Chapter 04

N. Mathivanan

4.2 PARALLEL I/O METHODS

Microprocessor communicates with peripherals in parallel or serial I/O methods. All the bits of 8-bit or 16-bit data word are transferred on 8 or 16 parallel lines at the same instant in a parallel I/O method. The data is sent as stream of bits on a single line in a serial I/O method.

The process of data transfer is very simple if the peripherals are always ready to send or receive data. However, in many cases, the peripherals do not generate or accept data at the speed of microprocessor. In such cases, the microprocessor checks the readiness of the peripheral before effecting the data transfer. Various parallel I/O methods are described in the following.

4.2.1 Simple I/O

In a simple I/O method, the microprocessor transfers data byte assuming that the peripheral is always ready to send or receive the byte. Figure 4.1 illustrates the simple I/O method. Switches and LEDs connected to input and output ports are always ready to send or receive data bytes to or from the microprocessor. An I/O read operation simply enables the input port and transfers the status information of the switches from the port to the data bus. Similarly, an I/O write operation enables the output port and transfers the data byte to the port. The port in turn outputs the byte to LEDs.

ready to receive a digital word for conversion from the microprocessor. The microprocessor simply writes the digital word into the output port for D/A conversion.

Similarly, analog to digital conversion by flash converter type ADCs is also very fast and almost instantaneous. Hence, the ADC is always ready to send the digitized data to the microprocessor. The microprocessor simply reads the digitized data from the input port at any time it requires.

4.2.2 Strobe I/O

Simple I/O technique is not suitable to transfer data between microprocessor and slow devices. Slow devices do not send or accept data at the speed of the microprocessor. In strobe I/O method, the microprocessor transfers the data only when the peripheral is ready. Slow devices generate a control signal called *strobe* along with valid data. Strobe signal indicates that valid data is present on the data lines. The readiness of the peripheral is communicated to the microprocessor in two ways, by polling or interrupt. Figure 4.3 illustrates the two schemes.

Polling

In the polling method, the strobe line is connected to an input port line. The microprocessor

keeps reading the input port repeatedly (polling) and checks the readiness of the peripheral. When the microprocessor determines that valid data is present, it effects data transfer. Figure 4.3(a) describes the technique.

Interrupt

In the interrupt method, the strobe line is connected to an interrupt input of the microprocessor. The microprocessor is interrupted when the peripheral is ready. In response, the microprocessor executes an interrupt service procedure meant for data transfer and effects the data transfer. Figure 4.3(b) explains the interrupt driven data transfer technique. The technique generally uses an interrupt controller to process interrupt signals from peripherals. The advantage of this technique is it frees the microprocessor to execute any other program when the device is not ready, rather than keeping in a polling loop.

Circuits in Figure 3.32 and Figure 3.41 illustrated polling and interrupt. The circuit in Figure 3.32 used the polling method for data transfer from the ADC. The microprocessor monitors the STS level by repeatedly reading the input port. When the microprocessor determines that the conversion is complete, it reads the data from the ADC.

The circuit in Figure 3.41 uses the interrupt method for data transfer from the ADC. When the ADC completes the conversion, it signals through 'STS' to 8259A, which in turn interrupts the microprocessor. In response, the microprocessor reads the data from the ADC.

Circuits in Figure 3.32 and Figure 3.41 illustrated polling and interrupt. The circuit in Figure 3.32 used the polling method for data transfer from the ADC. The microprocessor monitors the STS level by repeatedly reading the input port. When the microprocessor determines that the conversion is complete, it reads the data from the ADC.

The circuit in Figure 3.41 uses the interrupt method for data transfer from the ADC. When the ADC completes the conversion, it signals through 'STS' to 8259A, which in turn interrupts the microprocessor. In response, the microprocessor reads the data from the ADC.

4.2.3 Handshake I/O

In handshake I/O method, the microprocessor and the peripheral exchange handshake signals to indicate the readiness of the peripherals and synchronize the timing of data transfer. An interface device assists the exchange of data and handshake signals. Figure 4.4 illustrates the handshake I/O techniques for input and output operations.

Input operation

Strobe (STB) and input buffer full (IBF) are the two handshake signals used for input operation. The peripheral sends the STB signal along with a byte to the interfacing device. The interfacing

