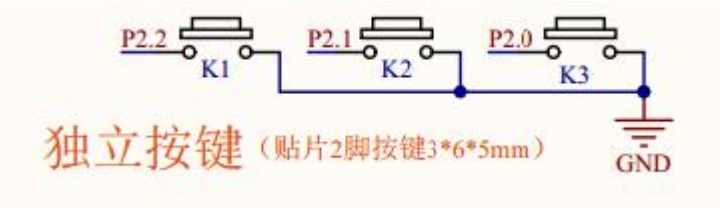
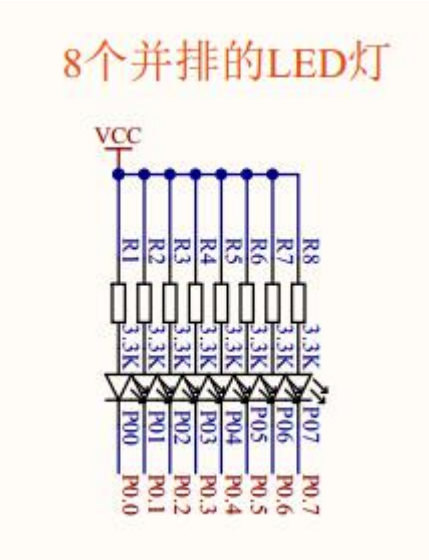
第九十七节： 独立按键按住不松手的连续均匀触发。

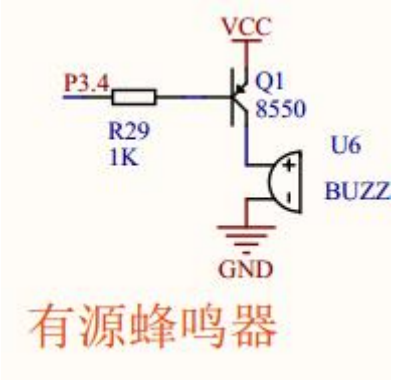
【97.1　按住不松手的连续均匀触发。】



上图 97.1.1　独立按键电路



上图 97.1.2　灌入式驱动 8 个 LED



上图 97.1.3　有源蜂鸣器电路

在电脑上删除某个文件某行文字的时候，单击一次“退格按键[Backspace]”，就删除一个文字，如果按住“退格按键[Backspace]”不松手，就会“连续均匀”的触发“删除”的功能，自动逐个把整行文字删

除清空，这就是“按住不松手的连续均匀触发”应用案例之一。除此之外，在很多需要人机交互的项目中都有这样的功能，为了快速加减某个数值，按住某个按键不松手，某个数值有节奏地快速往上加或者快速往下减。这种“按住不松手连续均匀触发”的按键识别，在程序上有“3 个时间”需要留意，第 1 个是按键单击的“滤波”时间，第 2 个是按键“从单击进入连击”的间隔时间（此时间是“单击”与“连击”的分界线），第 3 个是按键“连击”的间隔时间，

