http://www.Microcontrol.cn 微控设计网

中国 MSP430 单片机专业网站

MSP430 C 语言例题

由微控技术论坛会员 Slam 提供

基于模拟前端信号处理与控制技术的专业论坛、网站

本章选择了一些简单的 C 语言程序例题，这些程序的结构简单，编程技巧不多，题目虽然简单，但是非常适合入门单片机的学习者学习 MSP430 单片机的 C 语言编程。

如下列出了 C 语言例题运行的 MSP430F149 实验板硬件资源环境，熟悉这些硬件资源，对于理解程序非常重要。

（1）数码管：

左侧数码管与 P5 口相连，a~g，h 对应 P5.0~P5.7

右侧数码管与 P4 口相连，a~g，h 对应 P4.0~P4.7

（2）发光二极管

8 个发光二极管与 P3 口连接

（3）按钮：

左侧 8 个按钮与 P2 口相连，引脚号标在按钮上方

右侧 8 个按钮与 P1 口相连，引脚号标在按钮上方

（4）P2.3 引脚还是模拟比较器输入

（5）P6.0，P6.1 引脚连接模拟量电位器，用于模拟量实验

9.1 通过 C 语言编程例入门 MSP430C 语言编程

如下例子都在 MSP430F149 实验板上通过验证。

例 1：使与 P3 口的 P3.0 引脚连接的发光二极管闪烁。

```
#include <msp430x14x.h>  //声明库
void main(void)  //主函数
{
unsigned int i;  //变量声明
WDTCTL=WDTPW+WDTHOLD; //关掉看门狗
P3DIR |=BIT0; //设置P3.0为输出，这里BIT0=0x0001
 while(1)  //无限次while循环
 {
 for (i=0;i<20000;i++) //for语句，i为循环变量，i每次循环加1，当i<20000时，
                       //循环延时
 P3OUT=0x00; 使P3.0输出低电平，发光二极管亮，（低电平使发光二极管亮）
 for (i=0;i<20000;i++) //再次循环延时
 P3OUT=0x01; 使P3.0输出高电平，发光二极管灭，（高电平使发光二极管灭）
 }
}
```

例 2：8 个发光二极管 1、3、5、7 与 2、4、6、8 交替发光的例子

```
#include <msp430x14x.h>
void main(void)
{
unsigned int i;
WDTCTL=WDTPW+WDTHOLD;
P3DIR=0XFF; //设置P3口为输出
 while(1)
 {
 for (i=0;i<20000;i++)
 P3OUT=0X55; //使发光二极管1、3、5、7 灭，2、4、6、8亮
 for (i=0;i<20000;i++)
 P3OUT=0XAA;//使发光二极管1、3、5、7亮，2、4、6、8灭
 }
}
```

例 3：定时器控制的发光二极管闪烁。这里使用了 MSP430F149 芯片的 32768Hz 低频晶体振荡器作为时钟源。用定时器 A 定时 1s，发光二极管灭 0.5s，亮.0.5s。

```
#include <msp430x14x.h>
void main (void)
{
WDTCTL= WDTPW + WDTHOLD; //设置看门狗控制寄存器，关看门狗
TACTL = TASSEL0 + TACLR; // 设置定时器A控制寄存器，
                         // TASSEL0=0x0100，选择辅助时钟ACLK，
                         // TACLR=0x0004，清除定时器A计数器
CCTL0 = CCIE;  //设置捕获/比较控制寄存器，CCIE=0x0010，使能捕获比较中断
CCR0 =16384;   //设置捕获/比较寄存器，初始值为16384，对于32768Hz的频率，相当于0.5s
P3DIR |=BIT7; //P3.7为输出
TACTL |= MC0; //设置定时器A控制寄存器，MC0=0x0010，使计数模式为增计数
_EINT();   //使能中断，这是一个C编译器支持的内部过程。
while(1); //无限次while循环
}

interrupt[TIMERA0_VECTOR] void Timer_A (void) //定时器A的CC0中断处理程序
                      //TIMERA0_VECTOR=6*2,等于基地址0xFFE0+12=0xFFEC
{
    P3OUT ^= BIT7; //将P3.7引脚取反，就是使发光二极管闪烁
}
```

例 4：选择不同的时钟源，使 P3.7 连接的发光二极管闪烁。

（1）使用 XT2 时钟源，8MHz 频率，用定时器 A 分频，产生 1s 脉冲，使 P3.7 引脚的发光二极管闪烁。

```
#include <msp430x14x.h>
#define XT0FF 0x40;
void main (void)
{
WDTCTL= WDTPW + WDTHOLD; //关闭看门狗
BCSCTL1 &= ~XT2OFF;   //基础时钟控制寄存器BCSCTL1的第7位置0，使XT2启动
BCSCTL2 = SELS + DIVS1 + DIVS0; //基础时钟控制寄存器BCSCTL2设置，第3位置1，选择
                    //XT2CLK作为SMCLK时钟；将第2和第1位置1，使分频比为8
TACTL =0x02D4;
            //定时器A控制寄存器设置，第2位置1：清除；第4、5位置1、0：加计数模式
            //加计数至CCR0,然后重新开始；第6、7位1、1，所以是8分频；第8、9位是
                //0、1，所以TA使用SMCLK时钟。
CCTL0 = CCIE;  //CCIE=0x0010，使能定时器A中断
CCR0 =62500;  //设置计数器CCR0的初值，（（8MHz/8）/8）/2=62500，相当于0.5s的时间
P3DIR |=BIT7; //将P3.7设置为输出
_EINT();  //调用C430编译器内部函数，使能中断
while(1); //无限次循环
}
interrupt[TIMERA0_VECTOR] void Timer_A (void)  //定时器A中断函数
{
    P3OUT ^= BIT7; //P3.7位取反
}
```

（2）使用32768Hz晶体产生1s信号的程序如下：

```
#include <msp430x14x.h>
void main (void)
{
WDTCTL= WDTPW + WDTHOLD;
TACTL =TASSEL0+TACLR+MC0;
CCTL0 = CCIE;
CCR0 =16384;
P3DIR |=BIT7;
_EINT();
while(1);
```

```
}
interrupt[TIMERA0_VECTOR] void Timer_A (void)
{
      P3OUT ^= BIT7;
}
```

