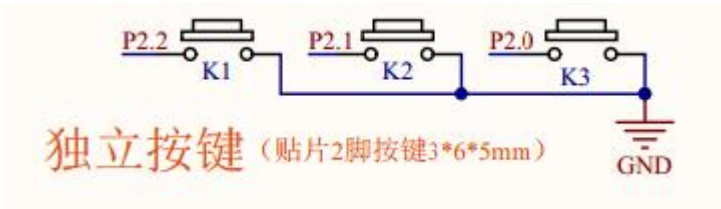


第一百零八节： 按键控制跑马灯的启动和暂停和停止。

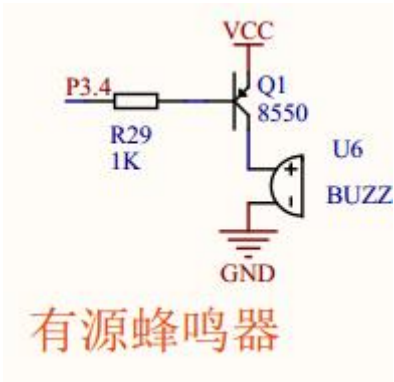
【108.1 按键控制跑马灯的启动和暂停和停止。】



上图 108.1.1 独立按键



上图 108.1.2 LED 电路



上图 108.1.3 有源蜂鸣器的电路

在我眼里，按键不仅仅是按键，跑马灯不仅仅是跑马灯。按键是输入设备，跑马灯是应用程序。本节表面上讲按键控制跑马灯的简单项目，实际上作者用心良苦立意深远，试图通过按键与跑马灯，来分享一种输入设备如何关联应用程序的程序框架。

本节例程实现的功能如下：

(1) 【启动暂停】按键 K1。按下【启动暂停】按键 K1 启动之后，跑马灯处于“启动”状态，4 个 LED 灯从左到右依次循环的变亮，给人“跑”起来的感觉。此时如果再按一次【启动暂停】按键 K1，则跑马灯处于“暂停”状态，如果再按一次【启动暂停】按键 K1，跑马灯又变回“启动”状态。因此，【启动暂停】按键 K1 是专门用来切换“启动”和“暂停”这两种状态。

(2) 【停止】按键 K2。当跑马灯处于“启动”或者“暂停”或者“停止”的状态时，只要按下【停止】按键 K2，当前的运动状态就终止，强制变回初始的“停止”状态，类似“复位”按键的作用。当跑马灯处于“停止”状态时，此时再按下【启动暂停】按键 K1 之后，跑马灯又处于“启动”状态。

```
#include "REG52.H"

#define KEY_VOICE_TIME    50
#define KEY_FILTER_TIME  25
#define RUN_TIME    200    //跑马灯的跑动速度的时间参数

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

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

//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;

//蜂鸣器的输出口
sbit P3_4=P3^4;

sbit KEY_INPUT1=P2^2;    //【启动暂停】按键 K1 的输入口。
sbit KEY_INPUT2=P2^1;    //【停止】按键 K2 的输入口。

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

volatile unsigned char vGu8KeySec=0;

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。
```

```
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
        RunTask(); //跑马灯的任务函数
    }
}
```

```
void TO_time() interrupt 1
```

```
{
    VoiceScan();
    KeyScan();

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

    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)
{
    //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
}

void BeepOpen(void)
{
    P3_4=0;
}

void BeepClose(void)
{
    P3_4=1;
}

void VoiceScan(void)
{
    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 KeyScan(void) //此函数放在定时中断里每 1ms 扫描一次
{
    static unsigned char Su8KeyLock1;
    static unsigned int  Su16KeyCnt1;
    static unsigned char Su8KeyLock2;
    static unsigned int  Su16KeyCnt2;

    // 【启动暂停】按键 K1 的扫描识别
    if(0!=KEY_INPUT1)
    {
        Su8KeyLock1=0;
        Su16KeyCnt1=0;
    }
    else if(0==Su8KeyLock1)
    {
        Su16KeyCnt1++;
        if(Su16KeyCnt1>=KEY_FILTER_TIME)
        {
            Su8KeyLock1=1;
            vGu8KeySec=1;    //触发 1 号键
        }
    }

    // 【停止】按键 K2 的扫描识别
    if(0!=KEY_INPUT2)
    {
        Su8KeyLock2=0;
        Su16KeyCnt2=0;
    }
    else if(0==Su8KeyLock2)
    {
        Su16KeyCnt2++;
        if(Su16KeyCnt2>=KEY_FILTER_TIME)
        {
            Su8KeyLock2=1;
            vGu8KeySec=2;    //触发 2 号键
        }
    }
}

