```
/*
 * humidity sensor.c
 *
 * Created: 13/03/2014 10:25:39
 * Author: SpinMos
  * www.spinmos.com
  ================================================================================
  * Copyright (c) 2013, SpinMos company
  ================================================================================
  *
  * Description
             Reads humidity value from HIH 6130 sensor
             TWI (I2C) protocol communications
                    C0 = SDA
                    C1 = SCL

             The relative humidity is stored at "humidity" float

  * This program uses SpinMos' TWI functions.

  * Works at 32Mhz internal clock
  */


#define F_CPU 32000000UL
#define __AVR_ATxmega128A3U
#include <avr/io.h>
#include "util/delay.h"
#include "math.h"

typedef unsigned int uint16;

// Global functions
void clock32M();
// TWI functions
       void TWI_error();
       void TWI_config();
       void TWI_sendcommand();
       void TWI_readD();
       void TWI_repstart();
       void TWI_stop();
       void TWI_nackstop();
       void TWI_ack();
       void TWI_idle();

#define TWI_R 1
#define TWI_W 0

int SLA_ADDR;
int hum[4];
int numbytes;
uint16 Hbyte;
float humidity;


// ------------------------------------ Functions
// Sets clock at 32Mhz
void clock32M()
```

```
{
	OSC.CTRL|=OSC_RC32MEN_bm;
	while (!(OSC.STATUS & OSC_RC32MRDY_bm));
	CCP=CCP_IOREG_gc;
	CLK.CTRL=CLK_SCLKSEL_RC32M_gc;
}

// TWI configuration
void TWI_config()
{
	// Clock SCL = 100kHz with 32Mhz internal oscillator
	TWIC_MASTER_BAUD =  0x9B;
	// Enable TWI Master
	TWIC_MASTER_CTRLA = TWI_MASTER_ENABLE_bm;
	// Force bus state into idle
	TWIC_MASTER_STATUS = TWI_MASTER_BUSSTATE_IDLE_gc;
}

// Send command
void TWI_sendcommand(int SLA_ADDR, int RW)
{
	// Slave address + RW bit
	TWIC_MASTER_ADDR = ((SLA_ADDR << 1) | RW);

	// Wait for byte transmitted WIF flag = 1, AND
	// acknowledged flag received, RxACK = 1
	while((!(TWIC_MASTER_STATUS & TWI_MASTER_WIF_bm)) && (!(TWIC_MASTER_STATUS &
TWI_MASTER_RXACK_bm))){}
}

// Send a repeated start
void TWI_repstart(int SLA_ADDR, int RW)
{
	TWIC_MASTER_CTRLC = TWI_MASTER_CMD_REPSTART_gc;
	TWIC_MASTER_ADDR = (SLA_ADDR | RW);
}

// Read data
void TWI_readdata(int numRBytes, int *adre)
{
	int i;

	for(i=0;i<numRBytes;i++)
	{
		// Wait for byte received, RIF flag
		while(!(TWIC_MASTER_STATUS & TWI_MASTER_RIF_bm)){}
		*adre = TWIC_MASTER_DATA;
		adre++;
		if (i<(numRBytes-1))
		{
			TWI_ack();
		}
		else
		{
			TWI_nackstop();
		}
	}
}
```

```
// Executes acknowledge action succeeded by a byte receive
void TWI_ack()
{
	TWIC_MASTER_CTRLC = TWI_MASTER_CMD_RECVTRANS_gc;
}

// Issues an stop condition
void TWI_stop()
{
	// Writes stop condition
	TWIC_MASTER_CTRLC = TWI_MASTER_CMD_STOP_gc;
}

// Issues NACK + stop condition
void TWI_nackstop()
{
	TWIC_MASTER_CTRLC = TWI_MASTER_ACKACT_bm | TWI_MASTER_CMD_STOP_gc;
}

// TWI error, stops the TWI interface
void TWI_error(){
	TWIC_MASTER_CTRLC = TWI_MASTER_CMD_STOP_gc;
}

// TWI idle state
void TWI_idle()
{
	while (!(TWIC_MASTER_STATUS & TWI_MASTER_BUSSTATE_IDLE_gc)){}
}

// ----------------------------------------- MAIN PROGRAM
int main(void)
{
	// Defines 32Mhz internal clock
	clock32M();

	// TWI configuration
	TWI_config();
	// Forces TWI into idle state
	TWI_idle();

	while(1)
	{
		// send command
		// TWI measurement request (slave ADDR = 0x27)
		SLA_ADDR = 0x27;
		TWI_sendcommand((SLA_ADDR << 1), TWI_W);
		// Stop command
		TWI_stop();

		_delay_us(200);

		TWI_repstart((SLA_ADDR << 1), TWI_R);
		TWI_readdata(4, &hum[0]);
		Hbyte = ((hum[0] & 0x3F) << 8) | (hum[1]);
		// humidity % RH
		humidity = (Hbyte / 16383.0) * 100.0;
```

```
                _delay_ms(200);
        }
}
```