

UNIT-III

PERIPHERALS INTERFACING

Interfacing of 8085 with: Keyboard & display unit [8279 IC] – Parallel peripheral interface [8255] – Interrupt controller interface [8259 PIC] – USART interface [8251] - DMA controller

Programmable Peripheral Interface [8255]:

- ❖ The 8255A is a widely used, general purpose programmable, parallel I/O device.
- ❖ It can be programmed to transfer data under various conditions, from simple I/O.
- ❖ It is flexible, versatile and economical (when multiple I/O ports are required), but somewhat complex.
- ❖ It is an important general purpose I/O device that can be used with almost any microprocessor.
- ❖ The 8255A is a 40 pin IC.
- ❖ It consists of 24 I/O pins that can be grouped primarily in two 8-bit parallel ports: A and B, with the remaining 8-bits as port C.
- ❖ The 8-bits of port C can be used as individual bits or be grouped in two 4-bit ports: C_{UPPER} (C_U) and C_{LOWER} (C_L), as in figure 3.1(a).
- ❖ The functions of these ports are defined by writing a control word in the control register.

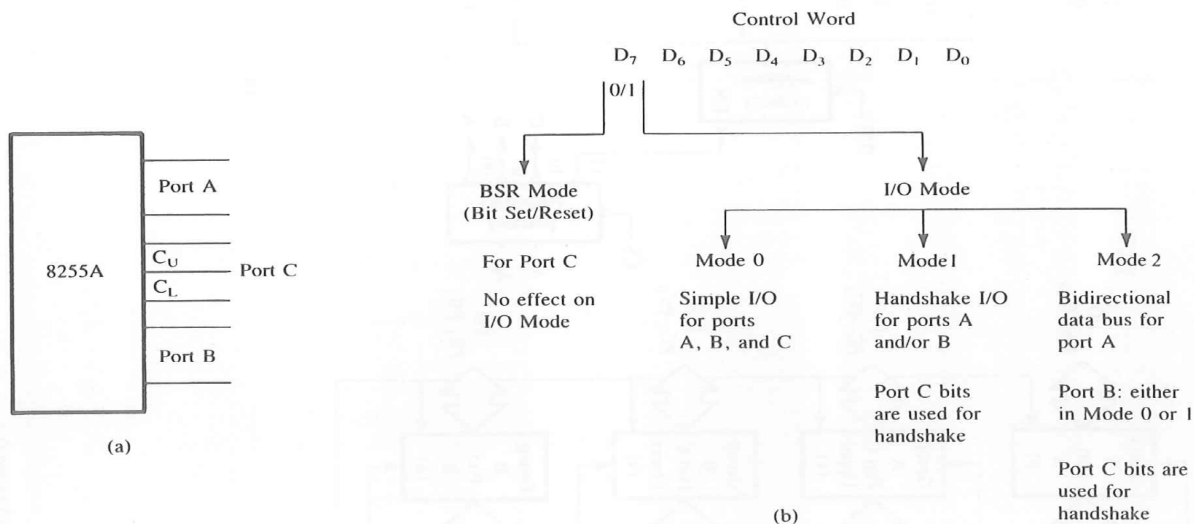


Figure 3.1 (a) 8255 I/O Ports and 3.1(b) 8255 Modes

- ❖ The figure 3.1(b) shows all the functions of the 8255A, classified according to two modes:
 - Bit Set/Reset (BSR) mode - is used to set or reset the bits in port C.
 - I/O mode – is further divided into three modes
 - Mode 0 – All ports function as simple I/O ports

- Mode 1 – Handshake mode, whereby Port A and/or Port B use bits from Port C as handshake signals. In this mode two types of I/O data transfer can be implemented:
 - Status Check
 - Interrupt
- Mode 2 – Port A can be setup for bidirectional data transfer using handshake signals from Port C, and Port B can setup either in Mode 0 or Mode 1

Block Diagram:

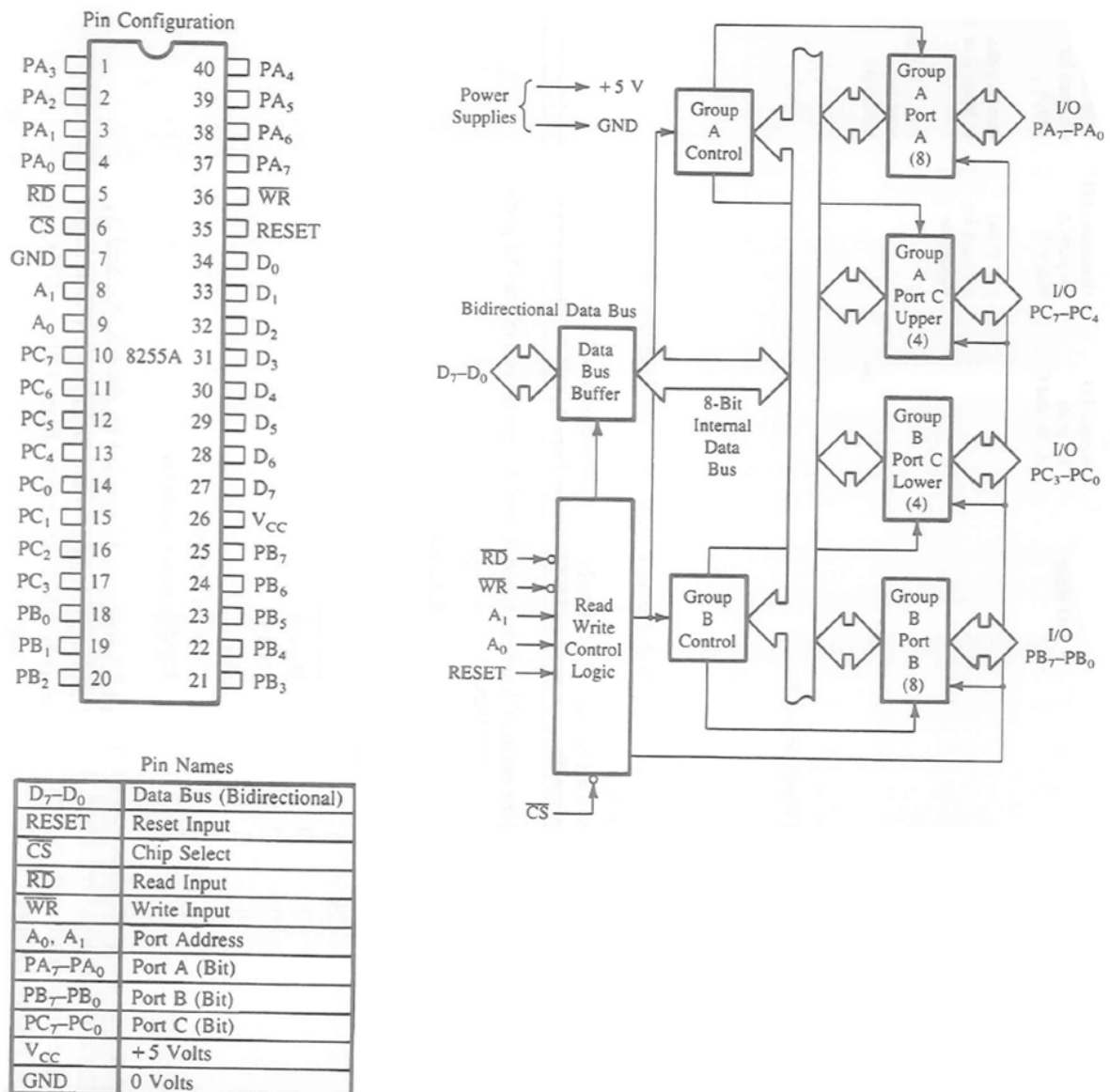


Figure 3.2 8255A Block Diagram

- ❖ The block diagram is shown in figure 3.2 shows 8-bit ports (A and B), two 4-bit ports C_{UPPER} (C_U) and C_{LOWER} (C_L), the data bus buffer, and control logic.

- ❖ This block diagram includes all the elements of a programmable device; Port C performs functions similar to that of the status register in addition to providing handshake signals.

❖ **Control Logic:**

- The control section has 6 lines. Their functions and connections are as follows:
 - **Read (\overline{RD}):** When the signal is low, the MPU reads data from a selected I/O port of the 8255A.
 - **Write (\overline{WR}):** When the signal goes low, the MPU writes into a selected I/O port of the 8255A or the control register.
 - **RESET (Reset):** This is an active high signal: it clears the control register and sets all ports in the input mode.
 - **\overline{CS} , A_0 and A_1 :** These signals are select signals. \overline{CS} is connected to a decoded address and A_0 and A_1 are generally connected to MPU address lines A_0 and A_1 respectively.
- \overline{CS} signal is the master chip select, and A_0 and A_1 specify one of the I/O ports or the control register as given below:

\overline{CS}	A_1	A_0	Selected
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control Register
1	X	X	8255A is not selected.

- As an example, the port addresses in figure 3.3(a) are determined by the \overline{CS} , A_0 and A_1 lines.
- The \overline{CS} line goes low when $A_7=1$ and A_6 through A_2 are at logic 0.
- When these signals are combined with A_0 and A_1 the port addresses range from 80H to 83H, as shown in figure 3.3(b).

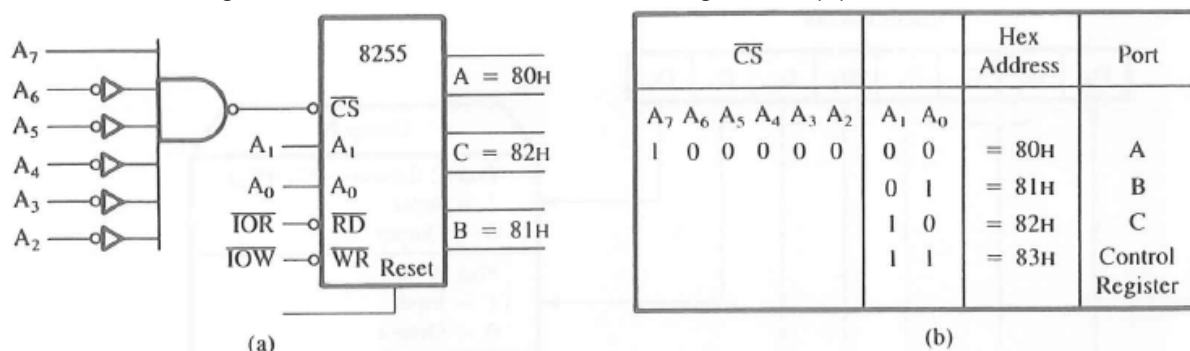


Figure 3.3(a) 8255A Chip Select Logic and (b) I/O Port Addresses

❖ **Control Word:**

- The figure 3.1(b) shows a register called the control register.

- The contents of this register, called the control word, specify an I/O function for each port.
- The figure 3.4 shows control word format for I/O mode.

