簡易ストップウォッチ(7セグ4桁)

概要

前回製作した、簡易時計(7セグ4桁)の回路を少し変更して、簡易なストップウォッチを製作しました。

<仕様>

- カウントの精度は、次の4種類から選択できます。(表示モード)
  - 1/1000秒単位、1/100秒単位、1/10秒単位、秒単位
- 4桁の7セグメントLEDに結果を表示します。

動作原理

<メイン処理>

- スタートスイッチ(START)が押下されると、クロック変数(clock_msec)を“0“クリアし、スタートフラグ(flag)を“1”にします。
- ストップスイッチ(STOP)が押下されると、スタートフラグ(flag)を“0“にします。
- クロック変数を引数にして、カウント表示関数(count_disp)を呼び出します。

<カウント表示処理>

- 表示モードに応じた桁数に、クロック変数を変換します。
- 変換した値(カウント値)を引数にして、セグメントデータ設定関数(segment_set_data)を呼び出します。

<セグメントデータ設定処理>

- 引数で渡されたカウント値(seg1,seg2,seg3,seg4)を、表示用変数(seg_dat)にセットします。
- 引数で渡されたドット値を、ドット変数(dot_point)にセットします。

<割り込み処理>

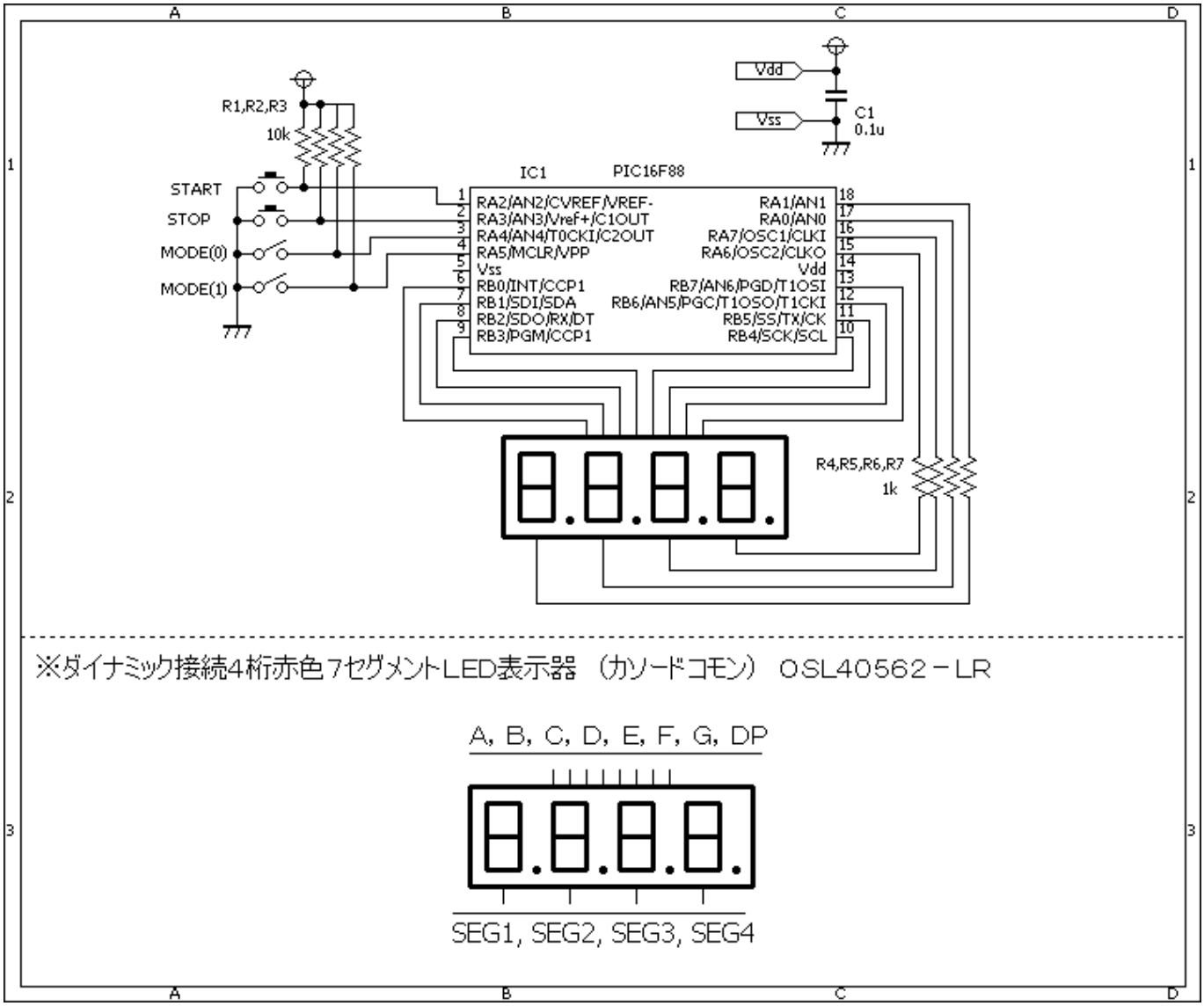
- TIMER2を使用して□1msecの割り込みを発生させます。
- ダイナミック点灯処理を呼び出します。
- スタートフラグ(flag)が“1”であれば、クロック変数(clock_msec)をインクリメントします。

