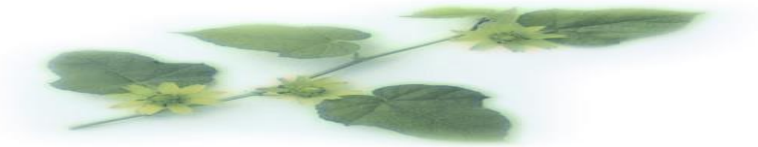
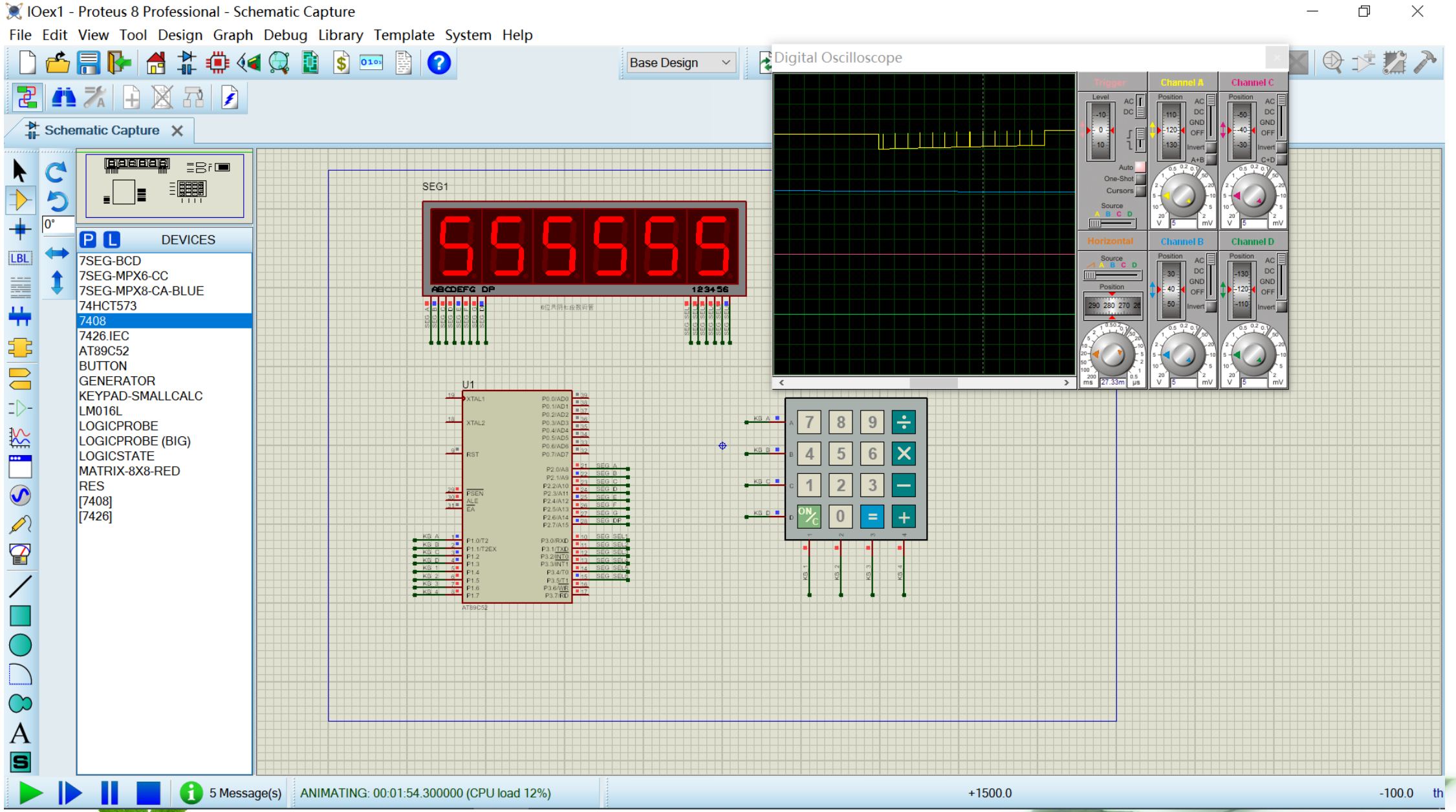
第4章 单片机的中断系统




