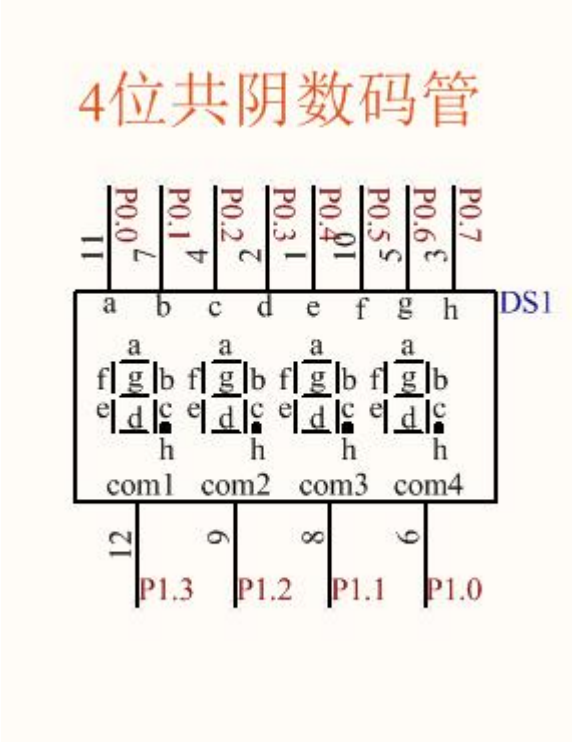
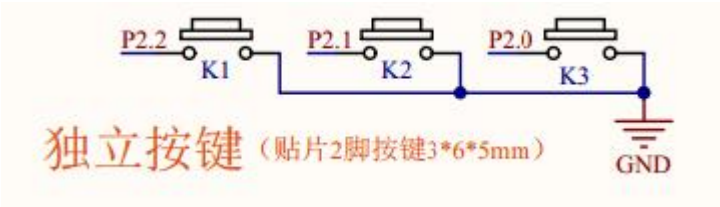
**第一百二十节： 按键切换窗口切换局部来设置参数。**

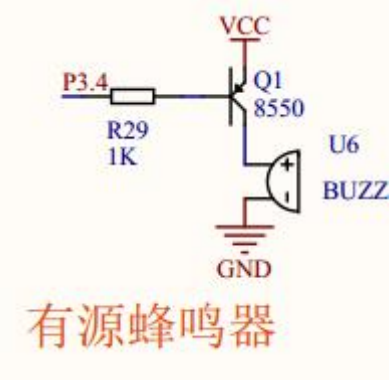
**【120.1 按键切换窗口切换局部来设置参数。】**




上图 120.1.1 数码管




上图 120.1.2 独立按键




上图 120.1.3 有源蜂鸣器

为了更好理解上一节提出的人机界面程序框架的脉络，本节程序恰好包含了整屏更新与局部更新的应用，同时也引入了一个新的知识点：在人机界面的程序框架中，常常会遇到需要以“位”来编辑某个数据的情况，这种情况实际上是先把“待编辑数据”分解成几个“位”中间临时个体，然后显示并且编辑这些“位”中间临时个体，编辑结束后，再把这些“位”中间临时个体合并成一个完整的数据赋值给“待编辑数据”。

本节程序功能如下：

（1）有 3 个窗口 1-XX，2-YY，3-ZZ，其中 XX，YY，ZZ 分别代表 3 个可编辑的数据 Gu8SetDate_1，Gu8SetDate_2，Gu8SetDate_3。数据范围是从 0 到 99。

（2）K1 按键。含“短按”与“长按”复合双功能。当数码管“没有闪烁”时，“短按”K1 按键可以切换窗口，而“长按”K1 按键会使数码管从“没有闪烁”进入到“闪烁模式”。当数码管处于“闪烁模式”时，“短按”K1 可以使数码管在十位和个位之间切换“闪烁”的“局部位”，而“长按”K1 表示更改完毕当前窗口数据并从“闪烁模式”退出到“没有闪烁”。

