



# 第7章 中断与定时技术





# 本章内容

1. 中断的基本概念
2. 8086的中断系统
3. 可编程中断控制器8259
4. 8259的应用举例
5. 硬件中断服务程序的编写
6. 定时与计数技术





# 7.1 中断的基本概念





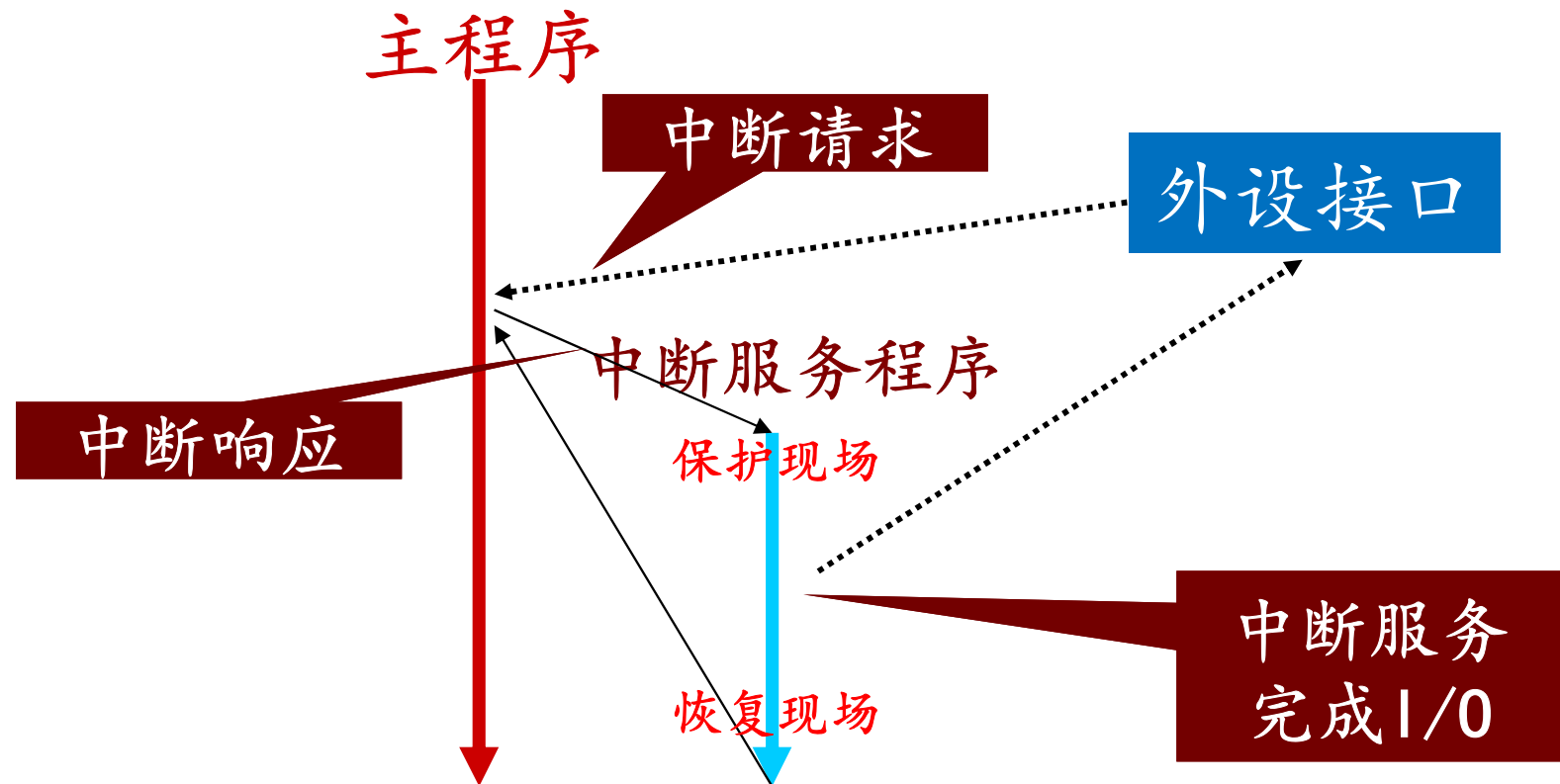
# 什么是中断?

## ■ 软中断

- ◆ 是用软中断指令来激活的

## ■ 硬中断

- ◆ 中断的产生具有随机性, 改变程序的执行顺序





# 中断源

## ■ 内部中断源

- ◆ 指令中断
- ◆ 出错中断
- ◆ 调试中断

## ■ 外部中断源

- ◆ 可屏蔽中断
- ◆ 不可屏蔽中断





# 中断识别

## ■ 什么是中断识别？

- ◆ CPU管理多个中断源时，在收到中断源发出的中断请求后，需判断是哪一个中断源提出的中断请求，以便对它进行服务（或处理）

## ■ 中断识别的方法

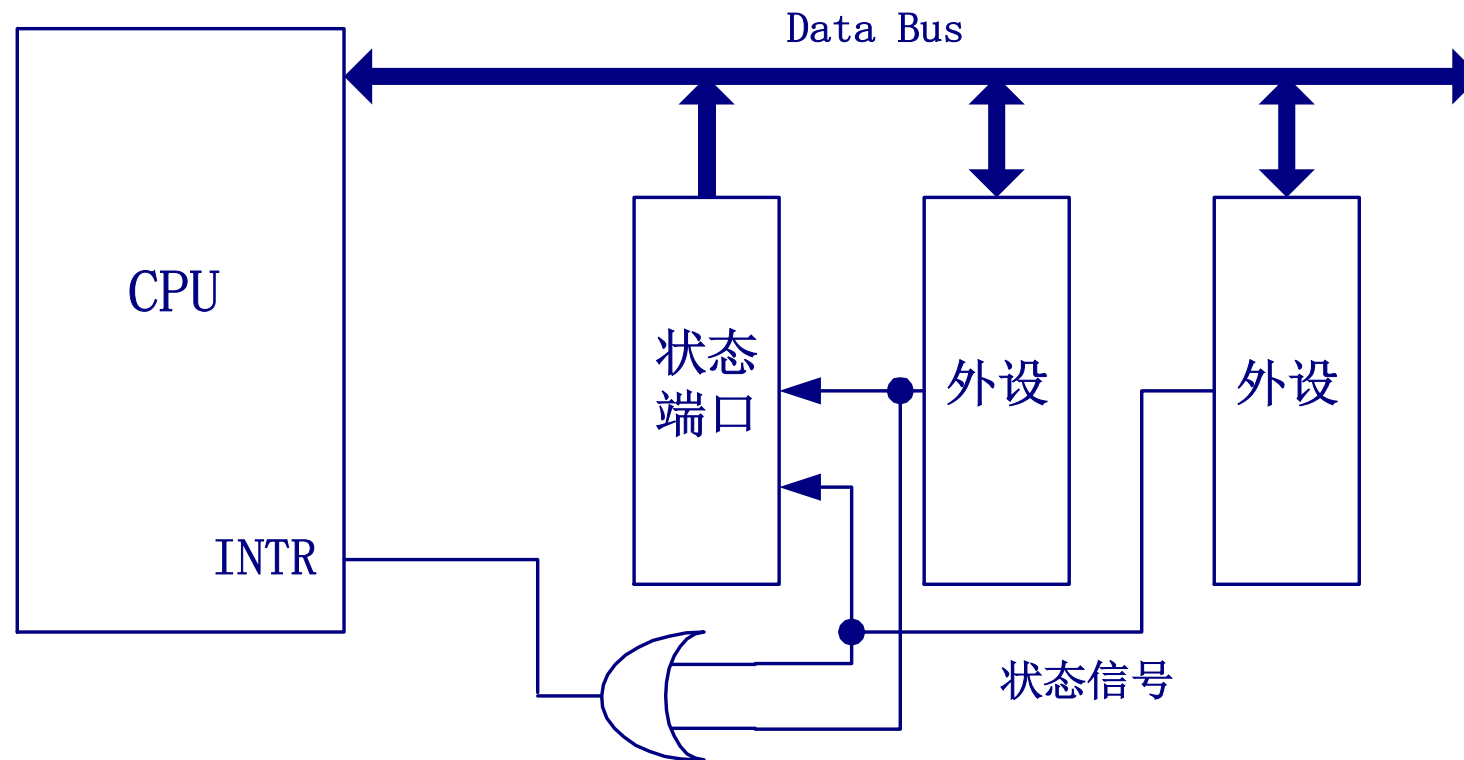
- ◆ 查询中断法
- ◆ 向量中断法





# 1. 查询中断法

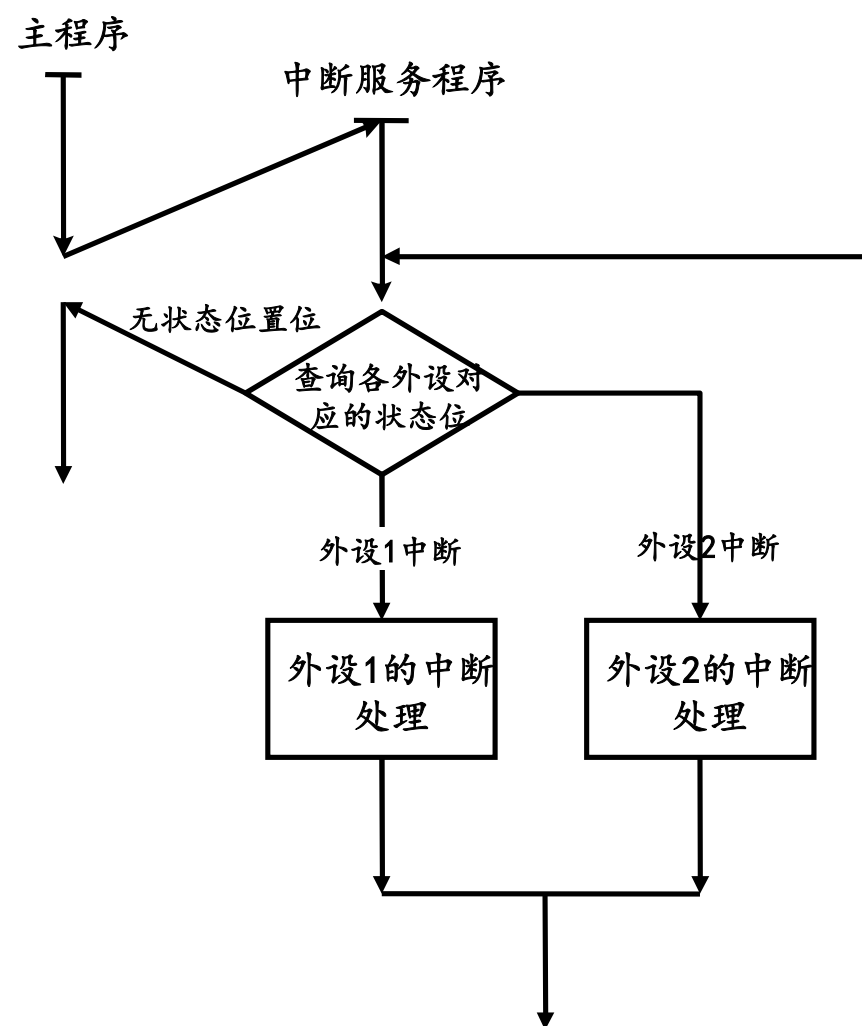
**CPU**响应中断后，转入执行一特定地址的中断服务程序，该中断服务程序查询状态口，确定发出中断请求的外设，然后进行相应的处理。





# 查询中断服务程序

外设的中断优先级由查询的次序决定

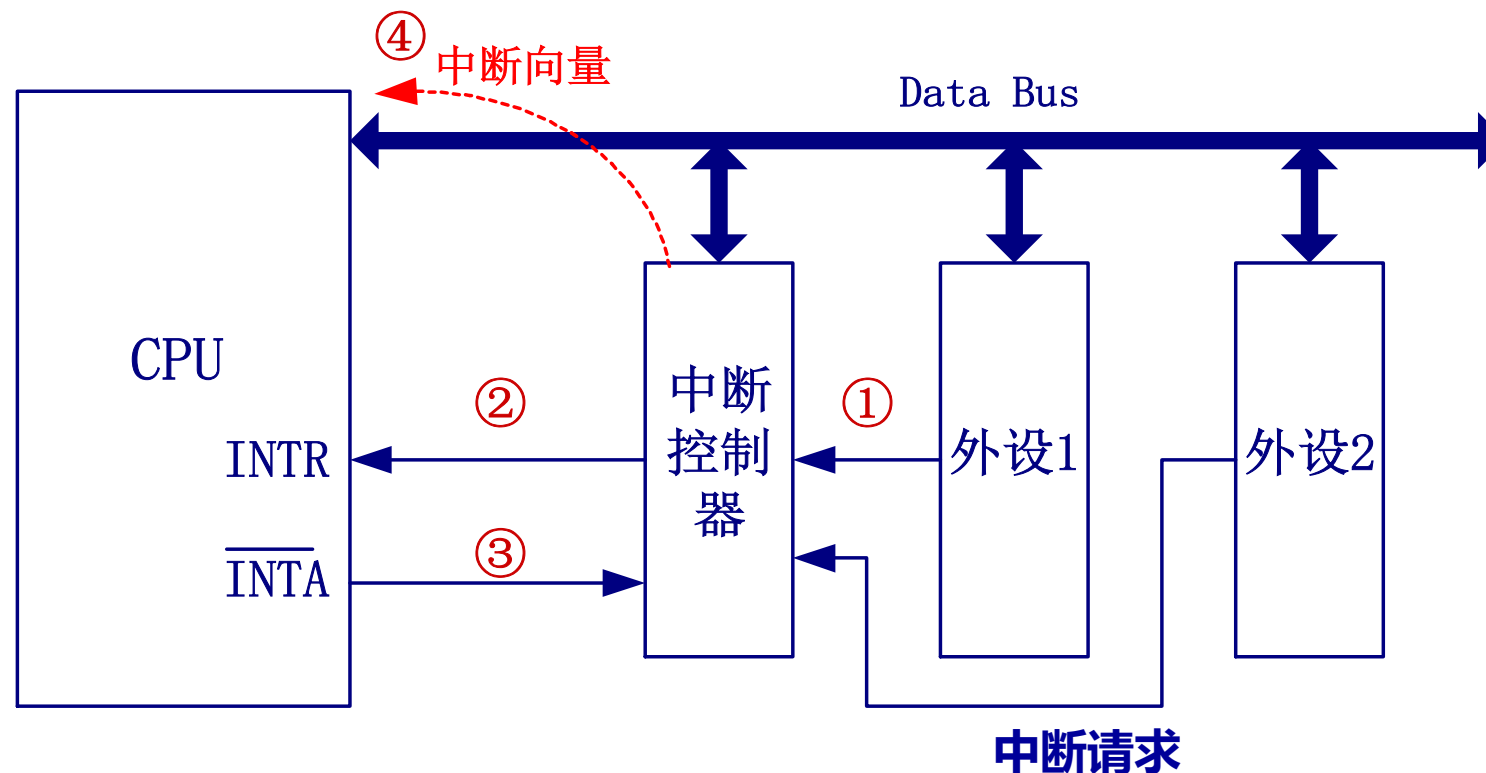






## 2. 向量中断法

多个外设经中断控制器向CPU提出中断请求，CPU响应中断发出INTA信号，中断控制器将相应的中断向量号（中断类型号）放在数据总线上，CPU读取后，即可确定中断源，查中断向量表进行相应处理。







# 中断向量表的修改 (1)

将中断类型号为**60H**的中断服务程序**intr**入口地址填入中断向量表

## 1. 直接修改中断向量表

```
xor ax,ax
```

```
mov es, ax
```

```
mov bx, 60h*4
```

;中断向量号x4

```
mov ax, offset intr
```

;中断服务程序的偏移地址

```
mov es:[bx], ax
```

```
mov ax, seg intr
```

;中断服务程序的段地址

```
mov es:[bx+2], ax
```





# 中断向量表的修改 (2)

## 2. 利用DOS调用修改中断向量表

```
mov ax, 3560h           ; 取原中断向量入口地址
int 21h                 ; ES:BX = 入口地址
mov old_off, bx
mov bx, es
mov old_seg, bx
.....

mov ax, 2560h           ; 置中断向量入口地址
mov dx, seg intr
mov ds, dx
mov dx, offset intr      ; DS:DX = 入口地址
int 21h
```





# 中断向量表的修改 (3)

```
mov ax, 2560h           ; 恢复原中断向量入口地址
mov dx, old_seg
mov ds, dx
mov dx, old_off          ; DS:DX = 入口地址
int 21h
```





# 中断优先级

当多个中断源同时提出中断请求时，**CPU**在一个时刻只能响应并处理一个中断请求，因此，响应优先级最高的中断请求。中断源的优先级可按如下方式确定：

- ◆ 按优先级排队

根据预先确定的原则，对每一中断源指定优先级。例如：  
软件的查询顺序，硬件方式

- ◆ 按轮循排队

所有中断源的优先级相等





# 中断嵌套

- 当前正在被执行的中断服务程序可被优先级更高的中断请求中断
- 优先级相同或更低的中断请求不能中断当前正在被执行的中断服务程序





## 7.2 8086中断系统







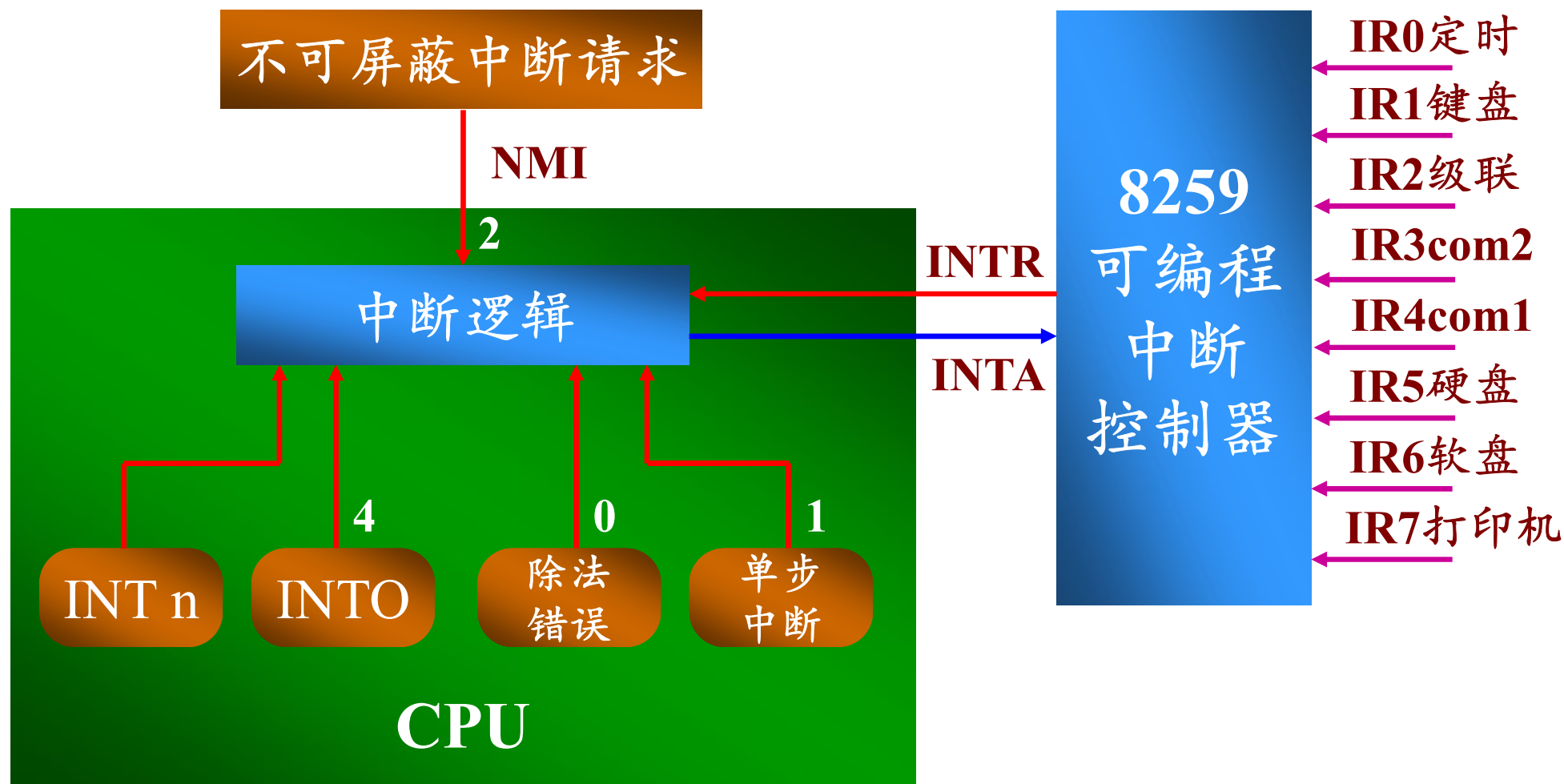
# 中断源类型

- 外部中断（硬中断）
  - ◆ 不可屏蔽中断NMI
  - ◆ 可屏蔽中断INT
- 内部中断（软中断）
  - ◆ 除法错中断
  - ◆ 溢出错中断
  - ◆ 指令中断
  - ◆ 单步中断





