ELEG 3230B Microprocessors and Computer Systems

Part 10 Interrupts

(PIC materials, D. Hall Ch8, Brey Ch12)

Interrupts Revisited

- Three basic categories of interrupts:
  1. Software interrupts: generated by command INT *interrupt-vector*
  2. Hardware interrupts: generated by asserting the NMI or INTR pin
  3. Automatic interrupts: generated by certain error conditions (eg. divide by 0), or at the end of every instruction if the TRAP flag is set in the FLAGS register.

- Refer to earlier lecture notes on software interrupts using the INT instruction to review the interrupt process (push FLAGS, clear IF, TF, push CS, push IP, load interrupt vector into CS:IP). All interrupts are processed in the same way.

Interrupts Revisited (cont.)

- Hardware interrupts are either maskable (i.e. INTR is only recognized if the interrupt flag in the FLAGS register is set) or non-maskable (ie. NMI is processed irrespective of the value of the interrupt flag).
- Hardware interrupts allow peripherals to be serviced by the microprocessor only when they need attention; the microprocessor can perform other tasks when the peripheral does not need the dedicated attention of microprocessor.
- (Read part 6 again about Interrupt related instructions).

Interrupts related Instructions

Mnemonic	Meaning	Format	Operation	Flags Affected
CLI	Clear interrupt flag	CLI	$0 \rightarrow (IF)$	IF
STI	Set interrupt flag	STI	$1 \rightarrow (IF)$	IF
INT *n*	Type n software interrupt	INT *n*	$(Flags) \rightarrow ((SP - 2)$ $0 \rightarrow TF, IF$ $(CS) \rightarrow ((SP) - 4)$ $(2+4\cdot n) \rightarrow (CS)$ $(IP) \rightarrow ((SP) - 6)$ $(4\cdot n) \rightarrow (IP)$	TF, IF
IRET	Interrupt return	IRET	$((SP)) \rightarrow (IP)$ $((SP) + 2) \rightarrow (CS)$ $((SP) + 4) \rightarrow (Flags)$ $(SP) + 6 \rightarrow (SP)$	All
INTO	Interrupt on overflow	INTO	INT 4 steps	TF, IF
HLT	Halt	HLT	Wait for an external Interrupt or reset to occur	None
WAIT	Wait	WAIT	Wait for $\overline{\text{TEST}}$ input to go active	None

(part-06, page 76)

Usually connected to $\overline{\text{BUSY}}$ of 8087

Some Special Types of Interrupts in the Intel Series

- Type 0 : Divide error interrupt

  Interrupt number 0 (interrupt vector is located at memory locations 00000 to 00003) is generated when an attempt is made to divide by zero. The interrupt is generated automatically.

- Type 1: Single-Step Interrupts

  When bit-8 of the FLAGS register (trap flag) is set to 1, an interrupt number 1 is generated after the execution of every instruction. When the microprocessor enters the interrupt service routine, the T flag is automatically cleared (otherwise it would not be able to go beyond the first instruction of the interrupt service routine!).

Some Special Types of Interrupts in the Intel Series (cont.)

- Type 3: Breakpoint (One-byte) Interrupt

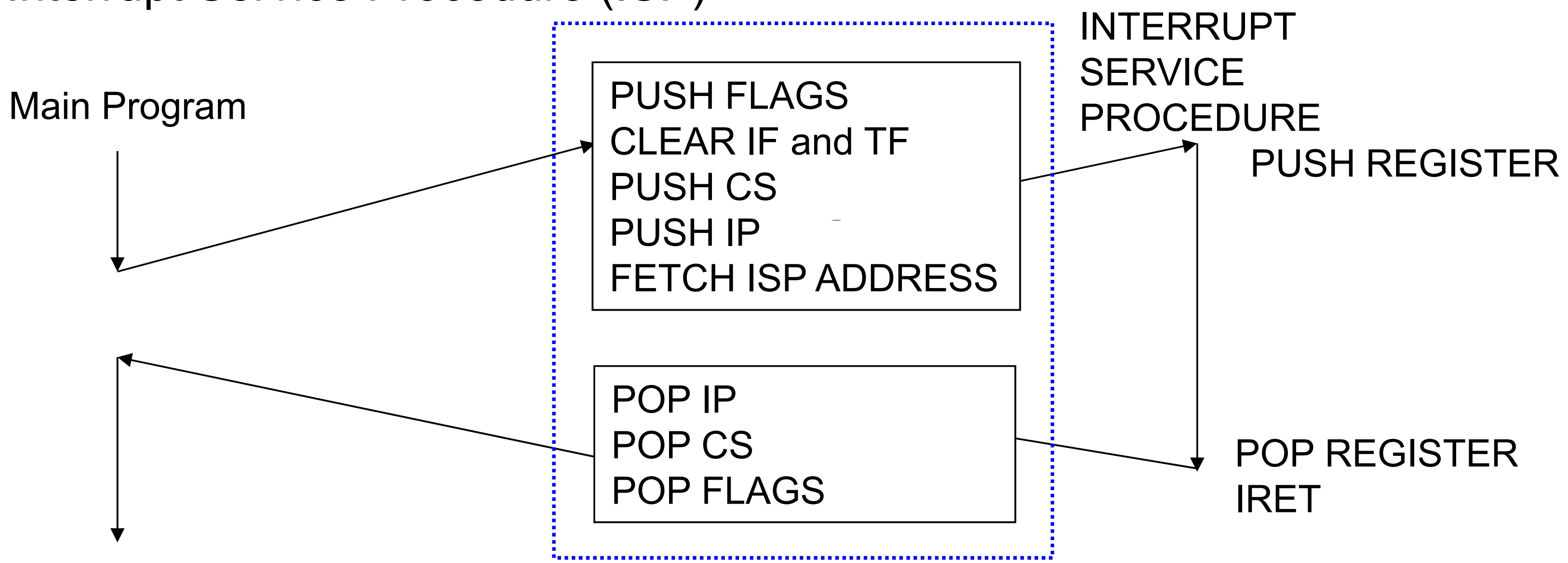
  Interrupt number 3 has a compact single byte opcode (CCh) which may be inserted anywhere in a program to generate a breakpoint in a program. This is also used for debugging, similar to Type 1 interrupt. However Type 3 it will stop at where you insert the INT3, not after every step.

- Type 6: Undefined Opcode Interrupt

  When the microprocessor encounters an opcode that it does not recognize, it generates interrupt number 6 (generated only in 80386 and later microprocessors).

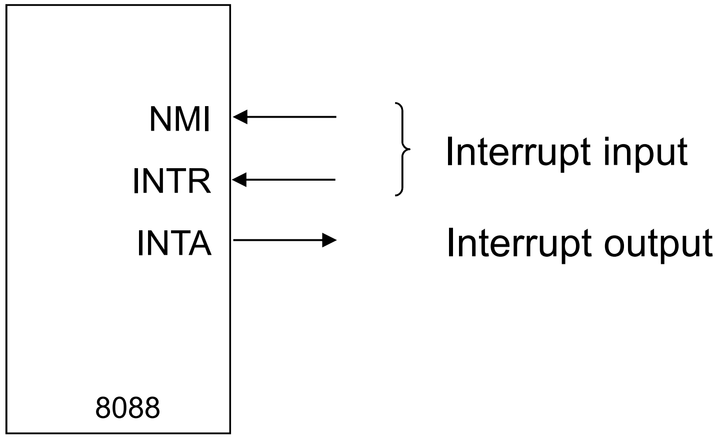
Interrupt Response Sequence

1. PUSH Flag Register
2. Clear IF and TF
3. PUSH CS
4. PUSH IP
5. Fetch Interrupt vector contents and place into both IP and CS to start the Interrupt Service Procedure (ISP)



8088 Hardware Interrupts

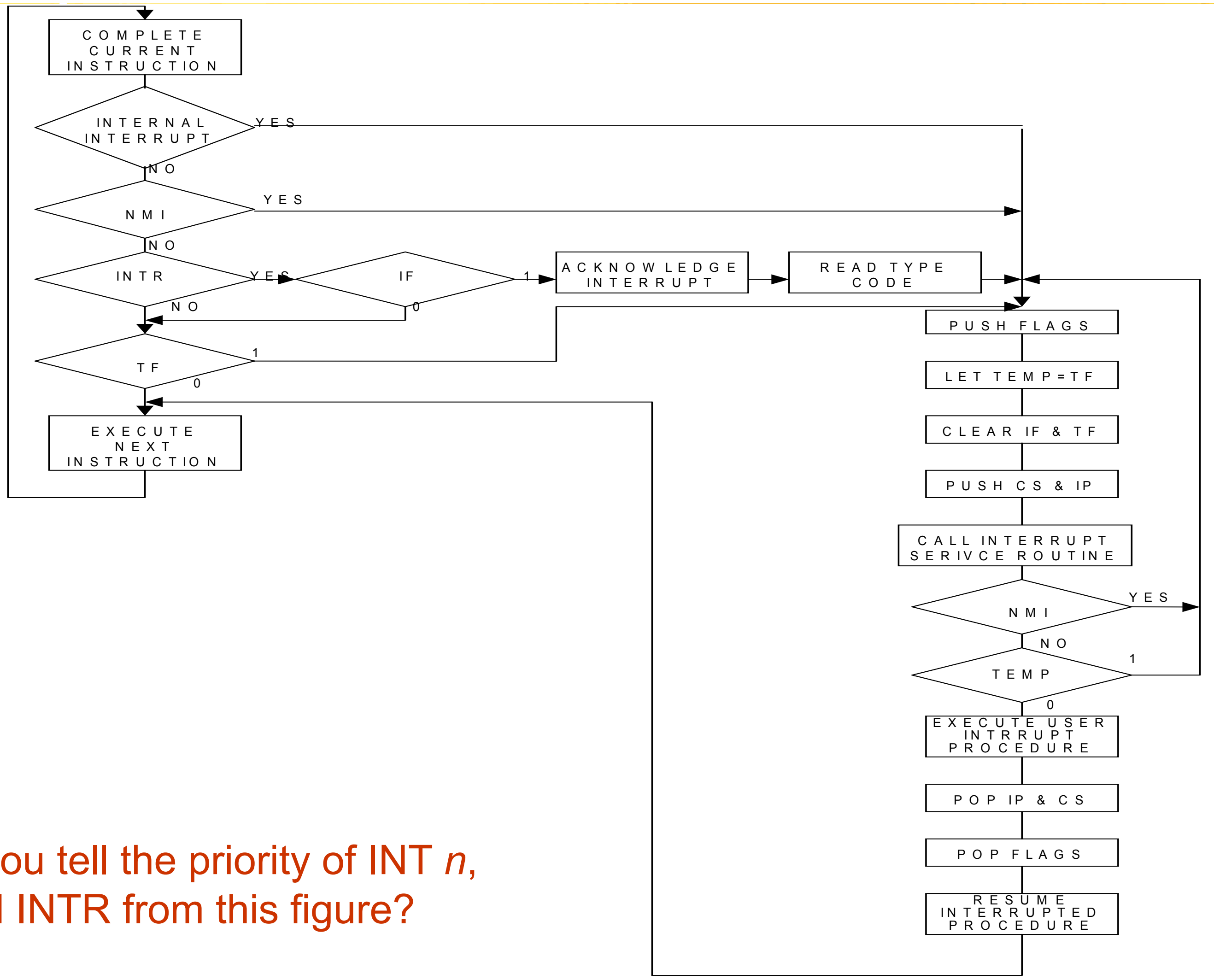
- Hardware interrupts are very efficient in handling peripheral devices, particularly I/O operations, since the processor can perform other tasks and only services the peripheral when required.
- The NMI is usually reserved for the most urgent type of interrupt (eg. imminent power failure of a peripheral), whereas the INTR is used for "normal" interrupts (where the peripheral can wait, if necessary, until the interrupt flag is set).



