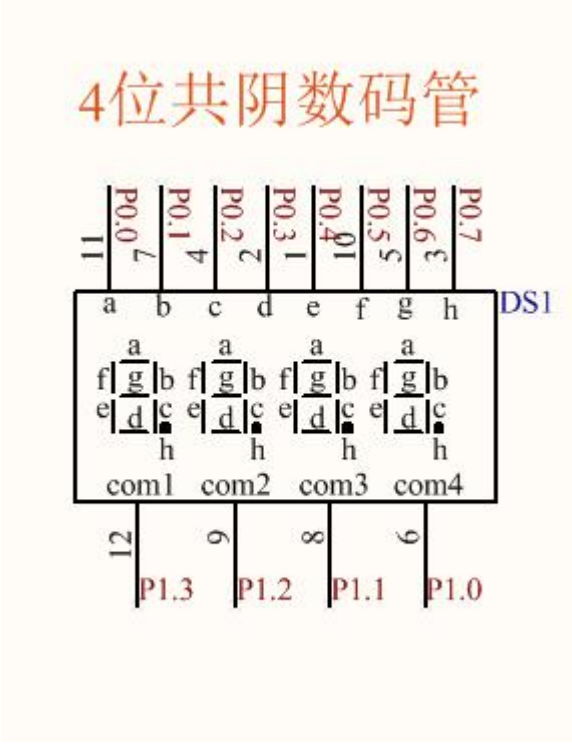
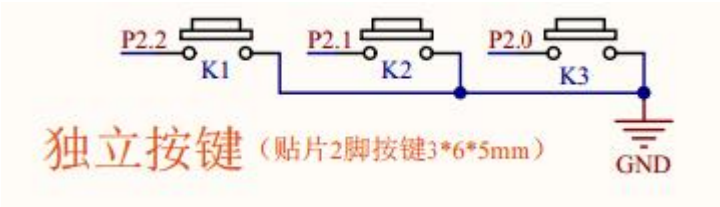
第一百二十四节： 数显仪表盘显示“速度、方向、计数器”的跑马灯。

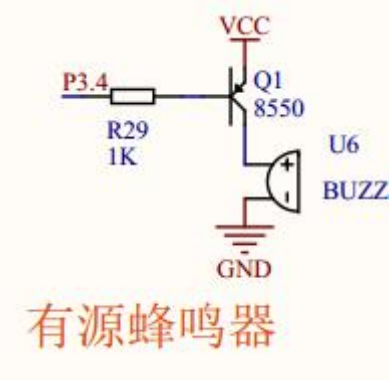
【124.1　数显仪表盘显示“速度、方向、计数器”的跑马灯。】



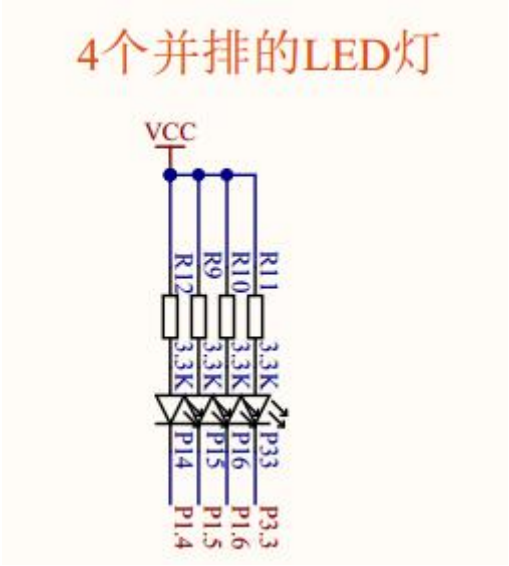
上图 124.1.1　数码管



上图 124.1.2　独立按键



上图 124.1.3　有源蜂鸣器



上图 124.1.4　LED 电路

本节小项目，意在“人机界面”与“过程控制”如何关联的练习。

程序功能如下：

（1）数码管显示的格式是“S.D.CC”。其中 S 是代表 3 档速度，能显示的数字范围是“1、2、3”，分别代表“慢、中、快”3 档速度。D 代表方向，往右跑显示符号“r”（right 的首字母），往左跑显示符号“L”（Left 的首字母）。CC 代表计数器，跑马灯每跑完一次，计数器自动加 1，范围是 0 到 99。

