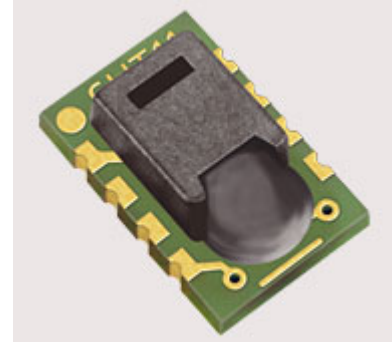


数字温湿度传感器 SHT 10

SHT 10

(请以英文为准, 译文仅供参考)

- _ 相对湿度和温度测量
- _ 兼有露点
- _ 全部校准, 数字输出,
- _ 卓越的长期稳定性
- _ 无需额外部件
- _ 超低能耗
- _ 表面贴片或4 引脚安装 完全互换
- _ 超小尺寸
- _ 自动休眠



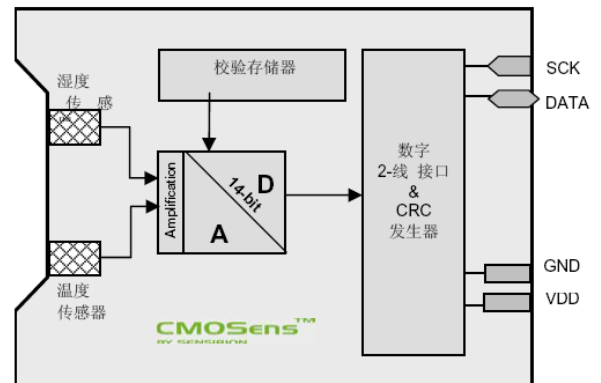
SHT10

SHT10 系列为贴片型温湿度传感器芯片; 全量程标定, 两线数字输出; 湿度测量范围: 0~100%RH; 湿度测量范围: -40~+123.8℃; 湿度测量精度: $\pm 4.5\%RH$; 温度测量精度: $\pm 0.5^{\circ}C$ 响应时间: $<8s$; 低功耗 (typ. 30 μW); 可完全浸没。

SHT10 产品概述

SHTxx 系列单芯片传感器是一款含有已校准数字信号输出的温湿度复合传感器。它应用专利的工业 COMS 过程微加工技术 (CMOSens®), 确保产品具有极高的可靠性与卓越的长期稳定性。传感器包括一个电容式聚合物测湿元件和一个能隙式测温元件, 并与一个 14 位的 A/D 转换器以及串行接口电路在同一芯片上实现无缝连接。因此, 该产品具有品质卓越、超快响应、抗干扰能力强、性价比极高等优点。每个 SHTxx 传感器都在极为精确的湿度校验室中进行校准。校准系数以程序的形式储存在 OTP 内存中, 传感器内部在检测信号的处理过程中要调用这些校准系数。两线制串行接口和内部基准电压, 使系统集成变得简易快捷。超小的体积、极低的功耗, 使其成为各类应用甚至最为苛刻的应用场合的最佳选择。产品提供表面贴片 LCC (无铅芯片) 或 4 针单排引脚封装。特殊封装形式可根据用户需求而提供。

框图



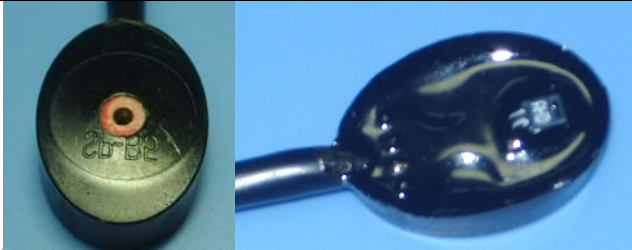
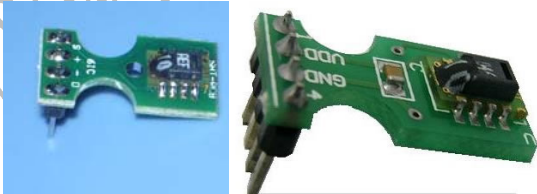
应用领域: 数据采集器 变送器 自动化过程控制 汽车行业 楼宇控制&暖通空调 电力 计量测试 医药业

传感芯片订货信息:

