深圳市捷迅易联科技有限公司
shenzhen jiexun yilian technology co.,ltd

YL-18HC 传感器模块--产品手册

·I²C 数字温度和湿度传感器·

电话：**0755-26031631　0755-26521631**

传真：**0755-26031631**

Email：**nog2004@126.com**

网址：**http://www.rf-module.cn**

地址：深圳市南山区高新科技园中区科智西路1号科苑西工业区23栋6楼

目 录

1 特点 ····· 3

2 应用 ····· 5

3 描述 ····· 5

4 I²C 接口 ····· 5

4.1 I²C 接口 ····· 5

4.1.1 相对湿度测量 ····· 5

4.1.2 温度测量 ····· 6

4.1.3 正常模式和快速模式 ····· 7

4.1.4 加热 ····· 8

4.1.5 设备标识 ····· 8

4.2 I²C 操作 ····· 8

4.2.1 I²C 写操作 ····· 8

4.2.2 I²C 读操作 ····· 9

4.2.3 I²C 接口时序 ····· 10

5 控制寄存器 ····· 12

5.1 STATUS 寄存器 ····· 12

5.2 DATAh 寄存器 ····· 13

5.3 DATAl 寄存器 ····· 13

5.4 CONFIG 寄存器 ····· 13

5.5 ID 寄存器 ····· 14

6 引脚说明 ····· 15

7 程序举例 ····· 16

一、特点

1. 工作电压：（1）2.1 V≤V_{DD}≤3.6 V（推荐值）

（2）V_{DD}=3.3V（典型值）

工作范围：（1）温度—— -40 °C≤T_A≤85 °C（推荐值）

（2）湿度—— 0 - 100 %RH

转换时间：（1）正常模式——t_{CONV} = 35 ms（典型值），t_{CONV} = 40 ms（最大值）

（2）快速模式——t_{CONV} = 18ms（典型值），t_{CONV} = 21 ms（最大值）

唤醒时间：（1）10ms（典型值） （2）15ms（最大值）

上电时间：（1）10ms（典型值） （2）15ms（最大值）

2. 相对湿度传感器

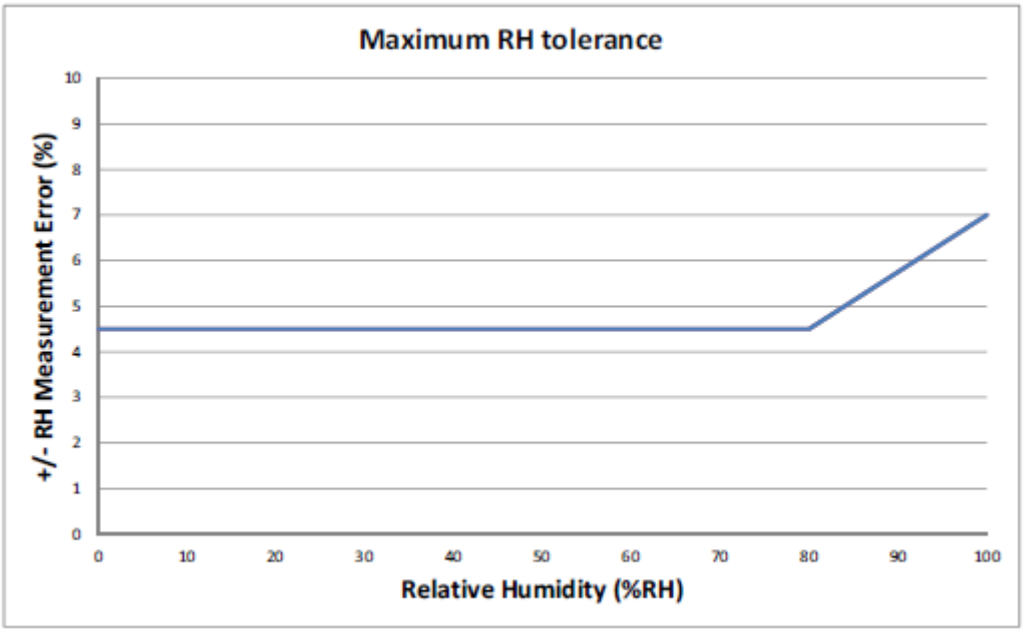
测试条件：20 - 80 %RH，T_A=25°C，t_{CONV} = 35 ms。

