

簡易時計(7セグ4桁)

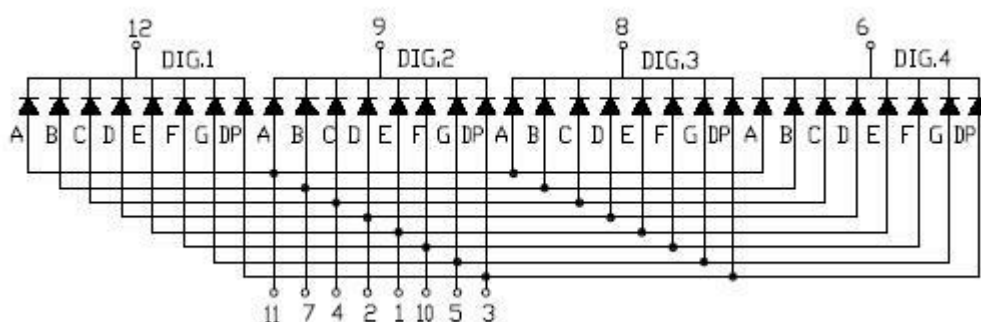
概要

4桁の7セグメントLEDが手に入りましたので、出来るだけ部品点数の少ない簡易時計を製作してみました。使用した7セグは、「ダイナミック接続4桁赤色7セグメントLED表示器(カソードコモン)OSL40562-LR」です。OSL40562-LRは、4桁表示の高輝度赤色7セグメントLEDで、ダイナミック点灯専用なので配線が少なく済みます。



<OSL40562-LRの概観>

<OSL40562-LRのピン配置>



<OSL40562-LRの特長> 素子仕様

- 色:赤(640nm)
- 輝度:30mcd(@IF=20mA)
- VF(順方向電圧):1.8~2.3V
- VR(逆方向耐圧):5V
- IFP(パルス最大電流):100mA(パルス幅:10ms、デューティ比:1/10)

サイズ

- 表示部:(高さ)14.2x(幅)8.1mm
- 全体(ピン含まず):(縦)19.0x(横)50.30x(高さ)8.0mm
- ピン間:2.54mm

内部配線

- カソード(-)コモン
- ダイナミック点灯用

動作原理

ハードウェア的には、必要最小限の部品点数を図ることを目標にしました。(簡易接続版) 通常は、標準接続版(A)(B)のようにします。

- 簡易接続版
 - トランジスタ(4個)と電流制限抵抗(8本)が不要になります。
 - 数値によって、点灯する明るさが若干異なります。(“ 1 ” が最も明るい、“ 8 ” が最も暗い)
- 標準接続版(A)(B)
 - 部品点数が多くなります。
 - 数値によって、点灯する明るさは影響を受けません。

<メイン処理>

- クロック変数(clock_msec)から、時分秒を求めます。
- セレクトスイッチ(SELECT)が、オフであれば、表示用変数(seg_dat)に、時分の値(4桁)をセットします。
- セレクトスイッチ(SELECT)が、オンであれば、表示用変数(seg_dat)に、分秒の値(4桁)をセットします。
- “ 時 ” アップスイッチ(HHUP)が、押下されると、“ 時 ” をインクリメントし、クロック変数(clock_msec)にセットします。この時 “ 秒 ” は “ 0 ” クリアされます。
- “ 分 ” アップスイッチ(MMUP)が、押下されると、“ 分 ” をインクリメントし、クロック変数(clock_msec)にセットします。この時 “ 秒 ” は “ 0 ” クリアされます。

