Sample Code for SHT21

Supporting Communication Software

Preface

This sample code is made to communicate with SHT2x humidity and temperature sensors. The purpose of the code is to ease the own software programming addressing SHT21 sensors. This sample code has been completed and besides the pure measurement it provides CRC checksum control, set resolution, read serial number, low battery indication and soft-reset. This sample code may be also applied on EK-H4 – and with debugging hard and software it may be modified and adapted.

1 Structure and Hierarchy of Code

The sample code is structured in various procedures. The relationship among the procedures is given in Figure 1.

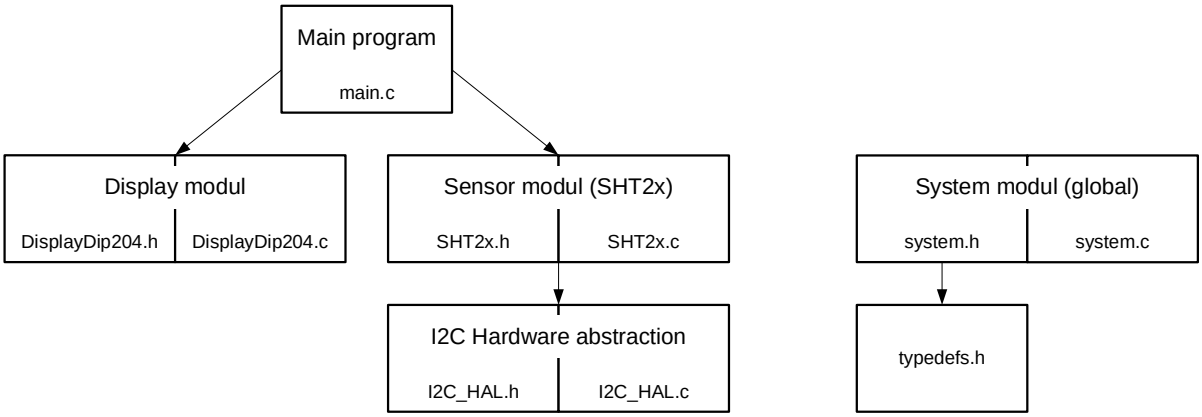

Figure 1 Structure of sample code for SHT2x.

2 Sample Code

2.1 Main.c

```
//===========================================================================
//    S E N S I R I O N   AG,  Laubisruetistr. 50, CH-8712 Staefa, Switzerland
//===========================================================================
// Project   :  SHT2x Sample Code (V1.2)
// File      :  main.c
// Author    :  MST
// Controller:  NEC V850/SG3 (uPD70F3740)
// Compiler  :  IAR compiler for V850 (3.50A)
// Brief     :  This code is an example how to implement basic commands for the
//              humidity and temperature sensor SHT2x.
//              Due to compatibility reasons the I2C interface is implemented
//              as "bit-banging" on normal I/O's. This code is written for an
//              easy understanding and is neither optimized for speed nor code
//              size.
//
// Porting to a different microcontroller (uC):
//   - define the byte-order for your uC (e.g. little endian) in typedefs.h
//   - definitions of basic types may have to be changed  in typedefs.h
//   - change the port functions / definitions for your uC    in I2C_HAL.h/.c
//   - adapt the timing of the delay function for your uC     in system.c
//   - adapt the HW_Init()                                    in system.c
//   - change the uC register definition file <io70f3740.h>   in system.h
//===========================================================================
//Revision:
//V1.1 Initial Version for SHT2x B-Samples
//V1.2 Changed calculation formula in SHT2x.c for C-Samples


//---------- Includes ------------------------------------------------------
#include "SHT2x.h"          //header file for SHT2x functions
#include "I2C_HAL.h"        //header file for I2C hardware abstraction
```

```
#include "DisplayDip204.h"  //header file for display functions
#include "System.h"         //header file for system settings
#include <stdio.h>          //header file standard input / output functions

//==============================================================================
int main()
//==============================================================================
{  // variables
  u8t  error = 0;              //variable for error code. For codes see system.h
  u8t  userRegister;           //variable for user register
  bt   endOfBattery;           //variable for end of battery

  nt16 sRH;                    //variable for raw humidity ticks
  ft   humidityRH;             //variable for relative humidity[%RH] as float
  char humititytOutStr[21];    //output string for humidity value
  nt16 sT;                     //variable for raw temperature ticks
  ft   temperatureC;           //variable for temperature[°C] as float
  char temperatureOutStr[21];  //output string for temperature value
  u8t  SerialNumber_SHT2x[8];  //64bit serial number

  Init_HW();                   //initializes Hardware (osc, watchdog,...)
  I2c_Init();                  //initializes uC-ports for I2C
  DisplayInit();               //initializes LCD
  DisplayEnableBacklight();    //enable LCD backlight
  DisplayWriteString(0,0," SHT2x Sample Code "); //write project title on LCD
  DelayMicroSeconds(15000);    //wait for sensor initialization t_powerUp (15ms)

  //note: The following code segments show how to use the different functions
  //      of SHT2x. The loop does not show a typical sequence in an application

  while(1)
  { error = 0;                                    // reset error status
    // --- Reset sensor by command ---
    error |= SHT2x_SoftReset();

    // --- Read the sensors serial number (64bit) ---
    error |= SHT2x_GetSerialNumber(SerialNumber_SHT2x);

    // --- Set Resolution e.g. RH 10bit, Temp 13bit ---
    error |= SHT2x_ReadUserRegister(&userRegister);  //get actual user reg
    userRegister = (userRegister & ~SHT2x_RES_MASK) | SHT2x_RES_10_13BIT;
    error |= SHT2x_WriteUserRegister(&userRegister); //write changed user reg

    // --- measure humidity with "Hold Master Mode (HM)"  ---
    error |= SHT2x_MeasureHM(HUMIDITY, &sRH);
    // --- measure temperature with "Polling Mode" (no hold master) ---
    error |= SHT2x_MeasurePoll(TEMP, &sT);

    //-- calculate humidity and temperature --
    temperatureC = SHT2x_CalcTemperatureC(sT.u16);
    humidityRH   = SHT2x_CalcRH(sRH.u16);

    // --- check end of battery status (eob)---
    // note: a RH / Temp. measurement must be executed to update the status of eob
    error |= SHT2x_ReadUserRegister(&userRegister);  //get actual user reg
    if( (userRegister & SHT2x_EOB_MASK) == SHT2x_EOB_ON ) endOfBattery = true;
    else endOfBattery = false;

    //-- write humidity and temperature values on LCD --
    sprintf(humititytOutStr,    "Humidity RH:%6.2f %% ",humidityRH);
    sprintf(temperatureOutStr, "Temperature:%6.2f€C",temperatureC);
    DisplayWriteString(2,0,humititytOutStr);
    DisplayWriteString(3,0,temperatureOutStr);

    //-- write error or low batt status un LCD --
    if(error != 0)
    { DisplayWriteString(1,3,"Error occurred");
      DisplayWriteString(2,0,"Humidity RH: --.-- %%");
      DisplayWriteString(3,0,"Temperature: --.--€C");
    }
    else if(endOfBattery) DisplayWriteString(1,3,"Low Batt");
    else DisplayWriteString(1,0,"                    ");

    DelayMicroSeconds(300000);     // wait 0.3s for next measurement
  }
}
```

