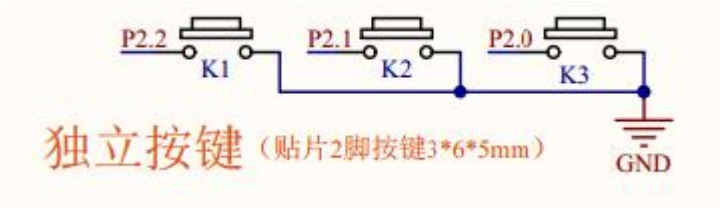
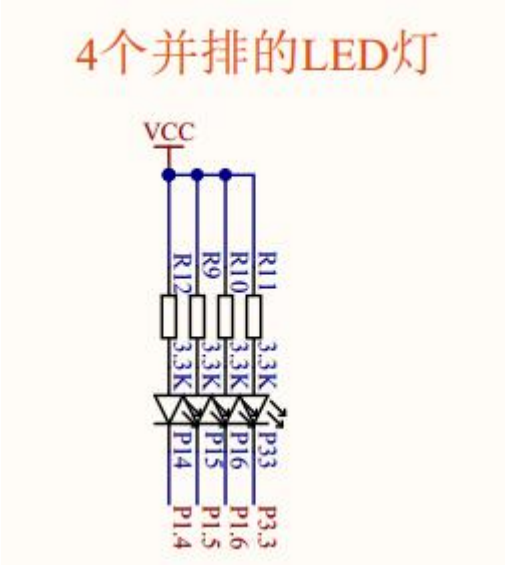
第九十六节： 独立按键“一键两用”的短按与长按。

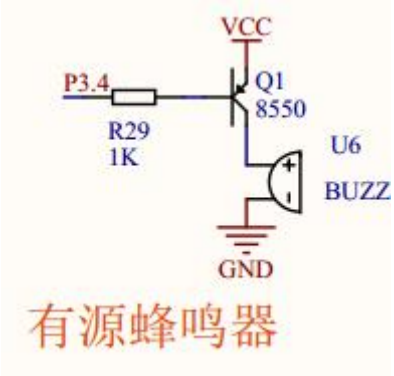
【96.1 “一键两用”的短按与长按。】



上图 96.1.1 独立按键电路



上图 96.1.2 LED 电路



上图 96.1.3 有源蜂鸣器电路

某些项目，当外部按键的资源比较少的时候，一个按键也可以“一键多用”。“一键多用”有很多种玩法，比如，谍战片的无线电通信，依赖一个按键的“不同敲击频率”就可以发送内容丰富的情报。本节“一

键两用”也是属于“一键多用”的众多玩法之一。“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）K1 每“短按”一次（25ms），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）K1 每“长按”一次（500ms），蜂鸣器发出“嘀”的一声。代码如下：

```
#include "REG52.H"

#define KEY_VOICE_TIME   50     //按键“长按”触发后发出的声音长度 50ms

#define KEY_SHORT_TIME  25      //按键的“短按”兼“滤波”的“稳定时间”25ms
#define KEY_LONG_TIME  500      //按键的“长按”兼“滤波”的“稳定时间”500ms

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);    //按键识别的驱动函数，放在定时中断里
void SingleKeyTask(void);   //单击按键任务函数，放在主函数内

sbit P3_4=P3^4;       //蜂鸣器
sbit P1_4=P1^4;       //LED

sbit KEY_INPUT1=P2^2;  //K1 按键识别的输入口。

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

unsigned char Gu8LedStatus=0; //记录 LED 灯的状态，0 代表灭，1 代表亮

volatile unsigned char vGu8SingleKeySec=0;  //单击按键的触发序号

void main()
{
    SystemInitial();
    Delay(10000);
    PeripheralInitial();
```

```
    while(1)
    {
       SingleKeyTask();       //单击按键任务函数
    }
}

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)
{
    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();
               }

          }
      }
}
```

```
/* 注释一：
* “长按”与“短按”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 和去抖动延时计数器 Su16KeyCnt1
*         一直被清零。此时属于按键“松手时间”，因此同时检测“短按”标志 Su8KeyShortFlag
*         是否有效，如果有效就触发一次“短按”。
* 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 和 KEY_LONG_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“短按”阀值 KEY_SHORT_TIME，马上把“短按”标志
*         Su8KeyShortFlag 置 1，如果此时还没有松手，直到发现按下的时间超过“长按”阀值
*         KEY_LONG_TIME 时，先把“短按”标志 ucShortTouchFlag1 清零，然后触发“长按”，同时，为
*         了防止按住按键不松手后一直触发，要及时把 Su8KeyLock1 置 1“自锁”。
* 第四步：等 K1 按键松手后，自锁标志 Su8KeyLock1 及时清零，为下一次自锁做准备，同时，也检测
*         “短按”标志 Su8KeyShortFlag 是否有效，如果有效就触发一次“短按”。
*/

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 char Su8KeyShortFlag=0;  //按键“短按”触发的标志

    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。
      if(1==Su8KeyShortFlag)  //松手的时候，如果“短按”标志有效就触发一次“短按”
      {
          Su8KeyShortFlag=0;     //先清零“短按”标志避免一直触发。
          vGu8SingleKeySec=1;    //触发 K1 的“短按”
      }
   }
   else if(0==Su8KeyLock1)//单个按键 K1 被按下
   {
      Su16KeyCnt1++; //累加定时中断次数

      if(Su16KeyCnt1>=KEY_SHORT_TIME) //“短按”兼“滤波”的“稳定时间”KEY_SHORT_TIME
      {
            //注意，这里不能“自锁”。后面“长按”触发的时候才“自锁”。
            Su8KeyShortFlag=1;    //K1 的“短按”标志有效，待松手时触发。
      }
```

```
      if(Su16KeyCnt1>=KEY_LONG_TIME) //“长按”兼“滤波”的“稳定时间”KEY_LONG_TIME
      {
            Su8KeyLock1=1;      //此时“长按”触发才“自锁”
            Su8KeyShortFlag=0;  //既然此时“长按”有效，那么就要废除潜在的“短按”。
            vGu8SingleKeySec=2; //触发 K1 的“长按”
      }
   }
}

void SingleKeyTask(void)    //单击按键任务函数，放在主函数内
{
    if(0==vGu8SingleKeySec)
    {
        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(vGu8SingleKeySec) //根据不同的按键触发序号执行对应的代码
    {
       case 1:     //K1“短按”触发的任务
            if(0==Gu8LedStatus)
            {
                Gu8LedStatus=1;
                LedOpen();    //LED 亮
            }
            else
            {
                Gu8LedStatus=0;
                LedClose();    //LED 灭
            }
            vGu8SingleKeySec=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:     //K1“长按”触发的任务
            vGu8BeepTimerFlag=0;
            vGu16BeepTimerCnt=KEY_VOICE_TIME;  //触发一次“长按”后，发出“嘀”一声
            vGu8BeepTimerFlag=1;
            vGu8SingleKeySec=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;

    }
}
```