[第5章]

初めてのPICプログラミング

MPLAB IDEを使ってテスト・ボードのプログラムの作り方を確認する

前章で製作したテスト・ボードのLEDを点滅させるソフトウェアを作成し，ボードのLEDを点滅させます．ソース・プログラムのひな形のテンプレートは，第2章で説明したリスト2-1をベースにします．

5-1　ソフトウェアの作成

第一段階であるハードウェアの基本的なテストを終えたら，次はソフトウェア(プログラム)を作成し実際に動作させます．元となるソース・ファイルを作成する環境は，MPLAB IDEで行います．前章で検討した内容をベースに実際に動かすにはどうすればよいかを考えていきましょう．

検討の結果を具体的な手順に書いてできあがったのが，**図5-1**のフローチャートです．このフローチャートに従いプログラムの具体的な記述を行います．プログラムを書く前には，このような処理の流れを図に現し，流れや判断に無理がないかなどを検討します．

プログラムを書くのが初めてとか，ハードウェアに密着したPICのプログラムを作ったことがない場合，このフローチャートさえ，思い浮かばないのが普通です．では，どうしたらよいのでしょうか．それは，はじめは，本書のような解説を読み，プログラムをわからないけれども眺めていきます．本書に限らず，他人の書いたプログラムを何度も読んでいると，処理の手順や書き方の定石が見えてくるようになります．

ここで説明するプログラム本体の命令部分は，各ブロック10行くらいのものが二つで，全体でも20行をそれほど超えないものになるはずです．最初は，少し我慢してじっくり内容を確認してください．

プログラムの中身は大別すると次のようになります．

① 初期化処理
　ⓐ アセンブラに対して，使用するデバイスのタイプ，コンフィギュレーション・ワードの設定
　ⓑ 汎用レジスタとして使用するレジスタの定義など，実際の命令の記述に先立って行う処理
　ⓒ PICのポート使用方法および条件などを初期設定処理として記述する部分
② データの読み取りおよびポートへの出力
③ 時間待ち処理
④ カウント・ダウンおよびゼロのチェック

