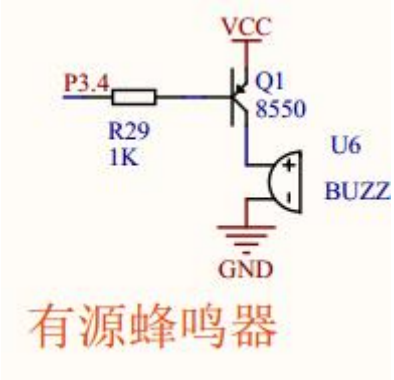
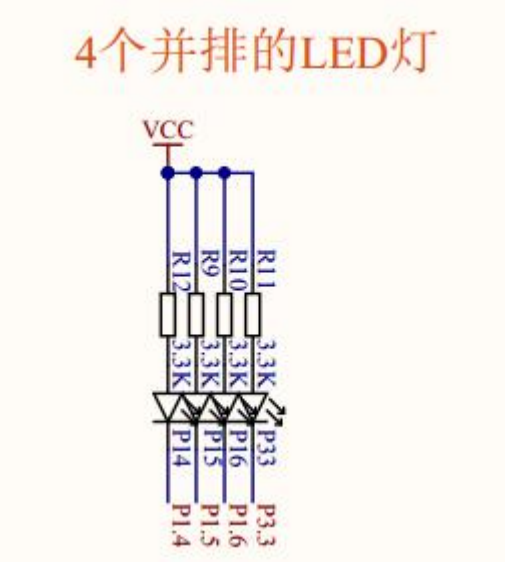
第一百零四节： 矩阵按键“一键两用”的短按与长按。

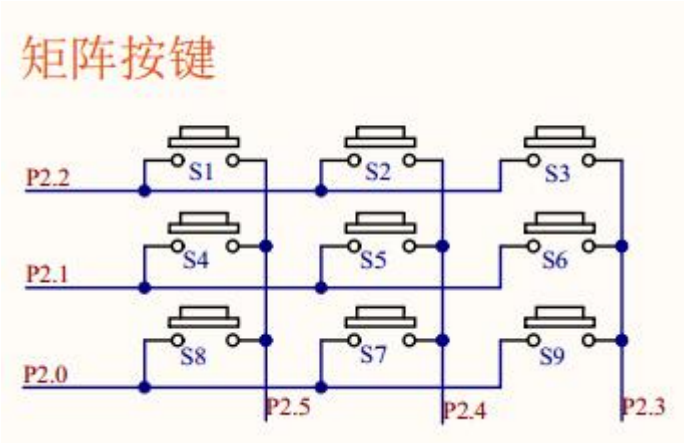
【104.1　“一键两用”的短按与长按。】



上图 104.1.1　有源蜂鸣器电路



上图 104.1.2　LED 电路



上图 104.1.3　3*3 矩阵按键的电路

矩阵按键与前面章节独立按键的“短按与长按”的处理思路是一样的，本节讲矩阵按键的“短按与长按”，也算是重温之前章节讲的内容。“短按与长按”的原理是依赖“按键按下的时间长度”来区分识别。“短按”是指从按下的“下降沿”到松手的“上升沿”时间，“长按”是指从按下的“下降沿”到一直按住不松手的“低电平持续时间”。本节的例程功能如下：（1）S1 每“短按”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）S1 每“长按”一次，蜂鸣器发出“嘀”的一声。代码如下：

```
#include "REG52.H"

#define KEY_VOICE_TIME   50

#define KEY_SHORT_TIME  20    //按键的“短按”兼“滤波”的“稳定时间”
#define KEY_LONG_TIME  400    //按键的“长按”兼“滤波”的“稳定时间”

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 VoiceScan(void);
void KeyScan(void);
void KeyTask(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_P1_4=0;
volatile unsigned char vGu8KeySec=0;  //短按与长按共用一个全局变量 vGu8KeySec 来传递按键信息

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

/* 注释一：
*  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
*  “短按”与“长按”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。
*/

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;

   static unsigned char Su8KeyShortFlag_S1=0;  //S1 按键专属的“短按”触发标志

   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;

           if(1==Su8KeyShortFlag_S1)  //松手的时候，如果“短按”标志有效就触发一次“短按”
           {
                Su8KeyShortFlag_S1=0;     //先清零“短按”标志避免一直触发。
                vGu8KeySec=1;    //触发 S1 的“短按”
            }

           Su8ColumnRecord++;
           if(Su8ColumnRecord>=3)
           {
              Su8ColumnRecord=0;
           }
        }
        else if(0==Su8KeyLock)
        {
            //以下第 1 行，直接把 S1 按键单独扣出来，用“&&0==Su8ColumnRecord”作为筛选条件
            if(0==ROW_INPUT1&&1==ROW_INPUT2&&1==ROW_INPUT3&&0==Su8ColumnRecord)
```

```
        {
            Su16KeyCnt++;
            if(Su16KeyCnt>=KEY_SHORT_TIME) //“短按”兼“滤波”的“稳定时间”
            {
                //注意，这里不能“自锁”。后面“长按”触发的时候才“自锁”。
                Su8KeyShortFlag_S1=1;    //S1 的“短按”标志有效，待松手时触发。
           }

           if(Su16KeyCnt>=KEY_LONG_TIME) //“长按”兼“滤波”的“稳定时间”
           {
                Su8KeyLock=1;       //此时“长按”触发才“自锁”
                Su8KeyShortFlag_S1=0;  //既然此时“长按”有效，那么就要废除潜在的“短按”。
                vGu8KeySec=21; //触发 S1 的“长按”
           }

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

                //既然 S1 按键已经被上面几行代码单独扣出来，这里就直接从 S2 按键开始判断
                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;
                    }
                }
            }

        }
        break;

   }

}

void KeyTask(void)
{
    if(0==vGu8KeySec)
    {
```

```
        return;
    }

    switch(vGu8KeySec)
    {
       case 1:     //S1 按键的“短按”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();
            }

            vGu8KeySec=0;
            break;

        //以下 S1 按键的“长按”直接选择 case 21 的“21”，是为了不占用前排其它按键的编号。
       case 21:     //S1 按键的“长按”任务，蜂鸣器发出“嘀”一声
            vGu8BeepTimerFlag=0;
            vGu16BeepTimerCnt=KEY_VOICE_TIME;  //蜂鸣器发出“嘀”一声
            vGu8BeepTimerFlag=1;

            vGu8KeySec=0;
            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)
{
    if(0==Gu8LedStatus_P1_4)
    {
        LedClose_P1_4();
    }
    else
    {
        LedOpen_P1_4();
    }

}

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 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();
               }

          }
      }
}
```