 0.0002	1000 0000 0001 0000 = -0.9995
0/4096	0000 0000 0000	0000 0000 0000 0000 = 0	1111 1000 0000 0000 = -2048	0000 0000 0000 0000 = 0.000	1000 0000 0000 0000 = -1.000

### 18.17 変換関数

A/D コンバータの理想的な変換関数は図 18-13 に示されています。入力電圧 ( $V_{INH} - V_{INL}$ ) の差分がリファレンス ( $V_{REFH} - V_{REFL}$ ) と比較されます。

- 入力電圧が ( $V_{REFH} - V_{REFL}/8192$ ) または 0.5 LSb の時、最初のコード遷移が発生します。
- 00 0000 0001 コードは ( $V_{REFH} - V_{REFL}/4096$ ) または 1.0 LSb を中心としています。
- 10 0000 0000 コードは ( $2048*(V_{REFH} - V_{REFL})/4096$ ) を中心としています。
- $(1*(V_{REFH} - V_{REFL})/8192)$  未満の入力電圧は 00 0000 0000 として変換されます。
- $(8192*(V_{REFH} - V_{REFL})/8192)$  を越える入力は 11 1111 1111 として変換されます。

図 18-13: A/D 変換関数



### 18.18 A/D の精度と誤差

A/D の正確性を論じる文書のリストについては、セクション 18.25 「関連するアプリケーションノート」をご参照ください。

### 18.19 接続条件

アナログ入力が ESD プロテクションを採用しているので、 $V_{DD}$  および  $V_{SS}$  に対してダイオードがあります。このことにより、アナログ入力は  $V_{DD}$  と  $V_{SS}$  の間である必要があります。この入力電力が 0.3V 以上この範囲を上回ると、ダイオードの 1 つが順方向にバイアスがかかり、入力電流仕様を上回る電流が流れるとデバイスがダメージを受ける可能性があります。

入力シグナルのアンチエイリアス処理のために、外部 RC フィルタが時折追加されます。R コンポーネントは、サンプリング時間要件が満たされるように選択する必要があります。アナログ入力ピンに接続される（高抵抗経由でも）すべての外部コンポーネント（コンデンサ、定電圧ダイオードなど）は、漏れ電流が非常に少ないものを使う必要があります。

## 18.20 初期化

A/D モジュールの単純初期化コード例は例 18-7 に示されています。

この特別な構成において、16 のアナログ入力ピンのすべて、AN0-AN15 は、アナログ入力として設定されます。IDLE モードでの操作は無効であり、出力データは符号なし固定小数フォーマットに指定され、AVDD および AVSS が、VREFH および VREFL のために使用されます。変換(変換トリガー)の開始とサンプリングの開始はソフトウェアで手動で実行されます。入力のスキヤニングは無効で、各サンプル・変換シーケンス(1 変換結果)ごとに割込みが発生します。A/D 変換クロックは Tcy/2 です。ANO が変換されます。

各変換が完了した後に、サンプリングは SAMP ビット (ADCON1<1>) の設定により手動で開始されるので、自動サンプル時間ビット、SAMC<4:0> (ADCON3<12:8>) は無視されます。その上、変換の開始(すなわちサンプリングの終了)もまた手動でトリガされているので、SAMP ビットは新しいサンプルが変換される必要があるたびにクリアされる必要があります。

例 18-7: A/D 初期化コード例

CLR	ADPCFG	; A/D ポートを構成します。 ; すべての入力ピンはアナログです。
MOV	#0x2200,W0	
MOV	W0,ADCON1	; サンプルクロックソースと変換トリガーモードを ; を構成します。 ; 符号なし固定小数フォーマット、 ; 手動変換トリガー、 ; サンプリングの手動開始、 ; IDLE モードでの動作停止。
CLR	ADCON2	; A/D 電圧リファレンスと Buffer Fill モード ; を構成してください。 ; AVDD および AVSS からの VREF、 ; 入力はスキヤンされません。 ; サンプルごとの割込み
CLR	ADCON3	; A/D 変換クロックを構成してください。
CLR	ADCHS	; 入力チャネルを構成してください。 ; CH0+ input は AN0 です。 ; CH0- input is VREFL (AVss)
CLR	ADCSSL	; 入力はまったくスキヤンされません。
BCLR	IFS0,#ADIF	; A/D 変換割込みフラグをクリアします。
		; 必要に応じて、A/D 割込み優先ビット (ADIP<2:0>) をここに構成してください。 ; (初期設定優先レベルは 4 です)
BSET	IEC0,#ADIE	; A/D 変換割込みを有効にしてください。
BSET	ADCON1,#ADON	; A/D をオンにしてください。
BSET	ADCON1,#SAMP	; 入力のサンプリングを開始してください。
CALL	DELAY	; 変換開始の前に、正確なサンプリング時間が経過 ; したことを確認してください。
BCLR	ADCON1,#SAMP	; A/D サンプリングを終了し、変換を開始してください。 ; 変換シーケンスが完了されると、DONE ビットは ハードウェアによりセットされます。
:		; ADIF ビットがセットされます。

## 18.21 SLEEP モードおよび IDLE モード時の動作

CPU、バス、周辺機器のデジタルアクティビティが最小化されるので、SLEEP モードと IDLE モードは変換ノイズを最小化するためには有効です。

### 18.21.1 RC A/D クロックなしの CPU SLEEP モード

デバイスが SLEEP モードに入った場合、モジュールに対するすべてのクロックソースは停止され、論理 ‘0’ にとどまります。

変換の途中で SLEEP が発生すると、A/D は内蔵 RC クロックジェネレータからクロック供給されることなく変換が中断されます。SLEEP からウェイクアップするときも、中断した変換動作を再開することはありません。

レジスタ内容は、SLEEP モードに入り出すデバイスの影響を受けません。

### 18.21.2 RC A/D クロック付きの CPU SLEEP モード

A/D クロックソースが内蔵 A/D RC オシレータ (ADRC = 1) に設定されると、A/D モジュールが SLEEP モード中でも動作可能です。このことにより、変換からデジタルスイッチノイズが除去されます。変換が完了されると CONV ビットはクリアされ、その結果は A/D 結果バッファ、ADCBUF に設定されます。

A/D 割込みが有効 (ADIE= 1) ならば、A/D 割込み発生時にデバイスは SLEEP からウェイクアップします。A/D 割込みが現状の CPU 優先順位より高いと、プログラム実行は、A/D 割込みサービスルーチンから実行されます。そうでないときは、デバイスが SLEEP モードにされた PWRSAV 命令の次の命令から実行が続けられます。

A/D 割込みが有効でなければ、ADON ビットがセットされたままになっていても、A/D モジュールがオフになります。

A/D モジュール動作におけるデジタルノイズの影響を最小にするために、ユーザーは A/D 変換が SLEEP モードでも動作する変換トリガーソースを選ばなくてはなりません。自動変換トリガーオプションは、 $SLEEP(SSRC<2:0>= 111)$  中のサンプリングおよび変換に使用できます。自動変換オプションを使用するために、ADON ビットは PWRSAV 命令に先立つ命令でセットされる必要があります。

**注：** A/D が SLEEP でも動作継続するためには、A/D クロックソースは RC (ADRC = 1) に設定される必要があります。

### 18.21.3 CPU が IDLE モード中の A/D 操作

A/D モジュールは、ADSIDL ビット (ADCON1<13>) により、モジュールが IDLE 中は動作を停止するか IDLE も動作を継続するかが選択されます。ADSIDL = 0 ならば、デバイスが IDLE モードに入る場合、モジュールにより通常動作が継続されます。A/D 割込みが有効 (ADIE = 1) ならば、A/D 割込み発生時にデバイスは IDLE モードからウェイクアップします。A/D 割込みが現状の CPU 優先順位より高いときは、プログラム実行は、A/D 割込みサービスルーチンから実行されます。そうでないときは、デバイスが IDLE モードに置かれた PWRSAV 命令の次の命令から実行が続けられます。

ADSIDL = 1 ならば、モジュールは IDLE 中は動作を停止します。変換中にデバイスが IDLE モードに入れば変換は中断されます。IDLE モードからウェイクアップした場合にも中断した変換動作を再開することはありません。

## 18.22 RESET の影響

デバイス RESET によりすべてのレジスタが強制的に RESET 状態にされます。このことにより A/D モジュールは強制的にオフになり、進行中のすべての変換は中断されます。アナログ入力として重複されたで増幅されたすべてのピンはアナログ入力として構成されます。対応する TRIS ビットがセットされます。

ADCBUF レジスタの値はパワーオンリセットの間、初期化されません。ADCBUFO から ADCBUFF には未定データが含まれます。

## 18.23 12 ビット A/D コンバータに付随する特別機能レジスタ

以下の表にはアドレス、フォーマットを含む dsPIC30F 12 ビット A/D 特別機能レジスタが一覧になっています。未実装レジスタまたはレジスタ内のビット（あるいはその両方）はすべてゼロとして読み込まれます。

表 18-5: ADC レジスタマップ

ファイル名	ADR	ビット 15	ビット 14	ビット 13	ビット 12	ビット 11	ビット 10	ビット 9	ビット 8	ビット 7	ビット 6	ビット 5	ビット 4	ビット 3	ビット 2	ビット 1	ビット 0	RESET 状態									
INTCON1	0080	NSTDIS	—	—	—	—	OVATE	OVBT	COVTE	—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000 0000 0000 0000									
INTCON2	0082	ALТИVT	—	—	—	—	—	—	—	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INTOEP	0000 0000 0000 0000									
IFS0	0084	CNIF	BCLIF	I2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0	0000 0000 0000 0000									
IEC0	008C	CNIE	BCLIE	I2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INTOIE	0000 0000 0000 0000									
IPC2	0098	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>			0100 0100 0100 0100									
ADCBUF0	0280	ADC データバッファ 0															uuuuu uuuuu uuuuu uuuuu										
ADCBUF1	0282	ADC データバッファ 1															uuuuu uuuuu uuuuu uuuuu										
ADCBUF2	0284	ADC データバッファ 2															uuuuu uuuuu uuuuu uuuuu										
ADCBUF3	0286	ADC データバッファ 3															uuuuu uuuuu uuuuu uuuuu										
ADCBUF4	0288	ADC データバッファ 4															uuuuu uuuuu uuuuu uuuuu										
ADCBUF5	028A	ADC データバッファ 5															uuuuu uuuuu uuuuu uuuuu										
ADCBUF6	028C	ADC データバッファ 6															uuuuu uuuuu uuuuu uuuuu										
ADCBUF7	028E	ADC データバッファ 7															uuuuu uuuuu uuuuu uuuuu										
ADCBUF8	0290	ADC データバッファ 8															uuuuu uuuuu uuuuu uuuuu										
ADCBUF9	0292	ADC データバッファ 9															uuuuu uuuuu uuuuu uuuuu										
ADCBUFA	0294	ADC データバッファ 10															uuuuu uuuuu uuuuu uuuuu										
ADCBUFB	0296	ADC データバッファ 11															uuuuu uuuuu uuuuu uuuuu										
ADCBUFC	0298	ADC データバッファ 12															uuuuu uuuuu uuuuu uuuuu										
ADCBUFD	029A	ADC データバッファ 13															uuuuu uuuuu uuuuu uuuuu										
ADCBUFE	029C	ADC データバッファ 14															uuuuu uuuuu uuuuu uuuuu										
ADCBUFF	029E	ADC データバッファ 15															uuuuu uuuuu uuuuu uuuuu										
ADCON1	02A0	ADON	—	ADSIDL	—	—	—	FORM[1:0]		SSRC[2:0]		—	—	ASAM	SAMP	CONV	0000 0000 0000 0000										
ADCON2	02A2	VCFG[2:0]			—	—	CSCNA	—	—	BUFS	—	SMPI[3:0]			BUFM	ALTS	0000 0000 0000 0000										
ADCON3	02A4	—	—	—	SAMC[4:0]				ADRC	—	ADCS[5:0]				0000 0000 0000 0000												
ADCHS	02A6	—	—	—	CHONB	CHOSB[3:0]			—	—	—	CHONA	CHOSA[3:0]			0000 0000 0000 0000											
ADPCFG	02A8	PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8	PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000 0000 0000 0000									
ADCSSL	02AA	ADC 入力スキャン選択レジスタ															0000 0000 0000 0000										

凡例: u = 未定

注: すべての割込みソースおよびその関連制御ビットは特定のデバイスでは利用できない可能性があります。詳細についてはデバイスデータシートをご参照ください。

## 18.24 設計の秘訣

質問 1: A/D コンバータのシステムパフォーマンスはどうすれば最適化できますか?

答え:

1. タイミング仕様をすべて満たしていることをご確認ください。モジュールをオフとオンにするとときは、サンプルを取得するまで待つ必要がある最小の遅れがあります。入力チャネルを変更中ならば、同じくこれのために待つ必要がある最小の遅れがあり、そして最終的に各ビット変換のために選択された時間である TAD が存在します。この TAD は ADCON3 で選択されますが、電気的特性に指定された特定の範囲内でなされる必要があります。TAD が短過ぎる場合、その結果は変換終了する前に十分に変換されない可能性があり、TAD が長過ぎる場合、サンプリングコンデンサの電圧が変換が完了される前に無くなるかもしれません。このタイミング特性はデバイスデータシートの「電気的仕様」セクションで提供されます。
2. しばしば、アナログシグナルのソース電気抵抗は高く ( $10 \text{ k}\Omega$  を超える)、そこで、ソースの漏れ電流や、ソース電流によるサンプルコンデンサの充電が、変換の正確さに影響する可能性があります。入力シグナルが変わるのが遅い場合は、 $0.1\mu\text{F}$  コンデンサをアナログ入力に接続してみてください。このコンデンサにより、サンプルされているアナログ電圧が充電され、 $18\text{pF}$  内蔵ホールドコンデンサを変更するために必要な瞬間電流が供給されます。
3. A/D 変換を開始する前にデバイスを SLEEP モードにしてください。RC クロックソース選択は SLEEP モードでの変換に必要です。CPU およびその他の周辺機器からのデジタルノイズが最小化されるので、このテクニックにより正確さが増します。

質問 2: A/D に関する良い参考文献をご存知ですか?

答え: A/D 変換を理解するための良い参考文献は Prentice Hall (ISBN 0-13-03-2848-0) 発行の『Analog-Digital Conversion Handbook』第 3 版です。

質問 3: チャネル・サンプルとサンプル・割込みのコンピネーションがバッファのサイズを超えています。バッファに何が起こるのでしょうか?

答え: この構成は推奨されません。このバッファには未定結果が含まれるでしょう。

## 18.25 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップしています。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正や可能な制限をして使用できる可能性もあります。12ビット A/D コンバータモジュールに関連する現在のアプリケーションノートは以下です。

タイトル	アプリケーションノート #
アナログ対デジタル (A/D) コンバータの使い方	AN546
ディスプレイ、キーボード付きの 4 チャネルデジタル電圧メーター	AN557
A/D コンバータ性能特性の理解	AN693

**注：** デバイスの dsPIC30F ファミリーに関しての、追加のアプリケーションノートやコード例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 18.26 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

dsPIC30F 12 ビット A/D コンバータモジュールの編集および技術情報の改訂を反映するために改訂されました。

注：



## 第 19 章 . UART

### ハイライト

この章は、以下の項目を含んでいます。

19.1 序章 .....	19-2
19.2 制御レジスタ .....	19-3
19.3 UART ポーレートレジスタ (BRG) .....	19-8
19.4 UART 構成 .....	19-10
19.5 UART トランスマッタ .....	19-11
19.6 UART レシーバ .....	19-14
19.7 9 ビット通信における UART の使用 .....	19-18
19.8 ブレーク文字の受信 .....	19-19
19.9 初期設定 .....	19-20
19.10 UART のその他の機能 .....	19-21
19.11 CPU SLEEP および IDLE モードの際の UART 動作 .....	19-21
19.12 UART モジュール関連レジスタ .....	19-22
19.13 設計の秘訣 .....	19-23
19.14 関連するアプリケーションノート .....	19-24
19.15 改訂履歴 .....	19-25

## 19.1 序章

ユニバーサル非同期レシーバトランスマッター(UART)モジュールは、dsPIC30Fデバイスファミリーで利用できるシリアル入出力モジュールの1つです。UARTは全二重通信方式の非同期システムで、パーソナルコンピュータなどの周辺デバイスRS-232、RS-485インターフェースでの通信が可能です。

UARTモジュールの主な特徴は次の通りです。

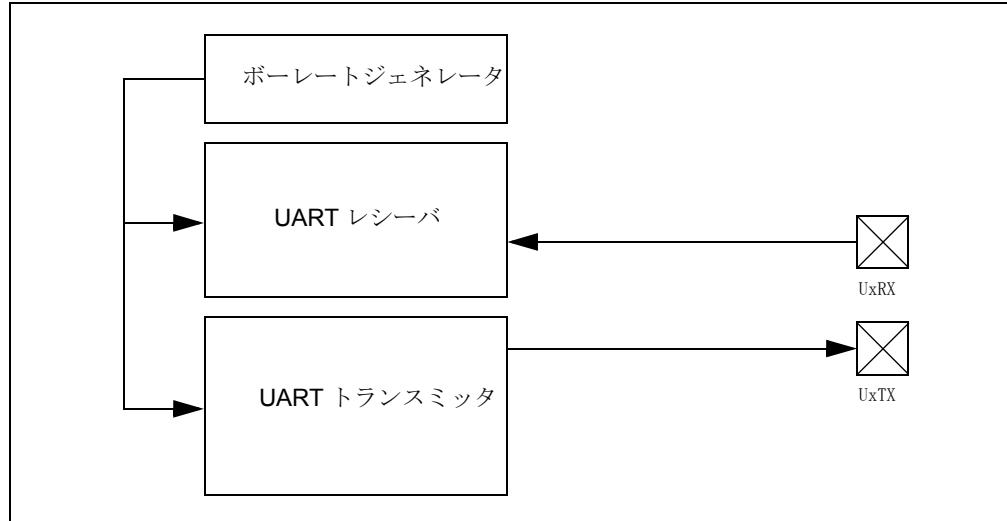
- UxTXおよびUxRXピン経由の全二重通信方式8または9ビットデータ送信
- 偶数、奇数、または非パリティオプション(8ビットデータに関して)
- 1または2個のSTOPビット
- 16ビットプレスケーラ付き完全統合ボーレートジェネレータ
- $F_{CY} = 30\text{ MHz}$ において29bpsから1.875Mbps間の範囲のボーレート
- 深さ4のファーストイントゥアウト(FIFO)送信データバッファ
- 深さ4のFIFO受信データバッファ
- パリティ、フレーミング、バッファオーバーランエラー検出
- アドレス検出による9ビットモードサポート(9個目のビット=1)
- 送受信割込み
- 診断サポートのためのループバックモード

**注:** 各dsPIC30Fデバイスバリエーションの中には1つ以上のUARTモジュールを内蔵しているものがあります。ピン、制御/状態ビット、レジスタの名称に用いられる‘x’は特にこのモジュール番号を示します。詳細は各デバイスデータシートを参照してください。

図19-1はUARTの単純ブロック図を示しています。UARTモジュールは重要なハードウェア要素で構成されます。

- ボーレートジェネレータ
- 非同期トランスマッタ
- 非同期レシーバ

図19-1: UART単純ブロック線図



## 19.2 制御レジスタ

レジスタ 19-1: Ux モード : UARTx モードレジスタ

上位バイト :							
R/W-0	U-0	R/W-0	U-0	U-0	R/W-0	U-0	U-0
UARTEN	—	USIDL	—	reserved	ALTIO	reserved	reserved
ビット 15							ビット 8

下位バイト :

R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
WAKE	LPBACK	ABAUD	—	—	PDSEL<1:0>	STSEL	ビット 0
ビット 7							ビット 0

ビット 15 **UARTEN: UART 有効化ビット**

1 = UART 有効化。UART ピンは UEN<1:0>により定義されたUART と UTXEN 制御ビットにより制御される。  
0 = UART 無効化。UART ピンは対応する PORT、LAT、TRIS ビットにより制御される。

ビット 14 未実装: ‘0’ として読み込み

ビット 13 **USIDL: IDLE モードで停止**

1 = デバイスが IDLE モードに入った場合、動作を停止。  
0 = IDLE モードで動作を続行。

ビット 12 未実装: ‘0’ として読み込み

ビット 11 予約済み: この位置に ‘0’ と書き込む

ビット 10 **ALTIO: UART 代替入出力選択ビット**

1 = UART は UXATX と UXARX 入出力ピンを使用して通信。  
0 = UART は UxTX と UxRX 入出力ピンを使用して通信。

注: 代替 UART 入出力ピンはすべてのデバイスで利用できるとは限りません。詳細はデバイスデータシートを参照してください。

ビット 9-8 予約済み: この位置に ‘0’ と書き込む

ビット 7 **WAKE: SLEEP モード中、スタートビット検出時のウェイクアップ有効化ビット**

1 = ウェイクアップ有効化  
0 = ウェイクアップ無効化

ビット 6 **LPBACK: UART ループバックモード選択ビット**

1 = ループバックモード有効化  
0 = ループバックモード無効化

ビット 5 **ABAUD: オートボーリング有効化ビット**

1 = UxRX ピンからキャプチャモジュールへ入力。  
0 = ICx ピンからキャプチャモジュールへ入力。

ビット 4-3 未実装: ‘0’ として読み込み

ビット 2-1 **PDSEL<1:0>: パリティおよびデータ選択ビット**

11 = 9 ビットデータ、パリティなし  
10 = 8 ビットデータ、奇数パリティ  
01 = 8 ビットデータ、偶数パリティ  
00 = 8 ビットデータ、パリティなし

ビット 0 **STSEL: STOP 選択ビット**

1 = 2 STOP ビット  
0 = 1 STOP ビット

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’ が読み込まれます  
ト

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です  
されます

## レジスタ 19-2: UXSTA: UARTx 状態および制御レジスタ

上位バイト :							
R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R-0	R-1
UTXISEL	—	—	—	UTXBRK	UTXEN	UTXBF	TRMT
ビット 15							ビット 8

下位バイト :							
R/W-0	R/W-0	R/W-0	R-1	R-0	R-0	R/C-0	R-0
URXISEL<1:0>	ADDEN	RIDLE	PERR	FERR	OERR	URXDA	
ビット 7							ビット 0

- ビット 15 **UTXISEL:** 送信割り込みモード選択ビット  
1 = 文字が送信シフトレジスタに転送され、その結果、送信バッファが空になった場合に割り込む。  
0 = 文字が送信シフトレジスタに転送された場合に割り込む(少なくとも1個の文字が送信バッファ内にあることを意味する)。
- ビット 14-12 未実装: ‘0’として読み込み
- ビット 11 **UTXBRK:** 送信ブレークビット  
1 = UxTX ピンはトランスマッタの状態にかかわらずローで動作。  
0 = UxTX ピンは通常通りに動作する。
- ビット 10 **UTXEN:** 送信有効化ビット  
1 = UART トランスマッタは有効化。UxTX ピンは UART によって制御される (UARTEN = 1 の場合)。  
0 = UART トランスマッタは無効化。実行待ちの送信はすべて中止され、バッファはリセットされる。UxTX ピンは PORT によって制御される。
- ビット 9 **UTXBF:** 送信バッファフル状態ビット (読み取り専用)  
1 = 送信バッファがフル。  
0 = 送信バッファはフルではない。データが少なくともあと1個書き込み可能。
- ビット 8 **TRMT:** 送信シフトレジスタが空ビット (読み取り専用)  
1 = 送信シフトレジスタは空で、送信バッファも空 (最後の送信が完了している)。  
0 = 送信シフトレジスタは空ではなく、送信が実行中であるか送信バッファで送信待ち。
- ビット 7-6 **URXISEL<1:0>:** 受信割り込みモード選択ビット  
11 = 受信バッファがフル (すなわち4データ文字が入力されている)の場合、割り込みフラグビットがセットされる  
10 = 受信バッファが3/4フルである(3データ文字が入力されている)場合、割り込みフラグビットがセットされる  
0x = 文字を受信した場合、割り込みフラグビットがセットされる
- ビット 5 **ADDEN:** アドレス文字検出 (受信データのビット 8 = 1)  
1 = アドレス検出モード有効化。9ビットモードが選択されない場合、この制御は効果を発しない。  
0 = アドレス検出モード無効化。
- ビット 4 **RIDLE:** レシーバ IDLE ビット (読み取り専用)  
1 = 受信は IDLE。  
0 = データ受信中。
- ビット 3 **PERR:** パリティエラー状態ビット (読み取り専用)  
1 = 現在の文字についてパリティエラーを検出。  
0 = パリティエラーは検出されなかった。
- ビット 2 **FERR:** フレーミングエラー状態ビット (読み取り専用)  
1 = 現在の文字についてフレーミングエラーを検出。  
0 = フレーミングエラーは検出されなかった。

## レジスタ 19-2: UxSTA: UARTx 状態および制御レジスタ (続き)

ビット 1 **OERR:** 受信バッファオーバーランエラー状態ビット (読み取り / クリア専用)

1 = 受信バッファはオーバーランした。

0 = 受信バッファはオーバーランしていない。

ビット 0 **URXDA:** 受信バッファデータ利用可能ビット (読み取り専用)

1 = 受信バッファにはデータがあり、少なくともあと 1 個の文字は読み取り可能。

0 = 受信バッファは空。

凡例 :

<b>R</b> = 読み込み可能 ビット	<b>W</b> = 書き込み可能 ビット	<b>U</b> = 未実装、'0' が読み込まれます	<b>C</b> = クリア可能なビット
<b>-n</b> = POR での値	'1' = ビットが セットされます	'0' = ビットはクリアされます	<b>x</b> = ビットは不定です

## レジスタ 19-3: UxRXREG: UARTx 受信レジスタ

上位バイト :							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	R-0
—	—	—	—	—	—	—	URX8
ビット 15							ビット 8

下位バイト :							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
URX<7:0>							
ビット 7							ビット 0

ビット 15-9 未実装: ‘0’ として読み込み

ビット 8 URX8: 受信文字中のデータビット 8 (9 ビットモードの場合)

ビット 7-0 URX<7:0>: 受信文字中のデータビット 7-0

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’ が読み込まれます

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です

されます されます

## レジスタ 19-4: UxTXREG: UARTx 送信レジスタ (書き込み専用)

上位バイト :							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	W-x
—	—	—	—	—	—	—	UTX8
ビット 15							ビット 8

下位バイト :							
W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
UTX<7:0>							
ビット 7							ビット 0

ビット 15-9 未実装: ‘0’ として読み込み

ビット 8 UTX8: 送信される文字中のデータビット 8 (9 ビットモードの場合)

ビット 7-0 UTX<7:0>: 送信される文字中のデータビット 7-0

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装、‘0’ が読み込まれます

-n = POR での値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定です

されます されます

## レジスタ 19-5: UxBRG: UARTx ポーレートレジスタ

上位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BRG<15:8>							
ビット 15							

下位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BRG<7:0>							
ビット 7							

ビット 15-0 **BRG<15:0>**: ポーレート除数ビット

凡例 :

R = 読み込み可能ビット	W = 書き込み可能ビット	U = 未実装、'0' が読み込まれます
-n = POR での値	'1' = ビットがセット	'0' = ビットはクリア
	されます	x = ビットは不定で
		されます

## 19.3 UART ボーレートレジスタ (BRG)

UART モジュールは専用の 16 ビットボーレートジェネレータを含んでいます。UxBRG レジスタはフリーランの 16 ビットタイマーの周期を制御します。式 19-1 はボーレートを求める計算式を示しています。

式 19-1: **UART ボーレート**

$$\text{Baud Rate} = \frac{FCY}{16 \cdot (UxBRG + 1)}$$

$$UxBRG = \frac{FCY}{16 \cdot \text{Baud Rate}} - 1$$

注: FCY は命令サイクルクロック周波数を表す。

例 19-1 は次の状態におけるボーレートエラーの計算である。

- FCY = 4 MHz
- 求めるボーレート = 9600

例 19-1: **ボーレートエラー計算**

$$\text{求めるボーレート} = FCY/(16 (UxBRG + 1))$$

以下の式で UxBRG 値を計算します。

$$\begin{aligned} UxBRG &= ((FCY/Desired\ Baud\ Rate)/16) - 1 \\ UxBRG &= ((4000000/9600)/16) - 1 \\ UxBRG &= [25.042] = 25 \end{aligned}$$

$$\begin{aligned} \text{算出されたボーレート} &= 4000000/(16 (25 + 1)) \\ &= 9615 \end{aligned}$$

$$\begin{aligned} \text{エラー} &= \frac{(\text{Calculated Baud Rate} - \text{Desired Baud Rate})}{\text{Desired Baud Rate}} \\ &= (9615 - 9600)/9600 \\ &= 0.16\% \end{aligned}$$

ボーレートの可能最大値はFCY/ 16 (UxBRG = 0に対して), 可能最小値はFCY/ (16 \* 65536)です。

UxBRG レジスタに新しい値を書き込むと BRG タイマーはリセット (クリア) されます。このため、BRG はタイマーオーバーフローを待たずに新しいボーレートを発生することができます。

## 19.3.1 ポーレート表

表 19-1 は共通デバイス命令サイクル周波数 ( $F_{CY}$ ) に対する UART ポーレートを示しています。各周波数に対するポーレートの最大値・最小値も示されています。

表 19-1: UART ポーレート

BAUD RATE (Kbps)	FCY = 30 MHz		25 MHz		20 MHz		16 MHz		
	KBAUD	% ERROR	BRG value (decimal)	KBAUD	% ERROR	BRG value (decimal)	KBAUD	% ERROR	BRG value (decimal)
0.3	0.3	0.0	6249	0.3	+0.01	5207	0.3	0.0	4166
1.2	1.1996	0.0	1562	1.2001	+0.01	1301	1.1996	0.0	1041
2.4	2.4008	0.0	780	2.4002	+0.01	650	2.3992	0.0	520
9.6	9.6154	+0.2	194	9.5859	-0.15	162	9.6154	+0.2	129
19.2	19.1327	-0.4	97	19.2901	0.47	80	19.2308	+0.2	64
38.4	38.2653	-0.4	48	38.1098	-0.76	40	37.8788	-1.4	32
56	56.8182	+1.5	32	55.8036	-0.35	27	56.8182	+1.5	21
115	117.1875	+1.9	15	111.6071	-2.95	13	113.6364	-1.2	10
250							250	0.0	4
500							500	0.0	1
MIN.	0.0286	0.0	65535	0.0238	0.0	65535	0.015	0.0	65535
MAX.	1875	0.0	0	1562.5	0.0	0	1250	0.0	0
BAUD RATE (Kbps)	FCY = 12 MHz		10 MHz		8 MHz		7.68 MHz		
	KBAUD	% ERROR	BRG value (decimal)	KBAUD	% ERROR	BRG value (decimal)	KBAUD	% ERROR	BRG value (decimal)
0.3	0.3	0.0	2499	0.3	0.0	2082	0.2999	-0.02	1666
1.2	1.2	0.0	624	1.1996	0.0	520	1.199	-0.08	416
2.4	2.3962	-0.2	312	2.4038	+0.2	259	2.4038	+0.16	207
9.6	9.6154	-0.2	77	9.6154	+0.2	64	9.6154	+0.16	51
19.2	19.2308	+0.2	38	18.9394	-1.4	32	19.2308	+0.16	25
38.4	37.5	+0.2	19	39.0625	+1.7	15	38.4615	+0.16	12
56	57.6923	-2.3	12	56.8182	+1.5	10	55.5556	-0.79	8
115			6				250	0.0	1
250	250	0.0	2				500	0.0	0
500									
MIN.	0.011	0.0	65535	0.010	0.0	65535	0.008	0.0	65535
MAX.	750	0.0	0	625	0.0	0	500	0.0	0
BAUD RATE (Kbps)	FCY = 5 MHz		4 MHz		3.072 MHz		1.8432 MHz		
	KBAUD	% ERROR	BRG value (decimal)	KBAUD	% ERROR	BRG value (decimal)	KBAUD	% ERROR	BRG value (decimal)
0.3	0.2999	0.0	1041	0.3001	0.0	832	0.3	0.0	639
1.2	1.2019	+0.2	259	1.2019	+0.2	207	1.2	0.0	159
2.4	2.4038	+0.2	129	2.4038	+0.2	103	2.4	0.0	79
9.6	9.4697	-1.4	32	9.6154	+0.2	25	9.6	0.0	19
19.2	19.5313	+1.7	15	19.2308	+0.2	12	19.2	0.0	9
38.4	39.0625	+1.7	7				38.4	0.0	4
56									
115									
250									
500									
MIN.	0.005	0.0	65535	0.004	0.0	65535	0.003	0.0	65535
MAX.	312.5	0.0	0	250	0.0	0	192	0.0	0

## 19.4 UART 構成

UART は標準の非ゼロ復帰 (NRZ) 形式 (START ビット 1 個、8 あるいは 9 個のデータビット、1 あるいは 2 個の STOP ビット) を使用しています。パリティはハードウェアにサポートされており、ユーザーは偶数、奇数、パリティなしのいずれかとして設定することができます。最も一般的なデータ形式は 8 ビット、パリティなし、1 個の STOP ビット (8, N, 1 と表されます) で、これがデフォルト (POR) に設定されています。データビットと STOP ビットの数、またパリティは、PDSEL<1:0> ( $\text{UxMODE}$ <2:1>) および STSEL ( $\text{UxMODE}$ <0>) ビットで設定されています。オンチップの専用 16 ビットボーレートジェネレータを用いて、発振器から標準ボーレート周波数を得ることができます。UART は LSb を最初に送受信します。UART のトランシミッタとレシーバは独立して機能しますが、同じデータ形式とボーレートを使用します。

### 19.4.1 UART 有効化

UART モジュールは UARTEN ( $\text{UxMODE}$ <15>) ビットおよび UTXEN ( $\text{UxSTA}$ <10>) ビットをセットすることにより有効化します。一度有効化すると、 $\text{UxTX}$  および  $\text{UxRX}$  ピンはそれぞれ出力および入力として設定され、対応する入出力ポートピンに対する TRIS および PORT ピンの設定を無効にします。送信が行われていない場合、 $\text{UxTX}$  ピンはロジック ‘1’ の状態になります。

### 19.4.2 UART 無効化

UART モジュールは UARTEN ( $\text{UxMODE}$ <15>) ビットをクリアすることにより無効化します。これはすべての RESET 操作後のデフォルト状態です。UART が無効化すると、すべての UART ピンは、対応する PORT および TRIS ビットの制御下でポートピンとして作動します。

UART モジュールの無効化は、バッファを空の状態にリセットします。バッファ内のデータ文字はすべて失われ、ボーレートカウンタはリセットされます。

UART モジュールを無効化した場合、モジュールに関するエラーおよび状態フラグはすべてリセットされます。URXDA, OERR, FERR, PERR, UTXEN, UTXBRK および UTXBF ビットはクリアされますが、Ridle および TRMT はセットされます。ADDEN, URXISEL<1:0>, UTXISEL, または  $\text{UxMODE}$  および  $\text{UxBRG}$  レジスタを含む他の制御ビットには影響はありません。

UART がアクティブである状態で UARTEN ビットをクリアすると、待機中の送受信はすべて中止され、モジュールは上記のようにリセットされます。UART を再有効化すると、UART は同じ構成で再起動します。

### 19.4.3 代替 UART 入出力ピン

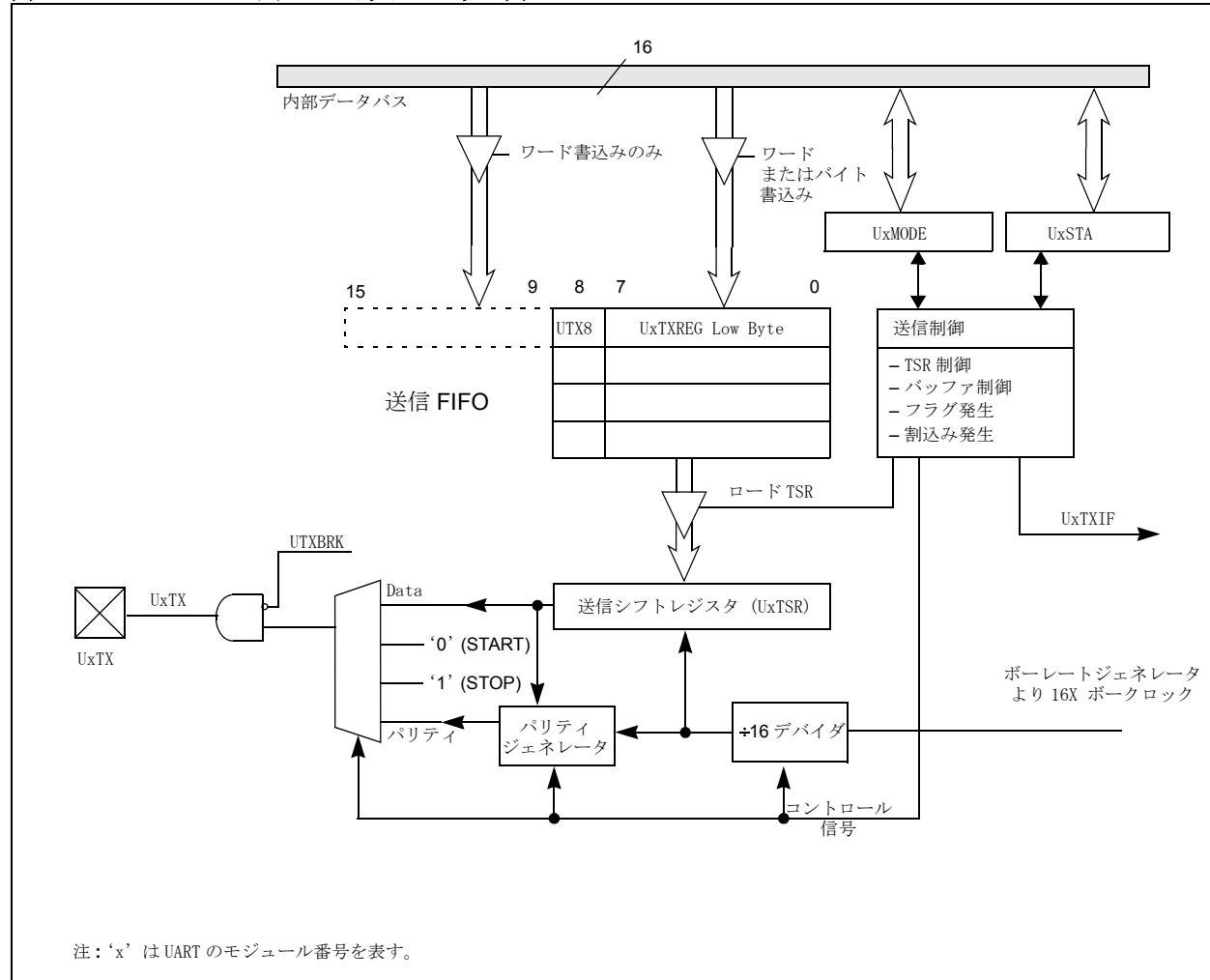
dsPIC30F デバイスの中には、通信用の代替 UART 送受信ピンを有するものがあります。代替 UART ピンは、一次 UART ピンを他の周辺装置と共有している場合に有用です。代替入出力ピンは ALTIO bit ( $\text{UxMODE}$ <10>) をセットすることにより有効化します。ALTIO = 1 の場合、 $\text{UxTX}$  および  $\text{UxRX}$  ピンの代わりに、 $\text{UxATX}$  および  $\text{UxARX}$  ピン（それぞれ代替送信および代替受信ピンに相当）が UART モジュールによって使用されます。ALTIO = 0 の場合、 $\text{UxTX}$  および  $\text{UxRX}$  ピンが UART モジュールによって使用されます。

## 19.5 UART トランシッタ

図 19-2 は UART トランシッタブロック図を示しています。トランシッタの中心は送信シフトレジスタ (UxTSR) です。シフトレジスタはデータを送信 FIFO バッファ、UxTXREG から取得します。UxTXREG レジスタはソフトウェア内でデータがロードされます。UxTSR レジスタは、その前の送信中データの STOP ビットが送信されるまでロードされません。STOP ビットが送信されると、ただちに UxTSR には UxTXREG レジスタからの新しいデータがロードされます（可能な場合）。

**注：** UxTSR レジスタはデータメモリにマップされていないため、ユーザーには利用できません。

図 19-2: UART トランシッタブロック図



送信は UTXEN (UxSTA<10>) をセットすることにより有効化します。実際の送信は UxTXREG レジスタにデータがロードされ、ボーレートジェネレータ (UxBRG) がシフトクロックを引き起こすまで行われません(図 19-2)。送信は、まず UxTXREG レジスタをロードし、次に UTXEN 有効化ビットをセットするという方法によっても開始できます。通常、送信が初めて開始された場合、UxTSR レジスタは空なので、UxTXREG レジスタへの転送に続いてただちに UxTSR への転送が行われます。UTXEN ビットを送信中にクリアすると送信は中止されて、トランスマッタはリセットされます。その結果、UxTX ピンは高インピーダンス状態に戻ります。

9 ビット送信を選択するためには、PDSEL<1:0> ビット (UxM0DE<2:1>) を '11' にセットし、9 個目のビットを UTX9 bit (UxTXREG<8>) に書き込みます。ワード書き込みは 9 個のビットすべてが同時に書き込まれるように、UxTXREG に対して実行する必要があります。

**注：** 9 ビットデータ送信の場合、パリティはありません。

## 19.5.1 送信バッファ (UxTXB)

すべての UART は深さ 4、幅 9 ビットの FIFO 送信データバッファを有しています。UxTXREG レジスタにより、ユーザーは次に利用可能なバッファ位置にアクセスできます。ユーザーはバッファ内に 4 ワードまで書き込むことができます。UxTXREG の内容が UxTSR レジスタに転送されると、現在のバッファ位置に新しいデータを書き込むことが可能となり、次のバッファ位置が UxTSR レジスタに転送用のデータとされます。UTXBF (UxSTA<9>) 状態ビットは、バッファがフルの場合いつでもセットされます。フル状態のバッファに書き込みを行おうとしても、新しいデータは FIFO に格納されません。

FIFO はすべてのデバイス RESET の際にリセットされます。ただし、デバイスがパワーセービングモードに入る、あるいはパワーセービングモードからウェイクアップする場合には影響を受けません。

## 19.5.2 送信割り込み

送信割り込みフラグ (UxTXIF) は対応する割り込みフラグ状態 (IFS) レジスタに配置されています。UART の送信割り込みがいつ発生するかは、UTXISEL 制御ビット (UxSTA<15>) によって決定されます。

1. UTXISEL = 0 の場合、送信バッファから送信シフトレジスタ (UxTSR) に 1 個のワードが転送されると割り込みが発生します。このことは、送信バッファに少なくとも 1 個、空のワードがあることを示しています。割り込みは 1 個 1 個のワードの転送後に発生するので、このモードは割り込みを頻繁に処理できる場合（すなわち、次のワードの送信前に ISR が完了する場合）に有益です。
2. UTXISEL = 1 の場合、送信バッファから送信シフトレジスタ (UxTSR) に 1 個のワードが転送され、送信バッファが空である時に割り込みが発生します。割り込みは 4 個のワードが送信された後にのみ発生するので、この「ブロック割り込み」モードはユーザーのコードが割り込みを迅速に処理（すなわち、次のワードの送信前に ISR が完了する場合）できない場合に有益です。

UxTXIF ビットはモジュールが最初に有効化した時にセットされます。

ISR 内で UxTXIF ビットをクリアしてください。

動作中に 2 種類のに割り込みモードを変更することができます。

**注：** When the UTXEN ビットがセットされる際、UTXISEL = 0 ならば、UxTXIF フラグビットもセットされます。これは、送信バッファがまだフル状態ではない (UxTXREG レジスタに送信データを移動させることができる) ためです。

UxTXIF フラグビットは UxTXREG レジスタの状態を示し、TRMT ビット (UxSTA<8>) は UxTSR レジスタの状態を示します。TRMT 状態ビットは読み取り専用ビットで、UxTSR レジスタが空の場合セットされます。このビットには割り込みロジックは附属していないので、UxTSR レジスタが空かどうか判断するためにはこのビットをポーリングする必要があります。

### 19.5.3 UART 送信のセットアップ

送信セットアップの際には以下の操作を行ってください。

1. UxBRG レジスタを適切なボーレートに初期化してください（セクション 19.3 「UART ボーレートレジスタ（BRG）」参照）。
2. PDSEL<1:0> (UxMODE<2:1>) および STSEL (UxMODE<0>) ビットに書き込みをして、データビットの数、STOP ビットの数、パリティ選択をセットしてください。
3. UEN<1:0> (UxMODE<9:8>) および RTSMD (UxMODE<11>) ビットに書き込みをして、要求された通りに UxCTS および UxRTS ピンを設定してください。
4. 送信割り込みを希望する場合は、対応する割り込み有効化制御レジスタ (IEC) 内の UxTXIE 制御ビットをセットしてください。対応する割り込み優先順位制御レジスタ (IPC) 内の UxTXIP<2:0> 制御ビットを使用して割り込み優先順位を指定してください。  
また、UTXISEL (UxSTA<15>) ビットに書き込みをして送信割り込みモードを選択してください。
5. UARLEN (UxMODE<15>) ビットをセットして UART モジュールを有効化してください。
6. UTXEN (UxSTA<10>) ビットをセットして送信を有効化してください。これによって、UxTXIF ビットもセットされます。UART 送信割り込みをサポートするソフトウェアルーチン中で、UxTXIF ビットをクリアする必要があります。UxTXIF ビット操作は、UTXISEL 制御ビットによって制御されます。
7. UxTXREG レジスタにデータをロードしてください（送信開始）。9 ビット送信を選択した場合、1 個のワードをロードしてください。8 ビット送信を使用する場合は、1 個のバイトをロードしてください。UxTXBF 状態ビット (UxSTA<9>) がセットされるまで、データをバッファにロードすることができます。

図 19-3: 送信 (8 ビットあるいは 9 ビットデータ)

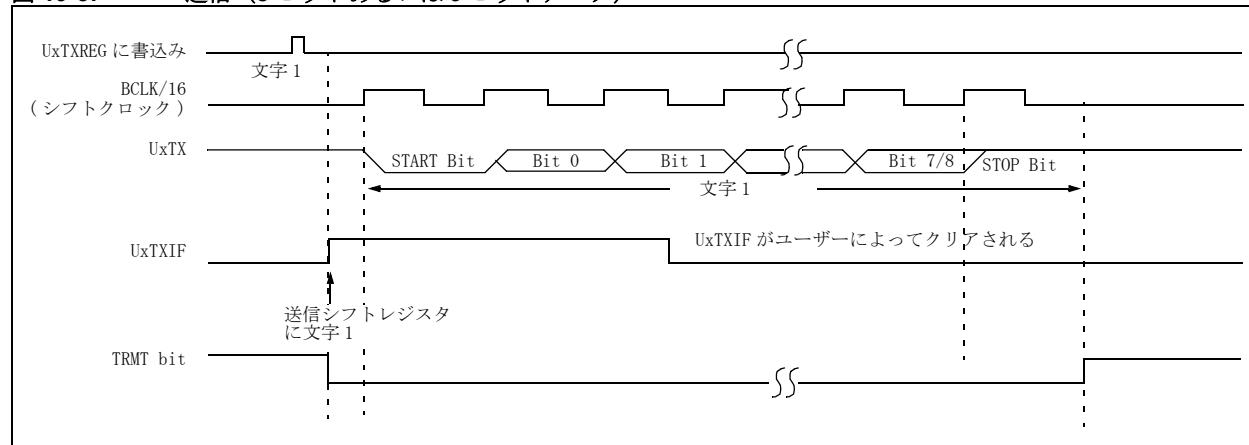
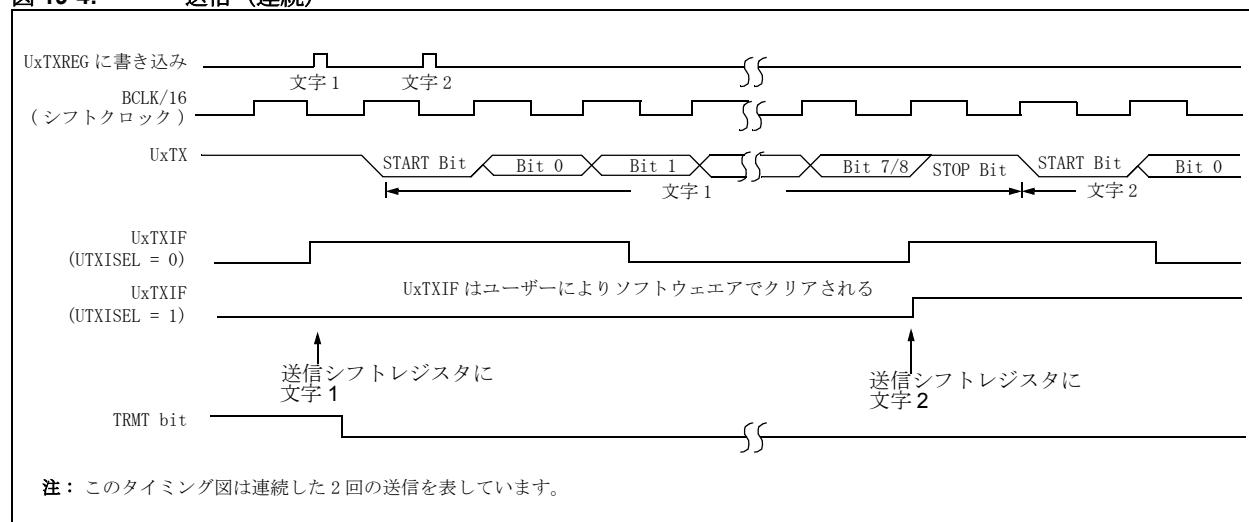


図 19-4: 送信 (連続)



## 19.5.4 区切り文字の送信

UTXBRK ビット ( $UxSTA<11>$ ) をセットすると、 $UxTX$  行は強制的に ‘0’ となります。UTXBRK は他のいかなるトランスマッタアクティビティをもオーバーライドします。トランスマッタが IDLE ( $TRMT = 1$ ) になるまで、UTXBRK をセットすることはできません。

区切り文字を送信するには、UTXBRK ビットがソフトウエアによってセットされ、その状態が最低 13 ポークロック間持続する必要があります。ポークロック周期はソフトウエア内で計時されます。続いて UTXBRK ビットがソフトウエアによってクリアされ、STOP ビットが発生します。STOP ビットが確実に発生するためには、UTXBUF のロードを再開したりトランスマッタアクティビティを再開する前に、少なくとも 1 あるいは 2 ポークロック待つ必要があります。

**注：** 区切り文字を送信しても、トランスマッタ割り込みは発生しません。

## 19.6 UART レシーバ

図 19-5 はレシーバブロック図を示しています。レシーバの中心は受信（シリアル）シフトレジスタ ( $UxRSR$ ) です。データは  $UxRX$  ピンに受信され、データ復元ブロックに送られます。データ復元ブロックはボーレート × 16 で動作しますが、メインの受信シリアルシフターはボーレートで作動します。 $UxRX$  ピンに STOP ビットをサンプリングしたあと、 $UxRSR$  内の受信データは受信 FIFO（空の場合）に転送されます。

**注：**  $UxRSR$  レジスタはデータメモリにマップされていないため、ユーザーには利用できません。

$UxRX$  ピンに高レベルあるいは低レベルが存在するかを決定するために、 $UxRX$  ピン上のデータは多くの検出回路では3回サンプリングされます。図 19-5 はサンプリング方式を示しています。

### 19.6.1 受信バッファ ( $UxRXB$ )

UART レシーバは深さ 4、幅 9 ビットの FIFO 受信データバッファです。 $UxRXREG$  は FIFO へのアクセスを提供するメモリマップされたレジスタです。バッファオーバーランが発生する前に、4 ワードのデータ受信と FIFO への転送、5 個目のワードの  $UxRSR$  レジスタへのシフト開始を行うことができます。

### 19.6.2 レシーバエラーハンドリング

FIFO がフル状態（4 文字）で、5 個目の文字が  $UxRSR$  レジスタに完全に受信されると、オーバーランエラービット、OERR ( $UxSTA<1>$ ) がセットされます。 $UxRSR$  内のワードは保存されますが、OERR ビットがセットされているかぎり、受信 FIFO へのそれ以上の転送は禁じられます。さらにデータを受信させるためには、ソフトウエアで OERR ビットをクリアする必要があります。

オーバーランの前に受信されたデータを保存したい場合は、まず 5 個の文字をすべて読み込んでから OERR ビットをクリアしてください。5 個の文字を破棄してもかまわない場合は、OERR ビットをそのままクリアすることができます。この操作により受信 FIFO は事実上リセットされ、前に受信されたデータはすべて失われます。

**注：** OERR ビットをクリアする前に受信 FIFO 内のデータを読み出す必要があります。 FIFO は OERR がクリアされるとリセットされ、バッファ内のすべてのデータは失われます。

フレーミングエラービット、FERR ( $UxSTA<2>$ ) は、ロジック低レベルの STOP ビットが検出されるとセットされます。

パリティエラービット、PERR ( $UxSTA<3>$ ) は、FIFO バッファの一番上にあるデータワード（すなわち、現在のワード）にパリティエラーが検出されるとセットされます。たとえば、パリティエラーは、パリティが偶数にセットされているが、データ内の 1 の総数が奇数であると検出された場合などに発生します。PERR ビットは 9 ビットモードでは使われません。FERR と PERR ビットは対応するワードとともにバッファされ、データワードの読み込み前に読み出す必要があります。

### 19.6.3 受信割込み

UART 受信割り込みフラグ (UxRXIF) は、対応する割り込みフラグ状態 (IFS) レジスタにあります。URXISEL<1:0> (UxSTA<7:6>) 制御ビットが、UART レシーバが割り込みを発生させる時期を決定します。

- a) URXISEL<1:0> = 00 あるいは 01 である場合、データワードが受信シフトレジスタ (UxRSR) から受信バッファへ転送されるたびに割り込みが発生します。受信バッファには 1 個以上の文字がある可能性があります。
- b) URXISEL<1:0> = 10 の場合、ワードが受信シフトレジスタ (UxRSR) から受信バッファに転送されて、その結果、受信バッファが 3 個または 4 個の文字を有するときに割り込みが発生します。
- c) URXISEL<1:0> = 11 の場合、ワードが受信シフトレジスタ (UxRSR) から受信バッファに転送されて、その結果、受信バッファが 4 個の文字を有する（すなわち、フル状態である）ときに割り込みが発生します。

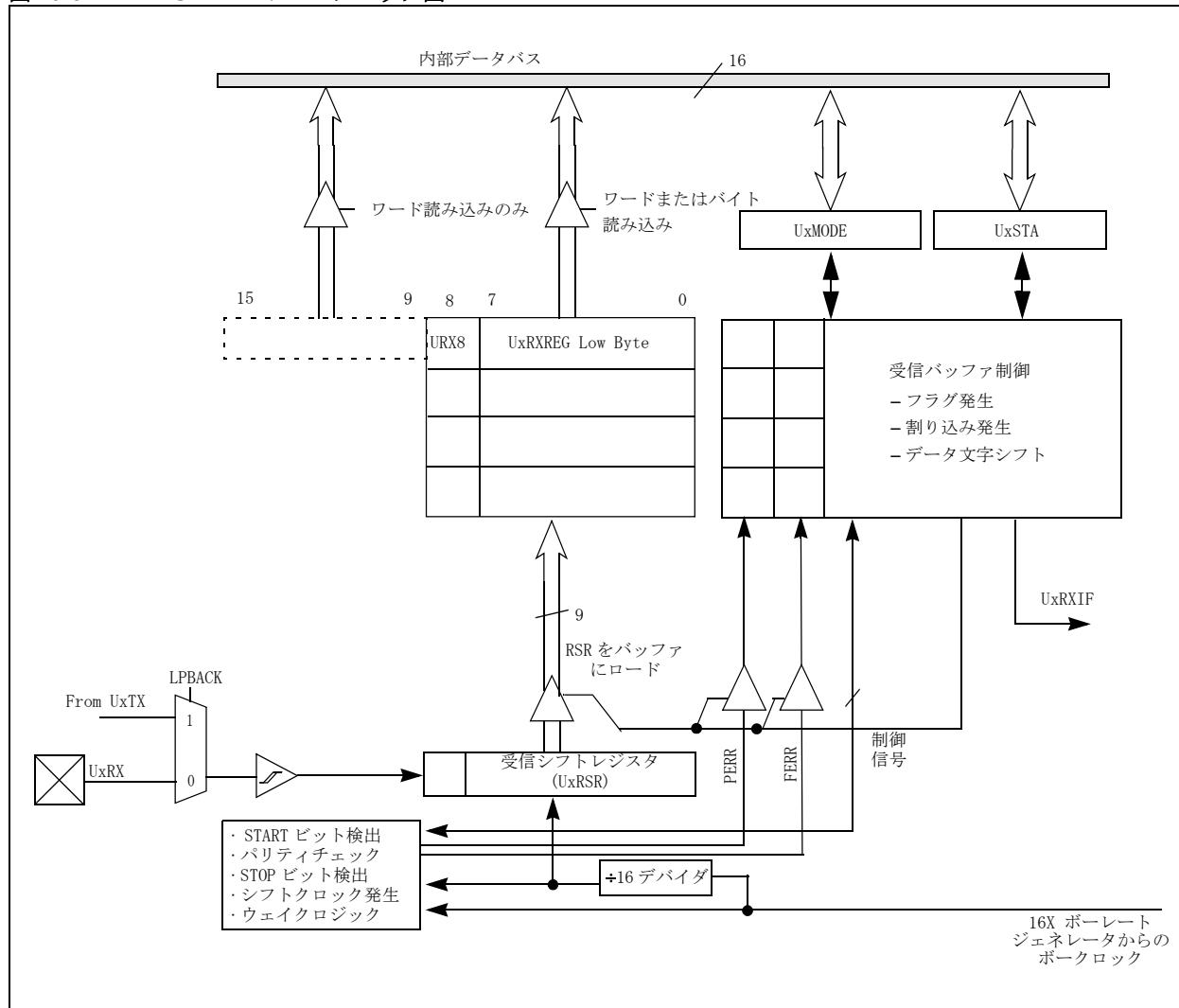
動作中に割り込みモードの変更を行うことができます。

URXDA および UxRXIF フラグビットは UxRXREG レジスタの状態を表し、RIDDLE ビット (UxSTA<4>) は UxRSR レジスタの状態を表します。RIDDLE 状態ビットは読み取り専用ビットで、レシーバが IDLE である場合（すなわち UxRSR レジスタが空のとき）セットされます。このビットには割り込みロジックは付属していないので、UxRSR が IDLE であるかを決定するためにはこのビットをポーリングする必要があります。

URXDA ビット (UxSTA<0>) は受信バッファにデータがあるかどうか、またバッファが空であるかどうかを示します。受信バッファから読み出す文字が少なくとも 1 個あるかぎり、このビットはセットされます。URXDA は読み取り専用ビットです。

図 19-5 は FUART レシーバのブロック図を示しています。

図 19-5: UART レシバブロック図



#### 19.6.4 UART 受信のセットアップ

受信セットアップの際には以下の操作を行ってください。

1. UxBRG レジスタを適切なボーレートに初期化します（セクション 19.3 「UART ボーレートレジスタ（BRG）」参照）。
2. PDSEL<1:0> (UxMODE<2:1>) および STSEL (UxMODE<0>) ビットに書き込みをして、データビットの数、STOP ビットの数、パリティ選択をセットします。
3. UEN<1:0> (UxMODE<9:8>) および RTSMD (UxMODE<11>) ビットに書き込みをして、要求されたとおりに UxCTS および UxRTS ピンを設定します。
4. 割り込みを希望する場合は、対応する割り込み有効化制御レジスタ (IEC) 内の UxRXIE ビットをセットします。対応する割り込み優先順位制御レジスタ (IPC) 内の UxRXIP<2:0> 制御ビットを使用して、割り込み優先順位を指定します。また、URXISEL<1:0> (UxSTA<7:6>) ビットに書き込みをして、受信割り込みモードを選択します。
5. UARLEN (UxMODE<15>) ビットをセットして UART モジュールを有効化します。
6. 受信割り込みは URXISEL<1:0> 制御ビットのセッティングに左右されます。受信割り込みが有効化されない場合は、URXDA ビットをポーリングすることができます。UART 受信割り込みをサポートするソフトウェアルーチン中で、UxRXIF ビットをクリアする必要があります。
7. 受信バッファからデータを読み込みます。9 ビット送信を選択した場合、1 個のワードを読み込んでください。それ以外の場合は、1 個のバイトを読み込みます。URXDA 状態ビット (UxSTA<0>) は、データがバッファで利用可能な状態であればいつもセットされます。

図 19-6: UART 受信

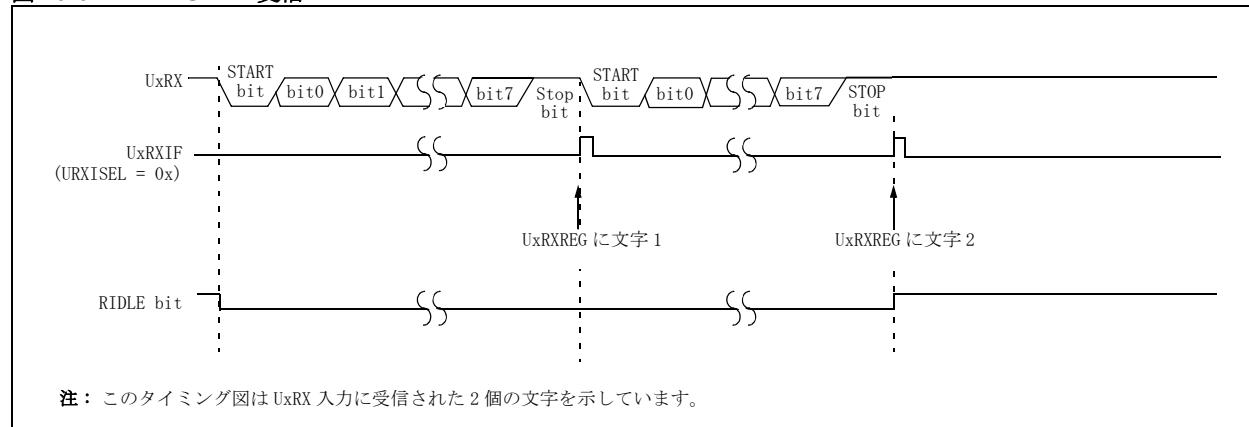
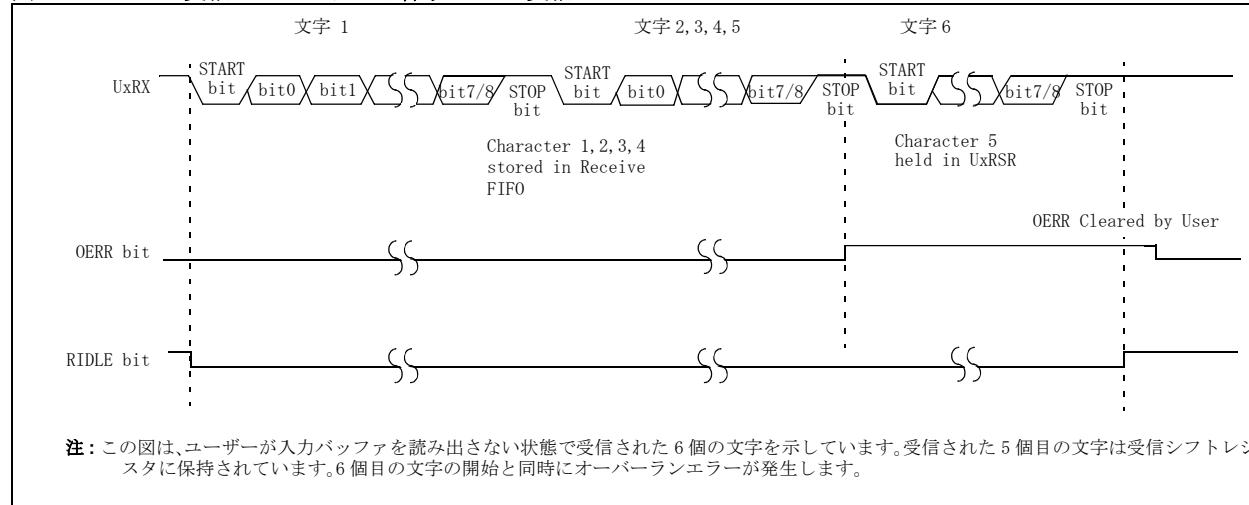


図 19-7: 受信オーバーランを伴う UART 受信



## 19.7 9 ビット通信における UART の使用

典型的なマルチプロセッサ通信プロトコルは、データバイトとアドレス / 制御バイトを区別します。一般的な方法は 9 個目のデータビットを使用して、データバイトがアドレスとデータ情報のどちらなのかを識別するというものです。9 個目のビットがセットされると、データはアドレスあるいは制御情報として処理されます。9 個目のビットがクリアされると、受信されたデータワードは先行するアドレス / 制御バイトに関連するデータとして処理されます。

プロトコルは以下のように動作します。

- 主要デバイスが 9 個目のビットのセットされた 1 個のデータワードを送信します。データワードにはスレーブデバイスのアドレスが含まれています。
- 通信チェーン中のすべてのスレーブデバイスはアドレスワードを受信し、スレーブアドレス値をチェックします。
- アドレス指定されたスレーブデバイスは、主要デバイスが続いて送信するデータバイトを受信・処理します。その他のすべてのスレーブデバイスは、続いて送信されるデータバイトを破棄します。この状態は新しいアドレスワード（9 個目のビットがセットされたもの）を受信するまで続きます。

### 19.7.1 ADDEN 制御ビット

UART レシーバにはアドレス検出モードが備わっています。このモードにより、9 個目のビットがクリアされているデータワードを無視することが可能です。9 個目のビットがクリアされたデータワードがバッファされないため、割り込みオーバーヘッドが削減されます。この機能は ADDEN ビット ( $\text{UxSTA}\langle 5 \rangle$ ) をセットすることによって有効化します。

アドレス検出モードを利用するには、UART が 9 ビットデータ対応に設定されている必要があります。レシーバが 8 ビットモードに設定されている場合、ADDEN ビットは効力を持ちません。

### 19.7.2 9 ビット送信のセットアップ

9 ビット送信のセットアップ手順は 8 ビット送信モードとほぼ同様ですが、 $\text{PDSEL}\langle 1:0 \rangle$  ( $\text{UxMODE}\langle 2:1 \rangle$ ) を ‘11’ にセットする必要があります（セクション 19.5.3 「UART 送信のセットアップ」参照）。

$\text{UxTXREG}$  レジスタにワード書き込みを行ってください（送信が開始されます）。

### 19.7.3 アドレス検出モードを利用した 9 ビット受信のセットアップ

9 ビット受信のセットアップ手順は 8 ビット受信モードとほぼ同様ですが、 $\text{PDSEL}\langle 1:0 \rangle$  ( $\text{UxMODE}\langle 2:1 \rangle$ ) は ‘11’ にセットする必要があります（セクション 19.6.4 「UART 受信のセットアップ」参照）。

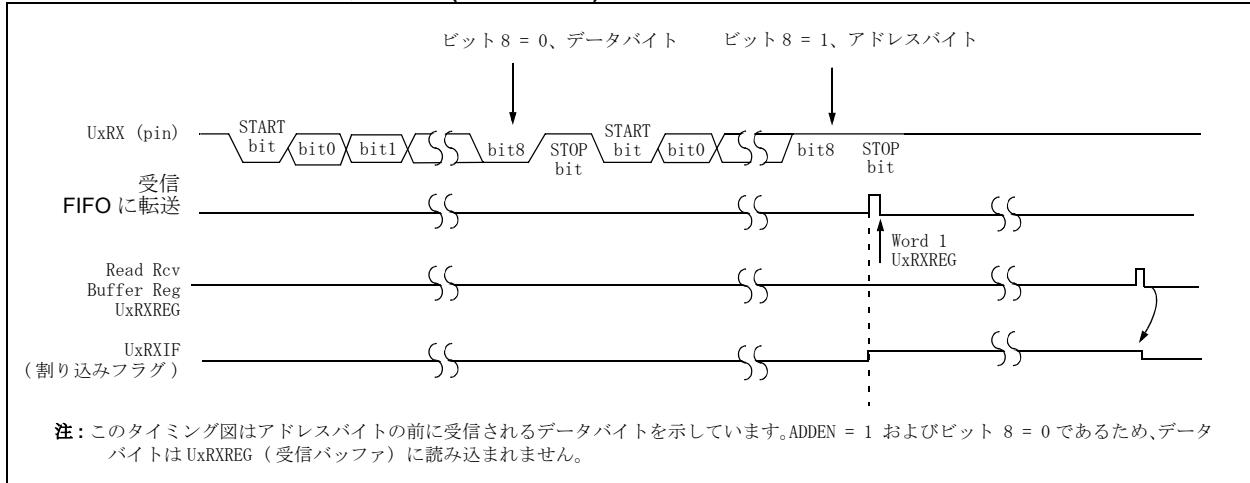
$\text{URXISEL}\langle 1:0 \rangle$  ( $\text{UxSTA}\langle 7:6 \rangle$ ) ビットに書き込むことによって受信割り込みモードを設定する必要があります。

**注：** アドレス検出モードが有効化されている場合 (ADDEN = 1)、 $\text{URXISEL}\langle 1:0 \rangle$  制御ビットを設定して、1 ワード受信毎に割り込みを発生させるようにする必要があります。受信されたデータワードはすべて、受信後ただちにアドレスマッチの有無をソフトウェア内でチェックする必要があります。

アドレス検出モード利用の手順は以下のとおりです。

- ADDEN ( $\text{UxSTA}\langle 5 \rangle$ ) ビットをセットし、アドレス検出を有効化します。1 ワード受信毎に割り込みを発生させるように、URXISEL 制御ビットが設定されていることを確認してください。
- $\text{UxRXREG}$  レジスタを読み込んで 8 ビットアドレスを 1 個ずつチェックし、デバイスがアドレスされているかどうかを決定してください。
- このデバイスがアドレスされていない場合は、受信したワードを破棄してください。
- このデバイスがアドレスされている場合は ADDEN ビットをクリアして、続いて受信するデータバイトが受信バッファに読み込まれ、CPU に割り込みをすることができる状態にしてください。データパケットが長いと予想される場合は、受信割り込みモードを変更して、割り込み間に 1 個以上のデータバイトをバッファするようにすることも可能です。
- 最後のデータバイトを受信したら ADDEN ビットをセットし、アドレスバイトだけ受信できる状態にしてください。また、URXISEL 制御ビットが 1 ワード受信毎に割り込みを発生させる設定になっていることを確認してください。

図 19-8: アドレス検出を伴う受信 (ADDEN = 1)



## 19.8 ブレーク文字の受信

レシーバは PDSEL (UxMODE<2:1>) および STSEL (UxMODE<0>) ビットにプログラムされた値に基づいて、一定数のビットタイムカウントしブレークを探します。

ブレークが 13 ビットタイムより長い場合、受信は PDSEL および STSEL によって指定されたビットタイム数の後、完了したとみなされます。URXDA ビットがセットされ、FERR がセットされ、受信 FIFO にゼロがロードされ、適切な場合、割り込みが発生して、RIDLE ビットがセットされます。

モジュールがブレーク信号を受信し、レシーバが START ビット・データビット・無効な STOP ビット（このビットは FERR をセットします）を検出した場合、レシーバは次の START ビットを探す前に有効な STOP ビットを待つ必要があります。レシーバはライン上のブレーク状態を次の START ビットであると仮定することはできません。ブレーク信号は FERR ビットをセットした状態で、すべての ‘0’ を含む文字とみなされます。ブレーク文字はバッファにロードされます。STOP ビットが受信されるまで、それ以上の受信は行われません。STOP ビットが受信された際、RIDLE が高くなるセットされることに注意してください。

## 19.9 初期設定

例 19-2 は 8 ビットモードでのトランスマッタ / レシーバの初期化ルーチンです。例 19-3 は 9 ビットアドレス検出モードでのアドレス可能 UART の初期化を示しています。いずれの例においても、UxBRG レジスタにロードされる値は望ましいボーレートとデバイス周波数に依存します。

### 例 19-2: 8 ビット送信 / 受信 (UART1)

```
MOV    #baudrate,W0      ; ボーレートセット
MOV    W0,U1BRG

BSET   IPC2,#U1TXIP2    ; UART TX 割込み優先順位セット
BCLR   IPC2,#U1TXIP1    ;
BCLR   IPC2,#U1TXIP0    ;
BSET   IPC2,#U1RXIP2    ; UART RX 割込み優先順位セット
BCLR   IPC2,#U1RXIP1    ;
BCLR   IPC2,#U1RXIP0    ;

CLR    U1STA

MOV    #0x8800,W0        ; UART を 8 ビットデータ対応に有効化、
                        ; パリティなし、1STOP ビット、
                        ; フロー制御なし、ウェイクアップなし
MOV    W0,U1MODE

BSET   U1STA,#UTXEN     ; 送信有効化

BSET   IEC0,#U1TXIE      ; 送信割り込み有効化
BSET   IEC0,#U1RXIE      ; 受信割り込み有効化
```

### 例 19-3: 9 ビット 送信 / 受信 (UART1)、アドレス検出モード有効

```
MOV    #baudrate,W0      ; ボーレートセット
MOV    W0,U1BRG

BSET   IPC2,#U1TXIP2    ; UART TX 割り込み優先順位セット
BCLR   IPC2,#U1TXIP1    ;
BCLR   IPC2,#U1TXIP0    ;
BSET   IPC2,#U1RXIP2    ; UART RX 割り込み優先順位セット
BCLR   IPC2,#U1RXIP1    ;
BCLR   IPC2,#U1RXIP0    ;

BSET   U1STA,#ADDEN     ; アドレス検出有効化

MOV    #0x8883,W0        ; UART1 を 9 ビットデータ対応に有効化、
                        ; パリティなし、1 STOP ビット、
                        ; フロー制御なし、ウェイクアップ有効化
MOV    W0,U1MODE

BSET   U1STA,#UTXEN     ; 送信有効化

BSET   IEC0,#U1TXIE      ; 送信割り込み有効化
BSET   IEC0,#U1RXIE      ; 受信割り込み有効化
```

## 19.10 UART のその他の機能

### 19.10.1 ループバックモードでの UART

LPBACK ビットをセットすると、この特別なモードが有効化されます。このモードにおいては UxTX 出力が UxRX 入力に内部接続されます。ループバックモード対応に設定された場合、UxRX ピンは内部 UART 受信ロジックから切断されます。ただし、UxTX ピンは通常通り機能します。このモードを選択するには次の手順に従ってください。

1. 希望する操作モードに **UART** を設定してください。
2. LPBACK = 1 をセットして、ループバックモードを有効化してください。
3. セクション 19.5 「UART トランスマッタ」の説明に従って送信を有効化してください。

表 19-2 に示されるように、ループバックモードは **UEN<1:0>** ビットに依存します。

表 19-2: ループバックモードピン機能

<b>UEN&lt;1:0&gt;</b>	<b>Pin Function, LPBACK = 1</b>
00	UxRX 入力は UxTX に接続 ; UxTX ピン機能 ; UxRX ピン無視 ; UxCTS/UxRTS 使用せず
01	UxRX 入力は UxTX に接続 ; UxTX ピン機能 ; UxRX ピン無視 ; UxRTS ピン機能 , UxCTS 使用せず
10	UxRX 入力は UxTX に接続 ; UxTX ピン機能 ; UxRX ピンは無視 ; UxRTS ピン機能 , UxCTS 入力は UxRTS に接続 ; UxCTS ピンは無視
11	UxRX 入力は UxTX に接続 ; UxTX ピン機能 ; UxRX ピン無視 ; BCLK ピン機能 ; UxCTS/UxRTS 使用せず

### 19.10.2 オートボーサポート

受信された文字のボーレートをシステムが決定できるように、UxRX 入力を選択された入力キャプチャチャネルに内部接続することができます。ABAUD ビット (**UXMODE<5>**) がセットされると、UxRX ピンが入力キャプチャチャネルに内部接続されます。ICx ピンは入力キャプチャチャネルから切断されます。

オートボーサポートに利用される入力キャプチャチャネルはデバイス特有です。詳細はデバイスデータシートを参照してください。

このモードは **UART** が有効化され (**UARTEN = 1**) 、ループバックモードが無効化された (**LPBACK = 0**) 場合にのみ有効です。また、START ビットの下降および上昇エッジを検出するように、キャプチャモジュールをプログラミングする必要があります。

## 19.11 CPU SLEEP および IDLE モードの際の UART 動作

UART は SLEEP モードでは機能しません。送信中に SLEEP モードに入ると送信は中断され、UxTX ピンはロジック ‘1’ に移行します。同様に、受信中に SLEEP モードに入ると受信は中断されます。

START ビットの検出に際して、dsPIC デバイスを SLEEP モードから随意でウェイクアップさせるために UART を利用することができます。WAKE ビット (**UXSTA<7>**) がセットされ、デバイスが SLEEP モードで、UART 受信割り込みが有効化されている (**UXRXIE = 1**) 場合、UxRX ピンの下降エッジが受信割り込みを発生させます。受信割り込み選択モードビット (**URXISEL**) はこの機能に対して影響しません。ウェイクアップ割り込みを発生させるには、**UARTEN** ビットをセットする必要があります。

デバイスが IDLE モードに入った場合、モジュールが動作を中止するか、また IDLE モードでもモジュールが通常通り動作を続けるかは、USIDL ビット (**UXMODE<13>**) によって選択されます。USIDL = 0 の場合、モジュールは IDLE モードの間も通常動作を続けます。USIDL = 1 の場合、IDLE モードでモジュールは停止します。進行中の送信・受信はすべて中断されます。

## 19.12 UART モジュール関連レジスタ

表 19-3: UART1 関連レジスタ

SFR名	ピット15	ピット14	ピット13	ピット12	ピット11	ピット10	ピット9	ピット8	ピット7	ピット6	ピット5	ピット4	ピット3	ピット2	ピット1	ピット0	RESET状態
U1MODE	UARTEN	—	USIDL	—	reserved	ALTI0	reserve d	reserved	WAKE	LPBACK	ABAUD	—	—	PDSEL<1:0>	STSEL	0000 0000 0000 0000	
U1STA	UTXISEL	—	—	—	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL<1:0>	ADDEN	RIDLE	PERR	FERR	OERR	URXDA	0000 0001 0001 0000	
U1TXREG	—	—	—	—	—	—	—	UTX8	トランシミットレジスタ							0000 0000 0000 0000	
U1RXREG	—	—	—	—	—	—	—	URX8	リシープレジスタ							0000 0000 0000 0000	
U1BRG	ボーレートジェネレータプレスケーラー															0000 0000 0000 0000	
IFS0	CNIF	BCLIF	I2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0	0000 0000 0000 0000
IECO	CNIE	BCLIE	I2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IPC2	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>			0100 0100 0100 0100

注: UART1 関連レジスタは参考のために示してあります。他の UART モジュール関連レジスタについてはデバイスシートを参照してください。

## 19.13 設計の秘訣

**質問 1:** *UART* で送信したデータが正しく受信されません。原因は何でしょう？

**回答 :** 受信エラーの原因として最もよくあるのは、UART ポーレートジェネレータに対して正しくない値が計算されているということです。UxBRG レジスタに書き込まれている値が正しいことを確認してください。

**質問 2:** *UART* 受信ピンの信号は正しいのに、フレーミングエラーが発生します。なぜでしょうか？

**回答 :** 以下の制御ビットが正しくセットアップされているか確認してください。

- UxBRG : UART ポーレートレジスタ
- PDSEL<1:0> : パリティおよびデータサイズ選択ビット
- STSEL : STOP ビット選択

## 19.14 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正し、制限を設けて（必要な場合）使用可能です。現状、UART に関する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 #
表の読み込みと書き込みを実行するシリアルポートユーティリティ	AN547

**注：** dsPIC30F ファミリーデバイスの他のアプリケーションノートおよびコードの例に関しては Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧ください。

## 19.15 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

この版は、dsPIC30F UART モジュールの完全な説明が追加されています。

注意：



**MICROCHIP**

## 第 20 章. シリアル周辺装置インターフェース (SPI)

### ハイライト

このセクションには以下の主要な項目が含まれます。

20.1 序章 .....	20-2
20.2 ステータスおよびコントロールレジスタ .....	20-4
20.3 動作モード .....	20-7
20.4 SPI マスター モードクロック周波数 .....	20-19
20.5 省電力モードでの動作 .....	20-20
20.6 SPI モジュールに関連の特殊機能レジスタ .....	20-22
20.7 関連するアプリケーションノート .....	20-23
20.8 改訂履歴 .....	20-24

20

シリアル周辺装置  
インターフェース  
(SPI)

## 20.1 序章

シリアル周辺装置インターフェース (SPI) モジュールは、他の周辺装置またはマイクロコントローラデバイスと通信するために役立つ同期シリアルインターフェースです。この周辺装置デバイスとしてはシリアル EEPROMs、シフトレジスタ、ディスプレイデバイス、A/D 変換器などがあります。SPI モジュールは、モトローラ社の SPI および SIOP インターフェースと互換性があります。

dsPIC30F ファミリーにより单一のデバイス上に 1 つまたは 2 つの SPI モジュールが提供されます。SPI1 と SPI2 は機能的に同じです。SPI1 モジュールがすべてのデバイスで利用できるのに対し、SPI2 モジュールは多くのより上位のピンカウントパッケージ (64 ピン以上) で利用できます。

**注:** この章では、SPI モジュールは SPIx として共に参照されるか、または SPI1 と SPI2 として個別に参照されます。特別関数レジスタは類似の表記に従います。例えば、SPIxCON により、SPI1 または SPI2 モジュールのためのコントロールレジスタが参照されます。

SPI シリアルポートは以下の特別機能レジスタ (SFR) から成り立っています。

- SPIxBUF : 送信されるバッファデータおよび受信されるデータに使用される SFR スペース内のこのアドレスは、SPIxTXB と SPIxRXB レジスタにより共有されます。
- SPIxCON : 多様な動作モードを設定するためのコントロールレジスタ。
- SPIxSTAT : 多様なステータス条件を示すステータスレジスタ。

さらに、16 ビットシフトレジスタ、SPIxSR があり、これはマップされたメモリではありません。SPI ポートの送受信のデータをシフトするために使われます。

SFR にマップされたメモリである SPIxBUF は、SPI データ受信・送信レジスタです。内部的には、SPIxBUF レジスタは実際には SPIxTXB と SPIxRXB の 2 つの個別レジスタをから成り立っています。受信バッファレジスタ、SPIxRXB および送信バッファレジスタ SPIxTXB は、2 つの一方向 16 ビットレジスタです。この 2 つのレジスタは、SPIxBUF と名付けられた SFR アドレスを共有します。ユーザーが SPIxBUF に送信されるデータを書き込めば、内部的にそのデータは SPIxTXB レジスタに書き込まれます。同様に、ユーザーが SPIxBUF から受信したデータを読み出せば、内部的にそのデータは SPIxRXB レジスタから読み込まれます。この送受信のダブルバッファリング操作により、バックグラウンドにおける連続したデータ転送が可能になります。送信受信は同時に発生します。

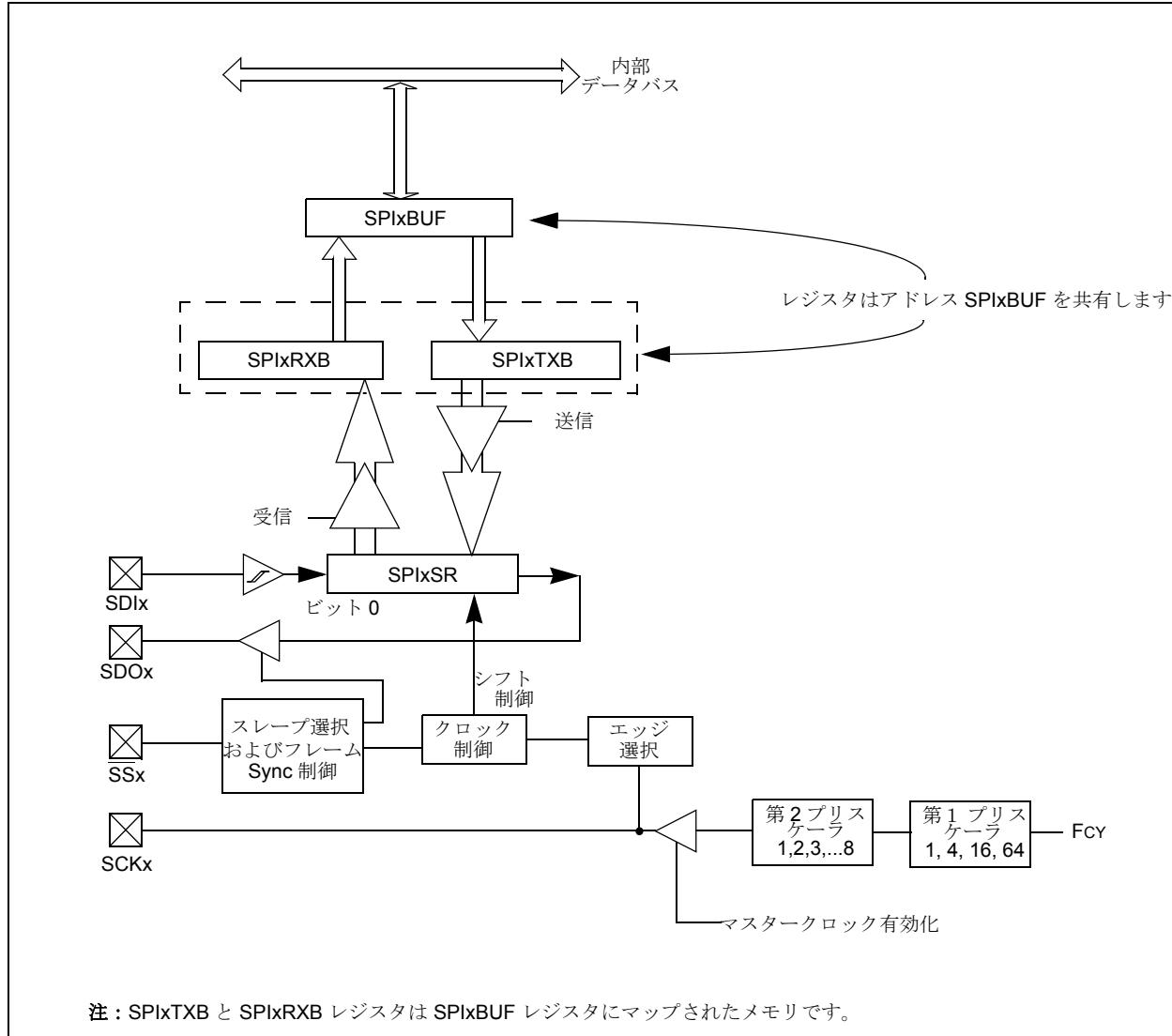
**注:** ユーザーは SPIxTXB レジスタに直接書き込んだり、SPIxRXB レジスタから直接読み出したりすることはできません。すべての書き込み、読み込みは SPIxBUF レジスタ上で実行されます。

SPI シリアルインターフェースは以下の 4 つのピンから成り立っています。

- SDIx: シリアルデータ入力
- SDOx: シリアルデータ出力
- SCKx: シフトクロック入力または出力
- SSx: 負論理のスレーブ選択またはフレーム同期化入出力パルス

**注:** SPI モジュールは 3 または 4 ピンを使用する動作をするために構成できます。3 ピンモードでは、SSx ピンは使われません。

図 20-1: SPI モジュールブロック図



## 20.2 ステータスおよびコントロールレジスタ

レジスタ 20-1: SPIxSTAT: SPI ステータスおよびコントロールレジスタ

上位バイト:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
SPIEN	—	SPIISIDL	—	—	—	—	—
ビット 15							ビット 8

下位バイト:							
U-0	R/W-0	U-0	U-0	U-0	U-0	R-0	R-0
HS	—	SPIROV	—	—	—	SPITBF	SPIRBF
ビット 7							ビット 0

- ビット 15 **SPIEN:** SPI 有効化ビット  
1 = SPI モジュールを有効化する。SCKx、SDOx、SDIx および SSx をシリアルポートピンとして構成する  
0 = モジュールが無効化されます。
- ビット 14 未実装: ‘0’ が読み込まれます
- ビット 13 **SPIISIDL:** IDLE モードビットでの停止  
1 = デバイスが IDLE モードに入った時のモジュール動作停止  
0 = IDLE モードでもモジュール動作継続
- ビット 12-7 未実装: ‘0’ が読み込まれます
- ビット 6 **SPIROV:** オーバーフロー発生ビット  
1 = 新しいバイト・ワードは完全に受信され破棄されます。ユーザーソフトウェアによって SPIxBUF レジスタにある以前のデータが読み出されていない。  
0 = オーバーフローは発生しませんでした
- ビット 5-2 未実装: ‘0’ が読み込まれます
- ビット 1 **SPITBF:** SPI 送信バッファフルステータスビット  
1 = まだ送信開始されておらず SPIxTXB にデータが残っている  
0 = 送信は開始され、SPIxTXB は空です  
CPU が SPIxBUF に書き込むことにより、SPIxTXB にロードされると自動的にハードウェアでセットされる。SPIx モジュールにより SPIxTXB から SPIxSR までデータが転送されると、ハードウェアで自動的にクリアされます。
- ビット 0 **SPIRBF:** SPI 受信バッファフルステータスビット  
1 = 受信完了、SPIxRXB にデータがあります  
0 = 受信未完了、SPIxRXB は空です  
SPIx モジュールにより SPIxSR から SPIxRXB までデータが転送されると、ハードウェアで自動的に設定されます。  
SPIxBUF を読み出す、つまり SPIxRXB のデータが CPU に読み込まれると、ハードウェアで自動的にクリアされます。

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、‘0’ が読み込まれます

HC = ハードウェアによって  
クリアされます

HS = ハードウェアに  
よって設定されます

-n = RESET での値

‘1’ = ビットがセット  
されます ‘0’ = ビットはクリア  
されます x = ビットは不定です

## レジスタ 20-2: SPIxCON: SPIx コントロールレジスタ

上位バイト:							
U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	FRMEN	SPIFSD	—	DISSDO	MODE16	SMP	CKE
ビット 15							ビット 8

下位バイト:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SSEN	CKP	MSTEN	SPRE<2:0>	SPRE<2:0>	PPRE<1:0>	PPRE<1:0>	PPRE<1:0>
ビット 7							ビット 0

- ビット 15 未実装: ‘0’ が読み込まれます
- ビット 14 **FRMEN:** フレーム化 SPI サポートビット  
1 = フレーム化サポートを有効化する  
0 = フレーム化 SPI サポート無効化
- ビット 13 **SPIFSD:** SSx ピン方向指定ビット  
1 = フレーム同期パルス入力 (スレーブ)  
0 = フレーム同期パルス出力 (マスター)
- ビット 12 未実装: ‘0’ が読み込まれます
- ビット 11 **DISSDO:** SDOx ピン無効化ビット  
1 = SDOx ピンはモジュールには使用されません。ピンは対応するポートレジスタにより制御されます。  
0 = SDOx ピンはモジュールにより制御されます。
- ビット 10 **モード 16:** ワード・バイト転送選択ビット  
1 = 転送はワード単位 (16 ビット) です  
0 = 転送はバイト単位 (8 ビット) です
- ビット 9 **SMP:** SPI データ入力サンプルフェーズビット  
マスター mode:  
1 = データ出力時間の終了時に入力データをサンプルする。  
0 = データ出力時間の中央で入力データをサンプルする。  
スレーブ mode:  
SPI がスレーブモードで使用される場合 SMP はクリアされる必要があります。
- ビット 8 **CKE:** SPI クロックエッジ選択ビット  
1 = アクティブクロック状態から IDLE クロック状態に遷移するエッジでシリアル出力データが変化します (ビット 6 をご参照ください)  
0 = IDLE 状態からアクティブクロック状態に遷移するエッジでシリアル出力データが変化します (ビット 6 をご参照ください)
- 注: CKE ビットはフレーム化 SPI モードでは使用されません。ユーザーはフレーム化 SPI モード (FRMEN = 1) のときはこのビットを ‘0’ に設定する必要があります。
- ビット 7 **SSEN:** SS ピン有効化 (スレーブモード) ビット  
1 = スレーブモードで SS ピンを使う。  
0 = SS ピンは使わない。対応するポートレジスタで制御される。
- ビット 6 **CKP:** クロック極性選択ビット  
1 = クロックの IDLE 状態は High レベルで、アクティブ状態は Low レベルです  
0 = クロックの IDLE 状態は Low レベルで、アクティブ状態は High レベルです
- ビット 5 **MSTEN:** マスター mode 有効化ビット  
1 = マスター mode  
0 = スレーブモード

## レジスタ 20-2: SPIxCON: SPIx コントロールレジスタ (継続)

ビット 4-2 **SPRE<2:0>**: 第 2 プリスケール (マスター モード) ビット  
(サポートされている設定: 1 : 1、2 : 1 から 8 : 1、すべて含まれます)  
111 = 第 2 プリスケール 1:1  
110 = 第 2 プリスケール 2:1  
...  
000 = 第 2 プリスケール 8:1

ビット 1-0 **PPRE<1:0>**: 第 1 プリスケール (マスター モード) ビット  
11 = 第 1 プリスケール 1:1  
10 = 第 1 プリスケール 4 : 1  
01 = 第 1 プリスケール 16:1  
00 = 第 1 プリスケール 64:1

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、'0' が読み込まれます

-n = POR での値

'1' = ビットがセット  
されます

'0' = ビットはクリア  
されます

x = ビットは不定です

## 20.3 動作モード

SPI モジュールには以下のサブセクションで説明される柔軟な動作モードがあります。

- 8 ビットおよび 16 ビットデータ送・受信
- マスターおよびスレーブモード
- フレーム化 SPI モード

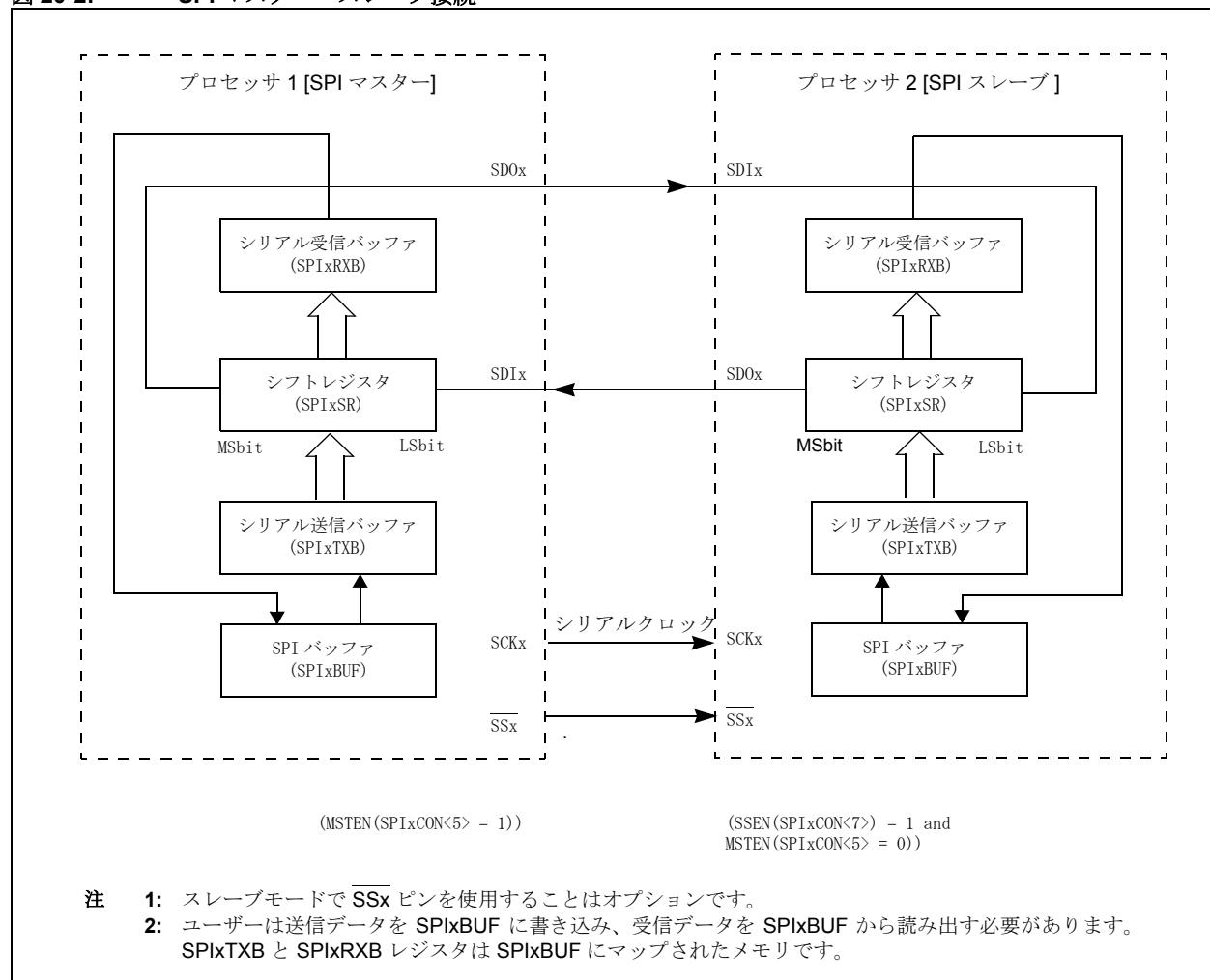
### 20.3.1 8 ビット対 16 ビット動作

コントロールビット、MODE16 (SPIxCON<10>) により、モジュールは 8 ビットモードまたは 16 ビットモードで情報のやりとりができます。送受信されるビットの数を除き、機能は各モードで同じです。さらに、以下について注意する必要があります。

- このモジュールは MODE16 (SPIxCON<10>) ビットの変更時にリセットされます。その結果、このビットは通常動作中に変更してはいけません。
- 送信する場合には、データは 8 ビット動作のときは SPIxSR のビット 7 から、16 ビット動作のときは SPIxSR のビット 15 から送信されます。どちらのモードでも、データは SPIxSR のビット ‘0’ にシフトされます。
- SCKx ピンのクロックは、8 ビットモードのときは 8 クロックパルスが、16 ビットモードのときは 16 クロックパルスが、データの入出力のために必要とされます。

### 20.3.2 マスター およびスレーブモード

図 20-2: SPI マスター・スレーブ接続



## 20.3.2.1 マスター モード

マスター モードに SPI モジュールを設定するために以下のステップが必要とされます。

1. 割込みを使う場合、
  - 個々の IFSn レジスタの SPIxIF ビットをクリアします。
  - 個々の IECn レジスタの SPIxIE ビットをセットします。
  - 個々の IPCn レジスタの SPIxIP ビットを書き込みます。
2. MSTEN (SPIxCON<5>) = 1 で SPIxCON レジスタに望ましい設定を書き込みます。
3. SPIROV ビット (SPIxSTAT<6>) をクリアします。
4. SPIEN ビット (SPIxSTAT<15>) をセットすることで、SPI 動作を有効化します。
5. SPIxBUF レジスタに送信されるデータを書き込みます。送信（および受信）はデータが SPIxBUF レジスタに書き込まれ次第開始されます。

マスター モードでは、システムロックは分周され、それからシリアルロックとして使用されます。分周比は PPRE<1 : 0> (SPIxCON<1 : 0>) および SPRE<1 : 0> (SPIxCON<4 : 2>) での設定に基づいています。シリアルロックは SCKx ピンを経由したスレーブデバイスへの出力です。クロックパルスは、送信するデータがある時に生成されるだけです。詳細はセクション 20.4 「SPI マスター モード クロック周波数」をご参照ください。

CKP および CKE ビットによりどちらのクロックのエッジでデータ送信が発生するかが決定されます。

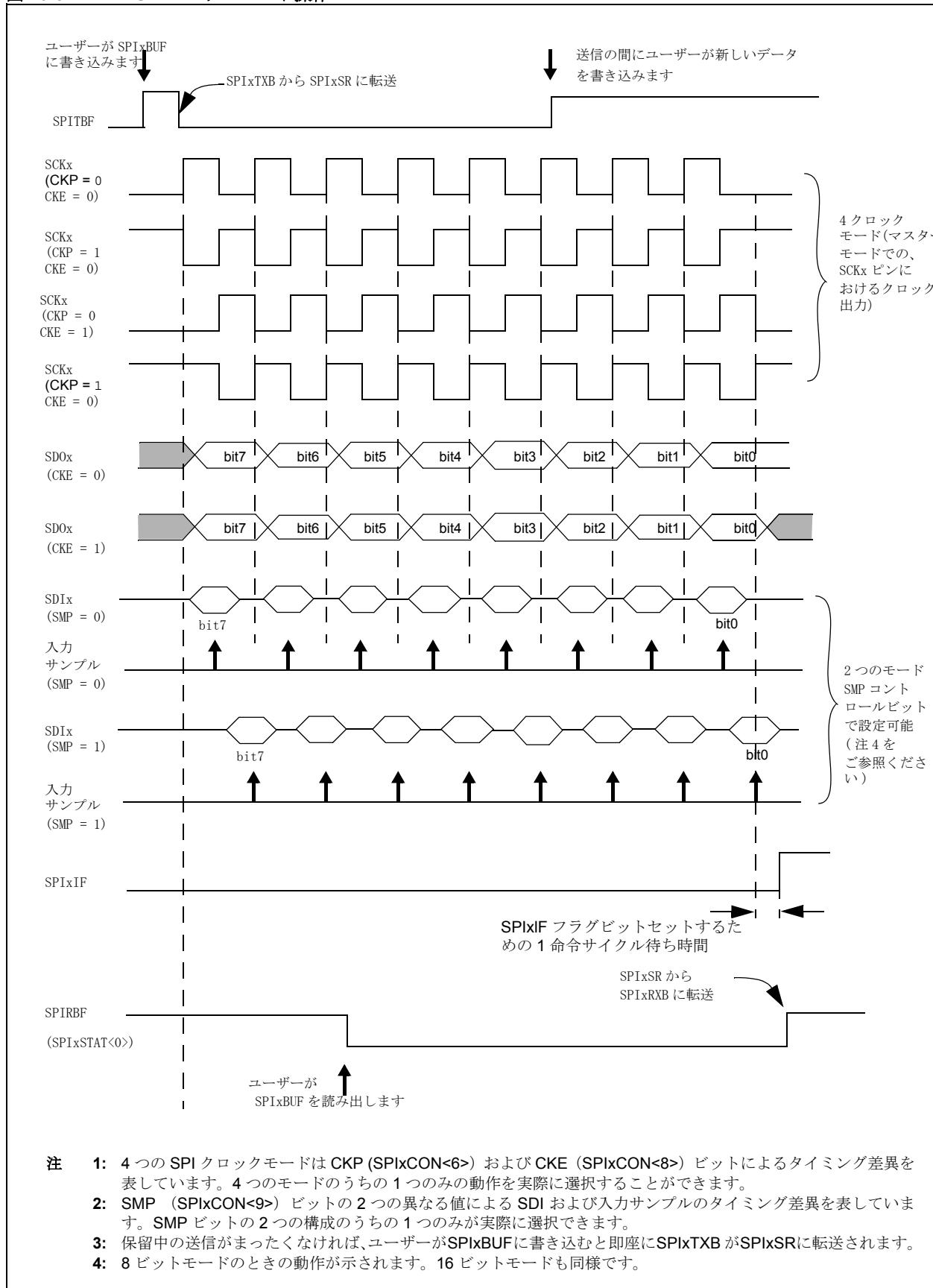
送信されるデータも受信されたデータも、両方ともそれぞれ SPIxBUF レジスタに書き込まれたり、SPIxBUF レジスタから読み込まれたりします。

