51单片机实验（一）定时/计数器及其中断

原创

haohulala 于 2019-06-04 16:13:22 发布 20593 收藏 141
分类专栏： 单片机实验 文章标签： 51 中断 定时器/计数器

我们这学期开了单片机的课，不知道为什么我们要用汇编语言写程序，感觉汇编程序真的挺难写的，所以把实验记录下来把。

如果没有学过汇编的小伙伴建议先去熟悉一下简单的汇编指令，之前简单的实验我就不记录了，我们从定时计数器实验开始吧。

首先来回顾一下和中断，定时有关的知识

51单片机中断级别

中断源	默认中断级别	序号（C语言用）
INT0---外部中断0	最高	0
T0---定时器/计数器0中断	第2	1
INT1---外部中断1	第3	2
T1----定时器/计数器1中断	第4	3
TX/RX---串行口中断	第5	4
T2---定时器/计数器2中断	最低	5

中断允许寄存器IE

位序号	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
符号位	EA	-------	ET2	ES	ET1	EX1	ET0	EX0

EA---全局中允许位。
EA=1，打开全局中断控制，在此条件下，由各个中断控制位确定相应中断的打开或关闭。
EA=0，关闭全部中断。
-------，无效位。
ET2---定时器/计数器2中断允许位。 EA总中断开关，置1为开；
ET2=1，打开T2中断。 EX0为外部中断0（INT0）开关，……
ET2=0，关闭T2中断。 ET0为定时器/计数器0（T0)开关，……
ES---串行口中断允许位。 EX1为外部中断1（INT1）开关，……
ES=1，打开串行口中断。 ET1为定时器/计数器1（T1)开关，……
ES=0，关闭串行口中断。 ES为串行口（TX/RX）中断开关，……
ET1---定时器/计数器1中断允许位。 ET2为定时器/计数器2（T2)开关，……
ET1=1，打开T1中断。
ET1=0，关闭T1中断。
EX1---外部中断1中断允许位。
EX1=1，打开外部中断1中断。
EX1=0，关闭外部中断1中断。
ET0---定时器/计数器0中断允许位。
ET0=1，打开T0中断。
ET0=0，关闭T0中断。
EX0---外部中断0中断允许位。
EX0=1，打开外部中断0中断。
EX0=0，关闭外部中断0中断。
中断优先级寄存器IP

位序号	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
位地址	---	---	---	PS	PT1	PX1	PT0	PX0

-------，无效位。
PS---串行口中断优先级控制位。
PS=1，串行口中断定义为高优先级中断。
PS=0，串行口中断定义为低优先级中断。
PT1---定时器/计数器1中断优先级控制位。
PT1=1，定时器/计数器1中断定义为高优先级中断。
PT1=0，定时器/计数器1中断定义为低优先级中断。
PX1---外部中断1中断优先级控制位。
PX1=1，外部中断1中断定义为高优先级中断。
PX1=0，外部中断1中断定义为低优先级中断。
PT0---定时器/计数器0中断优先级控制位。
PT0=1，定时器/计数器0中断定义为高优先级中断。
PT0=0，定时器/计数器0中断定义为低优先级中断。
PX0---外部中断0中断优先级控制位。
PX0=1，外部中断0中断定义为高优先级中断。
PX0=0，外部中断0中断定义为低优先级中断。
定时器/计数器工作模式寄存器TMOD

位序号	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
位符号	GATE	C/T	M1	M0	GATE	C/T	M1	M0

|----------定时器1----------|----------定时器0----------|

GATE---门控制位。

GATE=0，定时器/计数器启动与停止仅受TCON寄存器中TRX(X=0,1)来控制。

GATE=1，定时器计数器启动与停止由TCON寄存器中TRX(X=0,1)和外部中断引脚（INT0或INT1）上的电平状态来共同控制。

C/T---定时器和计数器模式选择位。

C/T=1，为计数器模式；C/T=0，为定时器模式。

M1M0---工作模式选择位。

M1	M0	工作模式
0	0	方式0，为13位定时器/计数器
0	1	方式1，为16位定时器/计数器
1	0	方式2，8位初值自动重装的8位定时器/计数器
1	1	方式3，仅适用于T0，分成两个8位计数器，T1停止工作

