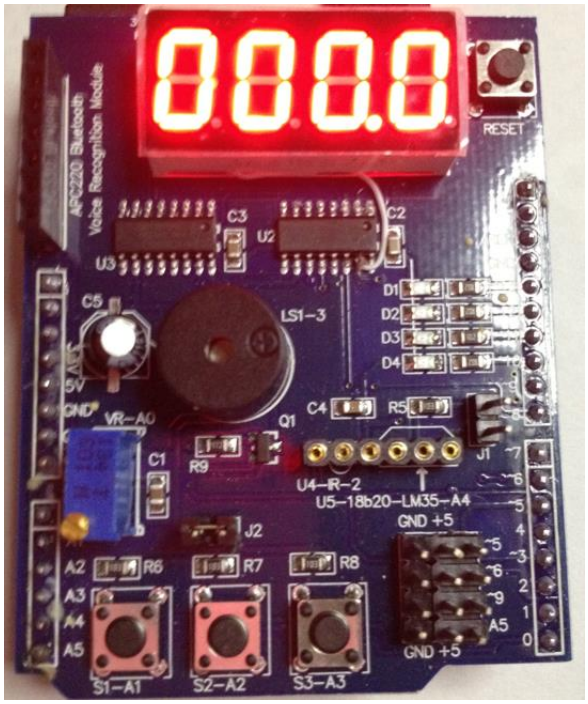
RL78/G23-64PFPB で Multi-function シールドを使う

はんだ付けが苦手で、二の足を踏んでいる人もいるとおもうので、出来合いのボード（シールド）を使ってみようということで、Multi-function シールドを使ってみました。

残念なことに、以下の Web ページにあるように、7セグ LED の左端の桁が点灯しませんでした。

Multi-function Shield（シールド）その１．｜マイコンと電子工作の部屋（wordpress.com）

そこで、付線（ジャンパー配線）を追加して動作させることができたのですが、下の写真に示すように、1.78mm ピッチの IC のピンにはんだ付けが必要になります。これは、2.54mm のユニバーサル・ボードでの配線より難しいかもしれません。もっとも、RL78/G11 のハーフピッチ（1.27mm ピッチ）よりは 0.5mm は広いのですが。



そこは、付線なしで（3 桁だけ使ったスケッチを）考えてみます。そもそも、RL78/G14FPB のオリジナルのサンプルコードは 3 桁の7セグ LED と 8 個の LED を使ったものです。分の桁を別の方法で表示すれば問題ありません。これについては、次回の投稿で説明します。

具体的なスケッチですが、Multi-function シールドの7セグ LED は、端子(ポート)を直に操作するのではなく、シリアルーパラレル変換のシフトレジスタ(74HC595)を 2 個使用し、セグメント用データと点灯させる桁のデータの 2 バイトのデータを送信することで各桁を点灯させるようになっています。

具体的な制御方法は、最初の 1 バイトでセグメント用データをセグメント DP～セグメント A の順に負論理のデータを送信し、2 バイト目の最後の 4 ビットで点灯させたい桁だけを 1 にして送信します。このようにして送信した 16 ビットのデータをラッチクロックへの立ちあがりエッジで HC595 の出力ラッチに転送して出力します。この出力で 7 セグ LED をドライブします。この動作を繰り返すことで、4桁の表示を行います。

前回の「RL78/G14-FPB で遊ぶ(1)を RL78/G23-64PFPB に移植」で作成したスケッチの表示制御部分(下に示す)だけを書き換えるだけです。

```
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
```

上記の処理では、switch 文の各 case 処理で、桁選択信号をオフしてから、表示したいセグメント・データをセットし、桁の選択信号を準備するだけになっています。switch 文の処理が完了し

た後で桁選択信号を出力して表示するようになっています。それに対して、Multi-function シールド用の処理は下記のように、case 文の中では単にセグメント用データと桁選択用データを準備するだけになっています。

```
        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
                dt_wk = ( sec_data / 10 );        // get 10second digit data
                seg_data = LED_SEGMENT[dt_wk];   // set segment of 10second digit
data
                com_sel = LED_DIGIT[1];            // set 10second digit
                break;

            case 0x04:                      // second digit timing
                dt_wk = ( sec_data % 10 );        // get second digit data
                seg_data = LED_SEGMENT[dt_wk];   // set segment of second digit data
                seg_data &= 0xFE;                  // set point
                com_sel = LED_DIGIT[2];            // set second digit
                break;

            case 0x08:                      // 100milli second digit timing
                dt_wk = precount2;                 // get 100milli second digit data
                seg_data = LED_SEGMENT[dt_wk];   // set segment of 100milli second
digit data
                com_sel = LED_DIGIT[3];            // set 100milli second digit
                break;

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


        }

        digitalWrite( SS_PIN, LOW );        // turn on SS signal(low level)

        shiftOut (MOSI_PIN, SCK_PIN, LSBFIRST, seg_data); // segment data out

        shiftOut (MOSI_PIN, SCK_PIN, LSBFIRST, com_sel);   // digit data out

        digitalWrite( SS_PIN, HIGH ); // turn off SS signal(rise up)
```

