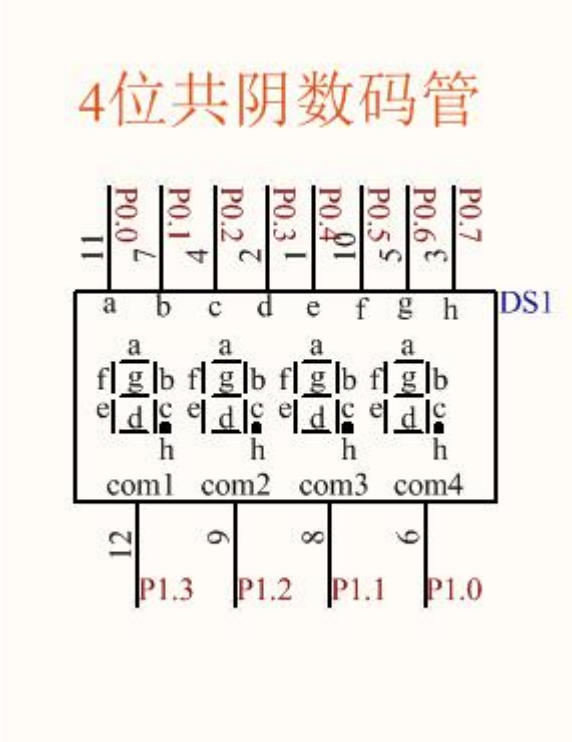
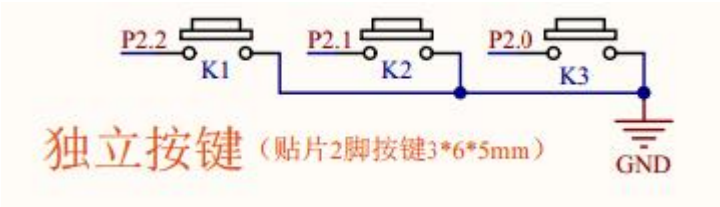
第一百二十二节： 利用定时中断做的“时分秒”数显时钟。

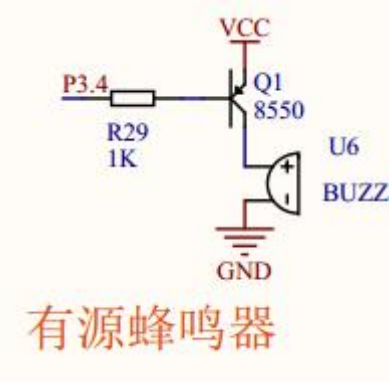
【122.1　利用定时中断做的“时分秒”数显时钟。】



上图 122.1.1　数码管



上图 122.1.2　独立按键



上图 122.1.3　有源蜂鸣器

本节的数显时钟小项目，意在人机界面程序框架的综合训练。程序功能如下：

（1）只有“时分秒”，没有“年月日”。

（2）平时时钟正常工作的时候，四位数码管的显示格式是这样的“HH.MM”，其中 HH 代表“时”，MM 代表“分”，而中间的小数点“.”每隔一秒闪烁一次。

（3）K1［设置键］与[切换窗口键]。当数码管“没有闪烁”时（处于正常工作模式），“长按”K1 键则进入“闪烁模式”（修改时钟模式），“闪烁模式”一共有 3 个窗口，分别是“1-HH”，“2-MM”，“3-SS”。其中“HH”“MM”“SS”分别代表可修改的“时”“分”“秒”，它们处于“闪烁”的状态，代表可编辑。此时，“短按”K1 按键代表[切换窗口键]，可以使数码管在“1-HH”，“2-MM”，“3-SS”三个窗口之间依次切换。修改完毕后，只需“长按”K1 键代表确定完成并且退出当前“闪烁模式”返回到时钟的“正常工作模式”。

