PIC17C75X

ハイパフォーマンス 8-Bit CMOS EPROM マイクロコントローラ

このデータシートには下記のデバイスが含まれています。

- PIC17C752
- PIC17C756

マイクロコントローラの特徴：

- 覚える必要があるのは 58 個のシングルワード命令のみ
- 2 サイクルのプログラム分岐とテーブル リード / ライトを除いて、全てシングルサイクル命令 (121ns)
- 動作速度：
  - DC - 33 MHz クロック入力
  - DC - 121 ns 命令サイクル

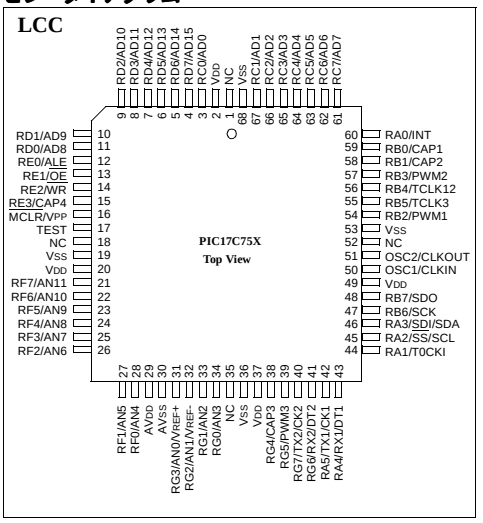
デバイス	メモリ	
	プログラム (x16)	データ (x8)
PIC17C752	*8K*	*454*
PIC17C756	*16K*	*902*

- ハードウェアマルチプライア
- 割込み機能
- 16 レベルディープハードウェア･スタック
- ダイレクト (直接)、インダイレクト (間接)、リラティブ (相対) の各アドレスモード
- 内部 / 外部プログラムメモリの実行
- 64K × 16 プログラムメモリスペースのアドレス可能

周辺機能の特徴：

- 個々に方向制御出来る 50 ピン I/O
- 直接 LED ドライブ用大シンク / ソース電流 -RA2 と RA3 はオープンドレインで高電圧 (12V)、大電流 (60mA) の I/O ピン
- ４個のキャプチャ入力ピン - キャプチャは 16bit で最大分解能 121ns
- ３個の PWM 出力 - PWM の分解能は 1 ～ 10bit
- TMR0：８ bit のプログラム可能なプリスケーラを持った 16bit のタイマ / カウンタ
- TMR1: 8-bit タイマ / カウンタ
- TMR2: 8-bit タイマ / カウンタ
- TMR3: 16-bit タイマ / カウンタ
- ２個のユニバーサル同期 / 非同期レシーバとトランスミッタ (USART/SCI)- 独立したボーレートジェネレータ
- 10-bit、12 チャンネルのアナログ - ディジタル･コンバータ
- SPI™ と I²C™ モードを持った（I²C マスタモードを含む）同期シリアルポート (SSP)

ピン・ダイアグラム



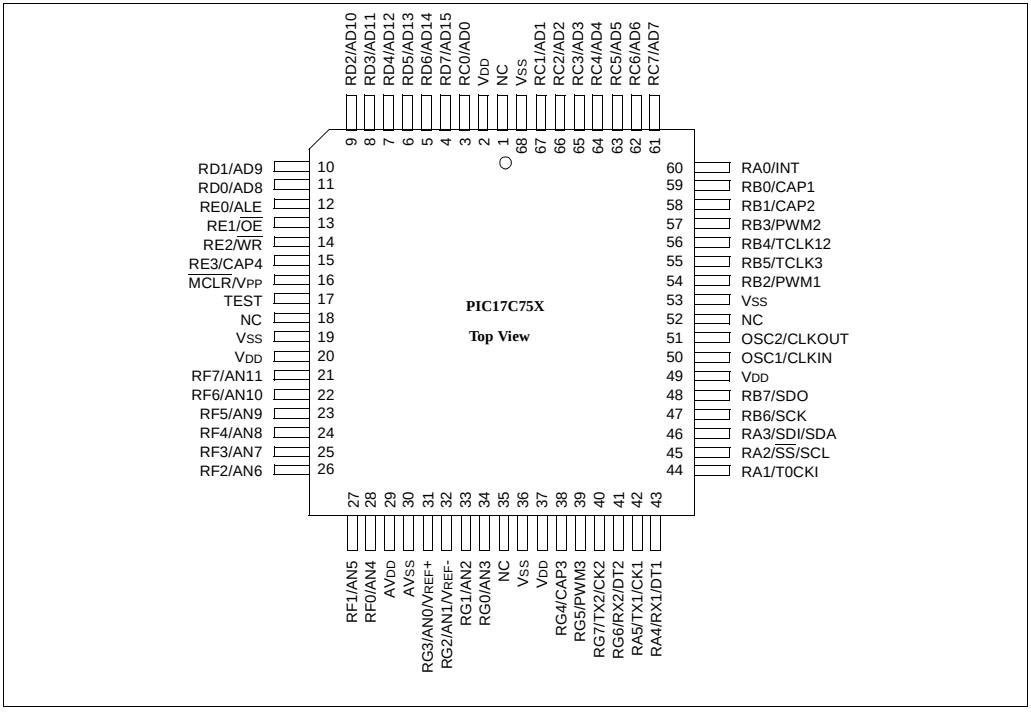
マイクロコントローラの特殊機能：

- パワーオンリセット (POR)、パワーアップタイマ (PWRT)、オシレータスタートアップタイマ (OST)
- 確実な動作のために専用の RC オシレータを内蔵したウォッチドッグタイマ (WDT)
- ブラウンアウトリセット
- コードプロテクション
- 消費電力を節約する SLEEP モード
- 選択可能なオシレータオプション

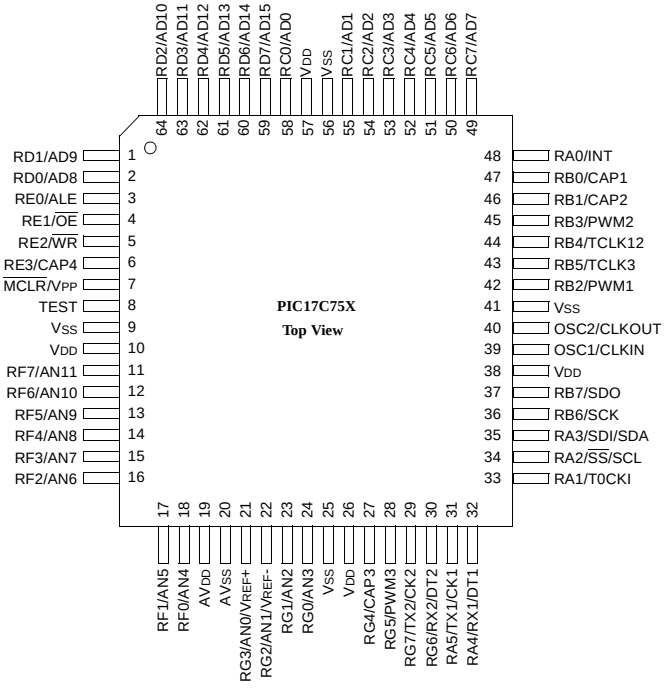
CMOS の技術：

- 低消費電力、高速 CMOS EPROM 技術
- 完全スタティック設計
- 幅広い動作電圧範囲 (2.5V-6.0V)
- 商業、工業用の温度範囲
- 低消費電力：
  - ＜ 5mA @ 5V,4MHz
  - 100 µA 標準@ 4.5V,32kHz
  - ＜１ µA 標準スタンバイ電流 @5V

ピン・ダイアグラム（続き）　　68 ピン LCC の PIC17C75X

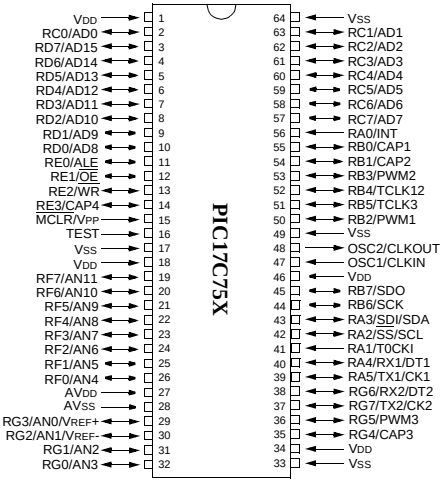


ピン・ダイアグラム（続き） 64 ピン TQ FP の PIC17C75X



Applicable to 14 x 14 mm TQFP

ピン・ダイアグラム（続き） 64 ピン Y-SHRINK DIP の PIC17C75X



目次

お客様へ

当社では、当社製品や解説書の品質を高めるために常に努力をしております。この解説書も正確を期すために非常に多くの時間を費やしておりますが、少しの見落としがあるかもしれません。もし見落としや間違っている情報にお気づきになられましたら、このデータ・シートの最後にある「読者のご意見」の書式に従って当社までお知らせください。より良い解説書を作るために皆様のご協力に感謝いたします。

1.0 概要

このデータシートは、マイクロコントローラ PIC17CXXX ファミリの中の PIC17C75X グループをカバーしています。このデータシートでは、次のデバイスについて説明されています。

- **PIC17C752**
- **PIC17C756**

PIC17C75X デバイスは、用途の広い PIC17CXXX ファミリの中で、68 本のピンを持ち、EPROM をベースにしたもので、低価格、高性能、CMOS、完全スタティックな 8bit マイクロコントローラです。

すべての PIC16/17 マイクロコントローラは、先進の RISC アーキテクチャを採用しています。PIC17CXXX には、拡張コア機能、16 レベル・ディープ・スタック、複数の内部、外部割込みソースが採用されています。ハーバードアーキテクチャによる命令とデータが分離したバスにより、分離された 8bit 幅データパスと 16bit 幅の命令語が可能になります。2 段の命令パイプラインにより、プログラム分岐 (2 サイクル必要) を除いて、すべての命令がシングルサイクルで実行できます。合計 58 個の命令 (縮小命令セット) が使用可能です。さらに、多くのレジスタセットは、高性能を実現するために利用されるアーキテクチャの革新をもたらします。演算を強調した応用に対しては、すべてのデバイスにシングルサイクル 8X8 のハードウェアマルチプライアがあります。

PIC17CXXX マイクロコントローラは、同クラスの他社製 8bit マイクロコントローラと比べて、一般的に 2:1 のコード圧縮、4:1 のスピード改善を実現します。

PIC17C75X デバイスは最大 902 バイトの RAM と 50 本の I/O ピンを持っています。さらに、 PIC17C75X には、下記のように高性能の応用に役立つ周辺機能が充実しています。

- **4 個のタイマ / カウンタ**
- **4 個のキャプチャ入力**
- **3 個の PWM 出力**
- **2 個の独立したユニバーサル同期 / 非同期レシーバとトランスミッタ (USART)**
- **1 個の A/D 変換器 (12 チャンネル、10bit の分解能)**
- **1 個の同期シリアルポート (SPI と I^2C w/ マスタモード)**

これらの特殊機能は外部コンポーネントを減らし、それにより、コストの削減、システムの信頼性向上、消費電力の低減が可能になります。

4 種類のオシレータオプションがあり、低価格化が可能なシングルピン構成の RC オシレータ、低周波数クリスタル用で消費電力を最小限におさえる LF オシレータ、標準クリスタルの XT、外部クロック入力用の EC の中から選択できます。

SLEEP(パワーダウン) モードによりさらに電力の節約ができます。複数の外部・内部割込みとデバイスのリセットにより SLEEP からのウェークアップができます。

専用の内蔵 RC オシレータを持った信頼性の高いウォッチドッグタイマによりソフトウエアを誤動作から保護することができます。

デバイスの動作モードには 4 種類の設定オプションがあります。

- **マイクロプロセッサ**
- **マイクロコントローラ**
- **拡張マイクロコントローラ**
- **保護マイクロコントローラ**

マイクロプロセッサと拡張マイクロコントローラモードでは、外部プログラムメモリを 64K ワードまで持つことが可能です。

ブラウンアウトリセットの回路もデバイスに含まれています。これにより、V_{DD} デバイスがブラウンアウト電圧トリップ点 (BV_{DD}) 以下に落ちた場合でもデバイスのリセットが可能です。そのチップは、V_{DD} が BV_{DD} 以上に上がるまでブラウンアウトリセット状態のままです。

表 1-1 が PIC17CXXX デバイスの特徴です。

UV 消去可能な CERQUAD パッケージ版 (PLCC と互換性あり) はコード開発時に、コストの点からワンタイムプログラマブル (OTP) 版は量産時に最適です。

PIC17C75X は、複雑なソフトウェアプログラムの実行を非常に速く必要とする応用に完全に適応します。これらの応用範囲は、精密なモータ制御や工業用プロセス制御から車載用、計器類、テレコムの応用まで含んでいます。

EPROM 技術により、応用プログラム (独特なセキュリティコード、コンビネーション、モデルナンバー、パラメータ保存など) を速く便利にカスタマイズできます。小型表面実装パッケージのオプション (ダイスセールを含む) により、高性能を必要とし、スペースに制限のある応用にも PIC17C75X は理想的です。

インサーキットシリアルプログラミング(ISP)の特徴は、

- **製造工程の最終段階の 1 つとして、ソフトウェアコードをプログラミングするという融通性があります。**

低価格ながら、高スピードの実行、利用価値の高い周辺機能、フレキシブルな I/O、低消費電力と、PIC17C75X は幅広い範囲の埋め込み制御の応用に理想的です。

1.1 ファミリと上位互換性

マイクロコントローラの PIC17CXXX ファミリは、PIC16C5X と PIC16CXX ファミリを越えたアーキテクチャの改良版です。これらの改良点により、ソフトウェアとハードウェアの要求にもデバイスは前より効率よく対応できます。改良点と変更点の詳細なリストは付録 A をご覧ください。PIC16C5X または PIC16CXX 用に書かれたコードは、PIC17CXXX デバイスに簡単に移植できます (付録 B 参照)。

1.2 開発サポート

PIC17CXXX ファミリには完全機能のマクロアセンブラ、ソフトウェアシュミレータ、インサーキットエミュレータ、ユニバーサルプログラマ、“C” コンパイラ、ファジーロジック・サポートツールがサポートされています。さらに詳しい情報は 19.0 章を参照してください。

表 1-1: PIC17CXXX ファミリのデバイス

Features		PIC17CR42	PIC17C42A	PIC17C43	PIC17CR43	PIC17C44	PIC17C752	PIC17C756
最大動作周波数		33 MHz	33 MHz	33 MHz	33 MHz	33 MHz	33 MHz	33 MHz
動作電源電圧範囲		2.5 - 6.0V	2.5 - 6.0V	2.5 - 6.0V	2.5 - 6.0V	2.5 - 6.0V	3.0 - 6.0V	3.0 - 6.0V
プログラムメモリ (x16)	(EPROM)	-	16 K	4K	-	8K	8K	16K
	(ROM)	2K	-	-	4K	-	-	-
データメモリ (bytes)		232	232	454	454	454	454	902
ハードウェアマルチプライア (8 x 8)		Yes	Yes	Yes	Yes	Yes	Yes	Yes
タイマ 0 (16-bit + 8-bit ポストスケーラ)		Yes	Yes	Yes	Yes	Yes	Yes	Yes
タイマ 1 (8-bit)		Yes	Yes	Yes	Yes	Yes	Yes	Yes
タイマ 2 (8-bit)		Yes	Yes	Yes	Yes	Yes	Yes	Yes
タイマ 3 (16-bit)		Yes	Yes	Yes	Yes	Yes	Yes	Yes
キャプチャ入力 (16-bit)		2	2	2	2	2	4	4
PWM 出力 (10-bit まで)		2	2	2	2	2	3	3
USART/SCI		1	1	1	1	1	2	2
A/D チャンネル (10-bit)		-	-	-	-	-	12	12
SSP (SPI/I^2C w/ マスターモード)		-	-	-	-	-	Yes	Yes
パワーオンリセット		Yes	Yes	Yes	Yes	Yes	Yes	Yes
ウォッチドッグタイマ		Yes	Yes	Yes	Yes	Yes	Yes	Yes
外部割込み		Yes	Yes	Yes	Yes	Yes	Yes	Yes
インターラプトソース		11	11	11	11	11	18	18
コード保護		Yes	Yes	Yes	Yes	Yes	Yes	Yes
ブラウンアウトリセット		-	-	-	-	-	Yes	Yes
インサーキットシリアルプログラミング		-	-	-	-	-	Yes	Yes
I/O ピン数		33	33	33	33	33	50	50
I/O 最大電流容量	Source	25 mA	25 mA	25 mA	25 mA	25 mA	25 mA	25 mA
	Sink	25 mA(1)	25 mA(1)	25 mA(1)	25 mA(1)	25 mA(1)	25 mA(1)	25 mA(1)
パッケージタイプ		40-pin DIP 44-pin PLCC 44-pin MQFP 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin MQFP 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin MQFP 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin MQFP 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin MQFP 44-pin TQFP	64-pin DIP 68-pin LCC 68-pin TQFP	64-pin DIP 68-pin LCC 68-pin TQFP

注 1: RA2 と RA3 ピンは 60mA までシンクできます。

2.0 デバイスの種類

それぞれのデバイスに、色々な周波数範囲とパッケージの種類があります。アプリケーションと量産の必要に応じて、このデータシートの最後にあるPIC17C75X製品識別システム情報により適当なデバイスの種類を選択できます。ご注文の際は、このデータシートの最後にある“ PIC17C75X製品の識別システム ”を使って、正しい製品番号をご指定ください。デバイスの機能性を検討される場合、メモリの技術や電圧範囲は問題ありません。

3種類のメモリタイプがあり、それらは部品ナンバーの真ん中の文字で記されています。

1. **C** タイプ (例えば PIC17**C**756) これらのデバイスは EPROM タイプのメモリです。
2. **CR** タイプ (例えば PIC17**CR**756) これらのデバイスは ROM タイプのメモリです。
3. **F** タイプ (例えば PIC17**F**756) これらのデバイスはフラッシュタイプのメモリです。

上記のデバイスはすべて標準の電圧レンジで動きます。また拡張した電圧レンジ (及び減少した周波数レンジ) で動くことも可能です。表 2-1 に、個々のデバイスに対して可能なメモリタイプと電圧レンジ指定を示します。これらの指定子を太字体で表わします。

表 2-1: デバイスのメモリの種類

Memory Type	Voltage Range	
	Standard	Extended
EPROM	PIC17**C**XXX	PIC17**LC**XXX
ROM	PIC17**CR**XXX	PIC17**LCR**XXX
Flash	PIC17**F**XXX	PIC17**LF**XXX

注意： 個々のデバイスに対してすべてのメモリ技術が利用できるわけではありません。.

2.1 UV 消去デバイス

紫外線消去タイプは、CERQUAD パッケージで提供されており、試作開発用とパイロットプログラムに最適です。

紫外線消去タイプは消去することができ、どのコンフィギュレーションモードにも再プログラムすることができます。Microchip 社の PIC17C75X プログラミング、およびサードパーティのプログラマも利用できます。詳細についてはサードパーティガイドを参照してください。

2.2 ワンタイムプログラマブル (OTP) デバイス

OTP デバイスは、頻繁なコード変更と更新が予想されるお客様に有益です。

プラスチックパッケージの OTP デバイスは、1 度のみプログラムするユーザに限られます。プログラムメモリに加え、コンフィギュレーションビットもプログラムする必要があります。

2.3 クイック・ターンアラウンド・プロダクション (QTP) デバイス

Microchip 社は、工場生産注文のための QTP プログラミングサービスを行なっています。このサービスは、中規模の量産ユニットで自社工場でプログラムを行ないたくない、コードパターンが安定しているユーザが利用できます。このデバイスは OTP デバイスと同等ですが、すべての EPROM ロケーションとコンフィギュレーションオプションが工場でプログラムされます。実際のコードと試作の検査手順は、出荷前に行なわれます。詳細については、お近くの Microchip Technology 社の販売店にお問い合わせください。

2.4 シリアル・クイック・ターンアラウンド・プロダクション (SQTP^{SM}) デバイス

Microchip 社は、各デバイスのシリアル番号を異なったロケーションにプログラムする特定ユーザに対する特殊なプログラミングサービスを行なっています。このシリアル番号には、ランダム / 擬似ランダム / シーケンシャル番号があります。シリアルプログラミングにより、各デバイスがエントリコードやパスワードや ID 番号として使用できる番号を持つことができます。

2.5 リード専用メモリ (ROM) デバイス

Microchip 社は、高品質の製品を低価格のオプションで提供するために、大量生産のパーツのいくつかにマスキングされた ROM バージョンを提供しています。

ROM デバイスはプログラムメモリのスペースに、シリアル番号情報を記憶させることはできません。ROM コードを提出する場合の方法については、近くの販売店にお問い合わせください。

注意： 現在、PIC17C75X デバイスの ROM バージョンはありません。

2.6 フラッシュメモリデバイス

これらのデバイスは電気的に消去されるので、低価格のプラスチックパッケージで提供されています。電気的に消去することで、これらのデバイスは消去され、インサーキットで再プログラムが可能です。製造と同様にプロトタイプの開発やパイロットプログラムでも同じです。

注意： 現在、PIC17C75X　デバイスのフラッシュバージョンはありません。

3.0 アーキテクチャの概要

高性能の PIC17CXXX には、RISC 型マイクロプロセッサに見られる多くのアーキテクチャの特徴があげられます。まず、PIC17CXXX は変更されたハーバードアーキテクチャを使っています。これはプログラムとデータが別のメモリからアクセスされるもので、デバイスはプログラムメモリバスとデータメモリバスを持つことになります。プログラムとデータが同じメモリからフェッチされる (同じバスにアクセスする) 伝統的なフォンノイマンアーキテクチャよりも帯域幅が改善されています。さらにプログラムとデータメモリを分離することにより、8bit 幅のデータワードと異なるサイズの命令が可能になります。PIC17CXXX のオペコードは 16bit 幅で、シングルワード命令を持つことができます。フルの 16bit 幅のプログラムメモリバスは、16bit 命令をシングルサイクルでフェッチします。2 層のパイプラインは、命令のフェッチと実行をオーバーラップします。したがって、プログラム分岐とプログラムとデータメモリ間のデータを転送する 2 つの特別な命令を除いて、すべての命令がシングルサイクル (121ns @33MHz) で実行されます。

PIC17CXXX は、64K × 16 までプログラムメモリスペースのアドレスが可能です。

PIC17C752 は、8K × 16 の内蔵された EPROM プログラムメモリをインテグレイトします。

PIC17C756 は、16K × 16 の EPROM プログラムメモリをインテグレイトします。

プログラムの実行は、マイクロコントローラまたは保護マイクロコントローラモードでは内部のみ、マイクロプロセッサモードでは外部のみ、拡張マイクロコントローラモードでは両方が可能です。拡張マイクロコントローラモードではコード保護はできません。

PIC17CXXX は、そのレジスタファイルやデータメモリを直接的または間接的にアドレスできます。プログラムカウンタ (PC) とワーキングレジスタ (WREG) を含むすべての特殊機能レジスタは、データメモリに配置されます。PIC17CXXX は直交的 (対称的) な命令を持っており、どのようなアドレシングモードを使用してもすべてのレジスタ上での実行が可能になります。このような対称的な性質と “ 特定の制約条件 ” がないために、PIC17CXXX のプログラミングが簡単で効率的にできます。さらに、学習効率が極めて向上します。

PIC16CXX ファミリのアーキテクチャの拡張版である PIC17CXXX ファミリの 1 つでは、2 個のファイルレジスタが 2 個のオペランド命令で使用できます。これにより、データが WREG レジスタを通らずに、2 個のレジスタ間を直接動くことが可能です。このようにパフォーマンスを向上し、プログラムメモリの使用を減らします。

PIC17CXXX デバイスには、8bitALU とワーキングレジスタが内蔵されています。ALU は、汎用演算ユニットです。これによってワーキングレジスタデータとすべてのレジスタファイルの間での演算とブール機能が実行できます。

ALU は 8bit 幅で、加算、減算、シフトと論理演算が可能です。指示がない限り、演算の実行は 2 の補数で行なわれます。

WREG レジスタは、ALU の実行に使われる 8bit のワーキングレジスタです。

PIC17C75X のデバイスはすべて、8 × 8 のハードウェアマルチプライアを持っています。このマルチプライアはシングルサイクルで 16bit の数値を発生します。

実行される命令によっては、ALU が ALUSTA レジスタのキャリ (C)、ディジットキャリ (DC)、ゼロ (Z) の各ビットの値に影響を与えます。減算命令では、C と DC ビットは、borrow 出力ビットと digit borrow 出力ビットとして扱われます。一例として SUBLW と SUBWF 命令をご覧ください。

ALU は符号付き演算は実行しませんが、オーバーフロービット (OV) は、符号付き演算をインプリメントするために使用できます。符号付き演算は、マグニチュードとサインビットからできています。オーバーフロービットはマグニチュードがオーバーフローし、サインビットの状態を変化させるかどうかを示します。それは符号付きオペレーションの結果が 128(7Fh) 以上か、-127(FFh) 以下かどうかということです。1 バイト以上が使用された場合、符号付き演算は、7 bit の値 (マグニチュード) 以上を持つことができます。オーバーフロービットを使う場合、ALU での値の bit 6 (MSb のマグニチュード) と bit 7 (サインビット) で動くだけです。つまりオーバーフロービットは、例えばマグニチュードが 11bit である符号付き演算をインプリメントしようとする場合には有用ではありません。符号付き演算の値が 7 bit 以上 (15-、24-、31-bit) なら、アルゴリズムは、低い順番のバイトがオーバーフローステータスビットを無視することを確実にしなければなりません。

正しい操作を確実に行なうために、符号付き数字を足したり引いたりする時は注意が必要です。例 3-1 に、符号付きでない機械で動く ALU で符号付き演算を行なう時に考慮すべき項目を示します。

例 3-1: 符号付き演算

Hex Value	Signed Value Math	Unsigned Value Math
FFh	-127	255
+ 01h	+ 1	+ 1
= ?	= -126 (FEh)	= 0 (00h); Carry bit = 1

```
Signed math requires the result to be FEh
(-126). This would be accomplished by
subtracting one as opposed to adding one.
```

簡単なブロック図を図 3-1 に示します。デバイスピンの説明は表 3-1 に示します。

図 3-1: PIC17C75X のブロック図

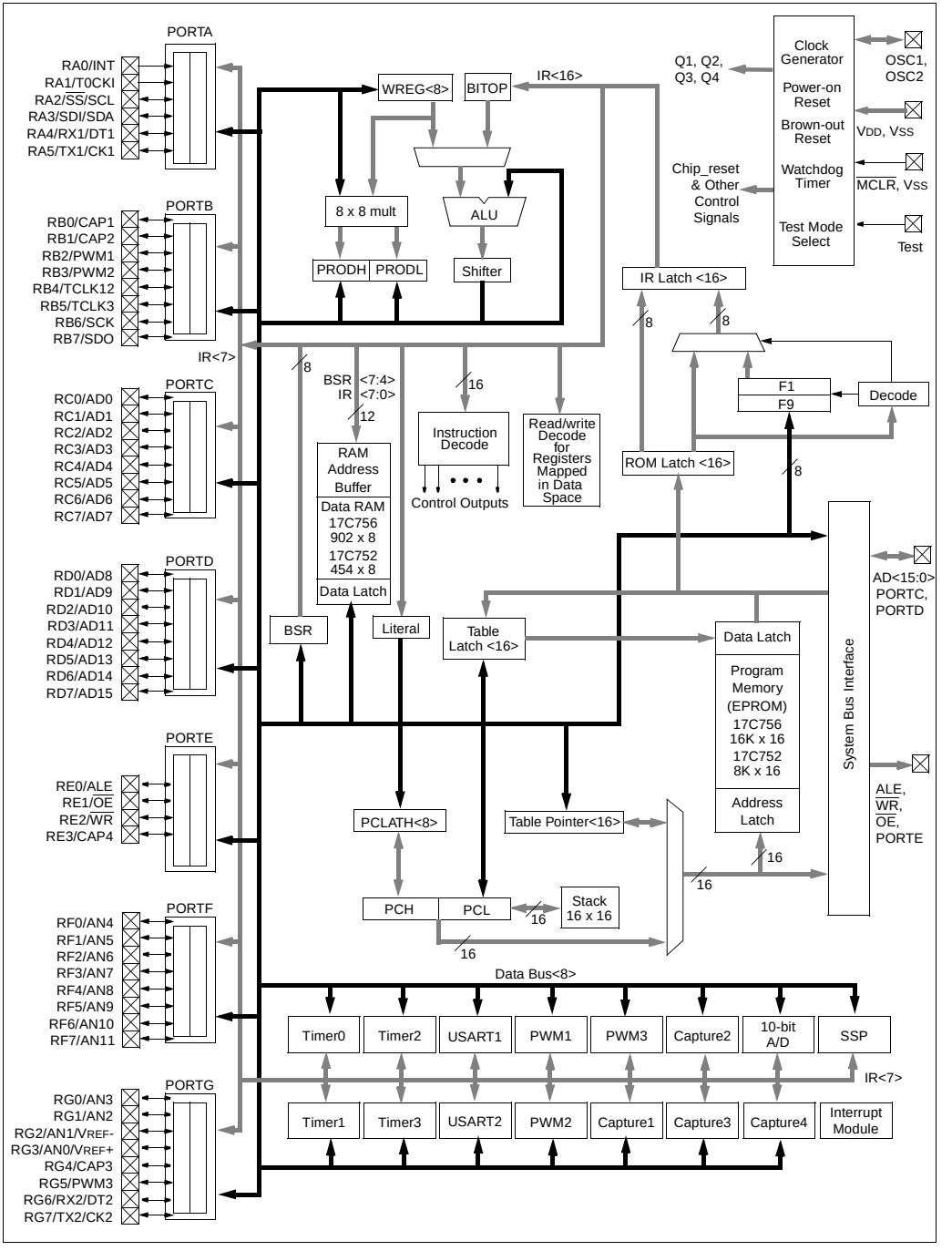


表 3-1: ピンアウトの説明

ピンの名称	DIP No.	PLCC No.	TQFP No.	I/O/P Type	Buffer Type	説明
OSC1/CLKIN	47	50	39	I	ST	クリスタル / レゾネータまたは RC オシレータモードでのオシレータ入力。外部クロックモードでの外部クロック入力。
OSC2/CLKOUT	48	51	40	O	—	オシレータ出力。クリスタルオシレータモードでクリスタルあるいはレゾネータと接続。RC オシレータまたは外部クロックモードでは、OSC2 ピンは、OSC1 の 1/4 の周波数 (Fosc/4) を持つ CLKOUT を出力し、命令サイクルの割合を表示。
$\overline{MCLR}$/VPP	15	16	7	I/P	ST	マスタクリア (リセット) 入力またはプログラム電圧 (Vpp) 入力。これはチップに対してアクティブローリセット。
						PORTA は、入力のみの RA0 と RA1 を除いて双方向 I/O ポート。
RA0/INT	56	60	48	I	ST	RA0 は外部割込み入力としても選択可能。割込みはポジティブまたはネガティブのエッジで設定可能。
RA1/T0CKI	41	44	33	I	ST	RA1 は外部割込み入力としても選択可能。割込みはポジティブまたはネガティブのエッジで設定可能。RA1 はタイマ 0 タイマ / カウンタへのクロック入力でも選択可能。
RA2/$\overline{SS}$/SCL	42	45	34	I/O	ST	RA2 は SPI 用のスレーブ選択入力または I^2C バス用のクロック入力としても使用可能。高電圧、大電流、オープンドレイン入力 / 出力ポートピン。
RA3/SDI/SDA	43	46	35	I/O	ST	RA3 は SPI 用のデータ入力または I^2C バス用のデータとしても使用可能。高電圧、大電流、オープンドレイン入力 / 出力ポートピン。
RA4/RX1/DT1	40	43	32	I/O †	ST	RA4 は USART1（SCI）非同期受信または USART1（SCI）同期データとしても選択可能。
RA5/TX1/CK1	39	42	31	I/O †	ST	RA5 は USART1（SCI）非同期送信または USART1（SCI）同期クロックとしても選択可能。
						PORTB はソフトウェアで設定可能なウィークプルアップを伴う双方向 I/O ポート。
RB0/CAP1	55	59	47	I/O	ST	RB0 はキャプチャ 1 入力ピンでも可能。
RB1/CAP2	54	58	46	I/O	ST	RB1 はキャプチャ 2 入力ピンでも可能。
RB2/PWM1	50	54	42	I/O	ST	RB2 は PWM1 出力ピンでも可能。
RB3/PWM2	53	57	45	I/O	ST	RB3 は PWM2 出力ピンでも可能。
RB4/TCLK12	52	56	44	I/O	ST	RB4 はタイマ 1 とタイマ 2 への外部クロック入力でも可能。
RB5/TCLK3	51	55	43	I/O	ST	RB5 はタイマ 3 への外部クロック入力でも可能。
RB6/SCK	44	47	36	I/O	ST	RB6 は SPI 用のマスタ / スレーブクロックとしても使用可能。
RB7/SDO	45	48	37	I/O	ST	RB7 は SPI 用のデータ出力としても使用可能。

凡例 : I= 入力のみ O= 出力のみ I/O= 入力 / 出力 P= 電源 ー = 未使用 TTL=TTL 入力 ST= シュミットトリガ入力

† 出力は周辺機能の動作によってのみ利用できます。

表 3-1: ピンアウトの説明

ピンの名称	DIP No.	PLCC No.	TQFP No.	I/O/P Type	Buffer Type	説明
						PORTC は双方向 I/O ポート。
RC0/AD0	2	3	58	I/O	TTL	これはマイクロプロセッサモードまたは拡張マイクロコントローラモードでの 16bit ワイドシステムバスの最小有効バイト (LSB) でもあります。マルチプレックスシステムバス設定では、これらのピンはデータ入力または出力と同様にアドレス出力です。
RC1/AD1	63	67	55	I/O	TTL	
RC2/AD2	62	66	54	I/O	TTL	
RC3/AD3	61	65	53	I/O	TTL	
RC4/AD4	60	64	52	I/O	TTL	
RC5/AD5	58	63	51	I/O	TTL	
RC6/AD6	58	62	50	I/O	TTL	
RC7/AD7	57	61	49	I/O	TTL	
						PORTD は双方向 I/O ポート。
RD0/AD8	10	11	2	I/O	TTL	これはマイクロプロセッサモードまたは拡張マイクロプロセッサモードまたは拡張マイクロコントローラモードでの 16bit システムバスの最大有効バイト (MSB) でもあります。マルチプレックスシステムバス設定では、これらのピンはデータ入力または出力と同様にアドレス出力です。
RD1/AD9	9	10	1	I/O	TTL	
RD2/AD10	8	9	64	I/O	TTL	
RD3/AD11	7	8	63	I/O	TTL	
RD4/AD12	6	7	62	I/O	TTL	
RD5/AD13	5	6	61	I/O	TTL	
RD6/AD14	4	5	60	I/O	TTL	
RD7/AD15	3	4	59	I/O	TTL	
						PORTE は双方向 I/O ポート。
RE0/ALE	11	12	3	I/O	TTL	マイクロプロセッサモードまたは拡張マイクロコントローラモードでは、RE0 はアドレス・ラッチ・イネーブル (ALE) 出力。アドレスは ALE 出力の立ち下がりエッジでラッチが必要。
RE1/$\overline{OE}$	12	13	4	I/O	TTL	マイクロプロセッサまたは拡張マイクロコントローラモードでは、RE1 は出力イネーブル (OE) 制御出力 (アクティブ・ロー)。
RE2/$\overline{WR}$	13	14	5	I/O	TTL	マイクロプロセッサまたは拡張マイクロコントローラモードでは、RE2 はライト・イネーブル ($\overline{WR}$) 制御出力 (アクティブ・ロー)。
RE3/CAP4	14	15	6	I/O	ST	RE3 はキャプチャ 4 入力ピンでも可能。
						PORTF は双方向 I/O ポート。
RF0/AN4	26	28	18	I/O	ST	RF0 はアナログ入力 4 でも可能。
RF1/AN5	25	27	17	I/O	ST	RF1 はアナログ入力 5 でも可能。
RF2/AN6	24	26	16	I/O	ST	RF2 はアナログ入力 6 でも可能。
RF3/AN7	23	25	15	I/O	ST	RF3 はアナログ入力 7 でも可能。
RF4/AN8	22	24	14	I/O	ST	RF4 はアナログ入力 8 でも可能。
RF5/AN9	21	23	13	I/O	ST	RF5 はアナログ入力 9 でも可能。
RF6/AN10	20	22	12	I/O	ST	RF6 はアナログ入力 10 でも可能。
RF7/AN11	19	21	11	I/O	ST	RF7 はアナログ入力 11 でも可能。

凡例 : I= 入力のみ O= 出力のみ I/O= 入力 / 出力 P= 電源 ー = 未使用 TTL=TTL 入力 ST= シュミットトリガ入力

† 出力は周辺機能の動作によってのみ利用できます。

表 3-1: ピンアウトの説明

ピンの名称	DIP No.	PLCC No.	TQFP No.	I/O/P Type	Buffer Type	説明
						PORTG は双方向 I/O ポート。
RG0/AN3	32	34	24	I/O	ST	RG0 はアナログ入力 3 でも可能。
RG1/AN2	31	33	23	I/O	ST	RG1 はアナログ入力 2 でも可能。
RG2/AN1/VREF-	30	32	22	I/O	ST	RG2 はアナログ入力 1 でも可能、または接地基準電圧。
RG3/AN0/VREF+	29	31	21	I/O	ST	RG3 はアナログ入力 0 でも可能、または正極基準電圧。
RG4/CAP3	35	38	27	I/O	ST	RG4 はキャプチャ 3 入力ピンでも可能。
RG5/PWM3	36	39	28	I/O	ST	RG5 は PWM3 出力ピンでも可能。
RG6/RX2/DT2	38	41	30	I/O	ST	RG6 は USART2(SCI) 非同期受信、または USART2(SCI) 同期データとしても選択可能。
RG7/TX2/CK2	37	40	29	I/O	ST	RG7 は USART2(SCI) 非同期送信、または USART2(SCI) 同期クロックとしても選択可能。
TEST	16	17	8	I	ST	テストモード選択制御入力。通常の動作に対して常に VSS に接続。
VSS	17, 33, 49, 64	19, 36,53, 68	9, 25, 41, 56	P		ロジックピンおよび I/O ピン用の接地基準。
VDD	1, 18, 34, 46	2, 20, 37, 49,	10, 26, 38, 57	P		ロジックピンおよび I/O ピン用の電源正極。
AVSS	28	30	20	P		A/D コンバータ用の接地基準。このピンは VSS と同じポテンシャルであることが**必要**。
AVDD	27	29	19	P		A/D コンバータ用の電源正極。このピンは VDD と同じポテンシャルであることが**必要**。
NC	-	1, 18, 35, 52	-			接続せず。これらのピンを接続しないままにしておくこと。

凡例 : I= 入力のみ O= 出力のみ I/O= 入力 / 出力 P= 電源 — = 未使用 TTL=TTL 入力 ST= シュミットトリガ入力

† 出力は周辺機能の動作によってのみ利用できます。

NOTES:

4.0 内蔵オシレータ回路

内部のオシレータ回路は、デバイスのクロックを発生させるために使われます。4 つのデバイスのクロック周期が 1 つの内部の命令クロック (TCY) を発生させます。オシレータが発生させることができるのは 4 つのモードです。これらは、デバイスのプログラミング中に、デバイスのコンフィギュレーションビットにより選ばれます。これらのモードは下記の通りです。

- **LF 低周波数 (Fosc<=2MHz)**
- **XT 標準クリスタル / レゾネータ周波数 (2MHz<=Fosc<=33MHz)**
- **EC 外部クロック入力 (オシレータ設定をデフォルト)**
- **RC 外部抵抗 / キャパシタ (Fosc<=4MHz)**

パワーアップ時に必要な遅延時間を作るために 2 個のタイマがあります。1 つはオシレータスタートアップタイマ (OST) で、クリスタルオシレータが安定するまでチップを RESET 状態に保つように考えられています。もう 1 つはパワーアップタイマ (PWRT) で、パワーアップ時にのみ 96ms(公称) の一定遅延時間を作り、電源が安定するまでパーツを RESET 状態に保つように設計されています。チップに内蔵されたこの 2 個のタイマにより、ほとんどのアプリケーションで外部リセット回路の必要がありません。

SLEEPモードは非常に少ない消費電流のパワーダウンモードを実現できるように設計されています。外部リセットやウォッチドッグタイマのリセット、割込みにより SLEEP からウェークすることができます。

いくつかのオシレータのオプションによっても、そのパーツをアプリケーションに適応させることができます。RC オシレータのオプションによりシステムコストを節約し、LF クリスタルにより電力を節約します。コンフィギュレーションビットで、いろいろなオプションを選択できます。

4.1 オシレータの設定

4.1.1 オシレータの種類

PIC17CXXX は、4 種類の異なったオシレータのモードで動作できます。次の 4 種類のモードから 1 つを選択するために、2 個のコンフィギュレーションビット (FOSC1:FOSC0) をプログラミングできます。

- **LF ローパワークリスタル**
- **XT クリスタル / レゾネータ**
- **EC 外部クロック入力**
- **RC 抵抗 / キャパシタ**

LF と XT モードの主な違いは、オシレータ回路の内部インバータのゲインで、異なった周波数幅が可能です。

デバイスのコンフィギュレーションビットの詳細については、17.0 章を参照してください。

4.1.2 クリスタルオシレータ / セラミックレゾネータ

XT または LF モードでは、発振を起こすためにクリスタルまたはセラミックレゾネータ (共振子) が OSC1/CLKIN と OSC2/CLKOUT のピンに接続されます (図 4-2 参照)。PIC17CXXX のオシレータの設計にはパラレルカットクリスタルを使用する必要があります。シリーズカットクリスタルの使用はクリスタル製造会社の規格外の周波数を与えることがあります。

20MHz 以上の周波数に対しては、オーバートーンモードクリスタルであることが普通です。オーバートーンモードクリスタルを使用するには、基本周波数でのゲインを減らすためにタンク回路が必要です。図 4-3 に、回路の例を示します。

4.1.2.1 オシレータ / レゾネータのスタートアップ

デバイスの電圧が VSS から減少すると、オシレータは発振を始めます。オシレータが発振を始めるのに必要な時間は、下記のようないろいろな要因によって違ってきます。

- クリスタル / レゾネータ周波数
- 使用されたキャパシタの値 (C1 と C2)
- デバイスの VDD の立ち上がり時間
- システムの温度
- 使用した場合、直列抵抗値（及び種類）
- デバイスのオシレータモードの選択 (内部オシレータインバータのゲインを選択する)

図 4-1 に、標準的なオシレータ / レゾネータの スタートアップの例を示します。オシレータの波形のピーク間電圧は、波形が VDD/2 で真ん中に来る時は非常に低くすることができます (VDD デバイスの 50%以下)。(電気仕様の章のパラメータ番号 D033 と D043 を参照してください)。

図 4-1: オシレータ / レゾネータのスタートアップ特性

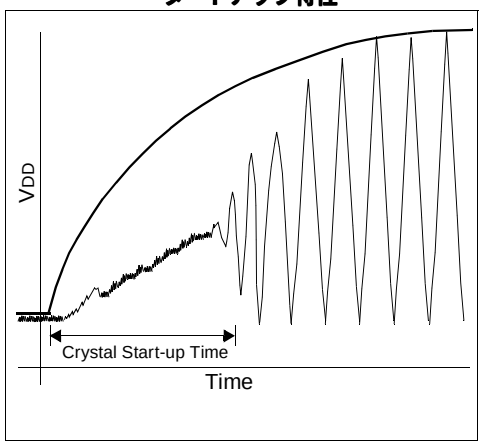
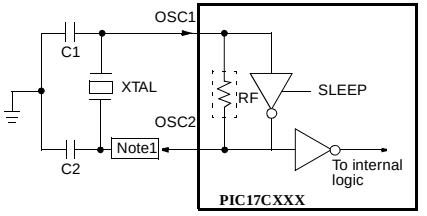


図 4-2: クリスタル / セラミックレゾネータ操作 (XT または LF の OSC 構成)



C1 と C2 の推奨値は、表 4-1 と 4-2 を参照してください。

注 1: AT ストリップカットクリスタルには、直列抵抗 (Rs) が必要な場合もあります。

表 4-1: セラミックレゾネータ用のキャパシタの選択

オシレータの種類	レゾネータ周波数	キャパシタ範囲 C1 = C2(1)
LF	455 kHz	15 - 68 pF
	2.0 MHz	10 - 33 pF
XT	4.0 MHz	22 - 68 pF
	8.0 MHz	33 - 100 pF
	16.0 MHz	33 - 100 pF

高キャパシタンスはオシレータの安定性を増しますが、スタートアップ時間も増えます。これらの値は設計の指針を示すためだけのものです。個々のレゾネータにはそれぞれの特性がありますので、外部コンポーネントの適切な値についてはレゾネータの製造元にお問い合わせください。

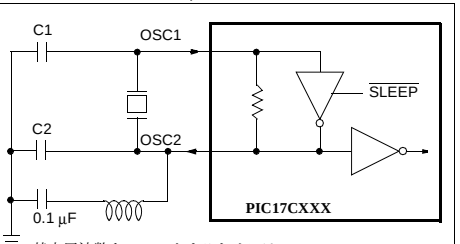
注 1: これらの値は、このピン上でのすべてのボードのキャパシタンスを含んでいます。現実のキャパシタの値はボードのキャパシタンスによります。

使用レゾネータ :

455 kHz	Panasonic EFO-A455K04B	± 0.3%
2.0 MHz	Murata Erie CSA2.00MG	± 0.5%
4.0 MHz	Murata Erie CSA4.00MG	± 0.5%
8.0 MHz	Murata Erie CSA8.00MT	± 0.5%
16.0 MHz	Murata Erie CSA16.00MX	± 0.5%

キャパシタ内蔵タイプは使用しておりません。

図 4-3: クリスタル操作、オーバートーンしたクリスタル (XT の OSC 構成)



基本周波数をフィルタするためには

$$\frac{1}{LC2} = (2\pi f)^2$$

f = タンク回路の共振周波数。これはクリスタルの基本周波数と 3 番目のオーバートーン周波数の間の中ほどにあることが必要です。

表 4-2: クリスタルオシレータ用のキャパシタの選択

Osc Type	Freq	C1 (3)	C2 (3)
LF	32 kHz(1)	100-150 pF	100-150 pF
	1 MHz	10-33 pF	10-33 pF
	2 MHz	10-33 pF	10-33 pF
XT	2 MHz	47-100 pF	47-100 pF
	4 MHz	15-68 pF	15-68 pF
	8 MHz (2)	15-47 pF	15-47 pF
	16 MHz	TBD	TBD
	25 MHz	15-47 pF	15-47 pF
	32 MHz (3)	10	10

高キャパシタンスはオシレータの安定性を増しますが、スタートアップ時間とオシレータの電流も増えます。これらの値は設計の指針を示すためだけのものです。低駆動レベルの設定でのクリスタルをオーバードライブしないために、XT モードでは Rs が必要となります。個々のクリスタルにはそれぞれの特性がありますので、外部コンポーネントの適切な値についてはクリスタルの製造元にお問い合わせください。

注 1: VDD>4.5V、C1=C2 30pF が推奨値です。
2: 15/15pF のキャパシタの組み合わせのためには、330Ω の Rs が必要です。
3: これらの値は、このピン上でのすべてのボードのキャパシタンスを含んでいます。現実のキャパシタの値はボードのキャパシタンスによります。

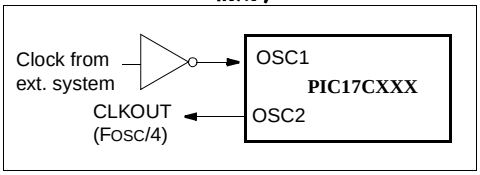
使用クリスタル :

32.768 kHz	Epson C-001R32.768K-A	± 20 PPM
1.0 MHz	ECS-10-13-1	± 50 PPM
2.0 MHz	ECS-20-20-1	± 50 PPM
4.0 MHz	ECS-40-20-1	± 50 PPM
8.0 MHz	ECS ECS-80-S-4 ECS-80-18-1	± 50 PPM
16.0 MHz	ECS-160-20-1	TBD
25 MHz	CTS CTS25M	± 50 PPM
32 MHz	CRYSTEK HF-2	± 50 PPM

4.1.3 外部クロックオシレータ

EC オシレータモードでは、CMOS ドライバにより OSC1 入力を行なうことができます。このモードでは、OSC1/CLKIN ピンは高インピーダンスで、OSC2/CLKOUT ピンは、CLKOUT 出力です (4Tosc)。

図 4-4: 外部クロック入力の操作 (EC OSC 構成)



4.1.4 外部クリスタルオシレータ回路

プレパックされたオシレータの使用や、TTL ゲートを使った簡単なオシレータ回路を作ることもできます。プレパックされたオシレータは広い動作範囲とすぐれた安定性を持っています。良く設計されたクリスタルオシレータは TTL ゲートで良い性能を発揮します。2 種類のクリスタルオシレータ回路は、直列共振または並列共振として使うことができます。

図 4-5 に並列共振オシレータ回路の構成を示します。この回路はクリスタルの基本周波数を使うために設計されています。74AS04 インバータが並列オシレータに必要な 180 度の位相シフトを行ないます。4.7kΩ の抵抗は安定用のネガティブフィードバックのためです。10kΩ のポテンショメータは 74AS04 を線形領域で使うためのバイアス用です。これは外部オシレータの設計に使えます。

図 4-5: 外部並列共振クリスタルオシレータ回路

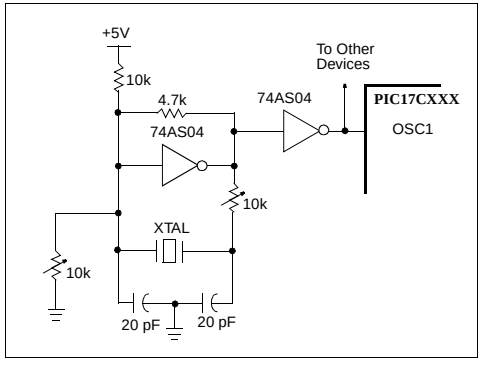
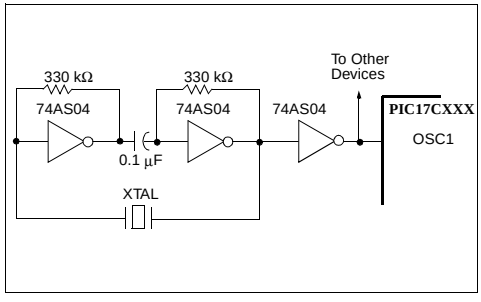


図 4-6 に直列共振オシレータ回路を示します。この回路もクリスタルの基本周波数を使うために設計されています。インバータが直列共振オシレータ回路での 180 度の位相シフトを行ないます。330kΩ の抵抗はインバータを線形領域にバイアスするためのネガティブフィードバックを構成します。

図 4-6: 外部直列共振クリスタルオシレータ回路



4.1.5 RC オシレータ

それほどタイミング精度を必要としないアプリケーションでは、RC 部品を使用すればさらにコストを節約できます。RC オシレータ周波数は供給電圧、抵抗 (Rext) とキャパシタ (Cext) の値、動作温度により変わります。これに加え、このオシレータ周波数は通常の生産パラメータのばらつきにより、同製品間でも異なります。さらに、パッケージの種類によるリードフレーム容量の差も、特に低い Cext の値で、発振周波数に影響を与えます。外部に使う R と C の部品に誤差があるため、そのばらつきを考慮する必要があります。図 4-7 に R/C の組み合わせがどのように PIC17CXXX に接続されるかを示します。2.2kΩ 以下の Rext 値では、オシレータ動作が不安定になるか、完全に停止することがあります。非常に高い Rext 値では (例えば、1MΩ)、オシレータがノイズ、湿度、漏れ電流の影響を受けやすくなります。したがって、Rext は 3kΩ から 100kΩ の間をお勧めします。

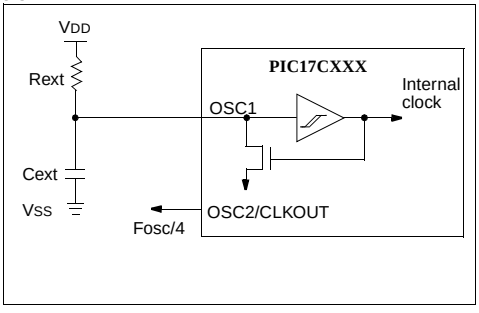
オシレータは外部キャパシタが無くても (Cext=0pF) 動作しますが、ノイズと安定のために、20pF 以上の値を使うことをお勧めします。外部容量が小さい、あるいは全くないと、PCB のトレース容量やパッケージのリードフレーム容量などの外部容量の変化で、発振周波数が非常に異なることがあります。

通常の製造工程のばらつきによる製品毎の RC 周波数の変化に関しては、21.0 章をご覧ください。その変化は、より大きな R(漏れ電流のばらつきが大きな R に対して RC 周波数に多くの影響を与えるため)、そしてより小さな C(入力容量のばらつきが RC 周波数に多くの影響を与えるため) に対して大きくなります。

与えられた R、C、VDD の値に対する動作温度による周波数変化と同様に、与えられた Rext/Cext の値に対する VDD によるオシレータ周波数変化に関しては、21.0 章をご覧ください。

4 分周されたオシレータ周波数が OSC2/CLKOUT ピンに出力されます。これはテストの目的や、他のロジックと同期するために使用できます (波形に関しては図 4-8 参照)。

図 4-7: RC オシレータモード



4.1.5.1 RC のスタートアップ

デバイスの電圧が増すと、ピンの電圧レベルが入力しきい値設定になるので、RC はただちにオシレーションを始めます (電気的仕様の章のパラメータ番号 D032、D042 を参照)。RC がオシレーションを始めるのに必要な時間は、下記のような要因によって異なります。

- **使用抵抗の値**
- **使用キャパシタの値**
- **デバイスの VDD 立ち上がり時間**
- **システムの温度**

4.2 クロック方式 / 命令サイクル

クロック入力 (OSC1 から) は、内部で 4 分周され、Q1、Q2、Q3、Q4 と呼ばれる 4 相のオーバーラップしない矩形波クロックを発生させます。内部で、プログラムカウンタ (PC) は Q1 毎にインクリメントされ、命令はプログラムメモリからフェッチされ、Q4 で命令レジスタにラッチされます。命令は次の Q1 から Q4 の間でデコードされ、実行されます。クロックと命令の実行フローを図 4-8 に示します。

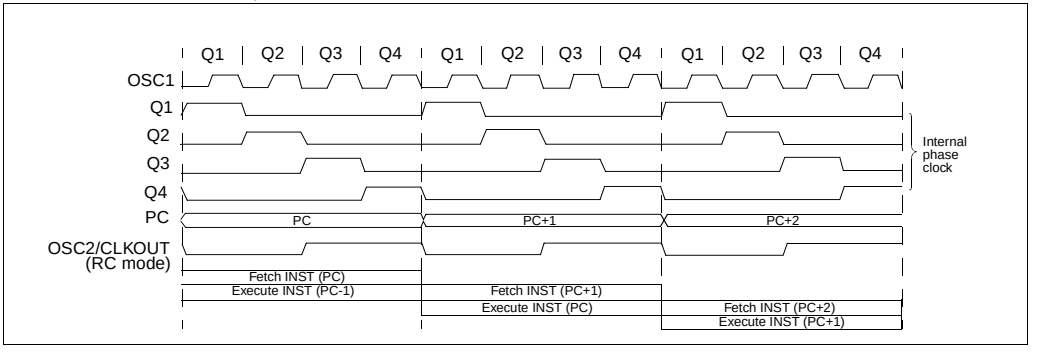
4.3 命令のフロー / パイプライン

命令サイクルは、4 つの Q サイクル (Q1、Q2、Q3、Q4) で構成されています。命令のフェッチと実行は、パイプライン方式で行なわれ、フェッチに 1 命令サイクル、デコードと実行に 1 命令サイクルがかかります。しかしパイプラインによって、各命令は効率よく 1 サイクルで実行されます。命令によってプログラムカウンタを変更する場合 (GOTO など) は、 命令を完了するために 2 サイクルが必要です (例 4-1 参照)。

フェッチサイクルはプログラムカウンタが Q1 でインクリメントすることで始まります。

実行サイクルでは、フェッチされた命令は Q1 サイクルで " 命令レジスタ (IR) " にラッチされます。この命令はそれから Q2、Q3、Q4 サイクルの間でデコードされ、実行されます。データメモリは、Q2 の間でリード (オペランドリード) され、Q4 の間でライト (デスティネーションライト) されます。

図 4-8: クロック / 命令サイクル



例 4-1: 命令パイプライン･フロー

	Tcy0	Tcy1	Tcy2	Tcy3	Tcy4	Tcy5
1. MOVLW 55h	Fetch 1	Execute 1				
2. MOVWF PORTB		Fetch 2	Execute 2			
3. CALL SUB_1			Fetch 3	Execute 3		
4. BSF PORTA, BIT3 (Forced NOP)				Fetch 4	Flush	
5. Instruction @ address SUB_1					Fetch SUB_1	Execute SUB_1

分岐命令を除くすべての命令はシングルサイクルで動作。次の命令がフェッチされ実行される間に、すでにフェッチされた命令がパイプラインから " フラッシュ " される。このフローは 2 サイクル。

NOTES:

5.0 リセット

PIC17CXXX は色々な種類のリセットを区別します。

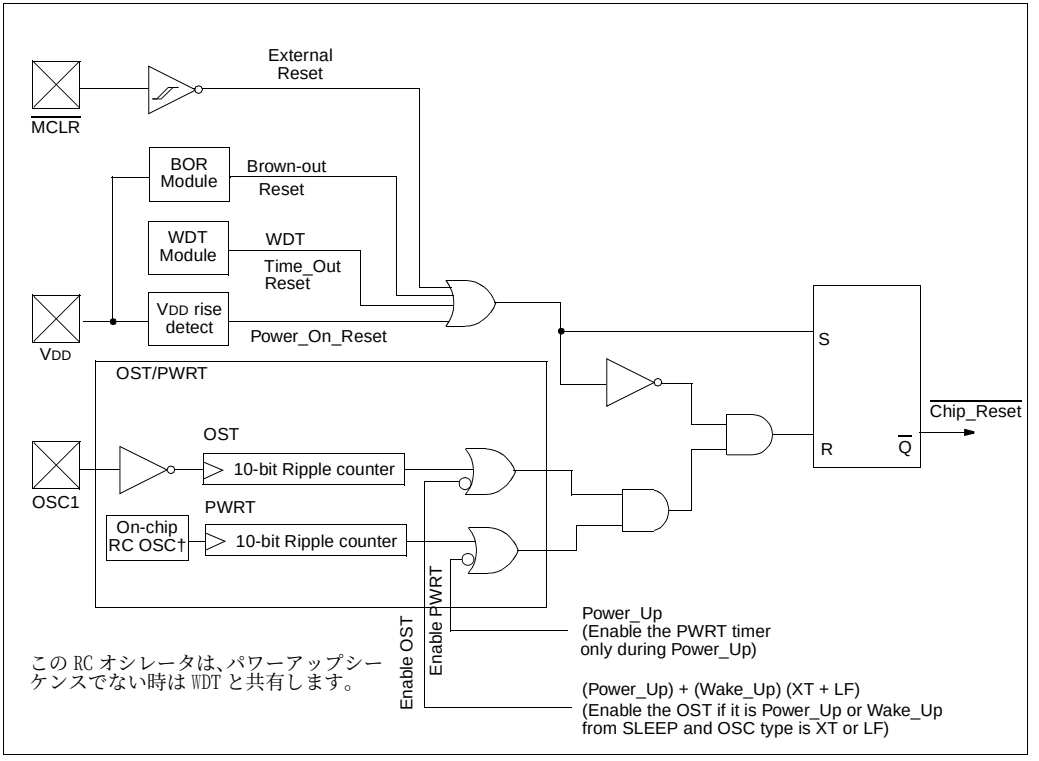
- **パワーオンリセット (POR)**
- **通常動作中の $\overline{MCLR}$ リセット**
- **ブラウンアウトリセット**
- **WDT リセット (通常動作時)**

いくつかのレジスタは、どのリセット条件からも影響を受けません。それらのステータスは POR では未知で、他のリセットでは変化しません。他のほとんどのレジスタは、パワーオンリセット (POR)、ブラウンアウトリセット (BOR)、$\overline{MCLR}$ または WDT リセット、SLEEP 中の $\overline{MCLR}$ リセットで強制的に " リセット状態 " になります。SLEEP 中の WDT リセットは、通常動作の再開として見なされます。$\overline{TO}$ と $\overline{PD}$ ビットは表 5-3 に示すように、異なったリセット状態では、それぞれ異なってセットまたはクリアされます。これらのビットはリセットの性質を判断するために、ソフトウエアで使われます。すべてのレジスタのリセット状態の詳細に関しては表 5-4 をご覧ください。

注意 : デバイスがリセット状態の間、内部位相クロックは Q1 状態にホールドされます。外部の実行が可能などんなプロセッサモードでも、強制的に RE0/ALE ピンを Low に、RE1/$\overline{OE}$ と RE2/$\overline{WR}$ ピンを High にします。

内蔵リセット回路の簡易ブロック図を図 5-1 に示します。

図 5-1: 内蔵リセット回路の簡易ブロック図



5.1 パワーオンリセット (POR)、パワーアップタイマ (PWRT)、オシレータスタートアップタイマ (OST)、ブラウンアウトリセット (BOR)

5.1.1 パワーオンリセット (POR)

VDD がトリップポイント以上 (1.4V-2.3V の範囲で) になるまで、パワーオンリセット回路はデバイスをリセット状態に保持します。デバイスは、立ち上がりと立ち下がりのVDD両方に対して内部リセットを引き起こします。POR を利用するためには、MCLR/VPP ピンを VDD に直接 (または抵抗を通して) 接続するだけです。これにより、パワーオンリセットに通常必要とされる外部の RC 部品を省略できます。VDD に対して最小立ち上がり時間が必要とされます。詳細については電気的仕様をご覧ください。

図 5-2 と 5-3 に、2 つの可能性のある POR 回路を示します。

図 5-2: 内蔵 POR の使用

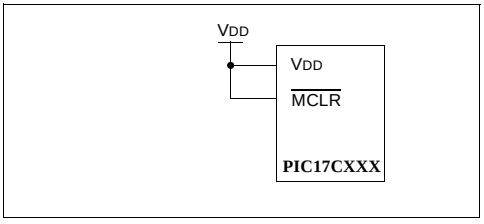
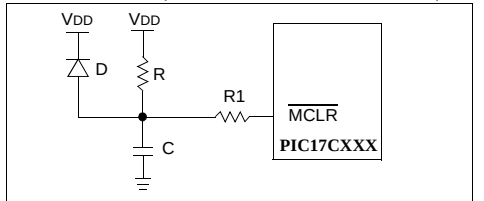


図 5-3: 外部パワーオンリセット回路 (VDD の立ち上がりが遅い時)



注 1: 外部パワーオンリセット回路は、VDD パワーアップ時間が非常に遅い場合にのみ必要となります。VDD がパワーダウンする時、ダイオード D はコンデンサが急速に放電するのを助けます。

2: R の電圧降下が 0.2V を確実に越えないようするためには、R<40kΩ をお勧めします (MCLR/Vpp ピンの最大漏れ電流の設定は 5μA)。より大きい電圧降下は MCLR/Vpp ピンの VIH レベルを下げます。

3: 静電放流 (ESD) または電気的オーバーストレス (EOS) による MCLR/Vpp ピンのブレークダウンが起きた時、R1 を 100Ω から 1kΩ にしておけば、外部キャパシタ C から MCLR への電流の流入を制限します。

5.1.2 パワーアップタイマ (PWRT)

パワーアップタイマは、パワーアップ時に 96ms のタイムアウト (公称) を生成します。これは POR シグナルの立ち上がりエッジからと、MCLR の最初の立ち上がりエッジの後に起こります (ハイを検出)。パワーアップタイマは内部 RC オシレータで動作します。PWRT が動作状態の間は、チップを RESET 状態に保持します。ほとんどの場合、PWRT の遅延時間は、VDD が規定の電圧レベルに達するのに十分です。

パワーアップ時間の遅れはチップごとにより、また VDD や温度により異なります。詳細は DC パラメータの章をご覧ください。

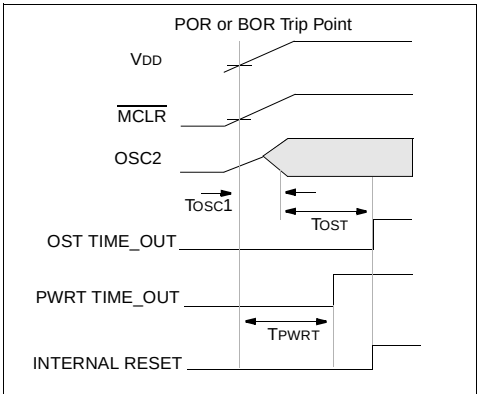
5.1.3 オシレータスタートアップタイマ (OST)

オシレータスタートアップタイマ (OST) は、MCLR がハイを検出するか、SLEEP イベントからのウェークアップが起こった後、1024 オシレータサイクル (1024Tosc) の遅延時間を作ります。OST タイムアウトは、パワーオンリセット時の XT と LF のオシレータモードか、SLEEP からのウェークアップ時にのみ起動します。

OST は、OSC1/CLKIN ピンでオシレータパルスを数えます。シグナルの振幅がオシレータ入力のしきい値に達した後、カウンタのみが増加し始めます。この遅延時間により、デバイスがリセット状態になる前に、クリスタルオシレータやレゾネータを安定させることができます。タイムアウトの長さは、クリスタル / レゾネータの周波数の関数です。

図 5-4 に、OST 回路の動作を示します。この図では、オシレータは低い周波数なので、パワーアップタイマがタイムアウトした後に、OST タイムアウトが起こります。

図 5-4: オシレータのスタートアップ時間



この図は、オシレータのスタートアップタイマを含んだタイミングの詳細を示しています。この例では、低周波数のクリスタルスタートアップ時間は、パワーアップ時間 (TPWRT) より大きいです。

Tosc1 ＝オシレータスタートアップタイマ (ost) により検出できるオシレーションレベルに反応するクリスタルオシレータ用の時間

TOST = 1024TOSC.

5.1.4 タイムアウトのシーケンス

パワーアップ時のタイムアウトシーケンスは次のようになります。POR トリップポイントに達すると、最初に内部 POR シグナルがハイになります。もし $\overline{MCLR}$ がハイなら、それから OST と PWRT 両方のタイマが始まります。一般に、低周波数のクリスタル / レゾネータを除いて、PWRT のタイムアウトの方が長くなります。全体のタイムアウトもオシレータの設定を基に変化します。表 5-1 に、オシレータ設定と関連する時間を示します。図 5-5 と 5-6 に、これらのタイムアウトシーケンスを示します。

デバイスの電圧がタイムアウトの最後で電気的仕様以内でない場合、$\overline{MCLR}$/Vpp ピンは、電圧がデバイスの仕様以内になるまで、ローを保持しなければなりません。外部 RC の遅延時間の使用は、これら多くの応用に十分です。

タイムアウトシーケンスは、$\overline{MCLR}$ の最初の立ち上がりエッジから始まります。

表 5-3 にいくつかの特殊レジスタのリセット条件を示し、表 5-4 にはすべてのレジスタの初期化条件を示します。

表 5-1: 様々な状態でのタイムアウト

Oscillator Configuration	Power-up	Wake up from SLEEP	$\overline{MCLR}$ Reset	BOR
XT, LF	Greater of: 96 ms or 1024TOSC	1024TOSC	—	—
EC, RC	Greater of: 96 ms or 1024TOSC	—	—	—

表 5-2: ステータスビットとその意味

$\overline{POR}$	$\overline{BOR}$ (1)	$\overline{TO}$	$\overline{PD}$	イベント
0	0	1	1	パワーオンリセット
1	1	1	0	SLEEP 時 $\overline{MCLR}$ リセットまたは SLEEP からの割込みウェークアップ
1	1	0	1	ノーマル動作時 WDT リセット
1	1	0	0	SLEEP 時 WDT ウェークアップ
1	1	1	1	ノーマル動作時 $\overline{MCLR}$ リセット
1	0	x	x	ブラウンアウトリセット
0	0	0	x	不定。$\overline{POR}$ 時に $\overline{TO}$ はセット
0	0	x	0	不定。$\overline{POR}$ 時に $\overline{PD}$ はセット
x	x	1	1	CLRWDT 命令実行

注 1: BOR がイネーブルの時です。その他の時は $\overline{BOR}$ ステータスビットは不定です。

表 5-3: プログラムカウンタと CPUSTA レジスタのリセット条件

イベント		PCH:PCL	CPUSTA (4)	OST アクティブ
パワーオンリセット		0000h	--11 1100	Yes
ブラウンアウトリセット		0000h	--11 1101	No
ノーマル動作時の $\overline{MCLR}$ リセット		0000h	--11 1111	No
SLEEP 時の $\overline{MCLR}$ リセット		0000h	--11 1011	Yes (2)
ノーマル動作時の WDT リセット		0000h	--11 0111	No
SLEEP 時の WDT ウェークアップ (3)		0000h	--11 0011	Yes (2)
SLEEP からのインターラプトウェークアップ	GLINTD is set	PC + 1	--11 1011	Yes (2)
	GLINTD is clear	PC + 1 (1)	--10 1011	Yes (2)

凡例: u= 不変、x= 不定、-= 未使用、‘0’ としてリード。

注 1: ウェークアップ時に、この命令が実行されます。適当な割込みベクトルでの命令がフェッチされそれから実行されます。

2: オシレータを XT または LF モードに設定した場合のみ、OST はアクティブです。

3: プログラムカウンタ =0 です。すなわちデバイスはリセットベクトルに分岐します。これはミッドレンジのデバイスとは異なります。

4: $\overline{BOR}$ がイネーブルの時です。その他の時は、$\overline{BOR}$ ステータスビットは不定です。

図 5-5、5-6、5-7 は、$T_{PWRT} > T_{OST}$ のような、高周波数クリスタルの場合です。低周波数クリスタルについては(すなわち 32kHz)、T_{OST} がより大きくなります。

図 5-5: パワーアップ時のタイムアウトシーケンス (MCLR が VDD に接続されている場合)

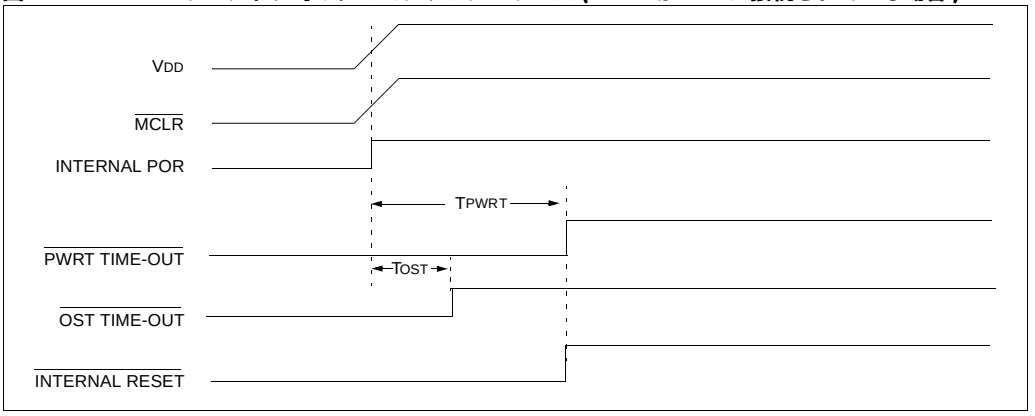


図 5-6: パワーアップ時のタイムアウトシーケンス (MCLR が VDD に接続されていない場合)

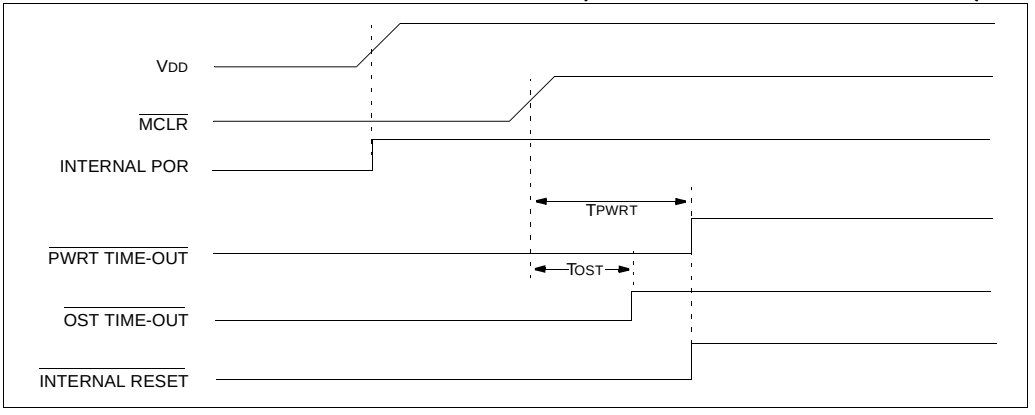


図 5-7: 立ち上がり時間が遅い時 (MCLR が VDD に接続されている場合)

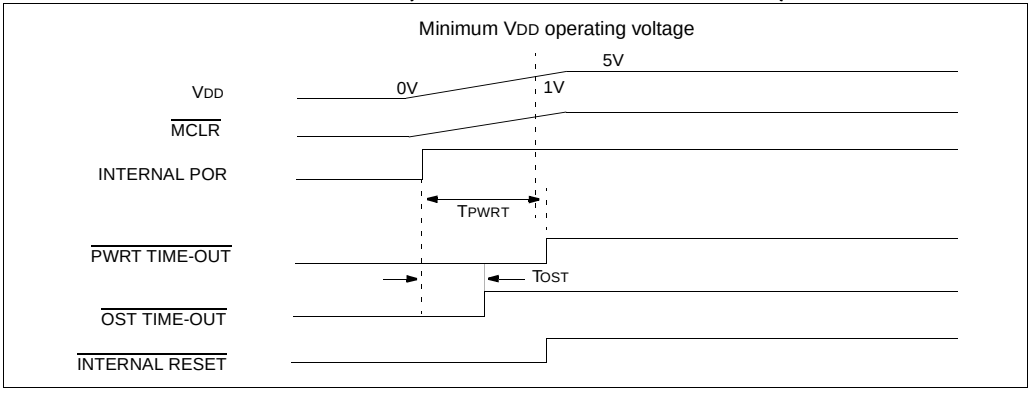


表 5-4: 特殊機能レジスタの初期化条件

レジスタ	アドレス	パワーオンリセット ブラウンアウトリセット	MCLR リセット WDT リセット	インターラプトによる SLEEP からのウェーク アップ
Unbanked				
INDF0	00h	N.A.	N.A.	N.A.
FSR0	01h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	02h	0000h	0000h	PC + 1(2)
PCLATH	03h	0000 0000	0000 0000	uuuu uuuu
ALUSTA	04h	1111 xxxx	1111 uuuu	1111 uuuu
T0STA	05h	0000 000-	0000 000-	0000 000-
CPUSTA(3)	06h	--11 1100(4)	--11 qquu(4)	--uu qquu(4)
INTSTA	07h	0000 0000	0000 0000	uuuu uuuu(1)
INDF1	08h	N.A.	N.A.	N.A.
FSR1	09h	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	0Ah	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR0L	0Bh	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR0H	0Ch	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLPTRL	0Dh	0000 0000	0000 0000	uuuu uuuu
TBLPTRH	0Eh	0000 0000	0000 0000	uuuu uuuu
BSR	0Fh	0000 0000	0000 0000	uuuu uuuu
Bank 0				
PORTA	10h	0-xx xxxx	0-uu uuuu	u-uu uuuu
DDRB	11h	1111 1111	1111 1111	uuuu uuuu
PORTB	12h	xxxx xxxx	uuuu uuuu	uuuu uuuu
RCSTA1	13h	0000 -00x	0000 -00u	uuuu -uuu
RCREG1	14h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXSTA1	15h	0000 --1x	0000 --1u	uuuu --uu
TXREG1	16h	xxxx xxxx	uuuu uuuu	uuuu uuuu
SPBRG1	17h	xxxx xxxx	uuuu uuuu	uuuu uuuu
Bank 1				
DDRC	10h	1111 1111	1111 1111	uuuu uuuu
PORTC	11h	xxxx xxxx	uuuu uuuu	uuuu uuuu
DDRD	12h	1111 1111	1111 1111	uuuu uuuu
PORTD	13h	xxxx xxxx	uuuu uuuu	uuuu uuuu
DDRE	14h	---- 1111	---- 1111	---- uuuu
PORTE	15h	---- xxxx	---- uuuu	---- uuuu
PIR1	16h	x000 0010	u000 0010	uuuu uuuu(1)
PIE1	17h	0000 0000	0000 0000	uuuu uuuu

凡例 : u= 不変、 x= 不定、-= 未使用、‘ 0 ’としてリード、q= 条件によって決まる値。

注 1: INTSTA、PIR1、PIR2 の 1 個以上のビットは影響を受けます (ウェークアップを引き起こすため)。

2: 割込みによりウェークアップし、GLINTD ビットがクリアされた場合 PC は割込みベクトルに分岐します。

3: 特定条件でのリセット値については表 5-3 を参照。

4: ブラウンアウトがイネーブルの場合です。その他の時は、$\overline{\text{BOR}}$ ビットは不定です。

表 5-4: 特殊機能レジスタの初期化条件 (Cont.1d)

レジスタ	アドレス	パワーオンリセット ブラウンアウトリセット	$\overline{\text{MCLR}}$ リセット WDT リセット	インターラプトによる SLEEP からのウェーク アップ
Bank 2				
TMR1	10h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR2	11h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	12h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3H	13h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PR1	14h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PR2	15h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PR3/CA1L	16h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PR3/CA1H	17h	xxxx xxxx	uuuu uuuu	uuuu uuuu
Bank 3				
PW1DCL	10h	xx-- ----	uu-- ----	uu-- ----
PW2DCL	11h	xx0- ----	uu0- ----	uuu- ----
PW1DCH	12h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PW2DCH	13h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CA2L	14h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CA2H	15h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TCON1	16h	0000 0000	0000 0000	uuuu uuuu
TCON2	17h	0000 0000	0000 0000	uuuu uuuu
Bank 4				
PIR2	10h	000- 0010	000- 0010	uuu- uuuu(1)
PIE2	11h	000- 0000	000- 0000	uuu- uuuu
Unimplemented	12h	---- ----	---- ----	---- ----
RCSTA2	13h	0000 -00x	0000 -00u	uuuu -uuu
RCREG2	14h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXSTA2	15h	0000 --1x	0000 --1u	uuuu --uu
TXREG2	16h	xxxx xxxx	uuuu uuuu	uuuu uuuu
SPBRG2	17h	xxxx xxxx	uuuu uuuu	uuuu uuuu
Bank 5				
DDRF	10h	1111 1111	1111 1111	uuuu uuuu
PORTF	11h	xxxx xxxx	uuuu uuuu	uuuu uuuu
DDRG	12h	1111 1111	1111 1111	uuuu uuuu
PORTG	13h	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	14h	0000 -0-0	0000 -0-0	uuuu uuuu
ADCON1	15h	000- 0000	000- 0000	uuuu uuuu
ADRESL	16h	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRESH	17h	xxxx xxxx	xxxx xxxx	uuuu uuuu

凡例 : u= 不変、 x= 不定、-= 未使用、‘0’としてリード、q= 条件によって決まる値。

注 1: INTSTA、PIR1、PIR2 の 1 個以上のビットは影響を受けます (ウェークアップを引き起こすため)。

2: 割込みによりウェークアップし、GLINTD ビットがクリアされた場合 PC は割込みベクトルに分岐します。

3: 特定条件でのリセット値については表 5-3 を参照。

4: ブラウンアウトがイネーブルの場合です。その他の時は、$\overline{\text{BOR}}$ ビットは不定です。

表 5-4: 特殊機能レジスタの初期化条件 (Cont.1d)