2.2 SHT2x.h

```
#ifndef SHT2x_H
#define SHT2x_H
//==============================================================================
//    S E N S I R I O N   AG,  Laubisruetistr. 50, CH-8712 Staefa, Switzerland
//==============================================================================
// Project   :  SHT2x Sample Code (V1.2)
// File      :  SHT2x.h
// Author    :  MST
// Controller:  NEC V850/SG3 (uPD70F3740)
// Compiler  :  IAR compiler for V850 (3.50A)
// Brief     :  Sensor layer. Definitions of commands and registers,
//              functions for sensor access
//==============================================================================
//---------- Includes ----------------------------------------------------------
#include "I2C_HAL.h"
#include "system.h"
//---------- Defines -----------------------------------------------------------
//  CRC
const u16t POLYNOMIAL = 0x131;  //P(x)=x^8+x^5+x^4+1 = 100110001

// sensor command
typedef enum{
  TRIG_T_MEASUREMENT_HM    = 0xE3, // command trig. temp meas. hold master
  TRIG_RH_MEASUREMENT_HM   = 0xE5, // command trig. humidity meas. hold master
  TRIG_T_MEASUREMENT_POLL  = 0xF3, // command trig. temp meas. no hold master
  TRIG_RH_MEASUREMENT_POLL = 0xF5, // command trig. humidity meas. no hold master
  USER_REG_W               = 0xE6, // command writing user register
  USER_REG_R               = 0xE7, // command reading user register
  SOFT_RESET               = 0xFE  // command soft reset
}etSHT2xCommand;

typedef enum {
  SHT2x_RES_12_14BIT       = 0x00, // RH=12bit, T=14bit
  SHT2x_RES_8_12BIT        = 0x01, // RH= 8bit, T=12bit
  SHT2x_RES_10_13BIT       = 0x80, // RH=10bit, T=13bit
  SHT2x_RES_11_11BIT       = 0x81, // RH=11bit, T=11bit
  SHT2x_RES_MASK           = 0x81  // Mask for res. bits (7,0) in user reg.
} etSHT2xResolution;

typedef enum {
  SHT2x_EOB_ON             = 0x40, // end of battery
  SHT2x_EOB_MASK           = 0x40, // Mask for EOB bit(6) in user reg.
} etSHT2xEob;

typedef enum {
  SHT2x_HEATER_ON          = 0x04, // heater on
  SHT2x_HEATER_OFF         = 0x00, // heater off
  SHT2x_HEATER_MASK        = 0x04, // Mask for Heater bit(2) in user reg.
} etSHT2xHeater;

// measurement signal selection
typedef enum{
  HUMIDITY,
  TEMP
}etSHT2xMeasureType;

typedef enum{
  I2C_ADR_W                = 128,   // sensor I2C address + write bit
  I2C_ADR_R                = 129    // sensor I2C address + read bit
}etI2cHeader;



//==============================================================================
u8t SHT2x_CheckCrc(u8t data[], u8t nbrOfBytes, u8t checksum);
//==============================================================================
// calculates checksum for n bytes of data and compares it with expected
// checksum
// input:  data[]       checksum is built based on this data
//         nbrOfBytes   checksum is built for n bytes of data
//         checksum     expected checksum
// return: error:       CHECKSUM_ERROR = checksum does not match
//                      0              = checksum matches

//==============================================================================
u8t SHT2x_ReadUserRegister(u8t *pRegisterValue);
//==============================================================================
// reads the SHT2x user register (8bit)
// input : -
// output: *pRegisterValue
// return: error

//==============================================================================
u8t SHT2x_WriteUserRegister(u8t *pRegisterValue);
//==============================================================================
```

```
// writes the SHT2x user register (8bit)
// input : *pRegisterValue
// output: -
// return: error

//==============================================================================
u8t SHT2x_MeasurePoll(etSHT2xMeasureType eSHT2xMeasureType, nt16 *pMeasurand);
//==============================================================================
// measures humidity or temperature. This function polls every 10ms until
// measurement is ready.
// input:  eSHT2xMeasureType
// output: *pMeasurand:  humidity / temperature as raw value
// return: error
// note:   timing for timeout may be changed

//==============================================================================
u8t SHT2x_MeasureHM(etSHT2xMeasureType eSHT2xMeasureType, nt16 *pMeasurand);
//==============================================================================
// measures humidity or temperature. This function waits for a hold master until
// measurement is ready or a timeout occurred.
// input:  eSHT2xMeasureType
// output: *pMeasurand:  humidity / temperature as raw value
// return: error
// note:   timing for timeout may be changed

//==============================================================================
u8t SHT2x_SoftReset();
//==============================================================================
// performs a reset
// input:  -
// output: -
// return: error

//==============================================================================
float SHT2x_CalcRH(u16t u16sRH);
//==============================================================================
// calculates the relative humidity
// input:  sRH: humidity raw value (16bit scaled)
// return: pHumidity relative humidity [%RH]

//==============================================================================
float SHT2x_CalcTemperatureC(u16t u16sT);
//==============================================================================
// calculates temperature
// input:  sT: temperature raw value (16bit scaled)
// return: temperature [°C]

//==============================================================================
u8t SHT2x_GetSerialNumber(u8t u8SerialNumber[]);
//==============================================================================
// gets serial number of SHT2x according application note "How To
// Read-Out the Serial Number"
// note:   readout of this function is not CRC checked
//
// input:  -
// output: u8SerialNumber: Array of 8 bytes (64Bits)
//         MSB                                      LSB
//         u8SerialNumber[7]             u8SerialNumber[0]
//         SNA_1 SNA_0 SNB_3 SNB_2 SNB_1 SNB_0 SNC_1 SNC_0
// return: error
#endif
```

