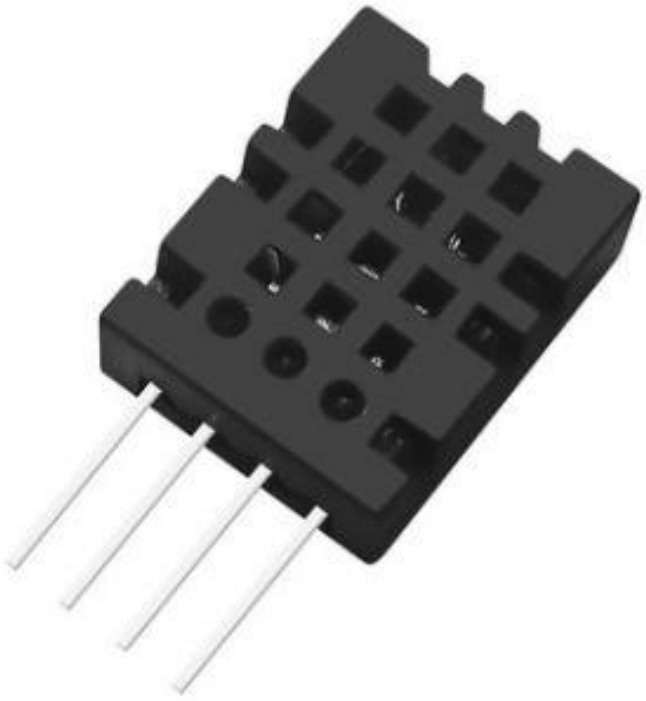
温湿度模块

DHTC12 产品手册

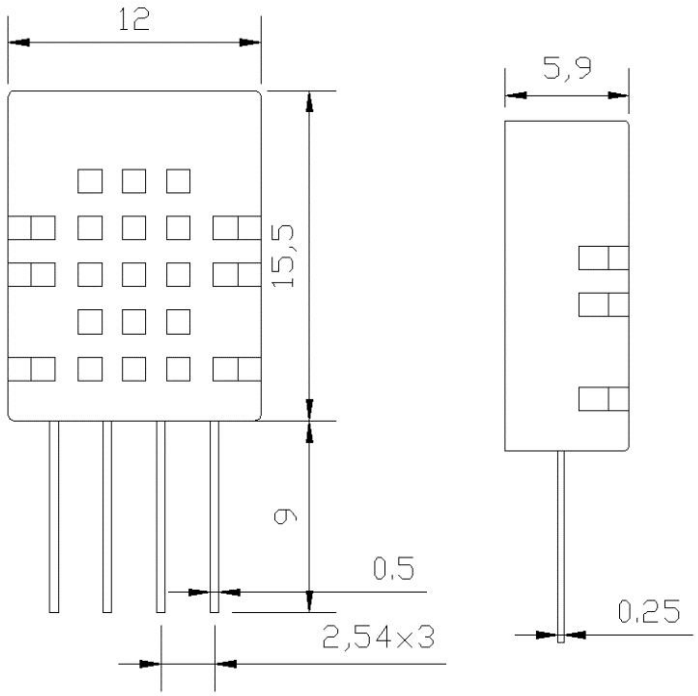


一、 产品概述

本产品是采用高稳定性电容式感湿元件作为传感元件，经过微处理器采集处理转化成数字信号输出。每一个传感器都经过标定校准和测试。具有长期稳定、可靠性高、 精度高、低功耗等特点。

二、 尺寸图

单位:mm(±0.5)



三、产品特点

DHTC12 数字温湿度模块具有以下特点：

1、数字输出，IIC 协议；

2、超低功耗；

3、0-100%相对湿度测量范围；

4、全标定、温漂校准。

5、使用独立感湿元器件，稳定性好，抗污染能力强

四、性能特征

相对湿度

参数	条件	最小	典型	最大	单位
分辨率			0.1		%RH
量程范围		0		99.9	%RH
精度	25℃		±3		%RH
重复性			±0.1		
响应时间	1/e(63%)		<8		S
迟滞			±0.5		
漂移	典型值		< 3		%RH/r