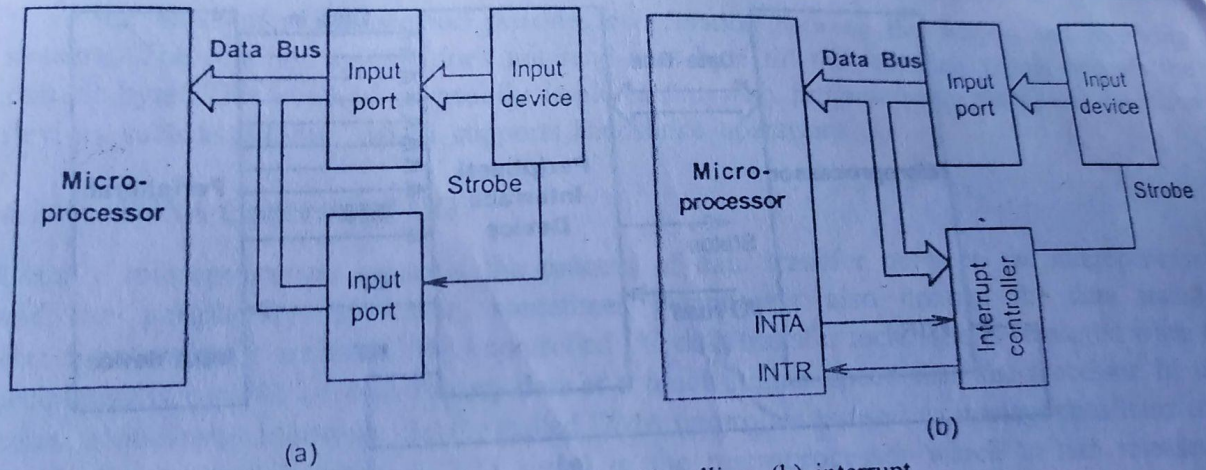
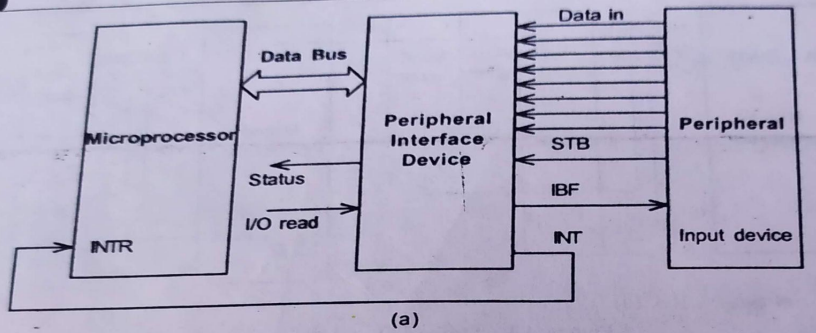
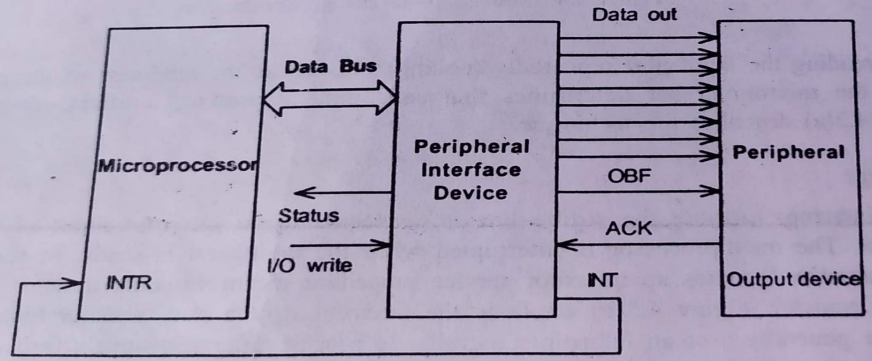


Figure 4.3 Strobe I/O—(a) polling, (b) interrupt.

keeps reading the input port repeatedly (polling) and checks the readiness of the peripheral. When the microprocessor determines that valid data is present, it effects data transfer. Figure 4.3(a) describes the technique.



(a)



(b)

Figure 4.4 Handshake I/O—(a) input operation, (b) output operation.

Example 4.4

Interface a DAC to an 8086 based system through 8255A for controlling the speed of a DC motor.

Solution

Figure 4.9 shows DAC-08 connected to 8255A for controlling the speed of a DC motor. DAC-08 is an 8-bit, low-cost digital to analog converter. It neither has internal latch nor op amp. It has only *R-2R* ladder network. It has two reference inputs (pins 14 and 15) for positive and negative voltages. Connection to positive reference input is shown in the figure. User can externally adjust the ladder reference current I_{ref} from 4 μA to 4 mA by choosing values for V_{ref} and R_{ref} ($I_{ref} = V_{ref}/R_{ref}$). The resolution of this 8-bit DAC is $I_{ref}/2^8$. The device has two analog current output terminals, I_{out} and $I_{out}(-)$. Current through $I_{out}(-)$ is the complement of current

