

簡易GPS時計V3(自動時刻同期+低消費電力)

概要

以前にも□GPS時計(調整不要)を製作しました。それは、常にGPS受信モジュールを起動させておき、そこから送られてくるデータ(1秒周期)の中から、時刻データのみを抜き出し、LCDに表示するものでした。しかし□GPS受信モジュールは、意外と電流(約200mA)を必要とするので、常時起動では乾電池の駆動には向きません。

そこで、基本的な時刻管理は、内部で通常の精度で行い、時刻の校正を必要とするときだけ□GPSモジュールを起動し、送られてくるデータを利用することにより、時刻の精度維持と低消費電力を図ってみました。

<仕様>

- 高精度のクリスタルオシレータを利用し、基本的な時刻管理は、内部(PIC単体)で行います。最小単位は、0.1秒です。
- 手動または自動(毎時零分)で、GPSモ受信ジュールからのデータを受信し、時刻を校正します。
- 時刻校正の時のみ□GPS受信モジュールを起動し、低消費電力化を図ります。
- スイッチ(SW4)の押下により、時刻データをRS232C経由で送信します。
このデータを利用することにより、高精度な時計機能を実現することが出来ます。

動作原理

<基準時間(0.1sec)> 時刻管理をするための最小単位である□0.1secの基準時間を得ます。

- 精度の高い、クリスタルオシレータ(8.000000MHz)を使用します。
- CCPモジュールをコンペアモードで使用し、0.1secの割り込みを発生させます。

<時刻管理> 基準時間(0.1sec)でクロック変数(clock)を、インクリメント(+1)します。クロック変数(clock)から、時分秒ミリ秒を求める式は、次のようになります。

```
hh = clock / 36000;  
mm = (clock - (36000 * (long)hh)) / 600;  
ss = (clock - (36000 * (long)hh) - (600 * (long)mm)) / 10;  
ms = (clock - (36000 * (long)hh) - (600 * (long)mm)) - (10 * (long)ss);
```

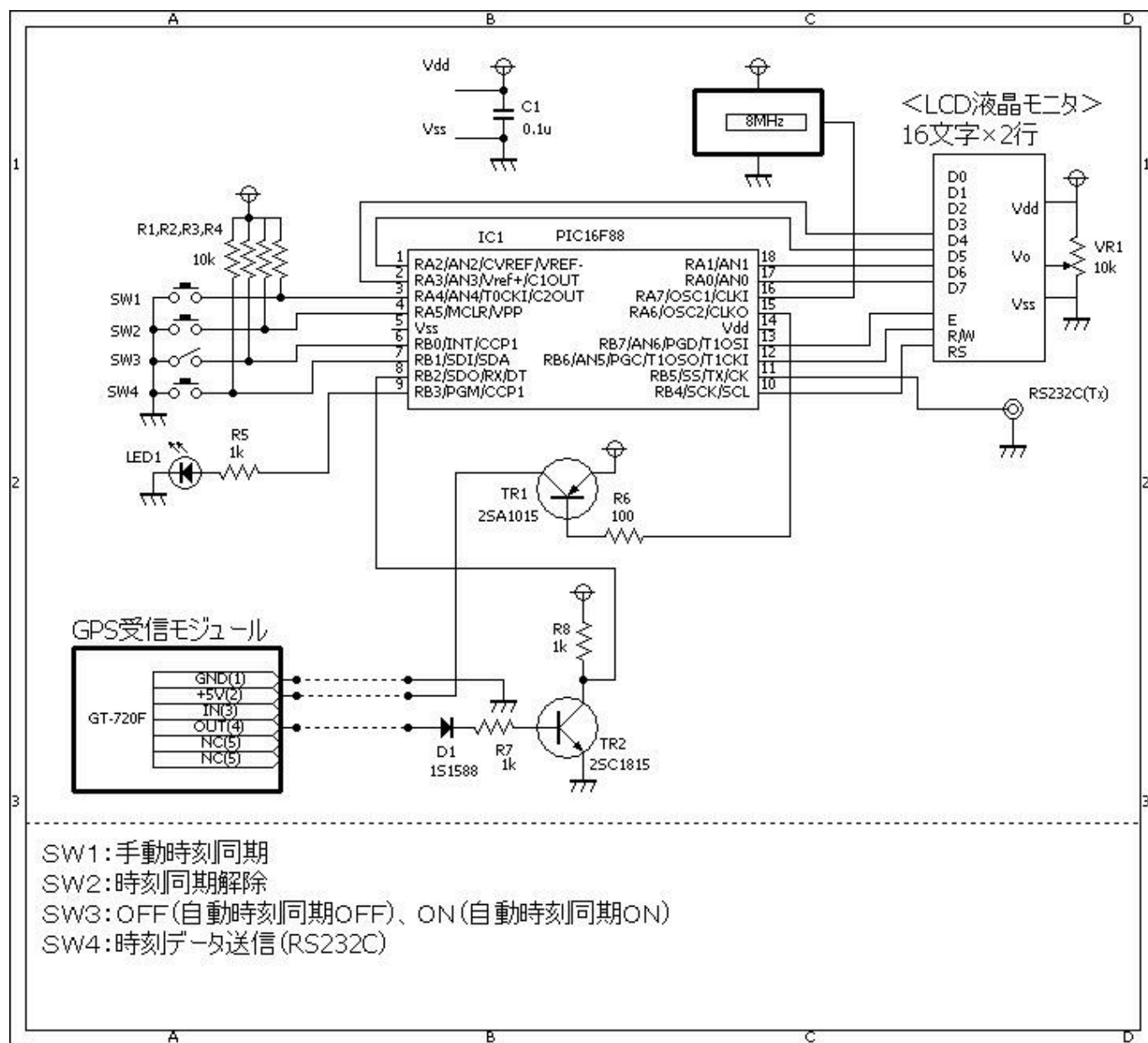
<時刻同期> 手動(スイッチ(SW1)押下)または、自動(スイッチ(SW3)ONおよび毎時零分)により□GPS受信モジュールを起動し、受信した時刻データを元に、クロック変数(clock)を校正します□GPS受信モジュールが、GPS衛星からの受信不可の場合(例えば、屋内などのGPS衛星からの、電波受信状況の悪い場所での本装置の使用時など)には、スイッチ(SW2)押下により、解除することが出来ます。その時には□GPS受信モジュールが保持している、時刻データを元に、クロック変数(clock)を校正します。

