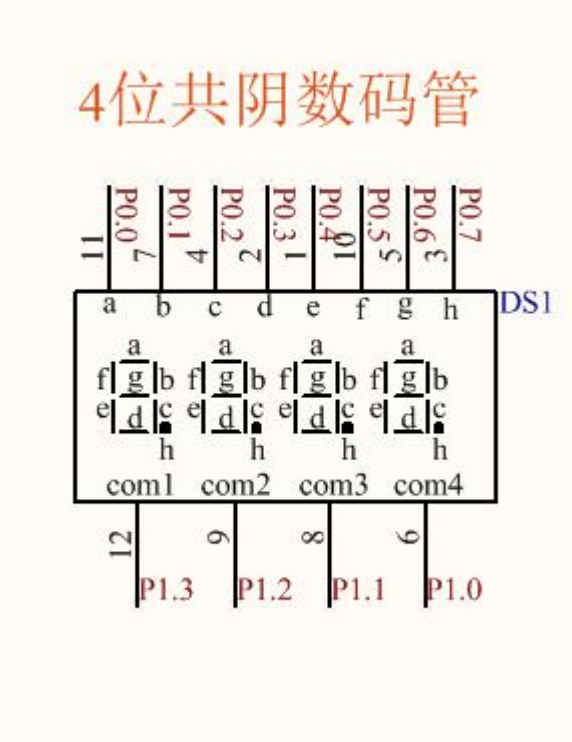
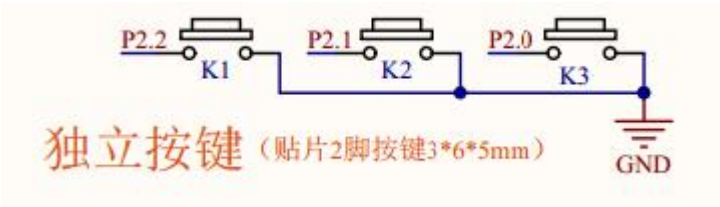
第一百一十八节： 按键让某位数码管闪烁跳动来设置参数。

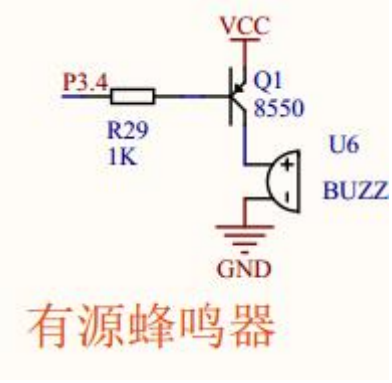
【118.1　按键让某位数码管闪烁跳动来设置参数。】



上图 118.1.1　数码管



上图 118.1.2　独立按键



上图 118.1.3　有源蜂鸣器

当一个窗口只有一个数据的时候，只需以“窗口”为支点，切换到某个窗口下去设置某个数据即可。但是，当某个窗口有几个数据时，就必须在以“窗口”为支点的前提下，再细分出一个二级的支点，这个二级支点就是“局部”（或者称为子菜单）。“窗口”对应一个“窗口选择”的全局变量 Gu8Wd，“局部”对应一个“局部选择”的全局变量 Gu8Part。数据需要更新显示输出到屏幕（数码管）时，有两种更新方式，一种是“整屏更新”，另一种是“局部更新”。“整屏更新”只有一个整屏的更新变量 Gu8WdUpdate，而“局部更新”有 N 个更新变量 Gu8PartUpdate_x（Gu8PartUpdate_1，Gu8PartUpdate_2，Gu8PartUpdate_3），一个窗口下有多少个数据就存在多少个局部的更新变量 Gu8PartUpdate_x，这些局部的更新变量在不同的窗口下是可以共用的。当某个局部被选中的时候，可以有很多种表现方式，比如在液晶屏上，常见的有光标跳动，某行文字的底色变色（反显），本节例程用的数码管，当某个局部被选中的时候，用某位数码管闪烁跳动的方式。