IOex1 - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Base Design
Schematic Capture
Digital Oscilloscope
DEVICES
7SEG-BCD
7SEG-MPX6-CC
7SEG-MPX8-CA-BLUE
74HCT573
7408
7426.IEC
AT89C52
BUTTON
GENERATOR
KEYPAD-SMALLCALC
LM016L
LOGICPROBE
LOGICPROBE (BIG)
LOGICSTATE
MATRIX-8X8-RED
RES
[7408]
[7426]
SEG1
U1
5 Message(s)
ANIMATING: 00:01:54.300000 (CPU load 12%)
+1500.0
-100.0

内容提要

★MCS-51单片机的中断系统

- ★ 中断的基本概念
- ★ 中断的系统结构
- ★ 中断的响应过程
- ★ 中断的的应用编程

4.2 MCS－51单片机的中断系统

4.2.1 8XX51中断系统结构

8XX51有5个中断源，3个在片内，2个在片外，它们在程序存贮器中有固定的中断入口地址， 当CPU响应中断时，硬件自动形成这些地址，由此进入中断服务程序；5个中断源有两级中断优先级，可形成中断嵌套；

中断优先级

当有多个中断源同时 向CPU申请中断时，CPU优先响应最需紧急处理的中断请求，处理完毕再响应优先级别较低的，这种预先安排的响应次序。

中断的嵌套

在中断系统中，高优先级的 中断请求能中断正在进行的较低级的中断源处理。

符号	名 称	中 断 引 起 原 因	中断服务程序入口
INT0	外部中断 0	P3.2引脚的低电平或下降沿信号	0003H
INT1	外部中断 1	P3.3引脚的低电平或下降沿信号	0013H
T0	定时器0中断	定时计数器0计数回零溢出	000BH
T1	定时器1中断	定时计数器1计数回零溢出	001BH
T2	定时器2中断	定时计数器2中断(TF2或T2EX信号)	002BH
TI/RI	串行口中断	串行通信完成一帧数据发送或接收引起中断	0023H

二、中断控制的有关寄存器

(1)中断的允许和禁止——中断控制寄存器IE

IE寄存器的各位对应相应的中断源，如果允许该中断源中断则该位置1，禁止中断则该位0 。

EA	–	ET2	ES	ET1	EX1	ET0	EX0
中断总控允/禁	不用	T2允/禁	串行口允/禁	T1允/禁	INT1允/禁	T0允/禁	INT0允/禁

(2)中断请求标志及外部中断方式选择寄存器TCON

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
T1 请求 有/无	T1 工作 启/停	T0 请求 有/无	T0 工作 启/停	$\overline{\text{INT1}}$ 请求 有/无	$\overline{\text{INT1}}$ 方式 下沿/ 低电平	$\overline{\text{INT0}}$ 请求 有/无	$\overline{\text{INT0}}$ 方式 下沿/低电平

说明：

1. IT0和IT1为外中断INT0 和INT1中断触发方式选择，若选下降沿触发则相应位置1；若选低电平触发，IT相应位置0。
2. 某中断源有中断请求，该中断标志自动置1，无中断请求，该标志置0
3. TR0 和 TR1 为定时器T0和T1 工作启动和停止控制。

