簡易スクロール・クロック(PIC18F2620)

概要

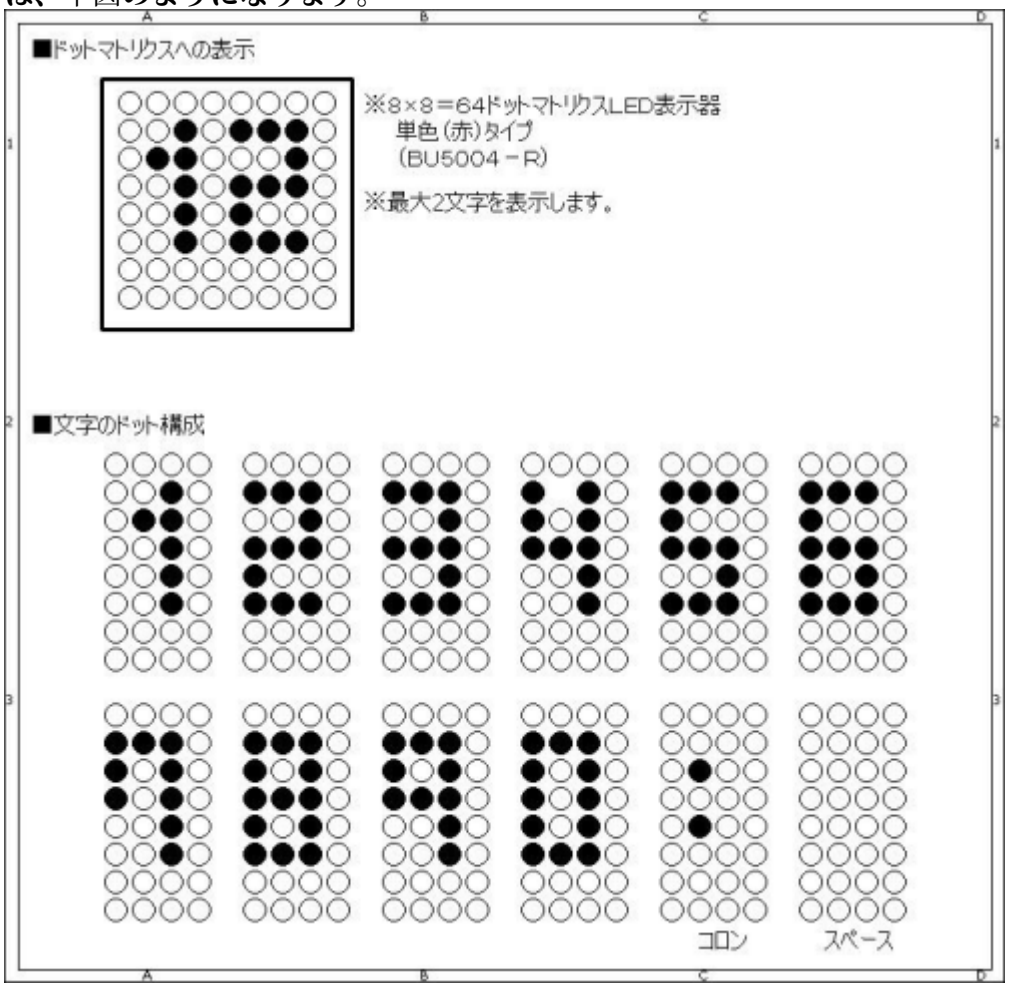
以前に8×8ドットマトリクスを使用して製作した、簡易レベルメータ(8チャネル×8ドット)を少しだけ改良して、 時刻表示をスクロール表示させる時計を製作しました。

<仕様>

- 12時間時計とします。
- 時分秒をスクロール表示させます。
- 一度に表示可能な数字は、最大2文字とします。
- スクロール表示のポーズ(一時停止)を可能とします。
- 時分の調整を可能とします。(プッシュスイッチでインクリメント方向のみ可能、秒は“0“クリアされます)
- クロックにはPIC内蔵の8MHzを使用し、小型化を図ります。

動作原理(ハードウェア)

◎時刻の表示 8×8のドットマトリクスに時刻(時分秒)をスクロール表示させます。 1文字のドット構成は、下図のようになります。



動作原理(ソフトウェア)

