第七章　并行通信与并行接口

一、概述

二、并行接口芯片 **Intel 8255A-5**

三、**IBM PC/XT** 中的 **8255A-5**的使用

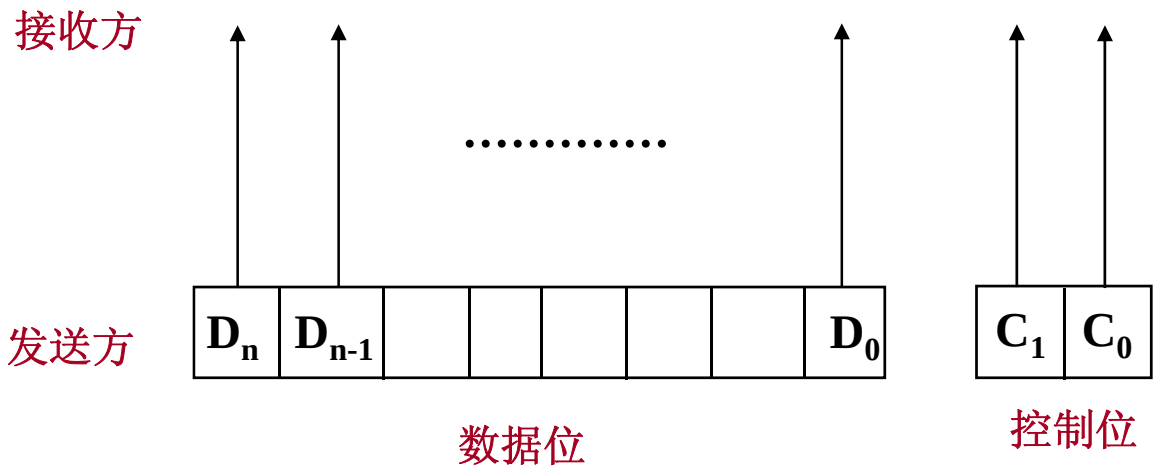
1. 喇叭接口

2. 键盘接口

一、概述

1. 并行通信

在多根传输线上同时传送数据。



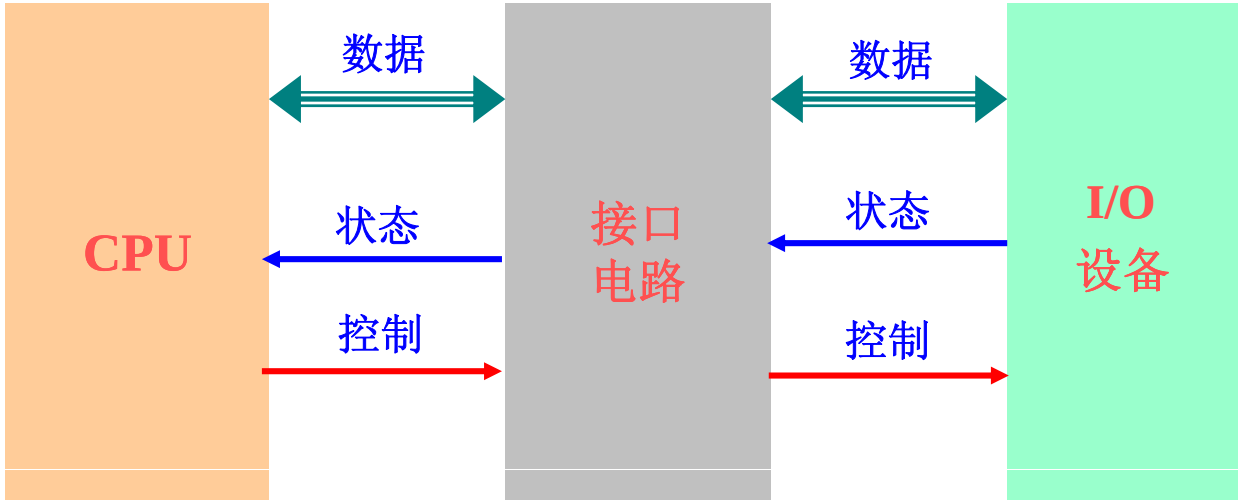
特点：速度快，但造价高；适合于短距离传送。

2. 接口

如前所述，CPU要从外设输入信息或输出信息给外设，可以采用：

- 程序查询方式；
- 中断方式；
- DMA方式。

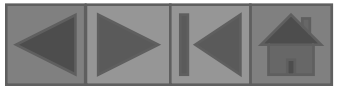
但不论用哪一种方式，CPU总是通过接口电路与外设联系的。



接口电路中要有：

- 端口的译码和控制电路
- 输入输出数据的锁存器和缓冲器
- 状态和控制命令寄存器——以便于CPU与接口电路之间用应答方式（程序查询式）来交换信息；也便于接口电路与外设间传送信息。
- 中断请求触发器 —— 为了与 CPU以中断的方式交换信息。

随着大规模集成电路技术的发展，生产了许多通用的可编程序的接口芯片，这些接口芯片按数据传送的方式可分为并行接口和串行接口两大类。



3. 通常并行接口芯片应具有以下功能

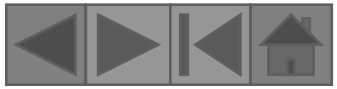
(1) 两个或两个以上的具有锁存器或缓冲器的数据端口；

(2) 每个数据端口都应有与CPU用应答方式交换信息所必需的电路；

(3) 通常每个数据端口还具有能用中断方式与CPU交换信息所必需的电路；

(4) 片选和控制电路；

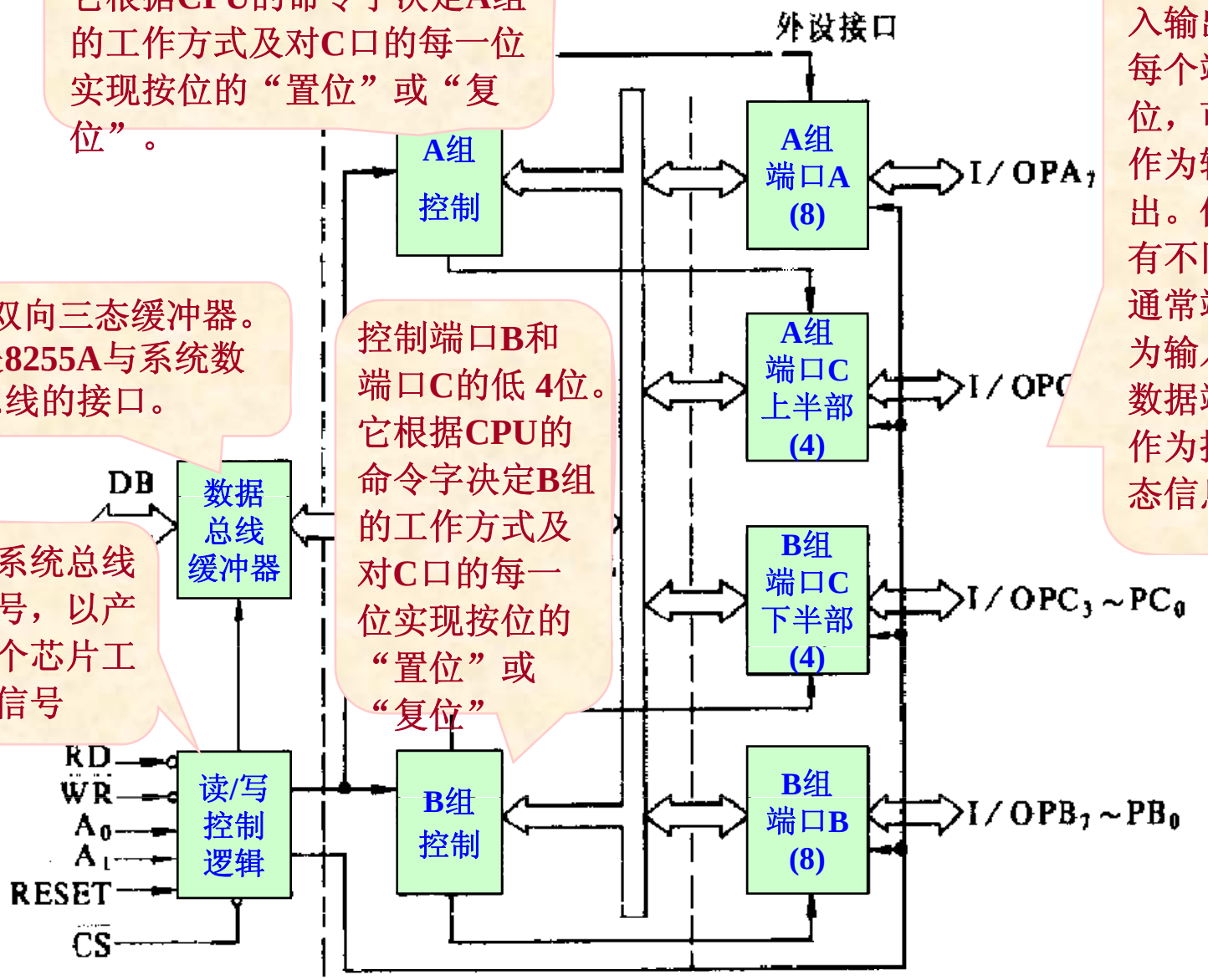
(5) 控制字寄存器 —— 通常这类片子可用程序选择数据端口，选择端口的传送方向（输入输出或双向）； 选择与CPU 交换信息的方式（中断或查询）等等，故片中要有能实现这些选择的控制字寄存器，它可由CPU用输出命令来写。



二、并行

1. 结

控制端口A和端口C的高4位。它根据CPU的命令字决定A组的工作方式及对C口的每一位实现按位的“置位”或“复位”。



8位双向三态缓冲器。它是8255A与系统数据总线的接口。

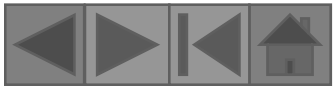
接收来自系统总线的控制信号，以产生控制整个芯片工作的控制信号

控制端口B和端口C的低 4位。它根据CPU的命令字决定B组的工作方式及对C口的每一位实现按位的“置位”或“复位”

端口A,B,C为输入输出端口，每个端口都是8位，可以选择作为输入或输出。但功能上有不同的特点。通常端口A,B作为输入输出的数据端口,C口作为控制或状态信息的端口。

8255A端口选择表

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	输 入 操 作 (读)
0	0	0	1	0	端口 A→数据总线
0	1	0	1	0	端口 B→数据总线
1	0	0	1	0	端口 C →数据总线
					输 出 操 作 (写)
0	0	1	0	0	数据总线→端口 A
0	1	1	0	0	数据总线→端口 B
1	0	1	0	0	数据总线→端口 C
1	1	1	0	0	数据总线→控制字寄存器
					断 开 功 能
×	×	×	×	1	数据总线→三态
1	1	0	1	0	非法状态
×	×	1	1	0	数据总线→三态