（4）K2［加键］。当数码管某位正在闪烁时，此时 K2 是［加键］，按 K2 会使数据“自加 1”。

（5）K3［减键］。当数码管某位正在闪烁时，此时 K3 是［减键］，按 K3 会使数据“自减 1”。

（6）处于“闪烁模式”时的 3 个窗口的数据范围。处于修改“时”的“1-HH”窗口时，HH 的范围是：0 到 23；处于修改“分”的“2-MM”窗口时，MM 的范围是：0 到 59；处于修改“秒”的“3-SS”窗口时，SS 的范围是：0 到 59。

代码如下：

```
#include "REG52.H"

#define KEY_FILTER_TIME  25
#define KEY_LONG_TIME    500

#define SCAN_TIME  1
#define VOICE_TIME   50
#define BLINK_TIME   250

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

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

void VoiceScan(void);
void DisplayScan(void);
void DisplayTask(void);
void Wd1(void);   //窗口 1。时钟正常工作的窗口“HH.MM”。小数点在闪烁跳动。
void Wd2(void);   //窗口 2。闪烁模式，修改“时”的“1-HH”的窗口。
void Wd3(void);   //窗口 3。闪烁模式，修改“分”的“2-MM”的窗口。
void Wd4(void);   //窗口 4。闪烁模式，修改“秒”的“3-SS”的窗口。

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

```
sbit KEY_INPUT1=P2^2;
sbit KEY_INPUT2=P2^1;
sbit KEY_INPUT3=P2^0;

sbit P1_0=P1^0;
sbit P1_1=P1^1;
sbit P1_2=P1^2;
sbit P1_3=P1^3;

sbit P3_4=P3^4;

//数码管转换表
code unsigned char Cu8DigTable[]=
{
0x3f,  //0       序号 0
0x06,  //1       序号 1
0x5b,  //2       序号 2
0x4f,  //3       序号 3
0x66,  //4       序号 4
0x6d,  //5       序号 5
0x7d,  //6       序号 6
0x07,  //7       序号 7
0x7f,  //8       序号 8
0x6f,  //9       序号 9
0x00,  //不显示  序号 10
0x40,  //横杠-   序号 11
};

volatile unsigned char vGu8ScanTimerFlag=0;
volatile unsigned int vGu16ScanTimerCnt=0;

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

volatile unsigned char vGu8BlinkTimerFlag=0;
volatile unsigned int vGu16BlinkTimerCnt=0;

//时钟的软件定时器，注意，这里是 unsigned long 类型，范围是 0 到 4294967295 毫秒
volatile unsigned char vGu8ClockTimerFlag=0;
volatile unsigned long vGu32ClockTimerCnt=0;

//时钟正常工作的时候，每 500ms 更新显示一次
volatile unsigned char vGu8UpdateTimerFlag=0;
```

```
volatile unsigned int vGu16UpdateTimerCnt=0;

unsigned char Gu8EditData_1=0;  //是中间变量，用于编辑窗口“1-HH”下的 HH 数据。
unsigned char Gu8EditData_2=0;  //是中间变量，用于编辑窗口“2-MM”下的 MM 数据。
unsigned char Gu8EditData_3=0;  //是中间变量，用于编辑窗口“3-SS”下的 SS 数据。

unsigned char Gu8Wd=0;   //窗口选择变量。人机交互程序框架的支点。
unsigned char Gu8WdUpdate=0;  //整屏更新变量。

unsigned char Gu8PartUpdate_1=0;   //局部 1 的更新变量,
unsigned char Gu8PartUpdate_2=0;   //局部 2 的更新变量
unsigned char Gu8PartUpdate_3=0;   //局部 3 的更新变量,

volatile unsigned char vGu8Display_Righ_4=0;
volatile unsigned char vGu8Display_Righ_3=0;
volatile unsigned char vGu8Display_Righ_2=0;
volatile unsigned char vGu8Display_Righ_1=0;

volatile unsigned char vGu8Display_Righ_Dot_4=0;
volatile unsigned char vGu8Display_Righ_Dot_3=1;    //开机默认保留显示 2 个小数点
volatile unsigned char vGu8Display_Righ_Dot_2=0;
volatile unsigned char vGu8Display_Righ_Dot_1=0;

volatile unsigned char vGu8KeySec=0;

void main()
{
    SystemInitial();
    Delay(10000);
    PeripheralInitial();
    while(1)
    {
        KeyTask();      //按键的任务函数
        DisplayTask();  //数码管显示的上层任务函数
    }
}

void KeyTask(void)    //按键的任务函数
{
    if(0==vGu8KeySec)
    {
        return;
    }
```