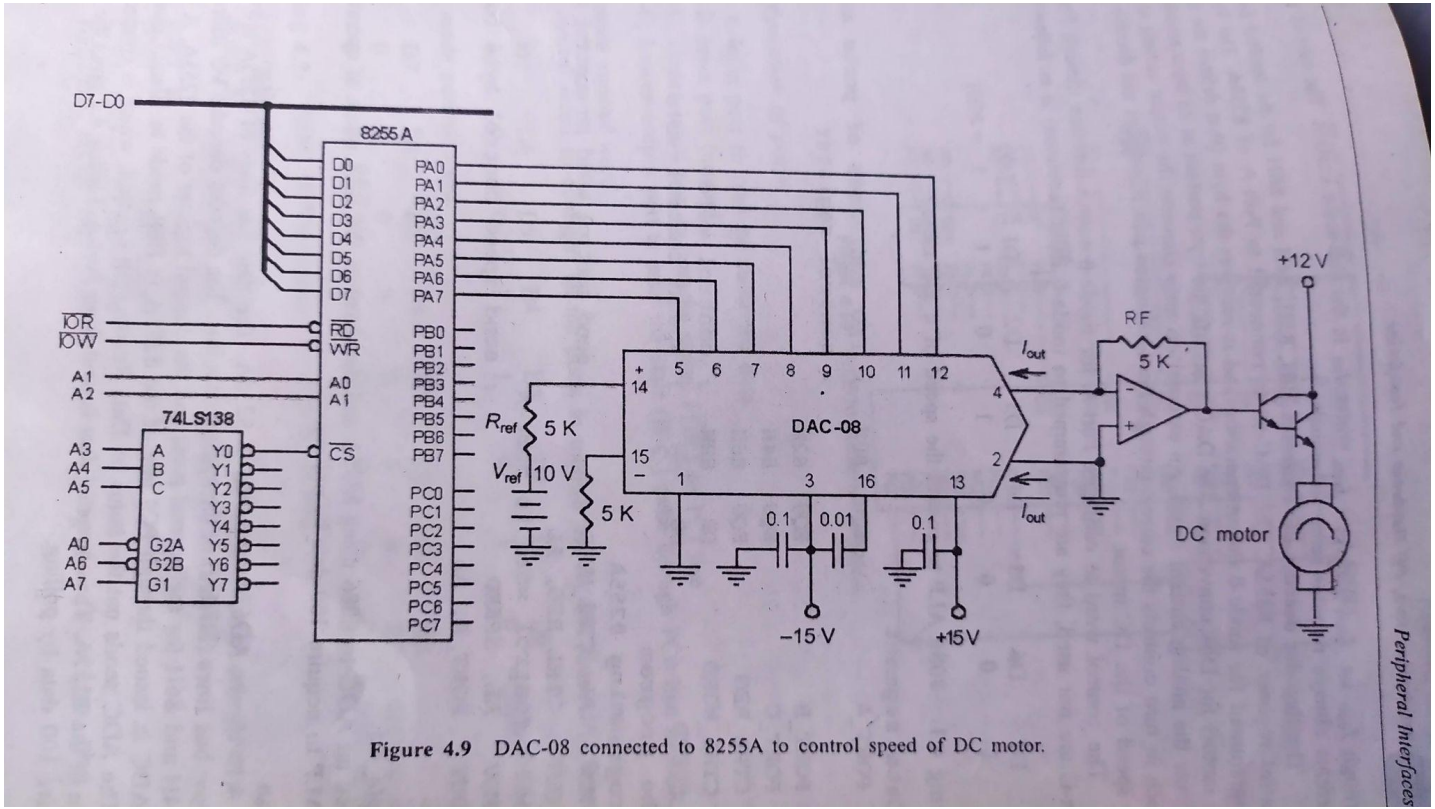


Figure 4.9 DAC-08 connected to 8255A to control speed of DC motor.

through I_{out}, i.e. $I_{out}(-) = I_{FS} - I_{out}$, where I_{FS} is the full-scale current. The sum of the two currents always remains same and equals I_{FS}.

The decoder decodes I/O addresses 80H, 82H, 84H and 86H for the internal ports and control register of 8255A. The DAC-08 is connected to Port-A of 8255A. The Port-A is programmed for mode-0 output operation and it receives the byte (that defines the speed of the motor) for D/A conversion. The DAC converts the byte present at its input terminals and outputs the analog current signal. An external op amp converts the output current to voltage which in turn controls the current through the Darlington pair transistors and thereby controls the speed of the DC motor.

The control word to configure Port-A for mode-0 output function (though Port-B and Port-C are not used, they are programmed to mode-0 input functions) is as follows:

D7	D6	D5	D4	D3	D2	D1	D0	
1	0	0	0	1	0	1	1	= 8BH

Listing 4.1: 8086 ALP to control the speed of a DC motor.

```
; Data segment
   PORT_A          EQU   80H   ; I/O addresses of ports and
                               ; control register
```

```
CTRL_WORD          DB    --     ; speed control word
SPEED              DB    --     ; speed control word
; The program
; Programming 8255A
    MOV    AL, CTRL_WORD        ; send control word to control register
    OUT    CTRL_REG, AL
; Speed control word to port-A
    MOV    AL, SPEED            ; send speed control byte to Port-A
    OUT    PORT_A, AL
    -
```

Example 4.5

Interface an ADC to 8086 using 8255A and demonstrate the BSR mode of operation. Write 8086 ALP to acquire 100 data by polling.