表 1

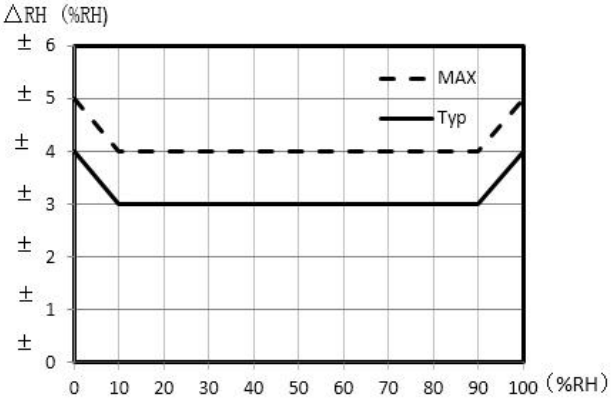


图 1

温度

参数	条件	最小	典型	最大	单位
分辨率			0.1		℃
工作范围		-40		80	℃
精度			±0.5		℃
重复性			±0.2		℃
响应时间	1/e(63%)	1			S
迟滞			±0.1		
漂移	典型值		< 0.2		℃/r

表 2

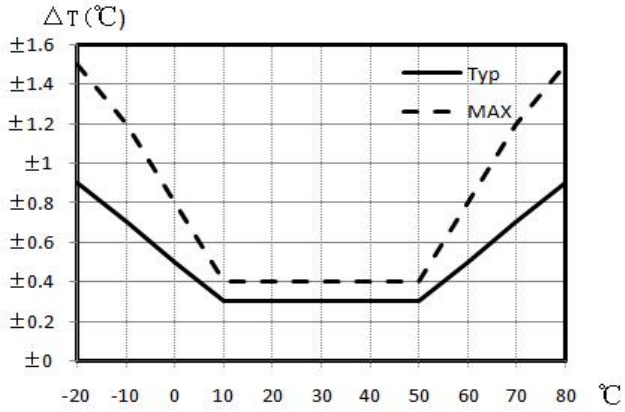


图 2

五、电气特性

参数	条件	最小	典型	最大	单位
供电电压 VDD		2.8	3.3	5.5	V
供电电流	休眠模式		0.2		uA
	测量模式		500		uA
采样周期			2.0		S
低电平输出电压	Io<4mA	0		250	mV
高电平输出电压	Rp<25kΩ	80%		100%	VDD
低电平输入电压	下降沿	0%		20%	VDD
高电平输入电压	上升沿	80%		100%	VDD
输出电流	On			4	mA
	三态门（Off）		10	20	μA

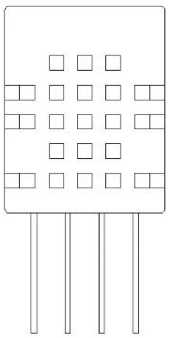
表 3　Io 表示低电平输出电流 Rp 代表上拉电阻

六、用户指南

1、 引脚分配

引脚	名称	描述
1	VDD	电源 2.8～5.5V
2	SDA	串行数据，双向口
3	GND	地
4	SCK	时钟线

表4：引脚分配



1. 2.3.4.
透气孔朝上

1.1、电源引脚（VDD GND）

本产品的供电电压为2.8～5.5V，建议供电电压为3.3V。

1.2、串行数据（SDA）

SDA 为数据口，三态结构，SDA 在 SCK 时钟下降沿之后改变状态，并仅在 SCK 上升沿有效

1.3、串行数据（SCK）

SCK 用于微处理器与 DHTC12 之间的通讯同步。注意频率范围。

2、通信协议

DHTC12 为了精确测量气体的湿度，减少温度对测量的影响，DHTC12 传感器在非工作期间，自动转为休眠模式，以降低传感器自身的发热对周围气体湿度的影响。DHTC12 采用被动式工作模式，即主机通过指令唤醒传感器后，传感器才开始测量、应答等动作。通讯结束后，传感器进入休眠状态。

