第二讲 LED 数码管的学习

提要：主要学习 Atmega8 通用数字 I/O 接口控制 LED 数码管的应用。

2.1 I/O 口的结构及特点：

Atmega8 有 23 个 I/O 引脚，分成 3 个 8 位的端口 B、C 和 D，其中 C 口只有 7 位。I/O 端口作为通用数字输入/输出口使用时，都具备真正的读-修改-写（R-M-W）特性。每个 I/O 引脚采用推挽式驱动，不仅能提供大电流的输出驱动，而且也可以吸收 20mA 的电流，因而能直接驱动 LED 显示器。Atmega8 采用 3 个 8 位寄存器来控制 I/O 端口，它们分别是方向寄存器 DDRx，数据寄存器 PORTx 和输入引脚寄存器 PINx（x 为 B 或 C 或 D，分别代表 B 口、C 口或 D 口；n 为 0~7，代表寄存器中的位置），其中 DDRx 和 PORTx 是可读写寄存器，而 PINx 为只读寄存器。每个 I/O 引脚内部都有独立的上拉电阻电路，可通过程序设置内部上拉电阻是否有效。

方向寄存器 DDRx 中的每个位用于控制 I/O 口一个引脚的输入输出方向，即控制 I/O 口的工作模式为输出模式还是输入模式。

当 DDRxn=1 时，I/O 的 Pxn 引脚处于输出模式。此时当 PORTxn=1 时，I/O 引脚呈高电平，同时可提供输出 20mA 的电流；当 PORTxn=0 时，I/O 引脚呈低电平，同时可吸收 20mA 的电流。

当 DDRxn=0 时，I/O 的 Pxn 引脚处于输入模式。此时引脚寄存器 PINxn 中的数据就是外部引脚的实际电平。此时可通过 PORTxn 的设置可控制内部的上拉电阻使用或不使用。

表 2-1 I/O 口设置表（n=7,6,…,1,0）

DDRxn	PORTxn	PUD	I/O 模式	内部上拉电阻	引脚状态说明
0	0	X	输入	无效	三态（高阻）
0	1	0	输入	有效	外部引脚拉低时输出电流
0	1	1	输入	无效	三态（高阻）
1	0	X	输出	无效	低电平推挽输出，吸收电流(≤20mA)
1	1	X	输出	无效	高电平推挽输出，输出电流(≤20mA)

