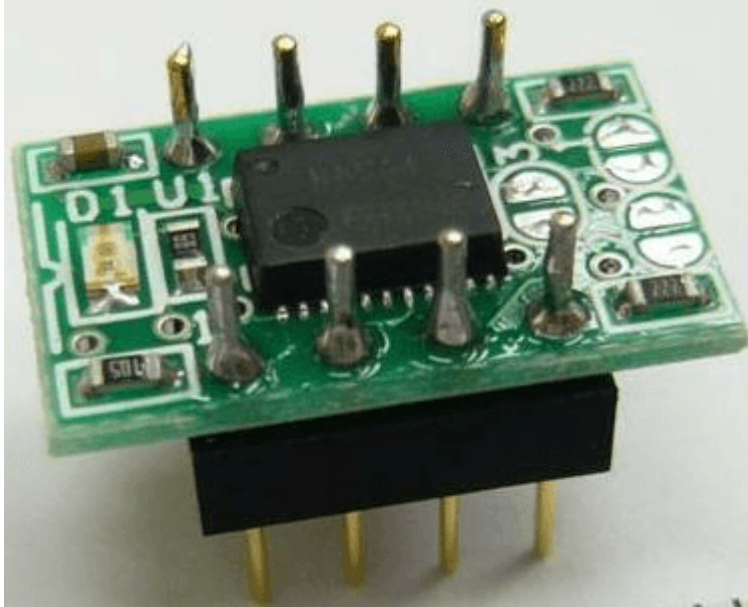


簡易RTC(Real Time Clock)(I2C対応)

概要

通常は、下図のようなI2Cインターフェース(2線式)に対応した、リアルタイムクロックモジュールを使用して、時計機能などを構築します。

今回は、このRTC機能そのものをPICで実現してみました。



動作原理

I2Cのスレーブ機能としての、基本的な構造は、以前に製作したLCDモニタ(I2C対応)を参照してください。時刻の、読み出し/書き込みは、以前に製作した、簡易時計(PIC単体)を参照してください。

<メモリ構造> RTCからの時刻データの読み出し、RTCへの時刻データの書き込みのデータを設定するために、7バイトのメモリを使用します
0x00 時データ(読み出し用) 0x01 分データ(読み出し用) 0x02 秒データ(読み出し用) 0x03 時データ(書き込み用) 0x04 分データ(書き込み用) 0x05 秒データ(書き込み用) 0x06 書き込み用のデータを、RTCの基準クロックに設定するための指示用(1:設定有り、0:設定無し)

<処理の流れ>

1. 0.1秒毎に、クロック変数をインクリメント(+1)する。
2. クロック変数を換算し、時分秒を、読み出し用のメモリ0x00~0x02に、設定する。
3. RTCへの時刻データの設定が指示されると、メモリ0x03~0x05を換算し、クロック変数に設定する。
4. 1.へ戻る。

