DS18B20接口的C语言程序设计

■ 天津大学 顾振宇 刘鲁源 杜振辉

摘 要 DS18B20是DALLAS公司生产的一款数字温度传感器，具有精度高、全数字化、连线少等优点；但其I/O时序要求严格，使大多数编程人员不得不用汇编语言编写接口程序。本文介绍DS18B20数字温度传感器的C51接口程序及其编程方法和编程思路。

关键词 DS18B20 C51 数字温度传感器 接口

引 言

DS18B20数字温度传感器是美国DALLAS公司推出的DS1820系列数字温度传感器中性能优异的一款，具有诸多优点：

① 精度高。12位二进制转换结果，确保±0.5℃的精度和0.0625℃的分辨率。

② 全数字化。直接将数字信号传给CPU，传输可靠，避免了模拟方式的干扰问题。

③ 连线少。仅有3根连线：+5 V电源、地线和1根数字I/O总线。如采用寄生电源方式，DS18B20会从数字I/O总线获取寄生电源，则仅连接I/O线和地线即可。

虽然DS18B20有诸多优点，但使用起来并非易事。由于采用单总线数据传输方式，DS18B20的数据I/O均由同一条线完成，因此，对读写的操作时序要求严格。一般情况下需要用汇编语言编写接口程序[1]，而如今单片机编程已广泛采用C语言。在分析了C51所编译生成的程序代码的基础上，依据C51的编译特点，采用C51编写了该数字温度传感器的接口程序。该程序采用Keil C51 v6.12编译通过，在采用12 MHz晶振的PHILIPS P89C51RD+单片机上试运行，完全可以满足单线数据传输的时序要求。

1 精确延时问题

为保证DS18B20的严格I/O时序，需要作较精确的延时。在DS18B20的操作中，延时分两种：短时间延时和较长时间延时。短时间延时指10 μs以下的延时，在汇编语言下采用若干个NOP指令即可。因C51提供了若干内部函数，_nop_()函数为其中之一，其编译结果就是在对应位置嵌入一个nop汇编指令，因此，短时间延时可利用_nop_()函数实现。较长时间延时指10 μs以上的延时。在DS18B20操作中，用到的较长时间延时有15 μs、90 μs、270 μs、540 μs等。因这些延时均为15 μs的整数倍，因此可编写一个Delay15（n）函数，用该函数进行大约15 μs × n的延时，源码如下：

```
//功能：延时，延时时间n × 15 μs
//输入：延时时间
//返回：无
void Delay15(n)
    unsigned char n;    //延时参数
{
    do{
        _nop_ (); //01
        _nop_ (); //02
        ⋮                  } 共13个_nop();_
        _nop_ (); //13
        n--;
    }while(n);
}
```

该函数编译生成的汇编为：

```
        ; 函数 _Delay15 (开始)
;---- 变量 'n' 分配到寄存器 'R7' ----
0000        ?C0003:
0000 00         NOP
0001 00         NOP     } 共13个NOP
    ⋮
000C 00         NOP
001C DFE2   DJNZ    R7,?C0003
001E 22         RET
        ; 函数 _Delay15 (结束)
```

从上面的C51源码与生成汇编的对比中不难发现，C51程序中的do{ n--;} while (n); 结构仅生成了一条DJNZ R7指令。这样一条DJNZ指令用去2 μs，加上13个NOP，正好15 μs一个循环；而

函数的循环圈数即为R7中的参数n。这样，就得到了一个大约15 μs × n的延时。这里没有将函数的调用、返回及参数传递的时间计算在内，但足以满足使用要求。如果需要，可以作更精细的计算。

2 底层基本操作

有了比较精确的延时保证，就可以对DS18B20进行底层基本操作了。DS18B20的底层基本操作有3个。

（1）初始化

初始化是DS18B20的底层基本操作之一。通过单线总线进行的所有操作都从一个初始化序列开始。初始化序列包括一个由CPU发出的复位脉冲及其后由DS18B20发出的存在脉冲。存在脉冲让CPU知道DS18B20在总线上且已做好操作准备。有了前面的延时函数，初始化实现起来很简单，源码如下：

```
//功能: 初始化 DS18B20,读存在脉冲,无存在脉冲则置位错误标志
//输入: 无
//返回: 无
void RST18B20(void){
    DS18B20=0;                      //复位脉冲
    Delay15(36);                    //延时 540 μs
    DS18B20=1;                      //恢复
    Delay15(6);                     //延时 90 μs
    Error_DS18B20=DS18B20;          //读存在脉冲
    Delay15(18);                    //延时 270 μs
}
```

（2）数据写

数据写是DS18B20的底层基本操作之一，所有的指令、数据发送均由该操作完成。DS18B20的写操作都是逐位进行的，因此，采用C51中的位右移操作来实现。源码如下：

```
//功能:写 DS18B20
//输入:待写字节
//返回:无
void WR18B20(d)
unsigned char d;        //待写的1字节数据
{   unsigned char i;        //循环变量
    ACC=d;
    for(i=8;i>0;i--) {
        DS18B20=0;      //起始
        Delay15(1);     //延时
        ACC=ACC>>1;     //将第i位待发数据送入CY
        DS18B20=CY;     //送出数据
        Delay15(1);     //延时
        DS18B20=1;      //停止
    }
}
```

