Programmable Interfacing Devices

General-Purpose Programmable Peripheral Devices

- PPI – Programmable Peripheral Interface
- It is an I/O port chip used for interfacing I/O devices with microprocessor.
- Very commonly used programmable devices from Intel family are:
  - The 8255A peripheral interface.
  - The 8254 Interval Timer.
  - The 8294A Interrupt Controller.
  - The 8237 DMA controller.

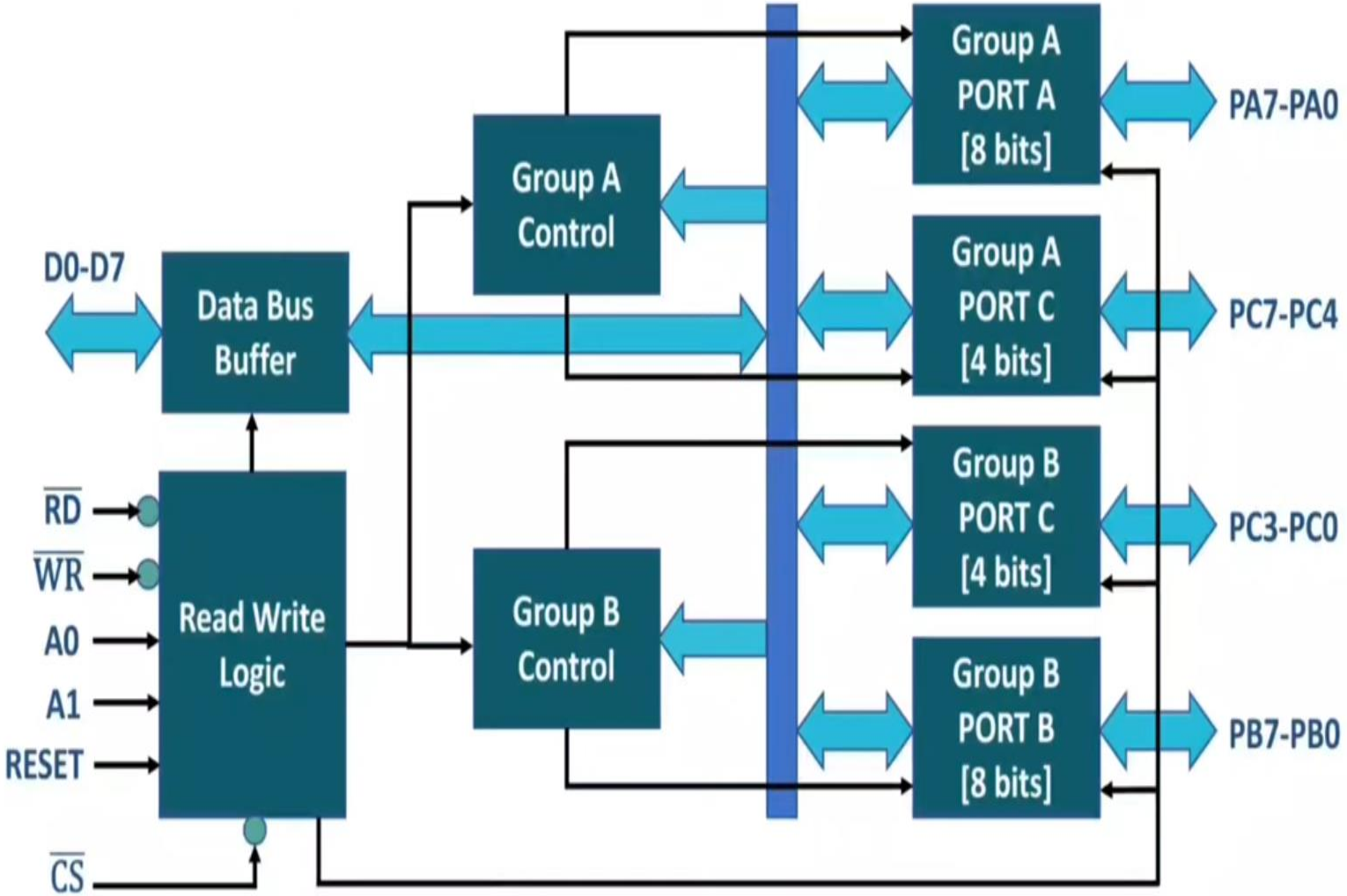
8255 Programmable Peripheral Interface

❖ Features of 8255 Programmable Peripheral Interface

- ❑ 8255 is designed to work with various microprocessors like 8085, 8086 etc.
- ❑ 8255 is designed to increase capacity of Input Output Interface.
- ❑ 8255 has three 8bits bidirectional IO ports.
- ❑ 8255 has three IO modes of transfer data:
  - Simple IO mode
  - Handshake IO mode
  - Bidirectional handshake IO mode
- ❑ 8255 has BSR mode to alter individual bits of port C.

8255A Programmable Peripheral Interface

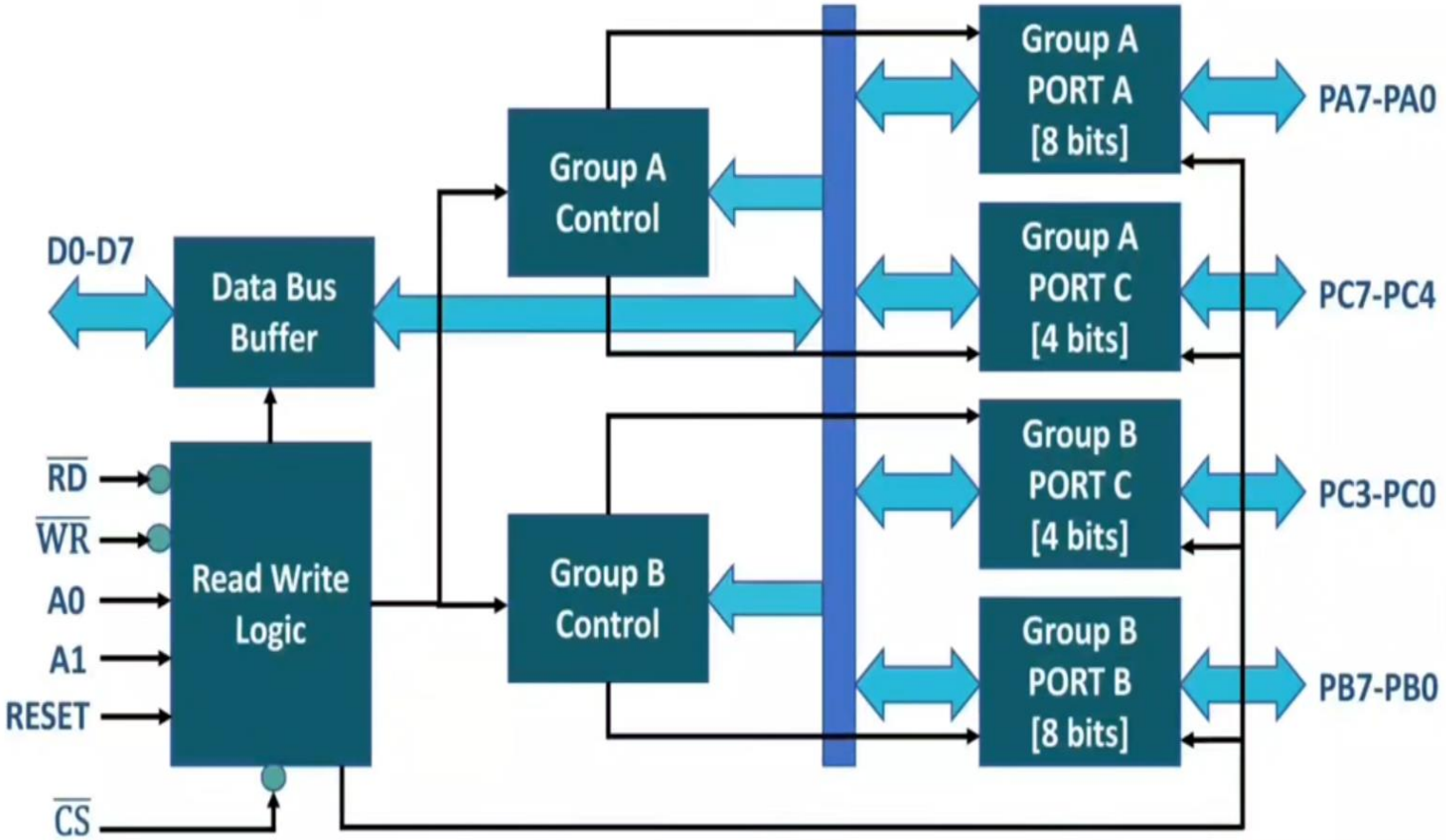
- The 8255A is a general purpose programmable I/O device designed to transfer the data from I/O to interrupt I/O under certain conditions as required. It can be used with almost any microprocessor.
- It consists of three 8-bit bidirectional I/O ports (24 I/O lines) which can be configured as per the requirement.



❖ Data Bus Buffer

- ❑ It has bidirectional data bus D0 – D7.
- ❑ D0 – D7 is interfaced with system data bus of Microprocessor.

Read Write Control Logic of 8255

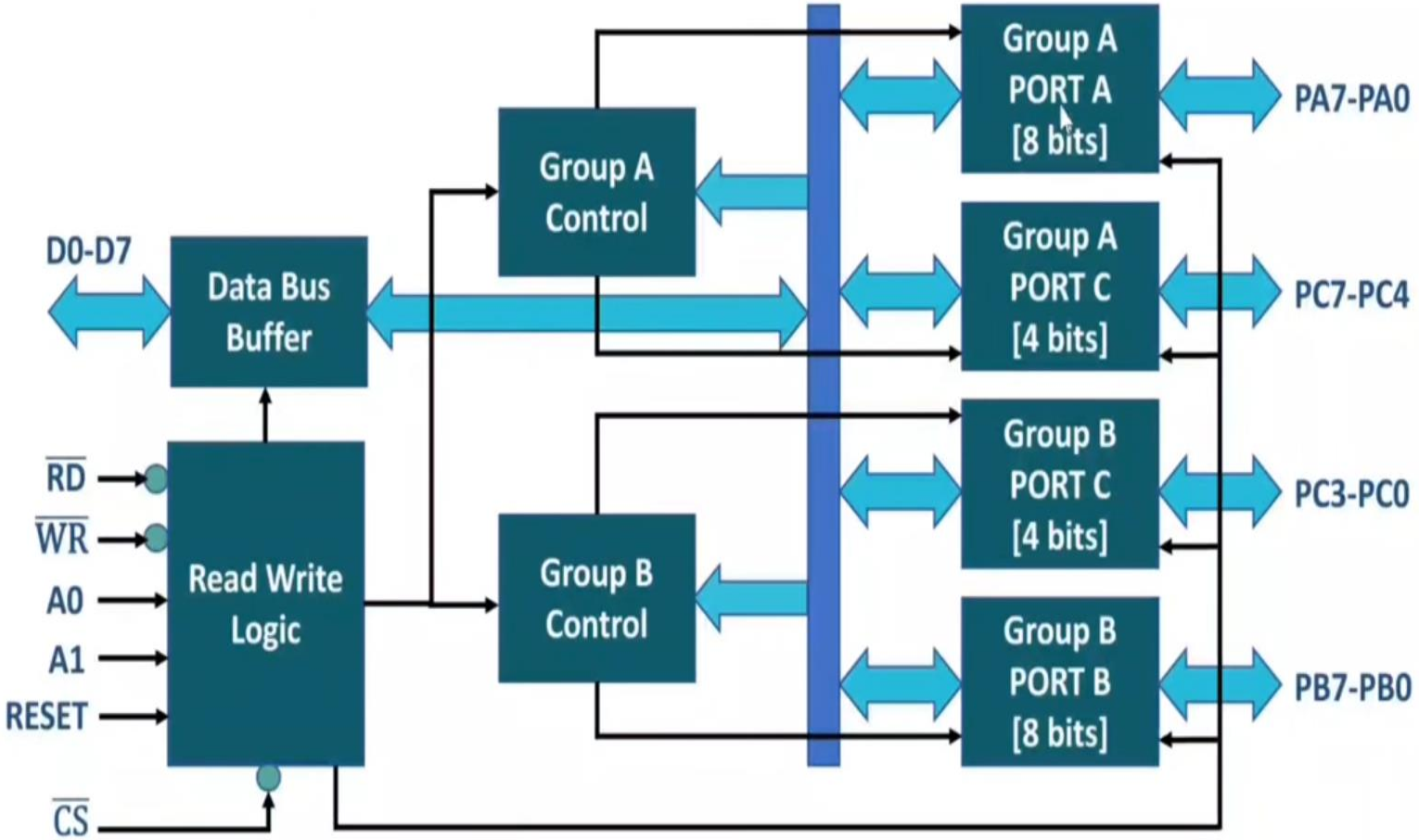


❖ Read Write Control Logic

- ❑ 8255 read and write data as per control signals $\overline{RD}$ and $\overline{WR}$ connected with microprocessor.
- ❑ RESET will reset 8255.
- ❑ A1 and A0 used to select port and control word.
- ❑ $\overline{CS}$ is used to select chip of 8255.

$\overline{CS}$	A1	A0	Selected	Sample Address
0	0	0	Port A	80H [1000 0000]
0	0	1	Port B	81H [1000 0001]
0	1	0	Port C	82H [1000 0010]
0	1	1	Control Register	83H [1000 0011]
1	X	X	8255 is not selected	

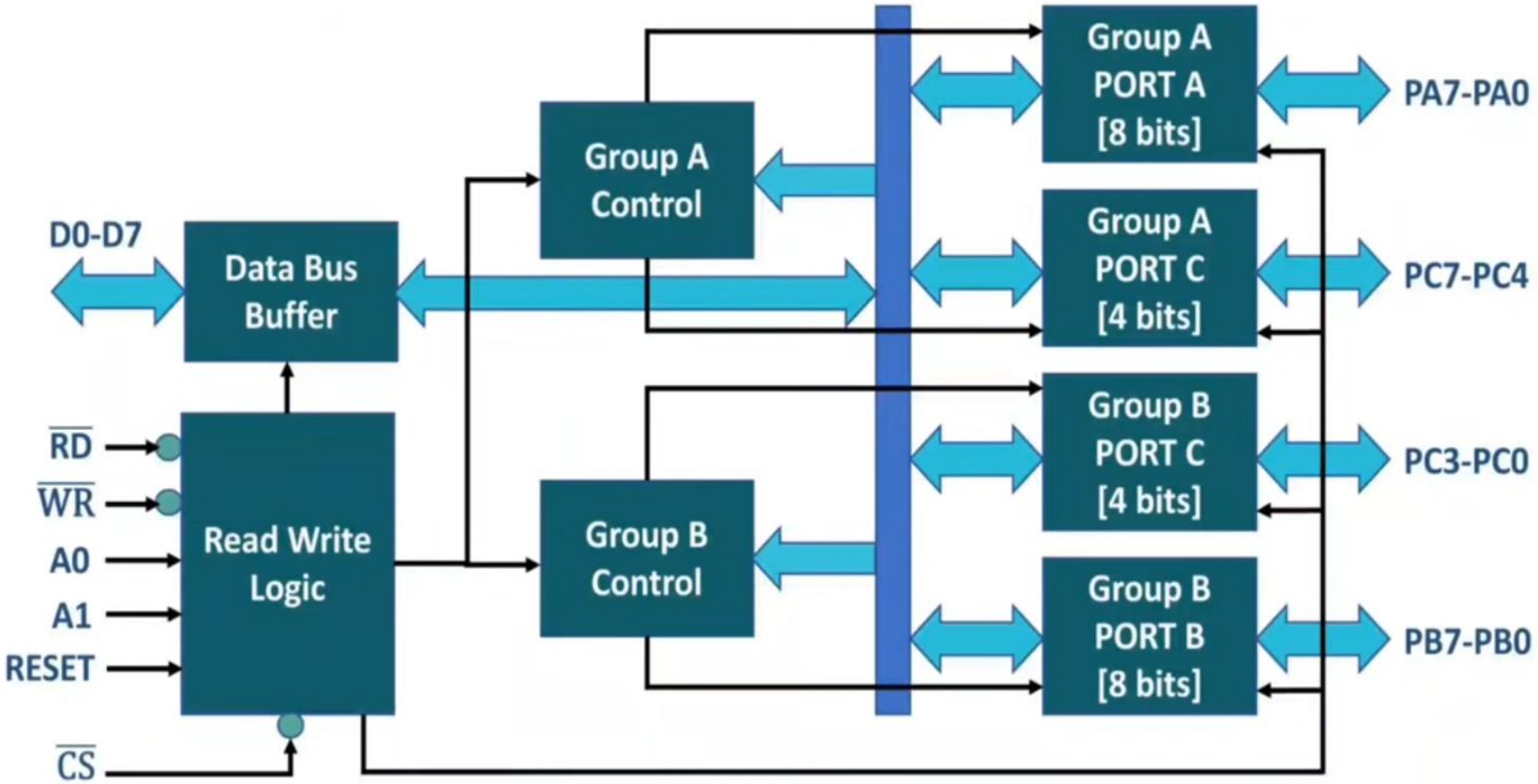
Group A & Group B Control of 8255



❖ Group A and Group B Control

- ❑ Group A control is used to control PORT A [PA7 – PA0] and UPPER PORT C [PC7 – PC4].
- ❑ Group B control is used to control PORT B [PB7 – PB0] and LOWER PORT C [PC3 – PC0].
- ❑ It takes control signals from control word and forwards it on respective ports.

