温度传感器 DS18B20 资料

2008-08-28 16:06

美国 Dallas 半导体公司的数字化温度传感器 DS1820 是世界上第一片支持 "一线总线"接口的温度传感器，在其内部使用了在板（ON-BOARD）专利技术。全部传感元件及转换电路集成在形如一只三极管的集成电路内。一线总线独特而且经济的特点，使用户可轻松地组建传感器网络，为测量系统的构建引入全新概念。现在，新一代的 DS18B20 体积更小、更经济、更灵活。使你可以充分发挥“一线总线”的优点。目前 DS18B20 批量采购价格仅 10 元左右。

在传统的模拟信号远距离温度测量系统中，需要很好的解决引线误差补偿问题、多点测量切换误差问题和放大电路零点漂移误差问题等技术问题，才能够达到较高的测量精度。另外一般监控现场的电磁环境都非常恶劣，各种干扰信号较强，模拟温度信号容易受到干扰而产生测量误差，影响测量精度。因此，在温度测量系统中，采用抗干扰能力强的新型数字温度传感器是解决这些问题的最有效方案，新型数字温度传感器 DS18B20 具有体积更小、精度更高、适用电压更宽、采用一线总线、可组网等优点，在实际应用中取得了良好的测温效果。

新的"一线器件"DS18B20 体积更小、适用电压更宽、更经济。

DS18B20、DS1822 的特性

DS18B20 可以程序设定 9～12 位的分辨率，精度为±0.5° C。可选更小的封装方式，更宽的电压适用范围。分辨率设定，及用户设定的报警温度存储在 EEPROM 中，掉电后依然保存。DS18B20 的性能是新一代产品中最好的！性能价格比也非常出色！DS1822 与 DS18B20 软件兼容，是 DS18B20 的简化版本。省略了存储用户定义报警温度、分辨率参数的 EEPROM，精度降低为±2° C，适用于对性能要求不高，成本控制严格的应用，是经济型产品。 继"一线总线"的早期产品后，DS1820 开辟了温度传感器技术的新概念。DS18B20 和 DS1822 使电压、特性及封装有更多的选择，让我们可以构建适合自己的经济的测温系统。

DS18B20、DS1822 "一线总线"数字化温度传感器

同 DS1820 一样，DS18B20 也支持"一线总线"接口，测量温度范围为-55° C～+125° C，在-10～+85° C 范围内，精度为±0.5° C。DS1822 的精度较差为±2° C。现场温度直接以"一线总线"的数字方式传输，大大提高了系统的抗干扰性。适合于恶劣环境的现场温度测量，如：环境控制、设备或过程控制、测温类消费电子产品等。与前一代产品不同，新的产品支持 3V～5.5V 的电压范围，使系统设计更灵活、方便。而且新一代产品更便宜，体积更小。

一、DS18B20 的主要特性

（1）适应电压范围更宽，电压范围：3.0～5.5V，在寄生电源方式下可由数据线供电

（2）独特的单线接口方式，DS18B20 在与微处理器连接时仅需要一条口线即可实现微处理器与 DS18B20 的双向通讯

（3）DS18B20 支持多点组网功能，多个 DS18B20 可以并联在唯一的三线上，实现组网多点测温

（4）DS18B20 在使用中不需要任何外围元件，全部传感元件及转换电路集成在形如一只三极管的集成电路内

（5）温范围－55℃～＋125℃，在-10～+85℃时精度为±0.5℃

（6)可编程的分辨率为 9～12 位,对应的可分辨温度分别为 0.5℃、0.25℃、0.125℃和 0.0625℃，

可实现高精度测温

（7）在 9 位分辨率时最多在 93.75ms 内把温度转换为数字，12 位分辨率时最多在 750ms 内把温度值转换为数字，速度更快

（8）测量结果直接输出数字温度信号，以"一线总线"串行传送给 CPU，同时可传送 CRC 校验码，具有极强的抗干扰纠错能力

（9）负压特性：电源极性接反时，芯片不会因发热而烧毁，但不能正常工作。

二、DS18B20 的外形和内部结构

DS18B20 内部结构主要由四部分组成：64 位光刻 ROM、温度传感器、非挥发的温度报警触发器 TH 和 TL、配置寄存器。DS18B20 的外形及管脚排列如下图 1:

图 1： DS18B20 外形及引脚排列图

DS18B20 引脚定义：

(1)DQ 为数字信号输入/输出端；
(2)GND 为电源地；
(3)VDD 为外接供电电源输入端（在寄生电源接线方式时接地）。

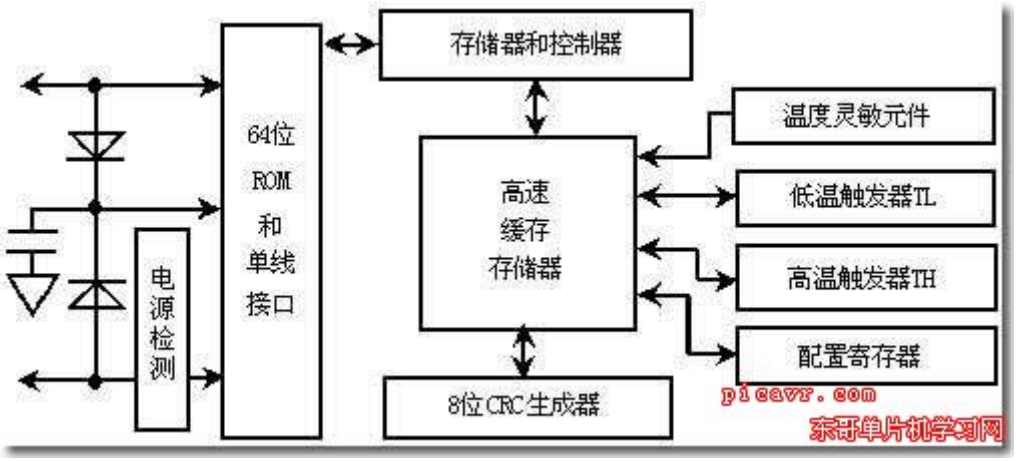
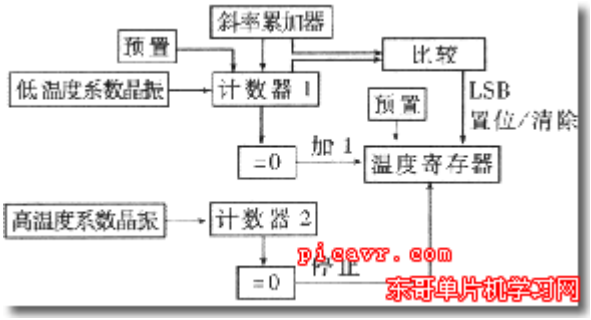


图 2： DS18B20 内部结构图

三、DS18B20 工作原理

DS18B20 的读写时序和测温原理与 DS1820 相同，只是得到的温度值的位数因分辨率不同而不同，且温度转换时的延时时间由 2s 减为 750ms。 DS18B20 测温原理如图 3 所示。图中低温度系数晶振的振荡频率受温度影响很小，用于产生固定频率的脉冲信号送给计数器 1。高温度系数晶振随温度变化其振荡率明显改变，所产生的信号作为计数器 2 的脉冲输入。计数器 1 和温度寄存器被预置在－55℃所对应的一个基数值。计数器 1 对低温度系数晶振产生的脉冲信号进行减法计数，当计数器 1 的预置值减到 0 时，温度寄存器的值将加 1，计数器 1 的预置将重新被装入，计数器 1 重新开始对低温度系数晶振产生的脉冲信号进行计数，如此循环直到计数器 2 计数到 0 时，停止温度寄存器值的累加，此时温度寄存器中的数值即为所测温度。图 3 中的斜率累加器用于补偿和修正测温过程中的非线性，其输出用于修正计数器 1 的预置值。

图 3： DS18B20 测温原理框图



DS18B20 有 4 个主要的数据部件：

（1）光刻 ROM 中的 64 位序列号是出厂前被光刻好的，它可以看作是该 DS18B20 的地址序列码。64 位光刻 ROM 的排列是：开始 8 位（28H）是产品类型标号，接着的 48 位是该 DS18B20 自身的序列号，最后 8 位是前面 56 位的循环冗余校验码（CRC=X8+X5+X4+1）。光刻 ROM 的作用是使每一个 DS18B20 都各不相同，这样就可以实现一根总线上挂接多个 DS18B20 的目的。