（2）【速度】按键 K1。每按一次【速度】按键 K1，速度档位显示的数字在“1、2、3”之间切换。

（3）【方向】按键 K2。跑马灯上电后默认处于“往右跑”的方向,默认显示字符“r”。每按一次【方向】按键 K2，跑马灯就在“往右跑”与“往左跑”两个方向之间切换，显示的字符在“r、L”之间切换。

（4）【启动暂停】按键 K3。上电后，按下【启动暂停】按键 K3 启动之后，跑马灯处于“启动”状态，4 个 LED 灯挨个依次循环的变亮，给人“跑”起来的感觉，此时再按一次【启动暂停】按键 K3，则跑马灯处于“暂停”状态，接着又按一次【启动暂停】按键 K3，跑马灯又变回“启动”状态。因此，【启动暂停】按键 K3 是专门用来切换“启动”和“暂停”这两种状态。

代码如下：

```
#include "REG52.H"

#define KEY_FILTER_TIME  25

#define SCAN_TIME  1
#define VOICE_TIME   50

#define RUN_TIME_SLOW    500   //“慢”档速度的时间参数
#define RUN_TIME_MIDDLE  300   //“中”档速度的时间参数
#define RUN_TIME_FAST    100   //“快”档速度的时间参数

void T0_time();
void SystemInitial(void) ;
```

```
void Delay(unsigned long u32DelayTime) ;
void PeripheralInitial(void) ;

void KeyScan(void);
void KeyTask(void);
void RunTask(void);   //跑马灯的任务函数

void VoiceScan(void);
void DisplayScan(void);
void DisplayTask(void);
void Wd1(void);   //窗口 1。
void BeepOpen(void);
void BeepClose(void);

sbit KEY_INPUT1=P2^2;
sbit KEY_INPUT2=P2^1;
sbit KEY_INPUT3=P2^0;

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

sbit P3_4=P3^4;

//4 个跑马灯的输出口
sbit P1_4=P1^4;
sbit P1_5=P1^5;
sbit P1_6=P1^6;
sbit P3_3=P3^3;


//数码管转换表
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
0x40,  //横杠-   序号 11
0x38,  //字符 L   序号 12
0x70,  //字符 r   序号 13
};

volatile unsigned char vGu8ScanTimerFlag=0;
volatile unsigned int vGu16ScanTimerCnt=0;

volatile unsigned char vGu8BeepTimerFlag=0;
volatile unsigned int vGu16BeepTimerCnt=0;

unsigned char Gu8Wd=0;   //窗口选择变量。人机交互程序框架的支点。
unsigned char Gu8WdUpdate=0;  //整屏更新变量。

unsigned char Gu8PartUpdate_1=0;   //局部 1 的更新变量,
unsigned char Gu8PartUpdate_2=0;   //局部 2 的更新变量
unsigned char Gu8PartUpdate_3=0;   //局部 3 的更新变量,

volatile unsigned char vGu8Display_Righ_4=0;
volatile unsigned char vGu8Display_Righ_3=0;
volatile unsigned char vGu8Display_Righ_2=0;
volatile unsigned char vGu8Display_Righ_1=0;

volatile unsigned char vGu8Display_Righ_Dot_4=1;  //需要显示的小数点
volatile unsigned char vGu8Display_Righ_Dot_3=1;  //需要显示的小数点
volatile unsigned char vGu8Display_Righ_Dot_2=0;
volatile unsigned char vGu8Display_Righ_Dot_1=0;

volatile unsigned char vGu8KeySec=0;

unsigned char Gu8RunCounter=0;    //计数器，范围是 0 到 99

unsigned char Gu8RunStep=0; //运行的步骤
unsigned char Gu8RunStart=0;      //控制跑马灯启动的总开关

unsigned char Gu8RunStatus=0;     //标识跑马灯当前的状态。0 代表停止，1 代表启动，2 代表暂停。
unsigned char Gu8RunDirection=0;  //标识跑马灯当前的方向。0 代表往右跑，1 代表往左跑。
unsigned char Gu8RunSpeed=1;      //当前的速度档位。1 代表“慢”，2 代表“中”，3 代表“快”。
unsigned int  Gu16RunSpeedTimeDate=0; //承接各速度档位的时间参数的变量

volatile unsigned char vGu8RunTimerFlag=0;   //用于控制跑马灯跑动速度的定时器
volatile unsigned int vGu16RunTimerCnt=0;
```

```
void main()
{
    SystemInitial();
    Delay(10000);
    PeripheralInitial();
    while(1)
    {
        KeyTask();      //按键的任务函数
        DisplayTask();  //数码管显示的上层任务函数
        RunTask();    //跑马灯的任务函数
    }