時間待ち処理の③については，何回も使われるので，サブルーチン・プログラムとして必要なときにいつでもCall命令で呼び出して利用ができるようにしてみます．

このアイコンは，章末に用語解説があります

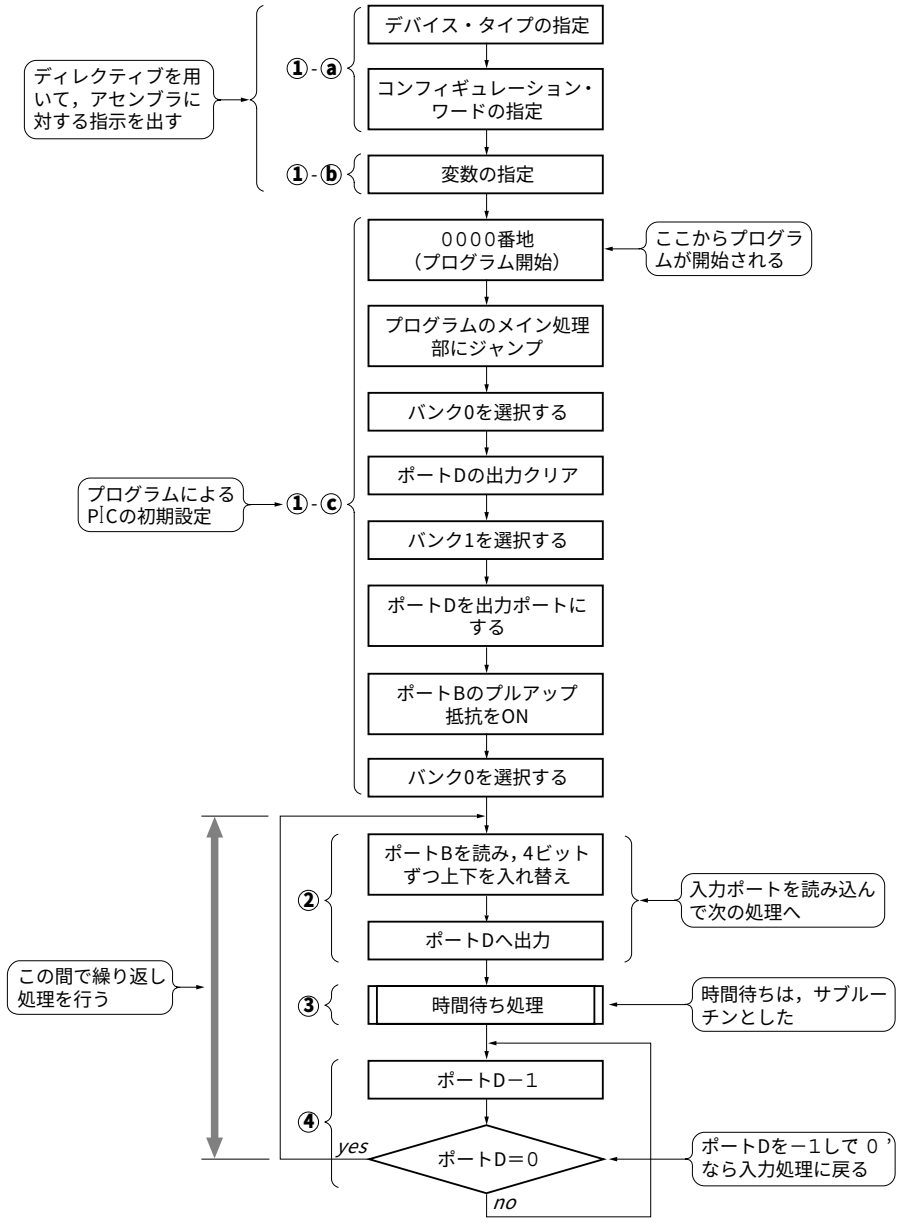


図5-1　テスト・プログラムf877010.asmの処理手順

5-2　MPLAB IDE提供のひな形を利用してプログラムの作成を始める

第2章で説明しましたが，ソース・プログラムのひな形として，MPLABに添付されているテンプレートの`f877atemp.asm`を読み込むと必要な事項が盛り込まれています．MPLAB IDEのデスクトップのメニュー・バーでFile＞OPENでテンプレート・ファイルの一覧表が表示されます．表示されない場合は，次のフォルダを選択します

C:¥Program Files¥MPLAB IDE¥MCHIP_Tools¥TEMPLATE¥Code

f877atemp.asmを読み込みソース・ファイルの元とします．追加修正してプロジェクトを作ったフォルダにSave Asで保存して，その後に，プロジェクトに追加します．

① 初期化処理

ⓐ具体的なコードを記述する前の準備

プログラムの先頭部には，

```
list  p=16f877A （デバイスの設定）
#include  <p16f877A.inc> （ヘッダ・ファイルの指定）
__CONFIGディレクティブでコンフィギュレーション・ワードの設定
```

を記述します．

__CONFIGについては変更の必要があります．元のコマンドは“__CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON & _RC_OSC & _WRT_OFF & _LVP_ON & _CPD_OFF”となっています．ウォッチ・ドッグ・タイマは利用しないので，_WDT_OFFのままとします．これがONのときは，ウォッチ・ドッグ・タイマの処理をしないとリセットが繰り返されプログラムが進行しなくなります．

また，低電圧プログラムの機能をOFFにするため“_LVP_OFF”とします．これを忘れると，RB_3/PGMピンが低電圧入力と設定され，何も接続しないと“H”と判断し，プログラム・メモリへプログラムを書き込むモードとなるので，プログラムが動きません．したがって，修正するのはこの1カ所です．

ⓑ変数定義部

その他に，変数の定義を行います．③の時間待ちをするサブルーチンの中で作るプログラムで使う，ループの回数をコントロールするためのワーク・エリアを二つ汎用レジスタに定義します．

