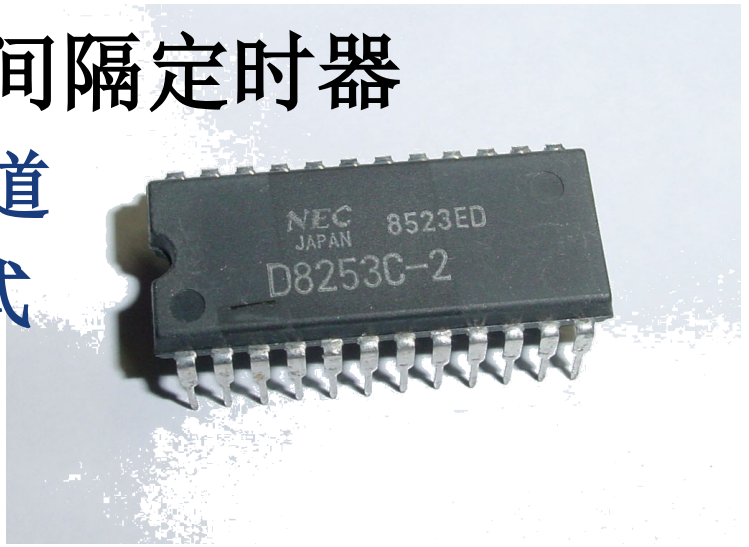
第 十一 章
常用接口技术

11.1 定时控制接口

- 定时控制具有极为重要的作用
  - 微机控制系统中常需要定时中断、定时检测、定时扫描等
  - 实时操作系统和多任务操作系统中要定时进行进程调度
  - PC 机的日时钟计时、 DRAM 刷新定时和扬声器音调控制都采用了定时控制技术
- 可编程定时器芯片
  - 软硬件相结合、方便灵活的定时电路
- 软件延时方法
  - 处理器执行延时子程序

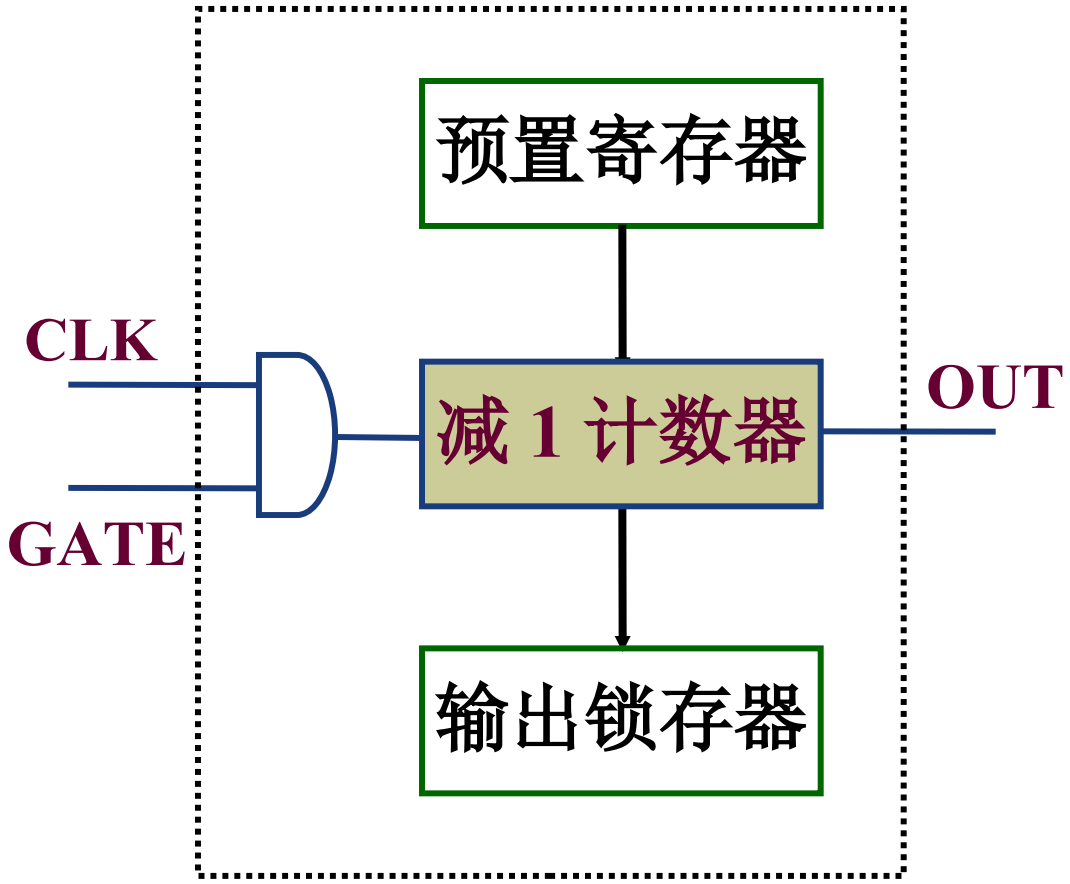
11.1.1 8253/8254 定时器

- 定时器（计数器）：由数字电路中的计数电路构成，记录输入脉冲的个数
  - 脉冲信号具有一定随机性，往往通过脉冲的个数可以获知外设的状态变化次数（计数）
  - 脉冲信号的周期固定（使用高精度晶振产生脉冲信号），个数乘以周期就是时间间隔（定时）
- Intel 8253/8254 可编程间隔定时器
  - 3 个独立的 16 位计数器通道
  - 每个计数器有 6 种工作方式



1. 内部结构和引脚

- 3 个相互独立的计数器通道，结构完全相同
  - 计数器 0
  - 计数器 1
  - 计数器 2
- 每个计数器通道
  - 16 位减法计数器
  - 16 位预置寄存器
  - 输出锁存器



定时器外设引脚

- CLK 时钟输入信号
  - 在计数过程中，此引脚上每输入一个时钟信号（下降沿），计数器的计数值减 1
- GATE 门控输入信号
  - 控制计数器工作，可分成电平控制和上升沿控制两种类型
- OUT 计数器输出信号
  - 当一次计数过程结束（计数值减为 0 ）， OUT 引脚上将产生一个输出信号

连接处理器引脚

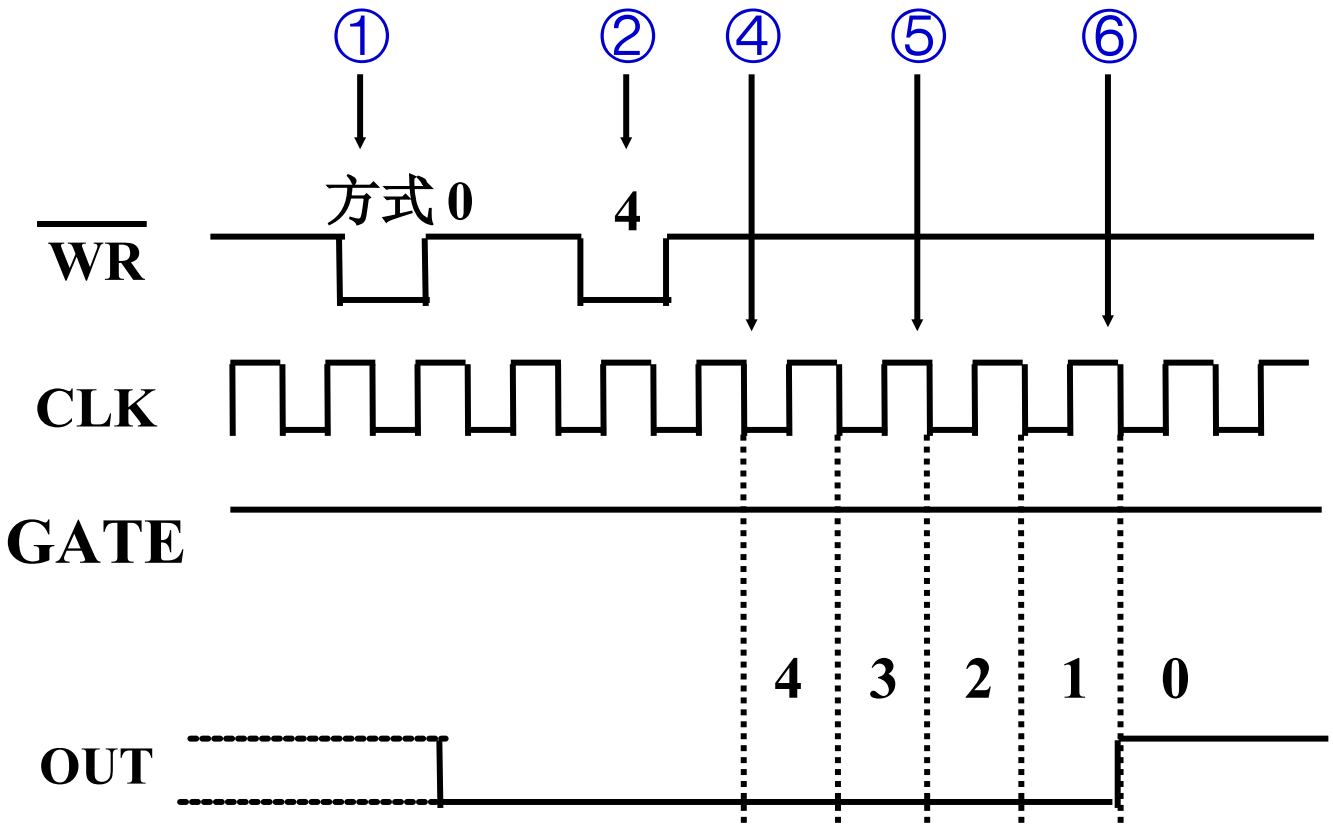
- D0 ～ D7 数据线　　A0 ～ A1 地址线
- RD* 读信号　　WR* 写信号
- CS* 片选信号

CS*	A1	A0	I/O 地址	读操作 RD*	写操作 WR*
0	0	0	40H	读计数器 0	写计数器 0
0	0	1	41H	读计数器 1	写计数器 1
0	1	0	42H	读计数器 2	写计数器 2
0	1	1	43H	无操作	写控制字

2． 工作方式

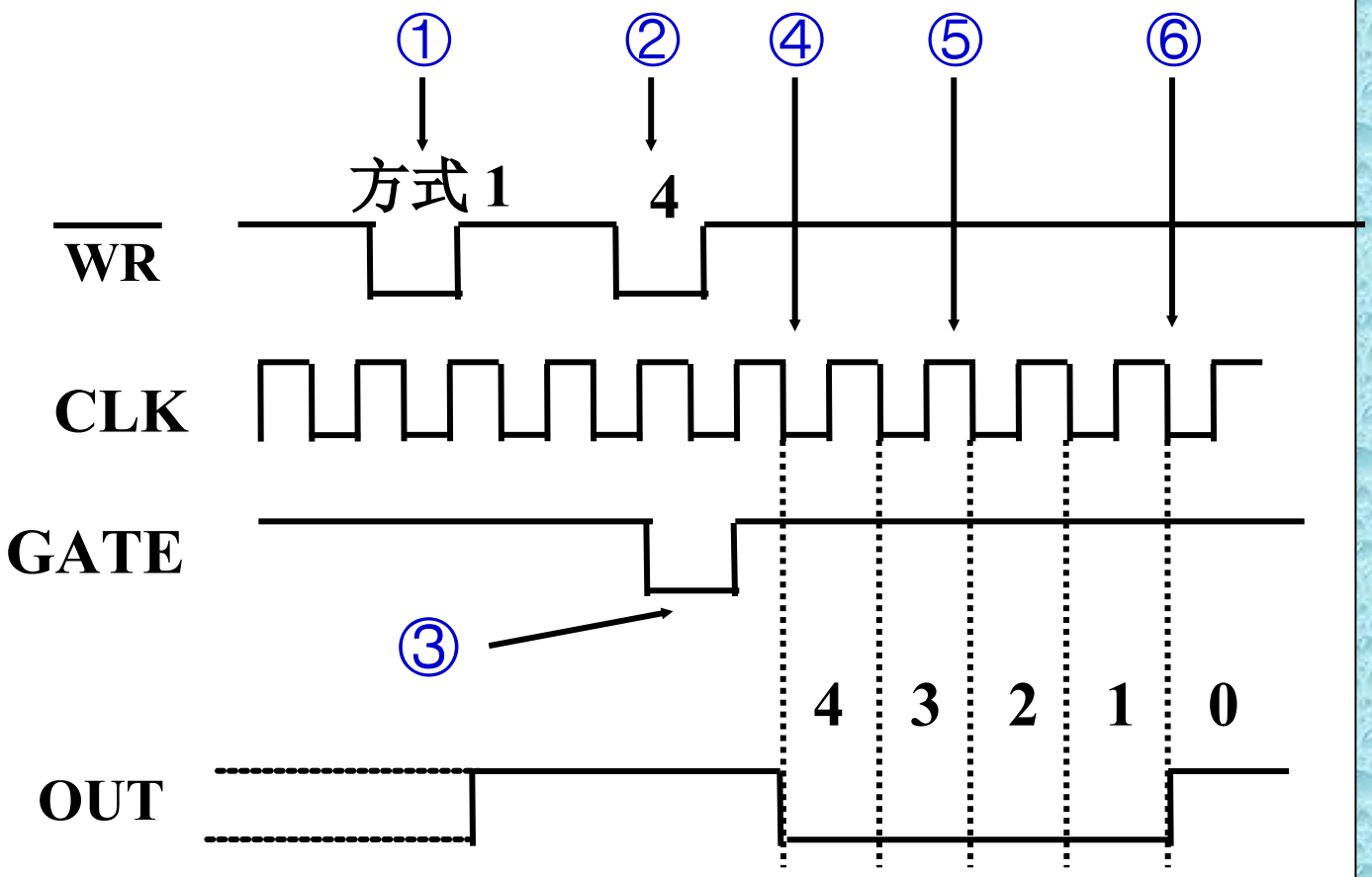
- 8253 有 6 种工作方式，由方式控制字确定
- 每种工作方式的过程类似：
  - ① 设定工作方式
  - ② 设定计数初值
  - [③ 硬件启动]
  - ④ 计数初值进入减 1 计数器
  - ⑤ 每输入一个时钟计数器减 1 的计数过程
  - ⑥ 计数过程结束