本节例程实现的功能如下：（1）8 个受按键控制的跑马灯在某一时刻只有 1 个 LED 亮，每触发一次 K1 按键，“亮的 LED”就“往左边跑一步”；相反，每触发一次 K2 按键，“亮的 LED”就“往右边跑一步”。如果按住 K1 或者 K2 不松手就连续触发，“亮的 LED”就“连续跑”，一直跑到左边或者右边的尽头。（2）按键每“单击”一次蜂鸣器就鸣叫一次，但是，当按键“从单击进入连击”后，蜂鸣器就不鸣叫。

```
#include "REG52.H"

#define KEY_VOICE_TIME   50

#define KEY_SHORT_TIME  25      //按键单击的“滤波”时间 25ms
#define KEY_ENTER_CONTINUITY_TIME    300  //按键“从单击进入连击”的间隔时间 300ms
#define KEY_CONTINUITY_TIME    80  //按键“连击”的间隔时间 80ms

#define BUS_P0    P0     //8 个 LED 灯一一对应单片机的 P0 口总线

void T0_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 DisplayTask(void);   //显示的任务函数（LED 显示状态）

sbit P3_4=P3^4;       //蜂鸣器

sbit KEY_INPUT1=P2^2;  //K1 按键识别的输入口。
sbit KEY_INPUT2=P2^1;  //K2 按键识别的输入口。

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

unsigned char Gu8LedStatus=0; //LED 灯的状态
unsigned char Gu8DisplayUpdate=1; //显示的刷新标志
```

```
volatile unsigned char vGu8KeySec=0;  //按键的触发序号
volatile unsigned char vGu8ShieldVoiceFlag=0;  //屏蔽声音的标志

void main()
{
    SystemInitial();
    Delay(10000);
    PeripheralInitial();
    while(1)
    {
        KeyTask();       //按键的任务函数
        DisplayTask();   //显示的任务函数（LED 显示状态）
    }
}



/* 注释一：
* Gu8DisplayUpdate 这类“显示刷新变量”在“显示框架”里是很常见的，而且屡用屡爽。
* 目的是，既能及时刷新显示，又能避免主函数“不断去执行显示代码”而影响程序效率。
*/

void DisplayTask(void)   //显示的任务函数（LED 显示状态）
{
    if(1==Gu8DisplayUpdate)  //需要刷新一次显示
    {
        Gu8DisplayUpdate=0;  //及时清零，避免主函数“不断去执行显示代码”而影响程序效率

        //Gu8LedStatus 是左移的位数，范围（0 至 7），决定了跑马灯的显示状态。
        BUS_P0=~(1<<Gu8LedStatus);  //“左移<<”之后的“取反~”，因为 LED 电路是灌入式驱动方式。
    }
}

/* 注释二：
* 按键“连续均匀触发”的识别过程：
* 第一步：平时只要 K1 没有被按下，按键的自锁标志 Su8KeyLock1、去抖动延时计数器 Su16KeyCnt1、
*         连击计数器 Su16KeyContinuityCnt1，一直被清零。
* 第二步：一旦 K1 按键被按下，去抖动延时计数器 Su16KeyCnt1 开始在定时中断函数里累加，在还没
*         累加到阀值 KEY_SHORT_TIME 时，如果在这期间由于受外界干扰或者按键抖动，
*         而使 IO 口突然瞬间触发成高电平，这个时候马上把延时计数器 Su16KeyCnt1 清零，
*         这个过程非常巧妙，非常有效地去除瞬间的杂波干扰。
* 第三步：如果 K1 按键按下的时间超过了阀值 KEY_SHORT_TIME，则触发一次“单击”， 同时，马上把自锁
*         标志 Su8KeyLock1 置 1 防止按住按键不松手后一直触发，并且把计数器 Su16KeyCnt1 清零为了下
*         一步用来累加“从单击进入连击的间隔时间 1000ms”。如果此时还没有松手，直到发现按下的时
*         间超过“从单击进入连击的间隔时间”阀值 KEY_ENTER_CONTINUITY_TIME 时，从此进入“连击”
```

```
*         的模式，连击计数器 Su16KeyContinuityCnt1 开始累加，每到达一次阀值
*         KEY_CONTINUITY_TIME 就触发 1 次按键，为了屏蔽按键声音及时把 vGu8ShieldVoiceFlag 也置 1，
*         同时，Su16KeyContinuityCnt1 马上清零为继续连击作准备。
* 第四步：等 K1 按键松手后，自锁标志 Su8KeyLock1、去抖动延时计数器 Su16KeyCnt1、
*         连击计数器 Su16KeyContinuityCnt1，及时清零，为下一次按键触发做准备。
*/

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 int  Su16KeyContinuityCnt1;  //连击计数器

   static unsigned char Su8KeyLock2;
   static unsigned int  Su16KeyCnt2;
   static unsigned int  Su16KeyContinuityCnt2;  //连击计数器

   //K1 按键
   if(0!=KEY_INPUT1)//单个 K1 按键没有按下，及时清零一些标志。
   {
      Su8KeyLock1=0; //按键解锁
      Su16KeyCnt1=0;  //去抖动延时计数器清零，此行非常巧妙，是全场的亮点。
      Su16KeyContinuityCnt1=0;  //连击计数器
   }
   else if(0==Su8KeyLock1)//单个按键 K1 被按下
