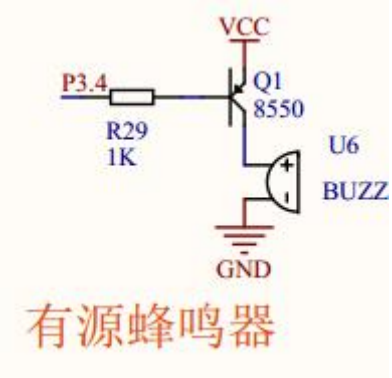
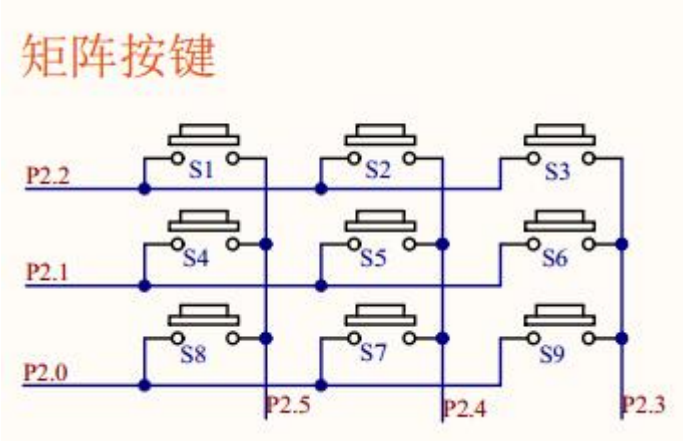
第一百零三节： 两个“任意行输入”矩阵按键的“无序”组合触发。

【103.1　“无序”组合触发。】




上图 103.1.1　有源蜂鸣器电路


上图 103.1.2　LED 电路



上图 103.1.3　3*3 矩阵按键的电路

“无序”是指两个组合按键不分先后顺序，都能构成组合触发。比如，要触发组合键（S1+S2），先按 S1 再按 S2，或者先按 S2 再按 S1，功能都是一样的。

本节程序功能如下：（1）S1 每单击一次，P1.4 所在的 LED 要么从“灭”变成“亮”，要么从“亮”变成“灭”，在两种状态之间切换。（2）S2 每单击一次，P1.5 所在的 LED 要么从“灭”变成“亮”，要么从“亮”变成“灭”，在两种状态之间切换。（3）如果先按住 S1 再按 S2，或者先按住 S2 再按 S1，都认为构造了“无序”组合键，蜂鸣器发出“嘀”的一声。

```
#include "REG52.H"

#define KEY_VOICE_TIME   50
#define KEY_SHORT_TIME  20

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

void BeepOpen(void);
void BeepClose(void);
void LedOpen_P1_4(void);
void LedClose_P1_4(void);
void LedOpen_P1_5(void);
void LedClose_P1_5(void);

void VoiceScan(void);
void KeyScan(void);
void SingleKeyTask(void);
void DoubleKeyTask(void);

sbit P3_4=P3^4;
sbit P1_4=P1^4;    //P1.4 所在的 LED
sbit P1_5=P1^5;    //P1.5 所在的 LED

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_P1_4=0;  //P1.4 所在的 LED 的状态
unsigned char Gu8LedStatus_P1_5=0;  //P1.5 所在的 LED 的状态
volatile unsigned char vGu8SingleKeySec=0;
volatile unsigned char vGu8DoubleKeySec=0;

void main()
{
    SystemInitial();
    Delay(10000);
    PeripheralInitial();
    while(1)
    {
        SingleKeyTask();
        DoubleKeyTask();
    }
}

/* 注释一：
*  矩阵按键“无序”触发的两个最关键地方：
*  （1）如果是 S1 按键先被按下并且单击触发之后，“马上更新输出列的信号状态”，然后切换到
*   “S1 后面所在的步骤里”，进入到 S1 和 S2 两个按键的轮番循环监控之中，如果发现 S1 按键率先
*   被松开了，就把步骤切换到开始的第一步，重新开始新一轮的按键扫描。
*  （2）如果是 S2 按键先被按下并且单击触发之后，“马上更新输出列的信号状态”，然后切换到
*   “S2 后面所在的步骤里”，进入到 S1 和 S2 两个按键的轮番循环监控之中，如果发现 S2 按键率先
*   被松开了，就把步骤切换到开始的第一步，重新开始新一轮的按键扫描。
*  （3）上面两个描述中的两种步骤，“S1 后面所在的步骤里”和“S2 后面所在的步骤里”是分开的，
*   不共用的，这是本节破题的关键。
*/

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;
           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)
                    {
                         vGu8SingleKeySec=1;     //单击任务，触发 1 号键 对应 S1 键

                         //“马上更新输出列的信号状态”
                         COLUMN_OUTPUT1=1;
                         COLUMN_OUTPUT2=0;   //列 2 也输出 0，下一步监控 S2，非常关键的代码!
                         COLUMN_OUTPUT3=1;

                         Su16KeyCnt=0;  //去抖动延时清零，为下一步计时做准备
                         Su8KeyStep=4;   //切换到“S1 后面所在的步骤里”，破题的关键!!!
                    }
                    else if(1==Su8ColumnRecord)
                    {
                         vGu8SingleKeySec=2;    //单击任务，触发 2 号键 对应 S2 键

                         //“马上更新输出列的信号状态”
                         COLUMN_OUTPUT1=0;  //列 1 也输出 0，下一步监控 S1，非常关键的代码!
                         COLUMN_OUTPUT2=1;
                         COLUMN_OUTPUT3=1;

                         Su16KeyCnt=0;  //去抖动延时清零，为下一步计时做准备
                         Su8KeyStep=8;   //切换到“S2 后面所在的步骤里”，破题的关键!!!
                    }
                    else if(2==Su8ColumnRecord)
                    {
                         vGu8SingleKeySec=3;
                    }
                }

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