定时器方式0 ：计数结束中断



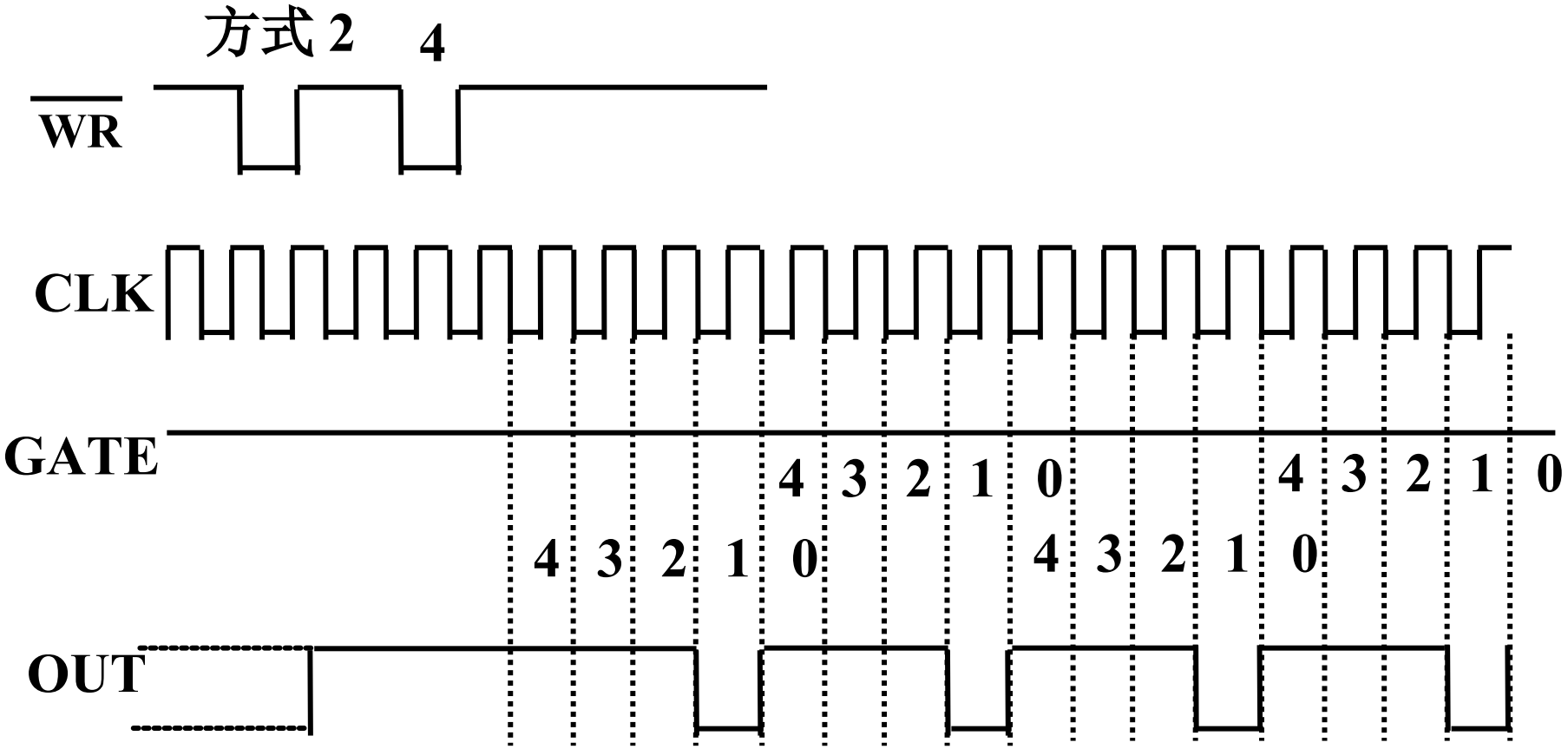
⑥ 计数结束

定时器方式 1 ：可编程单稳脉冲

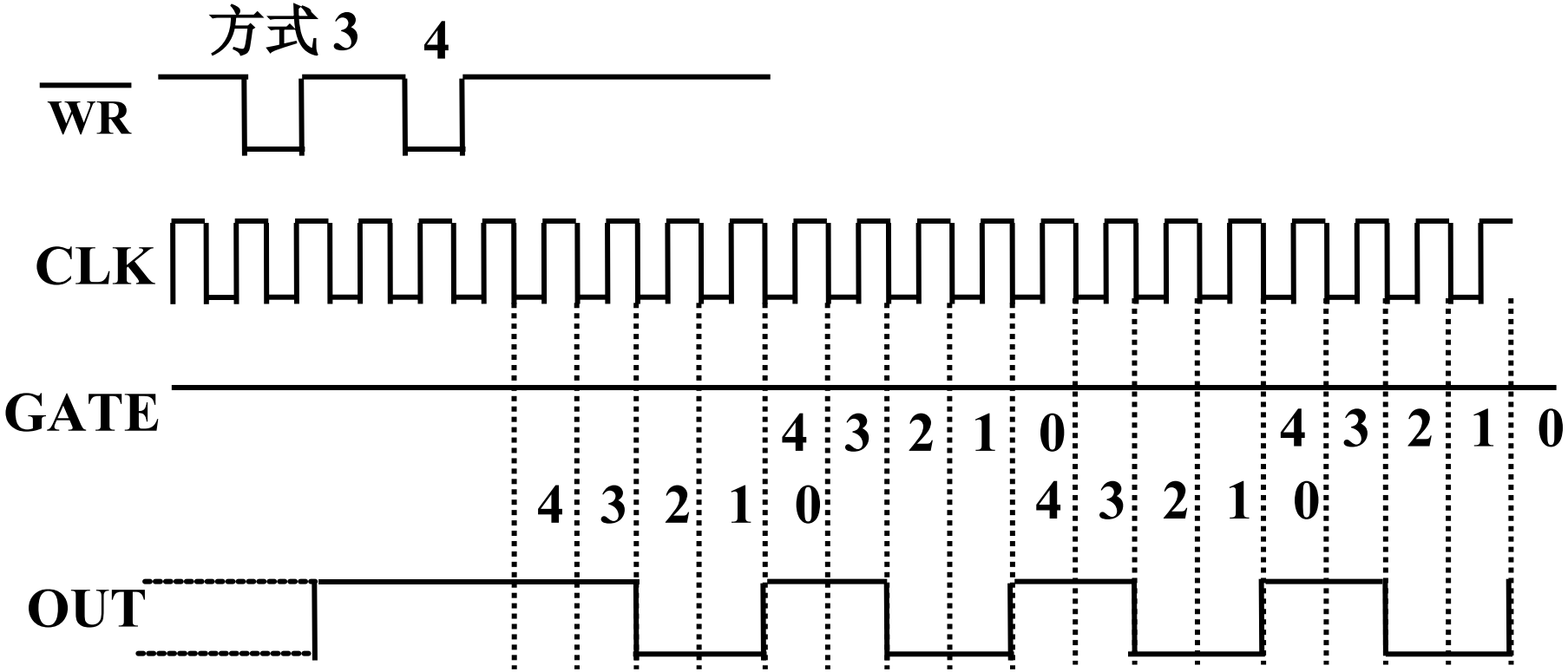


⑥ 计数结束

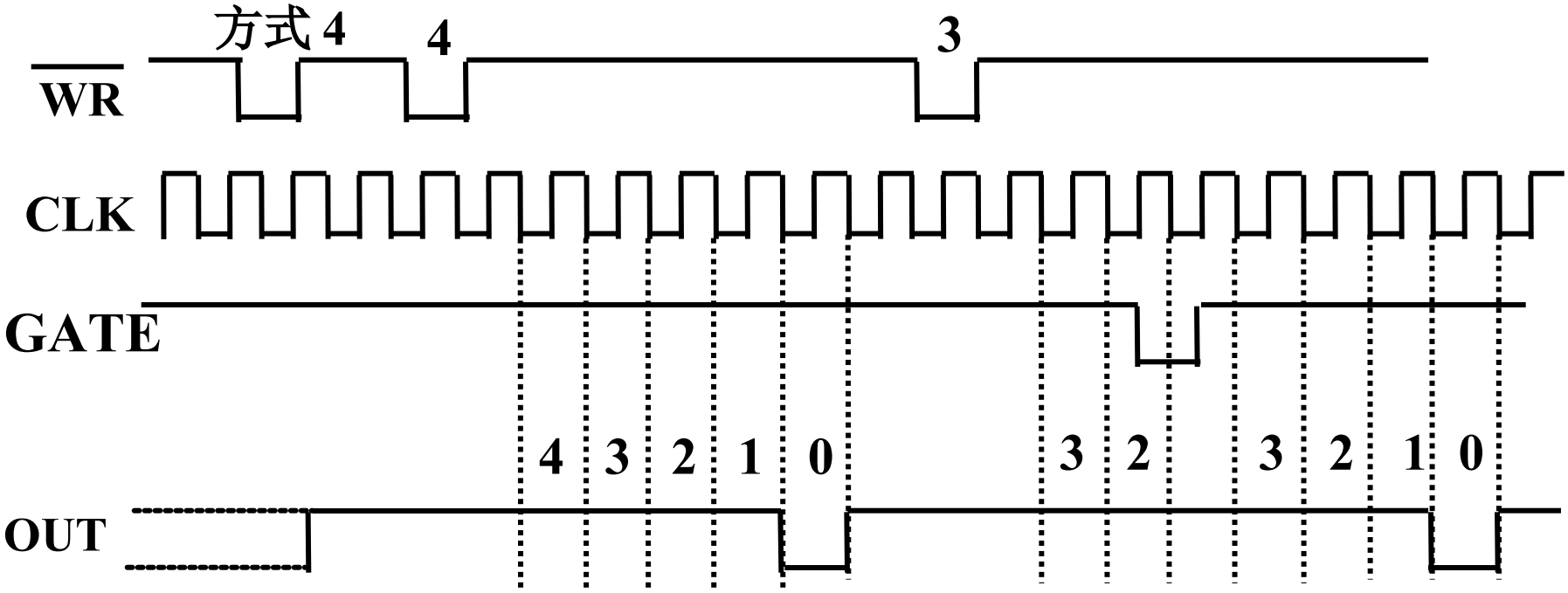
定时器方式 2 ：频率发生器（分频器）



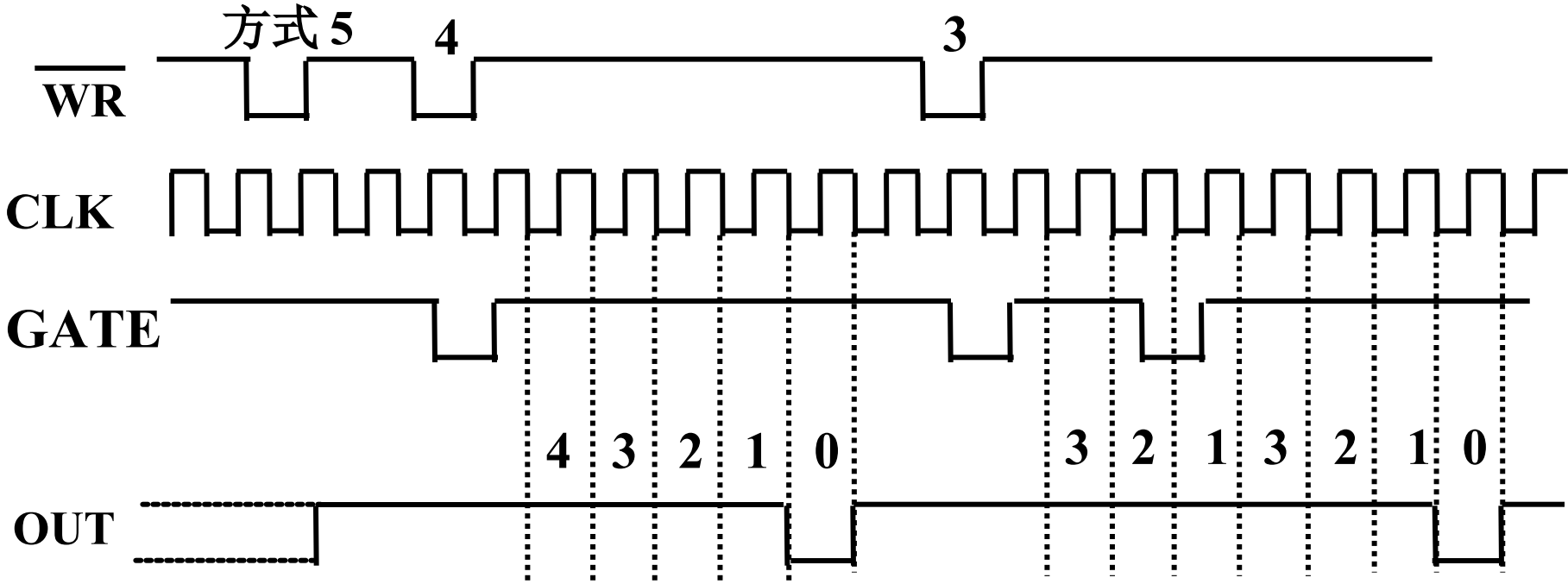
定时器方式 3 ：方波发生器



定时器方式 4 ：软件触发选通信号

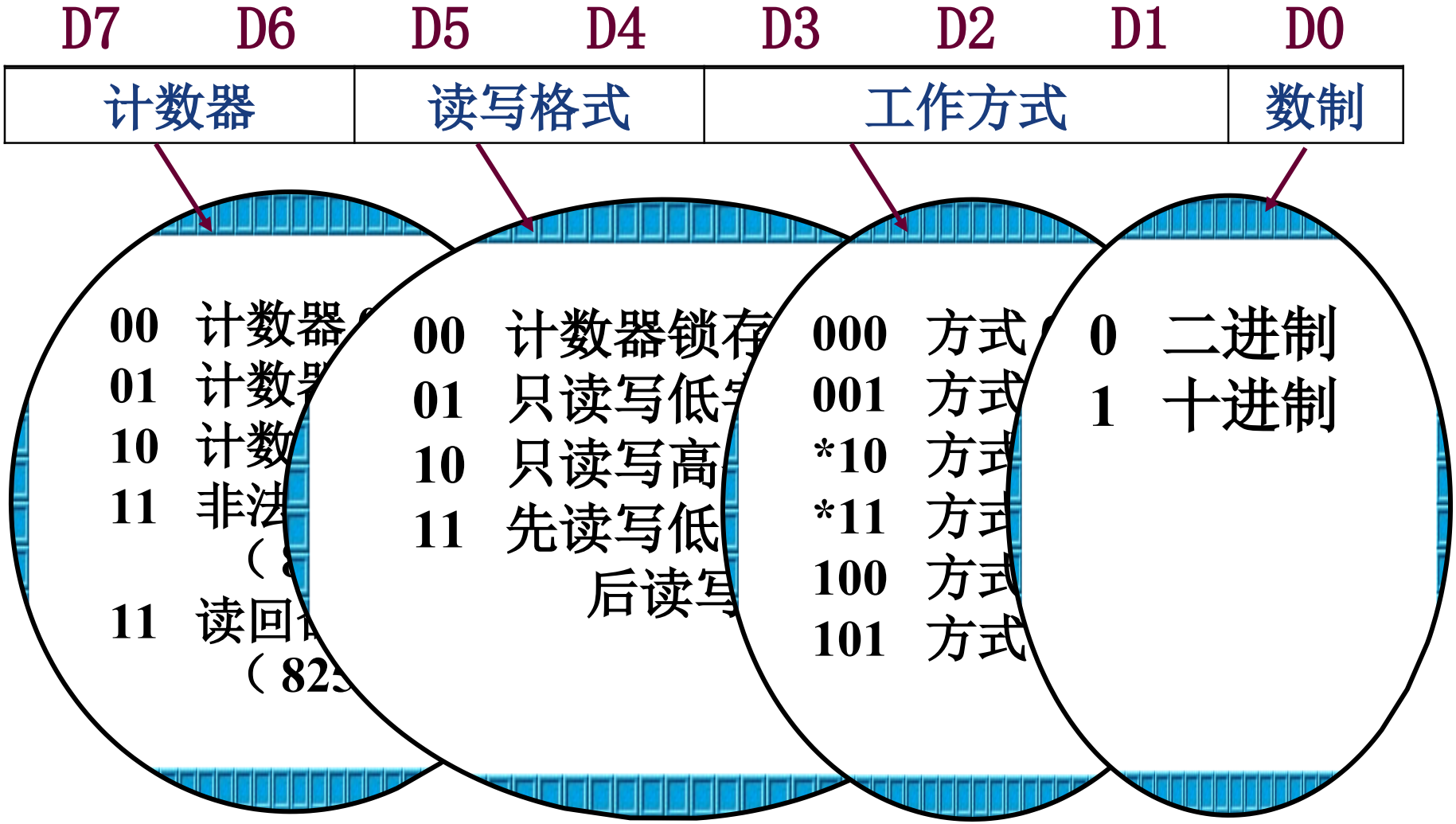


定时器方式 5 ：硬件触发选通信号



3． 编程：写入方式控制字

- 控制字写入控制字 I/O 地址： A1A0 ＝ 11



方式控制字编程示例

```
; 8253 的计数器 0 、 1 、 2 端口和控制端口地址： 40H ～ 43H
; 设置其中计数器 0 为方式 0
; 采用二进制计数，先低后高写入计数值
mov al,30h
; 方式控制字： 30H ＝ 00 11 000 0B
out 43h,al
; 写入控制端口： 43H
```

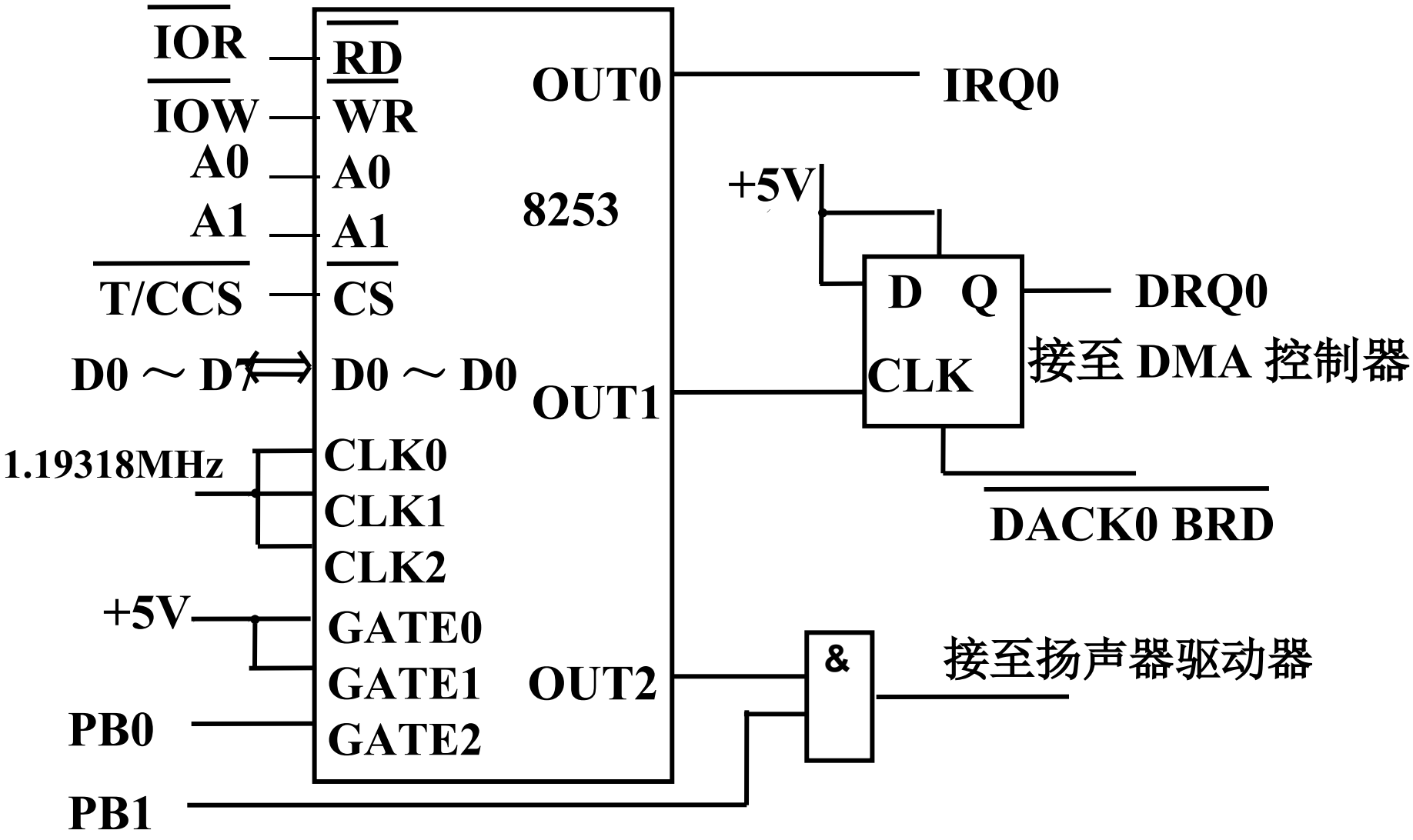
3． 编程：写入计数值

- 选择二进制时
  - 计数值范围： 0000H ～ FFFFH
  - 0000H 是最大值，代表 65536
- 选择十进制（ BCD 码）
  - 计数值范围： 0000 ～ 9999
  - 0000 代表最大值 10000
- 计数值写入计数器各自的 I/O 地址
- 按方式控制字规定的读写格式进行

计数值编程示例

```
; 8253 的计数器 0 、 1 、 2 端口和控制端口地址： 40H ～ 43H
; 设置计数器 0 采用二进制计数
; 写入计数初值： 1024 （＝ 400H ）
mov  ax,1024     ; 计数初值： 1024 （＝ 400H ）
; 写入计数器 0 地址： 40H
out  40h,al      ; 写入低字节计数初值
mov  al,ah
out  40h,al      ; 写入高字节计数初值
```

11.1.2 定时器的应用


IOR
RD
IOW
WR
A0
A0
A1
A1
T/CCS
CS
D0 ～ D7
D0 ～ D0
8253
OUT0
OUT1
OUT2
1.19318MHz
CLK0
CLK1
CLK2
+5V
GATE0
GATE1
GATE2
PB0
PB1
+5V
D
Q
CLK
DRQ0
接至 DMA 控制器
DACK0 BRD
&
接至扬声器驱动器
IRQ0

1. 定时中断

```
mov al,36h          ; 计数器0为方式3，二进制计数
; 先低后高写入计数值
out 43h,al          ; 写入方式控制字
mov al,0            ; 计数值为0
out 40h,al          ; 写入低字节计数值
out 40h,al          ; 写入高字节计数值
```

- 计数器0：方式3，计数值：65536，输出方波
  频率：1.19318MHz÷65536 ＝ 18.206Hz，不断产生
- OUT0端接8259A的IRQ0，每秒产生18.206次中断请求，或说每隔55ms（54.925493ms）申请一次中断
- DOS系统利用计数器0的这个特点，通过08号中断服务程序实现了日时钟计时功能

2． 定时刷新

- 需要重复不断提出刷新请求
- 门控总为高，选择方式 2 或 3
- 2ms 内刷新 128 次，即 15.6μs 刷新一次
- 计数初值为 18

```
mov  al,54h      ；计数器 1 为方式 2
；采用二进制计数，只写低 8 位计数值
out  43h,al      ；写入方式控制字
mov  al,18       ；计数初值为 18
out  41h,al      ；写入计数值
```

3. 扬声器控制

```
;发音频率设置子程序
;入口参数： AX = 1.19318×10^6÷ 发音频率
speaker proc
        push ax          ;暂存入口参数
        mov al,0b6h ;定时器 2 为方式 3，先低后
高
        out 43h,al   ;写入方式控制字
        pop ax           ;恢复入口参数
        out 42h,al   ;写入低 8 位计数值
        mov al,ah
        out 42h,al   ;写入高 8 位计数值
        ret
speaker endp
```

扬声器发音控制

```
speakon proc          ;扬声器开子程序
        push ax
        in al,61h     ;读取61H端口的原控制信息
        or al,03h     ;D1D0＝PB1PB0＝11，其他不变
        out 61h,al    ;直接控制发声
        pop ax
        ret
speakon endp
```

```
;扬声器关子程序
and al,0fch
;D1D0＝PB1PB0＝00，其他不变
```

〔例 11-1〕控制扬声器程序

```
        ;数据段
freq    dw 1193180/600      ;给一个 600Hz 的频率
        ;代码段
        mov ax,freq
        call speaker        ;设置扬声器的音调
        call speakon        ;打开扬声器声音
        call readc          ;等待按键
        call speakoff       ;关闭扬声器声音
        ;子程序
        ……
```

输出：明确向哪个端口输出什么数据
输入：清楚从哪个端口输入什么数据

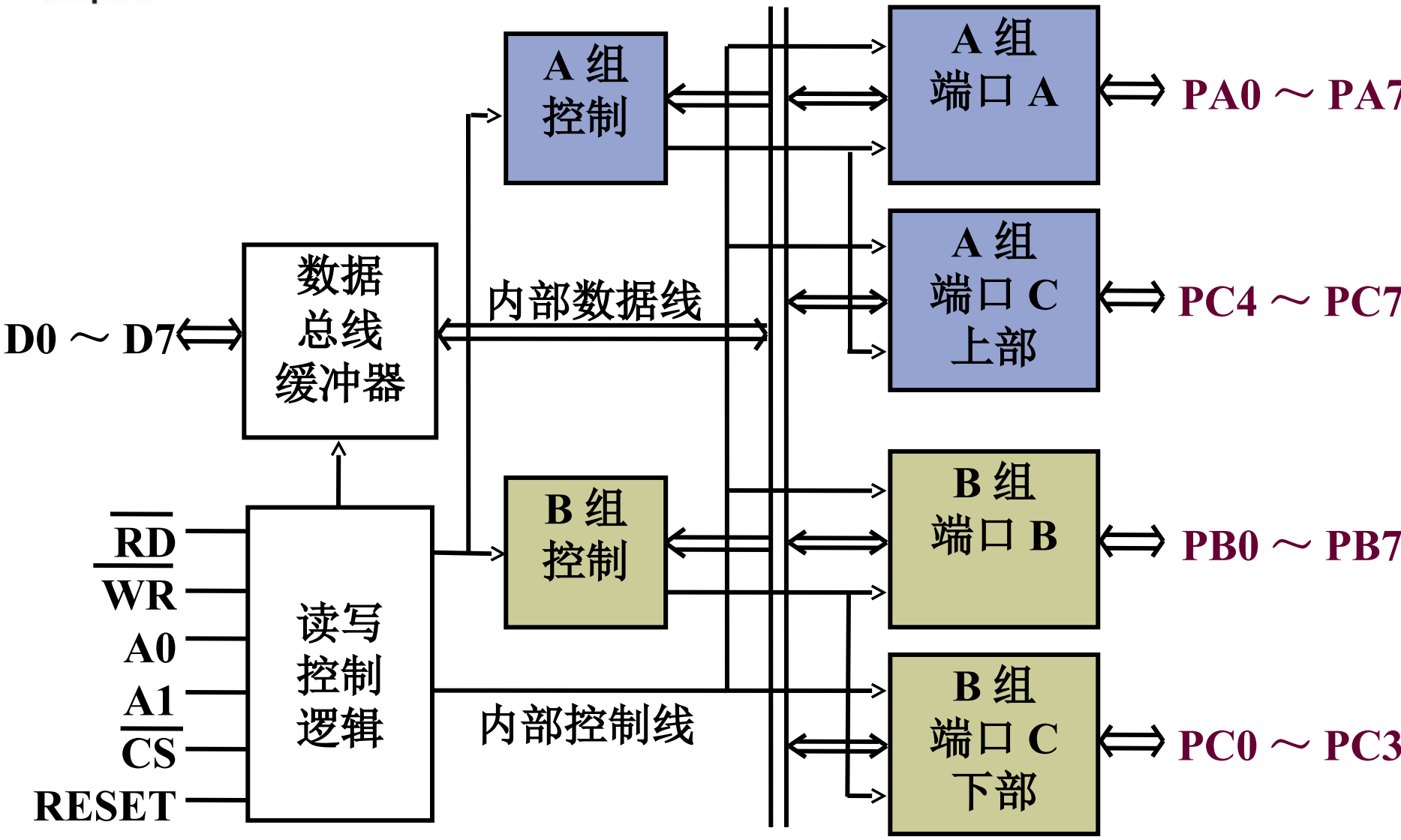
11.2 并行接口

- 并行数据传输：以计算机字长，通常是8、16或32位为传输单位，利用8、16或32个数据信号线一次传送一个字长的数据
  - 适合于外部设备与微机之间进行近距离、大量和快速的信息交换，如微机与并行接口打印机、磁盘驱动器等
  - 微机系统中最基本的信息交换方法，例如系统板上各部件之间的数据交换
- 并行数据传输需要并行接口的支持

11.2.1 并行接口电路 8255

- 具有多种功能的可编程并行接口电路芯片
  - 最基本的接口电路：三态缓冲器和锁存器
  - 与 CPU 间、与外设间的接口电路：状态寄存器和控制寄存器
  - 还有端口的译码和控制电路、中断控制电路
- 分 3 个端口，共 24 个外设引脚
- 共 3 种输入输出工作方式
  - 方式 0：基本输入输出方式
  - 方式 1：选通输入输出方式
  - 方式 2：双向选通传送方式

1. 内部结构和引脚


A 组
控制
A 组
端口 A
PA0 ～ PA7
A 组
端口 C
上部
PC4 ～ PC7
数据
总线
缓冲器
D0 ～ D7
内部数据线
B 组
控制
B 组
端口 B
PB0 ～ PB7
RD
WR
A0
A1
CS
RESET
读写
控制
逻辑
内部控制线
B 组
端口 C
下部
PC0 ～ PC3

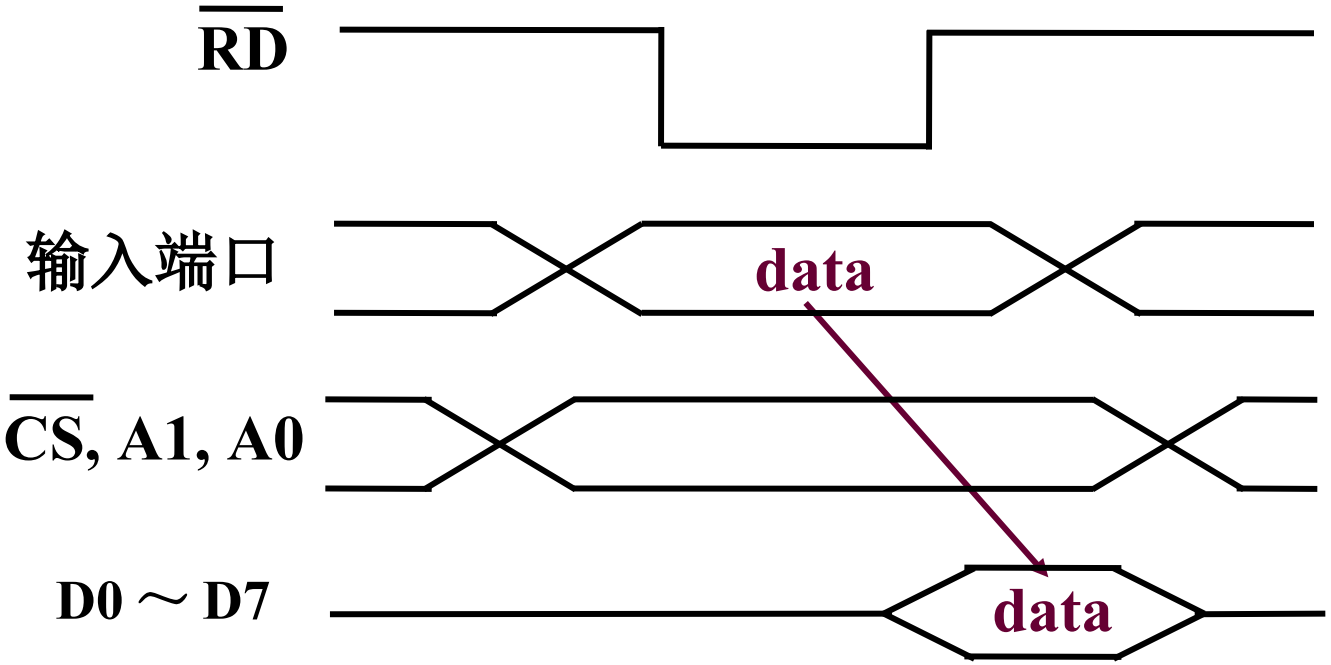
8255 外设数据端口

- 端口 A： PA0 ～ PA7
  - A 组，支持工作方式 0 、 1 、 2
  - 常作数据端口，功能最强大
- 端口 B： PB0 ～ PB7
  - B 组，支持工作方式 0 、 1
  - 常作数据端口
- 端口 C： PC0 ～ PC7
  - 仅支持工作方式 0 ，分两个 4 位，每位可独立操作
  - A 组控制高 4 位 PC4 ～ PC7 ， B 组控制低 4 位 PC0 ～ PC3
  - 可作数据、状态和控制端口
  - 控制最灵活，最难掌握

2. 工作方式0：基本输入输出方式

- 方式0输入

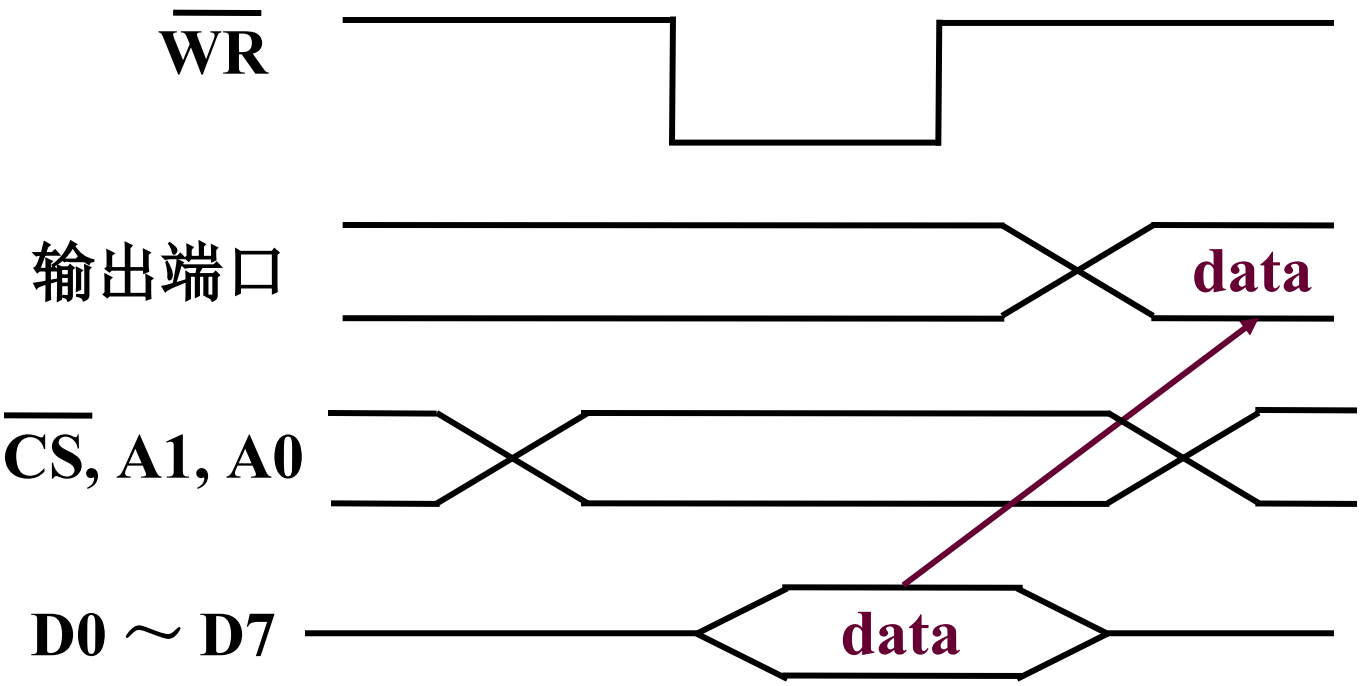
执行输入IN指令，输入外设数据



2. 工作方式0：基本输入输出方式

- 方式0输出

执行输出OUT指令，将数据送给外设

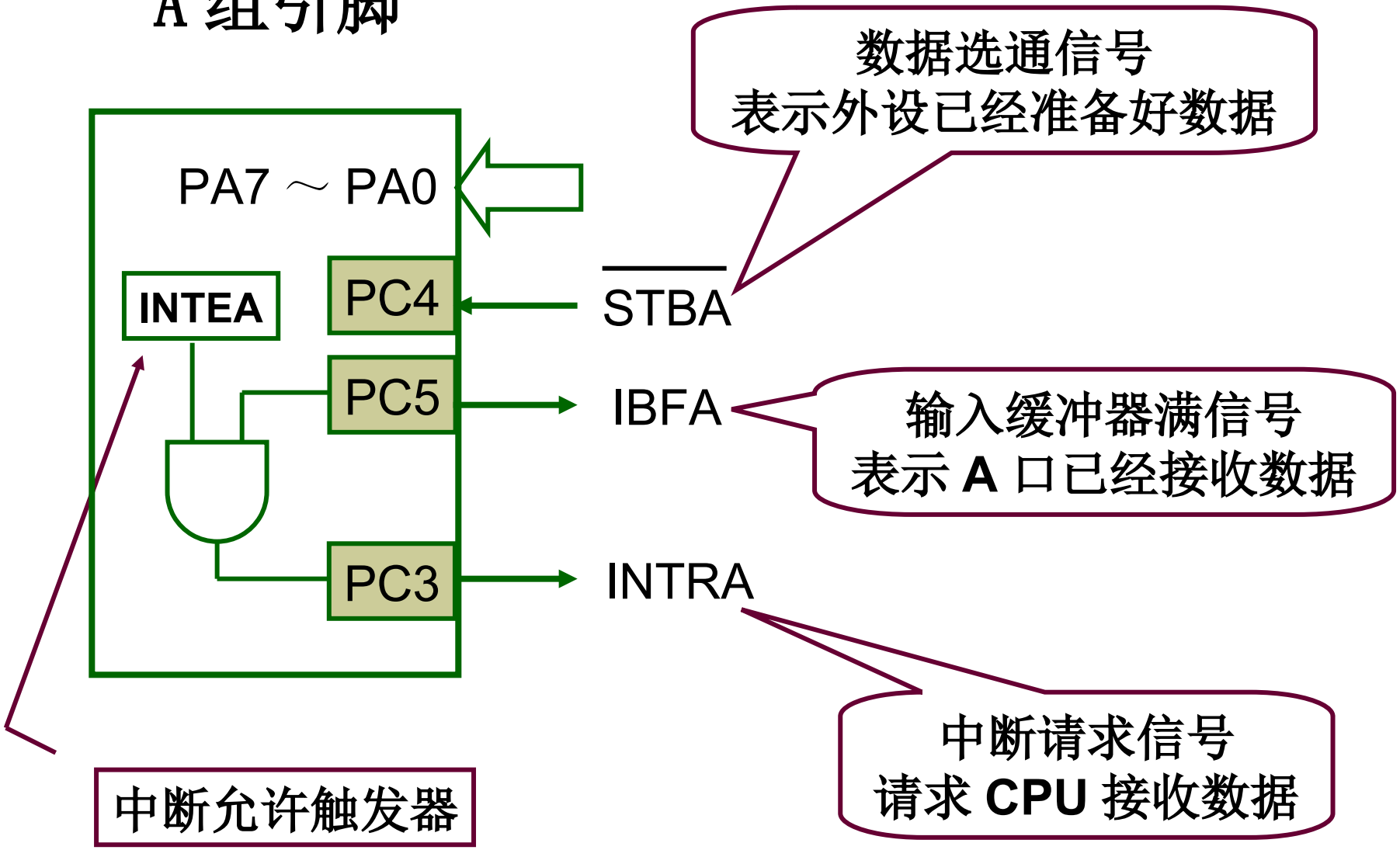


3. 工作方式1：选通输入输出方式

- 借助于选通（应答）联络信号进行输入或输出
- 只有端口 A 和端口 B 可以采用方式 1
- 作为输入或输出的数据端口
- 利用端口 C 的 3 个引脚作为应答联络信号
- 还提供有中断请求逻辑和中断允许触发器
- 对输入和输出的数据都进行锁存
- 适用于查询和中断方式的接口电路