（2）DS18B20 中的温度传感器可完成对温度的测量，以 12 位转化为例：用 16 位符号扩展的二进制补码读数形式提供，以 0.0625℃/LSB 形式表达，其中 S 为符号位。

表 1： DS18B20 温度值格式表

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LS Byte	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
MS Byte	S	S	S	S	S	2^6	2^5	2^4

这是 12 位转化后得到的 12 位数据，存储在 18B20 的两个 8 比特的 RAM 中，二进制中的前面 5 位是符号位，如果测得的温度大于 0，这 5 位为 0，只要将测到的数值乘于 0.0625 即可得到实际温度；如果温度小于 0，这 5 位为 1，测到的数值需要取反加 1 再乘于 0.0625 即可得到实际温度。

例如+125℃的数字输出为 07D0H，+25.0625℃的数字输出为 0191H，-25.0625℃的数字输出为 FF6FH，-55℃的数字输出为 FC90H。

表 2： DS18B20 温度数据表

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	0000 0111 1101 0000	07D0h
+85°C*	0000 0101 0101 0000	0550h
+25.0625°C	0000 0001 1001 0001	0191h
+10.125°C	0000 0000 1010 0010	00A2h
+0.5°C	0000 0000 0000 1000	0008h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1000	FFF8h
-10.125°C	1111 1111 0101 1110	FF5Eh
-25.0625°C	1111 1110 0110 1111	FE6Fh
-55°C	1111 1100 1001 0000	FC90h

*The power-on reset value of the temperature register is +85°C

（3）DS18B20 温度传感器的存储器

DS18B20 温度传感器的内部存储器包括一个高速暂存 RAM 和一个非易失性的可电擦除的 EEPRAM,后者存放高温度和低温度触发器 TH、TL 和结构寄存器。

（4）配置寄存器

该字节各位的意义如下：

表 3： 配置寄存器结构

TM	R1	R0	1	1	1	1	1

低五位一直都是"1"，TM 是测试模式位，用于设置 DS18B20 在工作模式还是在测试模式。在 DS18B20 出厂时该位被设置为 0，用户不要去改动。R1 和 R0 用来设置分辨率，如下表所示：（DS18B20 出厂时被设置为 12 位）

表 4： 温度分辨率设置表

R1	R0	分辨率	温度最大转换时间
0	0	9 位	93.75ms
0	1	10 位	187.5ms
1	0	11 位	375ms
1	1	12 位	750ms

四、 高速暂存存储器

高速暂存存储器由 9 个字节组成，其分配如表 5 所示。当温度转换命令发布后，经转换所得的温度值以二字节补码形式存放在高速暂存存储器的第 0 和第 1 个字节。单片机可通过单线接口读到该数据，读取时低位在前，高位在后，数据格式如表 1 所示。对应的温度计算：当符号位 S=0 时，直接将二进制位转换为十进制；当 S=1 时，先将补码变为原码，再计算十进制值。表?2 是对应的一部分温度值。第九个字节是冗余检验字节。

表 5： DS18B20 暂存寄存器分布

寄存器内容	**字节地址**
温度值低位 （LS Byte）	0
温度值高位 （MS Byte）	1
高温限值（TH）	2
低温限值（TL）	3
配置寄存器	4
保留	5
保留	6
保留	7
CRC 校验值	8

根据 DS18B20 的通讯协议，主机（单片机）控制 DS18B20 完成温度转换必须经过三个步骤：每一次读写之前都要对 DS18B20 进行复位操作，复位成功后发送一条 ROM 指令，最后发送 RAM 指令，这样才能对 DS18B20 进行预定的操作。复位要求主 CPU 将数据线下拉 500 微秒，然后释放，当 DS18B20 收到信号后等待 16～60 微秒左右，后发出 60～240 微秒的存在低脉冲，主 CPU 收到此信号表示复位成功。

表 6： ROM 指令表

指 令	**约定代码**	**功 能**
读 ROM	33H	读 DS1820 温度传感器 ROM 中的编码（即 64 位地址）
符合 ROM	55H	发出此命令之后，接着发出 64 位 ROM 编码，访问单总线上与该编码相对应的 DS1820 使之作出响应，为下一步对该 DS1820 的读写作准备。
搜索 ROM	0F0H	用于确定挂接在同一总线上 DS1820 的个数和识别 64 位 ROM 地址。为操作各器件作好准备。
跳过 ROM	0CCH	忽略 64 位 ROM 地址，直接向 DS1820 发温度变换命令。适用于单片工作。
告警搜索命令	0ECH	执行后只有温度超过设定值上限或下限的片子才做出响应。