2. 方式选择

8255A有三种基本工作方式，

(1) 方式0——基本输入输出方式

(2) 方式1——选通输入输出方式

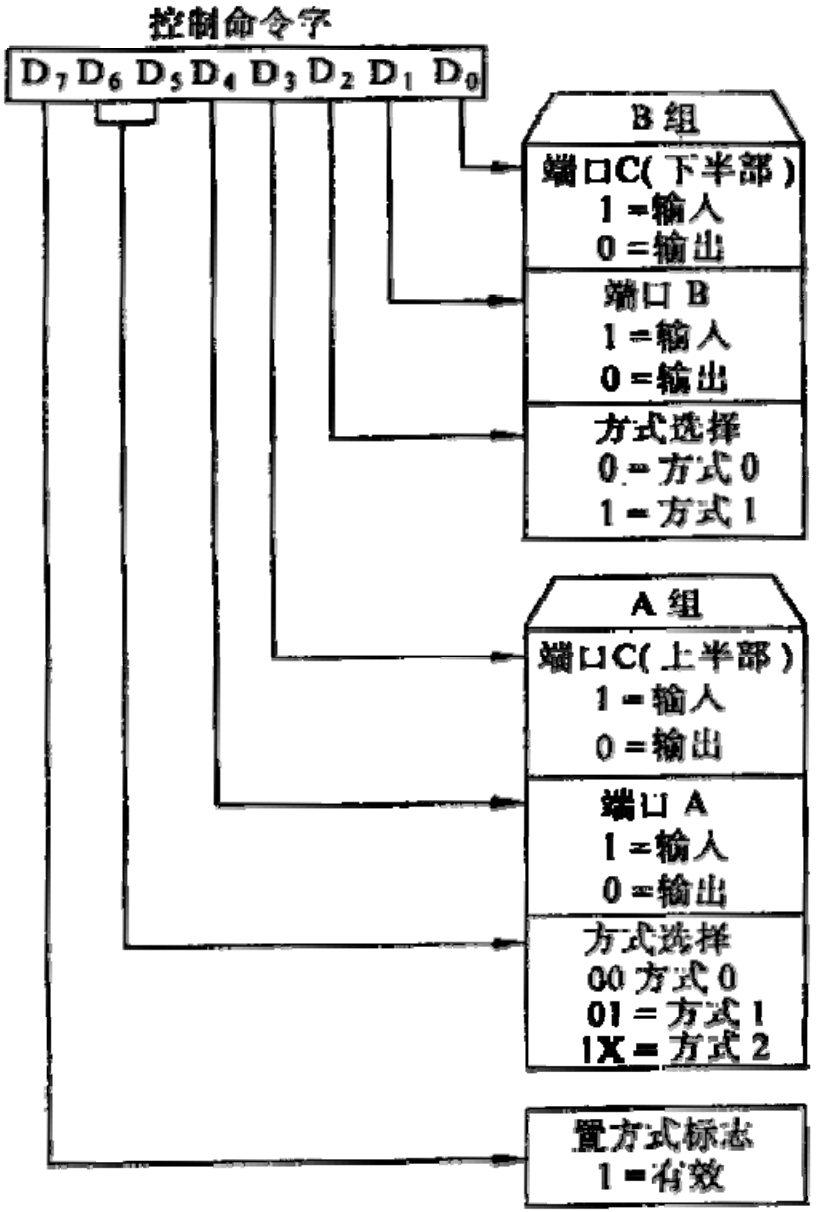
(3) 方式2——双向传送方式

方式由CPU输出的命令字决定。

由此可见：

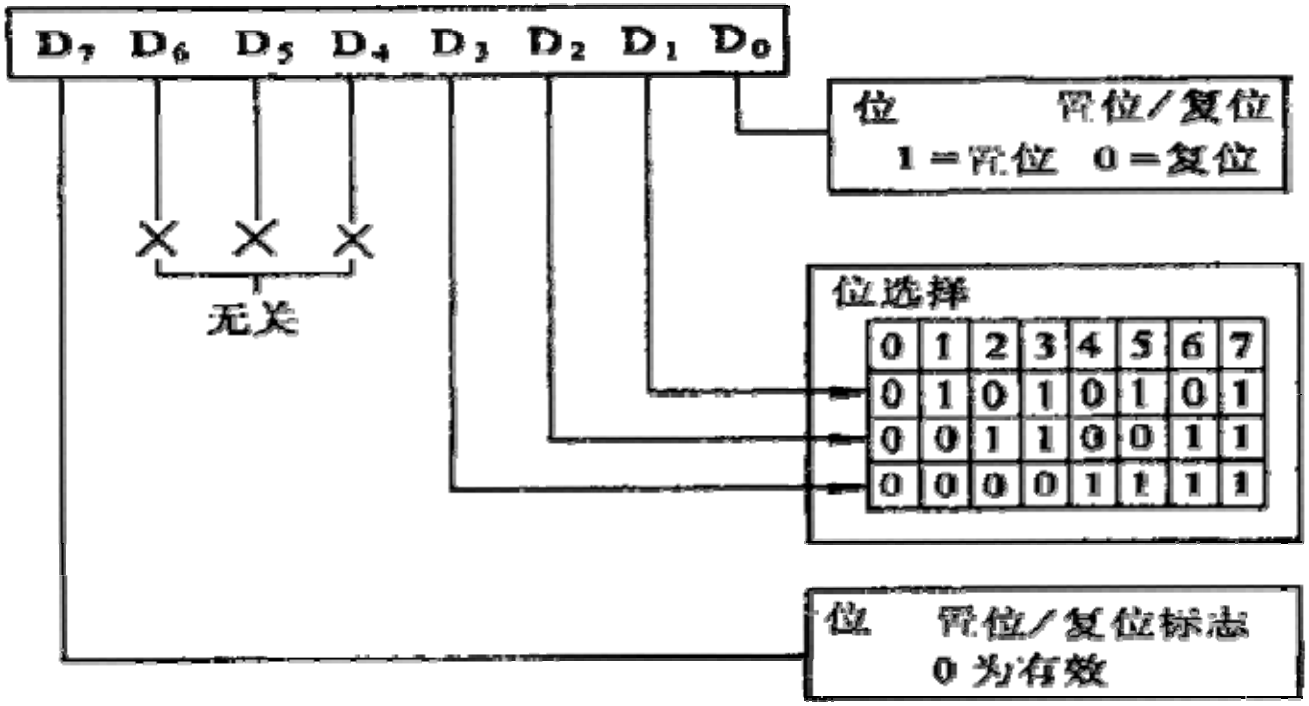
端口A有0，1，2 三种工作方式，

端口B只能工作于方式0，1。



3. 按位置位/复位功能

端口C的8位中的任一位，可用一条指令来“置位”和“复位”（其它位状态不
变），这个功能主要用于控制。实现此功能的控制字为：



4. 8255A的中断功能

当8255A按模式1 或模式2 工作时，能提供一个控制信号，用来作为CPU的中断请求。 INTE触发器定义如下：

INTE=1 允许中断

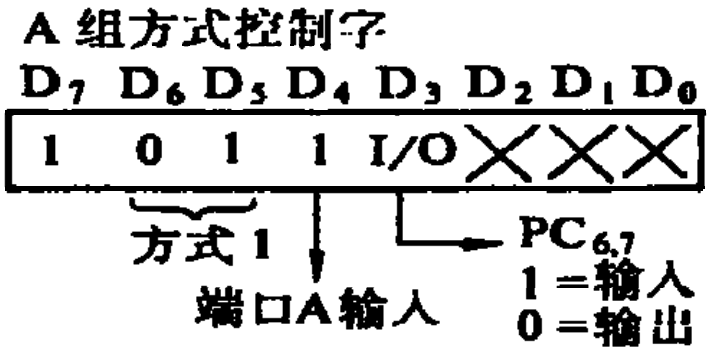
INTE=0 禁止中断

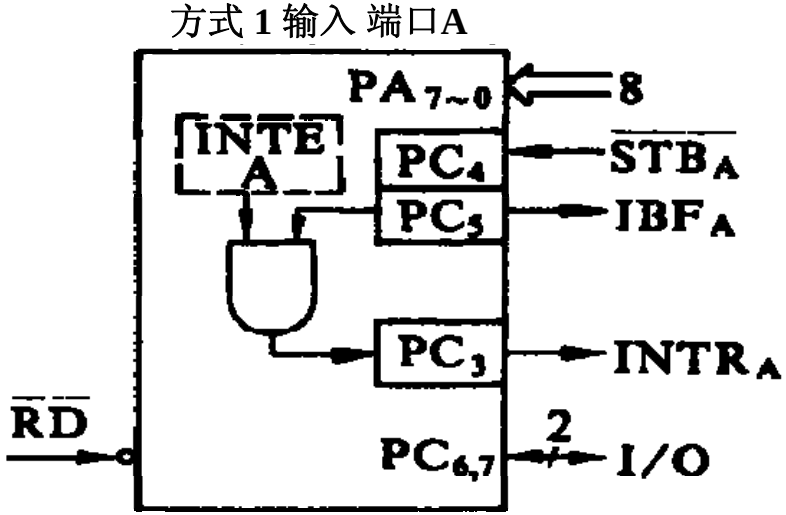
8255A规定：在方式 1 端口A 输入时，INTEA由PC_4（置位/复位）控制；