（3）K2 按键。当数码管处于“闪烁模式”时，每按一次 K2 按键就可以使当前闪烁的某位数码管“递增 1”。

（4）K3 按键。当数码管处于“闪烁模式”时，每按一次 K2 按键就可以使当前闪烁的某位数码管“递减 1”。

上述功能，在窗口切换和退出“闪烁模式”时用到整屏更新，在闪烁的某位数码管切换“局部”时用到局部更新。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 Wd1(void);   //窗口 1 显示函数
void Wd2(void);   //窗口 2 显示函数
void Wd3(void);   //窗口 3 显示函数

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;

unsigned char Gu8SetData_3=0; //单片机内部第 3 个可编辑的参数，在窗口 3
unsigned char Gu8SetData_2=0; //单片机内部第 2 个可编辑的参数，在窗口 2
unsigned char Gu8SetData_1=0; //单片机内部第 1 个可编辑的参数，在窗口 1
```

```
/* 注释一：
*      在人机界面的程序框架中，常常会遇到需要以“位”来编辑某个数据的情况，这种情况
*  实际上是先把“待编辑数据”分解成几个“位”临时中间个体，然后显示并且编辑这些“位”
*  临时中间个体，编辑结束后，再把这些“位”临时中间个体合并成一个完整的数据赋值给
*  “待编辑数据”。以下 Gu8EditData_2 和 Gu8EditData_1 就是“位”临时中间个体的中间变量。
*/

unsigned char Gu8EditData_2=0;  //对应显示左起第 3 位数码管的“位”数据，是中间变量。
unsigned char Gu8EditData_1=0;  //对应显示左起第 4 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


volatile unsigned char vGu8Display_Righ_4=1;   //左起第 1 位初始化显示窗口“1”
volatile unsigned char vGu8Display_Righ_3=11;  //左起第 2 位初始化显示横杠“-”
volatile unsigned char vGu8Display_Righ_2=0;   //左起第 3 位初始化显示数值“0”
volatile unsigned char vGu8Display_Righ_1=0;   //左起第 4 位初始化显示数值“0”

//不显示小数点
volatile unsigned char vGu8Display_Righ_Dot_4=0;
volatile unsigned char vGu8Display_Righ_Dot_3=0;
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 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;
    }

}

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

    switch(vGu8KeySec)
    {
       case 1:     //K1 按键的“短按”，具有“切换窗口”和“切换局部”的双功能。
            if(0==Gu8Part)  //处于“没有闪烁”的时候，是“切换窗口”
            {
                switch(Gu8Wd) //在某个窗口下
                {
                    case 1:     //在窗口 1 下
                          Gu8Wd=2;  //切换到窗口 2
                          Gu8EditData_2=Gu8SetData_2/10%10;  //“待编辑数据”分解成中间个体
                          Gu8EditData_1=Gu8SetData_2/1%10;   //“待编辑数据”分解成中间个体
                          Gu8WdUpdate=1;  //整屏更新
                          break;

                    case 2:     //在窗口 2 下
                          Gu8Wd=3;  //切换到窗口 3
                          Gu8EditData_2=Gu8SetData_3/10%10;  //“待编辑数据”分解成中间个体
                          Gu8EditData_1=Gu8SetData_3/1%10;   //“待编辑数据”分解成中间个体
                          Gu8WdUpdate=1;  //整屏更新
                          break;

                    case 3:     //在窗口 3 下
                          Gu8Wd=1;  //切换到窗口 1
                          Gu8EditData_2=Gu8SetData_1/10%10;  //“待编辑数据”分解成中间个体
                          Gu8EditData_1=Gu8SetData_1/1%10;   //“待编辑数据”分解成中间个体
```

```
                    Gu8WdUpdate=1;  //整屏更新
                    break;