◎メイン処理

- クロック変数(clock_msec)から、時分秒を求めます。
- “ 時 ” アップスイッチ(SW_HH_UP)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(SW_MM_UP)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。
- ポーズスイッチ(SW_PAUSE)が、押下されるとスクロールを一時停止します。
- スクロール処理を呼び出します。

◎割り込み処理

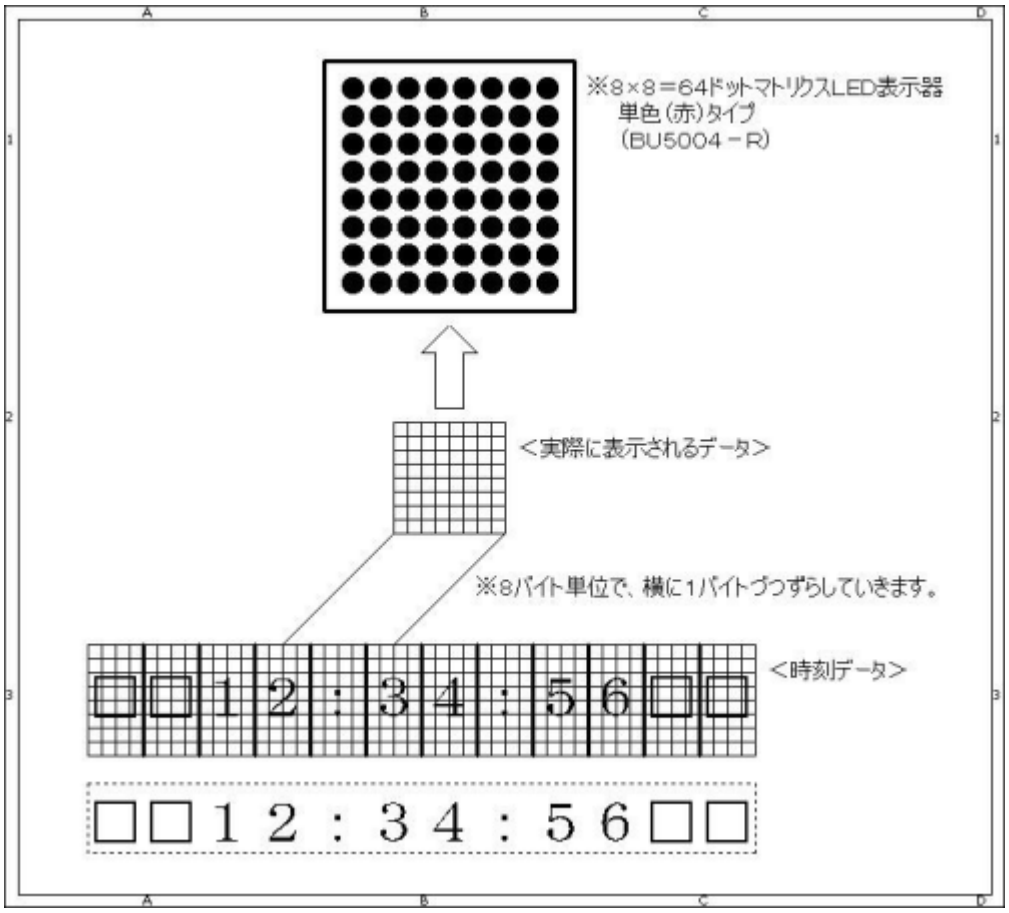
- TIMER2を使用して□1msecの割り込みを発生させます。
- ダイナミック点灯処理を呼び出します。
- クロック変数(clock_msec)をインクリメントします。(0~43200000を繰り返します)