Ports of 8255



❖ PORT A, PORT B and PORT C

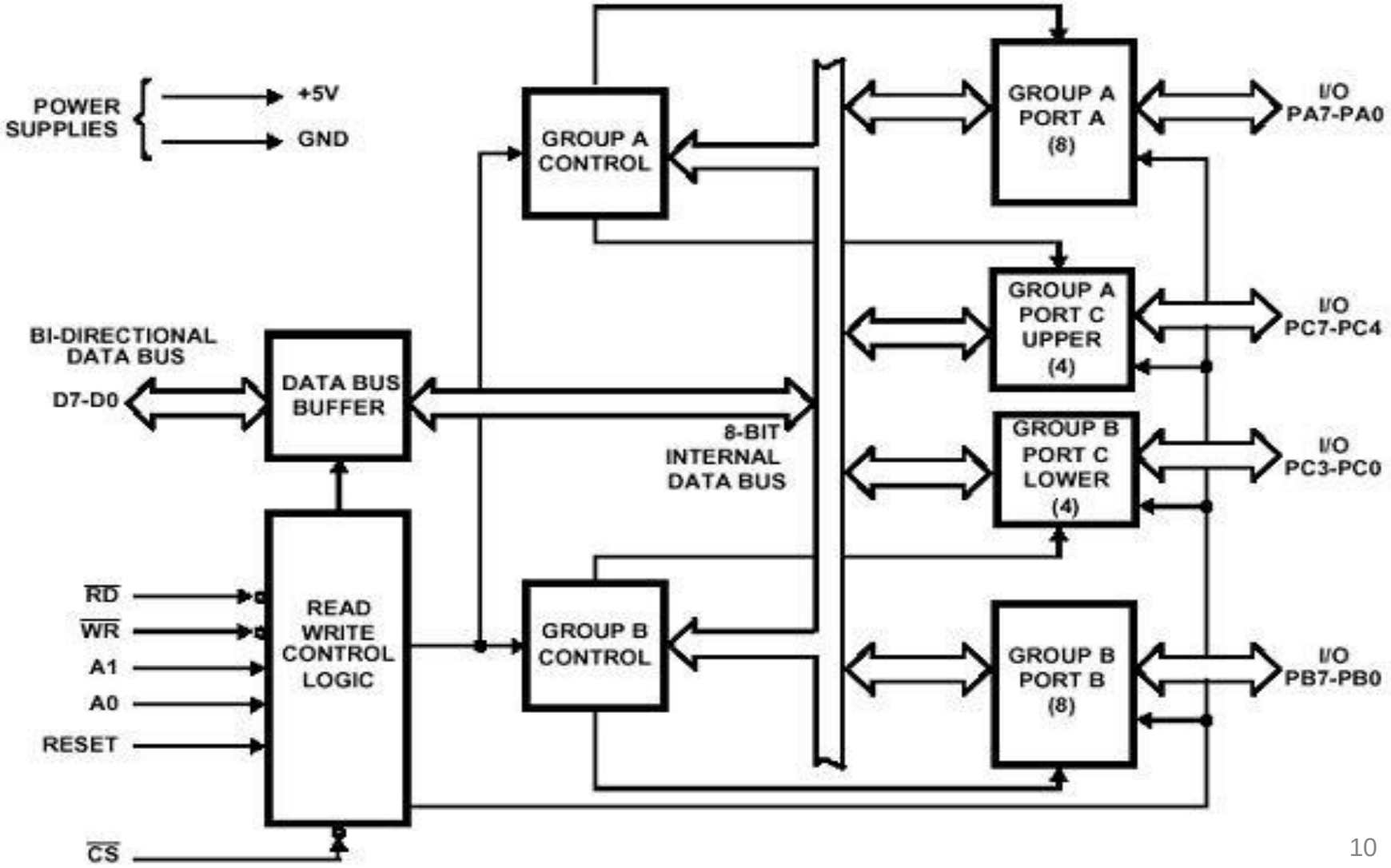
❑ These are 8 bits IO ports and works as follows:

Port	MODE 0	MODE 1	MODE 2	BSR MODE
PORT A	YES	YES	YES	NO
PORT B	YES	YES	NO	NO
PORT C	YES	NO [HS]	NO [HS]	YES

Ports of 8255A

- 8255A has three ports, i.e., PORT A, PORT B, and PORT C.
  - **Port A** contains one 8-bit output latch/buffer and one 8-bit input buffer.
  - **Port B** is similar to PORT A.
  - **Port C** can be split into two parts, i.e. PORT C lower (PC0-PC3) and PORT C upper (PC7-PC4) by the control word.
- These three ports are further divided into two groups, i.e. Group A includes PORT A and upper PORT C. Group B includes PORT B and lower PORT C. These two groups can be programmed in three different modes, i.e. the first mode is named as mode 0, the second mode is named as Mode 1 and the third mode is named as Mode 2.

Block diagram of 8255



Different Mode of Operation of 8255

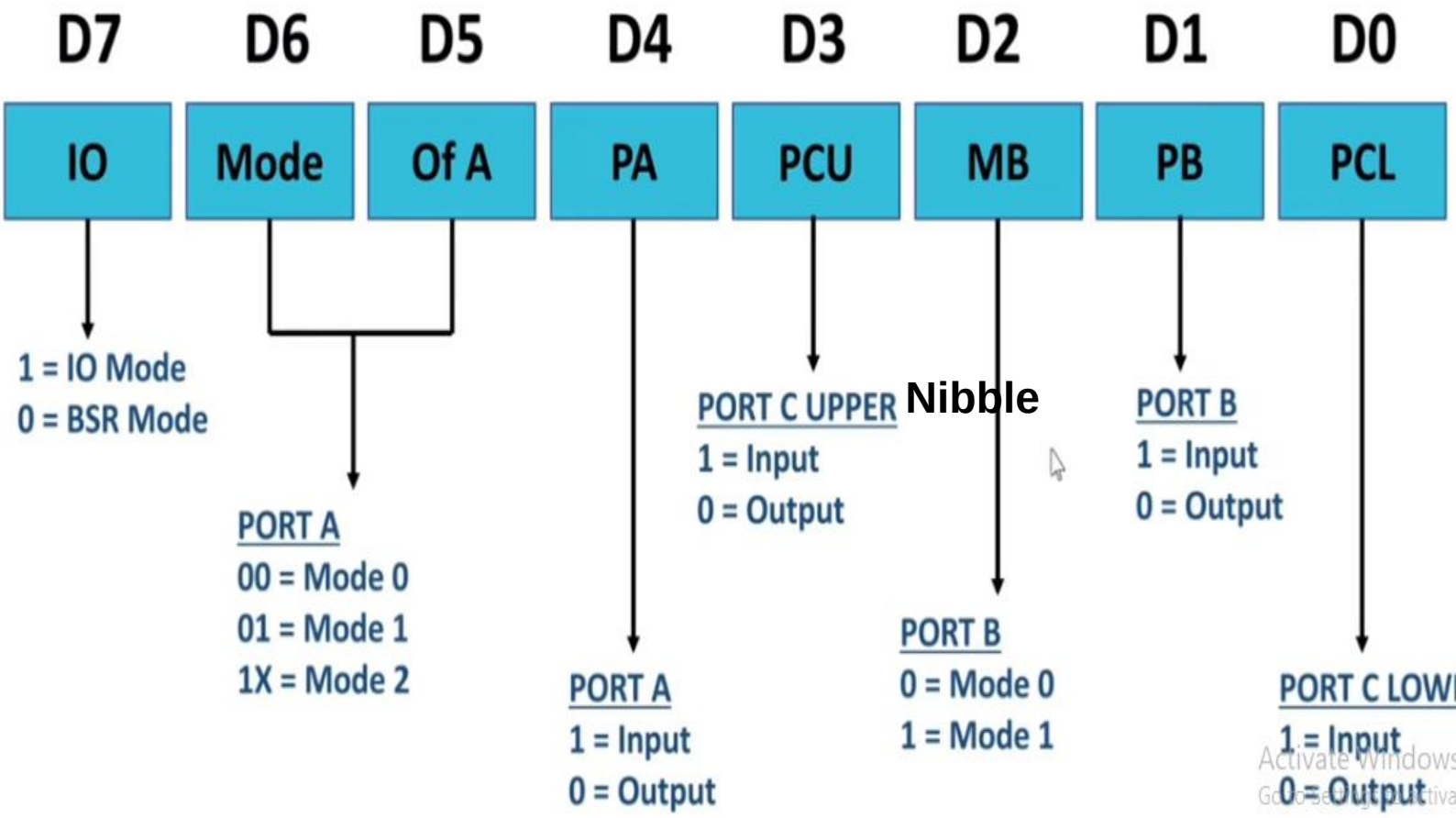
8255A has three different operating modes −

- **Mode 0** − In this mode, Port A and B is used as two 8-bit ports and Port C as two 4-bit ports. Each port can be programmed in either input mode or output mode where outputs are latched and inputs are not latched. Ports do not have interrupt capability.
- **Mode 1** − In this mode, Port A and B is used as 8-bit I/O ports. They can be configured as either input or output ports. Each port uses three lines from port C as **handshake** signals. Inputs and outputs are latched.
- **Mode 2** − In this mode, Port A can be configured as the bidirectional port and Port B either in Mode 0 or Mode 1. Port A uses five signals from Port C as **handshake** signals for data transfer. The remaining three signals from Port C can be used either as simple I/O or as handshake for port B.

Control Word and Modes of 8255

❖ Control Word

❑ 8255 has 8 bits of control word. It defines working of IO ports A, B and C.



Control Word and Modes of 8255

❖ BSR [Bit Set Reset] Mode of 8255

❑ BSR Mode only works with PORT C.

1 = IO Mode
0 = BSR Mode

BIT Select			Bit
0	0	0	PC0
0	0	1	PC1
0	1	0	PC2
0	1	1	PC3
1	0	0	PC4
1	0	1	PC5
1	1	0	PC6
1	1	1	PC7

1 = Set bit of Port C
0 = Reset bit of Port C

D1, D2, D3 Is used to select individual terminal of port-C

If the Terminal (PC0-PC7) values is 0/1 is selected by D0

Control Word and Modes of 8255

❖ Modes of 8255

❑ There are three 8 bits IO ports and works as follows:

Port	MODE 0	MODE 1	MODE 2	BSR MODE
PORT A	YES	YES	YES	NO
PORT B	YES	YES	NO	NO
PORT C	YES	NO [HS]	NO [HS]	YES

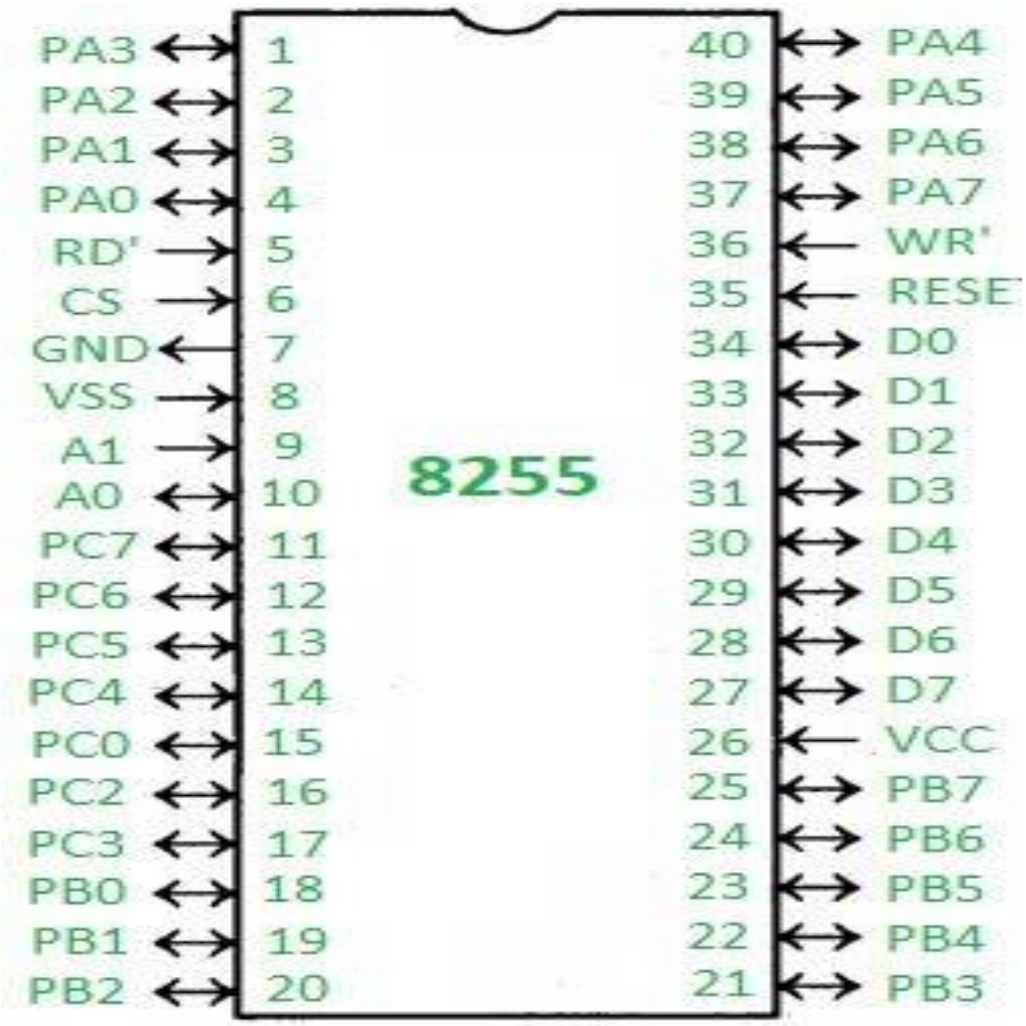
Works in handshaking mode

❑ Here, Mode 0 is simple IO mode.
- Output are latched and Input are not latched.
- Port Do not have Interrupt handling capacity.

❑ Here, Mode 1 is IO mode with handshake.
- Here each port uses three lines from port C as Handshake signals.
- Here, Input and Output are latched.
- Interrupt handling is supported.

❑ Here, Mode 2 is Bidirectional IO mode with handshake.
- Here port A uses five lines from port C as Handshake signals.
- Interrupt handling is supported.

