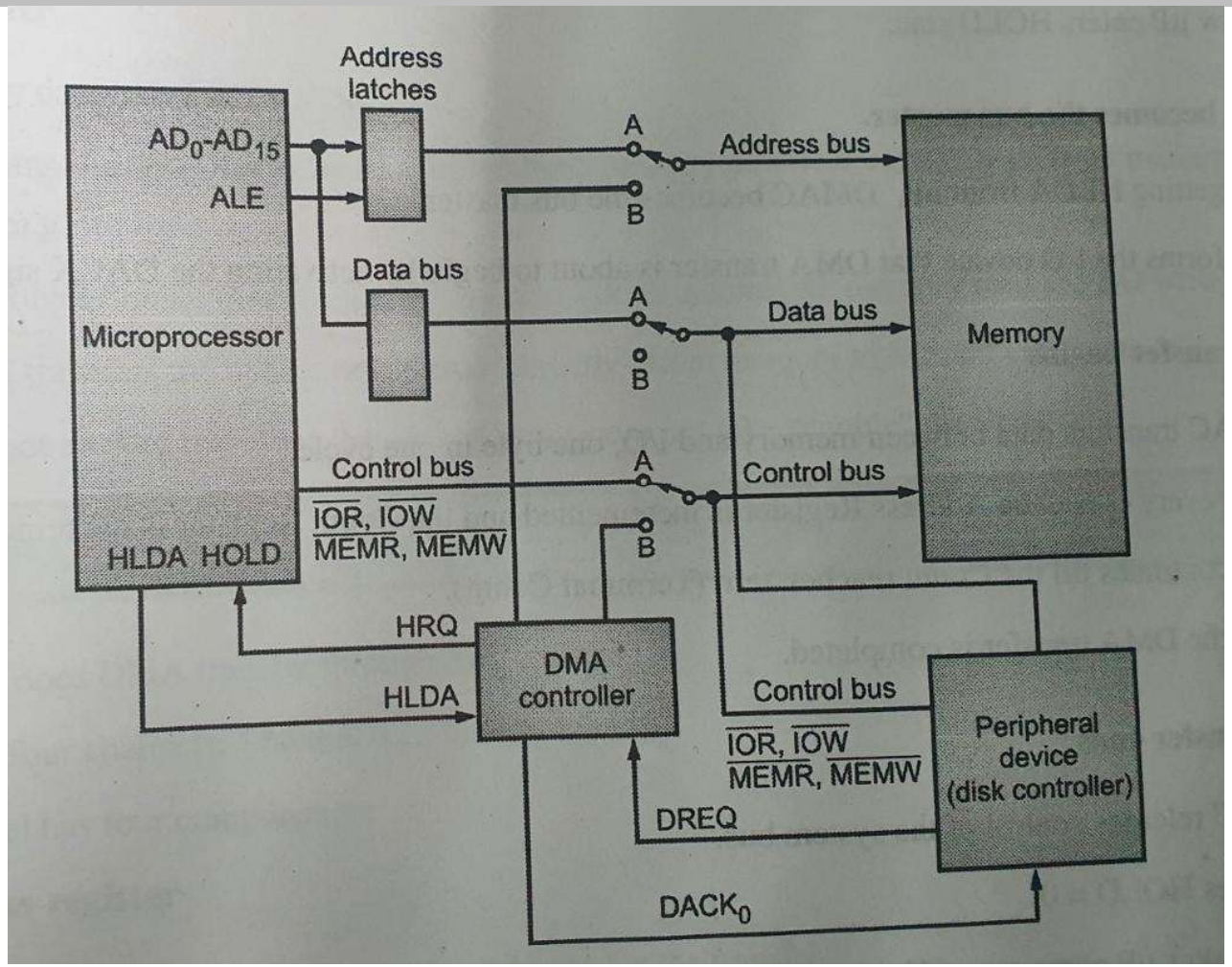
MODULE 4

Peripheral Chips

- Concept of DMA and study of 8237 (DMAC)
- Concept of parrel peripheral interface and study of 8255 (PPI)
- Interrupt structure of 8086 and study of 8259 (PIC)

8257 DMA CONTROLLER

Concept of DMA controller :



8257 DMA CONTROLLER

Advantage of DMA:

1. Hardware based : massive speed
2. Direct transfer : without involvement of processor

DMA Channels :

- DMAC does DMA transfer through its channels.
- DMAC has four channels : channel 0.....channel 3
- Each channel has four components:
  - Address register
  - Count register
  - DERQ
  - DACK

8257 DMA CONTROLLER

Steps for performing a DMA transfer :

1. µp initializes the DMAC
   - This is done by giving the starting address and the number of bytes to be transferred.
2. I/O device requests the DMAC
   - I/O device makes the DREQ signal = 1.
3. DMAC requests the uP for control of the system bus
   - DMAC makes HOLD = 1
4. Microprocessor releases control of the system bus
   - µp finishes the current machine cycle and releases control of the system bus.
   - µp informs the DMAC that the bus is released by making HLDA = 1. Now µp enters HOLD state.
5. DMAC becomes the bus master.
   - On getting HLDA from µP, DMAC becomes the bus master.
   - It informs the I/O device that DMA transfer is about to begin by activating the DACK signal
6. DMA Transfer begins
   - DMAC transfers data between memory and I/O, one byte in one cycle.
   - After every cycle, the Address Register is incremented and the Count Register is decremented
   - This continues till the Count reaches zero (Terminal Count). Now the DMA transfer is completed.
7. 7. DMA Transfer ends
   - DMAC releases control of the system bus. & It makes HOLD = 0.
   - This makes up come out of Hold state and once again become the bus master. µP takes control of the system bus and continues its operation.

8257 DMA CONTROLLER

Pin diagram :

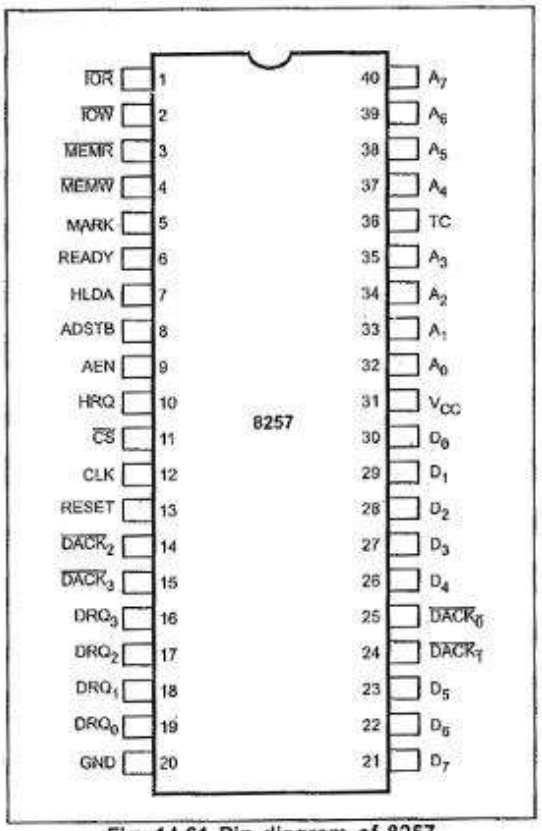
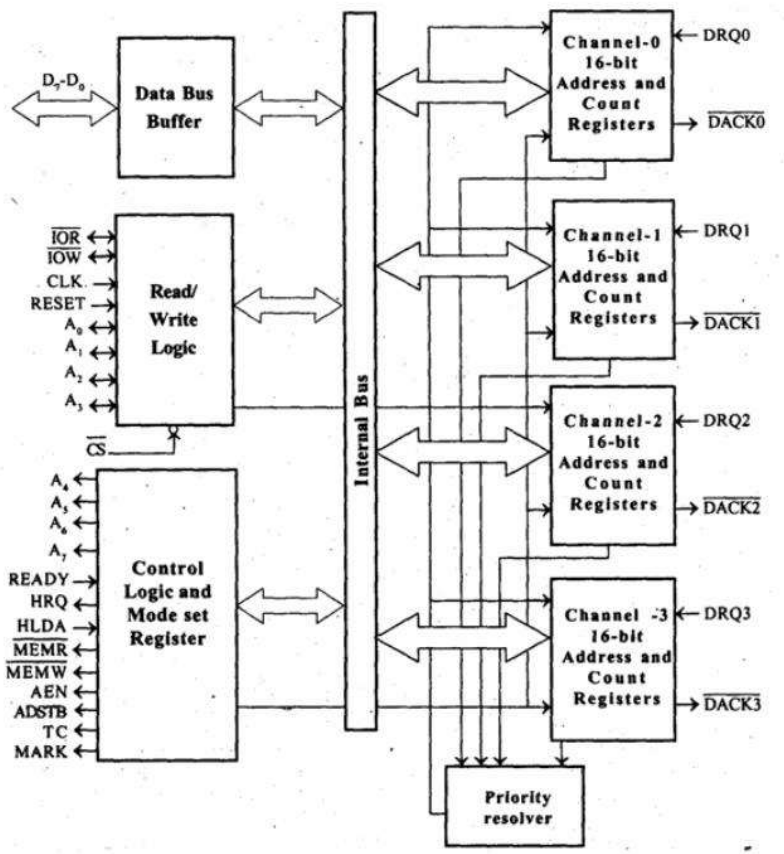


