SENSIRION
THE SENSOR COMPANY

SHTxx
Humidity & Temperature
Sensmitter

Application Note
Sample Code

1 Introduction

This application note gives an example for microcontroller C code. It includes code for:

- Readout of Humidity (RH) or Temperature (T) from SHTxx with basic error handling
- Calculation of RH linearization and temperature compensation
- Access to status register
- Dewpoint calculation from RH and T
- UART handling (to send the final data away, e.g. to a PC)

2 Sample Code

```
/***********************************************************************************
Project:          SHT11 demo program (V2.0)
Filename:         SHT11.c

Prozessor:        80C51 family
Compiler:         Keil Version 6.14

Autor:            MST
Copyrigth:        (c) Sensirion AG
***********************************************************************************/

#include <AT89s53.h> //Microcontroller specific library, e.g. port definitions
#include <intrins.h> //Keil library (is used for _nop()_ operation)
#include <math.h>    //Keil library
#include <stdio.h>   //Keil library

typedef union
{ unsigned int i;
  float f;
} value;

//----------------------------------------------------------------------------------
// modul-var
//----------------------------------------------------------------------------------
enum {TEMP,HUMI};

#define DATA   P1_1
#define SCK    P1_0

#define noACK 0
#define ACK   1
                            //adr  command  r/w
#define STATUS_REG_W 0x06   //000   0011    0
#define STATUS_REG_R 0x07   //000   0011    1
#define MEASURE_TEMP 0x03   //000   0001    1
#define MEASURE_HUMI 0x05   //000   0010    1
#define RESET        0x1e   //000   1111    0

//----------------------------------------------------------------------------------
char s_write_byte(unsigned char value)
//----------------------------------------------------------------------------------
// writes a byte on the Sensibus and checks the acknowledge
{
  unsigned char i,error=0;
  for (i=0x80;i>0;i/=2)             //shift bit for masking
  { if (i & value) DATA=1;          //masking value with i , write to SENSI-BUS
    else DATA=0;
    SCK=1;                          //clk for SENSI-BUS
    _nop_();_nop_();_nop_();        //pulswith approx. 5 us
    SCK=0;
  }
  DATA=1;                           //release DATA-line
  SCK=1;                            //clk #9 for ack
  error=DATA;                       //check ack (DATA will be pulled down by SHT11)
  SCK=0;
  return error;                     //error=1 in case of no acknowledge
}

//----------------------------------------------------------------------------------
char s_read_byte(unsigned char ack)
//----------------------------------------------------------------------------------
// reads a byte form the Sensibus and gives an acknowledge in case of "ack=1"
{
  unsigned char i,val=0;
  DATA=1;                           //release DATA-line
  for (i=0x80;i>0;i/=2)             //shift bit for masking
  { SCK=1;                          //clk for SENSI-BUS
    if (DATA) val=(val | i);        //read bit
    SCK=0;
  }
  DATA=!ack;                        //in case of "ack==1" pull down DATA-Line
  SCK=1;                            //clk #9 for ack
  _nop_();_nop_();_nop_();          //pulswith approx. 5 us
  SCK=0;
  DATA=1;                           //release DATA-line
  return val;
}

//----------------------------------------------------------------------------------
void s_transstart(void)
//----------------------------------------------------------------------------------
// generates a transmission start
//       _____         ________
// DATA:      |_______|
//           ___     ___
// SCK : ___|   |___|   |______
{
   DATA=1; SCK=0;                   //Initial state
   _nop_();
   SCK=1;
   _nop_();
   DATA=0;
   _nop_();
   SCK=0;
   _nop_();_nop_();_nop_();
   SCK=1;
   _nop_();
   DATA=1;
   _nop_();
   SCK=0;
}

//----------------------------------------------------------------------------------
void s_connectionreset(void)
//----------------------------------------------------------------------------------
// communication reset: DATA-line=1 and at least 9 SCK cycles followed by transstart
//       _____________________________________________________         ________
// DATA:                                                      |_______|
//          _    _    _    _    _    _    _    _    _        ___     ___
// SCK : __| |__| |__| |__| |__| |__| |__| |__| |__| |______|   |___|   |______
{
  unsigned char i;
  DATA=1; SCK=0;                    //Initial state
  for(i=0;i<9;i++)                  //9 SCK cycles
  { SCK=1;
    SCK=0;
  }
  s_transstart();                   //transmission start
}

//----------------------------------------------------------------------------------
char s_softreset(void)
//----------------------------------------------------------------------------------
// resets the sensor by a softreset
{
  unsigned char error=0;
  s_connectionreset();              //reset communication
  error+=s_write_byte(RESET);       //send RESET-command to sensor
  return error;                     //error=1 in case of no response form the sensor
}

//----------------------------------------------------------------------------------
char s_read_statusreg(unsigned char *p_value, unsigned char *p_checksum)
//----------------------------------------------------------------------------------
// reads the status register with checksum (8-bit)
{
  unsigned char error=0;
  s_transstart();                   //transmission start
  error=s_write_byte(STATUS_REG_R); //send command to sensor
  *p_value=s_read_byte(ACK);        //read status register (8-bit)
