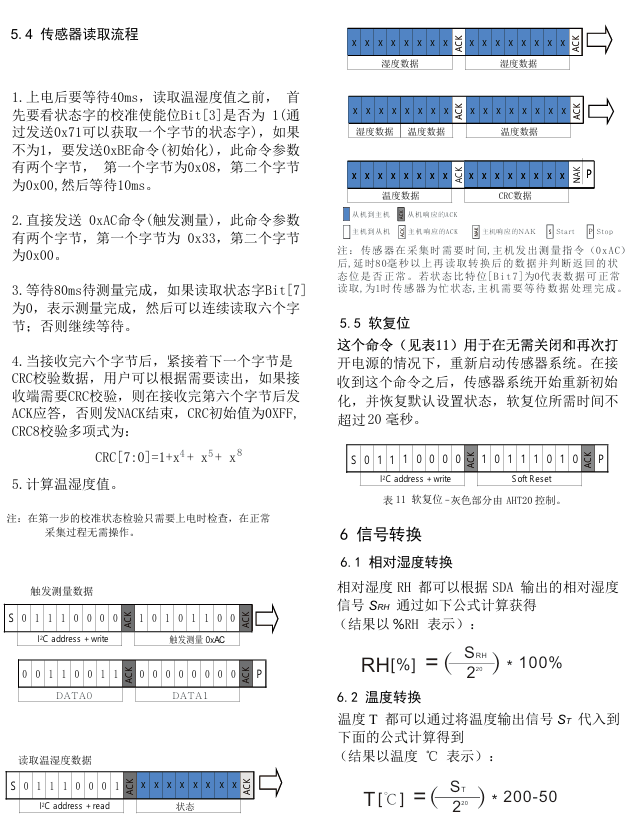
# AHT20温湿度传感器

## 1.模块介绍

AHT20是一款数字化温湿度传感器，广泛用于需要精确温湿度监测的场合，可通过I2C接口与外部设备通信。温度测量范围在-40℃-85℃之间，湿度测量范围在啊0%-100%RH(相对湿度)

AHT20温湿度传感器的读取流程和温湿度数据转换如下图所示：



## 2.程序实现

AHT20温湿度传感器的通信可使用标准IIC,具体IIC时序这里PIO，软件模拟IIC操作。

部分头文件如下：

#define AHT20\_SDA\_PIN       GPIO\_Pin\_11

#define AHT20\_SCL\_PIN       GPIO\_Pin\_10

#define AHT20\_GPIO\_PORT     GPIOB

#define AHT20\_GPIO\_CLK      RCC\_APB2Periph\_GPIOB

#define AHT20\_ADDRESS\_W     0x70 //读地址

#define AHT20\_ADDRESS\_R     0x71 //写地址

#define  AIIC\_SDA\_H  GPIO\_SetBits(AHT20\_GPIO\_PORT,AHT20\_SDA\_PIN)    //配置SDA接口高电平

#define  AIIC\_SDA\_L  GPIO\_ResetBits(AHT20\_GPIO\_PORT,AHT20\_SDA\_PIN)  //配置SDA接口低电平

#define  AIIC\_SCL\_H  GPIO\_SetBits(AHT20\_GPIO\_PORT,AHT20\_SCL\_PIN)    //配置SCL接口高电平

#define  AIIC\_SCL\_L  GPIO\_ResetBits(AHT20\_GPIO\_PORT,AHT20\_SCL\_PIN)  //配置SCL接口低电平

#define  AI2C\_SDA\_READ()  GPIO\_ReadInputDataBit(AHT20\_GPIO\_PORT, AHT20\_SDA\_PIN)  //读SDA口线状态

AHT20的7位IIC设备地址是0x38,第8位要跟一个方向位（读R：“1”，写W:“0”），因此AHT20的读地址是0x70,写地址是0x71。

**IIC开始信号**

//当IIC SCL线处于高电平时，SDA线由高电平向低电平跳变，为IIC开始信号，配置开始信号前必须保证IIC总线处于空闲状态

void AIIC\_Start()

{

    AIIC\_SDA\_H;

    AIIC\_SCL\_H;

    Delay\_Us(4);

    AIIC\_SDA\_L;

    Delay\_Us(4);

    AIIC\_SCL\_L;

    Delay\_Us(4);

}

**IIC停止信号**

//当IIC SCL线处于高电平时，SDA线由低电平向高电平跳变，为IIC停止信号

void AIIC\_Stop()

{

    AIIC\_SDA\_L;

    AIIC\_SCL\_H;

    Delay\_Us(4);

    AIIC\_SDA\_H;

