ELEG 3230B Microprocessors and Computer Systems

Part 9

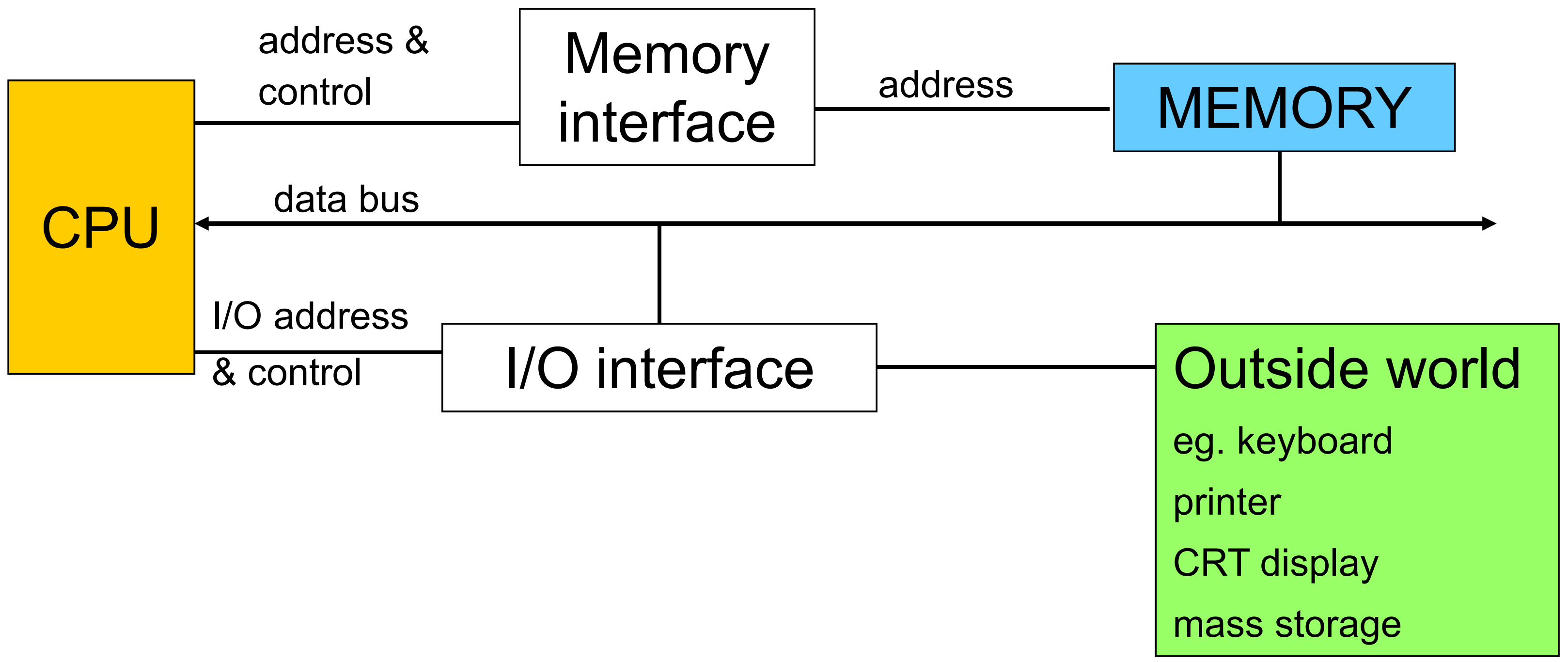
Basic Input & Output Interface

(Hall's Ch 9; Brey's Ch 10)

Outline

- Input-output Interface
- Serial/Parallel Interface
- 8255 Programmable Peripheral Interface
- Keyboard
- 8254 Software-Programmable Timer/Counter

Introduction to Input-Output



I/O interface

- Input-output involves the transfer to (or from) peripheral devices from (or to) the data bus of the cpu; eg. data transfer to the CRT display, from the keyboard, to/from a hard-disk drive, to/from a modem to another computer system.
- Input-output operations fall into one of the following types:
  - *Programmed I/O* - cpu polls peripherals to check if I/O is needed
  - *Interrupt I/O* - peripheral sends an interrupt request to cpu for I/O
  - *Direct memory access (DMA)* - peripheral writes directly to memory
- Considering the address of I/O, 8088 uses Isolated I/O (I/O addresses are not part of memory address) as distinct from Memory mapped I/O (peripherals are mapped to locations in the memory address space).

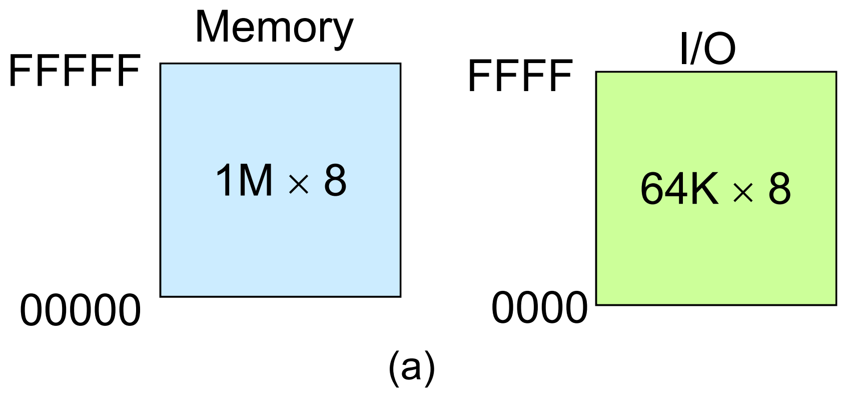
Q: pros and cons of isolated I/O and memory mapped I/O?

Memory and I/O Map

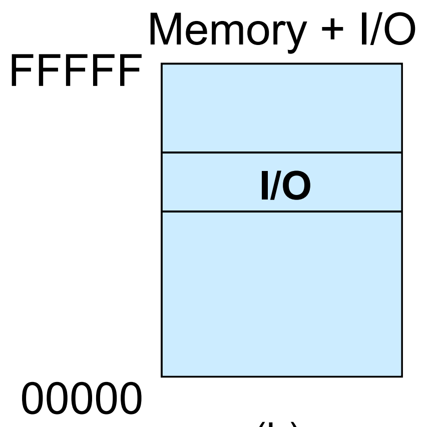
The Memory and I/O maps for the 8086/8088 microprocessor.

(a) Isolated I/O.

(b) Memory mapped I/O.



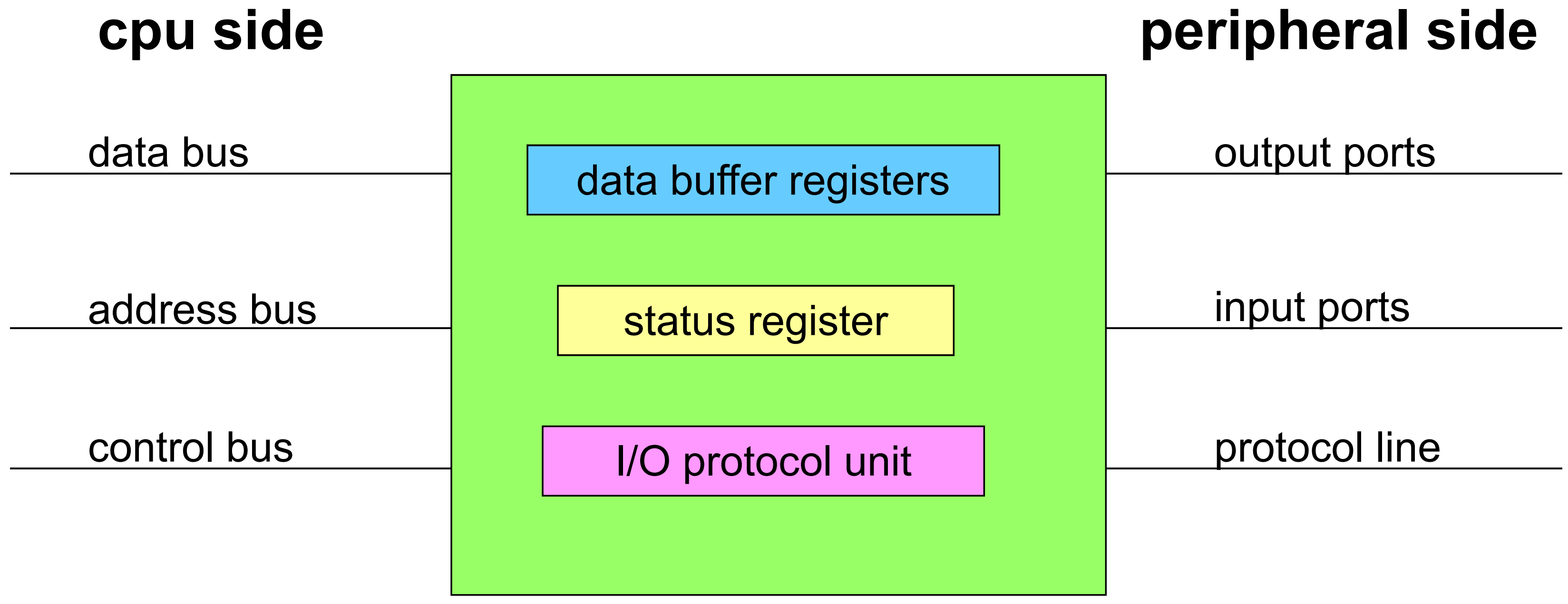
(a)



(b)

Q: In Isolated I/O, an I/O port has the same address as one of the memory byte. How does it know it is being addressed?

I/O interface



example of an I/O interface

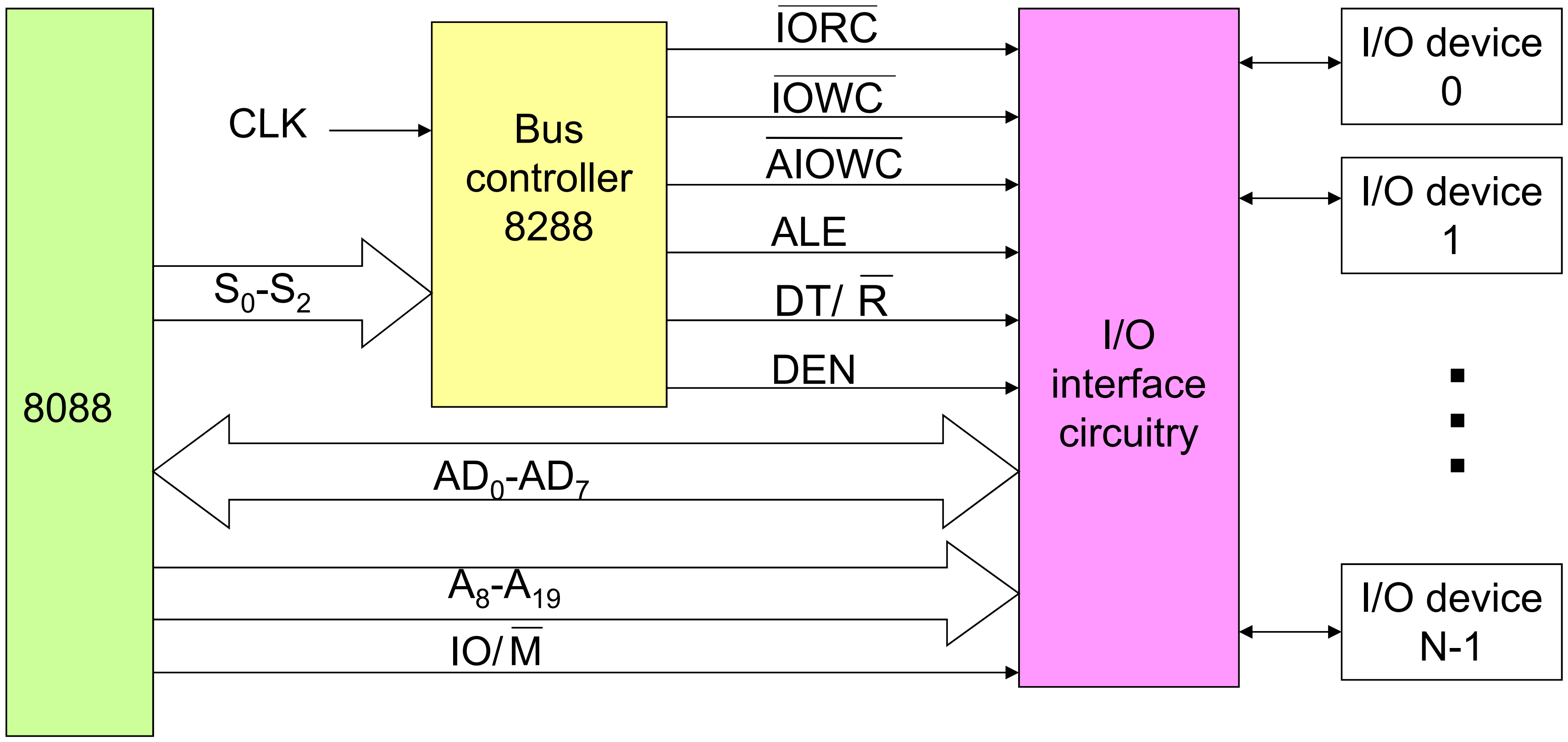
Typically, not all data, address or control lines are needed.
Input and output ports may be the same (shared).

I/O interface

- I/O interface functions include
  - *data storage buffer for sending and receiving data*
  - *low-level communications protocol (handshaking)*
  - *data format conversion (eg. parallel/serial)*
  - *error detection*
  - *addressing of different peripherals*

- I/O interface are typically implemented by LSI (large scale integration) - many different types are available from different manufacturers.
- Data can be transferred through I/O interface by either programmed I/O or interrupt I/O. DMA typically needs a separate controller.

8088 maximum mode I/O interface



Generalized I/O interface connections to maximum mode 8088

Parallel and Serial Data Transfer

- Data Transfer between the I/O interface and the peripheral can involve either parallel or serial data transmission, depending on the peripheral and the actual implementation of the I/O interface.

- Parallel data transfer involves using at least 8 separate lines for the 8 data bits in a byte. Normally, other lines are needed for the communications protocol (eg. STB [data strobe] line to indicate when data is valid, ACK line to acknowledge data has been read).

  STB is sent by sender   ACK is sent by receiver

- Parallel data transfer is usually faster. Each data bit typically needs its own ground return line to reduce noise. A popular parallel data transfer interface standard is the CENTRONICS type interface which uses a 36-pin connector. The Centronics interface is commonly used in printers.

Parallel and Serial Data Transfer (cont.)

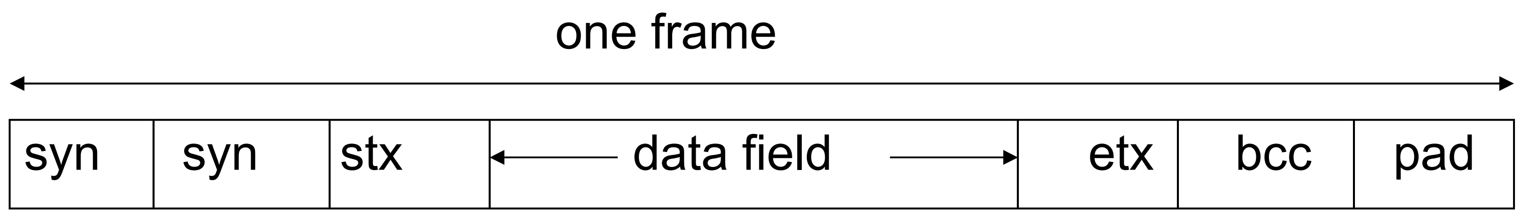
- Serial data transfer involves sending the data on a single line, bit by bit. The I/O interface converts the data from parallel to serial or vice-versa using shift registers.
- Serial connections are commonly used for data transfer over longer distances (eg over telephone line). A popular standard for serial data transmission is the RS232C standard.

Q: Why serial data transfer is more commonly used in long distance transmission?

Serial I/O

- Serial I/O can be either

  (1) Synchronous - data are sent in blocks, with *start* and *end-of-block* markers. Individual characters within a block do not need start and stop bits since the receiver identifies every 8 bits as one character, eg.



syn = sync character (ascii code 16)

stx = start of text (ascii code 02)

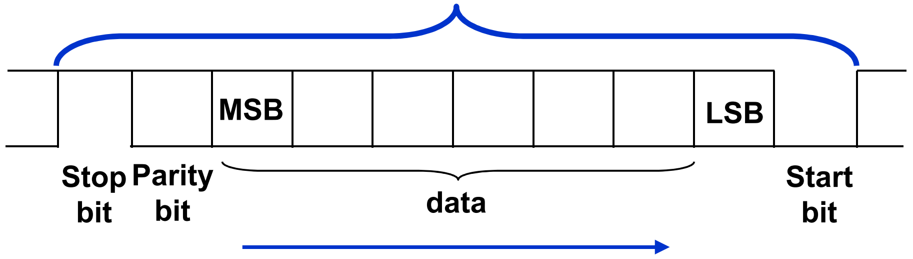
etx = end of text (ascii code 03)

bcc = block check characters (error detection)

pad = end of frame pad (ascii code ff)

Serial I/O (contd.)

(2) Asynchronous - no block synchronization bits. Each character is identified by the start and stop bit(s) (stop bits can be 1, 1.5, 2 bits) inserted at the start and end of each character.

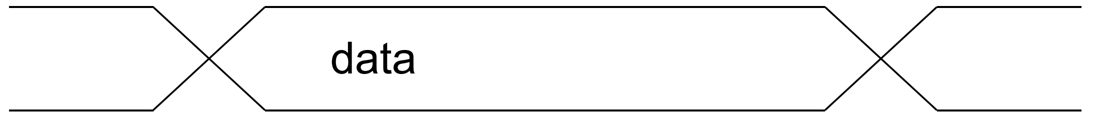


- Synchronous serial data transfer is more efficient (ie. faster) since asynchronous transfer “wastes” about 30% of the bits for start and stop bits in sending a 7-bit ASCII code.

- An example of asynchronous serial data transfer is the RS232 serial port found in most computers.