  *p_checksum=s_read_byte(noACK);   //read checksum (8-bit)
  return error;                     //error=1 in case of no response form the sensor
}

//----------------------------------------------------------------------------------
char s_write_statusreg(unsigned char *p_value)
//----------------------------------------------------------------------------------
// writes the status register with checksum (8-bit)
{
  unsigned char error=0;
  s_transstart();                   //transmission start
  error+=s_write_byte(STATUS_REG_W);//send command to sensor
  error+=s_write_byte(*p_value);    //send value of status register
  return error;                     //error>=1 in case of no response form the sensor
}

//----------------------------------------------------------------------------------
char s_measure(unsigned char *p_value, unsigned char *p_checksum, unsigned char mode)
//----------------------------------------------------------------------------------
// makes a measurement (humidity/temperature) with checksum
{
  unsigned error=0;
  unsigned int i;

  s_transstart();                   //transmission start
  switch(mode){                     //send command to sensor
    case TEMP  : error+=s_write_byte(MEASURE_TEMP); break;
    case HUMI  : error+=s_write_byte(MEASURE_HUMI); break;
    default     : break;
  }
  for (i=0;i<65535;i++) if(DATA==0) break; //wait until sensor has finished the measurement
  if(DATA) error+=1;                // or timeout (~2 sec.) is reached
  *(p_value)  =s_read_byte(ACK);    //read the first byte (MSB)
  *(p_value+1)=s_read_byte(ACK);    //read the second byte (LSB)
  *p_checksum =s_read_byte(noACK);  //read checksum
  return error;
}

//----------------------------------------------------------------------------------
void init_uart()
//----------------------------------------------------------------------------------
// Initializes the UART so the final data can be sent away, e.g. to a PC
//9600 bps @ 11.059 MHz
{SCON  = 0x52;
 TMOD  = 0x20;
 TCON  = 0x69;
 TH1   = 0xfd;
}

//----------------------------------------------------------------------------------------
void calc_sth11(float *p_humidity ,float *p_temperature)
//----------------------------------------------------------------------------------------
// calculates temperature [ C] and humidity [%RH]
// input :  humi [Ticks] (12 bit)
//          temp [Ticks] (14 bit)
// output:  humi [%RH]
//          temp [ C]
{ const float C1=-4.0;              // for 12 Bit
  const float C2= 0.0405;           // for 12 Bit
  const float C3=-0.0000028;        // for 12 Bit
  const float T1=0.01;              // for 14 Bit @ 5V
  const float T2=0.00008;           // for 14 Bit @ 5V

  float rh=*p_humidity;             // rh:      Humidity [Ticks] 12 Bit
  float t=*p_temperature;           // t:       Temperature [Ticks] 14 Bit
  float rh_lin;                     // rh_lin:  Humidity linear
  float rh_true;                    // rh_true: Temperature compensated humidity
  float t_C;                        // t_C   :  Temperature [ C]

  t_C=t*0.01 – 40;                  //calc. Temperature from ticks to [ C]
  rh_lin=C3*rh*rh + C2*rh + C1;     //calc. Humidity from ticks to [%RH]
  rh_true=(t_C-25)*(T1+T2*rh)+rh_lin;   //calc. Temperature compensated humidity [%RH]
  if(rh_true>100)rh_true=100;       //cut if the value is outside of
  if(rh_true<0.1)rh_true=0.1;       //the physical possible range

  *p_temperature=t_C;               //return temperature [ C]
  *p_humidity=rh_true;              //return humidity[%RH]
}
```

```
//--------------------------------------------------------------------
float calc_dewpoint(float h,float t)
//--------------------------------------------------------------------
// calculates dew point
// input:   humidity [%RH], temperature [ C]
// output:  dew point [ C]
{ float k,dew_point ;
  k = (log10(h)-2)/0.4343 + (17.62*t)/(243.12+t);
  dew_point = 243.12*k/(17.62-k);
  return dew_point;
}

//----------------------------------------------------------------------------------
void main()
//----------------------------------------------------------------------------------
// sample program that shows how to use SHT11 functions
// 1. connection reset
// 2. measure humidity [ticks](12 bit) and temperature [ticks](14 bit)
// 3. calculate humidity [%RH] and temperature [ C]
// 4. calculate dew point [ C]
// 5. print temperature, humidity, dew point

{ value humi_val,temp_val;
  float dew_point;
  unsigned char error,checksum;
  unsigned int i;

  init_uart();
  s_connectionreset();
  while(1)
  { error=0;
    error+=s_measure((unsigned char*) &humi_val.i,&checksum,HUMI);  //measure humidity
    error+=s_measure((unsigned char*) &temp_val.i,&checksum,TEMP);  //measure temperature
    if(error!=0) s_connectionreset();                 //in case of an error: connection reset
    else
    { humi_val.f=(float)humi_val.i;                   //converts integer to float
      temp_val.f=(float)temp_val.i;                   //converts integer to float
      calc_sth11(&humi_val.f,&temp_val.f);            //calculate humidity, temperature
      dew_point=calc_dewpoint(humi_val.f,temp_val.f); //calculate dew point
      //send final data to serial interface (UART)
      printf(“temp:%5.1fC humi:%5.1f%% dew point:%5.1fC\n”,temp_val.f,humi_val.f,dew_point);
    }
    //----------wait approx. 0.8s to avoid heating up SHTxx-------------------------------
    for (i=0;i<40000;i++);     //(be sure that the compiler doesn’t eliminate this line!)
    //------------------------------------------------------------------------------------
  }
}
```