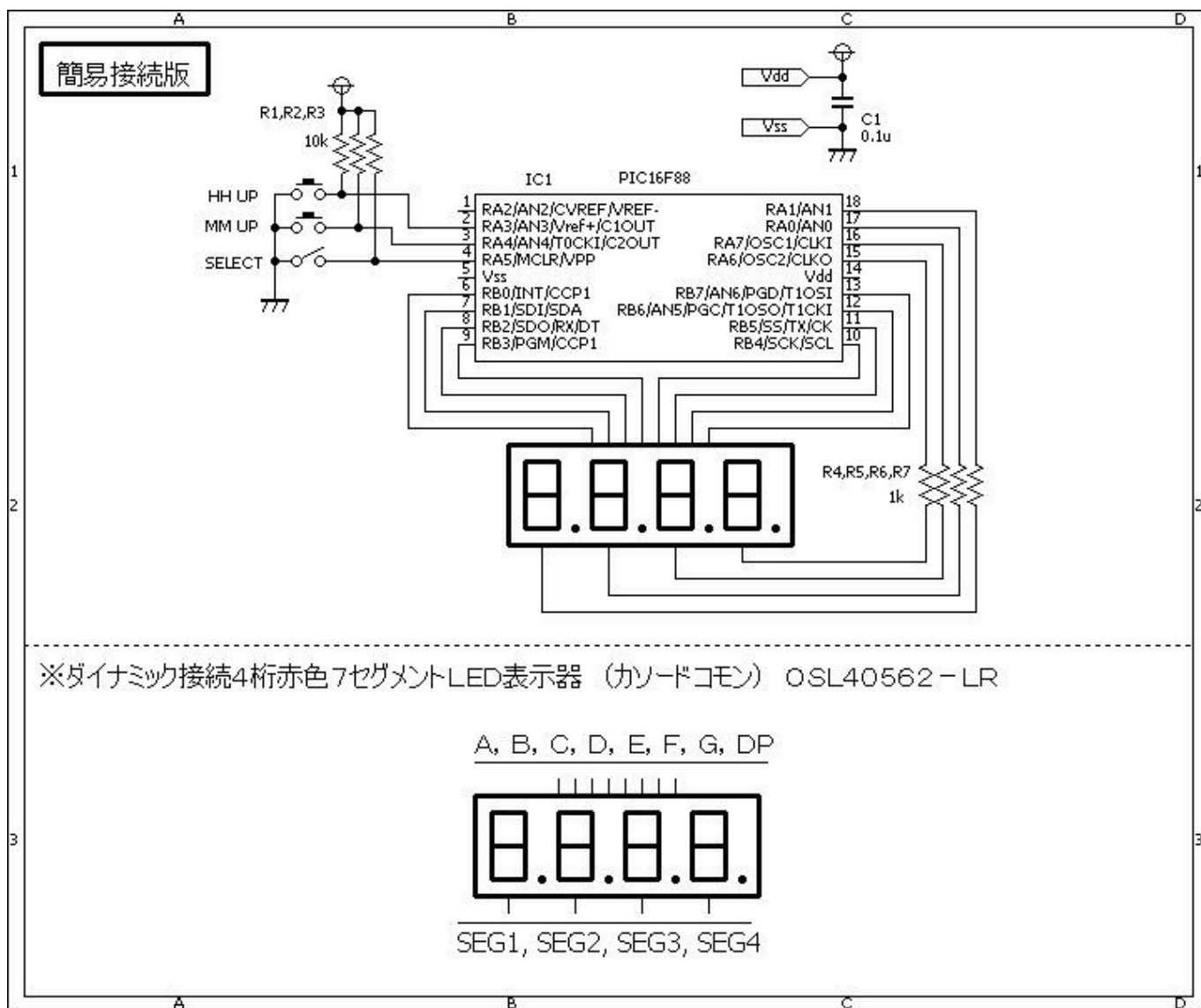
<割り込み処理>

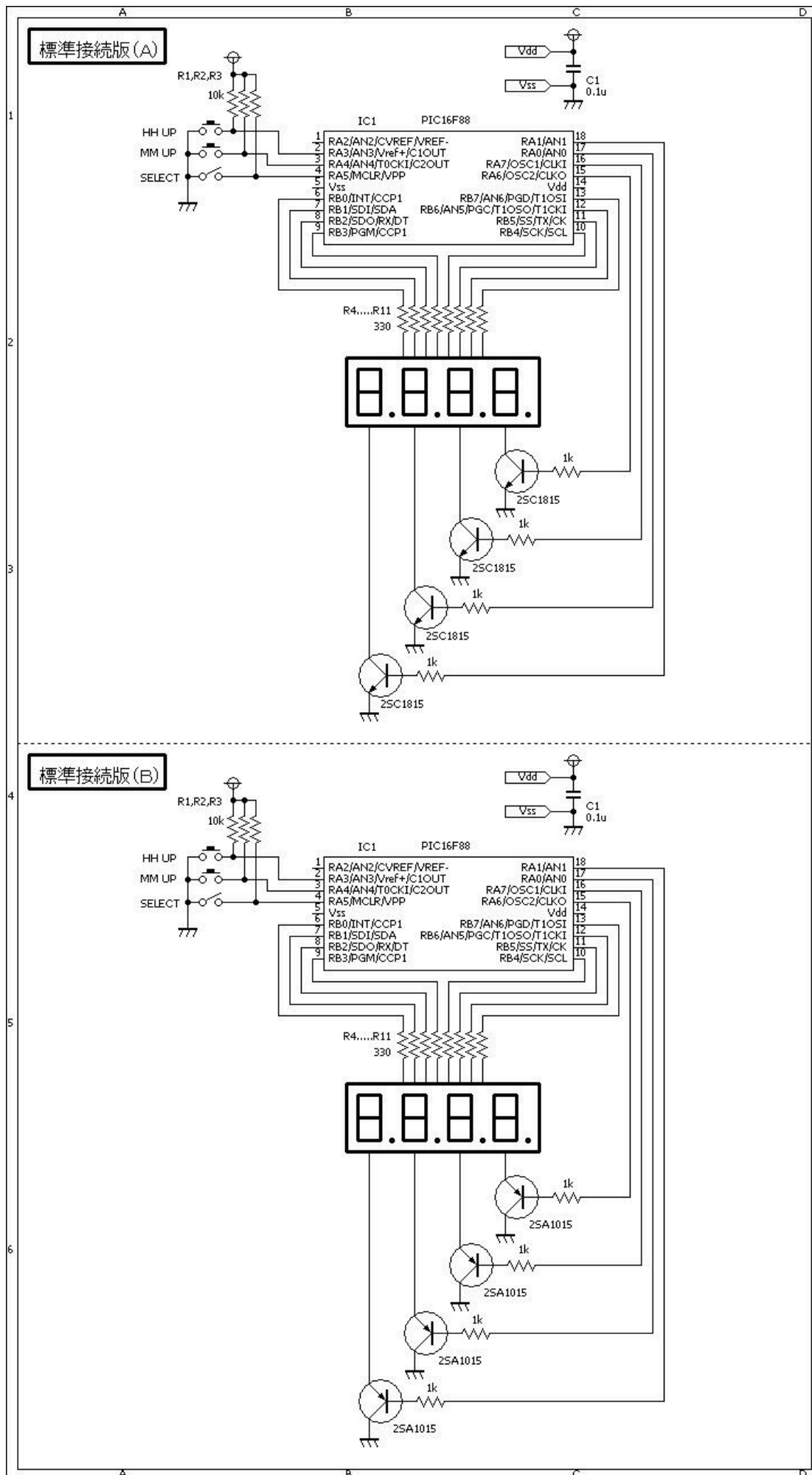
- TIMER2を使用して1msecの割り込みを発生させます。
- ダイナミック点灯処理を呼び出します。
- クロック変数(clock_msec)をインクリメントします。(0-86400000を繰り返します)