通用 I/O 口主要特点：

1 都具备真正的读-修改-写（R-M-W）。

2 双向，可独立位控制。

3 大电流驱动(≤20mA)，可直接驱动 LED。

4 可控制上拉电阻。

注意事项：

1 使用 AVR 的 I/O 口，首先应正确设置其工作模式（输入模式还是输出模式），设置 DDRx。

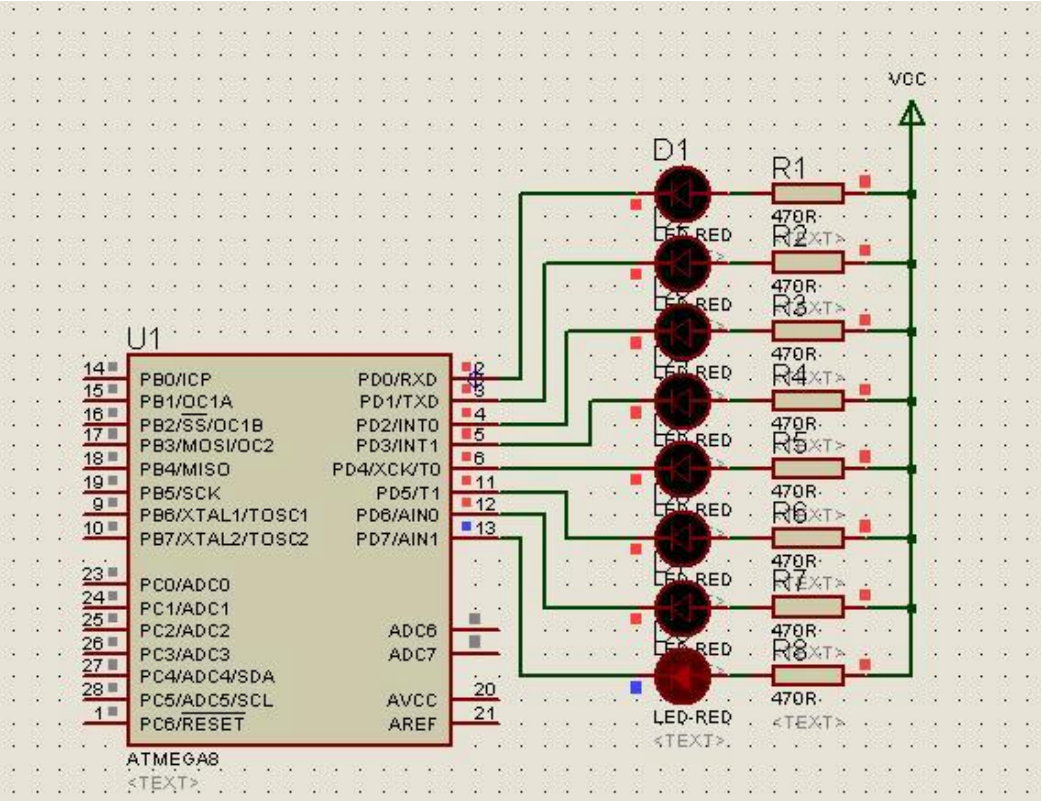
2 当 I/O 工作在输入模式(DDRxn=0)时，读取引脚上的电平应取 PINxn 的值，而不是 PORTxn 的值。

3 当 I/O 口工作在输入模式(DDRxn=0)时，应根据实际情况设置内部上拉电阻，利用内部上拉电阻可以节省外部上拉电阻。

4 将 I/O 空工作模式由输出模式设置为输入模式后，必须等待一个时钟周期后才能正确的读到外部引脚的值。

2.2 跑马灯程序控制发光二极管

回顾了这么久的理论，下面我们也做一个例子，功能是用 PD 口控制 8 个发光二极管循环点亮，形成“跑马灯”。试着在 Proteus 中画出电路图，看是否和我的一样呢，不一样也不怕，也许你比我画的更好呢？



在 ICC 中编写 C 程序，先把你的想法写出来，编译在 Proteus 跑一下，看你的“跑马灯” 是否真的跑起来了，下面是我的参考程序：

```
/*****************************************/
//文件名：HorseLight.c
//功能：  LED的控制
//作者：  young
//时间：  2006.10.1
//目标MCU:ATmega8
//晶振：  8MHZ
/*****************************************/
#include <iom8v.h>

//延时1MS
void delay_1ms(void)
{
    unsigned int i;
    for(i=1;i<(unsigned int)(1144-2);i++)
    ;
}

//延时nMS
void delay_ms(unsigned int n)
{
    unsigned int i=0;
    while(i<n)
    {
        delay_1ms();
        i++;
    }
}
```

```
//主函数，依次顺序打开LED
void main()
{
    unsigned char i;
    DDRD=0xFF;  //设置D口为输出模式
    PORTD=0xFF; //置高电平

    while (1)
    {
        for(i=0;i<8;i++) //顺序打开LED
        {
            PORTD=~(1<<i);
            delay_ms(200);
        }
     }
}
```

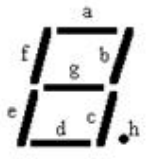
怎么样，你的灯亮了吗，很兴奋吧，很有成就感吧。现在你已经开始了解AVR单片机了，同时还掌握了两个软件的使用。