レジスタ	アドレス	パワーオンリセット ブラウンアウトリセット	MCLR リセット WDT リセット	インターラプトによる SLEEP からのウェーク アップ
Bank 6				
SSPADD	10h	0000 0000	0000 0000	uuuu uuuu
SSPCON1	11h	0000 0000	0000 0000	uuuu uuuu
SSPCON2	12h	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	13h	0000 0000	0000 0000	uuuu uuuu
SSPBUF	14h	xxxx xxxx	uuuu uuuu	uuuu uuuu
Unimplemented	15h	---- ----	---- ----	---- ----
Unimplemented	16h	---- ----	---- ----	---- ----
Unimplemented	17h	---- ----	---- ----	---- ----
Bank 7				
PW3DCL	10h	xxx- ----	uuu- ----	uuu- ----
PW3DCH	11h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CA3L	12h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CA3H	13h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CA4L	14h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CA4H	15h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TCON3	16h	-000 0000	-000 0000	-uuu uuuu
Unimplemented	17h	---- ----	---- ----	---- ----
Unbanked				
PRODL	18h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODH	19h	xxxx xxxx	uuuu uuuu	uuuu uuuu

凡例 : u= 不変、 x= 不定、-= 未使用、‘0’としてリード、q= 条件によって決まる値。

注 1: INTSTA、PIR1、PIR2 の 1 個以上のビットは影響を受けます (ウェークアップを引き起こすため)。

2: 割込みによりウェークアップし、GLINTD ビットがクリアされた場合 PC は割込みベクトルに分岐します。

3: 特定条件でのリセット値については表 5-3 を参照。

4: ブラウンアウトがイネーブルの場合です。その他の時は、BOR ビットは不定です。

5.1.5 ブラウンアウトリセット (BOR)

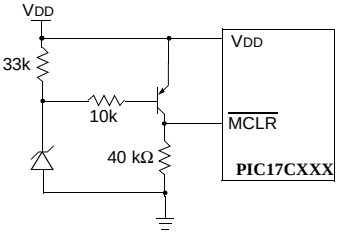
PIC17C75X デバイスには、内蔵されたブラウンアウトリセットの回路があります。デバイスの電圧がトリップポイント (BVDD) 以下に落ちる場合、この回路はデバイスをリセット状態にします。これにより確実に、デバイスが有効動作範囲外でプログラムの実行を続けないようにします。ブラウンアウトリセットは一般的に AC ラインのアプリケーションか、大きな負荷をスィッチする (車両のような)大容量の電池のアプリケーションに使われます。

注意 : 電圧の監視機能のために内蔵されたブラウンアウトを使う場合は、確実に要求を満たすために、前もって電気的仕様を再度お調べください。

コンフィギュレーションビット BODEN は、ブラウンアウトリセットの回路をディセーブルする (クリア / プログラムされた場合)、またはイネーブルする (セットされた場合) ことができます。VDD がパラメータ D035 よりも大きくなるために、BVDD(一般的に 4.0V、電気的仕様の章のパラメータ D005 を参照) 以下に落ちる場合は、ブラウンアウトの状態はチップをリセットします。VDD がパラメータ D035 よりも小さい範囲で BVDD 以下に落ちる場合は、リセットが起こることは保証されていません。VDD が BVDD 以上に上がるまで、チップはブラウンアウトリセットのままです。パワーアップタイマを起動し、チップをリセット状態に 96ms の間保ちます。パワーアップタイマが動作中に、VDD が BVDD 以下に落ちる場合は、チップはブラウンアウトリセットに戻り、パワーアップタイマは初期化されます。VDD が BVDD 以上に上がると、パワーアップタイマは 96ms の遅延を行ないます。図 5-10 に一般的なブラウンアウトの状態を示します。

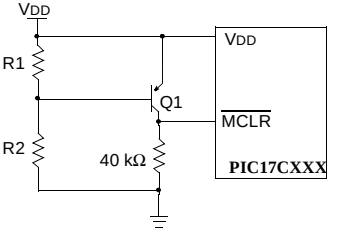
いくつかのアプリケーションでは、デバイスのブラウンアウトリセットのトリップポイントは、必要なレベルに達しないことがあります。図 5-8 と 5-9 に、インプリメントされる可能性のある外部回路の例を 2 つ示します。それらがアプリケーションの要求とあうかどうかを決定するためには、それぞれの数値を出すことが必要です。

図 5-8: 外部ブラウンアウトの保護回路 1



この回路は、VDD が (Vz+0.7V) 以下になるとリセットを作動します。ここで Vz はツェナー電圧です。

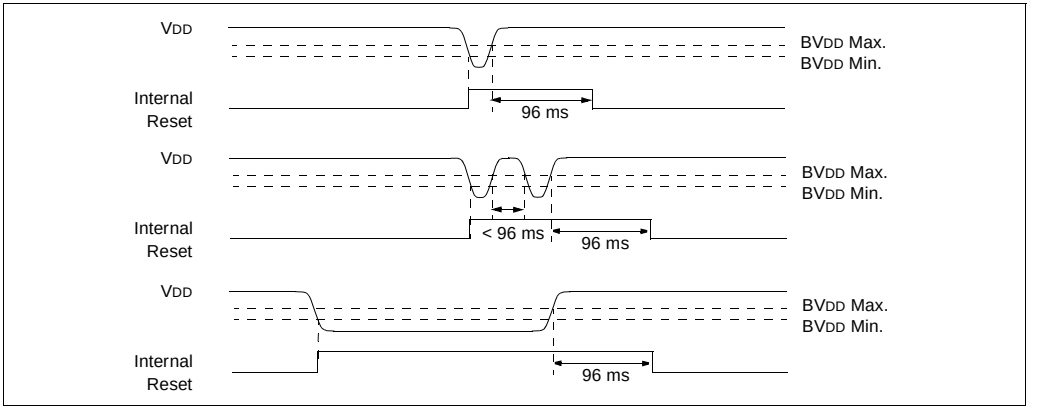
図 5-9: 外部ブラウンアウトの保護回路 2



このブラウンアウト回路は精度は落ちませんが、高価ではありません。VDD が次のようにあるレベル以下になると、トランジスタ Q1 はオフになります。

$$V_{DD} \cdot \frac{R1}{R1 + R2} = 0.7V$$

図 5-10: ブラウンアウトの状態



6.0 インターラプト

PIC17C75X のデバイスは、18 種類のインターラプトを持っています。

- **RA0/INT ピンからの外部インターラプト**
- **RB7:RB0 ピンの信号レベル変化**
- **TMR0 オーバーフロー**
- **TMR1 オーバーフロー**
- **TMR2 オーバーフロー**
- **TMR3 オーバーフロー**
- **USART1 送信バッファは空**
- **USART1 受信バッファはフル**
- **USART2 送信バッファは空**
- **USART2 受信バッファはフル**
- **SSP インターラプト**
- **SSP I^2C バス衝突インターラプト**
- **A/D 完全変換**
- **キャプチャ 1**
- **キャプチャ 2**
- **キャプチャ 3**
- **キャプチャ 4**
- **発生した T0CKI エッジ**

割込みの制御や状態に使われる下記の 6 つのレジスタがあります。

- **CPUSTA**
- **INTSTA**
- **PIE1**
- **PIR1**
- **PIE2**
- **PIR2**

CPUSTA レジスタは GLINTD ビットを含みます。これは、グローバルインターラプトディセーブルビットです。このビットがセットされると、すべての割込みがディセーブルになります。このビットは、コントローラのコア機能の一部で、メモリ構成の章で説明されています。

割込みが起きると、これ以降の割込みを禁止するために GLINTD ビットが自動的にセットされ、戻りアドレスをスタックにプッシュし、割込みベクタアドレスを PC にロードします。4 つの割込みベクタがあります。各ベクタのアドレスは、特定の割込み要因用です (すべてのベクタが同じアドレスになる周辺割込みを除く)。それらの要因は、

- **RA0/INT ピンからの外部インターラプト**
- **TMR0 オーバーフロー**
- **発生した T0CKI エッジ**
- **あらゆる周辺割込み**

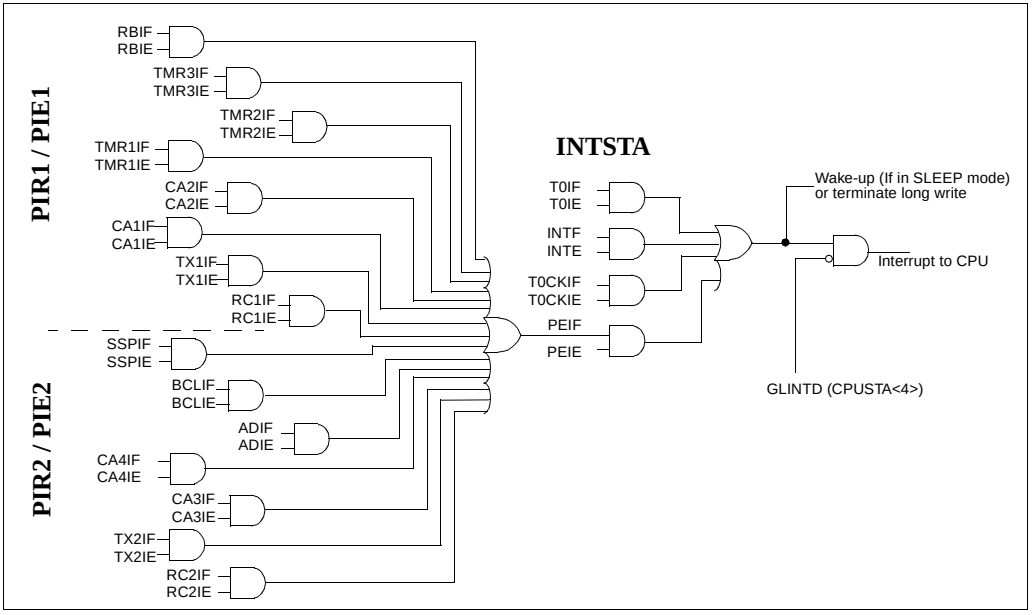
プログラムの実行がこれらの割込みベクタのアドレスの 1 つに分岐する場合 (周辺割込みを除く)、インターラプトフラグビットは自動的にクリアされます。周辺割込みベクタのアドレスに分岐しても、自動的に割込みの要因をクリアしません。周辺割込みのサービスルーチンでは、割込み要因はインターラプトフラグビットをテストすることにより決定することができます。繰り返される割込み要求を避けるためには、割込みを再イネーブルする前に、割込みフラグビットをソフトウェアでクリアしなければなりません。

割込み条件があうと、個々のインターラプトフラグビットは、それに対応するマスクビットや GLINTD ビットの状態にかかわらずセットされます。

外部割込みが起こると、割込みの遅れが出ます。その遅れは、2 サイクル命令の方が 1 サイクル命令よりも長いでしょう。

“ return from interrupt ” 命令について、RETFIE は割込みサービスルーチンの最後を指定するために使用することができます。この命令が実行される場合、スタックは “ POP ” され、GLINTD ビットはクリアされます (割込みを再イネーブルするため)。

図 6-1: 割込みロジック



6.1 インターラプトステータスレジスタ (INTSTA)

インターラプトステータス / コントロールレジスタ (INTSTA) は、フラグビットに個々の割込み要求を記録し、それぞれのインターラプトイネーブルビットを含んでいます (周辺機能用ではなく)。

PEIF ビットはリードのみで、PIR レジスタのすべての周辺フラグビットのビット論理和です (図 6-5 と 6-6 参照)。

注意 : T0IF、INTF、T0CKIF、PEIF は、対応するインターラプトイネーブルビットをクリアするか (割込みをディセーブル)、GLINTD ビットをセットしても (すべての割込みをディセーブル)、特定の条件によりセットされます。

割込みをイネーブルする場合 (GLINTD をクリア)、INTSTA レジスタイネーブルビットのどれかをクリアする時には注意が必要です。対応するインターラプトイネーブルビットをクリアするのと同じ命令サイクルで、INTSTA フラグビット (T0IF、INTF、T0CKIF、PEIF) のどれかをセットする場合、そのデバイスはリセットアドレス (0 × 00) に分岐します。INTSTA イネーブルビットのどれかをディセーブルする場合、GLINTD ビットをセットする (ディセーブルする) 必要があります。

図 6-2: INTSTA レジスタ (アドレス : 07h、バンクされていない)

R - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
- n =POR リセットでの値

bit 7: **PEIF**: 周辺割込みフラグビット
このビットは、すべての周辺割込みフラグビットの OR で、対応するイネーブルビットとともに AND されます。
1 = 周辺割込み中
0 = 周辺割込み終了

bit 6: **T0CKIF**: T0CKI ピンでの外部割込みフラグビット
割込みロジックがベクタ (18h) に分岐するために強制的にプログラムを実行する場合、このビットはハードウェアによりクリアされます。
1 = ソフトウェアで設定したエッジが RA1/T0CKI ピン上で起こった場合。
0 = ソフトウェアで設定したエッジが RA1/T0CKI ピン上で起こらなかった場合。

bit 5: **T0IF**: TMR0 オーバーフロー割込みフラグビット
割込みロジックがベクタ (10h) に分岐するために強制的にプログラムを実行する場合、このビットはハードウェアによりクリアされます。
1 = TMR0 がオーバーフローした場合。
0 = TMR0 がオーバーフローしなかった場合。

bit 4: **INTF**: INT ピンでの外部割込みフラグビット
割込みロジックがベクタ (08h) に分岐するために強制的にプログラムを実行する場合、このビットはハードウェアによりクリアされます。
1 = ソフトウェアで設定したエッジが RA0/INT ピン上で起こった場合。
0 = ソフトウェアで設定したエッジが RA0/INT ピン上で起こらなかった場合。

bit 3: **PEIE**: 周辺割込みイネーブルビット
このビットは、対応するイネーブルビットをセットするすべての周辺割込みをイネーブルにします。
1 = 周辺割込みをイネーブルにします。
0 = 周辺割込みをディセーブルにします。

bit 2: **T0CKIE**: T0CKI ピンでの外部割込みイネーブルビット
1 = RA1/T0CKI ピンでのソフトウェアで設定したエッジ割込みをイネーブルにします。
0 = RA1/T0CKI ピンでの割込みをディセーブルにします。

bit 1: **T0IE**: TMR0 オーバーフロー割込みイネーブルビット
1 = TMR0 オーバーフロー割込みをイネーブルにします。
0 = TMR0 オーバーフロー割込みをディセーブルにします。

bit 0: **INTE**: RA0/INT ピンでの外部割込みイネーブルビット
1 = RA0/INT ピンでのソフトウェアで設定したエッジ割込みをイネーブルにします。
0 = RA0/INT ピンでのソフトウェアで設定したエッジ割込みをディセーブルにします。

6.2 ペリフェラルインターラプトイネーブルレジスタ 1（PIE1）とレジスタ 2（PIE2）

これらのレジスタには、周辺割込みに対応した個々のイネーブルビットが含まれています。

図 6-3: PIE1 レジスタ (アドレス：17h、バンク 1)

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
-n = POR リセットでの値

bit 7: **RBIE**: チェンジイネーブルビットでの PORTB 割込み
1 = チェンジで PORTB 割込みをイネーブルにします。
0 = チェンジで PORTB 割込みをディセーブルにします。

bit 6: **TMR3IE**: TMR3 割込みイネーブルビット
1 = TMR3 割込みをイネーブルにします。
0 = TMR3 割込みをディセーブルにします。

bit 5: **TMR2IE**: TMR2 割込みイネーブルビット
1 = TMR2 割込みをイネーブルにします。
0 = TMR2 割込みをディセーブルにします。

bit 4: **TMR1IE**: TMR1 割込みイネーブルビット
1 = TMR1 割込みをイネーブルにします。
0 = TMR1 割込みをディセーブルにします。

bit 3: **CA2IE**: キャプチャ 2 割込みイネーブルビット
1 = キャプチャ 2 割込みをイネーブルにします。
0 = キャプチャ 2 割込みをディセーブルにします。

bit 2: **CA1IE**: キャプチャ 1 割込みイネーブルビット
1 = キャプチャ 1 割込みをイネーブルにします。
0 = キャプチャ 1 割込みをディセーブルにします。

bit 1: **TX1IE**: USART1 送信割込みイネーブルビット
1 = USART1 送信バッファ空割込みをイネーブルにします。
0 = USART1 送信バッファ空割込みをディセーブルにします。

bit 0: **RC1IE**: USART1 受信割込みイネーブルビット
1 = USART1 受信バッファフル割込みをイネーブルにします。
0 = USART1 受信バッファフル割込みをディセーブルにします。

図 6-4: PIE2 レジスタ (アドレス：11h、バンク 4)

R/W - 0	R/W - 0	R/W - 0	U - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
SSPIE	BCLIE	ADIE	–	CA4IE	CA3IE	TX2IE	RC2IE
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
-n = POR リセットでの値

bit 7: **SSPIE**: 同期シリアルポート割込みイネーブル
1 = SSP 割込みをイネーブルにします。
0 = SSP 割込みをディセーブルにします。

bit 6: **BCLIE**: バス衝突割込みイネーブル
1 = バス衝突割込みをイネーブルにします。
0 = バス衝突割込みをディセーブルにします。

bit 5: **ADIE**: A/D モジュール割込みイネーブル
1 = A/D モジュール割込みをイネーブルにします。
0 = A/D モジュール割込みをディセーブルにします。

bit 4: **Unimplemented:** 未使用：‘ 0 ’としてリード

bit 3: **CA4IE**: キャプチャ 4 割込みイネーブル
1 = キャプチャ 4 割込みをイネーブルにします。
0 = キャプチャ 4 割込みをディセーブルにします。

bit 2: **CA3IE**: キャプチャ 3 割込みイネーブル
1 = キャプチャ 3 割込みをイネーブルにします。
0 = キャプチャ 3 割込みをディセーブルにします。

bit 1: **TX2IE**: USART2 送信割込みイネーブル
1 = USART2 送信割込みをイネーブルにします。
0 = USART2 送信割込みをディセーブルにします。

bit 0: **RC2IE**: USART2 受信割込みイネーブル
1 = USART2 受信割込みをイネーブルにします。
0 = USART2 受信割込みをディセーブルにします。

6.3 ペリフェラルインターラプトリクエストレジスタ 1(PIR1) とレジスタ 2(PIR2)

これらのレジスタには、周辺割込みに対応する個々のフラグビットが含まれます。

注意： これらのビットは、対応する割込みイネーブルビットをクリア (割込みをディセーブル) するか、GLINTD ビットをセット (すべての割込みをディセーブル) しても、特定の条件によりセットされます。割込みをイネーブルする前に、プログラムがすぐに周辺割込みサービスルーチンに分岐しないよう確実にするために、割込みフラグをクリアすることもできます。

図 6-5: PIR1 レジスタ (アドレス：16h、バンク 1)

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R - 1	R - 0
RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
-n = POR リセットでの値

bit 7: **RBIF**: チェンジフラグビットでの PORTB 割込み
1 = PORTB 入力の 1 つが変化 (ソフトウェアでミスマッチ状態を終了することが必要です)。
0 = PORTB 入力の変化なし。

bit 6: **TMR3IF**: TMR3 割込みフラグビット
キャプチャ 1 がイネーブルならば (CA1/$\overline{PR3}$ = 1)
1 = TMR3 がオーバーフローした場合
0 = TMR3 がオーバーフローしなかった場合

キャプチャ 1 がディセーブルならば (CA1/$\overline{PR3}$ = 0)
1 = TMR3 の値がピリオドレジスタ (PR3H:PR3L) と等しい値から 0000h までロールオーバーした場合。
0 = TMR3 の値がピリオドレジスタ (PR3H:PR3L) と等しい値から 0000h までロールオーバーしなかった場合。

bit 5: **TMR2IF**: TMR2 割込みフラグビット
1 = TMR2 の値がピリオドレジスタ (PR2) と等しい値から 0000h までロールオーバーした場合。
0 = TMR2 の値がピリオドレジスタ (PR2) と等しい値から 0000h までロールオーバーしなかった場合。

bit 4: **TMR1IF**: TMR1 割込みフラグビット
TMR1 が 8 ビットモード (T16=0) ならば
1 = TMR1 の値がピリオドレジスタ (PR1) と等しい値から 0000h までロールオーバーした場合。
0 = TMR1 の値がピリオドレジスタ (PR1) と等しい値から 0000h までロールオーバーしなかった場合。

TMR1 が 16 ビットモード (T16=1) ならば
1 = TMR2:TMR1 の値がピリオドレジスタ (PR2:PR1) と等しい値から 0000h までロールオーバーした場合。
0 = TMR2:TMR1 の値がピリオドレジスタ (PR2:PR1) と等しい値から 0000h までロールオーバーしなかった場合。

bit 3: **CA2IF**: キャプチャ 2 割込みフラグビット
1 = キャプチャイベントが RB1/CAP2 ピン上で起こった場合。
0 = キャプチャイベントが RB1/CAP2 ピン上で起こらなかった場合。

bit 2: **CA1IF**: キャプチャ 1 割込みフラグビット
1 = キャプチャイベントが RB0/CAP1 ピン上で起こった場合。
0 = キャプチャイベントが RB0/CAP1 ピン上で起こらなかった場合。

bit 1: **TX1IF**: USART1 送信割込みフラグビット (ハードウェアで制御された状態)
1 = USART1 送信バッファが空。
0 = USART1 転送バッファがフル。

bit 0: **RC1IF**: USART1 受信割込みフラグビット (ハードウェアで制御された状態)
1 = USART1 受信バッファがフル。
0 = USART1 受信バッファが空。

図 6-6: PIR2 レジスタ (アドレス：10h、バンク 4)

R/W - 0	R/W - 0	R/W - 0	U - 0	R/W - 0	R/W - 0	R - 1	R - 0
SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
-n = POR リセットでの値

bit 7: **SSPIF**: 同期シリアルポート (SSP) 割込みフラグ
1 =SSP 割込み状態が起こり、割込みサービスルーチンからリターンする前にソフトウェアでクリアが必要な場合。このビットをセットする状態は次の通りです。:
SPI
送信 / 受信が起こった場合。
I^2C Slave / Master
送信 / 受信が起こった場合。
I^2C Master
最初のスタート状態が SSP モジュールにより終了した場合。
最初のストップ状態が SSP モジュールにより終了した場合。
最初の再スタート状態が SSP モジュールにより終了した場合。
最初の応答状態が SSP モジュールにより終了した場合。
SSP モジュールがアイドルの間にスタート状態が起こった場合 (マルチマスタシステム)。
SSP モジュールがアイドルの間にストップ状態が起こった場合 (マルチマスタシステム)。
0 = SSP 割込み状態が起こらなかった場合。

bit 6: **BCLIF**: バス衝突割込みフラグ
1 = I^2C マスタモードに設定された時、SSP 内でバス衝突が起こった場合。
0 = バス衝突が起こらなかった場合。

bit 5: **ADIF**: A/D モジュール割込みフラグ
1 = A/D 変換が完了。
0 = A/D 変換が未完了。

bit 4: **Unimplemented**: 未使用：‘ 0 ’としてリード

bit 3: **CA4IF**: キャプチャ 4 割込みフラグ
1 = キャプチャイベントが RE3/CAP4 ピン上で起こった場合。
0 = キャプチャイベントが RE3/CAP4 ピン上で起こらなかった場合。

bit 2: **CA3IF**: キャプチャ 3 割込みフラグ
1 = キャプチャイベントが RG4/CAP3 ピン上で起こった場合。
0 = キャプチャイベントが RG4/CAP3 ピン上で起こらなかった場合。

bit 1: **TX2IF**:USART2 送信割込みフラグ (ハードウェアで制御された状態)
1 = USART2 送信バッファが空。
0 = USART2 送信バッファがフル。

bit 0: **RC2IF**: USART2 受信割込みフラグ (ハードウェアで制御された状態)
1 = USART2 受信バッファがフル。
0 = USART2 受信バッファが空。

6.4 インターラプト動作

グローバルインターラプトディセーブルビット、つまり GLINTD(CPUSTA<4>) は、マスクされていない割込みすべてをイネーブルに (クリア時)、または全割込みをディセーブルに (セット時) 設定します。個別割込みは INTSTA レジスタに対応するイネーブルビットを操作することによりディセーブルできます。周辺割込みは、グローバルペリフェラルイネーブル PEIE ビットをディセーブルするか、特定のペリフェラルイネーブルビットをディセーブルすることが必要です。グローバルペリフェラルイネーブルビットでペリフェラルをディセーブルすることによりすべての周辺割込みをディセーブルにします。GLINTD はリセットでセットされます（割込みをディセーブル）。

RETFIE 命令は、割込みからのリターンと割込みの再イネーブルを同時に行なうことを可能にします。

割込みが起きると、これ以降の割込みを禁止するために、GLINTD ビットが自動的にセットされ、リターンアドレスをスタックにプッシュし、PC を割込みベクトルにロードします。割込みの遅延時間を減らすために 4 つの割込みベクトルがあります。

周辺割込みベクタは複数の割込み要因を持っています。周辺割込みサービスルーチン実行中に発生した別の割込みは、割込みフラグビットをポーリングすることにより処理を行ないます。繰り返しの割込みを避けるためには、割込みを再びイネーブルする前に、この周辺割込みフラグビットをソフトウエアでクリアする必要があります。

PIC17C75X のデバイスには、4 つの割込みベクタがあります。これらのベクタとそのハードウェアの優先順位を、表 6-1 に示します。もし 2 つのイネーブルされた割込みが " 同時に " 起こった場合、高い優先順位を持つものを最初に行ないます。これは、その割込みのベクトルアドレスがプログラムカウンタ(PC)にロードされるという意味です。

表 6-1: 割込みベクトルの優先順位

Address	Vector	Priority
0008h	External Interrupt on RA0/INT pin (INTF)	1 (Highest)
0010h	TMR0 overflow interrupt (T0IF)	2
0018h	External Interrupt on T0CKI (T0CKIF)	3
0020h	Peripherals (PEIF)	4 (Lowest)

注意 1: 個別割込みフラグビットは、それに対応するマスクビットやGLINTDビットの状況とは無関係にセットされます。

注意 2: INTSTA イネーブルビットのどれかをディセーブルする前に、GLINTD ビットをセットする (ディセーブルする) 必要があります。

6.5 RA0/INT 割込み

RA0/INT ピンの外部割込みはエッジトリガで、INTEDG ビット (T0STA<7>) をセットする場合は立ち上がりエッジ、INTEDG ビットをクリアする時は立ち下がりエッジです。 RA0/INT ピンに有効なエッジが与えられた時、INTF ビット (INTSTA<4>) がセットされます。この割込みは、INTE 制御ビット (INTSTA<0>) をクリアすることによりディセーブルできます。INT 割込みは、SLEEP からプロセッサをウェークアップすることができます。SLEEP 動作の詳細については、17.4 章をご覧ください。

6.6 T0CKI 割込み

RA1/T0CKI ピンの外部割込みはエッジトリガで、T0SE ビット (T0STA<6>) をセットする場合は立ち上がりエッジ、T0SE ビットをクリアする時は立ち下がりエッジです。 RA1/T0CKI ピンに有効なエッジが与えられた時、T0CKIF ビット (INTSTA<6>) がセットされます。この割込みは、T0CKI 制御ビット (INTSTA<2>) をクリアすることによりディセーブルできます。T0CKI 割込みは、SLEEP からプロセッサをウェークアップすることができます。SLEEP 動作の詳細については、17.4 章をご覧ください。

6.7 周辺割込み

周辺割込みフラグは、少なくとも周辺割込み (PEIF をセット) の起こったことを示します。PEIF ビットはリードオンリービットで、PIR レジスタのすべてのフラグビットの論理和で、PIE レジスタの対応するイネーブルビットと論理積されます。周辺割込みのいくつかは、SLEEP からプロセッサをウェークアップすることができます。SLEEP 動作の詳細については、17.4 章をご覧ください。

6.8 インターラプト期間中のコンテキストセーブ

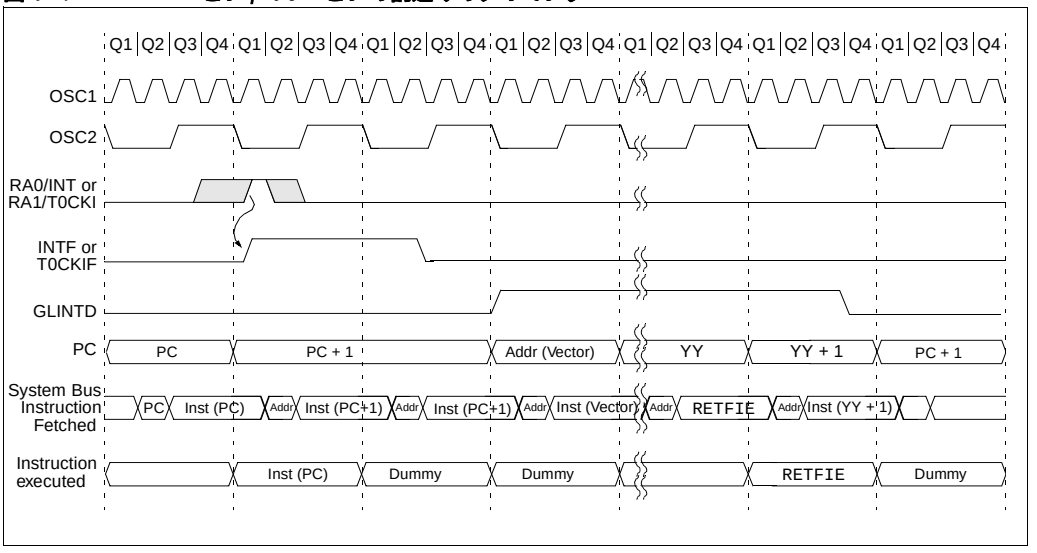
割込み中、リターンされた PC 値のみスタックにセーブされます。一般的に、ユーザーは割込み中に、重要なレジスタ、例えば WREG、ALUSTA、BSR レジスタなどをセーブしておきたいと思います。このためにはソフトウェアでの実行が必要です。

例 6-1 に、割込みサービスルーチンのセービングと復帰の情報を示します。これは簡単な割込みのスキームで、1 つの割込みだけが一度に起こります (割込みのネスティングはなし)。SFR はバンクされない GPR エリアで保持されます。

例 6-2 に、もっと複雑な割込みサービスルーチンのセービングと復帰の情報を示します。これは割込みのネスティングが必要とされる場合に有益です。この例では最大 6 レベルまで実行することができます。BSR はバンクされない GPR エリアで保持され、他のレジスタは特定のバンクで保持されます。それでこのルーチンでは 6 セーブをすることができます (6 つのバンクされない GPR レジスタがあるため)。これらのルーチンは専用の間接アドレスレジスタを必要とし、このために FSR0 を選択します。

PUSH と POP コードセグメントは、各割込みサービスルーチン内にあるか、コールされたサブルーチンのどちらかです。アプリケーションによっては、他のレジスタもセーブする必要があります。

図 6-7: INT ピン /T0CKI ピンの割込みのタイミング



例 6-1: セービングの状態と RAM での WREG(簡単なもの)

```
; The addresses that are used to store the CPUSTA and WREG values must be in the data memory
; address range of 1Ah - 1Fh. Up to 6 locations can be saved and restored using the MOVFP
; instruction. This instruction neither affects the status bits, nor corrupts the WREG register.
;
UNBANK1         EQU    0x01A       ; Address for 1st location to save
UNBANK2         EQU    0x01B       ; Address for 2nd location to save
UNBANK3         EQU    0x01C       ; Address for 3rd location to save
UNBANK4         EQU    0x01D       ; Address for 4th location to save
UNBANK5         EQU    0x01E       ; Address for 5th location to save
                                   ;    (Label Not used in program)
UNBANK6         EQU    0x01F       ; Address for 6th location to save
                                   ;    (Label Not used in program)
;
        :                          ; At Interrupt Vector Address
PUSH    MOVFP   ALUSTA, UNBANK1    ; Push ALUSTA value
        MOVFP   BSR, UNBANK2       ; Push BSR value
        MOVFP   WREG, UNBANK3      ; Push WREG value
        MOVFP   PCLATH, UNBANK4    ; Push PCLATH value
        ;
        :                          ; Interrupt Service Routine (ISR) code
        ;
POP     MOVFP   UNBANK4, PCLATH    ; Restore PCLATH value
        MOVFP   UNBANK3, WREG      ; Restore WREG value
        MOVFP   UNBANK2, BSR       ; Restore BSR value
        MOVFP   UNBANK1, ALUSTA    ; Restore ALUSTA value
;
        RETFIE                     ; Return from interrupt (enable interrupts)
```

例 6-2: セービングの状態と RAM での WREG(ネスト)

```
; The addresses that are used to store the CPUSTA and WREG values must be in the data memory
; address range of 1Ah - 1Fh. Up to 6 locations can be saved and restored using the MOVFP
; instruction. This instruction neither affects the status bits, nor corrupts the WREG register.
; This routine uses the FRS0, so it controls the FS1 and FS0 bits in the ALUSTA register.
;
Nobank_FSR     EQU    0x40
Bank_FSR       EQU    0x41
ALU_Temp       EQU    0x42
WREG_TEMP      EQU    0x43
BSR_S1         EQU    0x01A        ; 1st location to save BSR
BSR_S2         EQU    0x01B        ; 2nd location to save BSR (Label Not used in program)
BSR_S3         EQU    0x01C        ; 3rd location to save BSR (Label Not used in program)
BSR_S4         EQU    0x01D        ; 4th location to save BSR (Label Not used in program)
BSR_S5         EQU    0x01E        ; 5th location to save BSR (Label Not used in program)
BSR_S6         EQU    0x01F        ; 6th location to save BSR (Label Not used in program)
;
INITIALIZATION                     ;
        CALL    CLEAR_RAM          ; Must Clear all Data RAM
;
INIT_POINTERS                      ; Must Initialize the pointers for POP and PUSH
        CLRF    BSR, F             ; Set All banks to 0
        CLRF    ALUSTA, F          ; FSR0 post increment
        BSF     ALUSTA, FS1
        CLRF    WREG, F            ; Clear WREG
        MOVLW   BSR_S1             ; Load FSR0 with 1st address to save BSR
        MOVWF   FSR0
        MOVWF   Nobank_FSR
        MOVLW   0x20
        MOVWF   Bank_FSR
        :
        :                          ; Your code
        :
        :                          ; At Interrupt Vector Address
PUSH    BSF     ALUSTA, FS0        ; FSR0 has auto-increment, does not affect status bits
        BCF     ALUSTA, FS1        ; does not affect status bits
        MOVFP   BSR, INDF0         ; No Status bits are affected
        CLRF    BSR, F             ; Periperal and Data RAM Bank 0 No Status bits are affected
        MOVPF   ALUSTA, ALU_Temp   ;
        MOVPF   FSR0, Nobank_FSR   ; Save the FSR for BSR values
        MOVPF   WREG, WREG_TEMP    ;
        MOVFP   Bank_FSR, FSR0     ; Restore FSR value for other values
        MOVFP   ALU_Temp, INDF0    ; Push ALUSTA value
        MOVFP   WREG_TEMP, INDF0   ; Push WREG value
        MOVFP   PCLATH, INDF0      ; Push PCLATH value
        MOVPF   FSR0, Bank_FSR     ; Restore FSR value for other values
        MOVFP   Nobank_FSR, FSR0   ;
        ;
        :                          ; Interrupt Service Routine (ISR) code
        ;
POP     CLRF    ALUSTA, F          ; FSR0 has auto-decrement, does not affect status bits
        MOVFP   Bank_FSR, FSR0     ; Restore FSR value for other values
        DECF    FSR0, F            ;
        MOVFP   INDF0, PCLATH      ; Pop PCLATH value
        MOVFP   INDF0, WREG        ; Pop WREG value
        BSF     ALUSTA, FS1        ; FSR0 does not change
        MOVPF   INDF0, ALU_Temp    ; Pop ALUSTA value
        MOVPF   FSR0, Bank_FSR     ; Restore FSR value for other values
        DECF    Nobank_FSR, F      ;
        MOVFP   Nobank_FSR, FSR0   ; Save the FSR for BSR values
        MOVFP   ALU_Temp, ALUSTA   ;
        MOVFP   INDF0, BSR         ; No Status bits are affected
;
        RETFIE                     ; Return from interrupt (enable interrupts)
```

7.0 メモリ構成

PIC17C75X には、プログラムメモリとデータメモリの 2 つのメモリブロックがあります。各ブロックは専用のバスを持っているので、同じオシレータサイクルの間で各ブロックへのアクセスができます。

データメモリはさらに、汎用 RAM と特殊機能レジスタ (SFR) に分けることができます。" コア "を制御する SFR の動作をこの章で説明します。周辺モジュールを制御するのに使われるSFRは、それぞれの周辺モジュールについて述べた章で説明します。

7.1 プログラムメモリ構成

PIC17C75X デバイスには、64K × 16 のプログラムメモリ空間をアドレス指定できる16ビットのプログラムカウンタがあります。リセットベクタは 0000h に、割込みベクタは 0008h、0010h、0018h、0020h にあります (図 7-1)。

7.1.1 プログラムメモリ動作

PIC17C75X は、4 つのプログラムメモリ設定の 1 つで動作することができます。コンフィギュレーションビットで設定を選びます。可能なモードは下記の通りです。

- **マイクロプロセッサ**
- **マイクロコントローラ**
- **拡張マイクロコントローラ**
- **保護マイクロコントローラ**

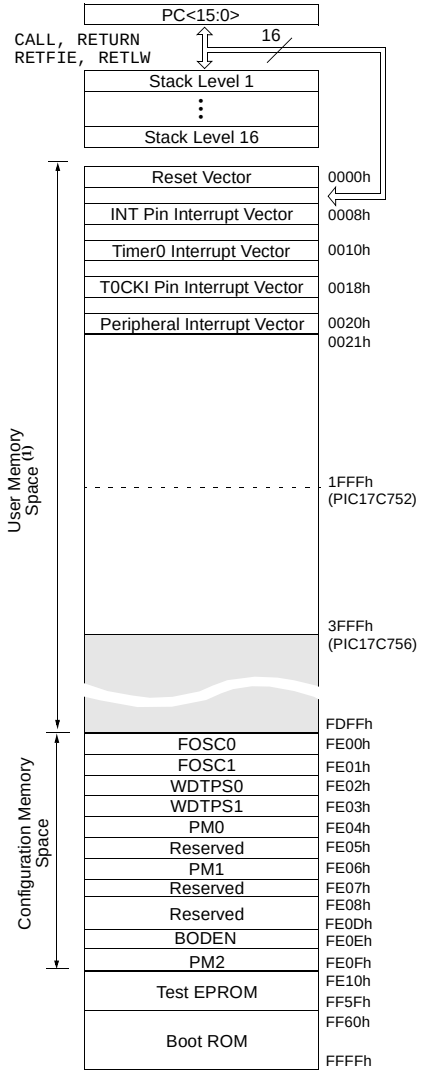
マイクロコントローラと保護マイクロコントローラモードは、内部の実行が可能なだけです。プログラムメモリを越えるアクセスはどれも不定のデータを読み込みます。保護マイクロコントローラモードも、コード保護の機能をイネーブルします。

拡張マイクロコントローラモードは、外部と内部のプログラムメモリ両方をアクセスします。内部と外部メモリの間で、自動的に実行が切り替わります。16 ビットのアドレスにより、64k ワードのプログラムメモリ幅を可能にします。

マイクロプロセッサモードは外部のプログラムメモリにのみアクセスします。内蔵されたプログラムメモリは無視されます。16 ビットのアドレスにより、64k ワードのプログラムメモリ幅を可能にします。マイクロプロセッサモードはプログラムされていないデバイスのデフォルトのモードです。

異なったモードにより、コンフィギュレーションビット、テストメモリ、ブート ROM と異なったアクセスが可能です。表 7-1 は、メモリのどのエリアにどのモードがアクセスできるかを示したものです。テストメモリとブートメモリは、デバイスの通常動作では必要ありません。これらのエリアに意図しない分岐が起きないように気をつけなければなりません。

図 7-1: プログラムメモリマップとスタック



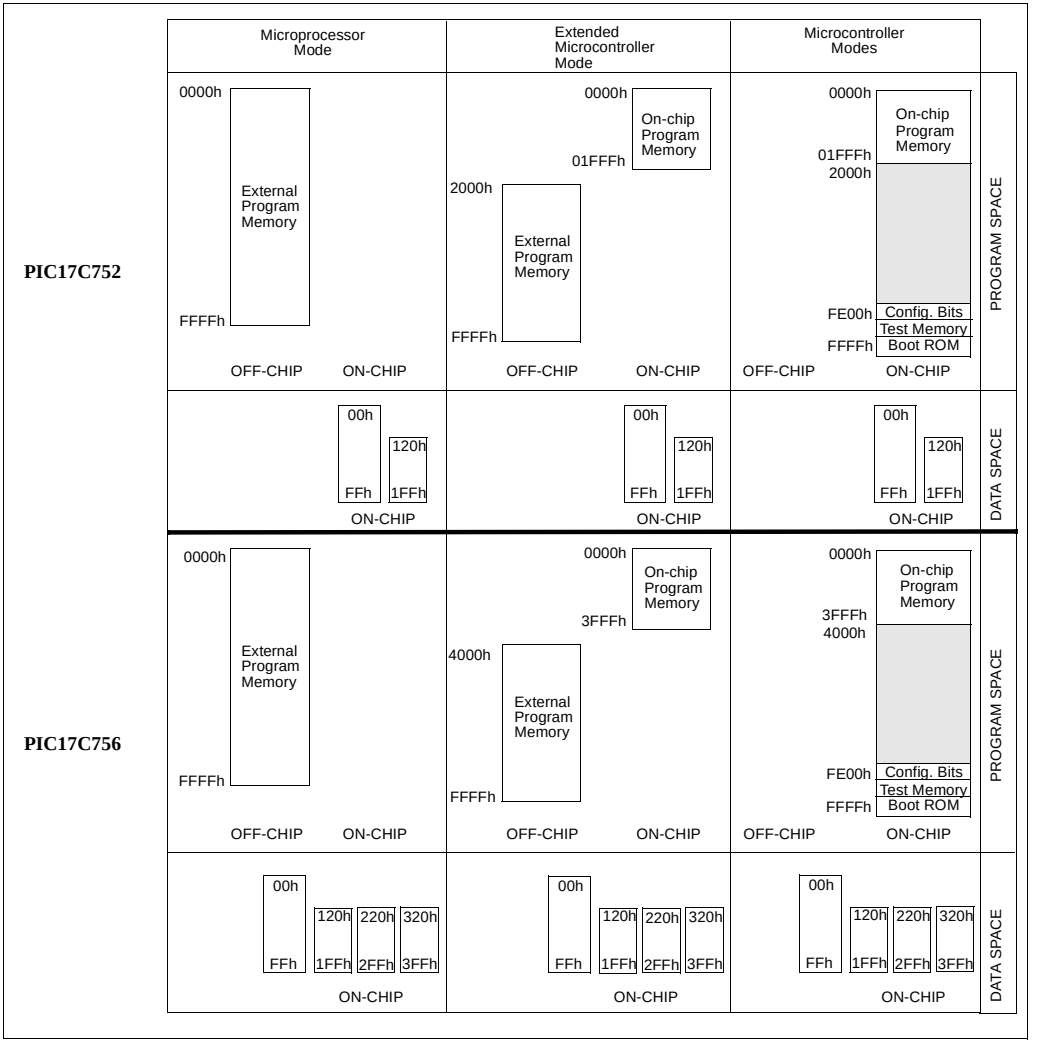
注 1: ユーザのメモリ空間は、内部、外部、両方のいずれかです。メモリ設定はプロセッサモードによります。

表 7-1: モードメモリのアクセス

動作モード	内部プログラムメモリ	コンフィギュレーションビット、テストメモリ、ブート ROM
マイクロプロセッサ	アクセス不可	アクセス不可
マイクロコントローラ	アクセス可	アクセス可
拡張マイクロコントローラ	アクセス可	アクセス不可
保護マイクロコントローラ	アクセス可	アクセス可

PiC17C75X はプログラムメモリが内蔵されていないモード、つまりマイクロプロセッサと拡張マイクロコントローラモードでも動きます。マイクロプロセッサモードはプログラムされていないデバイスにはデフォルトです。プロセッサモードにかかわらず、データメモリは常に内蔵されています。

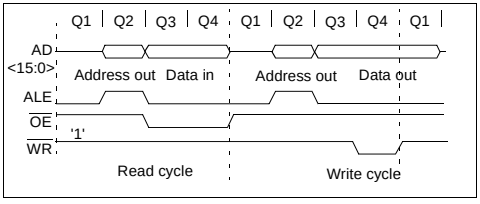
図 7-2: 異なったモードでのメモリマップ



7.1.2 外部メモリインターフェイス

マイクロプロセッサか拡張マイクロコントローラのどちらかのモードを選ぶと、PORTC、PORTD、PORTE がシステムバスとして設定されます。PORTC と PORTD は、マルチプレクスされたアドレス / データバスで、PORTE<2:0> は制御シグナル用です。外部の部品はアドレスとデータをデマルチプレクスする必要があります。これは、図 7-4 に示すように行なうことができます。アドレスとデータの波形は図 7-3 に示す通りです。タイミングを完全にするために、電気的仕様の章を参照してください。

図 7-3: 外部プログラムメモリアクセスの波形



システムバスは、バス衝突のないこと (最小の漏れ) が必要とされます。それで出力の値 (アドレス) は、最小限必要な値で保持されます。

プロセッサのスピードが増すにつれ、アクセスタイムがより速い外部 EPROM メモリを使用する必要があります。表 7-2 に、PIC17C75X デバイスの周波数に対して必要な外部メモリのスピードを示します。

拡張マイクロコントローラモードでは、デバイスが内部メモリの外で実行中の場合、制御シグナルが動作し続けます。すなわち、それらは内部メモリ内で起こっている動作を示します。外部メモリのアクセスは無視されます。

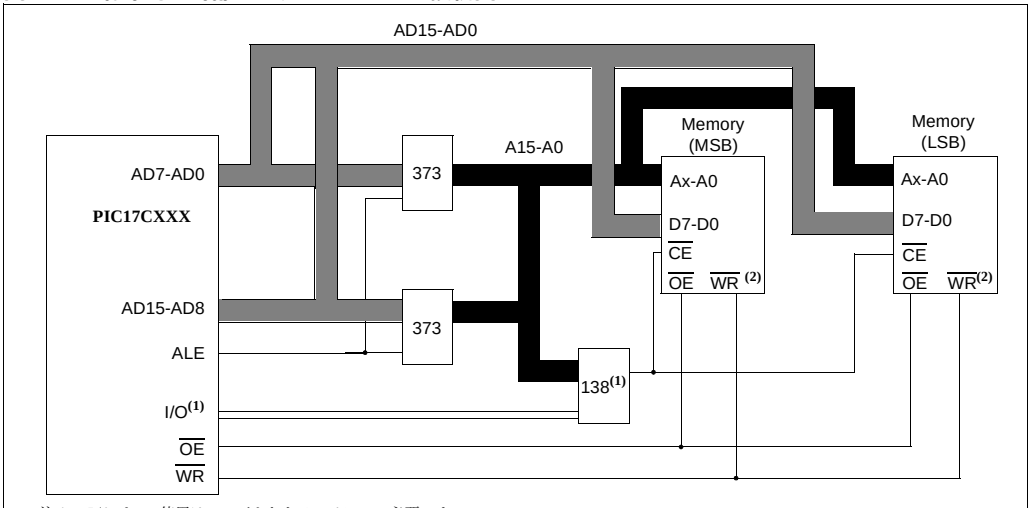
以下の選択は、Microchip 社の EPROM を使用する場合です。他社のメモリにインターフェイスする場合は、互換性のあるメモリデバイスと同様に、必要な PIC17C75X デバイスの電気的仕様を参照してください。

表 7-2: EPROM メモリアクセス時間の末尾のオーダリング

PIC17C75X Oscillator Frequency	Instruction Cycle Time (TCY)	EPROM Suffix
		PIC17C752 PIC17C756
8 MHz	500 ns	-25
16 MHz	250 ns	-15
20 MHz	200 ns	-10
25 MHz	160 ns	-70
33 MHz	121 ns	(1)

注 1: このアクセス時間は、速い SRAM を使う必要があります。

図 7-4: 標準的な外部プログラムメモリの接続図



注 1: I/O ピンの使用はページされたメモリにのみ必要です。
2: このシグナルは ROM と EPROM のデバイスには使用されません。

7.2 データメモリ構成

データメモリは 2 つのエリアに分割されます。1 つは汎用レジスタ (GPR) のエリアで、2 つめは特殊機能レジスタ (SFR) のエリアです。SFR はデバイスの動作を制御したりステータスを与えたりします。

データメモリの機能はバンクされており、これは両方のエリアで起こります。GPR エリアは汎用 RAM の 232 バイトより大きな値になることを可能にするためにバンクされます。

バンク選択のためには、制御ビットの使用が必要です。これらの制御ビットはバンクセレクトレジスタ (BSR) にあります。バンクされていない部分にアクセスされると、BSR ビットは無視されます。図 7-5 に、データメモリのマップ構成を示します。

MOVPF と MOVFP の命令は、周辺エリア (“P”) からレジスタファイル (“F”) のロケーションに (逆もあり)、値を動かすことを意味します。“P”レンジの定義は 0h から 1Fh で、“F”レンジは 0h から FFh です。“P”レンジは、汎用レジスタとして使用できる周辺レジスタよりも 6 つ多いロケーションがあります。これは、いくつかの応用でその変数を汎用RAMの他のロケーションにコピーする (例えば割り込み中にステータスの情報をセーブする) 必要がある時に有益です。

データメモリ全体は、ファイルセレクトレジスタ FSR0 と FSR1 を通して、直接または間接のどちらかでアクセスできます (7.4 章)。間接アドレシングは、データメモリのバンクされたエリアへのアクセスのために、適切な BSR の制御ビットを使用します。BSR についての詳細は、7.8 章で説明します。

7.2.1 汎用レジスタ (GPR)

すべてのデバイスに、いくつかの GPR エリアがあります。GPR は 8 ビット幅です。GPR が 232 より大きい時、追加したメモリスペースへのアクセスが可能になるようバンク化する必要があります。

すべての PIC17C75X デバイスには、GPR エリアにバンクされたメモリがあります。これらのバンクの間の切り替えを容易にするために、MOVLR バンク命令が命令セットに追加されています。GPRはパワーオンリセットでは初期化されず、他のすべてのリセットでも不変です。

7.2.2 特殊機能レジスタ (SFR)

SFR はデバイスの実行を制御するために、CPU と周辺機能により使われます (図 7-5)。これらのレジスタはスタティック RAM です。

SFR は 2 つに分類され、1 つは“コア”機能、もう 1 つは周辺機能と関係しています。“コア”に関連するレジスタは、この章で説明します。周辺機能に関連するレジスタは、各周辺機能の章で説明します。

周辺レジスタはメモリのバンクされた部分にあり、コアレジスタはバンクされていない部分にあります。周辺バンクの間の切り替えを容易にするために、MOVLB バンク命令が用意されています。

図 7-5: PIC17C75X のレジスタファイルマップ

Addr	Unbanked	Bank 1 (1)	Bank 2 (1)	Bank 3 (1)	Bank 4 (1)	Bank 5 (1)	Bank 6 (1)	Bank 7 (1)
00h	INDF0							
01h	FSR0							
02h	PCL							
03h	PCLATH							
04h	ALUSTA							
05h	TOSTA							
06h	CPUSTA							
07h	INTSTA							
08h	INDF1							
09h	FSR1							
0Ah	WREG							
0Bh	TMR0L							
0Ch	TMR0H							
0Dh	TBLPTRL							
0Eh	TBLPTRH							
0Fh	BSR							
	Bank 0							
10h	PORTA	DDRC	TMR1	PW1DCL	PIR2	DDRF	SSPADD	PW3DCL
11h	DDRB	PORTC	TMR2	PW2DCL	PIE2	PORTF	SSPCON1	PW3DCH
12h	PORTB	DDRD	TMR3L	PW1DCH	—	DDRG	SSPCON2	CA3L
13h	RCSTA1	PORTD	TMR3H	PW2DCH	RCSTA2	PORTG	SSPSTAT	CA3H
14h	RCREG1	DDRE	PR1	CA2L	RCREG2	ADCON0	SSPBUF	CA4L
15h	TXSTA1	PORTE	PR2	CA2H	TXSTA2	ADCON1	—	CA4H
16h	TXREG1	PIR1	PR3L/CA1L	TCON1	TXREG2	ADRESL	—	TCON3
17h	SPBRG1	PIE1	PR3H/CA1H	TCON2	SPBRG2	ADRESH	—	—
	Unbanked							
18h	PRODL							
19h	PRODH							
1Ah	General Purpose RAM							
1Fh								
	Bank 0 (2)	**Bank 1 (2)**	**Bank 2 (2, 3)**	**Bank 3 (2, 3)**				
20h	General Purpose RAM	General Purpose RAM	General Purpose RAM	General Purpose RAM				
FFh								

注 1: SFR ファイルのロケーション 10h-17h はバンクされています。BSR の下位は、バンクを設定します。バンクされていないすべての SFR は、バンクセレクトレジスタ (BSR) ビットを無視します。

2: 汎用レジスタ (GPR) のロケーション 20h-FFh、120h-1FFh、220h-2FFh、320h-3FFh はバンクされています。BSR の上位は、このバンクを設定します。すべての他の GPR は、バンクセレクトレジスタ (BSR) ビットを無視します。

3: これらの RAM のバンクは PIC17C752 ではインプリメントされていません。このバンクでどのレジスタを読んでも、'0' としてリードします。

表 7-3: 特殊機能レジスタ

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR, BOR での値	他のリセットでの値（注3）
Unbanked: 未知											
00h	INDF0	FSR0 の内容をアドレスデータメモリに使用（物理的には存在しません）								---- ----	---- ----
01h	FSR0	間接データメモリアドレスポインタ 0								xxxx xxxx	uuuu uuuu
02h	PCL	PC の下位 8 ビット								0000 0000	0000 0000
03h(1)	PCLATH	PC の上位 8 ビットの保持レジスタ								0000 0000	uuuu uuuu
04h	ALUSTA	FS3	FS2	FS1	FS0	OV	Z	DC	C	1111 xxxx	1111 uuuu
05h	T0STA	INTEDG	T0SE	T0CS	T0PS3	T0PS2	T0PS1	T0PS0	—	0000 000-	0000 000-
06h(2)	CPUSTA	—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	--11 1100	--11 qquu
07h	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
08h	INDF1	FSR1 の内容をアドレスデータメモリに使用（物理的には存在しません）								---- ----	---- ----
09h	FSR1	間接データメモリアドレスポインタ 1								xxxx xxxx	uuuu uuuu
0Ah	WREG	ワーキングレジスタ								xxxx xxxx	uuuu uuuu
0Bh	TMR0L	TMR0 レジスタ；下位バイト								xxxx xxxx	uuuu uuuu
0Ch	TMR0H	TMR0 レジスタ；上位バイト								xxxx xxxx	uuuu uuuu
0Dh	TBLPTRL	プログラムメモリテーブルポインタの下位バイト								0000 0000	0000 0000
0Eh	TBLPTRH	プログラムメモリテーブルポインタの上位バイト								0000 0000	0000 0000
0Fh	BSR	バンク選択レジスタ								0000 0000	0000 0000
Bank 0											
10h	PORTA	$\overline{RBPU}$	—	RA5/TX1/CK1	RA4/RX1/DT1	RA3/SDI/SDA	RA2/$\overline{SS}$/SCL	RA1/T0CKI	RA0/INT	0-xx xxxx	0-uu uuuu
11h	DDRB	PORTB のデータ方向レジスタ								1111 1111	1111 1111
12h	PORTB	RB7/SDO	RB6/SCK	RB5/TCLK3	RB4/TCLK12	RB3/PWM2	RB2/PWM1	RB1/CAP2	RB0/CAP1	xxxx xxxx	uuuu uuuu
13h	RCSTA1	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h	RCREG1	シリアルポート受信レジスタ								xxxx xxxx	uuuu uuuu
15h	TXSTA1	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
16h	TXREG1	シリアルポート送信レジスタ (USART1 用)								xxxx xxxx	uuuu uuuu
17h	SPBRG1	ボーレートジェネレータレジスタ (USART1 用)								xxxx xxxx	uuuu uuuu
Bank 1											
10h	DDRC	PORTC のデータ方向レジスタ								1111 1111	1111 1111
11h	PORTC	RC7/AD7	RC6/AD6	RC5/AD5	RC4/AD4	RC3/AD3	RC2/AD2	RC1/AD1	RC0/AD0	xxxx xxxx	uuuu uuuu
12h	DDRD	PORTD のデータ方向レジスタ								1111 1111	1111 1111
13h	PORTD	RD7/AD15	RD6/AD14	RD5/AD13	RD4/AD12	RD3/AD11	RD2/AD10	RD1/AD9	RD0/AD8	xxxx xxxx	uuuu uuuu
14h	DDRE	PORTE のデータ方向レジスタ								---- 1111	---- 1111
15h	PORTE	—	—	—	—	RE3/CAP4	RE2/$\overline{WR}$	RE1/$\overline{OE}$	RE0/ALE	---- xxxx	---- uuuu
16h	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	x000 0010	u000 0010
17h	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	0000 0000	0000 0000

凡例： x= 不定、u= 不変、－ = 未使用、‘0’としてリード、q= 条件によって決まる値。網掛け部分は未使用、‘0’としてリード。

注 1: プログラムカウンタの上位バイトは直接アクセスすることはできません。PCLATH は PC<15:8> の保持レジスタで、その内容はプログラムカウンタの上位バイトからアップデートされるか転送されます。

2: CPUSTA にある TO と PD のステータスビットは、MCLR リセットの影響を受けません。.

3: 他の（パワーアップ以外）リセットは、MCLR とウォッチドッグタイマリセットを通した外部リセットを含みます。

表 7-3: 特殊機能レジスタ (Cont.1d)

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR, BOR での値	他のリセットでの値（注3）
Bank 2											
10h	TMR1	Timer1 のレジスタ								xxxx xxxx	uuuu uuuu
11h	TMR2	Timer2 のレジスタ								xxxx xxxx	uuuu uuuu
12h	TMR3L	Timer3 のレジスタ；下位バイト								xxxx xxxx	uuuu uuuu
13h	TMR3H	Timer3 のレジスタ；上位バイト								xxxx xxxx	uuuu uuuu
14h	PR1	Timer1 の周期レジスタ								xxxx xxxx	uuuu uuuu
15h	PR2	Timer2 の周期レジスタ								xxxx xxxx	uuuu uuuu
16h	PR3L/CA1L	Timer3 の周期レジスタ；下位バイト / キャプチャ 1 レジスタ；下位バイト								xxxx xxxx	uuuu uuuu
17h	PR3H/CA1H	Timer3 の周期レジスタ；上位バイト / キャプチャ 1 レジスタ；上位バイト								xxxx xxxx	uuuu uuuu
Bank 3											
10h	PW1DCL	DC1	DC0	—	—	—	—	—	—	xx-- ----	uu-- ----
11h	PW2DCL	DC1	DC0	TM2PW2	—	—	—	—	—	xx0- ----	uu0- ----
12h	PW1DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
13h	PW2DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
14h	CA2L	キャプチャ 2 下位バイト								xxxx xxxx	uuuu uuuu
15h	CA2H	キャプチャ 2 上位バイト								xxxx xxxx	uuuu uuuu
16h	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/$\overline{PR3}$	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
Bank 4:											
10h	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	000- 0010	000- 0010
11h	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	000- 0000	000- 0000
12h	Unimplemented	—	—	—	—	—	—	—	—	---- ----	---- ----
13h	RCSTA2	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h	RCREG2	シリアルポート受信レジスタ (USART2 用)								xxxx xxxx	uuuu uuuu
15h	TXSTA2	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
16h	TXREG2	シリアルポート送信レジスタ (USART2 用)								xxxx xxxx	uuuu uuuu
17h	SPBRG2	ボーレートジェネレータ (USART2 用)								xxxx xxxx	uuuu uuuu
Bank 5:											
10h	DDRF	PORTF のデータ方向レジスタ								1111 1111	1111 1111
11h	PORTF	RF7/ AN11	RF6/ AN10	RF5/ AN9	RF4/ AN8	RF3/ AN7	RF2/ AN6	RF1/ AN5	RF0/ AN4	0000 0000	0000 0000
12h	DDRG	PORTG のデータ方向レジスタ								1111 1111	1111 1111
13h	PORTG	RG7/ TX2/CK2	RG6/ RX2/DT2	RG5/ PWM3	RG4/ CAP3	RG3/ AN0	RG2/ AN1	RG1/ AN2	RG0/ AN3	xxxx 0000	uuuu 0000
14h	ADCON0	CHS3	CHS2	CHS1	CHS0	—	GO/$\overline{DONE}$	—	ADON	0000 -0-0	0000 -0-0
15h	ADCON1	ADCS1	ADCS0	ADFM	—	PCFG3	PCFG2	PCFG1	PCFG0	000- 0000	000- 0000
16h	ADRESL	A/D 結果レジスタ下位バイト								xxxx xxxx	uuuu uuuu
17h	ADRESH	A/D 結果レジスタ上位バイト								xxxx xxxx	uuuu uuuu

凡例： x= 不定、u= 不変、－ = 未使用、‘0’としてリード、q= 条件によって決まる値。網掛け部分は未使用、‘0’としてリード。

注 1: プログラムカウンタの上位バイトは直接アクセスすることはできません。PCLATH は PC<15:8> の保持レジスタで、その内容はプログラムカウンタの上位バイトからアップデートされるか転送されます。

2: CPUSTA にある TO と PD のステータスビットは、MCLR リセットの影響を受けません。.

3: 他の（パワーアップ以外）リセットは、MCLR とウォッチドッグタイマリセットを通した外部リセットを含みます。

表 7-3: 特殊機能レジスタ (Cont.1d)

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR, BOR での値	他のリセットでの値（注3）
Bank 6:											
10h	SSPADD	I^2C スレーブモードでの SSP アドレスレジスタ。I^2C マスタモードでの SSP ボーレートリロードレジスタ。								0000 0000	0000 0000
11h	SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
12h	SSPCON2	GCEN	AKSTAT	AKDT	AKEN	RCEN	PEN	RSEN	SEN	0000 0000	0000 0000
13h	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	0000 0000
14h	SSPBUF	同期シリアルポート受信バッファ / 送信レジスタ								xxxx xxxx	uuuu uuuu
15h	未使用	—	—	—	—	—	—	—	—	---- ----	---- ----
16h	未使用	—	—	—	—	—	—	—	—	---- ----	---- ----
17h	未使用	—	—	—	—	—	—	—	—	---- ----	---- ----
Bank 7:											
10h	PW3DCL	DC1	DC0	TM2PW3	-	-	-	-	-	xx0- ----	uu0- ----
11h	PW3DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
12h	CA3L	キャプチャ 3 下位バイト								xxxx xxxx	uuuu uuuu
13h	CA3H	キャプチャ 3 上位バイト								xxxx xxxx	uuuu uuuu
14h	CA4L	キャプチャ 4 下位バイト								xxxx xxxx	uuuu uuuu
15h	CA4H	キャプチャ 4 上位バイト								xxxx xxxx	uuuu uuuu
16h	TCON3	—	CA4OVF	CA3OVF	CA4ED1	CA4ED0	CA3ED1	CA3ED0	PWM3ON	-000 0000	-000 0000
17h	未使用	—	—	—	—	—	—	—	—	---- ----	---- ----
Unbanked											
18h (5)	PRODL	16 ビット製品の下位バイト (8 × 8 のハードウェア掛け算)								xxxx xxxx	uuuu uuuu
19h (5)	PRODH	16 ビット製品の上位バイト (8 × 8 のハードウェア掛け算)								xxxx xxxx	uuuu uuuu

凡例： x= 不定、u= 不変、— = 未使用、‘0’としてリード、q= 条件によって決まる値。網掛け部分は未使用、‘0’としてリード。

注 1: プログラムカウンタの上位バイトは直接アクセスすることはできません。PCLATH は PC<15:8> の保持レジスタで、その内容はプログラムカウンタの上位バイトからアップデートされるか転送されます。

2: CPUSTA にある TO と PD のステータスビットは、MCLR リセットの影響を受けません。.

3: 他の（パワーアップ以外）リセットは、MCLR とウォッチドッグタイマリセットを通した外部リセットを含みます。

7.2.2.1 ALU STATUS レジスタ (ALUSTA)

ALUSTA レジスタは、演算と論理ユニットのステータスビットと間接アドレスレジスタ用のモード制御ビットを含んでいます。

ALUSTA レジスタは他のレジスタと同様に、命令のデスティネーションになることができます。ALUSTA レジスタが Z、DC、C ビットに影響する命令のデスティネーションである場合は、この 3 個のビットへのライトはできません。これらのビットはデバイスのロジックによってセットまたはクリアされます。したがってデスティネーションとしてALUSTAレジスタを伴う命令の結果が意図したものと異なることがあります。

例えば、CLRF ALUSTA は上位 4 ビットをクリアし、Z ビットをセットします。これによって ALUSTA レジスタは 0000 uluu(u= 不変) のままです。

したがって BCF、BSF、SWAPF、MOVWF の命令だけはどのステータスビットにも影響しないので、これらの命令だけをALUSTAレジスタを変えるのに使うことをお勧めします。他の命令がステータスビットにどのように影響を与えるかについては、" 命令セット一覧 "をご覧ください。

注意 1: C と DC のビットは、減算命令実行時には borrow と digit borrow ビットとして動作します。一例として SUBLW と SUBWF の命令をご覧ください。

注意 2: 2 の補数を取る結果が +127 を越えるか、-128 より少ない場合、オーバーフロービットがセットされます。

演算と論理ユニット (ALU) は、2 つまたはシングルのオペランドで演算と論理演算を実行することができます。すべてのシングルオペランド命令は WREG レジスタか与えられたファイルレジスタ上のどちらかで動作します。2 つのオペランド命令では、オペランドの 1 つが WREG レジスタ、もう一方がファイルレジスタか 8 ビットの直接の定数のどちらかです。

図 7-6: ALUSTA レジスタ (アドレス：04h、バンクされていない)

R/W - 1	R/W - 1	R/W - 1	R/W - 1	R/W - x	R/W - x	R/W - x	R/W - x
FS3	FS2	FS1	FS0	OV	Z	DC	C
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
-n = POR リセットでの値
(x = 不定)

bit 7-6: **FS3:FS2:** FSR1 モード選択ビット
00 = ポスト自動減分 FSR1 の値
01 = ポスト自動増分 FSR1 の値
1x = FSR1 の値は変化なし

bit 5-4: **FS1:FS0**: FSR0 モード選択ビット
00 = ポスト自動減分 FSR0 の値
01 = ポスト自動増分 FSR0 の値
1x = FSR0 の値は変化なし

bit 3: **OV**: オーバーフロービット
このビットは符号付演算用に使用します (2 の補数)。7 ビットのマグニチュードのオーバーフローを示し、状態を変化させるためにサインビット (ビット 7) になります。
1 =オーバーフローが符号付演算に対して起こった場合（この演算操作で）
0 =オーバーフローが起こらなかった場合

bit 2: **Z**: ゼロビット
1 =数値演算または論理演算の結果が 0
0 =数値演算または論理演算の結果が 0 以外

bit 1: **DC**: ディジットキャリィ / ボロウビット (Digit carry/$\overline{\text{borrow}}$ bit)
ADDWF、ADDLW 命令用
1 =結果の下位 4 ビット目からのキャリィが出力された場合
0 =結果の下位 4 ビット目からのキャリィが出力されない場合
注意：borrow では極性は逆です。

bit 0: **C**: キャリィ / ボロウビット (carry/$\overline{\text{borrow}}$ bit)
ADDWF、ADDLW 命令用
1 =結果の最上位ビットからのキャリィが出力された場合
注意: 減算は 2 番目のオペランドの 2 の補数を加えることにより実行されます。ローテート命令 (RRCF、RLCF) では、このビットはソースレジスタの上位ビットまたは下位ビットにロードされます。
0 =結果の最上位ビットからのキャリィが出力されない場合
注意：borrow では極性は逆です。

7.2.2.2 CPU STATUS レジスタ (CPUSTA)

CPUSTA レジスタは、CPU のステータスビットと制御ビットを含んでいます。このレジスタは、全体的に割込みをイネーブル / ディセーブルするために使われるビットを持っています。特定の割込みだけをイネーブル / ディセーブルすることが必要な場合は、INTerrput STAtus(INTSTA) レジスタと Peripheral Interrupt Enable (PIE) レジスタを参照してください。

CPUSTA レジスタは、スタックが利用でき、パワーダウン ($\overline{PD}$) とタイムアウト ($\overline{TO}$) ビットを含んでいるかどうかを示します。$\overline{TO}$、$\overline{PD}$、STKAV ビットは書き込みができません。これらのビットはデバイスのロジックによってセットまたはクリアされます。したがってデスティネーションとしてCPUSTAレジスタでの命令の結果が意図したものと異なることがあります。

$\overline{POR}$ ビットは、パワーオンリセット、外部 $\overline{MCLR}$ リセット、WDT リセットの間の区別が可能です。$\overline{BOR}$ ビットはブラウンアウトリセットが起こったかどうかを示します。

注意 1: ブラウンアウト回路がディセーブルの場合 (コンフィギュレーションワードの BODEN ビットをプログラムする時) BOR ステータスビットは無視され、必ずしも予知されません。

図 7-7: CPUSTA レジスタ (アドレス：06h、バンクされていない)

U - 0	U - 0	R - 1	R/W - 1	R - 1	R - 1	R/W - 0	R/W - 0
Ä|	Ä|	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
U = 未使用のビット、‘0’としてリード
- n = POR リセットでの値

bit 7-6: **Unimplemented**: 未使用：‘0’としてリード

bit 5: **STKAV**: スタックアベイラブルビット
このビットは、4 ビットのスタックポインタ値が Fh、または Fh から→ 0h にロールオーバーしていること (スタックオーバーフロー) を示します。
1 =スタックが利用できます。
0 =スタックがフルか、スタックオーバーフローが起こっています (このビットをスタックオーバーフローでクリアすると、デバイスのリセットだけがこのビットをセットします)。

bit 4: **GLINTD**: グローバルインターラプトディセーブルビット
このビットはすべての割込みをディセーブルします。割込みをイネーブルする時、イネーブルビットのセットされた要因だけが割込みをすることができます。
1 =すべての割込みをディセーブル
0 =マスクされていないすべての割込みをイネーブル

bit 3: **$\overline{TO}$**: WDT タイムアウトステータスビット
1 =パワーアップ後、または CLRWDT 命令を実行後
0 =ウォッチドッグタイマタイムアウトの発生

bit 2: **$\overline{PD}$**: パワーダウンステータスビット
1 =パワーアップ後、または CLRWDT 命令を実行後
0 =SLEEP 命令の実行後

bit 1: **$\overline{POR}$**: パワーオンリセットステータスビット
1 =パワーオンリセットが発生しなかった場合
0 =パワーオンリセットが発生した場合 (パワーオンリセット発生後、ソフトウェアによりセットが必要)

bit 0: **$\overline{BOR}$**: ブラウンアウトリセットステータスビット
1 =ブラウンアウトリセットが発生しなかった場合
0 =ブラウンアウトリセットが発生した場合 (ブラウンアウトリセット発生後、ソフトウェアによりセットが必要)

7.2.2.3 TMR0 STATUS/CONTROL レジスタ (T0STA)

このレジスタには様々な制御ビットがあります。ビット 7(INTEDG) はエッジの向きを制御するために使用され、RA0/INT ピンの信号のエッジ上で RA0/INT 割込みフラッグをセットします。その他のビットはタイマ 0 プリスケーラとクロックソースを設定します。

図 7-8: T0STA レジスタ (アドレス：05h、バンクされていない)

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	U - 0
INTEDG	T0SE	T0CS	T0PS3	T0PS2	T0PS1	T0PS0	—
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
U = 未使用のビット、'0' としてリード
-n = POR リセットでの値

bit 7: **INTEDG**: RA0/INT ピン割込みエッジ選択ビット
このビットはエッジの向きを選択し、そのエッジ上で割込みを検出します。
1 =RA0/INT ピンの立ち上がりエッジで割込みを発生
0 =RA0/INT ピンの立ち下がりエッジで割込みを発生

bit 6: **T0SE**: タイマ 0 クロック入力エッジ選択ビット
このビットはエッジの向きを選択し、そのエッジ上で TMR0 をインクリメントします。
T0CS=0 の時 (外部クロック)
1 =RA1/T0CKI ピンの立ち上がりエッジで TMR0 をインクリメントし、そして (または)T0CKIF 割込みを発生
0 =RA1/T0CKI ピンの立ち下がりエッジで TMR0 をインクリメントし、そして (または)T0CKIF 割込みを発生
T0CS=1 の時 (内部クロック)
ドントケア

bit 5: **T0CS**: タイマ 0 クロックソース選択ビット
このビットはタイマ 0 をクロックソースに選択します。
1 =内部命令クロックサイクル (T_{CY})
0 =T0CKI ピンからの外部クロック入力

bit 4-1: **T0PS3:T0PS0**: タイマ 0 プリスケール選択ビット
これらのビットはタイマ 0 のプリスケール値を選択します。

T0PS3:T0PS0	プリスケール値
0000	1:1
0001	1:2
0010	1:4
0011	1:8
0100	1:16
0101	1:32
0110	1:64
0111	1:128
1xxx	1:256

bit 0: **Unimplemented**: 未使用：'0' としてリード

7.3 スタックオペレーション

PIC17C75X デバイスには、16 × 16 ビットのハードウェアスタックがあります (図 7-1)。スタックはプログラムメモリまたはデータメモリ空間とは別に用意されており、スタックポインタは読み込み、書き込みともできません。CALL または LCALL 命令が実行されるか割込みが認められる場合、PC(プログラムカウンタ) がスタックに " PUSH " されます。スタックは RETURN、RETLW、RETFIE の命令実行時に " POP "されます。PCLATH は " PUSH " または " POP " 動作の影響を受けません。

スタックはサーキュラバッファとして動作し、スタックポインタはすべてのリセット後 ' 0 ' に初期化されます。スタックがオーバーフローしないことをソフトウエアで確認できるように、スタックアベイラブルビット (STKAV) があります。STKAV ビットはデバイスのリセット後セットされます。スタックポインタが Fh と等しい時、STKAV はクリアされます。スタックポインタが Fh から 0h にロールオーバーする時、STKAV はデバイスがリセットされるまでクリアを保持します。

注意 1: スタックのアンダーフローを示すステータスビットはありません。STKAV ビットはアンダーフローを検知するために使用することができます。その場合、スタックポインタはスタックのトップになります。

注意 2: PUSH または POP と呼ばれるニーモニック命令はありません。これらは CALL、RETURN、RETLW、RETFIE の命令の実行または割込みベクタに分岐することから発生する動作です。

注意 3: リセット後、" POP " 動作が " PUSH " 動作の前に発生した場合、STKAV ビットはクリアされますが、これはスタックがフルである (アンダーフローが発生した) ように見えます。" PUSH " 動作が次を (もう 1 つの " POP ") 発生させる場合、STKAV ビットはクリアのままです。このビットをセットすることができるのは、デバイスのリセットのみです。

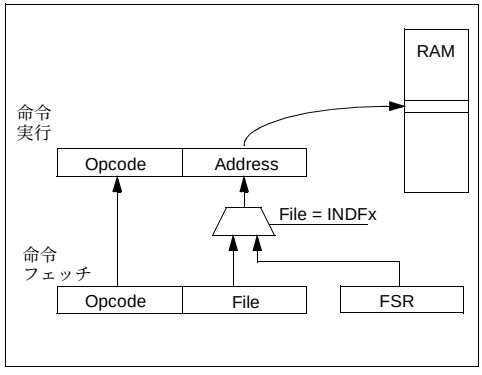
デバイスが 16 回 "PUSH" された後 ("POP" なしに)、17 回目のプッシュが 1 回目のプッシュの値をオーバーライトします。18 回目のプッシュは 2 回目のプッシュをオーバーライトします。

7.4 間接アドレシング

間接アドレシングは、データメモリをアドレシングするモードで、命令の中でデータメモリのアドレスは固定されていません。すなわちリードやライトをするためのレジスタはプログラムにより変更できます。これはデータメモリのデータテーブル用に有益です。図 7-9 に、間接アドレシングの動作を示します。これは、FSR レジスタの値により指定されたデータメモリのアドレスへの値の移動を示しています。

例 7-1 に、最小の命令数で RAM をクリアするための間接アドレシングの使用方法を示します。同様の考え方で、データの定義された数のバイト (ブロック) を USART 送信レジスタ (TXREG) に移動する時に使用できます。送信されたデータのブロックの始動アドレスは、簡単にプログラムにより変更できます。

図 7-9: 間接アドレシング



7.4.1 間接アドレシングレジスタ

PIC17C75X には、間接アドレシング用に 4 つのレジスタがあります。

- **INDF0 と FSR0**
- **INDF1 と FSR1**

INDF0 と INDF1 のレジスタは物理的にインプリメントされていません。これらのレジスタへのリードとライトは間接アドレシングを作動させますが、対応する FSR レジスタの値はデータのアドレスです。FSR は 8 ビットのレジスタで、256 バイトのデータメモリのアドレス幅の中ならどこへでもアドレシングが可能です。バンクされたメモリでは、アクセスするメモリのバンクは BSR の値により指定されます。

INDF0(または INDF1) のファイルを FSR により間接読み込みをする場合は、全て ' 0 ' が読み込まれます (ゼロビットをセット)。同様に INDF0(または INDF1) を間接書き込みする場合は、その動作は NOP と同じで、ステータスビットは影響されません。

7.4.2 間接アドレシングの動作

間接アドレシングの能力は PIC16CXX ファミリよりも拡張されています。それぞれの FSR レジスタに伴い、2 つの制御ビットがあります。これらの 2 つのビットは FSR レジスタを下記のように設定します。

- **間接アクセスの後 FSR の値 (アドレス) を自動的に減らす**
- **間接アクセスの後 FSR の値 (アドレス) を自動的に増やす**
- **間接アクセスの後 FSR の値 (アドレス) に変化なし**

これらの制御ビットは ALUSTA レジスタに置かれています。FSR1 レジスタが FS3:FS2 ビットにより制御され、FSR0 は FS1:FS0 ビットにより制御されます。

自動増分または自動減分機能を使用する場合、FSR への影響は ALUSTA レジスタには反映されません。例えば、間接アドレスにより FSR と ' 0 ' が等しくなっても、Z ビットはセットされません。

FSR レジスタが 0h の値を含んでいる場合、間接のライトが NOP と等しい間、間接リードが 0h を読み込みます (ゼロビットはセットされる)。(ステータスビットは影響を受けません)。

間接アドレシングは、全体のデータ空間の中でシングルサイクルのデータ転送が可能です。これは、MOVPF と MOVFP 命令を使用することにより可能で、' p '' f ' のどちらかが INDF0(または INDF1) として設定されます。

間接アドレスのソースとディストネーションがバンクされたメモリの中にある場合、アクセスされる場所は BSR の値により決まります。 .