Fig. 14.61 Pin diagram of 8257

8257 DMA CONTROLLER

8257 DMAC ARCHITECTURE :



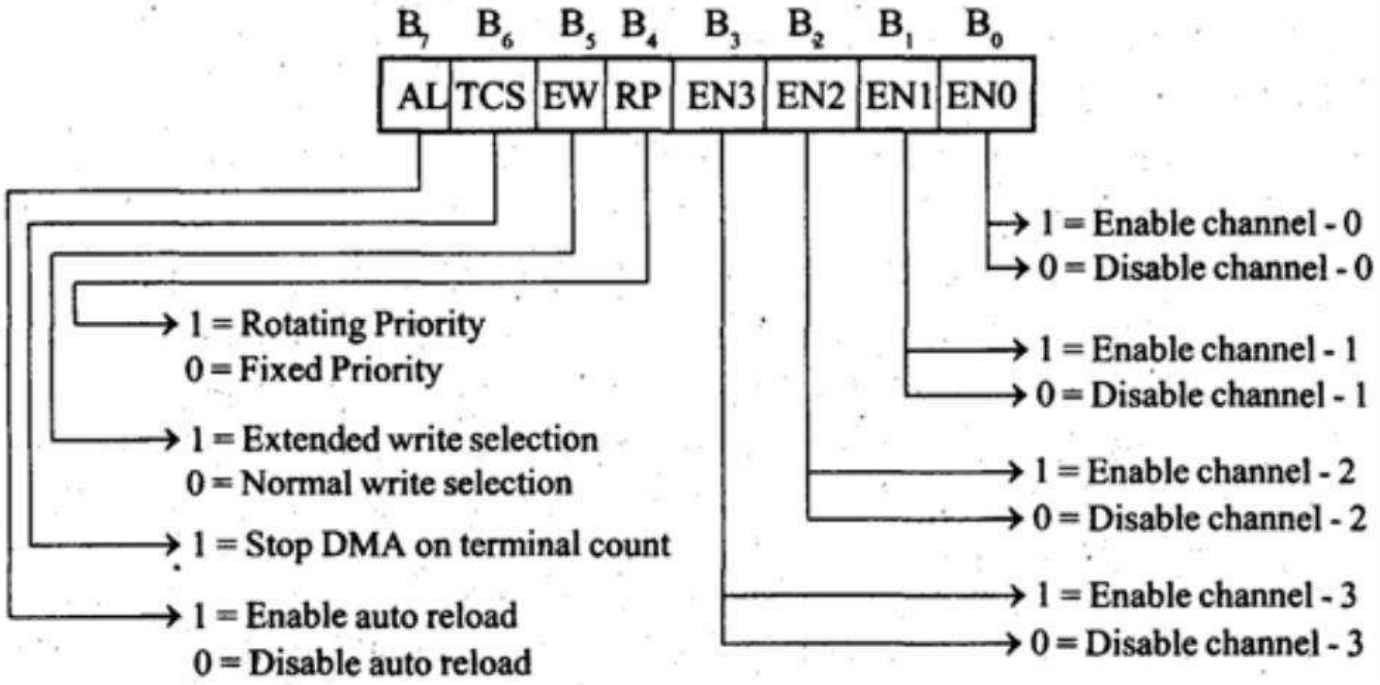
8257 DMA CONTROLLER

Registers on 8257

Register	Address			
	A_3	A_2	A_1	A_0
Channel-0 DMA address register	0	0	0	0
Channel-0 Count register	0	0	0	1
Channel-1 DMA address register	0	0	1	0
Channel-1 Count register	0	0	1	1
Channel-2 DMA address register	0	1	0	0
Channel-2 Count register	0	1	0	1
Channel-3 DMA address register	0	1	1	0
Channel-3 Count register	0	1	1	1
Mode set register (Write only)	1	0	0	0
Status register (Read only)	1	0	0	0

