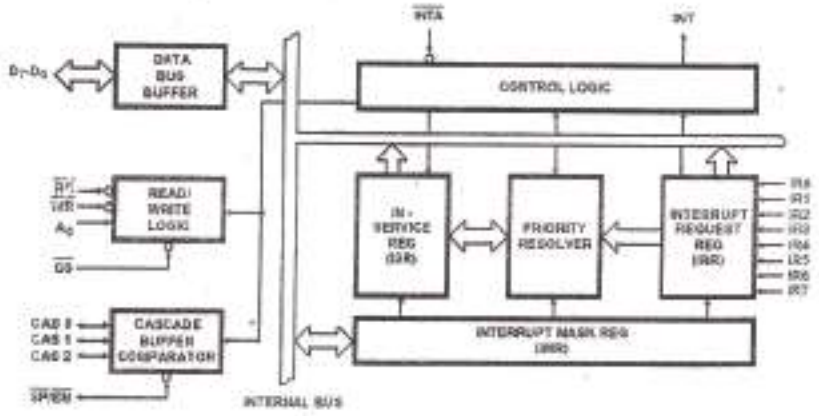
8259 --- PROGRAMMABLE INTERRUPT CONTROLLER

- PIC 8259 is a **Programmable Interrupt Controller** that can work with **8085, 8086** etc.
- A **single 8259** can handle **8 interrupts** while a **cascaded** configuration of 1 master 8259 and 8 slave 8259s can handle upto **64 interrupts.**
- 8259 can **handle edge** as well as **level triggered** interrupts.
- 8259 has a **flexible priority** structure.
- In 8259 interrupts can be **masked** individually.
- The **Vector address** of the interrupts is **programmable.**
- Status of interrupts (pending, in-service, masked) can be easily read by the µP.

ARCHITECTURE OF 8259



The architecture of 8259 can be divided into the following parts:

1) Interrupt Request Register (IRR)

- 8259 has **8 interrupt** input lines **IR_7 ... IR_0**.
- The IRR is an **8-bit register** having **one bit** for **each** of the **interrupt** lines.
- When an **interrupt request** occurs on any of these lines, the **corresponding bit** is **set** in the Interrupt Request Register (**IRR**).

2) In-Service Register (InSR)

- It is an **8-bit** register, which **stores** the **level** of the Interrupt Request, which is **currently being serviced**.

3) Interrupt Mask Register (IMR)

- It is an **8-bit** register, which stores the **masking pattern** for the interrupts of 8259. It stores **one bit per interrupt level**.

4) Priority Resolver

- It **examines** the **IRR**, **InSR**, and **IMR** and determines which interrupt is of **highest priority** and should be sent to the µP.

5) Control Logic

- It has **INT output** signal **connected to** the **INTR** of the µP, to **send** the **Interrupt** to the µP.
- It also has the **$\overline{\text{INTA}}$ input** signal **connected to** the **$\overline{\text{INTA}}$** of the µP, to **receive** the interrupt **acknowledge.**
- It is also used to control the remaining blocks.

6) Data Bus Buffer

- It is a bi-directional buffer used to **interface** the internal **data bus** of 8259 with the external (system) data bus.

7) Read/Write Logic

- It is used to accept the $\overline{RD}$, $\overline{WR}$, A_0 and $\overline{CS}$ signal.
- It also holds the Initialization Command Words (ICW's) and the Operational Command Words (OCW's). ICWs and OCWs are explained later.)