<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。
- ドット変数(dot_point)に応じたセグメントのドットを点灯させます。

回路図

7セグメントLEDは、“簡易接続方式“としています。“標準接続方式”の場合には、簡易時計(7セグ4桁)を参考にしてください□ PICの各I/Oピンの最大電流は、25mAなので、これを超えない範囲で電流制限抵抗(R4,R5,R6,R7)を調整してください。(200Ω~1kΩ)



ソースコード

stopwatch_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      void        count_disp(long cnt);
extern      long        clock_msec;
extern      short       flag;
//*****************************************************************************
```

```
*
//■■■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_START            at          PORTA.B2;
sbit        SW_START_Direction  at          TRISA.B2;
sbit        SW_STOP             at          PORTA.B3;
sbit        SW_STOP_Direction   at          TRISA.B3;
sbit        SW_MODE_0           at          PORTA.B4;
sbit        SW_MODE_0_Direction at          TRISA.B4;
sbit        SW_MODE_1           at          PORTA.B5;
sbit        SW_MODE_1_Direction at          TRISA.B5;
//other
#define     INPUT_MODE          1
#define     OUTPUT_MODE         0
//*****************************************************************
*
//■■■メイン関数■■■
void    main()
{
        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。
```

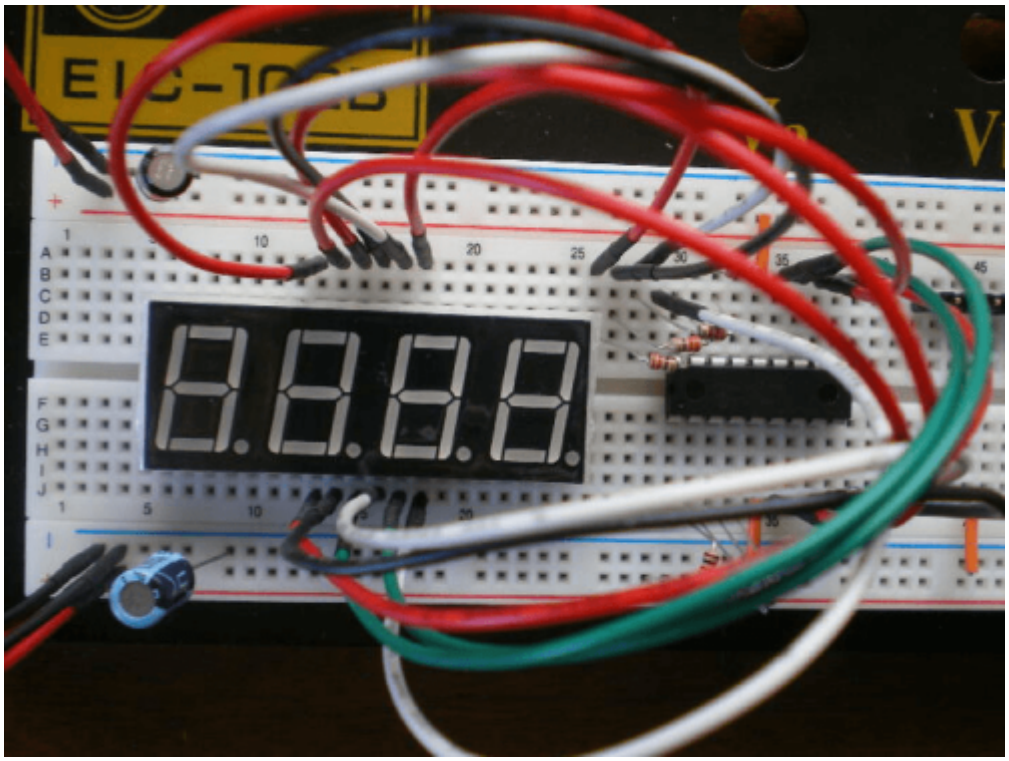
```
        //
        init_port();
        init_segment();
        init_timer();
         //割り込みを許可します。
        INTCON.PEIE = 1;
        INTCON.GIE = 1;
        //
        while (1) {
            if ((SW_START == 0) && (flag == 0)) {
                clock_msec = 0;
                flag = 1;
            }
            if ((SW_STOP == 0) && (flag == 1)) {
                flag = 0;
            }
            count_disp(clock_msec);
        }
}
//*********************************************************************
*
//■■■カウント表示関数■■■
void    count_disp(long cnt)
{
    char    buf[12];
    //
    if ((SW_MODE_1 == 1) && (SW_MODE_0 == 1)) {
            LongWordToStr(cnt, buf);
            buf[6] = (buf[6] == ' ') ? 0 : buf[6] - '0';
            buf[7] = (buf[7] == ' ') ? 0 : buf[7] - '0';
            buf[8] = (buf[8] == ' ') ? 0 : buf[8] - '0';
            buf[9] = (buf[9] == ' ') ? 0 : buf[9] - '0';
        segment_set_data(buf[6], buf[7], buf[8], buf[9], 1);
        return;
    }
    if ((SW_MODE_1 == 1) && (SW_MODE_0 == 0)) {
            LongWordToStr(cnt, buf);
            buf[5] = (buf[5] == ' ') ? 0 : buf[5] - '0';
            buf[6] = (buf[6] == ' ') ? 0 : buf[6] - '0';
            buf[7] = (buf[7] == ' ') ? 0 : buf[7] - '0';
            buf[8] = (buf[8] == ' ') ? 0 : buf[8] - '0';
        segment_set_data(buf[5], buf[6], buf[7], buf[8], 2);
        return;
    }
    if ((SW_MODE_1 == 0) && (SW_MODE_0 == 1)) {
            LongWordToStr(cnt, buf);
            buf[4] = (buf[4] == ' ') ? 0 : buf[4] - '0';
            buf[5] = (buf[5] == ' ') ? 0 : buf[5] - '0';
            buf[6] = (buf[6] == ' ') ? 0 : buf[6] - '0';
            buf[7] = (buf[7] == ' ') ? 0 : buf[7] - '0';
        segment_set_data(buf[4], buf[5], buf[6], buf[7], 3);
```

