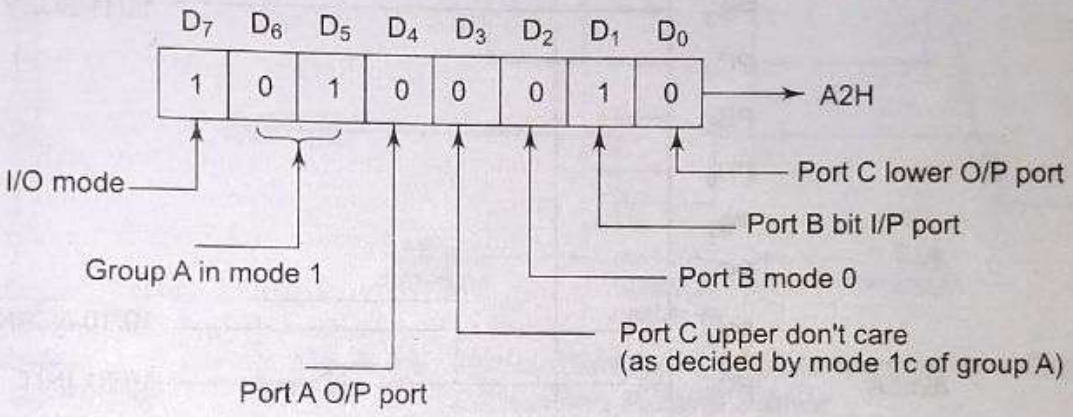
Table 5.12 (*Contd.*)

Signal Pin No.	*Return Pin No.*	*Signal*	*Direction*	*Description*
13	—	SLCT	OUT	This signal indicates that the printer is in the selected
14	—	$\overline{\text{AUTO FEED XT}}$	IN	With this signal being at "low" level, the paper is automatically fed one line after printing. (The signal level can be fixed to "low" with DIPSW Pin 2-3 provided on the control circuit board.)
15	—	NC		Not used.
16	—	0V		Logic GND Level.
17	—		CHASIS GND	Printer chasis GND. In the printer, the chasis GND and the logic GND are isolated from each other.
18	—	NC	—	Not used.
19–30	—	GND	—	"Twisted-Pair Return" signal; GND level.
31	—	$\overline{\text{INIT}}$	IN	When the level of this signal becomes "low" the printer controller is reset to its initial state and the print buffer is cleared. This signal is normally at "high" level, and its pulse width must be more than 50 µs at the receiving terminal.
32		$\overline{\text{ERROR}}$	OUT	The level of this signal becomes "low" when the printer is in "Paper End "state, "Offline" state and "Error" state.
33	—	GND	—	Same as with pin numbers 19 to 30.
34	—	NC	—	Not used.
35				Pulled up to + 5 Vdc through 4.7 k-ohms resistance.
36	—	$\overline{\text{SLCT IN}}$	IN	Data entry to the printer is possible only when the level of this signal is "low". (Internal fixing can be carried out with DIP SW 1-8. The condition at the time of shipment is set "low" for this signal.)

Note:
1. "Direction" refers to the direction of signal flow as viewed from the printer.
2. "Return" denotes "Twised-Pair Return" and is to be connected at signal-ground level. When wiring the interface, be sure to use a twisted-pair cable for each signal and never fail to complete connection on the return side. To prevent noise effectively, these cables should be shielded and connected to the chassis of the system unit.
3. All interface conditions are based on TTL, level. Both the rise and fall times of each signal must be less than 0.2 µs.
4. Data transfer must not be carried out by ignoring the $\overline{\text{ACKNLG}}$ or BUSY signal. (Data transfer to this printer can be carried out only after interfacing the $\overline{\text{ACKNLG}}$ signal or when the level of the BUSY signal is "low".)
5. Remaining pins on the connector are no connection pins.
6. x-not available in 25 pins connector.

Printer Operation The printer interface connections with 8255 and the printer connector shown in Fig. 5.31 and Fig. 5.32 respectively. First of all the printer should be initialised by sending a 50 µs (minimum) pulse on the $\overline{\text{INIT}}$ pin of the printer. Then the BUSY pin is to be checked to confirm if the printer is ready. If this signal is low, it indicates that the printer is ready to accept a character from the CPU. Port pins of 8255 may

not have sufficient drive capacity to drive the printer input signals so that the open collector buffers 74LSOY are used to enhance the drive capacity. When this happens the ASCII code of the character to be printed is sent on the eight parallel port lines. Once the data is sent on eight parallel lines, the $\overline{\text{STROBE}}$ signal is activated after at least 0.5 µs, to indicate that the data is available on the eight data lines. The falling edge of the $\overline{\text{STROBE}}$ signal causes the printer to make its BUSY pin high, indicating that the printer is busy. After a minimum period of 0.5 µs, the $\overline{\text{STROBE}}$ signal can be sent high. The data must be valid on the data lines for at least 0.5 µs after the $\overline{\text{STROBE}}$ signal goes high. After receiving the appropriate $\overline{\text{STROBE}}$ pulse, the printer starts the necessary electromechanical action to print the character and when it is ready to receive the next character, it asserts its $\overline{\text{ACKNLG}}$ signal low approximately for 5 ms. The rising edge of the $\overline{\text{ACKNLG}}$ signal indicates to the computer that it is ready to receive the next character. The rising edge of the $\overline{\text{ACKNLG}}$ signal also resets the BUSY signal from the printer. A low on the BUSY pin further indicates that the printer is ready to accept the next character. The $\overline{\text{ACKNLG}}$ and BUSY signals can be used interchangeably for handshaking purposes. The waveforms for the above printer operation are shown in Fig. 5.33.