```

```
}
```

```
/* 注释一：
```

```
* 本节破题的关键：
```

```
* 在 KeyTask 和 RunTask 两个任务函数之间，主要是靠 Gu8RunStart 和 Gu8RunStatus 这两个
```

```
* 全局变量来传递信息。
```

```
*/
```

```
void KeyTask(void)    //按键的任务函数，放在主函数内
```

```
{
```

```
    if(0==vGu8KeySec)
```

```
    {
```

```
        return; //按键的触发序号是 0 意味着无按键触发，直接退出当前函数，不执行此函数下面的代码
```

```
    }
```

```
    switch(vGu8KeySec) //根据不同的按键触发序号执行对应的代码
```

```
    {
```

```
        case 1:    //1 号按键。【启动暂停】按键 K1
```

```
            if(0==Gu8RunStatus) //当跑马灯处于“停止”状态时
```

```
            {
```

```
                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=KEY_VOICE_TIME; //触发按键后，发出固定长度的声音
```

```
        vGu8BeepTimerFlag=1;
```

```
        vGu8KeySec=0; //响应按键服务处理程序后，按键编号必须清零，避免一直触发
```

```
        break;
```

```
        case 2:    //2 号按键。【停止】按键 K2
```

```
            Gu8RunStart=0;    //总开关“关闭”。
```

```

        Gu8RunStatus=0; //状态切换到“停止”状态

        vGu8BeepTimerFlag=0;
        vGu16BeepTimerCnt=KEY_VOICE_TIME; //触发按键后，发出固定长度的声音
        vGu8BeepTimerFlag=1;
        vGu8KeySec=0; //响应按键服务处理程序后，按键编号必须清零，避免一直触发
        break;
    }
}

void RunTask(void) //跑马灯的任务函数，放在主函数内
{
    static unsigned char Su8RunStep=0; //运行的步骤

    //当总开关处于“停止”并且“步骤不为0”时，强制把步骤归零, 跑马灯初始化。
    if (0!=Su8RunStep&&0==Gu8RunStart)
    {
        Su8RunStep=0; //步骤归零

        //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
    }

    switch(Su8RunStep) //屡见屡爱的 switch 又来了
    {
        case 0:
            if(1==Gu8RunStart) //总开关“打开”
            {
                vGu8RunTimerFlag=0;
                vGu16RunTimerCnt=0; //定时器清零
                Su8RunStep=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=RUN_TIME;    //用于控制跑马灯跑动速度的定时器
        vGu8RunTimerFlag=1;    //启动定时器
        Su8RunStep=2;    //切换到下一步
    }

    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=RUN_TIME;    //用于控制跑马灯跑动速度的定时器
        vGu8RunTimerFlag=1;    //启动定时器
        Su8RunStep=3;    //切换到下一步
    }

    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=RUN_TIME;    //用于控制跑马灯跑动速度的定时器
        vGu8RunTimerFlag=1;    //启动定时器
        Su8RunStep=4;    //切换到下一步
    }

    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=RUN_TIME;    //用于控制跑马灯跑动速度的定时器
vGu8RunTimerFlag=1;    //启动定时器
Su8RunStep=1;    //返回到第 1 步，重新开始下一轮的循环!!!
}

break;

}

}
```