第6章 中断系统原理和应用

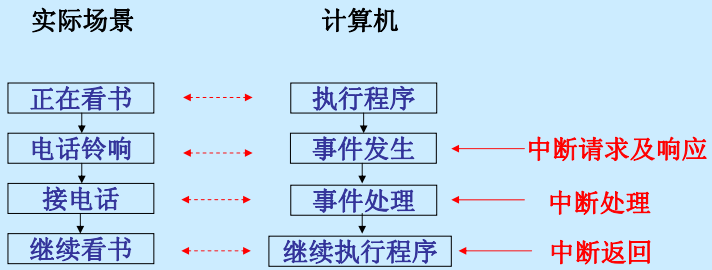
6.1 中断的概念

什么是中断，我们从一个生活中的例子引入。你正在家中看书，突然电话铃响了，你放下书本，去接电话，和来电话的人交谈，然后放下电话，回来继续看你的书。这就是生活中的“中断”的现象，就是正常的工作过程被外部的事件打断了。

6.1 中断的概念

定义：所谓中断是指中央处理器CPU正在处理某件事的时候，外部发生了某一事件，请求CPU迅速处理，CPU暂时中断当前的工作，转入处理所发生的事件，处理完后，再回到原来被中断的地方，继续原来的工作。这样的过程称为中断。

- 中断的基本概念
- 什么是中断?
  - 与生活场景的比较



必要性及应用

中断功能便于实现

1．分时操作

2．实时处理

3．故障处理

4. 主机与外设之间的速度匹配

CPU和外设同时工作；CPU可以通过分时操作启动多个外设同时工作，统一管理。大大提高了CPU的利用率，也提高了输入、输出的速度。

当计算机用于实时控制时，中断是一个十分重要的功能。现场的各个参数、信息，需要的话可在任何时候发出中断申请，要求CPU处理；CPU就可以马上响应（若中断是开放的话）加以处理。这样的及时处理在查询的工作方式是做不到的。

计算机在运行过程中，往往会出现事先预料不到的情况，或出现一些故障：如电源突跳，存储出错，运算溢出等等。计算机就可以利用中断系统自行处理，而不必停机或报告工作人员。

二　中断的概念

第一、什么可以引起中断，生活中很多事件可以引起中断：有人按了门铃了，电话铃响了，你的闹钟闹响了，你烧的水开了….等等诸如此类的事件，我们把可以引起中断的称之为中断源。

单片机中也有一些可以引起中断的事件，89S51中一共有5个：两个外部中断，两个计数/定时器中断，一个串行口中断。

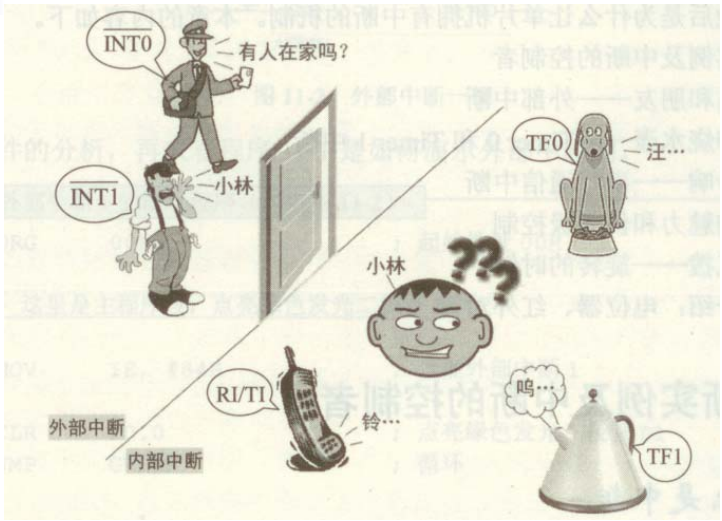


图　中断比喻

二　中断的概念

第二、中断的嵌套与优先级处理：设想一下，我们正在看书，电话铃响了，同时又有人按了门铃，你该先做那样呢？

存在一个优先级的问题。

单片机中也是如此，也有优先级的问题。优先级的问题不仅仅发生在两个中断同时产生的情况，也发生在一个中断已产生，又有一个中断产生的情况，比如你正接电话，有人按门铃的情况，或你正开门与人交谈，又有电话响了情况。考虑一下我们会怎么办吧。

二　中断的概念

当CPU正在处理一个中断请求，又发生了另一个优先级比它高的中断请求，CPU暂时中止对前一中断的处理，转而去处理优先级更高的中断请求，待处理完以后，再继续执行原来的中断处理程序。这样的过程称为中断嵌套，这样的中断系统称为多级中断系统。没有中断嵌套功能的中断系统称为单级中断系统。89S51五个中断源有两个优先级，可实现二级中断嵌套，二级中断嵌套的中断过程如图所示。