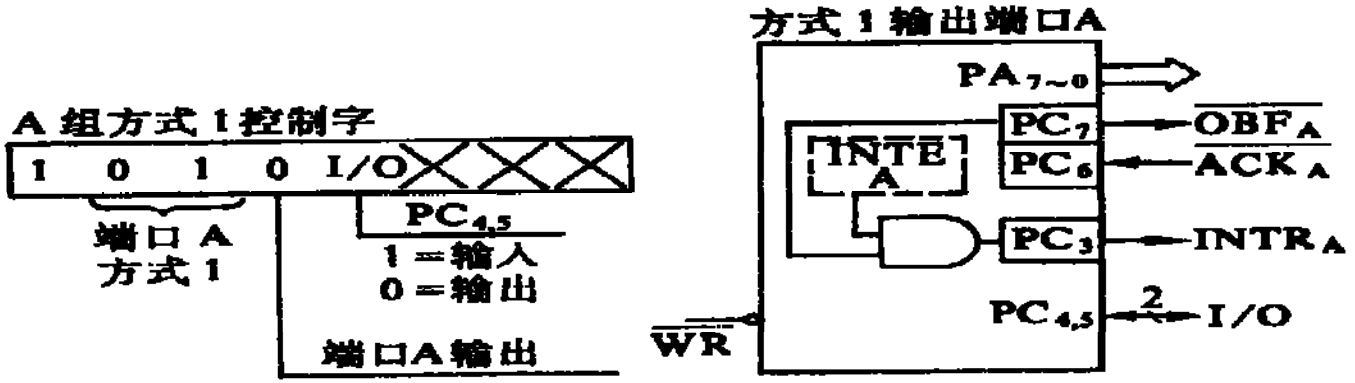
端口A 输出时，INTEA由PC_6（置位/复位）控制；

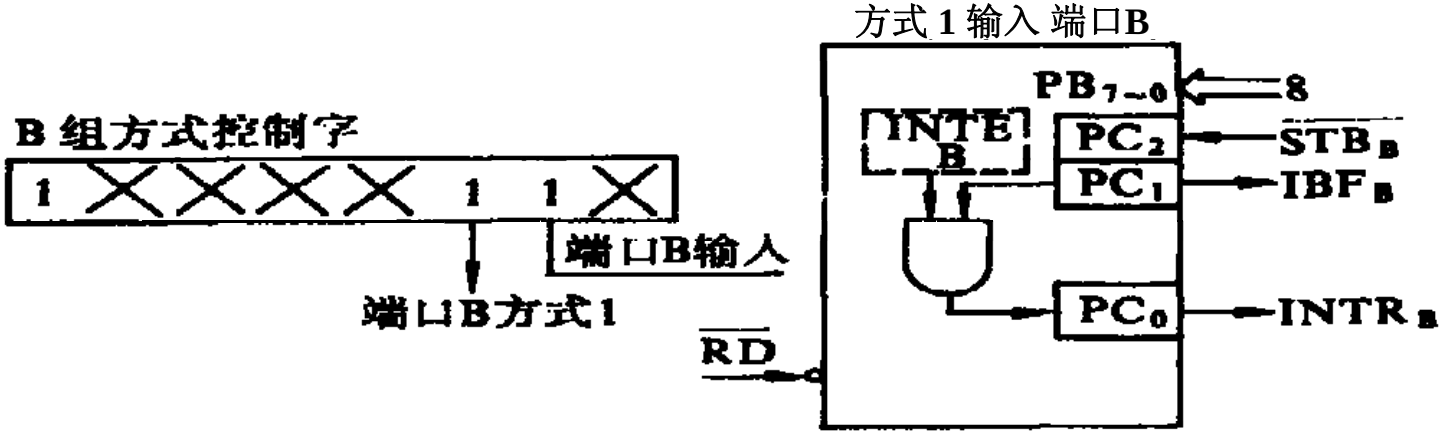
端口B 输入时，INTEB由PC_2（置位/复位）控制；

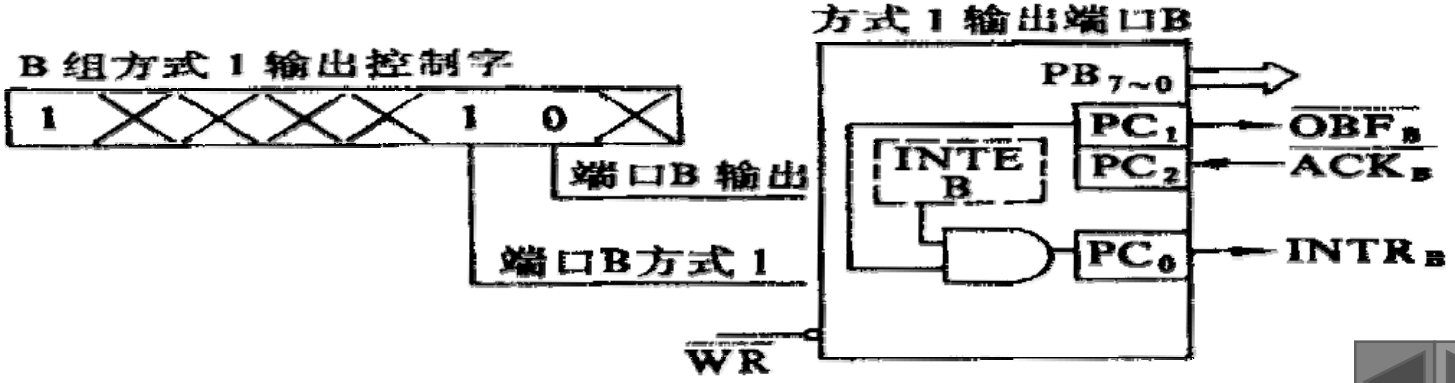
端口B 输出时， INTEB由PC_3（置位/复位）控制；





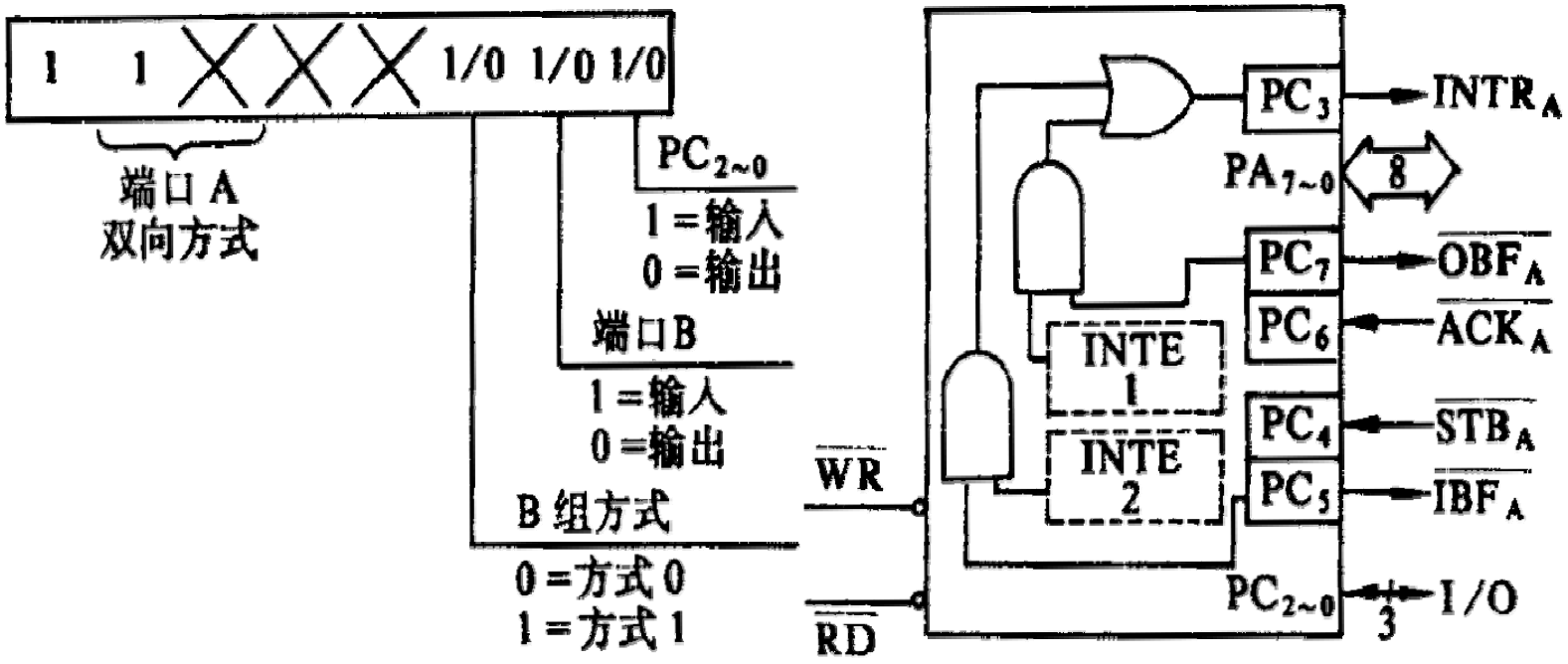

A 组方式 1 控制字
1 0 1 0 I/O
端口 A
方式 1
PC4,5
1＝输入
0＝输出
端口A输出
方式 1 输出端口A
PA7~0
INTE A
PC7
PC6
PC3
PC4,5
OBFA
ACKA
INTRA
2
I/O
WR


方式 1 输入 端口B
B 组方式控制字
1 1 1
端口B输入
端口B方式1
PB7~0
8
INTE B
PC2
PC1
PC0
STBB
IBFB
INTRB
RD


B 组方式 1 输出控制字
1 1 0
端口B 输出
端口B方式 1
方式 1 输出端口B
PB7~0
INTE B
PC1
PC2
PC0
OBFB
ACKB
INTRB
WR

在方式 2 输入时，INTE2由PC_4（置位/复位）控制；

输出时， INTE1由PC_6（置位/复位）控制；



* 但是，8255不能提供中断向量，可通过软件方式，或利用8259 解决。

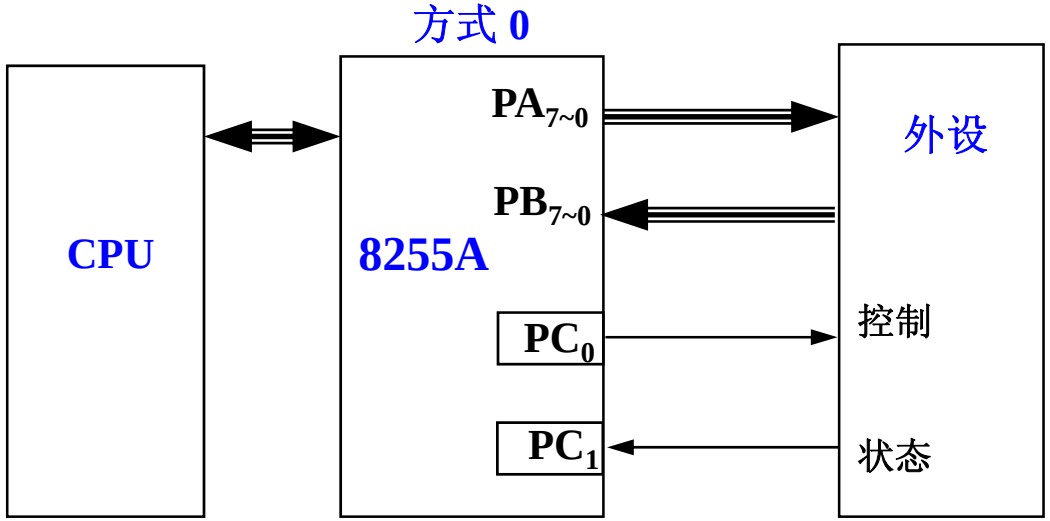
5. 8255A的端口的工作过程

(1) 方式 0

方式 0 是一种基本的输入或输出方式。

- 三个端口的每一个都可由程序选定作为输出或输入；
- 没有固定的用于应答的联络信号。

故方式 0：
- 可用于无条件传送的接口电路；
- 也可用于查询式输入输出接口电路，此时，只需将端口C（也可用端口A、端口B）的某些位作为两个数据端口的控制或状态信息。