（1）精度：±3.0 %RH（典型值）

±4.5 %RH（最大值）

（2）响应时间：8s（典型值）

图 1-1 RH 精度的最大值（@30℃）



3. 温度传感器

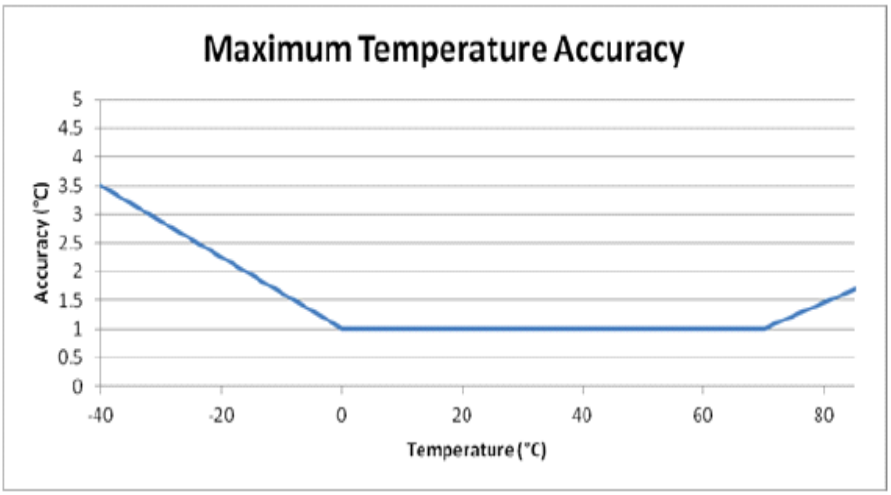
测试条件：0≤TA≤70°C，tCONV = 35 ms。

（1）精度：±0.5°C（典型值）

±1°C（最大值）

（2）响应时间：1.5s（典型值）

图 1-2　温度精度的最大值



4. 低功耗：240 μA（RH 转换期间）

5. I²C 主机接口

6. 集成芯片加热器

7. 4 mm x 4 mm QFN 封装

8. 优异的长期稳定性

9. 工厂校准

10. 可选的工厂安装防护罩

（1）薄型

（2）回流保护

（3）排除液体和颗粒物（疏水/疏油）

二、应用

1. 工业 HVAC/R　　2. 恒温器/恒湿器

3. 呼吸治疗　　4. 白色家电

5. 微环境/数据中心　　6. 汽车气候控制和去雾

7. 资产和物品追踪　　8. 气体、火和烟雾探测器

9. 洪水和水检测器　　10. 远程遥测装置

11. 通风和空调系统　　12. 无线传感器网络

三、描述

YL_18HC 是一个数字式相对湿度和温度传感器。它集成了温度和湿度传感器、AD 转换器、信号处理、数据校准和一个 I²C 主机接口。

四、I²C 接口

4.1 I²C 接口

YL_18HC 是一个从设备，具有 I2C 串行接口和 7 位地址 0x40，它支持的数据传输速率高达 400kHz。

4.1.1 相对湿度测量

1. 设置 CONFIG 寄存器（0x03）中的 START（D0）和 TEMP（D4），开始 RH(相对湿度)转换（START=1，TEMP=0，即 CONFIG=0x01）；
2. 检测 STATUS 寄存器（0x00）中的 RDY（D0），直到转换完成（RDY=0，即 STATUS=0x00）；
3. 读取 DATAh 寄存器（0x01）和 DATAl 寄存器（0x02）中的数据，RH 值的高字节和低字节存放在 DATAh：DATAl。

4. %RH 的计算公式：

$$\%RH=(\frac{RH}{16})\text{-}24$$

表 4-1 RH 值（12-Bit）存放在 DATAh 和 DATAl（正常模式，tCONV = 35 ms）

DATAh								DATAl							
D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
12-Bit Relative Humidity Code												0	0	0	0

表 4-2 RH 值（12-Bit）的典型代码（0 – 100 %RH）

%RH	12 Bit Code	
	Dec	Hex
0	384	180
10	544	220
20	704	2C0
30	864	360
40	1024	400
50	1184	4A0
60	1344	540
70	1504	5E0
80	1664	680
90	1824	720
100	1984	7C0