アドレス 20h から FFh までの RAM をクリアするための簡単なプログラムを例 7-1 に示します。

例 7-1: 間接アドレシング

```
    MOVLW  0x20          ;
    MOVWF  FSR0          ; FSR0 = 20h
    BCF    ALUSTA, FS1   ; Increment FSR
    BSF    ALUSTA, FS0   ; after access
    BCF    ALUSTA, C     ; C = 0
    MOVLW  END_RAM + 1   ;
LP  CLRF   INDF0         ; Addr(FSR) = 0
    CPFSEQ FSR0          ; FSR0 = END_RAM+1?
    GOTO   LP            ; NO, clear next
    :                    ; YES, All RAM is
    :                    ; cleared
```

7.5 テーブルポインタ (TBLPTRL と TBLPTRH)

ファイルレジスタの TBLPTRL と TBLPTRH は、64K のプログラムメモリ空間をアドレスするために16ビットのポインタを構成します。テーブルポインタはTABLWT、TABLRD 命令により使用されます。

TABLRD、TABLWT 命令により、プログラムとデータ空間の間でデータ転送が可能です。テーブルポインタは、プログラムメモリの範囲内で働きます。データワードの 16 ビットアドレスのとして適しています。これらのレジスタのさらに詳細な説明やテーブルリード・テーブルライトの動作については、8.0 章をご覧ください。

7.6 テーブルラッチ (TBLATH、TBLATL)

テーブルラッチ (TBLAT) は 16 ビットのレジスタで、レジスタの高位バイト低位バイトに関連する TBLATH、TBLATL を伴います。それはデータメモリまたはプログラムメモリにマップされません。データがプログラムメモリとデータメモリ間を転送する間、テーブルラッチは一時的な保持ラッチとして使用されます(TABLRD、TABLWT、TLRD、TLWT 命令の説明を参照)。これらのレジスタのさらに詳細な説明やテーブルリード・テーブルライトの動作については、8.0 章をご覧ください。

7.7 プログラムカウンタモジュール

プログラムカウンタ (PC) は 16 ビットのレジスタです。PCL は PC の下位バイトで、データメモリにマップされます。PCL は、その他のレジスタと全く同じように読み込みと書き込みができるレジスタです。PCH は PC の上位バイトで、直接アドレスすることはできません。PCH はデータメモリやプログラムメモリにマップされないので、8 ビットのレジスタ PCLATH(PC の高位ラッチ) が、PC の上位バイト用の保持するラッチとして使用されます。PCLATH はデータメモリにマップされます。PCLATH を通して PCH のリードとライトができます。

16 ビットワイドの PC は、Q1 中にそれぞれの命令をフェッチした後増分されます (下記を除く)。

- **GOTO、CALL、LCALL、RETURN、RETLW、RETFIE 命令により変更した場合。**
- **割込み応答により変更した場合。**
- **PCL を対象とする命令により書き換えが行われた場合**
- **PCL にライトする予定のある場合**

“スキップ”は、スキップされたアドレスでの強制 NOP サイクルと同等です。

図 7-10 と 7-11 に、様々な状態でのプログラムカウンタの動作を示します。

図 7-10: プログラムカウンタの動作

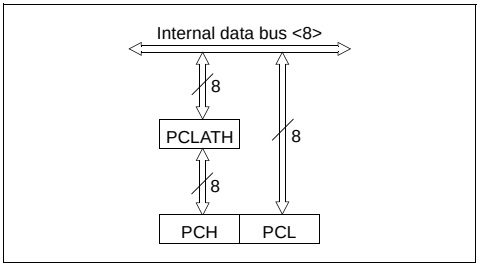


図 7-11: CALL、GOTO 命令を使用したプログラムカウンタ

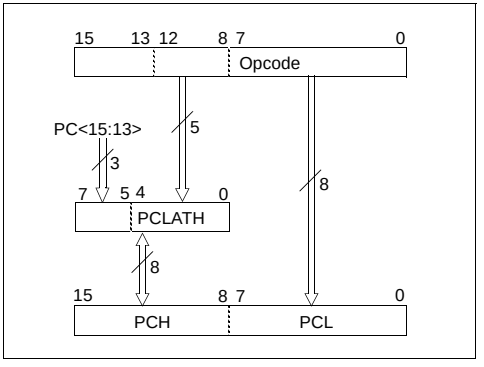


図 7-10 を使った場合、異なった命令での PC と PCLATH の動作は次のようになります。:

a) **LCALL 命令 :**
8 ビットのディストネーションアドレスが命令 (opcode) の中に用意されます。PCLATH は不変です。
PCLATH → PCH
Opcode<7:0> → PCL

b) **PCL での Read 命令 :**
PCL をリードする命令はどれでも。
PCL → データバス → ALU またはディストネーション
PCH → PCLATH

c) **PCL での Write 命令 :**
PCL にライトする命令はどれでも。
8-bit data → データバス → PCL
PCLATH → PCH

d) **PCL での Read-Modify-Write 命令 :**
PCL 上でリード - ライト - モディファイ動作をする命令はどれでも。たとえば ADDWF PCL.
Read: PCL → データバス → ALU
Write: 8-bit result → データバス → PCL
PCLATH → PCH

e) **RETURN 命令 :**
Stack<MRU> → PC<15:0>

図 7-11 を使った場合、GOTO と CALL 命令での PC と PCLATH の動作は次のようになります。:

f) **CALL, GOTO 命令 :**
13ビットのディストネーションアドレスが命令 (opcode) の中に用意されています。
Opcode<12:0> → PC<12:0>
PC<15:13> → PCLATH<7:5>
Opcode<12:8> → PCLATH<4:0>

リード - モディファイ - ライトは、その結果として PCL に影響を与えるだけです。PCH には PCLATH の値がロードされます。例えば、ADDWF PCL は現在のページの範囲内にジャンプすることになります。命令の前が PC=03F0h、WREG=30h、PCLATH=03h の場合、命令後は PC=0320h となります。本当に 16 ビットの計算したジャンプをするためには、16 ビットのディストネーションアドレスを計算し、PCLATH に上位バイトを書き込み、PCL に下位の値を書き込む必要があります。

次のような PC に関連した動作は PCLATH を変更しません。:

a) **LCALL, RETLW, RETFIE 命令の場合。**
b) **割込みベクタが強制的に PC に入った場合**
c) **PCL 上のリード - モディファイ - ライト命令 (例えば BSF PCL) の場合。**

7.8 バンク選択レジスタ (BSR)

BSR は、データメモリエリアの中のバンクを切り替えるために使用します (図 7-12)。PIC17C752 と PIC17C756 デバイスでは、バイト全体が使われます。下位の 4bit は周辺レジスタバンクを、上位の 4bit は汎用メモリバンクを選択するために使用します。

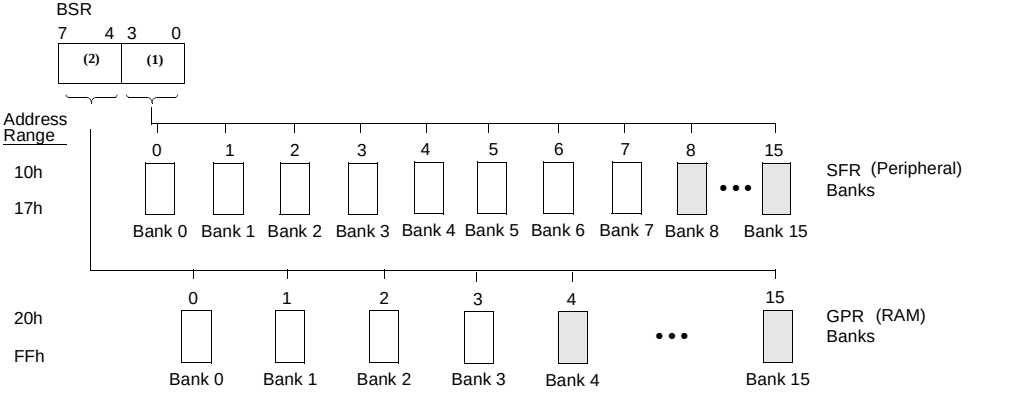
すべての特殊機能レジスタ (SFR) は、データメモリ空間にマップされています。多数のレジスタに適応させるために、バンキング方式を使用します。SFR のセグメントはアドレス 10h から 17h にバンクされています。バンク選択レジスタ (BSR) の下位 4bit は、現在アクティブな“周辺バンク”を選択します。1 つのバンク内に関連した機能の周辺レジスタを分類するように努力が払われています。しかし、シングルタスクに関連するすべての周辺機能をアドレスするためには、バンクからバンクに切り替えることがなお必要です。これを助けるために、MOVLB bank 命令が命令セットの中に含まれています。

大きな汎用メモリ空間が必要なことから、汎用 RAM バンキング方式を採用しました。BSR の上位 4bit は、現在アクティブな汎用 RAM バンクを選択します。これを助けるために、MOVLR bank 命令が命令セットの中に用意されています。

現在選択されたバンクがインプリメントされていない場合 (バンク 13 のような)、どのリードもすべてを‘0’と読み込みます。いかなるライトもビットバケットを満たされ、ALU ステータスビットは適当にセット / クリアされます。

注意： 特殊機能レジスタエリアのバンク 15 のレジスタは、Microchip 社用に指定されています。このバンクのレジスタをリードすると、そのためにランダムな値になることがあります。

図 7-12: BSR 動作



注 1: 0 から 7 のバンクだけがインプリメントされています。インプリメントされていないバンクを選ぶことはお勧めできません。
バンク 15 は、Microchip 社用に指定されています。このバンクのレジスタをリードすると、そのためランダムな値になることがあります。
2: バンク 0 とバンク 1 は PIC17C752 用にインプリメントされています。バンク 0 から 3 は PIC17C756 用にインプリメントされています。インプリメントされていないバンクを選ぶことはお勧めできません。.

NOTES:

8.0 テーブルリードとテーブルライト

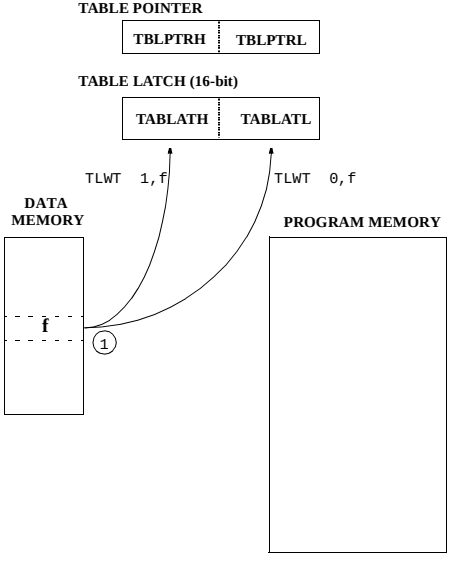
PIC17C75X にはプロセッサがデータメモリ空間からプログラムメモリ空間にデータを移動することが出来る 4 つの命令があります (逆も可)。プログラムメモリ空間は 16 ビット幅、データメモリ空間は 8 ビット幅なので、2 つの動作は 16 ビットの値をデータメモリへ (から) 移動することが必要です。

TLWT t,f と TABLWT t,i,f 命令は、データメモリ空間からプログラムメモリ空間へデータをライトする時に使用します。TLRD t,f と TABLRD t,i,f 命令は、プログラムメモリ空間からデータメモリ空間へデータをライトする時に使用します。

プログラムメモリは内部、外部とも可能です。プログラムメモリを外部へアクセスするためには、デバイスは拡張マイクロコントローラかマイクロプロセッサモードで動作していることが必要です。

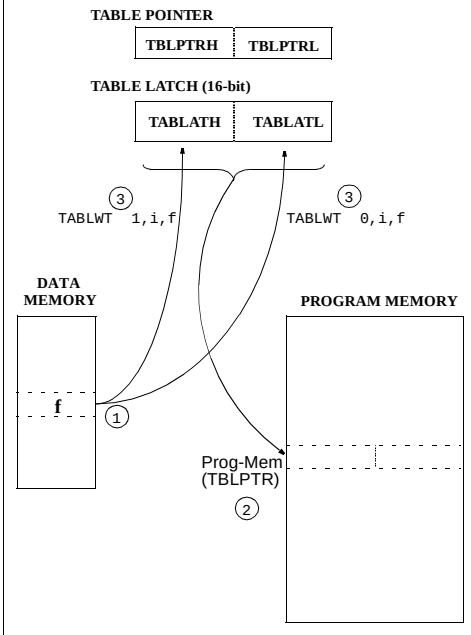
図 8-1 から 8-4 に、これら 4 つの命令の動作を示します。

図 8-1: TLWT 命令の動作



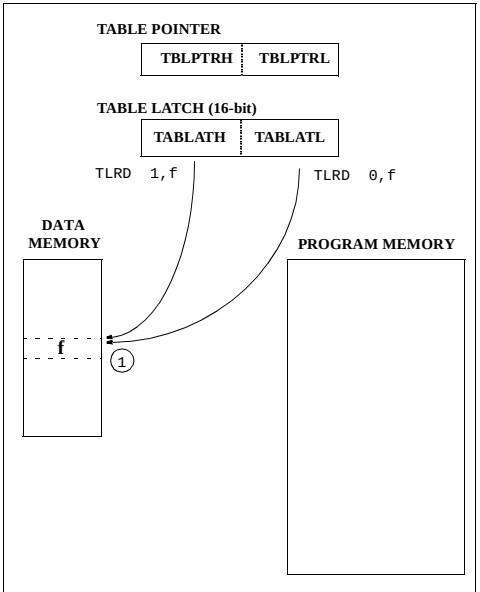
注 1: レジスタ ' f ' からの 8 ビットの値は、TABLAT (16 ビット) の上位または下位バイトにロードされます。

図 8-2: TABLWT 命令の動作



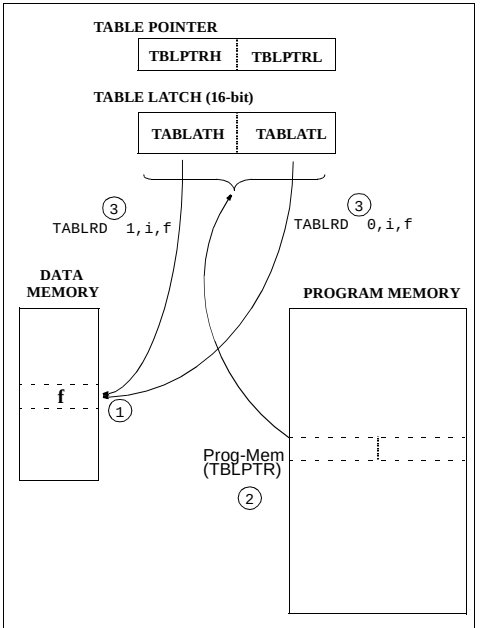
注 1: レジスタ ' f ' からの 8 ビットの値は、TABLAT(16 ビット) の上位または下位バイトにロードされます。
2: 16 ビットの TABLAT の値は、プログラムメモリ (TBLPTR) をアドレスするために書かれたものです。
3: " i " =1 なら、TBLPTR=TBLPTR+1、" i " =0 なら、TBLPTR は不変。

図 8-3: TLRD 命令の動作



注 1: TABLAT(16 ビット) の上位または下位バイトからの 8 ビットの値は、レジスタ ' f ' にロードされます。

図 8-4: TABLRD 命令の動作



注 1: TABLAT(16 ビット) の上位または下位バイトからの 8 ビットの値は、レジスタ ' f ' にロードされます。
2: プログラムメモリ (TBLPTR) の 16 ビットの値は、TABLAT レジスタにロードされます。
3: " i "=1 なら、TBLPTR=TBLPTR+1、" i "=0 なら、TBLPTR は不変。

8.1 内部メモリへのテーブルライト

内部メモリへのテーブルライトの動作は、ロングライト動作を引き起こします。ロングライトは、内部EPROM をプログラミングするために必要です。命令の実行はロングライトサイクルの間停止します。ロングライトは､イネーブルされた割込みにより終わります。EPROM がうまくプログラムされたことを確認するためには、最小のプログラミングタイムが必要です (仕様 # D114 参照)。ロングライトを終わらせるためにイネーブルされた 1 つの割込みだけを持つことにより、意図しない割込みがロングライトを途中で終わらせないようにします。

内部プログラムメモリのロケーションをプログラミングするためのイベントのシーケンスは、下記のとおりです。

1. EPROM プログラムのライトを終わらせる要因を除いて、すべての割込み要因をディセーブルにします。
2. $\overline{MCLR}$/Vpp ピンをプログラミングする電圧に引き上げます。
3. WDT をクリアします。
4. テーブルライトを実行します｡割込みはロングライトを終わらせます。
5. プログラムしたメモリロケーションをベリファイします (テーブルリード)。

注意 1: 要求通りのプログラミングが必要です。必要なデバイスの電気的仕様に書かれているタイミング仕様を参照してください。これらの仕様 (温度を含む) を守らないと、EPROM のロケーションに正常にデータがプログラムされない結果となり、状態のオーバータイムを失うことがあります。

注意 2: Vpp の電圧レベルが十分でない場合、テーブルライトは 2 サイクルライトで、プログラムメモリは変化しません。

8.1.1 ロングライトの終了

割込み要因またはリセットが、ロングライトの動作を終わらせる唯一のイベントです。割込み要因からロングライトを終わらせるためには割込みイネーブルビットとフラグビットがセットされることが必要です。GLINTD ビットだけが割込みベクタアドレスへの分岐をイネーブルにします。

TOCKI、RA0/INT、TMR0 割込み要因がロングライトを終わらせるために使われる場合、もっとも高い優先権のあるイネーブルされた割込みのインターラプトフラグが、ロングライトを終わらせ、自動的にクリアされます。

注意 1: 割込みがペンディングの場合､TABLWT は打ち切られます (NOP を実行)。イネーブルされている TOCKI、RA0/INT、TMR0 要因からの、もっとも高い優先権を持つペンディングの割込みがそのフラグをクリアします。

注意 2: 割込みがプログラムのライトタイミングに使用されていない場合、割込みをディセーブルする必要があります。これにより割込みが失われず、またロングライトを早く終わらせないようにします。

周辺割込みの要因をロングライトを終わらせるために使用する場合、割込みのイネーブルビットとフラグビットをセットする必要があります｡割込みフラグは、割込みベクタアドレスに分岐することで自動的にはクリアされません。

GLINTD ビットは、ロングライトが終わる時、プログラムが割込みベクタに分岐するかどうかを決定します。GLINTD をクリアするとプログラムは分岐され、GLINTDをセットするとプログラムは割込みベクタアドレスには分岐しません。

表 8-1: 割込み - テーブルライトの相互作用

Interrupt Source	GLINTD	Enable Bit	Flag Bit	Action
RA0/INT, TMR0, T0CKI	0	1	1	ロングテーブルライトを終了 (内部プログラムメモリに)、割込みベクトルに分岐 (分岐がフラグビットをクリア)。
	0	1	0	なし
	1	0	x	なし
	1	1	1	テーブルライトを終了、割込みベクトルに分岐しない (フラグが自動的にクリア)。
Peripheral	0	1	1	テーブルライトを終了、割込みベクトルに分岐。
	0	1	0	なし
	1	0	x	なし
	1	1	1	テーブルライトを終了、割込みベクトルに分岐しない (フラグがセットされたまま)。

8.2 外部メモリへのテーブルライト

外部メモリへのテーブルライトは、常に 2 サイクル命令です。2 番目のサイクルがデータを外部メモリのロケーションに書き込みます。外部メモリをライトするためのイベントのシーケンスは、内部メモリのライトと同様です。

注意： 割込みがペンディングの場合、または TABLWT の間に起こる場合、2 サイクルのテーブルライトが終了します。RA0/INT、TMR0、TOCKI の割込みフラグは自動的にクリアされるか、ペンディングの周辺割込みが認知されます。

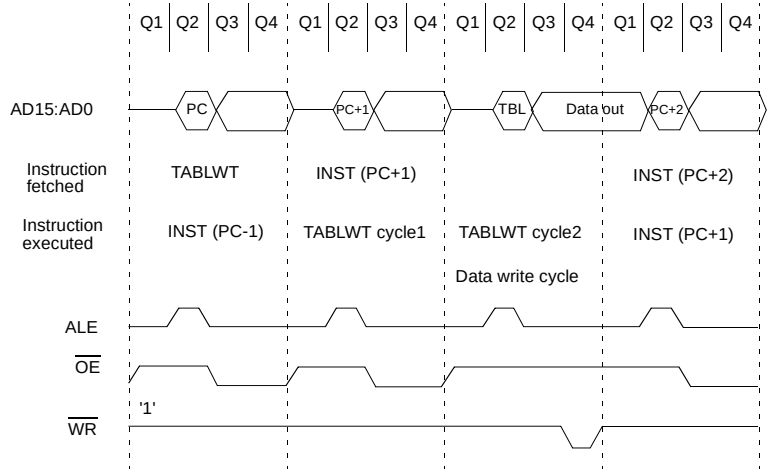
8.2.2 テーブルライトコード

TABLWT 命令の “ i ”オペランドは、16 ビット TBLPTR レジスタの値が自動的に増分するよう (次のライト用) に設定できます。 例 8-1 では、TBLPTR レジスタは自動的には増分しません。

例 8-1: テーブルライト

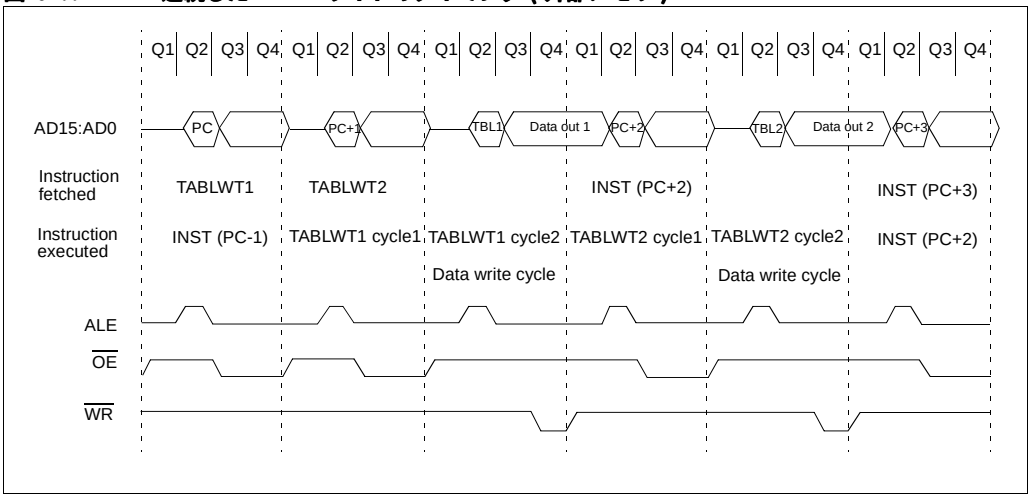
```
CLRWDT                  ; Clear WDT
MOVLW   HIGH (TBL_ADDR) ; Load the Table
MOVWF   TBLPTRH         ;   address
MOVLW   LOW (TBL_ADDR)  ;
MOVWF   TBLPTRL         ;
MOVLW   HIGH (DATA)     ; Load HI byte
TLWT    1, WREG         ;   in TABLATH
MOVLW   LOW (DATA)      ; Load LO byte
TABLWT  0,0,WREG        ;   in TABLATH
                        ;   and write to
                        ;   program memory
                        ;   (Ext. SRAM)
```

図 8-5: TABLWT ライトのタイミング （外部メモリ）



注意： 外部のライトで、GLINTD=‘1’、イネーブルビット =‘1’の場合、‘1’→フラグビットの時はテーブルライトを実行。もっとも上位のペンディング割込みがクリアされます。

図 8-6: 連続した TABLWT ライトのタイミング (外部メモリ)



8.3 テーブルリード

テーブルリードによりプログラムメモリを読み込むことが可能です。これにより定数がプログラムメモリ空間の中に保存され、必要時にデータメモリに取り出すことが可能です。例 8-2 はプログラムメモリアドレス TBLPTR での 16 ビットの値のリードです。ダミーのバイトが TABLATH からリードされた後、その TABLATH はプログラムメモリアドレス TBLPTR+1 から 16 ビットのデータをロードします。最初のリードはデータをラッチにロードし、ダミーのリードとみなすことが可能です (未知のデータが ' f ' にロードされる)。INDF0 は、自動増分または自動減分のどちらかに設定する必要があります。

例 8-2: テーブルリード

```
MOVLW   HIGH (TBL_ADDR) ; Load the Table
MOVWF   TBLPTRH         ;   address
MOVLW   LOW (TBL_ADDR)  ;
MOVWF   TBLPTRL         ;
TABLRD  0,0,DUMMY     ; Dummy read,
                      ;  Updates TABLATH
TLRD    1, INDF0      ; Read HI byte
                      ;   of TABLATH
TABLRD  0,1,INDF0     ; Read LO byte
                      ;   of TABLATH and
                      ;   Update TABLATH
```

図 8-7: TABLRD のタイミング

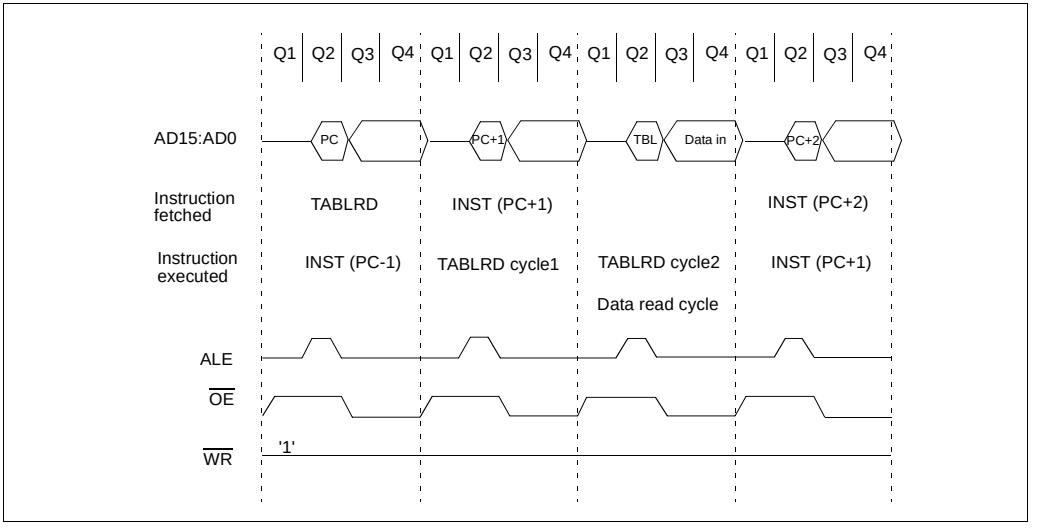
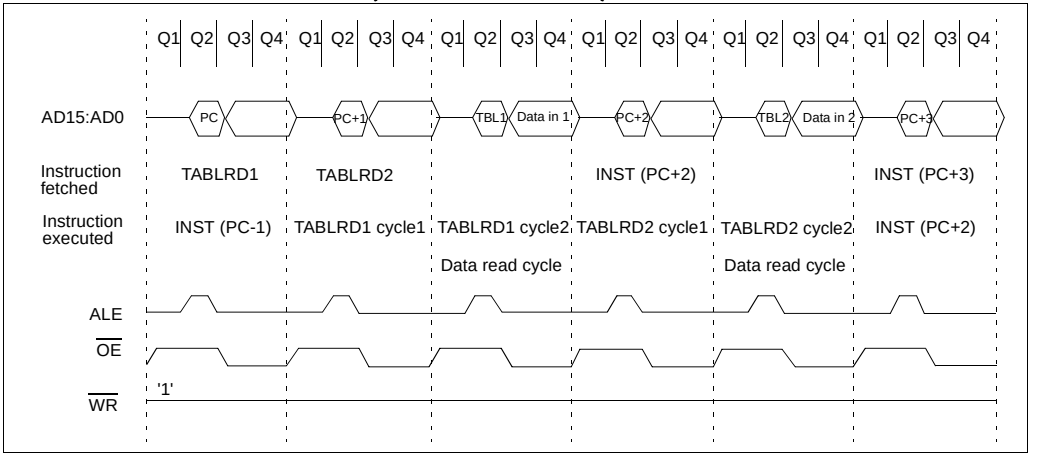


図 8-8: TABLRD のタイミング (連続した TABLRD 命令)



9.0 ハードウェアマルチプライア

PIC17C75X のデバイスにはすべて、デバイスの ALU に含まれる 8 × 8 のハードウェアマルチプライアがあります。ハードウェアで乗算をすることにより、必要な命令サイクルは 1 サイクルのみです。これは 16bit の結果を与える符号無し乗算です。その結果は 16bit の PRODuct レジスタ (PRODH:PRODL) に保存されます。乗算器は、ALUSTA レジスタのどのフラグにも影響を与えません。

8 × 8 のマルチプライアをシングルサイクルで実行させることにより、次のような利点があります。

- **より高度に計算できるスループット**
- **乗算のアルゴリズムに必要なコードサイズの縮小**

高性能化により、このデバイスを DSP(Digital Signal Processor)として応用することが出来るようになりました。

表 9-1 に、シングルサイクルのハードウェア乗算器を使った PIC17CXXX デバイスと、ハードウェア乗算器なしで同じ機能を実行する場合との性能の比較を示します。

例 9-1 に、8 × 8 の符号無し乗算をするためのシーケンスを示します。乗算の 1 つのアーギュメントがすでに WREG レジスタにロードされている場合、必要とされるのは 1 つの命令だけです。

例 9-2 に、8 × 8 の符号付き乗算をするためのシーケンスを示します。引数のサインビットを判断するためには、それぞれのアーギュメントの最上位ビット (MSb) をテストし、適切な減算を行ないます。

例 9-1: 8 × 8 の符号無し乗算ルーチン

```
MOVFP   ARG1, WREG  ;
MULWF   ARG2        ; ARG1 * ARG2 ->
                    ;   PRODH:PRODL
```

例 9-2: 8×8の符号付き乗算ルーチン

```
MOVFP   ARG1, WREG
MULWF   ARG2        ; ARG1 * ARG2 ->
                    ;   PRODH:PRODL
BTFSC   ARG2, SB    ; Test Sign Bit
SUBWF   PRODH, F    ; PRODH = PRODH
                    ;         - ARG1
MOVFP   ARG2, WREG
BTFSC   ARG1, SB    ; Test Sign Bit
SUBWF   PRODH, F    ; PRODH = PRODH
                    ;         - ARG2
```

表 9-1: 性能の比較

ルーチン	乗算の方法	プログラムメモリ (Words)	サイクル (Max)	時間 @ 33 MHz
8 x 8 符号無し	ハードウエア乗算器を使用しない	13	69	8.364 ms
	ハードウエア乗算器を使用	1	1	0.121 ms
8 x 8 符号付き	ハードウエア乗算器を使用しない	—	—	—
	ハードウエア乗算器を使用	6	6	0.727 ms
16 x 16 符号無し	ハードウエア乗算器を使用しない	21	242	29.333 ms
	ハードウエア乗算器を使用	24	24	2.91 ms
16 x 16 符号付き	ハードウエア乗算器を使用しない	52	254	30.788 ms
	ハードウエア乗算器を使用	36	36	4.36 ms

例 9-3 に、16 × 16 の符号無し乗算をするためのシーケンスを示します。式 9-1 は、使用したアルゴリズムを示します。32bit の結果は、4 つのレジスタ RES3:RES0 に保存されます。

式 9-1: 16 × 16 の符号無し乗算のアルゴリズム

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L)
\end{aligned}
$$

例 9-3: 16 × 16 の符号無し乗算ルーチン

```
   MOVFP    ARG1L, WREG
   MULWF    ARG2L       ; ARG1L * ARG2L ->
                        ;   PRODH:PRODL
   MOVPF    PRODH, RES1 ;
   MOVPF    PRODL, RES0 ;
;
   MOVFP    ARG1H, WREG
   MULWF    ARG2H       ; ARG1H * ARG2H ->
                        ;   PRODH:PRODL
   MOVPF    PRODH, RES3 ;
   MOVPF    PRODL, RES2 ;
;
   MOVFP    ARG1L, WREG
   MULWF    ARG2H       ; ARG1L * ARG2H ->
                        ;   PRODH:PRODL
   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F     ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F     ;
   CLRF     WREG, F     ;
   ADDWFC   RES3, F     ;
;
   MOVFP    ARG1H, WREG ;
   MULWF    ARG2L       ; ARG1H * ARG2L ->
                        ;   PRODH:PRODL
   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F     ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F     ;
   CLRF     WREG, F     ;
   ADDWFC   RES3, F     ;
```

例 9-4 に、16 × 16 の符号付き乗算をするためのシーケンスを示します。方程式 9-2 は、使用したアルゴリズムを示します。32bit の結果は、4 つのレジスタ RES3:RES0 に保存されます。アーギュメントのサインビットを数えるためには、それぞれのアーギュメントペアの最上位ビット (MSb) をテストし、適切な減算を行ないます。

式 9-2: 16 × 16 の符号付き乗算のアルゴリズム

RES3:RES0

= ARG1H:ARG1L • ARG2H:ARG2L

= (ARG1H • ARG2H • 2^{16}) +
(ARG1H • ARG2L • 2^{8}) +
(ARG1L • ARG2H • 2^{8}) +
(ARG1L • ARG2L) +
(-1 • ARG2H<7> • ARG1H:ARG1L • 2^{16}) +
(-1 • ARG1H<7> • ARG2H:ARG2L • 2^{16})

例 9-4: 16×16の符号付き乗算ルーチン

```
   MOVFP    ARG1L, WREG
   MULWF    ARG2L       ; ARG1L * ARG2L ->
                        ;   PRODH:PRODL
   MOVPF    PRODH, RES1 ;
   MOVPF    PRODL, RES0 ;
;
   MOVFP    ARG1H, WREG
   MULWF    ARG2H       ; ARG1H * ARG2H ->
                        ;   PRODH:PRODL
   MOVPF    PRODH, RES3 ;
   MOVPF    PRODL, RES2 ;
;
   MOVFP    ARG1L, WREG
   MULWF    ARG2H       ; ARG1L * ARG2H ->
                        ;   PRODH:PRODL
   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F     ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F     ;
   CLRF     WREG, F     ;
   ADDWFC   RES3, F     ;
;
   MOVFP    ARG1H, WREG ;
   MULWF    ARG2L       ; ARG1H * ARG2L ->
                        ;   PRODH:PRODL
   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F     ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F     ;
   CLRF     WREG, F     ;
   ADDWFC   RES3, F     ;
;
   BTFSS    ARG2H, 7    ; ARG2H:ARG2L neg?
   GOTO     SIGN_ARG1   ; no, check ARG1
   MOVFP    ARG1L, WREG ;
   SUBWF    RES2        ;
   MOVFP    ARG1H, WREG ;
   SUBWFB   RES3
;
SIGN_ARG1
   BTFSS    ARG1H, 7    ; ARG1H:ARG1L neg?
   GOTO     CONT_CODE   ; no, done
   MOVFP    ARG2L, WREG ;
   SUBWF    RES2        ;
   MOVFP    ARG2H, WREG ;
   SUBWFB   RES3
;
CONT_CODE
     :
```

NOTES:

10.0 I/O ポート

PIC17C75X デバイスは PORTA から PORTG の 7 つの I/O ポートを持っています。PORTB から PORTG には対応するデータディレクションレジスタ (DDR) があり、それらは入力または出力としてポートピンを設定するのに使用されます。これら 7 つのポートは 50 本の I/O ピンから成っています。これらのポートピンのいくつかは、代替機能で多重化されています。

PORTC、PORTD、PORTE はシステムバスで多重化されています。デバイスのコンフィギュレーションビットが、マイクロプロセッサモードまたは拡張マイクロコントローラモードに選択されている時は、これらのピンはシステムバスとして設定されます。他の 2 つのマイクロコントローラモードでは、これらのピンは汎用 I/O です。

PORTA、PORTB、PORTE<3>、 PORTF、PORTG はデバイスの周辺機能と多重化されています。これらの周辺機能は下記の通りです。

- **タイマモジュール**
- **キャプチャモジュール**
- **PWM モジュール**
- **USART/SCI モジュール**
- **SSP モジュール**
- **A/D モジュール**
- **外部割込みピン**

これらの周辺モジュールのいくつかがオンになると、ポートピンは自動的に代替機能を設定します。これを行なうモジュールは下記の通りです。

- **PWM モジュール**
- **SSP モジュール**
- **USART/SCI モジュール**

周辺モジュールによりピンが自動的に出力として設定される場合、ピンのデータディレクション(DDR)ビットは不定です。周辺モジュールをディセーブルした後、DDR ビットを必要な設定に再度初期化する必要があります。

他の周辺モジュール (入力を必要とする) は、それらのデータディレクションビットを適切に設定しなければなりません。

注意： 周辺入力であるピンは、出力として設定できます (DDRx<y> はクリア)。周辺のイベントはポートピンのアクション出力により決定されます。

10.1 PORTA レジスタ

PORTA は 6bit 幅のラッチです。PORTA には、対応するデータディレクションレジスタ(DDR)はありません。

PORTA を読み込むと、そのピンのステータスを読み出します。

RA1 ピンは TMR0 クロック入力と、RA2 と RA3 は SSP 機能と、RA4 と RA5 は USART1 機能と多重化されます。RA2、RA3、RA4、RA5 出力としての制御は、それぞれの多重化された周辺モジュールにより自動的に設定されます。

10.1.1 出力としての RA2、RA3 の使用法

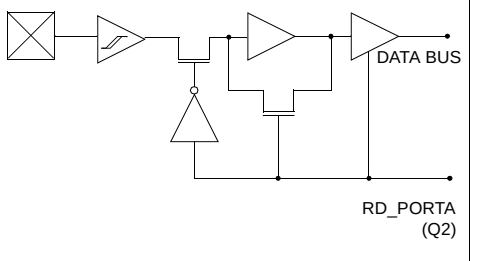
RA2 と RA3 ピンはオープンドレイン出力です。出力として RA2 と (または) RA3 ピンを使うには、単に POTRA レジスタに必要な値を書き込むだけです。‘ 0 ’ はピンをローで動かし、‘ 1 ’ はピンをフロート (高インピーダンス) にします。外部プルアップ抵抗器はピンをハイするために使用する必要があります。RA2 と RA3 ピンへのライトは、他の PORTA ピンには影響を与えません。

注意： 出力として RA2 や RA3 ピンを使う時は、PORTA でのリード - モディファイ - ライト命令 (例えば BCF、BSF、BTG) はお勧めできません。

それは、ポートピンを読み込み、必要な動作をし、それからこの値をデータラッチに書き込む動作です。これは不注意に RA2 や RA3 ピンが入力から出力に (逆もあり) 切り替わる場合があります。

この可能性をさけるために、PORTA 用のシャドウレジスタを使用します。このシャドウレジスタ上でビット動作をし、それを PORTA に移動します。

図 10-1: RA0 と RA1 のブロック図



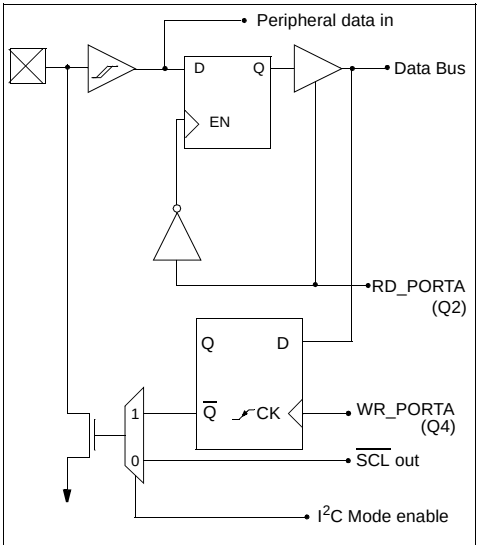
注意： I/O ピンは V_{DD} および V_{SS} に対する保護ダイオード付。

例 10-1 に、PORTA を初期化するための命令のシーケンスを示します。そのポートを初期化するためには、バンクセレクトレジスタ (BSR) をバンク 0 に選択する必要があります。次の例は、バンク切り換えのために MOVLB 命令を使用して BSR レジスタにデータをロードするものです。

例 10-1: PORTA の初期化

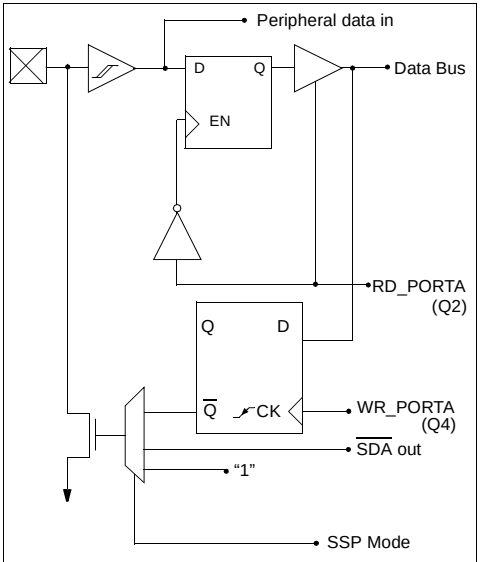
```
    MOVLB  0      ; Select Bank 0
    MOVLW  0xF3   ;
    MOVPF  PORTA  ; Initialize PORTA
                  ;   RA<3:2> are output low
                  ;   RA<5:4> and RA<1:0>
                  ;   are inputs
                  ;   (outputs floating)
```

図 10-2: RA2 のブロック図



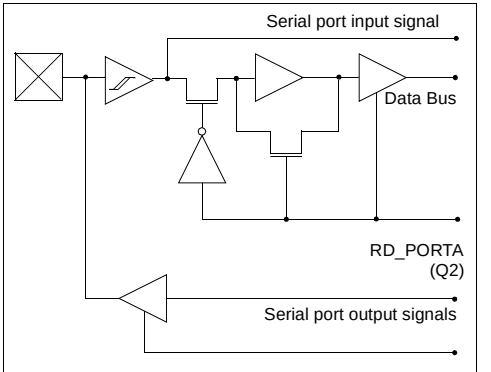
注意:I/O ピンは Vss に対する保護ダイオード付き。

図 10-3: RA3 のブロック図



注意:I/O ピンは Vss に対する保護ダイオード付き。

図 10-4: RA4 と RA5 のブロック図



$\overline{OE}$ = SPEN,SYNC,TXEN, $\overline{CREN}$, $\overline{SREN}$ for RA4

$\overline{OE}$ = SPEN ($\overline{SYNC}$+SYNC,$\overline{CSRC}$) for RA5

注意:I/O ピンは VDD と Vss に対する保護ダイオード付き。

表 10-1: PORTA の機能

名称	Bit0	Buffer Type	機能
RA0/INT	bit0	ST	入力または外部割込み入力
RA1/TOCKI	bit1	ST	入力または TMR0 タイマ / カウンタへのクロック入力、及び (または) 外部割込み入力。
RA2/$\overline{SS}$/SCL	bit2	ST	入出力または SPI 用のスレーブ選択入力または I^2C バス用のクロック入力。出力はオープンドレインタイプ。
RA3/SDI/SDA	bit3	ST	入出力または SPI 用のデータ入力または I^2C バス用のデータ。出力はオープンドレインタイプ。
RA4/RX1/DT1	bit4	ST	入出力または USART1 非同期受信、または USART1 同期データ。
RA5/TX1/CK1	bit5	ST	入出力または USART1 非同期送信、または USART1 同期クロック。
$\overline{RBPU}$	bit7	—	PORTB ウィークプルアップ用の制御ビット

凡例： ST= シュミットトリガ入力

表 10-2: PORTA に関連するレジスタ / ビット

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR, BOR での値	他のリセットでの値 (注 1)
10h, Bank 0	PORTA	$\overline{RBPU}$	—	RA5/ TX1/CK1	RA4/ RX1/DT1	RA3/ SDI/SDA	RA2/ SS/SCL	RA1/TOCKI	RA0/INT	0-xx xxxx	0-uu uuuu
05h, Unbanked	TOSTA	INTEDG	T0SE	T0CS	PS3	PS2	PS1	PS0	—	0000 000-	0000 000-
13h, Bank 0	RCSTA1	SPEN	RC9	SREN	CREN	—	FERR	OERR	RC9D	0000 -00x	0000 -00u
15h, Bank 0	TXSTA1	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u

凡例： x= 不定、u= 不変、－ = 未使用、‘0’ としてリード。網掛けの部分は PORTA では使われません。

注 1: 他の (パワーアップ以外) リセットは、MCLR を通る外部リセットとウォッチドッグタイマリセットを含みます。

10.2 PORTB と DDRB レジスタ

PORTB は 8 ビット幅の双方向性のポートです。対応するデータディレクションレジスタは DDRB です。DDRB の‘1’が入力として対応するポートピンを設定します。DDRB レジスタの‘0’が出力として対応するポートピンを設定します。PORTB を読み込むと、そのピンのステータスを読み出し、それに書き込むとポートラッチに書き入れます。

PORTB の各ピンには内部ウィークプルアップがあります。1本の制御ビットですべてのプルアップをオンにすることができます。これは $\overline{\text{RBPU}}$(PORTA<7>) ビットをクリアすることにより可能です。ウィークプルアップはポートピンが出力として設定された時は自動的にオフになります。このプルアップはどんなリセット時でもイネーブルされます。

PORTB にもレベル変化割込み機能があります。入力として設定されたピンのみこの割込みを発生させることはできます (すなわち、出力として設定された RB7:RB0 ピンがレベル変化割込みから除外されます)。入力ピン (RB7:RB0 の) は、PORTB のデータラッチの値と比較されます。RB7:RB0 のミスマッチ出力は、PORTB 割込みフラグビット RBIF(PIR1<7>) をセットするために、一緒に OR されます。

この割込みによりデバイスを SLEEP からウェークすることができます。次の方法により割込みサービスルーチンで割込みをクリアすることができます。

a) PORTB のリード - ライト (例えば MOVPF PORTB、PORTB)。これによりミスマッチ状態が終了します。

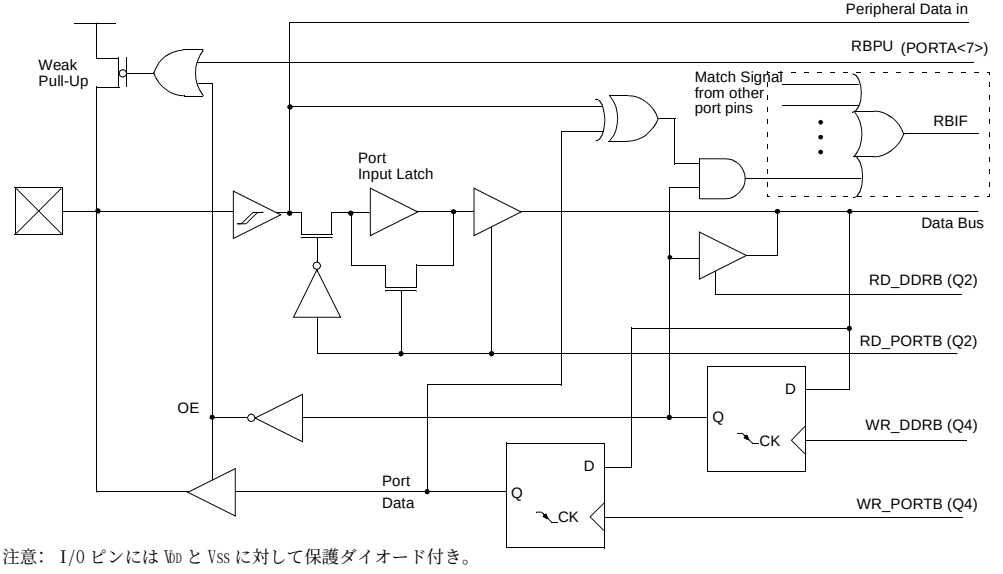
b) 次に RBIF ビットをクリアします。

ミスマッチ状態は RBIF ビットをセットし続けます。PORTB のリードとライトによりミスマッチ状態が終了しますので、RBIFビットをクリアすることができます。

このミスマッチの割込み機能は、このポートのソフトウェアで設定可能なプルアップにより、キーパッドとのインターフェイスを容易にし、キーを押すことによるウェークアップを可能にします。一例として、アプリケーションノート AN552“キーストロークにおけるウェークアップのインプリメンティング”を参照してください。

レベル変化割込みは、PORTB をレベル変化割込みにのみ使用している場合や、キーを押した時のウェークアップの使用にお勧めします。

図 10-5: RB5:RB4 と RB1:RB0 ポートピンのブロック図



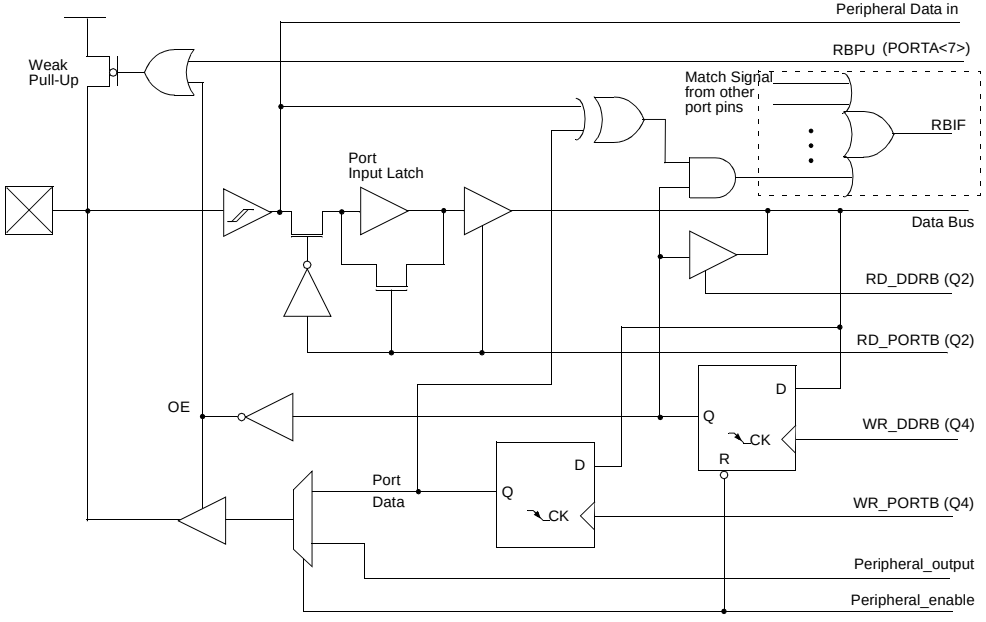
注意: I/O ピンには VDD と VSS に対して保護ダイオード付き。

例 10-2 に、PORTB を初期化するための命令のシーケンスを示します。そのポートを初期化するためには、バンクセレクトレジスタ (BSR) をバンク 0 に選択する必要があります。次の例は、MOVLB 命令を使用して BSR レジスタにデータをロードするものです。

例 10-2: PORTB の初期化

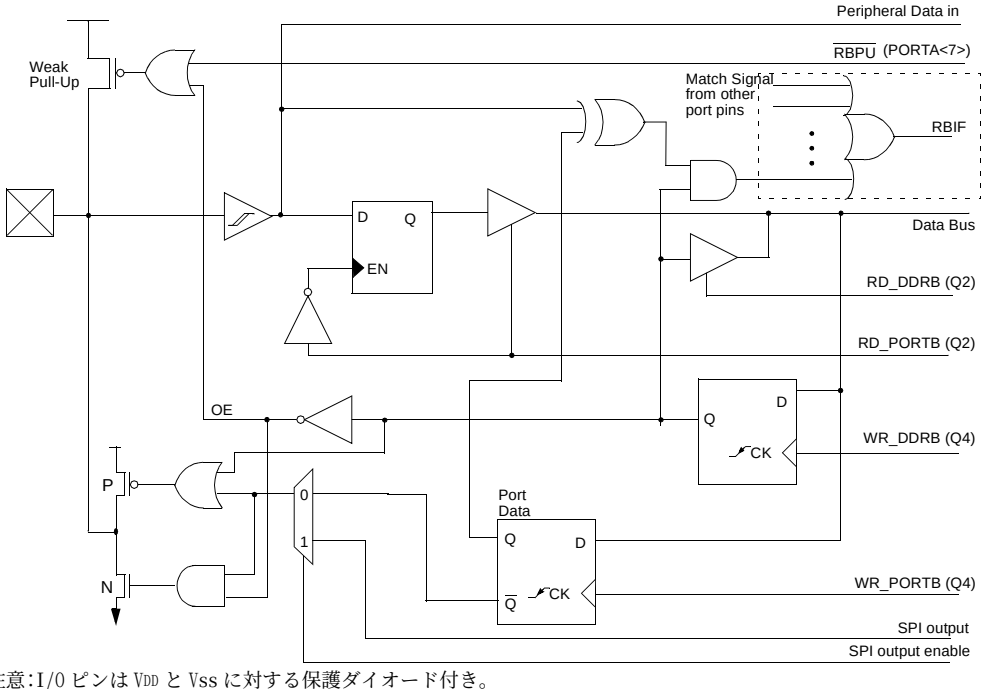
```
MOVLB   0      ; Select Bank 0
CLRF    PORTB  ; Initialize PORTB by clearing
               ;  output data latches
MOVLW   0xCF   ; Value used to initialize
               ;  data direction
MOVWF   DDRB   ; Set RB<3:0> as inputs
               ;   RB<5:4> as outputs
               ;   RB<7:6> as inputs
```

図 10-6: RB3:RB2 ポートピンのブロック図



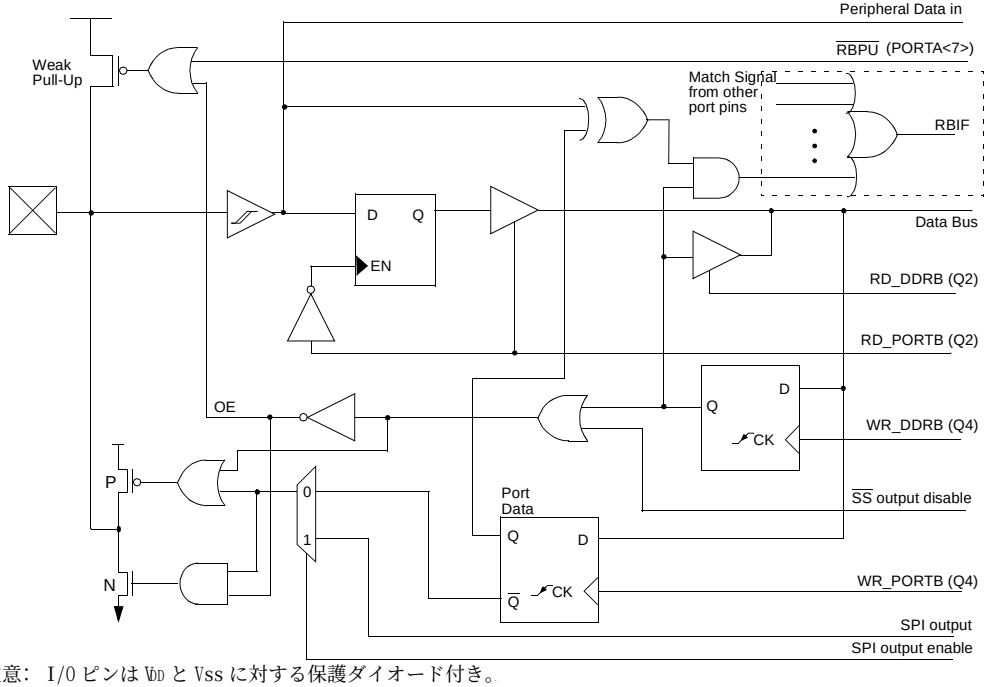
注意:I/O ピンは VDD と Vss に対する保護ダイオード付き。

図 10-7: RB6 ポートピンのブロック図



注意:I/O ピンは VDD と Vss に対する保護ダイオード付き。

図 10-8: RB7 ポートピンのブロック図



注意: I/O ピンは VDD と Vss に対する保護ダイオード付き。

表 10-3: PORTB の機能

名称	Bit	Buffer Type	機能
RB0/CAP1	bit0	ST	入出力またはキャプチャ 1 入力ピン。ソフトウェアでプログラム可能なウィークプルアップ、及びレベル変化割込み機能。
RB1/CAP2	bit1	ST	入出力またはキャプチャ 2 入力ピン。ソフトウェアでプログラム可能なウィークプルアップ、及びレベル変化割込み機能。
RB2/PWM1	bit2	ST	入出力または PWM1 出力ピン。ソフトウェアでプログラム可能なウィークプルアップ、及びレベル変化割込み機能。
RB3/PWM2	bit3	ST	入出力または PWM2 出力ピン。ソフトウェアでプログラム可能なウィークプルアップ、及びレベル変化割込み機能。
RB4/TCLK12	bit4	ST	入出力またはタイマ 1 とタイマ 2 に外部クロック入力。ソフトウェアでプログラム可能なウィークプルアップ、及びレベル変化割込み機能。
RB5/TCLK3	bit5	ST	入出力またはタイマ 3 に外部クロック入力。ソフトウェアでプログラム可能なウィークプルアップ、及びレベル変化割込み機能。
RB6/SCK	bit6	ST	入出力または SPI 用のマスタ / スレーブクロック。ソフトウェアでプログラム可能なウィークプルアップ、及びレベル変化割込み機能。
RB7/SDO	bit7	ST	入出力または SPI 用のデータ出力。ソフトウェアでプログラム可能なウィークプルアップ、及びレベル変化割込み機能。

凡例： ST= シュミットトリガ入力

表 10-4: PORTB に関連するレジスタ / ビット

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR・BOR での値	他のリセットでの値 (1)
12h	PORTB	RB7/ SDO	RB6/ SCK	RB5/ TCLK3	RB4/ TCLK12	RB3/ PWM2	RB2/ PWM1	RB1/ CAP2	RB0/ CAP1	xxxx xxxx	uuuu uuuu
11h, Bank 0	DDRB	PORTB のデータディレクションレジスタ								1111 1111	1111 1111
10h, Bank 0	PORTA	$\overline{\text{RBPU}}$	—	RA5/ TX1/CK1	RA4/ RX1/DT1	RA3/ SDI/SDA	_RA2/ SS/SCL	RA1/TOCKI	RA0/INT	0-xx xxxx	0-uu uuuu
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	--11 1100	--11 qq11
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
16h, Bank 1	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	0000 0010	0000 0010
17h, Bank 1	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	0000 0000	0000 0000
16h, Bank 3	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h, Bank 3	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/$\overline{\text{PR3}}$	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000

凡例： x= 未定、u= 不変、－= 未使用、'0' としてリード。q= 条件によって決まる値。網掛けの部分は PORTB では使われません。
注 1: 他の (パワーアップ以外) リセットは、$\overline{\text{MCLR}}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

10.3 PORTC と DDRC レジスタ

PORTC は 8 ビットの双方向性のポートです。対応するデータディレクションレジスタは DDRC です。DDRC の‘1’が入力として対応するポートピンを設定します。DDRC レジスタの‘0’が出力として対応するポートピンを設定します。PORTC を読み込むと、そのピンのステータスを読み出し、それに書き込むとポートラッチに書き入れます。

PORTC はシステムバスと多重化されています。システムバスとして動作する時､PORTC はアドレス / データバス (AD7:AD0) の下位バイトです。システムバスのタイミングは、電気的特性の章で示します。

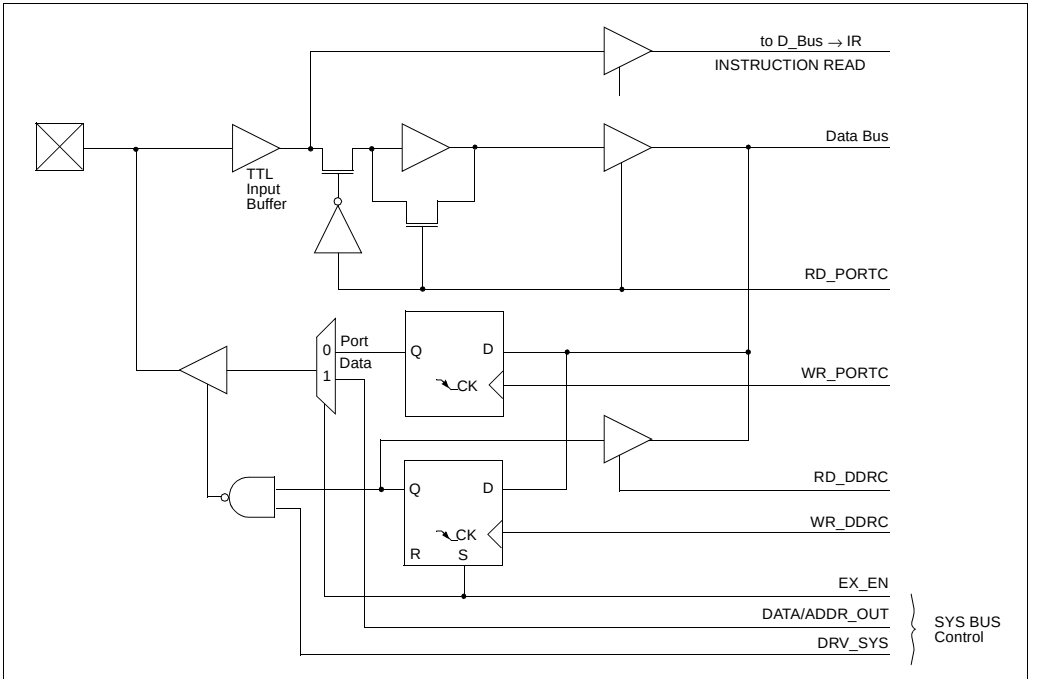
注意： デバイスのコンフィギュレーションビットが、マイクロプロセッサモードまたは拡張マイクロコントローラモードに選択されている時は、このポートはシステムバスとして設定されます。他の 2 つのマイクロコントローラモードでは、このポートは汎用 I/O です。

例 10-3 に、PORTC を初期化するための命令のシーケンスを示します。そのポートを初期化するためには、バンクセレクトレジスタ (BSR) をバンク 1 に選択する必要があります。次の例は、MOVLB 命令を使用して BSR レジスタにデータをロードするものです。

例 10-3: PORTC の初期化

```
MOVLB  1       ; Select Bank 1
CLRF   PORTC   ; Initialize PORTC data
               ; latches before setting
               ; the data direction register
MOVLW   0xCF   ; Value used to initialize
               ; data direction
MOVWF   DDRC   ; Set RC<3:0> as inputs
               ; RC<5:4> as outputs
               ; RC<7:6> as inputs
```

図 10-9: RC7:RC0 ポートピンのブロック図



注意:I/O ピンは VDD と Vss に対する保護ダイオード付き。

表 10-5: PORTC の機能

名称	Bit	Buffer Type	機能
RC0/AD0	bit0	TTL	入出力またはシステムバスのアドレス / データピン
RC1/AD1	bit1	TTL	入出力またはシステムバスのアドレス / データピン
RC2/AD2	bit2	TTL	入出力またはシステムバスのアドレス / データピン
RC3/AD3	bit3	TTL	入出力またはシステムバスのアドレス / データピン
RC4/AD4	bit4	TTL	入出力またはシステムバスのアドレス / データピン
RC5/AD5	bit5	TTL	入出力またはシステムバスのアドレス / データピン
RC6/AD6	bit6	TTL	入出力またはシステムバスのアドレス / データピン
RC7/AD7	bit7	TTL	入出力またはシステムバスのアドレス / データピン

凡例： TTL = TTL 入力

表 10-6: PORTC に関連するレジスタ / ビット

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR・BOR での値	他のリセットでの値 (1)
11h, Bank 1	PORTC	RC7/AD7	RC6/AD6	RC5/AD5	RC4/AD4	RC3/AD3	RC2/AD2	RC1/AD1	RC0/AD0	xxxx xxxx	uuuu uuuu
10h, Bank 1	DDRC	PORTC のデータディレクションレジスタ								1111 1111	1111 1111

凡例： x= 不定、u= 不変。

注 1: 他の (パワーアップ以外) リセットは、$\overline{\text{MCLR}}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

10.4 PORTD と DDRD レジスタ

PORTD は 8 ビットの双方向性のポートです。対応するデータディレクションレジスタは DDRD です。DDRD の‘1’が入力として対応するポートピンを設定します。DDRD レジスタの‘0’が出力として対応するポートピンを設定します。PORTD を読み込むと、そのピンのステータスを読み出し、それに書き込むとポートラッチに書き入れます。

PORTD はシステムバスで多重化されています。システムバスとして動作する時、PORTD はアドレス / データバス (AD15:AD8) の上位バイトです。システムバスのタイミングは、電気的特性の章で示します。

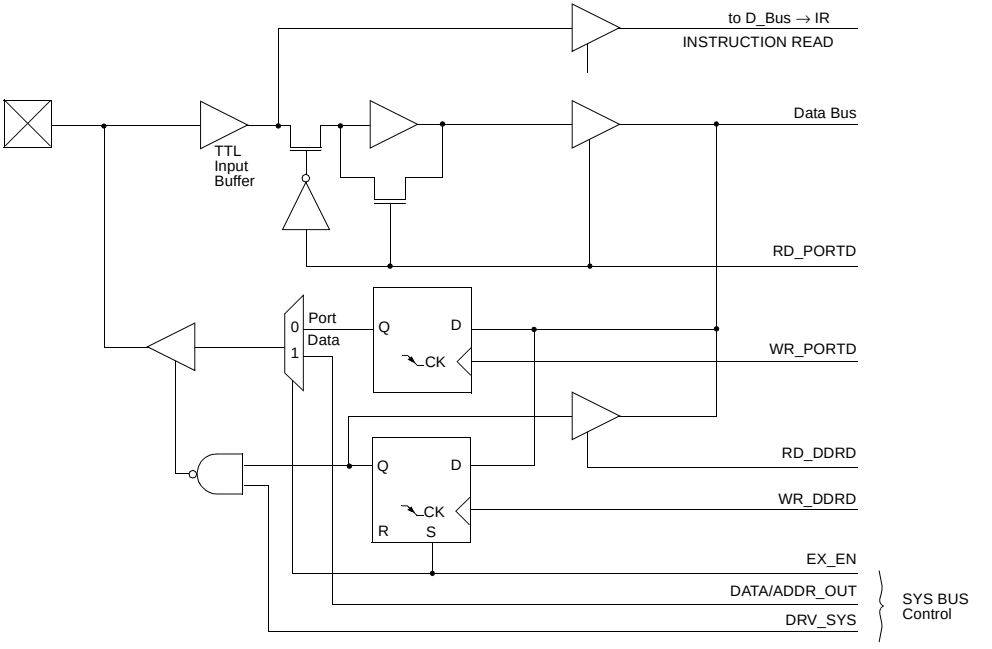
注意：デバイスのコンフィギュレーションビットが、マイクロプロセッサモードまたは拡張マイクロコントローラモードに選択されている時は、このポートはシステムバスとして設定されます。他の 2 つのマイクロコントローラモードでは、このポートは汎用 I/O です。

例 10-4 に、PORTD を初期化するための命令のシーケンスを示します。そのポートを初期化するためには、バンクセレクトレジスタ (BSR) をバンク 1 に選択する必要があります。次の例は、MOVLB 命令を使用して BSR レジスタにデータをロードするものです。

例 10-4: PORTD の初期化

```
MOVLB   1       ; Select Bank 1
CLRF    PORTD   ; Initialize PORTD data
                ; latches before setting
               ; the data direction register
MOVLW   0xCF    ; Value used to initialize
                ; data direction
MOVWF   DDRD    ; Set RD<3:0> as inputs
                ; RD<5:4> as outputs
                ; RD<7:6> as inputs
```

図 10-10: RD7:RD0 ポートピンのブロック図 (I/O ポートモード)



注意：I/O ピンは VDD と Vss に対する保護ダイオード付き。

表 10-7: PORTD の機能

名称	Bit	Buffer Type	機能
RD0/AD8	bit0	TTL	入出力またはシステムバスのアドレス / データピン
RD1/AD9	bit1	TTL	入出力またはシステムバスのアドレス / データピン
RD2/AD10	bit2	TTL	入出力またはシステムバスのアドレス / データピン
RD3/AD11	bit3	TTL	入出力またはシステムバスのアドレス / データピン
RD4/AD12	bit4	TTL	入出力またはシステムバスのアドレス / データピン
RD5/AD13	bit5	TTL	入出力またはシステムバスのアドレス / データピン
RD6/AD14	bit6	TTL	入出力またはシステムバスのアドレス / データピン
RD7/AD15	bit7	TTL	入出力またはシステムバスのアドレス / データピン

凡例： TTL = TTL 入力

表 10-8: PORTD に関連するレジスタ / ビット

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR・BOR での値	他のリセットでの値 (1)
13h, Bank 1	PORTD	RD7/ AD15	RD6/ AD14	RD5/ AD13	RD4/ AD12	RD3/ AD11	RD2/ AD10	RD1/ AD9	RD0/ AD8	xxxx xxxx	uuuu uuuu
12h, Bank 1	DDRD	PORTD のデータディレクションレジスタ								1111 1111	1111 1111

凡例： x= 不定、u= 不変。

注 1: 他の (パワーアップ以外) リセットは、$\overline{\text{MCLR}}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

10.5 PORTE と DDRE レジスタ

PORTE は 4 ビットの双方向性のポートです。対応するデータディレクションレジスタは DDRE です。DDRE の‘1’が入力として対応するポートピンを設定します。DDRE レジスタの‘0’が出力として対応するポートピンを設定します。PORTE を読み込むと、そのピンのステータスを読み出し、それに書き込むとポートラッチに書き入れます。

PORTE はシステムバスと多重化されています。システムバスとして動作する時、PORTE はアドレス / データバス (AD15:AD0) 用の制御シグナルを含んでいます。これらの制御シグナルは、アドレスラッチイネーブル (ALE)、アウトプットイネーブル ($\overline{OE}$)、ライト ($\overline{WR}$) です。制御シグナル $\overline{OE}$ と $\overline{WR}$ はアクティブ LOW のシグナルです。システムバスのタイミングは、電気的特性の章で示します。

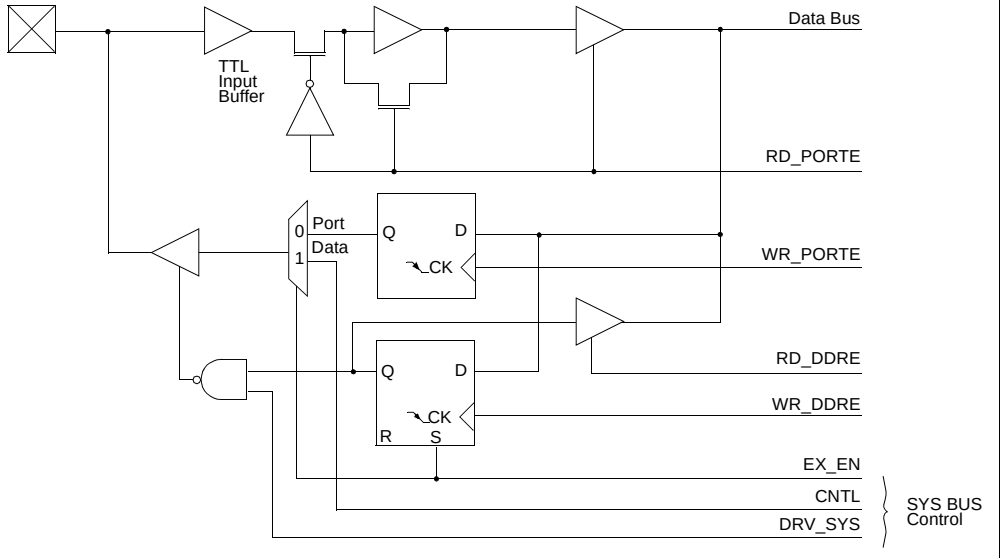
注意：デバイスのコンフィギュレーションビットが、マイクロプロセッサモードまたは拡張マイクロコントローラモードに選択されている時は、このポートのこれらのピンはシステムバスとして設定されます。他のピンは汎用 I/O またはキャプチャ 4 ピンです。他の 2 つのマイクロコントローラモードでは、RE2:RE0 が汎用 I/O ピンです。

例 10-5 に、PORTE を初期化するための命令のシーケンスを示します。そのポートを初期化するためには、バンクセレクトレジスタ (BSR) をバンク 1 に選択する必要があります。次の例は、MOVLB 命令を使用して BSR レジスタにロードするものです。

例 10-5: PORTE の初期化

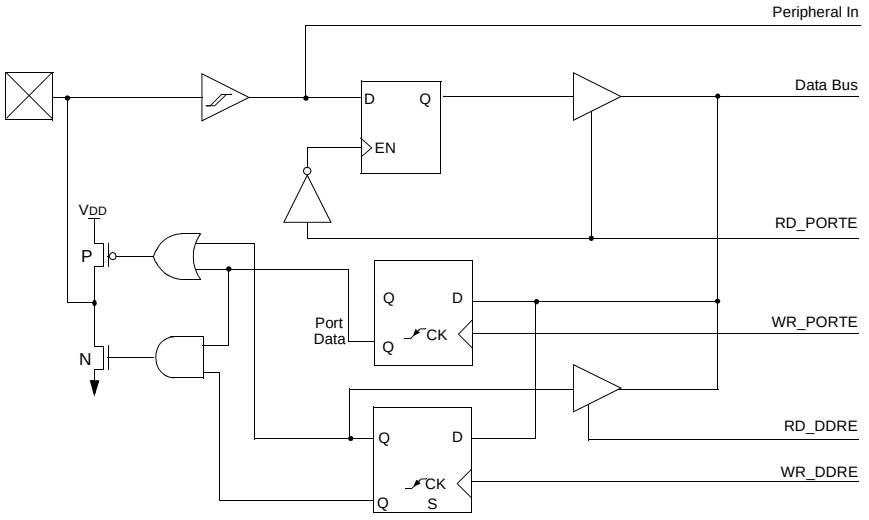
```
MOVLB  1       ; Select Bank 1
CLRF   PORTE   ; Initialize PORTE data
               ; latches before setting
               ; the data direction
               ; register
MOVLW   0x03   ; Value used to initialize
               ; data direction
MOVWF   DDRE   ; Set RE<1:0> as inputs
               ; RE<3:2> as outputs
               ; RE<7:4> are always
               ; read as '0'
```

図 10-11: RE2:RE0 のブロック図 (I/O ポートモード)



注意：I/O ピンは VDD と Vss に対する保護ダイオード付き。

図 10-12: RE3/CAP4 ポートピンのブロック図



注意:I/O ピンは VDD と Vss に対する保護ダイオード付き。

表 10-9: PORTE の機能

名称	Bit	Buffer Type	機能
RE0/ALE	bit0	TTL	入出力またはシステムバスのアドレスラッチイネーブル (ALE) 制御ピン
RE1/$\overline{OE}$	bit1	TTL	入出力またはシステムバスの出力イネーブル ($\overline{OE}$) 制御ピン
RE2/$\overline{WR}$	bit2	TTL	入出力またはシステムバスのライト ($\overline{WR}$) 制御ピン
RE3/CAP4	bit3	ST	入出力またはキャプチャ 4 入力ピン

凡例： TTL=TTL 入力、ST= シュミットトリガ入力

表 10-10: PORTE に関連するレジスタ / ビット

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR・BOR での値	他のリセットでの値 (1)
15h, Bank 1	PORTE	—	—	—	—	RE3/CAP4	RE2/$\overline{WR}$	RE1/$\overline{OE}$	RE0/ALE	---- xxxx	---- uuuu
14h, Bank 1	DDRE	PORTE データディレクションレジスタ								---- 1111	---- 1111
14h, Bank 7	CA4L	Capture4 の下位バイト								xxxx xxxx	uuuu uuuu
15h, Bank 7	CA4H	Capture4 の上位バイト								xxxx xxxx	uuuu uuuu
16h, Bank 7	TCON3	—	CA4OVF	CA3OVF	CA4ED1	CA4ED0	CA3ED1	CA3ED0	PWM3ON	-000 0000	-000 0000

凡例： = 不定、u= 不変、－ = 未使用、‘0’としてリード。網掛けの部分は PORTE では使われません。
注 1: 他の (パワーアップ以外) リセットは、$\overline{MCLR}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

10.6 PORTF と DDRF レジスタ

PORTF は 8bit 幅の双方向性のポートです。対応するデータディレクションレジスタは DDRF です。DDRF の‘1’が入力として対応するポートピンを設定します。DDRF レジスタの‘0’が出力として対応するポートピンを設定します。PORTF を読み込むと、そのピンのステータスを読み出し、それらに書き込むとそれぞれのポートラッチに書き入れます。

8bit の PORTF はすべて、10 ビット の A/D コンバータの 12 チャンネルの内の 8 チャンネルと多重化されています。

リセットによってポート全体が自動的にアナログ入力として設定されいますので、ディジタル I/O に設定するためにはソフトウェアで行います。

例 10-6 に、PORTF を初期化するための命令のシーケンスを示します。そのポートを初期化するためには、バンクセレクトレジスタ (BSR) をバンク 5 に選択する必要があります。次の例は、MOVLB 命令を使用して BSR レジスタにデータをロードするものです。

例 10-6: PORTF の初期化

```
MOVLB  5       ; Select Bank 5
MOVLW  0x0E    ; Configure PORTF as
MOVPF  ADCON1  ; Digital
CLRF   PORTF   ; Initialize PORTF data
               ; latches before setting
               ; the data direction
               ; register
MOVLW  0x03    ; Value used to initialize
               ; data direction
MOVWF  DDRF    ; Set RF<1:0> as inputs
               ; RF<7:2> as outputs
```

図 10-13: RF7:RF0 のブロック図

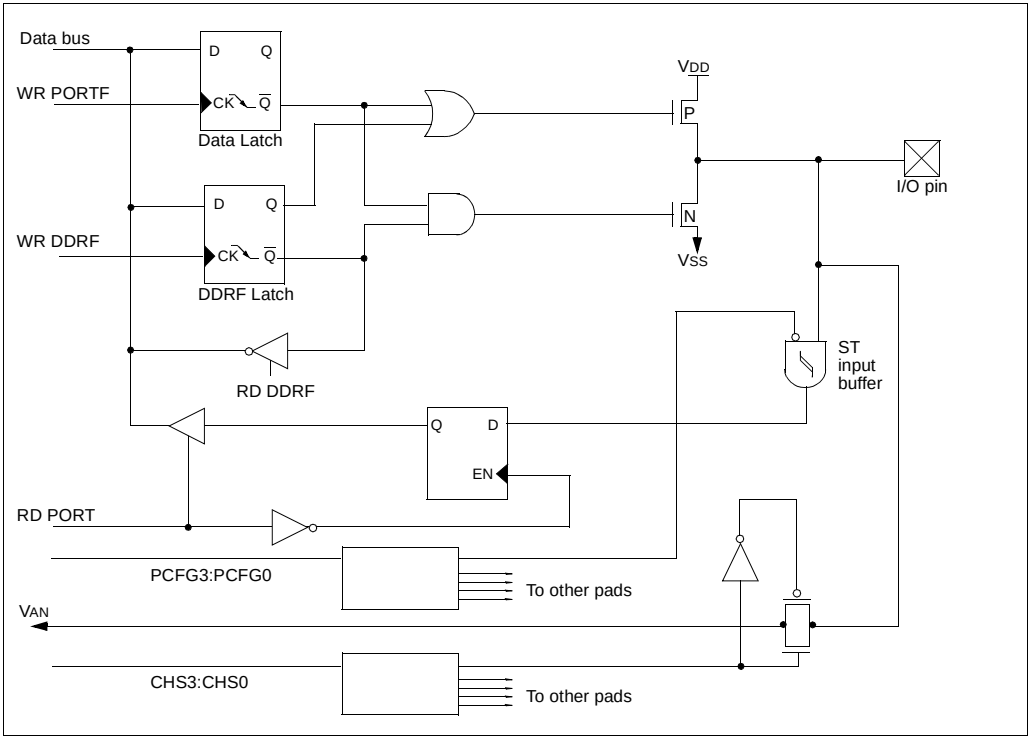


表 10-11: PORTF の機能

名称	Bit	Buffer Type	機能
RF0/AN4	bit0	ST	入出力またはアナログ入力 4
RF1/AN5	bit1	ST	入出力またはアナログ入力 5
RF2/AN6	bit2	ST	入出力またはアナログ入力 6
RF3/AN7	bit3	ST	入出力またはアナログ入力 7
RF4/AN8	bit4	ST	入出力またはアナログ入力 8
RF5/AN9	bit5	ST	入出力またはアナログ入力 9
RF6/AN10	bit6	ST	入出力またはアナログ入力 10
RF7/AN11	bit7	ST	入出力またはアナログ入力 11

凡例： ST= シュミットトリガ入力

表 10-12: PORTF に関連するレジスタ / ビット

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR・BOR での値	他のリセットでの値 (1)
10h, Bank 5	DDRF	PORTF のデータディレクションレジスタ								1111 1111	1111 1111
11h, Bank 5	PORTF	RF7/AN11	RF6/AN10	RF5/AN9	RF4/AN8	RF3/AN7	RF2/AN6	RF1/AN5	RF0/AN4	0000 0000	0000 0000

