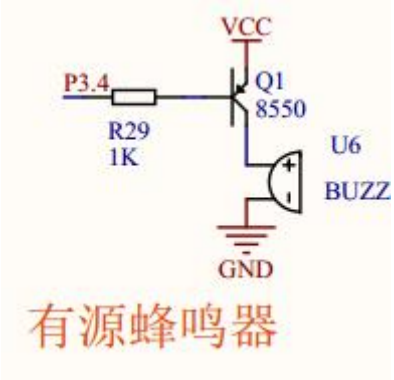
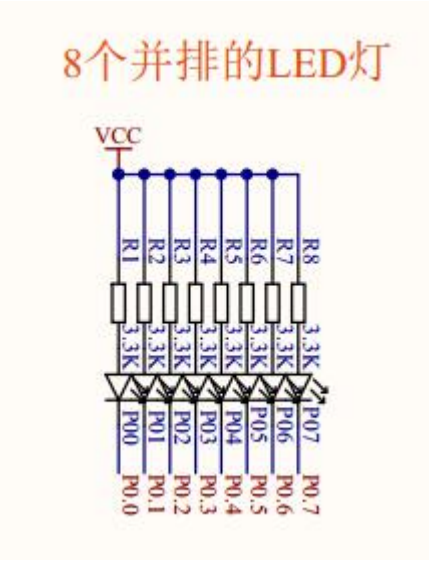
第一百零五节： 矩阵按键按住不松手的连续均匀触发。