Handshaking for I/O

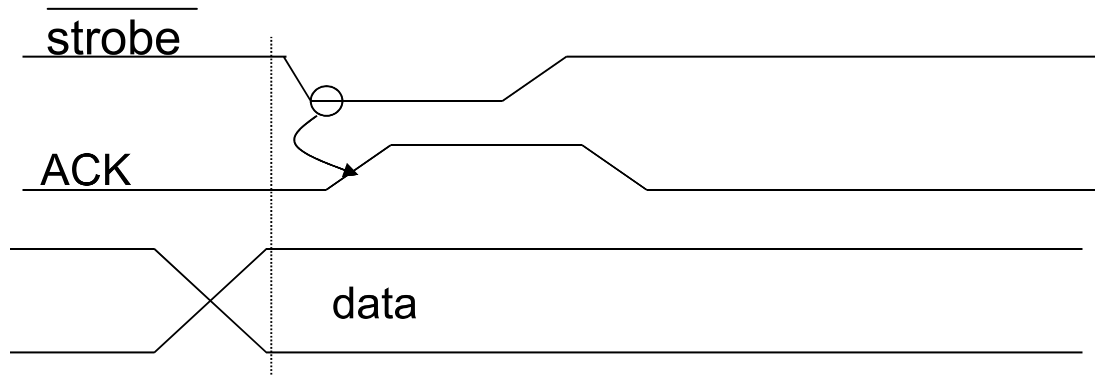
- Three types of handshaking:
  - Simple parallel I/O - no handshaking implemented
  - Single handshake I/O - uses STB-ACK handshake
  - Double handshake I/O - uses STB-ACK and STB-ACK



simple

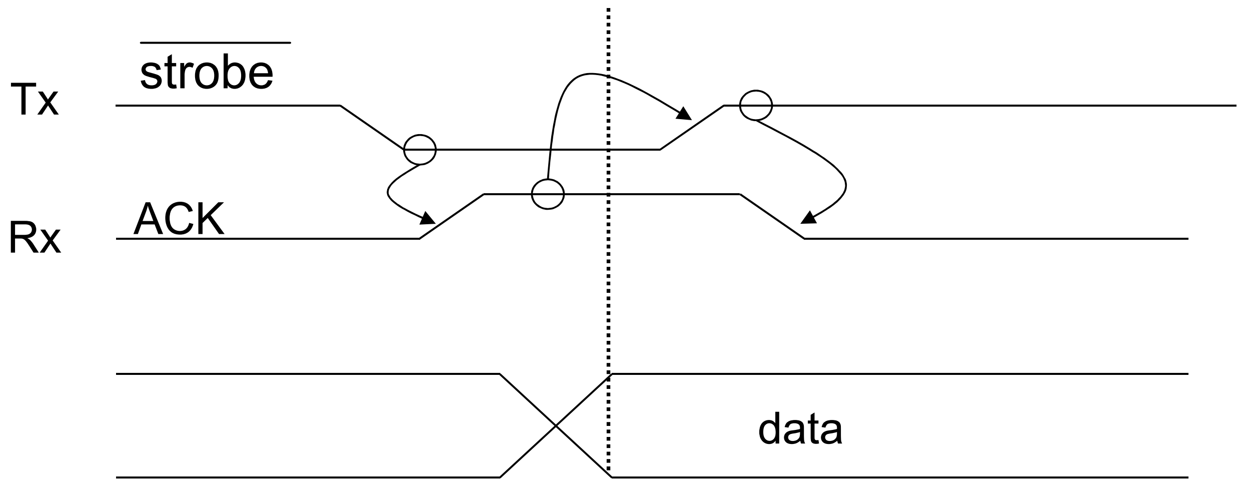
(no handshake)

Handshaking for I/O (cont.)



Single Handshake

: means the circle will trigger the event pointed by the arrow after a given time



Double Handshake

Data transferred after the first STB and ACK.

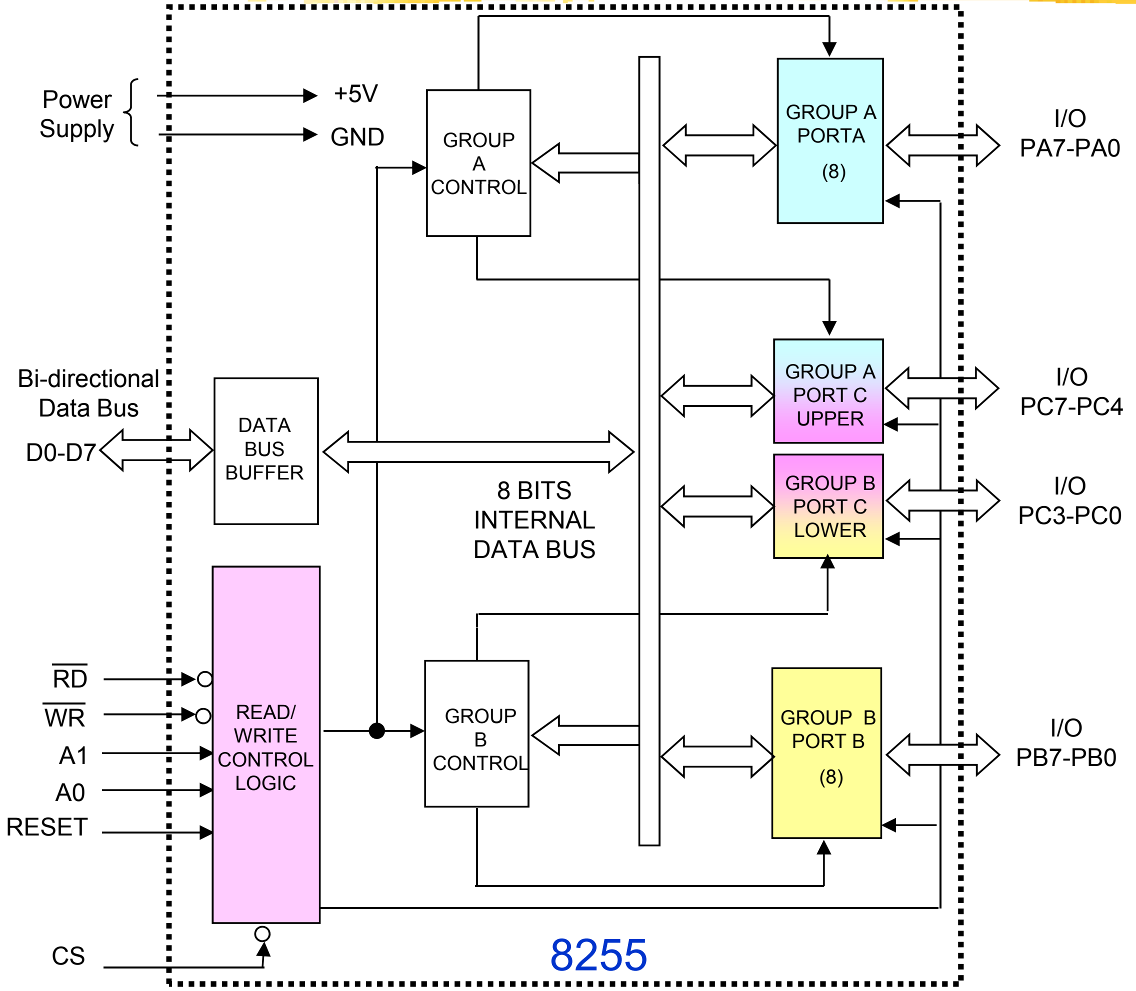
8255 Programmable Peripheral Interface (PPI)

- Intel 8255A is a general purpose parallel I/O interface. It provides three I/O port (A, B and C).
- Intel 8255A provides three modes of operations (mode 0, 1, and 2).
- 8255A's mode of operation is determined by the contents of its control register (see Intel data sheet for further details).
- Port A and Port B can be set to different mode and input/output independently.

Note:
Mode 2 is a bidirectional transmission mode; not necessary using double handshaking.

8255A Programmable Parallel Port Device

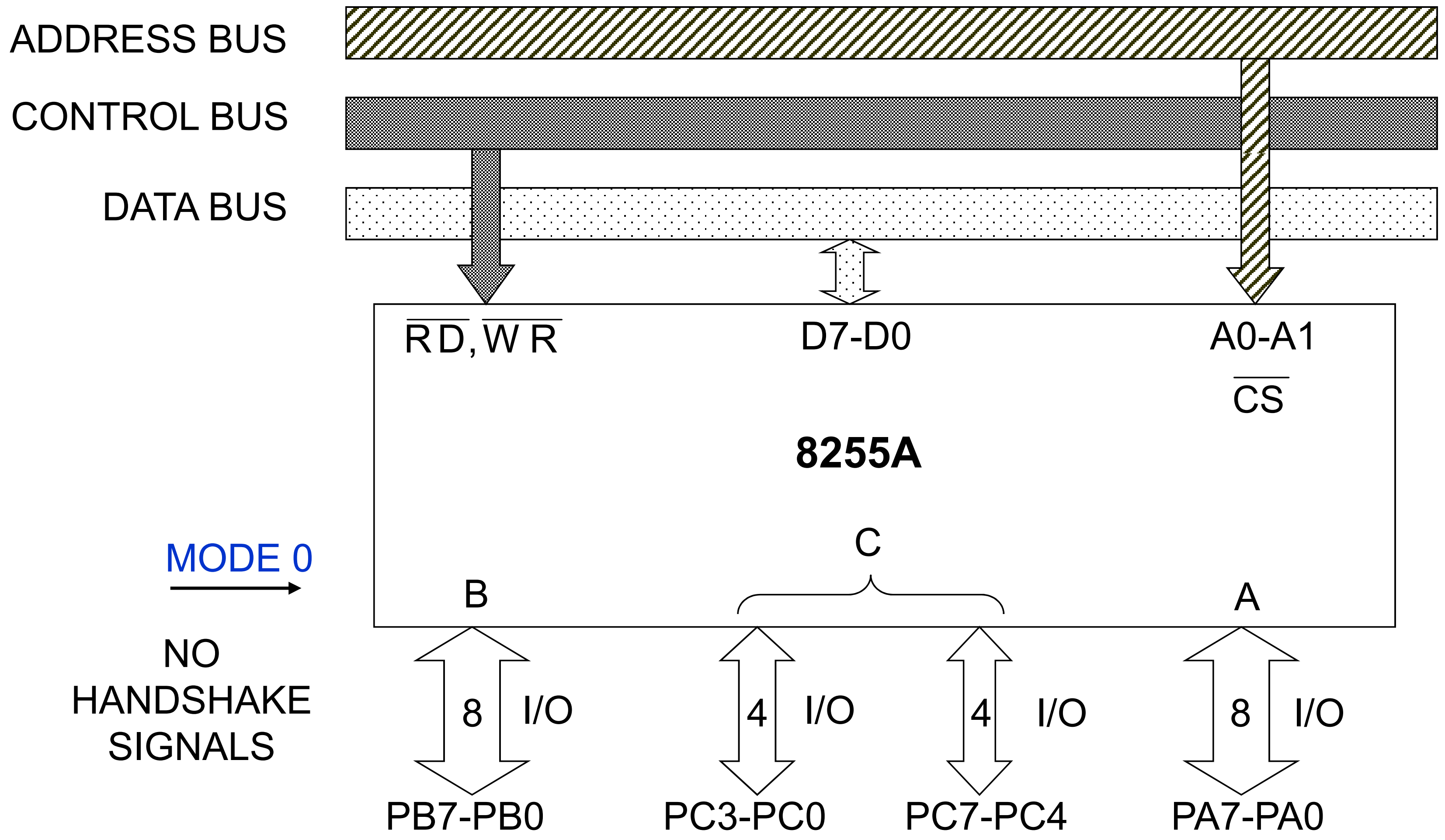
Internal structure block diagram



8255A Mode 0 Operation

- In Mode 0 operation, no handshaking will be used.
- If both port A and port B are initialized as mode 0 operation, port C can be used together as an additional 8-bit port, or two 4-bit ports.
- When used as an outputs, port C line can be set/reset by sending special control word to the *control register address*.
- The two halves of port C are independent and can be set as input or output port independently.

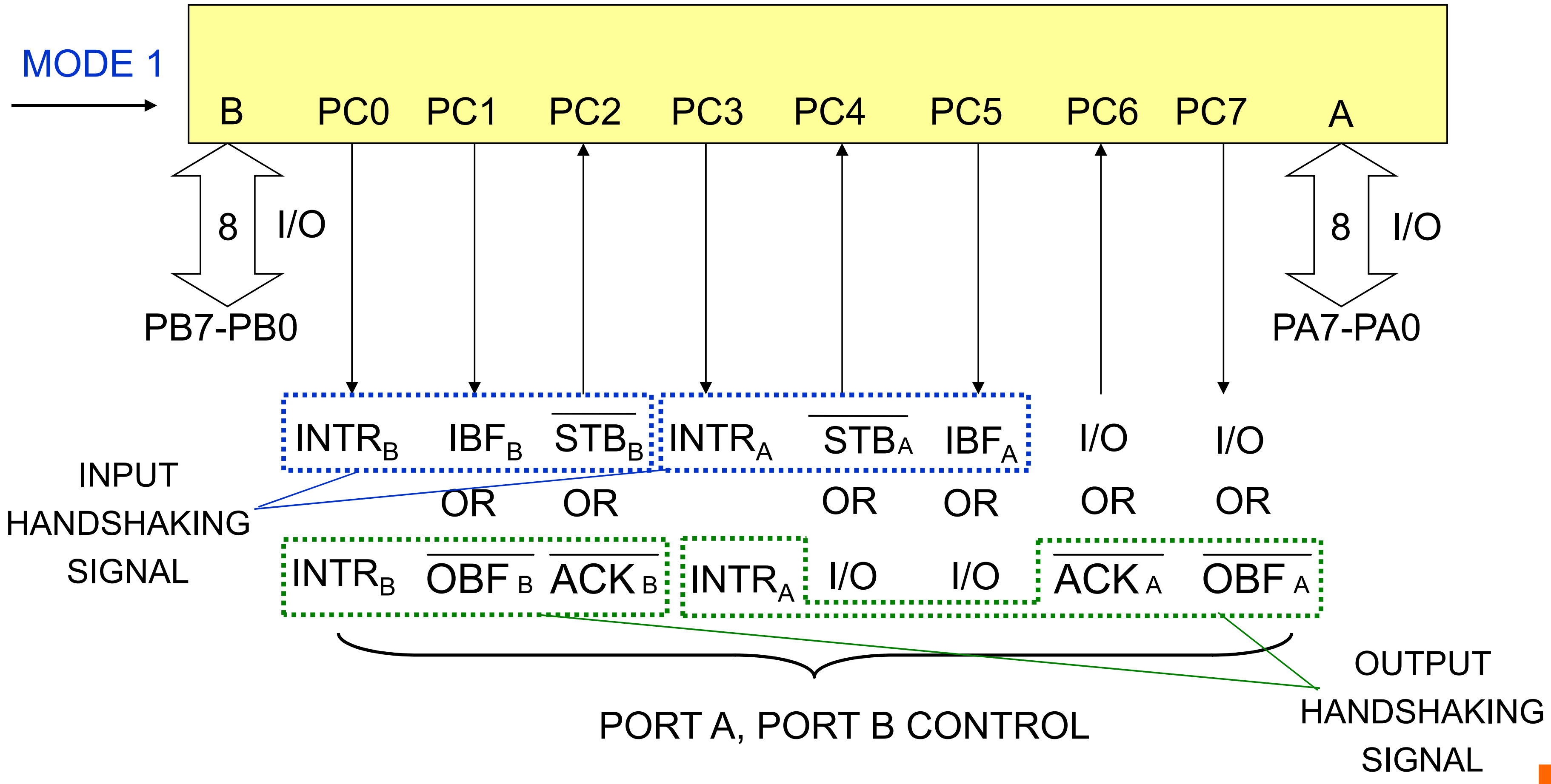
8255A Mode 0 Operation



8255A Mode 1 Operation

- In Mode 1 operation, single handshaking (strobed) is used.
- In this mode, some of the port C pins are used for handshaking purpose.
  - If Port A is set to mode 1 & input port: PC3, PC4, and PC5 pins are used.
  - If Port A is set to mode 1 & output port: PC3, PC6, and PC7 pins are used.
  - If Port B is set to mode 1 & input port: PC0, PC1, and PC2 pins are used.
  - If Port B is set to mode 1 & output port: PC0, PC1, and PC2 pins are used.

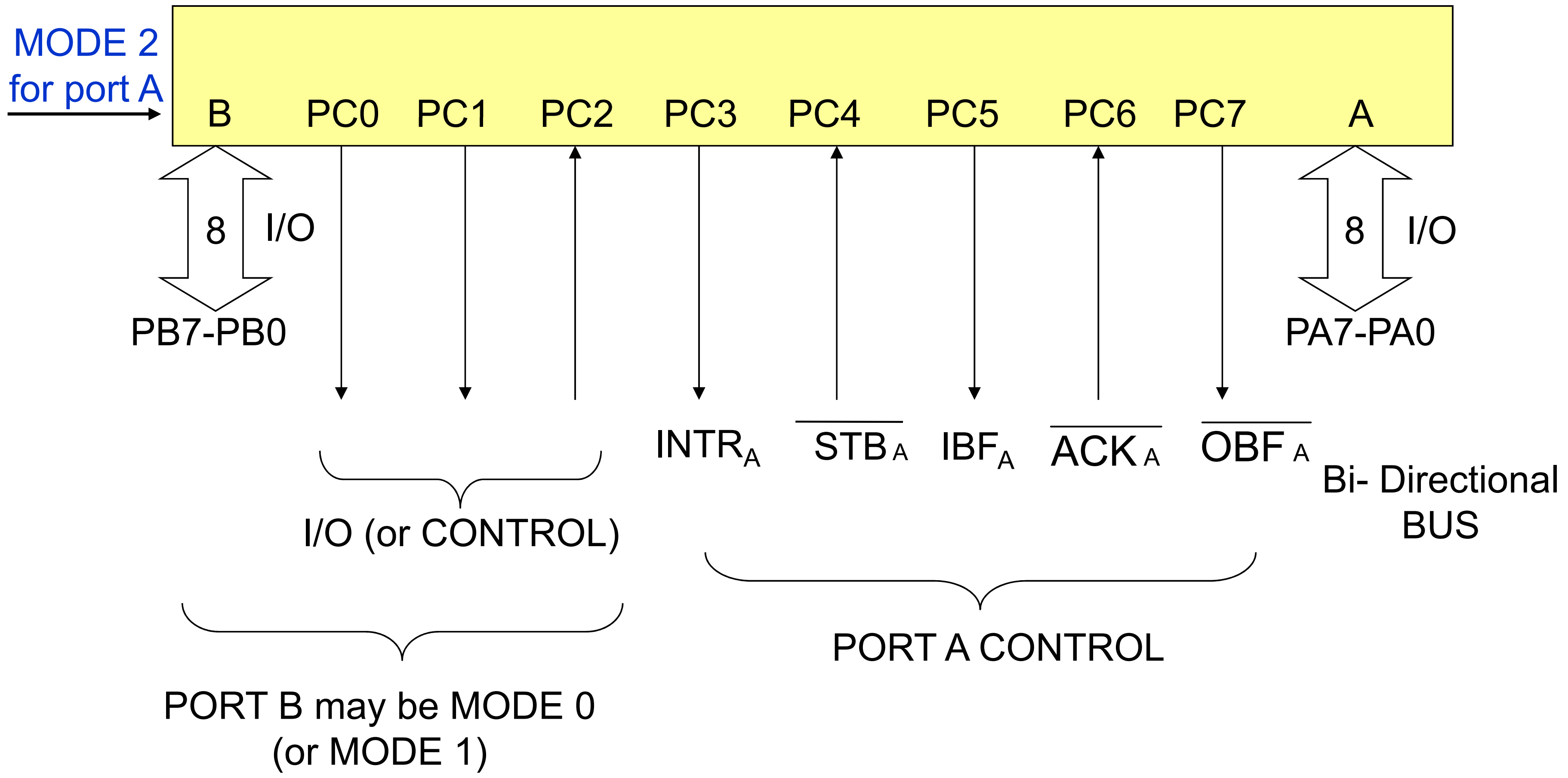
8255A Mode 1 Operation



8255A Mode 2 Operation

- Only port A can be initialized in mode 2.
- In mode 2, port A can be used as *bi-directional handshake data transfer*.
- In this mode, PC3-PC7 are used for handshake lines.
- PC0-PC2 can be used as I/O pins if port B is set to mode 0.
- PC0-PC2 are used as handshake lines if port B is set to mode 1.
- Note that Mode 2 does not mean that it uses double handshake I/O. It basically specify the bi-directional transmission of data flow. Both Mode 1 and Mode 2 require handshaking, but no specification which type is used.

Summary of Port C Usage when Port A is in Mode 2



8255A Control Words

- How to set 8255A? – by control words
- Two control word formats are used:
  - (1) mode-set control word format : to set the modes of each port
  - (2) port C bit set/rest control word format : to set the particular bit in port C. (such that 8255 can send a "1" or "0" via a particular bit to other devices)
- These two formats can be differentiated by the MSB of the control word.