ⓒPICの初期設定のための処理

orgディレクティブで0000番地からの開始を示し，メイン・プログラムへのジャンプを記述します．

電源投入後，またはリセット・ボタンを押した後などには，リセット，パワー・オン・リセットが働くので，プログラムは**図5-2**に示すように，プログラム・メモリの0000番地からスタートします．そのため，リセット後の最初に実行する命令をここに記述します．

また，0004番地に割り込み時の割り込み処理プログラムが割り当てられています．そのため，割り込み処理を行う場合は，割り込み処理プログラムをこの場所から書かなければなりません．0番と4番は近いためいろいろなプログラムをこの間に書くことができないので，初期設定のためのプログラムは，この割り込み処理のプログラム部分の後に記述します．つまり，0000番地にメイン・プログラムへジャンプする命令を書き，そこからが本当のプログラムの始まりになります．

割り込み処理をまったく使用しないのであれば，この割り込み処理の開始アドレスを無視することができます．本章では割り込み処理を使用しませんが，割り込み処理を考慮したプログラムとして割り込み処理の開始アドレスは空けておきます．

リセット後，レジスタの値は初期設定され0000番地からプログラムが起動されます．ポートなどの入出力などを初期設定の状態と異なった仕様で運用する場合は，最初にこれらの変更を行います．ここでは，ポートDが初期化時に入力と設定されているので，出力ポートに変更します．