型号	测湿精度 [%RH]	测温精度 [°C]在 25°C	封装
SHT 10	±4.5	±0.5	SMD (LCC)
SHT 11	±3.0	±0.4	SMD (LCC)
SHT 15	±2.0	±0.3	SMD (LCC)
SHT 71	±3.0	±0.4	4-pin 单排直插
SHT 75	±1.8	±0.3	4-pin 单排直插

防护型温湿度传感器及插针型产品订货信息

(以下产品由我司根据用户要求订制生产, 用户特殊要求可订制)

订货型号	名称	
SHT10-P	防护型温湿度传感器	
SHT11-P	防护型温湿度传感器	
SHT15-P	防护型温湿度传感器	
SHT10-D	插针型温湿度传感器	
SHT15- D	插针型温湿度传感器	
SHT15- D	插针型温湿度传感器	

系列仪器订货信息

SLIH1-1	壁挂智能温湿度测试仪	
SLIH1-1/I	一体化智能温湿度测试仪 (内置温湿度传感器)	
SLIH1-1/R	组网型智能温湿度测试仪 (带 RS485 接口)	
SLMT2-2	智能温湿度数据采集模块	
SLMT2-2/I	一体化智能温湿度数据采集模块 (内置温湿度传感器)	

1 传感器性能说明

参数	条件	Min.	Typ.	Max.	单位
湿度					
分辨率 ⁽¹⁾		0.5	0.03	0.03	%RH
		8	12	12 ⁽²⁾	Bit
重复性			±0.1		%RH
精度 ⁽¹⁾ 不确定性	线性化	参见图 1			
互换性		可完全互换			
非线性度	原始数据		±3		%RH
	线性化		<<1		%RH
量程范围		0		100	%RH
响应时间	1/e (63%) 25°C, 1m/s 空气	6	8	10	s
迟滞			±1		%RH
长期稳定性	典型值		< 0.5		%RH/yr
温度					
分辨率 ⁽²⁾		0.04	0.01	0.01	°C
		0.07	0.02	0.02	°F
		12	14	14	Bit
重复性			±0.1		°C
			±0.2		°F
精度 ⁽³⁾		参见图 1			
量程范围		-40		123.8	°C
		-40		254.9	°F
响应时间	1/e (63%)	5		30	s

