





















蓝桥杯按键考点



Author: 左岚

Time: 2024.4.7

蓝桥杯按键考点

-  基础按键底层
-  三行代码消抖
 -  调用
-  模式切换
-  参数设置
-  参数保存
-  特定使能
-  按键长短按
 -  按键方面
 -  定时器方面
-  按下跳转 松手返回
-  密码门输入
 -  按键方面
 -  显示部分
-  双按键（十四届国赛新增考点）
 -  底层代码
 -  按键方面
-  按键多击
 -  按键方面
 -  定时器方面

基础按键底层



```
1 unsigned char Key_Read()  
2 {  
3     unsigned char temp = 0;  
4     ET0 = 0; // 串口使用屏蔽  
5     P44 = 0;    P42 = 1;    P35 = 1;    P34 = 1;
```

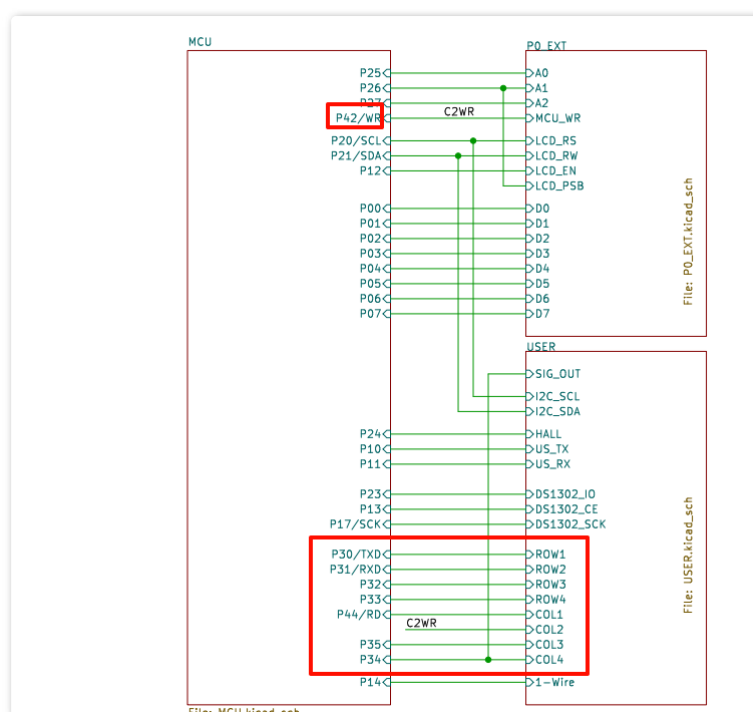
```

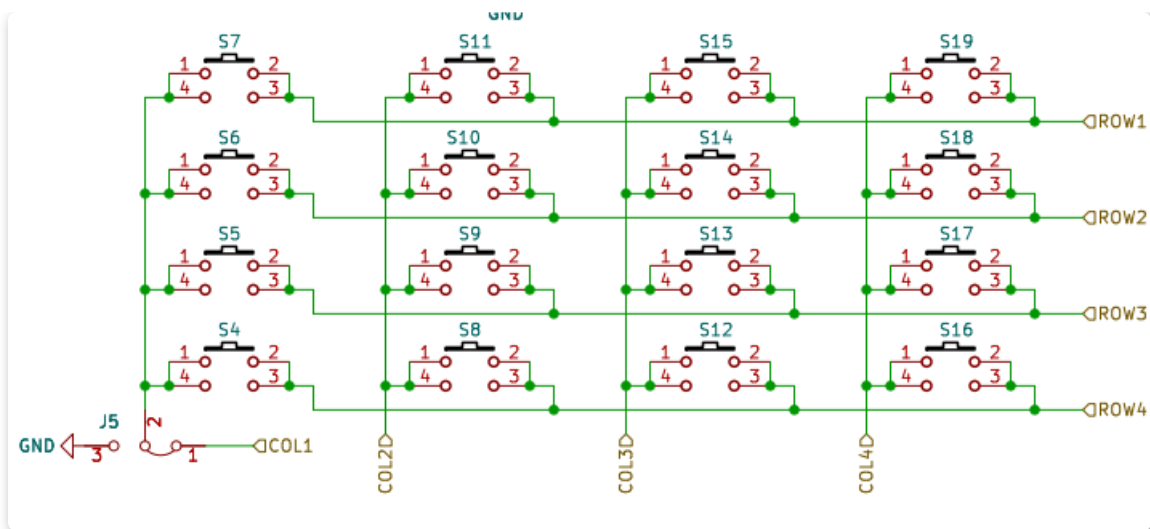
6     if (P33 == 0)    temp = 4;
7     if (P32 == 0)    temp = 5;
8     if (P31 == 0)    temp = 6;
9     if (P30 == 0)    temp = 7;
10    P44 = 1;    P42 = 0;    P35 = 1;    P34 = 1;
11    if (P33 == 0)    temp = 8;
12    if (P32 == 0)    temp = 9;
13    if (P31 == 0)    temp = 10;
14    if (P30 == 0)    temp = 11;
15    P44 = 1;    P42 = 1;    P35 = 0;    P34 = 1;
16    if (P33 == 0)    temp = 12;
17    if (P32 == 0)    temp = 13;
18    if (P31 == 0)    temp = 14;
19    if (P30 == 0)    temp = 15;
20    P44 = 1;    P42 = 1;    P35 = 1;    P34 = 0;
21    if (P33 == 0)    temp = 16;
22    if (P32 == 0)    temp = 17;
23    if (P31 == 0)    temp = 18;
24    if (P30 == 0)    temp = 19;
25    ET0 = 1; //串口使用屏蔽
26    P3 = 0xff;
27    return temp;
28 }