```
        return;
    }
    if ((SW_MODE_1 == 0) && (SW_MODE_0 == 0)) {
            LongWordToStr(cnt, buf);
            buf[3] = (buf[3] == ' ') ? 0 : buf[3] - '0';
            buf[4] = (buf[4] == ' ') ? 0 : buf[4] - '0';
            buf[5] = (buf[5] == ' ') ? 0 : buf[5] - '0';
            buf[6] = (buf[6] == ' ') ? 0 : buf[6] - '0';
            buf[7] = (buf[7] == ' ') ? 0 : buf[7] - '0';
        segment_set_data(buf[3], buf[4], buf[5], buf[6], 4);
        return;
    }
}
//*********************************************************************
*
//■■■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_START_Direction = INPUT_MODE;
        SW_START_Direction = INPUT_MODE;
        SW_MODE_0_Direction = INPUT_MODE;
        SW_MODE_1_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;
```

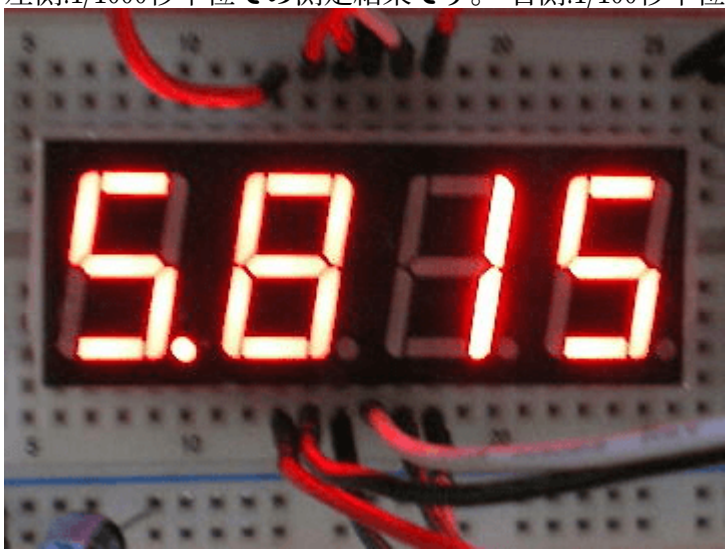
```
        TMR2 = 0;
        PIE1.TMR2IE = 1;
        PIR1.TMR2IF = 0;
        PR2 = 125;          //125=1msec/((1sec/8MHz)*4*16PS)
        T2CON.TMR2ON = 1;
}
//*********************************************************************
*
//■■■セグメントデータ設定関数■■■
short   seg_dat[4] = {0, 0, 0, 0};
short   dot_point = 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   flag = 0;
void    interrupt()
{
        if (PIR1.TMR2IF == 1) {
                PIR1.TMR2IF = 0;
                //
                segment_disp();
                //
                if (flag == 1) {
                        clock_msec++;
                }
        }
}
//*********************************************************************
*
//■■■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
```

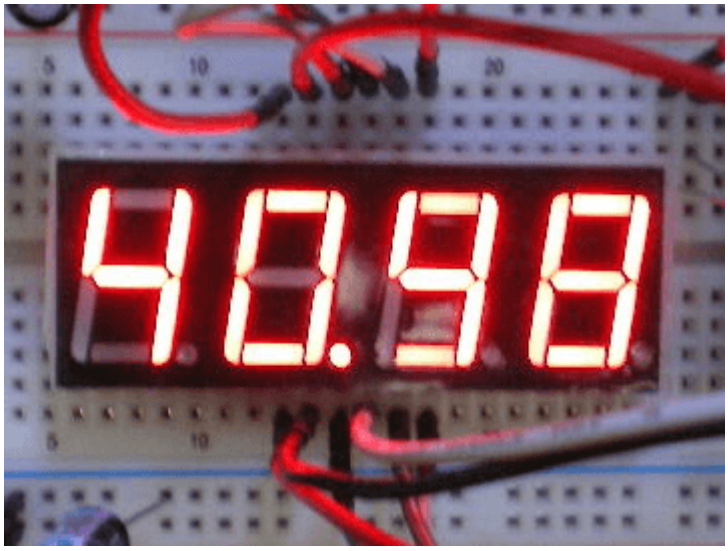
```
                0b01101111          //9
        };
short   seg_cnt = 0;
//
void    segment_disp()
{
        switch (seg_cnt) {
        case 0:
                SEG4_OFF;
                SEG_DATA = seg_tbl[seg_dat[0]];
                if (dot_point == 1) {
                        SEG_DATA.B7 = 1;
                }
                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

