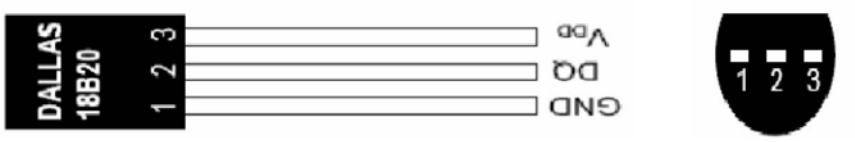
DS18B20 数字温度传感器

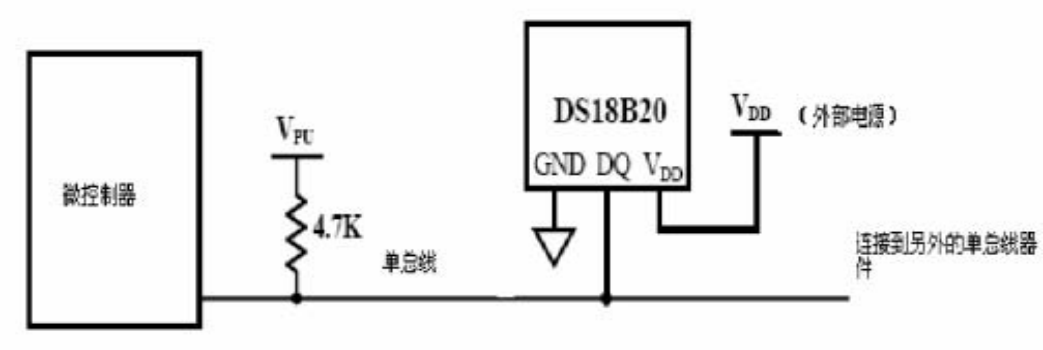
1.概念

本节是关于 DS18B20 实时数字温度传感器。模块上配的 DS18B20，一般上给人的感觉有点像三极管，其实 DS18B20 的内部结构与原理也挺复杂的，但是我们使用它，是为了实现温度传感的功能。

