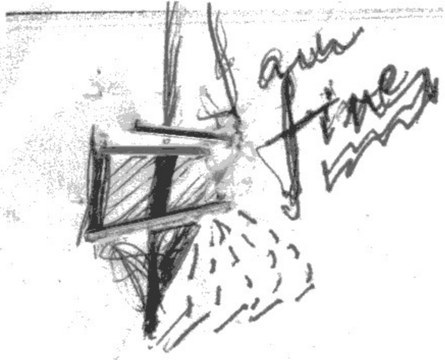

CHAPTER 13

Priority Interrupt Controller: 8259

1. The Priority Interrupt Controller (PIC) functions as an overall manager in an interrupt-driven system environment. It accepts interrupt requests from peripheral devices, resolves their priorities, and issues an interrupt to the 8085A on the INTR input of the 8085A. In response to the $\overline{\text{INTA}}$ generated by 8085A, the 8259 puts out a CALL instruction operation code followed by the pre-programmed adddress of the service routine associated with the interrupt, for the CPU to execute.

2. The 8259 can handle up to eight levels of interrupts; several 8259s may be cascaded to handle up to sixty-four levels of interrupts.

3. The 8259 is programmed by a set of Initialisation Command Words (ICWs). The ICWs are used to specify the *vectoring addresses* for the associated interrupts, to specify single or cascaded mode of operation, level- or edge-triggered mode, etc. Operation Command Words (OCWs) are used to operate the 8259 in various interrupt modes, which may be: (i) Fully Nested Mode, (ii) Rotating Priority Mode, (iii) Special Mask Mode, or (iv) Polled Mode. They may also be used for masking specific interrupts, status read operations, etc.

In an interrupt-driven system environment, an I/O device requiring the attention of the CPU activates one of the Interrupt inputs of the CPU. The CPU completes the instruction being executed and branches to the interrupt service routine of the interrupt. The CPU services the I/O device in accordance to the instructions in the interrupt service routine, and upon encountering a RETURN statement (which must be placed by the programmer), it returns to the main program.

The 8085A CPU has four interrupt inputs which can be used in the above manner by four separate I/O devices. The TRAP, RST 7.5, RST 6.5, and RST 5.5 interrupts cause the 8085A CPU to branch to *specific* addresses as given in Table 3.1 of Chapter 3. These together are called *vectored interrupts*. In addition to these inputs, the 8085A CPU can also be interrupted through its INTR input. An external device can use the INTR input to make the 8085A CPU *execute* an *instruction* that is placed on the data bus by the *device* (in response to the $\overline{\text{INTA}}$ signal that is

generated by the CPU). The timing diagrams for a CALL instruction placed by the external device are shown in Fig 3.10a and Fig 3.10b of Chapter 3. Instead of causing the 8085A CPU to branch to specific addresses while using vectored interrupts (thus limiting the number of I/O devices that may be connected to the CPU in an interrupt-driven environment to four), the INTR input may be used to make the CPU to branch to *any* address as specified by the external device in a CALL instruction. The number of I/O devices that may be used in an interrupt-driven environment may be made greater than four if an external device, which is capable of accepting interrupt requests and generating unique CALL instructions for the different interrupt inputs, is used to interrupt the 8085A on the INTR line. A Priority Interrupt Controller (PIC, as shown in Fig. 13.1), is such a device which accepts interrupt requests from a number of I/O devices, resolves the priority of servicing the requests, and issues an interrupt on the INTR input of the 8085A. In response to the $\overline{\text{INTA}}$ generated by the 8085A, it puts out a CALL instruction code on the data bus which is read by the 8085A. The 8085A decodes the instruction and puts out two more $\overline{\text{INTA}}$ s to read an address. The interrupt manager (PIC) puts out the address of the interrupt service routine of the interrupt currently *being serviced* by it in response to the additional $\overline{\text{INTA}}$ s. In order to perform these functions, the PIC must be initialised first by loading the addresses of the interrupt service routines that must be put out on the occurrence of the interrupts. In addition, the PIC must be capable of being operated in a variety of interrupt modes, as selected by the programmer.

