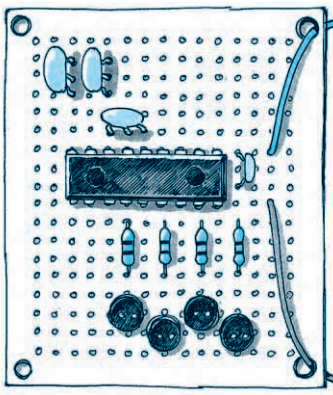

はじめてのPICマイコン入門〈第14回〉

LCDに放電電圧の時間変化を表示する
バッテリ放電特性モニタの製作(ソフトウェア編)

落合　正弘
Masahiro Ochiai

先月はハードウェア(**写真14－1**)を製作しました．今月はプログラムを制作します．

放電特性をモニタするためのプログラム

プログラムはそれぞれのブロックに分けて考えます．

● 初期化ルーチン

キャラクタ・モジュールの初期化，タイマの初期化，割り込みの準備，変数の初期化などを行います．

● キャラクタ・モジュールの初期化ルーチン

キャラクタ・モジュールで決められている初期化シーケンスにしたがって，キャラクタ・モジュールにコマンドを送ります．これを行わないと液晶画面に文字を表示させられません．より詳しい使いかたについては本誌2004年2月号の特集を，参照してください．

● 時間のカウント・ルーチン

▶タイマ1を使い水晶発振子を基準に1秒間隔の割り込みを作る

時間のカウントはタイマ割り込みを使って行います．今回の測定単位は秒なので，32.768 kHzの水晶発振子を使います．

PIC16F876はいくつかのタイマ・モジュールを内蔵していますが，タイマ1だけがこの水晶発振子を基準にカウントできます．**図14－1**にタイマ1のブロック・ダイアグラムを示します．青色矢印のように動作させるプログラムを**リスト14－1**に示します．

タイマ1は16ビット幅のタイマです．ちょうど32768をカウントすれば1秒となります．割り込みハンドラ内でタイマ値を8000hに更新すれば，1秒間隔で割り込みが発生します．

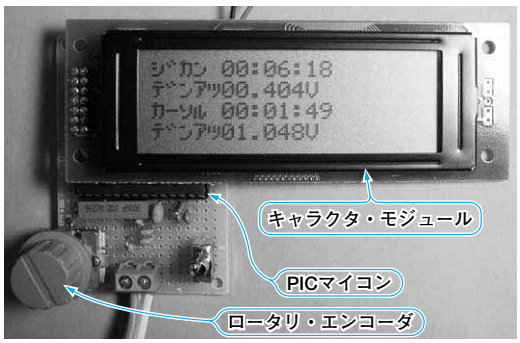

写真14－1　前回製作した放電特性モニタの外観

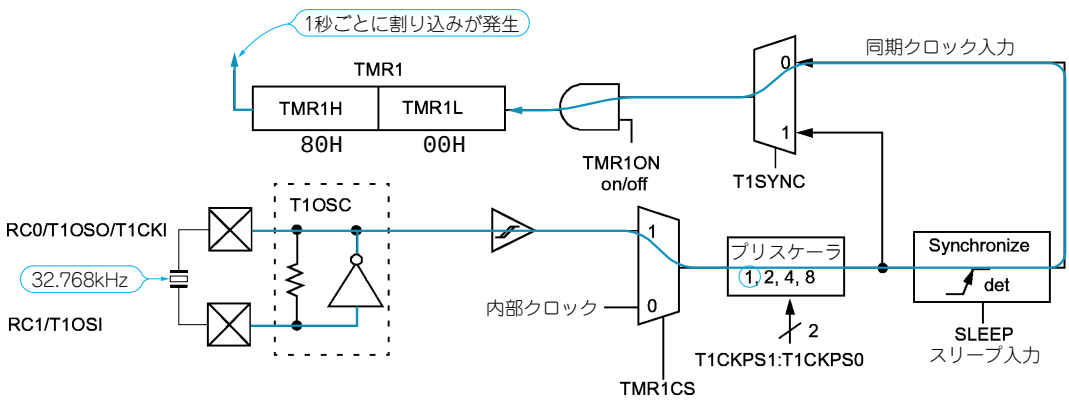

図14－1(1)　タイマ1のブロック・ダイアグラム

リスト14－1　タイマ1を図14－1の青色矢印のように設定する方法

```
MOVLW       00001011b
MOVWF       T1CON
```