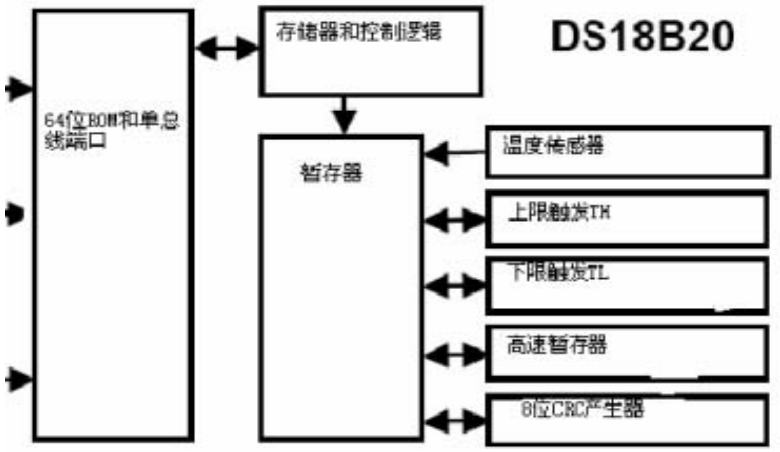
2.DS18B20介绍



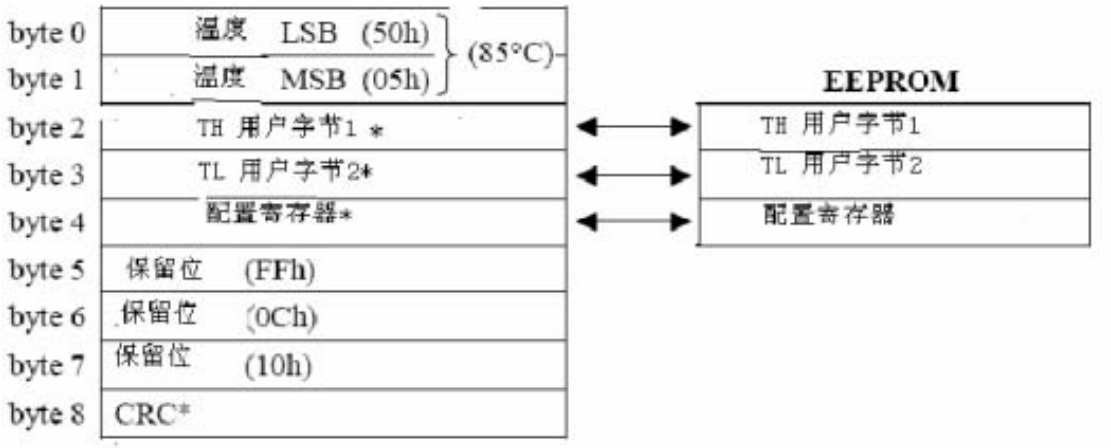
DS18B20 有三只引脚：VCC，DQ，和 VDD。



采用了外部供电的链接方式，而总线必须链接上拉电阻。此方式告诉我们，一总线在空置状态时，都是一直处于高电平的。



DS18B20的内部有 64位的 ROM单元和 9字节的暂存器单元。 64位ROM包含了DS18B20具有全球唯一的序列号。



以上是内部 9个字节的暂存单元（包括 EEPROM)。字节 0-1是转换好的温度。字节 2-3是用户用来设置最高报警和最低报警值。这个可以用软件来实现。字节4是用来配置转换精度，9-12位。字节5-8就不用看了。

字节: 0-1 转换好的温度

温度寄存器格式 图 2

	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

DS18B20 的温度操作是使用 16 位，也就是说分辨率是 0.0625。BIT15-BIT11 是符号位，为了表示转换的值是正数还是负数。

温度/数据关系 表 2

温度 ℃	数据输出（二进制）	数据输出（十六进制）
+125	0000 0111 1101 0000	07D0h
+85	0000 0101 0101 0000	0550h
+25.0625	0000 0001 1001 0001	0191h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-25.0625	1111 1110 0110 1111	FE6Fh
-55	1111 1100 1001 0000	FC90h

要求出正数的十进制值，必须将读取到的 LSB 字节，MSB 字节进行整合处理，然后乘以 0.0625 即可。

Exam 1: 假设从字节 0 读取到 0xD0 赋值于 Temp1，而字节 1 读取到 0x07 赋值于 Temp2，然后求出十进制值。

```
uint Temp1,Temp2,Temperature;
Temp1=0xD0; //低八位
Temp2=0x07; //高八位
Temperature =((Temp2<<8 ) | Temp1 ) *0.0625;
//或者 Temperature =(Temp1+Temp2*256) *0.0625;
//Temperature=125
```

在这里我们遇见了一个问题，就是如何求出负数的值，单片机不像人脑那样会心算，单片机要求我们必须判断 BIT11-15 是否为 1，然后人为置 1 负数标志。

Exam 2: 假设从字节 0 读取到 0x90 赋值于 Temp1，而字节 1 读取到 0xFC 赋值于 Temp2，然后求出该值是不是负数，和转换成十进制值。

```
uint Temp1,Temp2,Temperature;
bit Flag=0;
Temp1=0x90; //低八位
Temp2=0xFC; //高八位
//Temperature =(Temp1+Temp2*256)* 0.0625;
//Temperature=64656*0.0625
//很明显不是我们想要的答案
if((Temp2&0xFC)==1) //判断符号位是否为 1
{
    Flag=1; //负数标志置 1
    Temperature =((Temp2<<8 )|Temp1 ) //高八位第八位进行整合
    Temperature= ((~Temperature)+1); //求反，补一
    Temperature*= 0.0625; //求出十进制   //Temperature=55;
}
else
{
    Flag=0;
    Temperature = ((Temp2<<8 ) |Temp1 ) * 0.0625;
```

```
}
```

这个人为的负数标志，是很有用处的，看如何去利用它。

以上我们是求出没有小数点的正数。如果我要求出小数点的值的话，那么我应该这样做。

Exam 3: 假设从字节 0 读取到 0xA2 赋值于 Temp1，而字节 1 读取到 0x00 赋值于 Temp2，然后求出十进制值，要求连同小数点也求出。

```
unsigned int Temp1,Temp2,Temperature;
Temp1=0x90; //低八位
Temp2=0xFC; //高八位                          //实际值为 10.125
//Temperature =((Temp2<<8 )|Temp1 )*0.0625; //10，无小数点
//Temperature =((Temp2<<8 )|Temp1 )*(0.0625 *10); //101，一位小数点
//Temperature =((Temp2<<8 )|Temp1 )*(0.0625 *100); //1012，二位小数点
```

如以上的例题，我们可以先将 0.0625 乘以 10，然后再乘以整合后的 Temperature 变量，就可以求出后面一个小数点的值（求出更多的小数点，方法都是以此类推）。得出的结果是 101，然后再利用简单的算法，求出每一位的值。

```
uchar shi,ge,dot;
shi=Temperature/100; //1
ge=Temperature%100/10; //0
dot= Temperature %10; //1
```