（3）优先级别由IP寄存器管理，相应位置1，则该中断源优先级别高，置0的优先级 别低。

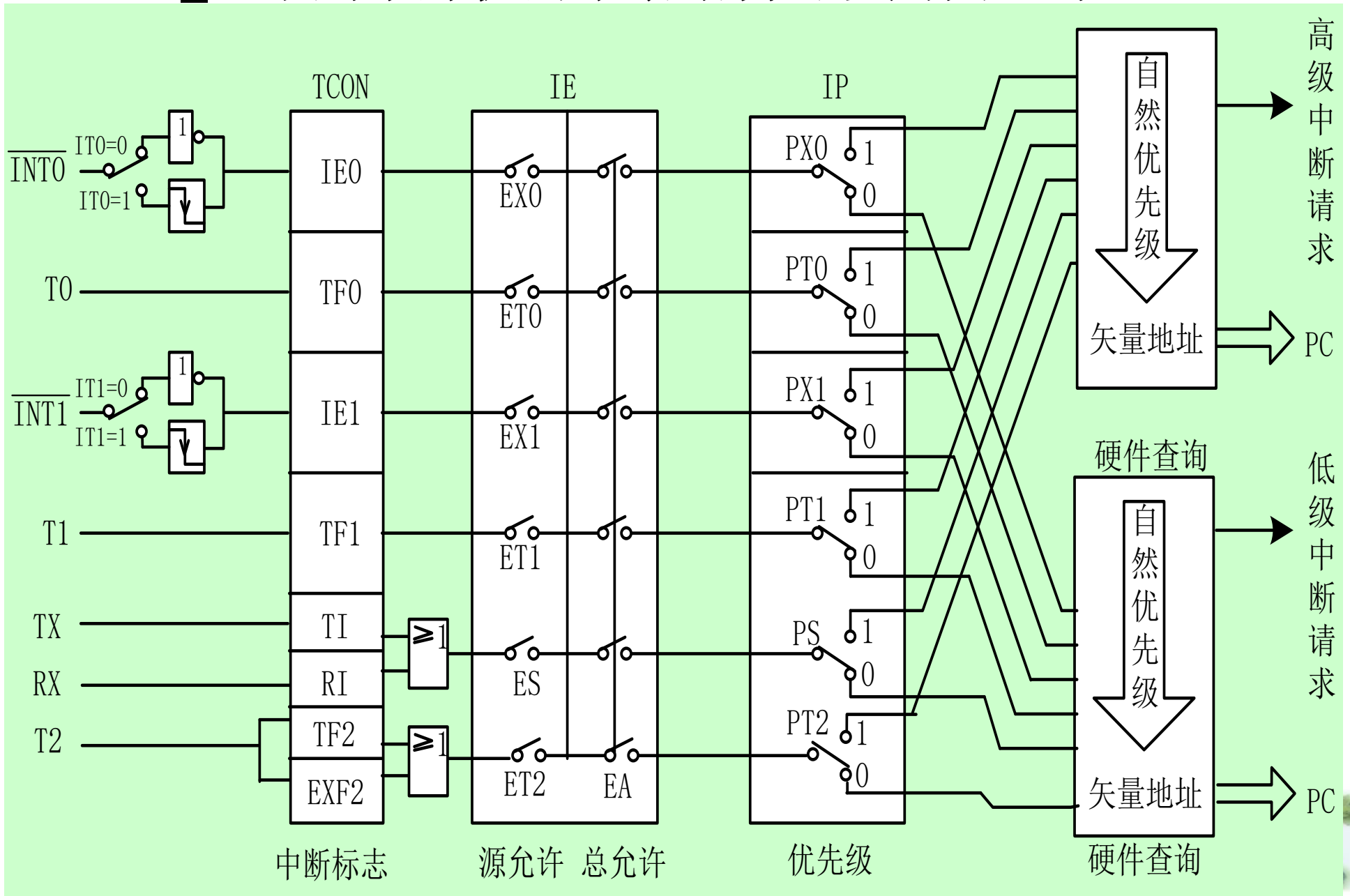
----	----	PT2	PS	PT1	PX1	PT0	PX0
无用位	无用位	T2 高/低	串行 口 高/低	T1 高/低	INT1 高/低	T0 高/低	INT0 高/低