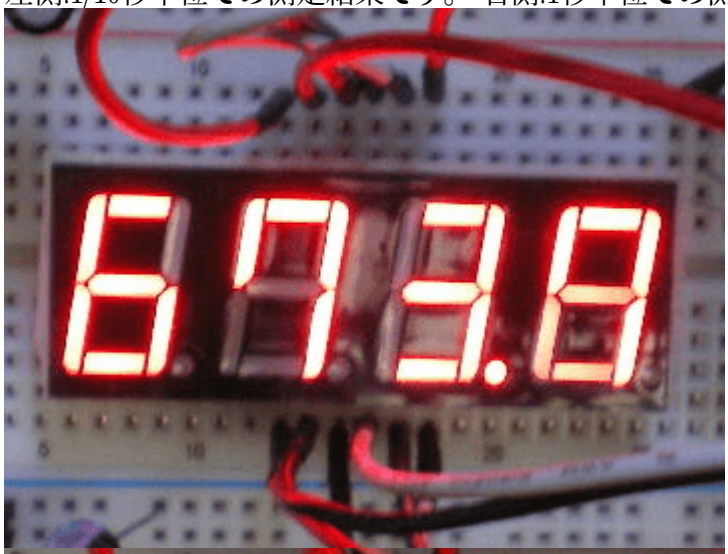


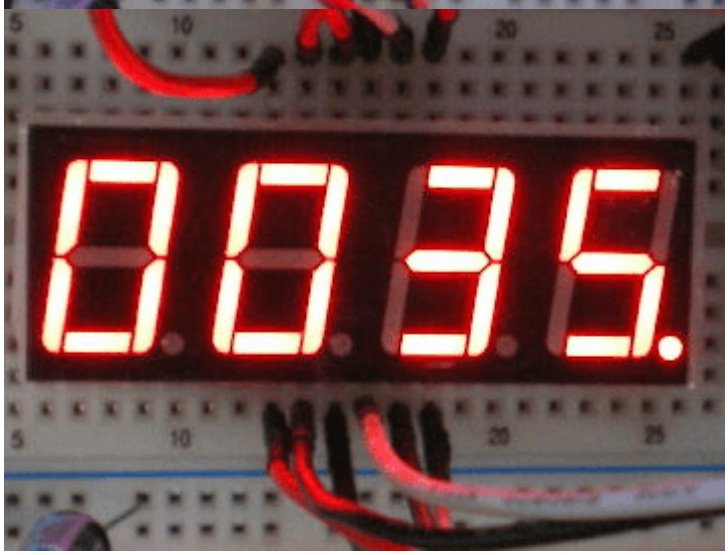
左側:1/1000秒単位での測定結果です。 右側:1/100秒単位での測定結果です。





左側:1/10秒単位での測定結果です。 右側:1秒単位での測定結果です。





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://www.deepsky.jp/wiki/ - **うごくといいな**

Permanent link:
http://www.deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic16f88:137&rev=1588325859

Last update: **2025/10/17 14:27**