マスター モードでの SPI モジュール動作が以下に示されています。

1. マスター モードにモジュールがいったん設定され有効化されると、送信されるデータは SPIxBUF レジスタに書き込みます。SPITBF (SPIxSTAT<1>) がセットされます。
2. SPIxTXB の内容はソフトレジスタである SPIxSR に移され、SPITBF ビットはモジュールによりクリアされます。
3. 一連の 8/16 クロックパルスによって SPIxSR から SDIx ピンに送信データの 8/16 ビットがシフトアウトされ、同時に SDIx ピンのデータが SPIxSR にシフトインされます。
4. 転送が完了すると、以下のイベントが発生します。
  - 割込みフラグビットである SPIxIF がセットされます。SPI 割込みはビット SPIxIE がセットされると有効化されます。SPIxIF フラグはハードウェアでは自動的にクリアされません。
  - また、継続中の送受信動作が完了すると、SPIxSR の内容は SPIxRXB レジスタに移されます。
  - SPIRBF (SPIxSTAT<0>) がモジュールによりセットされ、受信バッファが一杯であることが示されます。ユーザーコードにより SPIxBUF がいったん読み込まれると、ハードウェアにより SPIRBF ビットがクリアされます。
5. SPI モジュールが SPIxSR から SPIxRXB へデータを転送しなければならない時、SPIRBF ビットがセット（受信バッファは一杯）されていると、そのモジュールにより SPIROV (SPIxSTAT<6>) ビットがセットされ、オーバーフロー状態が示されます。
6. 送信されるデータは、SPITBF (SPIxSTAT<1>) がクリアされている限り、いつでもユーザー ソフトウェアにより SPIxBUF に書き込めます。SPIxSR により先に書き込まれたデータがシフトアウトされている間も、書き込みをすることができ、これにより連続送信が可能になっています。

**注：** SPIxSR レジスタはユーザーによって直接書き込むことはできません。SPIxSR レジスタへの書き込みはすべて SPIxBUF レジスタを通じて実行されます。

図 20-3: SPI マスター モード操作



## 20.3.2.2 スレープモード

スレープモードに SPI モジュールを設定するために以下のステップが必要とされます。

1. SPIxBUF レジスタをクリアしてください。
2. 割込みを使う場合、
  - 各々の IFSn レジスタの SPIxIF ビットをクリアしてください。
  - 各々の IECn レジスタの SPIxIE ビットをセットしてください。
  - 各々の IPCn レジスタの SPIxIP ビットを書き込んでください。
3. MSTEN(SPIxCON<5>) = 0 で SPIxCON レジスタに望ましい設定を書き込んでください。
4. SMP ビットをクリアしてください。
5. CKE をセットしたときは、SSEN ビットもセットする必要があります、これにより  $\overline{SSx}$  ピンが有効化されます。
6. SPIROV ビット (SPIxSTAT<6>) をクリアしてください、そして、
7. SPIEN ビット (SPIxSTAT<15>) をセットすることで、SPI 動作を有効化してください。

スレープモードでは、データは SCKx ピンに入力される外部クロックパルスにより送受信動作を実行します。CKP (SPIxCON<6>) および CKE (SPIxCON<8>) ビットにより、どちらのクロックのエッジでデータ送信が発生するかが決定されます。

送信されるデータも受信されたデータも、両方ともそれぞれ SPIxBUF レジスタに書き込まれたり、SPIxBUF レジスタから読み込まれたりします。

このモジュールのその他の動作は、マスター mode でのモジュールとすべて同じとなっています。

スレープモードで提供されるいくつかの追加機能は以下です。

**スレープ選択同期化：**  $\overline{SSx}$  ピンにより同期スレープモードが可能になります。SSEN (SPIxCON<7>) ビットがセットされると、 $\overline{SSx}$  ピンが Low の状態にドライブされる時のみ、スレープモードで送受信が有効化されます。ポート出力、その他の周辺出力は、 $\overline{SSx}$  ピンが入力として機能できるように無効化する必要があります。SSEN ビットがセットされ、 $\overline{SSx}$  ピンが High でドライブされると、SDOx ピンはもはやドライブされず、モジュールが送信中であってもハイインピーダンス状態になります。中断された送信は、次回  $\overline{SSx}$  ピンが Low にドライブされたとき、SPIxTXB レジスタに保留されているデータを使って再試行されます。SSEN ビットがセットされていなければ、スレープモードでのモジュール動作は  $\overline{SSx}$  ピンにより影響を受けません。

**SPITBF ステータスフラグ操作：** SPITBF (SPIxSTAT<1>) ビットの機能は、設定されたスレープモードで異なっています。スレープモードの設定による SPITBF の機能は、以下に記載されています。

1. SSEN (SPIxCON<7>) がクリアされれば、SPIxBUF がユーザーコードによりロードされると SPITBF がセットされます。モジュールにより SPIxTXB が SPIxSR に転送されると、クリアされます。これはマスター mode での SPITBF ビット機能に類似しています。
2. SSEN (SPIxCON<7>) がセットされれば、SPIxBUF がユーザーコードによりロードされると SPITBF がセットされます。ただし、SPIx モジュールによりデータ送信が完了されたときにのみクリアされます。 $\overline{SSx}$  ピンが High になると送信中断され、後ほど再送されます。すべてのビットがレシーバに送信されるまで、各データワードは SPIxTXB に保持されます。

**注：** モジュールタイミング仕様に合わせるために、CKE = 1 の時、 $\overline{SSx}$  ピンはスレープモードで有効化される必要があります。（詳細については図 20-6 を参照してください。）

図 20-4: SPI スレーブモード動作: スレーブ選択ピン無効化

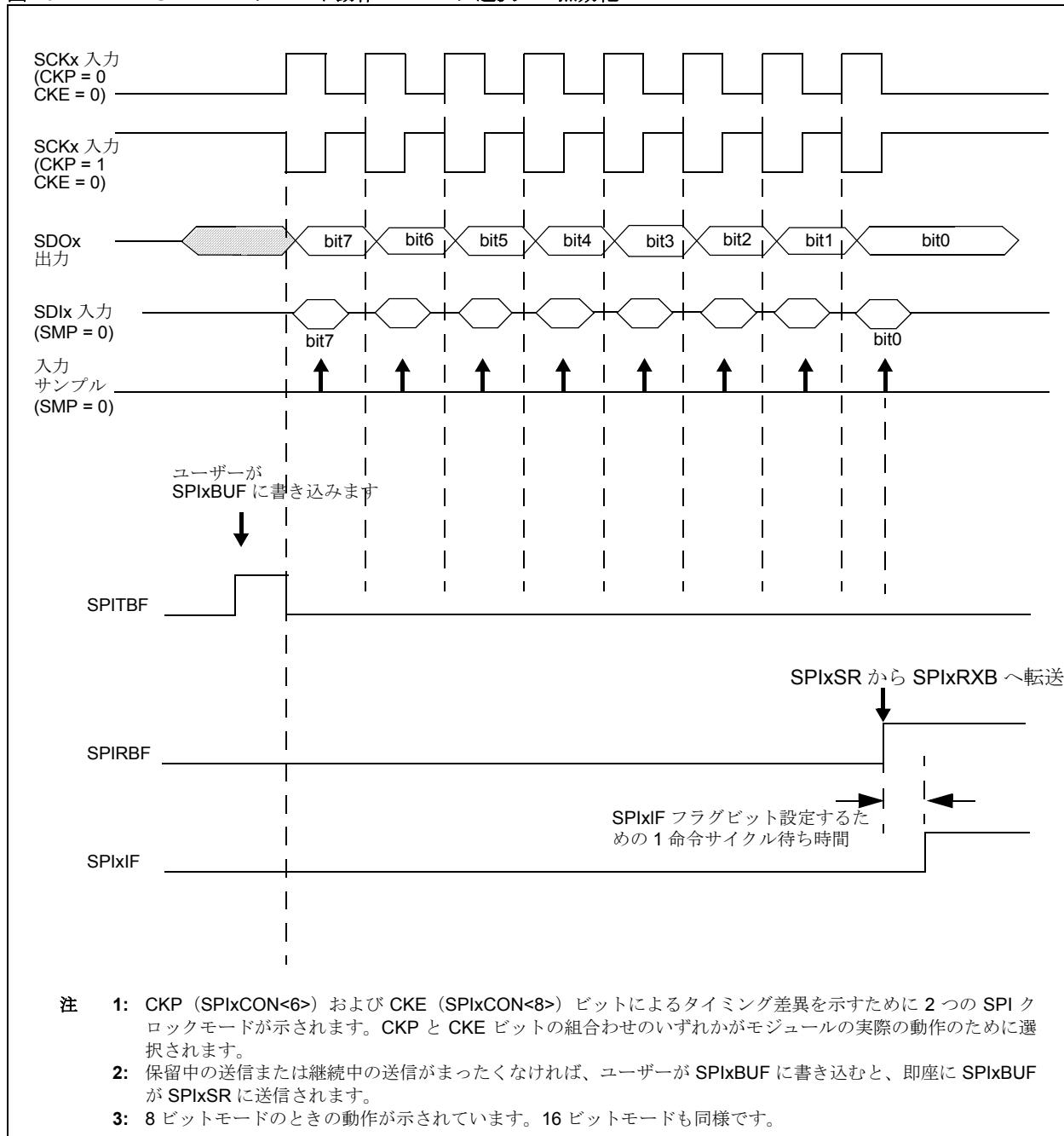


図 20-5: 有効化されたスレーブ選択ピンを伴う SPI スレーブモード操作

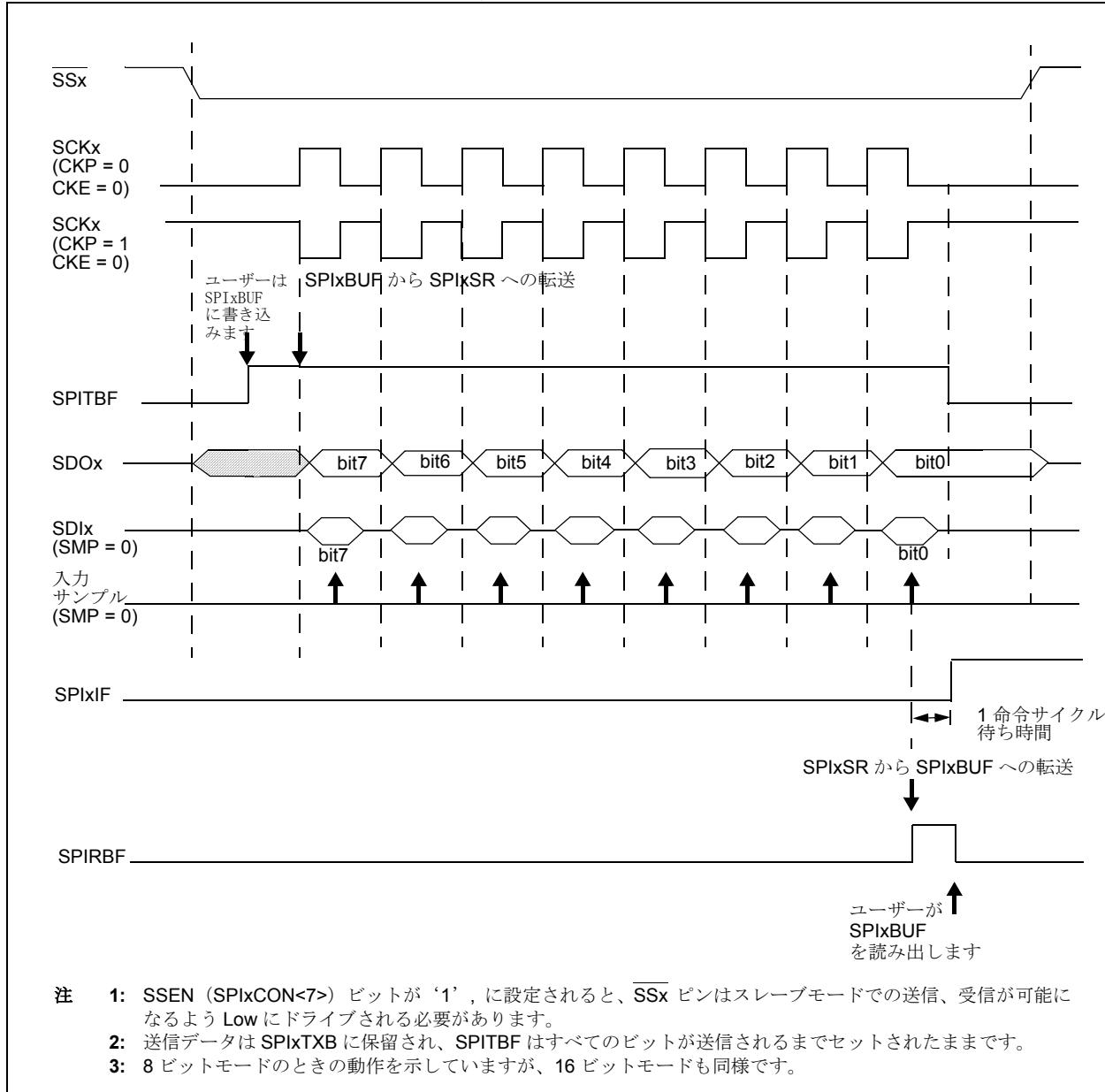
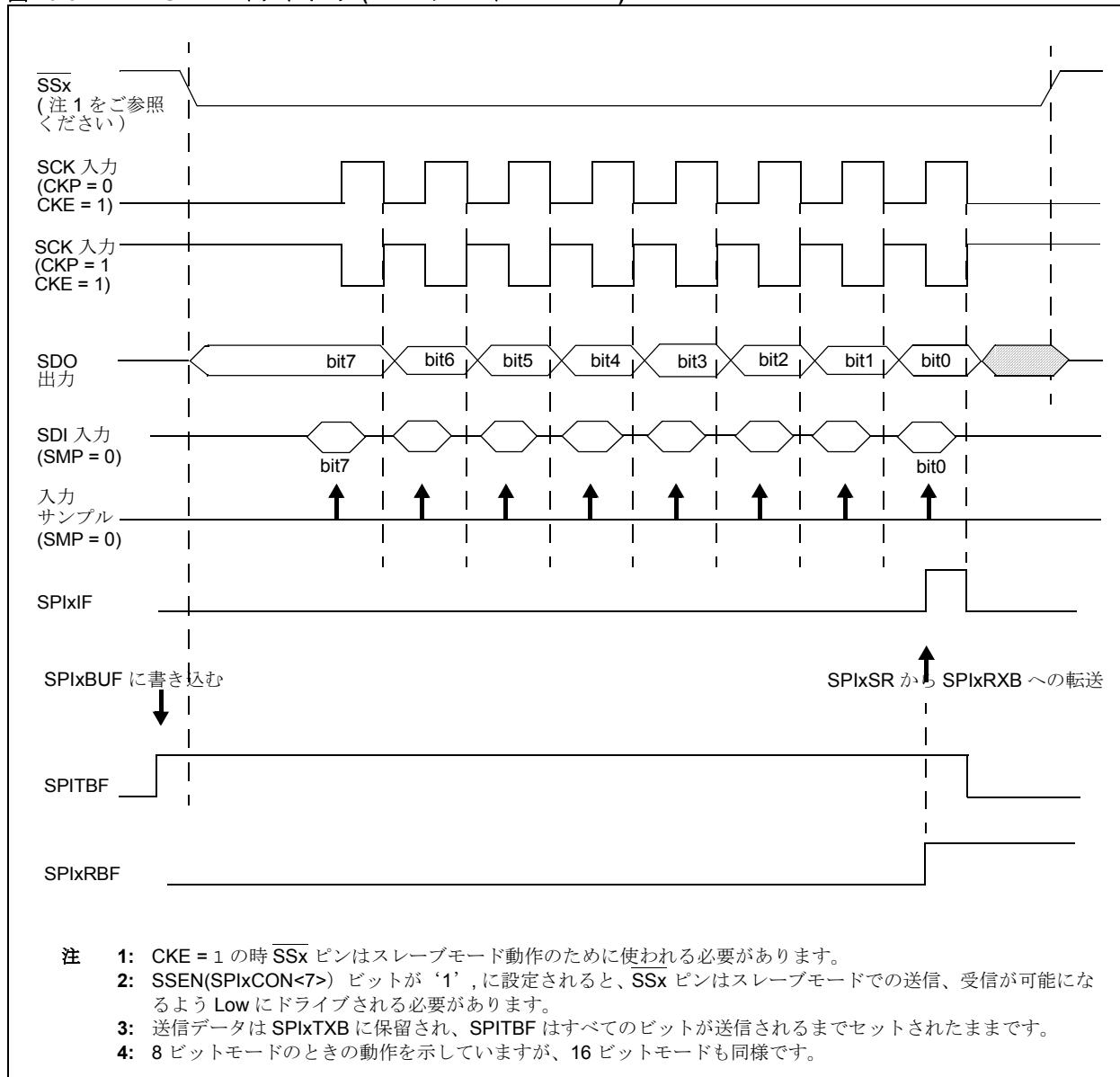


図 20-6: SPI モードタイミング (スレーブモード w/CKE = 1)



## 20.3.3 SPI エラー取扱い

新しいデータワードが SPIxSR にシフト完了したとき、SPIxRXB の以前の内容がユーザーソフトウェアによって読み込まれていなかった場合、SPIROV ビット (SPIxSTAT<6>) がセットされます。モジュールにより、受信されたデータが SPIxSR から SPIxRXB へ転送されることはありません。SPIROV ビットがクリアされるまで、さらにデータを受信することはできません。SPIROV ビットはモジュールにより自動的にクリアされず、ユーザーソフトウェアによってクリアされる必要があります。

## 20.3.4 SPI 受信のみの動作

制御ビット、DISSDO (SPIxCON<11>) をセットすることにより、SDOx ピンでの送信が無効化されます。このことにより、SPIx モジュールは受信のみのモードに構成されます。DISSDO がセットされると、SDOx ピンはそれぞれのポート関数により制御されます。

DISSDO 関数は、すべての SPI 動作モードに適用されます。

## 20.3.5 フレーム化 SPI モード

マスターまたはスレーブモードのどちらかで動作している間、非常に基本的にフレーム化された SPI プロトコルがモジュールによりサポートされています。以下の特徴は、フレーム化 SPI モードをサポートする SPI モジュールで提供されます。

- 制御ビット、FRMEN (SPIxCON<14>) により、フレーム化 SPI モードが有効化され、SSx ピンがフレーム同期化パルス入力または出力ピンとして使用される要因になります。SSEN (SPIxCON<7>) の状態は無視されます。
- 制御ビット、SPIFSD (SPIxCON<13>) により、SSx ピンが入力となるか出力となるかが決定されます（すなわち、モジュールがフレーム同期化パルスを受信するか生成するかということです）。
- フレーム同期化パルスは 1 個の SPI クロックサイクルのアクティブ High パルスです。

以下の 2 つのフレーム化 SPI モードが SPI にモジュールにサポートされています。

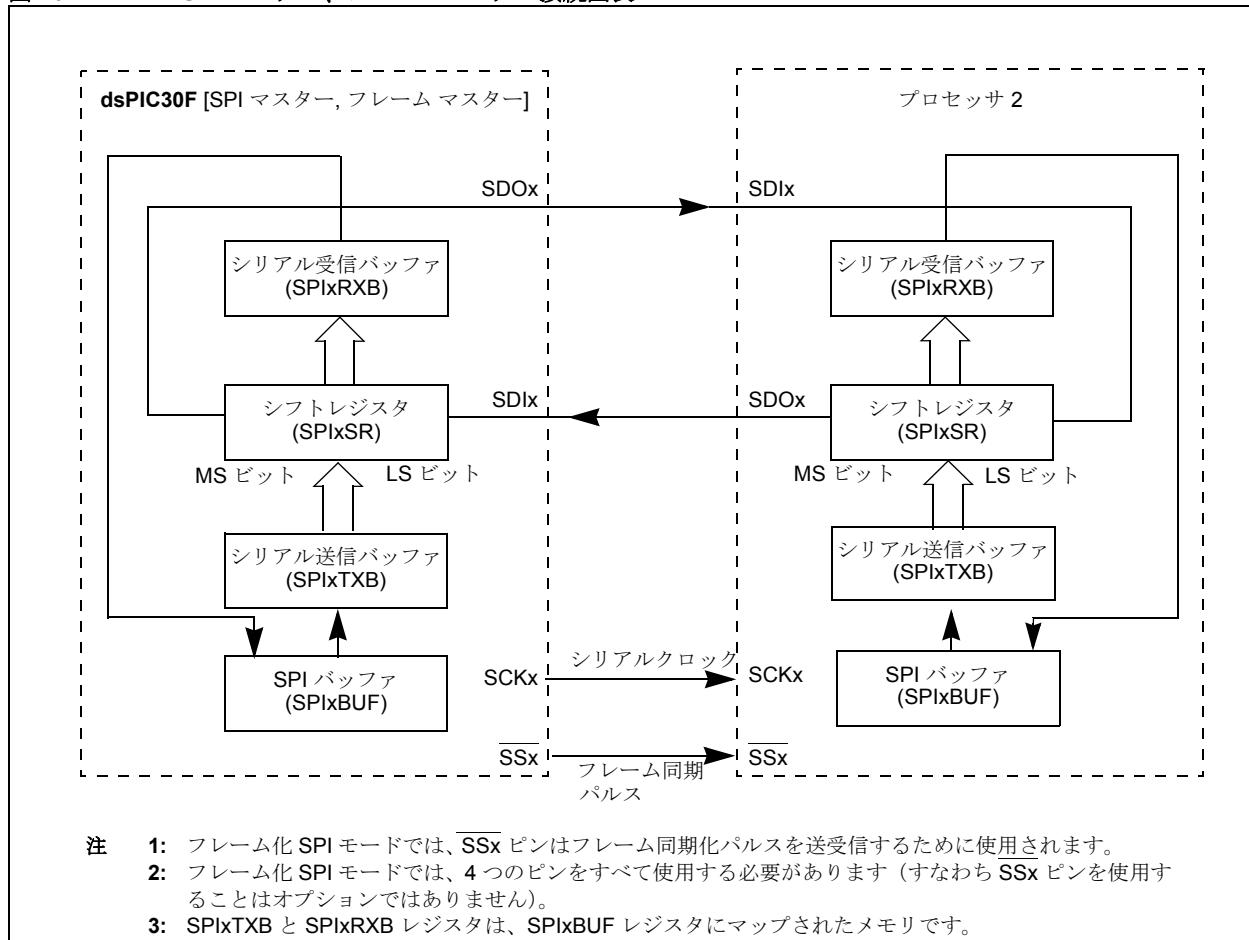
- フレームマスターモード：SPI モジュールによりフレーム同期化パルスが生成され、このパルスは、SSx ピンで他のデバイスに対し提供されます。
- フレームスレーブモード：SPI モジュールにより、SSx ピンで受信されるフレーム同期化パルスが使用されます。

フレーム化 SPI モードはマスターおよびスレーブモードと連動してサポートされています。このように、ユーザーは以下の 4 つのフレーム化構成を利用できます。

- SPI マスターモードおよびフレームマスター モード
- SPI マスターモードおよびフレームスレーブモード
- SPI スレーブモードおよびフレームマスター モード
- SPI スレーブモードおよびフレームスレーブモード

シリアルクロックとフレーム同期化パルスが SPIx モジュールにより生成されるかどうかが、この 4 つのモードにより決定されます。

図 20-7: SPI マスター、フレームマスター接続図表



### 20.3.5.1 フレーム化 SPI モードでの SCKx

FRMEN (SPIxCON<14>) = 1 および MSTEN (SPIxCON<5>) = 1 の時、SCKx ピンは出力になり、SCKx での SPI クロックはフリーランクロックになります。

FRMEN = 1 および MSTEN = 0 のとき、SCKx ピンは入力になります。SCKx ピンに提供されたソースクロックはフリーランクロックであると想定されます。

クロックの優先順位は CKP (SPIxCON<6>) ビットにより選択されます。CKE (SPIxCON<8>) ビットはフレーム化 SPI モードには使用されず、ユーザーソフトウェアによって ‘0’ にプログラムされる必要があります。

CKP = 0 の時、フレームシンクパルス出力および SDOx データ出力は、SCKx ピンのクロックパルスの立ち上がりで変化します。入力データはシリアルクロックの立下りで SDIx 入力ピンでサンプルされます。

CKP = 1 の時、フレームシンクパルス出力および SDOx データ出力は、SCKx ピンのクロックパルスの立ち下がりで変化します。入力データはシリアルクロックの立上がりで、SDIx 入力ピンでサンプルされます。

## 20.3.5.2 フレーム化 SPI モードにおける SPIx バッファ

SPIFSD (SPIxCON<13>) = 0 の時、SPIx モジュールはフレームマスター モードにあります。このモードでは、ユーザー ソフトウェアが SPIxBUF に送信データを書き込む時、フレームシンク パルスがモジュールによって開始されます（このように送信データを SPIxTXB レジスタに書き込みます）。フレーム同期パルスの終了時に、SPIxTXB は SPIxSR に転送され、データの送受信が始まります。

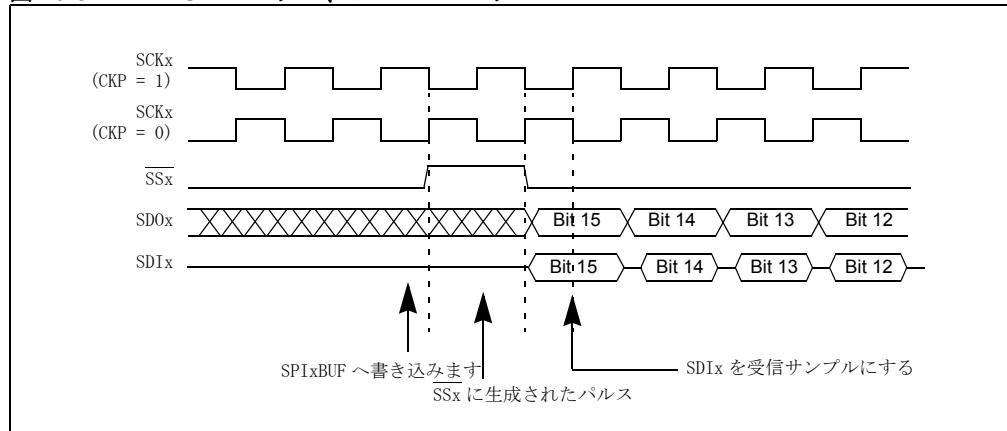
SPIFSD (SPIxCON<13>) = 1 の時、モジュールはフレームスレーブ モードにあります。このモードでは、フレーム同期パルスは外部ソースによって生成されます。モジュールによりフレーム同期パルスがサンプルされる時、SPIxTXB レジスタの内容は SPIxSR に転送され、データ送受信が始まります。ユーザーはフレーム同期パルスが受信される前に、正しいデータが送信用の SPIxBUF にロードされていることを確認する必要があります。

**注：** フレーム同期パルスの受信により、データが SPIxBUF に書き込まれたかどうかにかかわらず送信が開始されます。書き込みがまったく実行されなければ、SPIxTXB の古い内容が送信されます。

## 20.3.5.3 SPI マスター モードおよびフレームマスター モード

このフレーム化 SPI モードは、MSTEN (SPIxCON<5>) および FRMEN (SPIxCON<14>) ビット SPIFSD (SPIxCON<13>) に ‘0’ を設定することにより有効化されます。このモードでは、モジュールが送信中であるかどうかにかかわらず、シリアルクロックが SCKx ピンから継続的に出力されます。SPIxBUF に書き込まれると、SSx ピンは SCKx クロックの次の送信エッジで High にドライブされます。SSx ピンは 1 つの SCKx クロックサイクルの期間 High になります。図 20-8 に示されるように、モジュールにより SCKx の次の送信エッジでデータ送信が開始されます。この動作モードの信号方向を示している接続図表が図 20-7 に示されています。

図 20-8: SPI マスター、フレームマスター



## 20.3.5.4 SPI マスター モードおよびフレームスレーブモード

このフレーム化 SPI モードは、MSTEN, FRMEN および SPIFSD ビットを ‘1’ に設定することにより有効化されます。SSx ピンは入力で、SPI クロックのサンプルエッジでサンプルされます。High がサンプルされた時、図 20-9 に示されるように、データは SPI クロックのその後の送信エッジで送信されます。割り込みフラグ SPIxFIF は送信完了時にセットされます。ユーザーは、信号が SSx ピンで受信される前に、正しいデータが送信用の SPIxBUF にロードされていることを確認する必要があります。この動作モードのための信号方向を示す接続図表は図 20-10 で示されます。

図 20-9: SPI マスター、フレームスレーブ

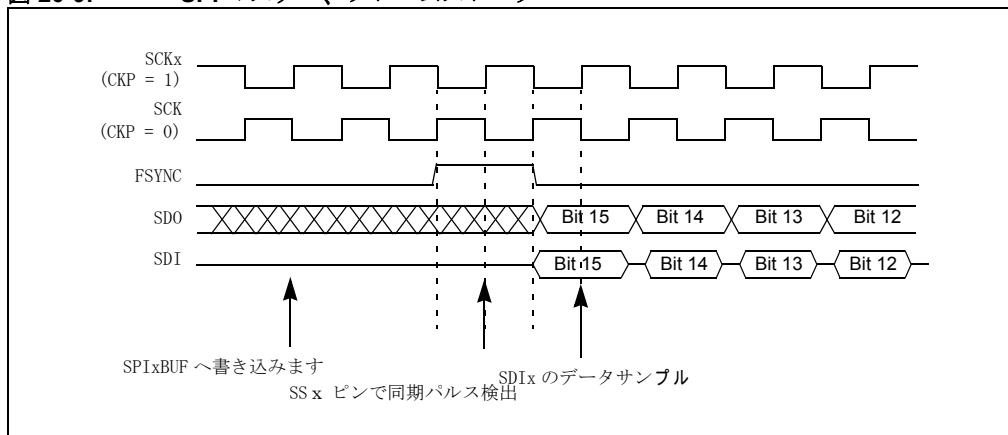
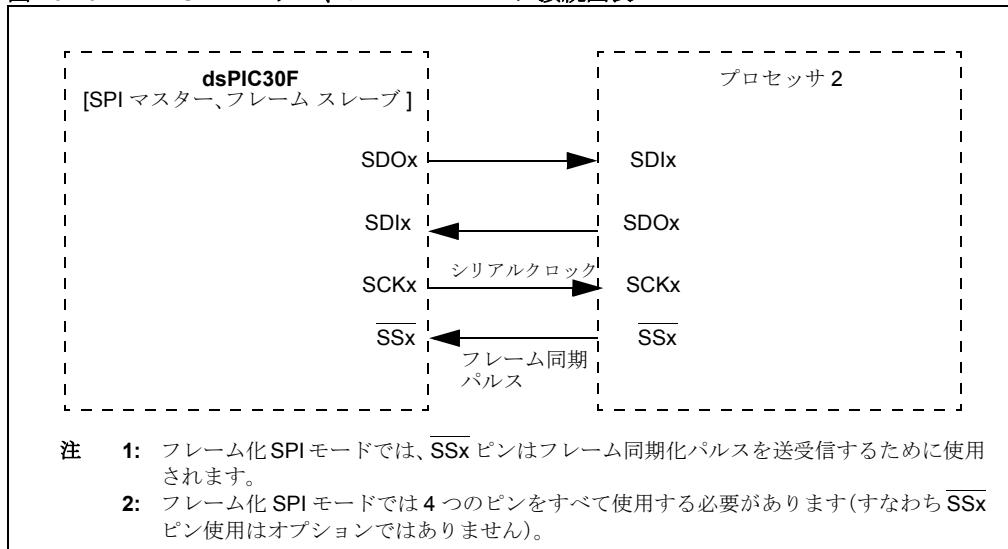


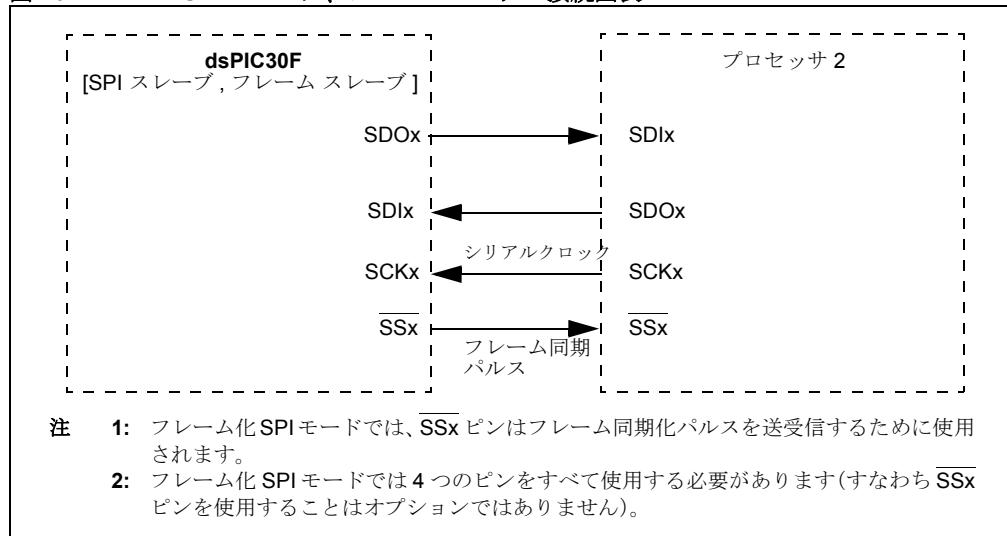
図 20-10: SPI マスター、フレームスレーブ接続図表



## 20.3.5.5 SPI スレーブモードおよびフレームマスター モード

このフレーム化 SPI モードは ‘0’ を MSEN (SPI<sub>x</sub>CON<5>) へ、‘1’ を FRMEN (SPI<sub>x</sub>CON<14>) ビットへ、および ‘0’ を SPIFSD (SPI<sub>x</sub>CON<13>) ビットへ設定することにより有効化されます。入力 SPI クロックはスレーブモードから連続的に出力されます。SSx ピンは SPIFSD ビットが Low の時、出力になります。SPIBUF が書き込まれたあと、モジュールにより SPI クロックの次の送信エッジで SSx ピンが High にドライブされます。SSx ピンは 1 つの SPI クロックサイクル期間 High にドライブされます。データは次の SPI クロック送信エッジで送信開始されます。この動作モードのための信号方向を示す接続図表は 図 20-11 で示されます。

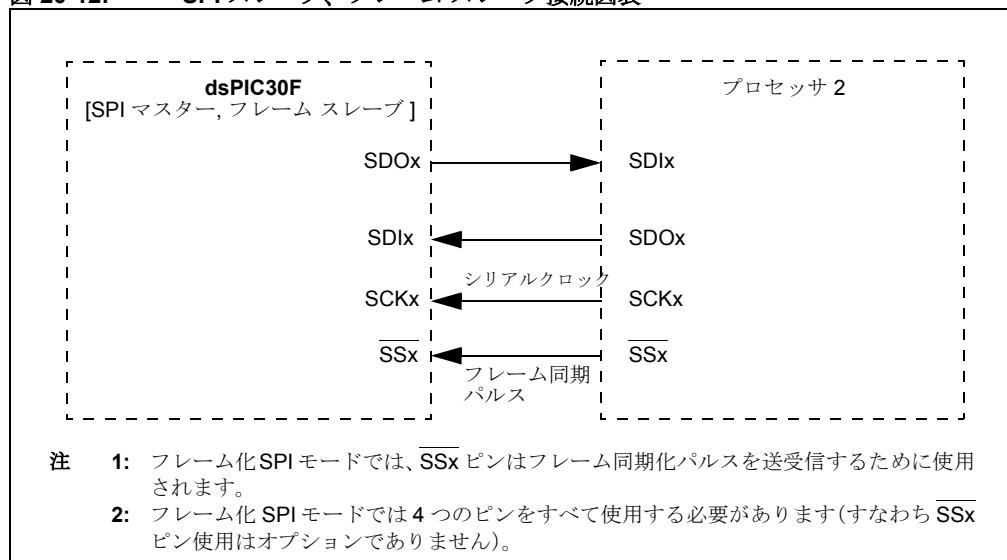
図 20-11: SPI スレーブ、フレーム マスター接続図表



## 20.3.5.6 SPI スレーブモードおよびフレームスレーブモード

このフレーム化 SPI モードは ‘0’ を MSEN (SPI<sub>x</sub>CON<5>) ビットへ、‘1’ を FRMEN ビット (SPI<sub>x</sub>CON<14>) へ、および ‘1’ を SPIFSD (SPI<sub>x</sub>CON<13>) ビットへ設定することにより有効になります。この場合、SCKx と SSx ピンは両方とも入力になります。SSx ピンは SPI クロックのサンプルエッジでサンプルされます。SSx の High がサンプルされると、データは SCKx の次の送信エッジで送信されます。この動作モードのための信号方向を示す接続図表は 図 20-12 で示されます。

図 20-12: SPI スレーブ、フレーム スレーブ接続図表



## 20.4 SPI マスター モード クロック周波数

マスター モードでは、SPI に供給されるクロックは命令サイクル ( $T_{CY}$ ) です。次にこのクロックは、( $PPRE<1:0>$ ( $SPIxCON<1:0>$ ) に設定された) 第 1 プリスケーラおよび ( $SPRE<2:0>$ ( $SPIxCON<4:2>$ ) に設定された) 第 2 プリスケーラによって分周されます。分周された命令クロックはシリアルクロックになり、 $SCKx$  ピンから外部デバイスに供給されます。

**注:**  $SCKx$  信号クロックは、ノーマル SPI モードのときはフリーランではないことにご注意ください。このクロックは  $SPIxBUF$  にデータがにロードされている時に、8 または 16 パルスのみで出力されます。ただし、フレーム化モードのときは連続です。

式 20-1 は第 1、第 2 プリスケーラ設定の関数として、 $SCKx$  クロック周波数を計算するために使用できます。

式 20-1:

$$F_{SCK} = \frac{F_{CY}}{\text{第 1 プリスケーラ} * \text{第 2 プリスケーラ}}$$

サンプル SPI クロック周波数 (kHz で) のいくつかの例を以下の表に示します。

表 20-1: サンプル  $SCKx$  周波数

$F_{CY} = 30 \text{ MHz}$		第 2 プリスケーラ設定				
		1:1	2:1	4:1	6:1	8:1
第 1 プリスケーラ設定	1:1	30000	15000	7500	5000	3750
	4:1	7500	3750	1875	1250	938
	16:1	1875	938	469	313	234
	64:1	469	234	117	78	59
$F_{CY} = 5 \text{ MHz}$						
第 1 プリスケーラ設定	1:1	5000	2500	1250	833	625
	4:1	1250	625	313	208	156
	16:1	313	156	78	52	39
	64:1	78	39	20	13	10

注:  $SCKx$  周波数は kHz で示されています。

**注:** すべてのクロックレートがサポートされるわけではありません。詳細は、特定デバイスデータシートの SPI タイミング仕様書をご参照ください。

## 20.5 省電力モードでの動作

デバイスの dsPIC30FXXXX ファミリーには 3 つの省電力モードがあります。

- 通常動作モード：コアおよび周辺装置は実行中です。
- 省電力モード：PWRSAV 命令の実行によって呼び出されます。dsPIC30F ファミリーデバイスでサポートされている省電力モードは 2 つあります。PWRSAV 命令のパラメータで特定されています。この 2 つのモードは以下です。
  - SLEEP モード：デバイスクロックソースおよび全デバイスがシャットダウンされます。これは以下の命令により達成されます。

```
;include device p30fxxxx.inc file  
PWRSAV #SLEEP_MODE
```

- IDLE モード：デバイスクロックは動作可能であり、CPU および選択された周辺装置はシャットダウンされます。

```
;include device p30fxxxx.inc file  
PWRSAV #IDLE_MODE
```

### 20.5.1 SLEEP モード

デバイスが SLEEP モードに入ると、システムクロックが無効化されます。

#### 20.5.1.1 マスター モード動作

以下は SPIx がマスター動作のために構成されている時に SLEEP モードに入ったときです。

- SPIx モジュールのボーレートジェネレータは停止し、リセットされます。
- SPIx モジュールが送受信の途中で SLEEP モードに入ると、その送受信は中断されます。送信や受信の途中で、SLEEP モードに入るのを防ぐための自動的な方法がないので、ユーザーソフトウェアにより送信が中断されるのを避けるために、SPI モジュールの動作に SLEEP へのエントリーが同期化される必要があります。
- トランスマッタおよびレシーバは SLEEP では停止します。トランスマッタまたはレシーバは、ウェイクアップ時に部分的に完了した送信は継続しません。

#### 20.5.1.2 スレーブモード動作

SCKx でのクロックパルスがスレーブモードのために外部から供給されるので、このモジュールは SLEEP モードでも機能を継続します。SLEEP への移行の間いざれの転送も完了されます。転送が完了すると、SPIRBF フラグがセットされます。その結果、SPIxIF ビットがセットされます。SPI 割り込みが有効化され (SPIxE = 1) ていれば、そのデバイスは SLEEP からウェイクアップします。SPI 割り込み優先レベルが現在の CPU 優先レベルより大きければ、コード実行が SPIx 割り込みベクトル位置で再起動されます。さもなければ、最初に SLEEP モードにした PWRSAV 命令の次の命令からコード実行が継続されます。モジュールは、スレーブデバイスとして動作中の場合、SLEEP モードに入るとリセットされません。

SPIx モジュールが SLEEP モードに出入りする場合、レジスタ内容は影響を受けません。

#### 20.5.2 IDLE モード

デバイスが IDLE モードに入ると、システムクロックソースは機能を続けます。SPISIDL ビット (SPIxSTAT<13>) により、モジュールは停止するか IDLE で機能を続けるかが選択されます。

- SPISIDL = 1 ならば、SPI モジュールは IDLE モードに入ったら動作は停止されます。SLEEP モードのときと同じ動作になります
- SPISIDL = 0 (デフォルト選択) ならば、モジュールにより IDLE モードで動作が継続されます。

表 20-2: Pins Associated with the SPI Modules

ピン名	ピン タイプ	バッファ タイプ	説明
SCK1	I/O	CMOS	SPI1 モジュールクロック入力または出力
SCK2	I/O	CMOS	SPI2 モジュールクロック入力または出力
SDI1	I	CMOS	SPI1 モジュールデータ受信ピン
SDI2	I	CMOS	SPI2 モジュールデータ受信ピン
SDO1	O	CMOS	SPI1 モジュールデータ送信ピン
SDO2	O	CMOS	SPI2 モジュールデータ送信ピン
SS1	I/O	CMOS	SPI1 モジュールスレーブ選択制御ピン 1) SSEN (SPI1CON<7>) が ‘1’ に設定されると、スレーブモードで送信・受信を有効化するように使用されます。 2) FRMEN および SPIFSD (SPI1CON<14:13>) が ‘11’ または ‘10’ に設定されると、フレーム同期入出力パルスとして使用されます。
SS2	I/O	CMOS	SPI2 モジュールスレーブ選択制御ピン 1) SSEN (SPI2CON<7>) が ‘1’ に設定されるスレーブモードで送信・受信を有効化するように使用されます。 2) FRMEN および SPIFSD (SPI2CON<14:13>) が ‘11’ または ‘10’ に設定されると、フレーム同期入出力パルスとして使用されます。

凡例： CMOS = CMOS 互換入力または出力、 ST = CMOS レベルでのシミュットトリガーアクション、 I = 入力、 O = 出力

## 20.6 SPI モジュールに関連の特殊機能レジスタ

表 20-3: SPI1 レジスタマップ

SFR 名称	アドレス	ピット 15	ピット 14	ピット 13	ピット 12	ピット 11	ピット 10	ピット 9	ピット 8	ピット 7	ピット 6	ピット 5	ピット 4	ピット 3	ピット 2	ピット 1	ピット 0	リセット状態
SPI1STAT	0220	SPIEN	—	SPISIDL	—	—	—	—	—	—	SPIROV	—	—	—	—	SPITBFI	SPIRBF	0000 0000 0000 0000
SPI1CON	0222	—	FRMEN	SPIFSD	—	DISSDO	MODE16	SMP	CKE	SSEN	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0	0000 0000 0000 0000
SPI1BUF	0224	SPI1TXB および SPI1RXB レジスタが共有しているバッファアドレスを送信し、受信してください														0000 0000 0000 0000		

表 20-4: SPI2 レジスタマップ

SFR 名称	アドレス	ピット 15	ピット 14	ピット 13	ピット 12	ピット 11	ピット 10	ピット 9	ピット 8	ピット 7	ピット 6	ピット 5	ピット 4	ピット 3	ピット 2	ピット 1	ピット 0	リセット状態
SPI2STAT	0226	SPIEN	—	SPISIDL	—	—	—	—	—	—	SPIROV	—	—	—	—	SPITBFI	SPIRBF	0000 0000 0000 0000
SPI2CON	0228	—	FRMEN	SPIFSD	—	DISSDO	MODE16	SMP	CKE	SSEN	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0	0000 0000 0000 0000
SPI2BUF	022A	SPI2TXB および SPI2RXB レジスタが共有しているバッファアドレスを送信し、受信してください														0000 0000 0000 0000		

表 20-5: SPI モジュール関係割込みレジスタ

SFR 名称	アドレス	ピット 15	ピット 14	ピット 13	ピット 12	ピット 11	ピット 10	ピット 9	ピット 8	ピット 7	ピット 6	ピット 5	ピット 4	ピット 3	ピット 2	ピット 1	ピット 0	リセット状態
INTCON1	0080	NSTDIS	—	—	—	OVATE	OVBTE	COVTE	—	—	—	SWTRAP	OVRFLOW	ADDRERR	STKERR	—	0000 0000 0000 0000	
INTCON2	0082	ALТИVT	DISI	—	—	—	LEV8F	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INTOEP	0000 0000 0000 0000		
IFS0	0084	CNIF	BCLIF	I2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INTO	0000 0000 0000 0000
IFS1	0086	IC6IF	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IEC0	008C	CNIE	BCLIE	I2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INTOIE	0000 0000 0000 0000
IEC1	008E	IC6IE	IC5IE	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IPC2	0098	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>			0100 0100 0100 0100
IPC6	00A0	—	C1IP<2:0>			—	SPI2IP<2:0>			—	U2TXIP<2:0>			—	U2RXIP<2:0>			0100 0100 0100 0100

## 20.7 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップしています。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありません。ただし、その概念は適切であり、修正や可能な制限をして使用できる可能性もあります。シリアル周辺装置インターフェース (SPI) モジュールに関連する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 #
Microchip 社製 MCP41XXX/MCP42XXX デジタルポテンションメーターの PICmicro® マイクロコントローラへのインターフェース	AN746
Microchip 社製 MCP3201 アナログデジタル変換器の PICmicro® マイクロコントローラへのインターフェース	AN719

**注：** デバイスの dsPIC30F ファミリーに関しての、追加のアプリケーションノートやコード例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。.

## 20.8 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

この改訂は dsPIC30F シリアル周辺装置インターフェース (SPI) モジュールの編集上および技術上の内容変更を反映しています。



MICROCHIP

21

I<sup>2</sup>C

## 第 21 章 . I<sup>2</sup>C<sup>TM</sup>

### ハイライト

この章には以下の主要な項目が含まれます。

21.1 概要 .....	21-2
21.2 I <sup>2</sup> C バスの特長 .....	21-4
21.3 ステータスおよびコントロールレジスタ .....	21-7
21.4 I <sup>2</sup> C 動作の有効化 .....	21-13
21.5 シングルマスター環境でマスターとして通信 .....	21-15
21.6 マルチマスター環境でマスターとして通信 .....	21-29
21.7 スレーブとして通信 .....	21-32
21.8 I <sup>2</sup> C バス接続についての検討事項 .....	21-47
21.9 PWRSAV 命令実行中のモジュール動作 .....	21-49
21.10 RESET の影響 .....	21-49
21.11 設計の秘訣 .....	21-50
21.12 関連するアプリケーションノート .....	21-51
21.13 改訂履歴 .....	21-52

## 21.1 概要

I<sup>2</sup>C モジュールは、他の周辺装置またはマイクロコントローラデバイスと通信する際に役立つシリアルインターフェイスです。ここでいう周辺装置とは、シリアル EEPROM、ディスプレイドライバ、A/D コンバータなどです。

I<sup>2</sup>C モジュールは以下の I<sup>2</sup>C システムのいずれかで動作します。

- dsPIC30F がスレーブデバイスとして動作するシステム
- dsPIC30F がシングルマスターシステム内でマスターデバイスとして動作する場合  
(スレーブとして動作する場合あり)
- dsPIC30F がマルチマスターシステム内でマスター/スレーブデバイスとして動作する場合  
(バス衝突の検出およびアビトレーション可能)

I<sup>2</sup>C モジュールは独立した I<sup>2</sup>C マスターロジックと I<sup>2</sup>C スレーブロジックを持ち、各ロジックがそのイベントに基づき割り込みを生成します。マルチマスターシステムでは、ソフトウェアによってマスターとスレーブに単純に分けられます。

I<sup>2</sup>C マスターロジックがアクティブな時、スレーブロジックもアクティブのままとなり、バスの状態を検出したり、シングルマスターシステム内で自らに宛てたメッセージやマルチマスターシステム内のその他マスターからのメッセージを常時受信します。マルチマスターはアビトレーションがアクティブになっている間はメッセージが失われることはありません。

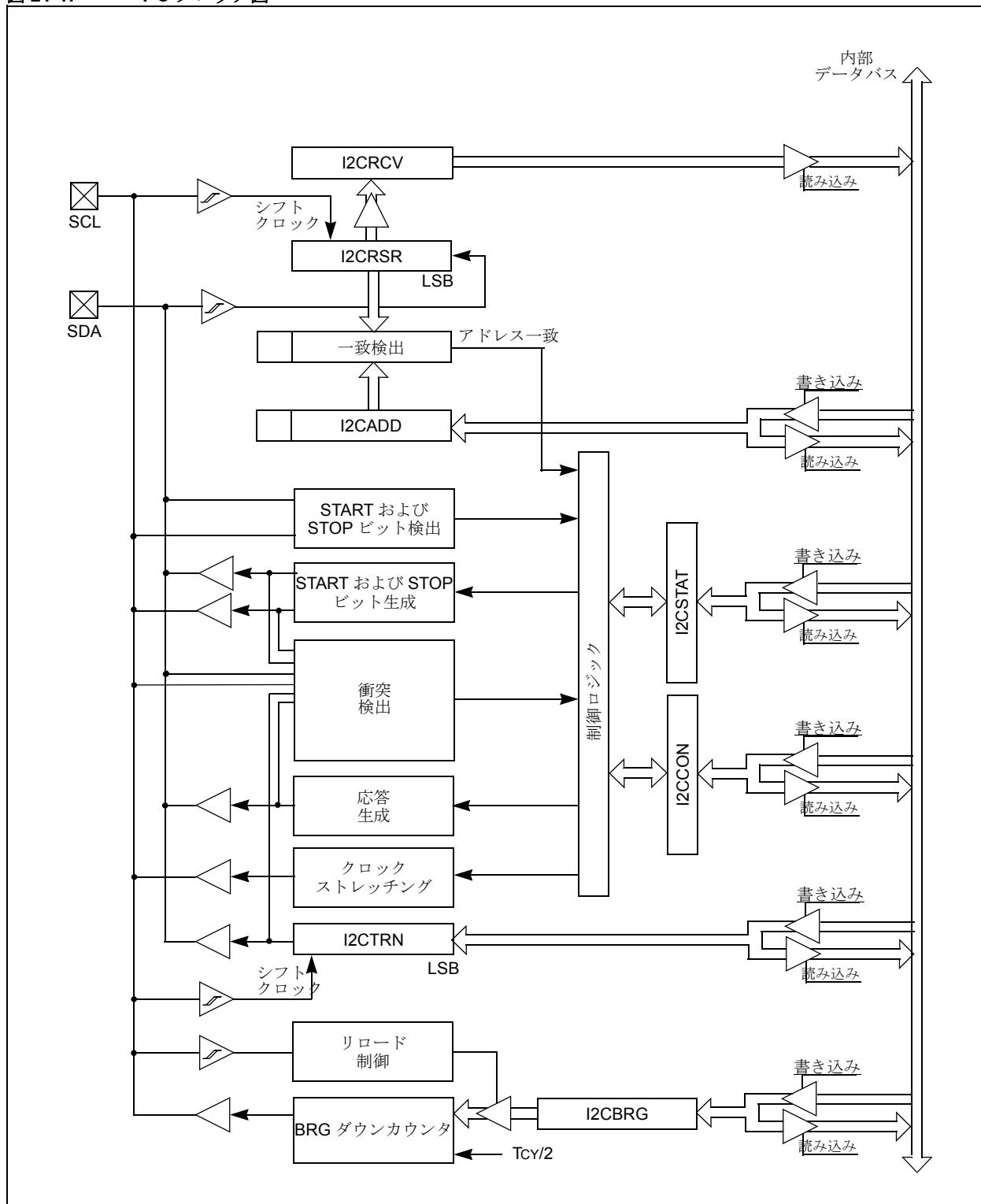
マルチマスターシステムでは、システム内での他のマスターとのバス衝突が検出されたとき、モジュールによってメッセージの終了と再起動の手段が提供されます。

I<sup>2</sup>C モジュールにはボーレート生成器が含まれます。I<sup>2</sup>C ボーレート生成器はデバイス内のほかのタイマーのリソースを消費しません。

### 21.1.1 モジュールの機能

- 独立したマスターおよびスレーブロジック
- マルチマスターが構成できアビトレーション中にメッセージが失なわれない
- 7 ビットおよび 10 ビットデバイスアドレス検出
- I<sup>2</sup>C プロトコルで定義された一斉呼び出しアドレス検出
- バスリピーター モードのときは、スレーブアドレスに関係なくすべてのメッセージを受信する
- 自動 SCL クロックストレッ칭がスレーブからの要求に応じて遅延を生成する
- 100 kHz および 400 kHz バス仕様をサポート

図 21-1 は I<sup>2</sup>C モジュールのブロック図を示しています。

図 21-1: I<sup>2</sup>C ブロック図

## 21.2 I<sup>2</sup>C バスの特長

I<sup>2</sup>C バスは 2 ワイヤシリアルインターフェースです。図 21-2 は dsPIC30F と 24LC256 I<sup>2</sup>C シリアル EEPROM 間の典型的な I<sup>2</sup>C 接続の回路図です。

I<sup>2</sup>C インターフェースは包括的なプロトコルを採用することによって、信頼性の高いデータの送受信を可能にしています。通信時には、1 つのデバイスがバスへの転送を開始する“マスター”となり、その転送を許可するクロック信号を生成し、他方のデバイス（複数の場合あり）が“スレーブ”としてその転送に応答します。クロック線“SCL”はマスターからスレーブへの出力で、場合によってはスレーブが SCL 線をドライブすることもあります。データ線“SDA”はマスターとスレーブの両方からの出力および入力になることがあります。

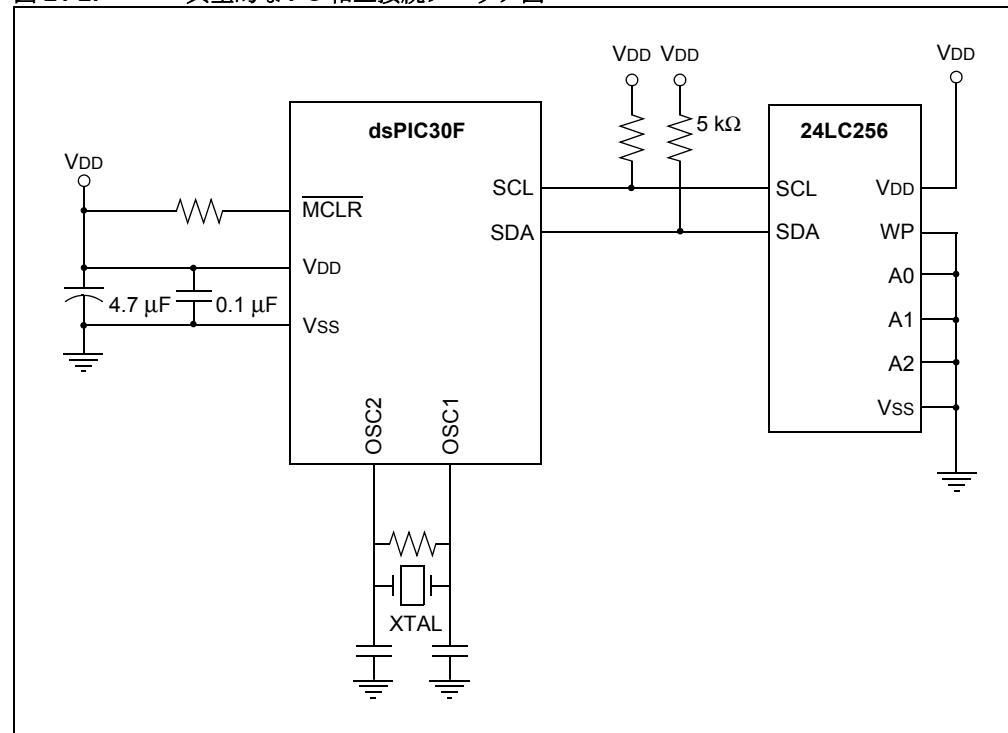
SDA と SCL は双方向性があるため、SDA 線と SCL 線をドライブするデバイスの出力ステージは wired-AND 機能が実現できるようオープンドレイン構成とする必要があります。ラインに出力するデバイスがない場合は、外部プルアップ抵抗によって High レベルが維持されます。

I<sup>2</sup>C インターフェースプロトコルでは、各デバイスが 1 つのアドレスを持っています。マスターがデータ転送を試みる場合、最初に、“通信”したいデバイスのアドレスを送信します。すべてのデバイスがこれを“受信”し、それが自分のアドレスかどうかを確認します。このアドレス内では、ビット‘0’はマスターがスレーブデバイスをに対し、送信、受信のどちらを行おうとしているかを指定します。データ転送中、マスターとスレーブは常にオペレーションの逆のモード（送信側 / 受信側）にあります。そのため、マスターとスレーブの動作は以下の関係のいずれかになると考えられます。

- マスター - 送信機、スレーブ - 受信機
- スレーブ - 送信機、マスター - 受信機

どちらの場合でも、マスターが SCL クロック信号を生成します。

図 21-2: 典型的な I<sup>2</sup>C 相互接続ブロック図



## 21.2.1 バスプロトコル

以下の I<sup>2</sup>C バスプロトコルが定義されています。

- バスがビジー状態でない場合のみデータ転送を開始
- データ転送中、SCL クロック線が HIGH の場合は常にデータ線は安定している必要があります。SCL 線が HIGH の間に発生したデータ線変更は START または STOP 条件として割り込みになる

上記により、以下のバス条件が定義されます(図 21-3)。

### 21.2.1.1 START データ転送 (S)

バス IDLE 状態の後、クロック (SCL) が HIGH になっている間の SDA 線の HIGH から LOW への遷移は START 条件と判断されます。すべてのデータ転送は START 条件で開始する必要があります。

### 21.2.1.2 STOP データ転送 (P)

クロック (SCL) が HIGH になっている間の SDA 線の LOW から HIGH への遷移は STOP 条件と判断されます。すべてのデータ転送は STOP 条件で終了する必要があります。

### 21.2.1.3 繰り返し START (R)

WAIT 状態の後、クロック (SCL) が HIGH になっている間の SDA 線の HIGH から LOW への遷移は繰り返し START 条件と判断されます。繰り返し START 条件により、マスターはバス制御を放棄せずにバスの方向を変更できます。

### 21.2.1.4 有効データ (D)

START 条件の後に SDA 線の状態が有効データを示した場合、SDA 線はクロック信号が HIGH の状態が続いている間は安定しています。1つの SCL クロックにつき 1 ビットのデータがあります。

### 21.2.1.5 応答 (A) または無視 (N)

すべてのデータバイト送信に対し、受信機は応答 (ACK) または無視 (NACK) を返す必要があります。受信機は ACK の場合は SDA 線を Low にし、NACK の場合は SDA 線を High にします。応答は 1 つの SCL クロックを使用した 1 ビット期間の処理です。

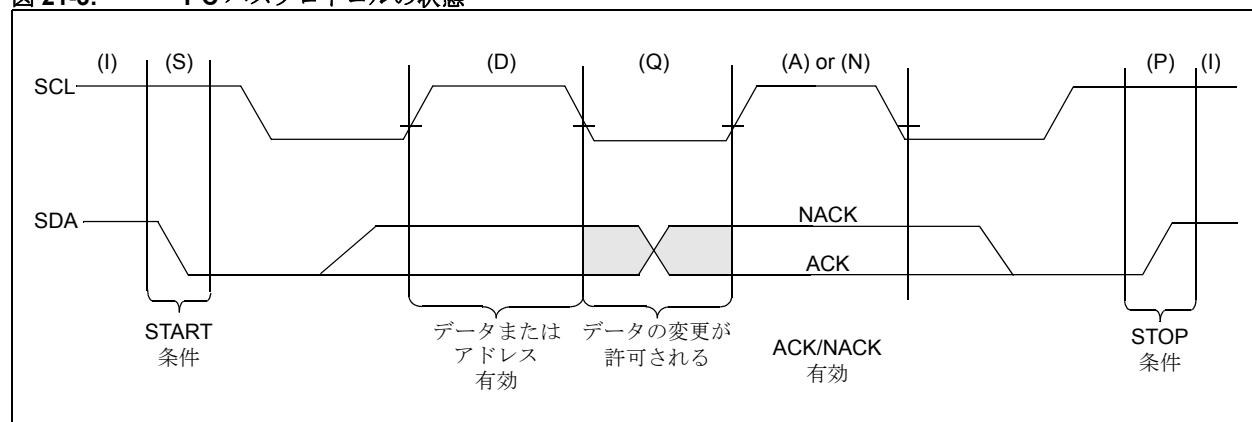
### 21.2.1.6 WAIT/データ無効 (Q)

線路上のデータはクロック信号が LOW の間に変更される必要があります。デバイスは SCL 線を LOW にドライブすることによりバス上に WAIT 状態を生じさせて SCL の LOW の時間を延長できます。

### 21.2.1.7 バス IDLE (I)

STOP 条件の後で次の START 条件が発生するまでの間、データ線とクロック線の両方が HIGH のままになります。

図 21-3: I<sup>2</sup>C バスプロトコルの状態

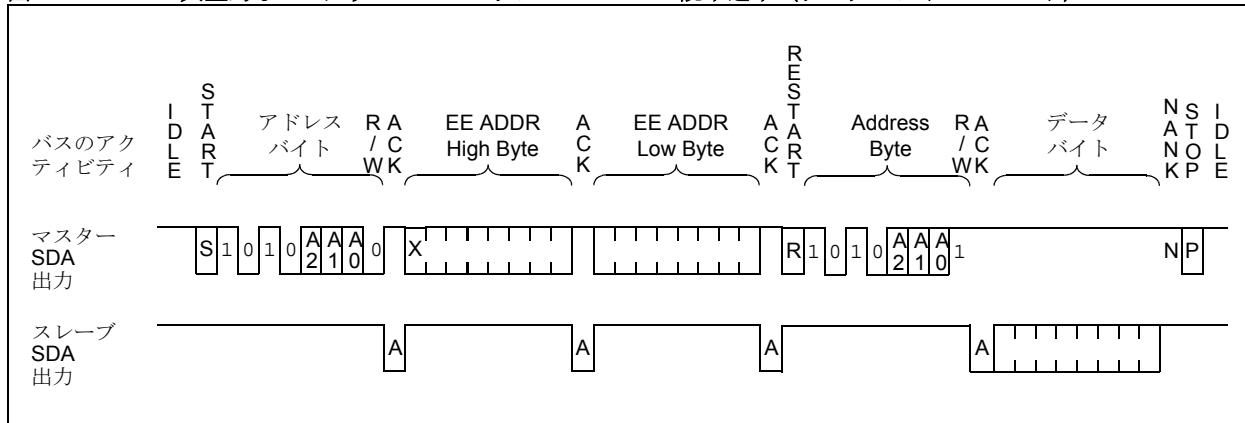


## 21.2.2 メッセージプロトコル

典型的な I<sup>2</sup>C メッセージは図 21-4 に示すとおりです。この例では、メッセージが 24LC256 I<sup>2</sup>C シリアル EEPROM から指定のバイトで読み込まれます。dsPIC30F デバイスはマスターとして動作し、24LC256 デバイスはスレーブとして動作します。

図 21-4 では、マスターデバイスにドライブされたデータとスレーブデバイスにドライブされたデータが示され、複合 SDA 線がマスターおよびスレーブデータの wired AND であることがわかります。マスターデバイスはプロトコルシーケンスを制御します。スレーブデバイスは指定された時間だけバスをドライブします。

図 21-4: 典型的な I<sup>2</sup>C メッセージ：シリアル EEPROM 読み込み（ランダムアドレスモード）



### 21.2.2.1 START メッセージ

各メッセージは “START” 条件により開始され、“STOP” 条件により終了します。START 条件から STOP 条件までの間に転送されるデータの各バイトはマスターデバイスによって決定されます。システムプロトコルに定義されたメッセージの各バイトは“デバイスアドレスバイト”、“データバイト”などの特定の意味を持ちます。

### 21.2.2.2 アドレススレーブ

図では、1 番目のバイトはデバイスアドレスバイトで、I<sup>2</sup>C メッセージの最初の部分になるべきものです。デバイスアドレスと R/W ビットを含みます。アドレスバイトのフォーマットについて詳しくは、第 26 章「付録」を参照してください。1 番目のアドレスバイトが R/W = 0 の場合、マスターが送信機、スレーブが受信機となることを意味します。

### 21.2.2.3 スレーブ応答

受信側デバイスは各バイトの受け入れ後、応答信号 “ACK” を生成する必要があります。マスターデバイスは応答ビットに対応する追加の SCL クロックを生成します。

### 21.2.2.4 マスター送信

次の 2 バイトがマスターからスレーブに送られます。これは要求された EEPROM データバイトのアドレス情報を含むデータバイトです。スレーブは各データバイトに応答する必要があります。

### 21.2.2.5 繰り返し START

ここまで、スレーブ EEPROM は要求されたデータバイトをマスターに返すのに必要なアドレス情報を得ました。ただし、1 番目のデバイスアドレスバイトの R/W ビットによってマスター送信とスレーブ受信と指定されています。このため、スレーブがマスターにデータを送信できるようにするには、バスを反対の方向に向ける必要があります。

メッセージを終了させることなくこれを行うため、マスターは “Repeated START” を送信します。Repeated START の後に、以前と同じデバイスアドレスを含むデバイスアドレスバイトが続きます。スレーブが送信し、マスターが受信するようにするために、R/W = 1 が付与されています。

### 21.2.2.6 スレーブ返信

スレーブが SDA 線をドライブしてデータバイトを送信します。マスターはクロックを生成する一方で自らの SDA ドライブを解放します。

### 21.2.2.7 マスター応答

読み込んだあと、マスターはメッセージの最後のバイト読み込みが完了したことを通知するため、NACK を返送する必要があります。

### 21.2.2.8 STOP メッセージ

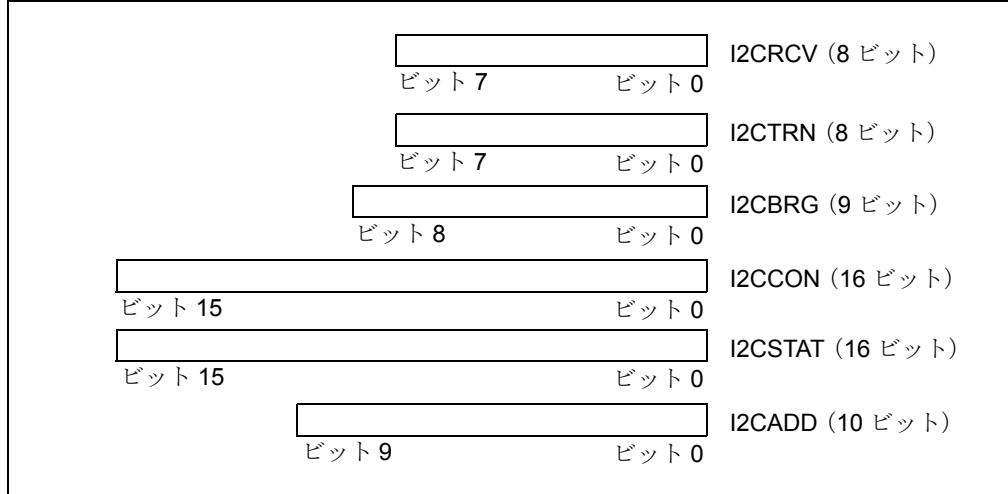
マスターはメッセージを終了するために STOP を送信し、バスを IDLE 状態に戻します。

## 21.3 ステータスおよびコントロールレジスタ

I<sup>2</sup>C モジュールは、ユーザーが利用可能な 6 つの I<sup>2</sup>C オペレーションのためのレジスタを持っています。レジスタはバイトモード、ワードモードのどちらでも使用可能です。図 21-5 で示されるレジスタは以下の通りです。

- 制御レジスタ (I2CCON) : このレジスタは I<sup>2</sup>C オペレーションの制御を可能にします。
- ステータスレジスタ (I2CSTAT) : このレジスタは I<sup>2</sup>C オペレーション中のモジュールの状態を示すステータスフラグを含みます。
- 受信バッファレジスタ (I2CRCV) : これは読み込み可能なデータバイトのバッファレジスタです。I2CRCV レジスタは読み取り専用レジスタです。
- 送信レジスタ (I2CTRN) : これは送信レジスタで、送信オペレーション中にバイトはこのレジスタに書き込まれます。I2CTRN レジスタは読み取り / 書き込みレジスタです。
- アドレスレジスタ (I2CADD) : I2CADD レジスタはスレーブデバイスアドレスを保持します。
- ボーレート生成用リロードレジスタ (I2CBRG) : ボーレート生成用リロード値を I<sup>2</sup>C モジュールボーレート生成器のために保持します。

図 21-5: I<sup>2</sup>C プログラマーモード I



レジスタ 21-1 と レジスタ 21-2 は、I<sup>2</sup>C モジュール制御とステータスレジスタである I2CCON と I2CSTAT を定義します。

I2CTRN は送信データが書き込まれるレジスタです。このレジスタはモジュールがスレーブヘーデータを送信するマスターとして、またはマスターへ返信データを送信するスレーブとして動作する場合に使用されます。メッセージの送信中は、I2CTRN レジスタの個々のビットがシフトします。このため、I2CTRN にはバスが IDLE 状態にある時以外は書き込みを行なうべきではありません。I2CTRN は現在のデータが送信されている間でもリロードできます。

マスターまたはスレーブのいずれの場合も受信されたデータは I2CRSR と呼ばれるアクセス不能のシフトレジスタにシフトされます。すべてのビットが受信されると、バイトは I2CRCV レジスタに移行します。受信オペレーションでは、I2CRSR および I2CRCV はダブルバッファを構成します。これにより、受信された現在のバイトを読み込む前に次のバイトの受信を開始できます。

ソフトウェアが I2CRCV レジスタから受信したバイトを読み込む前にモジュールが新たにバイトを受信完了してしまった場合、受信機はオーバーフローを起こし I2C0V (I2CCON<6>) をセットします。I2CRSR 内のバイトは失われます。

I2CADD レジスタはスレーブデバイスアドレスを保持します。10 ビットモードでは、すべてのビットが適合です。7 ビットアドレスモードでは、I2CADD<6:0> のみが適合します。A1OM (I2CCON<10>) はスレーブアドレスがどちらのモードかを指定します。

レジスタ 21-1: I2CCON: I<sup>2</sup>C 制御レジスタ

上位バイト :							
R/W-0	U-0	R/W-0	R/W-1 HC	R/W-0	R/W-0	R/W-0	R/W-0
I2CEN	—	I2CSIDL	SCLREL	IPMIEN	A10M	DISSLW	SMEN
ÉrÉbÉg15	ÉrÉbÉg8						

下位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0 HC	R/W-0 HC	R/W-0 HC	R/W-0 HC	R/W-0 HC
GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
ビット 7	ビット 0						

ビット 15 **I2CEN:** I<sup>2</sup>C 有効ビット

1 = I<sup>2</sup>C モジュールを有効化し、SDA ピンおよび SCL ピンをシリアルポートピンとして設定します。  
0 = I<sup>2</sup>C モジュールを無効化します。すべての I<sup>2</sup>C ピンはポート関数で制御されます。

## ビット 14 未実装: ‘0’ が読み込まれます

14

ビット 13 **I2CSIDL:** IDLE モードでの停止ビット

1 = デバイスが IDLE モードに入った時モジュール動作停止。  
0 = IDLE モードでもモジュール動作継続。

ビット 12 **SCLREL:** SCL 解放制御ビット (I<sup>2</sup>C スレーブとして動作時)

1 = SCL クロックを解放  
0 = SCL クロックを Low でホールド (クロック延長)

## STREN = 1 の場合 :

ビットは R/W です (ソフトウェアは延長を開始するために ‘0’ を書き込み、クロックの解放のために ‘1’ を書き込みます)

スレーブ送信開始時にはハードウェアでクリアにされます。

スレーブ受信終了時にもハードウェアでクリアにされます。

## STREN = 0 の場合 :

ビットは R/S です (ソフトウェアはクロックの解放のために ‘1’ のみを書き込みます)

スレーブ送信開始時にはハードウェアでクリアされます。

ビット 11 **IPMIEN:** Intelligent Peripheral Management Interface (IPMI) 有効化ビット

1 = IPMI サポートモードを有効化します。すべてのアドレスに応答します。  
0 = IPMI モードは有効化されません。

ビット 10 **A10M:** 10 ビットスレーブアドレスビット

1 = I2CADD は 10 ビットスレーブアドレスです。  
0 = I2CADD は 7 ビットスレーブアドレスです。

ビット 9 **DISSLW:** スルーレート制御ビットを無効化

1 = スルーレート制御を無効化します。  
0 = スルーレート制御を有効化します。

ビット 8 **SMEN:** SMBus 入力レベルビット

1 = SMBus 仕様準拠 I/O ピンしきい値を有効化します。  
0 = SMBus 入力しきい値を無効化します。

ビット 7 **GCEN:** 一斉呼び出し有効化ビット (I<sup>2</sup>C スレーブとして動作時)

1 = 一斉呼び出しアドレスが I2CRSR で受信された場合に割り込みを有効化します。  
(モジュールは受信のために有効化されます)  
0 = 一斉呼び出しアドレスが無効化されます。

ビット 6 **STREN:** SCL クロック延長有効化ビット (I<sup>2</sup>C スレーブとして動作時)。SCLREL ビットと関連して使用。

1 = クロック延長のソフトと受信を有効化する。  
0 = クロック延長のソフトと受信を無効化する。

ビット 5 **ACKDT:** 応答データビット (I<sup>2</sup>C マスターとして動作時)。マスター受信中に適用可能。

ソフトウェアが応答シーケンスを開始する際に送信される値。  
1 = 応答として ACK を送信します。  
0 = 応答として NACK を送信します。

## レジスタ 21-1: I2CCON: I<sup>2</sup>C 制御レジスタ (続き)

### ビット 4 ACKEN: 応答シーケンス有効化ビット (I<sup>2</sup>C マスターとして動作時。マスター受信時に適用可能)

1 = SDA ピンまたは SCL ピンで応答シーケンスを開始し、ACKDK データビット送信。  
ハードウェアによりマスター応答シーケンスの最後でクリアされます。  
0 = 応答シーケンスは使わない。

### ビット 3 RCEN: 受信有効化ビット (I<sup>2</sup>C マスターとして動作時)

1 = I<sup>2</sup>C の受信モードを有効化します。  
マスター受信データバイトの最後の 8 番目のビットをハードウェアがクリアします。  
0 = 受信シーケンスを有効化しない。

### ビット 2 PEN: STOP 条件有効化ビット (I<sup>2</sup>C マスターとして動作時)

1 = SDA ピンおよび SCL ピンで STOP 条件を開始します。  
マスター STOP シーケンス終了時にハードウェアによりクリアされます。  
0 = STOP 条件を有効化しない。

### ビット 1 RSEN: Repeated START 条件有効化ビット (I<sup>2</sup>C マスターとして動作時)

1 = SDA ピンおよび SCL ピンで Repeated START 条件を開始します。  
マスター繰り返し START シーケンス終了時にハードウェアでクリアされます。  
0 = Repeated START 条件を有効化しない。

### ビット 0 SEN: START 条件有効化ビット (I<sup>2</sup>C マスターとして動作時)

1 = SDA ピンおよび SCL ピンで START 条件を開始します。  
マスター START シーケンス終了時にハードウェアでクリアされます。  
0 = START 条件を有効化しない。

凡例：

R = 読み込み可能

W = 書き込み可能

C = クリア可能ビット

U = 未定義ビット、'0' が読み込まれます

HC = ハードウェアによりクリア  
済み

HS = ハードウェアにより  
設定済み

S = 設定可能ビット

'1' = ビットが POR でセットさ  
れます

'0' = ビットが POR でク  
リアされます

x = ビットが POR で不定です

レジスタ 21-2: I<sup>2</sup>CSTAT: I<sup>2</sup>C 状態レジスタ

上位バイト:							
R-0 HS, HC	R-0 HS, HC	U-0	U-0	U-0	R/C-0 HS	R-0 HS, HC	R-0 HS, HC
ACKSTAT	TRSTAT	—	—	—	BCL	GCSTAT	ADD10
ビット 15							ビット 8

下位バイト:							
R/C-0 HS	R/W-0 HS	R-0 HS, HC	R/C-0 HS, HC	R/C-0 HS, HC	R-0 HS, HC	R-0 HS, HC	R-0 HS, HC
IWCOL	I2COV	D_A	P	S	R_W	RBF	TBF
ビット 7							ビット 0

ビット 15 **ACKSTAT:** 応答状態ビット  
(I<sup>2</sup>C マスターとして動作時。マスター送信動作時に適用可能です。)  
1 = スレーブから NACK を受信しました。  
0 = スレーブから ACK を受信しました。  
スレーブ応答終了時にもハードウェアでセットまたはクリアにされます。

ビット 14 **TRSTAT:** 送信状態ビット  
(I<sup>2</sup>C マスターとして動作時。マスター送信動作時に適用可能です。)  
1 = マスター送信が進行中です (8 ビット + ACK)  
0 = マスター送信が進行していません。  
ハードウェアでマスター送信時にセットされます。  
スレーブ応答終了時にハードウェアでクリアにされます。

ビット 13-11 未実装: '0' が読み込まれます

ビット 10 **BCL:** マスターバス衝突検出ビット  
1 = マスター動作中にバス衝突が検出されました。  
0 = 衝突はありません。  
ハードウェアでバス衝突検出時にセットされます。

ビット 9 **GCSTAT:** 一斉呼び出し状態ビット  
1 = 一斉呼び出しアドレスが受信されました。  
0 = 一斉呼び出しアドレスは受信されていません。  
ハードウェアでアドレスが一斉呼び出しアドレスに一致するとセットされます。  
ハードウェアにより STOP 検出でクリアされます。

ビット 8 **ADD10:** 10 ビットアドレス状態ビット  
1 = 10 ビットアドレスが一致しました。  
0 = 10 ビットアドレスは一致しません。  
ハードウェアにより 10 ビットアドレスの 2 番目のバイトの一致でセットされます。  
ハードウェアにより STOP 検出でクリアされます。

ビット 7 **IWCOL:** 書き込み衝突検出ビット  
1 = I<sup>2</sup>C モジュールがビジー状態にあるため、I2CTRN レジスタ書き込みに失敗しました。  
0 = 衝突はありません。  
ハードウェアによりビジー状態の時に I2CTRN への書き込みが発生するとセットされます (ソフトウェアによりクリア)

ビット 6 **I2COV:** 受信オーバーフロー flag ビット  
1 = I2CRCV レジスタが以前のバイトを保持している間に新たに 1 つのバイトが受信されました。  
0 = オーバーフローはありません。  
ハードウェアにより I2CRSR から I2CRCV への転送が発生するときセットされます (ソフトウェアによりクリア)。

ビット 5 **D\_A:** データ / アドレスビット (I<sup>2</sup>C スレーブとして動作時)  
1 = 最後に受信したバイトがデータであることを示しています。  
0 = 最後に受信したバイトがデバイスアドレスであることを示しています。  
ハードウェアによりデバイスアドレス一致でクリアされます。  
ハードウェアにより I2CTRN への書き込みまたはスレーブバイトの受信でセットされます。

## レジスタ 21-2: I2CSTAT: I<sup>2</sup>C 状態レジスタ (続き)

### ビット 4 **P:** STOP ビット

1 = STOP ビットが最後に検出されたことを示します。

0 = STOP ビットは最後に検出されませんでした。

ハードウェアにより、START、Repeated START、STOP のいずれかを検出するとセットまたはクリアされます。

### ビット 3 **S:** START ビット

1 = START (または Repeated START) ビットが最後に検出されたことを示します。

0 = START ビットは最後に検出されませんでした。

ハードウェアは、START、Repeated START、STOP のいずれかを検出するとセットまたはクリアされます。

### ビット 2 **R\_W:** 読み込み / 書き込みビット情報 (I<sup>2</sup>C スレーブとして動作時)

1 = 読み込み - データ転送がスレーブからの出力であることを示します。

0 = 書き込み - データ転送がスレーブへの入力であることを示します。

ハードウェアにより I<sup>2</sup>C デバイスアドレスバイトの受信後セットまたはクリアされます。

### ビット 1 **RBF:** 受信バッファフルステータスビット

1 = 受信完了、I2CRCV は満杯です。

0 = 受信が完了していません。I2CRCV は空です。

ハードウェアにより I2CRCV に受信済みバイトが書き込まれるとセットされます。

ハードウェアによりソフトウェアが I2CRCV を読み込むとクリアされます。

### ビット 0 **TBF:** 送信バッファフルステータスビット

1 = 現在送信中。I2CTRN は満杯です。

0 = 送信完了。I2CTRN は空です。

ハードウェアによりソフトウェアが I2CTRN を書き込むとセットされます。

データ送信完了時にはハードウェアによりクリアにされます。

凡例：

R = 読み込み可能

W = 書き込み可能

C = クリア可能ビット

HC = ハードウェアによりクリア  
済み

HS = ハードウェアにより  
設定済み

U = 未定義ビット、'0' が読み込まれま  
す

'1' = ビットが POR でセットさ  
れます

'0' = ビットが POR でクリ  
アされます

x = ビットが POR で不定です  
アされます

## 21.4 I<sup>2</sup>C 動作の有効化

I2CEN ビット (I2CCON<15>) をセットして、モジュールを有効にできます。

I<sup>2</sup>C モジュールはすべてのマスターおよびスレーブ機能を実装しています。モジュールが有効化されると、マスター機能とスレーブ機能は同時にアクティブになりソフトウェアまたはバスでイベントが発生するたびに対応します。

最初に有効化された時、モジュールは SDA ピンと SCL ピンを解放し、バスを IDLE 状態にします。ソフトウェアにより制御ビットでマスターイベントを初期化するようにセットされるまで、マスター機能は IDLE 状態のままとなります。スレーブ機能はバスのモニターを開始します。スレーブロジックが START イベントを検出し有効なアドレスがバスにあると、スレーブロジックはスレーブ処理を開始します。

### 21.4.1 I<sup>2</sup>C 入出力の有効化

バスオペレーションには 2 つのピンが使用されます。クロックとしての SCL ピンと、データとしての SDA ピンです。モジュールが有効化されると、より高い優先順位を持つモジュールが制御を行っていないことを想定したうえで、モジュールは SDA ピンと SCL ピンの制御を行います。モジュールソフトウェアによってピンのポート入出力を制御しなくても、モジュールがポートの状態と方向をオーバーライドします。初期化の際は、ピンは tri 状態（解放状態）になります。

### 21.4.2 I<sup>2</sup>C 割り込み

I<sup>2</sup>C モジュールは 2 種類の割り込みを生成します。1 つ目はマスターイベントに割り当てられ、もう一方はスレーブイベントに割り当てられます。これらの割り込みは、対応する割り込みフラグビットをセットします。対応する割り込み有効化ビットがセットされ、そのビットの優先順位が高い時、ソフトウェアプロセスに割り込みをします。

マスター割り込みは MI2CIF と呼ばれ、マスターメッセージイベントが完了するとアクティブになります。

以下のイベントが MI2CIF 割り込みを生成します。

- START 条件
- STOP 条件
- データ転送バイト送信 / 受信
- 応答送信
- Repeated START
- バス衝突イベントの検出

スレーブ割り込みは SI2CIF と呼ばれ、スレーブに向けて送信されたメッセージを検出するとアクティブになります。

- 有効なデバイスアドレスの検出（一斉呼び出しを含む）
- データ送信要求
- データ受信

### 21.4.3 バスマスターとして動作している場合のボーレート設定

I<sup>2</sup>C マスターとしての動作中、モジュールはシステム SCL クロックを生成する必要があります。一般に、I<sup>2</sup>C システムクロックは 100 kHz、400 kHz または 1 MHz に指定します。システムクロックレートは最小 SCL Low タイム + 最小 SCL High タイムで指定されます。多くの場合、この値は 2TBRG インターバルで定義されます。

ボーレート生成器のリロード値は I2CBRG レジスタで、図 21-6 に示されるとおりです。ボーレート生成器がこの値でリロードされると、ジェネレータは他のリロードが発生するまで ‘0’ までカウントダウンします。各命令サイクル (TCY) ごとにジェネレータのカウントは 2 つ減分します。ボーレート再起動でボーレート生成器は自動的にリロードされます。例えば、クロック同期が発生すると、SCL ピンがハイでサンプルされた場合にボーレート生成器はリロードされます。

**注：** I2CBRG 値 0x0 はサポートされていません。

ボーレート生成器のリロード値を計算するには、以下の式を使用します。

式 21-1:

$$\text{I2CBRG} = \text{INT}((\text{FCY}) / \text{Fscl}) - 1$$

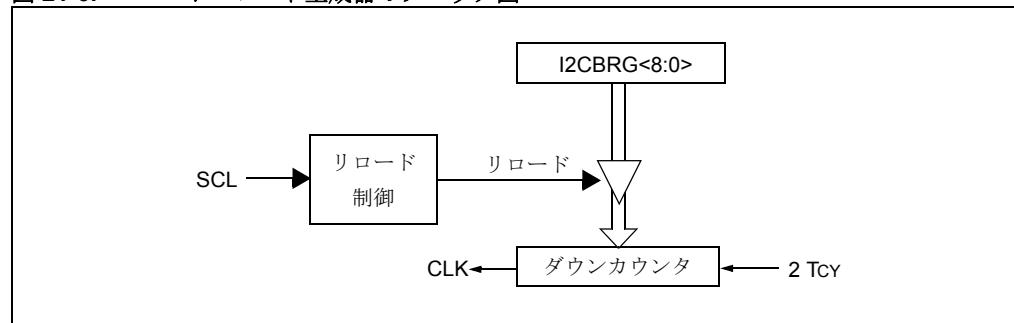
表 21-1: I<sup>2</sup>C クロックレート

必要なシステム Fscl	FCY	I2CBRG 10 進法	I2CBRG 16 進法	実質 Fscl
100 kHz	40 MHz	399	0x18F	100 kHz
100 kHz	30 MHz	299	0x12B	100 kHz
100 kHz	20 MHz	199	0x0C7	100 kHz
400 kHz	10 MHz	24	0x018	400 kHz
400 kHz	4 MHz	9	0x009	400 kHz
400 kHz	1 MHz	2	0x002	333 kHz <sup>**</sup>
1 MHz*	2 MHz	1	0x001	1 MHz <sup>*</sup>
1 MHz	1 MHz	0	0x000 (無効)	1 MHz

\*FCY = 2 MHz は Fscl = 1 MHz を保つための最小入力クロック周波数です。

\*\* これは、FCY 値を 400 kHz にするのに最も近い値です。

図 21-6: ボーレート生成器のブロック図



## 21.5 シングルマスター環境でマスターとして通信

システム内における一般的な I<sup>2</sup>C モジュールオペレーションとしては、シリアルメモリなどの I<sup>2</sup>C 周辺機器との通信に使用します。I<sup>2</sup>C システムでは、マスターがバス上のすべてのデータ通信のシーケンスを制御します。この例では、dsPIC30F の I<sup>2</sup>C モジュールはシステム内でシングルマスターの役割を果たしています。シングルマスターとして、SCL クロックを生成しメッセージプロトコルを制御します。

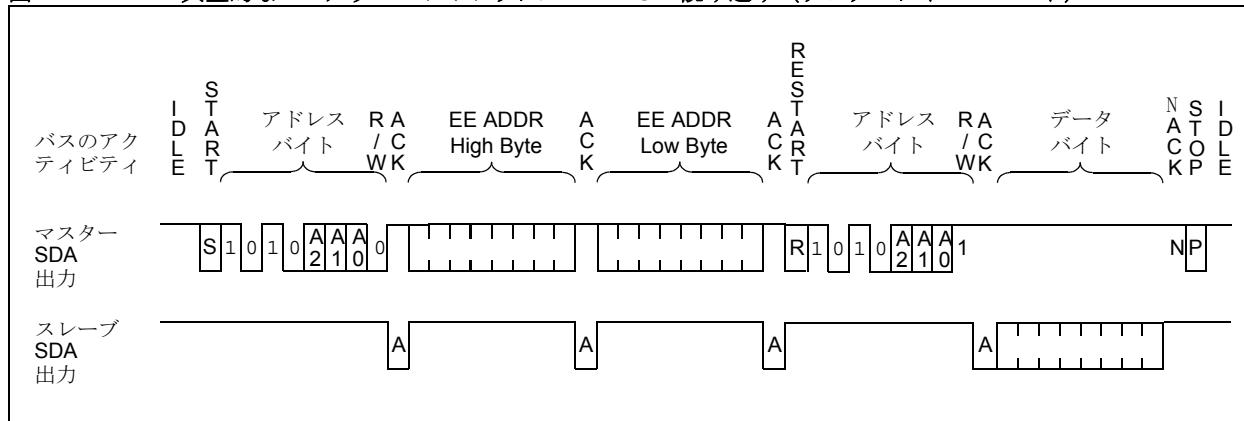
I<sup>2</sup>C モジュールでは、モジュールが I<sup>2</sup>C メッセージプロトコルの各部分を制御しますが、完全なメッセージを構築するためにプロトコルのシーケンスを制御するのはソフトウェアの仕事になります。

例えば、シングルマスター環境における典型的な動作として I<sup>2</sup>C シリアル EEPROM からのバイトの読み込みがあります。メッセージ例は図 21-7 に示すとおりです。

このメッセージを生成するためにソフトウェアは以下の手順でシーケンス制御を行います。

1. SDA および SCL で START 条件を出力します。
2. I<sup>2</sup>C デバイスアドレスバイトを書き込み指示とともにスレーブに送信します。
3. スレーブからの応答を待ち、確認します。
- 4.シリアルメモリアドレス高位バイトをスレーブに送信します。
5. スレーブからの応答を待ち、確認します。
- 6.シリアルメモリアドレス低位バイトをスレーブに送信します。
7. スレーブからの応答を待ち、確認します。
8. SDA および SCL で Repeated START 条件を出力します。
9. デバイスアドレスバイトを読み込み指示とともにスレーブに送信します。
10. スレーブからの応答を待ち、確認します。
11. マスター受信を有効にしシリアルメモリデータを受信します。
12. 受信したデータの最後のバイトで ACK または NACK 条件を生成します。
13. SDA および SCL で STOP 条件を生成します。

図 21-7: 典型的な I<sup>2</sup>C メッセージ：シリアル EEPROM 読み込み（ランダムアドレスモード）



I<sup>2</sup>C モジュールはマスターモード通信を START および STOP ジェネレータ、データバイト送信、データバイト受信、応答ジェネレータおよびボーレートジェネレータでサポートします。

通常、ソフトウェアはコントロールレジスタに書き込みを行って特定の手順を開始して割り込みを待つか、完了のためのポーリング行って待ちます。

後続のセクションでこれらのオペレーションについて詳細を説明します。

**注：** I<sup>2</sup>C モジュールはイベントのキューを許可しません。例えば、ソフトウェアで、START 条件を開始してすぐに I2CTRN レジスタを書き込み、START 条件が完了する前に通信を開始することはできません。この場合、I2CTRN は書き込まれず、書き込みが発生しなかったことを示す IWCOL ビットがセットされます。

## 21.5.1 START バスイベントの生成

START イベントを開始するため、ソフトウェアは START 有効化ビット SEN (I2CCON<0>) をセットします。START ビットをセットする前に、ソフトウェアは P (I2CSTAT<4>) 状態ビットをチェックしてバスが IDLE 状態にあることを確認できます。

図 21-8 は START 条件のタイミングを示しています。

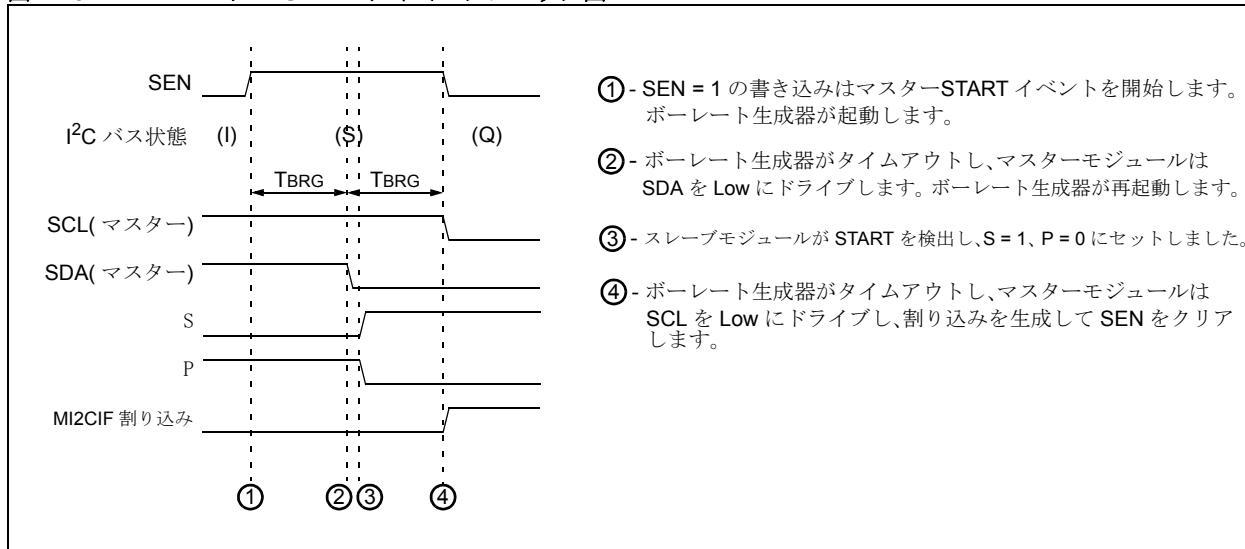
- スレーブロジックは START 条件を検出して S ビット (I2CSTAT<3>) をセットし、P ビット (I2CSTAT<4>) をクリアします。
- SEN ビットは START 条件の完了時に自動的にクリアされます。
- START 条件完了により MI2CIF 割り込みが生成されます。
- START 条件の後、SDA 線と SCL 線は Low (Q 状態) のままになります。

### 21.5.1.1 IWCOL 状態フラグ

START シーケンス進行中にソフトウェアが I2CTRN を書き込むと、IWCOL がセットされ送信バッファの内容が変更不可になります（書き込みは発生しません）。

**注：** イベントのキューは許可されていないため、I2CCON の下位 5 ビットへの書き込みは START 条件が完了するまで無効化されます。

図 21-8: マスター START タイミングブロック図



### 21.5.2 スレーブデバイスへのデータ送信

データバイト、7 ビットデバイスアドレスバイト、10 ビットアドレスの 2 番目のバイトは、I<sup>2</sup>CTR<sub>N</sub> レジスタに適切な書き込みを行うだけで送信できます。このレジスタにロードすると以下のプロセスが開始されます。

- ・ソフトウェアで送信するデータバイトを I<sup>2</sup>CTR<sub>N</sub> にロードします。
- ・I<sup>2</sup>CTR<sub>N</sub> 書き込みによりバッファフルフラグビットが TBF (I<sup>2</sup>CSTAT<0>) にセットされます。
- ・すべての 8 ビットデータが送信されるまで、データバイトが SDA ピンにシフト出力します。SCL のエッジ立下り後、アドレス/データの各ビットが SDA ピンにシフトします。
- ・9 番目の SCL クロックでモジュールがスレーブデバイスから ACK ビットに出力されるので、受信した値を ACKSTAT ビット (I<sup>2</sup>CCON<15>) に書き込みます。
- ・モジュールは 9 番目の SCL クロックサイクルの最後に MI2CIF 割り込みを生成します。

モジュールはデータバイトを生成、有効化しないことに注意してください。バイトの内容と用途はソフトウェアによるメッセージプロトコルの状態により異なります。

#### 21.5.2.1 7 ビットアドレスのスレーブへの送信

7 ビットデバイスアドレスを送信するには、バイトをスレーブに送信する必要があります。7 ビットアドレスバイトは 7 ビットの I<sup>2</sup>C デバイスアドレスとメッセージがスレーブへの書き込みか（マスターが送信、スレーブが受信）またはスレーブからの読み出しか（スレーブが送信、マスターが受信）を定義する R/W ビットを含む必要があります。

#### 21.5.2.2 10 ビットアドレスのスレーブへの送信

10 ビットデバイスアドレスを送信するには、2 つのバイトをスレーブに送信する必要があります。1 番目のバイトは 10 ビットアドレッシングモードのために決められている I<sup>2</sup>C デバイスアドレスの 5 ビットと 10 ビットアドレスの 2 ビットを含みます。2 番目のバイトはスレーブで受信されるべき 10 ビットアドレスの残りの 8 ビットを含むため、1 番目のバイトの R/W ビットはマスターが送信しスレーブが受信することを意味する ‘0’ にする必要があります。メッセージデータがスレーブに向けて送信される場合、マスターはデータの送信を続行できます。ただし、マスターがスレーブからの返信を求めている場合、R/W ビットが ‘1’ の Repeated START シーケンスを出力してステータスをスレーブ読み込みに変更します。

#### 21.5.2.3 スレーブから応答を受信

8 番目の SCL クロックの立下りエッジでは、TBF ビットはクリアされマスター側が出力を空けて、SDA ピンにスレーブが応答による返信が outputできるようにします。その後、マスターは 9 番目の SCL クロックを生成します。

これにより、アドレス一致が検出された場合やデータが正常に受信された場合にスレーブデバイスが 9 ビット目に ACK ビットでの応答が返せるようにします。スレーブはデバイスアドレス（一齊呼び出しを含む）が認識された場合やスレーブが正常にデータを受信した場合に応答を送信します。

ACK のステータスが、応答ステータスピット ACKSTAT (I<sup>2</sup>CSTAT<15>) に、9 番目の SCL クロックの立下りエッジで書き込まれます。9 番目の SCL クロックの後、モジュールは MI2CIF 割り込みを生成し次のデータバイトが I<sup>2</sup>CTR<sub>N</sub> にロードされるまで IDLE 状態になります。

#### 21.5.2.4 ACKSTAT ステータスフラグ

ACKSTAT ビット (I<sup>2</sup>CCON<15>) はスレーブが応答 (ACK = 0) を送信するとクリアされ、スレーブが応答しない (ACK = 1) とセットされます。

## 21.5.2.5 TBF ステータスフラグ

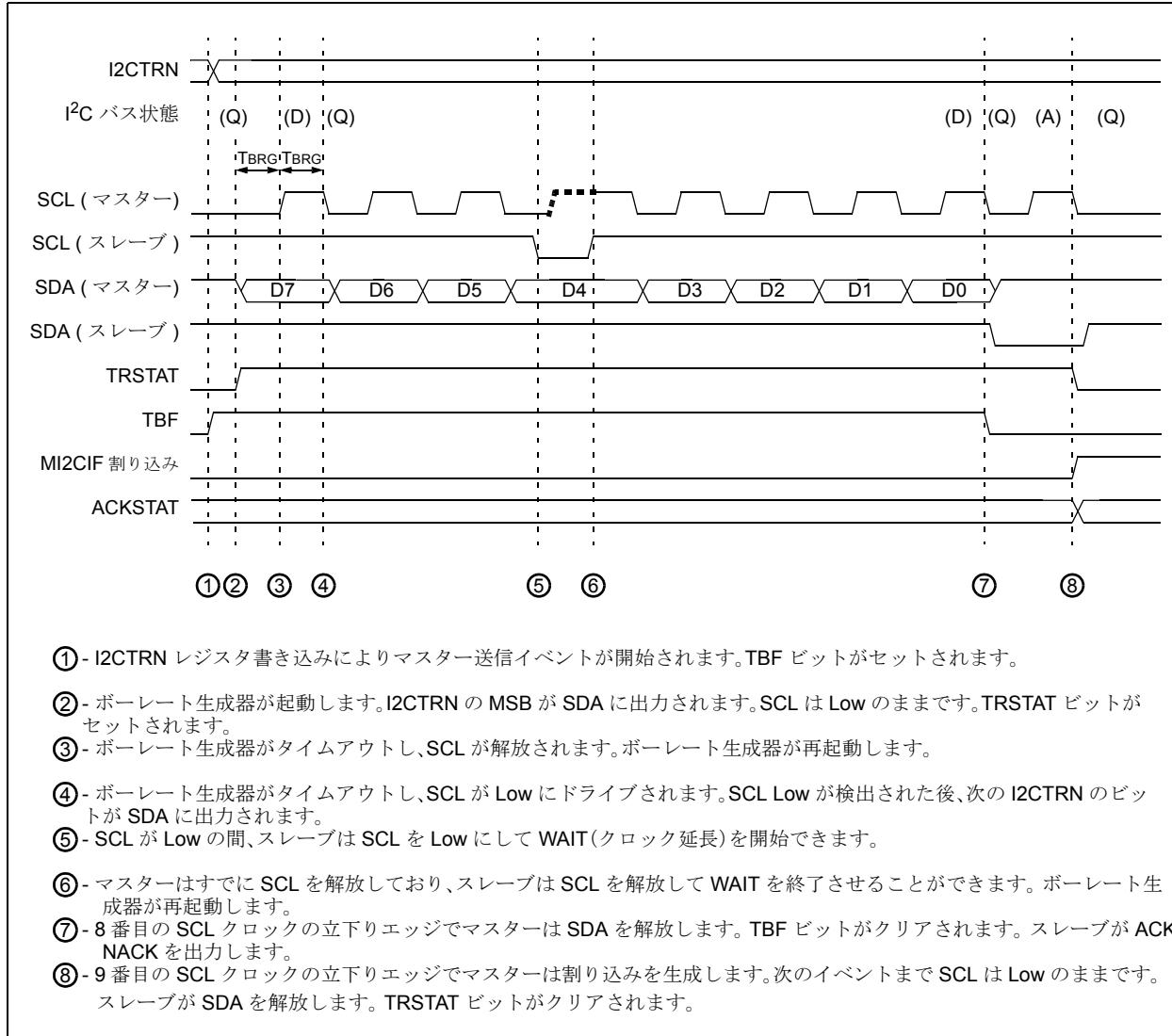
送信時に CPU が I2CTRН に書き込みを行うと TBF ビット (I2CSTAT<0>) がセットされ、すべての 8 ビットがシフト出力されるとクリアされます。

## 21.5.2.6 IWCOL ステータスフラグ

送信進行中（モジュールはデータバイトをシフト出力中です）にソフトウェアが I2CTRН に書き込むと、IWCOL がセットされ送信バッファの内容が変更不可になります（書き込みは発生しません）。IWCOL はソフトウェアでクリアされる必要があります。