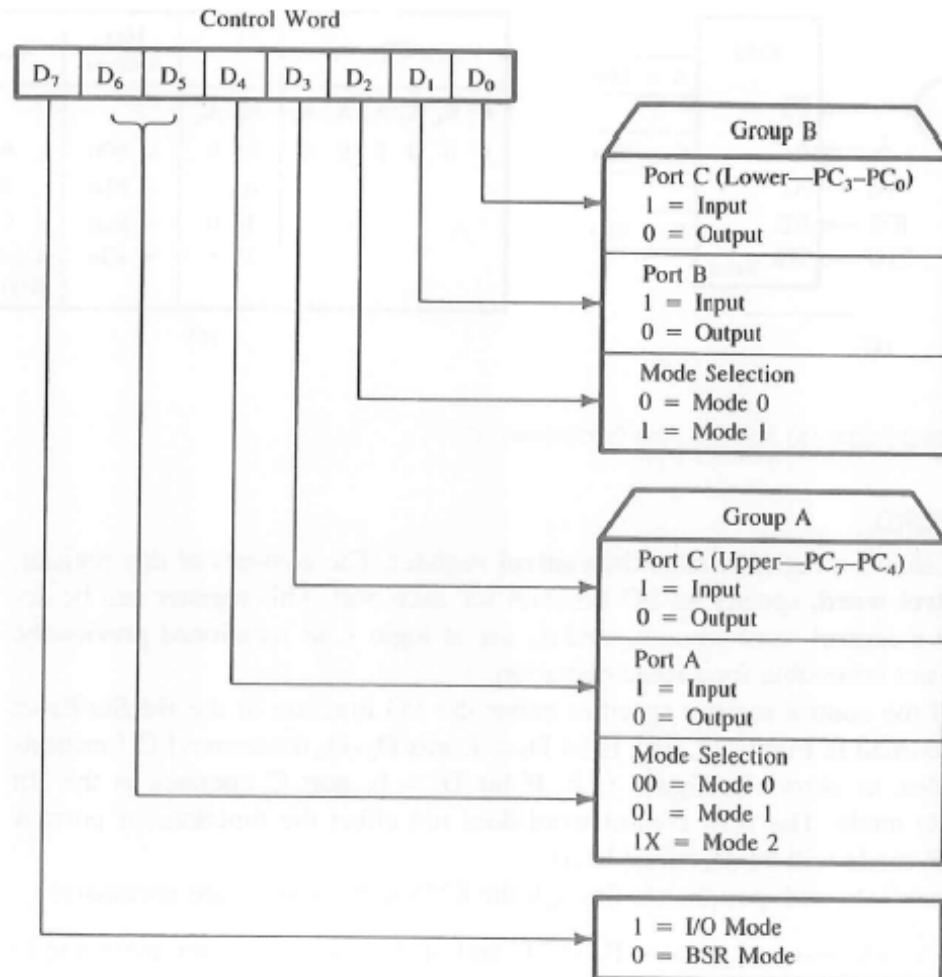


Figure 3.4 Control word format for I/O mode

BSR (Bit Set/Reset) Mode:

- ❖ The BSR mode is connected only with the 8 bits of PORT C, which can be set or reset by writing an appropriate control word in the control register.
- ❖ A control word with bit D₇=0 is recognized as a BSR control word, and it does not alter any previously transmitted control word with bit D₇=1; thus the I/O operations of PORT A and PORT B are not affected by a BSR control word.
- ❖ In the BSR mode, individual bits of PORT C can be used for applications such as an ON/OFF switch.
- ❖ **BSR Control Word:**
 - This control word, when written in the control register, sets or resets one bit at a time, as specified in figure 3.5.

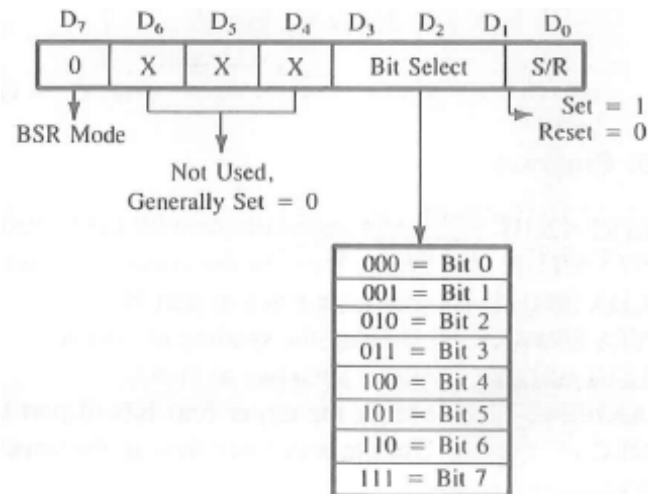


Figure 3.5 8255A Control Word Format in the BSR Mode

Example 1: Write a BSR control word subroutine to set bits PC7 and PC3 and reset them after 10ms.

Solution:

BSR CONTROL WORDS

	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
To set bit PC ₇	= 0	0	0	0	1	1	1	1	= 0FH
To reset bit PC ₇	= 0	0	0	0	1	1	1	0	= 0EH
To set bit PC ₃	= 0	0	0	0	0	1	1	1	= 07H
To reset bit PC ₃	= 0	0	0	0	0	1	1	0	= 06H

PORT ADDRESS

Control register address = 83H; refer to Figure 15.3(b).

SUBROUTINE

```

BSR:  MVI A,0FH      ;Load byte in accumulator to set PC7
      OUT 83H        ;Set PC7 = 1
      MVI A,07H      ;Load byte in accumulator to set PC3
      OUT 83H        ;Set PC3 = 1
      CALL DELAY      ;This is a 10-ms delay
      MVI A,06H      ;Load accumulator with the byte to reset PC3
      OUT 83H        ;Reset PC7
      MVI A,0EH      ;Load accumulator with the byte to reset PC7
      OUT 83H        ;Rest PC7
      RET

```

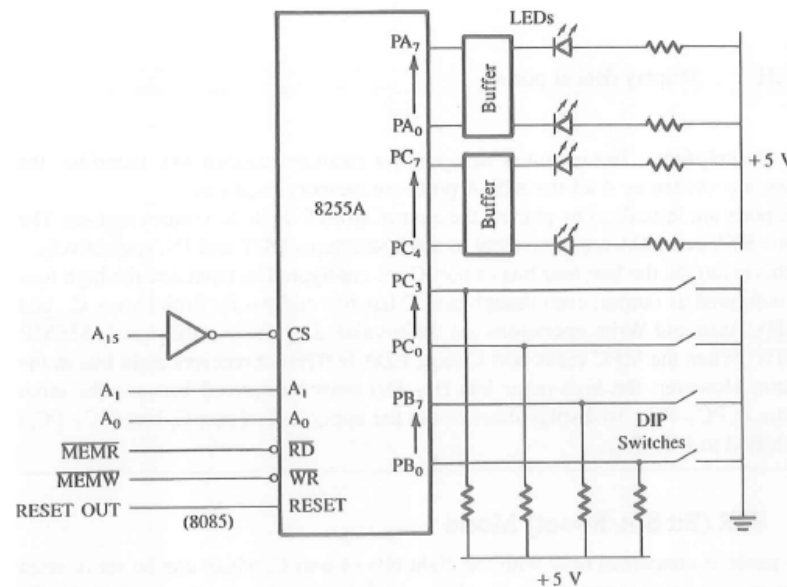
From an analysis of the above routine, the following points can be noted:

1. To set/reset bits in port C, a control word is written in the control register and not in port C.
2. A BSR control word affects only one bit in port C.
3. The BSR control word does not affect the I/O mode.

Mode 0: Simple I/O

- ❖ In this mode, PORT A and PORT B are used as two simple 8-bit I/O ports and PORT C as two 4-bit ports.
- ❖ Each port (or half port, in case of C) can be programmed to function as simply an input or output port.
- ❖ The input/output features in Mode 0 are as follows:
 - Outputs are latched
 - Inputs are latched
 - Ports do not have handshake or interrupt capability

Example 2: write a program to read the switches and display the reading from PORTB at PORTA and from PORTC lower at PORTC upper.

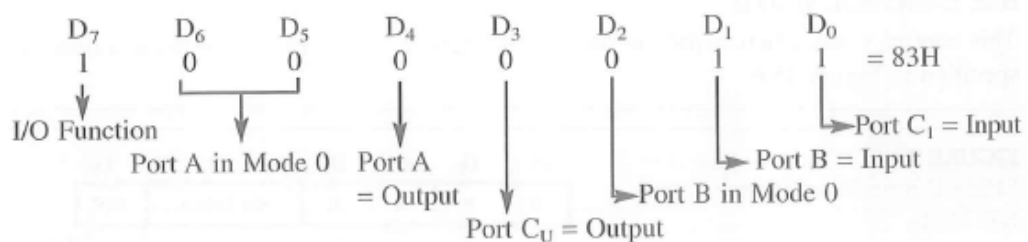


Solution:

Port Address:

Port A	=	8000H ($A_1 = 0, A_0 = 0$)
Port B	=	8001H ($A_1 = 0, A_0 = 1$)
Port C	=	8002H ($A_1 = 1, A_0 = 0$)
Control Register	=	8003H ($A_1 = 1, A_0 = 1$)

Control Word:



Program

```
MVI A,83H      ;Load accumulator with the control word
STA 8003H      ;Write word in the control register to initialize the ports
LDA 8001H      ;Read switches at port B
STA 8000H      ;Display the reading at port A
LDA 8002H      ;Read switches at port C
ANI 0FH       ;Mask the upper four bits of port C; these bits are not input data
RLC            ;Rotate and place data in the upper half of the accumulator
RLC
RLC
RLC
STA 8002H      ;Display data at port CU
HLT
```

Mode 1: I/O with Handshake Signals

- ❖ In Mode 1, handshake signals are exchanged between the MPU and peripherals prior to data transfer.
- ❖ The features of this mode include the following:
 - Two ports (A and B) function as 8-bit I/O ports. They can be configured either as input or output ports
 - Each port uses three lines from PORTC as handshake signals. The remaining two lines of PORTC can be used for simple I/O functions.
 - Input and output data are latched.
 - Interrupt logic is supported.
- ❖ In the 8255A, the specific lines from PORT C used for handshake signals vary according to the I/O functions of a port.
- ❖ Therefore, input and output functions in Mode 1 are discussed separately.

❖ Mode 1: Input Control Signals

- ❖ Figure 3.6 (a) shows the associated control signals used for handshaking when ports A and B are configured as input ports.
- ❖ Port A uses the upper three signals PC₃, PC₄ and PC₅.
- ❖ Port B uses the lower three signals PC₂, PC₁ and PC₀.
- ❖ The function of these signals are as follows:
 - **\overline{STB} (Strobe input):** This signal is generated by a peripheral device to indicate that it has transmitted a byte of data. The 8255A, in response to \overline{STB} , generates IBF and INTR.
 - **IBF (Input Buffer Full):** This signal is an acknowledgement by the 8255A to indicate that the input latch has received the data byte. This is reset when the MPU reads the data.
 - **INTR (Interrupt Request):** This is an output signal that may be used to interrupt the MPU. This signal is generated if \overline{STB} , IBF and INTE (Internal flip-flop) are all at logic 1. This is reset by the falling edge of the RD signal.

- **INTE (Interrupt Enable):** This is an internal flip-flop used to enable or disable the generation of the INTR signal. The two flip-flops $INTE_A$ and $INTE_B$ are set/reset using the BSR mode. The $INTE_A$ is enabled or disabled through PC_4 and $INTE_B$ is enabled or disabled through PC_2 .

