华侨大学
Huaqiao university

专 业 课 程 综 合 设 计

题目:基于 DS18B20 及数码管的两点温度测量显示装置

院　（系）　机电及自动化学院

专　　业　　测控技术与仪器

学　　号　　1611212008

姓　　名　　邓希祥

级　　别　　２０１６

指导老师　　杜 建 华

2020 年 1 月

摘要

随着时代的进步和发展，单片机技术已经普及到我们生活、工作、科研、各个领域，已经成为一种比较成熟的技术,本文介绍了基于 DS18B20 及数码管的两点温度测量显示装置,详细描述了利用数字温度传感器 DS18B20 开发测温系统的过程，重点对传感器在单片机下的硬件连接，软件编程以及各模块系统流程进行了详尽分析，特别是数字温度传感器 DS18B20 的数据采集过程。对各部分的电路也一一进行了介绍,该系统可以方便的实现温度采集和显示，并可根据需要任意设定上下限报警温度，它使用起来相当方便，具有精度高、量程宽、灵敏度高、体积小、功耗低等优点，适合于我们日常生活和工、农业生产中的温度测量，也可以当作温度处理模块嵌入其它系统中，作为其他主系统的辅助扩展。DS18B20 与 STC89C51 结合实现最简温度检测系统，该系统结构简单，抗干扰能力强，适合于恶劣环境下进行现场温度测量，有广泛的应用前景。

关键词：STC89C51, DS18B20, 温度传感器，数字温度计

目录

一． 系统功能定义

基于 DS18B20 及数码管的两点温度测量显示装置对两个通道中的温度值进行采集，通过一个四位七段数码管显示温度，并有 key3、key4 两个按键分别控制四位七段数码管显示两个通道的温度，当 key3 按下时，数码管显示通道一的温度，当 key4 按下时，数码管切换到通道二的温度。另外该系统配有温度报警装置，当温度溢出温度上下限时，指示灯亮且蜂鸣器响。温度的上下限由两个两位数码管显示，且可有按键调整，key1、key2 控制下限温度的升降，key5、key6 控制上限温度的升降。

二． DS18B20 简介

DS18B20 温度传感器是 DALLAS 公司生产的采用 1-Wire 总线技术的典型作品。他可以将被测温度直接转化成数字量，因此单片机可以方便的通过串行总线实现读取。另外，由于 1-Wire 具有成本低、节省 I/O 口、抗干扰能力强、便于总线扩展和维护等特点。

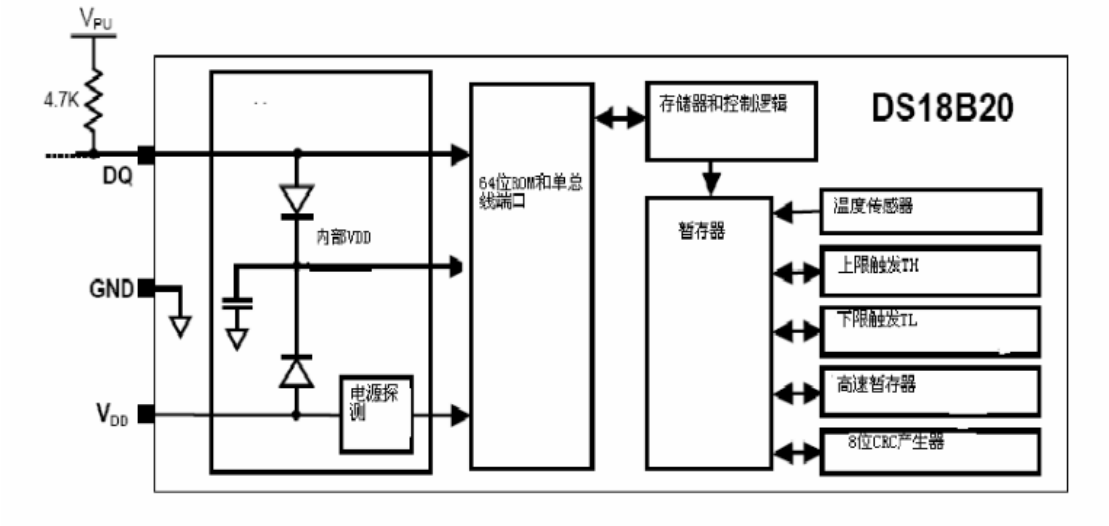
DS18B20 通过编程后，可以实现 9~12 位的温度度数。由于 DS18B20 可以有数据本身提供电源，因此单片机与其连接时，需要一根数据线和一根地线。由于每片 DS18B20 内部都有一个独特的片序列号，所以一条单总线上可以挂接多片 DS18B20，特别适合构成多点温度测控系统。

DS18B20 的工作性能如下：

- 1-Wire 数据通信；
- 可以数据线供电，电压范围 3~5.5V；
- 最高 12 位分辨率；
- 12 位分辨率时的最大工作周期；
- 可选择寄生工作方式；
- 检测温度范围为-55℃~+125℃；
- 为测温度在-10~+85℃时，精度为±0.5℃；
- 内置 E2PROM，限温报警功能；
- 64 位光刻 ROM，内置产品序列号，方便多机挂接；
- 封装形式多样；
- 负压特性，电源极性接反时，芯片不会烧毁。

2.1 DS18B20 内部结构及测温原理

如图 1 所示为 DS18B20 的内部结构图，它主要包括寄生电源电路、温度传感器、64 位激光 ROM 单线接口和 1.Wire 总线接口、存放中间数据的高速缓冲存储器、用于存储用户设定的温度下限值的 TH 和 TL 触发器存储与控制逻辑、8 位循环冗余检验码(CRC)发生器等 7 部分。