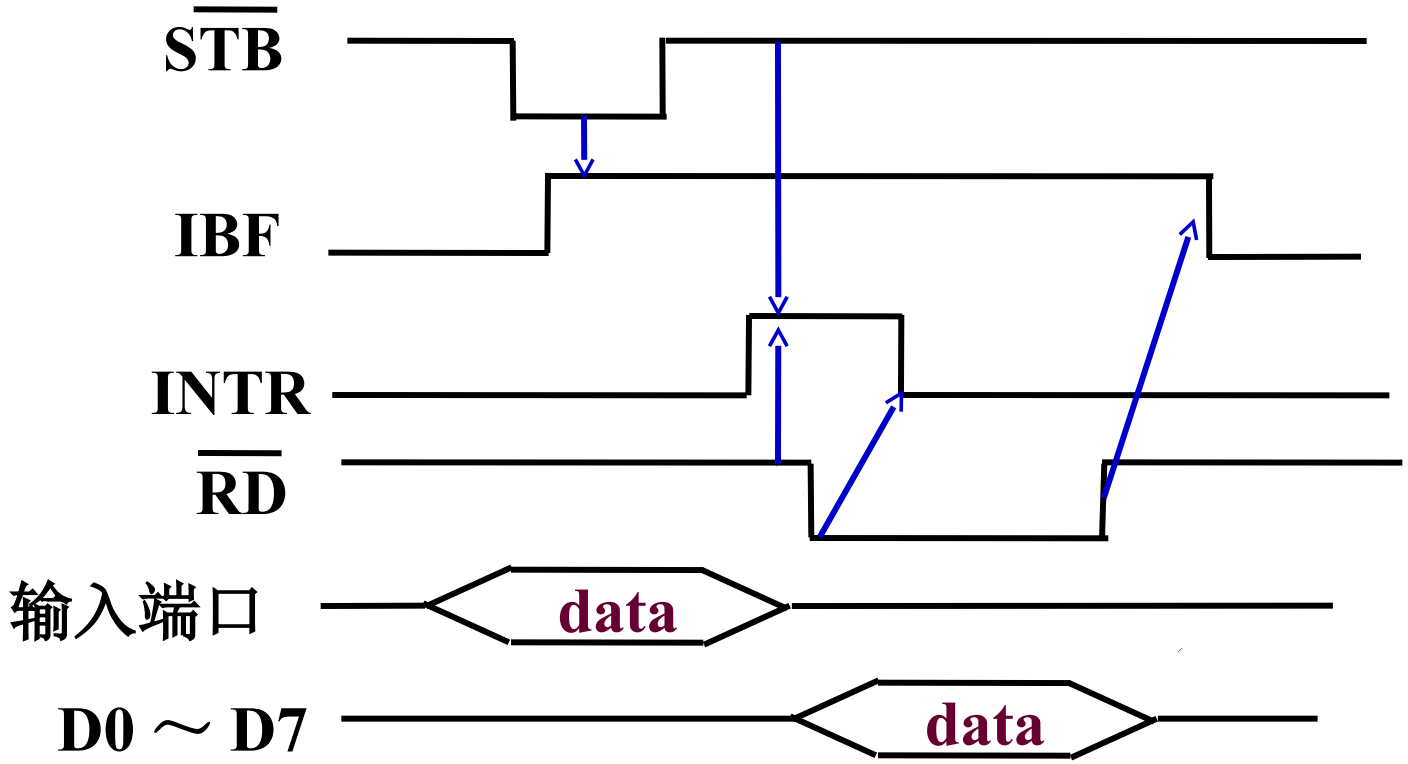
8255 工作方式 1 输入引脚

A 组引脚



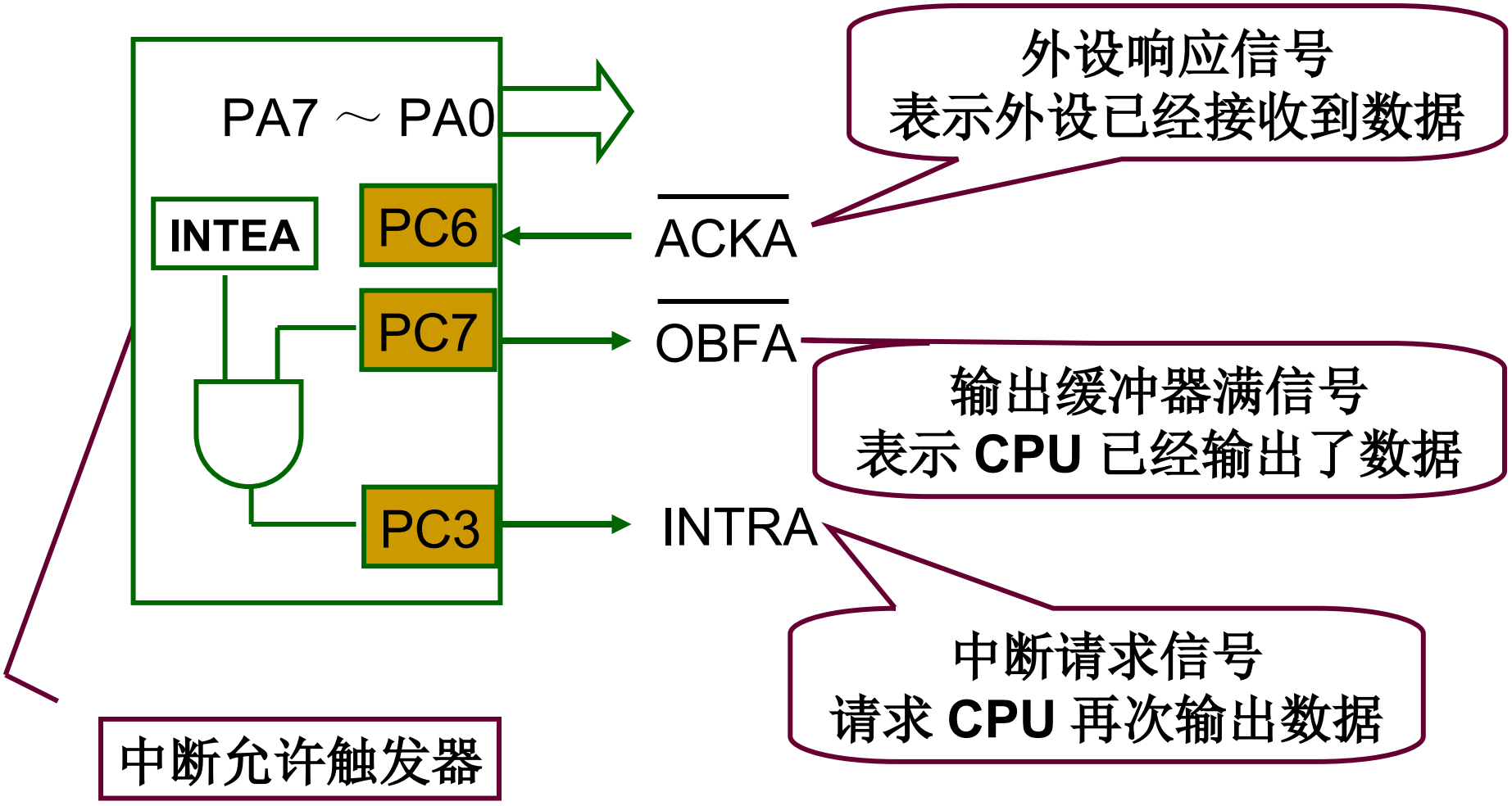
选通输入工作时序

- 异步时序：没有时钟，由引脚控制信号定时
- STB* 和 IBF 是外设和 8255A 间应答联络信号



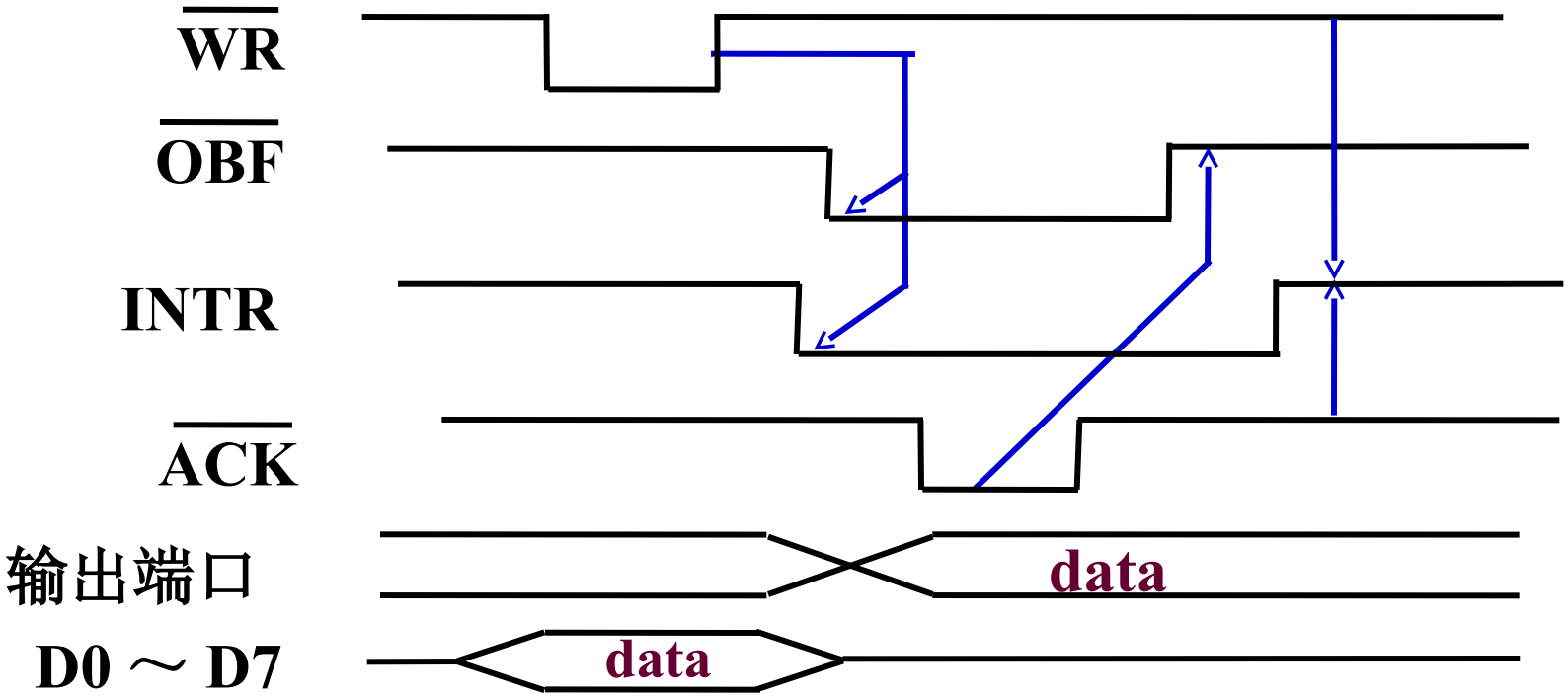
8255 工作方式 1 输出引脚

A 组引脚



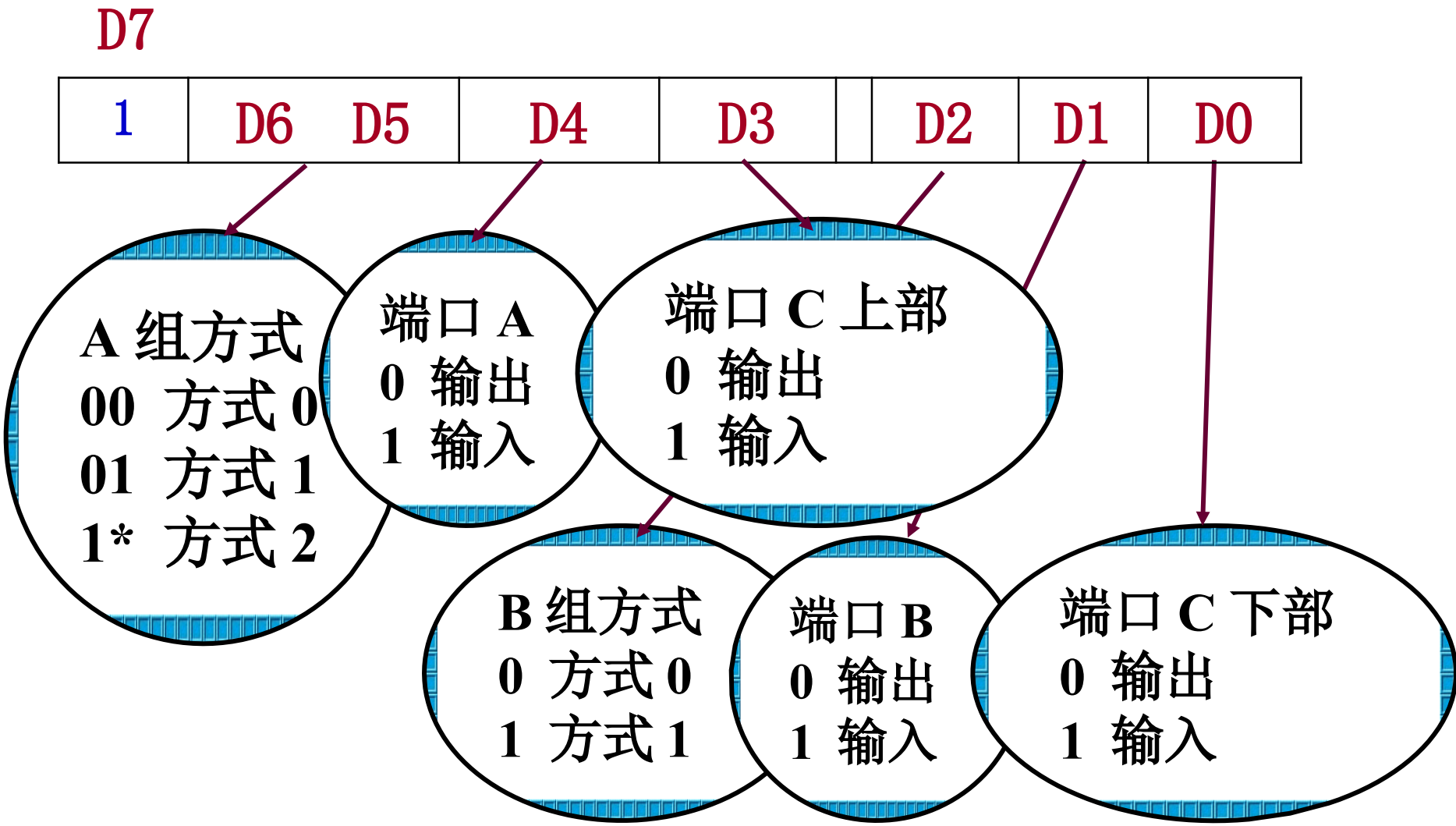
选通输出工作时序

- 异步时序：没有时钟，由引脚控制信号定时
- OBF* 和 ACK* 是外设和 8255A 间应答联络信号



4．8255 的编程：写入方式控制字

- 控制字写入控制字 I/O 地址： A1A0 ＝ 11



写入方式控制字示例

- 要求：
  - A 端口：方式 1 输入
  - C 端口上半部：输出， C 口下半部：输入
  - B 端口：方式 0 输出
- 方式控制字： 10110001B 或 B1H
- 初始化的程序段：

```
mov  dx,0fffeh          ；假设控制端口为FFFEH
mov al,0b1h             ；方式控制字
out dx,al               ；送到控制端口
```

4．8255 的编程：读写数据端口

- 利用数据端口 I/O 地址： A1A0 ＝ 00(A) 01(B) 10(C)
  - 作输入接口，执行输入 IN 指令获取外设数据
  - 作输出接口，执行输出 OUT 指令将数据送出
- 8255A 具有锁存输出数据的能力
  - 对输出方式的端口同样可以输入
  - 不是读取外设数据，而是上次给外设的数据
  - 可实现按位输出控制
- 对输出端口 B 的 PB7 位置位的程序段

```
mov  dx,0fffah          ;B 端口假设为 FFFAH
in  al,dx               ; 读出 B 端口原输出内容
or  al,80h              ; 使 PB7 ＝ D7 ＝ 1
out  dx,al              ; 输出新的内容
```

端口 C 的特点

- C 端口被分成两个 4 位端口
  - 只能以方式 0 工作，可分别选择输入或输出
  - 上半部和 A 端口编为 A 组
  - 下半部和 B 端口编为 B 组
- A 和 B 端口在方式 1 或方式 2 时
  - C 端口的部分或全部引脚将被征用
  - 其余引脚工作在方式 0

端口 C 的输出

- 通过端口 C 的 I/O 地址
  - 向 C 端口直接写入字节数据
  - 写进 C 端口的输出锁存器，并从输出引脚输出
  - 对设置为输入的引脚无效
- 通过控制端口的 I/O 地址
  - 向 C 端口写入位控字
  - 使 C 端口的某个引脚输出 1 或 0
  - 或置位复位内部的中断允许触发器

示意图

端口 C 的输入

- 未被 A 和 B 端口征用的引脚
  - 定义为输入的端口读到引脚输入的信息
  - 定义为输出的端口读到输出锁存器的信息
- 被 A 和 B 端口征用作为联络线的引脚
  - 读到反映 8255A 状态的状态字

示意图

11.2.2 并行接口的应用

- 端口 A， B， C 和控制字地址

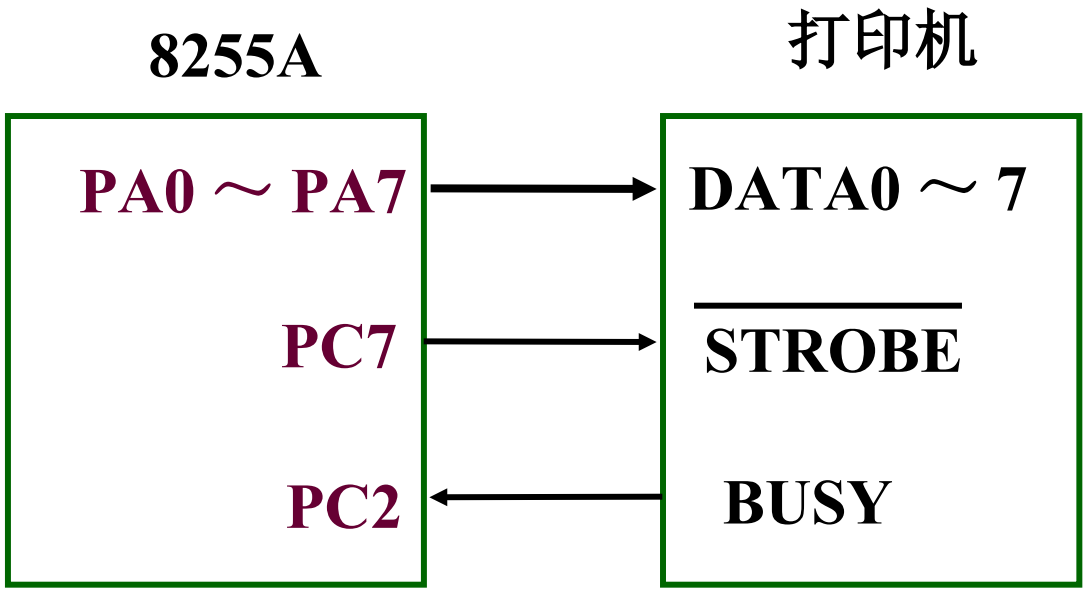
  60H， 61H， 62H 和 63H

- 工作在基本输入 / 输出方式 0
  - 端口 A 为方式 0 输入，用来读取键盘扫描码
  - 端口 B 工作于方式 0 输出，例如控制扬声器等
  - 端口 C 为方式 0 输入，读取系统状态和配置
- 系统的初始化编程：

```
mov al,10011001b   ；方式控制字 99H
out 63h,al
```

1. 用 8255 方式 0 与打印机接口

- 端口 A 为方式 0 输出打印数据
- PC7 引脚产生负脉冲选通信号
- PC2 引脚连接忙信号查询其状态
- 微处理器利用查询方式输出数据



打印机接口时序

- 典型的异步时序
- DATA0 ～ DATA7 （8 位并行数据）信号
  - 主机输出打印数据和命令
- STROBE* （选通）信号
  - 输出低有效，才能使打印机接收数据
- ACK* （响应）信号
  - 打印机接收数据结束回送负脉冲响应信号
- BUSY （忙状态）信号
  - 打印机忙于处理接收的数据，不接收新的数据

示意图

方式 0 初始化程序段

```
mov dx,0fffeh          ;控制端口地址为 FFFEH
mov al,10000001b       ;方式控制字
out dx,al
;A 端口方式 0 输出，端口 B 任意
;C 端口上半部输出、下半部输入
mov al,00001111b       ;端口 C 复位置位控制字
out dx,al
;使 PC7 ＝ 1，即置 STORE* ＝ 1
```

打印数据子程序 - 1

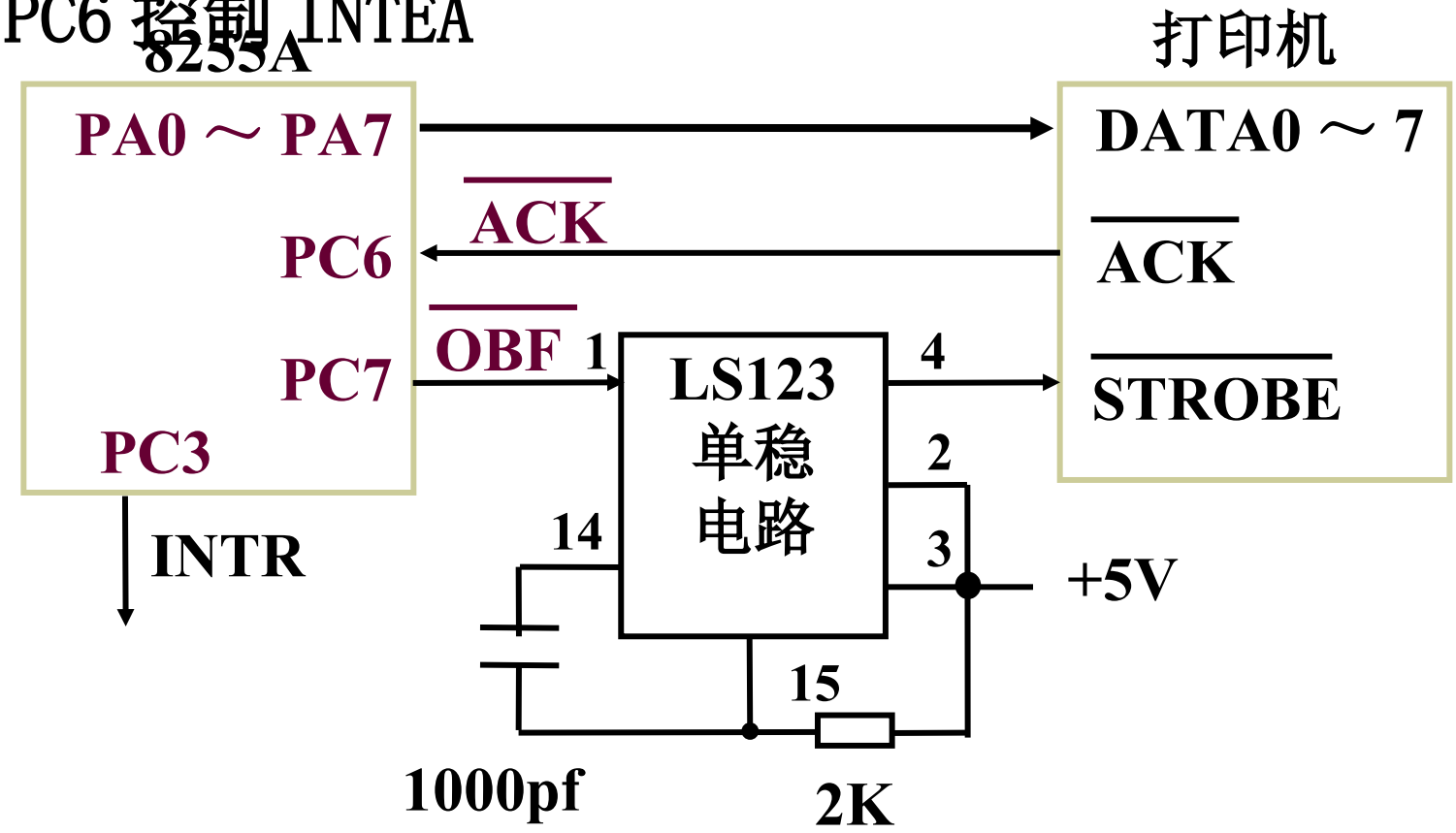
```
printc   proc              ;AH ＝打印数据
         push ax
         push dx
prn:     mov dx,0fffch     ; 读取端口 C
         in al,dx          ; 查询打印机的状态
         and  al,04h        ; 忙 否 （ PC2 ＝ BUSY ＝ 0 ）？
         jnz prn           ;PC2 ＝ 1 ，打印机忙，等待
         mov dx,0fff8h
         ;PC2 ＝ 0 ，打印机不忙，输出
         mov al,ah
         out dx,al         ; 将打印数据从端口 A 输出
```