}

void RunTask(void)    //跑马灯的任务函数，放在主函数内
{
    if(0==Gu8RunStart) //如果是停止的状态
    {
          return;  //如果是停止的状态，退出当前函数，不扫描余下代码。
    }

    switch(Gu8RunStep) //屡见屡爱的 switch 又来了
    {
       case 0:
            vGu8RunTimerFlag=0;
            vGu16RunTimerCnt=0;  //定时器清零
            Gu8RunStep=1;  //切换到下一步，启动

           break;
       case 1:
           if(1==Gu8RunStatus&&0==vGu16RunTimerCnt) //当前处于“启动”状态，并且定时器等于 0
            {
                P1_4=0;   //第 1 个灯亮
                P1_5=1;   //第 2 个灯灭
                P1_6=1;   //第 3 个灯灭
                P3_3=1;   //第 4 个灯灭

                vGu8RunTimerFlag=0;
                vGu16RunTimerCnt=Gu16RunSpeedTimeDate; //速度时间参数变量的大小，决定了速度
                vGu8RunTimerFlag=1;   //启动定时器

                //灵活切换“步骤变量”
                if(0==Gu8RunDirection) //往右跑
                {
```

```
                Gu8RunStep=2;
            }
            else  //往左跑
            {
                if(Gu8RunCounter<99)
                {
                    Gu8RunCounter++; //往左边跑完一次，运行的计数器自加 1
                }
                Gu8PartUpdate_3=1;   //局部 3 的更新变量，更新显示计数器

                Gu8RunStep=4;
            }

        }

        break;
    case 2:
        if(1==Gu8RunStatus&&0==vGu16RunTimerCnt) //当前处于“启动”状态，并且定时器等于 0
        {
            P1_4=1;   //第 1 个灯灭
            P1_5=0;   //第 2 个灯亮
            P1_6=1;   //第 3 个灯灭
            P3_3=1;   //第 4 个灯灭

            vGu8RunTimerFlag=0;
            vGu16RunTimerCnt=Gu16RunSpeedTimeDate; //速度时间参数变量的大小，决定了速度
            vGu8RunTimerFlag=1;   //启动定时器

            //灵活切换“步骤变量”
            if(0==Gu8RunDirection) //往右跑
            {
                Gu8RunStep=3;
            }
            else  //往左跑
            {
                Gu8RunStep=1;
            }
        }

        break;
    case 3:
        if(1==Gu8RunStatus&&0==vGu16RunTimerCnt) //当前处于“启动”状态，并且定时器等于 0
        {
            P1_4=1;   //第 1 个灯灭
```

```
            P1_5=1;   //第 2 个灯灭
            P1_6=0;   //第 3 个灯亮
            P3_3=1;   //第 4 个灯灭

            vGu8RunTimerFlag=0;
            vGu16RunTimerCnt=Gu16RunSpeedTimeDate; //速度时间参数变量的大小，决定了速度
            vGu8RunTimerFlag=1;   //启动定时器

            //灵活切换“步骤变量”
            if(0==Gu8RunDirection) //往右跑
            {
               Gu8RunStep=4;
            }
            else  //往左跑
            {
               Gu8RunStep=2;
            }
         }

        break;
    case 4:
        if(1==Gu8RunStatus&&0==vGu16RunTimerCnt) //当前处于“启动”状态，并且定时器等于 0
         {
            P1_4=1;   //第 1 个灯灭
            P1_5=1;   //第 2 个灯灭
            P1_6=1;   //第 3 个灯灭
            P3_3=0;   //第 4 个灯亮

            vGu8RunTimerFlag=0;
            vGu16RunTimerCnt=Gu16RunSpeedTimeDate; //速度时间参数变量的大小，决定了速度
            vGu8RunTimerFlag=1;   //启动定时器

            //灵活切换“步骤变量”
            if(0==Gu8RunDirection) //往右跑