表 6： RAM 指令表

指 令	约定代码	功 能
温度变换	44H	启动DS1820进行温度转换，12位转换时最长为750ms（9 位为 93.75ms）。结果存入内部 9 字节 RAM 中。
读暂存器	0BEH	读内部 RAM 中 9 字节的内容
写暂存器	4EH	发出向内部 RAM 的 3、4 字节写上、下限温度数据命令，紧跟该命令之后，是传送两字节的数据。
复制暂存器	48H	将 RAM 中第 3 、4 字节的内容复制到 EEPROM 中。
重调 EEPROM	0B8H	将 EEPROM 中内容恢复到 RAM 中的第 3 、4 字节。
读供电方式	0B4H	读 DS1820 的供电模式。寄生供电时 DS1820 发送“ 0 ”，外接电源供电 DS1820 发送“ 1 ”。

五、DS18B20 的应用电路

DS18B20 测温系统具有测温系统简单、测温精度高、连接方便、占用口线少等优点。下面就是 DS18B20 几个不同应用方式下的测温电路图：

[1]、DS18B20 寄生电源供电方式电路图

如下面图 4 所示，在寄生电源供电方式下，DS18B20 从单线信号线上汲取能量：在信号线 DQ 处于高电平期间把能量储存在内部电容里，在信号线处于低电平期间消耗电容上的电能工作，直到高电平到来再给寄生电源（电容）充电。

独特的寄生电源方式有三个好处：

1）进行远距离测温时，无需本地电源
2）可以在没有常规电源的条件下读取 ROM
3）电路更加简洁，仅用一根 I/O 口实现测温

要想使 DS18B20 进行精确的温度转换，I/O 线必须保证在温度转换期间提供足够的能量，由于每个 DS18B20 在温度转换期间工作电流达到 1mA，当几个温度传感器挂在同一根 I/O 线上进行多点测温时，只靠 4.7K 上拉电阻就无法提供足够的能量，会造成无法转换温度或温度误差极大。
因此，图 4 电路只适应于单一温度传感器测温情况下使用，不适宜采用电池供电系统中。并且工作电源 VCC 必须保证在 5V，当电源电压下降时，寄生电源能够汲取的能量也降低，会使温度误差变大。

注： 站长曾经就此电路做过实验，在实验中，降低电源电压 VCC，当低于 4.5V 时，测出的温度值比实际的温度高，误差较大。。。当电源电压降为 4V 时，温度误差有 3℃之多，这就应该是因为寄生电源汲取能量不够造成的吧，因此，站长建议大家在开发测温系统时不要使用此电路。

