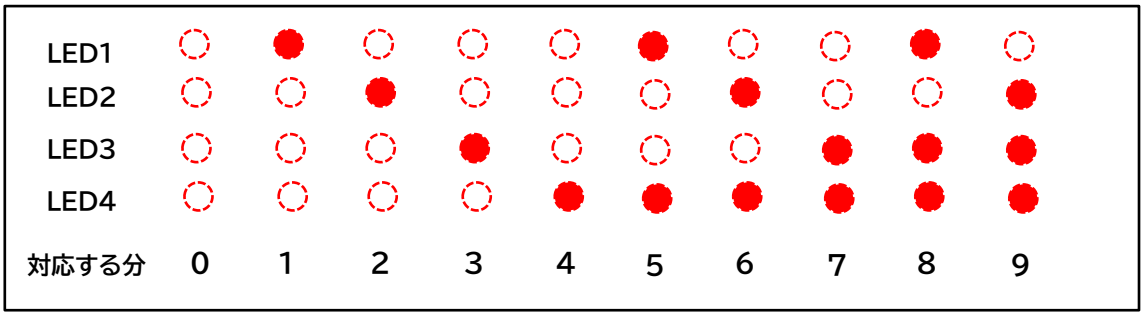
RL78/G23-64PFPB で Multi-function シールドを使う(2)

前回、「RL78/G23-64PFPB で Multi-function シールドを使う」で、7セグ LED の左端の桁が表示しない問題があると書きました。今回は、その状態でもスケッチをきちんと動作させる方法について解説します。

具体的には、Multi-function シールドの LED1～LED4 を使って、分の情報を表示するようにします。1 分経過したら、一番上の LED1 を点灯させます。さらに 1 分経過したら、点灯する LED を 1 つ下に動かして LED2 を点灯させます。次の 1 分でさらに下に動かし、LED3 を点灯させます。次の 1 分でさらに下に動かし、LED4 を点灯させます。LED は LED4 までしかないので、次の 1 分経過で LED1 を追加で点灯させます(LED1 と LED4 が点灯しています)。さらに 1 分経過して 6 分経過すると LED1 の点灯が下に動いて LED2(と LED4)が点灯します。7 分経過すると LED2 の点灯が下に動いて LED3(と LED4)が点灯します。8 分経過で、LED1 が追加で点灯します。9 分経過すると、LED1 の点灯が下に動いて LED2(と LED3,LED4)が点灯します。10 分経過したら、LED1～LED4 は全て消灯します。イメージ的には点灯した状態が上から下に動き、下側にたまっていくように動きになります。

これを図示すると、下のようなイメージになります。



LED は端子 10～13 に接続されているので、これを制御するために以下のような端子の定義を行います。

```
#define LED1_PIN   13          // assign D13 pin for LED1
#define LED2_PIN   12          // assign D12 pin for LED2
#define LED3_PIN   11          // assign D11 pin for LED3
#define LED4_PIN   10          // assign D10 pin for LED4
```

そのうえで、Setup()では、以下のように出力に設定します。

```
    pinMode(LED1_PIN, OUTPUT);      // set D13pin to output mode
    pinMode(LED2_PIN, OUTPUT);      // set D12pin to output mode
    pinMode(LED3_PIN, OUTPUT);      // set D11pin to output mode
    pinMode(LED4_PIN, OUTPUT);      // set D10pin to output mode
```

また、LED に表示するデータのパターンを配列で定義しておきます。LED はアノードが電源に接続され、カソードが端子に接続されているので、配列で 0 のビットが点灯することを示します。

```
const byte LED_DATA[] =
{
  0xFF,      // output data for 0
  0xF7,      // output data for 1
  0xFB,      // output data for 2
  0xFD,      // output data for 3
  0xFE,      // output data for 4
  0xF6,      // output data for 5
  0xFA,      // output data for 6
  0xFC,      // output data for 7
  0xF4,      // output data for 8
  0xF8,      // output data for 9
};
```

なお、ここで定義したデータ(下位 4 ビット)は LSB から順に LED4,LED3,LED2,LED1 となるように処理します。つまり、1 に対するデータはビット 3 が 0 なので、一番上の LED1 が点灯することになります。処理手順を逆にすると点灯の動きが逆になります。

実際にこれを制御するために loop()でローカル変数として" byte mdata;"を定義しておきます。これを使って制御するために、switch 文の「分の表示」を制御している"default"のところに処理を追加するだけです。

