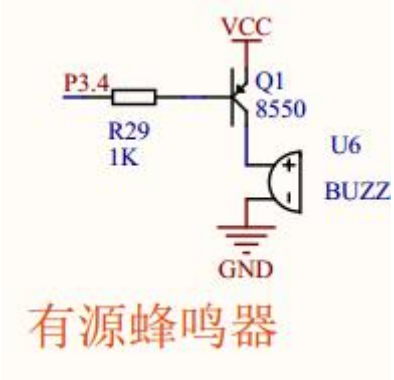
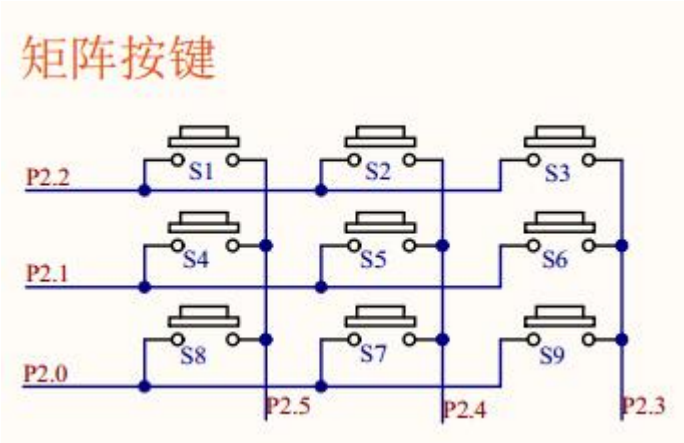
第一百节： “行列扫描式”矩阵按键的单个触发（优化版）。

【100.1 “行列扫描式”矩阵按键。】



上图 100.1.1 有源蜂鸣器电路



上图 100.1.2 3*3 矩阵按键的电路

写程序，凡是出现“重复性、相似性”的代码，都可以加入“循环，判断，数组”这类语句对代码进行压缩优化。上一节讲的矩阵按键，代码是记流水账式的，出现很多“重复性、相似性”的代码，是没有经过优化的“原始版”，本节的目的是对上一节的代码进行优化，让大家从中发现一些技巧。

多说一句，我一直认为，只要单片机容量够，代码多一点少一点并不重要，只要不影响运行效率就行。而且有时候，代码写多一点，可读性非常强，修改起来也非常方便。如果一味的追求压缩代码，就会刻意用很多“循环，判断，数组”等元素，代码虽然紧凑了，但是可分离性，可更改性，可阅读性就没那么强。因此，做项目的时候，某些代码要不要进行压缩，是没有绝对标准的，能因敌而取胜者谓之神。

