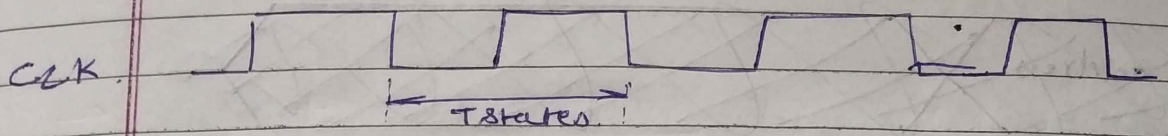
Date: / /
Page No.

Subroutine Handling Instructions

↳ small segment of program/instruction

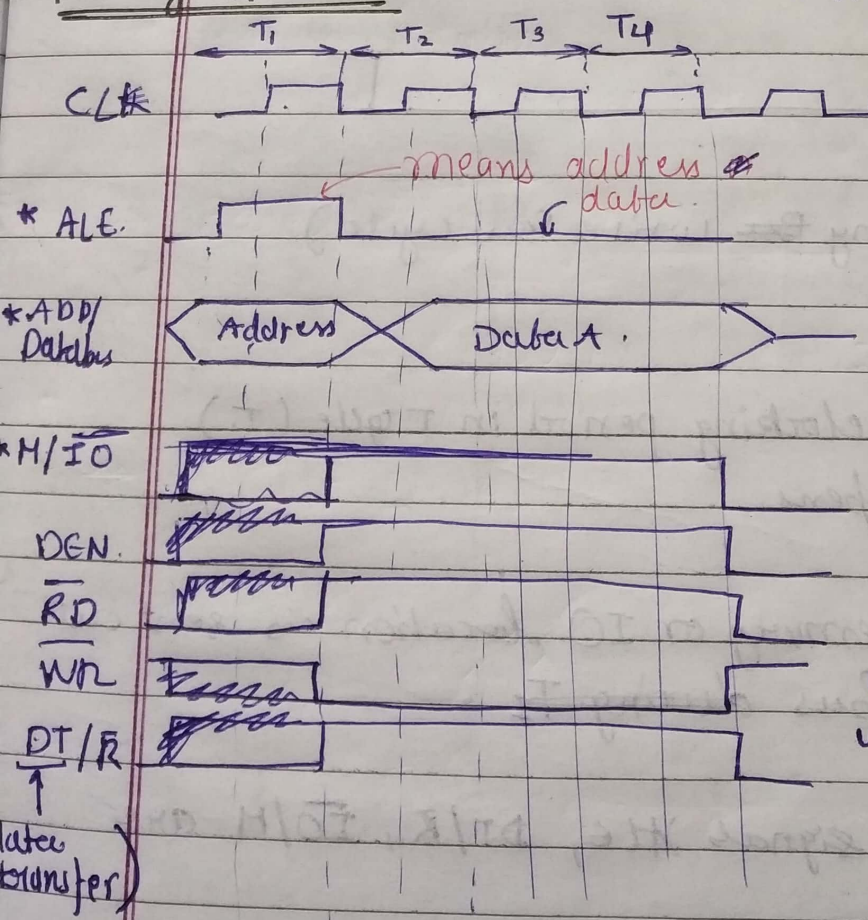
Bus Cycle CLK freq = 5 KHz



8086 → has atleast 4 Tstate ; at max 6 Tstates.

→ Bus cycle → sequence of events that start with an address being output, followed by read/write data bus. on the system bus.

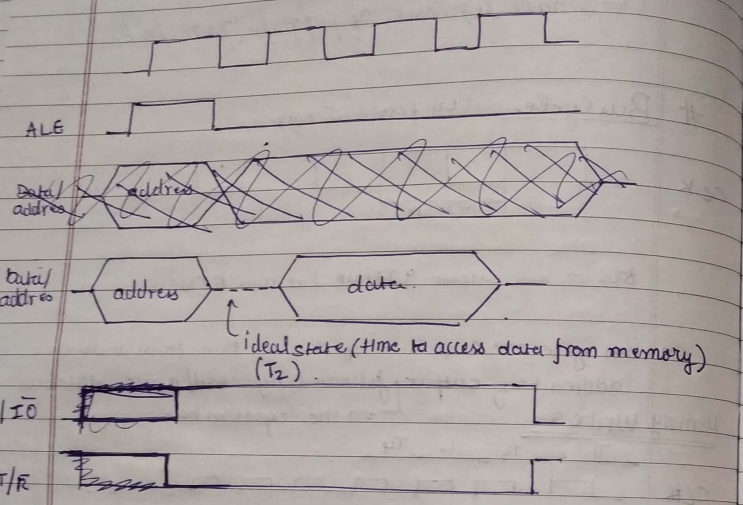
Memory Write :



IO Write : everything remains except M/IO goes Low

Page No.

Memory Read Bus cycle



(same as memory ~~Bus~~ write Bus cycle).

during the first clocking period in T cycle (T_1) many thing happens.

→ the address of memory or IO location is sent out via the address bus ~~during T_1~~.

During T_1 control signal ALE, $DT/\overline{R}$, $\overline{IO}/M$ are also output.

$M/\overline{IO}$ signal indicates weather the address indicates Memory or IO address.

Date: / /
Page No.

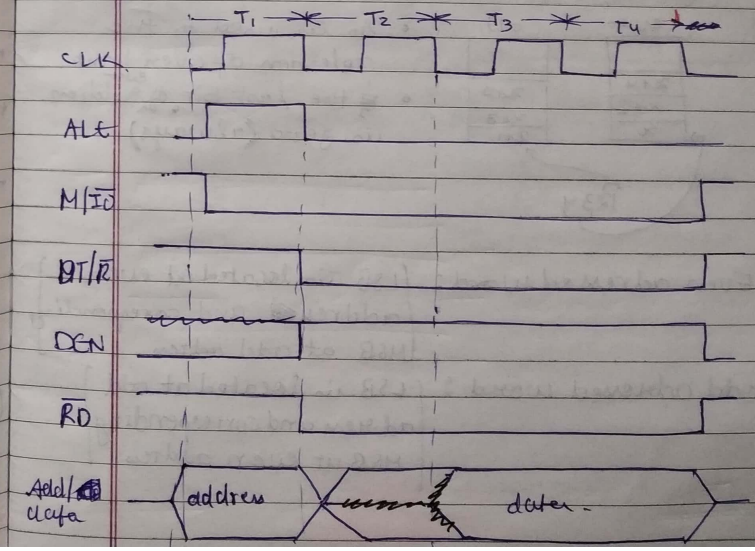
→ during read cycle T_2 becomes the wait state and this clocking period provides the memory, the time to access the data.

