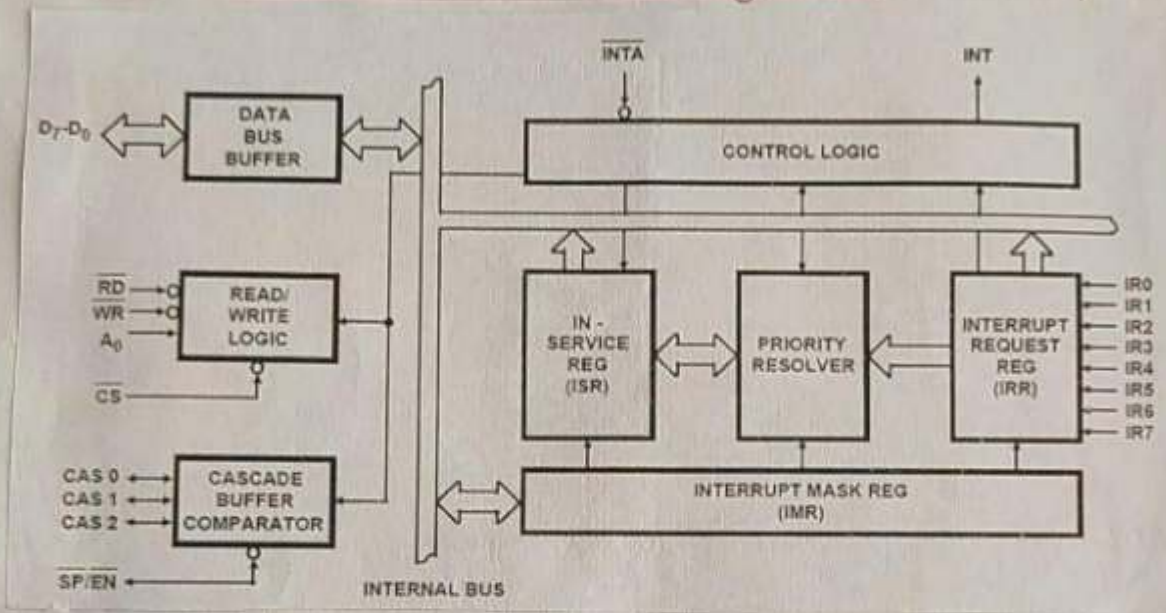
## 8259 Architecture / Block Diagram :




The main components of the architecture are as follows :

Control logic:
8259 takes interrupts on $IR_0$ - $IR_7$ and gives the interrupt to µP on INT pin. INT pin of 8259 is connected to INTR pin of µP. On response µP gives $\overline{INTA}$ signal which is received by $\overline{INTA}$ of 8259. (2 pulses)

On the first $\overline{INTA}$ pulse, 8259 will determine the vector number. With the second $\overline{INTA}$ 8259 will send the vector no. to the µP through the 8 bit data bus.

The control logic is also used to control the remaining blocks by sending internal control signals.

Read/Write logic:

It is used to accept the $\overline{RD}$, $\overline{WR}$, $A_0$ and CS signal.

$\overline{IOR}$ and $\overline{IOW}$ of µP is connected to $\overline{RD}$ and $\overline{WR}$ of 8259.

It is also used to hold some of the Initialization Command Words (ICW's) and the Operational Command Words (OCW's)

$A_0$ is used to identify the commands.

$\overline{CS}$ is used to select the 8259 chip.

Note:- When 8259 gets an interrupt, the interrupt is not directly sent to µP. 8259 checks whether the interrupt should be sent to µP or be discarded. For this purpose it has 4 registers:

- Interrupt Request Register (IRR)
- Interrupt Mask Register (IMR)
- In-service Register (InSR)
- Priority Resolver

Priority resolver takes input from the 3 registers and decides whether the interrupt should be sent to µP or held back or discarded.