Solution

Figure 4.10 shows AD570 connected to 8255A. The data bus lines of 8255A are connected to the low bus lines (D7-D0) of the system data bus. The decoder decodes I/O addresses 80H, 82H, 84H and 86H for the internal ports and the control register of the 8255A. A start pulse to the ADC is issued through PC0 line of the 8255A in BSR mode to Blank/Convert ($B/\overline{C}$) input. The ADC sends out the status on Data Ready ($\overline{DR}$) output, which is connected to the PC7 line of the 8255A. The converted data is read from Port-A. Listing 4.2 gives the 8086 ALP to acquire 100 data by polling.

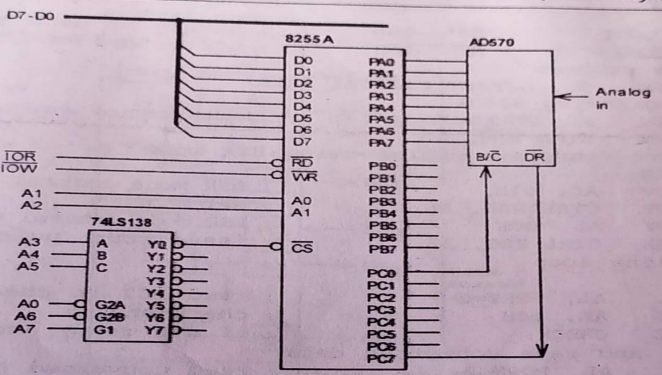


Figure 4.10 AD570 connected to 8255A.

Configuration of ports:

Port-A–input port to read the converted data
Port-B–input port (unused)
Port-C, Upper–input port to read the status ($\overline{DR}$) on PC7 line
Port-C, Lower–output port to issue the start ($B/\overline{C}$) pulse through PC0 line in BSR mode.

I/O mode control word:

The control word to configure all the ports in mode-0 and for the above I/O functions:

D7	D6	D5	D4	D3	D2	D1	D0	
1	0	0	1	1	0	1	0	= 5AH

BSR mode control word:

D7	D6	D5	D4	D3	D2	D1	D0	
0	0	0	0	0	0	0	1	= 01H, to set PC0
0	0	0	0	0	0	0	0	= 00H, to reset PC0

Example 4.6

Interface an ASCII keyboard to 8086 through 8255A for interrupt I/O and write ALP to get 100 characters.

Solution

Figure 4.12 shows an ASCII keyboard connected to 8255A. The decoding logic for interfacing 8255A, which is not shown in the figure, decodes even I/O addresses 40H, 42H, 44H, and 46H for the 8255A.

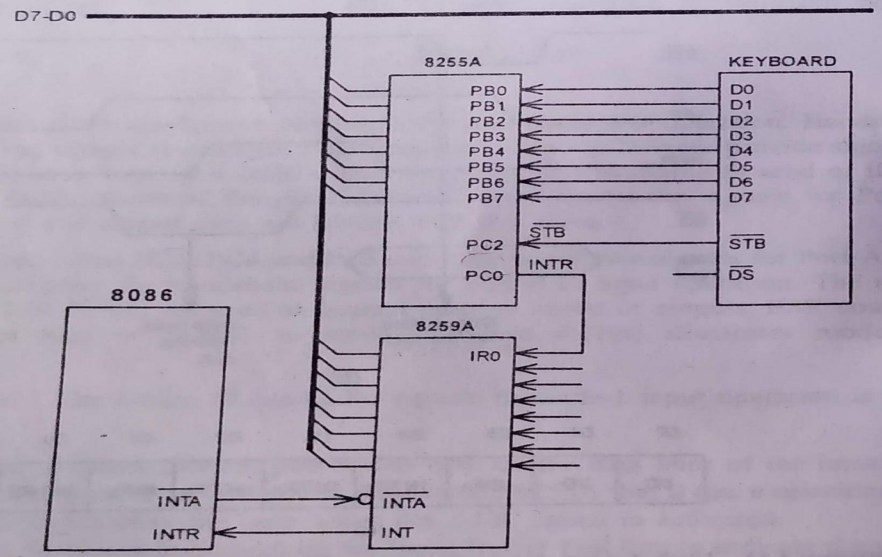


Figure 4.12 ASCII keyboard connected to 8255A—Handshake I/O, input.

- The Port-B is configured to mode-1 input function for interrupt driven I/O.
- When a key is pressed, the keyboard places the character code for the key on the data lines and activates the $\overline{STB}$ signal that latches the code in Port-B.
- The 8259A processes the interrupt request $\overline{INTR_B}$ and provides a type number (e.g. 0AH) to the microprocessor for the interrupt.
- The microprocessor executes the interrupt service procedure when the interrupt occurs and reads the character from Port-B.

