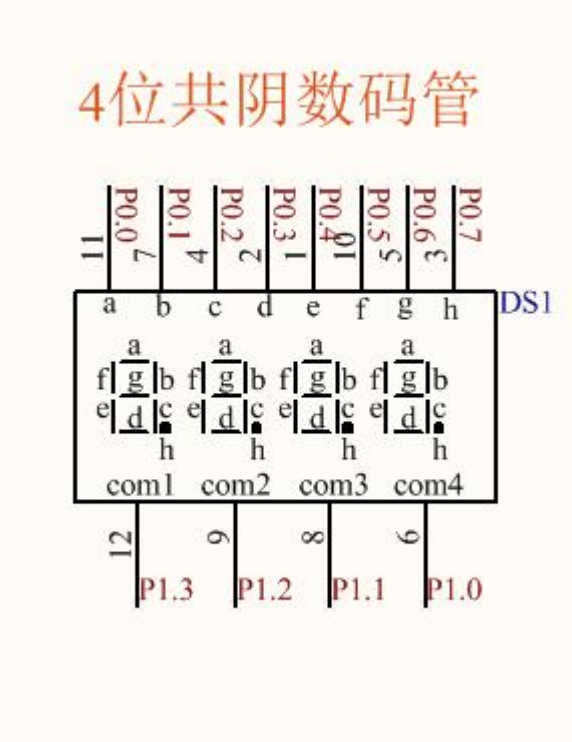
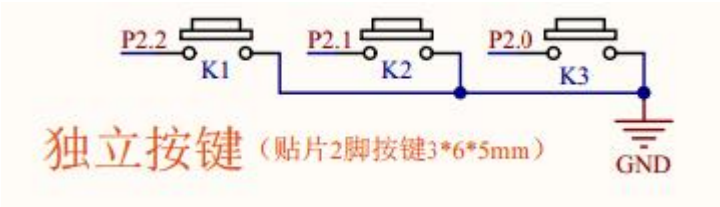
第一百二十一节： 可调参数的数码管倒计时。

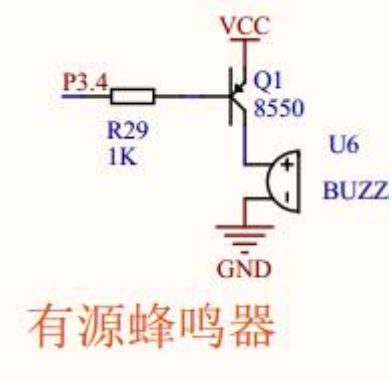
【121.1　可调参数的数码管倒计时。】



上图 121.1.1　数码管



上图 121.1.2　独立按键



上图 121.1.3　有源蜂鸣器

上节讲如何设置数据，本节讲“数据”如何关联“某种功能”，本节的“可调参数”就是“数据”，“倒计时”就是“某种功能”。程序功能如下：

（1）倒计时范围从 0.00 秒到 99.99 秒，范围可调。开机默认是:10.00 秒。

（2）K1［设置键］。当数码管“没有闪烁”时，“长按”K1 键则进入“闪烁模式”，某位数码管开始闪烁，闪烁的位代表可修改的位数据，此时再“短按”K1 按键可以使数码管在位之间切换闪烁。当数码管处于“闪烁模式”时，“长按”K1 按键，代表数据修改完成并停止闪烁。

（3）K2［加键］与［复位健］。当数码管某位正在闪烁时，此时 K2 是［加键］，按 K2 会使位数据“自加 1”。当数码管“没有闪烁”时，此时 K2 是［复位键］，按 K2 会使当前倒计时数据恢复“设置值”。

（4）K3［减键］与［开始健］。当数码管某位正在闪烁时，此时 K3 是［减键］，按 K3 会使位数据“自减 1”。当数码管“没有闪烁”时，此时 K3 是［开始键］，按 K3 开始倒计时。

代码如下：

```
#include "REG52.H"

#define KEY_FILTER_TIME  25    //按键的“短按”兼“滤波”的“稳定时间”
#define KEY_LONG_TIME    500   //按键的“长按”兼“滤波”的“稳定时间”

#define SCAN_TIME  1
#define VOICE_TIME   50
#define BLINK_TIME   250    //数码管闪烁跳动的时间的间隔

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);  //上层显示的任务函数
void RunTask(void);      //倒计时的应用程序

void Wd1(void);   //窗口 1 显示函数。用来设置参数。
void Wd2(void);   //窗口 2 显示函数。倒计时的运行显示窗口

void PartUpdate(unsigned char u8Part);  //局部选择对应的某个局部变量更新显示输出

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;   //数码管闪烁跳动的定时器
volatile unsigned int vGu16BlinkTimerCnt=0;

//倒计时的软件定时器，注意，这里是 unsigned long 类型，范围是 0 到 4294967295 毫秒
volatile unsigned char vGu8CountdownTimerFlag=0;
volatile unsigned long vGu32CountdownTimerCnt=10000;  //当前倒计时的计时值
unsigned long Gu32SetData_Countdown=10000;  //倒计时的设置值


//数码管上层每 10ms 就定时刷新一次显示的软件定时器。用于及时更新显示秒表当前的实时数值
volatile unsigned char vGu8UpdateTimerFlag=0;
volatile unsigned int vGu16UpdateTimerCnt=0;
```

```
unsigned char Gu8RunStart=0;  //应用程序的总启动
unsigned char Gu8RunStep=0;  //应用程序的总运行步骤。建议跟 vGu8RunStart 成双成对出现
unsigned char Gu8RunStatus=0; //当前倒计时的状态。0 代表停止，1 代表正在工作中

unsigned char Gu8EditData_4=0;  //对应显示右起第 4 位数码管的“位”数据，是中间变量。
unsigned char Gu8EditData_3=0;  //对应显示右起第 3 位数码管的“位”数据，是中间变量。
unsigned char Gu8EditData_2=0;  //对应显示右起第 2 位数码管的“位”数据，是中间变量。
unsigned char Gu8EditData_1=0;  //对应显示右起第 1 位数码管的“位”数据，是中间变量。

unsigned char Gu8Wd=1;   //窗口选择变量。人机交互程序框架的支点。初始化开机后显示第 1 个窗口。
unsigned char Gu8WdUpdate=1;  //整屏更新变量。初始化为 1 开机后整屏更新一次显示。
unsigned char Gu8Part=0;   //局部选择变量。0 代表当前窗口下没有数据被选中。
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,
unsigned char Gu8PartUpdate_4=0;   //局部 4 的更新变量

volatile unsigned char vGu8Display_Righ_4=1;  //显示“1”，跟 vGu32CountdownTimerCnt 高位一致
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
        RunTask();      //倒计时的应用程序
    }
}

void PartUpdate(unsigned char u8Part)  //局部选择对应的某个局部变量更新显示输出
```

```
{
    switch(u8Part)
    {
       case 1:
            Gu8PartUpdate_1=1;