8088 Hardware Interrupts (cont.)

- Asserting the NMI pin generates interrupt number 2 (new offset address IP is stored in 00008 and 00009; new segment address CS is stored in 0000Ah and 0000Bh). NMI is edge triggered (the interrupt is generated only at the 0-to-1 transition of the input signal on the NMI pin).
- When more than one peripheral are connected to the NMI pin, the interrupt service routine may need to check which peripheral generated the NMI (eg. by *polling* the possible sources of the NMI).
- Interrupt request pin INTR is level sensitive - it must be held at logic 1 until it is acknowledged by INTA. INTR is automatically disabled when the microprocessor is already servicing an INTR (this can be changed, though), and INTR is re-enabled at the end of the interrupt service routine.
- The interrupt number generated by an INTR is read from the data bus.

Interrupt processing sequence of the 8088 microprocessor



Q: Can you tell the priority of INT *n*, NMI, and INTR from this figure?

Interrupt Priority

- When different type of interrupt (ie. software, NMI, INTR or automatic interrupts) occur at the same time, the one with the highest priority is handled first. Intel microprocessors use the following order of priority:

Interrupt Type	Priority
divide error interrupt, INT n, INT0	highest
NMI	↕
INTR	
TRAP flag (single step)	lowest

(See Reference Note on Interrupt)

Interrupt Priority (cont.)

⌘ If several INTR are generated from different peripherals simultaneously, it is necessary to decide their priority and send the INTA signal only to the highest priority peripheral. (If more than one peripheral receive an INTA at the same time they will both try to put the interrupt number onto the data bus simultaneously, leading to collision).

How?

⌘ Two different methods can be used to establish the priority of interrupt requests from different peripherals

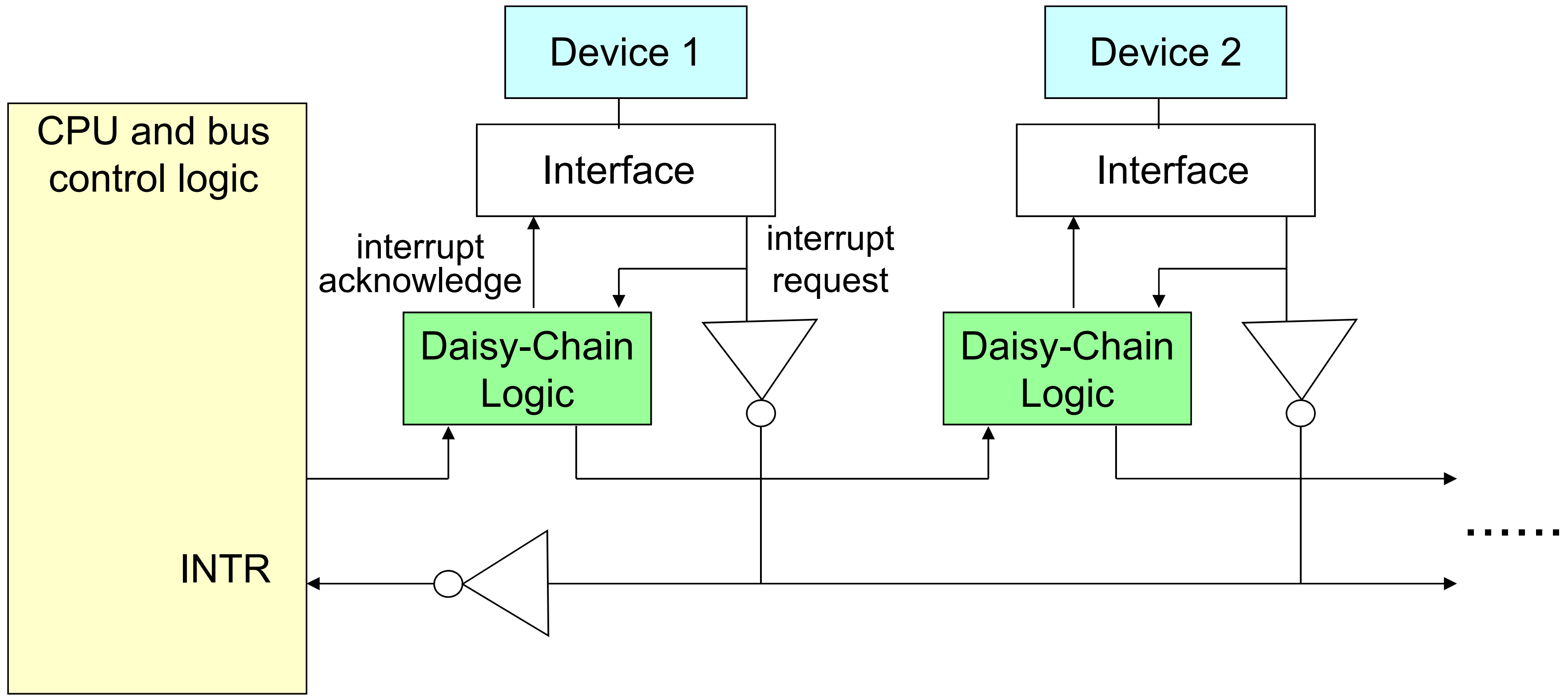
1. Polling (and daisy chaining)
2. Interrupt priority management hardware (e.g. priority encoder 74LS148, or more advanced 8259 programmable interrupt controller)

Priority Allocation by Polling and Daisy-Chaining

- Polling involves asking each peripheral, in a predetermined order, whether it needs attention from the microprocessor. The first peripheral which responds “yes” is served by the appropriate routine.

- Daisy-chaining is a method of implementing the polling scheme by hardware. The INTA signal passes along a series of peripherals from one peripheral to the next only if the peripheral is not requesting an interrupt. Hence the first peripheral in the daisy-chain has the highest priority and the last peripheral has the lowest priority.

Priority Allocation by Daisy-Chaining



⌘ Daisy chaining may be combined with software polling to determine which interrupt routine is needed by the peripheral

Interrupt Priority Management

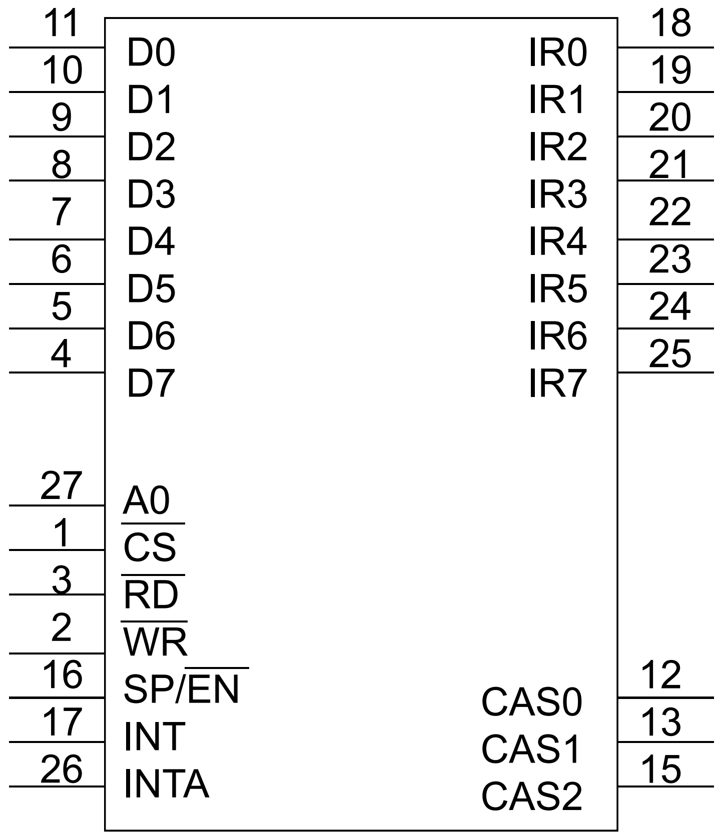
- The daisy chain method suffers the disadvantage of being limited in the number of devices which could be chained together because of the delays in passing the INTA through the chain. The microprocessor expects the interrupt number to be placed on the data bus within a certain time after the INTA is sent out. Too long a delay will lead to uncertain errors.

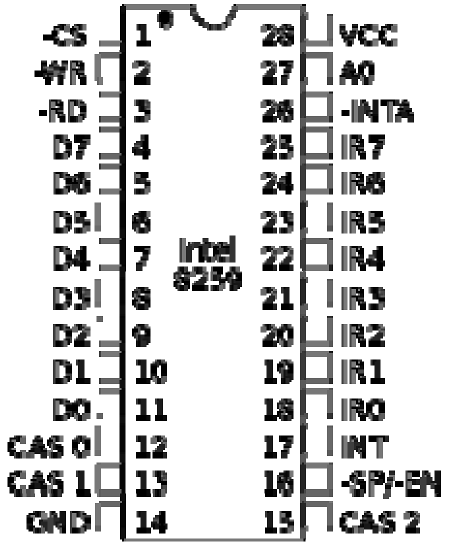
- In practical systems, usually a Programmable interrupt controller (8259A) is used to determine the priority of interrupts.

- The 8259A programmable interrupt controller assigns the priority of up to 8 vectored interrupt sources to be connected to the INTR pin. The 8259A may be cascaded (one master 8259A and eight slave 8259As) to provide 64 interrupt lines).

The 8259A Programmable Interrupt Controller (PIC)

⌘ 8259A is a 28-pin integrated circuit which was designed specifically for the 8088/8086 microprocessors.





