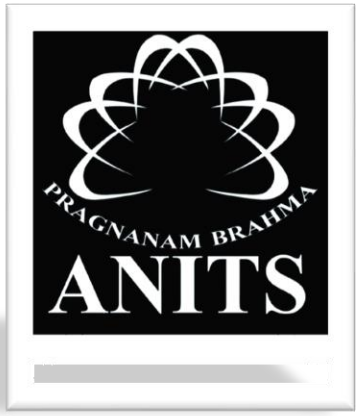
# Course Name: Micro Processors and Micro Controllers

## Topic Name: Microprocessor interfacing with Advanced devices




Dr. Appalabathula Venkatesh M.Tech, Ph.D
Assistant Professor,
Department of EEE, ANITS

# Cos

Course Objectives:

At the end of the course, students will be able to

- **CO1** (BL-2) Describe the architecture and various addressing modes of a typical 8085 microprocessor
- **CO2** (BL-4) Classify different memory devices to Discuss the interfacing between memory and 8085 microprocessor
- **CO3** (BL-3) Describe the architecture of a typical 8086 microprocessor to illustrate the general bus operations
- **CO4** (BL-3) Describe the various peripheral devices and show how the peripherals (8259,8251 & 8253) are interfaced with Microprocessor.
- **CO5** (BL-4) Use the architecture of 8051 microcontroller and illustrate how 8051 is interfaced with advanced applications.

# Syllabus

## UNIT 4

**Microprocessor interfacing with Advanced Devices:**

Stepper Motor Interfacing, Key board/display device: 8279 block diagram and its operation, 8251(USART): block diagram and functions of each block, 8253 timer: block diagram and modes of operation.

# Stepper Motor

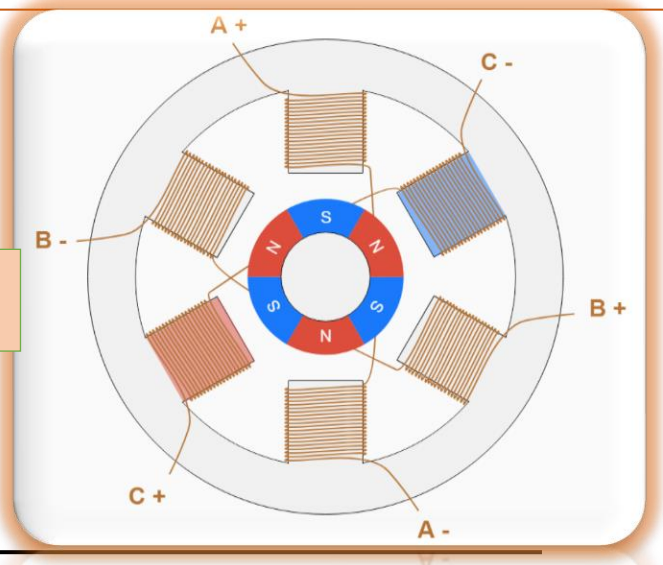
- Stepper motor is a brushless synchronous electric motor that converts train of electric pulses (digital pulses) applied at their excitation windings into mechanical shaft rotation.
- Speed of rotation depends on the rate at which the electrical signals are applied and the direction of rotation is dependent on the pattern of pulses that is followed.
- It is suitable for precise position, speed and direction control which are required in automation system.

**Principle of Operation:** The principle of operation of a stepper motor is based on the **interaction of electromagnets and permanent magnets.**

Inside a stepper motor, there are multiple sets of coils (windings) arranged in the stator and a permanent magnet rotor.

When electric current flows through the coils → They create magnetic fields that attract or repel the magnetic poles of the rotor, causing it to move in discrete steps → The direction and angle of rotation depend on the order and timing of the current pulses applied to the coils.

Permanent Magnet Stepper Motor



# Stepper Motor

**Modes of operation:**

Stepper motors operate in one of Four main modes: Wave mode, Full-step, Half-step, or Micro-stepping.

In wave mode, ⋙ Only one phase at a time is energized

In full-step mode, ⋙ The motor moves one full step per electrical pulse

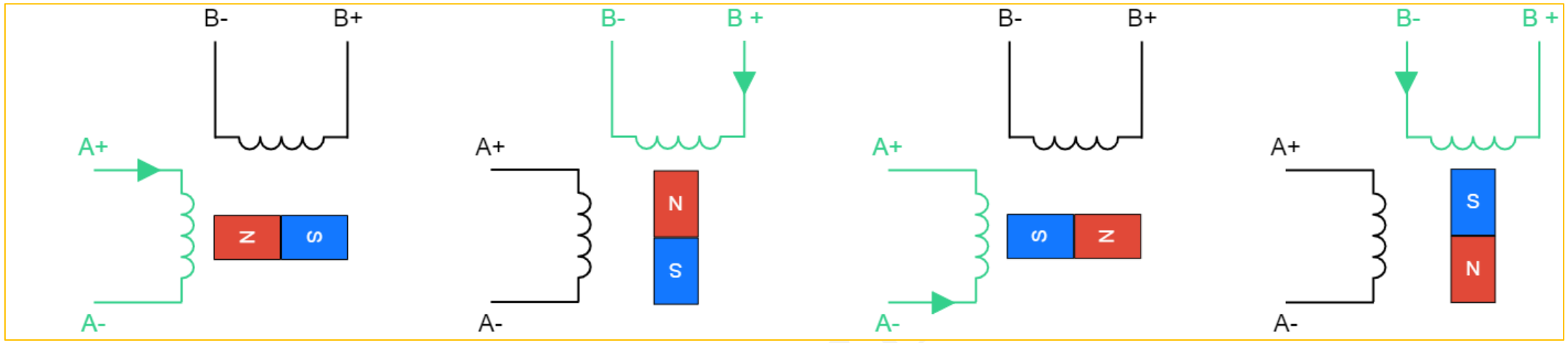
In half-step mode, ⋙ It moves half a step per pulse.

Micro-stepping ⋙ Allows for even finer control, where the motor moves in smaller increments between full steps, resulting in smoother motion and reduced vibration.

# Stepper Motor

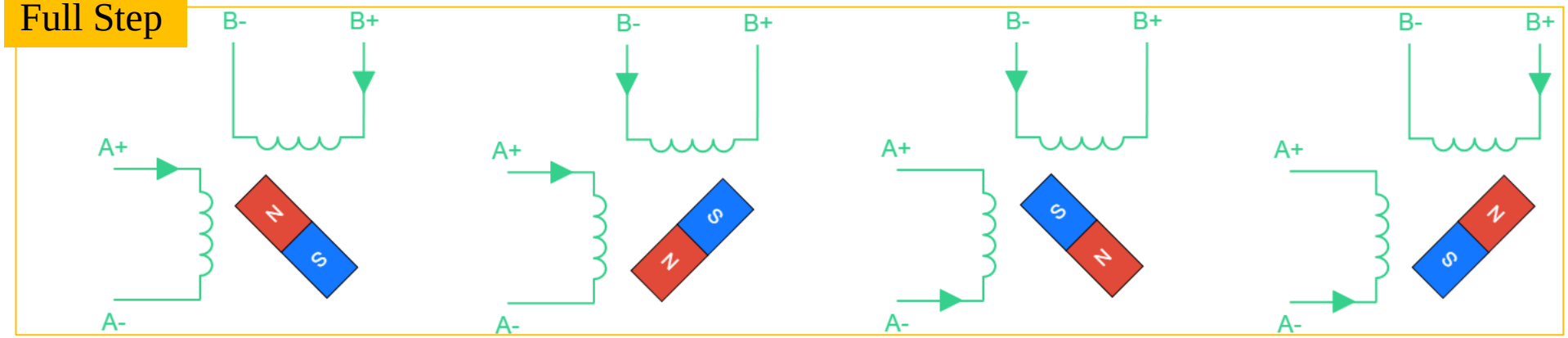
## Wave Mode

Rotor spins by 90°




Both Wave mode and Full Step modes are similar, the most significant difference being that with Full step mode, the motor is able to produce a higher torque since more current is flowing in the motor and a stronger magnetic field is generated