2.1、DHTC12 连接图

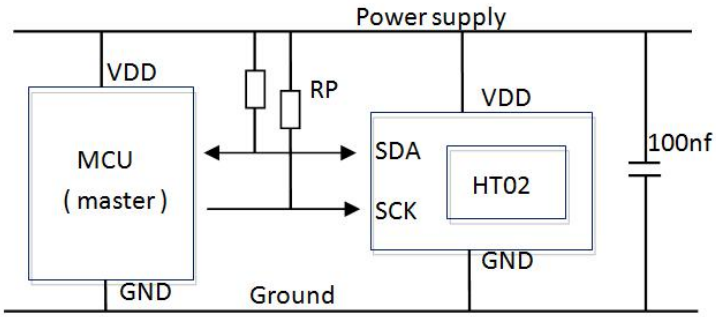


图 3　IIC 通讯连接图

2.2、输入/输出特性

电气特性，如功耗、输入和输出的高、低电平电压等，依赖于电源供电电压。表 3 详细解释了 DHTC12 的电气特性。若想与传感器获得最佳的通讯效果，请设计时严格遵照表 4 与图 5 的条件。

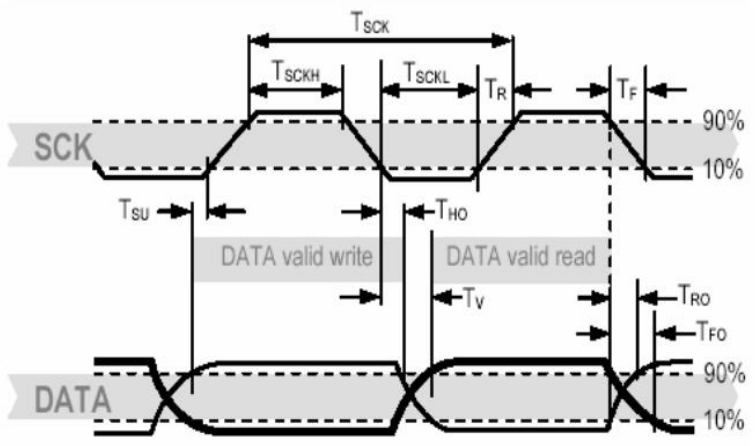


图 4

	参数	最小	典型	最大	单位
F_{SCK}	SCK 频率			100	KHZ
T_{SCKx}	SCK 高/低时间	5			us
T_R/T_F	SCK 升/降时间			1	us
T_{FO}	SDA 下降时间			800	ns
T_{RO}	SDA 上升时间			1000	ns
T_V	SDA 有效时间		400		ns
T_{SU}	SDA 设定时间		350		ns
T_{HO}	SDA 保持时间		100		ns

表 5

2.3、IIC 通讯协议（说明书末尾附带 C 语言程序 附件 1）

DHTC12 采用标准的 I2C 协议进行通讯。因此关于 I2C 的更多细节该手册将不做赘述。欲获取下述章节以外的关于 I2C 协议的资料，请自行参阅。

2.4、启动传感器

首先，选择合适的电源，上电速率不能低于 1V/ms。

2.5、启动/停止时序

每个传输系列都以 Start 状态作为开始并以 Stop 状态作为结束

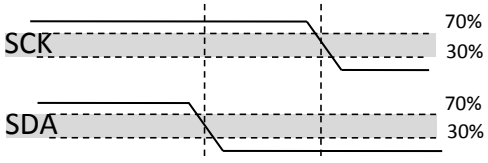


图 5

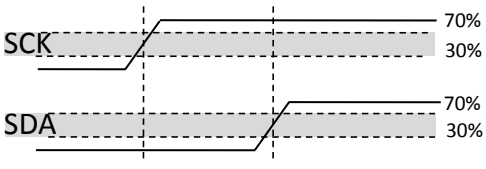


图 6

图 6 启动传输状态（S）-当 SCK 为高电平时，SDA 由高电平转换为低电平。开始状态是由于主机控制的一种特殊的总线状态，指示从机传输开始。

图 7 停止传输状态（P）-当 SCK 为高电平时，SDA 由低电平转换为高电平。停止状态是由于主机控制的一种特殊的总线状态，指示从机传输结束。

2.6、发送命令

在启动传输后，随后传输的 I2C 首字节包括 7 位的 I2C 设备地址（当前只支持 44H）和一个 SDA 方向位（读 R：‘1’，写 W：‘0’）。在第 8 个 SCL 时钟下降沿之后，通过拉低 SDA 信号线（ACK 位），指示传感器数据通信正常。