<マスター側から、本ユニットの時刻データを、読み込む方法および書き込む方法>

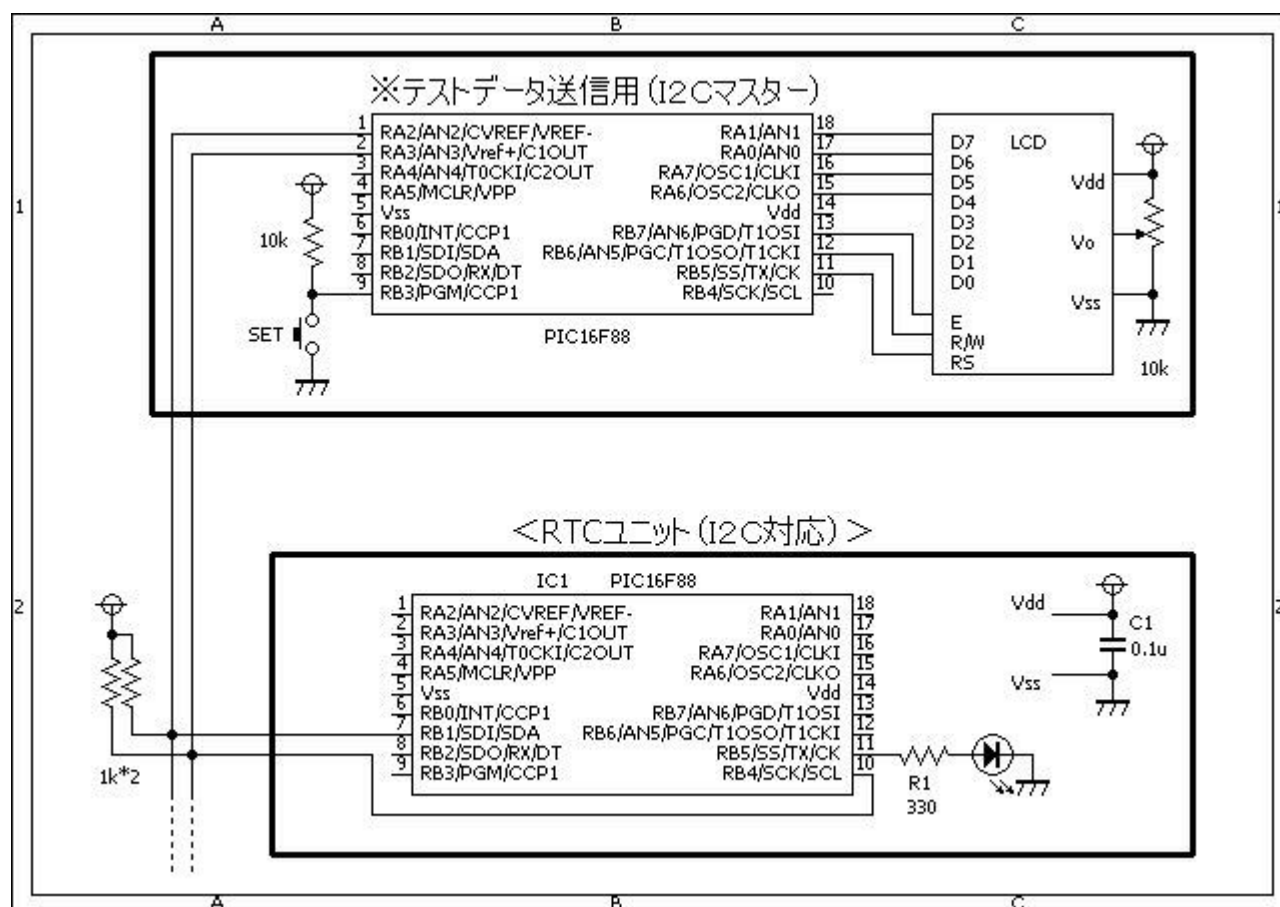
1. 時刻データを読み込む

```
Soft_I2C_Start();  
Soft_I2C_Write(0x80);  
Soft_I2C_Write(0x00);  
Soft_I2C_Start();  
Soft_I2C_Write(0x81);  
hh = Soft_I2C_Read(ACK);  
mm = Soft_I2C_Read(ACK);  
ss = Soft_I2C_Read(NO_ACK);  
Soft_I2C_Stop();
```

2. 時刻データを書き込む

```
Soft_I2C_Start();  
Soft_I2C_Write(0x80);  
Soft_I2C_Write(0x03);  
Soft_I2C_Write(12); // 12時  
Soft_I2C_Write(34); // 34分  
Soft_I2C_Write(56); // 56秒  
Soft_I2C_Write(0x01); // 設定指示  
Soft_I2C_Stop();
```

