```
/************************************************
 SHT-11 THERMOMETER driver function
            By nobcha all right reserved

  Ver 1. 10/14/2013 for SHT-11 & PIC16F1827

   Development Circumstance
   MPLAB IDE V8.92   HiTECH C V9.83
************************************************ */
#define _LEGACY_HEADERS
#define _XTAL_FREQ 16000000

#include <htc.h>

#include "sht.h"
extern char PBuf;
//-------------------------------------------------
/* エレキジャック  マイコンの応用「2線式 温・湿度センサSHT11」を参考。
作者の中尾さんに感謝します。
http://www.eleki-jack.com/mycom2/pic/cat94/2_sht11/
                                                    */
// SCKを"H"レベルに設定する関数
void SCK_H(void) {
   PBuf |= (1<<bitSCK);                          // SCK="H"
   SHT_PORT = PBuf;
}

// SCKを"L"レベルに設定する関数
void SCK_L(void) {
    PBuf &= ~(1<<bitSCK);                  // SCK="L"
    SHT_PORT = PBuf;
}

//-------------------------------------------------
/*     初期化 SHTInit()
  SHTを使うにあたり、I/Oポートの初期化。
  SCK、DATA両ポートの入出力方向を設定し、
  両ポートの状態をアイドルにする。
  "PBuf"はグローバル変数、出力ポート用バッファ。
  設定値を全ビット同時に更新するため一時変数で
  出力値を設定し  レジスタへ書き込む。
                                        */

//
// SHTポート初期化
//
void SHTInit(void) {
          unsigned char buf;
    SHT_TRIS = (1<<bitDATA)| SHT_TRIS;        // DATA input(Hiz-H)
```

```
    PBuf |= ~(1<<bitDATA);              // DATA = "L"
    PBuf &= ~(1<<bitSCK);                       // SCK = "L"
    SHT_PORT = PBuf;                    // 初期値を出力

    buf = SHT_TRIS;
    buf &= ~(1<<bitSCK);                // SCK output ("L")
    buf |= (1<<bitDATA);                // DATA input (HiZ-"H")
    SHT_TRIS = buf;                             // I/Oを設定
}

//---------------------------------------------------------------
/*  TS(トランスミッション・スタート)シーケンス発行 SHTTSSeq()
  TSシーケンスを発行する関数です。
  コマンドを送信する前に必ずTSシーケンスを発行する必要があります。
                                                              */

//
// SHT TSシーケンス
//
void SHTTSSeq(void) {
    DATA_H();
    SCK_H();

        DATA_L();
    SCK_L();

    SCK_H();
    DATA_H();

    SCK_L();
}

//-------------------------------------------------------------
/*  バイト・データ送信 SHTWrite()
  バイト・データそマスタ送信(書き込み)する関数。
  データ送信した後にSHT11がスレーブの送信したACKデータを受信。

  マスタ送信するデータは引数の"dat"で渡します。
  マスタ送信したあとにSHTがスレーブ送信した"ACK"/"NOACK"は、
  関数の戻り値で得られます。戻り値が"0"のときが"ACK"です。

  送信するデータは最上位から1ビットずつ取り出して、
  その値をDATAポートに設定しながら、
  SCKポートからクロック・パルスを出力します。
  SCKパルスの立ち下がりエッジで出力データを更新する。
                                                        */
```

```
unsigned char SHTWrite(unsigned char dat) {
    unsigned char i, rdat;                          // DATAビットの出力
    for(i = 0; i < 8; i++) {
        SCK_L();
        if(dat & 0x80) { // MSB
                                                    // "1"
            DATA_H();
        } else {
                                                    // "0"
            DATA_L();
        }
        SCK_H();
        dat <<= 1;                                  // 次のビット
    }
        SCK_L();

        // ACKビットの入力
    DATA_H();                                       // 入力に切り替え

        SCK_H();                                    // ↑ edge
        if(SHT_PORT & (1<<bitDATA)) {           // read DATA
                                                    // 1
            rdat = 1;                               // NOACK
    } else {
                                                    // 0
            rdat = 0;                               // ACK
    }
    SCK_L();

    return rdat;
}

//-----------------------------------------------------------
/*      バイト・データ受信 SHTRead()
  バイト・データをSHT11からマスタ受信（読み出し）関数。
  データを受信した後に"ACK"/"NOACK"をマスタ送信。
  引数の"ack"を0にすると"ACK"、"1"にすると"NOACK"で応答。
  受信したデータは関数の戻り値。

  SCKポートからクロック・パルスを出力しながら、
  DATAポートで受信したビットを順番に変数へ格納して、
  それを8回繰り返すことにより1バイトのデータを得る。
  受信データはSCKパルスの立ち上がりエッジで取り込み。
                                                    */
```

```
unsigned char SHTRead(unsigned char ack) {
        unsigned char i, dat;

        dat = 0;
        DATA_H();                       // 入力に切り替え

// DATAビットの入力
        for(i = 0; i < 8; i++) {
                dat <<= 1;              // 次のビット
        SCK_H();                        // ↑ edge
        if(SHT_PORT & (1<<bitDATA)) {       // read DATA
                                        // "1"
        dat |= 1;
        }
        SCK_L();
        }

// ACKビットの出力
        if(!ack) {
        DATA_L();
        } else {
        DATA_H();
        }
   SCK_H();
   SCK_L();                             // ↓ edge

   DATA_H();                            // 入力に切り替え

   return dat;
}

/*
湿度値 RHlinear = －4 ＋ 0.0405×(AD値) － 2.8×10-6×(AD値)2
                                                        */

int CalcHR10(int ADValHR) {
        int ad,x1, x2, x3, x4, cd10, b10;

        // 桁ごとの数値を取り出す
        ad = ADValHR;
        x1 = ad / 1000;
        ad = ad - 1000 * x1;
         x2 = ad / 100;
        ad = ad - 100 * x2;
        x3 = ad / 10;
        x4 = ad - 10 * x3;
```

```
        // 桁ごとにA-D値を掛け算する
        x1 = ADValHR * x1;
        x2 = (ADValHR * x2 + 5) / 10;           // 小数点以下四捨五入
        x3 = (ADValHR * x3 + 50) / 100;         // 小数点以下四捨五入
        x4 = (ADValHR * x4 + 500) / 1000;       // 小数点以下四捨五入

        x1 = x1 + x2 + x3 + x4;                 // 計算結果を合成
        x2 = (x1 + 5) / 10;                     // 1/10する. 小数点以下四捨五入

        cd10 = (28 * x2 + 50) / 100;

        x1 = (ADValHR * 4 + 5) / 10;
        x2 = (ADValHR * 5 + 500) / 1000;
        b10 = x1 + x2;

        return -40 + b10 - cd10;            // 湿度値を10倍した整数値
}
```