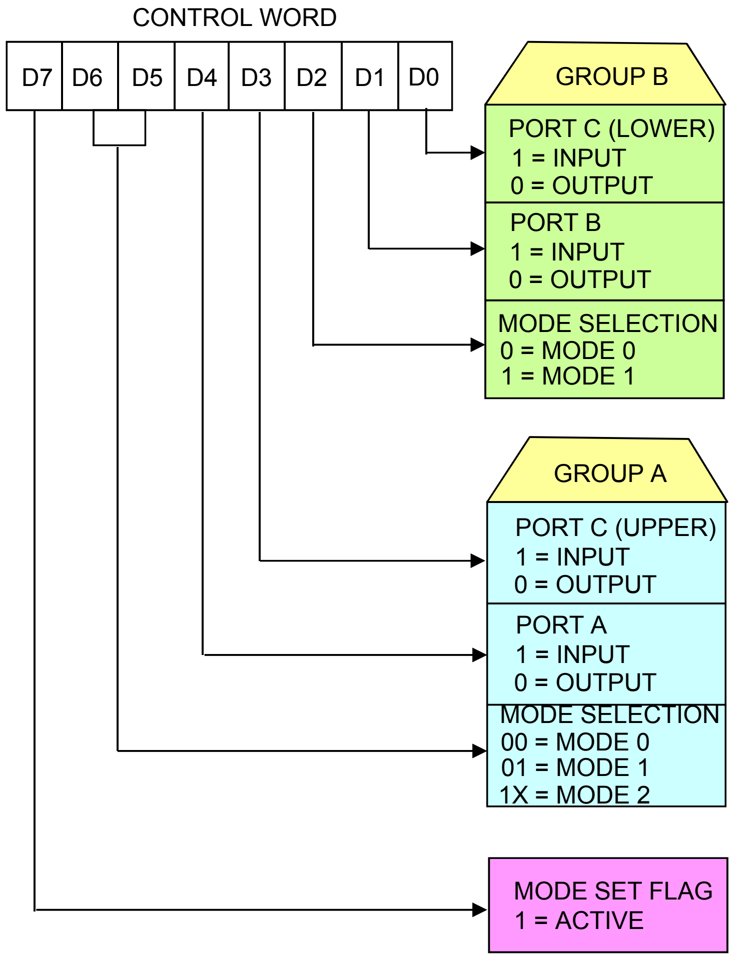
- The control words can be sent to the corresponding 8255A's address by using I/O instruction.

Ex. Assume 8255A is located at 0FFF8H, control register address is 0FFFEH, and the control word is to be set to 10001110B

→
```
MOV AL, 10001110B
MOV DX, 0FFFEH
OUT DX, AL
```

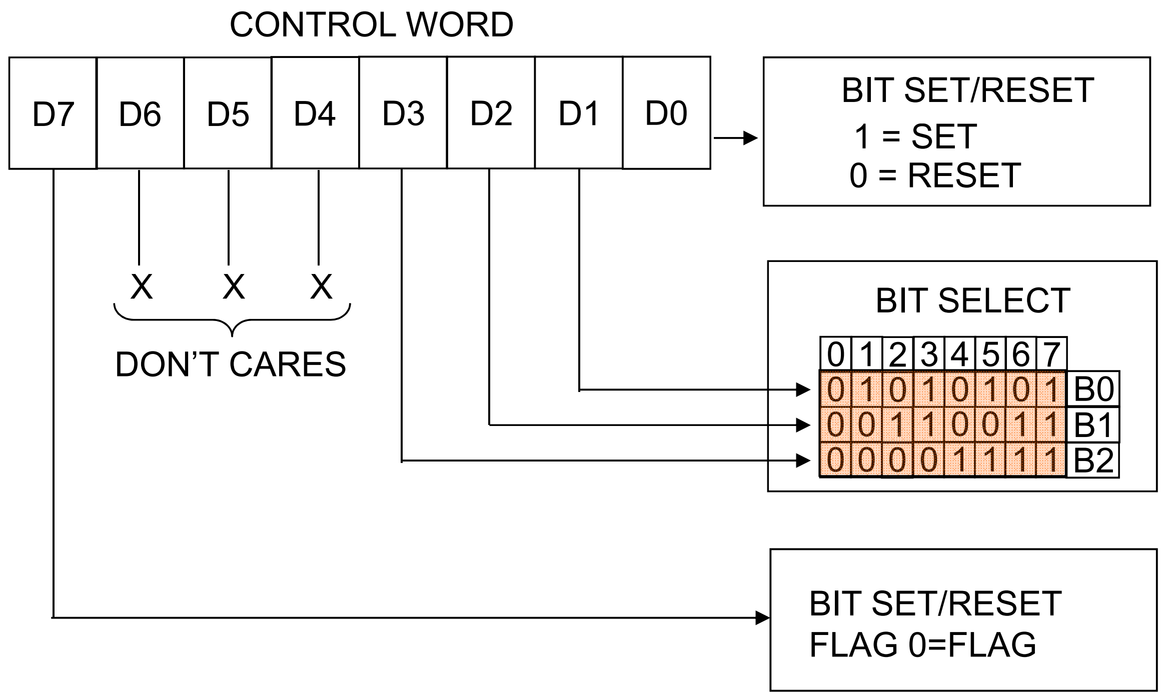
Q: Why there are two address for 8255?

8255A Control Word Formats



(a) Mode-set control word

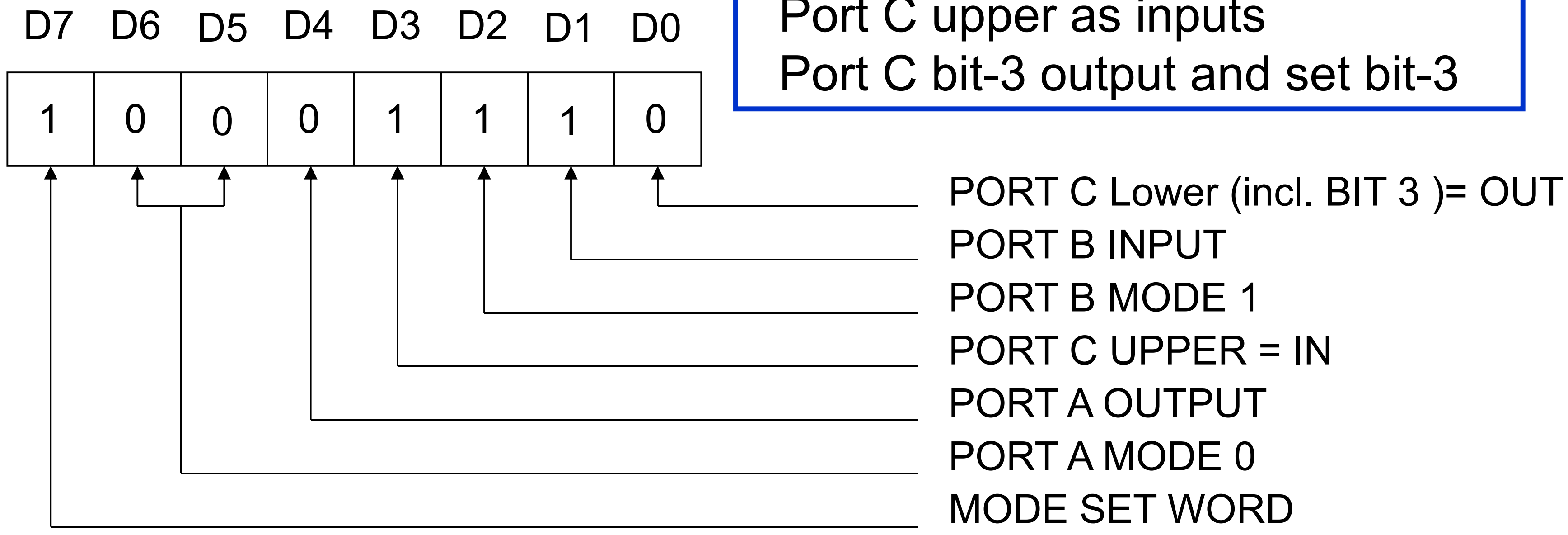
8255A Control Word Formats



(b) Port C bit set/reset control word

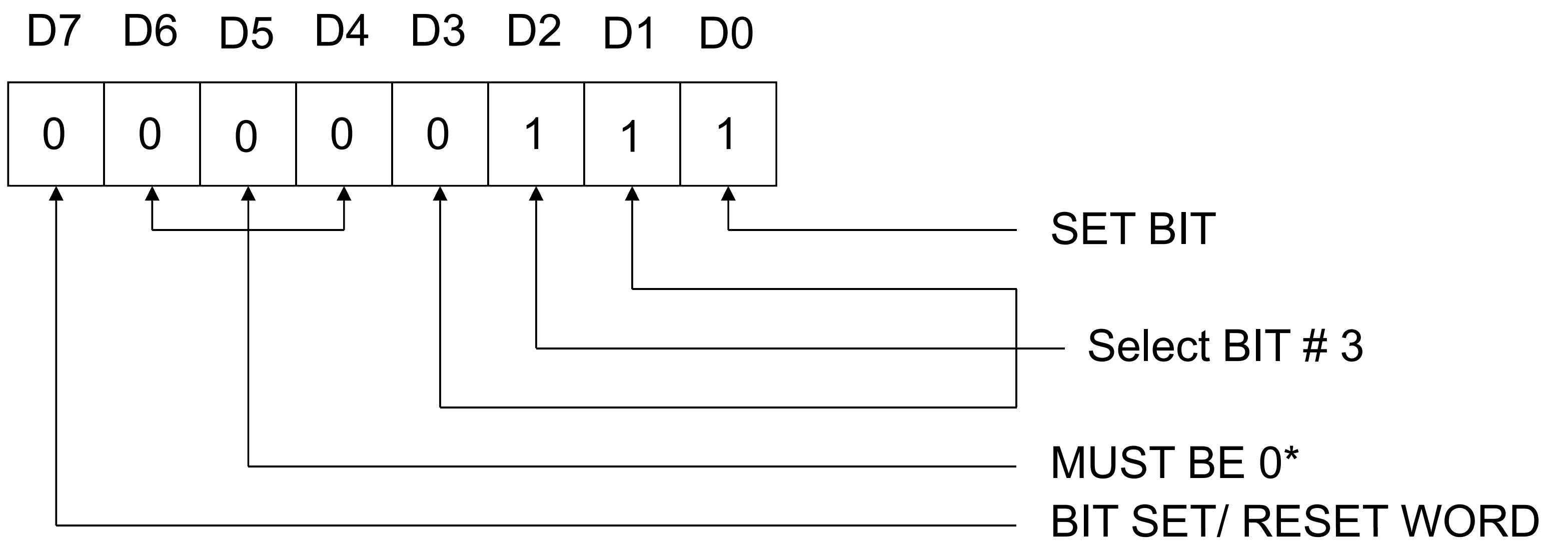
Control Word examples for 8255A

Suppose we want to set :
- Port A as mode 0 output
- Port B as mode 1 input
- Port C upper as inputs
- Port C bit-3 output and set bit-3



(a) mode-set control word

Control Word examples for 8255A (cont.)



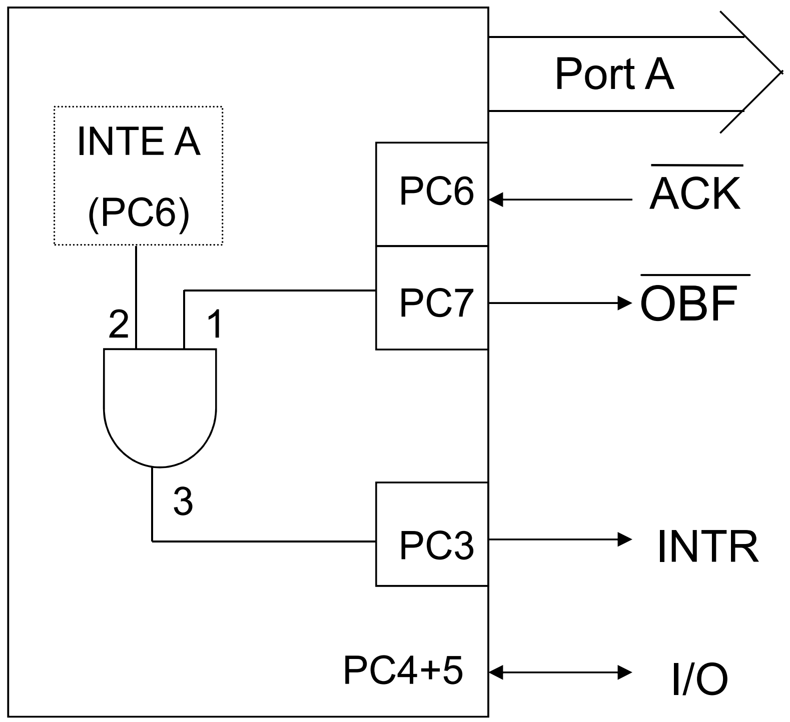
*:D6-D4 are set to 0 for simplicity and compatibility with future product.

(b) Port C bit set/reset control word to set bit-3

Strobed Output Operation of 8255A (Mode 1 Port A)

For example when outputting data to printer.

Mode 1 Port A



CONTROL WORD

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	1/0	X	X	X

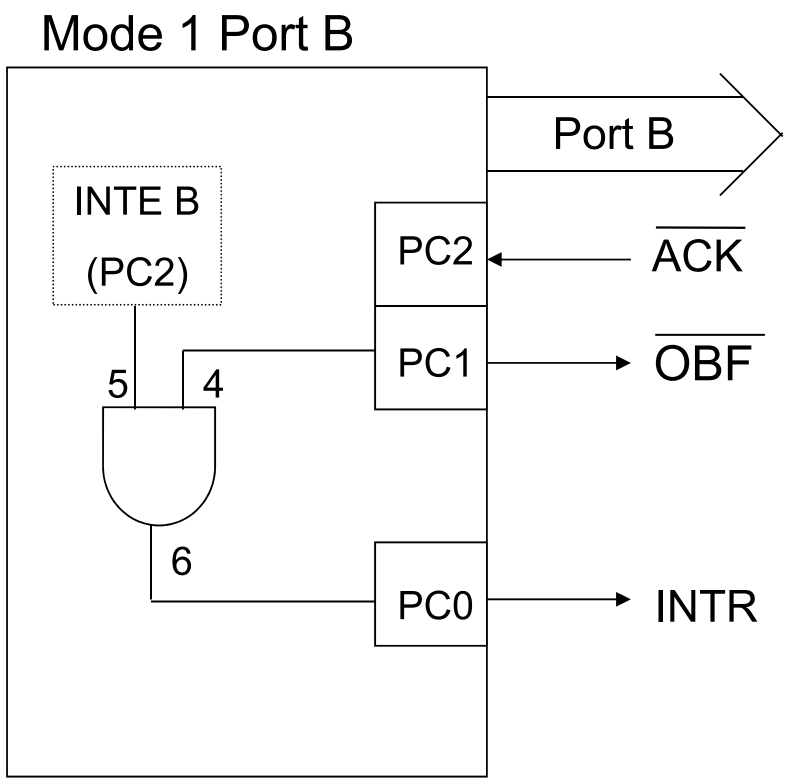
PC_{4+5}

1 = INPUT

0 = OUTPUT

Internal structure

Strobed Output Operation of 8255A (Mode 1 Port B)



CONTROL WORD

D7	D6	D5	D4	D3	D2	D1	D0
1	✗	✗	✗	✗	1	0	✗

Internal structure

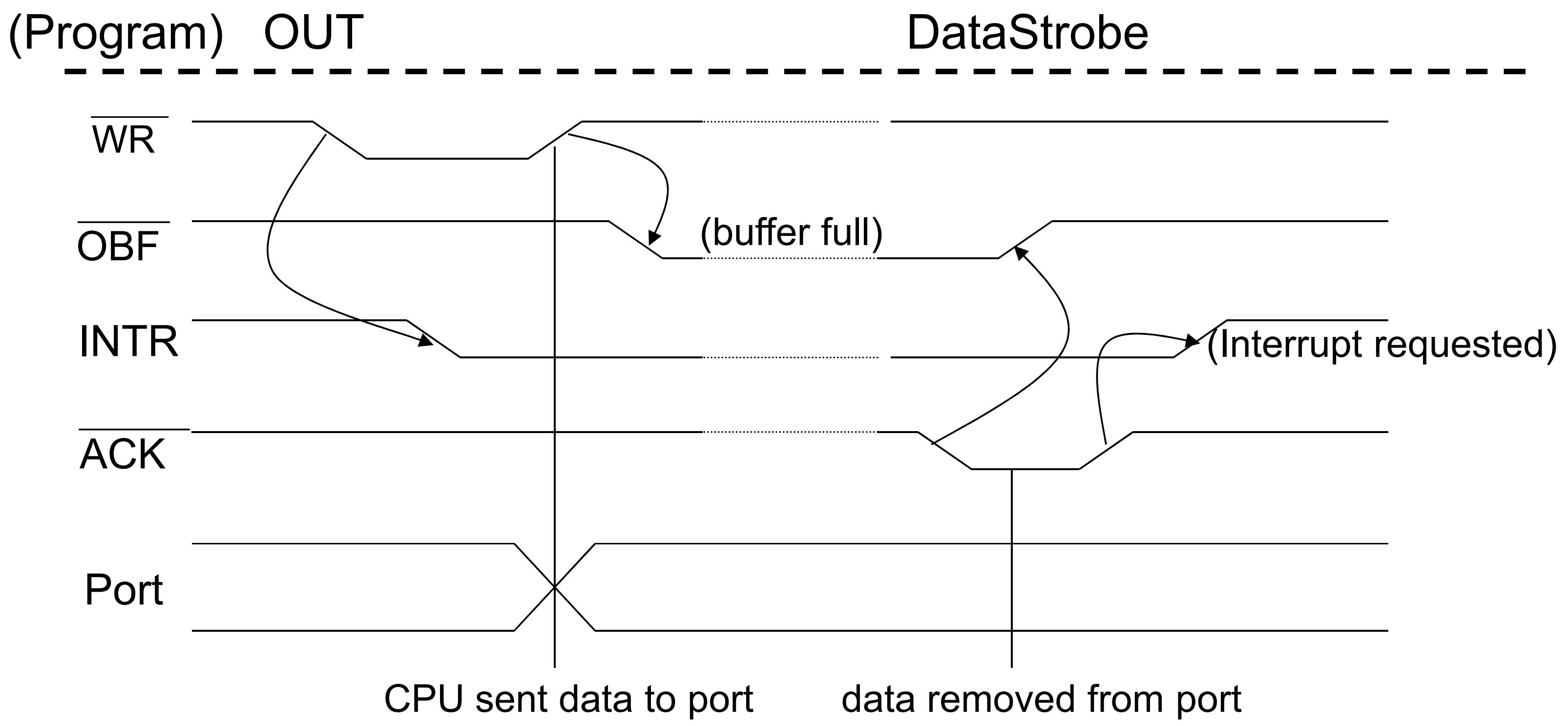
Signal Definition of Mode 1 Strobed Output

⌘ Whenever data are written to a port programmed as a strobed output port, the $\overline{\text{OBF}}$ signal becomes a logic 0 to indicate that data are present in the port latch (buffer) and is ready for external device to access. The external device strobes the $\overline{\text{ACK}}$ signal to indicate it has received the data. The $\overline{\text{ACK}}$ returns the $\overline{\text{OBF}}$ to logic 1.

⌘ $\overline{\text{OBF}}$	an output that goes low whenever data are output (using OUT) from CPU to the port A or port B latch; a strobe signal.
⌘ $\overline{\text{ACK}}$	an Acknowledge signal that causes $\overline{\text{OBF}}$ pin to return to a logic 1. This signal is a response from external device to indicate it has received data from 8255 port.
⌘ INTR	a signal that is often used to interrupts the processor when the external device receives the data and sends back the $\overline{\text{ACK}}$ signal.
⌘ INTE	an internal bit programmed to enable or disable the INTR pin. INTE A is programmed as PC6 (for output mode) or PC4 (for input mode) and INTEB is PC2 (for both input/output mode).