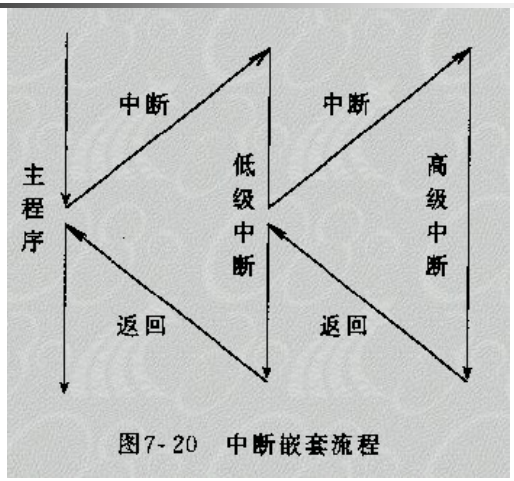


图7-20　中断嵌套流程

图　中断嵌套流程

二　中断的概念

第三、中断的响应过程：当有事件产生，进入中断之前我们必须先记住现在看书的第几页了，或拿一个书签放在当前页的位置，然后去处理不同的事情（因为处理完了，我们还要回来继续看书）：电话铃响我们要到放电话的地方去，门铃响我们要到门那边去，也说是不同的中断，我们要在不同的地点处理，而这个地点通常还是固定的。

计算机中也是采用的这种方法，五个中断源，每个中断产生后都到一个固定的地方去找处理这个中断的程序，当然在去之前首先要保存下面将执行的指令的地址，以便处理完中断后回到原来的地方继续往下执行程序。

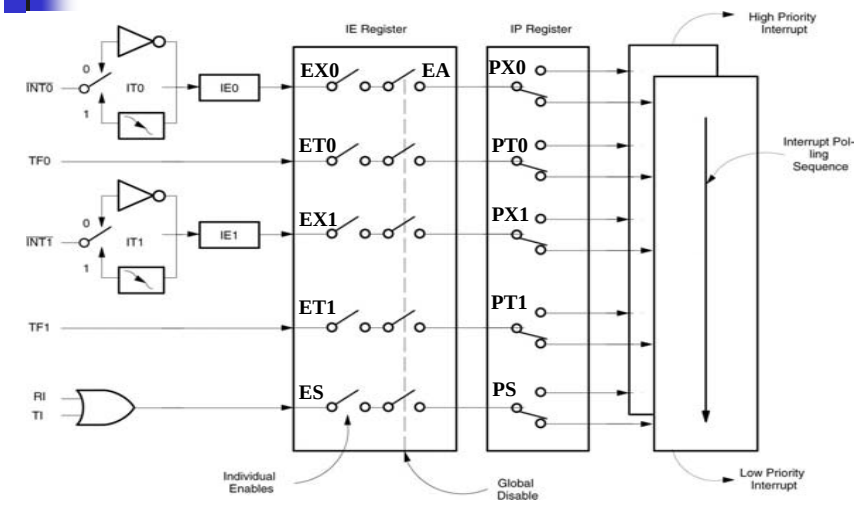
二　中断的概念

具体地说，中断响应可以分为以下几个步骤：1、保护断点，即保存下一将要执行的指令的地址，就是把这个地址送入堆栈。2、寻找中断入口，根据5个不同的中断源所产生的中断，查找5个不同的入口地址。以上工作是由计算机自动完成的，与编程者无关。在这5个入口地址处存放有中断处理程序（这是程序编写时放在那儿的，如果没把中断程序放在那儿，就错了，中断程序就不能被执行到）。3、执行中断处理程序。4、中断返回：执行完中断指令后，就从中断处返回到主程序，继续执行。

中断过程

- 五个步骤：
  - 中断请求
  - 中断判优(有时还要进行中断源识别)
  - 中断响应
  - 中断服务
  - 中断返回

89S51 的中断系统的结构



89S51的中断源和中断标志

89S51 共有5个中断源

- 外部中断0
- 外部中断1
- T0溢出中断
- T1溢出中断
- 串行口中断

外部中断请求0、1

中断请求信号输入引脚：

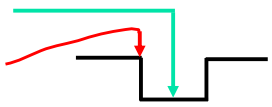
外部中断0请求引脚：$\overline{INT0}$(P3.2)

外部中断1请求引脚：$\overline{INT1}$(P3.3)

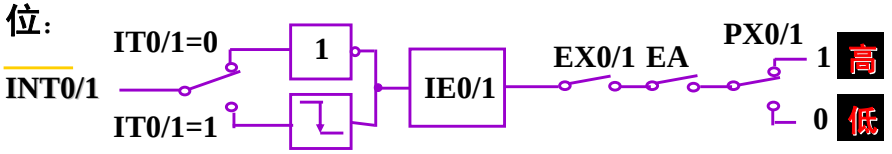
外部中断请求信号触发方式选择（IT0/1位）

电平触发：低电平

边沿触发：负边沿



外部中断请求标志位、中断允许位、优先级选择位：



TCON

D7							D0
TF1		TF0		IE1	IT1	IE0	IT0

- IT0（IT1）：外部中断请求0(1)的触发方式选择
  - IT0=0 电平触发方式；IT0=1 边沿触发方式
- IE0(IE1)：外部中断请求0(1)的中断申请标志
  - IT0=0 CPU每个机器周期采样/INT0，若/INT0=1 则IE0=0 否则/INT0=0， IE0=1申请中断
  - IT0=1 若第一个机器周期/INT0=1，第二个机器周期/INT0=0，则IE0=1申请中断，否则IE0=0
- 转向中断服务时边沿触发方式下IE由硬件清零
- 转向中断服务时电平触发方式下硬件不清IE，待申请信号撤消。