Listing 4.3: 8086 ALP to get characters from keyboard on interrupt

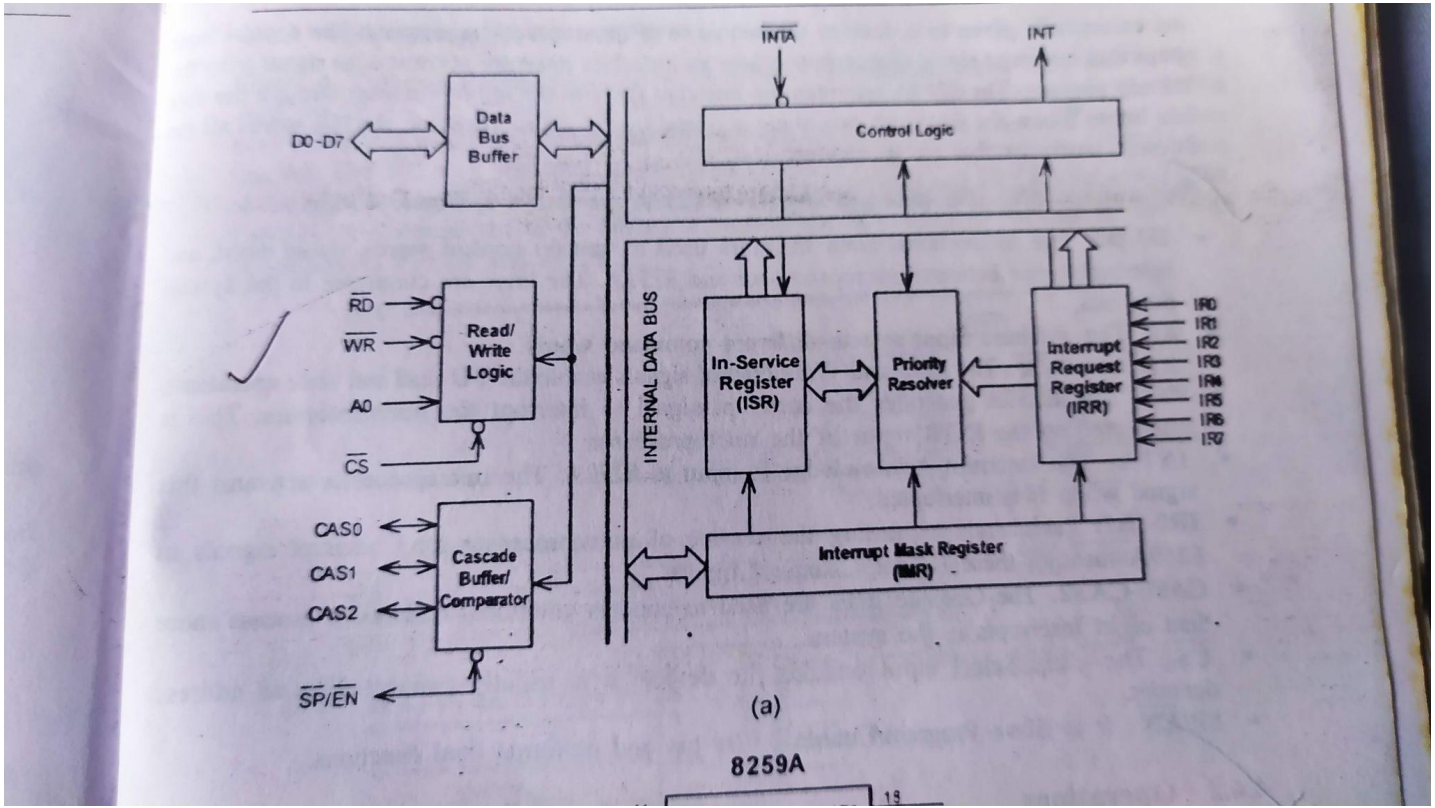
```
; Data segment
   CHARS      DB   100 DUP(0)    ; array for storing the characters
   CHAR_PTR   DW   OFFSET CHARS ; pointer to the character
   COUNT      DW   100           ; number of characters to be read
   PORT_A     DB   40H           ; I/O addresses of ports and
                                 ; control register

   PORT_B     DB   42H
   PORT_C     DB   44H
   CTRL_REG   DB   46H
; Programming 8255A              ; initialize 8255A
```

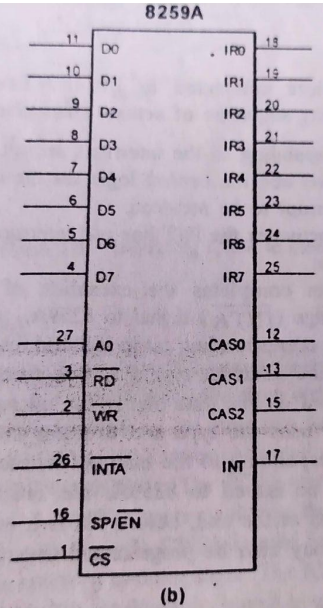
4.4 PRIORITY INTERRUPT CONTROLLER, 8259A

The programmable interrupt controller, 8259A from Intel, is interfaced to microprocessors to manage multiple hardware interrupts. It accepts eight interrupts, resolves priorities among the interrupts in several ways and vectors the requests to anywhere in the memory map. It enables or disables the interrupt requests individually. It can be expanded to process up to 64 interrupt requests by cascading additional 8259A devices. The device is used in personal computers.