回路図



ソースコード

rtc_i2c.c

```
//*****
*
/*
    □□□□□□□□対応) >
*/
//*****
*

#define        ADDR_BASE        0x80

#define        ON                1
#define        OFF              0

#define        MEMORY_SIZE      7

#define        LED               PORTB.F5

//*****
*

void    i2c_Write(unsigned short dat)
{
    while (SSPSTAT.BF == 1)
        ;
    while (1) {
        SSPCON.WCOL = 0;
        SSPBUF = dat;
        if (SSPCON.WCOL == 1)
            continue;
        //
        SSPCON.CKP = 1;
        return;
    }
}

//*****
*

static unsigned    short    i2c_memory[MEMORY_SIZE], i2c_pnt, i2c_flg,
i2c_tmp, i2c_start_flg, i2c_stop_flg;
/*
i2c_memory[0]      時
i2c_memory[1]      分
i2c_memory[2]      秒
*/
//*****
```

```
*  
  
void i2c_Handler()  
{  
    if (PIR1.SSPIF != 1)  
        return;  
    //  
    PIR1.SSPIF = 0;  
    //  
    LED = ON;  
    //  
    if (SSPSTAT.S == 1) { //スタートビットを検出。  
        i2c_start_flg = 1;  
    }  
    if (SSPSTAT.P == 1) { //ストップビットを検出。  
        i2c_stop_flg = 1;  
    }  
    //  
    i2c_tmp = SSPSTAT & 0b00101101;  
    //  
    if (i2c_tmp == 0b00001001) { //書き込みモード、デバイスアドレス  
        i2c_tmp = SSPBUF;  
        i2c_flg = 0;  
        i2c_pnt = 0;  
        LED = OFF;  
        return;  
    }  
    if (i2c_tmp == 0b00101001) { //書き込みモード、データ  
        if (i2c_flg == 0) {  
            i2c_pnt = SSPBUF;  
            i2c_flg = 1;  
            LED = OFF;  
            return;  
        }  
        if (i2c_flg == 1) {  
            if (i2c_pnt < MEMORY_SIZE) {  
                i2c_memory[i2c_pnt] = SSPBUF;  
                i2c_pnt++;  
            }  
            LED = OFF;  
            return;  
        }  
    }  
    if (i2c_tmp == 0b00001100) { //読み込みモード、デバイスアドレス  
        i2c_Write(i2c_memory[i2c_pnt]);  
        i2c_pnt++;  
        LED = OFF;  
        return;  
    }  
    if (i2c_tmp == 0b00101100) { //読み込みモード、データ□□□□□
```

```
        i2c_Write(i2c_memory[i2c_pnt]);
        i2c_pnt++;
        LED = OFF;
        return;
    }
    if (i2c_tmp == 0b00101000) {    //読み込みモード、データ□□□□□□□□
        i2c_tmp = SSPBUF;
        SSPCON = 0b00111110;
        LED = OFF;
        return;
    }
    LED = OFF;
}

//*****
*

static long    clock;

void    rtc_Handler()
{
