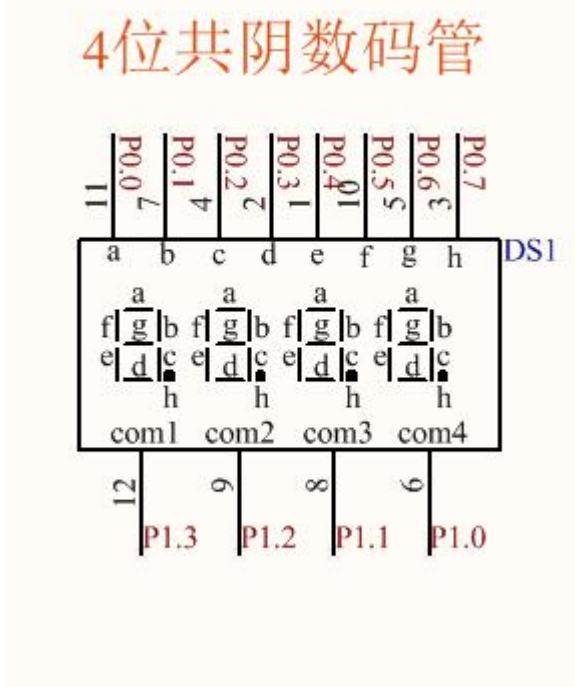


第一百一十四节： 动态扫描的数码管显示小数点。

【114.1 动态扫描的数码管显示小数点。】



上图 114.1.1 数码管

如上图，小数点的段码是 h，对应单片机的 P0.7 口。数码管编码转换表（类似字库）的 11 个以字节为单位的数据，把它们从十六进制转换成二进制后，可以发现第 7 位（对应 P0.7 口）都是 0。因此，从转换表里取数据后，得到的数据默认是让数码管的小数点不显示的。如果想显示这个小数点，就需要用到“或（|）”语句操作。比如，本节程序需要显示“1.234”这个带小数点的数值，代码如下：

```
#include "REG52.H"

#define SCAN_TIME 1

void TO_time();
void SystemInitial(void) ;
void Delay(unsigned long u32DelayTime) ;
void PeripheralInitial(void) ;

void DisplayScan(void);

sbit P1_0=P1^0;
sbit P1_1=P1^1;
sbit P1_2=P1^2;
sbit P1_3=P1^3;
```

//转换表，里面的 11 个数据，转换成二进制后，第 7 位数据都是 0 默认不显示小数点

```
code unsigned char Cu8DigTable[]=
```

```
{
0x3f, //0      序号 0
0x06, //1      序号 1
0x5b, //2      序号 2
0x4f, //3      序号 3
0x66, //4      序号 4
0x6d, //5      序号 5
0x7d, //6      序号 6
0x07, //7      序号 7
0x7f, //8      序号 8
0x6f, //9      序号 9
0x00, //不显示 序号 10
};
```

```
volatile unsigned char vGu8ScanTimerFlag=0;
```

```
volatile unsigned int vGu16ScanTimerCnt=0;
```

```
volatile unsigned char vGu8Display_Righ_4=1; //右起第 4 位数码管显示的变量。这里显示 “1”
volatile unsigned char vGu8Display_Righ_3=2; //右起第 3 位数码管显示的变量。这里显示 “2”
volatile unsigned char vGu8Display_Righ_2=3; //右起第 2 位数码管显示的变量。这里显示 “3”
volatile unsigned char vGu8Display_Righ_1=4; //右起第 1 位数码管显示的变量。这里显示 “4”
```

