3.22 DS18B20 实验

STM32 虽然内部自带了温度传感器，但是因为芯片温升较大等问题，与实际温度差别较大，所以 ALIENTEK MiniSTM32 开发板还预留了目前最常用的数字温度传感器：DS18B20 的接口，用户只需要自己找一个 DS18B20 焊接上去，就可以使用了。本节将向大家介绍，如何在 ALIENTEK MiniSTM32 开发板上，通过 DS18B20 来读取环境温度值。本节分为如下几个部分：

3.22.1 DS18B20 简介
3.22.2 硬件设计
3.22.3 软件设计
3.22.4 下载与测试

3.22.1 DS18B20 简介

DS18B20 是由 DALLAS 半导体公司推出的一种的“一线总线”接口的温度传感器。与传统的热敏电阻等测温元件相比，它是一种新型的体积小、适用电压宽、与微处理器接口简单的数字化温度传感器。一线总线结构具有简洁且经济的特点，可使用户轻松地组建传感器网络，从而为测量系统的构建引入全新概念，测量温度范围为-55~+125℃ ，精度为±0．5℃。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。它能直接读出被测温度，并且可根据实际要求通过简单的编程实现 9~l2 位的数字值读数方式。它工作在 3—5．5 V 的电压范围，采用多种封装形式，从而使系统设计灵活、方便，设定分辨率及用户设定的报警温度存储在 EEPROM 中，掉电后依然保存。其内部结构见下图：

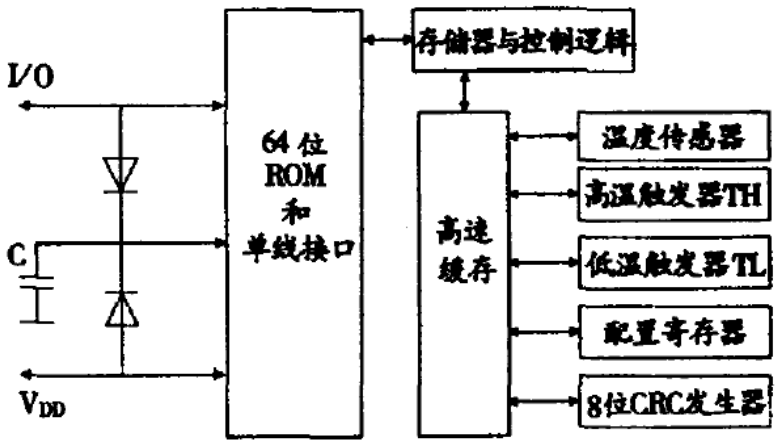

图 3.22.1.1 DS18B20 内部结构图

ROM 中的 64 位序列号是出厂前被光记好的，它可以看作是该 DS18B20 的地址序列码，每 DS18B20 的 64 位序列号均不相同。64 位 ROM 的排列是：前 8 位是产品家族码，接着 48 位是 DS18B20 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)。ROM 作用是使每一个 DS18B20 都各不相同，这样就可实现一根总线上挂接多个。

所有的单总线器件要求采用严格的信号时序，以保证数据的完整性。DS18B20 共有 6 种信号类型：复位脉冲、应答脉冲、写 0、写 1、读 0 和读 1。所有这些信号，除了应答脉冲以外，都由主机发出同步信号。并且发送所有的命令和数据都是字节的低位在前。这里我们简单介绍这几个信号的时序：

1）复位脉冲和应答脉冲

单总线上的所有通信都是以初始化序列开始。主机输出低电平，保持低电平时间至少 480 us,，以产生复位脉冲。接着主机释放总线，4.7K 的上拉电阻将单总线拉高，延时 15～60 us，并进入接收模式(Rx)。接着 DS18B20 拉低总线 60~240 us，以产生低电平应答脉冲，

若为低电平，再延时 480 us。

2）写时序

写时序包括写 0 时序和写 1 时序。所有写时序至少需要 60us，且在 2 次独立的写时序之间至少需要 1us 的恢复时间，两种写时序均起始于主机拉低总线。写 1 时序：主机输出低电平，延时 2us，然后释放总线，延时 60us。写 0 时序：主机输出低电平，延时 60us，然后释放总线，延时 2us。