注：快速模式，tCONV = 18 ms，RH 值为 11-Bit。

4.1.2 温度测量

1. 设置 CONFIG 寄存器（0x03）中的 START（D0）和 TEMP（D4），开始 TEMP (温度)转换（START=1，TEMP=1，即 CONFIG=0x11）；
2. 检测 STATUS 寄存器（0x00）中的 RDY（D0），直到转换完成（RDY=0，即 STATUS=0x00）；
3. 读取 DATAh 寄存器（0x01）和 DATAl 寄存器（0x02）中的数据，TEMP 值的高字节和低字节存放在 DATAh：DATAl。

4. Temperature(℃)的计算公式：

$$\text{Temperature} = (\frac{TEMP}{32}) - 50$$

表 4-3 TEMP 值（14-Bit）存放在 DATAh 和 DATAl（正常模式，tCONV = 35 ms）

DATAh								DATAl							
D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
14-Bit Temperature Code														0	0

表 4-4 TEMP 值（14-Bit）的典型代码（－40 ℃≤TA≤85 ℃）

%RH	14 Bit Code	
	Dec	Hex
–40	320	0140
–30	640	0280
–20	960	03C0
–10	1280	0500
0	1600	0640
10	1920	0780
20	2240	08C0
30	2560	0A00
40	2880	0B40
50	3200	0C80
60	3520	0DC0
70	3840	0F00
80	4160	1040
90	4480	1180
100	4800	12C0

注：快速模式，tCONV = 18 ms，TEMP 值为 13 Bit。

4.1.3 正常模式和快速模式

设置 CONFIG 寄存器（0x03）中的 FAST（D5），FAST=0 时为正常模式，FAST=1 时为快速模式。tCONV 为转换时间。

表 4-5　正常模式和快速模式

Parameter	Value	
	Normal Mode	Fast Mode
tCONV(typical)	35 ms	18 ms
Temperature resolution	14-bit	13-bit
RH resolution	12-bit	11-bit

注：正常模式，tCONV(maximum) = 40 ms；快速模式，tCONV(maximum) = 21 ms。

4.1.4 加热

设置 CONFIG 寄存器（0x03）中的 HEAT（D1），HEAT=1 时，启用加热模式。该模式用于提高湿度传感器的温度，进而抵消持续的高湿度条件下的“湿度记忆”。不过，温度传感器的读数会受到影响（温度升高）。

4.1.5 设备标识

ID 寄存器（0x11）中的值为 0x50。通过读取 ID，可以检测设备是否正常工作。

4.2 I²C 操作

表 4-6　I²C 从机地址字节

A6	A5	A4	A3	A2	A1	A0	R/W
1	0	0	0	0	0	0	1/0

4.2.1 I²C 写操作

主机向从机的寄存器写入数据：

1. 主机发出 I²C 启动命令（S）；
2. 主机向从机发送 8 bit 从机地址字节，从机地址只有 7bit（Slave Address=0x40），第 8 位是读/写标记，0 表示写（W=0）；
3. 从机一旦识别它的从机地址，就向主机发送应答信号（A=0）；
4. 主机向从机发送 8 bit 寄存器地址（Address Pointer），选择要写入数据的寄存器；