▶各ポートの入出力の設定方法

図5-3には，PICで一番利用されることの多い標準的なディジタル汎用入出力ポートの概要を示します．

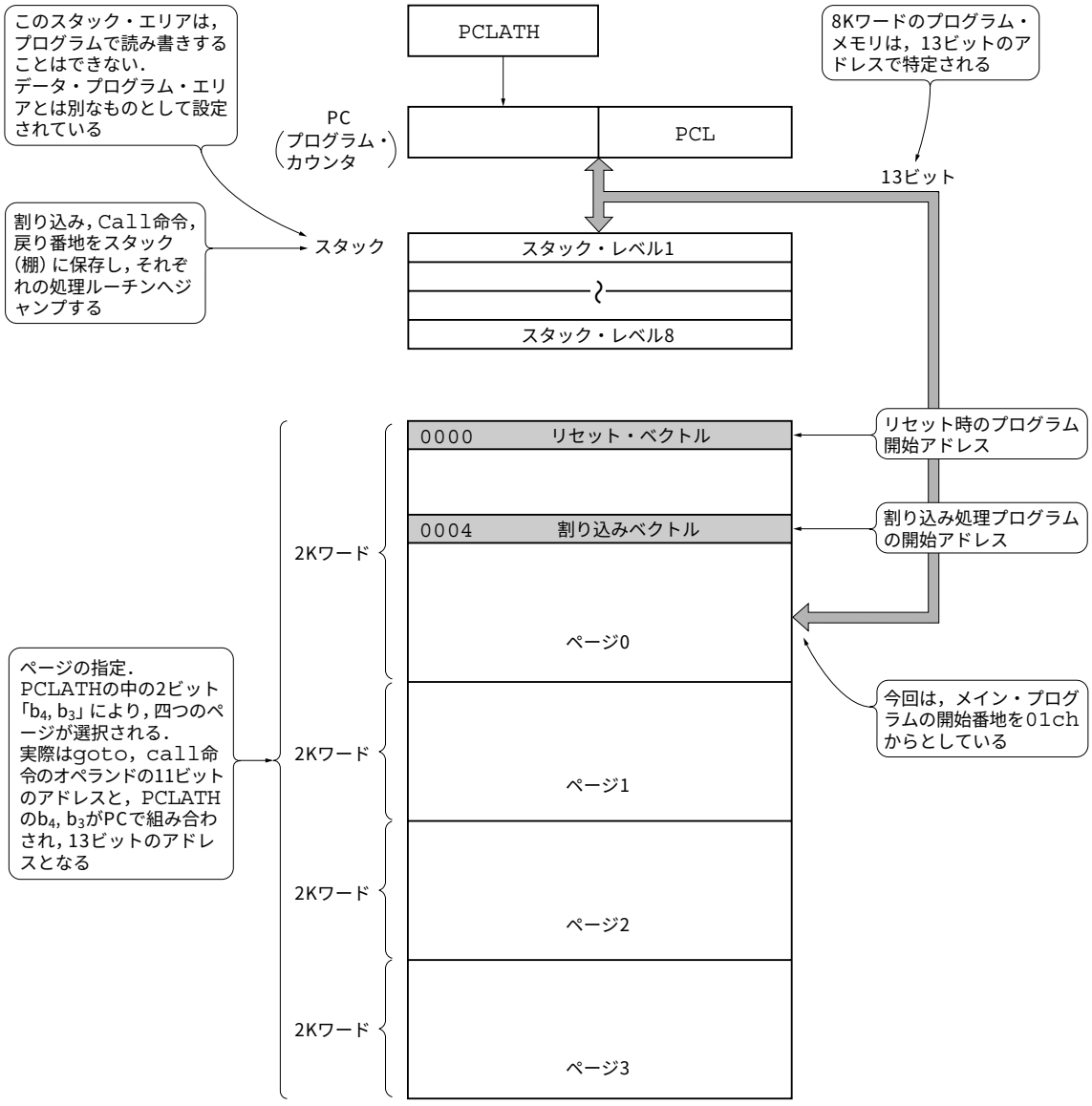
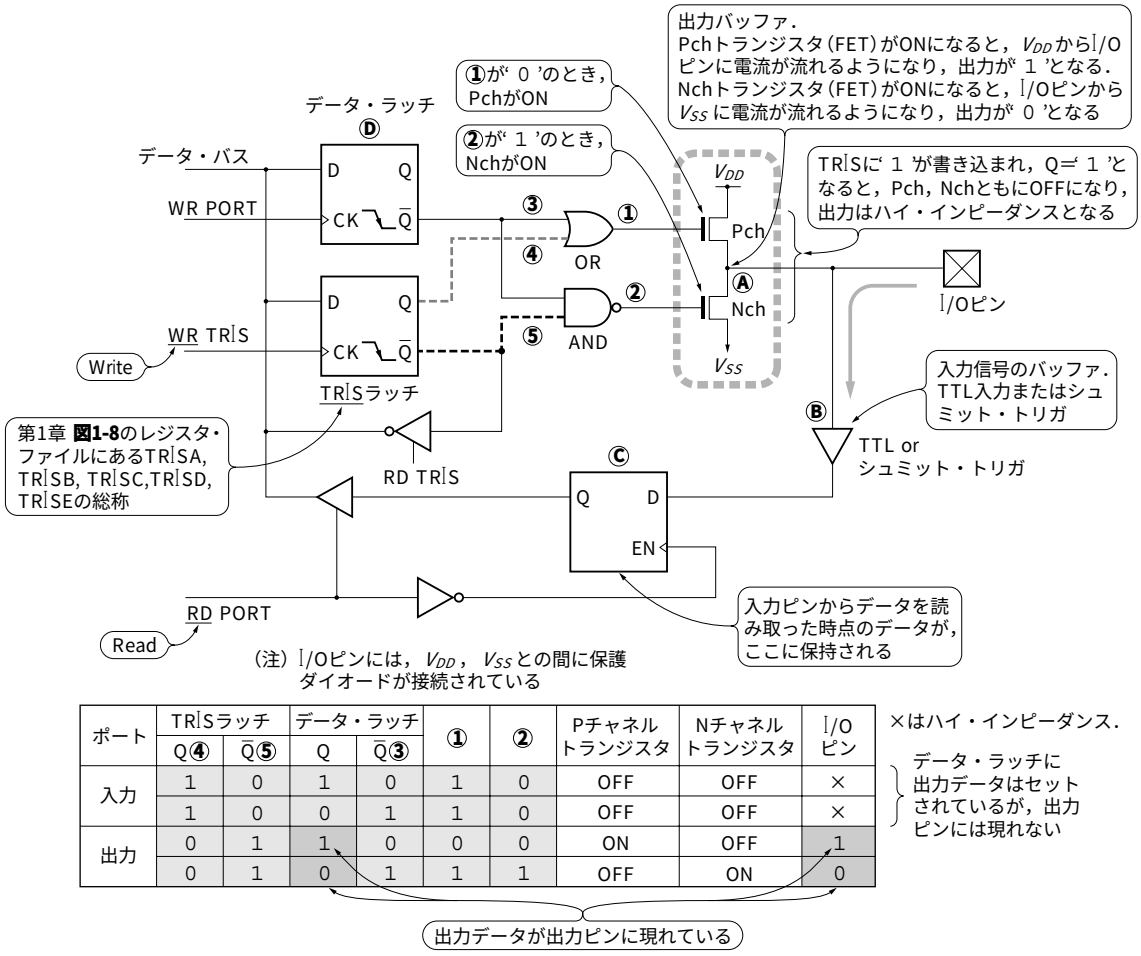