※GPS受信モジュールは、時刻同期を実施するときのみ接続し、後は外していても構いません。

<時刻データの送信> 必要に応じて(スイッチ(SW4)の押下)、時刻データをRS232C経由で送信することが出来ます。そのフォーマットは、次のようになります。(例、12時34分56秒□S:start□E:end)

S12:34:56E

回路図



ソースコード

[gps_clock_v2.c](#)

```

//*****
*
/*
□□□□□□時計 (自動時刻同期 & 低消費電力)
*/
//*****
*

```

```
#define      SW1      PORTA.F4
#define      SW2      PORTA.F5
#define      SW3      PORTB.F0
#define      SW4      PORTB.F1

#define      GT720F    PORTA.F6

#define      CR        0x0D
#define      LF        0x0A
#define      FF        0x0C

#define      LED       PORTB.F3
#define      ON        1
#define      OFF       0

//*****
*

static unsigned    long    clock;

void    interrupt()
{
    if (PIR1.CCP1IF == 1) {
        PIR1.CCP1IF = 0;
        //
        clock++;
    }
}

//*****
*

static unsigned    short    flg, len;
static unsigned    short    hd[8], utc[12], latitude[12],
longitude[12];
static unsigned    short    quality[4], satellites[4];

void    gps_recv()
{
    static    unsigned    short    rd;
    //
    flg = 0;
    len = 0;
    //
    while (flg != 8) {
        if (Usart_Data_Ready() == 0)
            continue;
        rd = Usart_Read();
        if (rd == LF) {
            len = 0;
        }
    }
}
```

```
        continue;
    }
    //
    switch (flg) {
    case 0:
        hd[len] = rd;
        len++;
        if (len == 7) {
            len = 0;
            if (strncmp(hd, "$GPGGA,", 7) == 0) {
                flg = 1;
            }
        }
        break;
    case 1:
        if (rd == ',') {
            utc[len] = 0x00;
            len = 0;
            flg = 2;
        } else {
            utc[len] = rd;
            len++;
        }
        break;
    case 2:
        if (rd == ',') {
            latitude[len] = 0x00;
            len = 0;
            flg = 3;
        } else {
            latitude[len] = rd;
            len++;
        }
        break;
    case 3:
        if (rd == ',') {
            len = 0;
            flg = 4;
        }
        break;
    case 4:
        if (rd == ',') {
            longitude[len] = 0x00;
            len = 0;
            flg = 5;
        } else {
            longitude[len] = rd;
            len++;
        }
        break;
    case 5:
```

```
        if (rd == ',') {
            len = 0;
            flg = 6;
        }
        break;
    case 6:
        if (rd == ',') {
            quality[len] = 0x00;
            len = 0;
            flg = 7;
        } else {
            quality[len] = rd;
            len++;
        }
        break;
    case 7:
        if (rd == ',') {
            satellites[len] = 0x00;
            len = 0;
            flg = 8;
        } else {
            satellites[len] = rd;
            len++;
        }
        break;
    }
}

}

}

//*****
*

static char    buf[16], tmp[8];
static unsigned short  hh, mm, ss, ms;

unsigned long  gps_get_clock()
{
    static unsigned long  t;
    //
    while (1) {
        gps_rcv();
        Lcd_Custom_Out(2, 1, utc);
        Lcd_Custom_Out(2, 13, quality);
