[第7章]

もっとPICを使いこなそう

光永 法明

具体的な事例はこれまでの各章で説明しました．ここでは，最初にデータ・シートの読み方のポイントを説明します．そして，スリープ命令，ウォッチ・ドッグ・タイマの使い方など，使いこなしのためにノウハウが必要な命令について説明します．さらに，条件アセンブル，マクロ機能など，アセンブラのより深い使い方を説明します．

また，ヘッダ・ファイルを読んでみることで，PICのアセンブラで用いるレジスタなどの定義を確認し，メーカが用意しているアプリケーション・ノートを紹介し，本書で扱っているPIC16F877Aと同じ14ビット・コアであるPIC12F675（8ピン）を使うときの注意点に触れます．最後に，デバッグのこつに触れます．

このアイコンは，章末に用語解説があります

7-1　データ・シートを読もう

それぞれの電子部品についての最良の情報源の一つはメーカの発行するデータ・シートです．現在はインターネットで各メーカが公開していることが多く，入手が容易になりました．PICマイコンについても例外ではありません．

残念ながら，PICのデータ・シートの多くは日本語に翻訳されていません．けれども幸いなことに，PICマイコンは同じコアのシリーズはよく似ているので，ほかのPICマイコンの日本語データ・シートを参考にすることができます．

PIC16F877Aの場合には，PIC16F877の日本語データ・シートが参考になります．注意としては型番が異なるマイコンにはやはり違いがあるということです．たとえば，PIC16F877にはコンパレータが内蔵されていないので，その点は参考になりません．また，書き込みについてもPIC16F877とPIC16F877Aは少し違うので，ライタの設定に注意します．

お勧めの読み方は，英語のデータ・シートを基本と考えて，参考にする日本語データ・シートは辞書を引く手間を省くためのものと考え，おかしいと思ったら辞書を引くことです．また，日本語と英語のデータ・シートがある場合についても，数値などに差があれば英語データ・シートのほうを優先します．英語データ・シートも更新されることがあるので，できるだけ新しいものを入手するようにします．

7-2　データ・シートを読んでみる（データ・シートの構成）

PIC16F87XAのデータ・シート（DS39582B）の構成は，次のようになっています．

特徴をまとめた概要
ピン配置
目次
1章　デバイス概要(ブロック図やピン配置など)
2章　メモリ構成(レジスタの構成とSFRの概要)
3章から13章　内蔵モジュールの解説
14章　リセット時の動作やコンフィグレーション・ビット，割り込み，ウォッチ・ドッグ・タイマ，スリープなどCPU部分の解説
15章　機械語の概要
16章　開発環境の概要
17章　電気的特性
18章　電気的特性(図，表)
19章　パッケージについての情報
付録A　データ・シートの更新について
付録B　データ・シートに記述されているデバイスの違い
付録C　ほかのPICとPIC16F87XAの比較表

データ・シートはPDFで入手できますから，アクロバット・リーダで画面に表示するか印刷したものを見ながら，以下の説明を読んでください．最初の概要を読むと，PIC16F873A/874A/876A/877Aは同じシリーズで，ピン配置とメモリの大きさが違うことがわかります．はじめてのPICを使う場合には，まず2章までに目を通します．そして，14章と，必要な内蔵モジュールの章を読みます．また，回路を決定するときには，17章，18章にも目を通しておきます．

7-3　データ・シートを読んでみる(内蔵EEPROMの使い方)

ここでは，データEEPROMについての3章を読んでみます(PIC16F87Xの日本語データ・シートDS30292A-Jでは4章にあたる)．

EEPROMというのは電気的に消去可能なROMのことです．第5章の1項で明るさを覚えるプログラムを紹介しましたが，レジスタに書いていたため電源を切ると忘れてしまいました．EEPROMを使うと電源を切っても記憶しておくとことができます．

EEPROMは書き換え可能回数が100万回程度です．100万回というと多いようですが，1秒に一度書き換えるような用途には向きません．1秒に一度書き換えると，300時間程度で100万回に達してしまいます．

● 概要と関連レジスタ

3.0章を読むと，EEPROMの読み書きには，EECON1，EECON2，EEDATA，EEADRが関係していることがわかります．EEDATAレジスタが読み書きする8ビット(1バイト)のデータを，EEADRレジスタがアドレスを扱います．

EEPROMが128バイトのデバイス(PIC16F873A/874A)では，アドレス0x00から0x7fが有効で，0x80から0xffは0x00から0x7fとみなされます．また，0x80から0xffでは書き込めないこと