図5-2(1)　プログラム・メモリの配置

この印は頁右上に略語の語源の説明があります

入出力は同一の端子を共有していますから，入力ポートとして設定する場合は出力バッファから出力を出さないようにします．そのために出力バッファはスリー・ステート出力(第4章Column-1参照)という構造になっており，この出力をハイ・インピーダンスにする制御をTRISラッチで行っています．

端子を入力とするビットには‘1’を，出力として設定する端子には‘0’を設定します．TRISラッチに‘1’を設定すると，**図5-3**に示すようにPチャネル，NチャネルのFETⓘをドライブするOR，ANDゲートの出力が‘1’，‘0’に固定され，出力データの値に関わらず両方のFETがOFFになったままなので，出力(Ⓐ)は入力電圧に影響を与えないハイ・インピーダンスのまま固定されます．その状態であれば，



ポート	TRISラッチ		データ・ラッチ		①	②	Pチャネル トランジスタ	Nチャネル トランジスタ	I/O ピン
	Q④	Q̄⑤	Q	Q̄③					
入力	1	0	1	0	1	0	OFF	OFF	×
	1	0	0	1	1	0	OFF	OFF	×
出力	0	1	1	0	0	0	ON	OFF	1
	0	1	0	1	1	1	OFF	ON	0

×はハイ・インピーダンス．

データ・ラッチに出力データはセットされているが，出力ピンには現れない

出力データが出力ピンに現れている

図5-3　PICの標準的なディジタル汎用入出力ポートの構成

この端子に接続された外部入力の値を入力バッファ(Ⓑ)から読み込むことができます．

▶ 入出力用，入出力方向制御のレジスタがある

PICの汎用ディジタル・ポートから入出力されるデータをプログラムで取り扱うのはとても簡単です．**図5-4**に示すように，データの入出力はポートA，B，C，D，Eに対応するレジスタ・ファイル(PORTA，PORTB，PORTC，PORTD，PORTE)の読み書きをすればよいだけなのです．

これらのポートの入出力の方向を決めるために，TRISA～TRISEまでの制御レジスタが用意されています．そして，この入出力の方向はビット単位で設定できます．

具体的には，制御レジスタの中のあるビットを‘1’にすると，対応するポートのビットが入力となり，‘0’を設定すると該当するポートのビットは出力になります．電源投入後のこの制御レジスタの値はALL 1となっているので，汎用ディジタル・ポートと設定されているポートは，入力ポートとしてスタートします．したがって，出力に使いたい場合はTRISレジスタの値を変更し，出力ポートになるように設定してから利用します．