想一想：如果改变delay_ms的时间会变的怎样呢；如果直接让PORTD=(1<<i)呢；能不能添加或修改程序，改变灯亮的顺序和时间呢？我相信你都能完成的。

2.3 单个 LED 数码管练习

好了，接下来我们进一步学习 LED 数码管的控制，在进行新的实验之前，我们先了解一下 LED 数码管。

其实 LED 数码管的应用就在我们周围，在电梯上和小家电上我们经常会看到它，LED 数码管显示器内部由七个条形发光二极管和一个小圆点发光二极管组成，每个发光二极管称为一字段。因而它的控制原理和发光二极管的控制原理是相同的。根据各管的接线形式，可分成共阴极型和共阳极型。

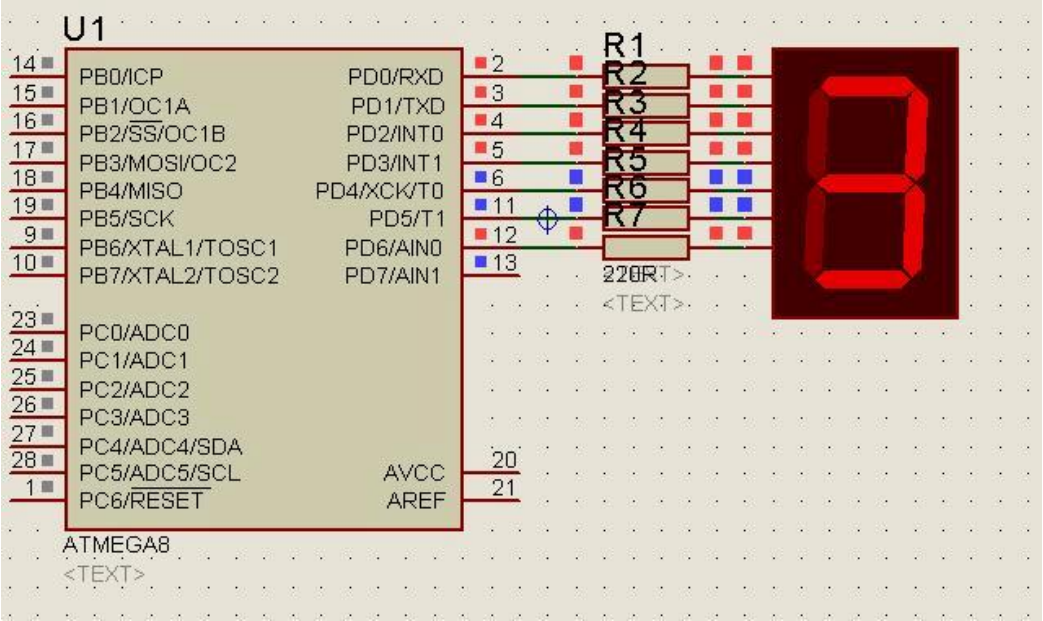


给 LED 数码管的七个发光二极管加不同的电平，二极管显示不同亮暗的组合就可以形成不同的字形，这种组合称之为字形码。下面以 1 为高电平，0 为低电平，给出字形码表。

显示字型	h	g	f	e	d	c	b	a	共阴极字形码	共阳极字形码
0	0	0	1	1	1	1	1	1	0x3F	0xC0
1	0	0	0	0	0	1	1	0	0x06	0xF9
2	0	1	0	1	1	0	1	1	0x5B	0xA4
3	0	1	0	0	1	1	1	1	0x4F	0xB0
4	0	1	1	0	0	1	1	0	0x66	0x99
5	0	1	1	0	1	1	0	1	0x6D	0x92
6	0	1	1	1	1	1	0	1	0x7D	0x82
7	0	0	0	0	0	1	1	1	0x07	0xF8
8	0	1	1	1	1	1	1	1	0x7F	0x80
9	0	1	1	0	1	1	1	1	0x6F	0x90
A	0	1	1	1	0	1	1	1	0x77	0x88
b	0	1	1	1	1	1	0	0	0x7C	0x83
C	0	0	1	1	1	0	0	1	0x39	0xC6
d	0	1	0	1	1	1	1	0	0x5E	0xA1
E	0	1	1	1	1	0	0	1	0x79	0x86
F	0	1	1	1	0	0	0	1	0x71	0x8E