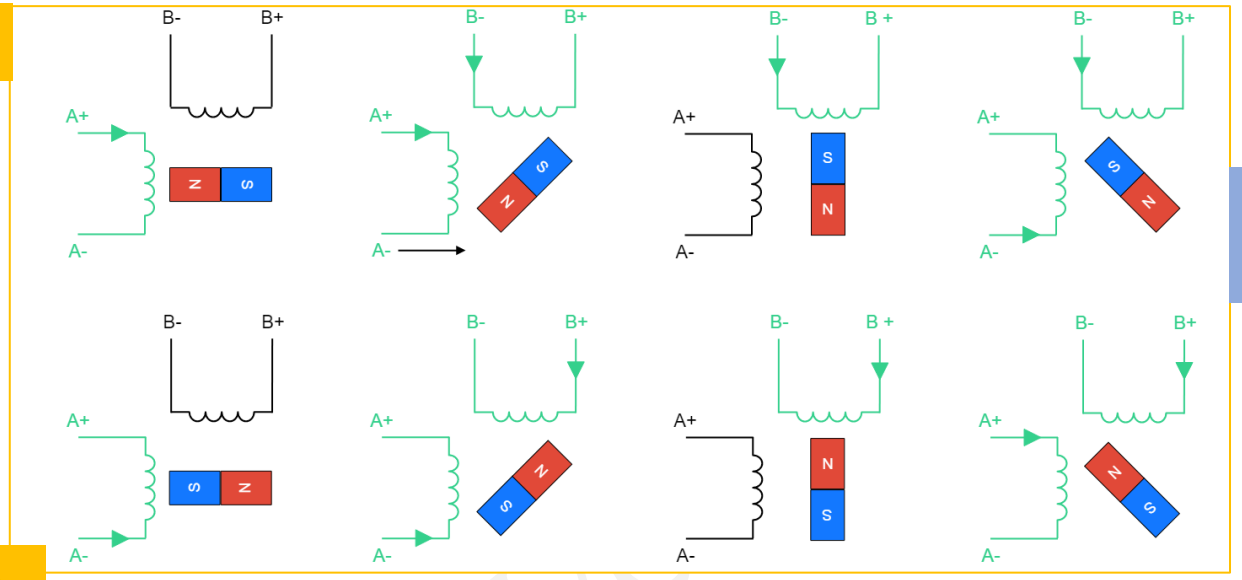
## Full Step




https://www.monolithicpower.com/en/stepper-motors-basics-types-uses

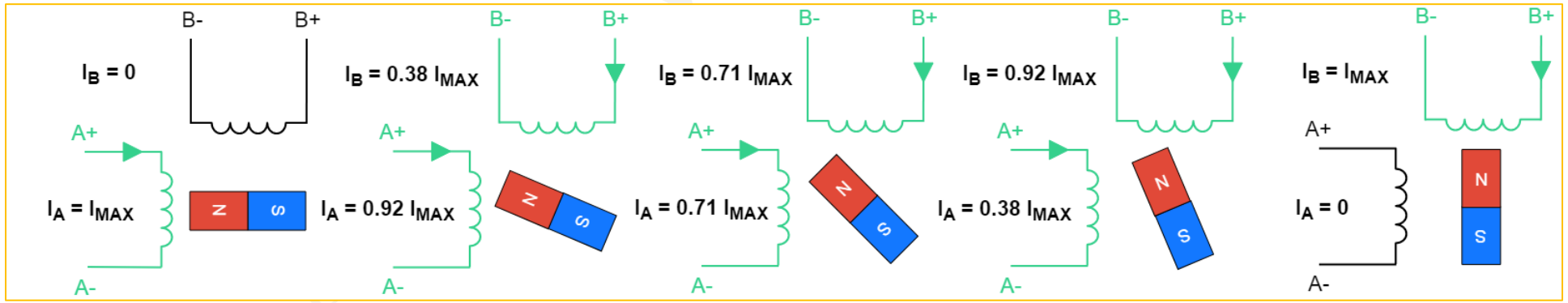
# Stepper Motors

## Half Step



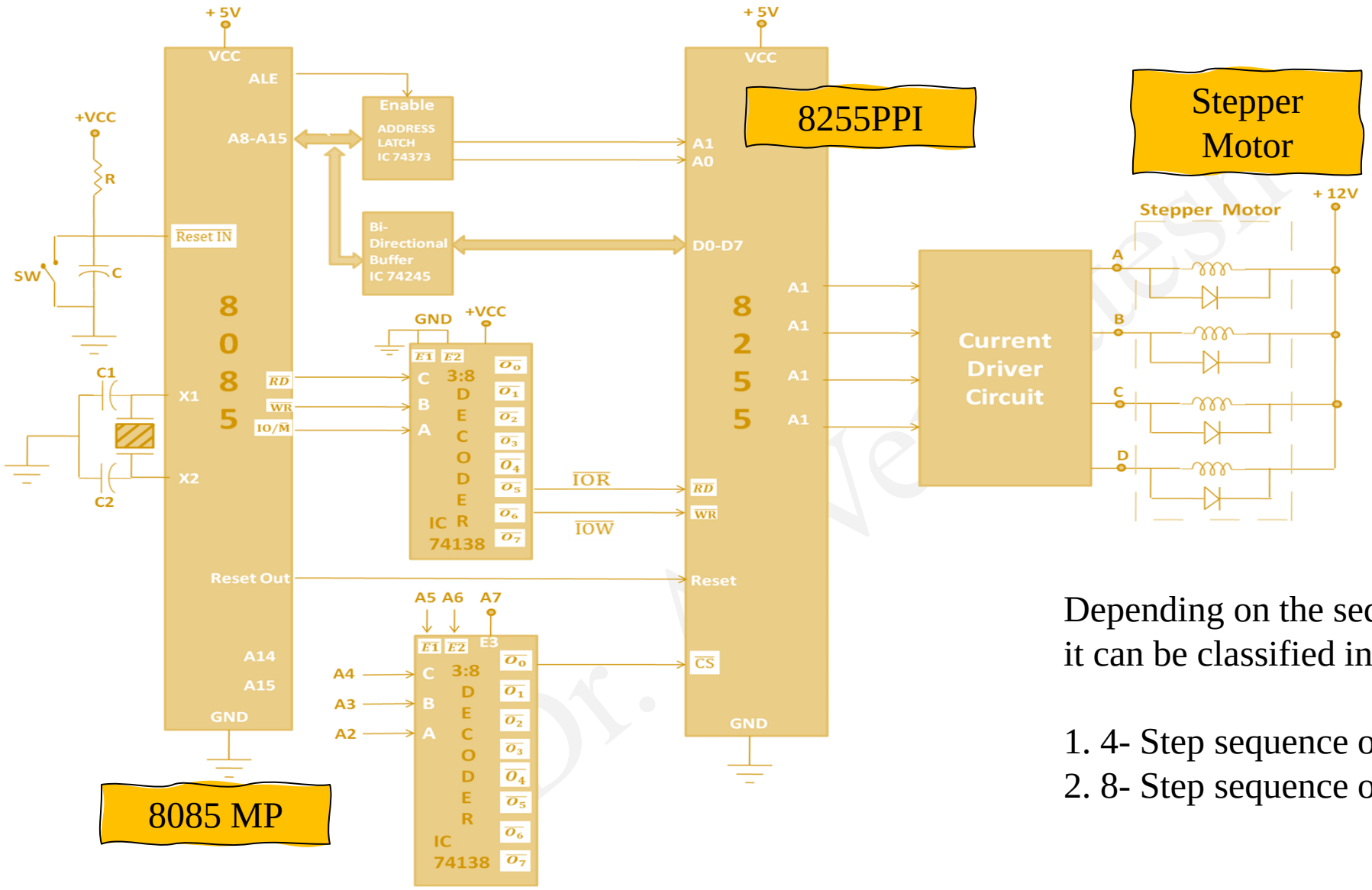

45°, 90° , 135° and so on

## Micro Step




22.5°, 45°, 67.5°, and 90°

# Stepper Motors interfacing with Microprocessor



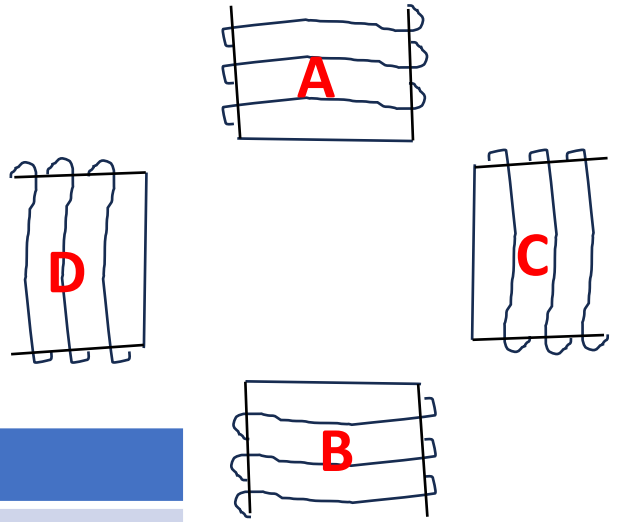

Depending on the sequence applied to stepper motor, it can be classified in two category:

1. 4- Step sequence or full step sequence
2. 8- Step sequence or half step sequence

# Stepper Motors interfacing with Microprocessor

## 4-Step sequence

| Step | A | B | C | D | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 0AH | For Clock-wise rotation |
| 2 | 1 | 0 | 0 | 1 | 09H | |
| 3 | 0 | 1 | 0 | 1 | 05H | |
| 4 | 0 | 1 | 1 | 0 | 06H | |
| | | | | | | |
| 4 | 0 | 1 | 1 | 0 | 06H | For Anti clock-wise rotation |
| 3 | 0 | 1 | 0 | 1 | 05H | |
| 2 | 1 | 0 | 0 | 1 | 09H | |
| 1 | 1 | 0 | 1 | 0 | 0AH | |