传感器地址	44H
触发温湿度测量	2C10H
触发温度测量	CC44H
触发湿度测量	CC66H

表 6　传感器通讯参数

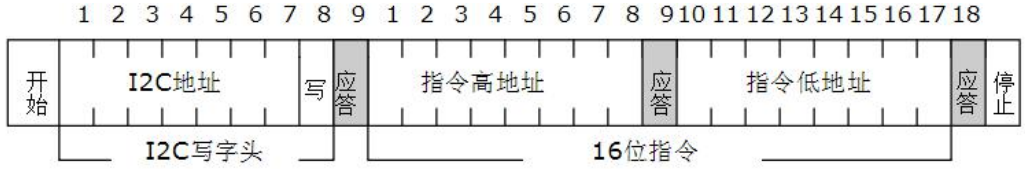
2.7、读取温湿度

DHTC12 读取温湿度时序图如下：白色由微控制器控制，传感器响应为灰色块

读取步骤如下：

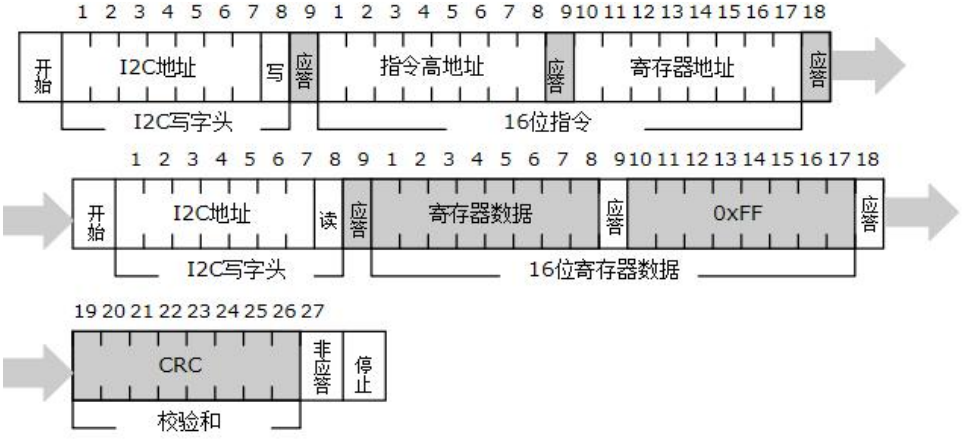
1)、复位传感器、初始化。

指令	十六进制编码
软复位	0x30A2



2）、获取湿度运算系数与系统配置（可不读取）

指令	十六进制编码
读取 HumA 高 8 位	0xD208
读取 HumA 低 8 位	0xD209
读取 HumB 高 8 位	0xD20A
读取 HumB 低 8 位	0xD20B
读取配置字 Cfg（可不读取）	0xD206