もわかります．

3.1章から，その理由が，EEPROMが128バイトのデバイスではEEADRの最上位ビットがないためとわかります．また，コード・プロテクトをコンフィグレーション・ビットで指定した場合にも，プログラムからはデータEEPROMを読み書きできることがわかります．

3.2章には，EECON1，EECON2レジスタの解説が書かれています．REGISTER 3-1の図を併せて読んでいきます．レジスタの図には，各ビットが読めるか(R)，書けるか(W)，セット(‘1’を書く)のみ(S)かが書かれています．またパワー・オン・リセット時の値(‘0’/‘1’)も書かれています．

EEPGDビットは，プログラム・メモリかデータ・メモリ(EEPROM)のどちらにアクセスするかを決めます．ここではEEPROMにアクセスするので‘0’にします．

RDビットとWRビットは，それぞれ読み込み，書き込みを指示します．また，これらのビットはセット(‘1’にする)ことはできますが，プログラムから‘0’にすることはできません．

WRENビットを‘1’にすると，書き込み消去ができるようになります(書き込みのためには‘1’にしなくてはいけない)．

パワー・オン・リセットでは，このビットは‘0’です．WRERRビットは，書き込みがリセット信号かウォッチ・ドッグ・タイマのために中断された場合にセットされます．このようなリセットがかかる可能性がある場合には，プログラムの初期化部分に，これに対応するようなプログラムを書く必要があることがわかります．

また，PIR2レジスタのEEIFビットが，書き込み終了時に‘1’になり，プログラムで‘0’にしなくてはいけないこともわかります．EECON2レジスタは通常のレジスタではなく，書き込み時にのみ使う特別なレジスタだともあります．

ここに注意書きがあり，プログラム・メモリをプログラム内で書き換える場合について，PIC16F87XからPIC16F87XAへの変更があることが書かれています．データEEPROMの場合には関係しませんが，プログラム・メモリを書き換える場合には注意します．

● プログラミング例

3.3章には，データEEPROMから読むための手順が書かれています．ここで重要なのは，EXAMPLE3-1のプログラム例です．バンク切り替えを含めて書かれているので，プログラムを書く際には，とても参考になります．もし，EEPGDビットが‘0’であることがわかっていれば，ここのクリア6行目は不要です．

EXAMPLE 3-1で6行目に書かれているのはバンク切り替えのための命令数が少なくなるように工夫されているのです．また，EXAMPLE 3-1の終了時にはバンク0ではないので注意します．

3.4章には，データEEPROMへの書き込み手順が書かれています．EEPROMへ書く際には，EECON2に0x55，0xaaを書き，WRビットを‘1’にします．このとき，割り込みが入ると手順が守れず，書き込みに失敗するので割り込み禁止にすることが推奨されています．

また，WRENビットは，誤って書き換えないように必要以外のときは‘0’にしておくようにと書かれ，書き込みが完了する前にビットを‘0’に戻してもよいとあります．

プログラムの例としてEXAMPLE 3-2が用意されています．EXAMPLE 3-2では，本文中の手順1(書き込みの前に前回の書き込みが完了するまで待つ)，手順10(書き込みの後で完了するまで待つ)は含まれていないので，いずれかを追加する必要があります．また，書き込みの完了のチェックには，EECON1のWRビットかPIR2のEEIFビットを調べます．

● 注意事項とまとめ

3.5章，3.6章にはプログラム・メモリの読み書きについて書かれています．3.7章には，データEEPROM，プログラム・メモリを書き換えるべきではないときには保護回路が働くことが書かれています．保護回路がないマイコンの場合には，自分でプログラム内に保護のための処理を書く必要があります．3.8章にはコード・プロテクトについて再度書かれています．TABLE 3-1は3章に関連するレジスタの一覧です．必要な情報がコンパクトにまとめられています．

データ・シートを読むのも難しくはないことがおわかりいただけたでしょうか．本書で紹介しきれなかった内蔵デバイスも，ぜひデータ・シートを読んで使いこなしてください．

7-4　スリープ

スリープ命令(sleep)を使うことで，PICを省電力モードにすることができます．省電力モードではPICのほとんどの動作が停止し，消費電流が大きく低下します．sleepを実行すると次のようになります．

- ウォッチ・ドッグ・タイマがクリアされる
- STATUSレジスタの$\overline{PD}$ビット(STATUS<3>)が'0'になる
- STATUSレジスタの$\overline{TO}$ビット(STATUS<4>)が'1'になる
- オシレータ(発振子を駆動している発振回路)が停止する