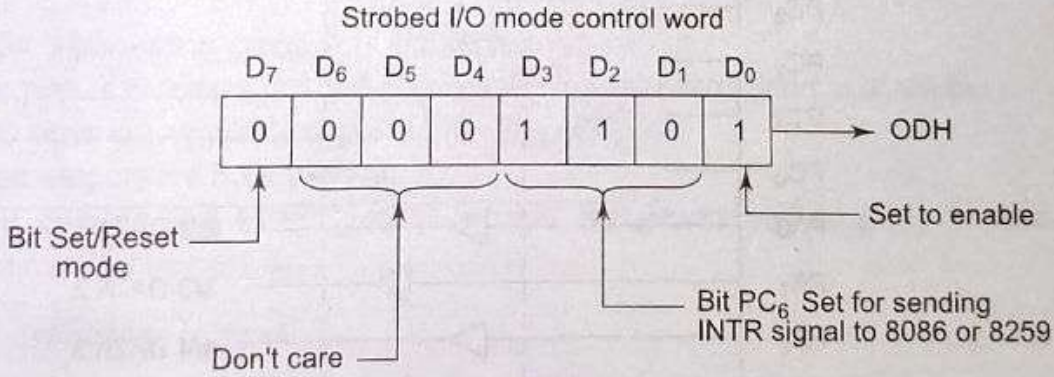


Fig. 5.30 *Bit Set/Reset Control Word*

```
        MOV BL,AL          ; Get the ASCII code in BL.
        MOV AL,0A2H        ; Control word for 8255
        OUT 0F6H,AL        ; Load CWR with the control word.
BUSY:   IN AL,0F2H         ; Read printer status from the BUSY pin.
        AND AL,08H         ; Mask all bits except PB3
        JNZ BUSY           ; If AL#0, printer is busy. Wait till
                           ; it becomes free.
        MOV AL, BL         ; Get the character for printer in AL
        OUT 0F0H,AL        ; Send it to the port for the printer
        NOP                ; Wait for some time
        MOV AL,08 H        ; Pull STROBE low
```

```
OUT 0F6H          ; Reset PC4
NOP               ; Wait
MOV AL,09 H       ; Raise STROBE high
OUT 0F6H          ; SET PC4
HLT
```

Program 5.8 *ALP for Printing a Character for Problem 5.14*

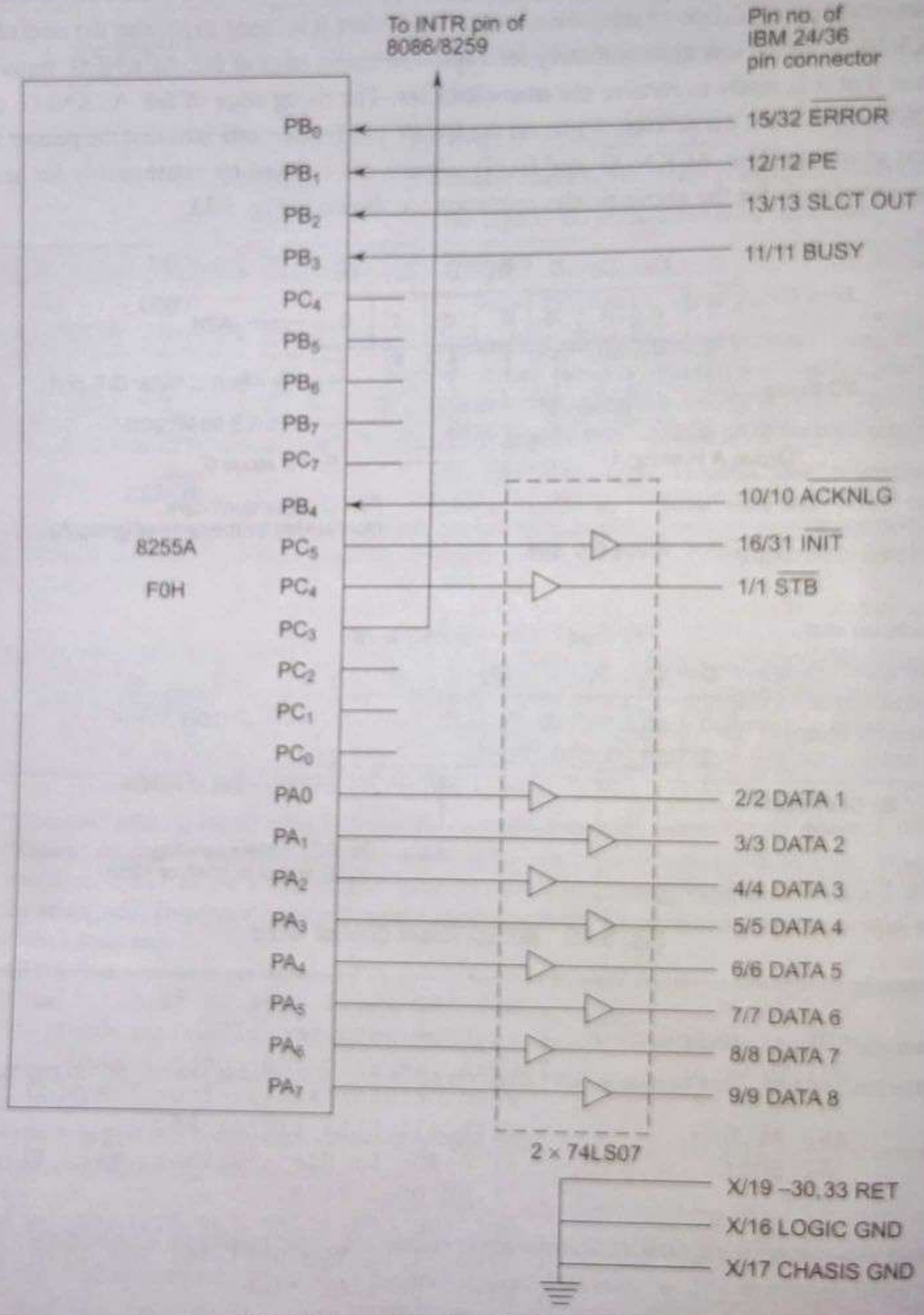


Fig. 5.31 *Printer Interface with 8255*

(c) MODE 2 (Strobed bidirectional I/O) :-

* provides an additional feature of communicating with a peripheral device on a 8-bit data bus.
* Single 8-bit port in Group A is available.
* 8-bit port is bidirectional and 5-bit control port is available
* Three I/O lines are available at port C (PC_2 to PC_0).
* Inputs and outputs are latched.
* 5-bit control port C (PC_3 - PC_7) is used for generating/accepting handshake signals for 8-bit data transfer on Port - A

Control signal definitions in mode 2 :-

INTR (Interrupt Request) : used to interrupt the microprocessor to ask for next data byte to/from it.

for output operations :

- $\overline{OBF}$ (Output Buffer full) : $\overline{OBF} = 0$ ⇒ CPU has written data to port A.
- $\overline{ACK}$ (Acknowledge) : $\overline{ACK} = 0$ ⇒ confirms the receipt of previous data byte by the destination and next data byte may be sent by the processor.
- INTE 1 (A flag associated with $\overline{OBF}$) : can be controlled by bit set/reset mode with PC_6.

for input operations :

- $\overline{STB}$ (strobe input) : $\overline{STB} = 0$ ⇒ data comes to input latches of 8255.
- IBF (Input buffer full) : IBF = 1 ⇒ data has been received into the input buffer.

⇒ Diagrams to be attached.