◎ダイナミック点灯処理

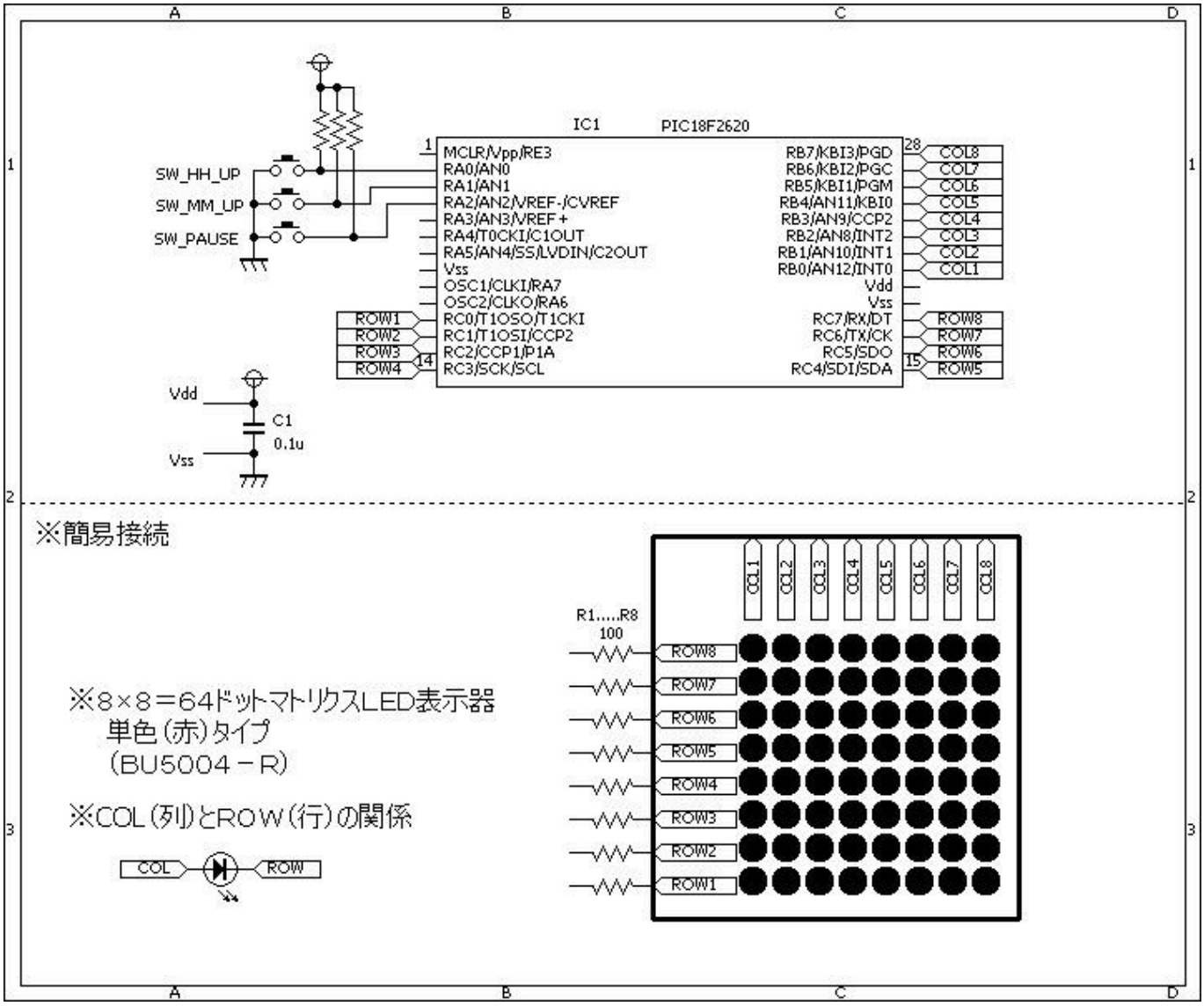
- 呼び出される毎に、ROW(行)を順次切り替えて、表示データ(view_data)の値を点灯させます。