2.3 SHT2x.c

```
//==============================================================================
//    S E N S I R I O N   AG,  Laubisruetistr. 50, CH-8712 Staefa, Switzerland
//==============================================================================
// Project   :  SHT2x Sample Code (V1.2)
// File      :  SHT2x.c
// Author    :  MST
// Controller:  NEC V850/SG3 (uPD70F3740)
// Compiler  :  IAR compiler for V850 (3.50A)
// Brief     :  Sensor layer. Functions for sensor access
//==============================================================================

//---------- Includes ----------------------------------------------------------
#include "SHT2x.h"

//==============================================================================
u8t SHT2x_CheckCrc(u8t data[], u8t nbrOfBytes, u8t checksum)
//==============================================================================
{
  u8t crc = 0;
  u8t byteCtr;
  //calculates 8-Bit checksum with given polynomial
  for (byteCtr = 0; byteCtr < nbrOfBytes; ++byteCtr)
  { crc ^= (data[byteCtr]);
```

```
    for (u8t bit = 8; bit > 0; --bit)
    { if (crc & 0x80) crc = (crc << 1) ^ POLYNOMIAL;
      else crc = (crc << 1);
    }
  }
  if (crc != checksum) return CHECKSUM_ERROR;
  else return 0;
}

//==========================================================================
u8t SHT2x_ReadUserRegister(u8t *pRegisterValue)
//==========================================================================
{
  u8t checksum;   //variable for checksum byte
  u8t error=0;    //variable for error code

  I2c_StartCondition();
  error |= I2c_WriteByte (I2C_ADR_W);
  error |= I2c_WriteByte (USER_REG_R);
  I2c_StartCondition();
  error |= I2c_WriteByte (I2C_ADR_R);
  *pRegisterValue = I2c_ReadByte(ACK);
  checksum=I2c_ReadByte(NO_ACK);
  error |= SHT2x_CheckCrc (pRegisterValue,1,checksum);
  I2c_StopCondition();
  return error;
}

//==========================================================================
u8t SHT2x_WriteUserRegister(u8t *pRegisterValue)
//==========================================================================
{
  u8t error=0;   //variable for error code

  I2c_StartCondition();
  error |= I2c_WriteByte (I2C_ADR_W);
  error |= I2c_WriteByte (USER_REG_W);
  error |= I2c_WriteByte (*pRegisterValue);
  I2c_StopCondition();
  return error;
}

//==========================================================================
u8t SHT2x_MeasureHM(etSHT2xMeasureType eSHT2xMeasureType, nt16 *pMeasurand)
//==========================================================================
{
  u8t  checksum;   //checksum
  u8t  data[2];    //data array for checksum verification
  u8t  error=0;    //error variable
  u16t i;          //counting variable

  //-- write I2C sensor address and command --
  I2c_StartCondition();
  error |= I2c_WriteByte (I2C_ADR_W); // I2C Adr
  switch(eSHT2xMeasureType)
  { case HUMIDITY: error |= I2c_WriteByte (TRIG_RH_MEASUREMENT_HM); break;
    case TEMP    : error |= I2c_WriteByte (TRIG_T_MEASUREMENT_HM);  break;
    default: assert(0);
  }
  //-- wait until hold master is released --
  I2c_StartCondition();
  error |= I2c_WriteByte (I2C_ADR_R);
  SCL=HIGH;                     // set SCL I/O port as input
  for(i=0; i<1000; i++)         // wait until master hold is released or
  { DelayMicroSeconds(1000);    // a timeout (~1s) is reached
    if (SCL_CONF==1) break;
  }
  //-- check for timeout --
  if(SCL_CONF==0) error |= TIME_OUT_ERROR;

  //-- read two data bytes and one checksum byte --
  pMeasurand->s16.u8H = data[0] = I2c_ReadByte(ACK);
  pMeasurand->s16.u8L = data[1] = I2c_ReadByte(ACK);
  checksum=I2c_ReadByte(NO_ACK);

  //-- verify checksum --
  error |= SHT2x_CheckCrc (data,2,checksum);
  I2c_StopCondition();
  return error;
}

//==========================================================================
u8t SHT2x_MeasurePoll(etSHT2xMeasureType eSHT2xMeasureType, nt16 *pMeasurand)
//==========================================================================
{
  u8t  checksum;   //checksum
  u8t  data[2];    //data array for checksum verification
  u8t  error=0;    //error variable
  u16t i=0;        //counting variable
```

```
  //-- write I2C sensor address and command --
  I2c_StartCondition();
  error |= I2c_WriteByte (I2C_ADR_W); // I2C Adr
  switch(eSHT2xMeasureType)
  { case HUMIDITY: error |= I2c_WriteByte (TRIG_RH_MEASUREMENT_POLL); break;
    case TEMP    : error |= I2c_WriteByte (TRIG_T_MEASUREMENT_POLL);  break;
    default: assert(0);
  }
  //-- poll every 10ms for measurement ready. Timeout after 20 retries (200ms)--
  do
  { I2c_StartCondition();
    DelayMicroSeconds(10000);  //delay 10ms
    if(i++ >= 20) break;
  } while(I2c_WriteByte (I2C_ADR_R) == ACK_ERROR);
  if (i>=20) error |= TIME_OUT_ERROR;

  //-- read two data bytes and one checksum byte --
  pMeasurand->s16.u8H = data[0] = I2c_ReadByte(ACK);
  pMeasurand->s16.u8L = data[1] = I2c_ReadByte(ACK);
  checksum=I2c_ReadByte(NO_ACK);

  //-- verify checksum --
  error |= SHT2x_CheckCrc (data,2,checksum);
  I2c_StopCondition();

  return error;
}

//==========================================================================
u8t SHT2x_SoftReset()
//==========================================================================
{
  u8t  error=0;          //error variable

  I2c_StartCondition();
  error |= I2c_WriteByte (I2C_ADR_W); // I2C Adr
  error |= I2c_WriteByte (SOFT_RESET);                          // Command
  I2c_StopCondition();

  DelayMicroSeconds(15000); // wait till sensor has restarted

  return error;
}

//==========================================================================
float SHT2x_CalcRH(u16t u16sRH)
//==========================================================================
{
  ft humidityRH;             // variable for result

  u16sRH &= ~0x0003;         // clear bits [1..0] (status bits)
  //-- calculate relative humidity [%RH] --

  humidityRH = -6.0 + 125.0/65536 * (ft)u16sRH; // RH= -6 + 125 * SRH/2^16
  return humidityRH;
}

//==========================================================================
float SHT2x_CalcTemperatureC(u16t u16sT)
//==========================================================================
{
  ft temperatureC;           // variable for result

  u16sT &= ~0x0003;          // clear bits [1..0] (status bits)

  //-- calculate temperature [°C] --
  temperatureC= -46.85 + 175.72/65536 *(ft)u16sT; //T= -46.85 + 175.72 * ST/2^16
  return temperatureC;
}

//==========================================================================
u8t SHT2x_GetSerialNumber(u8t u8SerialNumber[])
//==========================================================================
{
  u8t  error=0;                          //error variable

  //Read from memory location 1
  I2c_StartCondition();
  error |= I2c_WriteByte (I2C_ADR_W);    //I2C address
  error |= I2c_WriteByte (0xFA);         //Command for readout on-chip memory
  error |= I2c_WriteByte (0x0F);         //on-chip memory address
  I2c_StartCondition();
  error |= I2c_WriteByte (I2C_ADR_R);    //I2C address
  u8SerialNumber[5] = I2c_ReadByte(ACK); //Read SNB_3
  I2c_ReadByte(ACK);                     //Read CRC SNB_3 (CRC is not analyzed)
  u8SerialNumber[4] = I2c_ReadByte(ACK); //Read SNB_2
  I2c_ReadByte(ACK);                     //Read CRC SNB_2 (CRC is not analyzed)
  u8SerialNumber[3] = I2c_ReadByte(ACK); //Read SNB_1
```

```
  I2c_ReadByte(ACK);                    //Read CRC SNB_1 (CRC is not analyzed)
  u8SerialNumber[2] = I2c_ReadByte(ACK); //Read SNB_0
  I2c_ReadByte(NO_ACK);                 //Read CRC SNB_0 (CRC is not analyzed)
  I2c_StopCondition();

  //Read from memory location 2
  I2c_StartCondition();
  error |= I2c_WriteByte (I2C_ADR_W);   //I2C address
  error |= I2c_WriteByte (0xFC);        //Command for readout on-chip memory
  error |= I2c_WriteByte (0xC9);        //on-chip memory address
  I2c_StartCondition();
  error |= I2c_WriteByte (I2C_ADR_R);   //I2C address
  u8SerialNumber[1] = I2c_ReadByte(ACK); //Read SNC_1
  u8SerialNumber[0] = I2c_ReadByte(ACK); //Read SNC_0
  I2c_ReadByte(ACK);                    //Read CRC SNC0/1 (CRC is not analyzed)
  u8SerialNumber[7] = I2c_ReadByte(ACK); //Read SNA_1
  u8SerialNumber[6] = I2c_ReadByte(ACK); //Read SNA_0
  I2c_ReadByte(NO_ACK);                 //Read CRC SNA0/1 (CRC is not analyzed)
  I2c_StopCondition();

  return error;
}
```