打印数据子程序 - 2

```
        mov  dx,0fffeh  ;从PC7送出控制低脉冲
        mov  al,00001110b  ;使PC7 = STROBE* = 0
        out  dx,al
        nop             ;产生一定宽度的低电平
        nop
        mov  al,00001111b  ;使PC7 = STROBE* = 1
        out  dx,al      ;产生低脉冲STROBE*信号
        pop  dx
        pop  ax
        ret
printc  endp
```

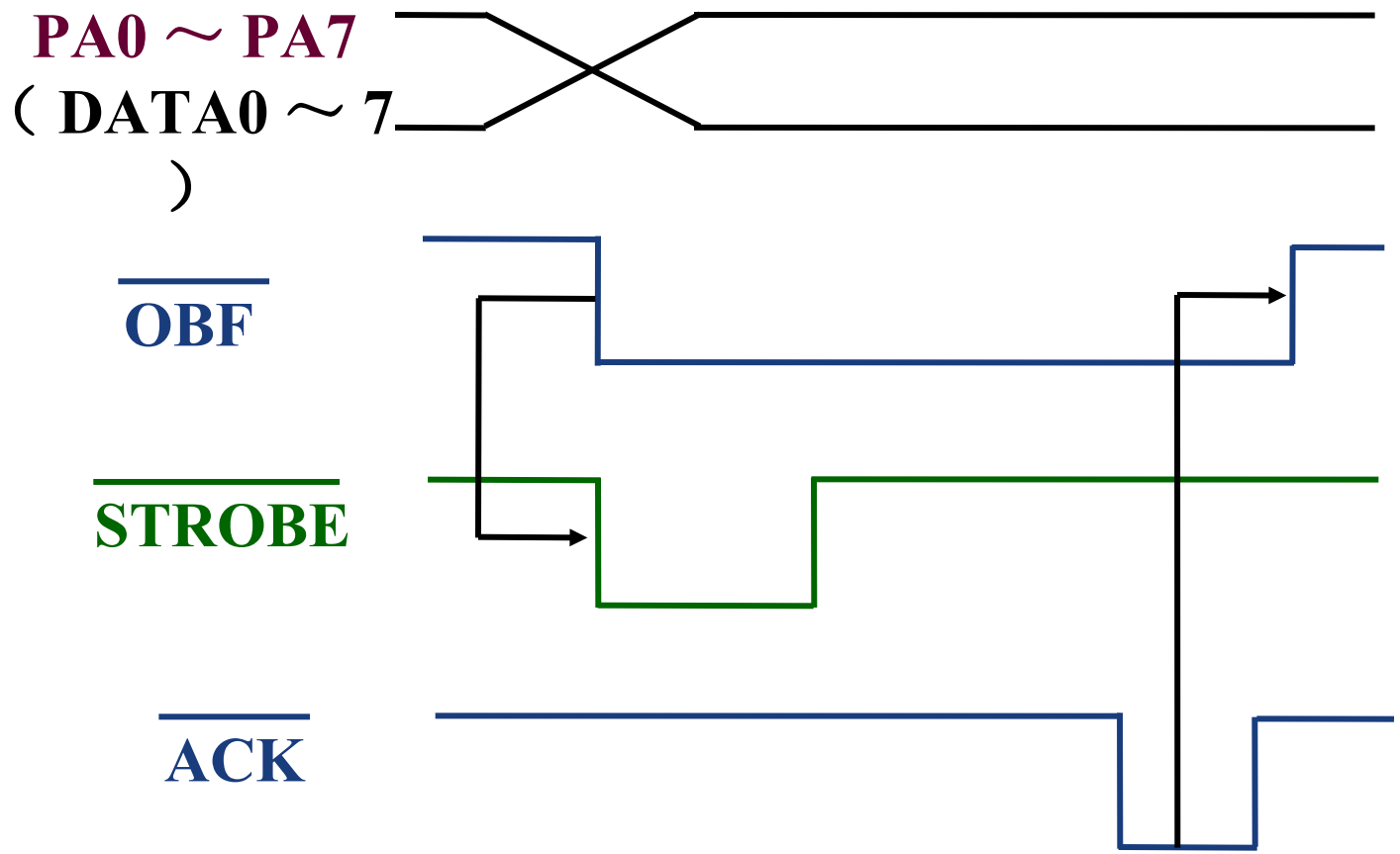
2. 用 8255 方式 1 与打印机接口

- 端口 A 选通输出连接打印机
- PC7 ＝ OBF* 输出， PC6 ＝ ACK* 输入， PC3 ＝ INTR 输出
- 通过 PC6 控制 INTEA



方式 1 时序配合

- 8255 的 OBF* 引脚对应打印机 STROBE* 引脚
- 略有差别，不能直接连接



方式 1 初始化程序段

```
mov dx,0fffeh          ;设定端口 A 为选通输出方式
mov al,0a0h
out dx,al
mov al,0ch
;使 INTEA （ PC6 ）为 0 ， 禁止中断
out dx,al
……
mov cx,counter         ;打印字节数送 CX
mov bx,offset buffer   ;取字符串首地址送 BX
call prints            ;调用打印子程序
```

打印字符串子程序 - 1

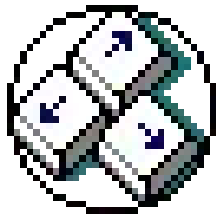
```
        ;入口参数： DS:BX =字符串首地址
        ;CX =字符个数
prints  proc
        push ax             ;保护寄存器
        push dx
print1: mov al,[bx]         ;取一个数据
        mov dx,0fff8h
        out dx,al           ;从端口 A 输出
```

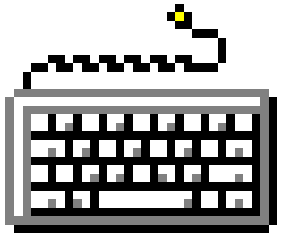
打印字符串子程序 - 2

```
        mov dx,0fffch
print2: in al,dx          ;读取端口C
        test al,80h       ;检测OBF*（PC7）为1否？
        jz print2
        ;为0，说明打印机没有响应，继续检测
        inc bx
        ;为1，说明打印机已接受数据
        loop print1       ;准备取下一个数据输出
        pop dx            ;打印结束，恢复寄存器
        pop ax
        ret               ;返回
prints  endp
```

11.2.3 键盘及其接口

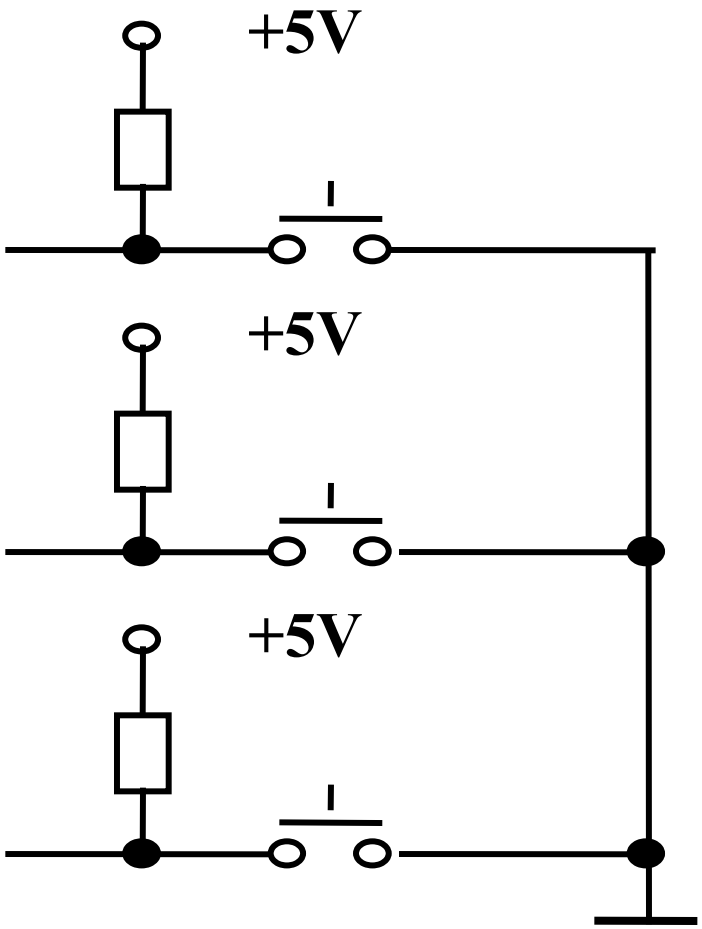
- 键盘是微机系统最常使用的输入设备
- 小键盘：适用于单板机或以处理器为基础的仪器，实现数据、地址、命令及指令等输入
- 独立键盘：通过 5 芯电缆与 PC 微机主机连接



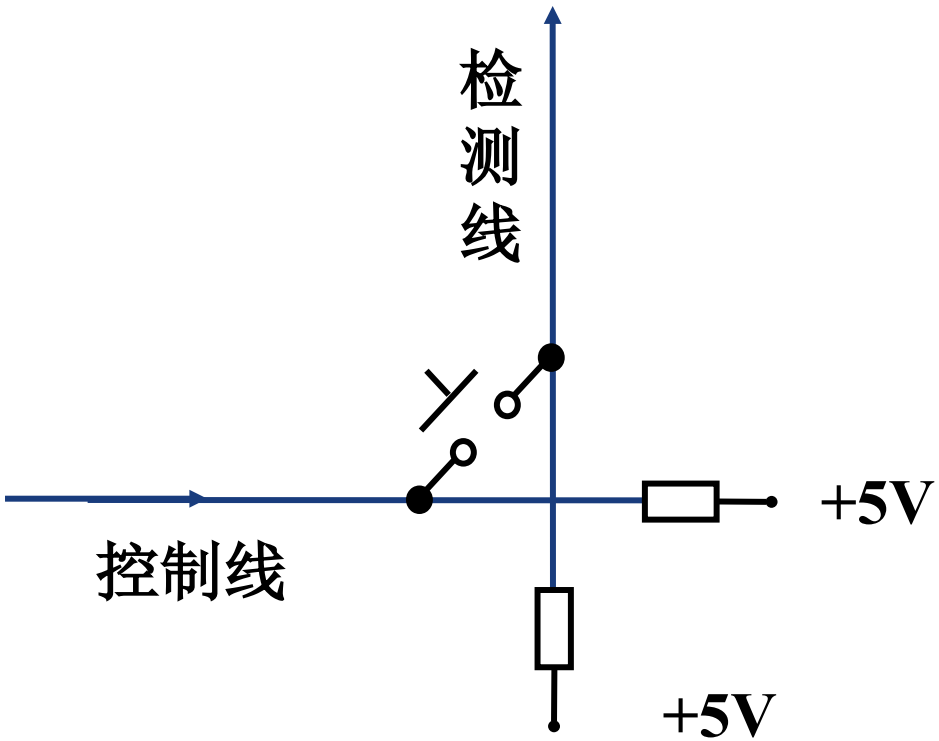


1． 简易键盘

- 线性结构键盘



- 矩阵结构键盘



识别按键的扫描方法

- 先使第 0 行接低电平，其余行为高电平，然后看第 0 行是否有键闭合（通过检查列线电位实现）
- 再将第 1 行接地，检测列线是否有变为低电位的线
- 如此往下一行一行地扫描，直到最后一行
- 扫描过程中，发现某一行有键闭合时（列线输入中有一位为 0 ），便在扫描中途退出
- 通过组合行线和列线识别此刻按下哪一键

键盘扫描程序第 1 段：判断是否有键按下

```
key1: mov al,00
      mov dx,rowport   ; rowport ＝行线端口地址
      out dx,al        ; 使所有行线为低电平
      mov dx,colport   ; colport ＝列线端口地址
      in al,dx         ; 读取列值
      cmp al,0ffh      ; 判定列线是否为低电平
      jz key1          ; 没有，无闭合键
      ; 则循环等待（或转向其他程序片断）
      call delay       ; 有，延迟 20ms 消除抖动
```

键盘扫描程序第 2 段：识别按键

```
        mov  cx,8           ;  行数送 CX
        mov  ah,0feh        ;  扫描初值送 AH
key2:   mov  al,ah
        mov  dx,rowport
        out  dx,al          ;  输出行值（扫描值）
        mov  dx,colport
        in  al,dx           ;  读进列值
        cmp  al,0ffh        ;  判断有无低电平的列线
        jnz  key3           ;  有，则转下一步处理
        rol  ah,1           ;  无，则移位扫描值
        loop  key2          ;  准备下一行扫描
        jmp  key1           ;  所有行都没有按键
key3:   ……                  ;  AL ＝列值， AH ＝行值
```

键盘扫描程序第 3 段：查找键代码 - 1

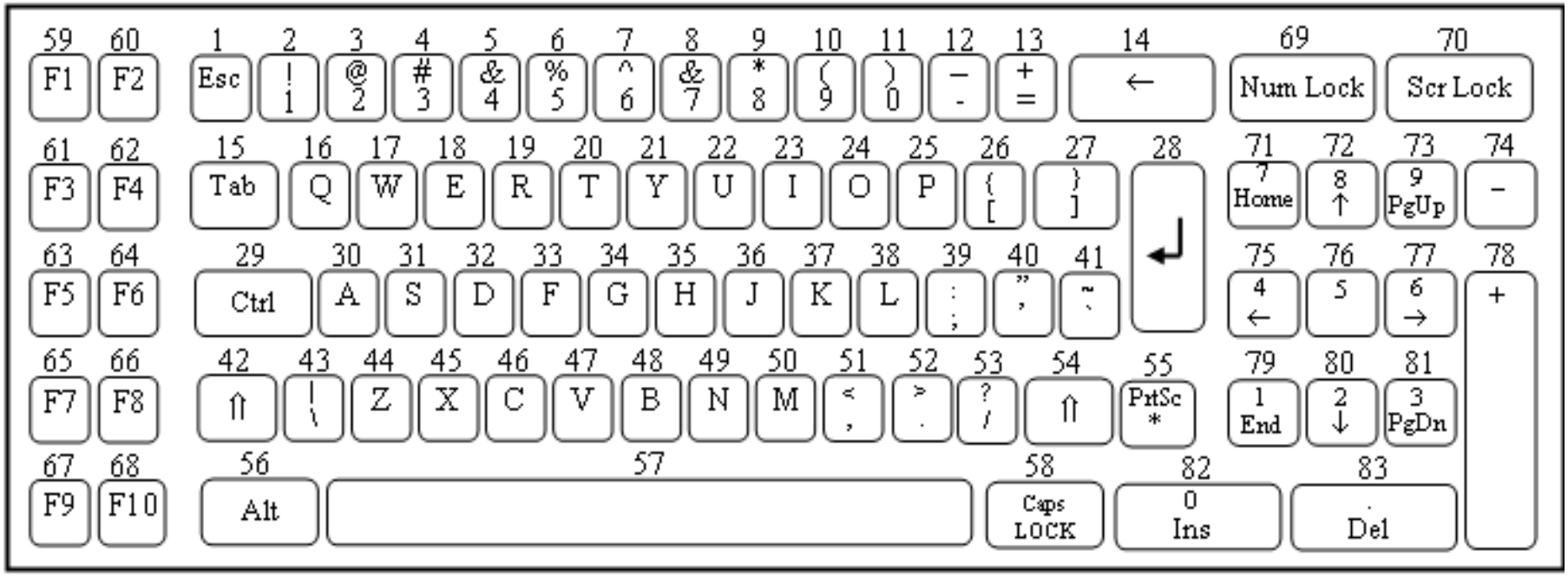
```
key3: mov si,offset table ; SI 指向键行列值表
      mov di,offset char  ; DI 指向键代码表
      mov cx,64           ; CX ＝键的个数
key4: cmp ax,[si]         ;  与按键的行列值比较
      jz key5             ;  相同，说明查到
      inc si              ;  不相同，继续比较
      inc si
      inc di
      loop key4
      jmp key1            ;  全部不相同
      ;  返回继续检测（或转向其他程序片断）
```

键盘扫描程序第 3 段：查找键代码 - 2

```
key5: mov al,[di]  ;  获取键代码送 AL
      ……           ;  判断按键释放，没有则等待
      call delay   ;  按键释放，延时消除抖动
      ……           ;  后续处理
;  键盘的行列值表：低字节是列值、高字节是行值
table word 0fefeh  ;  键 0 的行列值
      word 0fefdh  ;  键 1 的行列值
      ……           ;  其他键的行列值
      ;  键盘的键代码表
char  byte ……      ;  键 0 的代码值
      byte ……      ;  键 1 的代码值
      ……           ;  其他键的代码值
```

2．PC 机键盘

- 与主机箱分开的一个独立装置
- 通过一根五芯电缆与主机相连
- PC 及 PC/XT 机采用 83 （或 84 ）键的标准键盘



PC 机键盘的工作过程

- 键盘电路正常工作时不断地扫描键盘矩阵
- 有按键，则确定按键位置之后以串行数据形式发送给系统板键盘接口电路
- 键按下时，发送该键的接通扫描码
- 键松开时，发送该键的断开扫描码
- 若一直按住某键，则以拍发速率（每秒 2 ～ 30 次）连续发送该键的接通扫描码

接通扫描码＝键盘上的位置

断开扫描码＝接通扫描码＋ 80H

键盘接口电路的工作过程

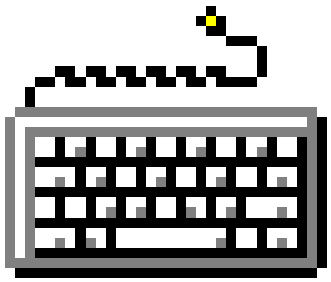
- 接收一个串行形式字符，进行串并转换
- 产生键盘中断 IRQ1 请求，等待读取键盘数据
- CPU 响应中断，进入 09H 键盘中断服务程序：
  ① 读取键盘扫描码：用 IN AL,60H 即可
  ② 响应键盘：系统使 PB7 ＝ 1
  ③ 允许键盘工作：系统使 PB7 ＝ 0
  ④ 处理键盘数据
  ⑤ 给 8259A 中断结束 EOI 命令，中断返回

〔例 11-2 〕键盘中断服务程序 - 1

- 09H 号中断服务程序（ kbint 过程）
  - 完成常规的操作
  - 处理键盘数据

  将扫描码通过查表转换为对应 ASCII 码送缓冲区

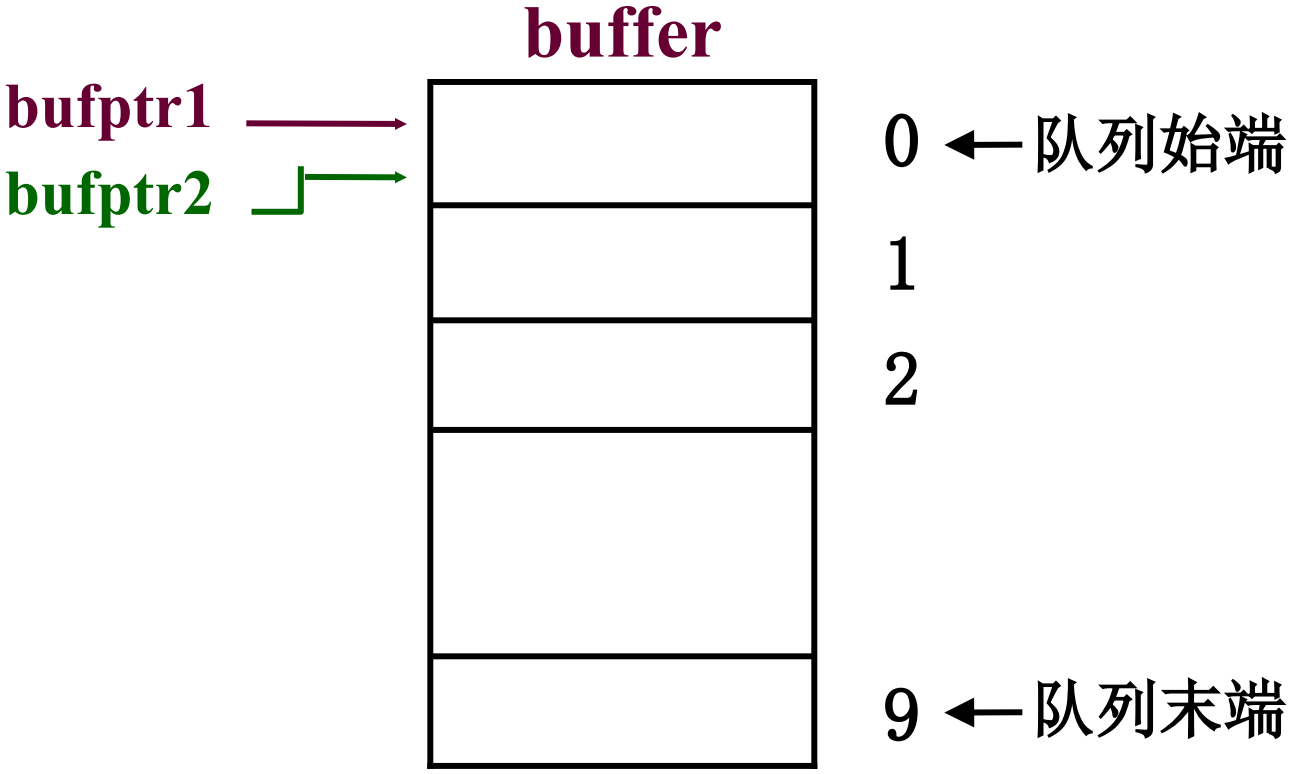
  不能显示的按键，转换为 0 ，且不再送至缓冲区
- 键盘 I/O 功能程序（ kbget 子程序）
  - 从缓冲区中读取转换后的 ASCII 码
- 功能调用（主程序）
  - 循环显示键入的字符



〔例 11-2 〕键盘中断服务程序 – 2

- 键盘缓冲区
  - 中断服务程序与子程序之间传递参数
  - 先进先出循环队列

- 队列空
- 进队列
- 出队列
- 队列满



〔例 11-2 〕键盘中断服务程序 - 3

```
        ;  数据段
buffer  byte 10 dup(0)    ;  键盘缓冲区
bufptr1 word 0            ;  队列头指针
bufptr2 word 0            ;  队列尾指针
        ;  按扫描码顺序给出字符的 ASCII 码
        ;  不能显示的按键为 0
        ;  第一个 0 不对应按键，仅用于查表指令
scantb  byte 0,1,'1234567890-=',08h
        ;  键盘第 1 排的按键，从 ESC 到退格
        ……
        byte 0,0,'789-456+1230.'
        ;  右边小键盘，从 Num Lock 到 Del
```

〔例 11-2 〕键盘中断服务程序 - 4

```
; 代码段
mov ax,3509h    ; 获取保存原中断向量表项
int 21h
push es
push bx
cli             ; 关中断
push ds         ; 设置 09H 号新中断向量表项
mov ax,seg kbint
mov ds,ax
mov dx,offset kbint
mov ax,2509h
int 21h
pop ds
```

〔例 11-2 〕键盘中断服务程序 - 5

```
        in al,21h          ;  允许 IRQ1 中断
        push ax
        and al,0fdh
        out 21h,al
        sti                ;  开中断
start1: call kbget
        ;  调用 KBGET 获取按键的 ASCII 码
        cmp al,1
        jz start2          ;  是 ESC 键，则退出
        push ax            ;  保护字符
        call dispc         ;  显示字符
        pop ax             ;  恢复字符
```

〔例 11-2 〕键盘中断服务程序 - 6

```
        cmp al,0dh      ;  该字符是回车符吗?
        jnz start1      ;  不是，取下个按键字符
        mov al,0ah      ;  是回车符，再进行换行
        call dispc
        jmp start1      ;  继续取字符
start2: cli             ;  恢复中断原状态
        pop ax
        out 21h,al
        pop dx
        pop ds
        mov ax,2509h
        int 21h
        sti
```

〔例 11-2 〕键盘中断服务程序 - 7

```
        ; KBGET 子程序从缓冲区取字符送 AL
kbget   proc
        push bx             ;  保护 BX
kbget1: cli                 ;  关中断
        mov bx,bufptr1      ;  取缓冲区队列头指针
        cmp bx,bufptr2      ;  与尾指针相等否?
        jnz kbget2          ;  不相等，有字符
        sti                 ;  相等，缓冲区空
        jmp kbget1          ;  等待缓冲区有字符
```

〔例 11-2 〕键盘中断服务程序 - 8

```
kbget2: mov al,buffer[bx]
        ; 从队列头取字符送 AL
        inc bx              ; 队列头指针增量
        cmp bx,10           ; 指针指向队列末端?
        jc kbget3           ; 没有，转移
        mov bx,0            ; 循环指向始端
kbget3: mov bufptr1,bx      ; 设定新队列头指针
        sti                 ; 开中断
        pop bx              ; 恢复 BX
        ret                 ; 子程序返回
kbget   endp
```

〔例 11-2 〕键盘中断服务程序 - 9

```
        ; KBINT 中断服务程序处理 09H 号键盘中断
kbint   proc
        sti                 ; 开中断
        push ax             ; 保护寄存器
        push bx
        in al,60h           ; 读取键盘扫描码
        mov bl,al           ; 扫描码保存在 BL
        in al,61h           ; 使 PB7 = 1，响应键盘
        or al,80h
        out 61h,al
        and al,7fh          ; 使 PB7 = 0，允许键盘
        out 61h,al
```

〔例 11-2 〕键盘中断服务程序 - 10

```
test bl,80h              ;  键盘数据处理
jnz kbint2               ;  是断开扫描码，退出
xor bh,bh
mov al,scantb[bx]        ;  是接通扫描码，转换
cmp al,0                 ;  合法的 ASCII 码?
jz kbint2                ;  不是，退出
mov bx,bufptr2           ;  是，取队列尾指针
mov buffer[bx],al        ;  存入缓冲区队列尾
inc bx                   ;  队列尾指针增量
cmp bx,10                ;  指针指向队列末端?
jc kbint1                ;  没有，转移
mov bx,0                 ;  循环指向始端
```