**注：** イベントのキューは許可されていないため、I2CCON の下位 5 ビットへの書き込みは送信条件が完了するまで無効化されます。

図 21-9：マスター送信タイミングプロック図



### 21.5.3 スレーブデバイスからのデータ受信

受信有効化ビット RCEN (I2CCON<3>) をセットすると、マスターはスレーブデバイスからのデータを受信できるようになります。

**注：** I2CCON の下位 5 ビットは RCEN ビットをセットする前に ‘0’ にする必要があります。これによりマスターロジックは確実に非アクティブになります。

マスターロジックは SCL の各立下りエッジの前にクロック生成を開始し、SDA 線がサンプルされ、データは I2CRSR にシフト入力されます。

8 番目の SCL クロックの立下りエッジ後、以下が実行されます。

- RCEN ビットが自動的にクリアされます。
- I2CRSR のコンテンツが I2CRCV に移行されます。
- RBF フラグビットがセットされます。
- モジュールが MI2CIF 割り込みを生成します。

CPU がバッファを読み込むと、RBF フラグビットが自動的にクリアされます。ソフトウェアはデータを処理した後、応答シーケンスを行います。

#### 21.5.3.1 RBF ステータスフラグ

データ受信時に、デバイスアドレスまたはデータバイトが I2CRSR から I2CRCV にロードされると RBF ビットがセットされます。ソフトウェアが I2CRCV レジスタを読み込むとクリアされます。

#### 21.5.3.2 I2COV ステータスフラグ

RBF ビットがセットされたまま前のバイトが I2CRCV レジスタに残った状態で他のバイトが I2CRSR で受信されると、I2COV ビットがセットされ I2CRSR 内のデータは失われます。

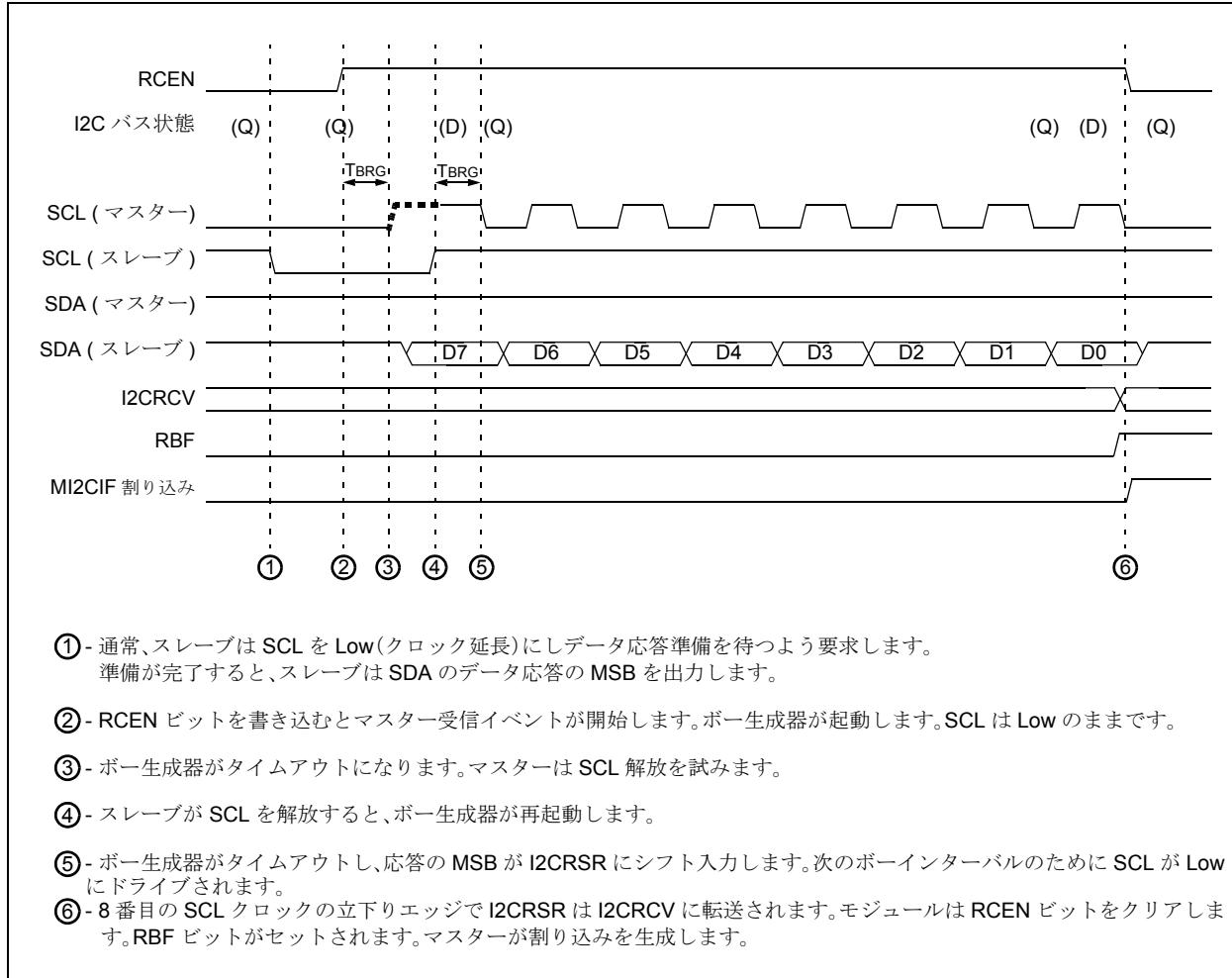
I2COV がセットされたままでも他のバイト受信は禁止されません。I2CRCV を読み込むと RBF はクリアされ、I2CRSR は他のバイトを受信し、そのバイトが I2CRCV に転送されます。

#### 21.5.3.3 IWCOL ステータスフラグ

受信進行中 (I2CRSR はデータバイト内でシフト入力中です) にソフトウェアが I2CTRN を書き込むと、IWCOL がセットされ送信バッファの内容が変更不可になります (書き込みは発生しません)。

**注：** イベントのキューは許可されていないため、I2CCON の下位 5 ビットへの書き込みは受信条件が完了するまで無効化されます。

図 21-10: マスター受信タイミングプロック図



### 21.5.4 応答生成

応答生成有効化ビット ACKEN (I2CCON<4>) をセットすると、マスターは応答シーケンスを生成できるようになります。

**注：** I2CCON の下位 5 ビットは ACKEN ビットをセットする前に ‘0’ (マスターロジック非アクティブ) にする必要があります。

図 21-11 は ACK シーケンスを示し、図 21-12 は NACK シーケンスを示しています。応答データビット ACKDT (I2CCON<5>) は ACK または NACK を指定します。

2 つのボーリー期間後、以下が実行されます。

- ACKEN ビットが自動的にクリアされます。
- モジュールが MI2CIF 割り込みを生成します。

#### 21.5.4.1 IWCOL ステータスフラグ

応答シーケンス進行中にソフトウェアが I2CTRН を書き込むと、IWCOL がセットされバッファの内容が変更不可になります（書き込みは発生しません）。

**注：** イベントのキューは許可されていないため、I2CCON の下位 5 ビットへの書き込みは応答条件が完了するまで無効化されます。

図 21-11：マスター応答 (ACK) タイミングブロック図

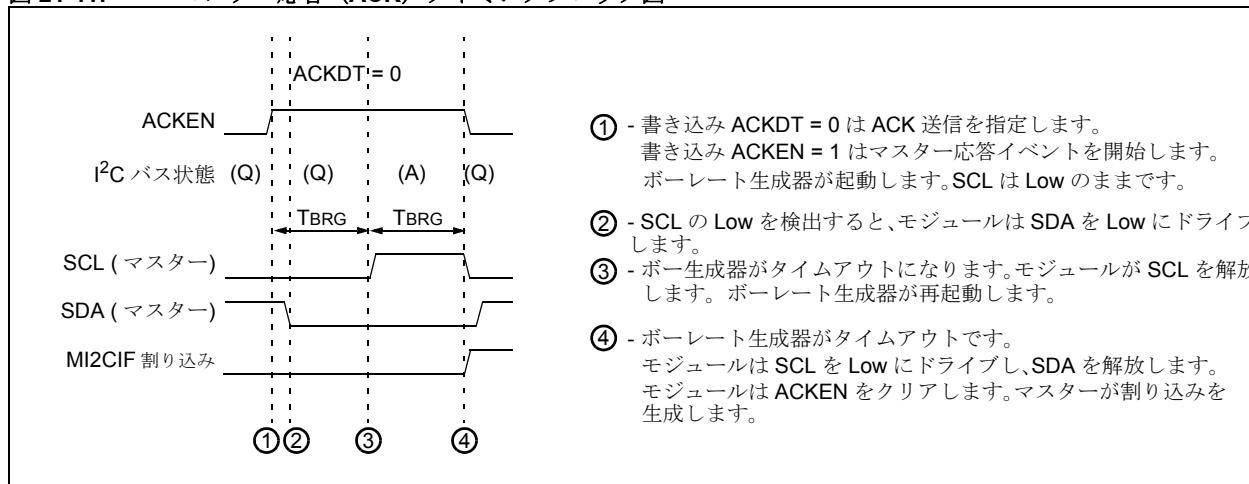
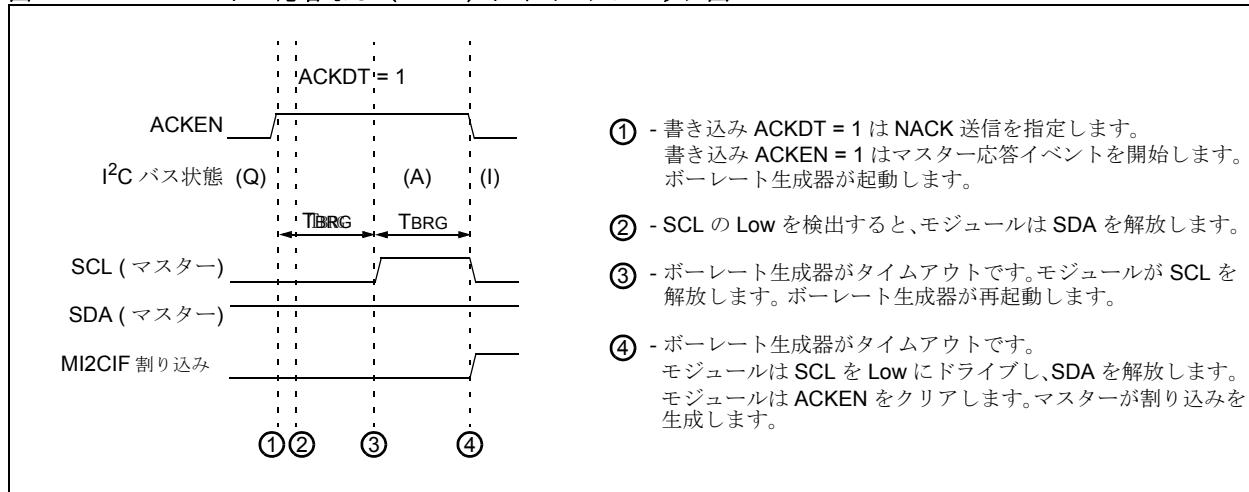


図 21-12：マスター応答なし (NACK) タイミングブロック図



## 21.5.5 STOP バスイベントの生成

STOP シーケンス有効化ビット PEN (I2CCON<2>) をセットすると、マスターは STOP シーケンスを生成できるようになります。

**注：** I2CCON の下位 5 ビットは PEN ビットをセットする前に ‘0’ (マスターロジック非アクティブ) にする必要があります。

PEN ビットがセットされると、マスターは図 21-13 で示すように STOP シーケンスを生成します。

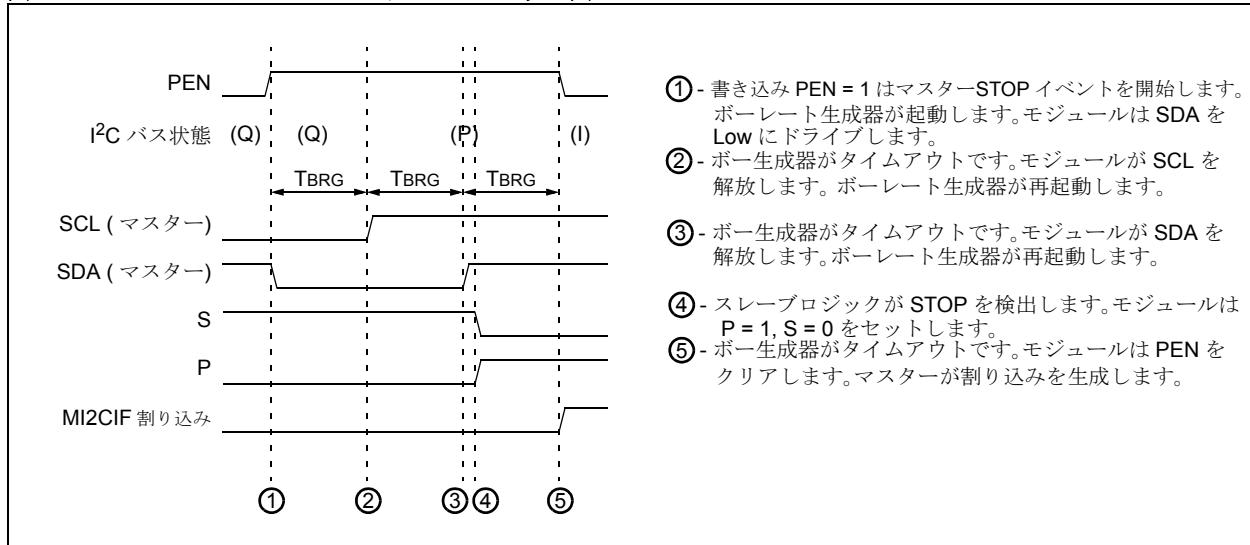
- スレーブは STOP 条件を検出して P ビット (I2CSTAT<4>) をセットし、S ビット (I2CSTAT<3>) をクリアします。
- PEN ビットが自動的にクリアされます。
- モジュールが MI2CIF 割り込みを生成します。

### 21.5.5.1 IWCOL ステータスフラグ

STOP シーケンス進行中にソフトウェアが I2CTRN を書き込むと、IWCOL ビットがセットされバッファの内容が変更不可になります（書き込みは発生しません）。

**注：** イベントのキューは許可されていないため、I2CCON の下位 5 ビットへの書き込みは STOP 条件が完了するまで無効化されます。

図 21-13: マスター STOP タイミングブロック図



### 21.5.6 Repeated START バスイベントの生成

Repeated START シーケンス有効化ビット RSEN (I2CCON<1>) をセットすると、マスターは Repeated START シーケンスを生成できるようになります (図 21-14 参照)。

**注:** I2CCON の下位 5 ビットは RSEN ビットをセットする前に ‘0’ (マスターロジック非アクティブ) にする必要があります。

Repeated START 条件を生成するため、ソフトウェアは RSEN ビット (I2CCON<1>) をセットします。モジュールが SCL ピンを Low に制御します。モジュールが SCL ピンの Low をサンプルすると、モジュールは 1 つのボーレート期間 (TBRG) だけ SDA ピンを解放します。ボーレート生成器がタイムアウトになると、モジュールが SDA を High でサンプルしている場合、モジュールは SCL ピンを解放します。モジュールが SCL ピンを High でサンプルしている場合、ボーレート生成器はリロードを行いカウントを開始します。1TBRG 期間だけ SDA と SCL は High でサンプルされる必要があります。この動作の後、1 TBRG 期間だけ SDA ピンに Low 出力をしてから、SCL が High になります。

以下が Repeated START シーケンスです。

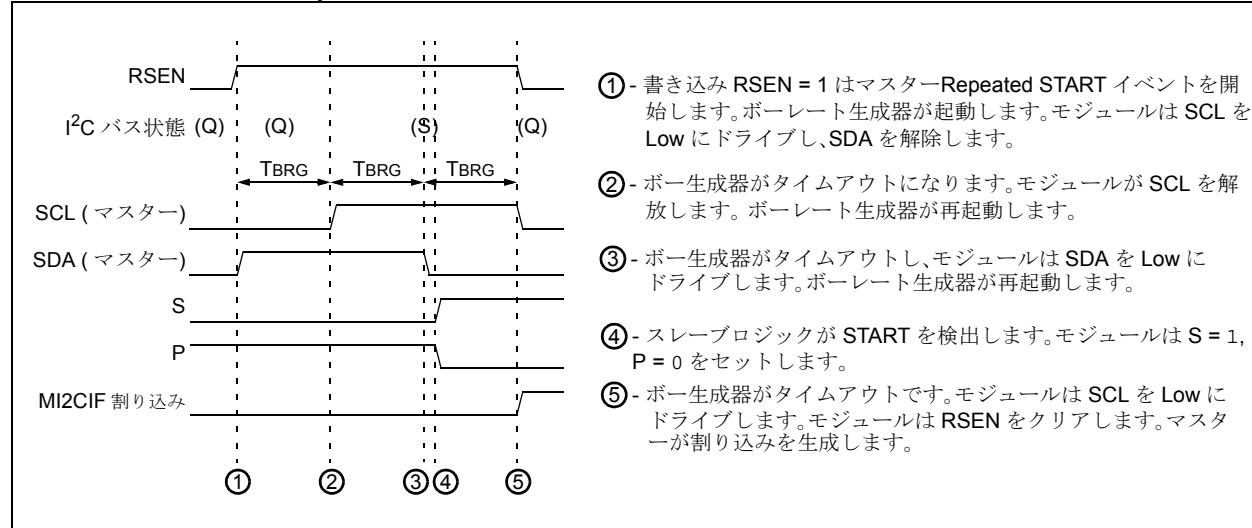
- スレーブは START 条件を検出して S ビット (I2CSTAT<3>) をセットし、P ビット (I2CSTAT<4>) をクリアします。
- RSEN ビットが自動的にクリアされます。
- モジュールが MI2CIF 割り込みを生成します。

#### 21.5.6.1 IWCOL ステータスフラグ

繰り返し START シーケンス進行中にソフトウェアが I2CTRN を書き込むと、IWCOL がセットされバッファの内容が変更不可になります (書き込みは発生しません)。

**注:** イベントのキューは許可されていないため、I2CCON の下位 5 ビットへの書き込みは繰り返し START 条件が完了するまで無効化されます。

図 21-14: マスター Repeated START タイミングブロック図



## 21.5.7 完全なマスターメッセージの構築

セクション 21.5で述べたように、ソフトウェアには正確なメッセージプロトコルでメッセージを構築する役割があります。モジュールは I<sup>2</sup>C メッセージプロトコルの各部分を制御しますが、完全なメッセージを構築するためにプロトコルの構成要素をシーケンスするのはシステムのタスクになります。

ソフトウェアはモジュール使用中にポーリングや割り込みメソッドを使用できます。以下の例では割り込みが使用されています。

ソフトウェアは SEN、RSEN、PEN、RCEN および ACKEN ビット (I2CCON レジスタの下位 5 ビット) を使用でき、メッセージ処理中に TRSTAT ビットをステータスフラグとして使用できます。例えば、表 21-2 はバスの状態に関するいくつかのステータス番号を示しています。

表 21-2: マスターメッセージプロトコルステータス

状態番号例	I2CCON<4:0>	TRSTAT (I2CSTAT<14>)	状態
0	00000	0	バスが IDLE または WAIT
1	00001	n/a	START イベント送信
2	00000	1	マスター送信
3	00010	n/a	Repeated START イベント送信
4	00100	n/a	STOP イベント送信
5	01000	n/a	マスター受信
6	10000	n/a	マスター応答

注： ステータス番号例は参照のためのものです。ユーザーのソフトウェアでは番号は自由に割り振ることができます。

ソフトウェアは START コマンド発行によりメッセージを開始できます。ソフトウェアは START に対応するステータス番号を記録します。

各イベントが完了し割り込みが生成されるごとに割り込みハンドラがステータス番号をチェックします。そのため、START 状態では、割り込みハンドラが START シーケンス実行を確認しマスター送信イベントを開始して I<sup>2</sup>C デバイスアドレスに送信します。この時、ステータス番号は対応するマスター送信に変更されます。

次の割り込みでは、割り込みハンドラは再度ステータスをチェックし、マスター送信が完了していることを確認します。割り込みハンドラはデータ送信が正常に完了していることを確認するとメッセージの内容に従って次のイベントを開始します。

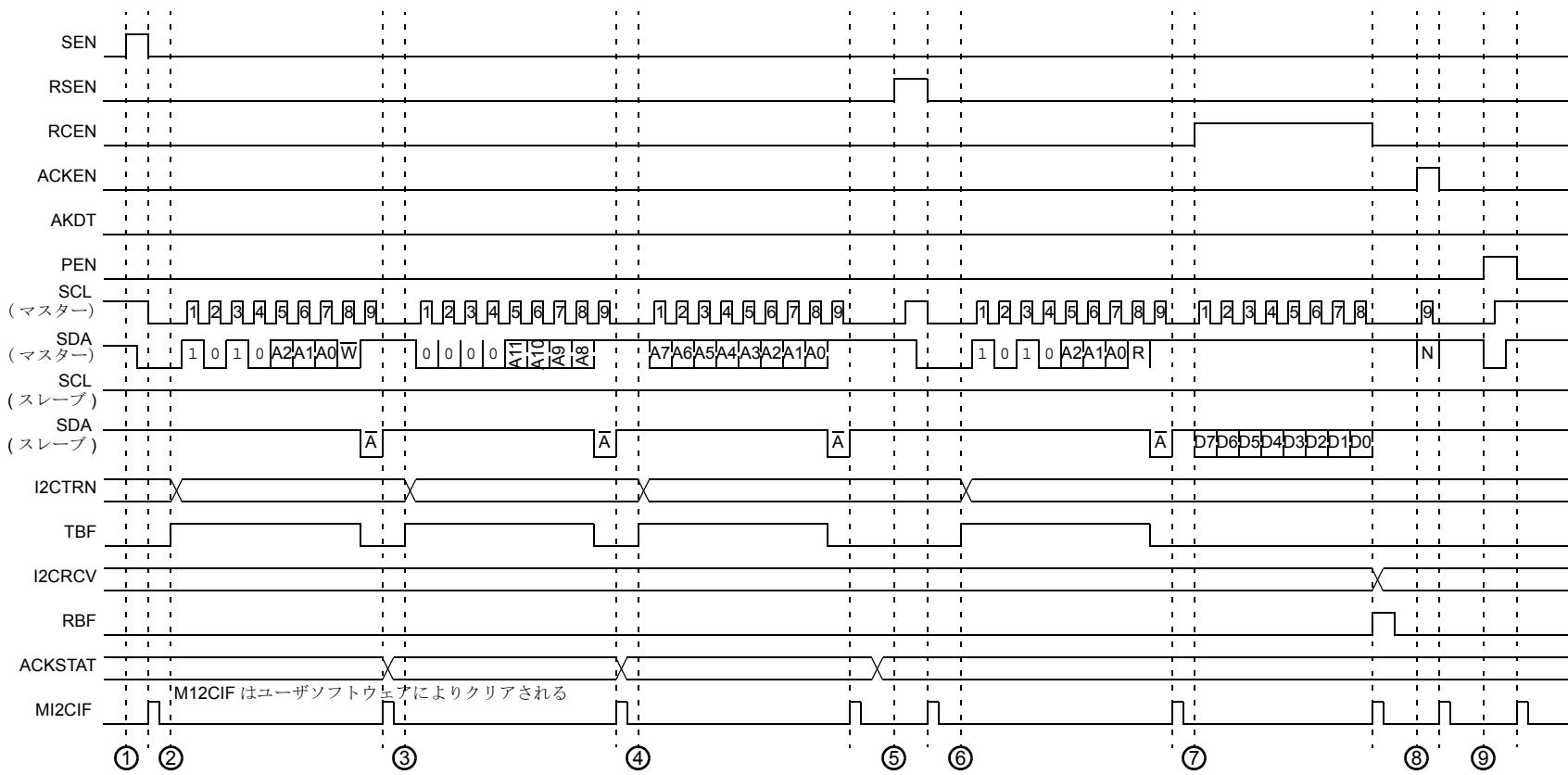
このようにして、割り込みが発生するたびに、割り込みハンドラがメッセージプロトコルをメッセージ送信の正常な完了まで監視します。

図 21-15 は図 21-7 と同じメッセージシーケンスをさらに詳しく解説したものです。

図 21-16 は 7 ビットアドレスフォーマットを使用したメッセージの簡単な例です。

図 21-17 はスレーブへの 10 ビットアドレスフォーマットメッセージ送信データの例を示しています。

図 21-18 はスレーブからの 10 ビットアドレスフォーマットメッセージ受信データの例を示しています。

図 21-15: マスターメッセージ (一般的な I<sup>2</sup>C メッセージ: シリアル EEPROM 読み込み)

- ① - SEN ビットセットで START イベントを開始。
- ② - I2CTRN レジスタ書き込みでマスター送信を開始。データはシリアル EE デバイス アドレスバイトで、R/W クリアで書き込みを示しています。
- ③ - I2CTRN レジスタ書き込みでマスター送信を開始。データは EE データアドレスの 1 番目のバイトです。
- ④ - I2CTRN レジスタ書き込みでマスター送信を開始。データは EE データアドレス の 2 番目のバイトです。
- ⑤ - RSEN ビットセットで Repeated START イベントを開始

- ⑥ - I2CTRN レジスタ書き込みでマスター送信を開始。データはシリアル EE デバイス アドレスバイトの再送信ですが、R/W セットで読み込みを示しています。
- ⑦ - RCEN ビットセットでマスター受信を開始割り込み時には、ソフトウェアで I2CRCV レジスタを読み込みます。これで RBF フラグがクリアされます。
- ⑧ - ACKEN ビットセットで応答イベントを開始。ACKDT = 0 で NACK を送信します。
- ⑨ - PEN ビットセットでマスター STOP イベントを開始

図 21-16: マスターメッセージ (7 ビットアドレス: 送信と受信)

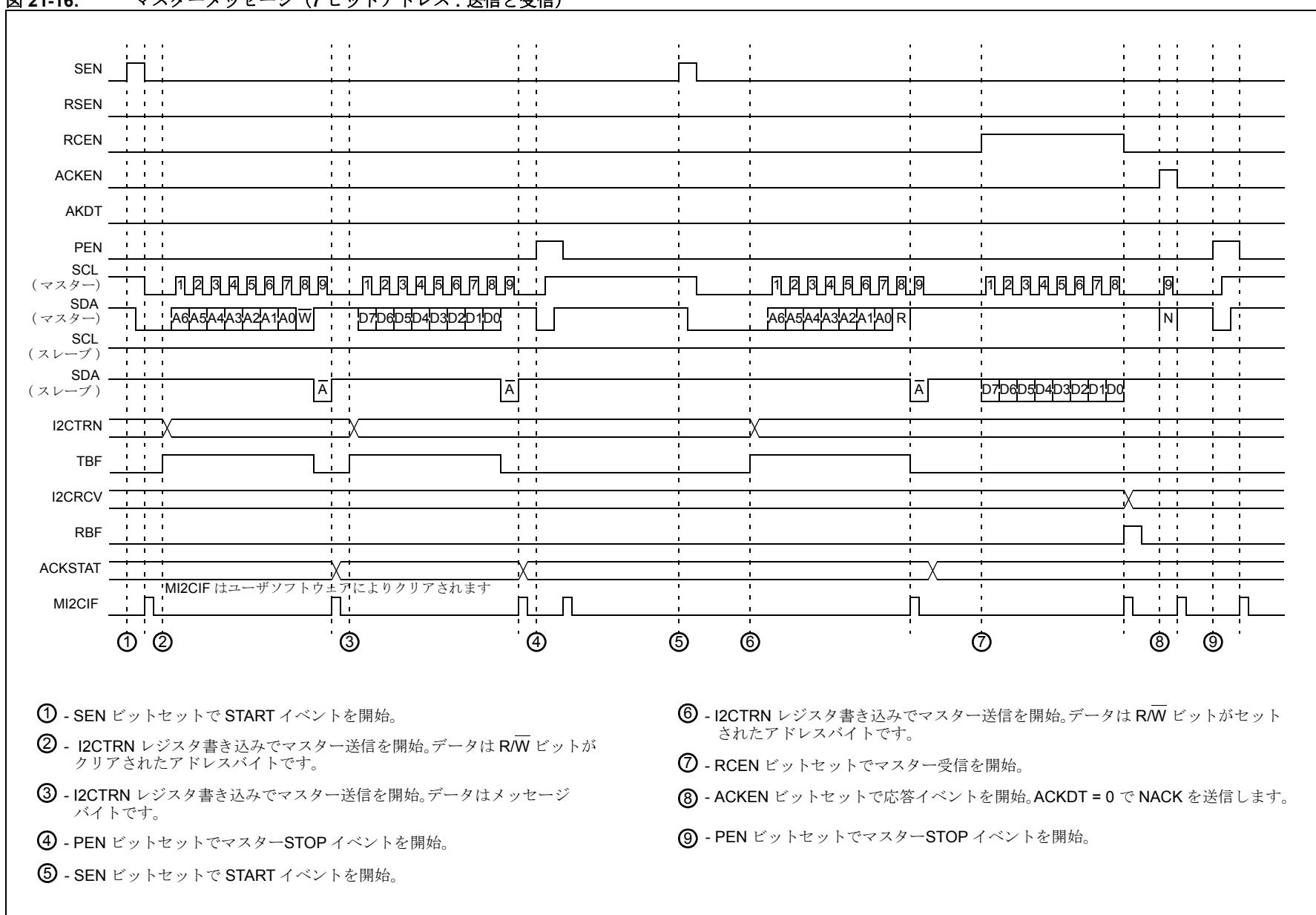


図 21-17: マスターメッセージ (10 ビット送信)

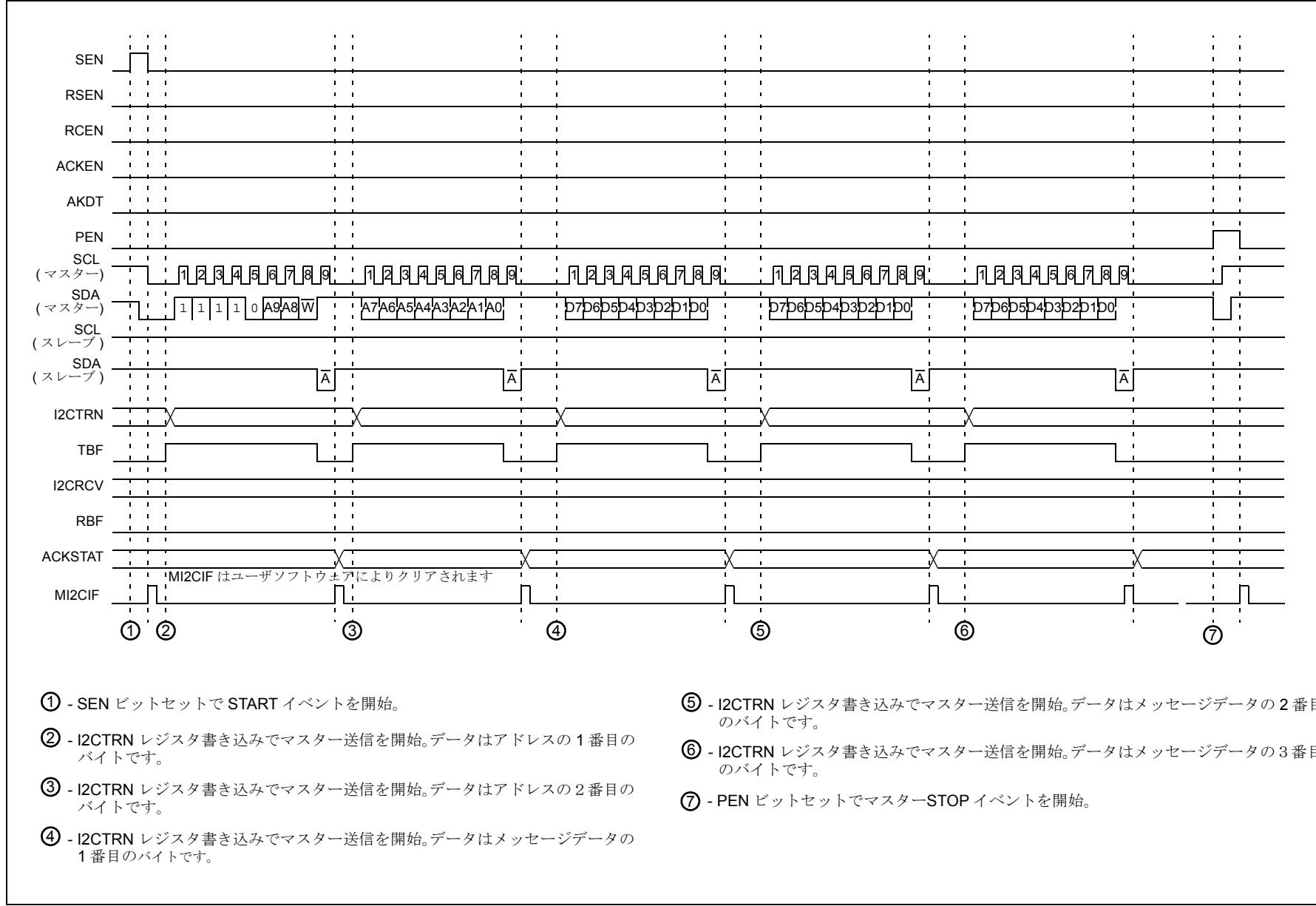
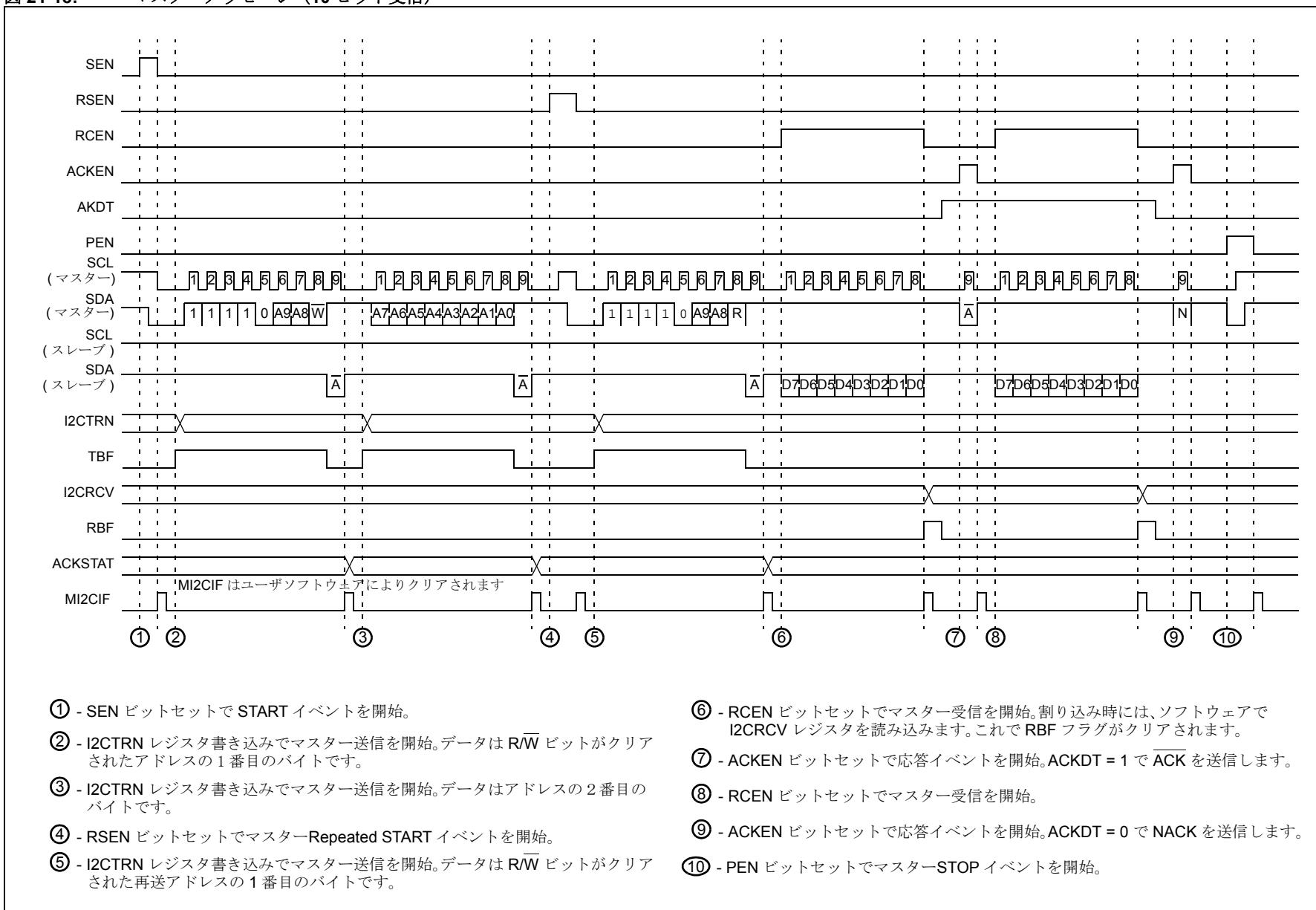


図 21-18: マスターメッセージ (10 ビット受信)



## 21.6 マルチマスター環境でマスターとして通信

I<sup>2</sup>C プロトコルでは 1 つ以上のマスターをシステムバスに接続することが許可されています。マスターはメッセージ処理を開始しバスのためのクロックを生成できるので、プロトコルは複数のマスターがバスを制御する状況を考慮したメソッドを持ちます。クロック同期により、複数のノードが SCL 線に出力される 1 つの共通クロックに同期して動作するようになります。バスアービトリレーションにより複数のノードが同時にメッセージ転送をしようとしたとき、1 つのマスターだけが転送を完了できるようにします。他のノードはバスアービトリレーションを失い、バス衝突発生として待たれます。

### 21.6.1 複数マスター操作

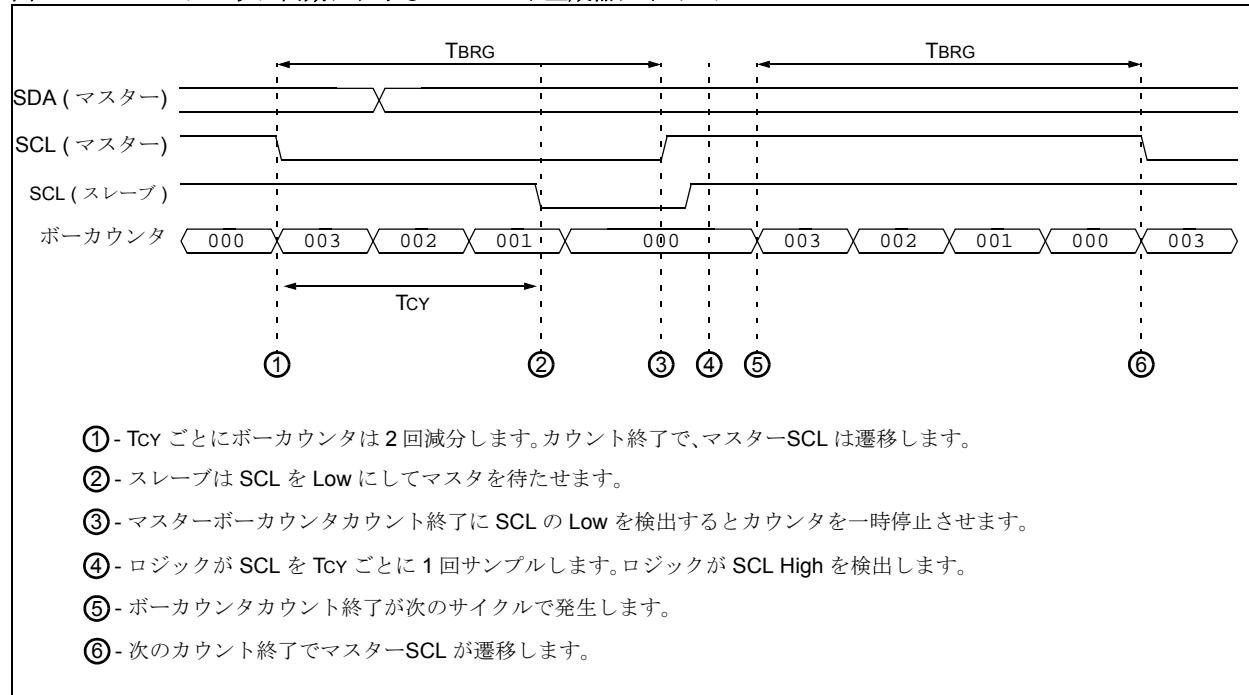
マスターモジュールは複数マスター操作のために特殊な設定を必要としません。モジュールはクロック同期とバスアービトリレーションを常に実行しています。モジュールがシングルマスター環境で使用されている場合、クロック同期はマスターとスレーブの間でのみ发生し、バスアービトリレーションは発生しません。

### 21.6.2 マスタークロック同期

マルチマスターシステムでは、異なるマスターが異なるボーレートを持ちます。クロック同期により、これらのマスターがバスのアービトリレーションを試みる場合にそのクロックに合わせて動作します。

クロック同期はマスターが SCL ピンを解放したときに行われます。(SCL はフローで High になります) SCL ピンが解放されると、ボーレート生成器 (BGS) は SCL ピンが実際に High でサンプルされるまで停止します。SCL ピンが High でサンプルされると、ボーレート生成器は I<sup>2</sup>CBRG<8:0> の内容がリロードされカウントを開始します。これにより、図 21-19 で示す通り、SCL High タイムは常にイベント内の 1 つ以上の BRG カウント周期となり、さらにクロックは外部デバイスにより Low にしたままにすることもできます。

図 21-19: クロック同期におけるボーレート生成器タイミング



## 21.6.3 バスアービトレーションとバス衝突

バスアービトレーションは複数マスターシステムオペレーションをサポートします。

SDA 線の **wired-and** 機能によりアービトレーションが可能になります。アービトレーションは 1 番目のマスターが SDA に ‘1’ を出力してフロート High の状態にしようとしたとき、同時に 2 番目のマスターが、SDA に ‘0’ を出力して、SDA 線を Low にしたとき発生します。SDA 信号はローになります。このケースでは、2 番目のマスターがバスアービトレーションを獲得します。1 番目のマスターはバスアービトレーションを失い、バス衝突発生となります。

1 番目のマスターでは、SDA で求められるデータは ‘1’ ですが、SDA でサンプルされたデータは ‘0’ となっています。これがバス衝突の定義となります。

1 番目のマスターはバス衝突ビット **BCL(I2CSTAT<10>)** をセットし、マスター割り込みを生成します。マスターモジュールが I<sup>2</sup>C ポートを IDLE 状態にリセットします。

複数マスターオペレーションでは、SDA 線はアービトレーションのために信号レベルが期待される出力レベルとなっているかどうかを、モニターされる必要があります。このモニターはマスターモジュールで実行され、結果は BCL ビットに置かれます。

アービトレーションが失われる状態は以下の通りです。

- START 条件
- Repeated START 条件
- アドレス、データまたは応答ビット
- STOP 条件

## 21.6.4 バス衝突と再送信メッセージの検出

バス衝突が発生すると、モジュールは BCL ビットをセットしマスター割り込みを生成します。バイト送信中に衝突が発生した場合、送信は中止され TBF フラグがクリアされて SDA ピンと SCL ピンが解放されます。START、Repeated START、STOP または応答実行中にバス衝突が発生した場合、その実行は破棄され、I2CCON レジスタ内の各制御ビットがクリアされて SDA 線および SCL 線が解放されます。

ソフトウェアはマスターイベントの完了時の割り込みを常時待っています。ソフトウェアは BCL ビットをチェックしてマスターイベントが正常に完了したか、衝突が発生したかを確認します。衝突が発生した場合、ソフトウェアは保留中の残りのメッセージ送信を中止し、バスが IDLE 状態に戻った後に START 条件で始まる完全なメッセージシーケンスを再送信する準備をします。ソフトウェアは S ビットと P ビットをモニターして IDLE バスを待機します。ソフトウェアがマスター割り込みサービスルーチンを実行し、I<sup>2</sup>C バスがフリーの場合、ソフトウェアは START 条件を出力して通信を再開できます。

### 21.6.5 START 条件期間中のバス衝突

START コマンドを発行する前に、ソフトウェアは S ステータスビットと P ステータスビットを使用してバスの IDLE ステータスを確認する必要があります。2 つのマスターがほぼ同時期にメッセージ開始を試みている可能性があります。通常、これらのマスターはクロックを同期し一方がアビトトレーションを失うまでメッセージのアビトトレーションを続行します。ただし、ある条件により START 期間中にバス衝突が発生することもあります。このように、START ビット期間中にアビトトレーションを失ったマスターはバス衝突割り込みを発生します。

### 21.6.6 Repeated START 条件期間中のバス衝突

2 つのマスターがアドレスバイトで衝突を起こさなかったとしても、一方が Repeated START の出力を試み、もう一方がデータを送信している時にバス衝突が発生する可能性があります。この場合、Repeated START を生成したマスターがアビトトレーションを失い、バス衝突割り込みを生成します。

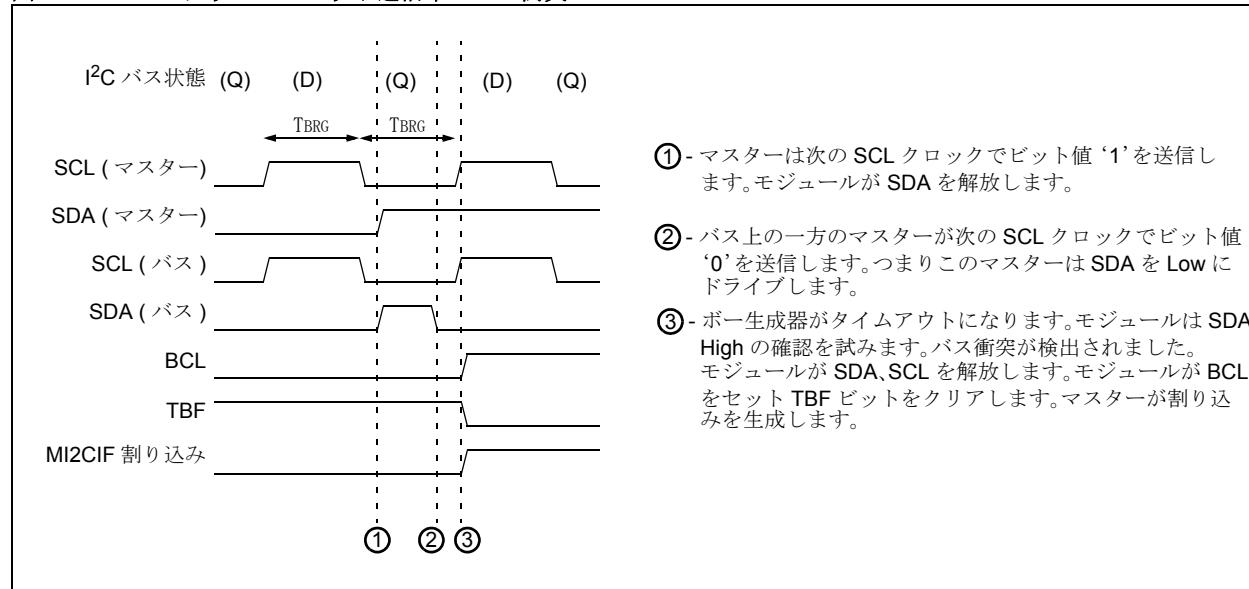
### 21.6.7 メッセージビット送信中のバス衝突

最も一般的なデータ衝突はマスターがデバイスアドレスバイト、データバイト、または応答ビットの送信を試みている時に発生します。

ソフトウェアが適切にバスの状態をチェックしていれば、START 条件でバス衝突が発生することはほとんどありません。ただし、他のマスターがほぼ同時にバスをチェックし START 条件を開始している場合、SDA アビトトレーションが発生して 2 つのマスターを同期させることができます。この条件では、両方のマスターがメッセージの送信を開始し、他方のマスターがメッセージビットのアビトトレーションを失うまでこれが続行されます。SCL クロック同期は一方がアビトトレーションを失うまで 2 つのマスターを同期させる点に注意してください。

図 21-20 はメッセージビットのアビトトレーションの例を示します。

図 21-20: メッセージビット送信中のバス衝突



### 21.6.8 STOP 条件期間中のバス衝突

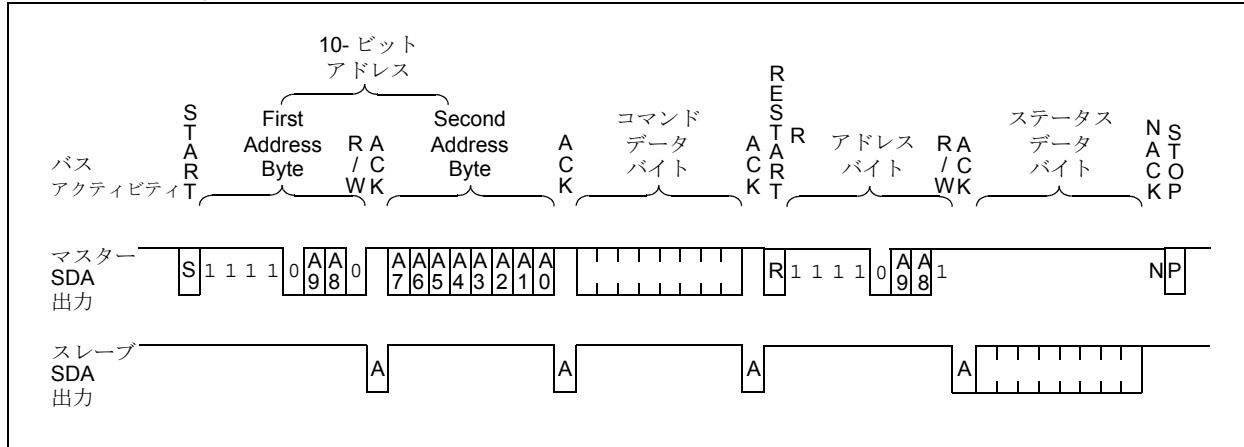
マスターソフトウェアが I<sup>2</sup>C バスの状態追跡を失った場合、STOP 条件中にバス衝突を起こす条件となります。この場合、STOP 条件を生成したマスターがアビトトレーションを失い、バス衝突割り込みを生成します。

## 21.7 スレーブとして通信

複数のプロセッサが相互に通信を行っているようなシステムでは、dsPIC30F はスレーブとして通信を行うことがあります（図 21-21 参照）。モジュールが有効化されると、スレーブモジュールがアクティブになります。スレーブはメッセージを開始することを許可されていないため、マスターが開始したメッセージシーケンスに応答するのみです。マスターは I<sup>2</sup>C プロトコル内でデバイスアドレスバイトにより定義された特定のスレーブから応答を要求します。スレーブモジュールはプロトコルにより定義された適切な時間にマスターに応答します。

マスター モジュールでは、転送プロトコルのシーケンス組み立てはソフトウェアのタスクです。ただし、デバイス アドレスがソフトウェアによって指定されたスレーブ用のアドレスと一致した場合にこれを検出するのはスレーブモードの役割です。

図 21-21: 一般的なスレーブ I<sup>2</sup>C メッセージ: マルチプロセッサコマンド/ステータス



**START** 条件の後、スレーブモジュールはデバイスアドレスを受信、チェックします。スレーブは 7 ビットアドレスまたは 10 ビットアドレスで区別されています。デバイスアドレスが一致した場合、モジュールはソフトウェアにそのデバイスが選択された旨を通知するための割り込みを生成します。マスターより送信された **R/W** ビットに基づき、スレーブはデータを受信または送信します。スレーブがデータを受信した場合、スレーブモジュールは自動的に応答(ACK)を生成し、I2CCSR にある受信済みデータを I2CRCV に転送してから、割り込みによってソフトウェアにこれを通知します。スレーブがデータを送信する場合、ソフトウェアは I2CTRN レジスタにデータをロードする必要があります。

### 21.7.1 受信データのサンプリング

すべての受信ビットはクロック (**SCL**) 線の立ち上がりエッジでサンプルされます。

### 21.7.2 START 条件と STOP 条件の検出

スレーブモジュールは START 条件と STOP 条件をバス上で検出し、S ビット (I2CSTAT<3>) と P ビット (I2CSTAT<4>) の状態を示します。START (S) と STOP (P) ビットは RESET が発生するか、モジュールが無効化されるとクリアされます。START または Repeated START イベント検出後、S ビットがセットされ P ビットがクリアされます。STOP イベント検出後、P ビットがセットされ S ビットがクリアされます。

### 21.7.3 アドレスの検出

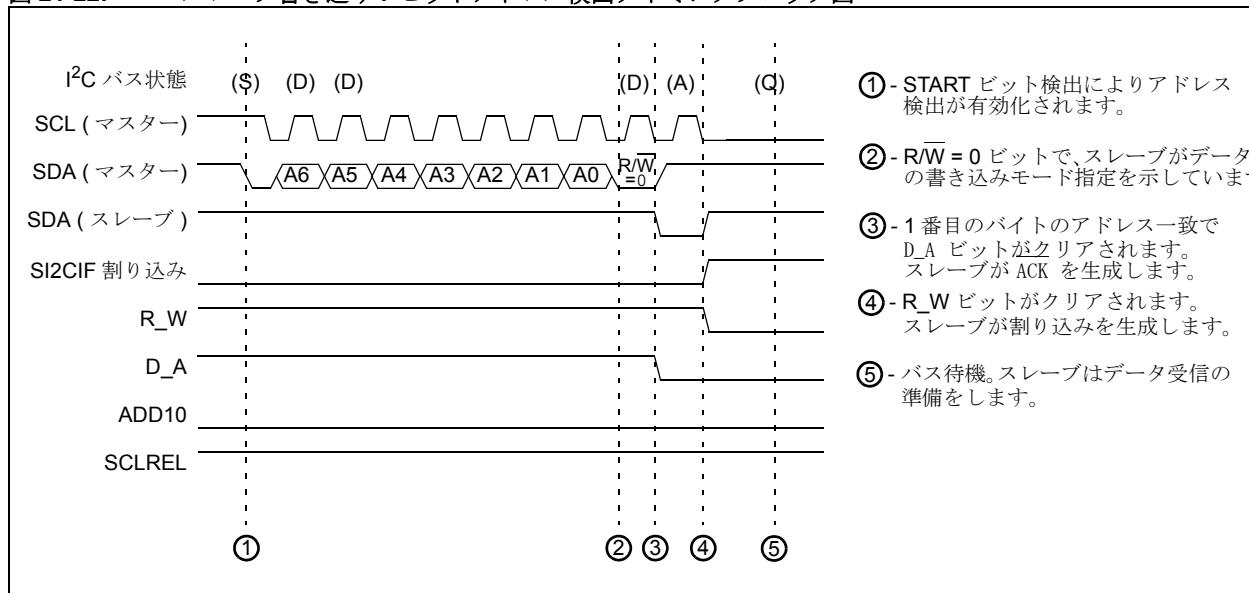
モジュールが有効化されると、スレーブモジュールは START 条件が発生するまで待機します。START 条件後、A10M ビット (I2CCON<10>) によって、スレーブは 7 ビットまたは 10 ビットアドレスの検出を試みます。スレーブモジュールは 7 ビットアドレスのときは 1 つの受信バイトと比較し、10 ビットアドレスのときは 2 つの受信バイトと比較します。7 ビットアドレスの中には、後のデータ転送の方向を指定する R/W ビットを持っています。R/W = 0 の場合、書き込みモードが指定され、スレーブはマスターからデータを受信します。R/W = 1 の場合、読み込みモードが指定され、スレーブはマスターにデータを送信します。10 ビットアドレスは R/W ビットを含みますが、スレーブが 10 ビットアドレスの 2 番目のバイトを受信しなくてはならないため、常に R/W = 0 になっています。

#### 21.7.3.1 7 ビットアドレスとスレーブ書き込みモード

START 条件に続いて、モジュールは 8 ビットを I2CRSR レジスタにシフトします（図 21-22 参照）。レジスタ I2CRSR<7:1> の値は I2CADD<6:0> レジスタの値と比較されます。デバイスアドレスは 8 番目のクロック (SCL) の立下りエッジで比較されます。アドレスが一致すると以下のイベントが発生します。

1. ACK が生成されます。
2. D\_A および R\_W ビットがクリアされます。
3. モジュールが 9 番目の SCL クロックの立下りエッジで SI2CIF 割り込みを生成します。
4. モジュールはマスターのデータ送信を待機します。

図 21-22: スレーブ書き込み 7 ビットアドレス検出タイミングブロック図



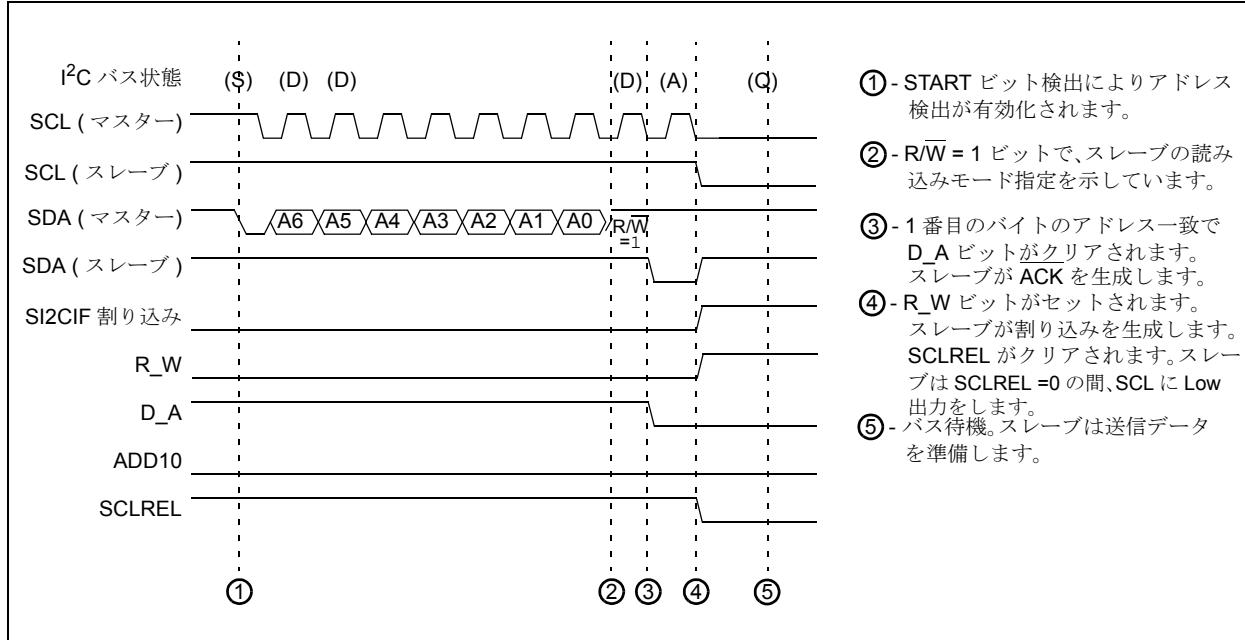
## 21.7.3.2 7 ビットアドレスとスレーブ読み込みモード

7 ビットバイトアドレスで  $R/W = 1$  となりスレーブ読み込みが指定されている場合、デバイスアドレスの検出プロセスはスレーブ書き込みと似ています（図 21-23 参照）。アドレスが一致すると以下のイベントが発生します。

1.  $\overline{ACK}$  が生成されます。
  2.  $D_A$  ビットがクリアされ、 $R_W$  ビットがセットされます。
  3. モジュールが 9 番目の SCL クロックの立下りエッジで SI2CIF 割り込みを生成します。
- スレーブモジュールはこの時点ですべてのデータを返信するよう要求されるため、I<sup>2</sup>C バスの動作を中止しソフトウェアが応答の準備をできるようにする必要があります。この動作はモジュールが SCLREL ビットをクリアすることで自動的に実行されます。SCLREL が Low の場合、スレーブモジュールは SCL クロック線を Low にして、I<sup>2</sup>C バス上で待機状態を発生させます。スレーブモジュールと I<sup>2</sup>C バスはソフトウェアが I2CTRN レジスタに応答データを書き込むまでこの状態を保ちます。

注： スレーブ読み込みアドレスが検出されると、STREN ビットの状態に関わらず、SCLREL は自動的にクリアされます。

図 21-23: スレーブ読み込み 7 ビットアドレス検出タイミングプロック図



### 21.7.3.3 10 ビットアドレス

10 ビットアドレスモードでは、スレーブは 2 つのデバイスアドレスバイトを受信する必要があります (図 21-24 参照)。1 番目のアドレスバイトの上位 5 ビット (MSbs) が 10 ビットアドレスモードであることを指定します。アドレスの R/W ビットは書き込みモードとなっており、これにより、スレーブデバイスが 2 番目のアドレスバイトを受信します。10 ビットアドレスでは、1 番目のバイトは ‘11110 A9 A8 0’ と等しくなり、A9 と A8 はアドレスの上位 2 ビットとなります。

START 条件に続いて、モジュールは 8 ビットを I2CRCSR レジスタにシフトします。レジスタ I2CRCSR<2:1> の値は I2CADD<9:8> レジスタの値と比較されます。I2CRCSR<7:3> の値は ‘11110’ と比較されます。デバイスアドレスは 8 番目のクロック (SCL) の立下りエッジで比較されます。アドレスが一致すると以下のイベントが発生します。

1.  $\overline{\text{ACK}}$  が生成されます。
2. D\_A および R\_W ビットがクリアされます。
3. モジュールが 9 番目の SCL クロックの立下りエッジで SI2CIF 割り込みを生成します。

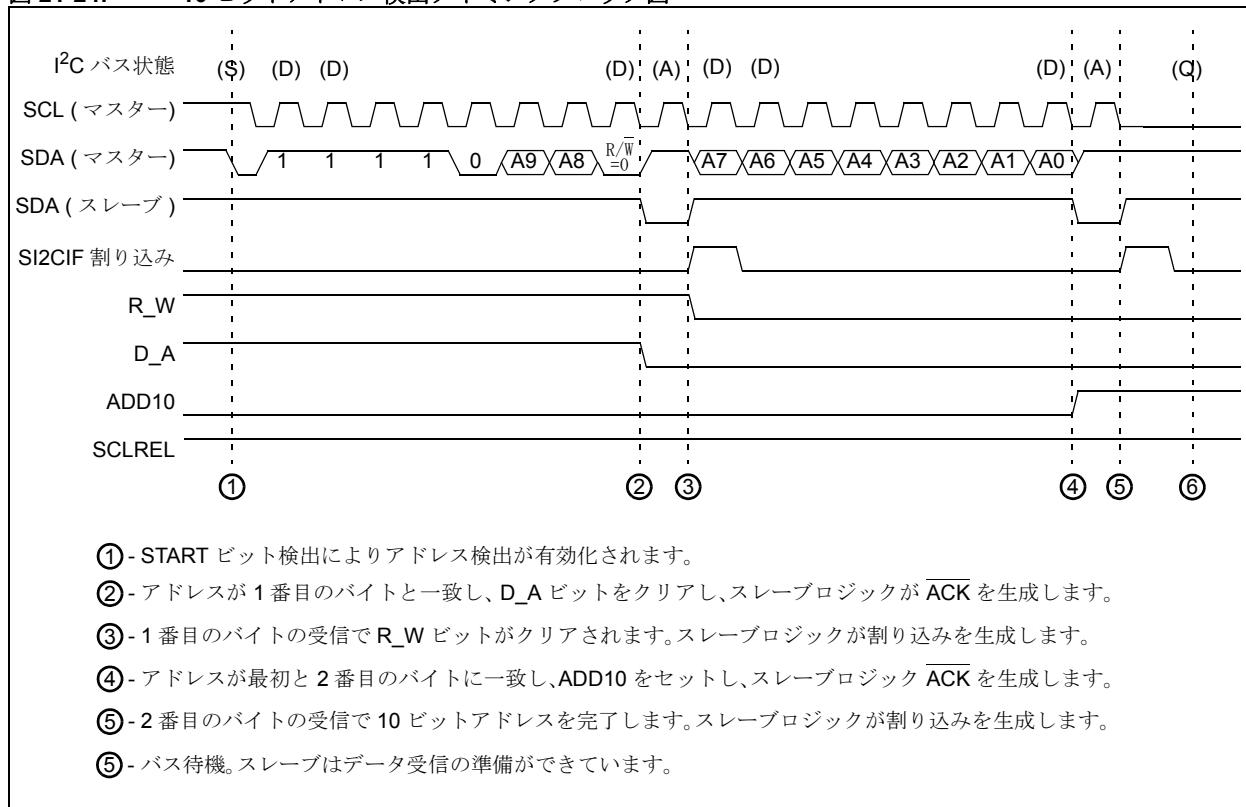
モジュールは 10 ビットアドレスの 1 番目のバイトを受信した後に割り込みを生成しますが、この生成はほとんど使用されません。

モジュールは 2 番目のバイトを I2CRCSR に受信し続けます。この時、I2CRCSR<7:0> が I2CADD<7:0> と比較されます。アドレスが一致すると以下のイベントが発生します。

1.  $\overline{\text{ACK}}$  が生成されます。
2. ADD10 ビットがセットされます。
3. モジュールが 9 番目の SCL クロックの立下りエッジで SI2CIF 割り込みを生成します。
4. モジュールはマスターからのデータ送信か、Repeated START 条件を待ちます。

注： 10 ビットモードでの Repeated START 条件の後、スレーブモジュールは 1 番目の 7 ビットアドレス ‘11110 A9 A8 0’ の一致条件のみとします。

図 21-24: 10 ビットアドレス検出タイミングブロック図



## 21.7.3.4 一斉呼び出し動作

I<sup>2</sup>C バスのアドレス手順とは、通常、START 条件の後に 1 番目のバイトを使用し、どのマスターがどのスレーブデバイスをアドレスするかを決めることです。ただし、一斉呼び出しアドレスは例外で、すべてのデバイスをアドレスできます。このアドレスが使用されると、すべての有効なデバイスは応答を返す必要があります。一斉呼び出しアドレスは特定の目的のために I<sup>2</sup>C プロトコルにより予約されている 8 つのアドレスのうちの 1 つです。R/W = 0 となる ‘0’ のみで構成されています。一斉呼び出しは常にスレーブ書き込みモードとなります。

一斉呼び出しアドレスはビット GCEN (I2CCON<7>) をセットして一斉呼び出しを有効化した場合に認識されます (図 21-25 参照)。START ビット検出後、8 ビットが I2CRSR にシフトされ、アドレスは I2CADD と比較されます。また、一斉呼び出しアドレスとも比較されます。

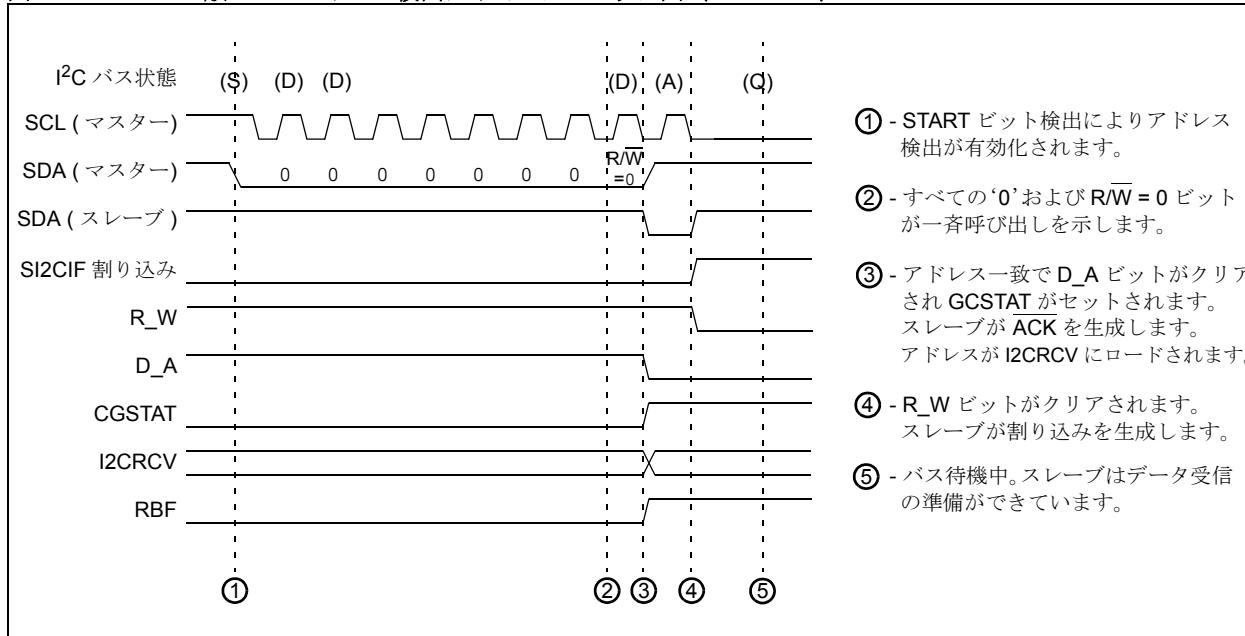
一斉呼び出しアドレスと一致すると以下のイベントが発生します。

1.  $\overline{\text{ACK}}$  が生成されます。
2. スレーブモジュールが GCSTAT ビット (I2CSTAT<9>) をセットします。
3. D\_A および R\_W ビットがクリアされます。
4. モジュールが 9 番目の SCL クロックの立下りエッジで SI2CIF 割り込みを生成します。
5. I2CRSR が I2CRCV に転送され、RBF フラグビットがセットされます (8 番目のビット処理中)。
6. モジュールはマスターのデータ送信を待機します。

割り込みが実行されると、GCSTAT ビットにより割り込みの原因がチェックされ、デバイスアドレスがデバイス特有か一斉呼び出しアドレスかが確認されます。

一斉呼び出しアドレスは 7 ビットアドレスであることに注意して下さい。A10M ビットがセットされている場合でも、10 ビットアドレスのスレーブモジュール構成ですが、GCEN がセットされた場合、スレーブモジュールは 7 ビット一斉呼び出しアドレス検出を続行します。

図 21-25: 一般コールアドレス検出タイミングブロック図 (GCEN = 1)

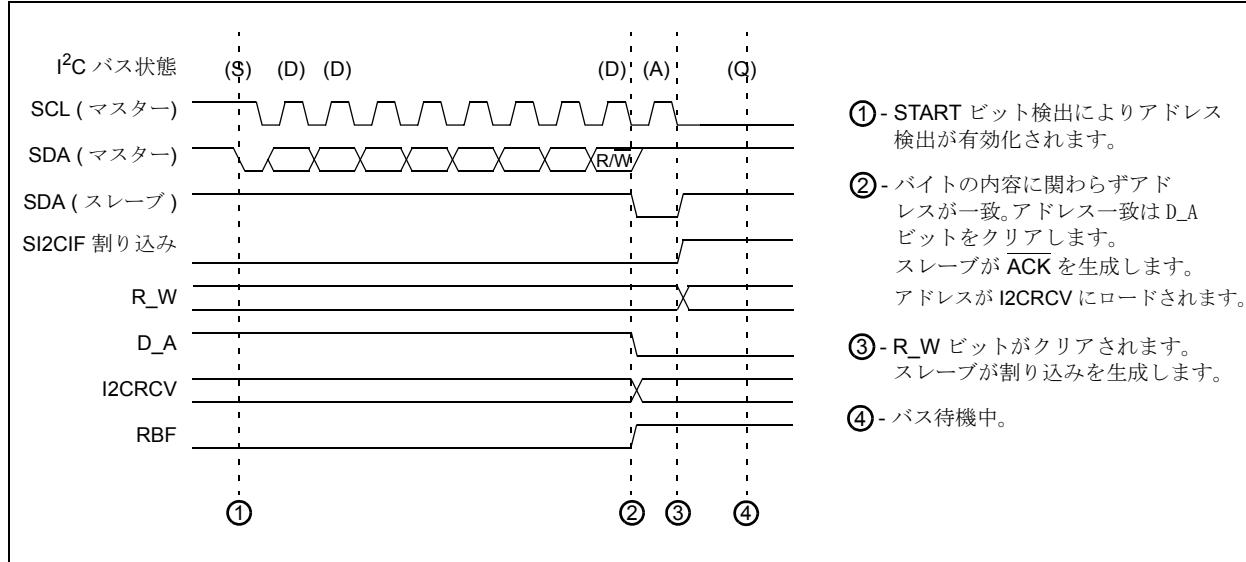


### 21.7.3.5 すべてのアドレスの受信 (IPMI 動作)

ある種の I<sup>2</sup>C システムプロトコルではスレーブがバス上のすべてのメッセージに対して動作する必要があります。例えば、IPMI (Intelligent Peripheral Management Interface) バスは I<sup>2</sup>C ノードを分散ネットワークのメッセージ中継器として使用します。ノードがすべてのメッセージを中継できるようにするには、スレーブモードはデバイスアドレスに関わらずすべてのメッセージを受け入れる必要があります。

IPMIEN ビット (I2CCON<11>) を設定するとこのモードを使用できます (図 21-26 参照)。I2CADD レジスタ、A10M、GCEN ビットの状態に関わらず、すべてのアドレスが受け入れられます。

図 21-26: アドレス検出タイミングプロック図 (IPMIEN = 1)



### 21.7.3.6 アドレスが無効のとき

7 ビットアドレスが I2CADD<6:0> の内容と一致しない場合、スレーブモジュールは IDLE 状態に戻り、STOP 条件が発生するまでバスアクティビティをすべて無視します。

10 ビットアドレスの 1 番目のバイトが I2CADD<9:8> の内容と一致しない場合、スレーブモジュールは IDLE 状態に戻り、STOP 条件が発生するまでバスアクティビティをすべて無視します。

10 ビットアドレスの 1 番目のバイトが I2CADD<9:8> の内容と一致しても、2 番目のバイトが I2CADD<7:0> と一致しない場合、スレーブモジュールは IDLE 状態に戻り、STOP 条件が発生するまでバスアクティビティをすべて無視します。

### 21.7.4 マスターデバイスからのデータ受信

デバイスアドレスバイトの R<sub>W</sub> ビットが 0 でアドレス一致が発生している場合、R<sub>W</sub> ビット (I2CSTAT<2>) はクリアされます。スレーブモジュールはマスターが送信するデータを待機する状態に入ります。デバイスアドレスバイトの後の、データの内容はシステムプロトコルにより特定されたスレーブモジュールでのみ受信されます。

スレーブモジュールは 8 ビットを I2CRSR レジスタにシフトします。8 番目の SCL クロックの立下りエッジ後、以下のイベントが発生します。

- モジュールは ACK または NACK 生成を開始します。
- 受信データありを表示するために RBF ビットがセットされます。
- I2CRSR バイトがソフトウェアでアクセスできるよう I2CRCV レジスタに転送されます。
- D\_A ビットがセットされます。
- スレーブ割り込みが生成されます。ソフトウェアは I2CSTAT レジスタの状態をチェックしてイベントの原因を特定し SI2CIF フラグをクリアします。
- モジュールは次のデータバイトの受信を待機します。

#### 21.7.4.1 応答生成

通常、スレーブモードは9番目のSCLクロックにACKを送信してすべての受信バイトに応答します。受信バッファがオーバーランを起こすと、スレーブモジュールはこのACKを生成しなくなります。次の状態（両方の場合もあり）でオーバーランが発生します。

1. バッファ満杯ビットRBF (I2CSTAT<1>) が転送受信前にセットされた。
2. オーバーフロービットI2COV (I2CSTAT<6>) が転送受信前にセットされた。

表21-3はデータ転送バイトが受信された場合にRBFビットとI2COVビットで発生するアクションです。スレーブモジュールがI2CRCVへの転送を試みる時点でRBFビットがすでにセットされていると、転送は発生しませんが割り込みが発生しI2COVビットがセットされます。RBFビットとI2COVビットの両方がセットされた場合でも、スレーブモジュールは同様に動作します。斜線のかかったセルはソフトウェアがオーバーフローを適切にクリアできなかった場合の状態を示しています。

I2CRCV読み込みによりRBFビットをクリアします。I2COVはソフトウェアを介して‘0’を書き込むとクリアされます。

表21-3: データ転送受信バイトのアクション

データバイト受信のステータスピット		転送 I2CRSR → I2CRCV	生成 ACK	SI2CIF 割り込み生成 (有効化されると割り込みが発生)	セット RBF	セット I2COV
RBF	I2COV					
0	0	はい	はい	はい	はい	変化なし
1	0	いいえ	いいえ	はい	変化なし	はい
1	1	いいえ	いいえ	はい	変化なし	はい
0	1	はい	いいえ	はい	はい	変化なし

注： 斜線のかかったセルはソフトウェアがオーバーフロー条件を適切にクリアできなかった場合の状態を示しています。

#### 21.7.4.2 スレーブ受信中のWAITステータス

スレーブモジュールがデータバイトを受信すると、マスターはすぐに次のバイト送信を開始できます。しかしこれは、スレーブのソフトウェアが9クロック周期内に、受信したデータを処理できるときでないと可能になりません。時間が不足した場合は、スレーブソフトウェアはバスWAIT期間の延長をすることができます。

STRENビット(I2CCON<6>)を使用すると、バスWAITをスレーブ受信時に発生させることができます。受信したバイトの9番目のSCLクロック周期の立下りエッジのときSTREN=1の場合、スレーブモジュールはSCLRELビットをクリアします。SCLREビットをクリアすると、スレーブモジュールがSCL線をLowにし、WAITを開始します。マスターとスレーブのSCLクロックの同期については、セクション21.6.2「マスタークロック同期」を参照してください。

ソフトウェアで受信の再開が可能になると、ソフトウェアはSCLRELをセットします。これにより、スレーブモジュールはSCL線を解放しマスターはクロッキングを再開します。

#### 21.7.4.3 スレーブ受信のメッセージ例

スレーブメッセージの受信はほぼ自動化されたプロセスです。ソフトウェアはスレーブプロトコルをスレーブ割り込みを使用して処理し、イベントと同期させます。

スレーブが有効なアドレスを検出すると、関連する割り込みがソフトウェアにメッセージが来ることを通知します。データ受信後、各データバイトは I2CRCV レジスタに転送され、割り込みでバッファを読み出すようソフトウェアに通知します。

図 21-27 は簡単な受信メッセージの例を示しています。7 ビットアドレスメッセージでは、アドレスバイトに対して発生する割り込みは 1 つのみです。その後、4 つのデータバイトのそれぞれに対し割り込みが発生します。

割り込みにより、ソフトウェアは RBF ビット、D\_A ビットと R\_W ビットをモニターして受信バイトの状態を確認します。

図 21-28 は 10 ビットアドレスを使用した同様のメッセージを示しています。このケースでは、アドレスに対し 2 つのバイトが必要です。

図 21-29 はソフトウェアが受信バイトに対応せず、バッファオーバーランが発生している例を示しています。2 番目のバイトの受信後、モジュールは自動的にマスター送信に対して NACK (応答せず) となります。一般的に、これによりマスターは以前のバイトを再送します。I2C0V ビットはバッファがオーバーランしていることを示します。I2CRCV バッファは 1 番目のバイトの内容を保持しています。3 番目のバイト受信後、バッファはまだ満杯でモジュールは再びマスターを NACK します。最後に、ソフトウェアはバッファを読み込みます。バッファの読み込みにより RBF ビットはクリアされますが、I2C0V ビットはセットされたままになります。ソフトウェアは I2C0V ビットをクリアする必要があります。次に受信されたバイトは I2CRCV バッファに移動し、モジュールは ACK でこれに応答します。

図 21-30 はデータ受信中のクロック延長をハイライトしています。その前の例では、STREN = 0 として、メッセージ受信中のクロック延長は無効になっていた点に注意してください。この例では、ソフトウェアは STREN をセットしクロック延長を有効化しています。STREN = 1 の場合、モジュールは各データバイト受信ごとに自動的にクロックを延長し、ソフトウェアがデータをバッファから移動する時間を確保します。9 番目のクロックの立下りエッジが RBF = 1 の場合は、モジュールは自動的に SCLREL ビットをクリアし SCL バス線を Low にします。2 番目の受信データバイトで示されている通り、9 番目のクロックの立下りエッジの前にソフトウェアがバッファを読み込み RBF をクリアにできる場合、クロック延長は発生しません。ソフトウェアはバスをいつでも中止できます。SCLREL ビットをクリアにすると、バスの SCL Low を検出した後、モジュールは SCL 線を Low にします。トランザクションが中止され、SCLREL ビットがセットされるまで SCL 線は Low のままであります。

図 21-27: スレーブメッセージ (スレーブヘデータ書き込み: 7 ビットアドレス、アドレス一致、A10M = 0、GCEN = 0、IPMIEN = 0)

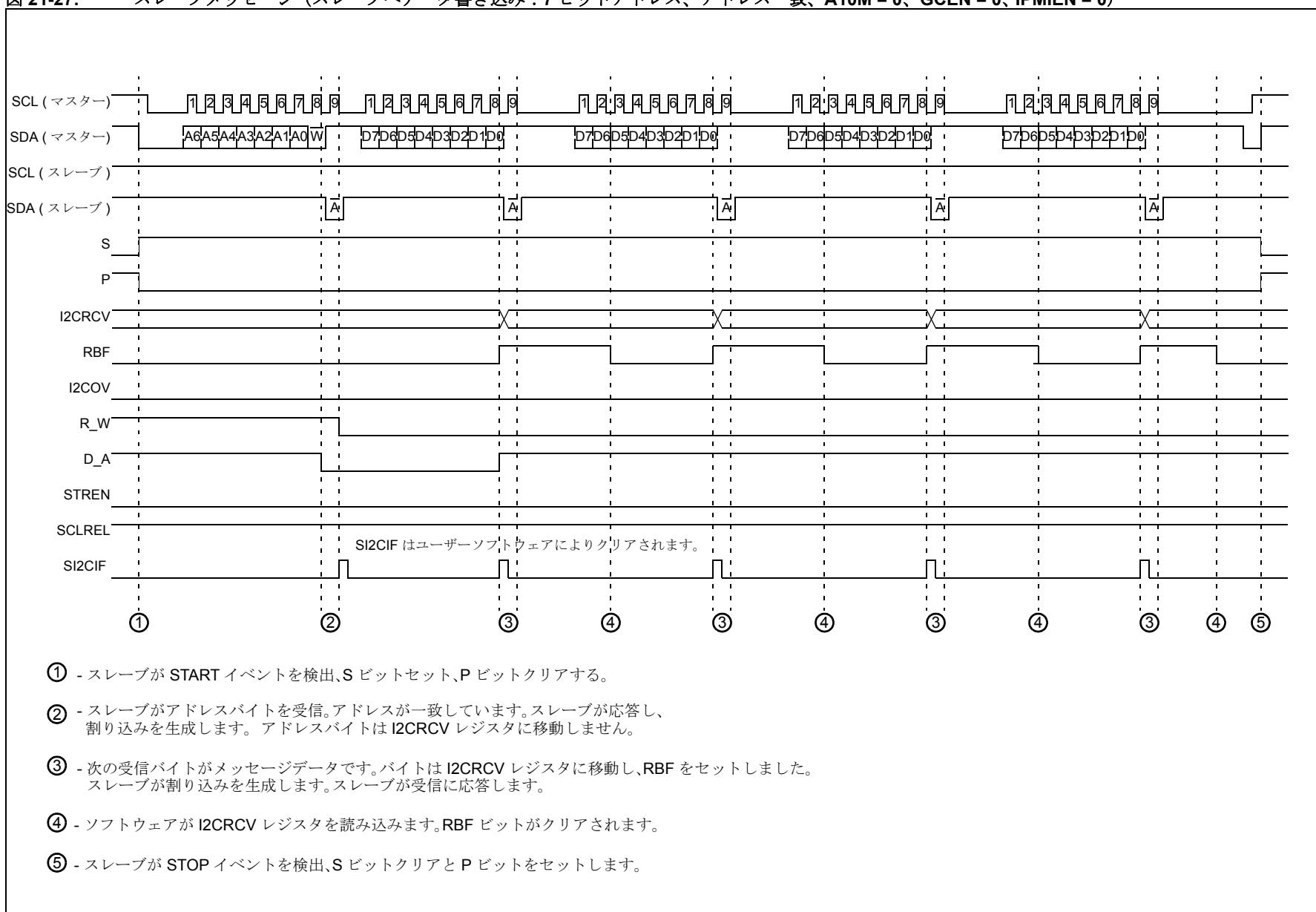


図 21-28: スレーブメッセージ (スレーブヘデータ書き込み: 10 ビットアドレス、アドレス一致、A10M = 1、GCEN = 0、IPMIEN = 0)

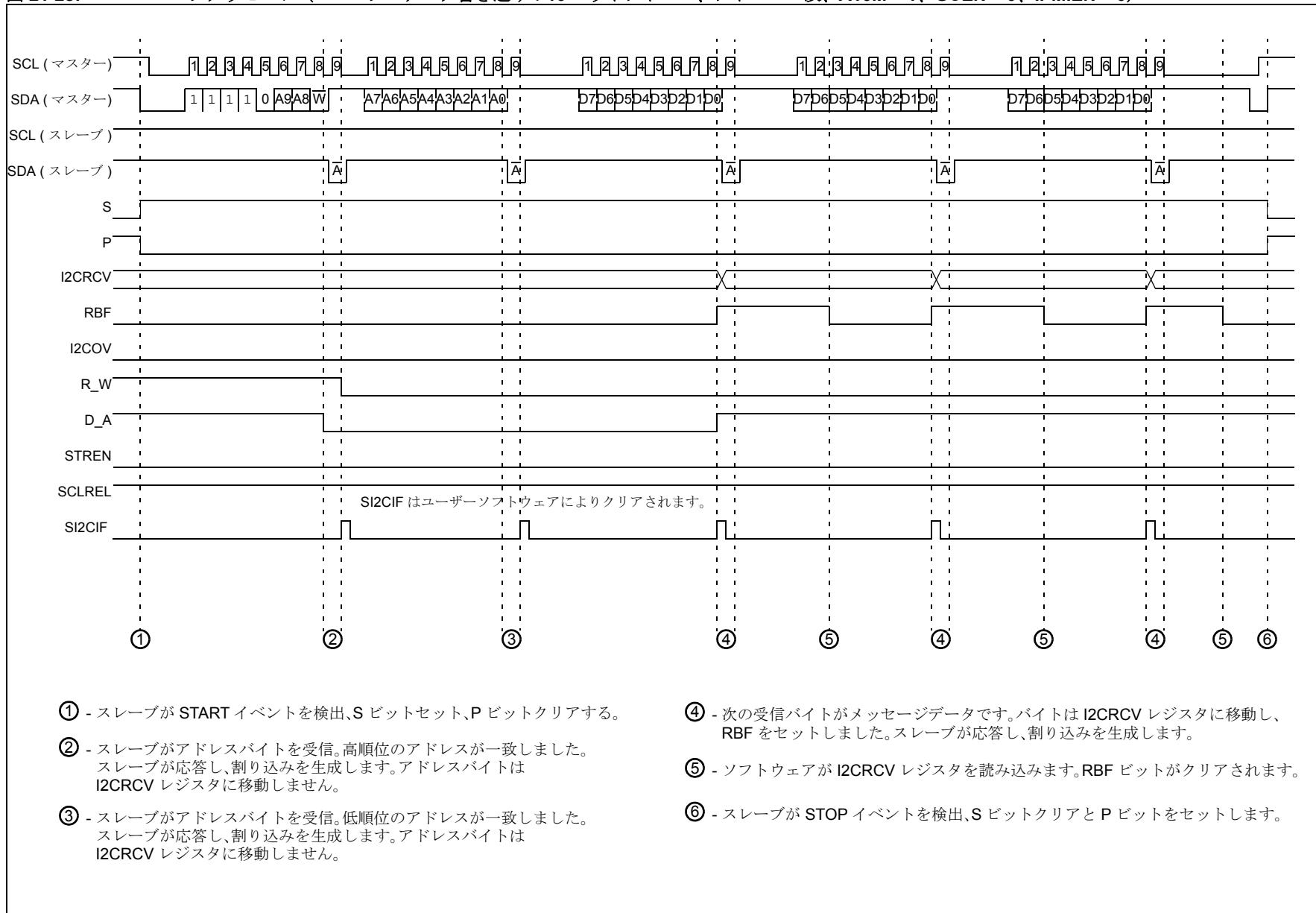


図 21-29: スレーブメッセージ (スレーブヘデータ書き込み: 7 ビットアドレス、バッファオーバーラン、A10M = 0、GCEN = 0、IPMIEN = 0)

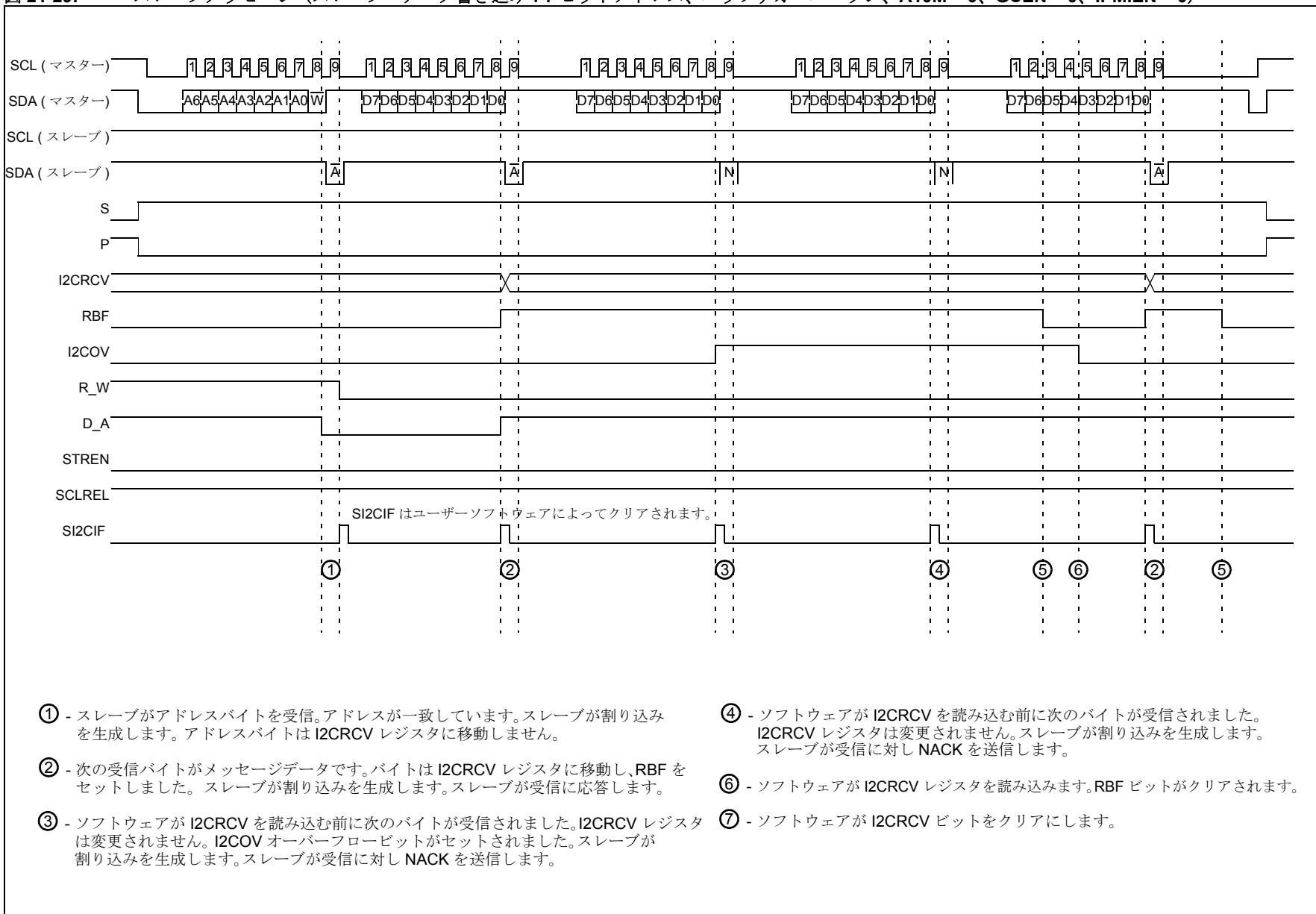
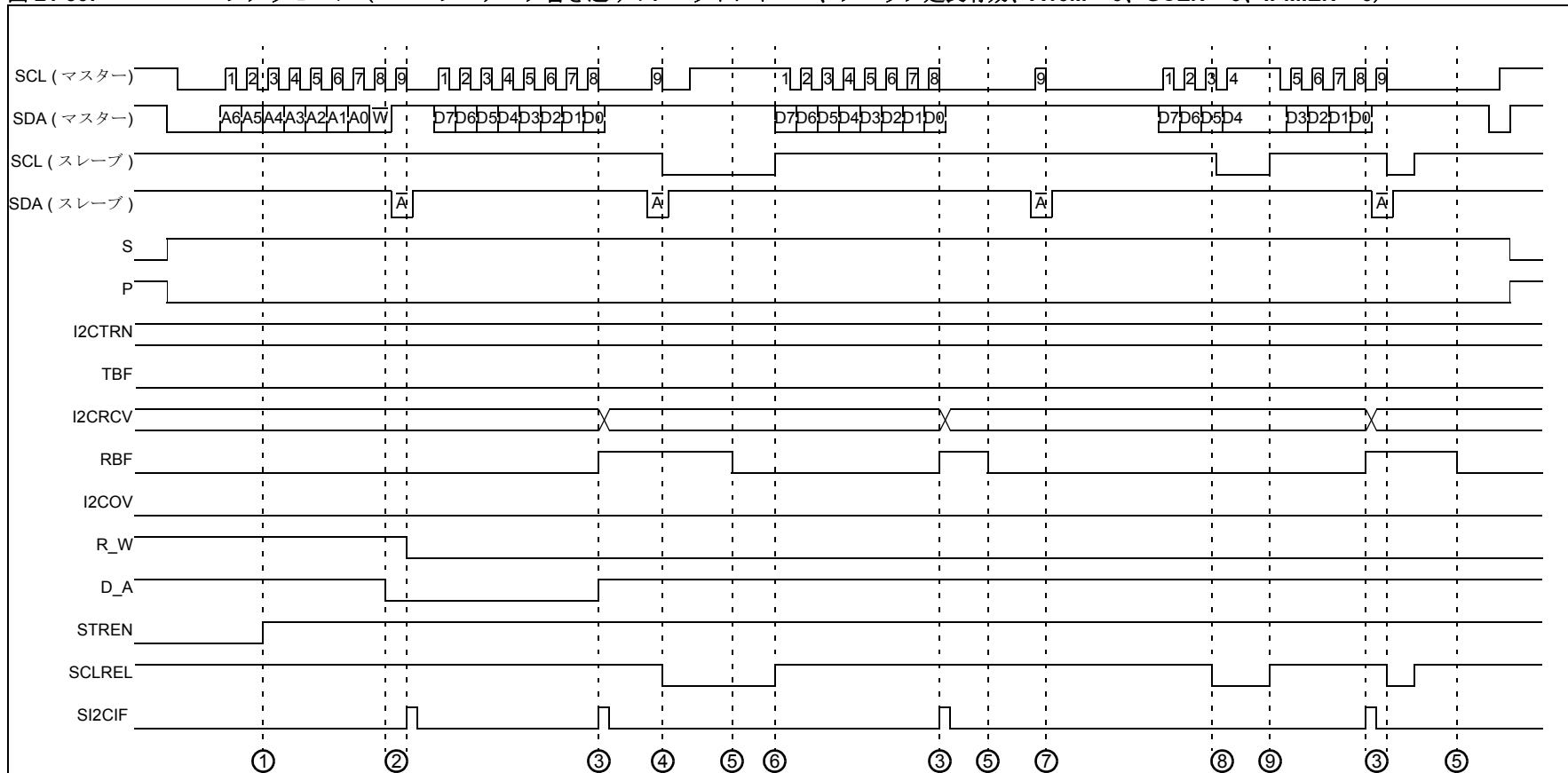


図 21-30: スレーブメッセージ (スレーブヘデータ書き込み : 7 ビットアドレス、クロック延長有効、A10M = 0、GCEN = 0、IPMIEN = 0)



- ① - ソフトウェアは STREN ビットをセットしクロック延長を有効化します。
- ② - スレーブがアドレスバイトを受信。
- ③ - 次の受信バイトがメッセージデータです。バイトは I2CRCV レジスタに移動し、RBF をセットしました。
- ④ - 9番目のクロックで RBF = 1 のため、自動クロック延長が開始されました。スレーブは SCLREL ビットをクリアします。スレーブは SCL 線を Low にしクロックを延長します。
- ⑤ - ソフトウェアが I2CRCV レジスタを読み込みます。RBF ビットがクリアされます。
- ⑥ - ソフトウェアは SCLREL ビットをセットしクロックを解除します。
- ⑦ - この時 RBF = 0 のため、スレーブは SCLREL をクリアしません。
- ⑧ - ソフトウェアは SCLREL をクリアしクロックのホールドを発生させます。モジュールは SCL Low の出力の前に SCL Low を検出する必要があります。
- ⑨ - ソフトウェアは SCLREL をセットしクロックの解放ができます。

## 21.7.5 マスターデバイスへのデータ送信

受信デバイスバイトの  $\overline{R/W}$  ビットが ‘1’ でアドレス一致が発生している場合、 $R_W$  ビット ( $I2CSTAT<2>$ ) はセットされます。この時点では、マスターデバイスはスレーブがデータのバイトを送信して応答することを求めます。バイトの内容はシステムプロトコルにより定義されスレーブモジュールだけが送信できます。

アドレス検出の割り込みが発生した場合、ソフトウェアはバイトを  $I2CTRN$  レジスタに書き込みデータ送信を開始できます。

スレーブモジュールが  $TBF$  ビットをセットします。8 データビットが  $SCL$  入力の立下りエッジでシフト出力されます。これにより、 $SCL$  High 時間に  $SDA$  信号が有効になります。すべての 8 ビットがシフト出力完了すると、 $TBF$  ビットはクリアされます。

スレーブモジュールが 9 番目の  $SCL$  クロックの立上りエッジでマスター受信機から応答を検出します。

$SDA$  線が  $Low$  で応答 ( $ACK$ ) を示している場合、マスターはさらにデータを要求していく、メッセージは完了していません。モジュールがスレーブ割り込みを生成し、さらにデータを要求されていることを通知します。

スレーブ割り込みが 9 番目の  $SCL$  クロックの立下りエッジで生成されます。ソフトウェアは  $I2CSTAT$  レジスタのステータスをチェックし、 $SI2CIF$  フラグをクリアにします。

$SDA$  線が  $High$  の場合、非応答 ( $NACK$ ) を示しデータ転送が完了します。スレーブモジュールがリセットされ割り込みを生成しません。スレーブモジュールは次の  $START$  ビット検出を待機します。

### 21.7.5.1 スレーブ送信中の WAIT ステータス

スレーブがメッセージを送信する間、マスターは  $R/W = 1$  のアドレスで指定されたスレーブから、直ぐ返信されることを期待しています。従って、スレーブモジュールはスレーブがデータを返信するたびに自動的にバス  $WAIT$  を生成する必要があります。

自動  $WAIT$  は有効なデバイスアドレスの 9 番目の  $SCL$  クロックの立下りエッジまたはマスターからのさらなるデータの送信を要求する応答ビットのときに行われます。

スレーブモジュールが  $SCLREL$  ビットをクリアにします。 $SCLRE$  ビットをクリアすると、スレーブモジュールが  $SCL$  線を  $Low$  にし、 $WAIT$  を開始します。マスターとスレーブの  $SCL$  クロックの同期については、セクション 21.6.2 「マスタークロック同期」を参照してください。

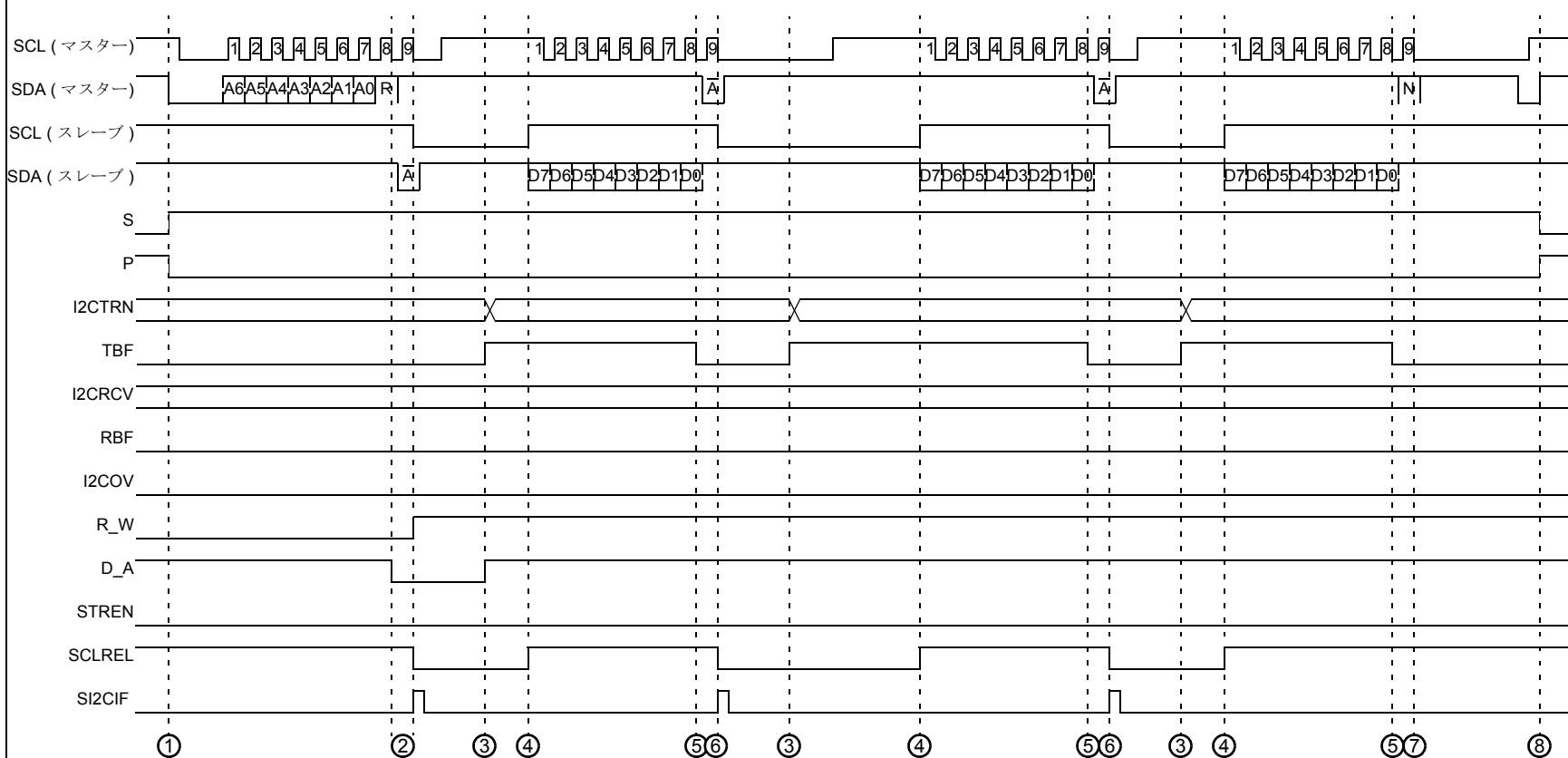
ソフトウェアが  $I2CTRN$  に送信データをロードし送信の再開が可能になると、ソフトウェアは  $SCLREL$  をセットします。これにより、スレーブモジュールは  $SCL$  線を解放しマスターはクロッキングを再開します。

### 21.7.5.2 スレーブ送信のメッセージ例

7 ビットアドレスメッセージのときのスレーブ送信は図 21-31 で示すとおりです。アドレスが一致しそのアドレスの  $R/W$  ビットがスレーブ送信を示している場合、モジュールは  $SCLREL$  ビットをクリアして自動的にクロック延長を開始し、応答バイトが必要なことを示す割り込みを生成します。ソフトウェアは応答バイトを  $I2CTRN$  レジスタに書き込みます。送信が完了すると、マスターは応答を返します。マスターが  $ACK$  で返信すると、そのマスターは追加のデータを要求し、モジュールは再度  $SCLREL$  ビットをクリアして他の割り込みを生成します。マスターが  $NACK$  で応答すると、追加のデータ要求はなくなり、モジュールはクロック延長や割り込みを行いません。