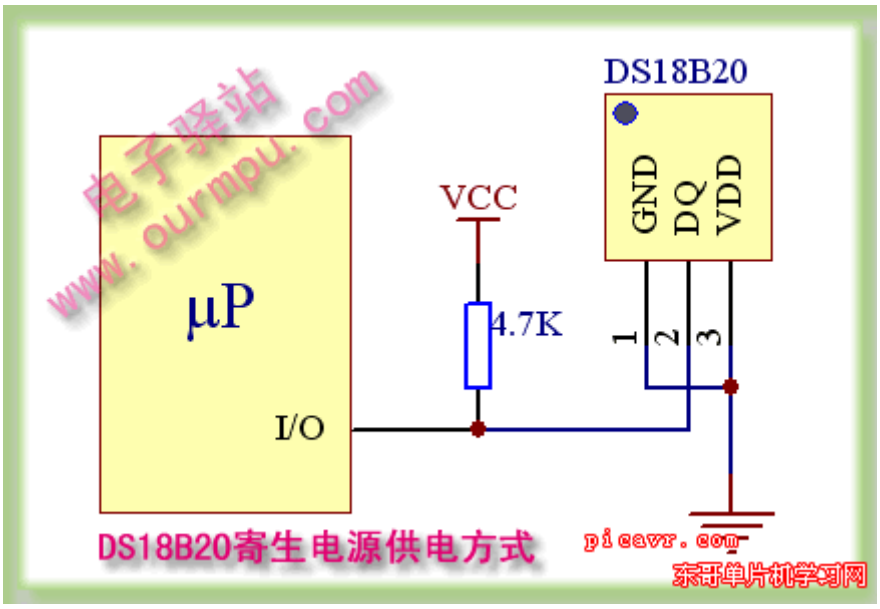


图 4

[2]、DS18B20 寄生电源强上拉供电方式电路图

改进的寄生电源供电方式如下面图 5 所示，为了使 DS18B20 在动态转换周期中获得足够的电流供应，当进行温度转换或拷贝到 E2 存储器操作时，用 MOSFET 把 I/O 线直接拉到 VCC 就可提供足够的电流，在发出任何涉及到拷贝到 E2 存储器或启动温度转换的指令后，必须在最多 10μS 内把 I/O 线转换到强上拉状态。在强上拉方式下可以解决电流供应不走的问题，因此也适合于多点测温应用，缺点就是要多占用一根 I/O 口线进行强上拉切换。

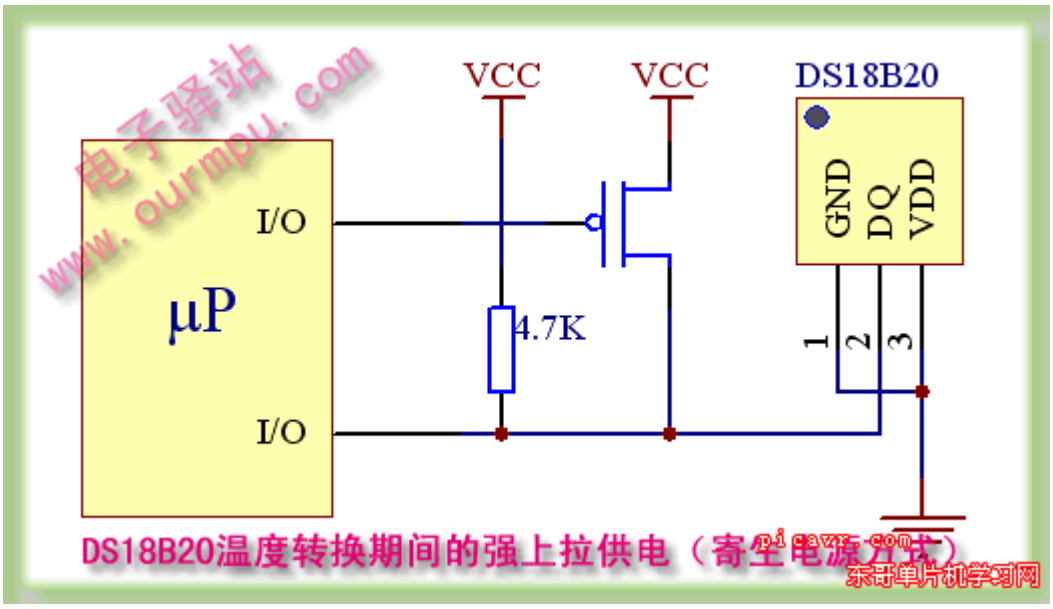


图 5

注意：在图 4 和图 5 寄生电源供电方式中，DS18B20 的 VDD 引脚必须接地

[3]、DS18B20 的外部电源供电方式

在外部电源供电方式下，DS18B20 工作电源由 VDD 引脚接入，此时 I/O 线不需要强上拉，不存在电源电流不足的问题，可以保证转换精度，同时在总线上理论可以挂接任意多个 DS18B20 传感器，组成多点测温系统。注意：在外部供电的方式下，DS18B20 的 GND 引脚不能悬空，否则不能转换温度，读取的温度总是 85℃。