定时器/控制器控制寄存器TCON

位序号	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
符号位	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1---定时器1溢出标志位。

当定时器1记满溢出时，由硬件使TF1置1，并且申请中断。进入中断服务程序后，由硬件自动清0。需要注意的是，如果使用定时器中断，那么该位完全不用人为去操作，但是如果使用软件查询方式的话，当查询到该位置1后，就需要用软件清0。

TR1---定时器1运行控制位。

由软件清0关闭定时器1。当GATE=1，且INIT为高电平时，TR1置1启动定时器1；当GATE=0时，TR1置1启动定时器1。

TF0---定时器0溢出标志，其功能及其操作方法同TF1。

TR0---定时器0运行控制位，其功能及操作方法同TR1。

IE1---外部中断1请求标志。

当IT1=0时，位电平触发方式，每个机器周期的S5P2采样INT1引脚，若NIT1脚为定电平，则置1，否则IE1清0。

当IT1=1时，INT1为跳变沿触发方式，当第一个及其机器周期采样到INIT1为低电平时，则IE1置1。IE1=1，表示外部中断1正向CPU中断申请。当CPU响应中断，转向中断服务程序时，该位由硬件清0。

IT1外部中断1触发方式选择位。

IT1=0，为电平触发方式，引脚INT1上低电平有效。

IT1=1，为跳变沿触发方式，引脚INT1上的电平从高到低的负跳变有效。

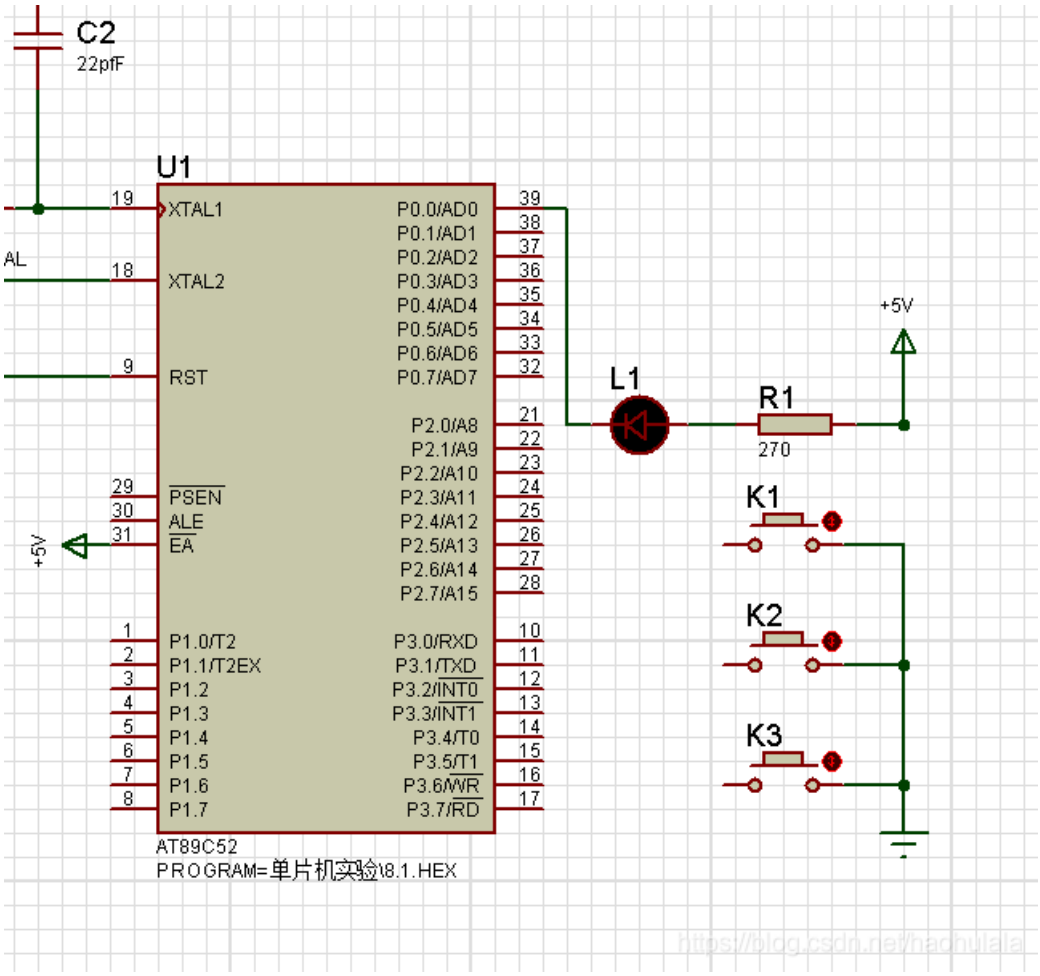
IE0---外部中断0请求标志，其功能及操作方法同IE1。

IT0---外部中断0触发方式选择位，其功能及操作方法同IT1。

PS:上面的内容是从别的论坛上复制的，贴在这复习也方面，如果侵权请联系我删除。

实验1 定时器/计数器及中断程序设计