表 1 传感器性能说明

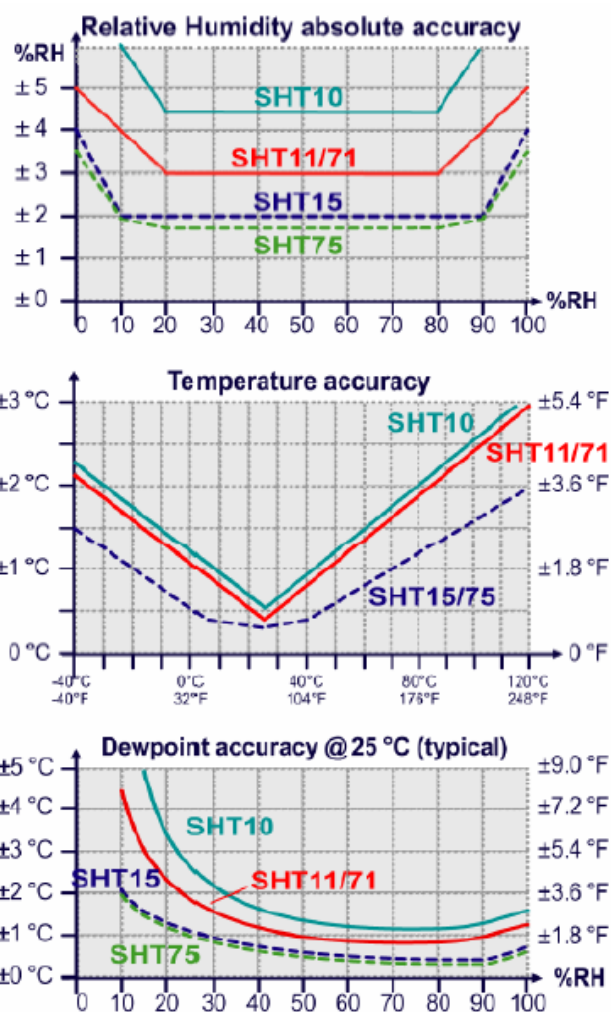


图 1 相对湿度、温度和露点的精度曲线

2 接口说明

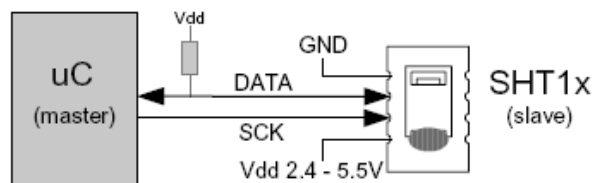


图 2 典型应用电路

2.1 电源引脚

SHTxx 的供电电压为 2.4~5.5V。传感器上电后,要等待 11ms 以越过“休眠”状态。在此期间无需发送任何指令。电源引脚(VDD, GND)之间可增加一个 100nF 的电容,用以去耦滤波。

2.2 串行接口 (两线双向)

SHTxx 的串行接口,在传感器信号的读取及电源功耗方面,都做了优化处理;但与 I²C 接口并不兼容,详情参见 FAQ。

2.2.3 发送命令

用一组“启动传输”时序,来表示数据传输的初始化。它包括:当 SCK 时钟高电平时 DATA 翻转为低电平,紧接着 SCK 变为低电平,随后是在 SCK 时钟高电平时 DATA 翻转为高电平。



图 3 “启动传输”时序

后续命令包含三个地址位（目前只支持“000”），和五个命令位。SHTxx 会以下述方式表示已正确地接收到指令：在第 8 个 SCK 时钟的下降沿之后，将 DATA 下拉为低电平（ACK 位）。在第 9 个 SCK 时钟的下降沿之后，释放 DATA（恢复高电平）。

命令	代码
预留	0000x
温度测量	00011
湿度测量	00101
读状态寄存器	00111
写状态寄存器	00110
预留	0101x-1110x
软复位, 复位接口、清空状态寄存器, 即清空为默认值 下一次命令前等待至少 11ms	11110

表 2 SHTxx 命令集

2.2.4 测量时序(RH 和 T)

2.2.1 串行时钟输入 (SCK)

SCK 用于微处理器与 SHTxx 之间的通讯同步。由于接口包含了完全静态逻辑,因此不存在最小 SCK 频率。

2.2.2 串行数据 (DATA)

DATA 三态门用于数据的读取。DATA 在 SCK 时钟下降沿之后改变状态,并仅在 SCK 时钟上升沿有效。数据传输期间,在 SCK 时钟高电平时,DATA 必须保持稳定。为避免信号冲突,微处理器应驱动 DATA 在低电平。需要一个外部的上拉电阻(例如:10kΩ)将信号提拉至高电平(参见图 2)。上拉电阻通常已包含在微处理器的 I/O 电路中。详细的 IO 特性,参见表 5。

数据。检测数据可以先被存储,这样控制器可以继续执行其它任务在需要时再读出数据。

接着传输 2 个字节的测量数据和 1 个字节的 CRC 奇偶校验。uC 需要通过下拉 DATA 为低电平,以确认每个字节。所有的数据从 MSB 开始,右值有效(例如:对于 12bit 数据,从第 5 个 SCK 时钟起算作 MSB;而对于 8bit 数据,首字节则无意义)。

用 CRC 数据的确认位,表明通讯结束。如果不使用 CRC-8 校验,控制器可以在测量值 LSB 后,通过保持确认位 ack 高电平,来中止通讯。

在测量和通讯结束后, SHTxx 自动转入休眠模式。

警告: 为保证自身温升低于 0.1℃, SHTxx 的激活时间不要超过 10%(例如,对应 12bit 精度测量,每秒最多进行 2 次测量)。

2.2.5 通讯复位时序

如果与 SHTxx 通讯中断,下列信号时序可以复位串口:

当 DATA 保持高电平时,触发 SCK 时钟 9 次或更多。在下一次指令前,发送一个“传输启动”时序。这些时序只复位串口,状态寄存器内容仍然保留。

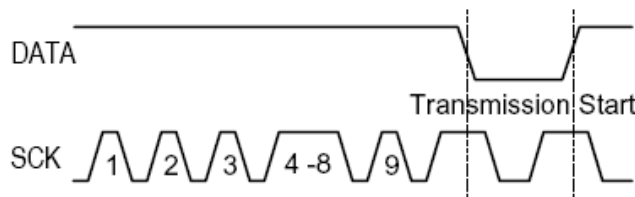


图 4 通讯复位时序

2.2.4 测量时序(RH 和 T)