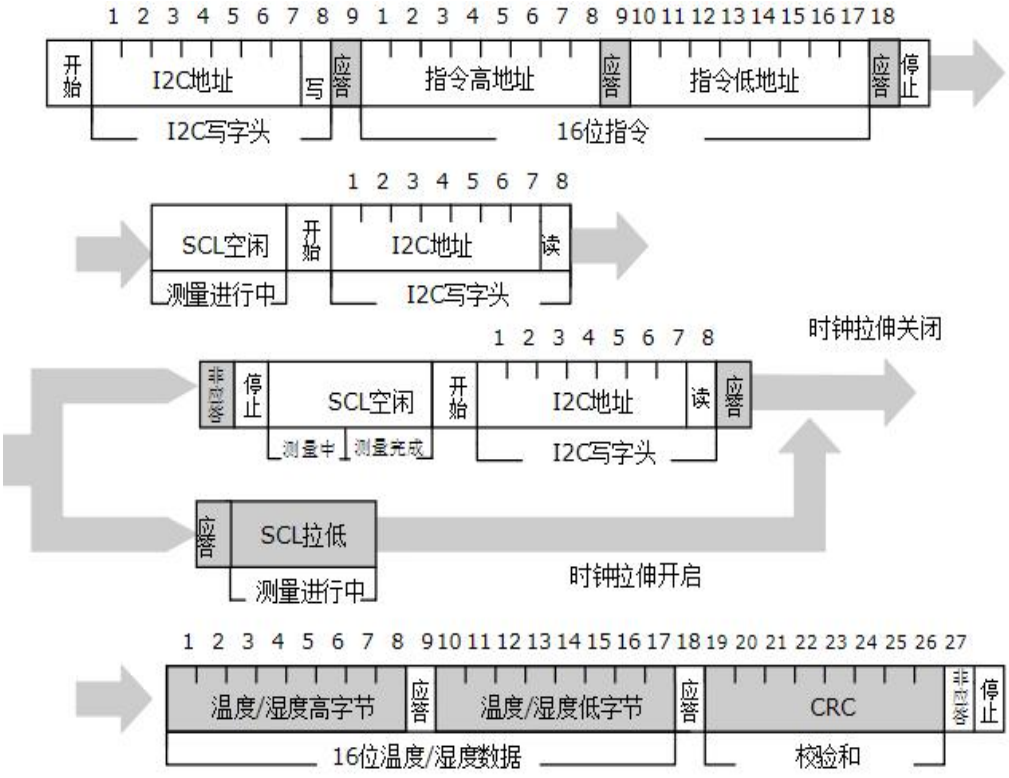


3)、触发测量温湿度

测温、测湿单独模式

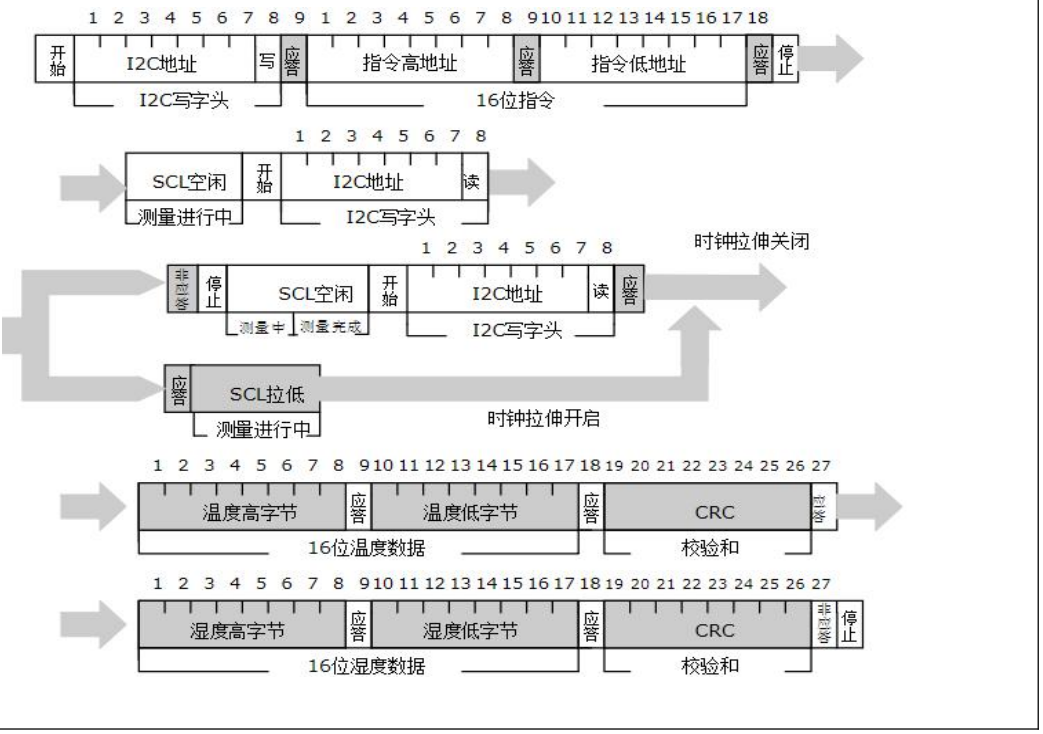
通过发送测量命令 CC44H 或 CC66H 触发一次湿度或温度数据的采集。在传输期间，每个数据值始终跟随 CRC 检验和。下表中显示了 16 位命令。重复性（低，中和高 默认为高）和时钟延展（启用或禁止 默认是禁止）可以通过修改配置寄存器实现。重复性影响测量持续时间，从而会影响传感器的总功耗