定时器计数器控制寄存器TCON

TCON

D7							D0
TF1		TF0		IE1	IT1	IE0	IT0

- TF0：51片内定时/计数器0溢出中断请求标志。
  - 定时/计数器0溢出时，TF0由硬件置1
  - CPU 响应中断时 自动清零 TF0
- TF1： 51片内定时/计数器1溢出中断请求标志

串行口控制寄存器SCON

SCON

D7						TI	D0

- RI：串行口接收中断标志

  RI=1串行口接收中断
- TI：串行口发送中断标志

  TI=1串行口发送中断
- RI、TI 由硬件置位

  必须由软件清零

中断允许寄存器IE

IE

D7							D0
EA			ES	ET1	EX1	ET0	EX0

- IE对中断开放和关闭实现两级控制。两级控制就是有一个总的中断开关控制位EA（IE.7位），当EA=0，所有中断请求被屏蔽，CPU对任何中断请求都不接受；当EA=1时，CPU开中断，但5个中断源的中断请求是否允许，还要由IE中的低5位所对应的5个中断请求允许控制位的状态来决定。

（1） EA—中断允许总开关控制位。

EA=0，所有的中断请求被屏蔽。

EA=1，所有的中断请求被开放。

（2）ES—串行口中断允许位。

ES=0，禁止串行口中断。 ES=1，允许串行口中断。

（3）ET1—定时器/计数器T1溢出中断允许位。

ET1=0，禁止T1溢出中断。ET1=1，允许T1溢出中断。

（4）EX1—外部中断1中断允许位。

EX1=0，禁止外部中断1中断。EX1=1，允许外部中断1中断。

（5）ET0—定时器/计数器T0的溢出中断允许位。

ET0=0，禁止T0溢出中断。ET0=1，允许T0溢出中断。

（6）EX0—外部中断0中断允许位。

EX0=0，禁止外部中断0中断。EX0=1，允许外部中断0中断。

中断优先级寄存器IP

	D7							D0
IP				PS	PT1	PX1	**PT0**	PX0

中断优先级寄存器IP各位含义：

（1）PS—串行口中断优先级控制位，1—高级；0—低级。

（2）PT1—T1中断优先级控制位，1—高级；0—低级。

（3）PX1—外部中断1中断优先级控制位，1—高级；0—低级。

（4）PT0—T0中断优先级控制位，1—高级；0—低级。

（5）PX0—外部中断0中断优先级控制位，1—高级；0—低级。

89S51 对中断请求的控制

(MSB) (LSB)

EA	X	X	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
	IE.6	Reserved.
	IE.5	Reserved.
ES	IE.4	Enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.
ET1	IE.3	Enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	Enables or disables External Interrupt 1. If EX1 = 0, External Interrupt 1 is disabled.
ET0	IE.1	Enables or disables the Timer 0 Overflow interrupt. If ET0 = 0, the Timer 0 interrupt is disabled.
EX0	IE.0	Enables or disables Exeternal Interrupt 0. If EX0 = 0, External Interrupt 0 is disabled.

Interrupt Enable (IE) Register

(MSB) (LSB)

X	X	X	PS	PT1	PX1	PT0	PX0

Symbol	Position	Function
	IP.7	Reserved.
	IP.6	Reserved.
	IP.5	Reserved.
PS	IP.4	Defines the Serial Port interrupt priority level. PS = 1 programs it to the higher priority level.
PT1	IP.3	Defines the Timer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
PX1	IP.2	Defines the External Interrupt 1 priority level. PX1 = 1 programs it to the higher priority level.
PT0	IP.1	Enables or disables the Timer 0 Interrupt priority level. PT) = 1 programs it to the higher priority level.
PX0	IP.0	Defines the External Interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

Interrupt Priority (IP) Register

总结中断源的各个触发器

中断源	中断标志1/0	中断允许1/0	中断级别1/0
外部$\overline{INT0}$	IE0 (TCON.1)	EX0(IE.0)	PX0 (IP.0)
外部$\overline{INT1}$	IE1(TCON.3)	EX1(IE.2)	PX1 (IP.2)
定时器0	TF0(TCON.5)	ET0(IE.1)	PT0 (IP.1)
定时器1	TF1 (TCON.7)	ET1(IE.3)	PT1 (IP.3)
串行口	RI (SCON.0) TI (SCON.1)	ES (IE.4)	PS (IP.4)
CPU标志		EA (IE.7)	