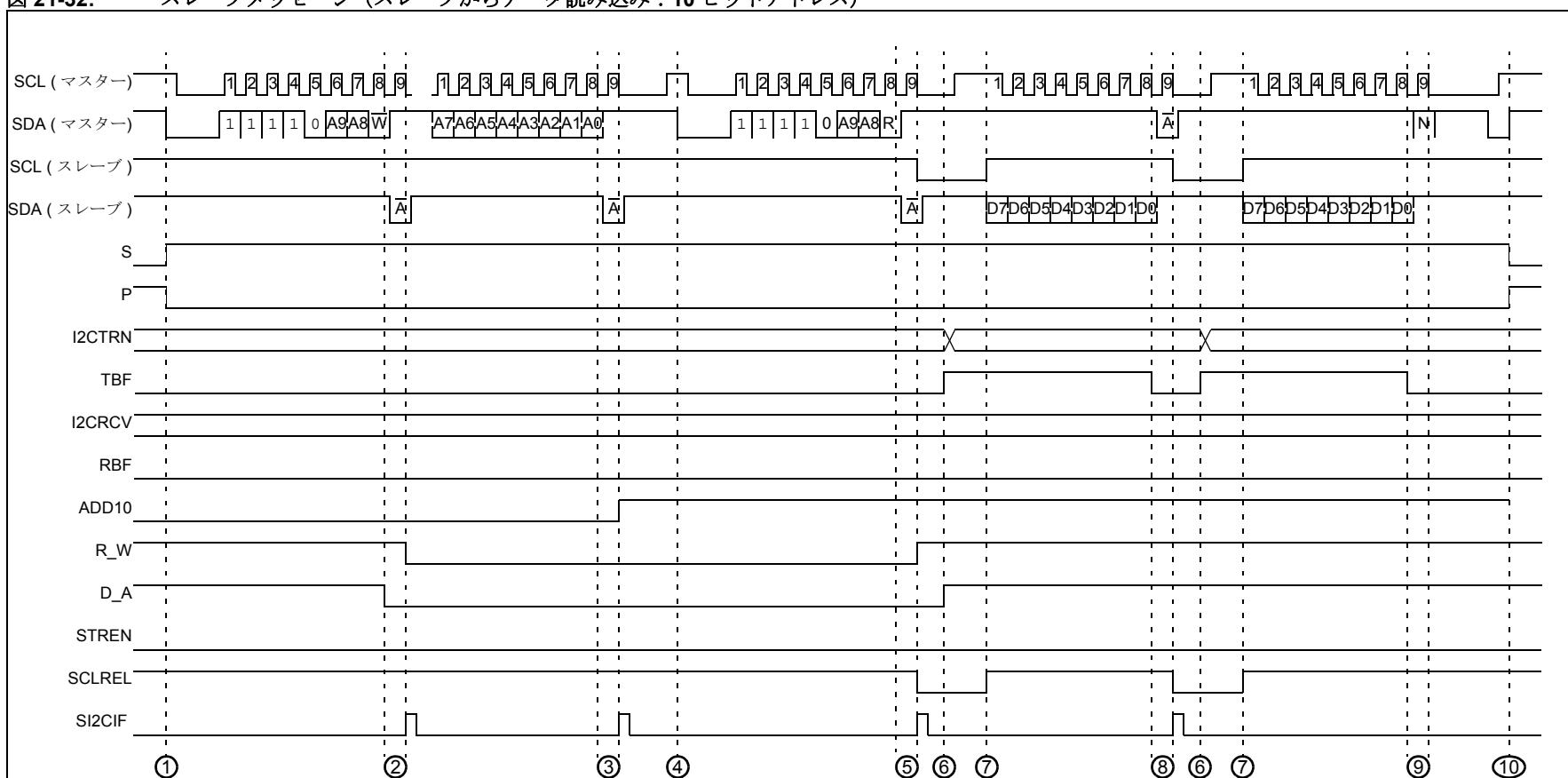
10 ビットアドレスメッセージのときのスレーブ送信では、スレーブは最初に 10 ビットアドレスを認識する必要があります。マスターはアドレスに 2 バイトを送信する必要があるため、アドレスの 1 番目のバイトの  $R/W$  ビットは書き込みを指定します。メッセージを読み込むために、マスターは  $Repeated START$  を送信し、 $R/W$  ビットで読み込みを指定してアドレスの 1 番目のバイトを繰り返します。この時点で、図 21-32 で示す通りスレーブ送信が開始されます。

図 21-31: スレーブメッセージ (スレーブからデータ読み込み : 7 ビットアドレス)



- ① - スレーブが START イベントを検出し、S ビットをセット、P ビットをクリアします。
- ② - スレーブがアドレスバイトを受信。アドレスが一致しています。スレーブが割り込みを生成します。アドレスバイトは I2CRCV レジスタに移動しません。R\_W = 1 でスレーブからの読み込みを示します。SCLREL = 0 でマスタークロックを延長します。
- ③ - ソフトウェアは I2CTRN に応答データを書き込みます。TBF = 1 はバッファが満杯であることを示しています。I2CTRN 書き込みで D\_A がセットされ、データバイトを示します。
- ④ - ソフトウェアは SCLREL ビットをセットしクロックのホールドを解除します。マスターはクロッキングを再開しスレーブはデータバイトを送信します。
- ⑤ - 最後のビットの後、モジュールは TBF ビットをクリアしバッファが次のバイトのために使用可能であることを示します。
- ⑥ - 9番目のクロックの終わりにマスターが ACK を送信した場合、モジュールは SCLREL をクリアし、クロックを中止します。スレーブが割り込みを生成します。
- ⑦ - 9番目のクロックの終わりにマスターが NACK を送信した場合、それ以上のデータ要求はありません。モジュールはクロック中止や割り込み生成を行いません。
- ⑧ - スレーブが STOP イベントを検出して S ビットクリア、P ビットをセットします。

図 21-32: スレーブメッセージ (スレーブからデータ読み込み : 10 ビットアドレス)



- ① - スレーブが START イベントを検出し、S ビットをセット、P ビットをクリアします。
- ② - スレーブが 1 番目のアドレスバイトを受信。書き込みが示されます。スレーブが応答し、割り込みを生成します。
- ③ - スレーブがアドレスバイトを受信。アドレスが一致しています。スレーブが応答し、割り込みを生成します。
- ④ - マスターが Repeated START を送信し、メッセージを再送します。
- ⑤ - スレーブが 1 番目のアドレスバイトを再受信。読み込みが示されます。スレーブがクロックを延長します。
- ⑥ - ソフトウェアは I2CTRN に応答データを書き込みます。

- ⑦ - ソフトウェアは SCLREL ビットをセットしクロックのホールドを解除します。マスターはクロッキングを再開しスレーブはデータバイトを送信します。
- ⑧ - 9 番目のクロックの終わりにマスターが ACK を送信した場合、モジュールは SCLREL をクリアし、クロックを中止します。スレーブが割り込みを生成します。
- ⑨ - 9 番目のクロックの終わりにマスターが NACK を送信した場合、それ以上のデータ要求はありません。モジュールはクロック中止や割り込み生成を行いません。
- ⑩ - スレーブが STOP イベントを検出して S ビットクリア、P ビットをセットします。

## 21.8 I<sup>2</sup>C バス接続についての検討事項

バス接続としての I<sup>2</sup>C バスの定義では、図 21-33 の RP で示すように、バス上にプルアップ抵抗器が必要です。図内で Rs として表されるシリーズ抵抗器はオプションで ESD 感度を改善するためには使います。抵抗器 RP および Rs の値は以下のパラメータにより変動します。

- 供給電力
- バス容量
- 接続デバイスの数（入力電流 + 漏洩電流）

デバイスは RP に対しバスを Low にできる必要があるため、RP に流れる電流がデバイス出力が VOL(MAX) = 0.4V の場合に 3 mA の I/O ピン最小シンク電流 IOL よりも小さくする必要があります。例えば、供給電力 VDD = 5V +10% で以下のようにになります。

$$RP(MIN) = (VDD(MAX) - VOL(MAX)) / IOL = (5.5-0.4) / 3 \text{ mA} = 1.7 \text{ k}\Omega$$

仕様では、最小立上り時間は 400 kHz システムでは 300 nsec、100 kHz システムでは 1000 nsec となっています。

RP は合計容量 CB に対しバスを引き揚げる必要があるため、0.7 VDD で最大立上り時間が 300 nsec の場合、RP の最大抵抗は次より小さくなる必要があります。

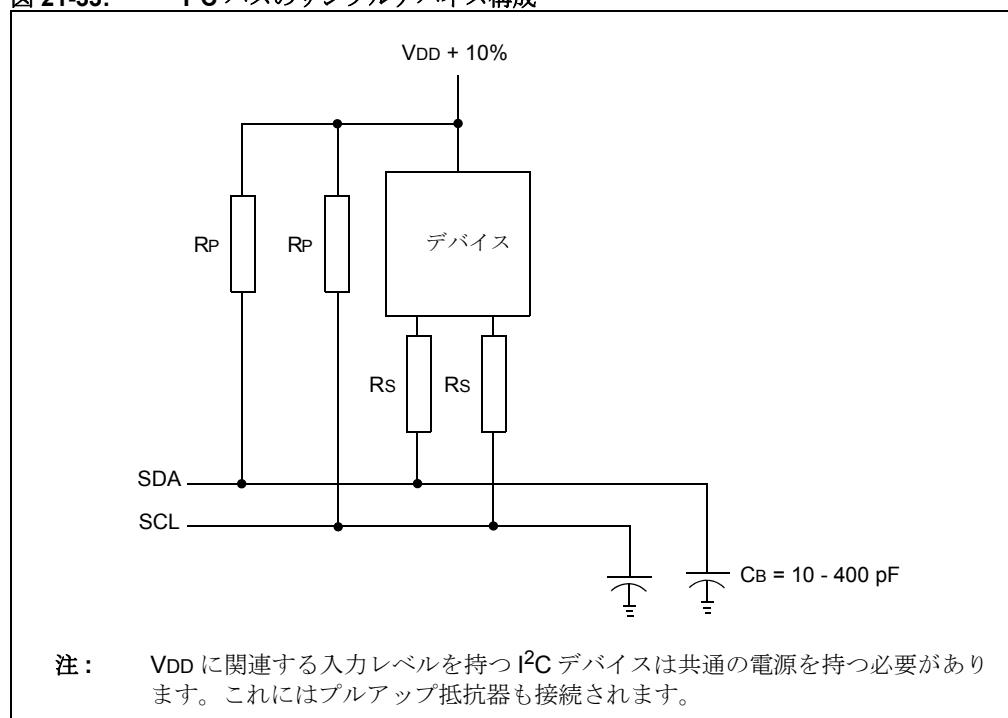
$$RP(MAX) = -tR / CB * \ln(1 - (VIL(MAX) - VDD(MAX))) = -300 \text{ nsec} / (100\text{pF} * \ln(1-0.7)) = 2.5 \text{ k}\Omega$$

Rs の最大値は Low レベルのときの推奨雑音余裕により決定します。Rs はデバイス VOL プラス Rs 降下電圧を最大 VIL より大きくすることができません。

$$Rs(MAX) = (VIL(MAX) - VOL(MIN)) / IOL(MAX) = (0.3 \text{ VDD}-0.4) / 3 \text{ mA} = 366\Omega$$

SCL クロック入力は正常な動作のための最小 High 時間及び Low 時間を確保する必要があります。I<sup>2</sup>C 仕様の High 時間と Low 時間は I<sup>2</sup>C モジュールの要件でもあり、特定のデバイスのデータシートの「電気的仕様」セクションで示されています。

図 21-33: I<sup>2</sup>C バスのサンプルデバイス構成



## 21.8.1 統合信号調整

SCL ピンおよび SDA ピンは入力グリッヂフィルタを持っています。 $\text{I}^2\text{C}$  バスはこのフィルタを 100 kHz システムと 400 kHz システムの両方で必要とします。

400 kHz バスで動作している場合、 $\text{I}^2\text{C}$  仕様ではデバイスピニ出力のスルーレート制御が必要となります。スルーレート制御はデバイスに組み込まれています。DISSLW ビット (I2CCON<9>) がクリアされると、スルーレート制御はアクティブになります。その他のバス速度では、スルーレート制御は不要で DISSLW を設定する必要があります。

ある種の  $\text{I}^2\text{C}$  バスのシステムでは  $V_{IL}$ ( 最大 ) と  $V_{IH}$ ( 最小 ) に異なる入力レベルを必要とする場合があります。

通常の  $\text{I}^2\text{C}$  システムでは以下のようにになります。

$V_{IL}$ ( 最大 ) = 1.5V かつ 0.3 VDD より小さい

$V_{IH}$ ( 最小 ) = 3.0V かつ 0.7 VDD より大きい

SMBus (システム管理バス) システムでは以下のようにになります。

$V_{IL}$ ( 最大 ) = 0.2 VDD

$V_{IH}$ ( 最小 ) = 0.8 VDD

SMEN ビット (I2CCON<8>) は入力レベルを制御します。SMEN は入力レベルを SMBus 仕様に合うよう変更して設定されます。

## 21.9 PWRSAV 命令実行中のモジュール動作

### 21.9.1 デバイスが SLEEP モードにある場合

デバイスが PWRSAV 0 命令を実行すると、デバイスは SLEEP モードに入ります。デバイスが SLEEP モードになるとマスターおよびスレーブモジュールは保留中のメッセージアクティビティを破棄しモジュールの状態をリセットします。デバイスが SLEEP からウェイクアップしても、進行中の送信 / 受信はすべて続行されません。デバイスが動作モードに戻ると、マスター モジュールは IDLE 状態になりメッセージコマンドを待ち、スレーブモジュールは START 条件を待ちます。SLEEP モードの間は、IWCOL、I2COV、BCL ビットはクリアされます。さらに、マスター機能が中止されるため、SEN、RSEN、PEN、RCEN、ACKEN および TRSTAT ビットがクリアされます。TBF および RBF もクリアされ、バッファがウェイクアップ時に使用可能になります。

送信 / 受信がアクティブまたは保留の状態では、SLEEP モードに入ることを自動的に防ぐ方法はありません。ソフトウェアは SLEEP への切り替えと I<sup>2</sup>C 動作を同期させてメッセージの破棄を防ぐ必要があります。

SLEEP の間、スレーブモジュールは I<sup>2</sup>C バスをモニターしません。そのため、I<sup>2</sup>C モジュールを使用して I<sup>2</sup>C バスに基づいたウェイクアップイベントを生成できません。状態変化割り込み入力などのその他割り込み入力は I<sup>2</sup>C バス上のメッセージトラフィックを検出するのに使用でき、デバイスをウェイクアップさせます。

### 21.9.2 デバイスが IDLE モードにある場合

デバイスが PWRSAV 1 命令を実行すると、デバイスは IDLE モードに入ります。モジュールは I2CSIDL ビット (I2CCON<13>) に従い IDEL モードの省電力状態になります。

I2CSIDL = 1 の場合、モジュールは省電力モードに入り、SLEEP モードの間はそれに従い動作します。

I2CSIDL = 0 の場合、モジュールは省電力モードに入りません。モジュールは通常通りに動作を続けます。

## 21.10 RESET の影響

RESET は I<sup>2</sup>C モジュールを無効にし、アクティブなまたは保留中のメッセージアクティビティを終了させます。これらのレジスタの RESET 条件については、I2CCON および I2CSTAT のレジスタ定義を参照してください。

注：	ここでは、「IDLE」とは CPU 省電力状態を意味します。カタカナで「アイドル」とある場合の状態では、I <sup>2</sup> C モジュールはバス上のデータを転送しません。
----	--

## 21.11 設計の秘訣

**質問 1:** バスをマスターとして使用しデータ送信も行っていますが、スレーブ割り込みや受信割り込みが発生します。

**答え:**マスター回路とスレーブ回路はそれぞれ独立しています。スレーブモジュールもマスターが送信したバスからのイベントを受信するようになっています。

**質問 2:** スレーブを使用してデータを I2CTRN レジスタに書き込んでいますが、データが送信されません。

**答え :**スレーブは送信準備中は自動的に待機状態になります。SCLREL ビットが I<sup>2</sup>C クロックを解放するように設定してください。

**質問 3:** マスター モジュールの状態をどのようにして知ることができますか？

**答え :**SEN、RSEN、PEN、RCEN、ACKEN、TRSTAT ビットの状態を見れば、マスター モジュールの状態が分かります。すべてのビットが '0' の時、モジュールは IDLE 状態にあります。

**質問 4:** スレーブ操作時に、STREN = 0 の状態でバイトを受信しました。次のバイト受信までにこのバイトを処理できない場合、ソフトウェアはどのような動作を行いますか？

**答え :**STREN が '0' であったため、このモジュールは自動的に受信バイトの WAIT を生成できなかったと思われます。ただし、メッセージ処理中のいつでもソフトウェアは STREN をセットし SCLREL をクリアできます。これにより次の受信時に WAIT を生成し SCL クロックと同期できます。

**質問 5:** マルチマスター I<sup>2</sup>C システムを使用しています。メッセージを送信しようとすると、文字化けしてしまいます。

**答え :**マルチマスター システムでは、他のマスターがバス衝突を発生させる場合があります。マスターの割り込みサービスルーチンで、BCL ビットをチェックしオペレーションが衝突なしに完了しているかどうかを確認してください。衝突が検出された場合、メッセージは最初から送信しなおす必要があります。

**質問 6:** マルチマスター I<sup>2</sup>C システムを使用しています。メッセージ開始のタイミングをどのようにして知ればいいでしょうか？

**答え :**S ビットと P ビットを見てください。S = 0、P = 0 の場合、バスは IDLE です。S = 0、P = 1 の場合も、バスは IDLE です。

**質問 7:** バスで START 条件を開始し I2CTRN レジスタに書き込みをしバイトを送信しようとしたら、バイトが送信されませんでした。何故でしょうか？

**答え :**I<sup>2</sup>C バスの各イベントが完了してから次のイベントを開始してください。この場合、SEN ビットをポーリングして START イベントがいつ完了するかを確認するか、データが I2CTRN に書き込まれる前にマスター I<sup>2</sup>C 割り込みが発生するのを待ちます。

## 21.12 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありません。ただし、その概念は適切であり、修正や可能な制限をして使用できる可能性もあります。I<sup>2</sup>C モジュールに関連する現在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 #
I <sup>2</sup> C <sup>TM</sup> マルチマスター環境での SSP モジュールの使用	AN578
スレーブ I <sup>2</sup> C <sup>TM</sup> 通信のための PICmicro <sup>®</sup> SSP の使用	AN734
マスター I <sup>2</sup> C <sup>TM</sup> 通信のための PICmicro <sup>®</sup> MSSP モジュールの使用	AN735
環境モニタリングのための I <sup>2</sup> C <sup>TM</sup> ネットワークプロトコル	AN736

**注：** デバイスの dsPIC30F ファミリー関しての、追加のアプリケーションノートやコード例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 21.13 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

この版は dsPIC30F I<sup>2</sup>C モジュールの完全な説明を含むよう情報が追加されています。



**MICROCHIP**

## 第 22 章. データ変換器インターフェース (DCI)

### ハイライト

この章は、以下の項目を含んでいます。

**22**

データ変換器  
インターフェース  
(DCI)

22.1 序章 .....	22-2
22.2 制御レジスタの説明 .....	22-2
22.3 コーデックインターフェースの基礎と技術 .....	22-8
22.4 DCI 動作 .....	22-10
22.5 DCI モジュールを使用する .....	22-17
22.6 省電力モードでの動作 .....	22-28
22.7 DCI に関連するレジスタ .....	22-28
22.8 設計の秘訣 .....	22-30
22.9 関連するアプリケーションノート .....	22-31
22.10 改訂履歴 .....	22-32

## 22.1 序章

dsPIC のデータ変換器インターフェース (DCI) モジュールは、オーディオコーダー / デコーダー( コーデック )、A/D 変換器および D/A 変換器のようなデバイスのインターフェースを簡単にします。

以下のインターフェースがサポートされます。

- フレーム化された同期式シリアル転送( 単チャネルもしくは複数チャネル )
- Inter-IC Sound(I<sup>2</sup>S) インターフェース
- AC-Link 準拠モード

オーディオアプリケーションでの使用を目的とした多くのコーデックは、8 kHz と 48 kHz の間のサンプリングレートをサポートしており、上記のインターフェースプロトコルのうちの 1つを使用します。DCI は自動的にこれらのコーデックに対応したインターフェースタイミングを扱います。CPU のオーバーヘッドは、要求された量のデータが DCI により送信および / もしくは受信されるまでありません。CPU 割り込みの間に、最大 4 ワードが転送されます。

DCI 用のデータワード長は、dsPIC30F CPU のデータサイズに合わせて、最大 16 ビットまでプログラム可能です。但し、多くのコーデックでは 16 ビット以上のデータワードサイズを持っています。ロングデータワード長は DCI でサポートされています。DCI は複数の 16 ビットタイムスロットでロングワードデータの送信 / 受信ができるように構成されています。この動作はユーザーにとって透過であり、ロングデータは連続するレジスタに格納されます。

DCI はデータフレーム内に最大 16 タイムスロットまで、最大 256 ビットのフレームサイズまでサポートできます。データフレーム内の各タイムスロット毎に DCI が送信 / 受信のどちらを行うかを決定する制御ビットがあります。

## 22.2 制御レジスタの説明

DCI は以下に示す 5 つの制御レジスタと 1 つのステータスレジスタを持ちます。

- DCICON1: DCI モジュール有効・モードビット
- DCICON2: DCI モジュールワード長、データフレーム長およびバッファセットアップ
- DCICON3: DCI モジュールビットクロックジェネレータセットアップ<sup>†</sup>
- DCISTAT: DCI モジュールステータス情報
- RSCON: データ受信用アクティブフレームタイムスロット制御
- TSCON: データ送信用アクティブフレームタイムスロット制御

これらの制御およびステータスレジスタに加えて、4 つの送信レジスタ TXBUF0...TXBUF3 および 4 つの受信レジスタ RXBUF0...RXBUF3 があります。

## レジスタ 22-1: DCICON1

上位バイト :							
R/W-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
DCIEN	—	DCISIDL	—	DLOOP	CSCKD	CSCKE	COFSD
ビット 15							ビット 8

下位バイト :							
R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
UNFM	CSDOM	DJST	—	—	—	COFSM<1:0>	
ビット 7							ビット 0

- ビット 15 **DCIEN:** DCI モジュール有効ビット  
 1 = モジュールを有効にする。  
 0 = モジュールを無効にする。
- ビット 14 **予約済み:** ‘0’ が読み込まれます。
- ビット 13 **DCISIDL:** アイドルモードでの DCI ストップ制御ビット  
 1 = CPU アイドルモードでモジュールを停止させる。  
 0 = CPU アイドルモードでモジュールを継続動作させる。
- ビット 12 **予約済み:** ‘0’ が読み込まれます。
- ビット 11 **DLOOP:** デジタルループバック制御ビット  
 1 = デジタルループバックを有効にする。CSDI と CSDO ピンが内部で接続されます。  
 0 = デジタルループバックを無効にする。
- ビット 10 **CSCKD:** サンプリングクロックの方向制御ビット  
 1 = DCI モジュールが有効の時 CSCK ピンを入力にする。  
 0 = DCI モジュールが有効の時 CSCK ピンを出力にする。
- ビット 9 **CSCKE:** サンプリングクロックエッジ制御ビット  
 1 = シリアルクロックの立下りエッジでデータが変化し、シリアルクロックの立ち上がりでデータがサンプリングされます。  
 0 = シリアルクロックの立上がりエッジでデータが変化し、シリアルクロックの立ち下がりでデータがサンプリングされます。
- ビット 8 **COFSD:** フレーム同期方向制御ビット  
 1 = DCI モジュールが有効の時 COFS ピンを入力にする。  
 0 = DCI モジュールが有効の時 COFS ピンを出力にする。
- ビット 7 **UNFM:** アンダーフローモードビット  
 1 = 送信アンダーフロー時には送信レジスタの最後の値が送信されます。  
 0 = 送信アンダーフロー時には ‘0’ が送信されます。
- ビット 6 **CSDOM:** シリアルデータ出力モードビット  
 1 = 送信タイムスロットが無効の間は CSDO ピンをトライステート状態にする。  
 0 = 送信タイムスロットが無効の間は CSDO ピンドライブを ‘0’ 状態にする。
- ビット 5 **DJST:** DCI データ詰め制御ビット  
 1 = データ送信 / 受信が、フレーム同期パルスと同じシリアルクロックサイクルの間に開始されます。  
 0 = データ送信 / 受信が、フレーム同期パルスの 1 シリアルクロックサイクル後に開始されます。
- ビット 4-2 **予約済み:** ‘0’ が読み込まれます。
- ビット 1-0 **COFSM<1:0>:** フレーム同期モードビット  
 11 = 20 ビット AC-Link モード  
 10 = 16 ビット AC-Link モード  
 01 = I<sup>2</sup>S フレーム同期モード  
 00 = 複数チャネルフレーム同期モード

凡例 :

R = 読み出し可能ビット

W = 書き込み可能ビット U = 未定ビット、‘0’ が読み込まれます

-n = POR の値

‘1’ = ビットがセット ‘0’ = ビットはクリア x = ビットは不定  
されます。

## レジスタ 22-2: DCICON2

上位バイト：							
U-0	U-0	U-0	U-0	R/W-0	R/W-0	U-0	R/W-0
—	—	—	—	BLEN<1:0>	—	—	COFSG3
ビット 15							ビット 8

下位バイト：							
R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
COFSG<2:0>	—	—	—	WS<3:0>	—	—	—
ビット 7							ビット 0

ビット 15-12 予約済み：‘0’が読み込まれます。

ビット 11-10 **BLEN<1:0>**: バッファ長制御ビット

11 = 4 データワードが割り込み中にバッファされます。  
10 = 3 データワードが割り込み中にバッファされます。  
01 = 2 データワードが割り込み中にバッファされます。  
00 = 1 データワードが割り込み中にバッファされます。

ビット 9 予約済み：‘0’が読み込まれます。

ビット 8-5 **COFSG<3:0>**: フレーム同期生成器制御ビット

1111 = データフレームが 16 ワードで構成される  
||  
0010 = データフレームが 3 ワードで構成される  
0001 = データフレームが 2 ワードで構成される  
0000 = データフレームが 1 ワードで構成される

ビット 4 予約済み：‘0’が読み込まれます。

ビット 3-0 **WS<3:0>**: DCI データワードサイズビット

1111 = データワードサイズが 16 ビットです。  
||  
0100 = データワードサイズが 5 ビットです。  
0011 = データワードサイズが 4 ビットです。  
0010 = 無効選択。使用禁止。予期せぬ結果が発生する可能性があります。  
0001 = 無効選択。使用禁止。予期せぬ結果が発生する可能性があります。  
0000 = 無効選択。使用禁止。予期せぬ結果が発生する可能性があります。

凡例：

R = 読み出し可能ビット

W = 書き込み可能ビット

U = 未定ビット、‘0’が読み込まれます

-n = POR の値

‘1’ = ビットがセット

‘0’ = ビットはクリア  
x = ビットは不定

## レジスタ 22-3: DCICON3

上位バイト：							
U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	BCG<11:8>			
ビット 15							ビット 8

下位バイト：							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BCG<7:0>				BCG<7:0>			
ビット 7							ビット 0

ビット 15-12 予約済み：‘0’が読み込まれます。

ビット 11-0 **BCG<11:0>:** DCI ビットクロックジェネレータ制御ビット

凡例：

R = 読み出し可能ビット      W = 書き込み可能      U = 未定ビット、‘0’が読み込まれます

ビット

-n = POR の値      ‘1’ = ビットがセット      ‘0’ = ビットはクリア      x = ビットは不定されます。

## レジスタ 22-4: DCISTAT

上位バイト :							
U-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
—	—	—	—		SLOT<3:0>		
ビット 15							ビット 8

下位バイト :							
U-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
—	—	—	—	ROV	RFUL	TUNF	TMPTY
ビット 7							ビット 0

ビット 15-12 予約済み: ‘0’ が読み込まれます。

ビット 11-8 **SLOT<3:0>:** DCI スロットステータスビット  
 1111 = スロット # 15 が現在アクティブです。  
 ||  
 0010 = スロット # 2 が現在アクティブです。  
 0001 = スロット # 1 が現在アクティブです。  
 0000 = スロット # 0 が現在アクティブです。

ビット 7-4 予約済み: ‘0’ が読み込まれます。

ビット 3 **ROV:** 受信オーバーフローステータスビット  
 1 = 少なくとも 1 つの受信レジスタで受信オーバーフローが発生しています。  
 0 = 受信オーバーフローは発生していません。

ビット 2 **RFUL:** 受信バッファフルステータスビット  
 1 = 新しいデータが受信可能です。  
 0 = 受信レジスタには旧データがあります。

ビット 1 **TUNF:** 送信バッファアンダーフローステータスビット  
 1 = 少なくとも 1 つの送信レジスタで送信オーバーフローが発生しています。  
 0 = 送信オーバーフローは発生していません。

ビット 0 **TMPTY:** 送信バッファエンプティステータスビット  
 1 = 送信レジスタは空です。  
 0 = 送信レジスタは空ではありません。

凡例 :

R = 読み出し可能ビット	W = 書き込み可能	U = 未定ビット、‘0’ が読み込まれます
	ビット	
-n = POR の値	‘1’ = ビットがセット	‘0’ = ビットはクリア x = ビットは不定
		されます。

## レジスタ 22-5: RSCON

上位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RSE15	RSE14	RSE13	RSE12	RSE11	RSE10	RSE9	RSE8
ビット 15							ビット 8

下位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RSE7	RSE6	RSE5	RSE4	RSE3	RSE2	RSE1	RSE0
ビット 7							ビット 0

ビット 11 **RSE<15:0>**: 受信スロット有効ビット  
 1 = CSDI データが、個別タイムスロット n の間に受信可能。  
 0 = CSDI データが、個別タイムスロット n の間に受信不可。

凡例 :

R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未定ビット、'0' が読み込まれます
-n = POR の値	'1' = ビットがセット	'0' = ビットはクリア
	x = ビットは不定されます。	

## レジスタ 22-6: TSCON

上位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TSE15	TSE14	TSE13	TSE12	TSE11	TSE10	TSE9	TSE8
ビット 15							ビット 8

下位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TSE7	TSE6	TSE5	TSE4	TSE3	TSE2	TSE1	TSE0
ビット 7							ビット 0

ビット 11 **TSE<15:0>**: 送信スロット有効制御ビット  
 1 = 送信バッファ内容が、個別タイムスロット n の間に送信可能。  
 0 = CSDOM ビットの設定に従って、個別タイムスロットの間に、CSDO ピンがトライステートかもしくは論理 '0' に駆動されます。

凡例 :

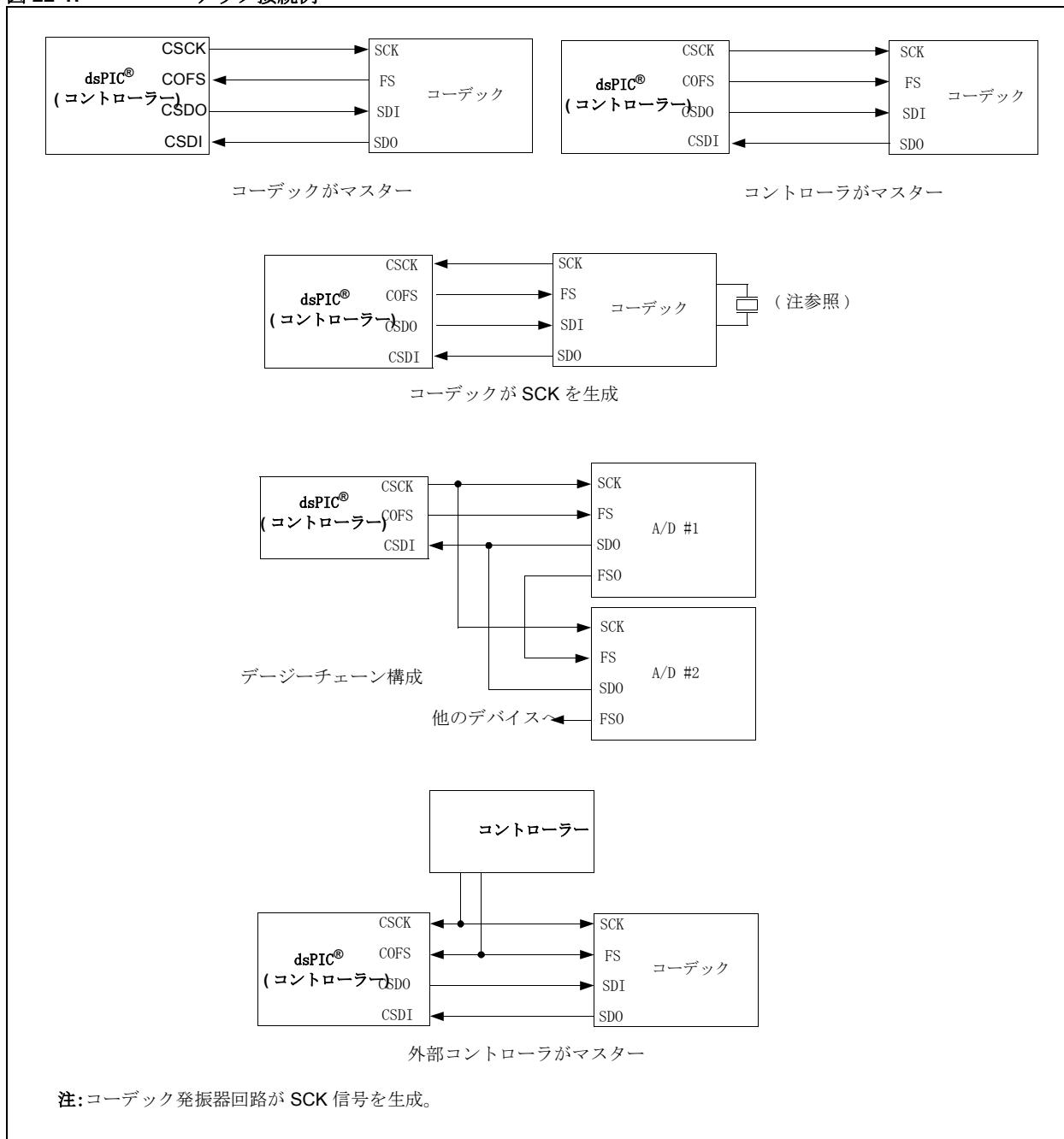
R = 読み出し可能ビット	W = 書き込み可能ビット	U = 未定ビット、'0' が読み込まれます
-n = POR の値	'1' = ビットがセット	'0' = ビットはクリア
	x = ビットは不定されます。	

## 22.3 コーデックインターフェースの基礎と技術

DCI でサポートされているインターフェースプロトコルは、2つのデバイス間のデータインターフェースを起動するために、フレーム同期 (fs) 信号を使用する必要があります。ほとんどの場合、fs の立ち上がりエッジで新しいデータ転送が開始されます。どんなコーデックアプリケーションでも、最低限、コントローラとコーデックデバイスがあります。どちらかのデバイスが FS を生成します。fs を生成するデバイスがマスターデバイスです。その結果、マスターデバイスが送信もしくは受信デバイスである必要はありません。種々の接続例を図 22-1 に示します。fs 信号の周波数は通常システムのサンプリング周波数 fs です。

**注:** この章で説明している詳細情報は、DCI モジュールにだけに限定していません。ここで議論は、ほとんどのコーデックデバイスで使用されるデジタルシリアルインターフェースプロトコルに関連したいくつかの背景と専門用語を、ユーザーに説明することを目的としています。

図 22-1: コーデック接続例



すべてのインターフェースは、シリアル転送クロック SCK を持っています。SCK 信号は、接続されたデバイスのどれかで生成されるか、外部から供給されます。ある場合は、SCK はビットクロックとも呼ばれます。ハイファイ用コーデックでは、SCK 信号は通常コーデックデバイス内の水晶発振器から供給されます。プロトコルは、データがサンプリングされる SCK のエッジを規定します。マスターデバイスが、SCK に同期して FS 信号を生成します。

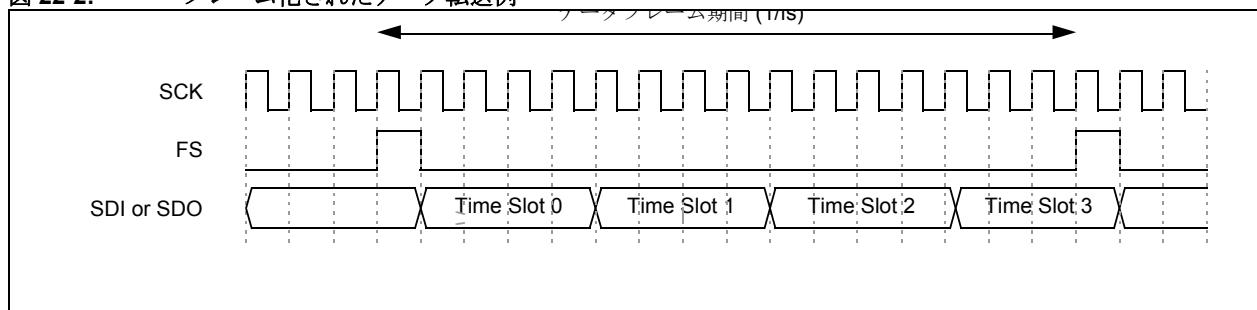
FS 信号の周期が、1 データフレームを形成します。この周期は、データのサンプリング周期と同じです。データフレーム間に発生する SCK サイクルの数は選択されたコーデックのタイプに依存します。システムのサンプリングレート対 SCK 周波数の比は n 比として表され、n はデータフレーム当たりの SCK 周期の数を表します。

フレーム化されたインターフェースプロトコルを用いる利点の 1 つは、それぞれのサンプリング周期間に複数のデータワードが転送可能であることです。データフレームのそれぞれの区分はタイムスロットと呼ばれます。タイムスロットは、複数のコーデックデータチャネルおよび / もしくは制御情報に用いられます。さらに、同じシリアルデータピンに複数デバイスを多重化できます。それぞれのスレーブデバイスは、シリアルデータ接続のデータの位置をそれぞれ決められたタイムスロットに割り当てられます。それぞれのスレーブデバイスの出力は、その他のデバイスがシリアルバスを使用することができるよう、割り当てられたスロット以外の時間はトライステート状態になります。

いくつかのデバイスでは、フレーム同期出力 (FSO) を使って FS 信号をデージーチェーンにより接続できるものもあります。典型的なデージーチェーンの構成を図 22-1 に示します。最初のスレーブデバイスからの転送が完了すると、FS パルスが、FSO を使って、チェーン上の第 2 のデバイスに送られます。このプロセスが、チェーン上の最後のデバイスがデータを送信するまで継続されます。コントローラ (マスター) デバイスは、転送されるデータワードのすべてが供給されるようにデータフレームサイズをプログラムする必要があります。

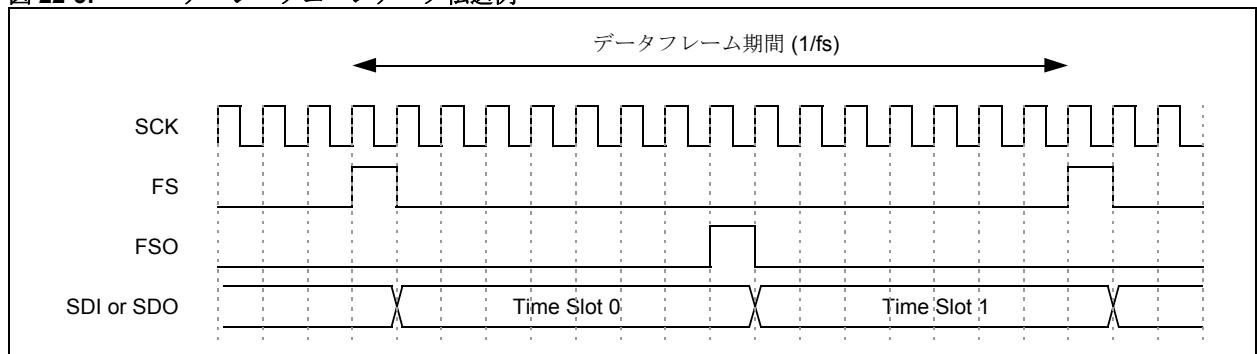
典型的なデータ転送のタイミングを図 22-2 に示します。ほとんどのプロトコルは、FS 信号が検出された 1 SCK サイクル後にデータ転送を開始します。この例では、16 fs クロックを用い、1 フレーム当たり 4 つの 4 ビットデータワードを転送します。

図 22-2: フレーム化されたデータ転送例



デージーチェーンデバイスを用いた典型的なデータ転送のタイミングを図 22-3 に示します。この例では、16 fs SCK 周波数を用い、1 フレーム当たり 2 つの 8 ビットデータワードを転送します。FS パルスが検出された後、チェーン上の最初のデバイスが最初の 8 ビットデータワードを転送し、転送の最後に FSO 信号を生成します。FSO 信号は、チェーン上の第 2 のデバイスからの第 2 のデータワードの転送を開始します。

図 22-3: デージーチェーンデータ転送例



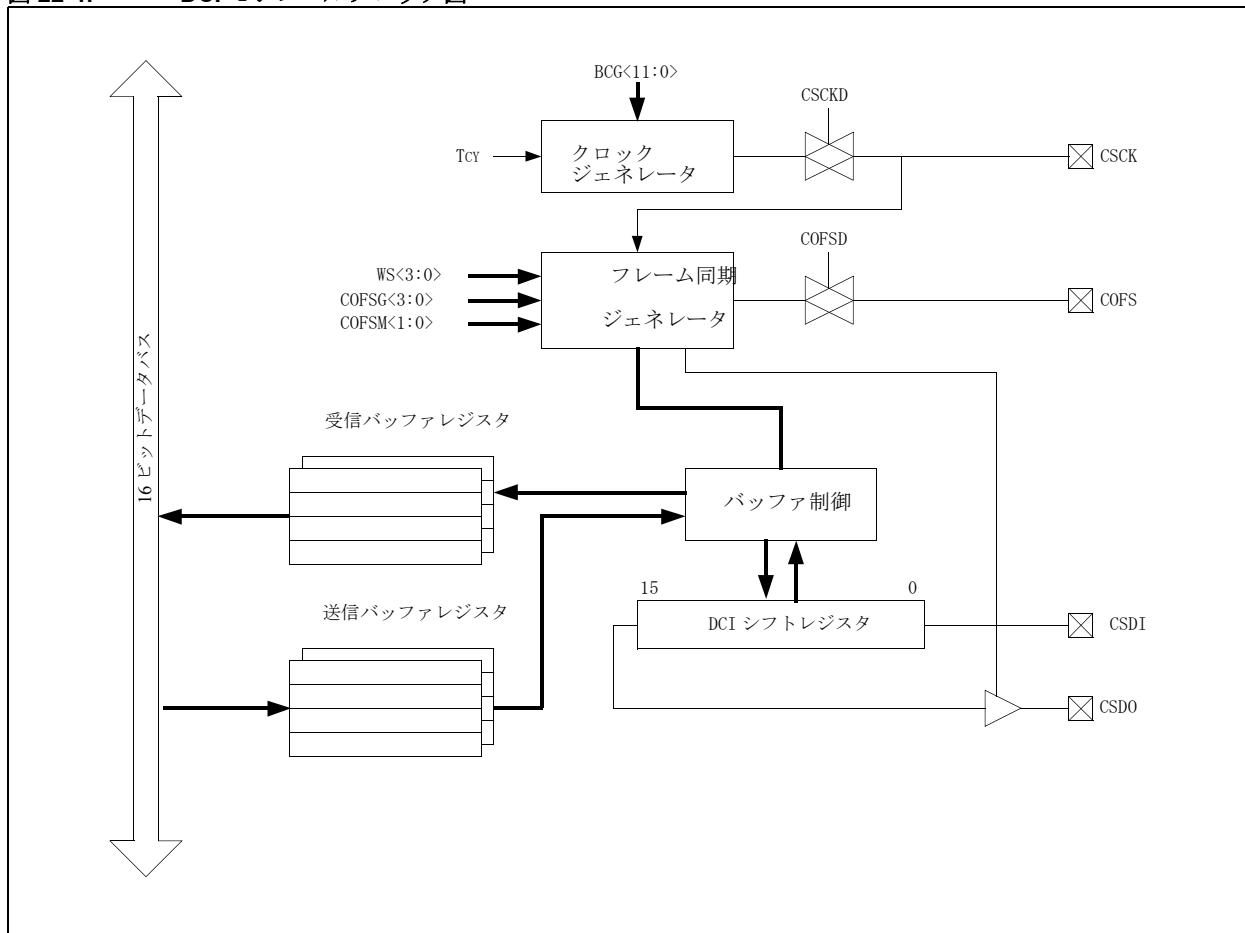
FS パルスは、最低 1 SCK 周期のアクティブ時間を持ち、それによりスレーブデバイスがデータフレームの開始を検出します。FS パルスのデューティーサイクルは、データフレーム内のある境界を示すために用いられる特定のプロトコルに依存して変わります。例えば、I<sup>2</sup>S プロトコルは、50% のデューティーサイクルを持つ FS 信号を用います。I<sup>2</sup>S プロトコルは、2 つのデータチャネル（左右チャネルのオーディオ情報）を転送するために最適化されています。FS 信号のエッジは左右チャネルのデータワードの境界を示します。AC-Link プロトコルは、高域では 16SCK 周期を、低域では 240SCK 周期の FS 信号を用います。AC-Link FS 信号のエッジは、フレーム内の制御情報とデータの境界を示します。

**注：** コーデック通信プロトコルに関する追加情報については本マニュアルの“付録”を参照してください。

## 22.4 DCI 動作

モジュールの簡略ブロック図を図 22-4 に示します。モジュールは、バッファ制御ユニットを経由して少容量のメモリバッファに接続された送信 / 受信シフトレジスタから構成されます。この配置により DCI が種々のコーデックシリアルプロトコルをサポートできます。DCI シフトレジスタは 16 ビット幅を持ちます。データは、DCI により MS ビットから送信 / 受信されます。

図 22-4: DCI モジュールブロック図



## 22.4.1 DCI 関連ピン

DCI に関連して 4 つの I/O ピンがあります。DCI を有効にしたら、4 つのピン 1 つ 1 つのデータ方向を制御します。

### 22.4.1.1 CSCK ピン

CSCK ピンは、DCI 用にシリアルクロックを供給します。CSCK ピンは、CSCKD 制御ビット DCICON1<10> を用いて、入力もしくは出力として構成されます。CSCK ピンが出力に構成される (CSCKD = 0) と、シリアルクロックは dsPIC30F システムクロックソースから与えられ、DCI により外部デバイスに供給されます。CSCK ピンが入力に構成される (CSCKD = 1) と、シリアルクロックは外部デバイスから供給される必要があります。

### 22.4.1.2 CSDO ピン

シリアルデータ出力 (CSDO) ピンは、モジュールが有効になると、出力専用ピンとして構成されます。CSDO ピンは、データが転送される時はいつでも、シリアルバスを駆動します。CSDO ピンは、CSDOM 制御ビット (DCICON1<6>) に対応して、データが転送されないシリアルクロック周期の間は、トライステートもしくは ‘0’ に駆動されます。トライステートオプションにより、CSDO 接続を、他のデバイスと多重できます。

### 22.4.1.3 CSDI ピン

シリアルデータ入力 (CSDI) ピンは、モジュールが有効になると入力専用ピンとして構成されます。

### 22.4.1.4 COFS ピン

フレーム同期 (COFS) ピンは、CSDO と CSDI ピンで発生するデータ転送の同期をとるために使用されます。COFS ピンは、入力もしくは出力として構成可能です。COFS ピンのデータ方向は、COFSD 制御ビット (DCICON1 <8>) により決定されます。COFSD ビットがクリアされると、COFS ピンは出力になります。DCI モジュールは、データ転送を開始するためにフレーム同期信号を生成します。この構成では DCI はマスターデバイスになります。COFSD ビットがセットされると、COFS ピンは入力になります。モジュールに対して入力される同期信号によってデータ転送を開始します。COSFD 制御ビットがセットされると、DCI はスレーブデバイスになります。

## 22.4.2 モジュール有効

DCI モジュールは、DCIEN 制御ビット (DCICON1<15>) をセット / クリアすることにより有効 / 無効にできます。DCIEN 制御ビットをクリアすることにより、モジュールをリセットできます。特に、シリアルクロック生成、フレーム同期およびバッファ制御論理がリセットされます。(追加情報については、[22.5.1.1 「DCI のスタートアップとデータバッファリング」](#) および [22.5.1.2 「DCI の無効化」](#) を参照してください。)

有効になると、DCI はモジュールに関連した、CSCK, CSDI, CSDO および COFS I/O ピンのデータ方向を制御します。これらの I/O ピンの PORT, LAT および TRIS レジスタ値は、DCIEN ビットがセットされた場合は、DCI モジュールにより上書きされます。

ビットクロック生成器が有効の場合は、CSCK ピンを独立に上書きすることもできます。これにより、DCI モジュールのその他の部分を有効にすることなくビットクロック生成器を動作させることができます。

## 22.4.3 ビットクロックジェネレータ

DCI モジュールは、ビットクロックを生成する専用の 12 ビットタイムベースを持っています。ビットクロックレート (周期) は、BCG<11:0> 制御ビット (DCICON3<11:0>) に非ゼロの 12 ビットの値を書き込むことでセットされます。BCG<11:0> ビットがゼロにセットされると、ビットクロックは無効になります。

**注：** DCIEN ビットがセットされるか、BCG<11:0> に非ゼロの値を書き込むことによりビットクロック生成器が有効になると、CSCK I/O ピンは DCI モジュールで制御されます。これにより、ビットクロック生成器は、DCI モジュールとは独立に動作させることができます。

CSCK ピンが DCI モジュールで制御されると、それに対応する PORT、LAT および TRIS 制御レジスタの CSCK 用の値は上書きされ、CSCK ピンのデータ方向は CSCKD 制御ビット (DCICON1<10>) により制御されます。

DCI 用シリアルクロックが外部デバイスから供給される場合、BCG<11:0> ビットは ‘0’ にセットし、CSCKD ビットは ‘1’ にセットする必要があります。

シリアルクロックが DCI モジュールにより生成される場合は、BCG<11:0> 制御ビットは非ゼロの値 (式 22-1 参照) に設定する必要があり、CSCKD 制御ビットはゼロにセットする必要があります。

ビットクロック周波数は式 22-1 で計算できます。

#### 式 22-1: DCI ビットクロック生成値

$$BCG<11:0> = \frac{f_{CY}}{2^2 f_{CSCK}} - 1$$

必要なビットクロック周波数は、システムのサンプリングレートとフレームサイズで決まります。典型的なビットクロック周波数は、使用するデータ変換器と通信プロトコルに依存しますが、16 倍から 512 倍の変換器サンプリングレートの範囲になります。

**注:** BCG<11:0> ビットは、CSCK 信号が外部から与えられる時 (CSCKD = 1) には、DCI モジュールの動作には影響を与えません。

#### 22.4.4 サンプリングクロックのエッジ選択

CSCKE 制御ビット (DCICON1<9>) はシリアルクロック信号用のサンプリングエッジを決定します。CSCKE ビットがクリア (デフォルト値) されると、データは CSCK 信号の立ち下がりエッジでサンプリングされます。AC-Link プロトコルとほとんどの複数チャネルフォーマットでは、CSCK 信号の立ち下りエッジでサンプリングすることが必要です。CSCKE ビットがセットされると、データは CSCK 信号の立ち上がりエッジでサンプリングされます。 $I^2S$  プロトコルでは、CSCK 信号の立ち上がりエッジでサンプリングすることが必要です。

#### 22.4.5 フレーム同期モード制御ビット

DCI でサポートされるインターフェースプロトコルの種類は、COFSM<1:0> 制御ビット (DCICON1<1:0>) を使用することで選択します。以下の動作モードが選択できます。

- 複数チャネルモード
- $I^2S$  モード
- AC-Link モード (16 ビット)
- AC-Link モード (20 ビット)

それぞれのプロトコルの特定情報については以下の章で説明します。

#### 22.4.6 ワードサイズ選択ビット

WS<3:0> ワードサイズ選択ビット (DCICON2<3:0>) は、1 つの DCI データワード内のビット数を決定します。4 から 16 ビットのうちの 1 つのデータ長が選択されます。

**注:** WS 制御ビットは、複数チャネルもしくは  $I^2S$  モードのみで使用されます。これらのビットは、プロトコルでデータサイズが固定化される AC-Link モードでは影響を与えません。

### 22.4.7 フレーム同期生成

フレーム同期ジェネレータ (FSG) は、データワード内のフレーム長を設定する 4 ビットカウンタです。FSG 用の周期は、COFSG<3:0> 制御ビット (DCICON2<8:5>) に書き込むことで設定されます。FSG 周期 (シリアルクロックサイクル中の) は以下の式で与えられます。

式 22-2:

フレーム長、CSCK サイクル

$$\text{FrameLength} = (\text{WS}<3:0> + 1) \times (\text{COFSG}<3:0> + 1)$$

最大 16 データワードまでのフレーム長が選択できます。シリアルクロック周期内のフレーム長は、選択されたワードサイズに依存して、最大 256 まで変化します。

**注：** COFSG 制御ビットは、AC-Link モードでは影響を与えません。なぜなら、プロトコルでフレーム長は 256 シリアルクロック周期と設定されているからです。

### 22.4.8 送信と受信レジスタ

DCI は 4 つの送信レジスタ、TXBUF0...TXBUF3 と 4 つの受信レジスタ、RXBUF0..RXBUF3 を持っています。送信レジスタと受信レジスタはすべてメモリ上にマッピングされます。

#### 22.4.8.1 バッファデータ配列

データ値は DCI レジスタ内ではつねに左詰めで格納されます、なぜなら、オーディオ PCM データは、符号つき 2 の補数小数で表示されるからです。プログラムされた DCI ワードサイズが 16 ビット以下の場合は、受信レジスタ内の未使用下位ビットはモジュールにより ‘0’ にセットされます。また、送信レジスタ内の未使用下位ビットもモジュールで無視されます。

#### 22.4.8.2 送信と受信バッファ

送信 / 受信レジスタはそれぞれユーザーがアクセスできないバッファセットを持っています。実質的には、送信 / 受信バッファは二重バッファとなっています。DCI は送信バッファからデータを送信し、受信データを受信バッファに書き込みます。バッファにより DCI がバッファのデータを使用中でも、ユーザーは RXBUF と TXBUF レジスタを読んだり書いたりできます。

### 22.4.9 DCI バッファ制御ユニット

DCI モジュールはバッファメモリとシリアルシフトレジスタ間のデータ転送を行うバッファ制御ユニットを含みます。バッファ制御ユニットは、バッファメモリと TXBUF と RXBUF レジスタ間のデータ転送も行います。バッファ制御ユニットの働きにより DCI は、CPU のオーバーヘッドなしに、複数データワードの送信 / 受信のキューイングができます。

DCI はバッファメモリと TXBUF/RXBUF レジスタ間の転送の度に割り込みを生成します。割り込みの間にバッファされるデータワード数は BLEN<1:0> 制御ビット (DCICON2<11:10>) により決定されます。送信 / 受信バッファリングのサイズは、BLEN<1:0> ビットを用いて 1 から 4 データワードまで変更できます。

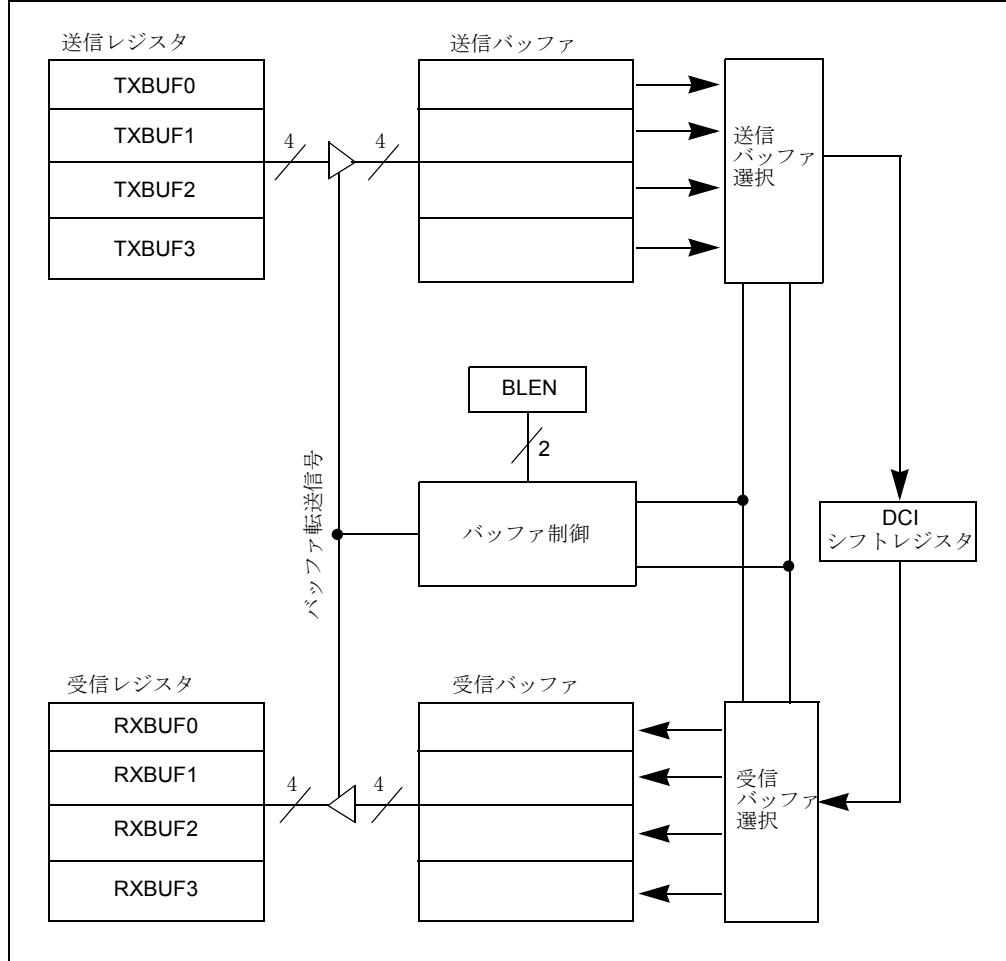
DCI シフトレジスタとバッファメモリ間のデータ転送が発生する度に、DCI バッファ制御ユニット内のポインタが増加して、次のバッファ位置を示します。送信もしくは受信データワードの数が BLEN 値 + 1 になると、以下の状況が発生します。

- バッファ制御ユニットはリセットされ最初のバッファ位置にポインタが戻ります。
- バッファに保持されている受信データは、RXBUF レジスタに転送されます。
- TXBUF 内のデータはバッファに転送されます。
- CPU 割り込みが生成されます。

DCI バッファ制御ユニットはまた、フレーム境界に達する度に、バッファポインタをリセットし最初のバッファ位置にポインタを戻します。この動作により、バッファ位置とデータフレーム内の有効タイムスロット間の配列が確定します。

DCI バッファ制御ユニットは常に送信 / 受信バッファ内の同じ相対位置をアクセスします。例えば、DCI が TXBUF3 からのデータを送信する場合、そのタイムスロット間に受信されたデータはどれも RXBUF3 に書き込まれます。

図 22-5: DCI バッファ制御ユニット



#### 22.4.10 送信スロット有効ビット

TSCON SFR は、最大 16 の送信用タイムスロットを有効にするための制御ビットを持っています。この制御ビットは TSE<15:0> ビットです。1 つ 1 つのタイムスロットのサイズは、WS<3:0> ワードサイズ選択ビットにより決定され、最大 16 ビットまで変化します。

送信タイムスロットが、TSE ビット (TSE<sub>x</sub> = 1) の 1 つにより有効になると、現送信バッファ位置にある内容は CSDO シフトレジスタに転送され、DCI バッファ制御ユニット内のポインタが増加して次のバッファ位置を示します。

選択されたフレームサイズが 16 タイムスロット以下の場合には、すべての TSE 制御ビットがモジュール動作に影響を与えるわけではありません。TSE 制御ビットの上位側ビットは使用されません。例えば、COFSG<3:0> = 0111(1 フレーム当たり 8 データスロット) の場合、TSE8 から TSE15 は DCI 動作に影響を与えません。

##### 22.4.10.1 CSDO モード制御

無効な送信タイムスロットの間は、CSDOM ビット (DCICON1<6>) の設定に従って CSDO ピンは ‘0’ を駆動するかトライステートになります。TSCON レジスタ内の対応する TSE<sub>x</sub> ビットがクリアされると、与えられた送信タイムスロットは無効になります。

CSDOM ビットがクリア(デフォルト値)されると、CSDO ピンは、無効タイムスロット周期の間は CSDO ピンを ‘0’ に駆動します。このモードは、シリアルバスに 2 つのデバイス(1 マスターと 1 スレーブ)のみが接続されている場合に使用されます。

CSDOM ビットがセットされると、CSDO ピンは、未使用タイムスロット周期の間はトライステートになります。このモードにより、複数の dsPIC30F デバイスが多重化されたアプリケーション内のみで同じ CSDO ラインを共有することができます。CSDO ライン上のそれぞれのデバイスは、特定の時間スロット内のみでデータ送信をするように構成されます。どの 2 つのデバイスも同じタイムスロット内ではデータを送信してはなりません。

### 22.4.11 受信スロット有効ビット

RSCON SFR は、最大 16 個の受信用タイムスロットを有効にするために用いられる制御ビット (RSE<15:0>) を含みます。1つ1つの受信タイムスロットのサイズは、WS<3:0> 制御ビットで決定され、4 から 16 ビットまで変化します。

受信タイムスロットが、RSE ビット (RSE<sub>x</sub> = 1) の 1つにより有効になった場合、シフトレジスタの内容は現 DCI 受信バッファ位置に書き込まれ、バッファ制御ロジックがバッファ位置を次に利用できる位置に進めます。

選択されたワードサイズが 16 ビット以下の場合、データは、受信メモリバッファ位置内では詰められません。1つ1つの受信スロットデータは、別々の 16 ビットバッファ内に格納されます。受信メモリバッファ内では、データは常に左詰めで格納されます。

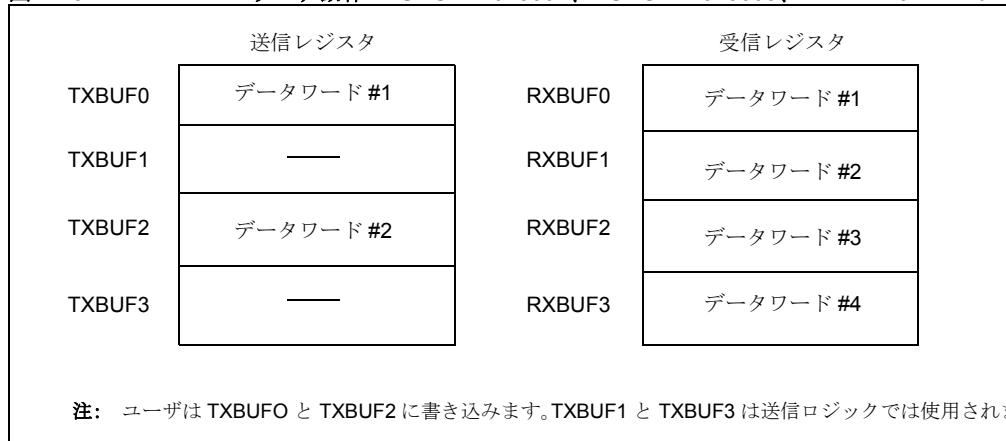
### 22.4.12 バッファ制御ユニットを用いた TSCON と RSCON の動作

スロットは、TSCON と RSCON レジスタ内のイネーブルビットにより個別に有効化されます。が、バッファ制御論理だけは別です。データフレーム内の 1つ1つのタイムスロット毎に、TSE<sub>x</sub> もしくは RSE<sub>x</sub> ビットのどちらかが現タイムスロット用に設定されると、バッファ位置が進みます。これは、データフレーム内の 1つ1つのタイムスロットで、送信 / 受信バッファ位置が常に同じ位置になるように、バッファ制御レジスタユニットが送信 / 受信バッファリングの同期を取ることです。

TSE<sub>x</sub> ビットと RSE<sub>x</sub> ビットと共に、データフレーム内で用いられるすべてのタイムスロット用に設定されると、DCI は同じ量のデータを送信 / 受信します。

あるアプリケーションでは、フレーム中に送信されるデータワードの数が受信されたワードの数と等しくないことがあります。例えば、DCI が 2 種類のワードデータフレームで構成 (TSCON = 0x0001 および RSCON = 0x0003) されるとしましょう。この構成では、DCI が 1 フレーム当たり 1 つのデータワードを送信し、1 フレーム当たり 2 つのデータワードを受信します。送信された 1 データワード当たり 2 つのデータワードが受信されるので、ユーザーは 1 つおきに送信バッファに書き込みます。つまり、TXBUFO と TXBUF2 のみが、データ送信に使用されます。

図 22-6: DCI バッファ動作: TSCON = 0x0001、RSCON = 0x0003、BLEN<1:0> = 11b



### 22.4.13 受信ステータスピット

2つの受信ステータスピット、RFUL と ROV があります。

受信ステータスピットは、モジュールで使用するため有効化されたレジスタに関してのステータスを表示するものです。これは BLEN<1:0> 制御ビットの機能です。バッファ長が 4 ワード以下に設定されている場合は、未使用的バッファは受信ステータスピットに影響を与えません。

RFUL ステータスピット (DCISTAT<2>) は読み出し専用で、受信レジスタに新しいデータが準備できたことを表示します。RFUL ビットは、使用中のすべての RXBUF レジスタがユーザーソフトウェアで読みだされると、自動的にクリアされます。

ROV ステータスピット (DCISTAT<3>) は読み出し専用で、受信レジスタ位置の少なくとも 1 つで受信オーバーフローが発生したことを表示します。受信オーバーフローは、バッファメモリから新しいデータが転送される前に、ユーザーソフトウェアで RXBUF レジスタが読み込まれていない時に発生します。受信オーバーフローが発生すると、レジスタ内の元のデータは上書きされます。オーバーフローを発生させたレジスタが読み込まれると、ROV ステータスピットは自動的にクリアされます。

## 22.4.14 送信ステータスピット

2つの送信ステータスピット、TMPTY と TUNF があります。

送信ステータスピットは、モジュールで使用されるレジスタに関してのステータスを表示するのみです。例えば、バッファ長が 4 ワード以下に設定されると、未使用のレジスタは送信ステータスピットに影響を与えません。

TMPTY ビット (DCISTAT<0>) は、読み出し専用で、有効な TXBUF レジスタの内容が送信バッファレジスタに転送されるとセットされます。TMPTY ビットは、送信レジスタに書き込める時を待つために、ソフトウェアでポーリングできます。TMPTY ビットは、使用中の TXBUF レジスタのどれかへの書き込みが発生すると、ハードウェアで自動的にクリアされます。

TUNF ビット (DCISTAT<1>) は読み出し専用で、使用中の送信レジスタの少なくとも 1 つで送信アンダーフローが発生したことを表示します。TUNF ビットは、TXBUF レジスタの内容が送信バッファメモリに転送されるとき、最後のレジスタがバッファに転送される前に、ユーザーがすべてのレジスタ書き込みが完了しなかった場合にセットされます。アンダーフローを発生させた TXBUF レジスタがユーザーソフトウェアで書き込まれると、TUNF ステータスピットは自動的にクリアされます。

## 22.4.15 スロットステータスピット

SLOT<3:0> ステータスピット (DCISTAT<11:7>) は、データフレーム中の現在動作中のタイムスロットを表示し、1 データフレーム当たり 4 つ以上のワードの転送が必要な場合に役に立ちます。ユーザーは、DCI 割り込みが発生した時に、これらのステータスピットをソフトウェアでポーリングすることで、どのタイムスロットデータが最後に受信され、どのタイムスロットデータが TXBUF レジスタに転送されるべきかを決定することができます。

## 22.4.16 デジタルループバックモード

デジタルループバックモードは、DLOOP 制御ビット (DCIC0N1<11>) をセットすることにより有効になります。DLOOP ビットがセットされると、モジュールは内部で CSD0 信号を CS defense に接続します。CS defense ピン上の実際のデータ入力はデジタルループバックモードでは無視されます。

## 22.4.17 アンダーフローモード制御ビット

送信アンダーフローが発生すると、UNFM 制御ビット (DCICON1<7>) の設定に従って、2 つのうち 1 つのアクションが発生します。UNFM ビットがクリア（デフォルト値）されていると、モジュールは、バッファ位置用の動作タイムスロット間に CSD0 ピンに ‘0’ を送信します。この動作モード中は、DCI モジュールに接続されたコードックデバイスは単にデジタル的な‘静寂’を与えられます。UNFM 制御ビットがセットされていると、モジュールは、バッファ位置に書き込まれた最後のデータを送信します。この動作モードでは、ソフトウェアのオーバーヘッド無く、コードックデバイスに継続してデータ値を送ることができます。

## 22.4.18 データ詰め制御

ほとんどのアプリケーションでは、データ転送は、FS 信号がアクティブになってから 1 シリアルクロックサイクル後に開始されます。これは DCI のデフォルトです。DJST 制御ビット (DCICON2<5>) を設定することにより代わりのデータ配列が選択されます。DJST=1 の時、データ転送は、FS 信号と同じシリアルクロックサイクル間に始まります。

## 22.4.19 DCI モジュール割り込み

DCI モジュール割り込みの頻度は、動作中のタイムスロット (TSCON と RSCON レジスタ) の数、データフレーム長 (WS と COFSG 制御ビット) および BLEN 制御ビットに依存します。割り込みは以下の時に生成されます。

- バッファ長に達した時
- フレーム境界に達した時

バッファメモリ転送は、上記のイベントが発生する度に発生します。バッファメモリ転送は、以前書き込まれた TXBUF の値が送信バッファメモリに転送され、受信バッファメモリ内の新しい受信データが RXBUF レジスタに転送された時として定義されます。

## 22.5 DCI モジュールを使用する

この章では、特定のデータ変換器と共に、DCI をどのように構成して使用するかを説明します。

### 22.5.1 DCI バッファ、ステータスビットおよび割り込みを用いてのデータの送信 / 受信の仕方

DCI は、BLEN 制御ビットの設定に依存しますが、CPU 割り込みの間に最大 4 つのデータワードをバッファできます。バッファされたデータは、TSCON と RSCON レジスターの設定に依存しますが、1 つのデータフレーム内もしくは複数のデータフレーム内で送信されるか受信されます。例えば、 $BLEN<1:0> = 00b$ ( 割り込みあたり 1 データワードをバッファする ) および  $TSCON=RSCON = 0x0001$  と仮定しましょう。この特定の構成は一番基本的な設定を示し、DCI がすべてのデータフレームの開始時点で 1 データワードを送信 / 受信します。CPU では、 $BLEN<1:0> = 00b$  なので毎回のデータワードが送信 / 受信された後で、割り込みが発生します。

2 番目の構成例では、 $BLEN<1:0> = 11b$ ( 割り込みあたり 4 データワードをバッファする ) および  $TSCON=RSCON = 0x0001$  と仮定しましょう。この構成では、DCI がすべてのデータフレーム開始時点で 1 データワードを送信 / 受信しますが、CPU 割り込みは 4 つのデータワードが送信 / 受信された後に発生します。この構成は、複数データサンプルが一度に処理されるような、ロックプロセッシング用に役に立ちます。

3 番目の構成例では、 $BLEN<1:0> = 11b$ ( 割り込みあたり 4 データワードをバッファする ) および  $TSCON=RSCON = 0x000F$  と仮定しましょう。この構成では、DCI がすべてのデータフレーム開始時点で 4 データワードを送信 / 受信します。この場合、DCI は 1 データフレーム内で 4 つのデータワードをバッファするように設定されたので、CPU 割り込みはデータフレーム毎に発生します。この構成は典型的な複数チャネルバッファリングの設定を示します。

DCI は 4 データワード以上バッファするようにも構成できます。例えば、 $BLEN<1:0> = 11b$ ( 割り込みあたり 4 データワードをバッファする ) および  $TSCON=RSCON = 0x00FF$  と仮定しましょう。この構成では、DCI がすべてのデータフレーム当たり 8 データワードを送信 / 受信します。割り込みはデータフレーム当たり 2 回発生します。それぞれの割り込みで、送信 / 受信レジスタ内に、データのどの部分があるかを決定するために、ユーザーは、現データフレーム位置を決定するための割り込みサービスルーチン内の SLOT ステータスビット (DCISTAT <11:7>) をチェックする必要があります。

送信 / 受信レジスタは二重にバッファされていますので、ユーザーソフトウェアがあるデータのセットを処理している間に、DCI モジュールは送信 / 受信データの別のセットに対して動作することができます。二重バッファ構造のため、データを受信し、そのデータを処理し、処理したデータを送信するために 3 つの割り込み期間が必要です。それぞれの DCI 割り込みに対して、CPU は、前の割り込み期間中に受信したデータワードを処理し、次の割り込み期間に送信されるべきデータワードを生成します。dsPIC デバイスがバッファリングおよびデータ処理時間のために、処理されるデータに対して、2 つの割り込み期間遅延を挿入します。このデータ遅延は多くの場合、無視できます。

DCI ステータスフラグと CPU 割り込みは、バッファ転送が発生し、CPU がさらにデータを処理する時であることを示します。典型的なアプリケーションでは、DCI データが処理されるたびに以下のステップが発生します。

1. RXBUF レジスタがユーザーソフトウェアにより読み込まれます。受信レジスタに新しいデータがあることを示すために、モジュールにより RFUL ステータスビット (DCISTAT<2>) がセットされます。RFUL ビットは、すべての有効な受信レジスタが読み込まれた後で自動的にクリアされます。
2. ユーザーソフトウェアは受信データを処理します。
3. 処理されたデータは TXBUF ジスタに書き込まれます。送信レジスタが更なるデータの書き込みの準備ができたことを示すために、TMPTY ステータスビット (DCISTAT<0>) が前もってセットされます。

データの送信 / 受信 (TSCON と RSCON が非ゼロ) を行うように構成されるアプリケーションでは、RFUL と TMPTY ステータスビットは、DCI バッファ転送が発生する時を決定するため、ユーザーソフトウェアでポーリングできます。DCI がデータの送信を行うためにのみ使用される場合 (RSCON = 0)、TMPTY ビットがバッファ転送を示すためにポーリングできます。DCI がデータの受信のみを行うために構成される場合 (TSCON = 0) は、RFUL ビットがバッファ転送を示すためにポーリングできます。

DCIIF ステータスビット (IFS2<9>) は、DCI バッファ転送が発生し CPU 割り込み（もし有効であれば）が発生するたびにセットされます。DCIIF ステータスビットは、RFUL と TMPTY ステータスビットの論理和を取ることで生成されます。

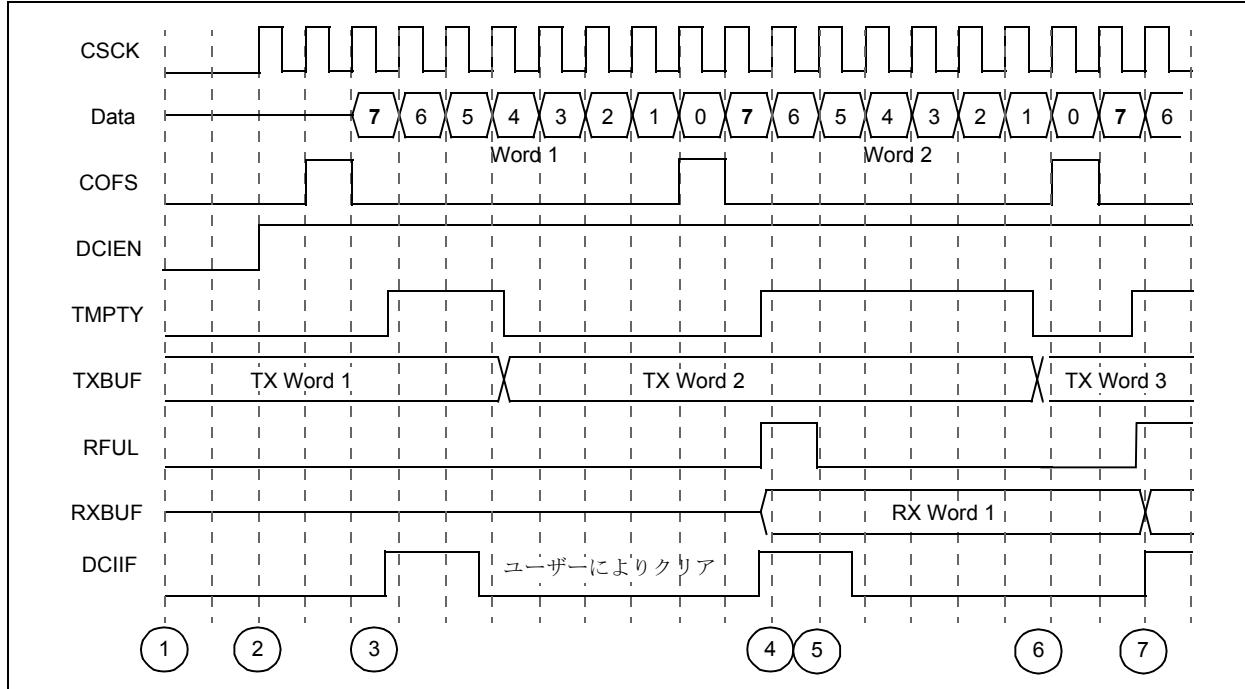
### 22.5.1.1 DCI のスタートアップとデータバッファリング

データ転送は、DCIEN 制御ビット (DCICON1<15>) を設定することで開始されます。この前に、DCI 制御レジスタは所望の動作モード用に初期化される必要があります。（セクション 22.5.4「複数チャネル動作」、セクション 22.5.5「I<sub>2</sub>S動作」、セクション 22.5.6「AC-Link動作」。）

DCI スタートアップのタイミング図を図 22-7 に示します。この例では、DCI は 8- ビットデータワード (WS<3:0> = 0111b) および 8- ビットデータフレーム (COFS<3:0> = 0000b) に構成されています。複数チャネルモード (COFSM<1:0> = 00b) が使用されています。送信 / 受信に必要な手順を以下に示します。

1. TXBUF レジスタは、モジュールが有効になる前に、送信される最初のデータが転送されている必要があります。送信データがコーデックから受信されるデータをベースにする場合は、ユーザーは単に TXBUF レジスタをクリアするだけです。これにより、コーデックから RXBUF レジスタに最初にデータが受信されるまで、デジタル的な‘静寂’を送信します。
2. DCIEN ビット (DCICON1<15>) を設定することにより DCI モジュールを有効にします。DCI がマスターデバイスの場合、TXBUF レジスタ内のデータは、送信バッファに転送され最初のデータフレームの送信が開始されます。そうでなければ、TXBUF データは、マスターデバイスからフレーム同期信号が受信されるまで、送信バッファに保持されます。
3. TMPTY ビットは、モジュールが有効にされると直ぐにセットされ、DCI 割り込みが発生します。この時、TXBU レジスタを再度クリアするモジュールは、第 2 データフレームで転送されるデータが TXBUF レジスタに転送される準備ができています。この時はモジュールによるデータ受信はできませんので、送信データが受信データから演算されるのであれば、TXBUF レジスタを再度クリアする必要があります。DCIIF ステータスピットは、割り込みが有効な場合は、ユーザーがソフトウェアでクリアする必要があります。
4. 最初のデータが転送された後、TMPTY ビットがセットされ、RFUL ステータスピットがセットされ、もし有効であれば DCI 割り込みが発生します。これが、DCI に接続されたデバイスから受信される最初のデータワードになります。
5. ユーザーは受信レジスタを読み出し、自動的に RFUL ステータスピットをクリアします。ユーザー ソフトウェアは今度は受信データを処理します。
6. 送信レジスタには、次のデータフレーム期間で送信されるべきデータを書き込みます。TMPTY ステータスピットは、書き込みが発生したら自動的にクリアされます。書き込みデータは、前の割り込みで受信されたデータから演算されます。
7. 次の DCI 割り込みが発生し、以降このサイクルが繰り返されます。

図 22-7: DCI のスタートアップとデータバッファリング例



## 22.5.1.2 DCI の無効化

DCI モジュールは、DCIEN 制御ビット (DCICON1<15>) をクリアすることで無効になります。DCIEN ビットがクリアされると、モジュールは処理中の現データフレーム転送を終了します。送信 / 受信バッファが、フレーム終了前に書き込み / 読み出しある必要がある場合は、割り込みが発生します。

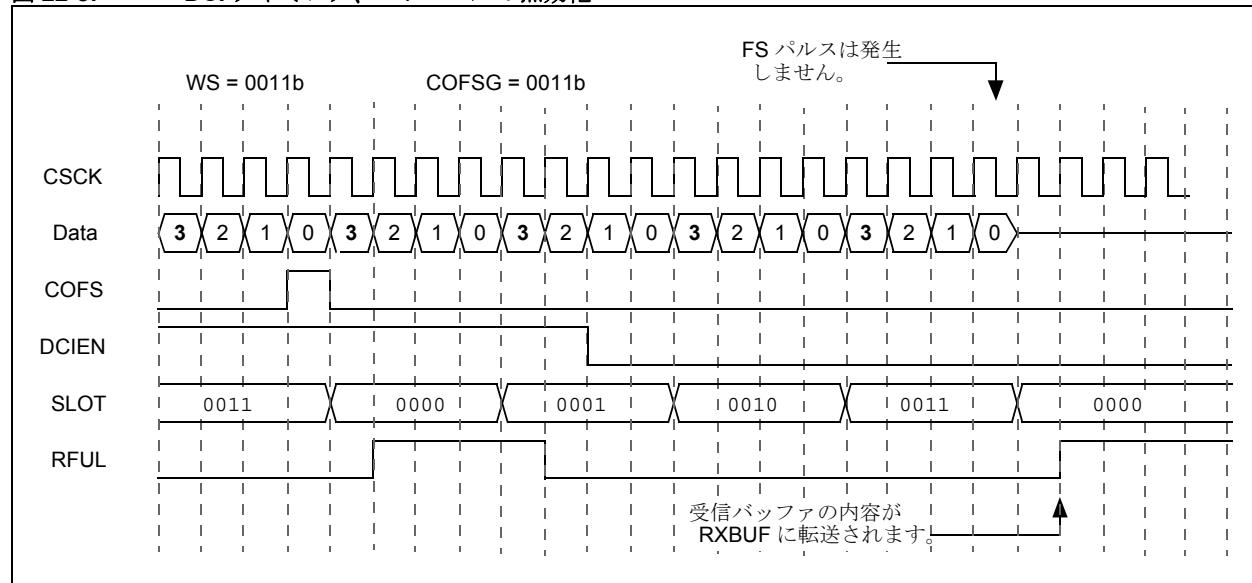
DCIEN ビットは、当該フレームでモジュールを無効にするためには、フレームの終わりより少なくとも 3 CSCK サイクル前にクリアされる必要があります。そうでなければ、モジュールは次のフレームで無効になります。

DCI は、DCIEN ビットがクリアされた後は、フレーム同期パルスを生成しませんし、入力フレーム同期パルスにも反応しません。

フレーム同期ジェネレータが、データフレームの最後のタイムスロットに達したら、DCI に関連するすべてのステートマシンはリセットされアイドル状態になり、モジュールに関連する I/O ピンの制御は開放されます。DCIEN ビットがクリアされた後 SLOT<3:0> ステータスピット (DCISTAT<11:7>) をポーリングすることで、ユーザーはモジュールがアイドルである時を知ることができます。DCI は SLOT<3:0> = 0000b かつ DCIEN = 0 の時アイドルになります。

モジュールがアイドルステートに入ると、受信シャドウレジスタ内のデータはすべて RXBUF レジスタに転送され、RFUL と ROV ステータスピットは、その結果として影響を受けます。

図 22-8: DCI タイミング、モジュールの無効化



## 22.5.2 マスター動作対スレーブ動作

DCI はマスターもしくはスレーブ動作に設定可能です。マスターデバイスは、データ転送を起動するためのフレーム同期信号を生成します。動作モード(マスターもしくはスレーブ)は、COFS 制御ビット(DCICON1<8>)により選択されます。

DCI モジュールがマスターデバイスとして動作する場合(COFS=0)は、COFSM モードビットが、フレーム同期ジェネレータで生成されるフレーム同期パルスの種類を決定します。新しいフレーム同期信号は、フレーム同期ジェネレータがリセットされた時に生成され、COFS ピンに出力されます。

DCI モジュールがフレーム同期スレーブとして動作する場合(COFS=1)は、データ転送は DCI モジュールに接続されたデバイスにより制御されます。COFSM 制御ビットは、DCI モジュールが入力 FS 信号に対してどのように応答するかを制御します。

複数チャネルモードの場合、新しいデータフレーム転送は、COFS ピンが High となった後 1シリアルクロック後に開始されます。COFS ピンへのパルスにより、フレーム同期ジェネレータがリセットされます。

I<sup>2</sup>S モードでは、新しいデータワードは、COFS ピンで Low から High または High から low への遷移が発生してから 1シリアルクロック後に転送されます。COFS ピンの立ち上がりもしくは立下りエッジで、フレーム同期ジェネレータはリセットされます。

AC-Link モードでは、タグスロットと次のフレーム用の引き続くデータスロットが、COFS ピンが High になった後 1シリアルクロック後に転送されます。

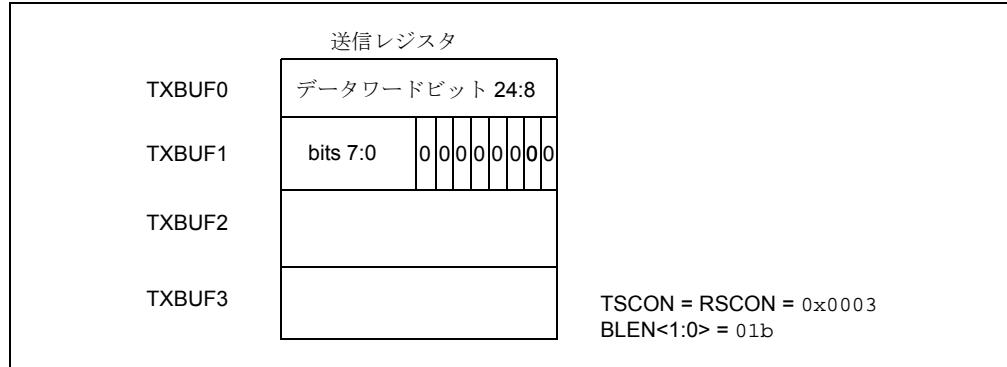
COFSG と WS ビットは、モジュールがスレーブモードで動作している時には、期待されたフレーム長を与えるように構成される必要があります。一旦有効なフレーム同期パルスが、モジュールによって COFS ピンでサンプリングされると、データフレーム転送全体が始まります。モジュールは、現データフレームの転送が完全に終了するまで、さらなるフレーム同期パルスに対しては応答しません。

## 22.5.3 ロングデータワードをサポートするデータパッキング

多くのコーデックは、16 ビットを越える長さのデータワード長を持っています。DCI は本来、最大 16 ビットのワード長までサポートしますが、複数の送信 / 受信スロットを有効にし、データを複数の送信 / 受信バッファにパッキングすることにより、より長いワード長もサポートできます。例えば、特定のコーデックが 24 ビットデータワードを送受信すると仮定しましょう。このデータは、BLEN<1:0> = 01b(割り込み当たり 2 データワード)かつ TSCON = RSCON = 0x0003 と設定することで、送信 / 受信が可能になります。これにより、データフレームの最初の 2 タイムスロット間で送受信が可能になります。送信データの上位 16 ビットは TXBUFO に書き込まれます。送信データの下位 8 ビットは、図 22-9 に示すように左詰めされて TXBUF1 に書き込まれます。TXBUF1 の下位 8 ビットには ‘0’ が書き込まれます。コーデックから受信された 24 ビットデータは、送信データと同じフォーマットで RXBUFO と RXBUF1 に転送されます。

ワードサイズと有効にされたタイムスロットのどのような組合せも、複数の送信 / 受信レジスタ内では、ロングデータワードを送信 / 受信するために使うことができます。例えば、図 22-9 に示されるような 24 ビットデータワードの例では、WS<3:0> = 0111(ワードサイズ 8 ビット)、BLEN<1:0> = 10(割り込み間で 3 ワードをバッファ)、TSCON = RSCON = 0x0007(データフレームの最初の 3 タイムスロット間に送信 / 受信)と設定することにより、3 つの連続したレジスタ内で送信 / 受信を行うことができます。それぞれの送信 / 受信レジスタは、8 ビットのデータワードを含みます。

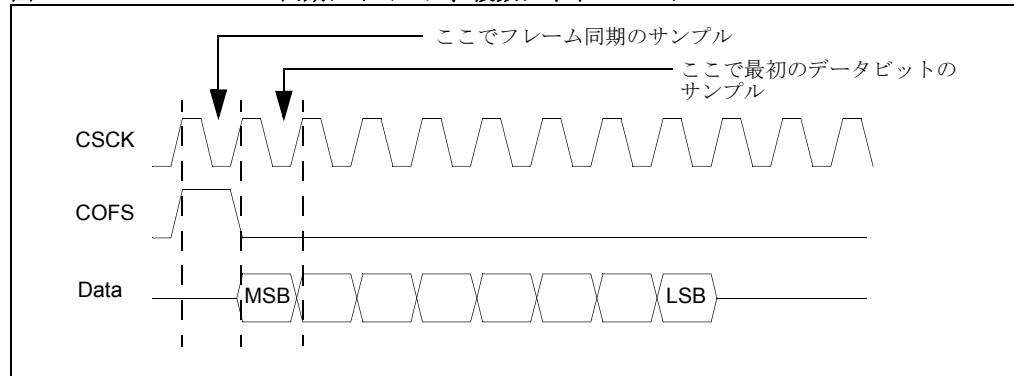
図 22-9: ロングデータワード時のデータパッキング例



### 22.5.4 複数チャネル動作

複数チャネルモード ( $COFSM<1:0> = 00$ ) は、データ転送を起動するよりも 1 シリアルクロック期間前に High に駆動されるフレーム同期パルスを必要とするコーデックに使用されます。データフレーム内で 1 つ以上のデータワードが転送できます。連続するフレーム同期パルス間のクロックサイクル数は、DCI モジュールに接続されたデバイスに依存します。複数チャネルモードにおけるフレーム同期信号のタイミング図を図 22-10 に示します。4 ワードデータ転送を示すタイミング例も、図 22-2 に示されます。

図 22-10: フレーム同期タイミング、複数チャネルモード



#### 22.5.4.1 複数チャネルセットアップの詳細

複数チャネルモードを使用するコーデック用に DCI を構成するために必要な手順を、この章で示します。この動作モードは、1 つ以上のデータチャネルを持つコーデックに用いられます。その設定は、チャネルの数に関わらず同様です。

このセットアップ例では、仮想のコーデックが想定されています。この設定例用として使用される単チャネルコーデックでは、それぞれのフレームの始めで 16 ビットデータワードが転送される 256 fs のシリアルクロック周波数を使用します。

セットアップと動作に必要な手順は以下のようになります。

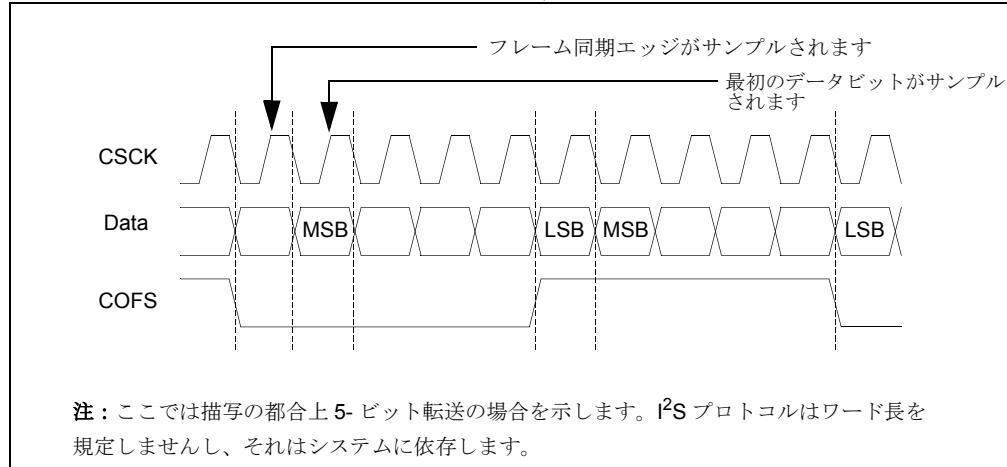
1. コーデックで要求されるサンプリングレートとデータワードサイズを決定します。この例では、8KHz サンプリングレートが仮定されています。
2. コーデックで要求されるシリアル転送クロック周波数を決定します。ほとんどのコーデックは、いくつか複数のサンプリング周波数のシリアルクロック信号を必要とします。この例では、256 fs もしくは 1.024MHz の周波数を必要とします。従って、フレーム同期パルスは、データ転送を開始させるために、256 シリアルクロックサイクル毎に生成される必要があります。
3. DCI は、シリアル転送クロック用に構成される必要があります。CSCK 信号が DCI で生成される場合は、CSCKD 制御ビット ( $DCICON1<10>$ ) をクリアし、正しいクロック周波数を生成する値を  $DCICON3$  に書き込みます(セクション 22.4.3 「ビットクロックジェネレータ」を参照してください。) CSCK 信号がコーデックもしくは他の外部ソースで生成される場合は、CSCKD 制御ビットをセットし、 $DCICON3$  レジスタをクリアします。
4.  $COFSM<1:0>$  制御ビット ( $DCICON1<1:0>$ ) をクリアして、複数チャネルモードのフレーム同期信号を設定します。
5. DCI がフレーム同期信号を生成する(マスター)場合、COFSD 制御ビット ( $DCICON1<8>$ ) をクリアします。DCI がフレーム同期信号を受信する(スレーブ)場合、COFSD 制御ビットをセットします。
6. CSCKE 制御ビット ( $DCICON1<9>$ ) をクリアし、CSCK の立下りエッジで入力データをサンプリングします。これはほとんどのコーデックでの典型的な構成です。正しいサンプリングエッジの使用を確認するには、コーデックのデータシートを参照してください。
7. 所望のデータワードサイズを決めるには  $WS<3:0>$  制御ビット ( $DCICON2<3:0>$ ) を書き込みます。本例のコーデックでは、16 ビットデータワードサイズなので  $WS<3:0> = 1111b$  とする必要があります。

8. COFSG<3:0> 制御ビット (DCICON2<8:5>) に、フレームあたりの所望のデータ数を書き込みます。WS と COFSG 制御ビットは、CSCK サイクル内のデータフレーム長を決定します。( セクション 22.4.7 「フレーム同期生成」を参照してください。) この例のコーデックで必要とされる 256 ビットデータフレームを供給するために、COFSG<3:0> = 1111b と設定します。
9. CSDOM 制御ビット (DCIC0N1<6>) を用いて CSDO ピンの出力モードを設定します。DCI に 1 つだけのデバイスが接続される場合、CSDOM をクリアします。これにより CSDO ピンは、未使用のデータスロット間は ‘0’ になります。CSDO ピンに複数のデバイスが接続されている場合は、CSDOM をセットする必要があります。
10. TSCON および RSCON レジスタに書き込み、フレーム内のどのデータタイムスロットがそれぞれ送信および受信されるかを決定します。この単チャネルコーデックの場合、TSCON = RSCON = 0x0001 を用い、データフレームの最初の 16 ビットタイムスロットの間に送信および受信ができるようにします。
11. BLEN 制御ビット (DCICON2<11:10>) を設定し、所望の量のデータワードのバッファを行います。この単チャネルコーデックでは、BLEN = 00 を設定することで、毎データフレームあたり割り込みを生成します。BLEN により大きい値を与えると、このコーデックでは、割り込み間で複数サンプルをバッファすることができます。
12. 割り込みが使用される場合は、DCIIF ステータスピット (IFS2<9>) をクリアし、DCIE 制御ビット (IEC2<9>) をセットします。
13. セクション 22.5.1.1 「DCI のスタートアップとデータバッファリング」で記載されているように動作を開始します。

## 22.5.5 I<sup>2</sup>S 動作

I<sup>2</sup>S 動作モードは、50% デューティーサイクルを持つフレーム同期信号を必要とするコーデックに用いられます。シリアルクロック内の I<sup>2</sup>S フレーム同期信号の期間は、DCI モジュールに接続されるコーデックのワードサイズにより決定されます。新しいワード境界の開始は、図 22-11 に示すように、COFS ピンの High から Low または Low から High への遷移エッジにより区分けされます。I<sup>2</sup>S コーデックは通常ステレオもしくは 2 チャネルデバイスで、1 つのデータワードはフレーム同期信号の Low の期間で転送され、もう一方のデータワードは High の期間で転送されます。

図 22-11: I<sup>2</sup>S インターフェースフレーム同期タイミング



DCI モジュールは、DCICON1 SFR 内の COFSM<1:0> 制御ビットに 01h の値を書き込むことで I<sup>2</sup>S モードに構成されます。I<sup>2</sup>S モードで動作中は、DCI モジュールは 50% デューティーサイクルを持つフレーム同期信号を生成します。フレーム同期信号のそれぞれのエッジは、新しいデータワード転送の境界を区分けします。I<sup>2</sup>S プロトコルに関する詳しい情報はこのマニュアルの付録を参照してください。ユーザーは、DCICON2 SFR 内の COFSG と WS 制御ビットを用いて、フレーム長とデータワードサイズも選択する必要があります。

### 22.5.5.1 I<sup>2</sup>S 設定の詳細

本章では、I<sup>2</sup>S コーデック用に DCI を構成するために必要な手順を示します。この設定例では、仮想的な I<sup>2</sup>S コーデックを想定します。

本設定例での I<sup>2</sup>S コーデックは、データフレームあたり 2 つの 16 ビットデータワードを持ち、64fs シリアルクロック周波数を使用します。従って、フレーム長は 64CSCK サイクルになり、32 サイクルの High と 32 サイクルの Low を持つ COFS 信号を持ちます。図 22-11 に示されるように、最初のデータワードは COFS の立ち下りエッジの 1CSCK サイクル後に送信され、2 番目のデータワードは COFS の立ち上がりエッジの 1CSCK サイクル後に送信されます。

1. CSCK 周波数を決定するためにコーデックで使用されるサンプリングレートを決定します。本例では、fs は 48KHz と仮定します。
2. コーデックで必要なシリアル転送クロック周波数を決定します。本例でのコーデックでは、64fs もしくは 3.072MHz を必要とします。
3. DCI は、シリアル転送クロック用に構成される必要があります。CSCK 信号が DCI で生成される場合は、CSCKD 制御ビット (DCICON1<10>) をクリアし、正しいクロック周波数を生成する値を DCICON3 に書き込みます(セクション 22.4.3 「ビットクロックジェネレータ」を参照してください。) CSCK 信号がコーデックもしくは他の外部ソースで生成される場合は、CSCKD 制御ビットをセットし、DCICON3 レジスタをクリアします。
4. 次に COFSM<1:0> = 01b を設定し、フレーム同期信号を I<sup>2</sup>S モードに設定します。
5. DCI がフレーム同期信号を生成する(マスター)場合、COFSD 制御ビット (DCICON1<8>) をクリアします。DCI がフレーム同期信号を受信する(スレーブ)場合、COFSD 制御ビットをセットします。
6. CSCKE 制御ビット (DCICON1<9>) をセットし、CSCK の立ち上がりエッジで入力データをサンプリングします。これはほとんどの I<sup>2</sup>S コーデックでの典型的な構成です。
7. 所望のデータワードサイズを決めるには WS<3:0> 制御ビット (DCICON2<3:0>) を書き込みます。本例のコーデックでは、16 ビットデータワードサイズ用として WS<3:0> = 1111b を使用します。
8. COFSG<3:0> 制御ビット (DCICON2<8:5>) に、フレームあたりの所望のデータ数を書き込みます。WS と COFSG 制御ビットは、CSCK サイクル内のデータフレーム長を決定します。( 22.4.7 「フレーム同期生成」を参照してください。) 本例のコーデックでは、COFSG<3:0> = 0001b を設定します。

**注:** I<sup>2</sup>S モードでは、COFSG ビットは、データフレームの 1/2 の長さに設定します。本例のコーデックでは、32 ビットフレームを生成するために、COFSG<3:0> = 0001b(フレームあたり 2 データワード)を設定します。これにより 64 ビットの長さを持つ I<sup>2</sup>S データフレームを生成します。

9. CSDOM 制御ビット (DCICON1<6>) を用いて CSDO ピンの出力モードを設定します。DCI に 1 つだけのデバイスが接続される場合、CSDOM をクリアします。CSDO ピンに複数のデバイスが接続されている場合は、CSDOM をセットする必要があります。
10. TSCON および RSCON レジスタに書き込み、フレーム内のどのデータタイムスロットがそれぞれ送信および受信されるかを決定します。このコーデックの場合、TSCON = 0x0001 および RSCON = 0x0001 を設定し、32 ビットデータフレームの最初の 16 ビットタイムスロットの間に送信および受信ができるようにします。隣接するタイムスロットは、16 ビットより長いデータワードをバッファする場合に有効にします。
11. BLEN<1:0> 制御ビット (DCICON2<11:10>) を設定し、所望の量のデータワードのバッファを行います。2 チャネル I<sup>2</sup>S コーデックでは、BLEN<1:0> = 01b を設定することで、2 データワードの転送後割り込みを生成します。
12. 割り込みが使用される場合は、DCIIF ステータスピット (IFS2<9>) をクリアし、DCIIE 制御ビット (IEC2<9>) をセットします。
13. セクション 22.5.1.1 「DCI のスタートアップとデータバッファリング」で記載されているように動作を開始します。I<sup>2</sup>S マスター モードでは、COFS ピンはモジュールが有効になった後 High に駆動され、TXBUFO 内に転送されたデータの送信が開始されます。

### 22.5.5.2 I<sup>2</sup>S チャネル配列の決定方法

ほとんどの I<sup>2</sup>S コーデックは、2 チャネルデータをサポートし、フレーム同期信号のレベルは、データフレームの半分の期間で転送されるチャネルを示しています。DCI 割り込みサービスルーチン内でピンの現在のレベルを知るために、対応するポートレジスタを用いて、ソフトウェアで COFS ピンをポーリングできます。これにより、どのデータが受信レジスタにあるか、次のフレームでの転送用としてどのデータが送信レジスタに書き込まれるべきかを知ることができます。

### 22.5.5.3 I<sup>2</sup>S データ詰め

I<sup>2</sup>S 規格によれば、データワード転送は、デフォルトでは、フレーム同期信号の遷移後 1 シリアルクロックサイクルで開始します。「上位ビット左詰め」オプションは、DJST 制御ビット (DCIC0N1<5>) を用いて選択されます。

DJST = 1 の場合、I<sup>2</sup>S データ転送は上位ビット左詰めで行われます。データの上位ビットは、FS 信号の立ち上がりもしくは立下りエッジで同じシリアルクロックサイクルの間に CSDO ピンに表れます。CSDO ピンは、データワードが送付された後はトライステートになります。

左詰めデータオプションにより、2 つのステレオコーデックが同じシリアルバスに接続できます。多くの I<sup>2</sup>S 準拠デバイスは、左詰めもしくは右詰めの構成オプションを持っています。ワードサイズ選択ビットでコーデックワード長を 2 回設定し、データを、パック化された形で DCI メモリに読み書きするようにすることができます。2 つの I<sup>2</sup>S コーデックシステム用の接続詳細を図 22-12 に示します。