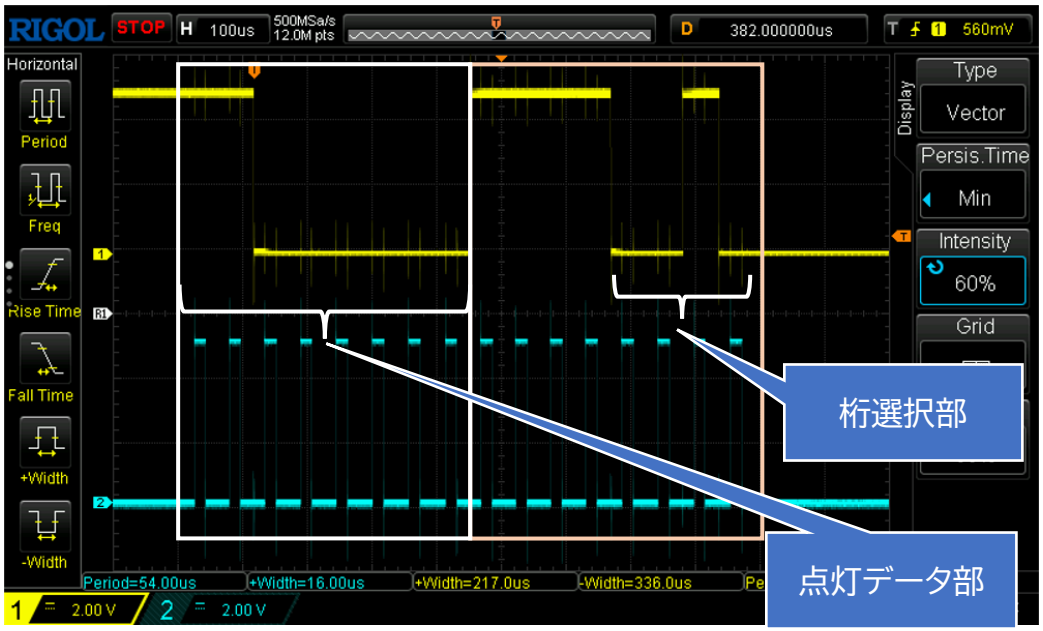
セグメント用データのテーブル(LED_SEGMENT)の参照用に dt_wk という変数を使ったのは、単に、1 行の文字数を少なくするためです。

switch 文が終了した後が HC595 を実際に制御している部分です。最初に、SS_PIN を立ち下げてから、shiftOut 関数を使って LSB ファーストで送信しています(オリジナルのライブラリは MSB ファーストだったのですが、同じでは面白くないと、変更しています。このため、セグメント用データの LSB と MSB が入れ替わっています)。最後に、SS_PIN を立ち上げて、シフトしたデータを 74HC595 の出力ラッチに書き込むことで、7セグ LED に出力します。