    static    short    hh, mm, ss;
    //
    if (PIR1.CCP1IF != 1)
        return;
    //
    PIR1.CCP1IF = 0;
    //
    clock++;
    if (clock == 86400)
        clock = 0;
    //
    hh = clock / 3600;
    mm = (clock - (3600 * (long)hh)) / 60;
    ss = (clock - (3600 * (long)hh) - (60 * (long)mm)) / 10;
    i2c_memory[0] = hh;
    i2c_memory[1] = mm;
    i2c_memory[2] = ss;
}

//*****
*

void    interrupt()
{
    i2c_Handler();
}

//*****
*
```

```
void    init_rtc()
{
    // TIMER1の設定
    PIE1.TMR1IE = 0;
    PIR1.TMR1IF = 0;
    T1CON.T1CKPS0 = 1;
    T1CON.T1CKPS1 = 1;
    T1CON.TMR1ON = 0;
    TMR1L = 0;
    TMR1H = 0;
    // CCPの設定
    PIE1.CCP1IE = 0;
    PIR1.CCP1IF = 0;
    CCP1CON.CCP1M3 = 1;
    CCP1CON.CCP1M2 = 0;
    CCP1CON.CCP1M1 = 1;
    CCP1CON.CCP1M0 = 1;
    CCPR1L = 0xA8;    // 0.1sec...10hz...クロックが8Mhzの時
    CCPR1H = 0x61;    // 0.1sec...(1÷8000000)*4*8*25000
    //
    clock = 0;
    //
    T1CON.TMR1ON = 1;
}

//*****
*

void    main()
{
    unsigned    short    cnt;
    //
    CMCON    = 0b00000111;    //コンパレータは使用しない。
    ANSEL    = 0b00000000;    //A/Dコンバータは使用しない。
    OSCCON    = 0b01110000;    //クロックは内臓8MHzを使用する。
    TRISA    = 0b00111100;    //PORTAを設定する。
    TRISB    = 0b00011111;    //PORTBを設定する。
    OPTION_REG.NOT_RBPU = 0;    //PORTBをブルアップする。
    //□□□を設定する。
    SSPSTAT.SMP = 1;
    SSPSTAT.CKE = 1;
    SSPCON.WCOL = 0;
    SSPCON.SSP0V = 0;
    SSPCON.SSPEN = 1;
    SSPCON.CKP = 1;
    SSPCON.SSPM0 = 0;
    SSPCON.SSPM1 = 1;
    SSPCON.SSPM2 = 1;
    SSPCON.SSPM3 = 1;
    SSPADD = ADDR_BASE;
```

```

    PIE1.SSPIE = 1;
    PIR1.SSPIF = 0;
    //
    LED = OFF;
    i2c_pnt = 0;
    i2c_flg = 0;
    i2c_start_flg = 0;
    i2c_stop_flg = 0;
    for (cnt = 0; cnt < MEMORY_SIZE; cnt++) {
        i2c_memory[cnt] = 0x00;
    }
    //
    for (cnt = 0; cnt < 10; cnt++) {
        LED = ON;
        Delay_ms(50);
        LED = OFF;
        Delay_ms(50);
    }
    // 割り込み(全体)の設定
    INTCON.PEIE = 1;
    INTCON.GIE = 1;
    //
    init_rtc();
    while (1) {
        rtc_Handler();
        if (i2c_memory[6] == 0x01) {
            i2c_memory[6] = 0x00;
            clock = ((long)i2c_memory[3] * 36000) +
                ((long)i2c_memory[4] * 600) + ((long)i2c_memory[5] * 10);
        }
    }
}

//*****
*
```