发布一组测量命令（‘00000101’表示相对湿度 RH，‘00000011’表示温度 T）后，控制器要等待测量结束。这个过程需要大约 20/80/320ms，分别对应 8/12/14bit 测量。确切的时间随内部晶振速度，最多可能有-30%的变化。SHTxx 通过下拉 DATA 至低电平并进入空闲模式，表示测量的结束。控制器在再次触发 SCK 时钟前，必须等待这个“数据备妥”信号来读出

2.2.6 CRC-8 校验

数字信号的整个传输过程由 8bit 校验来确保。任何错误数据将被检测到并清除。

详情可参阅应用说明“CRC-8 校验”。

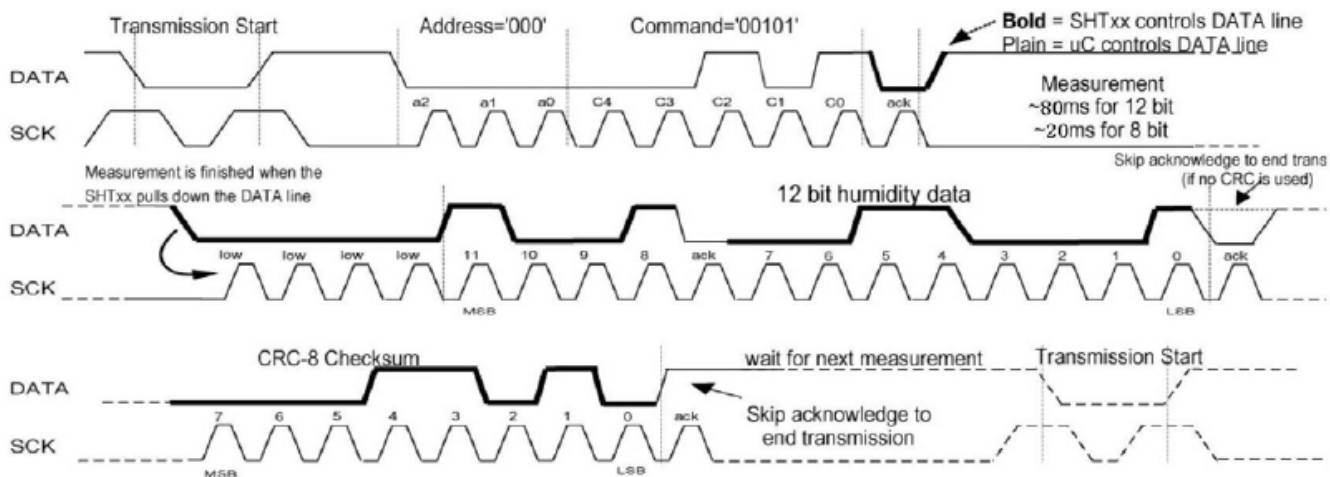


图 5 RH 测量时序举例：“0000’1001’0011’0001”= 2353 = 75.79 %RH (未包含温度补偿)