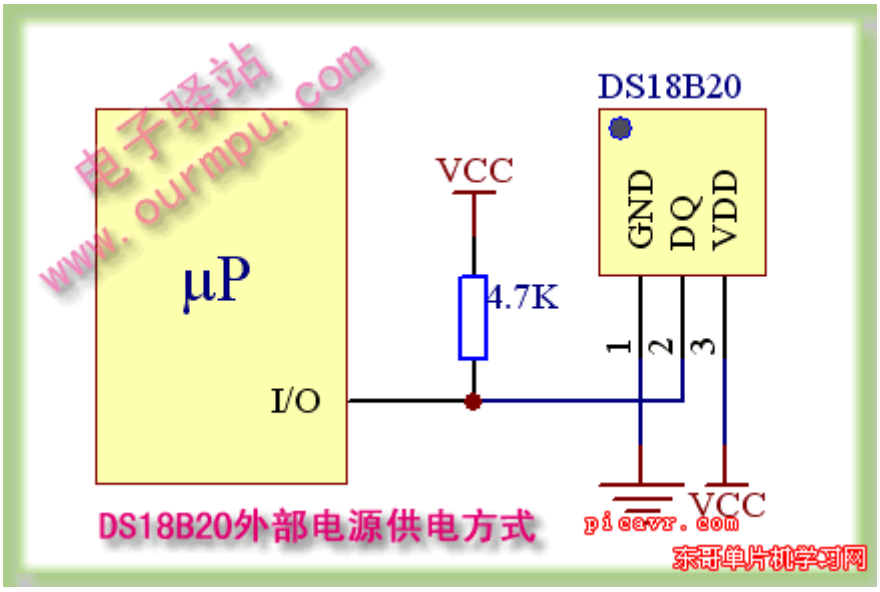


图 6：外部供电方式单点测温电路

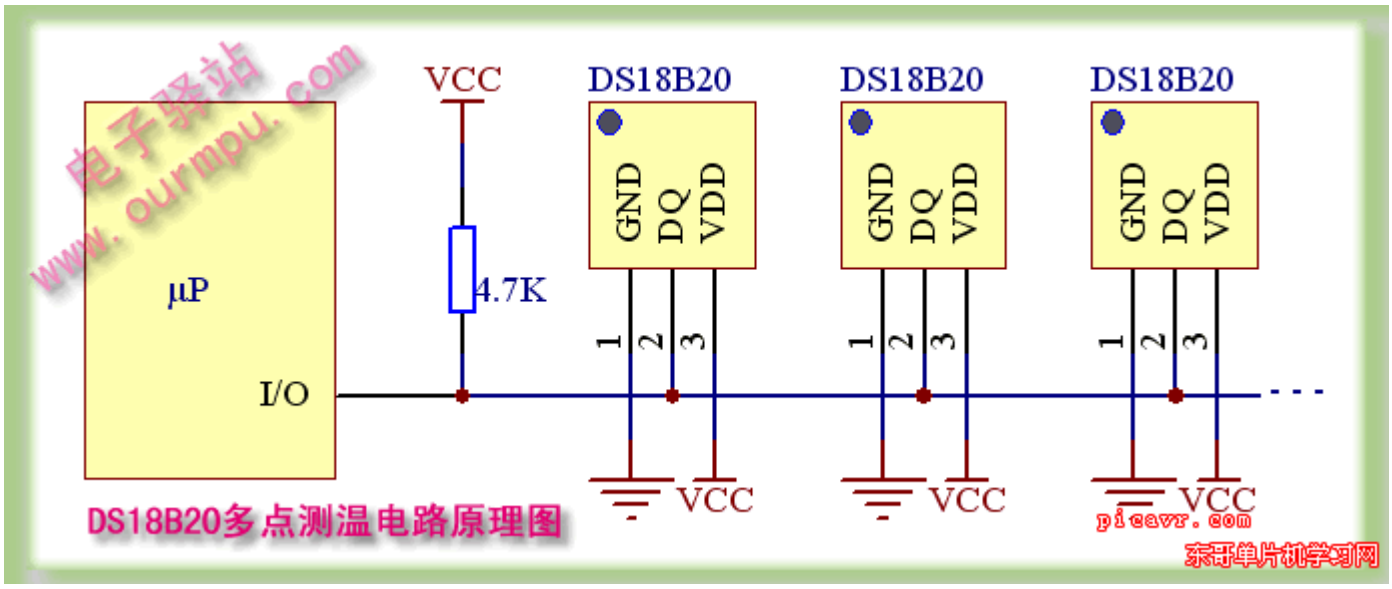


图 7：外部供电方式的多点测温电路图

外部电源供电方式是 DS18B20 最佳的工作方式，工作稳定可靠，抗干扰能力强，而且电路也比较简单，可以开发出稳定可靠的多点温度监控系统。站长推荐大家在开发中使用外部电源供电方式，毕竟比寄生电源方式只多接一根 VCC 引线。在外接电源方式下，可以充分发挥 DS18B20 宽电源电压范围的优点，即使电源电压 VCC 降到 3V 时，依然能够保证温度量精度。

```
/*************************************************************
*DS1820 驱动程序
*版本:V1.0
*************************************************************/

#include <reg52.h>
#define U8 unsigned char

sbit DS1820_DQ= P1^4; //单总线引脚

void DS18B20_Init() ;                    //DS18B20 初始化
bit    DS1820_Reset();                    //DS1820 复位
void DS1820_WriteData(U8 wData);   //写数据到 DS1820
U8       DS1820_ReadData();                //读数据

/*************************************************************
*DS18B20 初始化
*函数名称:DS1820_WriteData()
*说明：本初始化程序可以不要，因为 18B20 在出厂时就被配置为 12 位精度了
*************************************************************/
void DS18B20_Init()
{
   DS1820_Reset();
   DS1820_WriteData(0xCC);   // 跳过 ROM
   DS1820_WriteData(0x4E);   // 写暂存器
   DS1820_WriteData(0x20);   // 往暂存器的第三字节中写上限值
   DS1820_WriteData(0x00);   // 往暂存器的第四字节中写下限值
   DS1820_WriteData(0x7F);   // 将配置寄存器配置为 12 位精度

   DS1820_Reset();
}

/*************************************************************
*DS1820 复位及存在检测(通过存在脉冲可以判断 DS1820 是否损坏)
*函数名称:DS1820_Reset()
*说明:函数返回一个位标量(0 或 1)flag=0 存在,反之 flag=1 不存在
*************************************************************/
bit DS1820_Reset()