```
switch(vGu8KeySec)
{
   case 1:     //按键 K1 的“短按”。在“闪烁模式”下切换数码管的窗口。
        switch(Gu8Wd) //在某个窗口下
        {
            case 2:     //窗口 2。修改“时”的“1-HH”窗口。
                  Gu8Wd=3; //切换到窗口 3 的“2-MM”窗口
                  Gu8WdUpdate=1; //整屏更新
                  break;

            case 3:     //窗口 3。修改“分”的“2-MM”窗口。
                  Gu8Wd=4; //切换到窗口 4 的“3-SS”窗口
                  Gu8WdUpdate=1; //整屏更新
                  break;

            case 4:     //窗口 4。修改“秒”的“3-SS”窗口。
                  Gu8Wd=2; //切换到窗口 2 的“1-HH”窗口
                  Gu8WdUpdate=1; //整屏更新
                  break;
        }

        vGu8BeepTimerFlag=0;
        vGu16BeepTimerCnt=VOICE_TIME;  //蜂鸣器发出“滴”一声
        vGu8BeepTimerFlag=1;

        vGu8KeySec=0;
        break;

   case 2:     //按键 K2 [加键]

        switch(Gu8Wd) //在某个窗口下
        {
            case 2:     //窗口 2。修改“时”的“1-HH”窗口。
                  if(Gu8EditData_1<23) //“时”的范围是 0 到 23
                  {
                      Gu8EditData_1++;
                  }
                  Gu8PartUpdate_1=1;   //局部 1 更新显示
                  break;

            case 3:     //窗口 3。修改“分”的“2-MM”窗口。
                  if(Gu8EditData_2<59) //“分”的范围是 0 到 59
                  {
                      Gu8EditData_2++;
```

```
                }
                Gu8PartUpdate_1=1;   //局部 1 更新显示
                break;

            case 4:     //窗口 4。修改“秒”的“3-SS”窗口。
                if(Gu8EditData_3<59) //“秒”的范围是 0 到 59
                {
                    Gu8EditData_3++;
                }
                Gu8PartUpdate_1=1;   //局部 1 更新显示
                break;
        }

        vGu8BeepTimerFlag=0;
        vGu16BeepTimerCnt=VOICE_TIME;  //蜂鸣器发出“滴”一声
        vGu8BeepTimerFlag=1;

        vGu8KeySec=0;
        break;

    case 3:     //按键 K3［减键］与［开始健］
        switch(Gu8Wd) //在某个窗口下
        {
            case 2:     //窗口 2。修改“时”的“1-HH”窗口。
                if(Gu8EditData_1>0)
                {
                    Gu8EditData_1--;
                }
                Gu8PartUpdate_1=1;   //局部 1 更新显示
                break;

            case 3:     //窗口 3。修改“分”的“2-MM”窗口。
                if(Gu8EditData_2>0)
                {
                    Gu8EditData_2--;
                }
                Gu8PartUpdate_1=1;   //局部 1 更新显示
                break;

            case 4:     //窗口 4。修改“秒”的“3-SS”窗口。
                if(Gu8EditData_3>0)
                {
                    Gu8EditData_3--;
                }
```

```
                Gu8PartUpdate_1=1;   //局部 1 更新显示
                break;
        }

        vGu8BeepTimerFlag=0;
        vGu16BeepTimerCnt=VOICE_TIME;  //蜂鸣器发出“滴”一声
        vGu8BeepTimerFlag=1;

        vGu8KeySec=0;
        break;

    case 4:     //K1 按键的“长按”,具有进入和退出“闪烁模式”的功能。“退出”隐含“确定”

        switch(Gu8Wd) //在某个窗口下
        {
           case 1:     //窗口 1。时钟正常工作的窗口。
                vGu8ClockTimerFlag=0;  //停止时钟的定时器