# 80x86中断源





# 硬中断与软中断的比较

## ■ 硬中断的特点

- ◆ 由外部事件引起，具有随机性
- ◆ **CPU**需发中断响应信号
- ◆ 可以被屏蔽

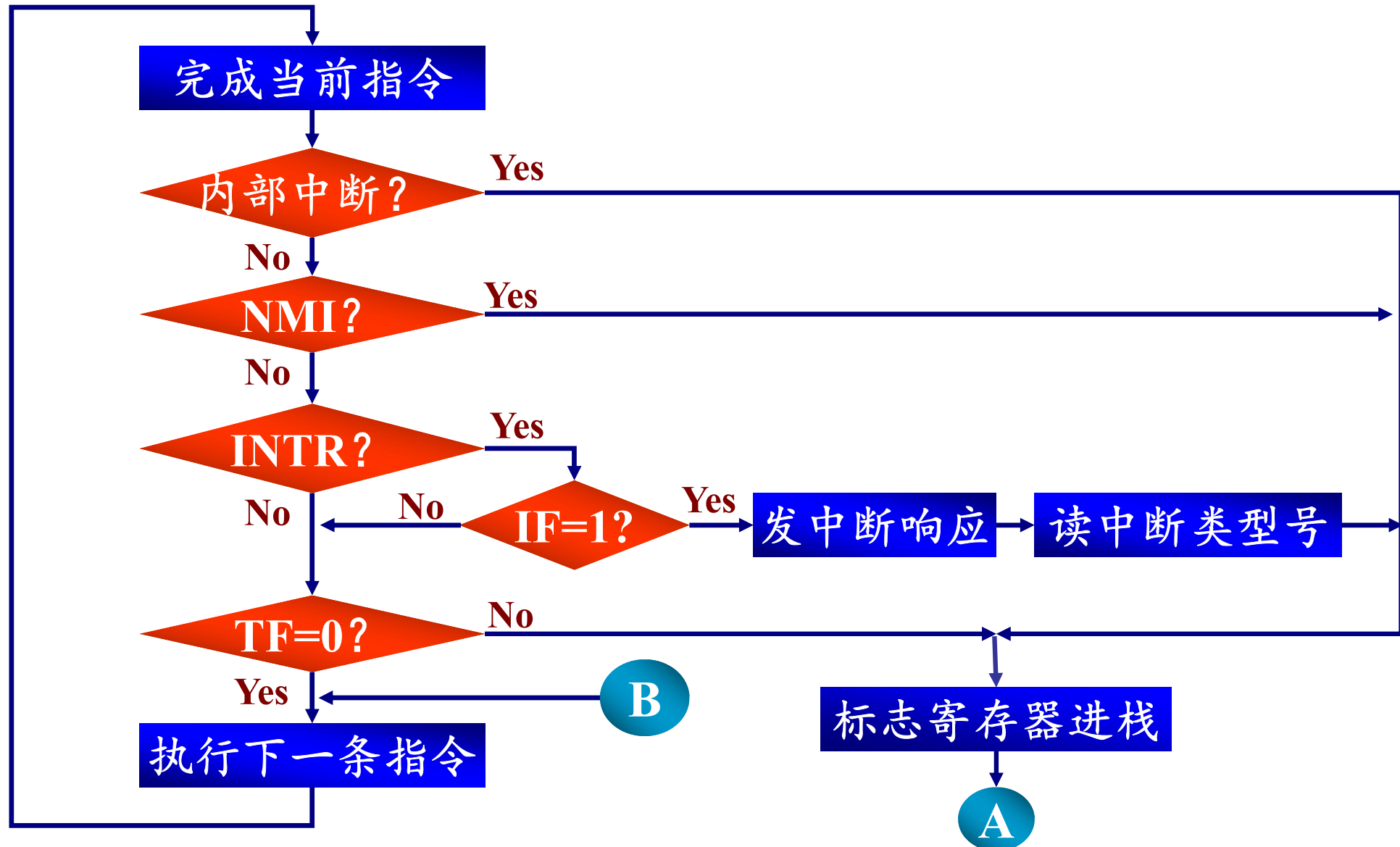
## ■ 软中断的特点

- ◆ 通常用软中断指令触发，中断的发生时刻是可知的
- ◆ **CPU**不发中断响应信号
- ◆ 中断类型号由指令直接给出
- ◆ 不可被屏蔽





# 8086中断处理过程 (1)





# 8086中断处理过程 (2)

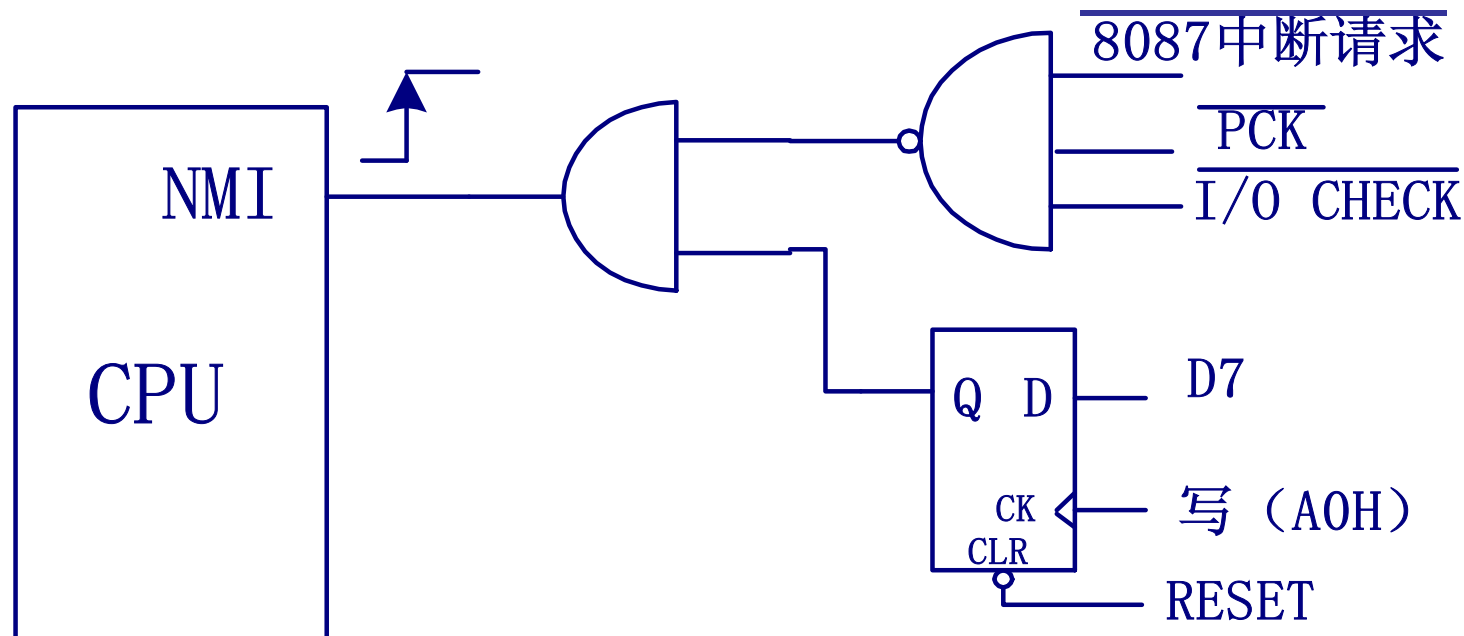




# 1. 不可屏蔽中断NMI (1)

CPU收到不可屏蔽中断NMI请求后，其默认的中断类型为02H。PC机里包括3个非屏蔽中断源

- ◆ 协处理器出错
- ◆ 系统RAM奇偶校验错
- ◆ I/O通道校验错





# 不可屏蔽中断NMI (2)

在PC机系统设计时，允许NMI请求被屏蔽

- ◆ 复位后或向A0H端口写00H，禁止NMI请求

```
mov al, 00h
```

```
out 0a0h, al
```

- ◆ 向A0H端口写80H，允许NMI请求

```
mov al, 80h
```

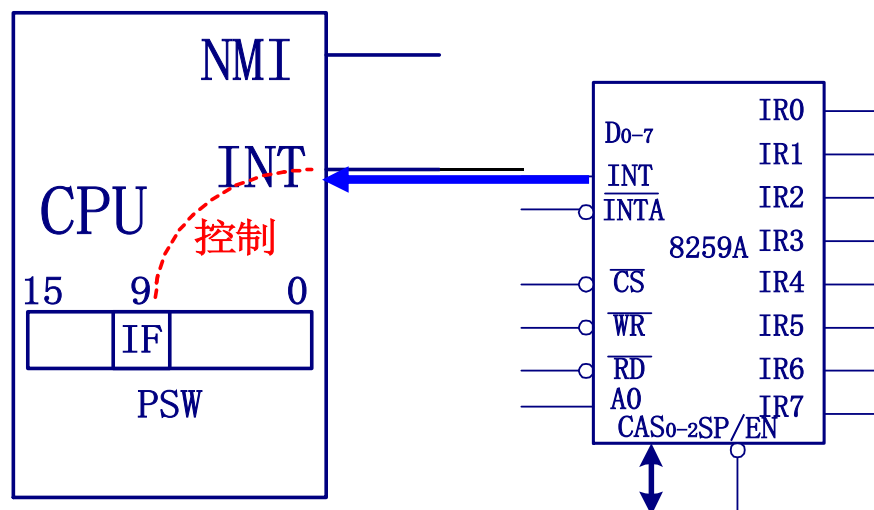
```
out 0a0h, al
```





## 2. 可屏蔽中断INTR

- 在CPU的INT引脚输入“高”有效信号时，则产生硬件可屏蔽中断请求信号。
- 是否响应该请求由PSW寄存器的IF位决定。
  - ◆ IF=0，屏蔽中断（执行CLI指令）
  - ◆ IF=1，允许中断（执行STI指令）
- 使用中断控制器可管理多个硬件中断源



**CPU响应可屏蔽中断条件：**

- 1) 当前指令执行完毕
- 2) IF=1
- 3) INTR信号有效







# 3. 内部中断（软中断）

## ■ 除法错中断

- ◆ 执行**DIV**或**IDIV**指令时，商超出机器表示的最大值，即产生**0**号中断

## ■ 溢出错中断

- ◆ 当**PSW**的**OF=1**时，执行**INTO**指令，即产生**4**号中断

## ■ 单步中断

- ◆ 当**PSW**的**TF=1**时，执行每条指令，即产生**1**号中断

## ■ 指令中断

- ◆ 断点中断 **INT 3H**（机器码为：**CCH**），单字节指令
- ◆ **INT nH**





# 中断处理过程（1）

## ■ 中断申请

- ◆ 外设向**CPU**发中断请求信号申请**CPU**给予服务，**CPU**决定是否响应

## ■ 中断响应

- ◆ **CPU**发中断响应信号，获取中断类型号，保存**PSW**和返回地址于堆栈，查表获得中断服务程序入口地址

## ■ 中断服务程序

- ◆ 保护现场，对外设进行服务（**I/O**操作），恢复现场

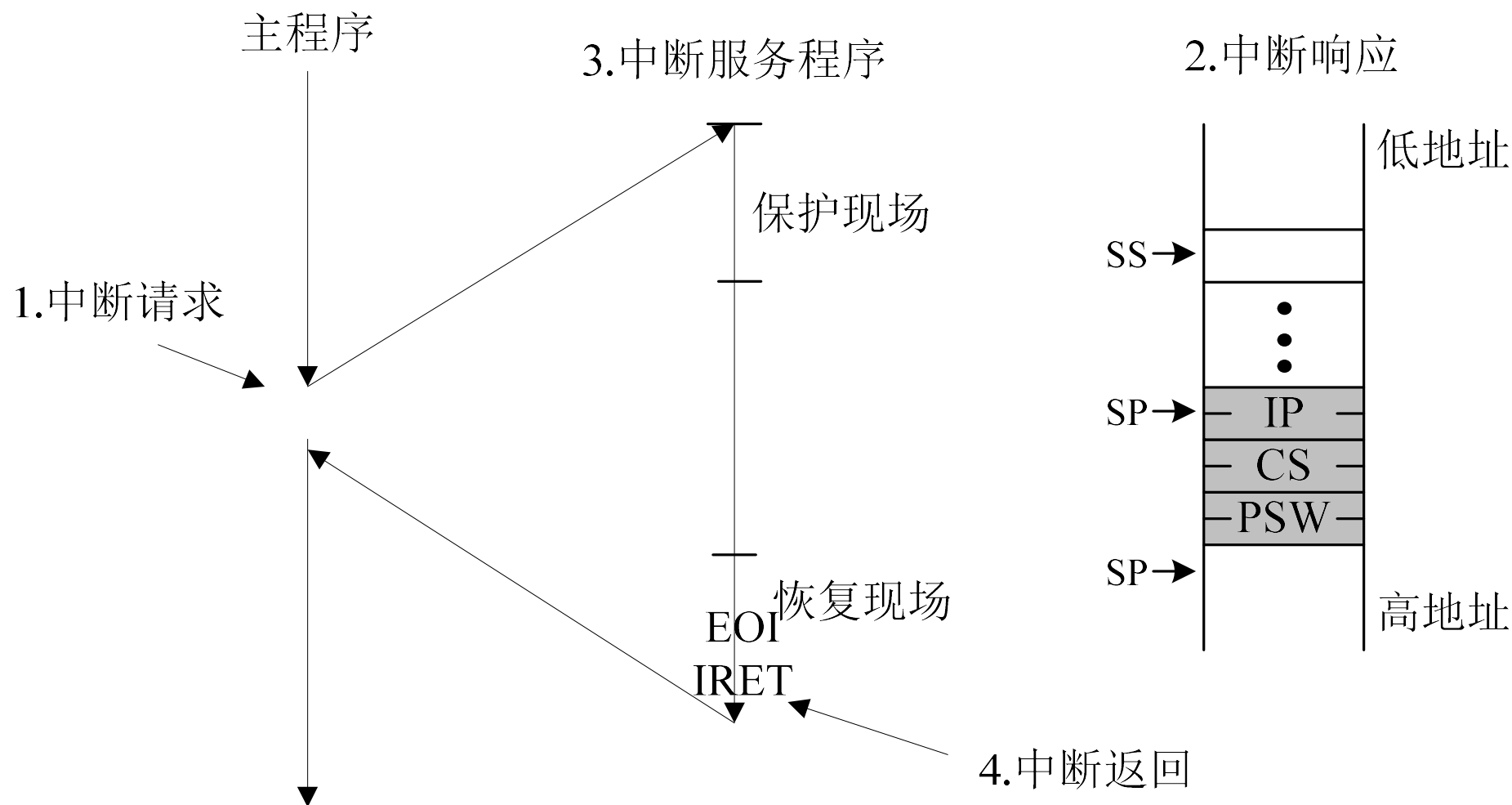
## ■ 中断返回

- ◆ 从堆栈弹出返回及**PSW**，回到中断前的下一条指令继续执行





# 中断处理过程 (2)





## 7.3 可编程中断控制器8259A





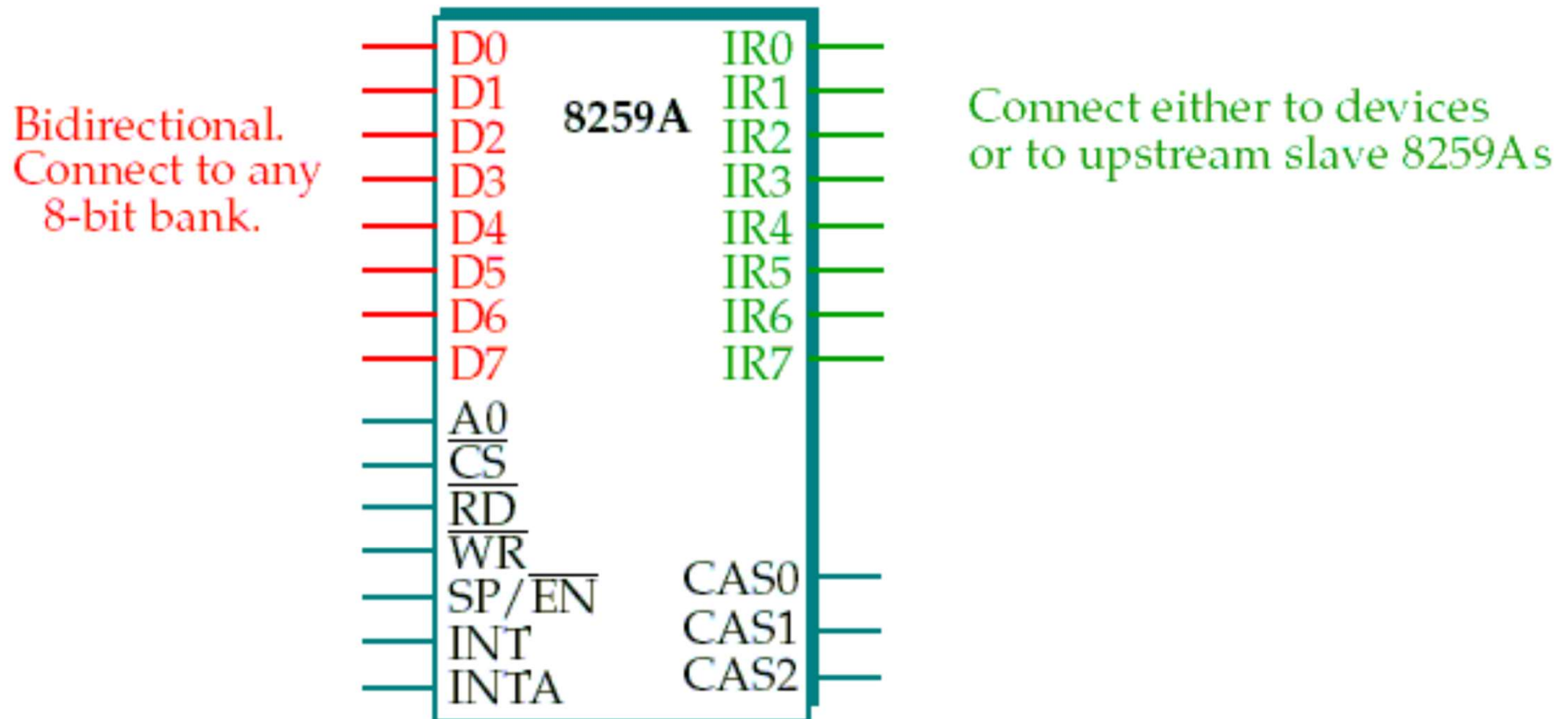
# 8259A中断控制器功能

- 优先级排队管理
  - ◆ 完全嵌套
  - ◆ 循环优先级
  - ◆ 特殊完全嵌套方式
- 接受和扩充外部设备的中断请求
  - ◆ 利用级联方式可扩展至8片，管理64个中断源
- 提供中断类型号
- 中断请求的允许与屏蔽





# 8259A的引脚图



$\overline{CS}$  and  $\overline{WR}$  must be decoded. Other connections are direct to micro.





# 8259A引脚定义及功能（1）

- $\overline{\text{WR}}$  (In)

写信号线

- $\overline{\text{RD}}$  (In)

读信号线

- $\text{INT}$  (Out)

中断请求线

- $\overline{\text{INTA}}$  (In)

中断应答线

- $\text{A0}$

用作芯片内的端口地址指示





# 8259A引脚定义及功能（2）

## ■ $\overline{\text{CS}}$ (In)

片选信号线。用于使能8259A芯片。

## ■ $\overline{\text{SP}}/\overline{\text{EN}}$ (In/Out)

从方式编程（1标识主方式，0标识从方式）/使能缓冲器（当工作在缓冲器模式时用于控制数据总线上的收发器的传送方向）

## ■ $\text{CAS2} \sim \text{CAS0}$ (In/Out)

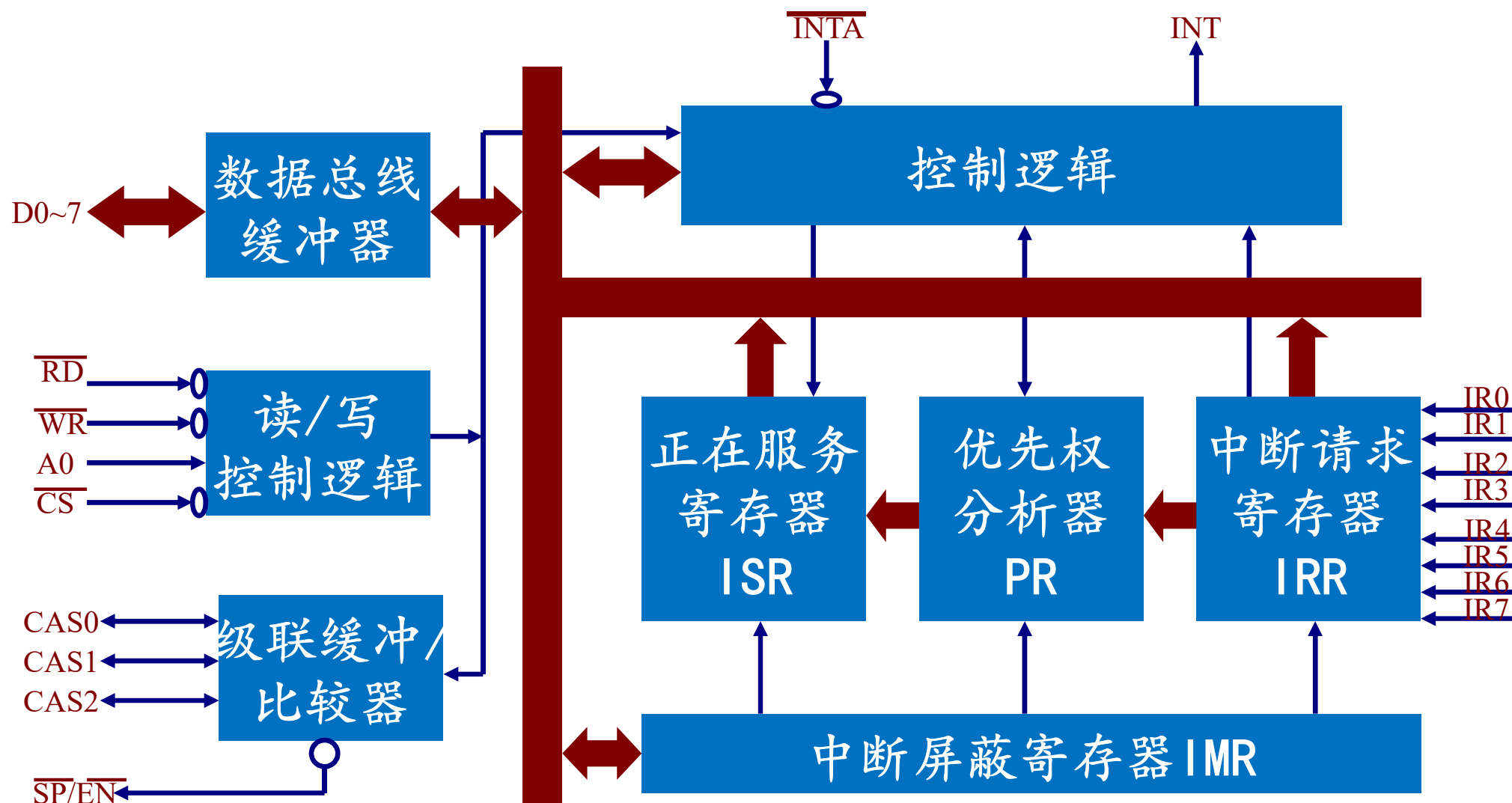
级联信号线。主片为输出，从片为输入。







# 8259A的内部结构 (1)





# 8259A的内部结构（2）

## ■ 中断请求寄存器（IRR）

- ◆ 存放外部中断源发出的中断请求信号，**Di**位为**1**表示**IRi**引脚有中断请求。具有锁存功能

## ■ 正在服务寄存器（ISR）

- ◆ 存放正在被服务中的中断请求信号。**Di**位为**1**表示**IRi**中断正在服务中。中断嵌套时，有多个比特同时被置“1”