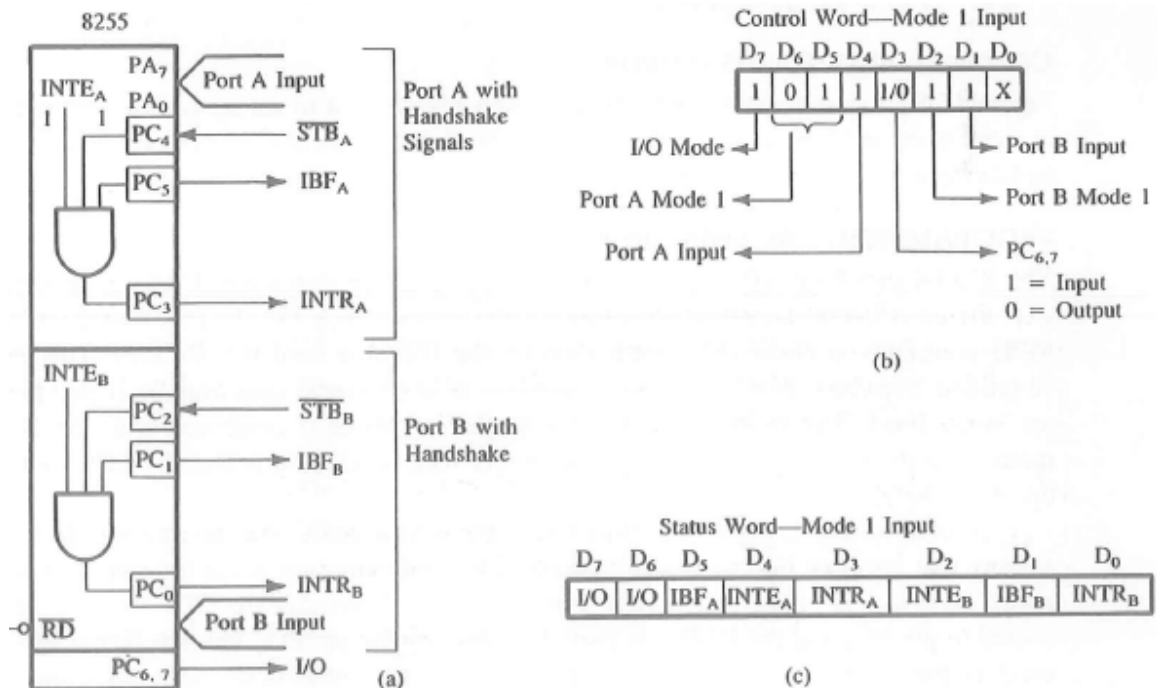


Figure 3.6 Mode 1: Input Configuration

❖ **Mode 1: Output Control Signals**

- ❖ Figure 3.7 shows the control signals when Ports A and B are configured as output ports. These signals are defined as follows:
 - **OBF (Output Buffer Full):** This is an output signal that goes low when the MPU writes data into the output latch of 8255A. This signal indicates to an output peripheral that new data are ready to be read. It goes high again after the 8255A receives an \overline{ACK} from the peripheral.
 - \overline{ACK} (Acknowledge): This is an input signal from a peripheral that must output a low when the peripheral receives the data from 8255A ports.
 - **INTR (Interrupt Request):** This is an output signal, and it is set by rising edge of the \overline{ACK} signal. This signal can be used to interrupt the MPU to request the next data byte for output. The INTR is set when \overline{OBF} , \overline{ACK} and INTE are all one and reset by the falling edge of \overline{WR} .
 - **INTE (Interrupt Enable):** This is an internal flip-flop to a port and needs to be set to generate the INTR signal. The two flip-flops $INTE_A$ and $INTE_B$ are controlled by bits PC6 and PC2, respectively, through the BSR mode.
 - **PC_{4,5}:** These two lines can be set up either as input or output.

- ❖ The remaining three signals from port C can be used either as simple I/O or as handshake for Port B.
- ❖ Figure 3.8 shows two configuration of mode 2.

Keyboard & display unit [8279 IC]:

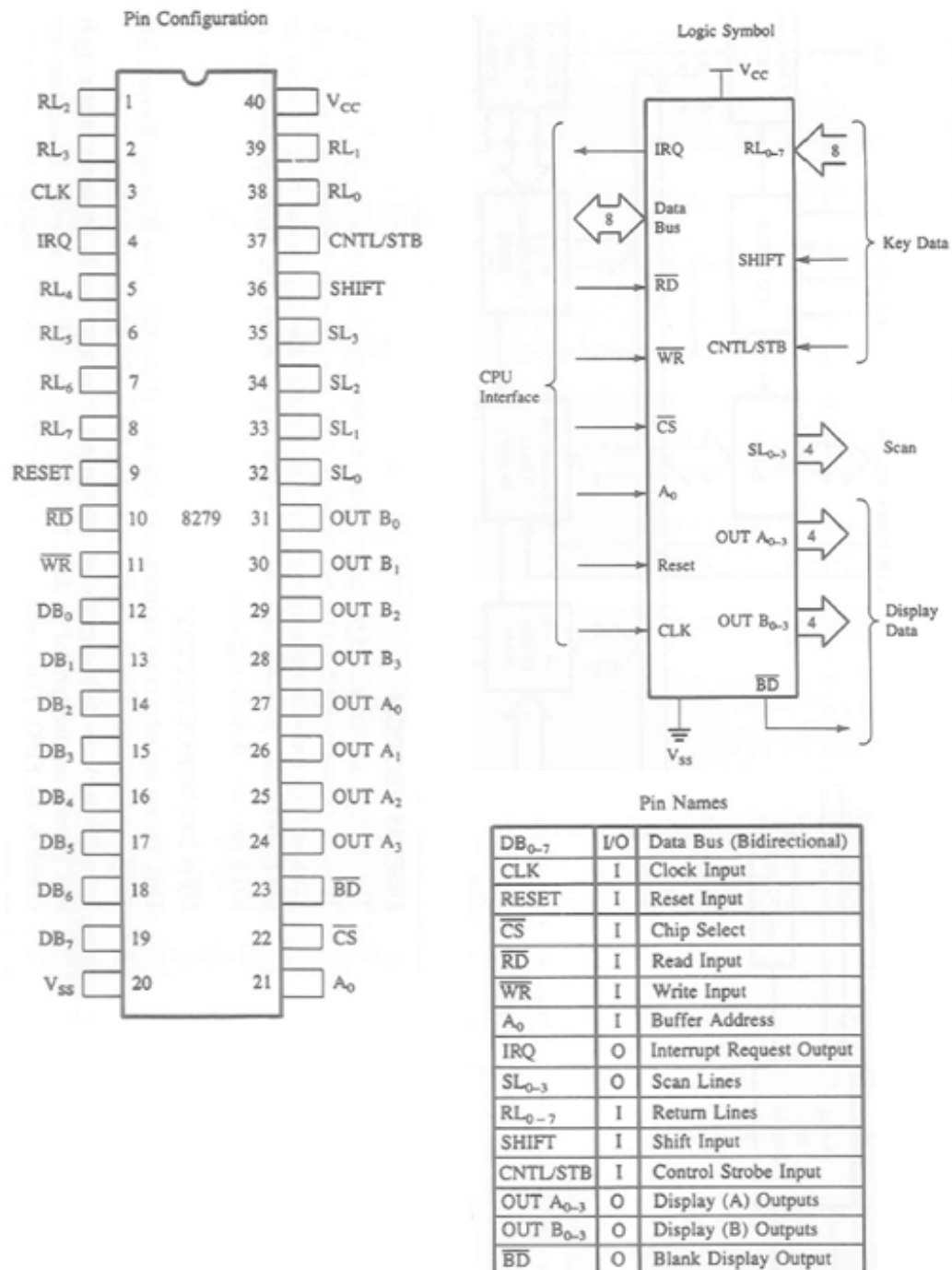


Figure 3.9 8279 Logic Pinout

- ❖ It is a hardware approach to interfacing a matrix keyboard and a multiplexed display.
- ❖ The software approach to interfacing a matrix keyboard and seven segment LEDs has a drawback that the microprocessor is occupied for a considerable amount of time in checking the keyboard and refreshing the display.
- ❖ The 8279 relieves the processor from these two tasks.
- ❖ The disadvantage of using 8279 is cost.
- ❖ The 8279 (Figure3.9) is a 40 pin device with two major segments:
 - Keyboard
 - Display.
- ❖ The keyboard segment can be connected to a 64 contact key matrix.
- ❖ Keyboard entries are debounced and stored in the internal FIFO memory; an interrupt signal is generated with each entry.
- ❖ The display segment can provide a 16 character scanned display interface with such devices as LEDs.
- ❖ This segment has 16 X 8 R/W memory (RAM), which can be used to read/write information for display purposes.
- ❖ The display can be set up in either right-entry or left-entry format.

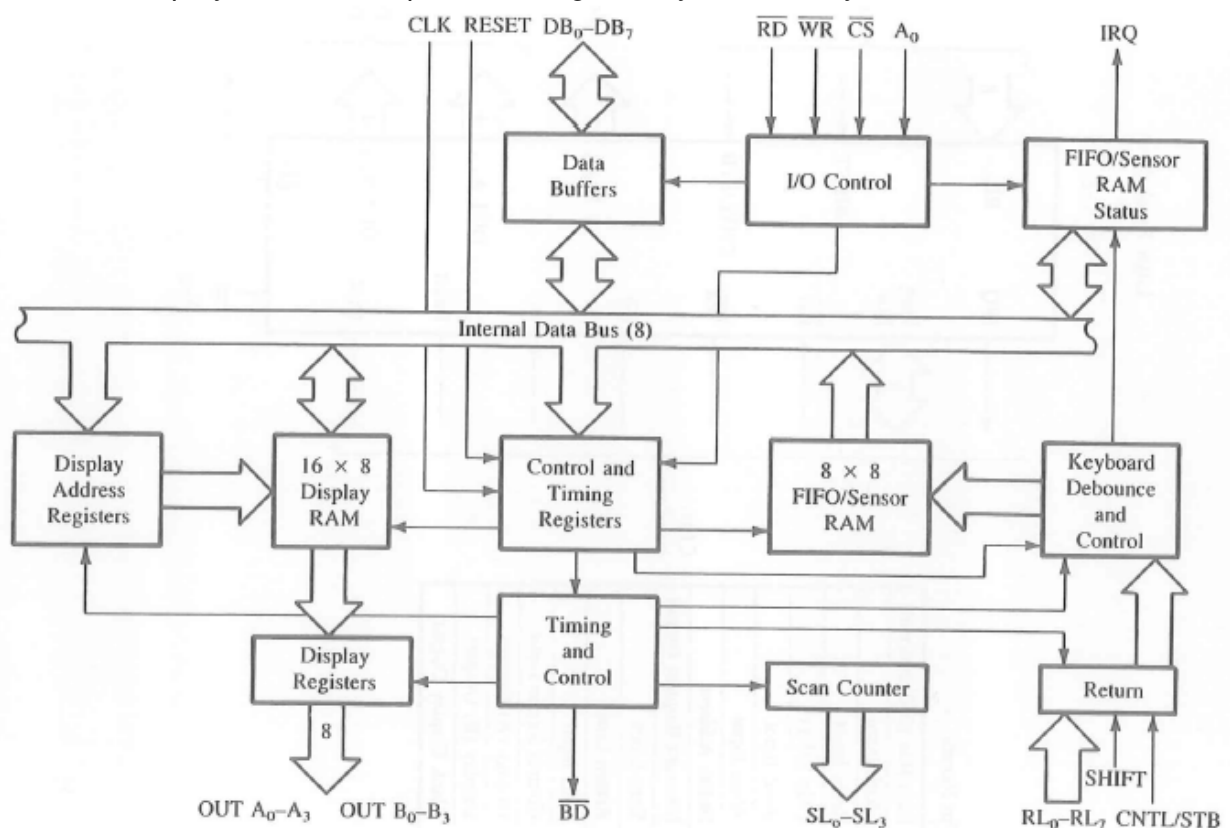


Figure 3.10 8279 Logic Block Diagram

Block Diagram:

- ❖ The block diagram (figure 3.10) shows four major sections of the 8279:
 - Keyboard
 - Scan
 - Display
 - MPU interface.
- ❖ The functions of these sections are described below.
- ❖ **Keyboard Section:**
 - This section has eight lines (RL₀-RL₇) that can be connected to eight columns of a keyboard, plus two additional lines: Shift and CNTL/STB (Control/Strobe).
 - The status of the SHIFT key and control key can be stored along with a key closure.
 - The keys are automatically debounced, and the keyboard can operate in two modes:
 - Two Key Lockout
 - N - Key Roller
 - In the **two key lockout mode**, if two keys are pressed simultaneously, only the **first key is recognized**.
 - In the N-key rollover mode, simultaneous keys are recognized and their codes are stored in the internal buffer; it can also be set up so that no key is recognized until only one key remains pressed.
 - The keyboard section also includes 8 X 8 FIFO RAM.
 - The FIFO RAM consists of eight registers that can store eight keyboard entries; each is then read in the order of entries.
 - The status logic keeps track of the number of entries and provides an IRQ (Interrupt Request) signal when the FIFO is not empty.
- ❖ **Scan Section:**
 - The scan section has a scan counter and four scan lines (SL₀-SL₃).
 - These lines can be decoded using 4 to 16 decoder to generate 16 lines for scanning.
 - These lines can be connected to the rows of a matrix keyboard and the digit drivers of a multiplexed display.
- ❖ **Display Section:**
 - This section has 8 output lines divided into two groups A₀-A₃ and B₀-B₃.
 - These lines can be used, either as a group of eight lines or as two groups of four, in conjunction with the scan lines for multiplexed display.
 - The display can be blanked by using the \overline{BD} line.
 - This section includes 16X8 display RAM.
 - The MPU can read from or write into any of these registers.

❖ **MPU interface Section:**

- This section includes eight bidirectional data lines (DB0-DB7), one Interrupt Request line (IRQ), and six lines for interfacing, including the buffer address line (A_0).
- When A_0 is high, signals are interpreted control words or status; when A_0 is low, signals are interpreted as data.
- The IRQ line goes high whenever data entries are stored in the FIFO.
- This signal is used to interrupt the MPU to indicate the availability of data.

❖ **Key Debounce:**

- When a mechanical pushbutton key, shown in figure (a), is pressed or released, the metal contacts of the key momentarily bounce before giving a steady-state reading, as shown in figure (b).

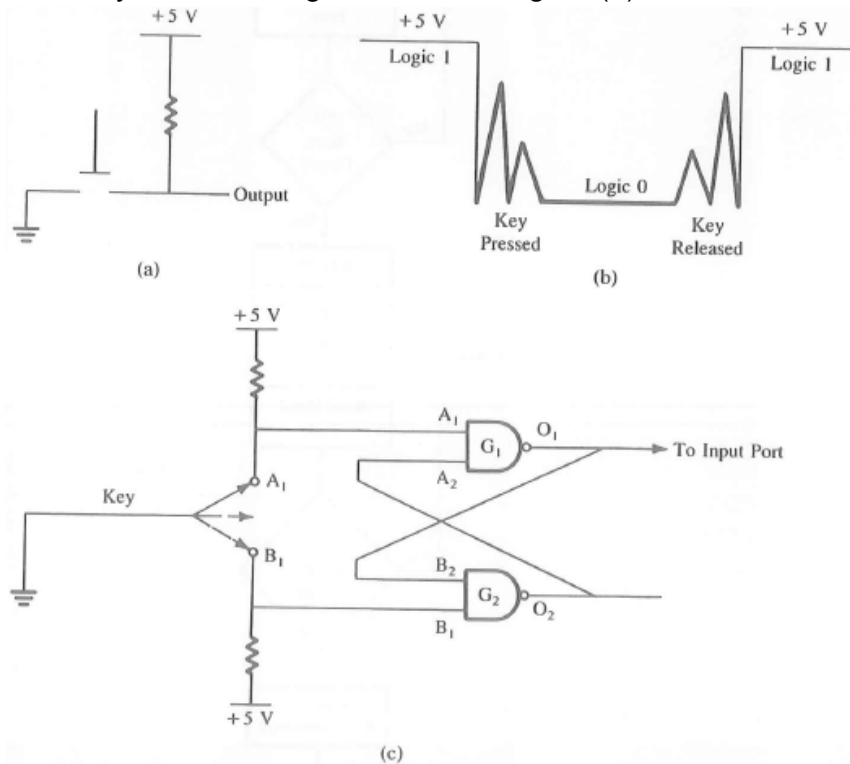


Figure 3.11 (a) Pushbutton Key, (b) Key Debounce and (c) Key Debounce Circuit Using NAND Gates

- Therefore, it is necessary that the bouncing of the key should not be read as in input.
- The key bounce can be eliminated from input data by the key-debounce technique, using either hardware or software.
- Figure (c), shows a key debounce circuit.
- In this circuit, the outputs of the NAND gates do not change even if the key is released from position A_1 .

- The outputs change when the key makes a contact with position B1. When the key is connected to A₁, A₁ goes low.
- If one of the inputs to gate G₁ is low, the output O₁ becomes 1, which makes B₂ high.
- Because line B₁ is already high, the output of O₂ goes low, which makes A₂ low.
- When the key connection is released from A₁, it goes high, but because A₂ is low the output doesn't change.
- When the key makes contact (+5V) to another contact (ground), the output does not change during the transition period, thus eliminating multiple readings.
- In the software technique, when a key closure is found, the microprocessor waits for 10 to 20 ms before it accepts the key as an input.
- The delay routine is as follows:

```

                                PUSH B
                                PUSH PSW
                                LXI B, COUNT
LOOP:                          DCX B
                                MOV A,C
                                ORA B
                                JNZ LOOP
                                POP PSW
                                POP BC
                                RET

```

Interrupt controller interface [8259] :

- ❖ The 8259A is a programmable interrupt controller designed to work with microprocessors 8085, 8086 and 8088.
- ❖ The 8259A interrupt controller can
 1. Manage 8 interrupts according to the instructions written into its control registers. This is equivalent to providing eight interrupt pins on the processor in place of one INTR (8085) pin.
 2. Vector an interrupt request anywhere in the memory map. However all eight interrupts are spaced at the interval of either four or eight locations. This eliminates the major drawback of the 8085 interrupts in which all interrupts are vectored to memory locations on page 00H.
 3. Resolve eight levels of interrupt priorities in a variety of modes, such as
 - Fully nested mode
 - Automatic rotation mode
 - Specific rotation mode
 4. Mask each interrupt request individually.

5. Read the status of pending interrupts, in-service interrupts and masked interrupts.
6. Be set up to accept either the level triggered or the edge triggered interrupt request.
7. Be expanded to 64 priority levels by cascading additional 8259A
8. Be setup to work with either the 8085 microprocessor mode or the 8086/8088 microprocessor mode.

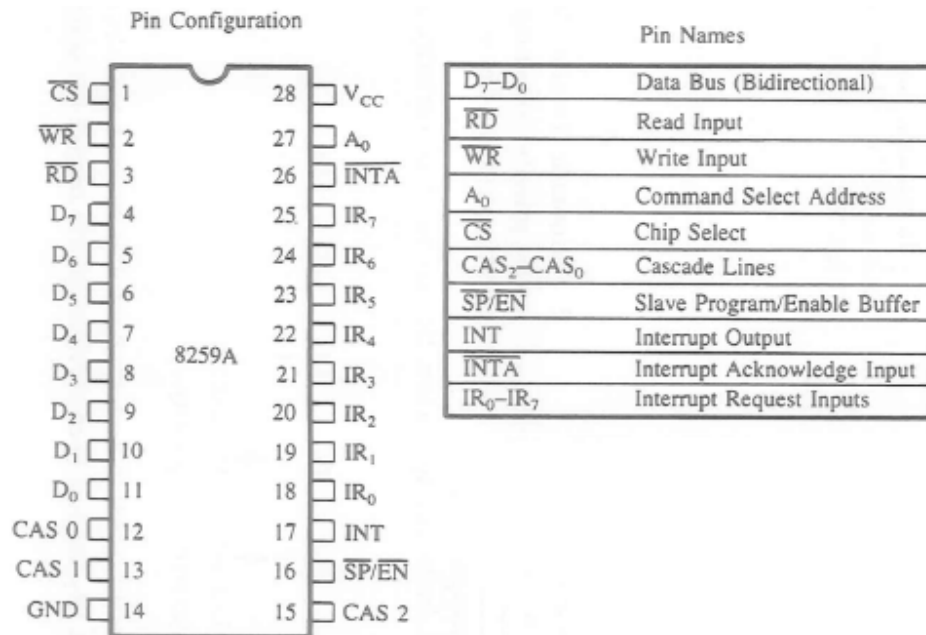


Figure 3.12 Pin Diagram of 8259

Block Diagram:

- ❖ Figure 3.12 shows the pin details of 8259A and figure 3.13 shows internal block diagram of it.
- ❖ It includes eight blocks:
 - Control logic
 - Read/Write logic
 - Data bus buffer
 - Three registers (IRR, ISR and IMR)
 - Priority resolver
 - Cascade buffer
- ❖ This diagram shows all the elements of a programmable device, plus additional blocks.
- ❖ The functions of some of these blocks need explanation which is given below.
- ❖ **READ/WRITE LOGIC**
 - This is a typical Read/Write control logic.

- When the address line A_0 is at logic 0, the controller is selected to write a command or read a status.
- The chip select logic and A_0 determine the port address of the controller.

❖ **CONTROL LOGIC:**

- This block has two pins:
 - \overline{INTA} (Interrupt Acknowledge) as an input
 - \overline{INT} (Interrupt) as an output
- The \overline{INT} is connected to the interrupt pin of the MPU.
- Whenever a valid interrupt is asserted, this signal goes high.
- The \overline{INTA} is the Interrupt Acknowledge signal from the MPU.