            {
                if(Gu8RunCounter<99)
                {
                    Gu8RunCounter++; //往右边跑完一次，运行的计数器自加 1
                }
                Gu8PartUpdate_3=1;   //局部 3 的更新变量，更新显示计数器

               Gu8RunStep=1;
            }
            else  //往左跑
```

```
                {
                    Gu8RunStep=3;
                }
            }

            break;

    }

}

void KeyTask(void)    //按键的任务函数
{
    if(0==vGu8KeySec)
    {
        return;
    }

    switch(vGu8KeySec)
    {
       case 1:     //【速度】按键 K1
            switch(Gu8Wd) //在某个窗口下
            {
                case 1:     //窗口 1。
                      //每按一次 K1 按键，Gu8RunSpeed 就在 1、2、3 三者之间切换,
                      //并且根据 Gu8RunSpeed 的数值，对 Gu16RunSpeedTimeDate 赋值
                      //不同的速度时间参数，从而控制速度档位。

                      if(1==Gu8RunSpeed)
                      {
                          Gu8RunSpeed=2;  //“中”档
                          Gu16RunSpeedTimeDate=RUN_TIME_MIDDLE; //赋值“中”档的时间参数
                      }
                      else if(2==Gu8RunSpeed)
                      {
                          Gu8RunSpeed=3;   //“快”档
                          Gu16RunSpeedTimeDate=RUN_TIME_FAST; //赋值“快”档的时间参数
                      }
                      else
                      {
                          Gu8RunSpeed=1;   //“慢”档
                          Gu16RunSpeedTimeDate=RUN_TIME_SLOW; //赋值“慢”档的时间参数
                      }
```

```
                Gu8PartUpdate_1=1;   //局部 1 的更新变量，更新显示“速度”

                vGu8BeepTimerFlag=0;
                vGu16BeepTimerCnt=VOICE_TIME;  //蜂鸣器发出“滴”一声
                vGu8BeepTimerFlag=1;
                break;
        }

        vGu8KeySec=0;
        break;

    case 2:     //【方向】按键 K2

        switch(Gu8Wd) //在某个窗口下
        {
            case 1:     //窗口 1。
                //每按一次 K2 按键，Gu8RunDirection 就在 0 和 1 之间切换,从而控制方向
                if(0==Gu8RunDirection)
                {
                    Gu8RunDirection=1;
                }
                else
                {
                    Gu8RunDirection=0;
                }

                Gu8PartUpdate_2=1;   //局部 2 更新显示，更新显示“方向”

                vGu8BeepTimerFlag=0;
                vGu16BeepTimerCnt=VOICE_TIME;  //蜂鸣器发出“滴”一声
                vGu8BeepTimerFlag=1;
                break;
        }
        vGu8KeySec=0;
        break;

    case 3:     //【启动暂停】按键 K3
        switch(Gu8Wd) //在某个窗口下
        {
            case 1:     //窗口 1。
                if(0==Gu8RunStatus) //当跑马灯处于“停止”状态时
                {
                    Gu8RunStep=0;    //运行步骤从 0 开始
                    Gu8RunStart=1;   //总开关“打开”。
```

```
                        Gu8RunStatus=1;  //状态切换到“启动”状态
                    }
                    else if(1==Gu8RunStatus) //当跑马灯处于“启动”状态时
                    {
                        Gu8RunStatus=2;  //状态切换到“暂停”状态