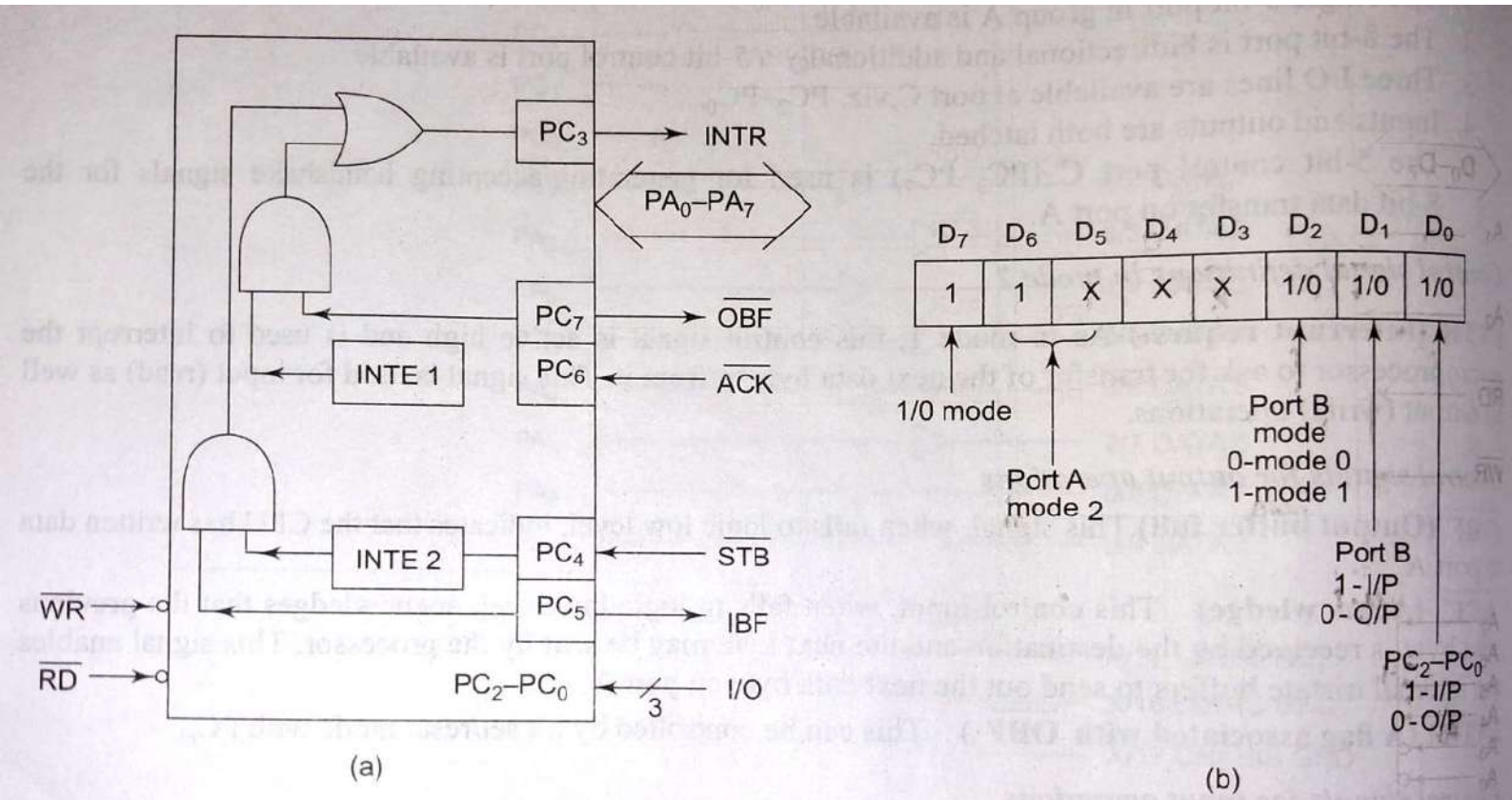


Fig. 5.35 *(a) Mode 2 pins (b) Mode 2 control word*

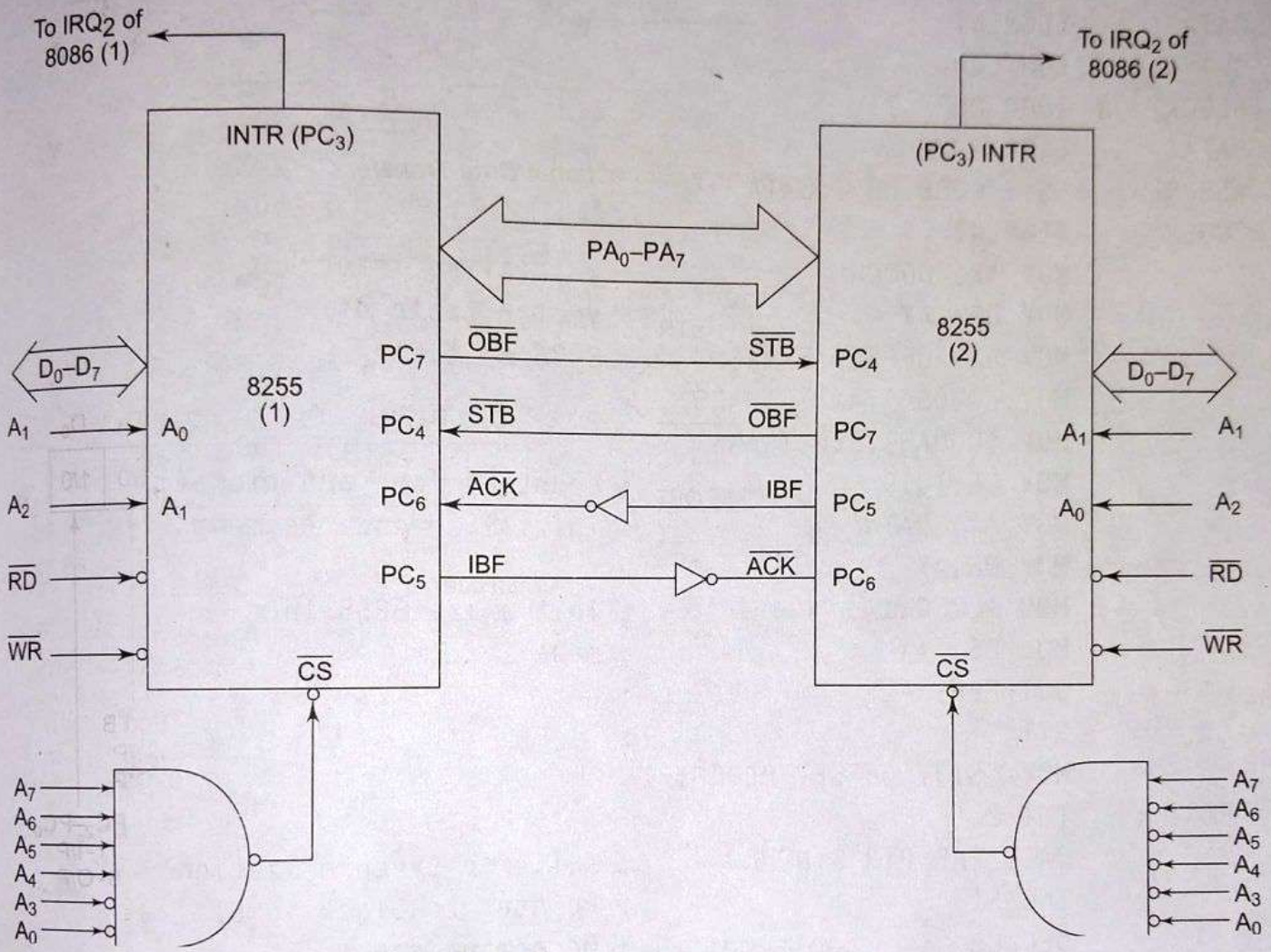


Fig. 5.36 *Interconnections between the two 8255s in Mode 2 for Problem 5.16*

Problem 5.15

An 8086 system with a 8255 interfaced at port A address F0H, has a block of 100 data bytes stored in it. Another 8086 system with another 8255 interfaced at port A address 80H has another block of 100 data bytes stored in it. Interchange these blocks of data bytes between the two 8086 systems. Draw the necessary hardware scheme and write the necessary sequence of instructions. Both systems run on the same CLK rate.

Solution The complete hardware schematic is shown in Fig. 5.36. The INTR pin of 8255 in mode 2 is applied to the respective 8086 processor at NMI pin. The inverted IBF signal of the first 8255 is connected to $\overline{\text{ACK}}$ of the second and the inverted IBF signal of the second 8255 is connected to $\overline{\text{ACK}}$ of the first 8255, so that input buffer full signal of one 8255 acknowledges the receipt of data byte sent by the other. The $\overline{\text{OBF}}$ signal of one 8255 is connected to $\overline{\text{STB}}$ signal of the other 8255 and vice versa, so that the output buffer full signal of an 8255 informs the other 8255 that the data is ready on the data bus for it. The 8-bit data bus, i.e. PA_0–PA_7 of the two 8255s are connected with each other.

```
          ; This program transmits parallel data byte by byte to another
            system through
          ; 8255 IN MODE 2.
DATA      SEGMENT
CW1       EQU COH
BLOCK1 DB 100D DUP (?)
DATA      ENDS
ASSUME    CS : CODE,DS : DATA
CODE      SEGMENT
          MOV AX, 0000H              ; Initialise interrupt
          MOV DS, AX                 ; vector table of
          MOV AX, OFFSET TRANS       ; 8086 at table
          MOV [0008H],AX ; TRANS.
          MOV [000AH],SEG TRANS
          MOV CL,101D                ; count CL (one additional)
          MOV AX, DATA               ; Initialise data segment
          MOV DS,AX
          MOV AL, CW1                ; Initialise 8255 in
          MOV DS, AX                 ; mode 2
          OUT F6H, AL                ;
          STI                        ;
          MOV [SI], OFFSET BLOCK1-1
TRANS :   INT 2
          CALL FAR PTR SYNCHRO       ; Wait for synchronization
          INC SI                     ; Pointer to block in SI
          DEC CL                     ; Decrement count
```

```
        JZ STOP                     ; If = 0, then stop else
        MOV AL,[SI]                 ; Go for transfer of the next
        OUT FOH,AL                  ; byte and out it to port A
WAIT :  JMP WAIT                    ; Wait for acknowledgement