这个实验比较简单，实验的基本连线如下



本次实验的内容是使用程序查询方式来判断定时器有没有溢出，从而控制LED的频闪。

我们使用的51单片机的频率是12MHz，时钟周期就是1/12M s，一个指令周期是12个时钟周期，也就是1us

我们要实验周期为200ms的闪烁，就需要每隔20*5ms就将p0.0为取反一次。

选用工作方式为0，也就是13位的定时器

这就需要计时器的初值=2^13-5ms/1us = 3192 = 0c78H

然后每当计数20次之后就将p0.0取反一次。

下面我们来看代码

```
ORG 0000H
 ljmp Start

 ORG 0100H
Start:
 mov TMOD,#00H    ;计数模式为0,13位计数模式
 mov TH0,#0CH ;6C78H 定时器0的高8位
 mov TL0,#78H     ;定时器0的低8位
 mov R7,#20       ;设置循环20次
 setb TR0         ;开定时器/计数器0的中断
```

首先是start 标记的初始化程序，设置TMOD寄存器，最低两位设置为00，然后将0C78H送入TH0 和 TH1，20送入R7，开中断。

```
Loop: jbc TF0,T0SVR   ;如果TF0位1就跳转,TF0是定时器0的溢出标志位
 sjmp Loop

T0SVR: mov TH0,#0CH ;0C78H
 mov TL0,#78H
 djnz R7,Next
 mov R7,#20
 cpl P0.0        ;取反p0.0位
Next:sjmp Loop
 END
```

接着就是判断TF0（定时器0的溢出标志位）为1就跳转到TOSVR标记的程序段，重新设置TH0和TL0的值，重新开始计时，如果已经重复过20次了，就继续运行下去，将R7重新赋值为20，并且取反p0.0，否则跳转到Loop语句标号出，继续计时。