本节小项目的程序功能，在一个窗口下，对单片机内部四个参数 Gu8SetData_4，Gu8SetData_3，Gu8SetData_2，Gu8SetData_1 进行编辑。这四个参数的范围是从 0 到 9，从左到右分别显示在四位数码管上，每一位数码管对应一个数据。比如左起第 1 位是 Gu8SetData_4，左起第 2 位是 Gu8SetData_3，左起第 3 位是 Gu8SetData_2，左起第 4 位是 Gu8SetData_1。K1 是局部选择的切换按键，每按一次，数码管从左到右，依次闪烁跳动，表示某个数据被选中。K2 是数字累加按键，每按一次，闪烁跳动的数字会累加 1。K3 是数字递减按键，每按一次，闪烁跳动的数字会递减 1。代码如下：

```
#include "REG52.H"

#define KEY_FILTER_TIME  25
#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 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_4=0; //单片机内部第 4 个可编辑的参数
unsigned char Gu8SetData_3=0; //单片机内部第 3 个可编辑的参数
unsigned char Gu8SetData_2=0; //单片机内部第 2 个可编辑的参数
unsigned char Gu8SetData_1=0; //单片机内部第 1 个可编辑的参数

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=0;   //左起第 1 位初始化显示数值“0”
volatile unsigned char vGu8Display_Righ_3=0;   //左起第 2 位初始化显示数值“0”
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”

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;
       case 3:
            Gu8PartUpdate_3=1;
            break;
       case 4:
            Gu8PartUpdate_4=1;
```

```
            break;
    }

}

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

    switch(vGu8KeySec)
    {
       case 1:     //局部切换的按键
            switch(Gu8Wd) //在某个窗口下
            {
               case 1:     //在窗口 1 下
                    //以下之所以有两个 PartUpdate(Gu8Part)，是因为相邻的两个局部发生了变化。

                    PartUpdate(Gu8Part);  //切换之前的局部进行更新。
                    Gu8Part++;  //切换到下一个局部
                    if(Gu8Part>4)
                    {
                        Gu8Part=0;
                    }
                    PartUpdate(Gu8Part);  //切换之后的局部进行更新。
                    break;
            }

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

            vGu8KeySec=0;
            break;

       case 2:     //累加的按键
            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，代表左起第 1 位数据 Gu8SetData_4 被选中。
```

```
                    Gu8SetData_4++;
                    if(Gu8SetData_4>9)
                    {
                        Gu8SetData_4=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 2 位数据 Gu8SetData_3 被选中。
                    Gu8SetData_3++;
                    if(Gu8SetData_3>9)
                    {
                        Gu8SetData_3=9;
                    }
                    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，代表左起第 3 位数据 Gu8SetData_2 被选中。
                    Gu8SetData_2++;
                    if(Gu8SetData_2>9)
                    {
                        Gu8SetData_2=9;
                    }
                    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，代表左起第 4 位数据 Gu8SetData_1 被选中。
                    Gu8SetData_1++;
                    if(Gu8SetData_1>9)
                    {
                        Gu8SetData_1=9;
                    }
                    PartUpdate(Gu8Part); //当前局部更新显示输出到数码管
                    break;

            }
            break;

        case 2:     //在窗口 2 下（本节只用到窗口 1）
            break;
    }

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

```
        vGu8BeepTimerFlag=1;

        vGu8KeySec=0;
        break;

    case 3:     //递减的按键
        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，代表左起第 1 位数据 Gu8SetData_4 被选中。
                          if(Gu8SetData_4>0)
                          {
                              Gu8SetData_4--;
                          }
                          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 2 位数据 Gu8SetData_3 被选中。
                          if(Gu8SetData_3>0)
                          {
                              Gu8SetData_3--;
                          }
                          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，代表左起第 3 位数据 Gu8SetData_2 被选中。
                          if(Gu8SetData_2>0)
                          {
                              Gu8SetData_2--;
                          }
                          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，代表左起第 4 位数据 Gu8SetData_1 被选中。
                          if(Gu8SetData_1>0)
                          {
                              Gu8SetData_1--;
                          }
                          PartUpdate(Gu8Part); //当前局部更新显示输出到数码管
                          break;

                }
```

```
                    break;

                case 2:     //在窗口 2 下（本节只用到窗口 1）
                    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:      //窗口 2 显示选择（本节只用到窗口 1）
            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

          //不显示任何一个小数点
          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
        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

        Su8Temp_4=Gu8SetData_4;  //显示左起第 1 个数据 Gu8SetData_4

        //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
}

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=Gu8SetData_3;  //显示左起第 2 个数据 Gu8SetData_3

        //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_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=Gu8SetData_2;  //显示左起第 3 个数据 Gu8SetData_2

        //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_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=Gu8SetData_1;  //显示左起第 4 个数据 Gu8SetData_1
```

```
        //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_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;  //左起第 1 个显示“不显示”（10 代表不显示）
              }
              else
              {
                    Su8BlinkFlag=0;
                    Su8Temp_4=Gu8SetData_4;  //左起第 1 个显示数据 Gu8SetData_4
              }

             break;

        case 2:
             if(0==Su8BlinkFlag)  //两种状态的切换判断
              {
                    Su8BlinkFlag=1;
                    Su8Temp_3=10;  //左起第 2 个显示“不显示”（10 代表不显示）
              }
              else
              {
                    Su8BlinkFlag=0;
                    Su8Temp_3=Gu8SetData_3;  //左起第 2 个显示数据 Gu8SetData_3
              }

             break;

        case 3:
             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=Gu8SetData_2;  //左起第 3 个显示数据 Gu8SetData_2
                }

                break;

            case 4:
                if(0==Su8BlinkFlag)  //两种状态的切换判断
                {
                    Su8BlinkFlag=1;
                    Su8Temp_1=10;  //左起第 3 个显示“不显示”（10 代表不显示）
                }
                else
                {
                    Su8BlinkFlag=0;
                    Su8Temp_1=Gu8SetData_1;  //左起第 4 个显示数据 Gu8SetData_1
                }

                break;
            default:   //都没有被选中的时候
                Su8Temp_4=Gu8SetData_4;  //左起第 1 个显示数据 Gu8SetData_4
                Su8Temp_3=Gu8SetData_3;  //左起第 2 个显示数据 Gu8SetData_3
                Su8Temp_2=Gu8SetData_2;  //左起第 3 个显示数据 Gu8SetData_2
                Su8Temp_1=Gu8SetData_1;  //左起第 4 个显示数据 Gu8SetData_1
                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 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;
 static unsigned char Su8KeyLock3;
static unsigned int  Su16KeyCnt3;

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;
   }
}

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)
{
```

```
}
```