求出负数的思路也一样，只不过多出人为置 1 负数标志，求反补一的计算而已。

字节 2-3: TH和TL配置

TH 与 TL 就是所谓的温度最高界限和温度最低界限的配置。其实这些可以使用软件来计算。

字节 4: 配置寄存器

配置寄存器 图 8

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	R1	R0	1	1	1	1	1

BIT7 出厂的时候就已经设置为 0,用户不建议去更改。而 R1 与 R0 位组合了四个不同的转换精度，00: 9 位转换精度而转换时间是 93.75ms，01: 10 位转换精度而转换时间是 187.5ms，10: 11 位转换精度而转换时间是 375ms，11: 12 位转换精度而转换时间是 750ms（默认）。该寄存器还是采用默认的，毕竟转换精度表示了转换的质量。

字节 5-7,8:保留位，CRC

不做要求

3.单片机访问 DS18B20

DS18B20 一般都是充当从机的角色，而单片机就是主机。

单片机通过一线总线访问 DS18B20 的话，需要经过以下几个步骤:

1.DS18B20 复位。

2.执行 ROM 指令。

3.DS18B20 功能指令（RAM 指令）。

一般上我们都是使用单点，也就是说单线总线上仅连接一个 DS18B20 存在。所以我们无需读取 ROM 里边的序列号来，然后匹配是哪个 DS18B20，而是更直接的跳过 ROM 指令，然后直接执行 DS18B20 功能指令。

DS18B20 复位：在某种意义上就是一次访问 DS18B20 的开始，或者可说成是开始信号。

ROM 指令: 也就是访问，搜索，匹配，DS18B20 的 64 位序列号的

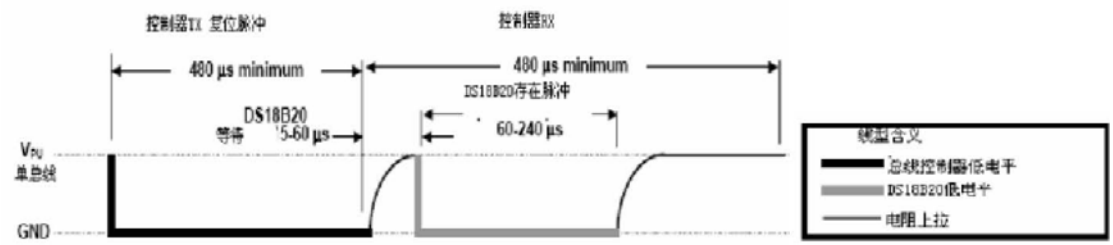
动作。在单点情况下，可以直接跳过 ROM 指令。而跳过 ROM 指令的字节是 0xCC。

DS18B20 功能指令有很多种，这里就不一一的介绍了，数据手册里有更详细的介绍。这里仅列出比较常用的几个 DS18B20 功能指令。

0x44: 开始转换温度。转换好的温度会储存到暂存器字节 0和 1。

0xEE: 读暂存指令。读暂存指令，会从暂存器 0到 9，一个一个字节读取，如果要停止的话，必须写下复位。

DS18B20复位



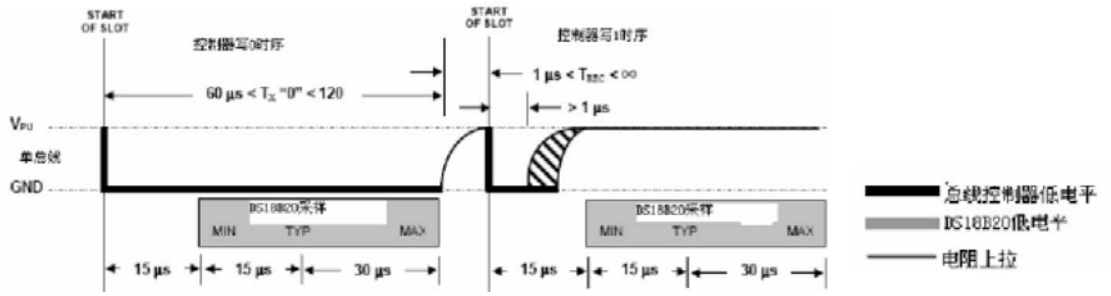
DS18B20的复位时序如下:

1. 480us-950us,然后释放总线（拉高电平）。
2. DS18B20会拉低信号，大约 60-240us表示应答。
3. DS18B20拉低电平的 60-240us之间，单片机读取总线的电平，如果是低电平，那么表示复位成功。
4. DS18B20拉低电平 60-240us之后，会释放总线。

```
/*************************************************************
初始化函数（开始复位函数）
*************************************************************/
void init()
{
    dq=1;
    delay_us(1);                    //拉高一段时间
    dq=0;
    delay_us(60);                   //延时 480us 以上
```

```
        dq=1;
        while(dq);                  //等待存在脉冲
        delay_us(10);               //存在脉冲存活时间
        dq=1;                       //拉高总线
    }
```