<ダイナミック点灯処理>

- 呼び出される毎に、セグメントを順次切り替えて、表示用変数(seg_dat)の値を点灯させます。

回路図





ソースコード

簡易接続および標準接続(B)のACTIVE_LOW

[clock_7seg_4digit.c](#)

```
//*****
*
//■■■関数&共有データ宣言■■■
extern void main();
extern void init_port();
extern void init_segment();
extern void init_timer();
extern void segment_disp();
extern void segment_set_data(short seg1, short seg2, short
SEG3, short seg4, short dot);
extern void interrupt();
extern long clock_msec;
extern short lock;
//*****
*
//■■■マクロ定義■■■
//7SEG-SELECT
#define ACTIVE_LOW
//簡易接続、標準接続の2SA1015
#ifdef ACTIVE_LOW
#define SEG_ON 0
#define SEG_OFF 1
#else
//標準接続の2SC1815
#define SEG_ON 1
#define SEG_OFF 0
#endif
sbit SEG1 at PORTA.B1;
sbit SEG1_Direction at TRISA.B1;
#define SEG1_ON SEG1 = SEG_ON
#define SEG1_OFF SEG1 = SEG_OFF
sbit SEG2 at PORTA.B0;
sbit SEG2_Direction at TRISA.B0;
#define SEG2_ON SEG2 = SEG_ON
#define SEG2_OFF SEG2 = SEG_OFF
sbit SEG3 at PORTA.B7;
sbit SEG3_Direction at TRISA.B7;
#define SEG3_ON SEG3 = SEG_ON
#define SEG3_OFF SEG3 = SEG_OFF
sbit SEG4 at PORTA.B6;
sbit SEG4_Direction at TRISA.B6;
#define SEG4_ON SEG4 = SEG_ON
#define SEG4_OFF SEG4 = SEG_OFF
//7SEG-DATA
```

```

#define SEG_DATA PORTB
#define SEG_DATA_Direction TRISB
//SWITCH
sbit SW_HH_UP at PORTA.B3;
sbit SW_HH_UP_Direction at TRISA.B3;
sbit SW_MM_UP at PORTA.B4;
sbit SW_MM_UP_Direction at TRISA.B4;
sbit SW_SELECT at PORTA.B5;
sbit SW_SELECT_Direction at TRISA.B5;
//other
#define INPUT_MODE 1
#define OUTPUT_MODE 0
#define LOCK 1
#define UNLOCK 0
//*****
*
//■■■メイン関数■■■
void main()
{
    short hh, mm, ss;
    long temp;
    //
    OSCCON = 0b01110000; //クロックを8MHzに設定します。
    ANSEL = 0b00000000; //A/D変換モジュールは使用しません。
    //
    init_port();
    init_segment();
    init_timer();
    // 割り込みを許可します。
    INTCON.PEIE = 1;
    INTCON.GIE = 1;
    //
    while (1) {
        temp = clock_msec / 1000;
        //
        hh = temp / 3600;
        mm = (temp % 3600) / 60;
        ss = temp % 60;
        //
        if (SW_SELECT == 1) {
            segment_set_data(hh / 10, hh % 10, mm / 10, mm
% 10, 2);
        } else {
            segment_set_data(mm / 10, mm % 10, ss / 10, ss
% 10, 2);
        }
        //
        if (SW_HH_UP == 0) {
            hh++;
            ss = 0;
            if (hh == 24) {

```

```

        hh = 0;
    }
    lock = LOCK;
    clock_msec = (((long)hh * 3600) + ((long)mm *
60) + (long)ss) * 1000;
    lock = UNLOCK;
}
if (SW_MM_UP == 0) {
    mm++;
    ss = 0;
    if (mm == 60) {
        mm = 0;
    }
    lock = LOCK;
    clock_msec = (((long)hh * 3600) + ((long)mm *
60) + (long)ss) * 1000;
    lock = UNLOCK;
}
//
Delay_ms(100);
}
}
//*****
*
// ■ ■ ■ セグメント初期化関数 ■ ■ ■
void    init_segment()
{
    SEG1_Direction = OUTPUT_MODE;
    SEG1_OFF;
    SEG2_Direction = OUTPUT_MODE;
    SEG2_OFF;
    SEG3_Direction = OUTPUT_MODE;
    SEG3_OFF;
    SEG4_Direction = OUTPUT_MODE;
    SEG4_OFF;
    //
    SEG_DATA_Direction = 0b00000000;
    SEG_DATA = 0x00;
}
//*****
*
// ■ ■ ■ 入出力ポート初期化関数 ■ ■ ■
void    init_port()
{
    SW_SELECT_Direction = INPUT_MODE;
    SW_HH_UP_Direction = INPUT_MODE;
    SW_MM_UP_Direction = INPUT_MODE;
}
//*****
*
// ■ ■ ■ タイマー初期化関数 ■ ■ ■