3）读时序

单总线器件仅在主机发出读时序时，才向主机传输数据，所以，在主机发出读数据命令后，必须马上产生读时序，以便从机能够传输数据。所有读时序至少需要 60us，且在 2 次独立的读时序之间至少需要 1us 的恢复时间。每个读时序都由主机发起，至少拉低总线 1us。主机在读时序期间必须释放总线，并且在时序起始后的 15us 之内采样总线状态。典型的读时序过程为：主机输出低电平延时 2us，然后主机转入输入模式延时 12us，然后读取单总线当前的电平，然后延时 50us。

在了解了单总线时序之后，我们来看看 DS18B20 的典型温度读取过程，DS18B20 的典型温度读取过程为：复位→发 SKIP ROM 命令（0XCC）→发开始转换命令（0X44）→延时→复位→发送 SKIP ROM 命令（0XCC）→发读存储器命令（0XBE）→连续读出两个字节数据(即温度)->结束。

DS18B20 的介绍就到这里，更详细的介绍，请大家参考 DS18B20 的技术手册。

3.22.2 硬件设计

由于开发板上标准配置是没有 DS18B20 这个传感器的，只有接口，所以大家需要自己焊接一个 DS18B20 上去。

本节实验功能简介：开机的时候先检测是否有 DS18B20 存在，如果没有，则提示错误。只有在检测到 DS18B20 之后才开始读取温度并显示在 LCD 上，如果发现了 DS18B20，则程序每隔 200ms 左右读取一次数据，并把温度显示在 LCD 上。同样我们也是用 LED0 来指示程序正在运行。

所要用到的硬件资源如下：

1）STM32F103RBT6。

2）DS0（外部 LED0）。

3）TFTLCD 液晶模块。

4）DS18B20 温度传感器。

前三部分，在之前的实例已经介绍过了，DS18B20与STM32的连接电路则很简单，入下图所示：

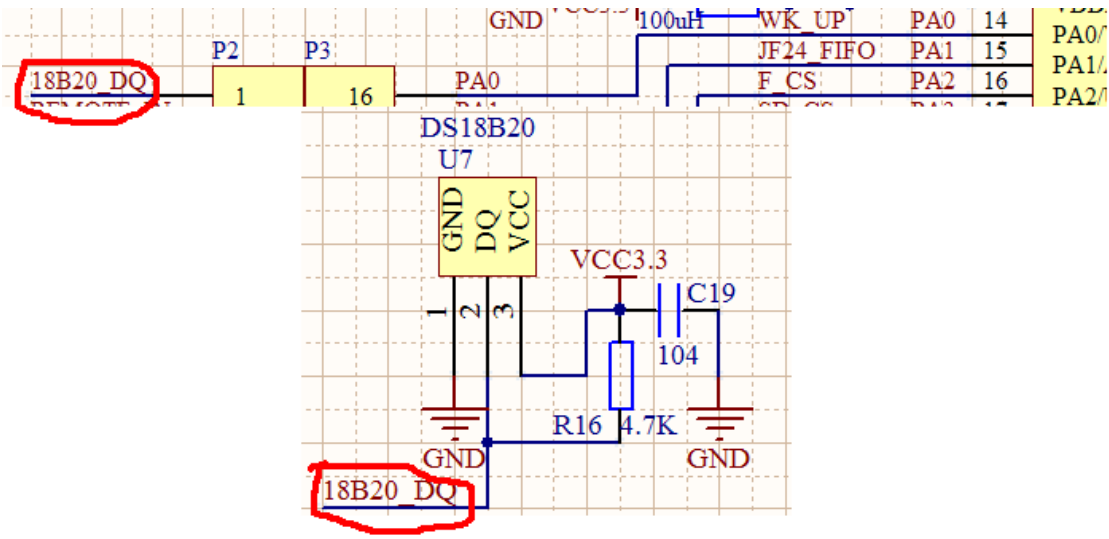

图 3.22.2.1 DS18B20 与 STM32 的连接电路图

DS18B20 与 STM32 的 PA0 通过跳线帽连接，这个在硬件上要连接上，否则 PA0 将得不到信号。

3.22.3 软件设计

打开上一节的工程，首先在 HARDWARE 文件夹下新建一个 DS18B20 的文件夹。然后新建一个 ds18b20.c 和 ds18b20.h 的文件保存在 DS18B20 文件夹下，并将这个文件夹加入头文件包含路径。