            }
        }
        else    //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>2)
            {
                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;

        vGu8KeySec=0;
        break;

    case 2:     //递增按键 K2
        switch(Gu8Wd) //在某个窗口下
        {
           case 1:     //在窗口 1 下
           case 2:     //在窗口 2 下，窗口 2 与窗口 1 的代码完全一模一样，因此可以这样共享
           case 3:     //在窗口 3 下，窗口 3 与窗口 1 的代码完全一模一样，因此可以这样共享
                switch(Gu8Part)  //二级支点的局部选择
                {
                    case 1:  //局部 1 被选中，代表左起第 3 位数码管被选中。
                          Gu8EditData_2++;  //编辑“十位”个体的中间变量
                          if(Gu8EditData_2>9)
                          {
                              Gu8EditData_2=9;
                          }
                          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，代表左起第 4 位数码管被选中。
                          Gu8EditData_1++;  //编辑“个位”个体的中间变量
                          if(Gu8EditData_1>9)
                          {
```

```
                            Gu8EditData_1=9;
                        }
                        PartUpdate(Gu8Part); //当前局部更新显示输出到数码管
                        break;
                }
                break;
        }

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

        vGu8KeySec=0;
        break;

    case 3:     //递减按键 K3
        switch(Gu8Wd) //在某个窗口下
        {
           case 1:     //在窗口 1 下
           case 2:     //在窗口 2 下，窗口 2 与窗口 1 的代码完全一模一样，因此可以这样共享
           case 3:     //在窗口 3 下，窗口 3 与窗口 1 的代码完全一模一样，因此可以这样共享
                switch(Gu8Part)  //二级支点的局部选择
                {
                    case 1:  //局部 1 被选中，代表左起第 3 位数码管被选中。
                        if(Gu8EditData_2>0)
                        {
                            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 2:  //局部 2 被选中，代表左起第 4 位数码管被选中。
                        if(Gu8EditData_1>0)
                        {
                            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;
        }

        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_2=Gu8SetData_1/10%10;  //先把“待编辑数据”分解成中间个体
                        Gu8EditData_1=Gu8SetData_1/1%10;   //先把“待编辑数据”分解成中间个体
                        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
                    {
                        Gu8SetData_1=Gu8EditData_2*10+Gu8EditData_1; //把个体合并还原成数据
                        Gu8Part=0;  //退出“闪烁模式”
                        Gu8WdUpdate=1;  //整屏更新
                    }
                    break;

               case 2:     //在窗口 2 下
                    if(0==Gu8Part)  //处于“没有闪烁”的时候，将进入“闪烁模式”
                    {
                        Gu8EditData_2=Gu8SetData_2/10%10;  //先把“待编辑数据”分解成中间个体
                        Gu8EditData_1=Gu8SetData_2/1%10;   //先把“待编辑数据”分解成中间个体
                        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
                    {
                        Gu8SetData_2=Gu8EditData_2*10+Gu8EditData_1; //把个体合并还原成数据
                        Gu8Part=0;  //退出“闪烁模式”
                        Gu8WdUpdate=1;  //整屏更新
                    }
                    break;

               case 3:     //在窗口 3 下
                    if(0==Gu8Part)  //处于“没有闪烁”的时候，将进入“闪烁模式”
                    {
                        Gu8EditData_2=Gu8SetData_3/10%10;  //先把“待编辑数据”分解成中间个体
                        Gu8EditData_1=Gu8SetData_3/1%10;   //先把“待编辑数据”分解成中间个体
```

```
                        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
                    {
                        Gu8SetData_3=Gu8EditData_2*10+Gu8EditData_1; //把个体合并还原成数据
                        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
            break;
        case 2:
            Wd2();   //窗口 2 显示函数
            break;
        case 3:
            Wd3();   //窗口 3 显示函数
            break;
    }

}

void Wd1(void)   //窗口 1 显示函数
{
    //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