```
                    if(0==Su8ColumnRecord)
                    {
                        vGu8SingleKeySec=4;
                    }
                    else if(1==Su8ColumnRecord)
                    {
                        vGu8SingleKeySec=5;
                    }
                    else if(2==Su8ColumnRecord)
                    {
                        vGu8SingleKeySec=6;
                    }
                }
            }
            else if(1==ROW_INPUT1&&1==ROW_INPUT2&&0==ROW_INPUT3)
            {
                Su16KeyCnt++;
                if(Su16KeyCnt>=KEY_SHORT_TIME)
                {
                    Su8KeyLock=1;
                    if(0==Su8ColumnRecord)
                    {
                        vGu8SingleKeySec=7;
                    }
                    else if(1==Su8ColumnRecord)
                    {
                        vGu8SingleKeySec=8;
                    }
                    else if(2==Su8ColumnRecord)
                    {
                        vGu8SingleKeySec=9;
                    }
                }
            }

        }
        break;

/*--------------“S1 后面所在的步骤里”------------------*/
    case 4:             //等待列输出稳定，但不是去抖动延时
        Su16KeyCnt++;
        if(Su16KeyCnt>=2)
        {
```

```
            Su16KeyCnt=0;
            Su8KeyLock=0;   //关键语句！自锁清零，为下一步自锁组合按键做准备
            Su8KeyStep++;
        }
        break;

    case 5:    //判断 S2 按键
        if(1==ROW_INPUT1&&1==ROW_INPUT2&&1==ROW_INPUT3) //S2 按键没有被按下
        {
           Su8KeyLock=0;
           Su16KeyCnt=0;

           //“马上更新输出列的信号状态”
           COLUMN_OUTPUT1=0;     //列 1 输出 0，下一步监控 S1，非常关键的代码!
           COLUMN_OUTPUT2=1;
           COLUMN_OUTPUT3=1;

           Su8KeyStep++;    //切换到下一个步骤，监控 S1 是否率先已经松开
        }
        else if(0==Su8KeyLock)
        {
            if(0==ROW_INPUT1&&1==ROW_INPUT2&&1==ROW_INPUT3) //S2 按键被按下
            {
                Su16KeyCnt++;
                if(Su16KeyCnt>=KEY_SHORT_TIME)
                {
                      Su8KeyLock=1;  //组合按键的自锁
                      vGu8DoubleKeySec=1;   //触发组合按键(S1+S2)
                }
            }



        }
        break;

    case 6:       //等待列输出稳定，但不是去抖动延时
        Su16KeyCnt++;
        if(Su16KeyCnt>=2)
        {
            Su16KeyCnt=0;
            Su8KeyLock=0;   //关键语句！自锁清零，为下一步自锁组合按键做准备
            Su8KeyStep++;
        }
```

```
        break;

    case 7:    //监控 S1 按键是否率先已经松开
        if(1==ROW_INPUT1&&1==ROW_INPUT2&&1==ROW_INPUT3)
        {
            Su16KeyCnt=0;
            Su8KeyLock=0;
            Su8KeyStep=1;   //如果 S1 按键已经松开，返回到第一个运行步骤重新开始扫描

            Su8ColumnRecord++;
            if(Su8ColumnRecord>=3)
            {
                Su8ColumnRecord=0;
            }
        }
        else
        {
            //“马上更新输出列的信号状态”
            COLUMN_OUTPUT1=1;
            COLUMN_OUTPUT2=0;     //列 2 输出 0，下一步监控 S2，非常关键的代码!
            COLUMN_OUTPUT3=1;
            Su8KeyStep=4;   //如果 S1 按键没有松开，继续返回判断 S2 是否已按下
        }
        break;