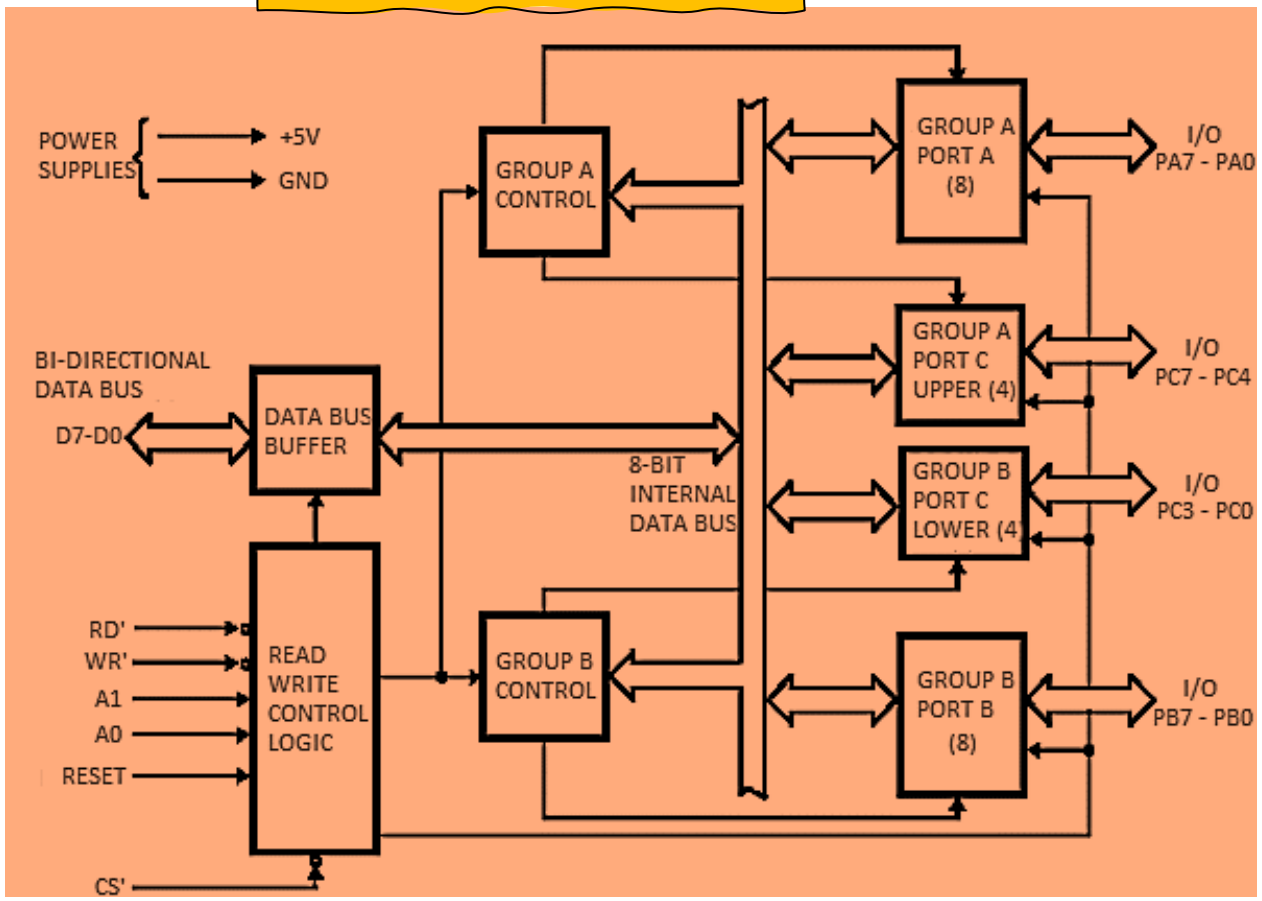



## 8-Step sequence

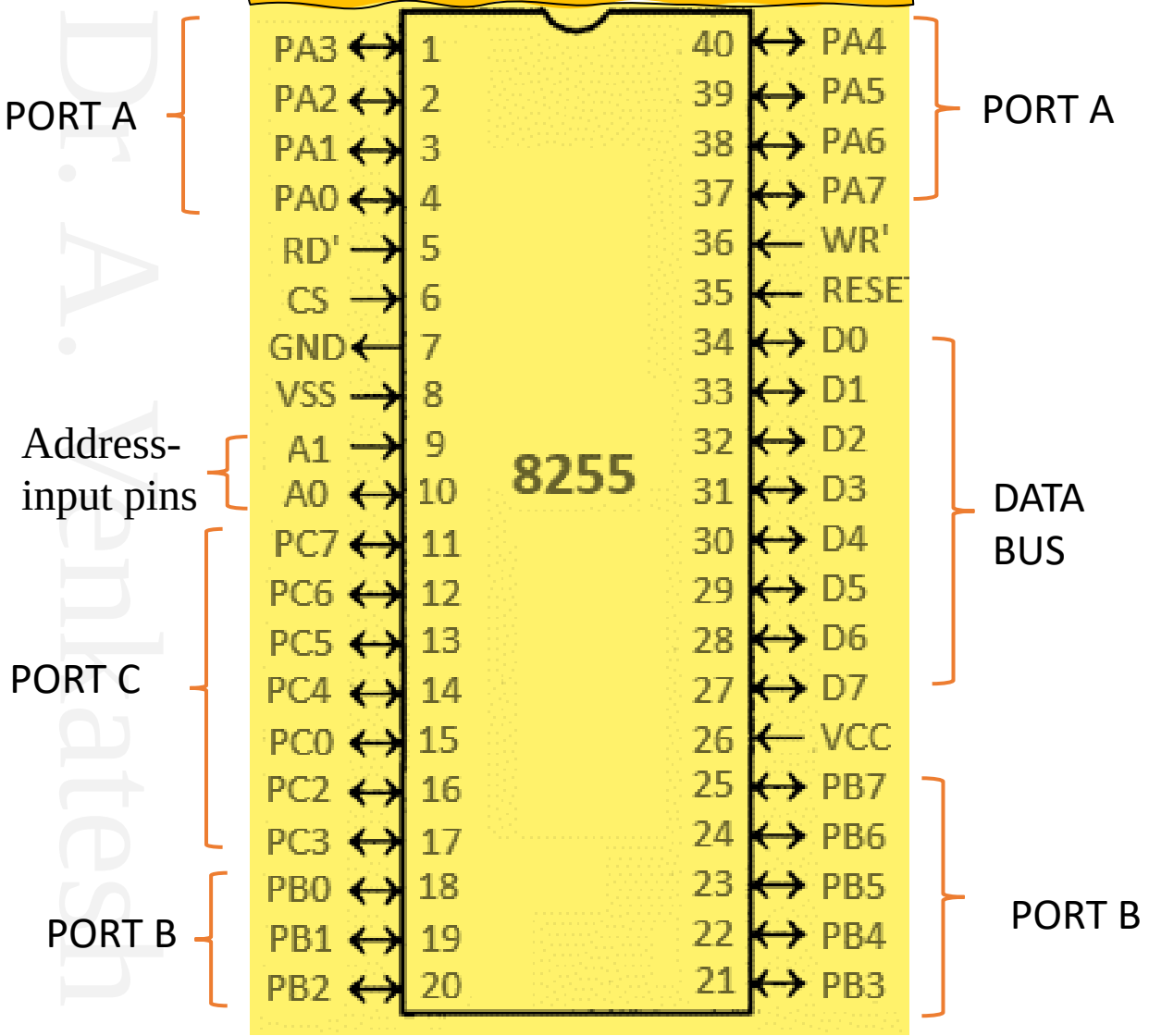
| Step | A | B | C | D | | |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 05H | Anti-clockwise rotation |
| 2 | 0 | 0 | 0 | 1 | 01H | |
| 3 | 1 | 0 | 0 | 1 | 09H | |
| 4 | 1 | 0 | 0 | 0 | 08H | |
| 5 | 1 | 0 | 1 | 0 | 0AH | |
| 6 | 0 | 0 | 1 | 0 | 02H | |
| 7 | 0 | 1 | 1 | 0 | 06H | |
| 8 | 0 | 1 | 0 | 0 | 04H | |

# Stepper Motors interfacing with Microprocessor

## Architecture of 8255PPI




## PIN Description of 8255PPI



# Stepper Motors interfacing with Microprocessor

## The pins of 8255 that are used for interfacing with a microprocessor

| | |
|---|---|
| CS* | It is an active low input pin for 8255. If this pin is at logic 0, the 8255 chip is selected for communication with the microprocessor. |
| D7-0 | They are connected to the data bus of the microcomputer system. |
| RD* | It is an active low input pin for 8255. It is connected to RD* output of 8085. The 8085 activates the RD* input of 8255 when it wants to read the data present in a port of 8255. |
| WR* | It is an active low input pin for 8255. It is connected to WR* output of 8085. The 8085 activates the WR* input of 8255 when it wants to write data to a port of 8255. |
| A1, A0 | These are address-input pins. They select one of the ports inside 8255 for communication with the microprocessor. |
| Reset | It is an active high input pin. It is connected to ResetOut output of 8085. It is used to reset the 8255. |

## The pins of 8255 that are used for interfacing with I/O devices

| | |
|---|---|
| $PA_{7-0}$ | These pins are output pins if Port A is programmed as an output port. They are input pins if Port A is programmed for input operation. |
| $PB_{7-0}$ | These pins are output pins if Port B is programmed as an output port. They are input pins if Port B is programmed for input operation. |
| $PC_{7-4}$ | These pins are output pins if Port C upper is programmed as an output port. They are input pins if Port C upper is programmed for input operation. |
| $PC_{3-0}$ | These pins are output pins if Port C lower is programmed as an output port. They are input pins if Port C lower is programmed for input operation. |

# Stepper Motors interfacing with Microprocessor

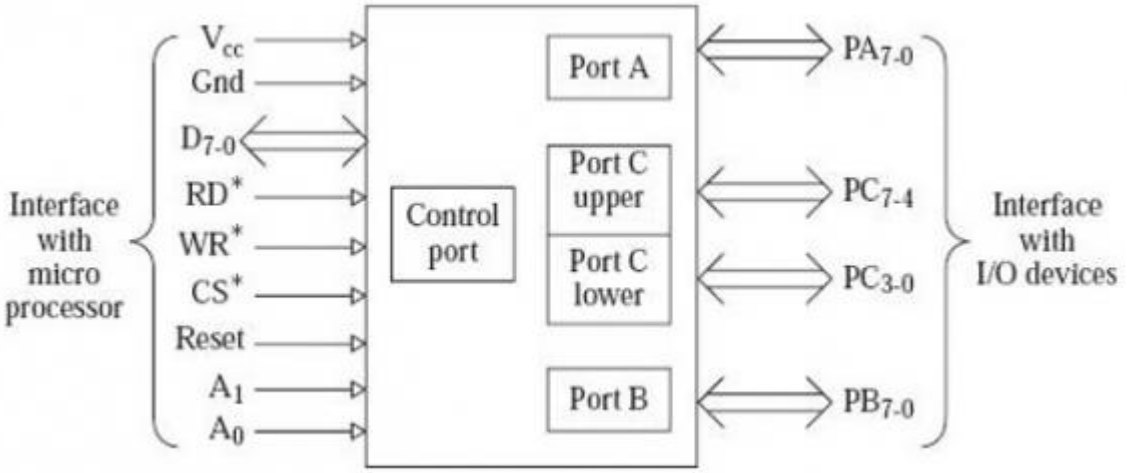
It is used for the connection of peripheral devices and interfacing

It consists of two programmable Input Output ports having of size 8 bits and
Two programmable Input Output ports of size 4 bits