Semester: IV

## Interrupt Request Register (IRR)

8259 has 8 interrupt i/p lines $IR_7 \ldots IR_0$
These lines are active high. By default, when no interrupt occurs these lines are at logic 0.
When an interrupt occurs the corresponding line becomes logic 1. and the corresponding bit in the IRR (8 bit register having one bit for each of the interrupt lines) is set.
This bit reminds the processor that the interrupt is pending ie the interrupt has occured but not yet serviced.
This bit will become 0 when µP acknowledges this interrupt and responds by sending 1st $\overline{INTA}$.

## Interrupt Mask Register (IMR)

It is an 8 bit register, which stores the masking pattern of the interrupts of 8259.
By default all 8 bits are zero, means no interrupt is masked.
The interrupts can be masked by the programmer by the OCW1 command.
To mask a bit, the corresponding bit must be made a 1.
Now, even if the interrupt occurs, it will not be sent to the µP, irrespective of its priority.

## In-Service Register (InSR)

It is an 8-bit register which tells which interrupt is in service.

When µP sends the 1st $\overline{INTA}$ signal the corresponding bit becomes 1 in the InSR indicating that from now on, this interrupt is being serviced.

This bit must stay 1 till the end of the ISR.

During this ISR, if any other interrupt occurs, the priority resolver compares its level (in IRR) with the interrupt which is being currently serviced (in InSR).

Only if the new interrupt is of higher priority than the one currently being serviced, will 8259 send the new interrupt to µP.

So this bit must be cleared by the time the ISR is completed else 8259 will forever be under the impression that the ISR is still being serviced even though the µP has finished and moved on.

To clear this bit from InSR, there are 2 options

- In normal EOI mode (default mode) the programmer must give an End of Interrupt (EOI) command at the end of ISR.
- In auto EOI mode, this bit is cleared automatically in the 2nd $\overline{INTA}$ for 8086 systems.

Assume µP is servicing IR0 (Highest priority). µP has finished the ISR and there is no EOI command. µP will go to main program and continue but 8259 stays under the impression that still ISR of IR0 is going on and hence no other interrupts will be serviced.

So it is necessary to give EOI command at the end of ISR.

IRR gives reliability that the interrupt is not lost. How?

Suppose IRR is not there, then priority resolver will directly look at the pins. Now if IR0 and IR1 both are 1, IR0 will be serviced first as its priority is higher.

Suppose the ISR of IR0 is taking too long. The device which has sent the request on IR1 may not give active high pulse (logic 1) for that long time. It may give just one high pulse and again it goes to level 0.

Once the ISR of IR0 is served, priority resolver will check the pins again and will find that IR1 is 0. It will not serve IR1 and thus the interrupt is lost.

But with IRR register this will not happen as when an interrupt occurs, that bit will become 1 and even if now the device goes to level 0, that bit remains 1. It will become 0 when its ISR is served. So IRR gives the reliability that interrupts will not be lost.

Semester: IV

## Priority Resolver:

It examines the IRR, InSR and INR to know which interrupts is of highest priority and should be sent to µP.

It checks IRR to know which interrupts have occured, INR to know which interrupts are masked and InSR to know which interrupt is in service.

**Note:-** If the interrupt that has occured is of highest priority amongst those that have occured, is unmasked and is higher priority than the one currently being serviced, only then the interrupt will be sent to the µP. So it is a two level priority mechanism.

For eg. Let's say that an interrupt occured on $IR_7$ and currently $IR_4$ is being served, then priority resolver will keep $IR_7$ as pending interrupt.

Let's say there is a request on $IR_0$ and currently $IR_4$ is being served. Since $IR_0$ has higher priority than $IR_4$, µP will stop stop ISR of $IR_4$ and serve $IR_0$ first and then continue with $IR_4$.

Suppose

InSR → 5th bit is a 1 ($IR_4$)
INR → 4th bit is a 1 ($IR_3$)
IRR → $IR_3$ & $IR_5$ is 1.