                    }
                    else  //当跑马灯处于“暂停”状态时
                    {
                        Gu8RunStatus=1;  //状态切换到“启动”状态
                    }

                    vGu8BeepTimerFlag=0;
                    vGu16BeepTimerCnt=VOICE_TIME;  //蜂鸣器发出“滴”一声
                    vGu8BeepTimerFlag=1;
                    break;
            }

            vGu8KeySec=0;
            break;
    }
}

void DisplayTask(void) //数码管显示的上层任务函数
{
    switch(Gu8Wd)  //以窗口选择 Gu8Wd 为支点，去执行对应的窗口显示函数。又一次用到 switch 语句
    {
        case 1:
            Wd1();   //窗口 1。
            break;
    }
}

void Wd1(void)   //窗口 1。
{
    //需要借用的中间变量，用来拆分数据位。
    static unsigned char Su8Temp_4,Su8Temp_3,Su8Temp_2,Su8Temp_1; //需要借用的中间变量

    if(1==Gu8WdUpdate) //如果需要整屏更新
    {
          Gu8WdUpdate=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

          //属于静态数据，起“装饰”作用，切换窗口后只扫描一次的代码。
          vGu8Display_Righ_Dot_4=1;  //显示小数点
          vGu8Display_Righ_Dot_3=1;  //显示小数点
```

```
        vGu8Display_Righ_Dot_2=0;
        vGu8Display_Righ_Dot_1=0;

        Gu8PartUpdate_1=1;  //局部 1 更新显示
        Gu8PartUpdate_2=1;  //局部 2 更新显示
        Gu8PartUpdate_3=1;  //局部 3 更新显示
    }

    if(1==Gu8PartUpdate_1) //局部 1 更新显示，速度
    {
        Gu8PartUpdate_1=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

        Su8Temp_4=Gu8RunSpeed;

        vGu8Display_Righ_4=Su8Temp_4;  //过渡需要显示的数据到底层驱动变量
    }

    if(1==Gu8PartUpdate_2) //局部 2 更新显示，方向
    {
        Gu8PartUpdate_2=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

        if(0==Gu8RunDirection) //往右跑
        {
            Su8Temp_3=13; //数码管的字模转换表序号 13 代表显示字符“r”
        }
        else
        {
            Su8Temp_3=12; //数码管的字模转换表序号 12 代表显示字符“L”
        }
        vGu8Display_Righ_3=Su8Temp_3;  //过渡需要显示的数据到底层驱动变量
    }

    if(1==Gu8PartUpdate_3) //局部 3 更新显示，计数器
    {
        Gu8PartUpdate_3=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

        Su8Temp_2=Gu8RunCounter%100/10; //提取十位
        Su8Temp_1=Gu8RunCounter%10/1;   //提取个位

        vGu8Display_Righ_2=Su8Temp_2;  //过渡需要显示的数据到底层驱动变量
        vGu8Display_Righ_1=Su8Temp_1;  //过渡需要显示的数据到底层驱动变量
    }

}
```

```
void KeyScan(void)  //按键底层的驱动扫描函数，放在定时中断函数里
{
   static unsigned char Su8KeyLock1;
   static unsigned int  Su16KeyCnt1;
   static unsigned char Su8KeyLock2;
   static unsigned int  Su16KeyCnt2;
   static unsigned char Su8KeyLock3;
   static unsigned int  Su16KeyCnt3;


   if(0!=KEY_INPUT1)
   {
      Su8KeyLock1=0;
      Su16KeyCnt1=0;
   }
   else if(0==Su8KeyLock1)
   {
      Su16KeyCnt1++;
      if(Su16KeyCnt1>=KEY_FILTER_TIME)
      {
            Su8KeyLock1=1;
            vGu8KeySec=1;
      }
   }

   if(0!=KEY_INPUT2)
   {
      Su8KeyLock2=0;
      Su16KeyCnt2=0;
   }
   else if(0==Su8KeyLock2)
   {
      Su16KeyCnt2++;
      if(Su16KeyCnt2>=KEY_FILTER_TIME)