We program **Port C lower as Input** and **Port C upper as Output**

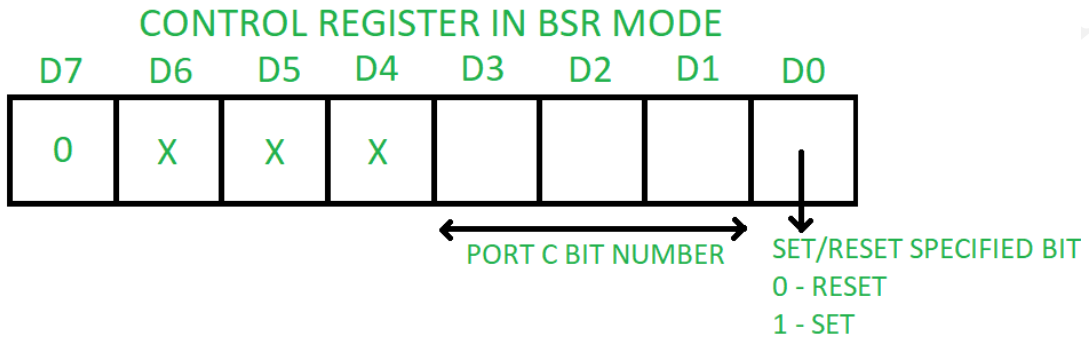
| *CS** | A1 | A0 | |
|---|---|---|---|
| 0 | 0 | 0 | Port A |
| 0 | 0 | 1 | Port B |
| 0 | 1 | 0 | Port C |
| 0 | 1 | 1 | Control Port |
| 1 | X | X | No selection |

- ✓ **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

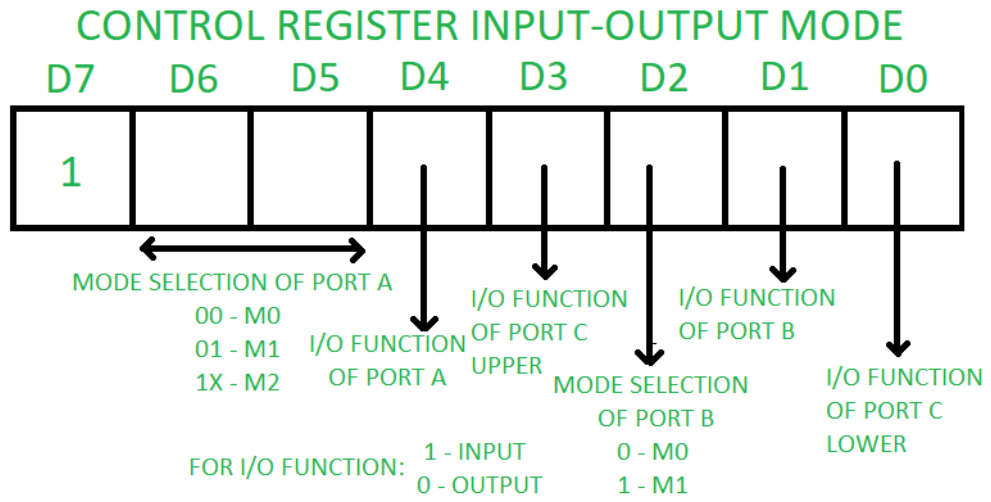


**Operating modes –**

**1.Bit set reset (BSR) mode –** If MSB of control word (D7) is 0, PPI works in BSR mode. In this mode only port C bits are used for set or reset.




**Input-Output mode –** If MSB of control word (D7) is 1, PPI works in input-output mode. This is further divided into three modes:



# Stepper Motors interfacing with Microprocessor

Mode 0(all the three ports (port A, B, C) can work) simple Input Output or the basic Input Output for performing the simplest mode of operation

Mode 1(either port A or port B can work as simple input port or simple output port, and port C bits are used for handshake signals before actual data transmission.) strobed Input Output or handshake Input Output. It is useful when data is supplied to the input device by the microprocessor at irregular interval of time

Example: A CPU wants to transfer data to a printer. In this case since speed of processor is very fast as compared to relatively slow printer, so before actual data transfer it will send handshake signals to the printer for synchronization of the speed of the CPU and the peripherals.


Mode 2 when the data is read by the processor the port informs the Input device that the processors already read the data. Bi-directional data bus mode. In this mode only port A works, and port B can work either in mode 0 or mode 1. 6 bits port C are used as handshake signals. It also has interrupt handling capacity.

# Special Motors(Stepper Motors)

Applications

- Consumer Electronics: They are used in devices like printers, scanners, and disk drives. Printers(Print heads, paper feed and scan bar)
- Robotics: Stepper motors are used in robotic systems to control the movement of robot arms, joints, and other mechanisms.
- DSLR cameras (autofocus and zoom control.)
- Security & surveillance cameras-Video Cameras(Pan, Tilt, Zoom and Focus)
- ATM machines(Bill Movement, Tray Elevators)
- Medical industry: stepper motors are widely used in samples, digital dental photography, respirators, fluid pumps, blood analysis machinery and medical scanners etc
- Stepper motors also used in elevators, conveyor belts and lane diverters

# Key Board Interfacing with Microprocessor-8279

The keyboard first scans the keyboard and identifies if any key has been pressed. It then sends their relative response of the pressed key to the CPU and vice-a-versa.

How Many Ways the Keyboard is Interfaced with the CPU?

The Keyboard can be interfaced either in the **interrupt or the polled mode.**

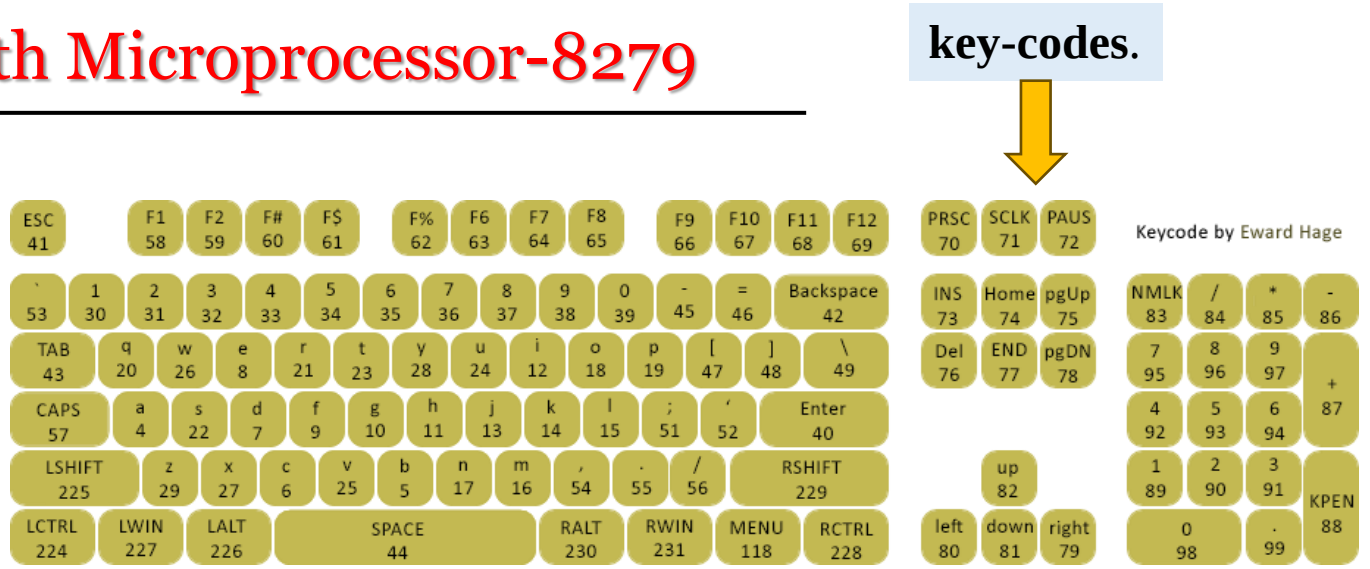
In the **Interrupt mode**, the processor is requested service only if any key is pressed, otherwise the CPU will continue with its main task.

In the **Polled mode**, the CPU periodically reads an **internal flag of 8279** to check whether any key is pressed or not with key pressure.

# Key Board Interfacing with Microprocessor-8279

## How Does 8279 Keyboard Work?

The keyboard consists of maximum 64 keys, which are **interfaced with the CPU** by using the **key-codes**. These key-codes are **de-bounced and stored** in an 8-byte **FIFORAM**, which can be accessed by the CPU.