                Gu8EditData_1=vGu32ClockTimerCnt/3600000;  //分解成“时”个体
                Gu8EditData_2=vGu32ClockTimerCnt%3600000/60000;  //分解成“分”个体
                Gu8EditData_3=vGu32ClockTimerCnt%3600000%60000/1000;  //分解成“秒”个体

                Gu8Wd=2; //切换到窗口 2 的“1-HH”的闪烁窗口
                Gu8WdUpdate=1; //整屏更新
                break;

           case 2:     //窗口 2。修改时钟时间的“1-HH”的闪烁窗口
           case 3:     //窗口 3。修改时钟时间的“2-MM”的闪烁窗口
           case 4:     //窗口 4。修改时钟时间的“3-SS”的闪烁窗口
                 //把个体合并还原成当前时钟时间的数据
        vGu32ClockTimerCnt=Gu8EditData_1*3600000+Gu8EditData_2*60000+Gu8EditData_3*1000;
                vGu8ClockTimerFlag=1;  //启动时钟的定时器

                Gu8Wd=1; //切换到窗口 1 的正常工作的窗口
                Gu8WdUpdate=1; //整屏更新
                break;
        }

        vGu8BeepTimerFlag=0;
        vGu16BeepTimerCnt=VOICE_TIME;  //蜂鸣器发出“滴”一声
        vGu8BeepTimerFlag=1;

        vGu8KeySec=0;
        break;
```

```
    }
}

void DisplayTask(void) //数码管显示的上层任务函数
{
    switch(Gu8Wd)  //以窗口选择 Gu8Wd 为支点，去执行对应的窗口显示函数。又一次用到 switch 语句
    {
        case 1:
            Wd1();   //窗口 1。时钟正常运行的窗口
            break;
        case 2:
            Wd2();   //窗口 2。修改“时”的“1-HH”窗口
            break;
        case 3:
            Wd3();   //窗口 3。修改“分”的“2-MM”窗口
            break;
        case 4:
            Wd4();   //窗口 4。修改“秒”的“3-SS”窗口
            break;
    }
}

void Wd1(void)   //窗口 1。时钟正常运行的窗口
{
    //需要借用的中间变量，用来拆分数据位。
    static unsigned char Su8Temp_4,Su8Temp_3,Su8Temp_2,Su8Temp_1; //需要借用的中间变量
    static unsigned char Su8BlinkFlag=0;  //两种状态的切换判断的中间变量


    if(1==Gu8WdUpdate) //如果需要整屏更新
    {
          Gu8WdUpdate=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

          //属于静态数据，起“装饰”作用，切换窗口后只扫描一次的代码。
          vGu8Display_Righ_Dot_4=0;
          vGu8Display_Righ_Dot_3=1;   //保留显示 2 位小数点
          vGu8Display_Righ_Dot_2=0;
          vGu8Display_Righ_Dot_1=0;

          Gu8PartUpdate_1=1;  //局部 1 更新显示
    }

    if(1==Gu8PartUpdate_1) //局部 1 更新显示，更新显示一次数据和闪烁的的小数点
```

```
    {
        Gu8PartUpdate_1=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

        Su8Temp_4=vGu32ClockTimerCnt/3600000/10; //时的十位
        Su8Temp_3=vGu32ClockTimerCnt/3600000%10; //时的个位
        Su8Temp_2=vGu32ClockTimerCnt%3600000/60000/10; //分的十位
        Su8Temp_1=vGu32ClockTimerCnt%3600000/60000%10; //秒的个位

        //小数点的闪烁
        if(0==Su8BlinkFlag)
        {
            Su8BlinkFlag=1;
            vGu8Display_Righ_Dot_3=1;   //显示第 2 位小数点。