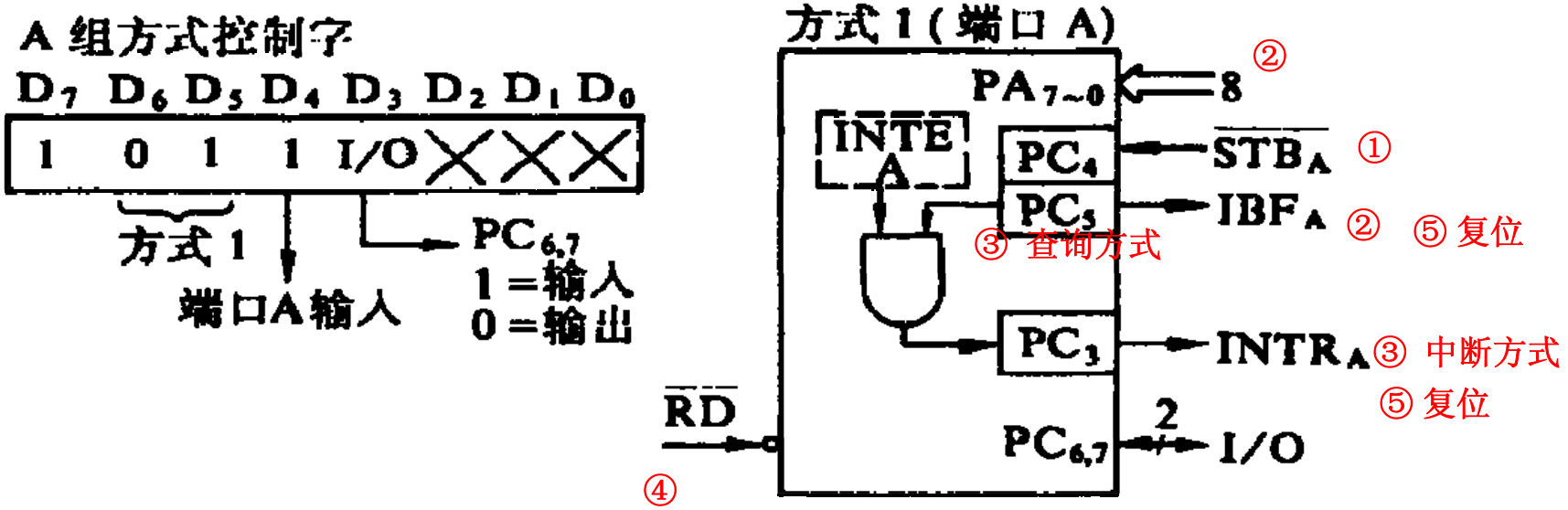
(2) 方式 1

方式 1 是一种选通方式。端口A 和端口B 仍作为输入输出，端口C的某些位作为控制和状态信息。

方式 1 ： • 可用作查询式输入\ 输出接口电路；

• 也可用于中断式输入\ 输出接口电路。

A口输入：



① 外设请求发送数据，发出请求信号 $\overline{\text{STB}}$；

② 8255A在 $\overline{\text{STB}}$ 下降沿将输入的数据锁存，同时将输入缓冲器满信号置 IBF 置 1，告诉外设暂缓送数；

③ 8255内设中断允许触发器

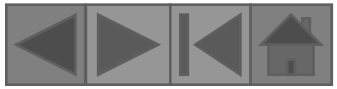
INTEA 置 1， 允许中断——即采用中断方式传送数据。此时中断请求信号通过 PC_3输出，向CPU发出中断请求信号。

INTEA 置 0， 不允许中断——即采用查询方式传送数据，CPU不断查询PC_5（IBF信号）。

④ • 若采用中断方式，CPU接收中断。发出读信号 $\overline{\text{RD}}$，使 INTR 复位；

• 若采用查询方式，CPU检查 IBF信号为1，数据准备好，则发出读信号 $\overline{\text{RD}}$ 。

⑤ CPU读结束，$\overline{\text{RD}}$ 上升，IBF 复位，允许外设送下一个数据。



B口输出：

① • 若采用中断传送方式：CPU接受中断请求，在 $\overline{WR}$ 信号的下降沿将输出的数据送入8255的相应口锁存，并经中断请求信号复位。

• 若采用查询方式：CPU检测到状态信号（$\overline{ACK}$）为Ready，在 $\overline{WR}$ 信号的下降沿将输出的数据送入 8255的相应口锁存。

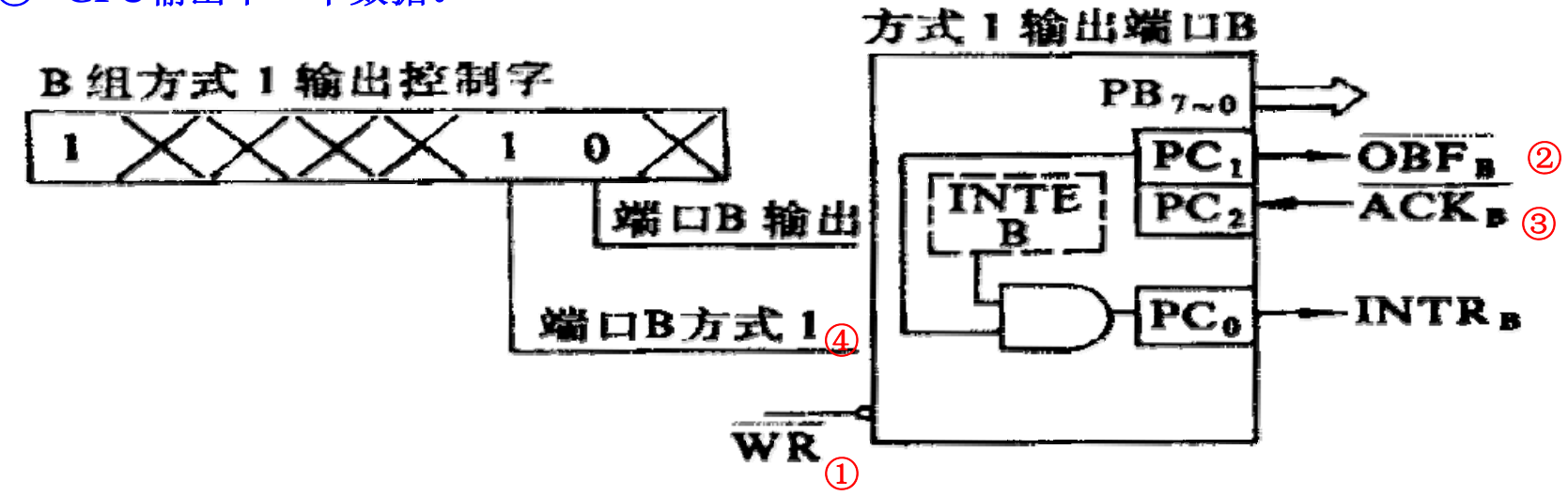
② 当CPU传送结束，则$\overline{WR}$上升沿将$\overline{OBF}$置为 0 ，表示输出缓冲器满，外设可接收数据。

③ 外设接收数据，将$\overline{ACK}$信号置为 0（有效），该信号：

• 使 $\overline{OBF}$ 无效（置 1）。

• 若采用中断方式 使 INTR 有效（若此时 INTE 置 1，而$\overline{OBF}$ 也为 1 。） 向CPU发出请求。

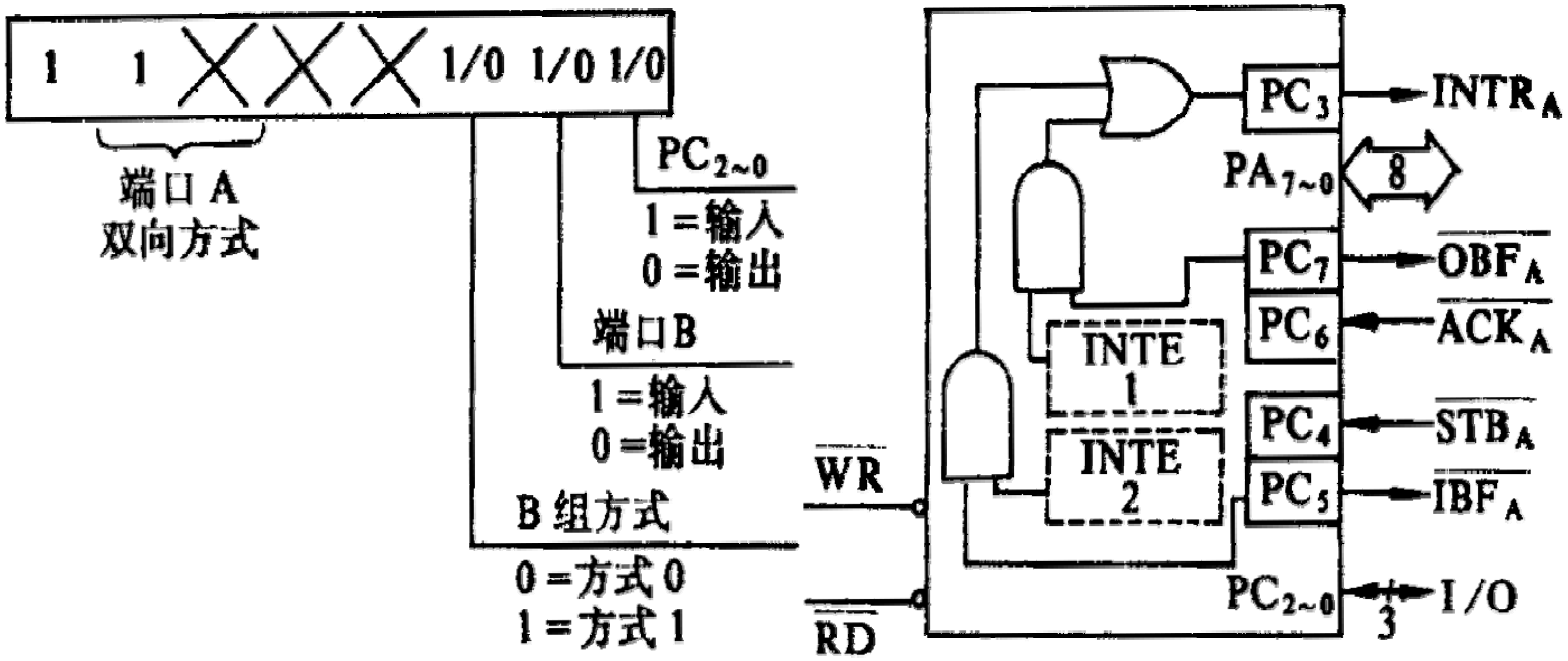
④ CPU输出下一个数据。



(3) 方式2

选同双向输入输出。此模式置限于A口使用。

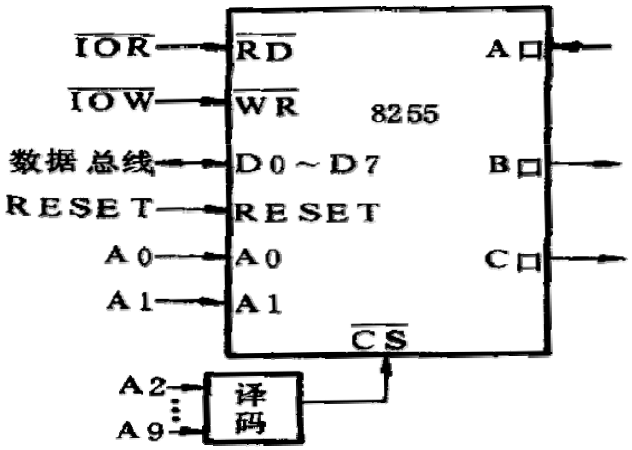
方式2：
- 可用作查询式输入\ 输出接口电路；
- 也可用于中断式输入\ 输出接口电路。