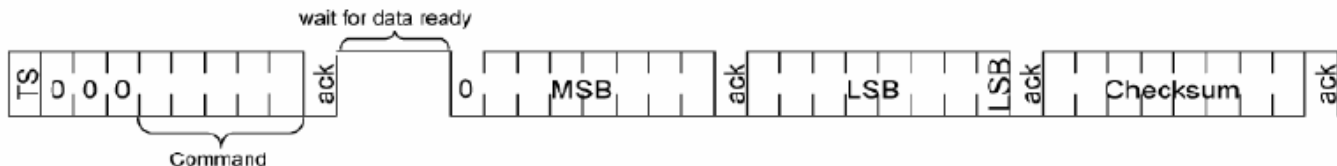
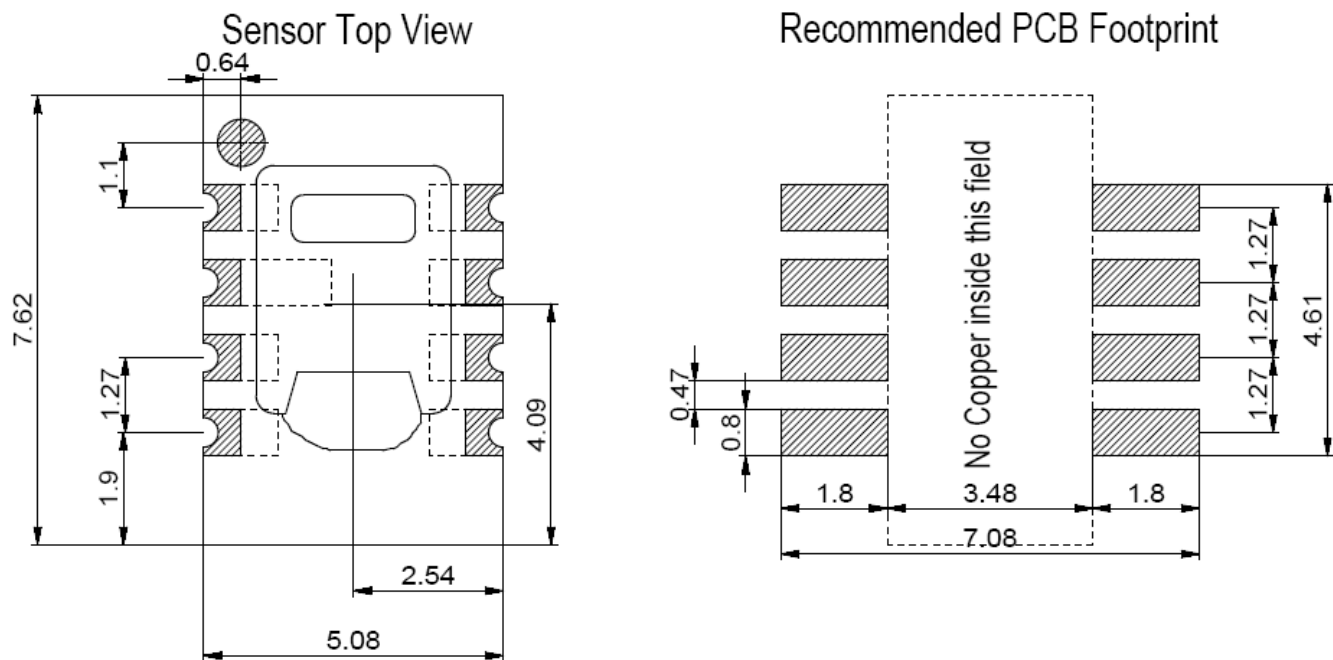


图 6

测量时序概览 (TS = 启动传输)

外形尺寸



焊接说明

1 说明

SHT1X 温湿度传感器在焊接过程中和焊接之后需要作特殊处理，以避免对温湿度敏感元件造成损坏。

SHT1X 温湿度传感器可以在 260℃温度下 30 秒之内进行标准回流焊接（包括无铅焊接）。

2 焊接规程

2.1 焊锡膏

请使用免清洗焊锡膏

2.2 建议焊接条件

注：以上所有温度均指传感器封装顶部温度，测量时请测传感器件的上表面。

	回流焊或红外 回流焊	VPR
温度平均上升速度（183℃到峰值）	最大 4℃/秒	最大 10℃/秒
温度预热时间 160（±10）℃	最大 90 秒	
温度保持 183℃以上	60 秒	
温度与实际峰值相差 5℃以内所需时间	30 秒	60 秒
峰值	260℃（无铅焊接） ¹	233℃（无铅焊接） ²
温度下降时间	最大 6℃/秒	最大 10℃/秒
25℃到峰值时间	最大 6 分钟	

2.3 PCB 清洗

传感器焊接之后请不要清洗PCB（印刷电路板），由于使用免清洗焊锡膏，清洗不是必需步

骤。

2.4 焊接后的处理

传感器焊接之后,湿度显示会出现暂时性改变,这是由于焊接时极高的温度引起的。因此,焊接之后的传感器需要在湿度>74%RH, 温度>20℃的条件下放置至少48 小时,促使其快速地恢复到正常状态。