当 某几个中断源在IP寄存器相应位同为1或同为零时，由内部查询确定优先级，查询的顺序是：

INT0 ⟶ T0 ⟶ INT1 ⟶ T1 ⟶ 串行口 ⟶ T2

CPU优先响应 先查询的中断请求

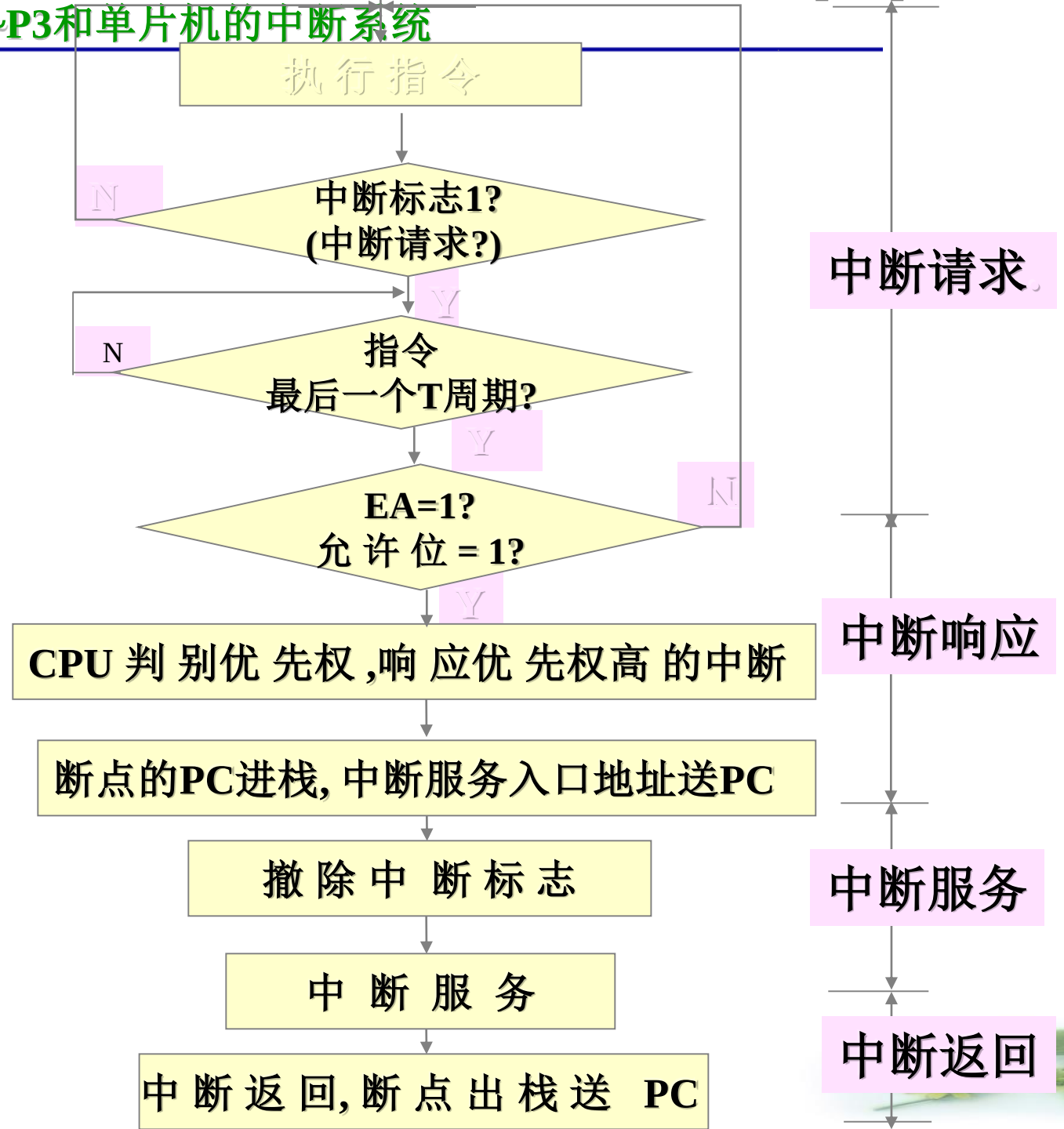
MCS_51系列单片机的中断结构可以用图6.1示。



4.2.2 中断响应过程

一、中断处理过程

中断处理过程分为四个阶段：中断请求，中断响应，中断处理、中断返回。MCS－51系列单 片机的中断过程流程如图6.2所示。


执行指令
中断标志1?
(中断请求?)
N
Y
指令
最后一个T周期?
N
Y
EA=1?
允许位=1?
N
Y
CPU判别优先权,响应优先权高的中断
断点的PC进栈, 中断服务入口地址送PC
撤除中断标志
中断服务
中断返回, 断点出栈送PC
中断请求
中断响应
中断服务
中断返回

中断请求、中断响应过程由硬件完成。

中断服务程序应根据需要进行编写。程序中要注意保护现场和恢复现场。

中断返回是通过执行一条**RETI**中断返回指令，使堆栈中被压入的断 点地址送PC，从而返回主程序的断点继续执行主程序。另外**RETI**还有恢复优先级状态触发器 的作用，因此不能以**RET**指令代替“**RETI**”指令。

二、中断请求的撤除---CPU响应中断后，应撤除该中断请求标志，否则会再次中断。

★ 对定时计数器T0、T1的溢出中断，CPU响应中断后，硬件自动清除中断请求标志TF0 TF1。

★ 对边沿触发的外部中断INT1和INT0，CPU响应中断后硬件自动清除中断请求标志IE0和IE1。

★ 对于串行口中断，CPU响应中断后，没有用硬件清除中断请求标志TI、RI，即这些中断标志 不会自动清除，必须用软件清除，这是在编串行通信中断服务中应该注意的。

★ 对电平触发的外部中断，CPU在响应中断时也不会自动清除中断标志，因此，在CPU响应中断后应立即撤除INT1或INT0的低电平信号。

4.2.3 中断的汇编语言程序和C语言程序设计

用户对中断的控制和管理，实际是对4个与中断有关的寄存器IE、TCON、IP、SCON进行控制 或管理。

- 开中断总控开关EA，置位中断源的中断允许位。
- 对外部中断INT0、INT1应选择中断触发方式
- 多个中断源中断，应设定中断优先级，预置IP。
- 编写中断服务程序，并注意用保护现场和恢复现场，以免中断返回时，丢失原寄存器、累加器中的信息。