        Lcd_Custom_Out(2, 15, satellites);
        if ((quality[0] != '0') || (SW2 == 0))
            break;
    }
    //
    hh = ((utc[0] - '0') * 10) + (utc[1] - '0');
    mm = ((utc[2] - '0') * 10) + (utc[3] - '0');
    ss = ((utc[4] - '0') * 10) + (utc[5] - '0');
```

```

    ms = ((utc[7] - '0') * 100) + ((utc[8] - '0') * 10) + (utc[8] -
'0');
    t = ((long)hh * 3600) + ((long)mm * 60) + ((long)ss * 10) +
((long)ms / 100);
    return (t + (9 * 3600));    //UTC -> JST
}

//*****
*

void    Usart_Write_Str(unsigned short *pData)
{
    while (*pData != 0x00) {
        Usart_Write(*pData);
        pData++;
        Delay_ms(10);
    }
}

//*****
*

void    ByteToStr2(unsigned short number, char *output)
{
    ByteToStr(number, output);
    output[0] = (output[1] == ' ') ? '0' : output[1];
    output[1] = output[2];
    output[2] = 0x00;
}

//*****
*

void    main()
{
    //
//    OSCCON = 0b01110000;
    CMCON  = 0b00000111;
    ANSEL  = 0b00000000;
    TRISA  = 0b10110000;
    TRISB  = 0b00000111;
    //
    LED = OFF;
    GT720F = 1;
    // 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 = 1;
PIR1.CCP1IF = 0;
CCP1CON.CCP1M3 = 1;
CCP1CON.CCP1M2 = 0;
CCP1CON.CCP1M1 = 1;
CCP1CON.CCP1M0 = 1;
CCPR1L = 0xA8; // 0.1sec...(1÷8000000)*4*8*25000
CCPR1H = 0x61; //
//
Lcd_Custom_Config(&PORTA, 0, 1, 2, 3, &PORTB, 4, 6, 7);
Lcd_Custom_Cmd(LCD_CURSOR_OFF);
Lcd_Custom_Cmd(LCD_CLEAR);
Lcd_Custom_Out(1, 1, "GpsClock v2");
Delay_ms(500);
Lcd_Custom_Cmd(LCD_CLEAR);
Lcd_Custom_Chr(1, 3, ':');
Lcd_Custom_Chr(1, 6, ':');
Lcd_Custom_Chr(1, 9, '.');
//
Usart_Init(9600);
//
clock = 0;
// 割り込みを許可する。
INTCON.PEIE = 1;
INTCON.GIE = 1;
//
T1CON.TMR1ON = 1;
//
while (1) {
    //clock変数から、時、分、秒、ミリ秒を求める。
    hh = clock / 36000;
    mm = (clock - (36000 * (long)hh)) / 600;
    ss = (clock - (36000 * (long)hh) - (600 * (long)mm)) / 10;
    ms = (clock - (36000 * (long)hh) - (600 * (long)mm)) - (10 *
(long)ss);
    //時を表示する。
    ByteToStr2(hh, buf);
    Lcd_Custom_Out(1, 1, buf);
    //分を表示する。
    ByteToStr2(mm, buf);
    Lcd_Custom_Out(1, 4, buf);
    //秒を表示する。
    ByteToStr2(ss, buf);
    Lcd_Custom_Out(1, 7, buf);
    //ミリ秒を表示する。
    ByteToStr(ms, buf);
    Lcd_Custom_Out(1, 10, &buf[2]);
    //GPSと時刻同期する。(手動)
    if (SW1 == 0) {
```

```

        GT720F = 0;
        clock = gps_get_clock();
        GT720F = 1;
    }
    //GPSと時刻同期する。(自動:毎時0分0秒)
    if ((SW3 == 0) && (mm == 0) && (ss == 0)) {
        GT720F = 0;
        clock = gps_get_clock();
        GT720F = 1;
    }
    //□□□□□□に時刻データを送信する。
    if (SW4 == 0) {
        while (SW4 == 0) {
            Delay_ms(100);
        }
        LED = ON;
        Usart_Write_Str("S");
        ByteToStr2(hh, buf);
        Usart_Write_Str(buf);
        Usart_Write_Str(":");
        ByteToStr2(mm, buf);
        Usart_Write_Str(buf);
        Usart_Write_Str(":");
        ByteToStr2(ss, buf);
        Usart_Write_Str(buf);
        Usart_Write_Str("E");
        LED = OFF;
    }
}

//*****
*
```