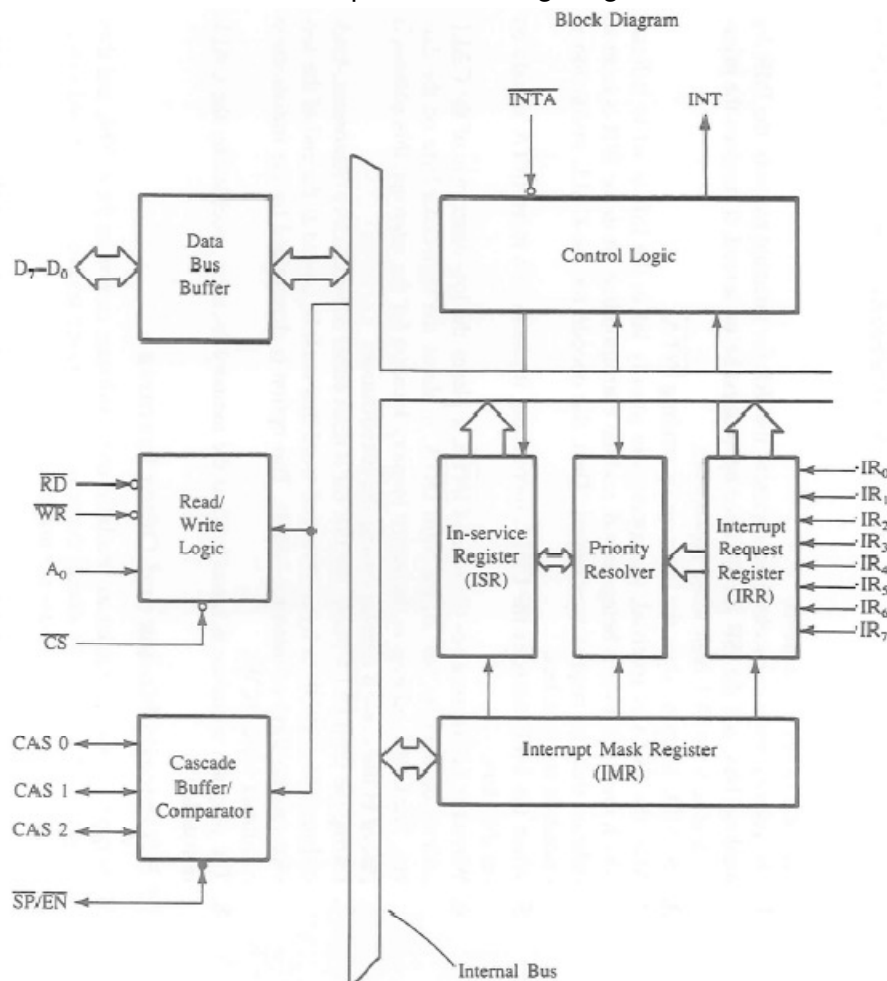


Figure 3.13 Internal Block Diagram

❖ **INTERRUPT REGISTERS AND PRIORITY RESOLVER:**

- The interrupt request register (IRR) has eight lines (IR_0 - IR_7) for interrupts.
- When these lines go high, the requests are stored in the register.

- The In-Service Register (ISR) stores all the levels that are currently being serviced, and the Interrupt Mask Register (IMR) stores the masking bits of the interrupt lines to be masked.
- The Priority Resolver (PR) examines these registers and determines whether INT should be sent to the MPU.

❖ **CASCADE BUFFER/ COMPARATOR:**

- This block is used to expand the number of interrupt levels by cascading two or more 8259As.

Interrupt operation:

- ❖ To implement interrupts, the Interrupt Enable flip-flop in the microprocessor should be enabled by writing the EI instruction, and the 8259A should be initialized by writing control words in the control register.
- ❖ The 8259A requires two types of control words:
 - Initialization Command Words (ICWs)
 - Operational Command Words (OCWs)
- ❖ The ICWs are used to set up the proper conditions and specify RST vector addresses.
- ❖ The OCWs are used to perform functions such as masking interrupts, setting up status-read operations etc.
- ❖ After the 8259A is initialized, the following sequence of events occurs when one or more interrupt request lines go high:
 1. The IRR stores the requests
 2. The priority resolver checks three registers: The IRR for interrupt requests, the IMR for masking bits, and the ISR for the interrupt request being served. It resolves the priority and sets the INT high when appropriate.
 3. The MPU acknowledges the interrupt by sending \overline{INTA} .
 4. After the \overline{INTA} is received, the appropriate priority bit in the ISR is set to indicate which interrupt level is being served, and the corresponding bit in the IRR is reset to indicate that the request is accepted. Then, the opcode for the CALL instruction is placed on the data bus.
 5. When the MPU decodes the CALL instruction, it places two more \overline{INTA} signals on the data bus.
 6. When the 8259A receives the second \overline{INTA} , it places the low-order byte of the CALL address on the data bus. At the third \overline{INTA} , it places the high-order byte on the data bus. The CALL address is the vector memory location for the interrupt; this address is placed in the control register during the initialization.
 7. During the third \overline{INTA} pulse, the ISR bit is reset either automatically (Automatic-End-of-Interrupt – AEOI) or by a command word that must be

issued at the end of the service routine (End-of-Interrupt – EOI). This option is determined by the Initialization Command Word (ICW).

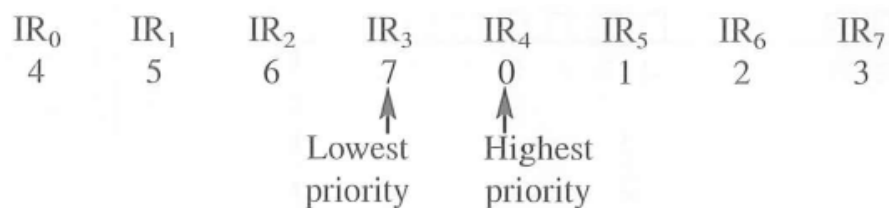
8. The program sequence is transferred to the memory location specified by the CALL instruction.

Priority Modes:

- ❖ Many types of priority modes are available under software control in the 8259A, and they can be changed dynamically during the program by writing appropriate command words. Commonly used priority modes are discussed below.

Fully Nested Mode:

- This is a general purpose mode in which all IRs (Interrupt Requests) are arranged from highest to lowest, with IR₀ as the highest and IR₇ as the lowest.
- In addition any IR can be assigned the highest priority in this mode; the priority sequence will then begin at the IR. In the example below, IR₄ has highest priority, and IR₃ has the lowest priority:



Automatic Rotation Mode:

- In this mode, a device, after being serviced, receives the lowest priority.
- Assuming that the IR₂ has just been serviced, it will receive the seventh priority as shown below:



Specific Rotation Mode:

- This mode is similar to the automatic mode, except that the user can select any IR for the lowest priority, thus fixing all other priorities.

End of Interrupt:

- ❖ After the completion of an interrupt service, the corresponding ISR bit needs to be reset to update the information in the ISR. This is called the End-of-Interrupt (EOI) command.

Nonspecific EOI Command:

- When this command is sent to the 8259A, it resets the highest priority ISR bit.

Specific EOI Command:

- This command specifies which ISR bit to reset.

❖ **Automatic EOI:**

- In this mode, no command is necessary. During the third INTA, the ISR bit is reset. The major drawback with this mode is that the ISR does not have information on which IR is being serviced. Thus, any IR can interrupt the service routine, irrespective of its priority, if the Interrupt Enable flip-flop is set.

Programming 8259A:

- ❖ As mentioned before, the 8259A requires two types of command words: Initialization Command Words (ICWs) and Operational Command Words (OCWs).
- ❖ The 8259A can be initialized with four ICWs; the first two are essential, and the other two are operational based on the modes being used.
- ❖ These words must be issued in a given sequence.
- ❖ Once initialized, the 8259A can be set up to operate in various modes by using three different OCWs; however, they no longer need be issued in a specific sequence.
- ❖ Figure 3.14 shows the bit specification of the first two ICWs. The ICW₁, shown in figure 3.14(a), specifies
 1. Single or multiple 8259As in the system.
 2. 4 or 8-bit interval between the interrupt vector locations.
 3. The address bits A₇-A₅ of the CALL instructions; the rest are supplied by the 8259A, as shown in figure 3.14(b).

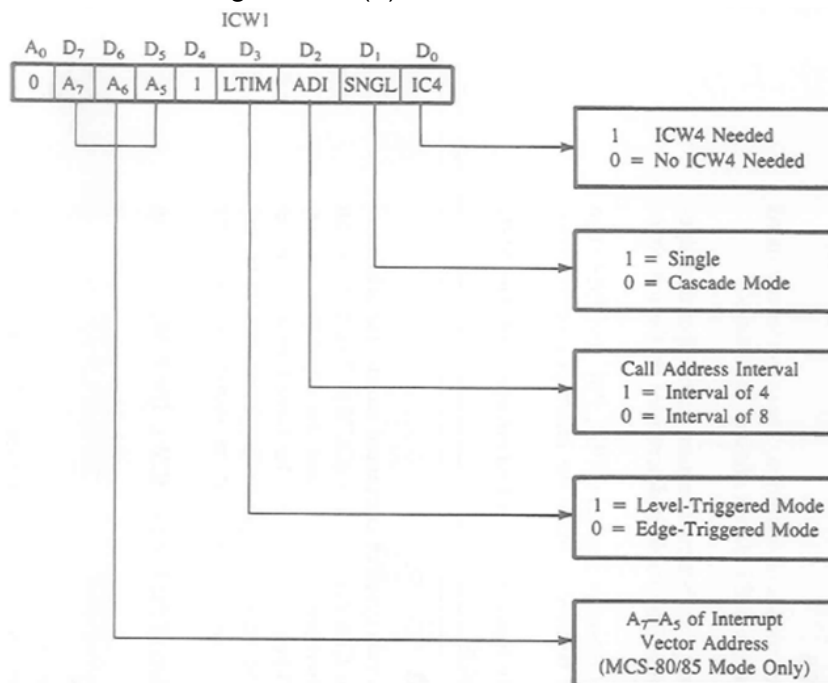


Figure 3.14(a)

Interval = 4								
IR	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
7	A ₇	A ₆	A ₅	1	1	1	0	0
6	A ₇	A ₆	A ₅	1	1	0	0	0
5	A ₇	A ₆	A ₅	1	0	1	0	0
4	A ₇	A ₆	A ₅	1	0	0	0	0
3	A ₇	A ₆	A ₅	0	1	1	0	0
2	A ₇	A ₆	A ₅	0	1	0	0	0
1	A ₇	A ₆	A ₅	0	0	1	0	0
0	A ₇	A ₆	A ₅	0	0	0	0	0

Interval = 8								
IR	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
7	A ₇	A ₆	1	1	1	0	0	0
6	A ₇	A ₆	1	1	0	0	0	0
5	A ₇	A ₆	1	0	1	0	0	0
4	A ₇	A ₆	1	0	0	0	0	0
3	A ₇	A ₆	0	1	1	0	0	0
2	A ₇	A ₆	0	1	0	0	0	0
1	A ₇	A ₆	0	0	1	0	0	0
0	A ₇	A ₆	0	0	0	0	0	0

Figure 3.14(b)

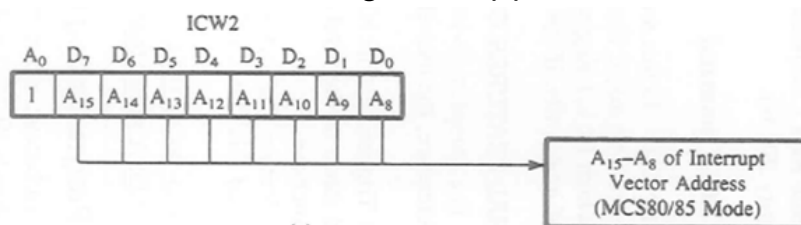


Figure 3.14(C)

- ❖ The ICW₂ of figure 3.14(c) specifies the higher order byte of the CALL instruction.