```

```
void    init_timer()
{
    T2CON.T2CKPS1 = 0;
    T2CON.T2CKPS0 = 0;
    T2CON.TOUTPS3 = 1;
    T2CON.TOUTPS2 = 1;
    T2CON.TOUTPS1 = 1;
    T2CON.TOUTPS0 = 1;
    TMR2 = 0;
    PIE1.TMR2IE = 1;
    PIR1.TMR2IF = 0;
    PR2 = 125;          //125=1msec/((1sec/8MHz)*4*16PS)
    T2CON.TMR2ON = 1;
}
//*****
*
// ■ ■ ■ セグメントデータ設定関数 ■ ■ ■
short  seg_dat[4] = {0, 0, 0, 0};
short  dot_point = 0;
//
void    segment_set_data(short seg1, short seg2, short SEG3, short
seg4, short dot)
{
    seg_dat[0] = seg1;
    seg_dat[1] = seg2;
    seg_dat[2] = seg3;
    seg_dat[3] = seg4;
    dot_point = dot;
}
//*****
*
// ■ ■ ■ 割り込み関数 ■ ■ ■
long    clock_msec = 0;
short   lock = UNLOCK;
void    interrupt()
{
    if (PIR1.TMR2IF == 1) {
        PIR1.TMR2IF = 0;
        //
        segment_disp();
        //
        if (lock == UNLOCK) {
            clock_msec++;
            if (clock_msec == 86400000) {
                clock_msec = 0;
            }
        }
    }
}
//*****
*
```

```
// ■ ■ ■ セグメントデータ表示関数 ■ ■ ■
short  seg_tbl[10] = {
    0b00111111,    //0
    0b00000110,    //1
    0b01011011,    //2
    0b01001111,    //3
    0b01100110,    //4
    0b01101101,    //5
    0b01111101,    //6
    0b00100111,    //7
    0b01111111,    //8
    0b01101111    //9
};

short  seg_cnt = 0;
//
void  segment_disp()
{
    switch (seg_cnt) {
    case 0:
        SEG4_OFF;
        SEG_DATA = seg_tbl[seg_dat[0]];
        if (dot_point == 1) {
            SEG_DATA.B7 = 1;
        }
        SEG1_ON;
        seg_cnt = 1;
        break;
    case 1:
        SEG1_OFF;
        SEG_DATA = seg_tbl[seg_dat[1]];
        if (dot_point == 2) {
            SEG_DATA.B7 = 1;
        }
        SEG2_ON;
        seg_cnt = 2;
        break;
    case 2:
        SEG2_OFF;
        SEG_DATA = seg_tbl[seg_dat[2]];
        if (dot_point == 3) {
            SEG_DATA.B7 = 1;
        }
        SEG3_ON;
        seg_cnt = 3;
        break;
    case 3:
        SEG3_OFF;
        SEG_DATA = seg_tbl[seg_dat[3]];
        if (dot_point == 4) {
            SEG_DATA.B7 = 1;
        }
    }
```

```

        SEG4_ON;
        seg_cnt = 0;
        break;
    }
}
//*****
*
```