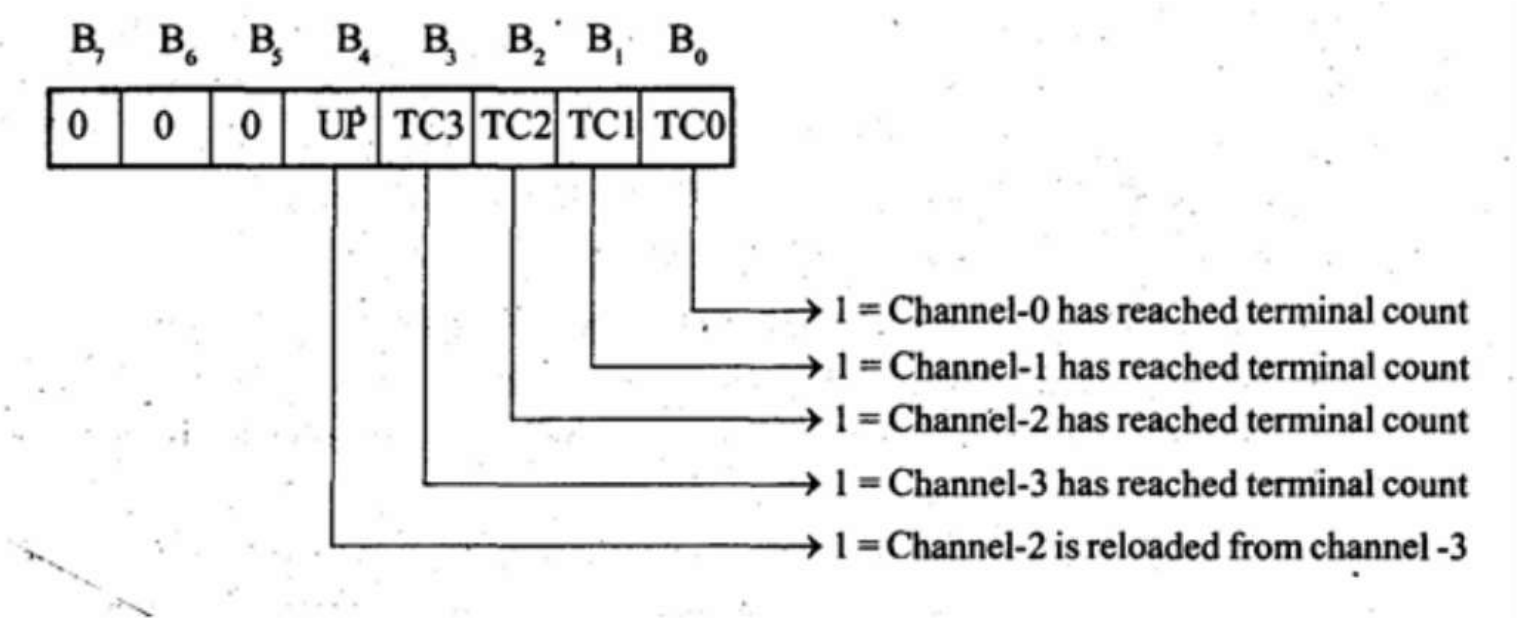
8257 DMA CONTROLLER

Mode Set register



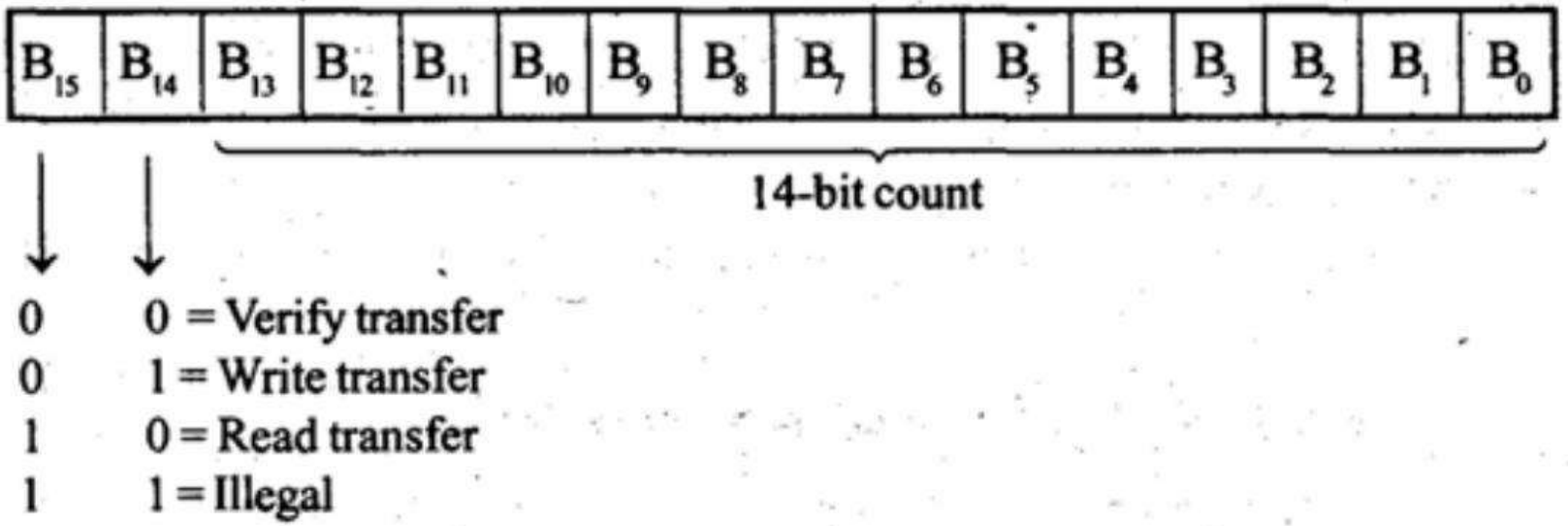
8257 DMA CONTROLLER

Status Register



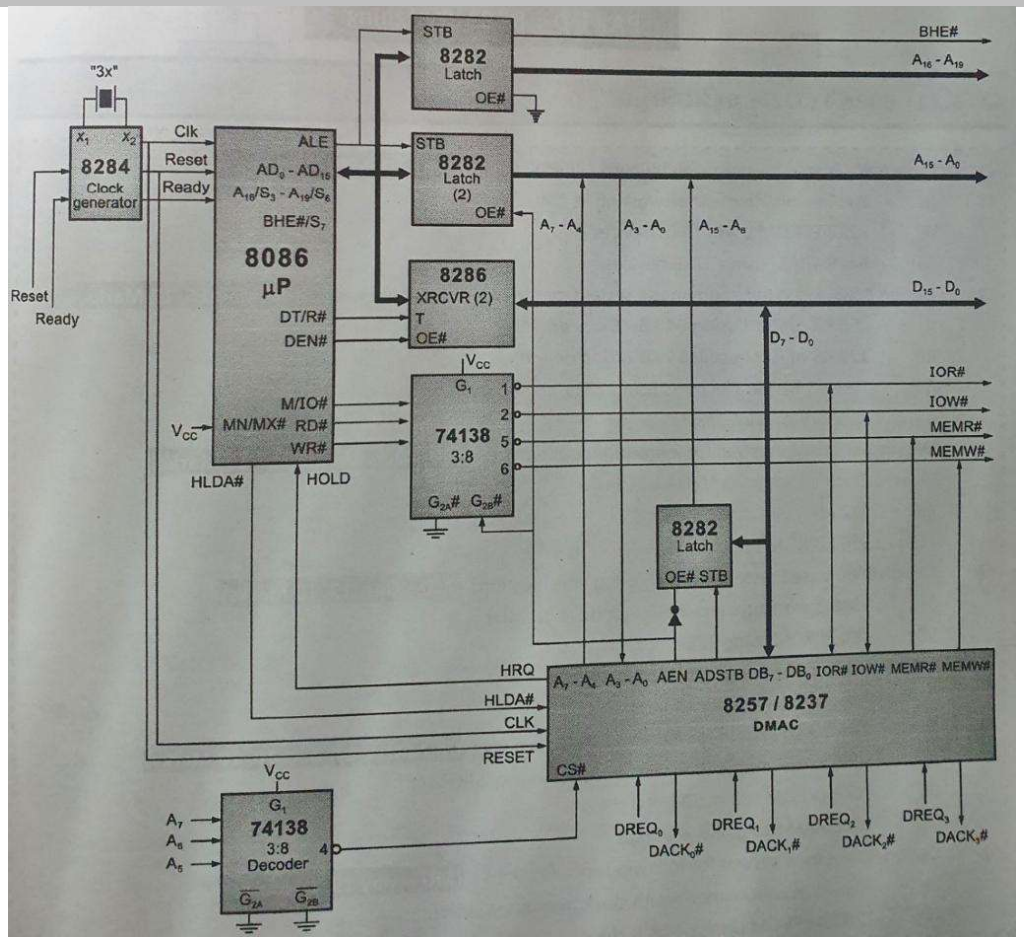
8257 DMA CONTROLLER

Count Register

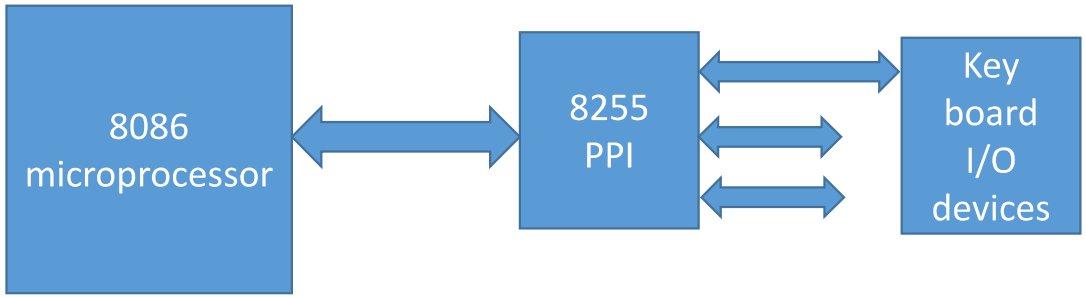


8257 DMA CONTROLLER

Interfacing with 8086 :



8255 PROGRAMMABLE PERIPHERAL INTERFACE



- I/O devices cant connect directly to µp (as transfer is very risky)
- I/O device connected through 8255 ports

8255 PROGRAMMABLE PERIPHERAL INTERFACE

Introduction :

1. It is used to interface (connect) up with I/O devices such as a printer, keyboard etc.
2. 8255 has three 8 bit bidirectional I/O ports called Port A, Port B and Port C
3. They can be used as input ports or as output ports.
4. These ports can operate in three different modes of data transfer.
5. Mode 0 is called Simple I/O. Here the three ports perform basic 8 bit data transfers.
6. Mode 1 is called Handshake I/O. Here port A and Port B transfer data using an advanced technique called handshaking. This prevents any loss of data and hence makes the transfer more reliable. To carry out handshaking, lines of Port C are used up by Port A and Port B.
7. Mode 2 is bidirectional handshaking in which as the name suggests, the port can perform input as well as output handshaking.
8. The modes are selected by the programmer by giving commands to 8255. Depending upon the system requirement, the appropriate command is formed and is first stored in a register like AL. Then using an instruction like OUT, the command is sent to 8255 through the data bus.
9. Additionally, 8255 has an excellent feature called BSR command. It stands for Bit Set Reset. Using BSR command, the programmer can individually set or reset (send 1 or 0) to any line of Port C without affecting other lines. This basically transforms Port C from being one 8 bit port to becoming eight 1 bit ports that can be individually controlled by the programmer. This feature is very useful in complex interfaces like ADC. LCD controller etc

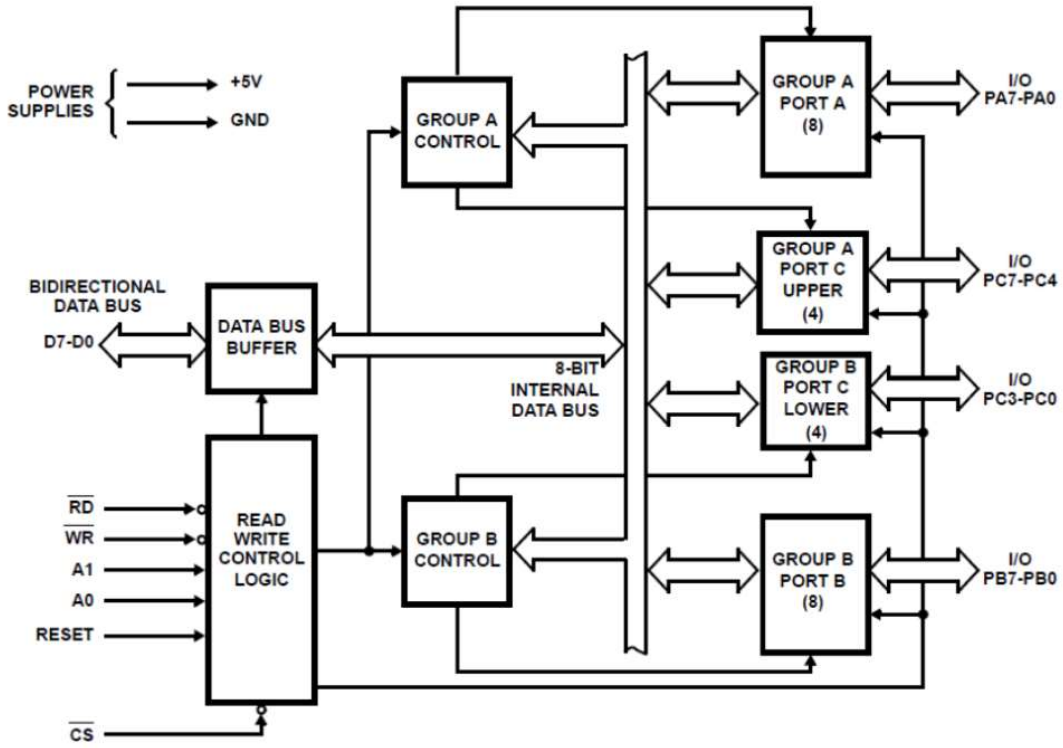
8255 PROGRAMMABLE PERIPHERAL INTERFACE

Features :

1. 8255 is a programmable peripheral interface / general-purpose I/O device.
2. It is used to connect microprocessor with I/O devices such printer, keyboard
3. It has 3 8-bit bi-directional I/O . ports: Port A, Port B, and Port C.
4. It provides 3 modes of data transfer: Simple I/O, Handshake I/O 4 and Bi-directional Handshake.
5. It also provides a Bit Set Reset Modes to alter individual bits of Port C.

8255 PROGRAMMABLE PERIPHERAL INTERFACE

ARCHITECTURE OF 8255



8255 PROGRAMMABLE PERIPHERAL INTERFACE

Main components of 8255 architecture:

- Data bus buffer
- Read/Write control logic
- Group A control
- Group B control
- Port A, Port B Port C

8255 PROGRAMMABLE PERIPHERAL INTERFACE

- Data bus buffer

➢ This is a 8-bit bi-directional buffer used to interface the internal data bus of 8255 with the external (system) data bus.

➢ The CPU transfers data to and from the 8255 through the data bus via this buffer.

➢ The commands given to 8255 (I/O command and BSR command) are also given through the same datu bus.

8255 PROGRAMMABLE PERIPHERAL INTERFACE

- Read/Write control logic :

➢ It accepts address and control signals from the µP.

➢ The Control signals determine whether it is a read or a write operation.

➢ During a read, data will be transferred from 8255 to the µp.

➢ During a write, data will be transferred from µp to 8255.

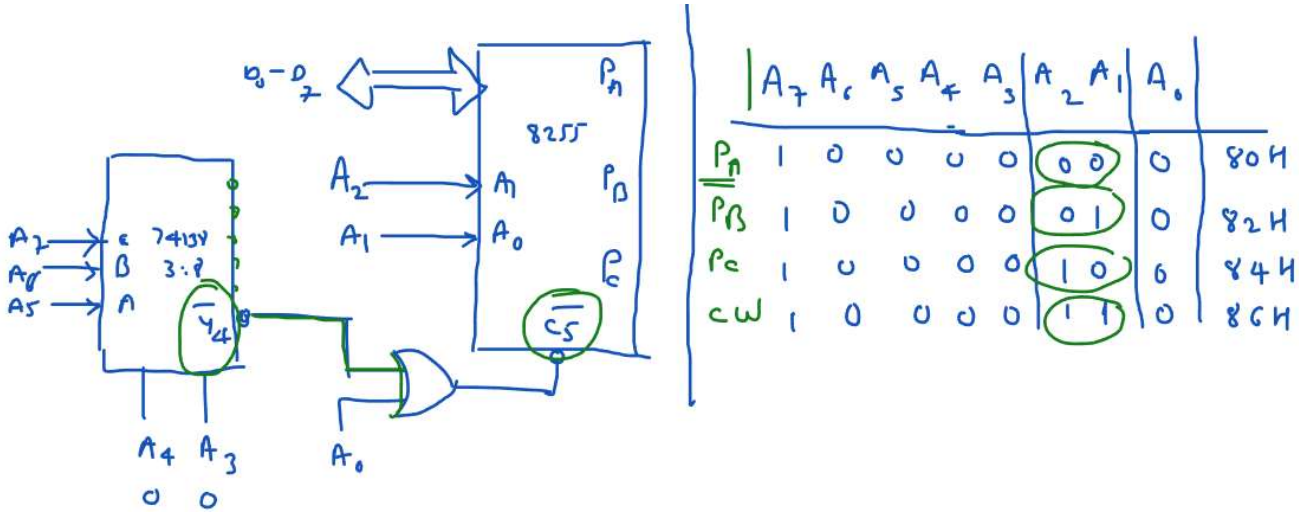
➢ There is a chip selection signal that selects 8255 on the basis of its address.

