[第3章]

PICプログラミングの考え方をシミュレータを使って理解する

プログラミングは手順の分解

光永 法明

プログラミングとは，やりたいことを手順に分解していくことです．小学校で掛け算や割り算を筆算で九九ができる単位に分けて計算していったように，少しずつ簡単な手順に分けていくことなのです．本章では，パソコン上のシミュレータを使って，PICがどのように動作するのかを見ていき，プログラミングのこつを紹介していきます．ぜひシミュレータを使ってPICの動作を自分で確かめてください．

3-1　シミュレータを使ってみる

シミュレータは，実際のPICの内部の動作をパソコン上で模擬してくれるものです．これを使うと，普通は見えないPICの内部動作が目に見えるようになります．MPLABのインストール，シミュレータの使い方はAppendix Aを見てください．

まずは足し算をPICで実行するシミュレーションをしてみましょう．プログラムは**リスト3-1**のとおりです．エディタ・ウィンドウで入力してください．

行頭の空白はタブです．スペース・バーではなく，タブ・キーで入力します．入力が終わったら，アセンブルです．Projectタブから Makeを選ぶかファンクション・キーF10を押します．

図3-1のように，BUILD　SUCCEEDEDと表示されればアセンブルは成功です．うまくアセンブルできない場合には，文字が間違っていないか見直してください．

リスト3-1　足し算プログラム

```
      list    p=16f877a          ;  PIC16F877A用のプログラムであることを宣言
      #include p16f877a.inc       ;  PIC16F877A用のヘッダ・ファイルを読み込む

REGA  equ     0x21 ←――― アドレス0x21のレジスタをREGAという名前をつけて使う

main
      movlw   0x1                ;  wレジスタに1を代入(w＝1)
      addlw   0x2                ;  wレジスタに2を加算(w＝w＋2＝1＋2)
      movwf   REGA               ;  REGAにwレジスタの値を代入(REGA←w＝3)
STOP_HERE                                                  ┐ ここを
      goto    STOP_HERE          ;  スリープ命令(プログラムはここで止まる)  ┘ ずっと繰り返す

      end                        ;  プログラムの最後には必ず書く
```

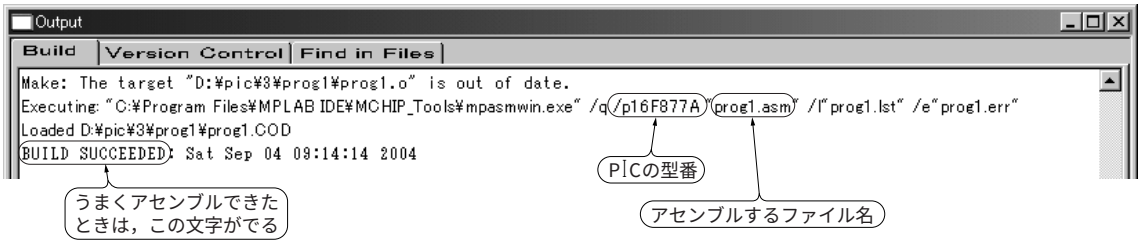

図3-1　うまくアセンブルできたときのOutputウィンドウ
MPLABでアセンブルするとOutputウィンドウが表示される．
アセンブルに成功したときには“BUILD SUCCEEDED”と表示されるので確認する．またタイプ・ミスなどではエラー(error)や警告(warning)が表示される．

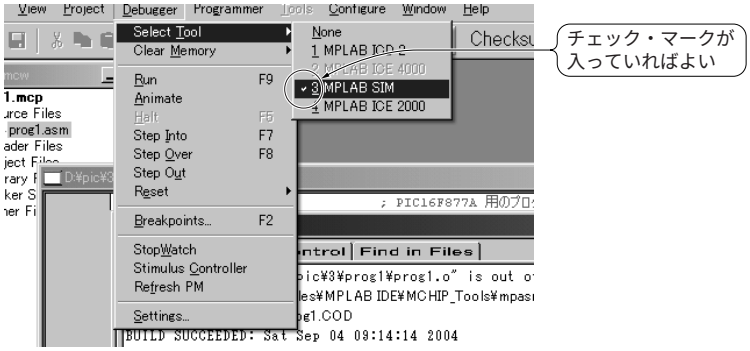

図3-2　MPLAB SIMにチェック・マークをつける
シミュレータを使うにはDebugger→Select Toolで選択できる“MPLAB SIM”を選択し，チェック・マークが表示された状態にしておく．

