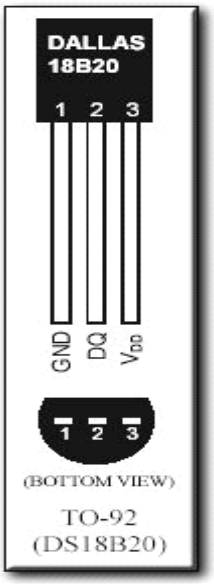


## 一、 DS18B20 的新性能

(1) 可用数据线供电，电压范围：3.0~5.5V；
(2) 测温范围：-55~+125℃，在-10~+85℃时精度为±0.5℃；
(3) 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℃；
(4) 12 位分辨率时最多在 750ms 内把温度值转换为数字；
(5) 负压特性：电源极性接反时，温度计不会因发热而烧毁，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 的管脚排列如下:

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）。

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。

| 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

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

（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 温度值分辨率设置表

| 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 |

**3. 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 | 0 |

根据 DS18B20 的通讯协议，主机控制 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，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，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。

| | |
|---|---|
| 温度值高位 | 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 检验 | 8 |

表 6 ROM 指令表

| 指 令 | 约定代码 | 功 能 |
|---|---|---|
| 读 ROM | 33H | 读 DS1820ROM 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 7 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，转换时最长为 500ms（典型为 200ms）。结果存入内部 9 字节 RAM 中。 |
| 读暂存器 | 0BEH | 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”。 |

4，时　序

主机使用时间隙(time slots)来读写 DSl820 的数据位和写命令字的位

(1)初始化

时序见图 2.25-2。主机总线 to 时刻发送一复位脉冲(最短为 480us 的低电平信号)，接着在 tl 时刻释放总线并进入接收状态，DSl820 在检测到总线的上升沿之后，等待 15-60us，接着 DS1820 在 t2 时刻发出存在脉冲(低电平，持续 60-240 us)，如图中虚线所示。

以下子程序在 MCS51 仿真机上通过，其晶振为 12M.。初始化子程序；

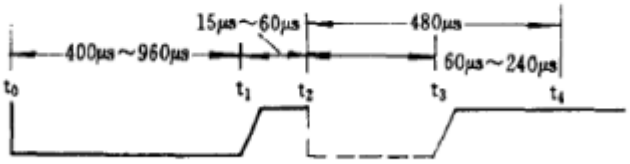


图 2.25-2　初始化时序

(2)写时间隙

当主机总线 t o 时刻从高拉至低电平时，就产生写时间隙，见图 2．25—3、图 2．25—4，从 to 时刻开始 15us 之内应将所需写的位送到总线上，DSl820 在 t。后 15-60us 间对总线采样。若低电平，写入的位是 0，见图 2．25—3；若高电平，写入的位是 1，见图 2．25—4。连续写 2 位间的间隙应大于 1us。

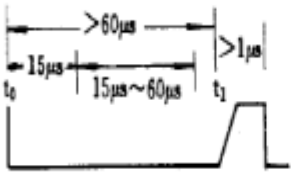


图 2.25-3　写 0 时序

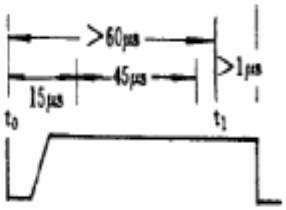


图 2.25-4　写 1 时序

(3)读时间隙

见图 2，25—5，主机总线 to 时刻从高拉至低电平时，总线只须保持低电平 1 7ts。之后在 t1 时刻将总线拉高，产生读时间隙，读时间隙在 t1 时刻后 t 2 时刻前有效。t z 距 to 为 15 捍 s，也就是说，t z 时刻前主机必须完成读位，并在 t o 后的 60 尸 s 一 120 fzs 内释放总线。读位子程序(读得的位到 C 中)：

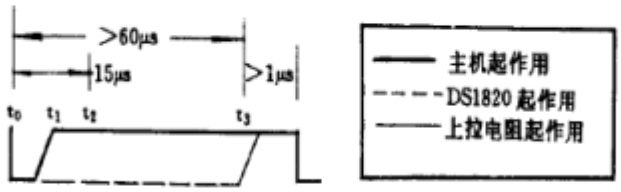


图 2.25-5 读时序

```
/*********************************************
*山东建筑大学 锐思实验室                      *
*函数名称：DS18B20 程序                       *
*CPU 型号：STC89C52                           *
*晶振参数：12MHZ                              *
*程序作者：2009 级机械 2 班 许方超            *
*联系方式：QQ：610949704                      *
*由于单纯的程序文件很难通过百度文库           *
*需要源码可以到                               *
* http://blog.sina.com.cn/xufangchao2010      *
*到博文里的模块程序中找到并复制               *
*********************************************/
```