2.5 手工焊接

手工焊接限制最高温度为350℃, 接触时间不得超过5 秒。

2.6 焊接质量

为保证传感器与PCB 板连接牢固,焊锡应填满至顶部。

2.7 注意

焊接过程中电容性湿敏元件与挥发性溶剂接触,可能会导致湿度暂时性漂移。

只需将传感器加热到100 ℃ 保持一天,可使其恢复到校准状态。

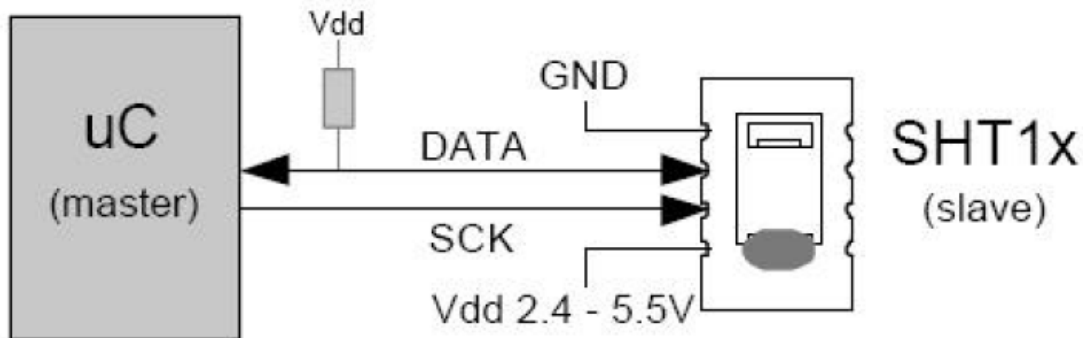
1 编号为144 之前的产品(2004 年14 周)峰值为235℃

2 编号为144 之前的产品(2004 年14 周)峰值为219℃

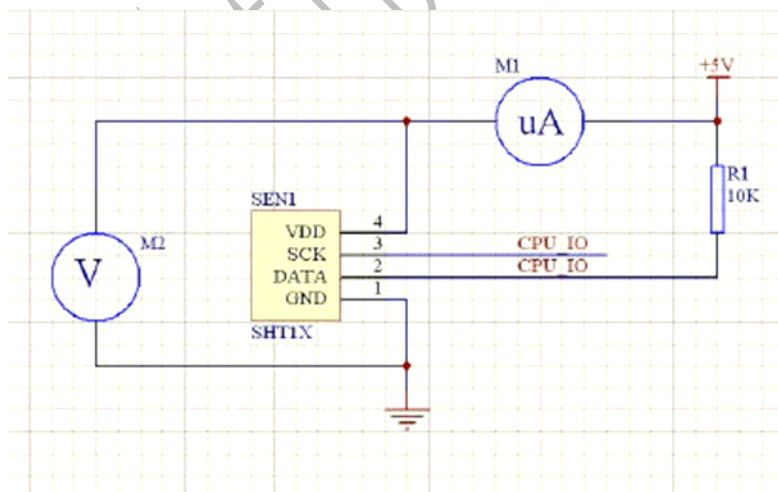
温湿度传感器 SHT 系列功耗的正确测量方法

(以下方法由总部 测试工程师:梁书成提供)

正确的功耗测量方法是传感器在正确的工作状态下的测量,我们传感器的正确接法如下:



在这个基础上我们的功耗测试的连接应该如下:



但这只是硬件的基本连接,这并不能代表我们的传感器就已经在正确的工作状态了。

Timing diagram for SPI transmission of a CRC-8 checksum. The diagram shows two signals: DATA and SCK. The DATA signal is high for the first 8 clock cycles (bits 7 to 0) and then goes high again. The SCK signal is a periodic clock. The first 8 clock cycles correspond to the 8 bits of the CRC-8 checksum. The 9th clock cycle is labeled 'ack' and is followed by a period labeled 'wait for next measurement'. An arrow points to the 'ack' cycle with the text 'Skip acknowledge to end transmission'.

SHTxx 驱动程序



The image shows a white, rectangular SLIMT-1 Humidity And Temperature Monitor. The device features a blue display area with two rows of seven-segment displays. The top row shows temperature in degrees Celsius (°C) and the bottom row shows humidity in percent (%). Below the displays, there are four status LEDs labeled ALM, OUT1, OUT2, and SNO1. At the bottom of the device, there are four buttons: a red button with a power symbol, a blue button labeled SET, a blue button with an up arrow, a blue button with a left arrow, and a blue button labeled ENT. The device is shown against a blue background.