            break;
       case 2:
            Gu8PartUpdate_2=1;
            break;
       case 3:
            Gu8PartUpdate_3=1;
            break;
       case 4:
            Gu8PartUpdate_4=1;
            break;
    }

}

void RunTask(void)  //倒计时的应用程序
{
    if(0==Gu8RunStart)
    {
       return;  // 如果总开关处于停止状态，则直接退出当前函数，不执行该函数以下的其它代码
    }

    switch(Gu8RunStep)
    {
        case 0:  //在这个步骤里，主要用来初始化一些参数

            vGu8UpdateTimerFlag=0;
            vGu16UpdateTimerCnt=10;  //每 10ms 更新显示一次当前倒计时的时间
            vGu8UpdateTimerFlag=1;

            Gu8RunStep=1; //跳转到每 10ms 更新显示一次的步骤里
            break;

        case 1:  //每 10ms 更新一次显示，确保实时显示当前倒计时的时间
           if(0==vGu16UpdateTimerCnt) //每 10ms 更新显示一次当前倒计时的时间
           {



               vGu8UpdateTimerFlag=0;
               vGu16UpdateTimerCnt=10;  //重置定时器，为下一个 10ms 更新做准备
```

```
                vGu8UpdateTimerFlag=1;

                Gu8WdUpdate=1;  //整屏更新一次显示当前倒计时的时间

                if(0==vGu32CountdownTimerCnt) //如果倒计时的时间到，则跳转到结束的步骤
                {
                    Gu8RunStep=2; //跳转到倒计时结束的步骤
                }

            }
            break;

        case 2:  //倒计时结束的步骤
            //Gu8RunStatus=0; //这行代码注释掉，让每次新启动之前都必须按一次 K1 复位按键才有效

            Gu8RunStart=0;  //倒计时的运行步骤的停止
            Gu8RunStep=0;   //总运行步骤归零。建议跟 vGu8RunStart 成双成对出现

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

            Gu8WdUpdate=1;  //整屏更新一次显示当前倒计时的时间

            break;

    }

}


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

    if(0!=Gu8RunStatus) //在“非停止”状态下，用 return 来拦截一些“不该响应”的按键
    {
        if(2==vGu8KeySec) //在“非停止”状态下，只响应[复位]这个按键
        {
           ;   //这里没有 return 语句，表示可以继续往下扫描本函数余下的代码，没有被拦截。
        }
```

```
    else
    {
       return;   //其余的按键则拦截退出
    }
}


switch(vGu8KeySec)
{
   case 1:     //按键 K1 的“短按”。切换数码管闪烁的位。
        switch(Gu8Wd) //在某个窗口下
        {
            case 1:     //在窗口 1 下

                if(0!=Gu8Part)  //处于“闪烁模式”的时候，是“切换局部”
                 {
                     PartUpdate(Gu8Part);  //切换之前的局部进行更新。
                     Gu8Part++;  //切换局部
                     if(Gu8Part>4)
                     {
                         Gu8Part=1;
                     }
                     PartUpdate(Gu8Part);  //切换之后的局部进行更新。

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

        vGu8KeySec=0;
        break;

   case 2:     //按键 K2 [加键] 与 [复位健]
        if(0!=Gu8Part)  //处于“闪烁模式”的时候，是 [加键]
        {
            switch(Gu8Wd) //在某个窗口下
            {
                case 1:     //在窗口 1 下
                     switch(Gu8Part)  //二级支点的局部选择
                     {
                          case 1:  //局部 1 被选中，代表右起第 4 位数码管被选中。