Pin Diagram of 8255



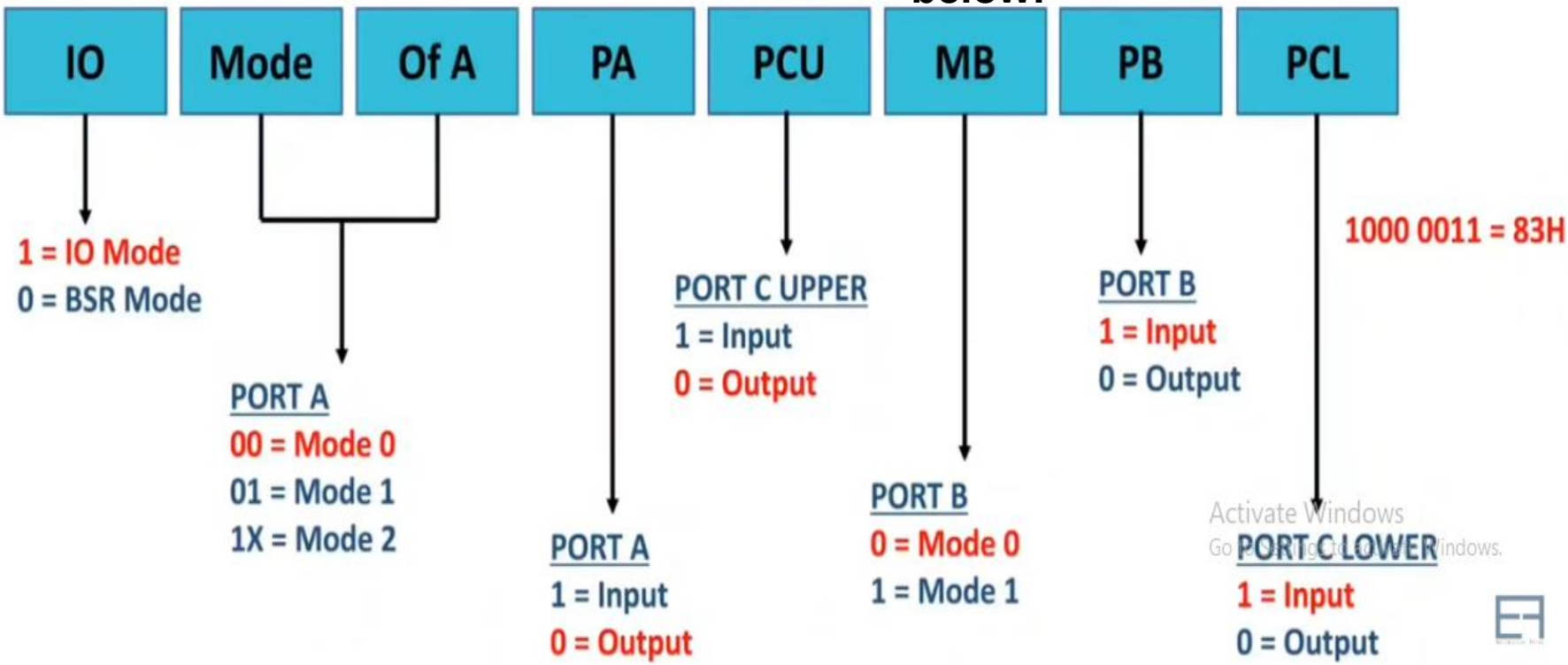
- **PA0 – PA7 –** Pins of port A
- **PB0 – PB7 –** Pins of port B
- **PC0 – PC7 –** Pins of port C
- **D0 – D7 –** Data pins for the transfer of data
- **RESET –** Reset input
- **RD' –** Read input
- **WR' –** Write input
- **CS' –** Chip select
- **A1 and A0 –** Address pins

Programming of 8255

Problem

❖ Identify the port Address, Identify the Mode 0 control word to configure port A & Upper port C as output port and port B & Lower port C as Input port. Write a program to read DIP from Input and display it with LED on Output.

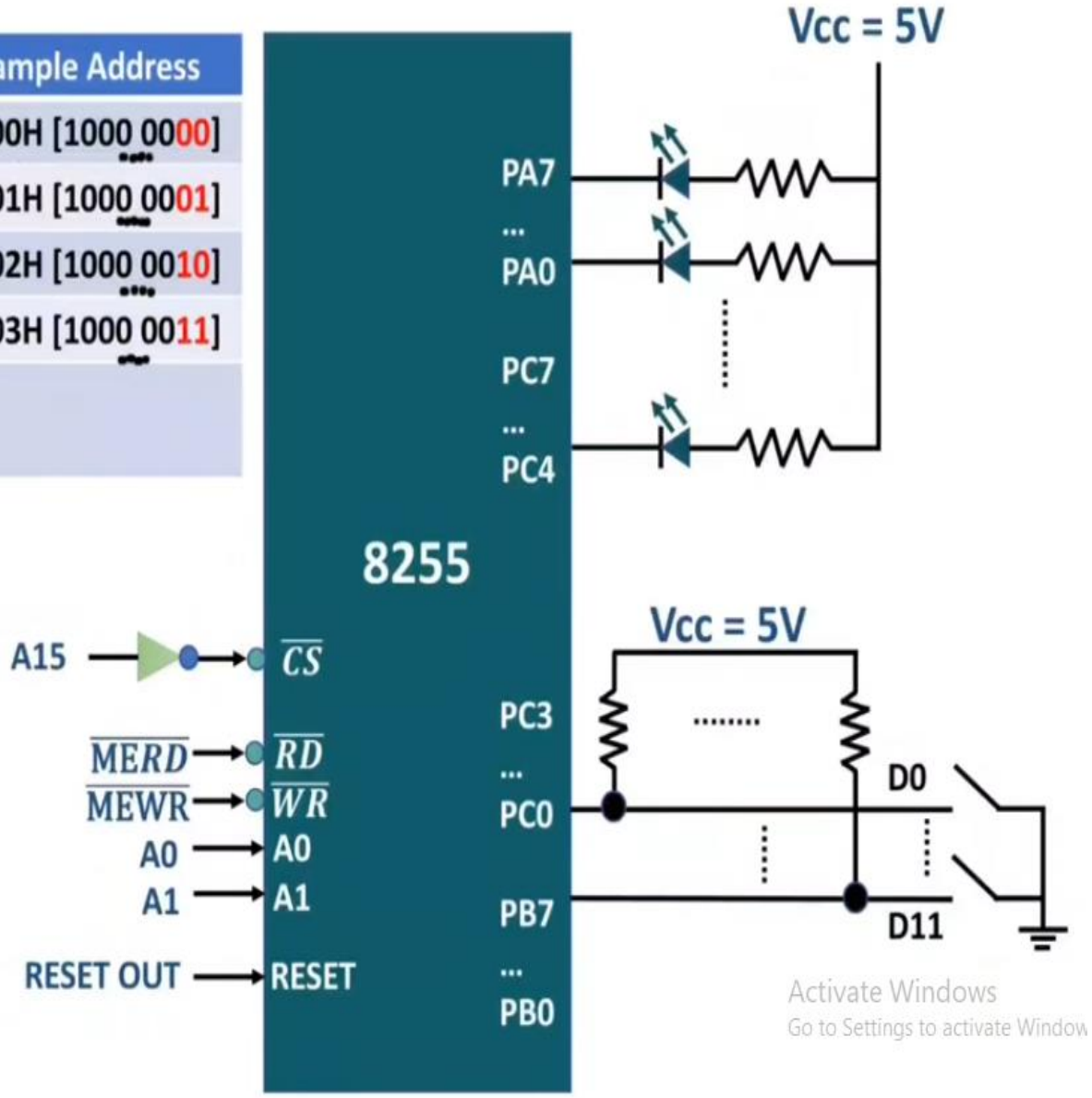
Control word for the problem below:



Interfacing, and Addressing Details

A15	A1	A0	Selected	Sample Address
1	0	0	Port A	8000H [1000 0000]
1	0	1	Port B	8001H [1000 0001]
1	1	0	Port C	8002H [1000 0010]
1	1	1	Control Register	8003H [1000 0011]
0	X	X	8255 is not selected	

```
XX:   MVI A,83H
      STA 8003H
      LDA 8001H
      STA 8000H
      LDA 8002H
      ANI 0FH
      RLC
      RLC
      RLC
      RLC
      STA 8002H
      JMP XX
```

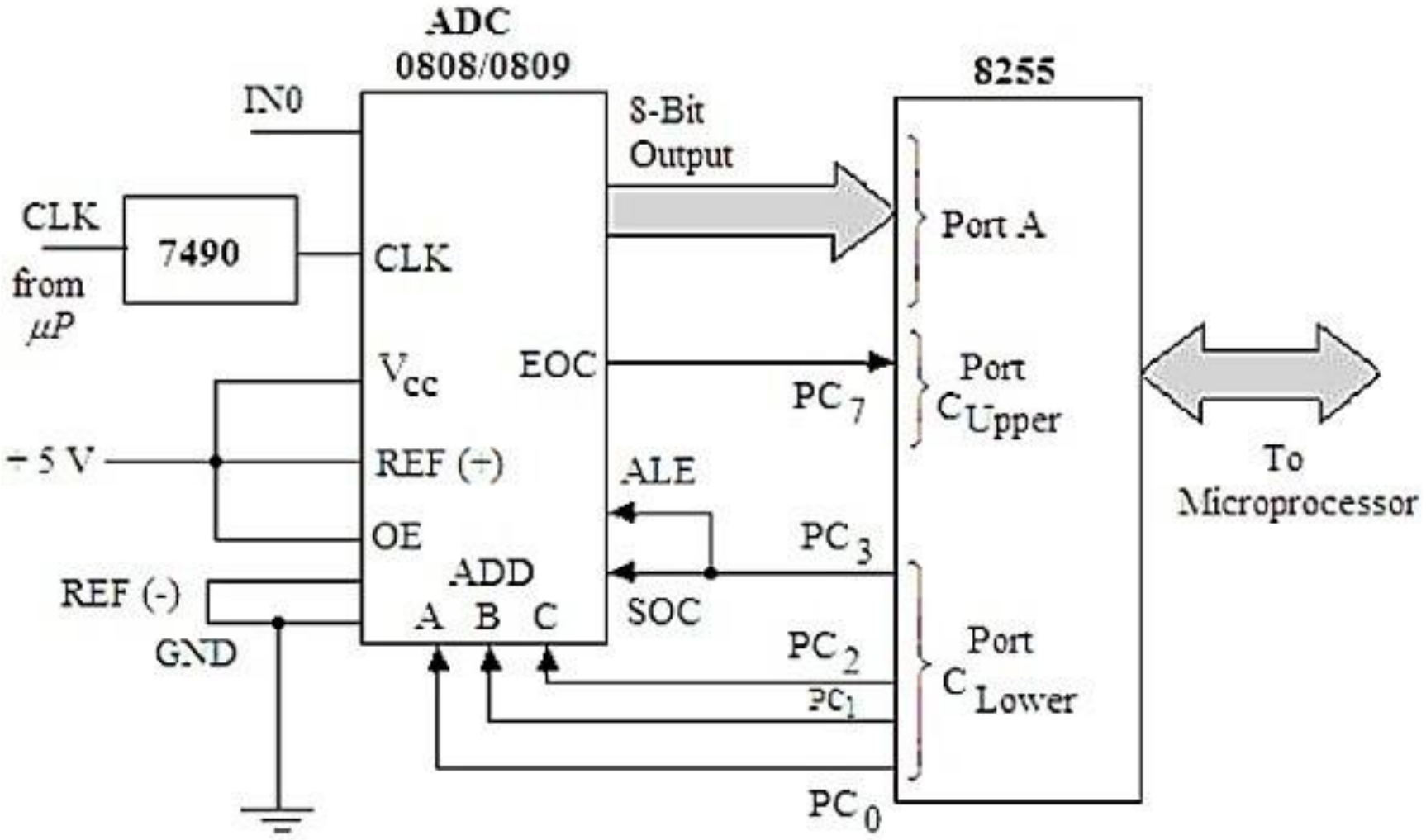


MVI=Move immediately ANI=Immediate addressing RLC=Rotate Left through Carry (Convert lower to upper nibble)

Interfacing A/D converter using 8255

- To interface the ADC with 8085, we need 8255 Programmable Peripheral Interface chip with it.
- The Port A of 8255 chip is used as the input port. The PC_7 pin of Port C_{upper} is connected to the End of Conversion (EOC) Pin of the analog to digital converter. This port is also used as input port. The C_{lower} port is used as output port. The PC_{2-0} lines are connected to three address pins of this chip to select input channels. The PC_3 pin is connected to the Start of Conversion (SOC) pin and ALE pin of ADC 0808/0809.

Interfacing A/D converter using 8255 (Cont.)



8253/54 Programmable Interval Timer

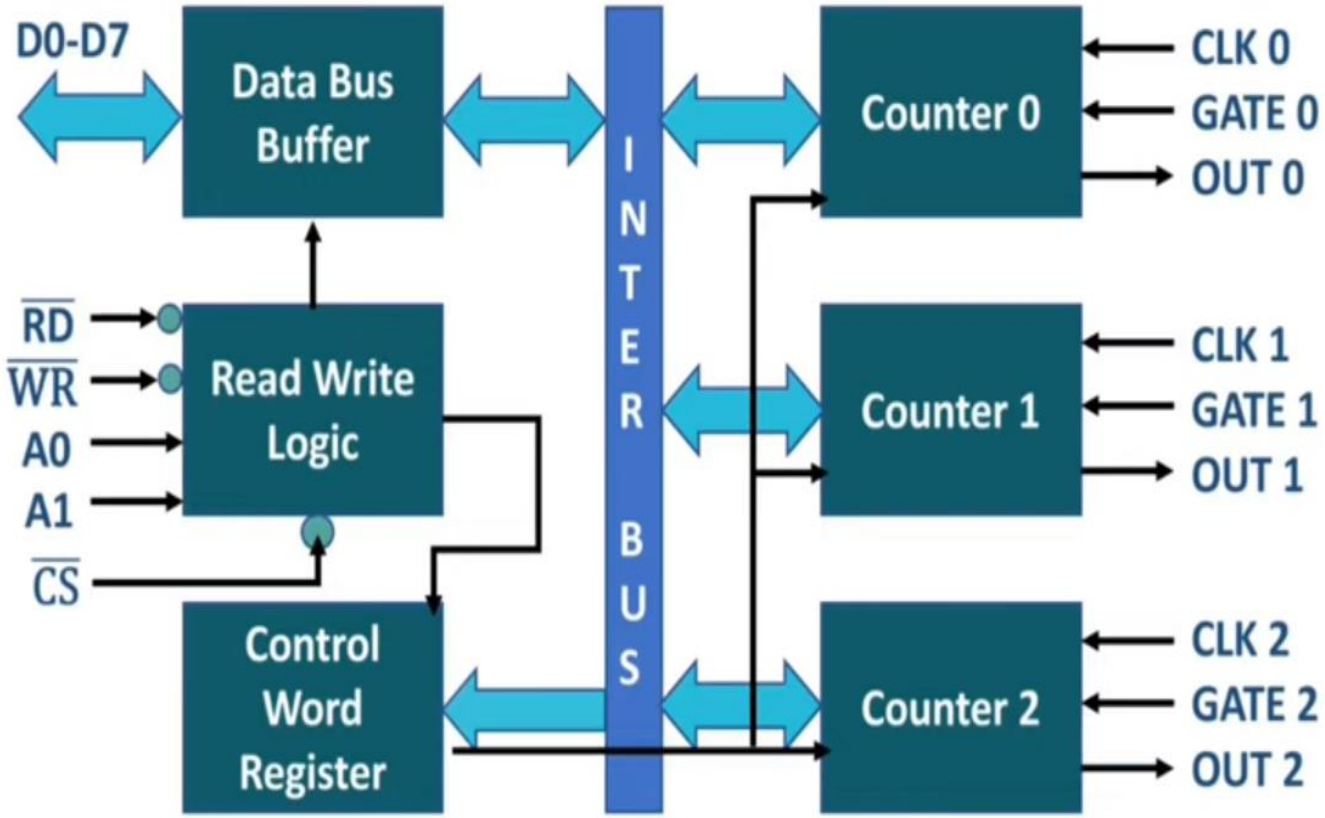
❖ Features of 8253/54 Programmable Interval Timer

- ❑ 8254 is designed to work with various microprocessors like 8085, 8086 etc.
- ❑ 8254 is used as Timer to generate Hardware Delay.
- ❑ 8254 can be used as real time clock or as square wave generator.
- ❑ Hardware delay is more useful then software delay as microprocessor is not actively involved in generating delay. So when delay is produced by 8254 at that time microprocessor is free do to execute other programs.
- ❑ 8254 has three independent 16 bits Down counters.
- ❑ These counters can take count in BCD or in Binary.
- ❑ Once counters finishes count [required delay], 8254 interrupts Microprocessor.

8253 Programmable Interval Timer

- The Intel 8253 is Programmable Interval Timers (PTIs) designed for microprocessors to perform timing and counting functions using three 16-bit registers.
- Each counter has 2 input pins, i.e. Clock & Gate, and 1 pin for “OUT” output.
- To operate a counter, a 16-bit count is loaded in its register.
- On command, it begins to decrement the count until it reaches 0, then it generates a pulse that can be used to interrupt the CPU.

