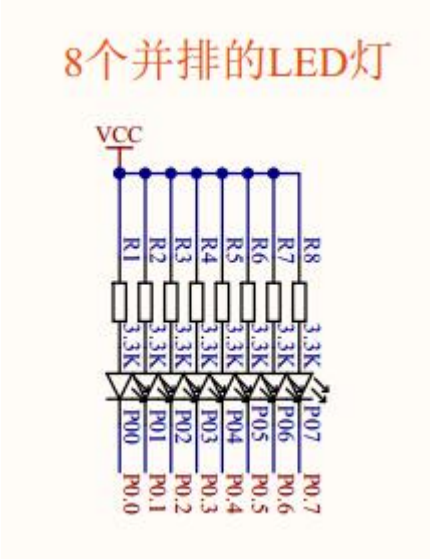
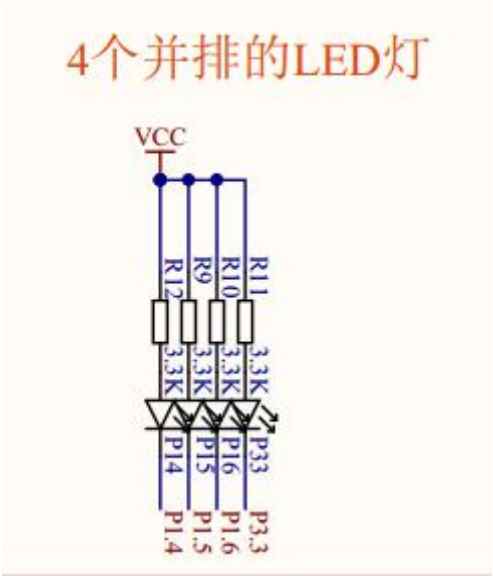
第九十节： 多任务并行处理两路跑马灯。