8) Cascade Buffer / Comparator

- It is used in **cascaded mode** of operation.
- It has two components:

i. CAS_2, CAS_1, CAS_0 lines:

- These lines are **output for the master, input for the slave.**
- The **Master sends** the **address of** the **slave** on these lines (hence output).
- The **Salves read** the **address** on these lines (hence input).
- As there are 8 interrupt levels for the Master, there are **3 CAS lines (Θ $2^3 = 8$).**

ii. $\overline{SP}/\overline{EN}$ (Slave Program/Master Enable):

- In **Buffered Mode**, it **functions** as the **$\overline{EN}$ line** and is used to **enable** the **buffer.**
- In **Non buffered mode**, it **functions** as the **$\overline{SP}$ output line.**
- **For Master** 8259 **$\overline{SP}$** should be **high**, and **for** the **Slave $\overline{SP}$** should be **low.**

PRIORITY MODES OF 8259

❖ Fully Nested Mode (FNM)

- It is the default mode of 8259.
- It is a fixed priority mode.
  IR_0 has the highest priority and **IR_7** has the **lowest** priority.

❖ Special Fully Nested Mode (SFNM)

- This mode can be **used for** the **Master 8259 in** a **cascaded** configuration.
- Its **priority structure** is fixed and is the **same as FNM** (IR_0 highest and IR_7 lowest).
- **Additionally**, in SFNM, the **Master would recognize a higher priority interrupt form a slave** *whose another interrupt is currently being serviced*. This is **possible only in SFNM**.

❖ Rotating Priority Modes

There are **two** rotating priority modes:
Automatic Rotation and Specific Rotation

❖ Automatic Rotation Mode

- This is a rotating priority mode.
- It is **preferred** when **several interrupt** sources are of **equal priority**.
- In this mode, **after** a device receives **service**, it **gets** the **lowest priority**. **All other priorities rotate subsequently**.

❖ Specific Rotation Mode

- It is also a rotating priority mode, **but here** the **user can select** any **IR level for lowest priority**, and thus for all other priorities.
  Eg: If IR_3 is has just been serviced, it will get the lowest priority as shown below:

IR_0	IR_1	IR_2	IR_3	IR_4	IR_5	IR_6	IR_7
5	6	7	0	1	2	3	4

Other Modes of 8259

❖ Special Mask Mode (SMM)

- **Usually** 8259 disables **interrupt requests lower or equal** to the interrupt, which is **currently** in service.
- **In SMM** 8259 permits **interrupts** of **all levels** (lower or higher) **except** the one **currently** in service.

❖ Poll Mode

- Here the **INT line** of 8259 is **disabled**.
- The **µP gives** the **Poll command** to the 8259 using OCW3.
- In **return**, **8259 provides** the **Poll Word** to the µP.
- The Poll Word as shown, **indicates** the **highest priority interrupt**, which requires service.
- Thereafter the **µP services the interrupt**.
- It is **preferred** when:
  i. **Subroutine** is **common** for **several** interrupt levels.
  ii. To expand number of interrupt levels **more than 64**.

Advantage
1) µP is not disturbed

Disadvantage
1) If polling interval is too large, interrupts will be serviced after a long time.
2) If polling interval is too small, a lot of time will be wasted in unnecessary polls.

❖ Buffered Mode

- In this mode **SP/EN** becomes **low** during **INTA** cycle.
- This signal is used to enable the buffer.

End Of Interrupt (EOI)

- When the **µP responds** to an interrupt request by **sending the first INTA** signal, the **8259 sets** the **corresponding bit** in the In Service Register (**InSR**).
  This **begins** the **service** of the interrupt.
- **When this bit** in the In Service Register is **cleared, it is called** as **End of Interrupt (EOI)**.
- There are three ways/commands for EOI:

i. Automatic EOI (AEOI)

- In this mode, **no command** is **necessary**.
- **During** the **second INTA** cycle, the **corresponding bit** in the **InSR** is **reset**.

ii. Non-Specific EOI

- This **command** is **sent** to **8259** at the **end** of a **service routine**.
- It would **clear** the **bit** of the currently serviced interrupt **in** the **InSR**.

iii. Specific EOI

- This **command** is also **sent** to **8259** at the **end** of a **service routine**.
- This command **specifies which InSR bit** is **to be reset**.

- If the **µP** is **interrupted by** the **slave**, then **2 EOI** commands have to be **sent**; **one for** the **master**, and **one for** the **slave**.