➢ The reset signal is to reset 8255 and stop all current transfers.

➢ Finally there are two address lines A1 and A0 to which we connect the address lines A2 and A1 of the µP 8086. These lines are uses to make internal selection within 8255.

8255 PROGRAMMABLE PERIPHERAL INTERFACE

For 8255 A_1 A_0	For 8086 A_2 A_1	Selection	Sample address
0 0	0 0	Port A	80 H (i.e. 1000 0**00**0)
0 1	0 1	Port B	82 H (i.e. 1000 0**01**0)
1 0	1 0	Port C	84 H (i.e. 1000 0**10**0)
1 1	1 1	Control Word	86 H (i.e. 1000 0**11**0)



8255 PROGRAMMABLE PERIPHERAL INTERFACE

Group A control :

- This Control block controls Port A and Port CUpper i.e. PC7-PC4.
- It accepts Control signals from the Control Word and forwards them to the respective Ports.

Group B Control :

- This Control block controls Port B and Port CLower i.e. PC3-PCO..
- It accepts Control signals from the Control Word and forwards them to the respective Ports.

8255 PROGRAMMABLE PERIPHERAL INTERFACE

Port A, Port B, Port C :

These are 8-bit Bi-directional PortsThey can be programmed to wok in the various modes as follows:

Port	Mode 0	Mode 1	Mode 2
Port A	Yes	Yes	Yes
Port B	Yes	Yes	**No** *(Mode 0 or Mode 1)*
Port C	Yes	**No** *(Handshake signals)*	**No** *(Handshake signals)*

ONLY Port C can also be programmed to work in Bit Set reset Mode to manipulate its individual bits.

8255 PROGRAMMABLE PERIPHERAL INTERFACE

Group A control :

- This Control block controls Port A and Port CUpper i.e. PC7-PC4.
- It accepts Control signals from the Control Word and forwards them to the respective Ports.

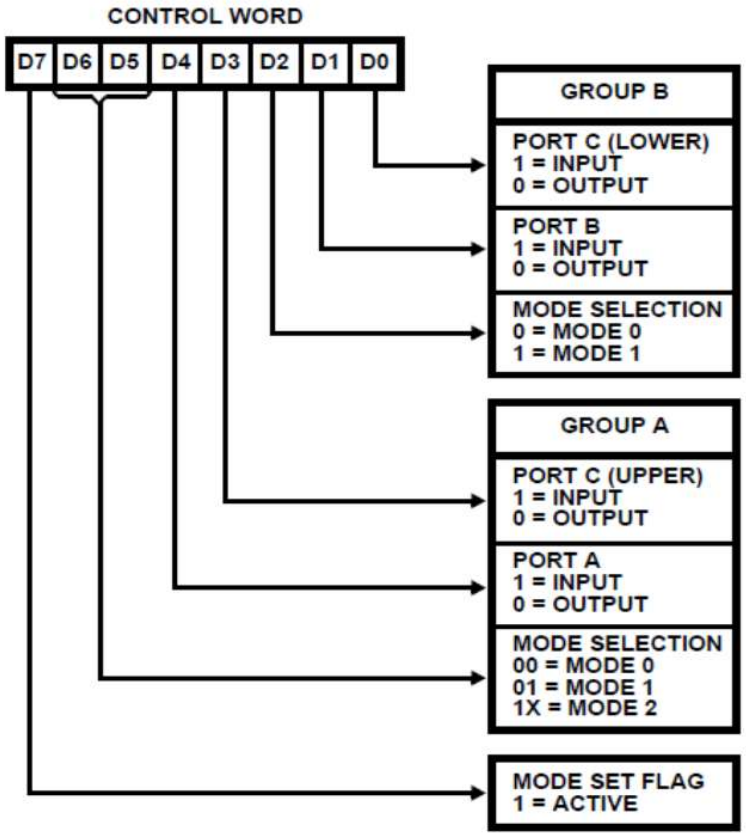
Group B Control :

- This Control block controls Port B and Port CLower i.e. PC3-PCO..
- It accepts Control signals from the Control Word and forwards them to the respective Ports.

8255 PROGRAMMABLE PERIPHERAL INTERFACE

I/O mode control word:

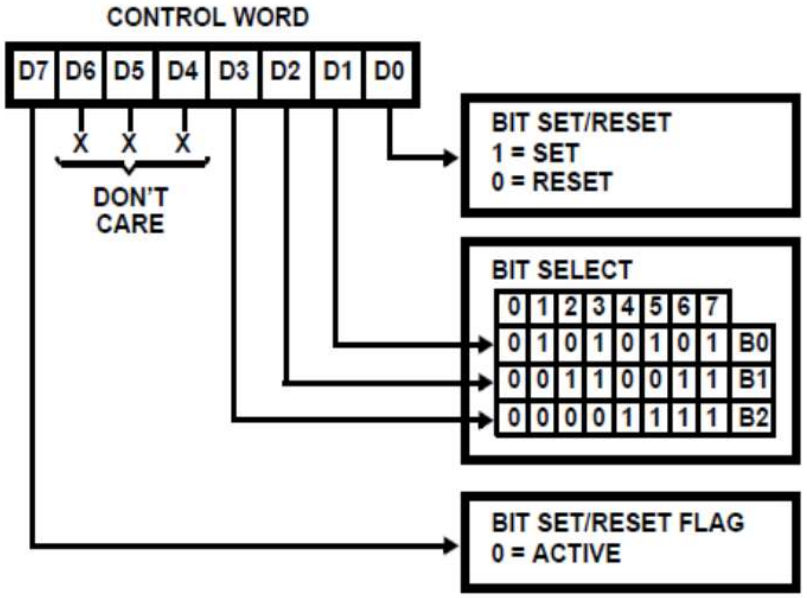
- To do 8-bit data transfer using the Ports A, B or C, 8255 needs to be in the IO mode.
- The bit pattern for the control word in the IO mode is as follows:



8255 PROGRAMMABLE PERIPHERAL INTERFACE

BSR mode control word:

- The BSR Mode is used ONLY for Port C.
- In In this Mode the individual bits of Port C can be set or reset.
- This is very useful as it provides 8 individually controllable lines which can be used while interfacing with devices like A to D Converter or a 7-segment display etc..
- The individual bit is selected and Set/reset through the control word.
- Since the D7 bit of the Control Word is 0, the BSR operation will not affect the modes



8255 PROGRAMMABLE PERIPHERAL INTERFACE

Data Transfer Modes:

1. Mode 0 (Simple Bi-directional I/O):

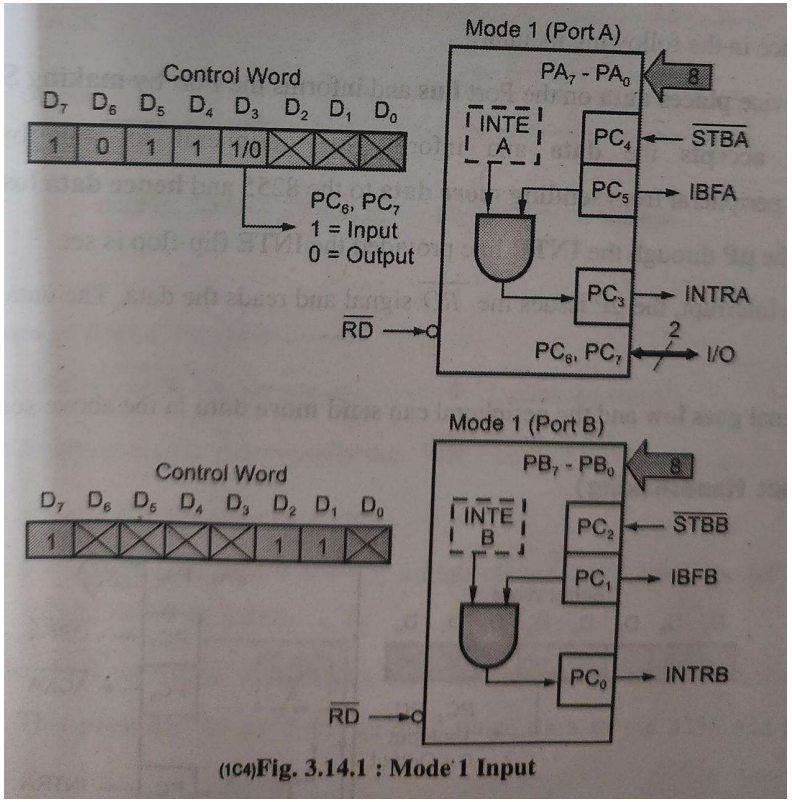
- Port A and Port B used as 2 Simple 8-bit I/O Ports,
- Port C is used as 2 simple 4-bit I/O Ports.
- Each port can be programmed as input or output individually.
- Ports do not have handshake or interrupting capability.
- Hence, slower devices cannot be interfaced.