若要在执行当前中断程序时，需要禁止更高优先级中断，可以采用软件关CPU中断．或禁止某中断源中断，在中断返回前再开放中断。

4.2.3 中断程序的设计

汇编语言的中断服务程序按规定的中断矢量地址存入，由于五个中断矢量地址0003H、000BH 、0013H、001BH、0023H之间相距很近，往往装不下一个中断服务程序，通常将中断服务程 序安排在程序存贮器的其他地址空间，而在矢量地址的单元中安排一条转移指令。

10.8.3 C51中断程序的编制

C51中断服务函数的完整定义如下:

返回值 函数名([参数])[模式] [再入] interrupt n [using m]

其中必选项 interrupt n表示将函数声明为中断服务函数，n为中断源编号，可以是0～31间的整数 ，不允许是带运算符的表达式，n通常取以下值：

0 外部中断0；

1 定时器/计数器0溢出中断

2 外部中断1；

3 定时器/计数器1溢出中断

4 串行口发送与接收中断

using m 定义函数使用的工作寄存器组，m的取值范围为0～3，可缺省。

函数由“RETI”指令终止

例1. 在图6.3中P1.4～P1.7接有四个发光二极管，P1.0～P1.3接有四个开关，消抖电路用于产生中断请求信号，当消抖电路的开关来回拔动一次将产生一个下降沿信号，通过INT0向CPU申请中断。