日付表示を追加したバージョンです□(mikroC PRO版)

gps_clock_v21.c

```

//*****
*
/*
□□□□□□時計(自動時刻同期&低消費電力□V2.1□□□日付表示を追加
*/
//*****
*

#define          SW1          PORTA.F4
#define          SW2          PORTA.F5
#define          SW3          PORTB.F0
#define          SW4          PORTB.F1
```

```

#define          GT720F          PORTA.F6

#define          CR              0x0D
#define          LF              0x0A
#define          FF              0x0C

#define          LED             PORTB.F3
#define          ON              1
#define          OFF             0

//*****
*

static          unsigned          long          clock;

void            interrupt()
{
    if (PIR1.CCP1IF == 1) {
        PIR1.CCP1IF = 0;
        //
        clock++;
    }
}

//*****
*
//$GPRMC,015315.598,A,3456.7835,N,13511.2730,E,000.0,000.0,160310,,,A*6
0

static          unsigned          short          flg, len;
static          unsigned          short          hd[8], utc[12], stat[4],
latitude[12], longitude[12], date[8];

void            gps_recv()
{
    static          unsigned          short          rd;
    //
    flg = 0;
    len = 0;
    //
    while (flg != 10) {
        if (UART1_Data_Ready() == 0)
            continue;
        rd = UART1_Read();
        if (rd == LF) {
            len = 0;
            continue;
        }
        //
        switch (flg) {