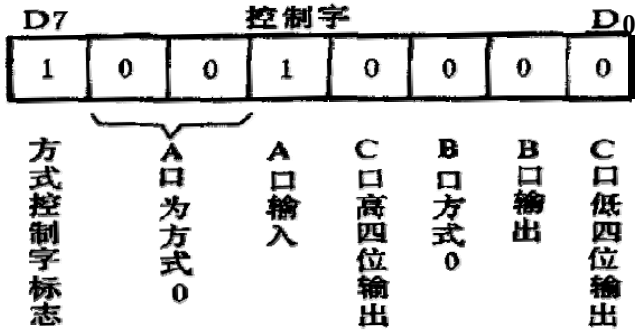


6. 应用举例

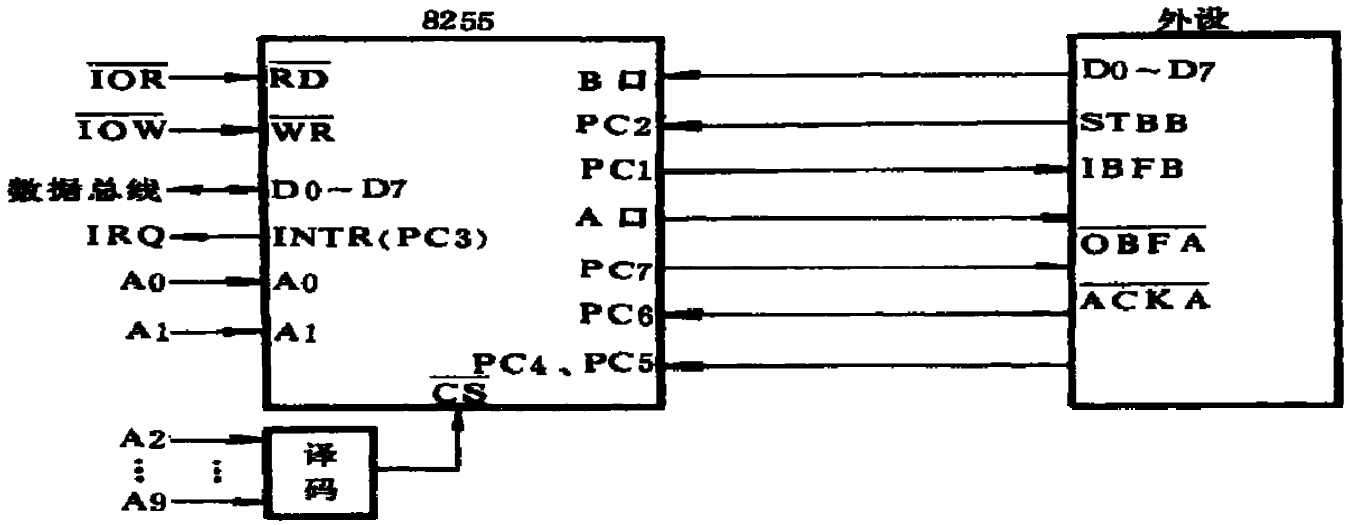
例1：在一系统中，要求8255端口地址为60H-63H且工作在方式0，A口为输入，B口、C口为输出。

```
Mov  al, 90h
Out  63h, al          ;送控制字到控制字寄存器。
Call  delay1
In  al,  60h          ; 从A口输入数据
Call  delay2
mov  al, data1
Out  61h, al          ; 从B口输出数据
Call  delay3
mov  al, data2
Out  62h, al          ; 从C口输出数据
```



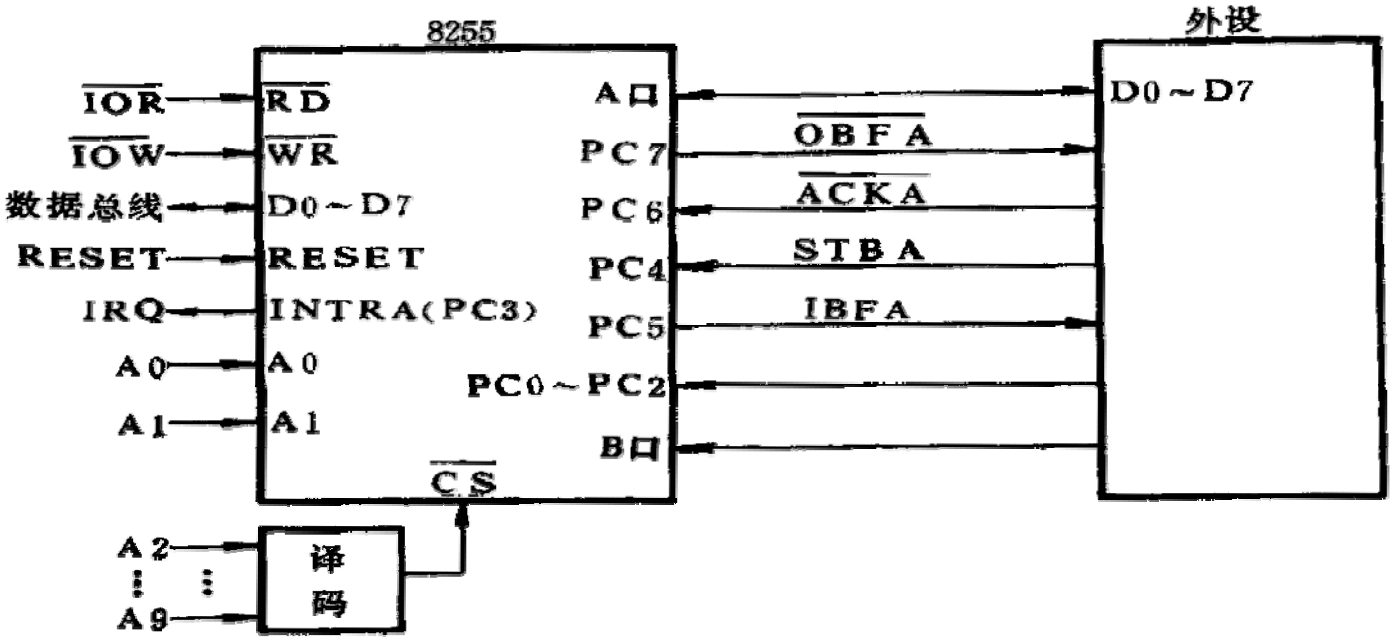


例2：假定在一个系统中，要求8255工作在方式1，端口A为输出，端口B为输入，PC_4~PC_5为输入，禁止端口B中断。




```
Mov  al, 0afh          ;控制字
Mov  dx, xxxxxx11b     ; 控制寄存器地址
Out   dx, al           ; 送入控制字寄存器
Mov  al, 09h           ; A口的INTE（PC4）置1
Out  dx, al            ; 送入控制字寄存器
Mov  al, 04h           ; B口的INTE（PC2）置0
Out  dx, al            ; 送入控制字寄存器
```

例3：假定在一个系统中，端口A工作在方式2，端口B工作在方式0且为输入，端口C的三位 PC_0~PC_2 位输入。




```
Mov  al, 11xxx011b        ;控制字
Mov  dx, xxxxxx11b        ; 控制寄存器地址
Out   dx, al              ; 送入控制字寄存器
```

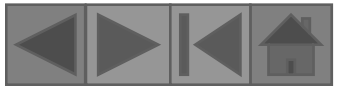
三、IBM PC/XT中 8255A—5 的使用

在 IBM PC/XT中，8255A-5工作在无应答联络信号的基本输入输出方式(方式0)下.

1. 在加电后系统自测试时，CPU通过PA口输出部件检测标志，如果检测到关键性故障停机时，测量PA口的输出电平，可以确定发生故障的部件。

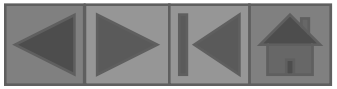
   PA口输出的部件检测标志主要有以下几种：

PA_1	PA_2	PA_0	
0	0	1	BIOS累加和错
0	1	0	8253—5 错
0	1	1	8237A—5错
1	0	0	前16KROM错
1	0	1	8259错
1	1	0	CRT适配器错

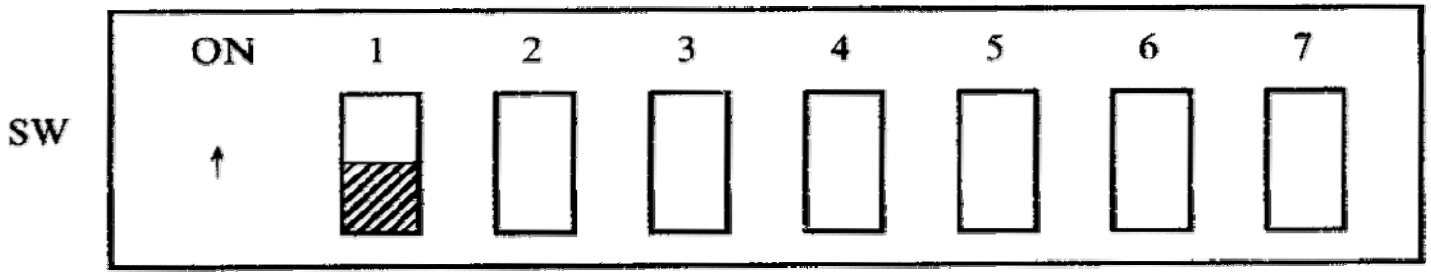


十六进制I/O口地址		功能			
60	输入或输出	PA 0	键盘扫描码 0	或	诊断输出 0
		1	1		1
		2	2		2
		3	3		3
		4	4		4
		5	5		5
		6	6		6
		7	7		7
61	输出	PB 0	+ 定时器2选通端控制		
		1	+ 扬声器数据		
		2	备用		
		3	− 允许读SW−1～SW−4/+允许读SW−5～SW−8		
		4	− 允许RAM奇偶校验		
		5	− 允许I/O通道校验		
		6	− 保持键盘时钟低电平		
		7	− 允许键盘数据/+清除键盘数据		
62	输入	PC 0	+ 正常工作 (SW−1)		显示类型0 (SW−5)
		1	+ 插入8087 (SW−2)	或	显示类型1 (SW−6)
		2	+ 板上RAM容量0 (SW−3)		$5\frac{1}{4}$英寸驱动器数0(SW−7)
		3	+ 板上RAM容量1 (SW−4)		$5\frac{1}{4}$英寸驱动器数0(SW−8)
		4	+ 扬声器状态		
		5	+ 定时器通道2输出		
		6	+ I/O通道校验		
		7	+ RAM奇偶校验		
63	命令寄存器: 加电自检时为89H(A口输出,B口输出,C口输入) 正常工作时为99H(A口输入,B口输出,C口输入)				