要求：初时发光二极管全黑，每中断一次，P1.0～P1.3所接的开关状态反映到发光二极管上，且要求开关断开的对应发光二极管亮，电路和现象如下：

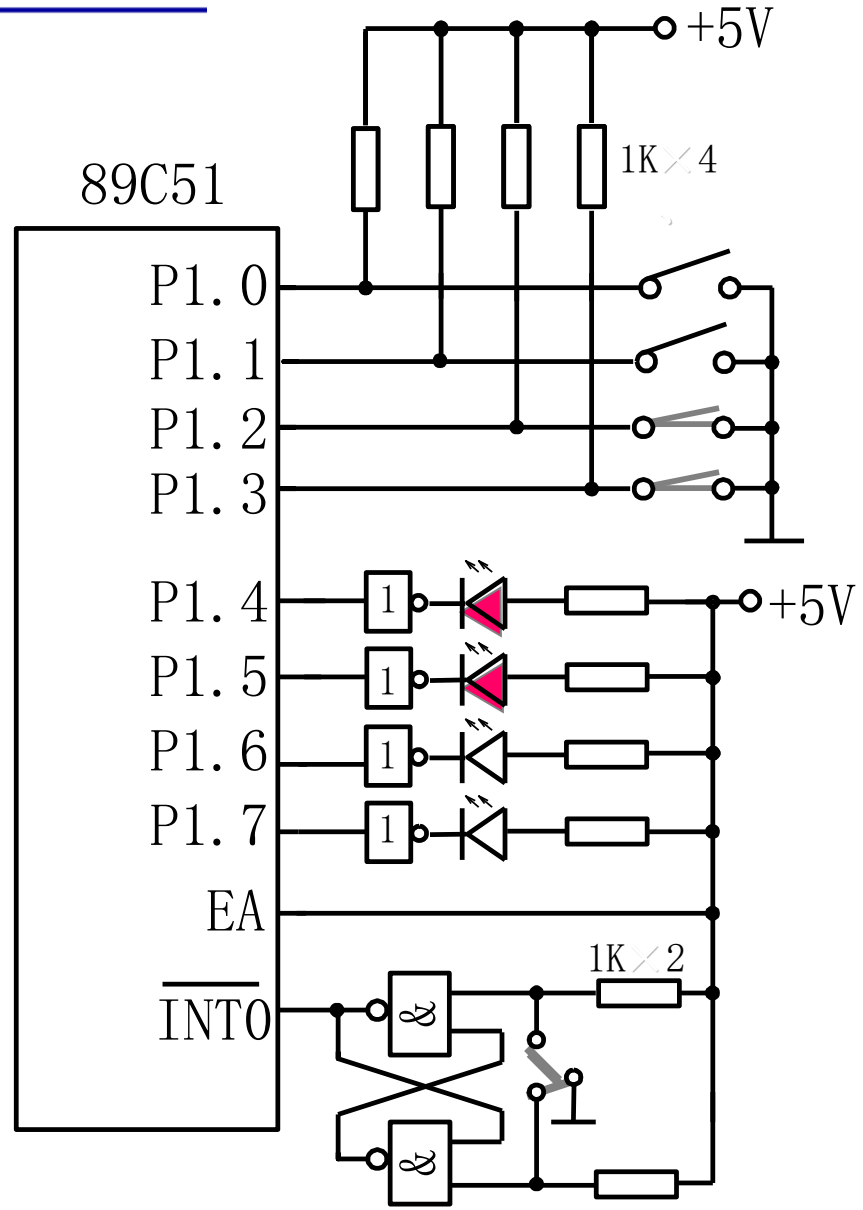


图6.3

```
        ORG   0000H
        AJMP  MAIN
        ORG   0003H       ；INT0中断入口
        AJMP  WBI         ；转中断服务程序
        ORG   0030H       ；主程序
MAIN:   MOV   P1，#0FH    ；全灯灭，低四位输入
        SETB  IT0         ；边沿触发中断
        SETB  EX0         ；允许外中断0中断
        SETB  EA          ；开中断开关
        SJMP  $
WBI:    MOV   P1，#0FH    ；P1先写入“1”且灯灭
        MOV   A，P1       ；输入开关状态
        SWAP  A
        MOV   P1，A       ；输出到P1高4位
        RETI
        END
```

例10-15 对10.2.3的例10-4（见图）要求每中断一次，发光二极管显示开关状态

用C语言编程

```
#include<reg51.h>
int0() interrupt 0   /*INT0中断函数*/
 {P1=0x0f;     /*输入端先置1，灯灭*/
  P1<<=4;}     /* 读入开关状态，并左移四位，
                  使开关反映在发光二极管上*/
 main()
 {EA=1;        /*开中断总开关*/
  EX0=1;       /*允许INT0中断*/
  IT0=1;       /*下降沿产生中断*/
  while(1); /*等待中断*/
 }
```

