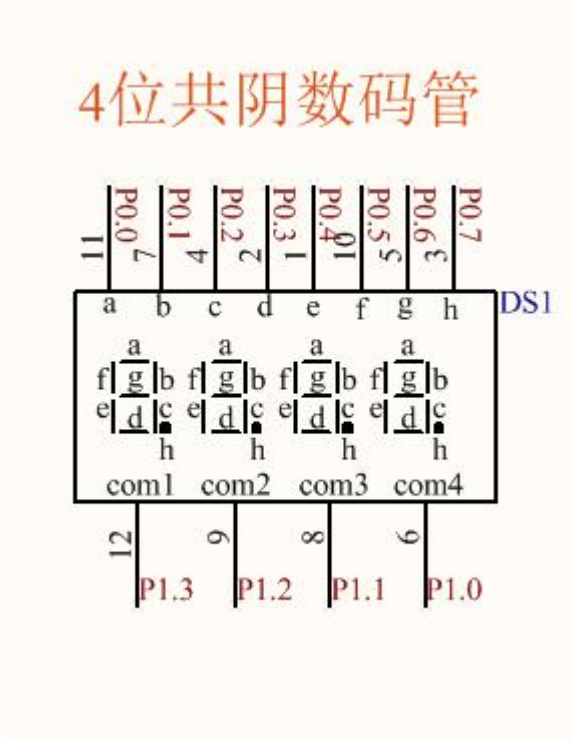
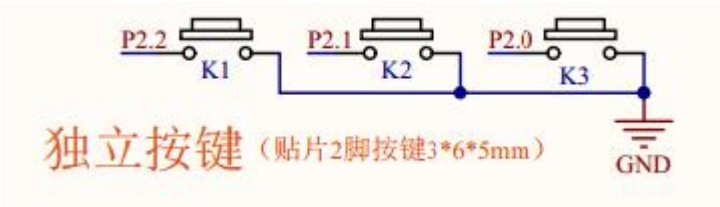
第一百一十六节： 按键控制数码管的倒计时。

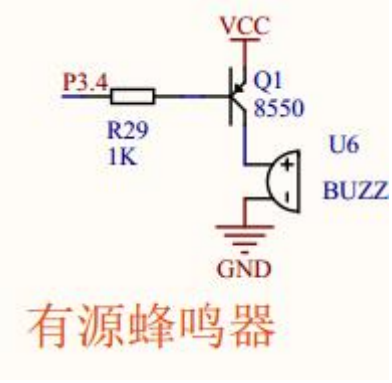
【116.1　按键控制数码管的倒计时。】



上图 116.1.1　数码管



上图 116.1.2　独立按键



上图 116.1.3　有源蜂鸣器

上一节讲“累加式”的秒表，这一节讲“递减式”的倒计时。通过此小项目，加深理解在显示框架中常用的更新变量（整屏更新和局部更新），以及应用程序与按键是如何关联的框架。同时，在拆分“个十百千万...”的时候，有一处地方必须注意，“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“先整除；后求余”，否则会有隐患会有 bug。除非，把四个临时变都改成 unsigned long 类型。上一节在此处残留了一个 bug 我发帖后还来不及修改过来，敬请大家注意。比如：

以下这样是对的：

```
static unsigned char Su8Temp_1;
Su8Temp_1=vGu32CountdownTimerCnt/10%10; //一气呵成，没毛病。
```

以下这样是有隐患有 bug 的（除非把 Su8Temp_1 改成 unsigned long 类型）：

```
static unsigned char Su8Temp_1;
Su8Temp_1=vGu32CountdownTimerCnt/10;
Su8Temp_1=Su8Temp_1%10; //拆分成两行代码后，有隐患有 bug。数据溢出的原因引起。
```

本节倒计时程序的功能：K1 按键是复位按键，每按一次，倒计时停止并且重新恢复初始值 10.00 秒。K2 按键是启动按键，当秒表处于复位后停止的状态时按一次则开始启动倒计时，当倒计时变为 0.00 秒的时候，蜂鸣器发出一次“滴”的提示声。此时，如果想启动新一轮的倒计时，只需按一次 K1 复位键，然后再按 K2 启动按键，就可以启动新一轮 10.00 秒的倒计时。代码如下：

```
#include "REG52.H"

#define KEY_FILTER_TIME  25

#define SCAN_TIME  1

#define VOICE_TIME   50     //蜂鸣器一次“滴”的声音长度 50ms

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

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

void VoiceScan(void);  //蜂鸣器的驱动函数
void DisplayScan(void);  //底层显示的驱动函数
void DisplayTask(void);  //上层显示的任务函数

void RunTask(void);  //倒计时的应用程序

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

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

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
};

//数码管底层驱动扫描的软件定时器
volatile unsigned char vGu8ScanTimerFlag=0;
volatile unsigned int vGu16ScanTimerCnt=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;  //开机默认显示初始值 10.000 秒

//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;

//蜂鸣器的软件定时器
volatile unsigned char vGu8BeepTimerFlag=0;
volatile unsigned int vGu16BeepTimerCnt=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 Gu8WdUpdate=1;  //开机默认整屏更新一次显示。此变量在显示框架中是非常重要的变量

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

    switch(vGu8KeySec)
    {
       case 1:     //复位按键

            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

            vGu8CountdownTimerFlag=0;    //禁止倒计时开始(而 Gu8RunStart 是启动开关，不矛盾)
            vGu32CountdownTimerCnt=10000;   //倒计时的软件定时器恢复初始值 10.000 秒

            Gu8WdUpdate=1;  //整屏更新一次显示

            vGu8KeySec=0;
            break;

        case 2:     //启动的按键
            if(0==Gu8RunStatus) //在停止状态下
            {

                vGu8CountdownTimerFlag=0;
                vGu32CountdownTimerCnt=10000;  //初始值是 10.000 秒
                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

                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
            }


            vGu8KeySec=0;
            break;

    }
}

void DisplayTask(void) //数码管显示的上层任务函数
{
    //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

    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 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 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 Su8KeyLock1;
   static unsigned int  Su16KeyCnt1;
   static unsigned char Su8KeyLock2;
   static unsigned int  Su16KeyCnt2;

   if(0!=KEY_INPUT1)
   {
      Su8KeyLock1=0;
      Su16KeyCnt1=0;
   }
   else if(0==Su8KeyLock1)
   {
      Su16KeyCnt1++;
      if(Su16KeyCnt1>=KEY_FILTER_TIME)
      {
         Su8KeyLock1=1;
         vGu8KeySec=1;
      }
   }

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

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
    }

    //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=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)
{

}
```