→ during T_4 all bus signals are deactivated and the prepration for next bus cycle.

ALE, Add/data, $M/\overline{IO}$ } Starts from T_1

DEN, $DT/\overline{R}$, $\overline{RD}$, $\overline{WR}$ } starts from the end of T_1 or from the starting of T_2.

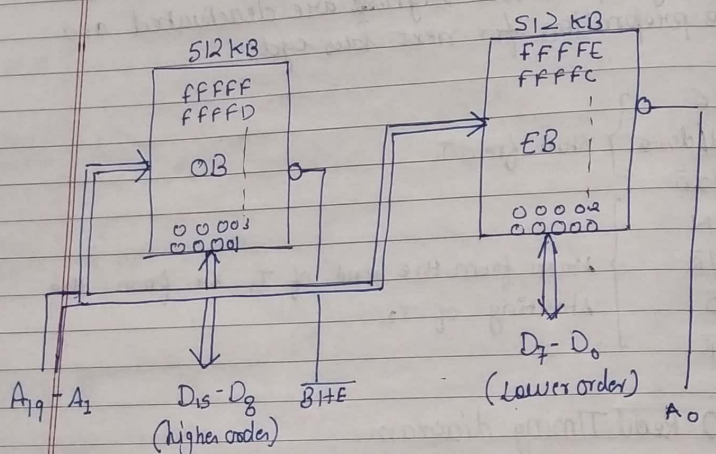
IO Read Timing diagram

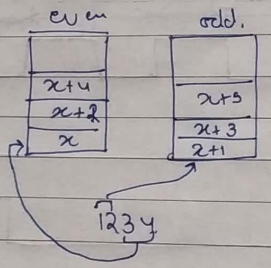


Date: / /
Page No.

~~Hardware Organization of Memory Space~~

Memory Adress Space Organization





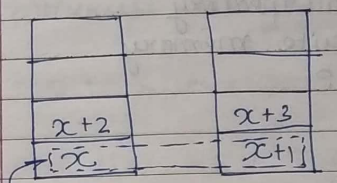
- A_0 is used for the selection of even bank.
- If the last bit of even address is zero (always)

Even adressed Word: [LSB is located at even address and coresponding MSB at odd adres.]

odd adressed word: [LSB is located at odd address and corresponding MSB at even address]

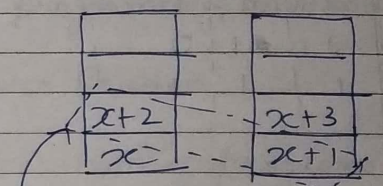
Date: / /
Page No.

- even addressed bus is said to be aligned and takes 1 bus cycle to execute.
- odd adress bus is unaligned and takes 2 bus cycle to execute.



aligned (even addressed bus)

[because only at one level hence 1 bus cycle]



odd address bus (unaligned)

[because data at two levels hence 2 bus cycle]

#

S_4	S_3	Segment Register.
0	0	extra
0	1	Stack
1	0	Code
1	1	Data.

Date: / /
Page No.

CPU cycle.

S_2	S_1	S_0	
0	0	0	INTA
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	1	1	Halt.
1	0	0	Instruction fetch
1	0	1	Read memory
1	1	0	Write memory
1	1	1	NOP

Queue Status

QS_1	QS_0	
0	0	Nop
0	1	First byte of instruction
1	0	Queue Empty
1	1	Subsequent byte of instruction

Date: / /
Page No.

INTERRUPTS

1. Internal Interrupt (eg. divide by 0)
2. NMI
3. Software Interrupt (eg. INT A5)
4. Hardware Interrupt.
   - Further divided into TYPE0 – TYPE255 → towards lower priority

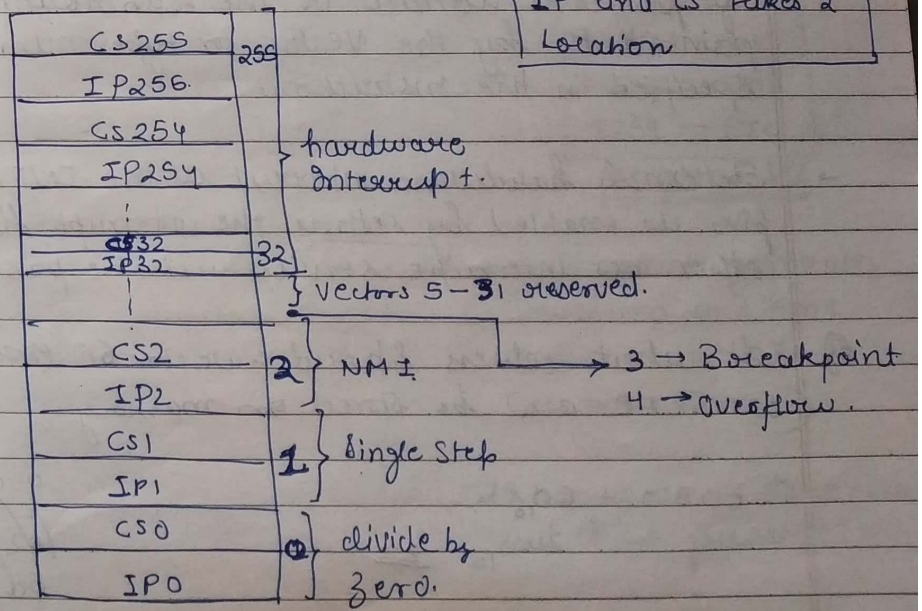
↑ High priority
↓ Low priority.

→ Interrupt changes the programming environment.

- ISR ← Interrupt Service Routine.
- MPR will stop execution of program and control is given to ISR.
- highest priority interrupt can halt lower priority interrupt.

Interrupt Vector table

- IP and CS takes 2 Location



→ Vector 0, 1, 3 & 4 are internal interrupt

→ The pointer table is located at the low adress end of the memory adress space. It starts at 0050H and ends at 03FEH

→ Each of 256 pointers requires 2 words of memory. These words are stored at even adress boundary (for faster execution).

→ The higher address word of 2-word vector is called base adress which identifies the program memory segment in which the service routine resides and for this reason it is loaded in code segment.