有了字形码表，我们就可以编程了。如果还用 PD 口来控制一个 LED 数码管的八个字段，那么当要显示 0~F 的一个字型时，我们给 PD 口输出对应的字形码就可以了。如要在共阴极的数码管显示 6，则 PD 口输出 0x7D 就可以了；要显示 9 则 PD 口输出 0x6F。

下面我们就做一个用 LED 数码管显示数字 0~9 的例子，我们还用 PD 口控制，循环显示，间隔 500MS 显示一个数字。好想想我们怎么用 Proteus 画出电路呢，我们的程序又该怎么设计呢。下面我给出自己的参考电路：



也给出我的 C 语言主程序的参考，延时函数还是用上面的，这个延时不是很准确，要用精确的延时我们后面会学到定时器，这里我们就先用软件延时。

```
//主函数，依次显示数字0~9
void main()
{
    unsigned char i;
    unsigned char num[10]={0x3F, 0x06, 0x5B, 0x4F, 0x66,
                           0x6D, 0x7D, 0x07, 0x7F, 0x6F};
    DDRD=0xFF;  //设置D口为输出模式
    PORTD=0xFF; //置高电平

    while (1)
    {
        for(i=0;i<10;i++) //依次显示0~9
        {
            PORTD=num[i];
            delay_ms(500);
        }
     }
}
```

怎么样你的数字也显示出来了吗，我想一定显示的很好吧。那么要是我想显示 A~F 怎么办呢；能不能显示象 H，L 一类的字母呢；改变了 delay_ms 函数的延时时间会怎样呢；要显示小数点我们应该怎么办呢？我想你都能帮我找出答案吧，那还不快点去试试。

2.4 多个 LED 数码管实验

学会用一个 LED 数码管显示数字了，一定很高兴吧，接下来我们就要试用多个数码管显示数字了，我想你住的楼房的层数也快超过 10 的限制了吧。

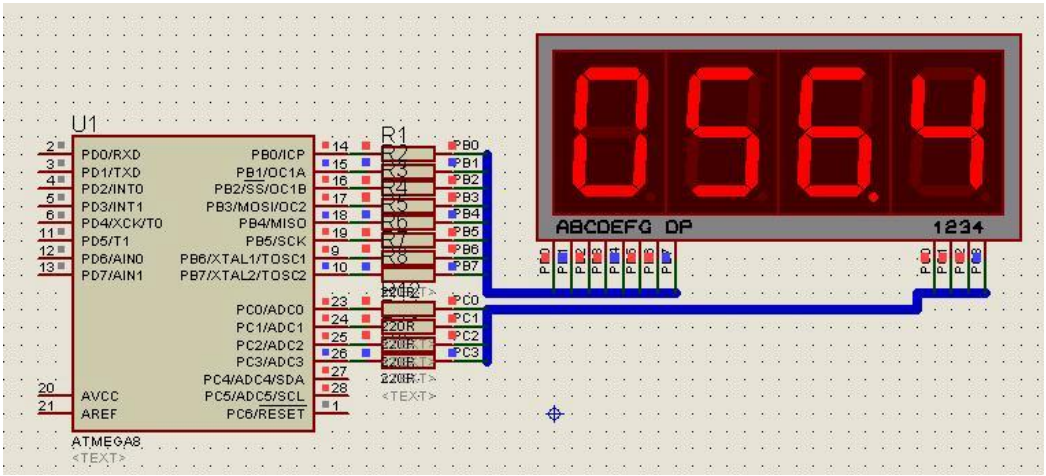
首先我来介绍一下多个数码管的显示：多位 LED 数码管显示电路按驱动方式分可分为静态显示和动态显示两种方法。

所谓静态显示就是除了要改变显示数据，所有的数码管都处于加电状态，上面的例子就是静态显示。它的优点是：显示稳定，亮度高，程序设计简单，MCU 负担小。缺点就是占用资源多，耗电量大，若用静态显示方式，一个 Atmega8 也就可以负载 3 个 LED 数码管，因此我们常考虑采用动态显示的方法。