本节例程实现的功能：9 个矩阵按键，每按下 1 个按键都触发一次蜂鸣器鸣叫。

```
#include "REG52.H"

#define KEY_VOICE_TIME   50

#define KEY_SHORT_TIME  20      //按键去抖动的“滤波”时间
```

```
void T0_time();
void SystemInitial(void) ;
void Delay(unsigned long u32DelayTime) ;
void PeripheralInitial(void) ;

void BeepOpen(void);
void BeepClose(void);

void VoiceScan(void);
void KeyScan(void);
void KeyTask(void);

sbit P3_4=P3^4;       //蜂鸣器

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;



volatile unsigned char vGu8KeySec=0;  //按键的触发序号

void main()
{
    SystemInitial();
    Delay(10000);
    PeripheralInitial();
    while(1)
    {
        KeyTask();       //按键的任务函数
    }
}

/* 注释一：
*  矩阵按键扫描的详细过程：
*  先输出某 1 列低电平，其它 2 列输出高电平，延时等待 2ms 后（等此 3 列输出同步稳定），
*  再分别判断 3 行的输入 IO 口， 如果发现哪一行是低电平，就说明对应的某个按键被触发。
```

```
*  依次循环切换输出的 3 种状态，并且分别判断输入的 3 行，就可以检测完 9 个按键。矩阵按键的
*  去抖动处理方法跟我前面讲的独立按键去抖动方法是一样的，不再重复多讲。
*/

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;  //本节多增加此变量用来切换当前列的输出

   switch(Su8KeyStep)
   {
     case 1:
          if(0==Su8ColumnRecord)  //按键扫描输出第一列低电平
          {
              COLUMN_OUTPUT1=0;
              COLUMN_OUTPUT2=1;
              COLUMN_OUTPUT3=1;
          }
          else if(1==Su8ColumnRecord)  //按键扫描输出第二列低电平
          {
              COLUMN_OUTPUT1=1;
              COLUMN_OUTPUT2=0;
              COLUMN_OUTPUT3=1;
          }
          else     //按键扫描输出第三列低电平
          {
              COLUMN_OUTPUT1=1;
              COLUMN_OUTPUT2=1;
              COLUMN_OUTPUT3=0;
          }
          Su16KeyCnt=0;  //延时计数器清零
          Su8KeyStep++;  //切换到下一个运行步骤
          break;

     case 2:     //延时等待 2ms 后（等此 3 列输出同步稳定）。不是按键的去抖动延时。
          Su16KeyCnt++;
          if(Su16KeyCnt>=2)
          {
             Su16KeyCnt=0;
             Su8KeyStep++;     //切换到下一个运行步骤
          }
```

```
        break;

    case 3:
        if(1==ROW_INPUT1&&1==ROW_INPUT2&&1==ROW_INPUT3)
        {
           Su8KeyStep=1;  //如果没有按键按下，返回到第一个运行步骤重新开始扫描!!!!!!
           Su8KeyLock=0;  //按键自锁标志清零
           Su16KeyCnt=0;  //按键去抖动延时计数器清零，此行非常巧妙
           Su8ColumnRecord++;  //输出下一列
           if(Su8ColumnRecord>=3)
           {
              Su8ColumnRecord=0; //依次输出完第 3 列之后，继续从第 1 列开始输出低电平
           }
        }
        else if(0==Su8KeyLock)  //有按键按下，且是第一次触发
        {
            if(0==ROW_INPUT1&&1==ROW_INPUT2&&1==ROW_INPUT3)
            {
                Su16KeyCnt++;  //去抖动延时计数器
                if(Su16KeyCnt>=KEY_SHORT_TIME)
                {
                    Su16KeyCnt=0;
                    Su8KeyLock=1;//自锁置 1,避免一直触发,只有松开按键,此标志位才会被清零

                    if(0==Su8ColumnRecord)  //第 1 列输出低电平
                    {
                         vGu8KeySec=1;  //触发 1 号键 对应 S1 键
                    }
                    else if(1==Su8ColumnRecord)  //第 2 列输出低电平
                    {
                         vGu8KeySec=2;  //触发 2 号键 对应 S2 键
                    }
                    else if(2==Su8ColumnRecord)  //第 3 列输出低电平
                    {
                         vGu8KeySec=3;  //触发 3 号键 对应 S3 键
                    }
                }

            }
            else if(1==ROW_INPUT1&&0==ROW_INPUT2&&1==ROW_INPUT3)
            {
                Su16KeyCnt++;  //去抖动延时计数器
                if(Su16KeyCnt>=KEY_SHORT_TIME)
                {
```

```
                    Su16KeyCnt=0;
                    Su8KeyLock=1;//自锁置 1,避免一直触发,只有松开按键,此标志位才会被清零

                    if(0==Su8ColumnRecord)  //第 1 列输出低电平
                    {
                        vGu8KeySec=4;  //触发 4 号键 对应 S4 键
                    }
                    else if(1==Su8ColumnRecord)  //第 2 列输出低电平