```



这个是最基础的底层，通过下面的原理图看出来的（这里使用最新的十五届原理图），这里就不过多赘述了，通过检测低电平来进行判断





三行代码消抖

```

1 Key_Val = Key_Read();
2 Key_Down = Key_Val & (Key_Old ^ Key_Val);
3 Key_Up = ~Key_Val & (Key_Old ^ Key_Val);
4 Key_Old = Key_Val;

```



这个代码其实在西风的文档里面单独列了一下，我这里将其放在一起来进行描述

瞬时值获取-> 获取按下的值，获取抬起的值->更新一下值

Down和Up都是瞬时值，Old比Val延后10ms，基于这个特性我们可以进行书写代码

这里以S4 (0100) 为例，来进行说明

键盘状态	Key_Old	Key_Val	Key_Old^Key_Val	Key_Down	Key_Up
未按下	0000	0000	$0000 \wedge 0000 = 0000$	0000	0000
按下过程中 (10ms)	0000	0100	$0000 \wedge 0100 = 0100$	$0100 \& 0100 = 0100$	$1011 \& 0100 = 0000$
按下稳定 (10ms 后)	0100	0100	$0100 \wedge 0100 = 0000$	$0100 \& 0000 = 0000$	$1011 \& 0000 = 0000$
抬起过程 (10ms)	0100	0000	$0100 \wedge 0000 = 0100$	$0000 \& 0100 = 0000$	$1111 \& 0100 = 0100$