STOP :  HLT                         ; Stop if the complete block
CODE    ENDS                        ; is transferred
        END
```

Program 5.9(a) *Transmitter ALP for Problem 5.15*

```
        ; The receiver program receives data bytes
        ; transmitted by the other system and stores
        ; them in the array as asked in the program.
STACK       SEGMENT
        STACKD DB   500H
STACK       ENDS
DATA        SEGMENT
        CW2 EQU COH
        BLOCK2 DB 100D DUP (?)
DATA        ENDS
CODE        SEGMENT
ASSUME CS : CODE, DS : DATA ,SS: STACK
          MOV AX, STACK
          MOV SS, AX
          MOV AX, 0000H             ; Initialise interrupt
          MOV DS, AX                ; vector table
          MOV [0008H],OFFSET NEXT
          MOV [000AH],SEG NEXT
          MOV CL,101 D              ; count for bytes
          MOV AX,DATA               ; Initialise data segment
          MOV DS,AX
          MOV AL,CW2                ; Initialise 8255 in mode 2
          OUT 86H,AL                ; to receive data
          MOV SI,OFFSET BLOCK2-1    ; Point to block 2
WAIT :    JMP WAIT                  ; to store received data and wait
NEXT :    INC SI                    ; Increment SI, point to start
          DEC CL                    ; of block 2 and decrement
                                    ; COUNTER
          JZ STOP
          IN 80H
          MOV [SI],AL
          JMP WAIT
STOP :    HLT
CODE      ENDS
    END
```

Program 5.9(b) *Receiver ALP for Problem 5.15*

After the transmitter transmits the data bytes, the receiver receives it and stores it in the array BLOCK 2 but acknowledge $\overline{ACK}$ is sent to the transmitter immediately. Hence the transmitter should wait before sending the next data byte to the receiver so as to give it the sufficient time to read, store and upgrade count for the received data. Transmitter runs its delay procedure SYNCHRO to solve this problem.

```
SYNCHRO    PROC   FAR
     INC DI              ; All the instructions in this routine are dummy
     DEC CH              ; instructions, just to cause the same delay as the
     JZ OK               ; requires takes, to be prepared for accepting
                                 the next data
                         ; byte after getting interrupted by the
                                      transmitter.
OK : IN AL,0F3H
     MOV [DI] AL         ; These are comparable with the
     IRET                ; corresponding receiver program
SYNCHRO    ENDP          ; instructions.
```

Program 5.10 *ALP for Synchronization Delay*

It may be noted that for the execution of this program, procedure SYNCHRO must be entered with the transmitter program before the END statement.