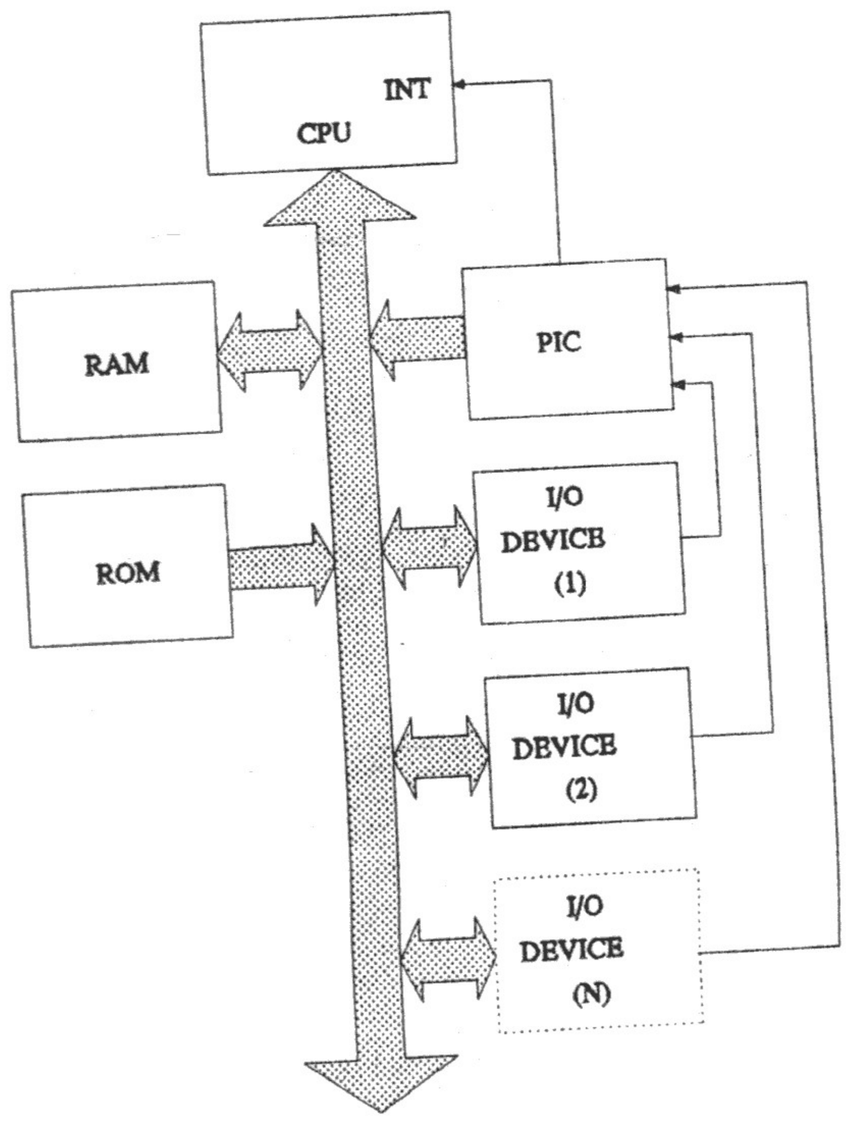

Figure 13.1 *PIC in an Interrupt-Driven Environment*
(Source: Intel Corporation)

The 8259 is a programmable Priority Interrupt Controller which can be used in interru microcomputer systems. It accepts interrupt requests from upto eight I/O devices, determ interrupt which at a given time has the highest priority, and issues an interrupt to the CPU. an acknowledgement from the CPU, the 8259 places a CALL instruction along with the associa address (pertaining to the interrupt service routine) of the current interrupt on the data bus. addition to these general features, the 8259 is capable of being programmed for a variety of modes of operation, which include priority setting for interrupts, cascade operation with other PICs, etc. The 8259 can be operated with 8080/8085 or 8088/8086 microprocessors. However, in this Chapter, 8259 operation in an 8085A environment is described.

13.1 FUNCTIONAL BLOCK DESCRIPTION

The 8259 is initialised by the CPU as an I/O device, by writing a set of Initialisation Command Words (ICWs). Operation Command Words (OCWs) are used to configure the 8259 in a variety of modes, which include masking specific interrupt inputs, priority setting of interrupts, etc. Fig. 13.2 and Fig. 13.3 show the pin diagram and functional block diagram of the 8259. The 8259 is constituted by four sections: (i) Interrupts and Control Logic Section, (ii) Data Bus Buffer, (iii) Read/Write Control Logic Section, and (iv) Cascade Buffer/Comparator Section. The four Sections are described below with reference to Fig. 13.3.

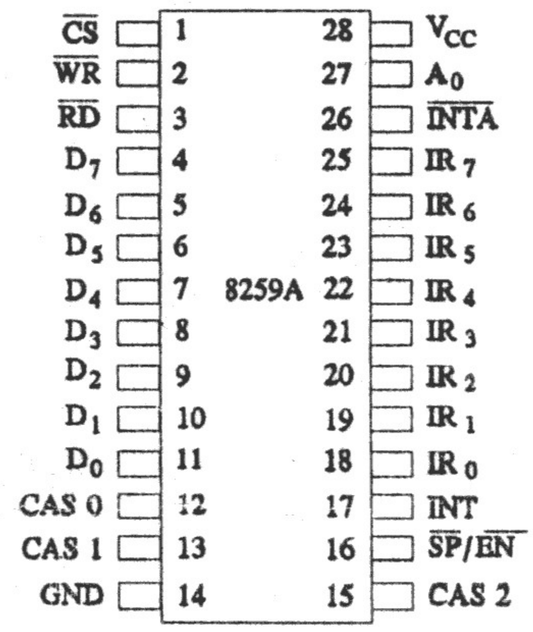

Figure 13.2 8259 Pin Diagram
(Source: Intel Corporation)

13.1.1 Interrupts and Control Logic Section

This section consists of: (a) Interrupt Request Register (IRR), (b) In-Service Register (ISR), (c) Priority Resolver, (d) Interrupt Mask Register (IMR), and (e) Control Logic Block.

(a) Interrupt Request Register (IRR)

The eight interrupt inputs set corresponding bits of the Interrupt Request Register. The IRR is used to store information about the interrupt inputs *requesting* service.