图一 DS18B202 内部结构图

通过寄生电源电路，DS18B20 可以从 1-Wrie 上取得其工作电源。在信号线为高电平的时间周期内，会把能量存储在内部的电容器中；在单信号线为低电平的时间期内，断开此电源，直到信号地变为高电平重新接上寄生(电容)电源为止。当然，DS18B20 也可以直接通过将+5V 电源接至 VID 引脚为其供电。

2.2 DS18B20 的封装形式及引脚功能

DS18B20 有 8 引脚 SO 封装、8 引脚 μ SOP 封装以及 3 引脚 TO-92 封装 3 种形式。图 3 给出了 DS18B20 芯片 TO-92 封装和 SO/ μ SOP 封装及引脚分布。

2.3 DS18B20 的储存器

DS18B20 的存储器由暂存器和 EEPROM 组成。暂存器用于 1-Wire 通信时确保数据传输的正确性。进行数据通信时，数据首先会被写入暂存器中，接着完成校验后才可以被送入 EEPROM 中。

DS18B20 内部暂存器的顺序如图 4 所示。

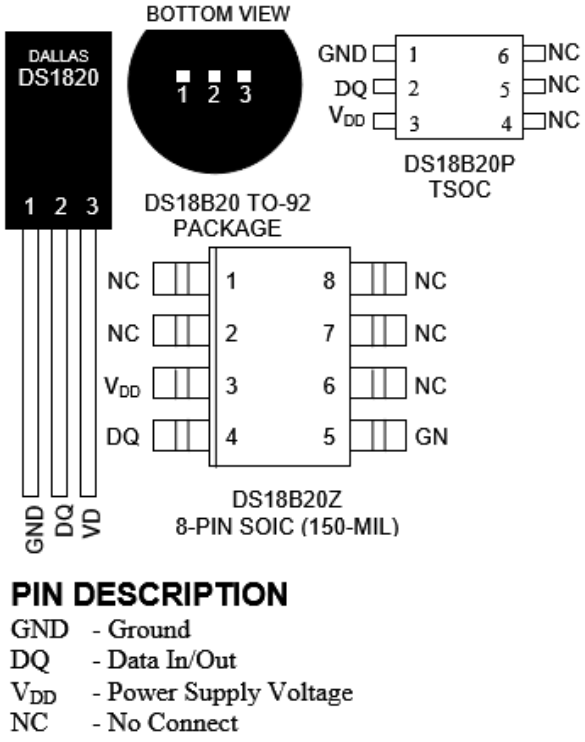


图 3 DS18B20 的封装形式及引脚分布

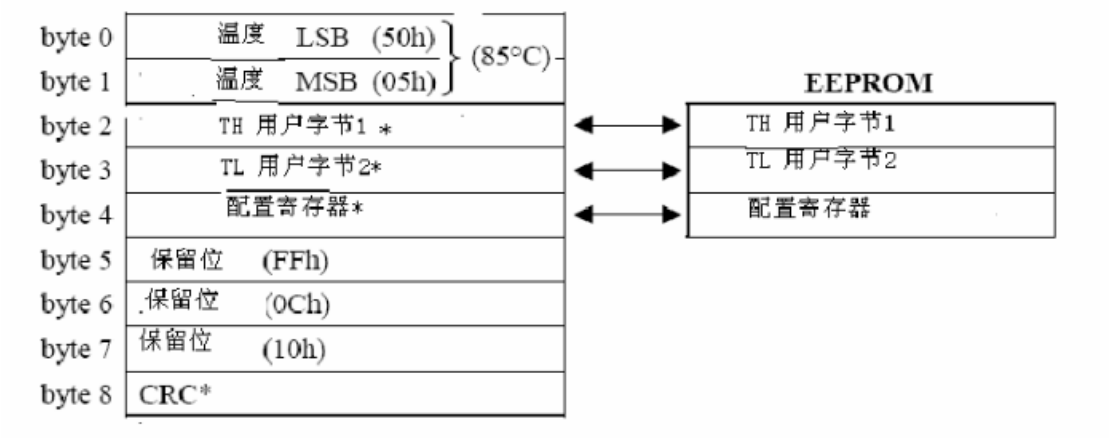


图 4 内部暂存器结构

字节 0 和字节 1：温度值。其中，字节 0 是温度值低位，字节 1 为高位。温度高位、低位寄存器格式如图 5。

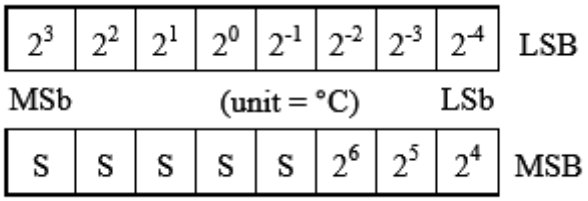


图 5 温度寄存器结构

温度高位寄存器中的 s 表示符号位。当 s=0 时，表示温度是正值，可以直接将二进制转化为十进制；当 s=1 时，表示温度是负值，需要先将补码转换为源码后，在转换为十进制数。

字节 2 和字节 3：非易失性报警温度上限和下限，可以通过软件进行修改。上电后，会自动 EEPROM 中复制到暂存器。

字节 4：配置寄存器。用于确定温度值的数字转化分辨率，上电时也会自动从 EEPROM 中复制到暂存器。该寄存器格式如下。

0	R1	R0	1	1	1	1	1

通过写命令来配置寄存器中的 R0 和 R1，能够设置 DS18B20 的分辨率。R0 和 R1 的取值与分辨率的关系以及不同分辨率下的转换时间如图 6 所示。

R1	R0	Thermometer Resolution	Max Conversion Time
0	0	9-bit	93.75ms ($t_{conv}/8$)
0	1	10-bit	187.5ms ($t_{conv}/4$)
1	0	11-bit	375ms ($t_{conv}/2$)
1	1	12-bit	750ms (t_{conv})

