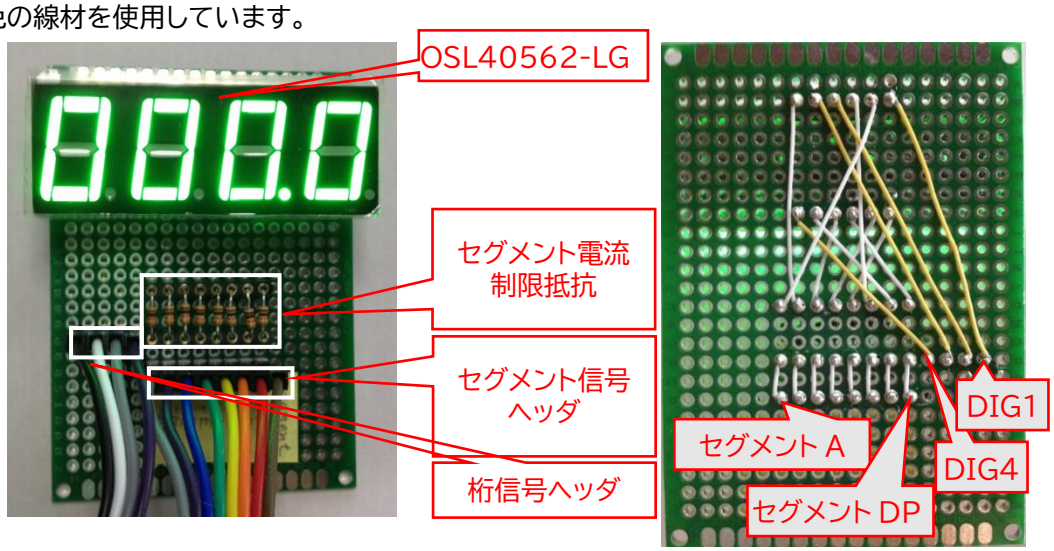
RL78/G14-FPB で遊ぶ(1)を RL78/G23-64PFPB に移植

RL78/G23-64PFPB の Arduino IDE 環境がリリースされたので、RL78/G14-FPB のプログラムを移植してみようかと思います。

今回は、ハードウェアを作り直しました。ユニバーサルボードをガラスエポキシで両面スルーホールのものに変えました。左が部品面、右が配線面です。セグメント信号は白色、桁(コモン)信号は黄色の線材を使用しています。



信号の接続を下の表に示します。OSL40562-Lx の桁は左から順に DIG1、DIG2、DIG3、DIG4 と並んでいるとデータシートには記載されていました。以前は DIG2 と DIG4 が入れ替わっていたのですが、今回使用した新しいものはこの通りでした。

信号名	OSL40562-Lx のピン番号	G23-64PFPB の信号
セグメント A	11	D0
セグメント B	7	D1
セグメント C	4	D2
セグメント D	2	D3
セグメント E	1	D4
セグメント F	10	D5
セグメント G	5	D6
セグメント DP	3	D7
DIG1	12	D11
DIG2	9	D10
DIG3	8	D9
DIG4	6	D8

セグメント信号は 1kΩのセグメント電流制限抵抗を介して G23-64PFPB と接続し、桁(DIG)信号は直接 G23-64PFPB と接続しています。

以上がハードウェア関係の内容です。次にプログラムの見直し内容です。

プログラムで今回移植するのは、AR_SKETCH.c だけで済みます。Arduino エディタで新規ファイルを選択し、最初の行の前に、端子の定義と変数、定数部分をコピーします。ボードが変わったことで、オンボードスイッチの接続端子が 26 に変更されているので、そこを修正します。

以下に端子の定義部分を示します。

```
/*****************************************************************
Global variables and functions
*****************************************************************/

#define LED_OFF ( 0x01 )
#define LED_ON  ( 0x00 )

/*---------------------------------------------------------------
      Definition area. Write pin definition here.
---------------------------------------------------------------*/
int segPinA = 0;                  // assign D0 pin to segPinA for 7SEG_LED.
int segPinB = 1;                  // assign D1 pin to segPinB for 7SEG_LED.
int segPinC = 2;                  // assign D2 pin to segPinC for 7SEG_LED.
int segPinD = 3;                  // assign D3 pin to segPinD for 7SEG_LED.
int segPinE = 4;                  // assign D4 pin to segPinE for 7SEG_LED.
int segPinF = 5;                  // assign D5 pin to segPinF for 7SEG_LED.
int segPinG = 6;                  // assign D6 pin to segPinG for 7SEG_LED.
int segPinDP = 7;                 // assign D7 pin to segPinDP for 7SEG_LED.

int comPin0 = 8;                  // assign D8 pin to comPin0 for 7SEG_LED.
int comPin1 = 9;                  // assign D9 pin to comPin1 for 7SEG_LED.
int comPin2 = 10;                 // assign D10 pin to comPin2 for 7SEG_LED.
int comPin3 = 11;                 // assign D11 pin to comPin3 for DOT_LEDs.

int ex_swPin = 13;                // assign D13 pin to ex_swPin for external SW.
int swPin = 26;                   // assign D26 pin to swPin for SW_USER.
```