我们可以从这里看出，当我们按下的时候，那一瞬间Down会是4，其余时候都是0；同理Up也是一样，当我们抬起的瞬间Up会是4，其余时候是0

调用

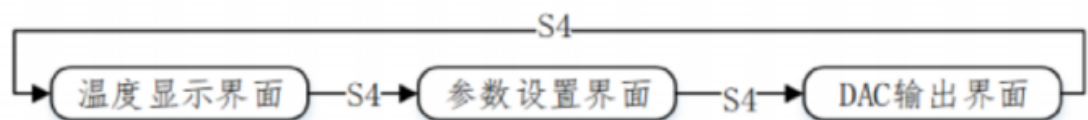


那么我们应该如何进行调用呢，其实很简单，如下所示

```
1 if (Key_Down == 4)
2     //执行按下4的函数
```

模式切换

- S4: 定义为“界面”按键，按下 S4 按键，切换温度显示界面、参数设置界面和 DAC 输出界面，按键 S4 切换模式如图 5 所示：



```
1 //以这个为例，我们三个界面
2 //首先需要定义一个界面显示的一个变量
3 unsigned char Seg_show_mode; //0 温度显示页面 1 参数设置页面 2 DAC
  输出页面
4 //然后开始我们的键盘函数
5 void Key_Proc()
6 {
7     static uchar Key_Val, Key_Down, Key_Up, Key_Old;
8     if (Key_Slow_Down)
9         return;
10    Key_Slow_Down = 1;
11
12    Key_Val = Key_Read();
13    Key_Down = Key_Val & (Key_Old ^ Key_Val);
14    Key_Up = ~Key_Val & (Key_Old ^ Key_Val);
15    Key_Old = Key_Val;
16    if (Key_Down == 4)
17        //0%3=0 1%3=1 2%3=2 3%3=0
18        Seg_show_mode = ( + Seg_show_mode) % 3;
19 }
```



这里我们直接进行取余，就可以保证这个值在0-2之间循环

参数设置



- S8: 定义为“加”按键，参数界面下，按下温度参数值加1。
- S9: 定义为“减”按键，参数界面下，按下温度参数值减1；时间回显子界面下，长按S9超过2秒后松开，清除所有已记录的数据，触发次数重置为0。



这里我们不说他那个长按的问题，我们就看单纯的土参数，调用如下所示

```
1 //以这个为例，我们有一个参数
2 //首先需要定义参数
3 unsigned char dat; //需要调整的参数
4 //然后开始我们的键盘函数
5 void Key_Proc()
6 {
7     static uchar Key_Val, Key_Down, Key_Up, Key_Old;
8     if (Key_Slow_Down)
9         return;
10    Key_Slow_Down = 1;
11
12    Key_Val = Key_Read();
13    Key_Down = Key_Val & (Key_Old ^ Key_Val);
14    Key_Up = ~Key_Val & (Key_Old ^ Key_Val);
15    Key_Old = Key_Val;
16    //一定注意上下限
17    if (Key_Down == 8)
18        //假定dat的取值的上限是100
19        dat = ( ++ dat > 100 ) ? 100 : dat;
20    if (Key_Down == 9)
21        //假定dat的取值下限是10
22        dat = ( -- dat < 10 ) ? 10 : dat;
23 }
```



这里其实要注意的点就是，一定要看清楚上下限，否则会导致超出扣分

参数保存



参数保存也是一个考点，主要是修改的数据是否实时生效，上面的参数设置就是实时生效，而这里的参数保存就是延迟生效，修改参数的时候，原本的参数不产生变化

通过 S4 按键，从参数设置界面退出，进入数据显示界面时，需要进行必要的参数合理性检查 ($T_{\text{MAX}} \geq T_{\text{MIN}}$)；若设置的参数合理，参数生效，进入数据界面；反之，自动恢复进入参数设置界面的有效参数，进入数据界面。

```
1 //以这个为例，我们有一个参数
2 //首先需要定义参数
3 unsigned char Tmax_Control, Tmin_Control; //参数设置值
4 unsigned char Tmax, Tmin; //参数保存值
5 //然后开始我们的键盘函数
6 void Key_Proc()
7 {
8     static uchar Key_Val, Key_Down, Key_Up, Key_Old;
9     if (Key_Slow_Down)
10         return;
11     Key_Slow_Down = 1;
12
13     Key_Val = Key_Read();
14     Key_Down = Key_Val & (Key_Old ^ Key_Val);
15     Key_Up = ~Key_Val & (Key_Old ^ Key_Val);
16     Key_Old = Key_Val;
17     //一定注意上下限
18     if (Key_Down == 8)
19         Tmax_Control ++ ;
20     if (Key_Down == 9)
21         Tmin_Control ++ ;
22     //按下S4确认输入的数据
23     if (Key_Down == 4)
24     {
25         //当我们满足条件
26         if (Tmax_Control == Tmin_Control)
27         {
28             Tmax = Tmax_Control;
29             Tmin = Tmin_Control;
30         }
```

```

31         //不满足条件，值不进行修改，并且控制值复原为原本的值
32         else
33         {
34             Tmax_Control = Tmax;
35             Tmin_Control = Tmin;
36         }
37     }
38 }