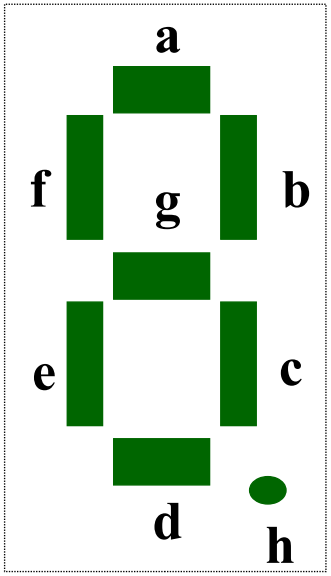
〔例 11-2 〕键盘中断服务程序 - 11

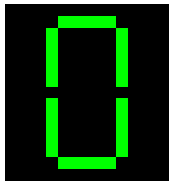
```
kbint1: cmp bx,bufptr1      ;  缓冲区是否已满?
        jz kbint2           ;  队列满，退出
        mov bufptr2,bx
        ;  队列不满，设置新的队列尾指针
kbint2: mov al,20h          ;  普通中断结束命令
        out 20h,al
        pop bx              ;  恢复寄存器
        pop ax
        iret                ;  中断返回
kbint   endp
```

示意图

11.2.4 数码管及其接口

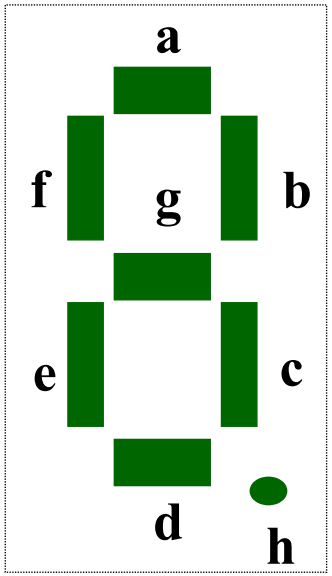
- 发光二极管 LED 是最简单的显示设备
- 由 7 段 LED 就可以组成的 LED 数码管
- LED 数码管广泛用于单板微型机、微型机控制系统及数字化仪器中
- LED 数码管可以显示内存地址和数据等





1．LED 数码管的工作原理

- 主要部分是 7 段发光管
- 顺时针分别称为 a、 b、 c、 d、 e、 f、 g
- 有的产品还附带有一个小数点 h
- 通过 7 个发光段的不同组合
  - 主要显示 0 ～ 9
  - 也可显示 A ～ F （ 16 进制数）
  - 还可显示个别特殊字符： －、 P
- 共阳极结构
  - 共用阳极接高电平
- 共阴极结构
  - 共用阴极接低电平



示意图

2． 单个数码管的显示

- LED 数码管显示一位十六进制数（ 4 位二进制数）
  - 硬件方法：专用的带驱动器的 LED 段译码器
  - 软件方法：组成显示代码表，通过查表进行译码

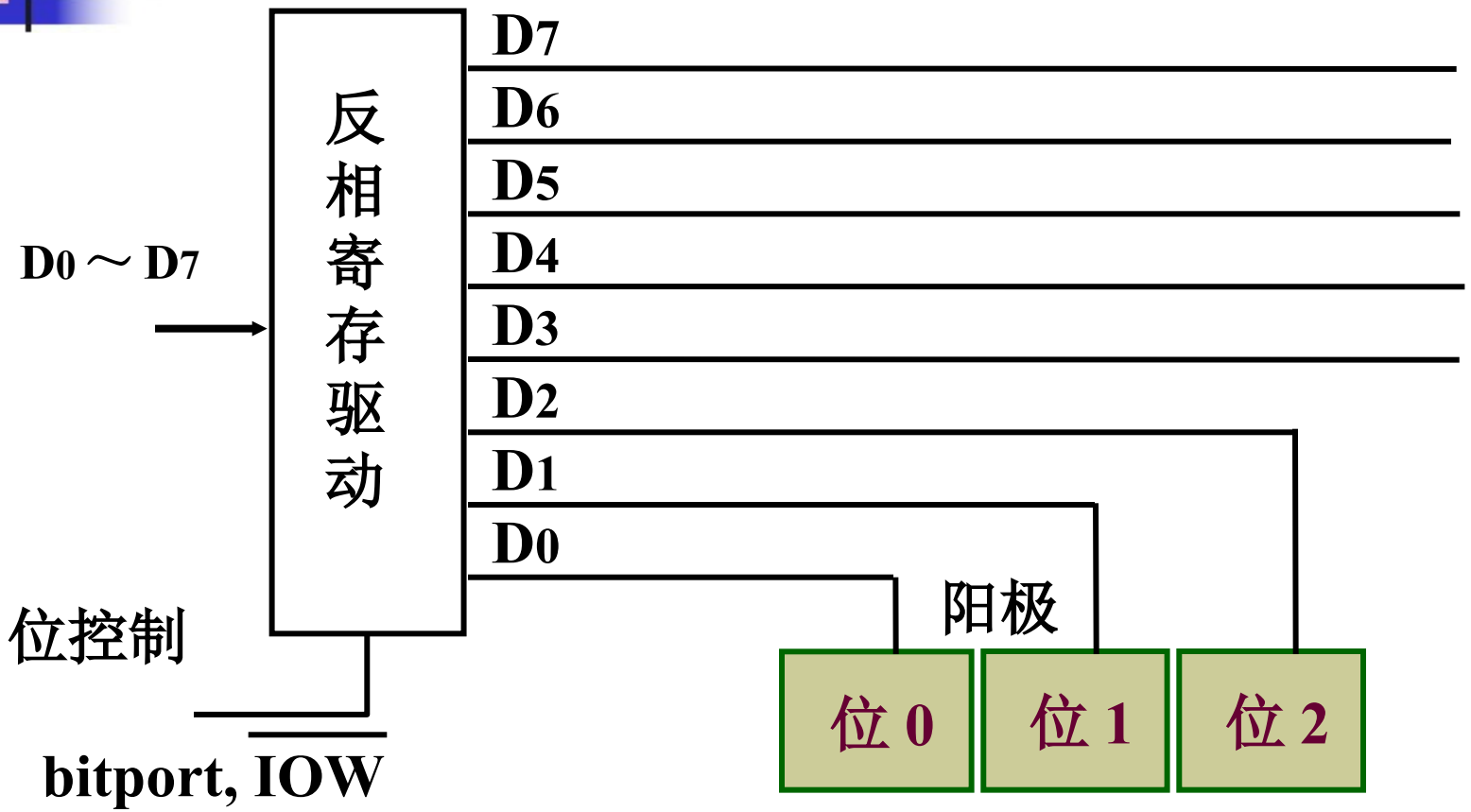
```
ledtb   byte 3fh,06h,5bh,……   ；显示代码表
        ；  实现 1 个 LED 数码管显示
        mov bx,1              ；BX← 要显示的数字
        mov al,ledtb[bx]
        ；  换码为显示代码：  AL←LEBTB[BX]
        mov dx,port           ；port ＝数码管端口地址
        out dx,al             ；  输出显示
```

示意图

3． 多个数码管的显示

- 硬件上用公用的驱动电路来驱动各数码管
- 软件上用扫描方法实现数码显示
- 8 个数码管：用 2 个 8 位输出端口控制
  - 位控制端口：控制哪个（位）数码管显示
  - 段控制端口：控制哪个段显示，决定具体显示什么数码
- 稳定数字显示：依次显示，不断重复
  - 重复频率越高，数字显示越稳定
  - 延时显示时间越长，显示亮度就越高
- 各种显示效果：控制重复频率和延时时间

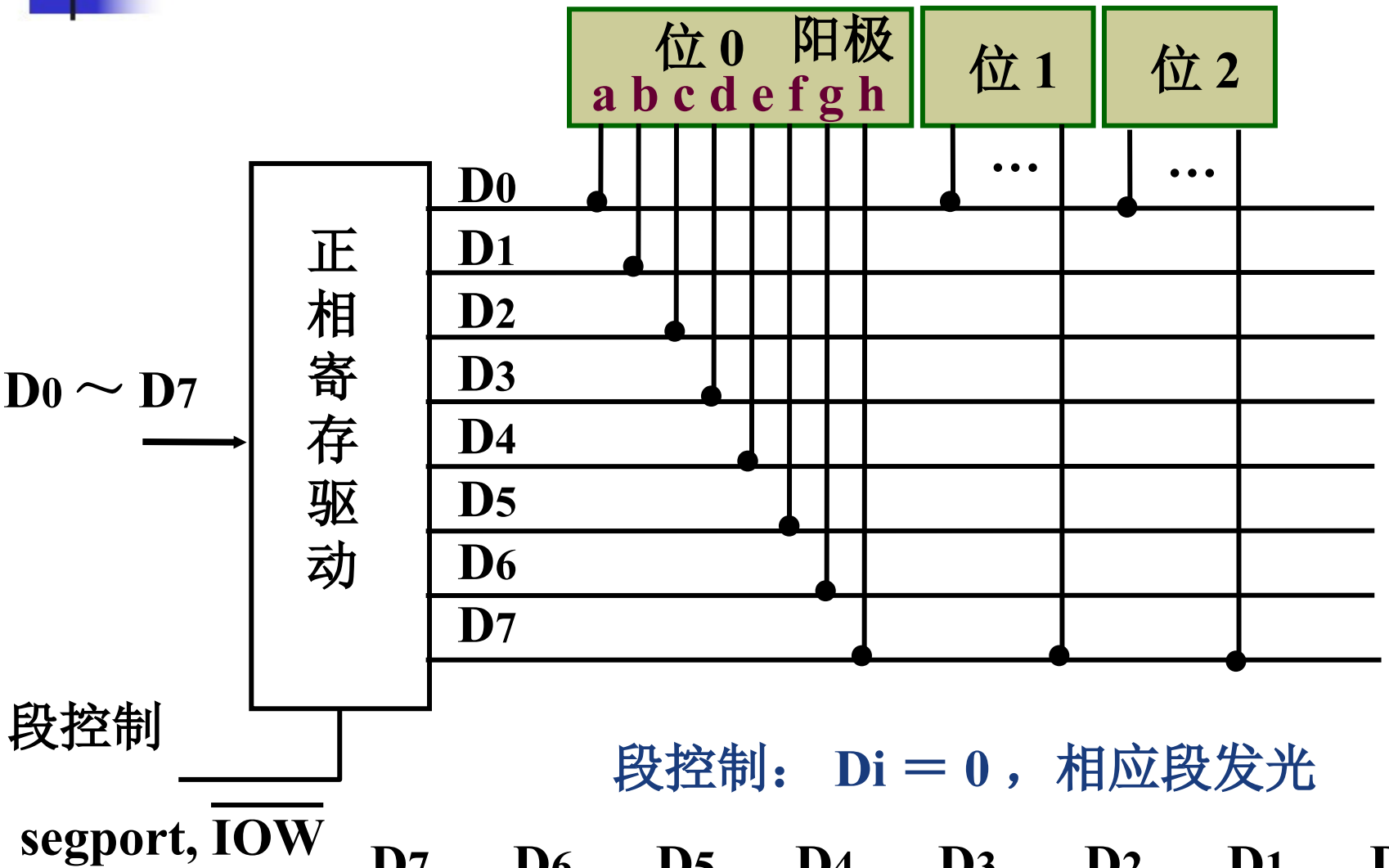
位控制端口



位控制： Di ＝ 0 ， 相应位发光

D7	D6	D5	D4	D3	D2	D1	D0
位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0

段控制端口



段控制： Di ＝ 0 ， 相应段发光

D7	D6	D5	D4	D3	D2	D1	D0
h	g	f	e	d	c	b	a

依次显示 8 位数码管程序 - 1

```
        ; 数据段
leddt   byte  8 dup(0)          ; 数码缓冲区
        ; 主程序
        mov si,offset leddt     ; 指向数码缓冲区
        call displed            ; 调用显示子程序
        ; 子程序：显示一次数码缓冲区的 8 个数码
        ; 入口参数： DS:SI ＝缓冲区首地址
displed proc
        push ax
        push bx
        push dx
```

依次显示 8 位数码管程序 - 2

```
        xor bx,bx
        mov ah,0feh          ; 指向最左边数码管
led1:   mov bl,[si]          ; 取出要显示的数字
        inc si
        mov al,ledtb[bx]
        ; 得到显示代码：AL←LEDTB[BX]
        mov dx,segport       ; segport 为段控制端口
        out dx,al            ; 送出段码
        mov al,ah            ; 取出位显示代码
        mov dx,bitport       ; bitport 为位控制端口
        out dx,al            ; 送出位码
```

依次显示 8 位数码管程序 - 3

```
        call delay      ;  实现数码管延时显示
        rol ah,1        ;  指向下一个数码管
        cmp ah,0feh     ;  是否指向最右边数码管
        jnz led1        ;  没有，显示下一个数字
        pop dx
        pop bx
        pop ax
        ret             ;  8 位数码管都显示一遍
        ;  显示代码表，按照 0 ～ 9 、 A ～ F 的顺序
ledtb   byte 0c0h,0f9h,0a4h,……,86h,8eh
displed endp
```

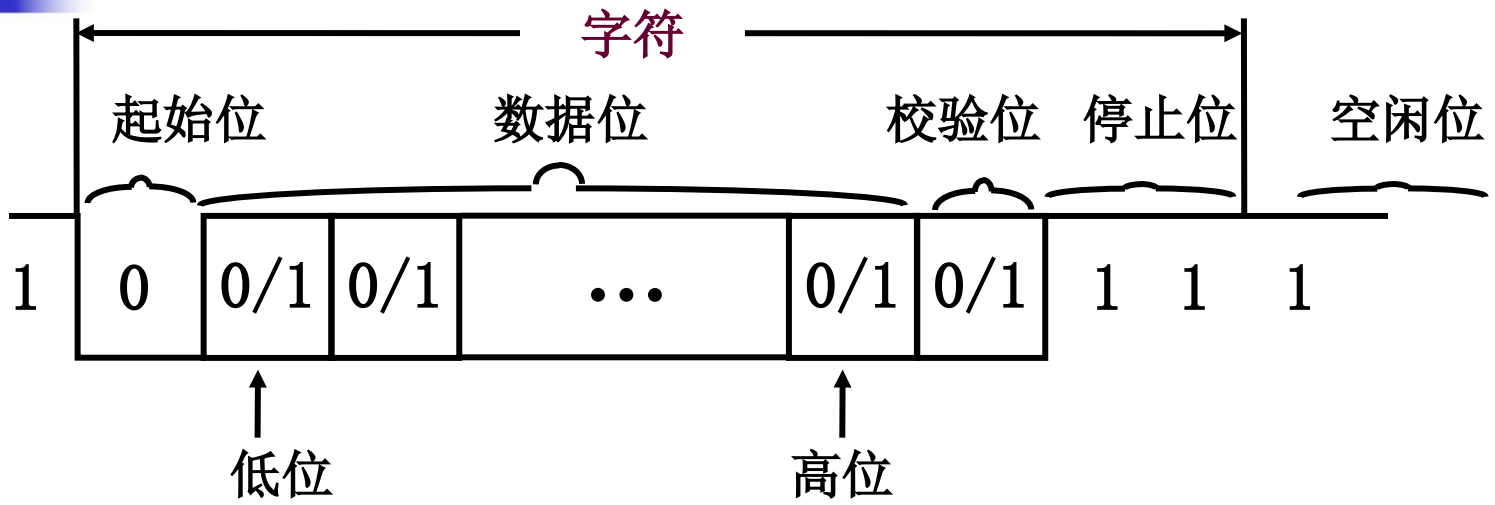
依次显示 8 位数码管程序 - 4

```
timer     = 10              ;  延时常量
delay     proc              ;  软件延时子程序
          push bx
          push cx
          mov bx,timer      ;  外循环：timer 次数
delay1:   xor cx,cx
delay2:   loop delay2       ;  内循环：2^16 次循环
          dec bx
          jnz delay1
          pop cx
          pop bx
          ret
delay     endp
```

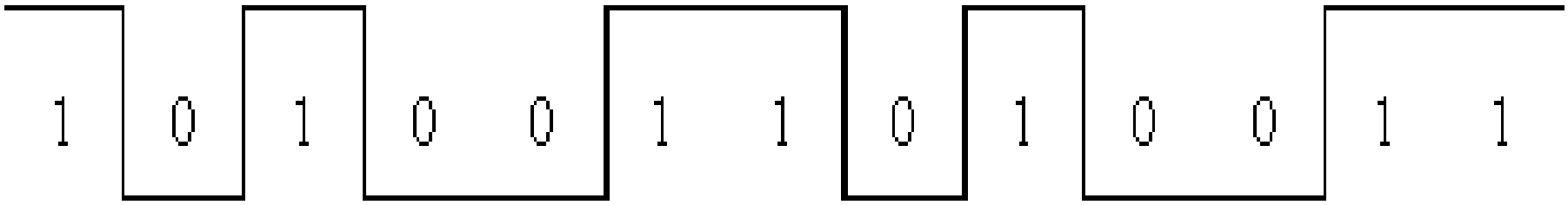
8.3 异步串行通信接口

- 串行通信：将数据分解成二进制位用一条信号线，一位一位顺序传送的方式
- 串行通信的优势：用于通信的线路少，因而在远距离通信时可以极大地降低成本
- 串行通信适合于远距离数据传送，也常用于速度要求不高的近距离数据传送
- 通信协议（通信规程）：收发双方共同遵守，解决传送速率、信息格式、位同步、字符同步、数据校验等问题
- 串行异步通信：以字符为单位进行传输
- 串行同步通信：以一个数据块（帧）为传输单位

8.3.1 异步串行通信格式



- 空闲位——传送字符之间的逻辑 1 电平，表示没有进行传送

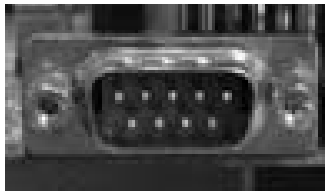


数据传输速率

- 数据传输速率＝比特率（ Bit Rate ）
  - 每秒传输的二进制位数 bps
  - 字符中每个二进制位持续的时间长度都一样，为数据传输速率的倒数
- 进行二进制数码传输，每位时间长度相等：比特率＝波特率（ Baud Rate ）
- 过去，限制在 50 bps 到 9600 bps 之间
- 现在，可以达到 115200 bps 或更高

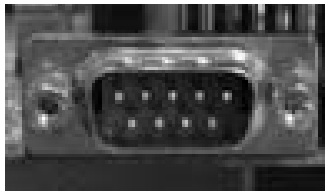
8.3.2 异步串行接口标准

- 美国电子工业协会 EIA 制定
  - 1962 年公布， 1969 年修订
  - 1987 年 1 月正式改名为 EIA-232D
  - 数据终端设备 DTE 与数据通信设备 DCE 标准接口



- 调制解调器 Modem
  - 通信线路信号与数字信号相互转换的设备
- 传输制式
  - 全双工：双根传输线，能够同时发送和接收
  - 半双工：单根传输线，不能同时发送和接收
  - 单工：单根传输线只用作发送或只用作接收

1．232C 的引脚定义



TxD： 发送数据

RxD： 接收数据

RTS： 请求发送

CTS： 清除发送（允许发送）

DTR： 数据终端准备好

DSR： 数据装置准备好

GND： 信号地

CD： 载波检测（DCD）

RI： 振铃指示

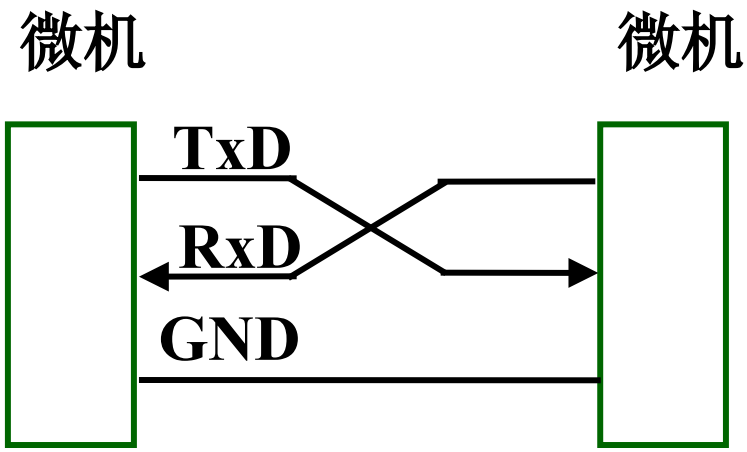
保护地（机壳地）

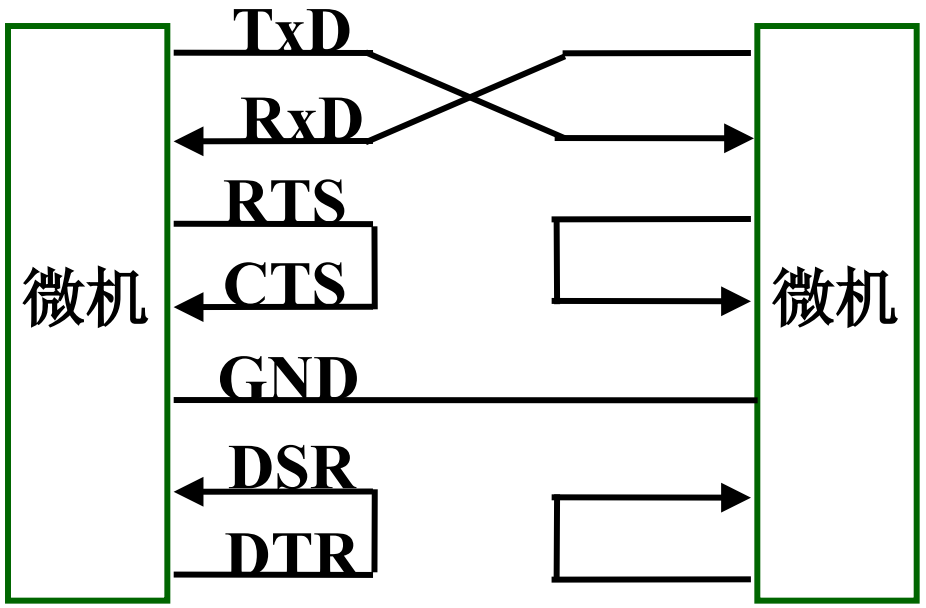
TxC： 发送器时钟

RxC： 接收器时钟

2．232C 的连接

- 连接调制解调器：通过电话线路远距离通信
- 直接 (零调制解调器) 连接：进行短距离通信
  - 不使用联络信号的 3 线相连
  - “ 伪”使用联络信号的 3 线相连

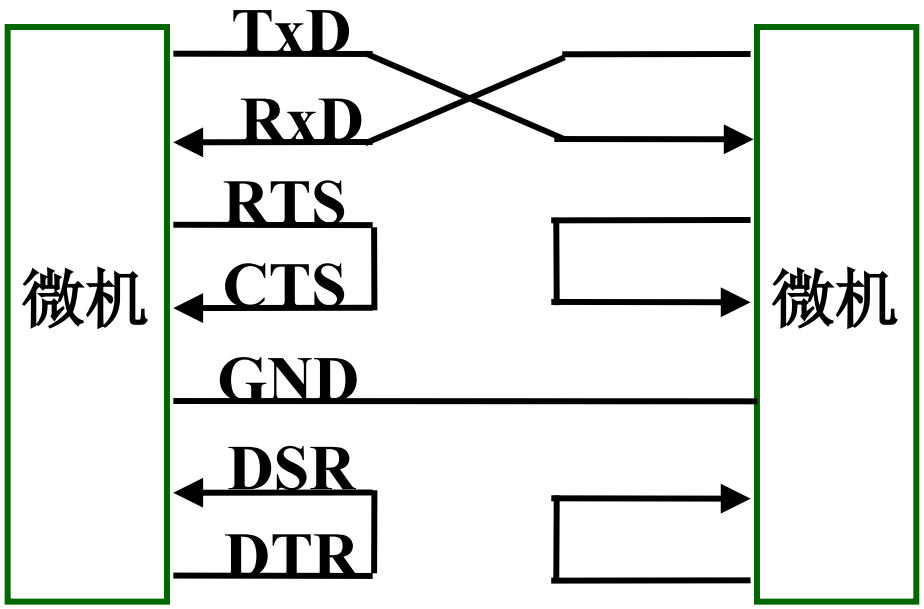




2．232C 的连接

- 连接调制解调器：通过电话线路远距离通信
- 直接 (零调制解调器) 连接：进行短距离通信

使用联络信号的多线相连



11.3.3 异步串行通信程序

- IBM PC/XT 机的 UART 芯片是 INS 8250
- 后续 PC 机采用兼容的 NS16450 和 NS16550
- 现在 32 位 PC 机芯片组兼容 NS16550
- 实现起止式串行异步通信协议
  - 数据位为 5 ～ 8 位，停止位 1 、 1.5 或 2 位
  - 奇偶校验，具有奇偶、帧和溢出错误检测电路
- 支持全双工通信
  - 8250 支持的数据传输速率为 50 ～ 9600 bps
  - 16550 支持的速率高达 115200 bps