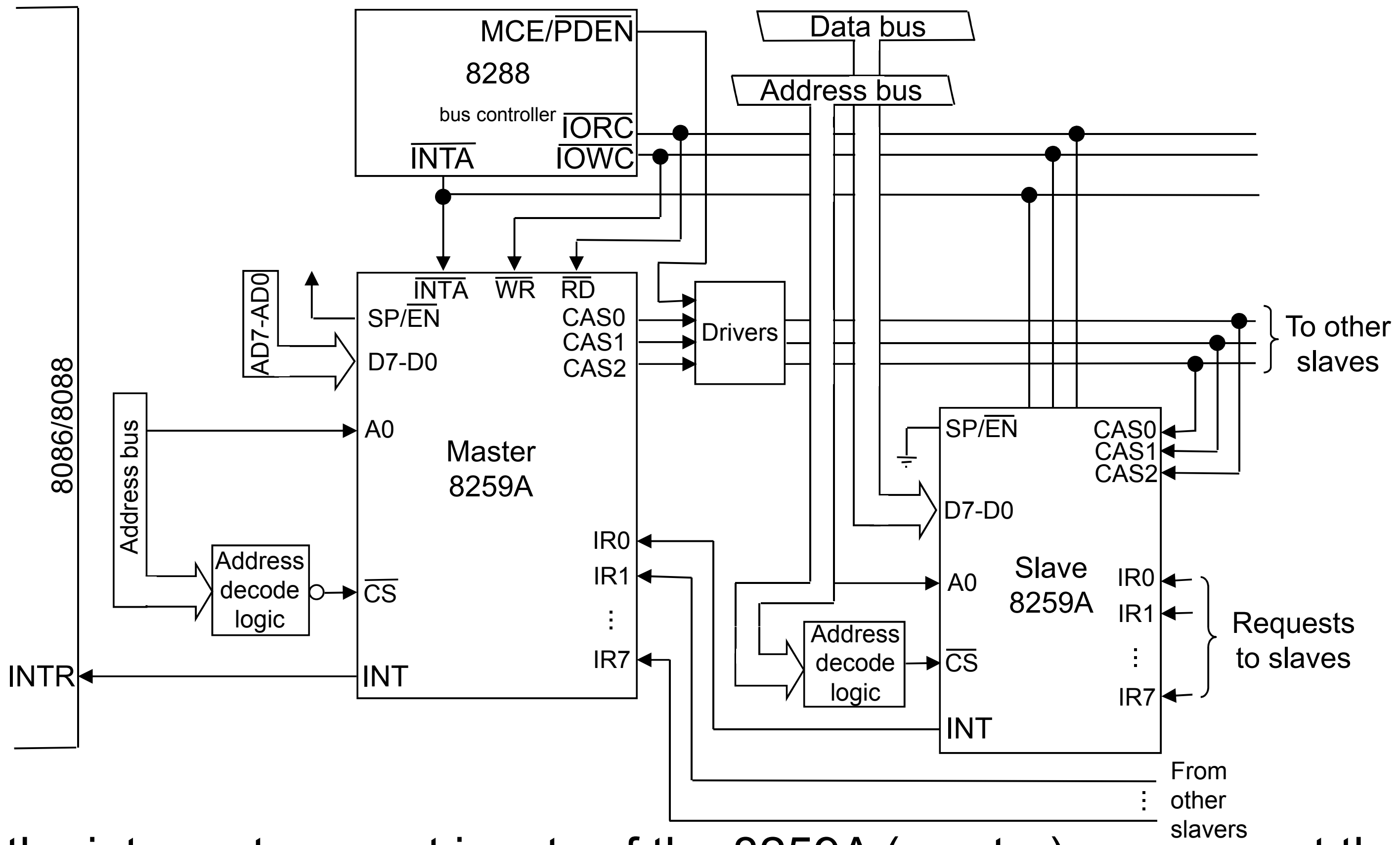
The 8259A Programmable Interrupt Controller (cont.)

- The 28 pins of the 8259A include:
  1. Data pins D0-D7 - connected to the data bus to allow programming
  2. IR0-IR7 - 8 interrupt inputs
  3. CAS0-CAS2 - cascade lines (used in a multi-8259A system)
  4. $SP/\overline{EN}$ - functions as data buffer enable output (when buffered mode) or as an input to program the 8259A as a master or slave
  5. A0 - input which selects different command words in the 8259A

- Usually IR0 has the highest priority and IR7 the lowest priority.

- Fully nested interrupts are supported (higher priority interrupts may interrupt the interrupt service routine of a lower priority interrupt) if bit-4 of ICW4 is set. (Refer to data sheets for further details.)

8259A Interrupt Controllers In Cascade



Each of the interrupt request inputs of the 8259A (master) may accept the INT output from another (slave) 8259A device, thus increasing the number of interrupt request lines beyond 8.

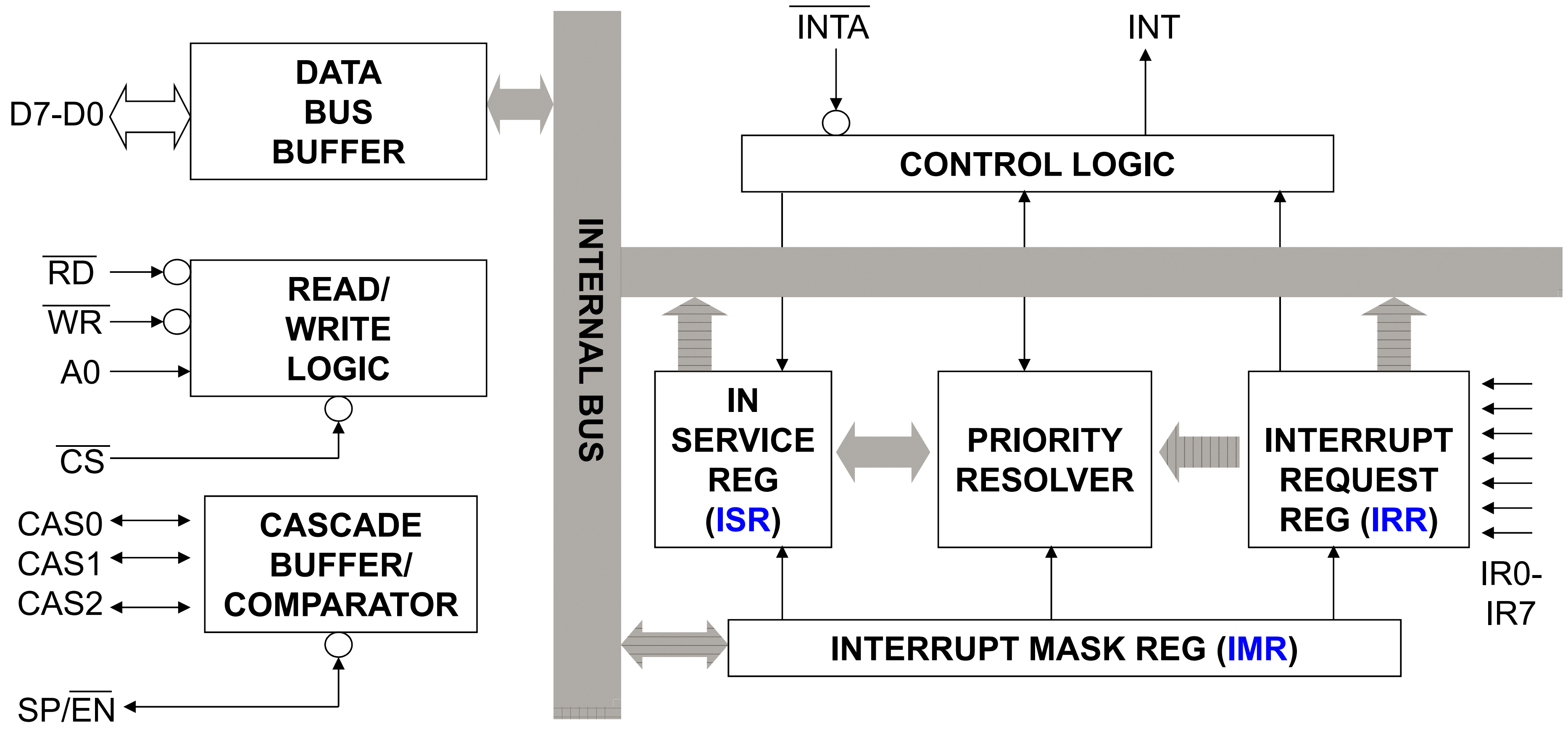
8259A Pin Description

Symbol	Pin No.	Type	Name and Function
V_{CC}	28	I	SUPPLY: +5V Supply
GND	14	I	GROUND
$\overline{CS}$	1	I	CHIP SELECT: A LOW on this pin enables the $\overline{RD}$ and $\overline{WR}$ communication between the CPU and the 8259A. INTA functions are independent of $\overline{CS}$.
$\overline{WR}$	2	I	WRITE: A LOW on this pin when $\overline{CS}$ is low enables the 8259A to accept command words from the CPU.
$\overline{RD}$	3	I	READ: A LOW on this pin when $\overline{CS}$ is low enables the 8259A to release data onto the data bus for the CPU.
D_7-D_0	4 -11	I/O	BIDIRECTIONAL DATA BUS: Control, status and interrupt-vector information are transferred via this bus.
CAS_0-CAS_2	12,13,15	I/O	CASCADE LINES: The CAS lines form a private 8259A bus to control a multiple 8259A structure. These pins are outputs for a master 8259A and inputs for a slave 8259A.

8259A Pin Description (cont.)

SP/$\overline{\text{EN}}$	16	I/O	SLAVE PROGRAM/ENABLE BUFFER: This is a dual-function pin. When in the Buffered Mode it can be used as an output to control buffer transceivers (EN). When not in the buffered mode it is used as an input to designate a master (SP =1) or slave (SP =0).
INT	17	O	INTERRUPT: This pin goes high whenever a valid interrupt request is asserted. It is used to interrupt the CPU, thus it is connected to the CPU's interrupt pin.
IR_0-IR_7	18-25	I	INTERRUPT REQUESTS: Asynchronous inputs. An interrupt request is executed by raising an IR input (low to high), and holding it high until it is acknowledged (*Edge Triggered Mode*), or just by a high level on an IR input (*Level Triggered Mode*).
$\overline{\text{INTA}}$	26	I	INTERRUPT ACKNOWLEDGE: This pin is used to enable 8259A interrupt-vector data onto the data bus by a sequence of interrupt acknowledge pulses issued by the CPU
A_0	27	I	A0 ADDRESS LINE: This pin acts in conjunction with the $\overline{\text{CS}}$, $\overline{\text{WR}}$, and $\overline{\text{RD}}$ pins. It is used by the 8259A to decipher various Command Words the CPU writes and status the CPU wishes to read.

Internal Architecture of 8259A



Internal Block Diagram

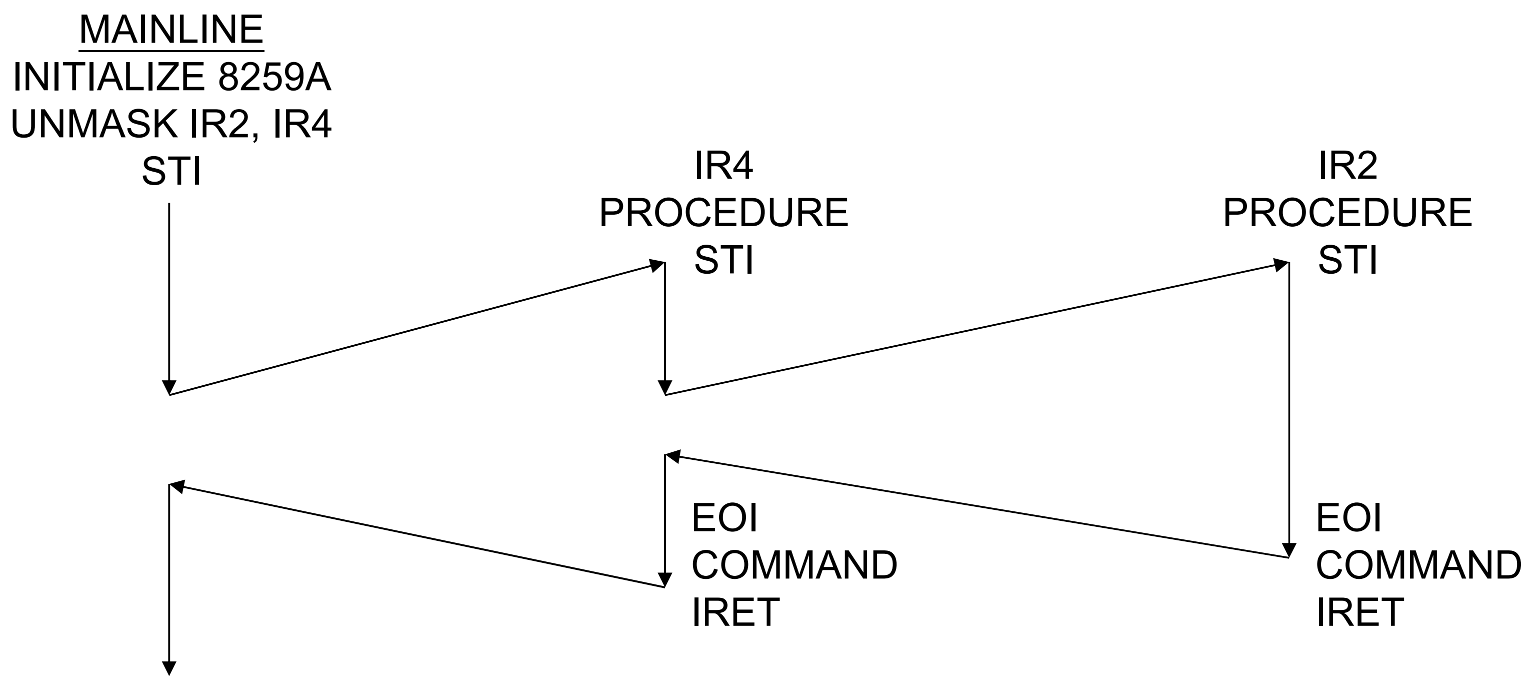
(D Hall Fig.8-27 or

W Triebel Fig. 9.11)

Internal Architecture of 8259A

- Data bus buffer: bidirectional three-state buffer through which MPU can access 8259A’s internal registers.
- R/W Logic: accept input $\overline{RD}$,$\overline{WR}$,A0, and $\overline{CS}$ to control the direction, timing, and source/destination of data transfer.
- IMR: used to mask out individual interrupt request input (IR0-IR7).
- IRR: stores the current status of the interrupt request input.
- Priority resolver: determine the interrupt priority of the active interrupt inputs.
- ISR: stores the interrupt level (meaning which pin of the IR0-IR7 pins) that is presently being served.
- Cascade buffer/ comparator: provides the interface between master and slave 8259A. Each slave has an ID code stored here.