5. 从机一旦识别这个寄存器地址，就向主机发送应答信号（A=0）；

6. 主机向从机的寄存器写入 8 bit 数据（Register Data）；

7. 从机一旦接收完 1 字节数据，就向主机发送应答信号（A=0）；

8. 主机发出 I²C 停止命令（P）。

表 4-7　I²C 写时序

Master	Slave

Sequence to Write to a Register

S	Slave Address	W	A	Address Pointer	A	Register Data	A	P

Sequence to Start a Relative Humidity Conversion

S	0x40	0	A	0x03	A	0x01	A	P

Sequence to Start a Temperature Conversion

S	0x40	0	A	0x03	A	0x11	A	P

4.2.2 I²C 读操作

主机向从机的寄存器读出数据：

1. 主机发出 I²C 启动命令（S）；

2. 主机向从机发送 8 bit 从机地址字节，从机地址只有 7bit（Slave Address=0x40），

   第 8 位是读/写标记，0 表示写（W=0）；

3. 从机一旦识别它的从机地址，就向主机发送应答信号（A=0）；

4. 主机向从机发送 8 bit 寄存器地址（Address Pointer），选择要读出数据的寄存器；

5. 从机一旦识别这个寄存器地址，就向主机发送应答信号（A=0）；

6. 主机重新发出 I²C 启动命令（S）；

7. 主机向从机发送 8 bit 从机地址字节，从机地址只有 7bit（Slave Address=0x40），

   第 8 位是读/写标记，1 表示读（R=1）；

8. 从机一旦识别它的从机地址，就向主机发送应答信号（A=0）；

9. 主机向从机的寄存器读取 8 bit 数据（Register Data）；

10. 主机一旦接收完 1 字节数据，就向从机发送非应答信号（A=0）；

11. 主机接收完最后 1 字节数据，就向从机发送非应答信号（$\overline{A}$=1）；

12. 主机发出 I²C 停止命令（P）。

表 4-8　I²C 读时序（单个寄存器）

Sequence to Read from a Single Register

S	Slave Address	W	A	Address Pointer	A	Sr	Slave Address	R	A	Register Data	$\overline{A}$	P

Sequence to Read Device ID

S	0X40	0	A	0x11	A	Sr	0x40	1	A	ID	$\overline{A}$	P

Sequence to Read $\overline{RDY}$ bit

S	0X40	0	A	0x00	A	Sr	0x40	1	A	—	$\overline{RDY}$	$\overline{A}$	P

表 4-9　I²C 读时序（多个寄存器）

I²C Read Sequence for RH or Temperature Conversion Result

S	Slave Address	W	A	Address Pointer	A	Sr	Slave Address	R	A	Register 1 Data	A	Register 2 Data	$\overline{A}$	P

Sequence to Read Conversion Result

S	0X40	0	A	0X01	A	Sr	0X40	1	A	Datah	A	Datal	$\overline{A}$	P

4.2.3 I²C 接口时序

1. I²C 总线空闲时，时钟线 SCK 和数据线 SDA 必须保持高电平（SCK=1，SDA=1）。

2.（1）I²C 总线上的时钟线 SCK 在高电平期间（SCK=1），数据线 SDA 必须保持稳定；

（2）时钟线 SCK 在低电平期间（SCK=0），数据线 SDA 的高电平或低电平才允许变化。

3.（1）时钟信号 SCK 保持在高电平期间（SCK=1），数据信号 SDA 由高电平（SDA=1）向低电平（SDA=0）变化定义为起始信号（S）；

（2）时钟信号 SCK 保持在高电平期间（SCK=1），数据信号 SDA 由低电平（SDA=0）向高电平（SDA=1）变化定义为终止信号（P）；

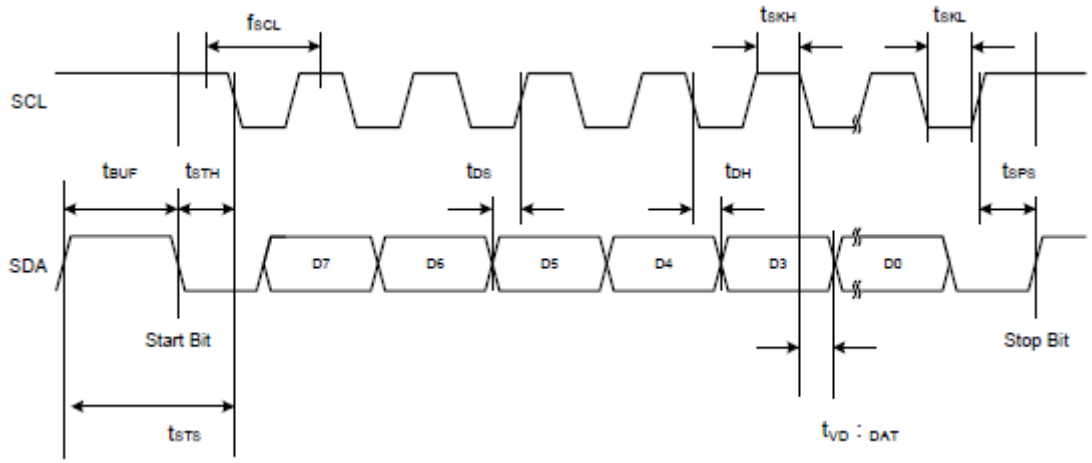
4.（1）主机向从机写入数据时，从机必须向主机发送低电平作为应答信号（A=0）。此时要求主机释放数据线 SDA，以便从机送出应答信号，将数据线 SDA 拉成低电平。

（2）主机向从机读取数据时，在最后一个字节，主机必须向从机发送高电平作为非应答信号（$\overline{A}$=1），使得从机释放数据线 SDA，以便主机产生一个终止信号。