        }
        else
        {
            Su8BlinkFlag=0;
            vGu8Display_Righ_Dot_3=0;   //不显示第 2 位小数点
        }

        vGu8Display_Righ_4=Su8Temp_4;  //过渡需要显示的数据到底层驱动变量
        vGu8Display_Righ_3=Su8Temp_3;  //过渡需要显示的数据到底层驱动变量
        vGu8Display_Righ_2=Su8Temp_2;  //过渡需要显示的数据到底层驱动变量
        vGu8Display_Righ_1=Su8Temp_1;  //过渡需要显示的数据到底层驱动变量
    }

    if(0==vGu16UpdateTimerCnt)  //每隔 500ms 就更新显示一次数据和闪烁的的小数点
    {
        vGu8UpdateTimerFlag=0;
        vGu16UpdateTimerCnt=500;  //重设定时器的定时时间
        vGu8UpdateTimerFlag=1;

        Gu8PartUpdate_1=1;  //局部 1 更新显示
    }

}

void Wd2(void)   //窗口 2。修改“时”的“1-HH”窗口。
{
    //需要借用的中间变量，用来拆分数据位。
    static unsigned char Su8Temp_2,Su8Temp_1; //需要借用的中间变量
    static unsigned char Su8BlinkFlag=0;  //两种状态的切换判断的中间变量
```

```
if(1==Gu8WdUpdate) //如果需要整屏更新
{
      Gu8WdUpdate=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

      //属于静态数据，起“装饰”作用，切换窗口后只扫描一次的代码。
      vGu8Display_Righ_4=1;   //显示数字“1”
      vGu8Display_Righ_3=11;  //显示横杠“-”

      vGu8Display_Righ_Dot_4=0;
      vGu8Display_Righ_Dot_3=0;
      vGu8Display_Righ_Dot_2=0;
      vGu8Display_Righ_Dot_1=0;

      Gu8PartUpdate_1=1;  //局部 1 更新显示
}

if(1==Gu8PartUpdate_1) //局部 1 更新显示
{
      Gu8PartUpdate_1=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

      Su8Temp_2=Gu8EditData_1/10;  //显示“时”的十位
      Su8Temp_1=Gu8EditData_1%10;  //显示“时”的个位

      vGu8Display_Righ_2=Su8Temp_2;  //过渡需要显示的数据到底层驱动变量
      vGu8Display_Righ_1=Su8Temp_1;  //过渡需要显示的数据到底层驱动变量
}


if(0==vGu16BlinkTimerCnt)  //某位被选中的数码管跳动闪烁的定时器
{
    vGu8BlinkTimerFlag=0;
    vGu16BlinkTimerCnt=BLINK_TIME;  //重设定时器的定时时间
    vGu8BlinkTimerFlag=1;

    if(0==Su8BlinkFlag)  //两种状态的切换判断
    {
        Su8BlinkFlag=1;
        Su8Temp_2=10;  //10 代表不显示
        Su8Temp_1=10;  //10 代表不显示
    }
    else
    {
        Su8BlinkFlag=0;
```

```
            Su8Temp_2=Gu8EditData_1/10;  //显示“时”的十位
            Su8Temp_1=Gu8EditData_1%10;  //显示“时”的个位

        }

        vGu8Display_Righ_2=Su8Temp_2;  //过渡需要显示的数据到底层驱动变量
        vGu8Display_Righ_1=Su8Temp_1;  //过渡需要显示的数据到底层驱动变量
    }
}

void Wd3(void)   //窗口 3。修改“分”的“2-MM”窗口。
{
    //需要借用的中间变量，用来拆分数据位。
    static unsigned char Su8Temp_2,Su8Temp_1; //需要借用的中间变量
    static unsigned char Su8BlinkFlag=0;  //两种状态的切换判断的中间变量

    if(1==Gu8WdUpdate) //如果需要整屏更新
    {
          Gu8WdUpdate=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