2. Mode 1 : (Handshake I/O)

- Mode 1 (Handshake 1/0)In Mode 1, handshake signals are exchanged between the devices before the data transfer takes place.
- Port A and Port B used as 2 8-bit I/O Ports that can programmed in Input OR in output mode.
- Each Port uses 3 lines from Port C for handshake. The remaining lines of Port C can be used for simple IO.
- Interrupt driven data transfer and Status driven data transfer possible.
- Hence, slower devices can be interfaced
- The handshake signals are different for input and output modes:

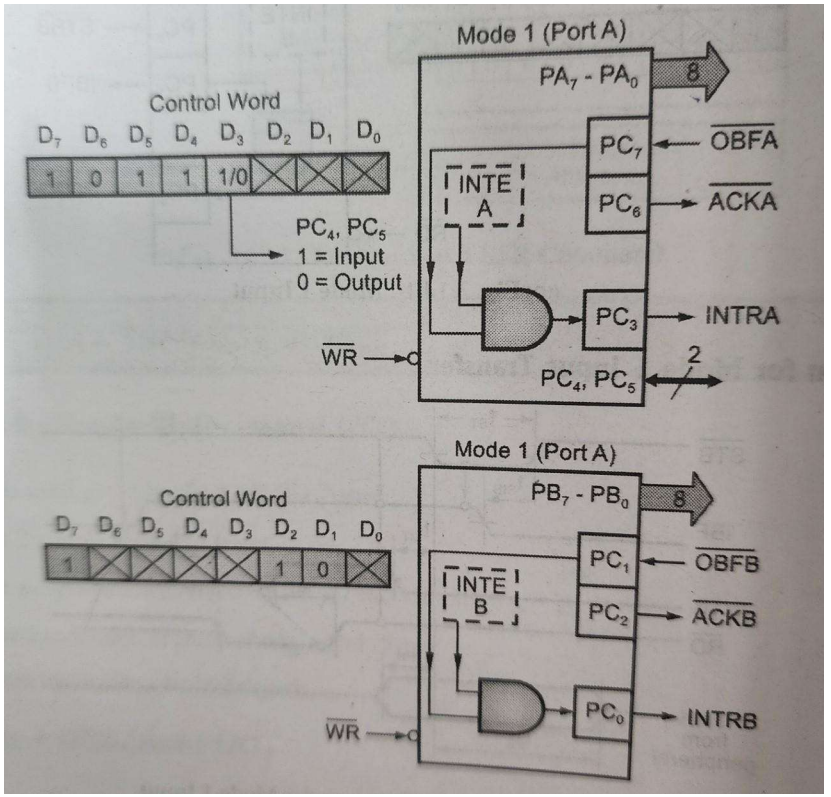
8255 PROGRAMMABLE PERIPHERAL INTERFACE

Mode 1 Input handshaking



(1C4)Fig. 3.14.1 : Mode 1 Input

Mode 1 output handshaking



8255 PROGRAMMABLE PERIPHERAL INTERFACE

For Input:

STB and IBF handshaking signals, INTR→ Interrupt signal

For Output:

OBF and ACK handshaking signals, INTR→ Interrupt signal.

Thus the 5 signals used from Port C are:

STB, IBF, INTR, OBF and ACK. :

8255 PROGRAMMABLE PERIPHERAL INTERFACE

- Working
  - Each port uses 3 lines of Port C for the following signals:
  - STB (Strobe), IBF (Input Buffer Full) Handhsake signals
  - INTR (interrupt) → Interrupt signal
  - Additionally the RD signal of 8255 is also used

Handshaking takes place in the following manner.

(1) The peripheral device places data on the Port bus and informs the Port by making STB low.

(2) The input Port accepts the data and informs the peripheral to wait by making IBF high. This prevents the peripheral from sending more data to the 8255 and hence data loss is prevented.

(3) 8255 interrupts the uP through the INTR line provided the INTE flip-flop is set.

(4) In response to the Interrupt, the uP issues the RD signal and reads the data. The data byte is thus transferred to the up

(5) Now, the IBF signal goes low and the peripheral can send more data in the above sequence.

8255 PROGRAMMABLE PERIPHERAL INTERFACE

- Working
  - Each port uses 3 lines of Port C for the following signals:
  - OBF (Output Buffer Full). ACK (Acknowledgement)→ Handhsake signals
  - INTR (interrupt) Interrupt signal. Additionally the WR signal of 8255 is also used.

Handshaking takes place in the following manner:

(1) When the output port is empty (indicated by a high on the INTR line), the uP writes data on the output port by giving the WR signal

(2) As soon as the WR operation is complete, the 8255 makes the INTR low, indicating that the up should wait. This prevents the uP from sending more data to the 8255 and hence data loss is prevented

(3) 8255 also makes the OBF low to indicate to the output peripheral that data is available on the databus:

(4) The peripheral accepts the data and sends an acknowledgement by making the ACK low. The data byte is thus transferred to the peripheral.

(5) Now, the OBF and ACK lines go high.

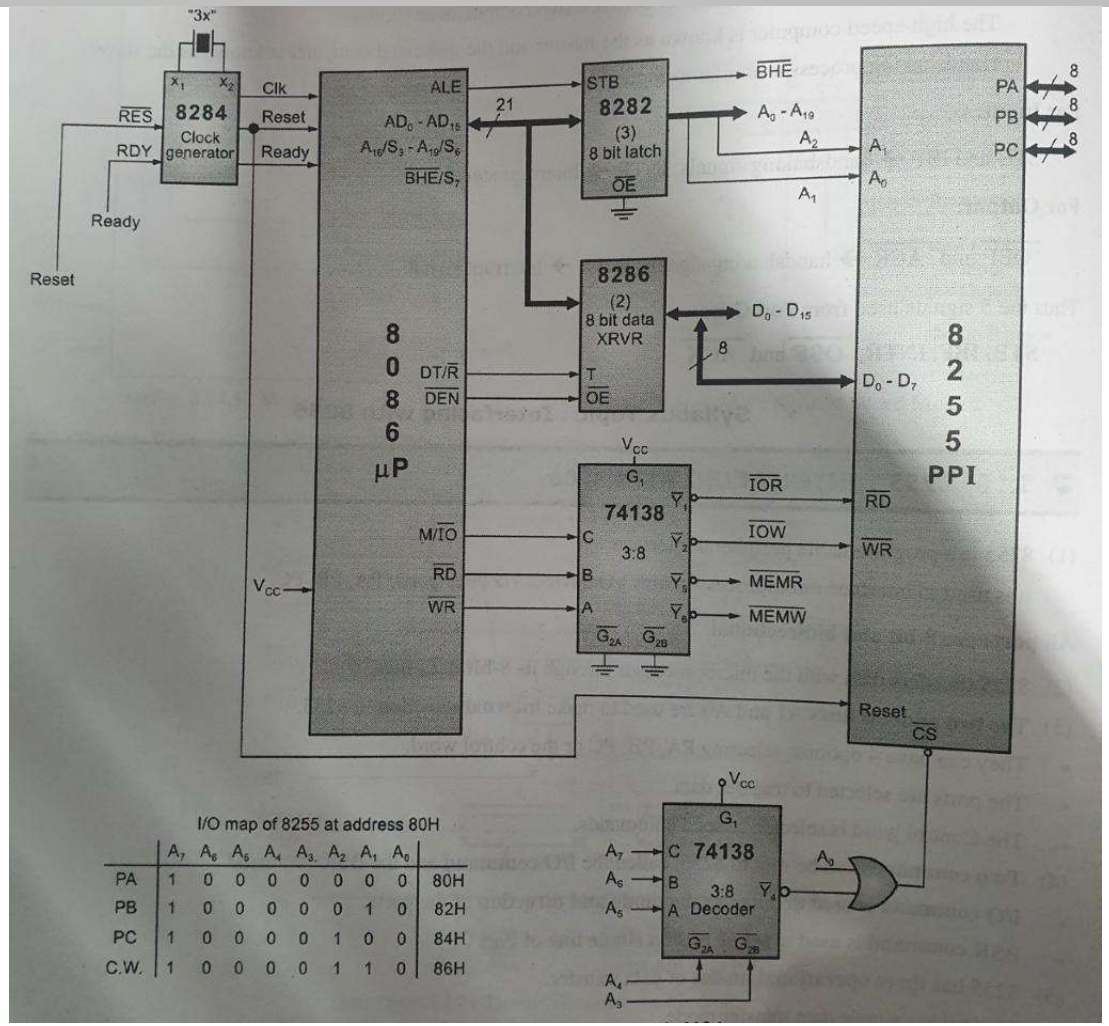
6) The INTR line becomes high to inform the uP that another byte can be sent. i.e. the output port is empty.

(6) This process is repeated for further bytes.

8255 PROGRAMMABLE PERIPHERAL INTERFACE

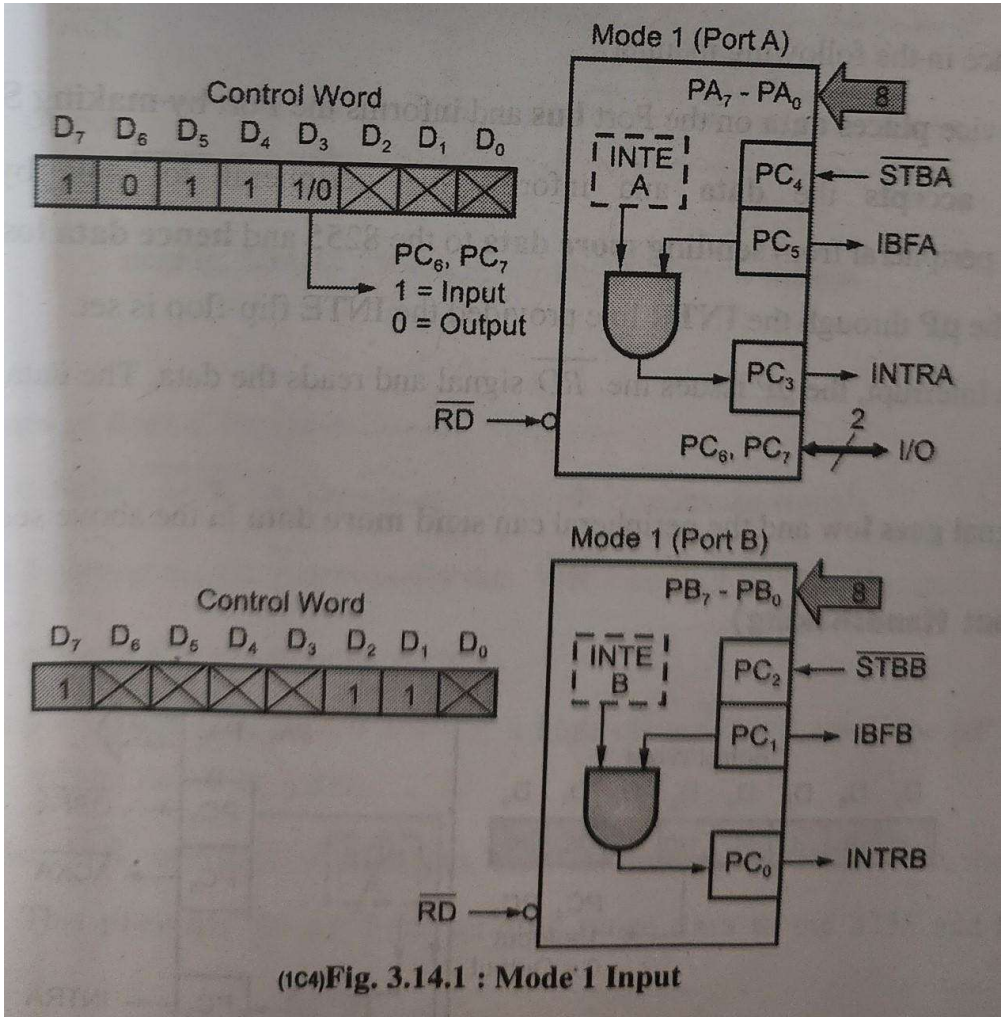
3. Mode 2 : (Bi- directional Handshake I/O)
   - In this mode, Port A is used as an 8-bit bi-directional Handshake I/O Port.
   - Port A requires 5 signals from Port C for doing Bi-directional handshake.
   - Port B has the following two options:
     - Use the remaining 3 lines of Port C for handshaking so that Port B is in Mode 1. Here Port C lines will be completely used for handshaking (5 by Port A and 3 by Port B).
     - Port B works in Mode 0 as simple I/O.
       - In this case the remaining 3 lines of Port C can be used for data transfer.
       - Port A can be used for data transfer between two computers as shown.
       - The high-speed computer is known as the master and the dedicated computer is known as the slave. Handshaking process is similar to Mode 1.For Input:STB and IBF handshaking signals, INTR→ Interrupt signalFor Output:OBF and ACKhandshaking signals, INTR→ Interrupt signal.Thus the 5 signals used from Port Care: STB, IBF, INTR, OBF and ACK.