<参考> テストデータ送信用(I2Cマスター/PIC16F88)のプログラムです。

rtc_i2c_master.c

```

//*****
*
/*
    □□□□のテストデータ送信用(マスター) >
*/
//*****
*

#define ACK 1
#define NO_ACK 0
```

```
//*****
*

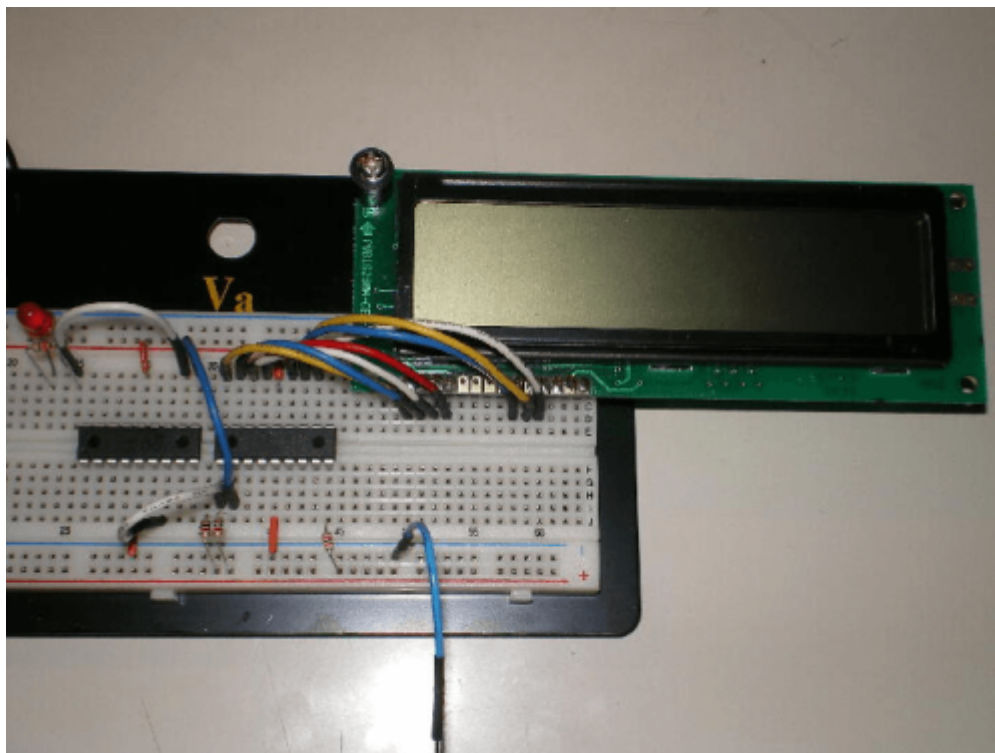
void main()
{
    static unsigned short cnt, hh, mm, ss;
    static char buf[8];
    //
    OSCCON = 0b01110000; // クロックを8Mhzに設定する。
    ANSEL = 0b00000000; // □□変換は使用しない。
    TRISA = 0b00110000;
    TRISB = 0b00001111;
    //
    Lcd_Custom_Config(&PORTA, 1, 0, 7, 6, &PORTB, 5, 6, 7);
    Lcd_Custom_Cmd(LCD_CURSOR_OFF);
    Lcd_Custom_Cmd(LCD_CLEAR);
    //
    for (cnt = 0; cnt < 16; cnt++) {
        Lcd_Custom_Chr(1, cnt + 1, 0xFF);
        Delay_ms(50);
    }
    for (cnt = 0; cnt < 16; cnt++) {
        Lcd_Custom_Chr(2, cnt + 1, 0xFF);
        Delay_ms(50);
    }
    Lcd_Custom_Cmd(LCD_CLEAR);
    //
    //
    Soft_I2C_Config(&PORTA, 2, 3); // SDA, SCL
    //
    while (1) {
        //
        Soft_I2C_Start();
        Soft_I2C_Write(0x80);
        Soft_I2C_Write(0x00);
        Soft_I2C_Start();