          //属于静态数据，起“装饰”作用，切换窗口后只扫描一次的代码。
          vGu8Display_Righ_4=2;   //显示数字“2”
          vGu8Display_Righ_3=11;  //显示横杠“-”

          vGu8Display_Righ_Dot_4=0;
          vGu8Display_Righ_Dot_3=0;
          vGu8Display_Righ_Dot_2=0;
          vGu8Display_Righ_Dot_1=0;

          Gu8PartUpdate_1=1;  //局部 1 更新显示
    }

    if(1==Gu8PartUpdate_1) //局部 1 更新显示
    {
          Gu8PartUpdate_1=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

          Su8Temp_2=Gu8EditData_2/10;  //显示“分”的十位
          Su8Temp_1=Gu8EditData_2%10;  //显示“分”的个位

          vGu8Display_Righ_2=Su8Temp_2;  //过渡需要显示的数据到底层驱动变量
          vGu8Display_Righ_1=Su8Temp_1;  //过渡需要显示的数据到底层驱动变量
    }
```

```
    if(0==vGu16BlinkTimerCnt)  //某位被选中的数码管跳动闪烁的定时器
    {
        vGu8BlinkTimerFlag=0;
        vGu16BlinkTimerCnt=BLINK_TIME;  //重设定时器的定时时间
        vGu8BlinkTimerFlag=1;

        if(0==Su8BlinkFlag)  //两种状态的切换判断
        {
            Su8BlinkFlag=1;
            Su8Temp_2=10;  //10 代表不显示
            Su8Temp_1=10;  //10 代表不显示
        }
        else
        {
            Su8BlinkFlag=0;
            Su8Temp_2=Gu8EditData_2/10;  //显示“分”的十位
            Su8Temp_1=Gu8EditData_2%10;  //显示“分”的个位
        }

        vGu8Display_Righ_2=Su8Temp_2;  //过渡需要显示的数据到底层驱动变量
        vGu8Display_Righ_1=Su8Temp_1;  //过渡需要显示的数据到底层驱动变量
    }
}

void Wd4(void)   //窗口 4。修改“秒”的“3-SS”窗口。
{
    //需要借用的中间变量，用来拆分数据位。
    static unsigned char Su8Temp_2,Su8Temp_1; //需要借用的中间变量
    static unsigned char Su8BlinkFlag=0;  //两种状态的切换判断的中间变量

    if(1==Gu8WdUpdate) //如果需要整屏更新
    {
          Gu8WdUpdate=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

          //属于静态数据，起“装饰”作用，切换窗口后只扫描一次的代码。
          vGu8Display_Righ_4=3;   //显示数字“3”
          vGu8Display_Righ_3=11;  //显示横杠“-”

          vGu8Display_Righ_Dot_4=0;
          vGu8Display_Righ_Dot_3=0;
          vGu8Display_Righ_Dot_2=0;
          vGu8Display_Righ_Dot_1=0;

          Gu8PartUpdate_1=1;  //局部 1 更新显示
```

```
    }

    if(1==Gu8PartUpdate_1) //局部 1 更新显示
    {
          Gu8PartUpdate_1=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

          Su8Temp_2=Gu8EditData_3/10;  //显示“秒”的十位
          Su8Temp_1=Gu8EditData_3%10;  //显示“秒”的个位

          vGu8Display_Righ_2=Su8Temp_2;  //过渡需要显示的数据到底层驱动变量
          vGu8Display_Righ_1=Su8Temp_1;  //过渡需要显示的数据到底层驱动变量
    }



    if(0==vGu16BlinkTimerCnt)  //某位被选中的数码管跳动闪烁的定时器
    {
        vGu8BlinkTimerFlag=0;
        vGu16BlinkTimerCnt=BLINK_TIME;  //重设定时器的定时时间
        vGu8BlinkTimerFlag=1;

        if(0==Su8BlinkFlag)  //两种状态的切换判断
        {
            Su8BlinkFlag=1;
            Su8Temp_2=10;  //10 代表不显示
            Su8Temp_1=10;  //10 代表不显示
        }
        else
        {
            Su8BlinkFlag=0;
            Su8Temp_2=Gu8EditData_3/10;  //显示“秒”的十位
            Su8Temp_1=Gu8EditData_3%10;  //显示“秒”的个位
        }