/\* 注释一：

```
* vGu8Display_Righ_Dot_4, vGu8Display_Righ_Dot_3, vGu8Display_Righ_Dot_2,
* vGu8Display_Righ_Dot_1, 这四个全局变量用来传递每位数码管是否需要显示它的小数点，如果是 1
* 代表需要显示其小数点，如果是 0 则不显示小数点。这四个变量作为对上面应用层调用的接口变量。
*/
```

```
volatile unsigned char vGu8Display_Righ_Dot_4=1; //右起第 4 位数码管的小数点。1 代表打开显示。
volatile unsigned char vGu8Display_Righ_Dot_3=0; //右起第 3 位数码管的小数点。0 代表关闭显示。
volatile unsigned char vGu8Display_Righ_Dot_2=0; //右起第 2 位数码管的小数点。0 代表关闭显示。
volatile unsigned char vGu8Display_Righ_Dot_1=0; //右起第 1 位数码管的小数点。0 代表关闭显示。
```

```
void main()
```

```
{
    SystemInitial();
    Delay(10000);
    PeripheralInitial();
    while(1)
    {
    }
}
```

```
}
```

```
void DisplayScan(void)
```

```
{
```

```
    static unsigned char Su8GetCode;
```

```
    static unsigned char Su8ScanStep=1;
```

```
    if(0==vGu16ScanTimerCnt)
```

```
    {
```

```
        P0=0x00;
```

```
        P1_0=1;
```

```
        P1_1=1;
```

```
        P1_2=1;
```

```
        P1_3=1;
```

```
        switch(Su8ScanStep)
```

```
        {
```

```
            case 1:
```

```
                Su8GetCode=Cu8DigTable[vGu8Display_Righ_1];
```

```
/* 注释二：
```

```
* 这里是本节的关键。通过判断全局的接口变量的数值，来决定是否打开显示小数点。
```

```
* 从转换表取出字模数据后再跟 0x80 进行“或”运算即可把第 7 位数据改为 1。
```

```
*/
```

```
        if(1==vGu8Display_Righ_Dot_1) //如果打开了需要显示第 1 个数码管的小数点
```

```
        {
```

```
            Su8GetCode=Su8GetCode|0x80; //把第 7 位数据改为 1，显示小数点
```

```
        }
```

```
        P0=Su8GetCode;
```

```
        P1_0=0;
```

```
        P1_1=1;
```

```
        P1_2=1;
```

```
        P1_3=1;
```

```
        break;
```

```
            case 2:
```

```
                Su8GetCode=Cu8DigTable[vGu8Display_Righ_2];
```

```
                if(1==vGu8Display_Righ_Dot_2) //如果打开了需要显示第 2 个数码管的小数点
```

```
                {
```

```
                    Su8GetCode=Su8GetCode|0x80; //把第 7 位数据改为 1，显示小数点
```

```

    }
    P0=Su8GetCode;
    P1_0=1;
    P1_1=0;
    P1_2=1;
    P1_3=1;
    break;

case 3:
    Su8GetCode=Cu8DigTable[vGu8Display_Righ_3];
    if(1==vGu8Display_Righ_Dot_3) //如果打开了需要显示第 3 个数码管的小数点
    {
        Su8GetCode=Su8GetCode|0x80; //把第 7 位数据改为 1，显示小数点
    }
    P0=Su8GetCode;
    P1_0=1;
    P1_1=1;
    P1_2=0;
    P1_3=1;
    break;

case 4:
    Su8GetCode=Cu8DigTable[vGu8Display_Righ_4];
    if(1==vGu8Display_Righ_Dot_4) //如果打开了需要显示第 4 个数码管的小数点
    {
        Su8GetCode=Su8GetCode|0x80; //把第 7 位数据改为 1，显示小数点
    }
    P0=Su8GetCode;
    P1_0=1;
    P1_1=1;
    P1_2=1;
    P1_3=0;
    break;

}

Su8ScanStep++;
if (Su8ScanStep>4)
{
    Su8ScanStep=1;
}

vGu8ScanTimerFlag=0;
vGu16ScanTimerCnt=SCAN_TIME;

```

```

        vGu8ScanTimerFlag=1;
    }
}

void T0_time() interrupt 1
{
    DisplayScan();

    if(1==vGu8ScanTimerFlag&&vGu16ScanTimerCnt>0)
    {
        vGu16ScanTimerCnt--;
    }

    TH0=0xfc;
    TL0=0x66;
}

```

```

void SystemInitial(void)
{
    P0=0x00;
    P1_0=1;
    P1_1=1;
    P1_2=1;
    P1_3=1;

    TMOD=0x01;
    TH0=0xfc;
    TL0=0x66;
    EA=1;
    ET0=1;
    TR0=1;
}

```

```

void Delay(unsigned long u32DelayTime)
{
    for(;u32DelayTime>0;u32DelayTime--);
}

```

```

void PeripheralInitial(void)
{
}

```