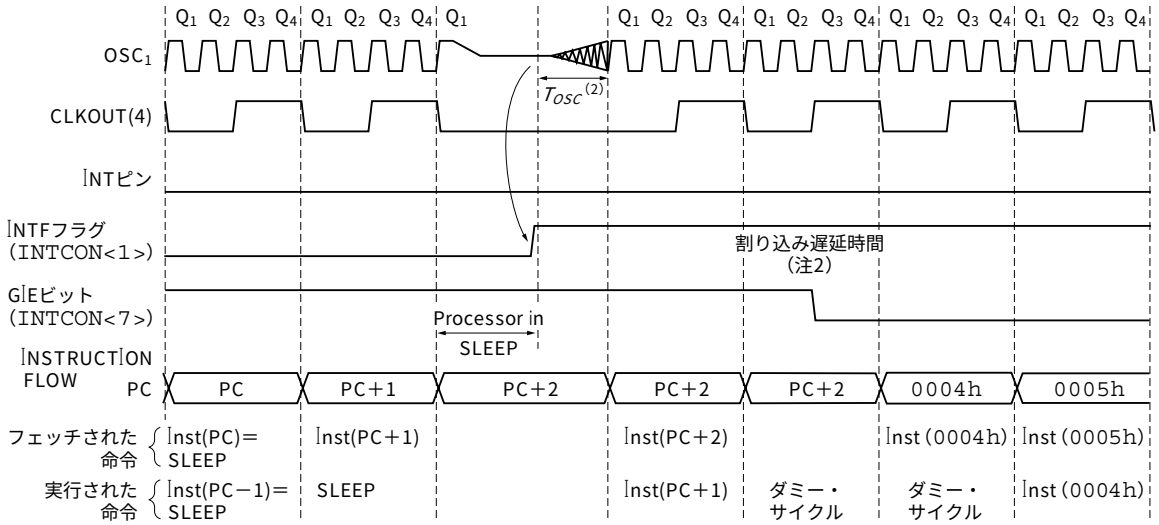
● スリープからの復帰

スリープからの復帰には，次のような方法があります．

- $\overline{MCLR}$ピンによるリセット
- ウォッチ・ドッグ・タイマ
- 割り込み
  - INTピン(ポートBビット0)からの割り込み
  - ポートBの変化割り込み
  - パラレル・スレーブ・ポート(PSP)の読み書き
  - TMR1割り込み(外部入力をカウントし，非同期モードであること)
  - CCPキャプチャ・モードの割り込み
  - CCPの特殊イベント・トリガ(TMR1は外部入力をカウントし，非同期モード)
  - 同期シリアル・ポート(SSP)のスタート/ストップ・ビット検出
  - 同期シリアル・ポート(SSP)のスレーブ・モード(SPI/I^2C)での送信または受信
  - USARTの同期スレーブ・モードでの送信または受信
  - A-D変換クロックが*RC*のときのA-D変換終了
  - データEEPROM書き込みの完了
  - コンパレータ出力の変化

スリープから復帰すると，必ずsleep命令の次の命令が実行されます．割り込みが有効な場合も，割り込みルーチンへジャンプする前に実行されます．割り込みによるスリープからの復帰時の動作は**図7-1**となります．

スリープ命令から復帰したときには，復帰の要因を調べる方法が用意されています．スリープ後の



注意 1：XT，HSまたはLPオシレータ・モードの場合
2：T_{OSC}＝1024T_{OSC}（図の時間軸は省略）．この遅延はRCオシレータ・モードで発生しない
3：GIE＝‘1’の場合．スリープからの起動後，割り込み処理ルーチンにジャンプする．GIE＝‘0’の場合はプログラムの実行が継続される
4：CLKOUTはこのOSCモードでは出力しないが，この図ではタイミングの参考のために示す

図7-1　割り込みによるスリープからの起動

OSC_1のパルス四つで1命令（PCが一つ）進む．スリープ中はPCは進まない．スリープから起動するとLP，XT，HSモードではT_{OSC}（1024 T_{OSC}）の期間，発振が安定するのを待つ時間がある．その後，sleep命令の次の番地の命令が実行され，割り込みが許可されている場合には割り込みルーチンへジャンプする．

$\overline{MCLR}$によるリセットをパワー・オン・リセットと区別するには，STATUSレジスタの$\overline{PD}$ビットを調べます．$\overline{PD}$が‘1’のときにはパワー・オン・リセット，‘0’のときにはスリープからの復帰です．

スリープ命令からの復帰がウォッチ・ドッグ・タイマによるものかは，STATUSレジスタの$\overline{TO}$ビットを調べます．$\overline{TO}$ビットが‘0’であれば，ウォッチ・ドッグ・タイマによる復帰です．割り込みについては，それぞれのフラグを調べます．