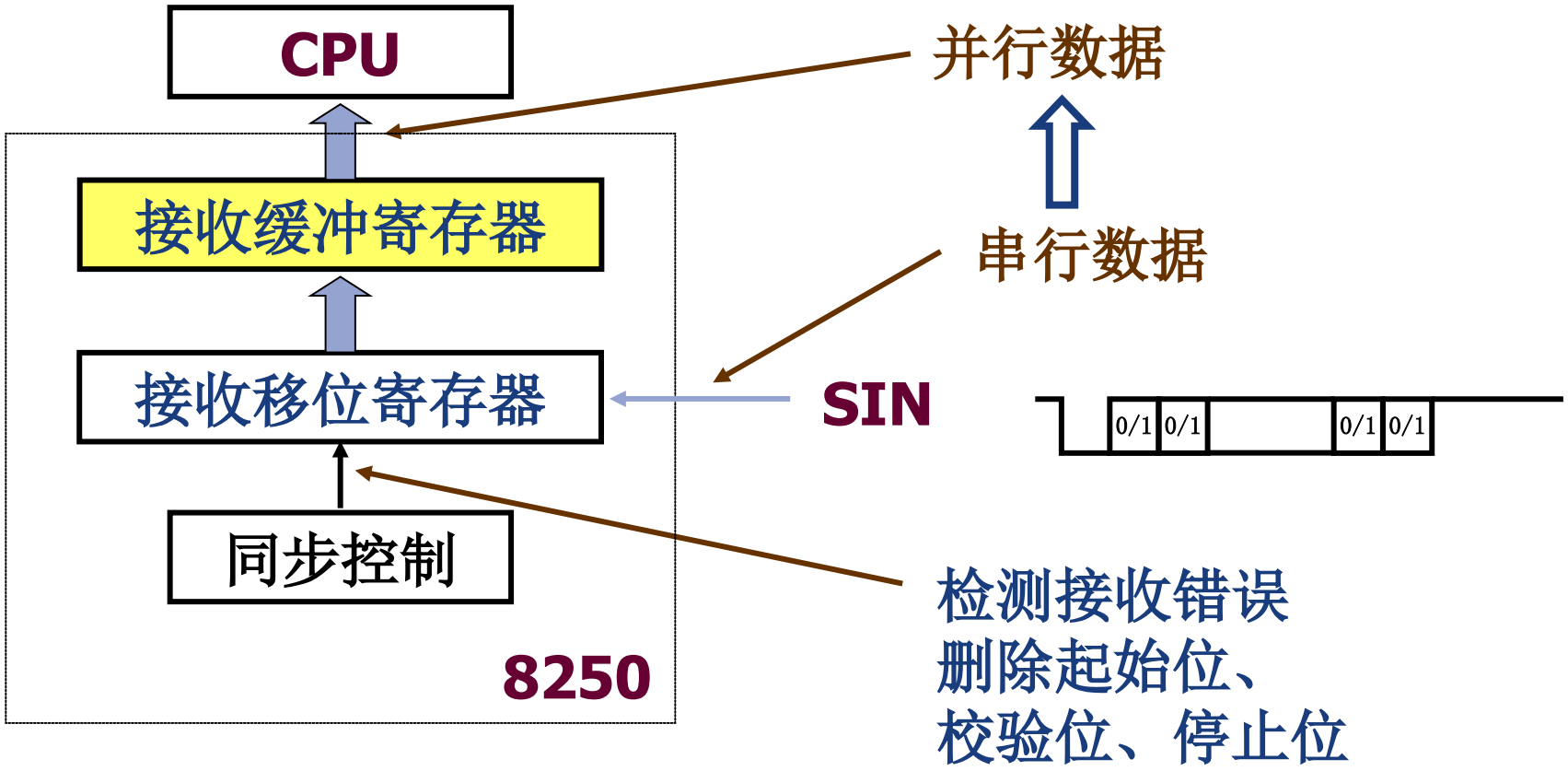
1. 8250 的寄存器

- 9 种可访问的寄存器
- 用引脚 A0 ～ A2 来寻址， DLAB 位区别

DLAB	A2 A1 A0	寄 存 器	COM1 地址	COM2 地址
0	0 0 0	读接收缓冲寄存器	3F8H	2F8H
0	0 0 0	写发送保持寄存器	3F8H	2F8H
0	0 0 1	中断允许寄存器	3F9H	2F9H
×	0 1 0	中断识别寄存器	3FAH	2FAH
×	0 1 1	通信线路控制寄存器	3FBH	2FBH
×	1 0 0	调制解调器控制寄存器	3FCH	2FCH
×	1 0 1	通信线路状态寄存器	3FDH	2FDH
×	1 1 0	调制解调器状态寄存器	3FEH	2FEH
×	1 1 1	不用	3FFH	2FFH
1	0 0 0	除数寄存器低 8 位	3F8H	2F8H
1	0 0 1	除数寄存器高 8 位	3F9H	2F9H

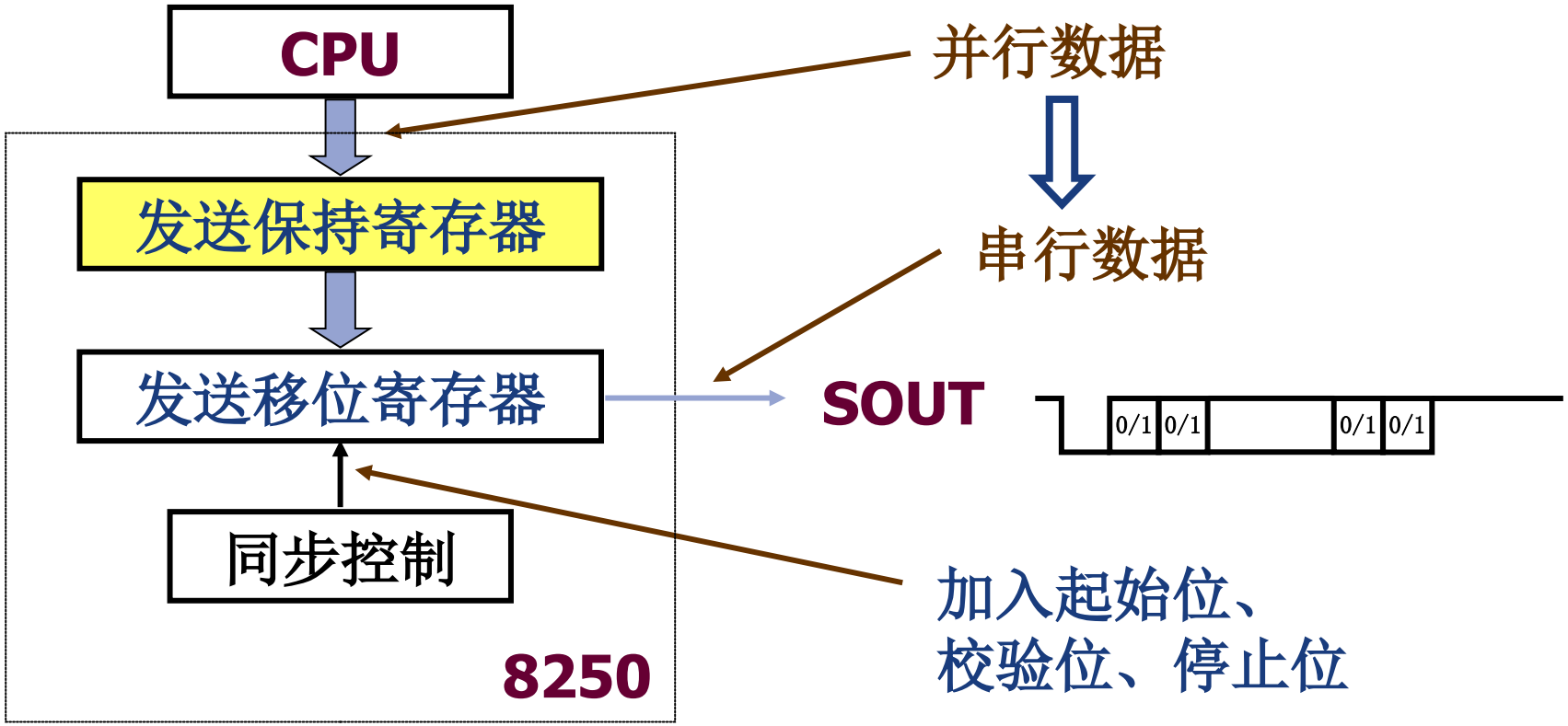
8250 的接收缓冲寄存器 RBR

- 存放串行接收后转换成并行的数据



8250 的发送保持寄存器 THR

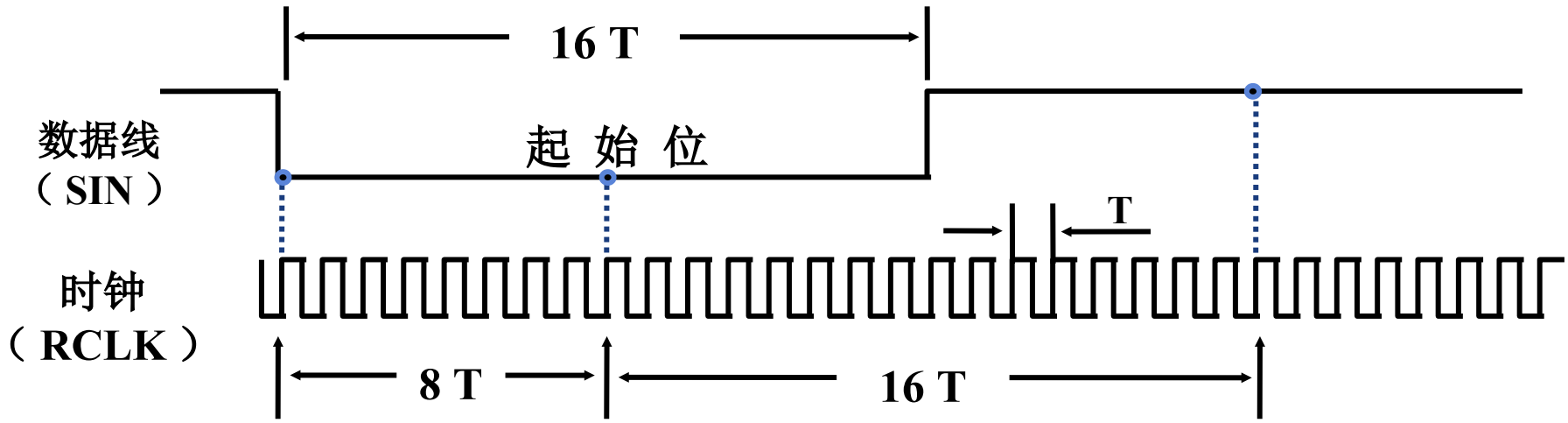
- 包含将要串行发送的并行数据



8250 的除数寄存器

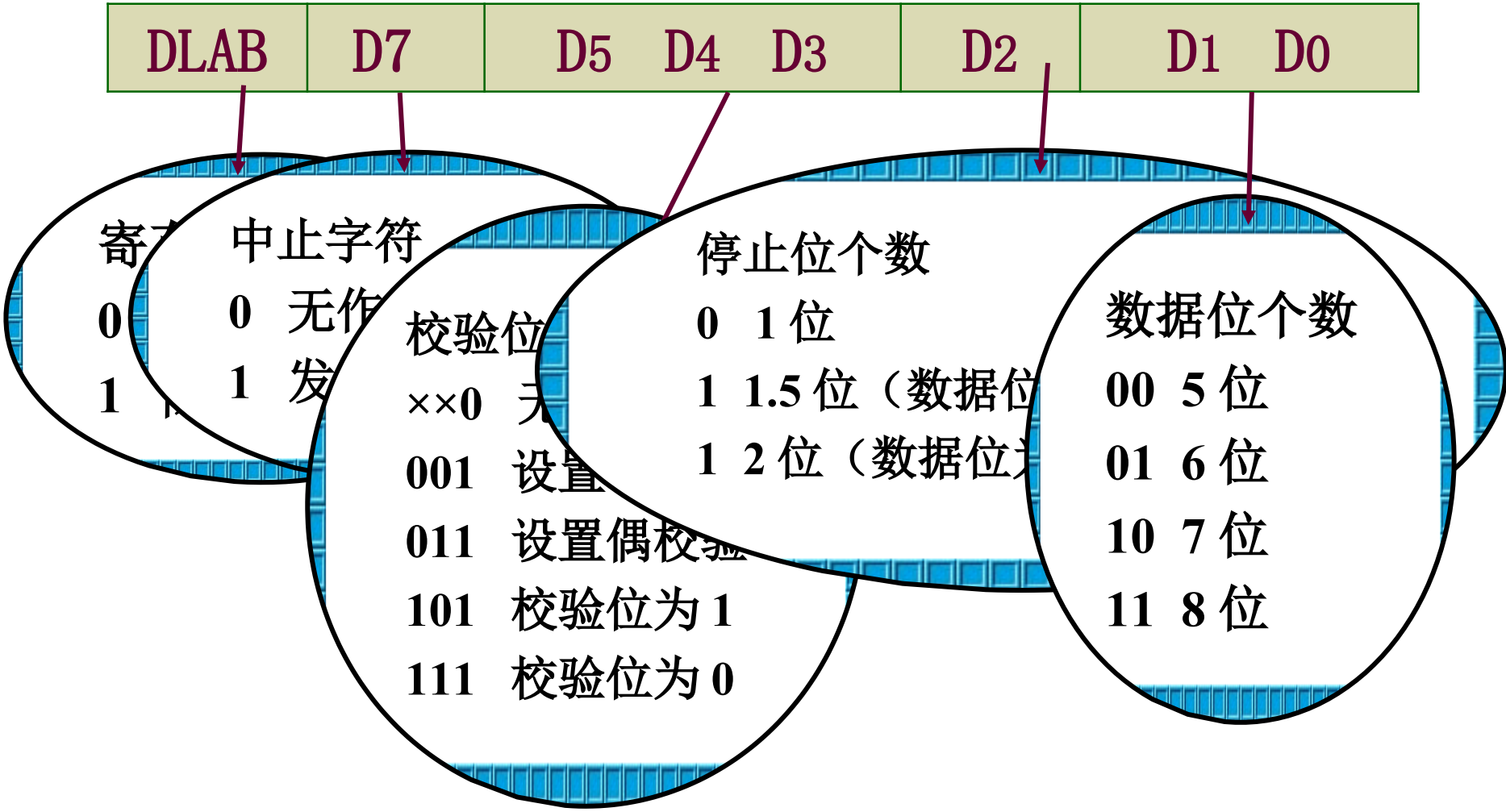
- 除数寄存器保存设定的分频系数

分频系数＝基准时钟频率 ÷ （ 16× 比特率）



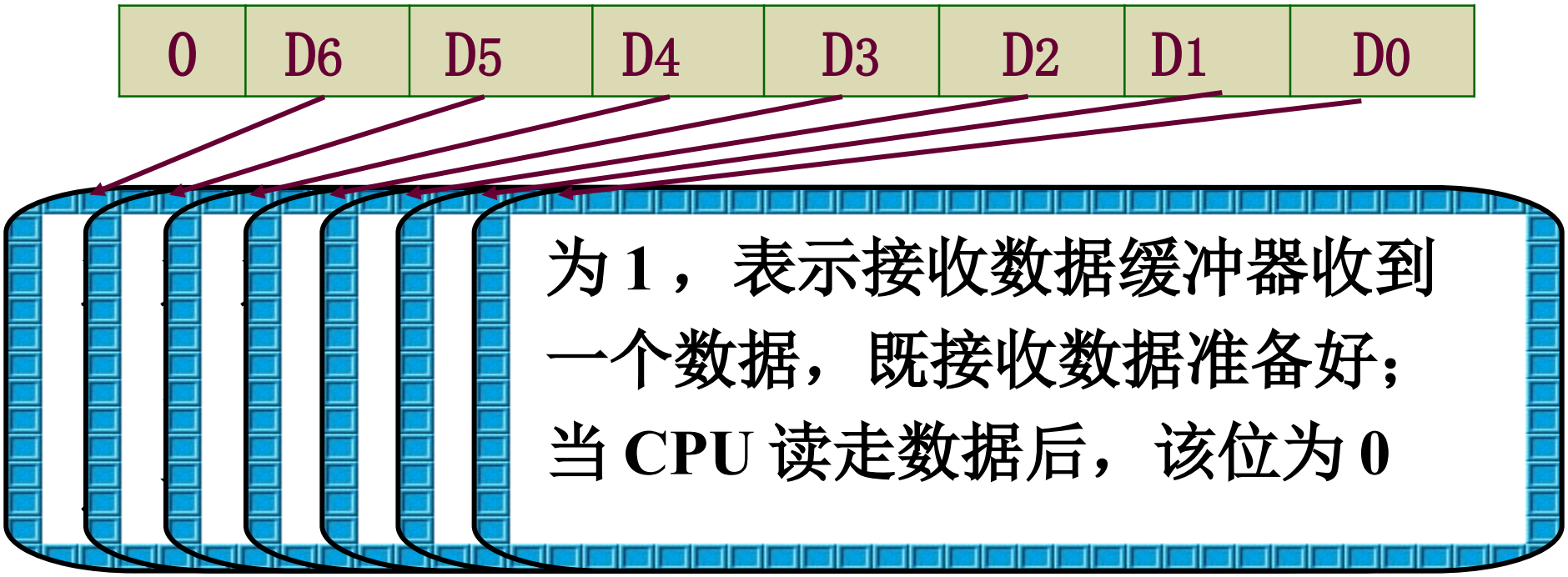
8250 的通信线路控制寄存器 LCR

- 指定串行异步通信的字符格式



8250 的通信线路状态寄存器 LSR

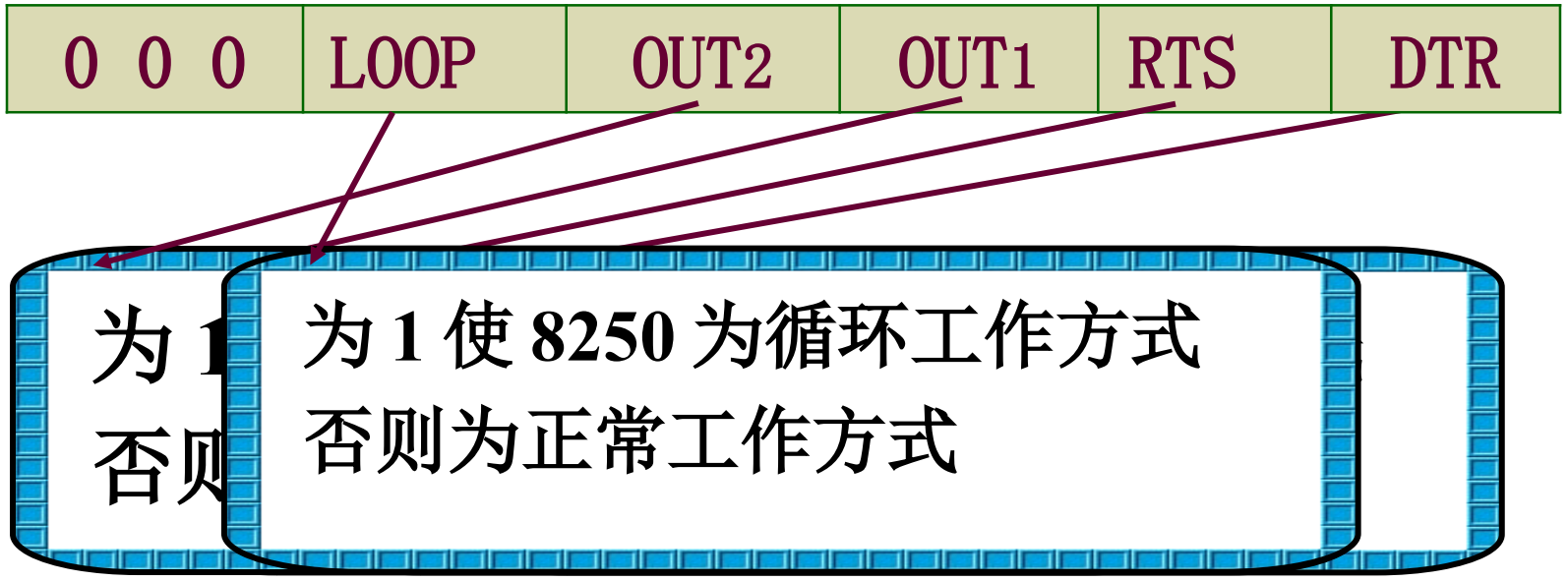
- 提供串行异步通信的当前状态



奇偶错误 PE　　帧错误 FE　　溢出错误 OE

8250 的调制解调器控制寄存器 MCR

- 设置 8250 与数据通信设备联络的输出信号



循环工作方式：

4 个控制输入信号在内部与 4 个控制输出信号相连

发送的串行数据在内部被接收

检测 8250 发送和接收功能，不必外连线

8250 的调制解调器状态寄存器 MSR

- 反映 4 个控制输入信号的当前状态及其变化
- MSR 高 4 位中某位为 1：
  - 相应输入信号当前为低有效
- MSR 低 4 位中某位为 1：
  - 从上次 CPU 读取该状态字后，相应输入信号已发生改变，从高变低或反之
  - 产生调制解调器状态中断，当 CPU 读取该寄存器或复位后，低 4 位被清零

8250 的中断允许寄存器 IER

- 8250 有 4 级 10 个中断
  - 接收线路状态中断（ 4 个）
    - 奇偶错、溢出错、帧错和中止字符
  - 接收器数据准备好中断
  - 发送保持寄存器空中断
  - 调制解调器状态中断（ 4 个）

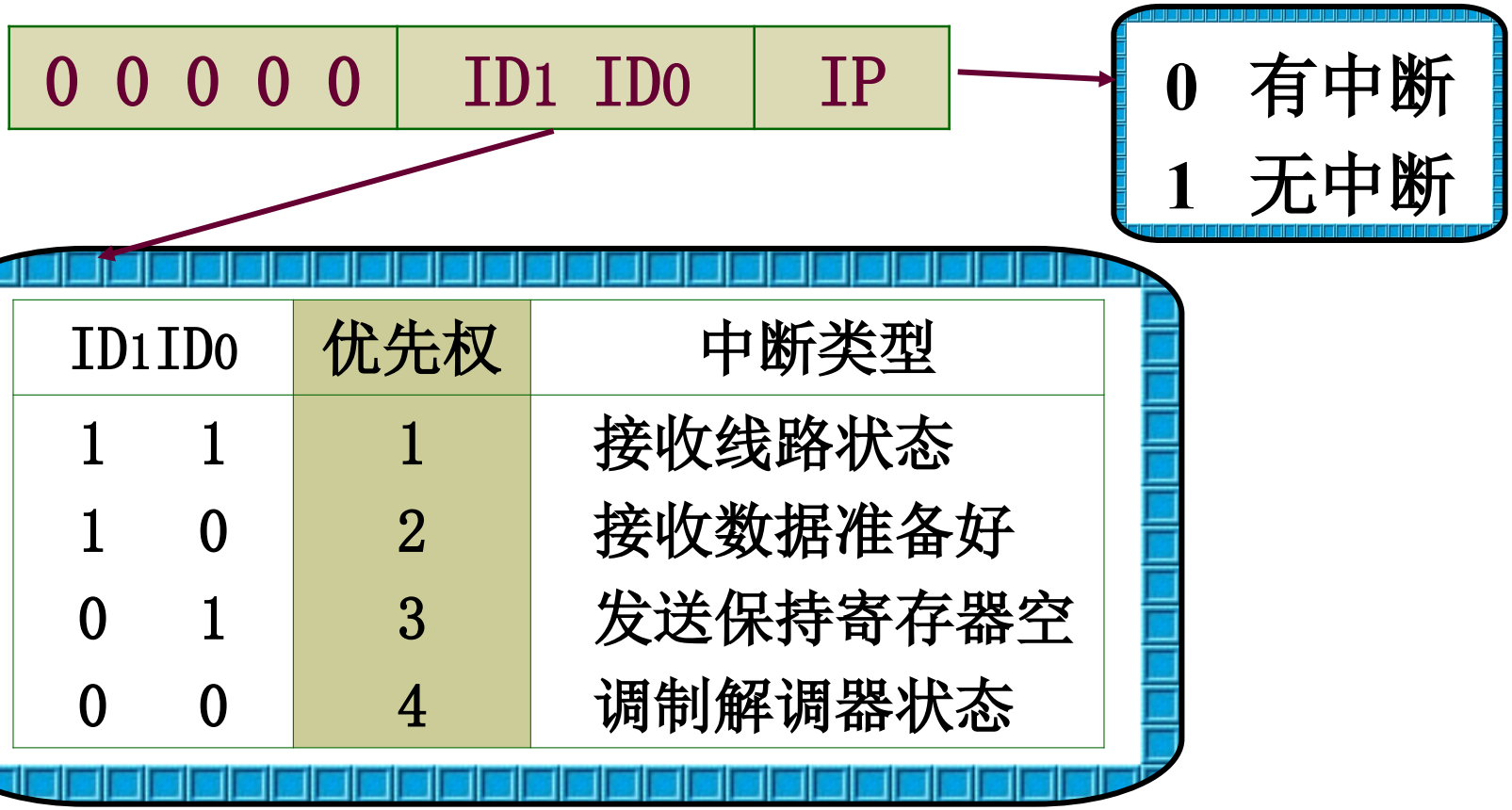
优先权高

优先权低

- 中断允许寄存器低 4 位控制 4 级中断是否允许
  - 某位为 1 ，则对应的中断被允许
  - 否则，被禁止

8250 的中断识别寄存器 IIR

- 表明是否有中断
- 保存正在请求中断优先权最高中断级别编码



ID1	ID0	优先权	中断类型
1	1	1	接收线路状态
1	0	2	接收数据准备好
0	1	3	发送保持寄存器空
0	0	4	调制解调器状态

2． 初始化编程

- 对 8250 的内部控制寄存器进行编程写入

(1) 设置传输率

写入除数寄存器

(2) 设置字符格式

写入通信线路控制寄存器

(3) 设置工作方式

写入调制解调器控制寄存器

(4) 设置中断允许或屏蔽位

写入中断允许寄存器

写入除数寄存器设置传输率

```
mov al,80h
mov dx,2fbh
out dx,al
;  写入通信线路控制寄存器，使 DLAB = 1
mov  ax,96          ;  分频系数
; 1.8432MHz÷(1200×16) = 96 = 60H
mov dx,2f8h
out dx,al           ;  写入除数寄存器低 8 位
mov al,ah
inc dx
out dx,al           ;  写入除数寄存器高 8 位
```

写入通信线路控制寄存器设置字符格式

```
; 假设使用 7 个数据位、 1 个停止位、奇校验
mov al,00001010b
mov dx,2fbh
out dx,al          ;写入通信线路控制寄存器
; 同时使 DLAB ＝ 0，以方便下述初始化过程
```

写入调制解调器控制寄存器设置工作方式

```
;  设置查询通信方式
mov  al,03h       ;  禁止中断（D3 ＝ 0）
mov  dx,2fch
out  dx,al        ;  写入调制解调器控制寄存器
;  设置中断通信方式
mov  al,0bh       ;  允许中断（D3 ＝ 1）
mov  dx,2fch
out  dx,al
;  设置查询的循环测试通信方式
mov  al,13h       ;  循环测试（D4 ＝ 1）
mov  dx,2fch      ;  禁止中断（D3 ＝ 0）
out  dx,al
```

写入中断允许寄存器设置中断允许或屏蔽位

```
;禁止所有中断
mov al,0
mov dx,2f9h
out dx,al      ;写入中断允许寄存器(DLAB＝0)
;仅允许接收中断
mov al,1
mov dx,2f9h
out dx,al      ;写入中断允许寄存器(DLAB＝0)
```

〔例 11-3〕异步通信程序 - 1

```
        ; 数据段
msg     byte 'What you see is …….',13,10,0
        ; 代码段
        ……                      ; 初始化编程
        mov si,offset msg ; SI 指向发送信息
        mov bx,1                ; BX = 1 需要发送信息
        mov cx,1                ; CX = 1 可以接收信息
        ; 读取通信线路状态，查询工作
statue: mov ax,bx
        or ax,cx                ; BX = CX = 0，接发完成
        jz done                 ; 转向结束
```

〔例 11-3 〕异步通信程序 - 2

```
mov dx,2fdh          ; 读取线路状态寄存器
in al,dx
test al,1eh          ; 接收有错误否？
jz statue1           ; 没有错误，继续
; 接收有错，响铃报警
mov dx,2f8h          ; 读出接收有误的数据
in al,dx
mov al,07h           ; 响铃控制的 ASCII 码
call dispc
jmp statue           ; 继续查询
```

〔例 11-3 〕异步通信程序 - 3

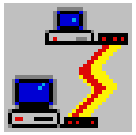
```
statue1: test al,01h      ;  接收到数据吗？
         jz statue2       ;  没有收到数据，继续
         ;  已接收字符，读取该字符并显示
         mov dx,2f8h      ;读取输入缓冲寄存器
         in al,dx
         cmp al,0         ;  是结尾字符吗?
         jnz receive
         xor cx,cx        ; CX ＝ 0 ，不再接收数据
         jmp statue       ;  继续查询
receive: and al,7fh       ;  标准 ASCII 码取低 7 位
         call dispc       ;  屏幕显示该数据
         jmp statue       ;  继续查询
```

〔例 11-3 〕异步通信程序 - 4

```
statue2: cmp bx,1          ;  有要发送的字符吗?
         jne statue        ;  无字符，继续查询
         test al,20h       ;  能输出数据吗？
         jz statue         ;  不能输出，继续查询
         ;  保持寄存器已空，可以发送数据
         mov al,[si]       ;  获得要发送的字符
         inc si
         cmp al,0          ;  是结尾字符吗?
         jnz transmit
         xor bx,bx         ;  无发送字符，  BX ＝ 0
         jmp statue        ;  继续查询
```

〔例 11-3 〕异步通信程序 - 5

```
transmit: mov dx,2f8h
          ; 将字符输出给发送保持寄存器
          out dx,al        ; 串行发送数据
          jmp statue       ; 继续查询
done:                      ; 返回 DOS
```

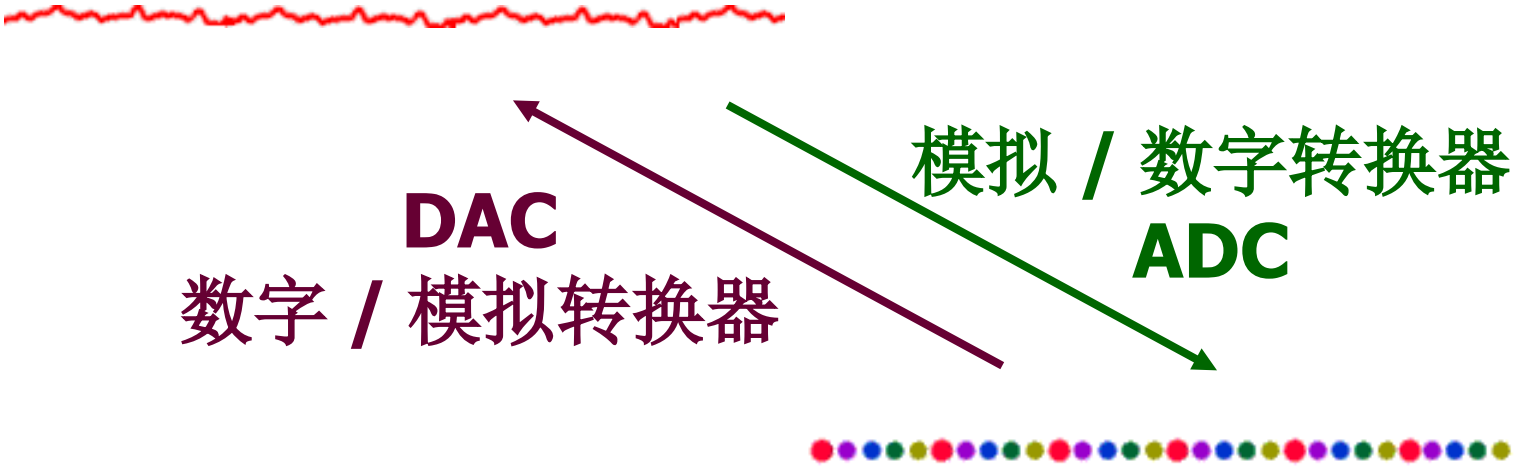


What you see is what you get.