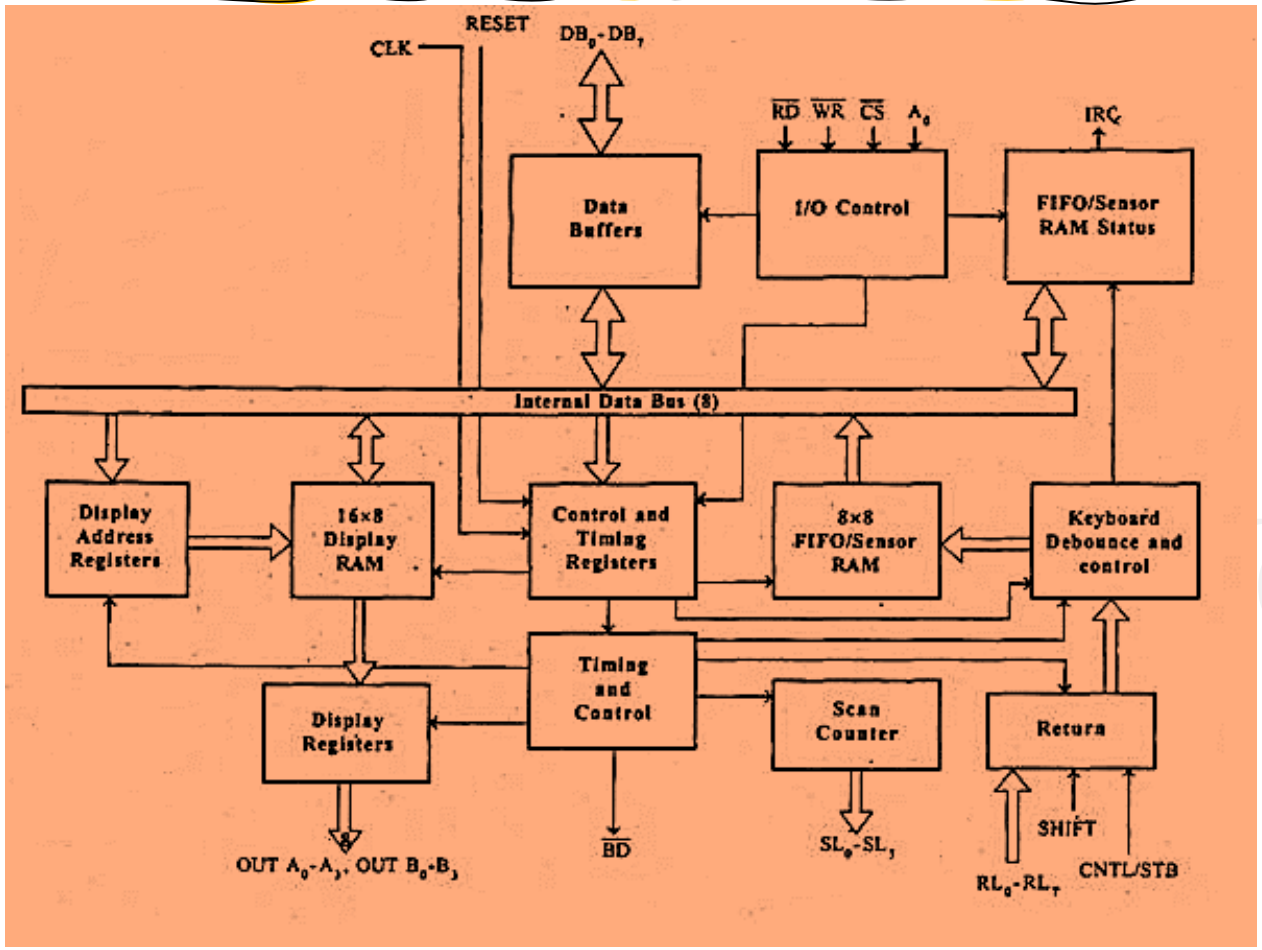



If more than 8 characters are entered in the FIFO, then it means more than eight keys are pressed at a time. This is when the overrun status is set.

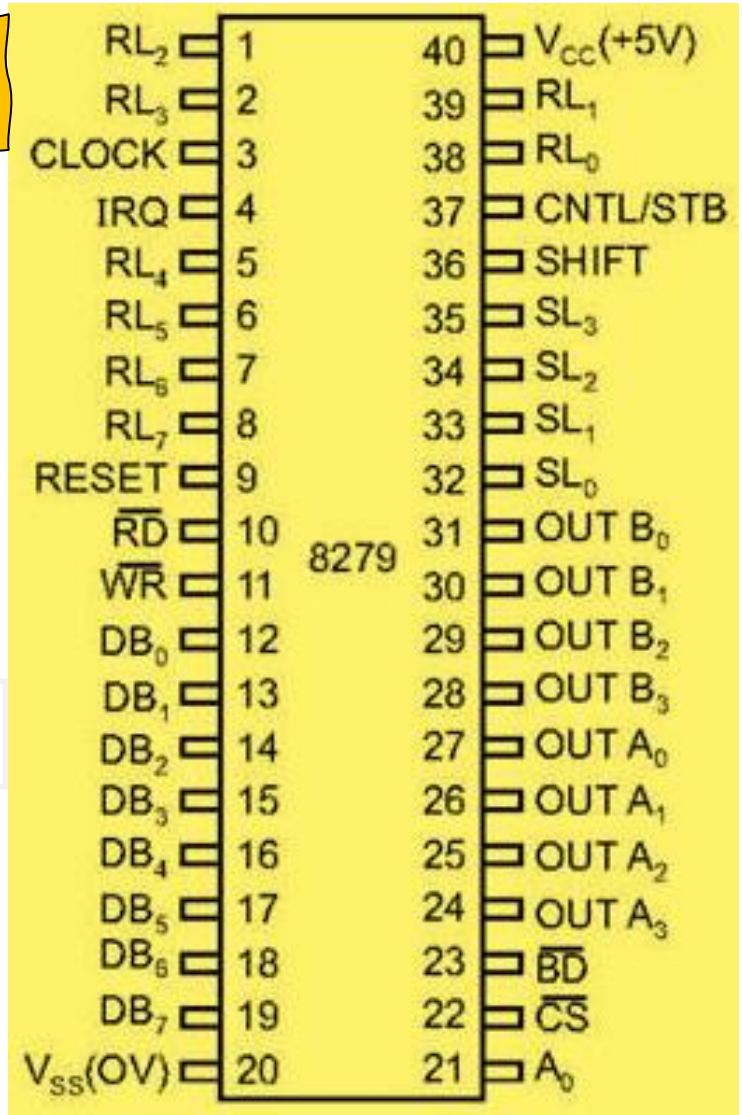
If a FIFO contains a valid key entry, then the CPU is interrupted in an interrupt mode else the CPU checks the status in polling to read the entry. Once the CPU reads a key entry, then FIFO is updated, and the key entry is pushed out of the FIFO to generate space for new entries.

# Architecture and Pin description of 8279

Architecture/Block diagram of of 8279PPI




PIN Description of 8279PPI



# 8279

**Data Bus Lines, $DB_0$ - $DB_7$**

These are 8 bidirectional data bus lines used to transfer the data to/from the CPU.

**CLK** →The clock input is used to generate internal timings required by the microprocessor.

**RESET**→As the name suggests this pin is used to reset the microprocessor.

**CS**→ Chip Select

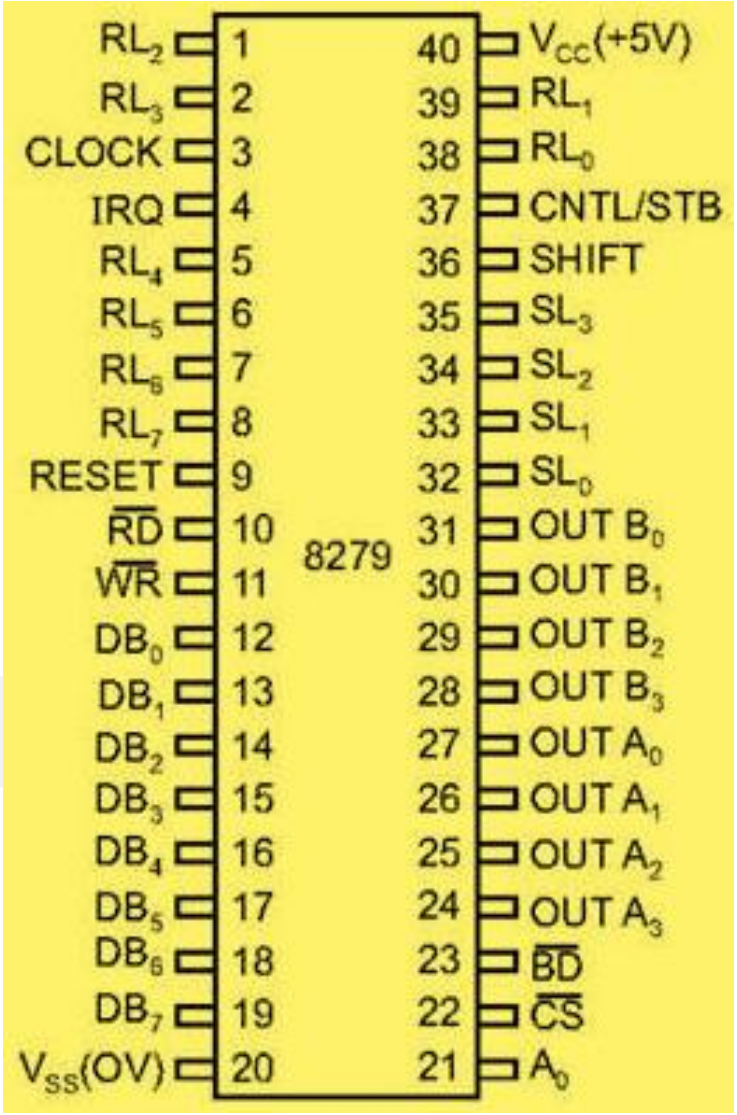
When this pin is set to low, it allows read/write operations, else this pin should be set to high.

**$A_0$** →This pin indicates the **transfer of command/status information**. When it is low, it indicates the transfer of data.

**RD, WR** →This Read/Write pin enables the data buffer to send/receive data over the data bus.

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

**$V_{ss}$, $V_{cc}$**→ These are the ground and power supply lines of the microprocessor.



# 8279

**$SL_0 - SL_3$ → Scan lines used to scan the keyboard matrix and display the digits**.

**$RL_0 - RL_7$ →Return Lines** which are connected **to one terminal of keys**, while the **other terminal of the keys is connected to the decoded scan lines**. These lines are set to 0 when any key is pressed.

**SHIFT**

The Shift input line status is stored along with every **key code in FIFO in the scanned keyboard mode**. Till it is pulled low with a key closure, it is pulled up internally to keep it high

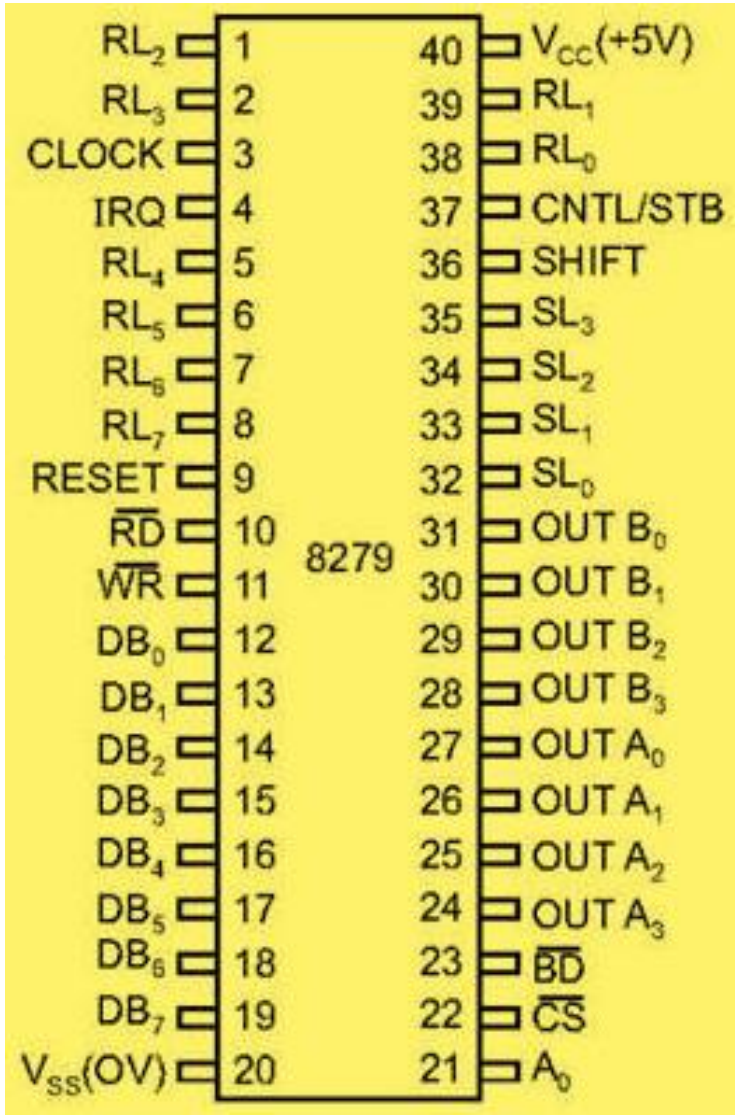
**CNTL/STB - CONTROL/STROBED I/P Mode**

In the keyboard mode, this line is used as a control input and stored in FIFO on a key closure. The line is a **strobe line that enters the data into FIFO RAM**, in the strobed input mode. It has an internal pull up. The line is pulled down with a key closure.