（3）看门狗使输出 P3.7 引脚连接的发光二极管每秒闪烁一次的例子：

```
#include <msp430x14x.h>
void main (void)
{
WDTCTL= WDTPW + WDTTMSEL+WDTSSEL;
IE1|=WDTIE;
P3DIR |=BIT7;
_EINT();
while(1);
}
interrupt[WDT_VECTOR] void WDT_interrupt (void)
{
      P3OUT ^= BIT7;
}
```

例 5： P4 和 P5 输出口连接的数码管显示 1 和 2。

```
#include <msp430x14x.h>
 void main(void)
 {
   unsigned char seg[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
                          //定义七段译码的共阳数码管显示数组
                          //  hgfg dcba
                          //0=1100 0000
                          //1=1111 1001
                          //2=1010 0100
                          //……
                          //9=1001 0000
   WDTCTL=WDTPW+WDTHOLD; //关闭看门狗，以便于调试
   P4DIR=0XFF; //设置P4口为输出
   P5DIR=0XFF;//设置P5口为输出
   P4OUT=seg[1];//向P4口输出数组的第1个元素，数字1的段码
   P5OUT=seg[2];//向P5口输出数组的第2个元素，数字2的段码
 }
```

例 6：与 P5 口连接的数码管加 1 计数，与 P4 口相连的数码管显示数字 8。

```
 #include <msp430x14x.h>
 void main(void)
 {
   int i,x; //声明数据类型
   unsigned char seg[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
   WDTCTL=WDTPW+WDTHOLD;//关看门狗
   P4DIR=0XFF; //P4口为输出，连接有共阳极数码管
   P5DIR=0XFF; //P5口为输出，连接有共阳极数码管
   P4OUT=seg[8]; //P4输出数字8
   P5OUT=seg[0]; //P5输出数字0
   while(1)  //无限次While循环
   {
   for(i=0;i<=9;i++) //循环变量I从0到9循环
   for(x=0;x<20000;x++) //没有循环体的for循环，用于延迟时间
   P5OUT=seg[i]; //按照循环变量i的数值，取出相应的数组元素
   }
 }
```

例7：使用定时器输出精确的秒信号。从0开始计时，数码管显示0～60秒，每隔10秒使数码管更换显示，并顺序点亮发光二极管。

基于模拟前端信号处理与控制技术的专业论坛、网站

```
#include <msp430x14x.h>
#define XTOFF 0x40;
unsigned int i=0,j=0; //声明数据类型
unsigned char seg_7[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
                      //数码管字型码数组
unsigned int bit[8]={0x0001,0x0002,0x0004,0x0008,0x0010,0x0020,0x0040,0x0080};
                      //发光二极管点亮顺序数组
void main (void)
{
WDTCTL= WDTPW + WDTHOLD; //关看门狗
TACTL = TASSEL0 + TACLR; // 设置定时器A控制寄存器，
                         // TASSEL0=0x0100，选择辅助时钟ACLK（32kHz）
                         // TACLR=0x0004，清除定时器A计数器
CCTL0 = CCIE; //使能定时器A捕捉与中断功能，CCIE=0x0010
CCR0 =32768;  // 设置计数器CCR0初值
TACTL |= MC0; //设置定时器工作模式为加计数到CCR0初值
P3DIR = 0XFF; //P3口为输出
P4DIR = 0XFF; //P4口为输出
P5DIR = 0XFF; //P5口为输出
P3OUT = 0X7E; //P3口输出为0111 1110
_EINT();  调用C430编译器内部函数使能中断
while(1); //没有循环体的无限次while循环
}
interrupt[TIMERA0_VECTOR] void Timer_A (void) //定时器A的中断函数
{
i+=1;  i每次循环加1
if (i==10) //如果i=1
      {
      i=0;   //使i=0
      j+=1; j每次加1
      P3OUT ^= bit[j]; //数组的第j个元素取反后从P3口输出，使发光二极管顺序点亮
        if (j==6)  //如果j=6
          {
            j=0; 使j=0
            }
        }
  P4OUT =seg_7[i];//数码管字型数组中取第i个元素，送到P4口输出
  P5OUT = seg_7[j]; //数码管字型数组中取第j个元素，送到P5口输出
}
```

例 8：连接在 P1.0 口的按键控制数码管显示数值，数码管显示按动次数。

```
#include  <msp430x14x.h> //声明库文件
char Key_Pressed(void); //声明被调用函数
void main(void)
{
  unsigned char seg[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
                      //共阳数码管字型码数组
  unsigned int i=0; //声明数据类型
  WDTCTL = WDTPW + WDTHOLD; //关看门狗
  P1DIR &= ~BIT0; //P1.0引脚设置成输入，该引脚连接的按键按下时，按键输出低电平
  P4DIR =0xff;    //P4口设置为输出，连接共阳数码管
  P4OUT=seg[8];  //P4口输出数字8

while (1) //无限次while循环
  {
  if(Key_Pressed()) //调用按键函数，如果按键函数返回1，
```

```
  P4OUT=seg[i++]; // 则数码管字型数组下标加1，选择相应的七段字型从
                  // P4口输出
  if (i>9) //如果i大于9，则使i=0
  i=0;
  }
}
char Key_Pressed(void) //按键函数
  {
    unsigned int i;  //声明变量i
    while(!(P1IN&BIT0)); //当P1输入寄存器P1IN的第0位为0时，开始while循环
    for(i=0;i<8000;i++);//延时一段时间，消除按键抖动
    if (!(P1IN&BIT0)) //如果P1输入寄存器P1IN的第0位还是0，则返回1，表示按键按下
       return 1;
    else             //否则认为按键未按下，返回0
        return 0;
 }
```