      {
         Su8KeyLock2=1;
         vGu8KeySec=2;
      }
   }

   if(0!=KEY_INPUT3)
   {
```

```
      Su8KeyLock3=0;
      Su16KeyCnt3=0;
   }
   else if(0==Su8KeyLock3)
   {
      Su16KeyCnt3++;
      if(Su16KeyCnt3>=KEY_FILTER_TIME)
      {
         Su8KeyLock3=1;
         vGu8KeySec=3;
      }
   }

}

void DisplayScan(void)    //数码管底层的驱动扫描函数，放在定时中断函数里
{
    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];

                if(1==vGu8Display_Righ_Dot_1)
                {
                    Su8GetCode=Su8GetCode|0x80;
                }
                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)
            {
                Su8GetCode=Su8GetCode|0x80;
            }
            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)
            {
                Su8GetCode=Su8GetCode|0x80;
            }
            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)
            {
                Su8GetCode=Su8GetCode|0x80;
            }
            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 VoiceScan(void) //蜂鸣器的驱动函数
{

      static unsigned char Su8Lock=0;

      if(1==vGu8BeepTimerFlag&&vGu16BeepTimerCnt>0)
      {
          if(0==Su8Lock)
          {
               Su8Lock=1;
               BeepOpen();
          }
          else
          {

               vGu16BeepTimerCnt--;

               if(0==vGu16BeepTimerCnt)
               {
                    Su8Lock=0;
                    BeepClose();
               }

          }
      }
}

void BeepOpen(void)
{
    P3_4=0;
}

void BeepClose(void)
```

```
{
    P3_4=1;
}

void T0_time() interrupt 1
{
    VoiceScan();    //蜂鸣器的驱动函数
    KeyScan();      //按键底层的驱动扫描函数
    DisplayScan();  //数码管底层的驱动扫描函数

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

    if(1==vGu8RunTimerFlag&&vGu16RunTimerCnt>0)  //用于控制跑马灯跑动速度的定时器
    {
        vGu16RunTimerCnt--;
    }

    TH0=0xfd;   //此参数可根据具体的时间来修改，尽量确保每定时中断一次接近 1ms
    TL0=0x40;   //此参数可根据具体的时间来修改，尽量确保每定时中断一次接近 1ms
}

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

    TMOD=0x01;
    TH0=0xfd;   //此参数可根据具体的时间来修改，尽量确保每定时中断一次接近 1ms
    TL0=0x40;   //此参数可根据具体的时间来修改，尽量确保每定时中断一次接近 1ms
    EA=1;
    ET0=1;
    TR0=1;

    //上电初始化一些关键的数据

    Gu8Wd=1;   //窗口 1。开机默认处于正常工作的窗口
    Gu8WdUpdate=1;  //整屏更新变量
    //跑马灯处于初始化的状态
```

```
    P1_4=0;   //第 1 个灯亮
    P1_5=1;   //第 2 个灯灭
    P1_6=1;   //第 3 个灯灭
    P3_3=1;   //第 4 个灯灭

    //根据当前的速度档位 Gu8RunSpeed，来初始化速度时间参数 Gu16RunSpeedTimeDate
    if(1==Gu8RunSpeed)
    {
        Gu16RunSpeedTimeDate=RUN_TIME_SLOW; //赋值“慢”档的时间参数
    }
    else if(2==Gu8RunSpeed)
    {
        Gu16RunSpeedTimeDate=RUN_TIME_MIDDLE; //赋值“中”档的时间参数
    }
    else
    {
        Gu16RunSpeedTimeDate=RUN_TIME_FAST; //赋值“快”档的时间参数
    }

}

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

void PeripheralInitial(void)
{

}
```