◎スクロール処理

- 12文字(48バイト)で構成される、時刻データ(clock_data)を、8バイト単位で、横に1バイトづつずらしながら、表示データにセットします。



回路図



ソースコード

clock_led_8x8.c

```
//*****************************************************************************
*
/*
    <簡易スクロール・クロック>
*/
//*****************************************************************************
*
extern  void    main();
extern  void    init_timer();
extern  void    interrupt();
extern  void    data_set(short p, char *dt);
extern  void    clock_data_set(short hh, short mm, short ss);
```

```
extern  void    data_scroll();
extern  long    clock_msec;
extern  short   lock;
//*****************************************************************************
*
#define  LOCK      1
#define  UNLOCK    0
#define  SW_HH_UP  PORTA.B0
#define  SW_MM_UP  PORTA.B1
#define  SW_PAUSE  PORTA.B2
//*****************************************************************************
*
char    view_data[8];
char    clock_data[48];
char    data_0[] = {
        0b01111100,
        0b01000100,
        0b01111100,
        0b00000000
};
char    data_1[] = {
        0b00100000,
        0b01111100,
        0b00000000,
        0b00000000
};
char    data_2[] = {
        0b01011100,
        0b01010100,
        0b01110100,
        0b00000000
};
char    data_3[] = {
        0b01010100,
        0b01010100,
        0b01111100,
        0b00000000
};
char    data_4[] = {
        0b01110000,
        0b00010000,
        0b01111100,
        0b00000000
};
char    data_5[] = {
        0b01110100,
        0b01010100,
        0b01011100,
        0b00000000
};
```

```
char    data_6[] = {
        0b01111100,
        0b01010100,
        0b01011100,
        0b00000000
};
char    data_7[] = {
        0b01110000,
        0b01000000,
        0b01111100,
        0b00000000
};
char    data_8[] = {
        0b01111100,
        0b01010100,
        0b01111100,
        0b00000000
};
char    data_9[] = {
        0b01110100,
        0b01010100,
        0b01111100,
        0b00000000
};
char    data_space[] = {
        0b00000000,
        0b00000000,
        0b00000000,
        0b00000000
};
char    data_colon[] = {
        0b00000000,
        0b00101000,
        0b00000000,
        0b00000000
};
char    *data_num_table[] = {
        data_0,
        data_1,
        data_2,
        data_3,
        data_4,
        data_5,
        data_6,
        data_7,
        data_8,
        data_9
};
//*****************************************************************************
*
//■■■データセット関数■■■
```

```
void    data_set(short p, char *dt)
{
        memcpy(&clock_data[p * 4], dt, 4);
}
//*****************************************************************************
*
//■■■時刻データセット関数■■■
void    clock_data_set(short hh, short mm, short ss)
{
        data_set(0, data_space);
        data_set(1, data_space);
        data_set(2, data_num_table[Dec2Bcd(hh) >> 4]);
        data_set(3, data_num_table[Dec2Bcd(hh) & 0x0F]);
        data_set(4, data_colon);
        data_set(5, data_num_table[Dec2Bcd(mm) >> 4]);
        data_set(6, data_num_table[Dec2Bcd(mm) & 0x0F]);
        data_set(7, data_colon);
        data_set(8, data_num_table[Dec2Bcd(ss) >> 4]);
        data_set(9, data_num_table[Dec2Bcd(ss) & 0x0F]);
        data_set(10, data_space);
        data_set(11, data_space);
}
//*****************************************************************************
*
//■■■データスクロール関数■■■
short   scroll_pnt = 0;
void    data_scroll()
{
        memcpy(view_data, &clock_data[scroll_pnt], 8);
        scroll_pnt++;
        if (scroll_pnt == 41) {
                scroll_pnt = 0;
        }
}
//*****************************************************************************
*
//■■■メイン関数■■■
void    main()
{
        short   cnt, hh, mm, ss;
        long    tmp;
        //クロックを8MHzに設定します。
        OSCCON.IRCF2 = 1;
        OSCCON.IRCF1 = 1;
        OSCCON.IRCF0 = 1;
        //
        TRISA = 0b11111111;
        TRISB = 0b00000000;
        TRISC = 0b00000000;
        //□□□変換を使用しません。
        ADCON1.PCFG3 = 1;
```