標準接続(A)□ACTIVE_HIGH ※上記のソースの1行をコメントにしてください。

clock_7seg_4digit_2.c

```

//*****
*
//■■■関数&共有データ宣言■■■
extern void main();
extern void init_port();
extern void init_segment();
extern void init_timer();
extern void segment_disp();
extern void segment_set_data(short seg1, short seg2, short
SEG3, short seg4, short dot);
extern void interrupt();
extern long clock_msec;
extern short lock;
//*****
*
//■■■マクロ定義■■■
//7SEG-SELECT
//#define ACTIVE_LOW
//簡易接続、標準接続□□2SA1015□□
#ifdef ACTIVE_LOW
#define SEG_ON 0
#define SEG_OFF 1
#else
//標準接続□□2SC1815□□
#define SEG_ON 1
#define SEG_OFF 0
#endif
sbit SEG1 at PORTA.B1;
sbit SEG1_Direction at TRISA.B1;
#define SEG1_ON SEG1 = SEG_ON
#define SEG1_OFF SEG1 = SEG_OFF
sbit SEG2 at PORTA.B0;
sbit SEG2_Direction at TRISA.B0;
#define SEG2_ON SEG2 = SEG_ON
#define SEG2_OFF SEG2 = SEG_OFF
sbit SEG3 at PORTA.B7;
sbit SEG3_Direction at TRISA.B7;
#define SEG3_ON SEG3 = SEG_ON
```

```

#define      SEG3_OFF          SEG3 = SEG_OFF
sbit        SEG4              at      PORTA.B6;
sbit        SEG4_Direction    at      TRISA.B6;
#define      SEG4_ON           SEG4 = SEG_ON
#define      SEG4_OFF          SEG4 = SEG_OFF
//7SEG-DATA
#define      SEG_DATA          PORTB
#define      SEG_DATA_Direction TRISB
//SWITCH
sbit        SW_HH_UP          at      PORTA.B3;
sbit        SW_HH_UP_Direction at      TRISA.B3;
sbit        SW_MM_UP          at      PORTA.B4;
sbit        SW_MM_UP_Direction at      TRISA.B4;
sbit        SW_SELECT         at      PORTA.B5;
sbit        SW_SELECT_Direction at      TRISA.B5;
//other
#define      INPUT_MODE        1
#define      OUTPUT_MODE       0
#define      LOCK               1
#define      UNLOCK             0
//*****
*
//■■■メイン関数■■■
void      main()
{
    short    hh, mm, ss;
    long     temp;
    //
    OSCCON = 0b01110000;    //クロックを8MHzに設定します。
    ANSEL  = 0b00000000;    //A/D変換モジュールは使用しません。
    //
    init_port();
    init_segment();
    init_timer();
    // 割り込みを許可します。
    INTCON.PEIE = 1;
    INTCON.GIE = 1;
    //
    while (1) {
        temp = clock_msec / 1000;
        //
        hh = temp / 3600;
        mm = (temp % 3600) / 60;
        ss = temp % 60;
        //
        if (SW_SELECT == 1) {
            segment_set_data(hh / 10, hh % 10, mm / 10, mm
% 10, 2);
        } else {
            segment_set_data(mm / 10, mm % 10, ss / 10, ss
% 10, 2);

```

```

    }
    //
    if (SW_HH_UP == 0) {
        hh++;
        ss = 0;
        if (hh == 24) {
            hh = 0;
        }
        lock = LOCK;
        clock_msec = (((long)hh * 3600) + ((long)mm *
60) + (long)ss) * 1000;
        lock = UNLOCK;
    }
    if (SW_MM_UP == 0) {
        mm++;
        ss = 0;
        if (mm == 60) {
            mm = 0;
        }
        lock = LOCK;
        clock_msec = (((long)hh * 3600) + ((long)mm *
60) + (long)ss) * 1000;
        lock = UNLOCK;
    }
    //
    Delay_ms(100);
}

//*****
*
// ■ ■ ■ セグメント初期化関数 ■ ■ ■
void    init_segment()
{
    SEG1_Direction = OUTPUT_MODE;
    SEG1_OFF;
    SEG2_Direction = OUTPUT_MODE;
    SEG2_OFF;
    SEG3_Direction = OUTPUT_MODE;
    SEG3_OFF;
    SEG4_Direction = OUTPUT_MODE;
    SEG4_OFF;
    //
    SEG_DATA_Direction = 0b00000000;
    SEG_DATA = 0x00;
}
//*****
*
// ■ ■ ■ 入出力ポート初期化関数 ■ ■ ■
void    init_port()
{
    SW_SELECT_Direction = INPUT_MODE;