表 4-10　I^2C 接口规格

参数	符号	测试条件	最小值	典型值	最大值	单位
SCL 频率	f_{SCL}		—	—	400	kHz
SCL 高电平时间	t_{SKH}		0.6	—	—	μs
SCL 低电平时间	t_{SKL}		1.3	—	—	μs
开始保持时间	t_{STH}		0.6	—	—	μs
启动建立时间	t_{STS}		1.9	—	—	μs
停止建立时间	t_{SPS}		0.6	—	—	μs
总线空闲时间	t_{BUF}	停止和启动之间	1.3	—	—	μs
SDA 建立时间	t_{DS}		100	—	—	ns
SDA 保持时间	t_{DH}		100	—	—	ns
SDA 有效时间	$t_{VD;DAT}$	从 SCL 低到数据有效	—	—	0.9	μs
SDA 确认有效时间	$t_{VD;ACK}$	从 SCL 低到数据有效	—	—	0.9	μs

图 4-1　I²C 接口时序图



五、控制寄存器

表 5-1　寄存器汇总

Register Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
I²C Register Summary									
0x00	STATUS	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	/RDY
0x01	DATAh	Relative Humidity or Temperature, High Byte							
0x02	DATAl	Relative Humidity or Temperature, Low Byte							
0x03	CONFIG	RSVD	RSVD	FAST	TEMP	RSVD	RSVD	HEAT	START
0x11	ID	ID3	ID2	ID1	ID0	0	0	0	0
注：保留寄存器位 RSVD 必须始终写成 0，对其读的结果是未定义的。									

5.1 STATUS 寄存器

表 5-2　STATUS 寄存器

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name								/RDY
Type	Read							
Reset Settings = 0000_0001								

Bit	Name	Function
7:1	保留位	读取的值未定义
0	/RDY	“准备好”状态。 /RDY=1：转换正在进行 /RDY=0：转换完成

5.2 DATAh 寄存器

表 5-3 DATAh 寄存器

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	Relative Humidity or Temperature, High Byte							
Type	Read							
Reset Settings = 0000_0000								

Bit	Name	Function
7:0	DATAh	数据高字节。 存放“相对湿度”和“温度”的转换值

5.3 DATAl 寄存器

表 5-4 DATAl 寄存器

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	Relative Humidity or Temperature, Low Byte							
Type	Read							
Reset Settings = 0000_0000								

Bit	Name	Function
7:0	DATAl	数据低字节。 存放“相对湿度”和“温度”的转换值

5.4 CONFIG 寄存器

表 5-5 CONFIG 寄存器

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name			FAST	TEMP			HEAT	START
Type	Read/Write							
Reset Settings = 0000_0000								

Bit	Name	Function
7:6	保留位	读取的值未定义，总是写入 0
5	FAST	“快速模式”使能。 FAST **=0**：转换时间tCONV为35ms (典型) FAST =1：转换时间 tCONV 为 18ms (典型)
4	TEMP	“温度”使能。 TEMP **=0**：湿度转换 TEMP =1：温度转换
3:2	保留位	读取的值未定义，总是写入 0
1	HEAT	“加热”使能。 HEAT **=0**：加热关闭 HEAT =1：加热打开
0	START	“转换”使能。 START **=0**：不启动转换 START =1：启动转换

5.5 ID 寄存器

表 5-5 ID 寄存器

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
Type	Read							
Reset Settings = 0101_0000								