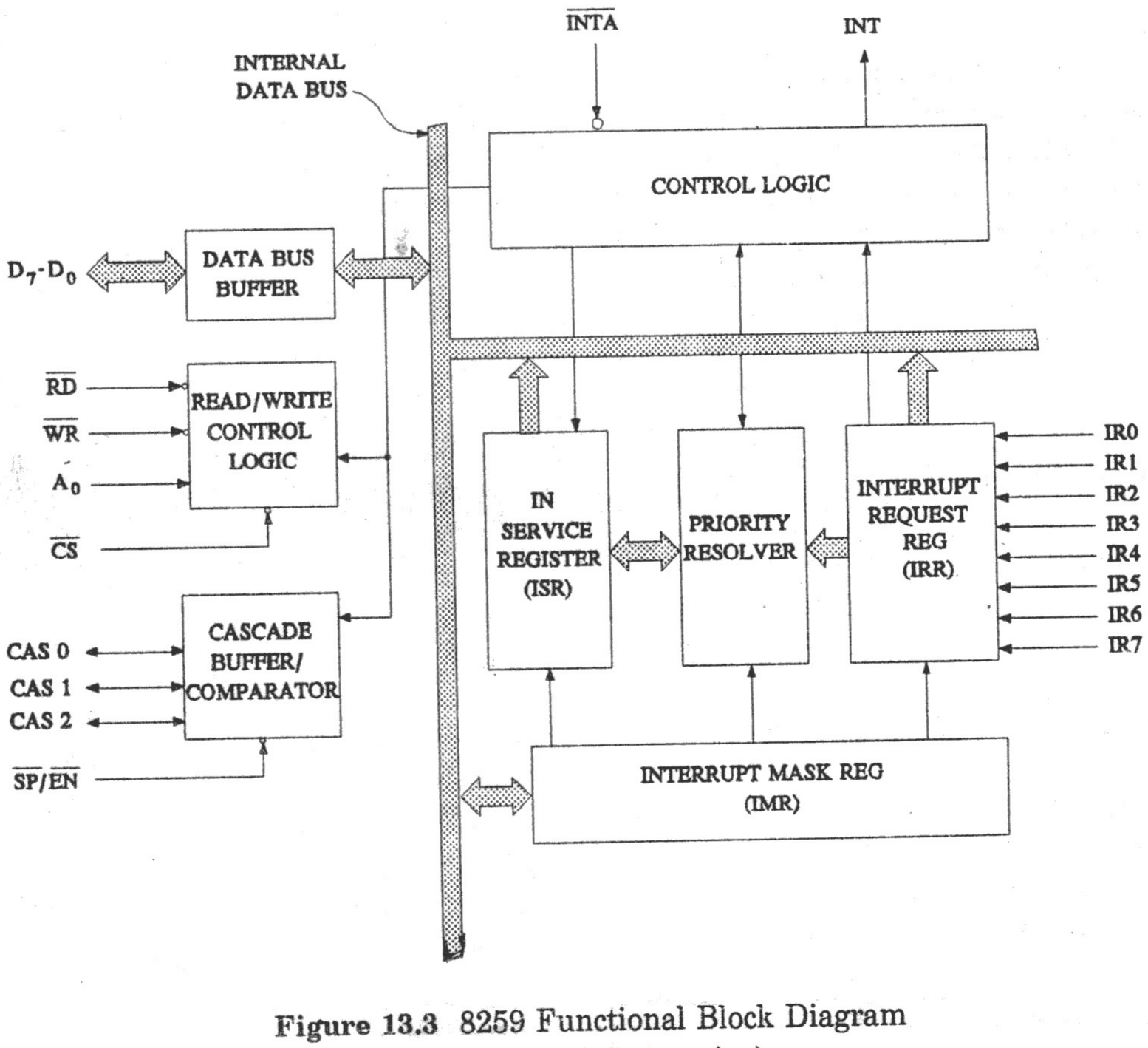

Figure 13.3 8259 Functional Block Diagram
(Source: Intel Corporation)

(b) In-Service Register (ISR)

The In-Service Register is used to store information about the interrupts currently *being* serviced.

(c) Priority Resolver

This determines the priorities of the interrupts requesting service (which set corresponding bits of the IRR). The resolver determines the priorities as dictated by the priority mode set by the OCWs. The bit corresponding to the highest priority interrupt input is set in the ISR during the $\overline{\text{INTA}}$ input.

(d) Interrupt Mask Register (IMR)

This register can be programmed by an OCW to store the bits which mask specific interrupts. The IMR operates on the IRR. An interrupt which is masked by software (by programing the IMR) will not be recognised and serviced even if it sets the corresponding bit in the IRR.

(e) Control Logic

This block has an input and an output line. The 8259, after resolving its input interrupt request priorities, puts out an interrupt request to the CPU, on the INT output. This is directly connected to the CPU interrupt input. In the 8085A, the INT output is connected to the INTR. The CPU responds to the request by putting out an $\overline{\text{INTA}}$. This signal is given to the 8259 on the $\overline{\text{INTA}}$ input. The 8259 then places the operation code for the CALL instruction on the data bus. This is read by the CPU which perceives that two additional $\overline{\text{INTA}}$s are required to read the address (*vectoring data*) of the service routine. The 8259 places the two address bytes on the data bus when the two additional $\overline{\text{INTA}}$ signals are received.

13.1.2 Data Bus Buffer

This 8-bit bidirectional tri-state buffer is used to interface the 8259 to the system data bus. Control words for the 8259, status words, and vectoring data are all passed through the data bus buffer.

13.1.3 Read/Write Control Logic Section

This contains the Initialisation Command Word Registers (ICW registers) and the Operation Command Word Registers (OCW registers) which are programmed by the CPU to set up the 8259, and to operate it in various modes. This section also accepts Read commands from the CPU to permit the CPU to read status words. The pins associated with this section are described below.

Chip Select CS

This is an *active low* input which is used to select the device.

WRITE ($\overline{\text{WR}}$)

This is an *active low* input and is used to write OCWs and ICWs onto the 8259.