(Also check 8255 data sheet in Blackboard Learn)

Strobed Output Operation (Mode 1) of 8255A

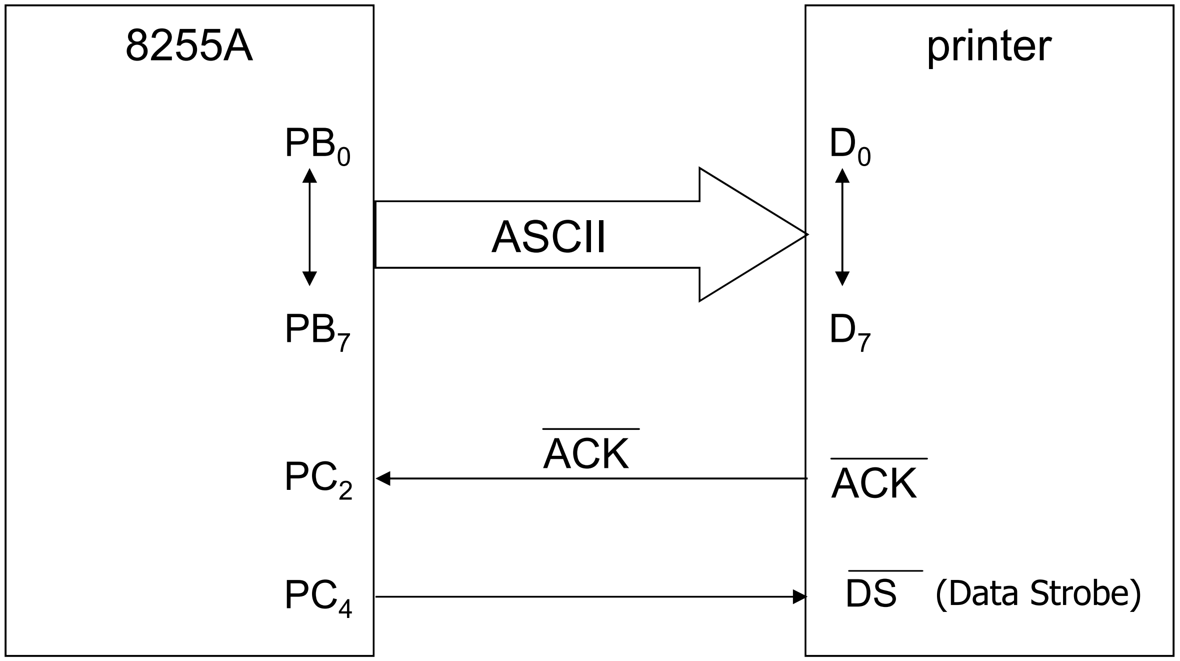


Timing diagram

Q: who sends what data?

Data flow: CPU → port buffer → external device

Example of Strobed Output Mode of Operation for 8255A



8255A connected to a parallel printer interface
(Note that PC4, rather than the $\overline{OBF}$ on PC1 for port B, is used for data strobe in this example.)

Example of Strobed Output Mode of Operation for 8255A (cont.)

;a procedure that transfers an ASCII-coded character from AH to the printer via port B

```
PORTC     EQU       62H
PORTB     EQU       61H
COMMAND             EQU       63H                       ; control word

BIT1      EQU       2
PRINT     PROC      NEAR
;check for printer ready
          IN        AL,PORTC            ;get OBF
          TEST      AL,BIT1             ;test OBF
          JZ        PRINT               ;jump if OBF = 0
;send character to printer via port B
          MOV       AL,AH               ;get character
          OUT       PORTB,AL            ;send it
;send DS to printer
          MOV       AL,8                ;clear DS (data strobe pin on printer)
          OUT       COMMAND,AL          ; 8=00001000B  -> reset PC4
          MOV       AL,9                ;set DS
          OUT       COMMAND,AL          ; 9=00001001B  -> set PC4
          RET
PRINT     ENDP
```

(OBF in the comments is printed with an overbar.)

Jump if Z=1 or AL AND BIT1=0 or $\overline{\text{OBF}}$=0

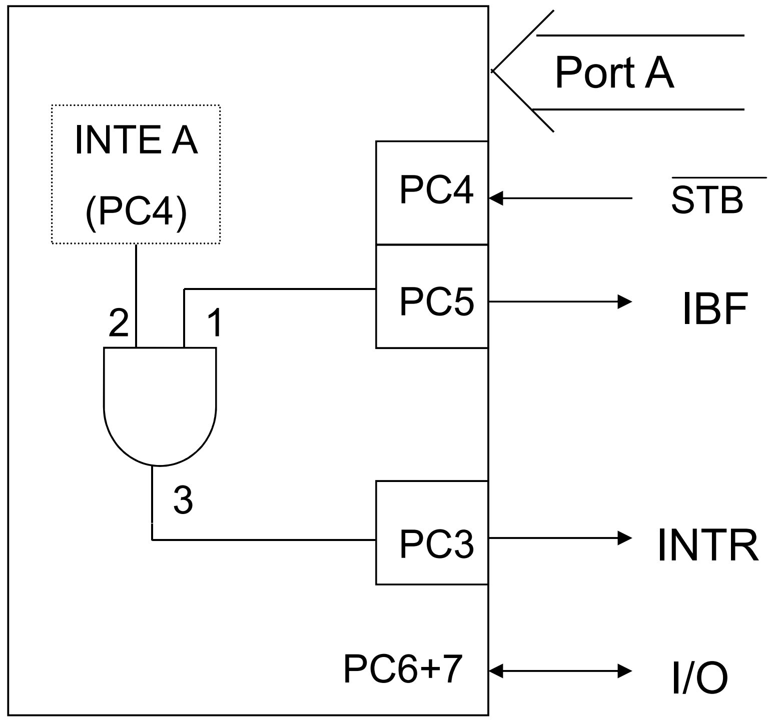
Check if printer has accessed the previous data in the port buffer

Check mode set control word format

Strobed Input Operation of 8255A (Mode 1 Port A)

Mode 1 Port A

For example when inputting data from keyboard.



CONTROL WORD

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	1/0	╳	╳	╳

For PC_{6+7}

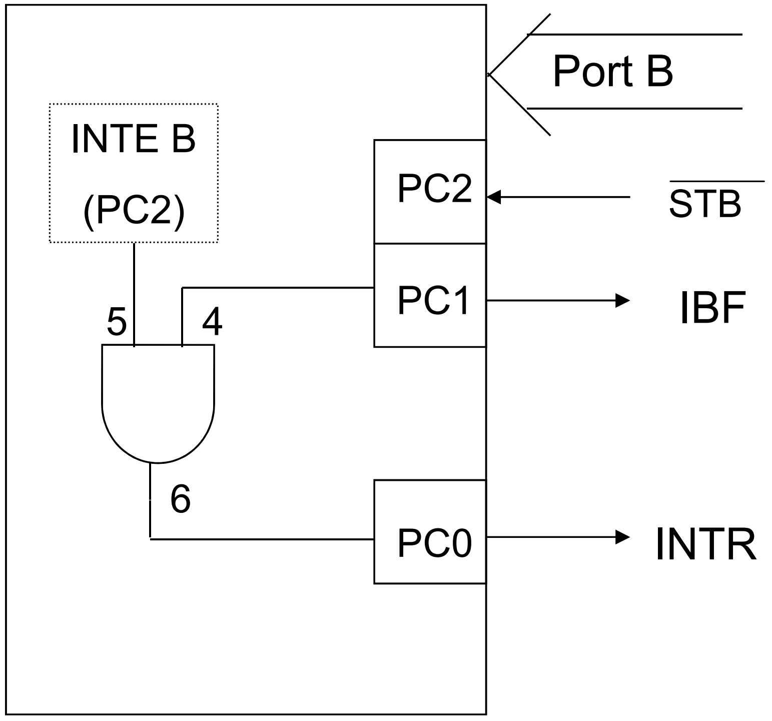
1 = INPUT

0 = OUTPUT

Internal Structure

Strobed Input Operation of 8255A (Mode 1 Port B)

Mode 1 Port B

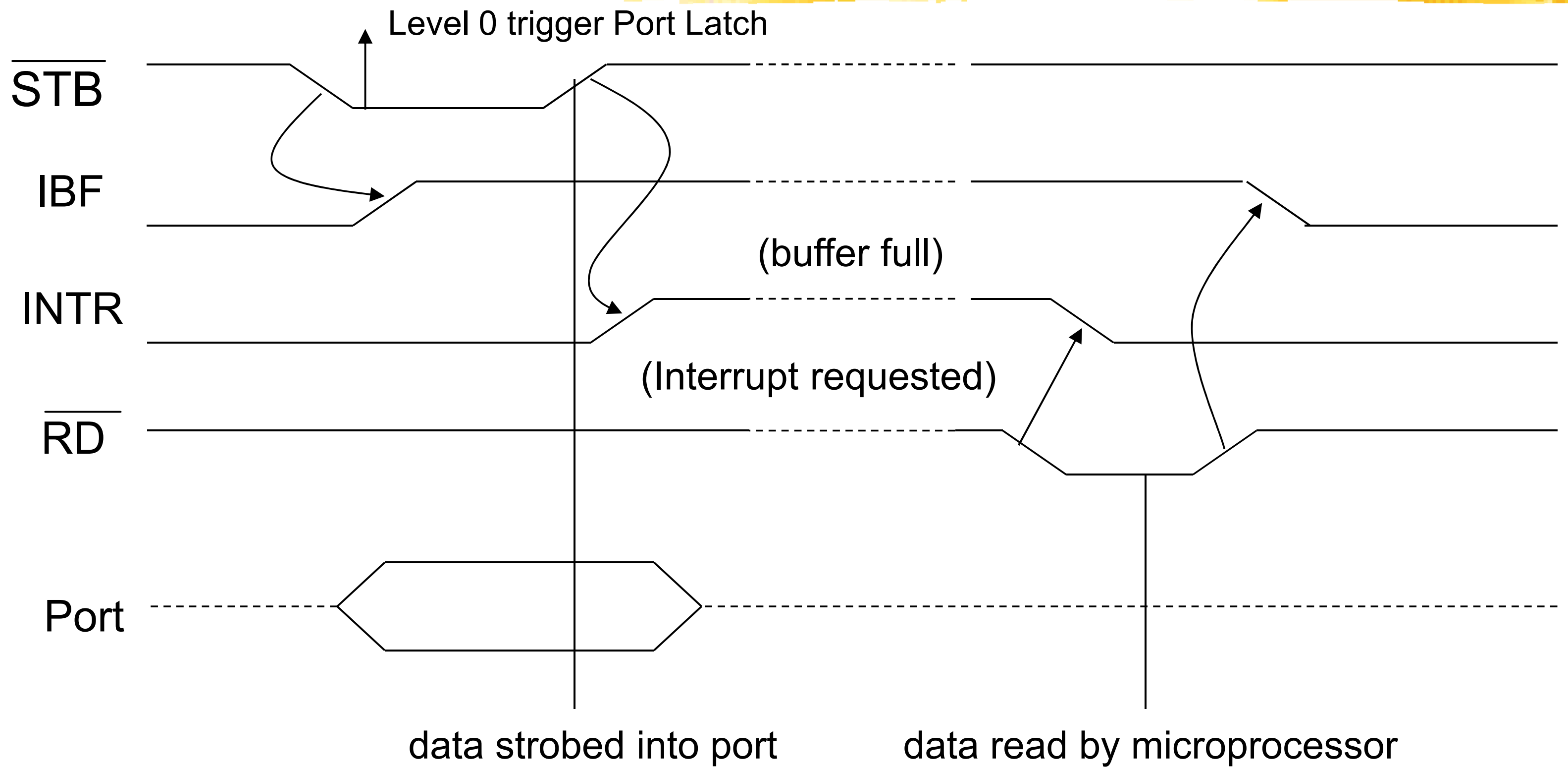


CONTROL WORD

D7	D6	D5	D4	D3	D2	D1	D0
1	X	X	X	X	1	1	X

Internal Structure

Strobed Input Operation (Mode 1) of 8255A



Timing diagram

Data flow: external device → port buffer → CPU

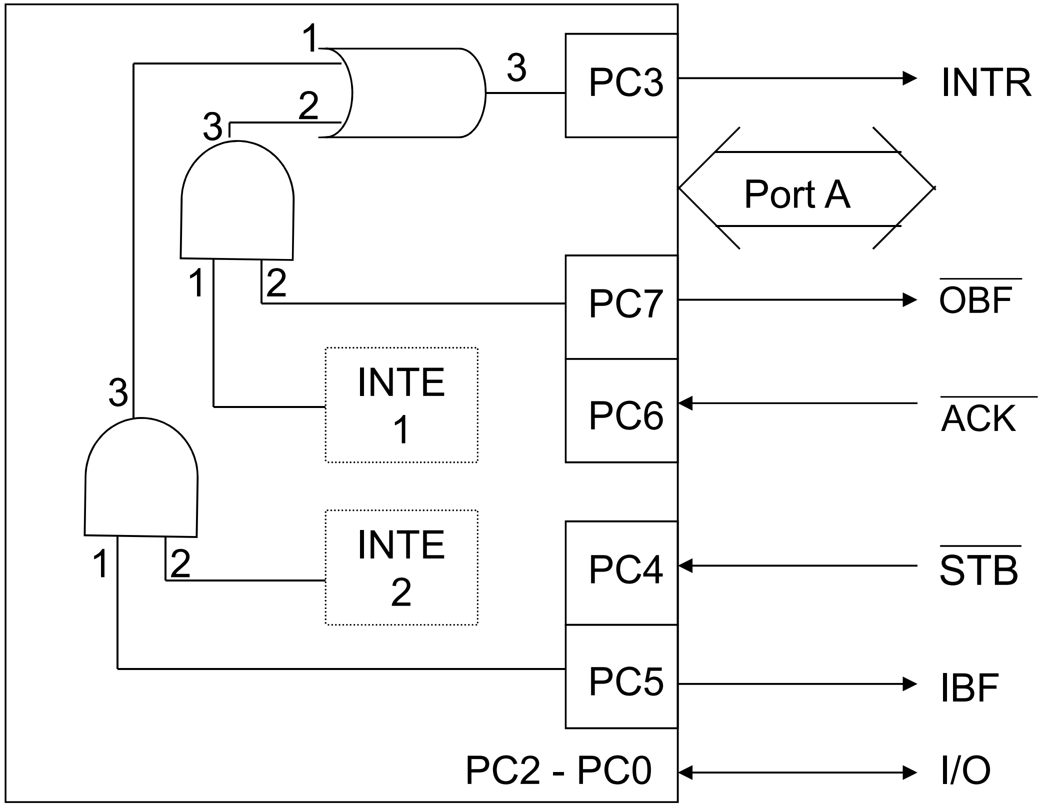
Example Procedure of keyboard encoder reading

```
;procedure that reads the keyboard encoder and returns with the ASCII character in AL
=0020             BIT5    EQU     20H
=0022             PORTC EQU     22H
=0020             PORTA EQU     20H

0000              READ    PROC    NEAR

0000 E4 22                IN      AL,PORTC          ;read port C
0002 A8 20                TEST    AL,BIT5           ;test IBF (PC5)
0004 74 FA                JZ      READ              ;jump if IBF = 0

0006 E4 20                IN      AL,PORTA          ;read ASCII code
0008 C3                   RET

0009             READ    ENDP
```

Junp if Z=1 or
AL AND BIT5=0
or IBF=0

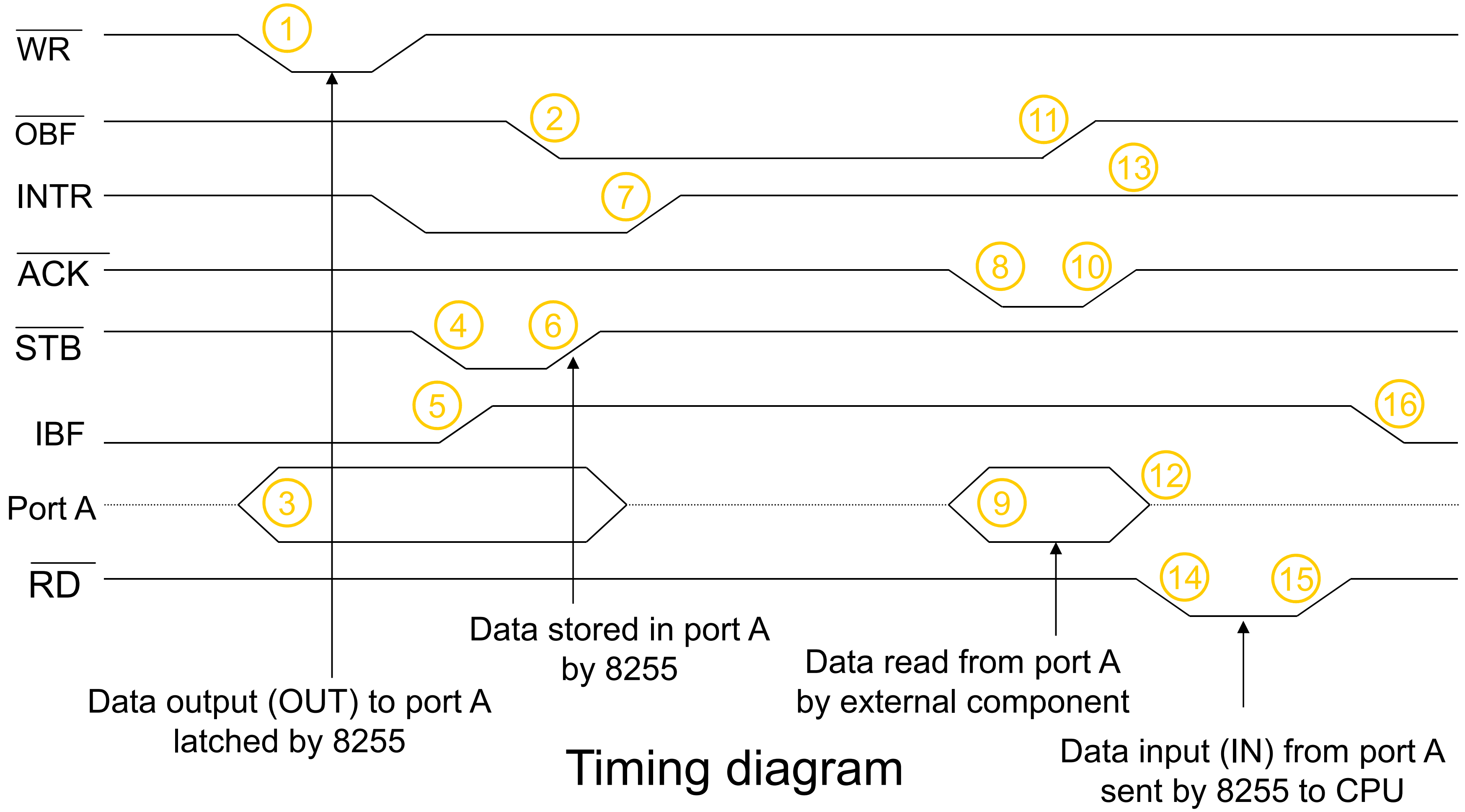
Mode 2 Operation of 8255A

Bi-directional mode 2



Internal Structure

Mode 2 Operation of 8255A



$\overline{WR}$
$\overline{OBF}$
INTR
$\overline{ACK}$
$\overline{STB}$
IBF
Port A
$\overline{RD}$
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Data stored in port A by 8255
Data output (OUT) to port A latched by 8255
Data read from port A by external component
Data input (IN) from port A sent by 8255 to CPU