## ■ 中断屏蔽寄存器（IMR）

- ◆ “0”允许中断；“1”屏蔽中断





# 8259A的内部结构 (3)

## ■ 优先权分析器 (PR)

- ◆ 把IRR的内容与ISR的内容进行比较, 响应优先级高的中断

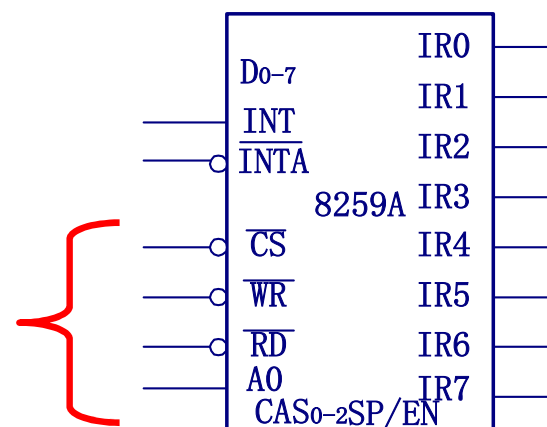
## ■ 数据总线缓冲器

- ◆ 三态双向8位缓冲器作为与系统总线的接口

## ■ 读/写控制逻辑

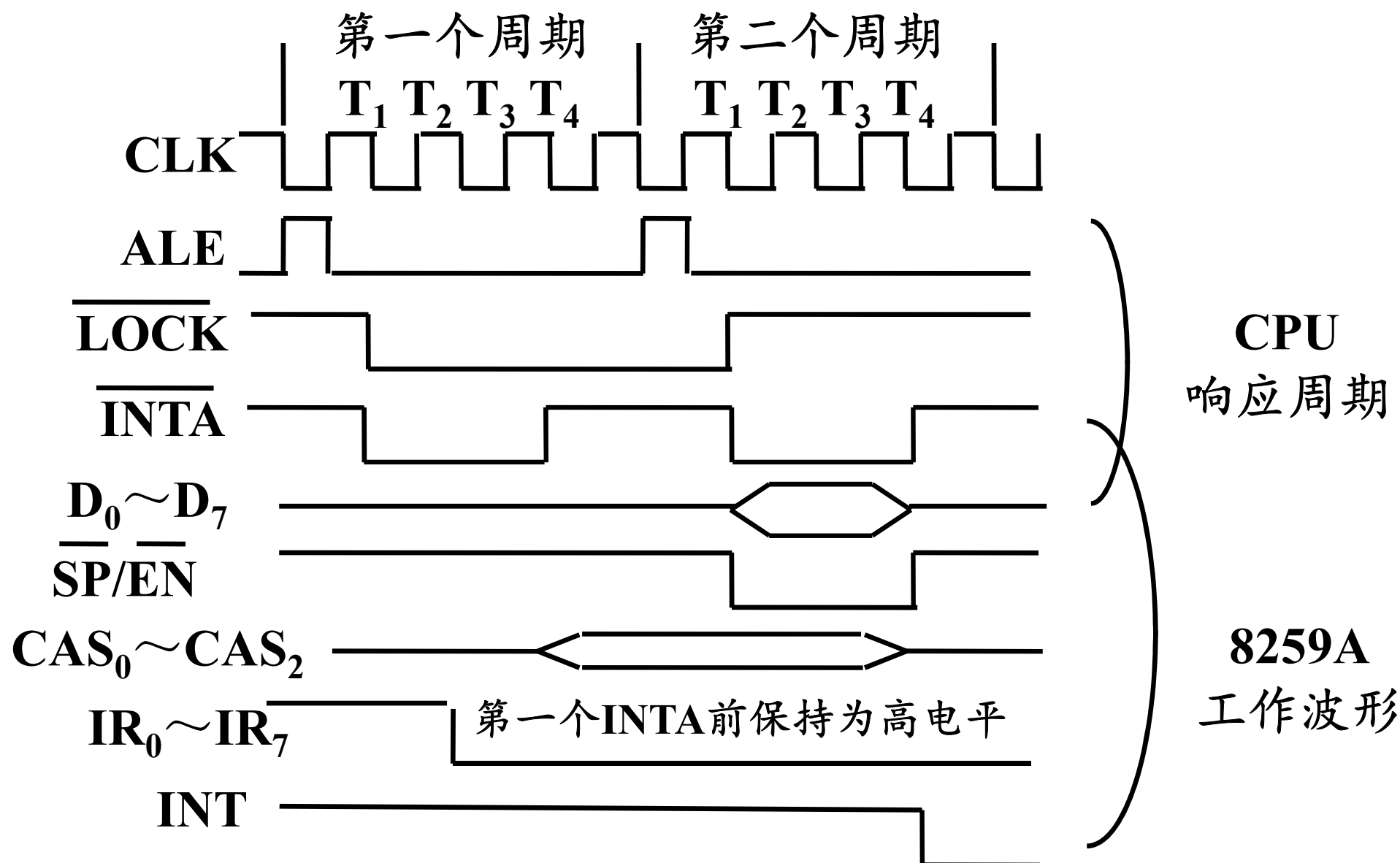
## ■ 级联缓冲器/比较器

- ◆ 用于存储和比较系统中所有8259的标识号





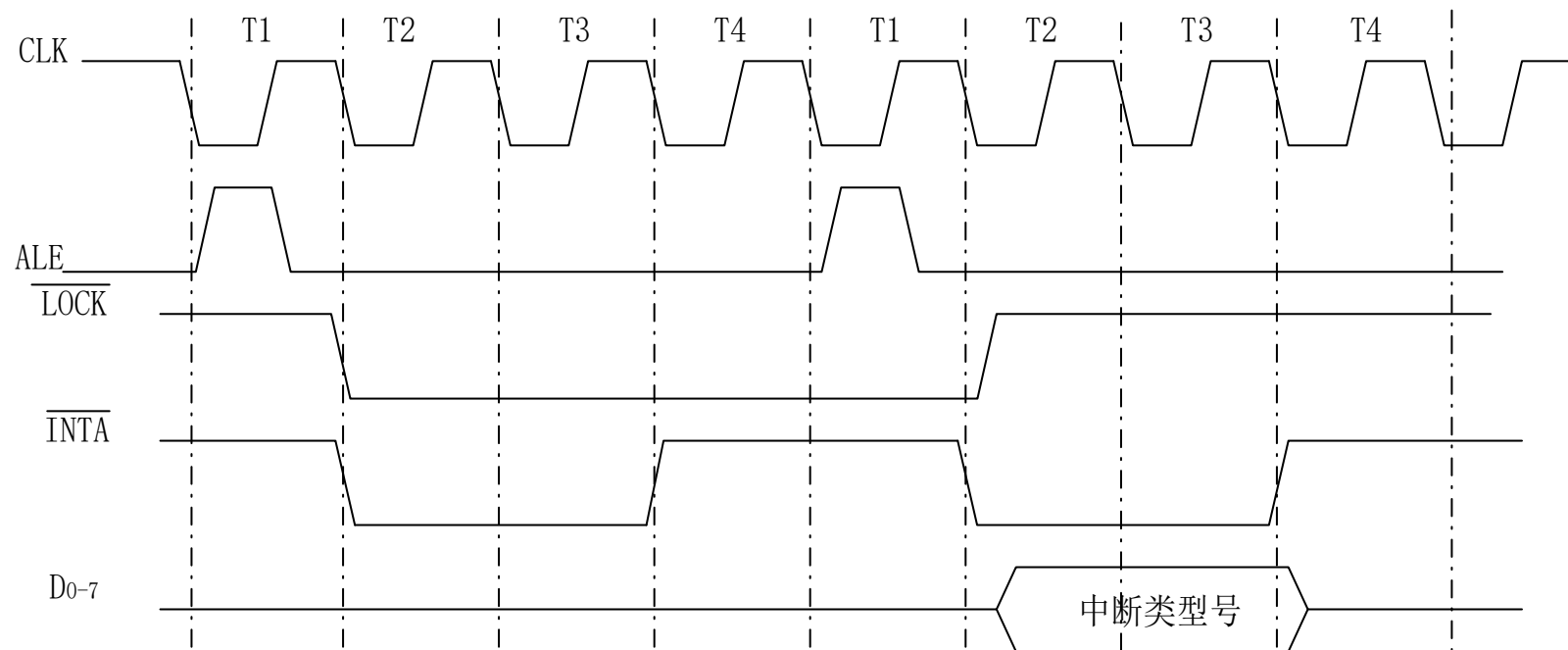
# 中断响应周期时序





# 中断响应过程说明

- 第一个INTA，CPU封锁总线（LOCK有效），8259A优先级最高的请求所对应的ISR中的位置1，而相应IRR中的位随之复位
- 第二个INTA，总线解锁，8259A将当前中断请求对应的中断类型号送到数据总线上

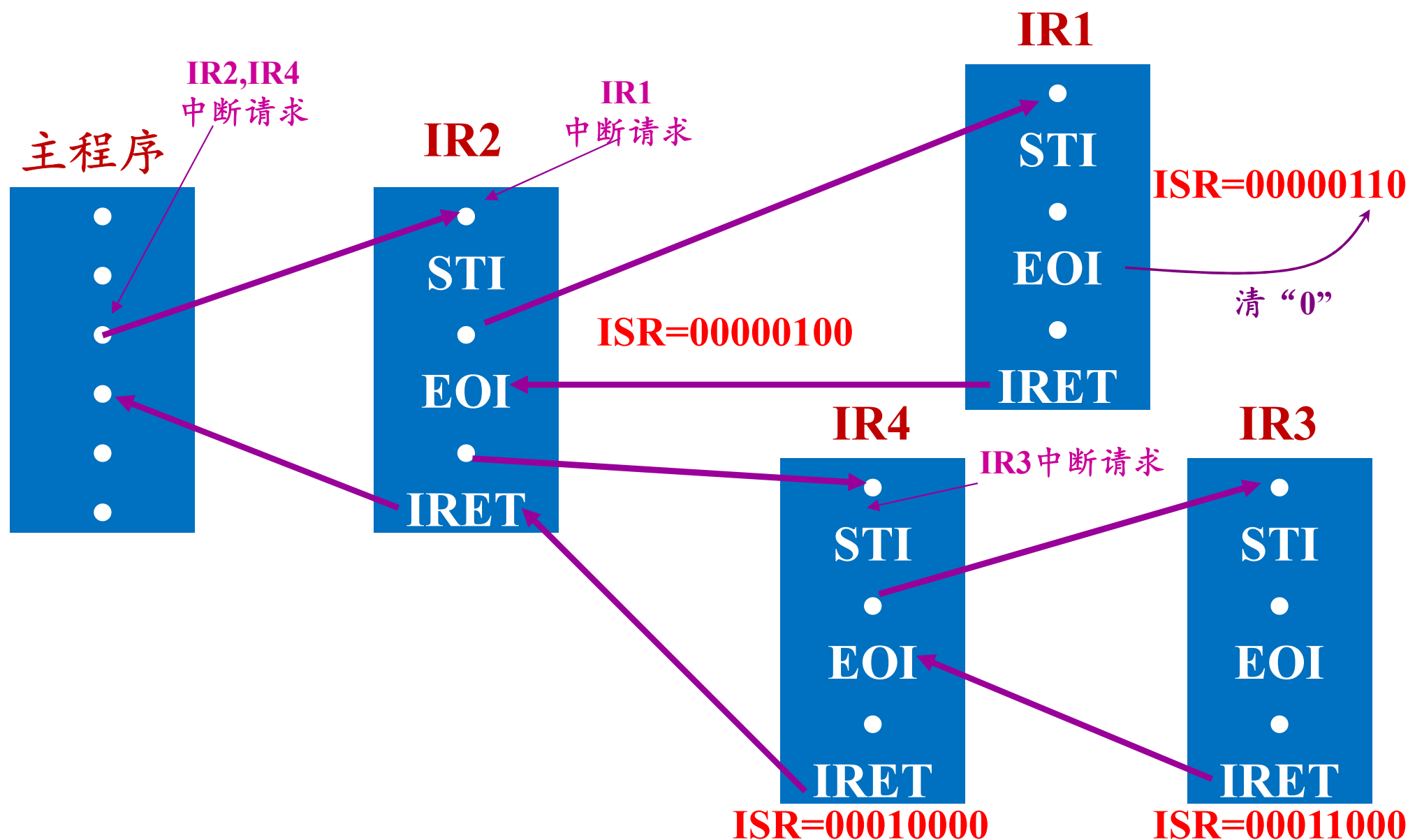


ISR的复位在自动结束/非自动结束时是不一样的



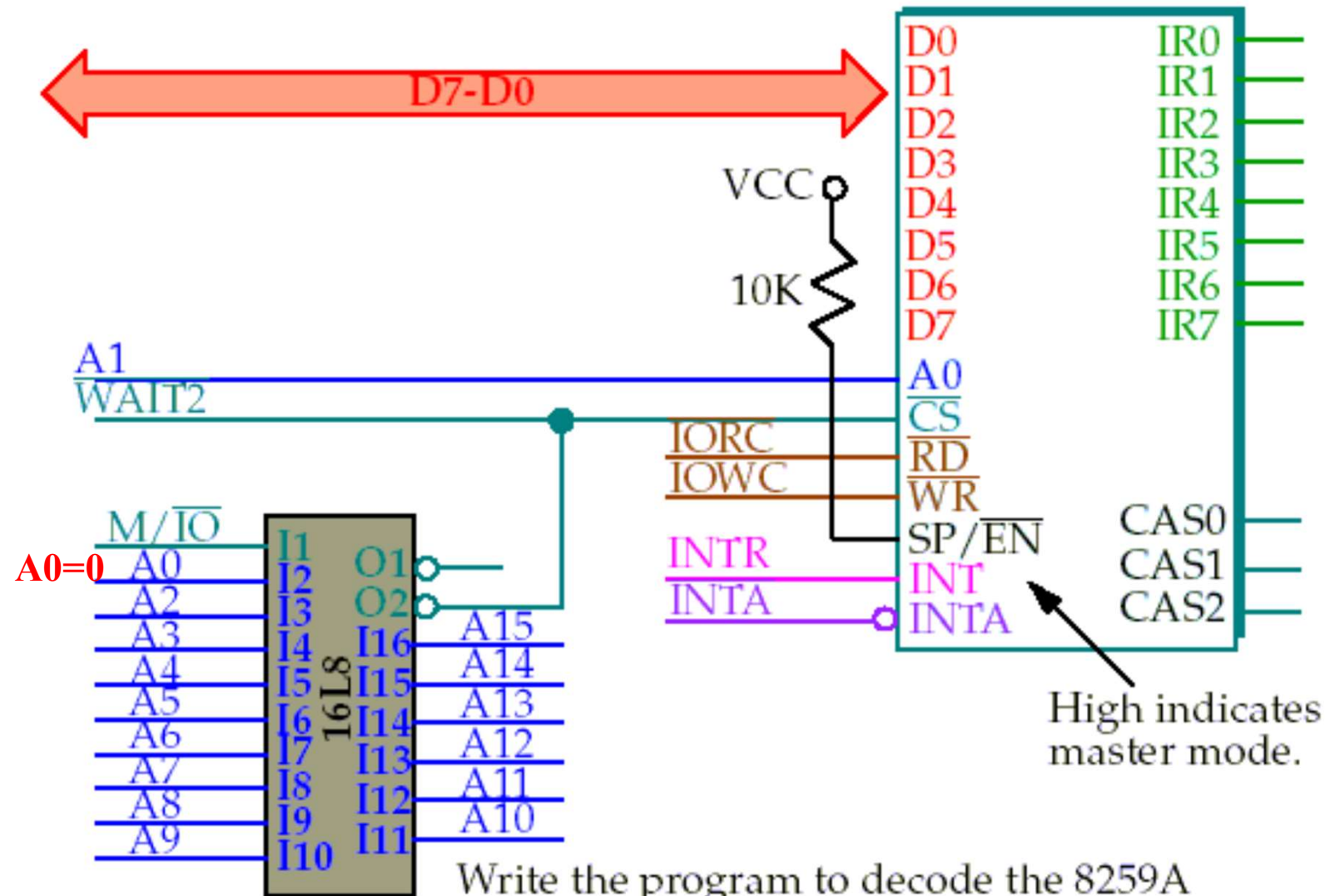


# ISR的作用





# 与8086处理器的连接



Write the program to decode the 8259A at ports 0400H and 0402H.





# 8259A的工作方式 (1)

- 中断触发方式
  - ◆ 边沿触发方式
  - ◆ 电平触发方式
- 屏蔽中断源方式
  - ◆ 普通屏蔽方式
  - ◆ 特殊屏蔽方式
- 中断嵌套方式
  - ◆ 完全嵌套方式
  - ◆ 特殊完全嵌套方式







# 8259A的工作方式（2）

- 优先级管理方式
  - ◆ 优先级固定方式
  - ◆ 优先级轮转方式
    - 自动轮转方式
    - 指定轮转方式
- 结束中断的处理方式
  - ◆ 自动中断结束方式
  - ◆ 手工中断结束方式
    - 不指定
    - 指定





# 8259A的工作方式 (3)

- 数据线连接方式
  - ◆ 缓冲方式
  - ◆ 非缓冲方式





# 8259A命令字

## ■ 初始化命令字

- ◆ 芯片控制（工作方式设置）**ICW1**
- ◆ 中断类型号**ICW2**
- ◆ 级联方式**ICW3**
- ◆ 特定完全嵌套、缓冲器方式**ICW4**

## ■ 操作命令字

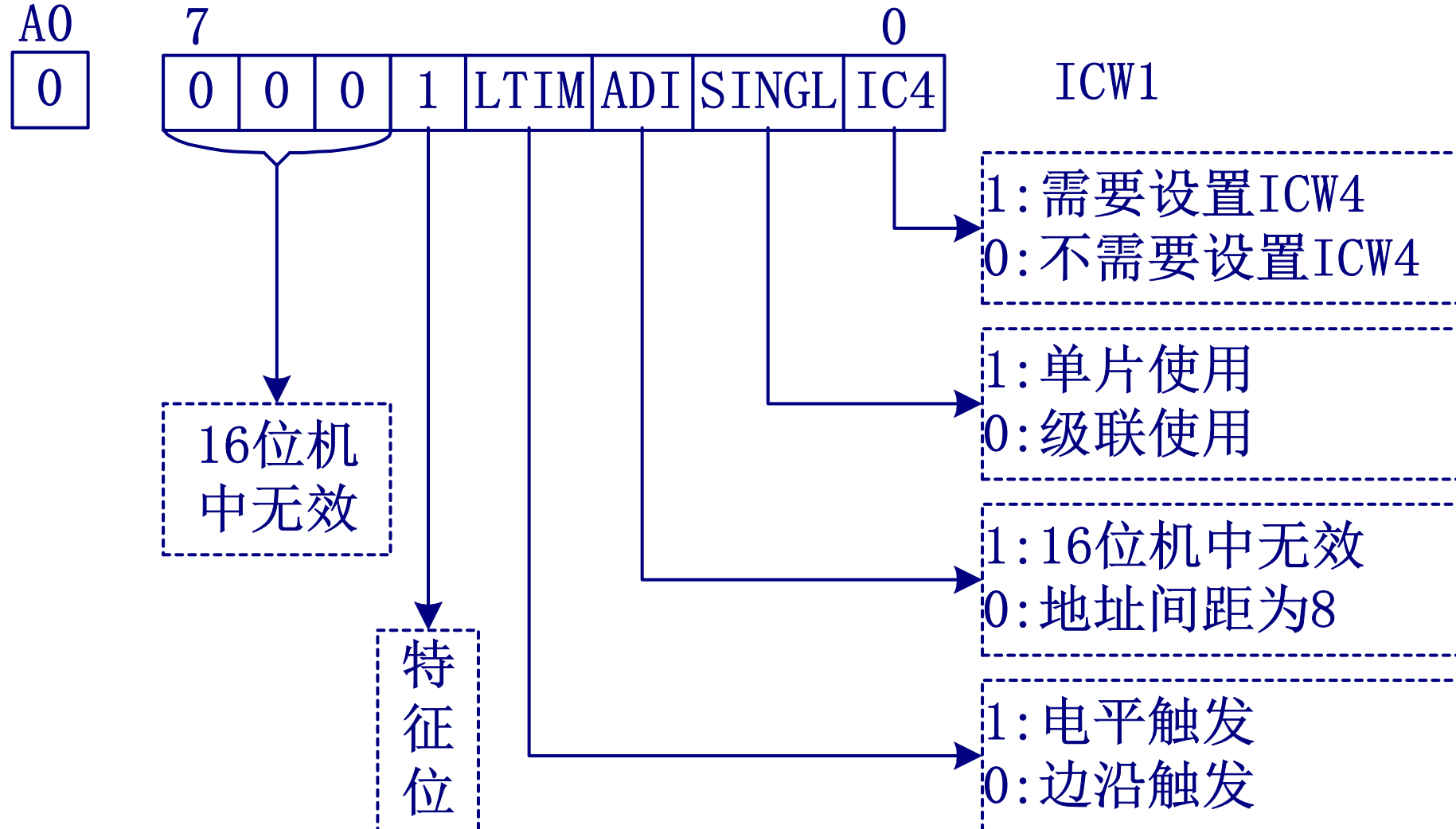
- ◆ 中断屏蔽字**OCW1**
- ◆ 中断结束方式**OCW2**
- ◆ 中断查询**OCW3**





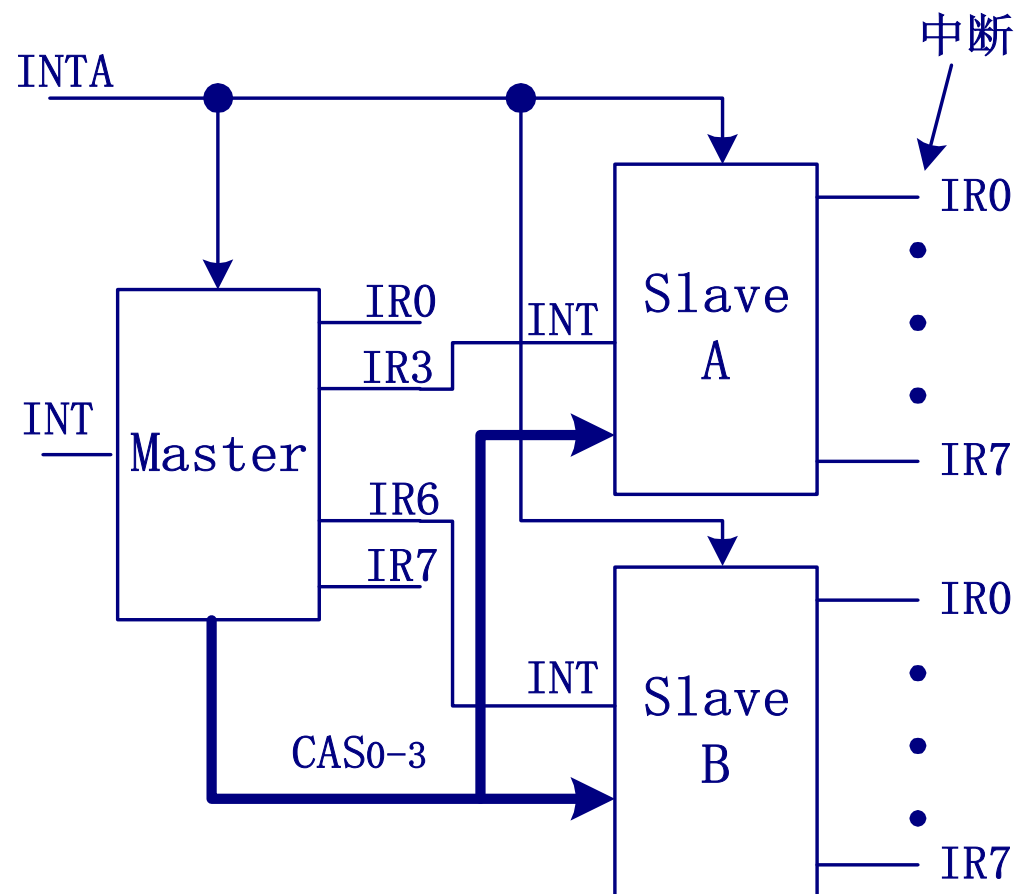
# 初始化命令字1 (ICW1)