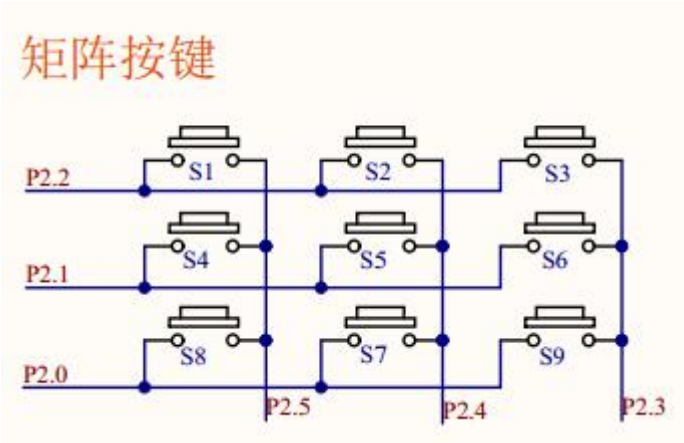
【105.1　按住不松手的连续均匀触发。】



上图 105.1.1　有源蜂鸣器电路



上图 105.1.2　LED 电路



上图 105.1.3　3*3 矩阵按键的电路

矩阵按键与前面章节独立按键的“按住不松手的连续均匀触发”的处理思路是一样的。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 S1 按键，“亮的 LED”就“往左边跑一步”；相反，每触发一次 S9 按键，“亮的 LED”就“往右边跑一步”。如果按住 S1 或者 S9 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 S1 或者 S9 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。代码如下：

```
#include "REG52.H"

#define KEY_VOICE_TIME   50

#define KEY_SHORT_TIME  20    //按键单击的“滤波”时间

#define KEY_ENTER_CONTINUITY_TIME    240  //按键“从单击进入连击”的间隔时间
#define KEY_CONTINUITY_TIME    64   //按键“连击”的间隔时间

#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;


sbit ROW_INPUT1=P2^2;  //第 1 行输入口。
```

```
sbit ROW_INPUT2=P2^1;  //第 2 行输入口。
sbit ROW_INPUT3=P2^0;  //第 3 行输入口。

sbit COLUMN_OUTPUT1=P2^5;  //第 1 列输出口。
sbit COLUMN_OUTPUT2=P2^4;  //第 2 列输出口。
sbit COLUMN_OUTPUT3=P2^3;  //第 3 列输出口。

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();
        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：
*  本节破题的关键：
*  矩阵按键涉及的按键数量很多，但是实际项目上一般只需要少数个别按键具备这种
*  “单击”与“连续均匀触发”的特殊技能，因此，在代码上，必须把这类“特殊技能按键”与
*  “大众按键”区分开来，才能相互清晰互不干扰。本节的“特殊技能按键”是 S1 和 S9。
*  如果觉得本节的讲解不够详细具体，请先阅读一下前面章节“独立按键按住不松手的连续均匀触发”。
*/

void KeyScan(void)  //此函数放在定时中断里每 1ms 扫描一次
{
   static unsigned char Su8KeyLock=0;
   static unsigned int  Su16KeyCnt=0;
   static unsigned char Su8KeyStep=1;

   static unsigned char Su8ColumnRecord=0;

   switch(Su8KeyStep)
   {
     case 1:
          if(0==Su8ColumnRecord)
          {
              COLUMN_OUTPUT1=0;
              COLUMN_OUTPUT2=1;
              COLUMN_OUTPUT3=1;
          }
          else if(1==Su8ColumnRecord)
          {
              COLUMN_OUTPUT1=1;
              COLUMN_OUTPUT2=0;
              COLUMN_OUTPUT3=1;
          }
          else
          {
              COLUMN_OUTPUT1=1;
              COLUMN_OUTPUT2=1;
              COLUMN_OUTPUT3=0;
          }
          Su16KeyCnt=0;
          Su8KeyStep++;
          break;

     case 2:      //等待列输出稳定，但不是去抖动延时
          Su16KeyCnt++;
          if(Su16KeyCnt>=2)
```

```
        {
           Su16KeyCnt=0;
           Su8KeyStep++;
        }
        break;

    case 3:
        if(1==ROW_INPUT1&&1==ROW_INPUT2&&1==ROW_INPUT3)
        {
            Su8KeyStep=1;    //返回步骤 1 继续扫描
            Su8KeyLock=0;
            Su16KeyCnt=0;

            Su8ColumnRecord++;
            if(Su8ColumnRecord>=3)
            {
               Su8ColumnRecord=0;
            }
        }
        else if(0==Su8KeyLock)
        {
            if(0==ROW_INPUT1&&1==ROW_INPUT2&&1==ROW_INPUT3)
            {
                Su16KeyCnt++;
                if(Su16KeyCnt>=KEY_SHORT_TIME)
                {
                    Su8KeyLock=1;

                    if(0==Su8ColumnRecord)
                    {
                          vGu8KeySec=1;    //触发一次单击
                          Su16KeyCnt=0;    //计时器清零，为即将来临的计时做准备
                          Su8KeyStep=4;    //跳到 S1 按键的专属区，脱离大众按键
                    }
                    else if(1==Su8ColumnRecord)
                    {
                          vGu8KeySec=2;
                    }
                    else if(2==Su8ColumnRecord)
                    {
                          vGu8KeySec=3;
                    }
                }
```

```
    }
    else if(1==ROW_INPUT1&&0==ROW_INPUT2&&1==ROW_INPUT3)
    {
        Su16KeyCnt++;
        if(Su16KeyCnt>=KEY_SHORT_TIME)
        {
            Su8KeyLock=1;

            if(0==Su8ColumnRecord)
            {
                 vGu8KeySec=4;
            }
            else if(1==Su8ColumnRecord)
            {
                 vGu8KeySec=5;
            }
            else if(2==Su8ColumnRecord)
            {
                 vGu8KeySec=6;
            }
        }
    }
    else if(1==ROW_INPUT1&&1==ROW_INPUT2&&0==ROW_INPUT3)
    {
        Su16KeyCnt++;
        if(Su16KeyCnt>=KEY_SHORT_TIME)
        {
