HT-01XX Code

ATmega32 Code(AVR studio)

- Humidity & Temperature Measurement Data Read Code

```
void Measure_TH(unsigned char Device_Address)
{
        // HT01S Measurement Mode
        // 1. Update Mode: The HT01S will continuously measure without Measurement
        //                 Request command (MR)
        // 2. Sleep Mode: The HT01S will only perform conversions when The HT01S receives
        //                 a Measurement Request command (MR)
        // the user will not be able to change Measurement Mode.

        // Measurement Request (MR) START ----------------------------------------
        i2c_start();
        write_i2c_byte(Device_Address);
        i2c_stop();
        Delay_100ms(1);                 // Measurement waiting Time
        // MR END ----------------------------------------------------------------

        //Data Fetch
        i2c_start();
        write_i2c_byte(Device_Address | 0x01);
        DATA[0]=read_i2c_byte(0);
        DATA[1]=read_i2c_byte(0);
        DATA[2]=read_i2c_byte(0);
        DATA[3]=read_i2c_byte(1);
        i2c_stop();
        ucNack=0;
}
```

- I2C Start, Stop, Send, Receive, ACK

```
void i2c_start(void)
{
        DDRC |= 0x02;                   // Set SDA to output
        PORTC = 0x03;                   // Set SCL, SDA High
        Delay_us(8);

        PORTC &= 0xFD;                  // Clear SDA Low
        Delay_us(9);
        PORTC &= 0xFE;                  // Clear SCL Low
        Delay_us(7);
}

void i2c_stop(void)
{
        DDRC |= 0x02;                   // Set SDA to output

        PORTC &= 0xFD;                  // Clear SDA Low
        PORTC &= 0xFE;                  // Clear SCL Low
        Delay_us(9);

        PORTC |= 0x01;                  // Set SCL High
        Delay_us(7);
        PORTC |= 0x02;                  // Set SDA High
}
```

```
unsigned char i2c_ackn(void)
{
        unsigned char ackn = 0;                 // Temp RAM for Ackn flag

        PORTC &= 0xFE;                          // Clear SCL Low
        DDRC &= 0xFD;                           // Set SDA to input
        PORTC |= 0x01;                          // Clock the ACK bit
        nops(5);

        if((PINC & 0x02) == 0x02)
                ackn=1;                         // Check the ACK bit on SDA
        else
                ackn =0;

        nops(7);
        PORTC &= 0xFE;                          // Clear SCL Low
        DDRC |= 0x02;                           `// SDA Output Mode
        PORTC &= 0xFD;                          // Clear SDA Low

        return ackn;                            // Return our ACK bit
}

void write_i2c_byte(unsigned char byte)
{
        unsigned char i;

        DDRC |= 0x02;                           // Set SDA to output
        PORTC &= 0xFE;                          // Clear SCL Low
        nop();

        for (i=0; i<8 ; i++)
        {
                if(byte & 0x80){
                        PORTC |= 0x02;          // Set SDA High
                }
                else{
                        PORTC &= 0xFD;          // Clear SDA Low
                }
                nops(10);
                PORTC |= 0x01;                  // Set SCL High, Clock data
                nops(10);                       // Small Delay
                PORTC &= 0xFE;                  // Clear SCL
                nops(12);                       // Small Delay
                byte = byte << 1;               // Shift data in buffer right one
        }

        PORTC &= 0xFD;                                                          //
Clear SDA Low
        if(i2c_ackn()==1)  ucNack=1;
}


unsigned char read_i2c_byte(unsigned char ch)
{
        unsigned char i, buff=0;

        DDRC |= 0x02;                           // Set SDA to Output
        PORTC &= 0xFD;
        nops(2);

        for(i=0; i<8; i++)
```

```
        {
                buff <<= 1;
                DDRC &= 0xFD;                   // Set SDA to input

                PORTC |= 0x01;                  // Set SCL High, Clock bit out
                nops(10);                       // Small Delay
                if ((PINC & 0x02)==0x02) {      //Read data on SDA pin
                        buff |= 0x01;

                }

                PORTC &= 0xFE;                  // Clear SCL
                nops(12);                       // Small Delay
        }

        DDRC |= 0x02;                           // Set SDA to Output

        if(ch==0)                               // Ack
        {
                PORTC &= 0xFD;                  // Clear SDA Low
                i2c_clock();
        }
        else                                    // No Ack
        {
                PORTC |= 0x02;                  // Clear SDA High
                i2c_clock();
                nops(8);                        // Small Delay
                PORTC &= 0xFD;                  // Clear SDA Low
                nops(4);                        // Small Delay
                PORTC |= 0x02;                  // Clear SDA High
        }
        return buff;
}

void i2c_clock(void)
{
        PORTC |= 0x01;                          // Set SCL High
        Delay_us(5);                            // Small Delay
        PORTC &= 0xFE;                          // Clear SCL Low
}
```