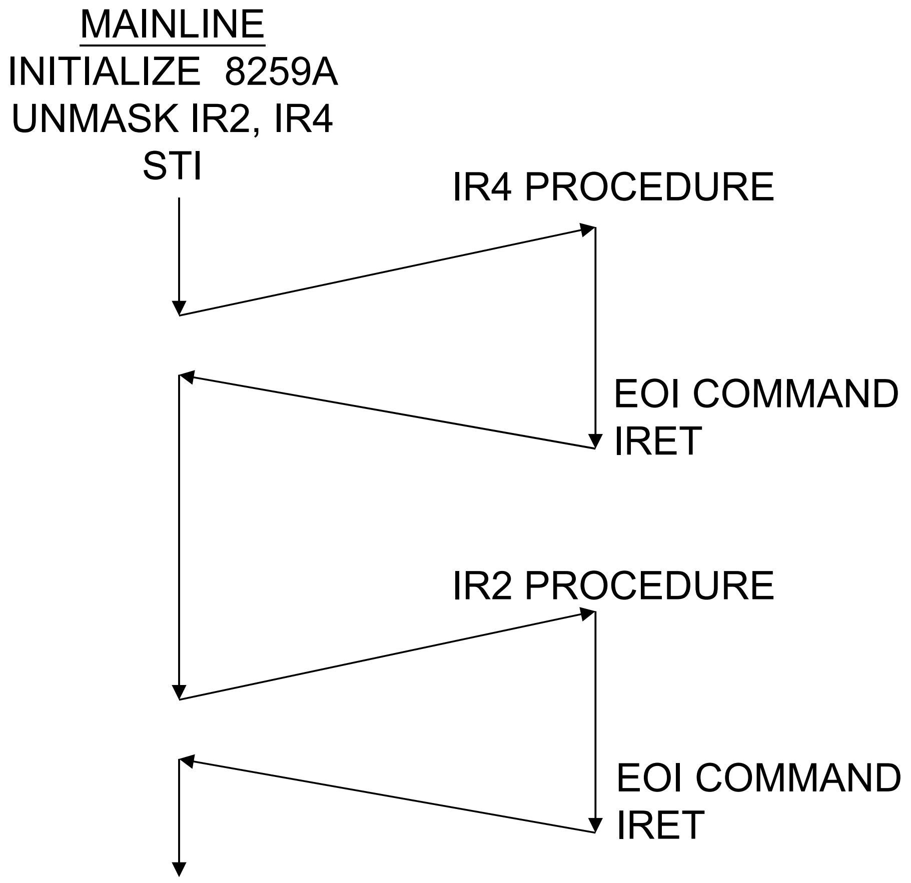
8259A and 8088 Program Flow For IR4 Followed by IR2



(a) Response with INTR enabled inside IR4 procedure (IF=1)

Q: what is the default value of I flag in an Interrupt Service Routine?

8259A and 8088 Program Flow For IR4 Followed by IR2 (cont.)



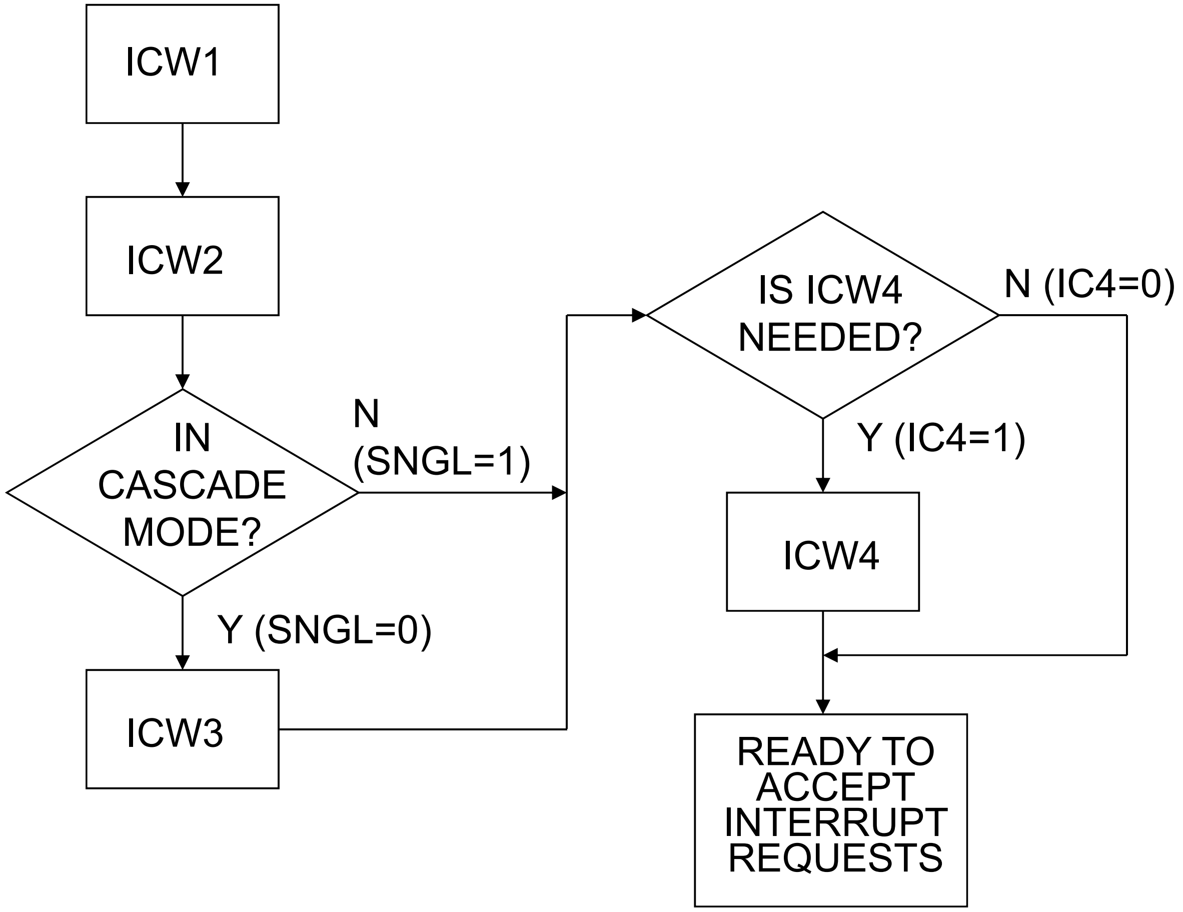
(b) Response with INTR **not** enabled inside the IR4 procedure (IF=0)

Programming the 8259A

(Brey's)

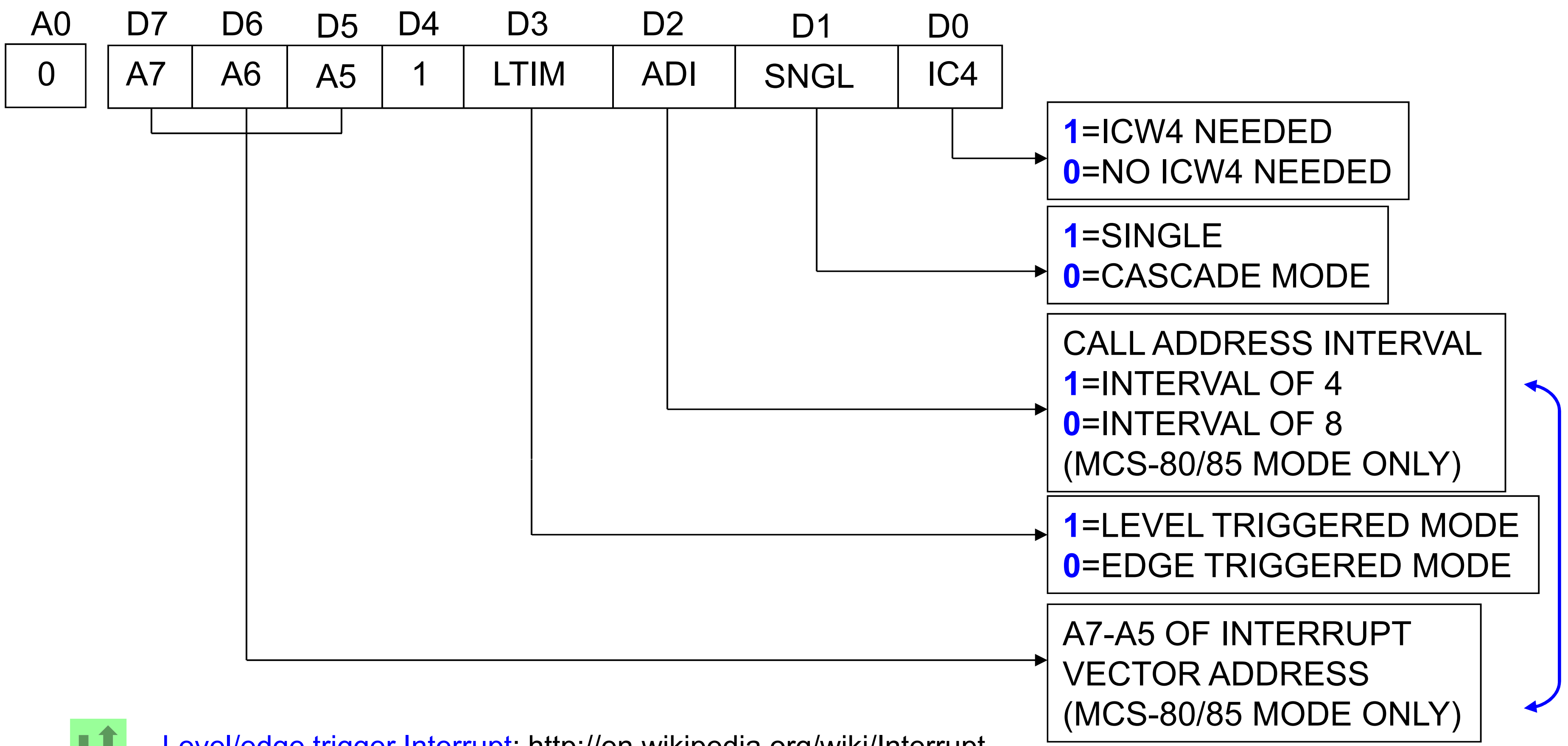
- Each 8259A can be programmed by writing appropriate bytes into its 7 internal registers (4 "initialization command words" ICW registers and 3 "operational command words" OCW).
- ICW commands are used to load the internal control registers of 8259A.
- OCW commands permit 8088 to initiate variations in the basic operating modes defined by ICW command.
- The ICW and OCW commands can be issued to 8259A using OUT (for I/O mapped) or MOV instruction (for memory mapped).
- The initialization sequence of ICW and OCW is shown next.