Timing diagram

Example Procedure of output through bidirectional bus data

```
;procedure that transmits AH through the bi-directional bus of port A
=0080           BIT7    EQU     80H
=0062           PORTC   EQU     62H
=0060           PORTA   EQU     60H

0000            TRANS   PROC    NEAR
               ;test OBF
0000 E4 62              IN      AL,PORTC        ;get OBF
0002 A8 80              TEST    AL,BIT7         ;test OBF
0004 74 FA              JZ      TRANS           ;jump if OBF = 0
               ;send data
0006 8A C4              MOV     AL,AH           ;get data
0008 E6 60              OUT     PORTA,AL        ;output data to port A
000A C3                 RET

000B           TRANS    ENDP
```

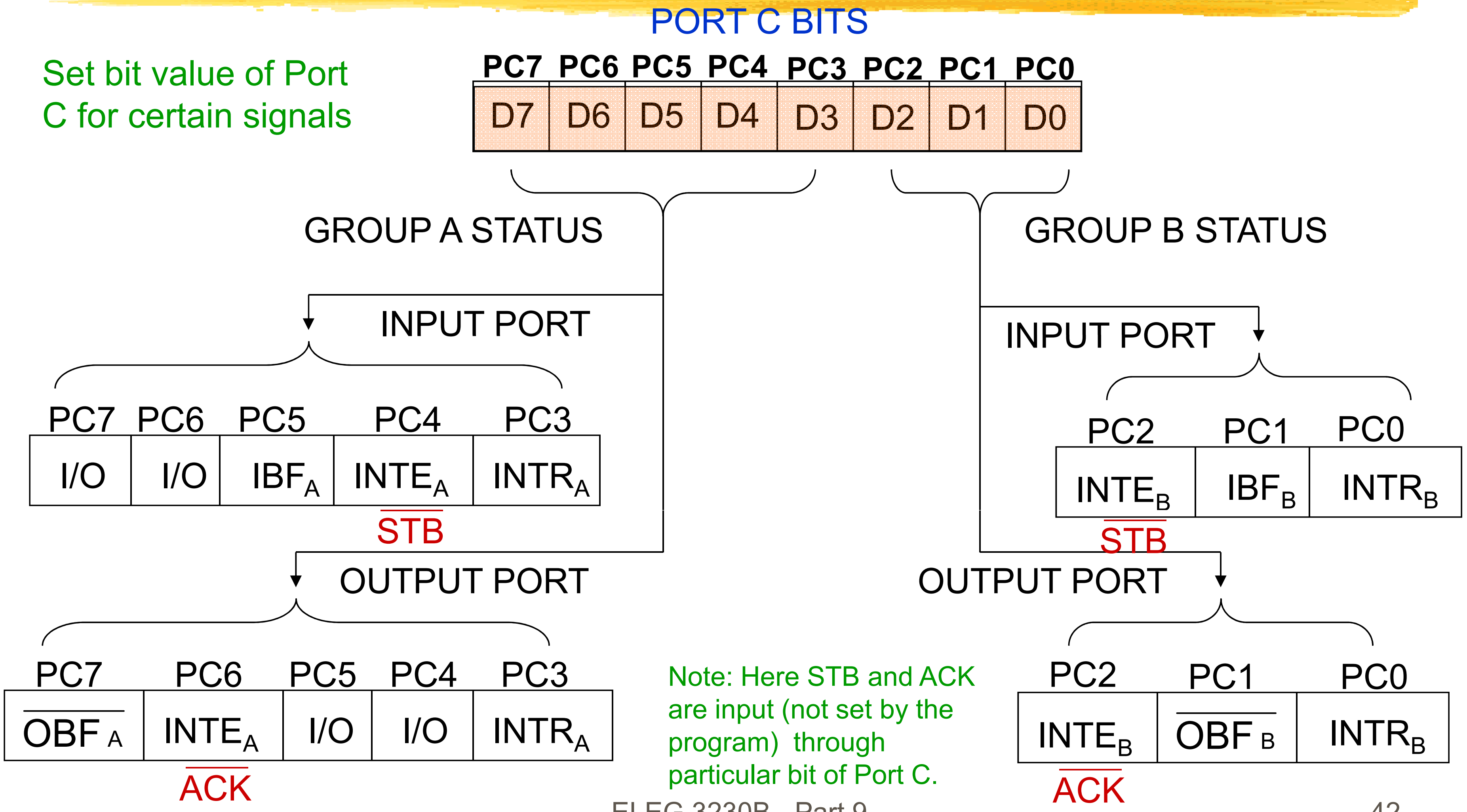
Example Procedure of bidirectional bus data input

;procedure that inputs data from the bi-directional bus and returns with it in AL

```
=0020              BIT5    EQU     20H
=0062              PORTC   EQU     62H
=0060              PORTA   EQU     60H

0000               READ    PROC    NEAR
                  ;test IBF
0000 E4 62                 IN      AL,PORTC        ;get IBF
0002 A8 20                 TEST    AL,BIT5         ;test IBF
0004 74 FA                 JZ      READ            ;jump if IBF = 0
                  ;read data
0006 E4 60                 IN      AL,PORTA
0008 C3                    RET

0009              READ     ENDP
```

8255A Status Word Format (Port C) for Mode 1 Input and Output

Set bit value of Port C for certain signals



PORT C BITS

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
D7	D6	D5	D4	D3	D2	D1	D0

GROUP A STATUS

INPUT PORT

PC7	PC6	PC5	PC4	PC3
I/O	I/O	IBF_A	$INTE_A$	$INTR_A$
			$\overline{STB}$	

OUTPUT PORT

PC7	PC6	PC5	PC4	PC3
$\overline{OBF}_A$	$INTE_A$	I/O	I/O	$INTR_A$
	$\overline{ACK}$			

GROUP B STATUS

INPUT PORT

PC2	PC1	PC0
$INTE_B$	IBF_B	$INTR_B$
$\overline{STB}$		

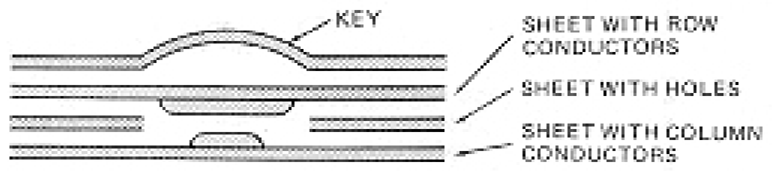
OUTPUT PORT

PC2	PC1	PC0
$INTE_B$	$\overline{OBF}_B$	$INTR_B$
$\overline{ACK}$		

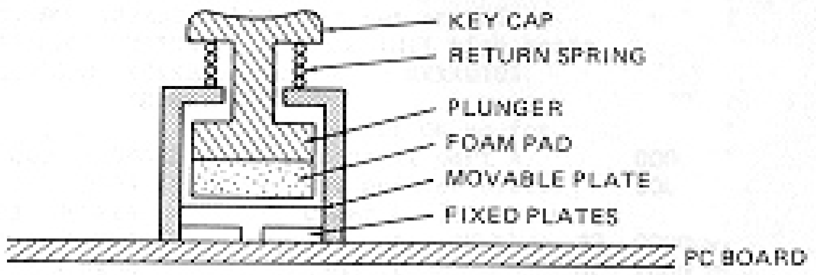
Note: Here STB and ACK are input (not set by the program) through particular bit of Port C.

Interfacing a Microprocessor to Keyboard

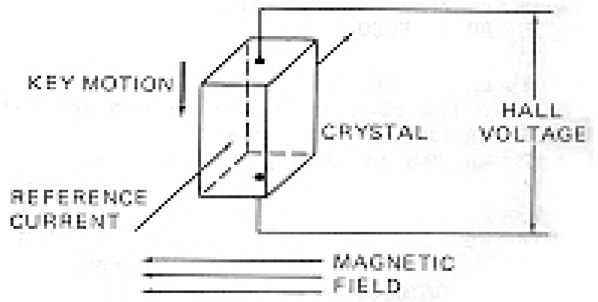
Keyswitch Types



(a) Membrane



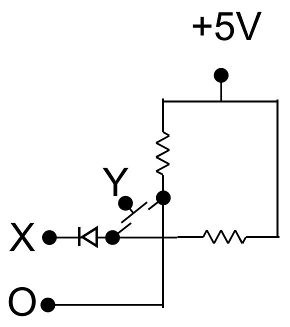
(b) Capacitive
20M key strokes



(c) Hall Effect
(100M key strokes)

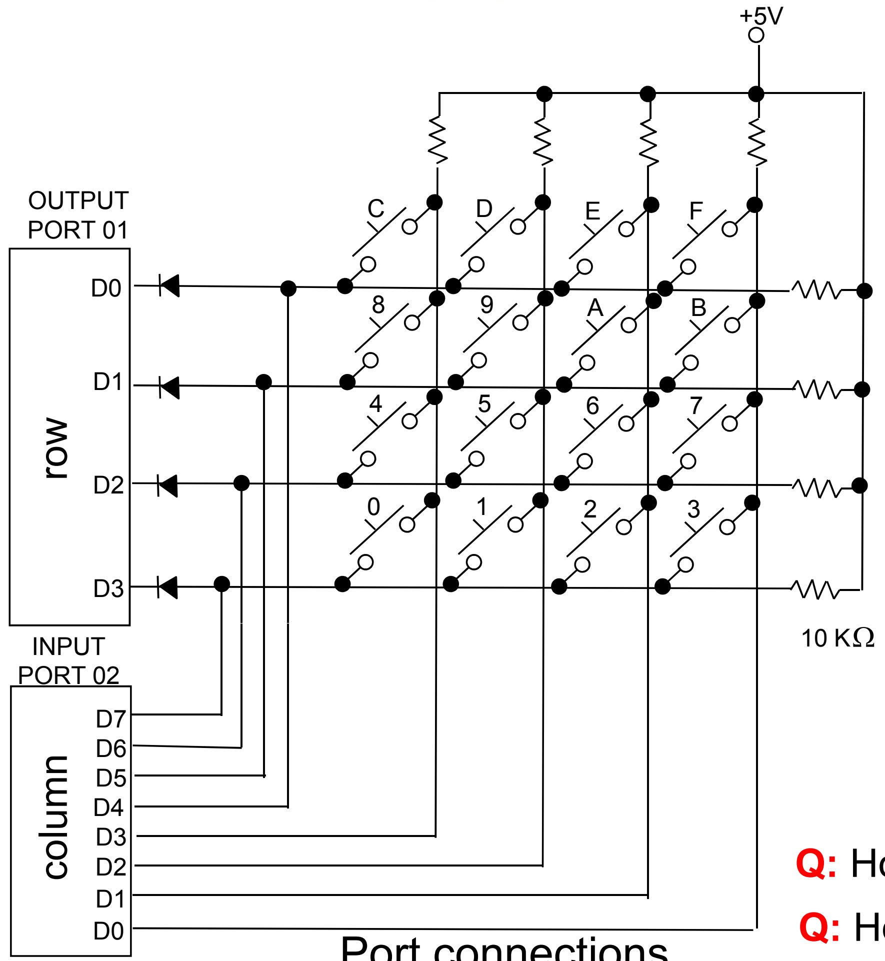
(D. Hall Fig 9-18)

Detecting a Matrix Keyboard Keypress



		Y	
	O	open	close
X	high (1)		
	low (0)		

X,Y: control
O: Output



Port connections

Q:
Assume key "9" is pressed. For the following cases, what are the values of D3-D0 at Port 2?
(1). Port 1, D3-D0=0000
(2). Port 1, D3-D0=1110
(3). Port 1, D3-D0=1101

(1) Port 2, D3-D0=

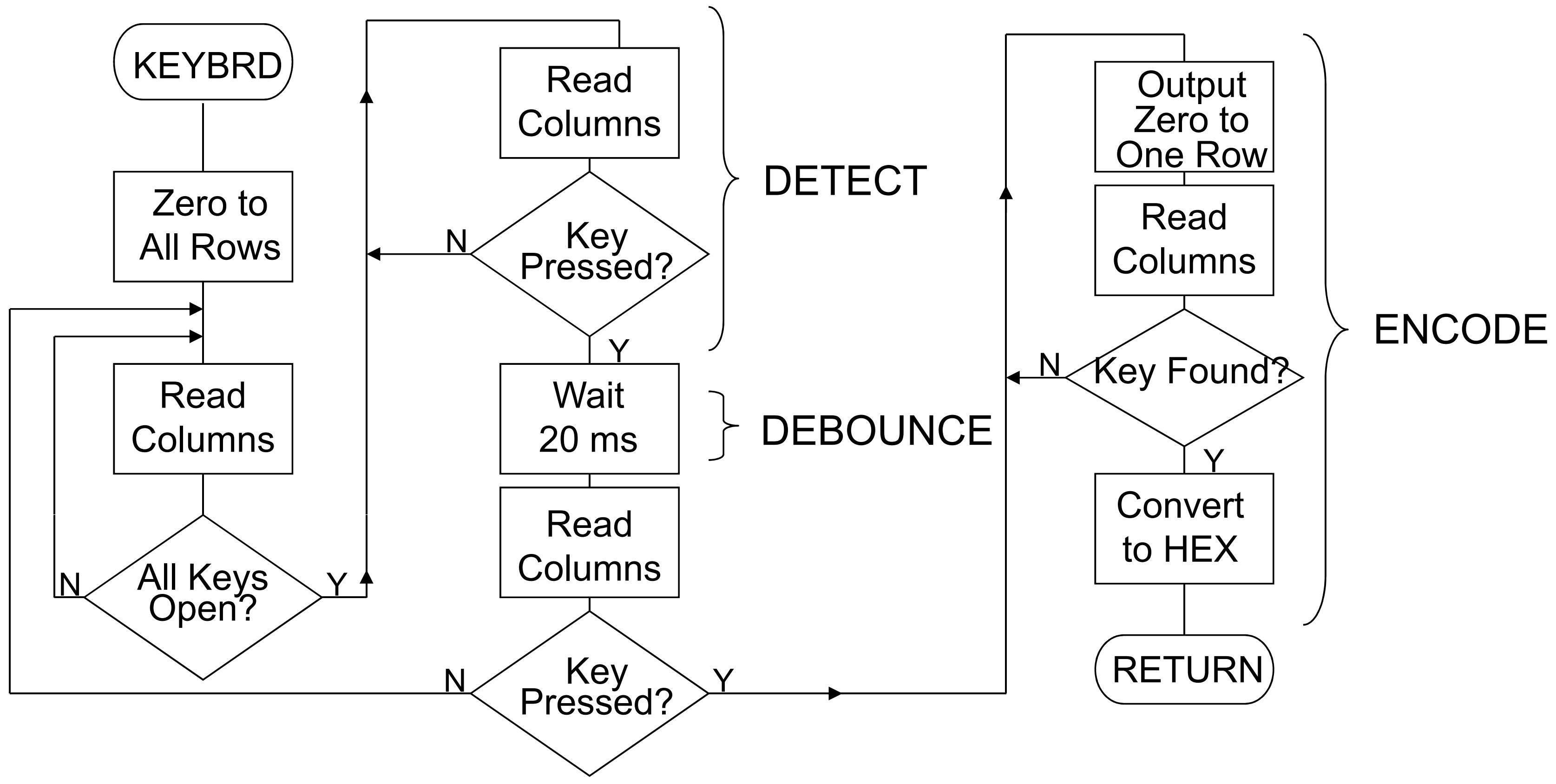
(2) Port 2, D3-D0=

(3) Port 2, D3-D0=

Q: How to detect if no key is pressed?

Q: How to detect which (one) key is pressed?

Detecting a Matrix Keyboard Keypress



(D. Hall)

Flowchart for procedure

Assembly Program for Keybrd Detect, Debounce and Encode

Q: Why, e.g., 7Eh corresponds to "3"?

```
                                ;8086 Program F9-20.ASM
                                ;Abstract    : Program scans and detect a 16 - switch keypad
                                             ; It initializes the ports below and then calls a procedure
                                             ; to input an 8 - bit value from a 16 - switch keypad and encode it.
                                ; Ports      : SDK-86 board Port P1A (FFF9H) - output, P1B(FFFBH) - input
                                ; Procedures : Calls KEYBRD to scan and decode 16 - switch keypad
                                ; Registers  : Uses CS, DS, SS, SP, AX, DX

0000                            DATA   SEGMENT   WORD   PUBLIC
                                ;                 0      1      2      3      4       5       6       7
0000 77 7B 7D 7E B7 BB BD BE    TABLE     DB     77H,   7BH,   7DH,   7EH,   0B7H,   0BBH,   0BDH,   0BEH

                                ;                 8       9       A       B       C       D       E       F
0008 D7 DB DD DE E7 EB ED +               DB     0D7H,   0DBH,   0DDH,   0DEH,   0E7H,   0EBH,   0EDH,   0EEH
     EE
0010                            DATA   ENDS
0000                            STACK_SEG   SEGMENT
0000 1E*(0000)                            DW         30          DUP(0)       ;set up stack of 30 words with value 0
                                       TOP_STACK   LABEL   WORD                 ;pointer to top of stack
003C                            STACK_SEG   ENDS
```

Assembly Program for Keybrd Detect, Debounce and Encode

```

0000               CODE   SEGMENT   WORD   PUBLIC
                        ASSUME      CS:CODE,      DS:DATA,     SS:STACK_SEG
0000  B8 0000s     START:  MOV   AX,   STACK_SEG              ; Initialize stack
0003  8E  D0               MOV   SS,   AX                     ; segment register and
0005  BC 003Cr             MOV   SP,   OFFSET   TOP_STACK     ; top of stack
0008  B8 0000s             MOV   AX,   DATA
000B 8E D8                 MOV   DS,   AX
                   ; initialize ports, mode 0, Port A for output, Ports B & C for input
000D BA FFFF               MOV   DX,   0FFFFH                 ; Put port control register address in DX
0010  B0 8B                MOV   AL,   10001011B              ; Code: control word 8BH
0012  EE                   OUT   DX,   AL                     ; Send control word
0013  E8 0001              CALL  KEYBRD
0016  90                    NOP
                   ; Program will continue here with other tasks


                   ; 8086   PROCEDURE   KEYBRD
                   ; Abstract     : Procedure gets a code from a 16 - switch and decodes it.
                                  ; It returns the code for the keypress in AL and AH = 00. If there
                                  ; is an error in the keypress then it returns AH = 01.
```

Assembly Program for Keybrd Detect, Debounce and Encode

```
                ; Ports      : Uses SDK - 86 ports P1A (FFF9H) for output and P1B (FFFBH) for input
                ; Inputs     : Keypress from port
                ; Outputs    : Keypress code or error message in AX
                ; Procedures : None used
                ; Registers  : Destroys AX

0017            KEYBRD  PROC  NEAR
0017  9C                PUSHF                          ; Save registers used
0018  53                PUSH  BX
0019  51                PUSH  CX
001A  52                PUSH  DX
                ; Send 0's to all rows
001B  B0 00             MOV   AL,  00
001D  BA FFF9           MOV   DX,  0FFF9H              ; Load output address
0020  EE                OUT   DX,  AL                  ; Send 0's
                ; Read columns to see if all keys are open
0021  BA FFFB           MOV   DX,  0FFFBH              ; Load input port address
0024  EC        WAIT_OPEN:  IN    AL,  DX
0025  24 0F                 AND   AL,  0FH             ; Mask row bits
0027  3C 0F                 CMP   AL,  0FH             ; Wait until no keys pressed
```

Assembly Program for Keybrd Detect, Debounce and Encode

```
0029  75 F9                 JNE   WAIT_OPEN       ; If key pressed, go back
                ; Read columns to see if a key is pressed
002B  EC        WAIT_PRESS: IN    AL,  DX         ; Read columns        }
002C  24 0F                 AND   AL,  0FH        ; Mask row bits       }  Detect
002E  3C 0F                 CMP   AL,  0FH        ; See if keypressed   }
0030  75 F9                 JE    WAIT_PRESS
                ; Debounce keypress
0032  B9 16EA               MOV   CX, 16EAH       ; Delay for 20ms
0035  E2 FE     DELAY:      LOOP DELAY
                ; Read columns to see if key still pressed
0037  EC                    IN    AL,  DX
0038  24 0F                 AND   AL,  0FH
003A  3C 0F                 CMP   AL,  0FH
003C  74 ED                 JE    WAIT_PRESS
                ; Find the key
003E  B0 FE                 MOV   AL,  0FEH       ; Initialize a row mask with bit-0
0040  8A C8                 MOV   CL,  AL         ;     low and save the mask
0042  BA FFF9   NEXT_ROW:   MOV   DX,  0FFF9H     ; Send out a low on one row
0045  EE                    OUT   DX,  AL
0046  BA FFFB               MOV   DX,  0FFFBH     ; Read columns and check for low
```

Q: Why sends 0FEh on Port A?