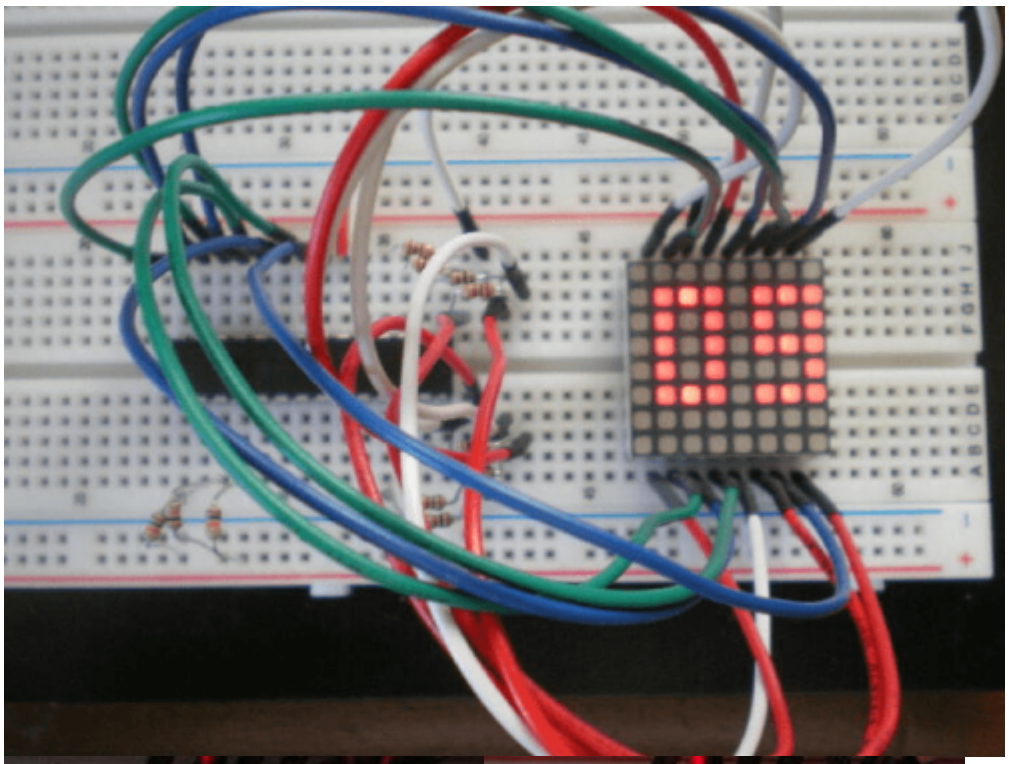
```
        ADCON1.PCFG2 = 1;
        ADCON1.PCFG1 = 1;
        ADCON1.PCFG0 = 1;
        //□□□□□を設定□□□□□□周期）します。
        init_timer();
        //割り込みを許可します。
        INTCON.PEIE = 1;
        INTCON.GIE = 1;
        //
        while (1) {
                tmp = clock_msec / 1000;
                hh = tmp / 3600;
                mm = (tmp % 3600) / 60;
                ss = tmp % 60;
                //
                if (SW_HH_UP == 0) {
                        hh++;
                        ss = 0;
                        if (hh == 12) {
                                hh = 0;
                        }
                        lock = LOCK;
                        clock_msec = (((long)hh * 3600) + ((long)mm *
60) + (long)ss) * 1000;
                        lock = UNLOCK;
                        scroll_pnt = 8;
                }
                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;
                        scroll_pnt = 20;
                }
                //
                clock_data_set(hh, mm, ss);
                if (SW_PAUSE == 1) {
                    data_scroll();
            }
                //
                Delay_ms(200);
        }
}
//*****************************************************************************
*
//■■■タイマー初期化関数■■■
```