动态显示就是采用各数码管循环轮流显示的方法，当循环显示频率教高时，利用人眼的暂留特性，感觉不到数码管的闪烁，就像看到数码管在同时发光一样，类似电影的原理。这种显示需要一个接口完成字形码的输出，另外一个接口完成各数码管的轮流显示。其优点是：占用资源少，耗电量小；缺点是：显示稳定性不易控制，程序设计复杂，MCU 负担重。

下面我们就试着用多个数码管做个计数器，用 PB 口做字形码的输出口，用 PC0~PC3 控制数码管的轮流显示。先是显示“000.0”四个数字，四秒钟后开始计数，由十分位开始，到十进位，最大为“999.9”最后变回初始状态。给你五分钟时间考虑一下程序怎么设计。

怎么样想好了吗，现在电路的设计已经不是问题了吧，我想你心里已经把它画好了吧。好我这里给出一个参考：



我写的主程序是这样的（延时函数还借用前面的），可以实现，但一定不是最好的，我想你一定能设计出比我写的更好的程序吧：

```
//主函数，依次显示数字0~9
void main()
{
    unsigned char i,j;
    static unsigned char LedNum[10]={0x3F, 0x06, 0x5B, 0x4F, 0x66,
                                     0x6D, 0x7D, 0x07, 0x7F, 0x6F};
    //显示小数点要加0x80
    unsigned char CountNum[4]={0, 0, 0, 0};     //计数的百、十、个位和十分位
    DDRB=0xFF;  //设置B口为输出模式
    PORTB=0xFF; //置高电平
    DDRC=0x0F;
    PORTC=0xFF;

    while (1)
    {
        i=0;
        for(j=0;j<100;j++) //依次显示4S"000.0"
        {
            i=++i%4;
            PORTC&=~(1<<i);
            if(2==i)
                PORTB=LedNum[0]+0x80;
            else
                PORTB=LedNum[0];
            delay_ms(10);
            PORTC|=0xFF;
        }
```

```
        CountNum[0]=CountNum[1]=CountNum[2]=CountNum[3]=0;
        while(1)
        {
            //计数加1
            if(CountNum[3]++==9)
            {
                CountNum[3]=0;
                if(9==CountNum[2]++)
                {
                    CountNum[2]=0;
                    if(9==CountNum[1]++)
                    {
                        CountNum[1]=0;
                        CountNum[0]++;
                    }
                }
            }

            //显示计数值
            for(i=0;i<4;i++)
            {
                PORTC&=~(1<<i);
                if(2==i)
                    PORTB=LedNum[CountNum[i]]+0x80;
                else
                    PORTB=LedNum[CountNum[i]];
                delay_ms(10);
                PORTC|=0xFF;
            }

            if(CountNum[0]==9 && CountNum[1]==9 && CountNum[2]==9
               && CountNum[3]==9)
                break;
        }
    }
}
```

很好，现在你已经对 Atmega8 的通用 I/O 口很熟悉了，而且对 Proteus 也很熟悉了吧，怎么样对 LED 数码管已经感觉熟烂于胸了吧。非常高兴看到你的进步！！

好了，到目前为止，我们对 LED 数码管的学习就可以告一段落了；但是我们还有一些新的事物还没看到，需要我们更加努力的去学习、实践在以后的工作中学习更多的知识。看看上面的程序你会发现 MCU 大多的时间都在忙于 LED 的显示，你把中间加一个延时就会发现问题了？如果我们还有别的事情要让 MCU 去做该怎么办呢，那样我们就要用到数码管控制器件了例如数码管驱动芯片 MAX7219。还有就是点阵 LED，如电梯里的方向指示灯，马路边的红绿灯等都是它的应用。不要着急，熟悉了上面的知识，别的你应用起来也会很快搞掂的。好了，放松一下，我们接着学习新的内容。