/*--------------“S2 后面所在的步骤里”------------------*/
    case 8:             //等待列输出稳定，但不是去抖动延时
        Su16KeyCnt++;
        if(Su16KeyCnt>=2)
        {
            Su16KeyCnt=0;
            Su8KeyLock=0;   //关键语句! 自锁清零，为下一步自锁组合按键做准备
            Su8KeyStep++;
        }
        break;

    case 9:    //判断 S1 按键
        if(1==ROW_INPUT1&&1==ROW_INPUT2&&1==ROW_INPUT3) //S1 按键没有被按下
        {
           Su8KeyLock=0;
           Su16KeyCnt=0;

           //“马上更新输出列的信号状态”
           COLUMN_OUTPUT1=1;
```

```
            COLUMN_OUTPUT2=0;  //列 2 输出 0，下一步监控 S2，非常关键的代码!
            COLUMN_OUTPUT3=1;

            Su8KeyStep++;    //切换到下一个步骤，监控 S2 是否率先已经松开
        }
        else if(0==Su8KeyLock)
        {
            if(0==ROW_INPUT1&&1==ROW_INPUT2&&1==ROW_INPUT3) //S1 按键被按下
            {
                Su16KeyCnt++;
                if(Su16KeyCnt>=KEY_SHORT_TIME)
                {
                      Su8KeyLock=1;  //组合按键的自锁
                      vGu8DoubleKeySec=1;   //触发组合按键(S1+S2)
                }
            }



        }
        break;

    case 10:       //等待列输出稳定，但不是去抖动延时
        Su16KeyCnt++;
        if(Su16KeyCnt>=2)
        {
            Su16KeyCnt=0;
            Su8KeyLock=0;   //关键语句! 自锁清零，为下一步自锁组合按键做准备
            Su8KeyStep++;
        }
        break;

    case 11:    //监控 S2 按键是否率先已经松开
        if(1==ROW_INPUT1&&1==ROW_INPUT2&&1==ROW_INPUT3)
        {
            Su16KeyCnt=0;
            Su8KeyLock=0;
            Su8KeyStep=1;   //如果 S2 按键已经松开，返回到第一个运行步骤重新开始扫描

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

```
        }
        else
        {
            //“马上更新输出列的信号状态”
            COLUMN_OUTPUT1=0;    //列 1 输出 0，下一步监控 S1，非常关键的代码!
            COLUMN_OUTPUT2=1;
            COLUMN_OUTPUT3=1;
            Su8KeyStep=8;   //如果 S2 按键没有松开，继续返回判断 S1 是否已按下
        }
        break;


   }

}

void SingleKeyTask(void)
{
    if(0==vGu8SingleKeySec)
    {
        return;
    }

    switch(vGu8SingleKeySec)
    {
       case 1:     //S1 按键的单击任务，更改 P1.4 所在的 LED 灯的显示状态

           if(0==Gu8LedStatus_P1_4)
           {
               Gu8LedStatus_P1_4=1;
               LedOpen_P1_4();
           }
           else
           {
               Gu8LedStatus_P1_4=0;
               LedClose_P1_4();
           }

           vGu8SingleKeySec=0;
           break;

       case 2:     //S2 按键的单击任务，更改 P1.5 所在的 LED 灯的显示状态

           if(0==Gu8LedStatus_P1_5)
```

```
            {
                Gu8LedStatus_P1_5=1;
                LedOpen_P1_5();
            }
            else
            {
                Gu8LedStatus_P1_5=0;
                LedClose_P1_5();
            }

            vGu8SingleKeySec=0;
            break;

       default:

            vGu8SingleKeySec=0;
            break;

    }
}

void DoubleKeyTask(void)
{
    if(0==vGu8DoubleKeySec)
    {
        return;
    }

    switch(vGu8DoubleKeySec)
    {
        case 1:     //S1 与 S2 的组合按键触发，发出“嘀”一声

              vGu8BeepTimerFlag=0;
              vGu16BeepTimerCnt=KEY_VOICE_TIME;
              vGu8BeepTimerFlag=1;

              vGu8DoubleKeySec=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)
{
    if(0==Gu8LedStatus_P1_4)
    {
        LedClose_P1_4();
    }
    else
    {
        LedOpen_P1_4();
    }

    if(0==Gu8LedStatus_P1_5)
    {
        LedClose_P1_5();
    }
    else
    {
        LedOpen_P1_5();
    }

}
```

```
void BeepOpen(void)
{
    P3_4=0;
}

void BeepClose(void)
{
    P3_4=1;
}

void LedOpen_P1_4(void)
{
    P1_4=0;
}

void LedClose_P1_4(void)
{
    P1_4=1;
}

void LedOpen_P1_5(void)
{
    P1_5=0;
}

void LedClose_P1_5(void)
{
    P1_5=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();
            }

        }
    }
}
```