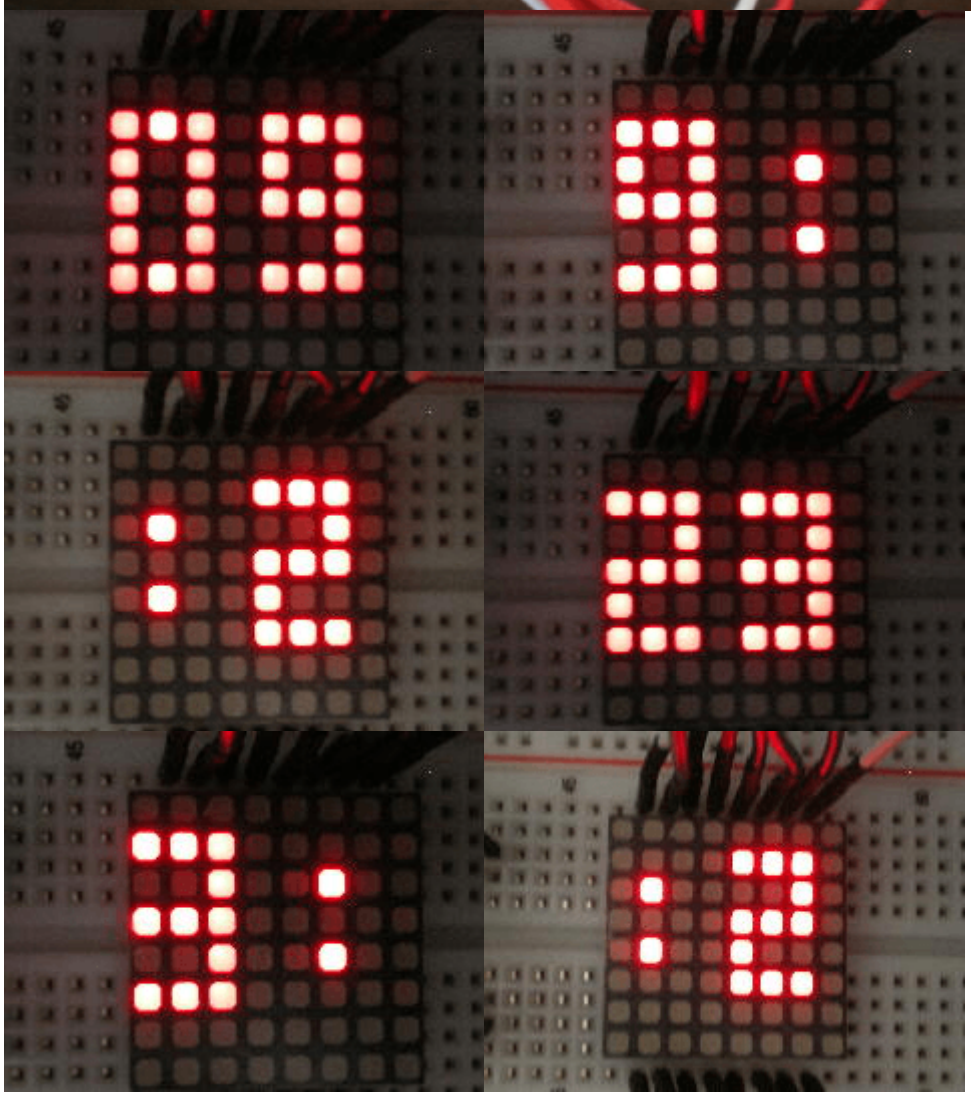
```
void    init_timer()
{
        T2CON.T2CKPS1 = 0;
        T2CON.T2CKPS0 = 0;
        T2CON.T2OUTPS3 = 1;
        T2CON.T2OUTPS2 = 1;
        T2CON.T2OUTPS1 = 1;
        T2CON.T2OUTPS0 = 1;
        TMR2 = 0;
        PIE1.TMR2IE = 1;
        PIR1.TMR2IF = 0;
        PR2 = 125;          //125=1msec/((1sec/8MHz)*4*16PS)
        T2CON.TMR2ON = 1;
}
//*****************************************************************************
*
//■■■割り込み関数■■■
short   bar = 0;
long    clock_msec = 0;
short   lock = UNLOCK;
void    interrupt()
{
        if (PIR1.TMR2IF == 1) {
                PIR1.TMR2IF = 0;
                //
                switch (bar) {
                case 0:
                        PORTB = view_data[0];
                        PORTC = 0b11111110;
                        bar++;
                        break;
                case 1:
                        PORTB = view_data[1];
                        PORTC = 0b11111101;
                        bar++;
                        break;
                case 2:
                        PORTB = view_data[2];
                        PORTC = 0b11111011;
                        bar++;
                        break;
                case 3:
                        PORTB = view_data[3];
                        PORTC = 0b11110111;
                        bar++;
                        break;
                case 4:
                        PORTB = view_data[4];
                        PORTC = 0b11101111;
                        bar++;
                        break;
```

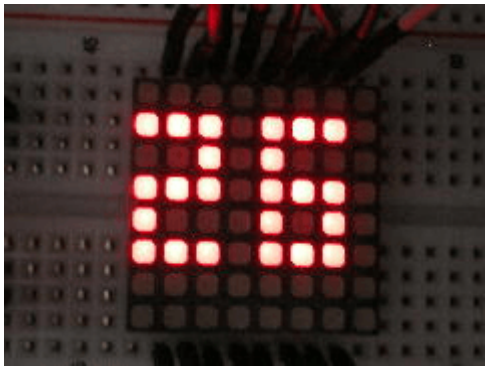
```
                case 5:
                        PORTB = view_data[5];
                        PORTC = 0b11011111;
                        bar++;
                        break;
                case 6:
                        PORTB = view_data[6];
                        PORTC = 0b10111111;
                        bar++;
                        break;
                case 7:
                        PORTB = view_data[7];
                        PORTC = 0b01111111;
                        bar = 0;
                        break;
                }
                //
                if (lock == UNLOCK) {
                        clock_msec++;
                        if (clock_msec == 43200000) {
                                clock_msec = 0;
                        }
                }
        }
}
//*****************************************************************************
*
```

動作確認

9時23分26秒をスクロール表示しているところです。







如何でしょうか?

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でしょうか。