USART interface [8251 – Programmable Communication Interface]:

- ❖ The 8251A is a programmable chip designed for synchronous and asynchronous serial data communication, packaged in a 28-pin DIP.
- ❖ The 8251A is the enhanced version of its predecessor, the 8251 and is compatible with the 8251.
- ❖ Figure 3.15 shows the block diagram of the 8251A.
- ❖ It includes five sections:
 - Read/Write Control Logic
 - Transmitter
 - Receiver
 - Data Bus Buffer
 - Modem Control
- ❖ The control logic interfaces the chip with the MPU, determines the functions of the chip according to the control word in its register and monitors the data flow.
- ❖ The transmitter section converts a parallel word received from the MPU into serial bits and transmits them over TxD line to a peripheral.
- ❖ The receiver section receives serial bits from a peripheral, converts them into a parallel word, and transfers the word to the MPU.

- ❖ The modem control is used to establish data communication through modems over telephone lines.

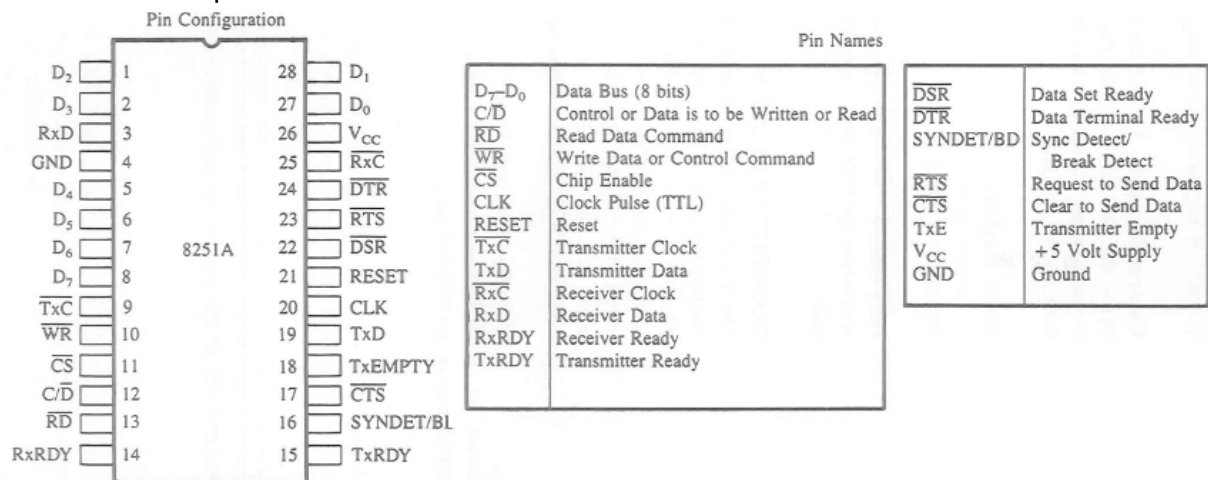


Figure 3.15(a) 8251A Pin Diagram

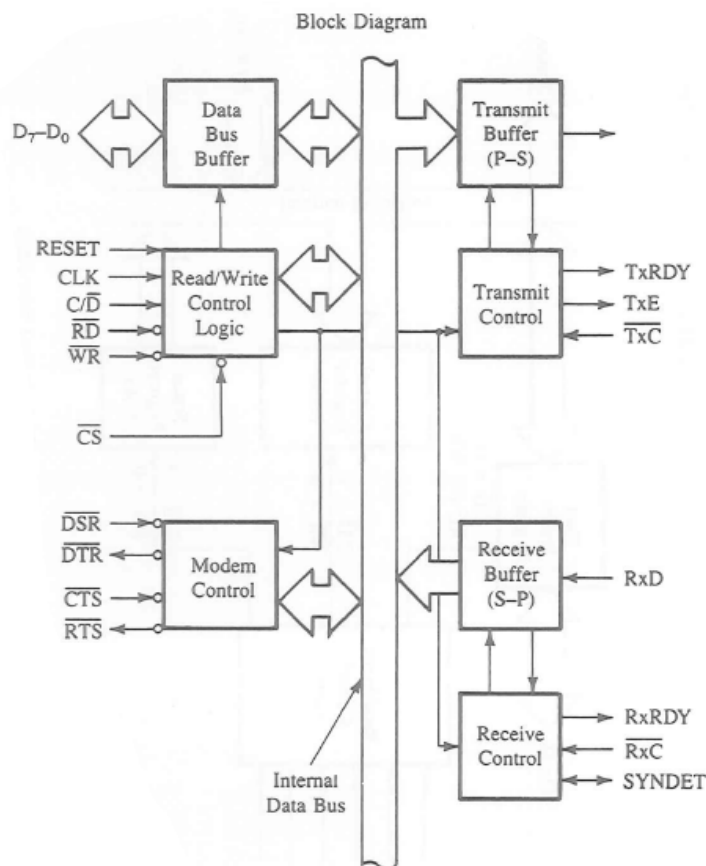


Figure 3.15(b) 8251A Block Diagram

- ❖ The 8251A is a complex device, capable of performing various functions.
- ❖ For the sake of clarity, this chapter focuses only on the asynchronous, mode of serial I/O and excludes synchronous mode.

- ❖ The asynchronous mode is often used for data communication between MPU and serial peripherals such as terminals and floppy disks.
- ❖ Figure 3.16 shows an expanded version of the 8251A block diagram.
- ❖ The block diagram shows all the elements of a programmable chip; it includes the interfacing signals, the control register, and the status register.
- ❖ The functions of various blocks are described below.

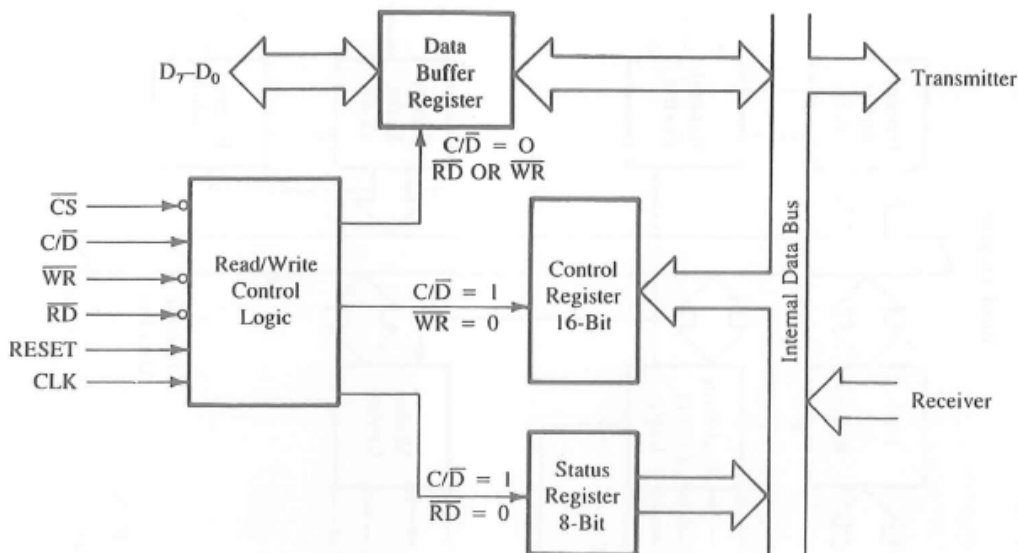


Figure 3.16 8251A Expanded Block Diagram of Control Logic and Registers

READ/WRITE CONTROL LOGIC AND REGISTERS:

- ❖ This section includes R/W control logic, six input signals, control logic and three buffer registers: data register, control register and status register. The input signals to the control logic are as follows.
- ❖ **Input Signals:**
- ❖ **CS – Chip Select:** when this signal goes low, the 8251A is selected by the MPU for communication. This is usually connected to a decoded address bus.
- ❖ **C/D – Control Data:** when this signal is high, the control register or the status register is addressed; when it is low, the data buffer is addressed. The control register and status register are differentiated by \overline{WR} and \overline{RD} signals, respectively.
- ❖ **\overline{WR} - Write:** When this signal goes low, the MPU either writes in the control register or sends output to the data buffer. This is connected to \overline{IOW} or \overline{MEMW} .
- ❖ **\overline{RD} - Read:** When this signal goes low, the MPU either reads a status from the status register or accepts (inputs) data from the data buffer. This is connected to either \overline{IOR} or \overline{MEMR} .
- ❖ **RESET:** A high on this input resets the 8251A and forces it into the idle mode.
- ❖ **CLK – Clock:** This is the clock input, usually connected to the system clock. This clock does not control either the transmission or the reception rate. The clock is necessary for communication with the microprocessor.

- ❖ **Control Register:** this 16-bit register for a control word consists of two independent bytes: the first byte is called mode instruction (word) and the second byte is called the command instruction (word). This register can be accessed as an output port when the C/\bar{D} pin is high.
- ❖ **Status Register:** This input register checks the ready status of a peripheral. This register is addressed as an input port when the C/\bar{D} pin is high; it has the same port address as the control register.
- ❖ **Data Buffer:** This bidirectional register can be addressed as an input port and an output port when the C/\bar{D} pin is low. Table 3.1 summarizes all the interfacing and control signals.

Table 3.1 Summary of Control Signals for the 8251A

\overline{CS}	C/\bar{D}	\overline{RD}	\overline{WR}	Function
0	1	1	0	MPU writes instructions in the control register
0	1	0	1	MPU reads status from the status register
0	0	1	0	MPU outputs data to the Data Buffer
0	0	0	1	MPU accepts data from the Data Buffer
1	X	X	X	USART is not selected

TRANSMITTER SECTION:

- ❖ The transmitter accepts parallel data from the MPU and converts them into serial data.
- ❖ It has two registers;
 - Buffer register to hold eight bits
 - Output register to convert eight bits into a stream of serial bits (figure 3.17)
- ❖ The MPU writes a byte in the buffer register; whenever the output register is empty, the contents of the buffer register are transferred to the output register.
- ❖ This section transmits data on the TxD pin with the appropriate framing bits (Start and Stop).
- ❖ Three output signals and one input signal are associated with the transmitter section.
- ❖ **TxD – Transmit Data:** Serial bits are transmitted on this line.
- ❖ **TxC – Transmitter Clock:** This input signal controls the rate at which bits are transmitted by the USART. The clock frequency can be 1, 16 or 64 times the baud.
- ❖ **TxRDY – Transmitter Ready:** This is an output signal. When it is high, it indicates that the buffer register is empty and the USART is ready to accept a byte. It can be used either to interrupt the MPU or to indicate the status. This signal is reset when a data byte is loaded into the buffer.
- ❖ **TxE – Transmitter Empty:** This is an output signal. Logic 1 on line indicates that the output register is empty. This signal is reset when a byte is transferred from the buffer to the output registers.