**DS18B20.C**

```
#include "DS18B20.h"
uchar TMPL,TMPH;
sbit DQ=P2^3;
/*****************************
DS18B20 复位函数
1、数据线至高（短暂延时）
2、数据线拉低（延时 480-960UM）
3、数据线拉高
4、延时等待响应
5、将数据线再次拉高
*****************************/
uchar DS18B20_Reset()
{
                uchar x;
                 DQ=1 ;                    //数据线拉高
```

```
                DQ=0;                       //数据线拉低
                Delay_700us();              //延时 480us-960us
                DQ=1;                       //数据线拉低
                Delay_30us();               //延时 15us-60us
                x=DQ;                       //读取此时数据线的值
                while(!DQ);                 //直到 DQ 为高
                 return x;                  //返回复位结果
}
/**************************
18B20 读一位数据函数
1、将数据线拉高（延时 2 微妙）
2、将数据线拉低（延时 6 微妙）
3、将数据线拉高（延时 4 微妙）
4、读数据线状态，得一个状态位
5、延时 30 微秒
**************************/
uchar DS18B20_readbit()
{               bit  x ;
                uchar i;
                DQ=1;           //数据线拉高
                i++;            //延时 2 微秒
                DQ=0;           //数据线拉低
                Delay_6us(); //延时 6 微秒
                DQ=1;           //数据线拉高
                i++;i++;     //延时 4 微秒
                x=DQ;           //读数据线状态
                Delay_50us();//延时 50 微秒
                return (x); //返回数值

}
/**********************************
DS18B20 写一位函数
1、将数据线拉高
2、延时大于 1 微秒（两次写片段的间隙）
3、将数据线拉低（开始写时序 ）
4、延时大于 1 微秒（写时序的低电平延时大于 1 微秒）
5、写入数据  15 微秒后开始采集
6、延时使写周期在 60-120 微秒
**********************************/
void            DS18B20_writebit(uchar aa)
{               uchar i;
                DQ=1;           //将数据线拉高
                i++;            //延时大于 1 微秒
                DQ=0;           //将数据线拉低
```

```
                    i++;                    //延时大于 1 微秒
                    DQ=aa;                  //写入数据
                    Delay_50us();           //延时使写周期在 60-120 微秒之间
                    DQ=1;
}
/************************************
DS18B20 读一个字节函数
************************************/
uchar DS18B20_readbyte()
{
                    uchar i,j,dat;
                    dat=0;
                     j=1;
                    for(i=0;i<8;i++)
                    {
                     if(DS18B20_readbit())
                     {
                           dat=dat+(j<<i);
                     }
                    }
                    return (dat);
}
/*******************************
DS18B20 写一个字节函数
*******************************/
void DS18B20_writebyte(uchar dat)
{
                    uint temp;
                    uchar j;
                    for(j=0;j<8;j++)
                    {
                          temp=dat>>j;
                          temp=temp&0x01;
                          DS18B20_writebit(temp) ;
                    }

}
/***************************
DS18B20 读供电方式
***************************/
bit DS18B20_readpower()
{
                    bit x;
                    while(DS18B20_Reset());                    //复位，通信前必须复位
```

```
                DS18B20_writebyte(0xcc);
                DS18B20_writebyte(0xb4);                //读供电方式命令
                x=DQ;
                Delay_10us();
                return x;
}
/*****************************
DS18B20 精度设置函数
*****************************/
void DS18B20_SetResolution(unsigned char res)
{
    switch (res)
                {
                  case  9:res=0;
                  break;
                  case 10:res=1;
                  break;
                  case 11:res=2;
                  break;
                  case 12:res=3;
                  break;
                }
 while(DS18B20_Reset());                //复位，通信前必须复位
 DS18B20_writebyte(0x4e);               //写暂存器指令
 DS18B20_writebyte(0xff);               //此值被写入 TH
 DS18B20_writebyte(0xff);               //此值被写入 TL
 DS18B20_writebyte(0x1f|(res<<5));      //设置精度   0 res[1-0] 11111
}
/*****************************
DS18B20 温度读取函数
*****************************/
long DS18B20_GetTemperature()
{               long wendu;
                while(DS18B20_Reset());
                DS18B20_writebyte(0xcc);//跳过 ROM
                DS18B20_writebyte(0x44);//温度转换
                while(DS18B20_Reset());
                DS18B20_writebyte(0xcc);//跳过 ROM
                DS18B20_writebyte(0xbe);//读暂存器
                TMPL=DS18B20_readbyte();
                TMPH=DS18B20_readbyte();
                wendu=TMPH<<8|TMPL;
                wendu=wendu*0.0625;
                return wendu;
```

```
}
```

**DS18B20.H**

```
#ifndef   _DS18B20_H_
#define   _DS18B20_H_
#include "delay.h"
#include <reg52.h>
#define uchar unsigned char
#define uint unsigned int
uchar DS18B20_Reset();
uchar DS18B20_readbit();
void    DS18B20_writebit(uchar aa);
uchar DS18B20_readbyte();
void    DS18B20_writebyte(uchar dat);
bit     DS18B20_readpower();
void    DS18B20_SetResolution(unsigned char res);
long    DS18B20_GetTemperature();
#endif
```