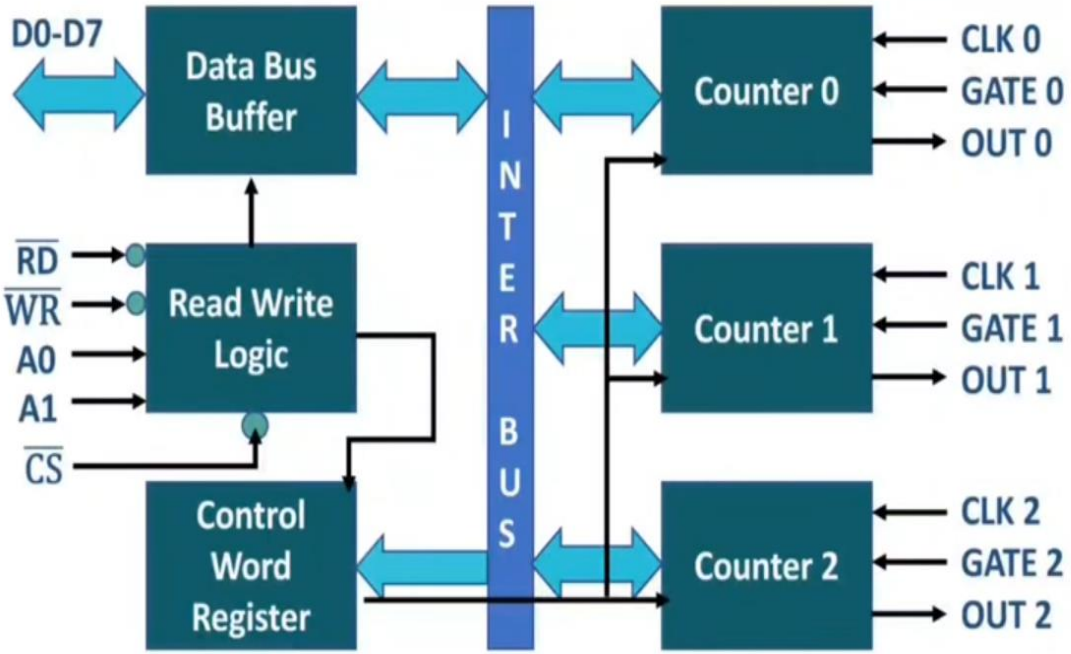
Block Diagram of 8253/54 Programmable Interval Timer



❖ Data Bus Buffer

- ❑ It has bidirectional data bus D0 – D7.
- ❑ D0 – D7 is interfaced with system data bus of Microprocessor.

Read Write logic for 8254



❖ Read Write Logic

- ❑ $\overline{RD}$ and $\overline{WR}$ is used to read write on D0 – D7.
- ❑ A1 and A0 lines are used to select counter and control word.
- ❑ $\overline{CS}$ will select chip of 8254.

$\overline{CS}$	A1	A0	Selected	Sample Address
0	0	0	Counter 0	80H [1000 0000]
0	0	1	Counter 1	81H [1000 0001]
0	1	0	Counter 2	82H [1000 0010]
0	1	1	Control Word	83H [1000 0011]
1	X	X	8254 is not selected	

Control Word and Modes of 8254

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

Select Counter [SC1 – SC0]

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

Read Write [RW1 – RW0]

RW1	RW0	Operation
0	0	Counter Latch command
0	1	R/W Least Byte significant Only
1	0	R/W Most Byte significant Only
1	1	R/W least Byte significant 1st then most significant Byte.

Mode selection [M2, M1 & M0]

Mode Select			Bit
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

Select Counter [SC1 – SC0]

BCD	Selected
0	Binary Counter 16 bits
1	BCD Counter of 4 Decades

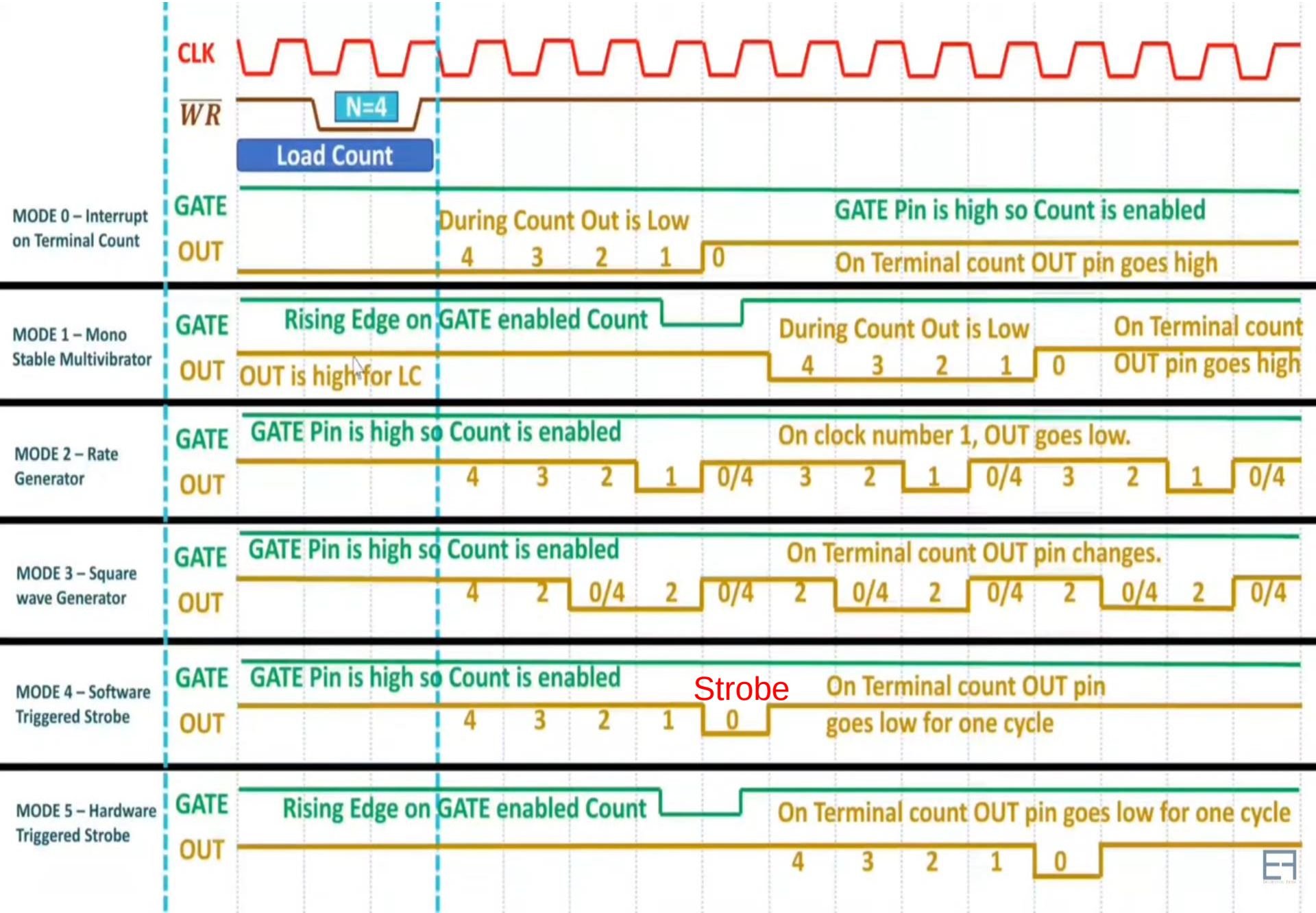
Modes of 8253/54 Programmable Interval Timer

- Modes of 8253/54 is selected by Control word with M0, M1 & M2 bits.
  - There are six modes & three counters with 8254.
  - All modes works with clock input.
  - First we need to load value of count in 8254 by 8085.
  - After every clock count works in auto decrement mode.

Mode selection [M2, M1 & M0]

Mode Select			Bit	Name of Mode
0	0	0	Mode 0	Interrupt on terminal count
0	0	1	Mode 1	Monostable Multivibrator
X	1	0	Mode 2	Rate Generator
X	1	1	Mode 3	Square Wave Generator
1	0	0	Mode 4	Software Trigger Strobe
1	0	1	Mode 5	Hardware Trigger Strobe

All MODES of 8254



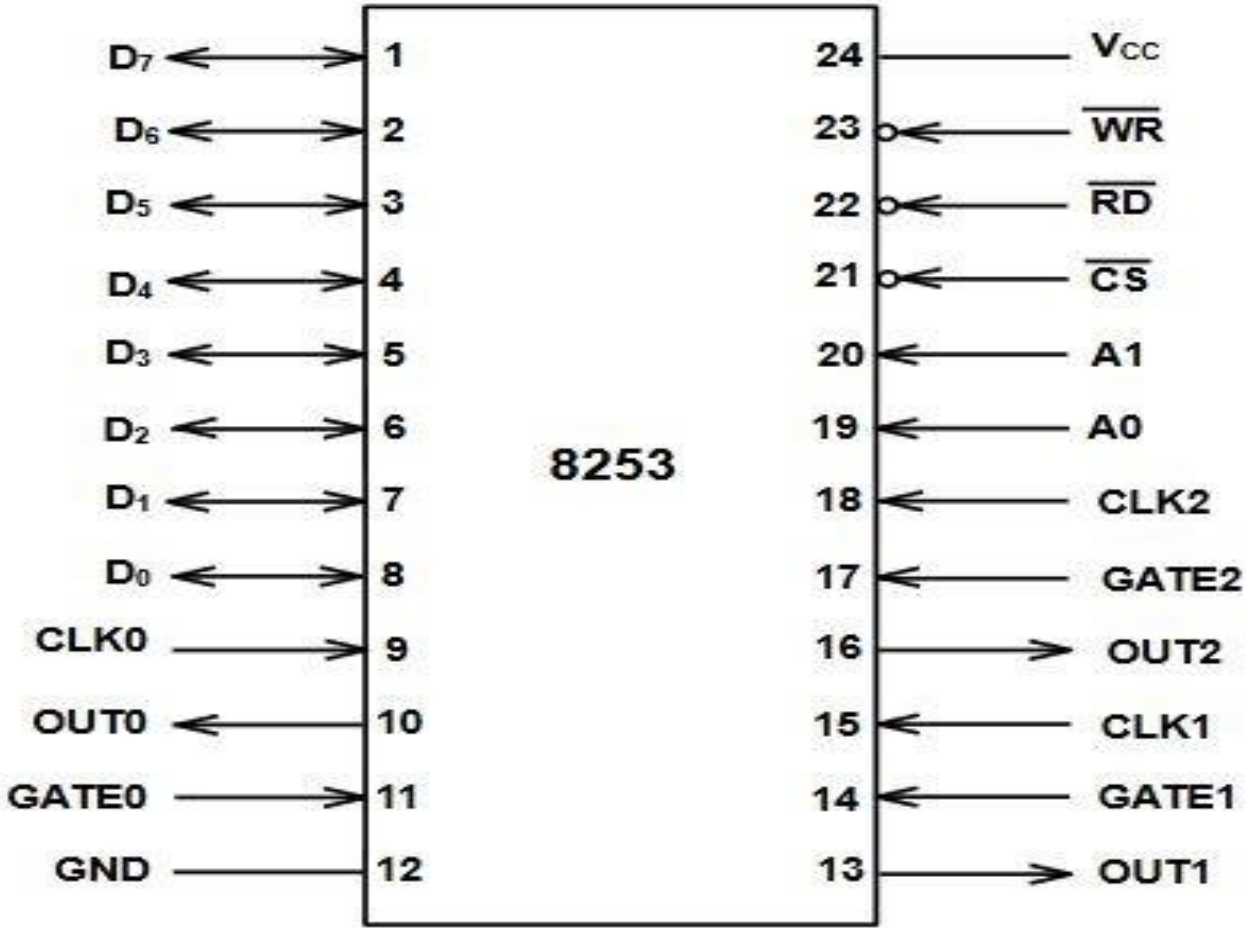


Fig.10.1 Pin Configuration of Intel 8253

Speed control and direction of the DC motor using 8254 timer

DC Motor Speed and Direction Control

One application of the 8254 timer is as a motor speed controller for a DC motor. Figure 11–45 shows the schematic diagram of the motor and its associated driver circuitry. It also illustrates the interconnection of the 8254, a flip-flop, and the motor and its driver.

The operation of the motor driver circuitry is straightforward. If the Q output of the 74ALS112 is a logic 1, the base Q2 is pulled up to +12 V through the base pull-up resistor, and the base of Q2 is open circuited. This means that Q1 is off and Q2 is on, with ground applied to the positive lead of the motor. The bases of both Q3 and Q4 are pulled low to ground through the inverters. This causes Q3 to conduction or turn on and Q4 to turn off, applying ground to the negative lead of the motor. The logic 1 at the Q output of the flip-flop therefore connects +12 V to the positive lead of the motor and ground to the negative lead. This connection causes the motor to spin in its forward direction. If the state of the Q output of the flip-flop becomes a logic 0, then the conditions of the transistors are reversed and +12 V is attached to the negative lead of the motor, with ground attached to the positive lead. This causes the motor to spin in the reverse direction.

If the output of the flip-flop is alternated between a logic 1 and 0, the motor spins in either direction at various speeds. If the duty cycle of the Q output is 50 percent, the motor will not spin at all and exhibits some holding torque because current flows through it. Figure 11–46 shows some timing diagrams and their effects on the speed and direction of the motor. Notice how each counter generates pulses at different positions to vary the duty cycle at the Q output of the flip-flop. This output is also called *pulse width modulation*.

To generate these wave forms, counters 0 and 1 are both programmed to divide the input clock (PCLK) by 30,720. We change the duty cycle of Q by changing the point at which counter 1 is started in relationship to counter 0. This changes the direction and speed of the motor. But why divide the 8 MHz clock by 30,720? The divide rate of 30,720 is divisible by 256, so we can develop a short program that allows 256 different speeds. This also produces a basic operating frequency for the motor of about 260 Hz, which is low enough in frequency to power the motor. It is important to keep this operating frequency below 1000 Hz, but above 60 Hz.

Example 11–27 lists a procedure that controls the speed and direction of the motor. The speed is controlled by the value of AH when this procedure is called. Because we have an 8-bit number to represent speed, a 50 percent duty cycle, for a stopped motor, is a count of 128. By changing the value in AH when the procedure is called, we can adjust the motor speed. The speed of the motor will increase in either direction by changing the number in AH when this procedure is called. As the value in AH approaches 00H, the motor begins to increase its speed in the reverse direction. As the value of AH approached FFH, the motor increases its speed in the forward direction.

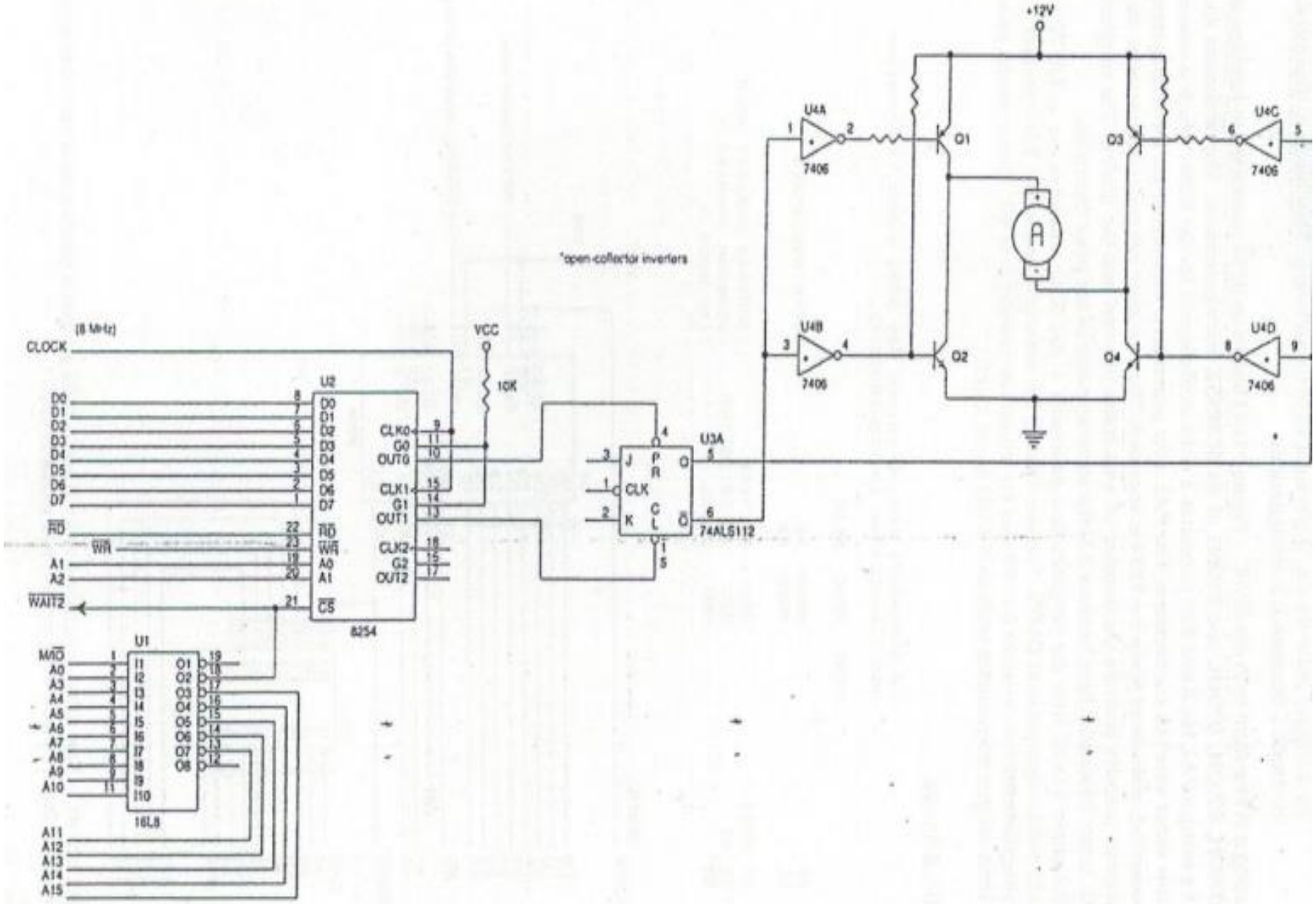


FIGURE 11–45 Motor speed and direction control using the 8254 timer.