These programs should be entered in both the 8086 systems. The procedure SYNCHRO is to be entered with the transmitter program as it is called by it. Thus after entering the complete set of these programs into the two 8086 systems, run the receiver program on the receiver 8086 kit. It will initialise the receiver 8086 IVT, its 8255 in receiver mode and wait for the interrupt from the transmitting terminal. Now run the program on the transmitter 8086 kit. This program initialises the transmitter 8086 IVT. Its 8255, in transmitter mode, goes on transmitting data byte by byte and waits for the delay caused by the SYNCHRO before each byte is transmitted so as to give sufficient time to the receiver 8086 to read data, store it in array and upgrade counters and pointers.

Note that the procedure SYNCHRO contains all the dummy instructions. They do not serve any purpose for the algorithm, but just provide their execution delay.The instructions in the procedure are the same as the instructions in the receiving program, from label NEXT to the JMP WAIT instruction. This is not merely a coincidence but an accurate way of providing the required delay for the receiver. As both the 8086 CPUs run at the same clock speed, each of the instructions, which from the procedure and correspondingly those from the receiver program will take the same time for execution. Thus the transmitter will provide the exact delay as required by the receiver.

KEYBOARD / DISPLAY CONTROLLER (8279) :-

* 8255 was also used in interfacing keyboards and displays but the disadvantage of the method is that the CPU has to refresh the display and check the status of the display periodically using polling technique. This leads to wastage of CPU cycles and hence the throughput.

* 8279 is a general purpose keyboard display controller that simultaneously drives the display of a system and interfaces a keyboard with the CPU, leaving CPU free for other tasks.

* Keyboard activity scanned in interrupt mode or polled mode. If any key is pressed, then the code is sent to the CPU.
* It also transmits the data from the CPU to the display device
* Above 2 functions are performed by the controller in repititive fashion without involving the CPU.

ARCHITECTURE OF 8279 :-

* The chip provides a set of 4 scan lines and eight return lines for interfacing keyboards.
* Also has a set of eight output lines for interfacing display

The various functional blocks of 8279 are as follows :-

① I/O Control and Data Buffers :-
- controls flow of data to/from 8279.
- data buffers interface external bus of system with internal bus of 8279.
- I/O section is enabled when $\overline{RD} = 0$.

② Control and Timing Registers and Timing control :-
- These registers store the keyboard and display modes programmed by the CPU.
- Registers are written when $A_0 = 1$ and $\overline{WR} = 0$.
- Timing and control unit controls the timings of the operations of the ckt.
- scan counter divides operating frequency to derive scan keyboard

and scan display frequencies.

③ Scan Counter:-

- scan key matrix and refresh the display.
- operates on 2 modes: (a) Encoded mode
  (b) Decoded mode.
- Encoded mode: provides a binary count that has to be externally decoded to provide scan lines for keyboard and display.
- Decoded mode: counter internally decodes 2 least significant bits and provides a decoded o/p i.e 1 out of 4 scan lines.
- keyboard and display both are in same mode at a particular instant.

④ Return Buffers and Keyboard Debounce and Control:-

- scans for key closure row-wise.
- If detected, keyboard debounces the key entry.
- After debounce period (10 mS) if key continues to be detected, code of the key is sent to sensor RAM along with SHIFT and CONTROL key status.

⑤ FIFO / Sensor RAM and status Logic:-

- In keyboard mode, this block acts as 8-bit FIFO RAM. Each keycode of the pressed key is entered in the order of entry and read by the CPU till the RAM is empty.
- In scanned sensor matrix mode, the unit acts as sensor RAM. Each row of sensor RAM is loaded with the status of the corresponding row of sensors in the matrix.
- If sensor changes its state, IRQ line goes high to interrupt the CPU.

⑥ Display Address Registers and Display Ram:-

- display address registers contain the address of the word currently being written or read by the CPU to/from Display RAM.
- 16-byte Display RAM → 16-byte data to be displayed on 16 7-segment displays in encoded scan mode.