*****/

```
typedef union
{ unsigned int i;
  float f;
} value;
```



```
//-----  
// modul-var  
//-----  
enum {TEMP,HUMI};  
  
#define DATA      P1_1  
#define SCK        P1_0  
  
#define noACK 0  
#define ACK      1  
  
                //adr  command  r/w  
#define STATUS_REG_W 0x06  //000   0011   0  
#define STATUS_REG_R 0x07  //000   0011   1  
#define MEASURE_TEMP 0x03  //000   0001   1  
#define MEASURE_HUMI 0x05  //000   0010   1  
#define RESET        0x1e  //000   1111   0  
  
//-----  
char s_write_byte(unsigned char value)  
//-----  
// writes a byte on the Sensibus and checks the acknowledge  
{  
    unsigned char i,error=0;  
    for (i=0x80;i>0;i/=2)        //shift bit for masking  
    { if (i & value) DATA=1;      //masking value with i , write to SENSI-BUS  
      else DATA=0;  
      SCK=1;                      //clk for SENSI-BUS  
      _nop();_nop();_nop();        //pulswith approx. 5 us  
      SCK=0;  
    }  
    DATA=1;                      //release DATA-line  
    SCK=1;                        //clk #9 for ack  
    error=DATA;                   //check ack (DATA will be pulled down by SHT11)  
    SCK=0;  
    return error;                 //error=1 in case of no acknowledge  
}  
  
//-----  
char s_read_byte(unsigned char ack)  
//-----  
// reads a byte form the Sensibus and gives an acknowledge in case of "ack=1"  
{
```

```
unsigned char i,val=0;
DATA=1;                                //release DATA-line
for (i=0x80;i>0;i/=2)                  //shift bit for masking
{ SCK=1;                                //clk for SENSI-BUS
  if (DATA) val=(val | i);              //read bit
  SCK=0;
}
DATA=!ack;                              //in case of "ack==1" pull down DATA-Line
SCK=1;                                  //clk #9 for ack
_nop_();_nop_();_nop_();                //pulswith approx. 5 us
SCK=0;
DATA=1;                                //release DATA-line
return val;
}
```

```
//-----
```

```
void s_transstart(void)
```

```
//-----
```

```
// generates a transmission start
```

```
// _____
```

```
// DATA:  |_____|
```

```
// _____
```

```
// SCK :  |_|  |_|  |_|
```

```
{
  DATA=1; SCK=0;                        //Initial state
```

```
  _nop_();
```

```
  SCK=1;
```

```
  _nop_();
```

```
  DATA=0;
```

```
  _nop_();
```

```
  SCK=0;
```

```
  _nop_();_nop_();_nop_();
```

```
  SCK=1;
```

```
  _nop_();
```

```
  DATA=1;
```

```
  _nop_();
```

```
  SCK=0;
```

```
}
```

```
//-----
```

```
void s_connectionreset(void)
```

```
//-----
```

```
// communication reset: DATA-line=1 and at least 9 SCK cycles followed by transstart
```

```
//transmission start

-----

reset

//reset communication
SET); //send RESET-command to sensor
//error=1 in case of no response form the sensor

-----

igned char *p_value, unsigned char *p_checksum)

-----

with checksum (8-bit)

//transmission start
TUS_REG_R); //send command to sensor
CK); //read status register (8-bit)
```

```
unsigned char error=0;
s_transstart();           //transmission start
error+=s_write_byte(STATUS_REG_W);//send command to sensor
error+=s_write_byte(*p_value); //send value of status register
return error;             //error>=1 in case of no response form the sensor
}

//-----
char s_measure(unsigned char *p_value, unsigned char *p_checksum, unsigned char mode)
//-----
// makes a measurement (humidity/temperature) with checksum
{
    unsigned error=0;
    unsigned int i;

    s_transstart();           //transmission start
    switch(mode){             //send command to sensor
        case TEMP : error+=s_write_byte(MEASURE_TEMP); break;
        case HUMI  : error+=s_write_byte(MEASURE_HUMI); break;