图 6　R0 和 R1 的取值与分辨率的关系以及不同分辨率下的转换时间

字节 5--字节 7：未定义，字节中各位全部位逻辑 1。

字节 9：前面 8 个字节的循环冗余效验字节。

三． 硬件设计

3.0 总体流程图

温度传感器 DS18B20 集成了 A/D 转换的功能，所以在连接单片机时无需进行 A/D 转换电路的连接，将采集的温度数据经过 DS18B20 的处理将温度值输出给单片机，通过单片机的控制输出使数码管显示。

3.1 温度采集设计

温度传感器采用 DS18B20，是一种单总线智能型温度传感器，只有三线接口，分别为地线（GND）、数据线（DQ）、电源线（VCC）。DS18B20 输出信号为数字信号，处理器与 DS18B20 通过数据线（DQ）来完成双向通信。温度变换功率可以来源于外电源，也可以由数据线直接供电。本实验有两路温度路数，由一个+5V 电源对其供电。两个 DS1820 分别接在 P3.6 和 P3.7 口，单片机分别对其控制和读取数据。如图 7。

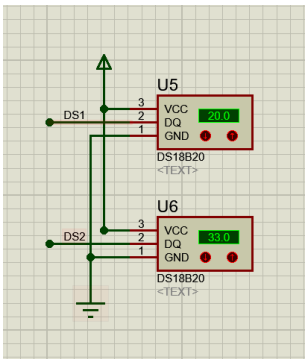


图 7 温度采集模块

3.2 温度显示设计

由于 DS18B20 所能检测到温度范围为-55℃~+125℃，当温度为正温度时，数码管显示温度的百位、十位、个位还有一个小数位；当温度为负温度时，数码管显示负号、十位、个位和小数位。所以本实验采用四位一体共阳极数码管显示温度，段选由单片机 P0 口控制，位选由 P2.0、P2.1、P2.2、P2.3 控制。并有两个

按键选择数码管显示的通道。当 key3 按下时，数码管显示通道一的温度，当 key4 按下时，数码管切换到通道二的温度。如图 8。

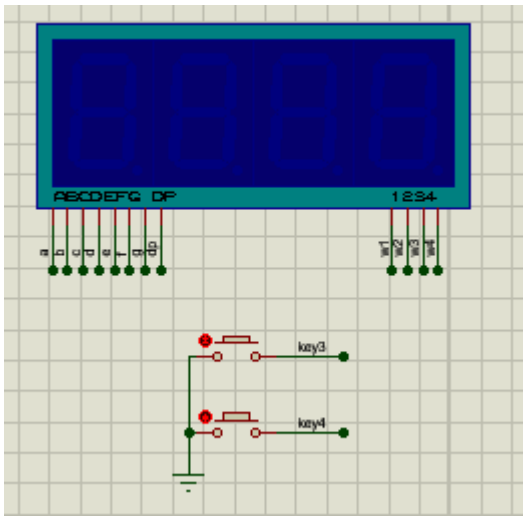


图 8 温度显示模块

3.3 上下限显示及按键调整设计

温度上下限由两个两位数码管显示，并且可由下方的两个按键对其调整。两个数码管的段选都有 P0 口控制，下限温度显示位选由 P2.4、P2.5 口控制，上限温度显示位选由 P2.6、P2.7 口控制。按键 Key1、Key2、Key3、Key4 分别由 P1.2、P1.3、P1.4、P1.5 口控制。

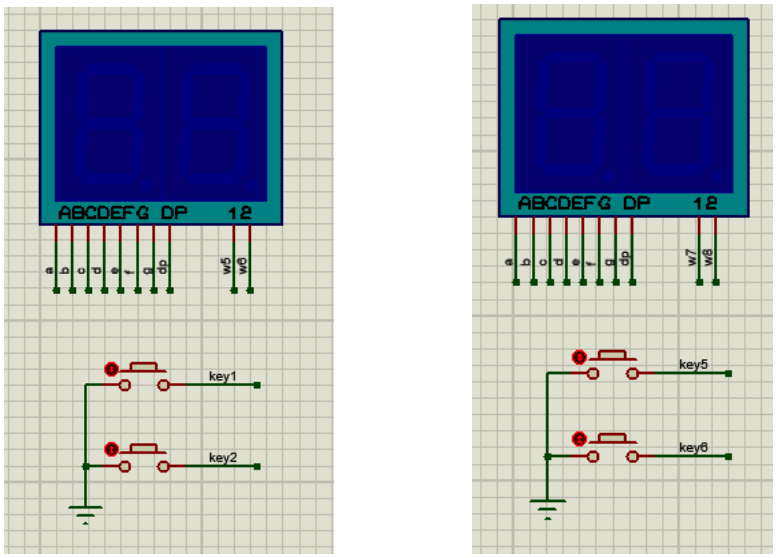


图 9 上下限温度模块

3.4 报警模块设计

当所测温度溢出上下限时，单片机会从 P1.6 口输出低电平，使二极管和蜂鸣器两端产生电势差，二极管发亮、蜂鸣器响起，由报警模块向外界发出警告，温度已溢出上下限。电路图如图 10 所示。