                                if(Gu8EditData_4<9)
```

```
                        {
                            Gu8EditData_4++;  //编辑“千位”个体的中间变量
                        }
                        PartUpdate(Gu8Part); //当前局部更新显示输出到数码管
                        break;

                    case 2:  //局部 2 被选中，代表右起第 3 位数码管被选中。
                        if(Gu8EditData_3<9)
                        {
                            Gu8EditData_3++;  //编辑“百位”个体的中间变量
                        }
                        PartUpdate(Gu8Part); //当前局部更新显示输出到数码管
                        break;

                    case 3:  //局部 3 被选中，代表右起第 2 位数码管被选中。
                        if(Gu8EditData_2<9)
                        {
                            Gu8EditData_2++;  //编辑“十位”个体的中间变量
                        }
                        PartUpdate(Gu8Part); //当前局部更新显示输出到数码管
                        break;

                    case 4:  //局部 4 被选中，代表右起第 1 位数码管被选中。
                        if(Gu8EditData_1<9)
                        {
                            Gu8EditData_1++;  //编辑“个位”个体的中间变量
                        }
                        PartUpdate(Gu8Part); //当前局部更新显示输出到数码管
                        break;
                }
                break;
        }
    }
    else   //处于“没有闪烁”的时候，是 [复位健]
    {
        Gu8EditData_4=Gu32SetData_Countdown/10000%10; //分解成“十秒”个体
        Gu8EditData_3=Gu32SetData_Countdown/1000%10;  //分解成“个秒”个体
        Gu8EditData_2=Gu32SetData_Countdown/100%10; //分解成“百毫秒”个体
        Gu8EditData_1=Gu32SetData_Countdown/10%10;  //分解成“十毫秒”个体

        Gu8RunStatus=0; //倒计时返回停止的状态

        Gu8RunStart=0;  //倒计时的运行步骤的停止
        Gu8RunStep=0;  //总运行步骤归零。建议跟 vGu8RunStart 成双成对出现
```

```
            Gu8Wd=1; //返回设置数据的窗口
            Gu8WdUpdate=1;  //整屏更新一次显示
        }

        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]
        if(0!=Gu8Part)  //处于“闪烁模式”的时候，是 [减键]
        {
            switch(Gu8Wd) //在某个窗口下
            {
                case 1:     //在窗口 1 下
                    switch(Gu8Part)  //二级支点的局部选择
                    {
                        case 1:  //局部 1 被选中，代表右起第 4 位数码管被选中。
                            if(Gu8EditData_4>0)
                            {
                                Gu8EditData_4--;  //编辑“十秒”个体的中间变量
                            }
                            PartUpdate(Gu8Part); //当前局部更新显示输出到数码管
                            break;

                        case 2:  //局部 2 被选中，代表右起第 3 位数码管被选中。
                            if(Gu8EditData_3>0)
                            {
                                Gu8EditData_3--;  //编辑“个秒”个体的中间变量
                            }