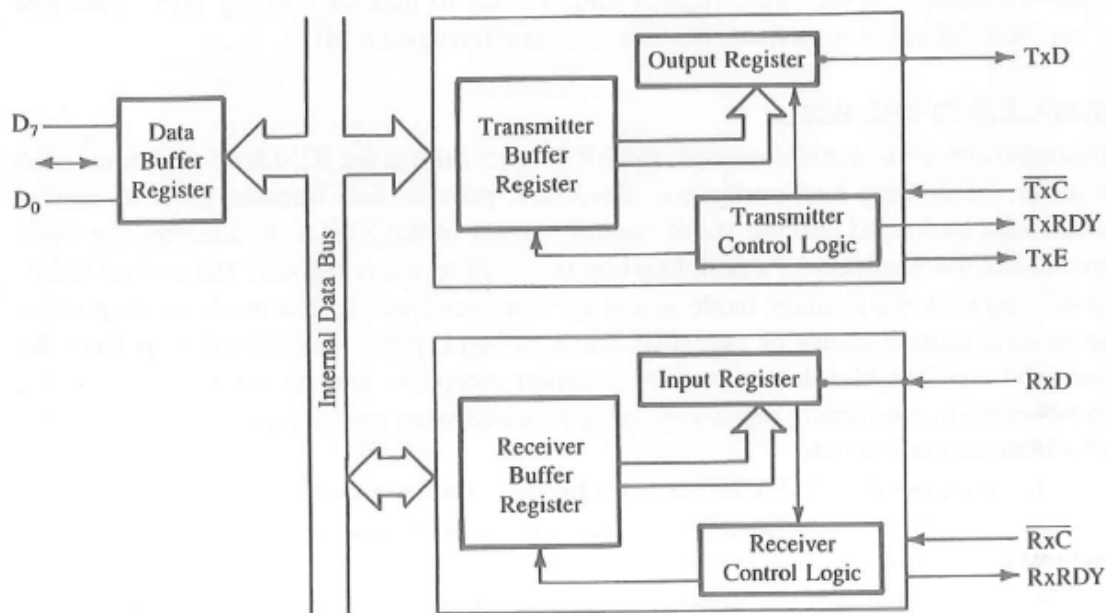


Figure 3.17 8251A Expanded Block Diagram of Transmitter and Receiver Sections

RECEIVER SECTION:

- ❖ The receiver accepts serial data on the RxD line from a peripheral and converts them into parallel data.
- ❖ The section has two registers (figure 3.17);
 - Receiver input register
 - Buffer register
- ❖ When the RxD line goes low, the control logic assumes it is a Start bit, waits for half a bit time, and samples the line again.
- ❖ If the line is still low, the input register accepts the following bits, forms a character, and loads it into the buffer register.
- ❖ Subsequently, the parallel byte is transferred to the MPU when requested.
- ❖ In the asynchronous mode, two input signals and one output are necessary, as described below.
- ❖ **RxD – Receiver Data:** Bits are received serially on this line and converted into a parallel byte in the receiver input register.
- ❖ **RxC – Receiver Clock:** This is a clock signal that controls the rate at which bits are received by the USART. In the asynchronous mode, the clock can be set to 1, 16 or 64 times the baud.
- ❖ **RxRDY – Receiver Ready:** This is an output signal. It goes high when the USART has a character in the buffer register and is ready to transfer it to MPU. This line can be used either to indicate the status or to interrupt the MPU.

DMA controller [8237]:

- ❖ It is an I/O technique commonly used for high speed data transfer, for example, data transfer between system memory and a floppy disk.
- ❖ In status check and interrupt I/O, data transfer is relatively slow because each instruction needs to be fetched and executed.
- ❖ In DMA, the MPU releases the control of the buses to a device called DMA controller.
- ❖ The controller manages data transfer between memory and a peripheral under its control, thus bypassing the MPU.
- ❖ Conceptually this is an important I/O technique, it introduces two new signals available on the 8085 – HOLD and HLDA.
 - HOLD – This is an active high input signal to the 8085 from another master requesting the use of the address and data buses. After receiving the Hold request, the MPU relinquishes the buses in the following machine cycle. All buses are tri-stated and the HLDA signal is sent out. The MPU regains the control of the buses after HOLD goes low.
 - HLDA – This is an active high output signal indicating that the MPU is relinquishing control of the buses.
- ❖ A DMA controller uses these signals as if it were a peripheral requesting the MPU for the control of the buses.
- ❖ The MPU communicates with the controller by using the Chip Select line, buses and control signals.
- ❖ However once the controller has gained control, it plays the role of a processor for data transfer.
- ❖ To perform this function the DMA controller should have
 - A Data Bus
 - An address Bus
 - Read/Write Control Signals
 - Control Signals to disable its role as a peripheral and to enable its role as a processor
- ❖ This process is called switching from the slave mode to the master mode.

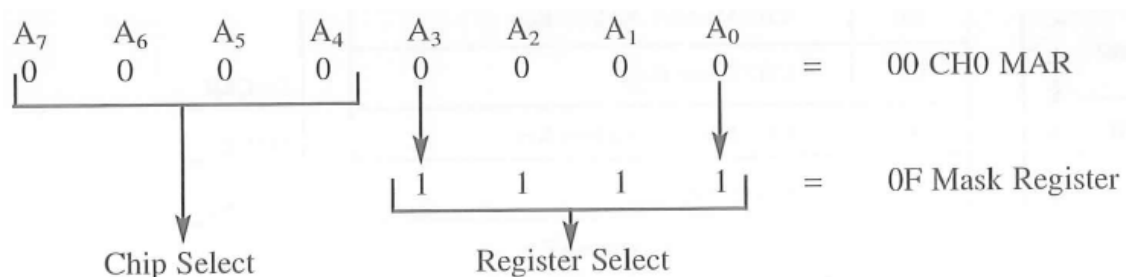
8237 DMA Controller

- ❖ The 8237 is a programmable Direct Memory Access (DMA) housed in a 40-pin package.
- ❖ It has four independent channels with each capable of transferring 64K bytes.
- ❖ It must interface with two types of devices: the MPU and peripherals such as floppy disks.
- ❖ As mentioned earlier, DMA plays two roles in a given system: It is an I/O to the microprocessor (slave mode) and it is a data transfer processor to peripherals such as floppy disks (master mode).

- ❖ Many of its signals that are input in the I/O mode become outputs in the processor mode.
- ❖ It also needs additional signal lines to communicate with the addresses 64K data bytes, and these signals must be generated externally by using latches and buffers.
- ❖ The 8237 is a complex device, to maintain clarity, the following discussion is divided into five segments:
 - DMA channels and Interfacing
 - DMA Signals
 - System Interface
 - Programming
 - DMA execution

❖ **DMA channels and Interfacing**

- ❖ Figure 3.18 shows a logical pin out and internal registers of the 8237.
- ❖ It also shows the interface with the 8085 using a 3-to-8 decoder.
- ❖ The 8237 has four independent channels, CH0 to CH3.
- ❖ Internally, two 16 bit registers are associated with each channel: One is used to load a starting address of the byte to be copied and the second is used to load a count of the number of bytes to be copied.
- ❖ Figure 3.18 shows eight such registers that can be accessed by the MPU.
- ❖ The addresses of these registers are determined by four address lines, A_3 to A_0 , and the Chip Select (\overline{CS}) signal.
- ❖ Address 000 lines A_3 to A_0 selects CH0 Memory Address Register (MAR) and address 001 selects the next register, CH0 Count.
- ❖ Similarly, all the remaining registers are selected in sequential order.
- ❖ The last eight registers are used to write commands or read status as shown.
- ❖ In figure 3.18, the MPU accesses the DMA controller by asserting the signal Y_0 of the decoder.
- ❖ Therefore, the addresses of these internal registers range from 00 to 0FH as follows:



❖ **DMA Signals:**

- ❖ In figure 3.18, signals are divided into two groups:
 - One group of signals shown on the left of the 8237 is used for interfacing with the MPU

- The second group shown on the Right hand side of the 8237 is for Communicating with peripherals.