I<sup>2</sup>S モード用タイミング図を図 22-13 に示します。参考として、これらの図では 8- ビットワードサイズ ( $WS<3:0> = 0111b$ ) を仮定しています。16 ビットサブフレーム ( $COFSG<3:0> = 0001b$ ) を達成するには、フレームあたり 2 つのデータワードが必要です。図 22-13 の 3 番目のタイミング図では、2 つのコーデックからのパック化されたデータの読み書きを使用しています。この例では、DCI モジュールは 16 ビットデータワード ( $WS<3:0> = 1111b$ ) で構成されています。2 つのパケット化された 8 ビットワードは、各々 DCI メモリバッファ内の 16 ビットバッファに書き込まれます。

図 22-12: 2 つの I<sup>2</sup>S コーデックインターフェース

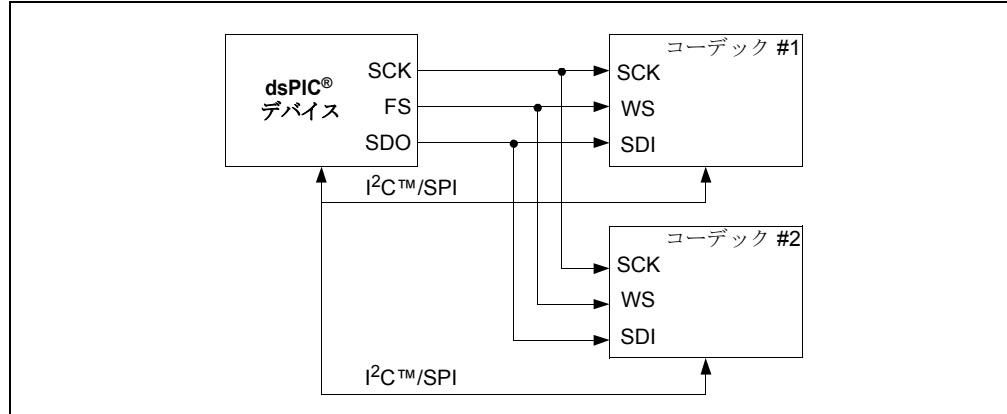
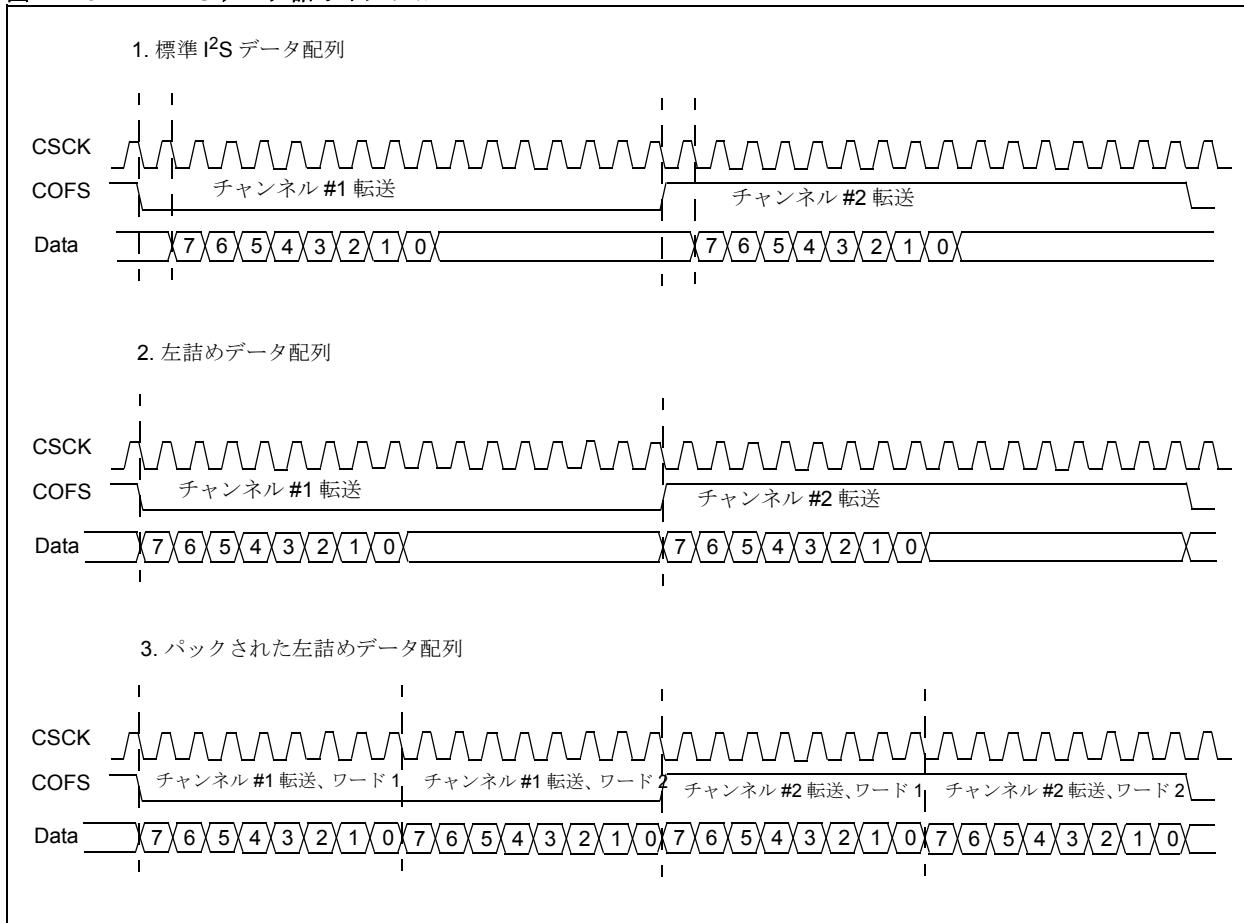


図 22-13: I<sup>2</sup>S データ詰めオプション

## 22.5.6 AC-Link 動作

この章では、AC-Link モードでの DCI の使い方を示します。AC-Link モードは、AC-'97 準拠のコードックデバイスと通信する場合に使用されます。

### 22.5.6.1 AC-Link データフレーム

AC-Link データフレームは 256 ビットで、中味は 1 つの 16 ビット制御スロットに 12 の 20 ビットデータスロットが続く構成に分割されています。AC'97 コードックは通常、図 22-14 に示されるように、水晶発振子によるシリアル転送クロックを備えています。コントローラはシリアルクロックを受信し、フレーム同期信号を生成します。デフォルトのデータフレームレートは 48KHz です。AC-Link システムで使用されるフレーム同期信号は、データフレームの最初の 16CSCK 期間が High で 240CSCK 期間が Low です。図 22-16 に示すようにフレーム同期信号の立ち上がりエッジから 1CSCK 後にデータ転送が開始されます。データは受信デバイス側で CSCK の立下りエッジでサンプリングされます。AC-Link 制御とデータタイムスロットは、図 22-15 に示すように、使い方がプロトコルで定義されています。AC-Link プロトコルの完全定義については、このマニュアルの付録もしくはインテル® AC'97 コードック仕様書 Rev2.2 を参照してください。

図 22-14: AC-リンク信号の接続

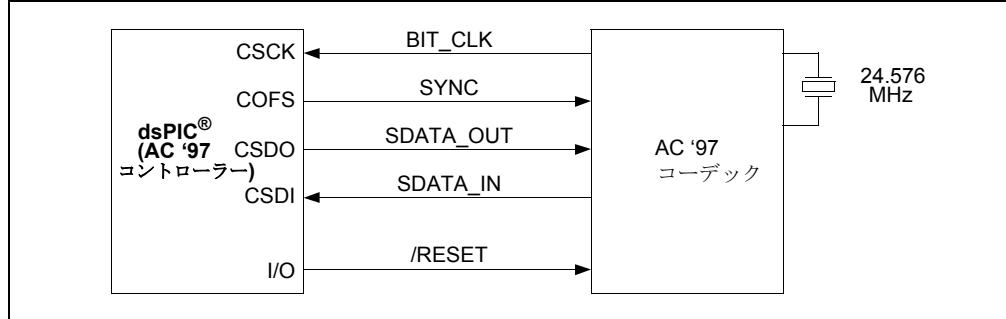


図 22-15: AC-Link データフレーム

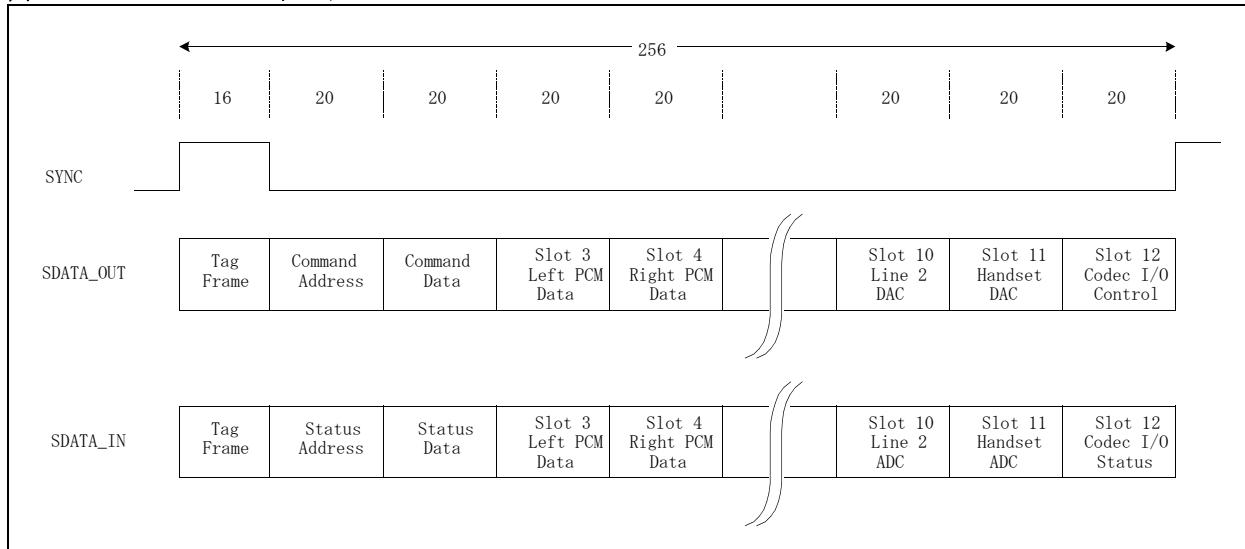
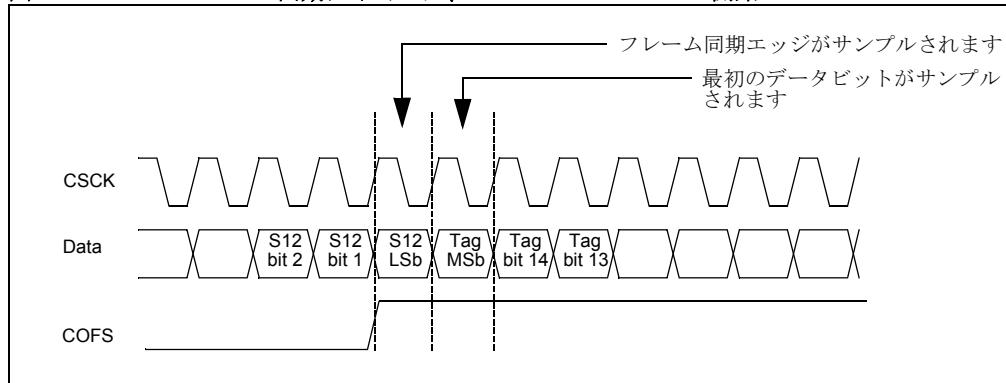


図 22-16: フレーム同期タイミング、AC-Link のフレームの開始



DCI モジュールは、AC-Link プロトコルで定義された、20 ビットデータタイムスロットを活用する 2 つの動作モードを持っています。この動作モードは COFSM<1:0> 制御ビット (DCICON1<1:0>) で選択されます。一つめの AC-Link モードは ‘16 ビット AC-Link モード’ と呼ばれ COFSM<1:0> = 10b を設定することで選択されます。二つめの AC-Link モードは ‘20 ビット AC-Link モード’ と呼ばれ、COFSM<1:0> = 11b を設定することで選択されます。

### 22.5.6.2 16 ビット AC-Link モード

16 ビット AC-Link モードでは、送信 / 受信データワードの長さは、DCI 送信 / 受信レジスタに適合させるために 16 ビットに制限されます。この制限は AC-Link プロトコルの 20 ビットデータに影響を与えるのみであることに注意してください。受信タイムスロットでは、入力データは 16 ビットで打ち切られます。出力タイムスロットでは、データワードの下位 4 ビットは、モジュールにより ‘0’ にセットされます。この動作モードにより、すべてのタイムスロットを 16 ビットタイムスロットとして扱うことにより AC-Link データフレームを簡単にしています。フレーム同期ジェネレータがタイムスロット境界を揃えるようにします。

### 22.5.6.3 20 ビット AC-Link モード

20 ビット AC-Link モードではデータタイムスロット内のすべてのビットが送信 / 受信されますが、AC-Link プロトコルで定義される特定タイムスロット境界にデータ配列を揃えることはできません。

20 ビット AC-Link モードは、機能的には DCI モジュールの複数チャネルモードに似ていますが、生成されるフレーム同期信号のデューティーサイクルが違います。AC-Link フレーム同期信号は 16 サイクルの High 期間と引き続く 240 サイクルの Low 期間を保持します。

20 ビットモードは一つの 256 ビット AC-Link フレームを 16 の 16 ビットタイムスロットとして扱います。20 ビット AC-Link モードでは、モジュールは  $COFSG<3:0> = 1111b$  かつ  $WS<3:0> = 1111b$  として動作します。20 ビットデータスロットのデータ配列はこの動作モードでは保持されません。例えば、256 ビットの AC-Link データフレーム全体は、TSCON と RSCON レジスタ内のすべてのビットを設定することによりパック化された形で送信 / 受信されます。トータル利用バッファ長は 64 ビットですので、AC-Link フレームを転送するためには 4 回の連続する割り込みが必要です。アプリケーションソフトウェアにより、 $SLOT<3:0>$  ステータスビット ( $DCISTAT<11:7>$ ) をモニターすることにより現 AC-Link フレームセグメントを追跡する必要があります。

### 22.5.6.4 AC-Link の設定の詳細

10h もしくは 11h を、DCICON1 SFR 内の  $COFSM<1:0>$  制御ビットに書き込むことにより、モジュールを AC-Link モード用に構成できます。ワードサイズ選択ビット ( $WS<3:0>$ ) とフレーム同期信号生成ビット ( $COFSG<3:0>$ ) は、フレームとワードサイズはプロトコルで設定されるので、16 と 20 ビット AC-Link モードいずれの場合も影響を与えません。

ほとんどの AC'97 コーデックは、データ転送を制御するクロック信号を生成します。従って、CSCKD 制御ビットはソフトウェアでセットされます。COFSD 制御ビットは、入力クロック信号から DCI が FS 信号を生成するので、クリアします。CSCKE ビットは、データが立ち上がりエッジでサンプルされるように、クリアします。

ユーザーは AC-Link データフレーム内のどのタイムスロットがバッファされるべきかを決定し、そうなるように TSE と RSE 制御ビットを設定する必要があります。最低限、送信 / 受信 TAG スロットをバッファする必要がありますので、TSCON<0> と RSCON<1> 制御ビットはソフトウェアでセットする必要があります。

**注：** AC-Link フレームは 13 のタイムスロットを持ちますので、TSCON<12:0> 制御ビットと RSCON<12:0> 制御ビットのみが 16 ビット AC-Link モードで効果があります。

1. DCI は、AC'97 コーデックからのシリアル転送クロックを受信するように構成する必要があります。CSCKD 制御ビットをセットし、DCICON3 レジスタをクリアします。
2. 次に、 $COFSM<1:0>$  制御ビット (DCICON1<1:0>) を 10b もしくは 11b に設定し所望の AC-Link フレーム同期モードに設定します。
3. COFSD 制御ビット (DCICON1<8>) をクリアすると、DCI がフレーム同期信号を出力します。
4. CSCKE 制御ビット (DCICON1<9>) をクリアし、CSCK の立下りエッジで入力データをサンプリングするようにします。

**注：** フレームとワードサイズはプロトコルで設定されますので、ワードサイズ選択ビット ( $WS<3:0>$ ) とフレーム同期生成ビット ( $COFSG<3:0>$ ) は、16 および 20 ビット AC-Link モードには影響を与えません。

5. CSDOM 制御ビット (DCICON1<6>) をクリアします。
6. TSCON および RSCON レジスタに書き込み、フレーム内のどのデータタイムスロットがそれぞれ送信および受信されるかを決定します。これは、AC-Link プロトコルでどのタイムスロットが使用されるかに依存します。最低限、スロット # 0(タグスロット)での通信が必要です。追加の情報については **セクション 22.5.6.2 「16 ビット AC-Link モード」**、**セクション 22.5.6.3 「20 ビット AC-Link モード」** およびこのマニュアルの付録での議論を参照してください。
7. BLEN 制御ビット (DCICON2<11:10>) を設定し、所望の量のデータワードのバッファを行います。単チャネルコーデックでは、BLEN = 00 を設定することで、毎データフレームごとに割り込みを生成します。BLEN により大きい値を与えると、このコーデックでは、割り込み間で複数サンプルをバッファすることができます。
8. 割り込みが使用される場合は、DCIIF ステータスビット (IFS2<9>) をクリアし、DCIIIE 制御ビット (IEC2<9>) をセットします。
9. セクション 22.5.1.1 「DCI のスタートアップとデータバッファリング」で記載されているように動作を開始します。

## 22.6 省電力モードでの動作

### 22.6.1 CPU アイドルモード

DCI モジュールは、CPU がアイドルモード中でも動作を継続させることができます。DCISIDL 制御ビット (DCICON1<13>) は、CPU がアイドルモードの時に DCI モジュールを動作させるかどうかを決定します。DCISIDL 制御ビットがクリアされている場合 (デフォルト)、モジュールはアイドルモードでも通常動作を継続します。DCISIL ビットがセットされている場合、モジュールは CPU がアイドルモードに入ると停止します。

### 22.6.2 スリープモード

CSCK がデバイス命令クロック TCY から駆動されている場合は、デバイスがスリープモードに入ると DCI は動作しません。

しかし、DCI モジュールは、CSCK 信号が外部デバイスで供給されている場合 (CSCKD = 1)、スリープモードでも動作することができますし、CPU を起動することもできます。スリープモードからの起動イベントを有効にするには、DCI 割り込みビット DCIIIE を設定する必要があります。DCI 割り込み優先レベルが現 CPU 優先度より大きい場合、DCI ISR からプログラムの実行を再開します。そうでなければ、プログラムの実行は、前にスリープモードに入った PWRSAV 命令に引き続く命令で再開されます。

### 22.7 DCI に関するレジスタ

表 22-1 に DCI モジュールに関するレジスタをリストアップします。

表 22-1: DCI レジスタマップ

名前	アドレス	ピット 15	ピット 14	ピット 13	ピット 12	ピット 11	ピット 10	ピット 9	ピット 8	ピット 7	ピット 6	ピット 5	ピット 4	ピット 3	ピット 2	ピット 1	ピット 0	オールリセット時の値
IFS2	0088	—	—	—	FLTBIF	FLTAIF	LVDIF	DCIIF	QEIIIF	PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF	0000 0000 0000 0000
IEC2	0090	—	—	—	FLTBIE	FLTAIE	LVDIE	DCIIE	QEIIIE	PWMIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE	0000 0000 0000 0000
IPC10	00A8	—	FLTAIP<2:0>			—	LVDIP<2:0>			—	DCIIP<2:0>			—	QEIIIP<2:0>			0100 0100 0100 0100
DCICON1	240	DCIEN	—	DCISIDL	—	DLOOP	CSCKD	CSCKE	COFSD	UNFM	SDOM	DJST	—	—	—	COFSM<1:0>		000- -000 000- --00
DCICON2	242	—	—	—	—	BLEN<1:0>		—	COFSG<3:0>			—	WS<3:0>			---- 00-0 000- 0000	---- 0000 0000 0000	
DCICON3	244	—	—	—	—	BCG<11:0>												---- 0000 0000 0000
DCISTAT	246	—	—	—	—	SLOT<3:0>				—	—	—	—	ROV	RFUL	TUNF	TMPTY	---- 0000 ---0 0000
TSCON	248	TSE15	TSE14	TSE13	TSE12	TSE11	TSE10	TSE9	TSE8	TSE7	TSE6	TSE5	TSE4	TSE3	TSE2	TSE1	TSE0	0000 0000 0000 0000
RSCON	24C	RSE15	RSE14	RSE13	RSE12	RSE11	RSE10	RSE9	RSE8	RSE7	RSE6	RSE5	RSE4	RSE3	RSE2	RSE1	RSE0	0000 0000 0000 0000
RXBUFO	250	受信 #0 データレジスタ															aaaaaa aaaaaa aaaaaa aaaaaa	
RXBUF1	252	受信 #1 データレジスタ															aaaaaa aaaaaa aaaaaa aaaaaa	
RXBUF2	254	受信 #2 データレジスタ															aaaaaa aaaaaa aaaaaa aaaaaa	
RXBUF3	256	受信 #3 データレジスタ															aaaaaa aaaaaa aaaaaa aaaaaa	
TXBUFO	258	送信 #0 データレジスタ															0000 0000 0000 0000	
TXBUF1	25A	送信 #1 データレジスタ															0000 0000 0000 0000	
TXBUF2	25C	送信 #2 データレジスタ															0000 0000 0000 0000	
TXBUF3	25E	送信 #3 データレジスタ															0000 0000 0000 0000	

凡例: r = 予約、x = 未定、u = 不変。

注: 灰色で塗られたビットは SFR マップ内で、将来のモジュール拡張用の予約空間を示します。予約ビットは ‘0’ が読み込まれます。

## 22.8 設計の秘訣

質問 1: DCI は 16 ビット以上のデータワード長をサポートできますか?

回答: はい。ロングデータワードは、複数送信 / 受信レジスタを用いて送信 / 受信できます。  
詳しくは、セクション 22.5.3 「ロングデータワードをサポートするデータパッキング」を参照してください。

## 22.9 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正したり、制限を設けて（必要な場合は）使用できます。現状、データ変換器インターフェース (DCI) モジュールに関連する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 #
現在のところ、関連するアプリケーションノートはありません。	

**注：** dsPIC30F ファミリーのデバイスに関しての、他のアプリケーションノートやコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 22.10 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

この版は、dsPIC30F の データ変換器インターフェース (DCI) モジュールに関する追加技術内容と変更を含んでいます。



**MICROCHIP**

---

## 第 23 章 .CAN モジュール

---

### ハイライト

この章は、以下の項目を含んでいます。

23.1 序章 .....	23-2
23.2 CAN モジュールの制御レジスタ .....	23-2
23.3 CAN モジュールの特徴 .....	23-28
23.4 CAN モジュールの実装 .....	23-29
23.5 CAN モジュールの動作モード .....	23-38
23.6 メッセージ受信 .....	23-41
23.7 送信 .....	23-51
23.8 エラー検出 .....	23-60
23.9 CAN ポーレート .....	23-62
23.10 割り込み .....	23-66
23.11 タイムスタンプ .....	23-67
23.12 CAN モジュール I/O .....	23-67
23.13 CPU パワー節約モードでの動作 .....	23-68
23.14 CAN プロトコルの概要 .....	23-70
23.15 関連するアプリケーションノート .....	23-74
23.16 改訂履歴 .....	23-75

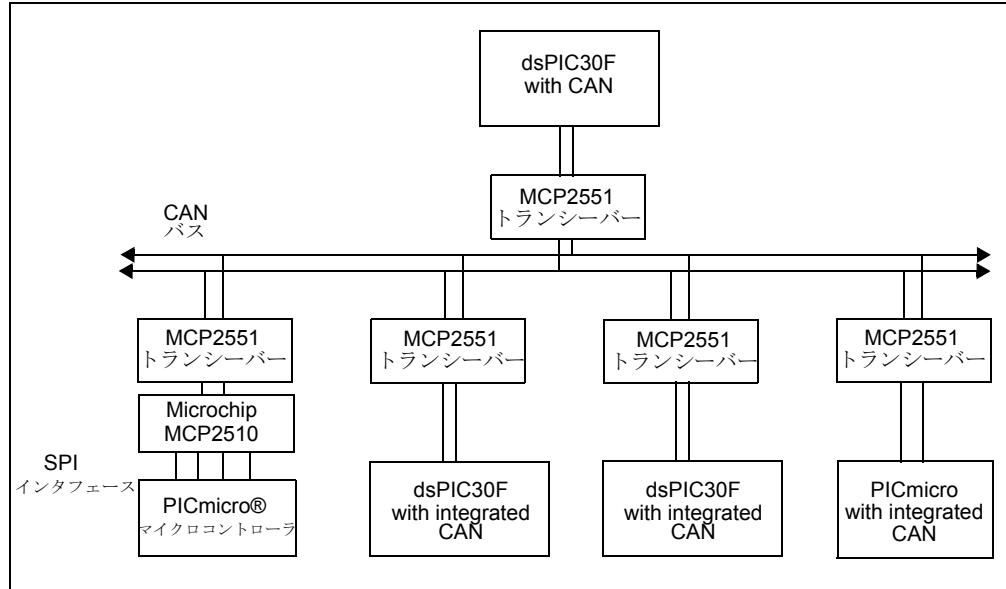
**23**

CAN モジュール

## 23.1 序章

制御エリアネットワーク (CAN) モジュールは、他の周辺もしくはマイクロコントローラデバイスとの通信用に役に立つシリアルインターフェースです。このインターフェース / プロトコルはノイズの多い環境内で通信ができるように設計されたものです。図 23-1 に CAN バスネットワークの例を示します。

図 23-1: CAN バスネットワークの例



## 23.2 CAN モジュールの制御レジスタ

CAN モジュールに関連する多くのレジスタがあります。これらのレジスタの説明は以下のようないくつかのセクションごとにグループ化されています。

- 制御とステータスレジスタ
- 送信バッファレジスタ
- 受信バッファレジスタ
- ポーレート制御レジスタ
- 割り込みステータスおよび制御レジスタ

**注 1:** レジスタ識別子内の ‘i’ は特定の CAN モジュール (CAN1 もしくは CAN2) を示します。  
**2:** レジスタ識別子内の ‘n’ はバッファ、フィルタもしくはマスク番号を示します。  
**3:** レジスタ識別子内の ‘m’ は特定の CAN データフィールド内のワード番号を示します。

### 23.2.1 CAN 制御とステータスレジスタ

レジスタ 23-1: CiCTRL: CAN モジュール制御とステータスレジスタ

上位バイト :							
R/W-x	U-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0
TSTAMP	—	CSIDL	ABAT	CANCKS		REQOP<2:0>	
ビット 15							ビット 8

下位バイト :							
R-1	R-0	R-0	U-0	R-0	R-0	R-0	U-0
	OPMODE<2:0>		—		ICODE<2:0>		—
ビット 7							ビット 0

- ビット 15 **TSTAMP:** CAN メッセージ受信キャプチャ有効ビット  
 1 = CAN キャプチャを有効にします。  
 0 = CAN キャプチャを無効にします。  
 注: TSTAMP は、CAN モジュールの動作モードにかかわらず常に書き込み可能です。
- ビット 14 未実装: ‘0’ が読み込まれます。
- ビット 13 **CSIDL:** IDLE モードでの停止ビット  
 1 = デバイスが IDLE モードに入ったら CAN モジュールを停止します。  
 0 = IDLE モードでも CAN モジュールの動作を継続します。
- ビット 12 **ABAT:** 中断中送信の全廃棄ビット  
 1 = 全ての送信バッファ内にある中断中の送信を廃棄します。  
 0 = 無影響  
 注: 全ての送信が廃棄されると、モジュールはこのビットをクリアします。
- ビット 11 **CANCKS:** CAN マスタークロック選択ビット  
 1 = FCAN クロックは FCY です。  
 0 = FCAN クロックは 4 FCY です。
- ビット 10-8 **REQOP<2:0>:** 動作モード要求ビット  
 111 = 全メッセージリスンモードを設定します。  
 110 = 予約  
 101 = 予約  
 100 = コンフィギュレーションモードを設定します。  
 011 = リスンオンリーモードを設定します。  
 010 = ループバックモードを設定します。  
 001 = 無効モードを設定します。  
 000 = 通常動作モードを設定します。
- ビット 7-5 **OPMODE<2:0>:** 動作モードビット  
 注: これらのビットは CAN モジュールの現動作モードを示します。REQOP bits (CiCTRL<10:8>) の説明を参照してください。
- ビット 4 未実装: ‘0’ が読み込まれます。

## レジスタ 23-1: CiCTRL: CAN モジュール制御とステータスレジスタ (続き)

ビット 3-1 **ICODE<2:0>**: 割り込みフラグコードビット

- 111 = ウエイクアップ割り込み
- 110 = RXB0 割り込み
- 101 = RXB1 割り込み
- 100 = TXB0 割り込み
- 011 = TXB1 割り込み
- 010 = TXB2 割り込み
- 001 = エラー割り込み
- 000 = 割り込み無し

ビット 0 未実装: ‘0’ が読み込まれます。

凡例:

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装ビット、‘0’が読み込まれます。

ト

-n = POR での値ト

‘1’ = ビットがセット ‘0’ = ビットがクリア x = ビットは不定です  
されています されています

### 23.2.2 CAN 送信バッファレジスタ

このサブセクションでは CAN 送信バッファと関連する送信バッファ制御レジスタの説明をします。

#### レジスタ 23-2: CiTXnCON: 送信バッファステータスおよび制御レジスタ

上位バイト :							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
ビット 15							ビット 8

下位バイト :								
U-0	R-0	R-0	R-0	R/W-0	U-0	R/W-0	R/W-0	
—	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI<1:0>		
ビット 7							ビット 0	

ビット 15-7 未実装: ‘0’ が読み込まれます。

ビット 6 TXABT: メッセージビット

1 = メッセージは廃棄されました。

0 = メッセージは廃棄されていません。

注： このビットは、TXREQ がセットされるとクリアされます。

ビット 5 TXLARB: メッセージアービトレーション喪失ビット

1 = 送信中にメッセージがアービトレーションを喪失した。

0 = 送信中にメッセージはアービトレーションを喪失していない。

注： このビットは、TXREQ がセットされるとクリアされます。

ビット 4 TXERR: 送信中のエラー検出ビット

1 = メッセージ送信中にバスエラーが発生しました。

0 = メッセージ送信中にバスエラーは発生していません。

注： このビットは、TXREQ がセットされるとクリアされます。

ビット 3 TXREQ: メッセージ送信要求ビット

1 = メッセージ送信を要求します。

0 = TXREQ がすでにセットされている場合、メッセージ送信を停止します。それ以外の場合は影響なしです。

注： メッセージの送信が成功すると、このビットは自動的にクリアされます。

ビット 2 未実装: ‘0’ が読み込まれます。

ビット 1-0 TXPRI<1:0>: メッセージ送信優先度ビット

11 = 最高位のメッセージ優先度

10 = 高中位のメッセージ優先度

01 = 低中位のメッセージ優先度

00 = 最低位のメッセージ優先度

凡例：

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装ビット、‘0’ が読み込まれます。  
ト

-n = POR での値ト

‘1’ = ビットがセット ‘0’ = ビットがクリア x = ビットは不定です  
されています されています

## レジスタ 23-3: CiTXnSID: 送信バッファ n の標準識別子

上位バイト :

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0
SID<10:6>				—	—	—	—
ビット 15							ビット 8

下位バイト :

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID<5:0>				SRR		TXIDE	
ビット 7							ビット 0

ビット 15-11 **SID<10:6>**: 標準識別子ビット

ビット 10-8 未実装: ‘0’ が読み込まれます。

ビット 7-2 **SID<6:0>**: 標準識別子ビット

ビット 1 **SRR**: 代替リモート要求制御ビット  
 1 = メッセージがリモート送信を要求します。  
 0 = 通常メッセージ

ビット 0 **TXIDE**: 拡張識別子ビット  
 1 = メッセージは拡張識別子を送信します。  
 0 = メッセージは標準識別子を送信します。

凡例 :

R = 読み込み可能ビット	W = 書き込み可能ビット	U = 未実装ビット、‘0’ が読み込まれます。
-n = POR での値ト	‘1’ = ビットがセットさ	‘0’ = ビットがクリアされ x = ビットは不定です
	れています	ています

## レジスタ 23-4: CiTXnEID: 送信バッファ n の拡張識別子

上位バイト :

R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0	U-0
EID<17:14>				—	—	—	—
ビット 15							ビット 8

下位バイト :

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID<13:6>				EID<13:6>			
ビット 7							ビット 0

ビット 15-12 **EID<17:14>**: 拡張識別子ビット 17-14

ビット 11-8 未実装: ‘0’ が読み込まれます。

ビット 7-0 **EID<13:6>**: 拡張識別子ビット 13-6

凡例 :

R = 読み込み可能ビット	W = 書き込み可能ビット	U = 未実装ビット、‘0’ が読み込まれます。
-n = POR での値ト	‘1’ = ビットがセットさ	‘0’ = ビットがクリアされ x = ビットは不定です
	れています	ています

## レジスタ 23-5: CiTXnDLC: 送信バッファ n のデータ長制御

上位バイト :							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID<5:0>					TXRTR	TXRB1	
ビット 15							ビット 8

下位バイト :							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0
TXRB0	DLC<3:0>					—	—
ビット 7							ビット 0

ビット 15-10 EID<5:0>: 拡張識別子ビット 5-0

ビット 9 TXRTR: リモート送信要求ビット

1 = メッセージがリモート送信を要求します。  
0 = 通常のメッセージ

ビット 8 TXRB<1:0>: 予約ビット

-7 注: CAN プロトコルに従って、ユーザーはこれらのビットを ‘1’ に設定しなければなりません。

ビット 6 DLC<3:0>: データ長コードビット

-3

ビット 2 未実装: ‘0’ が読み込まれます。  
-0

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装ビット、‘0’ が読み込まれます。

-n = POR での値ト

‘1’ = ビットがセット ‘0’ = ビットがクリア x = ビットは不定です  
されています されています

## レジスタ 23-6: CiTXnBm: 送信バッファ n のデータフィールドワード m

上位バイト :							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CTXB<15:8>							
ビット 15							ビット 8

下位バイト :							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CTXB<7:0>							
ビット 7							ビット 0

ビット 15-0 CTXB<15:0>: データフィールドバッファワードビット (2 バイト )

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装ビット、‘0’ が読み込まれます。

-n = POR での値ト

‘1’ = ビットがセットさ ‘0’ = ビットがクリアさ x = ビットは不定です  
れています されています

### 23.2.3 CAN 受信バッファレジスタ

このサブセクションでは受信バッファ制御レジスタと関連する受信バッファの説明をします。

レジスタ 23-7: CiRX0CON: 受信バッファ 0 ステータスおよび制御レジスタ

下位バイト：							
R/C-0	U-0	U-0	U-0	R-0	R/W-0	R/W-0	R-0
RXFUL	—	—	—	RXRTRRO	DBEN	JTOFF	FILHITO
ビット7				ビット0			

ビット 未実装: ‘0’ が読み込まれます。  
15-8

#### ビット7 RXEII・受信フルステータスビット

1 = 受信バッファには有効な受信メッセージがあります。

0 = 新しいメッセージを受信するために、受信バッファが空いています。

**注:** このビットはCAN?|×?によりセットされバッファが読み込まれた後ソフトウェアでクリアされねばなりません。

ビット 未塞装：‘0’が読み込まれます。

6-4

ビット3 RXRTRQ: 受信リモート転送要求ビット(読み込み専用)

リモート転送要求が受信されました。

0 = リモート転送要求は受信されていません。

注：このビットは、受信バッファ0に転送された最後のメッセージのステータスを反映します。

ビット2 DBEN: 受信バッファ0ダブルバッファ有效ビット

1 = 受信バッファオーバーフローが受信バッファ 1 に書き込まれます。

0 = 受信バッファ 0 オーバーフローは受信バッファ 1 に書き込まれません。

ビット1: **ITOFF**: ジャンプ素オフセットビット (DBREN のコピーで読み込み専用です)

1 ≒ 6 と 7 の間のジャンプ表オフセットが使用されます。

0 = 0 と 1 の間のジャンプ表オフセットが使用されます。

**ビット 0: ELL HITO:** ビのアクセプタンスフィルタがメッセージ受信を有効であるかを示すビット

### PIERRE. とのノックヒノターンスノイルタ 1 - アクセプタンスフィルタ 1(RXE1)

0 = アクセプタンスフィルタ 0(BXE0)

注：このビットは、受信バッファに転送された最後のメッセージのステータスを反映します。

四、例題：

**R** = 読み込み可能 ビット      **W** = 書き込み可能 ビット      **C** = クリア可能ビット      **U** = 未実装ビット、‘0’が読み込まれます。  
**-n** = POR での値ト ‘1’ = ビットが セットされています      ‘0’ = ビットがクリアされています      **X** = ビットは不定です

## レジスタ 23-8: CiRX1CON: 受信バッファ 1 ステータスおよび制御レジスタ

上位バイト :							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
ビット 15							
ビット 8							

下位バイト :										
R/C-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0			
RXFUL	—	—	—	RXRTRRO	FILHIT<2:0>					
ビット 7										
ビット 0										

ビット 15-8 未実装: ‘0’ が読み込まれます。

ビット 7 RXFUL: 受信フルステータスピット

1 = 受信バッファには有効な受信メッセージがあります。

0 = 新しいメッセージを受信するために、受信バッファが空いています。

注： このビットは CAN モジュールによりセットされ、バッファが読み込まれた後ソフトウェアでクリアされねばなりません。

ビット 6-4 未実装: ‘0’ が読み込まれます。

ビット 3 RXRTRRO: 受信リモート転送要求ビット (読み込み専用)

1 = リモート転送要求が受信されました。

0 = リモート転送要求は受信されていません。

注： このビットは、受信バッファ 1 に転送された最後のメッセージのステータスを反映します。

ビット 2-0 FILHIT<2:0>: どのアクセプタンスフィルタがメッセージ受信を有効であるかを示すビット

101 = アクセプタンスフィルタ 5(RXF5)

100 = アクセプタンスフィルタ 4(RXF4)

011 = アクセプタンスフィルタ 3(RXF3)

010 = アクセプタンスフィルタ 2(RXF2)

001 = アクセプタンスフィルタ 1(RXF1) (DBEN ビットがセットされている時のみ可能)

000 = アクセプタンスフィルタ 0(RXF0) (DBEN ビットがセットされている時のみ可能)

凡例：

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装ビット、‘0’ が読み込まれます。  
ト

-n = POR での値ト

‘1’ = ビットがセット ‘0’ = ビットがクリア X = ビットは不定です  
されています

## レジスタ 23-9: CiRXnSID: 受信バッファ n の標準識別子

上位バイト :

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x			
—	—	—		SID<10:6>						
ビット 15							ビット 8			

下位バイト :

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID<5:0>							SRR
ビット 7							ビット 0

ビット 15-13 未実装 : ‘0’ が読み込まれます。

ビット 12-2 SID<10:0>: 標準識別子ビット

ビット 1 SRR: 代替リモート要求制御ビット (RXIDE = 1 の時のみ)

1 = リモート転送要求が発生しています。  
0 = リモート転送要求は発生していません。

ビット 0 RXIDE: 拡張識別子フラグビット

1 = 受信メッセージは拡張データフレームです。SID<10:0> は EID<28:18> です。  
0 = 受信メッセージは標準識別子です。

凡例 :

R = 読み込み可能ビット	W = 書き込み可能ビット	U = 未実装ビット、‘0’ が読み込まれます。
-n = POR での値ト	‘1’ = ビットがセット されています	‘0’ = ビットがクリア されています

## レジスタ 23-10: CiRXnEID: 受信バッファ n の拡張識別子

上位バイト :

U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x			
—	—	—	—	EID<17:14>						
ビット 15							ビット 8			

下位バイト :

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID<13:6>							
ビット 7							ビット 0

ビット 15-12 未実装 : ‘0’ が読み込まれます。

ビット 11-0 EID<17:6>: 拡張識別子ビット 17-6

凡例 :

R = 読み込み可能ビット	W = 書き込み可能ビット	U = 未実装ビット、‘0’ が読み込まれます。
-n = POR での値ト	‘1’ = ビットがセットさ れています	‘0’ = ビットがクリアさ れています

## レジスタ 23-11: CiRXnBm: 受信バッファ n のデータフィールドワード m

上位バイト :							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CRXB<15:8>							
ビット 15							ビット 8

下位バイト :							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CRXB<7:0>							
ビット 7							ビット 0

ビット 15-0 CRXB<15:0>: データフィールドバッファワードビット (2 バイト )

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装ビット、'0' が読み込まれます。

-n = POR での値ト

'1' = ビットがセット '0' = ビットがクリア x = ビットは不定です  
されています

## レジスタ 23-12: CiRXnDLC: 受信バッファ n のデータ長制御

上位バイト :							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID<5:0>							
ビット 15							ビット 8

下位バイト :							
U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	RB0	DLC<3:0>			
ビット 7							ビット 0

ビット 15-10 EID<5:0>: 拡張識別子ビット

ビット 9 RXRTR: 受信リモート送信要求ビット

1 = リモート転送要求があります

0 = リモート転送要求はありません

注: このビットは、最後の受信メッセージ内にある RTR ビットのステータスを反映します。

ビット 8 RB1: 予約ビット 1

CAN 仕様により予約され、'0' が読み込まれます。

ビット 4 RB0: 予約ビット 0

CAN 仕様により予約され、'0' が読み込まれます。

ビット 3-0 DLC<3:0>: データ長コードビット (受信バッファの内容)

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装ビット、'0' が読み込まれます。

-n = POR での値ト

'1' = ビットがセットさ '0' = ビットがクリアさ x = ビットは不定です  
れています

## 23.2.4 メッセージアクセプタンスフィルタ

このサブセクションでは、メッセージアクセプタンスフィルタについて説明します。

レジスタ 23-13: CiRXFnSID: アクセプタンスフィルタ n 標準識別子

上位バイト:							
U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—		SID<10:6>			
ビット 15							ビット 8

下位バイト:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	R/W-x
			SID<5:0>			—	EXIDE
ビット 7							ビット 0

ビット 15-13 未実装: ‘0’ が読み込まれます。

ビット 12-2 SID<10:0>: 標準識別子ビット

ビット 1 未実装: ‘0’ が読み込まれます。

ビット 0 EXIDE: 拡張識別子ビット

MIDE = 1 ならば

1 = 拡張識別子用フィルタを有効にします。

0 = 標準識別子用フィルタを有効にします。

MIDE = 0 ならば、EXIDE は無視されます。

凡例:

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装ビット、‘0’ が読み込まれます。

ト

-n = POR での値ト

‘1’ = ビットがセットさ ‘0’ = ビットがクリアさ x = ビットは不定です

れています れています

レジスタ 23-14: CiRXFnEIDH: アクセプタンスフィルタ n 拡張識別子上位

上位バイト:							
U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	—	EID<17:14>			
ビット 15							ビット 8

下位バイト:

R/W-x R/W-x R/W-x R/W-x R/W-x R/W-x R/W-x R/W-x

EID<13:6>

ビット 7

ビット 0

ビット 15-12 未実装: ‘0’ が読み込まれます。

ビット 11-0 EID<17:6>: 拡張識別子ビット 17-6

凡例:

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装ビット、‘0’ が読み込まれます。

ト

-n = POR での値ト

‘1’ = ビットがセットさ ‘0’ = ビットがクリアさ x = ビットは不定です

れています れています

## レジスタ 23-15: CiRXFnEIDL: アクセプタンスフィルタ n の拡張識別子下位

上位バイト :							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0
EID<5:0>						—	—
ビット 15						ビット 8	

下位バイト :							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—

ビット 15-10 EID<5:0>: 拡張識別子ビット

ビット 9-0 未実装: ‘0’ が読み込まれます。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット  
U = 未実装ビット、‘0’ が読み込まれます。

-n = POR での値ト

‘1’ = ビットがセット ‘0’ = ビットがクリア x = ビットは不定です  
されています されています

## 23.2.5 アクセプタンスフィルタマスクレジスタ

レジスタ 23-16: CiRXMnSID: アクセプタンスフィルタマスク n の標準拡張子

上位バイト :

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x			
—	—	—		SID<10:6>						
ビット 15							ビット 8			

下位バイト :

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	R/W-x
			SID<5:0>			—	MIDE
ビット 7							ビット 0

ビット 15-13 未実装: ‘0’ が読み込まれます。

ビット 12-2 SID<10:0>: 標準識別子マスクビット

1 = フィルタ比較内のビットを含みます  
0 = フィルタ比較内のビットを含みません

ビット 1 未実装: ‘0’ が読み込まれます。

ビット 0 MIDE: 拡張子モード選択ビット

1 = フィルタ内のEXIDEビットにより決定されるようにメッセージ形式(標準しくは拡張アドレス)のみを一致させます。  
0 = フィルタが一致した場合、標準もしくは拡張アドレスメッセージを一致させます。

凡例 :

R = 読み込み可能ビット	W = 書き込み可能ビット	U = 未実装ビット、‘0’ が読み込まれます。
-n = POR での値ト	‘1’ = ビットがセットさ れています	‘0’ = ビットがクリアさ れています x = ビットは不定です れています

レジスタ 23-17: CiRXMnEIDH: アクセプタンスフィルタマスク n の拡張識別子上位

上位バイト :

U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x			
—	—	—	—	EID<17:14>						
ビット 15							ビット 8			

下位バイト :

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
			EID<13:6>			—	—
ビット 7							ビット 0

ビット 15-12 未実装: ‘0’ が読み込まれます。

ビット 11-0 EID<17:6>: 拡張識別子マスクビット 17-6

1 = フィルタ比較内のビットを含みます  
0 = フィルタ比較内のビットを含みません

凡例 :

R = 読み込み可能ビット	W = 書き込み可能ビット	U = 未実装ビット、‘0’ が読み込まれます。
-n = POR での値ト	‘1’ = ビットがセットさ れています	‘0’ = ビットがクリアさ れています x = ビットは不定です れています

## レジスタ 23-18: CiRXMnEIDL: アクセプタンスフィルタマスク n の拡張識別子下位

上位バイト :							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0
EID<5:0>						—	—
ビット 15						ビット 8	

下位バイト :							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—

ビット 15-10 EID<5:0>: 拡張識別子ビット

ビット 9-0 未実装: ‘0’ が読み込まれます。

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット  
U = 未実装ビット、‘0’ が読み込まれます。

-n = POR での値ト

‘1’ = ビットがセット ‘0’ = ビットがクリア x = ビットは不定です  
されています されています

## 23.2.6 CAN ボーレートレジスタ

このサブセクションでは CAN ボーレートレジスタについて説明します。

レジスタ 23-19: CiCFG1: ボーレート構成レジスタ

上位バイト：							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
ビット 15				ビット 8			

下位バイト：							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SJW<1:0>				BRP<5:0>			
ビット 7				ビット 0			

ビット 15-8 未実装：‘0’が読み込まれます。

ビット 7-6 **SJW<1:0>**: 同期化ジャンプ幅ビット  
11 = 同期化ジャンプ幅時間が  $4 \times TQ$  です。  
10 = 同期化ジャンプ幅時間が  $3 \times TQ$  です。  
01 = 同期化ジャンプ幅時間が  $2 \times TQ$  です。  
00 = 同期化ジャンプ幅時間が  $1 \times TQ$  です。

ビット 5-0 **BRP<5:0>**: ボーレートプリスケーラビット  
11 1111 =  $TQ = 2 \times (BRP + 1)/FCAN = 128/FCAN$

00 0000 =  $TQ = 2 \times (BRP + 1)/FCAN = 2/FCAN$

注： CANCKS ビットの設定により、 FCAN は FCY もしくは  $4 \times FCY$  になります。

凡例：

R = 読み込み可能ビット	W = 書き込み可能ビット	U = 未実装ビット、‘0’が読み込まれます。
-n = POR での値	ト	
‘1’ = ビットがセット	‘0’ = ビットがクリア	x = ビットは不定です
されています	されています	

## レジスタ 23-20: CiCFG2: ポーレート構成レジスタ 2

上位バイト :							
U-0	R/W-x	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x
—	WAKFIL	—	—	—	SEG2PH<2:0>		
ビット 15							ビット 8

下位バイト :							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SEG2PHTS	SAM	SEG1PH<2:0>			PRSEG<2:0>		
ビット 7							ビット 0

ビット 15 未実装: ‘0’ が読み込まれます。

ビット 14 **WAKFIL:** ウエイクアップ用の CAN バスラインフィルタの選択ビット  
1 = ウエイクアップ用に CAN バスラインフィルタを使用します  
0 = ウエイクアップ用に CAN バスラインフィルタを使用しません

ビット 13-11 未実装: ‘0’ が読み込まれます。

ビット 10-8 **SEG2PH<2:0>:** フェーズバッファセグメント 2 ビット  
111 = 長さは  $8 \times TQ$  です。  
·  
·  
000 = 長さは  $1 \times TQ$  です。

ビット 7 **SEG2PHTS:** フェーズセグメント 2 時間選択ビット  
1 = 自由にプログラム可能  
0 = SEG1PH の最大値もしくは情報処理時間 (3 TQ's) のうちどちらか大きい方

ビット 6 **SAM:** CAN バスラインのサンプルビット  
1 = サンプルポイントでバスラインが 3 回サンプルされます  
0 = サンプルポイントでバスラインが 1 回サンプルされます

ビット 5-3 **SEG1PH<2:0>:** フェーズバッファセグメント 1 ビット  
111 = 長さは  $8 \times TQ$  です。  
·  
·  
000 = 長さは  $1 \times TQ$  です。

ビット 2-0 **PRSEG<2:0>:** 伝達時間セグメントビット  
111 = 長さは  $8 \times TQ$  です。  
·  
000 = 長さは  $1 \times TQ$  です。

## 凡例 :

R = 読み込み可能ビット      W = 書き込み可能ビット      U = 未実装ビット、‘0’ が読み込まれます。  
ト

-n = POR での値ト      ‘1’ = ビットがセット      ‘0’ = ビットがクリア      x = ビットは不定です  
されています      されています

## 23.2.7 CAN モジュールエラーカウントレジスタ

このサブセクションでは、CAN モジュールの送信 / 受信エラーカウントレジスタについて説明します。種々のエラーステータスフラグが、CAN 割り込みフラグレジスタにあります。

### レジスタ 23-21: CiEC: 送信 / 受信エラーカウント

上位バイト：

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
TERRCNT<7:0>							
ビット 15							ビット 8

下位バイト：

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
RERRCNT<7:0>							
ビット 7							ビット 0

ビット 15-8 **TERRCNT<7:0>**: 送信エラーカウントビット

ビット 7-0 **RERRCNT<7:0>**: 受信エラーカウントビット

凡例：

R = 読み込み可能ビット

W = 書き込み可能ビット U = 未実装ビット、‘0’が読み込まれます。

ト

-n = POR での値ト

‘1’ = ビットがセット ‘0’ = ビットがクリア x = ビットは不定です

されています されています

### 23.2.8 CAN 割り込みレジスタ

このサブセクションでは、割り込みに関連する CAN レジスタを説明します。

#### レジスタ 23-22: CiINTE: 割り込み有効レジスタ

上位バイト :							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
ビット 15							ビット 8

下位バイト :							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IVRIE	WAKIE	ERRIE	TX2IE	TX1IE	TX0IE	RX1IE	RX0IE
ビット 7							ビット 0

ビット 15-8 未実装: ‘0’ が読み込まれます。

ビット 7 **IVRIE:** 無効メッセージ受信割り込み有効ビット

- 1 = 有効
- 0 = 無効

ビット 6 **WAKIE:** バスウェイクアップ動作割り込み有効ビット

- 1 = 有効
- 0 = 無効

ビット 5 **ERRIE:** エラー割り込み有効ビット

- 1 = 有効
- 0 = 無効

ビット 4 **TX2IE:** 送信バッファ 2 割り込み有効ビット

- 1 = 有効
- 0 = 無効

ビット 3 **TX1IE:** 送信バッファ 1 割り込み有効ビット

- 1 = 有効
- 0 = 無効

ビット 2 **TX0IE:** 送信バッファ 0 割り込み有効ビット

- 1 = 有効
- 0 = 無効

ビット 1 **RX1IE:** 受信バッファ 1 割り込み有効ビット

- 1 = 有効
- 0 = 無効

ビット 0 **RX0IE:** 受信バッファ 0 割り込み有効ビット

- 1 = 有効
- 0 = 無効

凡例 :

R = 読み込み可能ビット

W = 書き込み可能ビット

U = 未実装ビット、‘0’ が読み込まれます。

-n = POR での値ト

ト

‘1’ = ビットがセット ‘0’ = ビットがクリア x = ビットは不定です

されています されています

## レジスタ 23-23: CiINTF: 割り込みフラグレジスタ

上位バイト :

R/C-0	R/C-0	R-0	R-0	R-0	R-0	R-0	R-0
RX0OVR	RX1OVR	TXBO	TXEP	RXEPE	TXWAR	RXWAR	EWARN
ビット 15							

ビット 8

下位バイト :

| R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| IVRIF | WAKIF | ERRIF | TX2IF | TX1IF | TX0IF | RX1IF | RX0IF |
| ビット 7 |       |       |       |       |       |       |       |

ビット 0

- ビット 15 **RX0OVR:** 受信バッファ 0 オーバーフロー ビット  
1 = 受信バッファ 0 がオーバーフローしています。  
0 = 受信バッファ 0 はオーバーフローしていません。
- ビット 14 **RX1OVR:** 受信バッファ 1 オーバーフロー ビット  
1 = 受信バッファ 1 がオーバーフローしています。  
0 = 受信バッファ 1 はオーバーフローしていません。
- ビット 13 **TXBO:** 送信器がエラー状態、バスオフ ビット  
1 = 送信器がエラー状態で、バスオフします。  
0 = 送信器がエラー状態ではありません、バスオフします。
- ビット 12 **TXEP:** 送信器がエラー状態、バス受動 ビット  
1 = 送信器がエラー状態で、バスは受動状態です。  
0 = 送信器がエラー状態ではありません、バスは受動状態です。
- ビット 11 **RXEPE:** 受信器がエラー状態、バス受動 ビット  
1 = 受信器がエラー状態で、バスは受動状態です。  
0 = 受信器がエラー状態ではありません、バスは受動状態です。
- ビット 10 **TXWAR:** 送信器がエラー状態、警告 ビット  
1 = 送信器がエラー状態で、警告です。  
0 = 送信器がエラー状態ではありません、警告です。
- ビット 9 **RXWAR:** 受信器がエラー状態、警告 ビット  
1 = 受信器がエラー状態で、警告です。  
0 = 受信器がエラー状態ではありません、警告です。
- ビット 8 **EWARN:** 送信器もしくは受信器がエラー状態、警告 ビット  
1 = 送信器もしくは受信器がエラー状態、警告です。  
0 = 送信器も受信器もエラー状態ではありません。
- ビット 7 **IVRIF:** 無効メッセージ受信割り込みフラグ ビット  
1 = 最後のメッセージの受信中にある種のエラーが発生しました。  
0 = 受信エラーは発生していません。
- ビット 6 **WAKIF:** バスウェイクアップ動作割り込みフラグ ビット  
1 = 割り込み要求が発生しました。  
0 = 割り込み要求は発生していません。
- ビット 5 **ERRIF:** エラー割り込みフラグ ビット (CiINTF<15:8> レジスタ内の複数ソース )  
1 = 割り込み要求が発生しました。  
0 = 割り込み要求は発生していません。
- ビット 4 **TX2IF:** 送信バッファ 2 割り込みフラグ ビット  
1 = 割り込み要求が発生しました。  
0 = 割り込み要求は発生していません。
- ビット 3 **TX1IF:** 送信バッファ 1 割り込みフラグ ビット  
1 = 割り込み要求が発生しました。  
0 = 割り込み要求は発生していません。

## レジスタ 23-23: CiINTF: 割り込みフラグレジスタ (続き)

ビット 2 **TX0IF:** 送信バッファ 0 割り込みフラグビット

1 = 割り込み要求が発生しました。

0 = 割り込み要求は発生していません。

ビット 1 **RX1IF:** 受信バッファ 1 割り込みフラグビット

1 = 割り込み要求が発生しました。

0 = 割り込み要求は発生していません。

ビット 0 **RX0IF:** 受信バッファ 0 割り込みフラグビット

1 = 割り込み要求が発生しました。

0 = 割り込み要求は発生していません。

凡例 :

<b>R</b> = 読み込み可能 ビット	<b>W</b> = 書き込み可能 ビット	<b>C</b> = クリア可能ビット ビット	<b>U</b> = 未実装ビット、‘0’が読み込まれます。
<b>-n</b> = POR での値ト セットされています	<b>‘1’</b> = ビットが セットされています	<b>‘0’</b> = ビットがクリアさ れています	<b>x</b> = ビットは不定です

表 23-1: CAN1 レジスタマップ

ファイル名	アドレス	ピット															リセット
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
C1RXF0SID	300	—	—	—	SID<10:6>										—	EXIDE	XXXX
C1RXF0EIDH	302	—	—	—	—	EID<17:14>										EID<13:6>	XXXX
C1RXF0EIDL	304	EID<5:0>										—	—	—	—	—	XXXX
使わない	306	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	XXXX
C1RXF1SID	308	—	—	—	SID<10:6>										—	EXIDE	XXXX
C1RXF1EIDH	30A	—	—	—	—	EID<17:14>										EID<13:6>	XXXX
C1RXF1EIDL	30C	EID<5:0>										—	—	—	—	—	XXXX
使わない	30E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	XXXX
C1RXF2SID	310	—	—	—	SID<10:6>										—	EXIDE	XXXX
C1RXF2EIDH	312	—	—	—	—	EID<17:14>										EID<13:6>	XXXX
C1RXF2EIDL	314	EID<5:0>										—	—	—	—	—	XXXX
使わない	316	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	XXXX
C1RXF3SID	318	—	—	—	SID<10:6>										—	EXIDE	XXXX
C1RXF3EIDH	31A	—	—	—	—	EID<17:14>										EID<13:6>	XXXX
C1RXF3EIDL	31C	EID<5:0>										—	—	—	—	—	XXXX
使わない	31E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	XXXX
C1RXF4SID	320	—	—	—	SID<10:6>										—	EXIDE	XXXX
C1RXF4EIDH	322	—	—	—	—	EID<17:14>										EID<13:6>	XXXX
C1RXF4EIDL	324	EID<5:0>										—	—	—	—	—	XXXX
使わない	326	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	XXXX
C1RXF5SID	328	—	—	—	SID<10:6>										—	EXIDE	XXXX
C1RXF5EIDH	32A	—	—	—	—	EID<17:14>										EID<13:6>	XXXX
C1RXF5EIDL	32C	EID<5:0>										—	—	—	—	—	XXXX
使わない	32E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	XXXX
C1RXMOSID	330	—	—	—	SID<10:6>										—	MIDE	XXXX
C1R XM0EIDH	332	—	—	—	—	EID<17:14>										EID<13:6>	XXXX
C1R XM0EIDL	334	EID<5:0>										—	—	—	—	—	XXXX
使わない	336	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	XXXX
C1R XM1SID	338	—	—	—	SID<10:6>										—	MIDE	XXXX
C1R XM1EIDH	33A	—	—	—	—	EID<17:14>										EID<13:6>	XXXX
C1R XM1EIDL	33C	EID<5:0>										—	—	—	—	—	XXXX
使わない	33E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	XXXX

表 23-1: CAN1 レジスタマップ (続き)

ファイル名	アドレス	ビット															リセット				
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
C1TX2SID	340	SID<10:6>						—	—	—	SID<5:0>						SRR TX IDE xxxx				
C1TX2EID	342	EID<17:14>				—	—	—	—	EID<13:6>								xxxx			
C1TX2DLC	342	EID<5:0>						TX RTR	TX RB1	TX RB0	DLC<3:0>			—	—	—	xxxx				
C1TX2D01	346	送信バッファ0 バイト1								送信バッファ0 バイト0								xxxx			
C1TX2D23	348	送信バッファ0 バイト3								送信バッファ0 バイト2								xxxx			
C1TX2D45	34A	送信バッファ0 バイト5								送信バッファ0 バイト4								xxxx			
C1TX2D67	34C	送信バッファ0 バイト7								送信バッファ0 バイト6								xxxx			
C1TX2CON	34E	—	—	—	—	—	—	—	—	TX ABT	TX LARB	TX ERR	TX REQ	—	TXPRI[1:0]		0000				
C1TX1SID	350	SID<10:6>						—	—	—	SID<5:0>						SRR TX IDE xxxx				
C1TX1EID	352	EID<17:14>				—	—	—	—	EID<13:6>								xxxx			
C1TX1DLC	352	EID<5:0>						TX RTR	TX RB1	TX RB0	DLC<3:0>			—	—	—	xxxx				
C1TX1D01	356	送信バッファ0 バイト1								送信バッファ0 バイト0								xxxx			
C1TX1D23	358	送信バッファ0 バイト3								送信バッファ0 バイト2								xxxx			
C1TX1D45	35A	送信バッファ0 バイト5								送信バッファ0 バイト4								xxxx			
C1TX1D67	35C	送信バッファ0 バイト7								送信バッファ0 バイト6								xxxx			
C1TX1CON	35E	—	—	—	—	—	—	—	—	TX ABT	TX LARB	TX ERR	TX REQ	—	TXPRI[1:0]		0000				
C1TX0SID	360	SID<10:6>						—	—	—	SID<5:0>						SRR TX IDE xxxx				
C1TX0EID	362	EID<17:14>				—	—	—	—	EID<13:6>								xxxx			
C1TX0DLC	362	EID<5:0>						TX RTR	TX RB1	TX RB0	DLC<3:0>			—	—	—	xxxx				
C1TX0D01	366	送信バッファ0 バイト1								送信バッファ0 バイト0								xxxx			
C1TX0D23	368	送信バッファ0 バイト3								送信バッファ0 バイト2								xxxx			
C1TX0D45	36A	送信バッファ0 バイト5								送信バッファ0 バイト4								xxxx			
C1TX0D67	36C	送信バッファ0 バイト7								送信バッファ0 バイト6								xxxx			
C1TX0CON	36E	—	—	—	—	—	—	—	—	TX ABT	TX LARB	TX ERR	TX REQ	—	TXPRI[1:0]		0000				

表 23-1: CAN1 レジスタマップ (続き)

ファイル名	アドレス	ピット															リセット	
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		
C1RX1SID	370	—	—	—	SID<10:6>					SID<5:0>					SRR	RX IDE	xxxx	
C1RX1EID	372	—	—	—	—	EID<17:14>					EID<13:6>						xxxx	
C1RX1DLC	374	EID<0:5>					RX RTR	RX RB1	—	—	—	RX RB0	DLC[3:0]				xxxx	
C1RX1D01	376	受信バッファ 1 バイト 1							受信バッファ 1 バイト 0								xxxx	
C1RX1D23	378	受信バッファ 1 バイト 3							受信バッファ 1 バイト 2								xxxx	
C1RX1D45	37A	受信バッファ 1 バイト 5							受信バッファ 1 バイト 4								xxxx	
C1RX1D67	37C	受信バッファ 1 バイト 7							受信バッファ 1 バイト 6								xxxx	
C1RX1CON	37E	—	—	—	—	—	—	—	RX FUL	—	—	RX ERR	RX RTR R0	FILHIT[2:0]			0000	
C1RX1SID	380	—	—	—	SID<10:6>					SID<5:0>					SRR	RX IDE	xxxx	
C1RX1EID	382	—	—	—	—	EID<17:14>					EID<13:6>						xxxx	
C1RX1DLC	384	EID<0:5>					RX RTR	RX RB1	—	—	—	RX RB0	DLC[3:0]				xxxx	
C1RX0D01	386	受信バッファ 0 バイト 1							受信バッファ 0 バイト 0								xxxx	
C1RX0D23	388	受信バッファ 0 バイト 3							受信バッファ 0 バイト 2								xxxx	
C1RX0D45	38A	受信バッファ 0 バイト 5							受信バッファ 0 バイト 4								xxxx	
C1RX0D67	38C	受信バッファ 0 バイト 7							受信バッファ 0 バイト 6								xxxx	
C1RX0CON	38E	—	—	—	—	—	—	—	RX FUL	—	—	RX ERR	RX RTR R0	RXB0 DBEN	JTOFF	FIL HIT 0	0000	
C1CTRL	390	CAN CAP	CAN FRZ	CAN SIDL	ABAT	CAN CKS	REQOP[2:0]			OPMODE[2:0]			—	ICODE[2:0]			—	0480
C1CFG1	392	—	—	—	—	—	—	—	—	SJW[1:0]S		BRP[5:0]					0000	
C1CFG2	394	—	WAK FIL	—	—	—	SEG2PH[2:0]			SEG2 PHTS	SAM	SEG1PH[2:0]			PRSEG[2:0]			0000
C1INTF	396	RXB0 OVR	RXB1 OVR	TXB0	TXBP	RXBP	TX WARN	RX WARN	E WARN	IVR IF	WAK IF	ERR IF	TXB2 IF	TXB1 IF	TXB0 IF	RXB1 IF	RXB0 IF	0000
C1INTE	398	—	—	—	—	—	—	—	—	IVR IE	WAK IE	ERR IE	TXB2 IE	TXB1 IE	TXB0 IE	RXB1 IE	RXB0 IE	0000
C1TREC	39A	送信エラーカウンター							受信エラーカウンター								0000	
予約	39C 3FE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx	

凡例: x = 不明

表 23-2: CAN2 レジスタマップ<sup>a</sup>

ファイル名	アドレス	ピット															リセット	
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		
C2RXF0SID	400	—	—	—	SID<10:6>						SID<5:0>						EXIDE	xxxxx
C2RXF0EIDH	402	—	—	—	—	EID<17:14>						EID<13:6>						xxxxx
C2RXF0EIDL	404	EID<5:0>						—	—	—	—	—	—	—	—	—	xxxxx	
使わない	406	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxxx	
C2RXF1SID	408	—	—	—	SID<10:6>						SID<5:0>						EXIDE	xxxxx
C2RXF1EIDH	40A	—	—	—	—	EID<17:14>						EID<13:6>						xxxxx
C2RXF1EIDL	40C	EID<5:0>						—	—	—	—	—	—	—	—	—	xxxxx	
使わない	40E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxxx	
C2RXF2SID	410	—	—	—	SID<10:6>						SID<5:0>						EXIDE	xxxxx
C2RXF2EIDH	412	—	—	—	—	EID<17:14>						EID<13:6>						xxxxx
C2RXF2EIDL	414	EID<5:0>						—	—	—	—	—	—	—	—	—	xxxxx	
使わない	416	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxxx	
C2RXF3SIDH	418	—	—	—	SID<10:6>						SID<5:0>						EXIDE	xxxxx
C2RXF3EID	41A	—	—	—	—	EID<17:14>						EID<13:6>						xxxxx
C2RXF3EIDL	41C	EID<5:0>						—	—	—	—	—	—	—	—	—	xxxxx	
使わない	41E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxxx	
C2RXF4SID	420	—	—	—	SID<10:6>						SID<5:0>						EXIDE	xxxxx
C2RXF4EIDH	422	—	—	—	—	EID<17:14>						EID<13:6>						xxxxx
C2RXF4EIDL	424	EID<5:0>						—	—	—	—	—	—	—	—	—	xxxxx	
使わない	426	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxxx	
C2RXF5SID	428	—	—	—	SID<10:6>						SID<5:0>						MIDE	xxxxx
C2RXF5EIDH	42A	—	—	—	—	EID<17:14>						EID<13:6>						xxxxx
C2RXF5EIDL	42C	EID<5:0>						—	—	—	—	—	—	—	—	—	xxxxx	
使わない	42E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxxx	
C2RXM0SID	430	—	—	—	SID<10:6>						SID<5:0>						MIDE	xxxxx
C2RXM0EIDH	432	—	—	—	—	EID<17:14>						EID<13:6>						xxxxx
C2RXM0EIDL	434	EID<5:0>						—	—	—	—	—	—	—	—	—	xxxxx	
使わない	436	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxxx	
C2RXM1SID	438	—	—	—	SID<10:6>						SID<5:0>						MIDE	xxxxx
C2RXM1EIDH	43A	—	—	—	—	EID<17:14>						EID<13:6>						xxxxx
C2RXM1EIDL	43C	EID<5:0>						—	—	—	—	—	—	—	—	—	xxxxx	
使わない	43E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxxx	

表 23-2: CAN2 レジスタマップ (続き)

ファイル名	アドレス	ピット															リセット						
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1							
C2TX2SID	440	SID<10:6>							—	—	—	SID<5:0>							SRR TX IDE xxxx				
C2TX2EID	442	EID<17:14>							—	—	—	EID<13:6>							xxxx				
C2TX2DLC	442	EID<5:0>							TX RTR	TX RB1	TX RB0	DLC<3:0>							xxxx				
C2TX2D01	446	送信バッファ 0 バイト 1							送信バッファ 0 バイト 0							xxxx		xxxx					
C2TX2D23	448	送信バッファ 0 バイト 3							送信バッファ 0 バイト 2							xxxx		xxxx					
C2TX2D45	44A	送信バッファ 0 バイト 5							送信バッファ 0 バイト 4							xxxx		xxxx					
C2TX2D67	44C	送信バッファ 0 バイト 7							送信バッファ 0 バイト 6							xxxx		xxxx					
C2TX2CON	44E	—	—	—	—	—	—	—	—	TX ABT	TX LARB	TX ERR	TX REQ	—	TXPRI[1:0]		0000		xxxx				
C2TX1SID	450	SID<10:6>							—	—	—	SID<5:0>							SRR TX IDE xxxx				
C2TX1EID	452	EID<17:14>							—	—	—	EID<13:6>							xxxx				
C2TX1DLC	352	EID<5:0>							TX RTR	TX RB1	TX RB0	DLC<3:0>							xxxx				
C2TX1D01	456	送信バッファ 0 バイト 1							送信バッファ 0 バイト 0							xxxx		xxxx					
C2TX1D23	458	送信バッファ 0 バイト 3							送信バッファ 0 バイト 2							xxxx		xxxx					
C2TX1D45	45A	送信バッファ 0 バイト 5							送信バッファ 0 バイト 4							xxxx		xxxx					
C2TX1D67	45C	送信バッファ 0 バイト 7							送信バッファ 0 バイト 6							xxxx		xxxx					
C2TX1CON	45E	—	—	—	—	—	—	—	—	TX ABT	TX LARB	TX ERR	TX REQ	—	TXPRI[1:0]		0000		xxxx				
C2TX0SID	460	SID<10:6>							—	—	—	SID<5:0>							SRR TX IDE xxxx				
C2TX0EID	462	EID<17:14>							—	—	—	EID<13:6>							xxxx				
C2TX0DLC	462	EID<5:0>							TX RTR	TX RB1	TX RB0	DLC<3:0>							xxxx				
C2TX0D01	466	送信バッファ 0 バイト 1							送信バッファ 0 バイト 0							xxxx		xxxx					
C2TX0D23	468	送信バッファ 0 バイト 3							送信バッファ 0 バイト 2							xxxx		xxxx					
C2TX0D45	46A	送信バッファ 0 バイト 5							送信バッファ 0 バイト 4							xxxx		xxxx					
C2TX0D67	46C	送信バッファ 0 バイト 7							送信バッファ 0 バイト 6							xxxx		xxxx					
C2TX0CON	46E	—	—	—	—	—	—	—	—	TX ABT	TX LARB	TX ERR	TX REQ	—	TXPRI[1:0]		0000		xxxx				

表 23-2: CAN2 レジスタマップ (続き)

ファイル名	アドレス	ビット															リセット					
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
C2RX1SID	470	—	—	—	SID<10:6>						SID<5:0>						SRR	RX IDE	xxxxx			
C2RX1EID	472	—	—	—	—	EID<17:14>						EID<13:6>						xxxxx				
C2RX1DLC	474	EID<0:5>						RX RTR	RX RB1	—	—	—	RX RBO	DLC[3:0]				xxxxx				
C2RX1D01	476	受信バッファ 1 バイト 1								受信バッファ 1 バイト 0								xxxxx				
C2RX1D23	478	受信バッファ 1 バイト 3								受信バッファ 1 バイト 2								xxxxx				
C2RX1D45	47A	受信バッファ 1 バイト 5								受信バッファ 1 バイト 4								xxxxx				
C2RX1D67	47C	受信バッファ 1 バイト 7								受信バッファ 1 バイト 6								xxxxx				
C2RX1CON	47E	—	—	—	—	—	—	—	—	RX FUL	—	—	RX ERR	RX RTR R0	FILHIT[2:0]			0000				
C2RX1SID	480	—	—	—	SID<10:6>						SID<5:0>						SRR	RX IDE	xxxxx			
C2RX1EID	482	—	—	—	—	EID<17:14>						EID<13:6>						xxxxx				
C2RX1DLC	484	EID<0:5>						RX RTR	RX RB1	—	—	—	RX RBO	DLC[3:0]				xxxxx				
C2RX0D01	486	受信バッファ 0 バイト 1								受信バッファ 0 バイト 0								xxxxx				
C2RX0D23	488	受信バッファ 0 バイト 3								受信バッファ 0 バイト 2								xxxxx				
C2RX0D45	48A	受信バッファ 0 バイト 5								受信バッファ 0 バイト 4								xxxxx				
C2RX0D67	48C	受信バッファ 0 バイト 7								受信バッファ 0 バイト 6								xxxxx				
C2RX0CON	48E	—	—	—	—	—	—	—	—	RX FUL	—	—	RX ERR	RX RTR R0	RXB0 DBEN	JTOFF	FIL HIT 0	0000				
C2CTRL	490	CAN CAP	CAN FRZ	CAN SIDL	ABAT	CAN CKS	REQOP[2:0]				OPMODE[2:0]				—	ICODE[2:0]		—	0480			
C2CFG1	492	—	—	—	—	—	—	—	—	SJW[1:0]S			BRP[5:0]						0000			
C2CFG2	494	—	WAK FIL	—	—	—	SEG2PH[2:0]				SEG2 PHTS	SAM	SEG1PH[2:0]				PRSEG[2:0]		0000			
C2INTF	496	RXB0 OVR	RXB1 OVR	TXB0	TXBP	RXBP	TX WARN	RX WARN	E WARN	IVR IF	WAK IF	ERR IF	TXB2	TXB1 IF	TXB0	RXB1 IF	RXB0 IF	0000				
C2INTE	498	—	—	—	—	—	—	—	—	IVR IE	WAK IE	ERR IE	TXB2 IE	TXB1 IE	TXB0 IE	RXB1 IE	RXB0 IE	0000				
C2TREC	49A	送信エラーカウンター								受信エラーカウンター								0000				
予約	49C 4FE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxxx				

凡例: x = 不明

## 23.3 CAN モジュールの特徴

CAN モジュールは、BOSCH 規格で定義された CAN 2.0A/B プロトコルを組み込んだ通信コントローラです。本モジュールはプロトコルのうち、CAN 1.2、CAN 2.0A、CAN 2.0B パッシブおよび CAN 2.0B アクティブ版をサポートします。このモジュールの実装はフル CAN システムです。

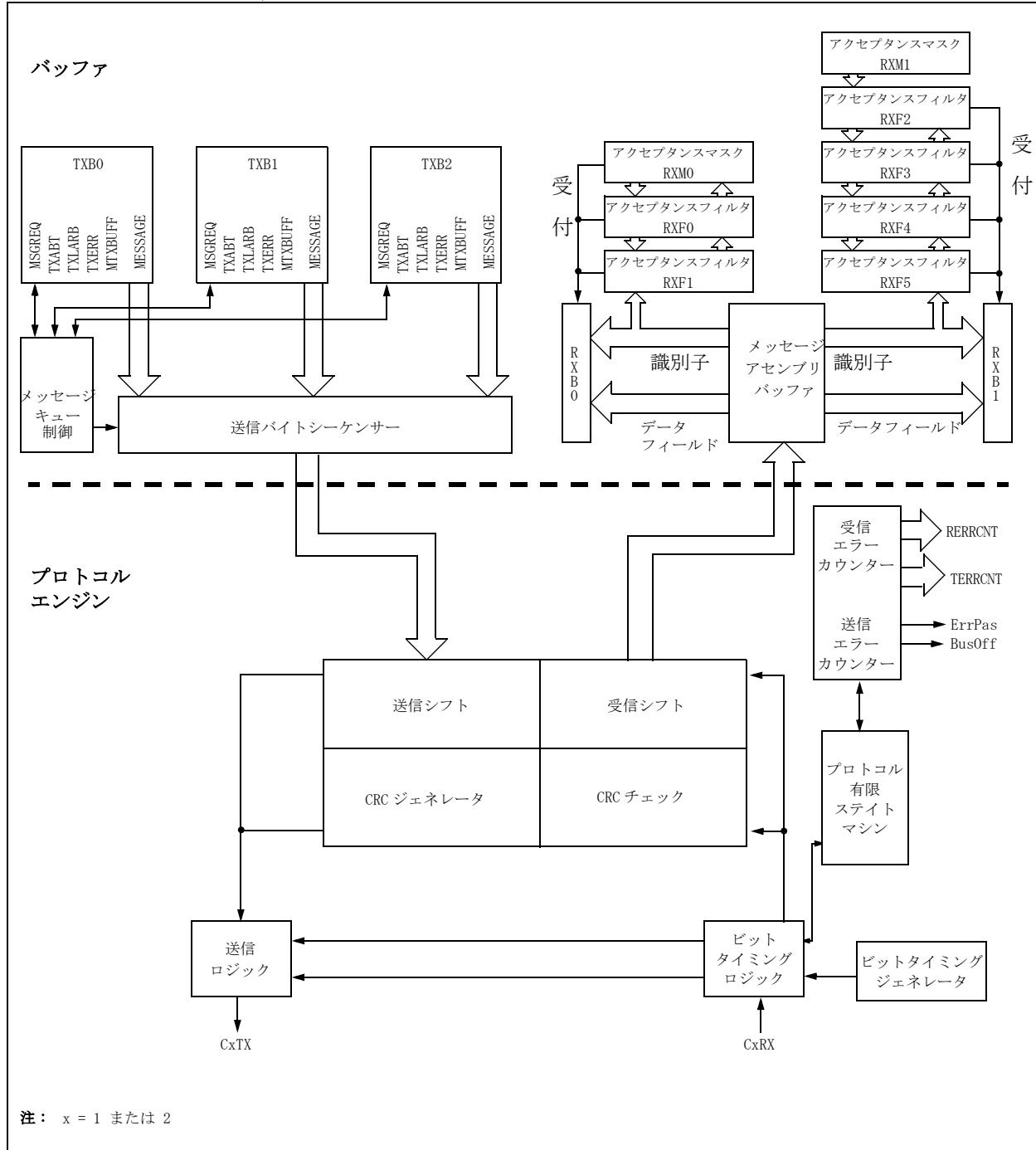
モジュールは以下のようないくつかの特徴を持ちます。

- CAN 1.2, CAN 2.0A および CAN 2.0B の CAN プロトコルを組み込んでいます。
- 標準および拡張データフレームをサポート
- データ長は 0-8 バイト
- 最大 1 Mbit/sec までのプログラム可能なビットレート
- リモートデータフレームをサポート
- 2 つの優先度を持つ受信メッセージ蓄積バッファを持つ二重バッファの受信器
- 6 つのフル (標準 / 拡張識別子) アクセプタンスフィルタを持ち、うち 2 つは高優先度受信バッファを持ち、4 つは低優先度受信バッファを持ちます。
- 2 つのフルアクセプタンスマスクを持ち、1 つは高優先度、1 つは低優先度のバッファを持ちます。
- アプリケーションで規定された優先度機能と停止機能を持つ 3 つの送信バッファ。
- 内蔵ローパスフィルタを持つプログラム可能なウェイクアップ機能
- プログラム可能なループバックモードにより自己テスト動作が可能です。
- 全ての CAN 受信および送信エラーステータスに対する割り込み信号生成
- プログラム可能なクロックソース
- タイムスタンプとネットワーク同期用のタイマモジュールへのプログラム可能なリンク
- 低電力 SLEEP モード

## 23.4 CAN モジュールの実装

CAN バスモジュールは、プロトコルエンジン、メッセージバッファと制御部から構成されます。プロトコルエンジンは、モジュールにより送信 / 受信されるデータフレームの形式を定義することにより、一番良く理解できます。これらのブロックを図 23-2 に示します。

図 23-2: CAN バッファとプロトコルエンジンブロック図



## 23.4.1 CAN メッセージフォーマット

CAN プロトコルエンジンは、CAN バス上の受信 / 送信メッセージに関する全ての機能について取扱います。メッセージは、最初に適切なデータレジスタに転送することで送信されます。ステータスとエラーは、適切なレジスタを読み出すことでチェックできます。CAN バス上で検出されたどんなメッセージについてもエラーのチェックがなされ、それを受信し、2 つの受信レジスタのうちの 1 つに格納すべきかどうかをフィルタと一致するかどうかで決定します。

CAN モジュールは以下のフレーム形式をサポートします。

- 標準データフレーム
- 拡張データフレーム
- リモートフレーム
- エラーフレーム
- オーバーロードフレーム
- インターフレーム空間

### 23.4.1.1 標準データフレーム

標準データフレームは、ノードがデータを送信したい時にノードによって生成されます。標準 CAN データフレームを、図 23-3 に示します。その他のフレームと共に、フレームは、全てのノードのハード上の同期を取るために、フレームスタートビット (SOF- ドミナント状態) から始まります。

SOF の後に、12 ビットからなるアービトレーションフィールドが続きますが、それは 11 ビットの識別子 (メッセージの内容と優先度を反映) と RTR ビット (リモート送信要求ビット) から構成されます。RTR ビットは、データフレーム (RTR ドミナント) とリモートフレームとを区別するために用いられます。

次のフィールドは制御フィールドで、6 ビットから構成されます。このフィールドの最初のビットは識別子拡張 (IDE) と呼ばれ、これがドミナント状態のとき標準フレームであることを表します。引続くビットは、CAN プロトコル、RBO により予約され、ドミナントビットとして規定されます。制御フィールドの残りの 4 ビットは、データ長コード (DLC) で、メッセージに含まれるデータのバイト数を規定します。

送信されるデータは、上記 DLC により規定される長さ (0-8 バイト) のデータフィールドが引続きます。

巡回型冗長チェック (CRC) フィールドが引続き、発生しうる転送エラーの検出に用いられます。CRC フィールドは 15 ビットの CRC シーケンスとデリミタビットで構成されます。メッセージはフレームの終わり (EOF) により完了し、EOF はビットスタッフのない 7 つのリセシプビットにより構成されます。

最終フィールドはアクノレッジフィールドです。ACK スロットビットでは、送信ノードはリセシプビットを送出します。エラーの無いフレームを受信したなどのノードも、(ノードが特定メッセージを受け付けるように構成されているかどうかに関わらず) ドミナントビットを返信することにより正しいフレームを受信したことを認識します。リセシプのアクノレッジデリミタでアクノレッジスロットを終了させ、エラーフレームが発生した時以外は、ドミナントビットにより上書きされることはありません。

### 23.4.1.2 拡張データフレーム

拡張 CAN データフレームでは、図 23-4 に示すように、フレーム開始ビット (SOF) に引続き、38 ビットで構成されるアービトレーションフィールドがあります。最初の 11 ビットは 29 ビットの識別子 ("Base-ID") の上位 11 ビットです。これらの 11 ビットに続き、リセシプとして送信される代替リモート要求ビット (SRR) がきます。SRR に続き、フレームが拡張 CAN フレームであることを示すリセシプの IDE ビットがきます。こうすると、識別子の最初の 11 ビットの送信後にアービトレーションが未解決の場合、アービトレーションにからんでいるノードの 1 つが標準 CAN フレーム (11 ビット識別子) を送信していると、標準 CAN フレームは、ドミナント IDE ビットであるためアービトレーションを獲得する、という点に注意が必要です。また、拡張 CAN フレーム内の SRR ビットは、標準 CAN フレームを送信しているノードによりドミナント RTR が発行できるようにリセシプでなければなりません。SRR と IDE ビットに引続き、識別子 ("ID-Extension") の残りの 18 ビットとドミナントリモート送信要求ビットが続きます。

共有ネットワーク間で、標準と拡張フレームが送信されることを有効にするために、29ビットの拡張メッセージ識別子を11ビット(上位)と18ビット(下位)セクションに分割する必要があります。この分割により、識別子拡張ビット(IDE)が標準と拡張の両フレーム内で同じ位置を維持することができます。

次のフィールドは制御フィールドで、6ビットで構成されます。このフィールドの最初の2ビットは予約され、ドミナント状態にあります。制御フィールドの残りの4ビットはデータ長コード(DLC)であり、データバイト数を規定します。

フレームの残りの部分(データフィールド、CRCフィールド、通知フィールド、フレーム終了および中断)は、標準データフレームと同様に構成されます。

#### 23.4.1.3 リモートフレーム

データ送信は、通常データソースノードを使って自動的に動作します(例えば、センサはデータフレームを送出します)。しかし、宛先ノードがソースからデータを要求することができます。この目的のために、宛先ノードは、要求されたデータフレームの識別子と一致する識別子と一緒にリモートフレームを送信します。対応するデータソースノードは、このリモート要求の応答としてデータフレームを送信します。

図23-5に示すように、リモートフレームとデータフレームとでは2つの違いがあります。第1に、RTRビットがリセッショング状態であること、第2にデータフィールドが無いことです。希にしか発生しませんが、同じ識別子を持ったデータフレームとリモートデータフレームが、同時に送信されるという場合、識別子に続くドミナントRTRのために、データフレームがアービトレーション権を獲得します。このようにして、リモートフレームを送信したノードは所望のデータを直ぐに受信します。

#### 23.4.1.4 エラーフレーム

エラーフレームは、バスエラーを検出したなどのノードによっても生成されます。図23-6に示すように、エラーフレームは、エラーフラグフィールドとそれに続くエラーデリミタフィールドの2つのフィールドにより構成されます。エラーデリミタは8つのリセッショングビットにより構成され、エラーの後、バスノードがバス通信をきれいに再開できるようにします。エラーフラグフィールドには2つの形式があります。エラーフラグフィールドの形式は、エラーを検出したノードのエラー状態に依存します。

エラーアクティブノードがバスエラーを検出した場合、そのノードはアクティブエラーフラグを生成して現メッセージの送信に割り込みをかけます。アクティブエラーフラグは、6個の連続するドミナントビットで構成されます。このビットシーケンスは、ビットスタッフルルルに敢えて違反します。その他の全てのノードは、ビットスタッフルルルを認識し、今度は自分自身で、エラーエコーフラグと呼ばれるエラーフレームを生成します。従って、エラーフラグフィールドは6から12の連続するドミナントビット(1つもしくは2つのノードにより生成されます)で構成されます。エラーデリミタによりエラーフレームが完了します。エラーフレーム完了後、バスは通常状態に戻り、割り込みをかけられたノードは中断されたメッセージを再送信します。

エラーパッシブノードがバスエラーを検出した場合、そのノードはエラーパッシブフラグを送信し、引き続き、再度エラーデリミタフィールドが続けます。エラーパッシブフラグは6個の連続するリセッショングビットで構成されます。このことから、バスエラーが送信ノードもしくは実際に送信を行っている他のエラーアクティブ受信器で検出されなければ、エラーパッシブノードによるエラーフレームの送信は、ネットワーク上の他のどのノードにも影響を与えません。バスマスターがエラーパッシブフラグを生成した場合は、その結果としてビットスタッフに違反するため、これ(エラーパッシブフラグを生成すること)により他のノードがエラーフレームを生成します。エラーフレームの送信後、エラーパッシブノードは、バス通信に復帰しようとする前にバス上の6個の連続するリセッショングビットを待たねばなりません。

## 23.4.1.5 オーバーロードフレーム

オーバーロードフレームは、図 23-7 に示すように、アクティブエラーフレームと同じフォーマットを持ちます。しかし、オーバーロードフレームは、インターフレーム空間の間にのみ生成できます。このようにして、オーバーロードフレームは、エラーフレームと区別されます（エラーフレームは、メッセージの送信の間に送信されます）。オーバーロードフレームはオーバーロードフラグとそれに続くオーバーロードデリミタの 2 フィールドで構成されます。オーバーロードフラグは 6 個のドミナントビットで構成され、それに続き他のノードで生成されたオーバーロードフラグが続けます（アクティブエラーフラグについては、最大 12 個のドミナントビットを与えます）。オーバーロードデリミッタは 8 個のリセシシブビットで構成されます。オーバーロードフレームは、次の 2 つの場合にノードで生成されます。第 1 にノードが、インターフレーム空間の間にドミナントビットを検出することであり、これは不法な条件です。第 2 に、内部状態により、ノードが、まだ次のメッセージの受信開始ができないことです。次のメッセージの開始を遅らせるために、ノードは最大 2 つのシーケンシャルなオーバーロードフレームを生成することがあります。

## 23.4.1.6 インターフレーム空間

インターフレーム空間は先行する（どんな形式の）フレームと、引き続くデータもしくはリモートエラーフレームとを分けます。インターフレーム空間は、少なくとも 3 個のリセシシブビットから構成され、インターミッションと呼ばれます。これは、次のメッセージフレームの開始以前に受信ノードによるメッセージの内部処理を行う時間を与えるために用意されています。インターミッション後、バスラインは、次の送信開始までリセシシブ状態（バスアイドル）を保持します。

送信ノードがエラーパッシブ状態の場合、そのノードにより他のメッセージが送信される前に、インターフレーム空間に追加で 8 個のリセシシブビット時間が挿入されます。この時間は送信中断フィールドと呼ばれます。送信中断フィールドは、他の送信ノードがバスを制御するためには追加の遅延時間を与えます。

図 23-3: 標準データフレーム

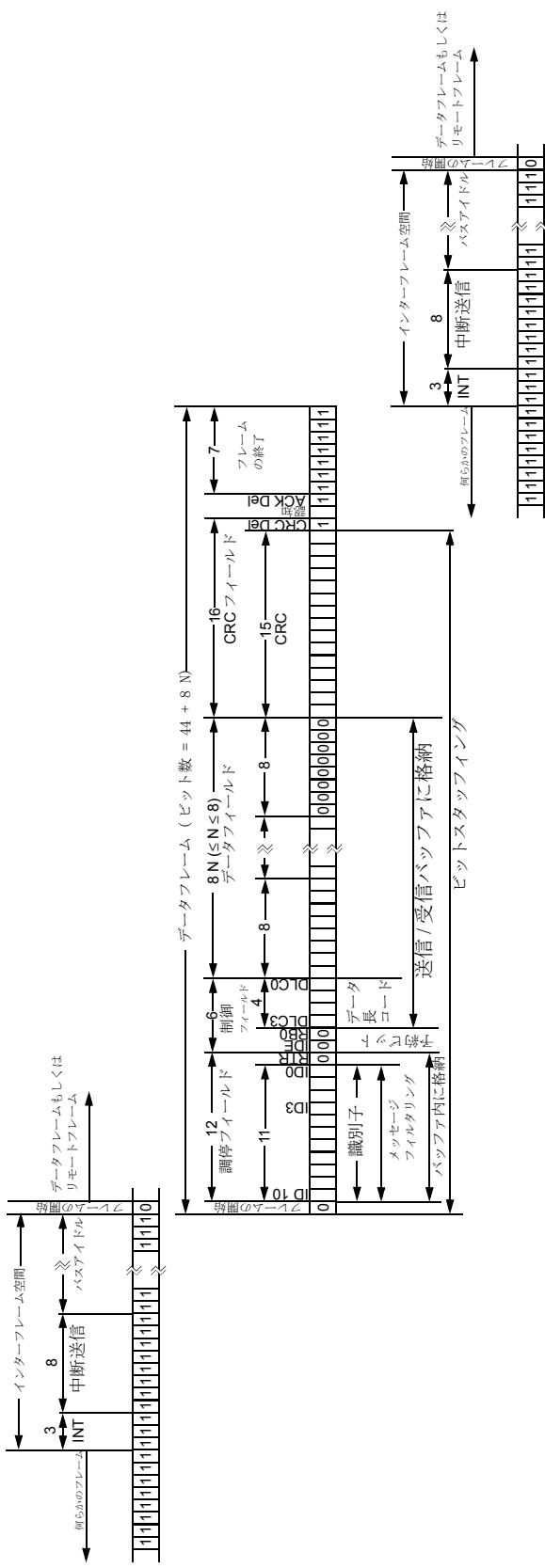
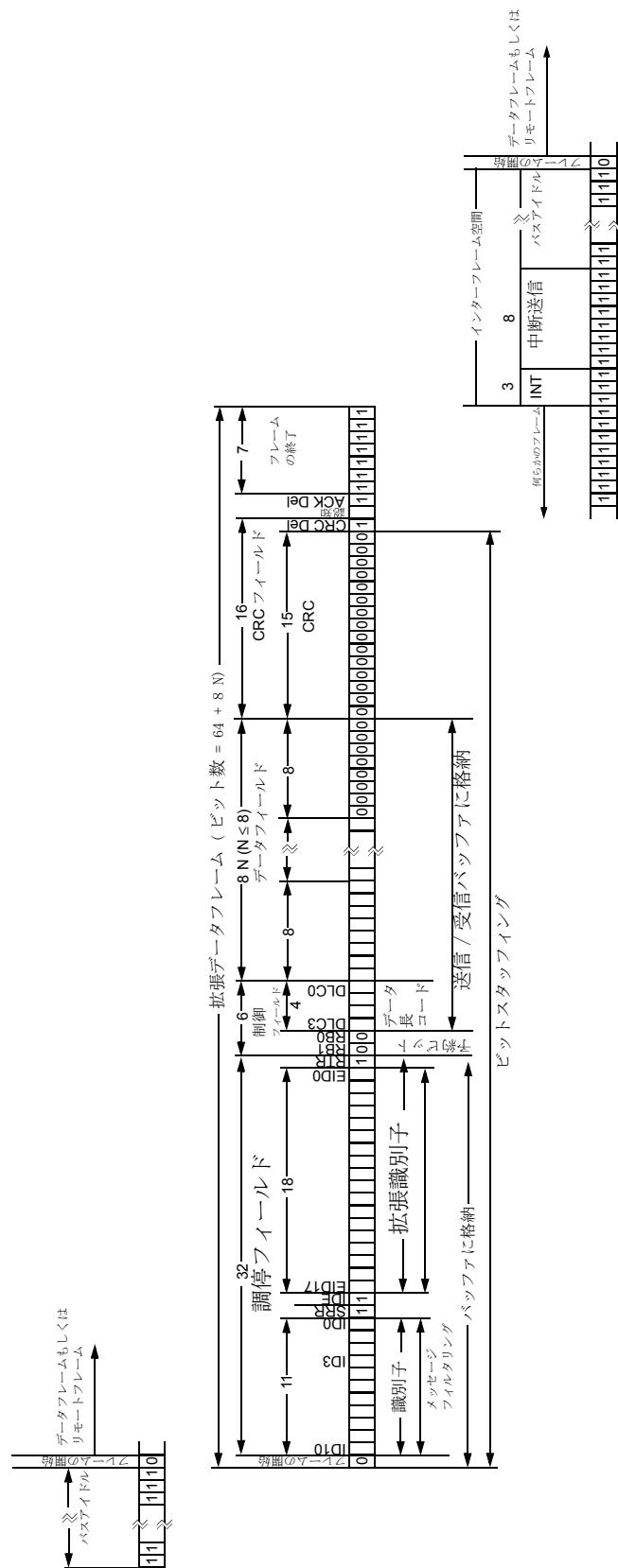


図 23-4: 拡張データフレーム



**図 23-5:** リモートデータフレーム

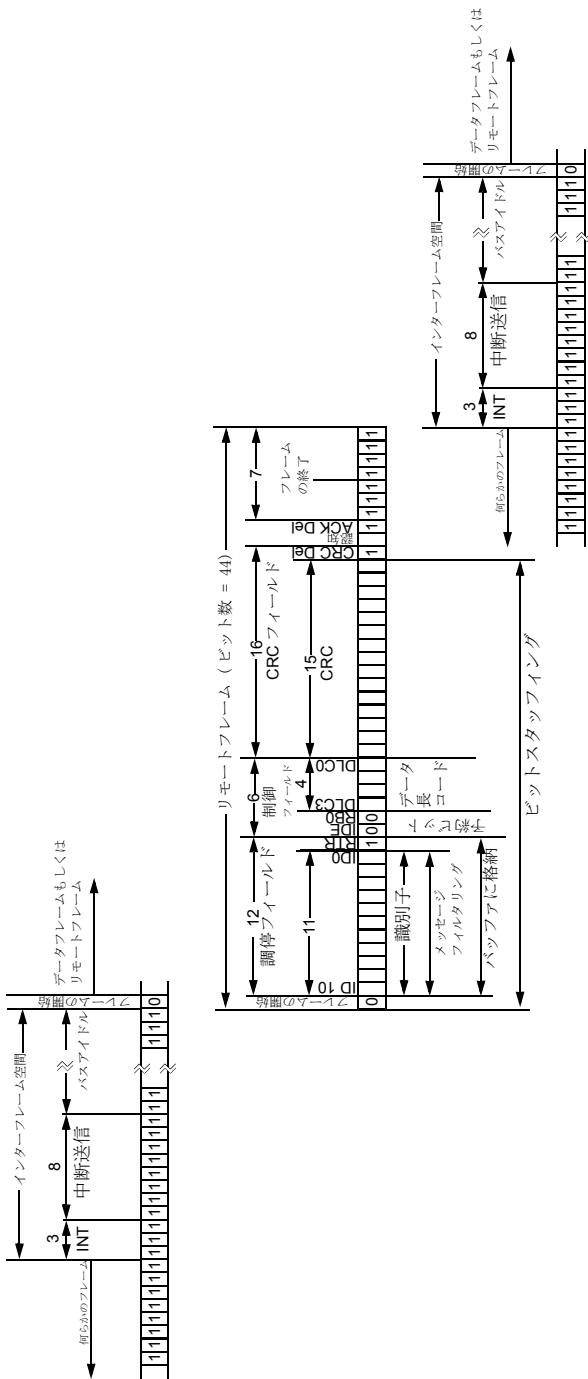


図 23-6: エラーフレーム

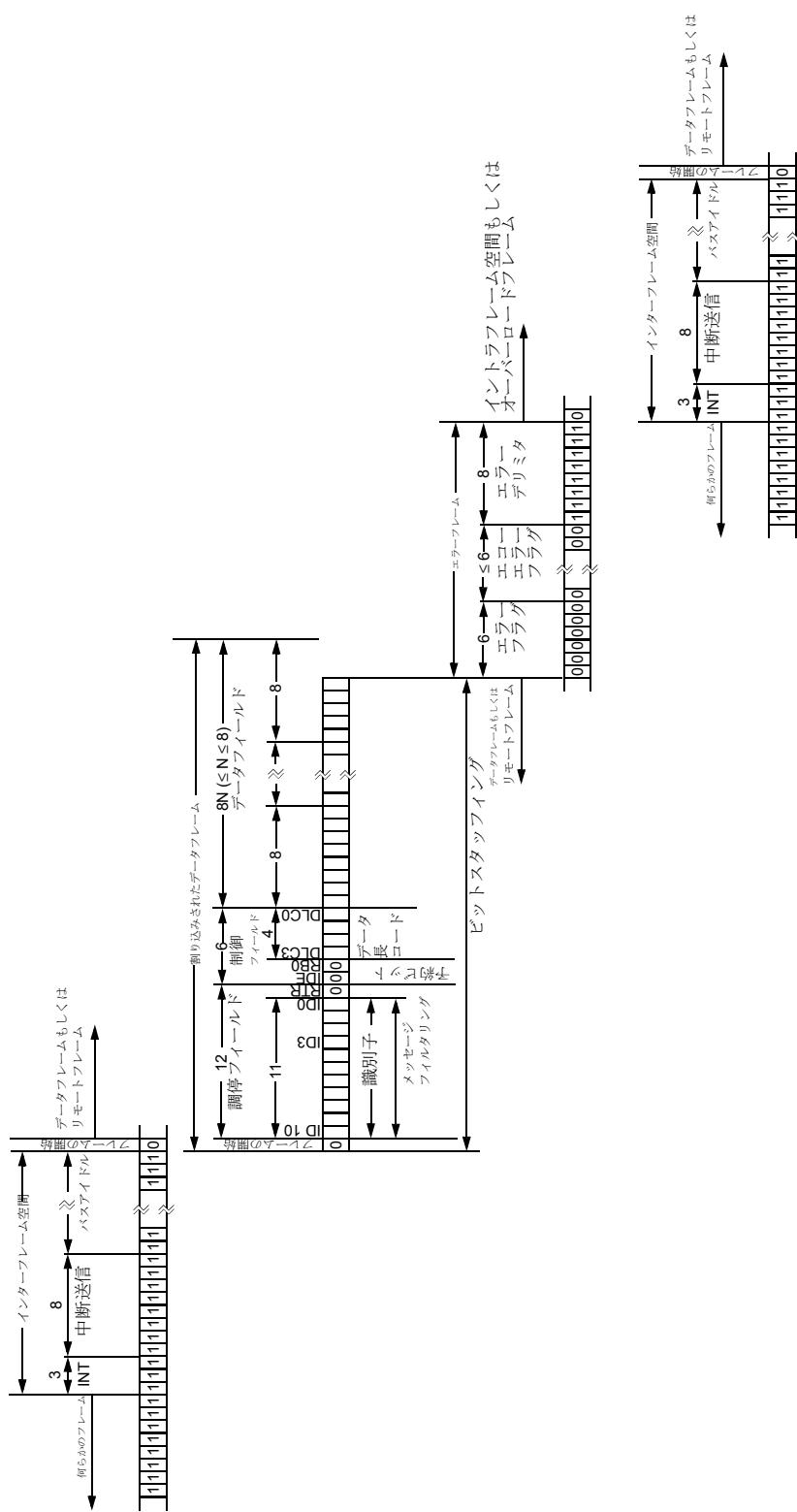
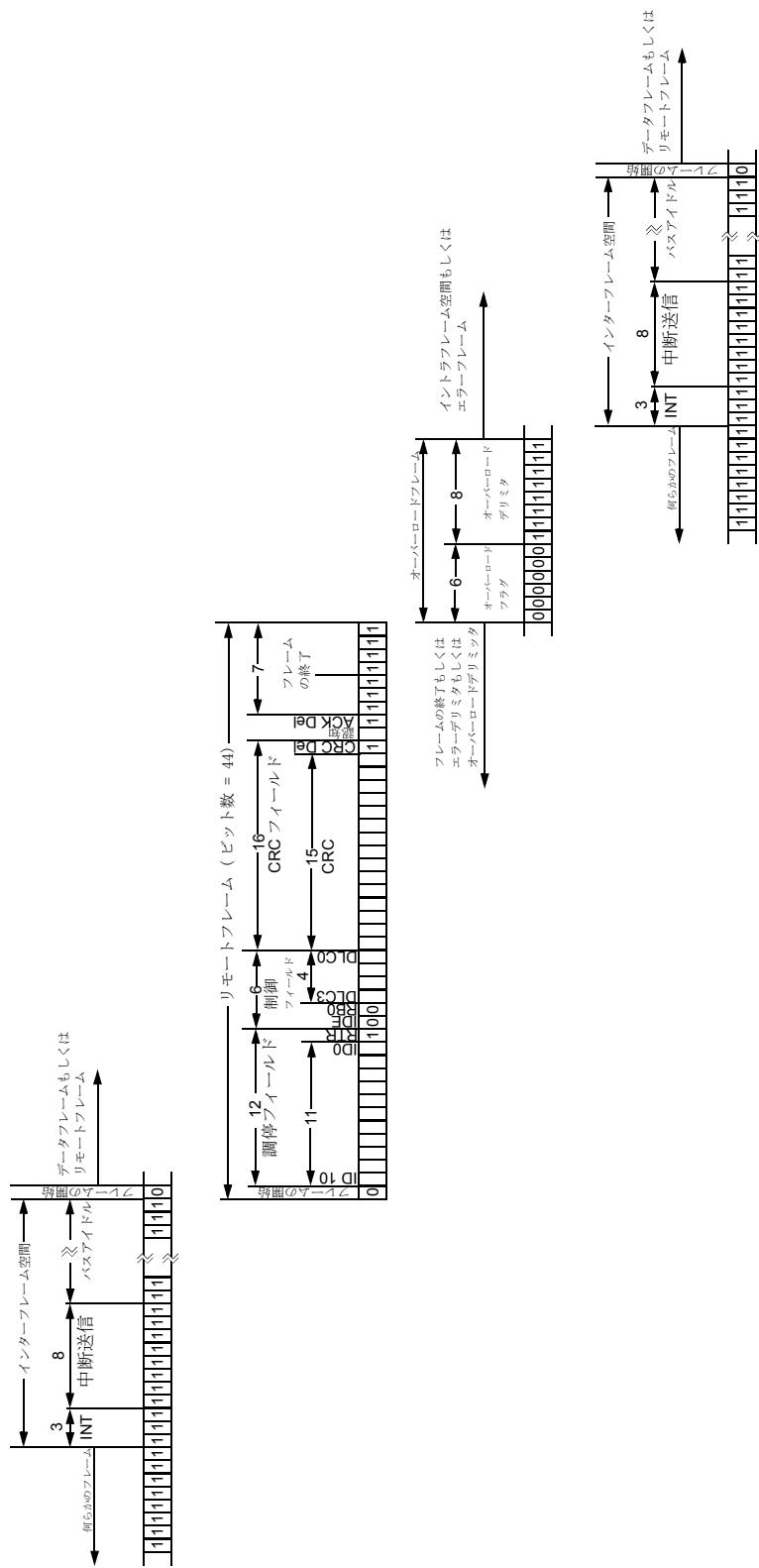