{
  U8 i;
  bit flag;

  DS1820_DQ = 0;                //拉低总线
  for (i=240;i>0;i--);     //延时 480 微秒,产生复位脉冲
  DS1820_DQ = 1;               //释放总线
  for (i=40;i>0;i--);       //延时 80 微秒对总线采样
  flag =    DS1820_DQ;          //对数据脚采样
```

注意 i 的类型

注意延时时间需要满足我们系统要求
可以替换成多个nop();函数

```
 for (i=200;i>0;i--);    //延时 400 微秒等待总线恢复
 return (flag);          //根据 flag 的值可知 DS1820 是否存在或损坏 ，可加声音告警提示 DS1820 故障
}

/*******************************************************
*写数据到 DS1820
*函数名称:DS1820_WriteData()
*******************************************************/
void DS1820_WriteData(U8 wData)
{
 U8 i,j;
 for (i=8;i>0;i--)
  {
  DS1820_DQ = 0;              //拉低总线,产生写信号
  for (j=2;j>0;j--);          //延时 4us
  DS1820_DQ = wData&0x01; //发送 1 位
  for (j=30;j>0;j--);         //延时 60us,写时序至少要 60us
  DS1820_DQ = 1;              //释放总线,等待总线恢复
  wData>>=1;                  //准备下一位数据的传送
  }
}

/*******************************************************
*从 DS1820 中读出数据
*函数名称:DS1820_ReadData()
*******************************************************/
U8 DS1820_ReadData()
{
 U8 i,j,TmepData;
 for (i=8;i>0;i--)
  {
   TmepData>>=1;
   DS1820_DQ = 0;              //拉低总线,产生读信号
   for (j=2;j>0;j--);          //延时 4us
   DS1820_DQ = 1;              //释放总线,准备读数据
   for (j=4;j>0;j--);          //延时 8 微秒读数据
   if (DS1820_DQ == 1)
     { TmepData |= 0x80;}
   for (j=30;j>0;j--);         //延时 60us
   DS1820_DQ = 1;              //拉高总线,准备下一位数据的读取.
  }
 return (TmepData);//返回读到的数据
}
```