DS18B20读写逻辑0、1

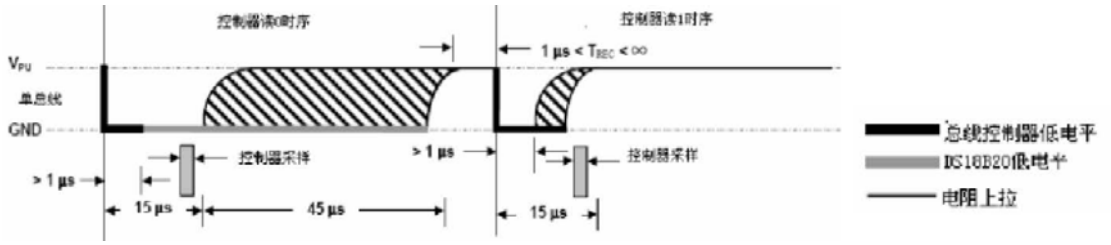


DS18B20 写逻辑 0 的步骤如下:

1. 单片机拉低电平大约 10-15us。
2. 单片机持续拉低电平大约 20-45us 的时间。
3. 释放总线。

DS18B20 写逻辑 1 的步骤如下:

1. 单片机拉低电平大约 10-15us。
2. 单片机拉高电平大约 20-45us 的时间。
3. 释放总线。



DS18B20 读逻辑 0 的步骤如下:

1. 在读取的时候单片机拉低电平大约 1us
2. 单片机释放总线，然后读取总线电平。
3. DS18B20会拉低电平。
4. 读取电平过后，延迟大约 40~45微妙

DS18B20 读逻辑 1 的步骤如下:

1. 在读取的时候单片机拉低电平大约 1us
2. 单片机释放总线，然后读取总线电平。
3. 这时候 DS18B20会拉高电平。
4. 读取电平过后，延迟大约 40~45微妙

如果要读或者写一个字节，就要重复以上的步骤八次。如以下的 C代码，使用 for循环，和数据变量的左移和或运算，实现一个字节读与写。

```
/*************************************************************************
    写指令函数，每次写入一个字节 dat
*************************************************************************/
    void write(uchar dat)
    {
        uchar i;
        for(i=0;i<8;i++)
        {
            dq=0;
```

```
            dq=dat&0x01;
            delay_us(2);
            dq=1;
            dat>>=1;
        }
    }
/*************************************************************************
    读函数，每次返回 16 位的温度值
*************************************************************************/
    uint read()
    {
        uchar i;
        uint dat;
    /*  for (i=8;i>0;i--)
        {
            dq = 0; // 给脉冲信号
            dat>>=1;
            dq = 1; // 给脉冲信号
            if(dq)
            dat|=0x80;
            delay_us(4);
        }*/
        for(i=0;i<16;i++)
        {
            dq=0;
            dq = 1;
            if(dq)
            {

                dat=(dat>>1)|0x8000;
            }
            else
                dat>>=1;
            dq=1;
            delay_us(1);     //
        }
        return(dat);
    }
```

注意： delay_us(1);函数延迟的时间，必须模拟非常准确，因为单线总线对时序的要求敏感。

4. 归纳一些重点: 单线总线高电平为闲置状态。

单片机访问 DS18B20 必须遵守:DS18B20 复位-->执行 ROM 指令-->执行 DS18B20 功能指令。而在单点上，可以直接跳过 ROM 指令。DS18B20 的转换精度默认为 12 位，而分辨率是 0.0625。

DS18B20 开始转换:

1.DS18B20 复位。

2.写入跳过 ROM 的字节命令，0xCC。

3.写入开始转换的功能命令，0x44。

4.延迟大约 750¯900毫秒。

DS18B20读暂存数据:

1. DS18B20复位。

2. 写入跳过 ROM的字节命令，0xCC。

3. 写入读暂存的功能命令，0xee。

4. 读入第 0个字节 LS Byte，转换结果的低八位。

5. 读入第 1个字节 MS Byte，转换结果的高八位。

数据求出十进制:

1.整合 LS Byte和 MS Byte的数据（可采用一次读出16位数据）

2.判断是否为正负数（可适当选择）

3.求得十进制值。正数乘以 0.0625，一位小数点乘以 0.625，二位小数点乘以 6.25。

4.十进制的各位求出。