- 完成触发方式设置及级联方式设置的功能



# 中断级联

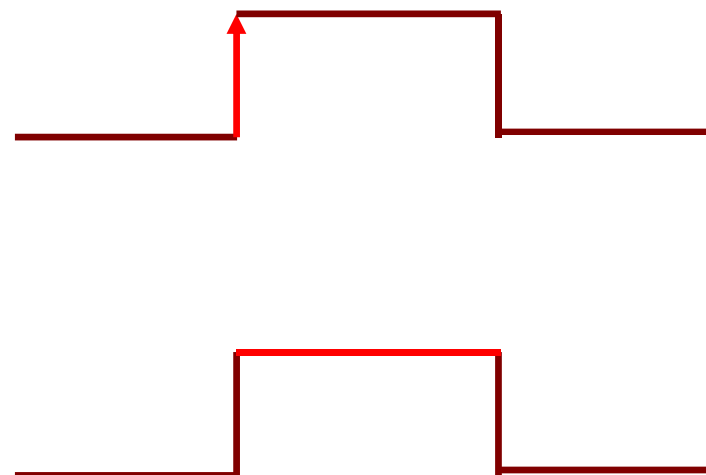
- 当系统中需要管理的中断源超过8个时，8259A可以采用级联的方式来管理。由一个主8259A和若干个从8259A（最多8个）构成。





# 中断触发方式

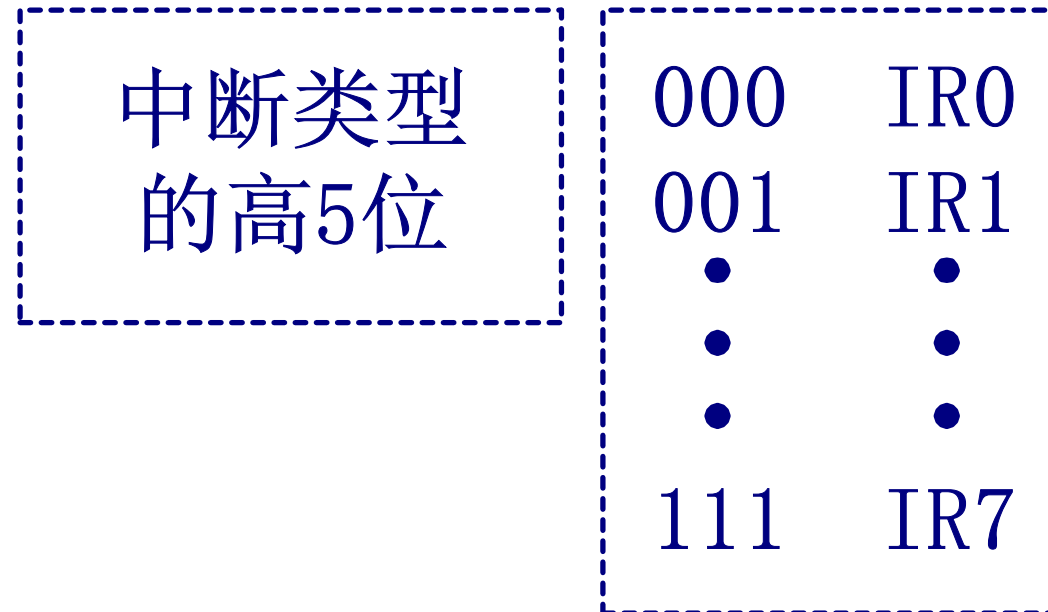
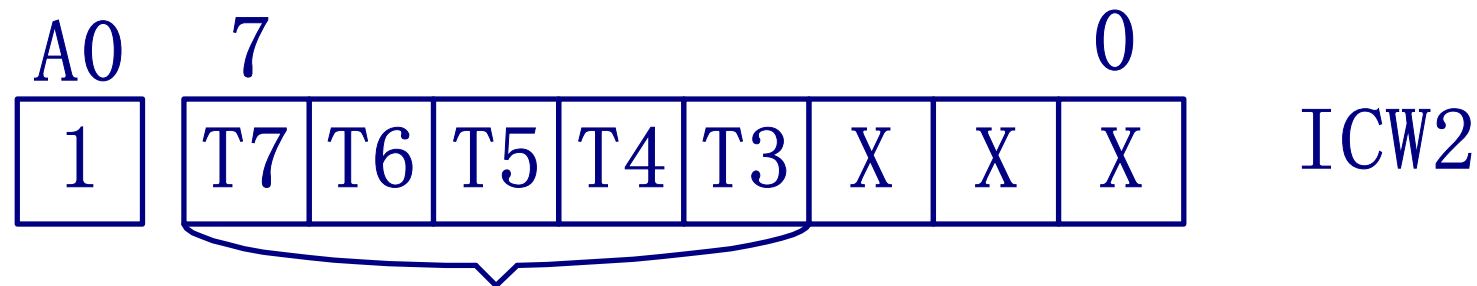
- 边沿触发方式
  - ◆ IRI端出现的上升沿视为有效的中断请求信号
- 电平触发方式
  - ◆ IRI端出现的高电平视为有效的中断请求信号





# 初始化命令字2 (ICW2)

- 完成中断类型号设置的功能





# 举例

- 在奇地址 ( $A_0=1$ ) 端口写入**00001000B**后, 对应的中断类型号为**08-0FH**
- 在奇地址 ( $A_0=1$ ) 端口写入**10000000B**后, 对应的中断类型号为**80-87H**







# 初始化命令字3 (ICW3) (1)

- 只在级联方式下使用。用来描述主、从片间的连接关系。
- 对主片的设置



$S_i=0$ :  $IR_i$ 上未接从片

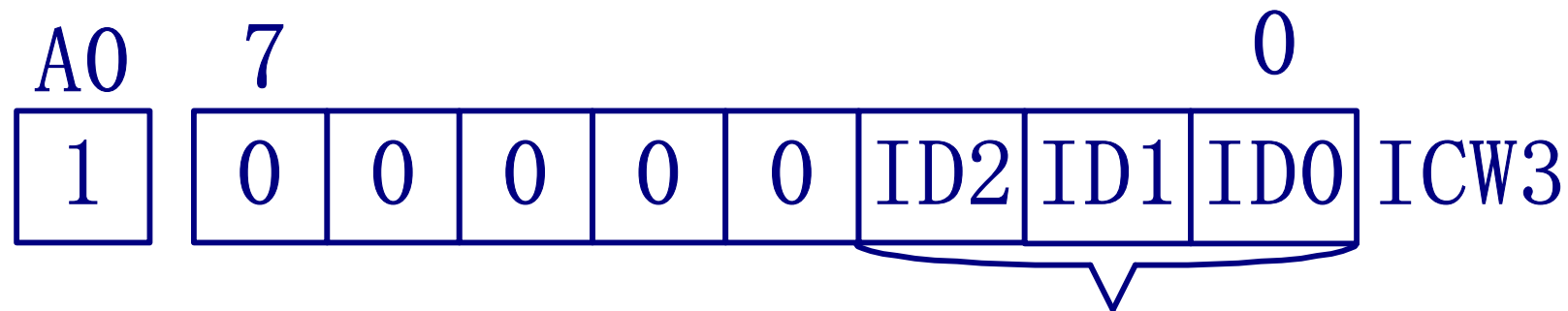
$S_i=1$ :  $IR_i$ 上接有从片





# 初始化命令字3 (ICW3) (2)

## ■ 对从片的设置

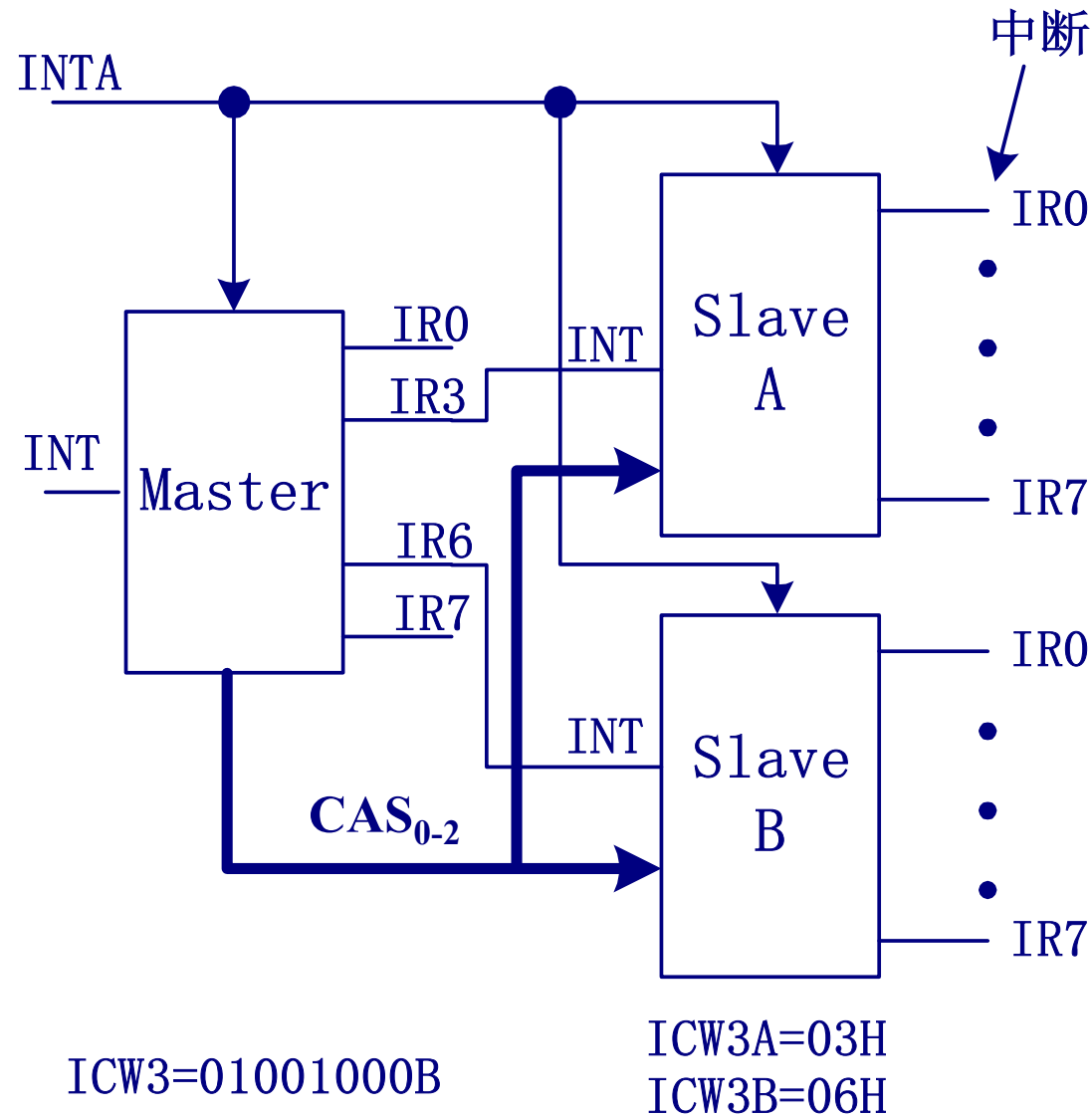


从片的识别地址



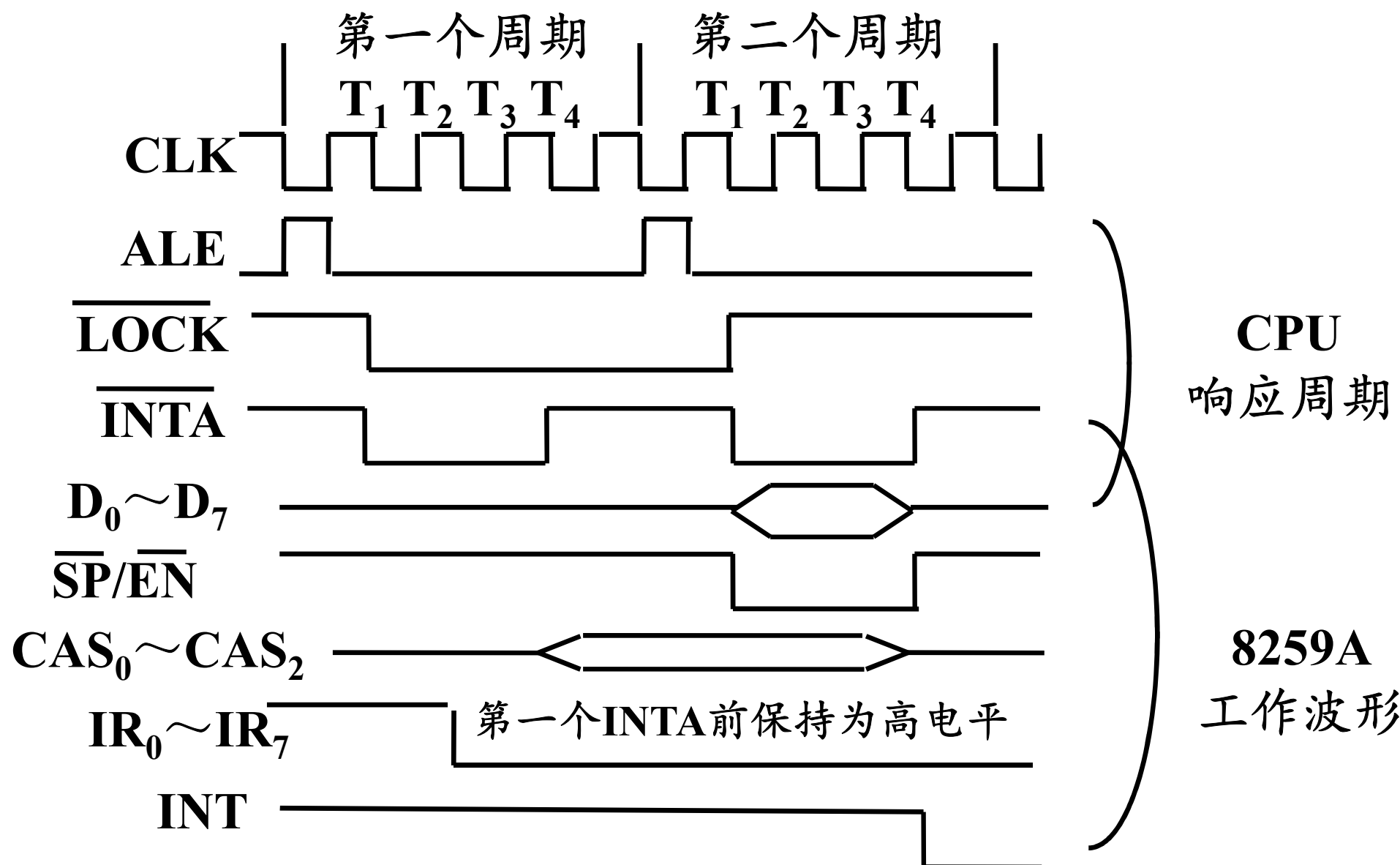


# 举例





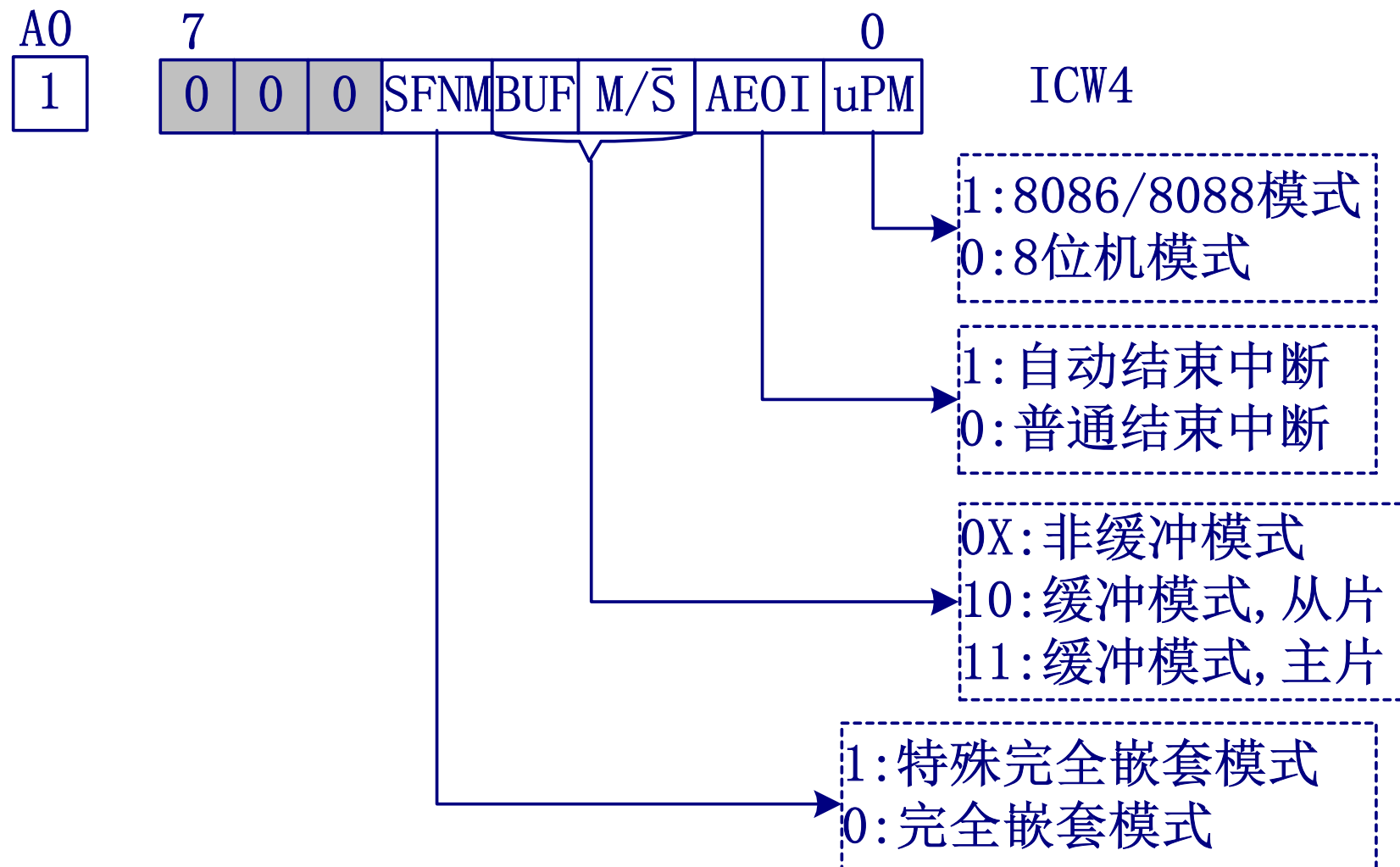
# 时序图





# 初始化命令字4 (ICW4)

- 完成结束中断方式、缓冲模式和嵌套模式的设置功能





# 结束中断（EOI）方式

## ■ 普通EOI

中断服务程序需向8259A送中断结束命令（EOI），将ISR的对应bit清0，以标识中断结束。

## ■ 自动EOI

中断程序无需送EOI命令。在第二个INTA脉冲信号的后沿，将ISR的对应bit清0。





# 缓冲模式

## ■ 非缓冲模式

- ◆  $\overline{SP/EN}=1$  将 8259A 置为主片方式,  $\overline{SP/EN}=0$  置为从片方式

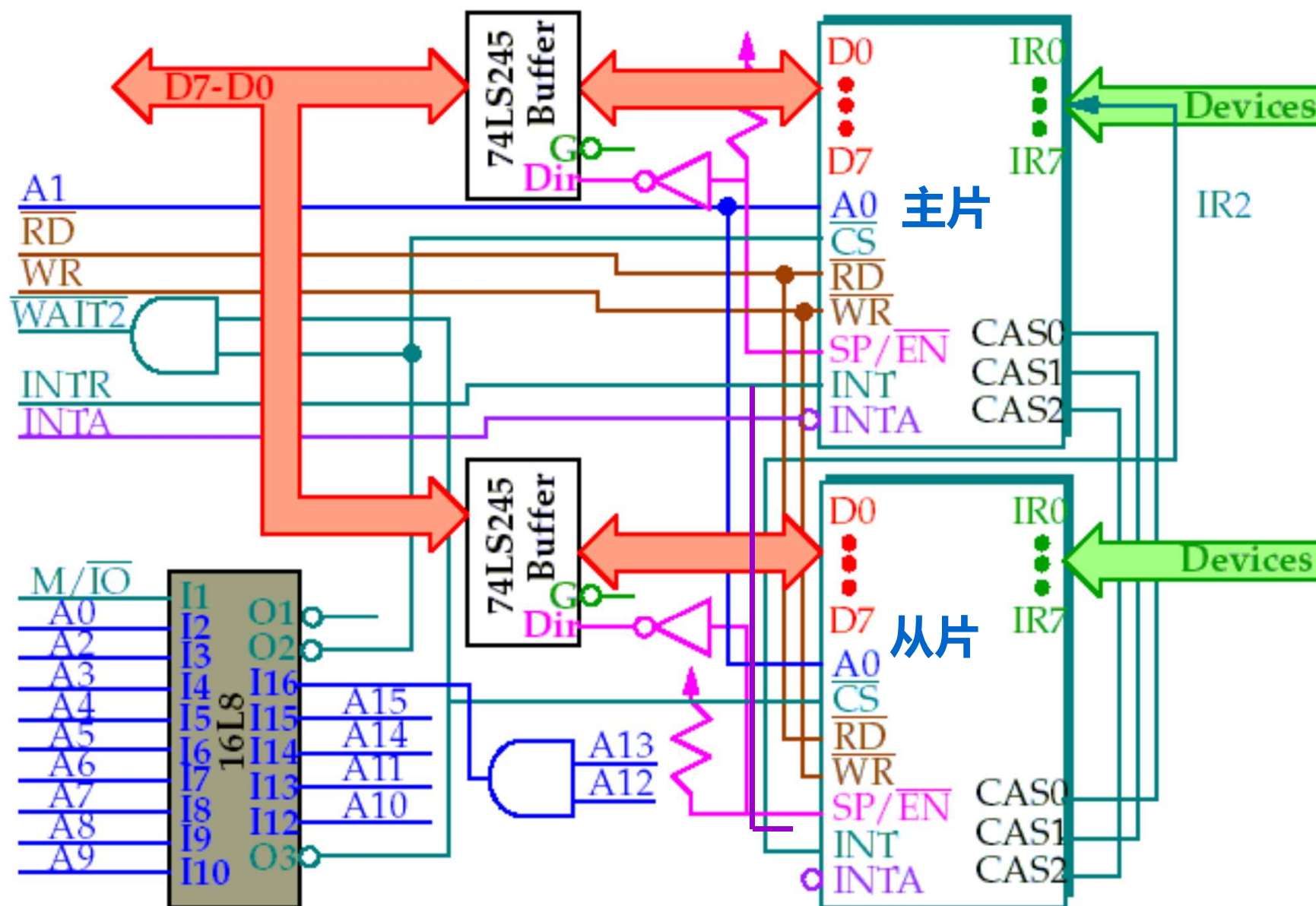
## ■ 缓冲模式

- ◆ 当系统中要求 8259A 必须通过总线缓冲器（总线收发器）挂在系统的数据总线上时，8259A 需设置工作在缓冲模式。此时， $\overline{SP/EN}$  作输出，用来控制总线缓冲器的传送方向。
- ◆ 由于  $\overline{SP/EN}$  已用作输出，只能用软件来设置 8259A 工作在主片还是从片。