8255 PROGRAMMABLE PERIPHERAL INTERFACE

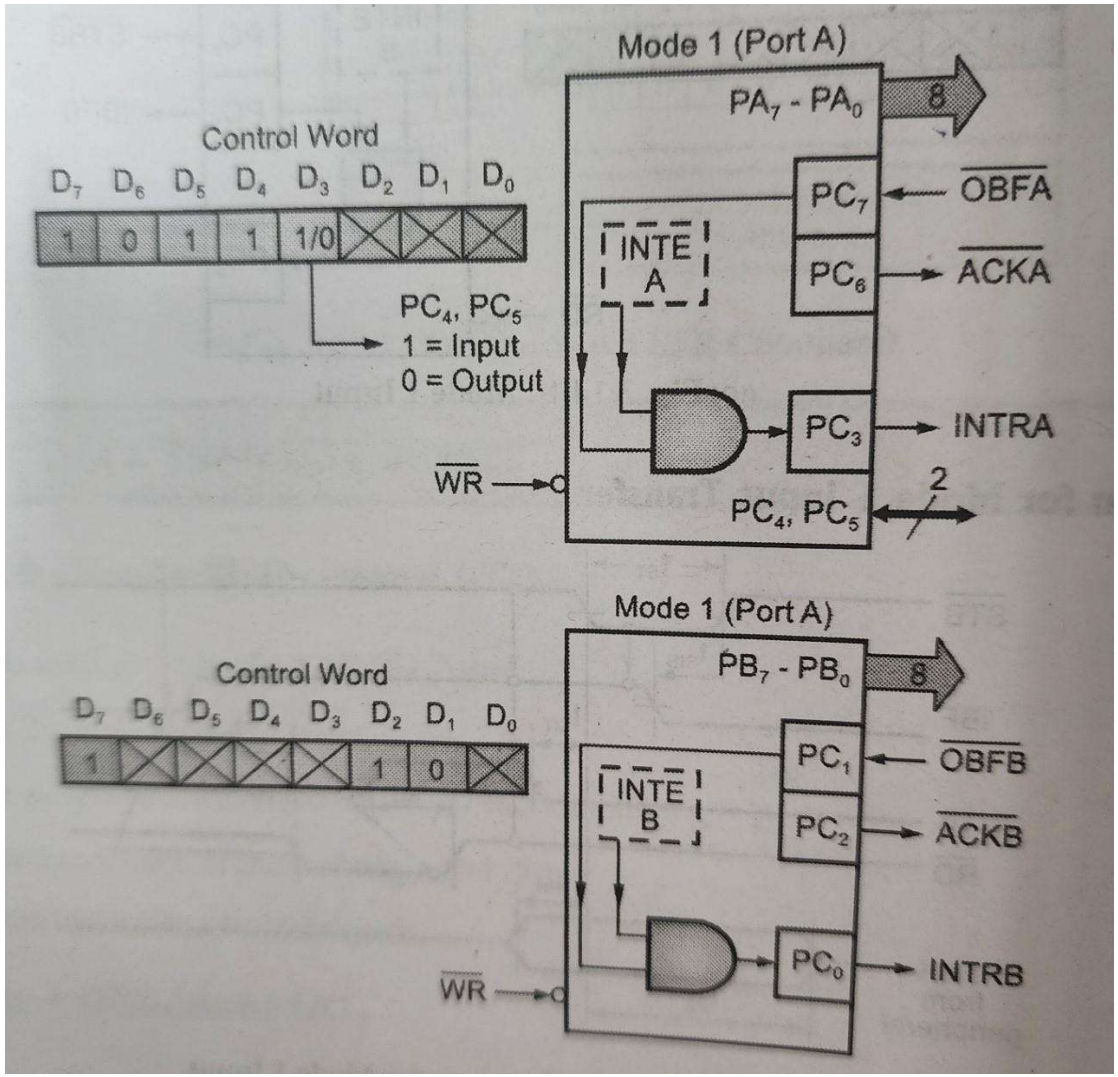


I/O map of 8255 at address 80H

	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
PA	1	0	0	0	0	0	0	0	80H
PB	1	0	0	0	0	0	1	0	82H
PC	1	0	0	0	0	1	0	0	84H
C.W.	1	0	0	0	0	1	1	0	86H



(1C4)**Fig. 3.14.1 : Mode 1 Input**


Mode 1 (Port A)
Control Word
D7 D6 D5 D4 D3 D2 D1 D0
1 0 1 1 1/0
PC4, PC5
1 = Input
0 = Output
PA7 - PA0
8
PC7
OBFA
INTE A
PC6
ACKA
PC3
INTRA
WR
2
PC4, PC5
Mode 1 (Port A)
Control Word
D7 D6 D5 D4 D3 D2 D1 D0
1 1 0
PB7 - PB0
8
PC1
OBFB
INTE B
PC2
ACKB
PC0
INTRB
WR

8259 PROGRAMMABLE INTERRUPT CONTROLLER

1. Introduction:
   - 8259 is a Programmable Interrupt Controller.
   - It is used to increase the number of hardware interrupts for the processor.
   - In single 8259 can provide 8 interrupts. A cascaded configuration can provide up to 64 interrupts using 1 Master 8259 and 8 Slave 8259s.
   - Though 8259 was created during the era of 8085 up, it was in service for various generations of the 186 family starting with the 8086 µP.
   - It has several flexible properties such has masking, edge/level triggering, priority modes, programmable vector numbers.

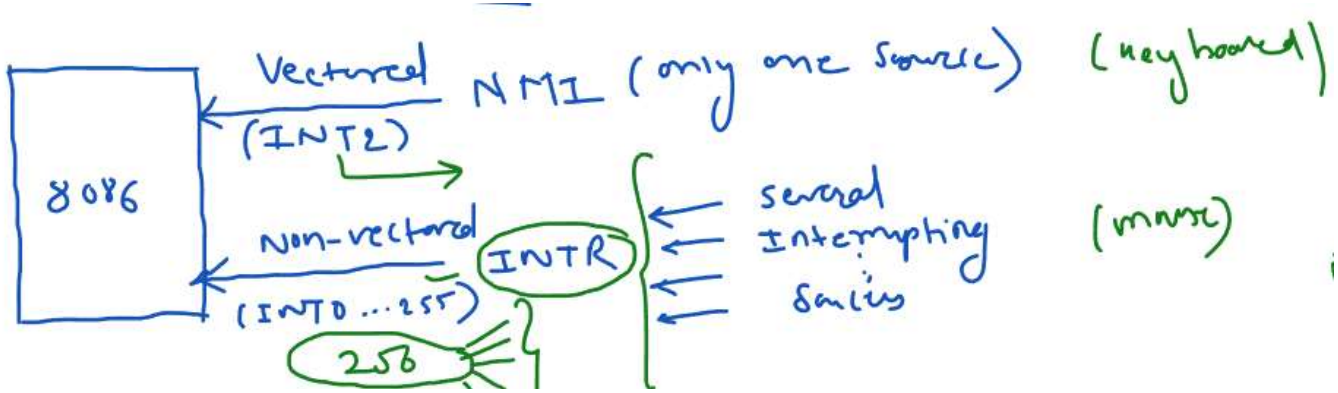
8259 PROGRAMMABLE INTERRUPT CONTROLLER

2. Need for 8259 PIC:

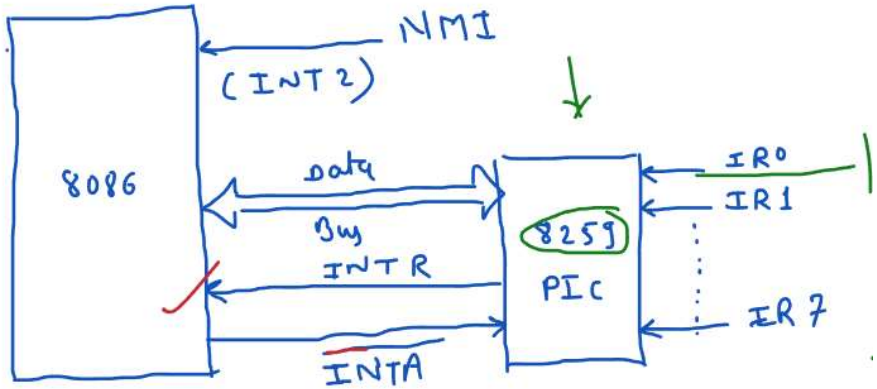
- 8086 has only two Hardware Interrupts: NMI and INTR Out of which, NMI is vectored. This means it has a fixed vector number in the IVT that is 2.So, whenever 8086 gets NMI interrupt, it will always execute the ISR of INT 2. This makes NMI "rigid".
- We can connect only a single device to interrupt 8086 through NMI because the ISR is fixed. INTR on the other hand is non-vectored It does not have a fixed vector number and hence 8086 can execute any ISR from INT 0... INT 255. This makes INTR a "flexible" interrupt line.
- We can combine several interrupting devices to give a common interrupt to 8086 on INTR .
- The idea makes sense, but we will need additional hardware .
- This is why we need 8259 as an Interface between the interrupting device and 8086.