例9：将P6口输入的模拟电压AD转换后，从P4、P5口连接的数码管输出。

使用AD单通道多次转换，采集P6.0输入的模拟电压值（变化范围：0～3.3V），转换为数字量。建立二维数组和通过顺序查表的方法得出采集回来的电压值。然后通过数码管显示当前电压值，显示跟随输入的模拟电压的变化。由于只有两位数码管，故显示电压值精确到小数点后一位，如当前输入电压2.37V，则显示2.4V。可用万用表检测显示是否准确。

```
#include  "msp430x14x.h"       声明库
void Init(void);//声明初始化函数
interrupt [ADC_VECTOR]void ADC12(void); //声明AD转换中断函数
unsigned int Result;  声明变量
unsigned int Table[4][10] = { {0x040,0x0BC,0x138,0x1B4,0x230, 0x2AC , 0x328 , 0x3A4 , 0x420 ,0x49C },
                  { 0x518 , 0x594 , 0x610 , 0x68C , 0x708 , 0x784 , 0x800, 0x87C , 0x8F8,0x974   },
                  { 0x9F0 , 0xA6C , 0xAE8 , 0xB64 , 0xBE0 , 0xC5C , 0xC08 , 0xD54 , 0xDD0,0xE4C },
                  { 0xEC8 , 0xF44 , 0xFC0 , 0xFFF } };
     //该数组元素用于与AD转换的电压数值相比较，如果某个数组元素稍大于等于AD转换后的电压数
     //值，则将此元素输出
void main(void)  //主函数
{  P4DIR = 0XFF; //P4口设置为输出
   P5DIR = 0XFF; //P5口设置为输出
   Init();       //调用初始化函数

   _EINT();     //使能中断
   ADC12CTL0 |= ENC+ADC12SC;  //设置转换控制寄存器ADC12CTL0，ENC=0x002使转换允许位为1，
                              //意味着可以启动转换，同时ADC12TL0中的低电平位可以被修改。
                             //ADC12SC=0x001使采样/转换控制位为1，如果采样信号SAMPCON由
                             //采样定时器产生（SHP=1），则ASC12SC=1将产生一次转换
   while (1);   //无限次的while循环
}
void Init(void)  //初始化函数
{
  WDTCTL = WDTPW+WDTHOLD;     //关看门狗
  P6SEL |= 0x01;              // 设置P6口的P6.0引脚为外围模块AD转换器的模拟信号输入引脚
  ADC12CTL0 &= ~ENC;          //复位转换允许位
  ADC12CTL0 = ADC12ON + SHT0_2 + REFON + REF2_5V;    // Turn on and set up ADC12
        //设置转换控制寄存器ADC12CTL0，ADC12ON=0x010，使ADC12内核工作
        //SHT0_2=2*0x100，确定采样周期为4×tADC12CLK×4
        //REFON=0x020，内部参考电压打开
        //REF2_5V=0x040，选择内部参考电压发生器的电压为2.5V
  ADC12CTL1 = SHP + CONSEQ_2 ;  // 设置AD转换控制寄存器ADC12CTL1
```

```
                    // SHP=0x0200 设置SAMPON来自采样定时器，采样信号上升沿触发采样
                    //CONSEQ_2=2*2 设置工作模式为单通道、多次转换模式
  ADC12MCTL0 = SREF_0;  //设置通道0的转换存储控制寄存器ADC12MCTL0，
                        //SREF_0=0*0x10 选择参考电压为VR+=AVCC,VR-=AVSS，因此输入模拟信号
                        //范围是3.3V~0V。
  ADC12IE |= BIT0;      //设置中断允许寄存器ADC12IE，将第0位置1，使通道A0转换后产生中断
}
interrupt[ADC_VECTOR] void ADC12 (void) //AD转换中断函数
{
  unsigned char seg_7[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
        //声明无小数点显示的数码管七段字型码数组
  unsigned char seg_8[10]={0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x10};
       //声明有小数点显示的数码管七段字型码数组
  unsigned i,j;         //声明变量数据类型
  ADC12CTL0 &= ~ENC;  //设置AD转换控制寄存器ADC12CTL0，ENC=0x002，~ENC=0xFFD，停止AD转换
  for ( i=0 ; i<4 ;i++) //扫描 Table 数组行下标
     {
        for (j=0; j<10; j++)            //扫描 Table 数组列下标
          { if (ADC12MEM0<=Table[i][j])
             goto xxx;                  //如果 Table 数组元素大于转换数值，则转到标号 xxx
             }
      }
    xxx:    {P4OUT = seg_7[j];   //P4 口输出
             P5OUT = seg_8[i]; }    //P5 口输出
        ADC12CTL0 |= ENC+ADC12SC;   // 使能再次转换
}
```

例10： 模拟比较器实验

接电位器于端口**P2.3**，用来输入模拟电压值（**0**～**3.3V**）。参考电压选取**0.5VCC**，待测电压由**P2.3**端输入，如果待测电压大于参考电压，**P1.0**端口的**LED**点亮，反之熄灭。

注意**:**顺时针调节电位器，输入的模拟电压值增大。

```
#include <msp430x14x.h>
void main (void)
{
 WDTCTL=WDTPW+WDTHOLD;
P3DIR |= BIT0;   //P3口的第0引脚为输入
CACTL1 =CARSEL + CAREF1 + CAON;//设置控制寄存器CACTL1，
       //CARSEL=0x40，设置内部参考电压，当CAEX=0时参考电平加在（－）端
       //CAREF1=0x20，选择0.5×VCC作为参考电压
       //CAON=0x08，  打开比较器
CACTL2 = P2CA0; //设置控制寄存器CACTL2，
              //P2CA0=0x04，设置外部引脚信号连接在比较器输入端
while (1) //无限次循环
{
    if ((CACTL2 & CAOUT )== CAOUT) //CAOUT=0x01，如果比较器输出为1
                                   //若CACTL2寄存器的第0位为1，则表示输入电压
                                   //大于参考电压
    P3OUT &= ~BIT0;      //则P3的第0引脚输出低电平，相连的发光二极管亮
  else
    P3OUT |= BIT0;      //否则，P3的第0引脚输出高电平，相连的发光二极管灭
 }
}
```

例11： MSPF149的UART向PC机的RS232串口发送字符串。

单片机**UART**以**9600**波特率，**8**个数据位，无校验位，**1**个停止位。单片机上电后连续向**PC**发送字符串，利用串口调试助手可以显示发送的内容。需要发送其他英文会话可以改变**Data[]**数组内容。

```
#include <msp430x14x.h>
void Init(void);//声明初始化函数
char Data[20]="xia lao shi ni hao!"; //发送的字符串
void main(void)
{
  unsigned int i;
  WDTCTL = WDTPW + WDTHOLD;
  Init();  //调用初始化函数
 while(1)  //无限次循环
 {
  for(i=0;i<=20;i++)
  {
  TXBUF0=Data[i]; //向缓冲器送入待发送数据
  while((UTCTL0&0x01)==0); //发送缓冲器有待发数据时，UTCTL0的第0位复位，进入等待
  }
 }
}
void Init(void)
{
  UCTL0 &= ~SWRST; //USART控制寄存器UCTL0，SWRST=0x01，~SWRST=0xFE，将
                   //UCTL0寄存器的第0位复位后，USART才能重新被允许
  UCTL0 =0X10; //UCTL0的第4位置1，设置数据长度为8位，第5位为0，设置1位停止位
  UBR00 = 0x03; //使用32768Hz晶体，波特率为9600
  UBR10 = 0x00;
  UMCTL0 = 0x4A;
  UTCTL0 = 0X10; //发送控制寄存器，第4位置1，选择辅助时钟ACLK1
  ME1 |= UTXE0 ; //设置模块允许寄存器ME1，UTXE0=0x80，设置ME1的第7位为1，
                 //使USART模式发送允许
  P3SEL|=BIT4;   //P3口选择寄存器的第4位置1，选择外围模块
  P3DIR|=BIT4;  //P3口方向寄存器的第4位置1，选择输出
}
```