}

**IIC空闲状态**

//当IIC总线的SDA和SCL两条信号线同时处于高电平时，规定为IIC总线的空闲状态

void AIIC\_Idle\_State()

{

    AIIC\_SDA\_H;

    AIIC\_SCL\_H;

    Delay\_Us(4);

}

**发送1字节数据**

void AIIC\_SendByte(u8 data)

{

    u8 i;

    //先发送字节的高位bit7

    for (i = 0; i < 8; i++)

    {

        if (data & 0x80)  //判断8位数据每一位的值（0或1）

        {

            AIIC\_SDA\_H;

        }

        else

        {

            AIIC\_SDA\_L;

        }

        Delay\_Us(4);      //控制SCL线产生高低电平跳变，产生通讯时钟，同时利用延时函数在SCL为高电平期间读取SDA线电平逻辑

        AIIC\_SCL\_H;

        Delay\_Us(4);

        AIIC\_SCL\_L;

        if (i == 7)

        {

            AIIC\_SDA\_H;    //控制SDA线输出高电平，释放总线，等待接收方应答信号

        }

        data <<= 1;       //左移一个bit

        Delay\_Us(4);

    }

}

**读取1字节数据**

u8 AIIC\_ReadByte(void)

{

    u8 i;

    u8 value;

    //读到第1个bit为数据的bit7

    value = 0;

    for(i = 0; i < 8; i++)

    {

        value <<= 1;

        AIIC\_SCL\_H;

        Delay\_Us(4);

        if (AI2C\_SDA\_READ()) //利用延时函数在SCL为高电平期间读取SDA线电平逻辑

        {

            value++;

        }

        AIIC\_SCL\_L;

        Delay\_Us(4);

    }

    return value;

}

**等待应答信号**

u8 AIIC\_WaitAck(void)

{

    uint8\_t rvalue;

    AIIC\_SDA\_H;     //发送端释放SDA总线，由接收端控制SDA线

    Delay\_Us(4);

    AIIC\_SCL\_H;     //在SCL为高电平期间等待响应，若SDA线为高电平，表示NACK信号，反之则为ACK信号

    Delay\_Us(4);

    if(AI2C\_SDA\_READ())  //读取SDA线状态判断响应类型，高电平，返回去，为NACK信号，反之则为ACK信号

    {

        rvalue = 1;

    }

    else

    {

        rvalue = 0;

    }

    AIIC\_SCL\_L;

    Delay\_Us(4);

    return rvalue;

}

**产生应答信号ACK**

void AIIC\_ACK(void)

{

    AIIC\_SDA\_L;

    Delay\_Us(4);

    AIIC\_SCL\_H;   //在SCL线为高电平期间读取SDA线为低电平，则为ACK响应

    Delay\_Us(4);

    AIIC\_SCL\_L;

    Delay\_Us(4);

    AIIC\_SDA\_H;

}

**产生非应答信号NACK**

void AIIC\_NACK(void)

{

    AIIC\_SDA\_H;

    Delay\_Us(4);

    AIIC\_SCL\_H;   //在SCL线为高电平期间读取SDA线为高电平，则为NACK响应

    Delay\_Us(4);

    AIIC\_SCL\_L;

    Delay\_Us(4);

}

完成模拟IIC的操作函数后，接下来就使用这些函数初始化AHT20。依据AHT20手册，上电后要发送0x71,然后读取1字节状态字，如果状态字的校准使能位Bit[3]不为1，就要发送0xBE（此命令有两个参数，第1个字节0x08,第2个字节0x00）初始化。

**读取AHT20状态**

uint8\_t AHT21\_ReadState(void) // 读取 AHT21 状态

{

    uint8\_t State;

    AIIC\_Start();//开始

    AIIC\_SendByte(AHT20\_ADDRESS\_W | 0x01);//从机地址＋读

    AIIC\_WaitAck();//接收应答

    State=AIIC\_ReadByte();//接收状态

    AIIC\_NACK();//发送非应答

    AIIC\_Stop();//停止

    return State;

}

**通用AHT20写入（命令+两字节）**

void AHT21\_WriteHalfWord(uint8\_t Command,uint8\_t Data0,uint8\_t Data1)

{

    AIIC\_Start();//开始

    AIIC\_SendByte(AHT20\_ADDRESS\_W);//从机地址＋写

    AIIC\_WaitAck();//应答

    AIIC\_SendByte(Command);//命令

    AIIC\_WaitAck();//应答

    AIIC\_SendByte(Data0);//写入Data0

    AIIC\_WaitAck();//应答

    AIIC\_SendByte(Data1);//写入Data1

    AIIC\_WaitAck();//应答

    AIIC\_Stop();//停止

}

**通用AHT20写入（命令）**

void AHT21\_WriteCommand(uint8\_t Command)