8259A Initialization Command Word Sending Order



8259A Initialization Command Word Formats ICW1

ICW1: used to program the basic operation of 8259a.



Level/edge trigger Interrupt: http://en.wikipedia.org/wiki/Interrupt

8259A Initialization Command Word Formats ICW2 & ICW3(master)

ICW2: is used to select vector number used with the interrupt request inputs.

A0
1

D7	D6	D5	D4	D3	D2	D1	D0
A15/T_7	A14/T_6	A13/T_5	A12/T_4	A11/T_3	A10	A9	A8

ICW2

A15-A8 OF INTERRUPT VECTOR ADDRESS (MCS80/85 MODE)
T_7-T_3 OF INTERRUPT VECTOR ADDRESS (8086/8088 MODE)

Note: The least three bits, T2-T0, are deduced from the pin number IR7-IR0.
Example: If the interrupt vectors are 0E8h-0EFh, then T7 toT3=11101

ICW3: is used only when ICW1 specifies that the system is operated in cascade mode; is used to indicate where the slave is connected to the master.

A0
1

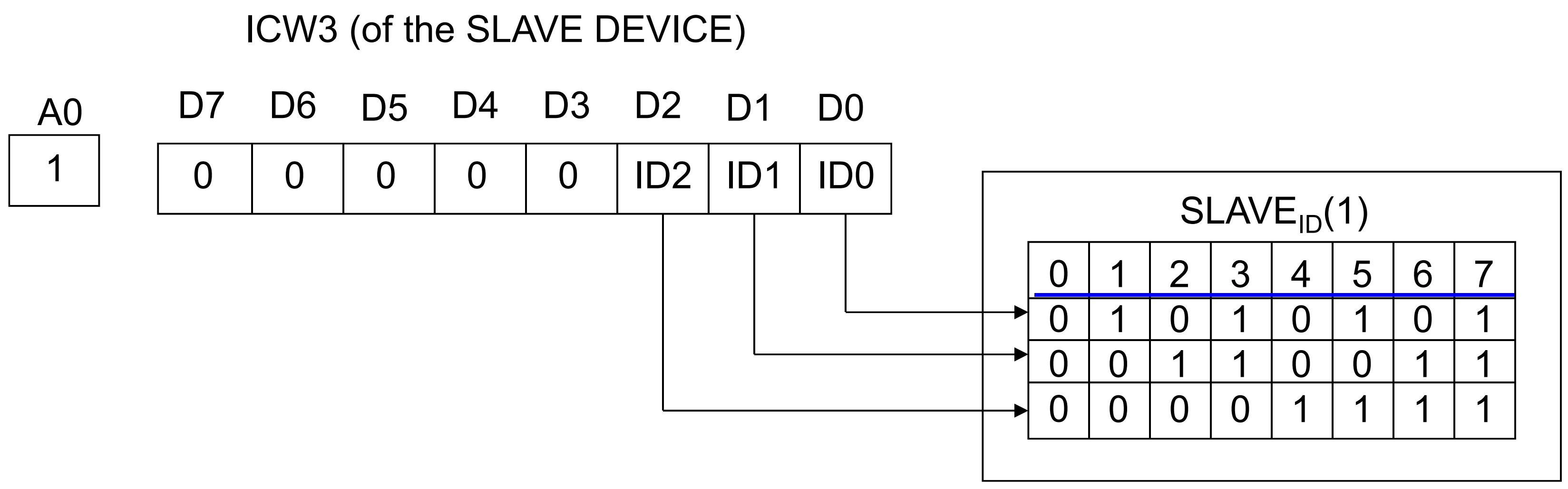
D7	D6	D5	D4	D3	D2	D1	D0
S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

ICW3 (of the MASTER DEVICE)

1=IR INPUT HAS A SLAVE
0=IR INPUT DOES NOT HAVE A SLAVE

Q: Assume a slave is at IR2, what is the ICW3 for the master? (Ans: 04H)

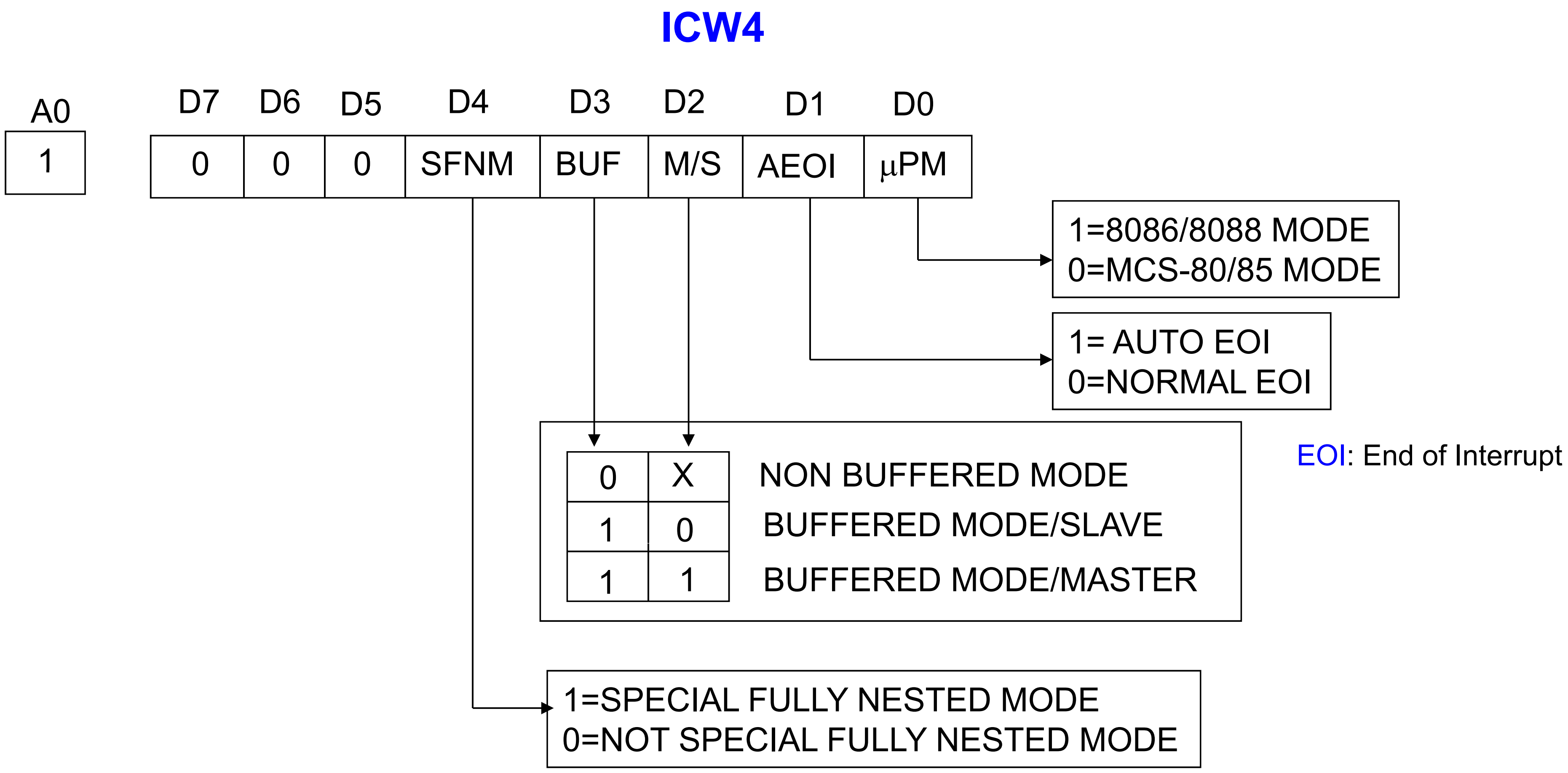
8259A Initialization Command Word Formats ICW3 (Slave)



0	1	2	3	4	5	6	7
0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1
0	0	0	0	1	1	1	1

NOTE 1: SLAVE ID IS EQUAL TO THE
CORRESPONDING MASTER IR INPUT

8259A Initialization Command Word Formats ICW4



EOI: End of Interrupt

Special Fully Nested Mode: allows master PIC to accept request on a (master) IR input that is already in service

- EOI (End of Interrupt): An **End Of Interrupt** (**EOI**) is a signal sent to a Programmable Interrupt Controller (PIC) to indicate the completion of interrupt processing for a given interrupt. An EOI is used to cause a PIC to clear the corresponding bit in the In-Service Register (ISR), and thus allow more interrupt requests of equal or lower priority to be generated by the PIC.

(http://en.wikipedia.org/wiki/End_of_interrupt)

Example of initializing ICW

Example: What values should be written into ICW1 in order to configure the 8259A such that ICW4 is needed in the initialization sequence, the system is going to use multiple 8259A, and its inputs are to be level sensitive? Assume that all unused bits are to be logic 0. Give the results in both binary and hexadecimal form.

Solution:

ICW1 :

D0=

D1=

D3=

D2=D5=D6=D7=

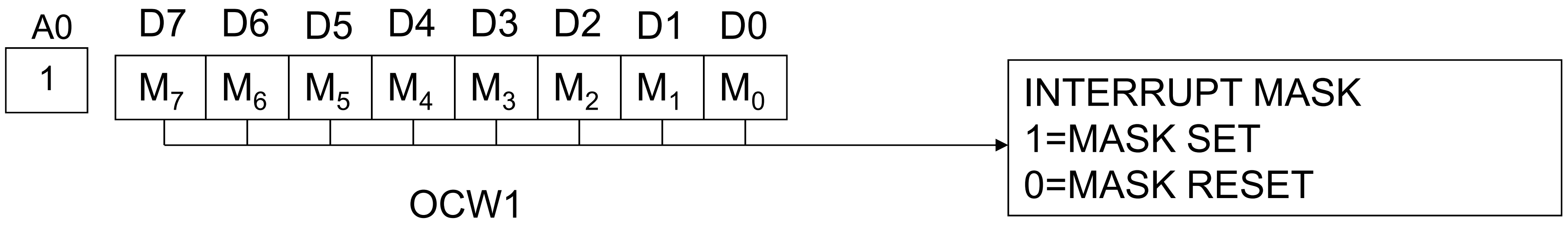
D4=

$\rightarrow$ ICW1=00011001 B = 19 H

8259A Operational Command Words OCW1

OCW1: is used to set and reset the interrupt mask register. This will mask out unwanted interrupt request inputs (IRs input).