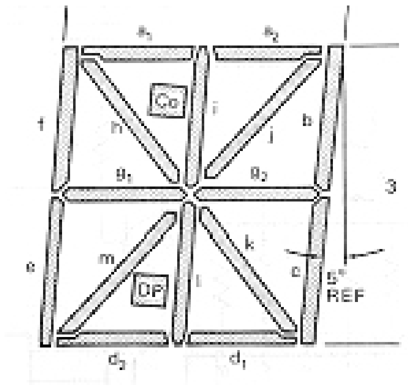
Assembly Program for Keybrd Detect, Debounce and Encode

```
0049  EC                       IN    AL,  DX
004A 24 0F                     AND  AL,  OFH                  ; Mask out row code
004C 3C 0F                     CMP  AL,  OFH                  ; If low in a column then
004E 75 06                     JNE   ENCODE                   ; key column found, so encode it
0050 D0 C1                     ROL   CL,  01                  ; else rotate mask
0052 8A C1                     MOV  AL,  CL
0054 EB EC                     JMP   NEXT_ROW                 ; and look at next row
                  ; Encode the row/column information
0056  BB 000F     ENCODE:      MOV   BX,  000FH               ; Set up BX as a counter
0059  EC                       IN     AL,  DX                 ; read row and column from port
005A 3A 87 0000r  TRY_NEXT:    CMP   AL, TABLE[BX]    ←       ; Compare row/column code with table entry
005E  74 08                    JE      DONE                   ; Hex code in BX
0060  4B                       DEC   BX                       ; Point at next  table entry
0061  79  F7                   JNS    TRY_NEXT
0063  B4 01                    MOV   AH,  01                  ; Pass an error code in AH
0065  EB 05 90                 JMP    EXIT
0068  8A C3       DONE:        MOV   AL,  BL                  ; Hex code for key in AL
006A  B4 00                    MOV   AH,  00                  ; Put key-valid code in AH
006C  5A          EXIT:        POP    DX                      ; Restore calling program
006D 59                        POP    CX                      ; registers
```

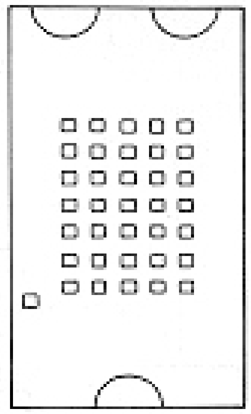
Assembly Program for Keybrd Detect, Debounce and Encode

```
101  006E 5B                          POP    BX
102  006F  9D                         POPF
103  0070  C3                         RET
104  0071               KEYBRD        ENDP
105  0071               CODE          ENDS
106                                   END
```

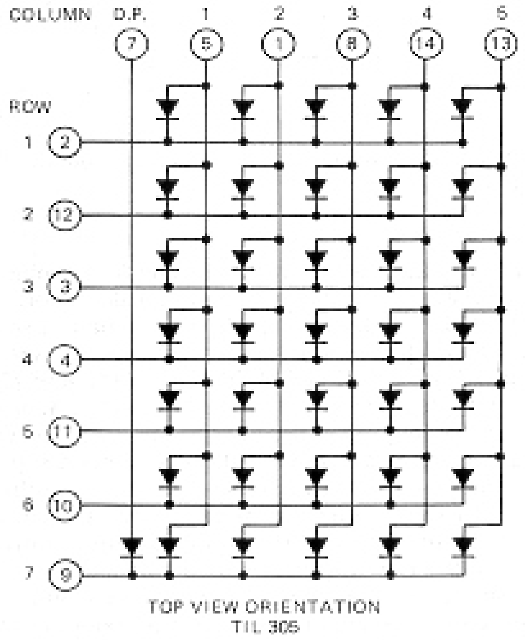
Interfacing to Alphanumeric Display



(a) 18-segment display



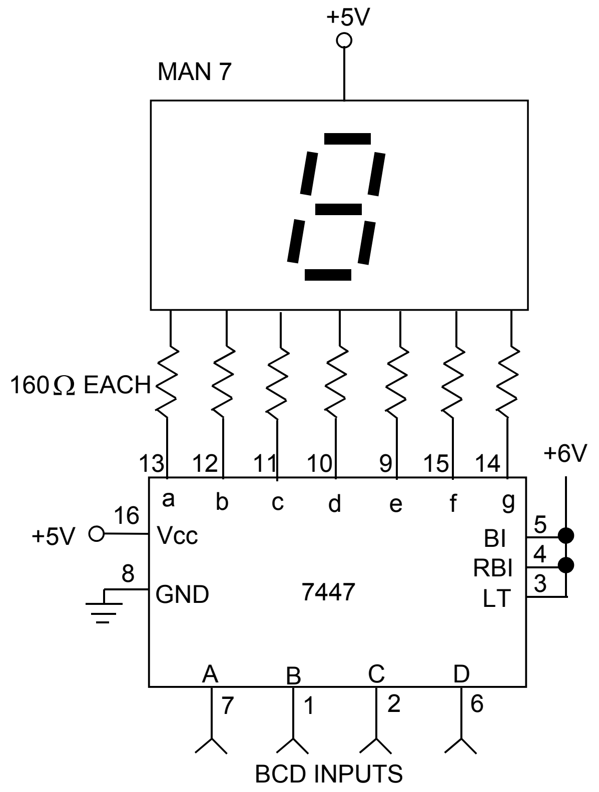
(b) 5 by 7 dot matrix display format



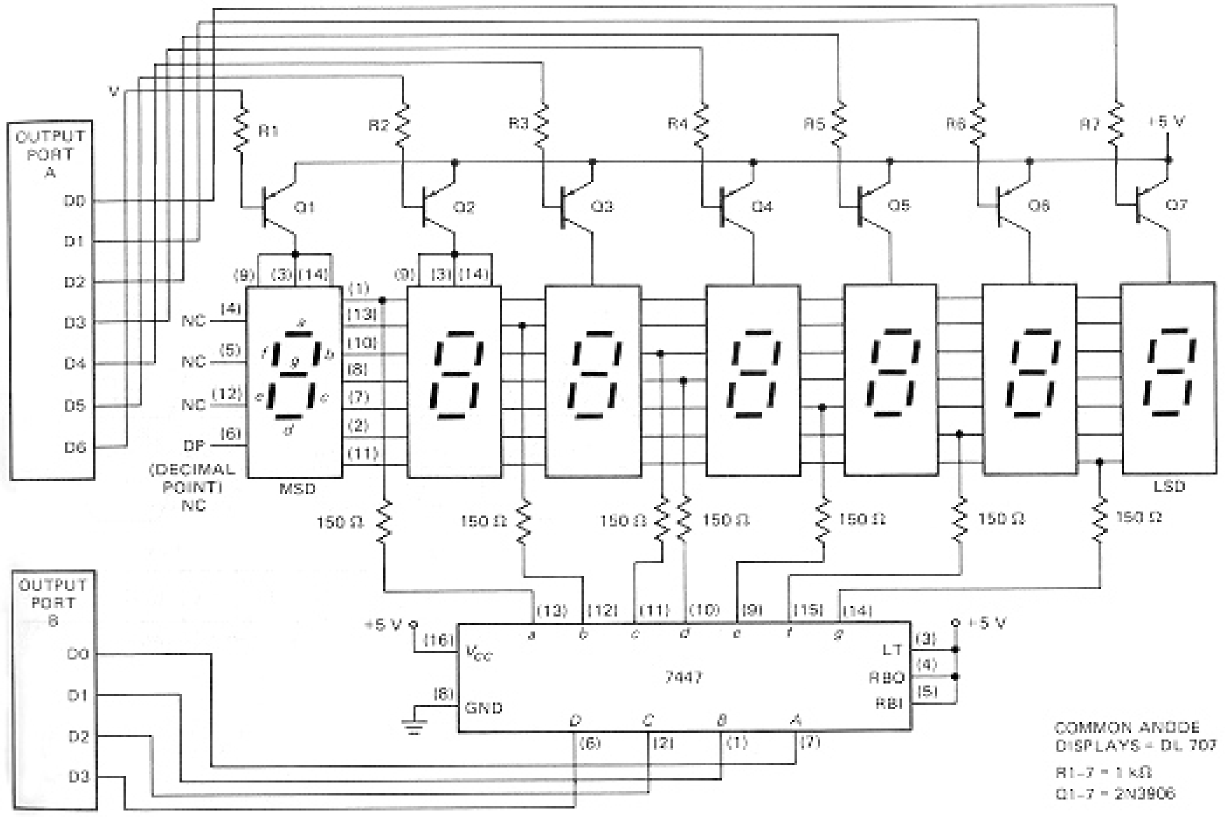
(c) 5 by 7 dot matrix circuit connections

(D Hall Fig 9-23)

Circuit Driving a 7-Segment LED with 7447



Multiplexing 7-segment Displays with a Microcomputer



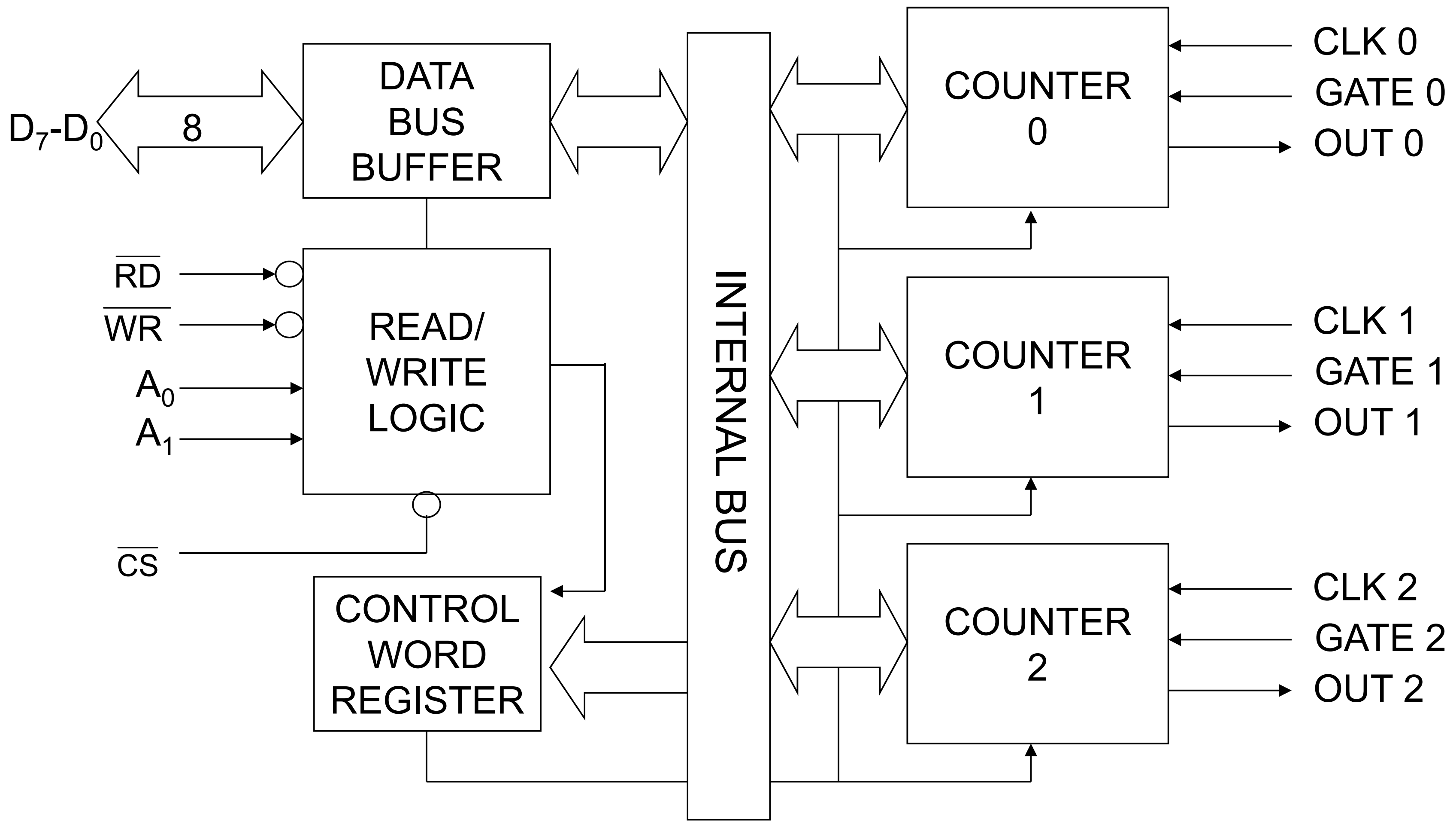
Q: Only one 7447 for all 7 digits. → display the same values for all?

A: time-multiplexing

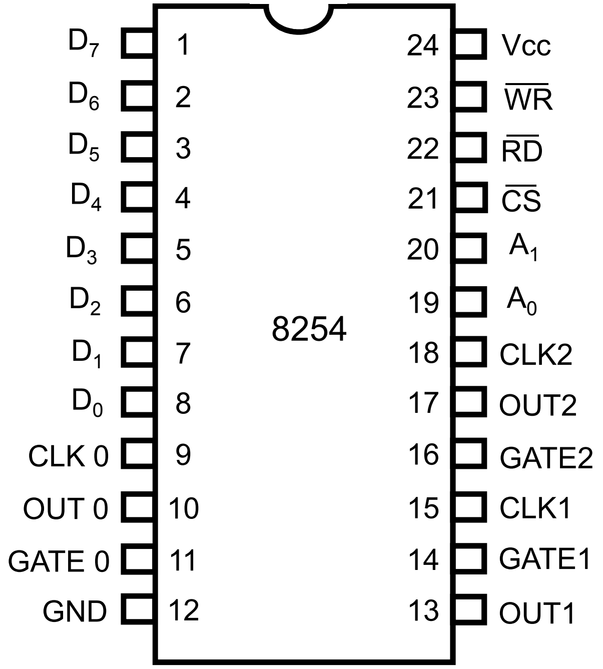
8254 Software-Programmable Timer/Counter

- 8254 is very versatile and can be used in many applications.
- There are several modes of operation for different applications.
- Intel 8253 and 8254 are almost pin-to-pin compatible except
  - The maximum input clock frequency for 8253 and 8254 is 2.6 MHz and 8 MHz, respectively. (10 MHz for 8254-2)
  - 8254 has a read-back feature which allows you to latch the count in all the counters and the status of the counter at any point. 8253 does not have this feature.
- 8254 contains three 16-bit counters. The counter can be programmed to load the initial count, start and stop the count.
- 8254 has an 8-bit interface to data bus, and two address input A_0 and A_1 to address each of the three counters.

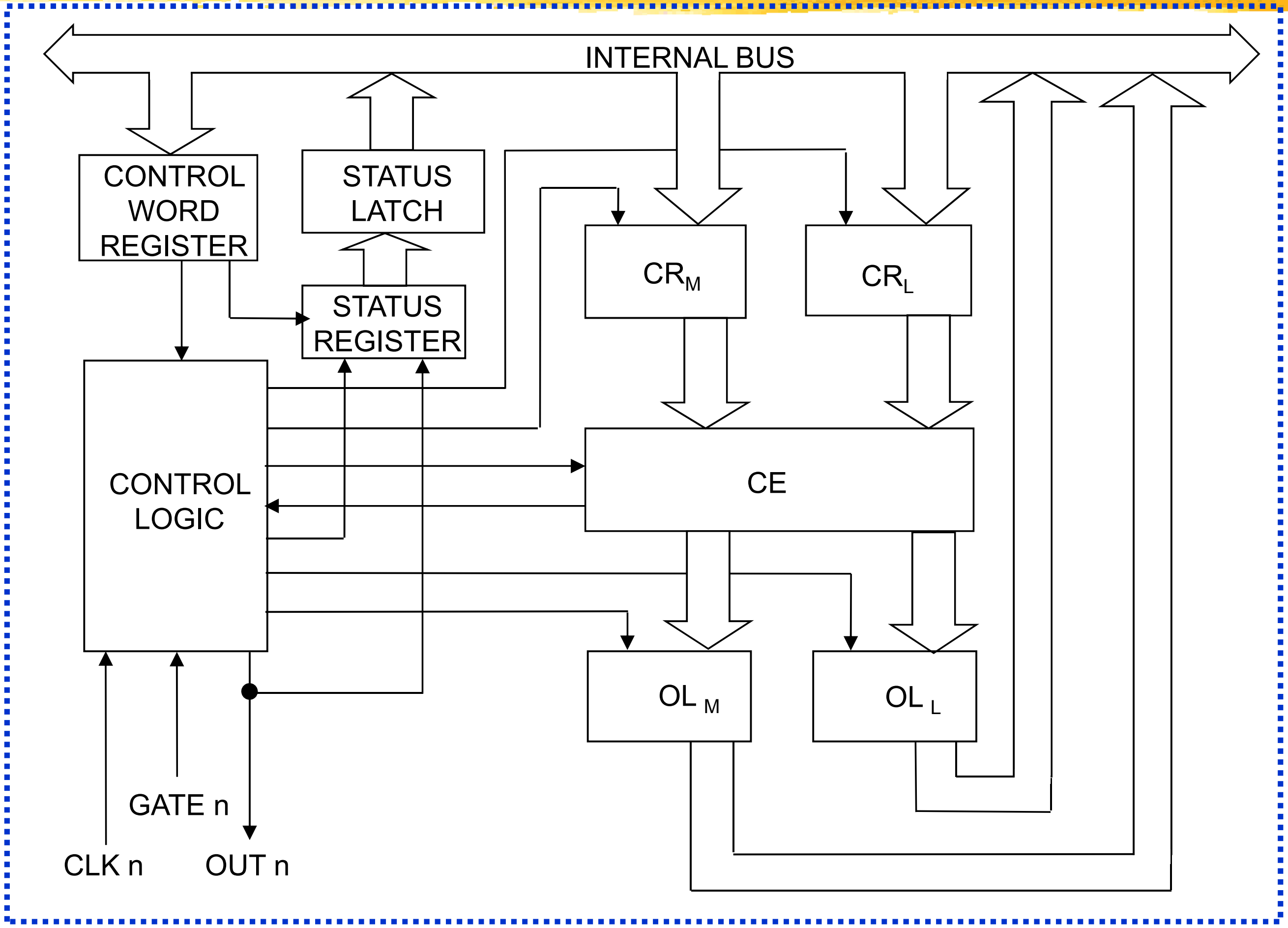
8254 Block Diagram



8254 Pin Configuration



Internal Block Diagram of a Counter



CR_M: Count Register MSB
CR_L: Count Register LSB
CE : Counter Element
OL_M: Output Latch MSB
OL_L: Output Latch LSB

Initializing 8254

- When power on, programmable peripheral devices such as 8254 are usually in *undefined* state. → need initialization
- Initialization steps: (D. Hall Fig 8-14, 8-15, 8-16)
  1. Determine the base address of the device from the address decode circuitry or the address decoder truth table.
  2. Determine the internal address for each 8254 internal device (control register, port, counters, status register, etc.)
  3. "Add" each of the internal address to the system base address to determine the system address of each device.
  4. Look in data sheet for the device for the format of the control word(s) that you have to send to the device to initialize it.
  5. Construct the control word required to initialize the device.
  6. Send the control word to the device. (programming part)
  7. In case of the 8254, you need to send the starting count to each of the counter registers.