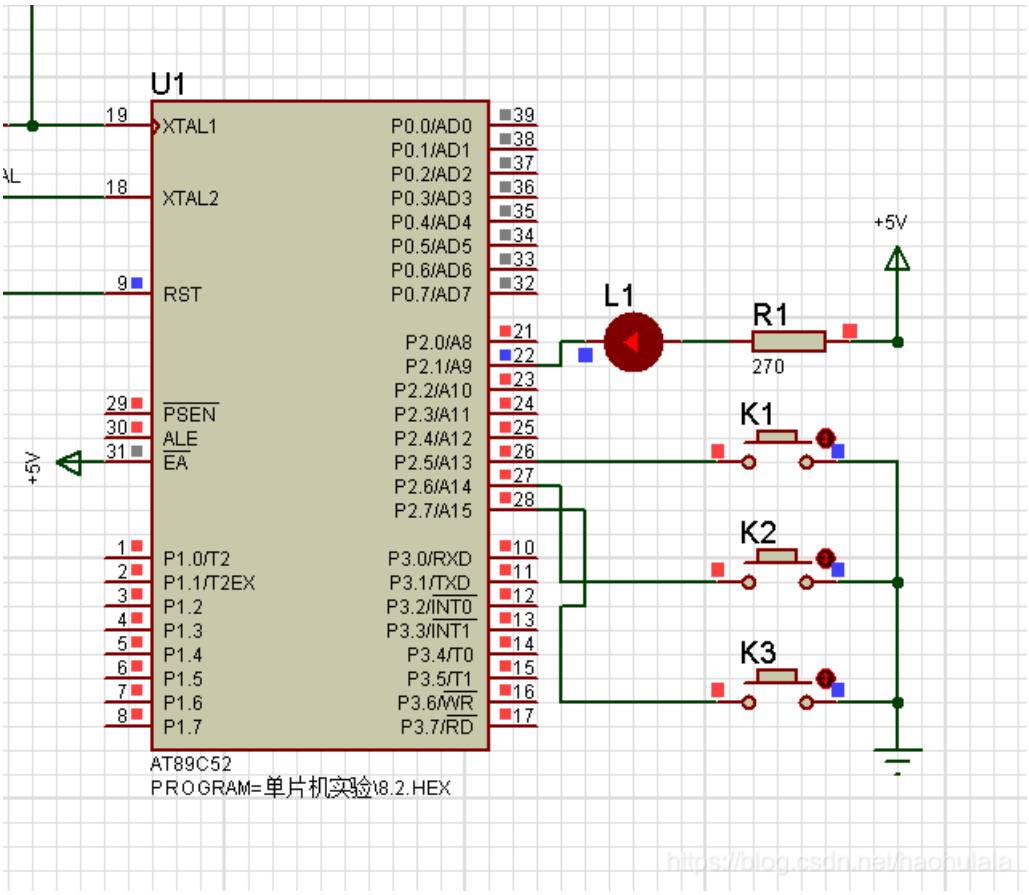
完整代码如下

```
;1:定时/计数器实验，方式0，查询方式编程
;系统时钟12MHz，T0每5ms溢出一次
;20次后取反P0.0
;执行后，与P0.0连接的LED亮0.1s灭0.1s，即以5Hz的频率闪烁
 ORG 0000H
 ljmp Start

 ORG 0100H
Start:
 mov TMOD,#00H   ;计数模式为0,13位计数模式
 mov TH0,#0CH ;6C78H 定时器0的高8位
 mov TL0,#78H    ;定时器0的低8位
 mov R7,#20      ;设置循环20次
 setb TR0        ;开定时器/计数器0的中断
    ;计数，溢出了之后就跳转
Loop: jbc TF0,T0SVR   ;如果TF0位1就跳转,TF0是定时器0的溢出标志位
 sjmp Loop

T0SVR: mov TH0,#0CH ;0C78H
 mov TL0,#78H
 djnz R7,Next
 mov R7,#20
 cpl P0.0        ;取反p0.0位
Next:sjmp Loop
 END
```

这个实验是不是很简单呢？不着急，我们马上进入下一个实验

实验2 使用T1方式完成上述实验

使用中断来控制闪烁频率，实验电路图连线如下



k3接在p2.7 k2 接在p2.6 k1接在p2.5，当开关断开的时候p2口都是高电平，当开关闭合的时候p2口就变成低电平

先来看一下初始化程序

```
XTH EQU  30H  ;存放定时器常数高8位
XTL EQU  31H    ;存放定时器常数低8位

 ORG 0000H
 ljmp Start

 ORG 001BH  ;用的是定时器1，自然是1B
 ljmp T1SVR

 ORG 0100H
Start: mov SP,#5FH  ;堆栈区设在未用RAM的高端
 mov TMOD,#00010000B ;T/C1，模式1
 mov TH1,#0FFH  ;FFF0啊，好像是负16，28800赫兹啊，
 mov TL1,#0F0H  ;对了，这行和上行可以有，也可以没有
 clr TR1  ;关闭定时器
 mov IE,#10001000B ;设置EA和ET1，好像在ISIS中EA这要接高电平
```

因为有中断，所以要初始化sp堆栈指针，用来进行中断返回，使用t/c1 T1模式，16位定时器，关闭定时器TR1，开中断。

接着是扫描程序，查询三个按钮的开关情况

```
ScanKey:
 mov A,P2
 cpl A   ;按下时为0，我们要当1处理，自然要翻转一下
 anl A,#11100000B ;只要最高3位，其它位屏蔽
 swap A   ;这一行和下一行共同作用，将3位按键移到最低3位，然后散转
 rr A  ;这一行和下一行可以同时去掉，不影响执行，但是程序上是有意义滴
 mov DPTR,#TABKey
 jmp @A+DPTR
```

