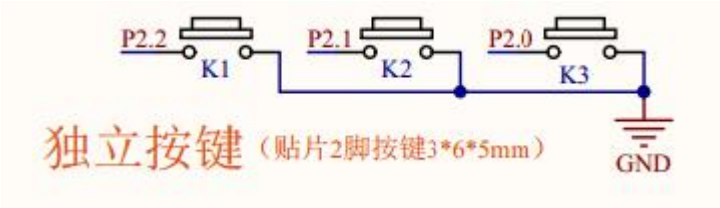
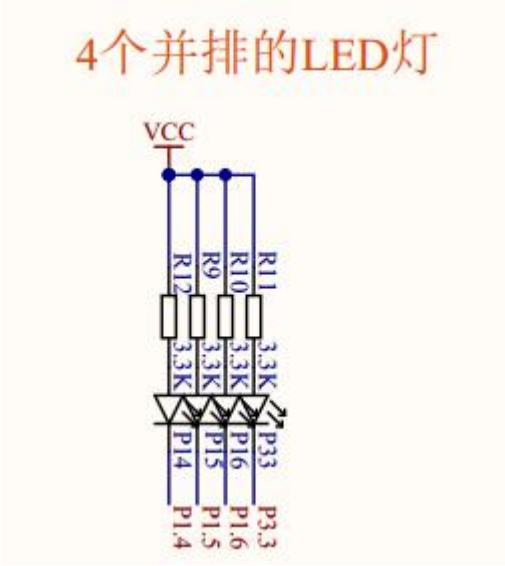
第九十四节： 两个独立按键构成的组合按键。

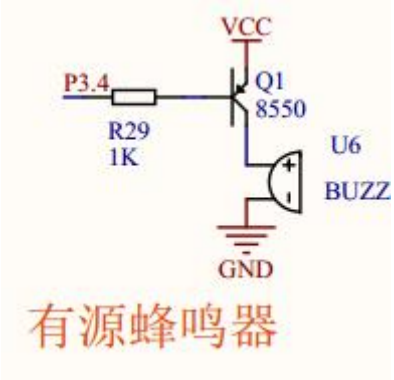
【94.1 组合按键。】



上图 94.1.1 独立按键电路



上图 94.1.2 LED 电路



上图 94.1.3 有源蜂鸣器电路

组合按键的触发，是指两个按键同时按下时的“非单击”触发。一次组合按键的产生，必然包含了三类按键的触发。比如，K1 与 K2 两个独立按键，当它们产生一次组合按键的操作时，就包含了三类触发：K1 单

击触发，K2 单击触发，K1 与 K2 的组合触发。这三类触发可以看作是底层的按键驱动程序，在按键应用层的任务函数 SingleKeyTask 和 CombinationKeyTask 中，可以根据项目的实际需要进行响应。本节程序例程要实现的功能是：（1）K1 单击让 LED 变成“亮”的状态。（2）K2 单击让 LED 变成“灭”的状态。（3）K1 与 K2 的组合按键触发让蜂鸣器发出“嘀”的一声。代码如下：

```
#include "REG52.H"

#define KEY_VOICE_TIME   50     //组合按键触发后发出的声音长度
#define KEY_FILTER_TIME  25     //按键滤波的“稳定时间”25ms

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
void CombinationKeyTask(void);   //组合按键任务函数，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。
sbit KEY_INPUT2=P2^1;  //K2 按键识别的输入口。

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

volatile unsigned char vGu8SingleKeySec=0;  //单击按键的触发序号
volatile unsigned char vGu8CombinationKeySec=0;  //组合按键的触发序号

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

```
        CombinationKeyTask();  //组合按键任务函数
        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)
{
    LedClose();  //初始化关闭 LED
}

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：
* 组合按键扫描的详细过程：
* 第一步：平时只要 K1 与 K2 两个按键中有一个没有被按下时，按键的自锁标志,去抖动延时计数器
* 一直被清零。
* 第二步：一旦两个按键都处于被按下的状态，去抖动延时计数器开始在定时中断函数里累加，在还没
*         累加到阀值 KEY_FILTER_TIME 时，如果在这期间由于受外界干扰或者按键抖动，而使其中一个
```

```
*          IO 口突然瞬间触发成高电平，这个时候马上把延时计数器 Su16CombinationKeyTimeCnt
*          清零了,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：如果两个按键按下的时间超过了阀值 KEY_FILTER_TIME，马上把自锁标志 Su8CombinationKeyLock
*          置 1，防止按住两个按键不松手后一直触发。并把按键编号 vGu8CombinationKeySec 赋值，
*          触发一次组合按键。
* 第四步：等其中一个按键松开后，自锁标志 Su8CombinationKeyLock 及时清零，为下一次自锁做准备。
*/