8254 Addresses

Internal

A1	A0	SELECTS
0	0	COUNTER 0
0	1	COUNTER 1
1	0	COUNRER 2
1	1	CONTROL WORD REGISTER

System*

SYSTEM ADDRESS	8254 PART
F F 0 1	COUNTER 0 Register
F F 0 3	COUNTER 1 Register
F F 0 5	COUNTER 2 Register
F F 0 7	CONTROL Register

***Note : This system address is for the example in next page, and may vary for different circuit. In this example, 8254's A1 and A0 are connected to CPU's address pin A2 and A1. Also check 784LS138 connections.**

Example of 8254 Circuit

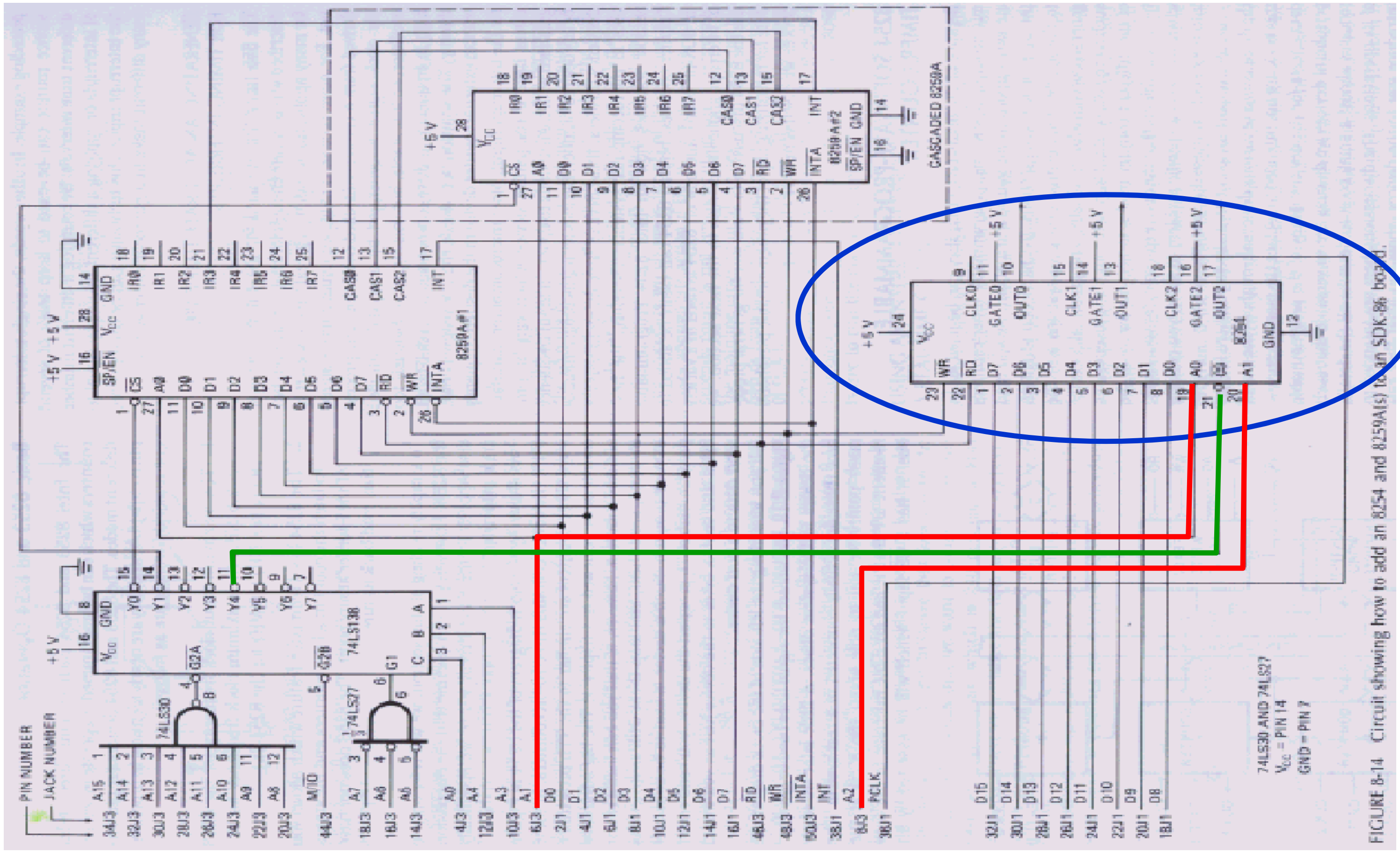


FIGURE 8-14 Circuit showing how to add an 8254 and 8259A(s) to an SDK-86 board.

74LS138

DM74LS138

Inputs					Outputs							
Enable		Select										
G1	G2 (Note 1)	C	B	A	YO	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

Q: Find the system port address for the three counters and control register.

Address Decoding via LS138

- To activate the output Y4 of LS138 in the previous example, G1 shall be high and G2(A&B) shall be low.
- From the previous figure, we note that

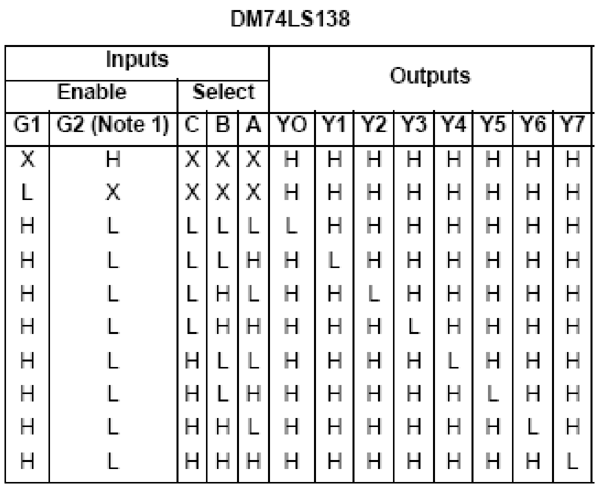
  G1=~A7*~A6*~A5

  ~G2A=~(A15*…*A8)

  ~G2B=M/~IO

- Thus LS138 will be selected when M/~IO is low and the system address is

  1111 1111 000X XXXX



DM74LS138

Inputs					Outputs							
Enable		Select										
G1	G2 (Note 1)	C	B	A	YO	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

- And Y4 is will be selected when CBA=100 or A0=1, A4=A3=0; therefore the system address for 8254 is

  1111 1111 0000 0XX1

- The remaining A2 and A1 pins are connected to A1 and A0 of 8284 for counter and register selection.
- So the system address for counter 0-2 and control register are 0FF01h, 0FF03h, 0FF05h, 0FF07h, respectively, in the previous example.

8254 Control Word Format

$A_1A_0 = 11$ $\overline{CS} = 0$ $\overline{WR} = 0$ $\overline{RD} = 1$

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

(1) SC - Select Counter

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command (see Read Operations)

8254 Control Word Format (cont.)

(2) RW - Read/ Write

RW1	RW0	
0	0	Counter Latch Command (see Read Operations)
0	1	Read/Write least significant byte only
1	0	Read/Write most significant byte only
1	1	Read/Write least significant byte first then most significant byte

(4) BCD

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

8254 Control Word Format (cont.)

(3) M - Mode

M2	M1	M0	
0	0	0	Mode 0 - Interrupt on Terminal Count
0	0	1	Mode 1 - Hardware One-Shot
×	1	0	Mode 2 - Pulse Generator
×	1	1	Mode 3 - Square Wave Generator
1	0	0	Mode 4 - Software Triggered Strobe
1	0	1	Mode 5 - Hardware Triggered Strobe

NOTE:

Don't Care bits (×) should be 0 to insure compatibility with future Intel products

A Few Possible Programming Sequences to initialize 8254

⌘ **The control words and initial counts of each counter can be sent in many different ways. But for each counter, always send the control word and then initial count(s).**

Example 1	A_1	A_0
Control Word - Counter 0	1	1
LSB of Count - Counter 0	0	0
MSB of Count - Counter 0	0	0
Control Word - Counter 1	1	1
LSB of Count - Counter 1	0	1
MSB of Count - Counter 1	0	1
Control Word - Counter 2	1	1
LSB of Count - Counter 2	1	0
MSB of Count - Counter 2	1	0

Example 2	A_1	A_0
Control Word - Counter 2	1	1
Control Word - Counter 1	1	1
Control Word - Counter 0	1	1
LSB of Count - Counter 2	1	0
MSB of Count - Counter 2	1	0
LSB of Count - Counter 1	0	1
MSB of Count - Counter 1	0	1
LSB of Count - Counter 0	0	0
MSB of Count - Counter 0	0	0

Example 3	A_1	A_0
Control Word - Counter 0	1	1
Control Word - Counter 1	1	1
Control Word - Counter 2	1	1
LSB of Count - Counter 2	1	0
LSB of Count - Counter 1	0	1
LSB of Count - Counter 0	0	0
MSB of Count - Counter 0	0	0
MSB of Count - Counter 1	0	1
MSB of Count - Counter 2	1	0

Example 4	A_1	A_0
Control Word - Counter 1	1	1
Control Word - Counter 0	1	1
LSB of Count - Counter 1	0	1
Control Word - Counter 2	1	1
LSB of Count - Counter 0	0	0
MSB of Count - Counter 1	0	1
LSB of Count - Counter 2	1	0
MSB of Count - Counter 0	0	0
MSB of Count - Counter 2	1	0

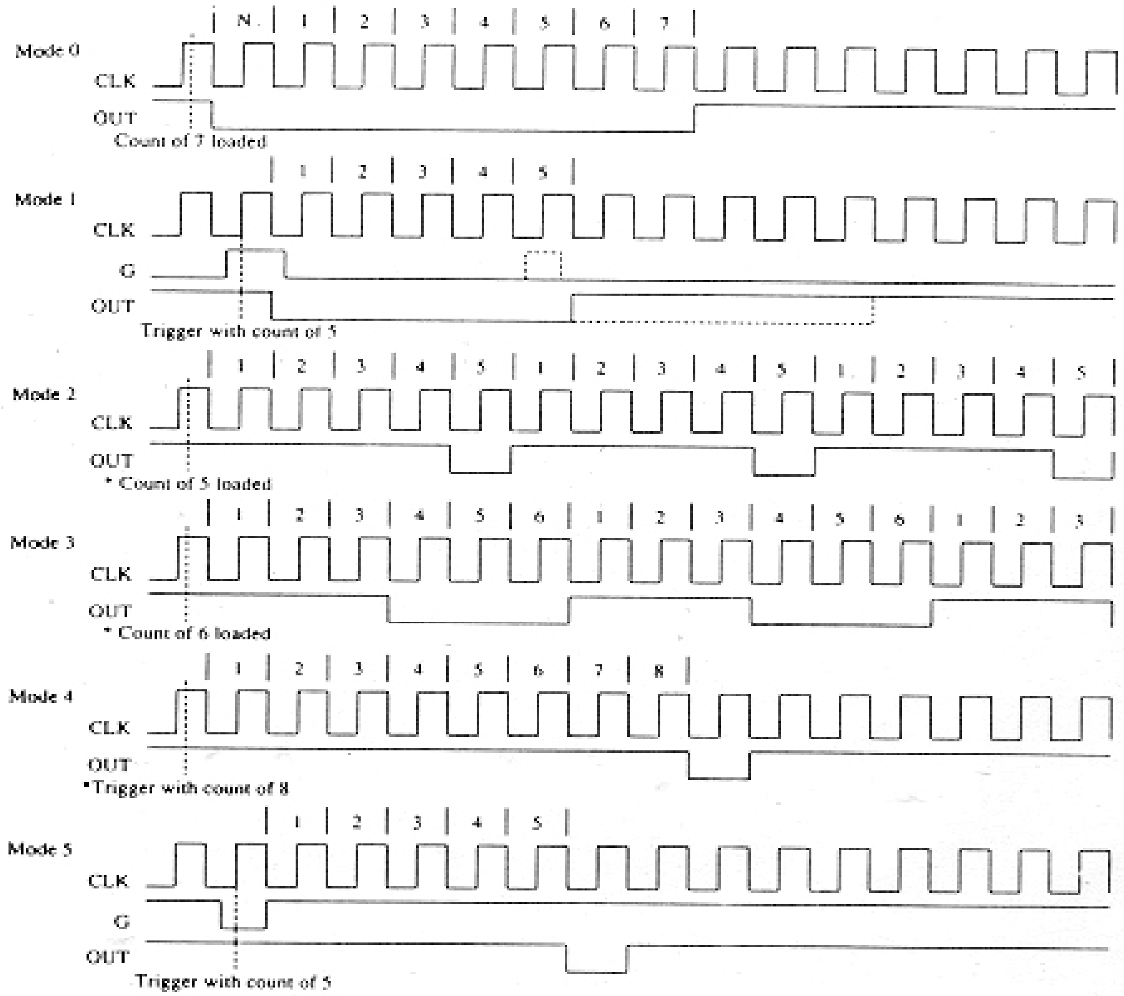
NOTE: (1) In all four examples, all Counters are programmed to read/write two-byte counts.
(2) When programmed in 2-byte, the first byte (LSB) will stop the count. The second byte (MSB) will start the counter with the new count.

Example of a control word

Task: Use counter 0 of the 8254 to divide a clock signal at 2.45 MHz to 78.6 kHz ($\rightarrow$1/32).

```
MOV   AL, 00010111B      ; Control word for counter 0
                         ; Read/write LSB only, mode 3, BCD countdown
                         ; 00   01   011   1
                         ;                           BCD countdown
                         ;                           Mode 3
                         ;                           R/W LSB only
                         ;                           Select counter 0

MOV   DX, 0FF07H         ; Point to 8254 control register (assume circuit as in pp.61)
OUT   DX, AL             ; Send control word
MOV   AL, 32H            ; Load lower byte of count
MOV   DX, 0FF01H         ; Point to counter 0 count register
OUT   DX, AL             ; Send count to count register
```

Six Modes of Operation for 8254 Programmable Interval Timer



(Brey Fig10-35)

The G input stops the count when 0 in modes 2, 3 and 4

8254 Mode Operation

- **Mode 0:** Allows 8254 counter to be used as an event counter. The output becomes a logic 0 when the control word (CW) is written and remains in "N" (null) states plus the number of programmed counts. The count starts after the count value is written. ▶
- **Mode 1:** Causes the counter to function as a retriggerable monostable multivibrator (one shot). In this mode, the G input triggers the counter so that it develops a pulse at the OUT pin that becomes a logic 0 for the duration of the counter. If the count is 10, then the OUT pin goes low for 10 clock period when triggered. If the G input occurs within the duration of the output pulse, the counter is again *reloaded* with the count and the OUT pin continues for the total length of the count. ▶
- **Mode 2:** Allows the counter to generate a series of continuous pules that are one clock pulse in width. The separation between pulses is determined by the count. (periodic pulse generator) ▶

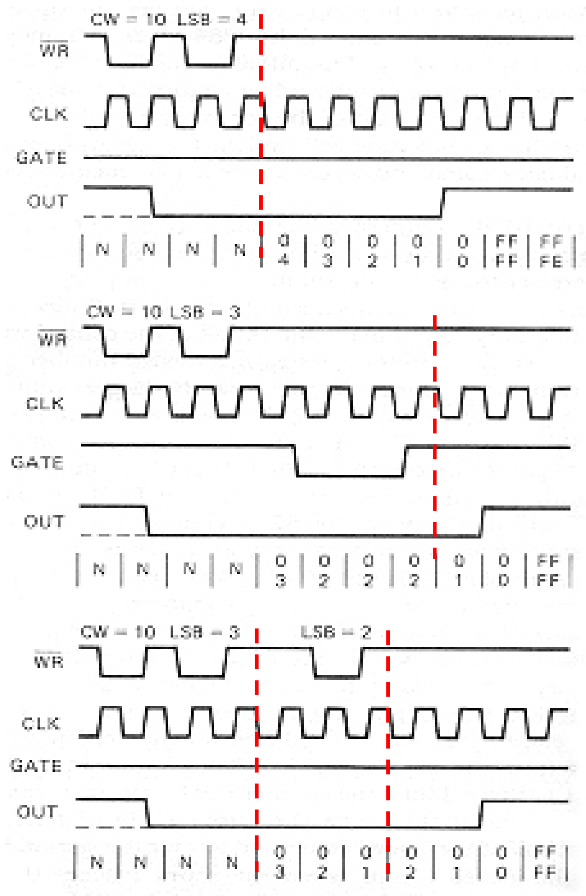
8254 Mode Operation (cont.)