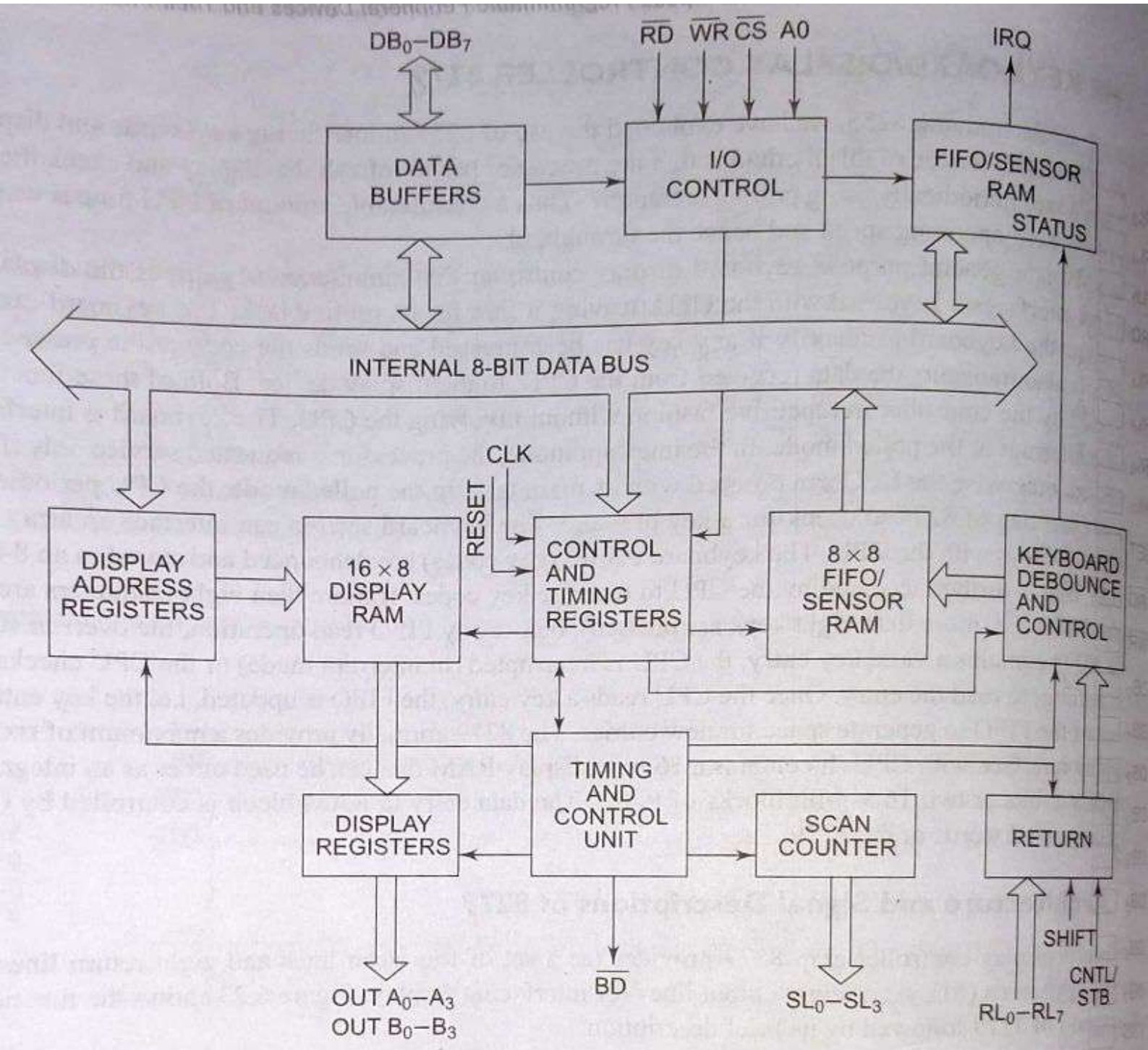


Fig. 6.23 *8279 Internal Architecture*

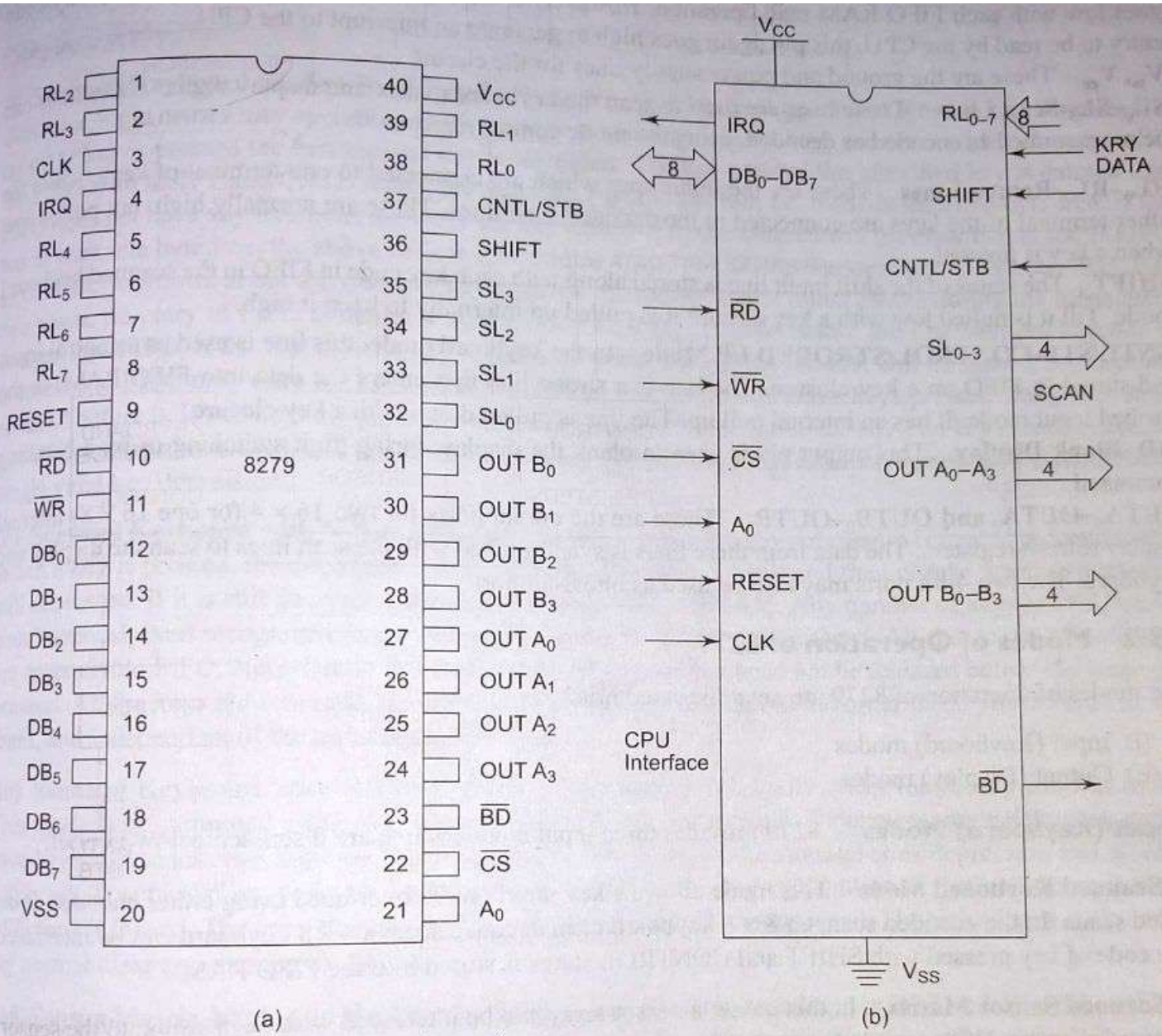


Fig. 6.24 *8279 Pin Configuration and Logic Diagram*

The signal discription of each of the pins of 8279 is presented below in brief:

DB_0–DB_7 These are bidirectional data bus lines. The data and command words to and from the CPU are transferred on these lines.

CLK This is a clock input used to generate internal timings required by 8279.

RESET This pin is used to reset 8279. A high on this line resets 8279. After resetting 8279, its in sixteen 8-bit display, left entry encoded scan, 2-key lock out mode. The clock prescaler is set to 31.

$\overline{CS}$ Chip Select: A low on this line enables 8279 for normal read or write operations. Otherwise, this pin should remain high.

A_0 A high on the A_0 line indicates the transfer of a command or status information. A low on this line indicates the transfer of data. This is used to select one of the internal registers of 8279.

$\overline{RD}$, $\overline{WR}$ (Input/Output) READ/WRITE input pins enable the data buffers to receive or send data over the data bus.

IRQ This interrupt output line goes high when there is data in the FIFO sensor RAM. The interrupt line goes low with each FIFO RAM read operation. However, if the FIFO RAM further contains any key-code entry to be read by the CPU, this pin again goes high to generate an interrupt to the CPU.

V_{ss}, V_{cc} These are the ground and power supply lines for the circuit.

SL_0–SL_3-Scan Lines These lines are used to scan the keyboard matrix and display digits. These lines can be programmed as encoded or decoded, using the mode control register.

RL_0–RL_7-Return Lines These are the input lines which are connected to one terminal of keys, while the other terminal of the keys are connected to the decoded scan lines. These are normally high, but pulled low when a key is pressed.