其中ACC=ACC>>1;指令所生成的汇编为：

```
0011C3  CLR   C
0012 13  RRC   A
```

这样放入ACC中的数据的最后一位就转入CY寄存器。利用该操作即可逐位将数据取出，发送出去。

（3）数据读

数据读是DS18B20的底层基本操作之一，温度值和其它状态信息的传回均由该操作完成。起初打算采用与逐位写相同的方式编制逐位读的函数，将数据读入CY寄存器后再利用位右移操作将数据逐位送入ACC；但实际写出的代码却不能正常工作。经分析C51所生成的汇编代码位发现：位右移操作“>>”所生成的汇编总是先清CY寄存器，再进行右移，这样数据在被送入ACC前就已经被清掉了。为了实现数据的逐位读，利用ACC的位寻址功能，在C51中将ACC的最高位定义为BIT7，然后利用它来实现逐位数据读功能，源码如下：

```
//功能: 读 DS18B20
//输入: 无
//返回: 读出的 1 字节数据
unsigned char RD18B20(void){
    unsigned char i;        //循环变量
    ACC=0;                  //清 ACC
    for(i=8;i>0;i--) {
        ACC=ACC>>1;     //右移位
        DS18B20=0;      //起始
        _nop_ ();       //延时
        DS18B20=1;      //恢复
        Delay15(1);     //延时
        BIT7=DS18B20;   //读第 i 位
    }
    return(ACC);            //返回 1 字节数据
}
```

上面的函数由于利用ACC的位寻址能力直接将数据置入，避开了CY寄存器，因此成功地将数据读出。

3 基本指令

DS18B20提供了一系列指令来控制传感器的工作。下面只简单介绍所用到的最基本的几条。

（1）Skip ROM [CCh]

用于1条I/O总线上只挂1个DS18B20的情况，使DS18B20跳过多个传感器的识别过程。如果一条I/O总线上挂了不止1个传感器，总线上就会发生数据冲突。

（2）Convert T [44h]

启动一次温度转换过程。温度转换命令被执行后，DS18B20保持等待状态。

（3）Read Scratchpad [BEh]

用于读取暂存器的内容。温度转换的结果和其它状态信息均以此命令读出。读取将从字节0开始，一直进行下去，直到字节8读完。如果不想读完所有字节，控制器可以在任何时间发出复位命令来中止读取。DS18B20的暂存器结构如图1所示。

暂存器

温度值低字节	0
温度值高字节	1
报警上限	2
报警下限	3
配置字	4
保留	5
保留	6
保留	7
校验	8

图1 暂存器结构

4 总体实现

有了上述几个基本操作指令，就可以对DS18B20进行操作了。为了操作简便，可编写两个操作函数，源码如下：

```
//功能: 启动DS18B20的1次温度转换
//输入: 无
//返回: 无
void ConvertT(void){
    RST18B20();       //初始化
    WR18B20(0xcc);  //Skip ROM，跳过多传感器识别
    WR18B20(0x44);  //Convert T，启动温度转换
}
//功能: 读取DS18B20并返回温度值
//输入: 无
//返回: DPTR-温度值,2字节
int ReadT(void){
    RST18B20();       //初始化
    WR18B20(0xcc);  //skip ROM，跳过多传感器识别
    WR18B20(0xbe);  //read scratchpad，读DS18B20暂存器
    DPL=RD18B20(); //温度值低位
    DPH=RD18B20();//温度值高位
    return(DPTR);     //返回读出的温度值,2字节
}
```

在编制温度读取函数时遇到了一个问题：如何将两个单字节数据合并为一个双字节数据？在汇编下这很容易，但在C51下似乎只有先将高字节乘256再加上低字节这一办法，但效率不高。于是利用C51的sfr16指令将DPH与DPL定义成一个16位特殊功能寄存器DPTR，然后，利用它来合并两个字节数据[2]。

有了上述两个函数，即可方便地获取当前温度。只需先用ConvertT()函数启动温度转换，再经过足够的转换时间（>750 ms）后用ReadT()函数读取温度即可。读出数值为一有符号整型数据，其数值以1/16 ℃为分度单位，尚需转换。

结束语

通过上述的编程过程发现，许多时候并不是非要用汇编语言不可的。在分析C51编译的汇编代码的前提下，充分利用C51的编译特点和各项功能，完全可以编制出准确而高效的程序代码，其效率比汇编差不了多少。ME

参考文献

1 蒋晴霞. DS18B20在桥梁混凝土测温中的应用. 单片机与嵌入式系统应用，2001(9):56~59

2 马忠梅. 单片机的C语言应用程序设计. 北京：北京航空航天大学出版社，1997

3 DALLAS Semiconductor Data Sheets CD-ROM