少しここで面倒なのは，ポートのレジスタとポートの制御レジスタは異なったアドレスのバンクにあ

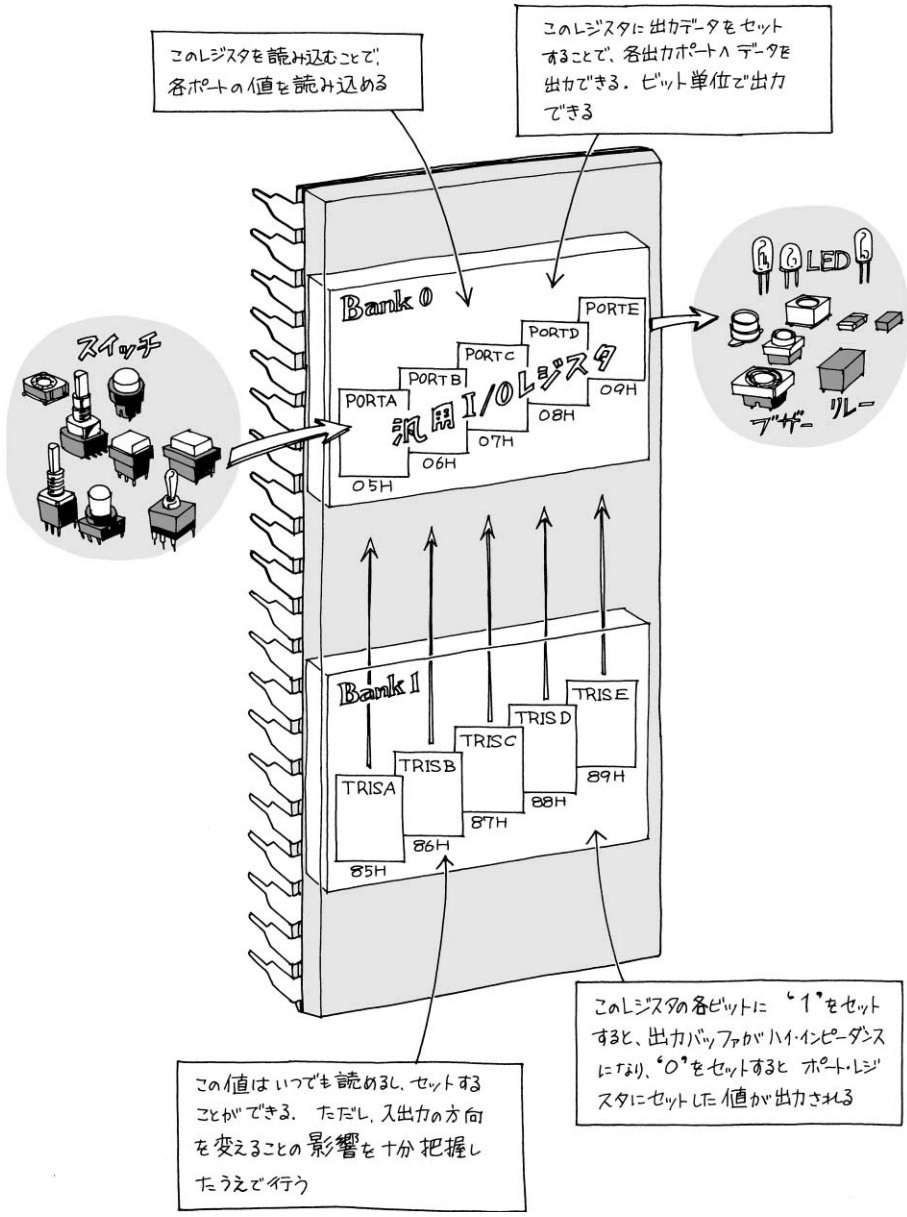


図5-4　PICのI/Oポートへの入出力
PICでは直接I/Oポートに対して入出力をするのではなく，I/Oポートと関連づけられたレジスタを読み書きすることで，同時に入出力処理が行われる．

ることです(第1章**図1-8**参照)．設定を行うときにバンクを切り替えなければなりませんが，その選択はSTATUSレジスタのRP1，RP0の両ビットで制御します．この2ビットでバンク3～0までの四通りを設定できますが，主にバンク0を利用しますから，特別なときにバンク1を選択し後はバンク0に戻すというプログラムを書きます．

今回は，ポートDを出力，ポートBを入力ポートとし，内蔵入力プルアップ抵抗をONにします．