        Soft_I2C_Write(0x81);
        hh = Soft_I2C_Read(ACK);
        mm = Soft_I2C_Read(ACK);
        ss = Soft_I2C_Read(NO_ACK);
        Soft_I2C_Stop();
        //
        ByteToStr(hh, buf);
        buf[1] = buf[1] == ' ' ? '0' : buf[1];
        Lcd_Custom_Out(1, 1, &buf[1]);
        Lcd_Custom_Chr(1, 3, ':');
        ByteToStr(mm, buf);
        buf[1] = buf[1] == ' ' ? '0' : buf[1];
        Lcd_Custom_Out(1, 4, &buf[1]);
    }
}
```

```

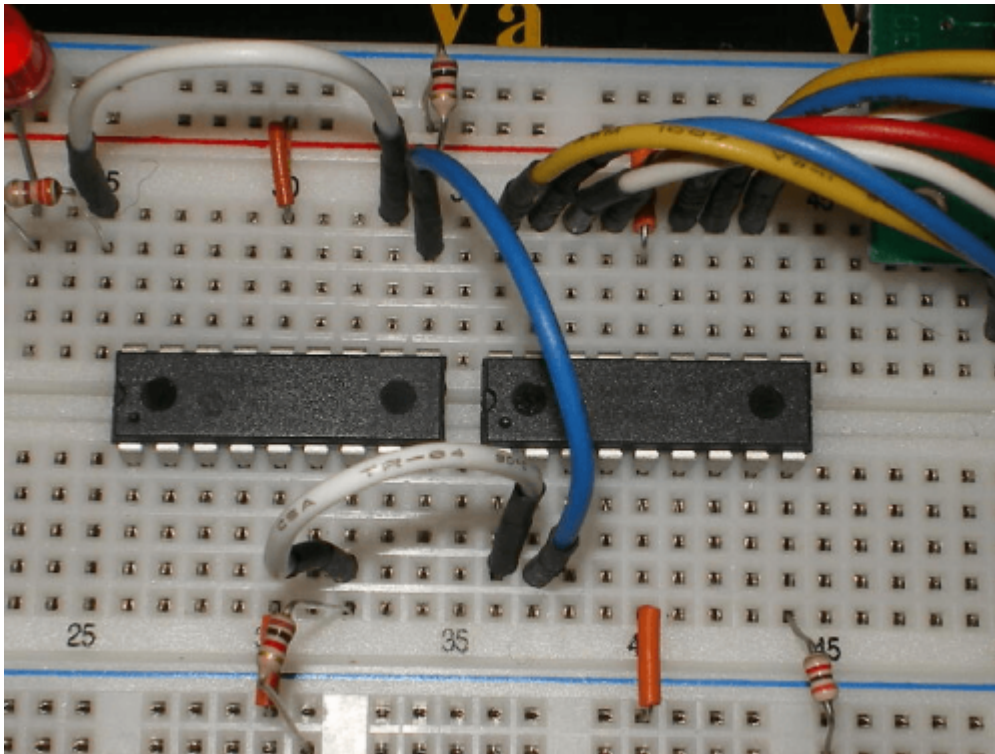
    Lcd_Custom_Chr(1, 6, ':');
    ByteToStr(ss, buf);
    buf[1] = buf[1] == ' ' ? '0' : buf[1];
    Lcd_Custom_Out(1, 7, &buf[1]);
    //
    if (PORTB.F3 == 0) {                //時刻設定
        Soft_I2C_Start();
        Soft_I2C_Write(0x80);
        Soft_I2C_Write(0x03);
        Soft_I2C_Write(12);            //12時
        Soft_I2C_Write(34);            //34分
        Soft_I2C_Write(56);            //56秒
        Soft_I2C_Write(0x01);          //設定指示
        Soft_I2C_Stop();
    }
    //
    Delay_ms(100);
}

//*****
*
```