```

```
SW_HH_UP_Direction = INPUT_MODE;
SW_MM_UP_Direction = INPUT_MODE;
}
//*****
*
//■■■タイマー初期化関数■■■
void    init_timer()
{
    T2CON.T2CKPS1 = 0;
    T2CON.T2CKPS0 = 0;
    T2CON.TOUTPS3 = 1;
    T2CON.TOUTPS2 = 1;
    T2CON.TOUTPS1 = 1;
    T2CON.TOUTPS0 = 1;
    TMR2 = 0;
    PIE1.TMR2IE = 1;
    PIR1.TMR2IF = 0;
    PR2 = 125;          //125=1msec/((1sec/8MHz)*4*16PS)
    T2CON.TMR2ON = 1;
}
//*****
*
//■■■セグメントデータ設定関数■■■
short    seg_dat[4] = {0, 0, 0, 0};
short    dot_point = 0;
//
void    segment_set_data(short seg1, short seg2, short SEG3, short
seg4, short dot)
{
    seg_dat[0] = seg1;
    seg_dat[1] = seg2;
    seg_dat[2] = seg3;
    seg_dat[3] = seg4;
    dot_point = dot;
}
//*****
*
//■■■割り込み関数■■■
long    clock_msec = 0;
short    lock = UNLOCK;
void    interrupt()
{
    if (PIR1.TMR2IF == 1) {
        PIR1.TMR2IF = 0;
        //
        segment_disp();
        //
        if (lock == UNLOCK) {
            clock_msec++;
            if (clock_msec == 86400000) {
                clock_msec = 0;
            }
        }
    }
}
```

```

    }
}

}

}

//*****
*
//■■■セグメントデータ表示関数■■■
short  seg_tbl[10] = {
    0b00111111,    //0
    0b00000110,    //1
    0b01011011,    //2
    0b01001111,    //3
    0b01100110,    //4
    0b01101101,    //5
    0b01111101,    //6
    0b00100111,    //7
    0b01111111,    //8
    0b01101111    //9
};

short  seg_cnt = 0;
//
void  segment_disp()
{
    switch (seg_cnt) {
    case 0:
        SEG4_OFF;
        SEG_DATA = seg_tbl[seg_dat[0]];
        if (dot_point == 1) {
            SEG_DATA.B7 = 1;
        }
        SEG1_ON;
        seg_cnt = 1;
        break;
    case 1:
        SEG1_OFF;
        SEG_DATA = seg_tbl[seg_dat[1]];
        if (dot_point == 2) {
            SEG_DATA.B7 = 1;
        }
        SEG2_ON;
        seg_cnt = 2;
        break;
    case 2:
        SEG2_OFF;
        SEG_DATA = seg_tbl[seg_dat[2]];
        if (dot_point == 3) {
            SEG_DATA.B7 = 1;
        }
        SEG3_ON;
        seg_cnt = 3;
        break;

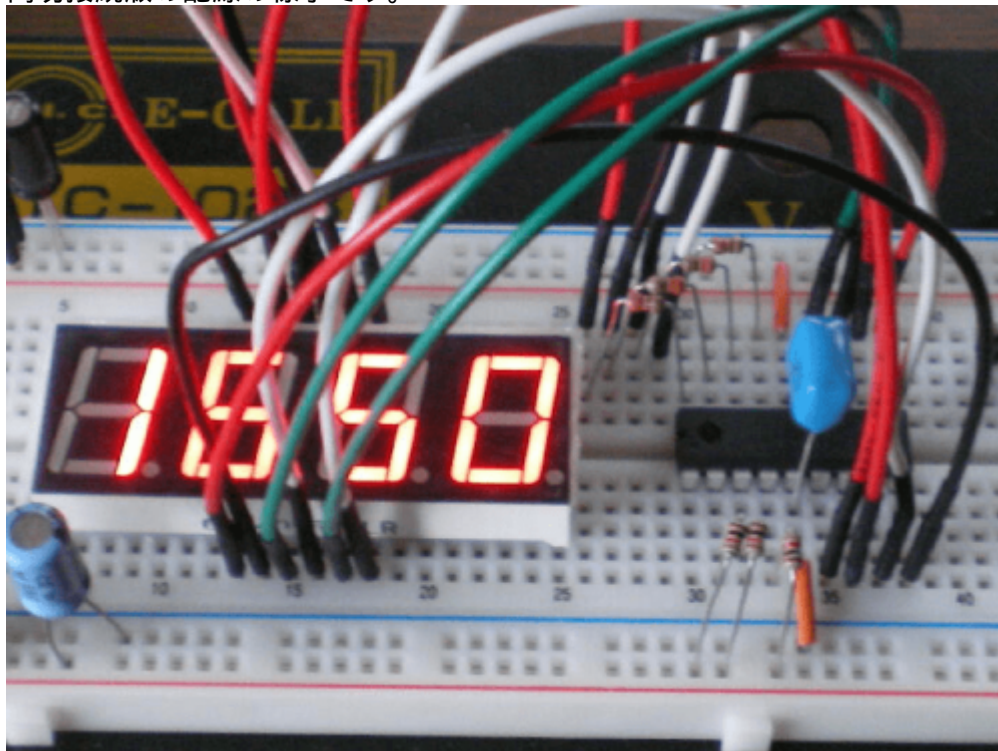
```

```
case 3:
    SEG3_OFF;
    SEG_DATA = seg_tbl[seg_dat[3]];
    if (dot_point == 4) {
        SEG_DATA.B7 = 1;
    }
    SEG4_ON;
    seg_cnt = 0;
    break;
}

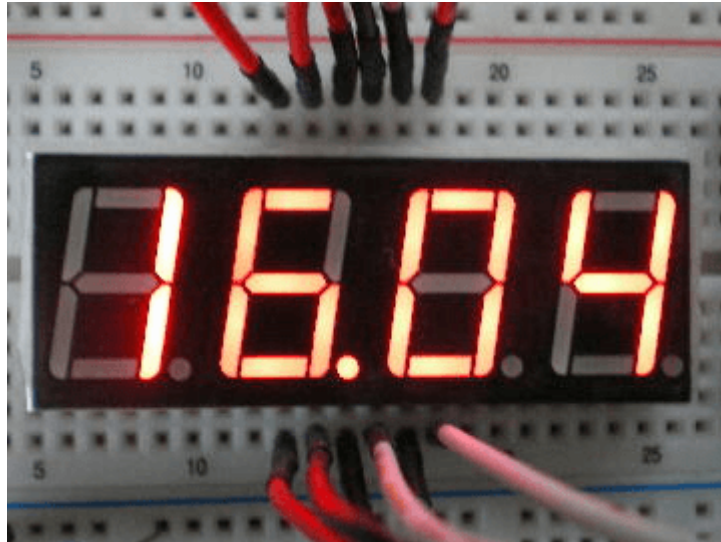
// *****
*
```