3 Revision History

Date	Revision	Changes
November 20, 2001	0.9 (Preliminary)	Initial revision
February 19, 2001	1.00	
July 10, 2002	2.00	Added delay of 0.8s between measurements to prevent selfheating Connection reset only after error during transmission Checks for RH<0% and >100%
October 23, 2002	2.01	Changed sign of Temperature coefficient T1 to match datasheet.
Oct. 17, 2003	2.02	Changed download link
July 15, 2004	2.03	Added comments about UART function
May 25, 2005	2.04	Changed company address
April, 28, 2006	2.05	The function *calc_dewpoint* is updated with the new Magnus parameters. This improves slightly the accuracy at low humidity and low temperature.
Oct 3, 2006	2.06	Sensirion Inc. address added

The latest version of this document and all application notes can be found at:
www.sensirion.com/humidity

Headquarters and Sales Office

SENSIRION AG	Phone: + 41 (0)44 306 40 00
Laubisrütistr. 50	Fax: + 41 (0)44 306 40 30
CH-8712 Stäfa ZH	e-mail: info@sensirion.com
Switzerland	http://www.sensirion.com/
SENSIRION Inc	Phone: 805-409 4900
Westlake Pl. Ctr. I, suite 240	Fax: 805-435 0467
2801 Townsgate Road	e-mail: michael.karst@sensirion.com
Westlake Village, CA 91361	http://www.sensirion.com/
USA	