寄存器： **TCON、SCON 、IE 、IP**

复位后，这四个寄存器均为

总结中断源的各个触发器

寄存器： TCON、SCON 、IE 、IP

1.TCON(定时器/计数器的控制寄存器)88H

8FH							88H
TF1		TF0		IE1	IT1	IE0	IT0

2. SCON(串行口控制寄存器)98H

9FH						99H	98H
						TI	RI

3.IE(中断允许寄存器)A8H

AFH							A8H
EA			ES	ET1	EX1	ET0	EX0

4.IP(中断优先级寄存器)B8H

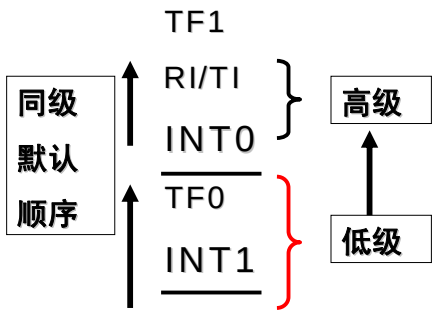
BFH							B8H
			PS	PT1	PX1	PT0	PX0

若设置串行口和定时器/计数器1为高级中断

问：设置后，那个中断源的优先级最高？

那个中断源的优先级最低？

答： 设置后，优先级的顺序为：



复位后(IP)=00H

5个中断源均为低级中断，同级默认顺序：

INT0
TF0
INT1
TF1
RI/TI

1、中断响应的条件：讲到这儿，我们依然对于计算机响应中断感到神奇，我们人可以响应外界的事件，是因为我们有多种“传感器“——眼、耳可以接受不同的信息，计算机是如何做到这点的呢？其实说穿了，一点都不希奇，89S51工作时，在每个机器周期中都会去查询一下各个中断标记，看他们是否是“1“，如果是1，就说明有中断请求了，所以所谓中断，其实也是查询，不过是每个周期都查一下而已。这要换成人来说，就相当于你在看书的时候，每一秒钟都会抬起头来看一看，查问一下，是不是有人按门铃，是否有电话。。。。很不智能，不是吗？可计算机本来就是这样，它根本没人聪明。

89S51中断响应的基本条件

- 89S51的中断响应的基本条件：
  - 首先要有中断源发出中断申请；
  - 中断总允许位EA=1，即CPU允许所有中断源申请中断；
  - 在中断源寄存器TCON和SCON中，申请中断的中断允许位为1，

即此中断源可以向CPU申请中断。

89S51中断响应的基本条件

了解了上述中断的过程，就不难解中断响应的条件了。在下列三种情况之一时，CPU将封锁对中断的响应：

1.CPU正在处理一个同级或更高级别的中断请求。

2.现行的机器周期不是当前正执行指令的最后一个周期。我们知道，单片机有单周期、双周期、三周期指令，当前执行指令是单字节没有关系，如果是双字节或四字节的，就要等整条指令都执行完了，才能响应中断（因为中断查询是在每个机器周期都可能查到的）。

3.当前正执行的指令是返回批令或访问IP、IE寄存器的指令，则CPU至少再执行一条指令才应中断。这些都是与中断有关的，如果正访问IP、IE则可能会开、关中断或改变中断的优先级，而中断返回指令则说明本次中断还没有处理完，所以都要等本指令处理结束，再执行一条指令才可以响应中断。

89S51响应中断的过程

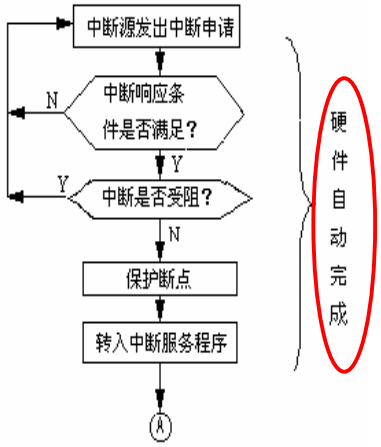
A 51CPU自动完成：

- CPU先在每个机器周期的S5P2期间，对各中断源重复进行查询，并设置相应的中断标志位 。
- 如果中断响应条件满足，且不存在中断阻断的情况，则CPU就响应中断。
- 硬件生成长调用指令自动地把断点地址压入堆栈保护，并随之将对应的中断入口装入程序计数器PC，使程序转向该入口地址，以执行中断服务程序。

用户必须完成：

在这些入口地址存放一条无条件跳转指令，使程序跳转到用户安排的中断服务程序起始地址上去 。

89S51 的中断响应过程



中断入口地址表

中断源	入口地址	同级中断优先级
IE0	0003H	
TF0	000BH	
IE1	0013H	
TF1	001BH	
TI/RI	0023H	

用户编程设计

低级中断源的中断服务程序

- 保护现场
  - 保护中断服务程序中用到的寄存器和状态标志的内容
- 中断服务
  - 相应的中断源服务，完成一定的I/O操作
- 恢复现场
  - 完成中断服务后，将保存在堆栈中的现场数据恢复
- 开中断和中断返回
  - 中断返回指令RETI(汇编)

2、中断响应过程

首先由硬件自动生成一条长调用指令“LCALL addr16”。即程序存储区中相应的中断入口地址。例如，对于外部中断1的响应，硬件自动生成的长调用指令为：

LCALL 0013H

生成LCALL指令后，紧接着就由CPU执行该指令。首先将程序计数器PC内容压入堆栈以保护断点，再将中断入口地址装入PC，使程序转向响应中断请求的中断入口地址。各中断源服务程序入口地址是固定的。