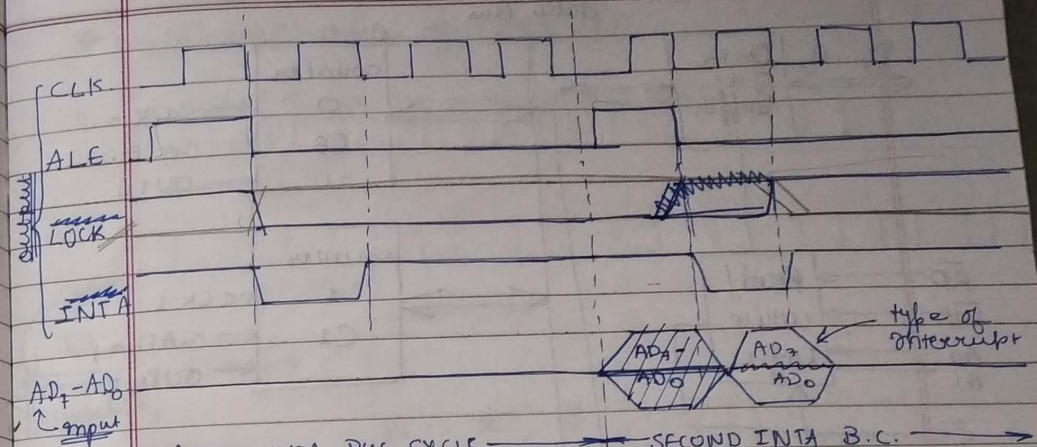
→ The software interrupt instruction (INT N) executing this instruction causes transfer of program control to the subroutine pointed to by the vector for type number N specified in the instruction.

→ External hardware interrut at the INTR pin is enabled by setting the interrupt flag, other no interrupt services will be provided

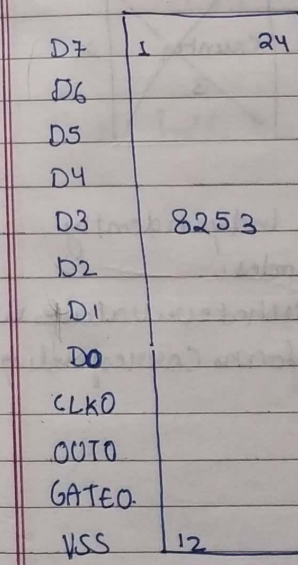
Q) At what adress shoud a vector 60 (CS = 60 IP = 60) be stored on memory.

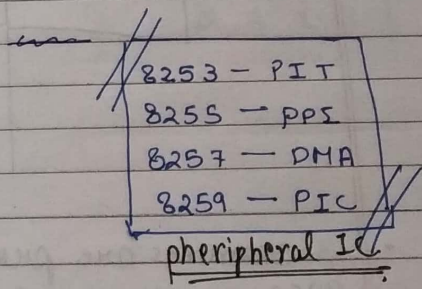
0050 + 60_{10} × 4

240_{10}

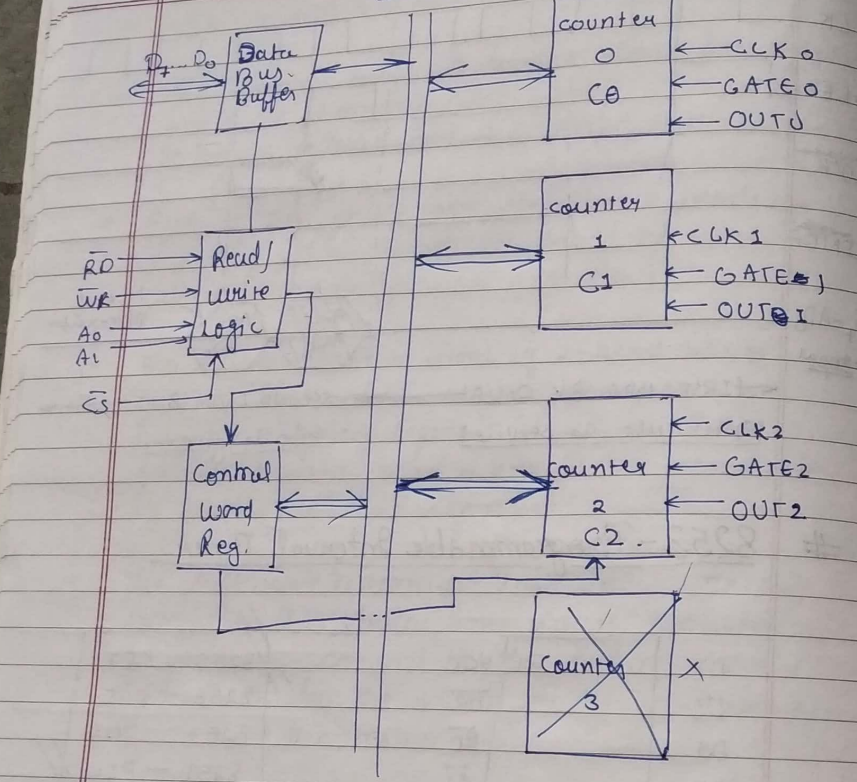


8253 - Programmable Interval Timer.





- It is used to generate waveform and time delay ranging from DC to megahertz.
- It has 3 down counter and programmable

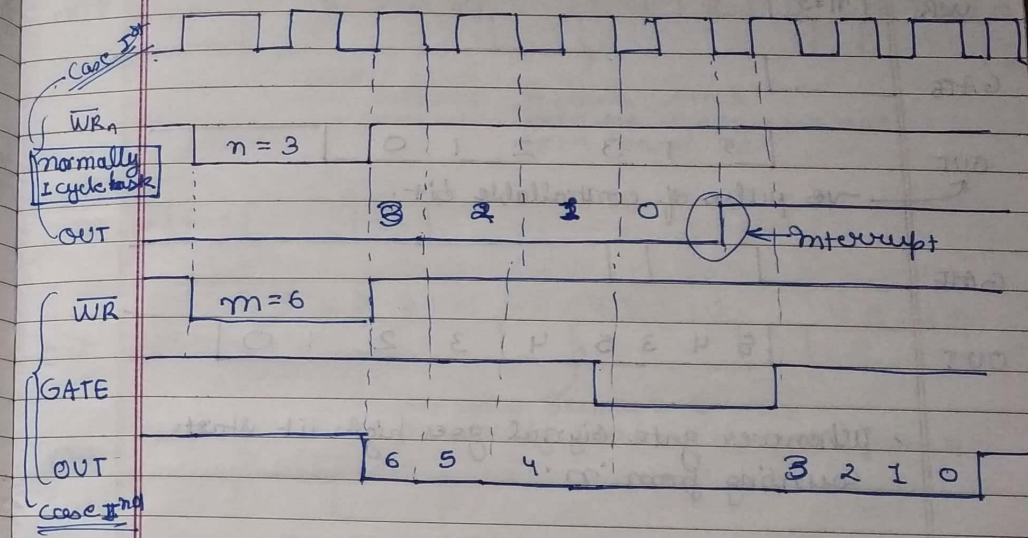


- 3 counters are programmed independently.
- 8253 works in 6 different modes.
- Control word reg → [8 bit reg. Whatever value we put, IC performs corresponding task]