   {
      Su16KeyCnt1++; //累加定时中断次数，每一次累加额度是 1ms
      if(Su16KeyCnt1>=KEY_SHORT_TIME) //按键的“滤波”时间 25ms
      {
            Su8KeyLock1=1;      //“自锁”
            vGu8KeySec=1;       //触发一次 K1 按键
            Su16KeyCnt1=0;      //清零，为了下一步用来累加“从单击进入连击的间隔时间 300ms”
      }
   }
   else if(Su16KeyCnt1<=KEY_ENTER_CONTINUITY_TIME)//按住不松手累加到 300ms
   {
      Su16KeyCnt1++; //累加定时中断次数，每一次累加额度是 1ms
   }
   else  //按住累加到 300ms 后仍然不放手，这个时候进入有节奏的连续触发
   {
       Su16KeyContinuityCnt1++; //连击计数器开始累加，每一次累加额度是 1ms
       if(Su16KeyContinuityCnt1>=KEY_CONTINUITY_TIME)  //按住没松手，每 0.08 秒就触发一次
       {
            Su16KeyContinuityCnt1=0; //清零，为了继续连击。
```

```
            vGu8KeySec=1;       //触发一次 K1 按键
            vGu8ShieldVoiceFlag=1;  //把当前按键触发的声音屏蔽掉
        }

    }

   //K2 按键
   if(0!=KEY_INPUT2)
   {
      Su8KeyLock2=0;
      Su16KeyCnt2=0;
      Su16KeyContinuityCnt2=0;
   }
   else if(0==Su8KeyLock2)
   {
      Su16KeyCnt2++;
      if(Su16KeyCnt2>=KEY_SHORT_TIME)
      {
            Su8KeyLock2=1;
            vGu8KeySec=2;       //触发一次 K2 按键
            Su16KeyCnt2=0;
      }
   }
   else if(Su16KeyCnt2<=KEY_ENTER_CONTINUITY_TIME)
   {
      Su16KeyCnt2++;
   }
   else
   {
       Su16KeyContinuityCnt2++;
       if(Su16KeyContinuityCnt2>=KEY_CONTINUITY_TIME)
       {
            Su16KeyContinuityCnt2=0;
            vGu8KeySec=2;       //触发一次 K2 按键
            vGu8ShieldVoiceFlag=1;  //把当前按键触发的声音屏蔽掉
       }

  }
}

void KeyTask(void)    //按键任务函数，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:     //K1 触发的任务
            if(Gu8LedStatus>0)
            {
                Gu8LedStatus--;  //控制 LED“往左边跑”
                Gu8DisplayUpdate=1;  //刷新显示
            }

            if(0==vGu8ShieldVoiceFlag) //声音没有被屏蔽
            {
                vGu8BeepTimerFlag=0;
                vGu16BeepTimerCnt=KEY_VOICE_TIME;  //触发一次“长按”后，发出“嘀”一声
                vGu8BeepTimerFlag=1;
            }

            vGu8ShieldVoiceFlag=0;  //及时把屏蔽标志清零，避免平时正常的单击声音也被淹没。
            vGu8KeySec=0;  //响应按键服务处理程序后，按键编号必须清零，避免一致触发
            break;

       case 2:     //K2 触发的任务
            if(Gu8LedStatus<7)
            {
                Gu8LedStatus++;  //控制 LED“往右边跑”
                Gu8DisplayUpdate=1;  //刷新显示
            }

            if(0==vGu8ShieldVoiceFlag) //声音没有被屏蔽
            {
                vGu8BeepTimerFlag=0;
                vGu16BeepTimerCnt=KEY_VOICE_TIME;  //触发一次“长按”后，发出“嘀”一声
                vGu8BeepTimerFlag=1;
            }

            vGu8ShieldVoiceFlag=0;  //及时把屏蔽标志清零，避免平时正常的单击声音也被淹没。
            vGu8KeySec=0;  //响应按键服务处理程序后，按键编号必须清零，避免一致触发
            break;

    }
}
```

```
void T0_time() interrupt 1
{
    VoiceScan();
    KeyScan();    //按键识别的驱动函数

    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)
{

}

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();
            }

        }
    }
}
```