```

特定使能



这个就是在特定界面下生效的代码，很简单

按键 S6 和按键 S7 的加、减功能仅在参数设置界面有效。

```

1 //以这个为例，我们有一个参数
2 //首先需要定义参数
3 unsigned char dat; //需要调整的参数
4 unsigned char Seg_show_mode; //0 温度显示页面 1 参数设置页面 2 DAC
   输出页面
5 //然后开始我们的键盘函数
6 void Key_Proc()
7 {
8     static uchar Key_Val, Key_Down, Key_Up, Key_Old;
9     if (Key_Slow_Down)
10         return;
11     Key_Slow_Down = 1;
12
13     Key_Val = Key_Read();
14     Key_Down = Key_Val & (Key_Old ^ Key_Val);
15     Key_Up = ~Key_Val & (Key_Old ^ Key_Val);
16     Key_Old = Key_Val;
17     //当前界面是参数设置界面的时候
18     if (Seg_show_mode == 1)
19     {
20         //一定注意上下限
21         if (Key_Down == 6)
22             //假定dat的取值的上限是100
23             dat = ( ++ dat > 100 ) ? 100 : dat;
24         if (Key_Down == 7)
25             //假定dat的取值下限是10
26             dat = ( -- dat < 10 ) ? 10 : dat;

```

```
27     }  
28 }
```

按键长短按



这里我没有找例题，但是这里实现的就是

按键长按为一个效果，按键短按为另一个效果

这里也分为了两个情况

长按S4 2S，直接执行结果

长按S5 2S以上，松手执行结果

按键方面



```
1 //以这个为例  
2 //首先需要定义参数  
3 unsigned int time_2s;  
4 bit Long_press_timing_flag;  
5 //然后开始我们的键盘函数  
6 void Key_Proc()  
7 {  
8     static uchar Key_Val, Key_Down, Key_Up, Key_Old;  
9     if (Key_Slow_Down)  
10         return;  
11     Key_Slow_Down = 1;  
12  
13     Key_Val = Key_Read();  
14     Key_Down = Key_Val & (Key_Old ^ Key_Val);  
15     Key_Up = ~Key_Val & (Key_Old ^ Key_Val);  
16     Key_Old = Key_Val;  
17     //长按S5 2s，执行函数（不需要弹起），短按S4执行另一个函数  
18     if (Key_Down == 5)  
19     {  
20         Long_press_timing_flag = 1;  
21     }  
22     //计时到达  
23     if (time_2s == 2000)  
24     {  
25         //执行长按的函数  
26     }  
27     //S5抬起  
28     if (Key_Up == 5)
```



```

29     {
30         //如果时间没有到达
31         if (time_2s < 2000)
32         {
33             //执行短按函数
34         }
35         time_2s = 0;
36         Long_press_timing_flag = 0;
37     }
38     //长按S4 2s, 执行函数（需要弹起），短按S4执行另一个函数
39     if (Key_Down = 4)
40     {
41         Long_press_timing_flag = 1;
42     }
43     //S4抬起
44     if (Key_Up = 4)
45     {
46         //按键时间超过2s
47         if (time_2s = 2000)
48         {
49             //执行按键长按的函数
50         }
51         //按键时间没有超过2s, 短按
52         else
53         {
54             //执行按键短按的函数
55         }
56         time_2s = 0;
57         Long_press_timing_flag = 0;
58     }
59 }

```

定时器方面



```

1 //当我们按下S4或者S5的时候，开始计时
2 if (Long_press_timing_flag)
3 {
4     if ( + time_2s = 2000)
5     {
6         time_2s = 2001; //锁死时间
7     }
8 }
9 else
10 {
11     time_2s = 0;
12 }

