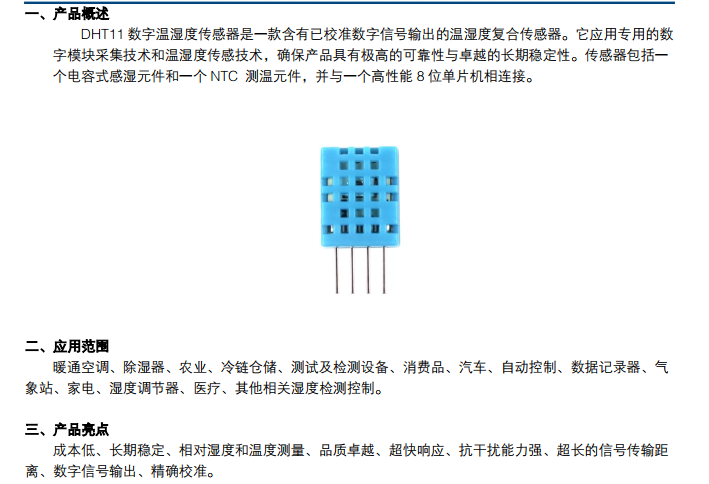
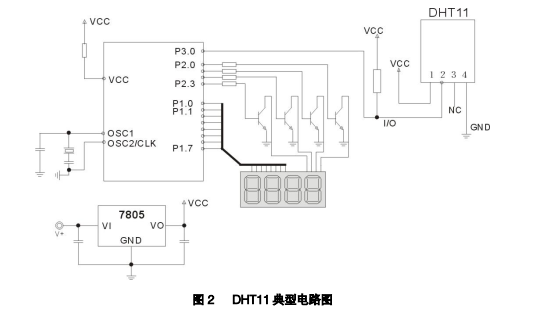
DHT11温湿度传感器

# DHT11温湿度传感器：

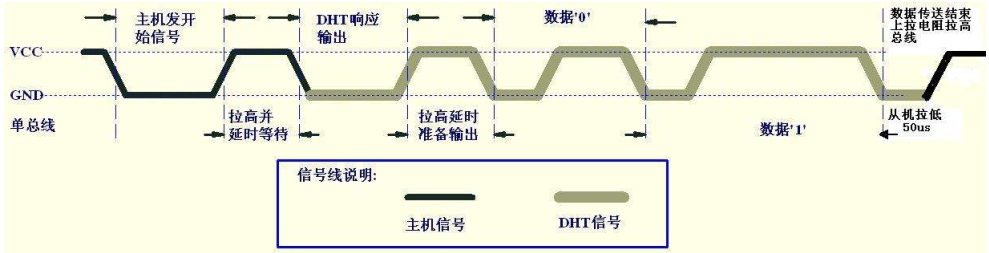


# P8913常见接线图



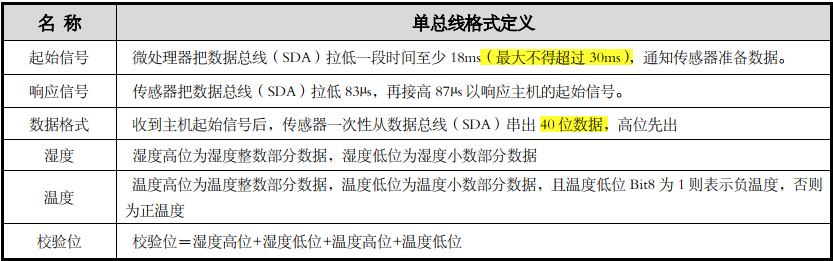
单线双线数据线DATA上拉后与微处理器的 I/O 端口相连。  
1.典型应用电路中建议连接线长度短于 5m 时用 4.7K 上拉电阻，大于 5m 时根据实际情况降低上拉电阻的阻值。  
2. 使用 3.3V 电压供电时连接线尽量短，接线过长会导致传感器供电不足，造成测量偏差。  
3. 每次读出的温湿度数值是上一次测量的结果，欲获取实时数据,需连续读取 2 次，但不建议连续多次读取传感器，每次读取传感器间隔大于 2 秒即可获得准确的数据。  
4. 电源部分如有波动，会影响到温度。如使用开关电源纹波过大，温度会出现跳动。

# 驱动时序图及工作参数



DHT11 器件采用简化的单总线通信。单总线即只有一根数据线，系统中的数据交换、控制均由单总线完成。设备（主机或从机）通过一个漏枀开路或三态端口连至该数据线，以允许设备在不发送数据时能够释放总线，而让其它设备使用总线；单总线通常要求外接一个约 4.7kΩ 的上拉电阻，这样，当总线闲置时，其状态为高电平。由于它们是主从结极，只有主机呼叫从机时，从机才能应答，因此主机访问器件都必须严格遵循单总线序列，如果出现序列混乱，器件将不响应主机。