其中两个中断入口间只相隔8字节，一般情况下难以安放一个完整的中断服务程序。

因此，通常总是在中断入口地址处放置一条无条件转移指令，使程序执行转向在其他地址存放的中断服务程序入口。

中断响应是有条件的，并不是查询到的所有中断请求都能被立即响应，当遇到下列3种情况之一时，中断响应被封锁：

（1）CPU正在处理同级或更高优先级的中断。

（2）所查询的机器周期不是当前正在执行指令的最后一个机器周期。

（3）正在执行的指令是RETI或是访问IE或IP的指令。

子程序与中断服务程序的区别

- 对强迫中断的服务程序具有随机性
  - 要考虑可能在程序的什么指令处发生，
  - 要保护什么内容，才能保证返回断点后正常工作。
- 对人为设置的软件中断与子程序调用的区别
  - 返回指令不同
    - 子程序返回用RET
    - 中断服务程序返回用RETI
  - 处理内容不同，中断一般处理I/O操作。

89S51中断系统的初始化及应用

1、89S51中断系统的初始化

中断系统初始化步骤为：

1）CPU开中断或关中断；

2）某中断源中断请求的允许或禁止（屏蔽）；

3）设定所用中断的中断优先级；

4）若为外部中断，则应规定低电平还是负边沿的中断触发方式。

中断过程

- 五个步骤：
  - 中断请求(IE0,IE1, TF0, TF1, TI, RI)
  - 中断判优(有时还要进行中断源识别)(IE, IP)
  - 中断响应
  - 中断服务
  - 中断返回

6.6 外部中断的响应时间

外部中断的最短的响应时间为3个机器周期：

（1）中断请求标志位查询占1个机器周期。

（2）子程序调用指令LCALL转到相应的中断服务程序入口，需2个机器周期。

外部中断响应的最长的响应时间为8个机器周期：

（1）发生在CPU进行中断标志查询时，刚好是开始执行RETI或是访问IE或IP的指令，则需把当前指令执行完再继续执行一条指令后，才能响应中断，最长需2个机器周期。

（2）接着再执行一条指令，按最长指令（乘法指令MUL和除法指令DIV）来算，也只有4个机器周期。

（3）加上硬件子程序调用指令LCALL的执行，需要2个机器周期。

所以，外部中断响应最长时间为8个机器周期。

如果已在处理同级或更高级中断，响应时间无法计算。

在一个单一中断的系统里，89S51单片机对外部中断请求的响应的时间总是在3～8个机器周期之间。

6.7 外部中断的触发方式选择

两种触发方式：电平触发方式和跳沿触发方式。

6.7.1 电平触发方式

CPU在每个机器周期采样到的外部中断输入线的电平。在中断服务程序返回之前，外部中断请求输入必须无效（即变为高电平），否则CPU返回主程序后会再次响应中断。

适于外中断以低电平输入且中断服务程序能清除外部中断请求（即外部中断输入电平又变为高电平）的情况。

6.7.2 跳沿触发方式

连续两次采样，一个机器周期采样到外部中断输入为高，下一个机器周期采样为低，则置“1”中断请求标志，直到CPU响应此中断时，该标志才清0。这样不会丢失中断，但输入的负脉冲宽度至少保持1个机器周期。

6.8 中断请求的撤消

1．定时器/计数器中断请求的撤消

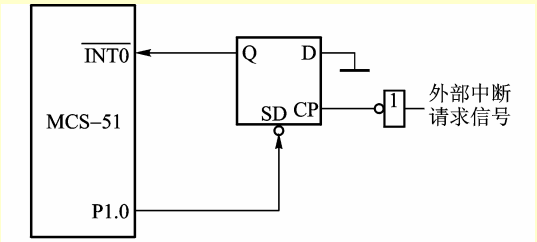
中断请求被响应后。硬件会自动清TF0或TF1。

2．外部中断请求的撤消

（1）跳沿方式外部中断请求的撤消是自动撤消的。

（2）电平方式外部中断请求的撤消:

除了标志位清“0”之外，还需在中断响应后把中断请求信号引脚从低电平强制改变为高电平，如图5-8所示。



5-8

只要P1.0端输出一个负脉冲就可以使D触发器置“1”，从而撤消了低电平的中断请求信号。

电平方式的外部中断请求信号的完全撤消，是通过软硬件相结合的方法来实现的。

3．串行口中断请求的撤消

响应串行口的中断后，CPU无法知道是接收中断还是发送中断，还需测试这两个中断标志位的状态，以判定是接收操作还是发送操作，然后才能清除。所以串行口中断请求的撤消只能用软件在中断服务程序中把串行口中断标志位TI、RI清0。

6.9 中断函数

为直接使用C51编写中断服务程序，C51中定义了中断函数。减小编写中断服务程序烦琐程度。

中断服务函数的一般形式为：

函数类型 函数名（形式参数表）interrupt n using n

关键字interrupt后面的 n是中断号，对于8051单片机，n的取值为0～4，编译器从8×n+3处产生中断向量。AT89S51中断源对应的中断号和中断向量见表6-3。