different modes of 8253 PIT

A. Mode 0: Interrupt on terminal count

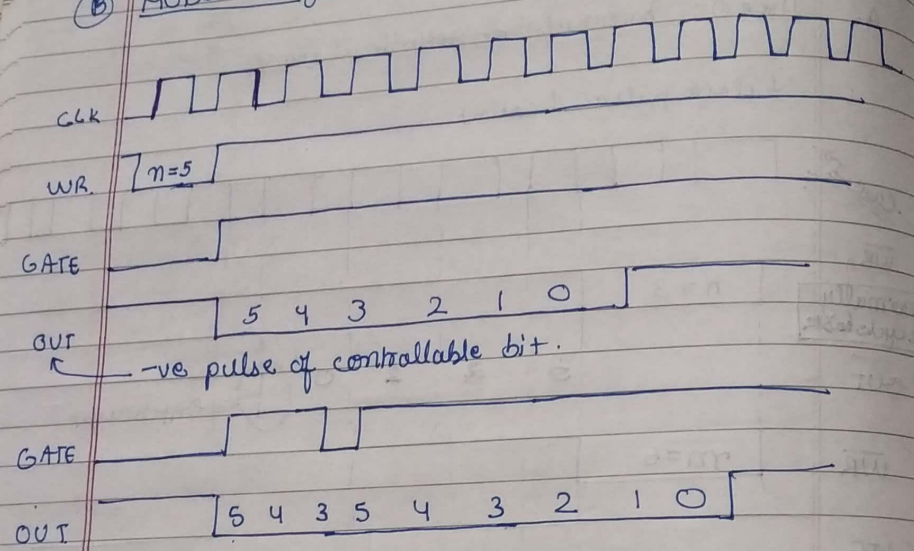
- 1 clock pulse 1 count.



- Gate holds the counting.

Date: / /
Page No.

(B) MODE 1: Programmable ONE shot.

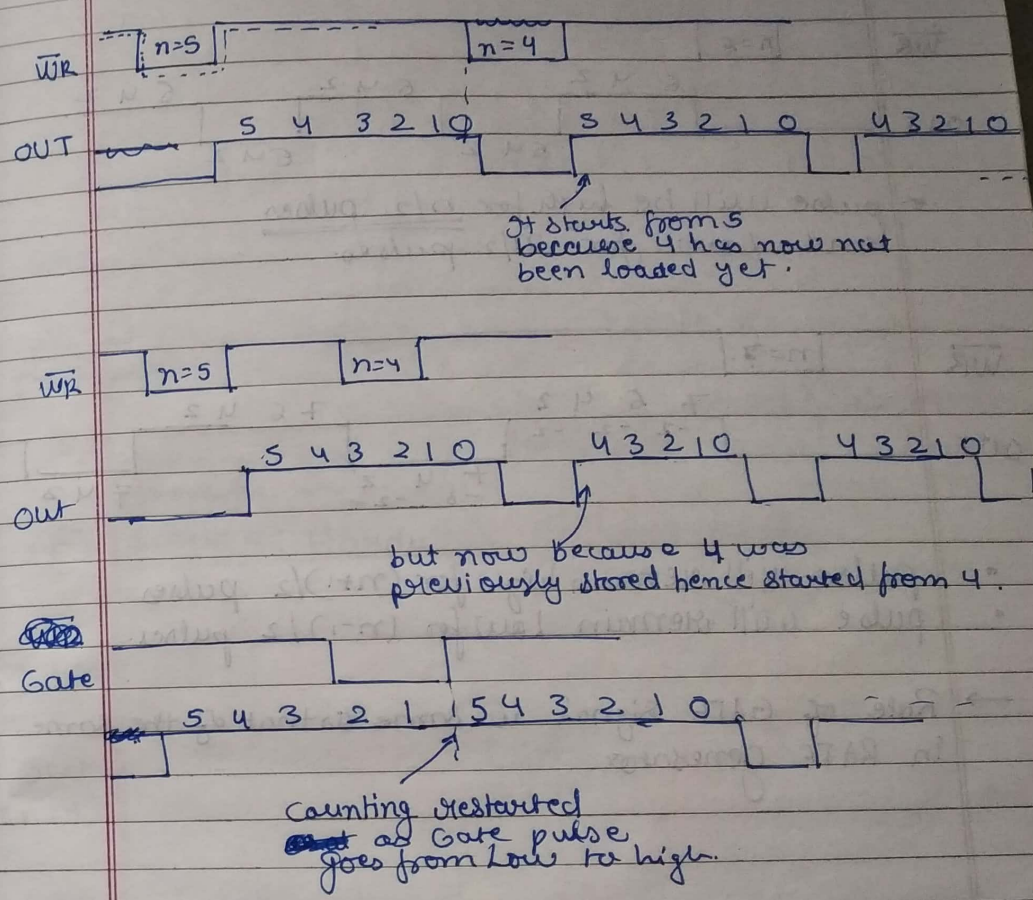


- Whenever gate signal goes high it starts counting from 'n'.

(C) ~~MOD~~

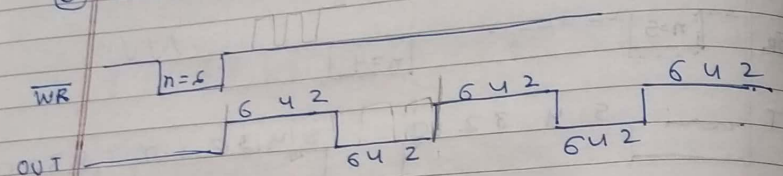
Date: / /
Page No.

(C) MODE 2: RATE Generator.