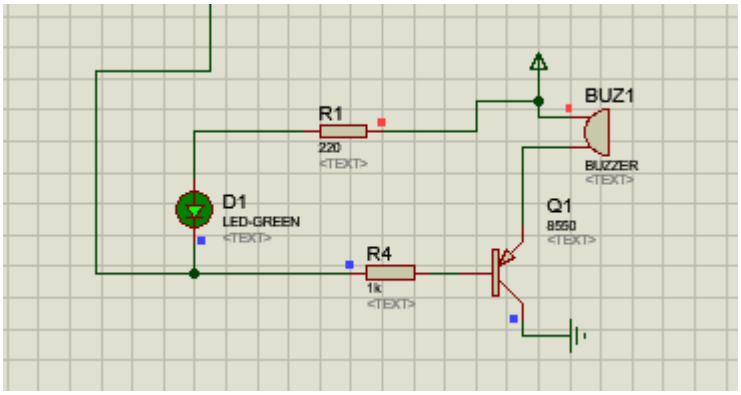


图 10 报警模块

四． 软件设计

对于 DS18B20 的程序编写要特别注意时序问题，如果采用 C 语言编程，其对时序要求很严，倘若时序错误会导致单片机读不到数据，或是读到的数据都是错误的，更严重就是传感器不工作，无法进行温度采集；汇编语言对时序要求没有那么严格，因为程序运行每一条汇编语句都会有一个机器周期。下面根据时序并结合子程序的介绍。

4.1 DS18B20 初始化

单总线上的所有通信都是以初始化序列开始的，包括主机发出的复位脉冲及从机发出的应答脉冲，如图 11 所示。当从机发出相应主机的应答时，即向主机表明它处于总线上，且工作准备就绪。在主机初始化过程中，要有以下几个步骤：

1. 主机拉低单总线至少 480μs，产生复位脉冲。
2. 释放总线，进入接收状态。
3. DS18B20 检测到上升沿后，延时 15~60μs，会拉低总线 60~240μs，已应答主机，该器件准备就绪；若没有应答脉冲说明该器件不存在或者连接有问题等。

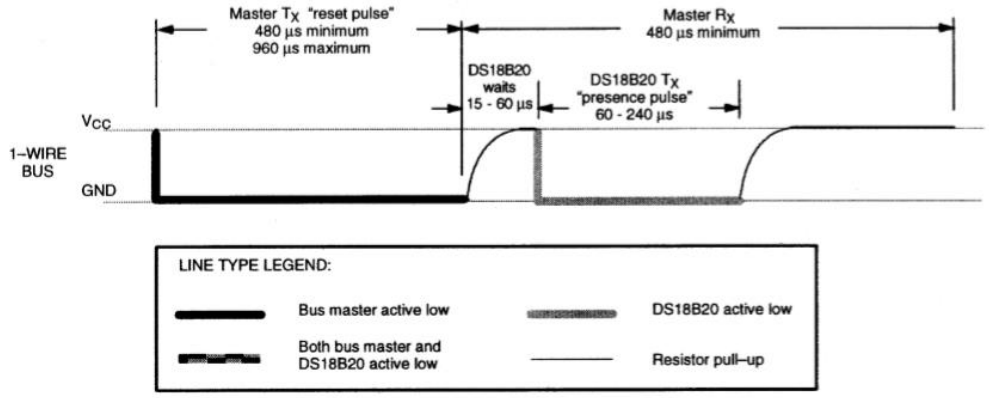


图 11 DS18B20 初始化时序

```
static void Init_DS18B20(void)
{
 uchar x=0;
 DQ_1 = 1;
 delay_18B20_1(8);
 DQ_1 = 0;                    //单片机将 DQ_1 拉低
 delay_18B20_1(250);          //精确延时  大于  480us
```

步骤一

```
 DQ_1 = 1;                  //拉高总线
 delay_18B20_1(10);
 x=DQ_1;                    //稍做延时后  如果 x=0 则初始化成功  x=1 则初始化失败
 delay_18B20_1(5);
}
```

步骤二

步骤三
复位检验

4.2 写时隙程序设计

在写时隙期间，主机向单总线器件写入数据。存在两种写时隙：写“1”和写“0”。主机采用写“1”时隙向从机写入 1，而采用写“0”向从机写入 0。所有写时隙至少需要 60μs，且在两次独立的写时隙之间至少需要 1μs 的恢复时间两种写时隙均起始于主机拉低总线，如图 12 所示。写时隙经过以下几个步骤：

写 1：

1. 主机拉低总线。
2. 在 15μs 内释放总线。
3. 写时隙开始后 15~60μs，DS18B20 采集总线电平状态 ，写入数据 1。
4. 拉回高电平，等待下一字节发送。

写 0：

1. 主机拉低总线。
2. 保持低电平。
3. 写时隙开始后 15~60μs，DS18B20 采集总线电平状态 ，写入数据 0。
4. 拉回高电平，等待下一字节发送。