Multi-function シールドの7セグ LED で使用する信号は以下の通りです。

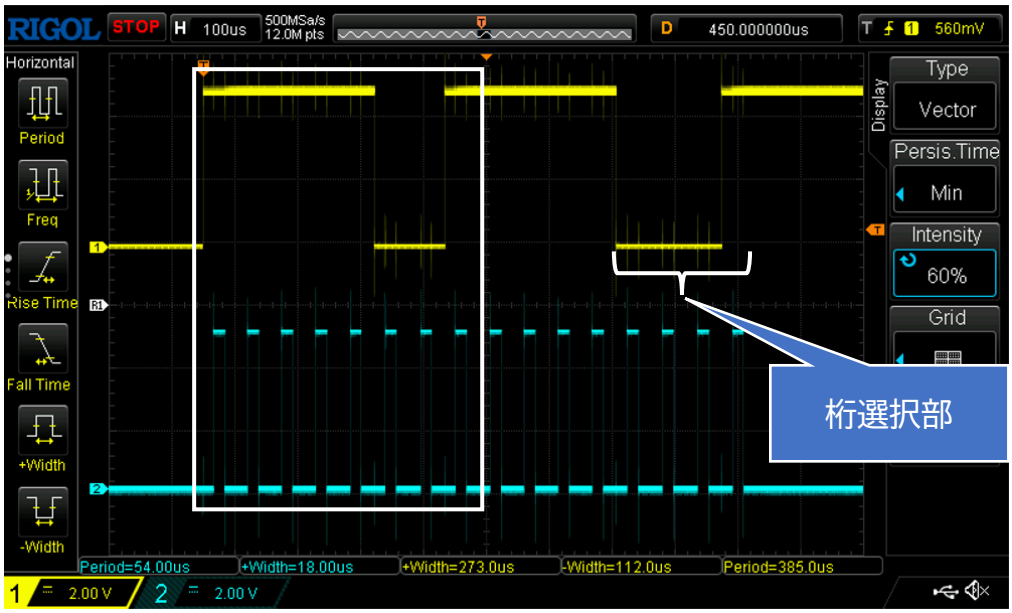
RL78/G23-64PFPB の端子	信号名	信号の意味
4	SS_PIN	HC595 のラッチ信号(立ち上がり)
7	SCK_PIN	シフト用クロック信号(立ち上がり)
8	MOSI_PIN	G23-64PFPB の出力信号

下に、送信時の MOSI_PIN 端子と SCK_PIN 端子の波形を示します。左の白い四角で囲んだ部分が、DP セグメントから始まるセグメントのデータです。セグメント A～セグメント F がロー(アクティブ)なので、表示データは"0"となります。右の四角で囲んだ部分が桁指定データで、最後から 2 ビット目が 1 なので左から 2 桁目(十秒の桁)を点灯させることが読み取れます。

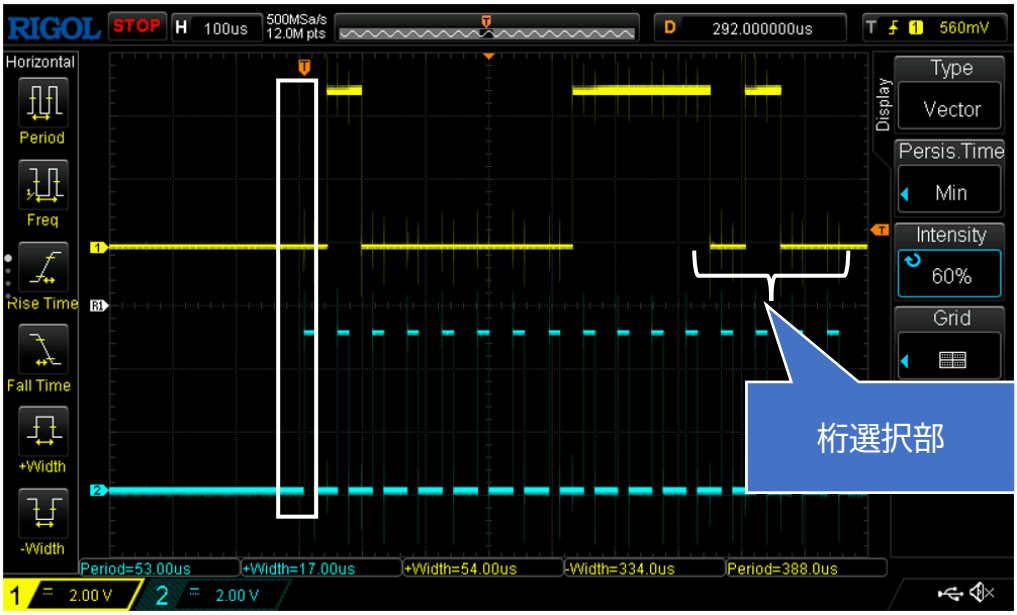


下の SCK_PIN 信号の周期が 54μs となっているので、18.5kbps の通信速度となります。転送データは 16 ビットなので、全体の時間が 864μs となり、表示データの転送のオーバーヘッドは 20%程度なので、処理時間に余裕はありそうです。

同様に、下に示す波形が問題の左端の桁（分の桁）の指定データです。セグメント B とセグメント C だけが点灯するようになっているので、分の桁の表示は”1”となります。



下に示す波形は、左から 3 桁目の秒の桁用の信号波形です。この桁の特徴は DP セグメント（ドット）が点灯していることです（白い四角で囲んだところ）。



次回は分の表示方法を変更します。

以上