凡例： x= 不定、u= 不変、－ = 未使用、‘0’としてリード。網掛けの部分は PORTF では使われません。

注 1: 他の（パワーアップ以外）リセットは、$\overline{\text{MCLR}}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

10.7 PORTG と DDRG レジスタ

PORTG は 8 ビット幅の双方向性のポートです。対応するデータ方向レジスタは DDRG です。DDRG の ' 1 ' が入力として対応するポートピンを設定します。DDRG レジスタの ' 0 ' が出力として対応するポートピンを設定します。PORTG を読み込むと、そのピンのステータスを読み出し、それらに書き込むとそれぞれのポートラッチに書き入れます。

PORTG の下位の 4 ビットは、10 ビットの A/D コンバータの 12 チャンネルの内の 4 チャンネルと多重化されています。

PORTG の残りのビットは周辺出力と入力で多重化されています。つまり RG4 は CAP3 入力で、RG5 は PWM 3 出力と、RG6 と RG7 は USART2 機能と多重化されています。

リセットによってはポート全体が自動的にアナログ入力として設定されますので、ディジタル I/O に設定するにはソフトウェアで行う必要があります。

例 10-7 に、PORTG を初期化するための命令のシーケンスを示します。そのポートを初期化するためには、バンクセレクトレジスタ (BSR) をバンク 5 に選択する必要があります。次の例は、MOVLB 命令を使用して BSR レジスタにデータをロードするものです。

例 10-7: PORTG の初期化

```
MOVLB  5       ; Select Bank 5
MOVLW  0x0E    ; Configure PORTG as
MOVPF  ADCON1  ; digital
CLRF   PORTG   ; Initialize PORTG data
               ; latches before setting
               ; the data direction
               ; register
MOVLW  0x03    ; Value used to initialize
               ; data direction
MOVWF  DDRG    ; Set RG<1:0> as inputs
               ; RG<7:2> as outputs
```

図 10-14: RG3:RG0 のブロック図

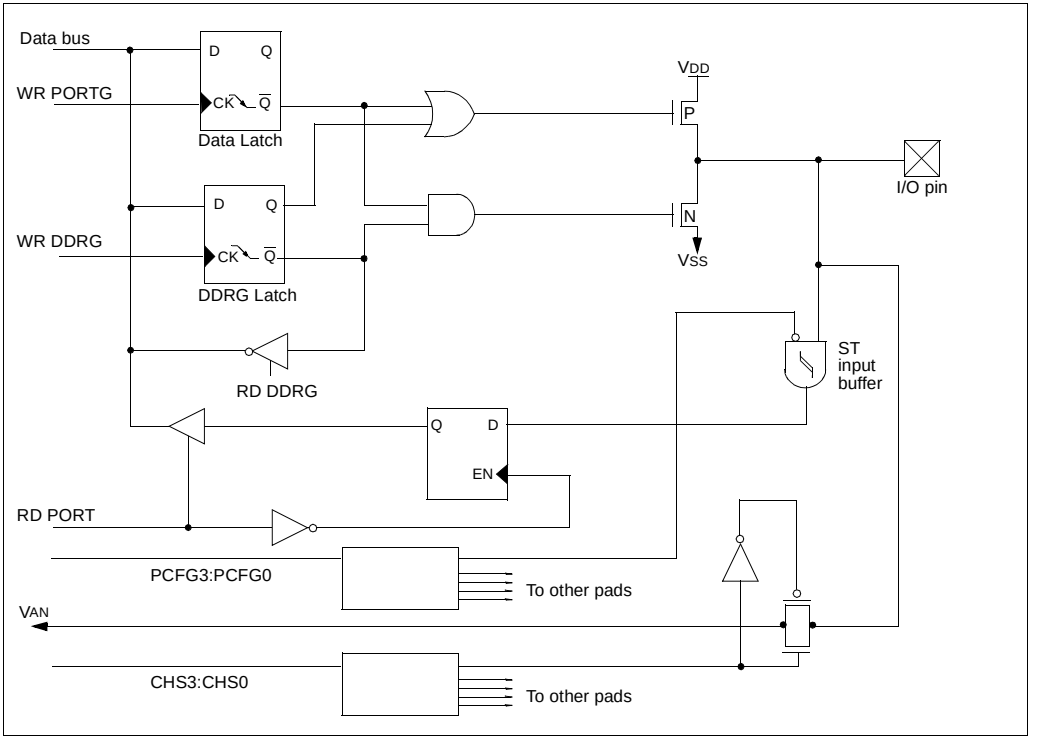
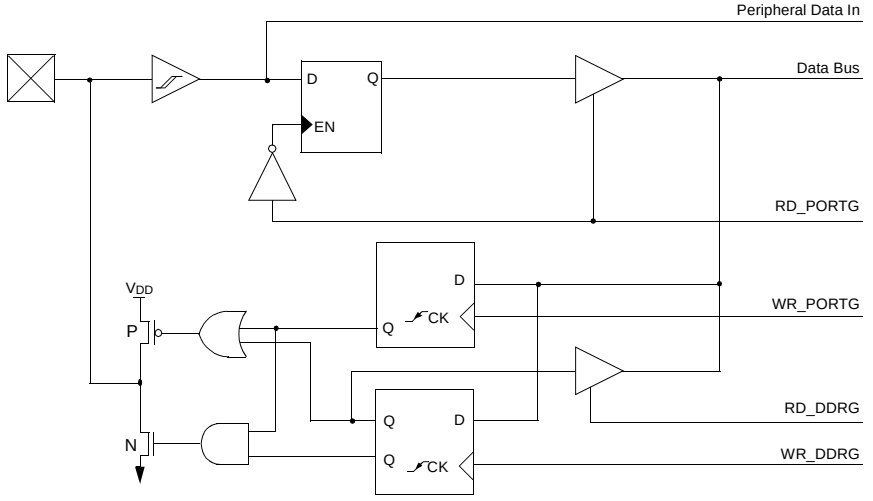
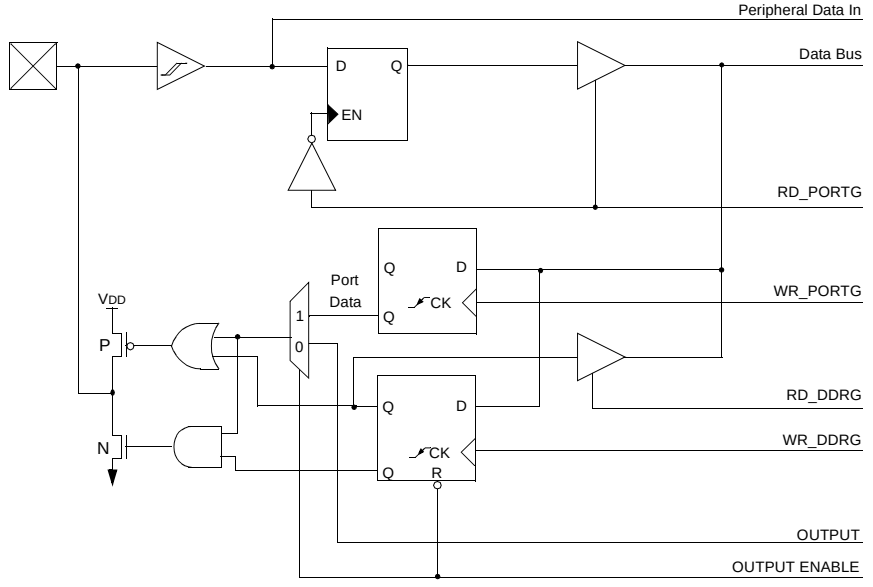


図 10-15: RG4 のブロック図



注意:I/O ピンは VDD と Vss に対する保護ダイオード付き。

図 10-16: RG7:RG5 のブロック図



注意: I/O ピンは VDD と Vss に対する保護ダイオード付き。

表 10-13: PORTG の機能

名称	Bit	Buffer Type	機能
RG0/AN3	bit0	ST	入出力またはアナログ入力 3
RG1/AN2	bit1	ST	入出力またはアナログ入力 2
RG2/AN1/VREF-	bit2	ST	入出力またはアナログ入力 1 または接地基準電圧
RG3/AN0/VREF+	bit3	ST	入出力またはアナログ入力 0 または正極基準電圧
RG4/CAP3	bit4	ST	RG4 はキャプチャ 3 入力ピンでも可能
RG5/PWM3	bit5	ST	RG5 は PWM3 出力ピンでも可能
RG6/RX2/DT2	bit6	ST	RG6 は USART2(SCI) 非同期受信または USART2(SCI) 同期データとしても選択可能
RG7/TX2/CK2	bit7	ST	RG7 は USART2(SCI) 非同期送信または USART2(SCI) 同期クロックとしても選択可能

凡例： ST= シュミットトリガ入力

表 10-14: PORTG に関連するレジスタ / ビット

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR・BOR での値	他のリセットでの値 (1)
12h, Bank 5	DDRG	PORTG のデータディレクションレジスタ								1111 1111	1111 1111
13h, Bank 5	PORTG	RG7/ TX2/CK2	RG6/ RX2/DT2	RG5/ PWM3	RG4/ CAP3	RG3/ AN0	RG2/ AN1	RG1/ AN2	RG0/ AN3	xxxx 0000	uuuu 0000

凡例： x= 不定、u= 不変、－ = 未使用、‘0’としてリード。網掛けの部分は PORTG では使われません。
注 1: 他の (パワーアップ以外) リセットは、$\overline{MCLR}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

10.8 I/O プログラミングの注意点

10.8.1 双方向 I/O ポート

どのライト命令も、内部でライト動作が後に続くリードのような動作をします。例えば、BCF と BSF 命令はそのレジスタを CPU に読み込み、ビット動作を実行し、その結果をレジスタに書き戻します。これらの命令は入力と出力が混在しているポートに使用する時は注意が必要です。例えば、PORTB のビット 5 への BSF 動作により、PORTB の 8 ビットすべてが CPU に読み込まれます。そして 5 ビット目をセットし、修正された 8 ビットのデータが PORTB の出力ラッチに書き込まれます。もし PORTB の 0 ビット目が双方向の入出力ピンとして使われ、0 ビット目が入力として定義されている場合、このピンに与えられている入力信号が CPU に読み込まれ、データラッチに再び書き込まれ、前の内容はオーバーライトされてしまいます。そのピンが入力モードを保持している限り、問題は起こりませんが、ビット0が出力モードに切り変わった場合、データラッチの内容が不明になり、意図しないデータが出力されることがあります。

ポートのリードはポートピンの値を読み込むことです。ポートレジスタへのライトはポートラッチに値を書き込むことです。あるポートでリード・モディファイ・ライト命令 (例えば BCF、BSF、BTG など) を使用する場合、そのポートピンの値が読み込まれ、この値に対して求められる操作が行われ、その結果がポートラッチに書き込まれます。

例 10-8 に、I/O ポートでの 2 つのシーケンシャルなリード・モディファイ・ライト命令の影響を示します。

例 10-8: I/O ポートでのリード・モディファイ・ライト命令

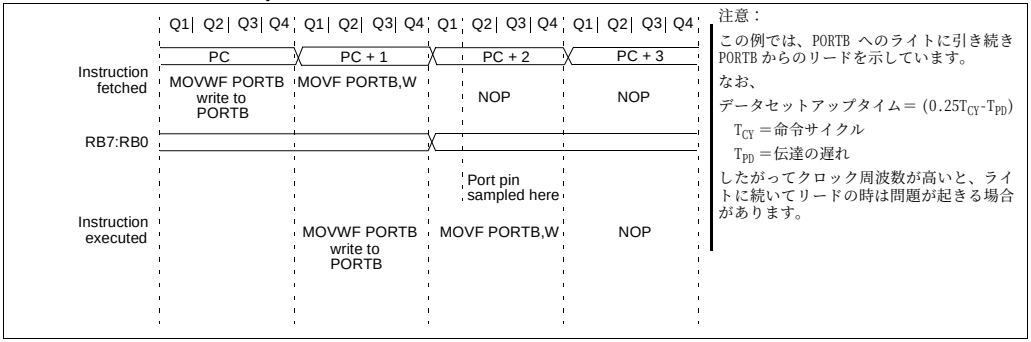
```
; Initial PORT settings: PORTB<7:4> Inputs
;                        PORTB<3:0> Outputs
; PORTB<7:6> have pull-ups and are
; not connected to other circuitry
;
;                     PORT latch  PORT pins
;                     ----------  ---------
;
   BCF   PORTB, 7    ; 01pp pppp   11pp pppp
   BCF   PORTB, 6    ; 10pp pppp   11pp pppp

   BCF   DDRB, 7     ; 10pp pppp   11pp pppp
   BCF   DDRB, 6     ; 10pp pppp   10pp pppp
;
; Note that the user may have expected the
; pin values to be 00pp pppp. The 2nd BCF
; caused RB7 to be latched as the pin value
; (High).
```

注意： ローまたはハイを出力しているピンに対し、外部デバイスから駆動 (ワイヤード or、ワイヤード and) することは、このピンのレベルが変化してしまうため避けてください。結果として、過大電流がデバイスにダメージを与えることがあります。

10.8.2 I/O ポートの連続動作

I/O ポートへの実際のライトは命令サイクルの最後に起こりますが、リードに対しては、データは命令サイクルの始めに有効になっている必要があります (図 10-17 参照)。そのためにリード動作が後に続くライトが同じ I/O ポートで実行された場合には注意が必要です。その I/O ポートの値を読み込む命令を実行する前に、ピンの電圧 (負荷による) が安定できるような命令のシーケンスにする必要があります。そうでないと、そのピンの " 新しい " 状態ではなく前の状態が CPU に読み込まれてしまうことがあります。不確かな時はNOP またはこの I/O ポートをアクセスしない他の命令を使用し、ポートをアクセスする命令が連続しないようにしてください。

図 10-17: 連続的な I/O 動作



NOTES:

11.0 タイマリソースの概要

PIC17C75X には 4 個のタイマモジュールがあります。各モジュールはイベントが起こったことを示すために割込みを発生できます。これらのタイマは下記のように呼ばれています。

- **タイマ 0 ー プログラム可能な 8bit のプリスケーラを伴う 16bit のタイマ**
- **タイマ 1 ー 8bit タイマ**
- **タイマ 2 ー 8bit タイマ**
- **タイマ 3 ー 16bit タイマ**

拡張タイムベース機能としては、4 つの入力キャプチャと 3 つのパルス幅モジュレーション (PWM) 出力があります。PWM はタイマ 1 とタイマ 2 のリソースを使用し、入力キャプチャはタイマ 3 のリソースを使用します。

11.1 タイマ 0 の概要

タイマ 0 モジュールは、単純な 16bit のオーバーフローカウンタです。クロックソースは内部システムクロック(Fosc/4)または外部クロックのどちらでも可能です。

タイマ 0 が外部クロックソースを使用する時、インクリメントするのに、エッジの立ち上がりか立ち下がりを選択できるようになっています。

タイマ 0 モジュールにはプログラム可能なプリスケーラがあります。PS3:PS0 ビット (T0STA<4:1>) がプリスケール値を決定します。TMR0 は次のような比でインクリメントできます。1：1、1：2、1：4、1：8、1：16、1：32、1：64、1：128、1：256。

外部クロックの同期は、プリスケーラの後で起こります。プリスケーラが使われる時、外部クロック周波数はデバイスの周波数より高いことがあります。T0CKI ピンでの最大外部周波数は、クロックに要求されるハイとローの幅から、50MHz です。

11.2 タイマ 1 の概要

タイマ 1 モジュールは、8bit の周期レジスタ (PR1) のついた 8bit のタイマ / カウンタです。TMR1 の値が周期とマッチする値から 0h にロールオーバーすると、TMR1IF フラグがセットされ、イネーブルなら割込みが発生します。カウンタモードでは、そのクロックは RB4/TCLK12 ピンからの入力で、タイマ 2 モジュール用のクロックとしても選択可能です。

TMR1 は 16bit タイマを作るために TMR2 と結合することもできます。TMR1 レジスタは LSB で、TMR2 は MSB です。16bit のタイマモード時は、対応する 16bit の周期レジスタ (PR2:PR1) があります。TMR2:TMR1 の値が周期とマッチする値から 0h にロールオーバーすると、TMR1IF フラグがセットされ、イネーブルなら割込みが発生します。

11.3 タイマ 2 の概要

タイマ 2 モジュールは、8bit の周期レジスタ (PR2) のついた 8bit のタイマ / カウンタです。TMR2 の値が周期とマッチする値から 0h にロールオーバーすると、TMR2IF フラグがセットされ、イネーブルなら割込みが発生します。カウンタモードでは、そのクロックは RB4/TCLK12ピンからの入力で、これはタイマ1モジュール用のクロックとして提供することも可能です。

TMR2 は 16bit タイマを作るために TMR1 と結合することもできます。TMR2 レジスタは MSB で、TMR1 は LSB です。16bit のタイマモード時は、対応する 16bit の周期レジスタ (PR2:PR1) があります。TMR2:TMR1 の値が周期とマッチする値から 0h にロールオーバーすると、TMR1IF フラグがセットされ、イネーブルなら割込みが発生します。

11.4 タイマ 3 の概要

タイマ 3 モジュールは、16bit の周期レジスタのついた 16bit のタイマ / カウンタです。TMR3H:TMR3L の値が 0h にロールオーバーすると、TMR3IF ビットがセットされ、イネーブルなら割込みが発生します。カウンタモードでは、そのクロックは RB5/TCLK3 ピンからの入力です。

4 つのキャプチャモードでの動作時には、周期レジスタは (4 つのうちの) 2 番目の 16bit のキャプチャレジスタになります。

11.5 タイマ / カウンタの役割

タイマモジュールは汎用ですが、それらと結合する専用のリソースを持っています。タイマ 1 とタイマ 2 は 3 つのパルス幅モジュレーション (PWM) 出力用のタイムベースで、タイマ 3 は 4 つの入力キャプチャ用のタイムベースです。

NOTES:

12.0 タイマ 0

タイマ 0 モジュールは、16bit のタイマ / カウンタ TMR0 から成っています。上位バイトはレジスタ TMR0H で、下位バイトはレジスタ TMR0L です。ソフトウェアでプログラム可能な 8bit のプリスケーラが、タイマ 0 を有効な 24 ビットオーバーフロータイマにします。クロックソースは、内部命令クロックまたは RA1/T0CKI ピンからの外部クロックのどちらかを、ソフトウェアでプログラム可能です。このモジュールの制御ビットは、レジスタ T0STA 内にあります (図 12-1)。

図 12-1: T0STA レジスタ (アドレス：05h、バンクされていない)

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	U - 0
INTEDG	T0SE	T0CS	T0PS3	T0PS2	T0PS1	T0PS0	—
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
U = 未使用のビット
'0' としてリード
-n = POR リセットでの値

bit 7: **INTEDG**: RA0/INT ピン割込みエッジ選択ビット
このビットはエッジを選択し、そのエッジで割込みを検出します。
1 =RA0/INT ピンの立ち上がりエッジで割込みを発生
0 =RA0/INT ピンの立ち下がりエッジで割込みを発生

bit 6: **T0SE**: タイマ 0 クロック入力エッジ選択ビット
このビットはエッジを選択し、そのエッジ上で TMR0 をインクリメントします。
T0CS=0 の時 (外部クロック)
1 =RA1/T0CKI ピンの立ち上がりエッジで TMR0 をインクリメントし、そして (または)T0CKIF 割込みを発生
0 =RA1/T0CKI ピンの立ち下がりエッジで TMR0 をインクリメントし、そして (または)T0CKIF 割込みを発生
T0CS=1 の時 (内部クロック)
Don't care （無視）

bit 5: **T0CS**: タイマ 0 クロックソース選択ビット
このビットは TMR0 のクロックソースを選択します。
1 =内部命令クロックサイクル (TCY)
0 =T0CKI ピンからの外部クロック入力

bit 4-1: **T0PS3:T0PS0**: タイマ 0 プリスケール選択ビット
これらのビットは TMR0 のプリスケール値を選択します。

T0PS3:T0PS0	プリスケール値
0000	1:1
0001	1:2
0010	1:4
0011	1:8
0100	1:16
0101	1:32
0110	1:64
0111	1:128
1xxx	1:256

bit 0: **Unimplemented**: 未使用：'0' としてリード

12.1 タイマ 0 の動作

T0CS(T0STA<5>) ビットがセットされると、TMR0 は内部クロックでインクリメントします。T0CS がクリアされると、TMR0 は外部クロックでインクリメントします(RA1/T0CKIピン)。外部クロックのエッジはソフトウェアで選択可能です。T0SE(T0STA<6>) ビットがセットされると、タイマは RA1/T0CKI ピンの立ち上がりエッジでインクリメントします。T0SE がクリアされると、タイマは RA1/T0CKI ピンの立ち下がりエッジでインクリメントします。プリスケーラは 1:1 から 1:256 にプリスケール出来るようにプログラムできます。タイマは 0000h から FFFFh までインクリメントし、0000h にロールオーバーします。オーバーフローでは、TMR0 割込みフラグビット (T0IF) がセットされます。TMR0 の割込みは対応する TMR0 割込みイネーブルビット (T0IE) をクリアすることによりマスクが可能です。TMR0 割込みフラグビット (T0IF) は、TMR0 割込みベクタに分岐すると自動的にクリアされます。

12.2 外部クロックによるタイマ 0 の使い方

外部クロック入力がタイマ 0 に使用されると、内部位相クロックで同期されます。図 12-3 に外部クロックの同期を示します。この同期はプリスケーラの後で行われます。プリスケーラの出力 (PSOUT) は、立ち上がりまたは立ち下がりエッジを検出するために命令サイクル毎に 2 度サンプリングされます。外部クロックに必要なタイミングは、電気的仕様の章で詳しく説明します。

12.2.0 外部クロックエッジからの遅延

プリスケーラ出力が内部クロックに同期しているので、外部クロックのエッジが起こった時から TMR0 が実際にインクリメントする時まで少しの遅れがあります。図 12-3 は、この遅延が 3Tosc と 7Tosc の間で起こることを示しています。このように、例えば 2 つのエッジ (例えば周期) 間の間隔を測定すると、± 4Tosc 以内の精度です (± 121ns @ 33MHz)。

図 12-2: タイマ 0 モジュールのブロック図

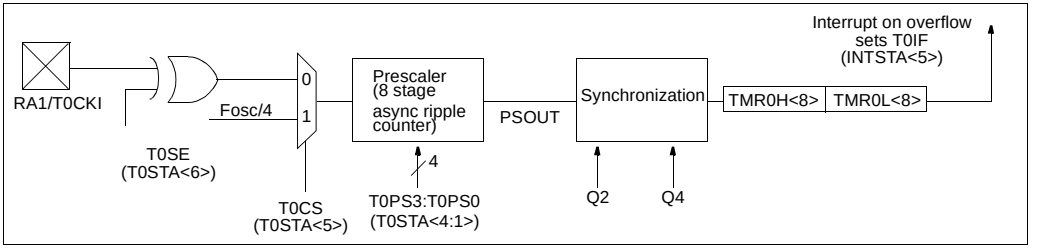
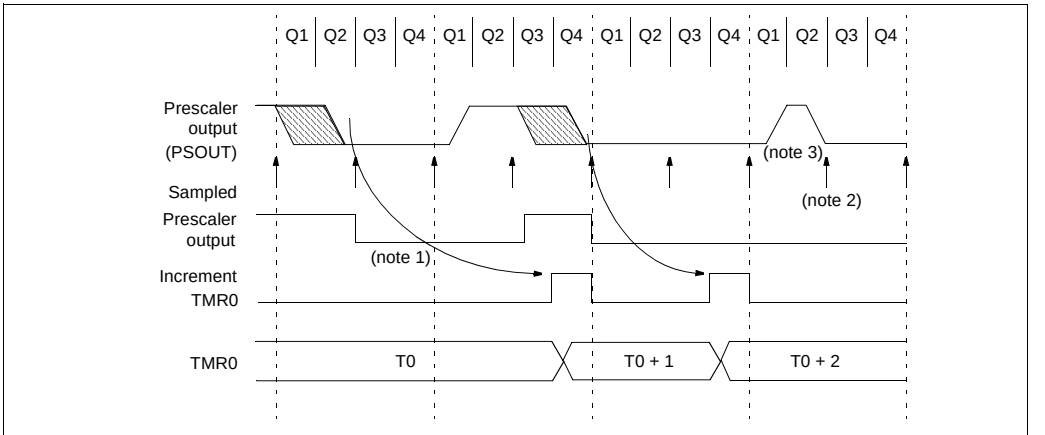


図 12-3: 外部クロックによる TMR0 のタイミング



注 1: T0CKI エッジから TMR0 のインクリメントまでの遅延は、3Tosc から 7Tosc。
2: ↑ =PSOUT はここでサンプルされます。
3: PSOUT のハイタイムは短すぎるので、サンプリング回路で取りそこないます。

12.3 TMR0 のリード / ライトの注意点

TMR0 は 16bit のタイマ / カウンタですが、単一の命令サイクルの間は一度に8bitのみリードまたはライトが可能です。いかなるリード、ライトの間も注意が必要です。

12.3.1 16bit 値のリード

16bit の値全体を読み込むと、下位 (または上位) バイトのリード後、その値が FFh から 00h に変化するという問題が起きることがあります。

例 12-1 に、16bit のリードを示します。適切なリードをするために、このルーチンの間に割込みをディセーブルする必要があります。

例 12-1: 16bit のリード

```
MOVPF   TMR0L, TMPLO    ;read low tmr0
MOVPF   TMR0H, TMPHI    ;read high tmr0
MOVFP   TMPLO, WREG     ;tmplo -> wreg
CPFSLT  TMR0L           ;tmr0l < wreg?
RETURN                  ;no then return
MOVPF   TMR0L, TMPLO    ;read low tmr0
MOVPF   TMR0H, TMPHI    ;read high tmr0
RETURN                  ;return
```

12.3.2 TMR0 への 16bit 値のライト

TMR0L または TMR0H のどちらかにライトすることにより、次のサイクル (続くライト) で TMR0 の半分のインクリメントを禁止し、残り半分のインクリメントを禁止しないので、例 12-2 に示すように、2 つの連続した命令で最初に TMR0L、次に TMR0H にライトする場合、割込みをディセーブルする必要があります。TMR0L、TMR0H どちらへのライトもプリスケーラをクリアします。

例 12-2: 16bit のライト

```
BSF     CPUSTA, GLINTD ; Disable interrupts
MOVFP   RAM_L, TMR0L   ;
MOVFP   RAM_H, TMR0H   ;
BCF     CPUSTA, GLINTD ; Done, enable
                       ;    interrupts
```

12.4 プリスケーラの指定

タイマ 0 には 8bit のプリスケーラがあります。プリスケーラの指定はソフトウェアにより完全に制御されます。すなわちプログラム実行中に変更することができます。プリスケーラの指定を変更する時、その前にプリスケーラのクリアをすることをお勧めします。プリスケーラの値は " 不定 " で、現在の値より少ない値を指定すると、その不定の時間を計算に入れることが難しくなります。

図 12-4: TMR0 のタイミング：上位または下位バイトのライト

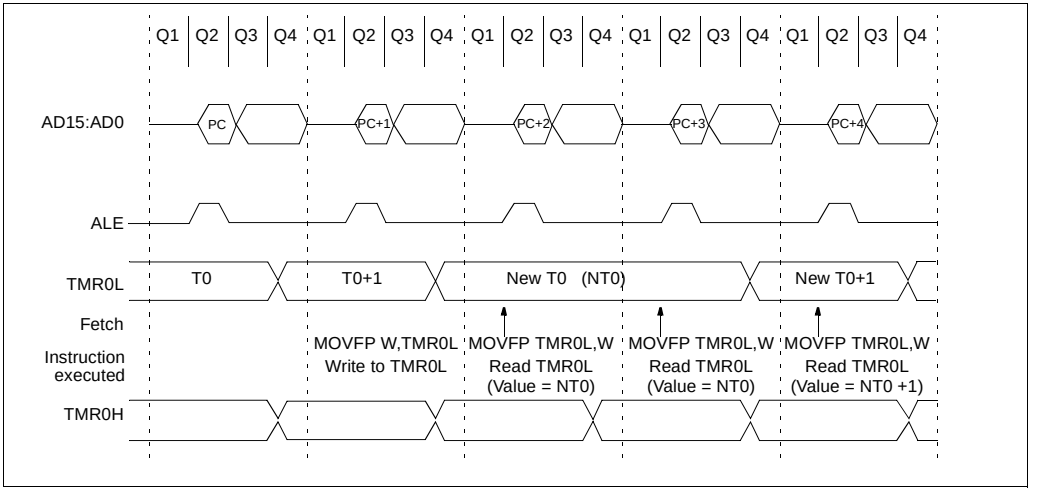
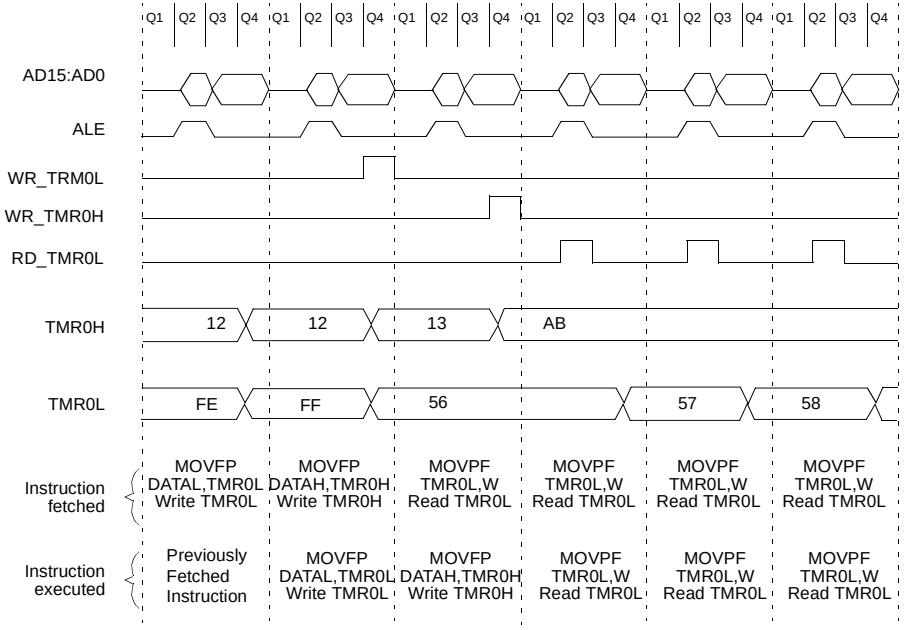


図 12-5: タイマモードでの TMR0 のリード / ライト



この例では、古い TMR0 の値は 12FEh で、AB56h の新しい値を書き込みます。

表 12-1: タイマ 0 に関連するレジスタ / ビット

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR・BOR での値	他のリセットでの値 (注 1)
05h, Unbanked	T0STA	INTEDG	T0SE	T0CS	T0PS3	T0PS2	T0PS1	T0PS0	—	0000 000-	0000 000-
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	--11 1100	--11 qq11
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
0Bh, Unbanked	TMR0L	TMR0 レジスタ；下位バイト								xxxx xxxx	uuuu uuuu
0Ch, Unbanked	TMR0H	TMR0 レジスタ；上位バイト								xxxx xxxx	uuuu uuuu

凡例： x= 未知、u= 不変、－ = 未使用、‘0’としてリード、q= 条件によって決まる値。網掛けの部分はタイマ 0 では使われません。
注 1: 他の（パワーアップ以外）リセットは、$\overline{MCLR}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

13.0 タイマ 1、タイマ 2、タイマ 3、PWM、キャプチャ

PIC17C75X は、制御に関する応用へのインプリメンテーションを容易にするために豊富なタイマとタイムベースの機能を備えています。これらのタイムベース機能は 3 つの PWM 出力と 4 つのキャプチャ入力を含みます。

タイマ 1 とタイマ 2 は、2 つの 8bit のインクリメントするタイマで、各々8bit の周期レジスタ (PR1 と PR2) と別々のオーバーフロー割込みフラグを持っています。タイマ 1 とタイマ 2 はタイマとして (内部 Fosc/4 クロックでインクリメント)、またはカウンタとして (RB4/TCLK12 ピンの外部クロックの立ち下がりエッジでインクリメント) のどちらかで動作することができます。それらは単一の 16bit のタイマ / カウンタとして動作するようソフトウェアの設定も可能です。これらのタイマは PWM(パルス幅モジュレーション) モジュールのタイムベースとしても使用します。

タイマ 3 は 16bit のタイマ / カウンタで、TMR3H と TMR3L レジスタを使用します。タイマ 3 にはさらに 2 つのレジスタ (PR3H/CA1H:PR3L/CA1L) も持ち、16bit の周期レジスタまたは 16bit のキャプチャレジスタとして設定可能です。TMR3 は内部システムクロック (Fosc/4) から、または RB5/TCLK3 ピンの外部シグナルからインクリメントするように、ソフトウェアで設定できます。タイマ 3 は 16bit のキャプチャすべてに対してのタイムベースです。

他の 6 つのレジスタはキャプチャ 2、キャプチャ 3、キャプチャ 4 レジスタ (CA2H:CA2L、CA3H:CA3L、CA4H:CA4L) を形成しています。

図 13-1、13-2、13-3 は、PWM1、PWM2、PWM3、キャプチャ 1、キャプチャ 2、キャプチャ 3、キャプチャ 4 と同様にタイマ 1、タイマ 2、タイマ 3 の動作のための制御レジスタです。

表 13-1 に、これらタイムベースの機能に必要なタイマのリソースを示します。各タイマは、多機能を動かすためにオープンリソースです。

表 13-1: 必要なタイムベース機能 / リソース

Time-base Function	Timer Resource
PWM1	Timer1
PWM2	Timer1 or Timer2
PWM3	Timer1 or Timer2
Capture1	Timer3
Capture2	Timer3
Capture3	Timer3
Capture4	Timer3

図 13-1: TCON1 レジスタ (アドレス :16h、バンク 3)

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
-n = POR リセットでの値

bit 7-6: **CA2ED1:CA2ED0**: キャプチャ 2 モード選択ビット
00 = 立ち下がりエッジ毎にキャプチャ
01 = 立ち上がりエッジ毎にキャプチャ
10 = 4 回毎の立ち上がりエッジでキャプチャ
11 = 16 回毎の立ち上がりエッジでキャプチャ

bit 5-4: **CA1ED1:CA1ED0**: キャプチャ 1 モード選択ビット
00 = 立ち下がりエッジ毎にキャプチャ
01 = 立ち上がりエッジ毎にキャプチャ
10 = 4 回毎の立ち上がりエッジでキャプチャ
11 = 16 回毎の立ち上がりエッジでキャプチャ

bit 3: **T16**: タイマ 2: タイマ 1 モード選択ビット
1 = タイマ 2 とタイマ 1 が 16bit のタイマを作る
0 = タイマ 2 とタイマ 1 は 2 個の 8bit のタイマ

bit 2: **TMR3CS**: タイマ 3 クロックソース選択ビット
1 = TMR3 は RB5/TCLK3 ピンの立ち下がりエッジでインクリメント
0 = TMR3 は内部クロックでインクリメント

bit 1: **TMR2CS**: タイマ 2 クロックソース選択ビット
1 = TMR2 は RB4/TCLK12 ピンの立ち下がりエッジでインクリメント
0 = TMR2 は内部クロックでインクリメント

bit 0: **TMR1CS**: タイマ 1 クロックソース選択ビット
1 = TMR1 は RB4/TCLK12 ピンの立ち下がりエッジでインクリメント
0 = TMR1 は内部クロックでインクリメント

図 13-2: TCON2 レジスタ (アドレス：17h、バンク 3)

R - 0	R - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/PR3	TMR3ON	TMR2ON	TMR1ON
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
-n = POR リセットでの値

bit 7: **CA2OVF**: キャプチャ 2 オーバーフローステータスビット
このビットは、次のキャプチャイベントが起こる前にキャプチャ値がキャプチャレジスタペア (CA2H:CA2L) からリードされていなかったことを示します。キャプチャレジスタは最も古いリードされていないキャプチャ値を保持します (オーバーフローする直前のキャプチャ)。続いて起こるキャプチャイベントは、キャプチャレジスタがリードされるまで (両方のバイト)、TMR3 の値でキャプチャレジスタをアップデートしません。
1 = オーバーフローがキャプチャ 2 レジスタで起こった場合
0 = オーバーフローがキャプチャ 2 レジスタで起こらなかった場合

bit 6: **CA1OVF**: キャプチャ 1 オーバーフローステータスビット
このビットは、次のキャプチャイベントが起こる前にキャプチャ値がキャプチャレジスタペア (PR3H/CA1H:PR3L/CA1L) からリードされていなかったことを示します。キャプチャレジスタは最も古いリードされていないキャプチャ値を保持します (オーバーフローする直前のキャプチャ)。続いて起こるキャプチャイベントは、キャプチャレジスタがリードされるまで (両方のバイト)、TMR3 の値でキャプチャレジスタをアップデートしません。
1 = オーバーフローがキャプチャ 1 レジスタで起こった場合
0 = オーバーフローがキャプチャ 1 レジスタで起こらなかった場合

bit 5: **PWM2ON**: PWM2 オンビット
1 = PWM2 はイネーブル (RB3/PWM2 ピンは DDRB<3> ビットの状態を無視)
0 = PWM2 はディセーブル (RB3/PWM2 ピンはデータ方向指示のため DDRB<3> ビットの状態を使用)

bit 4: **PWM1ON**: PWM1 オンビット
1 = PWM1 はイネーブル (RB2/PWM1 ピンは DDRB<2> ビットの状態を無視)
0 = PWM1 はディセーブル (RB2/PWM1 ピンはデータ方向指示のため DDRB<2> ビットの状態を使用)

bit 3: **CA1/$\overline{\text{PR3}}$**: CA1/$\overline{\text{PR3}}$ レジスタモード選択ビット
1 = キャプチャ 1 をイネーブル (PR3H/CA1H:PR3L/CA1L はキャプチャ 1 レジスタ。タイマ 3 は周期レジスタなしに動く)
0 = 周期レジスタをイネーブル (PR3H/CA1H:PR3L/CA1L はタイマ 3 用の周期レジスタ)

bit 2: **TMR3ON**: Timer3 オンビット
1 = タイマ 3 を始動
0 = タイマ 3 を停止

bit 1: **TMR2ON**: Timer2 オンビット
このビットは TMR2 レジスタのインクリメントを制御します。TMR2:TMR1 が 16bit のタイマを作る時 (T16 をセット)、TMR2ON をセットする必要があります。これによりタイマの MSB がインクリメントするのを可能にします。
1 = タイマ 2 を始動 (T16 ビット (TCON1<3> をセットするならイネーブルが必要)
0 = タイマ 2 を停止

bit 0: **TMR1ON**: Timer1 オンビット
T16 をセット時 (16bit タイマモード)
1 = 16bitTMR2:TMR1 を始動
0 = 16bitTMR2:TMR1 を停止

T16 をクリア時 (8bit タイマモード)
1 = 8bit タイマ 1 を始動
0 = 8bit タイマ 1 を停止

図 13-3: TCON3 レジスタ (アドレス：16h、バンク 7)

U-0	R - 0	R - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
-	CA4OVF	CA3OVF	CA4ED1	CA4ED0	CA3ED1	CA3ED0	PWM3ON
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
U = 未使用のビット、
‘0’としてリード
-n =POR リセットでの値

bit 7: **Unimplemented:** 未使用：‘0’としてリード

bit 6: **CA4OVF**: キャプチャ 4 オーバーフローステータスビット
このビットは､次のキャプチャイベントが起こる前にキャプチャ値がキャプチャレジスタペア(CA4H:CA4L)からリードされていなかったことを示します。キャプチャレジスタは最も古いリードされていないキャプチャ値を保持します (オーバーフローする直前のキャプチャ)。続いて起こるキャプチャイベントは、キャプチャレジスタがリードされるまで (両方のバイト)、TMR3 の値でキャプチャレジスタをアップデートしません。
1 = オーバーフローがキャプチャ 4 レジスタで起こった場合
0 = オーバーフローがキャプチャ 4 レジスタで起こらなかった場合

bit 5: **CA3OVF**: キャプチャ 3 オーバーフローステータスビット
このビットは､次のキャプチャイベントが起こる前にキャプチャ値がキャプチャレジスタペア(CA3H:CA3L)からリードされていなかったことを示します。キャプチャレジスタは最も古いリードされていないキャプチャ値を保持します (オーバーフローする直前のキャプチャ)。続いて起こるキャプチャイベントは、キャプチャレジスタがリードされるまで (両方のバイト)、TMR3 の値でキャプチャレジスタをアップデートしません。
1 = オーバーフローがキャプチャ 3 レジスタで起こった場合
0 = オーバーフローがキャプチャ 3 レジスタで起こらなかった場合

bit 4-3: **CA4ED1:CA4ED0**: キャプチャ 4 モード選択ビット
00 = 立ち下がりエッジ毎にキャプチャ
01 = 立ち上がりエッジ毎にキャプチャ
10 = 4 回毎の立ち上がりエッジでキャプチャ
11 = 16 回毎の立ち上がりエッジでキャプチャ

bit 2-1: **CA3ED1:CA3ED0**: キャプチャ 3 モード選択ビット
00 = 立ち下がりエッジ毎にキャプチャ
01 = 立ち上がりエッジ毎にキャプチャ
10 = 4 回毎の立ち上がりエッジでキャプチャ
11 = 16 回毎の立ち上がりエッジでキャプチャ

bit 0: **PWM3ON**: PWM3 オンビット
1 = PWM3 はイネーブル (RG5/PWM3 ピンは DDRG<5> ビットの状態を無視)
0 = PWM3 はディセーブル (RG5/PWM3 ピンはデータ方向定義に DDRG<5> ビットの状態を使用)

13.1 タイマ 1 とタイマ 2

13.1.1 8bit モードでのタイマ 1 とタイマ 2

タイマ 1 とタイマ 2 は両方とも、T16 ビットがクリアの時は 8bit モードで動作します。これら 2 つのタイマは、内部命令サイクルクロック (TCY)、または RB4/TCLK12 ピンからの外部クロックソースからインクリメントするようそれぞれに設定することができます。タイマクロックソースは TMRxCS ビットにより設定します (タイマ 1 では x=1、タイマ 2 では x=2)。TMRxCS がクリアされると、クロックソースは内部になり、命令サイクル毎に 1 度インクリメントします (Fosc/4)。TMRxCS がセットされると、クロックソースは RB4/ TCLK12 ピンになり、カウンタが RB4/ TCLK12 ピンの立ち下がりエッジ毎にインクリメントします。

タイマは 00h から、周期レジスタ (PRx) と等しくなるまでインクリメントします。さらに次のインクリメントサイクルで00hにリセットします。タイマがリセットされると、タイマ割込みフラグがセットされます。TMR1 と TMR2 は独立した割込みフラグビットを持っています。TMR1割込みフラグビットがTMR1IFにラッチされ、TMR2 割込みフラグビットが TMR2IF にラッチされます。

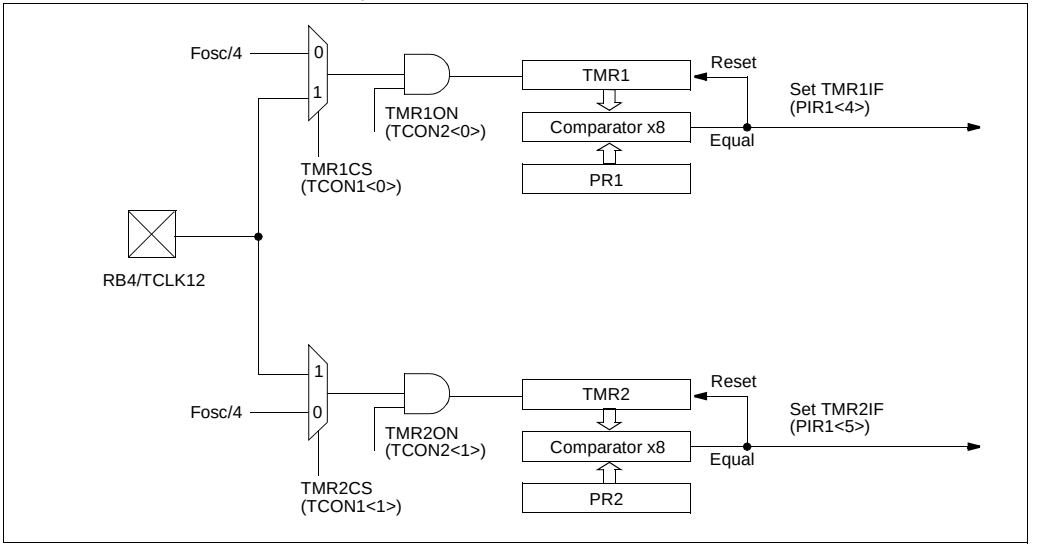
各タイマは対応する割込みイネーブルビット (TMRxIE) も持っています。タイマ割込みは、このビットをセット / クリアすることによりイネーブル / ディセーブルできます。周辺割込みをイネーブルするためには、周辺割込みイネーブルビットをセットし (PEIE= ‘ 1 ’)、全体割込みをイネーブルする (GLINTD= ‘ 0 ’) 必要があります。

タイマは、ソフトウェアの制御でオンやオフができます。制御ビットのタイマ (TMRxON) がセットされると、タイマはクロックソースからインクリメントします。TMRxON がクリアされると、タイマはオフになり、タイマ割込みフラグをセットできないようにします。

13.1.1.1 タイマ 1 とタイマ 2 の外部クロック入力

TMRxCS がセットされると、クロックソースは RB4/TCLK12 ピンになり、カウンタが RB4/TCLK12 ピンの立ち下がりエッジ毎にインクリメントします。TCLK12 入力は内部位相クロックと同期しています。これにより、立ち下がりエッジが TCLK12 に現れた時から、TMR1 か TMR2 が実際にインクリメントする時まで遅れがあります。必要な外部クロック入力のタイミングについては、電気的仕様の章を参照してください。

図 13-4: 2 つの 8bit のタイマ / カウンタモードでのタイマ 1 とタイマ 2



13.1.2 16bit モードでのタイマ 1 とタイマ 2

16bit モードを選択するには、T16 ビットをセットします。このモードでは、TMR2 と TMR1 は、16bit のタイマを作るように連結されます (TMR2:TMR1)。16bit タイマは、16bit 周期レジスタ (PR2:PR1) と一致するまでインクリメントします。続くタイマクロックでは、そのタイマの値は 0h にリセットし、TMR1IF ビットがセットされます。

16bit タイマ用のクロックソースを選択すると、TMR1CS ビットが 16bit のタイマ全体を制御し、TMR2CS は無視されますが、TMR2ON はセットを確実にする必要があります(TMR2をインクリメントさせるため)。TMR1CS をクリアすると、タイマは命令サイクル毎に 1 度インクリメントします (Fosc/4)。TMR1CS がセットされると、タイマは RB4/TCLK12 ピンの立ち下がりエッジ毎にインクリメントします。16bit のタイマをインクリメントするためには、TMR1ON とTMR2ON の両方のビットをセットする必要があります (表 13-2)。

表 13-2: 16bit タイマのターニング

T16	TMR2ON	TMR1ON	Result
1	1	1	16-bit timer (TMR2:TMR1) ON
1	0	1	Only TMR1 increments
1	x	0	16-bit timer OFF
0	1	1	Timers in 8-bit mode

13.1.2.1 TMR2:TMR1 の外部クロック入力

TMR1CS がセットされると、16bit の TMR2:TMR1 はクロック入力 TCLK12 の立ち下がりエッジでインクリメントします。RB4/TCLK12 ピンの入力は、命令サイクル毎に 2 回、内部位相クロックによりサンプリングされ同期されます。これにより、立ち下がりエッジが RB4/TCLK12 に現れた時から、TMR2:TMR1 が実際にインクリメントする時まで遅れがあります。必要な外部クロック入力のタイミングについては、電気的仕様の章を参照してください。

図 13-5: 16bit のタイマ / カウンタモードでの TRM2 と TRM1

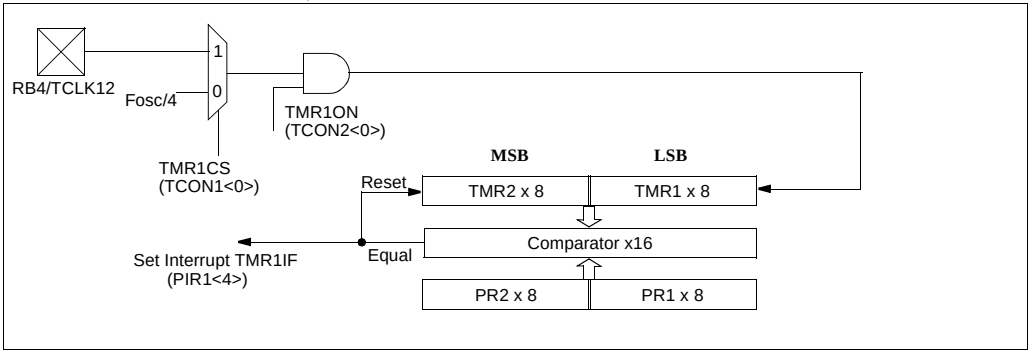


表 13-3: タイマ 1 とタイマ 2 レジスタの概要

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR・BOR での値	他のリセットでの値 (注 1)
16h, Bank 3	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h, Bank 3	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/$\overline{PR3}$	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
16h, Bank 7	TCON3	—	CA4OVF	CA3OVF	CA4ED1	CA4ED0	CA3ED1	CA3ED0	PWM3ON	-000 0000	-000 0000
10h, Bank 2	TMR1	タイマ 1 のレジスタ								xxxx xxxx	uuuu uuuu
11h, Bank 2	TMR2	タイマ 2 のレジスタ								xxxx xxxx	uuuu uuuu
16h, Bank 1	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	0000 0010	0000 0010
17h, Bank 1	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	0000 0000	0000 0000
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	--11 1100	--11 qq11
14h, Bank 2	PR1	タイマ 1 の周期レジスタ								xxxx xxxx	uuuu uuuu
15h, Bank 2	PR2	タイマ 2 の周期レジスタ								xxxx xxxx	uuuu uuuu
10h, Bank 3	PW1DCL	DC1	DC0	—	—	—	—	—	—	xx-- ----	uu-- ----
11h, Bank 3	PW2DCL	DC1	DC0	TM2PW2	—	—	—	—	—	xx0- ----	uu0- ----
10h, Bank 7	PW3DCL	DC1	DC0	TM2PW3	—	—	—	—	—	xx0- ----	uu0- ----
12h, Bank 3	PW1DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
13h, Bank 3	PW2DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
11h, Bank 7	PW3DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu

凡例 : x= 不定、u= 不変、ー = 未使用、‘ 0 ’としてリード、q= 条件によって決まる値。網掛けの部分はタイマ 1 とタイマ 2 では使われません。

注 1: 他の（パワーアップ以外）リセットは、$\overline{MCLR}$ を通る外部リセットと WDT タイマリセットを含みます。

13.1.3 タイマ 1 とタイマ 2 でのパルス幅変調 (PWM) 出力の使い方

3 つの高スピードのパルス幅変調 (PWM) 出力が用意されています。PWM1 出力はそのタイムベースとしてタイマ 1 を使用し、PWM2 と PWM3 は、タイムベースとしてタイマ1タイマ2のどちらかを使用するよう、それぞれにソフトウェアで設定できます。PWM 出力は、RB2/PWM1、RB3/PWM2、RG5/PWM3 ピン上にあります。

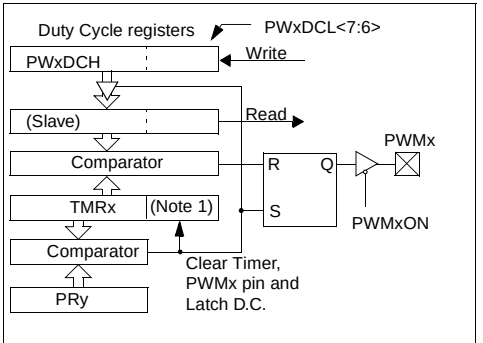
各 PWM 出力は 10bit の最大分解能を持っています。10bit の分解能では、PWM 出力の周波数は 32.2kHz(@ 32MHz クロック) で、8bit の分解能では PWM 出力の周波数は 128.9kHz です。出力のデューティサイクルは 0% から 100% まで変えることができます。

図 13-6 に、簡略化された PWM モジュールのブロック図を示します。

デューティサイクルレジスタは、グリッチ無しの動作のためにダブルバッファされています。図 13-7 に、デューティサイクルレジスタがダブルバッファされない場合、グリッチがどのように起こるかを示します。

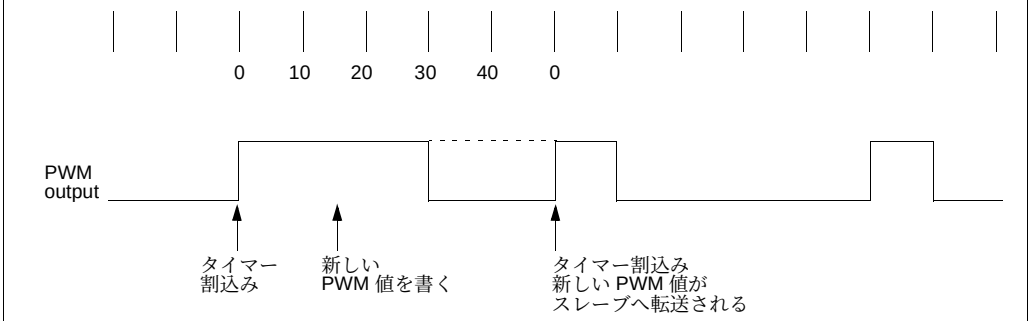
PWM1 出力をイネーブルするためには PWM1ON ビット (TCON2<4>) をセットする必要があります。PWM1ON ビットがセットされると、RB2/PWM1 ピンが PWM1 出力として設定され、データ方向指示ビット (DDRB<2>) に関係しない出力として無理に設定されます。PWM1ON ビットがクリアされると、ピンはポートピンとして動き、その方向はデータ方向指示ビット (DDRB<2>) により制御されます。同様に、PWM2ON (TCON2<5>) ビットが RB3/PWM2 ピンの設定を制御し、PWM3ON(TCON3<0>) ビットが RG5/PWM3 ピンの設定を制御します。

図 13-6: PWM の簡略ブロック図



注 1: 8bit のタイマは 10bit のタイムベースを生成するために 2 bit の内部 Q クロックまたは 2 bit のプリスケーラと結合しています。

図 13-7: PWM 出力



注意: 点線は、デューティサイクルレジスタがダブルバッファされなかった場合の PWM 出力を示します。
タイマが新しいデューティサイクルの値を越えた後にライトした場合、現在のサイクルが“ グリッチ ”を発生させている間は PWM は全くリセットされません。
この例では、PWM 周期 =50 です。古いデューティサイクルは 30 で、新しいデューティサイクルの値は 10 です。

13.1.3.1 PWM 周期

PWM1 出力の周期はタイマ 1 とその周期レジスタ (PR1) により決まります。PWM2 と PWM3 出力の周期は、タイムベースとしてタイマ 1 またはタイマ 2 のどちらを使用するかをソフトウェアで設定します。PWM2 に関して、TM2PW2 ビット (PW2DCL <5>) がクリアされると、タイムベースが TMR1 と PR1 により決定され、TM2PW2 がセットされると、タイムベースはタイマ 2 と PR2 により決定されます。PWM3 に関しては、TM2PW3 ビット (PW3DCL<5>) がクリアされると、タイムベースが TMR1 と PR1 により決定され、TM2PW3 がセットされると、タイムベースはタイマ 2 と PR2 により決定されます。

2 つの異なったタイマで 2 つの異なった PWM 出力を動かすことにより、異なった PWM 周期を可能にします。タイマ 1 からすべての PWM を動かすことにより、タイマ 2 を 8bit タイマとして自由に動かせるので、リソースを最も有効に使用できます。PWM のどれかが使用されている場合、タイマ 1 とタイマ 2 は 16bit のタイマとしては使用できません。

PWM 周期は次のように計算します。

PWM1 の周期 = [(PR1) + 1] x 4TOSC

PWM2 の周期 = [(PR1) + 1] x 4TOSC または
[(PR2) + 1] x 4TOSC

PWM3 の周期 = [(PR1) + 1] x 4TOSC または
[(PR2) + 1] x 4TOSC

PWMx のデューティサイクルは、10bit の値 DCx<9:0> により決まります。上位 8bit はレジスタ PWxDCH からで、下位 2bit は PWxDCL<7:6> からです (PWxDCH:PWxDCL<7:6>)。表 13-4 に、周期レジスタに与えた値の最大 PWM 周波数 (FPWM) を示します。

PWM が達成できる分解能のビット数は、PWM 周波数 (FPWM) と同様にデバイスの動作周波数によります。

与えられた PWM 周波数に対する最大 PWM 分解能 (ビット) は :

$$= \frac{\log\left(\frac{FOSC}{FPWM}\right)}{\log(2)} \text{ bits}$$

ここで FPWM = 1 / PWM の周期

PWMx duty cycle は次の通りです。

PWMx Duty Cycle =(DCx) x TOSC

ただし DCx は、PWxDCH:PWxDCL からの 10bit の値を表わします。

DCx=0 の場合デューティサイクルはゼロで、PRx= PWxDCH なら、PWM 出力は 4 つの Q クロックの 1 つに対して Low になります (PWxDCL<7:6> ビットの状態による)。100% のデューティサイクルにするには、PWxDCH の値は PRx の値より大きくなくてはいけません。

両方の PWM 出力に対するデューティサイクルレジスタは、ダブルバッファされています。これらのレジスタに書き込むと、マスタラッチに保持されます。TMR1(または TMR2) がオーバーフローし新しい PWM 周期が始まると、マスタラッチの値はスレーブラッチに転送され、PWMx ピンは強制的にハイになります。

注意 : PW1DCH、PW1DCL、PW2DCH、PW2DCL、PW3DCH、PW3DCL のレジスタに関しては、ライト動作は " マスタラッチ " に書き込み、リード動作は " スレーブラッチ " に読み込みます。結果として、デューティサイクルレジスタに書き込まれたものをリードバックすることはできません。

また、デューティサイクルレジスタでの ADDWF PW1DCH のような " リード・モディファイ・ライト " 動作はどれも避ける必要があります。これにより予測できないデューティサイクル出力を引き起こすことがあります。

表 13-4: PWM 周波数 対 33MHz での分解能

PWM Frequency	Frequency (kHz)				
	32.2	64.5	90.66	128.9	515.6
PRx Value	0xFF	0x7F	0x5A	0x3F	0x0F
High Resolution	10-bit	9-bit	8.5-bit	8-bit	6-bit
Standard Resolution	8-bit	7-bit	6.5-bit	6-bit	4-bit

13.1.3.2 PWM 割込み

PWM モジュールは、TMR1、TMR2 の割込みを使用します。TMR1 または TMR2 がその周期レジスタと等しくなると、タイマ割込みが発生し、それに続くインクリメントによりゼロにクリアされます。この割込みは、PWM サイクルの始まりもマークします。タイマがロールオーバーする前に新しいデューティサイクルの値を書き込むことができます。TMR1 割込みは TMR1IF ビットにラッチされ、TMR2 割込みは TMR2IF ビットにラッチされます。これらのフラグはソフトウェアでクリアする必要があります。

13.1.3.3 外部クロックソース

PWM はタイマのクロックソースにかかわらず動作します。外部クロックを使うには、理解が必要な派生する効果があります。外部 TCLK12 入力は内部で同期されているので (命令サイクル毎に 1 度サンプリング) タイマがインクリメントする時間の TCLK12 のタイム変化は、1TCY と同様に変わります (1 命令サイクル)。これにより PWM 出力の周期と同じようにデューティサイクルでジッタが起こります。

外部クロックがプロセッサクロックと同期しなければ、ジッタは± 1TCY です。PWM 出力の 1 つを TCLK12 入力のクロックソースとして使用すると、同期クロックを与えます。

一般に、PWM の外部クロックソースを使用する場合、周波数はデバイスの周波数 (Fosc) よりかなり小さくする必要があります。

13.1.3.3.1 外部クロック入力の最大分解能 / 周波数

PWM のタイムベース用 (タイマ 1 またはタイマ 2) に外部クロックを使用すると、PWM 出力を最大 8bit の分解能に制限します。PWxDCL<7:6> ビットをクリアし続ける必要があります。他のどんな値を使っても、PWM 出力は歪みます。すべての分解能は、内部クロックモードを選択するとサポートされます。最大の到達できる周波数も低くなります。これはタイマ用の外部クロック入力に必要なタイミングの結果です (電気的仕様の章を参照)。タイマクロックソースが RB4 /TCLK12 ピンである場合、最大の PWM 周波数は表 13-4 に示す通りです (標準分解能モード)。

表 13-5: PWM に関連するレジスタ / ビット

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR・BOR での値	他のリセットでの値 (注 1)
16h, Bank 3	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h, Bank 3	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/$\overline{PR3}$	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
16h, Bank 7	TCON3	—	CA4OVF	CA3OVF	CA4ED1	CA4ED0	CA3ED1	CA3ED0	PWM3ON	-000 0000	-000 0000
10h, Bank 2	TMR1	タイマ 1 のレジスタ								xxxx xxxx	uuuu uuuu
11h, Bank 2	TMR2	タイマ 2 のレジスタ								xxxx xxxx	uuuu uuuu
16h, Bank 1	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	0000 0010	0000 0010
17h, Bank 1	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	0000 0000	0000 0000
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	--11 1100	--11 qq11
14h, Bank 2	PR1	タイマ 1 の周期レジスタ								xxxx xxxx	uuuu uuuu
15h, Bank 2	PR2	タイマ 2 の周期レジスタ								xxxx xxxx	uuuu uuuu
10h, Bank 3	PW1DCL	DC1	DC0	—	—	—	—	—	—	xx-- ----	uu-- ----
11h, Bank 3	PW2DCL	DC1	DC0	TM2PW2	—	—	—	—	—	xx0- ----	uu0- ----
10h, Bank 7	PW3DCL	DC1	DC0	TM2PW3	—	—	—	—	—	xx0- ----	uu0- ----
12h, Bank 3	PW1DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
13h, Bank 3	PW2DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
11h, Bank 7	PW3DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu

凡例 : x= 不定、u= 不変、－ = 未使用、‘ 0 ’ としてリード、q= 条件によって決まる値。網掛けの部分は PWM モジュールでは使われません。

注 1: 他の（パワーアップ以外）リセットは、$\overline{MCLR}$ を通る外部リセットと WDT タイマリセットを含みます。

13.2 タイマ 3

タイマ 3 は TMR3H と TMR3L レジスタから成る 16bit のタイマです。TMR3H はタイマの上位バイトで、TMR3L は下位バイトです。このタイマは 16bit の周期レジスタが付いています (PR3H/CA1H:PR3L/CA1L)。この周期レジスタは、もう 1 つの 16bit キャプチャレジスタにソフトウェアで設定できます。

TMR3CS ビット (TCON1<2>) がクリアされると、タイマは命令サイクル毎にインクリメントします (Fosc/4)。TMR3CS がセットされると、カウンタが RB5/TCLK3 ピンの立ち下がりエッジ毎にインクリメントします。どちらのモードでも、タイマ / カウンタをインクリメントするためには、TMR3ON ビットをセットする必要があります。TMR3ON ビットがクリアされると、タイマはインクリメントしないか、フラグビット TMR3IF をセットします。

タイマ 3 には次のような 2 つの動作モードがあり、それは CA1/$\overline{PR3}$ ビット (TCON2<3>) によります。

- **3 つのキャプチャと 1 つの周期レジスタモード**
- **4 つのキャプチャレジスタモード**

PIC17C75X には、キャプチャピンで検出するイベントが起ったときに TMR3 の 16bit 値を取り込む最大 4 つの 16bit キャプチャレジスタがあります。4 つのキャプチャピンがあり (RB0/CAP1、RB1/CAP2、RG4/CAP3、RE3/CAP4)、各キャプチャレジスタペアに対して 1 つです。キャプチャピンは I/O ピンとマルチプレクスされています。イベントは次の通りです。

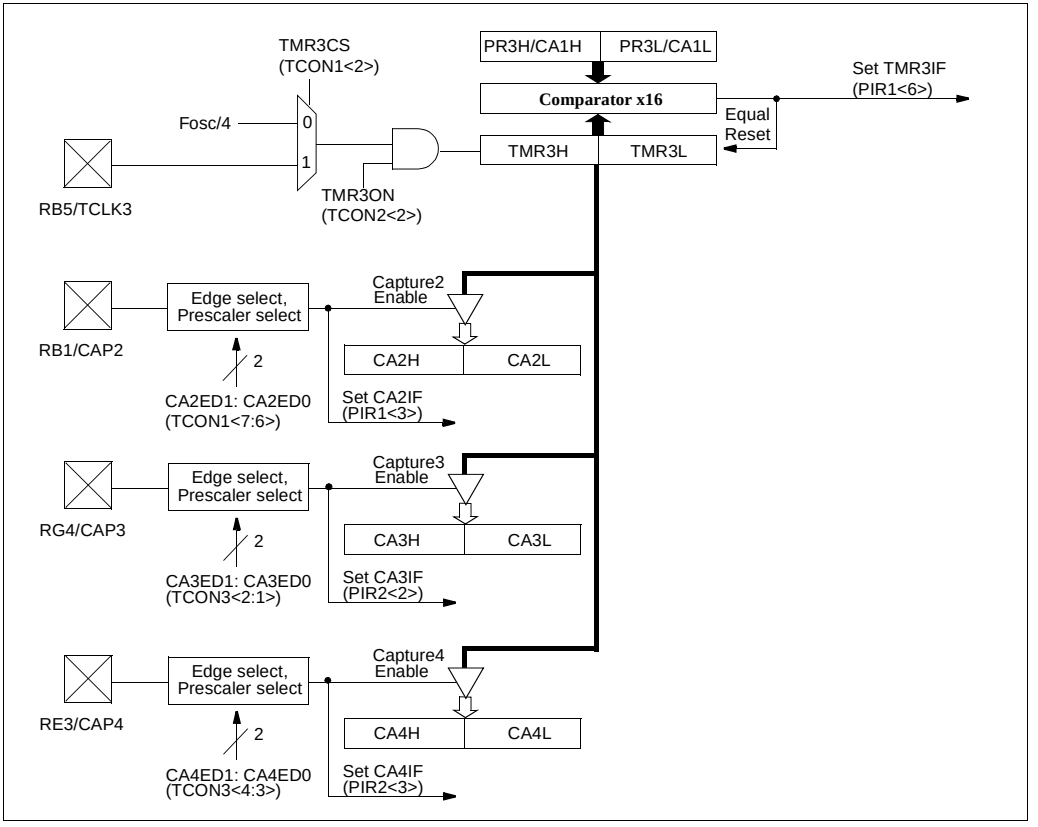
- **立ち上がりエッジ**
- **立ち下がりエッジ**
- **4 回毎の立ち上がりエッジ**
- **16 回毎の立ち上がりエッジ**

各 16bit キャプチャレジスタには、それぞれに伴う割込みフラグが付いています。キャプチャが実行されるとフラグがセットされます。キャプチャモジュールはタイマ 3 ブロックの核の部分です。図 13-8 と 13-9 に、2 つの動作モードのブロック図を示します。

13.2.1 3 つのキャプチャと 1 つの周期レジスタモード

このモードでは、レジスタ PR3H/CA1H と PR3L/CA1L は 16bit の周期レジスタを構成します。図 13-8 にブロック図を示します。タイマは周期レジスタと等しくなるまでインクリメントし、それから次のタイマクロックで 0000h にリセットします。TMR3 割込みフラグビット (TMR3IF) をこのポイントでセットします。この割込みは、TMR3 割込みイネーブルビット (TMR3IE) をクリアすることによりディセーブルできます。TMR3IF はソフトウェアでクリアする必要があります。

図 13-8: 3 つのキャプチャと 1 つの周期レジスタを持つタイマ 3 のブロック図



制御ビット CA1/PR3 をクリアすると、このモード (3 キャプチャ、1 周期) を選択します。このモードでは、キャプチャ1 レジスタは上位バイト (PR3H/CA1H) と下位バイト (PR3L/CA1L) から成り、TMR3 の周期制御レジスタとして設定されます。キャプチャ 1 はこのモードではディセーブルで、対応する割込みビット CA1IF はセットされません。TMR3 は周期レジスタの値と等しくなるまでインクリメントし、次のタイマクロックで 0000h にリセットします。

他のキャプチャはすべてこのモードではアクティブです。

13.2.1.1 キャプチャ動作

CAxED1 と CAxED0 ビットはどのキャプチャでイベントが起こるかを決定します。可能なイベントは下記の通りです。

- **立ち下がりエッジ毎のキャプチャ**
- **立ち上がりエッジ毎のキャプチャ**
- **4 回毎の立ち上がりエッジのキャプチャ**
- **16 回毎の立ち上がりエッジのキャプチャ**

キャプチャが起こると、割込みフラグが CAx1F ビットにラッチされます。この割込みは対応するマスクビットCAxIEをセットすることによりイネーブルにできます。割込みを認識するために、周辺割込みイネーブルビット (PEIE) をセットし、全体割込みディセーブルビット (GLINTD) をクリアする必要があります。CAxIF 割込みフラグビットはソフトウエアでクリアします。

キャプチャのプリスケール選択が変わると、プリスケーラはリセットされずイベントが発生することがあります。そのために、変化後の最初のキャプチャがあいまいになります。しかし次のキャプチャにはタイムベースをセットします。プリスケーラはチップリセットでリセットされます。

キャプチャピン CAPx は、多重化されたピンです。ポートピンとして使用すると、キャプチャはディセーブルになりません。しかし CAxIE をクリアすることによりキャプチャ割込みを簡単にディセーブルできます。CAPx ピンを出力ピンとして使用すると、ポートピンにライトすることによりキャプチャを作動できます。これは開発期間中にキャプチャ割込みをエミュレートするときに役に立ちます。

キャプチャピン CAPx の入力は内部位相クロックに内部で同期します。これは入力波形に一定の制限を強制的に与えます (タイミングの電気的仕様を参照)。

キャプチャのオーバーフローステータスフラグビットはダブルバッファされています。1 つのキャプチャされたワードがすでにキャプチャレジスタ (CAxH:CAxL) にあると、そのマスタビットはセットされ、もう 1 つの “ イベント ” が CAPx ピンに起こった場合、新しいイベントは TMR3 値をキャプチャレジスタに転送せず、前のリードしていないキャプチャ値を保護します。キャプチャレジスタの上位下位両バイトを (どの順番でも) リードする時は、マスタオーバーフロービットをスレーブオーバーフロービット (CAxOVF) に転送してから、マスタビットをリセットします。その時 CAxOVF の値を決めるために TCONx をリードできます。

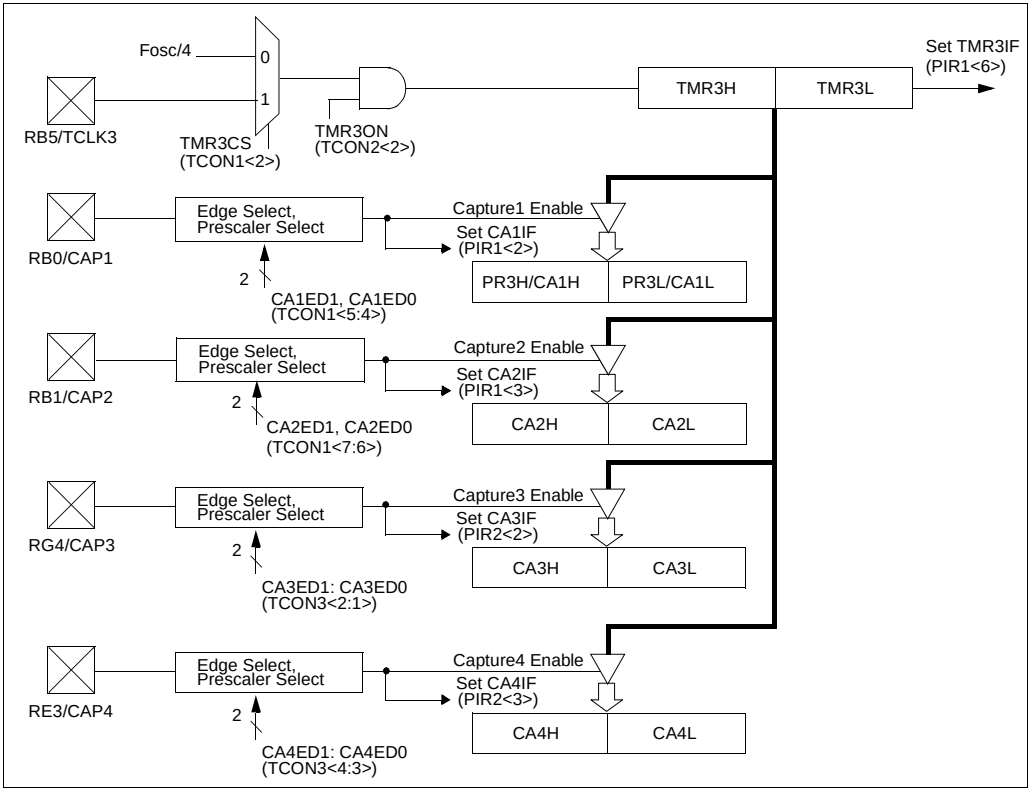
キャプチャレジスタとキャプチャオーバーフローフラグビットをリードするための推奨のシーケンスを例 13-1 に示します。

13.2.2 4つのキャプチャモード

このモードはビット CA1/$\overline{PR3}$ をセットすることにより選択されます。ブロック図を図 13-9 に示します。このモードでは、TMR3 は周期レジスタなしに動き、0000h から FFFFh にインクリメントし、0000h にロールオーバーします。TMR3 割込みフラグ (TMR3IF) はこのロールオーバーでセットされます。TMR3IF ビットをソフトウェアでクリアする必要があります。

レジスタ PR3H/CA1H と PR3L/CA1L は 16bit のキャプチャレジスタ (キャプチャ1) を作り、ピン RB0/CAP1 のイベントを取り込みます。キャプチャモードは CA1ED1 と CA1ED0 ビットにより設定します。キャプチャ 1 割込みフラグビット (CA1IF) は、キャプチャイベントの検出でセットされます。対応する割込みマスクビットは CA1IE です。キャプチャ 1 オーバーフローステータスビットは CA1OVF です。同じ方法ですべてのキャプチャが動作します。キャプチャの動作については 13.2.1 章を参照してください。

図 13-9: 4つのキャプチャを持つタイマ 3 のブロック図



13.2.3 キャプチャレジスタのリード

キャプチャのオーバーフローステータスフラグビットはダブルバッファされています。1 つのキャプチャされたワードがすでにキャプチャレジスタにあると、そのマスタビットはセットされ、もう 1 つの " イベント " が CAPx ピンに起こっています。新しいイベントは TMR3 値をキャプチャレジスタに転送せず、前のリードしていないキャプチャ値を保護します。キャプチャレジスタの上位下位両バイトを (どの順番でも) リードする時は、マスタオーバーフロービットをスレーブオーバーフロービット (CAxOVF) に転送してから、マスタビットをリセットします。その時 CAxOVF の値を決めるために TCONx をリードできます。

キャプチャレジスタとキャプチャオーバーフローフラグビットをリードするための命令のシーケンスを例 13-1 に示します。キャプチャソースによって、リードするには異なったレジスタが必要です。

例 13-1: キャプチャレジスタをリードするためのシーケンス

```
MOVLB 3                  ; Select Bank 3
MOVPF CA2L, LO_BYTE      ; Read Capture2 low byte, store in LO_BYTE
MOVPF CA2H, HI_BYTE      ; Read Capture2 high byte, store in HI_BYTE
MOVPF TCON2, STAT_VAL    ; Read TCON2 into file STAT_VAL
```

表 13-6: キャプチャに関連するレジスタ

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR・BOR での値	他のリセットでの値 (注 1)
16h, Bank 3	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h, Bank 3	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/PR3	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
16h, Bank 7	TCON3	—	CA4OVF	CA3OVF	CA4ED1	CA4ED0	CA3ED1	CA3ED0	PWM3ON	-000 0000	-000 0000
12h, Bank 2	TMR3L	16bitTMR3 レジスタの下位バイト用保持レジスタ								xxxx xxxx	uuuu uuuu
13h, Bank 2	TMR3H	16bitTMR3 レジスタの上位バイト用保持レジスタ								xxxx xxxx	uuuu uuuu
16h, Bank 1	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	0000 0010	0000 0010
17h, Bank 1	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	0000 0000	0000 0000
10h, Bank 4	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	000- 0010	000- 0010
11h, Bank 4	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	000- 0000	000- 0000
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	--11 1100	--11 qq11
16h, Bank 2	PR3L/CA1L	タイマ 3 周期レジスタ、下位バイト／キャプチャ 1 レジスタ、下位バイト								xxxx xxxx	uuuu uuuu
17h, Bank 2	PR3H/CA1H	タイマ 3 周期レジスタ、上位バイト／キャプチャ 1 レジスタ、上位バイト								xxxx xxxx	uuuu uuuu
14h, Bank 3	CA2L	キャプチャ 2 下位バイト								xxxx xxxx	uuuu uuuu
15h, Bank 3	CA2H	キャプチャ 2 上位バイト								xxxx xxxx	uuuu uuuu
12h, Bank 7	CA3L	キャプチャ 3 下位バイト								xxxx xxxx	uuuu uuuu
13h, Bank 7	CA3H	キャプチャ 3 上位バイト								xxxx xxxx	uuuu uuuu
14h, Bank 7	CA4L	キャプチャ 4 下位バイト								xxxx xxxx	uuuu uuuu
15h, Bank 7	CA4H	キャプチャ 4 上位バイト								xxxx xxxx	uuuu uuuu

凡例 : x= 不定、u= 不変、— = 未使用、' 0 ' としてリード、q= 条件によって決まる値。網掛けの部分はキャプチャでは使われません。

注 1: 他の（パワーアップ以外）リセットは、$\overline{MCLR}$ を通る外部リセットと WDT タイマリセットを含みます。

13.2.4 タイマ 3 の外部クロック入力

TMR3CS がセットされると、16bit の TMR3 はクロック入力 TCLK3 の立ち下がりエッジでインクリメントします。RB5/TCLK3 ピンの入力はサンプリングされ、内部位相クロックにより 2 回毎の命令サイクルで同期しています。これにより、立ち下がりエッジが TCLK3 に現れた時から、TMR3 が実際にインクリメントする時まで遅れがあります。必要な外部クロック入力のタイミングについては、電気的仕様の章を参照してください。図 13-10 に、外部クロックから動作する時のタイミング図を示します。

13.2.5 タイマ 3 のリードとライト

タイマ 3 は 16bit のタイマで、一度に 8bit しかリードまたはライトができないので、タイマが動作中にリードまたはライトする時は注意が必要です。最良の方法はタイマを停止してリードまたはライトの動作をし、それからタイマ 3 を再始動させます (TMR3ON ビットを使用)。しかしタイマ 3 をフリーランニングの状態にしておく必要がある場合、注意しなければなりません。16bit の TMR3 にライトする時は例 13-2、16bit の TMR3 をリードする時は例 13-3 のように使用します。割込みをこのルーチンの間はディセーブルにしなければなりません。

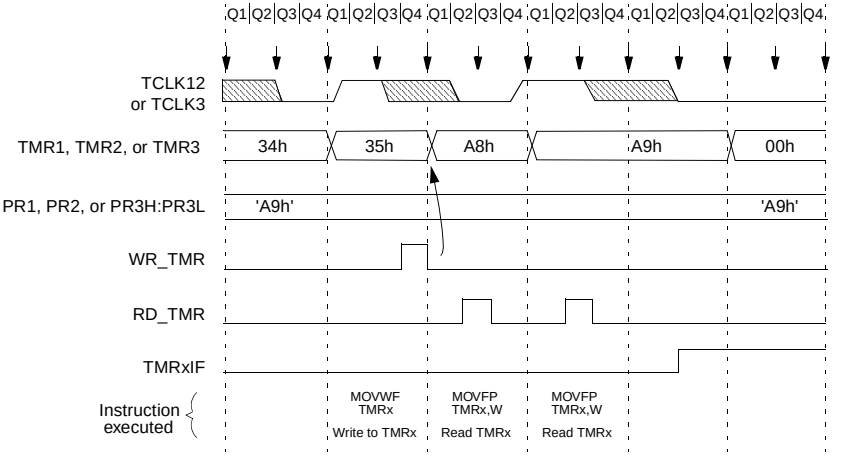
例 13-2: TMR3 へのライト

```
BSF     CPUSTA, GLINTD     ; Disable interrupts
MOVFP   RAM_L,  TMR3L      ;
MOVFP   RAM_H,  TMR3H      ;
BCF     CPUSTA, GLINTD     ; Done, enable interrupts
```