Priority Resolver first looks at IRR. $IR_3$ and $IR_5$ are 1. Since $IR_3$ is of higher priority, $IR_3$ is selected.

Now it looks in IMR, $IR_3$ is masked. So $IR_3$ will not be serviced.

Now it again checks IRR.

$IR_5$ will be selected.

$IR_5$ is not masked.

Now it looks at InSR: $IR_4$ is being served. but $IR_5$ is of lower prority. So $IR_5$ will become a pending interrupt.

Meanwhile suppose $IR_2$ occurs, PR will check IMR → not masked, check InSR (2 is of higher priority than 4)

∴ InSR 2nd bit will become 1 ($IR_2$)

5th bit will also remain a 1 ($IR_4$). µP will finish $IR_2$ first and then finish $IR_4$.

## Cascade Buffer Comparator

Cascading means more than 1 8259 is connected to µP.

1 Master & 1 Slave is cascading.

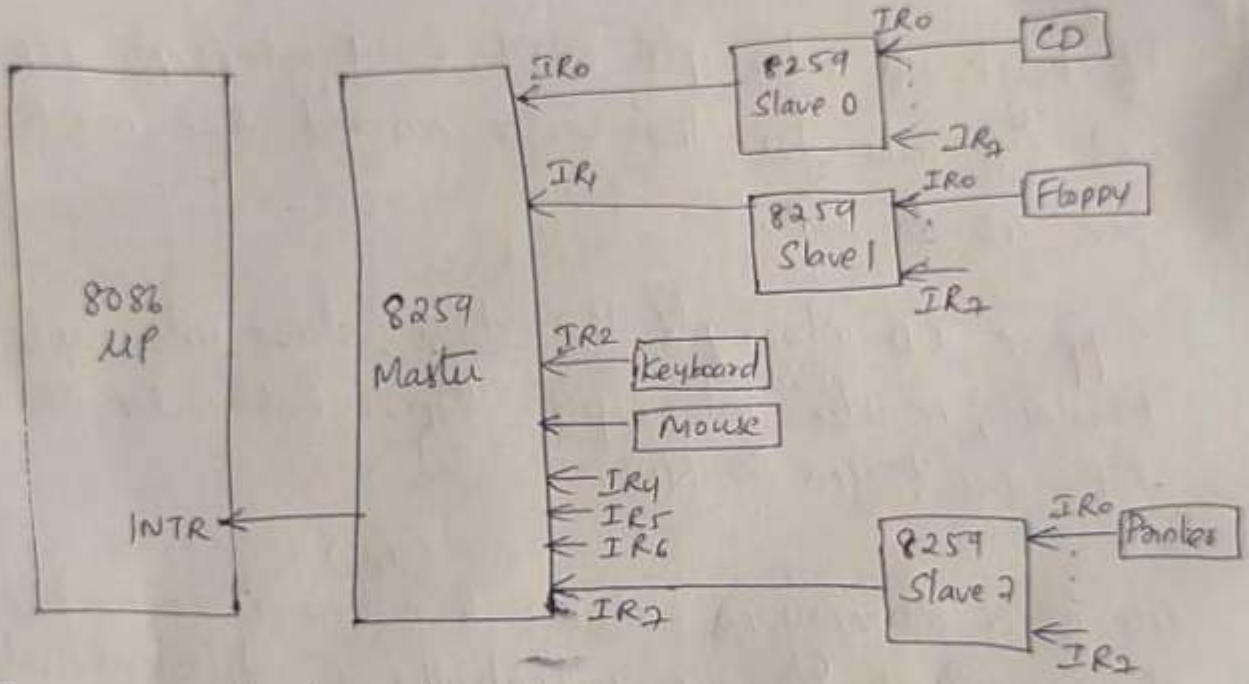
1 Master & 8 Slave is also cascading

Cascading is done to increase the no. of interrupts µP can handle.