        vGu8Display_Righ_2=Su8Temp_2;  //过渡需要显示的数据到底层驱动变量
        vGu8Display_Righ_1=Su8Temp_1;  //过渡需要显示的数据到底层驱动变量
    }
}

void KeyScan(void)  //按键底层的驱动扫描函数，放在定时中断函数里
{
   static unsigned char Su8KeyShortFlag=0;  //按键“短按”触发的标志
   static unsigned char Su8KeyLock1;
   static unsigned int  Su16KeyCnt1;
   static unsigned char Su8KeyLock2;
```

```
 static unsigned int  Su16KeyCnt2;
 static unsigned char Su8KeyLock3;
 static unsigned int  Su16KeyCnt3;

//需要详细分析以下这段“短按”与“长按”代码的朋友，请参考第 96 节。
 if(0!=KEY_INPUT1)
 {
    Su8KeyLock1=0;
    Su16KeyCnt1=0;
    if(1==Su8KeyShortFlag)
    {
        Su8KeyShortFlag=0;
        vGu8KeySec=1;    //触发 K1 的“短按”
    }
 }
 else if(0==Su8KeyLock1)
 {
    Su16KeyCnt1++;

    if(Su16KeyCnt1>=KEY_FILTER_TIME)
    {
          Su8KeyShortFlag=1;
    }

    if(Su16KeyCnt1>=KEY_LONG_TIME)
    {
          Su8KeyLock1=1;
          Su8KeyShortFlag=0;
          vGu8KeySec=4; //触发 K1 的“长按”
    }
 }

 if(0!=KEY_INPUT2)
 {
    Su8KeyLock2=0;
    Su16KeyCnt2=0;
 }
 else if(0==Su8KeyLock2)
 {
    Su16KeyCnt2++;
    if(Su16KeyCnt2>=KEY_FILTER_TIME)
    {
       Su8KeyLock2=1;
       vGu8KeySec=2;
```

```
      }
   }

   if(0!=KEY_INPUT3)
   {
      Su8KeyLock3=0;
      Su16KeyCnt3=0;
   }
   else if(0==Su8KeyLock3)
   {
      Su16KeyCnt3++;
      if(Su16KeyCnt3>=KEY_FILTER_TIME)
      {
         Su8KeyLock3=1;
         vGu8KeySec=3;
      }
   }

}

void DisplayScan(void)    //数码管底层的驱动扫描函数，放在定时中断函数里
{
    static unsigned char Su8GetCode;
    static unsigned char Su8ScanStep=1;

    if(0==vGu16ScanTimerCnt)
    {


        P0=0x00;
        P1_0=1;
        P1_1=1;
        P1_2=1;
        P1_3=1;

        switch(Su8ScanStep)
        {
           case 1:
                Su8GetCode=Cu8DigTable[vGu8Display_Righ_1];

                if(1==vGu8Display_Righ_Dot_1)
                {
                    Su8GetCode=Su8GetCode|0x80;
                }
```

```
            P0=Su8GetCode;
            P1_0=0;
            P1_1=1;
            P1_2=1;
            P1_3=1;
            break;

        case 2:
            Su8GetCode=Cu8DigTable[vGu8Display_Righ_2];
            if(1==vGu8Display_Righ_Dot_2)
            {
                Su8GetCode=Su8GetCode|0x80;
            }
            P0=Su8GetCode;
            P1_0=1;
            P1_1=0;
            P1_2=1;
            P1_3=1;
            break;

        case 3:
            Su8GetCode=Cu8DigTable[vGu8Display_Righ_3];
            if(1==vGu8Display_Righ_Dot_3)
            {
                Su8GetCode=Su8GetCode|0x80;
            }
            P0=Su8GetCode;
            P1_0=1;
            P1_1=1;
            P1_2=0;
            P1_3=1;
            break;

        case 4:
            Su8GetCode=Cu8DigTable[vGu8Display_Righ_4];
            if(1==vGu8Display_Righ_Dot_4)
            {
                Su8GetCode=Su8GetCode|0x80;
            }
            P0=Su8GetCode;
            P1_0=1;
            P1_1=1;
            P1_2=1;
            P1_3=0;
```