◎单总线传送数据位定义  
DATA 用于微处理器与 DHT11 之间的通讯和同步,采用单总线数据格式，一次传送 40 位数据，高位先出。  
数据格式:  
8bit 湿度整数数据 + 8bit 湿度小数数据 + 8bit 温度整数数据 + 8bit 温度小数数据 + 8bit 校验位。  
注：其中湿度小数部分为 0。  
◎校验位数据定义  
“8bit 湿度整数数据 + 8bit 湿度小数数据 + 8bit 温度整数数据 + 8bit 温度小数数据” 8bit 校验位等于所得结果的末 8 位



示例一：接收到的 40 位数据为：  
0011 0101 0000 0000 0001 1000 0000 0100 0101 0001  
湿度高 8 位 湿度低 8 位 温度高 8 位 温度低 8 位 校验位

计算：  
0011 0101+0000 0000+0001 1000+0000 0100= 0101 0001  
接收数据正确：  
湿度： 0011 0101(整数)=35H=53%RH 0000 0000(小数)=00H=0.0%RH =>53%RH + 0.0%RH =  
53.0%RH  
温度： 0001 1000(整数)=18H=24℃ 0000 0100(小数)=04H=0.4℃ =>24℃ + 0.4℃ = 24.4℃  
◎特殊说明：  
当温度低于 0 ℃ 时温度数据的低 8 位的最高位置为 1。  
示例： -10.1 ℃ 表示为 0000 1010 1000 0001  
温度： 0000 1010(整数)=0AH=10℃， 0000 0001(小数)=01H＝0.1℃ =>-(10℃+0.1℃)＝ -10.1℃

示例二：接收到的 40 位数据为：  
0011 0101 0000 0000 0001 1000 0000 0100 0100 1001  
湿度高 8 位 湿度低 8 位 温度高 8 位 温度低 8 位 校验位  
计算：  
0011 0101+0000 0000+0001 1000+0000 0100＝ 0101 0001  
0101 0001 不等于 0100 1001  
本次接收的数据不正确，放弃，重新接收数据。

# 驱动思路

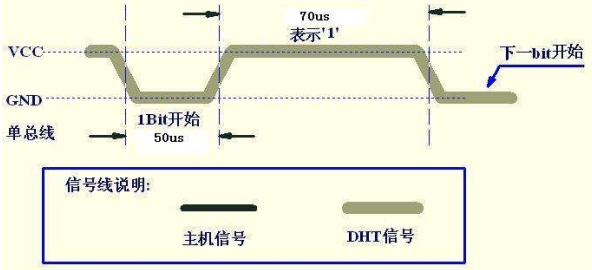
## 时序图分析

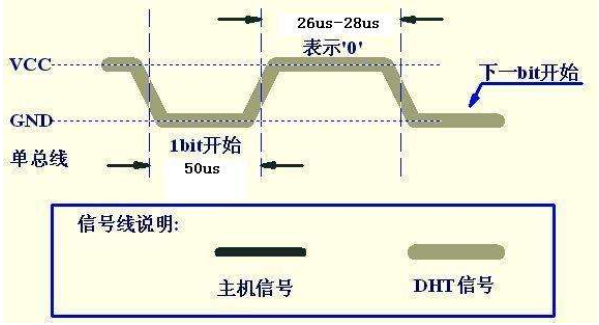
### 总线空闲状态为高电平,主机把总线拉低等待DHT11响应,主机把总线拉低必须大于18毫秒,保证DHT11能检测到起始信号。

### DHT11接收到主机的开始信号后,等待主机开始信号结束,然后发送80us低电平响应信号。主机发送开始信号结束后,延时等待20-40us后, 读取DHT11的响应信号。

### 主机发送开始信号后,可以切换到输入模式,或者输出高电平均可, 总线由上拉电阻拉高。

### 检测到总线为低电平,说明DHT11发送响应信号,DHT11发送响应信号后,再把总线拉高80us,准备发送数据,每一bit数据都以50us低电平时隙开始,高电平的长短定了数据位是0还是1.格式见下面图示，如果读取响应信号为高电平,则DHT11没有响应,请检查线路是否连接正常。当最后一bit数据传送完毕后，DHT11拉低总线50us，随后总线由上拉电阻拉高进入空闲状态。