2.4 I2C_HAL.h

```
#ifndef I2C_HAL_H
#define I2C_HAL_H
//==============================================================================
//    S E N S I R I O N   AG,  Laubisruetistr. 50, CH-8712 Staefa, Switzerland
//==============================================================================
// Project   :  SHT2x Sample Code (V1.2)
// File      :  I2C_HAL.h
// Author    :  MST
// Controller:  NEC V850/SG3 (uPD70F3740)
// Compiler  :  IAR compiler for V850 (3.50A)
// Brief     :  I2C Hardware abstraction layer
//==============================================================================
//---------- Includes ----------------------------------------------------------
#include "system.h"
//---------- Defines -----------------------------------------------------------
//I2C ports
//The communication on SDA and SCL is done by switching pad direction
//For a low level on SCL or SDA, direction is set to output. For a high level on
//SCL or SDA, direction is set to input. (pull up resistor active)
#define SDA      PM3H_bit.no0 //SDA on I/O P38 defines direction (input=1/output=0)
#define SDA_CONF P3H_bit.no0  //SDA level on output direction
#define SCL      PM3H_bit.no1 //SCL on I/O P39 defines direction (input=1/output=0)
#define SCL_CONF P3H_bit.no1  //SCL level on output direction
//---------- Enumerations ------------------------------------------------------
//  I2C level
typedef enum{
  LOW                   = 0,
  HIGH                  = 1,
}etI2cLevel;

// I2C acknowledge
typedef enum{
  ACK                   = 0,
  NO_ACK                = 1,
}etI2cAck;

//==============================================================================
void I2c_Init ();
//==============================================================================
//Initializes the ports for I2C interface

//==============================================================================
void I2c_StartCondition ();
//==============================================================================
// writes a start condition on I2C-bus
// input : -
// output: -
// return: -
// note  : timing (delay) may have to be changed for different microcontroller
//         _____
// SDA:       |_____
//         _______
// SCL :         |___

//==============================================================================
void I2c_StopCondition ();
//==============================================================================
// writes a stop condition on I2C-bus
// input : -
// output: -
// return: -
// note  : timing (delay) may have to be changed for different microcontroller
//               _____
```

```
// SDA:   _____|
//              _______
// SCL :  ___|

//===========================================================================
u8t I2c_WriteByte (u8t txByte);
//===========================================================================
// writes a byte to I2C-bus and checks acknowledge
// input:  txByte  transmit byte
// output: -
// return: error
// note: timing (delay) may have to be changed for different microcontroller

//===========================================================================
u8t I2c_ReadByte (etI2cAck ack);
//===========================================================================
// reads a byte on I2C-bus
// input:  rxByte  receive byte
// output: rxByte
// note: timing (delay) may have to be changed for different microcontroller
#endif
```