AT89S51内部RAM中可使用4个工作寄存器区，每个工作寄存器区包含8个工作寄存器（R0～R7）。关键字using后面的n用来选择4个工作寄存器区。using是一选项，如不选，中断函数中的所有工作寄存器内容将被保存到堆栈中。

表6-3 8051单片机的中断号和中断向量

中断号n	中 断 源	中断向量（8×n+3）
0	外部中断 0	0003H
1	定时器 0	000BH
2	外部中断 1	0013H
3	定时器 1	001BH
4	串行口	0023H
其他值	保留	8×n+3

关键字using对函数目标代码的影响如下：

在中断函数的入口处将当前工作寄存器区内容保护到堆栈中，函数返回前将被保护的寄存器区内容从堆栈中恢复。使用using在函数中确定一个工作寄存器区须十分小心，要保证任何工作寄存器区的切换都只在指定的控制区域中发生，否则将产生不正确的函数结果。

编写中断程序，应遵循以下规则：

（1）中断函数没有返回值，如果定义一个返回值，将会得到不正确结果。建议将中断函数定义为void类型，明确说明无返回值。

（2）中断函数不能进行参数传递，如果中断函数中包含任何参数声明都将导致编译出错。

（3）任何情况下都不能直接调用中断函数，否则会产生编译错误。因为中断函数的返回是由汇编语言指令RETI完成的。RETI指令会影响AT89S51硬件中断系统内的不可寻址的中断优先级寄存器的状态。如没有实际中断请求情况下，直接调用中断函数，也就不会执行RETI指令，其操作结果有可能产生一个致命错误。

（4）如在中断函数中再调用其他函数，则被调用的函数所用的寄存器区必须与中断函数使用的寄存器区不同。

6.10 中断服务程序的设计

一、中断服务程序设计的任务

基本任务：

（1）设置中断允许控制寄存器IE。

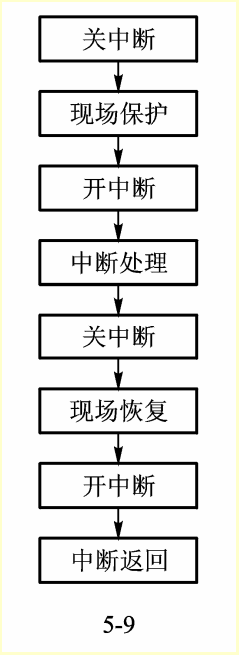
（2）设置中断优先级寄存器IP。

（3）对外中断源，是采用电平触发还是跳沿触发。

（4）编写中断服务程序，处理中断请求。

前2条一般放在主程序的初始化程序段中。

二、中断服务程序的流程



5-9

--- 中断应用举例

【例6-1】 在单片机P1口上接有8只LED。在外部中断0输入引脚（P3.2）接一只按钮开关K1。要求将外部中断0设置为电平触发。程序启动时，P1口上的8只LED全亮。每按一次按钮开关K1，使引脚接地，产生一个低电平触发的外中断请求，在中断服务程序中，让低4位的LED与高4位的LED交替闪烁5次。然后从中断返回，控制8只LED再次全亮。原理电路及仿真结果见下图6-1。