以下に示すのは、セグメントデータのテーブル部です。セグメント A が LSB で、セグメント DP が MSB として正論理で定義しています。

```
const int SEG_TABLE[]=
{                                 // segment data table for 7seg_LED
    0x3F,                         // data "0"
    0x06,                         // data "1"
    0x5B,                         // data "2"
    0x4F,                         // data "3"
    0x66,                         // data "4"
    0x6D,                         // data "5"
    0x7D,                         // data "6"
    0x27,                         // data "7"
    0x7F,                         // data "8"
    0x6F,                         // data "9"
    0x00                          // data " "
};
```

次に示すのは、プログラム(スケッチ)で使用する変数等の定義部分です。

```
int old_time = 0x0000;              // previous time(milli sec.)

char mini_data = 0;                 // minute data 0:1:3:7:F:1F:3F:7F:FF
char sec_data = 0;                  // second count data(0 to 59)
char precount1 = 0;                 // prescaler for 100milli seconds.(25)
char precount2 = 0;                 // prescaler(10) for 1second
char digi_sel = 0;                  // digit select

char prescale4sw = 0;               // prescaler for SW check( up to 5:20milli sec.)
int sw_data = 0xFFFF;               // SW data image(history of 20 milli sec. interval)
int time_mode = 0;                  // o:stop/1:run

void set_SEG( char );               // set segment data
```

次に、setup 部分をそのままコピーします。「// put your setup code here, to run once:」の後ろに以下の内容を貼り付けます。

```
    pinMode(segPinA, OUTPUT);               // set D0pin to output mode
    pinMode(segPinB, OUTPUT);               // set D1pin to output mode
    pinMode(segPinC, OUTPUT);               // set D2pin to output mode
    pinMode(segPinD, OUTPUT);               // set D3pin to output mode
    pinMode(segPinE, OUTPUT);               // set D4pin to output mode
    pinMode(segPinF, OUTPUT);               // set D5pin to output mode
    pinMode(segPinG, OUTPUT);               // set D6pin to output mode
    pinMode(segPinDP, OUTPUT);              // set D7pin to output mode
    pinMode(comPin0, OUTPUT);               // set D8pin to output mode
    pinMode(comPin1, OUTPUT);               // set D9pin to output mode
    pinMode(comPin2, OUTPUT);               // set D10pin to output mode
    pinMode(comPin3, OUTPUT);               // set D11pin to output mode
    pinMode(ex_swPin, INPUT_PULLUP);        // set D13pin to input mode
    pinMode(swPin, INPUT);                  // set D18pin to input mode
```

同様に loop 部分もコピーします。

loop の頭の部分は以下のようになります。

```
    int time_work;                          // present time buffer
    char com_sel;                           // common select signal
    char seg_data;                          // segment data
    char sw_work;                           // work for switch check

/*--------------------------------------
    wait for 4milli seconds interval.
--------------------------------------*/

    time_work = ( int )( millis() & 0x0FFC ); // read milli sec data

    if ( old_time != time_work )            // check timing of change LED
    {

/*--------------------------------------
    every 4milli seconds timing
    change display digit.
--------------------------------------*/
```

次の部分が、millis()関数を使って、4ms ごとに実行する表示関係の処理です。4ms ごとに、7セグ LED の各桁を表示しています。LED の配線が変更になったので、下記の赤で示したように、case 0x00 で"comPin0"→"comPin2"に、case 0x08 で "comPin0"→"comPin2"に変更してあります。

```
old_time = time_work;              // change current time data
time_work &= 0x00C;                  // get comsel data(= 4milli second unit)

switch ( time_work )
{
    case 0x00:                       // 10second digit timing
        digitalWrite(comPin3,LED_OFF);   // turn off minute digit
        seg_data = SEG_TABLE[(sec_data / 10)]; // get 10second digit data
        set_SEG( seg_data );      // set segment data
        com_sel = comPin2;         // set 10second digit
        break;

    case 0x04:                       // second digit timing
        digitalWrite(comPin2,LED_OFF);   // turn off 10second digit
        seg_data = SEG_TABLE[(sec_data % 10)]; // get second digit data
        set_SEG( ( seg_data + 0x80 ) ); // set segment data
        com_sel = comPin1;         // set second digit
        break;

    case 0x08:                       // second digit timing
        digitalWrite(comPin1,LED_OFF);   // turn off second digit
        seg_data = SEG_TABLE[precount2]; // get second digit data
        set_SEG( seg_data );      // set segment data
        com_sel = comPin0;         // set 100milli second digit
        break;

    default:                        // minute digit timing
        digitalWrite(comPin0,LED_OFF);   // turn off second digit
        seg_data = SEG_TABLE[mini_data];    // get minute display data
        set_SEG( seg_data );      // set segment data
        com_sel = comPin3;         // set minute digit


}
digitalWrite(com_sel,LED_ON);    // enable display
```