2.5 I2C_HAL.c

```
//==============================================================================
//    S E N S I R I O N   AG,  Laubisruetistr. 50, CH-8712 Staefa, Switzerland
//==============================================================================
// Project   :  SHT2x Sample Code (V1.2)
// File      :  I2C_HAL.c
// Author    :  MST
// Controller:  NEC V850/SG3 (uPD70F3740)
// Compiler  :  IAR compiler for V850 (3.50A)
// Brief     :  I2C Hardware abstraction layer
//==============================================================================
//---------- Includes ----------------------------------------------------------
#include "I2C_HAL.h"
//==============================================================================
void I2c_Init ()
//==============================================================================
{
  SDA=LOW;                // Set port as output for configuration
  SCL=LOW;                // Set port as output for configuration

  SDA_CONF=LOW;           // Set SDA level as low for output mode
  SCL_CONF=LOW;           // Set SCL level as low for output mode

  SDA=HIGH;               // I2C-bus idle mode SDA released (input)
  SCL=HIGH;               // I2C-bus idle mode SCL released (input)
}

//==============================================================================
void I2c_StartCondition ()
//==============================================================================
{
  SDA=HIGH;
  SCL=HIGH;
  SDA=LOW;
  DelayMicroSeconds(10);  // hold time start condition (t_HD;STA)
  SCL=LOW;
  DelayMicroSeconds(10);
}

//==============================================================================
void I2c_StopCondition ()
//==============================================================================
{
  SDA=LOW;
  SCL=LOW;
  SCL=HIGH;
  DelayMicroSeconds(10);  // set-up time stop condition (t_SU;STO)
  SDA=HIGH;
  DelayMicroSeconds(10);
}

//==============================================================================
u8t I2c_WriteByte (u8t txByte)
//==============================================================================
{
  u8t mask,error=0;
  for (mask=0x80; mask>0; mask>>=1)   //shift bit for masking (8 times)
  { if ((mask & txByte) == 0) SDA=LOW;//masking txByte, write bit to SDA-Line
    else SDA=HIGH;
    DelayMicroSeconds(1);             //data set-up time (t_SU;DAT)
    SCL=HIGH;                         //generate clock pulse on SCL
    DelayMicroSeconds(5);             //SCL high time (t_HIGH)
    SCL=LOW;
    DelayMicroSeconds(1);             //data hold time(t_HD;DAT)
  }
```

```
  SDA=HIGH;                           //release SDA-line
  SCL=HIGH;                           //clk #9 for ack
  DelayMicroSeconds(1);               //data set-up time (t_SU;DAT)
  if(SDA_CONF==HIGH) error=ACK_ERROR; //check ack from i2c slave
  SCL=LOW;
  DelayMicroSeconds(20);              //wait time to see byte package on scope
  return error;                       //return error code
}

//==============================================================================
u8t I2c_ReadByte (etI2cAck ack)
//==============================================================================
{
  u8t mask,rxByte=0;
  SDA=HIGH;                           //release SDA-line
  for (mask=0x80; mask>0; mask>>=1)   //shift bit for masking (8 times)
  { SCL=HIGH;                         //start clock on SCL-line
    DelayMicroSeconds(1);             //data set-up time (t_SU;DAT)
    DelayMicroSeconds(3);             //SCL high time (t_HIGH)
    if (SDA_CONF==1) rxByte=(rxByte | mask); //read bit
    SCL=LOW;
    DelayMicroSeconds(1);             //data hold time(t_HD;DAT)
  }
  SDA=ack;                            //send acknowledge if necessary
  DelayMicroSeconds(1);               //data set-up time (t_SU;DAT)
  SCL=HIGH;                           //clk #9 for ack
  DelayMicroSeconds(5);               //SCL high time (t_HIGH)
  SCL=LOW;
  SDA=HIGH;                           //release SDA-line
  DelayMicroSeconds(20);              //wait time to see byte package on scope
  return rxByte;                      //return error code
}
```