例 12： MSP430F149 的 USART 接受 PC 键盘输入的数值并显示。

在串口调试助手中的发送区选中：“十六进制发送”和“自动发送”，以十六进制形式,以字节为单位输入某个数字，如03，这样单片机会接受到，送到数码管显示该数字。若连续输入“030205”，单片机其实是接受到了来自PC的以9600波特率，8个数据位，一个停止位的无校验位的字符串，显示器瞬间显示了3、2、5，看到的是最后的数字5。

```
#include <msp430x14x.h>
void Init(void);
void main (void)
{
  P4DIR=0XFF; //P4口为输出
  WDTCTL=WDTPW+WDTHOLD;
  Init();
  _EINT();
  while(1); //无限次循环，等待接收中断
}
void Init(void)  //初始化函数
  {
    UCTL0 &=~SWRST;
    UCTL0 |=CHAR;   //8位数据，1位停止位
    UBR00 = 0X03;//9600波特率，32kHz时钟
    UBR01 = 0X00;
    UMCTL0 =0X4A;
```

```
    UCTL0 |= SSEL0;  //SSEL0=0x10，选择辅助时钟ACLK
     ME1 |= UTXE0+URXE0;//模块允许寄存器ME1
                        //UTXE0=0x80 发送允许
                        //URXE0=0x40 接收允许
    P3SEL |= BIT4+BIT5; //P3口第4、5引脚供外围模块使用
    P3DIR |=BIT4+BIT5; //P3口的第4、5位为输出
    IE1 |=URXIE0; //中断允许寄存器IE1，第6位为1，使能接收中断，URXIE0=0x40
   }
interrupt [UARTORX_VECTOR] void UARTORX (void) //接收中断函数
   {
    unsigned char seg[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
    unsigned int x=0;
        {
         x=RXBUF0;   //将接收缓冲器的内容送x，
         P4OUT=seg[x];  //再送P4口显示
         }
    }
```

例 13： RS232 串口通信接收发送数字

发送：从单片机 P1、P2 口输入按键信号， PC 机 RS232C 口接收。

接收：PC 机由 RS232C 口发送数字时，P4 口连接的数码管显示。需要发送十六进制数，例如，十进制数 3，应该发送 03。

```
#include <msp430x14x.h>
void Init(void);//声明初始化函数
void Delay(void);声明延迟函数
void main(void)  //主函数
{
  P1DIR=0X00; //设置P1口方向寄存器P1口作为输入
  P1IES=0X00; //设置P1口中断边沿选择寄存器，置1为下跳沿，置0为上跳沿
  P1IE=0XFF; //设置P1中断使能寄存器，置1为允许中断，置0为禁止中断
  P2DIR=0X00;//设置P2口方向寄存器，置0为输入，置1为输出
  P2IES=0X00;//设置P2口中断边沿选择寄存器，置1为下跳沿，置0为上跳沿
  P2IE=0XFF; //设置P2中断使能寄存器，置1为允许中断，置0为禁止中断
  P4DIR=0XFF; //设置P4口方向寄存器，使P4口为输出
WDTCTL = WDTPW + WDTHOLD; //关看门狗
  Init(); //调用初始化函数
  _EINT(); //调用C编译器内部函数使能中断
  _BIS_SR(LPM1_bits); //调用C编译器内部对状态寄存器某位置位的函数，
                      //LPM_bits=SCG0+CPUOFF，
                      // SCG0=0x0040，进入LPM1低功耗工作模式
                      // CPUOFF=0x0010 关闭CPU，唤醒所有允许的中断
  _NOP(); //调用C编译器内部空操作函数
}
void Init(void) //初始化函数
{
  UCTL0 &= ~SWRST;  //USART控制寄存器UCTL0，SWRST=0x01，~SWRST=0xFE，将
                 //UCTL0寄存器的第0位复位后，USART才能重新被允许
  UCTL0 =0X10; //8位数据，1位停止位
  UBR00 = 0x03; //9600波特率，32kHz时钟
  UBR10 = 0x00;
  UMCTL0 = 0x4A;
  UTCTL0 = 0X10; //发送控制寄存器，第4位置1，选择辅助时钟ACLK1
  ME1 |= UTXE0+URXE0;  //模块允许寄存器ME1
                       //UTXE0=0x80 发送允许
                       //URXE0=0x40 接收允许
  P3SEL |= BIT4+BIT5; //P3口第4、5引脚供外围模块使用
```

```
  P3DIR |=BIT4+BIT5; //P3口的第4、5位为输出
  IE1 |=URXIE0; //中断允许寄存器IE1，第6位为1，使能接收中断，URXIE0=0x40
}
interrupt[PORT1_VECTOR]void PORT1(void)  //P1口中断函数
{
      if(P1IFG&BIT0) //如果中断标志寄存器的第0位为1，则延迟一段时间
      {Delay(); //调用延迟函数
      if(P1IFG&BIT0)//若如果中断标志寄存器的第0位还为1
      {TXBUF0=0X30; //向USART的发送缓冲器送数字“0”
      P1IFG&=~BIT0;} //清除中断标志
      }              //如下部分只是向USART发送缓冲器所送数字不同
     else
       if(P1IFG&BIT1)
      {Delay();
      if(P1IFG&BIT1)
      {TXBUF0=0X31; P1IFG&=~BIT1;}}
      else
       if(P1IFG&BIT2)
      {Delay();
      if(P1IFG&BIT2)
      {TXBUF0=0X32; P1IFG&=~BIT2;}}
      else
       if(P1IFG&BIT3)
      {Delay();
      if(P1IFG&BIT3)
      {TXBUF0=0X33; P1IFG&=~BIT3;} }
      else
       if(P1IFG&BIT4)
      {Delay();
      if(P1IFG&BIT4)
      {TXBUF0=0X34; P1IFG&=~BIT4;}}
      else
       if(P1IFG&BIT5)
      {Delay();
      if(P1IFG&BIT5)
      {TXBUF0=0X35; P1IFG&=~BIT5;}}
      else
       if(P1IFG&BIT6)
      {Delay();
      if(P1IFG&BIT6)
      {TXBUF0=0X36; P1IFG&=~BIT6;}}
      else
       if(P1IFG&BIT7)
      {Delay();
      if(P1IFG&BIT7)
      {TXBUF0=0X30;  P1IFG&=~BIT7;}}
     }
interrupt[PORT2_VECTOR]void PORT2(void) //P2口中断函数
{
      if(P2IFG&BIT0)
      {Delay();
      if(P2IFG&BIT0)
      {TXBUF0=0X37;
      P2IFG&=~BIT0;}
      }
```

```
      else
        if(P2IFG&BIT1)
      {Delay();
      if(P2IFG&BIT1)
      {TXBUF0=0X38; P2IFG&=~BIT1;}}
      else
        if(P2IFG&BIT2)
      {Delay();
      if(P2IFG&BIT2)
      {TXBUF0=0X39; P2IFG&=~BIT2;}}
      else
       if(P2IFG&BIT3)
      {Delay();
      if(P2IFG&BIT3)
      {TXBUF0=0X30; P2IFG&=~BIT3;}}
      else
       if(P2IFG&BIT4)
      {Delay();
      if(P2IFG&BIT4)
      {TXBUF0=0X30; P2IFG&=~BIT4;}}
      else
       if(P2IFG&BIT5)
      {Delay();
      if(P2IFG&BIT5)
      {TXBUF0=0X30; P2IFG&=~BIT5; }}
      else
       if(P2IFG&BIT6)
      {Delay();
      if(P2IFG&BIT6)
      {TXBUF0=0X30; P2IFG&=~BIT6;} }
      else
       if(P2IFG&BIT7)
      {Delay();
      if(P2IFG&BIT7)
      {TXBUF0=0X30; P2IFG&=~BIT7;} }
    }
void Delay(void) //延迟函数
{
    unsigned long i;
    for(i=500;i>0;i--);
}
interrupt [UARTORX_VECTOR] void UARTORX (void)  //接收中断
{
   unsigned char seg[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
   unsigned int x=0;
         x=RXBUF0;  //将接收缓冲器的内容赋予x
         P4OUT=seg[x]; //送P4口显示，需要PC机发送十六进制数，
   }                   //例如数字3，必须输入03
                       //若要十进制，需要改成P4OUT=seg[x-0x30]
```