指令	十六进制编码
测量温度	0xCC44
测量湿度	0xCC66



同时测量模式

指令	十六进制编码
测量温度和湿度	0x2C10



温度、相对湿度转换

通过上面两种方式可以获得 16bit 温度原始数据 S_t、16bit 湿度原始数据 S_h

最终带入下面公式

温度（℃）：

$$T = 40 + \frac{St}{256}$$

湿度（%RH）：

$$RH = 30 + \frac{(Sh - HumB) * 60}{HumA - HumB}$$

$$RHrel = RH + 0.25*(T - 25.0)$$ （注：温漂 1℃为 -0.25%RH）

注：RH 限定在 0～100， if(RH>100) RH=100;
Else if(RH<0) RH=0;

4）修改配置寄存器

Cfg	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x02	Reserved	Reserved	clk_strench	Mps2	Mps1	Mps0	Repeatbility1	Repeatbility1
描述	保留		IIC 接口时钟拉伸使能： 0：不允许拉伸 1：允许拉伸	每秒测量次数选择： 000：单次			重复性设置： 00：低可重复性 01：中等可重复性 10：高可重复性	

指令	十六进制编码
修改配置字 Cfg	0x5206

5）循环冗余校验（CRC）计算

在每个数据字之后发送的 8 位 CRC 校验和由 CRC 算法生成。其属性显示在下表中。

属性	数值
名称	CRC-8
位宽	8 位
保护数据	读/写
多项式	0x31（$X^8+X^5+X^4+1$）
初始数据	0xFF
待测数据的每个字节是否按位反转	否

七、应用信息

1、工作条件

确保传感器性能正常稳定的工作，建议使用温度范围-40℃-80℃，湿度范围 0-99.9%RH。超出建议的范围可能导致测量结果暂时性漂移。

2、存储条件与恢复

湿度传感器为环境敏感型电子元器件，需要仔细防护。长期暴露在高浓度的化学蒸汽中将会致使传感器的测量产生漂移。因此建议将传感器存放于原包装内，并符合存储条件：温度范围 10℃-50℃；湿度范围 20-60%RH。在生产和运输过程中，要保证传感器远离高浓度的化学溶剂。要避免使用挥发性胶水、粘性胶带、不干胶贴纸，或者具有挥发性的包装材料，如发泡塑料袋、泡沫塑料。

3、温度影响

相对湿度，很大程度上依赖于温度。产品在出厂前都做了温度的校准补偿，测量湿度时，应尽可能的保证传感器在同一温度下工作，安装在产品上时要尽可能的远离热源。否则将无法准确的测试到气体的相对湿度。

八、特别说明

许可协议

以上内容由本公司提供，版权所有，未经本公司之书面许可，此手册中任何段落，章节内容均不得被摘抄、拷贝或以任何形式复制、传播，否则一切后果由违者自负，本公司保留一切法律权利。

本公司保留对手册所描述之产品规格进行修改的权利，恕不另行通知。订货前，请垂询当地代理商以获悉本产品的最新规格。

警告

使用及人身伤害

勿将本产品用于安全保护装置或急停设备上，以及由于本产品故障可能导致人身受到伤害的任何应用中；在使用本产品前，请仔细阅读本说明书中的内容；

禁止在易燃气体附近使用

禁止在易燃、易爆气体的场所使用；

严禁直接触及传感器

为防止污染感湿膜，避免手指直接触摸元件表面；汗液会污染感湿膜会导致性能漂移，接触传感器请戴防静电手指套；

工作环境

建议使用温度范围-40℃-80℃，湿度范围 0-100%RH。超出建议的范围可能导致测量结果暂时性漂移；本产品对光线不敏感，但长时间暴露在太阳光或则紫外线辐射中，同样加速老化；