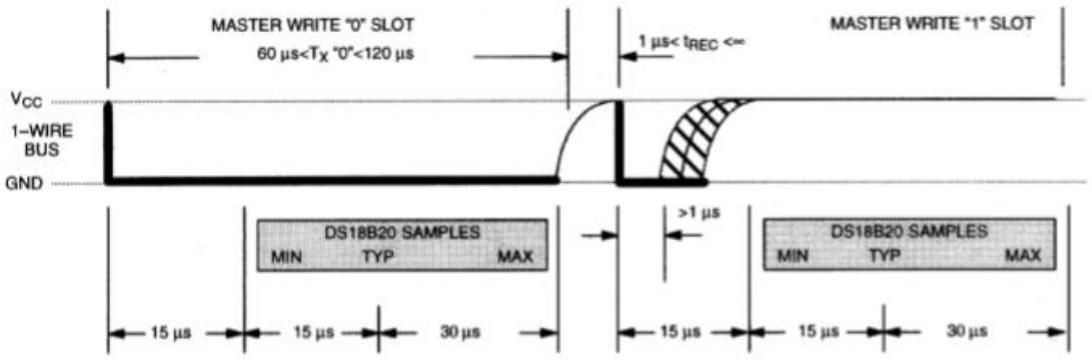


图 12 写时隙示意图

```
static void WriteOneChar(unsigned char dat)
{
 unsigned char i=0;
 for (i=8; i>0; i--)
 {
  DQ_1 = 0;
```

步骤一

```
   DQ_1 = dat&0x01;
   delay_18B20_1(5);

   DQ_1 = 1;

   dat>>=1;
 }
delay_18B20_1(5);
```

步骤二、三，将最低位送到总线，并延时 15μs

步骤四

循环八次，从低位到高位依次送出 8 个位

4.3 读时隙程序设计

在读时隙期间，主机读入来自从机的数据。在每一个时隙，总线只能传输以为数据。DS18B20 仅在主机发出读时隙时，才向主机传输数据，所以，在主机发出读数据命令之后，必须立即产生读时隙，一边从机能够传输数据。所有读时隙至少需要 60μs，且在两次独立的读时隙之间至少需要 1μs 的恢复时间。每个读时隙都由主机发起，至少拉低总线 1μs，如图 13 所示。在主机发起读时隙之后单总线才开始在总线上发送 0 或 1。步骤如下：

发送 1：

1. 保持总线为高电平。
2. 主机在 15μs 内采集数据。

发送 0：

1. 拉低总线。
2. 主机在 15μs 内采集数据。
3. 从机在 15μs 内释放总线，使之为高电平。

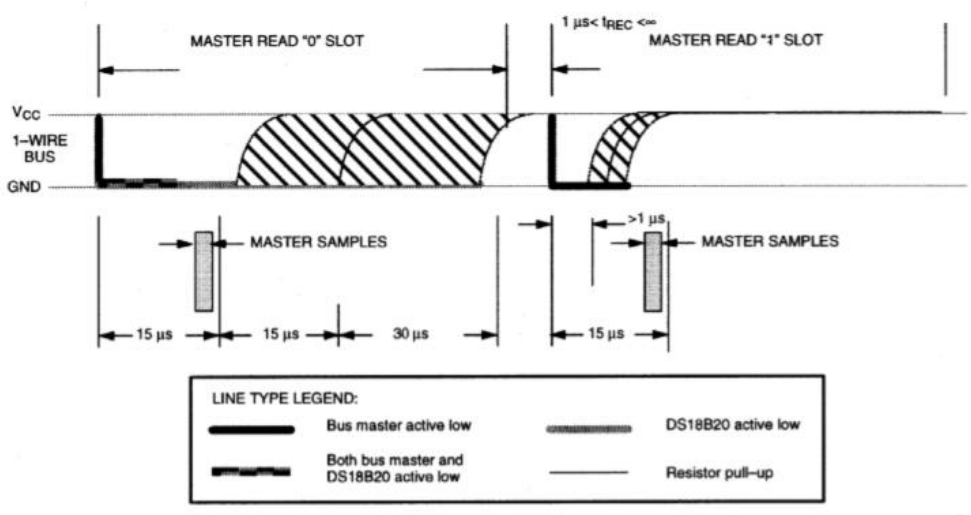


图 13 读时隙示意图

```
static unsigned char ReadOneChar(void)
{
unsigned char i=0;
unsigned char dat = 0;
for (i=8;i>0;i--)
 {
   DQ_1 = 0;                // 拉低总线
   dat>>=1;
   if(DQ_1)
   dat|=0x80;               //使最高位保持不变
   delay_18B20_1(5);
   DQ_1 = 1;                // 拉回高电平
 }
 return(dat);
```

步骤一，发起读时隙

步骤二，写位

步骤三

4.4 DS18B20 的温度选择显示

每个 DS18B20 有自己唯一的 ROM 码，用于对不同的 DS18B20 的选择判断，本系统选择将两个温度传感器接不同的 I\O 接口，省去了查找 ROM 的麻烦。单片机通过检测哪一个按键被按下，从而选择将哪一个温度传感器所连接的 P 口获得的温度数据显示到四位七段段共阳数码管上。选择程序如下，其中 flag_disp 为 0 时表示通道一，为 1 时表示通道二。

```
    if(flag_disp==0)
            {temperature=temperature_1;}
            else
            {temperature=temperature_2;}
```

结语：

基于 DS18B20 及数码管的两点温度测量显示装置这个课题较之前的课程设计难度大了很多，借阅了十几本书对本课题的讲解差距很大，不能从中找到太多的共通点，试着搬运两三个书上的电路图加程序到电脑上运行，全都失败，后来仔细研究阅读这几个仿真，都有很大问题，让我对出书的质量很费解，也耽误了很长很长的时间，虽然最终实现了功能，但是有两点很大的缺陷还没有修复，第一是 ROM 的查找，由于我是两点检测，如果两个 DS18B20 挂在同一条总线上，需要识别并匹配 ROM 来实现对不同温度传感器的分别显示。在这方面一直云里雾里，所以采用了两个 DS18B20 挂在两条总线的方法，避开了查找 ROM 的方式，感觉到这样的做法有悖老师的初衷。第二点是分辨率的设置，一般程序都是采用默认的 0.0625 的分辨率，于是我尝试去修改寄存器，在温度采集之前，将 0x5f 发送给温度传感器，并修改了采集后温度转化的算法，不管怎么修改，温度便显示出问题，这也是我另外没能解决的大问题。