```

按下跳转 松手返回

在“时钟显示”状态下，按下 S4 按键，显示温度数据，松开按键，返回“时钟显示”界面。



这里其实很好解释，按下可以使用Old进行判定，松开可以通过Up进行判定，为什么我们不用Old进行松手判定呢，因为Old在没有按键按下的时候一直是0，所以会导致错误计算。

```
1 //以这个为例
2 //首先需要定义一个界面显示的一个变量
3 unsigned char Seg_show_mode; //0 时间显示 1 温度显示
4 //然后开始我们的键盘函数
5 void Key_Proc()
6 {
7     static uchar Key_Val, Key_Down, Key_Up, Key_Old;
8     if (Key_Slow_Down)
9         return;
10    Key_Slow_Down = 1;
11
12    Key_Val = Key_Read();
13    Key_Down = Key_Val & (Key_Old ^ Key_Val);
14    Key_Up = ~Key_Val & (Key_Old ^ Key_Val);
15    Key_Old = Key_Val;
16    if(Key_Old==4)
17        Seg_show_mode=1;
18    if(Key_Up==4)
19        Seg_show_mode=0;
20 }
```

密码门输入



这里主要是说明一下输入，即从右到左输入显示数据，并且高位熄灭

我们首先需要修改一下底层，适应我们的键盘，这里提醒一下，我们不能把原本的temp=0给改了，这个地方必须锁死

然后为了防止按键冲突，这里建议使用100-109来映射键盘0-9，修改底层的代码这里就不写了

这里的题设条件为，按键输入0-9，按下S17删除上一个值，按下S18清空数据

按键方面



```
1 //以这个为例
2 //首先需要定义一个变量数组和一个指针来进行密码存放
3 unsigned char dat[8];
4 unsigned char dat_index;
5 //然后开始我们的键盘函数
6 void Key_Proc()
7 {
8     static uchar Key_Val, Key_Down, Key_Up, Key_Old;
9     unsigned char i;
10    if (Key_Slow_Down)
11        return;
12    Key_Slow_Down = 1;
13
14    Key_Val = Key_Read();
15    Key_Down = Key_Val & (Key_Old ^ Key_Val);
16    Key_Up = ~Key_Val & (Key_Old ^ Key_Val);
17    Key_Old = Key_Val;
18    if(Key_Down==17)
19    {
20        dat[dat_index]=0; //清除这一位
21        dat_index=(dat_index==0)?0:--dat_index; //下标前移
22    }
23    if(Key_Down==18)
24    {
25        dat_index=0;
26        //清空数组
27        for (i=0;i<8;i++)
28            dat[i]=0;
29    }
30    //密码没有输入到极限
31    if(dat_index<8)
32    {
33        //输入的数据在100-109之间->映射为0-9之间
34        if(Key_Down>=100&&Key_Down<=109)
35        {
36            dat[dat_index]=Key_Down;
37            dat_index++;
38        }
39    }
40 }
```



```
1 //有数据输入
2 if (input_data_index == 0)
3 {
4     //向右对齐
5     for (i = 0; i < input_data_index; i + )
6     {
7         Seg_Buf[7 - i] = input_data[input_data_index - i - 1];
8     }
9     //其他位灭掉
10    for (; i < 8; i + )
11    {
12        Seg_Buf[7 - i] = 10; // 我段码表里面定义的全灭的序号
13    }
14 }
15 //没有数据输入
16 else
17 {
18     for (i = 0; i < 8; i + )
19     {
20         Seg_Buf[i] = 10; // 我段码表里面定义的全灭的序号
21     }
22 }
```