```

```
case 0:
    hd[len] = rd;
    len++;
    if (len == 7) {
        len = 0;
        if (strncmp(hd, "$GPRMC,", 7) == 0) {
            flg = 1;
        }
    }
    break;
case 1:
    if (rd == ',') {
        utc[len] = 0x00;
        len = 0;
        flg = 2;
    } else {
        utc[len] = rd;
        len++;
    }
    break;
case 2:
    if (rd == ',') {
        stat[len] = 0x00;
        len = 0;
        flg = 3;
    } else {
        stat[len] = rd;
        len++;
    }
    break;
case 3:
    if (rd == ',') {
        latitude[len] = 0x00;
        len = 0;
        flg = 4;
    } else {
        latitude[len] = rd;
        len++;
    }
    break;
case 4:
    if (rd == ',') {
        len = 0;
        flg = 5;
    }
    break;
case 5:
    if (rd == ',') {
        longitude[len] = 0x00;
        len = 0;
        flg = 6;
```

```
        } else {
            longitude[len] = rd;
            len++;
        }
        break;
case 6:
    if (rd == ',') {
        len = 0;
        flg = 7;
    }
    break;
case 7:
    if (rd == ',') {
        len = 0;
        flg = 8;
    }
    break;
case 8:
    if (rd == ',') {
        len = 0;
        flg = 9;
    }
    break;
case 9:
    if (rd == ',') {
        date[len] = 0x00;
        len = 0;
        flg = 10;
    } else {
        date[len] = rd;
        len++;
    }
    break;
    }
}

}

//*****
*

static      char      buf[16], tmp[8];
static      unsigned   short   hh, mm, ss, ms;

unsigned     long      gps_get_clock()
{
    static     unsigned   long      t;
    //
    while (1) {
        gps_recv();
        Lcd_Out(2, 1, utc);
        date[4] = '/';
    }
}
```

```

        date[5] = date[0];
        date[6] = date[1];
        date[7] = 0x00;
        memcpy(date, &date[2], 6);
        Lcd_Out(1, 12, date);
        Lcd_Out(2, 16, stat);
        if ((stat[0] == 'A') || (SW2 == 0))
            break;
    }
    //
    hh = ((utc[0] - '0') * 10) + (utc[1] - '0');
    mm = ((utc[2] - '0') * 10) + (utc[3] - '0');
    ss = ((utc[4] - '0') * 10) + (utc[5] - '0');
    ms = ((utc[7] - '0') * 100) + ((utc[8] - '0') * 10) + (utc[8] -
'0');
    t = ((long)hh * 3600) + ((long)mm * 600) + ((long)ss * 10) +
((long)ms / 100);
    return (t + (9 * 3600));          //UTC -> JST
}

//*****
*

void    UART1_Write_Str(unsigned short *pData)
{
    while (*pData != 0x00) {
        UART1_Write(*pData);
        pData++;
        Delay_ms(10);
    }
}

//*****
*

void    ByteToStr2(unsigned short number, char *output)
{
    ByteToStr(number, output);
    output[0] = (output[1] == ' ') ? '0' : output[1];
    output[1] = output[2];
    output[2] = 0x00;
}

//*****
*

// Lcd pinout settings
sbit LCD_RS   at   RB4_bit;
sbit LCD_RW   at   RB6_bit;
sbit LCD_EN   at   RB7_bit;
sbit LCD_D7   at   RA0_bit;