8259 Programmable Interrupt Controller

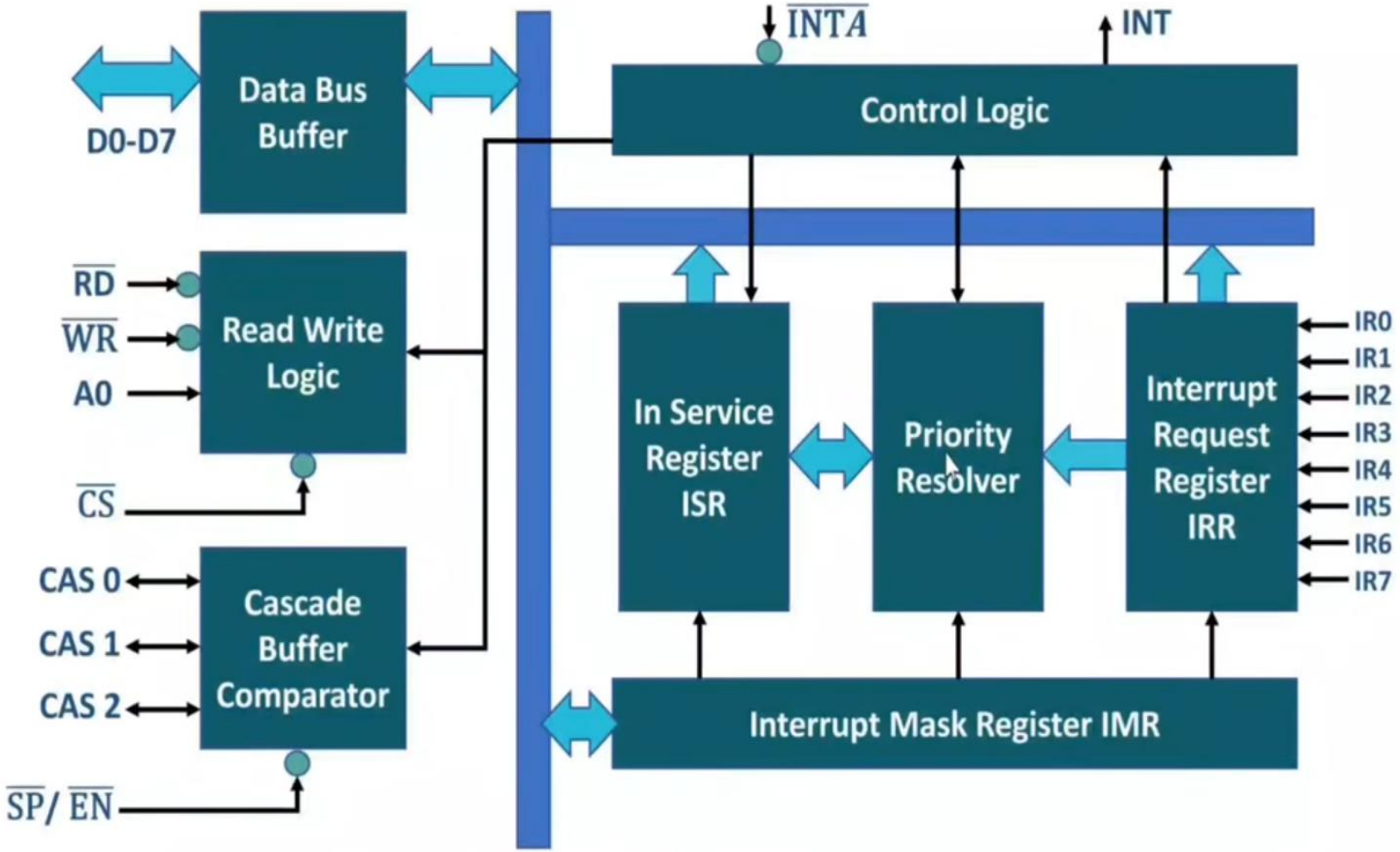
❖ Features of 8259 Programmable Interrupt Controller

- ❑ 8259 is designed to work with various microprocessors like 8085, 8086 etc.
- ❑ 8259 is designed to increase capacity of interrupts.
- ❑ 8259 can handle 8 interrupts with single IC.
- ❑ A cascade connection of 8259 can handle 64 interrupts.
  - ▪ One Master 8259 can works with eight Slave 8259, so total capacity will be 64.
- ❑ 8259 has flexible interrupt priority structure.
- ❑ Using 8259, we can mask interrupt as well.
- ❑ The vector address of 8259 is programmable.
- ❑ Status of interrupt can be observed by microprocessor :
  - ▪ Pending status
  - ▪ In service status
  - ▪ Masked status

8259 Programmable Interrupt Controller

- The 8259A is a programmable interrupt controller specially designed to work with Intel microprocessor 8080, 8085A, 8086, 8088.

Block diagram of 8259



❖ Interrupt Request Register - IRR

- ❑ Here, we have 8 interrupt lines IR7 – IR0.
- ❑ IRR is Eight bits register, each bit stores individual interrupt.
- ❑ When interrupt occurs on any lines, corresponding bit will get set to One.

Block diagram of 8259

In Service Register - ISR

- It is 8 bits register.
- It stores the data of currently served interrupt.

Interrupt Mask Register - IMR

- It is 8 bits register.
- It stores the masking pattern of 8259.
- Each bits holds masking of individual interrupt.

Priority Resolver

- It examines IRR, ISR and IMR. Based on that determines which interrupt has maximum priority and should be sent to microprocessor.

Control Logic

- It has INT pin connected with INTR of microprocessor to send interrupt request.
- It has $\overline{\text{INTA}}$ pin connected with $\overline{\text{INTA}}$ of microprocessor to receive acknowledgment.
- It is also used to control remaining blocks.

Block diagram of 8259

❖ Data Bus Buffer

- ❑ It has bidirectional data bus D0 – D7.
- ❑ D0 – D7 is interfaced with system data bus of Microprocessor.

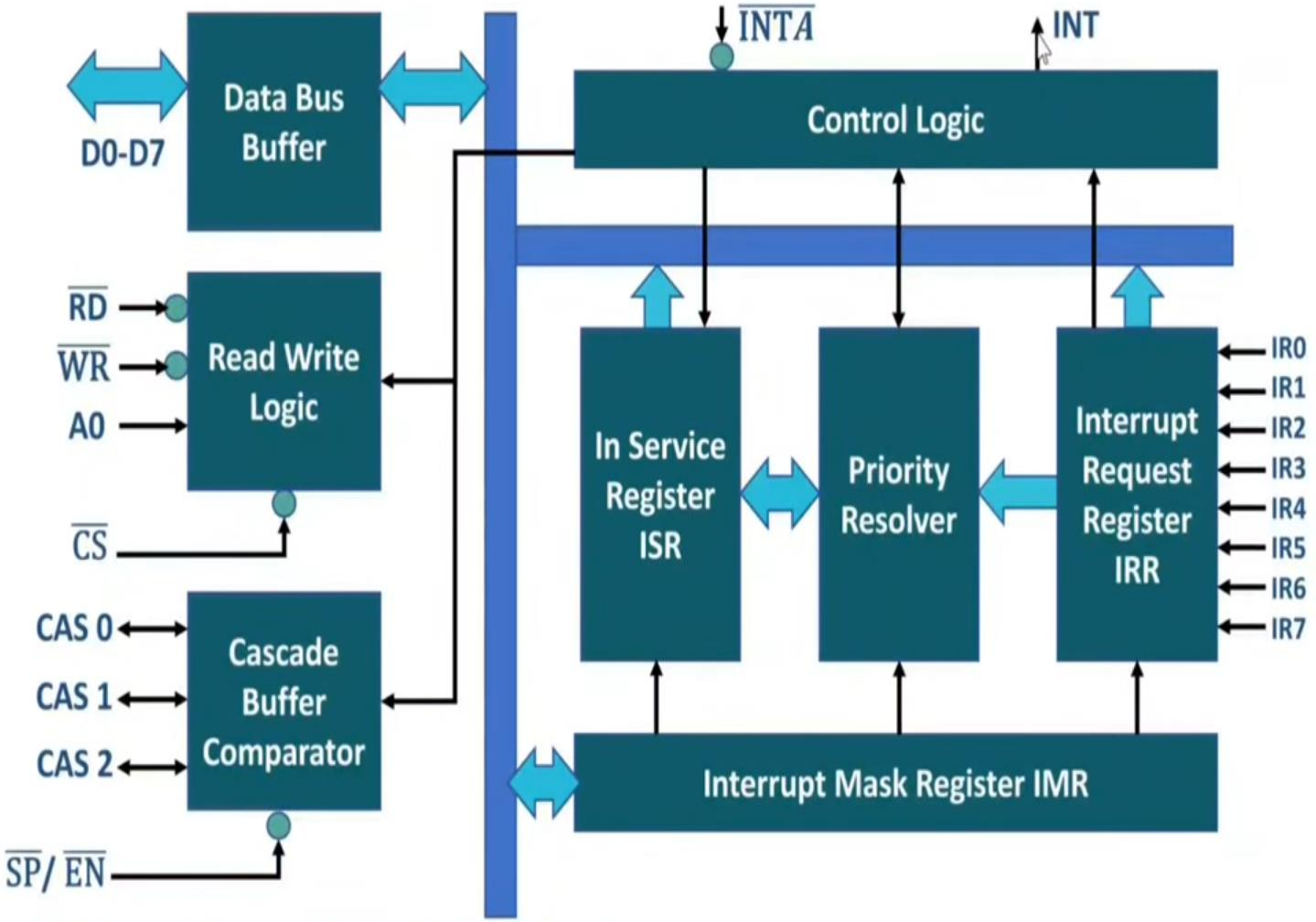
❖ Read Write Logic

- ❑ It is used to take read, write, A0 and Chip select.
- ❑ It also holds Initialization Command Words (ICW) and Operational Command Words (OCW).

❖ Cascade Buffer Comparator

- ❑ It is used in cascade mode operation.
- ❑ It is used for Master Slave Interrupt control from multiple 8259.
- ❑ $\overline{SP}/\overline{EN}$ – It is Slave Program/ Master Enable line. In buffer Mode, it function as enable line and In non buffer mode it functions as SP enable line.

Working of 8259 PPI



1. INTR of 8085 must be enabled with EI Instruction.
2. 8259 is initialized by necessary commands. [ICW]
3. Once 8259 is initialized, if multiple interrupts comes then corresponding bit of IRR is set to one.
4. Priority resolver checks IRR, IMR & ISR. Based on that, it will decide highest priority and then it gives signal to Microprocessor 8085 at INTR.

Working of 8259 PPI

5. The Microprocessor completes current instruction, after that it give acknowledgement to 8259 on $\overline{\text{INTA}}$.
6. On receiving $\overline{\text{INTA}}$ from Microprocessor, ISR will set corresponding bit to one in ISR to indicate service to this interrupt is started and the bit in IRR is reset to indicate request is accepted. Now 8259 can give opcode of CALL instruction to Microprocessor.

7. The microprocessor decodes the CALL instruction and sends two more $\overline{\text{INTA}}$ to 8259.
8. In response to $\overline{\text{INTA}}$ signals, 8059 sends address of interrupt service routine. So it completes three bytes CALL instruction.
9. Now Microprocessor perform Interrupt Service Routine by pushing content of PC on stack.
10. At the end of interrupt, Microprocessor will send EOI command to 8259, that makes corresponding bit 0 in ISR of 8259.

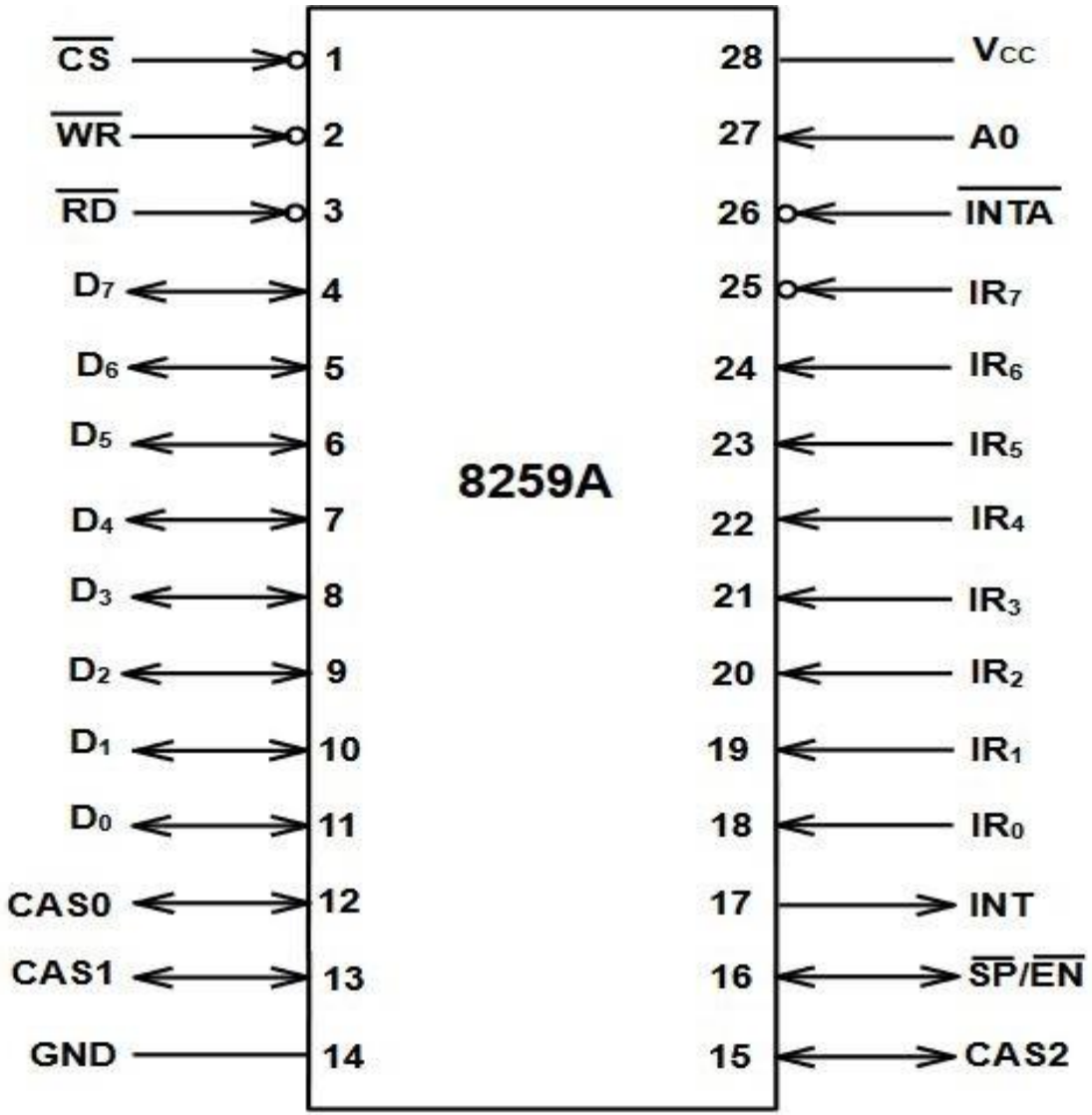


Fig.11.2 Pin Configuration of Intel 8259A

Modes of 8259A PIC

- Fully Nested mode
- Special Fully Nested mode
- Nonspecific Rotating
- Specific Rotating
- Special Mask
- Polling
- Fixed priority mode

- Fully nested mode:
  - This is a general purpose mode where all IR's are arranged in highest to lowest.
  - IR0 highest and IR7 lowest.
- Special Fully Nested Mode:
  - Used in more complicated systems.
  - Similar to, normal nested mode.
  - When an interrupt request from a certain slave is in service, this slave can further send requests to the master.
  - The master interrupts the CPU only.
- Automatic Rotation Mode:
  - In this mode a device after being serviced receives the lowest priority.