在软件上模拟的时候，p2口一直是高电平，只有当按键按下去的时候才会变成低电平，然后取反，然后将高三位析取出来，交换到低位最后右移得到散装程序入口地址。

下面是散转程序

```
TABKey: ajmp NoKey ;跳转表KEY1P、2P、3P分别为按下指定3键后的处理代码，用于产生不同的频率
 ajmp Key1P
 ajmp Key2P
 ajmp NoKey ;这样的是组合键，如果指定另一种频率，则1和2按下时可以起作用
 ajmp Key3P
 ajmp NoKey
 ajmp NoKey
 ajmp NoKey
 sjmp ScanKey
NoKey: clr TR1  ;所有键都抬起来时执行，停止闪烁!
 sjmp ScanKey

Key1P: mov XTH,#0FEH ;FE0D=  (-500)二进制补码，1KHZ的频率闪烁所需要，这里存的数都是补码
 mov XTL,#0DH
 setb TR1        ;开定时器1
 sjmp ScanKey

Key2P: mov XTH,#0FFH ;FF07=  （-250）二进制补码，以2KHZ的频率闪烁
 mov XTL,#07H
 setb TR1
 sjmp ScanKey

Key3P: mov XTH,#0FFH
 mov XTL,#84H ;FF84=  （-125）二进制补码，以4KHZ的频率闪烁
 setb TR1
 sjmp ScanKey
```

关于散转程序入口地址，我们可以看成k3k2k1组成的3位二进制数（按下的时候为1，没有按下的时候为0），然后就可以计算他们相对于TABKey的偏移地址了。

如果没有按下按钮的话，就关闭TR1中断，停止闪烁，跳回到扫描程序

如果按下了按钮的话，就进入相应的散转程序，设置TH1和TL1的值，我们使用XTH和XTL作为中间变量给TH1和TL1赋值，一开始赋值了30H和31H，也就是说计时器溢出了之后才会重新给TH1和TL1赋值，然后将p2.1位取反。

这里面还涉及到二进制补码的计算，简单说一下负数补码的算法：符号位不变，数值为取反之后加一。比如-500=FE0DH，计算机内存的都是补码

最后就是定时器溢出中断程序

```
T1SVR: mov TH1,XTH
 mov TL1,XTL
 cpl P2.1 ;翻转P2.1，每两次一个周期
 reti
 END
```