図 23-7: オーバーロードフレーム



## 23.5 CAN モジュールの動作モード

CAN モジュールは、ユーザーにより選択される、いくつかの動作モードの 1 つで動作します。これらのモードには以下があります。

- 通常動作モード
- 無効モード
- ループバックモード
- リスンオンリーモード
- 構成モード
- リスンオールメッセージモード

モードは、**REQOP<2:0>** ビット (**CiCTRL<10:8>**) を設定することで要求が出されます。モードに入ったことは、**OPMODE<2:0>** ビット (**CiCTRL<7:5>**) をモニターすることで認知できます。モジュールは、一般的に、少なくとも 11 個の連続するリセシシブビットで規定されるバスアイドル時間の間に、モードの変更が受け付けられるまで、モードや OPMODE モードの変更はありません。

### 23.5.1 通常動作モード

通常動作モードは **REQOP<2:0> = ‘000’** で選択されます。このモードでは、モジュールは動作状態で、I/O ピンは CAN バス機能として動作します。モジュールは以下のセクションで説明されるように CAN バスマッセージの送信 / 受信を行います。

### 23.5.2 無効モード

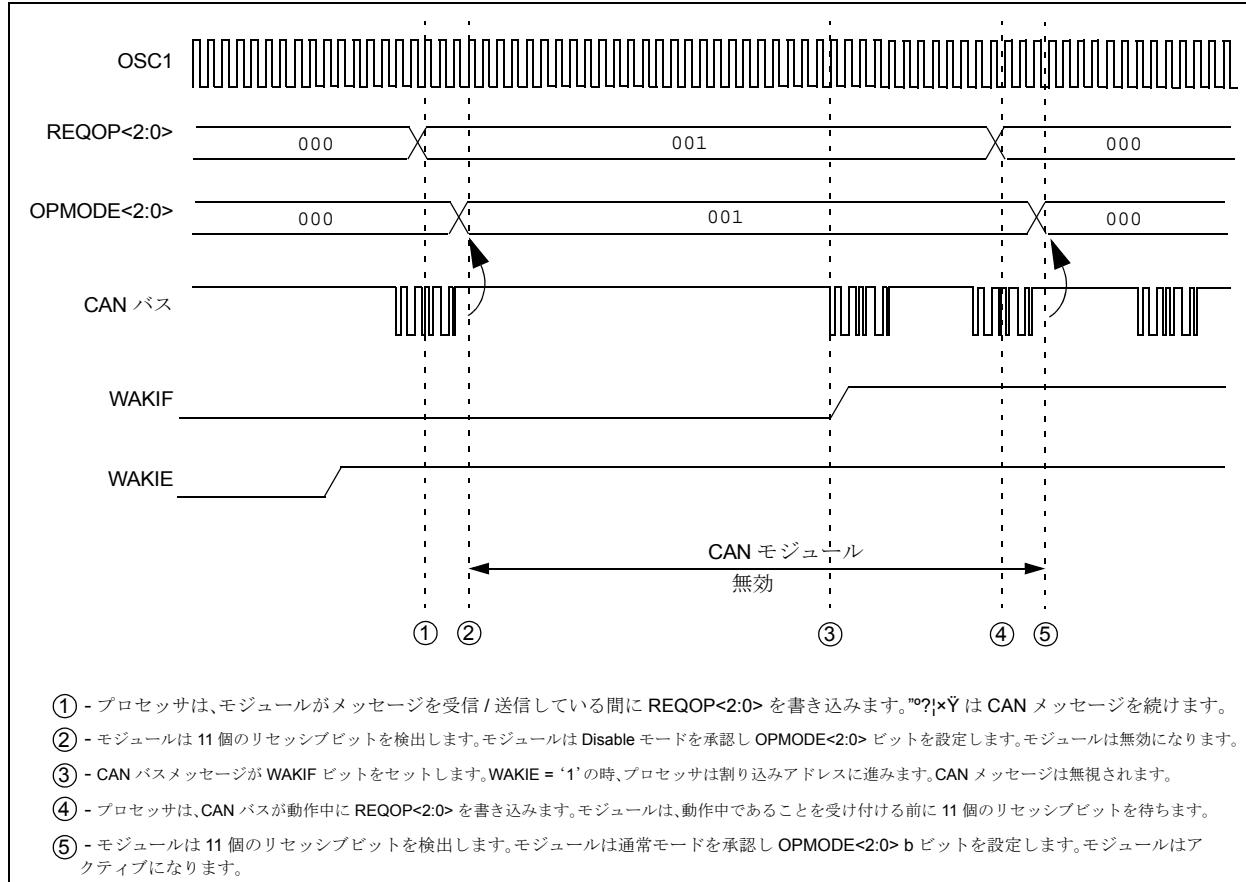
無効モードでは、モジュールは送信も受信もしません。モジュールは、バスの動作により **WAKIF** ビットを設定することができますが、中断中の割り込みはそのままであり、エラーカウンタはその値を保持します。

**REQOP<2:0>** ビット (**CiCTRL<10:8>**) = ‘001’ で、モジュールはモジュール無効モードに入ります。このモードは、モジュール有効信号をオフすることにより、他の周辺モジュールを無効にすることに似ています。これにより、モジュールがアクティブ（すなわち、メッセージの受信 / 送信を行う）になるまで、モジュール内部クロックを停止します。モジュールがアクティブのとき、モジュールは CAN バス上の 11 個のリセシシブビットを待ち、その状態をアイドルバスとして検出し、モジュール無効コマンドを受け付けます。**OPMODE<2:0>** ビットが (**CiCTRL<7:5>**) = ‘001’ の時、モジュールがうまくモジュール無効モードに入ったことを示します（図 23-8 を参照してください）。

**WAKIF** 割り込みは、モジュール無効モードでもアクティブである唯一のモジュール割り込みです。**WAKIE** ビット (**CiINTE<6>**) がセットされた場合、プロセッサは、フレームの開始 (SOF) のように、CAN バスがドミナント状態を検出するといつでも割り込みを受信します。

I/O ピンは、モジュールがモジュール無効モードに入ると、通常の I/O 機能に戻ります。

図 23-8: モジュール無効モードへの出入り



### 23.5.3 ループバックモード

ループバックモードが動作すると、モジュールは、モジュールの境界で、内部送信信号を内部受信信号に接続します。送信 / 受信ピンは、PORT I/O 機能に戻ります。

送信器は、送出メッセージ用の ACK を受信します。特殊なハードウェアにより送信器用の ACK を生成します。

### 23.5.4 リスンオンリーモード

リスンオンリーモードとループバックモードは、通常動作モードでシステムデバッグができる特殊モードです。リスンオンリーモードが動作していると、CAN バス上のモジュールは受動態になります。送信器バッファは PORT I/O 機能に戻ります。受信ピンは、CAN モジュールに対する入力ピンのままでです。受信器にとっては、エラーフラグやアノレジ信号は送出されません。エラーカウンタはこの状態では動作しません。リスンオンリーモードは、CAN バス上のボーレートを検出する時に使用されます。これを使用するためには、互いに通信する少なくとも 2 つのノードが存在することが必要です。ボーレートは、異なる値をテストすることにより経験的に検出できます。このモードはまたデータトラフィックに影響を与えないバスモニターとして使用できます。

## 23.5.5 コンフィギュレーションモード

コンフィギュレーションモードでは、モジュールは送信 / 受信をしません。エラーカウンタはクリアされ割り込みフラグは変化しません。プログラマは、他のモードではアクセス制限されるコンフィギュレーションレジスタにアクセスします。

デバイスリセットの後、CAN モジュールはコンフィギュレーションモード (OPMODE<2:0> = ‘100’ )になります。エラーカウンタはクリアされ、すべてのレジスタはリセット値になります。初期化は、REQOP<2> ビットがクリアされる前に行わなければなりません。

CAN モジュールは起動前に初期化されねばなりません。これを行うには、モジュールが コンフィギュレーションモードである場合のみ可能です。コンフィギュレーションモードは、REQOP<2> ビットをセットすることで要求されます。ステータスピット OPMODE<2> がハイの時のみ初期化が実行されます。その後、コンフィギュレーションレジスタ、アクセプタンスレジスタおよびアクセプタンスフィルタが書き込まれます。モジュールは、制御ビット REQOP<2:0> をクリアすることにより起動されます。

モジュールは、プログラミングエラーにより偶然に CAN プロトコル違反を犯すことがないよう保護します。モジュールのコンフィギュレーションを制御するすべてのレジスタは、モジュールが動作中の時は修正できません。CAN モジュールは、転送が行われている時は、コンフィギュレーションモードに入ることはできません。コンフィギュレーションモードは、以下のレジスタを保護するための錠として動作します。

- すべてのモジュール制御レジスタ
- ポーレートと割り込み構成レジスタ
- バスタイミングレジスタ
- 識別子アクセプタンスフィルタレジスタ
- 識別子アクセプタンスマスクレジスタ

## 23.5.6 リスンオールメッセージモード

リスンオールメッセージモードは、通常動作モードの特別な場合でありシステムのデバッグに使用されます。リスンオールメッセージモードが動作している場合、CAN バス上のモジュールは受動的です。受信器では、エラーフラグもしくはアクノレッジ信号は送信されません。エラーカウンタはこの状態では動作しません。フィルタは無効です。受信バッファ 0 はバス上で転送されているメッセージはどれでも受信します。このモードは、データへの影響を与えないバスモニターとして、すべてのバストラフィックを記録することに役に立ちます。

## 23.6 メッセージ受信

このサブセクションでは、CAN モジュールのメッセージ受信について説明します。

### 23.6.1 受信バッファ

CAN バスモジュールは 3 つの受信バッファを持っています。しかし、受信バッファのうちの 1 つは常に入力メッセージ用にバスをモニターしています。このバッファはメッセージアセンブリバッファ、MAB と呼ばれます。CPU からは 2 つの受信バッファ、RXB0 と RXB1 だけが見えることになり、これは基本的に一時的にプロトコルエンジンからの完全なメッセージを受信します。CPU が 1 つのバッファを処理している間、もう 1 つのバッファは受信もしくは前に受信されたデータを保持するために使用されます。

MAB は、バスラインからのスタッフビットを除かれたビットストリームを保持し、アクセプタンスマッチテストや受信バッファにフレームをパラレル転送するために、全データやリモートフレームに対し並列アクセスすることができます。これらのメッセージは、アクセプタンスフィルタ手順が一致した場合にのみ RXBn バッファに転送されます。メッセージが受信されると、RXnIF フラグ (CiINTF<0> もしくは CiINRF<1>) がセットされます。このビットは、メッセージが受信された時にのみモジュールによりセットされます。このビットは、バッファ内でのメッセージ処理が完了した時に CPU によりクリアされます。このビットは、CPU がメッセージバッファを処理するのを確実にするために積極的にバッファをロックするようにします。RXnIE ビット (CiINTE<0> もしくは CiINTE<1>) がセットされると、メッセージが受信された時に割り込みが生成されます。

受信バッファに関する 2 つのプログラマブルアクセプタンスフィルタマスクがあり、1 つのバッファに 1 つ割り当てられます。

メッセージが受信されると、FILHIT ビット (CiRX0CON<0> は受信バッファ 0 用、CiRX1CON<2:0> は受信バッファ 1 用) は、メッセージのアクセプタンス手順を示します。受信が可能なアクセプタンスフィルタの数は、受信メッセージがリモート転送要求であることを示すステータスビットと一緒に表されます。

**注：** 受信バッファ 0 の場合は、受信を有効にするには、限られた数のアクセプタンスフィルタが使用されます。ビット FILHITO (CiRX0CON<0>) は、2 つのフィルタ、RXF0 もしくは RXF1 のうちどちらかが、メッセージ受信を有効にするかを決定します。

#### 23.6.1.1 受信バッファ優先度

フレキシビリティを持つために、一つ一つの受信バッファに対応するいくつかのアクセプタンスフィルタがあります。受信バッファに対する暗黙の優先度もあります。**RXB0** がより高いプライオリティバッファであり、2つのアクセプタンスフィルタを持ちます。**RXB1** は低い優先度をもち、4つのアクセプタンスフィルタを持ちます。低い番号のアクセプタンスフィルタは、より限定的に **RXB0** の一致処理を行います。**RXB0** に対しより高い優先度を持っています。さらに、**RXB0** が有効なメッセージを持っているときに、他の有効なメッセージが受信されると、**RXB0** がオーバーランしないように、**RXB0** 用の新しいメッセージは **RXB1** に格納されます。図 23-9 に受信バッファのブロック図を示し、図 23-10 には受信動作のフローチャートを示します。

図 23-9: 受信バッファ

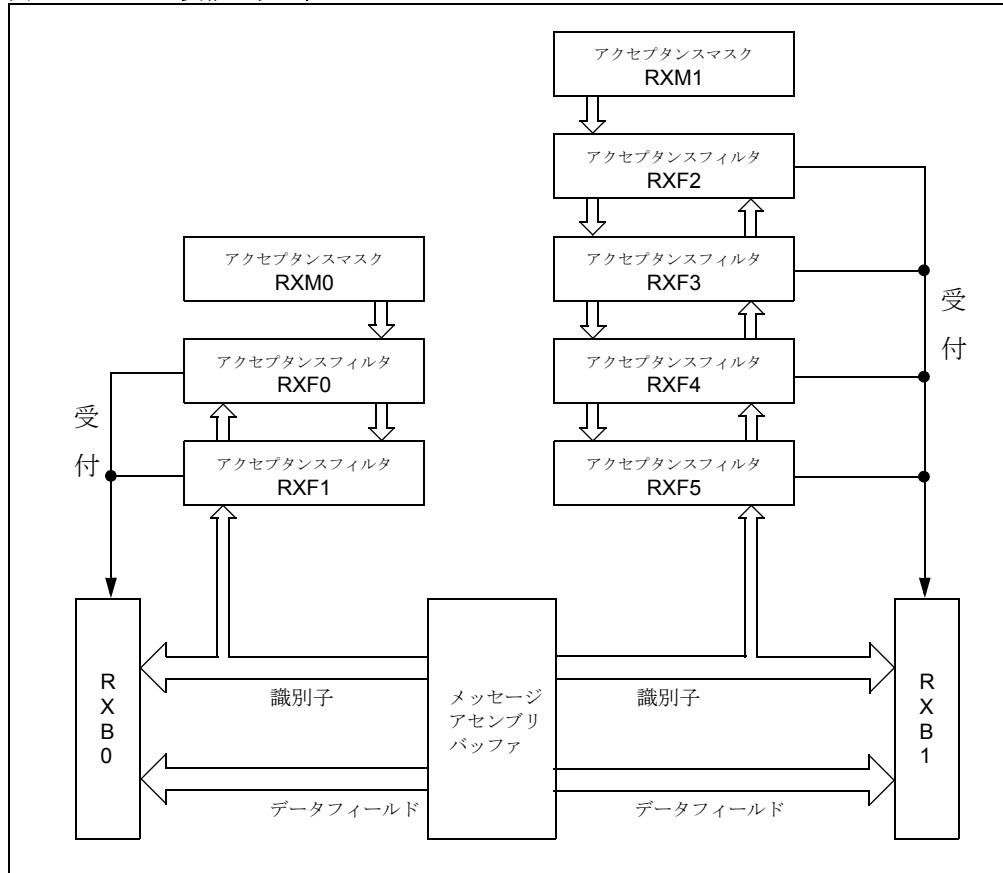
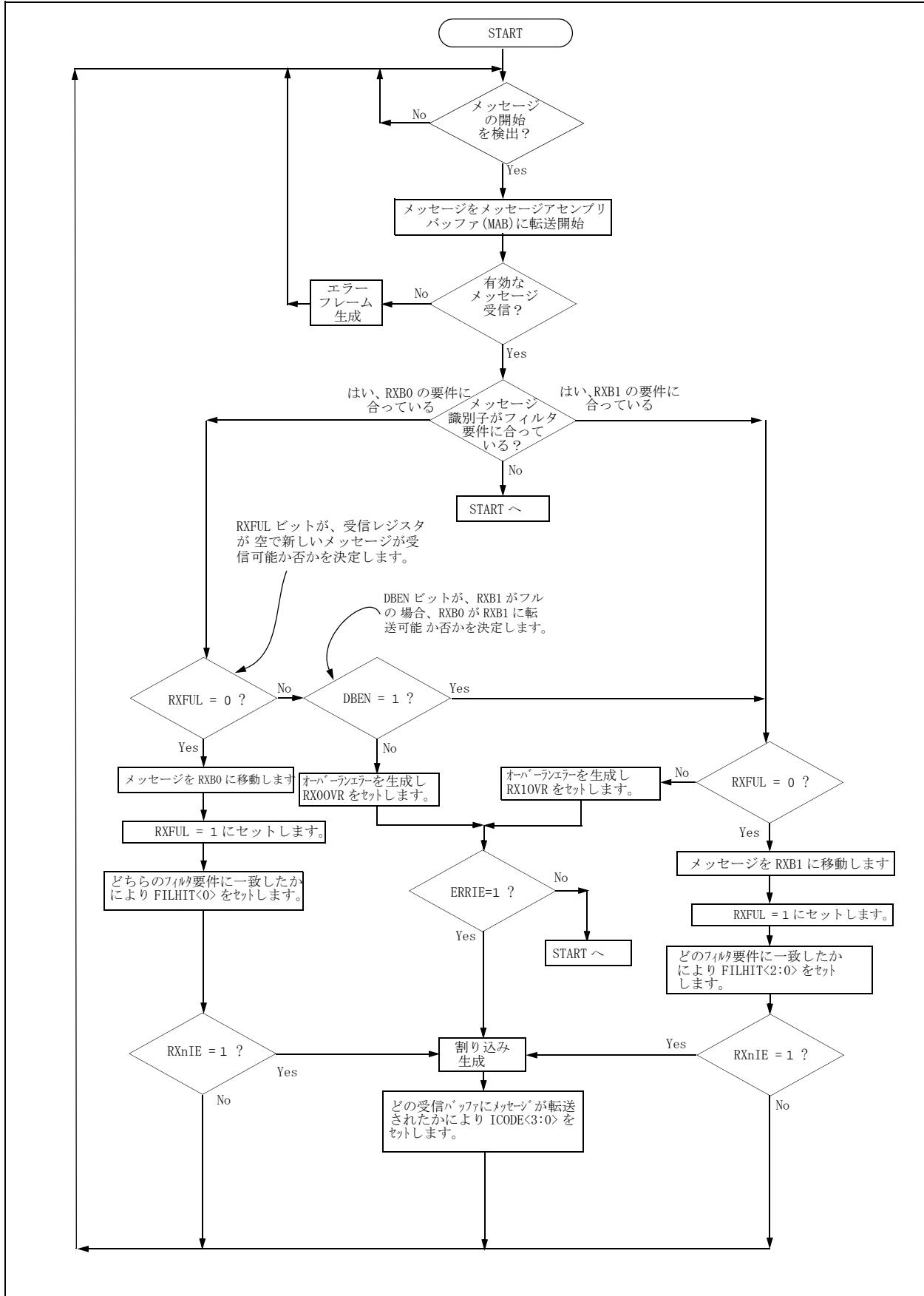


図 23-10: 受信フローチャート



### 23.6.2 メッセージアクセプタンスフィルタ

メッセージアクセプタンスフィルタとマスクは、メッセージアセンブリバッファ内のメッセージがどちらの受信バッファに転送されるかどうかを決定するために使用されます。有効なメッセージがメッセージアセンブリフィルタ (MAB) に受信されると、メッセージの識別子フィールドがフィルタ値と比較されます。一致すると、そのメッセージは適切な受信バッファに転送されます。フィルタマスクは、識別子内のどのビットがフィルタとの検査をされるかを決定するために使用されます。真理値表を表 23-3 に示しますが、この表は、メッセージを受信バッファに転送するかどうかを決定するために、どのようにして識別子内のビットがマスクやフィルタと比較されるかを示します。マスクビットは基本的にどのビットをフィルタに適用すべきかを決定します。どのマスクビットもゼロに設定されている場合は、フィルタビットに関わらずそのビットは自動的に受け付けられます。

表 23-3: フィルタ / マスク真理値表

マスクビット n	フィルタビット n	"Øæx°? 識別子ビット	受付もしくは拒絶ビット n
0	x	x	受付
1	0	0	受付
1	0	1	拒絶
1	1	0	拒絶
1	1	1	受付

凡例: x = 無視

#### 23.6.2.1 識別子モード選択

EXIDE 制御ビット (CiRXFnSID<0>) および MIDE 制御ビット (CiRXMnSID<0>) は、標準もしくは拡張識別子用のアクセプタンスフィルタを有効にします。アクセプタンスフィルタは、入力メッセージの RXIDE ビットを見て、識別子との比較方法を決定します。RXIDE ビットがクリアされている場合は、メッセージは標準フレームです。RXIDE ビットがセットされている場合は、メッセージは拡張フレームです。

フィルタ用の MIDE 制御ビットがセットされている場合は、フィルタの識別形式は、フィルタ用の EXIDE 制御ビットにより決定されます。EXIDE 制御ビットがクリアされていると、フィルタは標準識別子を受付ます。EXIDE ビットがセットされていると、フィルタは拡張識別子を受け付けます。ほとんどの CAN システムでは、標準識別子のみを用いるか、拡張識別子のみを用います。

フィルタ用の MIDE 制御ビットがクリアされている場合、フィルタビットが一致すればフィルタは標準・拡張両方の識別子を受け付けます。このモードは、同じバス上に標準・拡張両方の識別子をサポートする CAN システムで用いられます。

#### 23.6.2.2 FILHIT ステータスピット

図 23-9 の受信バッファブロック図に示すように、RXM0 マスクを持つ RXF0 および RXF1 フィルタは RXB0 と関連付けられます。フィルタ RXF2、RXF3、RXF4 および RXF5 と、マスク RXM1 は RXB1 と関連付けられます。フィルタが一致し、メッセージが受信バッファに転送されると、メッセージ受信を有効にしたフィルタ番号は FILHIT ビットを経由して CiRXnCON レジスタに示されます。CiRX0CON レジスタは 1 つの FILHIT ステータスピットを含み、RXF0 もしくは RXF1 フィルタのどちらがメッセージ受信を有効にしたかを示します。CiRX1CON レジスタは FILHIT<2:0> ビットを含みます。これらは表 23-4 に示すようにコード化されます。

表 23-4: アクセプタンスフィルタ

FILHIT<2:0>	アクセプタンスフィルタ	コメント
000 <sup>(1)</sup>	RXF0	DBEN = 1 の時のみ
001 <sup>(1)</sup>	RXF1	DBEN = 1 の時のみ
010	RXF2	—
011	RXF3	—
100	RXF4	—
101	RXF5	—

注 1: DBEN ビットがセットされている場合のみ有効。

DBEN ビット (CiRX0CON<2>) により FILHIT ビットの RXF0 および RXB1 のフィルタ一致が、RXB0 のデータによるものか、オーバーランした RXB1 のデータによるものかを区別します。

111 = アクセプタンスフィルタ 1 (RXF1)

110 = アクセプタンスフィルタ 0 (RXF0)

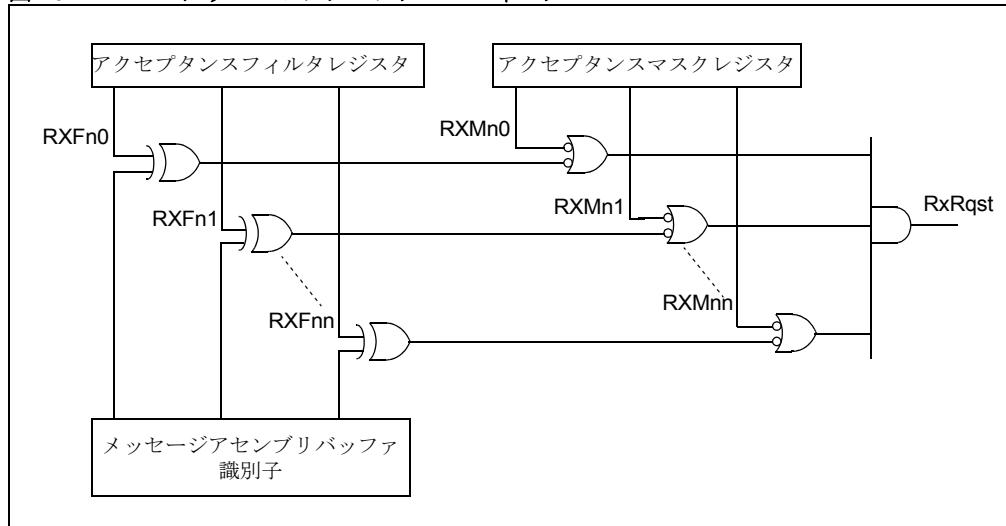
001 = アクセプタンスフィルタ 1 (RXF1)

000 = アクセプタンスフィルタ 0 (RXF0)

DBEN ビットがクリアされていると、6 つのフィルタに対応する 6 つのコードがあります。DBEN ビットがセットされていると、6 つのフィルタに対応する 6 つのコード、プラス、RXB1 にオーバーランした RXF0 と RXF1 に対応する 2 つの追加コードがあります。

1 つ以上のアクセプタンスフィルタが一致した場合、FILTHIT ビットは一致したフィルタの最小バイナリ値を符号化します。例えば、フィルタ 2 とフィルタ 4 が一致した場合は、FILTHIT は 2 の値をコード化します。これは基本的に、低い番号のアクセプタンスフィルタが優先度を持つように優先度付けをします。図 23-11 は、メッセージアクセプタンスフィルタのブロック図を示します。

図 23-11: メッセージアクセプタンスフィルタ



### 23.6.3 受信オーバーラン

メッセージアセンブリバッファ (MAB) が有効な受信メッセージを組み立てた時、メッセージがアクセプタンスフィルタを経由して受け付けられた時、およびそのフィルタに対応する受信バッファが、前のメッセージの処理を完了していない時、オーバーラン状態が発生します。

オーバーランエラーフラグ、RXnOVR (CiINTF<15> もしくは CiINTF<14>) と、ERRIF ビット (CiINTF<5>) がセットされ、MAB 内のメッセージは廃棄されます。オーバーラン状態では、モジュールは CAN バスと同期したままでメッセージを転送できますが、オーバーフローバッファに割り当てられたすべての入力メッセージは廃棄されます。

DBEN ビットがクリアされると、RXB1 と RXB0 は独立に動作します。この場合、RXB0 が未読のメッセージを含んでいるときの RXB0 用のメッセージは RXB1 に転用することはできず、RX0 OVR ビットがセットされます。

DBEN ビットがセットされていると、RXB0 のオーバーランは違った風に処理されます。RXB0 用の有効なメッセージが受信され、RXB0 がフルであることを示す RXFUL = 1 (CiRX0CON<7>) の時、および RXB1 が空であることを示す RXFUL = 0 (CiRX1CON<7>) の場合、RXB0 用のメッセージは RXB1 に転送されます。RXB0 のオーバーランエラーは生成されません。しかし、有効なメッセージが RXB0 用に受信したとき、RXB0 と RXB1 の両方がフルであることを示す RXFUL = 1 かつ RXFUL = 1 の場合、メッセージは失われ RXB1 のオーバーランがセットされます。

DBEN ビットがクリアされていると、6 つのフィルタに対応する 6 つのコードがあります。DBEN ビットがセットされていると、6 つのフィルタに対応する 6 つのコード、プラス、RXB1 にオーバーランした RXF0 と RXF1 に対応する 2 つの追加コードがあります。これらのコードは表 23-5 で与えられます。

表 23-5: バッファ受信とオーバーフロー真理値表

メッセージがフィルタ 0 もしくは 1 と一致	メッセージがフィルタ 2,3,4,5 と一致	RXFUL0 ビット	RXFUL1 ビット	DBEN ビット	アクション	結果
0	0	X	X	X	無し	メッセージ受信無し
0	1	X	0	X	MAB $\neq$ RXB1	RXB1 利用可のときの RXB1 へのメッセージ
0	1	X	1	X	MAB は廃棄 RX1OVR = 1	RXB1 がフルのときの RXB1 へのメッセージ
1	0	0	X	X	MAB $\neq$ RXB0	RXB0 利用可のときの RXB0 へのメッセージ
1	0	1	X	0	MAB は廃棄 RX0OVR = 1	RXB0 がフルのときの RXB0 へのメッセージ DBEN は無効
1	0	1	0	1	MAB $\neq$ RXB1	RXB0 がフルのときの RXB0 へのメッセージ DBEN は有効 RXB1 が利用可
1	0	1	1	1	MAB は廃棄 RX1OVR = 1	RXB0 がフルのときの RXB0 へのメッセージ DBEN は無効, RXB1 はフル
1	1	0	X	X	MAB $\neq$ RXB0	RXB0, RXB1 用メッセージ, RXB0 が利用可
1	1	1	X	0	MAB は廃棄 RX0OVR = 1	RXB0, RXB1 用メッセージ, RXB0 はフル, DBEN は無効

凡例 : X = 無視

### 23.6.4 E リセットの影響

どのリセットでも、CAN モジュールは初期化されねばなりません。すべてのレジスタはリセット値に応じてセットされます。受信メッセージの内容は失われます。初期化についてはセクション 23.5.5「コンフィギュレーションモード」で説明されます。

### 23.6.5 受信エラー

CAN モジュールは以下の受信エラーを検出します。

- 巡回冗長チェック (CRC) エラー
- ビットスタッフィングエラー
- 無効メッセージ受信エラー

これらのエラーは割り込みを発生しません。しかし、受信エラーカウンタは、これらのエラーのうちの 1 つが発生すると 1 づつ増加されます。RXWAR ビット (CiINTF<9>) は、受信エラーカウンタが CPU の警告リミット 96 に達したことを示し、割り込みが発生します。

#### 23.6.5.1 巡回冗長チェック (CRC) エラー

巡回冗長チェックにより、送信器は、フレームの開始からデータフィールドの終わりまでのビットシーケンスに対して特別なチェックビットを計算します。この CRC シーケンスは、CRC フィールドで送信されます。受信ノードでも、同じ式を用いて CRC シーケンスを計算し、受信シーケンスとの比較を行います。不一致が検出されると、CRC エラーが発生しエラーフレームが生成されます。メッセージは繰り返されます。受信エラー割り込みカウンタは 1 づつ増加されます。エラーカウンタが閾値を越えたとき初めて割り込みが生成されます。

#### 23.6.5.2 ビットスタッフィングエラー

フレームの開始から CRC デリミッタ間で、同じ極性を持つ 6 つの連続するビットが検出された場合、ビットスタッフィングルールに違反することになります。ビットスタッフィングエラーが発生しエラーフレームが生成されます。メッセージは繰り返されます。このイベントでは割り込みは生成されません。

#### 23.6.5.3 無効メッセージ受信エラー

メッセージの受信中にどのエラーが発生しても、エラーは IVRIF ビット (CiINTF<7>) により表示されます。このビットは、リスンオンリーモードのデバイスでの自動ボーレート検出用に(オプションとして割り込みも一緒に) 使用されます。このエラーはアクションを取る必要があるか否かは示しませんが、CAN バスでエラーがあったことを示しています。

#### 23.6.5.4 受信エラーカウンタを変更する規則

受信エラーカウンタは以下の規則に従い変更されます。

- 受信器がエラーを検出すると、受信エラーカウンタが 1 増加しますが、アクティブエラーフラグもしくはオーバーロードフラグの送信中にビットエラーが検出された場合はあてまりません。
- 受信器が、エラーフラグを送信後の最初のビットとして「ドミナント」ビットを検出すると、受信エラーカウンタは 8 増加します。
- 受信器が、アクティブエラーフラグもしくはオーバーロードフラグを送信中にビットエラーを検出すると、受信エラーカウンタは 8 増加します。
- どのノードも、アクティブエラーフラグ、パッシブエラーフラグもしくはオーバーロードフラグの送信後、最大 7 つの連続する“ドミナント”ビットまでは許容します。(アクティブエラーフラグもしくはオーバーロードフラグの場合)14 番目の連続する“ドミナント”ビットを検出後、もしくはパッシブエラーフラグに続く 8 番目の連続する“ドミナント”ビットを検出後、それぞれ 8 つの追加の連続する“ドミナント”ビットのシーケンスの後に、すべての送信器はその送信エラーカウンタを増加しすべての受信器は受信エラーカウンタを 8 増加します。
- メッセージの受信がうまくいった (ACK スロットまでエラー無しで受信し ACK ビットの送信がうまくいった) 後で、受信エラーカウンタが 1 と 127 の間の場合、受信エラーカウンタは 1 つ減少します。受信エラーカウンタが ‘0’ の場合は、‘0’ のままでです。受信エラーカウンタが 127 以上の場合、119 と 127 の間の値に変更されます。

## 23.6.6 受信割り込み

いくつかの割り込みはメッセージ受信にリンクしています。受信割り込みは 2 つの別のグループに分割できます。

- 受信エラー割り込み
- 受信割り込み

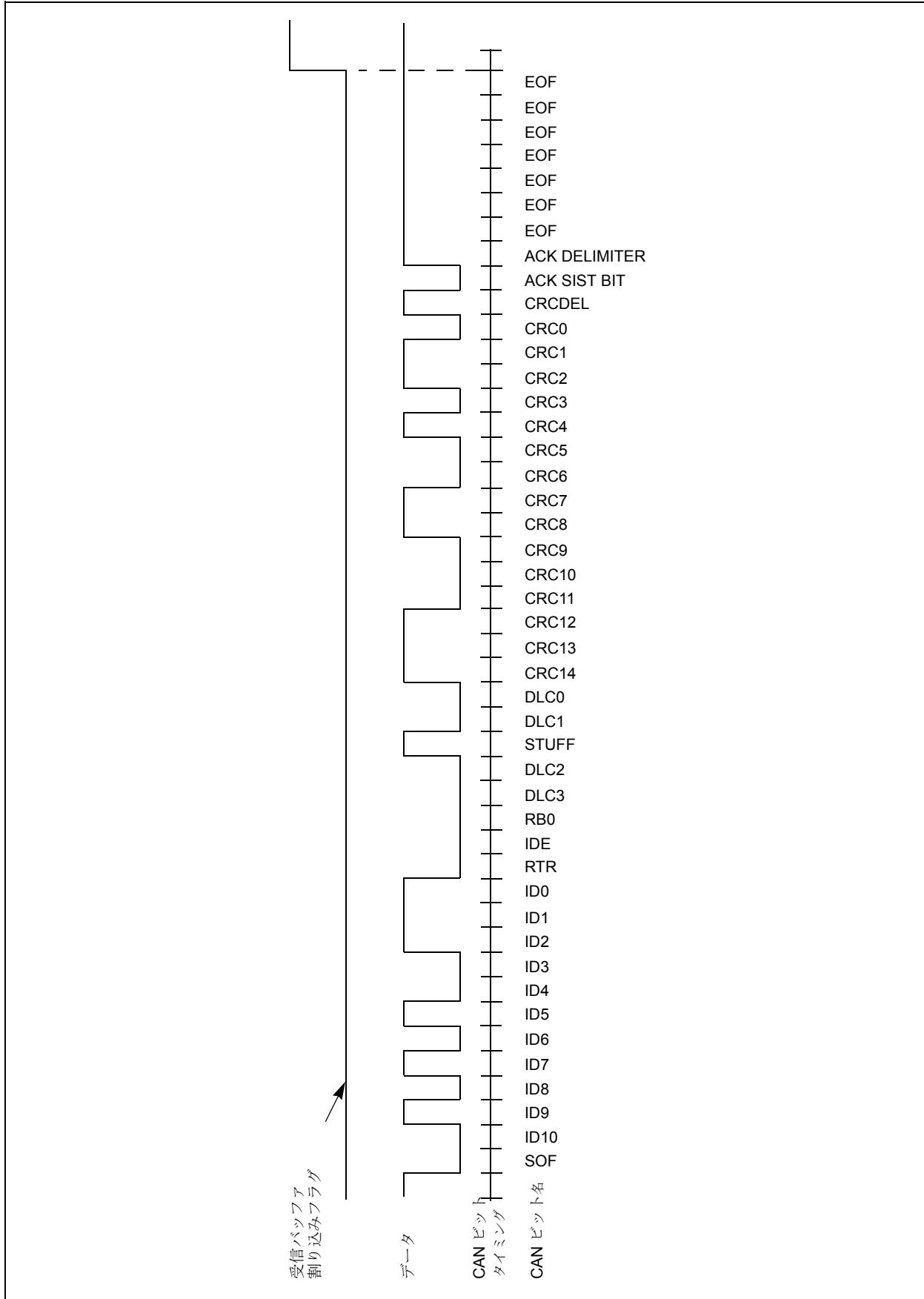
### 23.6.6.1 受信割り込み

最新のメッセージがうまく受信され、受信バッファのうちの 1 つに転送されます。この割り込みは、フレーム終了(EOF)フィールドを受信した直後に起動されます。RxnIF フラグを読み込むことにより、どちらの受信バッファが割り込みを発生させたかがわかります。図 23-12 に受信割り込みフラグ RxnIF がセットされる時の状況を示します。

### 23.6.6.2 ウェイクアップ割り込み

ウェイクアップ割り込みシーケンスは、セクション 23.13.1 「SLEEP モード時の動作」で説明されます。

図 23-12: 受信バッファ割り込みフラグ



## 23.6.6.3 受信エラー割り込み

受信エラー割り込みは **ERRIF** ビット (**CiINTF<5>**) で表示されます。このビットは、エラー状態が発生したことを示します。エラーのソースは、CAN 割り込みステータスレスタ **CiINTF** 内のビットをチェックすることで決定されます。このレジスタ内のビットは受信・送信エラーに関係しています。以下のシーケンスは、どのフラグが受信エラーに関係しているかを示します。

### 23.6.6.3.1 無効メッセージ受信割り込み

最新のメッセージの受信中にどんなエラーが発生しても、エラーは **IVRIF** ビット (**CiINTF<7>**) により表示されます。発生したエラが何であるかはわかりません。このビットは、リスンオンリーモードのデバイスでの自動ボーレート検出用に（オプションとして割り込みも一緒に）使用されます。このエラーは、アクションを取る必要があるか否かは示しませんが、CAN バス上でエラーが発生したことを示します。

### 23.6.6.3.2 受信オーバーラン割り込み

**RXnOVR** ビット (**CiINTF<15>, CiINTF<14>**) は、受信バッファでオーバーラン状態が発生したことを示します。オーバーラン状態は、メッセージアセンブリバッファ (**MAB**) が有効な受信メッセージを受信し、メッセージがアクセプタンスフィルタ経由で受け付けられたが、フィルタに対応した受信バッファで前のメッセージのクリアができていない場合に発生します。オーバーフロー エラー割り込みがセットされ、メッセージは廃棄されます。オーバーラン状態では、モジュールは CAN バスとの同期は取れたままで、メッセージの送信・受信が可能です。

### 23.6.6.4 受信器警告割り込み

**RXWAR** ビット (**CiINTF<8>**) は、受信器エラーカウンタが CPU 警告リミット値 96 に達したことを示します。**RXWAR** が ‘0’ から ‘1’ に遷移することにより、エラー割り込みフラグ **ERRIF** がセットされます。このビットは手動ではクリアされません、なぜなら、受信器エラーカウンタが CPU 警告リミット値 96 に達したことを、継続して示す必要があるからです。**RXWAR** ビットは、受信エラーカウンタが 95 以下になると自動的にクリアされます。**ERRIF** ビットは手動でクリアされ、**RXWAR** ビットに影響を与えることなく割り込みサービスルーチンが起動されるようにします。

### 23.6.6.5 受信器エラーパッシブ

**RXEP** ビット (**CiINTF<11>**) は、受信エラーカウンタがエラー不活性リミット 127 を越えたことを示し、モジュールはエラーパッシブ状態になります。**RXEP** ビットが ‘0’ から ‘1’ に遷移することにより、エラー割り込みフラグがセットされます。**RXEP** ビットは手動ではクリアされません、なぜなら、バスがエラーパッシブ状態であることを継続して示す必要があるからです。**RXEP** ビットは、受信エラーカウンタが 127 以下になると自動的にクリアされます。**ERRIF** ビットは手動でクリアされ、**RXEP** ビットに影響を与えることなく割り込みサービスルーチンが起動されるようにします。

## 23.7 送信

このサブセクションでは、CAN メッセージの送信のために、CAN モジュールがどのように使用されるかを説明します。

### 23.7.1 実時間通信と送信メッセージバッファリング

実時間で効果的にメッセージを送信するアプリケーションのために、CAN ノードはバスを占有しなければなりませんので、ノードメッセージは、バスのアービトレーションを勝ち取るくらい高い優先度を持つようにされています。ノードが 1 つしか送信バッファをもたない場合、まずメッセージを送信し、それから CPU がバッファにデータを転送する間はバスを解放します。ノードが 2 つの送信バッファを持つ場合は、1 つのバッファは送信に使用され、2 つ目のバッファにはデータが転送されます。しかし、最初のメッセージが完了する前に 2 つ目のバッファにデータが転送されることを確実にするため、CPU はバスの動作を細かく追跡し続ける必要があります。

典型的な応用例では、3 つのメッセージ送信バッファが必要です。3 つのバッファにより、1 つ目のバッファは送信し、2 つ目のバッファは最初のバッファの送信が完了次第送信するために準備でき、3 つ目のバッファは CPU によりデータが転送されます。これにより、バスとの同期を保つためのソフトウェアの負荷を軽減できます(図 23-13 を参照してください)。

さらに、3 つのバッファにより、ある程度、送出メッセージの優先度付けを行うことができます。例えば、アプリケーションソフトウェアは、第 3 のバッファについて処理をしている間に第 2 のバッファ内でメッセージを待ち行列にできます。アプリケーションソフトは、第 3 のバッファに行くメッセージが、すでに待ち行列にあるものよりもより高い重要度であることを必要とするかもしれません。2 つしかバッファが無い場合は、待ち行列のメッセージは消去され、第 3 のメッセージと入れ替えられねばなりません。メッセージを消去することはバスの制御を失わせることになるかもしれません。3 つバッファがあれば、第 2 および第 3 のメッセージ両方を待ち行列にして、第 3 のメッセージの方が第 2 のメッセージより高い優先度を持つことをモジュールに指示することができます。第 3 のメッセージは、次に送信され、その後に第 2 のメッセージの送信ができます。

### 23.7.2 送信メッセージバッファ

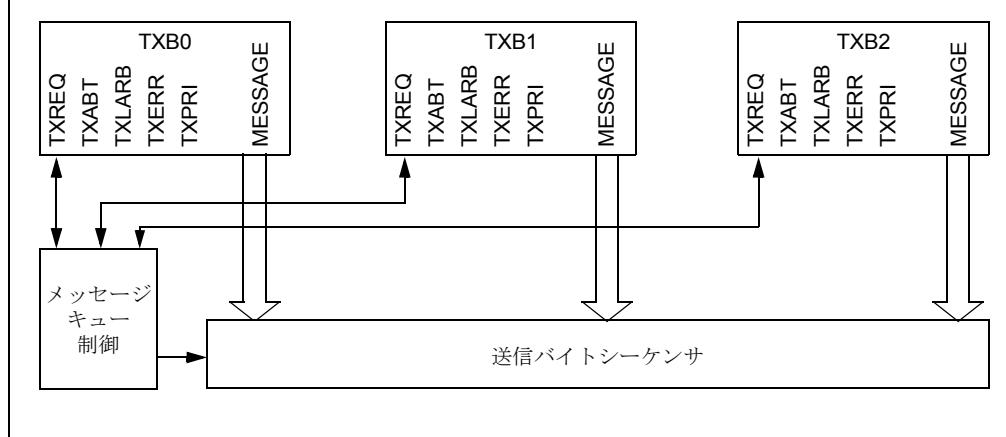
CAN モジュールは 3 つの送信バッファを持っています。1 つ 1 つは 14 バイトのデータを占領します。バイトの 8 つは、送信メッセージの最大 8 バイトです。5 バイトは標準および拡張識別子とその他のメッセージアービトレーション情報を保持します。

最後のバイトは、1 つ 1 つのメッセージに関連した制御ビットです。このバイト内の情報は、メッセージが送信される条件を決定し、メッセージの送信状態を表示します。

TxnIF ビット (*CiINTF<2>, CiINTF<3>* もしくは *CiINTF<4>*) がセットされ、TXREQ ビット (*CiTXnCON<3>*) がクリアされると、メッセージバッファが送信を完了したことを示します。それから CPU は送信されるべきメッセージの内容をメッセージバッファに転送します。少なくとも、標準識別子 *CiTXnSID* に転送されなければなりません。データバイトがメッセージ内に存在する場合は、*TXBnDm* レジスタに転送します。メッセージが拡張識別子を使用する場合は、*CiTXnEID* レジスタと *EID<5:0>* ビット (*CiTXnDLC<15:10>*) に転送し、TXIDE ビットをセット (*CiTXnSID<0>*) します。

メッセージの送信前に、ユーザーは TXnIE ビット (*CiINTE<2>, CiINTE<3>* もしくは *CiINTE<4>*) を初期化し、メッセージが送信される時に割り込みを有効にするか無効にするかを決めなければなりません。ユーザーはまた送信優先度を初期化しなければなりません。図 23-13 に、送信バッファのブロック図を示します。

図 23-13: 送信バッファ



### 23.7.3 送信メッセージの優先度

送信優先度は、送信待ちのメッセージを持つそれぞれのノード内で優先付けされます。SOF(フレームの開始)の送信前に、送信準備のできたすべてのバッファの優先度が比較されます。一番高い優先度を持った送信バッファが最初に送信されます。例えば、送信バッファ0が送信バッファ1よりも高い優先度を設定されている場合、バッファ0が最初に送信されます。2つのバッファが同じ優先度設定されている場合、一番高いアドレスを持ったバッファが送信されます。例えば、送信バッファ1が送信バッファ0と同じ優先度設定の場合、バッファ1が最初に送信されます。4つの送信優先度レベルがあります。特定のメッセージバッファのTXPRI<1:0> (CiTXnCON<1:0>) が ‘11’ に設定されている場合、そのバッファが最高の優先度を持ちます。特定のメッセージバッファのTXPRI<1:0> が ‘10’ もしくは ‘01’ に設定されている場合、そのバッファは中間の優先度を持ちます。特定のメッセージバッファのTXPRI<1:0> が ‘00’ に設定されている場合、そのバッファは最低の優先度を持ちます。

### 23.7.4 メッセージの送信

メッセージの送信を起動するには、TXREQ ビット (CiTXnCON<3>) をセットする必要があります。CAN バスモジュールは、TXREQ ビット設定と SOF 時間の時間衝突を解決し、優先度が変わった場合、SOF より前に問題が正しく解決されることを保証します。TXREQ がセットされると、TXABT (CiTXnCON<6>)、TXLARB (CiTXnCON<5>) および TXERR (CiTXnCON<4>) フラグビットが、モジュールによりクリアされます。

TXREQ ビットをセットすることによっては、メッセージ送信は実際には開始されず、メッセージバッファが送信のために待ち行列にあることを示すフラグをセットします。送信は、モジュールバスが SOF 用に利用できることを検出した時に開始されます。モジュールはそれから、最高位の優先度を持つと決定されたメッセージの送信を開始します。

送信が最初の実行でうまく完了すると、TXREQ ビットはクリアされ、TxnIE ビット (CiINTE<2>、CiINTE<3>、CiINTE<4>) がセットされている場合は、割り込みが発生します。

メッセージの送信が失敗した場合は、他の条件フラグがセットされ TXREQ はセットされた状態のままで、メッセージがまだ送信待ちであることを示します。メッセージの送信を試みたときエラー状態に出くわした場合は、TXERR ビット (CiTXnCON<4>) がセットされます。この場合、エラー条件は割り込みも発生させます。メッセージの送信を試みたときアービトレーション権を失った場合、TXLARB ビット (CiTXnCON<5>) がセットされます。この場合、アービトレーション権を失ったことを知らせるための割り込みは発生しません。

### 23.7.5 送信メッセージの停止

システムはそれぞれのメッセージバッファに関連した TXREQ ビットをクリアすることによりメッセージを停止することができます。ABAT ビット (CiCTRL<12>) を設定することにより、すべての送信待ちメッセージの停止を要求します (図 23-15 を参照してください)。待ち行列になったメッセージは TXREQ をクリアすることにより停止されます。待ち行列の停止は図 23-14 に示されています。メッセージがまだ送信を開始していない場合、もしくはメッセージが開始されたがアービトレーション権の喪失もしくはエラーにより割り込みが発生した場合は、停止処理がなされます。停止は、モジュールが TXABT ビット (CiTXnCON<6>) をセットした時に表示され、TxnIF フラグはセットされません。

メッセージが送信を開始した場合、現状のメッセージをすべて送信しようとします (図 23-16 を参照してください)。現メッセージがすべて送信された場合でアービトレーション権の喪失もしくはエラーが無い場合、TXABT はセットされません、なぜならメッセージが成功裡に送信されたからです。同様に、停止要求中にメッセージが送信された場合で、メッセージがアービトレーション権を喪失 (図 23-17 を参照してください) もしくはエラーが発生した場合、メッセージは再送信されず、TXABT ビットがセットされ、メッセージが成功裡に停止されたことを示します。

図 23-14: 待ち行列メッセージの停止

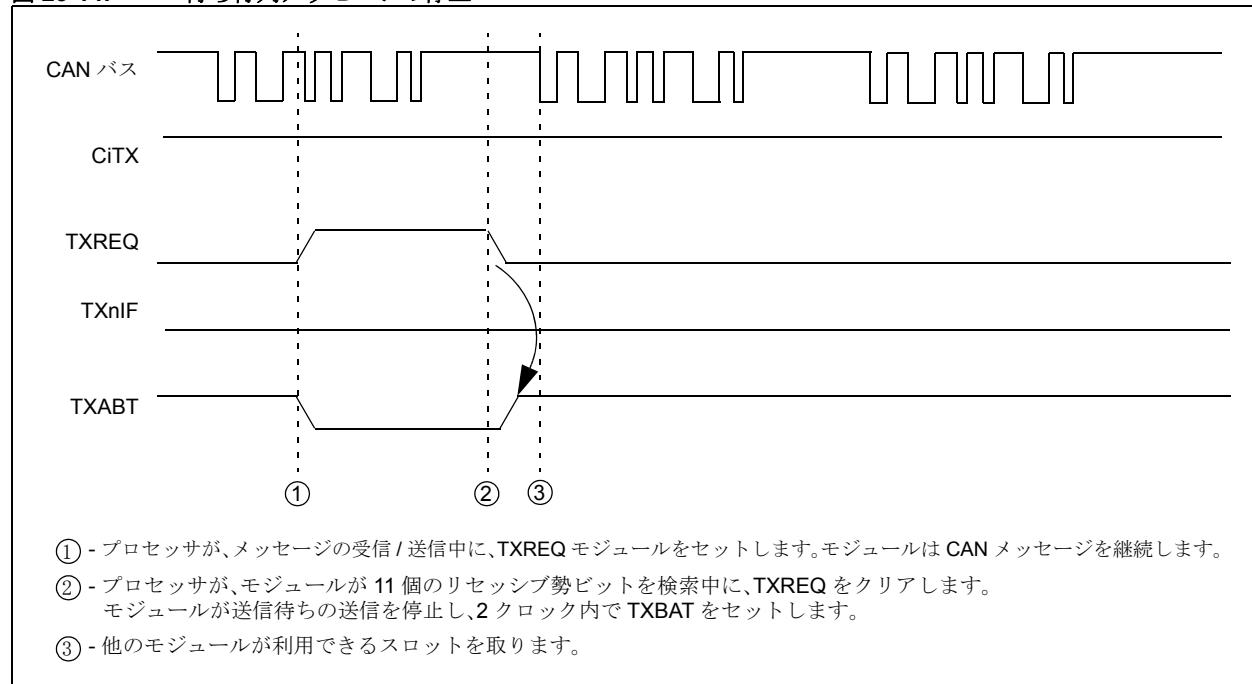


図 23-15: すべてのメッセージの停止

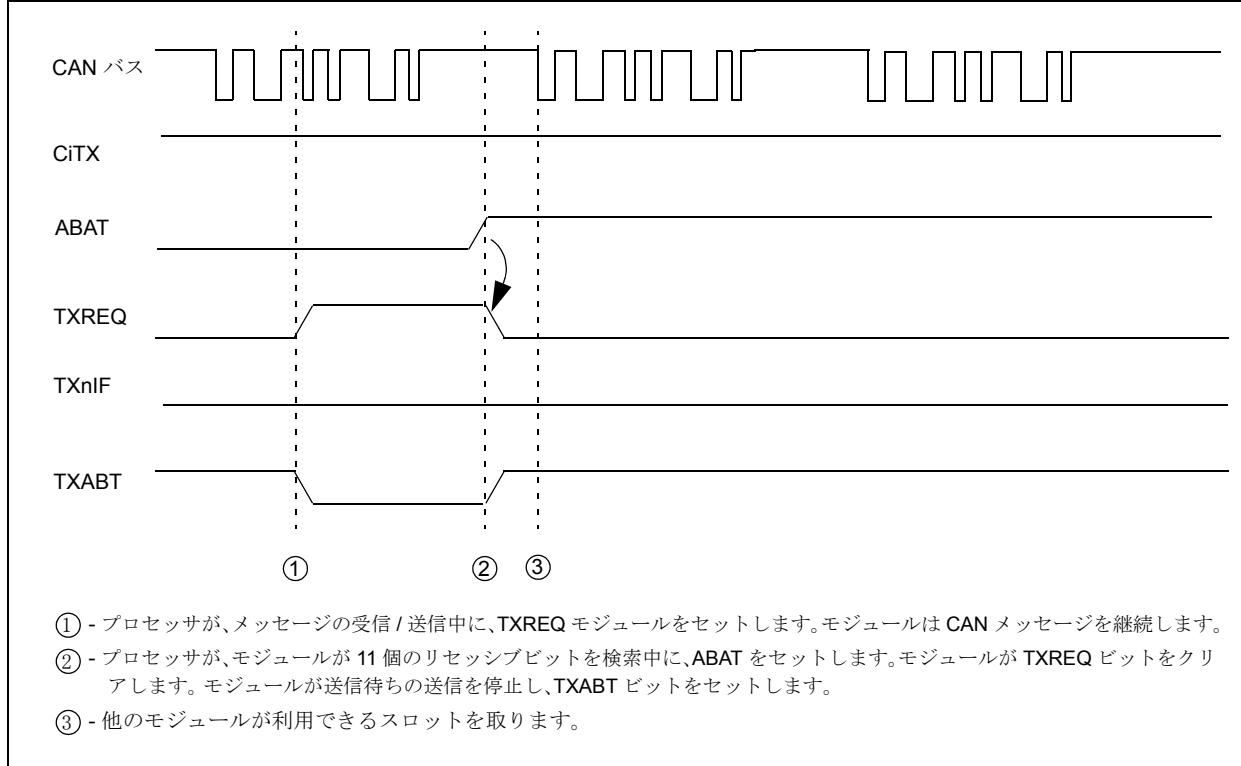


図 23-16: 送信中の停止の失敗例

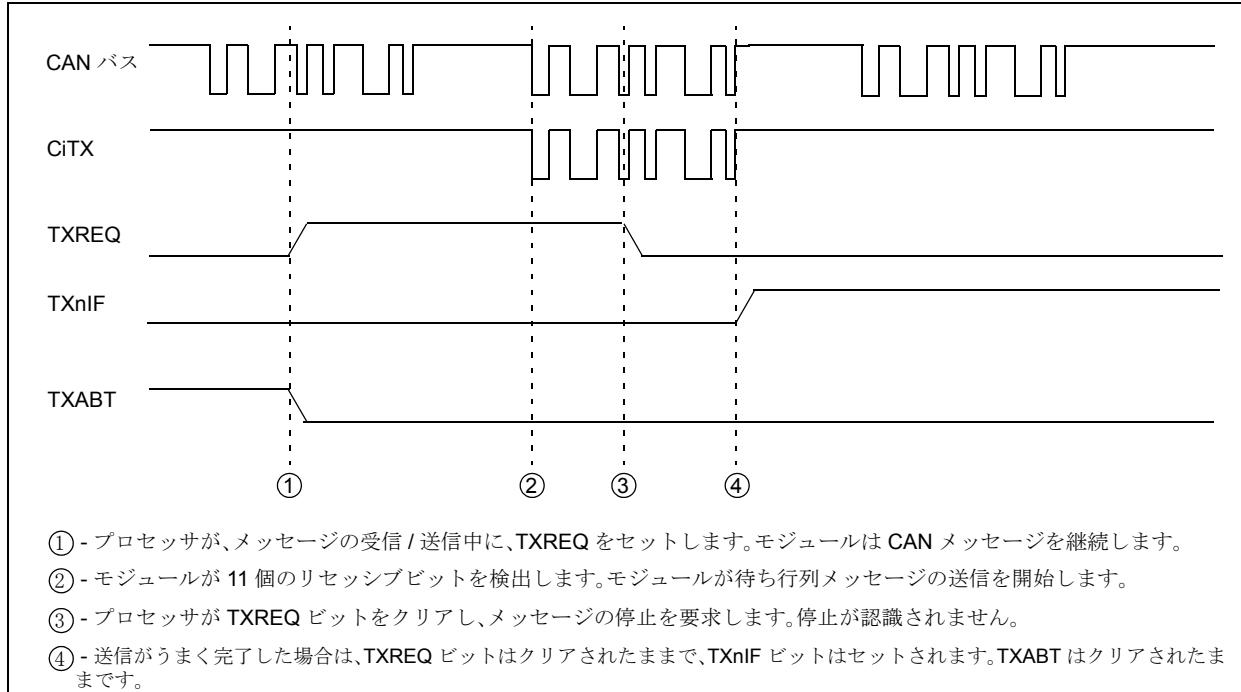


図 23-17: 送信中のアービトレーション権の喪失

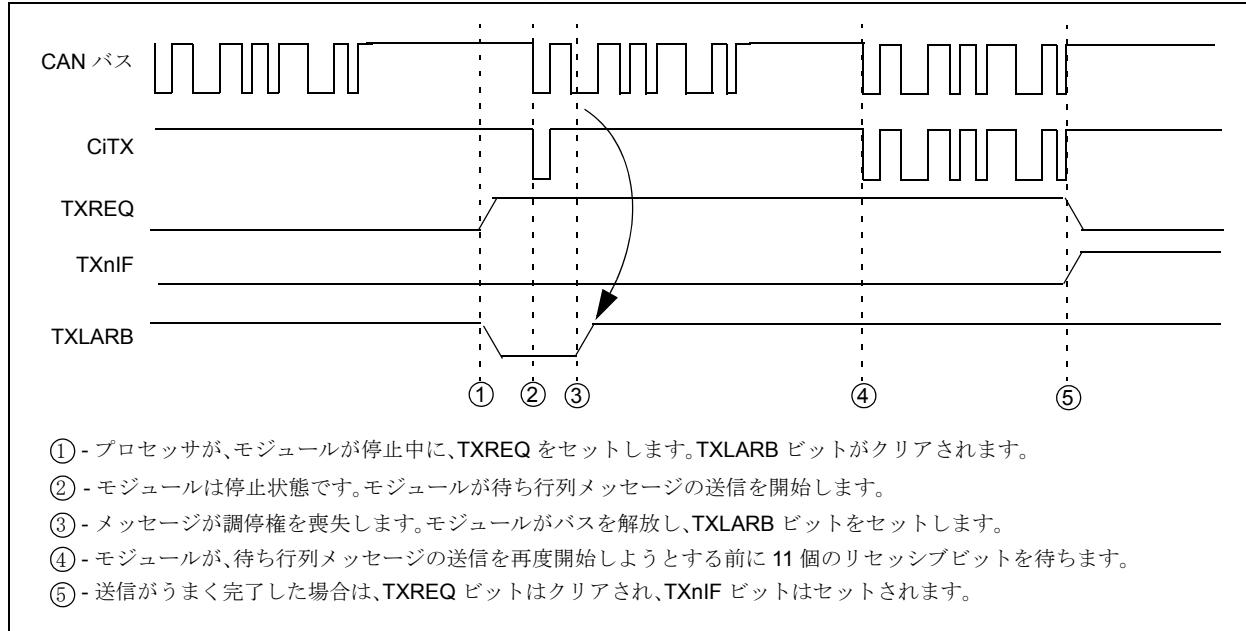
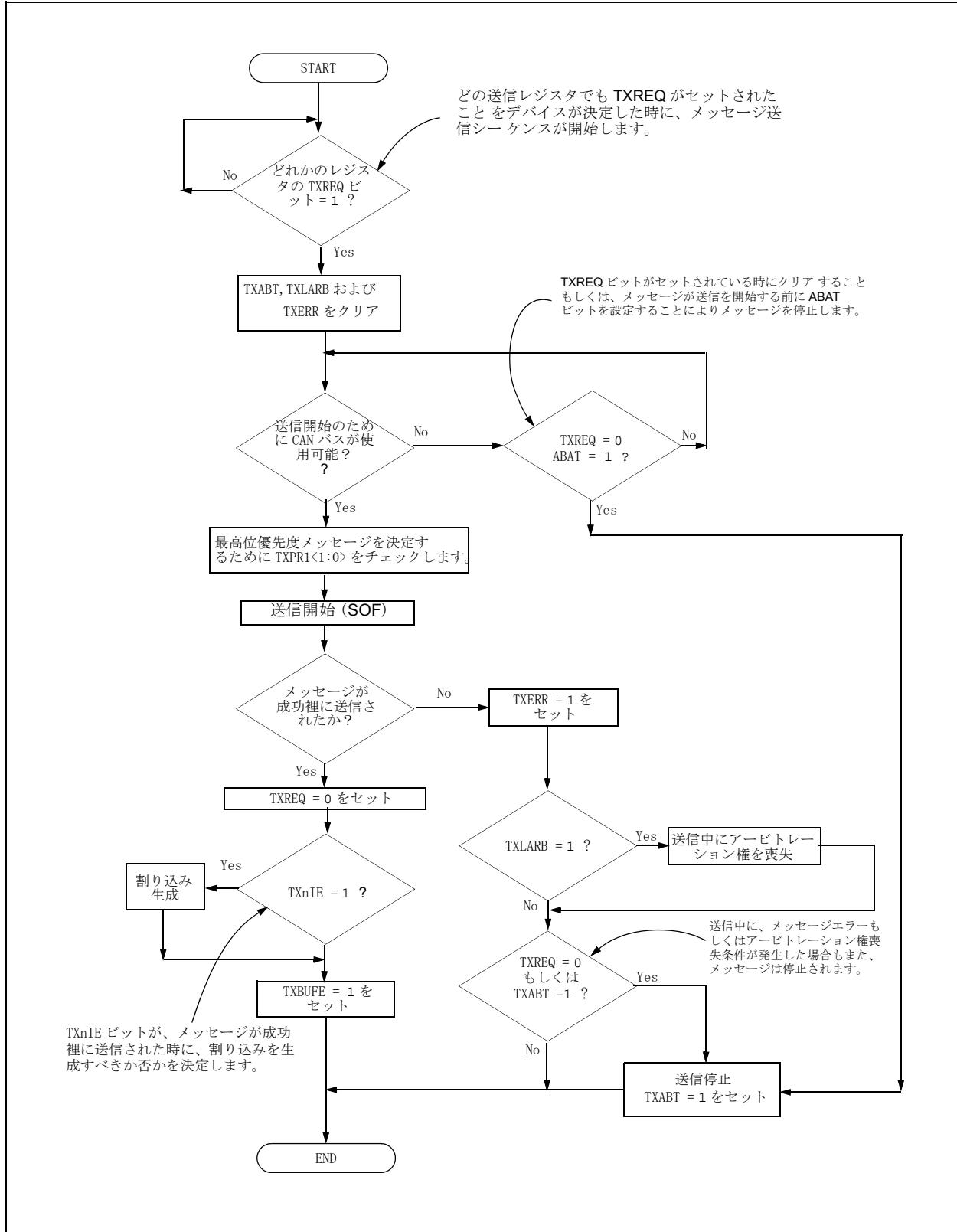


図 23-18: 送信フローチャート



### 23.7.6 送信境界条件

モジュールは、CAN バスメッセージのフレーミング時間に必ずしも同期しない送信コマンドを扱います。

#### 23.7.6.1 メッセージが開始したときに TXREQ をクリアする

TXREQ ビットは、メッセージを停止する目的で、メッセージがまさに送信を開始した時にクリアされます。メッセージが送信されていない場合は、TXABT ビットがセットされ、停止が成功裡に処理されたことを示します。

ユーザーが TXREQ ビットをクリアし TXABT ビットが 2 サイクル後にセットされない時は、メッセージはすでに送信を開始しています。

メッセージが送信中である場合、停止は直ぐには処理されず、ある時点以降に TXnIF 割り込みフラグもしくは TXABT ビットがセットされます。送信が始まっていると、メッセージは、エラーもしくはアビトレーション権の喪失が発生した場合のみ停止されます。

#### 23.7.6.2 メッセージが開始したときに TXABT をセットする

ABAT ビットをセットするとすべての送信待ちの送信バッファを停止し、すべてのバッファの TXREQ ビットをクリアします。境界条件は、TXREQ ビットをクリアする場合と同じです。

#### 23.7.6.3 メッセージが完了したときに TXREQ ビットをクリアする

TXREQ ビットは、メッセージが送信を成功裡に完了しそうなその時にクリアされます。TXREQ ビットが、メッセージが成功裡に送信を完了することによりクリアされるほんの少し前に、データバスによりクリアされても、送信がうまくいったことにより TXnIF フラグはセットされます。

#### 23.7.6.4 メッセージが完了したときに TXABT ビットをセットする

境界条件は、TXREQ ビットをクリアする場合と同じです。

#### 23.7.6.5 メッセージが送信を喪失したときに、TXREQ ビットをクリアする

TXREQ ビットは、メッセージがまさにアビトレーション権を喪失しそうになった時もしくはエラーが発生しそうになった時にクリアされます。

TXREQ 信号が、アビトレーション権信号の喪失もしくはエラー信号発生の前に立下った場合、送信中に TXREQ がクリアされた場合と同じ結果になります。アビトレーション権が喪失された場合もしくはエラーがセットされた場合、送信中にエラーが発生し、TXREQ ビットがセットされないことが観測されるので、TXABT ビットはセットされます。

アビトレーション権信号がブロックに入った後で TXREQ ビットが立ち下がる場合、送信中でないときに TXREQ がクリアされた場合と同じ結果になります。TXABT ビットがセットされます。

#### 23.7.6.6 メッセージが送信を喪失したときに TXABT ビットをセットする

境界条件は、TXREQ ビットをクリアする場合と同じです。

### 23.7.7 リセットの影響

リセットが発行されると、CAN モジュールは初期化されねばなりません。すべてのレジスタはリセット値に従ってセットされます。送信されたメッセージの内容は失われます。初期化は セクション 23.5.5「コンフィギュレーションモード」で説明されます。

## 23.7.8 送信エラー

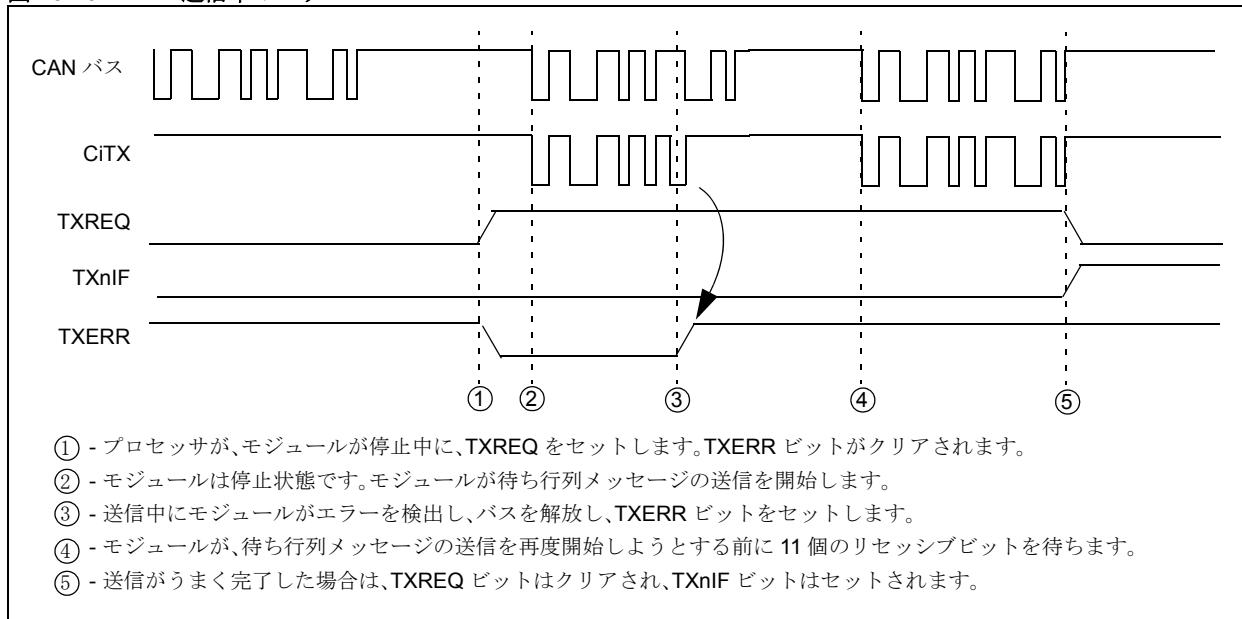
CAN モジュールは以下の送信エラーを検出します。

- 通知エラー
- 形式エラー
- ビットエラー

これらの送信エラーは必ずしも割り込みを生成しませんが、送信エラーカウンタにより表示されます。しかし、これらのエラー1つ1つは送信エラーカウンタを1づつ増加します。一旦エラーカウンタの値が96を越えたら、ERRIF (CiINTF<5>) および TXWAR ビット (CiINTF<10>) がセットされます。一旦エラーカウンタの値が96を越えたら、割り込みが生成され、エラーフラグレジスタ内の TXWAR ビットがセットされます。

送信エラーの例は図 23-19に図示されます。

図 23-19: 送信中のエラー



### 23.7.8.1 アクノレッジエラー

メッセージのアクノレッジフィールド内で、アクノレッジスロット（リセッショブピットとして送出されます）がドミナント優勢ビットを含むかどうかを送信器がチェックします。含まない場合、どのノードもフレームを正しく受信していません。通知エラーが発生し、メッセージは繰り返さなければなりません。エラーフレームは生成されません。

### 23.7.8.2 形式エラー

送信器が、フレームの終了 (EOF)、インターフレーム空間、アクノレッジデリミタもしくは CRC デリミタを含む4つのセグメントのうちの1つでドミナントビットが検出された場合、形式エラーが発生し、エラーフレームが生成されます。メッセージは繰り返されます。

### 23.7.8.3 ビットエラー

送信器がドミナントビットを送信し、リセッショブ劣勢ビットを検出した場合、ビットエラーが発生します。送信器がリセッショブビットを送信し、アービトレーションフィールドとアクノレッジスロットの期間中にドミナントビットを検出した場合は、正常なアービトレーションが行われているのでビットエラーは生成されません。

### 23.7.8.4 送信エラーカウンタを変更する規則

送信エラーカウンタは、以下の規則に従い変更されます。

- 送信器がエラーフラグを送信した場合、送信エラーカウンタは、以下の例外を除き 8 つづつ増加します。この 2 つの例外の場合は、送信エラーカウンタは変化しません。
  - 送信器が“エラーパッシブ”であり、“ドミナント”ACK を検出しないためにアクノレッジエラーを検出した場合、および受動エラーフラグを送信中に“ドミナント”ビットを検出しない場合。
  - RTR ビットの前にスタッフビットが位置するようなアビトレーション期間中にビットスタッフエラーが発生したことにより、送信器がエラーフラグを送信する場合、および“リッセシップ”であったはずの場合、および“リセッシップ”として送信されたが“ドミナント”としてモニターされた場合。
- 送信器は、アクティブエラーフラグもしくはオーバーロードフラグを送信中にビットエラーを検出した場合、送信エラーカウンタは 8 づつ増加します。
- どのノードも、アクティブエラーフラグ、パッシブエラーフラグもしくはオーバーロードフラグを送信した後、7 つまでの連続する“ドミナント”ビットを許容します。(アクティブエラーフラグもしくはオーバーロードフラグの場合)14 番目の連続“ドミナント”ビットを検出した後、もしくはパッシブエラーフラグに引き続く 8 番目の連続する“ドミナント”ビットを検出した後、および 8 つの追加の連続する“ドミナント”ビットのシーケンス後、すべての送信器は送信エラーカウンタを増加し、すべての受信器は受信エラーカウンタを 8 づつ増加します。
- メッセージが成功裡に送信した後(通知を受信し、フレーム終わりが完了するまでエラーがない場合)、送信エラーカウンタは、すでに 0 でない場合以外は 1 づつ減少します。

### 23.7.9 送信割り込み

メッセージの送信に関連していくつかの割り込みがあります。送信割り込みは 2 つのグループに分けられます。

- 送信割り込み
- 送信エラー割り込み

#### 23.7.9.1 送信割り込み

少なくとも 3 つの送信バッファのうちの 1 つは空(予定されない)であり、メッセージの送信を予定するためにデータが転送されます。CINTF レジスタ内の TXnIF フラグを読み込むことにより、どの送信バッファが利用でき、割り込みを発生させたかを知ることができます。

## 23.7.9.2 送信エラー割り込み

送信エラー割り込みは **ERRIF** フラグにより表示されます。このフラグはエラー状態が発生したことと示します。エラーのソースは CAN 割り込みステータスレジスタ **CiINTF** 内のエラーフラグをチェックすることで決定されます。このレジスタ内のフラグは受信 / 送信エラーに関連したものです。

**TXWAR** ビット (**CiINTF<10>**) は送信エラーカウンタが CPU 警告リミット 96 に達したことを示します。このビットが ‘0’ から ‘1’ へ遷移すると、エラー割り込みフラグがセットされます。**TXWAR** ビットは、送信エラーカウンタが CPU 警告リミット 96 に達したことを表示しつづける必要があるので、手動ではクリアされません。**TXWAR** ビットは、送信エラーカウンタが 95 以下になったら自動的にクリアされます。**ERRIF** フラグは手動でクリアされ、**TXWAR** ビットに影響を与えることなく割り込みサービスルーチンが起動されるようにします。

**TXEP** ビット (**CiINTF<12>**) は送信エラーカウンタがエラーパッシブリミット 127 を越えたことを示し、モジュールがエラーパッシブ状態になります。このビットが ‘0’ から ‘1’ へ遷移すると、エラー割り込みフラグがセットされます。**TXEP** ビットは、バスがエラーパッシブ状態であることを表示しつづける必要があるので、手動ではクリアされません。**TXEP** ビットは、送信エラーカウンタが 127 以下になったら自動的にクリアされます。**ERRIF** フラグは手動でクリアされ、**TXEP** ビットに影響を与えることなく割り込みサービスルーチンが起動されるようにします。

**TXBO** ビット (**CiINTF<13>**) は送信エラーカウンタが 255 を越えたことを示し、モジュールがバスオフ状態になります。このビットが ‘0’ から ‘1’ へ遷移すると、エラー割り込みフラグがセットされます。**TXBO** ビットは、バスオフ状態であることを表示しつづける必要があるので、手動ではクリアされません。**ERRIF** フラグは手動でクリアされ、**TXBO** ビットに影響を与えることなく割り込みサービスルーチンが起動されるようにします。

## 23.8 エラー検出

CAN プロトコルは賢いエラー検出メカニズムを持っています。以下のエラーが検出されます。これらのエラーは受信もしくは送信エラーです。

受信エラーは以下の通りです。

- 巡回型冗長チェック (CRC) エラー (セクション 23.6.5.1 「巡回冗長チェック (CRC) エラー」を参照してください)
- ビットスタッフィングエラー (セクション 23.6.5.2 「ビットスタッフィングエラー」を参照してください)
- 無効メッセージ受信エラー (セクション 23.6.5.3 「無効メッセージ受信エラー」を参照してください)

送信エラーは以下の通りです。

- アクノレッジエラー (セクション 23.7.8.1 「アクノレッジエラー」を参照してください)
- 形式エラー (セクション 23.7.8.2 「形式エラー」を参照してください)
- ビットエラー (セクション 23.7.8.3 「ビットエラー」を参照してください)

## 23.8.1 エラー状態

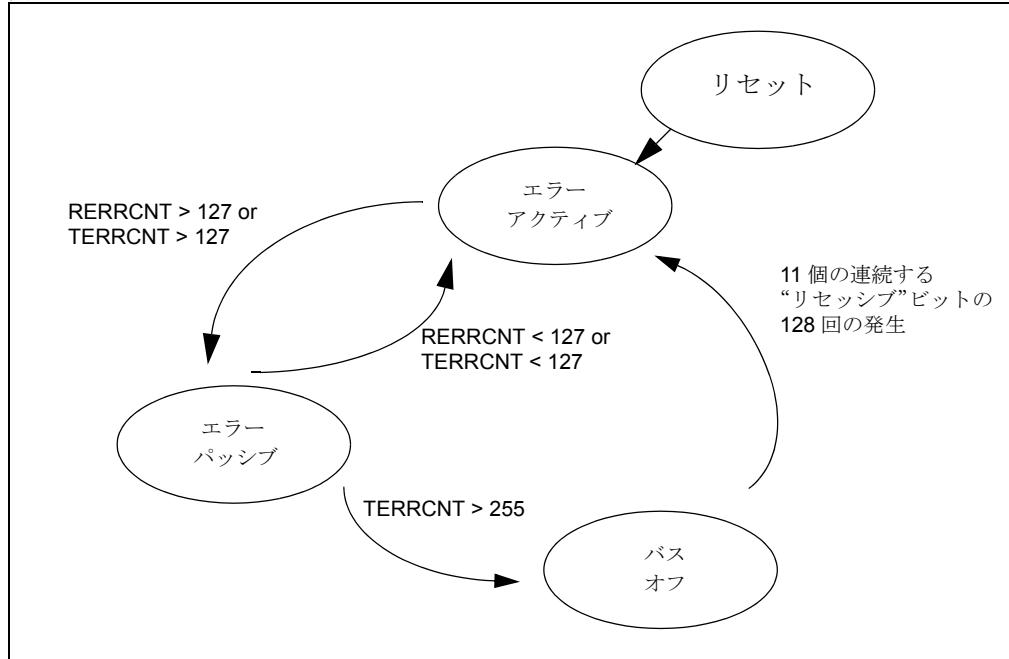
検出されたエラーは、エラーフレームによりすべての他のノードに公開されます。エラーのあるメッセージを送信することは停止され、フレームはできるだけ早く繰り返されます。さらに、おののの CAN ノードは、内蔵エラーカウンタの値によって 3 つのエラー状態 “エラーアクティブ” “エラーパッシブ” もしくは “バスオフ” のうちの 1 つになります。エラーアクティブ状態は、バスノードがメッセージとアクティブエラーフレーム (ドミナントビットにより構成されます) を何ら制約無しに送信する通常の状態です。エラーパッシブ状態では、メッセージとパッシブエラーフレーム (リセシシブビットにより構成されます) が送信されることがあります。バスオフ状態では、ステーションは一時的にバス通信に参加できません。この状態ではメッセージは受信も送信もされません。

### 23.8.2 エラーモードとエラーカウンタ

CAN コントローラは 2 つのエラーカウンタ、受信エラーカウンタ (RERRCNT) と送信エラーカウンタ (TERRCNT) を持っています。両方のカウンタの値は、エラーカウントレジスタ CiEC から CPU により読み込み可能です。これらのカウンタは、CAN バス仕様に従って増加したり減少したりします。

CAN コントローラは、両方のエラーカウンタがエラーパッシブリミット 128 より少ない場合にエラーアクティブになります。少なくとも 1 つのエラーカウンタが 128 以上になった場合、CAN コントローラはエラーパッシブになります。送信エラーカウンタがバスオフリミット 256 以上になった場合、CAN コントローラはバスオフになります。デバイスは、バスオフ回復シーケンス (128 個の連続する 11 のリセッショビット時間) が終了するまで、この状態を保ちます。さらに、エラー状態警告フラグビット EWARN (CiINTF<8>) があり、これは少なくともエラーカウンタがエラー警告リミット 96 以上になった場合にセットされます。EWARN は、両方のエラーカウンタがエラー警告リミットより小さくなったら場合にリセットされます。

図 23-20: エラーモード



### 23.8.3 エラーフラグレジスタ

エラーフラグレジスタ内の値はどのエラーがエラー割り込みを発生させたかを表示します。RXnOVR エラーフラグ (CiINTF<15> および CiINTF<14>) は、このレジスタ内のその他のエラーフラグビットとは異なる機能を持っています。RXnOVR ビットは、ERRIF 割り込みフラグをクリアするために、クリアされる必要があります。このレジスタ内のその他のエラーフラグビットは、送信・受信エラーカウンタの値が特定の閾値を過ぎた場合に ERRIF 割り込みフラグをセットします。この場合、ERRIF 割り込みフラグをクリアすることにより、繰り返し同じ割り込みが発生しないようにすることができます。エラーカウンタの値が閾値の近辺で上がったり下がったりすることで割り込みが繰り返し発生することを止めるために、特定の割り込みが発生した後でその特定割り込みを無効にするほうが望ましいでしょう。

## 23.9 CAN ポーレート

どんな特定の CAN バス上であれすべてのノードは同じ公称ビットレートを持つ必要があります。CAN バスは(クロックを抽出しない)NRZ コーディングを使用します。従って、受信器側のクロックは独立で、受信側ノードにより再生し送信器のクロックに同期する必要があります。

ポーレートをセットするために、以下のビットを初期化する必要があります。

- 同期ジャンプ幅(セクション 23.9.6.2「再同期」を参照してください)
- ポーレートプリスケーラー(セクション 23.9.2「プリスケーラの設定」を参照してください)
- フェーズセグメント(セクション 23.9.4「フェーズセグメント」を参照してください)
- フェーズセグメント 2 の長さ限界(セクション 23.9.4「フェーズセグメント」を参照してください)
- サンプル点(セクション 23.9.5「サンプル点」を参照してください)
- 伝達セグメントビット(セクション 23.9.3「伝達セグメント」を参照してください)

### 23.9.1 ビットタイミング

発振器と送信器時間はノード毎に変動するので、受信器はある種の PLL を持ち、受信器のクロックを同期化、保持するためにデータ送信エッジと同期を取る必要があります。データは NRZ コードなので、エッジを少なくとも 6 ビットごとに発生させて、デジタルフェーズロックループ(DPLL) の同期を保持することを確実にするためにビットスタッフィングを行う必要があります。

ビットタイムフレーム内で実行されるバスタイミング機能、例えば、ローカル発振器への同期、ネットワーク送信遅延の補償、およびサンプル点の位置取り、については、プログラム可能な DPLL のビットタイミングロジックにより規定されます。

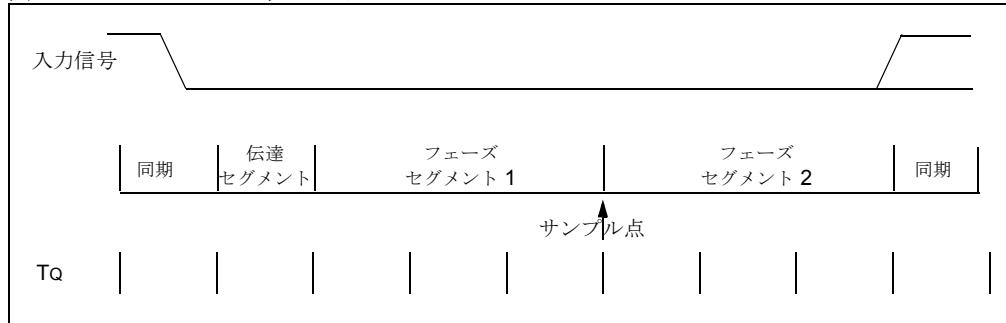
CAN バス上のすべてのコントローラは同じポーレートとビット長を持つ必要があります。しかし、異なるコントローラは同じマスター発信器クロックを持つ必要はありません。しかし、個別のコントローラの異なるクロック周波数でも、それぞれの時間量の数を調整することによりポーレートを合わせるように調整する必要があります。

公称ビットタイムは、重複しない時間セグメントに分割して考えられます。これらのセグメントを図 23-21 に示します。

- 同期セグメント(Sync Seg)
- 伝達時間セグメント(Prop Seg)
- フェーズバッファセグメント 1(Phase1 Seg)
- フェーズバッファセグメント 2(Phase2 Seg)

時間セグメントと公称ビット時間もまた、時間量もしくは  $T_Q$  と呼ばれる時間の整数倍で構成されます。定義では、公称ビット時間は最小 8  $T_Q$  で、最大は 25  $T_Q$  です。また、最小公称ビット時間は 1  $\mu\text{sec}$  で、このとき最大 1 MHz ビットレートです。

図 23-21: CAN ビットタイミング



### 23.9.2 プリスケーラの設定

クロック生成用の固定 2 分周器に加えて、1 から 64 までの範囲の分周比を持つプログラマブルプリスケーラがあります。時間量 (TQ) は、入力クロック周期 TCAN から計算される固定時間単位です。時間量は以下のように定義されます。

**式 23-1:** クロック生成用時間量

$$TQ = 2 \cdot (BRP + 1) \cdot TCAN$$

BRP は  $BRP < 5:0 >$  の 2 進値です。

TCAN は CANCKS ビットに依存し、TCY もしくは TCY/4 の値をとります。

**例 23-1:** ビットレート計算例

4FCY = 32 MHz, BRP<5:0> = 0x01 および CANCKS = 0 の場合、

$$TQ = 2 \cdot (BRP + 1) \cdot \frac{TCY}{4} = 2 \times 2 \times (1/32 \times 10^6) = 125\text{ns}$$

公称ビットタイム = 8 TQ の場合、  
公称ビットレートは  $1/(8 \times 125 \times 10^{-9})\text{Mbps}$  です。

**例 23-2:** ポーレートプリスケーラー計算例

CAN Baud Rate = 125 kHz  
FCY = 5 MHz, CANCKS = 1

1. ビット時間当たりの TQ クロック数を選択します。( 例えば K=16)
2. ポーレートから TQ を計算します。

$$TQ = \frac{1/(BaudRate)}{K} = \frac{1/125 \times 10^3}{16} = 500\text{ns}$$

3. BRP<5:0>: を計算します。

$$TQ = 2 \cdot (BRP + 1) \cdot TCAN$$

$$\begin{aligned} BRP &= (2TQ/TCY) \angle 1 \\ &= \frac{2(500 \times 10^{-9})}{1/(5 \times 10^6)} \angle 1 \\ &= 4 \end{aligned}$$

異なるノードにおける発振器の周波数は、システム全体で規定される時間量を与えるために、コードィネート（決定）される必要があります。これは、すべての発振器は、TQ の整数分の 1 である Tosc を持つ必要があることを意味します。

## 23.9.3 伝達セグメント

ビット時間のこの部分は、ネットワーク内の物理遅延時間を補正するために使用されます。これらの遅延時間は、バスライン上の信号伝達時間とノードの内部遅延時間からなります。遅延は、送信器から受信器までの往復として、バスライン上の信号の伝達時間、入力比較器の遅延および出力ドライバの遅延の 2 倍として計算されます。伝達セグメントは、PRSEG<2:0> ビット (CiCFG2<2:0>) を設定することで、1 TQ から 8 TQ までプログラムできます。

## 23.9.4 フェーズセグメント

フェーズセグメントは、送信されたビット時間以内で受信ビットのサンプリング位置を特定するため、オプション的に使用されます。サンプリング点はフェーズ 1 セグメントとフェーズ 2 セグメントの間にあります。これらのセグメントは、再同期により伸びたり縮んだりします。フェーズ 1 セグメントの終わりは、ビット周期内でのサンプリングポイントを決定します。セグメントは 1 TQ から 8 TQ の値がプログラムされます。フェーズ 2 セグメントは次に送信されるデータ遷移への遅延を与えます。このセグメントは 1 TQ から 8 TQ の値がプログラムされるかもしれません、フェーズ 1 セグメントもしくは情報処理時間 (3 TQ's) の大きい方と等しくなるように定義されます。フェーズセグメント 1 は、ビット SEG1PH<2:0> (CiCFG2<5:3>) を設定することにより初期化され、フェーズセグメント 2 は SEG2PH<2:0> (CiCFG2<10:8>) を設定することにより初期化されます。

## 23.9.5 サンプル点

サンプル点は、それぞれのビットの値として、バスレベルが読み込まれ解釈される時間点です。その位置はフェーズセグメント 1 の終わりです。ビットタイミングが遅くて多くの TQ を含んでいる場合、サンプル点でバスラインの複数サンプリングを規定することができます。この場合、CAN バスにより決定されたレベルは 3 値の多数決からの結果に相当します。多数サンプルはサンプル点で採取され、TQ/2 の距離を持って 2 つ前の点で採取されます。CAN モジュールでは、同じ点を 3 回サンプリングするかもしれません同じ点を 1 回サンプリングするかどちらかを選択できます。これは、SAM ビット (CiCFG2<6>) をセットするかもしれませんクリアすることにより実行されます。

## 23.9.6 同期化

異なるバスノードの発振周波数間の位相差を補正するために、CAN コントローラーは入力信号の適切な信号エッジに同期する必要があります。送信データ内のエッジが検出されると、論理は期待される時間（同期セグメント）のエッジの位置との比較を取ります。それから回路はフェーズ 1 セグメントとフェーズ 2 セグメントの値を調整します。同期を取るために使用される 2 つの機構があります。

### 23.9.6.1 ハード同期

バスがアイドル中に“リセッショブ”から“ドミナント”のエッジがあるとき、これはメッセージの開始を示しますが、その時はいつでも、ハード同期のみが行われます。同期化の後、ビットタイムカウンタは、同期セグメントとともに再スタートされます。ハード同期は、ハード同期を引き起こすエッジが、再スタートされたビット時間の同期化セグメント内にあるようになります。同期化のルールにより、ハード同期が取られた場合、そのビット時間内では、再同期は取られません。

## 23.9.6.2 再同期

再同期の結果、フェーズセグメント 1 が伸びたり、もしくはフェーズセグメント 2 が縮んだりするかもしれません。フェーズバッファセグメントの伸び縮みの量は、 $SJW <1:0>$  ビット (CiCFG1<7:6>) で規定され、再同期ジャンプ幅ビットの上限として与えられます。同期ジャンプ幅の値はフェーズセグメント 1 に追加されるかもしれませんフェーズセグメント 2 から引かれます。再同期ジャンプ幅は  $1 TQ$  から  $4 TQ$  の間の値をプログラムできます。

クロック情報は、リセッショナルドミナントバス状態への遷移からのみ引き出されます。連続するビットの固定最大数のみが同じ値を持つという性質により、フレーム期間中（例えばビットスタッフイング）にビットストリームにバスユニットを確実に再同期化させます。

エッジのフェーズエラーは、同期セグメントに対するエッジの位置により与えられ、時間量で計測されます。フェーズエラーは、以下のように  $TQ$  の大きさで定義されます。

- $e = 0$  エッジが同期セグメント以内にある場合
- $e > 0$  がサンプル点より前にある場合
- $e < 0$  エッジが、前ビットのサンプル点より後にある場合。

フェーズエラーが再同期ジャンプ幅のプログラムされた値以下の場合、再同期の影響はハード同期の影響と同じです。

フェーズエラーが再同期ジャンプ幅より大きい場合、およびフェーズエラーが正值の場合、フェーズセグメント 1 は、再同期ジャンプ幅と同じ量だけ長くなります。

フェーズエラーが再同期ジャンプ幅より大きい場合、およびフェーズエラーが負値の場合、フェーズセグメント 2 は、再同期ジャンプ幅と同じ量だけ短くなります。

図 23-22: ビット周期の延長化

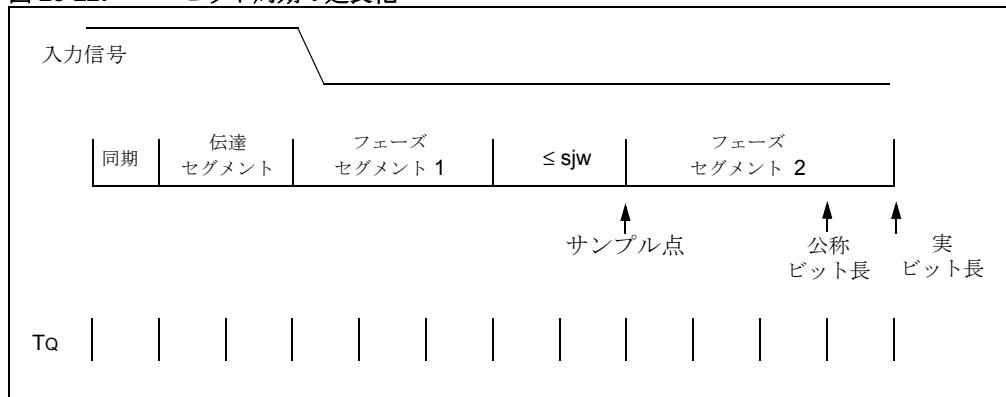
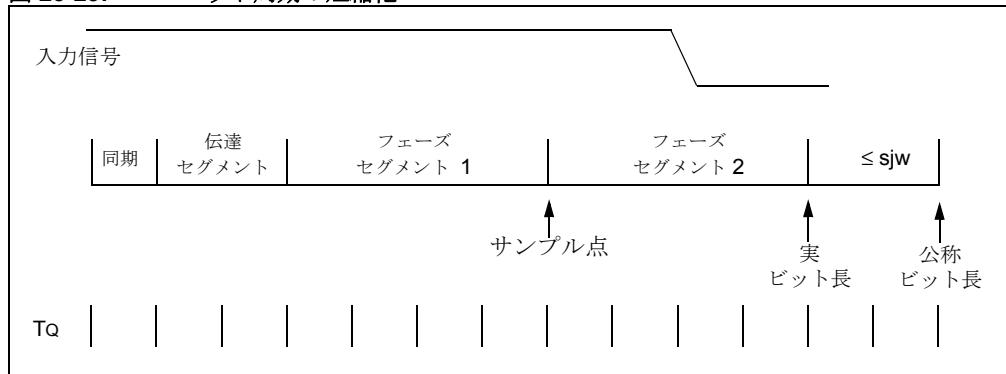


図 23-23: ビット周期の短縮化



## 23.9.7 時間セグメントのプログラミング

時間セグメントのプログラミングには、以下のようにいくつかの要求事項があります。

- ・伝達セグメント+フェーズ1セグメント>=フェーズ2セグメント
- ・フェーズ2セグメント>同期ジャンプ幅

典型的には、ビットのサンプリングは、システムのパラメータにも依存しますが、ビット時間の 60-70% の時点とすべきです

例 23-2 では、ビット時間は 16 TQ です。同期セグメント = 1 TQ かつ伝達セグメント = 2 TQ を選択した場合、フェーズセグメント 1 = 7 TQ を設定することにより、初期遷移後、10 TQ (ビット時間の 62%) の時点でサンプリング点が置かれます。これにより、フェーズセグメント 2 には 6 TQ が残されることになります。

規則によれば、フェーズセグメント 2 は 6 なので、**SJWS<1:0>** ビットは最大 4 TQ にセット可能です。しかし、通常は、セラミック共振器を使う場合等、異なるノードのクロック生成が低精度もしくは低安定性の場合にだけ、大きな同期ジャンプ幅が必要です。従って、同期ジャンプ幅は、典型的には 1 で十分です。

## 23.10 割り込み

モジュールはいくつかの割り込みソースを持ちます。これらの割り込みの 1 つ 1 つは、別々に有効にしたり無効にしたりすることができます。**CiINTF** レジスタは割り込みフラグを持ちます。**CiINTE** レジスタは 8 つの主な割り込みの有効化を制御します。**CiCTRL** レジスタ (**ICODE<2:0>**) 内の、特殊な読み込み専用ビットの組合せは、割り込みの効率的な操作のために、ジャンプテーブルと一緒に用いられます。

すべての割り込みは 1 つのソースを持ちますが、エラー割り込みは例外です。どのエラー割り込みソースも、エラー割り込みフラグをセットすることができます。エラー割り込みのソースは、**CiINTF** レジスタを読み込むことにより特定できます。

割り込みは 2 つのカテゴリー：受信と送信割り込みに区分できます。

受信に関連する割り込みは以下の通りです。

- ・受信割り込み
- ・ウェイクアップ割り込み
- ・受信器オーバーラン割り込み
- ・受信器ウォーニング割り込み
- ・受信器エラーパッシブ割り込み

送信に関連する割り込みは以下の通りです。

- ・送信割り込み
- ・送信器ウォーニング割り込み
- ・送信器エラーパッシブ割り込み
- ・バスオフ割り込み

### 23.10.1 割り込み通知

割り込みは、**CiINTF** レジスタ内の 1 つもしくはそれ以上のステータスフラグと直接関係します。割り込みは、対応するフラグの 1 つがセットされるまで待たされています。レジスタ内のフラグは、割り込みとのハンドシェークを取るため、割り込みハンドラ内でリセットされる必要があります。それぞれの条件が継続して有効である場合はフラグはクリアされませんが、エラーカウンタレジスタの 1 つ内のある値に達したことにより引き起こされる割り込みは例外です。

### 23.10.2 ICODE ビット

ICODE<2:0> ビット (CiCTRL<3:1>) は、ジャンプテーブル経由で割り込みを効率的に取り扱うように設計された読み込み専用ビットの組合せです。ICODE<2:0> ビットは一度に 1 つの割り込みのみを表示できます。なぜなら割り込みビットはこのレジスタに多重化されているからです。従って、最高位の優先度を持つ待機中、有効になった割り込みが、ICODE<2:0> ビットに反映されます。一旦最高位の優先度を持った割り込みフラグがクリアされると、次の優先度を持った割り込みコードが ICODE<2:0> ビットに反映されます。対応する割り込みの割り込みコードは、割り込みフラグと割り込み有効ビットの両方がセットされた場合にのみ表示されます。表 23-6 に ICODE<2:0> ビットの動作について説明します。

表 23-6: ICODE ビット復号表

ICODE<2:0>	プール表現
000	$\overline{\text{ERR}} \cdot \overline{\text{WAK}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}} \cdot \overline{\text{TX2}} \cdot \overline{\text{RX0}} \cdot \overline{\text{RX1}}$
001	ERR
100	$\overline{\text{ERR}} \cdot \overline{\text{TX0}}$
011	$\overline{\text{ERR}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}}$
010	$\overline{\text{ERR}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}} \cdot \overline{\text{TX2}}$
110	$\overline{\text{ERR}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}} \cdot \overline{\text{TX2}} \cdot \overline{\text{RX0}}$
101	$\overline{\text{ERR}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}} \cdot \overline{\text{TX2}} \cdot \overline{\text{RX0}} \cdot \overline{\text{RX1}}$
111	$\overline{\text{ERR}} \cdot \overline{\text{TX0}} \cdot \overline{\text{TX1}} \cdot \overline{\text{TX2}} \cdot \overline{\text{RX0}} \cdot \overline{\text{RX1}} \cdot \text{WAK}$

凡例:     $\text{ERR} = \text{ERRIF} \cdot \text{ERRIE}$

$\text{TX0} = \text{TX0IF} \cdot \text{TX0IE}$

$\text{TX1} = \text{TX1IF} \cdot \text{TX1IE}$

$\text{TX2} = \text{TX2IF} \cdot \text{TX2IE}$

$\text{RX0} = \text{RX0IF} \cdot \text{RX0IE}$

$\text{RX1} = \text{RX1IF} \cdot \text{RX1IE}$

$\text{WAK} = \text{WAKIF} \cdot \text{WAKIE}$

### 23.11 タイムスタンプ

CAN モジュールは、有効フレームが受け付けられたらいつでも、タイマキャプチャ入力に送信される信号を生成します。CAN 仕様では、EOF フィールドが成功裡に送信される前にエラーが発生しない場合は、フレームは有効であると規定するので、EOF の直後にタイマ信号が生成されます。1 ビット時間分のパルスが生成されます。

タイムサンプリングは、TSTAMP 制御ビット (CiCTRL<15>) により有効になります。タイマサンプリングに用いられるキャプチャ入力はデバイスにより変わります。詳しくは特定デバイスのデータシートを参照してください。

### 23.12 CAN モジュール I/O

CAN バスモジュールは最大 2 つの I/O ピンと通信します。1 つの送信ピンと 1 つの受信ピンです。これらのピンは、デバイスの通常のデジタル I/O 機能と多重化されています。

モジュールがコンフィギュレーションモード、モジュール無効モードもしくはループバックモードの時、I/O ピンはポート I/O 機能に復帰します。

モジュールがアクティブの時、CiTX ピン ( $i = 1$  もしくは  $2$ ) は常に CAN 出力機能専用になります。送信ピンに関連した TRIS ビットは CAN バスモードにより無効にされます。モジュールは、CiRX 入力ピンの CAN 入力を受信します。

## 23.13 CPU パワー節約モードでの動作

### 23.13.1 SLEEP モード時の動作

SLEEP モードへは、PWRSAV #0 命令を実行することにより入ります。これにより水晶発振子が停止し、すべてのシステムクロックが停止します。CPU が SLEEP モードに入る時、ユーザーは、モジュールがアクティブでないことを確実にする必要があります。TRIS レジスタ内の値に依存して、ピンは通常 I/O 機能に戻ります。

CAN バスは分割動作できないので、モジュールが動作モード中には PWRSAV #0 命令を実行してはいけません。モジュールはまず、REQOP<2:0> = 001 (CiCTRL<10:8>) を設定することにより無効モードに切替える必要があります。OPMODE<2:0> = 001 (CiCTRL<7:5>) の時、これは無効モードが達成されたことを示しますが、そうなれば SLEEP モード命令が使用できます。

図 23-24 に、CPU が SLEEP モードに入った時に CAN モジュールがどのように動作するか、バスの動作によってどのようにモジュールがウェイクアップするかを示します。CAN バスの動作により CPU が SLEEP モードから抜け出た時、WAKIF フラグ (CiINTF<6>) がセットされます。