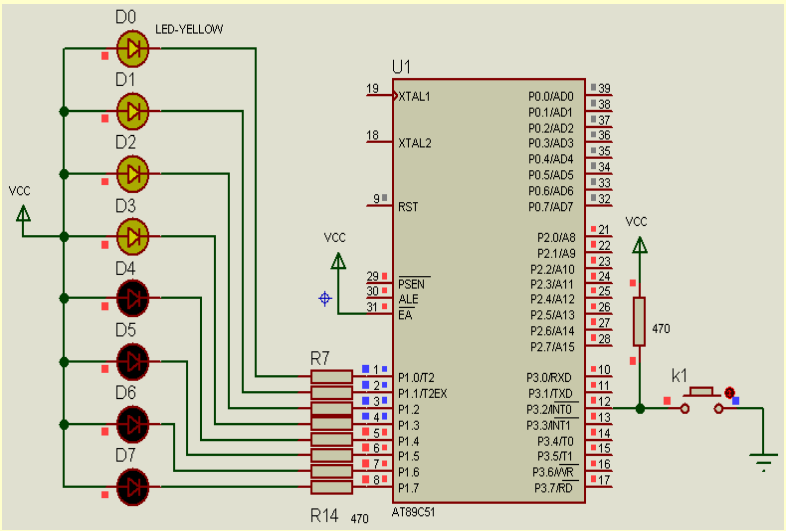


图6-1　利用中断控制8只LED交替闪烁1次的电路

参考程序如下：

```
#include <reg51.h>
#define uchar  unsigned char
void Delay(unsigned int i)
                //延时函数Delay( )，i形式参数，不能赋初值
{
    unsigned int j;
    for(;i > 0;i--)
    for(j=0;j<333;j++)
          //晶振为12MHz，j选择与晶振频率有关
    {;}                          //空函数
}
```

```
void  main( )                 //主函数
    {
       EA=1;                  //总中断允许
       EX0=1;                 //允许外部中断0中断
       IT0=1;                 //选外部中断0为跳沿触发方式
       while(1)               //循环
  { P1=0;}                    // P1口的8只LED全亮
    }
  void int0( ) interrupt 0  using 0
                              //外中断0的中断服务函数
```



```
{
       uchar  m;
       EX0=0;                 //禁止外部中断0中断
       for(m=0;m<5;m++)       //交替闪烁5次
       {
          P1=0x0f;            //低4位LED灭，高4位LED亮
       Delay(400) ;           //延时
       P1=0xf0;               //高4位LED灭，低4位LED亮
       Delay(400);            //延时
       EX0=1;                 //中断返回前，开外部中断0中断
    }
```

本例程包含两部分，一部分是主程序段，完成中断系统初始化，并把8个LED全部点亮。另一部分是中断函数部分，控制4个LED交替闪烁1次，然后从中断返回。

当需要多个中断源时，只需增加相应的中断服务函数即可。例6-2是处理两个外中断请求的例子。

【例6-2】如图6-2所示，在单片机P1口上接有8只LED。在外部中断0输入引脚（P3.2）接有一只按钮开关K1。在外部中断1输入引脚（P3.3）接有一只按钮开关K2。要求K1和K2都未按下时，P1口的8只LED呈流水灯显示，仅K1（P3.2）按下再松开时，上下各4只LED交替闪烁10次，然后再回到流水灯显示。如果按下再松开K2（P3.3）时，P1口的8只LED全部闪烁10次，然后再回到流水灯显示。设置两个外中断的优先级相同。

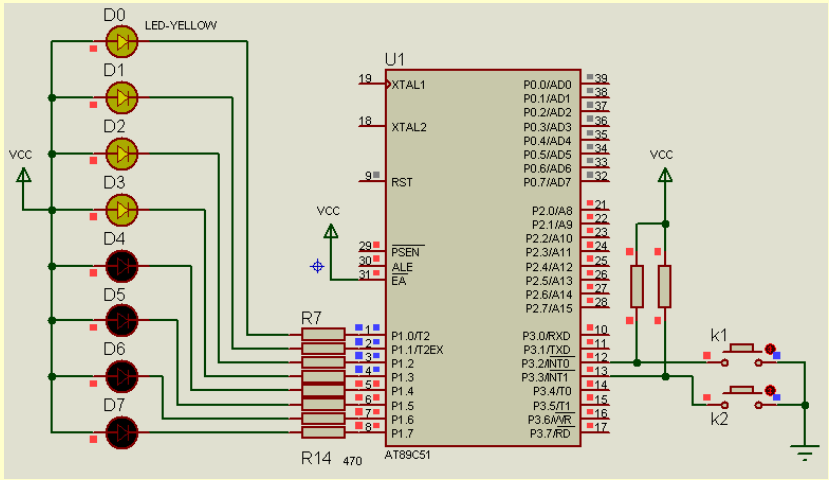


图6-2　两个外中断控制8只LED显示的电路

参考程序如下：

```
#include <reg51.h>
#define uchar unsigned char
void Delay(unsigned int i)      //延时函数Delay( ),i为形式参
                                //数，不能赋初值
{
   uchar j;
   for(;i>0;i--)
   for(j=0;j<125;j++)
   {;}                          //空函数
}
```



```
void  main( )                           //主函数
{
uchar display[9]={0xff,0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf, 0x7f};
                                //流水灯显示数据数组
unsigned int a;
for(;;)
{
     EA=1;                      //总中断允许
     EX0=1;                     //允许外部中断0中断
     EX1=1;                     //允许外部中断1中断
     IT0=1;                     //选择外部中断0为跳沿触发方式
     IT1=1;                     //选择外部中断1为跳沿触发方式
```



```
     IP=0;                  //两个外部中断均为低优先级
     for(a=0;a<9;a++)
     {
          Delay(500);       //延时
          P1=display[a];    //将已经定义的流水灯显示
数据送到P1口
     }
     }
 }
```



```
void int0_isr(void)  interrupt 0  using 1//外中断0的中断服务函
数
   {  uchar  n;
      for(n=0;n<10;n++)   //高、低4位显示10次
      {
           P1=0x0f;       //低4位LED灭，高4位LED亮
           Delay(500);    //延时
           P1=0xf0;       //高4位LED灭，低4位LED亮
           Delay(500);    //延时
      }
   }
```

```
 void int1_isr (void)  interrupt 2  using 2//外中断1中断服务函数
    {
      uchar  m;
      for(m=0;m<10;m++)                  //闪烁显示10次
      {
            P1=0xff;                     //全灭
            Delay(500);                  //延时
            P1=0;                        //全亮
            Delay(500);                  //延时
      }
    }
```