打开 ds18b20.c 在该文件下输入如下代码：

```
#include "ds18b20.h"
#include "delay.h"
//Mini STM32 开发板
//DS18B20  驱动函数
//正点原子@ALIENTEK
//2010/6/17

//复位 DS18B20
void DS18B20_Rst(void)
{
    DS18B20_IO_OUT(); //SET PA0 OUTPUT
    DS18B20_DQ_OUT=0; //拉低 DQ
    delay_us(750);    //拉低 750us
    DS18B20_DQ_OUT=1; //DQ=1
    delay_us(15);     //15US
}
//等待 DS18B20 的回应
//返回 1:未检测到 DS18B20 的存在
//返回 0:存在
u8 DS18B20_Check(void)
{
    u8 retry=0;
    DS18B20_IO_IN();//SET PA0 INPUT
    while (DS18B20_DQ_IN&&retry<200)
    {
        retry++;
        delay_us(1);
    };
    if(retry>=200)return 1;
    else retry=0;
    while (!DS18B20_DQ_IN&&retry<240)
    {
        retry++;
```

```
        delay_us(1);
    };
    if(retry>=240)return 1;
    return 0;
}
//从 DS18B20 读取一个位
//返回值：1/0
u8 DS18B20_Read_Bit(void)                    // read one bit
{
    u8 data;
    DS18B20_IO_OUT();//SET PA0 OUTPUT
    DS18B20_DQ_OUT=0;
    delay_us(2);
    DS18B20_DQ_OUT=1;
    DS18B20_IO_IN();//SET PA0 INPUT
    delay_us(12);
    if(DS18B20_DQ_IN)data=1;
    else data=0;
    delay_us(50);
    return data;
}
//从 DS18B20 读取一个字节
//返回值：读到的数据
u8 DS18B20_Read_Byte(void)    // read one byte
{
    u8 i,j,dat;
    dat=0;
    for (i=1;i<=8;i++)
    {
        j=DS18B20_Read_Bit();
        dat=(j<<7)|(dat>>1);
    }
    return dat;
}
//写一个字节到 DS18B20
//dat：要写入的字节
void DS18B20_Write_Byte(u8 dat)
 {
    u8 j;
    u8 testb;
    DS18B20_IO_OUT();//SET PA0 OUTPUT;
    for (j=1;j<=8;j++)
    {
```

```
            testb=dat&0x01;
            dat=dat>>1;
            if (testb)
            {
                DS18B20_DQ_OUT=0;// Write 1
                delay_us(2);
                DS18B20_DQ_OUT=1;
                delay_us(60);
            }
            else
            {
                DS18B20_DQ_OUT=0;// Write 0
                delay_us(60);
                DS18B20_DQ_OUT=1;
                delay_us(2);
            }
        }
}
//开始温度转换
void DS18B20_Start(void)// ds1820 start convert
{
    DS18B20_Rst();
    DS18B20_Check();
    DS18B20_Write_Byte(0xcc);// skip rom
    DS18B20_Write_Byte(0x44);// convert
}
//初始化 DS18B20 的 IO 口 DQ 同时检测 DS 的存在
//返回 1:不存在
//返回 0:存在
u8 DS18B20_Init(void)
{
    RCC->APB2ENR|=1<<2;      //使能 PORTA 口时钟
    GPIOA->CRL&=0XFFFFFFF0;//PORTA.0 推挽输出
    GPIOA->CRL|=0X00000003;
    GPIOA->ODR|=1<<0;          //输出 1
    DS18B20_Rst();
    return DS18B20_Check();
}
//从 ds18b20 得到温度值
//精度：0.1C
//返回值：温度值 （-550~1250）
short DS18B20_Get_Temp(void)
{
```

```
    u8 temp;
    u8 TL,TH;
    short tem;
    DS18B20_Start ();                       // ds1820 start convert
    DS18B20_Rst();
    DS18B20_Check();
    DS18B20_Write_Byte(0xcc);// skip rom
    DS18B20_Write_Byte(0xbe);// convert
    TL=DS18B20_Read_Byte(); // LSB
    TH=DS18B20_Read_Byte(); // MSB

    if(TH>7)
    {
        TH=~TH;
        TL=~TL;
        temp=0;//温度为负
    }else temp=1;//温度为正
    tem=TH; //获得高八位
    tem<<=8;
    tem+=TL;//获得底八位
    tem=(float)tem*0.625;//转换
    if(temp)return tem; //返回温度值
    else return -tem;
}
```