```

```
sbit LCD_D6 at RA1_bit;
sbit LCD_D5 at RA2_bit;
sbit LCD_D4 at RA3_bit;

// Pin direction
sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_RW_Direction at TRISB6_bit;
sbit LCD_EN_Direction at TRISB7_bit;
sbit LCD_D7_Direction at TRISA0_bit;
sbit LCD_D6_Direction at TRISA1_bit;
sbit LCD_D5_Direction at TRISA2_bit;
sbit LCD_D4_Direction at TRISA3_bit;

void init_lcd()
{
    LCD_RW_Direction = 0;
    LCD_RW = 0;
    Lcd_Init();
    Lcd_Cmd(_LCD_CURSOR_OFF);
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Out(1, 1, "GpsClock v2.10");
    Delay_ms(500);
    Lcd_Cmd(_LCD_CLEAR);
}

//*****
*

void init_timer()
{
    // 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 = 1;
    PIR1.CCP1IF = 0;
    CCP1CON.CCP1M3 = 1;
    CCP1CON.CCP1M2 = 0;
    CCP1CON.CCP1M1 = 1;
    CCP1CON.CCP1M0 = 1;
    CCPR1L = 0xA8; // 0.1sec...(1÷8000000)*4*8*25000
    CCPR1H = 0x61; //
}

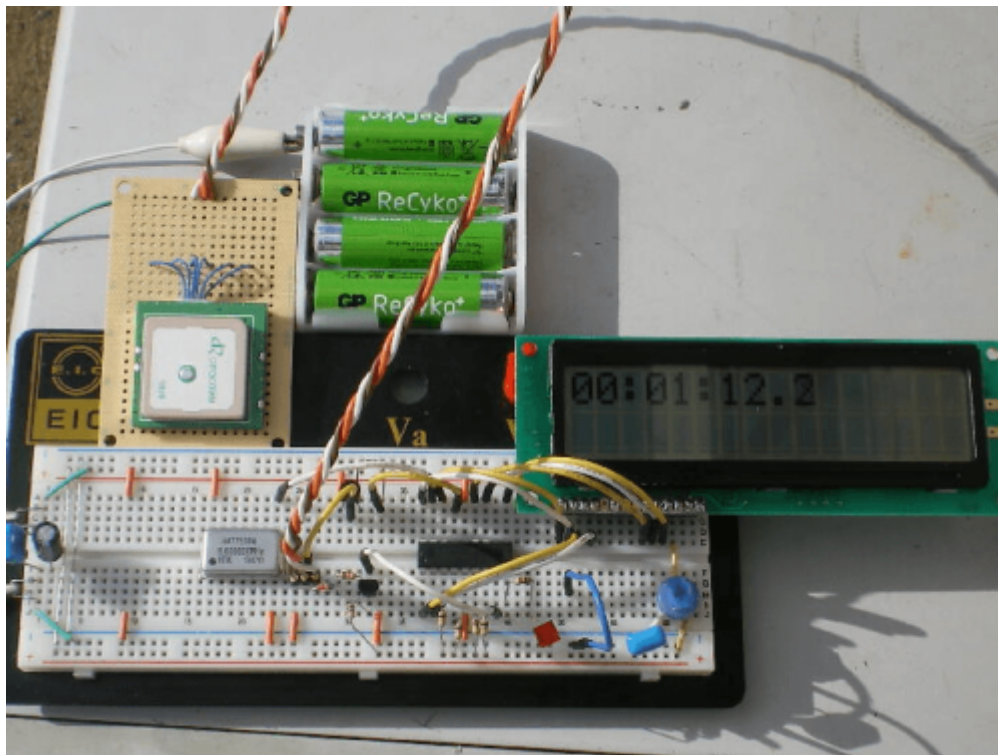
//*****
*
```

```
void    main()
{
    //
    //    OSCCON = 0b01110000;
    CMCON  = 0b00000111;
    ANSEL  = 0b00000000;
    TRISA  = 0b10110000;
    TRISB  = 0b00000111;
    //
    LED = OFF;
    GT720F = 1;
    //
    init_lcd();
    Lcd_Chr(1, 3, ':');
    Lcd_Chr(1, 6, ':');
    Lcd_Chr(1, 9, '.');
    //
    UART1_Init(9600);
    //
    clock = 0;
    strcpy(date, "??/??");
    //
    init_timer();
    // 割り込みを許可する。
    INTCON.PEIE = 1;
    INTCON.GIE = 1;
    //
    T1CON.TMR1ON = 1;
    //
    while (1) {
        //clock変数から、時、分、秒、ミリ秒を求める。
        hh = clock / 36000;
        mm = (clock - (36000 * (long)hh)) / 600;
        ss = (clock - (36000 * (long)hh) - (600 * (long)mm)) /
10;
        ms = (clock - (36000 * (long)hh) - (600 * (long)mm)) -
(10 * (long)ss);
        //時を表示する。
        ByteToStr2(hh, buf);
        Lcd_Out(1, 1, buf);
        //分を表示する。
        ByteToStr2(mm, buf);
        Lcd_Out(1, 4, buf);
        //秒を表示する。
        ByteToStr2(ss, buf);
        Lcd_Out(1, 7, buf);
        //ミリ秒を表示する。
        ByteToStr(ms, buf);
        Lcd_Out(1, 10, &buf[2]);
        //GPSと時刻同期する。(手動)
```