8255在
正常工作
时的功能

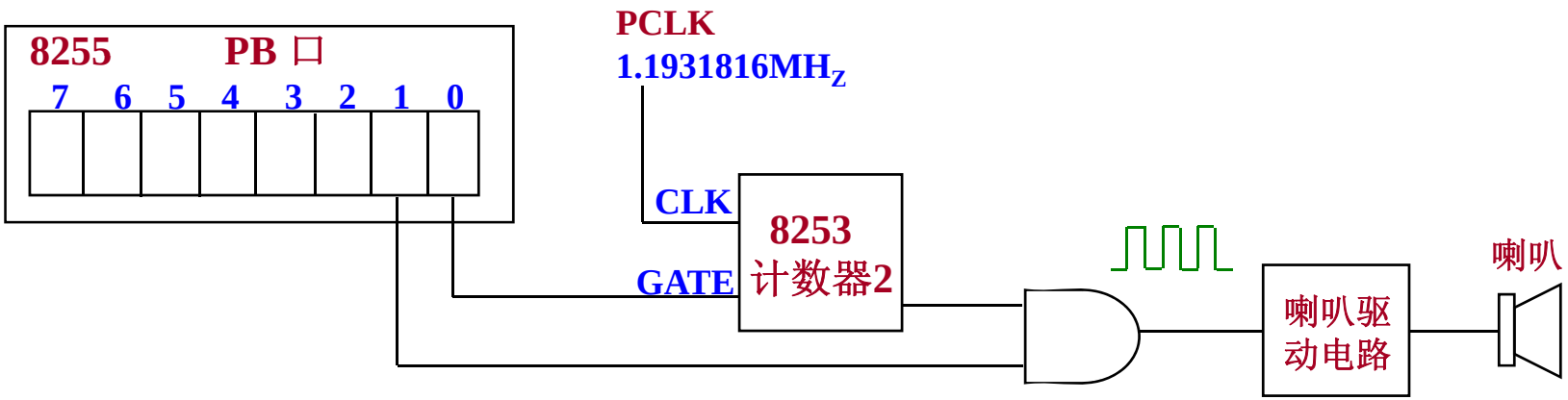


2. 系统配置开关



开关信号	功能
1	OFF——正常工作
	ON——循环执行加电自检
2	OFF——插入8087
	ON——未使用8087
3和4	系统板上RAM容量
5和6	当前使用的显示适配器类型
7和8	连接的软盘驱动器数量

3. 喇叭接口



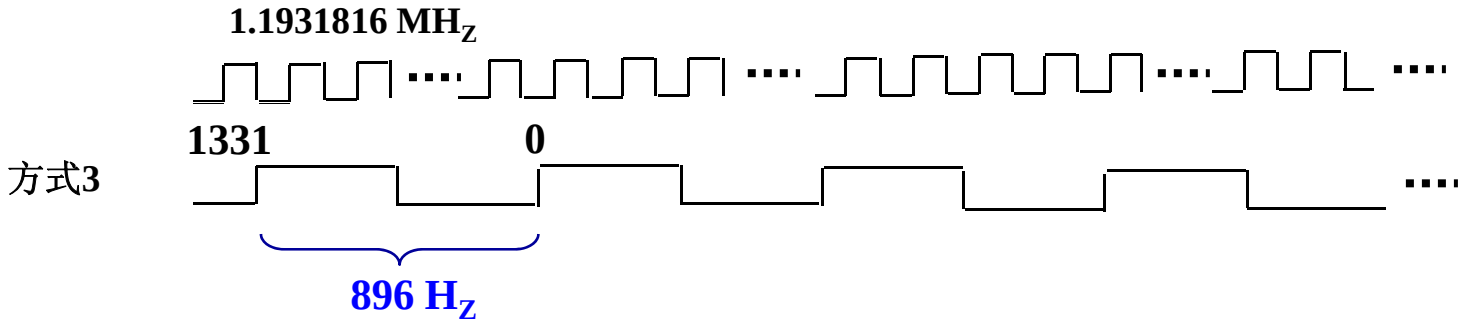
- 由系统并行接口芯片8255的PB口的最低两位:

  PB_0（TIM2GATESPK)

  PB_1（SPKRDATA)　　控制喇叭发声。

- 定时计数器8253工作在方式3下，预置初值为 533H（1331D)，

  输出频率为：1.1931816 MH_Z / 1331 = 896 H_Z



(1) 用程序控制发声

系统中，喇叭只产生 **896H_Z** 的音调。如果想改变这个音调，使喇叭产生任意频率的音调，该怎么做？

- 控制发声频率 —— 改变计数初值

  计数初值 = **1.1931816 MH_Z** / 给定频率

  = **1234dcH** / 给定频率

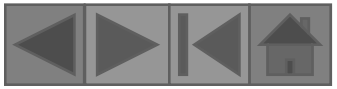
- 控制声音长短 —— 延时

  ① 用**BIOS**功能调用 **INT 1AH**；（由于**8253**计数器**0 55ms**申请一次中断，所以此方法实现不了任意时间的延时）。

  ② 改变 **8253** 计数器**0** 的计数初值，使其可以以任意时间申请中断；(太复杂)。

  ③ 延时子程序 ；

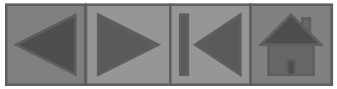
综上所述，我们得到了使喇叭发声的频率和时间，因此，就可以控制喇叭发声了。



下面我们看一个喇叭发声过程子程序。

声音频率在 **di** 寄存器中；发声时间在**bx**中（几个**10ms**）。

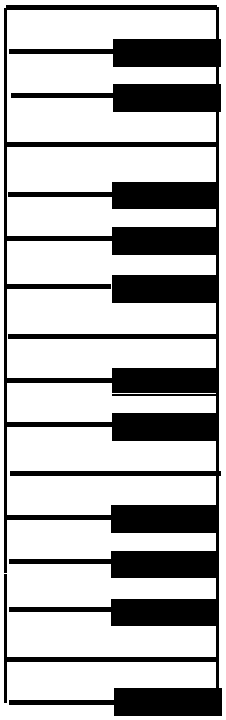
```
Sound   proc  far
        push  ax
        push  dx
        push  di
        push  bx
        push  cx
        mov  al, 0b6h        ; 8253控制字：通道2，方式3，
        out  43h, al         ;对计数器两次写操作，二进制计数
        mov  dx, 12h
        mov  ax, 34dch
        div  di
        out  42h, al
        mov  al, ah
        out  42h, al         ;计数初值送入计数器2
        in  al, 61h
        mov  ah, al          ;保存原值
        or  al, 3
        out  61h, al         ;将8255PB口最低两位置1
Delay:  mov  cx, 2801
DL10ms: loop  DL10ms
        dec   bx
        jnz  delay
        mov  al, ah
        out  61h, al     ;恢复原值
        pop  cx
        pop  bx
        pop  di
        pop dx
        pop  ax
        ret
Sound   ends
```



(2) 用喇叭凑乐。

1小节2拍，800ms

C调 2/4 | 3 3 3 5 | 3 3 2 | 1 6 1 2 | 3 —— | ……

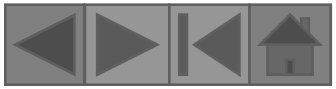


	(C)	130.8	
	(D)	146.8	
	(E)	164.8	
	(F)	174.6	
	(G)	190.0	
	(A)	220.0	
	(B)	246.9	
（中音）	(C)	261.7	
	(D)	293.7	
	(E)	329.6	
	(F)	349.2	
	(G)	392.0	
	(A)	440.0	
	(B)	493.9	
	(C)	523.3	

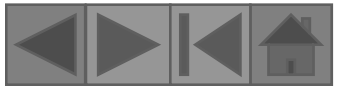
138.6 (C#, Db)
155.6 (D #, Eb)
185.0 (F #, Gb)
207.7 (G #, Ab)
233.1 (A #, Bb)
277.2 (C #, Db)
311.1 (D #, Eb)
370.0 (F #, Gb)
415.3 (G #, Ab)
466.2 (A #, Bb)

发声频率 sog_f dw 330, 330, 330, 392, 330, 330, 294, 262, 220, 262, 294, 330, 0

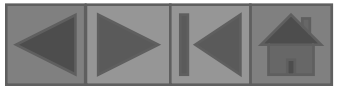
发声时间 sog_t dw 20, 20, 20, 20, 40, 20, 20, 20, 20, 20, 20, 80, 0



```
Stack    segment
         db  100 dup(?)
Stack    ends
Data     segment
Sog_f    dw   330, 330, 330, 392, 330, 330, 294, 262, 220, 262, 294, 330, 0
Sog_t    dw   20, 20, 20, 20, 40, 20, 20, 20, 20, 20, 20, 80, 0
Data     ends
Code     segment
         assume   cs:code, ds: data, ss: stack
Sign   proc  far
         push  ds
         xor  ax, ax
         push  ax
         mov  ax, data
         mov  ds, ax
         lea  si, sog_f
         lea  bp, ds:sog_t            ;设置数据
         call  play                   ;调用演奏子程序
         ret
Sign    endp
```



```
Play      proc
          push  bx
          push  di
          push  si
          push  bp
Freq:     mov  di, [si]          ;送频率信号
          cmp  di, 0
          je  end_play
          mov  bx, ds:[bp]       ;送发声时间
          call  sound            ;调用发声子程序
          add  si, 2
          add  bp, 2
          jmp  freq
End_play: pop  bp
          pop  si
          pop  di
          pop  bx
          ret
Play      ensp
code      ends
          end  sign
```



4. 键盘接口

(1) 键盘工作原理

键盘是由一组排列成矩阵方式的按键开关组成。通常分成全编码键盘和非编码键盘两类。

- 全编码键盘—— 对每一个按键，通过全编码电路产生唯一对应的编码信息（如**ASC Ⅱ码**）。

  优点：响应速度快；

  缺点：硬件结构复杂，且复杂性随着键数的增加而增加。

- 非编码键盘—— 不直接提供按键的编码信息，而是利用简单的硬件和一套专用的键盘程序来识别按下键的位置（即提供位置码，或叫扫描码）。（以上由键盘电路实现）。

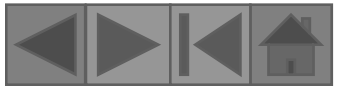
  然后由处理机将位置码通过查表程序转换成相应的编码信息（如**ASC Ⅱ码**）。