8259 PROGRAMMABLE INTERRUPT CONTROLLER

1. Interrupt Sources:



2. Interrupts via 8259:



8259 PROGRAMMABLE INTERRUPT CONTROLLER

3. Initialization of 8259 :

- 8086 does not know the vector number as INTR is non-vectored
- To obtain the vector number, 8086 issues INTA signal. There are 2 INTA pulses issued.
- When 8086 gives the 1" INTA signal, 8259 starts preparing the vector number.
- When 8086 gives the 2 INTA signal, 8259 sends the 8 bit the vector number to 8086 through an 8 bit data bus. Although 8086 has a 16 bit data bus, 8259 is an 8 bit device and hence has an 8 bit data bas.
- Once 8086 gets the vector number, it multiplies the number by 4 and goes to the appropriate location in the IVT to obtain the ISR address and hence execute the ISR

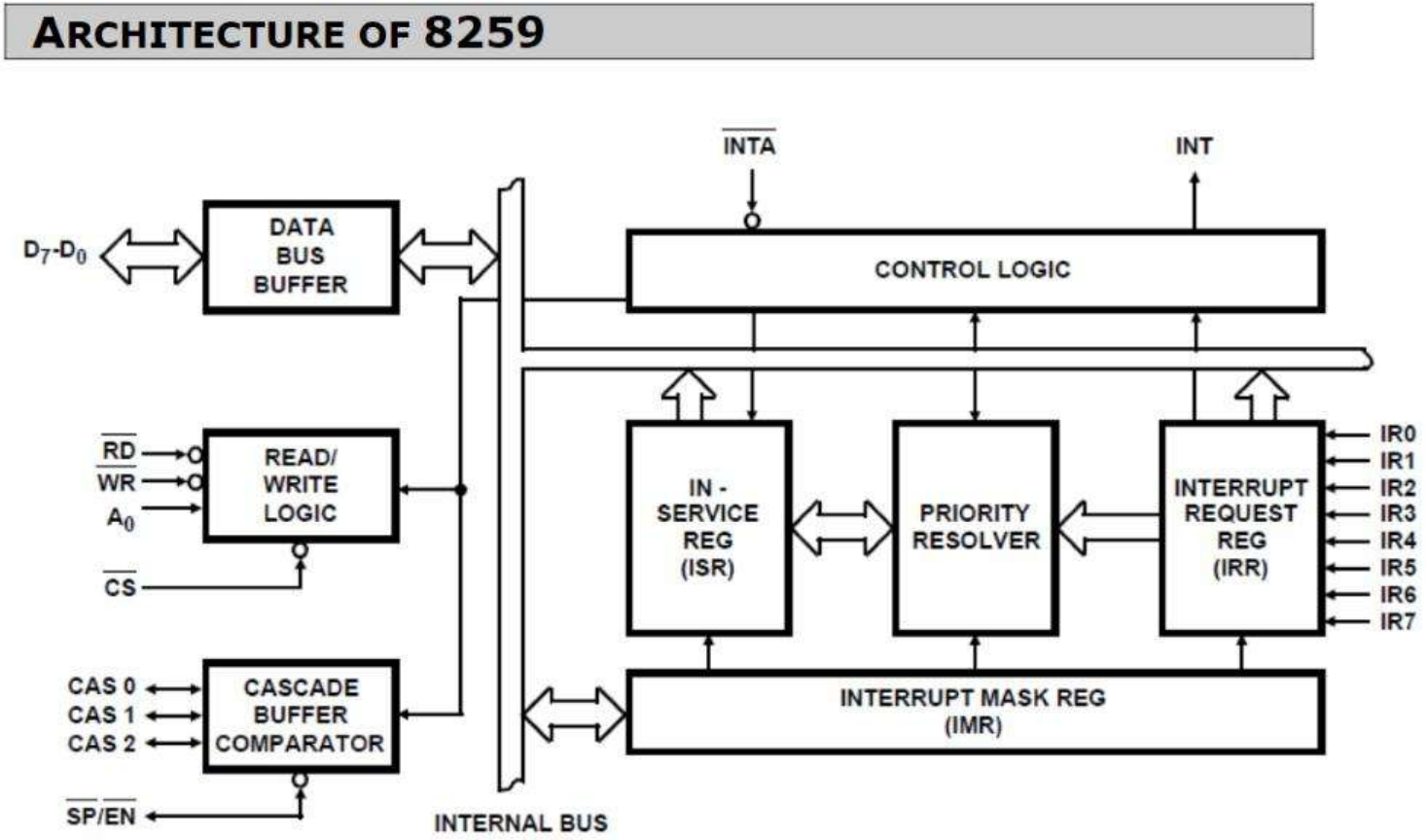
8259 PROGRAMMABLE INTERRUPT CONTROLLER

3. Initialization of 8259

- Before we start using 8259 to accept interrupts for 8086, we must first initialize 8259.Initialization of 8259 is compulsory.
- This is done by sending various commands to 8259 through the databus. We form these commands and put them into a register like AL.
- Then using the OUT instruction we send these commands from 8086 to 8259 through the data bus.
- The commands are called ICWs and OCWs.
- Using these commands we decide several properties such are Priority modes, Masking. Trigger modes(Edge or Level) and most importantly we inform the Vector numbers of all interrupts to 8259.

8259 PROGRAMMABLE INTERRUPT CONTROLLER

Architecture (Block diagram)



8259 PROGRAMMABLE INTERRUPT CONTROLLER

Architecture (Block diagram) :

Components :

1. Interrupt Request Register (IRR)
2. INTAIn-Service Register (InSR)
3. Interrupt Mask Register (IMR)
4. Priority Resolver

8259 PROGRAMMABLE INTERRUPT CONTROLLER

Interrupt Request Register (IRR)

- 8259 has 8 interrupt input lines IR7... IRO.
- 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.
- 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 (becomes 1)in the Interrupt Request Register (IRR).
- This bit reminds the processor that the interrupt is pending.
- It means, the interrupt has occurred but not yet serviced.
- he bit will only become 0 when the uP acknowledges this interrupt and responds by sending 1st $\overline{\text{INTA}}$

8259 PROGRAMMABLE INTERRUPT CONTROLLER

2. INTA In-Service Register (InSR)

- It is an 8-bit register. It stores the level of the Interrupt Request, currently being serviced.
- When the up send the 1st INTA the corresponding bit becomes 1 in this register, indicating that from now on , this particular interrupt is being serviced. This bit must stay 1 till the end of the ISR.
- During this same ISR if any other interrupt occurs, the priority resolver compares its level (in IRR) with the interrupt which is currently being 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 the uP.
- 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 uP has finished the ISR and moved on.
- To clear this bit from InSR, there are two options.
- In Normal EOI mode (default mode), the programmer must give an EOI command at the end of the ISR. EOI stand for End of Interrupt.
- In Auto EOI mode, this bit is cleared automatically in the 2nd INTA for 8086 systems.

8259 PROGRAMMABLE INTERRUPT CONTROLLER

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.
- The interrupts can be masked or unmasked by the programmer as many times as they may wish to.
- This is done by the OCW1 command.
- To mask a bit, the corresponding bit must be made 1.
- Now even if this interrupt occurs, it will not be sent to the JP, irrespective of its priority.

4. Priority Resolver

- It examines the IRR, InSR, and IMR and determines which interrupt is of highest priority and should be sent to the up.
- It checks IRR to know which interrupts have occurred, IMR to know which interrupts are masked and InSR to know which interrupt is in service.
- If the interrupt that has occurred is of the highest priority amongst those that have occurred, is unmasked and is higher priority than the one currently being serviced, only then the interrupt will be sent to the uP.

8259 PROGRAMMABLE INTERRUPT CONTROLLER

5. Control Logic:

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

6.Data Bus Buffer

- It is a bi-directional buffer used to interface the internal data bus of 8259 with the external (system) databus.
- The data bus is used to transfer commands from up to 8259 and also give the vector number from 8259 the uP

7. Read/Write Logic

- It is used to accept the RD, WR, A, and CS signal.
- It is also used to hold some of the Initialization Command Words (ICW's) and the Operational Command Words (OCW's)

8259 PROGRAMMABLE INTERRUPT CONTROLLER

8. Cascade Buffer / Comparator

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

1. CAS 2, CAS 1, CAS 0 lines

CAS lines are used by the master, to inform the slave, that it has been selected, between the 1st and 2nd INTA, so that the slave can give the correct vector number to the up.