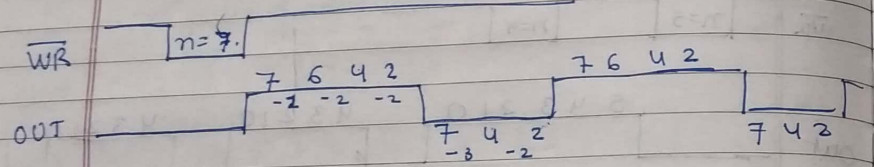


Date: / /
Page No.

(D) MODE 3: Square wave generator.

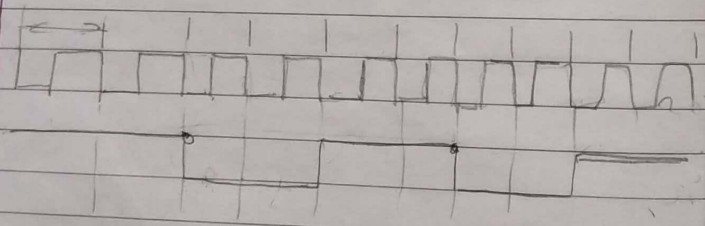


* pulse will be high for n/2 pulses and low for n/2 pulses.



- pulse will remain high for (n+1)/2 pulses
- pulse will remain low for (n-1)/2 pulses.

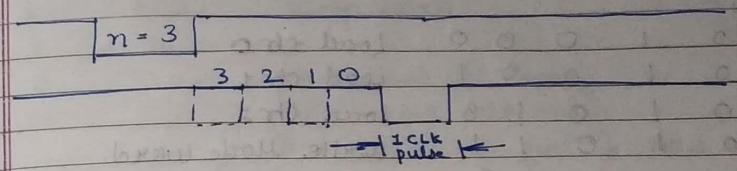
→ Rate of GATE signal is same as that of the same in RATE generator.

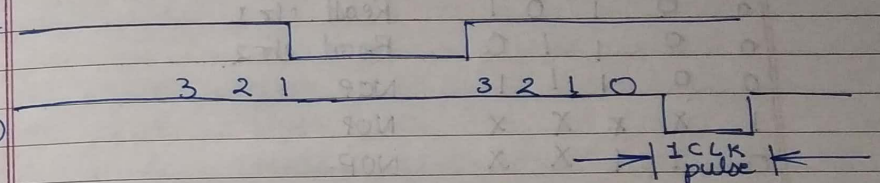


$4 f_{out} = f_{clk} \Rightarrow n = 4$

Date: / /
Page No.

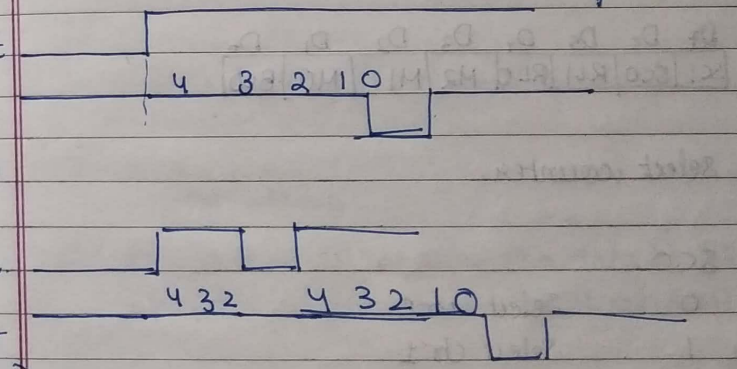
(E) MODE 4: Software triggered ~~strobe~~ Strobe





- Counting starts when ~~[illegible]~~ count is loaded (through $\overline{WR}$)

(F) MODE 5: Hardware Triggered Strobe.



- Counting will not start unless gate signal goes high

Date: / /
Page No.

→ Counter / Control Reg Selection.

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	
0	1	0	0	0	Load ctr 0
0	1	0	0	1	Load ctr 1
0	1	0	1	0	Load ctr 2.
0	1	0	1	1	Write Mode word.
0	0	1	0	0	Read ctr 0
0	0	1	0	1	Read ctr 1
0	0	1	1	0	Read ctr 2.
0	0	1	1	1	NOP
1	X	X	X	X	NOP
0	1	1	X	X	NOP.

control

Programming – 8253

→ Control word Format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

→ SC: Select counter.

SC1	SC0	
0	0	Select ctr 0
0	1	Select ctr 1
1	0	select ctr 2,
1	1	Invalid.

Date: / /
Page No.

RL – Read / Load.

→

RL1	RL0	Counter Latch.
0	0	
0	1	Read / Load LSB only
1	0	Read / Load MSB only.
1	1	Read/Load LSB first & then MSB.

→ M_2 M_1 M_0

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.

→ BCD

0 – Binary counter.
1 – BCD counter

→ 8086 instruction.

Port address

```
IN AL, 83H  ; whatever the data at 83H put it in AL
OUT 83H, AL ; put value in AL at 83H.
```

Date: / /
Page No.

Q) Write an instruction sequence to setup the 3 counters of 8253 located at IO addess of 40H as follows:

(a) Ctr 0 ← Binary counter operating in MODE 0 with an initial value of 1234H.

(b) Ctr 1 ← BCD counter operating in Mode 2 with an initial value of 200H.

(c) Ctr 2 ← Binary counter operating in Mode 4 with an initial value FFFH.

Sol It is given that counter at 40H. It means that

ctr 0 – 40H
Ctr 1 – 41H
Ctr 2 – 42H
C.WR – 43H.

· first task is to obtain C.W.R,

(a) CWR will have

$D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0$
0 0 1 1 0 0 0 0 = 30H.

(b) 01 11 x 10 1

(c) 10 11 1000
B 8

Date: / /
Page No.

```
MOV AL, 30H
OUT 43H, AL      (a)
MOV AL, 66H
OUT 43H, AL      (b)
MOV AL, B8H
OUT 43H, AL      (c)

MOV AL, 34H
OUT 40H, AL
MOV AL, 12H
OUT 40H, AL

MOV AL, 00H
OUT 41H, AL
MOV AL, 01H.
OUT 41H, AL.
~~MOV AL,~~

MOV AL, FFH
OUT 42H, AL
MOV AL, 1FH
OUT 42H, AL.
```

Q) Design a programmable timer using 8253 and 8086 interface 8253 at an address 40H for counter 0 and write the following ALP:

· 8086 runs at 6 MHz and 8253 runs at 1.5 MHz.

(a) to generate a square wave of period 1 millisecond.