次の部分は、計時処理を行っています。

```
/*-------------------------------------
    every 4milli seconds timing
    if count enable, then count time up.
-------------------------------------*/

        if (( 25 <= ++precount1 ) && ( 1 == time_mode ))
        {                                   // 100 milli seconds passing
            precount1 = 0;                  // clear 100milli second counter
            if (   10 <= ++precount2 )    // count for seconds
            {                               // one second passing
                precount2 = 0;              // clear one second counter
                if ( 60 <= ++sec_data )
                {                           // one minute passing
                    sec_data = 0;
                    if ( 9 == mini_data )
                    {                       // over flow
                        mini_data = 0;   // clear minute data
                    }
                    else
                    {                       // change minute data
                        ++mini_data;    // count minute data
                    }
                }
            }
        }
```

loop の最後の処理は、スイッチ入力部分です。20ms ごとに、スイッチの状態をサンプリングすることで、チャタリング対応を行っています。

```
/*-------------------------------------
    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
            }
        }

    }
```

以上が loop での処理です。

最後に、loop の後ろのある set_SEG 部分をコピーします。ここは、表示する桁のセグメントの値を引数として、セグメント信号のドライブを行います。

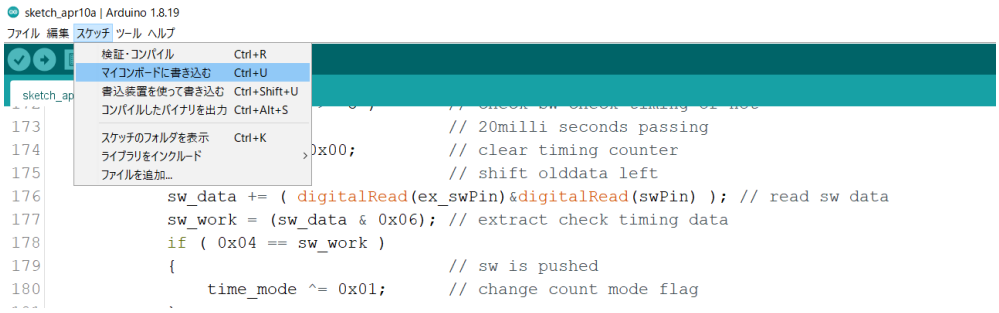
```
/*---------------------------------------------
    set segment data function
---------------------------------------------*/
void set_SEG( char data )
{
    char work_data;

    work_data = data;                       // set segment data

    digitalWrite(segPinA,( work_data & 0x01 )); // set segmentA data
    work_data >>= 1;                        // move next bit
    digitalWrite(segPinB,( work_data & 0x01 )); // set segmentB data
    work_data >>= 1;                        // move next bit
    digitalWrite(segPinC,( work_data & 0x01 )); // set segmentC data
    work_data >>= 1;                        // move next bit
    digitalWrite(segPinD,( work_data & 0x01 )); // set segmentD data
    work_data >>= 1;                        // move next bit
    digitalWrite(segPinE,( work_data & 0x01 )); // set segmentE data
    work_data >>= 1;                        // move next bit
    digitalWrite(segPinF,( work_data & 0x01 )); // set segmentF data
    work_data >>= 1;                        // move next bit
    digitalWrite(segPinG,( work_data & 0x01 )); // set segmentG data
    work_data >>= 1;                        // move next bit
    digitalWrite(segPinDP,( work_data & 0x01 ));// set segmentDP data

}
```

コピーが完了したら、以下に示すように「スケッチ」「マイコンボードに書き込む」と選択するとコンパイルして RL78/G23 に書き込んでくれます。最初だと 3 分近くかかります。ここらは使っている PC に依存します。



書き込みが完了すると、最初のページの左側の写真のような表示になります。その状態でオンボードのスイッチを押すと、カウントを開始します。再度、スイッチを押すとカウントを停止します。以降、スイッチを押すたびにカウントと停止を繰り返します。

スケッチ全体(sketch_apr10a.ino)は、ディレクトリごと zip で圧縮し” sketch_apr10a.zip”として添付してあります。解凍して、” sketch_apr10a.ino”を実行させてください。

次回は、このスケッチを”MultiFuncShield”に移植してみます。

以上