Read ($\overline{\text{RD}}$)

This is also an *active low* input. It is used by the CPU to read the status of the IRR, ISR, IMR, or the interrupt level.

A_0

13.1.4 Cascade Buffer Comparator

This section generates control signals necessary for cascade operations. It also generates Buffer-Enable signals. As stated earlier, the 8259 can be cascaded with other 8259s in order to expand the interrupt handling capacity to sixty-four levels. In such a case, the former is called a *master*, and the latter are called *slaves*. The 8259 can be set up as a master or a slave by the $\overline{SP}/\overline{EN}$ pin in the non-buffered mode, or by software if it is to be operated in the buffered mode of operation (buffered and non-buffered modes of operation are described later in this Section).

CAS 0-2

For a *master* 8259, the CAS0-CAS2 pins are outputs, and for *slave* 8259s, these are inputs. When the 8259 is a *master* (that is, when it *accepts* interrupt requests from *other* 8259s), the CALL opcode is generated by the Master in response to the *first* $\overline{INTA}$. The vectoring address must be released by the *slave* 8259. The master puts out an identification code of three-bits (to select one out of the eight possible slave 8259s) on the CAS 0-CAS 2 lines. The slave 8259s accept these three signals as *inputs* (on *their* CAS 0 - CAS 2 pins) and compare the code put out by the master with the codes assigned to them during initialisation. The slave thus selected (which had originally placed an interrupt request to the master 8259) then puts out the address of the interrupt service routine during the second and third $\overline{INTA}$ pulses from the CPU.

$\overline{SP}/\overline{EN}$ (Slave Program/Enable Buffer)

This pin is used to specify whether the 8259 is to act as a master or a slave. If this pin is kept at 5V the 8259 understands that it is to function as a master, and if it is kept at 0V, the 8259 understands that it is to function as a slave.

In large systems where buffers are used to drive the data bus, the data put out by the 8259 in response to $\overline{INTA}$ cannot be accessed by the CPU (due to the data bus buffer being disabled). If an 8259 is used in the buffered mode (buffered or non-buffered modes of operation can be specified at the time of initialising the 8259), the $\overline{SP}/\overline{EN}$ pin is used as an output which can be used to enable the system data bus buffer whenever the 8259's data bus outputs are enabled (when it is ready to put out data).

To summarise, in non-buffered mode, the $\overline{SP}/\overline{EN}$ pin of an 8259 is used to specify whether the 8259 is to operate as a master or as a slave, and in the buffered mode, the $\overline{SP}/\overline{EN}$ pin is used as an output to enable the data bus buffer of the system. Fig. 13.4 shows the 8259 interfaced to the system in I/O Mapped I/O Mode.

13.2 SUMMARY OF OPERATION

The sequence of operation of the 8259 after its initialisation is as follows:

(i) The peripherals raise one or more interrupt requests. The corresponding bits in the IRR are set.