参考文献：

【1】 吕俊.C 语言程序设计教程[m].南京大学出版社.2014 年 1 月

【2】 徐爱钧.单片机 C 语言编程与 proteus 仿真技术[m].中国工信出版社.2016 年 1 月

【3】 刘坤.51 单片机应用系统典型模块开发大全[m].中国铁道出版社.2013 年 11 月

【4】 徐爱钧.Keil C51 单片机高级语言应用编程技术[m]. 中国工信出版社.2015 年 10 月

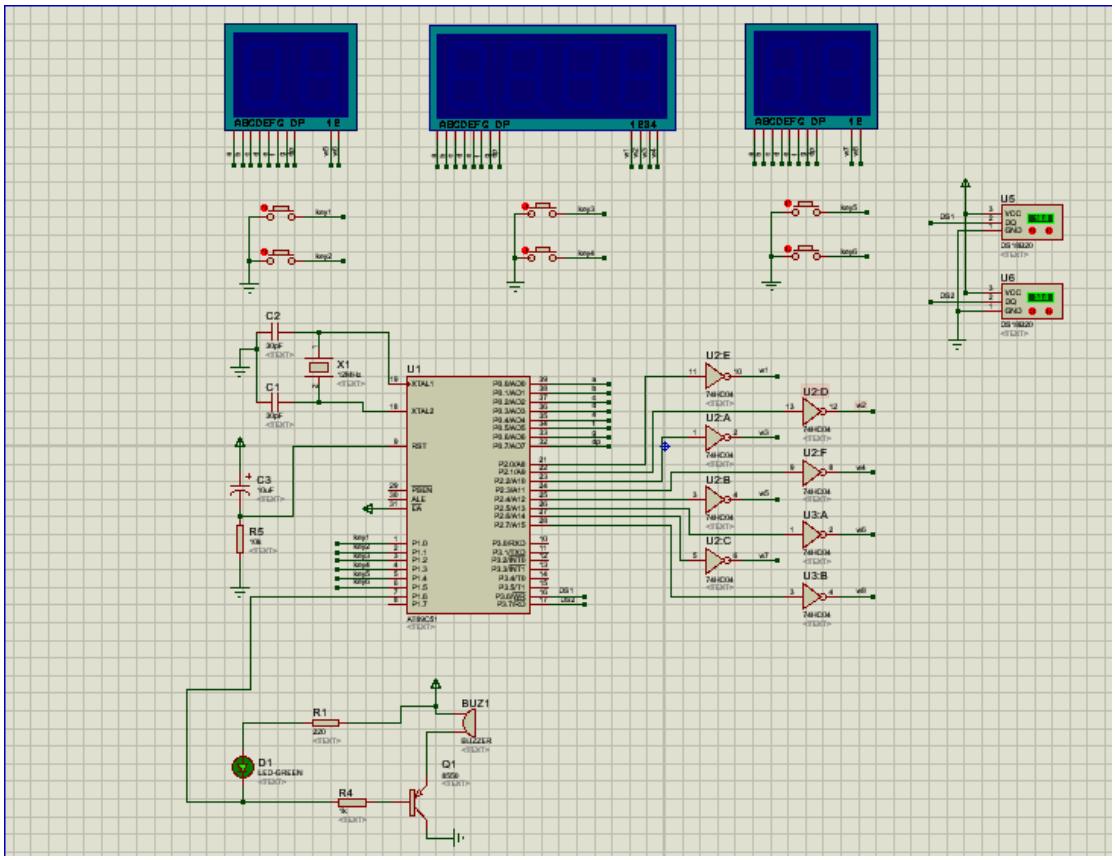
【5】 刘坤.51 单片机应用系统典型模块开发大全[m].中国铁道出版社.2013 年 2 月