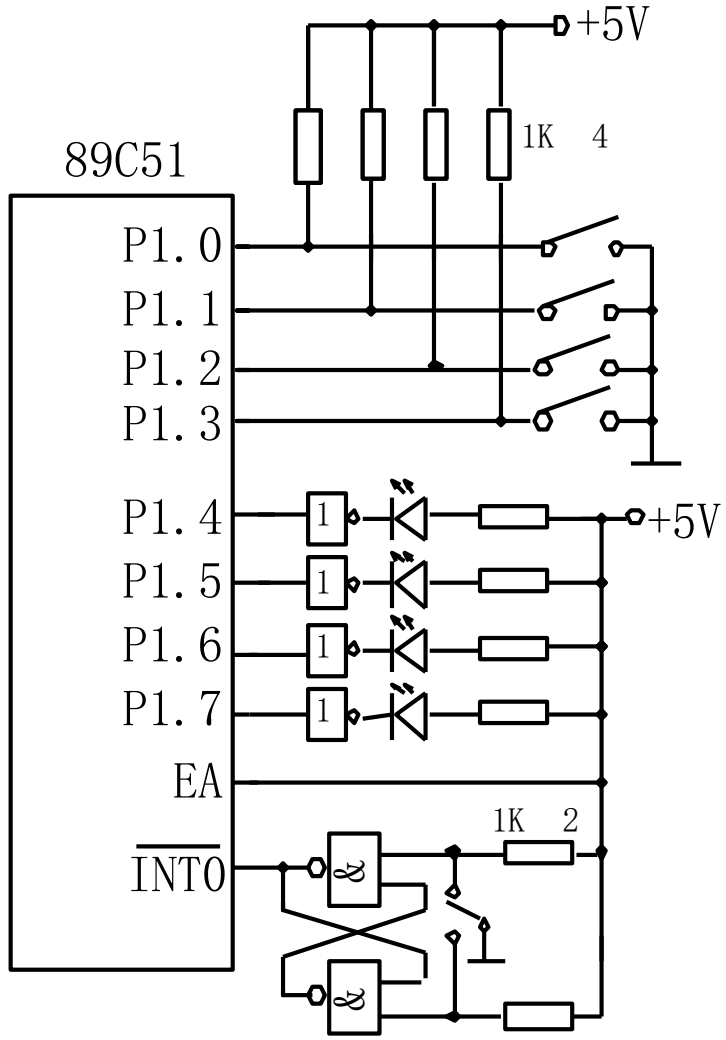


图6.3

例2. 89C51的P1口接一个共阴极的数码管，利用消抖开关产生中断请求信号，每来回拔动一 次开关，产生一次中断，用数码管显示中断的次数(最多不超过15次)。

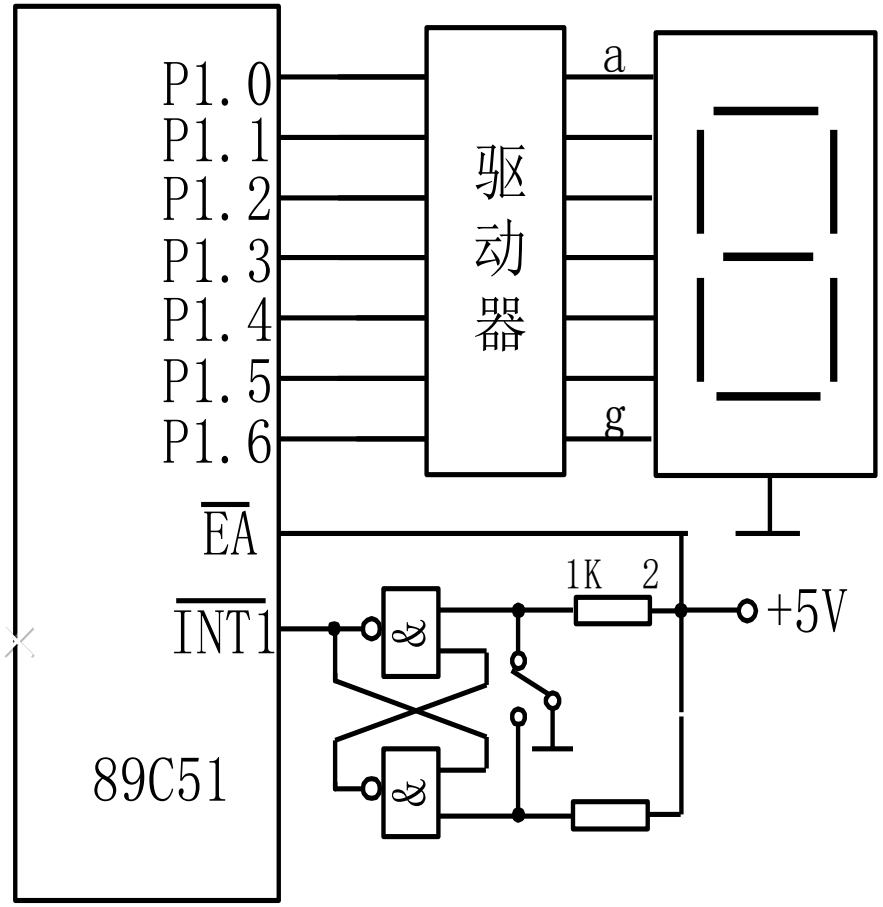


图6. 4

例10-16 记录并显示中断次数用C语言编程，可有两种编程方法。

法1：在主程序中判断中断次数,程序如下：

```
#include<reg51.h>
char i;
code char tab [16] = {0x3f,0x06,0x5b,0x4F,0x66,
  0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79
  ,0x71} ;
int() interrupt 2
 { i++;              /*计中断次数*/
 P1=tab [i] ; }     /*查表，次数送显示*/
main()
 { EA=1;
    EX1=1;
   IT1=1;
ap5: P1=0x3f         /*显示“0”*/
for(i=0;i<16;);      /*当i小于16等待中断*/
 goto ap5; }         /*当i=16重复下一轮16次中断*/
```