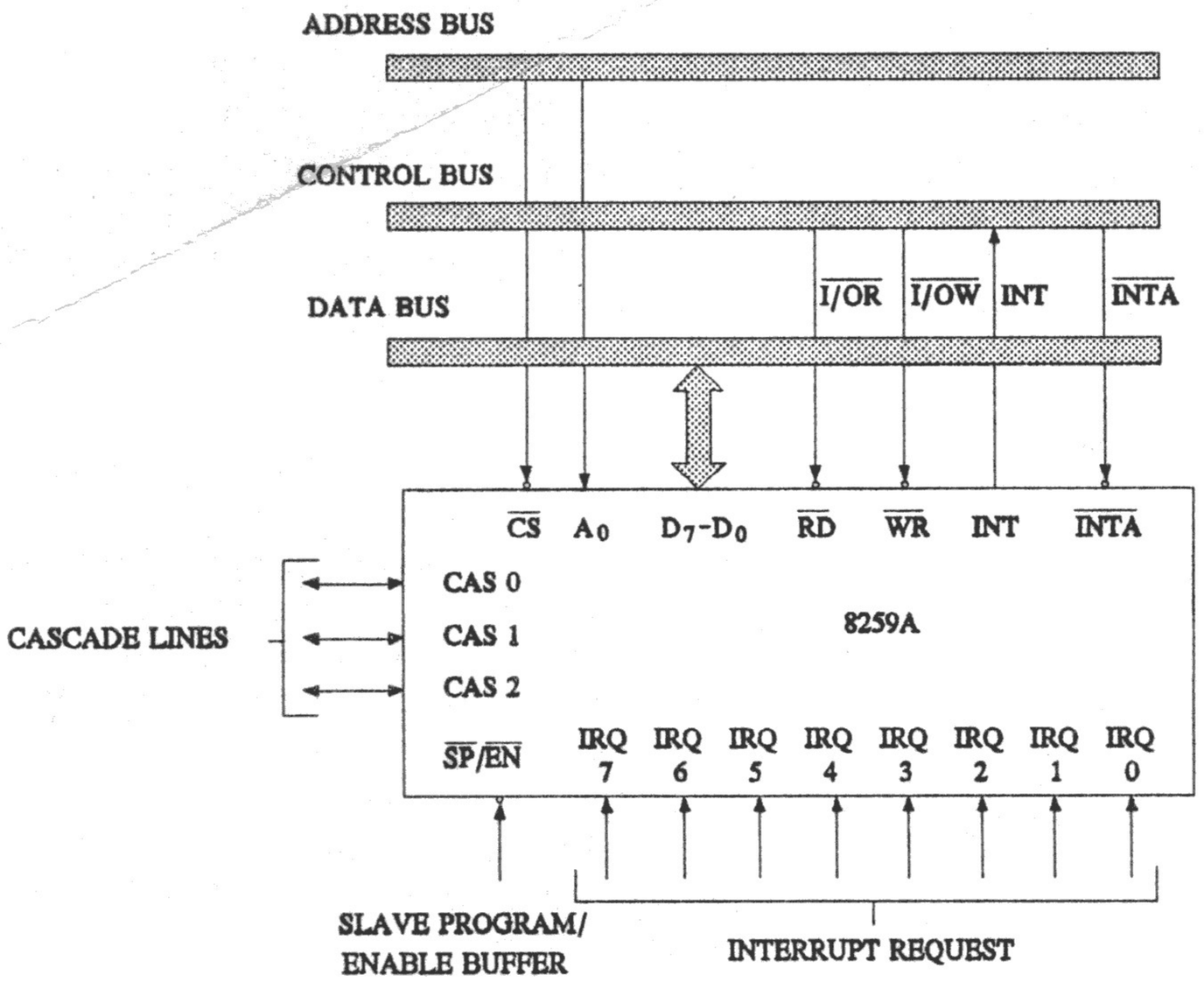

Figure 13.4 8259 Interfacing Connections
(Source: Intel Corporation)

(ii) The 8259 resolves their priorities and sends an interrupt to the CPU.

(iii) The CPU completes the execution of the instruction in process and responds by issuing an $\overline{\text{INTA}}$.

(iv) The highest priority ISR bit is set and the corresponding bit is reset in the IRR. The 8259 releases the opcode for CALL through its D_0-D_7 pins. The CALL instruction opcode is read by the CPU and two more $\overline{\text{INTA}}$ s are generated by the CPU.

(v) The two $\overline{\text{INTA}}$ s cause the 8259 to release the interrupt service routine address of the interrupt being serviced (the address is initially programmed into the 8259 by the CPU). The low-order 8-bits are released upon the receipt of the first $\overline{\text{INTA}}$ and the high-order 8-bits of the address are released upon the receipt of the second $\overline{\text{INTA}}$.

(vi) If the 8259 is programmed in AEOI mode (Automatic End of Interrupt, as explained in section 13.4), at the end of the third $\overline{\text{INTA}}$ pulse, the ISR bit is automatically reset. If the 8259 is not programmed in the AEOI mode, the ISR bit remains set and can be reset only by issuing an End of Interrupt Command to the 8259 through OCW 2.

(vii) If the interrupt is of a short duration and is not present in step (iv), the 8259 issues an interrupt level 7 opcode. (This operation is described in detail in Section 13.6.)

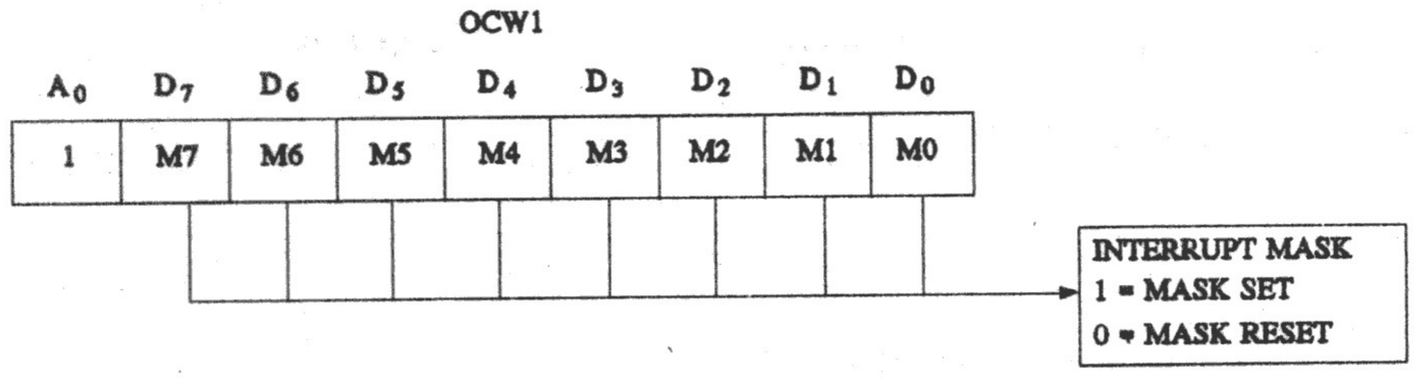

Figure 13.13 Format of Operation Command Words 1 and 2
(Source: Intel Corporation)

13.4 8259 INTERRUPT MODES

The various modes of operation of the 8259 are: (i) Fully Nested Mode, (ii) Rotating Priority Mode, (iii) Special Masked Mode, and (iv) Polled Mode.

13.4.1 Fully Nested Mode (FNM)

This is the default mode setting after initialisation. The 8259 continues to operate in the Fully Nested Mode until the mode is changed through Operation Command Words.