 50us开始，持续70us表示高电平1



50us开始，持续26~28代表发送低电平0

## 发送一位数据：

MCU：

//总线空闲状态为高电平,主机把总线拉低等待DHT11响应,主机把总线拉低必须大于18毫秒,保证DHT11能检测到起始信号。

推挽输出模式：

//发送起始信号，先拉低电平，延时至少18ms再拉高

DHT11\_SEND\_DATA=0;

Delay\_ms(18);

DHT11\_SEND\_DATA=1;

//DHT11接收到主机的开始信号后,等待主机开始信号结束,然后发送80us低电平响应信号。主机发送开始信号结束后,延时等待20-40us后, 读取DHT11的响应信号。

Delay\_us(20); //延时20us

DHT11\_Input( ); //切换IO口为输入模式

u8 DHT11\_Ack(void)

{

if(DHT11\_REV\_DATA!=0) //如果没有检测到低电平的应答信号就返回0

{

return 0;

}

while(DHT11\_REV\_DATA==0);//等待80us低电平应答结束

while(DHT11\_REV\_DATA==1);//等待80us高电平应答结束

//很有必要检测高低电平，防止时序出错

return 1; 检测到高电平，说明信号不是应答信号，返回1

}

**读取从机发送过来的数据，以8位为单位接收**

u8 DHT11\_Read\_Byte(void)

{

u8 data=0;

u8 i;

//接受数据,每个数据以50us低电平开始

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

{

data <<= 1; //左移一位，空位补0.

while(!DHT11\_REV\_DATA); //当低电平结束，等待准备数据的时间过去，消除影响

Delay\_us(40); //延时40us，卡时序，卡在低电平已经过，高电平还没结束

if(DHT11\_REV\_DATA==1)//如果还是高电平说明从机确实发送了高电平

//就算是低电平了也不会影响时序。

{

data |=1; //‘1’如果是高电平就将左移的一个空位

while(DHT11\_REV\_DATA);//等待高电平应答结束

}

}

return data;

}

# 实战代码

## 引脚初始化。

void DHT11\_Init(void)

{

RCC->APB2ENR |=0x01<<3;

DHT11\_Input( );// DHT11 的DATA 引脚处于输入状态，时刻检测外部信号

Delay\_ms(500);//DHT11 上电后要等待 1S 以越过不稳定状态在此期间不能发送任何指令

Delay\_ms(500);

}

将输出输入模式另外整合成函数，方便调用

void DHT11\_Output(void)

{

GPIOB->CRL &=~(0x0f<<12);

GPIOB->CRL |=(0x03<<12);

}

void DHT11\_Input(void)

{

GPIOB->CRL &=~(0x0f<<12);

GPIOB->CRL |=(0x04<<12);

}

## 发送起始信号

void DHT11\_Start(void)

{

DHT11\_SEND\_DATA=0;

Delay\_ms(18);

DHT11\_SEND\_DATA=1;

}

## 检测应答信号

//返回值 0--无应答 1--有应答

u8 DHT11\_Ack(void)

{

if(DHT11\_REV\_DATA!=0)

{

return 0;

}

while(DHT11\_REV\_DATA==0);//等待80us低电平应答结束

while(DHT11\_REV\_DATA==1);//等待80us高电平应答结束

return 1;

}

## 读取一个字节数据

u8 DHT11\_Read\_Byte(void)

{

u8 data=0;

u8 i;

//接受数据,每个数据以50us低电平开始

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

{

data <<= 1;

while(!DHT11\_REV\_DATA);//等待

Delay\_us(40);

if(DHT11\_REV\_DATA==1)

{

data |=1;//‘1’

while(DHT11\_REV\_DATA);//等待高电平应答结束

}

}

return data;

}

## 最终函数

//读取温湿度

u8 DHT11\_Read\_Data(u8 \*Temp\_H,u8\* Temp\_L,u8\* RH\_H,u8\* RH\_L)

//注意这里是址传递

{

u8 i;

u8 check;

DHT11\_Output( );

DHT11\_Start( );

Delay\_us(20);

DHT11\_Input( );//切换为输入

if( DHT11\_Ack( )==0 )

{

return -1;

}

//读取数据，分开读40个字节，调用5次读取函数

\*RH\_H=DHT11\_Read\_Byte( );

\*RH\_L=DHT11\_Read\_Byte( );

\*Temp\_H=DHT11\_Read\_Byte( );

\*Temp\_L=DHT11\_Read\_Byte( );

check=DHT11\_Read\_Byte( );

//结束读取

Delay\_ms(50);

DHT11\_Output( );//切换为输出

DHT11\_SEND\_DATA=1;//释放总线

//校验数据

if(check !=(\*Temp\_H+\*Temp\_L+\*RH\_H+\*RH\_L) )

{

return -2;

}

return 0;

}