9.2 简单控制类单片机程序

例 1： 定时控制程序

说明：彩灯类程序例

```
#include <msp430x14x.h>
interrupt[TIMERA0_VECTOR] void Timer_A (void);
unsigned data1;
void main (void)
{
unsigned x1,y1;
unsigned char seg[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
WDTCTL= WDTPW + WDTHOLD; //设置看门狗控制寄存器，关看门狗
TACTL = TASSEL0 + TACLR; // 设置定时器 A 控制寄存器，
                         // TASSEL0=0x0100，选择辅助时钟 ACLK，
                         // TACLR=0x0004，清除定时器 A 计数器
CCTL0 = CCIE;   //设置捕获/比较控制寄存器，CCIE=0x0010，使能捕获比较中断
CCR0 =16384;    //设置捕获/比较寄存器，初始值为 16384，对于 32768Hz 的频率，相当于 0.5s
P3DIR =0XFF; //P3 为输出
TACTL |= MC0; //设置定时器 A 控制寄存器，MC0=0x0010，使计数模式为增计数
_EINT();    //使能中断，这是一个 C 编译器支持的内部过程。
//p4 初始化
P4DIR=0XFF;
P5DIR=0XFF;
while(1) //无限次 while 循环
  {
  x1=data1%10;  //个位
  y1=data1/10;  //十位
  P4OUT=seg[x1]; //显示个位
  P5OUT=seg[y1]; //显示十位
//如下是彩灯变化部分，可以按照 data1 的值，设定多种显示模式
   if( data1==0)
      P3OUT=0XFF;
   if(data1==1)
      P3OUT=0X00;
   if( data1==2)
      P3OUT=0X55;
   }
}
//定时中断
interrupt[TIMERA0_VECTOR] void Timer_A (void) //定时器 A 的 CC0 中断处理程序
                          //TIMERA0_VECTOR=6*2,等于基地址 0xFFE0+12=0xFFEC
{
     //P3OUT ^= BIT7; //将 P3.7 引脚取反，就是使发光二极管闪烁
     data1++;
     if (data1>=3)
     data1=0;
}
```