SHIFT The status of the shift input line is stored along with each key code in FIFO in the scanned keyboard mode. Till it is pulled low with a key closure it is pulled up internally to keep it high.

CNTL/STB-CONTROL/STROBED I/P Mode In the keyboard mode, this line is used as a control input and stored in FIFO on a key closure. The line is a strobe line that enters the data into FIFO RAM, in the strobed input mode. It has an internal pull up. The line is pulled down with a key closure.

BD -Blank Display This output pin is used to blank the display during digit switching or by a blanking command.

$OUTA_0$–$OUTA_3$ and $OUTB_0$–$OUTB_3$ These are the output ports for two 16×4 (or one 16×8) internal display refresh registers. The data from these lines is synchronized with the scan lines to scan the display and keyboard. The two 4-bit ports may also be used as one 8-bit port.

6.3.2 Modes of Operation of 8279

The modes of operation of 8279 are next discussed briefly:

(i) Input (Keyboard) modes
(ii) Output (Display) modes

Input (Keyboard) Modes 8279 provides three input modes which are discussed below in brief:

1. Scanned Keyboard Mode This mode allows a key matrix to be interfaced using either encoded or decoded scans. In the encoded scan, an 8×8 keyboard or in decoded scan, a 4×8 keyboard can be interfaced. The code of key pressed with SHIFT and CONTROL status is stored into the FIFO RAM.

2. Scanned Sensor Matrix In this mode, a sensor array can be interfaced with 8279 using either encoded or decoded scans. With encoded scan 8×8 sensor matrix or with decoded scan 4×8 sensor matrix can be interfaced. The sensor codes are stored in the CPU addressable sensor RAM.

3. Strobed input In this mode, if the control line goes low, the data on return lines, is stored in the FIFO byte by byte.

Output (Display) Modes 8279 provides two output modes for selecting the display options. These are discussed briefly:

1. Display Scan In this mode, 8279 provides 8 or 16 character multiplexed displays those can be organized as dual 4-bit or single 8-bit display units.

2. Display Entry (right entry or left entry mode) 8279 allows options for data entry on the displays. The display data is entered for display either from the right side or from the left side. The concepts regarding these modes will be more clear when the commands are discussed later in this chapter.

All these modes may be selected by programming the 8279 suitably. Further, these modes are discussed in significant details.

Problem 6.5

Interface keyboard and display controller 8279 with 8086 at address 0080H. Write an ALP to set up 8279 in scanned keyboard mode with encoded scan, N-key rollover mode. Use a 16-character display in right entry display format. Then clear the display RAM with zeros. Read the FIFO for key closure. If any key is closed, store it's code to register CL. Then write the byte 55 to all the displays, and return to DOS. The clock input to 8279 is 2 MHz, operate it at 100 kHz.

Solution

The 8279 is interfaced with lower byte of the data bus, i.e. D_0–D_7. Hence the A_0 input of 8279 is connected with address line A_1. The data register of 8279 is to be addressed as 0080H, i.e. $A_0 = 0$. As already discussed, the data is either read from or written to this address ($A_0 = 0$). For addressing the command or status word A_0 input of 8279 should be 1 (the address line A_1 of 8086 should be 1), i.e. the address of the command word should be 0082H. Figure 6.25 shows the interfacing schematic.

The next step is to write all the required command words for this problem.

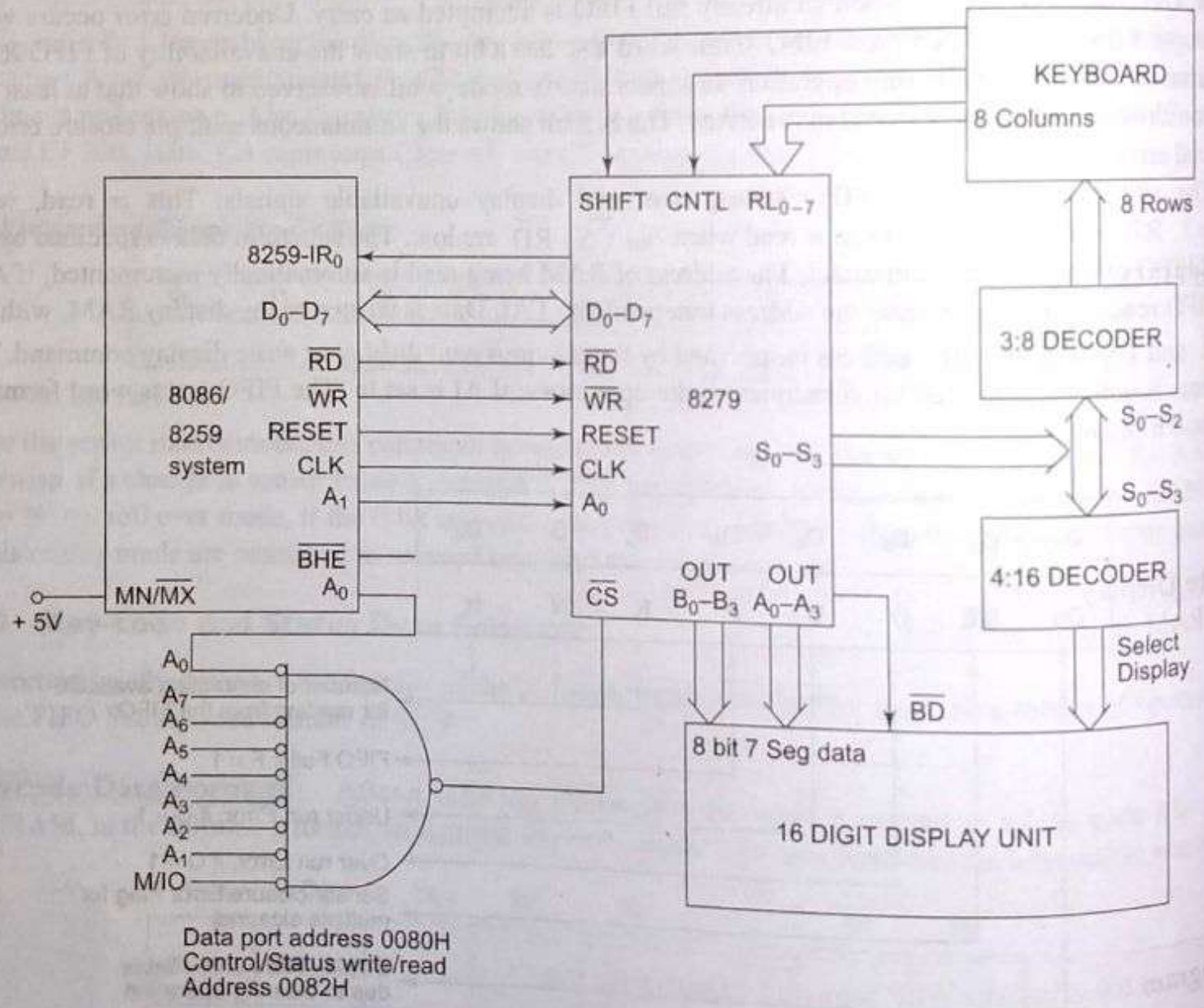
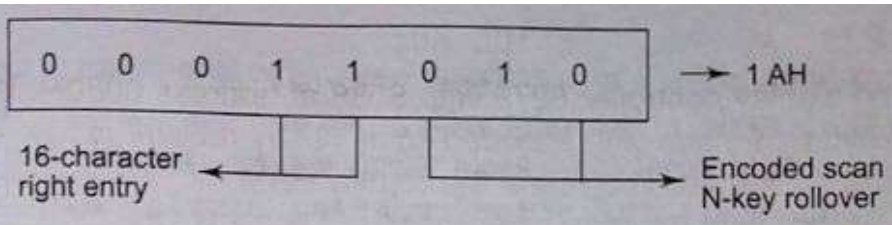
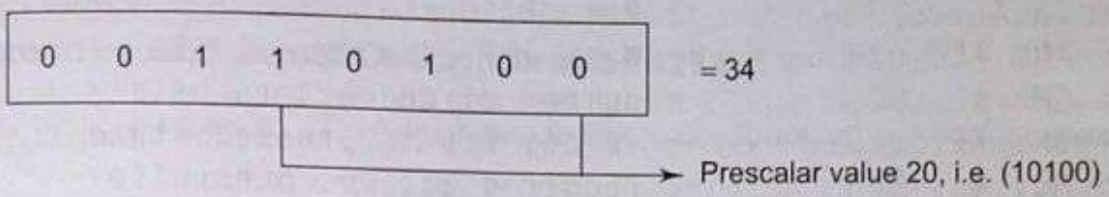