```
                break;

        }

        Su8ScanStep++;
        if(Su8ScanStep>4)
        {
            Su8ScanStep=1;
        }

        vGu8ScanTimerFlag=0;
        vGu16ScanTimerCnt=SCAN_TIME;
        vGu8ScanTimerFlag=1;
    }
}

void VoiceScan(void) //蜂鸣器的驱动函数
{

      static unsigned char Su8Lock=0;

      if(1==vGu8BeepTimerFlag&&vGu16BeepTimerCnt>0)
      {
          if(0==Su8Lock)
          {
               Su8Lock=1;
               BeepOpen();
          }
          else
          {

               vGu16BeepTimerCnt--;

               if(0==vGu16BeepTimerCnt)
               {
                    Su8Lock=0;
                    BeepClose();
               }

          }
      }
}

void BeepOpen(void)
```

```
{
    P3_4=0;
}

void BeepClose(void)
{
    P3_4=1;
}

void T0_time() interrupt 1
{
    VoiceScan();    //蜂鸣器的驱动函数
    KeyScan();      //按键底层的驱动扫描函数
    DisplayScan();  //数码管底层的驱动扫描函数

    if(1==vGu8ScanTimerFlag&&vGu16ScanTimerCnt>0)
    {
        vGu16ScanTimerCnt--;  //递减式的软件定时器
    }

    if(1==vGu8BlinkTimerFlag&&vGu16BlinkTimerCnt>0)   //数码管闪烁跳动的定时器
    {
        vGu16BlinkTimerCnt--;  //递减式的软件定时器
    }

    //在正常工作的窗口下，每 500ms 就定时更新一次显示的软件定时器
    if(1==vGu8UpdateTimerFlag&&vGu16UpdateTimerCnt>0)
    {
        vGu16UpdateTimerCnt--;  //递减式的软件定时器
    }

    //时钟实际走的时间的软件定时器，注意，这里是递增式的软件定时器
    if(1==vGu8ClockTimerFlag)
    {
        vGu32ClockTimerCnt++;  //递增式的软件定时器
        if(vGu32ClockTimerCnt>=86400000) //86400000 毫秒代表 24 时
        {
            vGu32ClockTimerCnt=0;
        }
    }

    TH0=0xfd;   //此参数可根据具体的时间来修改，尽量确保每定时中断一次接近 1ms
    TL0=0x40;   //此参数可根据具体的时间来修改，尽量确保每定时中断一次接近 1ms
}
```

```
void SystemInitial(void)
{
    P0=0x00;
    P1_0=1;
    P1_1=1;
    P1_2=1;
    P1_3=1;

    TMOD=0x01;
    TH0=0xfd;   //此参数可根据具体的时间来修改，尽量确保每定时中断一次接近 1ms
    TL0=0x40;   //此参数可根据具体的时间来修改，尽量确保每定时中断一次接近 1ms
    EA=1;
    ET0=1;
    TR0=1;

    //上电初始化一些关键的数据

    Gu8Wd=1;   //窗口 1。开机默认处于正常工作的窗口
    Gu8WdUpdate=1;  //整屏更新变量

    vGu8ClockTimerFlag=0;
    vGu32ClockTimerCnt=43200000;  //43200000 毫秒开机默认 12:00 点。12 时就是 43200000 毫秒
    vGu8ClockTimerFlag=1;   //启动时钟的定时器

    //时钟正常工作的时候，每 500ms 更新显示一次
    vGu16UpdateTimerCnt=500;
    vGu8UpdateTimerFlag=1; //启动小数点闪烁的定时器

}

void Delay(unsigned long u32DelayTime)
{
    for(;u32DelayTime>0;u32DelayTime--);
}

void PeripheralInitial(void)
{

}
```