(b) To interrupt a processor after 10 ms.
(c) To derive a monoshot pulse with a duration of 5 ms.

Sol (a) CLW = 00110111 = 37H.

MOV AL, 37H
OUT 43H, AL

MOV AL, 00H
OUT 40H, AL
MOV AL, 15H
OUT 40H, AL.

(b) CLW = 00110000 = 30H.

MOV AL, 30H
OUT 43H, AL.

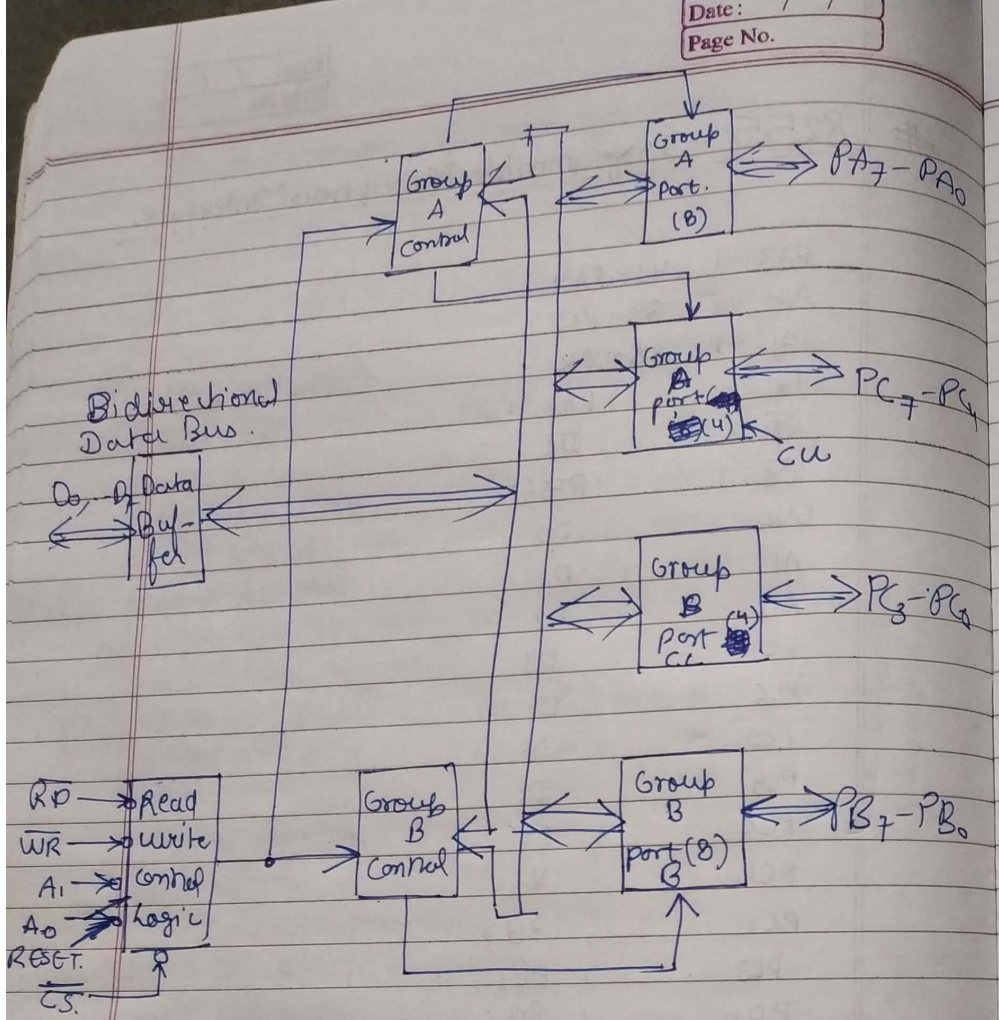
1500
33

MOV AL, 98H
OUT 40H, AL
MOV AL, 8AH
OUT 40H, AL

(c) CLW = 00110011 = 33H.
MOV AL, 33H
OUT 43H, AL
MOV AL, 00H
OUT 40H, AL
MOV AL, 35H.
OUT 40H, AL.

8255 - Programmable Peripheral Interface.

PA3	1	40	PA4
PA2	2	39	PA5
PA1	3	38	PA6
PA0			PA7
$\overline{RD}$			$\overline{WR}$
$\overline{CS}$			RESET
GND			D0
A1			D1
A0			D2
PC7			D3
PC6			D4
PC5			D5
PC4			D6
PC0			D7
PC1			V_{cc}
PC2			PB7
PC3			PB6
PB0			PB5
PB1	19	22	PB4
PB2	20	21	PB3



→ BSR – Bit set Reset

- works spec. port C to set or reset individual ~~mode~~ pin

→ I/O → MODE 0 ← used as simple I/O port.

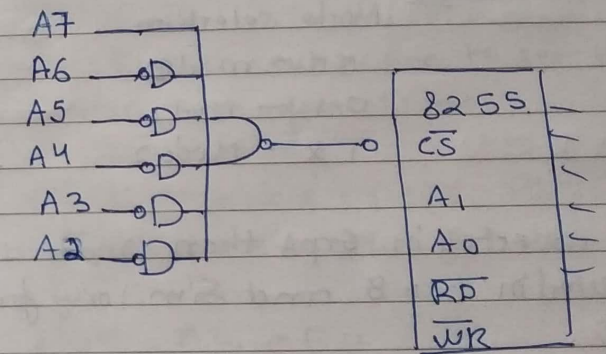
→ MODE 1 ← hand shake mode where Port A/Port B or both use bits of port C to hand shake.

→ MODE 2 ← Port A used for ~~signals~~ bidirectional data transfer and Port B can be setup either in mode 0 or Mode 1. uses handshake signals in Port C.

→ each pin of PA, PB & PC can act as input and output pins.

→ ~~Sig~~ handshake are signal ~~such~~ btw fast MPR and slow MPR / Peripheral to synchoronize them.