【6】 王巧芝. C51 单片机开发应用---从入门到精通[m]. 中国铁道出版社.2011 年 4 月

【7】 刘波. C51 单片机应用开发典型范例---基于 proteus 仿真[m]. 电子工业出版社.2011 年 4 月

附录

电路图：



程序一：

```
#include <reg52.h>
#include <DS18B20_1.h>
#include <DS18B20_2.h>
#include <math.h>

#define uchar unsigned char
#define uint  unsigned  int
sbit BUTON1=P1^0;
sbit BUTON2=P1^1;
sbit BUTON3=P1^2;
sbit BUTON4=P1^3;
sbit BUTON5=P1^4;
sbit BUTON6=P1^5;
sbit w1=P2^0;
sbit w2=P2^1;
```

```
sbit w3=P2^2;
sbit w4=P2^3;
sbit w5=P2^4;
sbit w6=P2^5;
sbit w7=P2^6;
sbit w8=P2^7;
sbit bz=P1^6;
#define all_off {P2=0XFF;}

bit flag_disp;
uchar count,yuzhi1=15,yuzhi2=30;
int temperature_1,temperature_2,temperature;                //存放温度值
code uchar shuma[10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90}; //显示段码 共阳数码管
//延时函数
void delay(uint num)
{
    while( --num );
}
void disp()     //数码管显示函数
{
    if(flag_disp==0)                                    //显示第一路
    {
        if(flag_fu1==1)
        {all_off;P0=0xBF;w1=0;delay(150);}     //如果温度为负，在第一位显示“-”
        else
        {all_off;P0=0xFF;w1=0;delay(150);}
    }
    else //显示第二路
    {
        if(flag_fu2==1)       //负温度
        {all_off;P0=0xBF;w1=0;delay(150);}
        else
        {all_off;P0=0xFF;w1=0;delay(150);}
    }
    all_off;P0=shuma[temperature/1000];w2=0;delay(150); //带小数点位
    all_off;P0=shuma[temperature/100%10]&0x7F;w3=0;delay(150);//显示温度
    all_off;P0=shuma[temperature/10%10];w4=0;delay(150);

    all_off;P0=shuma[yuzhi1/10];w5=0;delay(150);//显示上限
    all_off;P0=shuma[yuzhi1%10];w6=0;delay(150);

    all_off;P0=shuma[yuzhi2/10];w7=0;delay(150);//显示下限
    all_off;P0=shuma[yuzhi2%10];w8=0;delay(150);
```

```
}
void BUTONscan()      //按键扫描函数
{
      if(!BUTON1)      //检测按下
      {
            delay(10); //延时消抖动
            if(!BUTON1)    //
            {
                  if(yuzhi1<99)yuzhi1++;
                  while(!BUTON1){;}//检测松手
            }
      }
      if(!BUTON2)      //检测按下
      {
            delay(10); //延时消抖动
            if(!BUTON2)    //
            {
                  if(yuzhi1>1)yuzhi1--;
                  while(!BUTON2){;}//检测松手
            }
      }
      if(!BUTON3)      //检测按下
      {
            delay(10); //延时消抖动
            if(!BUTON3)    //
            {
                  flag_disp=0;
                  while(!BUTON3){;}//检测松手
            }
      }
      if(!BUTON4)      //检测按下
      {
            delay(10); //延时消抖动
            if(!BUTON4)    //
            {
                  flag_disp=1;
                  while(!BUTON4){;}//检测松手
            }
      }
      if(!BUTON5)      //检测按下
      {
            delay(10); //延时消抖动
            if(!BUTON5)    //
```

```
            {
                if(yuzhi2<99)yuzhi2++;
                while(!BUTON5){;}//检测松手
            }
        }
        if(!BUTON6)       //检测按下
        {
            delay(10); //延时消抖动
            if(!BUTON6)    //
            {
                if(yuzhi2>1)yuzhi2--;
                while(!BUTON6){;}//检测松手
            }
        }
}
void main()
{
        TMOD |= 0x01;     //初始化定时器 0
        TL0 = 0x00;       //设置定时初值
        TH0 = 0xee;       //设置定时初值 5MS
        EA=1;
        ET0=1;
        TR0=0;

        while(1)
        {
            BUTONscan();      //按键扫描函数
            temperature_1=ReadTemperature_1();    //读温度 1
            temperature_2=ReadTemperature_2();    //读温度 2

            temperature_1 = abs(temperature_1);//取绝对值
            temperature_1=temperature_1%10000;//温度整数和 2 位小数(现在是十进制数)
            temperature_2 = abs(temperature_2);//取绝对值
            temperature_2=temperature_2%10000;//温度整数和 2 位小数(现在是十进制数)
            if(flag_disp==0)
            {temperature=temperature_1;}
            else
            {temperature=temperature_2;}
            if(flag_fu1==1 || flag_fu2==1 || temperature_1<yuzhi1*100 || temperature_1>yuzhi2*100
|| temperature_2<yuzhi1*100 || temperature_2>yuzhi2*100)
                bz=0;
            else
                bz=1;
            disp();                 //数码管显示
```

```
    }
}
void Tim() interrupt 1
{
    TL0 = 0x00;         //设置定时初值
    TH0 = 0xee;         //设置定时初值

    count++;
    switch(count)   //数码管动态显示
    {
        case 1:all_off;P0=shuma[temperature/1000];w1=0;
        break;
        case 2:all_off;P0=shuma[temperature/100%10]&0x7F;w2=0;
        break;
        case 3:all_off;P0=shuma[temperature/10%10];w3=0;
        break;
        case 4:all_off;P0=shuma[temperature%10];w4=0;
        break;
        case 5:all_off;P0=shuma[56/10];w5=0;
        break;
        case 6:all_off;P0=shuma[56%10];w6=0;
        break;
        case 7:all_off;P0=shuma[15/10];w7=0;
        break;
        case 8:all_off;P0=shuma[15%10];w8=0;count=0;
        break;
        default:break;
    }
}
```

程序二：

```
#ifndef _DS18B20_1_H_
#define _DS18B20_1_H_
#include <intrins.h>
#include<reg52.h>
#define uchar unsigned char
#define uint unsigned int
sbit DQ_1=P3^6;                 //ds18b20  端口
int ReadTemperature_1(void);
extern bit flag_fu1;
#endif
```

程序三：

```
#ifndef _DS18B20_2_H_
#define _DS18B20_2_H_
#include <intrins.h>
#include<reg52.h>
#define uchar unsigned char
#define uint unsigned int
sbit DQ_2=P3^7;                    //ds18b20  端口
int ReadTemperature_2(void);
extern bit flag_fu2;
#endif
```