PIC16F877A Code (CCS-C)

Header File

```
#include <16F877A.h>

#use delay(crystal=4MHz,restart_wdt)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,stream=PORT1)
#use i2c(Master,Fast,sda=PIN_C4,scl=PIN_C3)
```

C File

```
#include <uart_test.h>
#include <string.h>
#include <stdio.h>
char data[6];
float temp[4];
float humi[4];

void main()
{
    set_tris_c(0x80);
    while(true)
    {
        //measure request
        i2c_start();
        i2c_write(0x50);          //Read Address
        i2c_stop();
        delay_ms(15);             //Important delay

        //data fetch
        i2c_start();
        i2c_write(0x51);           //Write Address
        data[0] = i2c_read();    //Humidity MSB
        data[1] = i2c_read();    //HUmidity LSB
        data[2] = i2c_read();    //Temperature MSB
        data[3] = i2c_read(0); //Temperature LSB
        i2c_stop();

        //Humi & Temp Calculate
        data[0] &= 0x3f;
        data[3] &= 0xfc;
        humi[0]=data[0];
        humi[1]=data[1];
        humi[0]*=256;
        humi[0]/=163.83;
        humi[1]/=163.83;
        temp[0]= data[2];
        temp[1]= data[3];
        temp[0]*=64;
        temp[0]/=16384;
        temp[0]*=165;
        temp[1]/=4;
        temp[1]/=16384;
        temp[1]*=165;

        humi[2] = humi[0]+humi[1];
        temp[2] = temp[0]+temp[1]-40;

        printf("humi %f\t",humi[2]);
        printf("temp %f\r\n",temp[2]);
        delay_ms(500);
```

```
    }
}
```

STM32F103ZE06 Code(Keil for ARM)

Header File

```
#ifndef __MYIIC_H
#define __MYIIC_H
#include "sys.h"

//IIC SETTING
#define SDA_IN()  {GPIOB->CRH&=0XFFFF0FFF;GPIOB->CRH|=8<<12;}
#define SDA_OUT() {GPIOB->CRH&=0XFFFF0FFF;GPIOB->CRH|=3<<12;}
#define IIC_SCL    PBout(10) //SCL PORTB 10
#define IIC_SDA    PBout(11) //SDA PORTB 11
#define READ_SDA   PBin(11)  //Read SDA PINB 11

//IIC Function define
void IIC_Init(void);
void IIC_Start(void);
void IIC_Stop(void);
void IIC_Send_Byte(u8 txd);
u8 IIC_Read_Byte(unsigned char ack);
u8 IIC_Wait_Ack(void);
void IIC_Ack(void);
void IIC_NAck(void);

#endif
```