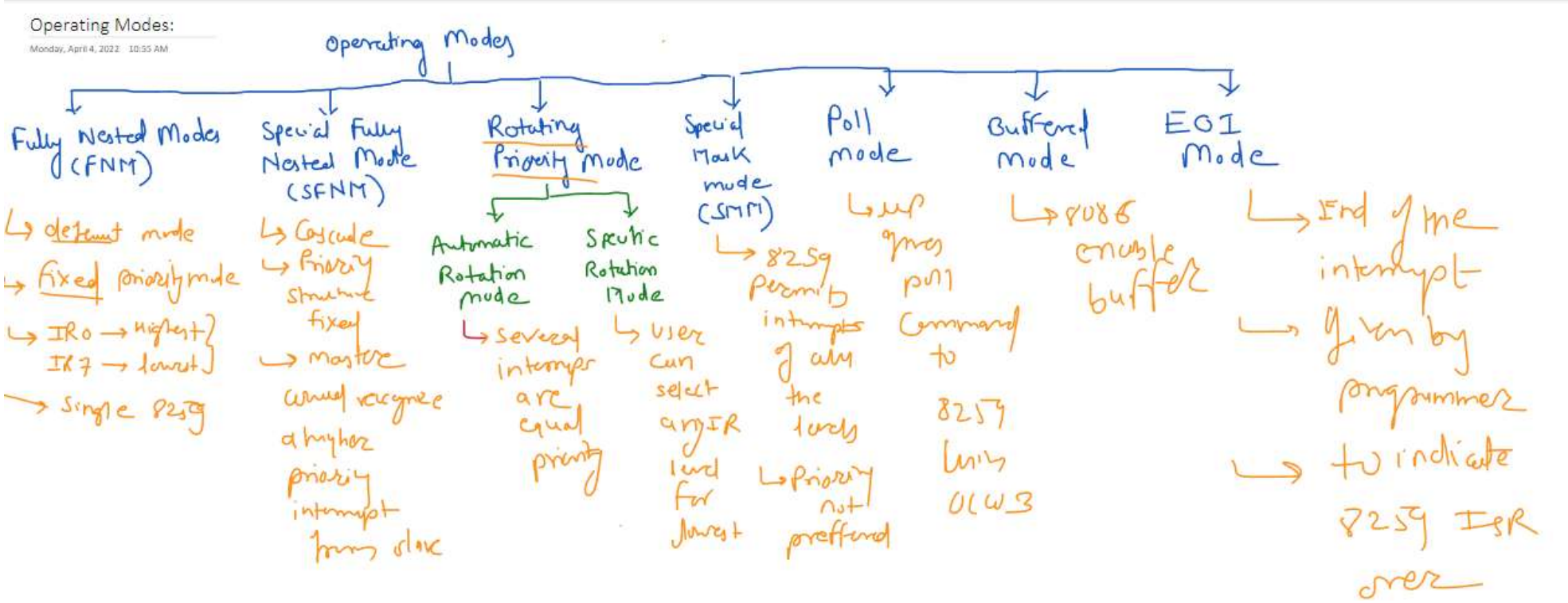
There are 3 CAS lines.

They can carry any number between 000... 111. This means there can be a maximum of 8 slaves connected to the master with the Slave Ids ranging from 000... 111.

2. SP/EN (Slave Program/Master Enable):In Buffered Mode, it functions as the EN and is used to enable the buffer. In Non buffered mode, it functions as the SP. For Master 8259 SP should be high, and for the Slave SP should be low.

8259 PROGRAMMABLE INTERRUPT CONTROLLER

Operating modes of 8259 :

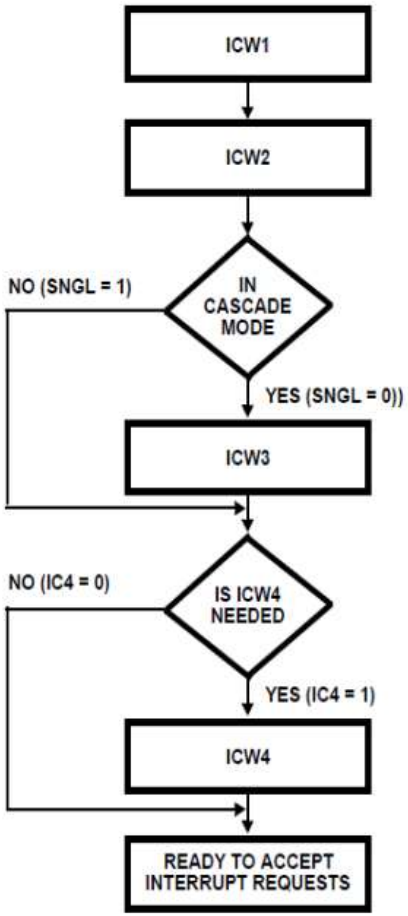


8259 PROGRAMMABLE INTERRUPT CONTROLLER

Initialization sequence of 8259 :

- ICWs

- ICWs have to be given during the initialization of 8259 (i.c. before the uP can start using 8259).
- ICW1 and ICW2 are compulsory.
- ICW1 indicates if the system is Single or Cascaded.
- If Cascaded, ICW3 has to be given.
- Whether ICW4 is required or not, is specified in the ICW1. If ICW4 is required, it has to be given.
- It is important that the ICWs are given in the above sequence only.
- None of the ICWs can be individually repeated, but the entire initialization can be repeated if required.

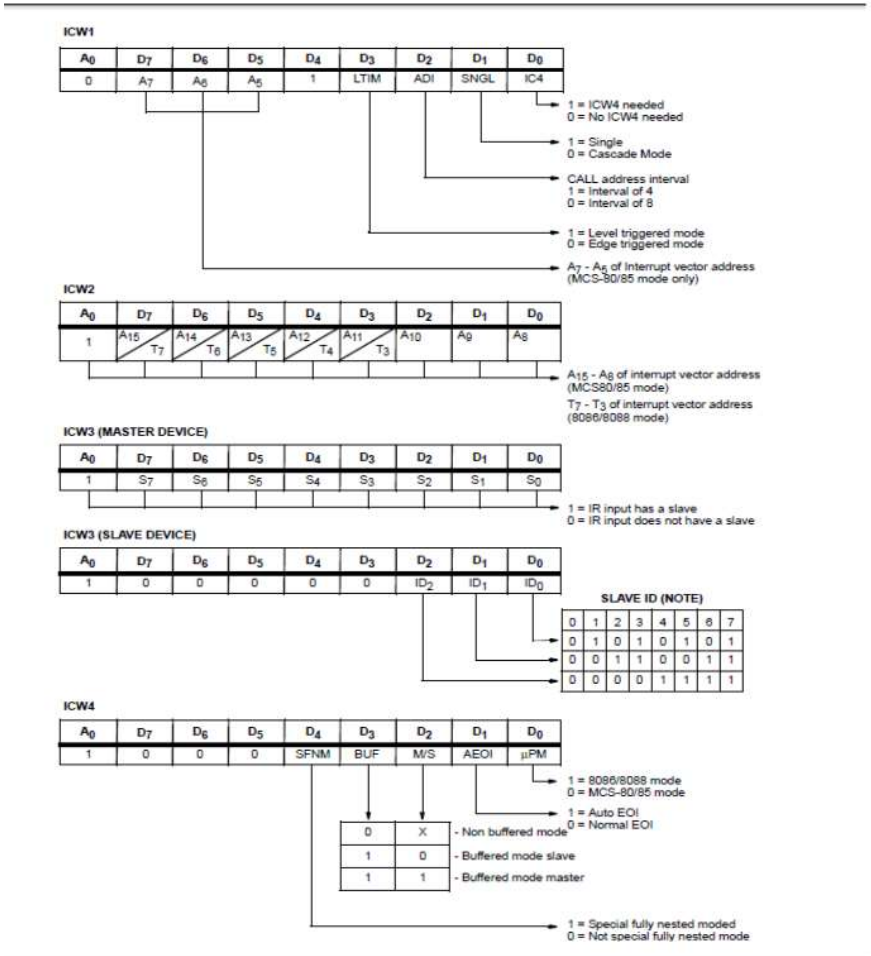


8259 PROGRAMMABLE INTERRUPT CONTROLLER

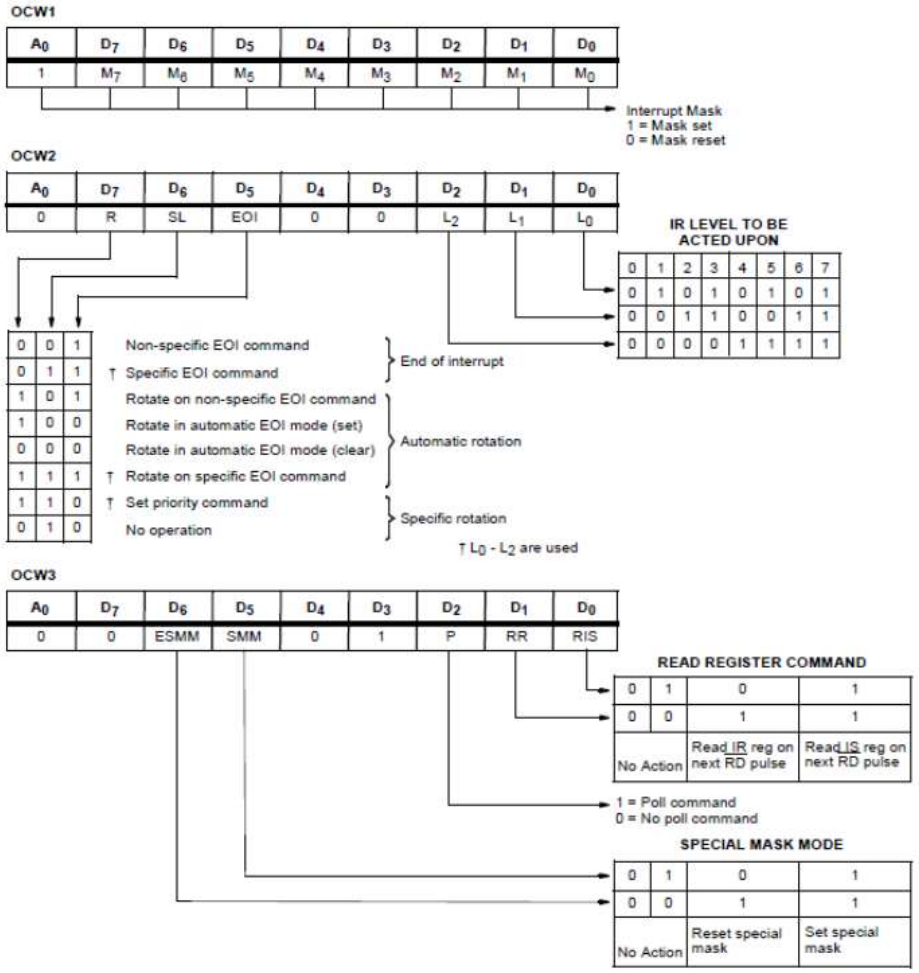
Operating modes of 8259 :

- OCWs
  - OCWs are given during the operation of 8259 (ie. after the uP has started using 8259).
  - OCWs are not compulsory. OCWs do not have to be given in a specific order.
  - OCWs can be individually repeated.
  - They are mainly used to alter the masking status and the operation modes of 8259.

8259 PROGRAMMABLE INTERRUPT CONTROLLER



8259 PROGRAMMABLE INTERRUPT CONTROLLER



8259 PROGRAMMABLE INTERRUPT CONTROLLER

Interfacing of 8259