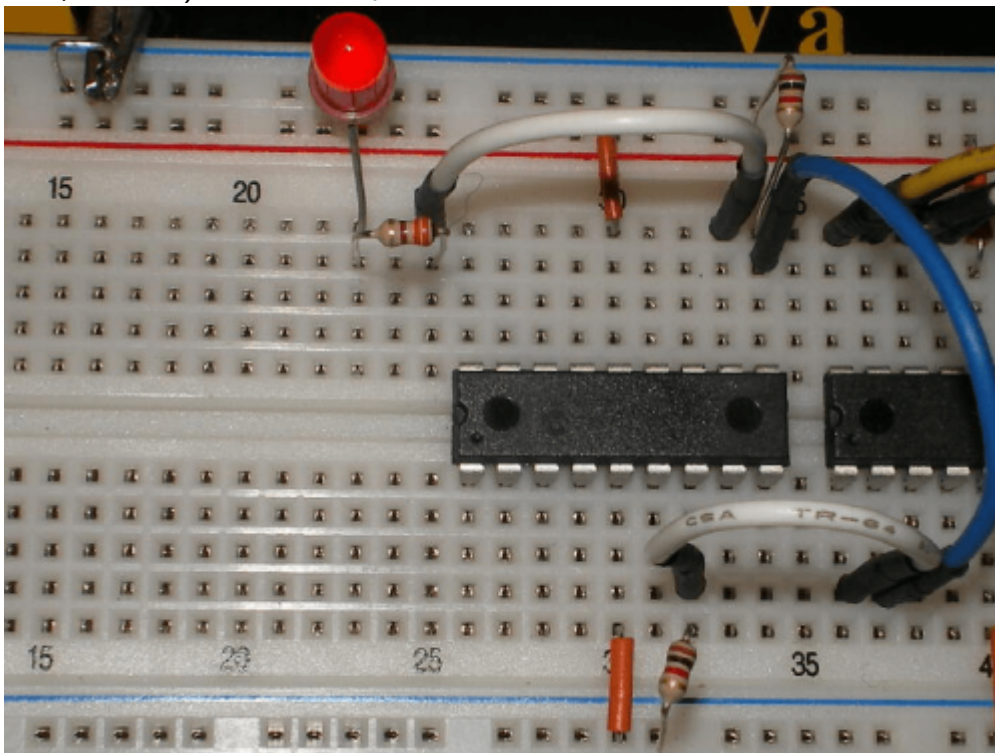
動作確認



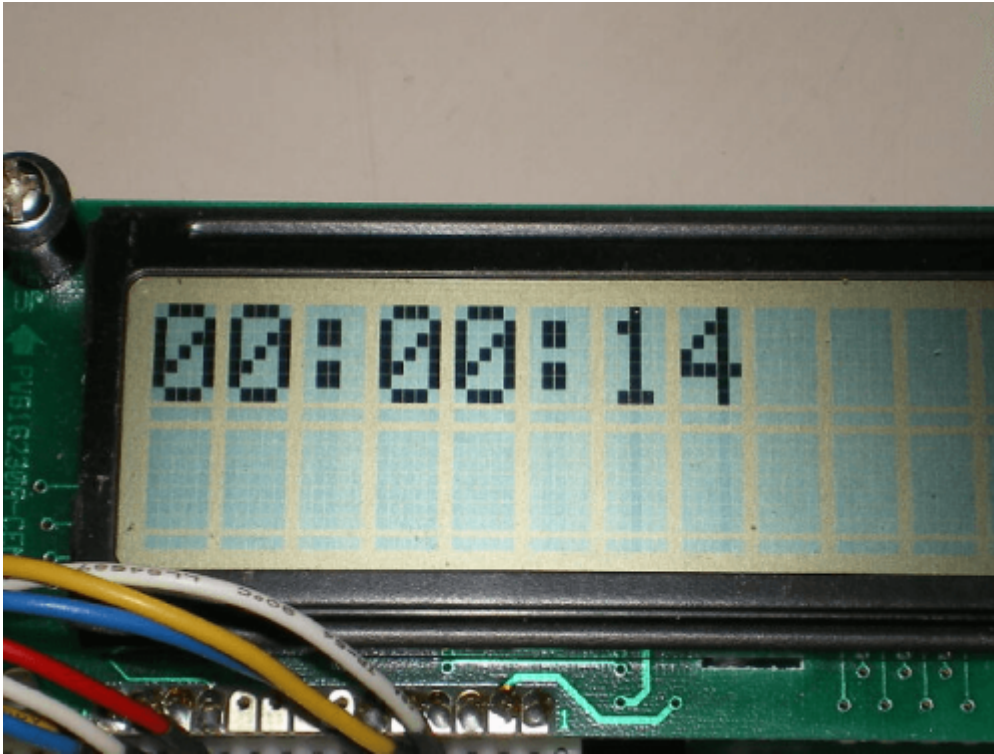
左側:RTC(スレーブ) 今回の主役です。抵抗3個□LED1個だけです。 右側:RTC(マスター)