In this mode, the *highest* priority (Priority 0) is assigned to IR0 and the *lowest* priority (Priority 7) is assigned to IR7. When an interrupt request is acknowledged, the 8259 determines the interrupt of the highest priority and sets its corresponding bit in the ISR. The vector address corresponding to this interrupt is then put out. The ISR bit remains set until an End Of Interrupt

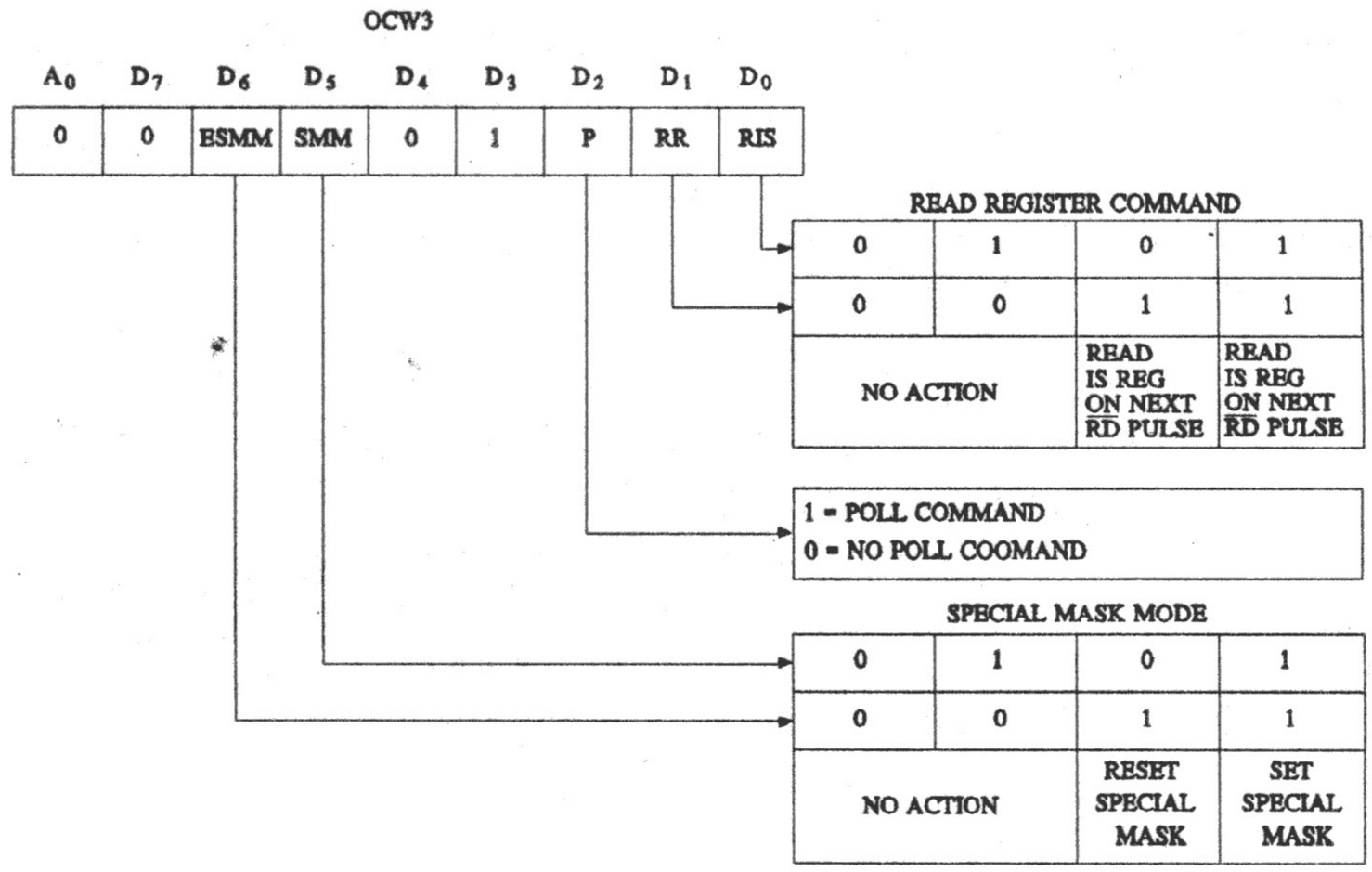

Figure 13.14 Format of Operation Command Word 3
(Source: Intel Corporation)

(EOI) command through an OCW is issued by the CPU before exiting from the interrupt service routine. However, if the AEOI mode is set in ICW4 during initialisation, the ISR bit is automatically reset on the trailing edge of the last $\overline{INTA}$ pulse.

While an ISR bit is set, all lower interrupt levels and another interrupt of the same level are inhibited. However, an interrupt level *higher* than the ISR bit currently being serviced will cause the 8259 to generate an INTR. This interrupt can however be acknowledged only if the Interrupt Enable flip-flop of the microprocessor has been enabled through software (in the interrupt service routine of the current interrupt).

The EOI and AEOI commands are described below.

End Of Interrupt (EOI)

The IS bit can be reset by an End Of Interrupt command issued by the CPU, usually just before exiting from the interrupt routine.

In the Fully Nested Mode, the highest level in the ISR would necessarily correspond to the *last* interrupt acknowledged and serviced. In such a case, a *non-specific* EOI command may be issued by the CPU (by an OUT instruction to the 8259, with $A_0=0$, $D_7D_6D_5D_4D_3$ being 00100, $D_2D_1D_0$ being of *no* significance, as can be deriyed from OCW2 in Fig. 13.13) before exiting from the routine. The 8259 then resets the highest level IS bit (among those that are set).

However, if the FNM is *not* used, the 8259 may not be able to determine the last interrupt acknowledged. In such a case, a *specific* EOI command will have to be issued by the CPU (by an

OUT instruction to the 8259 with $A_0 = 0$, $D_7D_6D_5D_4D_3$ being 01100, $D_2D_1D_0$ specifying the le which the EOI command is to act, as shown in OCW2 in Fig. 13.13) before exiting from interrupt service routine. The 8259 then resets the IS bit of the level *specified* by the E command.