                    {
                        vGu8KeySec=5;  //触发 5 号键 对应 S5 键
                    }
                    else if(2==Su8ColumnRecord)  //第 3 列输出低电平
                    {
                        vGu8KeySec=6;  //触发 6 号键 对应 S6 键
                    }
                }
            }
            else if(1==ROW_INPUT1&&1==ROW_INPUT2&&0==ROW_INPUT3)
            {
                Su16KeyCnt++;  //去抖动延时计数器
                if(Su16KeyCnt>=KEY_SHORT_TIME)
                {
                    Su16KeyCnt=0;
                    Su8KeyLock=1;//自锁置 1,避免一直触发,只有松开按键,此标志位才会被清零
                    if(0==Su8ColumnRecord)  //第 1 列输出低电平
                    {
                        vGu8KeySec=7;  //触发 7 号键 对应 S7 键
                    }
                    else if(1==Su8ColumnRecord)  //第 2 列输出低电平
                    {
                        vGu8KeySec=8;  //触发 8 号键 对应 S8 键
                    }
                    else if(2==Su8ColumnRecord)  //第 3 列输出低电平
                    {
                        vGu8KeySec=9;  //触发 9 号键 对应 S9 键
                    }
                }
            }

        }
        break;
    }

}
```

```
void KeyTask(void)    //按键任务函数，放在主函数内
{
    if(0==vGu8KeySec)
    {
        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(vGu8KeySec) //根据不同的按键触发序号执行对应的代码
    {
       case 1:     //S1 触发的任务

            vGu8BeepTimerFlag=0;
            vGu16BeepTimerCnt=KEY_VOICE_TIME;  //发出“嘀”一声
            vGu8BeepTimerFlag=1;

            vGu8KeySec=0;  //响应按键服务处理程序后，按键编号必须清零，避免一直触发
            break;

       case 2:     //S2 触发的任务

            vGu8BeepTimerFlag=0;
            vGu16BeepTimerCnt=KEY_VOICE_TIME;  //发出“嘀”一声
            vGu8BeepTimerFlag=1;

            vGu8KeySec=0;  //响应按键服务处理程序后，按键编号必须清零，避免一直触发
            break;

       case 3:     //S3 触发的任务

            vGu8BeepTimerFlag=0;
            vGu16BeepTimerCnt=KEY_VOICE_TIME;  //发出“嘀”一声
            vGu8BeepTimerFlag=1;

            vGu8KeySec=0;  //响应按键服务处理程序后，按键编号必须清零，避免一直触发
            break;

       case 4:     //S4 触发的任务

            vGu8BeepTimerFlag=0;
            vGu16BeepTimerCnt=KEY_VOICE_TIME;  //发出“嘀”一声
            vGu8BeepTimerFlag=1;

            vGu8KeySec=0;  //响应按键服务处理程序后，按键编号必须清零，避免一直触发
```

```
        break;

    case 5:     //S5 触发的任务

        vGu8BeepTimerFlag=0;
        vGu16BeepTimerCnt=KEY_VOICE_TIME;  //发出“嘀”一声
        vGu8BeepTimerFlag=1;

        vGu8KeySec=0;  //响应按键服务处理程序后，按键编号必须清零，避免一直触发
        break;

    case 6:     //S6 触发的任务

        vGu8BeepTimerFlag=0;
        vGu16BeepTimerCnt=KEY_VOICE_TIME;  //发出“嘀”一声
        vGu8BeepTimerFlag=1;

        vGu8KeySec=0;  //响应按键服务处理程序后，按键编号必须清零，避免一直触发
        break;

    case 7:     //S7 触发的任务

        vGu8BeepTimerFlag=0;
        vGu16BeepTimerCnt=KEY_VOICE_TIME;  //发出“嘀”一声
        vGu8BeepTimerFlag=1;

        vGu8KeySec=0;  //响应按键服务处理程序后，按键编号必须清零，避免一直触发
        break;

    case 8:     //S8 触发的任务

        vGu8BeepTimerFlag=0;
        vGu16BeepTimerCnt=KEY_VOICE_TIME;  //发出“嘀”一声
        vGu8BeepTimerFlag=1;

        vGu8KeySec=0;  //响应按键服务处理程序后，按键编号必须清零，避免一直触发
        break;

    case 9:     //S9 触发的任务

        vGu8BeepTimerFlag=0;
        vGu16BeepTimerCnt=KEY_VOICE_TIME;  //发出“嘀”一声
        vGu8BeepTimerFlag=1;
```

```
            vGu8KeySec=0;  //响应按键服务处理程序后，按键编号必须清零，避免一直触发
            break;

    }
}



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)
{

}

void BeepOpen(void)
{
    P3_4=0;
}

void BeepClose(void)
```

```
{
    P3_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();
                }

          }
      }
}
```