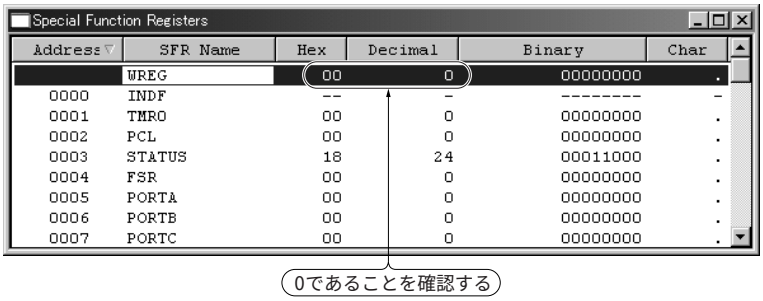

図3-3　Special Function Registers(特殊機能レジスタ，SFR)ウィンドウを見る
SFRを表示させることができる．ウィンドウ内にはSFRのアドレス，名前，値(16進，10進，2進と文字)が表示される．

つぎに，MPLAB SIMが使えるようになっているか確認しましょう．**図3-2**のように，Debugger→Select Tool→3 MPLAB SIMのところにチェック・マークがついていればOKです．

Viewタブから，Special Function Registersをクリックし，ウィンドウを出しましょう(**図3-3**)．このウィンドウを見るとwレジスタ(WREG)の値は0ですね．0でなかったらWREGのHEX(16進表記)かDecimal(10進表記)の数字をクリックして，0を入力します．もう一つ，Viewタブから，File Registersウィンドウを出しましょう(**図3-4**)．

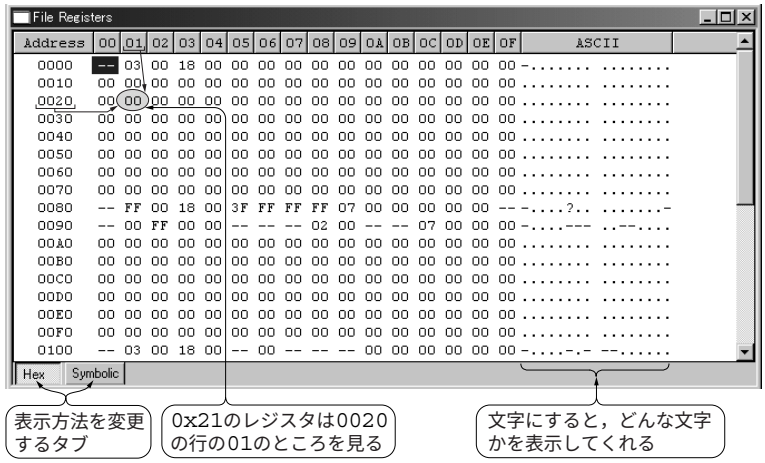

図3-4　File Registersウィンドウ（実行前）
リスト2-1のプログラムを実行する前にファイル・レジスタを表示させたところ．表の見方としてはファイル・レジスタ・アドレスを16進数で表記したときの上の3桁の行の，下1桁の値が一致するところを見る．

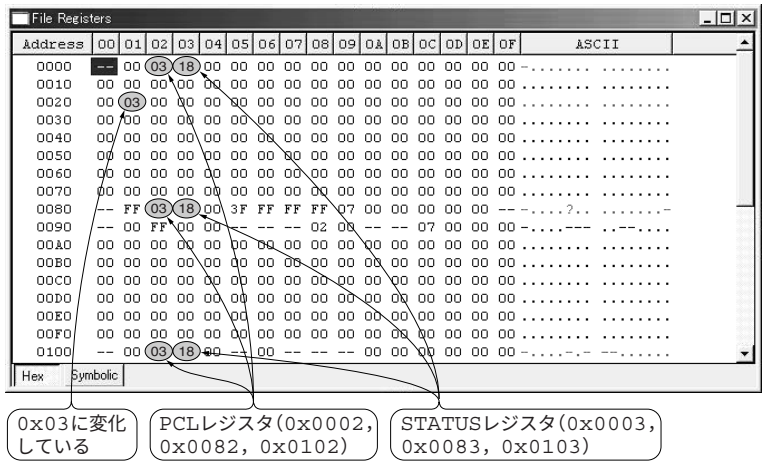

図3-5　File Registersウィンドウ（実行後）
リスト2-1のプログラムを実行後のファイル・レジスタの値．
前回にレジスタを表示していたときから変化があったところは赤く表示される．

では早速実行してみましょう．Debuggerタブから，Runの中のRunをクリックするか，ファンクション・キーF9を押します．すぐに，このプログラムは終わるので，Debugタブから，Runの中のHaltをクリックするか，ファンクション・キーF5を押し，シミュレーションを止めます．

Special Function Registerウィンドウのwの値とFile Registersウィンドウの0x21アドレスの値が3になりましたね(**図3-5**)．Symbolicタブをクリックすると**図3-6**のようにレジスタの名前が表示されるのでわかりやすいかもしれません．

リスト2-1のプログラムは，1+2を実行してwとREGA(0x21)に代入するプログラムです．数字を適当に変えてアセンブル，実行してみてください．wとアドレス0x21の値が変わりますね．

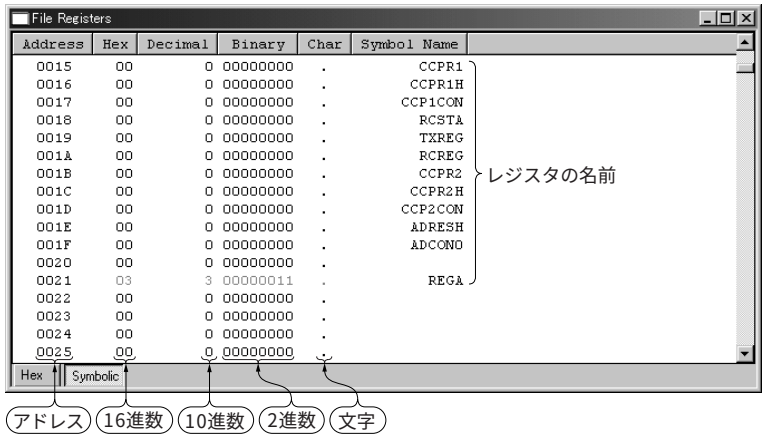

図3-6　File Registers ウィンドウ(実行後，シンボリック表記)
ファイル・レジスタの値は別の表示方法も選べる．こちらは一覧性はないが，SFR の表記のように一つの値については見やすい．

今度はステップ実行を試してみましょう．

Debug タブから，Run の中の Reset をクリックするかファンクション・キー F6 を押します．これで PIC がリセットされた状態になります．次は Debug タブから，Run の中の Step Into をクリックするかファンクション・キー F7 を押します．Step Into のクリックまたは F7 を押すたびに 1 命令ずつ実行されます．goto 命令のところでは，同じアドレスにジャンプしているので，プログラムは先へは進みません．

最後にアニメーション実行を試しましょう．

Debug タブから，Run の中の Reset をクリックするかファンクション・キー F6 を押し，リセットします．次は Debug タブから，Run の中の Animate をクリックします．Debug タブから，Run の中の Halt をクリックするか，ファンクション・キー F5 を押すとアニメーション実行は止まります．

三つのシミュレーションいずれも，リセットしなければ，止まったところから，シミュレーションを再開します．止まっているときには，レジスタの値を変えて動作がどうなるかを見ることもできます．たとえば，w レジスタの値を変えて 2＋3 や 4＋3 の結果を見ることができます．

うまくいかないときは，MPLAB をいったん終了し，ダウンロード・サービス(http://mycomputer.cqpub.co.jp/の本書のタイトルからリンク)から入手したファイルをプロジェクトの作成したファイルに上書きして，MPLAB を起動し直し，試してください．

Column … 1　プログラムの最後で無限ループにしているのは

プログラムの最後で無限ループにしているのは，次の行を実行してほしくないからです．

マイコンは常に次の命令を実行しようとします．プログラムを書いていないメモリ領域であっても実行します．PIC の場合にはプログラムを書いていない領域は addlw 0xff 命令が並んでいるため，それが実行されます．

これを防ぐには，無限ループで停止するのが簡単なのです．

実際のプログラムでは何かを実行しつづけるので，何もしない無限ループで停止にすることは少ないです．

3-2　アセンブラに慣れよう

ちょっと，わざとエラーを起こしてみましょう．

- ◆行頭の空白を消してみる．
- ◆ラベルを1文字変えてみる．
- ◆list の行のプロセッサ名を変えてみる．pic16f877z ではどうでしょうか．
- ◆include の行のファイル名を変えてみる．pic16f877x ではどうでしょうか．
- ◆命令を間違えてみる．movlw を movfw にしてみましょう．どうなりますか．
- ◆命令を間違えてみる．movlw を movlf にしてみましょう．
- ◆同じ名前のラベルをつくってみる．movlw の行頭に REGA と書いてみましょう．
- ◆同じ名前のラベルをつくってみる．movlw の行頭に STOP_HERE と書いてみましょう．
- ◆全角英数字や空白で記述してみる．
- ◆最後の goto をなくしてみる．
- ◆end をなくしてみる．

それぞれに，どういうエラー・メッセージが出ましたか．これで MPASM のエラーが出たら，どう直せばよいかわかりますね．movlw と movfw を間違えても，最後の goto をなくしてもエラーになりませんが，動作がおかしくなります．View タブから，Program Memory ウィンドウを出して，ステップ実行するとよくわかります．気をつけましょう．

3-3　加減算と論理演算をシミュレータで実行してみる

加減算と論理演算を実行して，STATUS レジスタの変化を見てみましょう．

リスト3-2をシミュレータでステップ実行してみてください．Special Function Registers ウィンドウを見ると，キャリ・フラグ（ボロー・フラグ，STATUS<0>）とzフラグ（STATUS<2>）が変化していくのがわかると思います．計算の順序を変えたり先に STATUS レジスタ内のフラグを変えたりして，ちゃんと計算によってフラグが変化していることを確かめてみてください．

プログラムはそのままでレジスタの値を変えて試すこともできるので，いろいろ試してみてください．

3-4　比較-基本は引き算-，条件分岐にはビット・テスト

1と2はどちらが大きいかは，すぐにわかりますね．でも PIC で1と2ではどちらが大きいかは，どうやって調べるのでしょうか．

そういうときには，引き算をして，結果が0か，正か，負かを STATUS レジスタで調べます．0なら，両者は一致，正なら引かれたほうが大きい，負なら引かれたほうが小さいということがわかります．

数字が大きい場合には REG_LARGER に1を，一致する場合には REG_EQUAL に1を，小さい場合には REG_SMALLER に1を代入するプログラムは**リスト3-3**のようになります．

プログラムでは，まず REG_LARGER，REG_EQUAL，REG_SMALLER を0にします．これはファイル・レジスタにリセット時には，どんな値が入っているかわからないからです．つぎに2と1を比較す

リスト3-2　加減算と論理演算

```
    list    p=16f877a         ; PIC16F877A 用のプログラムであることを宣言
    #include p16f877a.inc      ; PIC16F877A 用のヘッダ・ファイルを読み込む

REGA    equ 0x20

    bsf     STATUS, Z         ; STATUS レジスタの z フラグを 1 にする
    movlw   0                 ← w=0だがzフラグは変化しない

    movlw   D'1'              }
    addlw   D'1'              } w=1+1=2

    movlw   D'1'              }
    addlw   -D'1'             } w=1−1=0, z=1

    movlw   D'255'            }
    addlw   D'1'              } w=255+1→0, C=1

    movlw   D'2'              }
    sublw   D'1'              } w=1−2=−1, C=BO=0

    movlw   D'1'              }
    sublw   D'1'              } w=1−1=0, z=1

    movlw   D'1'              }
    sublw   D'2'              } w=2−1=1, C=BO=1

    movlw   0x0f              } 論理積
    andlw   0xff              } w=0x0f AND 0xff ← AND 0000 1111
                              }   =0x0f            1111 1111
                                                   ⇩共に1のところだけ1になる
                                                   0000 1111

    movlw   0x0f              } 論理和
    iorlw   0xff              } w=0x0f OR 0xff  ← OR 0000 1111
                              }   =0xff            1111 1111
                                                   ⇩どちらかが1なら1になる
                                                   1111 1111

    movlw   0x0f              } 排他的論理和
    xorlw   0xff              } w=0x0f XOR 0xff ← XOR 0000 1111
                              }   =0xf0            1111 1111
                                                   ⇩一方が1の場合のみ1となる
                                                   1111 0000

    movlw   0x0f              } 否定
    movwf   REGA              } w レジスタで演算でき ← 0000 1111
    comf    REGA, w           } ないので REGA に代入      ⇩0と1が入れ替わる
                              } してから演算する          1111 0000

    goto    $                 ← ここで無限ループで止まる

    end
```

るため，2−1の引き算を実行しています．そして，引き算の結果変化したSTATUSレジスタをチェックして，該当するレジスタを1にしています．

まず，zフラグをチェックします．zフラグが0ならば，次の行(goto)はスキップされ，1(引き算の結果が0)ならgotoが実行されます．2−1では0にならないので，スキップして，次はCフラグをチェックします．Cフラグは引き算の場合ボロー・フラグの反転なので，借り出しがあれば0，なければ1です．