例 13-3: TMR3 からのリード

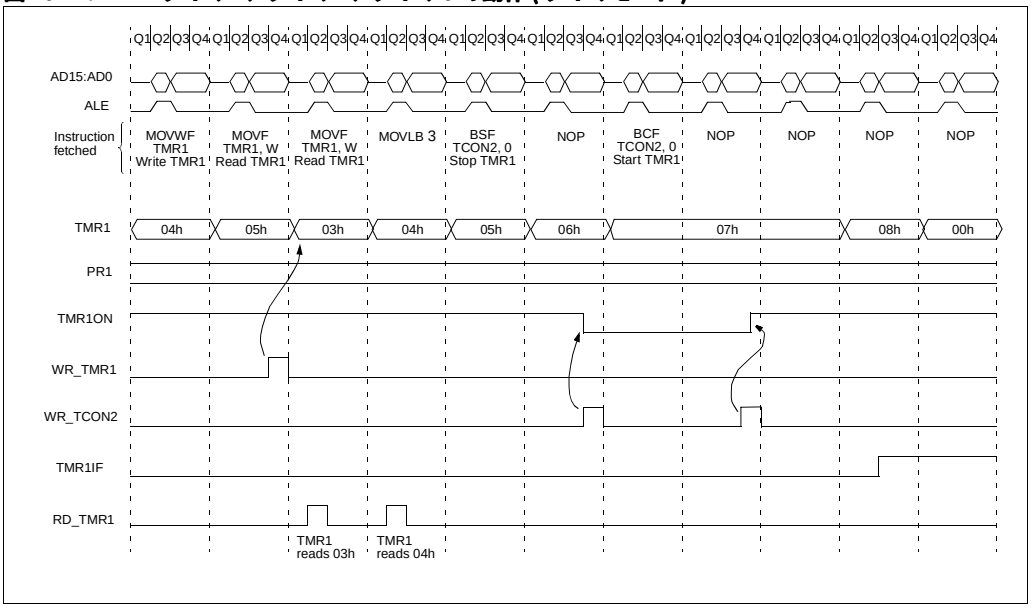
```
MOVPF   TMR3L, TMPLO       ; read low TMR3
MOVPF   TMR3H, TMPHI       ; read high TMR3
MOVFP   TMPLO, WREG        ; tmplo -> wreg
CPFSLT  TMR3L, WREG        ; TMR3L < wreg?
RETURN                     ; no then return
MOVPF   TMR3L, TMPLO       ; read low TMR3
MOVPF   TMR3H, TMPHI       ; read high TMR3
RETURN                     ; return
```

図 13-10: タイマ 1、タイマ 2、タイマ 3 の動作 (カウンタモード)



注意 1:TCLK12 は Q2 と Q4 でサンプリングされます。
2:↓はサンプリングポイントを示します。
3:TCLK12 ↓からタイマのインクリメントまでの潜伏期間は 2Tosc と 6Tosc の間です。

図 13-11: タイマ 1、タイマ 2、タイマ 3 の動作 (タイマモード)



NOTES:

14.0 ユニバーサル同期非同期の受信と送信 (USART) モジュール

各 USART モジュールは、シリアル I/O モジュールです。PIC17C75X で利用できるのは、2 つの USART モジュールで、それらは USART1 と USART2 として設定されています。これらのモジュールの動作の説明は、使用されているレジスタやピンの名称に関しての一般的なものです。表 14-1 に、一般的な名称と USART1 と USART2 の動作の説明で使用する実際の名称を示します。各レジスタの制御ビットは同じ機能なので、名称も同じです (区別する必要はありません)。

送信ステータスと制御レジスタ (TXSTA) を図 14-1 に示し、受信ステータスと制御レジスタ (RCSTA) を図 14-2 に示します。

表 14-1: USART モジュールの総称

総称	USART1 名称	USART2 名称
レジスタ		
RCSTA	RCSTA1	RCSTA2
TXSTA	TXSTA1	TXSTA2
SPBRG	SPBRG1	SPBRG2
RCREG	RCREG1	RCREG2
TXREG	TXREG1	TXREG2
割込み制御ビット		
RCIE	RC1IE	RC2IE
RCIF	RC1IF	RC2IF
TXIE	TX1IE	TX2IE
TXIF	TX1IF	TX2IF
ピン		
RX/DT	RA4/RX1/DT1	RG6/RX2/DT2
TX/CK	RA5/TX1/CK1	RG7/TX2/CK2

図 14-1: TXSTA1 レジスタ (アドレス：15h、バンク 0)
TXSTA2 レジスタ (アドレス：15h、バンク 4)

R/W - 0	R/W - 0	R/W - 0	R/W - 0	U - 0	U - 0	R - 1	R/W - x
CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
-n = POR リセットでの値
(x= 不定)

bit 7: **CSRC**: クロックソース選択ビット
同期モード時
1 = マスタモード (BRG から内部に発生したクロック)
0 = スレーブモード (外部ソースからのクロック)
非同期モード時
無視

bit 6: **TX9**: 9bit の送信選択ビット
1 = 9bit の送信を選択
0 = 8bit の送信を選択

bit 5: **TXEN**: 送信イネーブルビット
1 = 送信をイネーブル
0 = 送信をディセーブル
SREN/CREN は SYNC モードでの TXEN よりも優先される。

bit 4: **SYNC**: USART モード選択ビット
(同期 / 非同期)
1 = 同期モード
0 = 非同期モード

bit 3-2: **Unimplemented**: 未使用 : ‘ 0 ’ としてリード

bit 1: **TRMT**: 送信シフトレジスタ (TSR) エンプティビット
1 = TSR は空
0 = TSR はフル

bit 0: **TX9D**: 送信データの 9 番目のビット (ソフトウェアでパリティを計算するのに使用可能)

USART は、CRT 端子やパーソナルコンピュータのように周辺機能のデバイスと通信することができる全二重非同期システムとして設定することができます。または A/D あるいは D/A の集積回路やシリアル EEPROM などのように周辺機能のデバイスと通信することができる半二重同期システムとして設定することができます。USART は次のモードに設定することができます。

- **非同期 (全二重)**
- **同期ーマスタ (半二重)**
- **同期ースレーブ (半二重)**

SPEN(RCSTA<7>) ビットは、シリアルコミュニケーションインターフェイスとして I/O ピンを設定するためにセットする必要があります。

USART モジュールは RCSTA と TXSTA レジスタの USART コンフィギュレーションビットの状態によって、RX/DT と TX/CX ピンの方向を制御します。I/O の方向を制御するビットは次の通りです。

- **SPEN**
- **TXEN**
- **SREN**
- **CREN**
- **CSRC**

図 14-2: RCSTA1 レジスタ (アドレス：13h、バンク 0)
RCSTA2 レジスタ (アドレス：13h、バンク 4)

R/W - 0	R/W - 0	R/W - 0	R/W - 0	U - 0	R - 0	R - 0	R - x
SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D
bit7							bit 0

R = 読み込み可能なビット
W = 書き込み可能なビット
-n = POR リセットでの値
(x= 未知)

bit 7: **SPEN**: シリアルポートイネーブルビット
1 = TX/CK と RX/DT をシリアルポートピンとして設定
0 = シリアルポートをディセーブル

bit 6: **RX9**: 9bit の受信選択ビット
1 = 9bit の受信を選択
0 = 8bit の受信を選択

bit 5: **SREN**: シングル受信イネーブルビット
このビットはシングルバイトの受信をイネーブル。バイトの受信後、このビットは自動的にクリアされる。
同期モード時
1 = 受信をイネーブル
0 = 受信をディセーブル
注意：このビットは同期スレーブ受信では無視される。
非同期モード時
無視

bit 4: **CREN**: 継続受信イネーブルビット
このビットはシリアルデータの継続受信をイネーブル。
非同期モード時
1 = 継続受信をイネーブル
0 = 継続受信をディセーブル
同期モード時
1 = CREN がクリアされるまで継続受信をイネーブル (CREN が SREN よりも優先)
0 = 継続受信をディセーブル

bit 3: **Unimplemented**: 未使用 : ‘ 0 ’ としてリード

bit 2: **FERR**: フレーミングエラービット
1 = フレーミングエラー (RCREG のリードによりアップデート)
0 = フレーミングエラーなし

bit 1: **OERR**: オーバーランエラービット
1 = オーバーラン (CREN をクリアすることによりクリア)
0 = オーバーランエラーなし

bit 0: **RX9D**: 受信データの 9 番目のビット (ソフトウェアでパリティビットを計算可能)

図 14-3: USART 送信

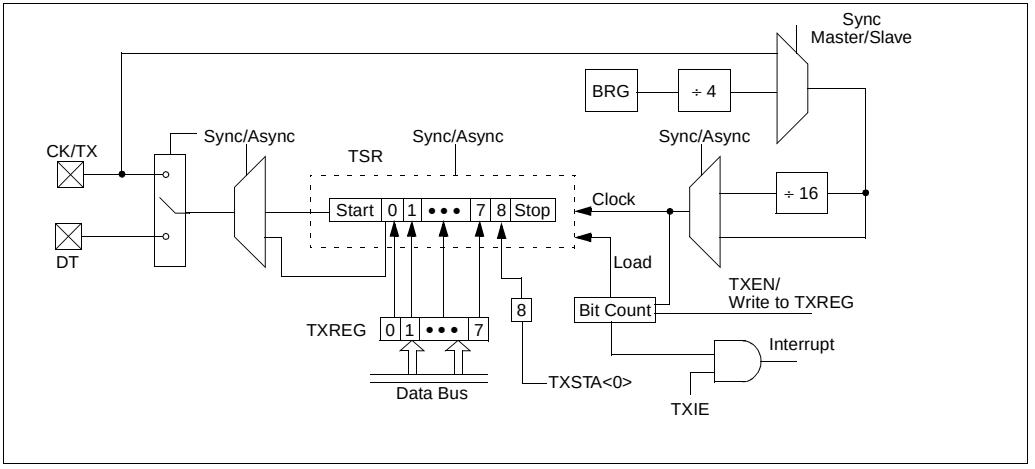
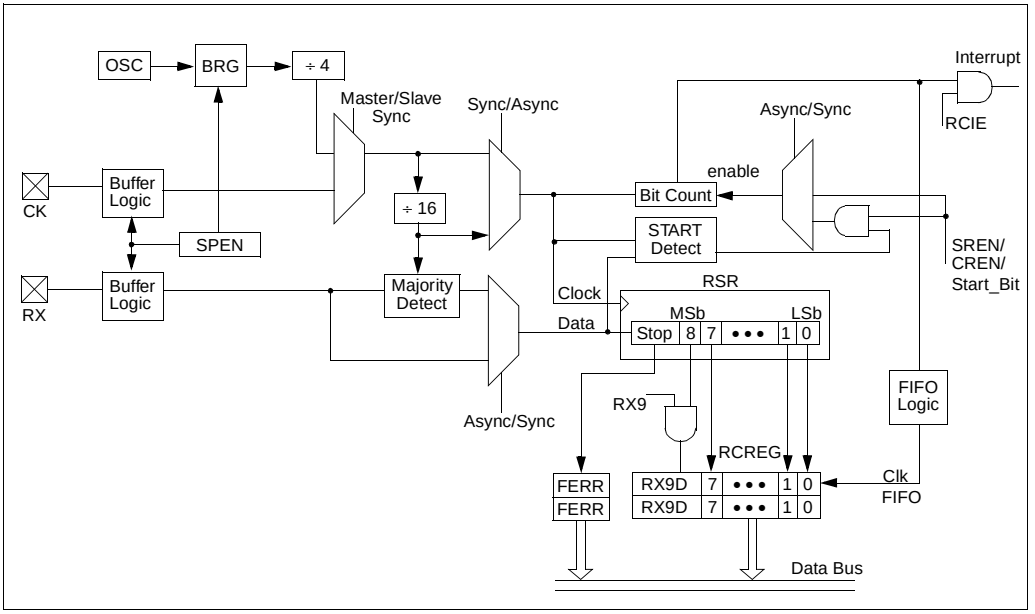


図 14-4: USART 受信



14.1 USART ボーレートジェネレータ (BRG)

BRG は USART の非同期、同期の両モードをサポートしています。それは与えられた 8bit のボーレートジェネレータです。SPBRG レジスタはフリーランニングの 8bit タイマの周期を制御します。表 14-2 に異なった USART モードに対するボーレートの計算の公式を示します。これらは、USART が同期マスタモード (内部クロック) か非同期モードの時にのみ適用します。

必要なボーレートと Fosc が与えられると、0 と 255 の間で最も近い整数値を下記の公式を使って数えることができます。それからボーレートのエラーが決まります。

表 14-2: ボーレートの公式

SYNC	Mode	Baud Rate
0	非同期	Fosc/(64(X+1))
1	同期	Fosc/(4(X+1))

X=SPBRG での値 (0 から 255)

例 14-1 に次の条件に対するボーレートエラーの計算を示します。

FOSC = 16 MHz
要求 ボーレート = 9600
SYNC = 0

例 14-1: ボーレートエラーの計算

要求ボーレート=Fosc / (64 (X + 1))

9600 = 16000000 /(64 (X + 1))

X = 25.042 = 25

計算されたボーレート=16000000 / (64 (25 + 1))

= 9615

Error = (計算されたボーレート - 要求ボーレート) / 要求ボーレート

= (9615 - 9600) / 9600

= 0.16%

SPBRG に新しい値を書き込むことにより BRG タイマがリセットされ (またはクリアされ)、これにより新しいボーレートを出力する前に BRG がタイマオーバーフローを待たないことを確実にします。

表 14-3: ボーレートジェネレータに関連するレジスタ

	アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR・BOR での値	他のリセットでの値 (注 1)
USART1	13h, Bank 0	RCSTA1	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
	15h, Bank 0	TXSTA1	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
	17h, Bank 0	SPBRG1	ボーレートジェネレータレジスタ								xxxx xxxx	uuuu uuuu
USART2	13h, Bank 4	RCSTA2	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
	15h, Bank 4	TXSTA2	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
	17h, Bank 4	SPBRG2	ボーレートジェネレータレジスタ								xxxx xxxx	uuuu uuuu

凡例 : x= 不定、u= 不変、－ = 未使用、‘ 0 ’ としてリード。網掛け部分はボーレートジェネレータでは使用しません。
注 1: 他の（パワーアップ以外）リセットは、$\overline{MCLR}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

表 14-4: 同期モードに関するボーレート

BAUD RATE (K)	FOSC = 33 MHz			FOSC = 25 MHz			FOSC = 20 MHz			FOSC = 16 MHz		
	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)
0.3	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
1.2	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
2.4	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
9.6	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
19.2	**NA**	**—**	**—**	**NA**	**—**	**—**	**19.53**	**+1.73**	**255**	**19.23**	**+0.16**	**207**
76.8	**77.10**	**+0.39**	**106**	**77.16**	**+0.47**	**80**	**76.92**	**+0.16**	**64**	**76.92**	**+0.16**	**51**
96	**95.93**	**-0.07**	**85**	**96.15**	**+0.16**	**64**	**96.15**	**+0.16**	**51**	**95.24**	**-0.79**	**41**
300	**294.64**	**-1.79**	**27**	**297.62**	**-0.79**	**20**	**294.1**	**-1.96**	**16**	**307.69**	**+2.56**	**12**
500	**485.29**	**-2.94**	**16**	**480.77**	**-3.85**	**12**	**500**	**0**	**9**	**500**	**0**	**7**
HIGH	**8250**	**—**	**0**	**6250**	**—**	**0**	**5000**	**—**	**0**	**4000**	**—**	**0**
LOW	**32.22**	**—**	**255**	**24.41**	**—**	**255**	**19.53**	**—**	**255**	**15.625**	**—**	**255**

BAUD RATE (K)	FOSC = 10 MHz			FOSC = 7.159 MHz			FOSC = 5.068 MHz		
	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)
0.3	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
1.2	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
2.4	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
9.6	**9.766**	**+1.73**	**255**	**9.622**	**+0.23**	**185**	**9.6**	**0**	**131**
19.2	**19.23**	**+0.16**	**129**	**19.24**	**+0.23**	**92**	**19.2**	**0**	**65**
76.8	**75.76**	**-1.36**	**32**	**77.82**	**+1.32**	**22**	**79.2**	**+3.13**	**15**
96	**96.15**	**+0.16**	**25**	**94.20**	**-1.88**	**18**	**97.48**	**+1.54**	**12**
300	**312.5**	**+4.17**	**7**	**298.3**	**-0.57**	**5**	**316.8**	**+5.60**	**3**
500	**500**	**0**	**4**	**NA**	**—**	**—**	**NA**	**—**	**—**
HIGH	**2500**	**—**	**0**	**1789.8**	**—**	**0**	**1267**	**—**	**0**
LOW	**9.766**	**—**	**255**	**6.991**	**—**	**255**	**4.950**	**—**	**255**

BAUD RATE (K)	FOSC = 3.579 MHz			FOSC = 1 MHz			FOSC = 32.768 kHz		
	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)
0.3	**NA**	**—**	**—**	**NA**	**—**	**—**	**0.303**	**+1.14**	**26**
1.2	**NA**	**—**	**—**	**1.202**	**+0.16**	**207**	**1.170**	**-2.48**	**6**
2.4	**NA**	**—**	**—**	**2.404**	**+0.16**	**103**	**NA**	**—**	**—**
9.6	**9.622**	**+0.23**	**92**	**9.615**	**+0.16**	**25**	**NA**	**—**	**—**
19.2	**19.04**	**-0.83**	**46**	**19.24**	**+0.16**	**12**	**NA**	**—**	**—**
76.8	**74.57**	**-2.90**	**11**	**83.34**	**+8.51**	**2**	**NA**	**—**	**—**
96	**99.43**	**_3.57**	**8**	**NA**	**—**	**—**	**NA**	**—**	**—**
300	**298.3**	**-0.57**	**2**	**NA**	**—**	**—**	**NA**	**—**	**—**
500	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
HIGH	**894.9**	**—**	**0**	**250**	**—**	**0**	**8.192**	**—**	**0**
LOW	**3.496**	**—**	**255**	**0.976**	**—**	**255**	**0.032**	**—**	**255**

表 14-5: 非同期モードに関するボーレート

BAUD RATE (K)	FOSC = 33 MHz			FOSC = 25 MHz			FOSC = 20 MHz			FOSC = 16 MHz		
	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)
0.3	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
1.2	**NA**	**—**	**—**	**NA**	**—**	**—**	**1.221**	**+1.73**	**255**	**1.202**	**+0.16**	**207**
2.4	**2.398**	**-0.07**	**214**	**2.396**	**0.14**	**162**	**2.404**	**+0.16**	**129**	**2.404**	**+0.16**	**103**
9.6	**9.548**	**-0.54**	**53**	**9.53**	**-0.76**	**40**	**9.469**	**-1.36**	**32**	**9.615**	**+0.16**	**25**
19.2	**19.09**	**-0.54**	**26**	**19.53**	**+1.73**	**19**	**19.53**	**+1.73**	**15**	**19.23**	**+0.16**	**12**
76.8	**73.66**	**-4.09**	**6**	**78.13**	**+1.73**	**4**	**78.13**	**+1.73**	**3**	**83.33**	**+8.51**	**2**
96	**103.12**	**+7.42**	**4**	**97.65**	**+1.73**	**3**	**104.2**	**+8.51**	**2**	**NA**	**—**	**—**
300	**257.81**	**-14.06**	**1**	**390.63**	**+30.21**	**0**	**312.5**	**+4.17**	**0**	**NA**	**—**	**—**
500	**515.62**	**+3.13**	**0**	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
HIGH	**515.62**	**—**	**0**	**—**	**—**	**0**	**312.5**	**—**	**0**	**250**	**—**	**0**
LOW	**2.014**	**—**	**255**	**1.53**	**—**	**255**	**1.221**	**—**	**255**	**0.977**	**—**	**255**

BAUD RATE (K)	FOSC = 10 MHz			FOSC = 7.159 MHz			FOSC = 5.068 MHz		
	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)
0.3	**NA**	**—**	**—**	**NA**	**—**	**—**	**0.31**	**+3.13**	**255**
1.2	**1.202**	**+0.16**	**129**	**1.203**	**_0.23**	**92**	**1.2**	**0**	**65**
2.4	**2.404**	**+0.16**	**64**	**2.380**	**-0.83**	**46**	**2.4**	**0**	**32**
9.6	**9.766**	**+1.73**	**15**	**9.322**	**-2.90**	**11**	**9.9**	**-3.13**	**7**
19.2	**19.53**	**+1.73**	**7**	**18.64**	**-2.90**	**5**	**19.8**	**+3.13**	**3**
76.8	**78.13**	**+1.73**	**1**	**NA**	**—**	**—**	**79.2**	**+3.13**	**0**
96	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
300	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
500	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
HIGH	**156.3**	**—**	**0**	**111.9**	**—**	**0**	**79.2**	**—**	**0**
LOW	**0.610**	**—**	**255**	**0.437**	**—**	**255**	**0.309**	**—**	**255**

BAUD RATE (K)	FOSC = 3.579 MHz			FOSC = 1 MHz			FOSC = 32.768 kHz		
	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)
0.3	**0.301**	**+0.23**	**185**	**0.300**	**+0.16**	**51**	**0.256**	**-14.67**	**1**
1.2	**1.190**	**-0.83**	**46**	**1.202**	**+0.16**	**12**	**NA**	**—**	**—**
2.4	**2.432**	**+1.32**	**22**	**2.232**	**-6.99**	**6**	**NA**	**—**	**—**
9.6	**9.322**	**-2.90**	**5**	**NA**	**—**	**—**	**NA**	**—**	**—**
19.2	**18.64**	**-2.90**	**2**	**NA**	**—**	**—**	**NA**	**—**	**—**
76.8	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
96	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
300	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
500	**NA**	**—**	**—**	**NA**	**—**	**—**	**NA**	**—**	**—**
HIGH	**55.93**	**—**	**0**	**15.63**	**—**	**0**	**0.512**	**—**	**0**
LOW	**0.218**	**—**	**255**	**0.061**	**—**	**255**	**0.002**	**—**	**255**

14.2 USART 非同期モード

このモードでは、USART は標準の非ゼロ復帰 (NRZ) フォーマットを使用します (1 スタートビット、8 または 9 データビット、1 ストップビット)。最も一般的なデータフォーマットは8bitです。8bitのオンチップボーレートジェネレータはオシレータから標準のボーレート周波数を得るために使われます。USART の送信機と受信機は機能的には独立していますが、同じデータフォーマットとボーレートを使います。ボーレートジェネレータは、ビットシフトレートの× 64 のクロックを作ります。パリティはハードウェアではサポートされていませんが、ソフトウェアで実行することができます (9番目のデータビットとして保持することも可能)。非同期モードは、SLEEP の間は止まっています。

非同期モードは SYNC ビット (TXSTA<4>) をクリアすることにより選ばれます。

USART の非同期モジュールは次のような重要な要素から成っています。

- **ボーレートジェネレータ**
- **サンプリング回路**
- **非同期送信機**
- **非同期受信機**

14.2.1 USART 非同期送信機

USART 送信機のブロック図を図 14-3 に示します。送信機の中心は、送信シフトレジスタです (TSR)。シフトレジスタはリード / ライト送信バッファ(TXREG) からのデータを得ます。TXREG はソフトウェアでのデータにロードします。TSR はストップビットが前の負荷から送信されるまでロードされません。ストップビットが送信されるとすぐ、TSR は TXREG(有効なら) からの新しいデータでロードされます。TXREG がそのデータを TSR に転送すると (現在の BRG サイクルの最後に 1 つの TCY の中で起こる)、TXREG は空になり、割込みビット TXIF がセットされます。この割込みは TXIE ビットをセット / クリアすることによりイネーブル / ディセーブルできます。TXIF は TXIE を無視してセットされ、ソフトウェアではリセットできず、新しいデータが TXREG にロードされた時にだけリセットします。TXIF が TXREG のステータスを示している間に、TRMT(TXSTA<1>) ビットが TSR のステータスを示します。TRMT は TSR が空の時にセットされるリードのみのビットです。割込みなしの論理はこのビットにつながるので、TSR が空かどうかを確かめるためにこのビットをポーリングしなければなりません。

注意 : TSR はユーザ用ではないので、データメモリにはマッピングされていません。

送信はTXEN(TXSTA<5>)ビットをセットすることによりイネーブルになります。現実の送信は TXREG にデータがロードされ、ボーレートジェネレータ (BRG) がシフトクロックを作るまで起こりません (図 14-5 参照)。送信は最初に TXREG をロードし、TXEN をセットすることにより始動させることもできます。一般的には、最初に送信を始動させたとき TSR は空で、TXREG への転送が結果的に、TSR にただちに転送することになり、TXREG を空にすることにもなります。このように連続した転送が可能です (図 14-6 参照)。送信中に TXEN をクリアすることにより送信は打ち切られます。これが送信機をリセットし、TX/CK ピンは高インピーダンスに戻ります。

9bit の送信を選ぶためには、TX9(TXSTA<6>) ビットをセットし、9 番目のビット値を TX9D(TXSTA<0>) に書き込まなければなりません。9 番目のビット値は、8bit のデータを TXREG に書き込む前に、書き込む必要があります。これは TXREG へのデータ書き込みが結果的にTSRへの即値データ転送になるからです(TSR が空の場合)。

非同期の送信をセットする場合は次のように進めます。

1. **適当なボーレートに SPBRG レジスタを初期化する。**
2. **SYNC ビットをクリアし、SPEN ビットをセットすることにより非同期のシリアルポートをイネーブルにする。**
3. **割込みが必要な場合、TXIE ビットをセットする。**
4. **9bit の送信が必要な場合は、TX9 ビットをセットする。**
5. **TXREG レジスタにデータをロードする。**
6. **9bit の送信を選んだ場合は、9 番目のビットは TX9D にロードする必要がある。**
7. **TXEN をセットすることにより送信をイネーブルにする (送信開始)。**

送信データを TXREG に書き込み、送信をイネーブル (TXEN をセット) することにより、反対の順番でこれら 2 つのイベントを実行するよりも早く送信を始動させることができます。

注意 : 送信を終了するためには、SPENビットかTXENビットをクリアします。これにより送信の論理をリセットし、送信が再イネーブルされた時に正しい状態にします。

図 14-5: 非同期マスタ送信

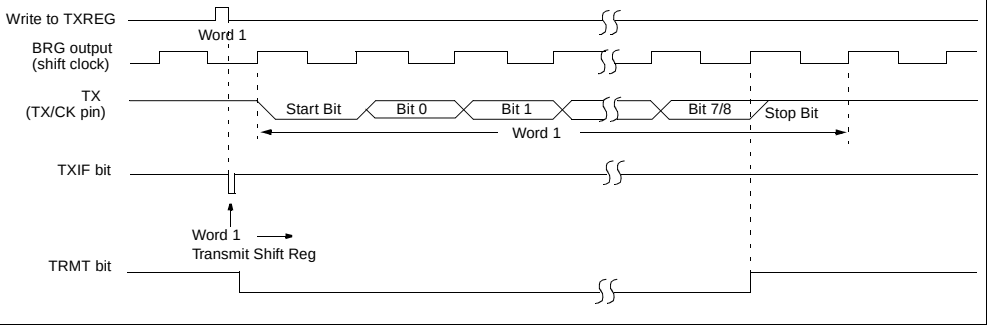
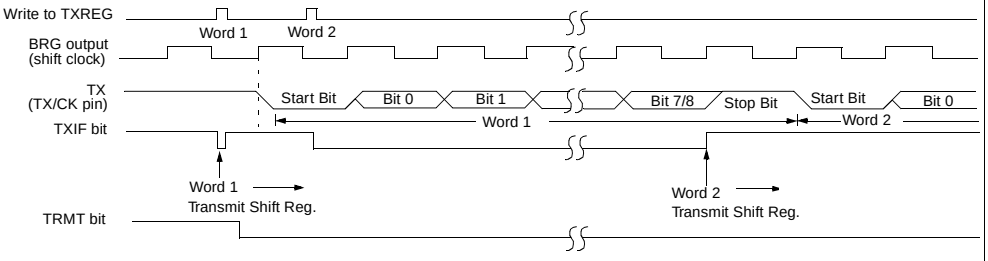


図 14-6: 非同期マスタ送信 (連続)



注意:このタイミング図は２つの連続した送信を示します。

表 14-6: 非同期送信に関連するレジスタ

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR・BOR での値	他のリセットでの値 (注 1)
16h, Bank 1	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	0000 0010	0000 0010
17h, Bank 1	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	0000 0000	0000 0000
13h, Bank 0	RCSTA1	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
16h, Bank 0	TXREG1	シリアルポート送信レジスタ (USART1)								xxxx xxxx	uuuu uuuu
15h, Bank 0	TXSTA1	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 0	SPBRG1	ボーレートジェネレータレジスタ (USART1)								xxxx xxxx	uuuu uuuu
10h, Bank 4	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	000- 0010	000- 0010
11h, Bank 4	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	000- 0000	000- 0000
13h, Bank 4	RCSTA2	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
16h, Bank 4	TXREG2	シリアルポート送信レジスタ (USART2)								xxxx xxxx	uuuu uuuu
15h, Bank 4	TXSTA2	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 4	SPBRG2	ボーレートジェネレータレジスタ (USART2)								xxxx xxxx	uuuu uuuu

凡例 : x= 不定、u= 不変、－ = 未使用、‘ 0 ’としてリード。網掛け部分は非同期送信には使われません。

注 1: 他の（パワーアップ以外）リセットは、$\overline{MCLR}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

14.2.2 USART の非同期受信機

受信機のブロック図を図 14-4 に示します。データは RX/DT ピンから来て、データ回復ブロックを動かします。主な受信のシリアルシフタがビットレートまたは Fosc で動作しているのに反して、データ回復ブロックは現実にはボーレートの１６倍のハイスピードでを動作しているシフタです。

非同期モードが選ばれると、受信はビット CREN(RCSTA<4>) をセットすることによりイネーブルになります。

受信機の中心は受信(シリアル)シフトレジスタです (RSR)。ストップビットをサンプリングした後、RSR に受信されたデータは RCREG に転送されます (空の場合)。転送が終了すると、割込みビット RCIF がセットされます。現実の割込みは RCIE ビットをセット / クリアすることによりイネーブル / ディセーブルが可能です。RCIF はリードのみのビットで、ハードウェアによりクリアされます。RCREG が読み込まれた時と空の時はクリアされます。RCREG はダブルバッファレジスタ (２個の深い FIFO) です。２バイトのデータが RCREG FIFO に受信・転送され、３番目のバイトが RSR にシフティングを始めることは可能です。3 番目のバイトのストップビットの検出では、RCREG がまだフルの場合は、オーバーランエラービット OERR(RCSTA<1>) がセットされます。RSR のワードは失われます。RCREG は FIFO の 2 個のバイトを取り出すために 2 度読み込むことができます。OERR ビットをソフトウェアでクリアする必要があり、これは受信の論理をリセットすることにより行われます(CREN をセット)。OERR ビットがセットされる場合、RSR から RCREG への転送は禁止されていますので、もしセットされている場合は OERR ビットをクリアすることが重要です。エラービット FERR(RCSTA<2>) のフレーミングは、ストップビットが検出されないとセットされます。

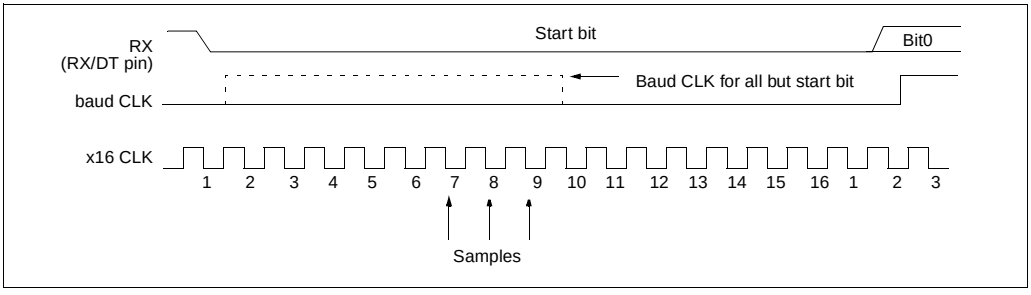
注意： FERR と 9 番目の受信ビットは受信データと同じ方法でバッファされます。RCREG レジスタを読み込むことにより、RX9D と FERR ビットに次の受信されたデータの値をロードできます。したがって古い FERR と RX9D の情報を失わないために、RCREG を読み込む前に RCSTA レジスタを読み込むことが重要です。

14.2.3 サンプリング

RX/DT ピンのデータは、RX/DT ピンが高レベルまたは低レベルであるかどうかを決めるために多数検出回路により 3 回サンプリングされます。サンプリングは x16 クロックの 7 番目、8 番目、9 番目の立ち下がりエッジで行われます (図 14-7)。

x16 のクロックはフリーランニングクロックで、3 回のサンプリングポイントは16回毎の立ち下がりエッジの頻度で起こります。

図 14-7: RX ピンのサンプリング方式

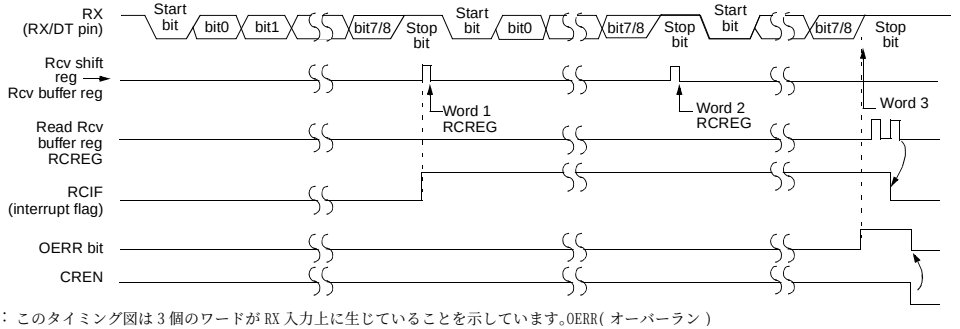


非同期の受信をセットする場合は次のように進めます。

1. 適当なボーレートに SPBRG レジスタを初期化する。
2. SYNC ビットをクリアし、SPEN ビットをセットすることにより非同期のシリアルポートをイネーブルにする。
3. 割込みが必要な場合、RCIE ビットをセットする。
4. 9bit の受信が必要な場合は、RX9 ビットをセットする。
5. CREN ビットをセットすることにより受信をイネーブルにする。
6. RCIF ビットは受信が完了した時にセットされ、RCIE ビットがセットされていると割込みが発生する。
7. ９番目のビットを取るためにはRCSTAを読み込み(イネーブルの場合)、受信中にエラーが起こったかどうかを確かめるためには FERR ビットを読み込む。
8. ８ bit の受信データに対して RCREG を読み込む。
9. オーバーランエラーが起こった場合は、OERRビットをクリアすることによりエラーをクリアする。

注意：受信を終了するためには、SREN ビット、CREN ビット、SPEN ビットのいずれかをクリアします。これにより受信の論理をリセットし、受信が再イネーブルされた時に正しい状態にします。

図 14-8: 非同期受信



注意：このタイミング図は 3 個のワードが RX 入力上に生じていることを示しています。OERR(オーバーラン) ビットがセットされるので、RCREG(受信バッファ) は 3 番目のワードの後読み込まれます。

表 14-7: 非同期受信に関連するレジスタ

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR・BOR での値	他のリセットでの値 (注 1)
16h, Bank 1	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	0000 0010	0000 0010
17h, Bank 1	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	0000 0000	0000 0000
13h, Bank 0	RCSTA1	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h, Bank 0	RCREG1	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	xxxx xxxx	uuuu uuuu
15h, Bank 0	TXSTA1	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 0	SPBRG1	ボーレートジェネレータレジスタ								xxxx xxxx	uuuu uuuu
10h, Bank 4	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	000- 0010	000- 0010
11h, Bank 4	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	000- 0000	000- 0000
13h, Bank 4	RCSTA2	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h, Bank 4	RCREG2	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	xxxx xxxx	uuuu uuuu
15h, Bank 4	TXSTA2	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 4	SPBRG2	ボーレートジェネレータレジスタ								xxxx xxxx	uuuu uuuu

凡例： x= 不定、u= 不変、－ = 未使用、‘0’としてリード。網掛け部分は非同期受信には使われません。

注 1: 他の（パワーアップ以外）リセットは、$\overline{MCLR}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

14.3 USART 同期マスタモード

マスタ同期モードでは、データは半二重の方法で送信されます。すなわち送信と受信は同時には起こりません。データを送信している時は受信は禁止され、逆もまた同じです。SYNC(TXSTA<4>) ビットをセットすることにより同期モードになります。さらに SPEN(RCSTA<7>) ビットは、I/O ピンをそれぞれ CK(クロック) と DT(データ) ラインに設定するためにセットされます。マスタモードはプロセッサが CK ライン上でマスタクロックを送信することを示します。CSRC(TXSTA<7>) ビットをセットすることにより、マスタモードになります。

14.3.1 USART 同期マスタ送信

USART 送信機のブロック図を図 14-3 に示します。送信機の中心は、送信 (シリアル) シフトレジスタです (TSR)。シフトレジスタはリード / ライト送信バッファ TXREG からのデータを得ます。TXREG にはソフトウェアでのデータがロードされます。TSR は最後のビットがその前の負荷から送信されるまでロードされません。最後のビットが送信されるとすぐ、TSR は TXREG(有効なら) からの新しいデータでロードされます。TXREG がそのデータを TSR に転送すると (現在の BRG サイクルの最後の 1 つの Tcy の中で起こる)、TXREG は空になり、TXIF ビットがセットされます。この割込みは TXIE ビットをセット / クリアすることによりイネーブル / ディセーブルできます。TXIF はビット TXIE の状態を無視してセットされ、ソフトウェアではクリアできません。新しいデータが TXREG にロードされた時にだけリセットします。TXIF が TXREG のステータスを示している間に、TRMT(TXSTA<1>) が TSR のステータスを示します。TRMT は TSR が空の時セットされるリードのみのビットです。割込みの論理はこのビットにつながれていないので、TSR が空かどうかを確かめるためにこのビットをポーリングしなければなりません。TSR はユーザ用ではないので、データメモリには表われません。

送信は TXEN(TXSTA<5>) ビットをセットすることによりイネーブルになります。現実の送信は TXREG にデータがロードされるまで起こりません。最初のデータビットは TK/CK ピンのクロックの次に有効な立ち上がりエッジでシフトアウトします。データ出力は、同期クロックの立ち下がりエッジのあたりでは安定しています (図 14-10 参照)。送信は最初に TXREG をロードし、TXEN をセットすることにより始動させることもできます。これは、遅いボーレートが選ばれた時に便利で、TXEN、CREN、SREN ビットがクリアされた時、BRG が RESET 状態を続けるためです。TXEN ビットをセットすると、ただちにシフトクロックを作りながら BRG を始動します。一般的には、最初に送信を始動させたとき TSR は空で、TXREG への転送が結果的に、TSR にただちに転送することになり TXREG を空にすることにもなります。連続した転送は可能です。

送信中に TXEN をクリアすることにより送信は打ち切られ、送信機をリセットします。RK/DT と TX/CK ピンは高インピーダンスに戻ります。送信中に CREN か SREN のどちらかがセットされると、送信は打ち切られ、RX/DT ピンが高インピーダンス状態に戻ります (受信として)。CSRC ビットがセットされると (内部クロック)、TX/CK ピンは出力のままです。送信機の論理は、ピンから分離するけれどもリセットされません。送信機をリセットするためには、TXEN ビットをクリアする必要があります。SREN ビットがセットされると (継続中のシングルワードの送信と受信に割込むために)、シングルワードが受信された後、SREN はクリアされます。またシリアルポートは、TXEN ビットがまだセットされているので送信状態に戻ります。DT ラインはただちに高インピーダンスの受信モードから送信に切り替わり、駆動を開始します。これを避けるためには、TXEN をクリアする必要があります。

9bit の送信を選ぶためには、TX9(TXSTA<6>) ビットをセットし、9 番目のビットを TX9D(TXSTA<0>) に書き込まなければなりません。9 番目のビットは、8bit のデータを TXREG に書き込む前に、書き込む必要があります。これは TXREG へのデータ書き込みが結果的に、TSR への即値データ転送になるからです (TSR が空の場合)。TSR が空で、TXREG が、“ 新しい ” TX9D を書き込む前に書き込まれる場合は、TX9D の “ 現在の ” 値はロードされます。

同期のマスター送信をセットする場合は次のように進めます。

1. **適当なボーレートに SPBRG レジスタを初期化する。(詳細はボーレートジェネレータの章参照)。**
2. **SYNC、SPEN、CSRC ビットをセットすることにより同期のマスタシリアルポートをイネーブルにする。**
3. **CREN ビットと SREN ビットを確実にクリアする (これらのビットはセットすると送信をオーバーライドする)。**
4. **割込みが必要な場合は、TXIE ビットをセットする (GLINTD ビットをクリアし PEIE ビットをセットすることが必要)。**
5. **9bit の送信が必要な場合は、TX9 ビットをセットする。**
6. **TXREG レジスタにデータをロードすることにより送信を開始する。**
7. **9bit の送信を選んだ場合は、9 番目のビットは TX9D にロードする必要がある。**
8. **TXEN をセットすることにより送信をイネーブルにする。**

送信データを TXREG に書き込み、送信をイネーブル (TXEN をセット) することにより、反対の順番でこれら 2 つのイベントを実行するよりも早く送信を始動させることができます。

注意 :	**送信を終了するためには、SPEN ビットか TXEN ビットをクリアします。これにより送信の論理をリセットし、送信が再イネーブルされた時に正しい状態にします。**

表 14-8: 同期マスタ送信に関連するレジスタ

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR, BOR での値	他のリセットでの値 (注 1)
16h, Bank 1	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	0000 0010	0000 0010
17h, Bank 1	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	0000 0000	0000 0000
13h, Bank 0	RCSTA1	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
16h, Bank 0	TXREG1	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	xxxx xxxx	uuuu uuuu
15h, Bank 0	TXSTA1	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 0	SPBRG1	ボーレートジェネレータレジスタ								xxxx xxxx	uuuu uuuu
10h, Bank 4	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	000- 0010	000- 0010
11h, Bank 4	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	000- 0000	000- 0000
13h, Bank 4	RCSTA2	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
16h, Bank 4	TXREG2	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	xxxx xxxx	uuuu uuuu
15h, Bank 4	TXSTA2	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 4	SPBRG2	ボーレートジェネレータレジスタ								xxxx xxxx	uuuu uuuu

凡例 : x= 不定、u= 不変、－ = 未使用、‘ 0 ’ としてリード。網掛け部分は同期マスタ送信には使用しません。
注 1: 他の（パワーアップ以外）リセットは、$\overline{MCLR}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

図 14-9: 同期送信

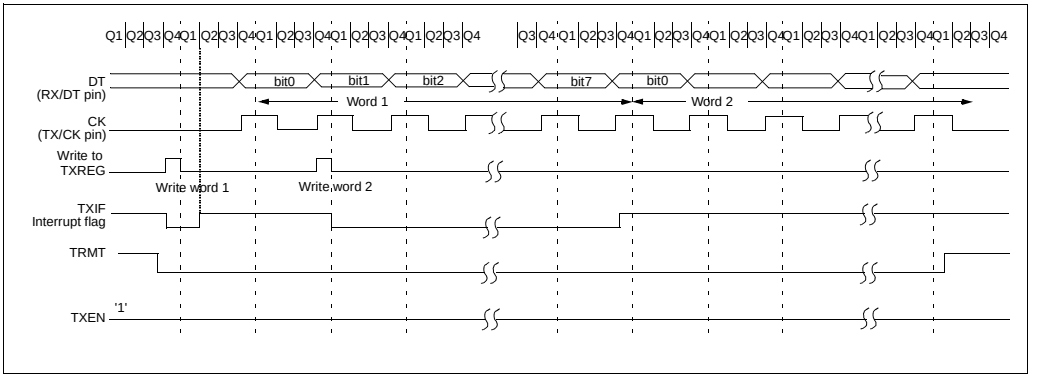
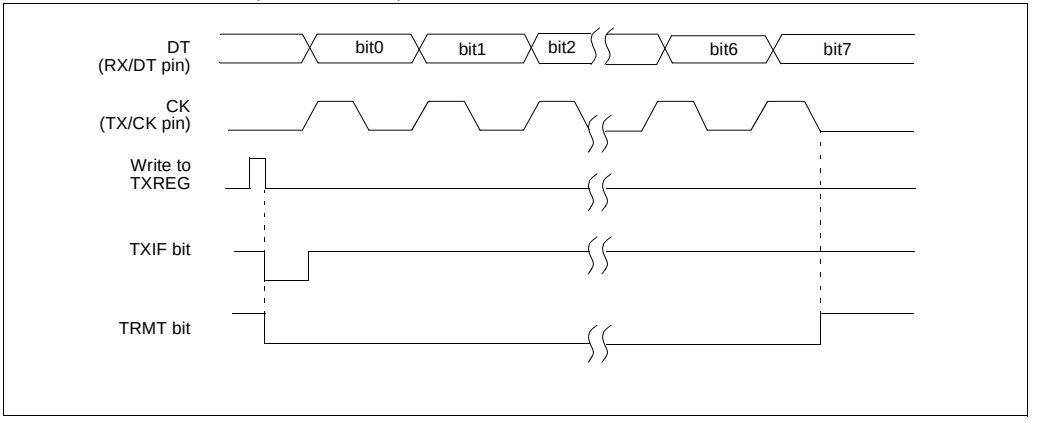


図 14-10: 同期送信 (TXEN を通る)



14.3.2 USART の同期マスタ受信

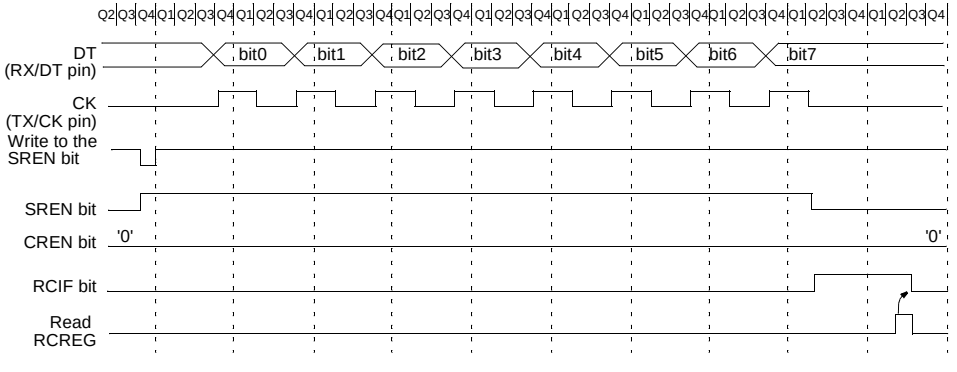
同期モードが選ばれると、受信は SREN(RCSTA<5>) ビットまたはCREN(RCSTA<4>)ビットのどちらかをセットすることによりイネーブルになります。データはクロックの立ち下がりエッジで RX/DT ピンにてサンプリングされます。SREN がセットされると、シングルワードのみが受信されます。CRENがセットされると、受信はCRENがリセットされるまで続きます。両方のビットがセットされる場合は、CREN が優先します。最後のビットをクロックした後、受信シフトレジスタ (RSR) に受信されたデータは RCREG に転送されます (空の場合)。転送が終了すると、割込みビット RCIF がセットされます。現実の割込みは RCIE ビットをセット / クリアすることによりイネーブル / ディセーブルが可能です。RCIF はリードオンリーのビットで、ハードウェアにより RESET されます。この場合ではRCREGが読み込まれた時と空の時にリセットされます。RCREG はダブルバッファレジスタで、2 個の深い FIFO です。2 バイトのデータが RCREG FIFO に受信・転送され、3 番目のバイトが RSR にシフティングを始めることは可能です。3 番目のバイトの最後のビットのクロッキングでは、RCREG がまだフルの場合は、オーバーランエラービット OERR(RCSTA<1>)がセットされます。RSR のワードは失われます。RCREG は FIFO の 2 個のバイトを取り出すために 2 度読み込むことができます。OERR ビットはソフトウェアでクリアする必要があり、これは CREN ビットをクリアすることにより実行します。OERR がセットされている場合、RSR から RCREG への転送は禁止されていますので、もしセットされている場合は OERR ビットをクリアすることが重要です。9 番目の受信ビットは受信データと同じ方法でバッファされます。RCREG レジスタのリードにより、RX9D ビットと FERR ビットを次の受信データの値でロードすることが可能です。したがって古い FERR と RX9D の情報を失わないために、RCREG を読み込む前に RCSTA レジスタを読み込むことが重要です。

同期のマスタ受信をセットする場合は次のように進めます。

1. **適当なボーレートに SPBRG レジスタを初期化する。詳細は 14.1 章を参照。**
2. **ビット SYNC、SPEN、CSRC をセットすることにより同期のマスタシリアルポートをイネーブルにする。**
3. **割込みが必要な場合、RCIE ビットをセットする。**
4. **9bit の受信が必要な場合は、RX9 ビットをセットする。**
5. **単独受信が必要な場合は、ビット SREN をセットする。継続受信の場合はビット CREN をセットする。**
6. **RCIF ビットは受信が完了した時にセットされ、割込みは RCIE ビットがセットされると発生する。**
7. **9番目のビットを取るためにはRCSTAを読み込み (イネーブルの場合)、受信中にエラーが起こったかどうかを確かめる。**
8. **RCREG を読み込むことにより 8bit の受信データを読み込む。**
9. **エラーが起こった場合は、CREN をクリアすることによりエラーをクリアする。**

注意： 受信を終了するためには、SREN ビット、CREN ビット、SPEN ビットのいずれかをクリアします。これにより受信の論理をリセットし、受信が再イネーブルされた時の正しい状態にします。

図 14-11: 同期受信 (マスタモード、SREN)



注意： タイミング図は SREN=1 の時の SYNC のマスタモードを表わしています。

表 14-9: 同期マスタ受信に関連するレジスタ

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR, BOR での値	他のリセットでの値（注 1）
16h, Bank 1	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	0000 0010	0000 0010
17h, Bank 1	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	0000 0000	0000 0000
13h, Bank 0	RCSTA1	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h, Bank 0	RCREG1	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	xxxx xxxx	uuuu uuuu
15h, Bank 0	TXSTA1	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 0	SPBRG1	ボーレートジェネレータレジスタ								xxxx xxxx	uuuu uuuu
10h, Bank 4	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	000- 0010	000- 0010
11h, Bank 4	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	000- 0000	000- 0000
13h, Bank 4	RCSTA2	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h, Bank 4	RCREG2	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	xxxx xxxx	uuuu uuuu
15h, Bank 4	TXSTA2	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 4	SPBRG2	ボーレートジェネレータレジスタ								xxxx xxxx	uuuu uuuu

凡例 : x= 不定、u= 不変、－ = 未使用、‘0’としてリード。網掛け部分は同期マスタ受信には使われません。

注 1: 他の（パワーアップ以外）リセットは、$\overline{MCLR}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

14.4 USART 同期スレーブモード

同期スレーブモードは、外部からシフトクロックがTX/CKピンに与えられているという点でマスタモードとは違います (マスタモードでは内部から供給されます)。これによりデバイスは SLEEP モード中にデータを転送または受信することができます。CSRC(TXSTA<7>) ビットをクリアすることによりスレーブモードになります。

14.4.1 USART 同期スレーブ送信

同期マスタとスレーブモードの操作は、SLEEP モードの場合を除いて全く同じです。

2 個のワードを TXREG に書き込み、SLEEP 命令を実行すると、次のことが起こります。シフトクロックが供給されると、最初のワードはただちに TSR に転送され送信されます。2 番目のワードは TXREG に残ります。TXIF はセットされません。最初のワードが TSR からシフトアウトされた時には、TXREG は 2 番目のワードを TSR に転送し、TXIF フラグがセットされます。TXIE がイネーブルになると、割込みが SLEEP からチップをウェークし、全割込みがイネーブルされると、プログラムは割込みベクタ (0020h) に分岐します。

同期スレーブ送信をセットする場合は次のように進めます。

1. **SYNC ビットと SPEN ビットをセットし、CSRC ビットをクリアすることにより同期のスレーブシリアルポートをイネーブルにする。**
2. **CREN ビットをクリアする。**
3. **割込みが必要な場合は、TXIE ビットをセットする。**
4. **9 bit の送信が必要な場合は、TX9 ビットをセットする。**
5. **TXREG にデータをロードすることにより送信を開始する。**
6. **9bit の送信を選んだ場合は、9 番目のビットは TX9D にロードする必要がある。**
7. **TXEN をセットすることにより送信をイネーブルにする。**

送信データを TXREG に書き込み、送信をイネーブル (TXEN をセット) することにより、反対の順番でこれら 2 つのイベントを実行するよりも早く送信を始動させることができます。

注意 : 送信を終了するためには、SPEN ビットか TXEN ビットをクリアします。これにより送信の論理をリセットし、送信が再イネーブルされた時に正しい状態にします。

14.4.2 USART 同期スレーブ受信

同期マスタとスレーブモードの操作は、SLEEP モードの場合を除いて全く同じです。さらに SREN もスレーブモードでは関係ありません。

SLEEP 命令より前に受信をイネーブルすると (CREN)、1 個のワードが SLEEP 中に受信されることがあります。完全にそのワードを受信することで、RSR はそのデータを RCREG に転送します (RCIF をセット)。そして RCIE ビットがセットされると、発生した割込みが SLEEP からチップをウェークさせます。全割込みがイネーブルされると、プログラムは割込みベクタ (0020h) に分岐します。

同期スレーブ受信をセットする場合は次のように進めます。

1. **SYNC ビットと SPEN ビットをセットし、CSRC ビットをクリアすることにより同期のマスタシリアルポートをイネーブルにする。**
2. **割込みが必要な場合は、RCIE ビットをセットする。**
3. **9bit の受信が必要な場合は、RX9 ビットをセットする。**
4. **受信をイネーブルするためには、CREN ビットをセットする**
5. **RCIF ビットは受信が完了した時にセットされる。RCIE ビットがセットされると割込みが発生する。**
6. **9 番目のビットを取るためには RCSTA を読み込み (イネーブルの場合)、受信中にエラーが起こったかどうかを確かめる。**
7. **RCREG を読み込むことにより 8bit の受信データを読み込む。**
8. **エラーが起こった場合は、CREN ビットをクリアすることによりエラーをクリアする。**

注意 : 受信を打ち切るためには、SPEN ビットか SREN ビット (単独受信モードの時)、または CREN ビット (継続受信モードの時) をクリアします。これにより受信の論理をリセットし、受信が再イネーブルされた時に正しい状態にします。

表 14-10: 同期スレーブ送信に関連するレジスタ

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR, BOR での値	他のリセットでの値（注 1）
16h, Bank 1	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	0000 0010	0000 0010
17h, Bank 1	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	0000 0000	0000 0000
13h, Bank 0	RCSTA1	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
15h, Bank 0	TXSTA1	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
16h, Bank 0	TXREG1	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	xxxx xxxx	uuuu uuuu
17h, Bank 0	SPBRG1	ボーレートジェネレータレジスタ								xxxx xxxx	uuuu uuuu
10h, Bank 4	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	000- 0010	000- 0010
11h, Bank 4	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	000- 0000	000- 0000
13h, Bank 4	RCSTA2	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
16h, Bank 4	TXREG2	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	xxxx xxxx	uuuu uuuu
15h, Bank 4	TXSTA2	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 4	SPBRG2	ボーレートジェネレータレジスタ								xxxx xxxx	uuuu uuuu

凡例 : x= 不定、u= 不変、－ = 未使用、‘ 0 ’としてリード。網掛け部分は同期スレーブ送信には使われません。
注 1: 他の（パワーアップ以外）リセットは、$\overline{MCLR}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

表 14-11: 同期スレーブ受信に関連するレジスタ

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR, BOR での値	他のリセットでの値（注 1）
16h, Bank1	PIR1	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF	0000 0010	0000 0010
17h, Bank1	PIE1	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TX1IE	RC1IE	0000 0000	0000 0000
13h, Bank0	RCSTA1	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h, Bank0	RCREG1	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	xxxx xxxx	uuuu uuuu
15h, Bank 0	TXSTA1	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 0	SPBRG1	ボーレートジェネレータレジスタ								xxxx xxxx	uuuu uuuu
10h, Bank 4	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	000- 0010	000- 0010
11h, Bank 4	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	000- 0000	000- 0000
13h, Bank 4	RCSTA2	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h, Bank 4	RCREG2	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	xxxx xxxx	uuuu uuuu
15h, Bank 4	TXSTA2	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 4	SPBRG2	ボーレートジェネレータレジスタ								xxxx xxxx	uuuu uuuu

凡例 : x= 不定、u= 不変、－ = 未使用、‘ 0 ’としてリード。網掛け部分は同期のスレーブ受信には使われません。
注 1: 他の（パワーアップ以外）リセットは、$\overline{MCLR}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

15.0 同期シリアルポート (SSP) モジュール

同期シリアルポート (SSP) モジュールは他の周辺回路またはマイクロコントローラデバイスと通信するのに便利なシリアルインターフェイスです。これらの周辺回路デバイスはシリアル EEPROM、シフトレジスタ、ディスプレイドライバ、A/D コンバータなどがあります。SSP モジュールは次の 2 種類のモードのどちらか 1 つで動作することができます。

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I^2C)

アプリケーションノート AN578 "***I^2C マルチマスタ環境での SSP モジュールの使い方***"を参照してください。

図 15-1、15-2、15-3 に、3 つの異なった動作モードのブロック図を示します。

図 15-1: SPI モードのブロック図

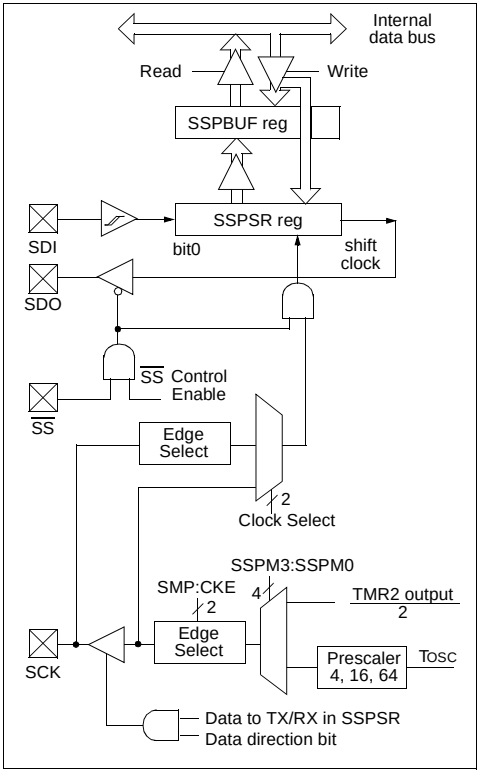


図 15-2: I^2C スレーブモードのブロック図

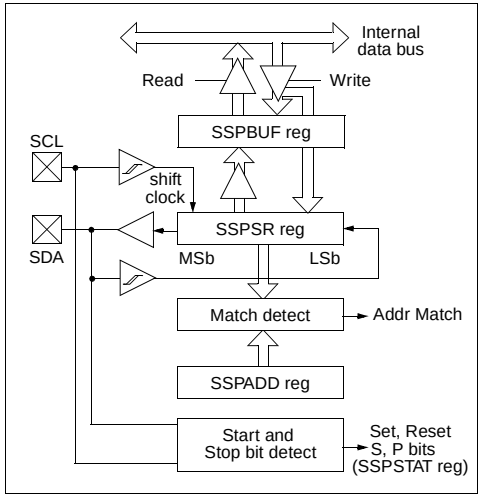


図 15-3: I^2Cマスタモードのブロック図

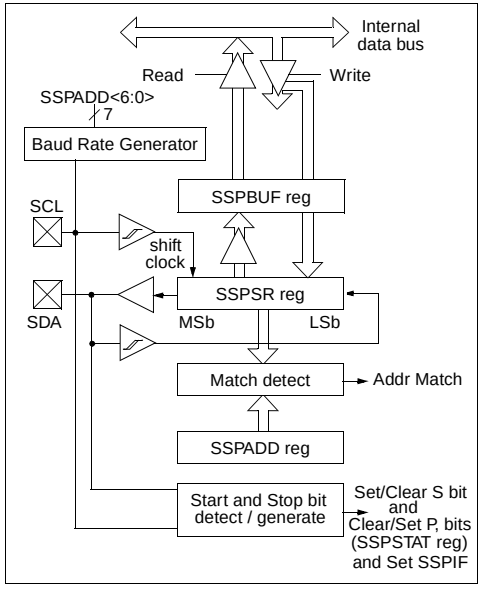


図 15-4: SSPSTAT：同期シリアルポートステータスレジスタ (アドレス：13h、バンク 6)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/A	P	S	R/W	UA	BF
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
U = 未使用のビット、‘0’としてリード
-n =POR リセットでの値

bit 7: **SMP:** SPI データ入力サンプル位相
SPI マスタモード時
1 = データ出力時間の最後にサンプリングされた入力データ
0 = データ出力時間の中間でサンプリングされた入力データ
SPI スレーブモード時
SPI をスレーブモードで使用する時は SMP をクリアする。
I^2C マスタまたはスレーブモードにおいて
1= 標準スピードモードのためにスルーレート制御 (100kHz と 1MHz) をディセーブル
0= 高スピードモードのためにスルーレート (400kHz) をイネーブル

bit 6: **CKE**: SPI クロックエッジ選択 (図 15-8、15-11、15-12 参照)
CKP = 0
1 = SCK の立ち上がりエッジで送信されたデータ
0 = SCK の立ち下がりエッジで送信されたデータ
CKP = 1
1 = SCK の立ち下がりエッジで送信されたデータ
0 = SCK の立ち上がりエッジで送信されたデータ

bit 5: **D/$\overline{A}$**: データ / アドレスビット (I^2C スレーブモードのみ)
1 = 受信または送信した最後のバイトがデータであることを示す。
0 = 受信または送信した最後のバイトがアドレスであることを示す。

bit 4: **P**: ストップビット (I^2C モードのみ。このビットは SSP モジュールがディセーブルされた時クリアされ、SSPEN がクリアされる)。
1 = ストップビットが最後に検出されたことを示す (このビットは RESET 上は ‘0’)。
0 = ストップビットが検出されなかったことを示す。

bit 3: **S**: スタートビット (I^2C モードのみ。このビットは SSP モジュールがディセーブルになり、SSPEN がクリアされた時クリアされる)。
1 = スタートビットが最後に検出されたことを示す (このビットは RESET 上は ‘0’)。
0 = スタートビットが検出されなかったことを示す。

bit 2: **R/$\overline{W}$**: リード / ライトビットの情報 (I^2C モードのみ)。
このビットは最終アドレスマッチに続く R/$\overline{W}$ ビット情報を保持。このビットはアドレスマッチから次のスタートビット、ストップビット、$\overline{ACK}$ ビットまでだけが有効。
I^2C スレーブモードにおいて
1 = リード
0 = ライト
I^2C マスタモードにおいて
1 = 送信中
0 = 送信中ではない。 SSP が IDLE モードであれば、このビットと SAE、RCE、SPE、または AKE との論理和をとることで示めされます。

bit 1: **UA**: アップデートアドレス (10bit の I^2C スレーブモードのみ)。
1 = SSPADD レジスタのアドレスを更新する必要があることを示す。
0 = アドレス更新の必要なし。

bit 0: **BF**: バッファフルステータスビット

受信 (SPI と I^2C モード)
1 = 受信完了。SSPBUF はフル。
0 = 受信中。SSPBUF は空。

送信 (I^2C モードのみ)
1 = データ送信中 ($\overline{ACK}$ とストップビットを含まず)。SSPBUF はフル。
0 = データ送信完了 ($\overline{ACK}$ とストップビットを含まず)。SSPBUF は空。

図 15-5: SSPCON1：同期シリアルポート制御レジスタ 1(アドレス 11h、バンク 6)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
U = 未使用のビット、‘0’としてリード
-n = POR リセットでの値

bit 7: **WCOL**: ライト衝突検出ビット
マスタモード
1 = I^2C 条件が送信をスタートさせるのに有効ではない間に、SSPBUF レジスタへのライトが試みられた場合
0 = 衝突なし
スレーブモード
1 = 前のワードをまだ送信中に SSPBUF レジスタに書き込みが (ソフトウェアでクリアが必要)。
0 = 衝突なし

bit 6: **SSPOV**: 受信オーバーフロー指示ビット

SPI モード
1 = SSPBUF レジスタがまだ前のデータを保持している時に新しい 1 バイトを受信。オーバーフローの場合には SSPSR のデータは失われる。オーバーフローはスレーブモードでのみ起こる。オーバーフローを避けるためには、データ送信を行なっている時でも SSPBUF を読み込む必要がある。マスタモードでは、それぞれの新しい受信 (および送信) は SSPBUF レジスタへのライトによって開始されるので、オーバーフロービットはセットされない。
0 = オーバーフローなし。

I^2C モード
1 = SSPBUF レジスタがまだ前のバイトを保持している時に新しい 1 バイトを受信。送信モードでは SSPOV は Don't care。SSPOV はどちらかのモードでもソフトウェアでクリアが必要。
0 = オーバーフローなし。

bit 5: **SSPEN**: 同期シリアルポートイネーブルビット

SPI モード
1 = シリアルポートをイネーブルにし、SCK、SDO、SDI をシリアルポートピンとして構成。
0 = シリアルポートをディセーブルにし、上記のピンを I/O ポートピンとして構成。

I^2C モード
1 = シリアルポートをイネーブルにし、SDA、SCL ピンをシリアルポートピンとして構成。
0 = シリアルポートをディセーブルにし、上記のピンを I/O ポートピンとして構成。

注意 : 両モードにおいて、イネーブルの時はこれらのピンは入力または出力として適切に構成する必要があります。

bit 4: **CKP**: クロック極性選択ビット
SPI モード
1 = クロックのアイドル状態はハイレベル。
0 = クロックのアイドル状態はローレベル。
I^2C スレーブモード
SCK リリース制御
1 = イネーブルクロック
0 = クロックをローに保持(クロックストレッチ)(データのセットアップ時間確保のために使用される)。
I^2C マスターモード
このモードでは使用されない。

bit 3-0: **SSPM3:SSPM0**: 同期シリアルポートモード選択ビット
0000 = SPI マスタモード、クロック = FOSC/4
0001 = SPI マスタモード、クロック = FOSC/16
0010 = SPI マスタモード、クロック = FOSC/64
0011 = SPI マスタモード、クロック = TMR2 output/2
0100 = SPI スレーブモード、クロック = SCK ピン。 $\overline{SS}$ ピン制御はイネーブル。
0101 = SPI スレーブモード、 クロック = SCK ピン。$\overline{SS}$ ピン制御はディセーブル。$\overline{SS}$ は I/O ピンとして使用可能。
0110 = I^2C スレーブモード、 7-bit アドレス
0111 = I^2C スレーブモード、 10-bit アドレス
1000 = I^2C マスタモード、クロック = FOSC / (4 * (SSPADD+1))
1xx1 = 予備
1x1x = 予備

図 15-6: SSPCON2：同期シリアルポート制御レジスタ 2(アドレス 12h、バンク 6)

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
U = 未使用のビット、
'0' としてリード
-n = POR リセットでの値

bit 7: **GCEN**: ジェネラルコールイネーブルビット (I2C スレーブモードのみ)
1 = ジェネラルコールアドレスが SSPSR に受信されると割込みをイネーブル。
0 = ジェネラルコールアドレスをディセーブル。

bit 6: **ACKSTAT**: アクノリッジステータスビット (I2C マスタモードのみ)
マスタ送信モード：
1 = アクノリッジはスレーブから受信されない。
0 = アクノリッジはスレーブから受信。

bit 5: **ACKDT**: アクノリッジデータビット (I2C マスタモードのみ)
マスタ受信モード：
受信の最後にアクノリッジのシーケンスを始める時に送信される値が
1 = 応答せず。
0 = 応答。

bit 4: **ACKEN**: アクノリッジシーケンスイネーブルビット (I2C マスタモードのみ)
マスタ受信モード：
1 = SDA と SCL ピンで応答のシーケンスを開始し、AKD データビットを送信。自動的にハードウェアでクリアされる。
0 = アクノリッジのシーケンスがアイドル。
注 : I2C モジュールがアイドルモードではない場合、このビットがセットされなかったり (スプーリングなし)、SSPBUF がライトされないことがあります (または SSPBUF へのライトがディセーブルされます)。

bit 3: **RCEN**: 受信イネーブルビット (I2C マスタモードのみ)
1 = I2C の受信モードをイネーブル。
0 = 受信がアイドル。
注 : I2C モジュールがアイドルモードではない場合、このビットがセットされなかったり (スプーリングなし)、SSPBUF がライトされないことがあります (または SSPBUF へのライトがディセーブルされます)。

bit 2: **PEN**: ストップコンディションイネーブルビット (I2C マスタモードのみ)
SCK が制御を解除
1 = SDA と SCL ピンでストップコンディションを開始。自動的にハードウェアでクリアされる。
0 = ストップコンディションがアイドル。
注 : I2C モジュールがアイドルモードではない場合、このビットがセットされなかったり (スプーリングなし)、SSPBUF がライトされないことがあります (または SSPBUF へのライトがディセーブルされます)。

bit 1: **RSEN**: 再スタートコンディションイネーブルビット (I2C マスタモードのみ)
1 = SDA と SCL ピンで再スタートコンディションを開始。自動的にハードウェアでクリアされる。
0 = 再スタートコンディションがアイドル。
注 : I2C モジュールがアイドルモードではない場合、このビットがセットされなかったり (スプーリングなし)、SSPBUF がライトされないことがあります (または SSPBUF へのライトがディセーブルされます)。

bit 0: **SEN**: スタートコンディションイネーブルビット (I2C マスタモードのみ)
1 = SDA と SCL ピンでスタートコンディションを開始。自動的にハードウェアでクリアされる。
0 = スタートコンディションがアイドル。
注 : I2C モジュールがアイドルモードではない場合、このビットがセットされなかったり (スプーリングなし)、SSPBUF がライトされないことがあります (または SSPBUF へのライトがディセーブルされます)。

15.1 SPI モード

SPI モードにより、8bit のデータを同期させて送信と受信を同時にすることができます。SPI の 4 つのモードすべてがサポートされています。通信を行うためには、一般的に次の 3 本のピンが使われます。

- **シリアルデータ出力 (SDO)**
- **シリアルデータ入力 (SDI)**
- **シリアルクロック (SCK)**

スレーブモード動作のときは、さらに 4 番目のピンが使われることもあります。

- **スレーブ選択 (SS)**

SPI の初期化時には、いくつかのオプションを設定する必要があります。それは SSPCON1 レジスタ (SSPCON1<5:0>) と SSPSTAT<7:6> の制御ビットを適切にプログラムすることにより行われます。これらの制御ビットにより、次のような設定が可能です。

- **マスタモード (SCK はクロック出力)**
- **スレーブモード (SCK はクロック入力)**
- **クロックの極性 (SCK のアイドル状態)**
- **データ入力サンプル位相 (データ出力時間の中間または終わり)**
- **クロックエッジ (出力データを SCK の立ち上がり / 立ち下がりエッジのどちらにするか)**
- **クロックレート (マスタモードのみ)**
- **スレーブ選択モード (スレーブモードのみ)**

SSP は送信 / 受信シフトレジスタ (SSPSR) とバッファレジスタ (SSPBUF) で構成されています。SSPSR はデバイスの入力と出力のデータをシフトします。MSb を先にシフトします。SSPBUF は受信されたデータが用意できるまで、SSPSR に書き込まれたデータを保持します。8bit のデータが受信されてしまうと、そのバイトは SSPBUF レジスタに移されます。それからバッファフル検出ビット BF(SSPSTAT<0>) と割込みフラグビット SSPIF(PIR2<7>) がセットされます。この受信されたデータ (SSPBUF) をダブルバッファすることにより、受信されたばかりのデータを読み出す前に次のバイトの受信が始まるのを可能にします。データの送信 / 受信中の SSPBUF レジスタへのライトは無視され、書き込み衝突検出ビット WCOL(SSPCON1<7>) がセットされます。ソフトウェアで WCOL ビットをクリアする必要があります。それによって SSPBUF レジスタへの次のライトをうまく完了するかどうかを決定できます。

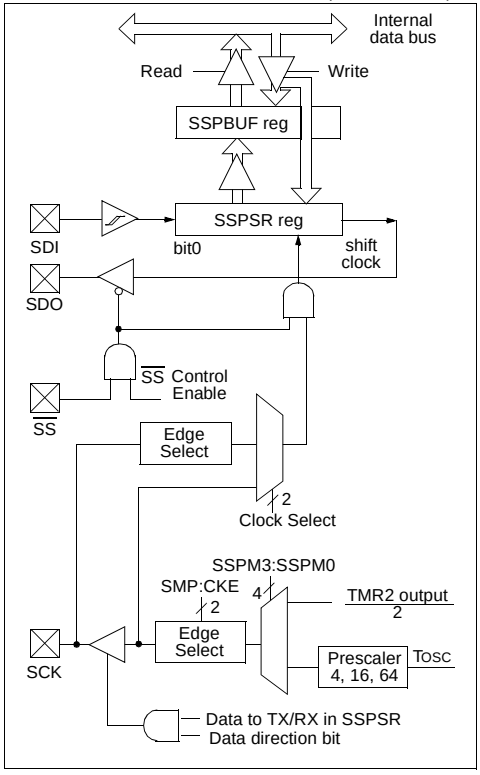
アプリケーションソフトウェアが有効なデータを受信しようとしている時、転送するデータの次のバイトを SSPBUF に書き込む前にその SSPBUF を読み込む必要があります。SSPBUF が受信されたデータにロードされている時 (送信完了)、バッファフルビット BF(SSPSTAT<0>) がそのことを示します。SSPBUF が読み込まれた時、ビット BF がクリアされます。SPI が送信のみの場合、このデータは不適切なことがあります。一般的には、送信 / 受信が完了した時を決めるのに SSP 割込みが使われます。SSPBUF はリードまたはライトされる必要があります。その割込み方法を使わない場合は、ソフトウェアポーリングによって、書き込み衝突が起こらないよう確実にすることができます。例 15-1 にデータ送信のための SSPBUF(SSPSR) のロード方法を示します。網掛けの命令は受信されたデータが有効な場合にだけ必要です。

例 15-1: SSPBUF(SSPSR) レジスタのロード

```
      MOVLB 6               ; Bank 6
LOOP BTFSS SSPSTAT, BF      ; Has data been
                            ;   received
                            ;   (transmit
                            ;   complete)?
     GOTO  LOOP             ; No
     MOVPF SSPBUF, RXDATA   ; Save in user RAM
     MOVFP TXDATA, SSPBUF   ; New data to xmit
```

SPIモード時のSSPモジュールのブロック図(図15-7)は、SSPSR　が直接リードまたはライトができず、SSPBUF レジスタをアドレス指定することによりアクセスできるのみであることを示します。さらに、SSPステータスレジスタ(SSPSTAT)が種々のステータス条件を示します。

図 15-7:　　SSP ブロック図 (SPI モード)



シリアルポートをイネーブルにするためには、SSPイネーブルビット SSPEN(SSPCON1<5>)をセットしなければなりません。SPI モードをリセットまたは再設定するためには、ビット SSPEN をクリアし、SSPCON レジスタを再初期化し、さらにビット SSPEN をセットします。これにより、SDI、SDO、SCK、$\overline{SS}$ ピンがシリアルポートピンとして設定されます。シリアルポート機能として作用するためのピンとして、それぞれのデータ方向指示ビット (DDR レジスタ内) を適切にプログラムする必要があります。すなわち、

- **SDI は自動的に SPI モジュールにより制御**
- **SDO は DDRB<7> をクリア**
- **SCK(マスタモード) は DDRB<6> をクリア**
- **SCK(スレーブモード) は DDRB<6> をセット**
- **$\overline{SS}$ は PORTA<2> をセット**

注意 :　スレーブ選択を動作するためには、入力として $\overline{SS}$ ピンを設定する必要があります。これは ' 1 ' を PORTA<2> にライトすることにより動作します。

必要でないシリアルポート機能は、対応するデータ方向(DDR)レジスタに反対の値をプログラムすることにより無視されます。例えば、データを (ディスプレィドライブバへ)送信しているだけのマスタモードでは、((DDR に) ' 0 ' をライトすることにより、SDI と $\overline{SS}$ の両方が汎用オープンドレイン出力として使われます。

図 15-9 に 2 個のマイクロコントローラ間の一般的な接続を示します。マスタコントローラ (プロセッサ 1) は SCK 信号を送ってデータ転送を開始します。データはプログラムされたクロックエッジで両方のシフトレジスタからシフトアウトされ、そのクロックの反対のエッジでラッチされます。両方のプロセッサは、同じクロック極性 (CKP) にプログラムされる必要があり、そうすれば両方のコントローラが同時にデータの送信と受信を行ないます。そのデータが有効かどうか (ダミーデータかどうか) はアプリケーションソフトウェアによります。これによりデータ送信に関して、次の 3 つの場合が考えられます。

- **マスタがデータを送る－スレーブがダミーデータを送る**
- **マスタがデータを送る－スレーブがデータを送る**
- **マスタがダミーデータを送る－スレーブがデータを送る**

15.1.1 マスタモード

マスタが SCK を制御するので、いつでもデータ転送を開始できます。マスタはスレーブ (プロセッサ 2、図 15-9) がいつデータを出すかをソフトウェアプロトコルによって決めます。

マスタモードでは、SSPBUF レジスタに書き込まれると同時にデータが送信 / 受信されます。SPI が受信だけを行なう場合は、SCK 出力をディセーブルすることができます (入力としてプログラムする)。SSPSR レジスタはプログラムされたクロックレートで SDI ピンに与えられた信号のシフトインを継続します。各バイトが受信されると、通常の受信されたバイトのように SSPBUF レジスタにロードされます (割込みとステータスビットがセット)。これは " ライン状態モニタ " モードとしての受信アプリケーションに便利です。