リスト14－2　PICマイコンで24ビット変数を扱う方法

```
MOVLW       1               ; はじめに1をWレジスタにセットする
ADDWF       now1,F          ; 下位8ビットに加算する
BTFSC       STATUS,C        ; キャリーが発生していないときはスキップ
ADDWF       now2,F          ; 中位8ビットを加算する
BTFSC       STATUS,C        ; キャリーが発生していないときはスキップ
ADDWF       now3,F          ; 上位8ビットを加算する
```

▶1秒間隔の割り込みを使い時間をカウントする

この1秒をファイル・レジスタでさらにカウントしていきます．ここでどのくらいの間，測定できるようにするかを考えます．1日中測定できるようにすると，1日は86400秒なので，16ビット幅（約65500）では足りないことになります．試験する電池が放電に1日以上かかることもありますから，24ビット（約1600万）でカウントするようにしました．24ビットあれば約200日測定できる計算になります．ただし，液晶表示の都合上，最大は99時間としています．

▶PICマイコンで24ビット変数を扱う方法

PICマイコンは8ビット・マイコンですから24ビット変数を直接扱うことはできません．しかし命令を組み合わせることで何ビットの変数でも計算できます．実際に作ったプログラムを**リスト14－2**に示します．演算結果がフラグに反映されないためINCF命令は使いません．同じように減算でも行うことができます．減算の場合は，ADDWFがSUBWFになり，BTFSCはBTFSSになります．

● バッテリの電圧値を読み込むA－D変換ルーチン

タイマ割り込みの中でA－D変換を開始させます．次の割り込み発生時にA－D変換の結果を作業領域に保存した後，次のA－D変換を開始します．メイン・ルーチンでその保存値を読み取り，EEPROMにデータを書き込みます．

▶レジスタの設定

A－D変換の設定はADCON0とADCON1レジスタで行います．ADCON1はアナログ入力でリファレンス電圧の設定を，ADCON0はA－D変換チャネルで変換の開始などの制御を行います．今回は4.096 Vのリファレンス電圧を使うので，PCFG3とPCFG0には0101bをセットします．

▶10ビットの変換結果を16ビット幅の変数に格納する方法

A－D変換結果は10ビットですが，変数は16ビット幅なので，右詰め（上位6ビットを0）にするか左詰め（下位6ビットを0）にするかを選ぶことができます．今回はプログラムしやすいので右詰め（ADFM＝1）を選びました．下位がADRESLに，上位がADRESHレジスタに入ります．

変換結果は10ビットの数値となります．範囲としては0～1023ですが，電圧に換算すると0～4.095 Vとなります．変換値を4倍すれば，表示に適した数値となります．入力端子を抵抗で分圧している場合は，その分をさらにN倍します．

● EEPROM書き込みルーチン

EEPROMの通信は2線で行います．データ線（SDA）は通信中に出力になったり，入力になったりするので制御はちょっと複雑です．でも実際はPICマイコンに内蔵されたI^2Cインターフェースを使うことで，コントロールはマイコン側で処理してくれます．プログラムは簡潔になり，ソフトウェアで細かい制御をする必要がなくなります．

● ロータリ・エンコーダ操作ルーチン

▶操作の検出はポートの変化割り込みを利用

ロータリ・エンコーダを操作したら，操作した場所のデータをEEPROMから読み取り，キャラクタ・モジュールに表示します．ロータリ・エンコーダはRBポートに接続しています．ポートの変化割り込みを利用することで，CPUはRBポートを監視していなくても，ユーザがエンコーダを回すだけで，操作ルーチンにジャンプします．

▶右回しと左回しの検出方法

ロータリ・エンコーダを回すと**図14－2**のようなパルスが出力されます．2組あるのは回転方向を知るためです．1相ではどのくらい回転したかしか検出できず，左右どちらに回転しているかを区別することはできませんが，2相の場合はそれを行うことができます．

プログラムとしては**図14－2**のように直前のエンコーダの状態との排他的論理和（XOR）をとります．これにより右に回ったのか左に回ったのかを判定できるのです．

ロータリ・エンコーダの処理は速やかに行わないと，反応が遅くなったり逆回転と認識してしまうことがあるので，割り込みを使うなどして，操作があったらすぐにCPUが反応するようにします．

▶EEPROM書き込み中には割り込み処理を行わない

割り込み処理では現在位置を更新するだけで，EEPROMからはデータを読み出しません．これはEEPROMにデータを書き込んでいる間に割り込み処理に入ってきている可能性があるからです．書き込ん