附件 1

```
#define V4_I2C_ADDR  0x44  /* Addr 引脚接低电平*/
//获取湿度校准参数
static u16 HumA,HumB;

void DHTC12Init(void)
{
    ResetMD();
    delay(10);    //10us 或者不用
    HumA = Read_Rg(8);
    HumA = (HumA<<8)|Read_Rg(9);
    HumB = Read_Rg(10);
    HumB = (HumB<<8)|Read_Rg(11);
}

//单次触发温湿度测量
u8 ReadDHTC12(s16 *tem,u16 *Hum)
{
    static u8 initFlag=0;
    u8 i,errRe,ReadDatSH[6],CalCRC[3],errorflag;
    s16 TemBuf;
long CapBuf;//s32

    start();
    write_byte(V4_I2C_ADDR<<1);// Add+W
    delay(1);
    respons();
    write_byte(0x2c);//单字节读取寄存器 指令 D2xx
    respons();
    write_byte(0x10);
    respons();
    stop();
    delay(465);//2ms  SCL 空闲最小 1ms

    for(i=0;i<50;i++)//查询 5 次看测完结果
    {
        delay(230);//1ms
        delay(230);
        delay(230);

        start();
        write_byte(V4_I2C_ADDR<<1|0x01);// Add+R
        errRe=respons();
        if(errRe ==0)
            break;//测量完成
        else
            stop();
```

```
  }

  if(errRe == 0)
{
     ReadDatSH[0] = read_byte();
     SendACK(ACK);
     ReadDatSH[1] = read_byte();
     SendACK(ACK);
     ReadDatSH[2] = read_byte();
     SendACK(ACK);
     ReadDatSH[3] = read_byte();
     SendACK(ACK);
     ReadDatSH[4] = read_byte();
     SendACK(ACK);
     ReadDatSH[5] = read_byte();
     SendACK(NACK);
     stop();
}
  CalCRC[0] = ReadDatSH[0];
  CalCRC[1] = ReadDatSH[1];
  CalCRC[2] = ReadDatSH[2];
  errorflag = SHT3x_CheckCrc(CalCRC,2);
  if(errorflag==0)
  {
     TemBuf = (u16)ReadDatSH[0]<<8|(ReadDatSH[1]);
     TemBuf = 400+TemBuf/25.6;//*10 结果*10 倍 286 即 28.6℃
     *tem = TemBuf;
  }

  CalCRC[0] = ReadDatSH[3];
  CalCRC[1] = ReadDatSH[4];
  CalCRC[2] = ReadDatSH[5];
  errorflag |= SHT3x_CheckCrc(CalCRC,2);
  if(errorflag==0)
  {
   CapBuf = (u16)ReadDatSH[3]<<8|(ReadDatSH[4]);
   CapBuf = (CapBuf-HumB)*600/(HumA-HumB)+300;
     //20℃为 5 个湿度点  即 1℃为 0.25 个湿度点 0.1℃ 为 0.025
   CapBuf = CapBuf+ 25*(TemBuf-250)/100;

   if(CapBuf>1000)
    CapBuf = 999;
   else if(CapBuf<0)
     CapBuf = 0;

     *Hum = (u16)CapBuf;//同样结果*10
  }
```

```
    return errorflag;
}



//子函数------------------------------
#define    ACK  0
#define   NACK   1
static u8 ACK_ret;

void start(void){
    HDCSDA_SET();
        delay(10); //IIC 时钟频率可以 0~100Hz
    HDCSCL_SET();
        delay(10);
    HDCSDA_CLR();
        delay(10);
    HDCSCL_CLR();
}
void stop(void){
    HDCSDA_CLR();
        delay(10);
    HDCSCL_SET();
        delay(10);
    HDCSDA_SET();
        delay(10);
}
u8 respons(void)
{
    u8 i=0,error=0;//写完后紧跟着 WaitACK，所有可以改变 SDA
    HDCSDA_Input();
    delay(10);
    HDCSCL_SET();
        delay(10);
    ACK_ret = 0;
    while(((HDCGet_SDA())!=0)&&i<50)i++;
    if(i>45)
    {
      error = 1;