这就是我们刚刚说的包括给TH1和TL1赋值和将p2.1取反的操作

完整程序如下

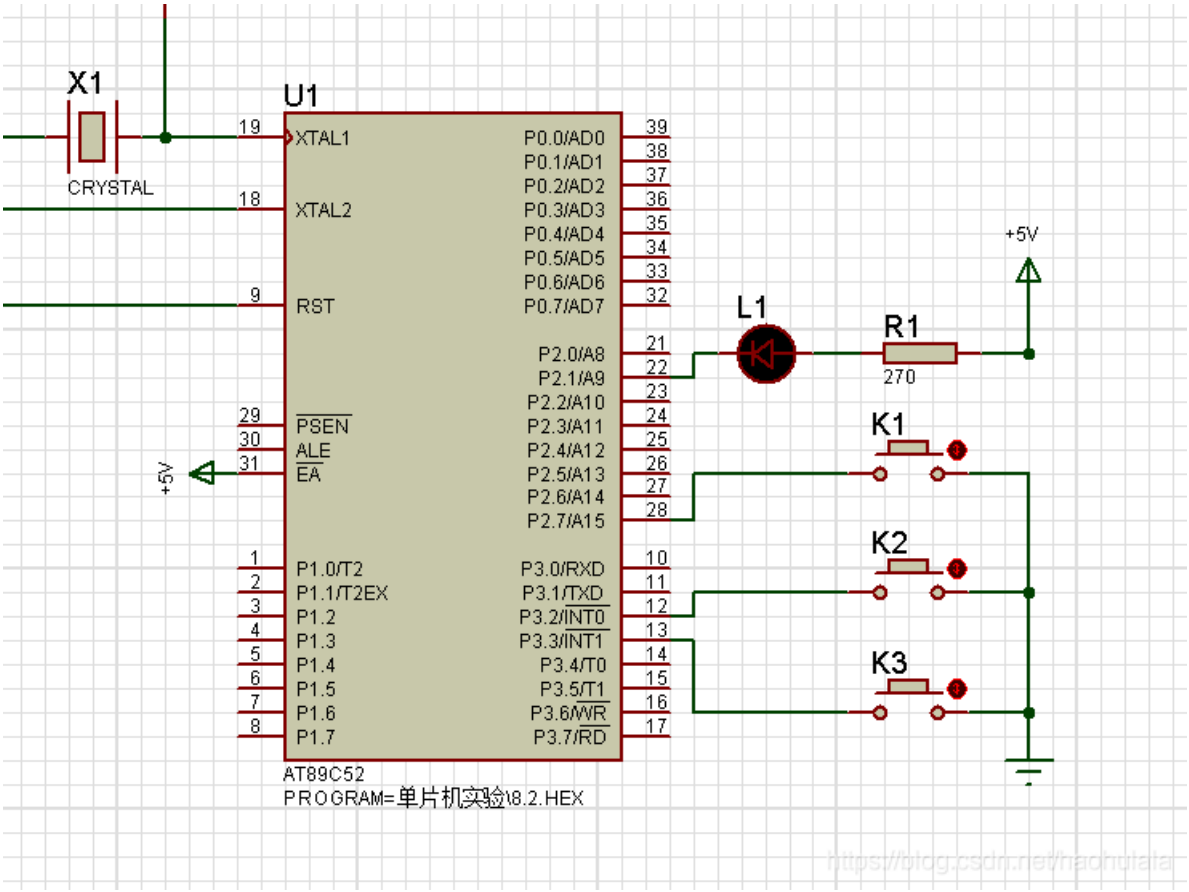
```
;2:定时/计数器实验，T1方式1
;系统时钟12MHz，P2.1连接LED，P2.5~P2.7分别连接K1~K3
;用户按下K1~K3，LED以不同频率(1KHz、2KHz、4KHz)闪烁,即定时时间分别为0.5ms，0.25ms，0.125ms
;宏定义常量
XTH EQU  30H  ;存放定时器常数高8位
XTL EQU  31H    ;存放定时器常数低8位

 ORG 0000H
 ljmp Start

 ORG 001BH  ;用的是定时器1，自然是1B
 ljmp T1SVR

 ORG 0100H
Start: mov SP,#5FH  ;堆栈区设在未用RAM的高端
 mov TMOD,#00010000B ;T/C1，模式1
 mov TH1,#0FFH  ;FFF0啊，好像是负16，28800赫兹啊，
 mov TL1,#0F0H  ;对了，这行和上行可以有，也可以没有
 clr TR1  ;关闭定时器
 mov IE,#10001000B ;设置EA和ET1，好像在ISIS中EA这要接高电平

ScanKey:
 mov A,P2
 cpl A   ;按下时为0，我们要当1处理，自然要翻转一下
 anl A,#11100000B ;只要最高3位，其它位屏蔽
 swap A   ;这一行和下一行共同作用，将3位按键移到最低3位，然后散转
 rr A  ;这一行和下一行可以同时去掉，不影响执行，但是程序上是有意义滴
 mov DPTR,#TABKey
 jmp @A+DPTR

TABKey: ajmp NoKey ;跳转表KEY1P、2P、3P分别为按下指定3键后的处理代码，用于产生不同的频率
 ajmp Key1P
 ajmp Key2P
 ajmp NoKey ;这样的是组合键，如果指定另一种频率，则1和2按下时可以起作用
 ajmp Key3P
 ajmp NoKey
 ajmp NoKey
 ajmp NoKey
 sjmp ScanKey
NoKey: clr TR1  ;所有键都抬起来时执行，停止闪烁!
 sjmp ScanKey

Key1P: mov XTH,#0FEH ;FE0D=  (-500)二进制补码，1KHZ的频率闪烁所需要，这里存的数都是补码
 mov XTL,#0DH
 setb TR1        ;开定时器1
 sjmp ScanKey

Key2P: mov XTH,#0FFH ;FF07    (-250)二进制补码，以2KHZ的频率闪烁
```

```
Key2P: mov XTH,#0FFH ;FF07=  （-250）二进制补码，以2KHZ的频率闪烁
 mov XTL,#07H
 setb TR1
 sjmp ScanKey

Key3P: mov XTH,#0FFH
 mov XTL,#84H ;FF84=  （-125）二进制补码，以4KHZ的频率闪烁
 setb TR1
 sjmp ScanKey

T1SVR: mov TH1,XTH
 mov TL1,XTL
 cpl P2.1 ;翻转P2.1，每两次一个周期
 reti
 END
```