Bit	Name	Function
7:0	ID	设备标识。

六、引脚说明

表 6-1　尺寸描述

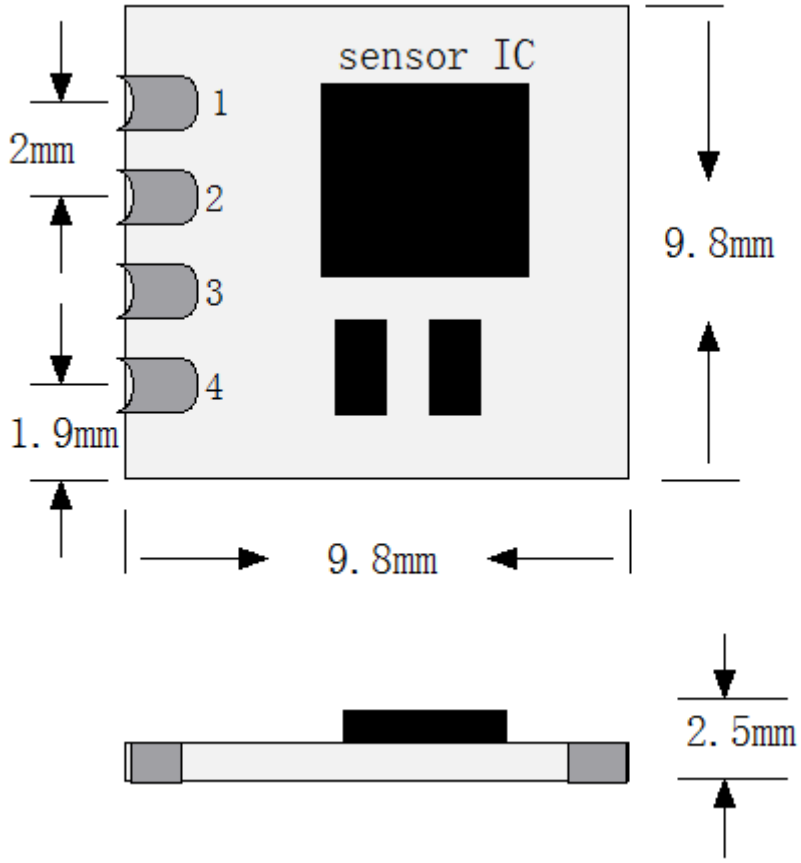


表 6-1　引脚描述

Pin #	Pin Name	Pin Type	Description
1	GND	Ground	接地
2	SCL	Input	I²C 时钟信号
3	SDA	Input/Output	I²C 数据信号
4	V_{DD}	Power	电源

参数	符号	测试条件	最小值	典型值	最大值	单位
输入高电压	V_{IH}	SCL,SDA 引脚	0.7xV_{DD}	—	—	V
输入低电压	V_{IL}	SCL,SDA 引脚	—	—	0.3xV_{DD}	V
输入电压范围	V_{IN}	SCL,SDA 引脚	0.0	—	3.6	V
输入漏电流	I_{IL}	SCL,SDA 引脚	—	—	±1	μA
输出低电压	V_{OL}	SDA 引脚; I_{OL}=8.5mA; V_{DD}=3.3 V	—	—	0.6	V
		SDA 引脚; I_{OL}=3.5mA; V_{DD}=2.1 V	—	—	0.4	V