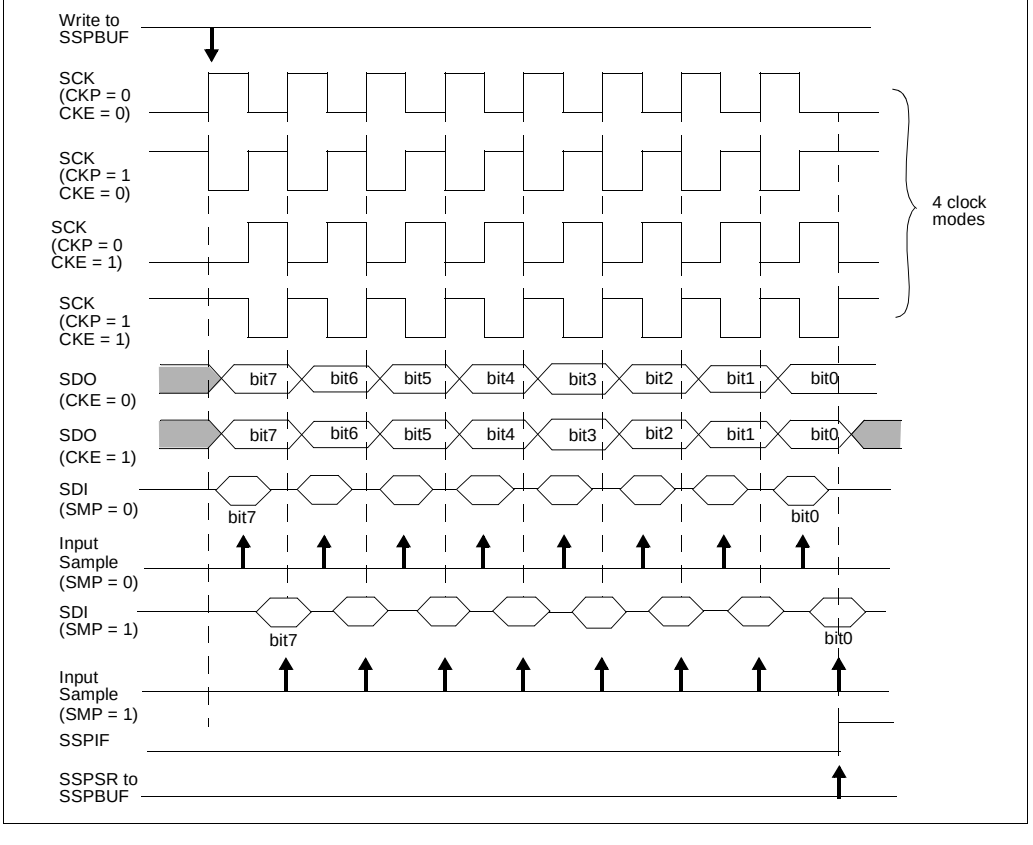
クロック極性はビット CKP(SSPCON1<4>) を適切にプログラムすることにより選択されます。これにより、図 15-8、15-11、15-12 に示すように、SPI 通信用の波形が与えられ、そこでは MSB が最初に送信されます。マスタモードでは、SPI クロックレート (ビットレート) は次のうちいずれか 1 つにプログラム可能です。

- **Fosc/4 (または Tcy)**
- **Fosc/16 (または 4・Tcy)**
- **Fosc/64 (または 16・Tcy)**
- **タイマ 2 出力 /2**

これにより、最大ビットクロック周波数 (33MHz にて)8.25MHz が可能になります。

図 15-8 に、マスタモードの波形を示します。CKE=1 の時、SDO データは SCK のクロックエッジの手前で有効です。入力サンプルの変化は SMP ビットの状態を基本にして示してあります。SSPBUF が受信データにロードされる時の時間が示されています。

図 15-8: SPI モードのタイミング (マスタモード)



15.1.2 スレーブモード

スレーブモードでは、SCK に現れた外部クロックパルスによりデータの送信と受信が行われます。最後のビットがラッチされると、割込みフラグビット SSPIF(PIR2<7>) がセットされます。一方スレーブモードでは、外部クロックが SCK ピンの外部クロックソースにより与えられます。この外部クロックは、電気的仕様に規定されている最小のハイタイムとロータイムを守る必要があります。

またスリープモードでは、スレーブはデータを送信 / 受信でき、スリープからデバイスをウェークできます。

15.1.3 スレーブ選択同期化

$\overline{SS}$ ピンによって同期スレーブモードが可能です。SPI はイネーブルされた $\overline{SS}$ ピン制御 (SSPCON1<3:0>=04h) でスレーブモードにする必要があります。そのピンは入力としての機能の $\overline{SS}$ ピンとしてローに駆動してはいけません。RA2 データラッチはハイにしなければなりません。$\overline{SS}$ ピンがローの時、送信と受信がイネーブルになり、SDO ピンが駆動されます。$\overline{SS}$ ピンがハイになると、SDO ピンが送信中のバイトの途中であっても駆動せず、フローティング出力となります。アプリケーションによっては、外部のプルアップ / プルダウン抵抗が望ましい場合があります。

注意 :	**SPI がイネーブルされた $\overline{SS}$ ピン制御 (SSPCON <3:0>=0100) でスレーブモードになると、$\overline{SS}$ ピンが V_{DD} にセットされる場合 SPI モジュールはリセットされます。**
注意 :	**SPI が CKE=' 1 ' のスレーブモードで使用される場合、$\overline{SS}$ ピン制御をイネーブルする必要があります。**

2 ワイヤ通信をエミュレートするには、SDO ピンを SDI ピンに接続することができます。SPI がレシーバとしての動作を必要とする時はSDOピンを入力として設定します。これにより、SDO からの送信がディセーブルされます。SDI はバス衝突を起こすことがないため、常に入力 (SDI 機能) のままにされます。

図 15-11 では、$\overline{SS}$ ピンが送信 / 受信を終了します。SSPIFビットはSCKの最後のエッジ後セットされます。図 15-12 では、$\overline{SS}$ ピンによりデータの最初のビットが出力になります。SSPIF ビットは最後の SCK エッジ後にセットされます。

図 15-9: SPI マスタ / スレーブ接続

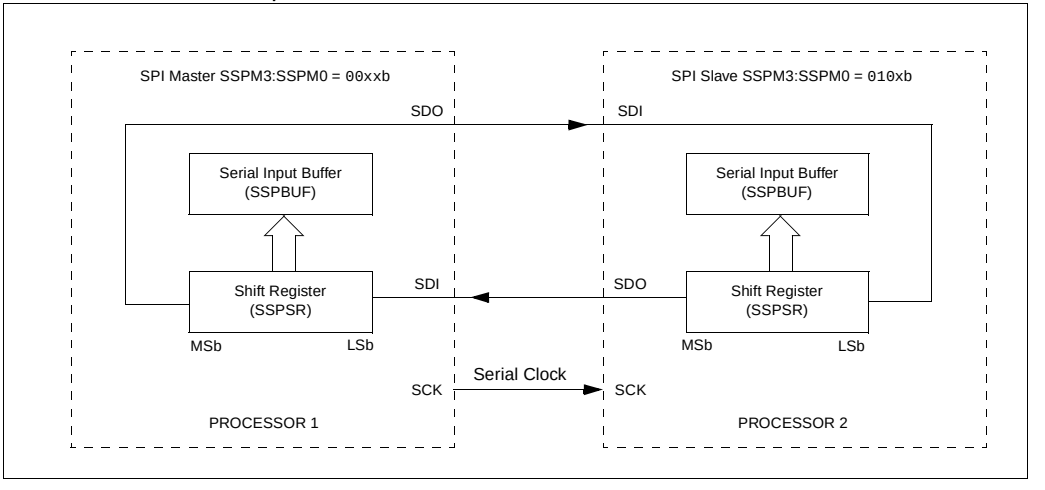


図 15-10: スレーブ同期化のタイミング

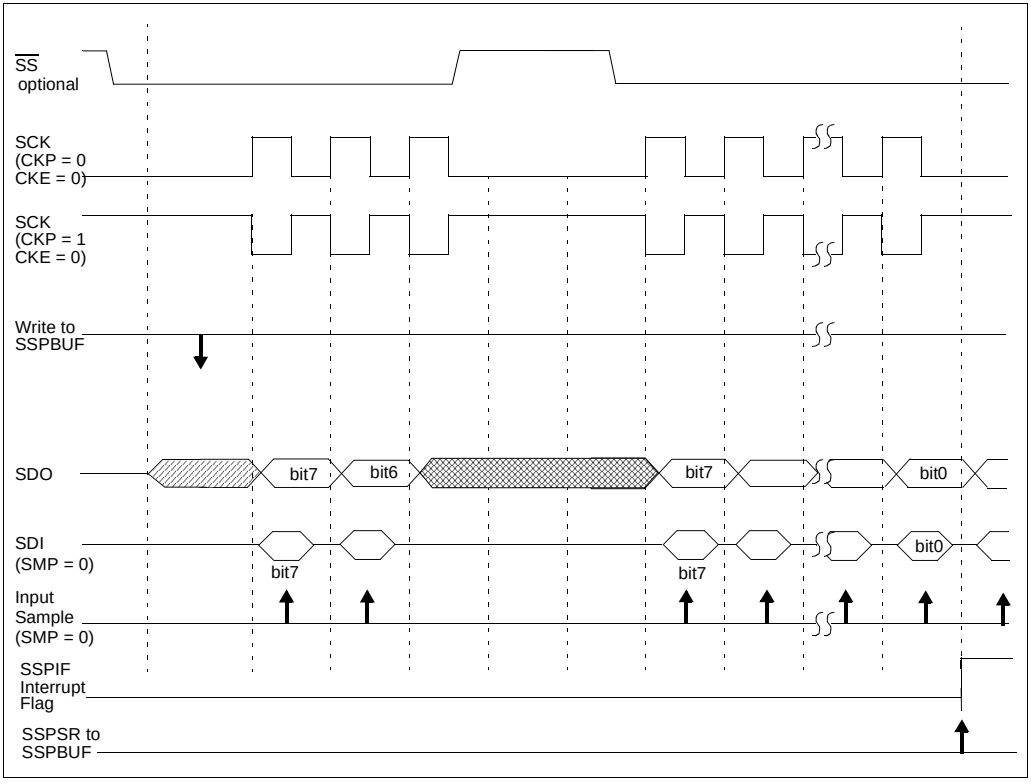


図 15-11: SPI モードのタイミング (CKE=0 でのスレーブモード)

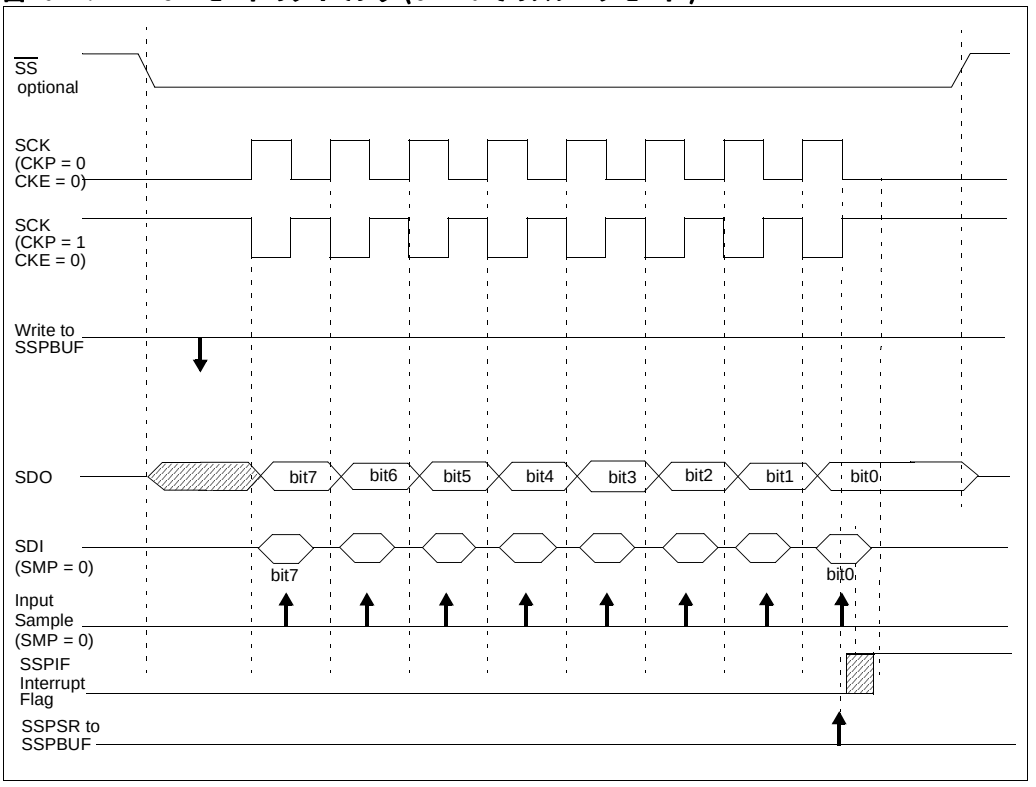


図 15-12: SPI モードのタイミング (CKE=1 でのスレーブモード)

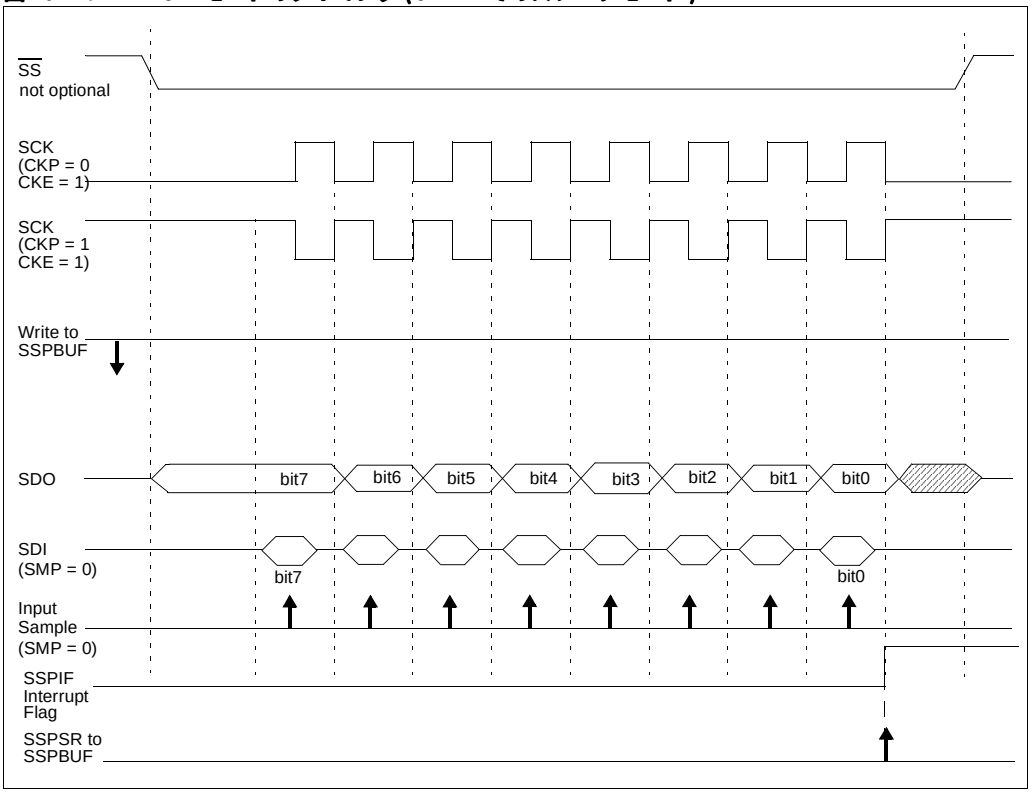


表 15-1: SPI 動作に関連するレジスタ

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR, BOR での値	他のリセットでの値（注 1）
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
10h, Bank 4	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	000- 0010	000- 0010
11h, Bank 4	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	000- 0000	000- 0000
14h, Bank 6	SSPBUF	同期シリアルポート受信バッファ / 送信レジスタ								xxxx xxxx	uuuu uuuu
11h, Bank 6	SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
13h, Bank 6	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	0000 0000

凡例 : x= 不定、u= 不変、— = 未使用、‘ 0 ’としてリード。網掛け部分は SPI モードの SSP では使われません。

注 1: 他の（パワーアップ以外）リセットは、$\overline{MCLR}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

15.2 SSP I²C の動作

I²C モードの SSP モジュールはすべてのマスタとスレーブ機能 (ジェネラルコールサポートを含む) を完全に満たし、フリーバス (マルチマスタ機能) を決定するためにハードウェアのスタートビットとストップビットでの割込みを備えています。SSP モジュールは 7bit と 10bit の標準モード仕様はもとより、規格を満たしています。付録 E に I²C バス設定の概要があります。

図 15-13: SSP ブロック図 (I²C モード)

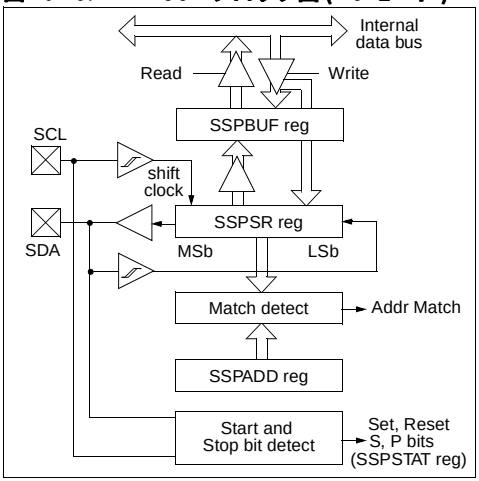
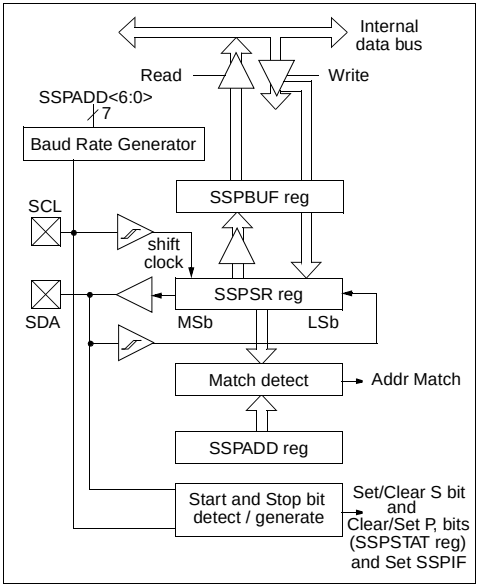


図 15-14: I²C マスタモードのブロック図



2 本のピンがデータ転送用に使われます。それらはクロックである SCL ピンと、データである SDA ピンです。I²C モードをイネーブルすると、ポート A 上にあるピンは自動的に設定されます。SSP イネーブルビット SSPEN(SSPCON1<5>) をセットすることにより、SSP モジュール機能をイネーブルにします。

SSP モジュールには I²C 動作のために、次のような 6 種類のレジスタがあります。

- **SSP 制御レジスタ 1(SSPCON1)**
- **SSP 制御レジスタ 2(SSPCON2)**
- **SSP ステータスレジスタ (SSPSTAT)**
- **シリアル受信 / 送信バッファ (SSPBUF)**
- **SSP シフトレジスタ (SSPSR) 直接アクセス不可**
- **SSP アドレスレジスタ (SSPADD)**

SSPCON1 レジスタにより I²C 動作の制御が可能です。4 種類のモード選択ビット (SSPCON1<3:0>) により次の I²C モードから 1 つを選択することができます。

- **I²C スレーブモード (7-bit アドレス)**
- **I²C スレーブモード (10-bit アドレス)**
- **I²C マスターモード、クロック = OSC/4 (SSPADD +1)**

SSPEN ビットのセットを持った I²C モードの選択により、SCL と SDA ピンがオープンドレインとなります。これらのピンが PORTA にあり、そのために入力にプログラムする必要はありません。

SSPSTAT レジスタはデータ転送の状態を示します。この情報には START と STOP ビットの検出が含まれ、受信されたバイトがデータかアドレスか、次のバイトが 10bit のアドレスの終わりかどうか、そしてこれがリードまたはライトのデータ転送かどうかを表わします。

SSPBUF は転送データをライトまたはリードするレジスタです。SSPSR レジスタはそのデータをデバイスに対してシフトインまたはシフトアウトします。受信動作では、SSPBUF と SSPSR はダブルバッファされたレシーバを作ります。これにより、受信したデータの最後のバイトを読み込む前に、次のバイトの受信開始が可能になります。完全なバイトが受信された時、SSPBUF レジスタにそれを転送し、フラグビット SSPIF がセットされます。SSPBUF レジスタが読み込まれる前に他の完全なバイトが受信されると、レシーバオーバーフローが起こり、ビット SSPOV(SSPCON1<6>) がセットされ、SSPSR のバイトは失われます。

SSPADD レジスタはスレーブアドレスを保持します。10bit のモードでは、アドレスの上位バイトを書き込む必要があります (1111 0 A9 A8 0)。上位バイトのアドレス一致に続き、アドレスの下位バイトをロードする必要があります (A7:A0)。

15.2.1 スレーブモード

スレーブモードでは、SCL と SDA のピンを入力として設定する必要があります。SSP モジュールは入力状態を無視して、必要な時にデータを出力します(スレーブトランスミッタ)。

アドレスが一致するか、一致アドレスからのデータ転送が受信された時、ハードウエアが自動的に応答($\overline{ACK}$) パルスを発生し、SSPSR レジスタが受信した値を SSPBUF レジスタにロードします。

SSP モジュールがこの $\overline{ACK}$ パルスを与えなくなる条件があります。それは次のうちのどちらかまたは両方です。

a) 転送が受信される前に、バッファフルビット BF(SSPSTAT<0>) がセットされていた。

b) 転送が受信される前に、オーバーフロービット SSPOV(SSPCON1<6>) がセットされていた。

この場合、SSPSR レジスタの値は SSPBUF にロードされませんが、ビット SSPIF(PIR2<7>) がセットされます。表 15-2 にデータ転送バイトが受信され、BF と SSPOV ビットの状態が与えられた時に起こる状況を示します。網かけの部分は、ソフトウェアでオーバーフロー条件を正しくクリアしなかった場合の条件を示します。フラグビット BF は、ビット SSPOV がソフトウェア中でクリアされている間に SSPBUF レジスタをリードすることでクリアされます。

SCL クロック入力は完全な動作をするために、最小のハイとローの時間規定を守ることが必要です。SSP モジュールの必要規格と同様に、I^2C の規格のハイタイムとロータイムをタイミングパラメータ #100 と #101 に示します。

15.2.1.1 アドレス指定

SSP モジュールがイネーブルされると、SSP は START コンディションが起きるのを待ちます。START コンディションの後、8 bit が SSPSR レジスタにシフトされます。入力されたすべてのビットはクロック (SCL) ラインの立ち上がりエッジでサンプリングされます。SSPSR<7:1> レジスタの値は SSPADD レジスタの値と比較されます。アドレスは 8 番目のクロック (SCL) パルスの立ち下がりエッジで比較されます。もしアドレスが一致し、BF と SSPOV ビットがクリアの場合、次のことが起こります。

a) SSPSR レジスタの値が SSPBUF レジスタにロードされる。

b) バッファフルビット BF がセットされる。

c) $\overline{ACK}$ パルスを発生。

d) SSP 割込みフラグビット SSPIF(PIR2<7>) が 9 番目の SCL パルスの立ち下がりエッジでセットされる (イネーブルの時、割込みが発生)。

10bit のアドレスモードでは、2 アドレスバイトがスレーブに受信される必要があります。最初のアドレスバイトの上位 5 bit(MSbs) は、これが 10bit アドレスかどうかを示しています。ビット R/$\overline{W}$(SSPSTAT<2>) はライトを示している必要があり、それによりスレーブデバイスは 2 番目のアドレスバイトを受信します。10bit アドレスでは最初のバイトは ' 1111 0 A9 A8 0 ' で、ただし A9 と A8 はアドレスの上位 2 bit です。10bit アドレスで起こるイベントの順番は次のようになり、ステップ 7 － 9 はスレーブトランスミッタ用です。

1. **アドレスの最初の (上位) バイトを受信 (ビット SSPIF、BF とビット UA(SSPSTAT<1>) をセット)。**
2. **アドレスの 2 番目の (下位) バイトで SSPADD レジスタをアップデート (ビット UA をクリアし、SCL ラインをリリース（開放）)。**
3. **SSPBUF レジスタをリード (ビット BF をクリア) し、フラグビット SSPIF をクリア。**
4. **アドレスの 2 番目の (下位) バイトを受信 (ビット SSPIF、BF、UA をセット)。**
5. **アドレスの最初の (上位) バイトで、SSPADD レジスタをアップデート。一致で SCL ラインがリリースされ、それによりビット UA がクリアされる。**
6. **SSPBUF レジスタをリード (ビット BF をクリア) し、フラグビット SSPIF をクリア。**
7. **繰り返し START 条件を受信。**
8. **アドレスの最初の (上位) バイトを受信 (ビット SSPIF、BF をセット)。**
9. **SSPBUF レジスタをリード (ビット BF をクリア) し、フラグビット SSPIF をクリア**

注意 : 10bit モードで RESTART 条件 (ステップ 7) の後に続けるには、最初の 7bit アドレスと一致させる必要があるだけです。アドレスの後半では SSPADD をアップデートしません。

表 15-2: データ転送受信時のバイト動作

データ転送受信時のステータスビット				
BF	SSPOV	SSPSR → SSPBUF	$\overline{ACK}$ パルスの生成	ビット SSPIF のセット (イネーブル時の SSP 割込み)
0	0	Yes	Yes	Yes
1	0	No	No	Yes
1	1	No	No	Yes
0	1	No	No	Yes

15.2.1.2 スレーブ受信

アドレスバイトの R/$\overline{W}$ ビットがクリアで、アドレスの一致が起こった時、SSPSTAT レジスタの R/$\overline{W}$ ビットはクリアされます。受信されたアドレスは SSPBUF レジスタにロードされます。

アドレスバイトのオーバーフロー条件があると、応答 ($\overline{ACK}$) パルスは与えられません。オーバーフロー条件では、ビット BF(SSPSTAT<0>) がセットされているか、ビット SSPOV(SSPCON1<6>) がセットされているかどちらかで定義されます。

SSP 割込みは各データの転送毎に発生します。フラグビットSSPIF(PIR2<7>)はソフトウェアでクリアされる必要があり、SSPSTAT レジスタはバイトの状態を確かめるために使われます。

注意 : SSPOV ビット =1 で、BF フラグ =0 の場合、SSPBUF はロードされます。SSPBUF のリードが実行された場合でも、次の受信が起こる前に SSPOV ビットの状態をクリアしません。$\overline{ACK}$ は送り出されず、SSPBUF がアップデートされます。

15.2.1.3 スレーブ送信

入力されたアドレスバイトの R/$\overline{W}$ ビットがセットされ、アドレスの一致が起こると、SSPSTAT レジスタの R/$\overline{W}$ ビットがセットされます。受信されたアドレスは SSPBUF レジスタにロードされます。$\overline{ACK}$ パルスは 9 番目のビットで送り出され、SCL ピンがローに保持されます。送信データを SSPBUF レジスタにロードする必要があり、そこから SSPSR レジスタをロードします。それからビット CKP(SSPCON1<4>) をセットすることにより、SCL ピンをイネーブルにする必要があります。マスタは別のクロックパルスを示すより前に SCL ピンをモニタしなければなりません。スレーブデバイスはクロックを伸ばすことによりマスタを待たせることがあります。8 個のデータビットは SCL 入力の立ち下がりエッジでシフトアウトされます。これにより SCL がハイタイムの間、SDA 信号の有効性を確実にします (図 15-16 参照)。

図 15-15: 受信時の I²C 波形 (7bit アドレス)

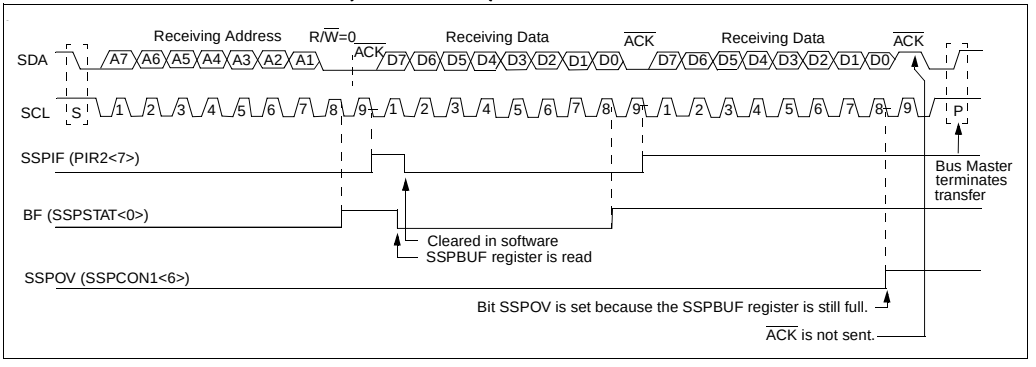


図 15-16: 送信時の I²C 波形 (7bit アドレス)

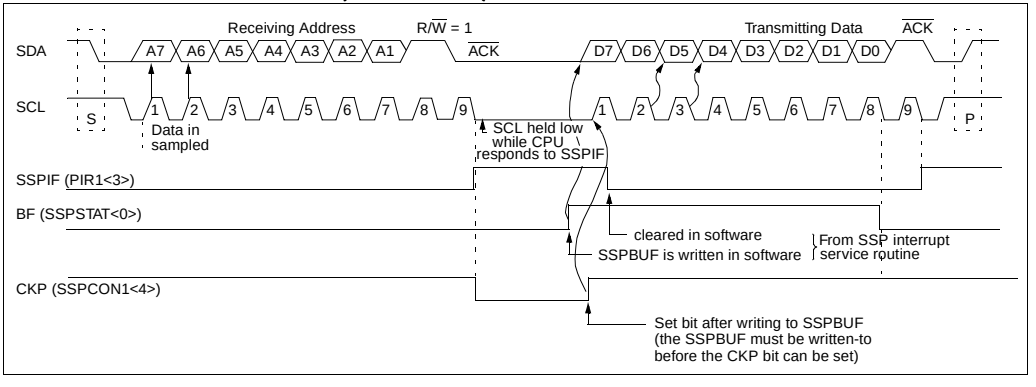


図 15-17: I²C のスレーブトランスミッタ (10bit アドレス)

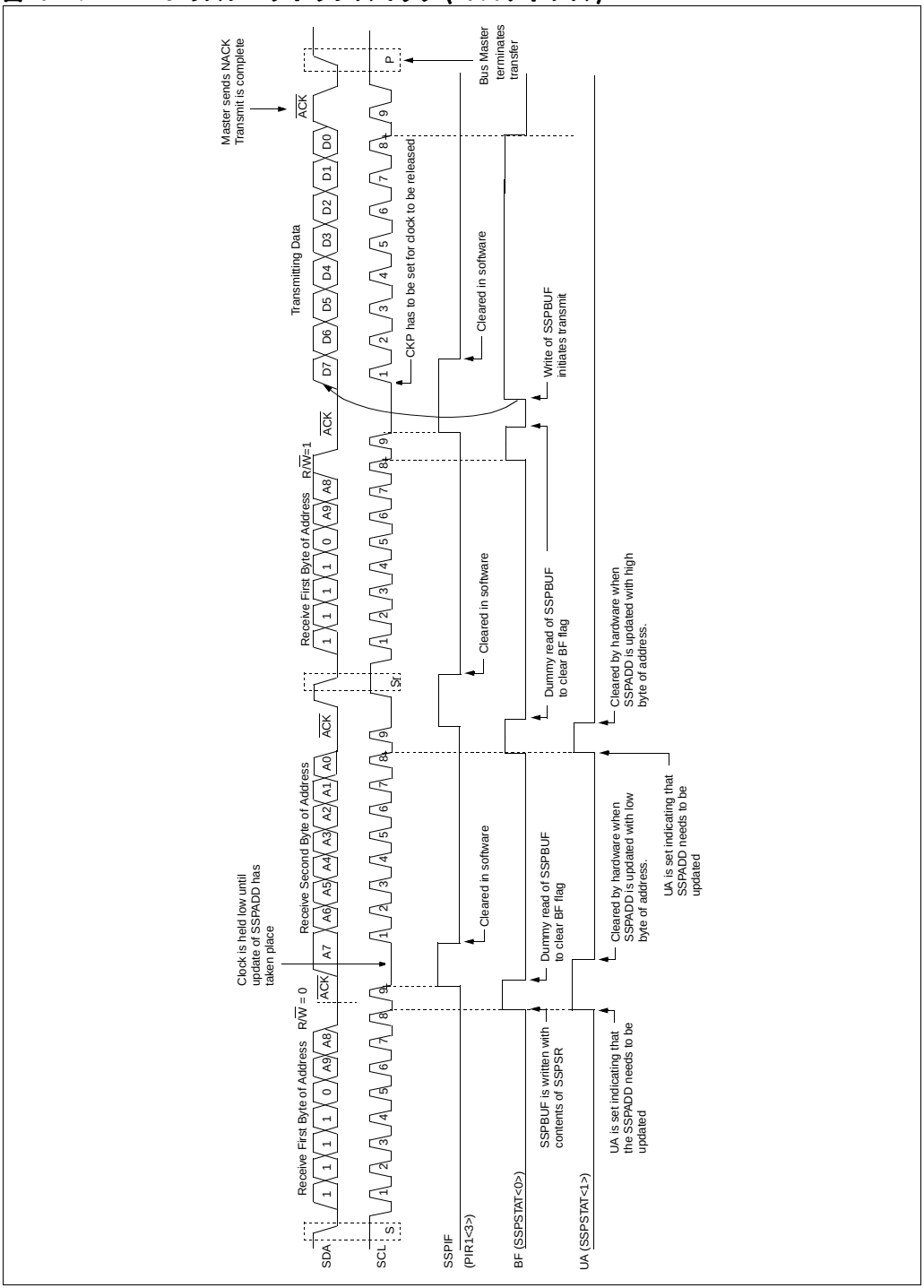
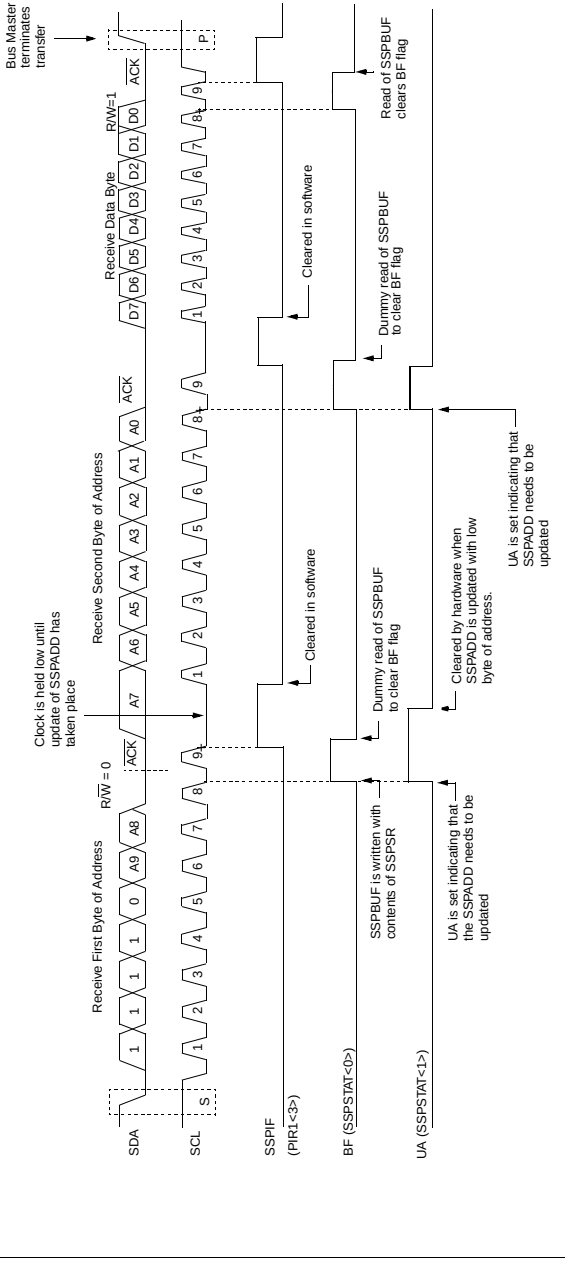


図 15-18: I²C のスレーブレシーバ (10bit アドレス)



SSP割込みは各データの転送バイト毎に発生します。フラグビット SSPIF はソフトウェアでクリアされる必要があり、SSPSTAT レジスタはバイトの状態を確かめるために使われます。フラグビット SSPIF は 9 番目のクロックパルスの立ち下がりエッジでセットされます。

スレーブトランスミッタとして、マスタレシーバからの $\overline{ACK}$ パルスは 9 番目の SCL 入力パルスの立ち上がりエッジでラッチされます。SDAラインがハイ ($\overline{ACK}$ なし) の時、データ転送が終わります。$\overline{ACK}$ がスレーブによりラッチされると、スレーブ論理がリセットされ、それからスレーブが他の START ビットが起こるのを監視します。SDA ラインがローの時 ($\overline{ACK}$)、送信データを SSPBUF レジスタにロードする必要があり、SSPBUF は SSPSR レジスタもロードします。そして SCL ピンを、ビット CKP をセットすることによりイネーブルにする必要があります。

15.2.2 ジェネラルコールアドレスサポート

I^2C バスにアドレス指定する手順は、デバイスがマスタによりアドレス指定されたスレーブであることを、START 条件後の最初のバイトが常に決めるということです。例外として、すべてのデバイスをアドレス指定できるジェネラルコールアドレスがあります。このアドレスが使われると、すべてのデバイスは理論上では応答に答える必要があります。

ジェネラルコールアドレスは、I^2C プロトコルにより特定の目的のための予備である 8 つのアドレスの 1 つです。R/$\overline{W}$=0 で、すべての 0 から成ります。

ジェネラルコールアドレスは、ジェネラルコールイネーブルビット (GCEN) がイネーブルされると認められます (SSPCON2<7>)。スタートビット検出に続き、8bit が SSPSR にシフトされ、アドレスは SSPADD と比較され、またハードウェアにあるジェネラルコールアドレスと比較されます。

ジェネラルコールアドレスが一致すると、SSPSR は SSPBUF に転送され、BF フラグがセットされ (8 番目のビット)、9 番目のビット ($\overline{ACK}$ ビット) の立ち下がりエッジでは、SSPIF 割込みがセットされます。

割込みが起こると、アドレスがデバイスに特定のものかジェネラルコールアドレスかを決めるために、割込み要因は SSPBUF の内容をリードすることによりチェックします。

10bit モードでは、SSPADD は一致するためにはアドレスの後半でアップデートすることが要求され、UA ビットがセットされます (SSPSTAT<1>)。スレーブが 10bit のアドレスモードで設定されている時に、ジェネラルコールアドレスが GCEN=1 の時にサンプリングされると、アドレスの後半は必要ではなく、UA ビットはセットされず、スレーブは応答後データを受信し始めます (図 15-19)。

図 15-19: ジェネラルコールアドレスのシーケンス (7bit または 10bit モード)

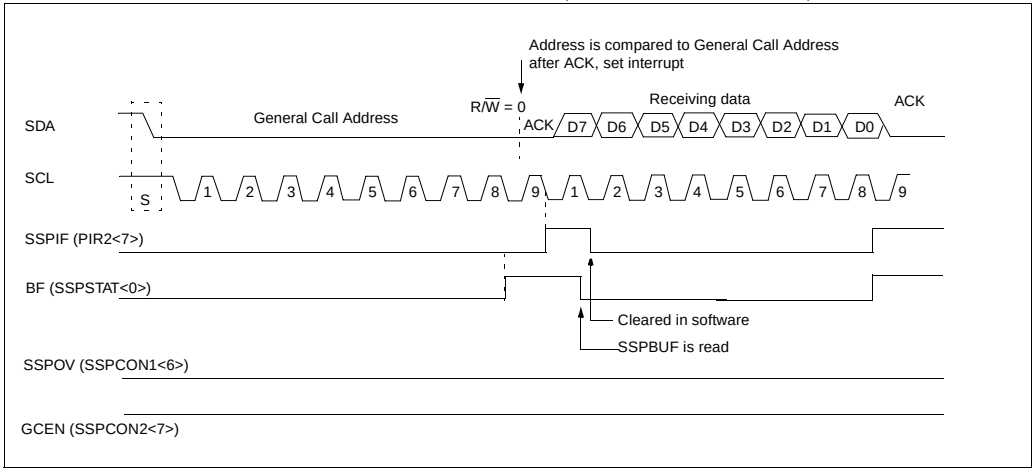


表 15-3: I²C 動作に関連するレジスタ

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR, BOR での値	他のリセットでの値（注 1）
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
10h, Bank 4	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	00-- 0000	00-- 0000
11h, Bank 4	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	00-- 0000	00-- 0000
10h. Bank 6	**SSPADD**	同期シリアルポート (I²C モード）アドレスレジスタ								0000 0000	0000 0000
14h, Bank 6	**SSPBUF**	同期シリアルポート受信バッファ / 送信レジスタ								xxxx xxxx	uuuu uuuu
11h, Bank 6	SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
12h, Bank 6	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	0000 0000
13h, Bank 6	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	0000 0000

凡例 : x= 不定、u= 不変、— = 未使用、‘0’としてリード。網掛け部分は I²C モードの SSP では使われません。

注 1: 他の（パワーアップ以外）リセットは、$\overline{MCLR}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

15.2.3 マスタモード

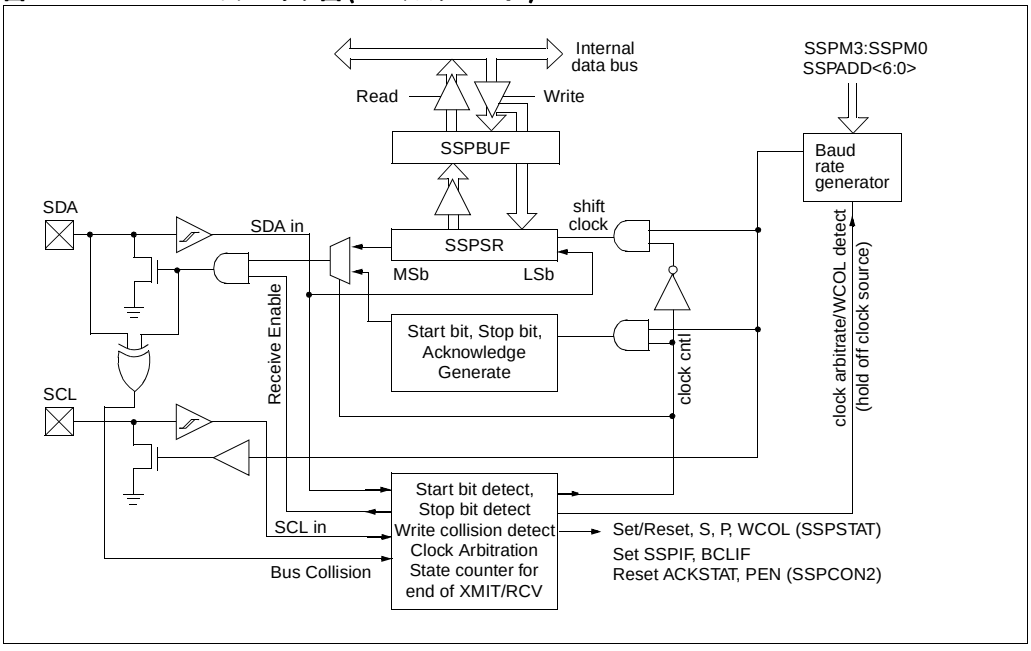
マスタモードの動作はSTARTとSTOP条件の検出時の割込み発生によりサポートされています。STOP(P)と START(S) のビットはリセットされた場合か、SSPモジュールがディセーブルになるとクリアされます。P ビットがセットされるか、バスがアイドル状態で S と P ビットの両方がクリアされる時、I^2C バスの制御が行われます。

マスタモードでは、SSP ハードウェアにより SCL と SDA のラインを操作します。

次のイベントで SSP 割込みフラグビット SSPIF がセットされます (イネーブルの時 SSP 割込み)。

- **START コンディション**
- **STOP コンディション**
- **データ転送バイトの送信 / 受信**

図 15-20: SSP のブロック図 (I^2C マスタモード)



15.2.4 マルチマスタモード

マルチマスタモードでは、START と STOP の条件検出時の割込み発生により、バスがフリーになった時を判定できます。STOP(P) と START(S) のビットはリセットされるか、SSP モジュールがディセーブルになるとクリアされます。ビット P(SSPSTAT<4>) がセットされるか、S ビットと P ビットの両方がクリアでバスがアイドル状態の時、I²C バスの制御が行われます。バスがビジーの時、SSP 割込みをイネーブルにすることで、STOP 条件が起こった時割込みを発生します。

マルチマスタ動作では、信号レベルが必要な出力レベルかどうか見るために、SDA ラインを監視する必要があります。このチェックは BCLIF ビットにある結果と共にハードウェアで実行されます。

アービトレーションが失われることがある状態は下記の通りです。

- **アドレス転送**
- **データ転送**
- **スタートコンディション**
- **再スタートコンディション**
- **応答コンディション**

15.2.5 I²C マスタモードサポート

マスタモードは SSPCON1 の中の適切な SSPM ビットをセット / クリアすることにより、また SSPEN ビットをセットすることによりイネーブルされます。マスタモードがイネーブルになると、6 つのオプションが利用できます。

- **SDA と SCL のスタートコンディションを示す。**
- **SDA と SCL の再スタートコンディションを示す。**
- **SSPBUF レジスタに書き込むことにより、データ / アドレスの送信を開始する。**
- **SDA と SCL ライン上にストップコンディションを発生させる。**
- **データを受信するために I²C ポートを設定する。**
- **データの受信されたバイトの最後に応答条件を発生する。**

注意 : I²C マスタモードに設定された時の SSP モジュールはイベントの待ち行列ができません。例えば、START コンディションを始め、START コンディションが終了する前に送信を始めるためにただちに SSPBUF レジスタに書き込むことはできません。この場合には、SSPBUF が書き込まれず、WCOL ビットがセットされ、SSPBUF へのライトが起こらなかったと示します。

15.2.5.1 I²C マスタモード動作

マスタデバイスはすべてのシリアルクロック位相と、START と STOP 条件を発生します。転送は STOP 条件または連続の START 条件で終わります。連続の START 条件は次のシリアル転送の開始でもあるので、I²C バスはリリースされません。

マスタ送信モードでは、シリアルデータは SDA を通して出力され、SCL はシリアルクロックを出力します。送信された最初のバイトは、受信しているデバイス (7bit) のスレーブアドレスとデータ方向指示ビットを含みます。この場合では、データ方向指示ビット (R/$\overline{W}$) は論理 ‘ 0 ’ です。シリアルデータは 1 度に 8bit を送信します。各バイトが送信された後、応答ビットが受信されます。START と STOP 条件はシリアル転送の最初と最後を示すために出力されます。

マスタ受信モードでは、送信された最初のバイトが送信しているデバイス (7bit) のスレーブアドレスとデータ方向指示ビットを含みます。この場合では、データ方向ビット (R/$\overline{W}$) は論理 ‘ 1 ’ です。このように送信された最初のバイトは、受信ビットを示すために 7bit のスレーブアドレスの後に ‘ 1 ’ が続きます。シリアルデータは SDA により受信され、SCL はシリアルクロックを出力します。シリアルデータは 1 度に 8bit を受信します。各バイトが受信された後、応答ビットが送信されます。START と STOP 条件は送信の最初と最後を示します。

SPI モード動作に使われたボーレートジェネレータは、100kHz、400kHz、1MHz の I²C 動作のいずれかで SCL クロック周波数をセットするために使用されます。 ボーレートジェネレータの再ロードされた値は、SSPADD レジスタの下位 7bit に含まれています。ボーレートジェネレータは自動的に SSPBUF へのライトで計算し始めます。与えられた動作が終了すると (最後のデータビットの送信の後に ACK が続く)、内部クロックは自動的に計算を止め、SCL ピンはその最後の状態で残ります。

標準の送信のシーケンスは次のように行ないます。

1. **SSPCON2 の START イネーブルビット (SEN) をセットすることによりスタートコンディションを発生させる。**
2. **SSPIF をセット。モジュールは他の動作が起こる前に必要とされるスタート時間を待つ。**
3. **送信のためのアドレスを載せた SSPBUF をロードする。**
4. **アドレスはすべての 8bit が送信されるまで SDA ピンからシフトされる。**
5. **SSP モジュールはスレーブデバイスから ACK ビットにシフトし、その値を SSPCON2 レジスタ (SSPCON2<6>) にライトする。**
6. **モジュールは SSPIF をセットすることにより 9 番目のクロックサイクルの最後に割込みを発生させる。**
7. **8bit のデータを載せた SSPBUF をロードする。**
8. **DATA はすべての 8bit が送信されるまで SDA ピンからシフトされる。**

9. SSP モジュールはスレーブデバイスから ACK ビットをシフトインし、その値を SSPCON2 レジスタ (SSPCON2<6>) にライトする。
10. モジュールは SSPIF をセットすることにより 9 番目のクロックサイクルの最後に割込みを発生させる。
11. SSPCON2 の STOP イネーブルビット PEN をセットすることにより STOP 条件を発生させる。

15.2.6 ボーレートジェネレータ

I²C マスタモード動作では、BRG に再ロードされた値は、SSPADD レジスタの下位 7bit に置かれています (図 15-21)。BRG がこの値をロードすると、BRG は、0 までカウントダウンされて、次の再ロードが起こるまで停止します。I²C マスタモードでは、BRG は自動的に再ロードしません。例えばクロックアービトレーションが起こる場合、SCL ピンがハイにサンプリングされると BRG は再ロードします (図 15-22)。

図 15-21: ボーレートジェネレータのブロック図

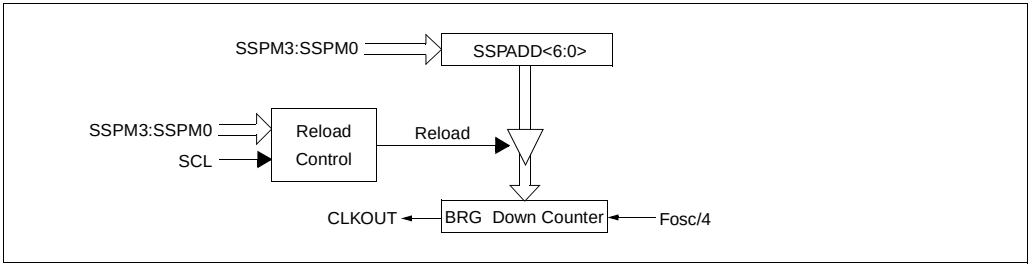
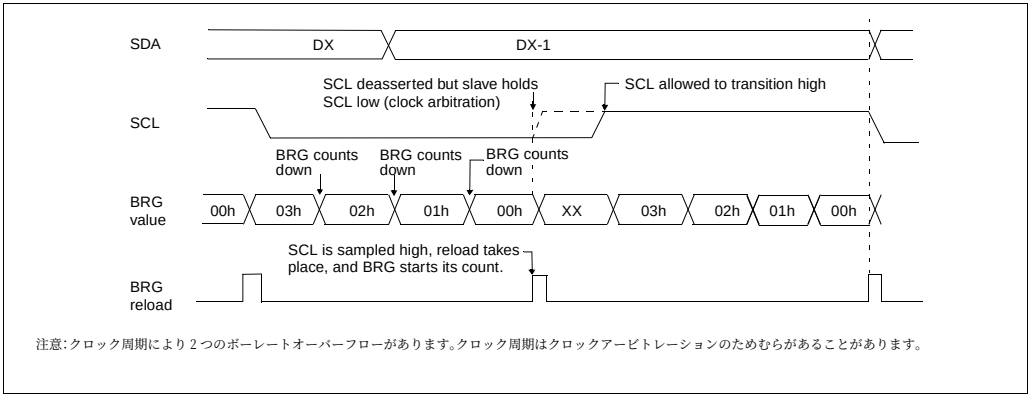


図 15-22: クロックアービトレーションを伴うボーレートジェネレータのタイミング



注意:クロック周期により 2 つのボーレートオーバーフローがあります。クロック周期はクロックアービトレーションのためむらがあることがあります。

15.2.7 I²C マスタモードのスタートコンディションタイミング

START コンディションを始めるには、スタートコンディションイネーブルビットまたは SEN ビット (SSPCON2<0>) をセットします。SDA と SCL ピンがハイにサンプリングされると、ボーレートジェネレータは SSPADD<6:0> の内容を再ロードし、そのカウントを始めます。ボーレートジェネレータがタイムアウトする時に (TBRG)、SCL と SDA が両方ともハイにサンプリングされると、SDA ピンがローに駆動されます。SCL がハイの間、ローで駆動している SDA の動作が START コンディションになり、S ビット (SSPSTAT<3>) がセットされます。I²C モジュールはマスタモードに設定されているので、S ビットの ' 1 ' により SSPIF フラグをセットすることができます。これに続き、ボーレートジェネレータは SSPADD<6:0> の内容を再ロードし、そのカウントを再び始めます。ボーレートジェネレータがタイムアウトすると (TBRG)、SSPCON2 レジスタの SEN ビットは自動的にクリアされ、ボーレートジェネレータはSDAラインがローを保持されるのを延期し、START コンディションが完結します。

注意 : START コンディションの始めに、SDA と SCL ピンがすでにローにサンプリングされている場合、または START 条件の間に、SDA ラインがローに駆動する前にSCLラインがローにサンプリングされる場合、バス衝突が起こり、バス衝突割込みフラグ (BCLIF) がセットされ、START コンディションが打ち切られ、I²C モジュールが IDLE 状態にリセットされます。

15.2.7.1 WCOL ステータスフラグ

START のシーケンスが進行中に SSPBUF にライトすると、WCOL がセットされバッファの内容が変化しません (ライトは起こりません)。

注意 : イベントの待ち行列ができないので、SSPCON2 の下位 5bit にライトすると、START コンディションが終了するまでディセーブルします。

図 15-23: 最初のスタートビットのタイミング

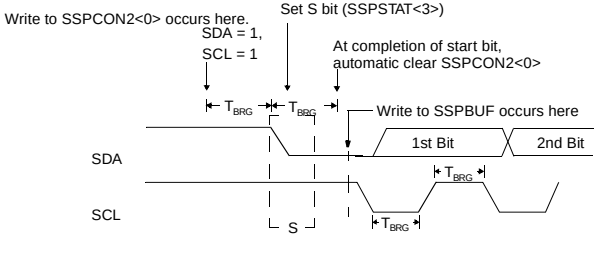
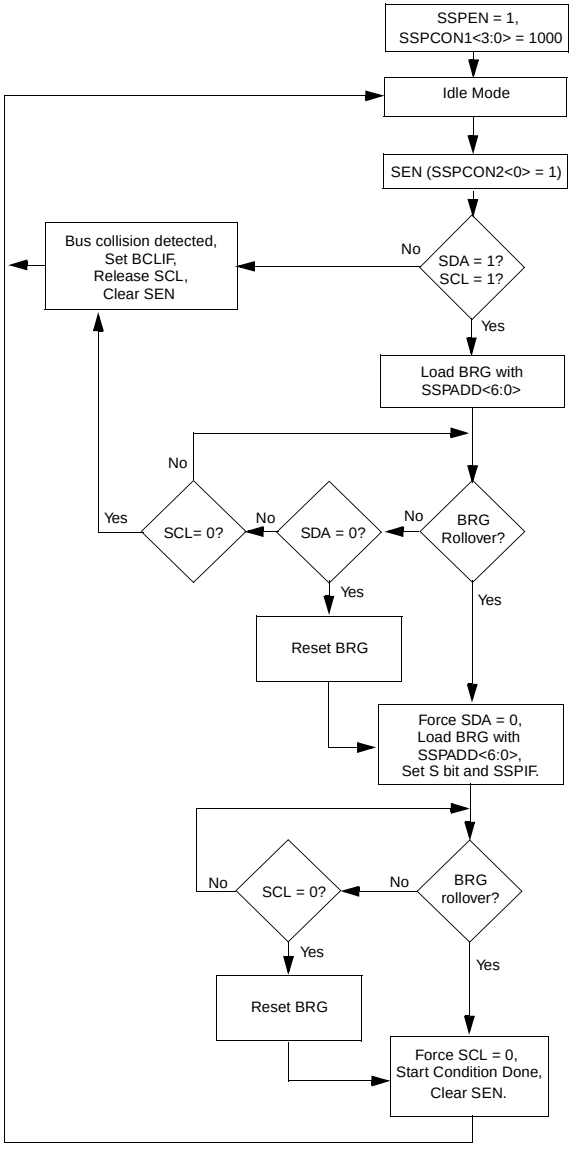


図 15-24: スタート条件のフローチャート



15.2.8 I²C マスタモードの再スタートコンディションタイミング

RESTART 条件は、RSEN ビット (SSPCON2<1>) がハイにプログラムされ、SSP モジュールがアイドル状態になると起こります。RSEN ビットがセットされると、SCL ピンがローを示します。SCL ピンがローにサンプリングされると、ボーレートジェネレータは SSPADD<5:0> の内容をロードし、そのカウントを始めます。SDA ピンは 1 つのボーレートジェネレータカウント (TBRG) にリリースされます (ハイになる)。ボーレートジェネレータがタイムアウトしている時、SDA がハイにサンプリングされると、SCL ピンは示されません (ハイになる)。 SCL がハイにサンプリングされると、ボーレートジェネレータは SSPADD<6:0> の内容を再ロードし、そのカウントを始めます。SDA と SCL は 1 つの TBRG に対してハイにサンプリングする必要があります。この動作は、SCL=1 の間 1 つの TBRG に対して SDA ピン (SDA=0) が断定され、続きます。これに続き、SSPCON2 レジスタの RSEN ビットが自動的にクリアされ、ボーレートジェネレータは SDA ピンをローに保持したままで、再ロードされません。スタートコンディションが SDA と SCL ピンで検出されると、S ビット (SSPSTAT<3>) がセットされます。SSPIF はボーレートジェネレータがタイムアウトしてしまうまでセットされません。

注 1: 送信が進行中に RSEN がプログラムされる場合は実行されません。

注 2: 次のような場合、RESTART 条件の間にバス衝突が起こります。

- **SCL がローからハイになる時に SDA がローにサンプリングされる。**
- **SDA がローを示す前に、SCL がローになる。これは多分もう 1 つのマスタがデータ “1” を送信しようとしていることを示しています。**

SSPIF ビットがセットされた直後、7bit モードの 7bit アドレス、または 10bit モードのデフォルトの最初のアドレスで、SSPBUF をライトします。最初の 8bit が送信され ACK が受信された後、さらにアドレスの 8bit(10bit モード)、またはデータの 8bit(7bit モード) を送信します。

SSPBUF へのライト後、すべての 7 アドレスビットと R/W ビットが終了するまで、アドレスの各ビットは SCL の立ち下がりエッジでシフトアウトされます。8 番目のクロックの立ち下がりエッジでは、スレーブが応答に答えるのを可能にするために、マスタは SDA ピンを示しません。9 番目のクロックの立ち下がりエッジでは、アドレスがスレーブにより認識されるかどうかを見るために、マスタは SDA ピンをサンプリングします。ACK ビットのステータスは、AKSTAT ステータスビット SSPCON2<6> にプログラムされます。アドレスの 9 番目のクロック送信の立ち下がりエッジに続いて、SSPIF がセットされ、BF フラグがクリアされます。そして SCL をローに保持し、SDA をフロートにするのは可能です、SSPBUF へのもう 1 つのライトが起こるまでボーレートジェネレータはオフになります。

15.2.8.1 WCOL ステータスフラグ

RESTART のシーケンスが進行中に SSPBUF にライトすると、WCOL がセットされバッファの内容が変化しません (ライトは起こりません)。

注意 : イベントの待ち行列ができないので、SSPCON2 の下位 5bit にライトすると、RESTART コンディションが終了するまでディセーブルにします。

図 15-25: 連続するスタート条件のタイミング

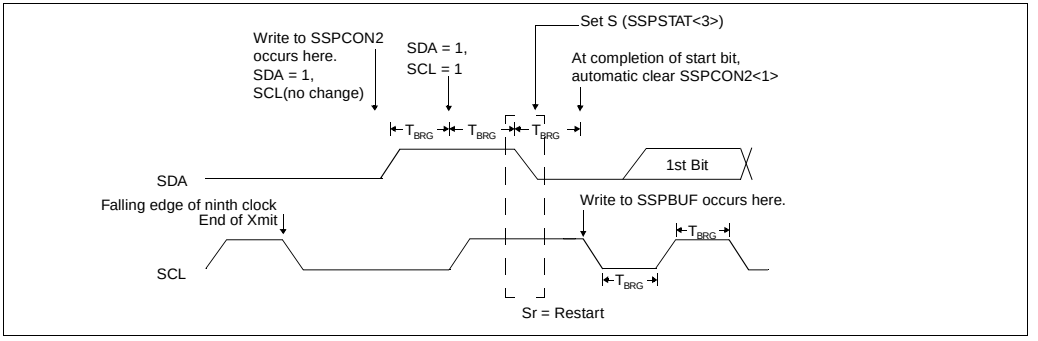


図 15-26: RESTART 条件のフローチャート (ページ 1)

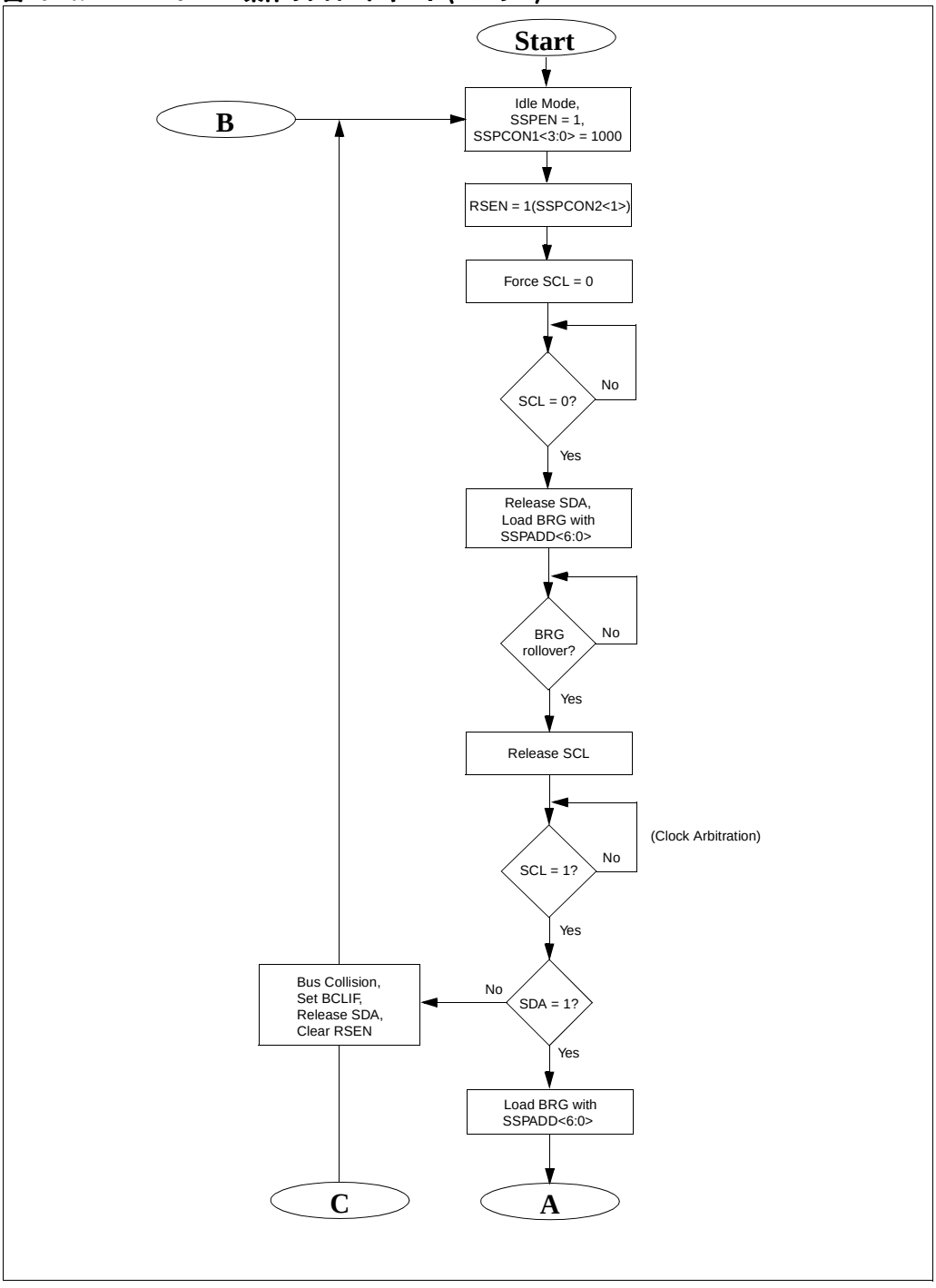
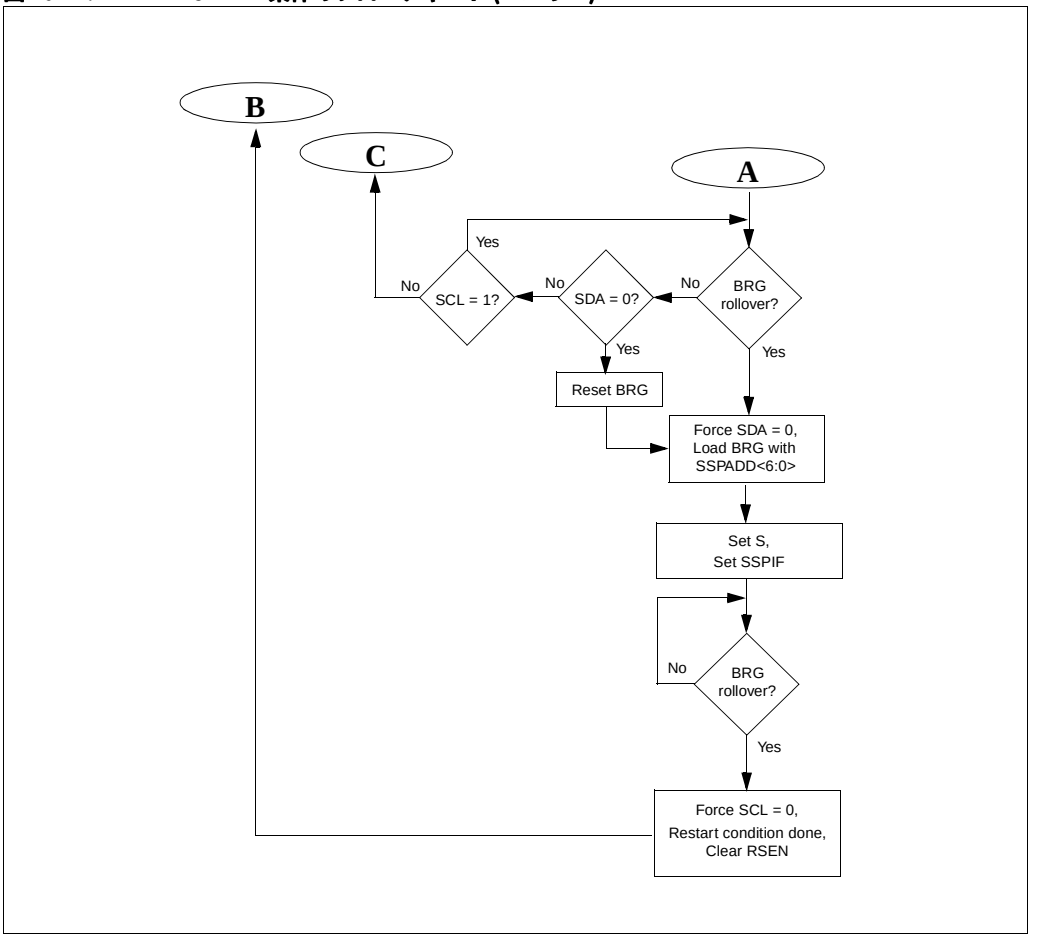


図 15-27: RESTART 条件のフローチャート (ページ 2)



15.2.9 I²C マスタモードの送信

データバイト、7bit アドレス、10bit アドレスのどちらか半分の送信は、単に SSPBUF レジスタに値をライトすることで完了します。この動作はバッファフルフラグ (BF) をセットし、ボーレートジェネレータがカウントを開始し、次の送信を始めるのを可能にします。アドレス / データの各ビットは SCL の立ち下がりエッジをアサートした後、SDA ピンにシフトアウトされます (データの保持時間設定を参照)。SCL は 1 つのボーレートジェネレータロールオーバーカウント(TBRG)でローに保持されています。データは SCL がハイにリリースされる前に有効である必要があります (データのセットアップ時間設定を参照)。SCL ピンがハイにリリースされ、TBRG に対してもその方法で保持されている時は、SDA ピンのデータを、SCL の次の立ち下がりエッジの後の保持時間や持続期間に対して安定した状態のままにする必要があります。アドレスの一致が起こったり、データが正しく受信された場合、8 番目のビットがシフトアウトされた後(8番目のクロックの立ち下がりエッジ)、BF フラグがクリアされ、マスタが SDA をリリースし、9 番目のビット時間の間に $\overline{ACK}$ ビットに答えるためにスレーブデバイスをアドレスすることが可能です。$\overline{ACK}$ のステータスは、9 番目のクロックの立ち下がりエッジの SSPCON2 レジスタの bit6 に読み込まれます。マスタが応答を受信すると、応答ステータスビット (AKSTAT) がクリアされます。受信しないとそのビットはセットされます。9番目のクロックの後、SSPIF はセットされ、マスタクロック (ボーレートジェネレータ) は、SCL をローにし SDA を変化させないまま次のデータバイトが SSPBUF にロードされるまで延期されます。

15.2.9.1 BF ステータスフラグ

送信モードでは、CPU が SSPBUF にライトすると BF ビット (SSPSTAT<0>) がセットされ、すべての 8bit をシフトアウトすると BF ビットはクリアされます。

15.2.9.2 WCOL ステータスフラグ

送信がすでに進行中 (すなわち SSPSR がまだデータバイトをシフトアウトしている時) に SSPBUF にライトすると、WCOL がセットされバッファの内容が変化しません (ライトは起こりません)。

WCOL は、ソフトウエアでクリアする必要があります。

15.2.9.3 AKSTAT ステータスフラグ

送信モードでは、スレーブが応答 ($\overline{ACK}$=0) を送る時に AKSTAT ビット (SSPCON2<6>) がクリアされ、スレーブが応答しない ($\overline{ACK}$=1) 時、セットされます。スレーブがそのアドレス (ジェネラルコールを含む) を認識したり、正しくそのデータを受信すると応答を送ります。

図 15-28:　　マスタ送信のフローチャート

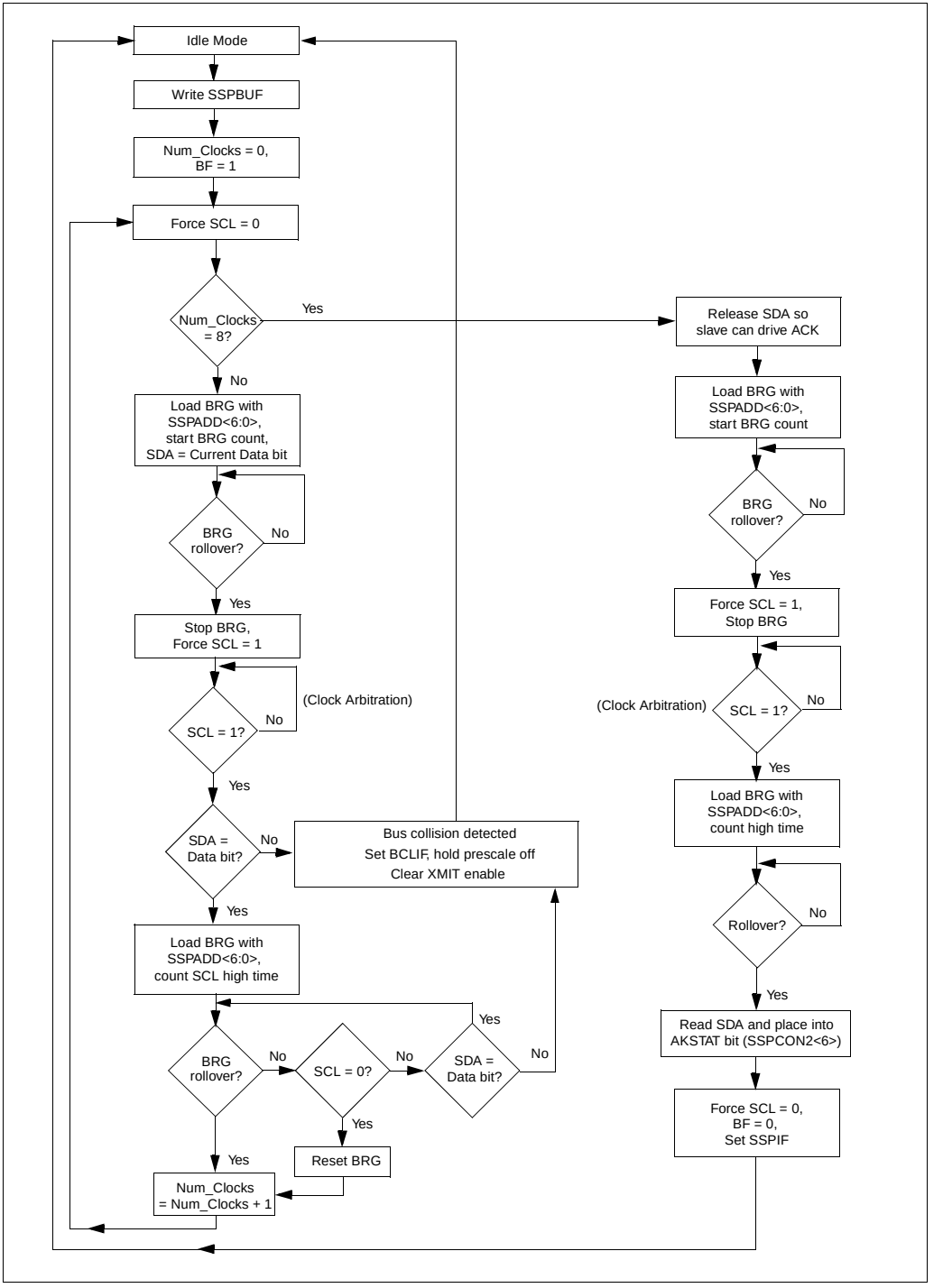
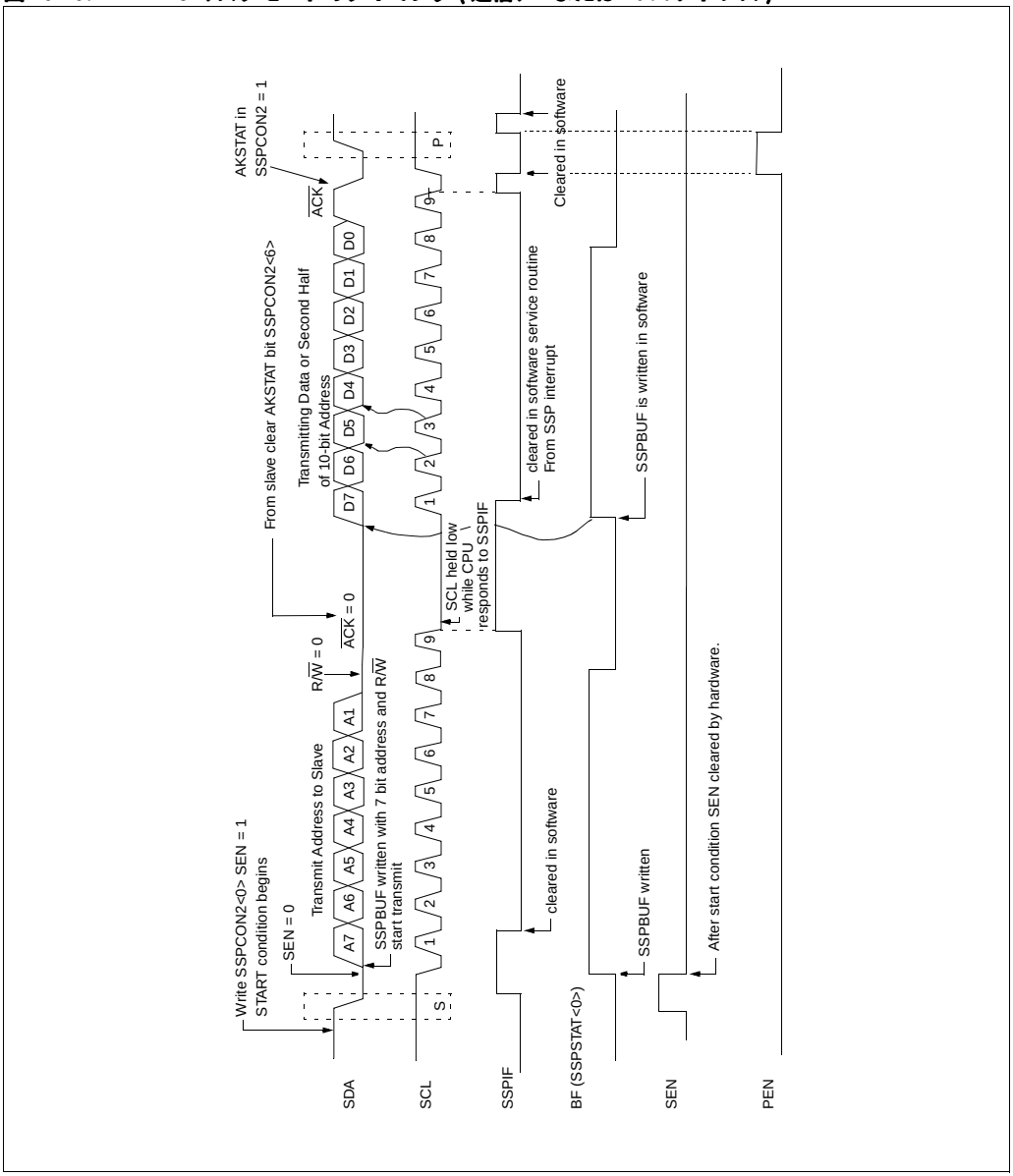


図 15-29: I²C マスタモードのタイミング (送信、7 または 10bit アドレス)



15.2.10 I²C マスタモードの受信

マスタモードの受信は受信イネーブルビット RCEN(SSPCON2<3>) をプログラミングすることによりイネーブルされます。

注意： RCE ビットがセットされる前や、RCEN ビットを無視するときは、SSP モジュールは IDLE モードでなければいけません。

ボーレートジェネレータがカウントを開始し、各ロールオーバーで SCL ピンの状態が変化し (ハイからロー / ローからハイ)、データが SSPSR にシフトインされます。8 番目のクロックの立ち下がりエッジの後、受信イネーブルフラグは自動的にクリアされ、SSPSR の内容が SSPBUF にロードされ、BF フラグや SSPIF がセットされ、ボーレートジェネレータが SCL をローに保持すると、カウントすることを延期します。SSP は IDLE 状態で、次のコマンドを待っています。バッファが CPU によって読み込まれると、BF フラグが自動的にクリアされます。その時、応答シーケンスイネーブルビット ACKEN(SSPCON2<4>) をセットすることにより、受信の最後に応答ビットを送ることができます。

15.2.10.1 BF ステータスフラグ

受信動作では、アドレスまたはデータバイトが SSPSR から SSPBUF にロードされると、BF がセットされます。SSPBUF をリードすると、クリアされます。

15.2.10.2 SSPOV ステータスフラグ

受信動作では、8bit が SSPSR に受信されると、SSPOV がセットされます。BF フラグはすでに前の受信からセットされています。

15.2.10.3 WCOL ステータスフラグ

受信がすでに進行中 (すなわち SSPSR がまだデータバイトでシフトインしている時) に SSPBUF にライトすると、WCOL がセットされバッファの内容が変化しません (ライトは起こりません)。