        ACK_ret = 1;
    }
    HDCSCL_CLR();//SCK:0 可以改变 SDA
        delay(10);
    HDCSDA_Output();
    return error;
}
```

```
void SendACK(u8 ack)
{
    if(ack == 0)
    {
          HDCSDA_CLR();
    }
    else
    {
      HDCSDA_SET();
    }
    delay(10);
    HDCSCL_SET();
     delay(10);
       HDCSCL_CLR();
     delay(10);
      HDCSDA_SET();
}

void write_byte(u8 date){
    u8 i,temp;
    temp=date;
    for(i=0;i<8;i++)
    {
       HDCSCL_CLR();
//  delay(10);
       if((temp&0x80)==0x80)
           {HDCSDA_SET();}else{HDCSDA_CLR();}
           delay(10);
       HDCSCL_SET();
           delay(10);
       temp=temp<<1;
    }
    HDCSCL_CLR();
    delay(10);
//  HDCSDA_SET();
//  delay(10);
}

u8 read_byte(void){
    u8 i,k;
    HDCSDA_SET();//让 P 口准备读数
    HDCSDA_Input();
    for(i=0;i<8;i++)
    {
       HDCSCL_SET();
           delay(10);
       k <<= 1;
```

```
        if(HDCGet_SDA())//==0x04
            k |= 0x01;
        HDCSCL_CLR();//SCK:0 可以改变 SDA
            delay(10);
    }
    HDCSDA_Output();//
  return k;
}

const u16 POLYNOMIAL = 0x131;  //P(x)=x^8+x^5+x^4+1 = 100110001
//===========================================================================
u8 SHT3x_CheckCrc(u8 data[], u8 nbrOfBytes)  //跟 SHT30 一样只是将 CRC 值到会到 data 后返
回
//===========================================================================
{
  u8 crc = 0xff; //0
  u8 byteCtr,bit;
  //calculates 8-Bit checksum with given polynomial
  for (byteCtr = 0; byteCtr < nbrOfBytes; ++byteCtr)
  { crc ^= (data[byteCtr]);
    for (bit = 8; bit > 0; --bit)
    { if (crc & 0x80) crc = (crc << 1) ^ POLYNOMIAL;
      else crc = (crc << 1);
    }
  }
  if (crc != data[nbrOfBytes])
    {
       data[nbrOfBytes] = crc;
       return 0x01;
    }
  else return 0;
}


//软复位
void ResetMD(void)
{
    start();
    write_byte(V4_I2C_ADDR<<1);// Add+W
    delay(1);
    respons();
    write_byte(0x30);
    respons();
    write_byte(0xA2);
    respons();
    stop();
}
```

```
//读取寄存器数据
u16 Read_Rg(u8 AddRg)
{
    u8 errRe,ReadDatSH[3];
    u16 ReDat;

    start();
    write_byte(V4_I2C_ADDR<<1);// Add+W
    delay(1);
    respons();
    write_byte(0xD2);//单字节读取寄存器 指令 D2xx
    respons();
    write_byte(AddRg);
    respons();

    delay(10);
    start();
    write_byte(V4_I2C_ADDR<<1|0x01);// Add+R
    errRe=respons();
    //if(ACK_ret == 0)
  //{
    ReadDatSH[0] = read_byte();
    SendACK(ACK);
    ReadDatSH[1] = read_byte();
    SendACK(ACK);
    ReadDatSH[2] = read_byte();
    SendACK(NACK);
    //}
    stop();
    errRe = SHT3x_CheckCrc(ReadDatSH,2);
    if(errRe==0)
    {
       ReDat = ReadDatSH[0];
    }else
    {
       ReDat = 0xff;
    }
    return ReDat;
}
```