2.6 DisplayDip204.h

```
#ifndef DISPLAYDIP204_H
#define DISPLAYDIP204_H
//==============================================================================
//    S E N S I R I O N   AG,  Laubisruetistr. 50, CH-8712 Staefa, Switzerland
//==============================================================================
// Project   :  SHT2x Sample Code (V1.2)
// File      :  DisplayDip204.h
// Author    :  SWE
// Controller:  NEC V850/SG3 (uPD70F3740)
// Compiler  :  IAR compiler for V850 (3.50A)
// Brief     :  Display Interface (DIP204)
//==============================================================================
//---------- Includes ----------------------------------------------------------
#include "system.h"
//---------- Defines -----------------------------------------------------------
// Define used pins
#define DISPLAY_RS_PIN             P9H_bit.no2
#define DISPLAY_RS_PIN_MODE        PM9H_bit.no2

#define DISPLAY_nRES_PIN           P9H_bit.no3
#define DISPLAY_nRES_PIN_MODE      PM9H_bit.no3

#define DISPLAY_RW_PIN             P9H_bit.no1
#define DISPLAY_RW_PIN_MODE        PM9H_bit.no1

#define DISPLAY_E_PIN              P9H_bit.no0
#define DISPLAY_E_PIN_MODE         PM9H_bit.no0

#define DISPLAY_BACKLIGHT_PIN      P3L_bit.no3
#define DISPLAY_BACKLIGHT_PIN_MODE PM3L_bit.no3

// Define used port
#define DISPLAY_PORT               P9L
#define DISPLAY_PORT_MODE          PM9L

//==============================================================================
void DisplayInit(void);
//==============================================================================
// Initializes display.
//==============================================================================
void DisplayEnableBacklight(void);
//==============================================================================
// Backlight enable

//==============================================================================
void DisplayDisableBacklight(void);
//==============================================================================
// Backlight disable
//==============================================================================
void DisplayWriteString(unsigned char aLine, unsigned char aColumn, char* aStringToWrite);
//==============================================================================
// Writes a character string to LCD on a specific position. If the string size
// exeeds the character size on the display, the surplus character are cut.
```

```
// input : aLine          line number [0...3]
//         aRow           start position of string in line [0...19]
//         aCharToWrite   output string
// output: -
// return: -

//===========================================================================
void DisplayWriteChar(unsigned char aLine, unsigned char aColumn, char aCharToWrite);
//===========================================================================
// Writes one character to LCD on a specific position.
// input : aLine          line number [0...3]
//         aRow           position of printing character line [0...19]
//         aCharToWrite   output character
// output: -
// return: -

//===========================================================================
void DisplayClear(void);
//===========================================================================
// Clears Display

//===========================================================================
void DisplayPrivateWriteByte(unsigned char aRs, unsigned char aCode);
//===========================================================================
// Internal function for writing a byte

//===========================================================================
void DisplayPrivateSetCursor(unsigned char aLine, unsigned char aColumn);
//===========================================================================
//Internal function to set the cursor

//===========================================================================
char DisplayPrivateConvertChar(char aChar);
//===========================================================================
// Internal function to convert a character (ASCII -> Displaycode)
#endif
```