**BD**→ It stands for **blank display**. It is used to blank the display **during digit switching**.

**$OUTA_0 - OUTA_3$ and $OUTB_0 - OUTB_3$**

These are the output ports for two 16x4 or one 16x8 internal **display refresh registers**. The data from these lines is synchronized with the scan lines to scan the display and the keyboard.



## Operating Modes of 8279

There are two modes of operation on 8279 − Input Mode and Output Mode.

- Input Mode

➢ This mode deals with the input given by the keyboard and this mode is further classified into 3 modes.

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

➢ **Scanned Sensor Matrix** − In this mode, a sensor array can be interfaced with the processor using either encoder or decoder scans. In the encoder scan, 8×8 sensor matrix or with decoder scan 4×8 sensor matrix can be interfaced.

➢ **Strobed Input** − In this mode, when the control line is set to 0, the data on the return lines is stored in the FIFO byte by byte.

- Output Mode

➢ This mode deals with **display-related operations**. This mode is further classified into two output modes.

➢ **Display Scan** − This mode allows 8/16 character multiplexed displays to be organized as dual 4-bit/single 8-bit display units.

➢ **Display Entry** − This mode allows the data to be entered for display either from the right side/left side

# Internal Architecture of 8251

8251 **universal synchronous asynchronous receiver transmitter (USART)→ A**cts as a mediator between microprocessor and peripheral to **transmit serial data into parallel form and vice versa.**

**Prerequisite**

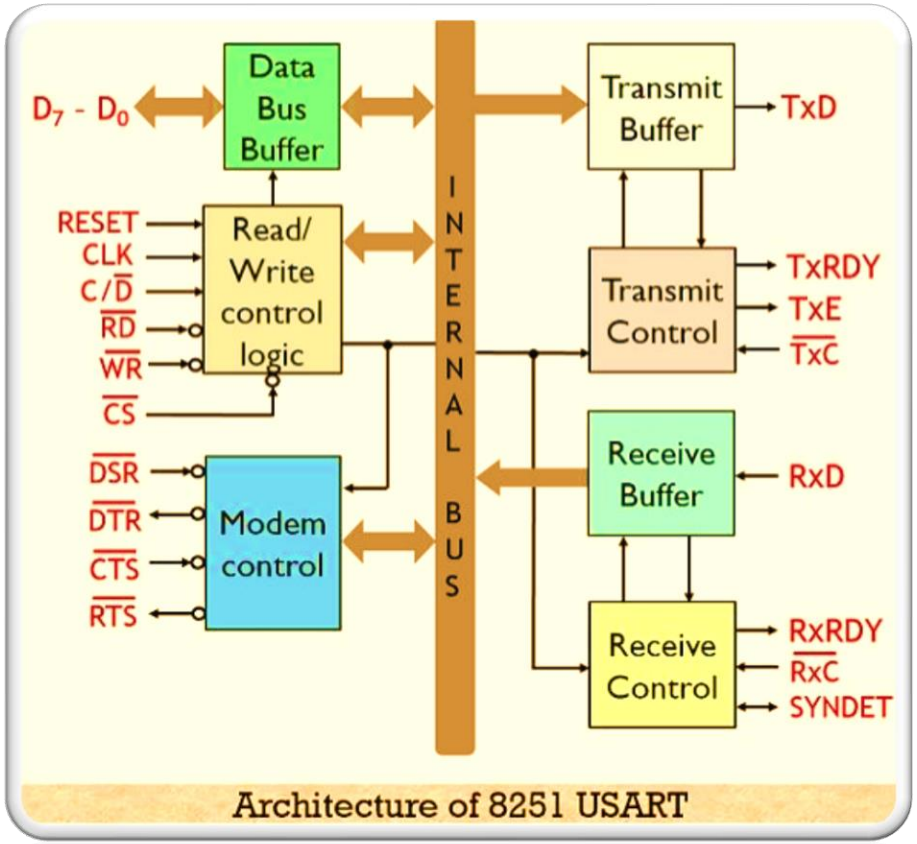
1.It takes data serially from peripheral (outside devices) and converts into parallel data.

2.After converting the data into parallel form, it transmits it to the CPU.

3.Similarly, it receives parallel data from microprocessor and converts it into serial form.

4.After converting data into serial form, it transmits it to outside device (peripheral).

## 8251 USART

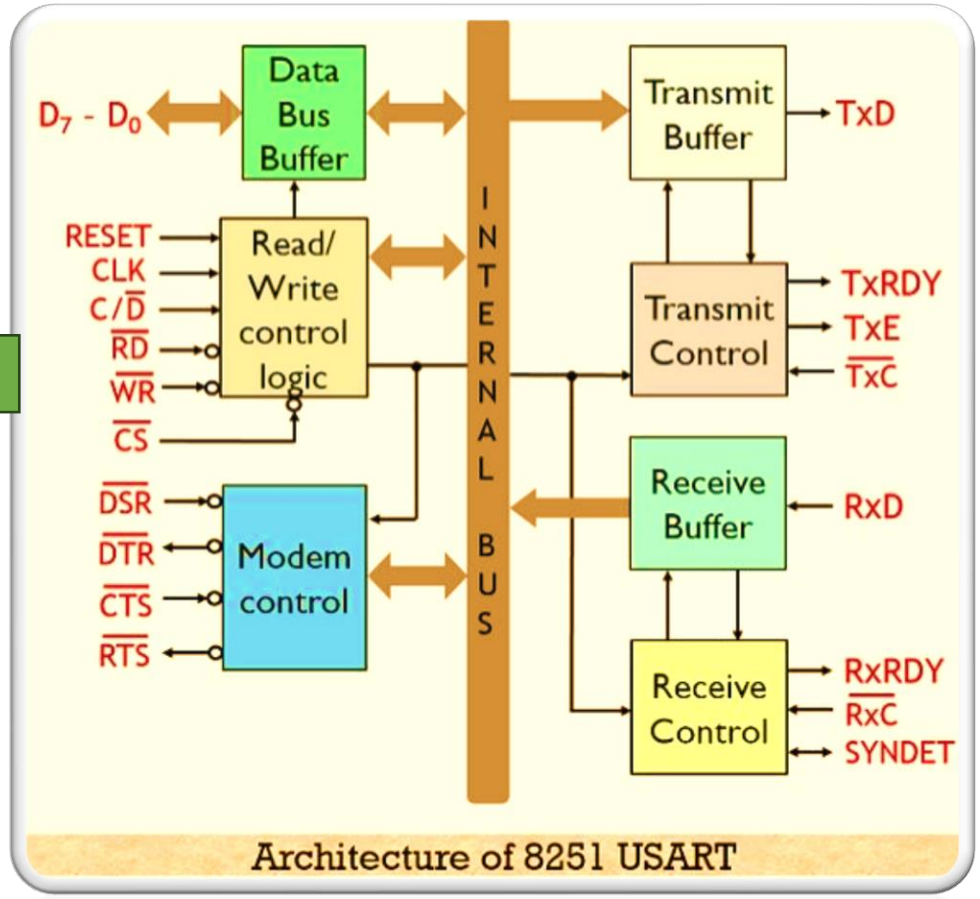


Architecture of 8251 USART

# Internal Architecture of 8251

**There are five different sections in the Internal Architecture of 8251.**

- **Read/ Write control logic**
- **Transmitter**
  - **Transmitter Buffer**
  - **Transmitter control**
- **Receiver**
  - **Receiver Buffer**
  - **Receiver control**
- **Data Bus Buffer**
- **Modem Control.**



# Internal Architecture of 8251

## Read/ Write control logic

It controls the overall working by selecting the operation to be done. The operation selection depends upon input signals as:

| $\overline{CS}$ | C/$\overline{D}$ | $\overline{RD}$ | $\overline{WR}$ | Operation |
|---|---|---|---|---|
| 1 | X | X | X | Invalid |
| 0 | 0 | 0 | 1 | data<br>CPU< ----- 8251 |
| 0 | 0 | 1 | 0 | data<br>CPU ----- > 8251 |
| 0 | 1 | 0 | 1 | Status word<br>CPU < ----------8251 |
| 0 | 1 | 1 | 0 | Control word<br>CPU---------- > 8251 |

## Transmitter

**Transmit buffer –** It is used as parallel to serial converter

**TXD:** If TXD=1→ transmitter will transmit the data.

**Transmit control –** It is used to control the data transmission with the help of following pins:

1. **TXRDY:** Transmitter is ready to transmit data
2. **TXEMPTY:** It means transmitted all the data characters and transmitter is empty now.
3. $\overline{TXC}$ **:** It controls the data transmission rate.

# Internal Architecture of 8251

## Receiver

**Receive buffer –** It acts as a buffer for the received data.

1. **RXD:** An input signal which receives the data.