It should be noted that in the cascade mode, the EOI command must be issued *twice*, once for the master and once for the slave.

Automatic End Of Interrupt (AEOI)

The AEOI mode is set by ICW4. If the AEOI mode is set, the 8259 will perform a *non-specific* EOI on its own on the trailing edge of the third $\overline{INTA}$ pulse. The AEOI mode can only be used for a master 8259 and *not* for a slave. If the interrupt system is enabled (by software) in the service routine, it is open to fresh interrupts from *any* level (as dictated by the ISR) since the bit corresponding to the interrupt being serviced is reset in the ISR.

Special Fully Nested Mode (SFNM)

In the FNM, on the acknowledgement of an interrupt, further interrupts from the *same* level are disabled. However, in large systems which use cascaded 8259s and where the interrupt levels *within each slave* have to be considered, this creates a problem. An interrupt input to a slave, in turn causes the slave to place an interrupt request to the master on one of the master's inputs. Further interrupts to the slave will cause the slave to place requests to the master on the same input to the master, but these will not be recognised because further interrupts on the same input level are disabled by the master.

The Special Fully Nested Mode (SFNM) is used to surmount this problem. The SFNM is set up by ICW4 during initialisation. It is similar to the FNM except for the following differences:

(a) When an interrupt request from a slave is being serviced, the slave is allowed to place further requests (these requests are of a higher priority than the request currently being serviced). These interrupts are recognised by the master and it initiates interrupt requests to the CPU.

(b) Before exiting from the interrupt service routine, a non-specific EOI must be sent to the slave and its ISR must be read to determine if it was the *only* interrupt to the slave. If the ISR is empty, a non-specific EOI command can be sent to the master. If it is not empty, it implies that the same IR level input to the master is to be serviced again due to more than one interrupt being presented to the slave, and an EOI must *not* be sent to the master.

13.4.2 Rotating Priority Mode

The Rotating Priority mode can be set in (a) Automatic Rotation, and (b) Specific Rotation.

(a) Automatic Rotation

This mode is used in an interrupt structure where the interrupts must be assigned *equal* priority In this mode the last serviced interrupt level IRX (X can vary from 0 to 7) is automatically

assigned the *lowest* priority and the interrupt level IR(X+1) is assigned the *highest* priority. For example, if the ISR status and the priority status are as shown in Fig. 13.15a (IR3 and IR5 are being serviced, but IR3 service routine is being executed, as it is of a higher priority), *after* an EOI from the IR3 routine, the priorities will be as shown in Fig. 13.15b. Automatic rotation is possible by a non-specific EOI or by an AEOI. For automatic rotation on non-specific EOI, an OCW2 with R=1, SL=0, and EOI=1 must be written into the 8259. Automatic rotation mode on an AEOI is *set* by loading an OCW2 with R=1, SL=0, and EOI=0, and is *reset* by loading an OCW2 with R=0, SL=0, and EOI=0 (see Fig. 13.13).