(Must be programmed after programming ICW)

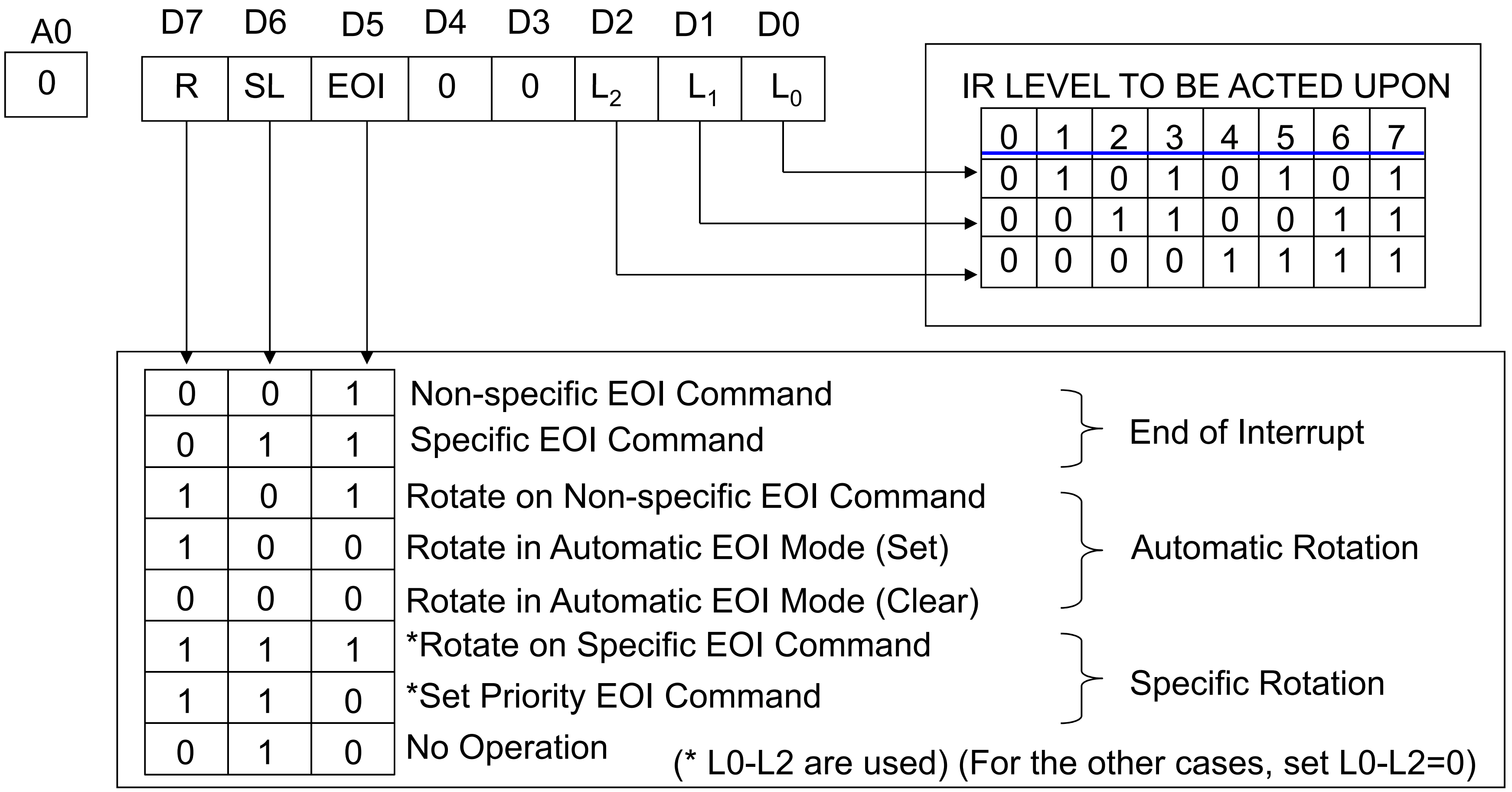


For example, if only IR0 and IR2 will be used

$\rightarrow$ OCW1= 11111010

8259A Operational Command Words OCW2

OCW2: to select how the 8259A responds to an interrupt: provide EOI and rotate priority in normal EOI and auto EOI mode.



A0	D7	D6	D5	D4	D3	D2	D1	D0
0	R	SL	EOI	0	0	L_2	L_1	L_0

IR LEVEL TO BE ACTED UPON

	0	1	2	3	4	5	6	7
L_0	0	1	0	1	0	1	0	1
L_1	0	0	1	1	0	0	1	1
L_2	0	0	0	0	1	1	1	1

R	SL	EOI		
0	0	1	Non-specific EOI Command	End of Interrupt
0	1	1	Specific EOI Command	End of Interrupt
1	0	1	Rotate on Non-specific EOI Command	Automatic Rotation
1	0	0	Rotate in Automatic EOI Mode (Set)	Automatic Rotation
0	0	0	Rotate in Automatic EOI Mode (Clear)	Automatic Rotation
1	1	1	*Rotate on Specific EOI Command	Specific Rotation
1	1	0	*Set Priority EOI Command	Specific Rotation
0	1	0	No Operation	

(* L0-L2 are used) (For the other cases, set L0-L2=0)

Priority Rotation

Objectives?

To control the priority of the interrupt request on IR0-7:
Who will be fed first? And Who will be the next one?

- 8259 can be operated in the following modes:
  1. Fully Nested (default mode)
  2. Special Fully Nested (for the master 8259)
  3. Non Specific Rotating
  4. Specific Rotating
  5. Special Mask
  6. Poll

(See 8259A.pdf for further details.)

E.g. Mode 3: Nonspecific Priority Rotation

Before Rotation

(Assume IR4 has the highest priority)

(ISR)	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
IS status	0	1	0	1	0	0	0	0

Priority status	7	6	5	4	3	2	1	0
	lowest							highest

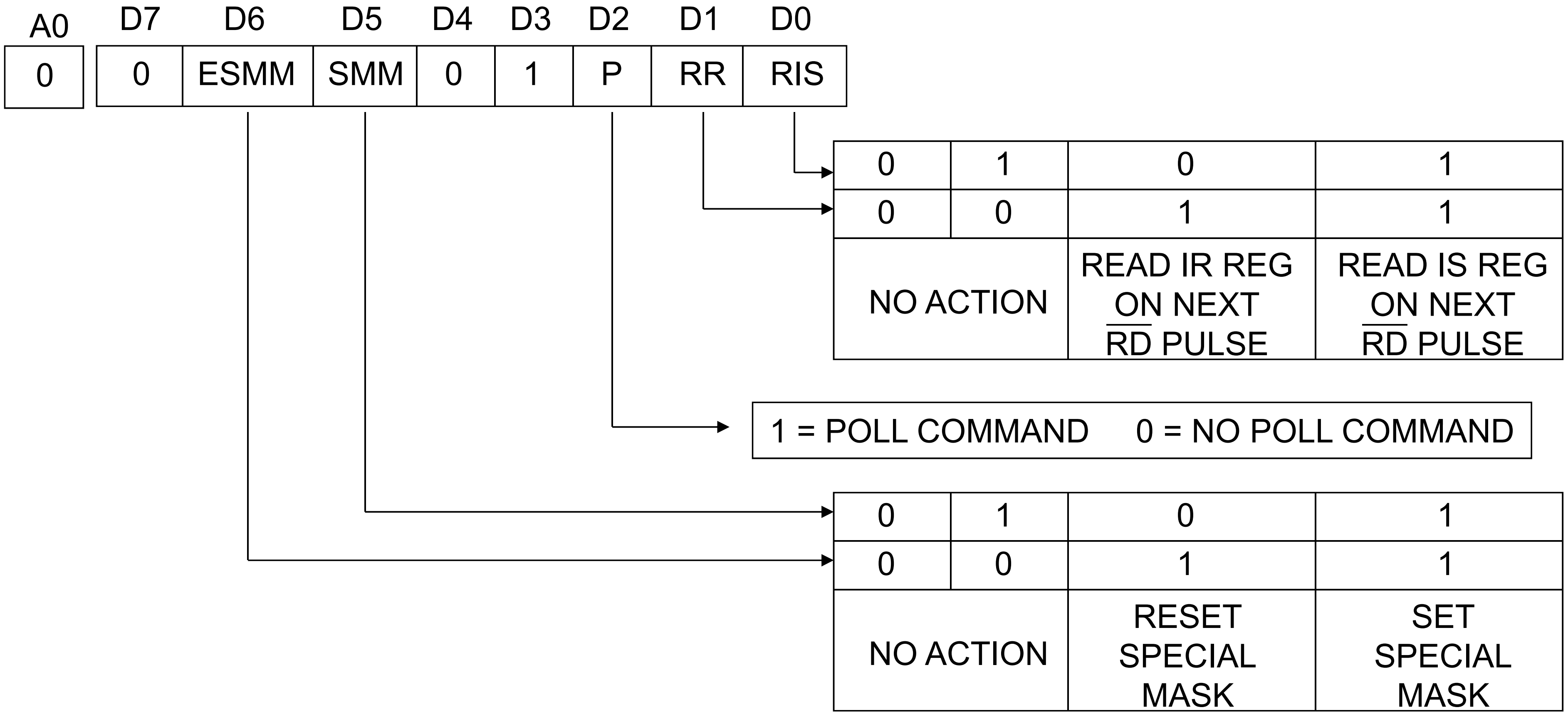
After Rotation

	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
IS status	0	1	0	0	0	0	0	0

Priority status	2	1	0	7	6	5	4	3
			highest	lowest				

8259A Operational Command Words OCW3

OCW3: to select the register to be read, the operation of the special mask register, and the poll command.



A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	ESMM	SMM	0	1	P	RR	RIS

0	1	0	1
0	0	1	1
NO ACTION		READ IR REG ON NEXT $\overline{\text{RD}}$ PULSE	READ IS REG ON NEXT $\overline{\text{RD}}$ PULSE

1 = POLL COMMAND 0 = NO POLL COMMAND

0	1	0	1
0	0	1	1
NO ACTION		RESET SPECIAL MASK	SET SPECIAL MASK

Example of Programming OCW

Example: What should OCW1 be if interrupt inputs IR0 through IR3 are to be disabled and IR4 through IR7 enable?

Solution: OCW1=$\underline{0000}\,\underline{1111}$B = 0FH

Example: What shall OCW2 be if the priority scheme rotate on nonspecific EOI command is to be selected?

Solution: OCW2=$\underline{1010}\,\underline{0000}$B=A0H

System address of ICW and OCW

- The system address for ICW and OCW commands word depends on the hardware wiring. ICW and OCW are sent to 8259 base address or base address+*n*, where *n* depends on which pin of 8088 that the A0 pin of 8259 is connecting to.