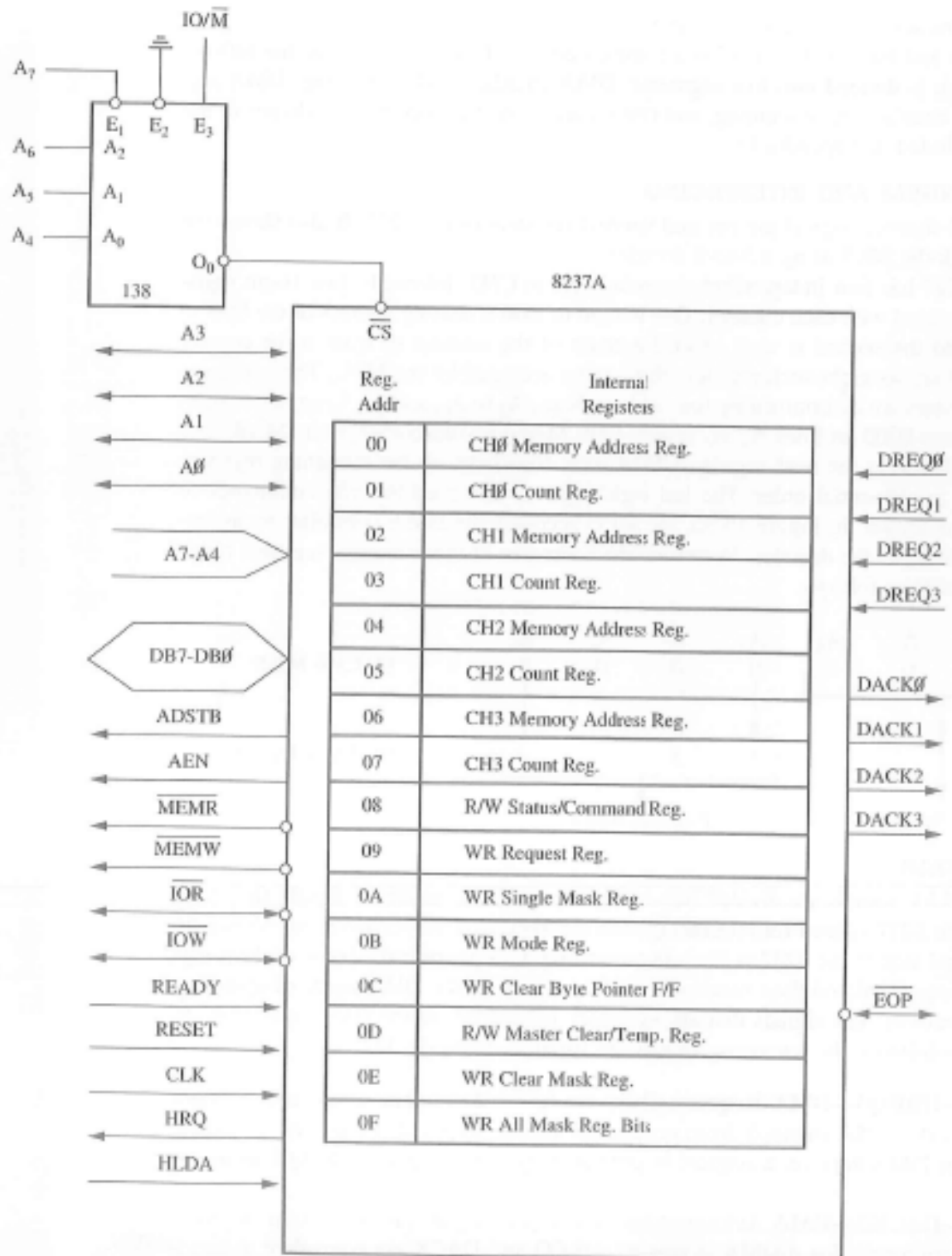


Figure 3.18 8237A – DMA Controller with Internal Registers

- ❖ Some of these signals are bidirectional and their functions are determined by the DMA mode of operation (I/O or Processor).
- ❖ The signals that are necessary to understand the DMA operation are explained as follows;
 - **DREQ0-DREQ3-DMA Request:** These are four independent, asynchronous input signals to the DMA channels from peripherals such as floppy disks and hard disk. To obtain DMA service, a request is generated by activating the DREQ line of the channel.
 - **DACK0-DACK3-DMA Acknowledge:** These are output lines to inform the individual peripherals that a DMA is granted. DREQ and DACK are equivalent to handshake signals in I/O devices.
 - **AEN and ADSTB – Address Enable and Address Strobe:** These are active high output signals that are used to latch a higher order address byte to generate 16-bit address.
 - **\overline{MEMR} and \overline{MEMW} – Memory Read and Memory Write:** These are output signals used during the DMA cycle to write and read from memory.
 - **A₃-A₀ and A₇-A₄ - Address:** A₃-A₀ are bidirectional address lines. They are used as inputs to access control registers. During the DMA cycle, these lines are used as output lines to generate a low-order address that is combined with remaining address lines A₇-A₄.
 - **HRQ and HLDA – Hold Request and Hold Acknowledge:** HRQ is an output signal used to request the MPU control of the system bus. After receiving the HRQ, the MPU completes the bus cycle in process and issues the HLDA signal.

System Interface:

- ❖ The DMA is used to transfer data bytes between I/O (such as a floppy disk) and system memory (or from memory to memory) at high speed.
- ❖ It includes eight data lines, four control signals (\overline{IOR} , \overline{IOW} , \overline{MEMR} and \overline{MEMW}), and eight address lines (A₇ – A₀).
- ❖ However, it needs 16 address lines to access 64K bytes.
- ❖ Therefore, an additional eight lines must be generated as shown in figure 3.19.
- ❖ When a transfer begins, the DMA places the low-order byte on the address bus and the high-order byte on the data bus and asserts AEN (Address Enable) and ADSTB (Address Strobe).
- ❖ These two signals are used to latch the high-order byte from the data bus; thus, it places the 16-bit address on the system bus.
- ❖ After the transfer of the first byte, the latch is updated when the lower byte generates a carry (or borrow).
- ❖ Figure 3.19 shows two latches: one latch (373#1) to latch a high-order address from the data bus by using the AEN and ADSTB signals, and the second latch

(373#2) to demultiplex the 8085 bus and generate the low-order address bus by using the ALE (from 8085) signal.

- ❖ The AEN signal is connected to the OE signal of the second latch to disable the low-order address bus from the 8085 when the first latch is enabled to latch the high-order byte of the address.

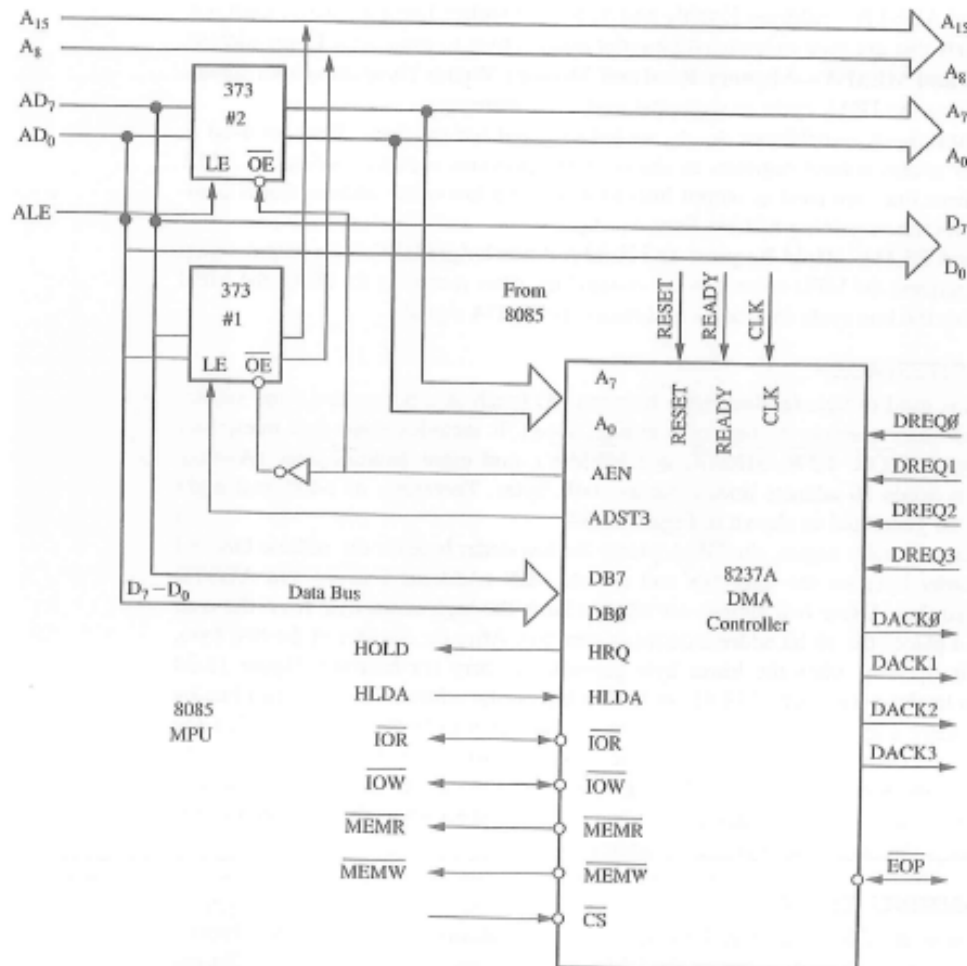


Figure 3.19 Interfacing 8237A-DMA Controller with the 8085

Programming the 8237:

- ❖ To implement the DMA transfer, the 8237 should be initialized by writing into various control registers.
- ❖ To initialize the 8237, the following steps are necessary.
 1. Write a control word in the Mode register that selects the channel and specifies the type of transfer (read, Write or Verify) and the DMA mode (block, single-byte, etc)
 2. Write a control word in the command register that specifies parameters such as priority among four channels, DREQ and DACK active levels, and timing, and enables the 8237.

3. Write the starting address of the data block to be transferred in the channel Memory Address Register (MAR)
4. Write the count (the number of the bytes in the data block) in the channel Count register.

Example:

Write initialization instructions for the DMA controller in figure 3.18 to meet the following specifications. Use the same port (register) address (00 to 0FH) as in figure 3.18.

1. Disable the DMA controller and begin writing initialization instructions.
2. Initialize Channel #3 (CH3) to transfer 1K of bytes from the system memory to the floppy disk assigned to CH3.
3. The starting address of the data block is 4075H and subsequent data bytes have memory addresses in increasing order.
4. The Command parameters should be: normal timing, fixed priority, late write. DREQ and DACK are both active low.
5. Set up the demand mode whereby the DMA can complete the data transfer without any interruption.

The initialization instructions to set up the DMA controller are as follows (for a detailed explanation see Appendix D):

Solution

```

MVI A, 00000100B ;Command: 0 0 0 0 0 1 0 0
                                ↓
                                Disable DMA

OUT 08H           ;Send to Command Reg.
MVI A, 00000111B ;Mode:  0 0      0      0      0 1      1 1
                                ↓      ↓      ↓      ↓      ↓
                                Demand Increment Disable Write Ch 3
                                Mode  Address  Auto  Load

OUT 0BH           ;Send to Mode Reg.
MVI A, 75H        ;Low-order byte of starting address
OUT 06H           ;Output to CH3 Memory Address Reg.
MVI A, 40H        ;High-order byte of starting address
OUT 06H           ;Output to CH3 Memory Address Reg.
MVI A, FFH        ;Low-order byte of the count 03FFH
OUT 07H           ;Output to CH3 Count Reg.
MVI A, 03H        ;High-order byte of the count 03FFH
OUT 07H           ;Output to CH3 Count Reg.
MVI A, 10000000B ;Command: 1 0      0      0      0      0 0
                                ↓      ↓      ↓      ↓      ↓      ↓
                                DACK Late Fixed Normal DMA Disable
                                DREQ Write Priority Time Enable Mem
                                High      to
                                           Mem

```

OUT 08H ;Send to Command Reg.

DMA Execution:

- ❖ The process of data transfer from the peripheral to the system memory under the DMA controller can be classified under two modes:
 1. The slave mode

2. The master mode.

❖ **Slave Mode:**

- ❖ In the slave mode, the DMA controller is treated as a peripheral, using the following steps:
 1. The MPU selects the DMA controller through Chip Select
 2. The MPU writes the control words as illustrated in above example in channel registers and command/status registers by using control signals \overline{IOW} and \overline{IOR} .
- ❖ In this mode, the output signals of the 8237, such as A7-A4, \overline{MEMW} and \overline{MEMR} are in tri-state.

❖ **Master Mode:**

- ❖ After the initialization, the 8237 in master mode keeps checking for a DMA request, and the steps in data transfer can be listed as follows:
 1. When the peripheral is ready for data transfer, it sends a high signal to DRQ.
 2. When the DRQ has been received and the channel enabled, the control logic sets HRQ (Hold Request) high, (HRQ is connected to the HOLD signal of the 8085)
 3. In the next cycle, the MPU relinquishes the buses and sends the HLDA (hold Acknowledge) signal to the 8237.
 4. After receiving the HLDA signal, the DMA asserts AEN signal high. The high AEN signal disables 373 Latch#2, thus disconnecting the demultiplexed bus A7-A0 of the MPU and enables 373 Latch#1 through an inverter. Next the DMA asserts ADSTB (Address Strobe) high that is connected to Latch Enable (LE) of 373 Latch#1 and places the contents of the data bus, which is a high-order byte of the starting address, A15-A8. At the same time, the DMA also outputs the low order address A7-A0 on the low-order address bus.
 5. When the entire address A15-A0 is available on the address bus, the DMA sends DACK to the peripheral.
 6. The DMA controller continues the data transfer by asserting the necessary control signals (\overline{IOR} , \overline{IOW} , \overline{MEMR} or \overline{MEMW}) until DACK remains high.
 7. At the end of the data transfer, the DMA asserts \overline{EOP} (End of Process) signal low that can be used to inform the peripheral that the data transfer is complete. The DMA data transfer can also be terminated by sending a low signal to \overline{EOP} from outside.