                            PartUpdate(Gu8Part); //当前局部更新显示输出到数码管
                            break;

                        case 3:  //局部 3 被选中，代表右起第 2 位数码管被选中。
                            if(Gu8EditData_2>0)
                            {
                                Gu8EditData_2--;  //编辑“百毫秒”个体的中间变量
                            }
                            PartUpdate(Gu8Part); //当前局部更新显示输出到数码管
                            break;
```

```
                        case 4:  //局部 4 被选中，代表右起第 1 位数码管被选中。
                              if(Gu8EditData_1>0)
                              {
                                  Gu8EditData_1--;  //编辑“十毫位”个体的中间变量
                              }
                              PartUpdate(Gu8Part); //当前局部更新显示输出到数码管
                              break;
                    }
                    break;
            }
        }
        else   //处于“没有闪烁”的时候，是 [开始健]
        {
            if(0==Gu8RunStatus) //在停止状态下
            {

                vGu8CountdownTimerFlag=0;
                vGu32CountdownTimerCnt=Gu32SetData_Countdown;   //从“设置值”开始倒计时
                vGu8CountdownTimerFlag=1;       //允许倒计时的软件定时器的启动

                Gu8RunStatus=1;  //倒计时处于工作状态（并且，这一瞬间才正式启动倒计时）

                Gu8RunStart=1;   //倒计时的运行步骤的总开关开启
                Gu8RunStep=0;    //总运行步骤归零。建议跟 vGu8RunStart 成双成对出现

                Gu8Wd=2; //进入倒计时运行的窗口
                Gu8WdUpdate=1;  //整屏更新一次显示，确保在启动的时候能显示到最新的数据
            }
        }

        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:     //在窗口 1 下
                if(0==Gu8Part)  //处于“没有闪烁”的时候，将进入“闪烁模式”
                {
```

```
                        Gu8EditData_4=Gu32SetData_Countdown/10000%10; //分解成“十秒”个体
                        Gu8EditData_3=Gu32SetData_Countdown/1000%10;  //分解成“个秒”个体
                        Gu8EditData_2=Gu32SetData_Countdown/100%10; //分解成“百毫秒”个体
                        Gu8EditData_1=Gu32SetData_Countdown/10%10;  //分解成“十毫秒”个体
                        Gu8Part=1;  //进入“闪烁模式”，从“局部 1”开始闪烁
                    }
                    else    //处于“闪烁模式”的时候，将退出到“没有闪烁”，隐含“确定”功能
                    {
                        //把个体合并还原成数据
                        Gu32SetData_Countdown=Gu8EditData_4*10000+Gu8EditData_3*1000;
                        Gu32SetData_Countdown=Gu32SetData_Countdown+Gu8EditData_2*100;
                        Gu32SetData_Countdown=Gu32SetData_Countdown+Gu8EditData_1*10;

                        Gu8Part=0;  //退出“闪烁模式”
                        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 显示函数。用来设置参数。
            break;
        case 2:
            Wd2();   //窗口 2 显示函数。倒计时的运行显示窗口。
            break;
    }

}
```

```