★★★

```
/*******************************************************
*采用 DS1820+1602 精确到 4 位小数的温度计
```

```
*编程:沈阳
*优化:weishao  于 09.7.19 优化
*版本:V1.1
*DS1820 设置为 12 位精度
************************************************************/

#include <reg52.h>
#include <temp.h>

U8 code string[16]={"Temp Display"};
U8 code DispStr[16]={"www.sxcckj.com"};

U8 temperature[2]; //存放温度数据

void display();              //待显温度值转换


/************************************************************
*主程序
************************************************************/
void main()
{
    U8 i;
    DelayMs(255);//等待电源稳定,液晶复位完成

    LCDInit();                  //1602 液晶初始化
    DS18B20_Init();             //18B20 初始化，可不用初始化，因为 18B20 出厂时默认是 12 位精度

    SetLight(1);                //打开背光灯
    DisplayStr(0,0,string);//显示"温度:"
    DisplayStr(0,1,DispStr);//显示"系列号"
    DelayMs(250);
    DelayMs(250);
    DelayMs(250);
    DelayMs(250);
    DelayMs(250);
    DelayMs(250);
    DelayMs(250);
    DelayMs(250);
    DelayMs(250);
    DelayMs(250);
    DelayMs(250);
    WriteCMD(0x01,1);          //1601 清屏
    DelayMs(50);
    SetLight(0);                //关闭背光灯

    while (1)
    {
```

```
        DS1820_Reset();              //复位
        DS1820_WriteData(0xcc); //跳过 ROM 命令
        DS1820_WriteData(0x44); //温度转换命令

        DS1820_Reset();              //复位
        DS1820_WriteData(0xcc); //跳过 ROM 命令
        DS1820_WriteData(0xbe); //读 DS1820 温度暂存器命令

        for (i=0;i<2;i++)
        {
            temperature[i]=DS1820_ReadData();      //采集温度
        }
        DS1820_Reset();          //复位,结束读数据

        display();   //显示温度值
        DelayMs(50);

    }
}


/*************************************************************
*转换显示子程序
*************************************************************/
void display()
{
    U8 temp_data,temp_data_2;
    U8 temp[7];          //存放分解的 7 个 ASCII 码温度数据
    U16 TempDec; //用来存放 4 位小数


    temp_data = temperature[1];
    temp_data &= 0xf0;   //取高 4 位

    if (temp_data==0xf0) //判断是正温度还是负温度读数
    {
        //负温度读数求补,取反加 1,判断低 8 位是否有进位
        if (temperature[0]==0)
        { //有进位,高 8 位取反加 1
            temperature[0]=~temperature[0]+1;
            temperature[1]=~temperature[1]+1;
        }
        else
        { //没进位,高 8 位不加 1
            temperature[0]=~temperature[0]+1;
            temperature[1]=~temperature[1];
        }
```

(1) 先读低字节、再读高字节
(2) 16进制数转成十进制温度值**.**℃
(3) 将温度值送入显缓单元

```
    }

    temp_data = temperature[1]<<4;     //取高字节低 4 位(温度读数高 4 位)，注意此时是 12 位精度
    temp_data_2 = temperature[0]>>4; //取低字节高 4 位(温度读数低 4 位)，注意此时是 12 位精度
    temp_data = temp_data|temp_data_2; //组合成完整数据

    temp[0] = temp_data/100+0x30;     //取百位转换为 ASCII 码
    temp[1] = (temp_data%100)/10+0x30;    //取十位转换为 ASCII 码
    temp[2] = (temp_data%100)%10+0x30;    //取个位转换为 ASCII 码

    temperature[0]&=0x0f;                    //取小数位转换为 ASCII 码
    TempDec = temperature[0]*625;            //625=0.0625*10000，表示小数部分,扩大 1 万倍 ，方便显示
    temp[3] = TempDec/1000+0x30;             //取小数十分位转换为 ASCII 码
    temp[4] = (TempDec%1000)/100+0x30;     //取小数百分位转换为 ASCII 码
    temp[5] = ((TempDec%1000)%100)/10+0x30;//取小数千分位转换为 ASCII 码
    temp[6] = ((TempDec%1000)%100)%10+0x30;//取小数万分位转换为 ASCII 码

    if(temp[0]==0x30) DisplayChar(3,0,' ');//如果百位为 0，显示空格
        else DisplayChar(3,0,temp[0]);            //否则正常显示百位

    DisplayChar(4,0,temp[1]);//十位
    DisplayChar(5,0,temp[2]);//个位
    DisplayChar(6,0,0x2e);     //小数点 .
    DisplayChar(7,0,temp[3]);
    DisplayChar(8,0,temp[4]);
    DisplayChar(9,0,temp[5]);
    DisplayChar(10,0,temp[6]);
    DisplayChar(11,0,'\'');    //显示'
    DisplayChar(12,0,'C');     //显示 C
}
```