七、程序举例

STM8 单片机通过端口模拟 I² C 协议与 YL-18HC 进行通信，读取温度和相对湿度数据。

```
#include "stm8l10x.h"

#define SDA_H       GPIO_SetBits(GPIOC, GPIO_Pin_0)
#define SDA_L       GPIO_ResetBits(GPIOC, GPIO_Pin_0)
#define SCK_H       GPIO_SetBits(GPIOC, GPIO_Pin_1)
#define SCK_L       GPIO_ResetBits(GPIOC, GPIO_Pin_1)
#define SDA       GPIO_ReadInputDataBit(GPIOC,GPIO_Pin_0)
#define SlaveAddress    0x40
#define RegisterAddress0    0x00
#define RegisterAddress1    0x01
#define RegisterAddress2    0x02
#define RegisterAddress3    0x03
#define RegisterAddress11    0x11

void CLK_INT(void);
void GPIO_INT(void);
void DELAY(uint16_t n);
void I2C_START(void);
void I2C_STOP(void);
void I2C_WRITE(uint8_t Data);
uint8_t I2C_READ(void);
void I2C_ACK(uint8_t a);
uint8_t I2C_SEND(uint8_t SlaveAdd,uint8_t RegisterAdd,uint8_t *s);
uint8_t I2C_RECEIVE(uint8_t SlaveAdd,uint8_t RegisterAdd,uint8_t *s);
void Temperature_READ(uint8_t *s);
void RelativeHumidity_READ(uint8_t *s);

uint8_t Ack;
uint8_t id;
uint8_t RelativeHumidity;
uint8_t Temperature;
uint8_t Start1=0x01;
uint8_t Start2=0x11;

void main()
{
        CLK_INT();
        GPIO_INT();
        I2C_RECEIVE(SlaveAddress,RegisterAddress11,&id);
```

```
        while (1)
        {
            RelativeHumidity_READ(&RelativeHumidity);
            Temperature_READ(&Temperature);
        }
}

void CLK_INT()
{
        CLK_DeInit();
        DELAY(500);
        CLK_CCOCmd(ENABLE);
        CLK_MasterPrescalerConfig(CLK_MasterPrescaler_HSIDiv1);
}

void GPIO_INT()
{
    GPIO_DeInit(GPIOC);
    GPIO_Init(GPIOC,GPIO_Pin_1,GPIO_Mode_Out_PP_Low_Slow);
    GPIO_Init(GPIOC,GPIO_Pin_0,GPIO_Mode_Out_OD_Low_Slow);
}

void DELAY(uint16_t n)
{
        uint8_t i;
        while(n--)
        {
           for(i=0;i<16;i++);
        }
}

void I2C_START()
{
        SDA_H;
        SCK_H;
        SDA_L;
        SCK_L;
}

void I2C_STOP()
{
        SDA_L;
        SCK_H;
        SDA_H;
```

```
}

void I2C_WRITE(uint8_t Data)
{
        uint8_t i;
        for(i=0;i<8;i++)
        {
          if(Data&0x80)
            SDA_H;
          else
            SDA_L;
          SCK_H;
          SCK_L;
       Data<<=1;
        }
        SDA_H;
        SCK_H;
        if(SDA==1)
            Ack=0;
        else
            Ack=1;
        SCK_L;
}

uint8_t I2C_READ()
{
        uint8_t Data=0;
        uint8_t i;
        SDA_H;
        for(i=0;i<8;i++)
        {
          SCK_L;
          SCK_H;
          Data<<=1;
          if(SDA==1)
            Data=Data+1;
        }
        SCK_L;
        return Data;
}

void I2C_ACK(uint8_t a)
{
        if(a==0)
```

```
        SDA_L;
    else
        SDA_H;
    SCK_H;
    SCK_L;
}

uint8_t I2C_SEND(uint8_t SlaveAdd,uint8_t RegisterAdd,uint8_t *s)
{
    SlaveAdd=SlaveAdd<<1;
    I2C_START();
    I2C_WRITE(SlaveAdd);
    if(Ack==0)
        return 0;
    I2C_WRITE(RegisterAdd);
    if(Ack==0)
        return 0;
        I2C_WRITE(*s);
        if(Ack==0)
            return 0;
    I2C_STOP();
    return 1;
}

uint8_t I2C_RECEIVE(uint8_t SlaveAdd,uint8_t RegisterAdd,uint8_t *s)
{
    SlaveAdd=SlaveAdd<<1;
    I2C_START();
    I2C_WRITE(SlaveAdd);
    if(Ack==0)
        return 0;
    I2C_WRITE(RegisterAdd);
    if(Ack==0)
        return 0;
    I2C_START();
    I2C_WRITE(SlaveAdd+1);
    if(Ack==0)
        return 0;
        *s=I2C_READ();
    I2C_ACK(1);
    I2C_STOP();
    return 1;
}
```

```
void RelativeHumidity_READ(uint8_t *s)
{
        uint8_t Status=1;
        uint8_t RelativeHumidityH;
        uint8_t RelativeHumidityL;
        uint16_t RelHum;
        I2C_SEND(SlaveAddress,RegisterAddress3,&Start1);
        while(Status==1)
        {
          I2C_RECEIVE(SlaveAddress,RegisterAddress0,&Status);
        }
        I2C_RECEIVE(SlaveAddress,RegisterAddress1,&RelativeHumidityH);
        I2C_RECEIVE(SlaveAddress,RegisterAddress2,&RelativeHumidityL);
        RelHum=RelativeHumidityH;
        RelHum=RelHum<<8;
        RelHum+=RelativeHumidityL;
        RelHum=RelHum>>4;
        *s=RelHum/16-24;
}

void Temperature_READ(uint8_t *s)
{
        uint8_t Status=1;
        uint8_t TemperatureH;
        uint8_t TemperatureL;
        uint16_t Temp;
        I2C_SEND(SlaveAddress,RegisterAddress3,&Start2);
        while(Status==1)
        {
          I2C_RECEIVE(SlaveAddress,RegisterAddress0,&Status);
        }
        I2C_RECEIVE(SlaveAddress,RegisterAddress1,&TemperatureH);
        I2C_RECEIVE(SlaveAddress,RegisterAddress2,&TemperatureL);
        Temp=TemperatureH;
        Temp<<=8;
        Temp+=TemperatureL;
        Temp>>=2;
        *s=Temp/32-50;
}
```

修订时间： 2013 年 7 月 4 日