- Specific Rotation Mode:
  - In this user can select any IR for lowest priority thus fixing all priorities.
- Special Mask Mode
  - When a mask bit is set in OCW, it inhibits further interrupts at that level and enables interrupt from other levels, which are not mastered.
- Poll command
  - The INT output is neglected, though it functions normally by not connecting INT output or by masking INT input of the microprocessor.
  - This mode is entered by setting p=1 in OCW3.
  - A poll command may give more than 64 priority levels.

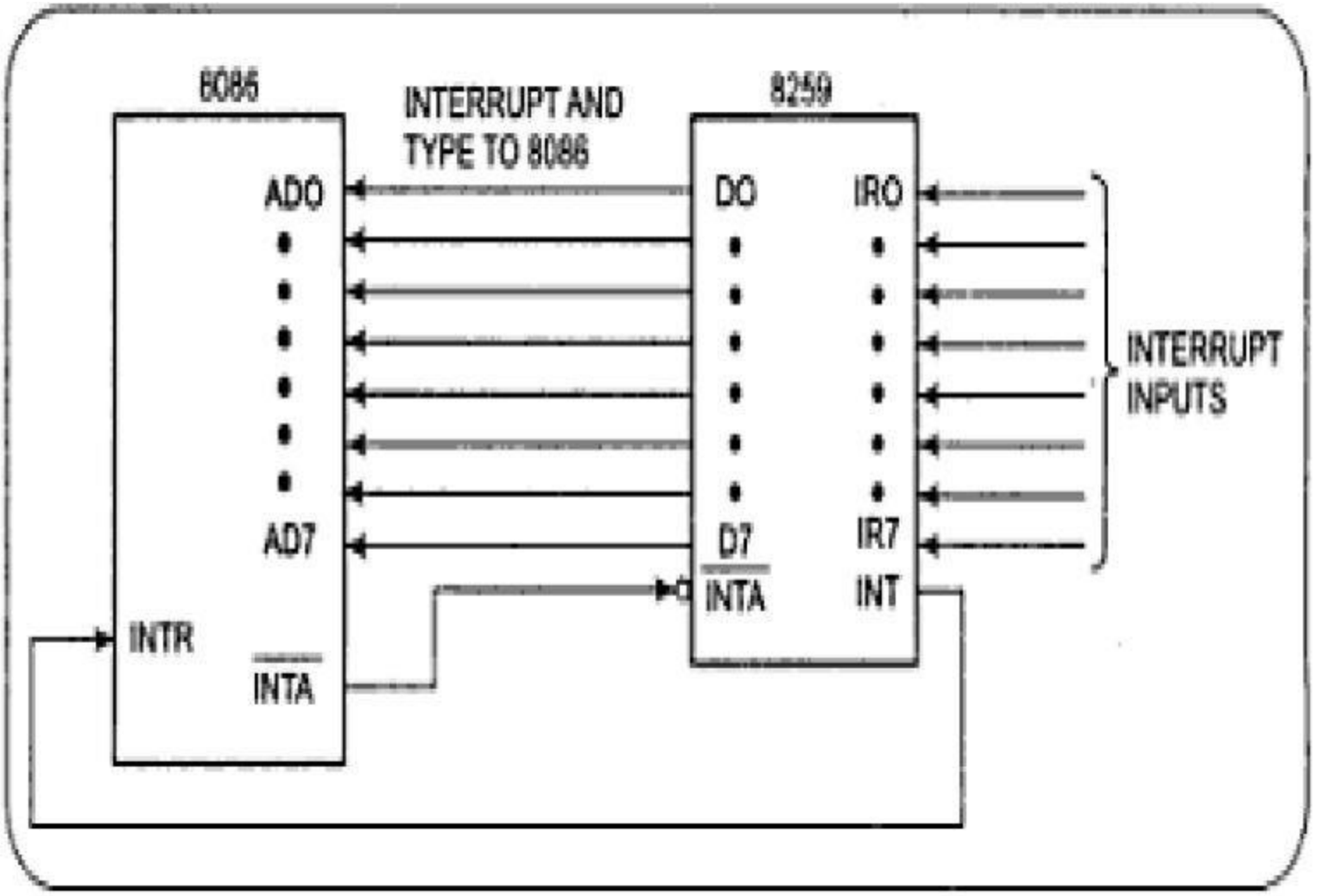
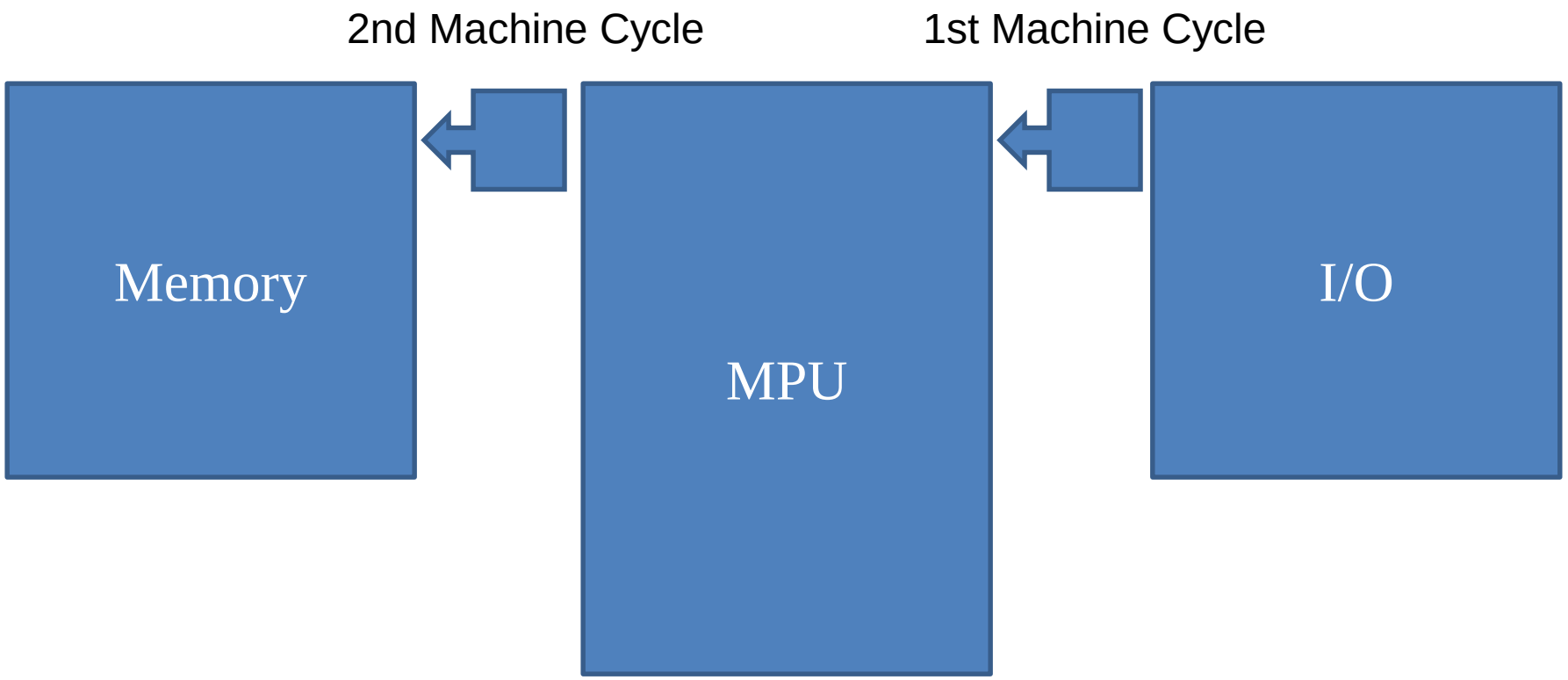


Fig:- Interface 8259 PIC with 8086 Microprocessor

8257 DMA Controller



In typical process data transfer rate slow down. DMA's roles is to bypass the microprocessor so that faster data exchanges happens between Memory and I/O

Basics 8257 DMA Controller

- DMA stands for Direct Memory Access. It is designed by Intel to transfer data at the fastest rate. It allows the device to transfer the data directly to/from memory without any interference of the CPU.
- Using a DMA controller, the device requests the CPU to hold its data, address and control bus, so the device is free to transfer data directly to/from the memory. The DMA data transfer is initiated only after receiving HLDA signal from the CPU.

8257 Direct Memory Access

❖ Basics of 8257 Direct Memory Access

- ❑ 8257 is used to for high speed data transfer in between in IO devices and memory.
- ❑ Using Microprocessor data transfer is slow, as microprocessor have to execute instructions and it have to check interrupt as well.
- ❑ Using 8257, MPU releases the control of the buses to the DMA.
- ❑ Here, Data transfer between memory and IO is been done by bypassing MPU.
- ❑ To take control of Address and Data bus from MPU, it has HOLD and HLDA control terminals which are used by DMA.

HLD & HLDA of 8257

HOLD and HLDA

HOLD :

- This is active high signal input MPU.
- It gives request to MPU for address and data bus control.
- After receiving HOLD signal, MPU relinquishes the buses in following machine cycle.
- All buses are tri stated and HLDA sent out by MPU.
- MPU regains the control of buses after HOLD goes low.

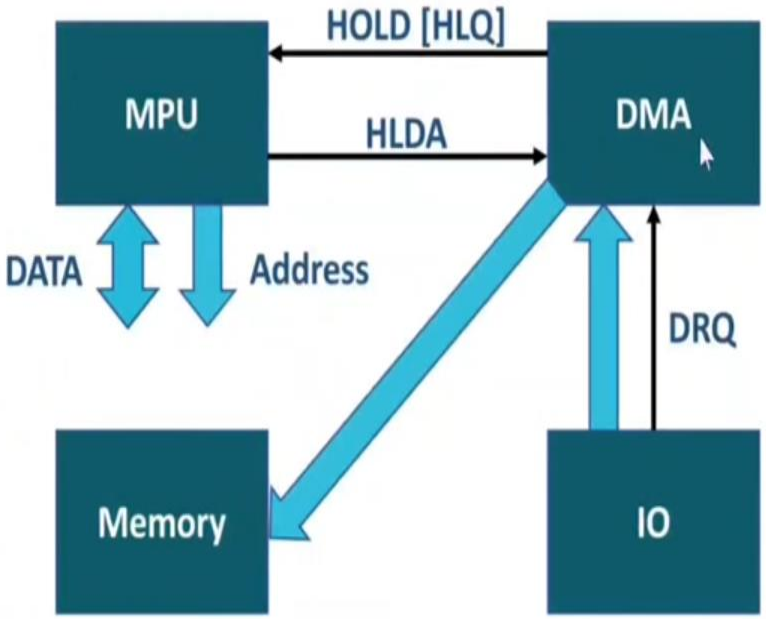
HLDA :

- This is active high signal indicating that the MPU is relinquishing control of buses.

Working of 8257

❖ Working of DMA

- If IO wants to send data to memory, then it will send request to DMA. [DRQ]
- To take control of system buses, DMA will send HOLD signal to MPU.
- To give control of address and data, MPU will give HLDA [HOLD Acknowledge]. Which indicates that now DMA is master of Buses. So now buses will be managed by DMA for memory and IO.

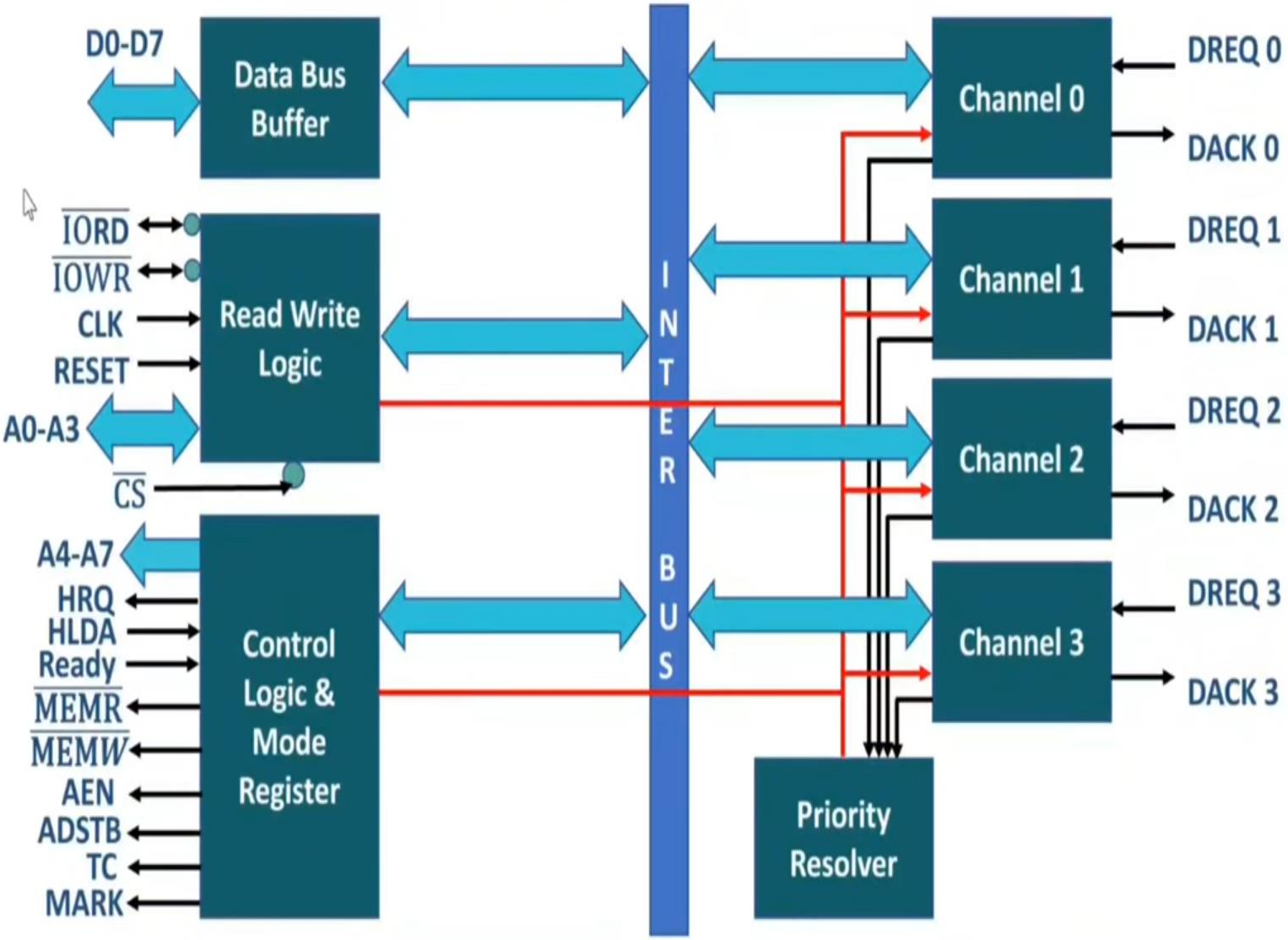


- Now data transfer can happen without involvement of MPU. Here now MPU don't need to execute instructions, so data exchange will be faster.
- Once HOLD signals goes low, MPU will take control of system buses and then MPU becomes master.