4.4.1 Internal Block Diagram

Figure 4.16(a) shows the internal blocks of 8259A. It has interrupt request register (IRR), in-service register (ISR), interrupt mask register (IMR), a control logic, Read/Write logic, data bus buffer, priority resolver and cascade buffer. Peripherals requesting the service of the microprocessor send interrupt signals through interrupt request inputs, IR0-IR7, of 8259A. The IRR stores all the interrupt requests. Any interrupt request can be disabled by a command to 8259A and the IMR holds the bits that mask the interrupt requests. The priority resolver as per


INTA
INT
D0-D7
Data Bus Buffer
Control Logic
RD
WR
A0
CS
Read/ Write Logic
INTERNAL DATA BUS
In-Service Register (ISR)
Priority Resolver
Interrupt Request Register (IRR)
IR0
IR1
IR2
IR3
IR4
IR5
IR6
IR7
CAS0
CAS1
CAS2
Cascade Buffer/ Comparator
Interrupt Mask Register (IMR)
SP/EN
(a)
8259A



(b)

Figure 4.16 The priority interrupt controller, 8259A—(a) block diagram, (b) pins and signals.

the commands given to it decides the sequence of interrupts to be serviced. The control logic generates interrupt signal to microprocessor and receives interrupt acknowledge signal from the microprocessor. The 8259A provides the interrupt types to the microprocessor through the data bus lines. When the interrupts are being serviced by the microprocessor, the ISR stores all the interrupt requests that are in service.

Pins and signals. The pins and signals of 8259A are shown in Figure 4.16(b).

- **D7-D0.** The bidirectional Data lines are used to transfer control words, status word, and interrupt types between microprocessor and 8259A. The lines are connected to the system data bus.
- **A0.** The Address input selects different command words.
- **$\overline{RD}$ and $\overline{WR}$.** The Read and Write control signals enable the I/O read and write operations.
- **INT.** The 8259A generates the Interrupt signal to interrupt the microprocessor. This is connected to the INTR input of the microprocessor.
- **$\overline{INTA}$.** The Interrupt Acknowledge is input to 8259A. The microprocessor activates this signal when it is interrupted.
- **IR0-IR7.** Peripherals requesting the service of microprocessor send interrupt signals to 8259A through these Interrupt Request inputs.
- **CAS0-CAS2.** The Cascade lines are used to connect additional 8259As to process more than eight interrupts in the system.
- **$\overline{CS}$.** The Chip Select input enables the device. It is usually connected to an address decoder.
- **SP/$\overline{EN}$.** It is Slave Program/Enable buffer pin and performs dual functions.

4.4.2 Operations

4.4.2 Operations

When one or more devices connected to IR0-IR7 lines request the service of the microprocessor, the following sequence of actions takes place:

(a) Bits in IRR corresponding to the interrupts are set.

(b) The priority resolver and the control logic use the information in ISR and IMR and determine the interrupt to be serviced.

(c) The control logic activates the INT line and interrupts (if the INTR input is enabled) the microprocessor.

(d) The microprocessor completes the execution of current instruction and sends interrupt acknowledge ($\overline{INTA}$) signal to 8259A.

(e) The 8259A sets the corresponding bit in ISR and resets the corresponding bit in IRR. The 8259A sends the interrupt type that was specified for the interrupt when the 8259A was initialized on the data bus to the microprocessor.

(f) The microprocessor uses the type and executes the interrupt service procedure.

(g) The bit in ISR corresponding to the interrupt is reset. It requires an end of interrupt command (EOI) to be issued to 8259A. The interrupt service procedure includes necessary instructions at the end, before IRET. It enables the next interrupt to take effect. The 8259A may also be programmed to automatic end of interrupt (AEOI) mode.

4.4.3 Interfacing 8259A to 8086

Figure 4.17 shows interfacing of 8259A to 8086 microprocessor. Since only one 8259A is used, the SP/$\overline{EN}$ pin is tied to logic high and CAS0, CAS1 and CAS2 are left open. The 8259A has two internal addresses that are selected through the pin A0. It is connected to the address line A1. The $\overline{RD}$ and $\overline{WR}$ pins are connected to I/O read ($\overline{IOR}$) and I/O write ($\overline{IOW}$) lines of the system. The INT and $\overline{INTA}$ pins are connected to INTR and $\overline{INTA}$ pins of the 8086. The 8259A is connected to the lower I/O bank of the I/O map. It is selected for the 16-bit I/O addresses FFF0H and FFF2H.