  （以上由键盘接口电路及**BIOS**键盘处理程序实现）。

  优点：可通过软件编码重新定义键盘的某些减。

  缺点：速度不如全编码键盘。

IBM PC/XT 采用非编码键盘。

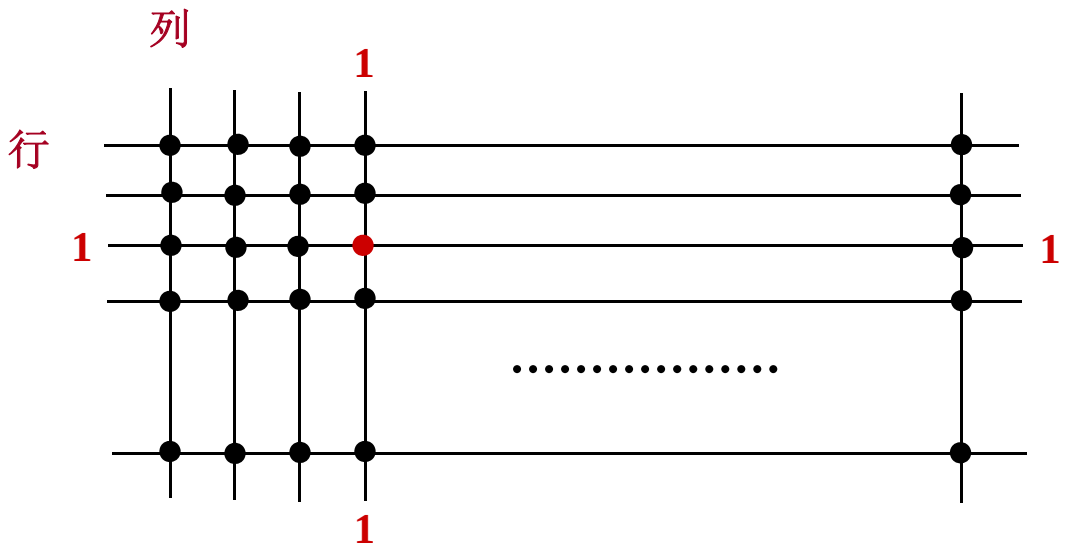


非编码键盘

对非编码键盘， 我们怎样确定按下键的位置呢？

—— 可采用行扫描法；行列扫描法；行反转法；

IBM PC/XT 采用行列扫描法。



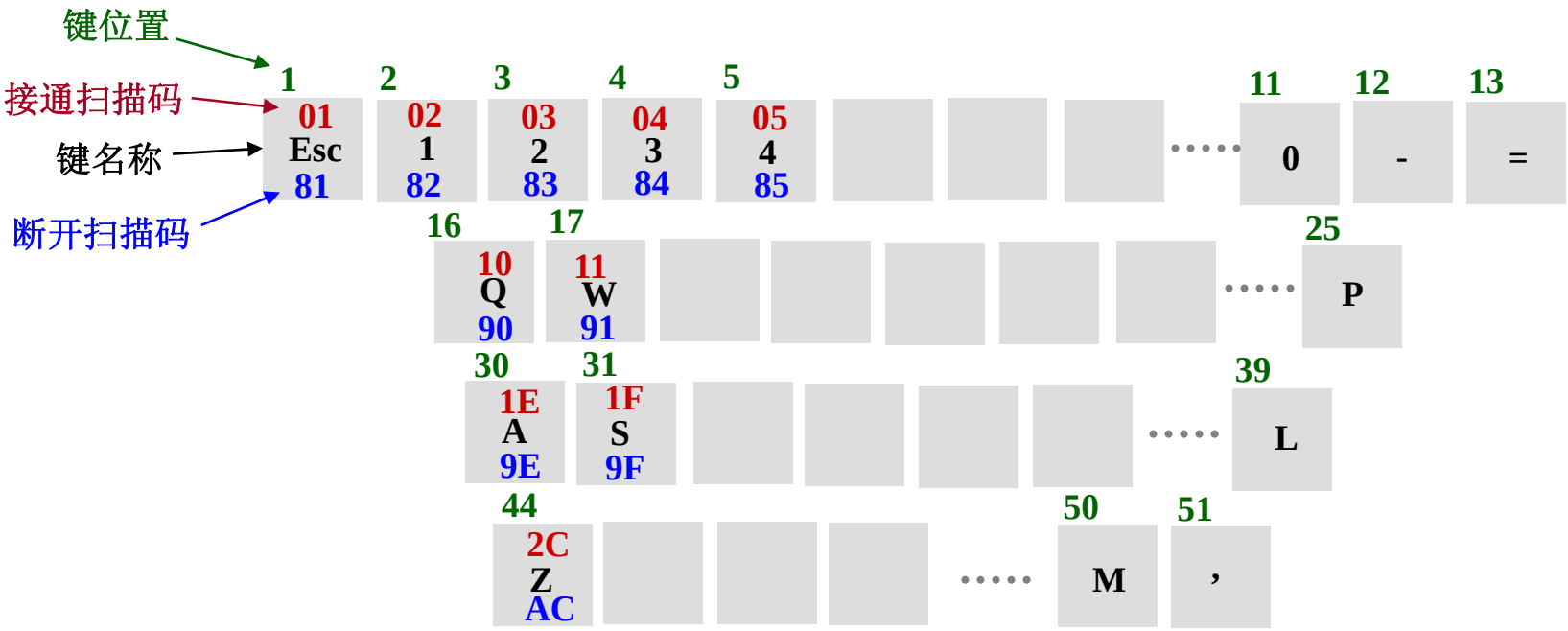
方法： • 先在列线上加步进的 “1” 信号，依次检查哪一列上有键按动。

（向列扫描线加 “1” 信号，若 行接收线由按动的键时，能从交点上获得“1”信号，否则为“0”。

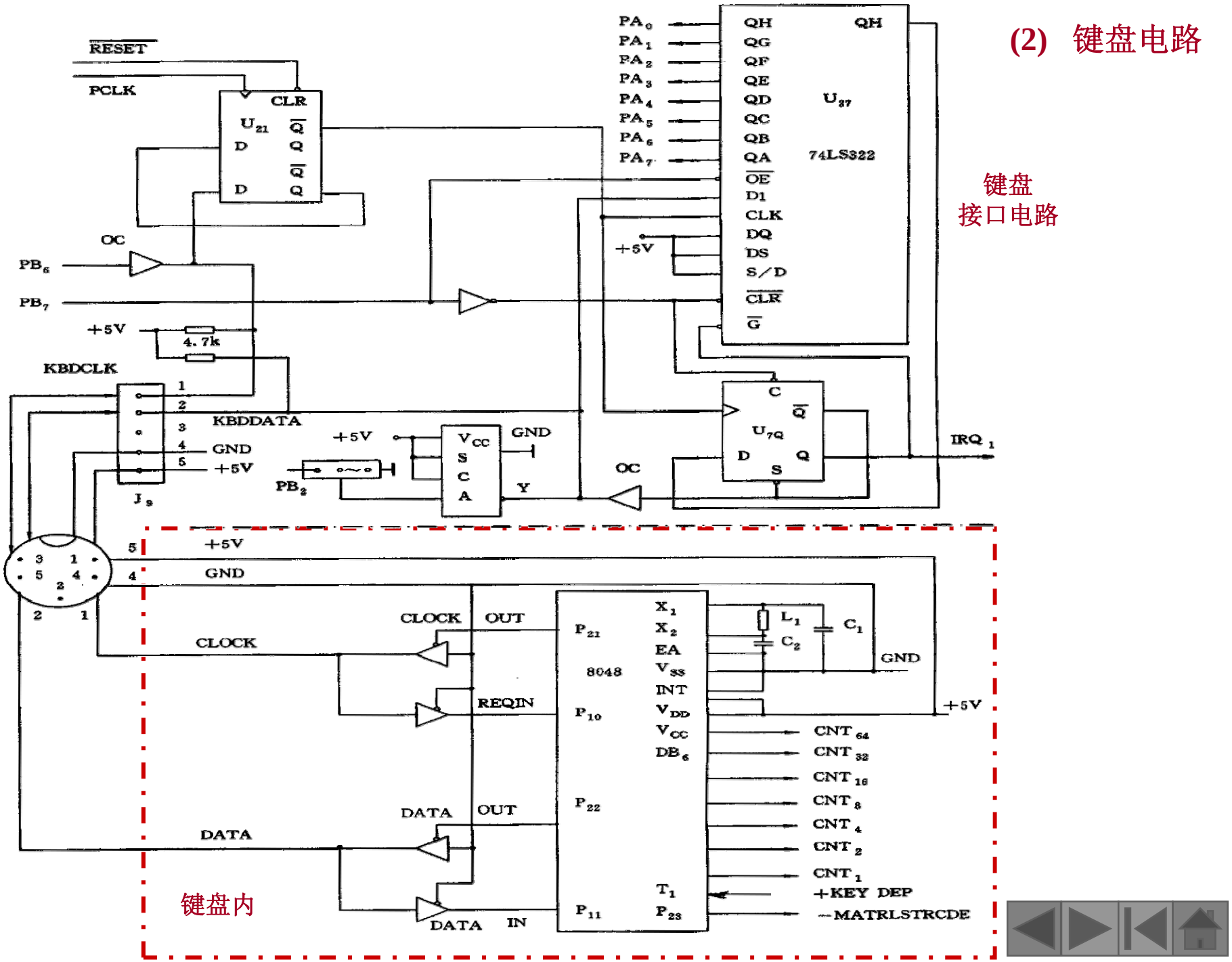
• 然后再在航线上加步进的“1”信号，…….