Fig. 6.25 *8279 Interfacing with 8086*

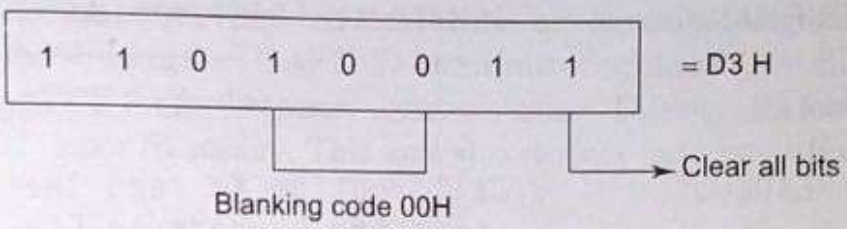
Keyboard/Display Mode Set CW This command byte sets the 8279 in 16-character right entry and encoded scan N-key rollover mode.



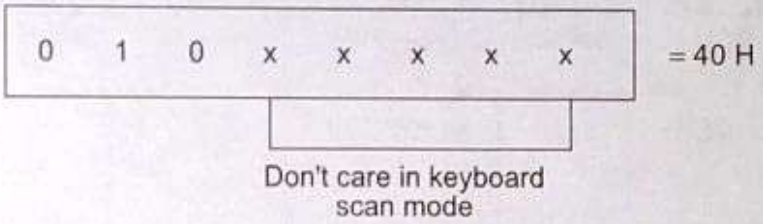
Program Clock Selection The clock input to 8279 is 2 MHz, but the operating frequency is to be 100 kHz, i.e. the clock input is to be divided by 20 (10100). Thus the prescalar value is 10100 and the command byte is set as given.



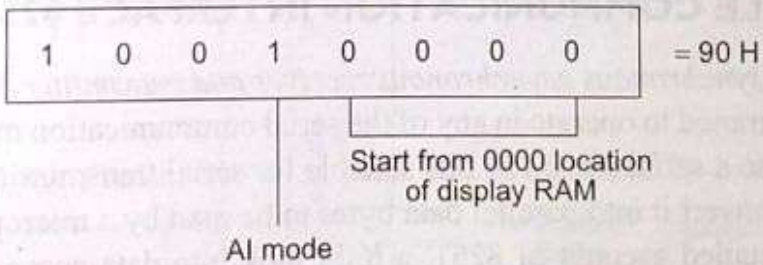
Clear Display RAM This command clears the display RAM with the programmed blanking code.



Read FIFO This command byte enables the programmer to read a key code from the FIFO RAM.



Write Display RAM This command enables the programmer to write the addressed display locations of the RAM as presented below.



Program 6.6 gives the ALP required to initialise the 8279 as required.

```
ASSUME    CS : CODE
CODE      SEGMENT
START:    MOV AL, 1AH    ; SEt 8279 in Encoded scan,
          OUT 82H, AL    ; N key rollover, 16 display, Right entry mode.
          MOV AL, 34H    ; Set clock prescalar to
```

```
        OUT 82H, AL          ; 100 kHz
        MOV AL, 0D3H         ; Clear display ram
        OUT 82H, AL          ; command
WAIT:   MOV AL, 40H          ; Read FIFO command
        OUT 82H, AL          ; for checking display RAM
        IN AL, 82H           ; Wait for clearing of
        AND AL, 80H          ; Display RAM by reading
        CMP AL, 80H          ; FIFO Du bit of the status word i.e.
        JNZ WAIT             ; If DU bit is not set wait, else proceed
        IN AL, 82H           ; Read FIFO status
        AND AL, 07H          ; Mask all bits except the
        CMP AL, 00           ; number of characters bits
        JNZ KEYCODE          ; If any key is pressed, take
WRAM:   MOV AL, 90H          ; required action, otherwise
        OUT 82H, AL          ; proceed to write display
        MOV AL, 55H          ; RAM by using write display
        MOV CH, 10H          ; command.Write the byte
NEXT:   OUT 80H, AL          ; 55H TO ALL DISPLAY RAM
        DEC CH               ; locations
        JNZ NEXT             ;
        JMP STOP             ;
KEYCODE: CALL READCODE       ; Call routine to read the key
        MOV CL, AL           ; store the keycode in CL.
        JMP WRAM             ; code of the pressed key is assumed available
READCODE: MOV AL, 40H
        OUT 82H, AL
        IN AL, 80H
        RET
STOP:   MOV AH, 4CH          ; stop
        INT21H
CODE    ENDS
        END START
```

Program 6.7 *Initialisation of 8279 using an 8086 ALP for Problem 6.6*