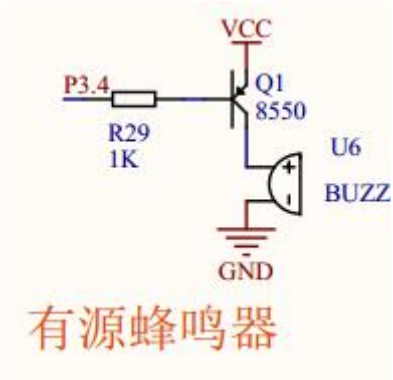


第一百零二节： 两个“任意行输入” 矩阵按键的“有序” 组合触发。

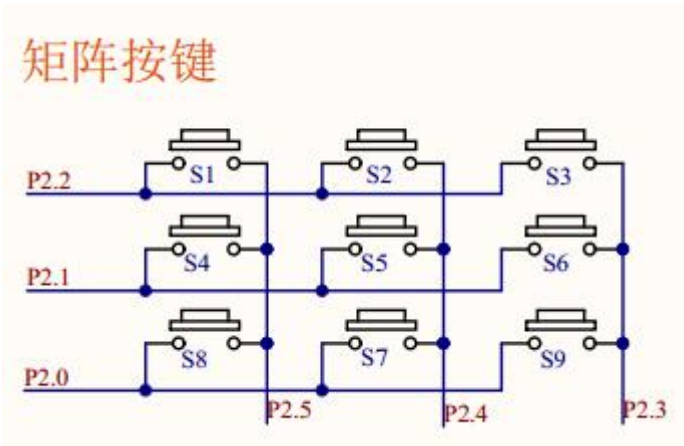
【102.1 “异行输入” “同行输入” “有序”。】



上图 102.1.1 有源蜂鸣器电路



上图 102.1.2 LED 电路



上图 102.1.3 3*3 矩阵按键的电路

“任意行输入”是指能兼容“异行输入”与“同行输入”这两种按键状态。

何谓“异行输入”何谓“同行输入”？如上图矩阵按键，P2.2, P2.1, P2.0 是输入行，P2.5, P2.4, P2.3 是输出列。以 S1 按键为例，很明显，S2 和 S3 都是属于 S1 的“同行输入”，都是属于 P2.2 的输入行。除了 S2 和 S3 以外，其它所有的按键都是 S1 的“异行输入”，比如 S5 按键就是 S1 的“异行输入”，因为 S1 是属于 P2.2 的输入行，而 S5 是属于 P2.1 的输入行。

何谓“有序”组合触发？就是两个按键的触发必须遵守“先后顺序”才能构成“组合触发”。比如，像电脑的复制快捷键（Ctrl+C），你必须先按住 Ctrl 再按住 C 此时“复制快捷键”才有效，如果你先按住 C 再按住 Ctrl 此时“复制快捷键”无效。

“异行输入”与“同行输入”，相比之下，“同行输入”更难更有代表性，如果把“同行输入”的程序写出来了，那么完全按“同行输入”的思路，就可以把“异行输入”的程序写出来。因此，只要把“同行输入”的程序写出来了，也就意味着“任意行输入”的程序也就实现了。本节以 S1 和 S2 的“同行输入”按键为例，S1 是主键，类似复制快捷键的 Ctrl 键；S2 是从键，类似复制快捷键的 C 键。要触发组合键（S1+S2），必须先按住 S1 再按 S2 才有效。功能如下：（1）S1 每单击一次，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）如果先按住 S1 再按 S2，就认为构造了“有序”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(void);
void LedClose(void);

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

sbit P3_4=P3^4;
sbit P1_4=P1^4;

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;
volatile unsigned char vGu8SingleKeySec=0;
volatile unsigned char vGu8DoubleKeySec=0;
```

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

/* 注释一：

```
* 两个“任意行输入”矩阵按键“有序”触发的两个最关键地方：
* （1）当 S1 按键被按下单击触发之后，“马上更新输出列的信号状态”，然后切换到后面的步骤。
* （2）在后面的步骤里，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。
* （3）按照这个模板，只需“更改不同的列输出，判断不同的行输入”，就可以实现“任意行输入”
* 矩阵按键的“有序”组合触发。
*/
```

```
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;
            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，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++;    //切换到下一步步骤
            }
            else if(1==Su8ColumnRecord)
            {
                vGu8SingleKeySec=2;
            }
            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;
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，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，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;
}

}

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

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

            if(0==Gu8LedStatus)
            {
                Gu8LedStatus=1;
                LedOpen();
            }
            else
            {
                Gu8LedStatus=0;
                LedClose();
            }

            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)
    {
        LedClose();
    }
    else
    {
        LedOpen();
    }
}

void BeepOpen(void)
{
    P3_4=0;
}

void BeepClose(void)
{
    P3_4=1;
}

void LedOpen(void)
{
    P1_4=0;
}

void LedClose(void)
{
    P1_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();
            }

        }
    }
}
```