Let's say slaves are connected to $IR_0$, $IR_1$ and $IR_7$ pins of master.

On $IR_2$ - keyboard, $IR_3$ - mouse is connected, $IR_7$ - printer.

$IR_4$, $IR_5$, $IR_6$ is free




If CD wants to interrupt µP, CD will interrupt slave 0, slave will interrupt the Master and Master interrupts the µP.

Remember, initialization of 8259 is compulsory.

So whom should we initialize? Master or Slave?

→ ALL

Every 8259 has to be individually initalized.

Why?

Suppose only the master is initialized. During initialization vector no. of $IR_0$ is given and the rest is taken in sequence.

Suppose the vector no. of $IR_0$ is 40, then $IR_1$ - 41, $IR_2$ - 42 and so on.

↑
Keyboard.

If keyboard interrupts 8259, 8259 interrupts µP. and µP asks for the vector no and then master will give 42 to µP.

Suppose CD interrupts the slave, slave interrupts the master, master interrupts µP. What vector no should µP give? 40 ? NO.

40 is a dummy no because this line ($IR_0$) is connected to a slave. which means potentially 8 different interrupts can come on this line. How can 8 different interrupts have the same vector no ?
So for this line alone, there should be 8 different vector numbers which are with the slave.
That is why every 8259 has to be initialized separately.

Suppose CD interrupts slave, slave interrupts master, master interrupts µP and µP asks for vector no. Who should give vector no? Master or Slave ?

Slave

So master needs to know to which lines slaves are connected. so that if interrupts occur on these lines 8259 will interrupt µP but when µP asks vector no, slave will give the vector no. on the data bus.

Note:- All slaves are connected on the data bus.

A slave needs to know on which line it is connected to master (during initialization)

Here slave [0] is connected to $IR_0$
[2] is connected to $IR_1$
[7] is connected to $IR_7$.

This is called slave identification no.

So,

⇒ Every 8259 is initialized.
⇒ Every 8259 needs to be told the vector no.
⇒ Every master needs to know to which all lines slaves are connected.
⇒ Every slave needs to know its identification no.

Suppose CD ($IR_0$ ← Slave 0) interrupted the µP (through master) Suppose at the same time printer ($IR_0$ - Slave 7) generates an interrupt. Who is selected?

CD - $IR_0$ of master
Printer - $IR_7$ of master
} $IR_0$ selected (highest priority) ie, CD is selected first. and send to INTR

When µP receives the interrupt first $\overline{INTA}$ will occur.
Who will µP give the $\overline{INTA}$? to master or to slave?
to both (to master and to all slaves).

So now, all 8259's (master & slaves) know that some interrupt is going to be serviced.
Who knows which interrupt is actually going to be serviced? — Master
So this is when, master will inform the correct slave (CD) using CAS lines (000).

Very soon, µP will give the 2nd $\overline{INTA}$ which also goes to all 8259s, but now only the selected slave will give the vector no.