2.7 DisplayDip204.c

```
//===========================================================================
//    S E N S I R I O N   AG,  Laubisruetistr. 50, CH-8712 Staefa, Switzerland
//===========================================================================
// Project   :  SHT2x Sample Code (V1.2)
// File      :  DisplayDip204.c
// Author    :  SWE
// Controller:  NEC V850/SG3 (uPD70F3740)
// Compiler  :  IAR compiler for V850 (3.50A)
// Brief     :  Display Interface (DIP204)
//===========================================================================
//---------- Includes -------------------------------------------------------
#include "DisplayDip204.h"
//===========================================================================
void DisplayInit(void)
//===========================================================================
{
  // Defines port direction
  DISPLAY_RS_PIN_MODE        = 0;     // output
  DISPLAY_nRES_PIN_MODE      = 0;     // output
  DISPLAY_RW_PIN_MODE        = 0;     // output
  DISPLAY_E_PIN_MODE         = 0;     // output
  DISPLAY_PORT_MODE          = 0x00;  // output
  DISPLAY_BACKLIGHT_PIN_MODE = 0;     // output

  // Defines initial port state
  DISPLAY_RW_PIN   = 1;
  DISPLAY_E_PIN    = 1;
  DISPLAY_nRES_PIN = 1;      // reset inactive
  DISPLAY_BACKLIGHT_PIN = 0; // backlight off

  // Power up time display
  DelayMicroSeconds(10000);

  // Initialize display
  DisplayPrivateWriteByte(0, 0x34);  // 8 bit data length, extension bit RE = 1
  DelayMicroSeconds(50);
  DisplayPrivateWriteByte(0, 0x09);  // 4 line mode
  DelayMicroSeconds(50);
  DisplayPrivateWriteByte(0, 0x30);  // 8 bit data length, extension bit RE = 0
  DelayMicroSeconds(50);
  DisplayPrivateWriteByte(0, 0x0C);  // display on, cursor off, blink off
  DelayMicroSeconds(50);
  DisplayPrivateWriteByte(0, 0x01);  // clear display, cursor 1st. row, 1st. line
  DelayMicroSeconds(2000);
  DisplayPrivateWriteByte(0, 0x06);  // cursor will be automatically incremented
  DelayMicroSeconds(50);
}
//===========================================================================
void DisplayEnableBacklight(void)
```

```
//===========================================================================
{
  DISPLAY_BACKLIGHT_PIN = 1;
}

//===========================================================================
void DisplayDisableBacklight(void)
//===========================================================================
{
  DISPLAY_BACKLIGHT_PIN = 0;
}
//===========================================================================
void DisplayWriteString(unsigned char aLine, unsigned char aColumn, char* aStringToWrite)
//===========================================================================
{
  // set cursor
  DisplayPrivateSetCursor(aLine, aColumn);
  // write character
  for(int i=aColumn; i<20; i++) // start at given position in line (row)
  {
    if(aStringToWrite[i-aColumn] == 0x00) break; // if NUL character -> exit
    else
    {
      DisplayPrivateWriteByte(1, DisplayPrivateConvertChar(aStringToWrite[i-aColumn]));
      DelayMicroSeconds(50);
    }
  }
}

//===========================================================================
void DisplayWriteChar(unsigned char aLine, unsigned char aColumn, char aCharToWrite)
//===========================================================================
{
  // set cursor
  DisplayPrivateSetCursor(aLine, aColumn);
  // write character
  DisplayPrivateWriteByte(1, DisplayPrivateConvertChar(aCharToWrite));
  DelayMicroSeconds(50);
}

//===========================================================================
void DisplayClear(void)
//===========================================================================
{
  DisplayPrivateWriteByte(0, 0x01);  // clear display, cursor 1st. row, 1st. line
  DelayMicroSeconds(2000);
}

//===========================================================================
void DisplayPrivateWriteByte(unsigned char aRs, unsigned char aCode)
//===========================================================================
{
  // set Register Select (RS)
  DISPLAY_RS_PIN = aRs;
  // set R/W  to write
  DISPLAY_RW_PIN = 0;
  // set data on bus
  DISPLAY_PORT = aCode;
  // enable to low
  DISPLAY_E_PIN = 0;
  // wait minimal low time
  DelayMicroSeconds(1);
  // enable to high
  DISPLAY_E_PIN = 1;
}

//===========================================================================
void DisplayPrivateSetCursor(unsigned char aLine, unsigned char aColumn)
//===========================================================================
{
  // Line number must be between 0..3
  assert(aLine < 4);
  // Row number must be between 0..19
  assert(aColumn < 20);

  DisplayPrivateWriteByte(0, 0x80 | (aLine * 0x20 + aColumn)); // Set DDRAM Adr.
  DelayMicroSeconds(2000);
}

//===========================================================================
char DisplayPrivateConvertChar(char aChar)
//===========================================================================
{ return aChar;
}
```

2.8 System.h

```
#ifndef SYSTEM_H
#define SYSTEM_H
```

```
//===========================================================================
//    S E N S I R I O N   AG,  Laubisruetistr. 50, CH-8712 Staefa, Switzerland
//===========================================================================
// Project   :  SHT2x Sample Code (V1.2)
// File      :  System.h
// Author    :  MST
// Controller:  NEC V850/SG3 (uPD70F3740)
// Compiler  :  IAR compiler for V850 (3.50A)
// Brief     :  System functions, global definitions
//===========================================================================
//---------- Includes -------------------------------------------------------
#include "io70f3740.h"             // controller register definitions
#include <assert.h>                // assert functions
#include <intrinsics.h>            // low level microcontroller commands
#include "typedefs.h"              // type definitions
//---------- Enumerations ---------------------------------------------------
// Error codes
typedef enum{
  ACK_ERROR               = 0x01,
  TIME_OUT_ERROR          = 0x02,
  CHECKSUM_ERROR          = 0x04,
  UNIT_ERROR              = 0x08
}etError;

//===========================================================================
void Init_HW (void);
//===========================================================================
// Initializes the used hardware

//===========================================================================
void DelayMicroSeconds (u32t nbrOfUs);
//===========================================================================
// wait function for small delays
// input:  nbrOfUs   wait x times approx. one micro second (fcpu = 4MHz)
// return: -
// note: smallest delay is approx. 30us due to function call
#endif
```