- **Mode 3:** Generates a continuous *square-wave* at the OUT pin, provided the G pin is a logic 1. If the count is even, the output is high for one-half of the count and low for *one-half* of the count. If the count is *odd*, the output is high for one clocking period longer than it is low. ▶

- **Mode 4:** Allows the counter to produce a single pulse at the output. If the count is 10, the output is high for 10 clocking period and then low for 1 clocking period. (software-triggered strobe) (Writing the count to the counter register starts the count.) ▶

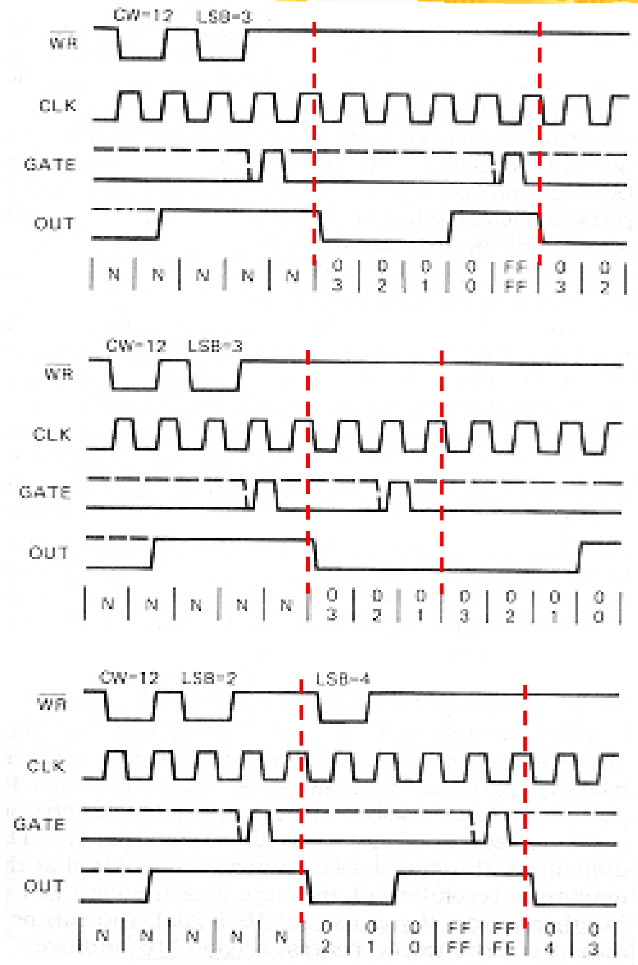
- **Mode 5:** A hardware triggered one-shot that functions as mode 4 except that it is started by a trigger pulse on the G pin instead of by software. This mode is also similar to mode 1 because it is retriggerable. (hardware-triggered strobe) (Count will be transferred to counter register after trigger goes high.) ▶

8254 Example Timing Diagrams


CW = 10 LSB = 4
WR
CLK
GATE
OUT
N N N N 04 03 02 01 00 FFFF FFFE
CW = 10 LSB = 3
WR
CLK
GATE
OUT
N N N N 03 02 02 02 01 00 FFFF
CW = 10 LSB = 3
LSB = 2
WR
CLK
GATE
OUT
N N N N 03 02 01 02 01 00 FFFF

CW=10 H → counter 0, mode 0, LSB R/W only, binary count

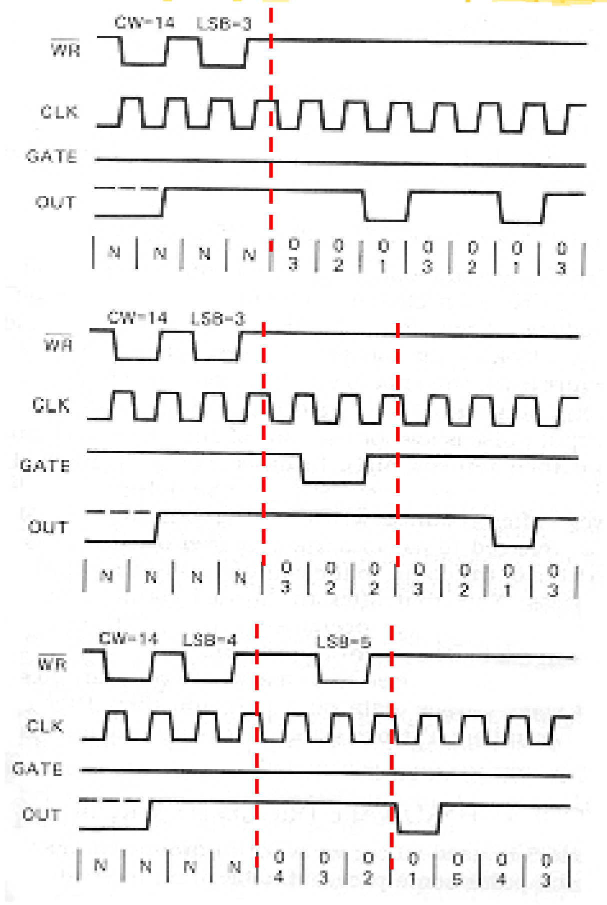
CW: control word


CW=12 LSB=3
WR
CLK
GATE
OUT
N N N N N 03 02 01 00 FFFF 03 02
CW=12 LSB=3
WR
CLK
GATE
OUT
N N N N N 03 02 01 03 02 01 00
CW=12 LSB=2
LSB=4
WR
CLK
GATE
OUT
N N N N N 02 01 00 FFFF FFFE 04 03

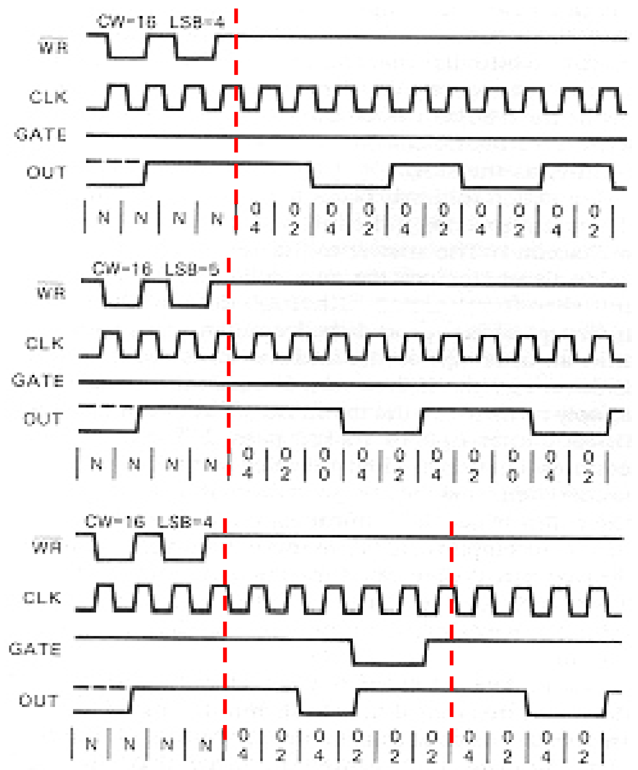
MODE 0

MODE 1

8254 Example Timing Diagrams



CW=14 LSB=3
WR
CLK
GATE
OUT
N N N N 03 02 01 03 02 01 03
CW=14 LSB=3
WR
CLK
GATE
OUT
N N N N 03 02 02 03 02 01 03
CW=14 LSB=4
LSB=5
WR
CLK
GATE
OUT
N N N N 04 03 02 01 05 04 03



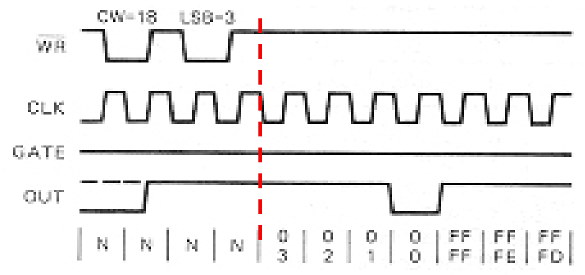
CW=16 LSB=4
WR
CLK
GATE
OUT
N N N N 04 02 04 02 04 02 04 02 04 02
CW=16 LSB=5
WR
CLK
GATE
OUT
N N N N 04 02 00 04 02 04 02 00 04 02
CW=16 LSB=4
WR
CLK
GATE
OUT
N N N N 04 02 04 02 02 02 04 02 04 02

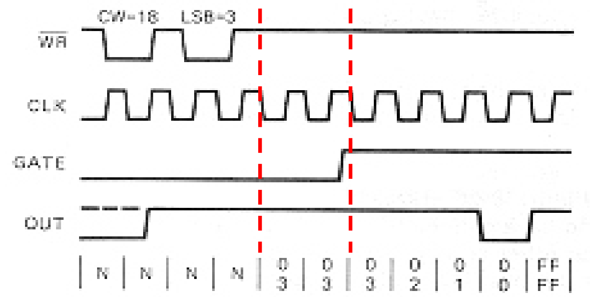
New count value will not be reloaded until end of the current count or until the gate is re-enable.

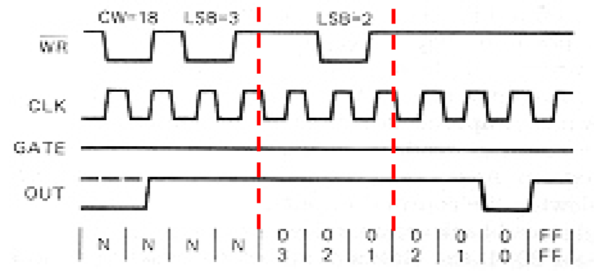
MODE 2
(periodic pulse generator)

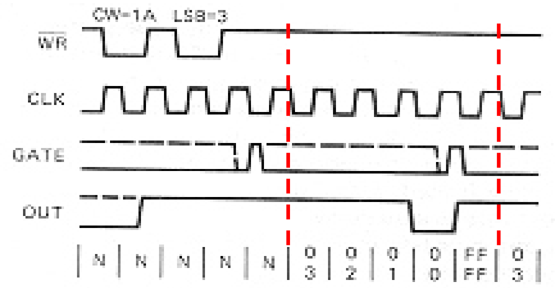
MODE 3
(square-wave generator)

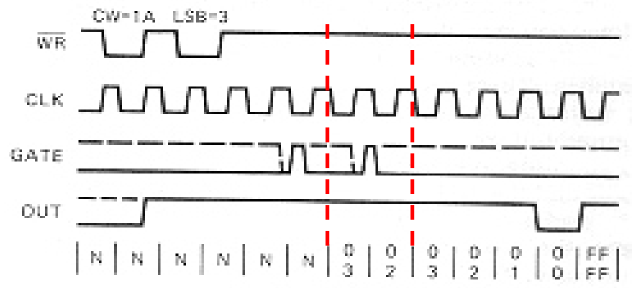
8254 Example Timing Diagrams

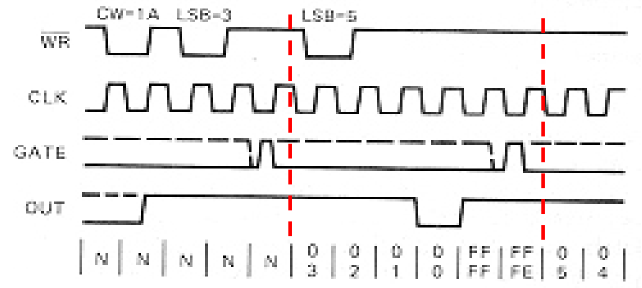












MODE 4
(software-triggered strobe)

MODE 5
(hardware-triggered strobe)

Gate Pin Operations Summary

Signal Status Modes	Low or Going Low	Rising	High
0	Disables Counting	-----------	Enables Counting
1	-----------	1) Initializes Counting 2) Resets Output after Next Clock	-----------
2	1) Disables Counting 2) Sets Output Immediately High	Initializes Counting	Enables Counting
3	1) Disables Counting 2) Sets Output Immediately High	Initializes Counting	Enables Counting
4	Disables Counting	-----------	Enables Counting
5	-----------	Initializes Counting	-----------

Minimum and Maximum Initial Counts

Mode	Min Count	Max Count
0	1	0
1	1	0
2	2	0
3	2	0
4	1	0
5	1	0

NOTE:
0 is equivalent to 2^{16} (0FFFFh+1) for binary counting and 10^4 (9999+1) for BCD counting

Reading a Counter's <u>count</u> and <u>status</u>

count

- Each counter has an internal latch that can be read with the read counter port operation.
- The latch value usually follows the count.
- When the count is needed, the latch can remember the count by programming the *counter latch control word*, which causes the content of the counter to be held in a latch until it is read.
- When there are more than one counters to be read, we use the *read-back control word*.

status

- If the status register is to be latched, then $\overline{\text{STATUS}}$=0. (pp.79)
- The status register shows the state of the OUT pin: whether the counter is at its null state, and how the counter is programmed.

Counter Latching Command Format

$A_1A_0 = 11$; $\overline{CS} = 0$; $\overline{RD} = 1$; $\overline{WR} = 0$

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	×	×	×	×

SC1,SC0 - Specify counters to be latched

SC1	SC0	Counter
0	0	0
0	1	1
1	0	2
1	1	Read-Back Command

for latching single counter only

for latching multiple counters

D5, D4 - 00 designates Counter Latch Command

× - don't care

NOTE: Don't Care bits (×) should be 0 to insure compatibility with future intel products

Q: How do we know it is counter latching command?

Read-Back Command Format

For latching multiple counters:

$A_1A_0 = 11$ $\overline{CS} = 0$ $\overline{RD} = 1$ $\overline{WR} = 0$

D7	D6	D5	D4	D3	D2	D1	D0
1	1	$\overline{\text{COUNT}}$	$\overline{\text{STATUS}}$	CNT 2	CNT 1	CNT 0	0

D5: 0 = Latch the count of selected counter(s)
D4: 0 = Latch the status of selected counter(s)
D3: 1 = Select counter 2
D2: 1 = Select counter 1
D1: 1 = Select counter 0
D0: Reserved for future expansion. Must be 0

8254 Status Register

The meaning of each bit in the 8254's status register:

D7	D6	D5	D4	D3	D2	D1	D0
OUTPUT	NULL COUNT	RW1	RW0	M2	M1	M0	BCD

D7	1 = OUT Pin is 1 0 = OUT Pin is 0
D6	1 = NULL Count 0 = Count available for reading
D5-D4	Read/Write operation
D3-D1	Counter mode
D0	Logic 1 for BCD counter

An Example of Read-Back Command Sequence and Its Results

$\overline{\text{Count}}$ (D5)

Assume these commands are sent sequentially

D7	D6	D5	D4	D3	D2	D1	D0	Description	Result
1	1	0	0	0	0	1	0	To Read back count and status of counter 0	Count and status latched for counter 0
1	1	1	0	0	1	0	0	To Read back status of counter 1	Status latched for Counter 1
1	1	1	0	1	1	0	0	To Read back status of counter 2, 1	Status latched for counter 2, but not* counter 1
1	1	0	1	1	0	0	0	To Read back count of counter 2	Count latched for counter 2
1	1	0	0	0	1	0	0	To Read back count and status of counter 1	Count latched for counter 1, but not* status
1	1	1	0	0	1	0	0	To Read back status of counter 1	Command ignored; status already latched for counter 1

$\overline{\text{Status}}$ (D4)

*: already latched

Read-Back Command Example (cont.)

- If multiple count and/or status read-back commands are issued to the same counter(s) without intervening reads, all but the first commands are ignored.

- After the counter register or status register is latched, the content can be read by instruction

```
    IN AL, counter_n_address        ; counter_n_address is the address
                                    ; of counter 0, 1, or 2
```

- If both the count and status of a counter are latched, the first read operation of that counter will return the latched status, regardless of what was latched first. The next one or two read will return the latched count.
- Subsequent read will return unlatched count (just follow the counter).

FAQ

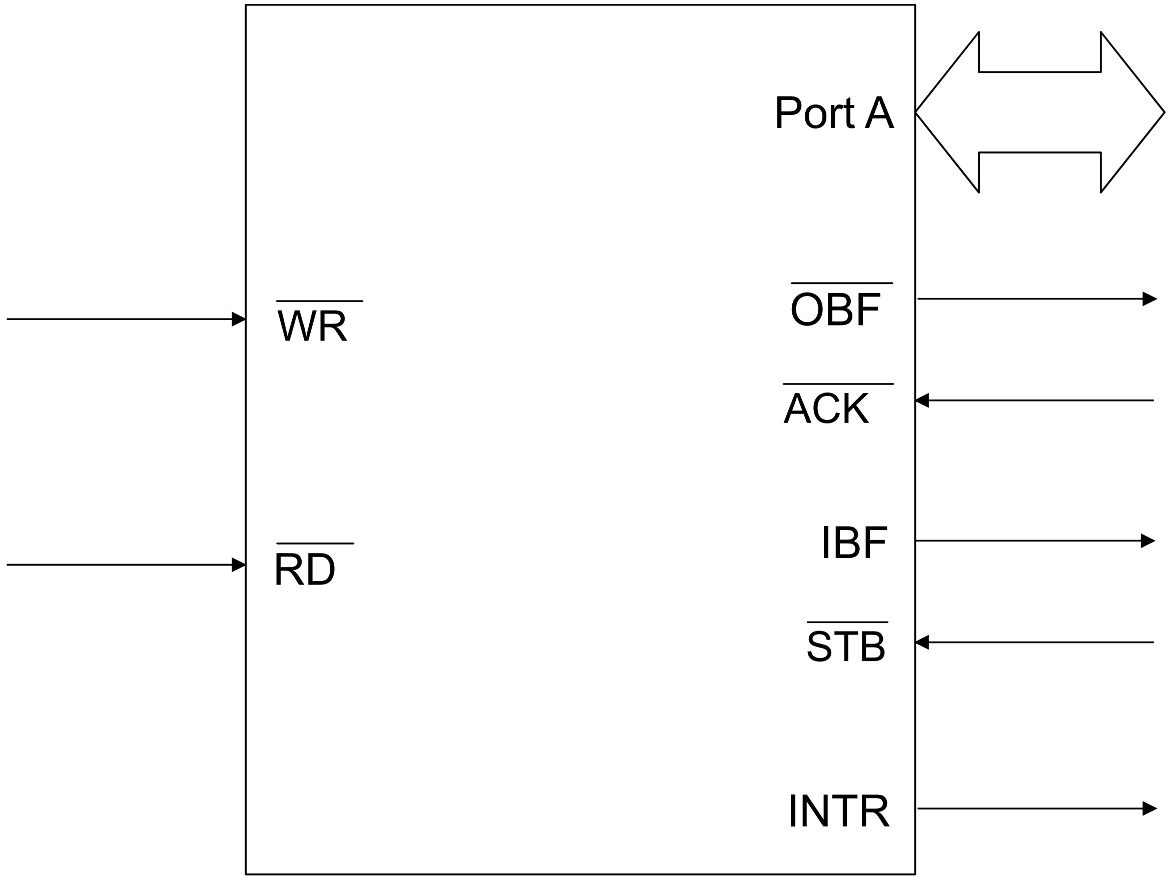
- **Q:** When does the counter decide whether to continue the count? For example, in mode 0, the middle case, the G becomes low during the second half cycle of the (03)cycle. (See Fig 8-18 or Lecture note part 10, pp70). Why does the count still go low in the next cycle.
- **A:** (see Intel data book) The GATE input is always sampled on the rising edge of CLK. In mode 0, 2, 3 and 4, the the GATE input is level-sensitive, and the logic level is sampled on the rising edge of CLK. In mode 1,2,3, and 5 the rising edge of GATE (trigger) sets an edge-sensitive flip-flop in the counter. This flip-flop will then be sampled on the next rising edge of CLK; the flip-flop is reset immediately after it is sampled. In this way, the G does not need to be maintained at high until the next rising edge if CLK. Note that mode 2 and 3 are both edge- and level-sensitive, both the level and the rising edge of G has effect on the counter output. (see page 72 about gate pin operation summary).

- **Q:** In mode 4, it generates a low pulse when it reaches 0 and continue to count down from 0FFFFh. Will it generate another low pulse when it reaches 0 count again?
- **A:** In mode 4, only one low pulse is generated while zero count is reached and there is no pulse from the OUT till next write to counter latch operation,i.e. OUT keeps high all the way. The counter may continue counting.

FAQ

- **Q:** What is the purpose of INTR in the diagram on pp30? Is the INTR sent to 8088? But isn't the 8088 outputting data low? why should 8255 interrupt it?
- **A:**Yes. 8088 is outputting data to the external device through 8255. Here, the key is 8255. When 8088 has finished writing data into 8255. The "out dx, al" command will be completed regardless whether the external device has received the data from 8255 successfully. So, if 8088 needs to output data to the device again, how can it know if the device is free? Well, two ways: (i) The first one is to loop and check the state of 8255 OBF pin to determine if the previous output is successful. (ii) the other method is to ask 8255 to "report" immediately to 8088 when the operation has finished. And this is exactly what the INTR in the diagram does. When the external device acknowledges 8255 that data has been received, 8255 will pull high the INTR which if 8088 has interrupt flag set, it will be interrupted and then can proceed to the next operation

Mode 2 Operation of 8255A



Mode 2 Operation of 8255A

1. $\overline{\text{WR}}$ is asserted to indicate CPU is outputting data using OUT instruction.
2. Output data is latched in the output buffer by 8255, and $\overline{\text{OBF}}$ is asserted.
3. Input data from external component appears at Port A.
4. $\overline{\text{STB}}$ is asserted to indicate external component is inputting data.
5. Input data is latched in the input buffer by 8255, and IBF is asserted.
6. $\overline{\text{STB}}$ is unasserted, and input data from external component can be removed.
7. INTR is asserted to indicate the data input has been completed.
8. $\overline{\text{ACK}}$ is asserted by the external component, indicating it is ready to receive the output data from 8255.
9. Output data for external component appears at Port A.
10. $\overline{\text{ACK}}$ is unasserted by the external component, indicating the output data has been received.
11. $\overline{\text{OBF}}$ is unasserted indicating the data in output buffer is sent.
12. Output data for external component is removed at Port A.
13. INTR is asserted to indicate the data output has been completed.
14. $\overline{\text{RD}}$ is asserted to indicate CPU wants to read the input data using IN instruction.
15. $\overline{\text{RD}}$ is unasserted by the CPU after the input data is received.
16. IBF is unasserted indicating the data in input buffer is read.