動作確認

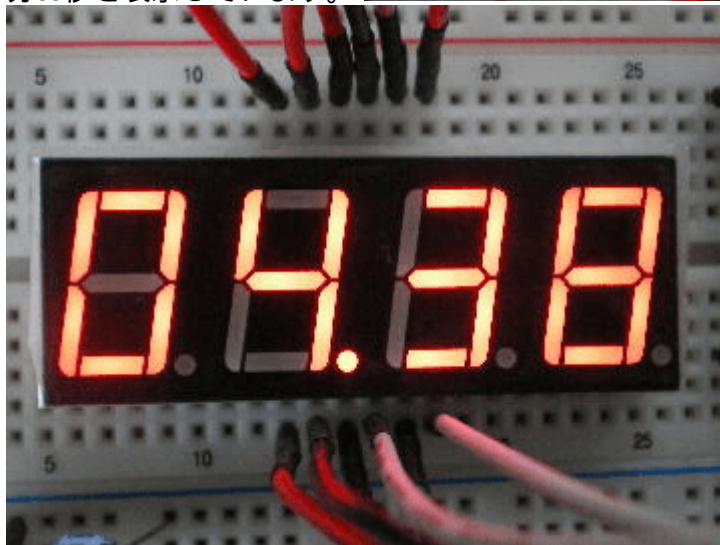
簡易接続版の配線の様子です。



左側:16時04分を表示しています。 “ 1 “ の文字が、他に比べ少し明るいのがわかりますね。 右側:04



分38秒を表示しています。



左側:20時02分を表示しています。 右側:02分20秒を表示しています。





如何でしょうか? 輝度的には、簡易接続版でも、通常利用する分には問題なさそうですね 😊

クロックにはPIC内蔵の8MHzを使用しているので、あまり高い精度は望めません。しかしOSCTUNEレジスタの値を変更して、キャリブレーションを行えば、市販されている安価な時計と同等の精度が得られるのではないのでしょうか。

著作権表示 copyright notice

このページは稲崎様の閉鎖したHPのコピーで、著作権は稲崎様にあります。詳細 This page is a copy of Mr. Inasaki's closed website, and the copyright is held by him. [Details](#)

From:

<http://deepsky.jp/wiki/> - うごくといいな

Permanent link:

<http://deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic16f88:136>

Last update: 2020/05/01 18:43