2.9 System.c

```
//===========================================================================
//    S E N S I R I O N   AG,  Laubisruetistr. 50, CH-8712 Staefa, Switzerland
//===========================================================================
// Project   :  SHT2x Sample Code (V1.2)
// File      :  System.c
// Author    :  MST
// Controller:  NEC V850/SG3 (uPD70F3740)
// Compiler  :  IAR compiler for V850 (3.50A)
// Brief     :  System functions
//===========================================================================
//---------- Includes -------------------------------------------------------
#include "system.h"
//===========================================================================
void Init_HW (void)
//===========================================================================
{
  //-- initialize system clock of V850 (fcpu = fosc, no PLL) --
  PRCMD = 0x00;   // unlock PCC register
  PCC = 0x00;     // perform settings in PCC register
  RCM = 0x01;     // disable ring oscillator
  //-- watchdog --
  WDTM2 = 0x0f;   // stop watchdog
  //-- interrupts --
  __EI();         // enable interrupts for debugging with minicube

  //Settings for debugging with Sensirion EKH4 and minicube2, power up sensor
  //Not needed for normal use
  PMDLL = 0xF0;
  PDLL  = 0x04;
}
//===========================================================================
#pragma optimize = s none
void DelayMicroSeconds (u32t nbrOfUs)
//===========================================================================
{
  for(u32t i=0; i<nbrOfUs; i++)
  {  //__asm("nop");  //nop's may be added for timing adjustment
  }
}
```

2.10 Typedefs.h

```
#ifndef TYPEDEFS_H
#define TYPEDEFS_H
```

```
//==============================================================================
//    S E N S I R I O N   AG,  Laubisruetistr. 50, CH-8712 Staefa, Switzerland
//==============================================================================
// Project   :  SHT2x Sample Code (V1.2)
// File      :  typedefs.h
// Author    :  MST
// Controller:  NEC V850/SG3 (uPD70F3740)
// Compiler  :  IAR compiler for V850 (3.50A)
// Brief     :  Definitions of typedefs for good readability and portability
//==============================================================================

//---------- Defines -----------------------------------------------------------
//Processor endian system
//#define BIG ENDIAN   //e.g. Motorola (not tested at this time)
#define LITTLE_ENDIAN  //e.g. PIC, 8051, NEC V850
//==============================================================================
// basic types: making the size of types clear
//==============================================================================
typedef unsigned char   u8t;      ///< range: 0 .. 255
typedef signed char     i8t;      ///< range: -128 .. +127

typedef unsigned short  u16t;     ///< range: 0 .. 65535
typedef signed short    i16t;     ///< range: -32768 .. +32767

typedef unsigned long   u32t;     ///< range: 0 .. 4'294'967'295
typedef signed long     i32t;     ///< range: -2'147'483'648 .. +2'147'483'647

typedef float           ft;       ///< range: +-1.18E-38 .. +-3.39E+38
typedef double          dt;      ///< range:            .. +-1.79E+308

typedef bool            bt;       ///< values: 0, 1 (real bool used)

typedef union {
  u16t u16;               // element specifier for accessing whole u16
  i16t i16;               // element specifier for accessing whole i16
  struct {
    #ifdef LITTLE_ENDIAN  // Byte-order is little endian
    u8t u8L;              // element specifier for accessing low u8
    u8t u8H;              // element specifier for accessing high u8
    #else                 // Byte-order is big endian
    u8t u8H;              // element specifier for accessing low u8
    u8t u8L;              // element specifier for accessing high u8
    #endif
  } s16;                  // element spec. for acc. struct with low or high u8
} nt16;

typedef union {
  u32t u32;               // element specifier for accessing whole u32
  i32t i32;               // element specifier for accessing whole i32
 struct {
    #ifdef LITTLE_ENDIAN  // Byte-order is little endian
    u16t u16L;            // element specifier for accessing low u16
    u16t u16H;            // element specifier for accessing high u16
    #else                 // Byte-order is big endian
    u16t u16H;            // element specifier for accessing low u16
    u16t u16L;            // element specifier for accessing high u16
    #endif
  } s32;                  // element spec. for acc. struct with low or high u16
} nt32;
#endif
```

Revision History

Date	Version	Page(s)	Changes
17 August 2009	1.0	1 – 11	Initial release
27 October 2009	1.1	1 – 12	Complete rework. Features added (CRC, Set Resolution, Low Bat Indication, Get Serial Number, Softreset)
26 February 2010	1.2	1 – 14	Adaption of C-sample coefficients (datasheet SHT21, version 1.0)

Headquarter and Sales Offices

Headquarter
SENSIRION AG
Laubisruetistr. 50
CH-8712 Staefa ZH
Switzerland
Phone: +41 44 306 40 00
Fax: +41 44 306 40 30
info@sensirion.com
http://www.sensirion.com/

Sales Office USA:
SENSIRION Inc.
2801 Townsgate Rd., Suite 204
Westlake Village, CA 91361
USA
Phone: +1 805 409 4900
Fax: +1 805 435 0467
michael.karst@sensirion.com
http://www.sensirion.com/

Sales Office Japan:
SENSIRION JAPAN Co. Ltd.
Postal Code: 108-0074
Shinagawa Station Bldg. 7F,
4-23-5, Takanawa, Minato-ku
Tokyo, Japan
Phone: +81 3 3444 4940
Fax: +81 3 3444 4939
info@sensirion.co.jp
http://www.sensirion.co.jp

Sales Office Korea:
SENSIRION KOREA Co. Ltd.
#1414, Anyang Construction Tower B/D,
1112-1, Bisan-dong, Anyang-city
Gyeonggi-Province
South Korea
Phone: +82 31 440 9925~27
Fax: +82 31 440 9927
info@sensirion.co.kr
http://www.sensirion.co.kr

Sales Office China:
Sensirion China Co. Ltd.
Room 2411, Main Tower
Jin Zhong Huan Business Building,
Futian District, Shenzhen,
Postal Code 518048
PR China
phone: +86 755 8252 1501
fax: +86 755 8252 1580
info@sensirion.com.cn
www.sensirion.com.cn

Find your local representative at: http://www.sensirion.com/reps