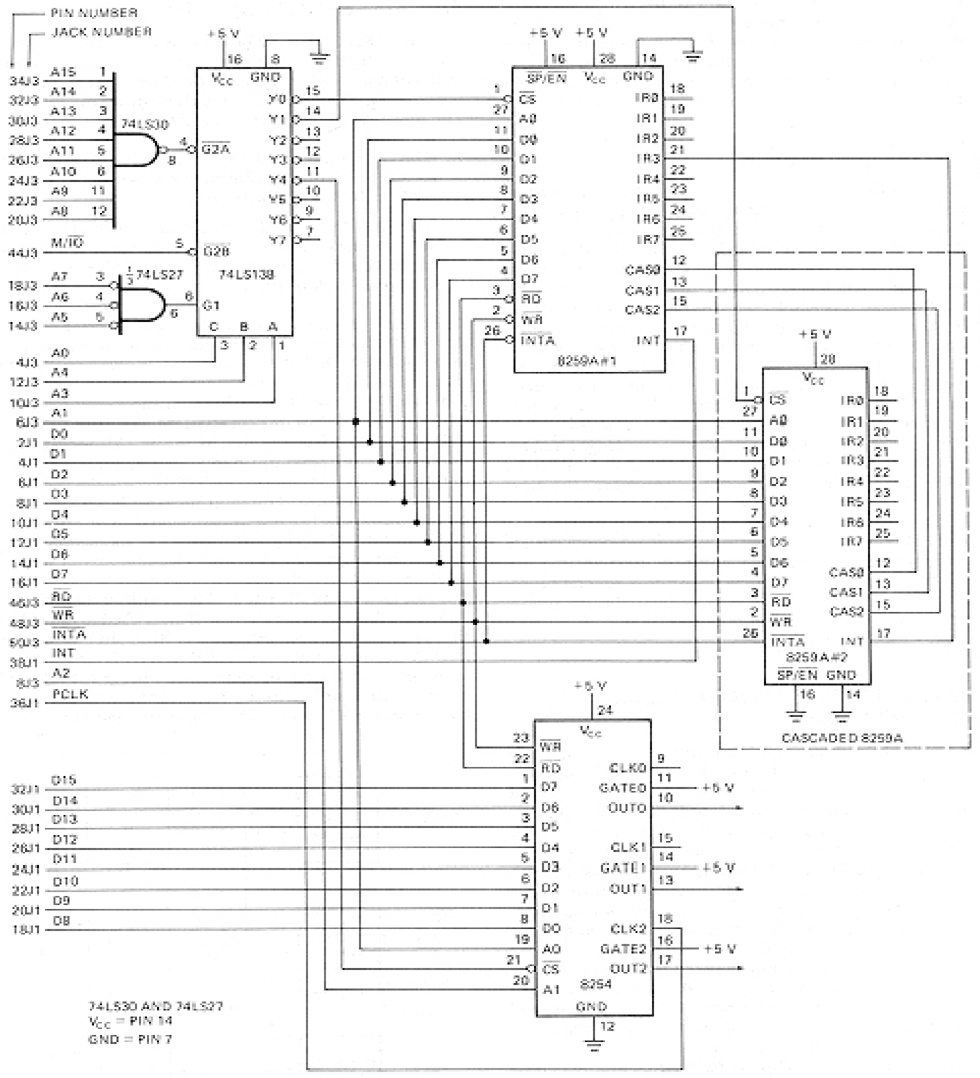
  E.g. If A0 is connected to the A0 of 8088, then *n*=1. If A0 is connected to the A1 of 8088, then *n*=2.

- ICW1, OCW2, and OCW3 (A0=0): base-address
- ICW2,ICW3,ICW4, and OCW1 (A0=1): base-address+*n*

- ICW2-ICW4 need to be sent sequentially to differentiate them
- OCW2 and OCW3 are differentiated by their D4,D3 pins.
- ICWs need to be initialized before OCWs.

(See Brey's example 12-8.)

Adding 8254 and 8259A(s) to an SDK-86 board



(D Hall Fig 8-14)

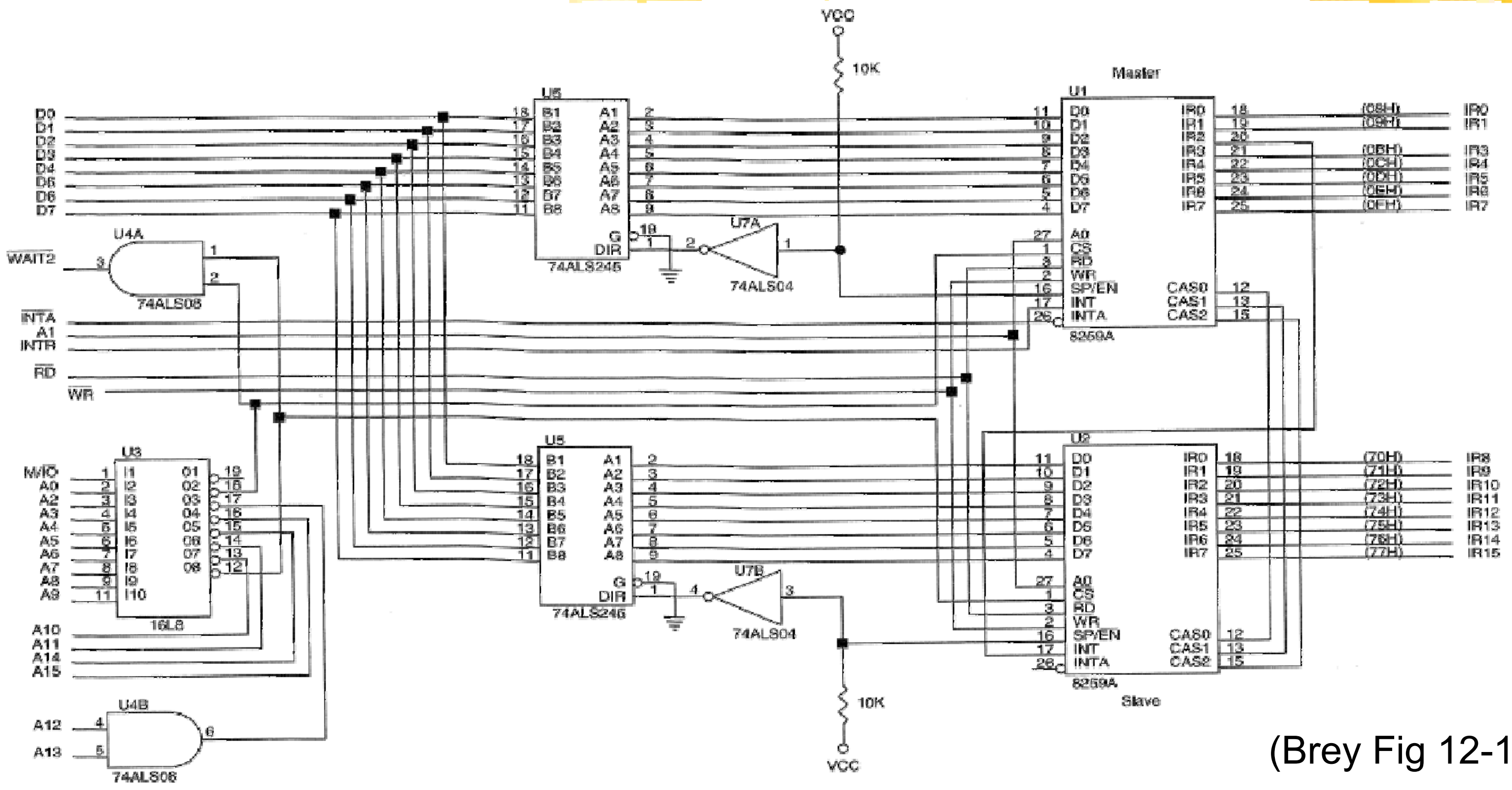
(Fig 8-15)

8259#1 :0FF00h

8259#2 :0FF08h

8254 :0FF01h

Example of Multiple 8259As Interfacing



VCC
10K
Master
U1
U5
74ALS245
U4A
WAIT2
74ALS08
U7A
74ALS04
8259A
D0 D1 D2 D3 D4 D5 D6 D7
INTA A1 INTR RD WR
M/IO A0 A2 A3 A4 A5 A6 A7 A8 A9
A10 A11 A14 A15
U3
16L8
U2
U7B
Slave
U4B
A12 A13
(08H) (09H) (0BH) (0CH) (0DH) (0EH) (0FH)
IR0 IR1 IR3 IR4 IR5 IR6 IR7
(70H) (71H) (72H) (73H) (74H) (75H) (76H) (77H)
IR8 IR9 IR10 IR11 IR12 IR13 IR14 IR15
CAS0 CAS1 CAS2
SP/EN INT INTA CS RD WR

(Brey Fig 12-17)

Two 8259As interfaced to microprocessor at I/O ports 0300H and 0302H for the master and 0304H and 0306H for the slave

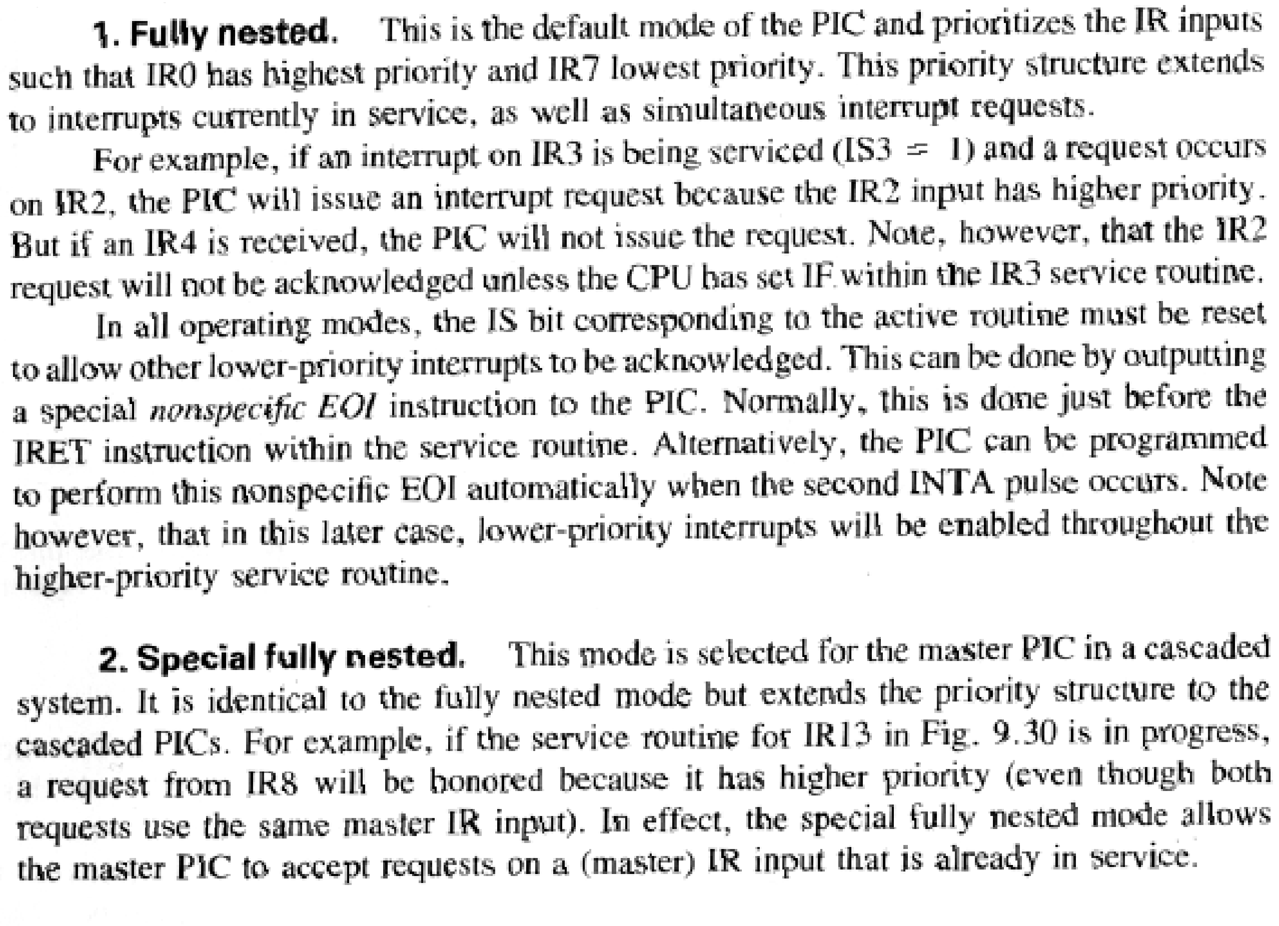
Q&A

- Q: What is the system address that store the interrupt subroutine's IP and CS for the hardware interrupt INTR?
- A: After the PIC receives INTA from CPU, it will put up the interrupt vector type on the data bus. The new IP and CS are located by (vector type x 4), derived the same way as the INT n.