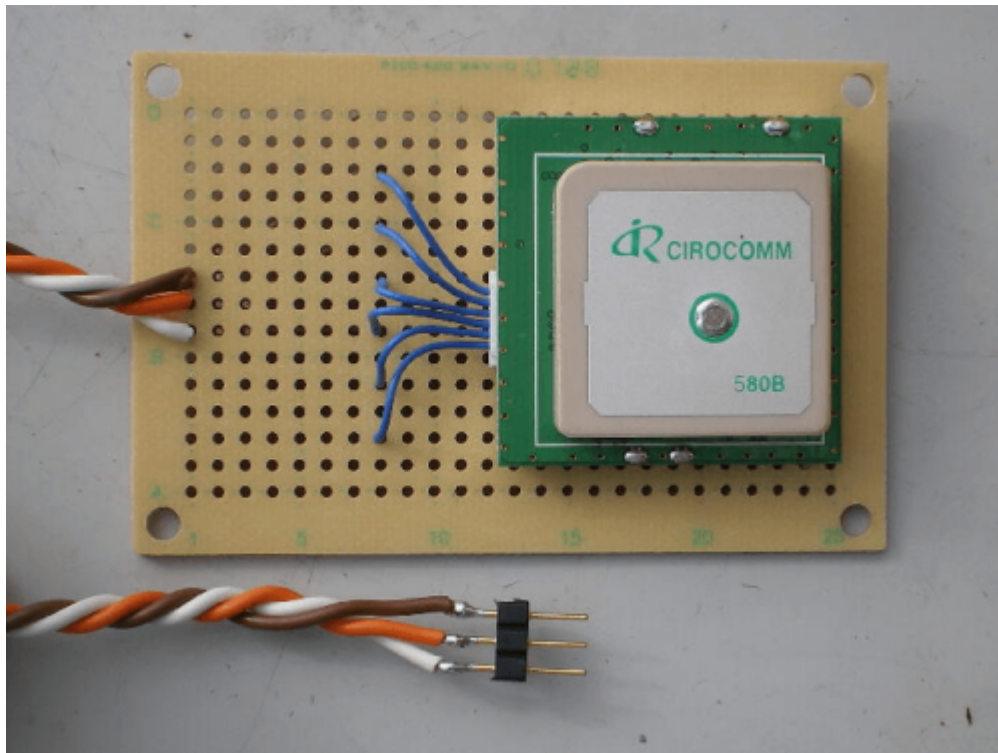
```
if (SW1 == 0) {
    GT720F = 0;
    clock = gps_get_clock();
    GT720F = 1;
}
//GPSと時刻同期する。(自動:毎時0分0秒)
if ((SW3 == 0) && (mm == 0) && (ss == 0)) {
    GT720F = 0;
    clock = gps_get_clock();
    GT720F = 1;
}
//□□□□□□に時刻データを送信する。
if (SW4 == 0) {
    while (SW4 == 0) {
        Delay_ms(100);
    }
    LED = ON;
    UART1_Write_Str("S");
    UART1_Write_Str(date);
    UART1_Write_Str(" ");
    ByteToStr2(hh, buf);
    UART1_Write_Str(buf);
    UART1_Write_Str(":");
    ByteToStr2(mm, buf);
    UART1_Write_Str(buf);
    UART1_Write_Str(":");
    ByteToStr2(ss, buf);
    UART1_Write_Str(buf);
    UART1_Write_Str("E");
    LED = OFF;
}
}

//*****
*
```