双按键（十四届国赛新增考点）



S8、S9：定义“恢复出厂设置”功能。

若在任意界面下，检测到 S8、S9 按键均处于按下状态，且状态持续时间超过 2 秒，则恢复到初始状态（四、初始状态）。



这里需要修改一下底层，修改如下所示，新增temp=89

底层代码



```
1 unsigned char Key_Read()
2 {
3     unsigned char temp = 0;
4     ET0 = 0; //串口使用屏蔽
5     P44 = 0;    P42 = 1;    P35 = 1;    P34 = 1;
6     if (P33 == 0)    temp = 4;
```

```

7     if (P32 == 0)      temp = 5;
8     if (P31 == 0)      temp = 6;
9     if (P30 == 0)      temp = 7;
10    P44 = 1;      P42 = 0;      P35 = 1;      P34 = 1;
11    if (P33 == 0)      temp = 8;
12    if (P32 == 0)      temp = 9;
13    if (P31 == 0)      temp = 10;
14    if (P30 == 0)      temp = 11;
15    if (P33 == 0 && P32 == 0) temp = 89;
16    P44 = 1;      P42 = 1;      P35 = 0;      P34 = 1;
17    if (P33 == 0)      temp = 12;
18    if (P32 == 0)      temp = 13;
19    if (P31 == 0)      temp = 14;
20    if (P30 == 0)      temp = 15;
21    P44 = 1;      P42 = 1;      P35 = 1;      P34 = 0;
22    if (P33 == 0)      temp = 16;
23    if (P32 == 0)      temp = 17;
24    if (P31 == 0)      temp = 18;
25    if (P30 == 0)      temp = 19;
26    ET0 = 1; //串口使用屏蔽
27    P3 = 0xff;
28    return temp;
29 }

```

按键方面



这里需要加锁，因为有些时候，可能S8或者S9有单击按键的功能，可能误触发

```

1 //以这个为例
2 //首先需要定义一个变量数组和一个指针来进行密码存放
3 unsigned char dat[8];
4 unsigned char dat_index;
5 //然后开始我们的键盘函数
6 void Key_Proc()
7 {
8     static uchar Key_Val, Key_Down, Key_Up, Key_Old;
9     unsigned char i;
10    if (Key_Slow_Down)
11        return;
12    Key_Slow_Down = 1;
13
14    Key_Val = Key_Read();
15    Key_Down = Key_Val & (Key_Old ^ Key_Val);

```

```

16     Key_Up = ~Key_Val & (Key_Old ^ Key_Val);
17     Key_Old = Key_Val;
18     if (Key_Old == 89)
19     {
20         Key_lock = 1; // 按键上锁
21         //执行相关函数
22     }
23     //当按键被锁住（我们需要同时按下两个按键），并且有按键未被弹起
24     if (Key_lock & Key_Old)
25         return;
26     Key_lock = 0; //按键解锁
27     //执行其他的正常函数
28 }
29
30

```

按键多击



这个是群友提到的一个问题，这里做个简单说明，首先，我们要明确一下按键按下的阈值时间（我这里认为是100ms），然后就是进行一个计数就行

这里用S4来举例

按键方面



```

1 //以这个为例
2 bit Key_click_flag; //按键触发标志
3 unsigned char time_100ms;
4 unsigned char Key_click_count;
5 //然后开始我们的键盘函数
6 void Key_Proc()
7 {
8     static uchar Key_Val, Key_Down, Key_Up, Key_Old;
9     unsigned char i;
10    if (Key_Slow_Down)
11        return;
12    Key_Slow_Down = 1;
13
14    Key_Val = Key_Read();
15    Key_Down = Key_Val & (Key_Old ^ Key_Val);
16    Key_Up = ~Key_Val & (Key_Old ^ Key_Val);
17    Key_Old = Key_Val;
18    //按下S4的时候，

```

```

19     if (Key_Down == 4)
20     {
21         Key_click_count++;
22         Key_click_flag = 1;
23         time_100ms = 0;
24     }
25     //当定时到达100ms的时候，也就是超过阈值了,我们就认为按键结束，开始进
    行判断
26     if (time_100ms == 100)
27     {
28         switch(Key_click_count)
29         {
30             case 1:
31                 //执行单击
32                 break;
33             case 2:
34                 //执行双击
35                 break;
36                 //可以添加更多case来进行n击的操作
37         }
38         time_100ms = 0;
39         Key_click_flag = 0; //等待下一次多击按键的触发
40         Key_click_count = 0; // 记录值归零
41     }
42 }

```

■ 定时器方面



```

1  if (Key_click_flag)
2  {
3      if ( ++time_100ms == 100)
4      {
5          time_100ms = 101; //锁死时间
6      }
7  }
8  else
9      time_100ms=0;

```