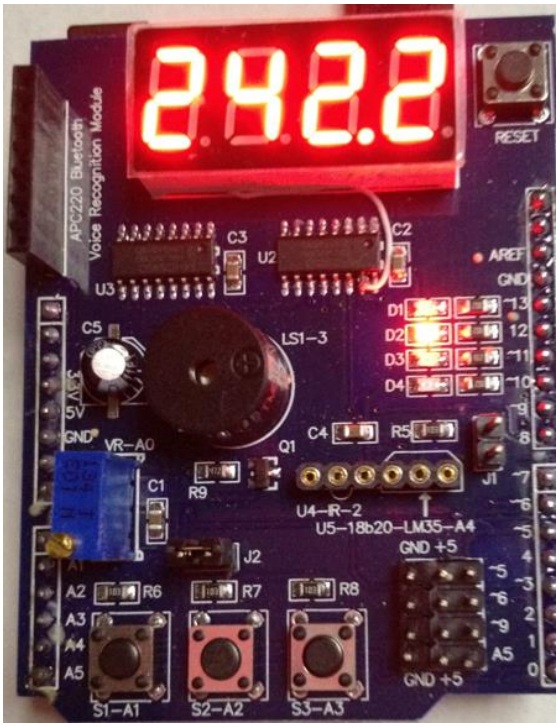
「分」の表示データの具体的な処理内容を以下に示します。赤字の部分が追加された処理です。

```
        default:                                // minute digit timing
            dt_wk = mini_data;                  // get minute display data
            seg_data = LED_SEGMENT[dt_wk];      // get segment of minute digit
            com_sel = LED_DIGIT[0];             // select minute digit

            mdata = LED_DATA[dt_wk];            // get data for minute LED

            digitalWrite( LED4_PIN, (mdata & 1 ));
            mdata >>=1;
            digitalWrite( LED3_PIN, (mdata & 1 ));
            mdata >>=1;
            digitalWrite( LED2_PIN, (mdata & 1 ));
            mdata >>=1;
            digitalWrite( LED1_PIN, (mdata & 1 ));
```

「分」のデータ(0～9)を用いて LED のデータパターンのテーブルを引くことで、mdata の下位 4 ビットに LED の点灯パターンが入ります。mdata の値を LSB からチェックして LED に出力しシフトしていくことで LED4～LED1 を目的のパターンで点灯させます。

このように、処理を追加しただけなので、分の桁の7セグ LED がきちんと点灯する場合にはこれに追加して LED1～LED4 が点灯します。

下の写真では分かりにくいですが、LED2 だけが点灯している状態です(目で直接見る分にはきれいに判別できます)。



分の桁の7セグ LED での表示が邪魔なら、”seg_data”に代入する値を”0xFF”とすることで、表示は空白となります。

これで、RL78/G14-FPB のサンプル・コードから始まったデジタル入出力関連のスケッチの移植は完了です。次回からは、新たなスケッチに移るので、最後にブザー機能を追加しておこうと思います。Multi-function シールドには自励式のブザーが搭載されています。Arduino のスケッチでは、圧電スピーカーはオーバーヘッドがかかり過ぎるので、自励式のブザーは妥当なところかと思います。自励式のブザーは、端子 3 に接続されていて、ローに引くと音が出ます。

実際のスケッチですが、まず、以下のように端子の定義を行います。

```
#define BEEP_PIN   3              // assign D3 pin for beep
```

Setup()には、以下の処理を追加します。

```
    pinMode(BEEP_PIN, OUTPUT);     // set D3pin to output mode
    digitalWrite( BEEP_PIN, HIGH );
```

ブザーはスイッチを押したときに、短く鳴動するようにします。ブザーを鳴動させる時間を計測するための loop()でローカル変数として” beep_time”を定義しておきます。

```
        byte beep_time = 0x00;                  // beep counter
```

実際のブザーの制御は、20ms ごとにスイッチの状態をチェックしているところで制御することになります。下にその部分を示します。赤字で示したのが追加したブザーの制御を行っている部分です。20ms×5 回で 100ms 鳴動するようにしています。

```
/*--------------------------------------
    switch data check timing
--------------------------------------*/

        if ( ++prescale4sw >= 5 )         // check SW check timing or not
        {                                 // 20milli seconds passing
            prescale4sw = 0x00;           // clear timing counter
            sw_data <<= 1;                // shift olddata left
            sw_data  += ( digitalRead(ex_swPin)&digitalRead(swPin) ); // read sw
data
            sw_work = (sw_data & 0x06); // extract check timing data
            if ( 0x04 == sw_work )
            {                             // sw is pushed
                time_mode ^= 0x01;        // change count mode flag
                beep_time = 5;
                digitalWrite( BEEP_PIN, LOW );
            }
            if ( 0x00 == beep_time )
            {
                digitalWrite( BEEP_PIN, HIGH );
            }
            else
            {
                --beep_time;
            }
        }
```

これで、スイッチを押すとピッと短くブザーが鳴ります。(少しうるさいです。)

これで、Multi-function シールドに搭載された機能で使用していないのは、アナログ入力によるポテンショメータの状態の読み出しくらいになります。

今回は以下に示す 3 つのスケッチを作成(単に機能の一部追加ですが)しました。

sketch_apr25a.ino	:分情報を LED で表示する
sketch_apr25b.ino	:分情報を LED で表示する(7セグ LED の分は消灯)
sketch_apr26a.ino	:sketch_apr25a にブザーを追加

次回に何をやるかは、連休中に考えておきます。

以上