# 电路连接图

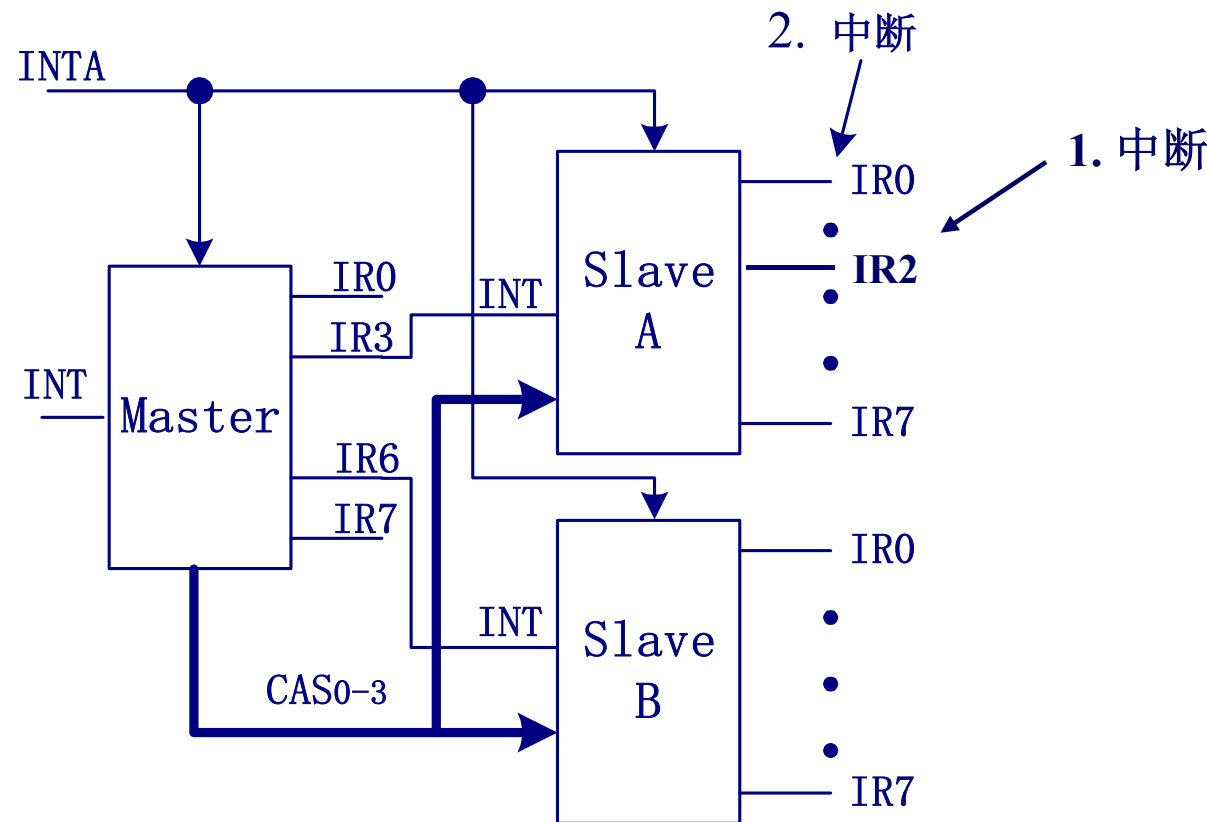






# 特殊完全嵌套方式

当前正在被执行的中断服务程序可被优先级相等或更高的中断请求中断



ICW3=01001000B

ICW3A=03H  
ICW3B=06H





# 操作命令字1 (OCW1)

- 完成中断屏蔽IR0-7的设置功能。OCW1可读可写



$M_i=1$ : 屏蔽由IR<sub>i</sub>引入的中断请求

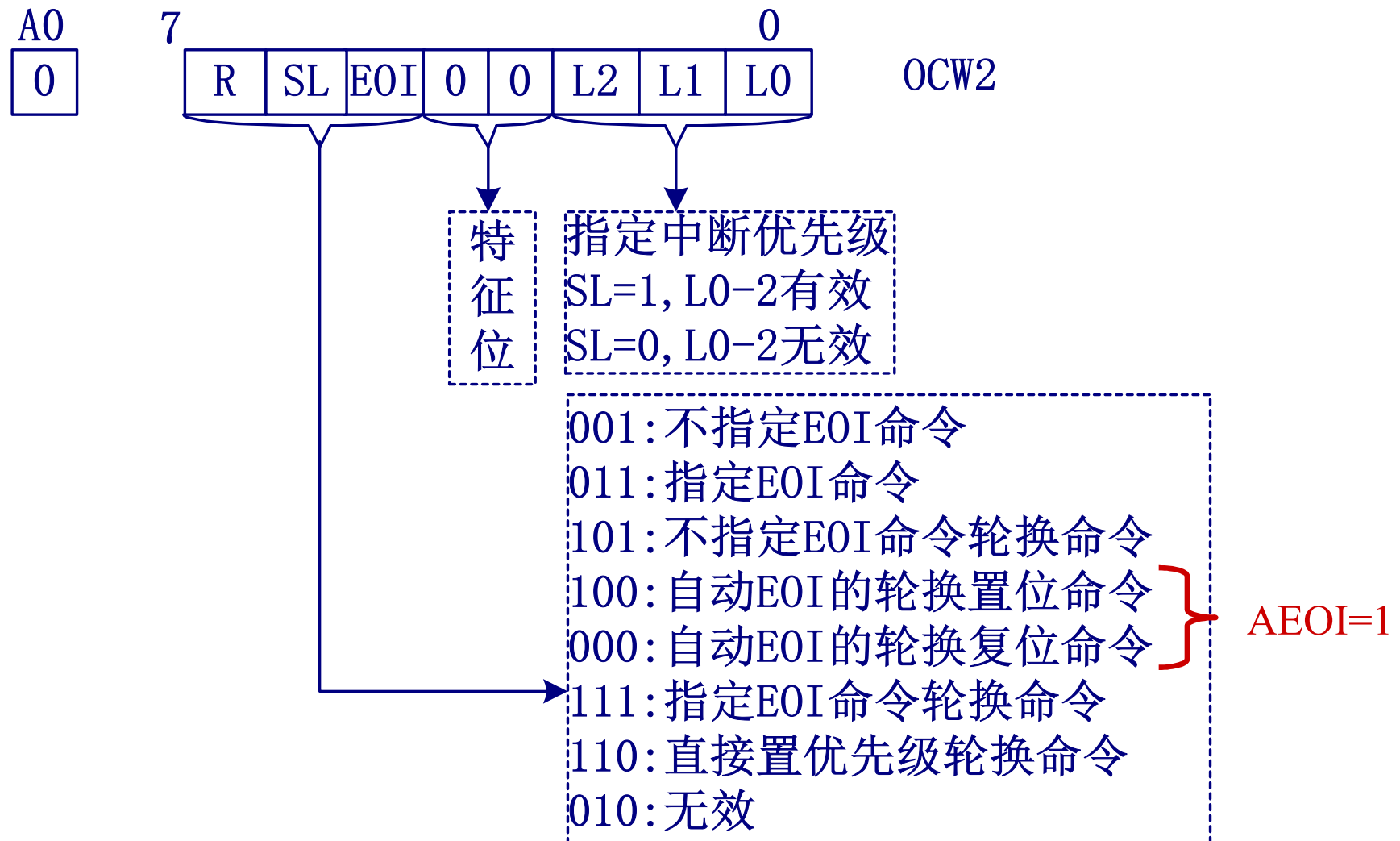
$M_i=0$ : 允许由IR<sub>i</sub>引入的中断请求





# 操作命令字2 (OCW2)

- 完成非自动中断结束方式、中断排队方式的设置功能





# 1. 不指定EOI

- 如果采用完全嵌套方式，可采用不指定EOI方式。  
在发出EOI后，中断控制器将ISR中优先级最高的bit复位。

例：若ISR=00100100，执行

`mov al, 20h`

`out 20h, al` 后，

ISR=00100000





## 2. 指定EOI

- 如果中断优先级被打乱（即：当前正在被服务的中断服务程序的优先级不是最高的），则必须采用指定EOI方式。
- 在发出EOI时，必须指定要复位的ISR的bit的位置。

例：ISR=00100100，当前服务程序对应ISR的bit5

mov al, 01100101b

out 20h, al 后，

ISR=00000100

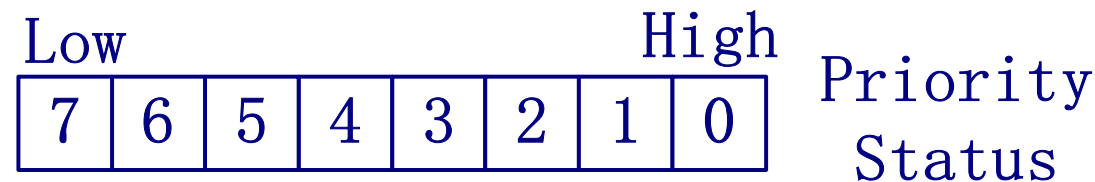
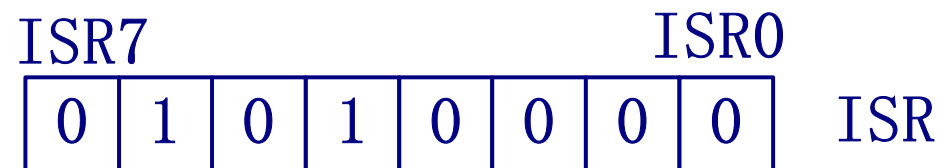




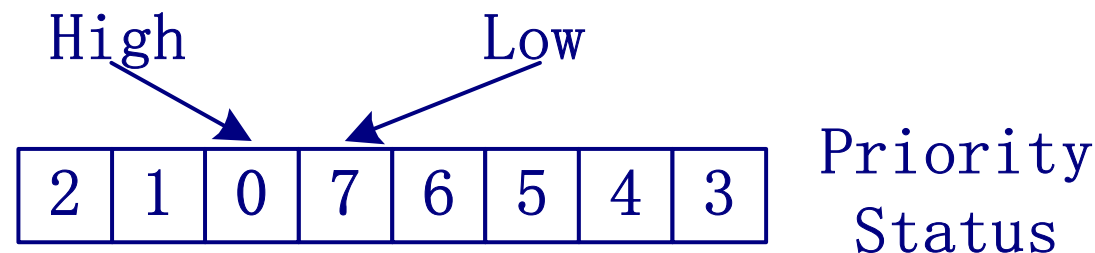
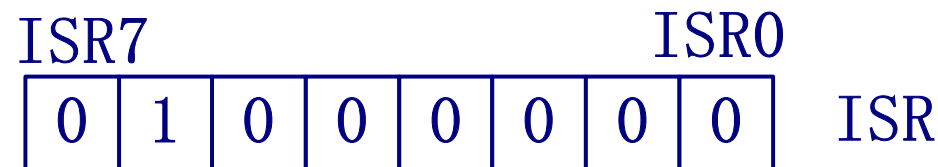
### 3. 自动轮换（相同优先级）

- 某中断源被服务后，优先级自动降为最低级。

- 例：轮换前



轮换后





## 4. 指定轮换

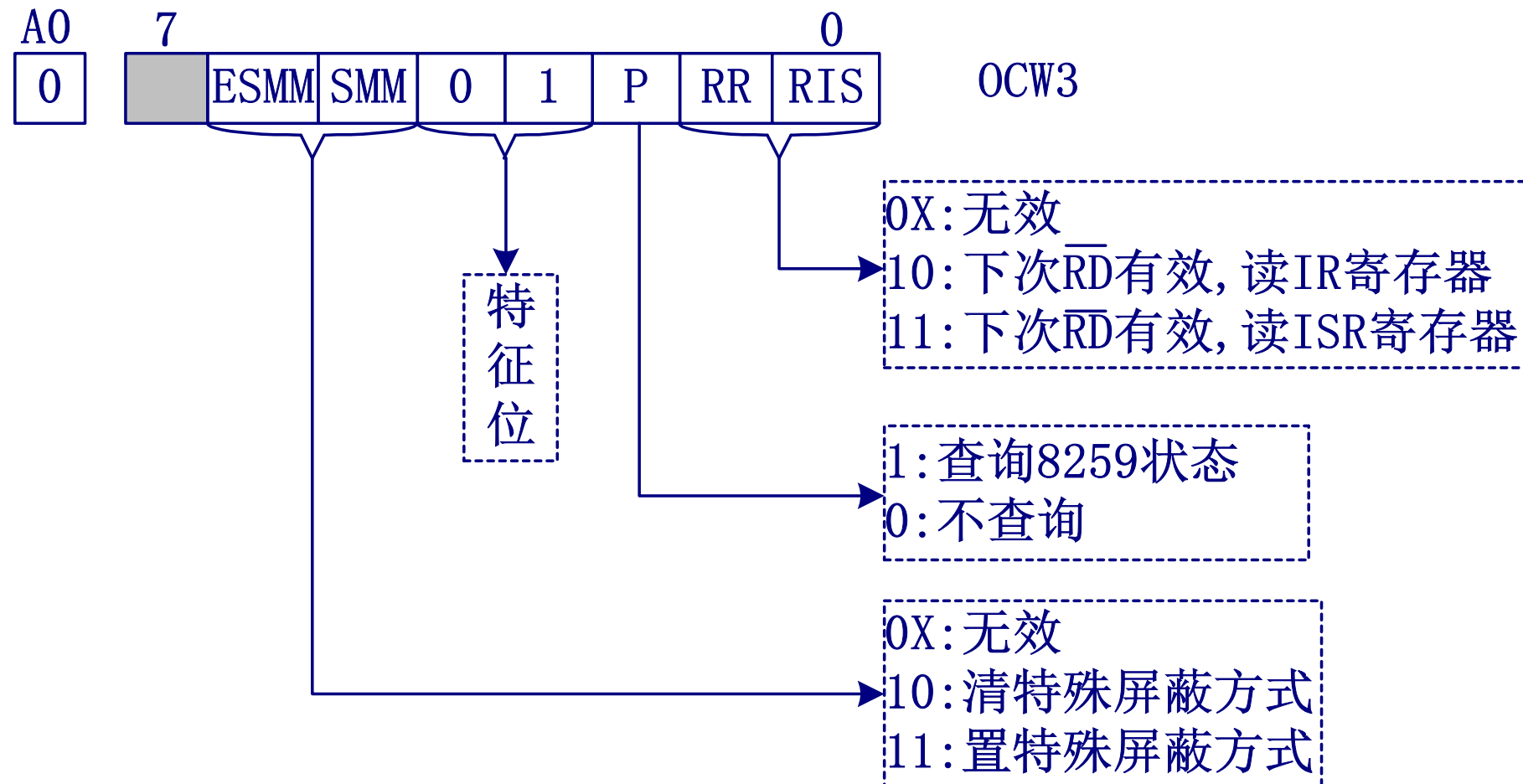
- 利用命令将指定中断源的优先级置为最低  
**R=1, SL=1, L0-L2**指定需轮换的中断源号





# 操作命令字3 (OCW3)

- 完成IR和ISR寄存器、状态字查询，特殊屏蔽方式设置的功能







# 读IR或IS寄存器

## ■ 读IRR

```
mov al, 00001010b
```

```
out 20h, al
```

```
nop ; 延时
```

```
in al, 20h ; IRR⇒al
```

## ■ 读ISR

```
mov al, 00001011b
```

```
out 20h, al
```

```
nop ; 延时
```

```
in al, 20h ; ISR⇒al
```





# 读IMR寄存器

## ■ 读IMR

in al, 21h

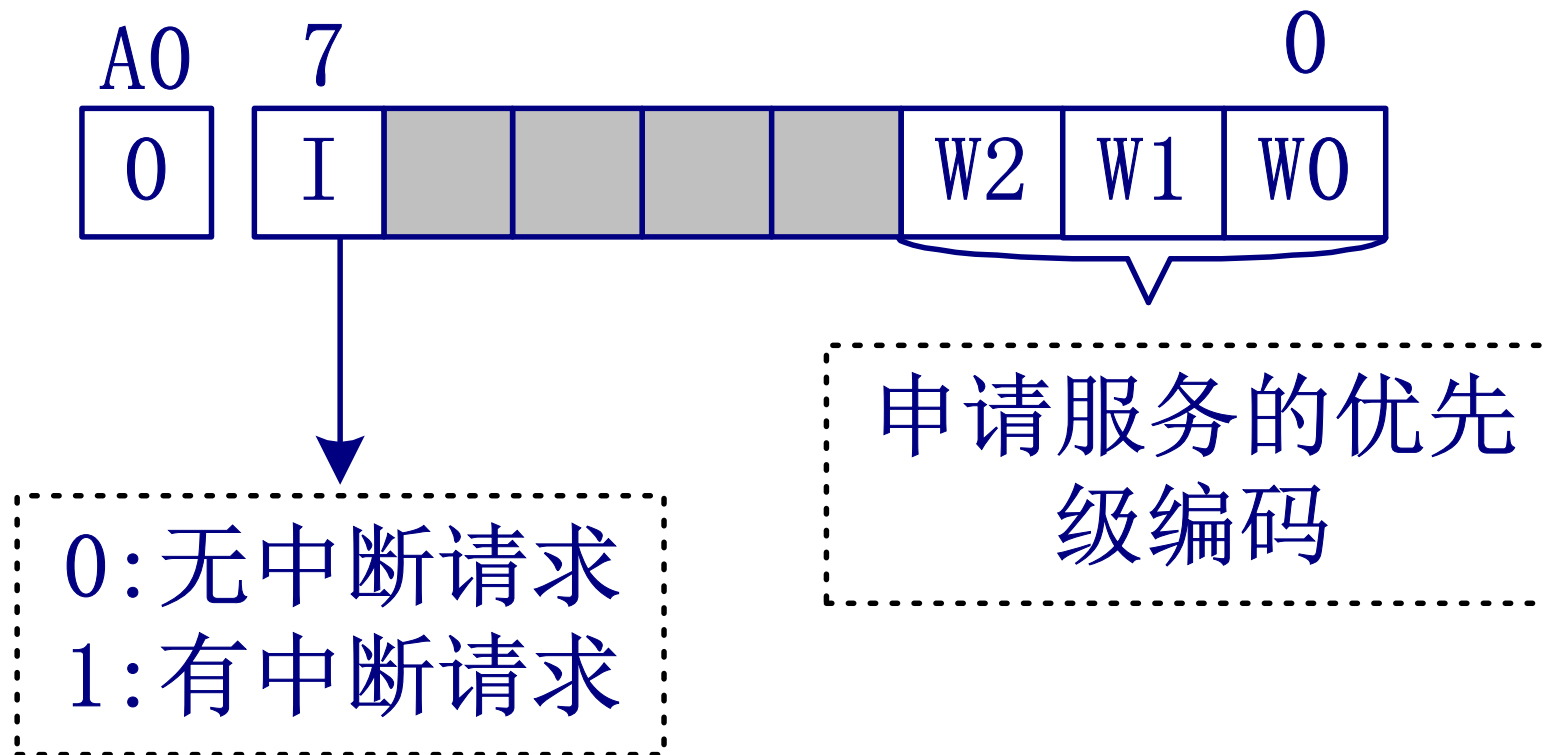
; IMR $\Rightarrow$ al





# 查询（状态）字

- 用于表示8259A状态





# 读8259A查询字

例:

**mov al, 00001100b**

; 查询8259状态

**out 20h, al**

**nop**

; 延时

**in al, 20h**

; 查询字⇒al





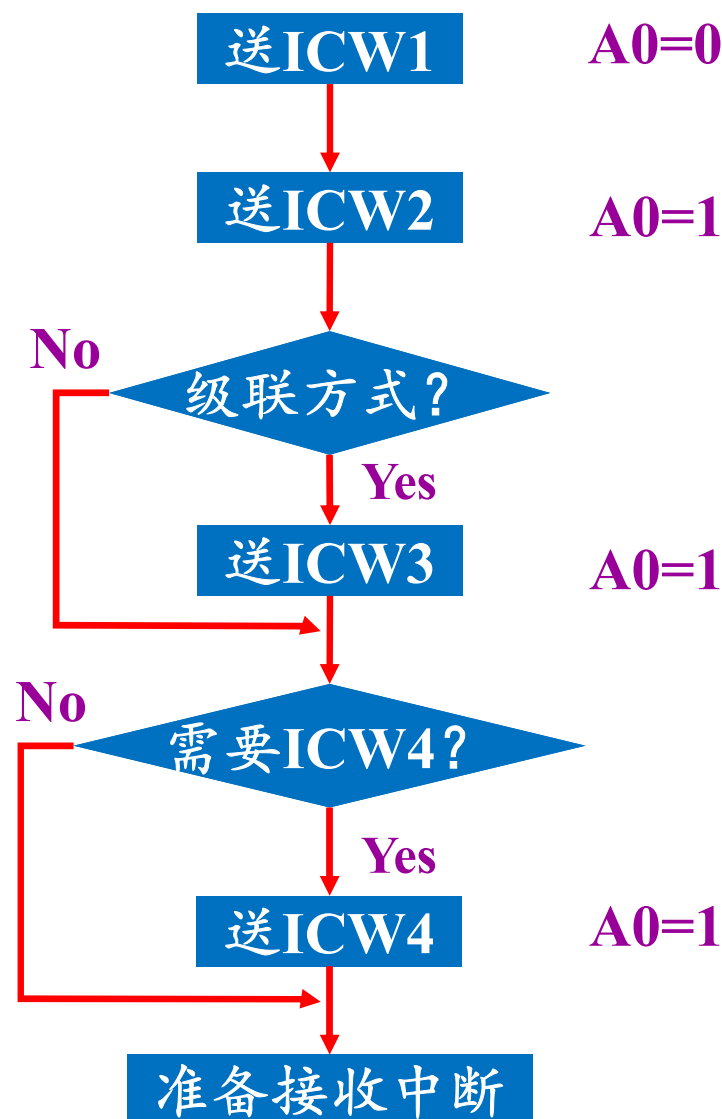
## 7.4 8259A应用举例





# 8259A初始化流程

$A_0=0$	$A_0=1$	$D_4D_3$
ICW <sub>1</sub>		1×
	ICW <sub>2</sub>	×
	ICW <sub>3</sub>	×
	ICW <sub>4</sub>	×
初始化完成		
	OCW <sub>1</sub>	×
OCW <sub>2</sub>		00
OCW <sub>3</sub>		01





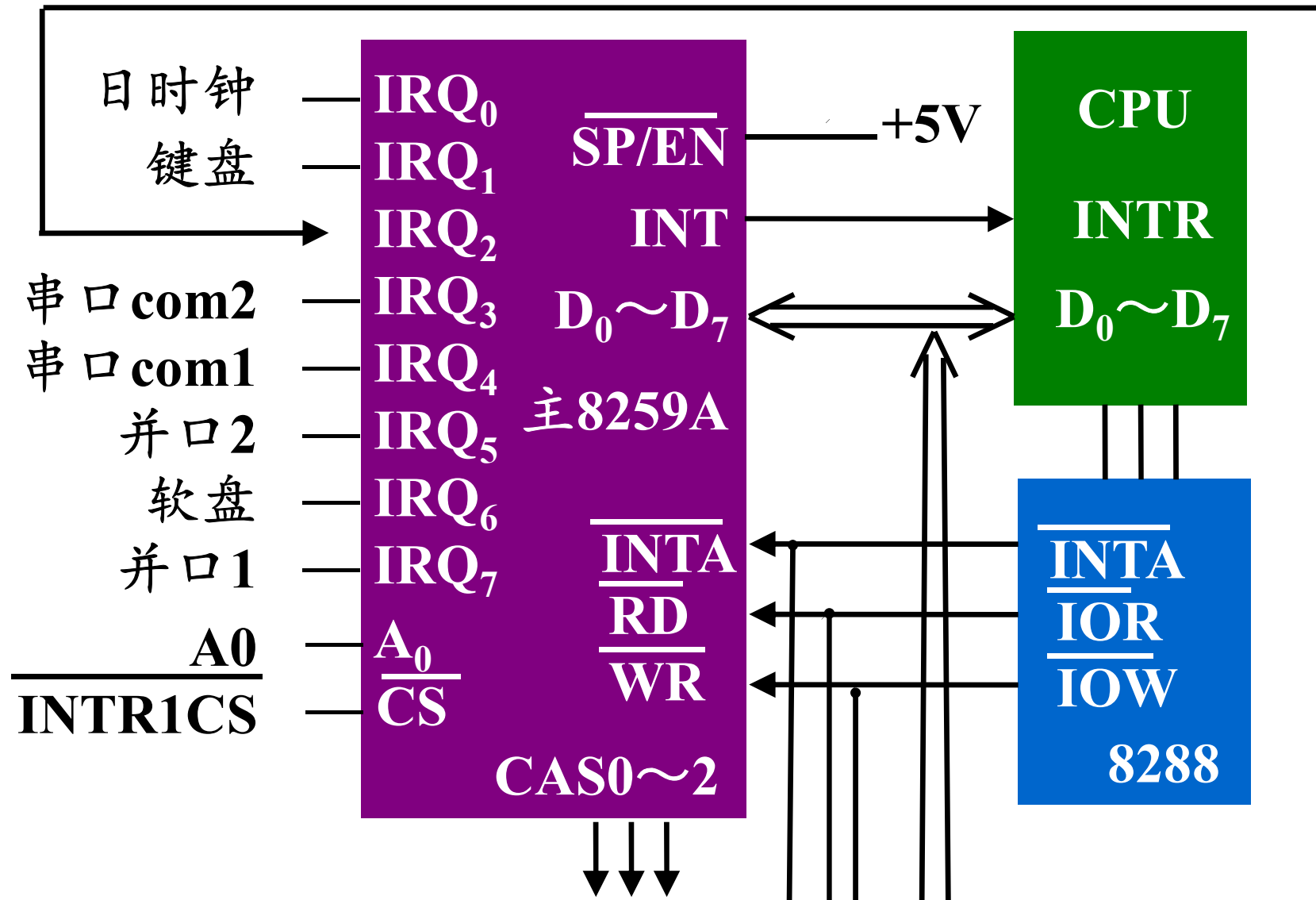
# 8259A初始化编程举例

- 两片8259主从级联，从片INT直接连到主片的IR2上。均采用非缓冲器方式，中断请求信号均采用边沿触发。
- 主片：端口地址20H和21H，中断类型号为08H~0FH。
- 从片：端口地址A0H和A1H，中断类型号为70H~77H。





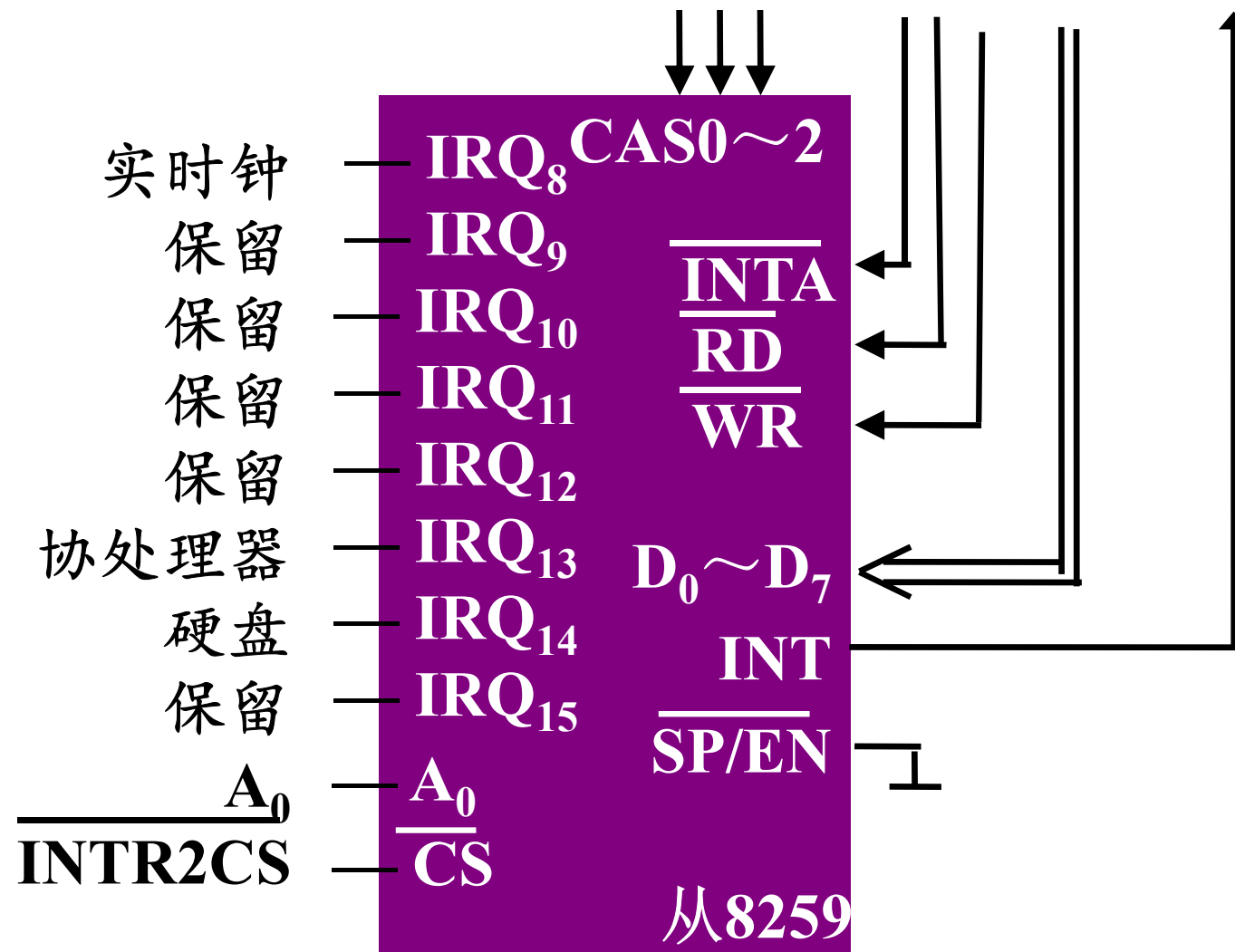
# 双片8259A中断系统的硬件连接







# 双片8259A中断系统的硬件连接 (2)





# 初始化程序

```
mport0 equ 20h
mport1 equ 21h
```

```
...
mov al,11h;边沿触发,多片,icw4
out mport0,al
nop
mov al,8;中断向量
out mport1,al
nop
mov al,04;IR2上接从片
out mport1,al
nop
mov al,11h;非缓冲,特殊嵌套,非AEOI
out mport1,al
...
```

主片

```
sport0 equ a0h
sport1 equ a1h
```

```
...
mov al,11h
out sport0,al
nop
mov al,70h
out sport1,al
nop
mov al,02
out sport1,al
nop
mov al,01h
out sport1,al
...
```

从片





## 7.5 硬件中断服务程序的编写





# 硬件中断服务程序的编写

## ■ 主程序

- ◆ 设置中断向量表
- ◆ 设置8259A的中断屏蔽寄存器
- ◆ 使能CPU中断允许标志，即：IF←1

## ■ 中断服务程序

- ◆ 中断服务程序要尽量短
- ◆ 参数传递、临时变量要使用存储单元
- ◆ 向中断控制器8259A发送中断结束命令EOI





# 举例1

- 定时器8253每55ms通过8259A的IR0脚向CPU发出中断请求，编写中断服务程序（中断向量号为08H），每次中断显示一信息，共显10次。





# 程序片段 (1)

```

msg      db 'Interrupt'
count    db 30h
          db 0dh,0ah,0

          . . .
          mov ax,3508h
          int 21h
          push es
          push bx; 保存旧中断入口
          cli
          push ds
          mov dx,offset intr
          mov ax,seg intr
          mov ds,ax
          mov ax,2508h
          int 21h
          pop ds
          in al,21h; 读旧中断屏蔽字
          push ax; 保存
          and al,0feh; 允许IR0
          out 21h,al
          sti
again:    cmp count,3ah; 等待中断
          jb again
  
```

```

          cli
          pop ax
          out 21h,al
          pop dx
          pop ds
          mov ax,2508h
          int 21h
          sti
          mov ax,4c00h
          int 21h
  
```





# 程序片段 (2)

```

intr  proc
      sti
      push ax
      push ds
      push si
      mov ax,@data
      mov ds,ax
      inc count
      mov si,offset msg
      call disp
      mov al,20h
      out 20h,al; EOI命令
      pop si
      pop ds
      pop ax
      iret
intr  endp

```

```

dis  proc
      push ax
      push bx
lp1: lodsb                ; al←(si)
      cmp al,0
      jz lp2
      mov bx,0
      mov ah,0eh
      int 10h; 显示, BIOS调用
      jmp lp1
lp2:  pop bx
      pop ax
      ret
Disp endp

```





# 举例2

对例1的要求做修改

- 定时器8253每55ms通过8259A的IR0脚向CPU发出中断请求，试编写一秒表显示程序（即：00、01、02、... 59、00、...）。







# 程序片段 (1)

```
count db 0
last db 1
secd db 0
```

```
again: sti
        mov al, secd
        cmp al, 60
        jc lp1
        mov secd, 0 ; 满60清0
lp1:    mov al, secd
        cmp al, last ; 显示值是否已更新
        jz lp1
        mov last, al
        mov bl, 10
        mov ah, 0
        div bl
        mov bx, ax
        add bx, 3030h ; 变为ASC码
```

```
mov ah, 02
mov dl, bl
int 21h ; 显示十位
mov ah, 02
mov dl, bh
int 21h ; 显示个位
mov dl, 0dh
mov ah, 02 ; 回车
int 21h
jmp again
```

指令格式

**DIV SRC**

执行操作:

$(AL) \leftarrow (AX) / (SRC)$  的商

$(AH) \leftarrow (AX) / (SRC)$  的余数





# 程序片段 (2)

```
intr  proc
      sti
      push ax
      push ds
      mov ax, @data
      mov ds, ax
      inc count
      cmp count, 18
      jnz 1p2
      mov count, 0
      inc secd
1p2:  mov al, 20h
      out 20h, al; EOI命令
      pop ds
      pop ax
      iret
intr  endp
```





## 7.6 定时/计数技术





# 7.6.1 基本概念





# 定时与计数

## ■ 定时

提供一个时间基准。

定时的信号具有周期性。

**本质上，定时是通过计数来实现的。**

## ■ 计数

对特定事件（即：脉冲）进行计数。

计数的信号可能具有随机性。

## ■ 用途

- ◆ 系统时钟
- ◆ **DRAM**刷新定时
- ◆ 定时采样
- ◆ 实时控制
- ◆ 脉冲的计数





# 计算机系统中的定时

- 在一个计算机系统中，定时功能是不可却少的
- 用硬件定时器驱动操作系统的系统时钟，其定时长度（称为**Tick**）为**10ms**、**100ms**或**1s**等
- 操作系统根据系统时钟完成任务调度等功能





# 定时方法

## ■ 软件定时

由于指令执行周期的长短是可知的，因此，用一段循环程序可实现定时。但占用**CPU**时间，而且主机频率不同，同一软件执行时其延时就可能不同，定时程序通用性差

## ■ 硬件定时

采用硬件或可编程定时器芯片实现。定时准确，不占用**CPU**时间。但需附加硬件

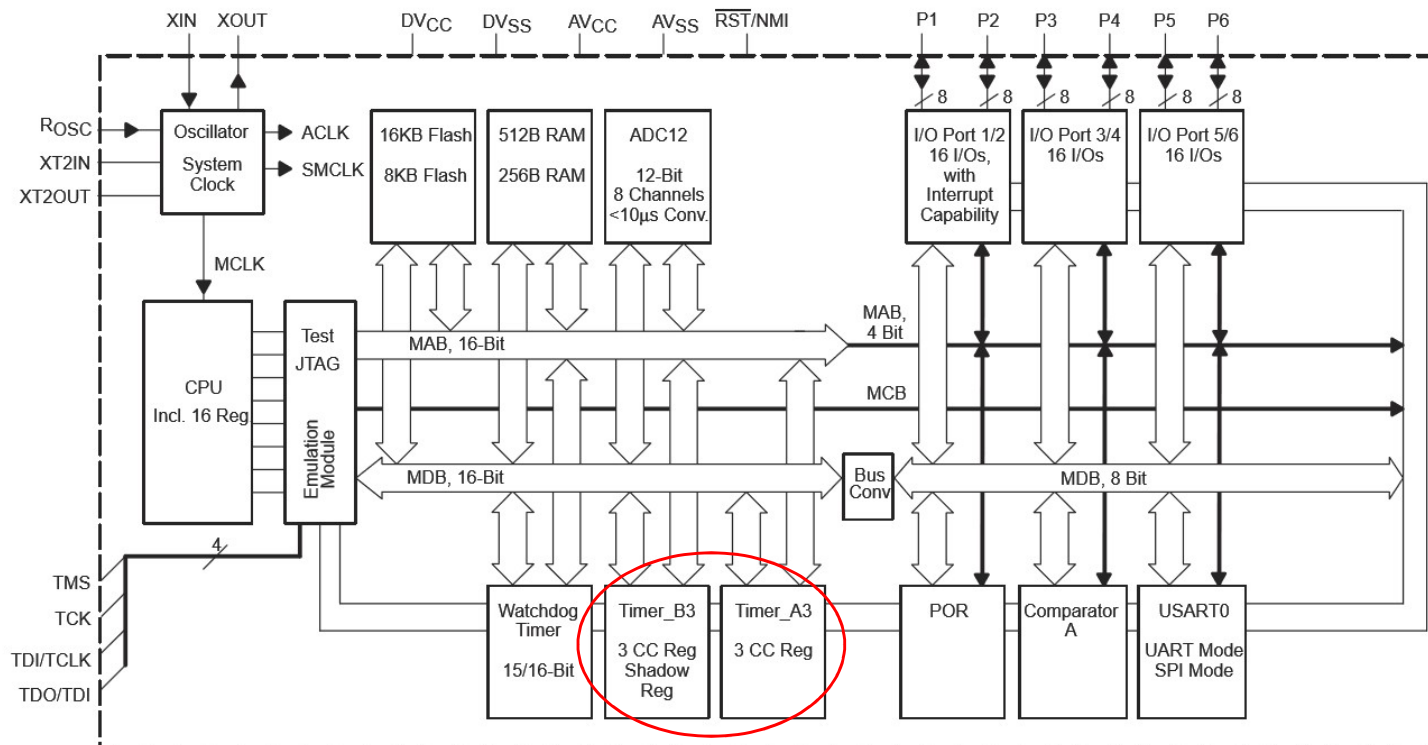




# 典型芯片

- Intel8253/8254  
8254是8253的改进型。
- Zilog CTC
- 各种单片机、SoC等

MSP430x13x







# 7.6.2 可编程定时/计数器 8253/8254





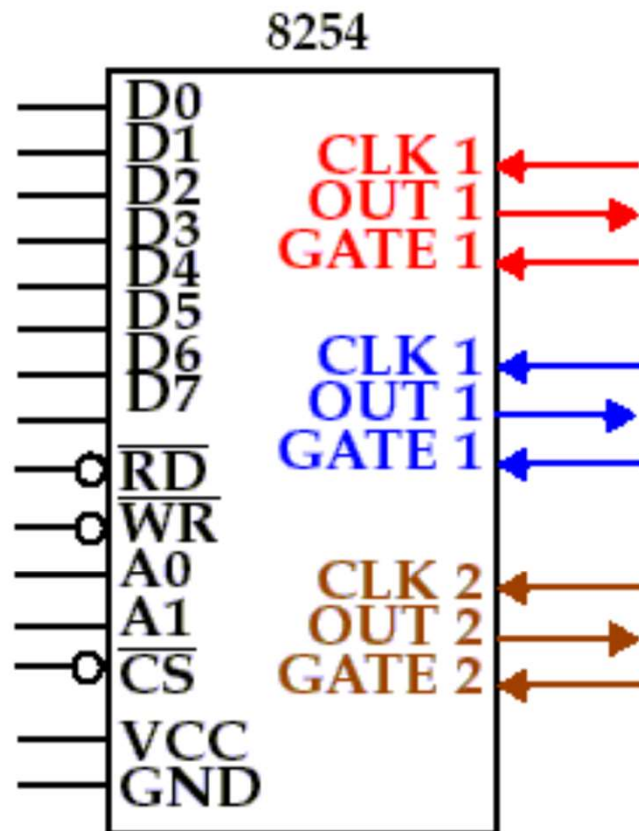
# 基本功能

- 具有3个独立的**16**位定时/计数器（**T/C**）
- 每个**T/C**功能：
  - ◆ 可按二、十进制计数
  - ◆ 有**6**种不同的工作方式
  - ◆ 最高频率**10MHz**
  - ◆ 有读回状态功能。（**8253**没有）

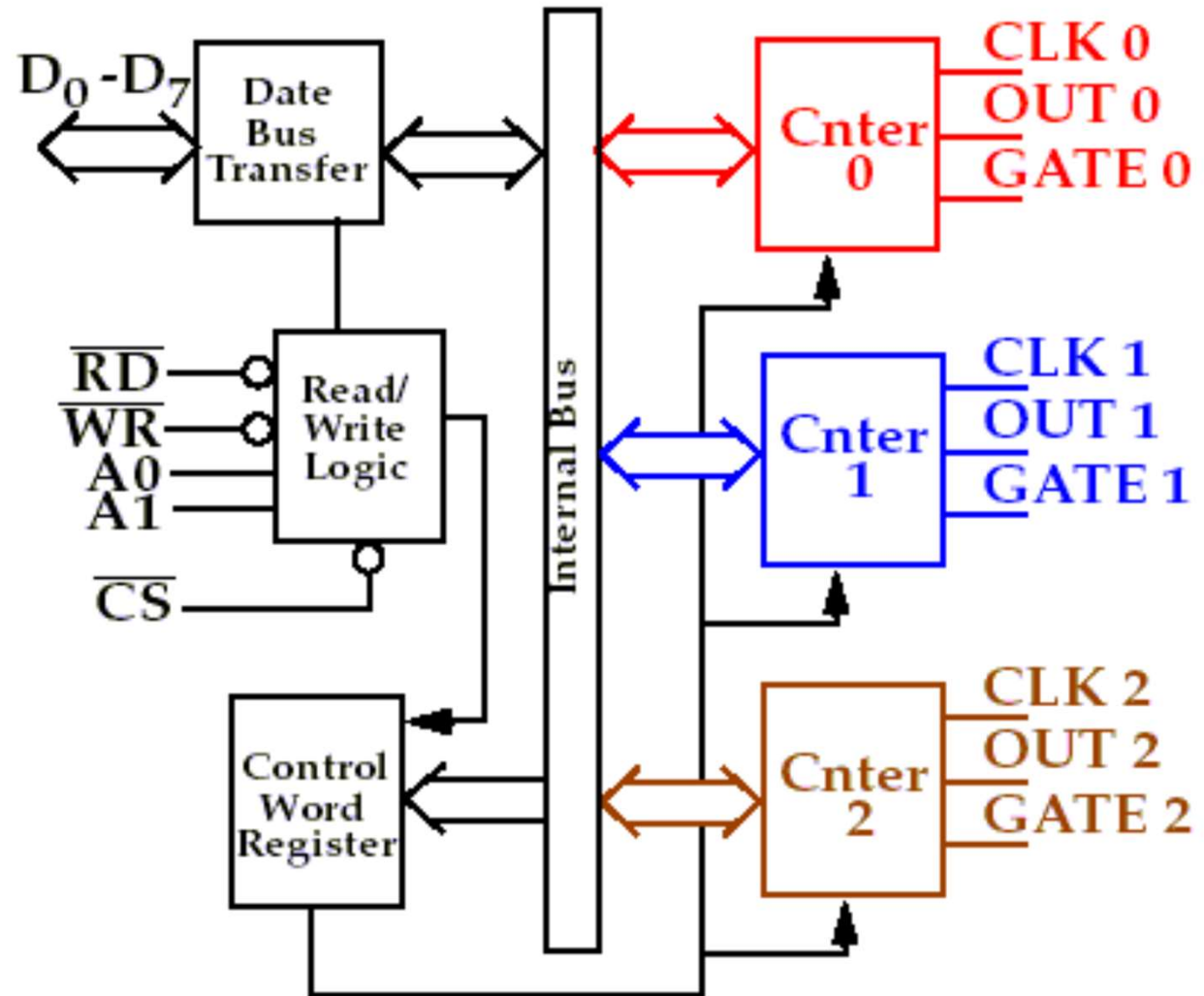




# 8253/8254外部特性及内部结构



Internal structure





# 管脚定义

- 面向CPU的信号线
  - ◆ **D0~D7 (In/Out)** 三态双向数据线
  - ◆  **$\overline{RD}$  (In)** 读信号线
  - ◆  **$\overline{WR}$  (In)** 写信号线
  - ◆  **$\overline{CS}$  (In)** 片选
  - ◆ **A0, A1 (In)** 地址线。占有4个I/O端口
- 面向I/O的信号线（包括3组同样的信号）
  - ◆ **CLK (In)** 计数时钟输入
  - ◆ **GATE (In)** 门控信号。“0”电平禁止计数器工作。
  - ◆ **OUT (Out)** 计数器输出。表示定时或计数已到。