図 15-30:　　マスタ受信のフローチャート

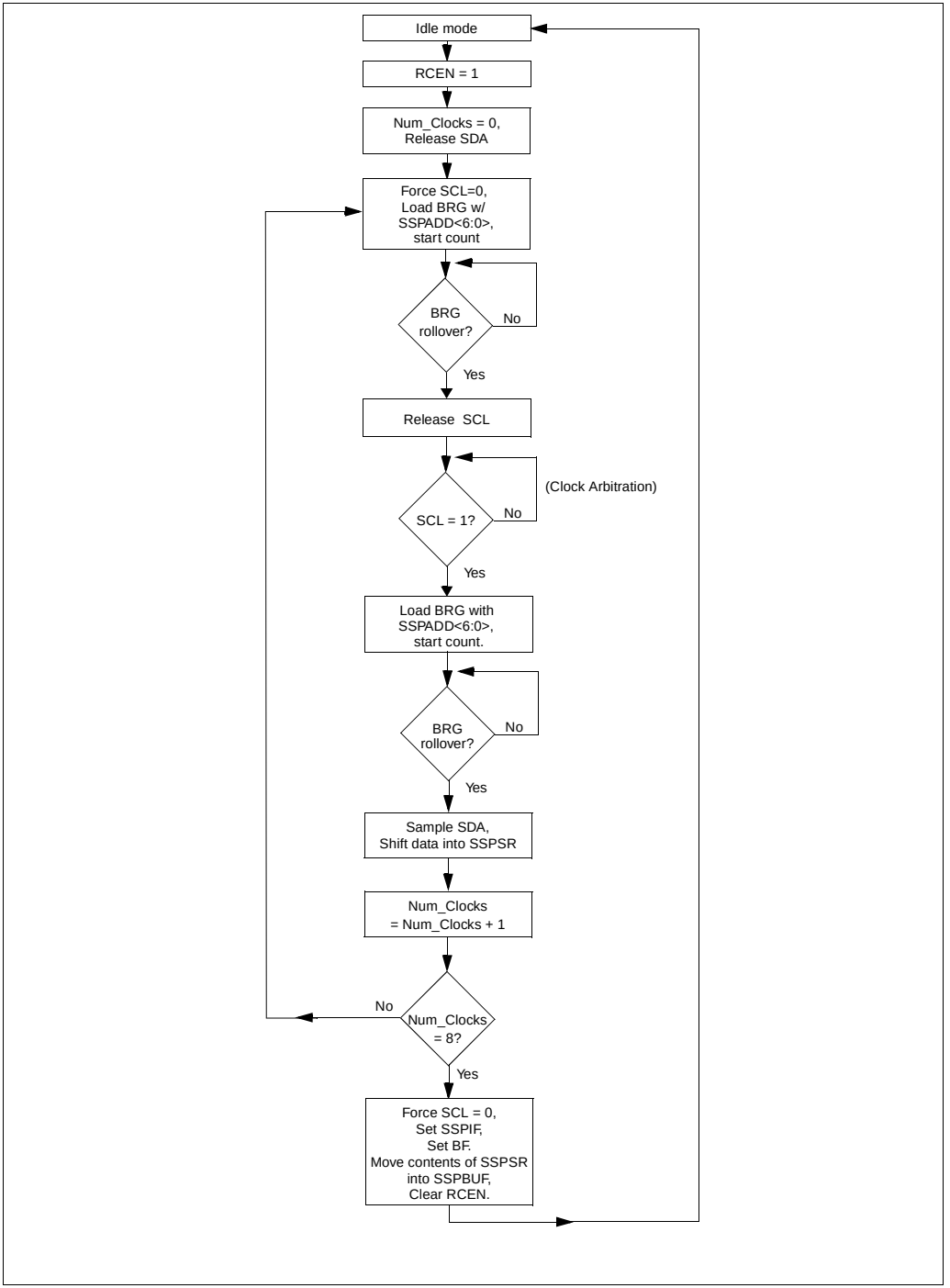
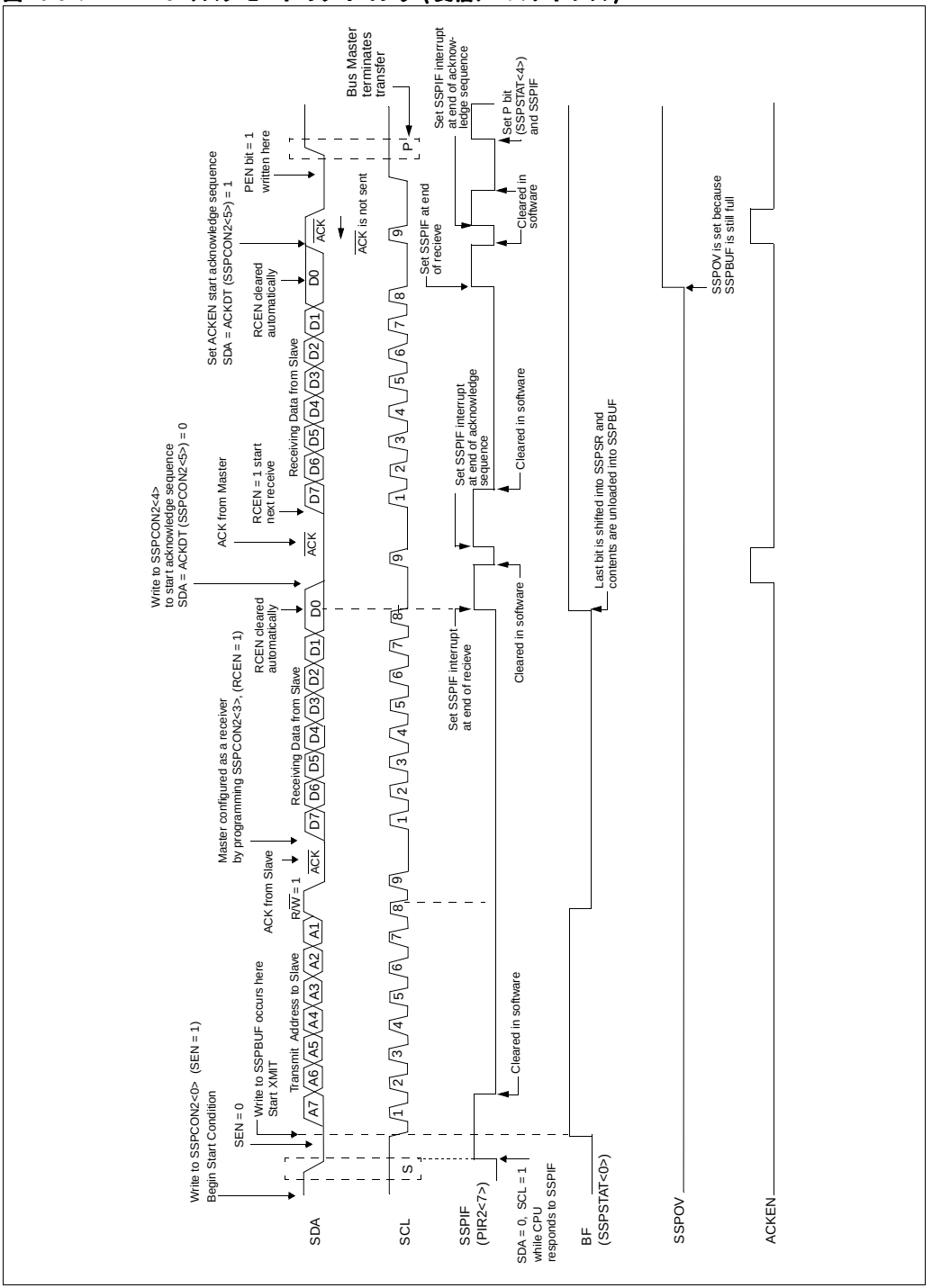


図 15-31: I²C マスタモードのタイミング (受信、7bit アドレス)



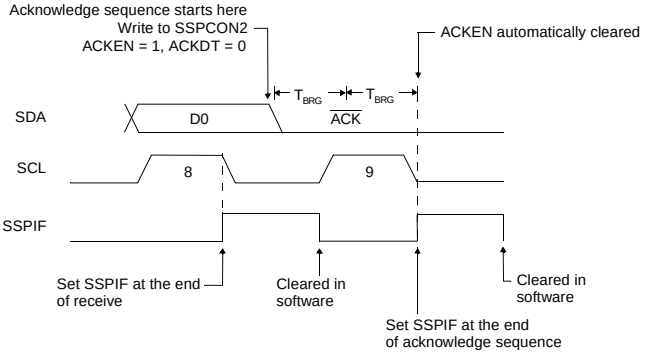
15.2.11 応答シーケンスのタイミング

応答シーケンスは応答シーケンスイネーブルビット ACKEN(SSPCON2<4>) をセットすることによりイネーブルされます。このビットがセットされると、SCL ピンがローに引っ張られ、応答データビットの内容が SDA ピンに与えられます。応答を発生させたい場合、ACKDT ビットをクリアする必要があります。そうでない場合は、応答のシーケンスを開始する前にACKDT ビットをセットする必要があります。その時ボーレートジェネレータは1つのロールオーバー周期 (TBRG) のためのカウントをし、SCL ピンはアサートされません (ハイに引っ張られる)。SCL ピンがハイにサンプリングされると (クロックアービトレーション)、ボーレートジェネレータは TBRG のためのカウントをします。SCL ピンが 1 つの TBRG に対してローに引っ張られます。これに続いて、ACKEN ビットは自動的にクリアされ、ボーレートジェネレータはオフになり、SSP モジュールは IDLE モードになります (図 15-32)。

15.2.11.1 WCOL ステータスフラグ

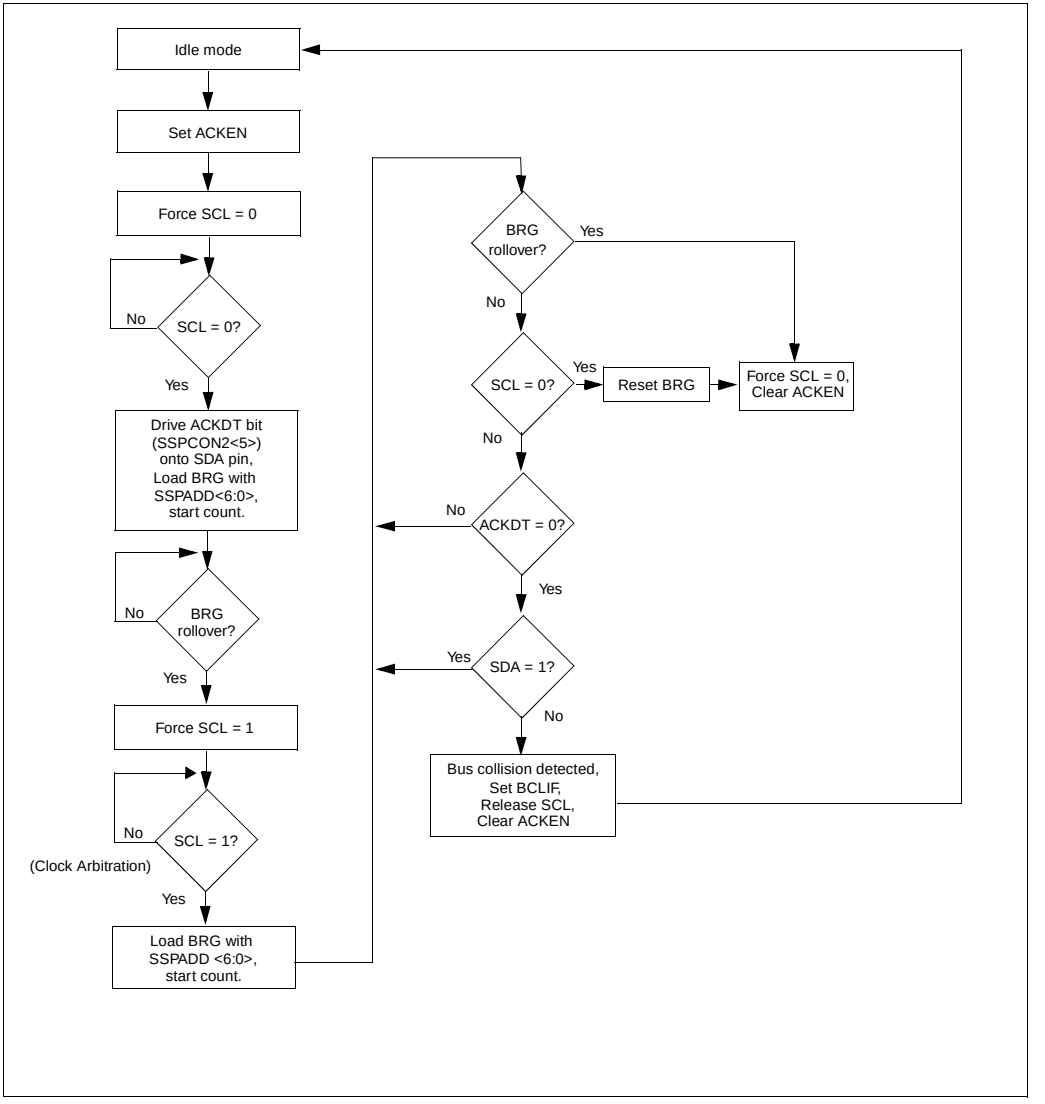
応答のシーケンスが進行中に SSPBUF にライトすると、WCOL がセットされバッファの内容が変化しません (ライトは起こりません)。

図 15-32: 応答シーケンスのタイミング



注意:T_{BRG}=1 つのボーレートジェネレータ周期

図 15-33:　　応答のフローチャート



15.2.12 ストップ条件のタイミング

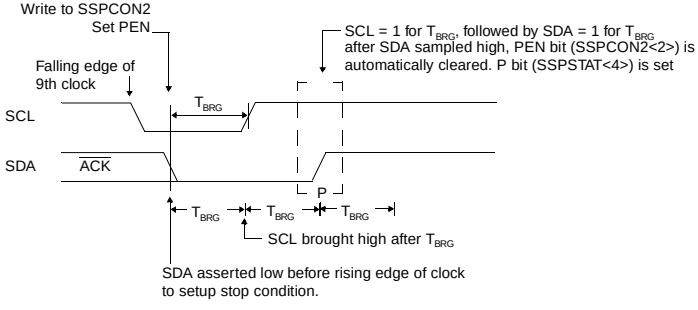
ストップビットはストップシーケンスイネーブルビット PEN(SSPCON2<2>) をセットすることにより受信 / 送信の最後に SDA ピンでアサートされます。受信 / 送信の最後に、9 番目のクロックの立ち下がりエッジの後 SCL ラインがローに保持されます。PEN ビットがセットされると、マスタは SDA ラインをローにします。SDA ラインがローにサンプリングされると、ボーレートジェネレータが再ロードされ、0 までカウントダウンします。ボーレートジェネレータがタイムアウトすると、そして SCL ピンが 1TBRG 後 (ボーレートジェネレータロールオーバーカウント) ハイに持ち上げられ、SDA ピンはアサートされません。SCL がハイの間に SDA ピンがハイにサンプリングされると、PEN ビットは自動的にクリアされ、P ビット (SSPSTAT<4>) が、SSPIF フラグをセットするターンの中でセットされます (図 15-34)。

CPU がバスを制御しようとする時はいつでも、SSPSTAT レジスタの S と P のビットをチェックすることによりバスがビジーかどうかを最初に確かめます。バスがビジーの場合、STOP ビットが検出されると (バスがフリー)、CPU に割込む (通告する) ことができます。

15.2.12.1 WCOL ステータスフラグ

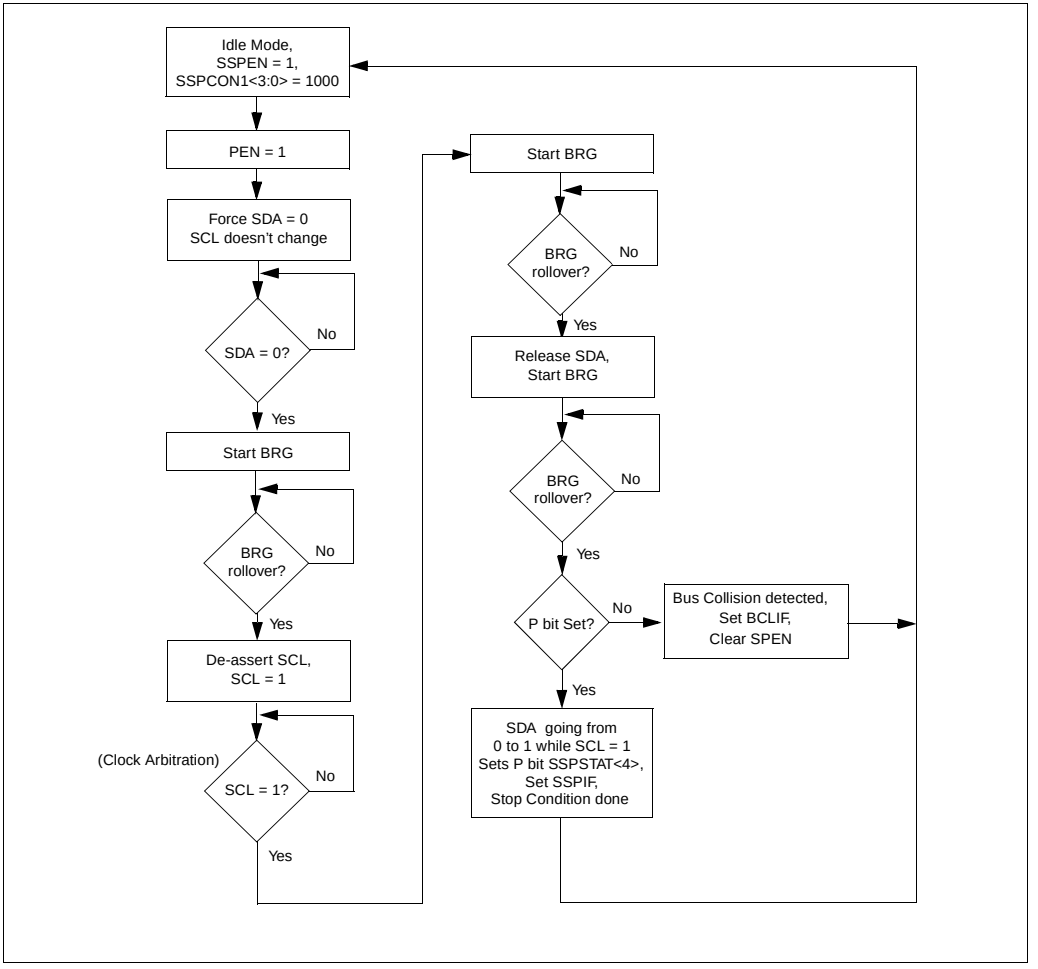
STOP シーケンスが進行中に SSPBUF をライトすると、WCOL がセットされバッファの内容が変化しません (ライトは起こりません)。

図 15-34: 受信 / 送信モードのストップ条件



注意:TBRG=1 つのボーレートジェネレータ周期

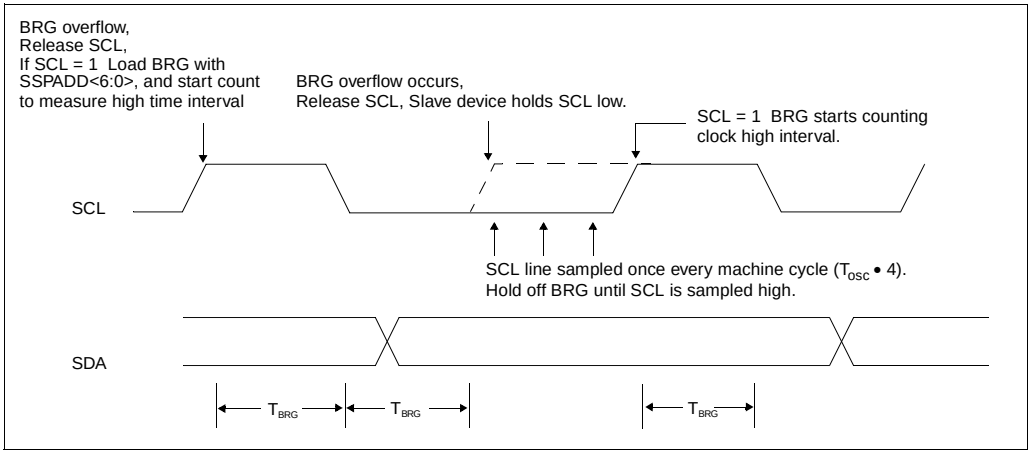
図 15-35: STOP 条件のフローチャート



15.2.13 クロックアービトレーション

クロックアービトレーションは、受信、送信、再スタート / ストップ条件のいずれの間でもマスタが SCL ピンをアサートしない時に起こります (SCL ピンはハイにフロートすることが可能)。SCL ピンがハイにフロートすることが可能になると、SCL ピンが実際にハイにサンプリングするまで、ボーレートジェネレータ (BRG) はカウントを延期します。SCL ピンがハイにサンプリングされると、ボーレートジェネレータは SSPADD<6:0> の内容を再ロードし、カウントを開始します。これにより、クロックが外部デバイスによりローに保持されるイベントでは、SCL のハイタイムが常に少なくとも 1BRG ロールオーバーカウントにあることが確実になります。

図 15-36: マスタ送信モードでのクロックアービトレーションのタイミング



15.2.14 マルチマスタ通信、バス衝突、バスアービトレーション

マルチマスタモードのサポートは、バスアービトレーションにより達成されます。マスタがアドレス / データのビットを SDA ピンに出力する時、SDA をハイにフロートさせることでマスタが SDA に ‘ 1 ’ を出力し、もう 1 つのマスタが ‘ 0 ’ をアサートすると、アービトレーションは起こります。SCL ピンがハイの間は、データを固定させる必要があります。SDA の予期したデータが ‘ 1 ’ で、SDA ピンにサンプリングされたデータが = ‘ 0 ’ の場合、バス衝突は起こっています。マスタはバス衝突割込みフラグ BCLIF をセットし、I^2CポートをIDLE状態にリセットします(図15-37)。

バス衝突が起こった時に送信が進行中の場合は、送信は停止し、BF フラグがクリアされ、SDA と SCL ラインがアサートされず、SSPBUF をライトすることができます。バス衝突割込みサービスルーチンを使う時、I^2C バスがフリーの場合は START 条件を示すことにより通信を再開することができます。

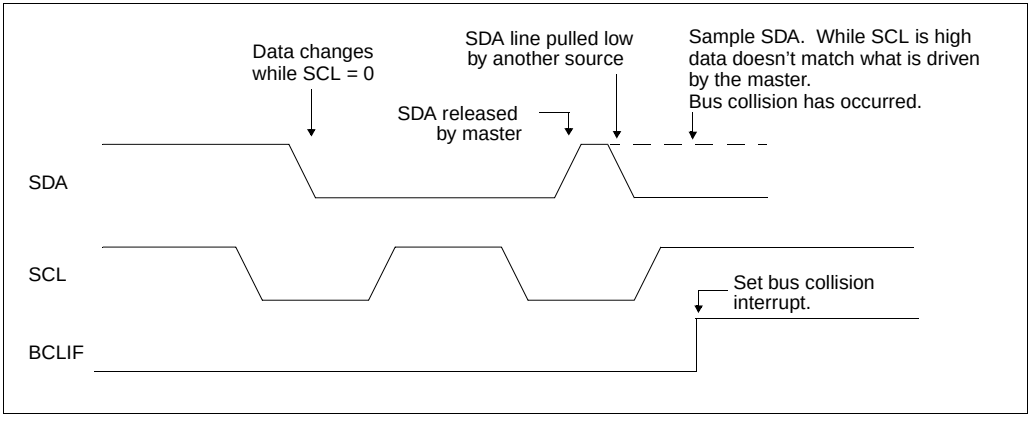
バス衝突が起こった時に、START、RESTART、STOP、応答条件が進行中の場合は、この条件は打ち切られ、SDA と SCL ラインがアサートされず、SSPCON2 レジスタのそれぞれの制御ビットがクリアされます。バス衝突割込みサービスルーチンを使う時、I^2C バスがフリーの場合は START コンディションを示すことにより通信を再開することができます。

マスタは SDA と SCL ピンを監視し続けます。そして STOP コンディションが起こると、SSPIF ビットがセットされます。

SSPBUF へのライトは、バス衝突が起こった時にそこでの送信機を止めるにもかかわらず、最初のデータビットでデータの送信を開始します。

マルチマスタモードでは、スタートとストップ条件の検出で割込みが発生することにより、バスがフリーの時の決定が可能です。SSPSTAT レジスタの P ビットがセットされるか、バスがアイドルでSビットとPビットがクリアされる時に、I^2C バスを制御することができます。

図 15-37: 送信と応答のバス衝突のタイミング



15.2.14.1　スタートコンディション中のバス衝突

START コンディション中、バス衝突が次のような場合に起こります。

a)　START コンディションの始めに、SDA と SCL がローにサンプリングされている (図 15-38)。

b)　SDA がローを示す前に SCL がローにサンプリングされている (図 15-39)。

START コンディション中、SDA と SCL ピンは両方とも監視されています。

次のような場合：

SDA ピンがすでにローである。または

SCL ピンがすでにローである。

その時は：

START コンディションが打ち切られる。

そして、BCLIF フラグがセットされる。

そして、SSP モジュールが IDLE 状態にリセットされる (図 15-38)。

START コンディションは、アサートされていない SDA と SCL ピンで始まります。SDA ピンがハイにサンプリングされると、ボーレートジェネレータは SSPADD<6:0> からロードされ、0 までカウントダウンします。SDA がハイの間 SCL ピンがローにサンプリングされると、バス衝突が起こりますが、それはもう 1 つのマスタが START コンディション中にデータ‘ 1 ’を駆動しようとしていると考えられるからです。

SDA ピンがこのカウント中にローにサンプリングされると、BRG はリセットされ、SDA ラインが早くアサートされます (図 15-40)。しかし ‘ 1 ’ が SDA ピンにサンプリングされると、SDA ピンは BRG カウントの最後にローをアサートします。ボーレートジェネレータは再ロードされ、0 までカウントダウンし、この間に SCL ピンが ‘ 0 ’ としてサンプリングされると、バス衝突は起こりません。BRG カウントの最後に、SCL ピンはローを示します。

注意：　バス衝突が START コンディション中の要因ではない理由は、2 つのバスマスタが正確に同じ時間で START コンディションをアサートすることができないからです。そのために 1 つのマスタは常に他より前に SDA をアサートします。この条件ではバス衝突は起こりませんが、それは 2 つのマスタは START コンディションに続く最初のアドレスをアービトレイトすることが可能でなければならないからです。そしてアドレスが同じ場合、アービトレーションはデータ部分である RESTART、または STOP コンディションに続けることが必要です。

図 15-38:　　スタート条件中のバス衝突 (SDA のみ)

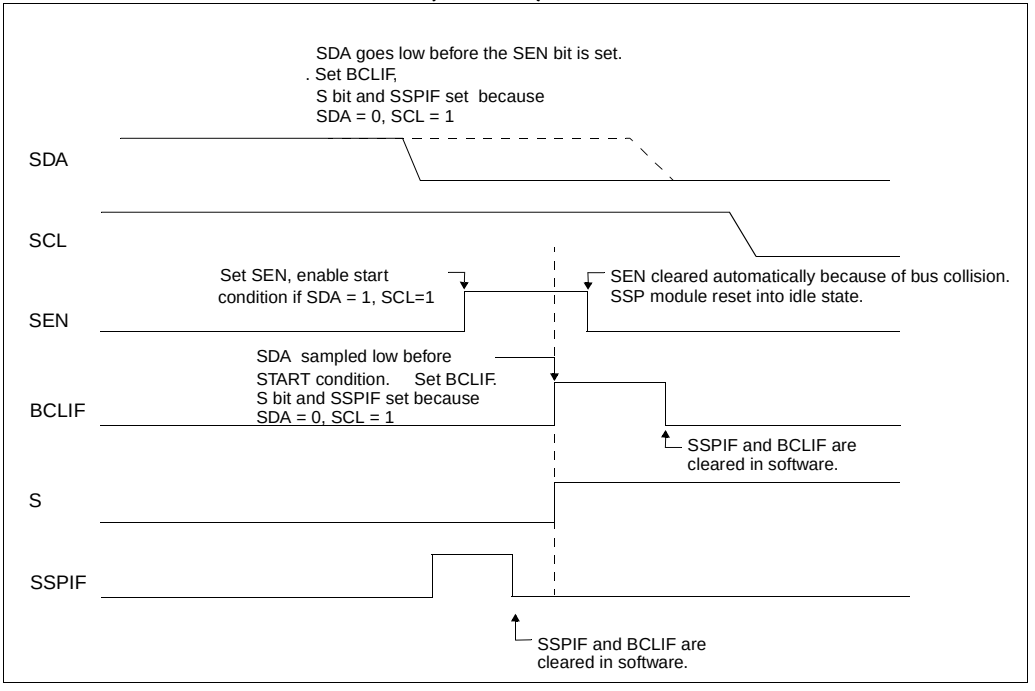


図 15-39: スタート条件中のバス衝突 (SCL=0)

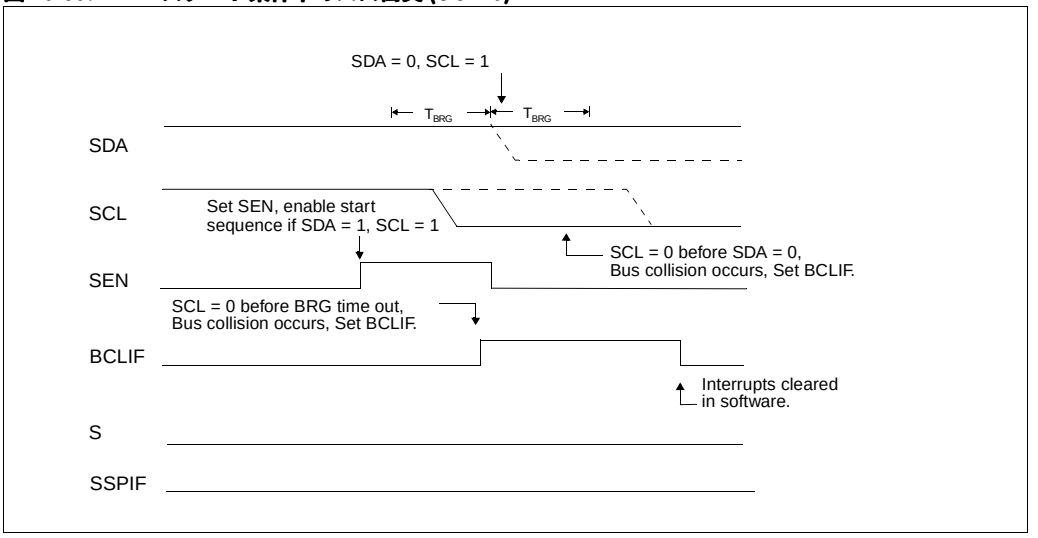
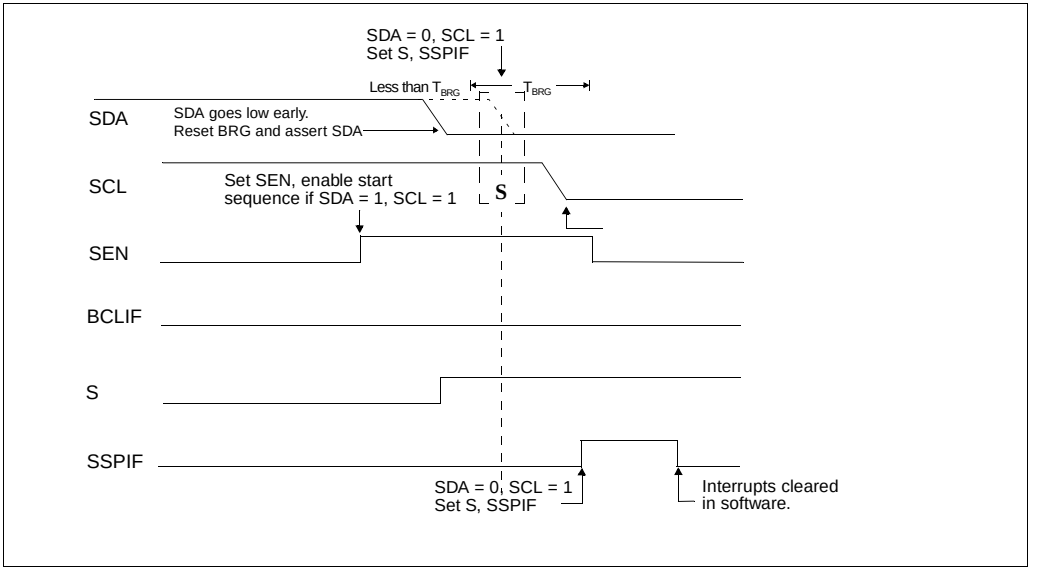


図 15-40: スタート条件中の SDA バス衝突のための BRG リセット



15.2.14.2 RESTART 条件中のバス衝突

RESTART 条件中、バス衝突が次のような場合に起こります。

a) SCL が ' 0 ' から ' 1 ' になる時に、' 0 ' が SDA にサンプリングされている。

b) SDA がローをアサートする前に SCL がローになり、もう 1 つのマスタがデータ ' 1 ' を送信しようとしているのを示している。

SDA をアサートせず、そのピンがハイにフロートするのが可能な時、BRG は SSPADD<6:0> をロードし、0 までカウントダウンします。その時 SCL ピンはアサートされず、ハイにサンプリングされると、SDA ピンがサンプリングされます。SDA がローの場合、バス衝突が起こっています (つまり、もう 1 つのマスタがデータ ' 0 ' を送信しようとしている)。しかし SDA がハイにサンプリングされると、BRG は再ロードされカウントを始めます。BRG がタイムアウトする前に SDA がハイからローになると、2 つのマスタは正確に同じ時間でSDAをアサートすることができないのでバス衝突は起こりません。

しかし BRG がタイムアウトする前で SDA がまだアサートされる前に SCL がハイからローになると、バス衝突は起こります。この場合は、もう 1 つのマスタが RESTART 条件中にデータ ' 1 ' を送信しようとしています。

BRG のタイムアウトの最後に、SCL と SDA の両方がまだハイの場合、SDA ピンがローで駆動され、BRG が再ロードされ、カウントを始めます。SCL ピンの状態にかかわらず、カウントの最後には SCL ピンはローに駆動され、RESTART コンディションが終了します (図 15-41)。

図 15-41: RESTART 条件中のバス衝突 (ケース 1)

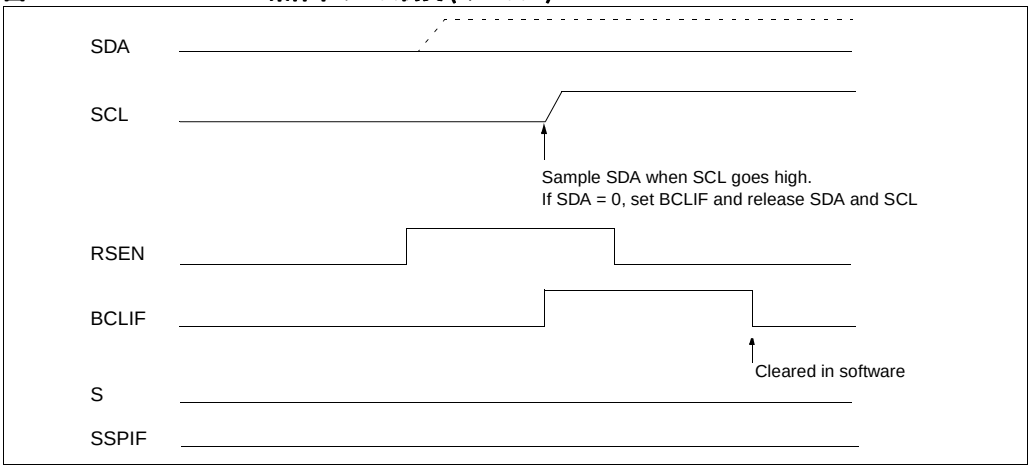
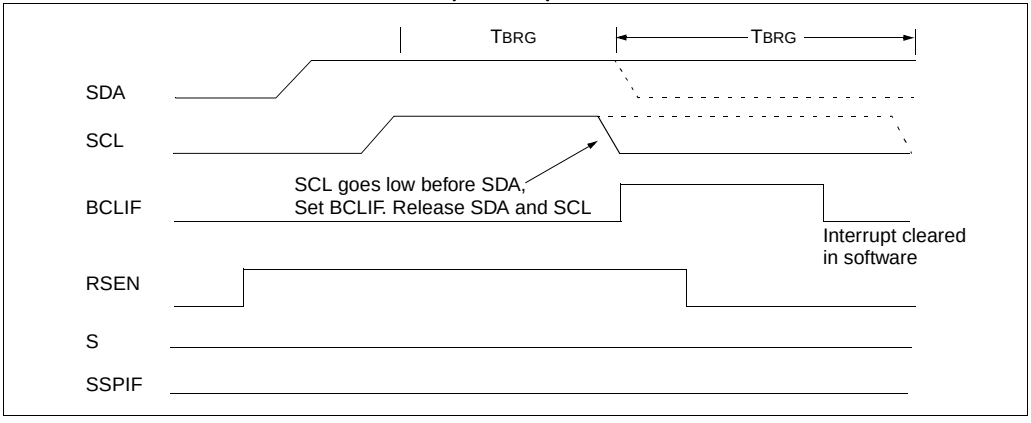


図 15-42: RESTART 条件中のバス衝突 (ケース 2)



15.2.14.3 STOP 条件中のバス衝突

STOP コンディション中､次のような場合にバス衝突が起こります。

a) SDA ピンがアサートされず､ハイにフロートするのを許可した後で､BRG がタイムアウトした後に SDA がローにサンプリングされる。

b) SCL ピンがアサートれない後で､SDA がハイになる前に SCL がローにサンプリングされる。

STOP コンディションは SDA がローにアサートされることで始まります。SDA がローにサンプリングされる時、SCL ピンはフロートすることが可能です。そのピンがハイにサンプリングされると (クロックアービトレーション)、ボーレートジェネレータは SSPADD<6:0> をロードし、0 までカウントダウンします。BRG のタイムアウト後、SDA がサンプリングされます。SDA がローにサンプリングされると、バス衝突が起こっています。これは､もう 1 つのマスタがデータ ' 0 ' を駆動しようとしているためです。SDA がハイにフロートする前に、SCL ピンがローにサンプリングされると、バス衝突が起こります。これは、もう 1 つのマスタがデータ ' 0 ' を駆動しようとしている別のケースです。

図 15-43: STOP コンディション中の M バス衝突 (ケース 1)

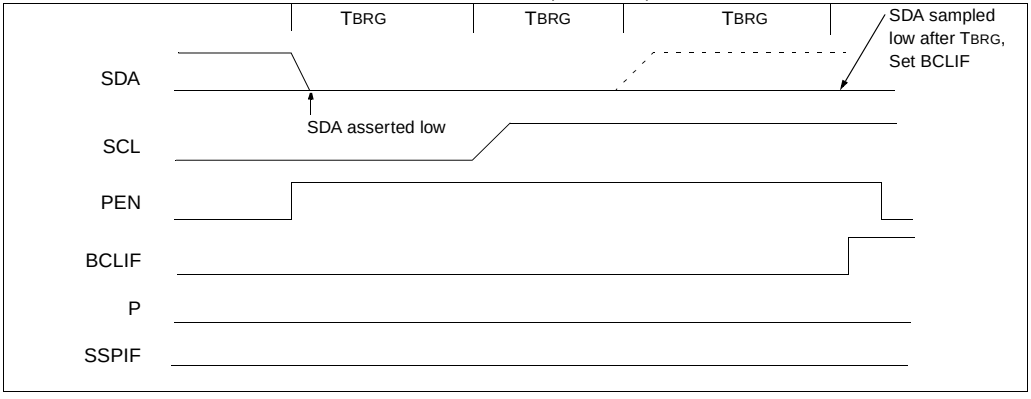
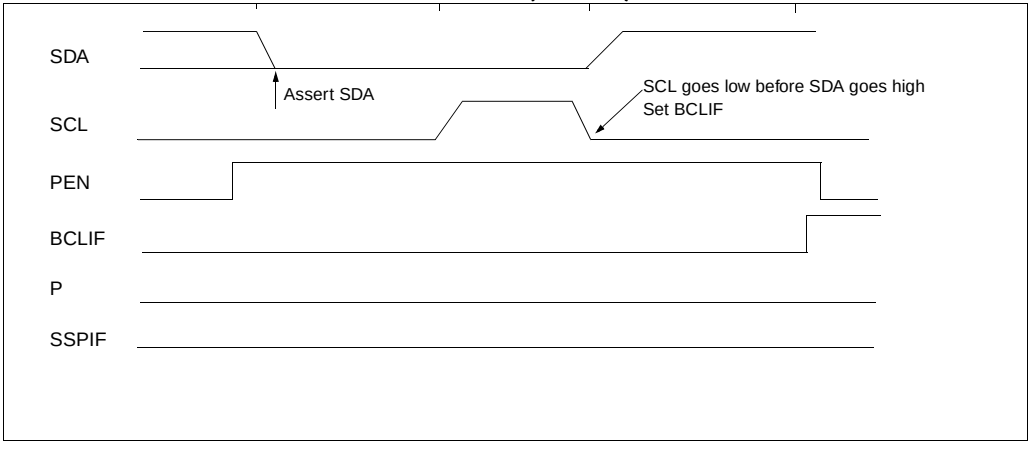


図 15-44: STOP コンディション中のバス衝突 (ケース 2)



15.3 I²C バスの接続の注意点

標準モードの I²C バスデバイスでは、図 15-45 の抵抗器 $R_P R_S$ の値は次のようなパラメータによります。

- **電源電圧**
- **バスキャパシタンス**
- **接続デバイスの数 (入力電流 + 漏れ電流)**

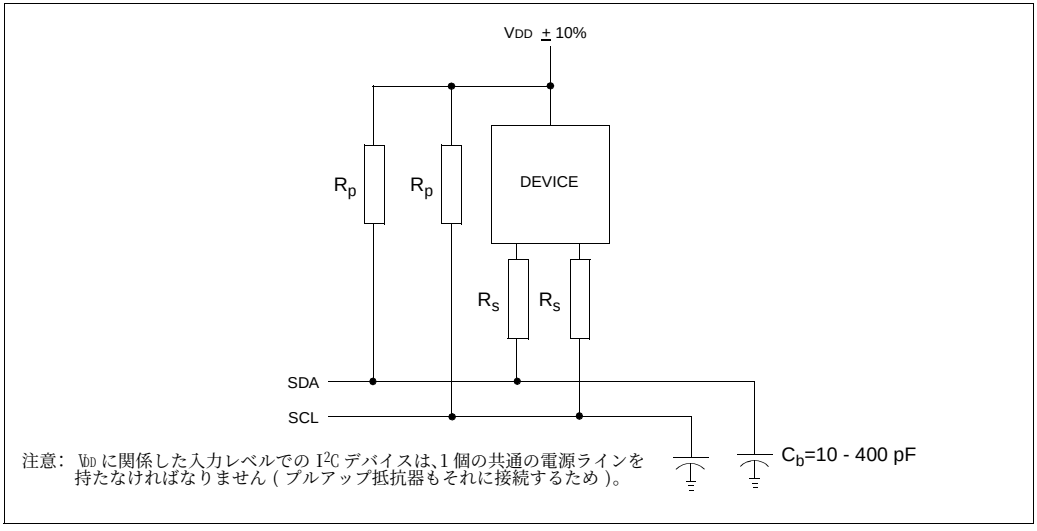
電源電圧は、設定された出力ステージ用に V_{OL} 最大 =0.4V で設定された最小シンク電流 3mA のために、抵抗器 R_P の最小値を制限します。例えば、3mA での V_{DD} の電源電圧 =5V ± 10% と V_{OL} 最大電圧 =0.4V では、R_P 最小 =(5.5-0.4)/0.003=1.7kΩ です。R_P の機能としての V_{DD} は、図 15-45 に示されます。ローレベルに対して $0.1V_{DD}$ の要求されるノイズマージンは、R_S の最大値を制限します。直列抵抗器はオプションです。

バスキャパシタンスはワイヤ、接続、ピンのトータルキャパシタンスです。このキャパシタンスは、設定された立ち上がり時間のために R_P の最大値を制限します (図 15-45)。

SMP ビットはスルーレート制御イネーブルビットです。このビットは SSPSTAT レジスタの中にあり、I²C モード (マスタまたはスレーブ) の時 I/O ピンのスルーレートを制御します。

この制御により、SCL と SDA ピンの立ち上がりと立下がり時間が確実に、400kHz の動作において I²C 仕様で必要とされる最小の時間に適合します。

図 15-45: I²C バスのデバイス設定のサンプル



注意：VDD に関係した入力レベルでの I²C デバイスは、1 個の共通の電源ラインを持たなければなりません (プルアップ抵抗器もそれに接続するため)。

NOTES:

16.0 アナログ - ディジタル変換器 (A/D) モジュール

アナログ - ディジタル (A/D) 変換器モジュールは、PIC17C75X デバイスでは 12 個のアナログ入力があります。

A/D は、アナログ入力信号を対応する 10 ビットのディジタルに変換することができます。サンプルホールドの出力は変換器への入力であり、それは逐次近似によって数値を発生します。

アナログの基準電圧 (正電源と負電源) は、デバイスの電源電圧 (AVDD、AVSS) か RG3/AN0/VREF+ と RG2/AN1/VREF- ピンの電圧レベルのどちらかをソフトウェアで選択可能です。

A/D 変換器にはデバイスが SLEEP モードの間に動作できるという独特の特徴があります。スリープで動作するには、A/D クロックが A/D の内部 RC オシレータから得ている必要があります。

A/D モジュールには次のような 4 個のレジスタがあります。:

- **A/D 結果ハイレジスタ (ADRESH)**
- **A/D 結果ローレジスタ (ADRESL)**
- **A/D 制御レジスタ 0(ADCON0)**
- **A/D 制御レジスタ 1(ADCON1)**

図 16-1 に示すように、ADCON0 レジスタは A/D モジュールの動作を制御します。ADCON1 レジスタは図 16-2 に示すように、ポートピンの機能を設定します。ポートピンはアナログ入力 (RG3 と RG2 は基準電圧となることも可能)、またはディジタル I/O として設定することもできます。

図 16-1: ADCON0 レジスタ (アドレス：14h、バンク 5)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0	R/W-0
CHS3	CHS2	CHS1	CHS0	—	GO/$\overline{\text{DONE}}$	—	ADON
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
U = 未使用のビット、‘0’としてリード
-n = POR リセットでの値

bit 7-4: **CHS2:CHS0**: アナログチャンネル選択ビット
0000 = channel 0, (AN0)
0001 = channel 1, (AN1)
0010 = channel 2, (AN2)
0011 = channel 3, (AN3)
0100 = channel 4, (AN4)
0101 = channel 5, (AN5)
0110 = channel 6, (AN6)
0111 = channel 7, (AN7)
1000 = channel 8, (AN8)
1001 = channel 9, (AN9)
1010 = channel 10, (AN10)
1011 = channel 11, (AN11)
11xx = 予備、選択しない。

bit 3: **Unimplemented**: 未使用：‘0’としてリード

bit 2: **GO/$\overline{\text{DONE}}$:A/D 変換ステータスビット**

ADON = 1 の場合
1 = A/D 変換進行中 (このビットをセットすると A/D 変換が始動し、A/D 変換が完了すると、このビットは自動的にハードウェアによりクリアされる)。
0 = A/D 変換は進行していない。

bit 1: **Unimplemented**: 未使用：‘0’としてリード

bit 0: **ADON**: A/D オンビット
1 = A/D 変換器モジュールが動作中。
0 = A/D 変換器モジュールがシャットオフされ、動作電流を消費しない。

図 16-2: ADCON1 レジスタ (アドレス 15h、バンク 5)

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADCS1	ADCS0	ADFM	—	PCFG3	PCFG2	PCFG1	PCFG0
bit7							bit0

R = 読み込み可能なビット
W = 書き込み可能なビット
U = 未使用のビット、‘0’としてリード
-n = POR リセットでの値

bit 7-6: **ADCS1:ADCS0**: A/D 変換クロック選択ビット
00 = FOSC/8
01 = FOSC/32
10 = FOSC/64
11 = FRC (内部 RC オシレータから出たクロック)

bit 5: **ADFM**: A/D 結果フォーマット選択
1 = 右揃え。ADRESH の 6 最上位ビットを ‘0’ としてリード。
0 = 左揃え。ADRESL の 6 最下位ビットを ‘0’ としてリード。

bit 4: **Unimplemented:** 未使用 :‘0’ としてリード

bit 3-1: **PCFG3:PCFG1**: A/D ポート設定制御ビット

PCFG3:PCFG1	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
000	A	A	A	A	A	A	A	A	A	A	A	A
001	A	A	A	A	D	A	A	A	A	A	A	A
010	A	A	A	A	D	D	A	A	A	A	A	A
011	A	A	A	A	D	D	D	A	A	A	A	A
100	A	A	A	A	D	D	D	D	A	A	A	A
101	D	A	A	A	D	D	D	D	D	A	A	A
110	D	D	A	A	D	D	D	D	D	D	A	A
111	D	D	D	D	D	D	D	D	D	D	D	D

A = アナログ入力　　D = ディジタル I/O

bit 0: **PCFG0**: A/D 基準電圧選択ビット
1 = A/D 基準が VREF+ と VREF- ピン。
0 = A/D 基準が AVDD と AVSS。
注 : このビットをセットする場合、A/D 基準電圧の仕様と合っていることを確認のこと。

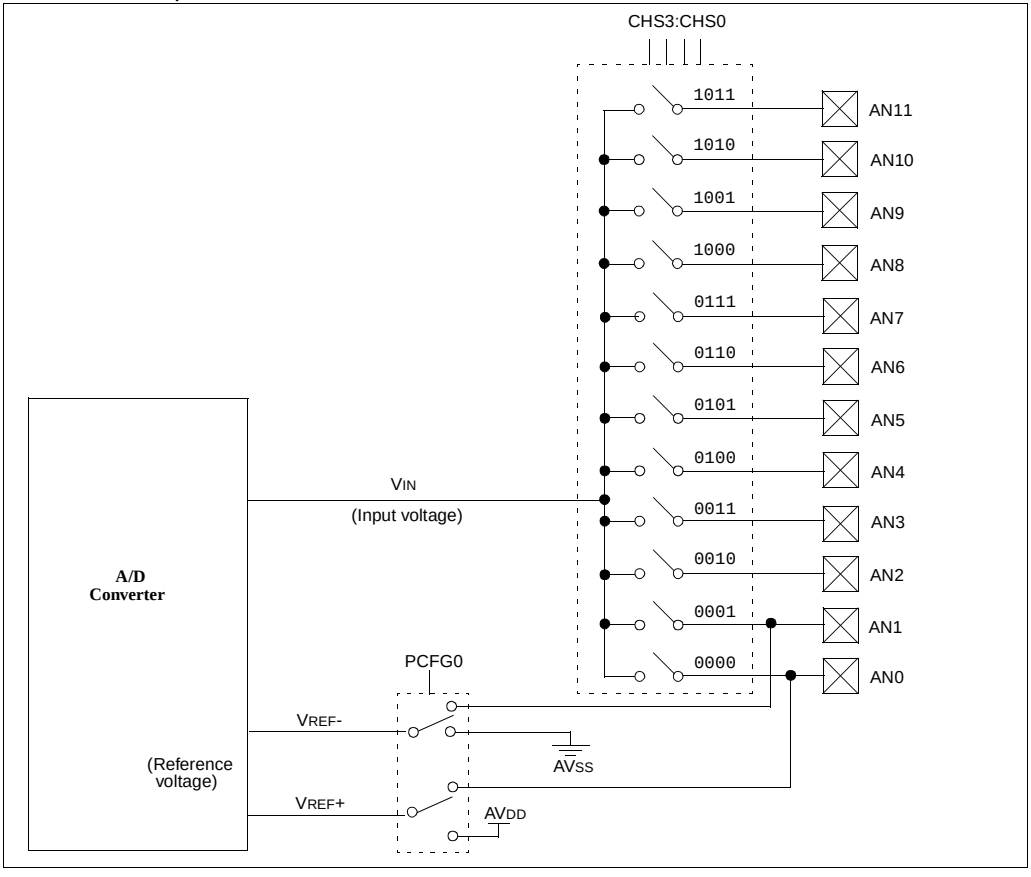
ADRESH:ADRESL レジスタは、10 ビットの A/D 変換の結果を含んでいます。A/D 変換が完了すると、結果はこの A/D 結果レジスタペアにロードされ、GO/DONE ビット (ADCON0<2>) がクリアされ、A/D 割込みフラグビット ADIF がセットされます。A/D モジュールのブロック図を図 16-3 に示します。

A/D モジュールを必要により設定した後、変換が始まる前に選択したチャンネルを取得する必要があります。アナログ入力チャンネルは、対応する DDR ビットを入力として選択する必要があります。取得時間を決めるためには、16-1 章を参照してください。この取得時間が経過した後、A/D 変換を始動することができます。A/D 変換は次のように進めます。:

1. **A/D モジュールの設定 :**
   - **アナログピン / 電圧基準 / ディジタル I/O(ADCON1) を設定**
   - **A/D 入力チャンネルを選択 (ADCON0)**
   - **A/D 変換クロックを選択 (ADCON0)**
   - **A/D モジュールをオン (ADCON0)**
2. **A/D 割込みの設定 (必要な場合):**
   - **ADIF ビットをクリア**
   - **ADIE ビットをセット**
   - **GLINTD ビットをクリア**
3. **必要とされる取得時間を待つ。**
4. **変換を始動 :**
   - **GO/DONE ビットをセット (ADCON0)**
5. **下記のいずれかにより A/D 変換が完了するのを待つ :**
   - **GO/DONE ビットがクリアされるためのポーリング**

   または
   - **A/D 割込みを待つ**
6. **A/D 結果レジスタペア (ADRESH:ADRESL) を読み込み、必要ならビット ADIF をクリアする。**
7. **次の変換のために、必要ならステップ 1 または 2 に戻る。ビットごとの A/D 変換時間は TAD として定義される。2TAD の最少のウェイト時間が、次の取得が始まる前に要求されます。**

図 16-3: A/D ブロック図



16.1 A/D 取得リクエスト

A/D 変換器の設定された精度を満たすためには、チャージ保持キャパシタ(CHOLD)が入力チャンネル電圧レベルに完全に充電されなければなりません。図 16-4 にアナログ入力モデルを示します。電源インピーダンス (RS) と内部サンプリングスイッチインピーダンス (RSS) は直接キャパシタ CHOLD を充電するために必要な時間に影響します。図 16-4 に示すように、サンプリングスイッチインピーダンス (RSS) はデバイス電圧 (VDD) で変わります。電源のインピーダンスはアナログ入力でのオフセット電圧に影響します(ピンからの漏れ電流による)。**アナログの電源インピーダンスは最大 10kΩ にしてください。**アナログ入力チャンネルを選択した (変化した) 後、この取得は変換が始動する前に行われなければなりません。

最小の取得時間を計算するためには、方程式 16-1 を使用します。この方程式は 1/2LSb エラー以内で取得時間を計算します (A/D に関して 1024 ステップ)。1/2LSb エラーは、設定された精度を満たすために A/D に許されている最大のエラーです。

式 16-1: A/D の最小充電時間（C HOLD)

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \cdot (1 - e^{(-Tcap/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$

given $V_{HOLD} = (V_{REF}/2048)$, for 1/2 LSb resolution

$V_{REF} = V_{REF+} - V_{REF-}$

or

$$Tcap = -(200\ pF)(1\ k\Omega + R_{SS} + R_S)\ \ln(1/2047)$$

例 16-1 は必要とされる最小の取得時間 TACQ の計算を示しています。この計算は次のような応用システムの仮定に基づいています。

CHOLD = 200 pF

RS = 10 kΩ

1/2 LSb error

VDD = 5V → RSS = 7 kΩ

Temp (application system max.) = 50°C

VHOLD = 0 @ t = 0

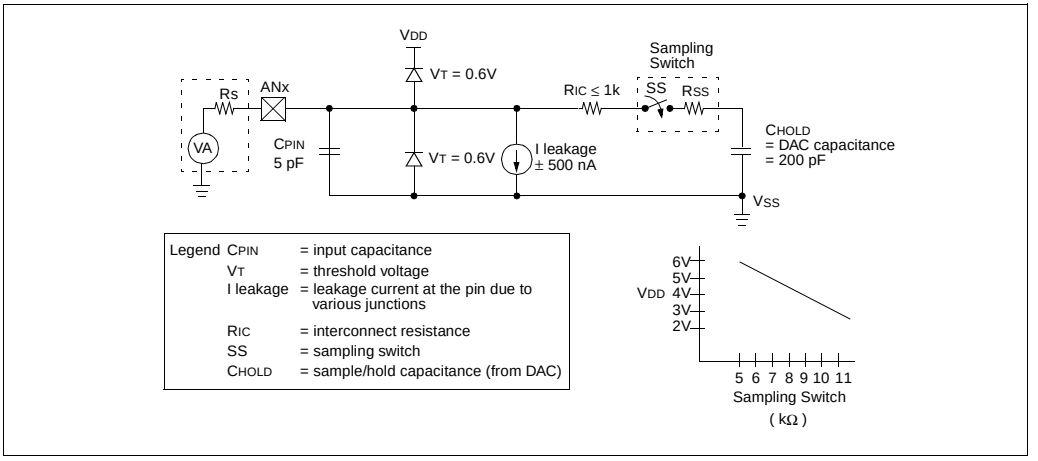
注 1: 基準電圧 (VREF) はそれ自体の出力をキャンセルするので、方程式に影響を与えません。

注 2: チャージ保持キャパシタ (CHOLD) はそれぞれの変換後放電しません。

注 3: アナログ電源のインピーダンスは最大 10kΩ にしてください。これはピンからの漏れ設定を考慮する上で重要です。

注 4: 変換が完了した後、2.0TAD 遅延時間は取得が再び始まる前にとらなければなりません。この時間中、保持キャパシタは選択された A/D 入力チャンネルに接続されていません。

図 16-4: アナログ入力モデル



例 16-1: 必要とされる最小取得時間の計算

TACQ =Amplifier Settling Time +
Holding Capacitor Charging Time +
Temperature Coefficient)

) Only required for temperatures ≠ 25°C

TACQ =10 µs + Tcap + [(Temp - 25°C)(0.05 µs/°C)]

TCAP =-CHOLD (RIC + RSS + RS) ln(1/2047)
-200 pF (1 kΩ + 7 kΩ + 10 kΩ) ln(0.0004885)
-200 pF (18 kΩ) ln(0.0004885)
-3.6 µs (-7.6241)
27.447 µs

TACQ =10 µs + 27.447 µs + [(50°C - 25°C)(0.05 µs/°C)]
37.447 µs + 1.25 µs
38.697 µs

16.2 A/D 変換クロックの選択

ビットごとの A/D 変換時間は TAD として定義されています。A/D 変換は 10 ビット変換につき最小 12TAD 必要です。A/D 変換クロックソースはソフトウェアで選択されます。TAD には次のような 4 個のオプションが使用できます。

- 8TOSC
- 32TOSC
- 64TOSC
- 内部 RC オシレータ

正しい A/D 変換のためには、1.6 µs の最小の TAD 時間を確実にするために A/D 変換クロック (TAD) を選択する必要があります。

表 16-1 と 16-2 に、デバイスの動作中の周波数と選択された A/D クロックソースから求めた TAD 時間を示します。これらのタイマは標準の電圧範囲のデバイス用です。

表 16-1: TAD 対デバイスの動作周波数 (標準デバイス (c))

AD クロックソース (TAD)		デバイスの周波数				
Operation	ADCS1:ADCS0	33 MHz	20 MHz	5 MHz	1.25 MHz	333.33 kHz
8TOSC	00	242 ns(2)	400 ns(2)	1.6 µs	6.4 µs	24 µs
32TOSC	01	970 ns(2)	1.6 µs	6.4 µs	25.6 µs(3)	96 µs(3)
64TOSC	10	1.94 µs	3.2 µs	12.8 µs(3)	51.2 µs(3)	192 µs(3)
RC	11	2 - 6 µs(1, 4)	2 - 6 µs(1, 4)	2 - 6 µs(1, 4)	2 - 6 µs(1, 4)	2 - 6 µs(1)

凡例 : 網掛けの部分は推奨範囲外です。
注 1: RC ソースには 4 µs の標準的な TAD 時間があります。
2: これらの値は必要とされる最小の TAD 時間に違反します。
3: 変換時間をより速くするためには、別のクロックソースを選ぶことをお勧めします。
4: デバイスの周波数が 1MHz 以上の場合、RC の A/D 変換クロックソースは、スリープ動作用だけにお勧めします。

表 16-2: TAD 対デバイスの動作周波数 (拡張電圧デバイス (LC))

AD クロックソース (TAD)		デバイスの周波数				
Operation	ADCS1:ADCS0	8 MHz	4 MHz	2 MHz	1 MHz	333.33 kHz
8TOSC	00	1.0 µs(2)	2.0 µs(2)	4 µs	8 µs	24 µs
32TOSC	01	4.0 µs	8 µs	16 µs	32 µs(3)	96 µs(3)
64TOSC	10	8.0 µs	16 µs	32 µs(3)	64 µs(3)	192 µs(3)
RC	11	3 - 9 µs(1, 4)	3 - 9 µs(1, 4)	3 - 9 µs(1, 4)	3 - 9 µs(1)	3 - 9 µs(1)

凡例 : 網掛けの部分は推奨範囲外です。
注 1: RC ソースには 4 µs の標準的な TAD 時間があります。
2: これらの値は必要とされる最小の TAD 時間に違反します。
3: 変換時間をより速くするためには、別のクロックソースを選ぶことをお勧めします。
4: デバイスの周波数が 1MHz 以上の場合、RC の A/D 変換クロックソースは、スリープ動作用だけにお勧めします。

16.3 アナログポートピンの設定

ADCON1 と DDR レジスタは A/D ポートピンの動作を制御します。アナログ入力として要求されるポートピンは、対応する DDR ビットをセット (入力に) しなければなりません。DDR ビットがクリア (出力に) されると、ディジタル出力レベル (VOH または VOL) は変換されます。

A/D 動作は CHS2:CHS0 ビットと DDR ビットの状態から独立しています。

注 1: ポートレジスタを読み込む時、アナログ入力チャンネルとして設定されたピンはどれも、クリアされたものとして読み込まれます (Low レベル)。ディジタル入力として設定されたピンは、アナログ入力を変換します。ディジタルに設定された入力でのアナログレベルは、変換の精度には影響しません。

注 2: ディジタル入力として定義されたピン (AN11:AN0 ピンを含む) のアナログレベルにより、入力バッファがデバイスの仕様を越えた電流を消費することがあります。

16.4 A/D 変換

例 16-2 に、A/D 変換の実行方法を示します。PORTF と下位の 4 個の PORTG ピンはアナログ入力として設定されます。アナログ基準 (VREF+ と VREF-) は、デバイス電圧 AVDD と AVSS です。A/D 割込みはイネーブルされ、A/D 変換クロックは FRC です。その変換は RG3/AN0 ピン (チャンネル 0) 上で実行されます。

注意 : GO/$\overline{\text{DONE}}$ ビットはA/Dをオンする同じ命令でセットしてはいけません。

変換中に GO/$\overline{\text{DONE}}$ ビットをクリアすると現在の変換を打ち切ります。A/D 結果レジスタペアは部分的に完了した A/D 変換のサンプルではアップデートされません。すなわち ADRESH:ADRESL レジスタは最後に完了した変換の値を持ち続けます (または ADRESH:ADRESL レジスタに書き込まれた最後の値)。A/D 変換が打ち切られた後、次の取得が始動する前に 2TAD ウェイトが必要です。この 2TAD ウェイトの後、選ばれたチャンネル上の取得は自動的に始動します。

例 16-2: A/D 変換

```
     MOVLB   5                ; Bank 5
     CLRF    ADCON1, F        ; Configure A/D inputs
     MOVLW   0xC1             ; RC Clock, A/D is on, Channel 0 is selected
     MOVWF   ADCON0           ;
     MOVLB   4                ; Bank 4
     BCF     PIR2, ADIF       ; Clear A/D interrupt flag bit
     BSF     PIE2, ADIE       ; Enable A/D interrupts
     BSF     INTSTA, PEIE     ; Enable peripheral interrupts
     BCF     CPUSTA, GLINTD   ; Enable all interrupts
;
;  Ensure that the required sampling time for the selected input channel has elapsed.
;  Then the conversion may be started.
;
     MOVLB   5                ; Bank 5
     BSF     ADCON0, GO       ; Start A/D Conversion
       :                      ;   The ADIF bit will be set and the GO/DONE bit
       :                      ;   is cleared upon completion of the A/D Conversion
```

16.4.1 A/D 結果レジスタ

ADRESH:ADRESL レジスタペアは 10 ビットの A/D 結果が A/D 変換の完了でロードされる位置にあります。レジスタペアは 16 ビット幅です。A/D モジュールは、16 ビット結果レジスタに 10 ビットの結果を左揃えか右揃えに調整するという融通がききます。A/D フォーマット選択ビット (ADFM) はこの調整を制御します。図 16-5 に、A/D 結果調整の動作を示します。余分なビットには ' 0 ' をロードします。A/D 結果がこれらのロケーションをオーバーライトしない (A/D がディセーブル) 時は、これらのレジスタは 2 つの汎用 8 ビットレジスタとして使用することもできます。

16.5 スリープ中の A/D 動作

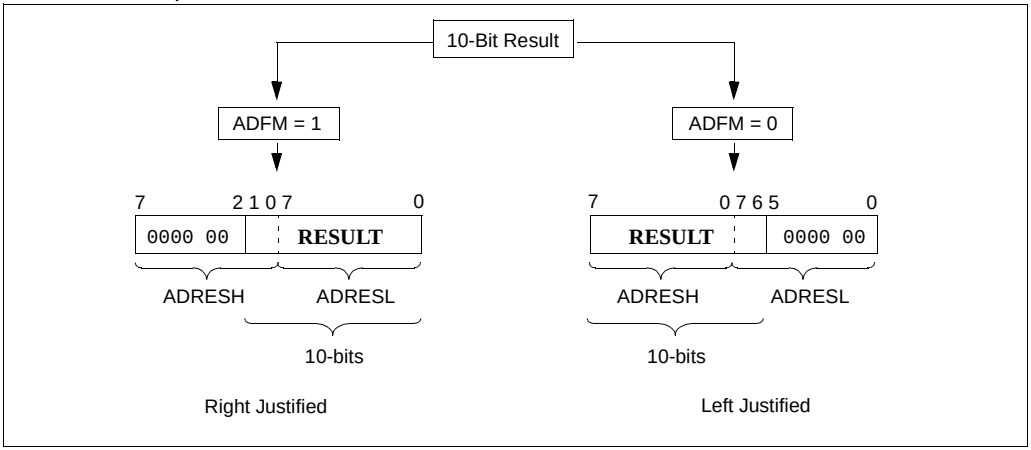
A/D モジュールは SLEEP モード中に動作することができます。これは A/D クロックソースを RC(ADCS1:ADCS0=11) にセットすることが要求されます。RC クロックソースが選ばれると、A/D モジュールは変換を始める前に 1 つ命令サイクルを待ちます。これにより SLEEP 命令が実行されるのを可能にし、変換からすべてのディジタルスイッチングノイズを消します。変換が完了すると、GO/$\overline{\text{DONE}}$ ビットがクリアされ、その結果が ADRES レジスタにロードされます。A/D 割込みがイネーブルされると、デバイスは SLEEP からウェークアップします。A/D 割込みがイネーブルされない場合、ADON ビットはセットされたままですが、A/D モジュールはオフになります。

A/D クロックソースが別のクロックオプション (RC ではなく) の時は、ADON ビットはセットされたままですが、SLEEP 命令により現在の変換が打ち切られ、A/D モジュールがオフになります。

A/D のオフは、A/D モジュールを最も低い電流消費状態にすることです。

注意 : A/D モジュールが SLEEP で動作するためには、A/D クロックソースが RC(ADCS1:ADCS0=11) にセットされる必要があります。SLEEP 中に起こる変換を可能にするには、SLEEP 命令が GO/$\overline{\text{DONE}}$ ビットをセットする命令に続かなければいけません。

図 16-5: A/D 結果調整



16.6 A/D の精度 / エラー

A/D 変換器に設定された絶対精度は、量子化エラー、積分エラー、微分エラー、フルスケールエラー、オフセットエラー、単純エラーのすべての寄与の合計を含みます。それは、どのコードでも実際の変化から理想的な変化に対して最大偏差として設定されています。A/D 変換器の絶対エラーは、V_{DD}=V_{REF}(デバイスの設定された動作範囲にわたって) に対して < ± 1LSb で設定されています。しかし A/D 変換器の精度は、V_{DD} が V_{REF} から分岐するので下がります。

アナログ入力の与えられた範囲に対して、出力ディジタルコードは同じです。これはディジタルコードへのアナログ入力の量子化によります。量子化エラーは一般に± 1/2LSb で、アナログからディジタルへの変換過程では本質的です。量子化エラーを減らす唯一の方法は、A/D 変換器の分解能を増やすことです。

オフセットエラーは、コードの最初の理想的な変化に対して、コードの最初の実際の変化を測定します。オフセットエラーは伝達関数全体をシフトします。オフセットエラーはシステム外での校正や、アナログ入力での全体のリーク電流とソースインピダンスの相互作用によりシステムに導入することも可能です。

ゲインエラーは、すぐ前の実際の変化とオフセットエラーを調整したすぐ前の理想的な変化の最大偏差を測定します。このエラーは伝達関数の傾きの変化のように見えます。ゲインエラーとフルスケールエラーの差は、フルスケールはオフセットエラーを考慮に入れないということです。ゲインエラーはソフトウェアで校正できます。

直線性エラーはコード変化の一様性に関連します。直線性エラーはシステム外で校正することはできません。積分の非直線性エラーは、各コードのゲインエラーを調整した理想的なコード変化に対して実際のコード変化を測定します。

微分の非直線性は、理想的なコード幅に対して実際の最大コード幅を測定します。この測定は調整されません。

ピンの最大リーク電流は± 1μA です。

デバイス周波数が低いシステムでは、A/D RC クロックの使用を選択した方が良いでしょう。高い周波数では、T_{AD} はデバイスオシレータから引き出さなければなりません。T_{AD} は最小を妨げることなく、選ばれた動作に対して≦ 8μs である必要があります。これは、T_{AD} が Tosc から出ている時、チップ上のクロックの位相変化からはずれているためです。これは広範囲にディジタルスイッチングノイズの影響を減らします。これは RC からのクロックでは不可能です。ディジタルスイッチングノイズによる精度の損失は、多くの I/O ピンが動作中に大きくなります。

デバイスが A/D 変換の始動後 SLEEP モードに入るシステムでは、RC クロックソースの選択が必要です。このモードではSLEEP中のモジュールからのディジタルノイズは停止します。この方法により高精度となります。

16.7 RESET の影響

デバイスのリセットにより、すべてのレジスタがリセット状態になります。これにより A/D モジュールはオフになり、どんな変換も打ち切られます。ADRESH:ADRESL レジスタにある値はパワーオンリセットに関しては変更されません。ADRESH:ADRESL レジスタはパワーオンリセット後の不明のデータを含みます。

16.8 接続の考察

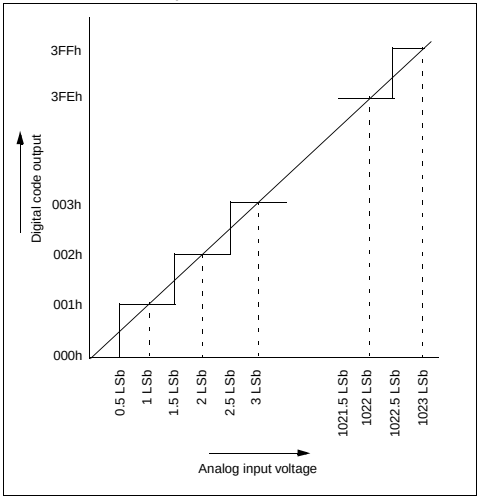
入力電圧がレール値 (V_{SS}-0.3V または V_{DD}+0.3V) を越える場合、変換の精度は保証範囲外です。

外部 RC フィルタは時々入力シグナルのアンチ・エイリアシングのために付け加えられます。R コンポーネントは、信号源インピーダンスが推奨値 10kΩ 以下となるような値を選択することを推奨します。アナログ入力ピン (ハイインピーダンス) に接続された外部のコンポーネント (キャパシタ、ツェナーダイオードなど) は、ピンのもとでの漏れ電流が非常に僅かでなければいけません。

16.9 伝達関数

A/D 変換器の伝達関数は、次の通りです。最初の変化は、アナログ入力電圧 (V_{AIN}) がアナログ V_{REF} /1024 と等しい時に起こります (図 16-6)。

図 16-6: A/D 伝達関数



16.10 参考文献

A/D 変換器を理解するための参考文献として、“ アナログディジタル変換ハンドブック ” 第三版 (Prentice Hall 著・ISBN 0-13-2848-0) があります。

図 16-7: A/D 動作のフローチャート

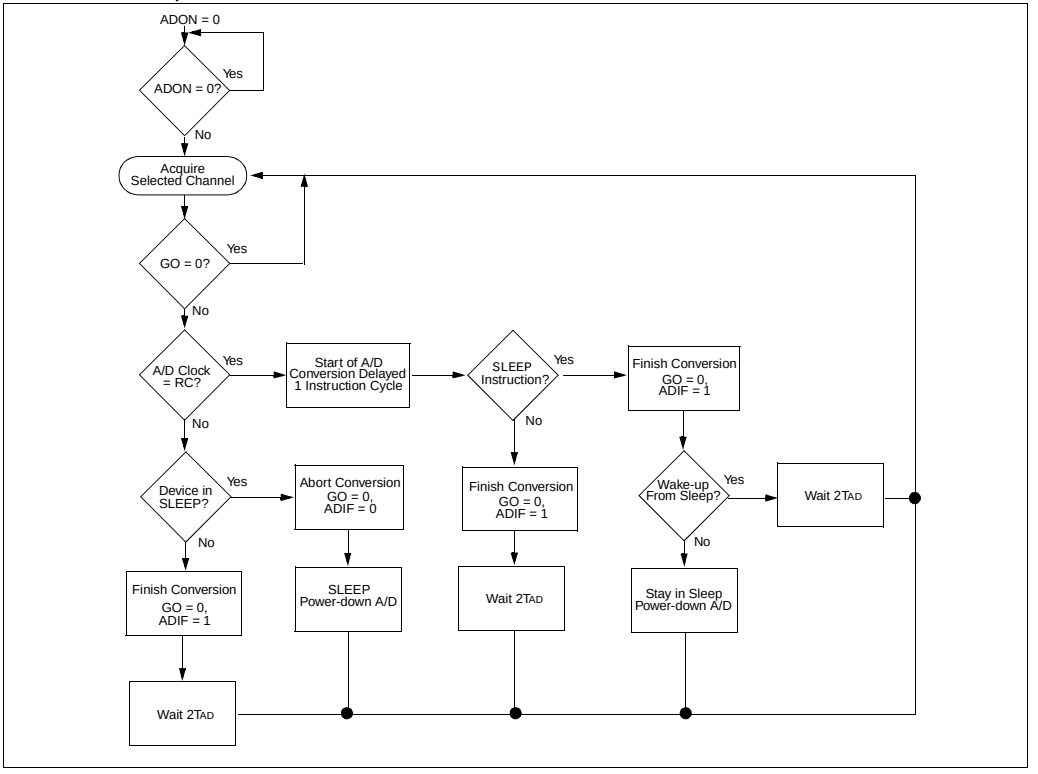


表 16-3: A/D に関連するレジスタ / ビット

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR, BOR での値	他のリセットでの値 (注 1)
06h, unbanked	CPUSTA	—	—	STAKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	--11 1100	--11 qq11
07h, unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
10h, Bank 4	PIR2	SSPIF	BCLIF	ADIF	—	CA4IF	CA3IF	TX2IF	RC2IF	000- 0010	000- 0010
11h, Bank 4	PIE2	SSPIE	BCLIE	ADIE	—	CA4IE	CA3IE	TX2IE	RC2IE	000- 0000	000- 0000
10h, Bank 5	**DDRF**	**PORTF のデータ方向指示レジスタ**								1111 1111	1111 1111
11h, Bank 5	**PORTF**	**RF7/ AN11**	**RF6/ AN10**	**RF5/ AN9**	**RF4/ AN8**	**RF3/ AN7**	**RF2/ AN6**	**RF1/ AN5**	**RF0/ AN4**	0000 0000	0000 0000
12h, Bank 5	**DDRG**	**PORTG のデータ方向指示レジスタ**								1111 1111	1111 1111
13h, Bank 5	**PORTG**	**RG7/ TX2/CK2**	**RG6/ RX2/DT2**	**RG5/ PWM3**	**RG4/ CAP3**	**RG3/ AN0/VREF+**	**RG2/ AN1/VREF-**	**RG1/ AN2**	**RG0/ AN3**	xxxx 0000	uuuu 0000
14h, Bank 5	**ADCON0**	**CHS3**	**CHS2**	**CHS1**	**CHS0**	**ム**	**GO/DONE**	**ム**	**ADON**	0000 -0-0	0000 -0-0
15h, Bank 5	**ADCON1**	**ADCS1**	**ADCS0**	**ADFM**	**ム**	**PCFG3**	**PCFG2**	**PCFG1**	**PCFG0**	000- 0000	000- 0000
16h, Bank 5	**ADRESL**	**A/D 結果 ローレジスタ**								xxxx xxxx	uuuu uuuu
17h, Bank 5	**ADRESH**	**A/D 結果 ハイレジスタ**								xxxx xxxx	uuuu uuuu

凡例 : x= 不定、u= 不変、— = 未使用、‘ 0 ’としてリード。網掛けの部分は A/D 変換では使われません。

注 1: 他の (non power-up) リセットは、$\overline{MCLR}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。

NOTES:

17.0 CPU の特殊機能

マイクロコントローラが他のプロセッサと違うところは、リアルタイムアプリケーションを必要とする事柄を取り扱える特殊回路である点です。PIC17CXXX ファミリには、システムの信頼性を最大限に高め、外部部品を無くしてコストを最小限にし、消費電力節減動作モードを備え、コードを保護するために考えられた次のような多くの機能があります。

- **オシレータ選択 (4.0 章)**
- **リセット (5.0 章)**
- **パワーオンリセット (POR)**
- **パワーアップタイマ (PWRT)**
- **オシレータスタートアップタイマ (OST)**
- **ブラウンアウトリセット (BOR)**
- **インターラプト (6.0 章)**
- **ウォッチドッグタイマ (WDT)**
- **SLEEP モード**
- **コード保護**

PIC17CXXX には EPROM ビットによってのみ停止できるウォッチドッグタイマがあります。それは信頼性を上げるために、専用の RC オシレータで動作しています。パワーアップ時に必要な遅延時間を作るために 2 個のタイマがあります。1 つはオシレータスタートアップタイマ (OST) で、クリスタルオシレータが安定するまでチップを RESET 状態に保つよう考えられています。もう 1 つはパワーアップタイマ (PWRT) で、パワーアップ時にのみ 96ms(公称) の一定遅延時間を作り、電源が安定するまでパーツを RESET 状態に保つよう設計されています。チップに内蔵されたこの 2 個のタイマにより、ほとんどのアプリケーションで外部リセット回路の必要がありません。

SLEEP モードは非常に少ない電流のパワーダウンモードを実現できるように設計されています。外部リセット、ウォッチドッグタイマのリセット、割込みにより SLEEP からウェークすることができます。いくつかのオシレータのオプションによっても、そのパーツをアプリケーションに適応させることができます。RC オシレータのオプションによりシステムコストを節約し、LF クリスタルにより電力を節約します。コンフィギュレーションビットはいろいろなオプションを選択できます。このコンフィギュレーションワードには、図 17-1 に示すようなフォーマットがあります。

図 17-1: コンフィギュレーションワード

U - x	R/P - 1	R/P - 1	U - x	U - x	U - x	U - x	U - x	U - x	High (H) Table Read Addr.
—	PM2	BODEN	—	—	—	—	—	—	FE0Fh - FE08h
bit15 bit 8	bit 7							bit 0	

U - x	U - x	R/P - 1	U - x	R/P - 1	R/P - 1	R/P - 1	R/P - 1	R/P - 1	Low (L) Table Read Addr.
—	—	PM1	—	PM0	WDTPS1	WDTPS0	FOSC1	FOSC0	FE07h - FE00h
bit15 bit 8	bit 7							bit 0	

bit 6H **BODEN**: ブラウンアウト検出イネーブル
1 = ブラウンアウト検出回路はイネーブル
0 = ブラウンアウト検出回路はディセーブル

bits 7H:6L:4L **PM2, PM1, PM0**, プロセッサモード選択ビット
111 = マイクロプロセッサモード
110 = マイクロコントローラモード
101 = 拡張マイクロコントローラモード
000 = コード保護マイクロコントローラモード

bits 2L:3L **WDTPS1:WDTPS0**, WDT ポストスケーラ選択ビット
11 = WDT はイネーブル、ポストスケーラ =1
10 = WDT はイネーブル、ポストスケーラ =256
01 = WDT はイネーブル、ポストスケーラ =64
00 = WDT はディセーブル、16 ビットオーバーフロータイマ

bits 1L:0L **FOSC1:FOSC0**, オシレータ選択ビット
11 = EC オシレータ
10 = XT オシレータ
01 = RC オシレータ
00 = LF オシレータ