● スリープ命令を使うときの注意

スリープ命令の利用にはいくつかの注意事項があります．

(1) オシレータが停止するため，これに依存したモジュールによる割り込みでは復帰できない
(2) LP/XT/HSモードでは復帰に1024クロック必要
(3) 復帰する要因となるモジュールの割り込みは有効にする必要があるが，グローバル割り込みは有効でなくてもよい（GIEは‘0’でもよく，割り込みルーチンへのジャンプはしないこともできる）
(4) 割り込みが有効な場合，sleep命令の次の命令を実行してから，割り込みルーチンへジャンプする
(5) ウォッチ・ドッグ・タイマを有効にしているとき，割り込みが入るタイミングによっては，ウォッチ・ドッグ・タイマがクリアされない

一つ目の制限のため，非同期シリアルの受信割り込みや，内蔵オシレータによるタイマ割り込みなどは使えません．二つ目は，発振が安定するために必要な制限ですが，早い応答が必要な場合には*RC*発振でないとスリープは使えない場合があります．

割り込みルーチンで処理する必要がない場合には，三つ目の特性を利用します．モジュールの割り込み

を許可をして，GIEを‘0’にしておきます．これにより割り込みで復帰はしますが，割り込みルーチンへはジャンプしません．

四つ目は注意が必要です．スリープから復帰後，割り込みルーチンの実行前に，sleep命令の次の命令が実行されます．動作が意図どおりにならないときには確認してください．sleep命令の次にnop命令を書いておくとよいでしょう．nop命令は何も変化しないので，割り込みルーチンが呼ばれる前には何も実行しないことと同じになります．

五つ目も注意が必要です．sleep命令の実行直前に割り込みが働くと，スリープしません(nop命令として扱われる)．このとき，$\overline{PD}$ビット，$\overline{TO}$ビットは変化せず，ウォッチ・ドッグ・タイマのクリアも行われません．スリープ命令が実行されたかどうかは，$\overline{PD}$ビットを調べます．$\overline{PD}$ビットが‘1’ならスリープしていません．

また，ウォッチ・ドッグ・タイマのクリアが行われないため，意図しないリセットがかかる可能性があります．ウォッチ・ドッグ・タイマのクリアはsleep命令に頼らず，clrwdt命令をsleep命令の直前に書いておくと確実です．

7-5　ウォッチ・ドッグ・タイマ

● ウォッチ・ドッグ・タイマとは

ウォッチ・ドッグというのは番犬のことです．実際の役割は，遊んでやらないといたずらしてくる子犬のイメージです．コンフィギュレーション・ビットでウォッチ・ドッグ・タイマ(WDT)を有効にすると，一定時間内にclrwdt命令を実行しないとリセットがかかります．そのため，意図せずにコンフィギュレーション・ビットでWDTが有効になっているとリセットが働いてうまく動かないという現象が起きます．

ウォッチ・ドッグ・タイマの一つの使い方は，なんらかの原因でプログラムが暴走したときにリセットをかけることで動作を安全なものにするというものです〔**図7-2**(**a**)〕．この場合，プログラム中の適当なところにclrwdtを書いておきます．書く場所は，clrwdt命令が実行される間隔が，ウォッチ・ドッグ・タイマが働く時間より短く，かつできるだけ少なくします．間隔が長いと正常動作時にもリセットが働いてしまいますし，多いと暴走したときにたまたまclrwdtが実行される可能性が高くなります．

ウォッチ・ドッグ・タイマのもう一つの使い方は，sleep命令からの復帰です〔**図7-2**(**b**)〕．ウォッチ・ドッグ・タイマはメインの発振子が停止しても動作するため，sleepからの復帰に使うことができます．

● ウォッチ・ドッグ・タイマの働く時間

ウォッチ・ドッグ・タイマの働く時間を決めているのは内蔵の*RC*発振子です．この発振周期T_{WDT}は，約18[ms]で働きますが，個体差と温度による変動があり，8[ms]から33[ms]に変化します（5 V，−40[℃]から85[℃]）．

この信号はプリスケーラに通すことで，ウォッチ・ドッグ・タイマの働く周期を整数倍にできます．**図5-23**のようにOPTION_REGレジスタで設定し，1倍から128倍までの範囲で選択できます．ただし，**図7-3**のようにプリスケーラはタイマ0と兼用のため，タイマ0に割り当てた場合にはウォッチ・ドッグ・タイマの周期はT_{WDT}から変えることはできません．

プリスケーラの割り当てを，タイマ0からウォッチ・ドッグ・タイマ，あるいは逆に変更するためには，意図しないリセットを防ぐため**リスト7-1**，**リスト7-2**のようにします．ウォッチ・ドッグ・タイマをコ