void Wd1(void)   //窗口 1 显示函数。用来设置参数。
{
    //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
          Gu8PartUpdate_2=1  ;//局部 2 更新显示
          Gu8PartUpdate_3=1  ;//局部 3 更新显示
          Gu8PartUpdate_4=1  ;//局部 4 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

          if(Gu32SetData_Countdown<10000)
          {
              Su8Temp_4=10;  //显示“无”
          }
          else
          {
              Su8Temp_4=Gu8EditData_4;  //显示“十秒”的临时中间个体，属于动态数据。
          }

          vGu8Display_Righ_4=Su8Temp_4;  //过渡需要显示的数据到底层驱动变量
    }

    if(1==Gu8PartUpdate_2) //局部 2 更新显示
    {
          Gu8PartUpdate_2=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_3=Gu8EditData_3;  //显示“个秒”的临时中间个体，属于动态数据。

        vGu8Display_Righ_3=Su8Temp_3;  //过渡需要显示的数据到底层驱动变量
}

if(1==Gu8PartUpdate_3) //局部 3 更新显示
{
        Gu8PartUpdate_3=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;  //显示“百毫秒”的临时中间个体，属于动态数据。

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

if(1==Gu8PartUpdate_4) //局部 4 更新显示
{
        Gu8PartUpdate_4=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_1=Gu8EditData_1;  //显示“十毫秒”的临时中间个体，属于动态数据。

        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;

    switch(Gu8Part)  //某个局部被选中，则闪烁跳动
    {
        case 1:
             if(0==Su8BlinkFlag)  //两种状态的切换判断
              {
                    Su8BlinkFlag=1;
                    Su8Temp_4=10;  //右起第 4 个显示“不显示”（10 代表不显示）
              }
              else
              {
                    Su8BlinkFlag=0;
                    Su8Temp_4=Gu8EditData_4;  //显示“十秒”的临时中间个体，属于动态数据。
              }
```

```
            break;

        case 2:
            if(0==Su8BlinkFlag)  //两种状态的切换判断
             {
                   Su8BlinkFlag=1;
                   Su8Temp_3=10;  //右起第 3 个显示“不显示”（10 代表不显示）
             }
             else
             {
                   Su8BlinkFlag=0;
                   Su8Temp_3=Gu8EditData_3;  //显示“个秒”的临时中间个体，属于动态数据。
             }

            break;

        case 3:
            if(0==Su8BlinkFlag)  //两种状态的切换判断
             {
                   Su8BlinkFlag=1;
                   Su8Temp_2=10;  //右起第 2 个显示“不显示”（10 代表不显示）
             }
             else
             {
                   Su8BlinkFlag=0;
                   Su8Temp_2=Gu8EditData_2;  //显示“百毫秒”的临时中间个体，属于动态数据。
             }

            break;

        case 4:
            if(0==Su8BlinkFlag)  //两种状态的切换判断
             {
                   Su8BlinkFlag=1;
                   Su8Temp_1=10;  //右起第 1 个显示“不显示”（10 代表不显示）
             }
             else
             {
                   Su8BlinkFlag=0;
                   Su8Temp_1=Gu8EditData_1;  //显示“十毫秒”的临时中间个体，属于动态数据。