**Receive control –** It controls the receiving data.

1. **RXRDY:** An input signal indicates that it is ready to receive the data.
2. $\overline{RXC}$ **:** It controls the data transmission rate.
3. **SYNDET/BD:** An input or output terminal. External synchronous mode-input terminal and asynchronous mode-output terminal

## Data Bus Buffer

It helps in interfacing the internal data bus of 8251 to the system data bus.



## Modem Control

**Modulator/demodulator–** A device converts analog signals to digital signals and vice-versa and helps the computers to communicate over telephone lines or cable wires.

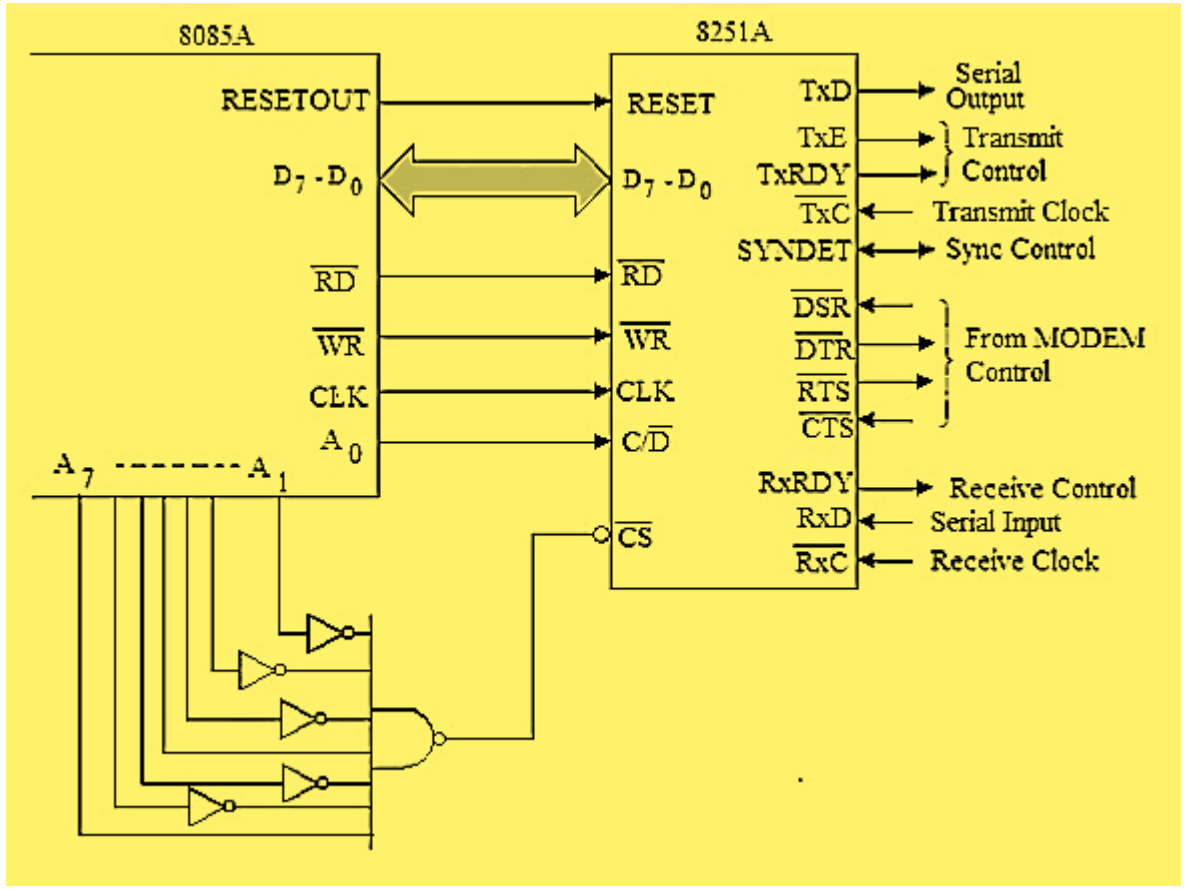
1. $\overline{DSR}$**(Data Set Ready)**→An input signal.
2. $\overline{DTR}$**(Data terminal Ready)**→ is an output signal.
3. $\overline{CTS}$ Input signal which controls the data transmit circuit.
4. $\overline{RTS}$ Output signal which is used to set the status RTS.

# Interfacing 8251 with Microprocessor

Now let us see how 8251 can be interfaced with 8085.

- The 8 data lines $D_{7\text{-}0}$ are connected to the data bus of the microprocessor.
- $\overline{RD}$ and $\overline{WR}$ of the 8251 are also connected with the $\overline{RD}$ and $\overline{WR}$ of microprocessor.
- The 8251 is getting the clock from the CLK OUT pin of 8085.
- RESET is also connected to the RESET OUT pin of the microprocessor.
- The $C/\overline{D}$ pin is used to select either control register or data register. This pin is connected to the A0 pin of 8085.
- The $\overline{CS}$ pin of 8251 is attached to the output of an address decoder circuit. The address decoder uses A7 to A1 lines of the microprocessor.
- $\overline{CS}$ will be enabled when A7 and A4 is at logic 1, and all other lines are at logic 0.

- From the following table, we can see how to read or write data word, read the status word and write control word.




| A0 | $\overline{RD}$ | $\overline{WR}$ | Task | Port Address |
|---|---|---|---|---|
| 0 | 0 | 1 | Read Data Word | 90H |
| 0 | 1 | 0 | Write Data Word | 90H |
| 1 | 0 | 1 | Read Status Word | 91H |
| 1 | 1 | 0 | Write Control Word | 91H |

# Internal Architecture of 8253

The most prominent features of 8253/54 are as follows −

- It has three independent 16-bit down counters.
- It can handle inputs from DC to 10MHz.
- These three counters can be programmed for either binary or BCD count.
- It is compatible with almost all microprocessors.

**There are 4 different sections in the Internal Architecture of 8253.**

- **Read/ Write control logic**
- **Data Bus Buffer**
- **Control word Register(CWR)**
- **Counter**

## 8253 Programmable Interval Timer

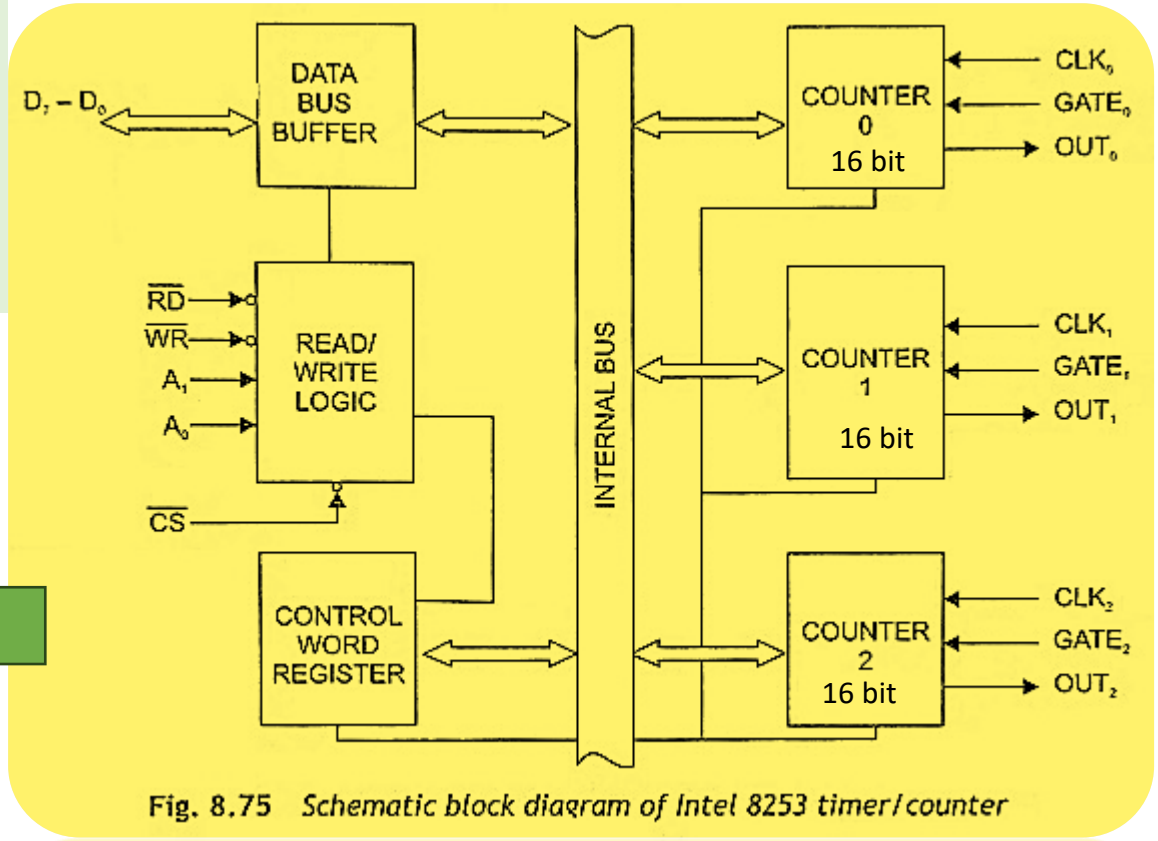



Fig. 8.75 *Schematic block diagram of Intel 8253 timer/counter*

# Internal Architecture of 8253

## Data Bus Buffer

It is a tristate, bi-directional, 8-bit buffer, which is **used to interface the 8253/54 to the system data bus**. It has three basic functions −

- Programming the modes of 8253/54.
- Loading the count registers.
- Reading the count values.

## Counters

- Each counter consists of a single, **16 bit-down counter**, which can be **operated in either binary or BCD**.
- Its input and output is configured by the **selection of modes stored in the control word register**.
- The programmer can read the contents of any of the three counters without disturbing the actual count in process.