モジュールは、デバイスが SLEEP モードの間、CiRX ラインの動作をモニターします。

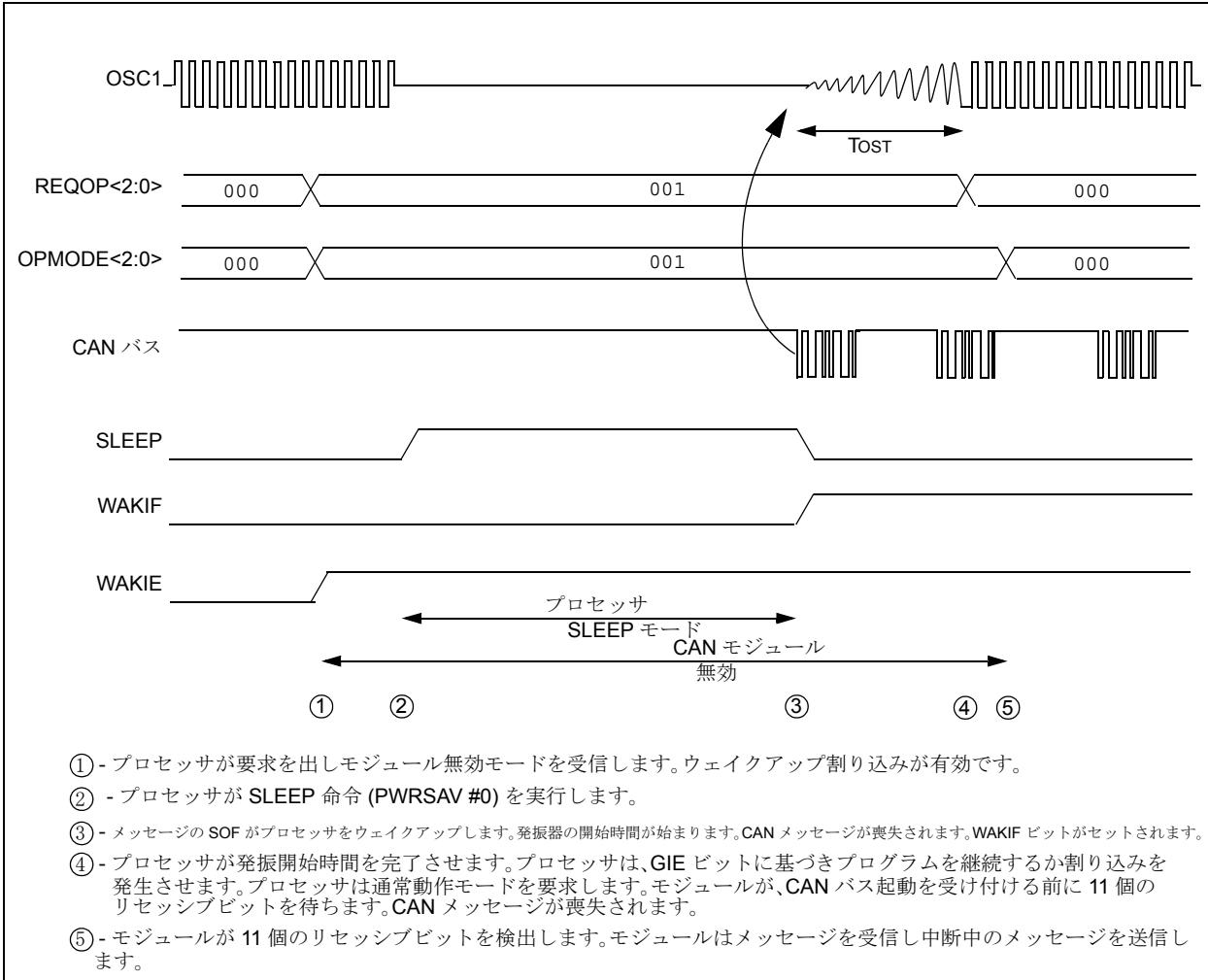
デバイスが SLEEP モードで WAKIE ウェイクアップ割り込み有効がセットされている場合、モジュールは割り込みを発生させ、CPU をウェイクアップします。発振器と CPU の立上げの遅延により、ウェイクアップを引き起こすメッセージ動作は失われます。

モジュールが CPU SLEEP モードで WAKIE がセットされていない場合、割り込みは発生せず、CPU と CAN はスリープを継続します。

CAN モジュールが無効モードの場合、モジュールはウェイクアップし、WAKIE ビットの状態に依存して、割り込みを発生します。モジュールは、SLEEP モードからウェイクアップさせるメッセージを正しく受信することが期待されます。

モジュールもしくは CPU が SLEEP モードの間は CiRX 入力ラインヘロウパスフィルタ機能を適用するように、モジュールはプログラムされます。この特徴は、CAN バスライン上の短いグリッヂによるウェイクアップからモジュールを保護するために使用されます。そのようなグリッヂは、ノイズの多い環境内での電磁影響の結果で発生しえます。WAKFIL ビット (CiCFG2<14>) はフィルタを有効化もしくは無効化します。

図 23-24: プロセッサ SLEEP と CAN バスのウェイクアップ割り込み



### 23.13.2 CPU IDLE モード期間中の CAN モジュール動作

CPU アイドル命令 (PWRSAV #1) を実行時、CAN モジュールの動作は、CSIDL ビット (CiC-TRL<13>) の状態により決まります。

CSIDL = 0 の場合、モジュールは、IDLE モードのアサート動作を継続します。CAN モジュール割り込みが有効になった場合、CAN モジュールはデバイスを IDLE モードからウェイクアップさせることができます。

CSIDL = 1 の場合、モジュールは IDLE モードの動作を停止します。同じ規則と条件が、SLEEP モードへ入ったり、SLEEP モードからウェイクアップするために適用されます。詳しくは、セクション 23.13.1 「SLEEP モード時の動作」を参照してください。

## 23.14 CAN プロトコルの概要

制御エリアネットワーク (CAN) は、大変高いレベルの強さを持った分散型実時間制御を効率的にサポートできる、シリアル通信プロトコルです。CAN プロトコルは、Robert Bosch GmbH により、1991 年から CAN 仕様書 V2.0B すべて規定されています。

その応用の領域は高速ネットワークから低価格の多重ワイヤリングに及びます。自動車エレクトロニクス部品 (すなわち、エンジン制御ユニット、センサー、横滑り対策システム等等) は、最大 1 Mbit/sec のビットレートを持った CAN で繋がれています。CAN ネットワークにより、自動車内のワイヤーハーネスをコスト面で効果的に置き換えることができます。ノイズの多い環境内でのバスの強さと、エラー状態の検出とそれからの回復の能力により、デバイスネット、SDS およびその他のフィールドバスプロトコルといった産業プロトコル用に適しています。

CAN は非同期シリアルバスシステムで、1 つの論理バスラインを持っていました。それは、等しいバスノードを持った、オープンで線形なバス構造を持っています。CAN バスは 2 つもしくはそれ以上のノードから構成されます。バス上のノード数は、他のノードの通信を乱すことなく動的に変更できます。これにより、バスノードの接続・切断が容易にできます。( 例えれば、システム機能の追加、エラー回復もしくはバスモニターが容易です )。

バス論理は “ワイヤー AND” 機構に対応し、“リセッショビット” (ほとんどの場合はそうですが、必ずしも論理レベル ‘1’ に相当するわけではありません) は、“ドミナント” ビット (ほとんどの場合論理レベル ‘0’ ) により上書きされます。バスのノードがどれもドミナントビットを送信していない場合、バスラインはリセッショビット状態にありますが、どのバスノードからでもドミナントビットはドミナントバス状態を生成します。従って、CAN バスラインにとって、2 つのあり得るビット状態 (ドミナントおよびリセッショビット) を送信することができる媒体が選択される必要があります。もっと一般的で安価な方法の 1 つは、撲り線ペアを使用することです。バス線は “CANH” および “CANL” と呼ばれ、ノードに直接もしくはコネクタを経由して接続されます。コネクタの要求事項に関しては CAN によって規定される標準規格はありません。撲り線ペアは、バス線のそれぞれの端で終端抵抗によって終端処理されます。最大バススピードは 1 Mbit で、最大 40 メータのバスの長さでも達成されます。40 メータを越える長さについては、バススピードは押さえる必要があります (40K ビットのバススピードであれば 1000 メートルのバスが実現できます)。1000 メートル以上の長さのバスについては、特別なドライバが必要です。追加装置無しに、少なくとも 20 のノードが接続可能です。送信の差動の性質により、CAN はもともと電磁エネルギーの輻射に対して敏感ではありません、なぜなら両方のバスラインは同様に影響を受けるため、差動信号は影響を受けないからです。バスラインもまた、特に高いボーレートにおいては、バス自体からの電磁放出輻射を減らすために遮蔽されます。

バイナリデータは NRZ コード (Non-Return-to-Zero、ロウレベル=ドミナント状態、ハイレベル=リセッショビット状態) に対応して符号化されます。すべてのバスノードのクロック同期を確実にするために、ビットスタッフィングが使用されます。これによりメッセージの送信期間中に最大 5 つの連続ビットが同じ極性を持ちます。同じ極性を持つ 5 つの連続ビットが送信された時はいつでも、送信器は、その後のビットを送信する前にビットストリームに 1 つの逆極性をもつ追加ビットを挿入します。受信器もまた、同じ極性を持つビット数をチェックし、ビットストリームからスタッフィングを取り除きます (デスタッフィング)。

CAN プロトコルでは、アドレスされるのはバスノードではありません。アドレス情報は、送信されるメッセージに含まれます。これは、メッセージの内容 (例えばエンジンスピード、オイル温度等等) を識別する識別子 (メッセージの一部) により行われます。この識別子はまた、メッセージの優先度を表示します。識別子のバイナリ値が小さくなると、メッセージの優先度は上がります。

バスアービトレーションに関しては、NDA (Non-Destructive Arbitration) を持つ CSMA/CD (Carrier Sense Multiple Access/Collision Detection) が使用されます。バスノード A がネットワークに亘ってメッセージを送信したい場合、まずバスがアイドル状態であることをチェックします (“Carrier Sense”) (すなわち、どのノードも現在送信していないこと)。これが正しい (しかも同じ瞬間にどのノードも送信を開始しようとしていない) 場合、ノード A はバスマスターになりメッセージを送信します。すべての他ノードは、最初の送信データビット (フレームの開始ビット) の間に受信モードに切替わります。メッセージが正しく受信された後 (それぞのノードにより通知されたことを示します)、それぞのバスノードは、メッセージ識別子をチェックし、必要であればメッセージを格納します。そうでなければ、メッセージは廃棄されます。

2つ以上のノードが同時に送信を開始した場合（“複数アクセス”）、メッセージの衝突は、ビット単位のアービトレーションにより回避されます。（“ワイヤ AND” 機構と一緒に“衝突検出 / 非破壊アービトレーション”、“ドミナント”ビットは“リセシップ”ビットを上書きします）。それぞれのノードはメッセージ識別子（MS ビットファースト）のビットを送信し、バスレベルをモニターします。リセシップ識別子ビットを送信したのにドミナントビット識別子を読み込んだノードはバスアービトレーションを失い、受信モードに切替わります。この条件は、競合しているノードのメッセージ識別子がより小さいバイナリ値（ドミナント状態=論理 0）の時に発生し、競合ノードは、より高い優先度を持ったメッセージを送信します。このようにして、最高位の優先度メッセージを持ったバスノードが、メッセージを繰り返すことによる時間のロスなしにアービトレーション権を勝ち取ります。その他のすべてのノードは、バスが一旦アイドル状態に戻った場合、自動的に送信を繰り返そうとします。アービトレーションに失敗し、後に衝突やエラーにつながるので、異なるノードが同じ識別子を持ったメッセージを送信することは許されていません。

オリジナルの CAN 仕様（バージョン 1.0、1.2 および 2.0A）では、メッセージ識別子は 11 ビットの長さを持ち、2048 個のメッセージ識別子を与えることができます。仕様は、それ以降（バージョン 2.0B に）改定され、識別子の数という制限は撤廃されています。CAN 仕様バージョン 2.0B では、メッセージ識別子として 11 ビットかつ／もしくは 29 ビット長が用いられるようになっています（識別子長 29 ビットにより 5 億 36 百万以上のメッセージ識別子を与えることが出来ます）。バージョン 2.0BCAN はまた“拡張 CAN”とも呼ばれ、バージョン 1.0、1.2 および 2.0A は“標準 CAN”と呼ばれます。

#### 23.14.1 標準 CAN vs 拡張 CAN

データフレームとリモートフレームは 11 ビットの識別子しか持っていないが、CAN 仕様 V2.0A によれば、標準フレームと呼ばれます。これらのフレームでは 2048 の異なるメッセージが識別できます（0-2047 を識別します）。しかし、最低位の優先度（2032-2047）を持つ 16 のメッセージは予約されています。CAN 仕様 V2.0B によれば、拡張フレームは 29 ビットの識別子を持ちます。すでに述べたように、この 29 ビットの識別子は、11 ビットの識別子（“標準 ID”）と 18 ビットの拡張識別子（“拡張 ID”）から構成されます。

CAN V2.0A により規定される CAN モジュールは、標準 CAN プロトコルにより標準フレームの送信・受信のみができます。29 ビットの識別子を用いたメッセージはエラーになります。デバイスが CAN V2.0B により規定される場合、もう 1 つの特徴があります。“Part B Passive”と名づけられたモジュールは、標準フレームの送信・受信のみが可能ですが、エラーフレームを生成すること無しに、拡張フレームを許容します。“Part B Active” デバイスでは標準・拡張両方のフレームの送信・受信が可能です。

#### 23.14.2 ISO モデル

ISO/OSI 参照モデルは、図 23-25 に示されるように、通信システムのプロトコルレイヤーを定義するために用いられます。最高位では、アプリケーションはお互いの間で通信する必要があります。最低位では、いくつかの物理メディアが、電気的信号を与えるために使用されます。

プロトコルの高いレベルはソフトウェアにより実行されます。CAN バス仕様では、メッセージの形式や内容や送信されるメッセージの意味については定義がありません。これらの定義は Volcano、Volvo 自動車の CAN 仕様 J1939、米国重トラック多重ワイヤリング仕様；および Allen-Bradley デバイスネットや Honeywell SDS、産業プロトコルのようなシステムで決められます。

CAN バスモジュール定義は、プロトコル全体の 2 つのレイヤを包括します。

- データリンク層
  - 論理リンク制御 (LLC) 副層
  - メディアアクセス制御 (MAC) 副層
- 物理層
  - 物理信号 (PLS) 副層

LLC 副層は、メッセージフィルタリング、過負荷通知およびエラー回復管理に関係します。LCC 副層の範囲は下記の通りです。

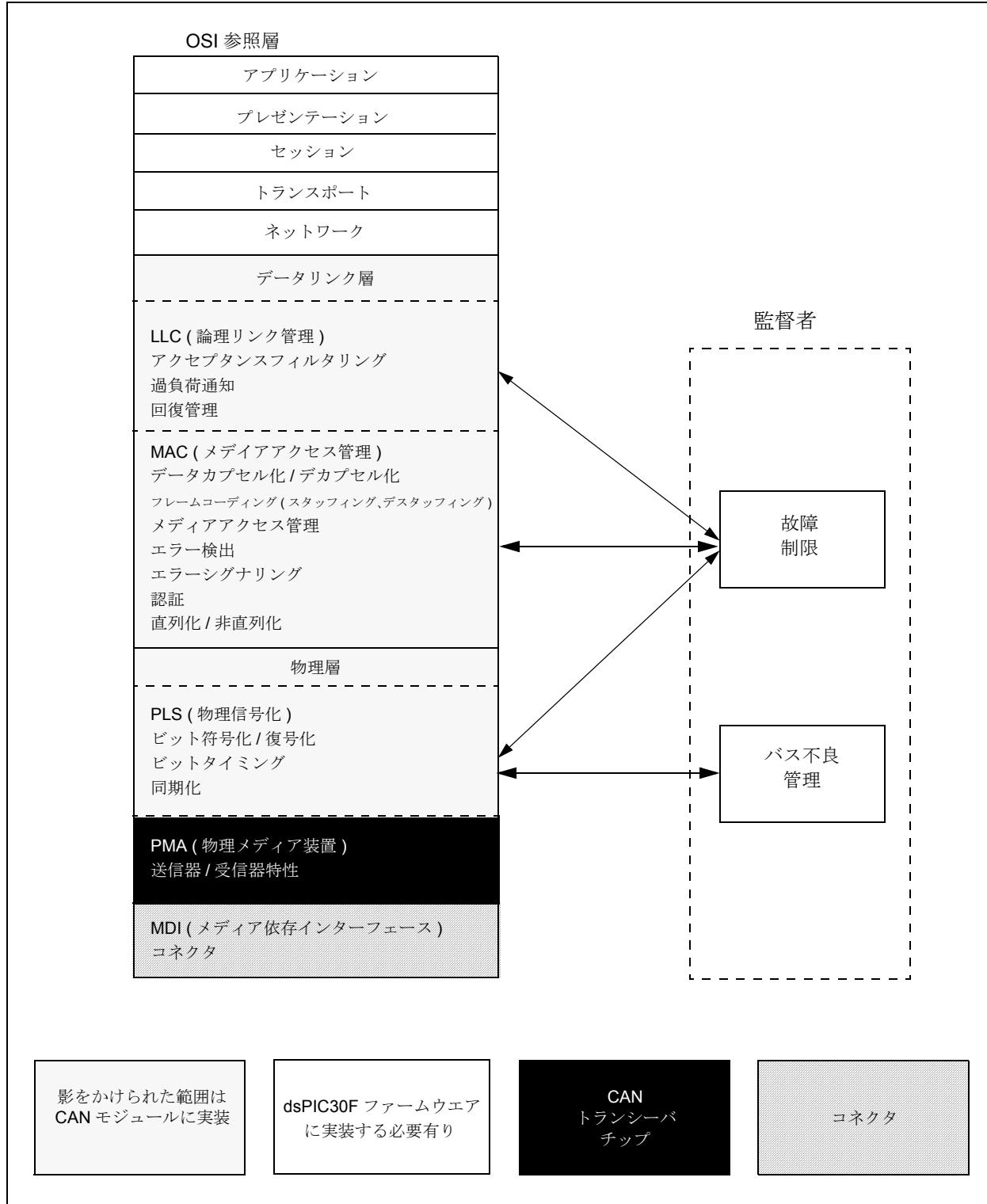
- データ転送やリモートデータ要求のサービスを提供します。
- LCC 副層により受信されたメッセージのうちどのメッセージが実際に受け付けられるべきかを決定します。
- エラー回復管理および過負荷通知の手段を提供します。

MAC 副層は CAN プロトコルのカーネルを表します。MAC 副層はトランスポートプロトコル（すなわち、フレーミング、アービトレーションの実施、エラーチェック、エラー信号および故障制限の制御）を規定します。MAC 副層は LLC 副層から受信されたメッセージを提供し、LLC 副層へ送信されるメッセージを受け付けます。MAC 副層内は、バスが新しい送信を開始できるかどうか、受信が開始したばかりかどうかを決定するところです。MAC 副層は、永久的な故障と短期的な乱れとを区別するセルフチェック機構である故障制限と呼ばれる管理本体により監視されます。

物理副層は、すべての電気的仕様に関して異なるノード間で実際のビットの転送を規定します。PLS 副層は、どのようにして信号が実際に送信されるかを規定し、従って、ビットタイミング、ビット復号化および同期化をとり扱います。

プロトコルの低位層は、ドライバ / レシーバチップおよび、撲り線ペアワイヤリングもしくはオプティカルファイバー等に実装されます。1 つのネットワーク内には、すべてのノードで物理層は同じである必要があります。物理層のドライバ / レシーバの特性は、送信メディアと信号レベルの実装がアプリケーションに最適化されるようにするために、CAN 仕様では規定されません。物理層メディアの最も共通的な例は、多重化ワイヤの仕様であるロードビークル ISO11898 で定義されています。

図 23-25: ISO/OSI 参照モデルにおける CAN バス



## 23.15 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありませんが、その概念は適切であり、修正したり制限を設けて使用できる場合もあります。現状、CAN モジュールに関連する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 #
CAN プロトコルの紹介	AN713

**注：** dsPIC30F ファミリのデバイスに関しての、その他のアプリケーションノートやコードの例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 23.16 改訂履歴

### A 版

これは本ドキュメントの初版です。

### B 版

この版は、dsPIC30F CAN モジュール用の追加技術情報を含みます。

注：



**MICROCHIP**

## 第 24 章. デバイスコンフィギュレーション

### ハイライト

この章は、以下の項目を含んでいます。

24.1 序章 .....	24-2
24.2 デバイスコンフィギュレーションレジスタ .....	24-2
24.3 コンフィギュレーションビットの説明 .....	24-7
24.4 デバイス識別レジスタ .....	24-8
24.5 関連するアプリケーションノート .....	24-9
24.6 改訂履歴 .....	24-10

**24**

デバイスコンフィギュレーション

## 24.1 序章

ユーザーはデバイス構成レジスタを使用してデバイスの特定の部分をカスタマイズし、各自のニーズに合わせた設定を適用できます。デバイス構成レジスタはプログラムメモリマップ内にある非揮発性メモリロケーションで、電源が切れている間も dsPIC デバイスの設定を保持します。構成レジスタは発振器ソース、ウォッチドッグタイマーモード、コード保護設定など、デバイスの全体的な設定情報を保持します。

プログラムメモリロケーションにマップされ、アドレス 0xF80000 から始まります。通常のデバイス操作時にアクセス可能です。この範囲は「コンフィギュレーション空間」とも呼ばれます。

様々なデバイスコンフィギュレーションを選択し、コンフィギュレーションビットをプログラムする ('0' として読み込み) か、プログラムしない ('1' として読み込み) かを決定できます。

## 24.2 デバイスコンフィギュレーションレジスタ

各デバイスコンフィギュレーションレジスタは 24 ビットレジスタですが、各レジスタの下位 16 ビットがコンフィギュレーションデータを保持するのに使用されます。ユーザーは以下の 4 つのデバイスコンフィギュレーションレジスタを使用できます。

- FOSC (0xF80000): 発振器コンフィギュレーションレジスタ
- FWDT (0xF80002): ウォッチドッグタイマーコンフィギュレーションレジスタ
- FBORPOR (0xF80004): BOR および POR コンフィギュレーションレジスタ
- FGS (0xF8000A): 一般コードセグメントコンフィギュレーションレジスタ

デバイスコンフィギュレーションレジスタは、ランタイム自己プログラム (RTSP)、インサーキットシリアルプログラミング™ (ICSP™) を使用してプログラムするか、デバイスのプログラマーでプログラムします。

**注:** 後述のコンフィギュレーションレジスタ説明で示される全てのデバイスコンフィギュレーションビットが全てのデバイスで使用できるとは限りません。詳細についてはデバイスデータシートを参照してください。



## レジスタ 24-2: FWDT: ウオッチドッグタイマーコンフィギュレーションレジスタ

上位バイト:

U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
ビット 23							ビット 16

中位バイト:

R/P	U	U	U	U	U	U	U
FWDTEN	—	—	—	—	—	—	—
ビット 15							ビット 8

下位バイト:

U	U	R/P	R/P	R/P	R/P	R/P	R/P
		FWPSA<1:0>		FWPSB<3:0>			
ビット 7							ビット 0

ビット 23-16 未実装: ‘0’ が読み込まれます。

ビット 15 **FWDTEN:** ウオッチドッグタイマー有効化構成ビット  
 1 = ウオッチドッグ有効 (LPRC 発振器は無効化できません。RCON レジスタで SWDTEN ビットをクリアしても効果はありません。)  
 0 = ウオッチドッグ無効 (LPRC 発振器は RCON レジスタの SWDTEN ビットをクリアして無効化できます。)

ビット 14-6 未実装: ‘0’ が読み込まれます。

ビット 5-4: **FWPSA<1:0>:** ウオッチドッグタイマー プリスケーラ A のプリスケール値選択  
 11 = 1:512  
 10 = 1:64  
 01 = 1:8  
 00 = 1:1

ビット 3-0 **FWPSB<3:0>:** ウオッチドッグタイマー プリスケーラ B のプリスケール値選択  
 1111 = 1:16  
 1110 = 1:15  
 •  
 •  
 •  
 0001 = 1:2  
 0000 = 1:1

凡例:

R = 読み取り可能ビット

P = プログラム可能ビット

U = 未実装ビット

**レジスタ 24-3: FBORPOR: BOR および POR コンフィギュレーションレジスタ**

上位バイト:							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—

ビット 23

ビット 16

中位バイト:							
R/P	U	U	U	U	R/P	R/P	R/P
MCLREN	—	—	—	—	PWMPIN	HPOL	LPOL

ビット 15

ビット 8

下位バイト:							
R/P	U	R/P	R/P	U	U	R/P	R/P
BOREN	—	BORV<1:0>	—	—	—	FPWRT<1:0>	—

ビット 7

ビット 0

ビット 23-16 未実装: ‘0’ が読み込まれます。

ビット 15 **MCLREN:** MCLR ピン機能有効ビット  
 1 = ピン機能は MCLR (デフォルトの場合)  
 0 = ピン機能は無効

ビット 14-11 未実装: ‘0’ が読み込まれます。

ビット 10 **PWMPIN:** モーター制御 PWM モジュールピンモードビット  
 1 = PWM モジュールピンはデバイス RESET で PORT レジスタで制御されます。(3 値状態)  
 0 = PWM モジュールピンはデバイス RESET で PWM モジュールで制御されます。(出力モードとなる)

ビット 9 **HPOL:** モーター制御 PWM モジュール高位側極性制御ビット  
 1 = PWM モジュール高位側出力ピンはアクティブ High 極性とします。  
 0 = PWM モジュール高位側出力ピンはアクティブ Low 極性とします。

ビット 8 **LPOL:** モーター制御 PWM モジュール低位側極性制御ビット  
 1 = PWM モジュール低位側出力ピンはアクティブ High 極性とします。  
 0 = PWM モジュール低位側出力ピンはアクティブ Low 極性とします。

ビット 7 **BOREN:** PBOR 有効化ビット  
 1 = PBOR 有効  
 0 = PBOR 無効

ビット 6 未実装: ‘0’ が読み込まれます。

ビット 5-4 **BORV<1:0>:** ブラウンアウト電圧選択ビット  
 11 = 2.0V  
 10 = 2.7V  
 01 = 4.2V  
 00 = 4.5V

ビット 3-2 未実装: ‘0’ が読み込まれます。

ビット 1-0 **FPWRT<1:0>:** パワーオンリセットタイマー値選択ビット  
 11 = PWRT = 64 ms  
 10 = PWRT = 16 ms  
 01 = PWRT = 4 ms  
 00 = パワーアップタイマー無効

凡例:

R = 読み取り可能ビット

P = プログラム可能ビット

U = 未実装ビット

## レジスタ 24-4: FGS: 一般コードセグメントコンフィギュレーションレジスタ

上位バイト :							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
ビット 23							ビット 16

中位バイト :							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
ビット 15							ビット 8

下位バイト :							
U	U	U	U	U	U	P	P
—	—	—	—	—	—	GCP	GWRP
ビット 7							ビット 0

ビット 未実装 : ‘0’ が読み込まれます。

23-2

ビット 1 **GCP:** 一般コードセグメントコード保護ビット

1 = ユーザープログラムメモリはコードで保護されません。  
0 = ユーザープログラムメモリはコードで保護されます。

ビット 0 **GWRP:** 一般コードセグメント書き込み保護ビット

1 = ユーザープログラムメモリは書き込み保護されません。  
0 = ユーザープログラムメモリは書き込み保護されます。

注: BCP および GWRP コンフィギュレーションビットは ‘0’ にしかプログラムできません。

凡例 :

R = 読み取り可能ビット

P = プログラム可能ビット

U = 未実装ビット

## 24.3 コンフィギュレーションビットの説明

このセクションでは、各デバイスコンフィギュレーションビットに関する特定の機能を説明します。

### 24.3.1 発振器コンフィギュレーションビット

FOSC デバイスコンフィギュレーションレジスタ内のコンフィギュレーションビットについて詳しい説明は、[第7章.「発振器」](#)を参照してください。

### 24.3.2 BOR および POR コンフィギュレーションビット

FBORPOR コンフィギュレーションレジスタ内の BOR および POR コンフィギュレーションビットは、デバイスのプラウンアウトリセット電圧を設定するのに使用します。これにより、プローショナウトリセット回路を有効にしパワーアップタイマー遅延時間を設定します。これらの構成ビットについての詳細情報は、[第8章.「リセット」](#)を参照してください。

### 24.3.3 モーター制御 PWM モジュールコンフィギュレーションビット

モーター制御 PWM モジュールコンフィギュレーションビットは FBORPOR コンフィギュレーションレジスタ内に位置し、PWM モジュールを持つデバイスにのみ存在します。PWM モジュールに関するコンフィギュレーションビットには以下の 2 つの機能があります。

1. デバイス RESET で PWM ピンの状態を選択 (ハイインピーダンスか出力)
2. PWM ピンのアクティブ信号極性を選択。高位側および低位側 PWM ピンの極性は独立して選択されます。

これらのコンフィギュレーションビットについての詳細情報は、[第15章.「モーター制御PWM」](#)を参照してください。

### 24.3.4 一般コードセグメントコンフィギュレーションビット

FGS コンフィギュレーションレジスタ内的一般コードセグメントコンフィギュレーションビットは、ユーザープログラムメモリ空間のコード保護または書き込み保護に使用します。一般コードセグメントには、割り込みベクトルテーブル空間 (0x000000-0x0000FE) 以外の全てのユーザープログラムメモリが含まれます。

一般コードセグメントが GCP コンフィギュレーションビット (FGS<1>) を ‘0’ にプログラムすることにより保護されている場合、デバイスプログラムメモリはインサーキットシリアルプログラミング (ICSP) またはデバイスのプログラマーによってデバイスから読み出すことはできません。さらに、一般コードセグメントを先に消去しなくてはデバイスに追加のコードをプログラムできません。

一般セグメントがコード保護されている場合でも、ユーザーコードテーブル読み込み命令を介してプログラムメモリにアクセスすることができます。または、データ空間からのプログラム空間可視性 (PSV) アクセスも可能です。

GWRP (FGS<0>) コンフィギュレーションビットがプログラムされている場合、ユーザープログラムメモリ空間への書き込みは全て無効になります。

#### 24.3.4.1 一般コードセグメントコンフィギュレーションビットグループ

FGS コンフィギュレーションレジスタ内の GCP および GWRP コンフィギュレーションビットはグループ単位でプログラム/消去する必要があります。これらコンフィギュレーションビットのいずれか 1 つまたは両方が ‘0’ にプログラムされている場合、いずれかのビットの状態を変更するためには、フルチップ消去を実行する必要があります。

## 24.4 デバイス識別レジスタ

dsPIC30Fデバイスはコンフィギュレーション空間に識別情報を提供する2つのレジスタのセットを持ちます。

### 24.4.1 デバイス ID (DEVID) レジスタ

コンフィギュレーションメモリ空間ロケーション 0xFF0000 および 0xFF0002 はデバイス製造時に設定された読み取り専用デバイス ID 番号を格納するのに使用されます。この番号は dsPIC30F デバイスのタイプとシリコン改訂情報を識別します。

ユーザーはデバイス ID レジスタをテーブル読み取り命令を使用して読み取ります。

### 24.4.2 ユニット ID フィールド

ユニット ID フィールドはコンフィギュレーションメモリ空間ロケーションの 0x800600 から 0x80063E に位置します。このフィールドは 32 のプログラムメモリロケーションから成り、Microchip の工場で一意のデバイス情報がプログラムされています。このフィールドはユーザーによって書き込み、消去が可能ですが、書き込みはテーブル読み取り命令を使用して行われます。

詳細については、Microchip テクニカルサポートまたは最寄の Microchip 代理店にお問い合わせ下さい。

## 24.5 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありません。ただし、その概念は適切であり、修正や可能な制限をして使用できる可能性もあります。現状、デバイス構成モジュールに関連する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 #

現在のところ、関連するアプリケーションノートはありません。

**注:** デバイスの dsPIC30F ファミリーに関しての、追加のアプリケーションノートやコード例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 24.6 改訂履歴

### A 版

これは本ドキュメントの最初の改訂版です。

### B 版

この版は、dsPIC30F のデバイスコンフィギュレーションモジュールに関する技術情報の変更を含んでいます。



**MICROCHIP**

## 第 25 章 . 開発ツールのサポート

### ハイライト

この章は、以下の項目を含んでいます。

25.1 序章 .....	25-2
25.2 Microchip ハードウェアおよび言語ツール .....	25-2
25.3 サードパーティハードウェア / ソフトウェアツールおよびアプリケーションライブラリ .....	25-6
25.4 dsPIC30F ハードウェア開発ボード .....	25-11
25.5 関連するアプリケーションノート .....	25-15
25.6 改訂履歴 .....	25-16

25

開発ツール  
サポート

注： この章に記載の開発ツールのうち、このマニュアル発行の時点ではまだ入手できないものもありますが、現在開発中です。ツールの詳細が変更になる可能性もあります。最新の情報および各ツールの入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セールスオフィスにお問い合わせくださいますよう、お願いします。

## 25.1 序章

Microchip は dsPIC アーキテクチャをサポートするための総括的な開発ツールとライブラリのパッケージを提供しています。さらに、追加の dsPIC デバイスのサポートのために、多くのサードパーティのツール製造業者と提携しています。

## 25.2 Microchip ハードウェアおよび言語ツール

この章で紹介するツールは以下の通りです。

- MPLAB® 統合開発環境 (IDE)
- MPLAB C30 C コンパイラ、アセンブラー、ライブラリアンを含む dsPIC 言語スイーツ
- MPLAB SIM ソフトウェアシミュレータ
- MPLAB ICE 4000 インサーキットエミュレータ
- MPLAB ICD 2 インサーキットデバッガ
- PRO MATE® II ユニバーサルデバイスプログラマー
- PICSTART® Plus 開発プログラマー

### 25.2.1 MPLAB 6.XX 統合開発環境ソフトウェア

注： この製品は現在、Microchip 社の Web サイト、[www.microchip.com](http://www.microchip.com) で入手いただけます。

MPLAB 統合開発環境 (IDE) は無償で配布されています。MPLAB IDE ソフトウェアは Microchip コントローラ設計アプリケーションの開発とデバッグのためのツールセットを持つデスクトップ開発環境です。MPLAB IDE を使用すると、異なる開発およびデバッグの作業を素早く切り替えることができます。Windows® オペレーティング環境向けに設計された、パワフルかつ入手容易な開発ツールです。また、MPLAB エディター、MPLAB ASM30 アセンブラー、MPLAB SIM ソフトウェアシミュレータ、MPLAB LIB30 ライブラリ、MPLAB LINK30 リンカー、MPLAB ICE 4000 インサーキットエミュレータ、PRO MATE II プログラマー、インサーキットデバッガ (ICD 2) などの Microchip 社の開発システムツールに共通のユーザーインターフェースになります。MPLAB IDE を使用することにより、ユーザーは同一のユーザーインターフェースで編集、コンパイル、エミュレートを柔軟に行えます。エンジニアは PICmicro® マイクロコントローラと同じ設計環境で dsPIC デバイスの開発コードを設計できます。

MPLAB IDE は 32 ビットの Windows ベースアプリケーションです。多くの高度な機能を、最新の操作の簡単なインターフェースでエンジニアに提供します。MPLAB IDE には以下が統合されています。

- 機能満載のカラーコード化テキストエディタ
- ビジュアルな表示で操作の容易なプロジェクトマネージャ
- ソースレベルのデバッグ
- ‘C’ 向けの拡張ソースレベルデバッガ
  - (構造、自動変数など)
- カスタマイズ可能なツールバーとキーマッピング
- プロセッサの状態が一目で分かるように表示する動的ステータスバー
- コンテキストに対応した対話式オンラインヘルプ
- 統合 MPLAB SIM 命令シミュレータ
- PRO MATE II および PICSTART Plus デバイスプログラマ向けユーザーインターフェース (別売り)
- MPLAB ICE 4000 インサーキットエミュレータ (別売り) 向けユーザーインターフェース
- MPLAB ICD 2 インサーキットデバッガ (別売り) 向けユーザーインターフェース

MPLAB IDE を使用して、エンジニアは以下を実行できます。

- ・アセンブリまたは ‘C’ によるソースファイルの編集
- ・ワンタッチのコンパイルとエミュレータまたはシミュレータへの dsPIC プログラムメモリのダウンロード。  
すべてのプロジェクト情報が更新されます。
- ・以下を使用したデバッグ
  - ソースファイル
  - マシンコード
  - ミクスドモードソースおよびマシンコード

MPLAB IDE は複数の開発およびデバッグ対象で使用できるため、ユーザーは最小の待ち時間でコスト費用効果の高いシミュレータと機能満載のエミュレータの間を容易に行き来できます。

## 25.2.2 dsPIC 言語スイーツ

**注：** この製品は Microchip 社の Web サイト [www.microchip.com](http://www.microchip.com) で入手いただけます。アセンブラー、リンカー、ライブラリアンは MPLAB IDE に含まれています。MPLAB C30 C コンパイラの入手については、最寄りの Microchip のセールスオフィスにお問い合わせください。

Microchip テクノロジー MPLAB C30 C コンパイラは、完全な機能を備えながら操作の容易な言語製品です。この製品を使用することにより、dsPIC アプリケーションコードを高水準 C 言語で記述し、マシンオブジェクトコードに完全に変換してマイクロコントローラのプログラミングに使用できます。コード障害を取り除き、設計者がプログラム要素よりもプログラムフローそのものに集中できるようにすることで、コード開発を簡素化します。コンパイルではいくつかのオプションがあり、ユーザーはコードの特性の効率性を最大限にできるオプションを選択できます。

この製品は ANSI に完全に準拠しており、マイクロコントローラの dsPIC ファミリーのための標準ライブラリを有しています。dsPIC デバイスの多くの高度な機能を持ち、非常に効率的なアセンブリコード生成ができます。

MPLAB C30 はまた、割り込みや周辺装置などのハードウェアを効率よくサポートする拡張機能も提供しています。MPLAB IDE とも完全に統合されており、高度なソースデバッグが可能です。この製品の特徴は以下の通りです。

- ・本来の 16 ビットデータタイプ
- ・レジスタベースの 3 オペランド命令を有効活用
- ・複合アドレス指定モード
- ・効率的なマルチビットシフトオペレーション
- ・効率的な符号付き / 符号無し比較

MPLAB C30 にはアセンブラー、リンカー、ライブラリアンが付属しています。これにより、ミクスドモード C およびアセンブリプログラムを記述でき、作成したオブジェクトファイルを 1 つの実行ファイルにリンクできます。コンパイラは別売りですが、アセンブラー、リンカー、ライブラリアンは MPLAB IDE の付属品として無料で配布されています。

## 25.2.3 MPLAB SIM ソフトウェアシミュレータ

**注：** この製品は MPLAB IDE に同梱されています。

MPLAB SIM ソフトウェアシミュレータを使用すると、命令レベルで dsPIC デバイスをシミュレートし PC ホスト環境でのコード開発が可能です。命令により、データ領域が検査、修正できるようになります。実行はシングルステップ、ブレークまで実行続行またはトレースモード<sup>(1)</sup>で行えます。

MPLAB SIM ソフトウェアシミュレータは MPLAB C30 コンパイラおよびアセンブラーを使用したシンボリックデバッグを完全にサポートしています。ラボラトリ環境外の開発およびデバッグも柔軟に行うことができるこの製品は、優れたマルチプロジェクトソフトウェア開発ツールといえます。

**注 1:** 周辺装置サポートなどの機能は、このマニュアルの発行時点では実装されていません。最新の情報については、Microchip 社の Web サイトを訪れて確認してくださいか、最寄りの Microchip セールスオフィスにお問い合わせください。

## 25.2.4 MPLAB ICE 4000 インサーキットエミュレータ

**注:** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セルスオフィスにお問い合わせください。

MPLAB ICE 4000 インサーキットエミュレータは、製品開発エンジニアに dsPIC デバイスの完全なハードウェア設計ツールを提供します。エミュレータのソフトウェア制御は MPLAB IDE で提供されます。

MPLAB ICE 4000 はトレース、トリガー、データモニタリングなどの拡張機能を持つ機能満載のエミュレータシステムになる予定です。互換性のあるプロセッサモジュールにより、異なるプロセッサのエミュレーションを容易に認識できるようになります。

MPLAB ICE 4000 は拡張されたハイエンド PICmicro Microchip コントローラ PIC18CXXX、PIC18FXXX デバイスやデジタル信号コントローラの dsPIC ファミリーをサポートします。 MPLAB ICE 4000 インサーキットエミュレータのモジュラー構造は新しいデバイスのサポートのための拡張も可能にします。

MPLAB ICE 4000 インサーキットエミュレータシステムはリアルタイムエミュレーションシステムとして設計されており、他社の高価な開発ツールが通常有するような高度な機能も搭載される予定です。特徴は以下の通りです。

- 最高 50 MHz バス速度または 200 MHz 外部クロック速度の高速エミュレーション
- 最低 1.8 ボルトの低電圧エミュレーション
- 2 メガバイトプログラムエミュレーションメモリで構成され、可能な追加モジュラーメモリは最高 16 メガバイト
- 64K x 136 ビット幅トレースメモリ
- 無制限のソフトウェアブレークポイント
- 複合ブレーク、トレース、トリガーロジック
- 最多 4 レベルのマルチレベルトリガー
- 特定のイベントをトレースするフィルタトリガー機能
- イベントの連続でトリガーされる 16 ビットパスカウンタ
- 16 ビット遅延カウンタ
- 48 ビットタイムスタンプ
- ストップウォッチ機能
- イベント間の時間記録
- 統計に基づくパフォーマンス分析
- コードカバレッジ分析
- USB およびパラレルプリンターポート PC 接続

### 25.2.5 MPLAB ICD 2 インサーキットデバッガ

**注：** この製品は入手可能ですが、現在、dsPIC30F デバイスはサポートしていません。製品のアップグレードについては、Microchip 社の Web サイトでご確認ください。

Microchip 社のインサーキット デバッガおよび MPLAB ICD は、パワフルかつ低コストなランタイム開発ツールになる予定です。PICmicro® および dsPIC FLASH デバイスをベースにしており。

MPLAB ICD 2 は様々なデバイスに組み込まれているインサーキットデバッギング 機能を活用します。この機能を Microchip インサーキットシリアルプログラミング™ プロトコル (ICSP™) と組み合わせて使用することで、MPLAB IDE のグラフィカルユーザーインターフェースで対コスト効果の高いインサーキットデバッグを実行できます。これにより、設計者は変数を確認し、各手順ごとにブレークポイントを設定しながらソースコードを開発、デバッグできます。高速実行でハードウェアのテストをリアルタイムに行えます。以下はこの製品の特徴の一部です。

- デバイス速度までカバーする高速オペレーション
- シリアルコネクタまたは USB PC コネクタ
- 外部電源を持つシリアルインターフェース
- PCインターフェースとしての USB
- アナログアプリケーションおよびその他ノイズに敏感なアプリケーション向け低ノイズ電源 (VPP および VDD)
- 最低 2.0V の低電圧オペレーション
- ICD または低価格のシリアルプログラマとして使用可
- MPLAB ICD としてのモジュラーアプリケーションコネクタ
- 設定可能ブレークポイント
- “スマートウォッチ” 変数ウインドウ
- いくつかのチップリソース (RAM、プログラムメモリ、2 ピン) が必要

### 25.2.6 PRO MATE II ユニバーサルデバイスプログラマー

**注：** この製品は入手可能ですが、現在、dsPIC30F デバイスはサポートしていません。製品のアップグレードについては、Microchip 社の Web サイトでご確認ください。

PRO MATE II ユニバーサルデバイスプログラマーはスタンドアロンモードおよび PC ホストモードで動作する機能満載のプログラマになる予定です。

PRO MATE II デバイスプログラマーはプログラム可能な VDD および VPP 電源を持ち、プログラミングでこのツールが必要になった場合に、プログラムされたメモリを VDDMIN および VDDMAX で検査できます。命令とエラーメッセージを表示する LCD 画面と、コマンドを入力するキーを持ちます。互換性のあるオプションのソケットモジュールはすべてのパッケージタイプをサポートします。

スタンドアロンモードで、PRO MATE II デバイスプログラマーは PICmicro デバイスおよび dsPIC30F デバイスを読み込み、検査、プログラムできます。同じモードでコード保護を設定することもできます。PRO MATE II の特徴は以下の通りです。

- MPLAB IDE で動作
- 使用中にフィールドのアップグレード可能なファームウェア
- 生産時に使用できる DOS コマンドラインインターフェース
- 安全な“スタンドアロン”オペレーションをホスト
- オブジェクトファイルの自動ダウンロード
- プログラムされた各デバイスに一意のシリアル番号を付与する SQTP™ シリアライゼーション
- インサーキットシリアルプログラミングキット (別売り)
- すべてのパッケージオプションをサポートする互換性のあるソケットモジュール (別売り)

## 25.3 サードパーティハードウェア / ソフトウェアツールおよびアプリケーションライブラリ

**注:** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セルスオフィスにお問い合わせください。

Microchip 社は主要なサードパーティツール製造業者とパートナー関係にあり、dsPIC30F 製品ファミリーをサポートした高性能ハードウェアおよびソフトウェアツールの開発が可能です。Microchip 社はこのツールおよびライブラリの初期セットを提供する予定で、これにより、dsPIC30F ベースのアプリケーションを迅速に開発できるようになります。

現在顧客に提供可能な製品のリストに、付加価値サービス（つまり、高いスキルと資格を持つアプリケーション技術サポートの連絡先、参考設計、ハードウェアおよびソフトウェア開発者のレポジトリ）が追加される予定です。

dsPIC30F デバイスファミリーのサードパーティのサポートに関する最新の情報は、Microchip 社の Web サイト ([www.microchip.com](http://www.microchip.com)) でご確認ください。

dsPIC30F ソフトウェアツールおよびライブラリには以下が含まれます。

- サードパーティの C コンパイラ
- 浮動小数および 2 倍精度算術ライブラリ
- DSP アルゴリズムライブラリ
- デジタルフィルタ設計ソフトウェアユーティリティ
- 周辺装置ドライバライブラリ
- CAN ライブラリ
- リアルタイムオペレーティングシステム (RTOS)
- OSEK オペレーティングシステム
- TCP/IP プロトコルスタック
- V.22/V.22bis および V.32 ITU 仕様

dsPIC30F ハードウェア開発ボードツールには、以下が含まれます。

- 汎用開発ボード
- モーター制御開発システム
- 通信用開発ボード

### 25.3.1 サードパーティの C コンパイラ

**注:** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セルスオフィスにお問い合わせください。

Microchip MPLAB C30 C コンパイラに加えて、dsPIC30F は IAR、HI-TECH、カスタムコンピュータサービス (CCS) で開発する ANCI C コンパイラもサポートする予定です。

この製品を使用することにより、dsPIC アプリケーションコードを高水準 C 言語で記述し、マシンオブジェクトコードに完全に変換にしてマイクロコントローラのプログラミングに使用できます。各コンパイラツールは複数のコンパイラのオプションを提供し、生成したコードの特性の効率性を最大限に活かせるオプションを選択できるようにします。

異なる価格帯と機能を持つ C コンパイラソリューションを提案し、顧客がアプリケーションの要件に見合った最適なコンパイラを選択できるようにします。

### 25.3.2 算術ライブラリ

**注:** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セールスオフィスにお問い合わせください。

算術ライブラリは以下を含む（ただしこれらに限らない）複数の標準 C 関数をサポートします。

- `sin()`、`cos()`、`tan()`
- `asin()`、`acos()`、`atan()`,
- `log()`、`log10()`
- `sqrt()`、`power()`
- `ceil()`、`floor()`
- `fmod()`、`frexp()`

算術関数ルーチンは dsPIC30F アセンブリ言語で開発、最適化されアセンブリと C 言語の両方で呼び出し可能です。各関数について不動小数および 2 倍精度も提供される予定です。また、Microchip MPLAB C30 および IAR C コンパイラもサポートされます。

### 25.3.3 DSP アルゴリズムライブラリ

**注:** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セールスオフィスにお問い合わせください。

DSP ライブラリは複数のフィルタリング、畳み込み、ベクトル、マトリックス関数をサポートする予定です。いくつかの関数は以下（ただし、これらに限らない）を含みます。

- カスケイドインフィニットインパルス応答 (MR) フィルタ
- 相関
- 畳み込み
- 有限インパルス応答 (FIR) フィルタ
- 窓関数
- FFT
- LMS フィルタ
- ベクトル加算および減算
- ベクトルドット乗積
- ベクトル累乗
- マトリックス加算および減算
- マトリックス乗算

## 25.3.4 DSP デジタルフィルタ設計ソフトウェアユーティリティ

**注：** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セールスオフィスにお問い合わせください。

Microchip 社はローパス、ハイパス、バンドパス、バンド停止 IIR および FIR フィルタのための最適なアセンブリコードの開発を可能にするデジタルフィルタ設計ソフトウェアツールを提供する予定です。このツールにはグラフィカルユーザーインターフェースの 16 ビット小数部データサイズフィルタ係数生成機能も含みます。アプリケーション開発者が必要なフィルタ周波数特性を入力すると、DSP フィルタデザインソフトウェアはフィルタコードと係数を生成します。理想的なフィルタ周波数応答と時間領域プロットが分析用に生成されます。

最大 513 タップの FIR フィルタ長、最大 10 カスケードセクションの IIR フィルタがサポートされることになります。

すべての IIR および FIR ルーチンはアセンブリ言語で生成され、アセンブリと C 言語の両方で呼び出し可能です。また、Microchip MPLAB C30 C コンパイラもサポートされます。

## 25.3.5 周辺装置ドライバライブラリ

**注：** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セールスオフィスにお問い合わせください。

Microchip 社は以下を含む（ただし、これらに限らない）dsPIC30F ハードウェア周辺装置の設定と制御をサポートする周辺装置ドライバライブラリを提供する予定です。

- アナログ - デジタルコンバータ
- モーター制御 PWM
- 直交エンコーダインターフェース
- UART
- SPI
- データコンバータインターフェース
- I<sup>2</sup>C<sup>TM</sup>
- 汎用タイマー
- 入力キャプチャ
- 出力比較 / 単純 PWM

### 25.3.6 CAN ライブラリ

**注:** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セールスオフィスにお問い合わせください。

Microchip 社は dsPIC30F CAN 周辺装置をサポートする CAN ドライバライブラリを提供する予定です。サポートされる CAN 関数の例は以下の通りです。

- CAN モジュール初期化
- CAN オペレーションモード設定
- CAN ポーレート設定
- CAN マスク設定
- CAN フィルタ設定
- CAN メッセージ送信
- CAN メッセージ受信
- CAN シーケンス放棄
- CAN TX エラーカウント取得
- CAN RX エラーカウント取得

### 25.3.7 リアルタイムオペレーティングシステム (RTOS)

**注:** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セールスオフィスにお問い合わせください。

dsPIC30F 製品ファミリー向けのリアルタイムオペレーティングシステム(RTOS)ソリューションが提供される予定です。これらの RTOS ソリューションは必要な関数コールおよびオペレーティングシステムルーチンを提供し、マルチタスクアプリケーションの効率的な C および / またはアセンブリコードの記述を可能にします。さらに、プログラムおよびデータメモリリソースが限られているこれらのアプリケーションを可能とする RTOS ソリューションも提供されます。様々な RTOS アプリケーション要件をサポートできるように設定可能な最適化されたカーネルも使用可能になります。

RTOS ソリューションには真のプリエンプティブなマルチタスクスケジューラと協調タイマスケジューラがあり、いずれも dsPIC30F デバイスで適切に稼動するように設計される予定です。RTOS 実装によって、システムカーネルで提供される関数コールの例は以下の通りです。

- タスク制御
- メッセージ受送信
- イベント処理
- リソース制御
- セマフォ制御
- 様々な方法でのタイミング調整
- メモリ管理提供
- 割り込みおよびスワップタスク処理

大部分の機能が ANSI C で記述されていますが、タイムクリティカルな関数だけはアセンブリで最適化されていますので、実行時間を短縮しコードを最大限に効率化できます。これらの ANSI C およびアセンブリルーチンは Microchip MPLAB C30 C コンパイラーでサポートされる予定です。

RTOS には電子マニュアルが付属され、ユーザーが RTOS を効率的に理解し、自分のアプリケーションに実装するのを助けています。

## 25.3.8 OSEK オペレーティングシステム

**注：** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セルスオフィスにお問い合わせください。

ビーグルソフトウェア標準 OSEK/VDX のオペレーティングシステムが dsPIC30F 製品ファミリーのサポートのために開発される予定です。OSEK、“Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug”（自動車用エレクトロニクスオープンシステムおよび対応インターフェース）の機能は VDX（自動車分散エグゼクティブ）と融合し、OSEK/VDX が誕生しました。

標準インターフェースおよびプロトコルに基づいた構成可能なモジュール構成の RTOS ソフトウェアが提供される予定です。構成可能なモジュール実装により、自動車分散制御装置に可搬性と拡張性が加わります。

様々な OSEK COM モジュールが提供される予定で、例えば以下のようないわゆるがあります。

- OSEK/COM 標準 API
- OSEK/COM 通信 API
- OSEK/COM ネットワーク API
- OSEK/COM 標準プロトコル
- OSEK/COM デバイスドライバインターフェース

Microchip 社はまた、内部および外部 CAN ドライバもサポートする予定です。物理的レイヤーは通信制御装置のハードウェアに統合され、OSEK 仕様ではカバーされません。

大部分のモジュールは ANSI C で記述されていますが、タイムクリティカルな機能や周辺についてはアセンブリで最適化されていますので、実行時間を短縮しコードを最大限に効率化できます。また、Microchip MPLAB C30 C コンパイラもサポートされます。

## 25.3.9 TCP/IP プロトコルスタック

**注：** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セルスオフィスにお問い合わせください。

Microchip 社は dsPIC30F 製品ファミリーに実装されるインターネット接続ソリューションとして様々な TCP/IP スタックレイヤーソリューションを提供する予定です。簡易実装とフルスタック実装の両方が提供され、アプリケーションに最適な TCP/IP スタックソリューションを選択できるようになります。

FTP、TFTP、SMTP などのアプリケーションプロトコルレイヤー、TCP、UDP、ICMP、IP などのトランスポートレイヤーおよびインターネットレイヤー、PPP、SLIP、ARP、DHCP などのネットワークアクセスレイヤーが提供されます。最小 UDP/IP スタックなどの様々な設定が、接続要件の制限の範囲内で使用できます。

ほとんどのスタックプロトコル関数は Microchip の MPLAB C30 C 言語で開発、最適化されます。特定の dsPIC30F ハードウェア周辺装置およびイーサネットドライバのためのアセンブリ言語コーディングを開発してコードサイズと実行時間を最適化する可能性もあります。これらのアセンブリ言語特有のルーチンはアセンブリと C の両方から呼び出せるようになる予定です。

TCP/IP プロトコルスタックには電子マニュアルが付属され、ユーザーがこのプロトコルスタックを効率的に理解し、自分のアプリケーションに実装するのを助けます。

## 25.3.10 V.22/V.22bis および V.32 仕様

**注：** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セールスオフィスにお問い合わせください。

Microchip 社は ITU Compliant V.22/V.22bis (1200/2400 bps) および V.32 (9600 bps) 非トレリスコーディングモデム仕様を提供し、“つながる”アプリケーションの範囲をサポートする予定です。

これらのモデム仕様よりメリットを得るアプリケーションは数多くあります。例えば、以下のようなアプリケーションにメリットを与えます。

- ・インターネット対応ホームセキュリティシステム
- ・インターネット接続電気、ガス、水道メーター
- ・インターネット接続自動販売機
- ・スマートアプリケーション
- ・工業モニタリング
- ・POS 端末
- ・セットトップボックス
- ・ドロップボックス
- ・ファイアパネル

ほとんどの ITU 仕様モジュールは Microchip の MPLAB C30 C 言語で開発、最適化されます。特定の dsPIC30F ハードウェア周辺装置およびキートラニッシュタ、受信フィルタリングルーチンなどはアセンブリ言語コーディングで開発してコードサイズと実行時間を最適化する可能性もあります。これらのアセンブリ言語特有のルーチンはアセンブリと C の両方から呼び出せるようになる予定です。

モデムライブラリには電子マニュアルが付属され、ユーザーがライブラリ関数を効率よく理解、実装するのを助けます。

## 25.4 dsPIC30F ハードウェア開発ボード

**注：** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セールスオフィスにお問い合わせください。

Microchip 社は、アプリケーション開発者が主要な設計要件を迅速に試作、検証することを可能にする 3 つのハードウェア開発ボードを初期に提供する予定です。各ボードでは主要な dsPIC30F 周辺装置が動作し、Microchip MPLAB インサーキット デバッガ (ICD 2) ツールのサポートにより、コスト効率の高い dsPIC30F のデバッグとプログラミングの環境を提供します。以下の 3 つの初期ボードが提供されます。

- ・汎用開発ボード
- ・モーター制御開発システム
- ・通信開発ボード

## 25.4.1 汎用開発ボード

**注:** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セールスオフィスにお問い合わせください。

dsPIC30F 汎用開発ボードはアプリケーション設計者に dsPIC30F 16 ビットアーキテクチャ、高性能周辺装置およびパワフルな命令セットの構築が容易にできる安価な開発ツールを提供します。この開発ボードは主要な設計要件の迅速な開発、検証を可能にする理想的な試作ツールとなります。

汎用開発ボードの主な特徴と属性は以下の通りです。

- 様々な dsPIC30F パッケージをサポート
- CAN 通信チャネル
- RS-232 および RS-485 通信チャネル
- イン/アウトジャックを持つ Codec インターフェース
- インサーキット デバッガインターフェース
- MPLAB ICE 4000 エミュレーションをサポート
- Microchip 温度センサー
- ユーザー入力信号をサポートする Microchip Op Amp 回路
- Microchip アナログ - デジタルコンバータ
- 2x16 LCD
- 汎用プロトタイプ領域
- 様々な LED、スイッチ、ポテンショメータ

汎用開発ボードは 9V 電源、RS-232 I/O ケーブル、プログラム可能 dsPIC30F デバイス、開発ボードでのデモンストレーションプログラム実行を可能にするソフトウェア例および適切なマニュアルと共に出荷されます。

## 25.4.2 モーター制御開発システム

**注：** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただか、最寄りの Microchip セールスオフィスにお問い合わせください。

dsPIC30F モーター制御開発システムは 3 つの主要な初期コンポーネントをアプリケーション開発者に提供し、BLDC、PMAC および ACIM アプリケーションの迅速な試作と検証を可能にします。3 つの主要コンポーネントは以下の通りです。

- dsPIC30F モーター制御メインボード
- 3 相低電圧モジュール
- 3 相高電圧電源モジュール

メイン制御ボードは dsPIC30F6010 デバイス、様々な周辺装置インターフェース、カスタムインターフェースヘッダーシステムをサポートし、複数の異なるモーター電源モジュールの接続を可能にします。制御ボードはまた、インクリメンタルロータリエンコーダやホール効果センサーなどの機械式位置センサー、およびカスタム回路用のブレッドボード領域も提供する予定です。主要制御ボードは電源を標準のプラグイン変圧器から取ります。

低電圧モジュールは 60 ボルト未満の DC バス電圧を必要とする 3 相モーターアプリケーション向けに最適化され、400W の電源出力を提供します。3 相低電圧モジュールは BLDC および PMAC モーターの電源となります。

高電圧電源モジュールは最大 400 ボルトの DC バス電圧と最大 1 キロワットの電源出力を必要とする 3 相モーターアプリケーション向けに最適化されています。高電圧モジュールは dsPIC30F デバイスで制御されるアクティブな力率改善回路を持ちます。この電源モジュールは AC インダクションモーターおよび電源インバータアプリケーション向けです。

両方の電源モジュールが制御インターフェースから制御可能な自動 FAULT 保護と電気的アイソレーション機能を持ちます。また、両方の電源モジュールボードがメイン制御ボードへの安定化された電圧と電流信号を提供します。インクリメンタルエンコーダ、ホール効果センサまたはタコメーターセンサーなどのすべての位置フィードバックデバイスは、モーター制御回路からは分離され、メイン制御ボードに直接接続されます。両方のモジュールにモーター駆動回路が組み込まれる予定です。

## 25.4.3 通信開発ボード

**注：** この製品は、現在（このマニュアル発行の時点）開発段階にあります。ツールの詳細が変更になる可能性もあります。最新の情報および入手の可否についての情報は、Microchip 社の Web サイトを訪れて入手していただくか、最寄りの Microchip セールスオフィスにお問い合わせください。

dsPIC30F 通信開発ボードは PSTN またはイーサネット通信チャネル上の V.22/V.22bis および V.32 (非トレリスコーディング) ITU 仕様を組み合わせた TCP/IP プロトコルレイヤーを実装し、アプリケーション開発者に様々な通信ソリューションの開発と評価のための基本的なプラットフォームを提供します。

通信開発ボードの主な特徴と属性は以下の通りです。

- dsPIC30F6014 デバイスをサポート
- メディアアクセス制御 (MAC) および PHY インターフェース
- DAA/AFE を持つ PSTN インターフェース
- RS-232 および RS-485 通信チャネル
- インサーキット デバッグインターフェース
- MPLAB ICE 4000 エミュレーションをサポート
- Microchip 温度センサー
- Microchip デジタル - アナログコンバータ
- 2x16 LCD
- 汎用プロトタイプ領域
- 様々な LED、スイッチ、ポテンショメータ

通信開発ボードは 9V 電源、RS-232 I/O ケーブル、プログラム可能 dsPIC30F デバイス、開発ボードでの通信デモンストレーションプログラム実行を可能にする通信ソフトウェア例および適切なマニュアルと共に出荷されます。

## 25.5 関連するアプリケーションノート

このセクションでは、この章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に dsPIC30F 製品ファミリー用に書かれているわけではありません。ただし、その概念は適切であり、修正や可能な制限をして使用できる可能性もあります。現状、開発ツールのサポートに関する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 #

現在のところ、関連するアプリケーションノートはありません。

**注：** デバイスの dsPIC30F ファミリーに関しての、追加のアプリケーションノートやコード例に関しては、Microchip のウェブサイト ([www.microchip.com](http://www.microchip.com)) をご覧下さい。

## 25.6 改訂履歴

### A 版

これは dsPIC30F 開発ツールのサポートの説明文書の初版です。

### B 版

本章に関する技術内容や編集改訂版はありませんが、マニュアル全体を通して B 版を反映するために、この章は更新されています。



## 第 26 章 . 付録

### ハイライト

この章には以下の項目が含まれています。

付録 A: I <sup>2</sup> C™ 概要 .....	26-2
付録 B: CAN 概要 .....	26-12
付録 C: CODEC プロトコル概要 .....	26-26

## 付録 A: I<sup>2</sup>C<sup>TM</sup> 概要

本付録では、I<sup>2</sup>C<sup>TM</sup> バスに関する概要を示し、付録内の A.2 項「I<sup>2</sup>C デバイスのアドレスを指定する」では、I<sup>2</sup>C モードにおける SSP モジュールの動作について説明します。

I<sup>2</sup>C バスは、2 線式のシリアルインターフェースです。当初の仕様、すなわち標準のモードは、100Kbps までのデータ転送向けです。高度仕様では、高速モード (400Kbps) がサポートされています。標準モードおよび高速モードデバイスの両方を同じバスに接続する場合には、低速バスの場合であれば動作します。

I<sup>2</sup>C インターフェースには全体を総括するプロトコルが用いられ、確実にデータを送信、受信できます。データ送信時には、あるデバイスが「マスター」となり、バスにおける送信を開始させ、クロック信号を発生させてその送信を許可する役割を果たし、一方別のデバイスはいわば「スレーブ」の役割を果たします。スレーブプロトコールのすべてが、一齊呼び出しサポートを除き SSP モジュールのハードウェアで実行されますが、マスタープロトコルの部分は PIC16CXX ソフトウェアでアドレスを作成する必要があります。これに対し、MSSP モジュールは、I<sup>2</sup>C マスタープロトコル、ジェネラルコールアドレスおよび 1Mbps までのデータ送信をすべてハードウェアでサポートしています。1Mbps データ送信は、マイクロチップのシリアル EEPROMs の一部によりサポートされています。表 A-1 は、I<sup>2</sup>C バスと関連する用語の一部を定義したものです。

I<sup>2</sup>C インターフェースプロトコルでは、各デバイスにアドレスが指定されています。マスターがデータ送信を開始させようとする場合、まず最初に「talk」しようとするデバイスのアドレスを送信します。どのデバイスも、もしそのデバイスのアドレスが指定された場合は、これを「Listen」します。このアドレス内のあるビットにより、マスターがスレーブデバイスから読み出すのか、それともスレーブデバイスに書き込むかが決められます。マスターとスレーブは、データ送信中、常に逆のモード（送信 / 受信）で作動します。すなわち、マスターとスレーブのモードの関係は以下の 2 種類のいずれかということになります。

- マスターが送信モード、スレーブが受信モード
- スレーブが送信モード、マスターが受信モード

いずれの場合も、マスターによりクロック信号が発せられます。

クロック (SCL) およびデータ (SDA) ラインの出力は、バスのワイヤード AND 機能を実行するためには、オープン・ドレインまたはオープン・コレクタである必要があります。デバイスが信号線を Low にドライブしていない時は、バスが High レベルとなるように、バスのプルアップ抵抗を接続します。I<sup>2</sup>C バスに接続できるデバイスの数は、負荷容量が 400pF 以下という仕様と、アドレス数だけで制限されます。

### A.1 データ転送の開始および終了

データ転送のない時間（アイドルタイム）には、クロックライン（SCL）およびデータライン（SDA）の両方が外部プルアップレジスタにより High レベルに維持されます。START および STOP 条件により、データ送信の開始と停止が決定されます。START 条件は、SCL が High の間の場合 SDA の High から Low への遷移と定義されます。STOP 条件は、SCL が High の間の SDA の Low から High への遷移と定義されます。マスターは、データ転送の開始および終了にあたりこれらの条件を発生させます。START および STOP 条件を図 A-1 に示します。この START と STOP の定義により、データ送信中、SCL ラインが Low の間にのみ、SDA ラインは状態を変化させることが可能です。

図 A-1: スタートおよびストップ条件

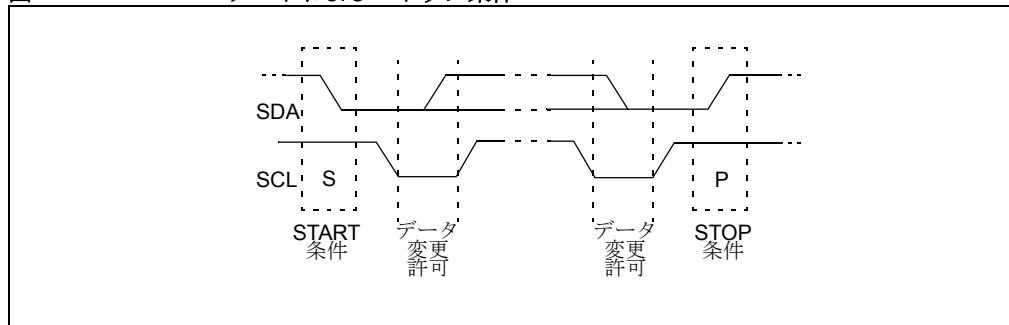


表 A-1:I<sup>2</sup>C バスに関連する用語

Term	Description
トランシミッタ	データをバスに送信するデバイス
レシーバ	データをバスから受信するデバイス
マスター	転送開始、クロック発生、転送終了を行うデバイス
スレーブ	マスターによりアドレスで呼び出されるデバイス
マルチマスター	1 つのシステムに 2 つ以上のマスターデバイスが存在すること。これらマスターは、メッセージを破損せずに同時にバスの制御を試みることが可能です。
アービトレー ション	マスターデバイスのうち 1 つのみがバスを制御するようにする手順。アービトレーションにより、転送データの破損を防げます。
同期化	2 つ以上のデバイスのクロック信号を同期動作させる手順。

## A.2 I<sup>2</sup>C デバイスのアドレスを指定する

アドレスフォーマットには 2 種類あります。最も簡単なのは、R/W ビットを含む 7 ビットアドレスフォーマット(図 A-2)です。より複雑なのは、R/W ビット(図 A-3)を含む 10 ビットのアドレスです。10 ビットのアドレスフォーマットの場合、2 バイトが送信される必要があります。最初の 5 ビットにより、10 ビットのアドレスフォーマットであることが指定されます。最初に送信されるバイトは、10 ビットアドレスモードを指定する 5 ビットと、アドレスのうち上位 2 ビット、および R/W ビットを指定する役割を含みます。次に送信される 1 バイトはアドレスのうち残り 8 ビットです。

図 A-2: 7 ビットアドレスフォーマット

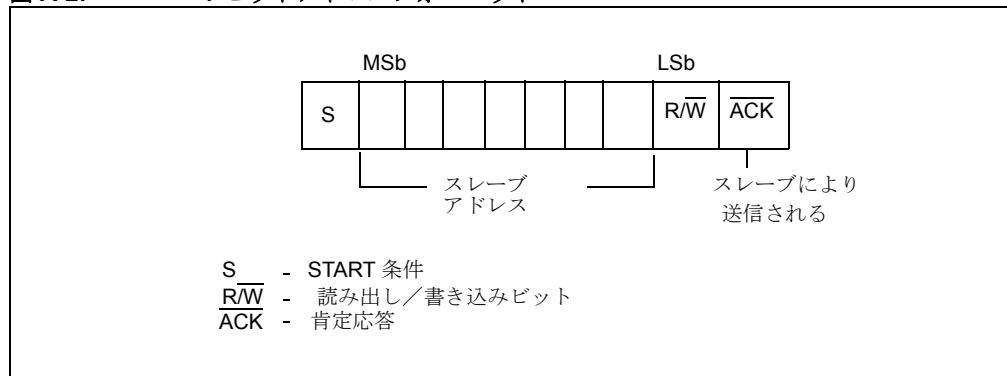
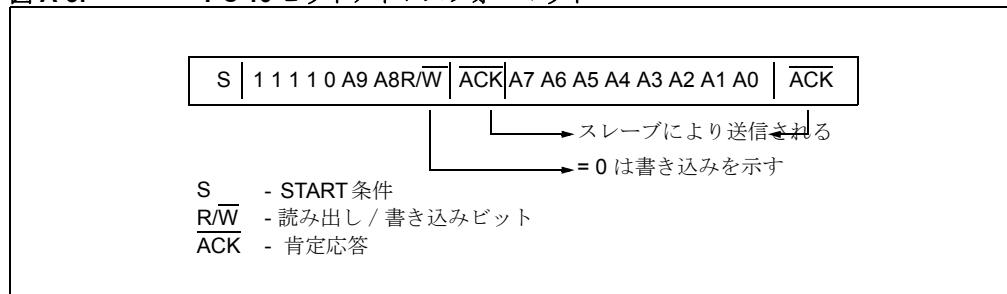


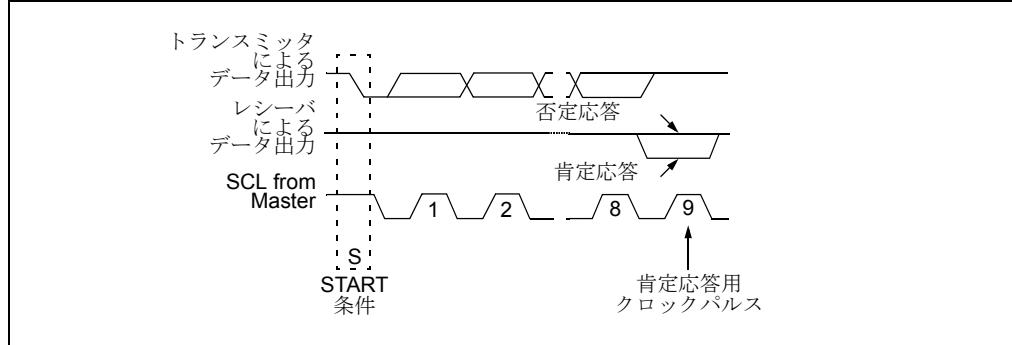
図 A-3: I<sup>2</sup>C 10 ビットアドレスフォーマット



## A.3 転送肯定応答

全てのデータはバイト単位で送信されなければなりませんが、データ転送ごとに送信されるバイト数には制限はありません。各バイト送信後、スレーブレシーバは肯定応答ビット(ACK)を発生させます(図 A-4)。スレーブレシーバがスレーブアドレスあるいは受信したデータに対して肯定応答を出さない場合、マスターは転送を中断する必要があります。スレーブは、マスターがSTOP条件を発することができるようSDAをHighにしておく必要があります(図 A-1)。

図 A-4: スレーブレシーバの肯定応答



マスターがデータを受信する場合(マスターレシーバ)、最後に受信するバイトを除き受信データの受信バイトごとに肯定応答信号を発します。データの終了をトランスマッタであるスレーブに伝達するためには、マスターは肯定応答を発しません(否定応答)。スレーブは、この時マスターがSTOP条件を生じさせられるようにSDAラインを解放します。マスターは、データ転送の終了を示すために肯定応答パルス中にSTOP条件を発することができます。

スレーブが次のバイト送信を遅らせる必要がある場合、SCLラインをLowに維持することによりマスターを待ち状態にさせます。スレーブがSCLラインを解放したときにデータ転送が継続します。これによりスレーブは、クロックを開始させる前に受信したデータを移動させたり、転送する必要のあるデータを取り出したりすることができます。この待ち状態技術は、図 A-5に示されているようにビット単位で実行することができます。

図 A-5: データ転送待ち状態

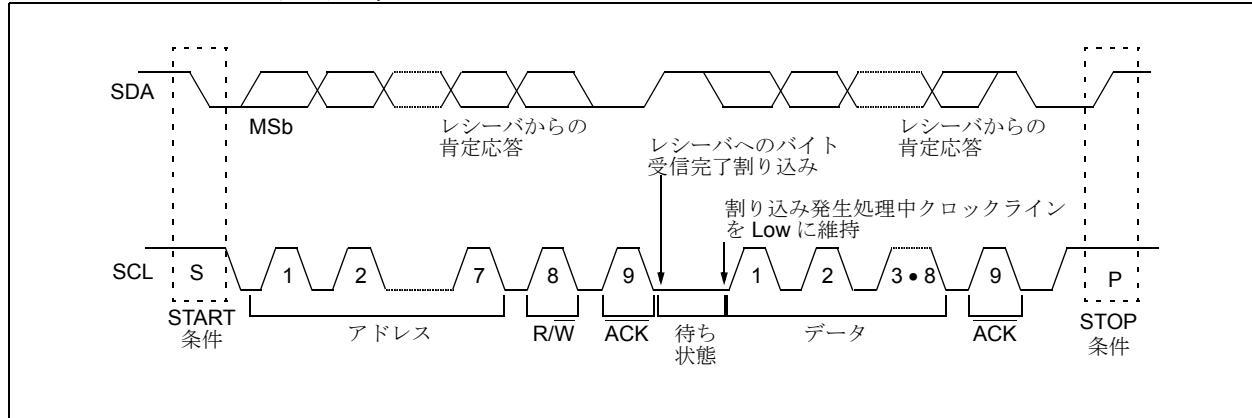


図 A-6 および図 A-7 は、マスター・トランシミッタおよびマスター・レシーバのデータ転送シーケンスを示したもので

図 A-6: マスタートランスマッタのシーケンス



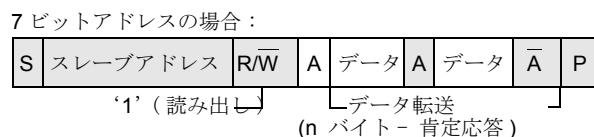
マスタートランスマッタが、7ビットのアドレスを持つスレーブレシーバのアドレスを指定。転送方向は変更なし。



マスタートランスマッタは**10**ビットのアドレスを持つ  
スレーブレシーバをアドレス指定。

	マスターからスレーブへ	A = 肯定応答 (SDA Low)
	スレーブからマスターへ	A = 肯定応答せず (SDA High)
		S = START 条件
		P = STOP 条件

図 A-7: マスター・レシーバのシーケンス



マスターが、最初のバイト後すぐにスレーブを読み出し。

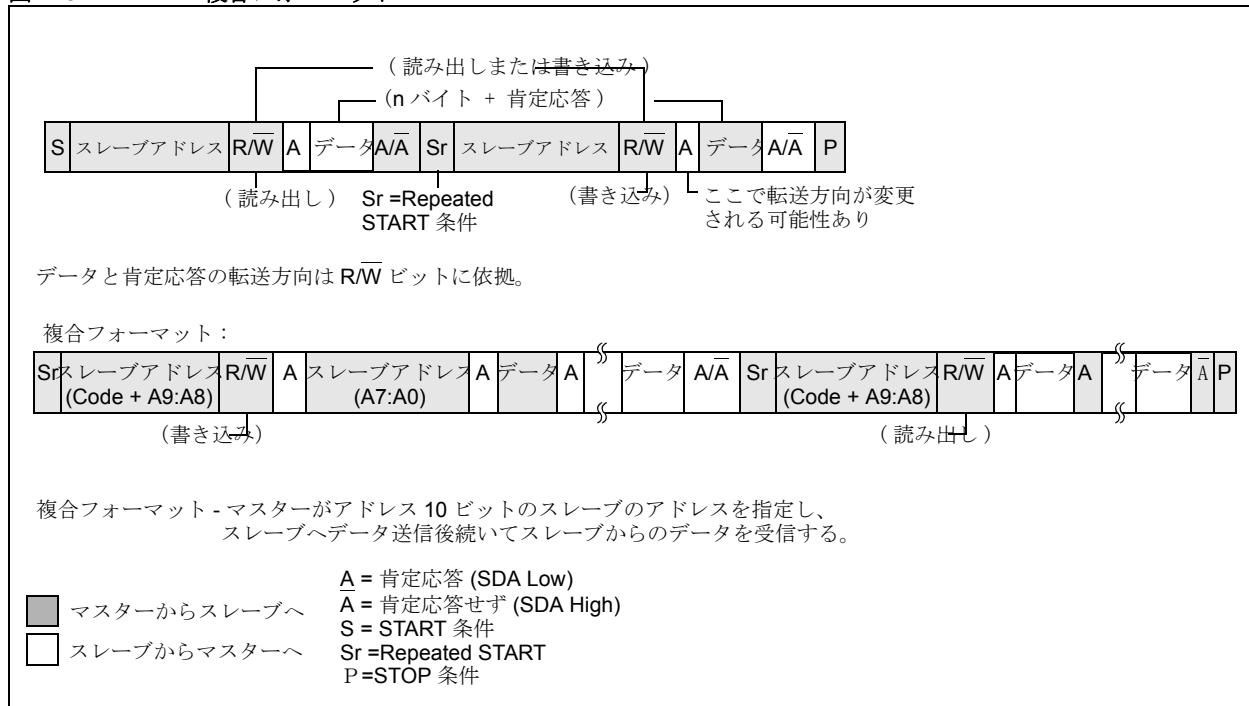


マスターransミッタがアドレス10ビットの  
スレーブレシーバをアドレス指定。

A = 肯定応答 (SDA Low)  
A = 肯定応答せず (SDA High)  
S = START 条件  
Sr= Repeated START 条件  
P=STOP 条件

マスターがバスを放棄 (STOP 条件を生成させることで発生) したくない場合、Repeated START 条件 (Sr) を生成する必要があります。この条件はスタート条件と同一ですが (SCL が High 中の SDA の High から Low への遷移)、データ転送肯定応答ビット (バス解放状態ではない) 後に発生します。この仕組みによりマスターはスレーブに対し「命令」を発することができ、要求した情報を受信したり、異なるスレーブデバイスのアドレスを指定することができます。このシーケンスは図 A-8 に示されています。

図 A-8: 複合フォーマット



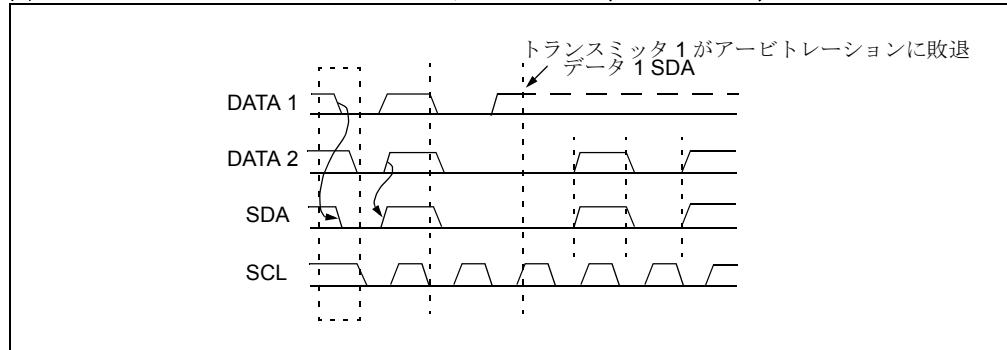
## A.4 マルチマスター

I<sup>2</sup>C プロトコルにより、システムが 2 つ以上のマスターを持つことが可能になります。これをマルチマスターシステムと呼びます。2 つ以上のマスターが同時にデータを転送しようとした場合、アービトレーションおよび同期化が行われます。

### A.4.1 アービトレーション

アービトレーションは、SCL ラインが High の場合に SDA ラインで生じます。ほかのマスターが SDA に Low を出力しているときに、High を出力しようとするマスターはアービトレーションで敗退し（図 A-9）、データ出力ステージを中止させます。アービトレーションで敗退したマスターは、アービトレーションで敗退したデータバイトの終了までクロックパルスを発生させることができます。マスターデバイスが同一のデバイスのアドレス指定をする際、アービトレーションはデータまで続けられます。

図 A-9: マルチマスターアービトレーション（マスター 2 つ）



スレーブ機能を組み込んでいて、アービトレーションで敗退したマスターは直ちにスレーブレスモードに切り替える必要があります。その理由は、勝利したマスタートランスマッタがアドレス指定を行う可能性があるからです。

以下の間にはアービトレーションは起こり得ません。

- Repeated START 条件
- STOP 条件およびデータビット
- Repeated START 条件および STOP 条件

これらの条件が発生しないように注意を払う必要があります。

#### A.4.2 クロック同期化

クロック同期化は、デバイスがアビトレーション開始後に発生します。これは SCL ラインがワイヤード AND 接続となっていることを利用しています。SCL ラインで High から Low への遷移が生じることで、接続されたデバイスが Low にある時間をカウント開始します。デバイスクロックが Low になると、SCL が High の状態になるまで SCL ラインは Low に保たれます。このデバイスクロックが Low から High に遷移しても、他のデバイスクロックがまだ Low にある場合 SCL ラインの状態は変化しない場合もあります。SCL ラインは、Low の状態が最も長時間継続するデバイスにより Low に保たれます。Low 状態の継続時間がより短いデバイスの場合、SCL ラインが High になるまでに High の待ち状態に入ります。SCL ラインが High になると、全デバイスは High 状態にある時間をカウント開始します。High の時間をまず先に終了したデバイスにより、SCL ラインが Low 状態に移行します。SCL ラインが High の時間は、High の状態にある時間が最も短いデバイスにより決定されます。図 A-10。

図 A-10: クロック同期化

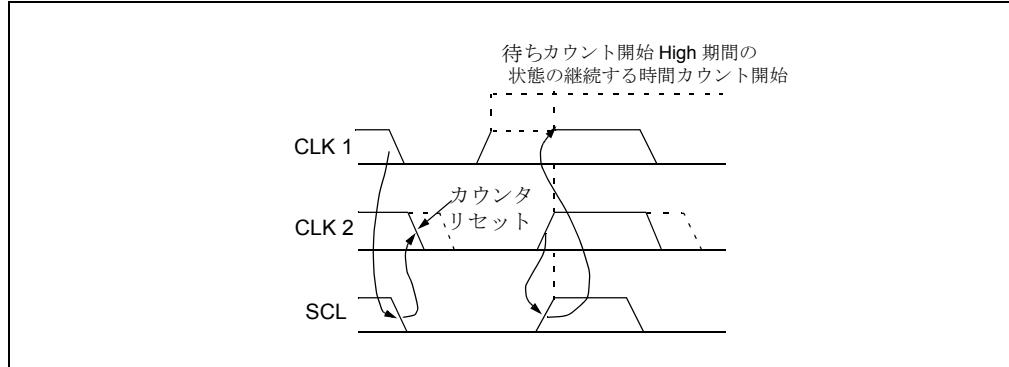


表 A-2 および表 A-3 は、I<sup>2</sup>C バスの規格を示しています。パラメータ番号と記された欄は、ユーザーがデバイスデータシート上で対応するパラメータを見つけることを容易にするために用意されたもので、図 A-11 と図 A-12 は、適切な波形でこれらの時間を示したものです。

図 A-11: I<sup>2</sup>C バス START/STOP ビットタイミング指定

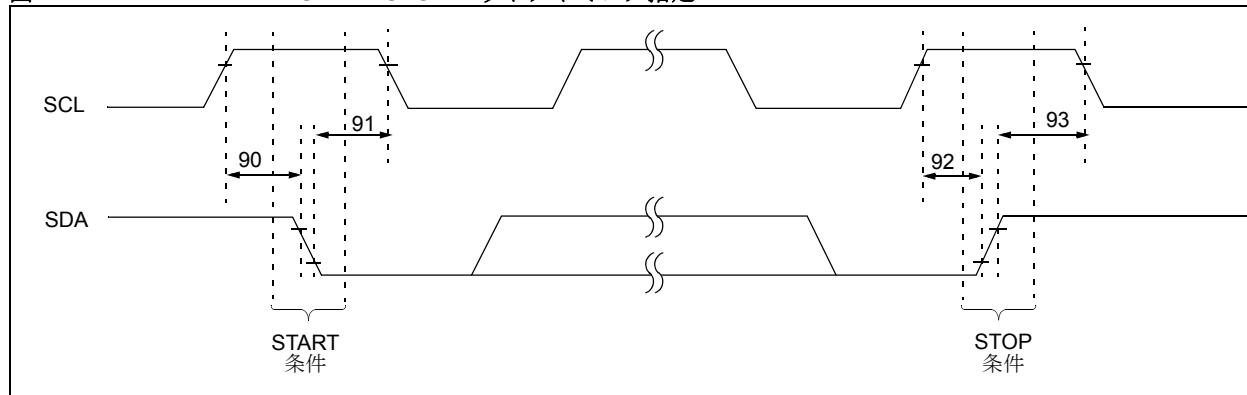


表 A-2: I<sup>2</sup>C バス START/STOP ビットタイミング指定

パラメータ No.	Sym	特徴	最低	Typ	最大	ユニット	条件
90	TSU:STA	START 条件	100 kHz モード	4700	—	—	Repeated START 条件にのみ有効
		セットアップタイム	400 kHz モード	600	—	—	
91	THD:STA	START 条件	100 kHz モード	4000	—	—	この期間後、最初のクロックパルスが生成
		一時保持時間	400 kHz モード	600	—	—	
92	TSU:STO	STOP 条件	100 kHz モード	4700	—	—	Repeated STOP 条件にのみ有効
		セットアップタイム	400 kHz モード	600	—	—	
93	THD:STO	STOP 条件	100 kHz モード	4000	—	—	この期間後、最初のクロックパルスが生成
		一時保持時間	400 kHz モード	600	—	—	

図 A-12: I<sup>2</sup>C バスデータタイミング指定

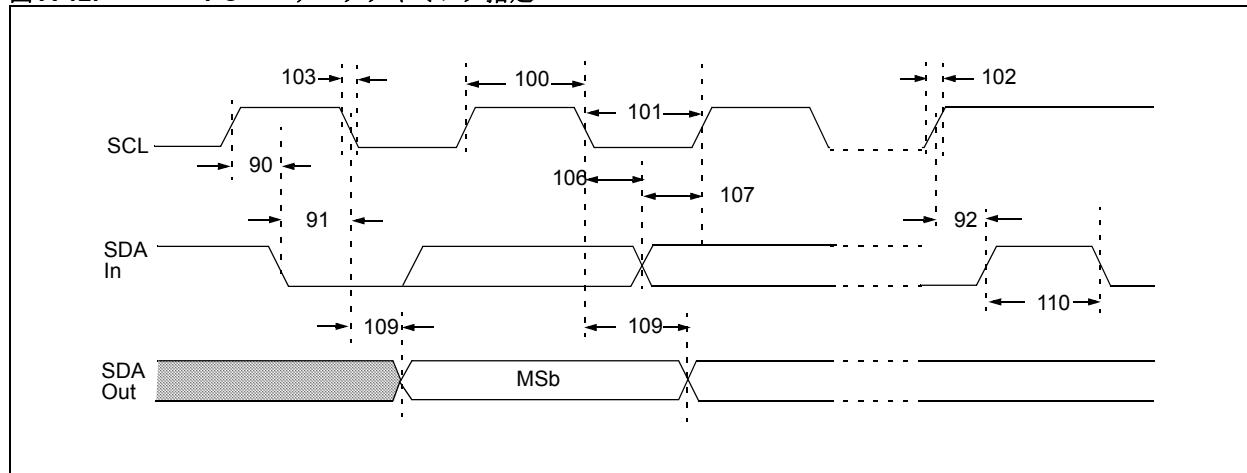


表 A-3: I<sup>2</sup>C バスデータタイミング仕様

パラメータ No.	Sym	特徴		最低	最大	ユニット	条件
100	THIGH	クロック High タイム	100 kHz モード	4.0	—	μs	
			400 kHz モード	0.6	—	μs	
101	TLOW	クロック Low タイム	100 kHz モード	4.7	—	μs	
			400 kHz モード	1.3	—	μs	
102	TR	SDA および SCL 立ち上がり時間	100 kHz モード	—	1000	ns	
			400 kHz モード	20 + 0.1Cb	300	ns	Cb は、10 から 400pF になるよう規定
103	TF	SDA および SCL 立ち下がり時間	100 kHz モード	—	300	ns	
			400 kHz モード	20 + 0.1Cb	300	ns	Cb は、10 から 400pF になるよう規定
90	TSU:STA	START 条件 セットアップタイム	100 kHz モード	4.7	—	μs	反復 START 条件の場合にのみ適切
			400 kHz モード	0.6	—	μs	
91	THD:STA	START 条件 一時保持時間	100 kHz モード	4.0	—	μs	この期間後、最初のクロックパルスが生成
			400 kHz モード	0.6	—	μs	
106	THD:DAT	データ入力一時保持時間	100 kHz モード	0	—	ns	
			400 kHz モード	0	0.9	μs	
107	TSU:DAT	データ入力セットアップ時間	100 kHz モード	250	—	ns	注 2
			400 kHz モード	100	—	ns	
92	TSU:STO	STOP 条件 セットアップ時間	100 kHz モード	4.7	—	μs	
			400 kHz モード	0.6	—	μs	
109	TAA	出力がクロックから有効になるまで	100 kHz モード	—	3500	ns	注 1
			400 kHz モード	—	1000	ns	
110	TBUF	バスフリータイム	100 kHz モード	4.7	—	μs	新たな送信が始まる前に、バスが空く必要のある時間
			400 kHz モード	1.3	—	μs	
D102	Cb	バス負荷容量			400	pF	

注 1: トランスマッタとして、デバイスは START あるいは STOP 条件が予期せず生成されることを防ぐために、SCL の立ち下がりエッジの未定義領域を回避するようにこの内部最低遅延時間を考慮する必要があります（最低 300ns）。

- 2: 高速モード I<sup>2</sup>C バスデバイスは、標準モード I<sup>2</sup>C バスシステムで使用できますが、 $T_{SU: DAT} \geq 250 \text{ ns}$  という必要条件が満たされる必要があります。デバイスが SCL 信号の Low 期間を延長することがなければ自動的にこの条件は満たされます。そのようなデバイスが SCL 信号の Low 期間を延長させる場合は、SCL ラインがリリースされる前に、 $TR_{max} + T_{SU: DAT} = 1000 + 250 = 1250 \text{ ns}$ （標準モード I<sup>2</sup>C バス仕様により）の時間以上のデータビットを SDA ラインに出力させる必要があります。

## 付録 B: CAN 概要

本付録では、コントローラエリアネットワーク (CAN) バスの概要を示します。本リファレンスマニュアルの CAN に関する項では、内蔵モジュールへの CAN プロトコルの実装について検討します。

### B.1 CAN バスのバックグラウンド

コントローラ・エリア・ネットワーク (CAN) は、シリアル通信プロトコルであり、非常に高水準の安全性によりリアルタイムの分散制御を効果的にサポートします。

高速のネットワークから、低価格の多重配線に至るまで幅広い範囲に利用可能です。自動車の電子工学では、エンジンの制御ユニット、センサー、アンチスキッドシステムなどが、速度最大 1 メガビット / 秒の CAN を用いて接続されています。シリコンはまた低コストであるため、自動車のワイヤリングハーネスの代わりの役割を果たしています。ノイズの大きい環境におけるバスの頑丈さ、およびフォールト条件を検出し、そこから復帰する能力が優れているため、バスはデバイスネット、SDS その他のフィールドバスプロトコルといった産業用制御アプリケーションに適しています。

CAN は非同期シリアルバスシステムで、一本の論理的バスラインで構成されています。CAN は開放型で線状のバス構造となっており、バスノードは等しくなっています。CAN バスは 2 つ以上のノードにより構成されています。バス上のノードの数は、他のノードの通信を阻害することなく大幅に変更することができます。このためバスノードの接続および切断が容易となっています（例えば、システム機能の追加、エラー回復あるいはバスのモニタリングなど）。

バスロジックは「wired-AND」機構と対応しており、「リセッショブ」ビット（大部分はロジックレベル “1” と同等ですが、必ずしも同等とは限りません）は「ドミナント」ビット（大部分はロジックレベルが “0”）により上書きされます。ドミナントビットを送信するバスノードがない限り、バスラインはリセッショブ状態ですが、いずれかのバスノードからドミナントビットが送信された場合バスはドミナント状態になります。それゆえに、CAN バスラインには、可能な 2 つのビット状態（ドミナント状態およびリセッショブ状態）を伝送可能な媒体が選ばれる必要があります。最も一般的で安価な方法の 1 つは、対より線を用いることです。この場合バスラインは “CANH” および “CANL” と呼ばれ、直接ノードに接続したりコネクタ経由で接続することが可能です。使用するコネクタについては、CAN で定められた標準というものは存在しません。バスラインの各末端で抵抗で終端させることにより、対より線が終端します。バス速度は最大 1 メガビットで、バスの長さが最大 40 メートルまでこの速度が実現できます。バスの長さが 40 メートル以上の場合は、バスの速度は減速します（1000 メートルの場合、バス速度は 40 キロビットになります。）バスの長さが 1000 メートル超の場合、特殊なドライバを使用する必要があります。設備を追加することなく最低 20 のノードを接続することができます。伝送の差動的な性質により、CAN は EMI に対し反応しません。なぜなら、両方のバスラインが同じように作用を受け、差動信号が作用を受けないからです。バスラインはまた、特にボーレートの数値が高い場合、バス自体の電磁放射を減少させるようシールドします。

バイナリデータは、NRZ コードに従い符号化されます（非ゼロ復帰；低レベル = ドミナント状態、高レベル = リセッショブ状態）。全てのバスノードの正確な同期化を保障するために、ビットスタッフィングが用いられています。すなわち、メッセージの送信中、最大 5 連続のビットが同一の極性を有することがあることを意味しています。同一の極性を持つ 5 つの連続するビットが送信されたときはいつでも、トランスマッタはさらにビットを送信するよりも前に、反対の極性を持つ別のビットをビットストリームに組み込みます。レシーバはまた、同一の極性を持つビットの数を確認し、ビットストリームからスタッフィングを除去します（デスタッフィング）。

CAN プロトコルでは、バスノードはアドレス指定されませんが、アドレス情報は送信されるメッセージに含まれています。このことはメッセージの内容（例 エンジンのスピード、油温など）を特定する識別子（各メッセージの一部）を通して行われます。識別子はさらに、メッセージの優先順位を示します。識別子のバイナリ値が低いほど、メッセージの優先度は高くなります。

バスのアービトレーションには、CSMA/CD (Carrier Sense Multiple Access/Collision) が用いられ NDA(Non-Destructive Arbitration) が行われます。バスノード A がネットワークを通してメッセージを送信したい場合、まずバスがアイドル状態（すなわち、現在どのノードも送信を行っていないこと）にあることを確認します（“キャリア検知”）。バスがアイドル状態にある場合（かつ、他のノードが同時に送信を開始しようとしていない場合）、ノード A はバスマスターになり、メッセージを送信します。他の全てのノードは最初のビット送信（フレーム開始ビット）の間受信モードに切り替わります。メッセージを正常に受信した後（各ノードにより認証される）、各バスノードはメッセージの識別子を確認し、必要な場合メッセージを保存します。その他の場合、メッセージは破棄されます。

2つ以上のノードが同時に送信を開始した場合 (“Multiple Access”)、メッセージの衝突はビットごとのアービトレイションにより回避されます (“Collision Detection/Non-Destructive Arbitration” と、“Wired AND” メカニズムが合わさり、“ドミナント”なビットが“リセシップ”なビットに優越します)。各ノードはメッセージ識別子のビットを送信し(最上位ビットがまず先)、バスのレベルを監視します。リセシップな識別ビットを送信するものの、ドミナントビットをリードバックするノードはバスアービトレイションで敗退し、受信モードに切り替わります。競合するノードのメッセージ識別子のバイナリ値が低く(ドミナント状態 = ロジック0)すなわち競合するノードが優先順位の高いメッセージを送信する場合にこの条件が発生します。こうして、メッセージの優先順位の最も高いバスノードがアービトレイションに勝利し、メッセージを反復することで時間を無駄にすることもありません。その他全てのノードは、バスが IDLE 状態に戻ると自動的に送信反復を試みるようになります。アービトレイションに失敗し、衝突とエラーにつながるため、異なるノードが、同一の識別子でメッセージを送信することは認められていません。

当初の CAN の仕様 (バージョン 1.0、1.2 および 2.0A) では、メッセージ識別子は 11 ビットの長さを持ち、2048 のメッセージ識別子が使用可能でした。この仕様は後に更新され (バージョン 2.0B)、識別子の数という制限をなくしました。CAN のバージョン 2.0B での仕様では、11 ビットと 29 ビットのいずれか、または両方の長さのメッセージ識別子を使用可能です (長さ 29 ビットの識別子では、5 億 3600 万以上のメッセージ識別子を利用することが可能になります)。CAN のバージョン 2.0B はまた、 $\phi$  拡張 CAN $\epsilon$  と呼ばれ、バージョン 1.0、1.2 および 2.0A は、 $\phi$  標準 CAN $\epsilon$  と呼ばれています。

## B.2 異なる CAN の実装方法

### B.2.1 標準 CAN、拡張 CAN

11 ビットの識別子のみが含まれるデータフレームおよびリモートフレームは、標準フレームと呼ばれ、CAN バージョン 2.0A の仕様に準拠しています。これらのフレームでは、2048 の異なるメッセージが識別できます (識別子 0-2047)。しかし、優先順位の最も低い 16 の識別子 (2032-2047) は保留されています。CAN のバージョン 2.0B の仕様に準拠する拡張フレームは、29 ビットの識別子が用いられています。すでに述べたように、この 29 ビットの識別子は 11 ビットの識別子 (“ベース ID”) と 18 ビットの拡張識別子 (“ID 拡張”) により構成されています。

CAN のバージョン 2.0A が規定する CAN モジュールは、標準 CAN プロトコルに従い標準フレームのみを送受信可能です。29 ビットの識別子を用いるメッセージは、エラーを発生させます。デバイスに CAN バージョン 2.0B が用いられている場合、1 つ別の特徴があります。 $\phi$  パート B パッシブ $\epsilon$  と名づけられたモジュールは、標準フレームのみ送受信可能ですが、拡張フレームの場合もエラーフレームを発生させず許容します。 $\phi$  パート B アクティブ $\epsilon$  のデバイスは、標準フレームと拡張フレームの両方を送受信可能です。

### B.3 ベーシック CAN、フル CAN

CAN モジュールとホスト CPU の間のインターフェースに関してはもう 1 つ別の CAN の特性があり、CAN のチップが  $\phi$  ベーシック CAN $\epsilon$  と「フル CAN」デバイスに分類されます。しかしこれは使用されるプロトコル (標準 CAN または拡張 CAN) とは関係ないため、同一のネットワークでベーシック CAN デバイスとフル CAN デバイスの両方が使用可能です。

ベーシック CAN デバイスでは、ハードウェアではプロトコルの基本的機能のみが実行可能です (例、ビットストリームの生成およびチェック)。受信したメッセージを保存するべきかどうかの決断 (アクセプタンスフィルタリング) およびメッセージ管理の全体は、ソフトウェア (すなわちホスト CPU) により行われる必要があります。多くの場合、CAN チップには送信バッファは 1 つのみ、受信バッファは 1 つまたは 2 つしかありません。従ってベーシック CAN モジュールを使用する CPU の負荷はきわめて高いため、このようなデバイスは低いポートレートで使用する必要があります、また異なるメッセージはごく少数としてバス負荷を小さくして用いる必要があります。ベーシック CAN の利点は、チップのサイズが小さく、デバイスが低コストとなることです。

多くの場合、フル CAN デバイスは、アクセプタンスフィルタリングとメッセージマネジメントも含めてハードウェアによってバスプロトコル全体を実行します。フル CAN デバイスには複数のいわゆるメッセージオブジェクトが含まれ、識別子、データ、方向 (受信か送信か)、および標準 CAN/拡張 CAN の情報が扱われています。デバイス初期化の間に、ホスト CPU は、どのメッセージを送信・受信するか決定します。ホスト CPU は、受信したメッセージの識別子が、プログラムされた (受信) メッセージオブジェクトと一致する場合割り込みにより知らされます。こうして CPU 負荷が軽減されます。フル CAN デバイスを用いることで、メッセージ数が多くボードレートやバスロードが高い場合にも対処可能となります。しかし、このようなチップはベーシック CAN デバイスに比べて高価となります。

多くのフル CAN チップには「標準 CAN 機能」が備えられています。そのメッセージオブジェクトの 1 つに対するプログラムにより、他のメッセージオブジェクトと一致しない全てのメッセージが記憶されるようになっています。数多くのアプリケーションにおいて、この機能は非常に有用です。

## B.4 ISO モデル

ISO/OSI リファレンスモデルは、図 B-1 で示されるような通信システムのプロトコルのレイヤーを定義するために用いられます。最高位レベルでは、アプリケーションは相互間で通信する必要があります。最低位レベルでは、電子シグナリングを発するためにある種の物理的媒体が用いられます。

高水準のプロトコルはソフトウェアにより実行されます。通常、アプリケーションのレイヤーのみが実行されます。CAN のバス仕様内では、メッセージの種類や転送されるメッセージの内容あるいは意味に関する定義は行われません。こうした定義は、例えば Volcano と呼ばれるボルボ自動車の CAN 仕様 ; J1939 、アメリカ合衆国大型トラックマルチプレックスワイアリングスペック ; アレンブラッドリーのデバイスネットやハネウェル SDS といった産業用プロトコルで行われます。

CAN バスのモジュール定義は、全体で 2 つのレイヤのプロトコルを包含しています。

- データリンクレイヤ
  - ロジカルリンク制御 (LLC) サブレイヤ
  - メディアアクセスコントロール (MAC) サブレイヤ
- - 物理的レイヤ
  - 物理信号制御 (PLS) サブレイヤ