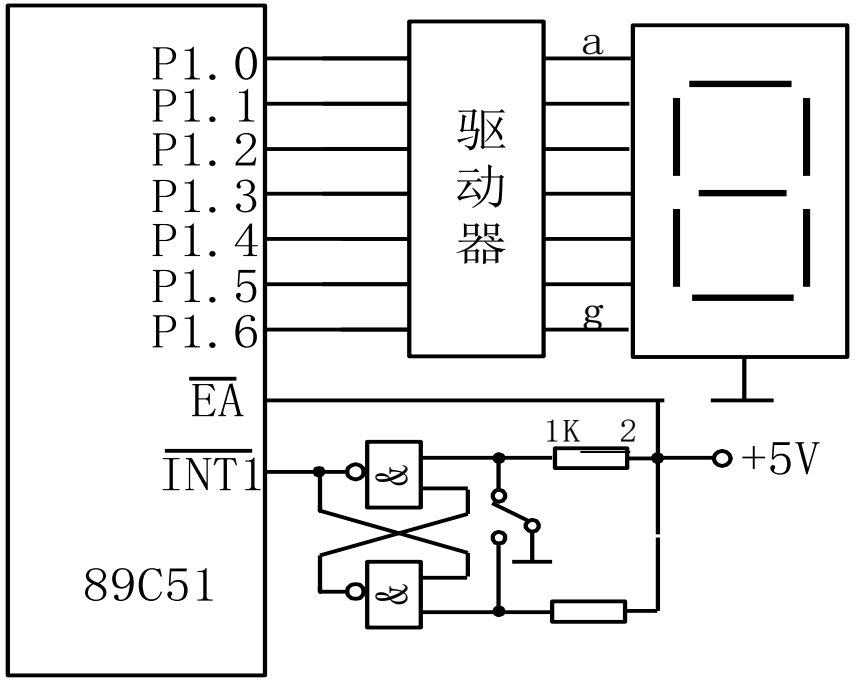


图6.4

法2：在中断程序中判断中断次数：

```
#include<reg51.h>
char i;
code char tab [16] = {0x3f,0x06,0x5b,0x4F,0x66, 0x6d,0x7d,0x07,
                      0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71} ;
int() interrupt 2
 { i++;
 if(i<16)P1=tab [i] ;
    else {i=0;P1=0x3f;} }
  main() {
  EA=1;
  EX1=1;
  IT1=1;
  P1=0x3f;
  while(1);  /*等待中断*/
  }
```

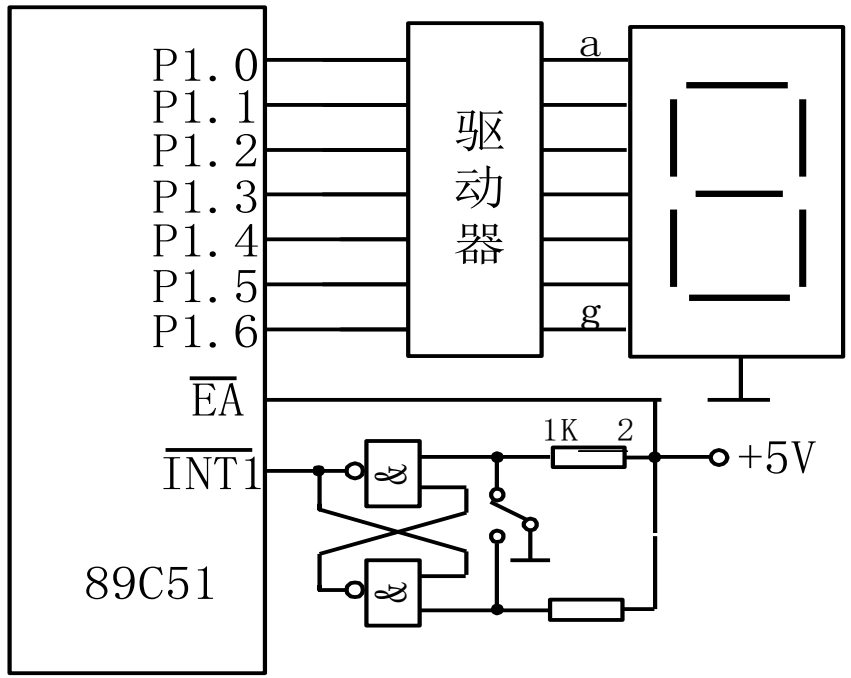


图6.4

4.3 小 结

（1）中断技术是实时控制中的常用技术，51系列单片机有三个内部中断，二个外部中断。所谓 外部中断就是在外部引脚上有产生中断所需要的信号。

每个中断源有固定的中断服务程序的入口地址(称矢量地址或向量地址)。当CPU响应中断以 后单片机内部硬件保证它能自动的跳转到该地址。因此，此地址是应该熟记的，在汇编程序 中，中断服务程序应存放在正确的向量地址内。

而在C语言中是靠Interrupt n的关键字n自动设置的。

（2）单片机的中断是靠内部的寄存器管理的，这就是中断允许寄存器IE，中断优先权寄存器IP，CPU必须在开全局中断开关EA，开各中断源的中断开关，CPU才能响应该中断源的中断请求，其中缺一不可。

（3）从程序表面看来，主程序和中断服务程序好象是没有关连的，只有掌握中断响应的过程，才能理解中断的发生和返回，看得懂中断程序，并能编写高质量中断程序。

（4）本章重点应掌握中断的基本概念，并能熟练编制中断程序。