例 2. 按键中断显示程序

说明：P1、2 口按键中断后，P4、5 口输出按键值 keyvaluep1、keyvaluep2 显示数字的例子。

```
#include <msp430x14x.h>
void delay(int v);
unsigned keyvaluep1;
unsigned keyvaluep2;
//unsigned v;
void main (void)
{
unsigned                                                                    char
seg[16]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90,0x88,0x83,0xC6,0xA1,0x86,0x8E};
WDTCTL= WDTPW + WDTHOLD; //设置看门狗控制寄存器，关看门狗
//p3 初始化
P3DIR =0XFF; //P3 为输出
//p4 初始化
```

```
P4DIR=0XFF; //P4 为输出
P5DIR=0XFF; //P5 为输出
        P1DIR=0x00;                         // P1 口为输入
        P2DIR=0x00;                         // P2     口为输入
        P1IFG=0x00;                         //清除 P1 口的中断标志
        P2IFG=0x00;                         //清除 P2 口的中断标志
        P1IES=0xff;                         //设置 P1 口中断是下降沿触发
        P2IES=0xff;                         //设置 P2 口中断是下降沿触发
        P1IE=0xff;                          //允许 P1 口中断
        P2IE=0xff;                          //允许 P2 口中断
_EINT();    //使能中断，这是一个 C 编译器支持的内部过程。
while(1) //无限次 while 循环
  {
  P4OUT=seg[keyvaluep1]; //显示个位
  P5OUT=seg[keyvaluep2]; //显示十位

   }
}
      void delay(int v)                 //延时子程序
      {
        while(v!=0)
        v--;
      }
 //以下是 port1 的中断服务程序
interrupt[PORT1_VECTOR] void PORT1 (void)
   {
      unsigned temp1;      //局部变量：?                          //temp 暂时存放端口的中断标志寄存器
中的值
                           //temp 暂时存放端口的中断标志寄存器中的值
        delay(2666);                        //  消除抖动延时
        if ((P1IN&0xff)!=0xff)              //如果有键按下
        {
              temp1=P1IFG;                  //temp1 记录中断标志
              switch(temp1)
              { case 1: keyvaluep1=0;break;
                case 2: keyvaluep1=1;break;
                case 4: keyvaluep1=2;break;
                case 8: keyvaluep1=3;break;
                case 16: keyvaluep1=4;break;
                case 32: keyvaluep1=5;break;
                case 64: keyvaluep1=6;break;
                case 128: keyvaluep1=7;break;
                //default: keyvaluep1=0;break;
              }
         }
        P1IFG=0X00;                                 //清除中断标志，返回主程序
   }
//以下是 port2 的中断服务程序
interrupt[PORT2_VECTOR] void PORT2 (void)
{
     unsigned char temp1;
        delay(2666);
        if ((P2IN&0xff)!=0xff)
        {
              temp1=P2IFG;
             switch(temp1)
              { case 1: keyvaluep2=8;break;
                case 2: keyvaluep2=9;break;
                case 4: keyvaluep2=10;break;
```

```
                    case 8: keyvaluep2=11;break;
                    case 16: keyvaluep2=12;break;
                    case 32: keyvaluep2=13;break;
                    case 64: keyvaluep2=14;break;
                    case 128: keyvaluep2=15;break;
                    //default: keyvaluep2=0;break;
                  }
            }
      P2IFG=0X00;
}
```

例 3. 90 延时开关程序

说明：当按键 P1.0 第一次按下时，P3.0 和 P3.1 连接的发光二极管亮，当第二次按下时，一个发光管灭，90 秒后全灭。P4、5 口输出时间值。

```
#include <msp430x14x.h>
interrupt[TIMERA0_VECTOR] void Timer_A (void);
interrupt[PORT1_VECTOR] void PORT1 (void) ;
void delay(int v);                    //延时子程序
unsigned data1;
unsigned keyvaluep1;
void main (void)
{
unsigned state;
unsigned x1,y1;
unsigned char seg[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
WDTCTL= WDTPW + WDTHOLD; //设置看门狗控制寄存器，关看门狗
TACTL = TASSEL0 + TACLR; //  设置定时器 A 控制寄存器，
                         // TASSEL0=0x0100，选择辅助时钟 ACLK，
                         // TACLR=0x0004，清除定时器 A 计数器
CCTL0 = CCIE;   //设置捕获/比较控制寄存器，CCIE=0x0010，使能捕获比较中断
CCR0 =16384;    //设置捕获/比较寄存器，初始值为 16384，对于 32768Hz 的频率，相当于 0.5s
TACTL |= MC0; //设置定时器 A 控制寄存器，MC0=0x0010，使计数模式为增计数
_EINT();    //使能中断，这是一个 C 编译器支持的内部过程。
    P3DIR =0XFF; //P3 为输出
    P4DIR=0XFF;//P4 为输出
    P5DIR=0XFF;//P5 为输出
 P1DIR=0x00;                          // P1 口为输入
       P2DIR=0x00;                     // P2      口为输入
       P1IFG=0x00;                     //清除 P1 口的中断标志
       P2IFG=0x00;                     //清除 P2 口的中断标志
       P1IES=0xff;                     //设置 P1 口中断是下降沿触发
       P2IES=0xff;                     //设置 P2 口中断是下降沿触发
       P1IE=0xff;                      //允许 P1 口中断
       P2IE=0xff;                      //允许 P2 口中断
state=0;
keyvaluep1=7;
   P3OUT=0XFF;
while(1) //无限次 while 循环
   {
   x1=data1%10;   //个位
   y1=data1/10;   //十位
   P4OUT=seg[x1]; //显示个位
   P5OUT=seg[y1]; //显示十位
    if( state==0 && keyvaluep1==0 )
       {P3OUT=0XFC;
       state=1;
       keyvaluep1=7;}
    if(state==1 && keyvaluep1==0)
       {P3OUT=0XFE;
```


基于模拟前端信号处理与控制技术的专业论坛、网站

```
        state=2;
        keyvaluep1=7;
        data1=0;}
    if( state==2 && data1==15)//data1 应该为 90s
        {P3OUT=0XFF;
        state=0;
        keyvaluep1=7;
    }
}
interrupt[TIMERA0_VECTOR] void Timer_A (void) //定时器 A 的 CC0 中断处理程序
                            //TIMERA0_VECTOR=6*2,等于基地址 0xFFE0+12=0xFFEC
{
//P3OUT ^= BIT7; //将 P3.7 引脚取反，就是使发光二极管闪烁
data1++;
if (data1>=91)
data1=0;
}

void delay(int v)                   //延时子程序
      {
        while(v!=0)
        v--;
      }
 //以下是 port1 的中断服务程序
interrupt[PORT1_VECTOR] void PORT1 (void)
    {
      unsigned temp1;      //局部变量：//temp 暂时存放端口的中断标志寄存器中的值
        delay(2666);                        //  消除抖动延时
        if ((P1IN&0xff)!=0xff)              //如果有键按下
        {
              temp1=P1IFG;                  //temp1 记录中断标志
              switch(temp1)
              { case 1: keyvaluep1=0;break;
                case 2: keyvaluep1=1;break;
                case 4: keyvaluep1=2;break;
                case 8: keyvaluep1=3;break;
                case 16: keyvaluep1=4;break;
                case 32: keyvaluep1=5;break;
                case 64: keyvaluep1=6;break;
                case 128: keyvaluep1=7;break;
                //default: keyvaluep1=0;break;
              }
         }
        P1IFG=0X00;                         //清除中断标志，返回主程序
    }
```

例 4. 乡村小路交叉大路的交通灯程序

说明：小路有传感器 p1.0，有车时，两路按照时间交换红绿灯，无车时，大路一直绿灯，小路绿灯时，若无车立即向小路红灯、大路绿灯转换。

```
#include <msp430x14x.h>
interrupt[TIMERA0_VECTOR] void Timer_A (void);
void Key(void);
void delay(int v);                  //延时子程序
unsigned data1;//全局变量，用于传递时间信值
unsigned data2; //全局变量，用于传递计满数就停止计数的时间值
unsigned kk;
void main (void)
{
unsigned state;
unsigned x1,y1;
```