# 功能描述 (1)

## ■ 数据总线缓冲器

三态、双向8位寄存器

## ■ 读/写控制电路

A1、A0 占用4个端口。

## ■ 控制命令寄存器

存放CPU送来的控制字

A1	A0	
0	0	计数器0
0	1	计数器1
1	0	计数器2
1	1	控制寄存器





# 功能描述 (2)

## ■ 计数器

芯片内部有**3**个独立、完全相同的计数器。每一个计数器有**6**种不同的工作模式，包含有**3**个**16**位寄存器

### ◆ 16位计数初值寄存器**CR**

用于存放计数初值。然后将初值装入**CE**。可自动重载初值

### ◆ 16位计数单元**CE**

减“1”计数器。**CPU**不能直接读、写**CE**的值。

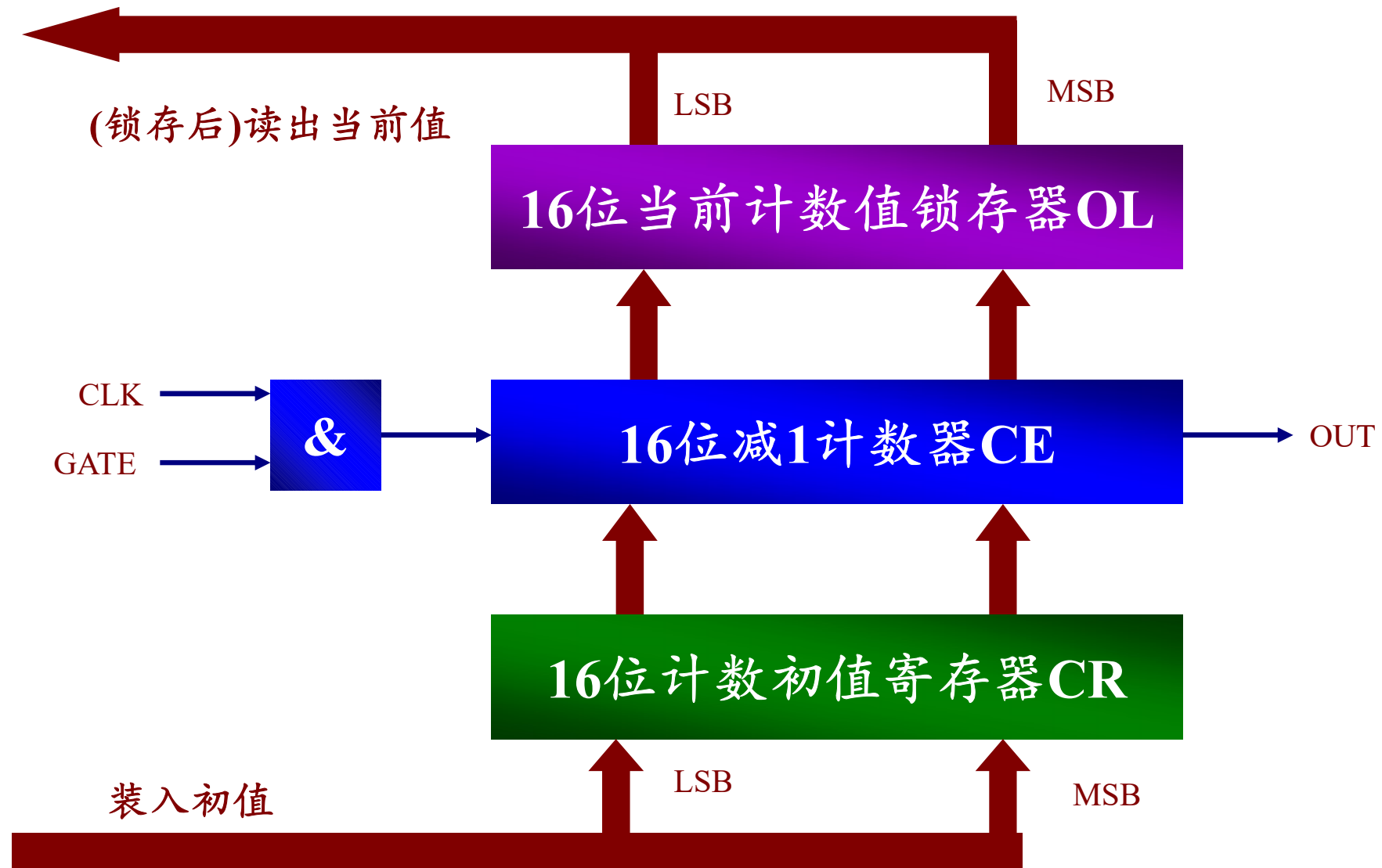
### ◆ 16位当前计数值锁存器**OL**

**OL**“跟随” **CE**值变化。收到“锁存”命令后，锁存当前**CE**的值。**CPU**读走后，恢复到“跟随” **CE**值的状态。





# 计数通道内部逻辑框图





# 8253/8254读写操作类型

/CS	/RD	/WR	A1	A0	操作	特征位D7D6
0	1	0	0	0	计数初值写入0#计数器	**
0	1	0	0	1	计数初值写入1#计数器	**
0	1	0	1	0	计数初值写入2#计数器	**
0	1	0	1	1	向控制字REG写控制字	00~10
					写读回命令(读状态)	11
0	0	1	0	0	读0#计数器当前计数值	无
					读0#计数器状态	无
0	0	1	0	1	读1#计数器当前计数值	无
					读1#计数器状态	无
0	0	1	1	0	读2#计数器当前计数值	无
					读2#计数器状态	无
0	0	1	1	1	无操作	
1	*	*	*	*	禁止使用	
0	1	1	*	*	无操作	

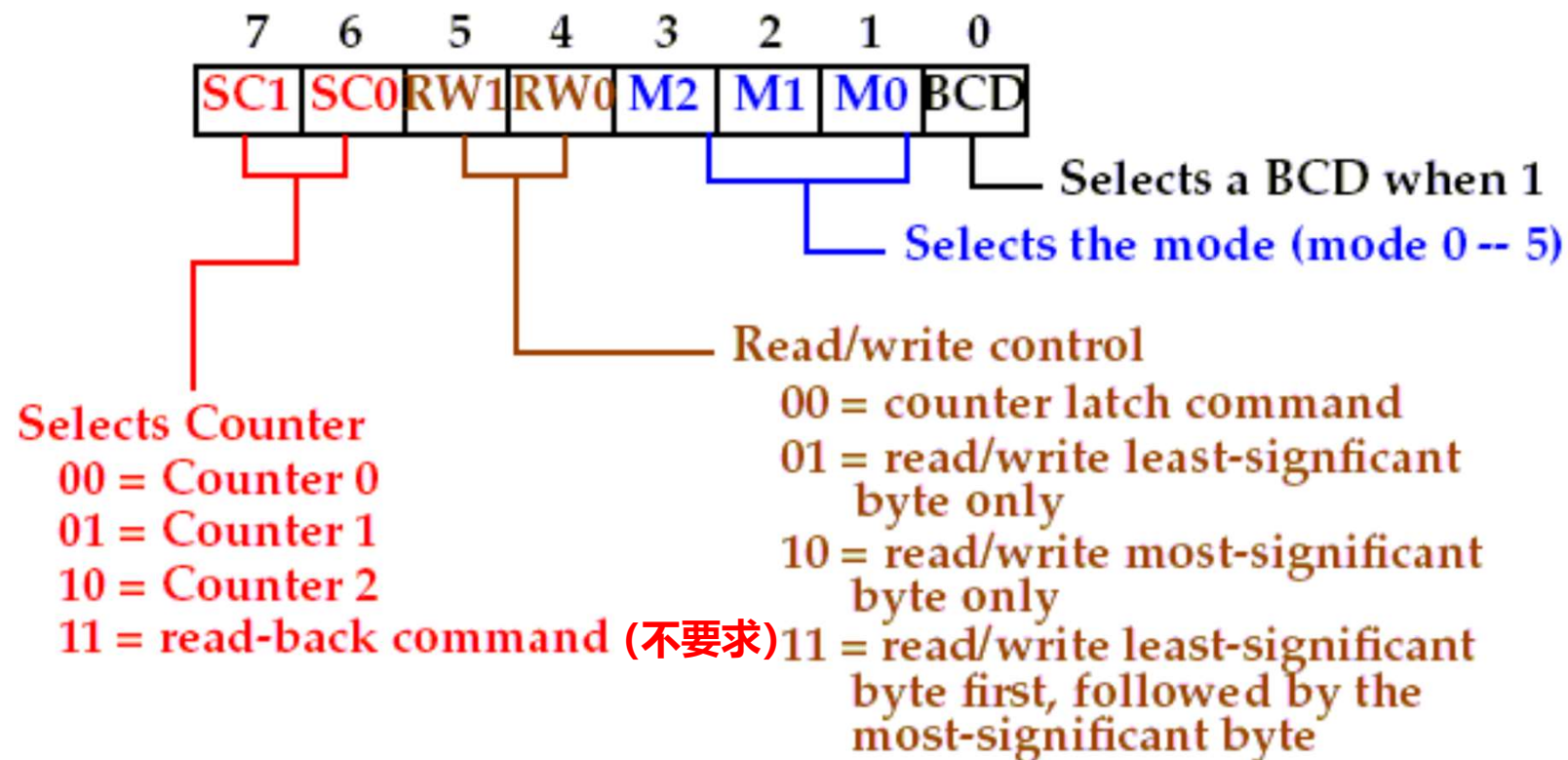






# 控制字格式

上电后，8253/4的状态是未知的，工作模式、计数值和输出也是未知的。因此，需初始化。



8254与8253相比，增加了状态回读命令





# 写操作规则

- 对每一计数器，控制字必须先于计数初值写入
- 计数初值的写入必须遵循控制字定义的计数格式（即：只读/写低字节，只读/写高字节或先读/写低字节再读/写高字节）
- 例：设置计数器1、模式3、计数初值2345H

```
timer equ 40h
```

```
mov al, 01110110b
```

```
out timer+3, al
```

```
mov ax, 2345h
```

```
out timer+1, al
```

```
mov al, ah
```

```
out timer+1, al
```





# 读操作 (1)

## ■ 简单读操作

直接读**OL**的内容。但必须使**GATE**无效或用外部逻辑禁止**CLK**信号。否则，可能读到不正确的值。

## ■ 计数器锁存方式

使用“锁存”命令，然后再读**OL**的内容。

## ■ 上述两种当前计数值的读出必须遵循控制字定义的计数格式（即：只读/写低字节，只读/写高字节或先读/写低字节再读/写高字节）





# 读操作 (2)

- 例：读计数器1的当前计数值

**mov al, 01000000b**

**out timer+3, al;**发计数器锁存命令

**in al, timer+1;**假定初始芯片时设定为先

**mov bl, al** ;读写低字节后读写高字节

**in al, timer+1**

**mov bh, al**





## 7.6.3 8253/8254工作方式





# 工作方式

- 方式0: 计数结束中断
- 方式1: 可编程单稳触发器
- 方式2: 频率发生器
- 方式3: 方波发生器
- 方式4: 软件触发选通信号
- 方式5: :硬件触发选通信号





# 工作方式公共操作说明

- 写入控制字后，**OUT**端进入已知初始状态（高电平或低电平）
- 写入初值后，经过**CLK**  $\uparrow\downarrow$ ，计数初值装入**CE**部件。
- 在**CLK**的 $\uparrow$ ，采样门控信号**GATE**

**GATE**电平触发: **GATE**

**CLK**

**GATE**边沿触发: **GATE**

**CLK**

边沿触发器置位  
采样

- 在**CLK**的 $\downarrow$ ，计数器**CE**减“1”或“2”





# 方式0 – 计数结束中断

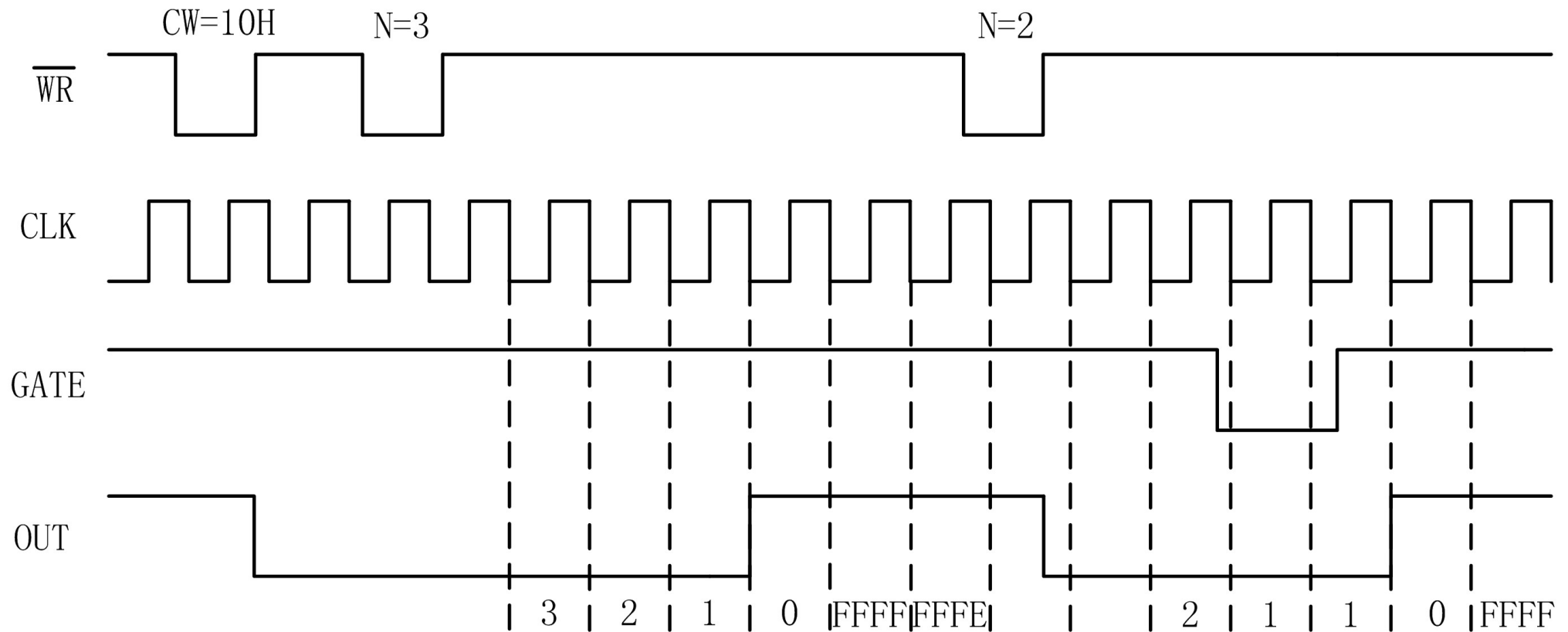
- 由软件启动、不能自动重复的计数方式。计数结束时产生的上升沿可作为中断请求信号。
- 写入控制字后，**OUT**输出为“0”
- **GATE=1**，使能计数器  
    **GATE=0**，暂停计数器。**GATE**不影响**OUT**
- 写入计数初值后，在下一个**CLK**的  $\uparrow\downarrow$ ，置计数初值到计数执行部件**CE**
- 计数为“0”后，**OUT**输出为“1”
- 计数初值一次有效
- 在计数过程中可重新设置计数初值





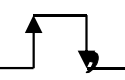
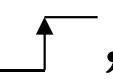


# 方式0波形





# 方式2 – 频率发生器 (1)

- 由软件 (**GATE=1**, 写入初值) 或硬件 (**GATE**的上升沿) 启动、可**自动**重复计数。可用作频率发生器。
- 写入控制字后, **OUT**输出为 “1”
- 写入计数初值后, 在下一个**CLK**的  置计数初值到计数执行部件**CE**。若在计数过程中写入新计数初值, 不影响当前计数过程, 而是在该计数过程结束后被加载
- **GATE=1**, 使能计数器  
**GATE=0**, 暂停计数器。但当**OUT=0**时, 如果 **GATE=0**, 则强置**OUT**输出为 “1”。  
**GATE**的 , 重置计数初值。





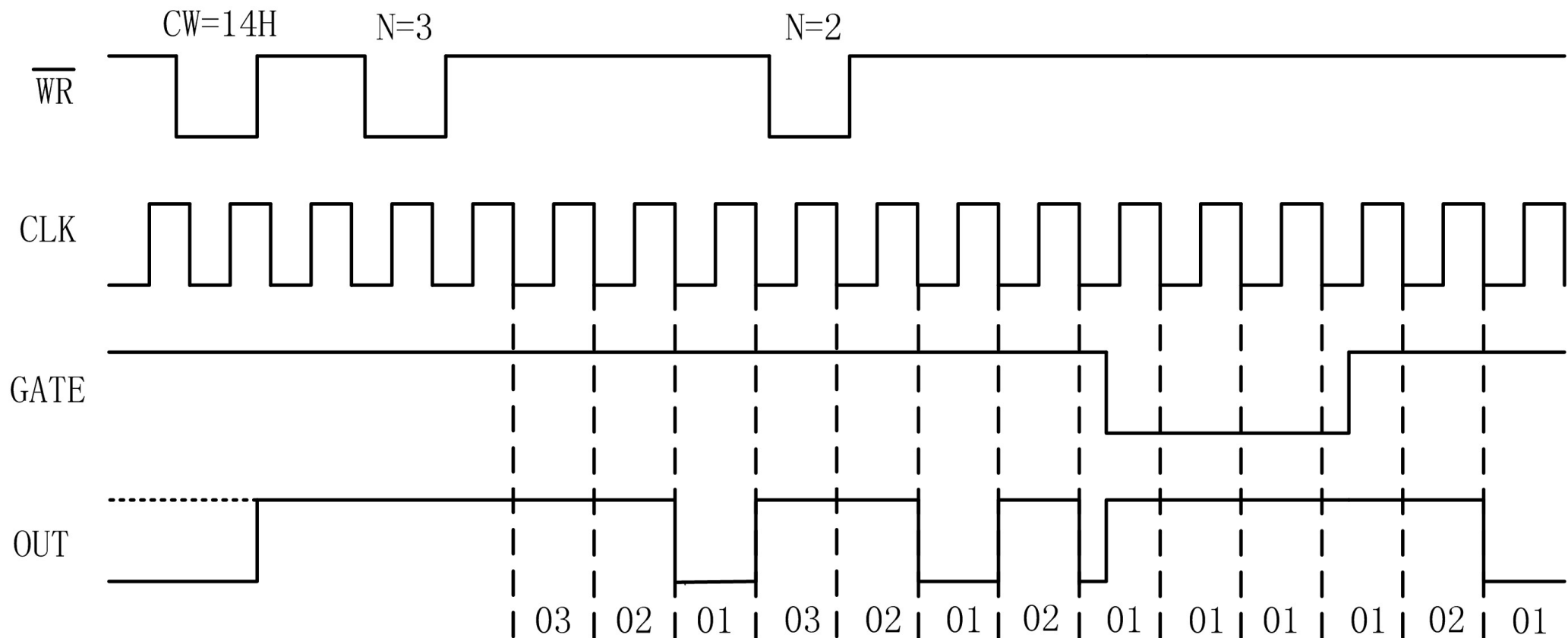
# 方式2 – 频率发生器 (2)

- 计数为“1”后，OUT输出为一个CLK宽度的负脉冲。然后自动将计数初值寄存器的内容加载到CE部件。





# 方式2波形





# 方式3 – 方波发生器（1）

- 由软件或硬件启动、可**自动**重复计数。**OUT**端的输出为一半时间为高另一半时间为低电平方波。类似方式2。
- 写入控制字后， **OUT**输出为“1”
- 写入计数初值后，在下一个**CLK**的 $\downarrow$ ，置计数初值（奇数初值需要减“1”）到计数执行部件**CE**。
- **GATE=1**，使能计数器  
**GATE=0**，暂停计数器。当**OUT=0**时，如果**GATE=0**，则置**OUT**为“1”。  
**GATE**的 $\uparrow$ ，重置计数初值。