【90.1　多任务并行处理。】

两路速度不同的跑马灯，代表了两路独立运行的任务，单片机如何“并行”处理这两路任务，就涉及到“多任务并行处理的编程思路”。



上图 90.1.1　灌入式驱动 8 个 LED　第 1 路跑马灯



上图 90.1.2　灌入式驱动 4 个 LED　新增加的第 2 路跑马灯

如上图，本节特别值得一提的是，新增加的第 2 路跑马灯用的是 4 个 LED，这 4 个 LED 的驱动 IO 口是“散装的”，因为，前面 3 个是 P1 口的（P1.4,P1.5,P1.6），最后 1 个是 P3 口的（P3.3），这种情况下，肯定用不了“移位”的处理思路，只能用跑马灯第 3 种境界里所介绍的“状态切换非阻塞”思路，可见，“IO 口拆分”和“switch 状态切换”又一次充分体现了它们“程序框架万能扩展”的优越性。代码如下：

```
#include "REG52.H"

void T0_time();
void SystemInitial(void) ;
void Delay(unsigned long u32DelayTime) ;
void PeripheralInitial(void) ;
void Led_1_Task(void);
void Led_2_Task(void);

#define  BLINK_TIME_1   1000  //控制第 1 路跑马灯的速度，数值越大“跑动”越慢。
#define  BLINK_TIME_2   200   //控制第 2 路跑马灯的速度，数值越大“跑动”越慢。

sbit P0_0=P0^0;
sbit P0_1=P0^1;
sbit P0_2=P0^2;
sbit P0_3=P0^3;
sbit P0_4=P0^4;
sbit P0_5=P0^5;
sbit P0_6=P0^6;
sbit P0_7=P0^7;

sbit P1_4=P1^4;
sbit P1_5=P1^5;
sbit P1_6=P1^6;
sbit P3_3=P3^3;

volatile unsigned char vGu8TimeFlag_1=0;
volatile unsigned int vGu16TimeCnt_1=0;

volatile unsigned char vGu8TimeFlag_2=0;
volatile unsigned int vGu16TimeCnt_2=0;

void main()
{
    SystemInitial();
    Delay(10000);
    PeripheralInitial();
    while(1)
    {
        Led_1_Task();   //第 1 路跑马灯
        Led_2_Task();   //第 2 路跑马灯
    }
}
```

```
void T0_time() interrupt 1
{
    if(1==vGu8TimeFlag_1&&vGu16TimeCnt_1>0) //软件定时器 1
    {
            vGu16TimeCnt_1--;
    }

    if(1==vGu8TimeFlag_2&&vGu16TimeCnt_2>0) //软件定时器 2
    {
            vGu16TimeCnt_2--;
    }


    TH0=0xfc;
    TL0=0x66;
}



void SystemInitial(void)
{
    TMOD=0x01;
    TH0=0xfc;
    TL0=0x66;
    EA=1;
    ET0=1;
    TR0=1;
}

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

void PeripheralInitial(void)
{

}

//第 1 路跑马灯
void Led_1_Task(void)
{
      static unsigned char Su8Step=0;    //加 static 修饰的局部变量，每次进来都会保留上一次值。
```

```
    switch(Su8Step)
    {
        case 0:
            if(0==vGu16TimeCnt_1)  //时间到
            {

                vGu8TimeFlag_1=0;
                vGu16TimeCnt_1=BLINK_TIME_1;   //重装定时的时间
                vGu8TimeFlag_1=1;

                P0_0=1;   //第 0 个灯熄灭
                P0_1=0;
                P0_2=0;
                P0_3=0;
                P0_4=0;
                P0_5=0;
                P0_6=0;
                P0_7=0;

                Su8Step=1;  //切换到下一个步骤，精髓语句!
             }
            break;

        case 1:
            if(0==vGu16TimeCnt_1)  //时间到
            {

                vGu8TimeFlag_1=0;
                vGu16TimeCnt_1=BLINK_TIME_1;   //重装定时的时间
                vGu8TimeFlag_1=1;

                P0_0=0;
                P0_1=1;   //第 1 个灯熄灭
                P0_2=0;
                P0_3=0;
                P0_4=0;
                P0_5=0;
                P0_6=0;
                P0_7=0;

                Su8Step=2;  //切换到下一个步骤，精髓语句!
             }
            break;
```

```
        case 2:
            if(0==vGu16TimeCnt_1)  //时间到
            {

                vGu8TimeFlag_1=0;
                vGu16TimeCnt_1=BLINK_TIME_1;   //重装定时的时间
                vGu8TimeFlag_1=1;

                P0_0=0;
                P0_1=0;
                P0_2=1;   //第 2 个灯熄灭
                P0_3=0;
                P0_4=0;
                P0_5=0;
                P0_6=0;
                P0_7=0;

                Su8Step=3;  //切换到下一个步骤，精髓语句!
             }
            break;

        case 3:
            if(0==vGu16TimeCnt_1)  //时间到
            {

                vGu8TimeFlag_1=0;
                vGu16TimeCnt_1=BLINK_TIME_1;   //重装定时的时间
                vGu8TimeFlag_1=1;

                P0_0=0;
                P0_1=0;
                P0_2=0;
                P0_3=1;   //第 3 个灯熄灭
                P0_4=0;
                P0_5=0;
                P0_6=0;
                P0_7=0;

                Su8Step=4;  //切换到下一个步骤，精髓语句!
             }
            break;

        case 4:
            if(0==vGu16TimeCnt_1)  //时间到
```

```
        {

            vGu8TimeFlag_1=0;
            vGu16TimeCnt_1=BLINK_TIME_1;   //重装定时的时间
            vGu8TimeFlag_1=1;

            P0_0=0;
            P0_1=0;
            P0_2=0;
            P0_3=0;
            P0_4=1;   //第 4 个灯熄灭
            P0_5=0;
            P0_6=0;
            P0_7=0;

            Su8Step=5;  //切换到下一个步骤，精髓语句!
         }
        break;

    case 5:
        if(0==vGu16TimeCnt_1)  //时间到
        {

            vGu8TimeFlag_1=0;
            vGu16TimeCnt_1=BLINK_TIME_1;   //重装定时的时间
            vGu8TimeFlag_1=1;

            P0_0=0;
            P0_1=0;
            P0_2=0;
            P0_3=0;
            P0_4=0;
            P0_5=1;   //第 5 个灯熄灭
            P0_6=0;
            P0_7=0;

            Su8Step=6;  //切换到下一个步骤，精髓语句!
         }
        break;

    case 6:
        if(0==vGu16TimeCnt_1)  //时间到
        {
```

```
                    vGu8TimeFlag_1=0;
                    vGu16TimeCnt_1=BLINK_TIME_1;   //重装定时的时间
                    vGu8TimeFlag_1=1;

                    P0_0=0;
                    P0_1=0;
                    P0_2=0;
                    P0_3=0;
                    P0_4=0;
                    P0_5=0;
                    P0_6=1;   //第 6 个灯熄灭
                    P0_7=0;

                    Su8Step=7;  //切换到下一个步骤，精髓语句!
                }
                break;

            case 7:

               if(0==vGu16TimeCnt_1)     //时间到
                {
                    vGu8TimeFlag_1=0;
                    vGu16TimeCnt_1=BLINK_TIME_1;  //重装定时的时间
                    vGu8TimeFlag_1=1;

                    P0_0=0;
                    P0_1=0;
                    P0_2=0;
                    P0_3=0;
                    P0_4=0;
                    P0_5=0;
                    P0_6=0;
                    P0_7=1;   //第 7 个灯熄灭

                    Su8Step=0;   //返回到第 0 个步骤重新开始往下走，精髓语句!
                }
                break;
        }
}

//第 2 路跑马灯
void Led_2_Task(void)
{
/*
```