基于模拟前端信号处理与控制技术的专业论坛、网站

```
unsigned char seg[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
WDTCTL= WDTPW + WDTHOLD; //设置看门狗控制寄存器，关看门狗
TACTL = TASSEL0 + TACLR; // 设置定时器 A 控制寄存器，
                         // TASSEL0=0x0100，选择辅助时钟 ACLK，
                         // TACLR=0x0004，清除定时器 A 计数器
CCTL0 = CCIE;  //设置捕获/比较控制寄存器，CCIE=0x0010，使能捕获比较中断
CCR0 =16384;   //设置捕获/比较寄存器，初始值为 16384，对于 32768Hz 的频率，相当于 0.5s

TACTL |= MC0; //设置定时器 A 控制寄存器，MC0=0x0010，使计数模式为增计数
_EINT();   //使能中断，这是一个 C 编译器支持的内部过程。
    P3DIR =0XFF; //P3 为输出
    P4DIR=0XFF;//P4 为输出
    P5DIR=0XFF;//P5 为输出
  P1DIR=0x00;                          // P1 口为输入
  state=0;
  P3OUT=0XFF;
  data1=0;
  kk=1;
while(1) //无限次 while 循环
  {
  x1=data1%10;  //个位
  y1=state;//P1IN;//data1/10;  //十位
  P4OUT=seg[x1]; //显示个位
  P5OUT=seg[y1]; //显示十位
  Key();
   if( state==0 && data1==4 )
         {P3OUT=0X7D;  //主路红，小路绿
          state=1;
          data1=0;
          data2=0;}
    if(state==1 && data2==7 && kk==1)
       {P3OUT=0X7B;     //主路红，小路黄
       state=2;
       data1=0;}
    if( state==2 && data1==4 )
       {P3OUT=0XD7;     //主路绿，小路黄
       state=3;
       data1=0;}
    if(state==3 && (data1==9 || kk==0))
       {P3OUT=0XB7;
       state=0;
       data1=0;}  //0 状态，主路黄，小路红

    }
}

interrupt[TIMERA0_VECTOR] void Timer_A (void) //定时器 A 的 CC0 中断处理程序
                          //TIMERA0_VECTOR=6*2,等于基地址 0xFFE0+12=0xFFEC
{
//P3OUT ^= BIT7; //将 P3.7 引脚取反，就是使发光二极管闪烁
data1++;
if (data1>=91)
data1=0;
data2++;          //产生一个大于 7 就等于 7 的计数值
if (data2>=7)
data2=7;
}
void delay(int v)                  //键盘判断延时子程序
      {
```


基于模拟前端信号处理与控制技术的专业论坛、网站

```
        while(v!=0)
        v--;
      }
 void Key(void)                  //接 P1.0 的按键函数
{
  if(P1IN==0xFE)//如果按键按下
 {
     delay(2666);
     if(P1IN ==0xFE)     //如果按键按下
          kk=0;
  }
 else
     kk=1;
 }
```

例 5. 两路模拟量 AD 转换

说明：按钮 P1.0 控制交替显示，P1.1 显示 P6.0 输入的模拟电压，P1.2 显示 P6.1 输入的模拟电压。

```
#include<msp430x14x.h>
unsigned int Volt0,Volt1=0;                    //设置电压变量
unsigned long int Volttem0,Volttem1;
unsigned   data0=0,data1=0,kk0=0,kk1=0,counter=0,keyvaluep1;
unsigned int ADresult0,ADresult1; //设置 A/D 转换结果变量
void Adcvolt (void);
void main(void)
{
unsigned int a[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};//数码管不带小数点译码
unsigned int b[10]={0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x10};//数码管带小数点译码
WDTCTL=WDTPW+WDTHOLD;          //停看门狗
BCSCTL1&=~XT2OFF;               //开启 XT2CLK 振荡器，ACLK=32768Hz
BCSCTL2|=SELM_2+SELS;          //主时钟 MCLK 选择 8MHz 时钟，子时钟 SMCLK=8MHz
P1DIR=0x00;                     //P1 口为输入
P1IFG=0x00;                     //清除 P1 口的中断标志
P1IES=0xff;                     //设置 P1 口中断是下降沿触发
P1IE=0xff;                      //允许 P1 口中断
P4DIR=0xFF;                      //P4 口外接数码管
P4OUT=0xFF;                      //熄灭 P4 口数码管
P5DIR=0xFF;                      //P5 口外接数码管
P5OUT=0xFF;                      //熄灭 P5 口数码管
P3DIR=0XFF;
P6SEL|=BIT0+BIT1;               //P6.0，P6.1 用于模拟输入通道
TACTL=TASSEL0+TACLR;            //TIMERA 初始化，时钟源为 ACLK=32768
CCTL0=CCIE;                      //TIMERA 中断使能
CCR0=2048;                      //设置比较值，定时 1S
TACTL|=MC0;                      //TIMERA 增计数模式，同时启动 TIMERA
Adcvolt ();                     //AD 初始化
_EINT();                         //总中断使能
while(1)
 {
 if(kk0==1)            //按键 P1.0 和 P1.1 控制下只显示第 1 路采样值
```