Default IR_7 Routine

- The IR inputs of 8259 can be level or edge triggered.
- The **interrupt request signal should be maintained high, until** the **falling edge** of the **first INTA** pulse.
  Otherwise, it is treated as an **invalid** request (**noise**).
- **When** it is an **invalid** request, **8259 by default sends** the **IR_7 level** to the µP.
- The **µP** will **execute** the **ISR corresponding** to **IR_7**.
- **In** this **ISR**, it **checks** whether it is a **noise** by **checking** the **D_7** bit of the **InSR**.
- **If** this bit is **Zero**, it will **not execute** the ISR.

3) **HRQ** (Hold Request)
- This **output** pin is used by the DMAC to **request the µP** to release the system bus.
- It is **connected** to the **HOLD** pin of the µP.

4) **HLDA** (Hold Acknowledge)
- This **input pin** is used by the µP **to inform the DMAC** that it es released to system bus.
- It is **connected** to the **HLDA** pin of the µP.

5) **AEN** (Address Enable)
- This is an **output** pin from the DMAC.
- When this pin is high, **it disconnects the µP from the system bus.**
- It is **also** used to **enable** the **external latch.**

6) **ADSTB** (Address Strobe)
- It is an **output** signal.
- It is used to **strobe** the higher order **address** byte **into the latch.**

7) **DB_7 – DB_0** (Data Bus)
- These are **8 bi-directional data lines** used to connect the internal data bus of 8237 with the external (system) data bus.
- In **idle cycle, µP writes or reads** from 8237 **using this bus.**
- In **active cycle**, this bus **carries** the **8 higher order bits** of the 16 bit **address** (the other 8 bits being in the **A_7–A_0**).
- During **memory-to-memory transfer**, this bus **carries** the **data** byte to be transferred.

8) **A_7 – A_4** (Address bits)
- These are **4 output address lines.**
- In **active cycle**, these lines **carry the A_7–A_4 bits of the address** at which the transfer is to be done.
  As this address is generated by the 8237, these are output lines.

9) **A_3 – A_0** (Address bits)
- These are **4 bi-directional address lines.**
- In **idle cycle, µP sends** the **address A_3–A_0**, to select one of its registers.
  Since µP sends the address to 8237, these are input lines.
- In **active cycle**, these lines **carry the A_3–A_0 bits of the address** at which the transfer is to be done.
  As this address is generated by the 8237, these are output lines.

10) **IOR, IOW**
- These are **bi-directional control lines.**
- During **idle state**, the **µP issues** these signals **to read from or write into the 8237** (as the DMAC itself is an I/O device w.r.t. the µP).
- During **active state**, the **DMAC issues** these signals **to read from or write into an I/O Device.**

11) **MEMR, MEMW**
- These are **output control lines.**
- During **active state**, the **DMAC issues** these signals **to read from or write into the Memory.**

12) **EOP**

- This is a **bi-directional signal** indicating the **end of DMA process.**
- During **active cycle, after each byte** is transferred through DMA, the **Count Register decrements** by 1.
- When the **Terminal Count** is reached, it means that **all** the required **bytes are transferred.**
- At this point, the **DMAC issues** this signal and the **DMA operation is terminated.** Hence, it is an output signal.
- During the course of the transfer, the **DMA operation can be explicitly terminated by giving this signal** externally to the DMAC. Hence, it is also an input signal.

13) **CLK**

- This is a **clock-input** signal for the DMAC.
- It is usually connected to the system clock.

14) **RESET**

- This is a **reset-input** signal for the DMAC.
- This signal **clears the internal registers** of the DMAC and causes it to enter **Idle State.**

15) **READY**

- This is an **input** signal to the DMAC.
- It is used to **synchronize** the DMAC with "**Slower**" **peripherals.**
- **If** a peripheral is slow (**not ready**), it put a **low** on this line causing the **DMAC** to **wait.**

Only after the DMAC finds a high on this pin, it executes the DMA operation.