            Su8KeyLock=1;
            if(0==Su8ColumnRecord)
            {
                 vGu8KeySec=7;
            }
            else if(1==Su8ColumnRecord)
            {
                 vGu8KeySec=8;
            }
            else if(2==Su8ColumnRecord)
            {
                  vGu8KeySec=9;     //触发一次单击
                  Su16KeyCnt=0;     //计时器清零，为即将来临的计时做准备
                  Su8KeyStep=6;     //跳到 S9 按键的专属区，脱离大众按键
            }
        }
    }
```

```
        }
        break;

/*---------S1 按键的专属区----------------*/
    case 4:
        if(0==ROW_INPUT1&&1==ROW_INPUT2&&1==ROW_INPUT3) //仅判断 S1 按键，避免交叉影响
        {
            Su16KeyCnt++;
            if(Su16KeyCnt>=KEY_ENTER_CONTINUITY_TIME)//该时间是“单击”与“连击”的分界线
            {
                Su16KeyCnt=0;    //计时器清零，为即将来临的计时做准备
                Su8KeyStep=5;    //S1 按键进入有节奏的连续触发
            }
        }
        else //如果期间检查到 S1 按键已经松手
        {
            Su8KeyStep=1;    //返回步骤 1 继续扫描
            Su8KeyLock=0;
            Su16KeyCnt=0;

            Su8ColumnRecord++;
            if(Su8ColumnRecord>=3)
            {
               Su8ColumnRecord=0;
            }
        }

        break;
    case 5:  //S1 按键进入有节奏的连续触发
        if(0==ROW_INPUT1&&1==ROW_INPUT2&&1==ROW_INPUT3) //仅判断 S1 按键，避免交叉影响
        {
            Su16KeyCnt++;
            if(Su16KeyCnt>=KEY_CONTINUITY_TIME)  //该时间是“连击”的时间
            {
                Su16KeyCnt=0;           //清零，为了继续连击。
                vGu8KeySec=1;           //触发一次 S1 按键
                vGu8ShieldVoiceFlag=1;  //因为连击，把当前按键触发的声音屏蔽掉
            }
        }
        else //如果期间检查到 S1 按键已经松手
        {
            Su8KeyStep=1;    //返回步骤 1 继续扫描
            Su8KeyLock=0;
```

```
            Su16KeyCnt=0;

            Su8ColumnRecord++;
            if(Su8ColumnRecord>=3)
            {
               Su8ColumnRecord=0;
            }
        }
        break;

/*---------S9 按键的专属区----------------*/
    case 6:
        if(1==ROW_INPUT1&&1==ROW_INPUT2&&0==ROW_INPUT3) //仅判断 S9 按键，避免交叉影响
        {
            Su16KeyCnt++;
            if(Su16KeyCnt>=KEY_ENTER_CONTINUITY_TIME)//该时间是“单击”与“连击”的分界线
            {
                Su16KeyCnt=0;    //计时器清零，为即将来临的计时做准备
                Su8KeyStep=7;    //S9 按键进入有节奏的连续触发
            }
        }
        else //如果期间检查到 S9 按键已经松手
        {
            Su8KeyStep=1;    //返回步骤 1 继续扫描
            Su8KeyLock=0;
            Su16KeyCnt=0;

            Su8ColumnRecord++;
            if(Su8ColumnRecord>=3)
            {
               Su8ColumnRecord=0;
            }
        }

        break;
    case 7:  //S9 按键进入有节奏的连续触发
        if(1==ROW_INPUT1&&1==ROW_INPUT2&&0==ROW_INPUT3) //仅判断 S9 按键，避免交叉影响
        {
            Su16KeyCnt++;
            if(Su16KeyCnt>=KEY_CONTINUITY_TIME)  //该时间是“连击”的时间
            {
                Su16KeyCnt=0;           //清零，为了继续连击。
                vGu8KeySec=9;           //触发一次 S9 按键
                vGu8ShieldVoiceFlag=1;  //因为连击，把当前按键触发的声音屏蔽掉
```

```
            }
        }
        else //如果期间检查到 S9 按键已经松手
        {
            Su8KeyStep=1;    //返回步骤 1 继续扫描
            Su8KeyLock=0;
            Su16KeyCnt=0;

            Su8ColumnRecord++;
            if(Su8ColumnRecord>=3)
            {
               Su8ColumnRecord=0;
            }
        }
        break;


   }

}

void KeyTask(void)
{
    if(0==vGu8KeySec)
    {
        return;
    }

    switch(vGu8KeySec)
    {
       case 1:     //S1 按键的任务
            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;  //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 9:     //S9 按键的任务
            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;  //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;

        default:

            vGu8KeySec=0;
            break;

    }
}

void T0_time() interrupt 1
{
    VoiceScan();
    KeyScan();

    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();
                }

            }
        }
}
```