基于模拟前端信号处理与控制技术的专业论坛、网站

```
  {
    P5OUT=b[Volt0/10];          //P5 口显示第 1 路采样值的十位
    P4OUT=a[Volt0%10];          //P4 口显示第 1 路采样值的个位
    P3OUT|=0X0F;
    P3OUT&=0XFE;
  }
  if(kk0==2)            //按键 P1.0 和 P1.1 控制下只显示第 2 路采样值
  {
    P5OUT=b[Volt1/10];          //P5 口显示第 2 路采样值的十位
    P4OUT=a[Volt1%10];          //P4 口显示第 2 路采样值的个位
    P3OUT|=0X0F;
    P3OUT&=0XFD;
  }
 if(data1==0&&kk0==0)          //自动交替显示下，data1 为零期间显示第 1 路采样值
  {
    P5OUT=b[Volt0/10];          //P5 口显示第 1 路采样值的十位
    P4OUT=a[Volt0%10];          //P4 口显示第 1 路采样值的个位
    P3OUT|=0X0F;
    P3OUT&=0XFE;
  }
 if(data1==1&&kk0==0)          //自动交替显示下，data1 为 1 期间显示第 2 路采样值
  {
    P5OUT=b[Volt1/10];          //P5 口显示第 2 路采样值的十位
    P4OUT=a[Volt1%10];          //P4 口显示第 2 路采样值的十位
    P3OUT|=0X0F;
    P3OUT&=0XFD;
  }
 }
}
void Adcvolt (void)             //进行电压转换时 ADC12 的初始化
{
ADC12CTL0&=~ENC;                //ENC 为低电平，设置 AD 控制寄存器
ADC12CTL0|=ADC12ON+MSC;         //打开 ADC12，可以进行 AD 转换，参考电压选 3.3V
ADC12CTL1=CSTARTADD_0+CONSEQ_1+SHP;  //单通道单次转换，采样频率源自采样定时器
ADC12MCTL0=INCH_0;              //选择模拟输入通道 2
ADC12MCTL1=EOS+INCH_1;          //选择模拟输入通道 2
ADC12IE|=BIT0+BIT1;             //AD 转换中断允许
ADC12CTL0|=ENC;                 //转换允许
ADC12CTL0|=ADC12SC;             //开始 A/D 转换
}
interrupt[PORT1_VECTOR] void PORT1 (void)
   {
     unsigned temp,i;           //局部变量 temp 暂时存放端口的中断标志寄存器中的值
     for(i=0;i<=2000;i++); //消除抖动延时
     if ((P1IN&0xff)!=0xff)//如果有键按下
       {
```


基于模拟前端信号处理与控制技术的专业论坛、网站

```
            temp=P1IFG;     //temp 记录中断标志
            switch(temp)
            { case 1: keyvaluep1=0;break;
              case 2: keyvaluep1=1;break;
              case 4: keyvaluep1=2;break;
              case 8: keyvaluep1=3;break;
              case 16: keyvaluep1=4;break;
              case 32: keyvaluep1=5;break;
              case 64: keyvaluep1=6;break;
              case 128: keyvaluep1=7;break;
              default: keyvaluep1=8;break;
            }
        }
        if(keyvaluep1==0 ) //P1.0?
          {kk0=0;
           P3OUT |= 0XF0;
           P3OUT &= 0X7F;
           }
        if(keyvaluep1==1 )                //
          {kk0=1;
           P3OUT |= 0XF0;
           P3OUT &= 0XBF;}
        if(keyvaluep1==2)
        {kk0=2;
          P3OUT |= 0XF0;
        P3OUT &= 0XDF;}
        P1IFG=0X00;                       //清除中断标志，返回主程序
    }
interrupt[ADC_VECTOR] void ADC(void) //A/D 转换中断子程序
{
    ADresult0=ADC12MEM0;          //转换结果寄存器给了变量 ADresult0
    Volttem0=((long)ADresult0*33)/4095;     //计算实际电压值
    ADresult1=ADC12MEM1;          //转换结果寄存器给了变量 ADresult1
    Volttem1=((long)ADresult1*33)/4095;     //计算实际电压值
    Volt0=Volttem0;
    Volt1=Volttem1;
}
interrupt[TIMERA0_VECTOR] void Timer_A(void)   //TIMERA 中断子程序
{
  data0++;
  if(data0>=30)
    data1=1;
  else
  data1=0;
  if (data0>=60)
     data0=0;
```


基于模拟前端信号处理与控制技术的专业论坛、网站

```
  P3OUT ^= BIT3;
 ADC12CTL0|=ADC12SC;            //每隔一定时间进行一次转换
}
```

例 6. PWM 方式控制发光二极管的亮度

说明： 按钮控制 PWM 输出，使 P4.1 的发光二极管（数码管的 b 段），由亮到灭分为 8 级控制，一个按钮增加亮度，另外一个减少亮度,数码管显示亮度等级,P1.0 控制变亮，P1.1 控制变暗.

```
#include <msp430x14x.h>
unsigned int count=0;        //定义亮度等级变量
const unsigned char seg[]={0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80};    //显示段码表
void main(void)
{
WDTCTL=WDTPW+WDTHOLD; //停止看门狗
TBCTL=TBSSEL_2+TBCLR+MC0;
//定时器 B 的时钟是 SMCLK（800K），16 位计数，只用 CCR0，增计数模式，允许中断
TBCCTL1=OUTMOD_3; //PWM 输出模式：置位/复位
TBCCR0=8000;          //PWM 中断周期
TBCCR1=1000;          //PWM 的低电平时间
P4DIR=0X02;           //P4.1 输出，P4.1 受 TBCCR1 控制
P4SEL=0X02;           //P4.1 作为定时器 B 的 PWM 输出
P1DIR=0xFC;           //P1.0 和 P1.1 输入
P1IE=0x03;            //允许中断 P1.0 和 P1.1
P1IES=0xfc;           //上升沿中断
P5DIR=0XFF;           //P5 口为输出
P5OUT=0xf9;           //输出等级为 1
_EINT();              //开总中断
while(1);
}
interrupt [PORT1_VECTOR] void PORT1_INTERRUPT(void)
{
unsigned int i;                    //定义延时常数
for(i=8000;i>0;i--);               //延时，消抖
  if (P1IFG&BIT0)
  {
    if(TBCCR1>=8000)               //如果已达到最亮，则复位，重新开始
    {
      TBCCR1=1000;                 //低电平时间不变
      count=0;                     //亮度等级不变
      P5OUT=seg[count];            //显示亮度等级 1~8
    }
  else
  {
  TBCCR1=TBCCR1+1000;
  count=count+1;                   //亮度等级递增
  P5OUT=seg[count];                //显示亮度等级 1~8
      //如果 P1.0 的按键确实是按下了一次，则低电平时间增长一次，P4.0 的发光亮度增强一次
```


基于模拟前端信号处理与控制技术的专业论坛、网站

```
}
}
if   (P1IFG&BIT1) //如果 P1.1 的按键确实是按下了一次
   {
   if (TBCCR1<=1000)          //则判断是否是最低亮度
   {
      TBCCR1=1000;                  //如果是，则亮度不变
      count=0;                      //亮度等级不变
      P5OUT=seg[count];          //显示亮度等级 1~8
   }
   else
   {
      TBCCR1=TBCCR1-1000;    //如果不是，则亮度递减
      count=count-1;               //亮度等级递减
      P5OUT=seg[count];           //显示亮度等级 1~8
    }
   }
P1IFG=0X00;                    //清除按键中断标志
}                              //中断返回
```