疑点讲解(1)：

这里第 2 路跑马灯的“Su8Step”与第 1 路跑马灯的“Su8Step”虽然同名，但是，因为它们是静态的局部变量，在两个不同的函数内部，是两个不同的变量，这两个变量所分配的 RAM 内存地址是不一样的，因此，它们虽然同名，但是不矛盾不冲突。

*/

```
    static unsigned char Su8Step=0;    //加 static 修饰的局部变量，每次进来都会保留上一次值。

     switch(Su8Step)
     {
         case 0:
             if(0==vGu16TimeCnt_2)  //时间到
             {

                 vGu8TimeFlag_2=0;
                 vGu16TimeCnt_2=BLINK_TIME_2;   //重装定时的时间
                 vGu8TimeFlag_2=1;

                 P1_4=1; //第 0 个灯熄灭
                 P1_5=0;
                 P1_6=0;
                 P3_3=0;

                 Su8Step=1;  //切换到下一个步骤，精髓语句!
              }
             break;

         case 1:
             if(0==vGu16TimeCnt_2)  //时间到
             {

                 vGu8TimeFlag_2=0;
                 vGu16TimeCnt_2=BLINK_TIME_2;   //重装定时的时间
                 vGu8TimeFlag_2=1;

                 P1_4=0;
                 P1_5=1;   //第 1 个灯熄灭
                 P1_6=0;
                 P3_3=0;

                 Su8Step=2;  //切换到下一个步骤，精髓语句!
              }
             break;

         case 2:
```

```
            if(0==vGu16TimeCnt_2)  //时间到
            {

                vGu8TimeFlag_2=0;
                vGu16TimeCnt_2=BLINK_TIME_2;   //重装定时的时间
                vGu8TimeFlag_2=1;

                P1_4=0;
                P1_5=0;
                P1_6=1;   //第 2 个灯熄灭
                P3_3=0;

                Su8Step=3;  //切换到下一个步骤，精髓语句!
             }
            break;

        case 3:
            if(0==vGu16TimeCnt_2)  //时间到
            {

                vGu8TimeFlag_2=0;
                vGu16TimeCnt_2=BLINK_TIME_2;   //重装定时的时间
                vGu8TimeFlag_2=1;

                P1_4=0;
                P1_5=0;
                P1_6=0;
                P3_3=1;   //第 3 个灯熄灭

                Su8Step=0;   //返回到第 0 个步骤重新开始往下走，精髓语句!
             }
            break;

    }
}
```