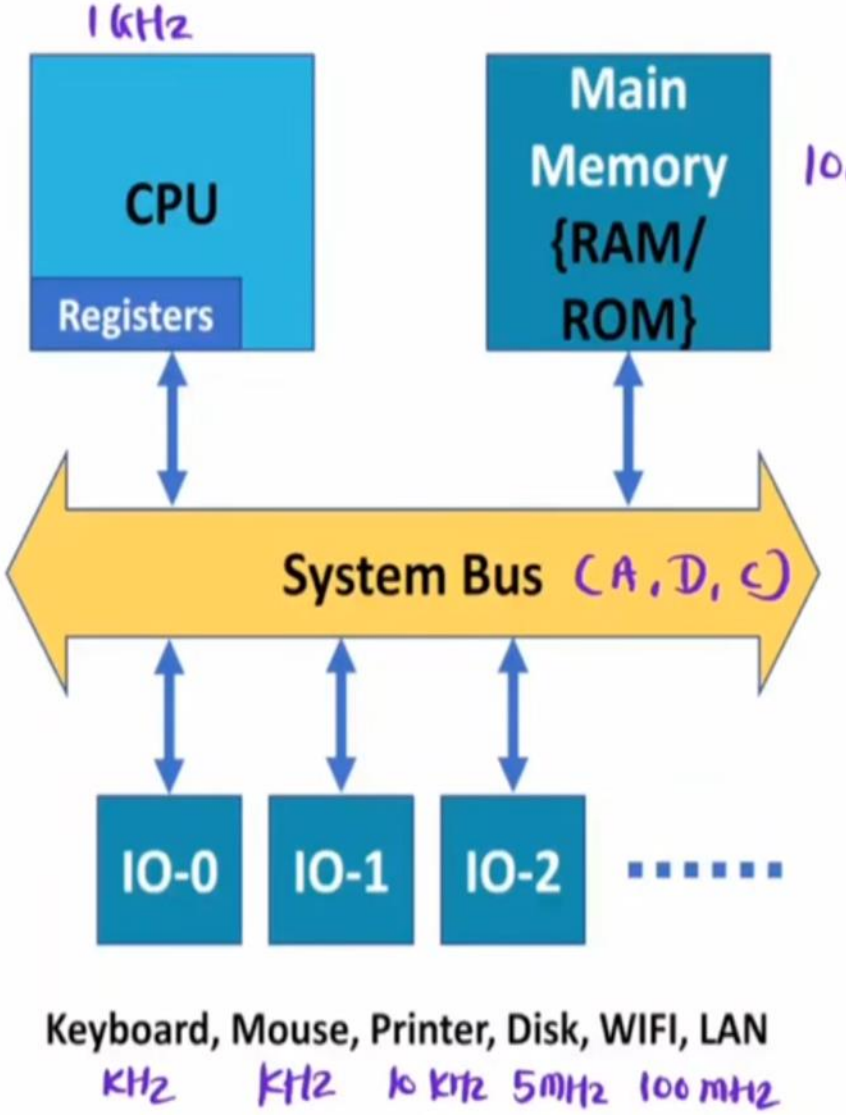
Features of 8257

- It has four channels which can be used over four I/O devices.
- Each channel has 16-bit address and 14-bit counter.
- Each channel can transfer data up to 64kb.
- Each channel can be programmed independently.
- Each channel can perform read transfer, write transfer and verify transfer operations.
- It generates MARK signal to the peripheral device that 128 bytes have been transferred.
- It requires a single phase clock.
- Its frequency ranges from 250Hz to 3MHz.
- It operates in 2 modes, i.e., **Master mode** and **Slave mode**.

Block Diagram of 8257

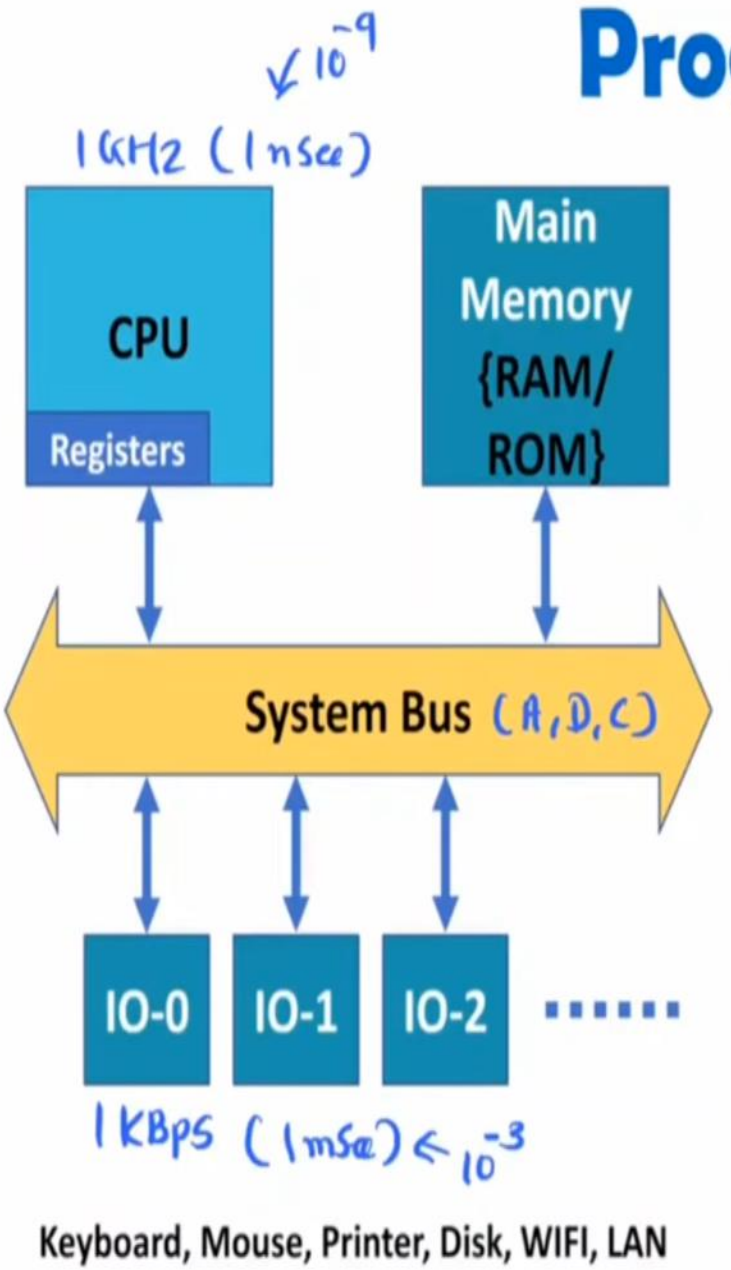


Input Output Organization



- IO Transfer Modes:
  1. Programmed IO
  2. Interrupt Driven IO
  3. DMA

Programmed IO



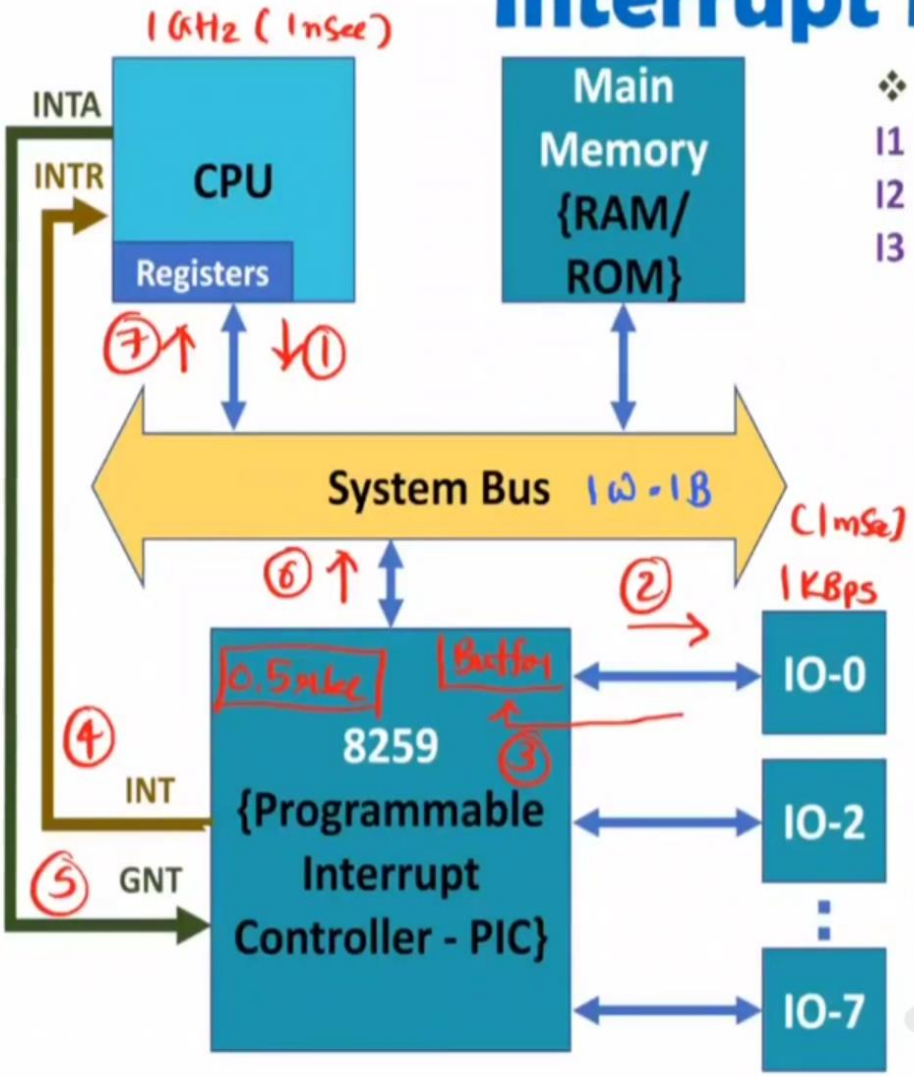
❖ Program execution:

I1 ...
I2 LDA IO-0
I3 ...

❖ Disadvantage:

1. CPU will have more wait time.
2. This type of IO configuration makes performance of CPU Slower

Interrupt Driven IO



❖ Program execution:

I1	...
I2	LDA IO-0
I3	...

❖ Another Program:

I1	...
I2	...
I3	...

❖ Advantage:

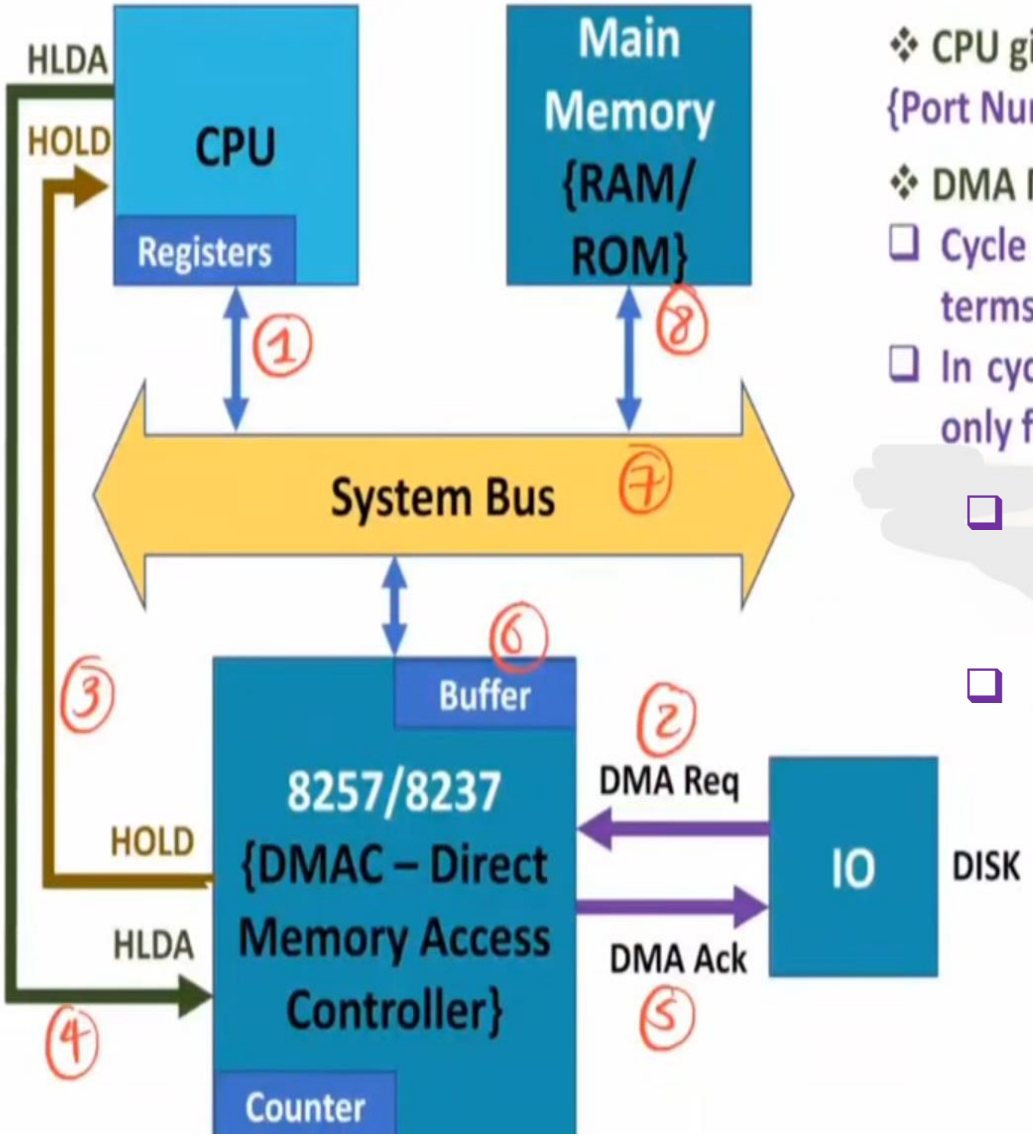
- ❑ CPU will have less block time compared to Programmed IO.

❖ Disadvantage:

- ❑ For data transfer in between IO and Memory CPU stays busy.

Keyboard,
Mouse,
Printer,
Disk, (1 MB)
WIFI,
LAN

DMA – Direct Memory Access



- CPU gives IO command at start.
  {Port Number, Memory Address, Count, Control Signal}
- DMA Modes {It depends on size of buffer}
  - Cycle stealing: in this mode, DMA sends data in terms of byte by byte or word by word.
  - In cycle stealing, DMAC waits till buffer is full then only for one cycle only. it takes control of system.
  - Burst / Batch Mode: In this mode DMA sends data in bulk. Here size of buffer is large
  - * In Burst mode once DMAC buffer is full, it takes control of system bus for data transfer.

What is DMA in Cycle-Stealing Mode?

•DMA allows an I/O device to transfer data directly to or from main memory **without CPU intervention**.

•In **cycle-stealing mode**, the DMA controller "steals" memory cycles to transfer data, temporarily **blocking the CPU** from accessing the memory.

We are calculating how much of the **CPU time** is spent in a **blocked state** (i.e., when the CPU cannot access the memory due to DMA operations). The calculation involves:

1. **Buffer Time (T_fill)**:
   - The time it takes for the I/O device to generate enough data to fill a buffer for a DMA transfer.
   - Formula:

$$T_{\text{fill}} = \frac{\text{Buffer Size}}{\text{Data Rate}}$$

2. **Transfer Time (T_transfer)**:
   - The time it takes the DMA controller to transfer the buffer's data to memory.
   - This transfer **blocks the CPU** because memory cycles are used for DMA instead of the CPU.
3. **Cycle Time (T_cycle)**:
   - The total time for a single cycle includes:
     - Time to fill the buffer (T_{fill}).
     - Time for the DMA to transfer the buffer to memory (T_{transfer}).
4. **Percentage of Time CPU is Blocked**:
   - The ratio of **blocked time** (DMA transfer time) to the **total cycle time**.
   - Formula:

$$\text{Blocked Percentage} = \frac{T_{\text{transfer}}}{T_{\text{cycle}}} \times 100$$

DIRECT MEMORY ACCESS NUMERICAL

Q. What is the **%age of time CPU is busy** & the **%age of time CPU is blocked** if the program execution required **x** time units for **Data Collection** & **y** time units for **Data Transfer?**

Solution:

CPU Busy → during Data Collection = x time units

CPU Idle → during Data Transfer = y time units

% of time CPU Busy= (x/x+y)×100

% of time CPU Idle = (y/x+y)×100

Question: (RTU)

Consider a typical disk that: Rotates at **15,000 rotations per minute (RPM)** and Has a **transfer rate** of 50×10^6 bytes/sec.

If the **average seek time** of the disk is **twice the average rotational delay** and the **controller's transfer time** is **10 times** the disk transfer time, Find the average time (in milliseconds) to read or write a **512 bytes** sector of the disk is?

Solution:

Step 1: Average Rotational Delay

Time for one rotation:

$$\text{Time for 1 rotation} = \frac{60}{15,000} = 4\,\text{ms}.$$

Average rotational delay is half of this:

$$\text{Average Rotational Delay} = \frac{4}{2} = 2\,\text{ms}.$$

Step 2: Average Seek Time

Given that the average seek time is twice the average rotational delay:

$$\text{Average Seek Time} = 2 \times 2 = 4\,\text{ms}.$$

Step 3: Disk Transfer Time

Transfer rate: $50 \times 10^6\,\text{bytes/sec}$

Sector size: $512\,\text{bytes}$

$$\text{Disk Transfer Time} = \frac{\text{Sector Size}}{\text{Transfer Rate}} = \frac{512}{50 \times 10^6} = 0.01024\,\text{ms}.$$

Step 4: Controller Transfer Time

Given that the controller transfer time is 10 times the disk transfer time:

$$\text{Controller Transfer Time} = 10 \times 0.01024 = 0.1024\,\text{ms}.$$

Step 5: Total Average Time

The total average time to read/write a 512-byte sector is:

$$\text{Total Time} = \text{Seek Time} + \text{Rotational Delay} + \text{Controller Time} + \text{Disk Transfer Time}$$

Substitute the values:

$$\text{Total Time} = 4 + 2 + 0.1024 + 0.01024 = 6.11264\,\text{ms}.$$

Question: (RTU)

Consider a disk drive with the following specifications:

- 16 surfaces, 512 tracks/surface, 512 sectors/track;
- 1KB/sector, rotation speed 3000 rpm.