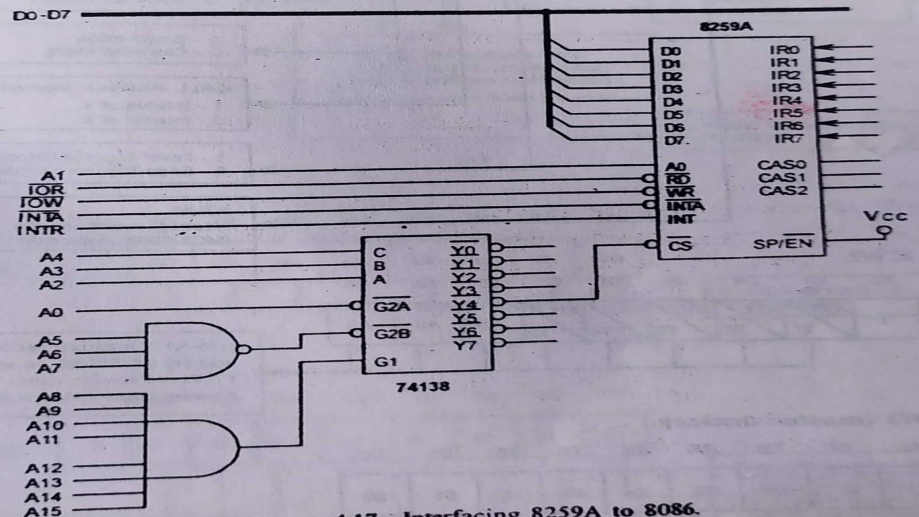


Figure 4.17 Interfacing 8259A to 8086.

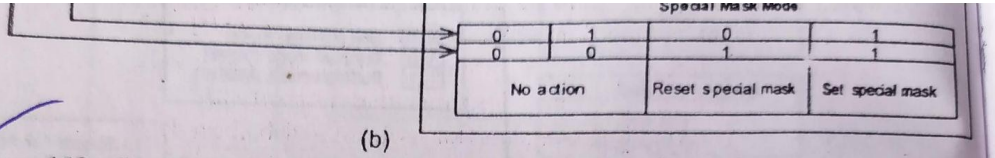


Figure 4.18 Command words of 8259A—(a) ICW1–ICW4, (b) OCW1–OCW3.

Initialization command words

1. The ICW1 is used to tell whether, (i) the system uses single or multiple 8259As, (ii) it is an 8085 or 8086/above microprocessor based system, and (iii) the interrupt request pins are level or edge sensitive.
2. The ICW2 specifies the interrupt type number to be sent to 8086 to an interrupt request on the IR0 input. The 8259A uses the type numbers in sequence for the other interrupt requests.
3. The ICW3 is used only if the ICW1 specifies that the system uses multiple 8259As. It indicates the interrupt request line used by a slave.
4. The ICW4 is not used if it is not an 8086 or an advanced microprocessor based system. It tells the option for automatic end of interrupt, (AEOI) mode. If AEOI is selected, the 8259A automatically resets the ISR bit corresponding to the interrupt being responded. This command also selects the special fully nested mode of operation for the 8259A.

Operational command words

The ICW4 is not used if it is not an 8086 or an advanced microprocessor based system. It tells the option for automatic end of interrupt, (AEOI) mode. If AEOI is selected, the 8259A automatically resets the ISR bit corresponding to the interrupt being responded. This command also selects the special fully nested mode of operation for the 8259A.

Operational command words

1. The OCW1 is used to set and read the interrupt mask register. The OCW1 is sent to 8259A to enable or disable an interrupt request.
2. The OCW2 is used to reset a bit in the in-service register and used only if AEOI is not selected by ICW4. This is done at the end of an interrupt service procedure. This EOI command is issued in several formats. The non-specific EOI command resets the interrupt that is active. The specific EOI command resets specific interrupt request. The rotate on non-specific EOI command does the function of non-specific EOI and further it rotates interrupt priorities. When priority is rotated, the interrupt just serviced gets the lowest priority. The rotate on specific EOI command does the function of specific EOI and further it rotates the interrupt priority.
3. The OCW3 command selects the register to be read, the operation of the special mask register, and the poll command. To read the status of IRR and ISR, this command is sent.

Status registers

The status of the three registers IRR, ISR and IMR can be read from the 8259A to find out pending interrupts, in-service interrupt and masked off interrupts. Both the IRR and ISR are read by sending OCW3 and IMR is read by sending OCW1. The status of IMR is read with logic high on A0 pin and the status of IRR and ISR are read with a logic low on A0 pin.