— **予備**

17.1 コンフィギュレーションビット

PIC17CXXX には 8 つのコンフィギュレーションロケーションがあります (表 17-1)。いろいろなデバイスの設定を選択するために、これらのロケーションをプログラムした状態 (‘ 0 ’ としてリード)、またはプログラムしないままの状態 (‘ 1 ’ としてリード) にできます。データにかかわらず、コンフィギュレーションロケーションへのライトはどれも、そのコンフィギュレーションビットをプログラムします。TABLWT 命令はプログラムメモリロケーションへのライトが必要です。コンフィギュレーションビットは TABLRD 命令を使用することによりリードが可能です。FE00h と FE07h 間のどのコンフィギュレーションロケーションをリードしても、TABLATL レジスタのコンフィギュレーションワード (図 17-1) の下位バイトを読み込みます。TABLATL レジスタは FFh にあります。FE08h と FE0Fh 間のコンフィギュレーションロケーションをリードすると、TABLATL レジスタのコンフィギュレーションワードの上位バイトを読み込みます。TABLATH レジスタは FFh にあります。

FE00h から FE0Fh のアドレスは、マイクロコントローラモードとコード保護マイクロコントローラモード用のプログラムメモリの範囲内のみです。デバイスのプログラマは、どのプロセッサモードでもコンフィギュレーションワードのリードが可能です。詳細については、プログラミングの仕様をご覧ください。

表 17-1: コンフィギュレーションロケーション

ビット	アドレス
FOSC0	FE00h
FOSC1	FE01h
WDTPS0	FE02h
WDTPS1	FE03h
PM0	FE04h
PM1	FE06h
BODEN	FE0Eh
PM2	FE0Fh

注意 : 必要なコンフィギュレーションロケーションをプログラミングする時は、昇順でプログラムする必要があります。スタートは FE00h のアドレスです。

17.2 オシレータの設定

17.2.1 オシレータの種類

PIC17CXXX は 4 種類の異なったオシレータのモードで動作できます。次の 4 種類のモードから 1 つを選択するために、2 個のコンフィギュレーションビット (FOSC1:FOSC0) をプログラミングできます。

- **LF ローパワークリスタル**
- **XT クリスタル / レゾネータ**
- **EC 外部クロック入力**
- **RC 抵抗 / 容量**

異なったオシレータの種類の情報やそれらの使用方法については、4.0 章を参照してください。

17.3 ウォッチドッグタイマ (WDT)

ウォッチドッグタイマの機能は、ソフトウェアの機能不全からリカバーすることです。WDT は、そのクロックソース用に内部フリーランニングオンチップ RC オシレータを使用します。これには外部部品は不要です。この RC オシレータは、OSC1/CLKIN ピンの RC オシレータとは別のものです。つまり、OSC1/CLKIN と OSC2/CLKOUT ピンのクロックが、たとえば SLEEP 命令の実行により停止した場合でも、WDT は動作します。通常動作中や SLEEP モード中に、WDT タイムアウトが発生するとデバイスを RESET します。コンフィギュレーションビット WDTPS1:WDTPS0 を ' 00 ' としてプログラムすることで、永久的に WDT をディセーブルできます (17.1 章参照)。

通常動作の下では、WDT を標準のインターバルでクリアする必要があります。この時間は最小の WDT オーバーフロータイムより少ないです。この時間の枠内で WDT をクリアしないと、WDT がオーバーフローしデバイスをリセットします。

17.3.1 WDT の周期

公称 (ポストスケーラ =1 の時) での WDT のタイムアウト周期は 12ms です。このタイムアウト周期は温度、V_{DD}、製品毎の製造プロセスのばらつきで異なります (DC スペック参照)。もっと長いタイムアウト周期を必要とする場合は、1:256 までの分周比率のポストスケーラを WDT に設定することができます。これにより、一般的には 3.0 秒までのタイムアウト周期を実現できます。

CLRWDT と SLEEP 命令は、WDT とポストスケーラ (WDT に割り当てられている場合) をクリアし、タイムアウトからデバイスの RESET 状態が発生することを防止します。

WDT のタイムアウトにより CPUSTA レジスタの $\overline{TO}$ ビットはクリアされます。

17.3.2 WDT とポストスケーラのクリア

WDT とポストスケーラは次の場合にクリアされます。

- **デバイスがリセット状態の場合。**
- **SLEEP 命令を実行した場合。**
- **CLRWDT 命令を実行した場合。**
- **割込みによる SLEEP からのウェークアップの場合。**

WDT カウンタ / ポストスケーラはデバイスがリセット状態になった後、最初のエッジでカウントを始めます。

17.3.3 WDT プログラミング上の注意

最悪条件 (V_{DD}= 最小、温度 = 最大、最大 WDT ポストスケーラ) の下では、WDT のタイムアウトが起こるまで数秒かかる場合があることを計算にいれる必要があります。

WDT とポストスケーラは、パワーオンリセットのシーケンスの間パワーアップタイマです。

17.3.4 通常のタイマとしての WDT

WDT が通常のタイマとして選択される場合、クロックソースはデバイスのクロックです。WDT もポストスケーラも直接読み込みも書き込みもできません。オーバーフロータイムは 65536 Tosc サイクルです。オーバーフローでは、$\overline{TO}$ ビットはクリアされます (デバイスはリセットされません)。CLRWDT 命令は $\overline{TO}$ ビットをセットするのに使用することができます。これにより WDT は単純なオーバーフロータイマになることが可能です。この単純なタイマはスリープ時にインクリメントしません。

図 17-2: ウォッチドッグタイマのブロック図

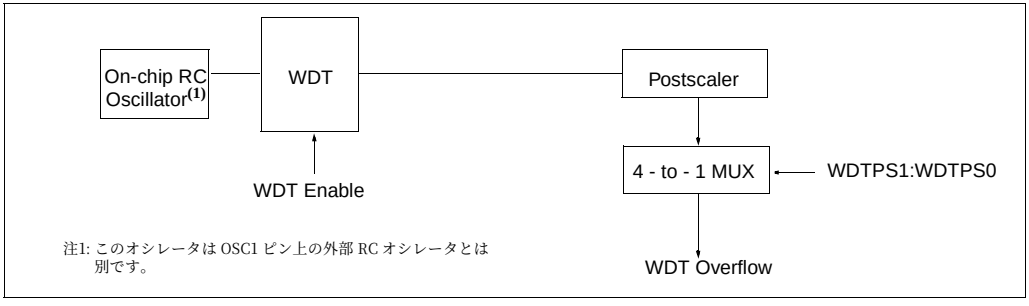


表 17-2: ウォッチドッグタイマに関連するレジスタ / ビット

アドレス	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR, BOR での値	他のリセットでの値 (注 1)
—	Config	コンフィギュレーションワードの WDTPSx のロケーションについては図 17-1 参照								(注 2)	(注 2)
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	--11 1100	--11 qq11

凡例 : — = 未使用、' 0 ' としてリード、q= 条件によって決まる値。網掛けの部分は WDT では使われません。
注 1: 他の (non power-up) リセットは、$\overline{MCLR}$ を通る外部リセットとウォッチドッグタイマリセットを含みます。
2: この値はデバイスをプログラムした場合で、プログラムしない場合はすべて ' 1 ' としてリードします。

17.4 パワーダウンモード (SLEEP)

SLEEP 命令を実行すると、パワーダウンモードに入ります。これはウォッチドッグタイマとポストスケーラをクリアします (イネーブルの場合)。$\overline{PD}$ ビットはクリア、$\overline{TO}$ ビットはセットされます (CPUSTA レジスタで)。SLEEP モードでは、オシレータのドライバはオフになります。I/O ポートはその状態を維持します (ハイ、ローを出力、または高インピーダンス)。

$\overline{MCLR}$/Vpp ピンは論理ハイレベル (VIHMC) にする必要があります。WDT タイムアウト RESET は、$\overline{MCLR}$/Vpp ピンをローでは駆動しません。

17.4.1 SLEEP からのウェークアップ

デバイスは次にあげるイベントによって SLEEP からウェークアップできます。

- **POR**
- **$\overline{MCLR}$/Vpp ピンに外部リセットを入力した時。**
- **WDT のリセット (WDT がイネーブル時)**
- **BOR**
- **RA0/INT ピンからの入力、RB ポート変化、T0CKI の割込み、いくつかの周辺割込み。**

次にあげる周辺割込みが、デバイスを SLEEP からウェークできます。

- **キャプチャ割込み**
- **USART 同期スレーブ送信割込み**
- **USART 同期スレーブ受信割込み**
- **A/D 変換の完了**
- **SPI スレーブ送信 / 受信の完了**
- **I^2C スレーブ受信**

SLEEP 中は内蔵された Q クロックが与えられないので、他の周辺機器は割込みを発生できません。

リセットイベントはどれもデバイスのリセットを引き起こします。割込みイベントはどれもプログラム実行の継続とみなされます。CPUSTA レジスタの $\overline{TO}$ と $\overline{PD}$ ビットは、デバイスのリセットの原因を決めるのに使用できます。$\overline{PD}$ ビットはパワーアップでセットされ、SLEEP を起動するとクリアされます。WDT のタイムアウトが起こった (さらにウェークアップが起こった) 場合、$\overline{TO}$ ビットはクリアされます。

SLEEP 命令実行中の時、次の命令 (PC+1) がプリフェッチされます。割込みによりデバイスをウェークアップさせるために、対応する割込みイネーブルビットをセット (イネーブルに) する必要があります。ウェークアップは GLINTD ビットの状態とは無関係です。GLINTD ビットがセット (ディセーブル) されている場合、デバイスは SLEEP 命令後の命令を実行し続けます。GLINTD ビットがクリア (イネーブル) されている場合、デバイスは SLEEP 命令後の命令を実行し、割込みベクタアドレスへ分岐します。SLEEP の後の命令の実行が影響する場合は、SLEEP 命令の後に NOP を挿入する必要があります。

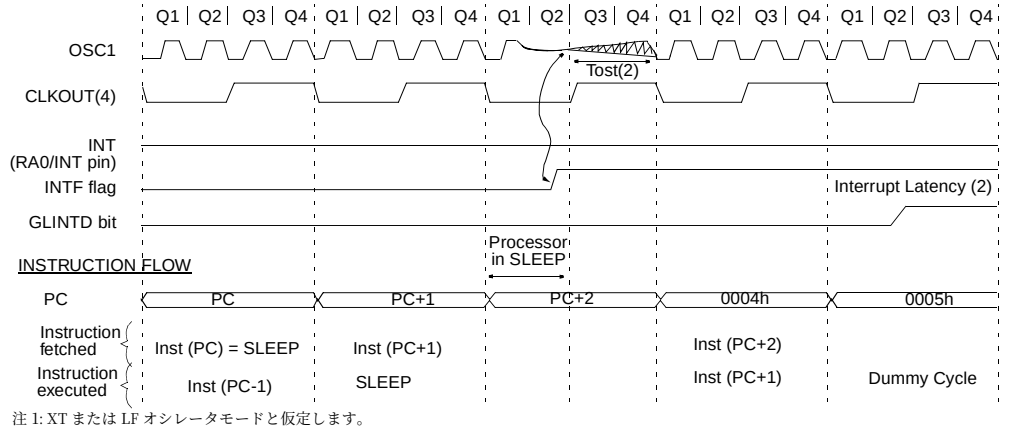
注意 : 全インターラプトがディセーブル (GLINTD をセット) でありながら、割込み要因として 1 つでも割込みイネーブルビットと対応する割込みフラグビットの両方をセットされている場合、デバイスはただちに SLEEP からウェークアップします。$\overline{TO}$ ビットはセットされ、$\overline{PD}$ ビットはクリアされます。

デバイスが SLEEP からウェークする時、ウェークアップの要因に関係なく、WDT がクリアされます。

17.4.1.1 ウェークアップの遅延時間

オシレータの種類が XT または LF モードで設定されている時、オシレータのスタートアップタイマ (OST) は、ウェークアップで作動します。OST は 1024Tosc のリセットでデバイスを保持します。これは SLEEP からの割込み応答時間を考慮に入れて計算する必要があります。

図 17-3: 割込みによる SLEEP からのウェークアップ



注 1: XT または LF オシレータモードと仮定します。
2: Tost=1024Tosc(図は実測ではありません)。この遅延時間は、RCosc モードではそこで発生しません。
3: GLINTD=0 の時、プロセッサがウェークアップ後、割込みルーチンにジャンプします。GLINTD=1 の場合、実行はライン内で継続します。
4: CLKOUT はこれらの osc モードでは機能しませんが、この図では参考のためにそのタイミングを示します。

17.4.2 最小消費電流

消費電流を最小にするためには、すべての I/O ピンは VDD、Vss のどちらかで、外部回路は I/O ピンからの電流を引き込まないようにしなければなりません。フロート入力による電流のスイッチングを避けるために、ハイインピーダンス入力である I/O ピンを外部でハイかローに引っ張る必要があります。T0CKI 入力は VDD か Vss にあるべきです。PORTB の内蔵されたプルアップからの負担も考慮に入れる必要があり、可能ならディセーブルにします。

17.5 コードプロテクション

プログラムメモリでのコードは、コードプロテクトモード (PM2:PM0='000') のマイクロコントローラを選択することにより保護が可能です。

このモードでは、内蔵されたプログラムメモリ空間にある命令はプログラムメモリのリードまたはライトを継続できます。内部プログラムメモリの範囲外で実行される命令は、プログラムメモリへのライトもプログラムメモリからのリードも禁止されます。

注 : Microchip 社は、窓付きデバイスでのコードプロテクトは保証いたしません。

コードプロテクトビットをプログラムしない場合、プログラミングの検証のために、内蔵されたプログラムメモリの内容を読み出すことができます。

17.6 インサーキットシリアルプログラミング

ハイエンドファミリ (PIC17CXXX) の PIC17C75X グループは、最終アプリケーション回路の場合でもシリアルプログラミングを可能にする機能が加わりました。これは簡単で、クロックとデータの 2 本のラインと、電源、グランド、プログラミング電圧の 3 本のラインで実現できます。これによりプログラミングされていないデバイスでボードを組み立てておき、製品の出荷直前にマイクロコントローラをプログラミングすることができます。またこれにより、最新のファームウエアやプログラムするための特別仕様のファームウエアが可能になります。

デバイスは、依頼があれば単独製品や“スペシャル”の変形製品を作るようシリーズ化されており、コードのアップデートも可能です。これによりデザインの柔軟性が増します。

デバイスをシリアルプログラミングテストモードに置くためには、2 個のピンを V_{IHH} に置く必要があります。これらは TEST ピンと $\overline{MCLR}/V_{PP}$ です。イベントのシーケンスは次のようにしなければなりません。

1. **TEST ピンを V_{IHH} に置きます。**
2. **$\overline{MCLR}/V_{PP}$ ピンを V_{IHH} に置きます。**

ステップ 1 とステップ 2 の間にセットアップタイムが必要です。

このシーケンスの後、プログラムカウンタはプログラムメモリアドレス 0xFF60 を示します。このロケーションはブート ROM にあります。コードは USART/SCI を初期化し、それでコマンドを受信できます。このためにデバイスをクロックする必要があります。このモードでのデバイスのクロックソースは、RA1/T0CKI ピンです。USART/SCI の初期化を可能にするために遅延した後で、コマンドは受信ができます。これら 3 ステップのフローを示します。

1. **デバイスのクロックソースが始まります。**
2. **USART/SCI を設定するためにブート ROM コード用の 80 個のデバイスクロックを待ちます。**
3. **コマンドを送れます。**

シリアルプログラミングの詳細については、PIC17C75X の Programming Specification をご参照ください (お近くの Microchip technology 販売事務所へご連絡ください)。

図 17-4: 標準的なインサーキットシリアルプログラミングの接続図

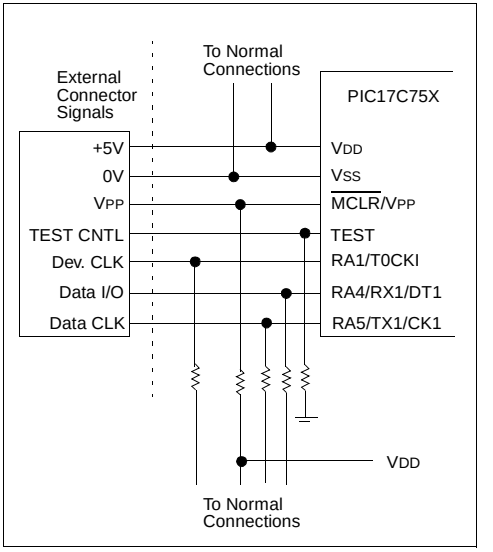


表 17-3: ISP インターフェイスピン

	During Programming		
Name	Function	Type	Description
RA4/RX1/DT1	DT	I/O	Serial Data
RA5/TX1/CK1	CK	I	Serial Clock
RA1/T0CKI	OSCI	I	Device Clock Source
TEST	TEST	I	Test mode selection control input. Force to V_{IHH},
$\overline{MCLR}/V_{PP}$	$\overline{MCLR}/V_{PP}$	P	Master Clear reset and Device Programming Voltage
V_{DD}	V_{DD}	P	Positive supply for logic and I/O pins
V_{SS}	V_{SS}	P	Ground reference for logic and I/O pins

18.0 命令セットの概要

PIC17CXXX の命令セットは 58 の命令から成っています。各命令は 16bit ワードで、OPCODE と、1 個または 2 個以上のオペランドに分かれています。オペコードは命令の種類を指定し、オペランドは命令の実行方法を示します。PIC17CXXX の命令セットは次の 3 つに分類することができます。:

- **バイト操作**
- **ビット操作**
- **リテラルおよびコントロール操作**

これらのフォーマットを図 18-1 に示します。

表 18-1 には、オペコードのフィールドの説明をまとめました。これらの説明は、表 18-2 や特定の命令でのオペコードを理解するのに役立ちます。

バイト操作命令:‘ f ’をファイルレジスタのデジネータ、‘ d ’をディスティネションのデジネータとして使います。ファイルレジスタデジネータでは、命令で使用するファイルレジスタを指定します。

ディスティネションデジネータでは、命令の実行結果を格納する場所を指定します。‘ d ’=0 の時、結果は WREG レジスタに格納されます。‘ d ’=1 の時、結果は命令で指定されたファイルレジスタに格納されます。

ビット操作命令:ビットフィールドデジネータ‘ b ’を使って、この命令実行により影響を受けるビットの番号を選択します。また、ファイルレジスタデジネータ‘ f ’を使って、そのビットが置かれているファイルレジスタの番号を指定します。

リテラル及びコントロール操作:‘ k ’を使って、8 ビットまたは13ビットの定数やリテラル値を指定します。

命令セットは非常に直接的で、次のように分類されます。:

- **バイト対応の命令**
- **ビット対応の命令**
- **リテラル及びコントロール命令**

すべての命令は、下記を除いて 1 つのシングル命令サイクルの中で実行されます。:

- **条件付きテストの結果が真である場合。**
- **命令を実行した結果として、プログラムカウンタが変化した場合。**
- **テーブルリードまたはテーブルライト命令を実行した場合 (この場合、その命令の実行に 2 命令サイクルかかり、2 番目のサイクルは NOP として実行されます)。**

1 命令サイクルは 4 つのオシレータ周期から成ります。したがって、25MHz のオシレータ周波数では、通常の命令実行時間は 160μs です。条件付きテストの結果が真であったり、命令を実行した結果プログラムカウンタが変化する場合は、命令実行時間は320μsです。

表 18-1: オペコードフィールドの説明

フィールド	説明
f	レジスタファイルのアドレス (00h ～ FFh))
p	ペリフェラルレジスタファイルのアドレス (00h ～ 1Fh)
i	テーブルポインタ制御 i = ‘ 0 ’ (変化せず) i = ‘ 1 ’ (命令実行後インクリメント)
t	テーブルバイト選択 t= ‘ 0 ’ (下位バイトで動作) t= ‘ 1 ’ (上位バイトのリテラルフィールドで動作、定数データ)
WREG	ワーキングレジスタ (アキュームレータ)
b	8bit ファイルレジスタ内のビットアドレス
k	リテラルフィールド、定数データまたはラベル
x	無効ロケーション (= ‘ 0 ’ または ‘ 1 ’) アセンブラは x= ‘ 0 ’ を伴うコードを生成。これは Microchip 社のあらゆるソフトウエアツールとの互換性を確保するために推奨されている形式です。
d	ディスティネション選択 0= 結果を WREG に格納 1= 結果をファイルレジスタ f に格納 デフォルトは d= ‘ 1 ’
u	使用せず、‘ 0 ’ としてコード化
s	ディスティネション選択 0= 結果をファイルレジスタ f と WREG に格納 1= 結果をファイルレジスタ f に格納 デフォルトは s= ‘ 1 ’
label	ラベル名
C,DC, Z,OV	ALU ステータスビット (キャリ、ディジットキャリ、ゼロ、オーバーフロー)
GLINTD	グローバル割込みディセーブルビット (CPUSTA<4>)
TBLPTR	テーブルポインタ (16bit)
TBLAT	テーブルラッチ (16bit) は上位バイト (TBLATH) と下位バイト (TBLATL) から成る
TBLATL	テーブルラッチは下位バイト
TBLATH	テーブルラッチは上位バイト
TOS	最上位スタック
PC	プログラムカウンタ
BSR	バンク選択レジスタ
WDT	ウォッチドッグタイマ / カウンタ
$\overline{TO}$	タイムアウトビット
$\overline{PD}$	パワーダウンビット
dest	ディスティネション (WREG レジスタまたは指定レジスタファイルのロケーション)
[ ]	オプション
( )	内容
→	割当て先
< >	レジスタビットフィールド
∈	セットを表わす
italics	ユーザ定義用語 (フォントはクーリエ)

表18-2にMPASMアセンブラが認識する命令のリストを示します。

注意 1: 使用しないオペコードはどれも予備です。予備のオペコードを使用すると、予期しない動作をすることがあります。

命令の例はすべて、次のフォーマットで16進数を表わします。

0xhh

hは16進数を表わします。

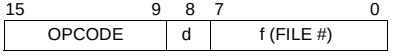
2進数を表わすためには、

0000 0100b

bは2進数であることを示します。

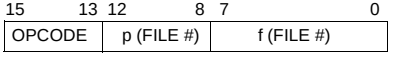
図 18-1: 命令の一般的なフォーマット

Byte-oriented file register operations



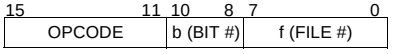
d = 0 for destination WREG
d = 1 for destination f
f = 8-bit file register address

Byte to Byte move operations



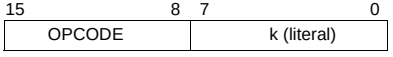
p = peripheral register file address
f = 8-bit file register address

Bit-oriented file register operations



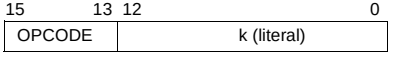
b = 3-bit address
f = 8-bit file register address

Literal and control operations



k = 8-bit immediate value

CALL and GOTO operations



k = 13-bit immediate value

18.1 ソース / ディスティネションとしての特殊機能レジスタ

PIC17C75X の直交の命令セットにより、特殊機能レジスタを含むすべてのファイルレジスタのリードとライトが可能ですが、注意すべきいくつかの状態があります。:

18.1.1 ディスティネションとしての ALUSTA

1つの命令が ALUSTA にライトすると、Z、C、DC、OV ビットが命令の結果としてセットまたはクリアされたり、書き込まれた元のデータビットをオーバーライトすることがあります。例えば CLRF ALUSTA を実行すると、レジスタ ALUSTA をクリアしてからレジスタの 0000 0100b を残しながら Z ビットをセットします。

18.1.2 ソースまたはディスティネションとしての PCL

PCL でのリード、ライト、リード・モディファイ・ライトは次のような結果になることがあります。:

Read PC:	PCH → PCLATH; PCL → dest
Write PCL:	PCLATH → PCH; 8-bit destination value → PCL
Read-Modify-Write:	PCL→ ALU operand PCLATH → PCH; 8-bit result → PCL

PCH= プログラムカウンタの上位バイト (アドレス可能のレジスタではない)、PCLATH= プログラムカウンタのハイ保持ラッチ、dest= ディスティネション、WREG、f です。

18.1.3 ビット操作

すべてのビット操作命令は、最初にレジスタ全体をリードし、選択されたビット上で動作し、結果を書き戻すことにより行ないます (リード・モディファイ・ライト (R-M-W))。ポートのような特殊機能レジスタで動作する時は、このことに注意が必要です。

注意 : デバイスにより操作されたステータスビットは (割込みフラグビットを含む)、Q1 サイクルでセットまたはクリアされます。それで、これらのビットを含むレジスタでの R-M-W 命令を行なうことに問題はありません。

18.2 クロックサイクルの活動

各命令サイクル (Tcy) は、4 つのクロックサイクル (Q1-Q4) から成ります。クロックサイクルはデバイスのオシレータサイクル (Tosc) と同じです。クロックサイクルは、各命令サイクルのデコード、リード、データ処理、ライトなどにタイミング / デジネーションを与えます。次の図はクロックサイクルと命令サイクルの関係を示します。

命令サイクル (Tcy) を作る 4 つのクロックサイクルは次のようにまとめることができます。:

Q1: 命令デコードサイクルまたは無動作状態。

Q2: 命令リードサイクルまたは無動作状態。

Q3: データを処理。

Q4: 命令ライトサイクルまたは無動作状態。

各命令は、その命令の詳細なクロックサイクル動作を示します。

図 18-2: クロックサイクルの動作

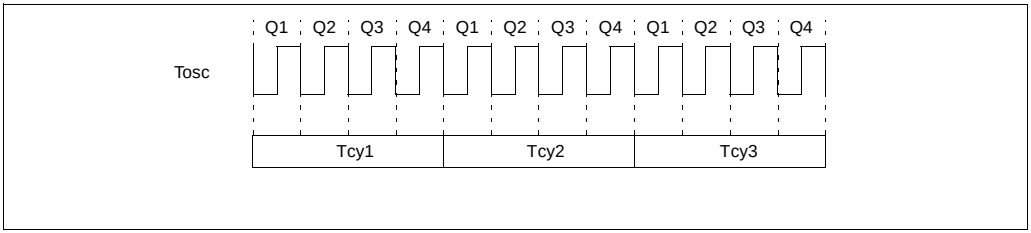


表 18-2: PIC17CXXX の命令セット

ニモニック オペランド		説明	サイクル	16-bit Opcode MSb LSb	影響を受けるステータス	注意事項
バイト対応のファイルレジスタ命令						
ADDWF	**f,d**	ADD WREG to f	1	0000 111d ffff ffff	OV,C,DC,Z	
ADDWFC	**f,d**	ADD WREG and Carry bit to f	1	0001 000d ffff ffff	OV,C,DC,Z	
ANDWF	**f,d**	AND WREG with f	1	0000 101d ffff ffff	Z	
CLRF	**f,s**	Clear f, or Clear f and Clear WREG	1	0010 100s ffff ffff	None	3
COMF	**f,d**	Complement f	1	0001 001d ffff ffff	Z	
CPFSEQ	**f**	Compare f with WREG, skip if f = WREG	1 (2)	0011 0001 ffff ffff	None	6,8
CPFSGT	**f**	Compare f with WREG, skip if f > WREG	1 (2)	0011 0010 ffff ffff	None	2,6,8
CPFSLT	**f**	Compare f with WREG, skip if f < WREG	1 (2)	0011 0000 ffff ffff	None	2,6,8
DAW	**f,s**	Decimal Adjust WREG Register	1	0010 111s ffff ffff	C	3
DECF	**f,d**	Decrement f	1	0000 011d ffff ffff	OV,C,DC,Z	
DECFSZ	**f,d**	Decrement f, skip if 0	1 (2)	0001 011d ffff ffff	None	6,8
DCFSNZ	**f,d**	Decrement f, skip if not 0	1 (2)	0010 011d ffff ffff	None	6,8
INCF	**f,d**	Increment f	1	0001 010d ffff ffff	OV,C,DC,Z	
INCFSZ	**f,d**	Increment f, skip if 0	1 (2)	0001 111d ffff ffff	None	6,8
INFSNZ	**f,d**	Increment f, skip if not 0	1 (2)	0010 010d ffff ffff	None	6,8
IORWF	**f,d**	Inclusive OR WREG with f	1	0000 100d ffff ffff	Z	
MOVFP	**f,p**	Move f to p	1	011p pppp ffff ffff	None	
MOVPF	**p,f**	Move p to f	1	010p pppp ffff ffff	Z	
MOVWF	**f**	Move WREG to f	1	0000 0001 ffff ffff	None	
MULWF	**f**	Multiply WREG with f	1	0011 0100 ffff ffff	None	
NEGW	**f,s**	Negate WREG	1	0010 110s ffff ffff	OV,C,DC,Z	1,3
NOP	**—**	No Operation	1	0000 0000 0000 0000	None	
RLCF	**f,d**	Rotate left f through Carry	1	0001 101d ffff ffff	C	
RLNCF	**f,d**	Rotate left f (no carry)	1	0010 001d ffff ffff	None	
RRCF	**f,d**	Rotate right f through Carry	1	0001 100d ffff ffff	C	
RRNCF	**f,d**	Rotate right f (no carry)	1	0010 000d ffff ffff	None	
SETF	**f,s**	Set f	1	0010 101s ffff ffff	None	3
SUBWF	**f,d**	Subtract WREG from f	1	0000 010d ffff ffff	OV,C,DC,Z	1
SUBWFB	**f,d**	Subtract WREG from f with Borrow	1	0000 001d ffff ffff	OV,C,DC,Z	1
SWAPF	**f,d**	Swap f	1	0001 110d ffff ffff	None	
TABLRD	**t,i,f**	Table Read	2 (3)	1010 10ti ffff ffff	None	7

凡例： オペコードフィールドの説明は表 18-1 を参照。

注 1: 2 の補数演算。

2: 符号無し演算。

3: s=‘1’の場合、ファイルのみが影響を受け、s=‘0’の場合、WREG レジスタとファイルの両方が影響を受けます。ワーキングレジスタ (WREG) のみが影響を受けることが必要とされる場合、f=WREG を設定する必要があります。

4: LCALL の間、PCLATH の内容は PC の MSB にロードされ、kkkk kkkk は PC(PCL) の LSB にロードされます。

5: テーブルポインタが内部の EPROM を選択する時、EPROM の書き込みには複数サイクルの命令になります。その命令は割込みイベントを発生して終了します。外部のプログラムメモリにライトする時は、2 サイクル命令です。

6: 条件が真の時は 2 サイクル命令で、その他はシングルサイクル命令です。

7: PCL(プログラムカウンタがローバイト) への TABLRD の場合は 3 サイクルが必要ですが、それ以外は 2 サイクル命令です。

8: “skip”は、現在の命令の実行中にフェッチされた命令が実行されず、代りに NOP が実行されたことを意味します。

表 18-2: PIC17CXXX の命令セット （続き）

ニモニック オペランド		説明	サイクル	16-bit Opcode MSb LSb	影響を受けるステータス	注意事項
TABLWT	**t,i,f**	Table Write	2	1010 11ti ffff ffff	None	5
TLRD	**t,f**	Table Latch Read	1	1010 00tx ffff ffff	None	
TLWT	**t,f**	Table Latch Write	1	1010 01tx ffff ffff	None	
TSTFSZ	**f**	Test f, skip if 0	1 (2)	0011 0011 ffff ffff	None	6,8
XORWF	**f,d**	Exclusive OR WREG with f	1	0000 110d ffff ffff	Z	
ビット対応のファイルレジスタ命令						
BCF	**f,b**	Bit Clear f	1	1000 1bbb ffff ffff	None	
BSF	**f,b**	Bit Set f	1	1000 0bbb ffff ffff	None	
BTFSC	**f,b**	Bit test, skip if clear	1 (2)	1001 1bbb ffff ffff	None	6,8
BTFSS	**f,b**	Bit test, skip if set	1 (2)	1001 0bbb ffff ffff	None	6,8
BTG	**f,b**	Bit Toggle f	1	0011 1bbb ffff ffff	None	
リテラルおよびコントロール命令						
ADDLW	**k**	ADD literal to WREG	1	1011 0001 kkkk kkkk	OV,C,DC,Z	
ANDLW	**k**	AND literal with WREG	1	1011 0101 kkkk kkkk	Z	
CALL	**k**	Subroutine Call	2	111k kkkk kkkk kkkk	None	7
CLRWDT	**—**	Clear Watchdog Timer	1	0000 0000 0000 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	**k**	Unconditional Branch	2	110k kkkk kkkk kkkk	None	7
IORLW	**k**	Inclusive OR literal with WREG	1	1011 0011 kkkk kkkk	Z	
LCALL	**k**	Long Call	2	1011 0111 kkkk kkkk	None	4,7
MOVLB	**k**	Move literal to low nibble in BSR	1	1011 1000 uuuu kkkk	None	
MOVLR	**k**	Move literal to high nibble in BSR	1	1011 101x kkkk uuuu	None	
MOVLW	**k**	Move literal to WREG	1	1011 0000 kkkk kkkk	None	
MULLW	**k**	Multiply literal with WREG	1	1011 1100 kkkk kkkk	None	
RETFIE	**—**	Return from interrupt (and enable interrupts)	2	0000 0000 0000 0101	GLINTD	7
RETLW	**k**	Return literal to WREG	2	1011 0110 kkkk kkkk	None	7
RETURN	**—**	Return from subroutine	2	0000 0000 0000 0010	None	7
SLEEP	**—**	Enter SLEEP Mode	1	0000 0000 0000 0011	$\overline{TO}$, $\overline{PD}$	
SUBLW	**k**	Subtract WREG from literal	1	1011 0010 kkkk kkkk	OV,C,DC,Z	
XORLW	**k**	Exclusive OR literal with WREG	1	1011 0100 kkkk kkkk	Z	

凡例： オペコードフィールドの説明は表 18-1 を参照。

注 1: 2 の補数演算。

2: 符号無し演算。

3: s=‘1’の場合、ファイルのみが影響を受け、s=‘0’の場合、WREG レジスタとファイルの両方が影響を受けます。ワーキングレジスタ (WREG) のみが影響を受けることが必要とされる場合、f=WREG を設定する必要があります。

4: LCALL の間、PCLATH の内容は PC の MSB にロードされ、kkkk kkkk は PC(PCL) の LSB にロードされます。

5: テーブルポインタが内部の EPROM を選択する時、EPROM の書き込みには複数サイクルの命令になります。その命令は割込みイベントを発生して終了します。外部のプログラムメモリにライトする時は、2 サイクル命令です。

6: 条件が真の時は 2 サイクル命令で、その他はシングルサイクル命令です。

7: PCL(プログラムカウンタがローバイト) への TABLRD の場合は 3 サイクルが必要ですが、それ以外は 2 サイクル命令です。

8: “skip”は、現在の命令の実行中にフェッチされた命令が実行されず、代りに NOP が実行されたことを意味します。

ADDLW **ADD Literal to WREG**

Syntax: [*label*] ADDLW k

Operands: 0 ≤ k ≤ 255

Operation: (WREG) + k → (WREG)

Status Affected: OV, C, DC, Z

Encoding:

1011	0001	kkkk	kkkk

Description: WREG の内容を 8bit のリテラル ‘ k ’ に加え、その結果を WREG に格納します。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example: ADDLW 0x15

命令実行前
WREG = 0x10

命令実行後
WREG = 0x25

ADDWF **ADD WREG to f**

Syntax: [*label*] ADDWF f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

Operation: (WREG) + (f) → (dest)

Status Affected: OV, C, DC, Z

Encoding:

0000	111d	ffff	ffff

Description: WREG の内容をレジスタ ‘ f ’ の内容と加算し、その結果を ‘ d ’ =0 であれば WREG に、‘ d ’ =1 であれば、レジスタ ‘ f ’ に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: ADDWF REG, 0

命令実行前
WREG = 0x17
REG = 0xC2

命令実行後
WREG = 0xD9
REG = 0xC2

ADDWFC **ADD WREG and Carry bit to f**

Syntax: [*label*] ADDWFC f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

Operation: (WREG) + (f) + C → (dest)

Status Affected: OV, C, DC, Z

Encoding:

0001	000d	ffff	ffff

Description: WREG の内容と、キャリフラグ、レジスタ ' f ' の内容を加算し、その結果を ' d ' =0 であれば WREG に、' d '=1 であれば、データメモリロケーション ' f ' に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: ADDWFC REG 0

命令実行前
Carry bit = 1
REG = 0x02
WREG = 0x4D

命令実行後
Carry bit = 0
REG = 0x02
WREG = 0x50

ANDLW **And Literal with WREG**

Syntax: [*label*] ANDLW k

Operands: 0 ≤ k ≤ 255

Operation: (WREG) .AND. (k) → (WREG)

Status Affected: Z

Encoding:

1011	0101	kkkk	kkkk

Description: WREG の内容と 8bit のリテラル ' k ' の AND を取り、その結果を WREG に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example: ANDLW 0x5F

命令実行前
WREG = 0xA3

命令実行後
WREG = 0x03

ANDWF **AND WREG with f**

Syntax: [*label*] ANDWF f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

Operation: (WREG) .AND. (f) → (dest)

Status Affected: Z

Encoding:

0000	101d	ffff	ffff

Description: WREG の内容とレジスタ ‘ f ’ の内容との AND を取り、その結果を ‘ d ’ =0 であれば WREG に、‘ d ’ =1 であれば、レジスタ ‘ f ’ に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `ANDWF REG, 1`

命令実行前
WREG = 0x17
REG = 0xC2

命令実行後
WREG = 0x17
REG = 0x02

BCF **Bit Clear f**

Syntax: [*label*] BCF f,b

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7

Operation: 0 → (f<b>)

Status Affected: None

Encoding:

1000	1bbb	ffff	ffff

Description: レジスタ ‘ f ’ のビット ‘ b ’ の内容を 0 にクリアします。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `BCF FLAG_REG, 7`

命令実行前
FLAG_REG = 0xC7

命令実行後
FLAG_REG = 0x47

BSF **Bit Set f**

Syntax: [*label*] BSF f,b

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7

Operation: 1 → (f<b>)

Status Affected: None

Encoding:

1000	0bbb	ffff	ffff

Description: レジスタ ' f ' のビット ' b ' の内容を 1 にセットします。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `BSF FLAG_REG, 7`

命令実行前
FLAG_REG= 0x0A

命令実行後
FLAG_REG= 0x8A

BTFSC **Bit Test, skip if Clear**

Syntax: [*label*] BTFSC f,b

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

1001	1bbb	ffff	ffff

Description: レジスタ ' f ' のビット ' b ' の内容が 0 であれば、次の命令をスキップします。
ビット ' b ' が 0 であれば、すでにフェッチされている次の命令が廃棄され、これを 2 サイクル命令にするために NOP を実行します。

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example:
```
HERE    BTFSC   FLAG,1
FALSE   :
TRUE    :
```

命令実行前
PC = address (HERE)

命令実行後
If FLAG<1> = 0;
PC = address (TRUE)
If FLAG<1> = 1;
PC = address (FALSE)

BTFSS **Bit Test, skip if Set**

Syntax: [*label*] BTFSS f,b

Operands: 0 ≤ f ≤ 127
0 ≤ b < 7

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding:

1001	0bbb	ffff	ffff

Description: レジスタ ' f ' のビット ' b ' の内容が 1 であれば、次の命令をスキップします。
ビット ' b ' が 1 であれば、すでにフェッチされている次の命令が廃棄され、これを 2 サイクル命令にするために NOP を実行します。

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example:
```
HERE    BTFSS   FLAG,1
FALSE   :
TRUE    :
```

命令実行前
PC = address (HERE)

命令実行後
If FLAG<1> = 0;
PC = address (FALSE)
If FLAG<1> = 1;
PC = address (TRUE)

BTG **Bit Toggle f**

Syntax: [*label*] BTG f,b

Operands: 0 ≤ f ≤ 255
0 ≤ b < 7

Operation: $(\overline{f<b>}) \rightarrow (f<b>)$

Status Affected: None

Encoding:

0011	1bbb	ffff	ffff

Description: レジスタ ' f ' のビット ' b ' の内容を反転します。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example:
```
BTG     PORTC,  4
```

命令実行前：
PORTC = 0111 0101 [0x75]

命令実行後：
PORTC = 0110 0101 [0x65]

CALL **Subroutine Call**

Syntax: [*label*] CALL k

Operands: 0 ≤ k ≤ 4095

Operation: PC+ 1→ TOS, k → PC<12:0>, k<12:8> → PCLATH<4:0>; PC<15:13> → PCLATH<7:5>

Status Affected: None

Encoding:

111k	kkkk	kkkk	kkkk

Description: 8K ページ内のサブルーチンコール。まず戻りアドレス (PC+1) をスタックにセーブします。13bit の値を PC ビット <12:0> にロードし、それから PC の上位 8 ビットを PCLATH から読み込みます。CALL は 2 サイクルの命令です。

8K メモリ空間を越えるサブルーチンコールに関しては LCALL を参照。

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>, Push PC to stack	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: HERE CALL THERE

命令実行前
PC = Address(HERE)

命令実行後
PC = Address(THERE)
TOS = Address (HERE + 1)

CLRF **Clear f**

Syntax: [*label*] CLRF f,s

Operands: 0 ≤ f ≤ 255

Operation: 00h → f, s ∈ [0,1]
00h → dest

Status Affected: None

Encoding:

0010	100s	ffff	ffff

Description: 設定されたレジスタの内容をクリアします。

S=0：レジスタ ' f ' と WREG をクリアします。

S=1：レジスタ ' f ' をクリアします。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f' and if specified WREG

Example: CLRF FLAG_REG

命令実行前
FLAG_REG = 0x5A

命令実行後
FLAG_REG = 0x00

CLRWDT **Clear Watchdog Timer**

Syntax: [*label*] CLRWDT

Operands: None

Operation: 00h → WDT
0 → WDT postscaler,
1 → $\overline{TO}$
1 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

0000	0000	0000	0100

Description: CLRWDT 命令によりウォッチドッグタイマをリセットし、さらに WDT のプリスケーラもリセットします。ステータスレジスタの TO と PD をセットします。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	No operation

Example: CLRWDT

命令実行前
WDT counter = ?

命令実行後
WDT counter = 0x00
WDT Postscaler = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

COMF **Complement f**

Syntax: [*label*] COMF f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

Operation: $(\overline{f})$ → (dest)

Status Affected: Z

Encoding:

0001	001d	ffff	ffff

Description: レジスタ ' f ' の内容の補数を取り (内容をすべて反転)、その結果を ' d ' =0 であれば WREG に、' d ' =1 であれば、レジスタ ' f ' に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: COMF REG1,0

命令実行前
REG1 = 0x13

命令実行後
REG1 = 0x13
WREG = 0xEC

CPFSEQ **Compare f with WREG, skip if f = WREG**

Syntax: [*label*] CPFSEQ f

Operands: 0 ≤ f ≤ 255

Operation: (f) – (WREG), skip if (f) = (WREG) (unsigned comparison)

Status Affected: None

Encoding:

0011	0001	ffff	ffff

Description: 符号無し減算を実行することにより、レジスタ‘ f ’の内容と WREG の内容を比較します。
‘ f ’=WREG の状態であれば、次の命令をスキップします。この時すでにフェッチされた命令は廃棄され、これを 2 サイクル命令にするために NOP を実行します。

Words: 1

Cycles: 1 (2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example:
```
HERE    CPFSEQ REG
NEQUAL  :
EQUAL   :
```

命令実行前
PC Address = HERE
WREG = ?
REG = ?

命令実行後
If REG = WREG;
PC = Address (EQUAL)
If REG ≠ WREG;
PC = Address (NEQUAL)

CPFSGT **Compare f with WREG, skip if f > WREG**

Syntax: [*label*] CPFSGT f

Operands: 0 ≤ f ≤ 255

Operation: (f) – (WREG), skip if (f) > (WREG) (unsigned comparison)

Status Affected: None

Encoding:

0011	0010	ffff	ffff

Description: 符号無し減算を実行することにより、レジスタ‘ f ’の内容と WREG の内容を比較します。
‘ f ’の内容が WREG の内容よりも大きければ、次の命令をスキップします。この時すでにフェッチされた命令は廃棄され、これを 2 サイクル命令にするために NOP を実行します。

Words: 1

Cycles: 1 (2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example:
```
HERE      CPFSGT REG
NGREATER  :
GREATER   :
```

命令実行前
PC = Address (HERE)
WREG = ?

命令実行後
If REG > WREG;
PC = Address (GREATER)
If REG ≤ WREG;
PC = Address (NGREATER)

CPFSLT — **Compare f with WREG, skip if f < WREG**

Syntax: [*label*] CPFSLT f

Operands: 0 ≤ f ≤ 255

Operation: (f) – (WREG), skip if (f) < (WREG) (unsigned comparison)

Status Affected: None

Encoding:

0011	0000	ffff	ffff

Description: 符号無し減算を実行することにより、レジスタ‘ f ’の内容と WREG の内容を比較します。
‘ f ’ の内容が WREG の内容よりも小さければ同じです。フェッチされた命令は廃棄され、これを 2 サイクル命令にするために NOP を実行します。

Words: 1

Cycles: 1 (2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example:

```
HERE    CPFSLT REG
NLESS   :
LESS    :
```

命令実行前
PC = Address (HERE)
W = ?

命令実行後
If REG < WREG;
PC = Address (LESS)
If REG ≥ WREG;
PC = Address (NLESS)

DAW — **Decimal Adjust WREG Register**

Syntax: [*label*] DAW f,s

Operands: 0 ≤ f ≤ 255
s ∈ [0,1]

Operation:
If [WREG<3:0> >9] .OR. [DC = 1] then
WREG<3:0> + 6 → f<3:0>, s<3:0>;
else
WREG<3:0> → f<3:0>, s<3:0>;

If [WREG<7:4> >9] .OR. [C = 1] then
WREG<7:4> + 6 → f<7:4>, s<7:4>
else
WREG<7:4> → f<7:4>, s<7:4>

Status Affected: C

Encoding:

0010	111s	ffff	ffff

Description: DAW は、加算命令の結果として反映された WREG の 8bit の値を調整し (BCD フォーマット)、正しい BCD 結果を作ります。
S=0：結果をレジスタ‘ f ’と WREG に書き込みます。
S=1：結果をレジスタ‘ f ’に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f' and other specified register

Example1:

```
DAW   REG1, 0
```

命令実行前
WREG = 0xA5
REG1 = ??
C = 0
DC = 0

命令実行後
WREG = 0x05
REG1 = 0x05
C = 1
DC = 0

Example 2:

命令実行前
WREG = 0xCE
REG1 = ??
C = 0
DC = 0

命令実行後
WREG = 0x24
REG1 = 0x24
C = 1
DC = 0

DECF **Decrement f**

Syntax: [*label*] DECF f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

Operation: (f) – 1 → (dest)

Status Affected: OV, C, DC, Z

Encoding:

0000	011d	ffff	ffff

Description: レジスタ‘ f ’の内容をデクリメントし、その結果を‘ d ’=0 であれば WREG に、‘ d ’=1 であれば、レジスタ‘ f ’に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: DECF CNT, 1

命令実行前
CNT = 0x01
Z = 0

命令実行後
CNT = 0x00
Z = 1

DECFSZ **Decrement f, skip if 0**

Syntax: [*label*] DECFSZ f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

Operation: (f) – 1 → (dest);
skip if result = 0

Status Affected: None

Encoding:

0001	011d	ffff	ffff

Description: レジスタ‘ f ’の内容をデクリメントし、その結果を‘ d ’=0 であれば WREG に、‘ d ’=1 であれば、レジスタ‘ f ’に書き込みます。
結果が 0 の場合は、次の命令をスキップします。この時すでにフェッチされている次の命令を廃棄し、これを 2 サイクル命令にするために NOP を実行します。

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example:
```
HERE    DECFSZ  CNT, 1
        GOTO    LOOP
CONTINUE
```

命令実行前
PC = Address (HERE)

命令実行後
CNT = CNT - 1
If CNT = 0;
PC = Address (CONTINUE)
If CNT ≠ 0;
PC = Address (HERE+1)

DCFSNZ **Decrement f, skip if not 0**

Syntax: [*label*] DCFSNZ f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

Operation: (f) – 1 → (dest);
skip if not 0

Status Affected: None

Encoding:

0010	011d	ffff	ffff

Description: レジスタ‘ f ’の内容をデクリメントし、その結果を ‘ d ’ =0 であれば WREG に、‘ d ’ =1 であれば、レジスタ ‘ f ’ に書き込みます。
結果が 0 でない場合は、次の命令をスキップします。この時すでにフェッチされている次の命令を廃棄し、これを 2 サイクル命令にするために NOP を実行します。

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example:
```
HERE    DCFSNZ  TEMP, 1
ZERO    :
NZERO   :
```

命令実行前
TEMP_VALUE = ?

命令実行後
TEMP_VALUE = TEMP_VALUE - 1,
If TEMP_VALUE = 0;
PC = Address (ZERO)
If TEMP_VALUE ≠ 0;
PC = Address (NZERO)

GOTO **Unconditional Branch**

Syntax: [*label*] GOTO k

Operands: 0 ≤ k ≤ 8191

Operation: k → PC<12:0>;
k<12:8> → PCLATH<4:0>,
PC<15:13> → PCLATH<7:5>

Status Affected: None

Encoding:

110k	kkkk	kkkk	kkkk

Description: GOTO により、8K のページ内ならどこでも無条件の分岐が可能です。13bit の値を PC ビット <12:0> にロードし、次に PC の上位 8 ビットを PCLATH よりロードします。GOTOは常に2サイクル命令です。

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: GOTO THERE

命令実行後
PC = Address (THERE)

INCF **Increment f**

Syntax: [*label*] INCF f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

Operation: (f) + 1 → (dest)

Status Affected: OV, C, DC, Z

Encoding:

0001	010d	ffff	ffff

Description: レジスタ ' f ' の内容をインクリメントし、その結果を ' d ' =0 であれば WREG に、' d ' =1 であれば、レジスタ ' f ' に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:
```
INCF    CNT, 1
```
命令実行前
CNT = 0xFF
Z = 0
C = ?

命令実行後
CNT = 0x00
Z = 1
C = 1

INCFSZ **Increment f, skip if 0**

Syntax: [*label*] INCFSZ f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

Operation: (f) + 1 → (dest)
skip if result = 0

Status Affected: None

Encoding:

0001	111d	ffff	ffff

Description: レジスタ ' f ' の内容をインクリメントし、その結果を ' d ' =0 であれば WREG に、' d ' =1 であれば、レジスタ ' f ' に書き込みます。
結果が 0 の場合は、次の命令をスキップします。この時すでにフェッチされている次の命令を廃棄し、これを 2 サイクル命令にするために NOP を実行します。

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example:
```
HERE    INCFSZ  CNT, 1
NZERO   :
ZERO    :
```
命令実行前
PC = Address (HERE)

命令実行後
CNT = CNT + 1
If CNT = 0;
PC = Address(ZERO)
If CNT ≠ 0;
PC = Address(NZERO)

INFSNZ **Increment f, skip if not 0**

Syntax: [*label*] INFSNZ f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

Operation: (f) + 1 → (dest),
skip if not 0

Status Affected: None

Encoding:

0010	010d	ffff	ffff

Description: レジスタ‘f’の内容をインクリメントし、その結果を‘d’=0 であれば WREG に、‘d’=1 であれば、レジスタ‘f’に書き込みます。
結果が 0 でない場合は、次の命令をスキップします。この時すでにフェッチされている次の命令を廃棄し、これを 2 サイクル命令にするためにNOPを実行します。

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example:
```
HERE    INFSNZ  REG, 1
ZERO
NZERO
```

命令実行前
REG = REG

命令実行後
REG = REG + 1
If REG = 1;
PC = Address (ZERO)
If REG = 0;
PC = Address (NZERO)

IORLW **Inclusive OR Literal with WREG**

Syntax: [*label*] IORLW k

Operands: 0 ≤ k ≤ 255

Operation: (WREG) .OR. (k) → (WREG)

Status Affected: Z

Encoding:

1011	0011	kkkk	kkkk

Description: WREG の内容と 8bit のリテラル‘k’の OR を取り、その結果を WREG に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example: `IORLW 0x35`

命令実行前
WREG = 0x9A

命令実行後
WREG = 0xBF

IORWF **Inclusive OR WREG with f**

Syntax: [*label*] IORWF f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

Operation: (WREG) .OR. (f) → (dest)

Status Affected: $\overline{Z}$

Encoding:

0000	100d	ffff	ffff

Description: WREG とレジスタ ' f ' の OR を取り、その結果を ' d ' =0 であれば WREG に、' d ' =1 であれば、レジスタ ' f ' に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `IORWF RESULT, 0`

命令実行前
RESULT = 0x13
WREG = 0x91

命令実行後
RESULT = 0x13
WREG = 0x93

LCALL **Long Call**

Syntax: [*label*] LCALL k

Operands: $0 \le k \le 255$

Operation: PC + 1 → TOS;
k → PCL, (PCLATH) → PCH

Status Affected: None

Encoding:

1011	0111	kkkk	kkkk

Description: LCALL により、64K のプログラムメモリ空間内ならどこへでも無条件のサブルーチンコールが可能です。まず戻りアドレス (PC+1) をスタックに保存し、次に16bitのジャンプアドレスをプログラムカウンタにロードします。ジャンプアドレスの下位8ビットは命令に指定された値が反映され、PC の上位 8 ビットは PC の上位バイトを保持するメモリである PCLATH からロードされます。

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write register PCL
No operation	No operation	No operation	No operation

Example:
```
MOVLW  HIGH(SUBROUTINE)
MOVPF  WREG, PCLATH
LCALL  LOW(SUBROUTINE)
```

命令実行前
SUBROUTINE = 16-bit Address
PC = ?

命令実行後
PC = Address (SUBROUTINE)

MOVFP **Move f to p**

Syntax: [*label*] MOVFP f,p

Operands: 0 ≤ f ≤ 255
0 ≤ p ≤ 31

Operation: (f) → (p)

Status Affected: None

Encoding:

011p	pppp	ffff	ffff

Description: レジスタ‘ f ’からレジスタ‘ p ’にデータを移動します。ロケーション‘ f ’は256ワードのデータ空間内(00hからFFh)ならどこでも可能で、‘ p ’は 32 バイトのデータ空間内 (oohから1Fh)で可能です。
‘ p ’または‘ f ’のどちらかがWREGを指定してもかまいません。
MOVFPは特に、データメモリロケーションから周辺レジスタ (例えば、送信バッファやI/Oポート) に転送するのに便利です。‘ f ’と‘ p ’両方とも間接アドレシングが可能です。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'p'

Example: `MOVFP REG1, REG2`

命令実行前
REG1 = 0x33,
REG2 = 0x11

命令実行後
REG1 = 0x33,
REG2 = 0x33

MOVLB **Move Literal to low nibble in BSR**

Syntax: [*label*] MOVLB k

Operands: 0 ≤ k ≤ 15

Operation: k → (BSR<3:0>)

Status Affected: None

Encoding:

1011	1000	uuuu	kkkk

Description: 4bitのリテラル‘ k ’をバンクセレクトレジスタ(BSR)にロードします。バンクセレクトレジスタの下位4ビットだけが影響を受けます。BSRの上位半分は不変です。アセンブラは 0 ’として“ u ”フィールドをコード化します。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write literal 'k' to BSR<3:0>

Example: `MOVLB 5`

命令実行前
BSR register = 0x22

命令実行後
BSR register = 0x25 (Bank 5)

MOVLR **Move Literal to high nibble in BSR**

Syntax: [*label*] MOVLR k

Operands: 0 ≤ k ≤ 15

Operation: k → (BSR<7:4>)

Status Affected: None

Encoding:

1011	101x	kkkk	uuuu

Description: 4bit のリテラル‘k’をバンクセレクトレジスタ (BSR) の最上位 4 ビットにロードします。バンクセレクトレジスタの上位 4 ビットだけが影響を受けます。BSR の下位半分は不変です。アセンブラは‘0’として“u”フィールドをコード化します。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write literal 'k' to BSR<7:4>

Example: MOVLR 5

命令実行前
BSR register = 0x22

命令実行後
BSR register = 0x52

MOVLW **Move Literal to WREG**

Syntax: [*label*] MOVLW k

Operands: 0 ≤ k ≤ 255

Operation: k → (WREG)

Status Affected: None

Encoding:

1011	0000	kkkk	kkkk

Description: 8bit のリテラル‘k’を WREG にロードします。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example: MOVLW 0x5A

命令実行後
WREG = 0x5A

MOVPF **Move p to f**

Syntax: [*label*] MOVPF p,f

Operands: 0 ≤ f ≤ 255
0 ≤ p ≤ 31

Operation: (p) → (f)

Status Affected: Z

Encoding:

010p	pppp	ffff	ffff

Description: レジスタ ' p ' からレジスタ ' f ' にデータを移動します。ロケーション ' f ' は 256バイトのデータ空間内(00hからFFh)ならどこでも可能で、' p ' は 32 バイトのデータ空間内 (oohから1Fh)で可能です。
' p ' または ' f ' のどちらかが WREG を指定することができます。
MOVPF は特に、周辺レジスタ (例えば、タイマやI/Oポート) からデータメモリロケーションに転送するのに便利です。' f ' と ' p ' 両方とも間接アドレシングが可能です。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'p'	Process Data	Write register 'f'

Example: MOVPF REG1, REG2

命令実行前
REG1 = 0x11
REG2 = 0x33

命令実行後
REG1 = 0x11
REG2 = 0x11

MOVWF **Move WREG to f**

Syntax: [*label*] MOVWF f

Operands: 0 ≤ f ≤ 255

Operation: (WREG) → (f)

Status Affected: None

Encoding:

0000	0001	ffff	ffff

Description: WREG からレジスタ ' f ' にデータを移動します。ロケーション ' f ' は 256 ワードのデータ空間内ならどこでも可能です。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: MOVWF REG

命令実行前
WREG = 0x4F
REG = 0xFF

命令実行後
WREG = 0x4F
REG = 0x4F

MULLW **Multiply Literal with WREG**

Syntax: [*label*] MULLW k

Operands: 0 ≤ k ≤ 255

Operation: (k x WREG) → PRODH:PRODL

Status Affected: None

Encoding:

1011	1100	kkkk	kkkk

Description: WREG の内容と 8bit のリテラル‘k’の間で、符号無し掛け算を実行します。16bit の結果は PRODH:PRODL レジスタペアに書き込みます。PRODH は上位バイトを含みます。
WREG は不変です。
ステータスフラグはどれも影響を受けません。オーバーフローもキャリもこの操作では不可能なので注意が必要です。0 の結果は可能ですが、検出されません。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL

Example: MULLW 0xC4

命令実行前
WREG = 0xE2
PRODH = ?
PRODL = ?

命令実行後
WREG = 0xC4
PRODH = 0xAD
PRODL = 0x08

MULWF **Multiply WREG with f**

Syntax: [*label*] MULWF f

Operands: 0 ≤ f ≤ 255

Operation: (WREG x f) → PRODH:PRODL

Status Affected: None

Encoding:

0011	0100	ffff	ffff

Description: WREG の内容とレジスタ‘f’の間で、符号無し掛け算を実行します。16 ビットの結果は PRODH:PRODL レジスタペアに書き込みます。PRODH は上位バイトを含みます。
WREG と‘f’両方とも不変です。
ステータスフラグはどれも影響を受けません。オーバーフローもキャリもこの操作では不可能なので注意が必要です。0 の結果は可能ですが、検出されません。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write registers PRODH: PRODL

Example: MULWF REG

命令実行前
WREG = 0xC4
REG = 0xB5
PRODH = ?
PRODL = ?

命令実行後
WREG = 0xC4
REG = 0xB5
PRODH = 0x8A
PRODL = 0x94

NEGW **Negate W**

Syntax: [*label*] NEGW f,s

Operands: $0 \le F \le 255$
$s \in [0,1]$

Operation: $\overline{WREG} + 1 \rightarrow (f)$;
$\overline{WREG} + 1 \rightarrow s$

Status Affected: OV, C, DC, Z

Encoding:

0010	110s	ffff	ffff

Description: WREG を 2 の補数を取ります。‘ s ’ =0 であれば、その結果を WREG とレジスタ‘ f ’ に書き込み、‘ s ’ =1 であれば、レジスタ ‘ f ’ にのみ書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f' and other specified register

Example: NEGW REG,0

命令実行前
WREG = 0011 1010 [0x3A],
REG = 1010 1011 [0xAB]

命令実行後
WREG = 1100 0111 [0xC6]
REG = 1100 0111 [0xC6]

NOP **No Operation**

Syntax: [*label*] NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding:

0000	0000	0000	0000

Description: 何の操作も行なわれません。ステータスも変化しません。実行時間を消費します。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

Example:

None.

RETFIE **Return from Interrupt**

Syntax: [*label*] RETFIE

Operands: None

Operation: TOS → (PC);
0 → GLINTD;
PCLATH is unchanged.

Status Affected: GLINTD

Encoding:

0000	0000	0000	0101

Description: 割込みからの復帰。まずスタックから戻りアドレスを取り出し、その値を PC にロードします。その後、GLINTD ビットをクリアして割込みをイネーブルにします。GLINTD はグローバル割込みディセーブルビット (CPUSTA<4>) です。

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Clear GLINTD	POP PC from stack
No operation	No operation	No operation	No operation

Example: RETFIE

インターラプト後
PC = TOS
GLINTD = 0

RETLW **Return Literal to WREG**

Syntax: [*label*] RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → (WREG); TOS → (PC);
PCLATH is unchanged

Status Affected: None

Encoding:

1011	0110	kkkk	kkkk

Description: サブルーチンからの復帰命令。まず8bit のリテラル ' k ' を WREG にロードし、次にプログラムカウンタにスタックにある戻りアドレスをロードします。
上位アドレスラッチ (PCLATH) は不変のままです。

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	POP PC from stack, Write to WREG
No operation	No operation	No operation	No operation

Example:

```
        CALL TABLE ; WREG contains table
                   ;  offset value
                   ;  WREG now has
                   ;  table value
        :
TABLE
        ADDWF PC   ; WREG = offset
        RETLW k0   ; Begin table
        RETLW k1   ;
        :
        :
        RETLW kn   ; End of table
```

命令実行前
WREG = 0x07

命令実行後
WREG = value of k7

RETURN **Return from Subroutine**

Syntax: [*label*] RETURN

Operands: None

Operation: TOS → PC;

Status Affected: None

Encoding:

0000	0000	0000	0010

Description: サブルーチンからの復帰。まずスタックから戻りアドレスを取り出し、その値をプログラムカウンタにロードします。

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	POP PC from stack
No operation	No operation	No operation	No operation

Example: RETURN

インターラプト後
PC = TOS

RLCF **Rotate Left f through Carry**

Syntax: [*label*] RLCF f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

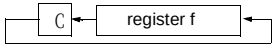
Operation: f<n> → d<n+1>;
f<7> → C;
C → d<0>

Status Affected: C

Encoding:

0001	101d	ffff	ffff

Description: レジスタ ' f ' の内容をキャリフラグを通して 1bit 左に回転します。その結果を ' d ' =0 であれば WREG に、' d ' =1 であれば、レジスタ ' f ' に書き込みます。



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RLCF REG,0

命令実行前
REG = 1110 0110
C = 0

命令実行後
REG = 1110 0110
WREG = 1100 1100
C = 1

RLNCF **Rotate Left f (no carry)**

Syntax: [*label*] RLNCF f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

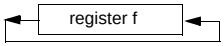
Operation: f<n> → d<n+1>;
f<7> → d<0>

Status Affected: None

Encoding:

0010	001d	ffff	ffff

Description: レジスタ ' f ' の内容を 1bit 左に回転します。その結果を ' d ' =0 であれば WREG に、' d ' =1 であれば、レジスタ ' f ' に書き込みます。



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RLNCF REG, 1

命令実行前
C = 0
REG = 1110 1011

命令実行後
C =
REG = 1101 0111

RRCF **Rotate Right f through Carry**

Syntax: [*label*] RRCF f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

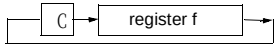
Operation: f<n> → d<n-1>;
f<0> → C;
C → d<7>

Status Affected: C

Encoding:

0001	100d	ffff	ffff

Description: レジスタ ' f ' の内容をキャリフラグを



通して 1bit 右に回転します。その結果を ' d ' =0 であれば WREG に、' d ' =1 であれば、レジスタ ' f ' に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RRCF REG1,0

命令実行前
REG1 = 1110 0110
C = 0

命令実行後
REG1 = 1110 0110
WREG = 0111 0011
C = 0

RRNCF **Rotate Right f (no carry)**

Syntax: [*label*] RRNCF f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

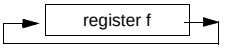
Operation: f<n> → d<n-1>;
f<0> → d<7>

Status Affected: None

Encoding:

0010	000d	ffff	ffff

Description: レジスタ ' f ' の内容を 1bit 右に回転します。その結果を ' d ' =0 であれば WREG に、' d ' =1 であれば、レジスタ ' f ' に書き込みます。



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1: RRNCF REG, 1

命令実行前
WREG = ?
REG = 1101 0111

命令実行後
WREG = 0
REG = 1110 1011

Example 2: RRNCF REG, 0

命令実行前
WREG = ?
REG = 1101 0111

命令実行後
WREG = 1110 1011
REG = 1101 0111

SETF **Set f**

Syntax: [*label*] SETF f,s

Operands: $0 \le f \le 255$
$s \in [0,1]$

Operation: FFh → f;
FFh → d

Status Affected: None

Encoding:

0010	101s	ffff	ffff

Description: 強制的にメモリを FFh にする命令です。' s ' =0 であれば、レジスタ ' f ' と WREG 両方を FFh にセットし、' s ' =1 であれば、レジスタ ' f ' のみ FFh にセットします。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f' and other specified register

Example1: SETF REG, 0

命令実行前
REG = 0xDA
WREG = 0x05

命令実行後
REG = 0xFF
WREG = 0xFF

Example2: SETF REG, 1

命令実行前
REG = 0xDA
WREG = 0x05

命令実行後
REG = 0xFF
WREG = 0x05

SLEEP **Enter SLEEP mode**

Syntax: [*label*] SLEEP

Operands: None

Operation: 00h → WDT;
0 → WDT postscaler;
1 → $\overline{TO}$;
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

0000	0000	0000	0011

Description: CPU をスリープ状態にする命令です。パワーダウンステータスビット ($\overline{PD}$) をクリアし、タイムアウトステータスビット ($\overline{TO}$) をセットし、ウォッチドッグタイマとそのプリスケーラをクリアします。
その後、オシレータの停止とともにプロセッサを SLEEP モードにします。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	Go to sleep

Example: SLEEP

命令実行前
$\overline{TO}$ = ?
$\overline{PD}$ = ?

命令実行後
$\overline{TO}$ = 1 †
$\overline{PD}$ = 0

† WDT がウェークアップを引き起こす場合、このビットをクリアします。

SUBLW **Subtract WREG from Literal**

Syntax: [*label*] SUBLW k

Operands: 0 ≤ k ≤ 255

Operation: k – (WREG) → (WREG)

Status Affected: OV, C, DC, Z

Encoding:

1011	0010	kkkk	kkkk

Description: 8bit のリテラル ' k ' から WREG の内容を引きます。その結果を WREG に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example 1: SUBLW 0x02

命令実行前
WREG = 1
C = ?

命令実行後
WREG = 1
C = 1 ; result is positive
Z = 0

Example 2:

命令実行前
WREG = 2
C = ?

命令実行後
WREG = 0
C = 1 ; result is zero
Z = 1

Example 3:

命令実行前
WREG = 3
C = ?

命令実行後
WREG = FF ; (2's complement)
C = 0 ; result is negative
Z = 1

SUBWF **Subtract WREG from f**

Syntax: [*label*] SUBWF f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

Operation: (f) – (W) → (dest)

Status Affected: OV, C, DC, Z

Encoding:

0000	010d	ffff	ffff

Description: レジスタ ' f ' から WREG の内容を引きます (2 の補数演算)。その結果を ' d ' =0 であれば WREG に、' d '=1 であれば、レジスタ ' f ' に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1: SUBWF REG1, 1

命令実行前
REG1 = 3
WREG = 2
C = ?

命令実行後
REG1 = 1
WREG = 2
C = 1 ; result is positive
Z = 0

Example 2:

命令実行前
REG1 = 2
WREG = 2
C = ?

命令実行後
REG1 = 0
WREG = 2
C = 1 ; result is zero
Z = 1

Example 3:

命令実行前
REG1 = 1
WREG = 2
C = ?

命令実行後
REG1 = FF
WREG = 2
C = 0 ; result is negative
Z = 0

SUBWFB **Subtract WREG from f with Borrow**

Syntax: [*label*] SUBWFB f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

Operation: (f) – (W) – $\overline{C}$ → (dest)

Status Affected: OV, C, DC, Z

Encoding:

0000	001d	ffff	ffff

Description: レジスタ ' f ' から WREG とキャリフラグ (ボロー) の内容を引きます (2 の補数演算)。その結果を ' d ' =0 であれば WREG に、' d ' =1 であれば、レジスタ ' f ' に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1: SUBWFB REG1, 1

命令実行前
REG1 = 0x19 (0001 1001)
WREG = 0x0D (0000 1101)
C = 1

命令実行後
REG1 = 0x0C (0000 1011)
WREG = 0x0D (0000 1101)
C = 1 ; result is positive
Z = 0

Example2: SUBWFB REG1,0

命令実行前
REG1 = 0x1B (0001 1011)
WREG = 0x1A (0001 1010)
C = 0

命令実行後
REG1 = 0x1B (0001 1011)
WREG = 0x00
C = 1 ; result is zero
Z = 1

Example3: SUBWFB REG1,1

命令実行前
REG1 = 0x03 (0000 0011)
WREG = 0x0E (0000 1101)
C = 1

命令実行後
REG1 = 0xF5 (1111 0100) [2's comp]
WREG = 0x0E (0000 1101)
C = 0 ; result is negative
Z = 0

SWAPF **Swap f**

Syntax: [*label*] SWAPF f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

Operation: f<3:0> → dest<7:4>;
f<7:4> → dest<3:0>

Status Affected: None

Encoding:

0001	110d	ffff	ffff

Description: レジスタ ' f ' の上位 4 ビットと下位 4 ビットを入れ替えます。その結果を ' d ' =0 であれば WREG に、' d ' =1 であれば、レジスタ ' f ' に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: SWAPF REG, 0

命令実行前
REG = 0x53

命令実行後
REG = 0x35

TABLRD **Table Read**

Syntax: [*label*] TABLRD t,i,f

Operands: 0 ≤ f ≤ 255
i ∈ [0,1]
t ∈ [0,1]

Operation: If t = 1,
TBLATH → f;
If t = 0,
TBLATL → f;
Prog Mem (TBLPTR) → TBLAT;
If i = 1,
TBLPTR + 1 → TBLPTR

Status Affected: None

Encoding:

1010	10ti	ffff	ffff

Description:
1. テーブルラッチ (TBLAT) の 1 バイトをレジスタファイル ' f ' に移動します。t=0：上位バイトを移動します。t=1：下位バイトを移動します。
2. 次に 16bit のテーブルポインタ (TBLPTR) により示されたプログラムメモリロケーションの内容を、16bit のテーブルラッチにロードします (TBLAT)。
3. i=1：TBLPTR をインクリメントします。 i=0：TBLPTR をインクリメントしません。

Words: 1

Cycles: 2 (3 cycle if f = PCL)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register TBLATH or TBLATL	Process Data	Write register 'f'
No operation	No operation (Table Pointer on Address bus)	No operation	No operation ($\overline{OE}$ goes low)

TABLRD **Table Read**

Example1: `TABLRD 1, 1, REG ;`

命令実行前
- REG = 0x53
- TBLATH = 0xAA
- TBLATL = 0x55
- TBLPTR = 0xA356
- MEMORY(TBLPTR) = 0x1234

命令実行後（テーブルライト完了）
- REG = 0xAA
- TBLATH = 0x12
- TBLATL = 0x34
- TBLPTR = 0xA357
- MEMORY(TBLPTR) = 0x5678

Example2: `TABLRD 0, 0, REG ;`

命令実行前
- REG = 0x53
- TBLATH = 0xAA
- TBLATL = 0x55
- TBLPTR = 0xA356
- MEMORY(TBLPTR) = 0x1234

命令実行後（テーブルライト完了）
- REG = 0x55
- TBLATH = 0x12
- TBLATL = 0x34
- TBLPTR = 0xA356
- MEMORY(TBLPTR) = 0x1234

TABLWT **Table Write**

Syntax: [*label*] TABLWT t,i,f

Operands: $0 \le f \le 255$
$i \in [0,1]$
$t \in [0,1]$

Operation:
If t = 0,
  f → TBLATL;
If t = 1,
  f → TBLATH;
  TBLAT → Prog Mem (TBLPTR);
If i = 1,
  TBLPTR + 1 → TBLPTR

Status Affected: None

Encoding:

1010	11ti	ffff	ffff

Description:
1. 'f'の値を16bitのテーブルラッチ（TBLAT）にロードします。t=0：下位バイトにロード。 t=1：上位バイトにロード。
2. TBLAT の内容を、TBLPTR により示されたプログラムメモリロケーションに書き込みます。TBLPTRが外部プログラムメモリロケーションを示す場合、その命令は 2 サイクルかかります。TBLPTR が内部 EPROM ロケーションを示す場合、割込みが受信された時にその命令を終了します。

注意： $\overline{MCLR}$/Vpp ピンは、内部メモリのプログラミングのために、プログラミング電圧に昇圧する必要があります。$\overline{MCLR}$/Vpp=VDD の場合、内部メモリのプログラミングシーケンスを割込み、ショートライトが起こり(2TCY)、内部メモリのロケーションは影響を受けません。

3. TBLPTRは自動的にインクリメントできます。
   i=0：TBLPTR をインクリメントしません。
   i=1：TBLPTR をインクリメントします。

Words: 1

Cycles: **2（内蔵された EPROM プログラムメモリにライトする場合は、多くなる）**

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register TBLATH or TBLATL
No operation	No operation (Table Pointer on Address bus)	No operation	No operation (Table Latch on Address bus, $\overline{WR}$ goes low)

TABLWT **Table Write**

Example1: `TABLWT 1, 1, REG`

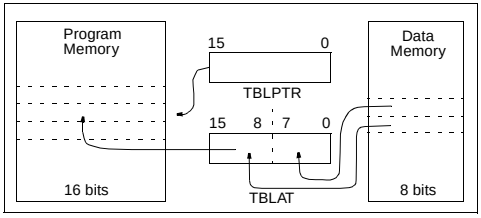
命令実行前
REG = 0x53
TBLATH = 0xAA
TBLATL = 0x55
TBLPTR = 0xA356
MEMORY(TBLPTR) = 0xFFFF

命令実行後（テーブルライト完了）
REG = 0x53
TBLATH = 0x53
TBLATL = 0x55
TBLPTR = 0xA357
MEMORY(TBLPTR - 1) = 0x5355

Example 2: `TABLWT 0, 0, REG`

命令実行前
REG = 0x53
TBLATH = 0xAA
TBLATL = 0x55
TBLPTR = 0xA356
MEMORY(TBLPTR) = 0xFFFF

命令実行後（テーブルライト完了）
REG = 0x53
TBLATH = 0xAA
TBLATL = 0x53
TBLPTR = 0xA356
MEMORY(TBLPTR) = 0xAA53



TLRD **Table Latch Read**

Syntax: [*label*] TLRD t,f

Operands: $0 \le f \le 255$
$t \in [0,1]$

Operation: If t = 0,
TBLATL → f;
If t = 1,
TBLATH → f

Status Affected: None

Encoding:

1010	00tx	ffff	ffff

Description: 16bit のテーブルラッチ（TBLAT）からファイルレジスタ‘ f ’にリードします。テーブルラッチは影響されません。
t=1：上位バイトをリード。
t=0：下位バイトをリード。
データをプログラムメモリからデータメモリに転送するために、この命令をTABLRD と共に使います。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register TBLATH or TBLATL	Process Data	Write register 'f'

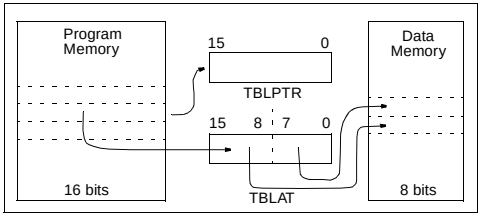
Example: `TLRD t, RAM`

命令実行前
t = 0
RAM = ?
TBLAT = 0x00AF (TBLATH = 0x00)
(TBLATL = 0xAF)

命令実行後
RAM = 0xAF
TBLAT = 0x00AF (TBLATH = 0x00)
(TBLATL = 0xAF)

命令実行前
t = 1
RAM = ?
TBLAT = 0x00AF (TBLATH = 0x00)
(TBLATL = 0xAF)

命令実行後
RAM = 0x00
TBLAT = 0x00AF (TBLATH = 0x00)
(TBLATL = 0xAF)



TLWT **Table Latch Write**

Syntax: [*label*] TLWT t,f

Operands: $0 \le f \le 255$
$t \in [0,1]$

Operation: If t = 0,
f → TBLATL;
If t = 1,
f → TBLATH

Status Affected: None

Encoding:

1010	01tx	ffff	ffff

Description: ファイルレジスタ ' f ' からのデータを 16bit のテーブルラッチ (TBLAT) にライトします。
t=1：上位バイトをライト。
t=0：下位バイトをライト。
データをデータメモリからプログラムメモリに転送するために、この命令を TABLWT と共に使います。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register TBLATH or TBLATL

Example: `TLWT t, RAM`

命令実行前
t = 0
RAM = 0xB7
TBLAT = 0x0000 (TBLATH = 0x00)
(TBLATL = 0x00)

命令実行後
RAM = 0xB7
TBLAT = 0x00B7 (TBLATH = 0x00)
(TBLATL = 0xB7)

命令実行前
t = 1
RAM = 0xB7
TBLAT = 0x0000 (TBLATH = 0x00)
(TBLATL = 0x00)

命令実行後
RAM = 0xB7
TBLAT = 0xB700 (TBLATH = 0xB7)
(TBLATL = 0x00)

TSTFSZ **Test f, skip if 0**

Syntax: [*label*] TSTFSZ f

Operands: $0 \le f \le 255$

Operation: skip if f = 0

Status Affected: None

Encoding:

0011	0011	ffff	ffff

Description: ファイルレジスタを Q と比較し、' f '=0 の場合、次の命令をスキップします。すでにフェッチされている次の命令が廃棄され、これを 2 サイクル命令にするために NOP を実行します。

Words: 1

Cycles: 1 (2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example:
```
HERE    TSTFSZ  CNT
NZERO     :
ZERO    :
```

命令実行前
PC = Address(HERE)

命令実行後
If CNT = 0x00,
PC = Address (ZERO)
If CNT ≠ 0x00,
PC = Address (NZERO)

XORLW **Exclusive OR Literal with WREG**

Syntax: [*label*] XORLW k

Operands: $0 \le k \le 255$

Operation: (WREG) .XOR. k → (WREG)

Status Affected: Z

Encoding:

1011	0100	kkkk	kkkk

Description: WREG の内容と 8bit のリテラル ' k ' との XOR を取り、その結果を WREG に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example: XORLW 0xAF

命令実行前
WREG = 0xB5

命令実行後
WREG = 0x1A

XORWF **Exclusive OR WREG with f**

Syntax: [*label*] XORWF f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

Operation: (WREG) .XOR. (f) → (dest)

Status Affected: Z

Encoding:

0000	110d	ffff	ffff

Description: WREG の内容とレジスタ ' f ' との XOR を取り、その結果を ' d '=0 であれば WREG に、' d '=1 であれば、レジスタ ' f ' に書き込みます。

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: XORWF REG, 1

命令実行前
REG = 0xAF
WREG = 0xB5

命令実行後
REG = 0x1A
WREG = 0xB5

NOTES: