[第2章]

PICマイコンの基本動作を理解する

PICをソフトウェアの面から知ろう

光永 法明

本章では，まずPICの概要と，PICの動く仕組みを紹介します．そして，PICのプログラミングに必要な基礎知識をソフトウェアの面から紹介していきます．

2-1　PICの中はどうなっている？

PICの中には，プログラム・メモリ（ROM），ファイル・レジスタ（RAM），計算ユニット，周辺回路が組み込まれています（**図2-1**）．プログラム・メモリは，楽譜や台本にあたります（**図2-2**）．少し違うのは，台詞の代わりに数字や計算の手順が書かれていることです．

計算ユニットは電卓にあたります．この電卓は，足し算と引き算，論理演算と呼ばれる計算ができますが，掛け算や割り算，メモリ機能がなく，桁数が少ないものです．

このアイコンは，章末に用語解説があります

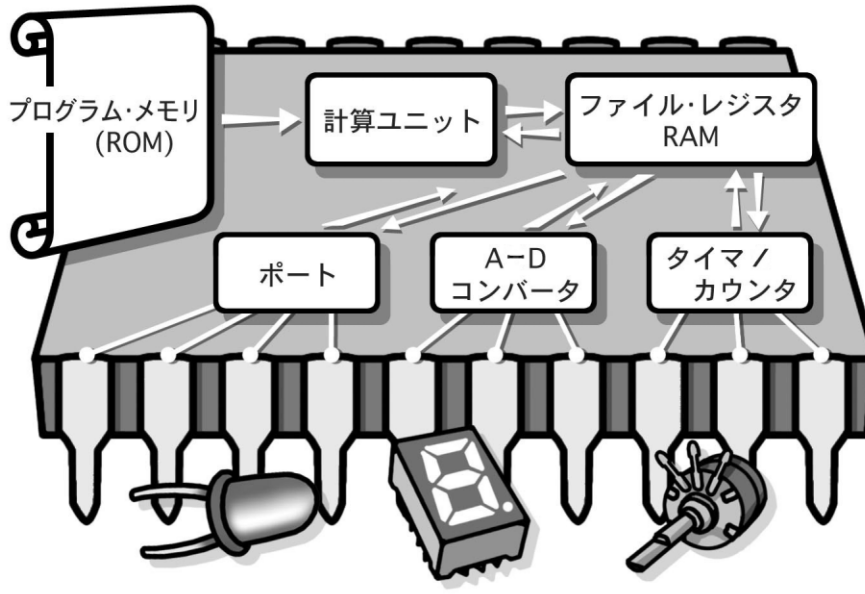

図2-1　PICを使った回路のイメージ図
PICの中には，プログラム・メモリ（ROM），ファイル・レジスタ（RAM），計算ユニット，周辺回路が組み込まれている．PICの外部にはLEDやボリュームなどをつける．

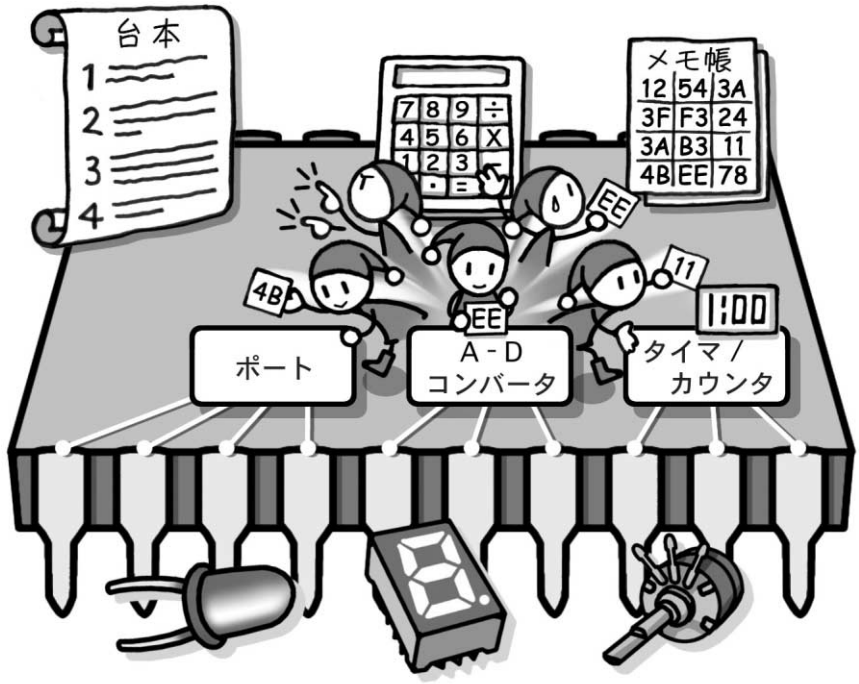

図2-2　PICの内部のイメージ図
PICの中には楽譜や台本にあたるプログラム・メモリと，電卓に相当する計算ユニット，升目があるメモ帳に相当するRAMがある．PICの中には台本と電卓，メモ帳にかかわる妖精さん(回路)，ポートにかかわる妖精さん，A-Dコンバータやタイマ・カウンタにかかわる妖精さんがいるようなもの．

ファイル・レジスタはメモ帳に当たります．このメモ帳は何度でも書き直しができるのですが，升目が切ってあり，そこに数字を書きます．周辺回路はメモ帳のある決まったところを見たり，書き込みをして外の世界とつなぐためのものです．

PICの中には妖精さんが何人かいると思ってください．

一番重要なのは，台本とメモ帳，電卓を見て，計算結果をメモ帳に書いてくれる妖精さんです．ただ，この妖精さんがメモ帳に書いたことをPICの外から直接見ることはできません．周辺回路の近くにいる妖精さんは，メモ帳の決まった升目を見て，外の世界にメモ帳の一部を見せたり，外のことをメモ帳に書く役目を受け持っています．PICが役に立つことができるのは，この妖精さんたちのおかげです．

2-2　プログラムはどうやって動くか？

プログラム・メモリ(台本)のどこを見たらよいかは，ファイル・レジスタ(メモ帳)のプログラム・カウンタ(PCレジスタ)に書かれています．妖精さんはプログラム・カウンタの指す位置の命令を読んで，必要に応じてメモ帳を見ます(**図2-3**)．そして，電卓をたたいた結果をメモ帳に書きます．計算の途中でメモ帳に書かなくてもよい場合には，電卓に表示したままにしておきます．この電卓の表示に当たるのがwレジスタです．

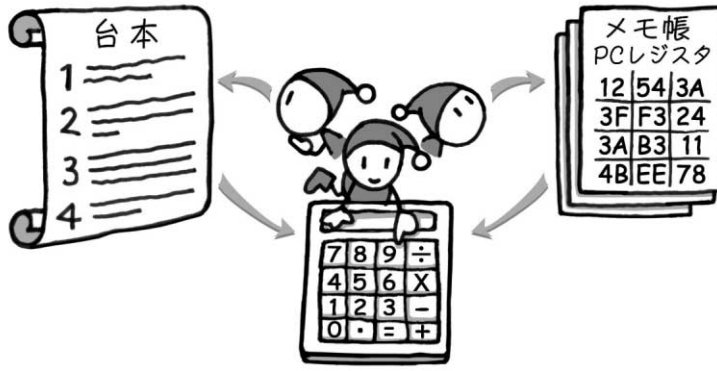

図2-3　PICの動作のイメージ図
PIC内の妖精さんは，ファイル・レジスタ(メモ帳)のプログラム・カウンタ(PCレジスタ)のさす，プログラム・メモリ(台本)の行に書かれた計算を電卓を使ってこなしていく．電卓に入れる数字や，メモのとり方は台本通りに行う．

妖精さんは暗算をせず必ず電卓を使うので，計算には必ずwレジスタが必要です．一つの命令の計算が終わったら，PCレジスタに1を足します．これが妖精さんの1回の仕事です．妖精さんは1回の仕事が終わったら，すぐにPCレジスタを見て次の仕事をこなします．

お客さんの反応に合わせて変化をつける台本だと，順に進むのではなく別のページに飛びたいことがあります．そういうときには，台本には分岐命令と呼ばれる命令を書いておきます．その命令のときには，妖精さんは普通の計算の代わりにPCレジスタについて計算します．妖精さんは前に台本のどこを見ていたかは覚えていませんから，PCレジスタが変わると台本の違うところを見ることになるのです．

PICは，電源が入った場合など，リセットされるとPCレジスタの値は0になります．そこで，プログラムは0番から順に書いていきます．また，PICの中にプログラムを書くには，ライタ，書き込み器などと呼ばれるツールとパソコン上のソフトを使います．

また，PICからLED✎を点灯したい場合や，A-Dコンバータ✎で値を読みたい場合などは，特定のファイル・レジスタ(メモ帳の特定の場所)に指示を書いたり，結果を読み込んだりして，ほかの妖精さんと連係して動作するのです．

2-3　アセンブリ言語と機械語

PICのプログラム・メモリには，機械語と呼ばれる言語で書かないといけません．PIC16F877Aの場合，

```
11000000000001
11111000000001
```

(**a**) 機械語で記述

```
movlw   0x1 ; wレジスタ(電卓)に1を入れる
addlw   0x1 ; 1をwレジスタに(電卓上で)足す
```

(**b**) アセンブリ言語

図2-4　1＋1の足し算をする機械語とアセンブリ言語の例
(a)は機械語で1＋1の足し算を書いた場合の数値，(b)はアセンブリ言語での書き方．PICには機械語が必要だが，アセンブリ言語のほうがわかりやすい．アセンブリ言語から機械語への変換はパソコン上でアセンブラを使って行う．

14ビット(0か1が14個並んでいる)の数値で表されます．これではわかりにくいので，もう少し人の言葉に近づけたのがアセンブリ言語です．

図2-4に1+1を行う(**a**)機械語と(**b**)アセンブリ言語を書いています．アセンブリ言語から，機械語を生成してくれるのがアセンブラです．本書ではMicrochip社のMPASMを使います．MPASMのアセンブリ言語には，PICの命令のほかに，ラベル，コメント，疑似命令を書くことができます．

2-4　アドレスとラベル

プログラム・メモリ上のどこを実行するよう妖精さんに指示をするかは，プログラム・カウンタで決まります．このプログラム上での位置をアドレス(番地)といいます．

一方，ファイル・レジスタの区別もアドレス(メモ帳の升の番号)で区別します．

機械語では，どちらも数字で表し，14個の0/1の中に埋め込みます．ところが，どちらのアドレスも一つ数え間違うとプログラムは正しく動いてくれません．そこでアセンブリ言語上では，ラベルというものが使えるようになっています．たとえば，プログラム・メモリの先頭から，先ほどの1+1の足し算を行い，計算結果をファイル・レジスタの0x20に書くことを繰り返すプログラムを書くと**リスト2-1**(**a**)となります．

一方，ラベルを使うと，**リスト2-1**(**b**)のように書くことができます．ファイル・レジスタの位置は疑似命令equを使って先頭に書かなければいけませんが，プログラム・メモリ上のplus_1_1のアドレスはアセンブラが自動で計算してくれるのです．今の場合は，プログラムの先頭が0番地だったので，数えるのは簡単でしたが，123番地へのジャンプを，プログラムが少し短くなったために115番地へ変更といったことをしなくて済むのです．

ファイル・レジスタの位置は疑似命令が必要で不便と思うかもしれませんが，アセンブラはメモ帳のどこを使うかを決めることができないので仕方ありません．ですが，メモした結果を後で取り出したいときに，アドレスの数値を覚えておくのと，ラベルを覚えておくのではラベルを覚えるほうが簡単です．携帯

リスト2-1　ラベルを使う場合と使わない場合

```
    movlw   0x1         ; wレジスタ(電卓)に1を入れる
    addlw   0x1         ; 1をwレジスタに(電卓上で)足す
    movwf   0x20        ; ファイル・レジスタのアドレス0x20にwレジスタの値を書く

    goto    0           ; 0番地へジャンプ
```

(**a**) ラベルを使わない場合

```
RESULT  equ 0x20          ; ラベル

plus_1_1                  ; ラベルplus_1_1を定義
    movlw   0x1           ; wレジスタ(電卓)に1を入れる
    addlw   0x1           ; 1をwレジスタに(電卓上で)足す
    movwf   RESULT        ; ファイル・レジスタRESULTにwレジスタの値を書く
    goto    plus_1_1      ; plus_1_1の番地へジャンプ
```

(**b**) ラベルを使う場合

表2-1　2進数，10進数，16進数のアセンブリ言語での書き方
2進数の場合にはB'11000001'のようにB'と'で囲む．10進数の場合にはD'12'のようにD'と'で囲む．16進数の場合には0xf1のように先頭に0xと書く．

2進数	10 01000010 10000000	B'10', B'010, B'00000010' B'1000010', B'01000010' B'10000000'
10進数	10 2 5 5 −10	D'10' D'255' -D'10'
16進数	10	0x10, (10h)

のアドレス帳機能のようなものです．

以下では，説明のときに，具体的なレジスタ・ファイルのアドレスを決める疑似命令を書かずにラベルを使うことがありますが，シミュレーションやプログラム内に書くときにはラベルの定義が必要になることを覚えておいてください．

2-5　数字の書き方（2進数，10進数，16進数）

MPASMでの2進数，10進数，16進数の数字は，**表2-1**のように書きます．2進数の場合にはB'とで'の間に数字を，10進数の場合にはD'とで'の間に数字を書きます．16進数の場合には数字の前に0xをつけます．

Microchip社のデータ・シートなどでは，数字の後にｈをつけることで16進数を表している場合があります．

10進数は普段使っていますね．10になると桁が一つ上がります．一つの桁は0から9までです．

2進数というのは2になると桁が一つ上がるので，一つの桁には0と1しかありません．

16進数の場合は16になると桁が一つ上がり，一つの桁には0から15があります．10から15を書くと2桁になって紛らわしいので10から15には，ａからｆのアルファベットを割り当てます．大文字小文字は好みでどちらを使ってもかまいません．

● 2進数

では2進数は，どういうときに便利なのでしょうか．ランプが8個並んでいて，それぞれ点灯と消灯が

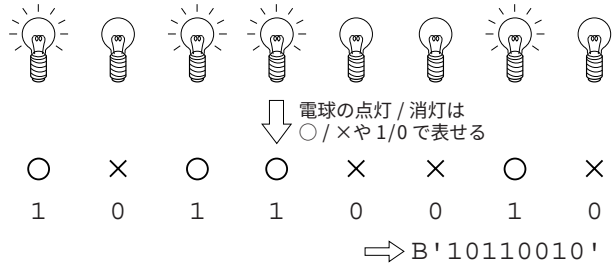

図2-5　ランプの点灯，消灯を2進数で表した例
ランプの点灯や消灯は，二つの状態で表せる．点灯/消灯を○と×で表すと，8個の○×で表せる．○×の代わりに1と0を使って並べると8桁の2進数になる．つまり8桁の2進数で8個の電球の状態が表せる．

できる場合を考えてみましょう．

一つずつの点灯/消灯を○と×で表すと，8個の○×を並べることでランプの状態を表現できます．○と×を１と０に置き換えると，2進数で8桁の数字としてランプの状態が表せることがわかります(**図2-5**)．

このときランプの状態は数字で表すので，10進数や16進数で表現することもできます．たとえば交互に点灯というのは2進数では，10101010ですが，10進数で204，16進数で0xccとなります．

2進数が直接でわかりやすいですね．

レジスタの一覧を見てみましょう．PIC16F877AはAppendix D(p.273)，PIC12F629/675は第7章**表7-4**(p.198)にあります．レジスタの1ビット✎ごとに役割が決められている場合がありますね．こういった場合にはランプと同じで2進数が便利ですね．

● 16進数

16進数はどうでしょうか．**表2-2**をみてください．2進数だと4桁ですが，16進数だと1桁で表せます．

PICでは8ビットの数字を扱うので，アセンブリ言語で書く場合も8ビットの数字を書きます．ですから，8桁と2桁の差になります．０や１が続くとタイプ・ミスや見間違えが増えるので，この表を覚えていれば，2進数で書くより16進数で書くほうがミスが少なくなる場合があるのです．

慣れないと難しいかもしれませんが，2進数の0000，0001，0010，0100，1000，1111がそれぞれ0x0，0x1，0x2，0x4，0x8，0xfであること，11111111が0xffであることは覚えておくとよいでしょう．

● 2進数から10進数へ

2進数を10進数に変換するには，各桁が，1，2，4，…，128に対応していることを利用して足し算で求めます(**表2-3**)．

たとえば，01010101なら1＋4＋16＋64＝85となります．

● 16進数から10進数へ

16進数を10進数に変換するには，各桁(0～f)を０から１５までの10進数に直し，上の桁に16を掛けて下の桁を足します．たとえば，0xa6なら10×16＋6＝166です(**図2-6**)．

2進数も一度16進数に変換して，10進に直すほうが早い場合もあると思います．

Column … 1　*n* 進数の変換

本文中に書いた変換は，8ビット(0～255)の場合の変換です．

一般の場合の *n* 進数の変換の手順は，

(1) 変換したい数値を *n* で割り，余りを一番下の桁におく

(2) 商を *n* で割り，余りを次の桁におく

(3) 商が0なら終わり，そうでないなら(2)に戻る

です．*n* が10のときは，簡単ですね．