# Internal Architecture of 8253

## Read/ Write control logic

- It includes 5 signals, i.e. RD, WR, CS, and the address lines A0 & A1.
- In the peripheral I/O mode, the RD and WR signals are connected to IOR and IOW, respectively.
- In the memory mapped I/O mode, these are connected to MEMR and MEMW.
- Address lines A0 & A1 of the CPU are connected to lines A0 and A1 of the 8253/54, and CS is tied to a decoded address. The control word register and counters are selected according to the signals on lines A0 & A1

| $A_1$ | $A_0$ | Result |
|---|---|---|
| 0 | 0 | Counter 0 |
| 0 | 1 | Counter 1 |
| 1 | 0 | Counter 2 |
| 1 | 1 | Counter Word Register |
| X | X | No Selection |

It is used to write a command word, which **specifies the counter to be used, its mode, and either a read or write operation**.

| $A_1$ | $A_0$ | $RD$ | $WR$ | $CS$ | Result |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | Write Counter 0 |
| 0 | 1 | 1 | 0 | 0 | Write Counter 1 |
| 1 | 0 | 1 | 0 | 0 | Write Counter 2 |
| 1 | 1 | 1 | 0 | 0 | Write control |
| 0 | 0 | 0 | 1 | 0 | Read Counter 0 |
| 0 | 1 | 0 | 1 | 0 | Read Counter 1 |
| 1 | 0 | 0 | 1 | 0 | Read Counter 2 |
| 1 | 1 | 0 | 1 | 0 | No operation |
| X | X | 1 | 1 | 0 | No operation |
| X | X | X | X | 1 | No operation |

# Internal Architecture of 8253

All of the operating modes for the counters are selected by writing bytes to the control register. This is the control word format.

**CWR(Control Word Register)**

| CONTROL BYTE D7 - D0 | | | | | | | |
|---|---|---|---|---|---|---|---|
| **D7** | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| **SC1** | SC0 | RL1 | RL0 | M2 | M1 | M0 | BCP |

## Select Counter

The $SC_0$ and $SC_1$ bits of the control word select a counter.

| $A_1$=$SC_1$ | $A_0$=$SC_0$ | Result |
|---|---|---|
| 0 | 0 | Counter 0 |
| 0 | 1 | Counter 1 |
| 1 | 0 | Counter 2 |
| 1 | 1 | Counter Word Register |
| X | X | No Selection |

## Read/Load

The $RL_0$ and $RL_1$ are used to load/read counts as follows

| $RL_1$ | $RL_0$ | Result |
|---|---|---|
| 0 | 0 | Counter latching |
| 0 | 1 | Read/load LSB only |
| 1 | 0 | Read/load MSB only |
| 1 | 1 | Read/load LSB first later MSB |

## Mode

Mode selecting bits $M_0$, $M_1$ and $M_2$ select any one of six modes as given below:

| $M_2$ | $M_1$ | $M_0$ | Mode |
|---|---|---|---|
| 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 |

0 Binary counter (16 bits)
1 Binary coded decimal (BCD) counter (4 decades)

# Modes of operation of 8253

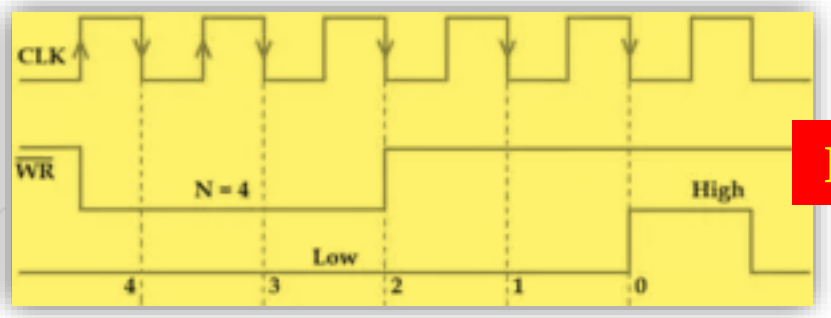
**Mode 0 – Interrupt on Terminal Count**

•It is used to generate an interrupt to the microprocessor after a certain interval.

•Initially the output is low after the mode is set. The output remains LOW after the count value is loaded into the counter.

•The process of decrementing the counter continues till the terminal count is reached, i.e., the count become zero and the output goes HIGH and will remain high until it reloads a new count.

•The GATE signal is high for normal counting. When GATE goes low, counting is terminated and the current count is latched till the GATE goes high again.



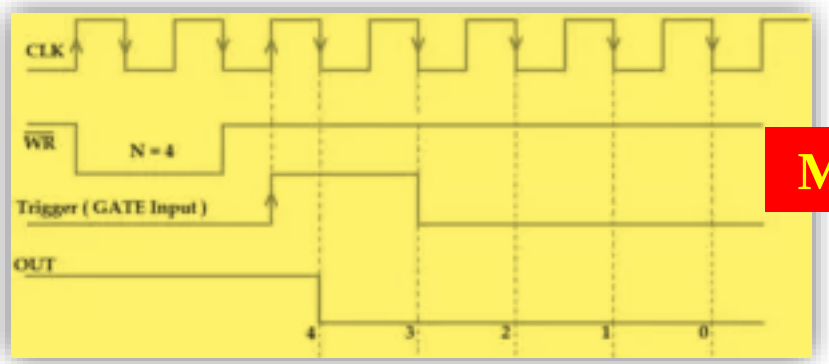

Mode 0

**Mode 1 – Programmable One Shot**

•It can be used as a mono stable multi-vibrator.

•The gate input is used as a trigger input in this mode.

•The output remains high until the count is loaded and a trigger is applied.

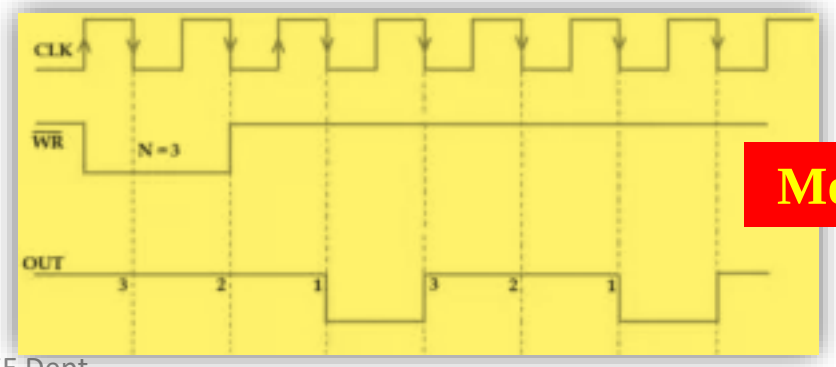



Mode 1

**Mode 2 – Rate Generator**

•The output is normally high after initialization.

•Whenever the count becomes zero, another low pulse is generated at the output and the counter will be reloaded.




Mode 2

# Modes of operation of 8253

**Mode 3 – Square Wave Generator**

•This mode is similar to Mode 2 except the output remains low for half of the timer period and high for the other half of the period.

**Mode 4 − Software Triggered Mode**

•In this mode, the output will remain high until the timer has counted to zero, at which point the output will pulse low and then go high again.

•The count is latched when the GATE signal goes LOW.

•On the terminal count, the output goes low for one clock cycle then goes HIGH. This low pulse can be used as a strobe.

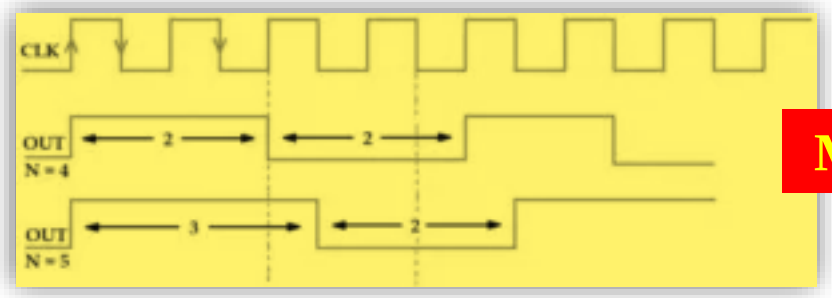
**Mode 5 – Hardware Triggered Mode**

•This mode generates a strobe in response to an externally generated signal.

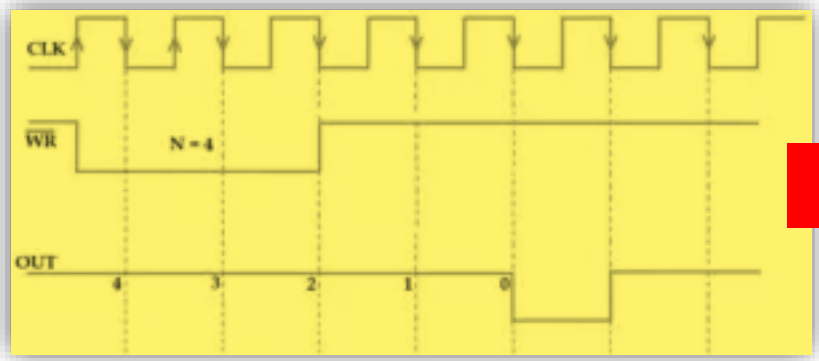
•This mode is similar to mode 4 except that the counting is initiated by a signal at the gate input, which means it is hardware triggered instead of software triggered.

•After it is initialized, the output goes high.

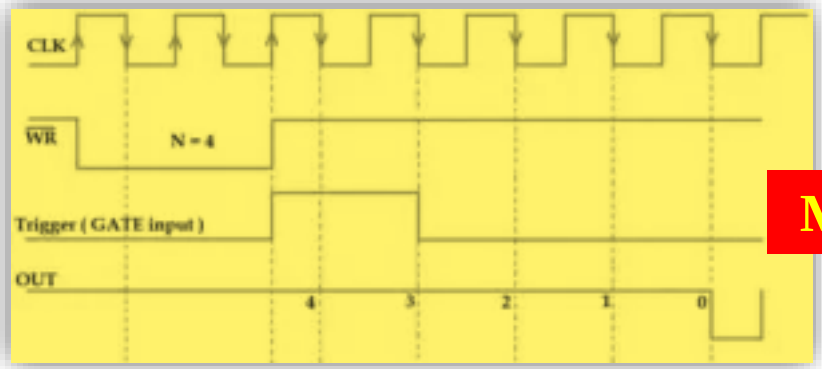
•When the terminal count is reached, the output goes low for one clock cycle.




Mode 3




Mode 4




Mode 5