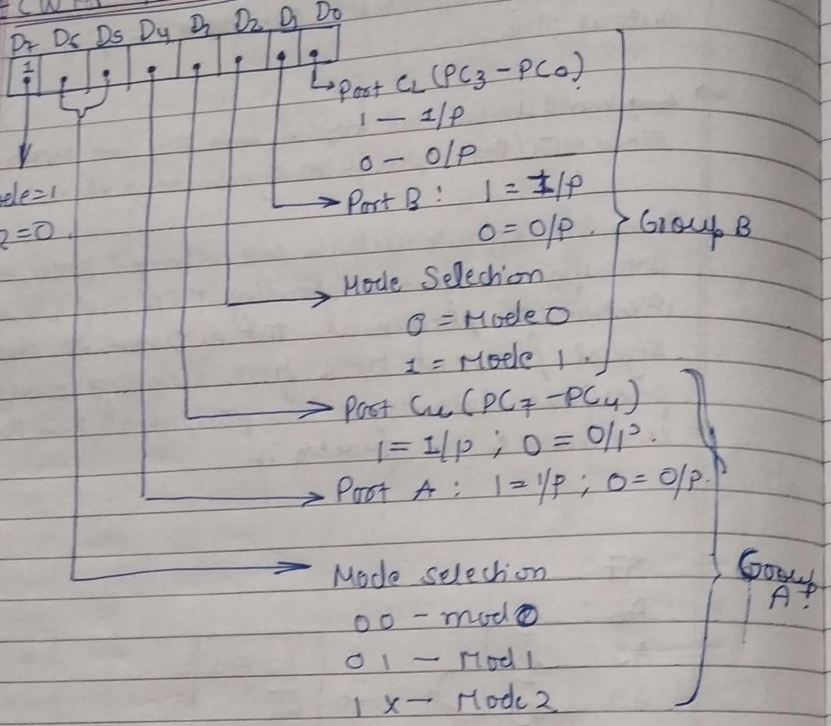
these are signals which are xchanged prior to data transfer btw fast responding MPR and slow responding Peripheral. The exchange of these signal prevent MPR from writing over the previous data before the peripheral has had a chance to process it



CS								Hex Address	Port
A7	A6	A5	A4	A3	A2	A1	A0		
1	0	0	0	0	0	0	0	80	A
						0	1	81	B
						1	0	82	C
						1	1	83	Control Reg.

Date: / /
Page No.

CW for I/O mode

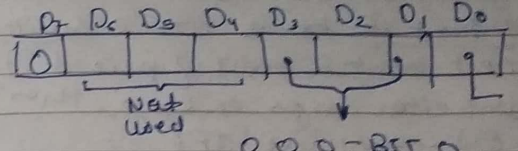


if mode 1 is selected in GrpA then mode 1 will be selected in Grp B. and Similarly for Mode 0.

But Mode 1 in Group A then Mode 0 and 1 can be Selected in group B.

Date: / /
Page No.

CW for BSR mode (works only for port C)



000 - BIT 0
001 - BIT 1
010 - BIT 2
011 - BIT 3
100 - BIT 4
101 - BIT 5
110 - BIT 6
111 - BIT 7.

0111

a) Write BSR CW subroutine to set bit PC7 and PC3 and reset them after 10 ms.

Sol

	S	R
CW: 0 x x x 1 1 1 1 PC7 =	0F	0E
CW: 0 x x x 0 1 1 1 PC3 =	0F	06

```
MOV AL, 0FH
OUT 83H, AL
MOV AL, 0EH
OUT 83H, AL
CALL DELAY ; 10 ms delay
MOV AL, 0EH
OUT 83, AL
MOV AL, 06H
OUT 83, AL
```

Date: / /
Page No.

8255

Q) An 8086 base micro computer is required to drive an LED connected to bit-2 of Port B base on 2 switch inputs connect to bit 6 and 7 of port A. If both switches are either high or low, then turn Led ON otherwise OFF.

Sol 10010000 = 90 H. ; set up port A as input and Port B as output.

```
** MOV AL, 90H
   IN
   OUT 83H, AL
   IN 80, AL, 80
   CMP AL, COH.
   JE *
   JMP **
*  OUT 81, 04.
   JMP **.
```

00 H
C0 H
40 H
80 H

```
original  MOV AL, 90H
          OUT 83H, AL
BEGIN:    IN AL, 80
          AND AL, COH.
          JPE *
          MOV AL, 00H.
          OUT 81H, AL
          JMP BEGIN.
*         MOV AL, 04H
          OUT 81H, AL
          JMP Begin
          HLT.
```

Date: / /
Page No.

Q) Interface 8 Keys and LED with 8086 through 8255. Write a program to flash 8 LEDs connected to port B until port C becomes FFH & port address of Port A = A8H.

Sol CW = 10001001 = 89H.

```
   MOV AL, 89H
   OUT ABH, AL.
*  MOV AL, FFH.
   OUT A9H, AL
   CALL DELAY; is delay.
   MOV AL, 00H.
   OUT A9H, AL.
   JMP BEGIN.
BEGIN: → IN AL, AAH.
   CMP AL, FFH.
   JNE*
   HLT
```

Q) To interface Stepper motor using 8086 and 8255

Motion	Step	Hex value
clockwise	1	03
	2	06
	3	0C
	4	09
	5	03
	1	03
anticlock wise	2	09
	3	0C
	4	06
	5.	03

Page No.

Rotate 8H coils in clockwise and anti clockwise direction.

10000000 = 80H

MOV AL, 80H
OUT 83H, AL
MOV

6) Interface an 8 bit DAC to 8086 through 8255 and write the following program:
(a) to generate a square wave
(b) to generate Triangular wave.

use Port A as 90 H.

MOV CX, FFFFH

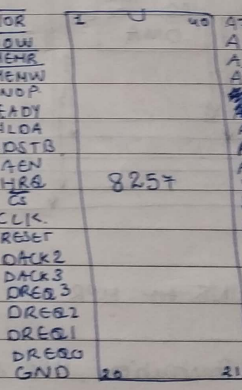
MOV AL,
MOV AL, FFH
OUT 90H, AL
CMP AL, 00H
JE *

Date: / /
Page No.

MOV AL, FFH.

CMP BX, [illegible] MOV AX
DR*
CMP AL, 00H.
JE*
MOV BL, 00H. JMP **
* MOV BL, ffH
** CMP BL, ffH
JE *2
DEC AL
JMP *3
*2 INC AL
*3 OUT 90H, AL
CALL DELAY;
JMP *4

DMA - 8257 (direct memory acess controller)



- transfers bytes of data bw peripheral to RAM through MPR
- It takes 1 μs to transfer 1 byte of data whereas MPR takes > 5 μs for the same operation.

- 2 modes → Block transfer DMA → when peripheral is much faster than MPR, entire block of data is transfered and then MPR continues.
  → Cycle Stealing → associated with 1 byte of data → when MPR is in wait state, it performs data transfer. (Transfering small data).