             }

            break;
```

```
            default:   //都没有被选中的时候
                  if(Gu32SetData_Countdown<10000)
                  {
                     Su8Temp_4=10;  //显示“无”
                  }
                  else
                  {
                      Su8Temp_4=Gu8EditData_4;  //显示“十秒”的临时中间个体，属于动态数据。
                  }
                 Su8Temp_3=Gu8EditData_3;  //显示“个秒”的临时中间个体，属于动态数据。
                 Su8Temp_2=Gu8EditData_2;  //显示“百毫秒”的临时中间个体，属于动态数据。
                 Su8Temp_1=Gu8EditData_1;  //显示“十毫秒”的临时中间个体，属于动态数据。
                 break;
        }

        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

    }
}

void Wd2(void)   //窗口 2 显示函数。倒计时的运行显示窗口。
{
    //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

    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

          //先分解数据，注意，这里分解的时候，“先整除后求余”必须用一行代码一气呵成，不能拆
          //分成两行代码，否则会有隐患会有 bug。除非，把四个临时变都改成 unsigned long 类型。

          //Su8Temp_4 提取“十秒”位。
          Su8Temp_4=vGu32CountdownTimerCnt/10000%10; //实际精度是 0.001 秒,但显示精度是 0.01 秒

         //Su8Temp_3 提取“个秒”位。
          Su8Temp_3=vGu32CountdownTimerCnt/1000%10; //实际精度是 0.001 秒,但显示精度是 0.01 秒

         //Su8Temp_2 提取“百毫秒”位。
          Su8Temp_2=vGu32CountdownTimerCnt/100%10; //实际精度是 0.001 秒,但显示精度是 0.01 秒
```

```
        //Su8Temp_1 提取“十毫秒”位。
        Su8Temp_1=vGu32CountdownTimerCnt/10%10; //实际精度是 0.001 秒,但显示精度是 0.01 秒

        //判断数据范围，来决定最高位数码管是否需要显示。
        if(vGu32CountdownTimerCnt<10000) //10.000 秒。实际 4 位数码管最大只能显示 99.99 秒
        {
           Su8Temp_4=10;  //在数码管转换表里，10 代表一个“不显示”的数据
        }

        //上面先分解数据之后，再过渡需要显示的数据到底层驱动变量里，让过渡的时间越短越好
        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;
        vGu8Display_Righ_2=Su8Temp_2;
        vGu8Display_Righ_1=Su8Temp_1;

        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;

    }
}

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
    }

    //每 10ms 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
    }

    //倒计时实际走的时间的软件定时器，注意，这里还附加了启动状态的条件“&&1==Gu8RunStatus”
    if(1==vGu8CountdownTimerFlag&&vGu32CountdownTimerCnt>0&&1==Gu8RunStatus)
    {
        vGu32CountdownTimerCnt--;  //递减式的软件定时器
    }

    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;

    //上电初始化开机显示的窗口
    Gu8EditData_4=Gu32SetData_Countdown/10000%10; //分解成“十秒”个体
    Gu8EditData_3=Gu32SetData_Countdown/1000%10;  //分解成“个秒”个体
```

```
    Gu8EditData_2=Gu32SetData_Countdown/100%10; //分解成“百毫秒”个体
    Gu8EditData_1=Gu32SetData_Countdown/10%10;  //分解成“十毫秒”个体
    Gu8Wd=1; //返回设置数据的窗口
    Gu8WdUpdate=1;  //整屏更新一次显示

}

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

void PeripheralInitial(void)
{

}
```