LLC サブレイヤは、メッセージフィルタリング、オーバーロード通知、エラー修復マネジメントと関係しています。LLC サブレイヤの役割は以下の通りです。

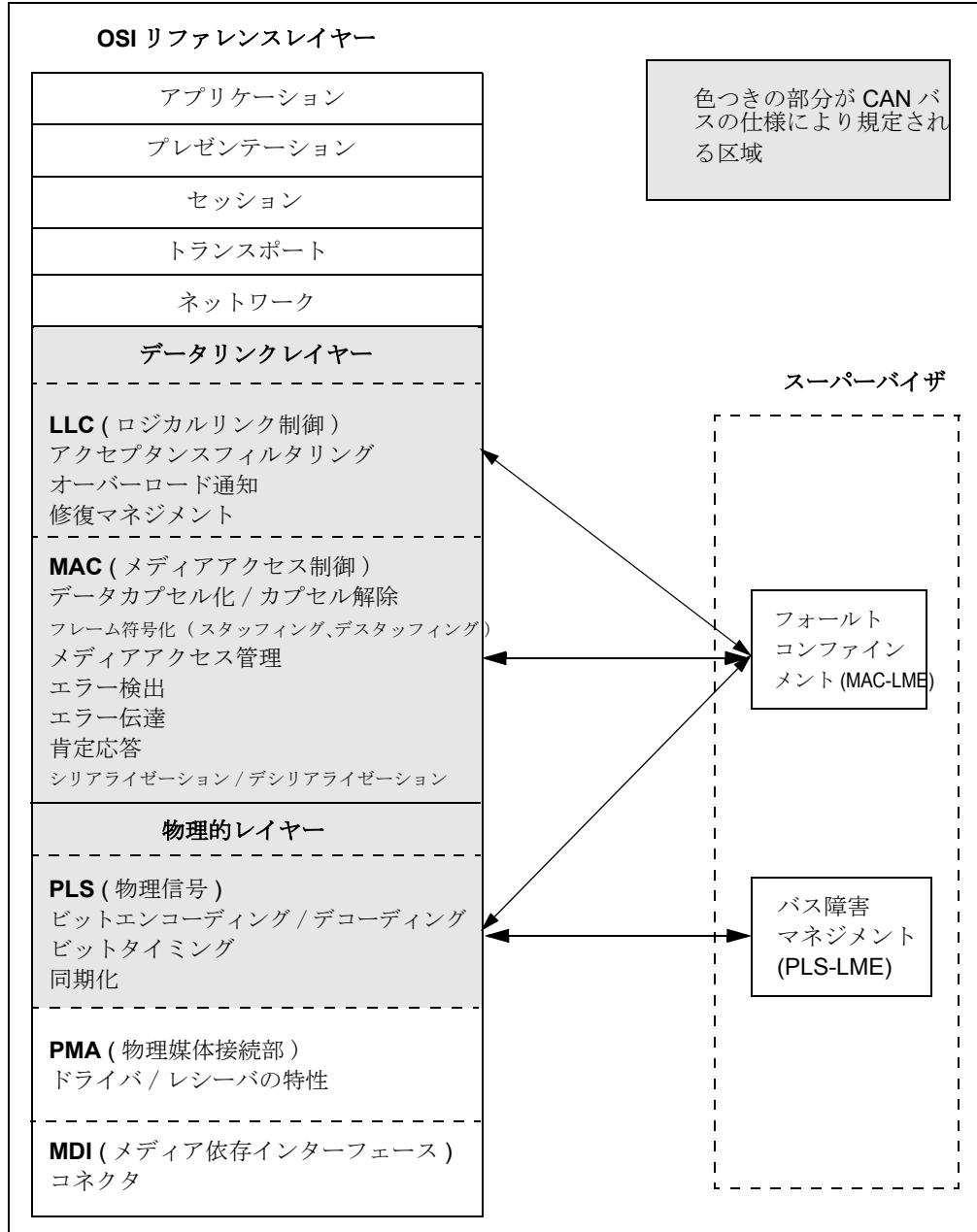
- データ送信および遠隔データリクエストにあたってサービスを提供する
- LLC サブレイヤの受信したメッセージのうち、実際にどれを受容できるか決定する
- エラー修復マネジメントおよびオーバーロード通知の手段を提供する

MAC サブレイヤは CAN プロトコルのいわば核といえるものです。MAC サブレイヤは、転送プロトコルを定義します (例、フレーム指示制御、アビトレーション実施、エラーチェック、エラー伝達およびフォールト制限など)。MAC サブレイヤは LLC サブレイヤから受信したメッセージを提示し、メッセージの LLC サブレイヤへの送信を認めます。MAC サブレイヤ内では、バスが新たな送信を行う準備ができているか、それとも受信をすぐに始めるかが決定されます。MAC サブレイヤは、フォールトコンファインメントと呼ばれるマネジメントエンティティにより管理されます。マネジメントエンティティとは、短期的な障害と恒常的な故障を区別するための自己チェック機構です。また、ビットタイミングの一般的な機能の一部は MAC サブレイヤの一部と考えられています。

物理的レイヤにより、全ての電気的特性に関して様々なノード間の実際のビット送信が決定されます。PLS サブレイヤにより実際の信号送信が決定されるため、PLS サブレイヤがビットタイミング、ビットエンコーディングおよび同期化を扱うといえます。

低レベルのプロトコルは、ドライバ / レシーバチップや、対より線や光ファイバーなどのインターフェースにおいて実装されます。1 つのネットワーク内では、物理的レイヤーは全てのノードに対し同一である必要があります。物理的レイヤーのドライバおよびレシーバとしての特性は、送信媒体およびシグナルレベル実装がアプリケーション向けに最適化されるようにするために、定義されません。最も一般的な例は、ISO11898 自動車複合ワイアリング規定に定義されています。

図 B-1: ISO/OSI リファレンスモデルにおける CAN バス



## B.5 CAN バスの特徴

CAN は以下のようなプロパティを有しています。

- メッセージの優先順位決定
- 待ち時間確保
- 柔軟な設定
- 時間の同期化を伴うマルチキャスト受信
- システム全体に及ぶデータ一貫性
- マルチマスター
- エラー検出と伝達
- 不正なメッセージの自動再送信
- ノードの一時的エラーと恒久的障害の区別、および障害のあるノードの自動除去
- 1. メッセージ - バス上の情報は、長さは多様だが制限のある決まったフォーマットのメッセージの形にされ送信されます。バスが空いている場合、接続されているどのユニットも新たなメッセージを送信開始できます。
- 2. 情報ルーチン - CAN システムでは、CAN ノードはシステム設定（ステーションアドレスなど）に関するいかなる情報も利用しません。
- 3. システムの柔軟性 - ノードが CAN ネットワークに追加される際、ノードのソフトウェアまたはハードウェア、およびアプリケーションレイヤーに変更も必要なしに行なうことが可能です。
- 4. メッセージルーチン - メッセージの内容は識別子により指定されます。識別子はメッセージの方向は示しませんが、データの意味を記述するため、ネットワーク中の全てのノードは、データをノードが処理するかどうかをメッセージフィルタリングにより決定することができます。
- 5. マルチキャスト - メッセージフィルタリングのコンセプトゆえに、どんな数のノードも同一のメッセージを受信し、同時に処理することができます。
- 6. データの一貫性 - CAN ネットワーク内では、あるメッセージが同時に全てのノードにより受容されるか、またはいずれのノードも受容しないようになっています。したがって、マルチキャストの概念およびエラー処理により、システムデータの一貫性が保障されます。
- 7. ビット速度 - CAN の速度はシステムにより異なります。しかし、同じシステムでは、ビットの速度は均一で固定しています。
- 8. 優先順位決定 - 識別子により、バスアクセス中の静的メッセージ優先順位が決定されます。
- 9. リモートデータリクエスト - リモートフレームを送信することにより、データを必要とするノードは、別のノードに対し対応するデータフレームを送信するように要求することができます。データフレームおよび対応するリモートフレームは、同一の識別子により指定されます。
- 10. マルチマスター - バスが空いている時は、どのユニットもメッセージ送信を開始することができます。送信にあたり優先順位の高いメッセージを有するユニットがバスアクセスを得ます。
- 11. アービトレーション - バスが空いているときは、どのユニットもメッセージ送信を開始することができます。2つ以上のユニットが同時にメッセージ送信を開始しようとしているときは、バスアクセスをめぐる衝突は識別子を利用したビット単位のアービトレーションにより解決されます。アービトレーションの仕組みにより、情報も時間も失われないようになります。同一の識別子を持つデータフレームとリモートフレームが同時に送信開始されたときは、データフレームがリモートフレームより優先されます。アービトレーション中、全てのトランスマッタは送信されるビットのレベルを、バス上でモニタされているレベルと比較します。この 2つのレベルが対等である場合、ユニットは送信を続けることができます。\$ リセシプト \$ レベルが送信され、\$ ドミナント \$ レベルがモニタされる場合は、ユニットはアービトレーションに敗退したことになり、これ以上ビットを送信せず引き下がることになります。
- 12. 安全性 - データ送信の安全性を最大限に確保するために、全ての CAN ノードでエラー検出、送信および自己チェックを行う強力な対策が実施されています。
- 13. エラー検出 - エラー検出に関しては、以下の対策が取られています。
  - モニタリング (トランスマッタが、送信されるビットのレベルを、バスで検出されるビットレベルと比較すること)
  - CRC チェック
  - ビットスタッフィング
  - メッセージフレームチェック

エラー検出メカニズムは次のような特性を有しています。

- 全体的エラーは全て検出されます。
  - トランスマッタにおける局所的エラーは全て検出されます。
  - メッセージ中、最大 5 つのランダムに分布したエラーが検出されます。
  - メッセージ中の、長さ 15 未満のバーストエラーは検出されます。
  - メッセージ中の奇数のエラーは検出されます。
14. エラー伝達および回復時間 - 不正なメッセージは、エラーを検出したあらゆるノードによりフラグされます。そのようなメッセージは破壊され、自動的に再送信されます。エラーを検出してから次のメッセージが開始するまでの回復時間は、他にエラーがない場合最大で 31 ビットタイムです。
  15. フォールトコンファインメント - CAN ノードにより、短期的な障害と恒久的な故障が区別されます。欠陥のあるノードは除去されます。
  16. 接続 - CAN シリアル通信リンクとは、複数のユニットが接続可能なバスのことです。接続可能な数には理論上制限はありません。実際には、ユニットの総数はバスライン上の遅延時間や電気負荷により制限されます。
  17. 単一チャンネル - バスはビットを運搬する单一チャンネルにより構成されています。このデータ再同期化により情報が得られます。チャンネルの実装方法は、この仕様により固定されていません(例 単線(プラスアース)、2 つの差動ワイヤ、光学ファイバなど)。
  18. バス値 - バスは、相補的な 2 つの論理値のいずれかを有しています。すなわち  $\phi$  ドミナント  $\&$  値と  $\phi$  リセシプ  $\&$  値です。 $\phi$  ドミナント  $\&$  ビットと  $\phi$  リセシプ  $\&$  ビットが同時に送信された場合のバス値は  $\phi$  ドミナント  $\&$  になります。例えば、バスがワイヤード AND 実装された場合、 $\phi$  ドミナント  $\&$  レベルは論理 ‘0’ により示され、 $\phi$  リセシプ  $\&$  レベルは論理 ‘1’ により示されます。論理レベルを示す物理的状態(例、電圧、明るさなど)は、この仕様では表現されません。
  19. 承認 - 全レシーバは受信するメッセージの一貫性をチェックし、一貫したメッセージを承認し、一貫していないメッセージをフラグします。
  20. SLEEP モード；ウェイクアップ - システムの電力消費を減少させるために、CAN デバイスを SLEEP モードに設定し、内部動作を停止させ、バスドライバを切断することができます。SLEEP モードは、バスの動作あるいはシステム内部の条件によりウェイクアップすることで終了します。起動時には、内部の動作は再開しますが、MAC のサブレイヤはシステムの発振器が安定するまで待機し、発振器は(11 回連続で  $\phi$  リセシプ  $\&$  ビットをチェックすることにより)バスの動作に同期化するまで待機し、その後ようやくバスドライバが再び  $\phi$  オンバス  $\&$  にセットされます。

## B.6 フレームの種類

### B.6.1 標準データフレーム

ノードがデータを送信しようとするとき、ノードによりデータフレームが生成されます。標準 CAN データフレームは図 B-2 に示されています。他の全てのフレームと同様、このフレームもフレーム開始ビット(SOF、ドミナント状態)で開始され、全てのノードが厳密に同期されます。

SOF に続き、11 ビットの識別子(メッセージの内容と優先順位を反映する)と RTR ビット(リモート伝達リクエストビット)の合計 12 ビットから構成されるアービトレーションフィールドが使用されます。RTR ビットは、データフレーム(RTR ドミナント)をリモートフレームと区別するために用いられます。

次のフィールドは制御フィールドで、6 ビットで構成されます。このフィールドの最初のビットは IDE(識別子拡張)ビットと呼ばれ、ドミナント状態にあり、フレームが標準フレームであることを規定しています。その次のビットは予約ビット RB0 で、ドミナントビットと規定されています。制御フィールドの残り 4 ビットはデータ長コード(DLC)で、メッセージに含まれるデータのバイト数を規定します。

送信されるデータはデータフィールドに送られますが、このデータフィールドは上記の DLC で規定された長さ(1 – 8 バイト)となっています。

その先の巡回冗長検査(CRC)フィールドは、起き得る送信エラーを検出するために用いられます。CRC は 15 ビットの CRC シーケンスにより構成され、それを補うためにリセシプ CRC デリミタが用意されています。

最後のフィールドは ACK フィールドです。ACK スロットビットでは、送信ノードがリセシプビットを送信します。エラーのないフレームを受信したノードは、ドミナントビットを送り返すことによりフレームを正しく受信したことを確認します（ノードが、特定のメッセージを受理するように設定されているかどうかに関係なく）。このことから、CAN はプロトコルの  $\phi$  ビット内応答  $\epsilon$  グループに属することがわかります。リセシプ承認デリミタは承認スロットを補う役割を果たし、ドミナントビットによっても上書きされません。

## B.7 拡張データフレーム

図 B-3 に示されている拡張 CAN データフレームでは、フレーム開始ビット (SOF) に続き、38 ビットで構成されるアービトレーションフィールドが使用されます。最初の 11 ビットは、29 ビットの識別子（“ベース ID”）の中でも特に重要な 11 ビットです。これら 11 ビットに続き、代理リモートリクエストビット (SRR) が用いられます。SRR はリセシプビットとして送信されます。SRR の次に IDE ビットが用いられています。IDE ビットはリセシプビットで、フレームが拡張 CAN フレームであることを示します。このことから、識別子の最初の 11 ビット送信後アービトレーションが未解決のままで、アービトレーションに関与しているノードのうち 1 つが標準 CAN フレーム (11 ビット識別子) を送信中の場合、ドミナントの IDE ビットが宣言されるため標準 CAN フレームがアービトレーションに勝利することに留意が必要です。また、拡張 CAN フレームにおける SRR ビットは必ずリセシプになり、標準 CAN リモートフレームを送信中のノードによるドミナント RTR ビットの宣言が可能となります。SRR ビットと IDE ビットに続いて、残り 18 ビットの識別子（“ID 拡張”）とリモートトランスマッショングリエストビットが用いられます。

フレームおよび拡張フレームが、共有ネットワーク全体に送信されるようにするには、29 ビットの拡張メッセージ識別子を 11 ビット（最重要）と 18 ビット（重要度低）に分割する必要があります。この分割により、識別子拡張ビット (IDE) は、標準フレームにおいても拡張フレームにおいても、同じビットポジションにとどまることが可能となります。

続いて制御フィールドと呼ばれる 6 ビットのフィールドがあります。このフィールドの最初の 2 ビットは予約済みで、ドミナント状態にあります。制御フィールドの残り 4 ビットはデータ長コード (DLC) で、データバイト数を規定します。

フレームの残り部分（データフィールド、CRC フィールド、承認フィールド、フレーム終了および中断）は、標準データフレームの場合と同じように構築されています。

## B.8 リモートフレーム

通常データ転送は自動的にデータソースノード（例、データフレームを出すセンサー）を用いて行われます。しかし、宛先ノードがソースからデータを要求することもあります。このために、宛先ノードが、必要とするデータフレームの識別子と一致する識別子を有する  $\phi$  リモートフレーム  $\epsilon$  を送信します。すると、このリモートリクエストに対応するデータソースノードがデータフレームを送信します。

図 B-4 に示されているように、リモートフレームとデータフレームには 2 つの差異があります。最初に、RTR ビットはリセシプ状態にあり、第二に、データフィールドがありません。非常に稀なことですが、同じ識別子を持つデータフレームとリモートフレームが同時に送信された場合、データフレームは識別子に続くドミナントの RTR ビットによりアービトレーションに勝利します。こうして、リモートフレームを送信したノードが、要求しているデータを直ちに受信します。

## B.9 エラーフレーム

エラーフレームを検出したノードによりエラーフレームが生成されます。図 B-5 に示されるように、エラーフレームは 2 フィールドから構成され、エラーフラグフィールドにエラーデリミタフィールドが続きます。エラーデリミタは 8 つのリセシプビットから構成され、バスノードがエラー後にきちんとバス通信を再開できるようにする役割を持ちます。エラーフラグフィールドには 2 形態あります。エラーフィールドの形態は、エラーを検出するノードのエラーステータスによって決まります。

エラーアクティブノードがバスエラーを検出した場合、ノードはアクティブなエラーフラグを生成し、現在のメッセージ送信を中断します。アクティブエラーフラグは、6 つの連続するドミナントビットにより構成されています。このビット列は能動的にビットスタッフィングの規則に違反しています。他の全てのステーションは、結果として生じたビットスタッフィングエラーを認識し、今度は自らエラーエコーフラグと呼ばれるエラーフレームを生成します。そのため、エラーフラグフィールドは、6 から 12 の連続するドミナントビット（1 つまたは複数のノードにより生成）により構成されています。エラーデリミタフィールドはエラーフレームを補完するものです。エラーフレーム終了後、バスの動作は通常に戻り、中断されたノードは破棄されたメッセージを再び送信しようと試みます。

エラーパッシブノードによりバスエラーが検出された場合、ノードによりエラーパッシブフラグが送信され、続いて再びエラーデリミタフィールドが送信されます。エラーパッシブフラグは 6 連続のリセッショビットから構成されるため、エラーパッシブノードのエラーフラグは 14 のリセッショビットから構成されます。このことから、能動的に送信を行うバスマスターノードによりバスエラーが検出されない限り、エラーパッシブノードによるエラーフレームの送信は、ネットワーク上の他のノードに影響を及ぼしません。バスマスターノードがエラーパッシブフラグを生成する場合、この生成によりビットスタッフィング違反が生じるため他のノードによりエラーフレームが生成される場合があります。エラーフレーム送信後、エラーパッシブノードは、バス通信に再び加わるまでにバス上の 6 連続するリセッショビットを待つ必要があります。

#### B.10 オーバーロードフレーム

図 B-6 に示されるオーバーロードフレームは、アクティブエラーフレームと同一の形式を有しています。しかしオーバーロードフレームは、インターフレームスペースの間でのみ生成されます。この点で、オーバーロードフレームはエラーフレームと区別されます（エラーフレームは、メッセージ送信中に送信されます）。オーバーロードフレームは 2 フィールドから構成され、オーバーロードフラグにオーバーロードデリミタが続きます。オーバーロードフラグは 6 つのドミナントビットから構成され、このドミナントビットに他のノードの生成するオーバーロードフラグが続きます（アクティブエラーフラグに関しては、最大でドミナントビットは 12 となります）。オーバーロードデリミタは 8 つのリセッショビットから構成されます。オーバーロードフレームは、2 つの条件によりノードにより生成されることがあります。その条件の第一は、ノードがインターフレームスペース中にドミナントビットを検出することですが、これは不正な条件です。もう 1 つは、内部の条件により、ノードがまだ次のメッセージ受信を開始できない場合です、ノードは、続くメッセージの開始を遅らせるために、最大 2 つの連続オーバーロードフレームを生成することがあります。

#### B.11 インターフレームスペース

インターフレームスペースは、(あらゆる種類の)進行中フレームを、続くデータやリモートフレームから分離させます。インターフレームスペースは、インターミッションと呼ばれる最低 3 つのリセッショビットから構成されます。インターミッションにより、ノードには次のメッセージフレーム開始前に、内部プロセッシングを行う時間が与えられます。インターミッション後、バスラインは次の送信開始までリセッショブ状態（バスアイドル）にとどまります。

図 B-2: 標準データフレーム

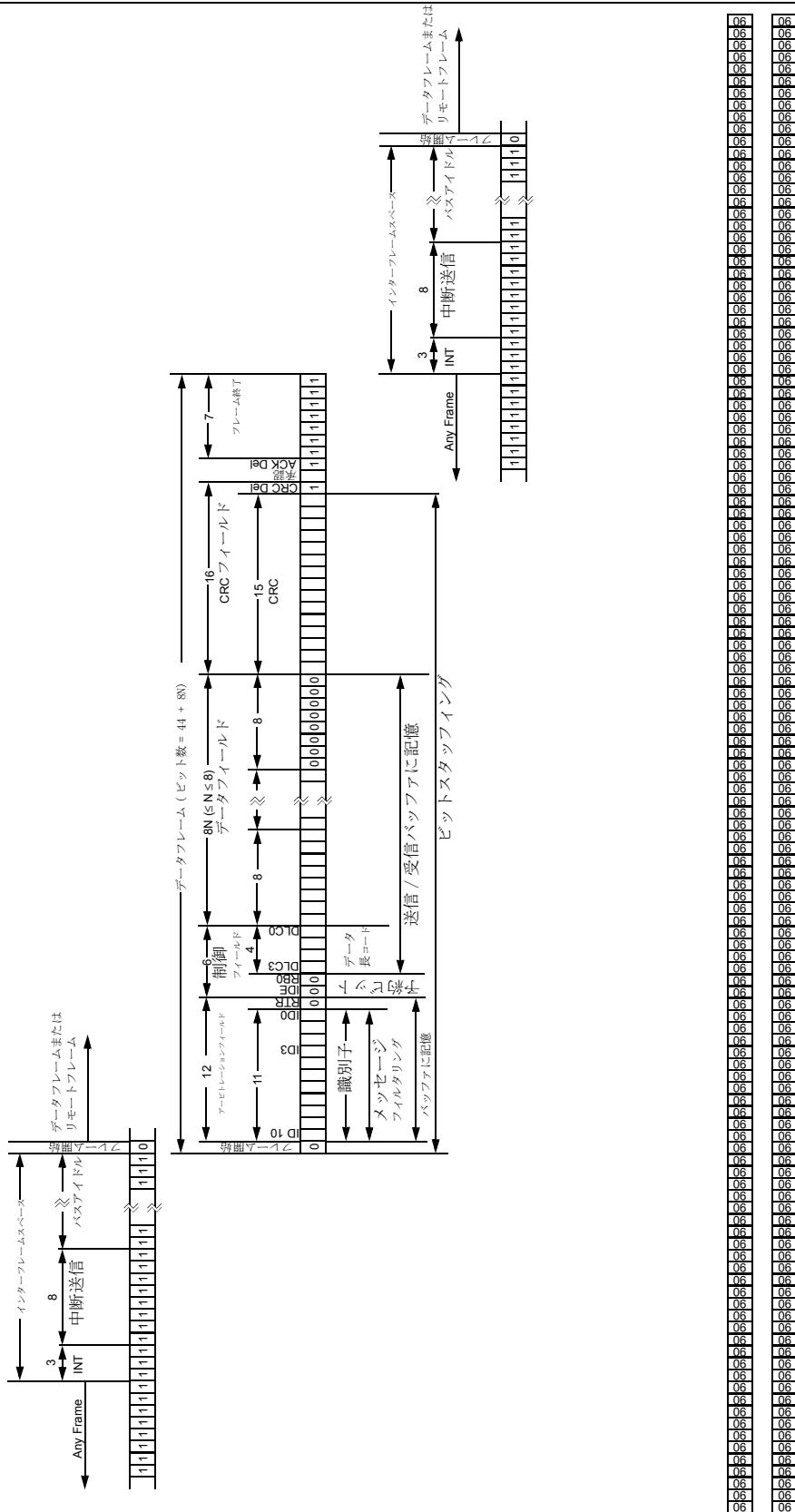


図 B-3: 拡張データフォーマット

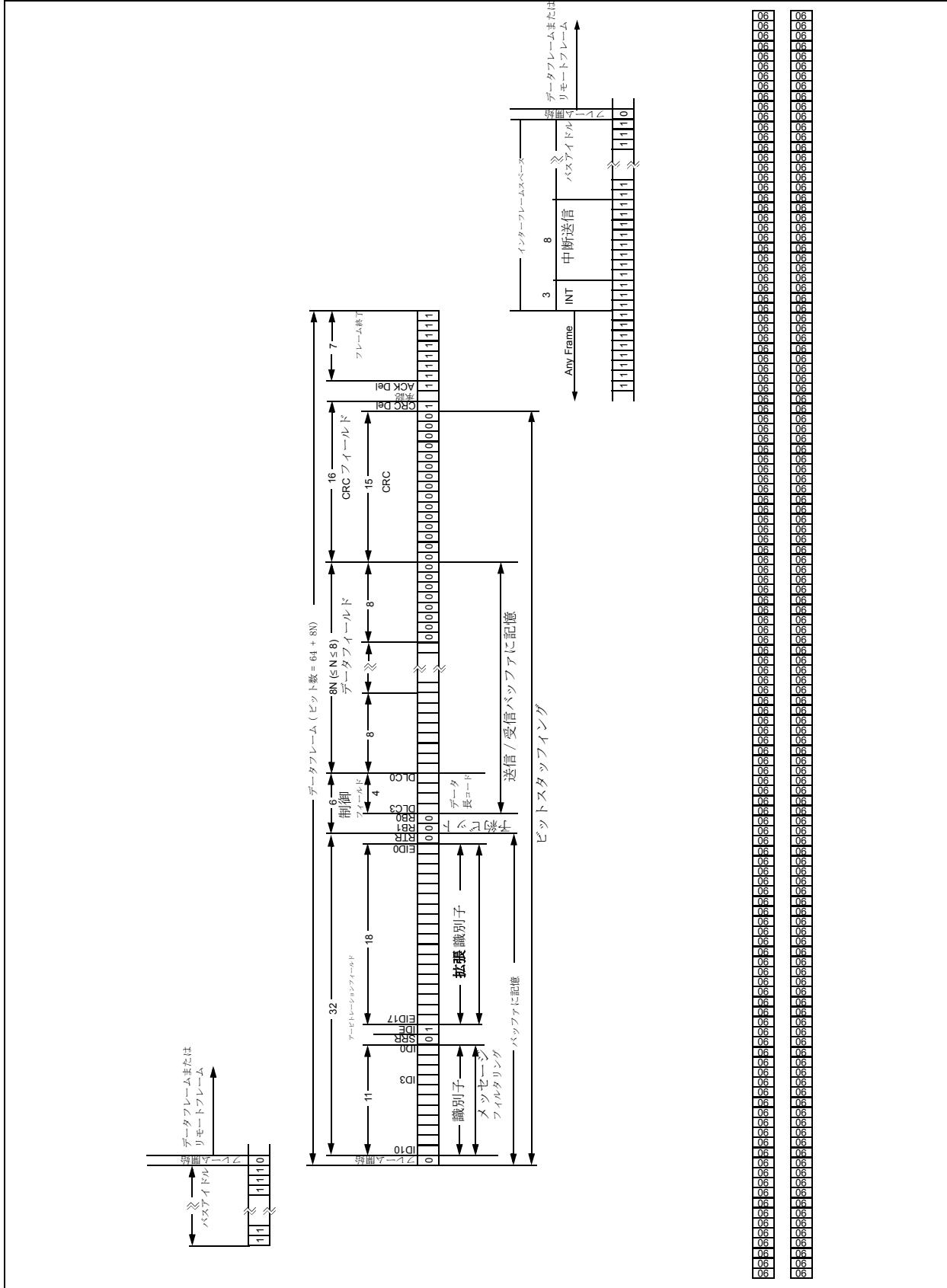


図 B-4: リモートデータフレーム

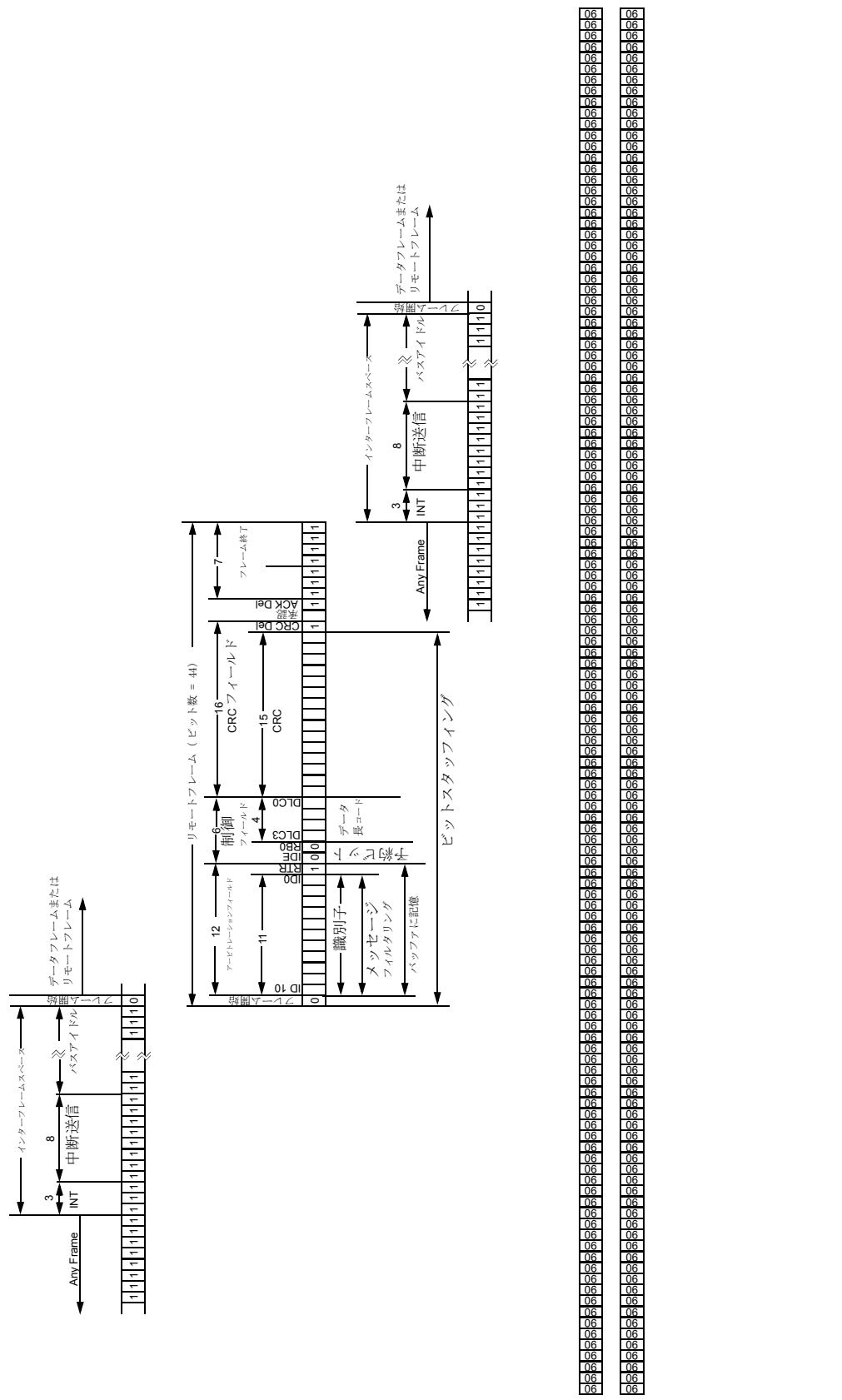


図 B-5: エラーフレーム

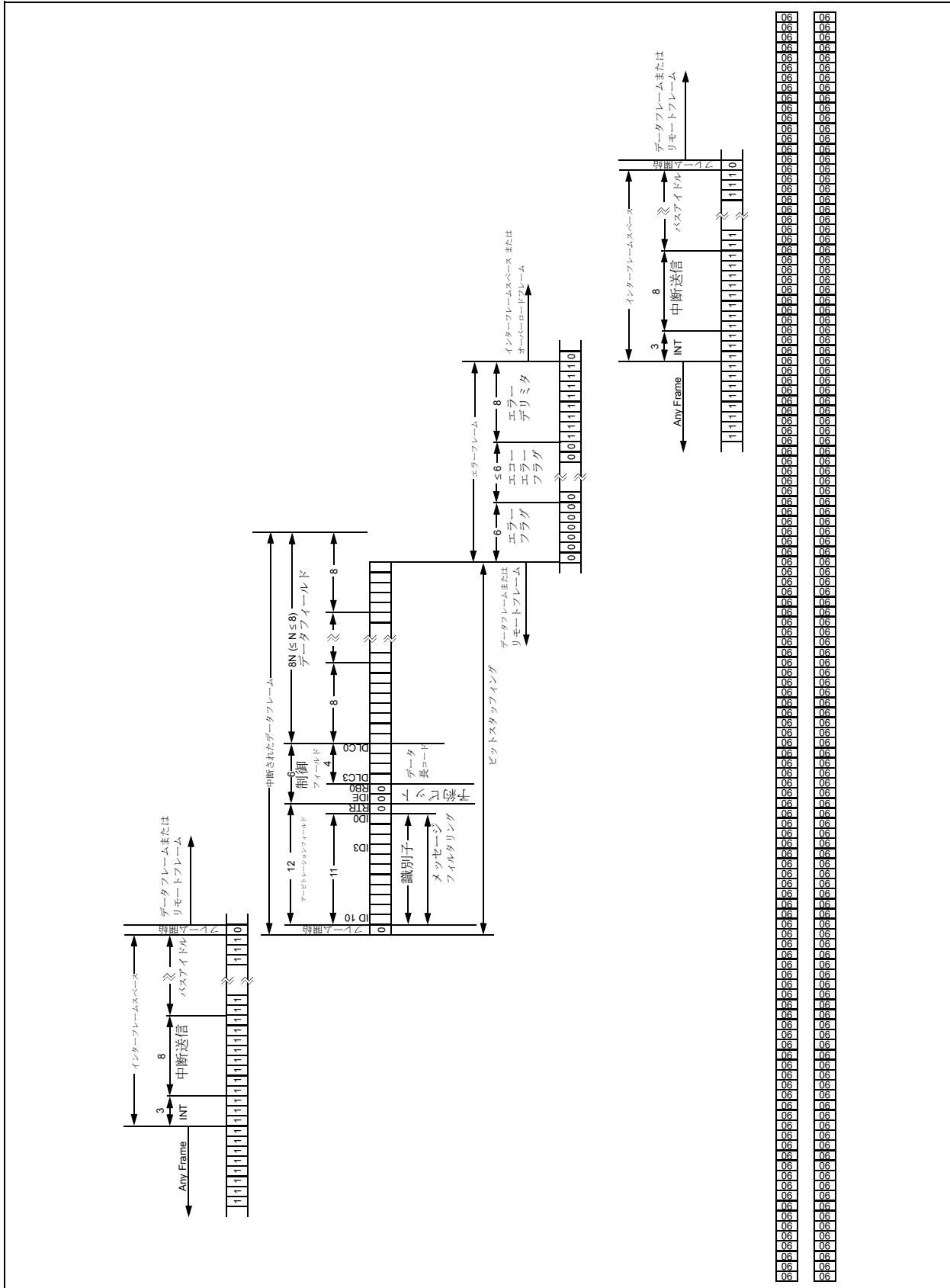
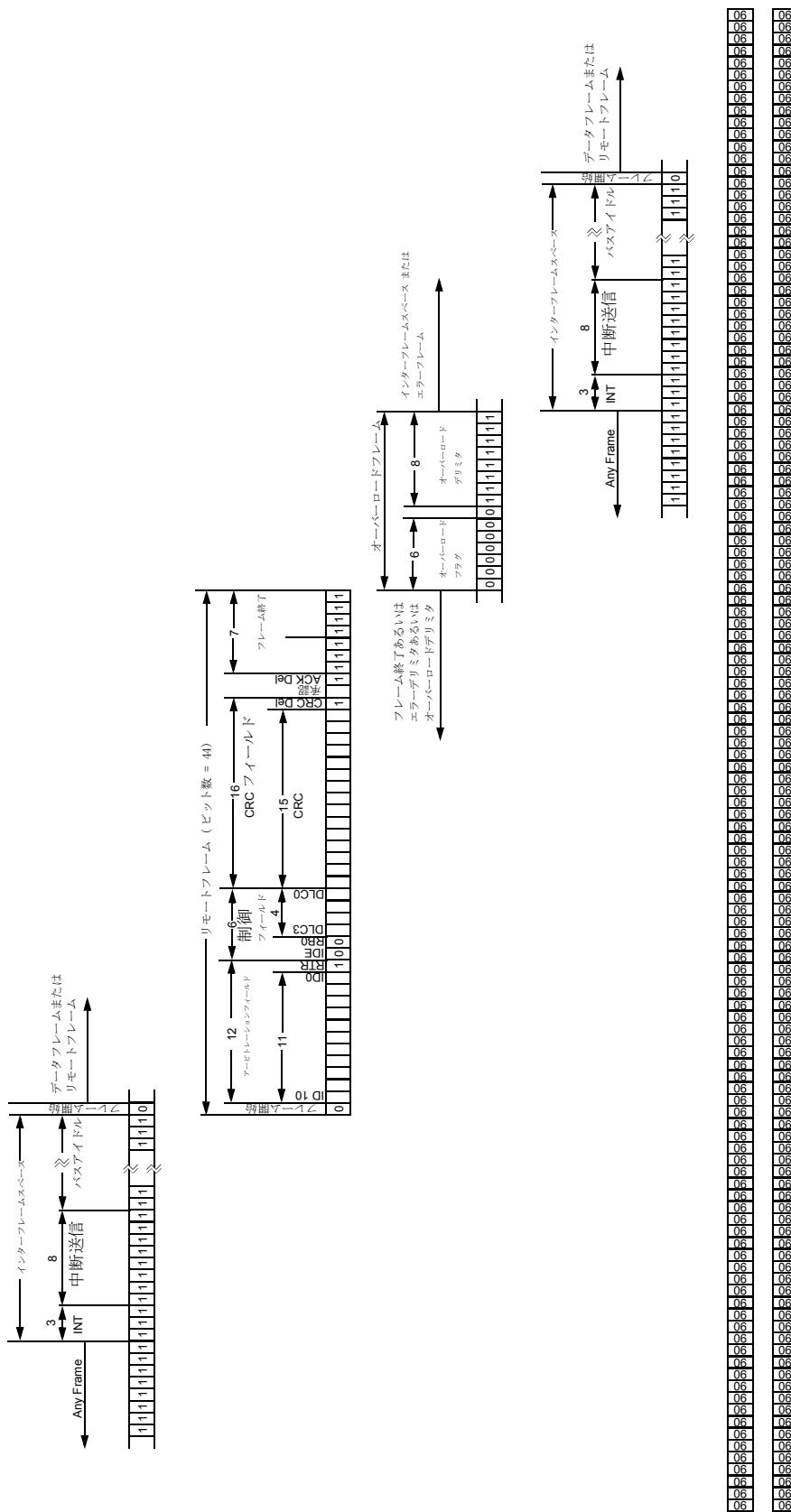


図 B-6: オーバーロードフレーム



### B.12 参照ドキュメント

タイトル

ドキュメント

自動車；デジタル情報相互交換、制御エリアネットワーク ISO11898  
Bosch CAN 仕様 バージョン 2.0

## 付録 C: コーデックプロトコル概要

本付録では、インター IC サウンド ( $I^2S$ ) 向けのオーディオコーダー / デコーダ (CODEC) プロトコル、および AC-Link コンプライアントモードインターフェースについて簡単に説明します。オーディオアプリケーション向けの多くのコーデックでは、8 kHz から 48 kHz のサンプリング周波数が用いられ、多くの場合に挙げたインターフェースプロトコルのうち 1 つが用いられます。データコンバータインターフェース (DCI) モジュールは、これらコーデックと関連するインターフェースタイミングを自動的に処理します。要求された量のデータが DCI により送信または受信、あるいはその両方がなされるまで CPU からのオーバーヘッドは必要です。CPU 割り込み間では最大 4 つのデータワードが転送されます。

### C.1 $I^2S$ プロトコルについての記述

インター IC サウンド ( $I^2S$ ) は、簡単な 3 線のバスインターフェースで、以下のデバイス間のデジタルオーディオデータの送信に用いられます。

- DSP プロセッサ
- A/D および D/A コンバータ
- デジタル入力 / 出力インターフェース

この付録に含まれる情報は、フィリップス株式会社の発行する  $I^2S$  プロトコル仕様書<sup>®</sup> の補足を意図したものであります。

$I^2S$  バスは、時分割多重方式が用いられており、2 つのデータチャンネルを転送します。これら 2 つのデータチャンネルは、多くの場合デジタルオーディオストリームの左右のチャンネルです。

$I^2S$  バスには以下の接続ピンが備えられています。

- SCK:  $I^2S$  シリアルクロックライン
- SDx:  $I^2S$  シリアルデータライン (入力または出力)
- WS:  $I^2S$  ワード選択ライン

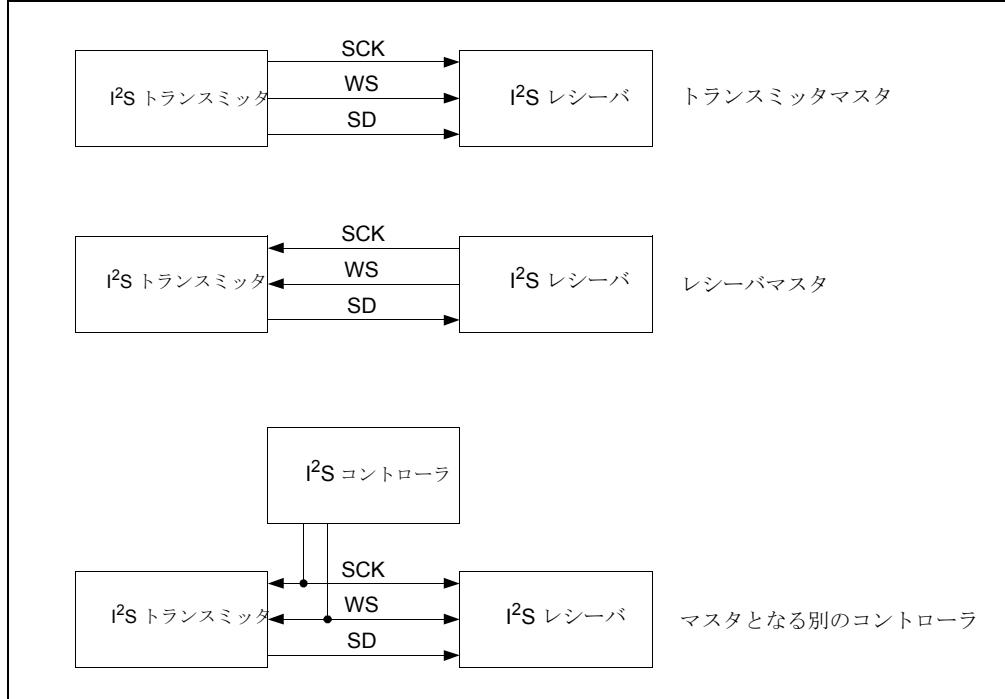
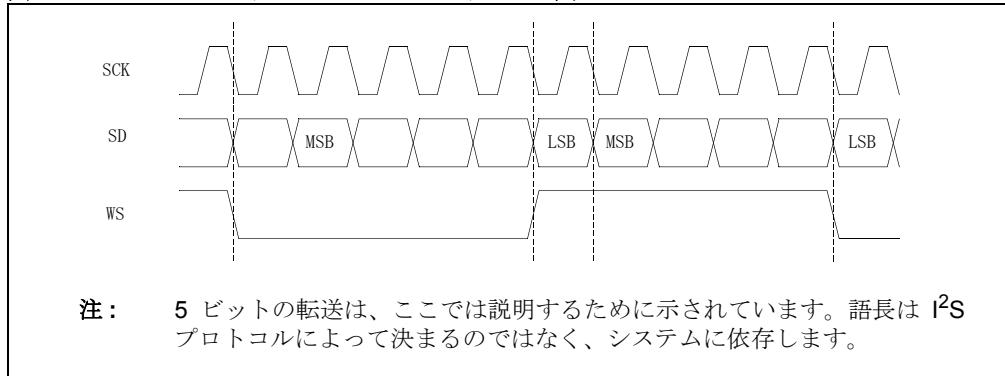
データ転送のタイミング図は図 C-2 に示されています。シリアルデータは  $I^2S$  バス上で送信され、2 つの補数フォーマットが用いられ、最上位ビットが最初に送信されます。プロトコルではトランスマッタやレシーバの様々なデータ語長が認められるため、最上位ビットが最初に転送される必要があります。レシーバに対し、データワードの面で受容できる以上のビットが送信された場合、下位ビットは無視されます。レシーバに対し、ネイティブな語長以下のビットが送信された場合、レシーバは残りの下位ビットを内部でゼロに設定する必要があります。

WS ラインは、送信されるデータチャンネルを指し示しています。以下の標準が用いられます。

- WS = 0: チャンネル 1 または左側オーディオチャンネル
- WS = 1: チャンネル 2 または右側オーディオチャンネル

WS ラインは、SCK の立ち上がりエッジ時にレシーバによりサンプリングされ、次のデータワードの最上位ビットは、WS 変更後 1 SCK 周期後に送信されます。WS 変更後 1 ピリオドの遅れにより、レシーバは以前送信されたワードを記憶し、次のワードに備える時間が与えられます。トランスマッタにより送信されるシリアルデータは SCK の立ち下がりエッジ時にバスに送られ、SCK の立ち上がりエッジ時にレシーバによりラッチされます。

$I^2S$  システムにおいては、どのデバイスもシステムマスターとして作動します。システムマスターは、SCK および WS 信号を生成します。一般的に、トランスマッタがシステムマスターですが、レシーバや別のデバイスがシステムマスターの機能を果たすこともあります。図 C-1 は、可能な  $I^2S$  のバス設定を示したものです。図 C-1 には示されていませんが、接続された 2 つのデバイスがデータ送信およびデータ受信接続の両方を有することもあります。

図 C-1: I<sup>2</sup>S バス接続図 C-2: I<sup>2</sup>S インターフェースタイミング図

## C.2 AC '97 プロトコル

オーディオ CODEC '97 (AC '97) の仕様書では、PC のプラットフォームに使用されるオーディオ CODEC の標準的なアーキテクチャーとデジタルインターフェースが規定されています。AC'97 に従い作られた CODEC のデジタルインターフェースプロトコルは AC-Link と呼ばれ、本書における議論の中心となります。AC'97 制御デバイス固有の必要事項および機能はここで取り上げられていません。

本付録中の情報は、インテル株式会社が発行する AC'97 コンポーネント設計ドキュメントを補足するためのものです。

## C.3 AC-Link シグナルに関する記述

全ての AC-Link シグナルは、AC'97 のマスタークロックソースから発せられるものです。クロックを最低限にできる推奨クロックソースは、AC'97 に接続される 24.576MHz のクリスタルです。24.576MHz のクロックは、AC'97 コントローラあるいは外部ソースにより提供されることもあります。

全ての AC-Link シグナルの名称は、AC'97 CODEC ではなく、AC'97 コントローラを参照しています。コントローラとは、データ転送を開始するために SYNC シグナルを生成するデバイスです。それぞれのデバイスについては後の項で記述されています。

## C.3.1 ビットクロック (BIT\_CLK)

12.288 MHz の BIT\_CLK シグナルは、システムにおいてマスター AC'97 コデックにより発せられます。BIT\_CLK シグナルは、システムにおいて AC'97 コントローラ、および最大 3 つのスレーブ AC'97 コデックデバイスに対し入力されます。AC-Link トランジション上の全てのデータは、BIT\_CLK の立ち上がりで遷移し、BIT\_CLK の立ち下がりエッジで受信デバイスによりサンプルされます。

## C.3.2 シリアルデータ出力 (SDO)

SDO とは、AC'97 コデックに送信される時間分割多重データストリームです。

## C.3.3 シリアルデータ入力 (SDI)

SDI は、AC'97 コデックからの時間分割多重データストリームです。

## C.3.4 SYNC

SYNC とは、AC'97 コントローラから AC'97 コデックに発信される、周波数が 48kHz に固定されたサンプル同期化シグナルです。SYNC シグナルは、BIT\_CLK シグナルを 256 分周することにより得られます。SYNC シグナルは 16BIT\_CLK 周期では High であり、240BIT\_CLK 周期の間 Low となっています。SYNC シグナルは BIT\_CLK の立ち上がりエッジ時にのみ変更され、シグナルの周期によりオーディオデータフレームの境界が規定されます。

## C.3.5 リセット

RESET シグナルはシステム上で各 AC'97 コデックに入力され、コデックハードウェアをリセットします。

## C.4 AC-Link プロトコル

### C.4.1 AC-Link シリアルインターフェースプロトコル

AC-Link シリアルデータストリームでは、256 ビットのデータフレームを有する時間分割多重 (TDM) スキームが用いられます。各データフレームは 13 のタイムスロットに分割され、それぞれスロット #0 からスロット #12 まで番号がつけられます。スロット #0 は、16 ビットを含む特別なタイムスロットです。残りの 12 スロットは 20 ビットとなっています。

AC-Link フレームの例は図 C-4 に示されています。フレームは BIT\_CLK の立ち上がりと同時に発生する SYNC の立ち上がりエッジにより開始されます。AC'97 コデックは、すぐ後に続く BIT\_CLK の立ち下がりエッジの際に SYNC の宣言をサンプルします。この立ち下がりエッジは、コデックとコントローラの両方が新たなフレーム開始を準備する時点となります。その後の BIT\_CLK の立ち上がりエッジ発生時に、コデックは SDATA\_IN の最上位ビットを送信し、またコデックは SDATA\_OUT の最初のエッジを送信します。このシーケンスにより、データの移行、および後に続く内向き・外向きのデータストリーム両方のサンプリングが時間順に整列されます。

スロット #0、#1、#2 は、AC-Link プロトコルにおいてステータスおよび制御のために特殊な用途で用いられます。残りのタイムスロットは、ある種のデジタルオーディオデータに割り当てられます。スロット #3 からスロット #12 のデータ割り当ては、選択される AC'97 によって決まるため、スロットの用途をここで簡潔に要約しました。スロットの用途に関する詳細については、AC'97 コンポーネントの仕様書を参照してください。

### C.4.2 スロット #0、タグフレーム

スロット #0 は通常  $\$$  タグフレーム  $\$$  と呼ばれています。タグフレームには、AC-Link プロトコルの各データタイムスロット毎にビットが対応します。これらビットは、コントローラが使用するにあたり、フレーム中のどのタイムスロットが有効か決定するために用いられます。スロット #0 のあるビット位置で  $\$1E$  と記されていた場合、現在のオーディオフレーム中の対応するタイムスロットがデータストリームに割り当てられたこと、タイムスロット中に有効なデータが含まれていることを意味しています。スロットが無効であると  $\$$  タグ  $\$$  をつけられた場合、スロットがアクティブな時間内に、データソースの責任で(入力ストリームの場合は AC'97 コデック、出力ストリームの場合は AC'97 コントローラ) 全てのビット位置に 0 が詰め込まれます。

タグフレームには特殊なビットもあります。SDATA\_OUT 用タグフレームの最上位ビットは、 $\$$  フレーム有効  $\$$  を示すステータスビットです。フレーム有効ビットは、フレームの最低 1 つのタイムスロットには有効なデータが含まれていることをコデックに指示する役割を果たします。フレーム全体が無効であるとタグ付けされた場合、コデックはフレームにおけるその後の全てのスロットを無視することができます。この機能は、48kHz 以外のサンプルレートを実行するために用いられます。

**SDATA\_OUT** タグフレームの 2 つの最下位ビットにより、コデックのアドレスが示されます。システム中、最大 4 つの AC '97 コデックが接続可能です。システム内で 1 つのコデックのみが使用されている場合、このビットは 0 のままで。

**SDATA\_IN** の最上位ビットは、 $\phi$  コデック準備完了 £ ステータスビットとして用いられます。このビット位置が ‘0’ の場合、コデックは電源が落ちており、通常の動作を行う準備はできません。 $\phi$  コデック準備完了 £ ビットがセットされていると、コントローラによってコデックのステータスレジスタに対し、どのサブセクションが動作可能か問い合わせされます。

#### C.4.3 スロット #1 (コマンドアドレス) およびスロット #2 (コマンドデータ)

スロット #1 およびスロット #2 も、AC-Link プロトコルで特殊な用途を持っています。これらのタイムスロットは、AC '97 のコデック制御レジスタを読み出したり書き込みする場合、アドレスおよびデータ値として用いられます。このようなタイムスロットは、制御レジスタから読み出したり、書き込みするためには、スロット #0 において有効とタグ付けされる必要があります。AC '97 コンポーネントの仕様は、コデック中に 64 の 16 ビット制御レジスタが用意されています。7 つのアドレスビットが AC-Link プロトコルにあります。偶数のアドレスのみが用いられています。奇数のアドレス値は保留されています。

**SDATA\_OUT** ラインのスロット #1 とスロット #2 は、それぞれコマンドアドレスおよびコマンドデータと呼ばれています。**SDATA\_OUT** ラインのコマンドアドレススロットは、コデックレジスタアドレスを指定し、レジスタアクセスが読み出しか書き込みか決定するために用いられています。**SDATA\_OUT** のコマンドデータスロットには 16 ビットの値が含まれ、この値がコデック制御レジスタの 1 つに書き込まれます。コデックレジスタの読み取り中、コマンドデータビットは  $\phi 0E$  にセットされます。

**SDATA\_IN** ラインのスロット #1 とスロット #2 は、それぞれステータスアドレスおよびステータスデータスロットと呼ばれています。ステータスアドレスタイムスロットは、以前にコデックに送信されたレジスタアドレスを繰り返します。もしレジスタアドレスが  $\phi 0E$  の場合、以前にコデックに無効なアドレスが送信されています。

ステータスアドレスタイムスロットにはまた、10 のスロットリクエストビットが用意されています。スロットリクエストビットは、様々なサンプルレートを持つアプリケーション向けにコデックにより操作されています。

ステータスデータタイムスロットは、コデックの制御 / ステータスレジスタから、16 ビットのデータ読み取りを返します。

#### C.4.4 スロット #3 (PCM 左側チャンネル)

**SDATA\_OUT** シグナルのスロット #3 は、コンポジットデジタルオーディオの左側プレーバックストリームに用いられます。サウンドカードアプリケーションでは、一般的に WAV オーディオと MIDI シンセサイザを合わせた出力になります。

**SDATA\_IN** シグナルのスロット #3 は、AC'97 コデック入力ミキサから取られた左側チャンネルの記録データです。

#### C.4.5 スロット #4 (PCM 右側チャンネル)

**SDATA\_OUT** シグナルのスロット #4 は、コンポジットデジタルオーディオの右側プレーバックストリームに用いられます。サウンドカードアプリケーションでは、一般的に WAV オーディオと MIDI シンセサイザを合わせた出力になります。

**SDATA\_IN** シグナルのスロット #4 は、AC'97 コデック入力ミキサから取られた右側チャンネルの記録データです。

#### C.4.6 スロット #5 (モデムライン 1)

**SDATA\_OUT** シグナルのスロット #5 は、モデム DAC データに用いられます。モデム互換 AC'97 コデックのデフォルト解像度は 16 ビットです。あらゆるタイムスロットと同様に、全スロットの未使用ビットは ‘0’ にセットされます。

**SDATA\_IN** シグナルのスロット #5 は、モデム ADC データに使用されます。

#### C.4.7 スロット #6

**SDATA\_OUT** シグナルのスロット #6 は、4 チャンネルまたは 6 チャンネルのサウンド設定において、PCM センターチャンネル DAC データに用いられます。

**SDATA\_IN** シグナルのスロット #6 は、専用マイクロホン録音データに用いられます。スロット内のデータにより、エコーキャンセルアルゴリズムをスピーカーホンのアプリケーションで利用することができます。

## C.4.8 スロット #7

SDATA\_OUT シグナルのスロット #7 は、4 チャンネルまたは 6 チャンネルのサウンド設定で PCM 左側チャンネルの DAC データに用いられます。

SDATA\_IN シグナルのスロット #7 は、AC'97 のコンポーネント設計では将来的に利用するため に保留されています。

## C.4.9 スロット #8

SDATA\_OUT シグナルのスロット 8# は、4 チャンネルまたは 6 チャンネルのサウンド設定で PCM 右側チャンネル DAC データに用いられます。

SDATA\_IN シグナルのスロット #8 は、AC'97 のコンポーネント設計では将来的に利用するため に保留されています。

## C.4.10 スロット #9

SDATA\_OUT シグナルのスロット #9 は、6 チャンネルサウンド設定で PCM LFE DAC データ に用いられます。

SDATA\_IN シグナルのスロット #9 は、AC'97 のコンポーネント設計では将来的に利用するため に留保されています。

## C.4.11 スロット #10 (モデムライン 2)

スロット #10 は、モデム互換デバイスにおいてモデムライン 2 の ADC および DAC データに用 いられます。

## C.4.12 スロット #11 (モデムハンドセット)

スロット #11 は、モデム互換デバイスにおいてモデムハンドセットの ADC および DAC データ に用いられます。

## C.4.13 スロット #12 (GPIO 制御 / ステータス )

スロット #12 のビットは、AC'97 コデックにおいて GPIO ピンの読み出し、書き込みに用いられ ます。GPIO ピンはモデム互換デバイスでモデム制御機能に用いられます。

図 C-3: AC-Link シグナルの接続

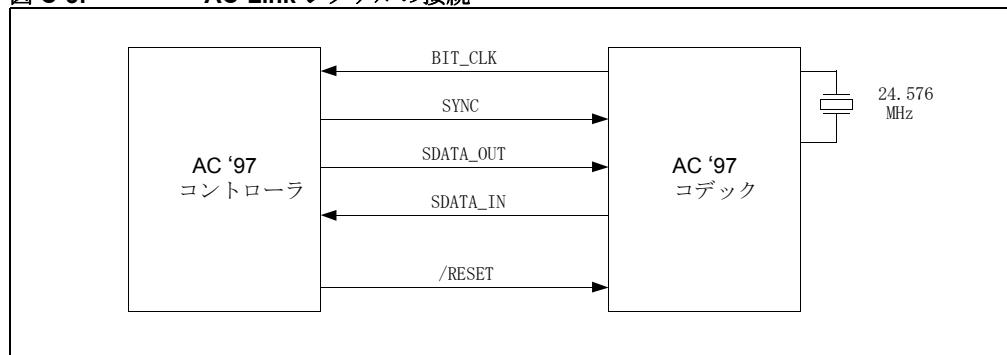


図 C-4:

AC-Link データフレーム

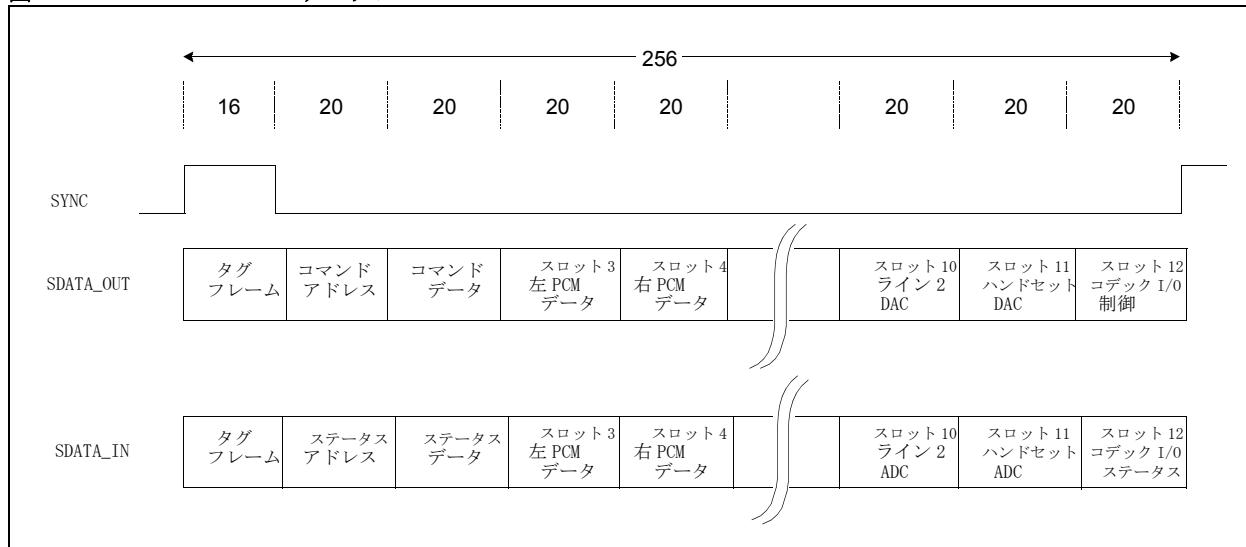


図 C-5:

スロット #0、スロット #1 およびスロット #2 の SDATA\_IN ビットの位置

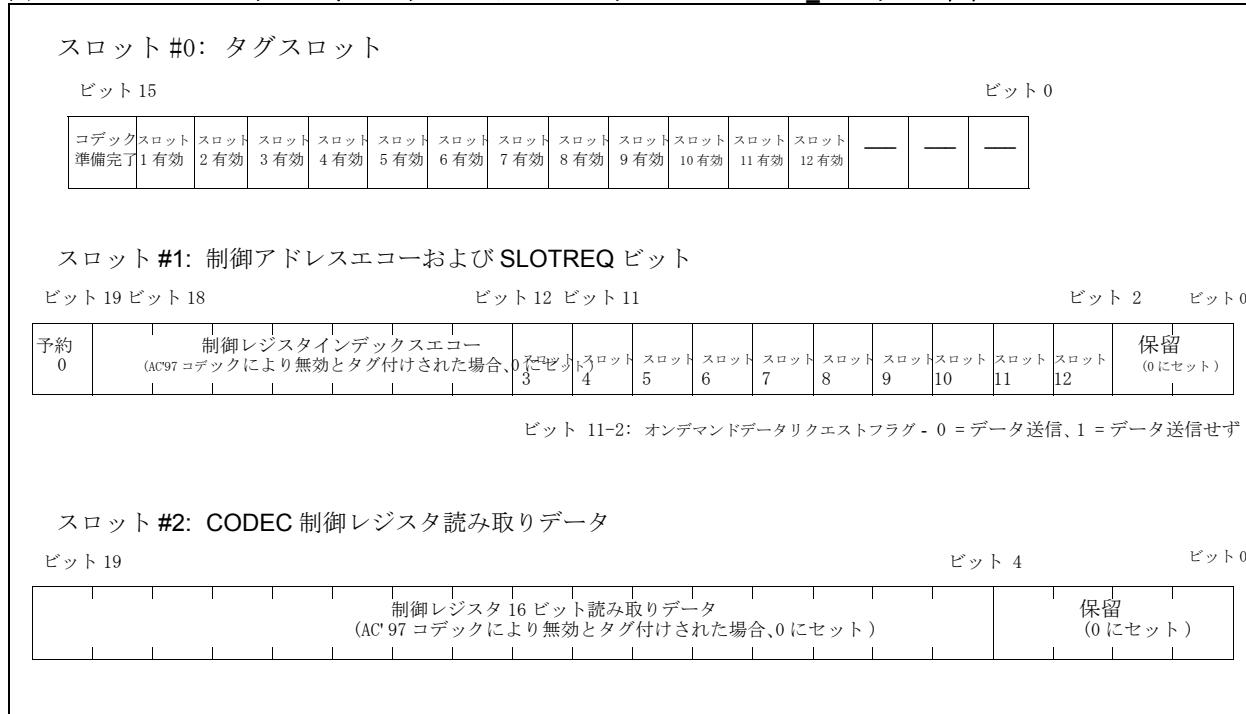
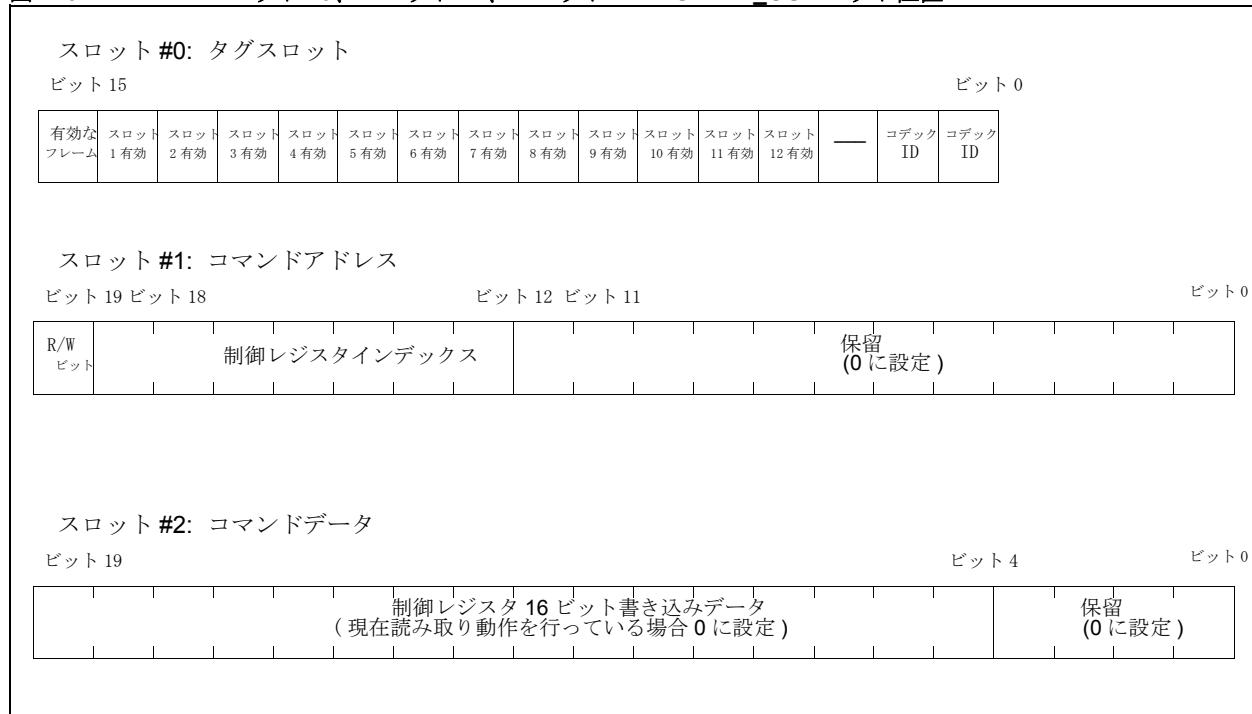


図 C-6: スロット #0、スロット #1、スロット #2 の **SDATA\_OUT** ビット位置



**Numerics**

10-bit address mode .....	21-35
10-bit High Speed A/D	
TAD vs. Device Operating Frequencies Table.....	17-14
12-Bit A/D	
ADCHS .....	17-4, 18-4
ADPCFG.....	17-4, 18-4
12-bit A/D	
Operation During CPU IDLE Mode .....	17-52, 18-33
Operation During CPU SLEEP Mode .....	17-52, 18-33
16-bit Up/Down Position Counter.....	16-11
32-bit Timer Configuration .....	12-16

**A****A/D**

Accuracy/Error .....	17-50, 18-31
ADCON0 Register.....	17-4, 18-4
Configuring Analog Port Pins.....	17-15, 18-13
Edge Detection Mode .....	13-8
Effects of a Reset.....	16-19, 17-52
Effects of a Reset (12-bit) .....	18-33
Enabling the Module .....	17-17
Enabling the Module (12-bit).....	18-15
How to Start Sampling .....	17-18
How to Stop Sampling and Start Conversions .....	17-19
Reading the A/D Result Buffer.....	17-49
Sampling Requirements.....	17-46, 18-27
Sampling Time .....	17-47, 18-28
Transfer Function .....	17-50, 18-31
Transfer Function (12-bit) .....	18-31
A/D Accuracy/Error .....	17-50
A/D Accuracy/Error (12-bit).....	18-31
A/D Converter External Conversion Request .....	6-10
A/D Module Configuration .....	17-13
A/D Module Configuration (12-bit) .....	18-11
A/D Sampling Requirements (12-bit) .....	18-27
A/D Special Event Trigger .....	12-22
A/D Terminology and Conversion Sequence .....	17-11
A/D Terminology and Conversion Sequence (12-bit) .....	18-10
Accumulator 'Write Back' .....	2-25
ACK.....	21-38
Acknowledge Pulse.....	21-38
Address Generator Units and DSP Class Instructions .....	3-6
Address Register Dependencies .....	2-35
AKS.....	21-17
Alternate Vector Table .....	6-2
Arithmetic Logic Unit (ALU). ....	2-17

**B**

Barrel Shifter.....	2-26
Baud Rate	
Generator (BRG).....	19-8
Tables .....	19-9
BF .....	21-18, 21-19
Bit-Reversed Addressing .....	3-14
and Modulo Addressing .....	3-15
Code Example .....	3-18
Intro .....	3-14
Modifier Value .....	3-16
Operation .....	3-15
Block Diagrams	
Dedicated Port Structure.....	11-2
DSP Engine .....	2-19
dsPIC30F CPU Core.....	2-3
External Power-on Reset Circuit (For Slow VDD Rise Time).....	8-7

Input Capture .....	13-2
Input Change Notification .....	11-5
Low Voltage Detect (LVD).....	9-3
Oscillator System.....	7-3
Output Compare Module .....	14-2
RESET System.....	8-2
Shared Port Structure .....	11-4
Type A Timer .....	12-3
Type B - Type C Timer Pair (32-bit Timer) .....	12-17
Type B Timer .....	12-4
Type C Timer .....	12-5
UART .....	19-2
UART Receiver .....	19-16
UART Transmitter .....	19-11
WDT .....	10-5
BOR and POR Configuration Register .....	24-5
BOR AND POR Fuses (0000h) .....	24-5
Brown-out Reset (BOR) .....	8-8
Configuration .....	8-9
Current Consumption for Operation .....	8-9
Byte to Word Conversion .....	2-17

**C****CAN**

Buffer Reception and Overflow Truth Table .....	23-46
Message Acceptance Filters .....	23-12
CAN Library .....	25-9
Capture Buffer Operation .....	13-8
Clock Switching	
Aborting .....	7-23
Enable .....	7-21
Entering SLEEP Mode During .....	7-23
Operation .....	7-21
Recommended Code Sequence .....	7-23
Tips .....	7-23

**CN**

Change Notification Pins .....	11-5
Configuration and Operation .....	11-6
Control Registers .....	11-6
Operation in SLEEP and IDLE Modes .....	11-6

**Code Examples**

8-bit Transmit/Receive (UART1) .....	19-20
9-bit Transmit/Receive (UART1), Address Detect Enabled .....	19-20
Clock Switching .....	7-24
Compare Mode Toggle Mode Pin State Setup .....	14-8
Compare Mode Toggle Setup and Interrupt Servicing .....	14-8
Configuration Register Write .....	5-14
Continuous Output Pulse Setup and Interrupt Servicing .....	14-16
Initialization Code for 16-bit Asynchronous Counter Mode Using an External Clock Input .....	12-11
Initialization Code for 16-bit Gated Time Accumulation Mode .....	12-13
Initialization Code for 16-bit Synchronous Counter Mode Using an External Clock Input .....	12-10
Initialization Code for 16-bit Timer Using System Clock .....	12-9
Initialization Code for 32-bit Gated Time Accumulation Mode .....	12-20
Initialization Code for 32-bit Synchronous Counter Mode Using an External Clock Input .....	12-19

Initialization Code for 32-bit Timer Using Instruction Cycle as Input Clock.....	12-18
Prescaled Capture .....	13-7
PWM Mode Pulse Setup and Interrupt Servicing...	14-22
Reading from a 32-bit Timer .....	12-21
Single Output Pulse Setup and Interrupt Servicing.....	14-12
Single Row Programming .....	5-13
CODEC Interface Basics and Terminology.....	22-8
Complementary PWM Output Mode .....	15-24
Configuration Bit Descriptions .....	24-7
BOR and POR .....	24-7
General Code Segment .....	24-7
Motor Control PWM Module.....	24-7
Oscillator .....	24-7
Connection Considerations .....	17-50
Connectivity Development Board.....	25-14
Control Register Descriptions .....	3-18
Control Registers .....	12-6, 17-4, 18-4
Assignment of Interrupts .....	6-14
Controlling Sample/Conversion Operation.....	17-30
Controlling Sample/Conversion Operation (12-bit) .....	18-20
Conversion Sequence Examples .....	17-32
Conversion Sequence Examples (12-bit).....	18-22
CPU	
Register Maps.....	2-38
Related Application Notes.....	2-40
Revision History .....	2-41
CPU Clocking Scheme.....	7-4
CPU Priority Status .....	6-5
CPU Register Descriptions .....	2-11
Crystal Oscillators, Ceramic Resonators .....	7-9
<b>D</b>	
Data Accumulator	
Status Bits .....	2-23
Data Accumulator Adder/Subtractor .....	2-23
Data Accumulators .....	2-20
Data Alignment .....	3-7
Data EEPROM Programming .....	5-14
Erasing One Row .....	5-18
Erasing One Word .....	5-16
Reading Memory .....	5-20
Row Algorithm.....	5-15
Single Word Algorithm .....	5-15
Write One Row .....	5-19
Writing One Word .....	5-17
Data Memory	
Map .....	3-3
Near .....	3-4
Data Space Address	
Generator Units (AGUs).....	3-5
X Address Generator Unit.....	3-5
Y Address Generator Unit.....	3-5
Data Space Write Saturation.....	2-24
DCI	
Buffer Control Unit .....	22-13
Control Register Descriptions .....	22-2
Operation .....	22-10
Using the Module .....	22-17
Dead Time Control .....	15-25
Determining Best Values for Crystals,	
Clock Mode, C1, C2 and Rs .....	7-11
Device Configuration	
Register Descriptions.....	24-2
Device Identification Registers .....	24-8
Device ID (DEVID).....	24-8
Unit ID Field .....	24-8
Device RESET Times .....	8-11
Device Start-up Time Lines .....	8-13
Device Wake-up on SLEEP/IDLE.....	13-10
Disable Interrupts Instruction .....	6-8
Divide Support .....	2-27
DSP Algorithm Library .....	25-7
DSP Engine .....	2-18
DSP Engine Mode Selection .....	2-26
DSP Engine Trap Events .....	2-26
DSP Filter Design Software Utility .....	25-8
dsPIC Language Suite .....	25-3
dsPIC30F Hardware Development Boards.....	25-11
<b>E</b>	
Equations	
Calculating the PWM Period.....	14-19
Calculation for Maximum PWM Resolution .....	14-20
Modulo End Address for Incrementing Buffer.....	3-9
Modulo Start Address for Decrementing Buffer .....	3-9
WDT Time-out Period .....	10-6
External Clock Input.....	7-12
External Interrupt Support.....	6-10
External RC Oscillator .....	7-13
Operating Frequency .....	7-14
Start-up .....	7-14
with I/O Enabled .....	7-14
External RESET (EXTR).....	8-7
<b>F</b>	
Fail-Safe Clock Monitor (FSCM).....	7-19
and Slow Oscillator Start-up .....	7-20
and WDT .....	7-20
Delay .....	7-19
FLASH and Data EEPROM Programming	
Control Registers .....	5-5
NVMADR .....	5-6
NVMCON .....	5-5
NVMKEY .....	5-6
FLASH Program Memory	
Erasing a Row .....	5-11
Loading Write Latches .....	5-12
Programming Algorithm.....	5-10
FSCM	
and Device RESETS .....	8-12
Delay for Crystal and PLL Clock Sources.....	8-12
FWDT	
Watchdog Timer Configuration Register.....	24-4, 24-6
<b>G</b>	
General Purpose Development Board .....	25-12
<b>H</b>	
Hard Traps .....	6-7
Address Error (Level 13).....	6-8
Oscillator Failure (Level 14).....	6-8
Priority and Conflicts .....	6-7
Stack Error (Level 12).....	6-6
How to Start Sampling (12-bit).....	18-15
How to Start Sampling and Start Conversions (12-bit) ..	18-15

**I**

I/O Multiplexing with Multiple Peripherals .....	11-4
I/O Pin Control .....	12-22, 13-10, 14-23, 16-18
I/O Port Control Registers .....	11-3
I/O Ports	
Related Application Notes .....	11-9
Revision History .....	11-10
I <sup>2</sup> C	
Acknowledge Generation .....	21-21
Building Complete Master Messages .....	21-24
Bus Arbitration and Bus Collision .....	21-30
Bus Collision During a Repeated Start Condition ..	21-31
Bus Collision During a Start Condition .....	21-31
Bus Collision During a Stop Condition .....	21-31
Bus Collision During Message Bit Transmission ..	21-31
Bus Connection Considerations .....	21-47
Communicating as a Master in a	
Multi-Master Environment .....	21-29
Communicating as a Slave .....	21-32
Detecting Bus Collisions and Re-sending	
Messages .....	21-30
Detecting Start and Stop Conditions .....	21-32
Detecting the Address .....	21-33
Enabling I/O .....	21-13
Enabling Operation .....	21-13
Generating Repeated Start Bus Event .....	21-23
Generating Start Bus Event .....	21-16
Generating Stop Bus Event .....	21-22
Initiating and Terminating Data Transfer .....	1-3
Interrupts .....	21-13
Master Message Protocol States .....	21-24
Module Operation during PWRSAV Instruction .....	21-49
Receiving Data from a Master Device .....	21-37
Receiving Data from a Slave Device .....	21-19
Sending Data to a Master Device .....	21-44
Sending Data to a Slave Device .....	21-17
START .....	1-3
STOP .....	1-3
I <sup>2</sup> C Module	
10-bit Address mode .....	21-35
Multi-master Mode .....	21-29
IDLE Mode .....	10-4
Time Delays on Wake-up from .....	10-5
Wake-up from on Interrupt .....	10-4
Wake-up from on RESET .....	10-5
Wake-up from on WDT Time-out .....	10-5
Independent PWM Output Mode .....	15-28
Initialization .....	17-51
Input Capture	
Associated Special Function Registers .....	13-11
Buffer Not Empty (ICBNE) .....	13-9
Design Tips .....	13-12
Interrupts .....	13-9
Control Bits .....	13-9
Operation in Power Saving States .....	13-10
Overflow (ICOV) .....	13-9
Related Application Notes .....	13-13
Input Capture Event Modes .....	13-4
Input Capture Registers .....	13-3
Instruction Flow Types .....	2-27
Instruction Stall Cycles .....	2-36
Internal Fast RC Oscillator (FRC) .....	7-18
Internal Low Power RC (LPRC) Oscillator .....	7-19
Enabling .....	7-19
Internal Voltage Reference .....	9-3

Interrupt Control and Status Registers .....	6-14
CORCON .....	6-14
IECx .....	6-14
IFSx .....	6-14
INTCON1, INTCON2 .....	6-14
IPCx .....	6-14
SR .....	6-14
Interrupt Controller	
Associated Special Function Registers .....	6-43
Interrupt Latency	
One-Cycle Instructions .....	6-11
Two-Cycle Instructions .....	6-12
Interrupt Operation .....	6-9
Nesting .....	6-9
Return From Interrupt .....	6-9
Interrupt Priority .....	6-5
Interrupt Processing Timing .....	6-11
Interrupt Setup Procedures .....	6-42
Initialization .....	6-42
Interrupt Disable .....	6-42
Interrupt Service Routine .....	6-42
Trap Service Routine .....	6-42
Interrupt Vector Table .....	6-2
Interrupts	
Design Tips .....	6-44
Related Application Notes .....	6-45
Revision History .....	6-46
Interrupts Coincident with Power Save Instructions .....	10-5
Initialization (12-bit) .....	18-32
Introduction	
Revision History .....	1-7
IWCOL .....	21-22
<b>L</b>	
LAT (I/O Latch) Registers .....	11-3
Loop Constructs .....	2-30
DO .....	2-32
REPEAT .....	2-30
Low Power 32 kHz Crystal Oscillator .....	7-18
Low Power 32 kHz Crystal Oscillator Input .....	12-15
LP Oscillator	
Continuous Operation .....	7-18
Enable .....	7-18
Intermittent Operation .....	7-18
Operation with Timer1 .....	7-18
LVD	
Control Bits .....	9-3
Current Consumption for Operation .....	9-5
Design Tips .....	9-6
Initialization Steps .....	9-5
Operation .....	9-5
Operation During SLEEP and IDLE Mode .....	9-5
Related Application Notes .....	9-7
Trip Point Selection .....	9-3
<b>M</b>	
Math Library .....	25-7
Microchip Hardware and Language Tools .....	25-2
Modes of Operation .....	14-4
Compare Mode Output Driven High .....	14-5
Compare Mode Output Driven Low .....	14-6
Compare Mode Toggle Output .....	14-7
Dual Compare Match .....	14-9
Dual Compare, Continuous Output Pulses .....	14-14
Dual Compare, Generating Continuous Output Pulses Special Cases (table) .....	14-17

Dual Compare, Single Output Pulse	
Special Cases (table)	14-13
Single Compare Match	14-4
Modulo Addressing	3-7
Applicability	3-11
Calculation	3-9
Initialization for Decrementing Buffer	3-13
Initialization for Incrementing Modulo Buffer	3-12
Start and End Address Selection	3-8
W Address Register Selection	3-9
Modulo Start and End Address Selection	
XMODEND Register	3-8
XMDSRT Register	3-8
YMODEND Register	3-8
YMODSRT Register	3-8
Motor Control Development Board	25-13
MPLAB 6.XX Integrated Development Environment	
Software	25-2
MPLAB ICD 2 In-Circuit Debugger	25-5
MPLAB ICE 4000 In-Circuit Emulator	25-4
MPLAB SIM Software Simulator	25-3
Multi-Master Mode	21-29
Multiplier	2-20
Multiply Instructions	2-22
<b>N</b>	
Non-Maskable Traps	6-6
Address Error	6-6
Arithmetic Error	6-6
Oscillator Failure	6-6
Stack Error	6-6
<b>O</b>	
Oscillator	
Configuration	7-5
Clock Switching Mode Bits	7-5
Design Tips	7-25
Related Application Notes	7-26
Resonator Start-up	7-9
Revision History	7-27
System Features Summary	7-2
Oscillator Control Register (OSCCON)	7-6
Oscillator Mode Selection Guidelines	7-8
Oscillator Start-up From SLEEP Mode	7-10
Oscillator Start-up Timer (OST)	7-18
Oscillator Switching Sequence	7-22
OSEK Operating Systems	25-10
Other dsPIC30F CPU Control Registers	2-16
DISICNT	2-16
MODCON	2-16
PSVPAG	2-16
TBLPAG	2-16
XBREV	2-16
XMDSRT, XMODEND	2-16
YMODSRT, YMODEND	2-16
Output Compare	
Associated Register Map	14-24
Design Tips	14-26
Related Application Notes	14-27
Revision History	14-28
Output Compare Operating in Power Saving States	14-23
Output Compare Operation in Power Saving States	
IDLE Mode	14-23
SLEEP Mode	14-23
Output Compare Registers	14-3
<b>P</b>	
Peripheral Driver Library	25-8
Peripheral Multiplexing	11-4
Peripherals Using Timer Modules	12-22
Phase Locked Loop (PLL)	7-17
Frequency Range	7-17
Lock Status	7-17
Loss of Lock During a Power-on Reset	7-17
Loss of Lock During Clock Switching	7-17
Loss of Lock During Normal Device Operation	7-17
POR and Long Oscillator Start-up Times	8-12
Port (I/O Port) Registers	11-3
Power Saving Modes	10-2
Power-on Reset (POR)	8-5
Using	8-7
Power-up Timer (PWRT)	8-7
Prescaler Capture Events	13-6
Primary Oscillator	7-8
Operating Modes	7-8
PRO MATE II Universal Device Programmer	25-5
Program Memory	
Address Map	4-2
Counter	4-4
Data Access From	4-4
Data Storage	4-7
High Word Access	4-7
Low Word Access	4-6
Program Space Visibility from Data Space	4-8
Related Application Notes	4-11
Table Address Generation	4-6
Table Instruction Summary	4-5
Writes	4-10
Programmable Digital Noise Filters	16-8
Programmable Oscillator Postscaler	7-20
Programmer's Model	2-3, 2-4
Register Description	2-4
Protection Against Accidental Writes to OSCCON	7-6
PSV Configuration	4-8
PSV Mapping with X and Y Data Spaces	4-8
Timing	4-10
Instruction Stalls	4-10
Using PSV in a REPEAT Loop	4-10
Pulse Width Modulation Mode	14-18
Duty Cycle	14-20
Period	14-19
With FAULT Protection Input Pin	14-19
PWM Duty Cycle Comparison Units	15-20
PWM Fault Pins	15-32
PWM Output and Polarity Control	15-32
PWM Output Override	15-29
PWM Special Event Trigger	15-35
PWM Timebase	15-16
PWM Update Lockout	15-35
<b>Q</b>	
QEI Operation During Power Saving Modes	16-19
Quadrature Decoder	16-9
Quadrature Encoder Interface Interrupts	16-17
<b>R</b>	
R/W bit	21-35, 1-4
Read-After-Write Dependency Rules	2-35
Reading A/D Result Buffer (12-bit)	18-30
Reading and Writing 16-bit Timer Module Registers	12-15
Reading and Writing into 32-bit Timers	12-21
Real-Time Operating System (RTOS)	25-9

## Registers

10-bit A/D Converter Special Function.....	17-53
12-bit A/D Converter Special Function.....	18-34
6-Output PWM Module .....	15-39
8-Output PWM Module .....	15-38
A/D Input Scan Select (ADCSSL - 12 bit).....	18-9
A/D Input Scan Select (ADCSSL).....	17-10
ADCHS (A/D Input Select) Register.....	17-9, 18-8
ADCHS A/D Input Select .....	17-9
ADCHS A/D Input Select (12-bit).....	18-8
ADCON1 (A/D Control) Register1.....	17-5, 18-5, 21-11
ADCON1 A/D Control 1 .....	17-5, 17-6
ADCON1 A/D Control 1 (12-bit).....	18-5
ADCON2 (A/D Control) Register2.....	17-7, 18-6
ADCON2 A/D Control 2 .....	17-7
ADCON2 A/D Control 2 (12-bit).....	18-6
ADCON3 (A/D Control) Register3.....	17-8, 18-7
ADCON3 A/D Control 3 .....	17-8
ADCON3 A/D Control 3 (12-bit).....	18-7
ADPCFG (A/D Port Configuration) Register .....	17-10, 18-9
ADPCFG A/D Port Configuration .....	17-10
ADPCFG A/D Port Configuration (12-bit) .....	18-9
CiCTRL CAN Module Control and Status Register.	23-3
CiRXFnEIDH Acceptance Filter n	
Extended Identifier High .....	23-12
CiRXMnEIDH Acceptance Filter Mask n	
Extended Identifier High .....	23-14
CNEN1 (Input Change Notification Interrupt	
Enable 1) .....	11-7
CNEN2 (Input Change Notification Interrupt	
Enable 2) .....	11-7
CNPU1 (Input Change Notification Pull-up	
Enable 1) .....	11-8
CNPU2 (Input Change Notification Pull-up	
Enable 2) .....	11-8
Control and Status .....	16-4
Control Registers .....	15-4
CORCON (Core Control) .....	2-14, 6-15
DCICON1 .....	22-3
DCICON2 .....	22-4
DCICON3 .....	22-5
DCISTAT .....	22-6
DFLTCON Digital Filter Control .....	16-7
DTCN1 Dead Time Control 1 .....	15-9
DTCN2 Dead Time Control 2 .....	15-10
FBORPOR BOR and POR Device Configuration .....	15-15
FGS (General Code Segment Configuration) .....	24-6
FLTACON Fault A Control .....	15-11
FLTBCON Fault B Control .....	15-12
FOSC (Oscillator Configuration) .....	24-3
I2CSTAT (I <sup>2</sup> C Status) Register .....	21-9, 21-10, 21-11
ICxCON (Input Capture x Control) .....	13-3
IEC0 (Interrupt Enable Control 0) .....	6-24
IEC1 (Interrupt Enable Control 1) .....	6-26
IEC2 (Interrupt Enable Control 2) .....	6-28, 6-29
IFS0 (Interrupt Flag Status 0) .....	6-18
IFS1 (Interrupt Flag Status 1) .....	6-20
IFS2 (Interrupt Flag Status 2) .....	6-22, 6-23
INTCON1 (Interrupt Control 1) .....	6-16
INTCON2 (Interrupt Control 2) .....	6-17
IPC0 (Interrupt Priority Control 0) .....	6-30
IPC1 (Interrupt Priority Control 1) .....	6-31
IPC10 (Interrupt Priority Control 10) .....	6-40
IPC11 (Interrupt Priority Control 11) .....	6-41
IPC2 (Interrupt Priority Control 2) .....	6-32
IPC3 (Interrupt Priority Control 3) .....	6-33
IPC4 (Interrupt Priority Control 4) .....	6-34

IPC5 (Interrupt Priority Control 5) .....	6-35
IPC6 (Interrupt Priority Control 6) .....	6-36
IPC7 (Interrupt Priority Control 7) .....	6-37
IPC8 (Interrupt Priority Control 8) .....	6-38
IPC9 (Interrupt Priority Control 9) .....	6-39
MODCON (Modulo and Bit-Reversed Addressing Control) .....	3-19
NVMADR (Non-Volatile Memory Address) .....	5-8
NVMCON (Non-Volatile Memory Control) .....	5-7
NVMKEY (Non-Volatile Memory Key) .....	5-8
OCxCON (Output Compare x Control) .....	14-3
OSCCON (Oscillator Control) .....	7-7
OVDCON Override Control .....	15-13
PDC1 PWM Duty Cycle 1 .....	15-13
PDC2 PWM Duty Cycle 2 .....	15-14
PDC3 PWM Duty Cycle 3 .....	15-14
PDC4 PWM Duty Cycle 4 .....	15-15
PTCON PWM Timebase Control .....	15-5
PTMR PWM Timebase .....	15-6
PTPER PWM Timebase Period .....	15-6
PWMCON1 PWM Control 1 .....	15-7
PWMCON2 PWM Control 2 .....	15-8
QEI Special Function .....	16-20
QEICON QEI Control .....	16-5, 16-6
RCON (RESET Control) .....	8-3, 9-4
RSCON .....	22-7
SEVTCMP Special Event Compare .....	15-7
SPIxCON (SPI Control) .....	23-5, 23-6, 23-7, 23-8, 23-9, 23-10, 23-11, 23-12, 23-13, 23-14, 23-15, 23-16, 23-17, 23-18, 23-19, 23-20
SR (CPU Status) .....	2-12
SR (Status in CPU) .....	6-15
TSCON .....	22-7
TxCON (Timer Control for Type A Time Base) .....	12-6
TxCON (Timer Control for Type B Time Base) .....	12-7
TxCON (Timer Control for Type C Time Base) .....	12-8
UxBAUD (UARTx Baud Rate) .....	19-7
UxMODE (UARTx Mode) .....	19-3
UxRXREG (UARTx Receive) .....	19-6
UxSTA (UARTx Status and Control) .....	19-4
UxTXREG (UARTx Transmit - Write Only) .....	19-6
XBREV (X Write AGU Bit-Reversal Addressing Control) .....	3-22
XMODEEND (X AGU Modulo Addressing End) .....	3-20
XMODSRT (X AGU Modulo Addressing Start) .....	3-20
YMODEEND (Y AGU Modulo Addressing End) .....	3-21
YMODSRT (Y AGU Modulo Addressing Start) .....	3-21
Reset	
Design Tips .....	8-17
Illegal Opcode .....	8-9
Trap Conflict .....	8-9
Uninitialized W Register .....	8-9
RESET Sequence .....	6-2
Returning From Interrupt .....	6-13
Round Logic .....	2-25
Run-Time Self Programming (RTSP) .....	5-9
FLASH Operations .....	5-9
Operation .....	5-9
S	
Saturation and Overflow Modes .....	2-24
Selecting A/D Conversion Clock .....	17-14
Selecting Analog Inputs for Sampling .....	17-15
Selecting Analog Inputs for Sampling (12-bit) .....	18-13
Selecting the A/D Conversion Clock (12-bit) .....	18-12
Selecting the Voltage Reference Source .....	17-13
Selecting the Voltage Reference Source (12-bit) .....	18-11

Setup for Continuous Output Pulse Generation .....	14-15	Timing Diagrams .....	
Shadow Registers .....	2-6	Brown-out Situations .....	8-8
DO Loop .....	2-7	Clock Transition .....	7-22
PUSH.S and POP.S .....	2-7	Clock/Instruction Cycle .....	7-4
Simple Capture Events .....	13-4	Data Space Access .....	2-35, 3-6
SLEEP and IDLE Modes Operation .....	17-52	Dead-Time .....	15-26
SLEEP and IDLE Modes Operation (12-bit) .....	18-33	Device RESET Delay, Crystal + PLL Clock Source, PWRT Disabled .....	8-13
SLEEP Mode .....	10-2	Device RESET Delay, Crystal + PLL Clock Source, PWRT Enabled .....	8-14
and FSCM Delay .....	10-3	Device RESET Delay, EC + PLL Clock, PWRT Enabled .....	8-15
Clock Selection on Wake-up from .....	10-2	Device RESET Delay, EC or RC Clock, PWRT Disabled .....	8-16
Delay on Wake-up from .....	10-3	Dual Compare Mode .....	14-10
Delay Times for Exit .....	10-3	Dual Compare Mode (Continuous Output Pulse, PR2 = OCxRS) .....	14-14, 14-15
Wake-up from on Interrupt .....	10-4	Dual Compare Mode (Single Output Pulse, OCxRS > PR2) .....	14-10
Wake-up from on RESET .....	10-4	Edge Detection Mode .....	13-8
Wake-up from on Watchdog Time-out .....	10-4	Gated Timer Mode Operation .....	12-13
Wake-up from with Crystal Oscillator or PLL .....	10-3	Interrupt Timing During a Two-Cycle Instruction .....	6-12
Slow Oscillator Start-up .....	10-3	Interrupt Timing for Timer Period Match .....	12-14
Soft Traps .....	6-6	Interrupt Timing, Interrupt Occurs During 1st Cycle of a Two-Cycle Instruction .....	6-12
Arithmetic Error (Level 11) .....	6-7	POR Module for Rising VDD .....	8-6
Software RESET Instruction (SWR) .....	8-7	Postscaler Update .....	7-21
Software Stack .....		PWM Output .....	14-18, 14-21
Examples .....	2-9	Reception with Address Detect (ADDEN = 1) .....	19-19
Pointer .....	2-8	Return From Interrupt .....	6-13
Pointer Overflow .....	2-10	Simple Capture Event, Time-base Prescaler = 1:1 .....	13-5
Pointer Underflow .....	2-10	Simple Capture Event, Time-base Prescaler = 1:4 .....	13-5
W14 Stack Frame Pointer .....	2-10	Single Compare Mode (Force OCx Low on Compare Match Event) .....	14-6
Special Conditions for Interrupt Latency .....	6-13	Single Compare Mode (Set OCx High on Compare Match Event) .....	14-5
Special Features for Device Emulation .....	15-37	Single Compare Mode (Toggle Output on Compare Match Event, PR2 = OCxR) .....	14-7
Special Function Register RESET States .....	8-16	Single Compare Mode (Toggle Output on Compare Match Event, PR2 > OCxR) .....	14-7
Specifying How Conversion Results are Written .....		SPI Mode Timing (No SS Control) .....	20-12, 20-13
Into Buffer .....	17-31	Transmission (8-bit or 9-bit Data) .....	19-13
Specifying How Conversion Results are Written .....		Transmission (Back to Back) .....	19-13
into Buffer (12-bit) .....	18-21	UART Reception .....	19-17
SSPOV .....	21-19	UART Reception with Receive Overrun .....	19-17
<b>T</b>		TRIS (Data Direction) Registers .....	11-3
Table Instruction Operation .....	5-2	Tuning the Oscillator Circuit .....	7-10
TCP/IP Protocol Stack .....	25-10	Type A Timer .....	12-3
Third Party C Compilers .....	25-6	Type B Timer .....	12-4
Third Party Hardware/Software Tools and .....		Type C Timer .....	12-5
Application Libraries .....	25-6		
Time-base for Input Capture/Output Compare .....	12-22	<b>U</b>	
Timer as an External Interrupt Pin .....	12-22		
Timer Interrupts .....	12-14	<b>UART</b> .....	
Timer Modes of Operation .....	12-9	ADDEN Control Bit .....	19-18
32-bit Timer .....	12-18	Alternate I/O Pins .....	19-10
Synchronous Counter Using External Clock Input .....	12-10	Associated Registers .....	19-22
Timer Mode .....	12-9	Baud Rate Generator .....	19-8
Type A Timer Asynchronous Counter Mode .....		Configuration .....	19-10
Using External Clock Input .....	12-11	Control Registers .....	19-3
Timer Modules .....		Design Tips .....	19-23
Associated Special Function Registers .....	12-23	Disabling .....	19-10
Timer Operation in Power Saving States .....	12-21	Enabling .....	19-10
Timer Operation Modes .....			
Gated Time Accumulation .....	12-12		
with Fast External Clock Source .....	12-12		
Timer Prescalers .....	12-14		
Timer Selection .....	13-4		
Timer Variants .....	12-3		
Timers .....			
Design Tips .....	12-24		
Related Application Notes .....	12-25		
Revision History .....	12-26		

Other Features.....	19-21
Auto Baud Support .....	19-21
Loopback Mode .....	19-21
Operation During CPU SLEEP and IDLE Modes.....	19-21
Receiver.....	19-14
Buffer (UxRXB).....	19-14
Error Handling.....	19-14
Interrupt .....	19-15
Setup for Reception .....	19-17
Related Application Notes.....	19-24
Setup for 9-bit Transmit .....	19-18
Transmitter.....	19-11
Buffer (UxTXB).....	19-12
Interrupt .....	19-12
Setup .....	19-13
Transmission of Break Characters .....	19-14
Using for 9-bit Communication.....	19-18
UART Autobaud Support .....	13-9
Uninitialized W Register RESET.....	2-7
USART	
Initialization .....	19-20
Introduction .....	19-2
Receiving Break Characters .....	19-19
Revision History .....	19-25
Setup for 9-bit Reception Using Address	
Detect Mode .....	19-18
Using QEI as Alternate 16-bit Timer/Counter .....	16-16
Using Table Read Instructions .....	5-3
Byte Mode.....	5-3
Word Mode .....	5-3
Using Table Write Instructions .....	5-4
Byte Mode.....	5-5
Holding Latches .....	5-4
Word Mode .....	5-4
Using the RCON Status Bits .....	8-10
V	
V.22/V.22bis and V.32 Specification.....	25-11
W	
Wake-up from SLEEP and IDLE.....	6-10
Watchdog Time-out Reset (WDTR).....	8-7
Watchdog Timer .....	10-5
Enabling and Disabling.....	10-6
Operation.....	10-6
Operation in SLEEP and IDLE Modes .....	10-7
Period Selection .....	10-6
Prescalers .....	10-6
Resetting .....	10-7
Software Controlled .....	10-6
WCOL.....	21-18, 21-19, 21-21
WDT and Power Saving Modes	
Design Tips.....	10-8
Related Application Notes .....	10-9
Revision History .....	10-10
Working Register Array.....	2-6
W Register Memory Mapping .....	2-6
W Registers and Byte Mode Instructions .....	2-6
W0 and File Register Instructions .....	2-6
Writing to Device Configuration Registers .....	5-13
Write Algorithm .....	5-13



# MICROCHIP

## 全世界の販売及びサービス拠点

### AMERICAS

**Corporate Office**  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7200  
Fax: 480-792-7277  
Technical Support:  
<http://support.microchip.com>  
Web Address:  
[www.microchip.com](http://www.microchip.com)

**Atlanta**

Alpharetta, GA  
Tel: 770-640-0034  
Fax: 770-640-0307

**Boston**

Westborough, MA  
Tel: 774-760-0087  
Fax: 774-760-0088

**Chicago**

Itasca, IL  
Tel: 630-285-0071  
Fax: 630-285-0075

**Dallas**

Addison, TX  
Tel: 972-818-7423  
Fax: 972-818-2924

**Detroit**

Farmington Hills, MI  
Tel: 248-538-2250  
Fax: 248-538-2260

**Kokomo**

Kokomo, IN  
Tel: 765-864-8360  
Fax: 765-864-8387

**Los Angeles**

Mission Viejo, CA  
Tel: 949-462-9523  
Fax: 949-462-9608

**San Jose**

Mountain View, CA  
Tel: 650-215-1444  
Fax: 650-961-0286

**Toronto**

Mississauga, Ontario,  
Canada  
Tel: 905-673-0699  
Fax: 905-673-6509

### ASIA/PACIFIC

**Australia - Sydney**  
Tel: 61-2-9868-6733  
Fax: 61-2-9868-6755

**China - Beijing**  
Tel: 86-10-8528-2100  
Fax: 86-10-8528-2104

**China - Chengdu**  
Tel: 86-28-8676-6200  
Fax: 86-28-8676-6599

**China - Fuzhou**  
Tel: 86-591-8750-3506  
Fax: 86-591-8750-3521

**China - Hong Kong SAR**  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**China - Qingdao**  
Tel: 86-532-8502-7355  
Fax: 86-532-8502-7205

**China - Shanghai**  
Tel: 86-21-5407-5533  
Fax: 86-21-5407-5066

**China - Shenyang**  
Tel: 86-24-2334-2829  
Fax: 86-24-2334-2393

**China - Shenzhen**  
Tel: 86-755-8203-2660  
Fax: 86-755-8203-1760

**China - Shunde**  
Tel: 86-757-2839-5507  
Fax: 86-757-2839-5571

**China - Wuhan**  
Tel: 86-27-5980-5300  
Fax: 86-27-5980-5118

**China - Xian**  
Tel: 86-29-8833-7250  
Fax: 86-29-8833-7256

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-4182-8400  
Fax: 91-80-4182-8422

**India - New Delhi**  
Tel: 91-11-5160-8631  
Fax: 91-11-5160-8632

**India - Pune**  
Tel: 91-20-2566-1512  
Fax: 91-20-2566-1513

**Japan - Yokohama**  
Tel: 81-45-471-6166  
Fax: 81-45-471-6122

**Korea - Gumi**  
Tel: 82-54-473-4301  
Fax: 82-54-473-4302

**Korea - Seoul**  
Tel: 82-2-554-7200  
Fax: 82-2-558-5932 or  
82-2-558-5934

**Malaysia - Penang**  
Tel: 60-4-646-8870  
Fax: 60-4-646-5086

**Philippines - Manila**  
Tel: 63-2-634-9065  
Fax: 63-2-634-9069

**Singapore**  
Tel: 65-6334-8870  
Fax: 65-6334-8850

**Taiwan - Hsin Chu**  
Tel: 886-3-572-9526  
Fax: 886-3-572-6459

**Taiwan - Kaohsiung**  
Tel: 886-7-536-4818  
Fax: 886-7-536-4803

**Taiwan - Taipei**  
Tel: 886-2-2500-6610  
Fax: 886-2-2508-0102

**Thailand - Bangkok**  
Tel: 66-2-694-1351  
Fax: 66-2-694-1350

### EUROPE

**Austria - Wels**  
Tel: 43-7242-2244-399  
Fax: 43-7242-2244-393

**Denmark - Copenhagen**  
Tel: 45-4450-2828  
Fax: 45-4485-2829

**France - Paris**  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

**Germany - Munich**  
Tel: 49-89-627-144-0  
Fax: 49-89-627-144-44

**Italy - Milan**  
Tel: 39-0331-742611  
Fax: 39-0331-466781

**Netherlands - Drunen**  
Tel: 31-416-690399  
Fax: 31-416-690340

**Spain - Madrid**  
Tel: 34-91-708-08-90  
Fax: 34-91-708-08-91

**UK - Wokingham**  
Tel: 44-118-921-5869  
Fax: 44-118-921-5820