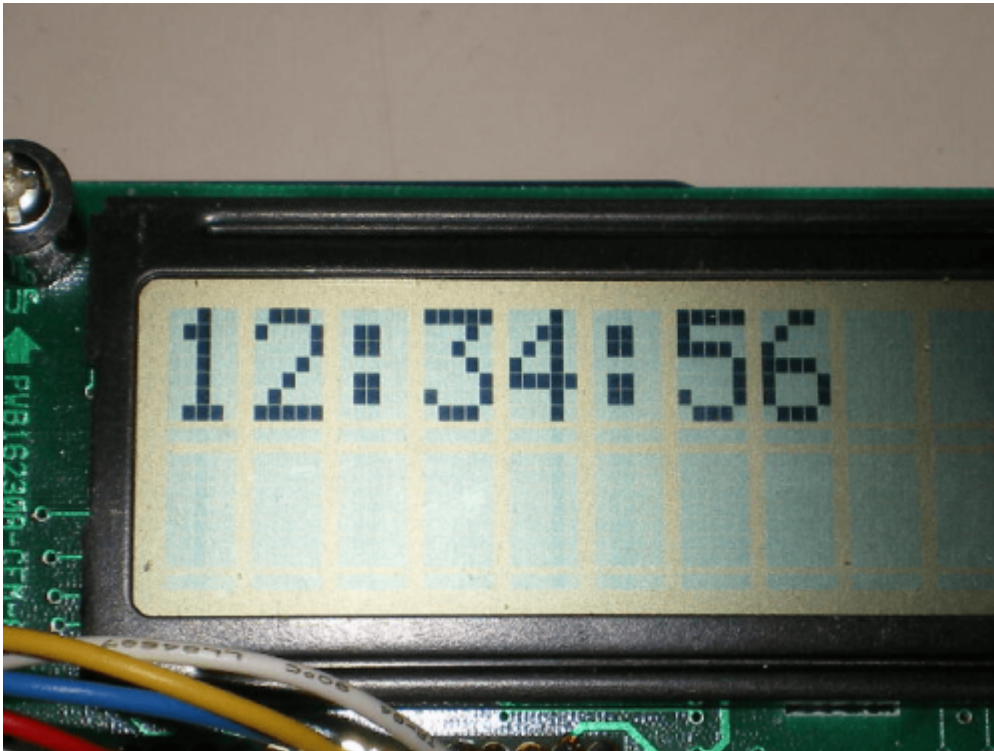
RTC(スレーブ)の拡大図です。



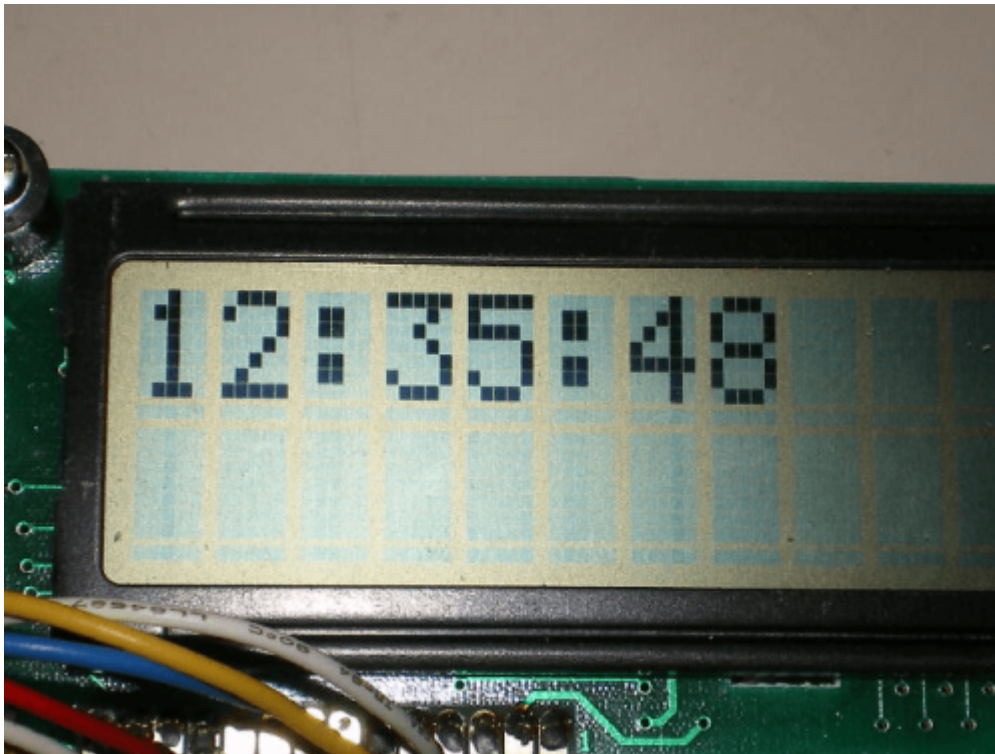
起動直後に、RTC(マスター)がRTC(スレーブ)から時刻データを取得し、表示したところです。



RTC(マスター)からRTC(スレーブ)に対して、時刻を設定(12時34分56秒)したところです。



時刻設定後の、時刻経過を表示したところです。



From:

<http://deepsky.jp/wiki/> - うごくといいな

Permanent link:

<http://deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic16f88:109&rev=1588207499>

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