{

    AIIC\_Start();     //开始

    AIIC\_SendByte(AHT20\_ADDRESS\_W);//从机地址＋写

    AIIC\_WaitAck();//应答

    AIIC\_SendByte(Command);//命令

    AIIC\_WaitAck();//应答

    AIIC\_Stop();//停止

}

**AHT20初始化寄存器**

void AHT21\_WriteInit(void) //AHT21 写入[初始化 寄存器指令]

{

    AHT21\_WriteHalfWord(0xBE,0x08,0x00);

}

**AHT20初始化**

void AHT21\_Init(void)

{

    uint8\_t State = 0;

    AIIC\_Init();

    //1 上电读取状态

    State = AHT21\_ReadState();//读取状态

    //2 判断状态位 bit[3] 0x18未初始化寄存器/0x1c已初始化寄存器

    if((State&0x08) != 0x08)//不等于0000 0100，进行初始化寄存器

    {

        AHT21\_WriteInit();//如果第三位是0，则初始化寄存器

        Delay\_Ms(10);

    }

}

完成硬件初始化后，就可以读取温湿度值了。发送0xAC命令（此命令有两个参数，第一个字节为0x33,第二个字节为0x00）

**写入温湿度触发测量指令**

void AHT21\_WriteMeasure(void) //写入触发测量指令

{

    AHT21\_WriteHalfWord(0xAC,0x33,0x00);

}

然后等待80ms测量完成，如果读取状态字Bit[7]为0，表示测量完成，然后可以连续读取6个字节。

**发送测量命令，读取温度、湿度**

float\* AHT21\_ReadTemperatureHumidity(void)//传回TH地址

{

    uint8\_t Array[6] = {0};     //定义一个5位数据的数组 存放字节

    uint32\_t Temp = 0;          //定义32位数组 暂时存放 20位数据

    uint32\_t Result[2] = {0};   //定义一个2位数据的数组 存放数据结果

    static float TH[2] = {0};   //静态变量，返回最终数值

    AHT21\_WriteMeasure();//发送测量命令

    Delay\_Ms(500);

    AIIC\_Start();//开始

    AIIC\_SendByte(AHT20\_ADDRESS\_W | 0x01);   //从机地址＋读

    AIIC\_WaitAck();

    Array[0]=AIIC\_ReadByte();                    //状态字

    AIIC\_ACK();//发送应答

    Array[1]=AIIC\_ReadByte();         //湿度

    AIIC\_ACK();//发送应答

    Array[2]=AIIC\_ReadByte();         //湿度

    AIIC\_ACK();//发送应答

    Array[3]=AIIC\_ReadByte();         //湿度和温度

    AIIC\_ACK();//发送应答

    Array[4]=AIIC\_ReadByte();         //温度

    AIIC\_ACK();//发送应答

    Array[5]=AIIC\_ReadByte();         //温度

    AIIC\_ACK();//不发送应答，结束

    AIIC\_Stop();

    if(((Array[0]&0x80) == 0x00)) //如果测量完成 bit[7]=0,进行数据处理

    {

        Temp = (Temp | Array[1]) << 8;  //执行后0000 0000  0000 0000  aaaa aaaa  0000 0000

        Temp = (Temp | Array[2]) << 8;  //执行后0000 0000  aaaa aaaa  bbbb bbbb  0000 0000

        Temp = (Temp | Array[3]);   //执行后0000 0000  aaaa aaaa  bbbb bbbb  cccc cccc

        Temp =Temp >> 4;//数据右移  //执行后0000 0000  0000 aaaa  aaaa bbbb  bbbb cccc

        Result[0] = Temp;//湿度

        Temp = 0;

        Temp = (Temp | Array[3]) << 8;  //执行后0000 0000  0000 0000  cccc cccc  0000 0000

        Temp = (Temp | Array[4]) << 8;  //执行后0000 0000  cccc cccc  eeee eeee  0000 0000

        Temp = (Temp | Array[5]);       //执行后0000 0000  cccc cccc  eeee eeee  ffff ffff

        Temp = Temp & 0x000fffff;//置0   //执行后0000 0000  0000 cccc  eeee eeee  ffff ffff

        Result[1] = Temp;//温度

        TH[0] = (float)Result[0]/AHT21\_Pow(2,20);

        TH[1] = (float)Result[1]/AHT21\_Pow(2,20)\*200-50;

        return TH;

    }

    else

    {

        printf("读失败/r/n");

        TH[0] = 0;

        TH[1] = 0;

        return TH;

    }

}

**软复位**

void AHT21\_WriteReset(void)//软复位AHT21

{

    AHT21\_WriteCommand(0xBA);

}