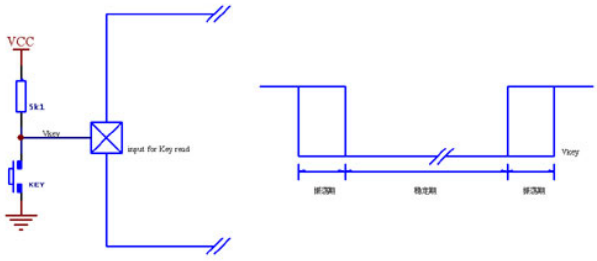
Key 读取和事件处理

按键（轻触开关）作为一种简单实用的人机交换界面，在嵌入式控制系统中随处可见。

Key虽然简单，在软件的处理上，一些有效的方法还是令人愉悦的。

按键特别的机械特性，其触点在Key被按下和释放的初期都有数十ms的机械振荡。同样，反映在I/O口上的电气信号也是不稳定的，如图。



下面的例子是可以放进分时框架的Key的底层驱动和事件驱动程序，时基为0.5ms。底层驱动程序通过计数方式将Key的状态传递给事件驱动程序。

```
//============================================================
//
//                   Program for Key_HMI
//
//
// MCU:   ATmega169@2MHZ
// DATE:  V10  @ 07022008THUAM
//
//
//============================================================

#define uchar  unsigned char
#define uint   unsigned int
#define ulong  unsigned long

#define schar signed char
#define sint  signed int
#define slong signed long

//----------     RAM & BIT define    -----------------

uchar        KEY_BUFF0_R,KEY_BUFF1_R,KEY_BUFF2_R,            // KEY read buffer
             KEY_BUFF3_R,KEY_BUFF4_R,KEY_BUFF5_R,
             KEY_BUFF6_R,KEY_BUFF7_R,KEY_BUFF8_R;


//----------     I/O PIN define     ------------------
                                                  // KEY input I/O
#define   I_KEY_0_PORT    PINE
#define   I_KEY_0_B       2
#define   I_KEY_1_PORT    PINE
#define   I_KEY_1_B       3
#define   I_KEY_2_PORT    PINE
#define   I_KEY_2_B       4
#define   I_KEY_3_PORT    PINE
#define   I_KEY_3_B       5
#define   I_KEY_4_PORT    PINE
#define   I_KEY_4_B       6
#define   I_KEY_5_PORT    PINE
#define   I_KEY_5_B       7
#define   I_KEY_6_PORT    PINB
#define   I_KEY_6_B       0
#define   I_KEY_7_PORT    PINF
#define   I_KEY_7_B       7
#define   I_KEY_8_PORT    PINF
#define   I_KEY_8_B       6


//----------     const    define    ------------------

#define   KeyConti_D       200
#define   KEY_D            5


//---------------  key_pro / 按键底层驱动   -----------------

void key_pro( void )
  {
                                                        // time_base: 16ms
   if( (T_BASE0_R.BYTE & 0B00011100) == 0B00001000 )
     {
                                                        // KEY read
       if( (I_KEY_0_PORT & (1<<I_KEY_0_B)))  KEY_BUFF0_R = 0;
       if( (I_KEY_1_PORT & (1<<I_KEY_1_B)))  KEY_BUFF1_R = 0;
       if( (I_KEY_2_PORT & (1<<I_KEY_2_B)))  KEY_BUFF2_R = 0;
       if( (I_KEY_3_PORT & (1<<I_KEY_3_B)))  KEY_BUFF3_R = 0;
       if( (I_KEY_4_PORT & (1<<I_KEY_4_B)))  KEY_BUFF4_R = 0;
       if( (I_KEY_5_PORT & (1<<I_KEY_5_B)))  KEY_BUFF5_R = 0;
       if( (I_KEY_6_PORT & (1<<I_KEY_6_B)))  KEY_BUFF6_R = 0;
       if( (I_KEY_7_PORT & (1<<I_KEY_7_B)))  KEY_BUFF7_R = 0;
```

```
       if( (I_KEY_8_PORT & (1<<I_KEY_8_B)))  KEY_BUFF8_R = 0;


       if( KEY_BUFF0_R != 255 )   KEY_BUFF0_R++;
       if( KEY_BUFF1_R != 255 )   KEY_BUFF1_R++;
       if( KEY_BUFF2_R != 255 )   KEY_BUFF2_R++;
       if( KEY_BUFF3_R != 255 )   KEY_BUFF3_R++;
       if( KEY_BUFF4_R != 255 )   KEY_BUFF4_R++;
       if( KEY_BUFF5_R != 255 )   KEY_BUFF5_R++;
       if( KEY_BUFF6_R != 255 )   KEY_BUFF6_R++;
       if( KEY_BUFF7_R != 255 )   KEY_BUFF7_R++;
       if( KEY_BUFF8_R != 255 )   KEY_BUFF8_R++;

     }
   }

//---------------  key_process_pro / 按键事件驱动   -----------------

void key_process_pro( void )
  {
                                                                    // time_base: 16ms
    if( (T_BASE0_R.BYTE & 0B00011100) == 0B00001100 )
     {

       if( KEY_BUFF0_R == KEY_D )
         {
                                                                    // Key0 press
         }

       if( KEY_BUFF1_R == KEY_D )
         {
                                                                    // Key1 press
         }

       if( KEY_BUFF2_R == KeyConti_D )
         {
                                                                    // Key2 long_time_press
         }

       if( KEY_BUFF3_R >  KeyConti_D )
         {
                                                                    // Key3 long_time_press
         }
     }
  }
```

Note:

---

1) 文中提及的名称和商标为相关的所有者所有

2) 本文由 SPM专用编程器 提供赞助，相关源码可到 SPM专用编程器 下载。