该部分代码就是根据我们前面介绍的单总线操作时序来读取 DS18B20 的温度值的，DS18B20 的温度通过 DS18B20_Get_Temp 函数读取，该函数的返回值为带符号的短整形数据，返回值的范围为-550~1250，其实就是温度值扩大了 10 倍。保存 ds18b20.c，并把该文件加入到 HARDWARE 组下，然后我们打开 ds18b20.h，在该文件下输入如下内容：

```
#ifndef __DS18B20_H
#define __DS18B20_H
#include "sys.h"
//Mini STM32 开发板
//DS18B20 驱动函数
//正点原子@ALIENTEK
//2010/6/17

//IO 方向设置
#define DS18B20_IO_IN()  {GPIOA->CRL&=0XFFFFFFF0;GPIOA->CRL|=8<<0;}
#define DS18B20_IO_OUT() {GPIOA->CRL&=0XFFFFFFF0;GPIOA->CRL|=3<<0;}
////IO 操作函数
#define DS18B20_DQ_OUT PAout(0) //数据端口  PA0
#define DS18B20_DQ_IN  PAin(0)  //数据端口  PA0
```

```
u8 DS18B20_Init(void);//初始化 DS18B20
short DS18B20_Get_Temp(void);//获取温度
void DS18B20_Start(void);//开始温度转换
void DS18B20_Write_Byte(u8 dat);//写入一个字节
u8 DS18B20_Read_Byte(void);//读出一个字节
u8 DS18B20_Read_Bit(void);//读出一个位
u8 DS18B20_Check(void);//检测是否存在 DS18B20
void DS18B20_Rst(void);//复位 DS18B20
#endif
```

关于这段代码，我们就不做多的解释了，接下来我们先保存这段代码，然后打开 test.c，在该文件下修改 main 函数如下：

```
int main(void)
{
    short temp;
    Stm32_Clock_Init(9);//系统时钟设置
    delay_init(72);     //延时初始化
    uart_init(72,9600); //串口 1 初始化
    LCD_Init();         //初始化液晶
    LED_Init();         //LED 初始化
    POINT_COLOR=RED;//设置字体为红色
    LCD_ShowString(60,50,"Mini STM32");
    LCD_ShowString(60,70,"DS18B20 TEST");
    LCD_ShowString(60,90,"ATOM@ALIENTEK");
    LCD_ShowString(60,110,"2010/6/17");
    while(DS18B20_Init())//初始化 DS18B20,兼检测 18B20
    {
        LCD_ShowString(60,130,"DS18B20 Check Failed!");
        delay_ms(500);
        LCD_ShowString(60,130,"Please Check!        ");
        delay_ms(500);
        LED0=!LED0;//DS0 闪烁
    }
    LCD_ShowString(60,130,"DS18B20 Ready!     ");
    POINT_COLOR=BLUE;//设置字体为蓝色
    LCD_ShowString(60,150, "Temperate:   . C");
    while(1)
    {
        temp=DS18B20_Get_Temp();
        if(temp<0)
        {
            temp=-temp;
            LCD_ShowChar(140,150,'-',16,0);//显示负号
        }
```

```
            LCD_ShowNum(148,150,temp/10,2,16);//显示温度值
            LCD_ShowNum(172,150,temp%10,1,16);//显示温度值
            //printf("t1:%d\n",temp);
            delay_ms(200);
            LED0=!LED0;
        }
    }
```

至此，我们本节的软件设计就结束了。

3.22.4 下载与测试

在代码编译成功之后，我们通过下载代码到 ALIENTEK MiniSTM32 开发板上，可以看到 LCD 显示开始显示当前的温度值（假定 DS18B20 已经焊接上去了），如下图所示：

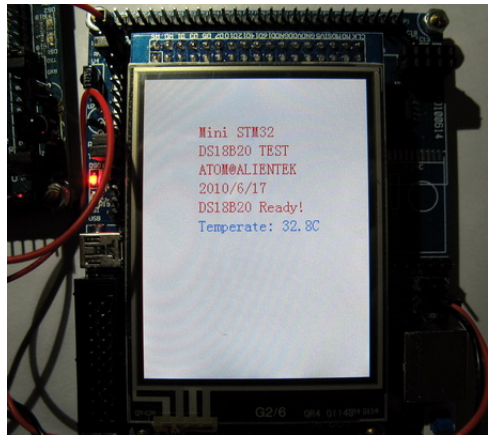

图 3.22.4.1 DS18B20 读取到的温度值

改程序还可以读取并显示负温度值的，只是由于作者在广州，是没办法看到了（除非放到冰箱），具备条件的大家可以测试一下。