中断嵌套只发生正执行一个低优先级中断，此时又有一高优先级中断产生，就会去执行高优先级中断服务程序。高优先级中断服务程序完成后，再继续执行低优先级中断程序。

【例6-3】电路同图6-2，设计一中断嵌套程序：要求K1和K2都未按下时，P1口8只LED呈流水灯显示，当按一下K1时，产生一个低优先级外中断0请求（跳沿触发），进入外中断0中断服务程序，上下4只LED交替闪烁。此时按一下K2时，产生一个高优先级的外中断1请求（跳沿触发），进入外中断1中断服务程序，使8只LED全部闪烁。当显示5次后，再从外中断1返回继续执行外中断0中断服务程序，即P1口控制8只LED，上、下4只LED交替闪烁。设置外中断0为低优先级，外中断1为高优先级。

参考程序如下：

```
#include <reg51.h>
#define uchar unsigned char
void Delay(unsigned int i)          //延时函数Delay( )
{
     unsigned int j;
     for(;i > 0;i--)
   for(j=0;j<125;j++)
     {;}                            //空函数
}
void  main( )                       //主函数
  {
     uchar display[9]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};
                                    //流水灯显示数据组
      uchar a;
```



```
for(;;)
{
            EA=1;                   //总中断允许
            EX0=1;                  //允许外部中断0中断
            EX1=1;                  //允许外部中断1中断
            IT0=1;                  //选择外部中断0为跳沿触发方式
            IT1=1;                  //选择外部中断1为跳沿触发方式
            PX0=0;                  //外部中断0为低优先级
            PX1=1;                  //外部中断1为高优先级
for(a=0;a<9;a++)
{
     Delay(500);                    //延时
     P1=display[a];                 //流水灯显示数据送到P1口驱动LED显示
  }
```

```
}
}
void int0_isr(void)  interrupt 0  using 0   //外中断0中断函数
{
    for(;;)
    {
        P1=0x0f;              //低4位LED灭，高4位LED亮
        Delay(400);           //延时
        P1=0xf0;              //高4位LED灭，低4位LED亮
        Delay(400);           //延时
    }
}
```



```
void int1_isr (void)  interrupt 2  using 1  //外中断1中断函数
{
    uchar m;
    for(m=0;m<5;m++)              //8位LED全亮全灭5次
    {
        P1=0;                     //8位LED全亮
        Delay(500);               //延时
        P1=0xff;                  //8位LED全灭
        Delay(500);               //延时
    }
}
```

本例如设置外中断1为低优先级，外中断0为高优先级，仍然先按下再松开K1，后按下再松开K2或者设置两个外中断源的中断优先级为同级，均不会发生中断嵌套。

思考题:

判断下列说法的正误。

1. 中断允许寄存器 IE 的最高位(EA)为 1 是 MCS-51 系统响应中断的重要条件。
2. TCON 及 SCON 寄存器内的 6 个中断标志位都可以用软件指令对其置位或复位。
3. 8031 的$\overline{INT_0}$信号只能以“保持足够宽度的低电平”这一种方式来触发中断。
4. 中断响应过程中必定有堆栈操作。
5. MCS-51 单片机中,外部中断源的优先级比内部中断源的优先级高。
6. 当多个中断源同时请求服务时,CPU 将响应优先级别最高的中断申请,这就意味着其他低级中断申请信号无效,再也不会被响应。

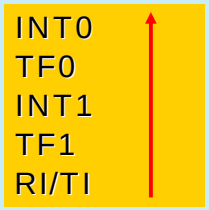
思考题:

答 1. 错误。EA 只是中断总允许位,还要与各个源允许位配合才能真正开中断。因此,EA=1 只是单片机系统响应中断的必要条件,而非重要条件。2. 正确。因为 TCON 和 SCON 都在片内的 SFR 区,且既可字节寻址又可位寻址,用户完全可用指令改写。3. 错误。外部中断源可以有两种触发方式,除了低电平触发外,还可以通过改变 TCON 中的有关位将其设置为下跳沿触发。4. 正确。当 CPU 响应中断时,会自动将当前主程序断点处的 PC 值压入堆栈,等中断服务完成时,又会自动将该 PC 值弹出堆栈,从而实现保护断点和正确返主的功能。5. 错误。优先级别的高低可以通过改变 IP 相应位而重新安排。即使是所有中断源处于同一级别,内部中断源定时器 0 的自然优先级别也比外部中断 1 要高。6. 错误。只要有关的中断请求标志位(在 TCON/SCON 中)不被改变,CPU 就会在结束前一个中断服务子程序之后马上响应其他请求中优先级别较高的一个,并依次顺延响应下去。

思考题:

若所有的中断源同时发出中断请求,下列哪些情况的中断优先顺序能够实现?

1. 外部中断 1>定时器 T_0 中断>外部中断 0。
2. 串行口中断>定时器 T_0 中断>外部中断 1。
3. 定时器 T_0 中断>定时器 T_1 中断>外部中断 0。
4. 定时器 T_0 中断>定时器 T_1 中断>串行口中断。



答 1. 不能实现。2. 可以实现。只要将串行口中断源设为高优先级,其他两个设为低优先级即可。3. 可以实现。只要将外部中断 0 设为低优先级,其他两个设为高优先级即可。4. 可以实现。这三个中断源符合自然优先顺序,即使处于同一级别也会实现。