      Su8Temp_4=1;   //左起第 1 位数码管，显示窗口“1”，属于静态数据，起“装饰”作用。
      Su8Temp_3=11;  //左起第 2 位数码管，显示横杠“-”，属于静态数据，起“装饰”作用。

      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

      //不显示任何一个小数点，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=0;
      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
}

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;  //显示“十位”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_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_1=Gu8EditData_1;  //显示“个位”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_2=10;  //左起第 3 个显示“不显示”（10 代表不显示）
                  }
                  else
                  {
                        Su8BlinkFlag=0;
                        Su8Temp_2=Gu8EditData_2;  //显示“十位”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_1=10;  //左起第 4 个显示“不显示”（10 代表不显示）
                  }
                  else
                  {
                        Su8BlinkFlag=0;
                        Su8Temp_1=Gu8EditData_1;  //显示“个位”的临时中间个体，属于动态数据。
                  }

                 break;

            default:   //都没有被选中的时候
                 Su8Temp_2=Gu8EditData_2;  //显示“十位”的临时中间个体，属于动态数据。
                 Su8Temp_1=Gu8EditData_1;  //显示“个位”的临时中间个体，属于动态数据。
                 break;
        }

        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
{
    //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

          Su8Temp_4=2;   //左起第 1 位数码管，显示窗口“2”，属于静态数据，起“装饰”作用。
          Su8Temp_3=11;  //左起第 2 位数码管，显示横杠“-”，属于静态数据，起“装饰”作用。

          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

          //不显示任何一个小数点，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=0;
          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
    }

    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;  //显示“十位”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_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_1=Gu8EditData_1;  //显示“个位”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_2=10;  //左起第 3 个显示“不显示”（10 代表不显示）
              }
              else
              {
                    Su8BlinkFlag=0;
                    Su8Temp_2=Gu8EditData_2;  //显示“十位”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_1=10;  //左起第 4 个显示“不显示”（10 代表不显示）
              }
              else
              {
                    Su8BlinkFlag=0;
                    Su8Temp_1=Gu8EditData_1;  //显示“个位”的临时中间个体，属于动态数据。
              }

             break;

        default:   //都没有被选中的时候
             Su8Temp_2=Gu8EditData_2;  //显示“十位”的临时中间个体，属于动态数据。
             Su8Temp_1=Gu8EditData_1;  //显示“个位”的临时中间个体，属于动态数据。
             break;
    }
```

```
        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 显示函数
{
    //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

          Su8Temp_4=3;   //左起第 1 位数码管，显示窗口“3”，属于静态数据，起“装饰”作用。
          Su8Temp_3=11;  //左起第 2 位数码管，显示横杠“-”，属于静态数据，起“装饰”作用。

          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

          //不显示任何一个小数点，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=0;
          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
    }

    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;  //显示“十位”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_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_1=Gu8EditData_1;  //显示“个位”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_2=10;  //左起第 3 个显示“不显示”（10 代表不显示）
              }
              else
              {
                    Su8BlinkFlag=0;
                    Su8Temp_2=Gu8EditData_2;  //显示“十位”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_1=10;  //左起第 4 个显示“不显示”（10 代表不显示）
              }
              else
              {
                    Su8BlinkFlag=0;
                    Su8Temp_1=Gu8EditData_1;  //显示“个位”的临时中间个体，属于动态数据。
              }

             break;
```

```
            default:   //都没有被选中的时候
                 Su8Temp_2=Gu8EditData_2;  //显示“十位”的临时中间个体，属于动态数据。
                 Su8Temp_1=Gu8EditData_1;  //显示“个位”的临时中间个体，属于动态数据。
                 break;
        }

        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
```

```
    }

    TH0=0xfc;
    TL0=0x66;
}


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

    TMOD=0x01;
    TH0=0xfc;
    TL0=0x66;
    EA=1;
    ET0=1;
    TR0=1;
}

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

void PeripheralInitial(void)
{

}
```