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 Su8KeyLock2;
   static unsigned int  Su16KeyCnt2;

   static unsigned char Su8CombinationKeyLock;  //组合按键的自锁
   static unsigned int  Su16CombinationKeyCnt;  //组合按键的计时器


   //K1 按键与 K2 按键的组合触发
   if(0!=KEY_INPUT1||0!=KEY_INPUT2)//两个按键只要有一个按键没有按下，处于“非组合按键”的状态。
   {
      Su8CombinationKeyLock=0; //组合按键解锁
      Su16CombinationKeyCnt=0;  //组合按键去抖动延时计数器清零，此行非常巧妙，是全场的亮点。
   }
   else if(0==Su8CombinationKeyLock)//两个按键被同时按下，且是第一次被按下。此行请看专题分析。
   {
      Su16CombinationKeyCnt++; //累加定时中断次数
      if(Su16CombinationKeyCnt>=KEY_FILTER_TIME) //滤波的“稳定时间”KEY_FILTER_TIME。
      {
         Su8CombinationKeyLock=1;  //组合按键的自锁,避免一直触发
         vGu8CombinationKeySec=1;   //触发 K1 与 K2 的组合键操作
      }
   }

   //K1 按键的单击
   if(0!=KEY_INPUT1)
   {
      Su8KeyLock1=0;
      Su16KeyCnt1=0;
   }
   else if(0==Su8KeyLock1)
   {
      Su16KeyCnt1++;
```

```
      if(Su16KeyCnt1>=KEY_FILTER_TIME)
      {
         Su8KeyLock1=1;
         vGu8SingleKeySec=1;    //触发 K1 的单击键
      }
   }

   //K2 按键的单击
   if(0!=KEY_INPUT2)
   {
      Su8KeyLock2=0;
      Su16KeyCnt2=0;
   }
   else if(0==Su8KeyLock2)
   {
      Su16KeyCnt2++;
      if(Su16KeyCnt2>=KEY_FILTER_TIME)
      {
         Su8KeyLock2=1;
         vGu8SingleKeySec=2;    //触发 K2 的单击键
      }
   }

}

void CombinationKeyTask(void)    //组合按键任务函数，放在主函数内
{
    if(0==vGu8CombinationKeySec)
    {
        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(vGu8CombinationKeySec) //根据不同的按键触发序号执行对应的代码
    {
       case 1:     //K1 与 K2 的组合按键任务

            vGu8BeepTimerFlag=0;
            vGu16BeepTimerCnt=KEY_VOICE_TIME;  //触发一次组合按键后，发出“嘀”一声
            vGu8BeepTimerFlag=1;
            vGu8CombinationKeySec=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;
    }

}
```

```
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单击任务
            LedOpen();    //LED亮

            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:     //K2单击任务
            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;

    }

}
```

【94.2　专题分析：else if(0==Su8CombinationKeyLock)。】

疑问：

```
  if(0!=KEY_INPUT1||0!=KEY_INPUT2)
  {
     Su8CombinationKeyLock=0;
     Su16CombinationKeyCnt=0;
  }
  else if(0==Su8CombinationKeyLock)//两个按键被同时按下，且是第一次被按下。为什么？
  {
     Su16CombinationKeyCnt++;
     if(Su16CombinationKeyCnt>=KEY_FILTER_TIME)
     {
        Su8CombinationKeyLock=1;
        vGu8CombinationKeySec=1;
     }
  }
```

解答：

首先，我们要明白 C 语言的语法中，

```
if(条件 1)
{

}
else if(条件 2)
{

}
```

以上语句是一对组合语句，不能分开来看。当（条件 1）成立的时候，它是绝对不会判断（条件 2）的。当（条件 1）不成立的时候，才会判断（条件 2）。

回到刚才的问题，当程序执行到（条件 2） else if(0==Su8CombinationKeyLock)的时候，就已经默认了（条件 1） if(0!=KEY_INPUT1||0!=KEY_INPUT2)不成立，这个条件不成立，就意味着 0==KEY_INPUT1 和 0==KEY_INPUT2，也就是有两个按键被同时按下，因此，这里的 else if(0==Su8CombinationKeyLock)等效于 else if(0==Su8CombinationKeyLock&&0==KEY_INPUT1&&0==KEY_INPUT2)，而 Su8CombinationKeyLock 是一个自锁标志位，一旦组合按键被触发后，这个标志位会变 1，防止两个按键按住不松手的时候不断触发组合按键。这样，组合按键只能同时按下一次触发一次，任意松开其中一个按键后再同时按下一次两个按键，又触发一次新的组合按键。