- Q: What is the function for IMR?
- A: It is used to disable (mask) the interrupt request from some of the IR pins. If no hardware is connected a specific IR pin, it is not necessary to disable the pin.

- Q: What is SP/EN pin in PIC ?
- A: See the specification sheet for PIC in buffered mode (pp 18)

8259 Mode of operation for Priority arrangement



1. Fully nested. This is the default mode of the PIC and prioritizes the IR inputs such that IR0 has highest priority and IR7 lowest priority. This priority structure extends to interrupts currently in service, as well as simultaneous interrupt requests.

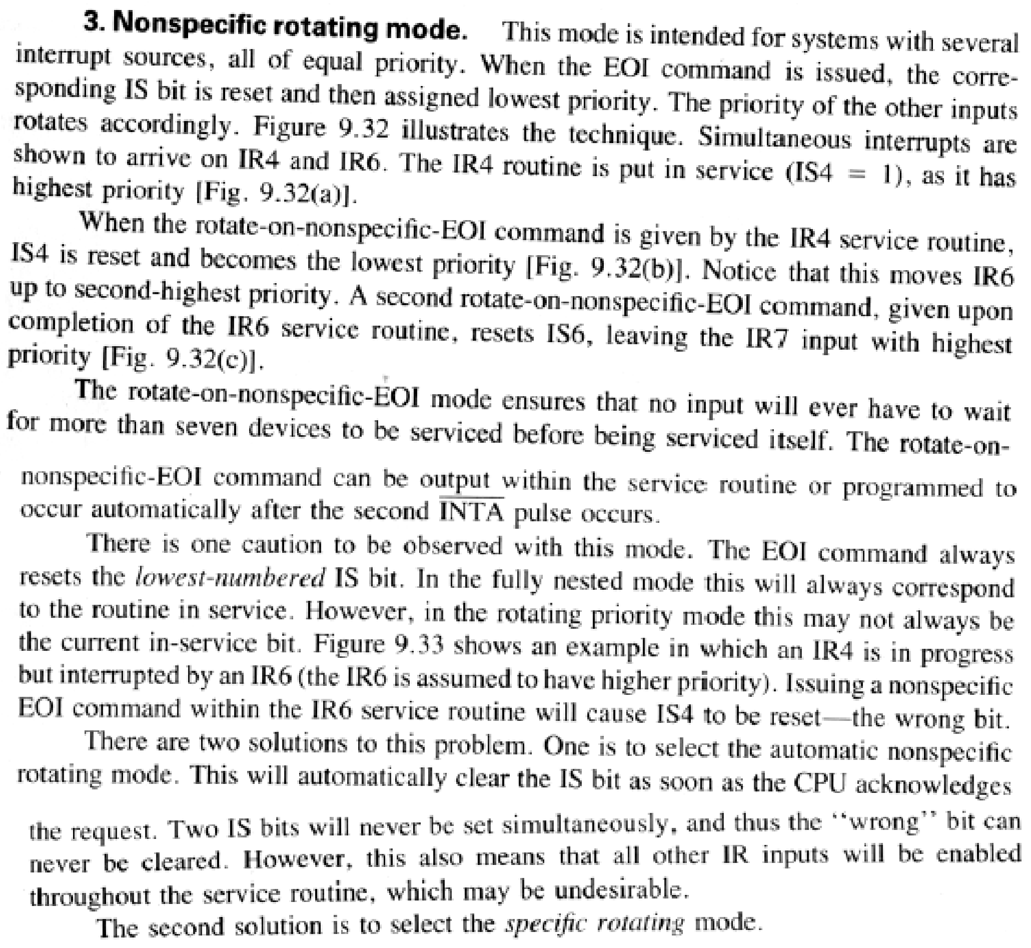
For example, if an interrupt on IR3 is being serviced (IS3 = 1) and a request occurs on IR2, the PIC will issue an interrupt request because the IR2 input has higher priority. But if an IR4 is received, the PIC will not issue the request. Note, however, that the IR2 request will not be acknowledged unless the CPU has set IF within the IR3 service routine.

In all operating modes, the IS bit corresponding to the active routine must be reset to allow other lower-priority interrupts to be acknowledged. This can be done by outputting a special *nonspecific EOI* instruction to the PIC. Normally, this is done just before the IRET instruction within the service routine. Alternatively, the PIC can be programmed to perform this nonspecific EOI automatically when the second INTA pulse occurs. Note however, that in this later case, lower-priority interrupts will be enabled throughout the higher-priority service routine.

2. Special fully nested. This mode is selected for the master PIC in a cascaded system. It is identical to the fully nested mode but extends the priority structure to the cascaded PICs. For example, if the service routine for IR13 in Fig. 9.30 is in progress, a request from IR8 will be honored because it has higher priority (even though both requests use the same master IR input). In effect, the special fully nested mode allows the master PIC to accept requests on a (master) IR input that is already in service.

#: for reference only

8259 Mode of operation for Priority arrangement



3. Nonspecific rotating mode. This mode is intended for systems with several interrupt sources, all of equal priority. When the EOI command is issued, the corresponding IS bit is reset and then assigned lowest priority. The priority of the other inputs rotates accordingly. Figure 9.32 illustrates the technique. Simultaneous interrupts are shown to arrive on IR4 and IR6. The IR4 routine is put in service (IS4 = 1), as it has highest priority [Fig. 9.32(a)].

When the rotate-on-nonspecific-EOI command is given by the IR4 service routine, IS4 is reset and becomes the lowest priority [Fig. 9.32(b)]. Notice that this moves IR6 up to second-highest priority. A second rotate-on-nonspecific-EOI command, given upon completion of the IR6 service routine, resets IS6, leaving the IR7 input with highest priority [Fig. 9.32(c)].

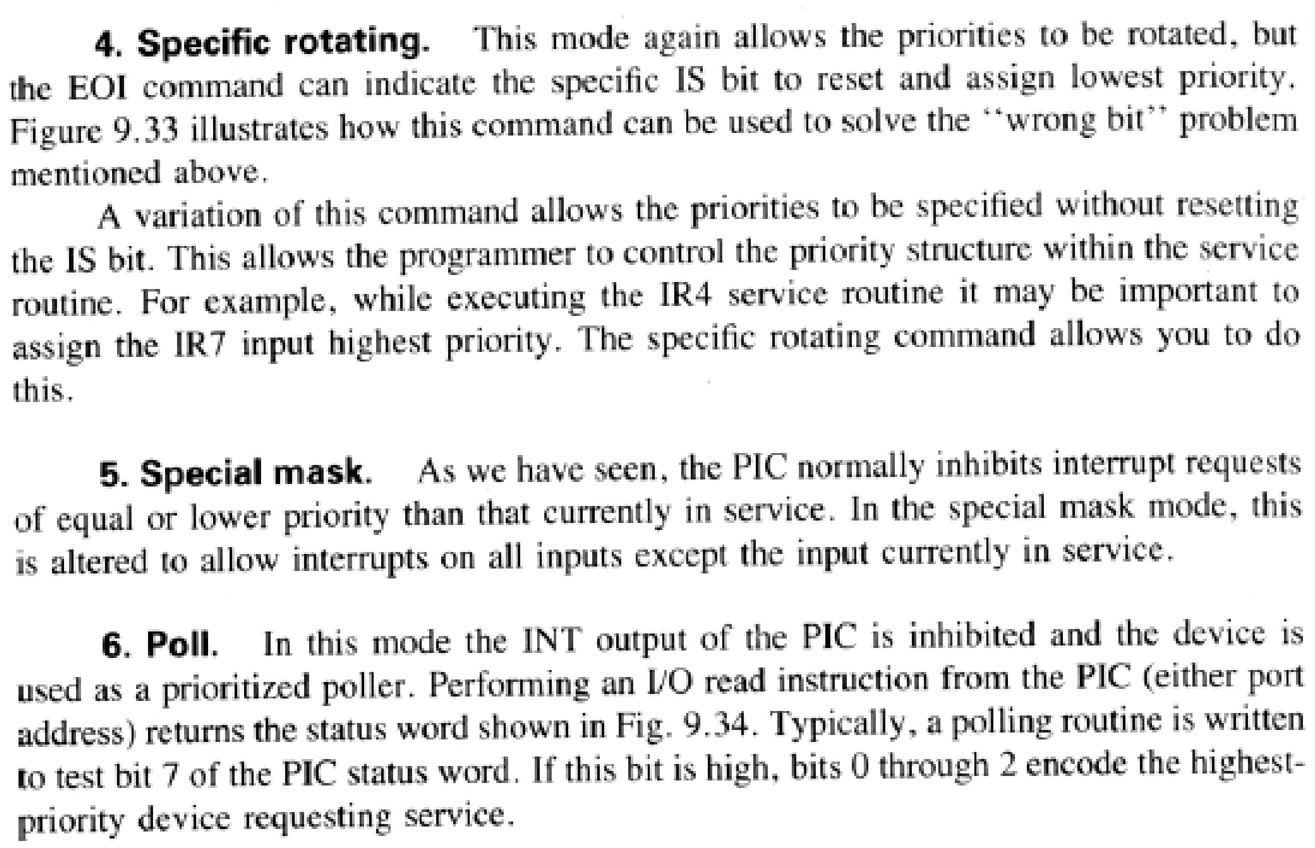
The rotate-on-nonspecific-EOI mode ensures that no input will ever have to wait for more than seven devices to be serviced before being serviced itself. The rotate-on-nonspecific-EOI command can be output within the service routine or programmed to occur automatically after the second $\overline{\text{INTA}}$ pulse occurs.

There is one caution to be observed with this mode. The EOI command always resets the *lowest-numbered* IS bit. In the fully nested mode this will always correspond to the routine in service. However, in the rotating priority mode this may not always be the current in-service bit. Figure 9.33 shows an example in which an IR4 is in progress but interrupted by an IR6 (the IR6 is assumed to have higher priority). Issuing a nonspecific EOI command within the IR6 service routine will cause IS4 to be reset—the wrong bit.

There are two solutions to this problem. One is to select the automatic nonspecific rotating mode. This will automatically clear the IS bit as soon as the CPU acknowledges the request. Two IS bits will never be set simultaneously, and thus the "wrong" bit can never be cleared. However, this also means that all other IR inputs will be enabled throughout the service routine, which may be undesirable.

The second solution is to select the *specific rotating* mode.

8259 Mode of operation for Priority arrangement



4. Specific rotating. This mode again allows the priorities to be rotated, but the EOI command can indicate the specific IS bit to reset and assign lowest priority. Figure 9.33 illustrates how this command can be used to solve the "wrong bit" problem mentioned above.

A variation of this command allows the priorities to be specified without resetting the IS bit. This allows the programmer to control the priority structure within the service routine. For example, while executing the IR4 service routine it may be important to assign the IR7 input highest priority. The specific rotating command allows you to do this.

5. Special mask. As we have seen, the PIC normally inhibits interrupt requests of equal or lower priority than that currently in service. In the special mask mode, this is altered to allow interrupts on all inputs except the input currently in service.

6. Poll. In this mode the INT output of the PIC is inhibited and the device is used as a prioritized poller. Performing an I/O read instruction from the PIC (either port address) returns the status word shown in Fig. 9.34. Typically, a polling routine is written to test bit 7 of the PIC status word. If this bit is high, bits 0 through 2 encode the highest-priority device requesting service.