显示结果

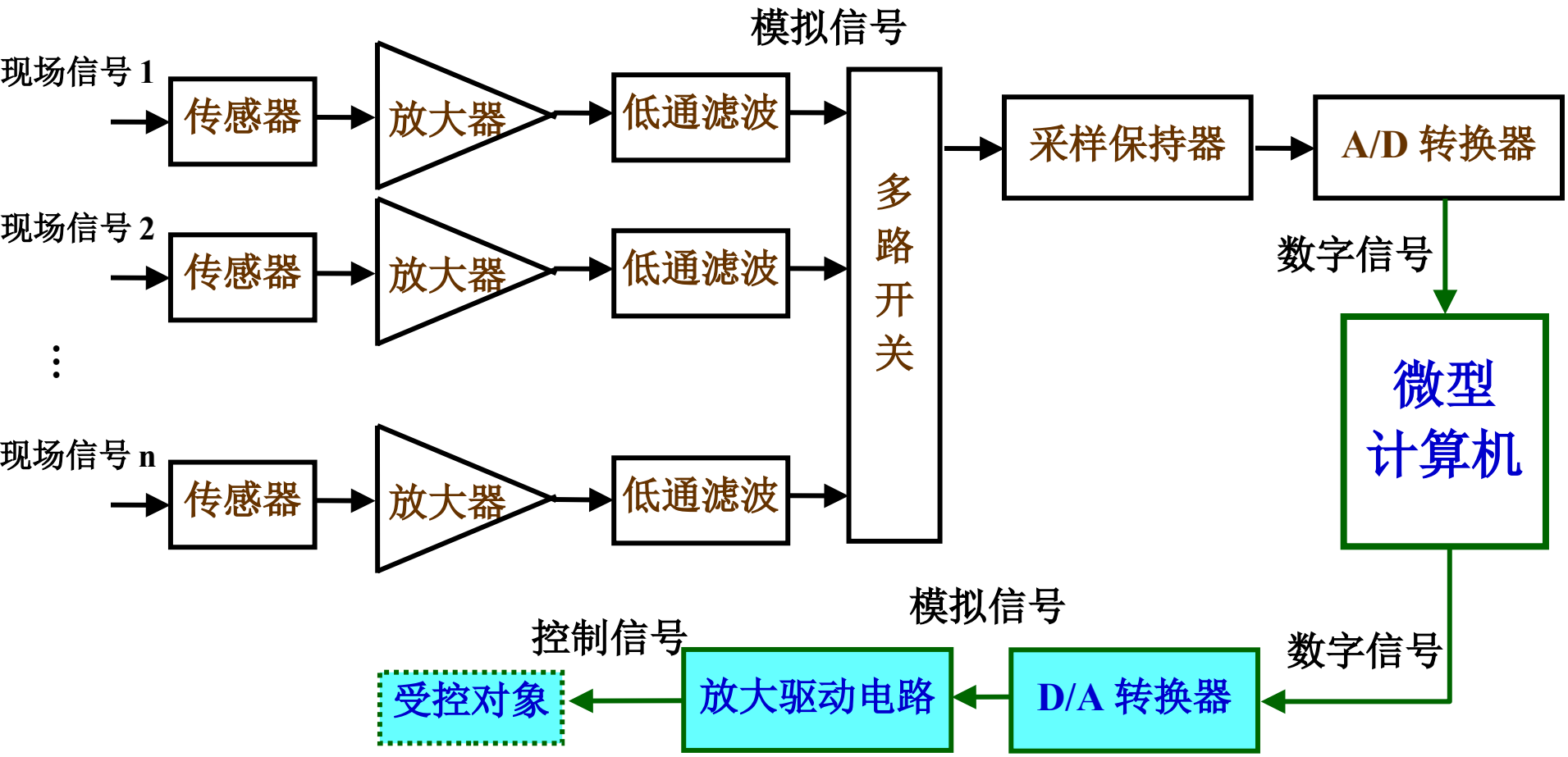
11.4 模拟接口

- 模拟量——连续变化的物理量

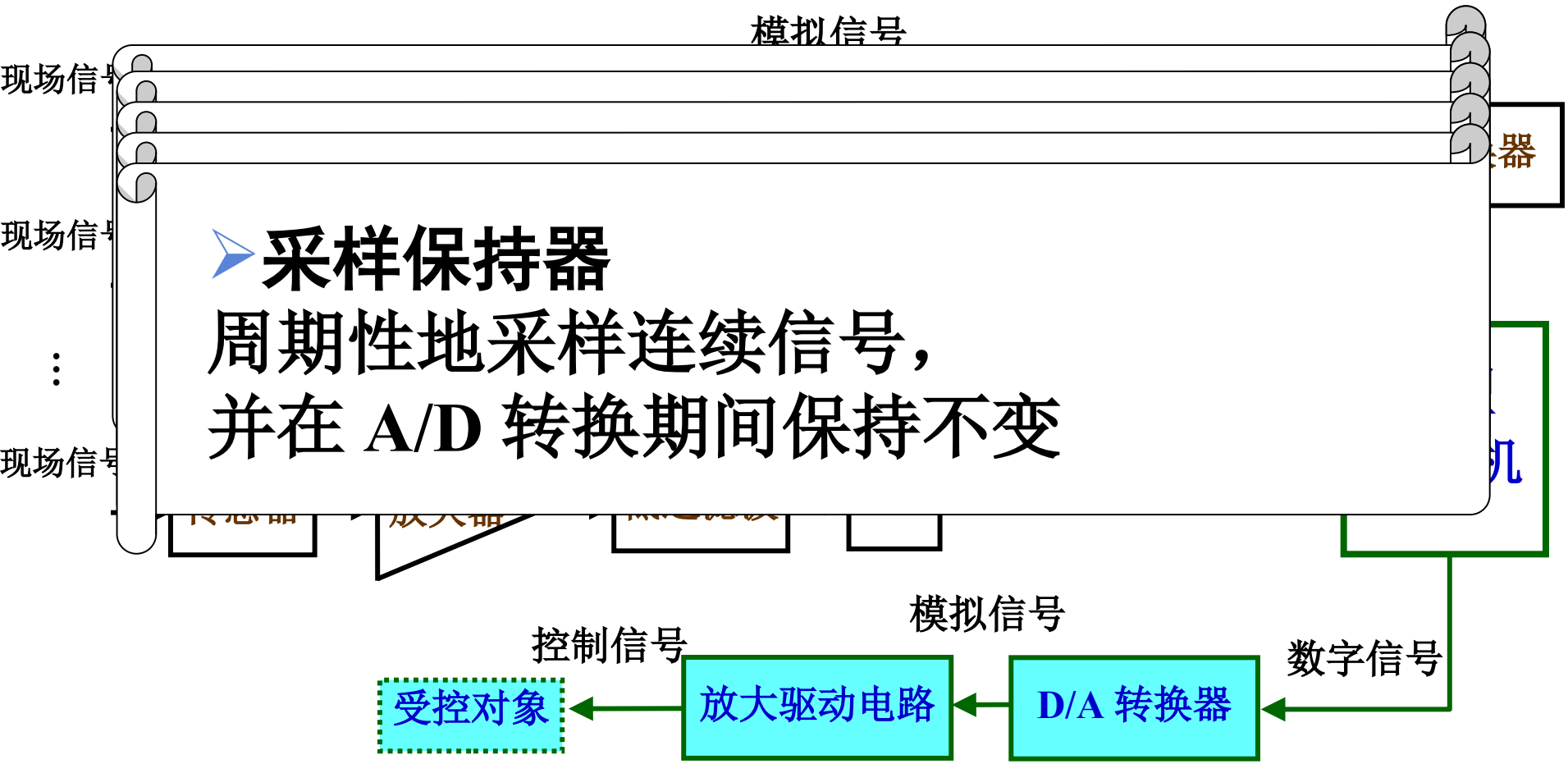


- 数字量——时间和数值上都离散的量

11.4.1 模拟输入输出系统



11.4.1 模拟输入输出系统



11.4.2 D/A 转换器

- D/A 转换器（ DAC ）
  - 将数字量转换成为模拟量（电压或电流）
- DAC 芯片有多种类型
  - 按 DAC 的性能：通用、高速和高精度等转换器
  - 按内部结构：不包含、包含数据寄存器
  - 按位数： 8 位、 12 位、 16 位等
  - 按输出模拟信号：电流输出型和电压输出型

1．D/A转换原理

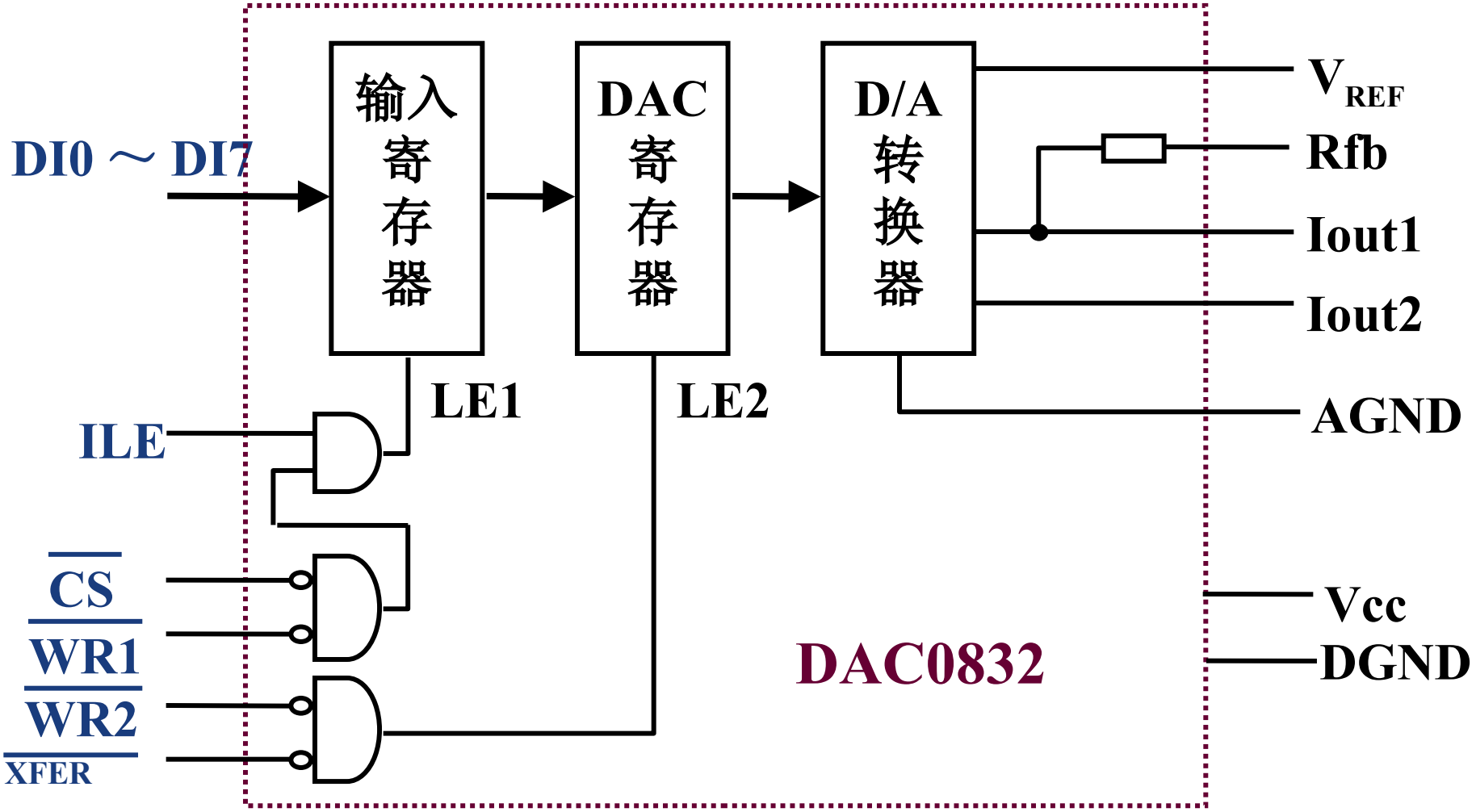
- 把每位代码按其权的大小转换成相应的模拟分量，将各模拟分量相加

$$1101B = 1\times23 + 1\times22 + 0\times21 + 1\times20 = 13$$

- D/A转换器：
  - 主要由电阻网络、电子开关和基准电压组成
  - DAC集成电路多采用R-2R梯形解码网络
  - 输入数字量控制电子开关
  - 使电阻网络中的不同电阻和基准电压接通
  - 输出端产生成比例的模拟电流或电压
  - 基准电压（参考电压VREF）：稳定的电压源

DAC0832 的内部结构

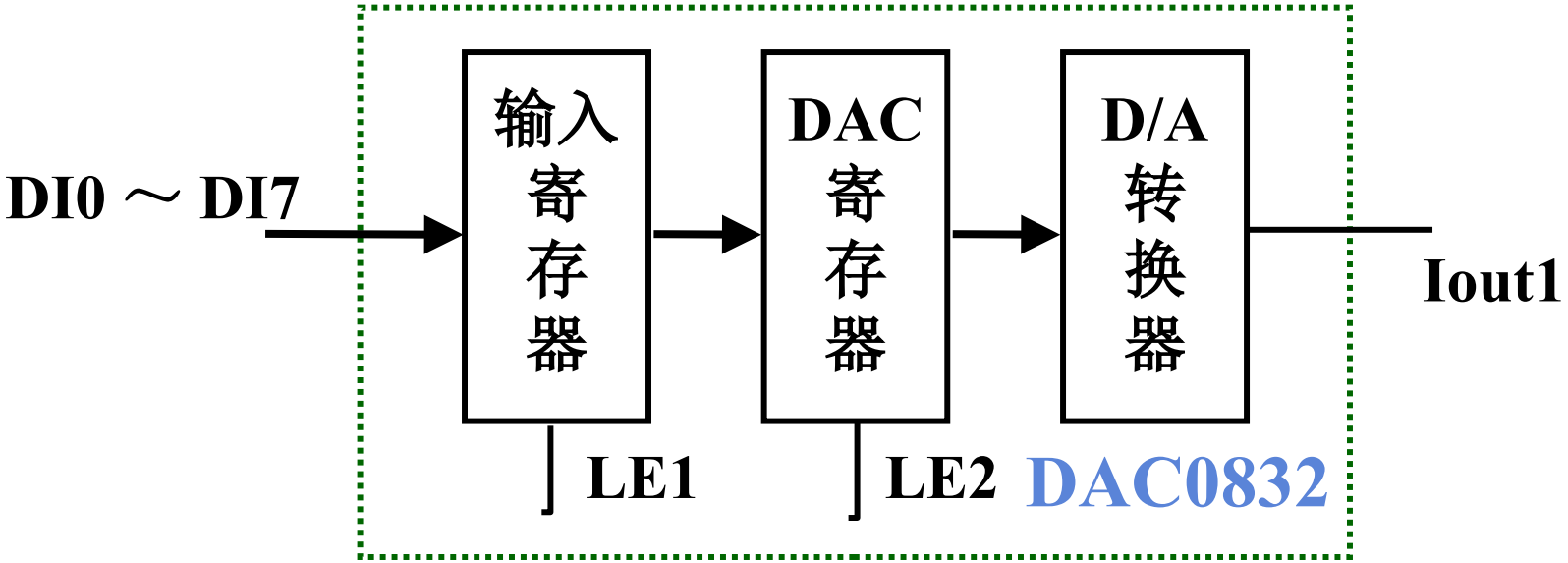
- 典型的 8 位、电流输出型、通用 DAC 芯片



2. DAC0832 的数字接口

- 8 位数字输入端
  - DI0 ～ DI7 （DI0 为最低位）
- 输入寄存器（第 1 级）控制端
  - ILE 、 CS* 、 WR1*
- DAC 寄存器（第 2 级）控制端
  - XFER* 、 WR2*

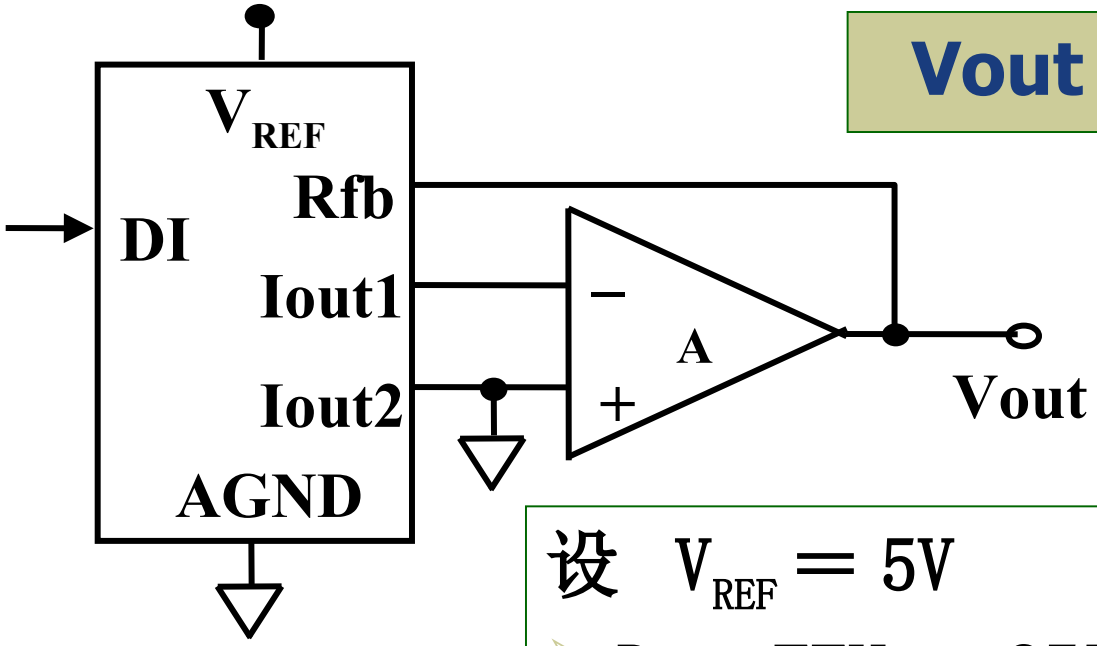
- 直通方式
- 单缓冲方式
- 双缓冲方式



3. DAC0832 的模拟输出

- Iout1 、 Iout2—— 电流输出端
- Rfb—— 反馈电阻引出端（电阻在芯片内）
- V_{REF}—— 参考电压输入端
  - ＋ 10V ～－ 10V
- AGND—— 模拟信号地
- V_{CC}—— 电源电压输入端
  - ＋ 5V ～＋ 15V
- DGND—— 数字信号地

单极性电压输出



$$Vout = -(D/2^n) \times V_{REF}$$

设 $V_{REF} = 5V$

- D = FFH = 255 时，最大输出电压：

$Vmax = -(255/256) \times 5V = -4.98V$

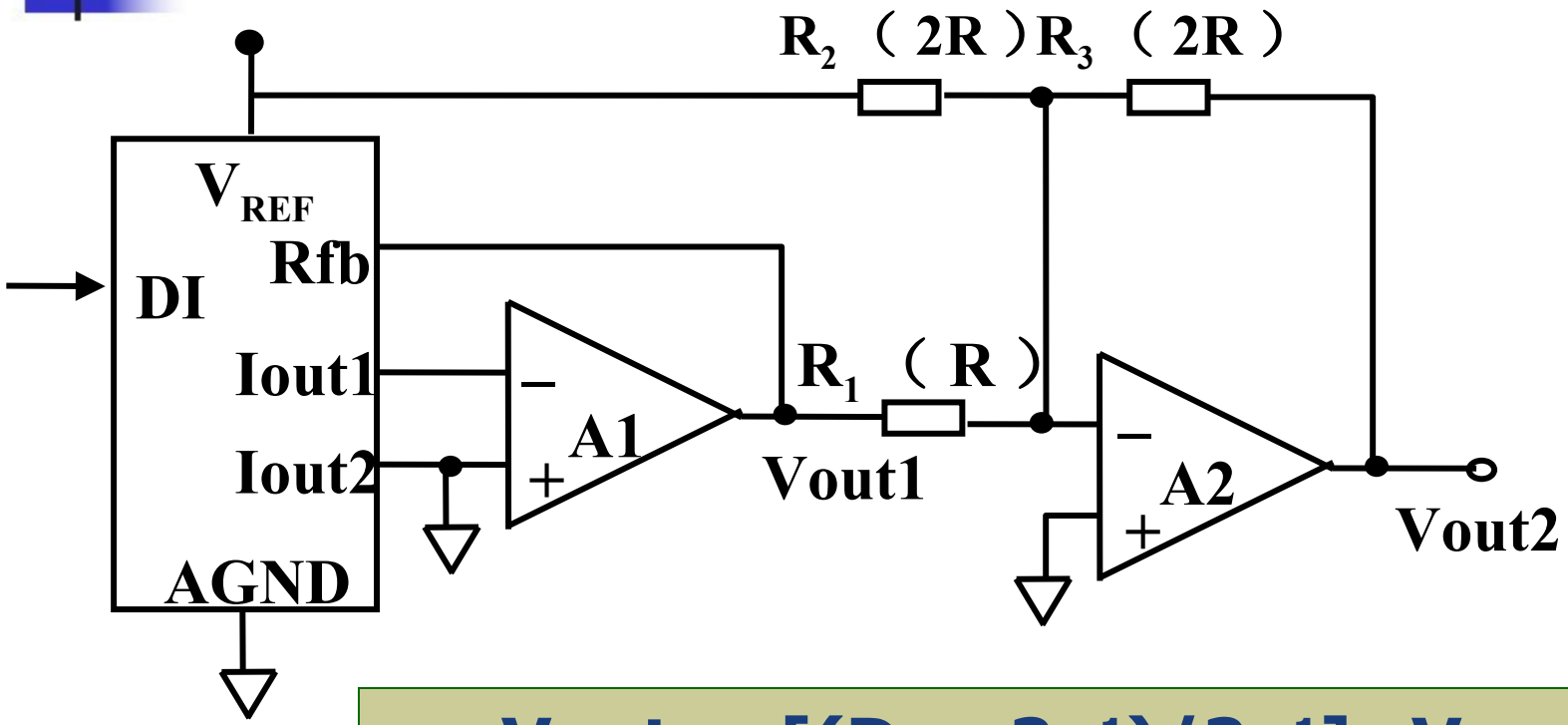
- D = 00H 时，最小输出电压：

$Vmin = -(0/256) \times 5V = 0V$

- D = 01H 时，一个最低有效位电压：

$V_{LSB} = -(1/256) \times 5V = -0.02V$

双极性电压输出



$$Vout = [(D - 2^{n-1})/2^{n-1}] \times V_{REF}$$

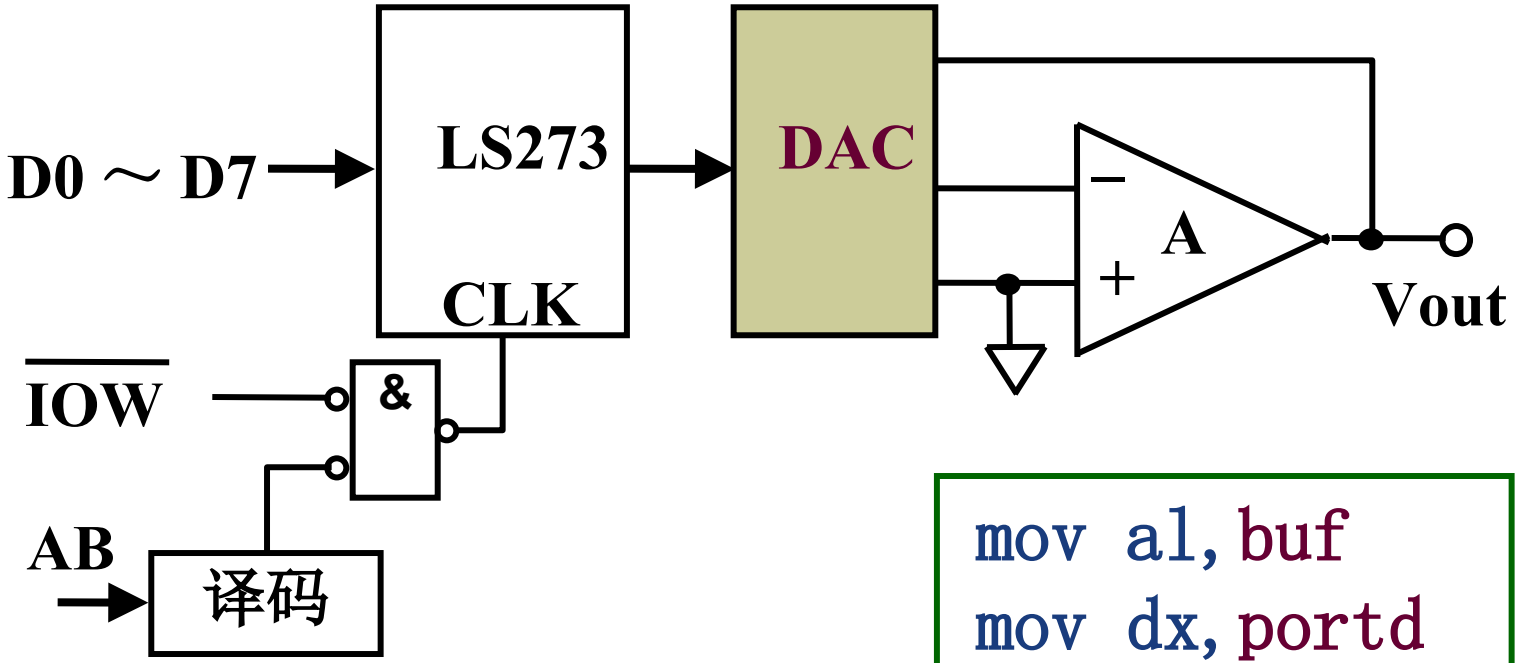
$V_{max} = [(255-128)/128] \times 5V = 4.96V$