The disk is operated in **cycle stealing mode** whereby whenever one byte word is ready it is sent to memory; similarly, for writing, the disk interface reads a 4-byte word from the memory in each DMA cycle. Memory cycle time is **40 nsec**. The maximum percentage of time that the CPU gets blocked during the DMA operation is:

Solution:

Revolution Speed = 3000rpm
= 3000/50 rotations per sec
= 60
In 1 track rotation = 512 sectors
1 sectors = 1 KB data
1 track rotation = 512Kb data transfer
In 50 rotations = 512*50
= 25600 byte
Now 50 rotations in 1second
25600 bytes = 1 second
1 byte = 1/25600 sec
= 39.06 ns
For 1 byte = 39.06 ns so for 4 bytes = 4*39.06 = 156ns
*** % age of time that CPU gets blocked During DMA operation = 40/156 * 100%
= 25 %

The size of the data count register of a DMA controller is 16 bits. The processor needs to transfer a file of 29,154 kilobytes from disk to main memory. The memory is byte addressable. The minimum number of times the DMA controller needs to get the control of the system bus from the processor to transfer the file from the disk to main memory is _____.

Concept:

A hardware device used for **DMA (Direct memory access)** is DMA controller. DMA controller is a control unit, which can transfer blocks of data between I/O devices and main memory with intervention from the processor.

Data count register provides the number of words DMA can transfer in one cycle.

If memory is byte addressable; 1 word = 1 byte

1 kilobytes = 2^{10} bytes

Calculation:

Size of data counter register of DMA controller = 16 bits

It means, 2^{16} words can be transferred in one cycle.

As, memory is byte addressable.

So, 2^{16} bytes in one cycle. (2^{16} bytes = 2^{6} kilobytes)

File size = 29154 kilobytes

Minimum number of times the DMA controller needs to get control of the system bus from the processor to transfer the file from the disk to main memory is = $\left\lceil \frac{File\ size}{bytes\ in\ one\ cycle} \right\rceil = \left\lceil \frac{29154\ KB}{2^{16}} \right\rceil = 456$

Consider a computer system with DMA support. The DMA module is transferring one 8-bit character in one CPU cycle from a device to memory through cycle stealing at regular intervals. Consider a 2 MHz processor. If 0.5% processor cycles are used for DMA, the data transfer rate of the device is _____ bits per second.

Explanation :

We have given 2 MHz processor, 8-bit character in one CPU cycle, 0.5% of CPU cycles are used for DMA.

Frequency of processor = 2 MHz

In 1 second = 2×10^6 cycles

From these cycles 0.5% cycles are used for DMA transfers

= $0.005 \times 2 \times 10^6$

= **10000 cycles per second** used for DMA transfer

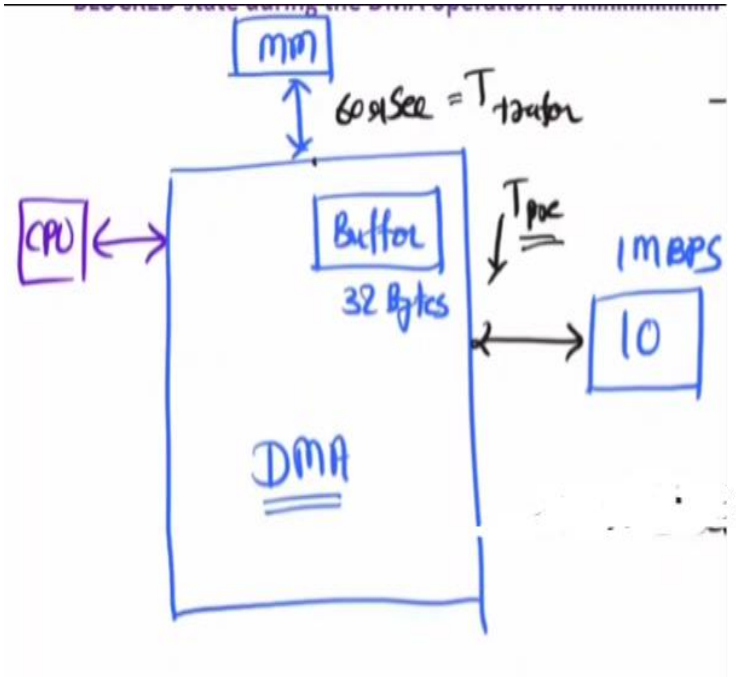
And **also 8 bit character in one cpu cycle** .

= 10000×8= 80000 bits in **each second** (total)

So **transfer rate** = **80000 bits per second**.

Examples on DMA: Consider a 1MBPS IO device interfacing with CPU using DMA in cycle stealing mode. Whenever 32bytes are present in the buffer, it is transferred to the main memory. Main memory cycle time is 60micro second. Percentage of time CPU is in BLOCKED state during the DMA operation is……..

Solution:



One cycle time, T_{cycle} = 10^-6 sec = 1micro sec

To fill the buffer time takes is, T_{prep} = 32 * 1 cycle time period = 32 micro sec

Transfer (Buffer to MM) Time, $T_{transfer}$ = 60 micro sec

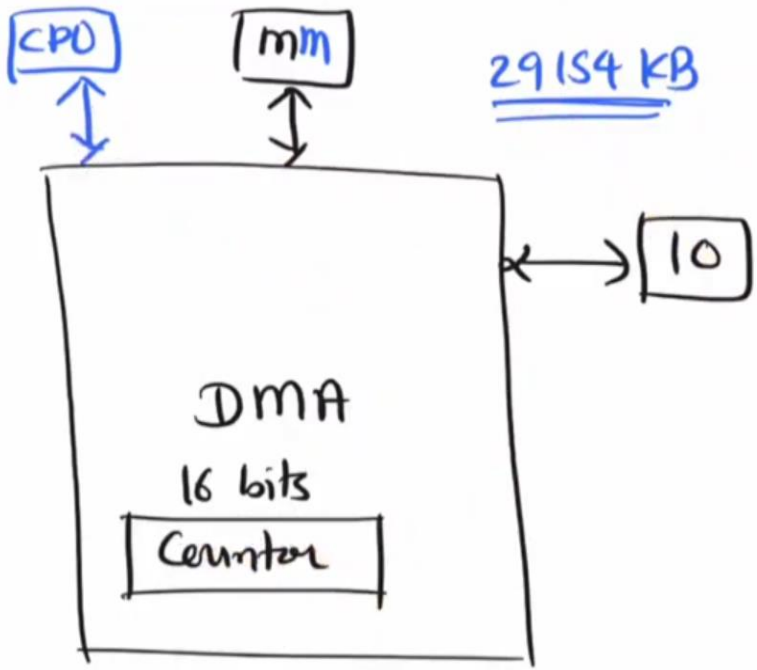
Percentage (%) Block = ($T_{transfer}$ / $T_{transfer}$ + T_{prep}) *100%

= (60 / 60+32) 100%

= 65.21 %

Examples on DMA 2:

- The size of the data count register of a DMA controller is 16 bits. The processor needs to transfer a file of 29154 KB from disk to main memory. Here memory is byte addressable. The minimum number of times the DMA controller needs to get control of the system bus from the processor to transfer the file from the disk to main memory is .....



→ In one go data transfer
= 2^{16} B
= 64 KB

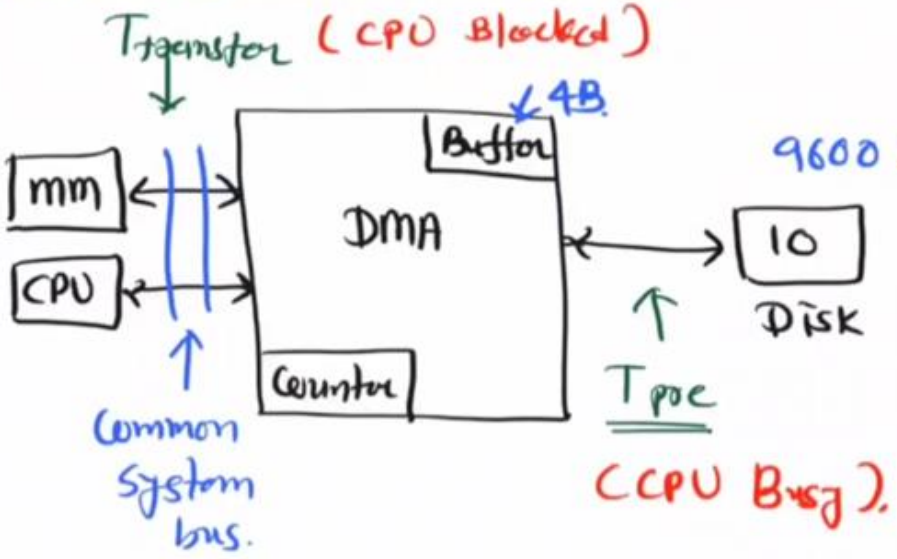
→ = $\frac{29154}{64}$ = 455.5 = 456

Examples on DMA 3:

Examples on DMA

- A DMA controller transfers 32 bit words to memory using cycle stealing. The words are assembled from a device that transmits characters at a rate of 9600 characters per second. Assume one character is encoded using 1 byte. The CPU is fetching and executing instructions at an average rate of two million instructions per second. By how much % with the CPU be slowed down because of the DMA transfer?

A. 0.6%
B. 0.12%
C. 1.2%
D. 2.5%



$\rightsquigarrow T_{proc} = \frac{4B}{9600\ Bps}$

$= 4.166 \times 10^{-4}$ sec

$\approx$

$T_{transfer} = 1/ 2*10^6 = .5*10^-6$ sec

Percentage (%) Block = $(T_{transfer} / T_{transfer} + T_{prep})$ *100%

= 0.12% (Ans)

Examples on DMA 4:

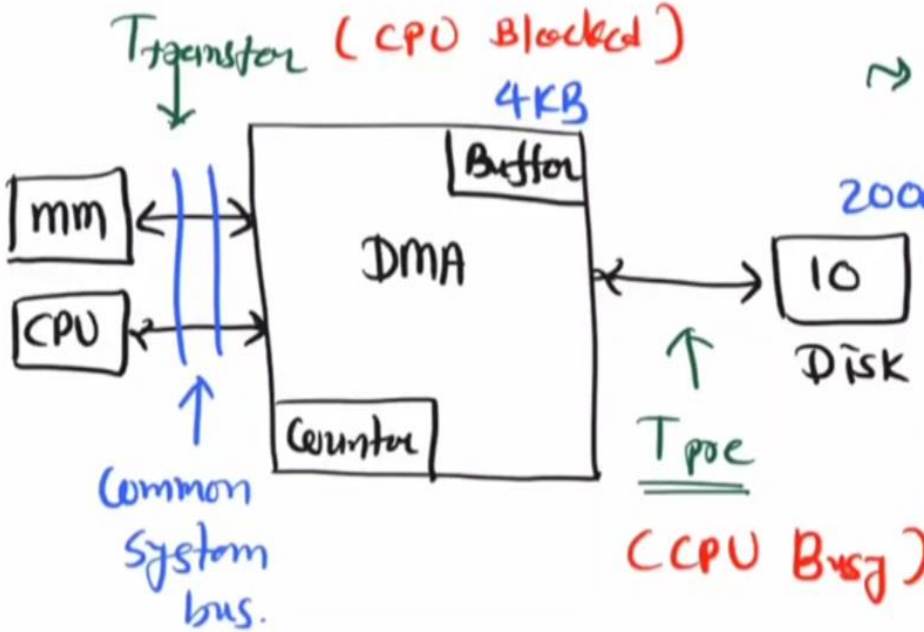
- A hard disk is connected to a 50 MHz processor through a DMA controller. Assume that the initial set up of DMA transfer takes 1000 clock cycles for the processor. The hard disk has a transfer rate of 2000 KBPS and average block transferred is 4KB. What fraction of the processor time is consumed by the disk, if the disk is actively transferring 100% of the time.

$\rightsquigarrow T_{proc} = \frac{4KB}{2000\ KBPS} = 2 \times 10^{-3}$ sec

$\rightsquigarrow T_{transfer} = 1000\ t_{cyc}$

$= 1000 \times \frac{1}{50 \times 10^6}$

$= 20 \times 10^{-6}$ sec



$\rightsquigarrow$ % Block of CPU $= \frac{T_{transfer}}{T_{transfer} + T_{proc}} \times 100$

= 1% (Ans)

1 Byte	= 8 bits	
1 KB	= 1024 B(ytes)	= 2^{10} B
1 MB	= 1024 KB	= 2^{20} B
1 GB	= 1024 MB	= 2^{30} B

DMA - Problems

Problem 1:

A hard disk with a transfer rate of 20 Mbytes/second is constantly transferring data to memory using DMA. The processor runs at 300MHz, and takes 300 and 900 clock cycles to initiate and complete DMA transfer respectively. If the size of the transfer is 20 Kbytes, what is the percentage of processor time consumed for the transfer operation?

Solution:

$$Total\ Time = \frac{Total\ Data}{Total\ Transfer\ Rate}$$

Step 1: Find Total Transfer Time

Transfer rate=20 MB per second = 20×2^{20} bytes/sec

Data=20 KB=20×2^{10} bytes

Time=$(20 \times 2^{10})/(20 \times 2^{20}) = 1 \times 2^{-10} = 1 \times 10^{-3}$ =1 ms

$$Processor\ Time = \frac{Cycles\ for\ DMA\ transfer}{Processor\ Speed}$$

$$\%\ Processor\ Time = \frac{Processor\ Time}{Total\ Time} \times 100$$

Step 2 :Find Processor Time

1 MHz = 10^6 cycles/sec

Processor speed= 300 MHz = 300×10^6 Cycles/sec

Cycles required by CPU for DMA Transfer =300+900 =1200 cycles

Time=1200 cycles/(300×10^6) cycles/sec = 4×10^{-6} sec = .004 ms

Step 3 : Find % of Processor Time

% Processor Time = $(.004/1) \times 100 = 0.4\%$

Term	Normal Usage	Usage as Power of 2
K (Kilo)	10^3	2^{10} = 1,024
M (Mega)	10^6	2^{20} = 1,048,576
G (Giga)	10^9	2^{30} = 1,073,741,824
T (Tera)	10^{12}	2^{40} = 1,099,511,627,776

1 Byte	= 8 bits	
1 KB	= 1024 B(ytes)	= 2^{10} B
1 MB	= 1024 KB	= 2^{20} B
1 GB	= 1024 MB	= 2^{30} B

DMA - Problems

Problem 2:

A hard disk with a transfer rate of 10 Mbytes/second is constantly transferring data to memory using DMA. The processor runs at 600MHz, and takes 300 and 900 clock cycles to initiate and complete DMA transfer respectively. If the size of the transfer is 20 Kbytes, what is the percentage of processor time consumed for the transfer operation?

Solution:

$$Total\ Time = \frac{Total\ Data}{Total\ Transfer\ Rate}$$

Step 1: Find Total Transfer Time

Transfer rate=10 MB per second = 10×2^{20} bytes/sec

Data=20 KB=20×2^{10} bytes

Time=$(20 \times 2^{10})/(10 \times 2^{20}) = 2 \times 2^{-10} = 2 \times 10^{-3} = 2$ ms

$$Processor\ Time = \frac{Cycles\ for\ DMA\ transfer}{Processor\ Speed}$$

$$\%\ Processor\ Time = \frac{Processor\ Time}{Total\ Time} \times 100$$

Step 2 :Find Processor Time

1 MHz = 10^6 cycles/sec

Processor speed= 600 MHz = 600×10^6 Cycles/sec

Cycles required by CPU for DMA Transfer =300+900 =1200 cycles

Time=1200 cycles/(600×10^6) cycles/sec = 2×10^{-6} sec = .002 ms

Step 3 : Find % of Processor Time

% Processor Time = $(.002/2) \times 100 = 0.1\%$

Term	Normal Usage	Usage as Power of 2
K (Kilo)	10^3	$2^{10} = 1,024$
M (Mega)	10^6	$2^{20} = 1,048,576$
G (Giga)	10^9	$2^{30} = 1,073,741,824$
T (Tera)	10^{12}	$2^{40} = 1,099,511,627,776$