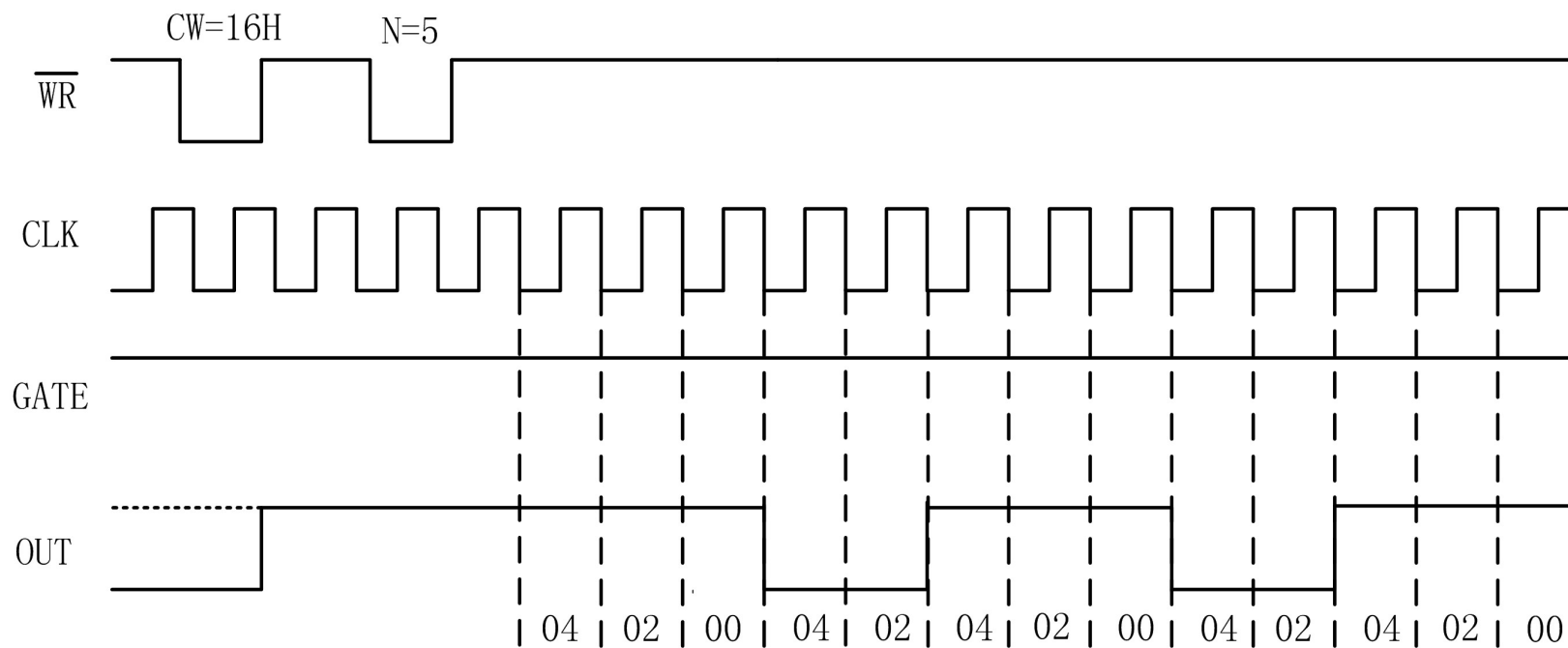
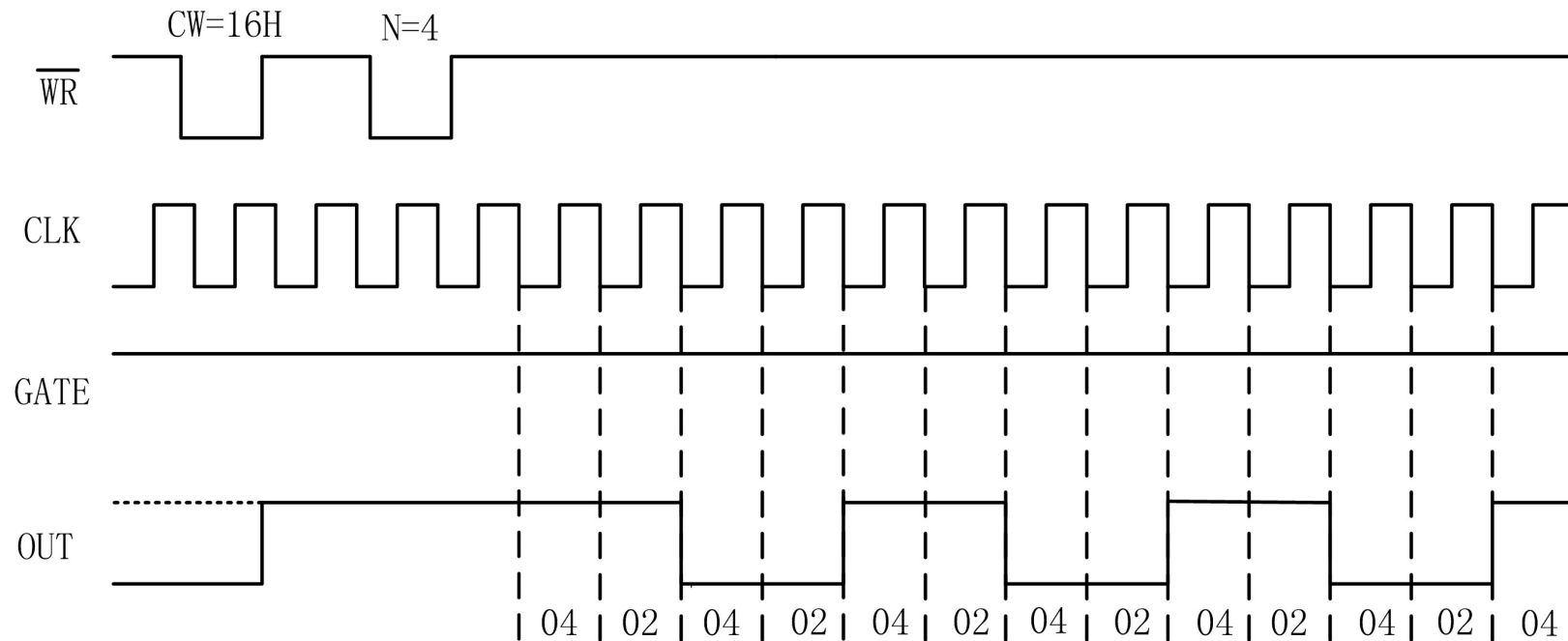
# 方式3 – 方波发生器 (2)

- 计数初值为偶数，则输出为CLK的 $n$ 分频，占空比为1: 1。每一个CLK的下降沿使计数值减“2”，到“0”后使OUT的输出反转，并重载计数初值。
- 计数初值为奇数，则输出为CLK的 $n$ 分频，占空比为 $(n+1/2) : (n-1/2)$ 。每一个CLK的下降沿使计数值减“2”，计数值为“0”后或再减“2”使OUT的输出反转，并重载计数初值。





# 方式3波形





# 扩展定时/计数范围

当定时长度不够时，可把2个或3个计数通道**串联**起来使用，甚至可把多个**8253**串联起来使用。  
例如：**CLK**频率为**1MHz**，要求在**OUT1**端产生频率**1Hz**的方波。

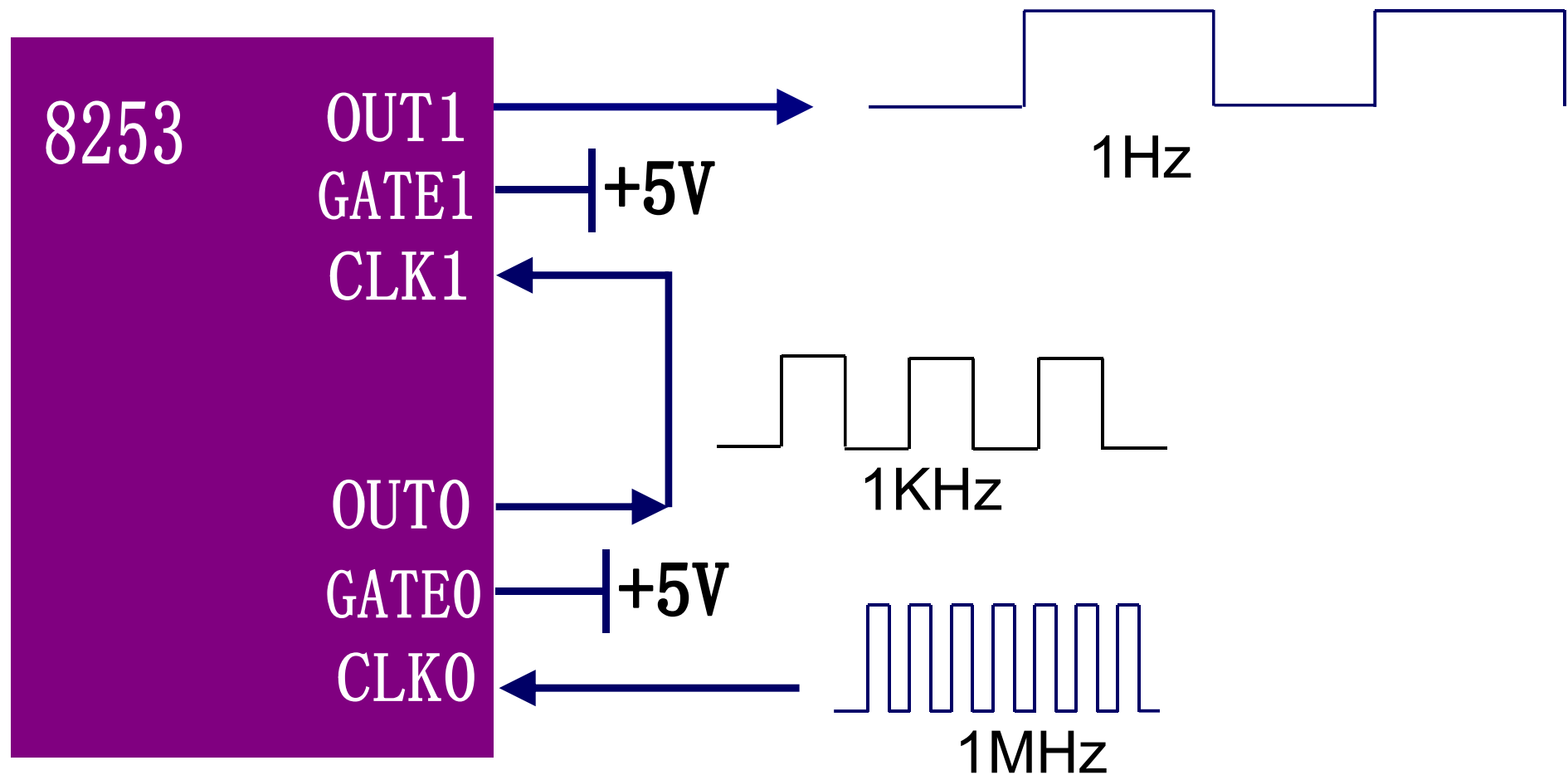
可将计数器**0**、**1**串联，工作方式都均为方式**3**，计数初值均为**1000**。连接方法见下页。







# 扩展定时/计数范围





## 7.6.4 8253/4的应用举例





# 例1：波形输出

- 采用**8253**作定时/计数器，已知其端口地址为**120H~123H**，其输入**8253**的时钟频率为**2MH**。
- 要求计数器**0**每**10ms**输出一个**CLK**脉冲宽度的负脉冲；用计数器**1**产生**10KHz**的连续方波信号，计数器**2**在定时**5ms**后产生输出高电平。
- 画出线路连接图，并编写初始化程序。





# 初值计算及方式选择

## ■ 确定各计数器工作方式:

**CNT0: 方式2, 16位计数值**

**CNT1: 方式3, 低8位计数值**

**CNT2: 方式0, 16位计数值**

## ■ 计算计数初值:

**CNT0:  $10\text{ms}/0.5\mu\text{s}=20000$**

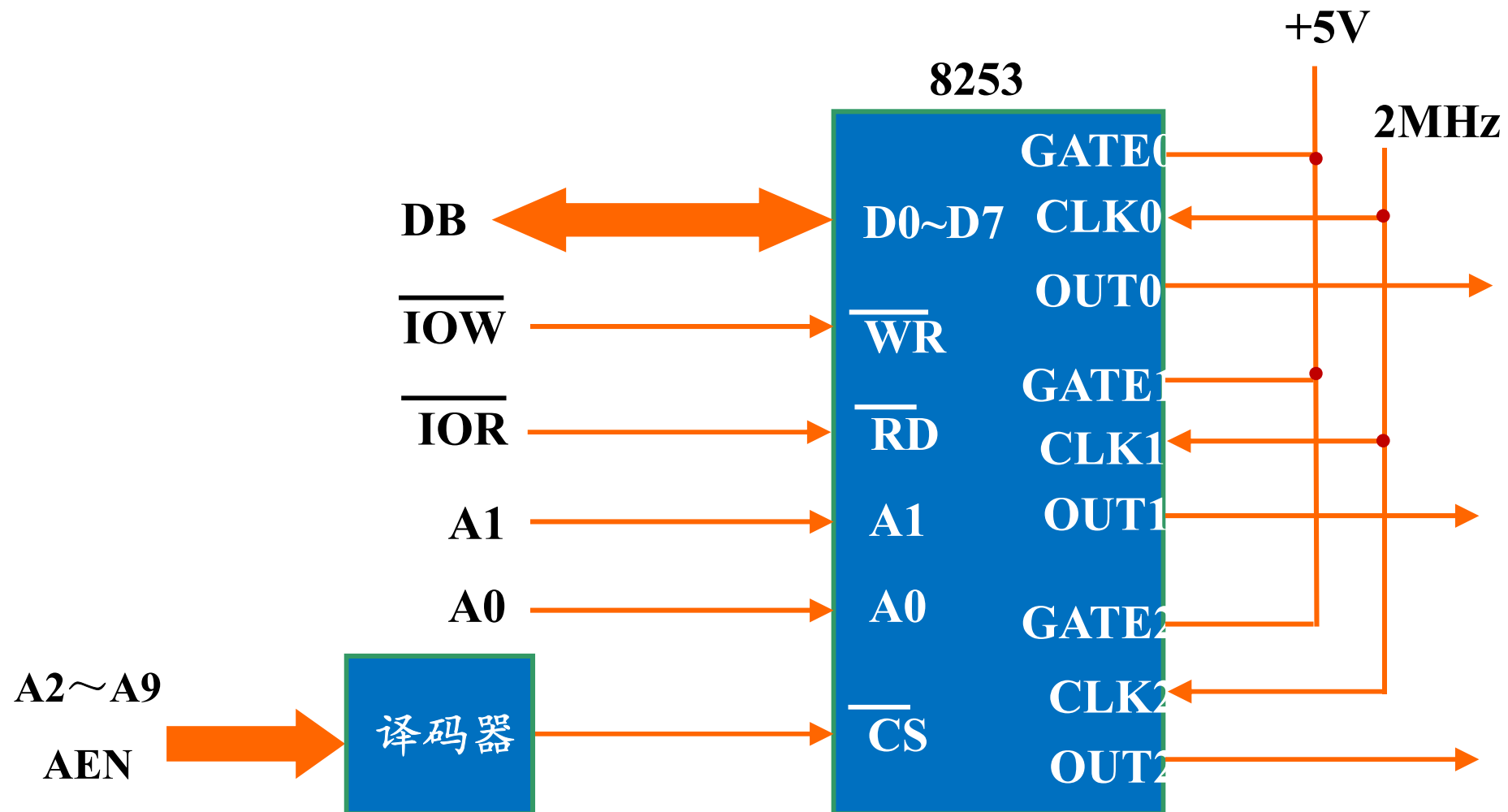
**CNT1:  $2\text{ MHz}/10\text{KHz}=200$**

**CNT2:  $5\text{ms}/0.5\mu\text{s}=10000$**





# 电路连接





# 初始化程序

CNT0:

MOV DX, 0123H

MOV AL, 34H

OUT DX, AL

MOV DX, 0120H

MOV AX, 20000

OUT DX, AL

MOV AL, AH

OUT DX, AL

CNT1:

.....





# 8253/4在PC系列机中的应用

- 8253/4的计数器0、1、2占用的端口地址分别为40H、41H、42H。控制口占用的端口地址为43H。
- 计时器0（工作在方式3）  
产生实时时钟信号，占用8259的IRQ0。频率为18.2次/秒。
- 计时器1（工作在方式2）  
用于DRAM的周期刷新。15.12us刷新一次
- 计时器2（工作在方式3）  
扬声器发声。其频率由程序设定。





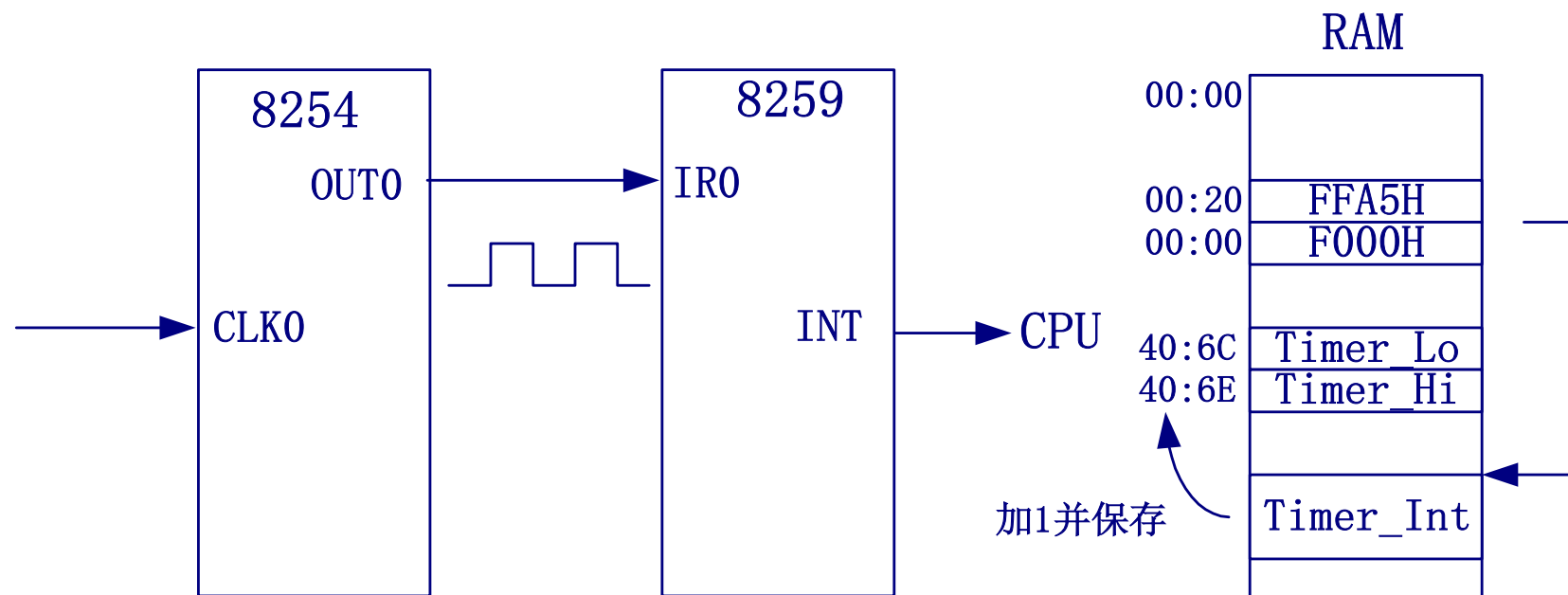
# 例2：日时钟的实现

- 方波频率为18.2次/秒（每隔55ms中断一次）
- 计时单元Timer\_Hi:Timer\_Lo的计数范围为：

**0000:0000 ~ 0018:00B0H**

**0018:00B0H = 1573040**

**$1573040 \times 0.055\text{s} = 24 \text{ hour}$**







# 例3：硬件延时

用计时器0完成5秒延时

- 利用BIOS的INT 1AH的0号功能调用，出口参数为：  
CX=计时器的高字  
DX=计时器的低字
- 5秒内的计数为： $5000\text{ms}/54.945\text{ms}=91$

```
mov ah, 00h
```

```
int 1ah ;读日时钟
```

```
add dx, 91
```

```
mov bx, dx
```

```
lp: mov ah, 00h
```

```
int 1ah ;再读日时钟
```

```
cmp dx, bx
```

```
jnz lp
```

```
...
```

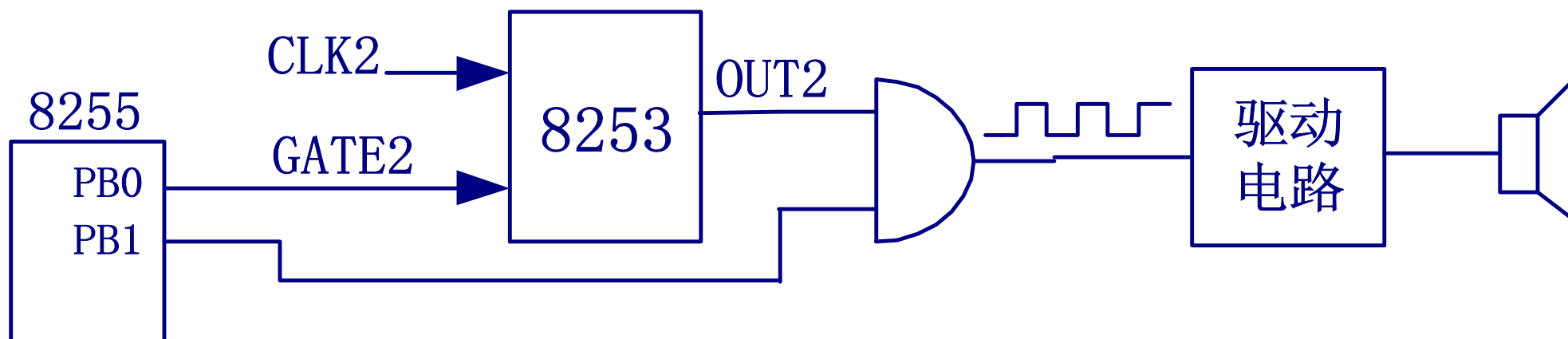




## 例4：发声器

利用8253发出600Hz的长/短音，按任意键开始发声；按ESC键停止发声。CLK2的频率为1.19318MHz。8253的端口地址为40~43H。8255的端口地址为60~63H。

- 输出600Hz的方波即可产生600Hz的单频音，其计数初值为 $1.19318 \times 10^6 / 600 = 1983$





# 汇编源程序 (1)

```

mov al, 10110110b
out 43h, al
mov ax, 1983
out 42h, al
mov al, ah
out 42h, al
in al, 61h ;8255B口
and al, 0fch ;或0fdh
out 61h, al ;关闭扬声器

```

wait:

```

mov ah, 0bh
int 21h
cmp al, 00h

```

```

je wait ;有键按下?
lop:mov bl, 6
call ssp;发长音
mov ah, 0bh
int 21h
cmp al, 00h
je contin ;无键按下发短音
mov ah, 08h ;检测ESC键
int 21h
cmp al, 1bh
je quit ;是ESC键退出

```





# 汇编源程序 (2)

```

contin:mov bl, 1
        call ssp      ;发短音
        jmp lop
quit:   in al, 61h
        and al, 0fch
        out 61h, al
        mov ax, 4c00h
        int 21h;返回DOS
  
```

```

ssp proc
        in  al, 61h
        or  al, 03h;或02h
        out 61h, al
        xor cx, cx
lp1: loop lp1
        dec bl
        jnz lp1      ;延时
        and al, 0fch ;或0fdh
        out 61h, al ;关闭扬声器
        ret
ssp endp
  
```