$V_{min} = [(0-128)/128] \times 5V = -5V$

$V_{LSB} = [(129-128/128] \times 5V = 0.04V$

4．DAC芯片与主机的连接

- DAC芯片相当于一个“输出设备”
- 至少需要一级锁存器作为接口电路



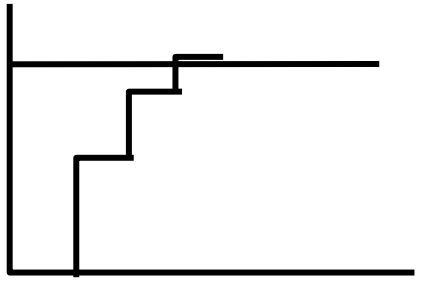

```
mov al,buf
mov dx,portd
out dx,al
```

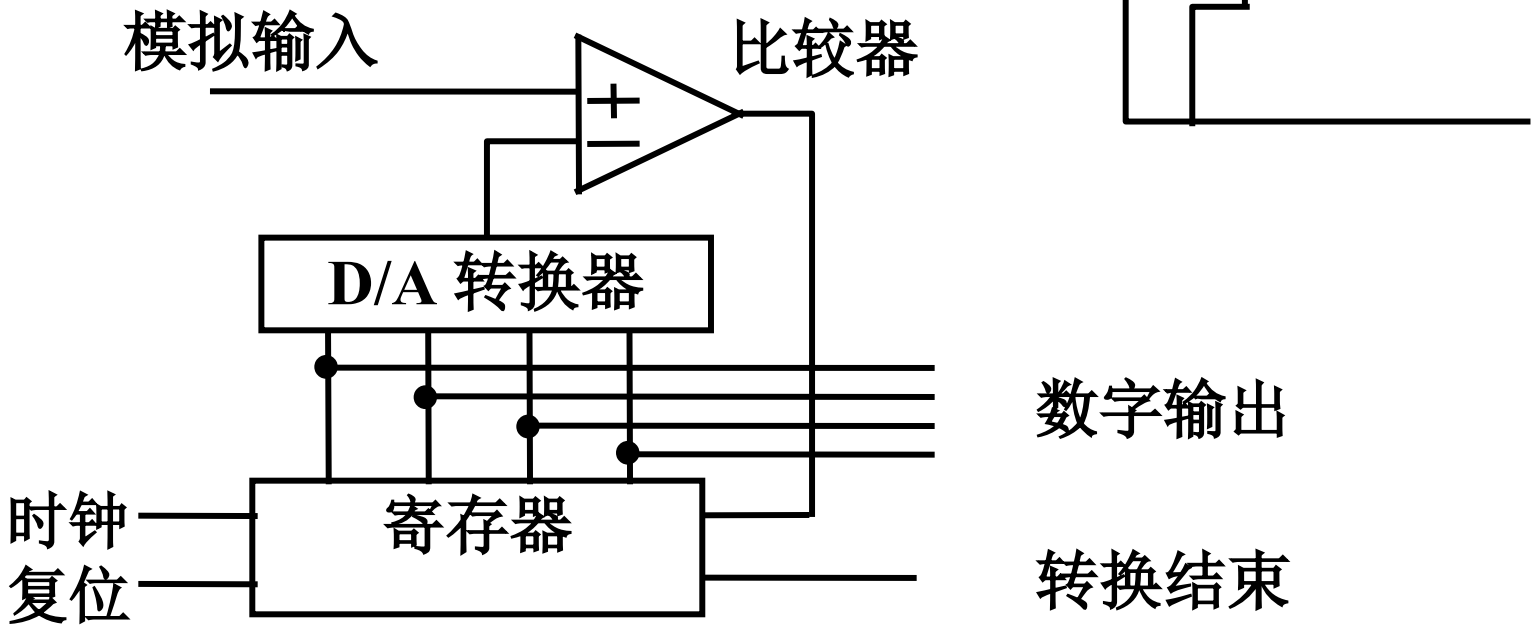
11.4.3 A/D 转换器

- A/D 转换器（ ADC ）
  - 将模拟量（电压或电流）转换成为数字量
- ADC 芯片主要以模拟到数字转换技术区别
- 有些 ADC 芯片不仅具有 A/D 转换的基本功能，还包含内部放大器和三态输出锁存器
- 有的还包括多路开关、采样保持器等

1. A/D 转换原理

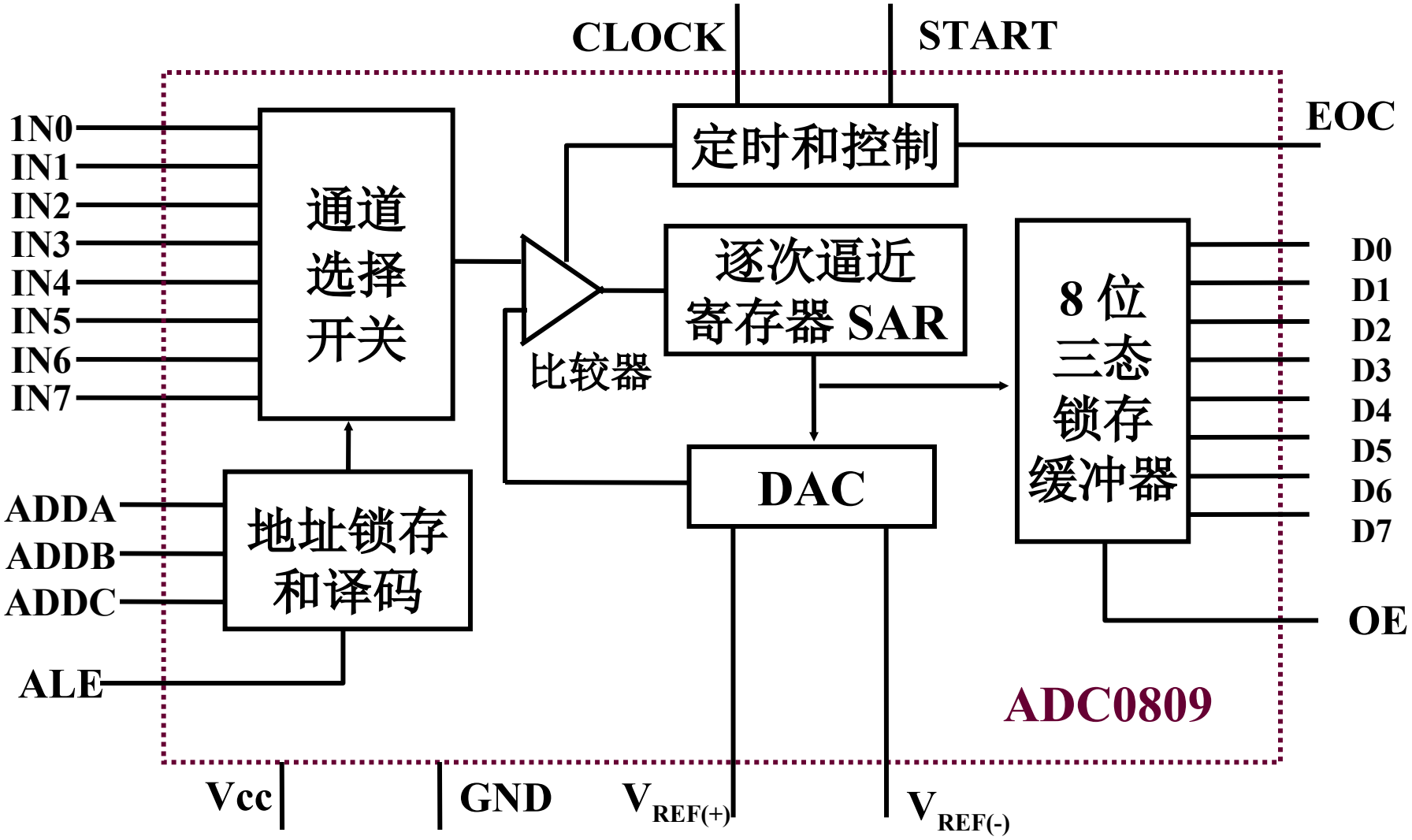
- 逐次逼近式 A/D 转换原理
- 从最高位开始的逐位试探法





ADC0809 的内部结构

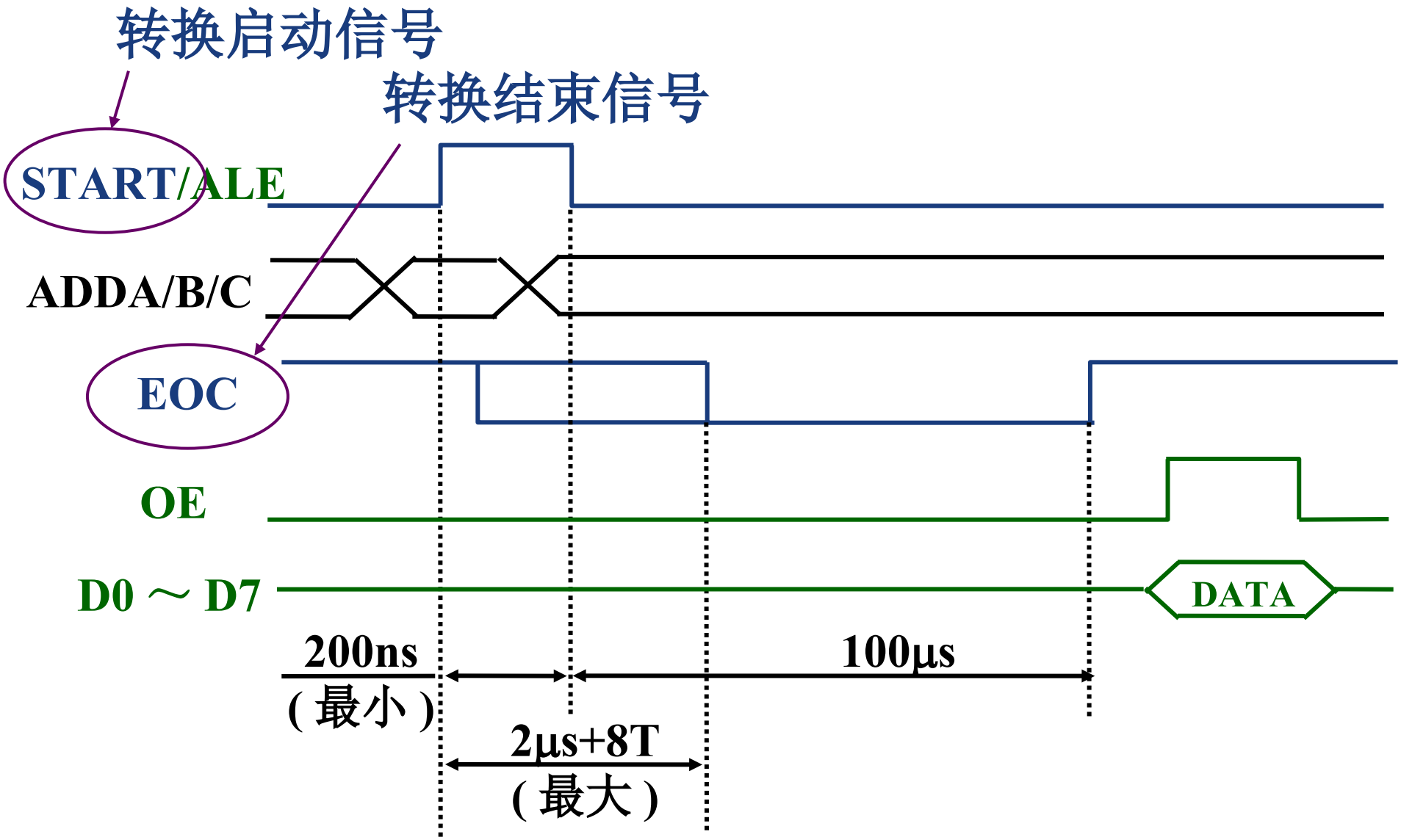
- 8 位逐次逼近式 ADC，多路开关和三态锁存缓冲器



2. ADC0809 的模拟输入

- ✓ IN0 ～ IN7 ： 8 个模拟电压输入端
- ✓ ADDA 、 ADDB 、 ADDC ： 3 个地址输入线
- ✓ ALE ：地址锁存允许信号
- ➢ 提供一个 8 通道的多路开关和寻址逻辑
- ➢ ALE 的上升沿用于锁存 3 个地址输入的状态
- ➢ 译码器从 8 个选择一个模拟输入进行 A/D 转换

3. ADC0809 的转换时序



4. ADC0809 的数字输出

- 输出允许信号 OE 高电平有效
- 将三态锁存缓冲器的数字量从 D0 ～ D7 输出

$$N = \frac{V_{in} - V_{REF(-)}}{V_{REF(+)} - V_{REF(-)}} \times 2^8$$

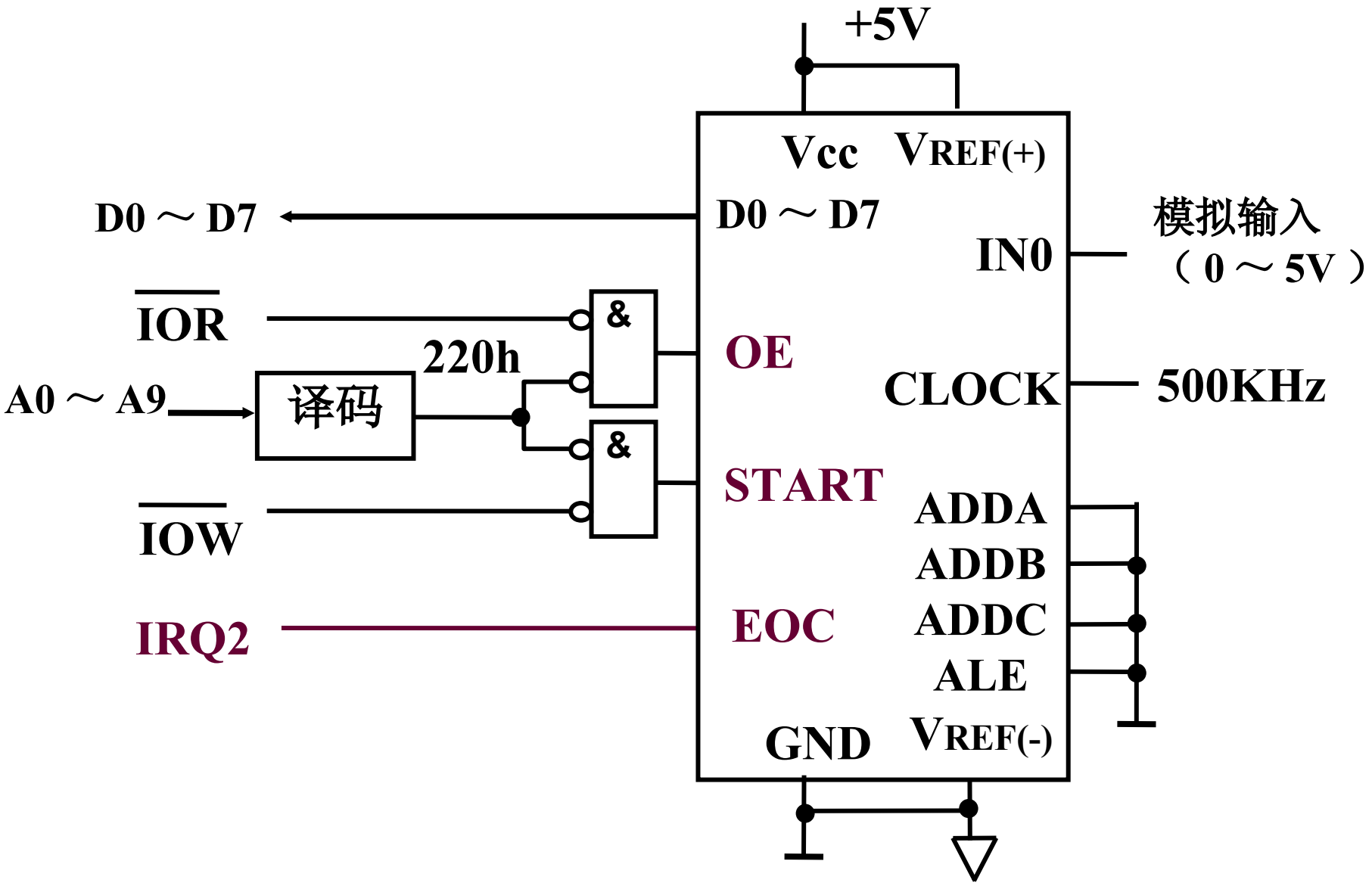
- 基准电压 $V_{REF(+)} = 5V$，$V_{REF(-)} = 0V$
- 输入模拟电压 $V_{in} = 1.5V$

$$N = (1.5 - 0) \div (5 - 0) \times 256 = 76.8 \approx 77 = 4DH$$

5．ADC 芯片与主机的连接

- ADC 芯片相当于“输入设备”
- 需要接口电路提供数据缓冲器
- 转换开始需要启动信号
  - 软件编程或硬件定时产生脉冲信号或电平信号
- 转换结束输出结束信号
  - 主机获知转换是否结束，进行数据输入
  - 查询方式：把结束信号作为状态信号
  - 中断方式：把结束信号作为中断请求信号
  - 延时方式：不使用转换结束信号
  - DMA 方式：把结束信号作为 DMA 请求信号

6. ADC 芯片的应用：中断方式



中断方式：主程序

```
        ;数据段设置缓冲区
adtemp  db 0                ;给定一个临时变量
        ;代码段
        ……                  ;设置中断向量等工作
        sti                 ;开中断
        mov dx,220h
        out dx,al           ;启动 A/D 转换
        ……                  ;其他工作
```

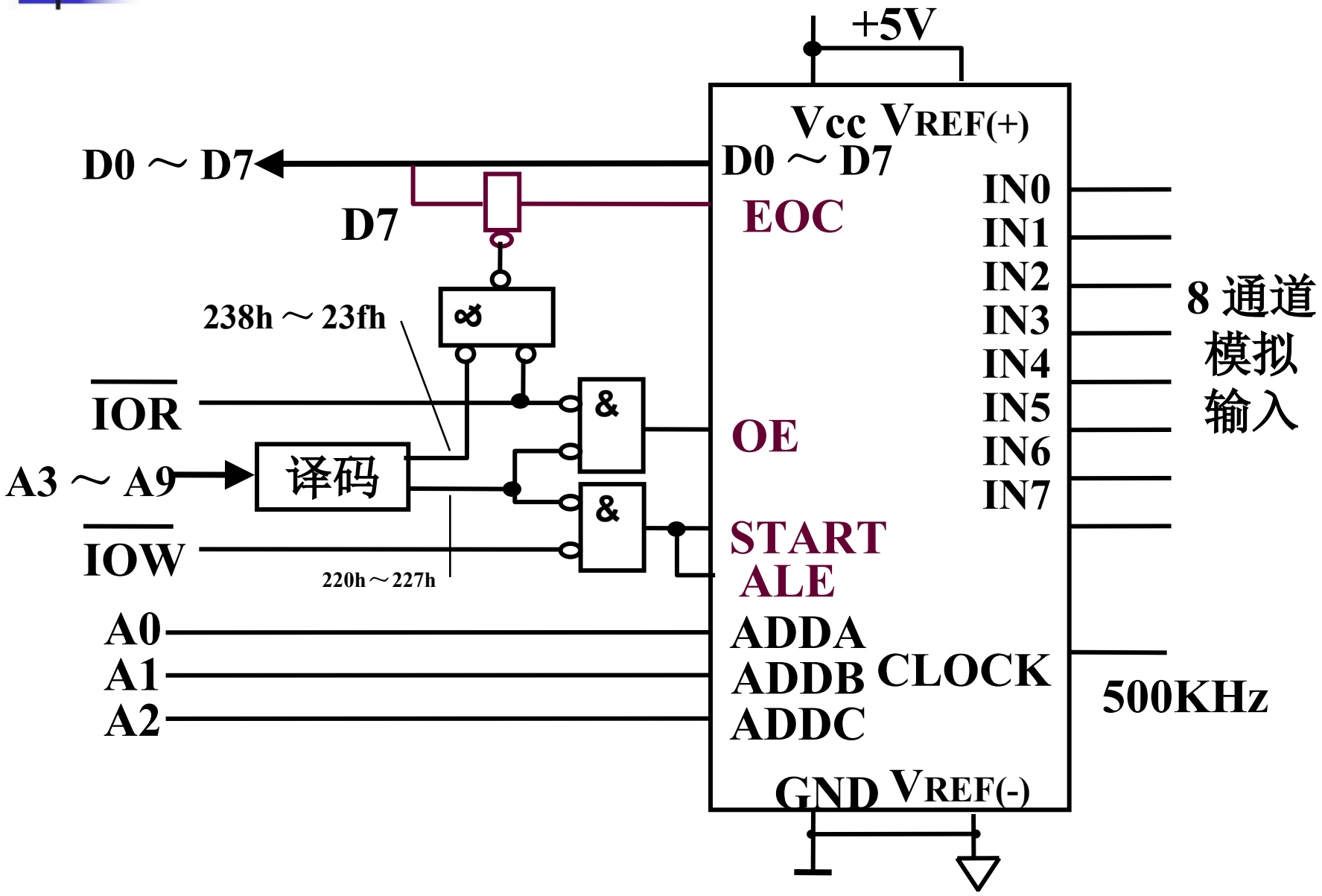
中断方式：中断服务程序 - 1

```
adint   proc                ; 中断服务程序
        sti                 ; 开中断
        push ax             ; 保护寄存器
        push dx
        push ds
        mov ax,@data        ; 设置数据段 DS 的段地址
        mov ds,ax
        mov dx,220h
        in al,dx            ; 读取 A/D 转换后的数字量
        mov adtemp,al       ; 送入缓冲区
```

中断方式：中断服务程序 - 2

```
        mov  al,20h     ;给中断控制器发送 EOI 命令
        out  20h,al
        pop  ds         ;恢复寄存器
        pop  dx
        pop  ax
        iret            ;中断返回
adint   endp
```

7．ADC 芯片的应用：查询方式



查询方式程序 - 1

```
        ;数据段
counter equ  8
buf     db counter dup(0)  ;设立数据缓冲区
        ;代码段
        mov  bx,offset  buf  ;BX← 缓冲区偏移地址
        mov  cx,counter      ;CX← 检测的数据个数
        mov  dx,220h         ;从 IN0 开始转换
start1: out  dx,al           ;启动 A/D 转换
        push dx
```

查询方式程序 - 2

```
        mov dx,238h     ;循环查询是否转换结束
start2: in al,dx        ;读入状态信息
        test al,80h     ;D7 = 1，转换结束否？
        jz start2       ;没有结束，则继续查询
        pop dx          ;转换结束
        in al,dx        ;读取数据
        mov [bx],al     ;存入缓冲区
        inc bx
        inc dx
        loop start1     ;转向下一个模拟通道
        ……              ;数据处理
```

- 掌握 8253 引脚、工作方式、编程和应用
- 熟悉 8255A 的结构特点和引脚功能
- 掌握 8255A 的方式 0/1 的编程及应用
- 掌握简易键盘编程和理解 PC 机键盘的工作原理
- 掌握 LED 数码管编程
- 掌握起止式通信协议、 232C 引脚定义和连接
- 了解 8250 的内部寄存器功能
- 理解异步通信适配器的初始化编程和通信程序
- 了解模拟输入输出系统
- 熟悉 DAC0832 和 ADC0809
- 理解 DAC 和 ADC 芯片与主机连接问题
- 掌握 ADC 芯片的应用