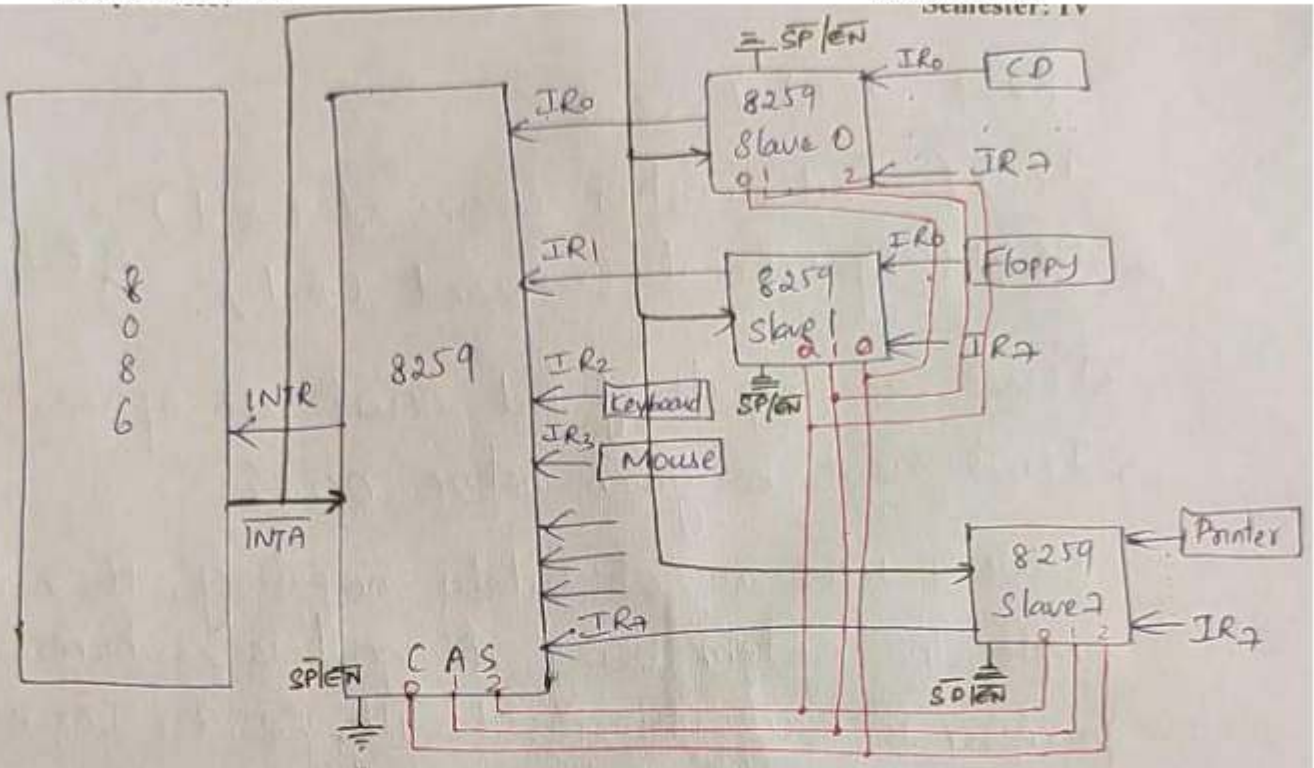
So before the 2nd $\overline{INTA}$ comes, but after the 1st $\overline{INTA}$ µP inform through CAS lines which slave has been selected.

So who uses the CAS lines to tell who, what, when and why?

- who → µP
- what → inform slave
- which → which slave is selected
- when → blw first and second $\overline{INTA}$
- why → so that it gives vector no. to µP

**Academic Year: 2022-23** **Semester: IV**

**Class/Branch: SE** **Subject: MP**




Note:- CAS lines are unidirectional ie, they go from master to slave, but in the architecture CAS lines are shown as bidirectional. This is because 8259 can act as a master 8259 or slave 8259 when they are cascaded.

How does 8259 know whether it is a master or a slave?

$\overline{SP}/\overline{EN}$

$\overline{SP}$ means (Slave Program/Slave Progress)

If $\overline{SP} = 0 \rightarrow$ 8259 is a slave

$\overline{SP} = 1 \rightarrow$ 8259 is a master.

## 8259 ICW & OCW

ICW ( Initialization Command Word ) } 8 bit commands
OCW ( Operational Command Word ). }

→ ICW's are compulsory but OCW's are optional.
→ ICW's are to be given before OCW's.

ICW1 & ICW2 are absolutely compulsory. No matter how simple the system being designed is, whichever processor is being interfaced with 8259. ICW1 and $ICW_2$ have to be given.

→ ICW3 is given only if 8259 is cascaded.
→ $ICW_4$ is used if needed.
→ ICW's have to be given in order and cannot be repeated individually.
→ OCWs are mainly used to alter the masking status and the operation modes of 8259