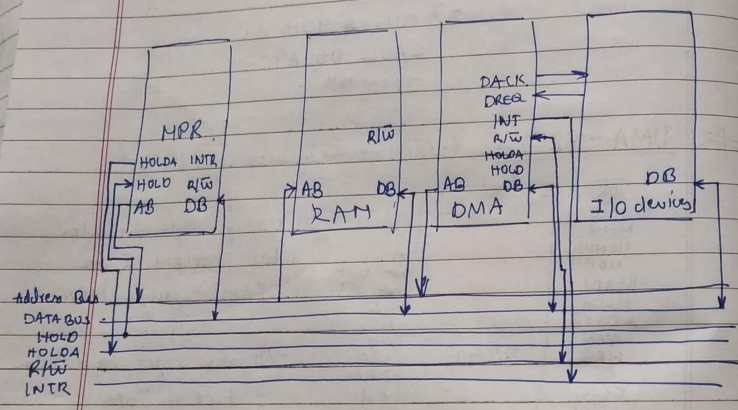
Date: / /
Page No.

3 Register → Address Register → store the address from the locn data has to transfer. automatically execute when data transfer.

→ Terminal Count Reg → Stores the length of data that has to be transfered. (Whenever 1 byte of data is transfered, this Reg is dec. by 1)

→ Status Reg → gives info. if DMA is processing / over.

→ MPR sends req. to DMA to transfer data.
→ In response DMA sends HOLA signal.
→ When all current inst. are over then MPR will give HOLDA signal to DMA



→ On DREQ, DMA sends INT to MPR and HOLD to MPR.
→ after completion of current instruction, MPR send HOLDA
→ When DMA recieves HOLDA, DMA send DACK to IO device.
→ DMA send Address to address bus which is recieved by RAM

Date: / /
Page No.

8257

ADSTB ← address strobe is similar to ALE of 8086.
HRQ ← hold request. Similar to HOLD of 8086
EOP ← End of process - when DMA transfer is over then it goes high.

ADSTB ← Enable. DMA, adress latch connected to address line in to DB7 to DB0

→ higher order address bus of 8086
→ 8086 has address and data bus multiplexed.
→ ADSTB works similar to 8086 ALE but with higher order adress bys of 8086.

basically it locks the value at higher order addre bus. Higher order bits are not used in DMA but are just locked.

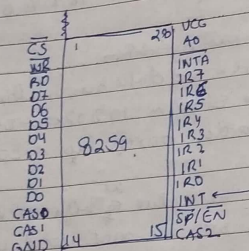
A_0-A_7 and DB_0-DB_7 are both connected to lower order addressbus of 8086 and info that signal on that are data or address is represed by AEN signal.

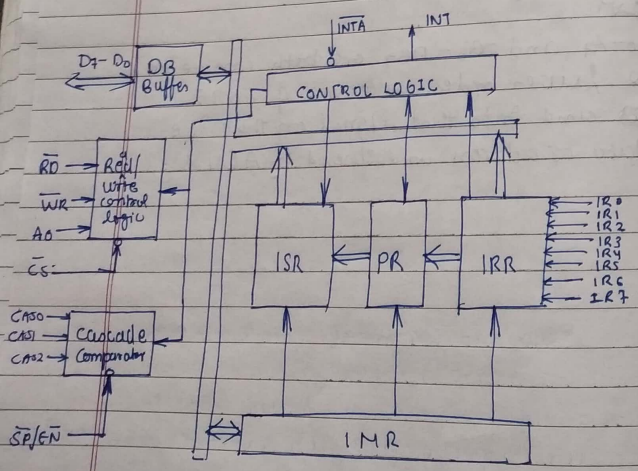
Date: / /
Page No.

8259 - Priority Interrupt Controller

→ ~~Arch~~ Acts as a manager in an interrupt driven systems

→ Can handle upto 8 levels of interrupts which can further be cascaded upto 64 levels of interrupt request.





Date: / /
Page No.

IR0 - IR7 ← interrupt request.

SR|EN ← mode ↔ ~~spare~~ slave enable (master)

- the slave program pin is used to specify if 8259 is working as master or slave. If this pin is high then 8259 work as master otherwise slave.

IR0 - IR7 ← 8 interrupt request given by 8259 8 independent ~~devices~~ through INT pin of each of them.

→ 8259 resolves priority of different interrupt and then sends to ~~MPR~~

→ CAS0 - CAS2 ← for a master 8259 CAS pins are output pins and for a slave 8259 CAS pins are input.

- When 8259 is a master i.e. when it accepts interrupt req. from other 8259's, the CALL opcode is generated by the master in response to the first ~~Ack~~ acknowledge interrupt.

∴ The ~~vector~~ vectoring address is given

- The master puts out an id code of 3 bits on the CAS0-2 lines
- The slave 8259 accepts these 3 signals as input and compare the code put out by the master with the codes assigned to them during initialization. The slave thus selected which had originally placed an interrupt request to master 8259 put out the address during the IInd ~~RD~~ INTA pulse.

Date: / /
Page No.

IRR ← Interrupt Request Register
PR ← Priority Resolver.
- resolves priority of IR based on mode.

ISR ← In Service Register
IMR ← Interrupt Mask Register.
- If a lower priority interrupt is required to service then it is used as it masks a higher priority interrupt.

→ Mode 0: fully nested mode. (default)

- In this mode, the highest priority is assigned to IR0 and lowest priority to IR7

→ Automatic Rotation mode

This mode is used in an interrupt structure, where the interrupt must be assigned equal priority.

The last serviced interrupted IRX is automatically lowest priority then the next interrupt IR(X+1) is assigned highest priority

→ Specific Rotation mode.

~~last serviced interrupt is assigned lowest~~
Lowest priority can be assigned to any of the interrupt levels and the next interrupt level/reg is automatically assigned highest priority.

Date: / /
Page No.

→ Special Mask Mode.

~~In this mode of operation~~,
This mode is used to selectively enable lower priority interrupt level and simultaneously mask higher priority interrupt.

→ ISR: It is used to store information about the interrupt currently being serviced.

→ ~~PR~~ → PR: It determines the priority of interrupt requesting service based on priority mode set out in the command words

→ IRR: This is used to store information about the interrupt input requesting service

→ IMR: This Reg. can be programmed by the command word to store the bits which masks the specific interrupt. So the interrupts that are being masked by programmer in IMR will not be recognized and serviced even if it sets the corresponding bits in IRR.