        default    : break;
    }
    for (i=0;i<65535;i++) if(DATA==0) break; //wait until sensor has finished the measurement
    if(DATA) error+=1;         // or timeout (~2 sec.) is reached
    *(p_value) =s_read_byte(ACK); //read the first byte (MSB)
    *(p_value+1)=s_read_byte(ACK); //read the second byte (LSB)
    *p_checksum =s_read_byte(noACK); //read checksum
    return error;
}

//-----
void init_uart()
//-----
//9600 bps @ 11.059 MHz
{SCON  = 0x52;
  TMOD  = 0x20;
  TCON  = 0x69;
  TH1    = 0xfd;
}

//-----
void calc_sth11(float *p_humidity ,float *p_temperature)
//-----
// calculates temperature [度] and humidity [%RH]
```



```
// input : humi [Ticks] (12 bit)
//          temp [Ticks] (14 bit)
// output: humi [%RH]
//          temp [度]

{ const float C1=-4.0;           // for 12 Bit
  const float C2=+0.0405;        // for 12 Bit
  const float C3=-0.0000028;     // for 12 Bit
  const float T1=+0.01;          // for 14 Bit @ 5V
  const float T2=+0.00008;       // for 14 Bit @ 5V

  float rh=*p_humidity;          // rh: Humidity [Ticks] 12 Bit
  float t=*p_temperature;         // t: Temperature [Ticks] 14 Bit
  float rh_lin;                   // rh_lin: Humidity linear
  float rh_true;                  // rh_true: Temperature compensated humidity
  float t_C;                      // t_C : Temperature

  t_C=t*0.01 - 40;                //calc. temperature from ticks
  rh_lin=C3*rh*rh + C2*rh + C1;    //calc. humidity from ticks to [%RH]
  rh_true=(t_C-25)*(T1+T2*rh)+rh_lin; //calc. temperature compensated humidity [%RH]
  if(rh_true>100)rh_true=100;      //cut if the value is outside of
  if(rh_true<0.1)rh_true=0.1;     //the physical possible range

  *p_temperature=t_C;              //return temperature
  *p_humidity=rh_true;             //return humidity[%RH]
}

//-----
float calc_dewpoint(float h,float t)
//-----
// calculates dew point
// input: humidity [%RH], temperature [度]
// output: dew point
{ float logEx,dew_point;
  logEx=0.66077+7.5*t/(237.3+t)+(log10(h)-2);
  dew_point = (logEx - 0.66077)*237.3/(0.66077+7.5-logEx);
  return dew_point;
}

//-----
void main()
//-----
// sample program that shows how to use SHT11 functions
// 1. connection reset
```

```
// 2. measure humidity [ticks](12 bit) and temperature [ticks](14 bit)
// 3. calculate humidity [%RH] and temperature
// 4. calculate dew point
// 5. print temperature, humidity, dew point

{ value humi_val,temp_val;
  float dew_point;
  unsigned char error,checksum;
  unsigned int i;

  init_uart();
  s_connectionreset();
  while(1)
  { error=0;
    error+=s_measure((unsigned char*) &humi_val.i,&checksum,HUMI); //measure humidity
    error+=s_measure((unsigned char*) &temp_val.i,&checksum,TEMP); //measure temperature
    if(error!=0) s_connectionreset(); //in case of an error: connection reset
    else
    { humi_val.f=(float)humi_val.i; //converts integer to float
      temp_val.f=(float)temp_val.i; //converts integer to float
      calc_sth11(&humi_val.f,&temp_val.f); //calculate humidity, temperature
      dew_point=calc_dewpoint(humi_val.f,temp_val.f); //calculate dew point
      printf("temp:%5.1fC humi:%5.1f%% dew point:%5.1fC\n",temp_val.f,humi_val.f,dew_point);
    }
    //-----wait approx. 0.8s to avoid heating up SHTxx-----
    for (i=0;i<40000;i++); // (be sure that the compiler doesn't eliminate this line!)
    //-----
  }
}
```

因为专业，所以更好

SONBest**搜博仪器**

本公司专业代理销售 SENSIRION（盛世瑞恩）系列传感器。

联系电话：021-51083595

网址：<http://www.sonBest.com>

地址：上海市黄浦区广西北路 668 号 1604 室

展销部地址：

展销部电话：021-61209285 网址：<http://www.dwn.com.cn>

地址：上海市北京东路 668 号上海赛格电子市场 G537 室

SONBest地址：上海市广西北路528号安基大厦1604室
电话：021-51083595 总部：<http://www.sonbest.com>