動作確認

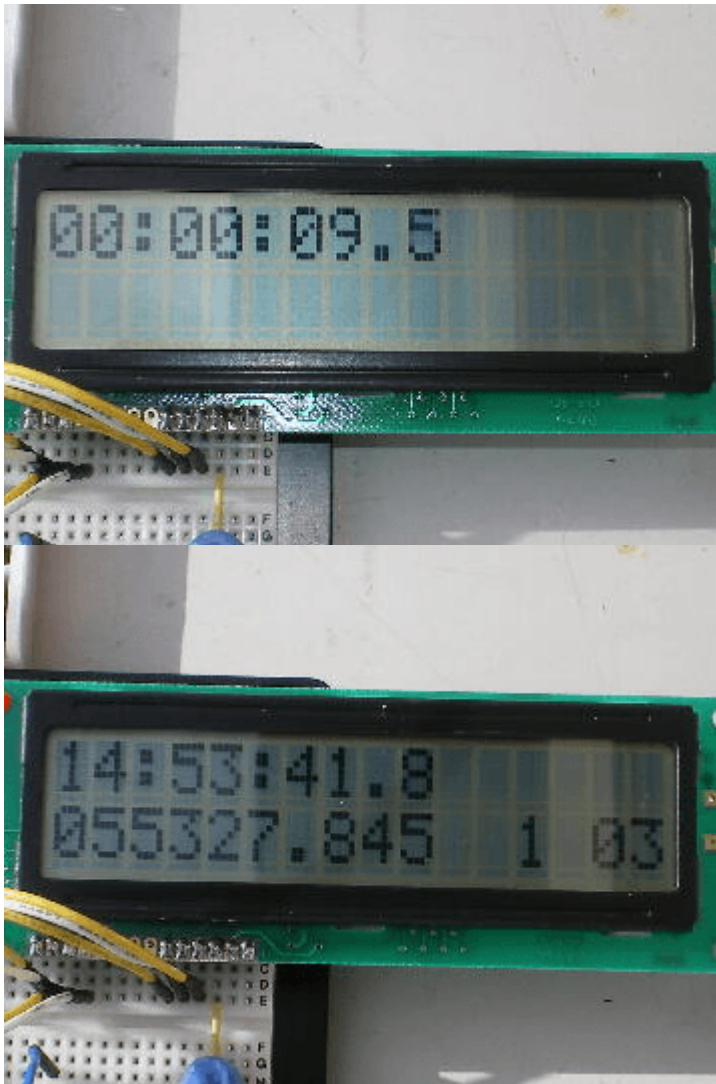


GT-720Fは、接続線が細く切れやすいので、基板に固定し、少し太めの線に変換します。

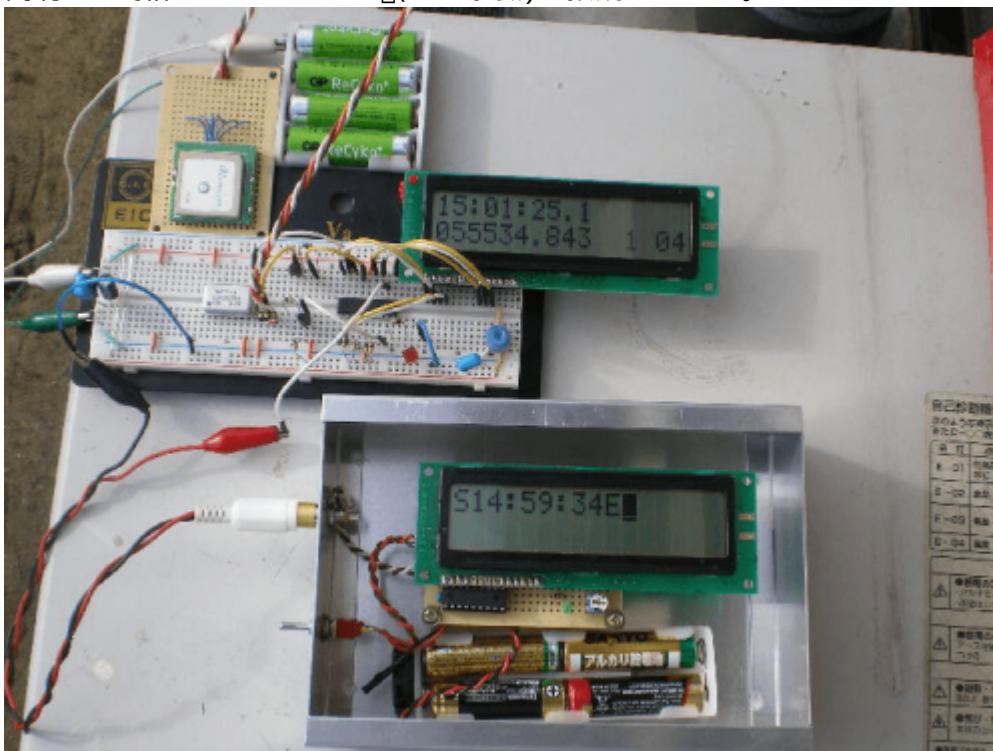


左側:起動直後(時刻同期前)です。(時刻データは、現在時刻とは全く無関係です)右側:SW1を押下すると、時刻同期が行われ、正常な現在時刻を表示します。

上の行=JST(Japan Standard Time□日本標準時)です。
 下の行の左=UTC(Coordinated Universal Time□協定世界時)です。
 下の行の右=GPSのクオリティ(1)と受信衛星数(03)です。



簡易GPS時計V3にLCDモニター(3.3V駆動)を接続しました。



SW4を押下するとLCDモニターに、送信されてきた時刻データが表示されます。



如何ですか? 使用するクリスタルオシレータの精度によっては、毎時零分の時刻同期ではなく、一日1回の時刻同期でも十分 実用になると思います。

<参考> 消費電流は、

- 通常時(GPS受信モジュールOFF) 約10mA
- 校正時(GPS受信モジュールON) 約200mA

でした。

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:pic16f88:133>

Last update: **2020/05/01 18:34**

