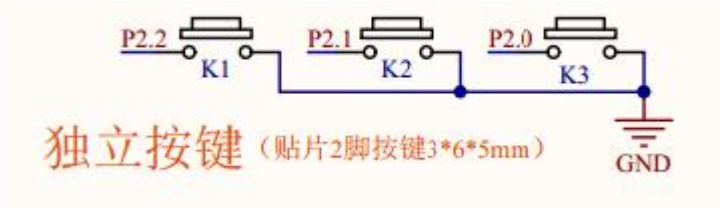
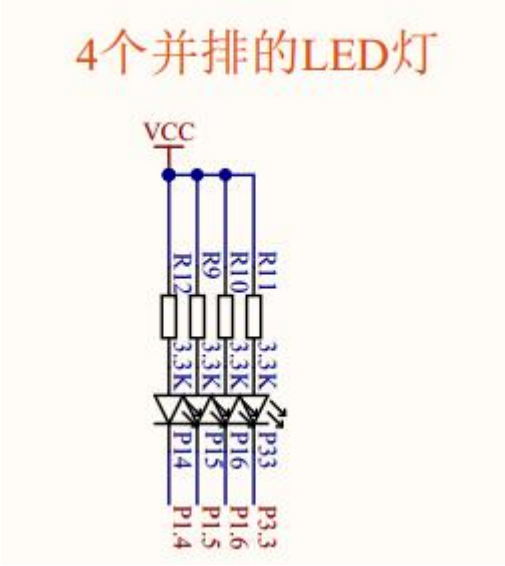
第九十五节： 两个独立按键的“电脑键盘式”组合按键。

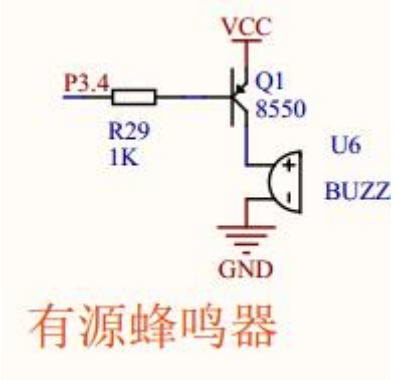
【95.1 “电脑键盘式”组合按键。】



上图 95.1.1 独立按键电路



上图 95.1.2 LED 电路



上图 95.1.3 有源蜂鸣器电路

上一节也讲了由 K1 和 K2 构成的组合按键，但是这种组合按键是普通的组合按键，因为它们的 K1 和 K2 是不分先后顺序的，你先按住 K1 然后再按 K2，或者你先按住 K2 然后再按 K1，效果都一样。本节讲的组合

按键是分先后顺序的，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。本节讲的例程就是要实现这个功能，用 K1 代表 C 这类“字符数字键”，用 K2 代表 Ctrl 这类“辅助按键”，因此，要触发组合键（K2+K1），必须先按住 K2 再按 K1 才有效。本节讲的例程功能如下：（1）K1 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 K2 再按 K1，就认为构造了“电脑键盘式”组合键，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 50ms
#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;

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
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)
{
    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：
* “电脑键盘式”组合按键扫描的详细过程：
* 第一步：K2 与 K1 构成的组合按键触发是融合在 K1 单击按键程序里的，只需稍微更改一下 K1 单击的程序
*         ，就可以兼容到 K2 与 K1 构成的“电脑键盘式”组合按键。平时只要 K1 没有被按下时，按
*         键的自锁标志 Su8KeyLock1 和去抖动延时计数器 Su16KeyCnt1 一直被清零。
* 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_FILTER_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 K1 按键按下的时间超过了阀值 KEY_FILTER_TIME，马上把自锁标志 Su8KeyLock1 置 1，
*         防止按住按键不松手后一直触发，此时才开始判断一次 K2 按键的电平状态，如果 K2 为低电
*         平就认为是组合按键，并给按键编号 vGu8CombinationKeySec 赋值，否则，就认为是 K1 的单击
*         按键，并给按键编号 vGu8SingleKeySec 赋值。
* 第四步：等 K1 按键松开后，自锁标志 Su8KeyLock1 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;


   //K1 的单击，或者 K2 与 K1 构成的“电脑键盘式组合按键”。
   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。
   }
   else if(0==Su8KeyLock1)//单个按键 K1 被按下
   {
      Su16KeyCnt1++; //累加定时中断次数
      if(Su16KeyCnt1>=KEY_FILTER_TIME) //滤波的“稳定时间”KEY_FILTER_TIME。
      {
         if(0==KEY_INPUT2)  //此时才开始判断一次 K2 的电平状态，为低电平则是组合按键。
         {
            Su8KeyLock1=1;
            vGu8CombinationKeySec=1;  //组合按键的触发
         }
         else
```

```
        {
           Su8KeyLock1=1;
           vGu8SingleKeySec=1;    //K1 单击按键的触发
         }
      }
   }
}

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
            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;
    }

}
```