程序四：

```
#include <DS18B20_1.h>

bit flag_fu1=0;
/****************以下是测温程序******************/
void delay_18B20_1(unsigned int i)//延时函数
{
  while(i--);
}
//18b20 初始化函数
static void Init_DS18B20(void)
{
  unsigned char x=0;
  DQ_1 = 1;       //DQ_1 复位
  delay_18B20_1(8);    //稍做延时
  DQ_1 = 0;       //单片机将 DQ_1 拉低
  delay_18B20_1(80); //精确延时  大于  480us
  DQ_1 = 1;       //拉高总线
  delay_18B20_1(10);
  x=DQ_1;           //稍做延时后  如果 x=0 则初始化成功  x=1 则初始化失败
  delay_18B20_1(5);
}

//读一个字节
static unsigned char ReadOneChar(void)
{
unsigned char i=0;
unsigned char dat = 0;
for (i=8;i>0;i--)
  {
    DQ_1 = 0; //  给脉冲信号
```

```
    dat>>=1;
    DQ_1 = 1; // 给脉冲信号
    if(DQ_1)
    dat|=0x80;
    delay_18B20_1(5);
  }
  return(dat);
}
//写一个字节
static void WriteOneChar(unsigned char dat)
{
  unsigned char i=0;
  for (i=8; i>0; i--)
  {
    DQ_1 = 0;
    DQ_1 = dat&0x01;
    delay_18B20_1(5);
    DQ_1 = 1;
    dat>>=1;
  }
delay_18B20_1(5);
}
//读取温度
int ReadTemperature_1(void)
{
      int value;                    //存放温度数值
      unsigned int tmpvalue;        //温度整合的中间辅助变量

      unsigned char a=0;            //存放温度低位
      unsigned char b=0;            //存放温度高位
      float t;

      Init_DS18B20();
      WriteOneChar(0xCC);           // 跳过读序号列号的操作
      WriteOneChar(0x44);           // 启动温度转换
      delay_18B20_1(100);
      Init_DS18B20();
      WriteOneChar(0xCC); //跳过读序号列号的操作
      WriteOneChar(0xBE); //读取温度寄存器等（共可读 9 个寄存器） 前两个就是温度

      a=ReadOneChar();
      b=ReadOneChar();

      if((b&0xf8)==0xf8) //高字节的高 5 位为 1 时，温度为负
```

```
        {flag_fu1=1;}
    else
        {flag_fu1=0;}

    //将高低两个字节合成一个整形变量
    //计算机中对于负数是利用补码来表示的
    //若是负值，读取出来的数值是用补码表示的，可直接赋值给 int 型的 value
    tmpvalue = b;
    tmpvalue <<= 8;
    tmpvalue |= a;
    value = tmpvalue;
    if(flag_fu1==1)   //温度为负值
    {
        value=(~value)+1; //取反再加 1
    }


    t = value * 0.0625;
    value = t * 100 + (value > 0 ? 0.5 : -0.5); //大于 0 加 0.5，小于 0 减 0.5

    return value;
}
```

程序五：

```
#include <DS18B20_2.h>

bit flag_fu2=0;
/***************以下是测温程序*****************/
void delay_18B20_2(unsigned int i)//延时函数
{
  while(i--);
}
//18b20 初始化函数
static void Init_DS18B20(void)
{
  unsigned char x=0;
  DQ_2 = 1;       //DQ_2 复位
  delay_18B20_2(8);   //稍做延时
  DQ_2 = 0;       //单片机将 DQ_2 拉低
  delay_18B20_2(80); //精确延时  大于  480us
  DQ_2 = 1;       //拉高总线
  delay_18B20_2(10);
```

```
  x=DQ_2;           //稍做延时后  如果 x=0 则初始化成功  x=1 则初始化失败
  delay_18B20_2(5);
}

//读一个字节
static unsigned char ReadOneChar(void)
{
unsigned char i=0;
unsigned char dat = 0;
for (i=8;i>0;i--)
  {
    DQ_2 = 0; //  给脉冲信号
    dat>>=1;
    DQ_2 = 1; //  给脉冲信号
    if(DQ_2)
      dat|=0x80;
    delay_18B20_2(5);
  }
  return(dat);
}
//写一个字节
static void WriteOneChar(unsigned char dat)
{
  unsigned char i=0;
  for (i=8; i>0; i--)
  {
    DQ_2 = 0;
    DQ_2 = dat&0x01;
    delay_18B20_2(5);
    DQ_2 = 1;
    dat>>=1;
  }
delay_18B20_2(5);
}
//读取温度
int ReadTemperature_2(void)
{
        int value; //存放温度数值
        unsigned int tmpvalue;

        unsigned char a=0;
        unsigned char b=0;
        float t;
```

```
    Init_DS18B20();
    WriteOneChar(0xCC); //  跳过读序号列号的操作
    WriteOneChar(0x44); //  启动温度转换
    delay_18B20_2(100);
    Init_DS18B20();
    WriteOneChar(0xCC); //跳过读序号列号的操作
    WriteOneChar(0xBE); //读取温度寄存器等（共可读 9 个寄存器） 前两个就是温度
    a=ReadOneChar();
    b=ReadOneChar();
    if((b&0xf8)==0xf8) //高字节的高 5 位为 1 时，温度为负
        {flag_fu2=1;}
    else
        {flag_fu2=0;}

    //将高低两个字节合成一个整形变量
    //计算机中对于负数是利用补码来表示的
    //若是负值，读取出来的数值是用补码表示的，可直接赋值给 int 型的 value
    tmpvalue = b;
    tmpvalue <<= 8;
    tmpvalue |= a;
    value = tmpvalue;
    if(flag_fu2==1)   //温度为负值
    {
        value=(~value)+1; //取反再加 1
    }

    //使用 DS18B20 的默认分辨率 12 位，精确度为 0.0625 度，即读回数据的最低位代表
0.0625 度
    t = value * 0.0625;

    value = t * 100 + (value > 0 ? 0.5 : -0.5); //大于 0 加 0.5，小于 0 减 0.5

    return value;
}
```