• 根据两者检查的结果确定被按键的位置。

IBM PC/XT键盘位置及对应的扫描码。



(2) 键盘电路

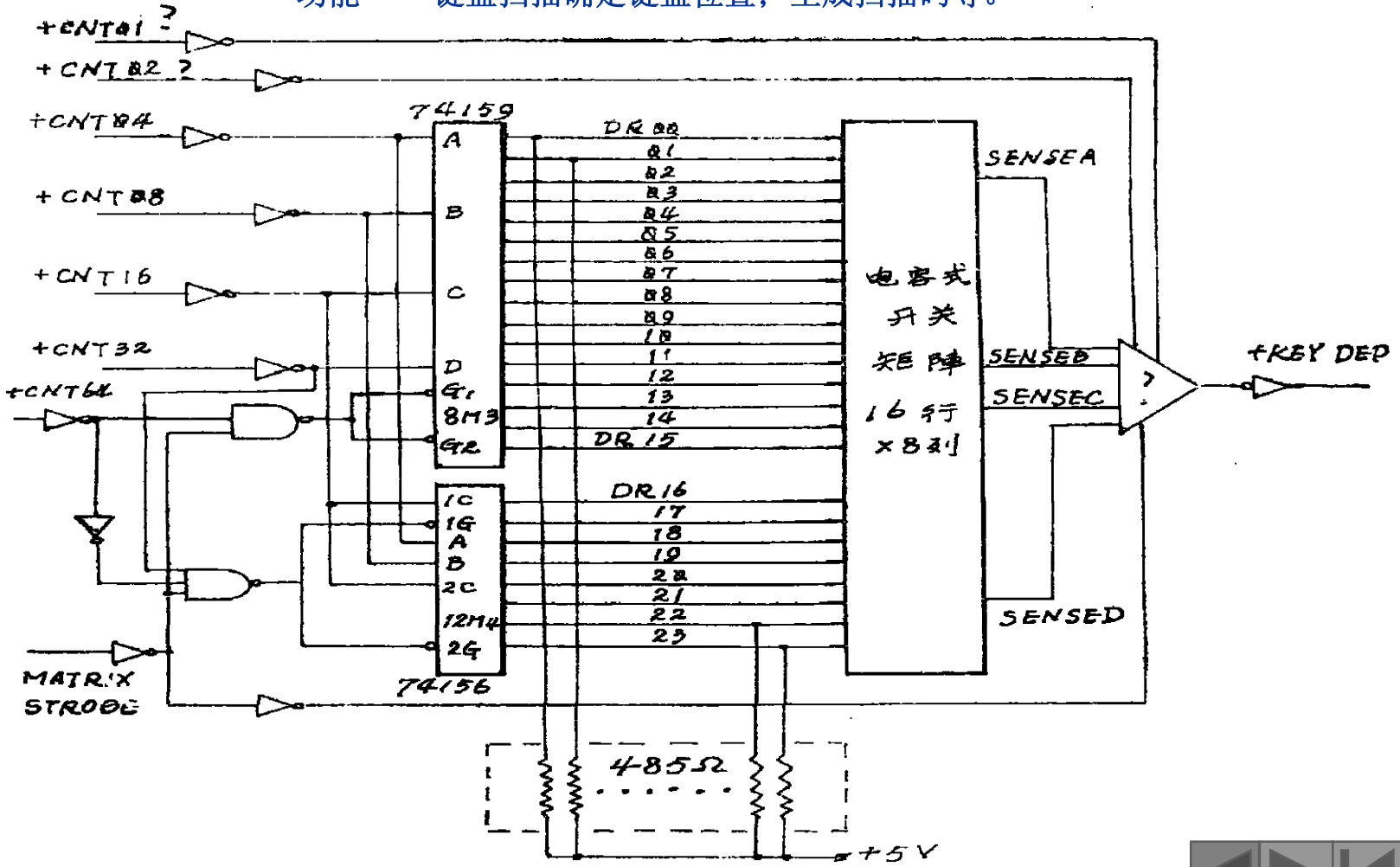

RESET
PCLK
CLR
U_{21}
D
Q
$\overline{Q}$
OC
PB_6
PB_7
+5V
4.7k
KBDCLK
KBDDATA
GND
+5V
J_9
PB_2
V_{CC}
S
C
A
Y
PA_0
PA_1
PA_2
PA_3
PA_4
PA_5
PA_6
PA_7
QH
QG
QF
QE
QD
QC
QB
QA
$\overline{OE}$
D1
CLK
DQ
$\overline{DS}$
S/D
$\overline{CLR}$
$\overline{G}$
U_{27}
74LS322
U_{7Q}
C
D
S
IRQ_1
键盘
接口电路
键盘内
CLOCK
OUT
REQIN
DATA
IN
8048
P_{21}
P_{10}
P_{22}
P_{11}
X_1
X_2
EA
V_{SS}
INT
V_{DD}
V_{CC}
DB_6
T_1
P_{23}
L_1
C_1
C_2
CNT_{64}
CNT_{32}
CNT_{16}
CNT_8
CNT_4
CNT_2
CNT_1
+KEY DEP
—MATRLSTRCDE

工作过程：

- **8048** 送出计数信号CNT_1~CNT_{64}到键盘扫描电路，通过行译码器给**16 x 8**的开关矩阵的行列线加步进信号，扫描键盘。

（**8048**单片机：组成—— **8**位**CPU**、**RAM**、**ROM**、定时/计数器等；

功能—— 键盘扫描确定键盘位置；生成扫描码等。



- 每次行、列扫描结果由 Key DEP 线送到 8048 的条件测试端T_1, 8048 根据这一结果确定键盘上按键的位置。
- 8048 确定按键的位置后，由 P_{22}口串行地输出扫描码，通过电缆插座的脚2和 J_9 的脚2送入键盘接口电路（在主板上）的右移寄存器U_{27}的数据接收端D_1。

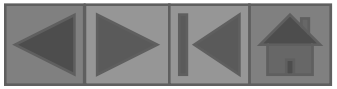
  （应先使 PB_7 =1，将LS322 IRQ_1复位，

  再使 PB_7=0，使LS322 处于工作状态； PB_6=1，保证时钟通道工作。
- U_{27}是带符号扩展端的8位移位寄存器，它的主要功能是从数据输入端D_1每次接收一位数据，其各位依次右移到下一位， 即：

  QA ⟶ QB ⟶ QC ⟶ ····· ⟶ QH ⟶ QH’

  9个周期后，由 8284 送来的 8位数据依次右移，并保存到 QH~QA寄存器中；

  标志码通过QH’ 端送入中断请求触发器U_{70}的D端，使触发器置1，产生键盘中断请求信号IRQ_1。
- CPU接收中断请求后，由键盘中断处理程序通过对8255端口A的访问，读到扫描码，并对扫描码进行识别和相应的转换，把转换后的ASCⅡ字符送入键盘缓冲区。



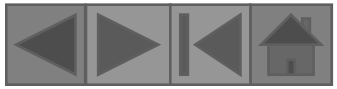
(3) 键盘中断处理程序

如上所述，当我们在键盘上“按下” 或 “放开” 一个键时，如果键盘中断是允许的，就产生一个类型 9 的中断，CPU 即转入 BIOS 的键盘中断处理程序。

该键盘中断处理程序名为 **KB_INT**，入口地址为 **F000:E987**；

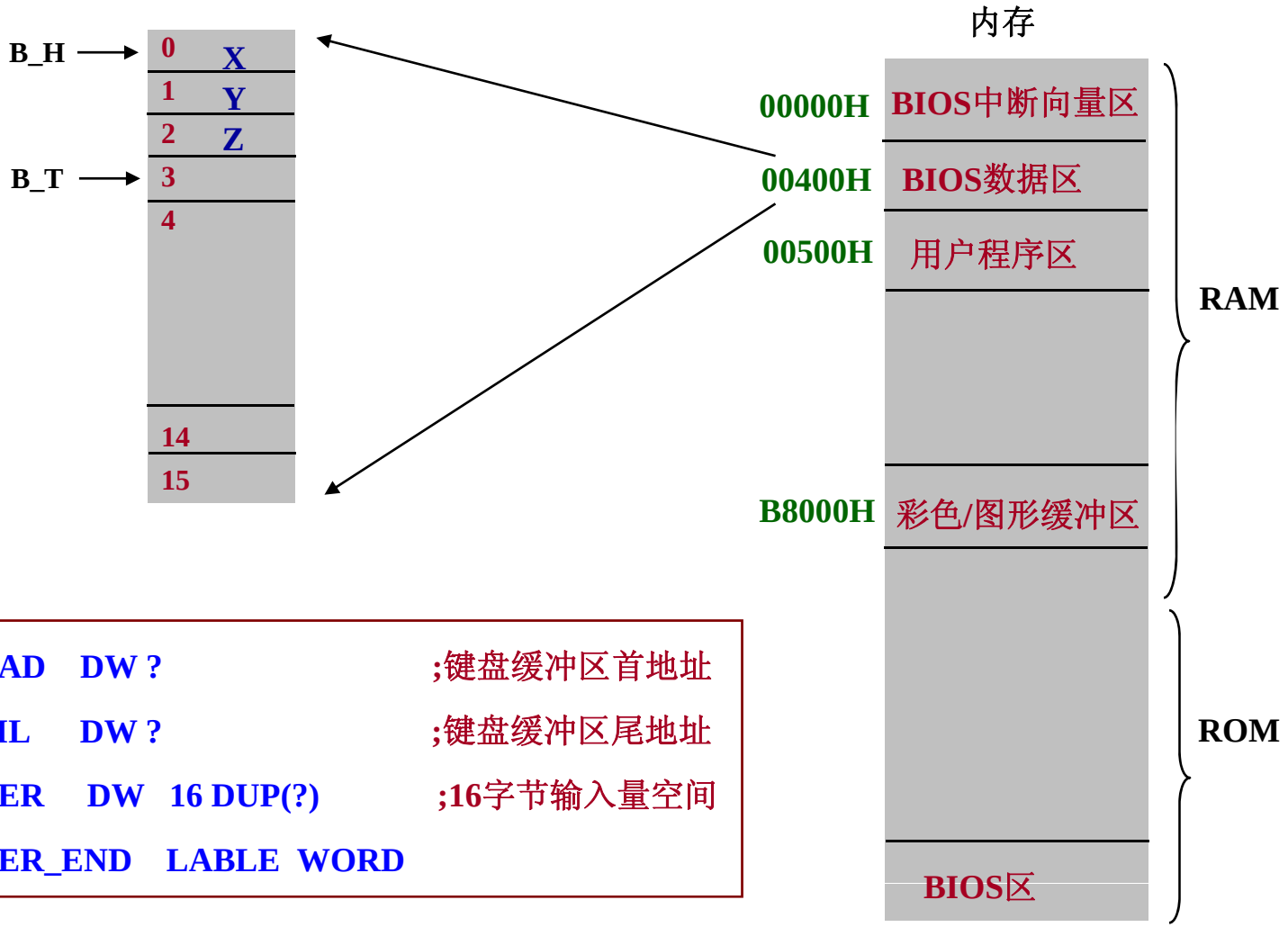
其功能 是读取扫描码，转换成相应的ASCⅡ，并将ASCⅡ（低8位）及其扫描码（高8 位）送入键盘缓冲区。

（没有相应的ASCⅡ的键，如：ALT、F_1～F_{10} 等，字符码存为 0 。）



(4) 键盘缓冲区

键盘缓冲区是一个先进先出的循环队列，位于内存的高端，BIOS数据区。




```
0040: 001A   BUFF_HEAD    DW ?                  ;键盘缓冲区首地址
0040: 001C   BUFF_TAIL    DW ?                  ;键盘缓冲区尾地址
0040: 001E   KB_BUFFER    DW  16 DUP(?)         ;16字节输入量空间
0040: 003E   KB_BUFFER_END   LABLE  WORD
```

(5) 键盘的 BIOS 功能调用与 DOS 功能调用

BIOS 功能调用：

INT 16H	**AH= 0**	**；从键盘读入一个字符**
	AH=1	**；读有无键入字符**
	AH=2	**；读特殊键状态**

DOS 功能调用：

INT 21H	**AH=1**	**；从键盘输入一个字符并回显**
	AH=6	
	AH=7	
	AH=8	
	AH=A	
	AH=B	
	AH=C