那么到这里，实验2就结束啦

实验3 中断的嵌套

实验3就是在实验2的基础上加上了中断的嵌套，所以我们需要先设置中断优先寄存器IP的初始值，电路图如下



我们开看一下初始化程序

```
XTH  EQU  30H  ;存放定时器常数高8位
XTL  EQU  31H    ;存放定时器常数低8位
  ORG 0000H
  ljmp Start
  ORG 0003H
  ljmp Ex0SVR
  ORG 0013H
  ljmp Ex1SVR
  ORG 001BH
  ljmp T1SVR

  ORG 0100H
Start: mov SP,#5FH  ;堆栈区设在未用RAM的高端
  mov TMOD,#00010000B
  mov TH1,#0FFH
  mov TL1,#0F0H
  clr TR1          ;关TR1中断
  mov IE,#10001101B
  mov IP,#00001000B;设置不同的优先级，观察执行效果
  mov P2,#0FFH
  mov P3,#0FFH
```

设置sp堆栈指针（有中断的程序不要忘了设置sp的初始值），设置T/C1的T1模式（16位定时器），关TR1中断

设置IE中 EA =1（总的中断允许控制位） ES=1（串行口中断允许控制位） ET1 = 1（定时器1） ET0 = 1（定时器0）

设置IP中的 PT1 = 1，即定时器1的优先级高

这里顺便附上中断入口地址的地址

外部中断0：入口：0003H

定时器0： 入口：000BH

外中断1： 入口：0013H

定时器1： 入口：001BH

串口中断：入口：0023H

下面是中断程序和扫描程序

```
ScanKey:mov A,P2
  jb ACC.7,ScanKey
Key2P: mov XTH,#0FEH
  mov XTL,#0CH
  setb TR1
  sjmp ScanKey
Ex0SVR: mov XTH,#0FFH
  mov XTL,#06H
  setb TR1
  reti
Ex1SVR: mov XTH,#0FFH
  mov XTL,#83H
  setb TR1
  reti

T1SVR: mov TH1,XTH
  mov TL1,XTL
  cpl P2.1        ;p2.1口取反
  reti
  END
```

从电路图可以看出来，K3是控制ACC.7；K2是中断0，K1是中断1

也就是一开始小灯是不闪的，按下任意一个按钮后，就开中断了，就会不停闪，通过按不同的按钮就可以改变定时器TH1和TL1的值从而控制灯闪烁的频率，还是挺简单的嘛

下面是最后一个实验的完整代码

```
XTH  EQU  30H  ;存放定时器常数高8位
XTL  EQU  31H    ;存放定时器常数低8位
  ORG 0000H
  ljmp Start
  ORG 0003H
  ljmp Ex0SVR
  ORG 0013H
  ljmp Ex1SVR
  ORG 001BH
  ljmp T1SVR

  ORG 0100H
Start: mov SP,#5FH  ;堆栈区设在未用RAM的高端
  mov TMOD,#00010000B
  mov TH1,#0FFH
  mov TL1,#0F0H
  clr TR1         ;关TR1中断
  mov IE,#10001101B
  mov IP,#00001000B;设置不同的优先级，观察执行效果
  mov P2,#0FFH
  mov P3,#0FFH
ScanKey:mov A,P2
  jb ACC.7,ScanKey
Key2P: mov XTH,#0FEH
  mov XTL,#0DH        ;1KHZ
  setb TR1
  sjmp ScanKey
Ex0SVR: mov XTH,#0FFH
  mov XTL,#07H        ;2KHZ
  setb TR1
  reti
Ex1SVR: mov XTH,#0FFH
  mov XTL,#84H        ;4KHZ
  setb TR1
  reti

T1SVR: mov TH1,XTH
  mov TL1,XTL
  cpl P2.1        ;p2.1口取反
  reti
  END
```

那么，这次实验就做完啦，✿✿ヽ(°▽°)ノ✿