IIC C File

```
void IIC_Init(void)   //IIC Initial
{
        RCC->APB2ENR|=1<<3;
        GPIOB->CRH&=0XFFFF00FF;
        GPIOB->CRH|=0X00003300;
        GPIOB->ODR|=3<<10;
}

void IIC_Start(void) //IIC Start Condition
{
        SDA_OUT();
        IIC_SDA=1;
        IIC_SCL=1;
        delay_us(4);
        IIC_SDA=0;//START:when CLK is high,DATA change from high to low
        delay_us(4);
        IIC_SCL=0;
}

void IIC_Stop(void)          //IIC Stop Condition
{
        SDA_OUT();
        IIC_SCL=0;
        IIC_SDA=0; //STOP:when CLK is high DATA change from low to high
        delay_us(4);
        IIC_SCL=1;
        IIC_SDA=1;
        delay_us(4);
}

u8 IIC_Wait_Ack(void) //ACK
{
        u8 ucErrTime=0;
```

```
	SDA_IN();
	IIC_SDA=1;delay_us(1);
	IIC_SCL=1;delay_us(1);
	while(READ_SDA)
	{
		ucErrTime++;
		if(ucErrTime>250)
		{
			IIC_Stop();
			return 1;
		}
	}
	IIC_SCL=0;
	return 0;
}

void IIC_Ack(void) //ACK
{
	IIC_SCL=0;
	SDA_OUT();
	IIC_SDA=0;
	delay_us(2);
	IIC_SCL=1;
	delay_us(2);
	IIC_SCL=0;
}

void IIC_NAck(void) //NACK
{
	IIC_SCL=0;
	SDA_OUT();
	IIC_SDA=1;
	delay_us(2);
	IIC_SCL=1;
	delay_us(2);
	IIC_SCL=0;
}

void IIC_Send_Byte(u8 txd) //IIC Write Condition
{
    u8 t;
	SDA_OUT();
    IIC_SCL=0;
    for(t=0;t<8;t++)
    {
	SDA_OUT();
        IIC_SDA=(txd&0x80)>>7;
        txd<<=1;
		delay_us(2);
		IIC_SCL=1;
		delay_us(2);
		IIC_SCL=0;
		delay_us(2);
    }
}

u8 IIC_Read_Byte(unsigned char ack) //IIC Read Condition
{
	unsigned char i,receive=0;
	SDA_IN();
    for(i=0;i<8;i++ )
	{
```

```
        IIC_SCL=0;
        delay_us(2);
                IIC_SCL=1;
        receive<<=1;
        if(READ_SDA)receive++;
                delay_us(1);
    }
    if (!ack)
        IIC_NAck();
    else
        IIC_Ack();
    return receive;
}
```

Main C File

```
#include "sys.h"
#include "usart.h"
#include "delay.h"

int data[4];
float temp[5];
float humi[5];
int main(void)
{

        Stm32_Clock_Init(9);
        uart_init(72,9600);
        delay_init(72);
        IIC_Init();

        while(1)
        {
                //Measure Request
                IIC_Start();
                IIC_Send_Byte(0x50); //Write Address
                IIC_Wait_Ack();                 //Receive signal form sensor
                IIC_Stop();
                delay_ms(15);                   //Important delay

                //Data Fetch
                IIC_Start();
                IIC_Send_Byte(0x51); //Read Address
                IIC_Wait_Ack();
                data[0]=IIC_Read_Byte(1); //Humidity MSB
                data[1]=IIC_Read_Byte(1); //Humidity LSB
                data[2]=IIC_Read_Byte(1); //Temperature      MSB
                data[3]=IIC_Read_Byte(0); //Temperature LSB
                IIC_Stop();

                //Humidity & Temperature calculate
                data[0] &= 0x3f;
                data[3] &= 0xfc;
                humi[0]=data[0];
        humi[1]=data[1];
        humi[0]*=256;
        humi[0]/=163.83;
        humi[1]/=163.83;
        temp[0]= data[2];
        temp[1]= data[3];
        temp[0]*=64;
        temp[0]/=16384;
```

```
        temp[0]*=165;
        temp[1]/=4;
        temp[1]/=16384;
        temp[1]*=165;
        temp[2]=temp[1]+temp[0]-40;
        humi[2] = humi[0]+humi[1];

                printf("humi=%4.2f RH%% temp=%4.2f C\n\r",humi[2],temp[2]);
                delay_ms(500);
        }
}
```