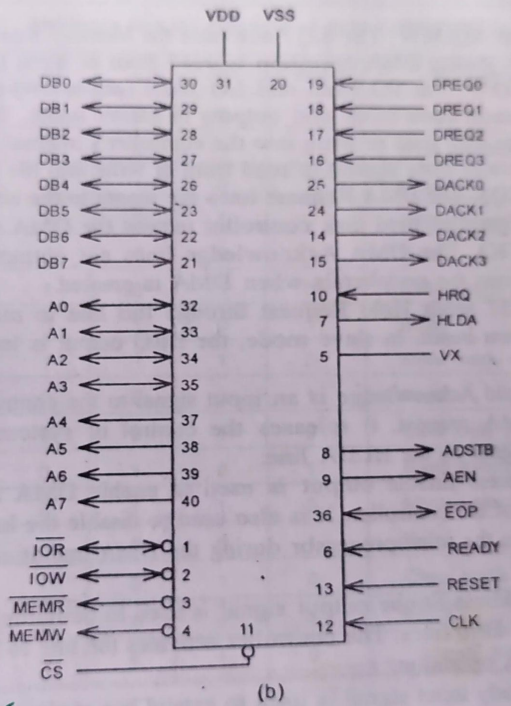
Example 4.8

Initialize the 8259A in Example 3.10 for the interrupt driven data acquisition application.

Solution

The application is explained in Example 3.10. Figure 4.19 shows the circuit. The PAL decoder is programmed to generate device select outputs for the following input conditions.

$$\overline{Y0} = \overline{A7} \times A6 \times \overline{A5} \times \overline{A4} \times \overline{A3} \times \overline{A2} \times \overline{A1} \times \overline{A0} \times \overline{BHE}$$



(b)

Figure 4.20 The DMA controller, 8237—(a) internal blocks, (b) pins and signals.

Figure 4.20 The DMA controller, 8237—(a) internal blocks, (b) pins and signals.

Pins and signals

The pins and signals of 8237 are shown in Figure 4.20(b). The DMA controller in a system operates in either master or slave mode. When the microprocessor is in control of system buses, the DMA controller serves in slave mode as a simple I/O interface device. As an I/O interface device, it receives addresses and commands from microprocessor. When the DMA controller is in control of system buses, it serves as a controller (master mode) and provides addresses, control signals and data to memory and peripherals. Many of its signals that are inputs in slave mode become outputs in master mode. The DMA signals and their functions are outlined in the following:

- **DB7-DB0**. The Data lines are bidirectional. The controller multiplexes the data and high-order address byte on these lines during DMA operation.
- **A3-A0**. The Address lines are inputs in slave mode and used to access the internal registers. In master mode, the lines are outputs and provide a part of memory address.
- **A7-A4**. They are Address output lines. In DMA operation the A7-A0 lines carry the low-order byte of memory address.
- $\overline{\text{CS}}$. The Chip Select input enables 8237 for programming. It is connected to an output of a decoder.

- **$\overline{\text{MEMR}}$, and $\overline{\text{MEMW}}$**. The 8237 activates the Memory Read and Memory Write control output signals during DMA operation to read from or write into memory.
- **$\overline{\text{IOR}}$, and $\overline{\text{IOW}}$**. The I/O Read and I/O Write (active-low) are bidirectional signals and serve as inputs in slave mode and outputs in master mode. The microprocessor activates these signals to read from or write into the controller's internal registers in slave mode. The controller activates these signals to read from or write into the I/O devices in master mode.
- **DREQ0-DREQ3**. The DMA Request lines are inputs to the controller. Peripherals such as floppy disk controller, hard disk controller request the DMA service through these lines.
- **DACK0-DACK3**. The DMA Acknowledge lines are outputs from the controller. The controller informs the peripherals when DMA is granted.
- **HRQ**. The 8237 sends Hold Request through this line to microprocessor to release the control of system buses. In slave mode, the HRQ output is low and in master mode it is high.
- **HLDA**. The Hold Acknowledge is an input signal to the controller. As the microprocessor detects the DMA request, it releases the control of system buses and send the hold acknowledge signal on the HLDA line.
- **AEN**. The Address Enable output is used to enable DMA address latch connected to DB0-DB7 pins of the controller. It is also used to disable the latches to virtually delink the address bus from the microprocessor during the DMA operation. In slave mode, it outputs low.
- **ADSTB**. The Address Strobe output signal is used to demultiplex high-order address byte and data on DB7-DB0 lines. The controller activates the line to latch the high-order address byte during DMA operation.
- **READY**. The Ready input signal is used to extend bus cycles by inserting wait states when servicing slow devices.
- **RESET**. The Reset input when activated, clears the control, status, and temporary registers and request flags, and sets mask flags.
- **EOP**. The End of Process is a bidirectional line. The controller indicates the end of DMA process through this line. An interface device may input a low to the controller on this line to terminate a DMA process.