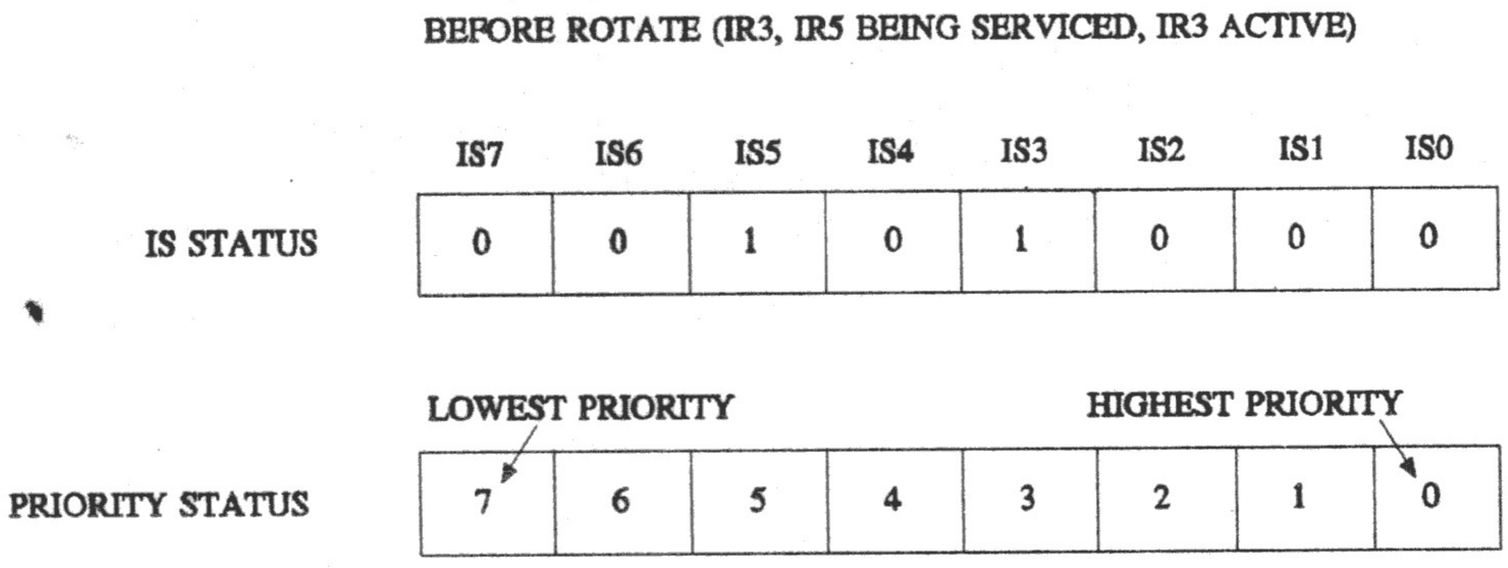

Figure 13.15a Priority Status Before Rotation

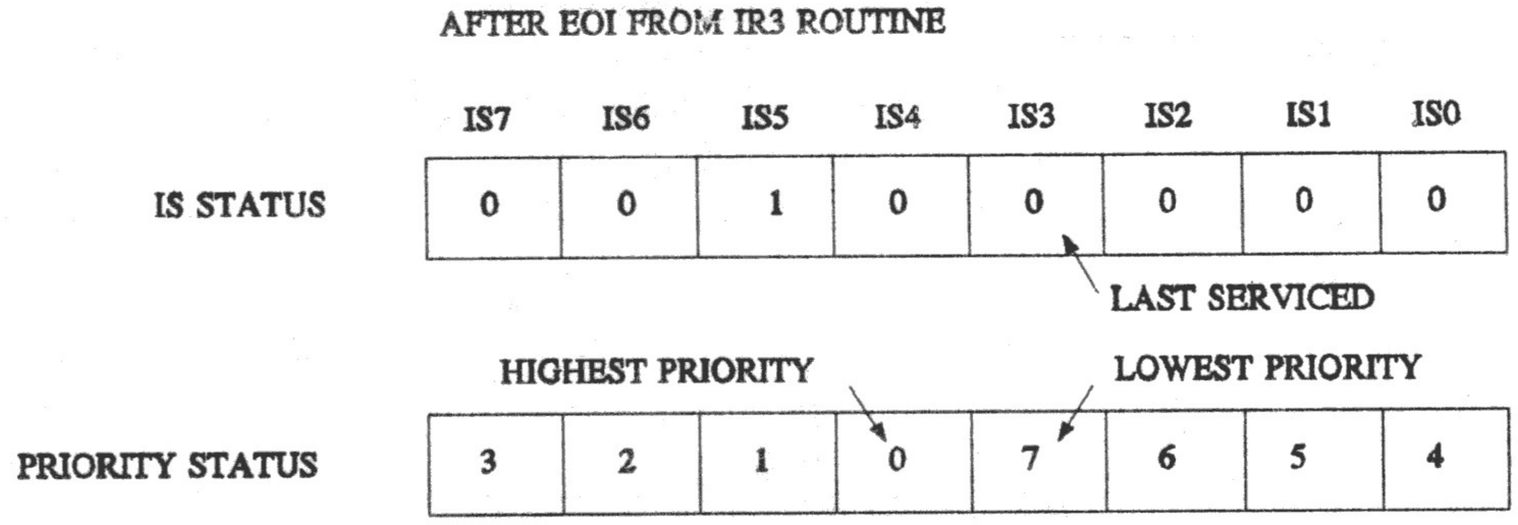

Figure 13.15b Priority Status After Rotation

(b) Specific Rotation

In the Automatic Rotation mode, the interrupt request *last* serviced is assigned the lowest priority, whereas in the Specific Rotation mode, the lowest priority can be assigned to *any* level IRX (X can vary from 0 to 7) as specified by OCW2. I(X+1) is then assigned the highest priority.

This mode is set by the CPU by issuing an OUT instruction with $A_0=0$, $D_7D_6D_5D_4D_3$ set to 11000, with D_2-D_0 specifying the interrupt level IRX that is to be assigned the lowest priority. It should be noted that this operation is independent of an EOI command. However, specific rotation can be accomplished by using the Rotate on Specific EOI option in OCW2 (R=1, SL=1, EOI=1,

L_2-L_0 specify the interrupt level IRX that is to be assigned the lowest priority on an EOI; see Fig. 13.13). In such a case, the priority changes are automatically made *after* the EOI command.

13.4.3 Special Mask Mode

It may be sometimes desirable to selectively enable *lower* priority interrupts. Usually, if an EOI command is not given to the 8259, the IS bit of the last serviced interrupt is not reset, and consequently all lower priority interrupts are kept disabled.

The Special Mask Mode can be set by making the ESMM and SMM bits '1' in OCW3. When a mask bit is set in OCW1, all further interrupts at that level are inhibited, while interrupts on all other levels *that are not masked by OCW1* (both lower and higher) are enabled. It is thus possible to selectively enable interrupts by programming the mask register.

The Special Mask Mode can be cleared by loading an OCW3 with ESSM=1 and SMM=0.

13.4.4 Polled Mode

In the Polled mode of operation, the INT output of the 8259 is either not connected to the INTR input of the 8085A, or the system interrupts are kept disabled by software. The devices are then serviced by the 8085A by polling the interrupt requests.

The Polled mode is set by making P=1 in OCW3. A subsequent Read command issued to the 8259 (with $\overline{RD}=0$, and $\overline{CS}=0$) is treated as an $\overline{INTA}$ by the 8259. It then sets the ISR bit corresponding to the highest level interrupt in the IRR and puts out a byte (the format of this byte is shown in Fig. 13.16) on the data bus. The 8085A can examine the status of D_7 to check if an interrupt needs to be serviced ($D_7=1$ implies that an interrupt is to be serviced, and $D_7=0$ implies that no interrupt has occured). W_0-W_2 give the code of the highest priority interrupt level requesting service.

If an interrupt is detected, the software must include a CALL to an interrupt service routine (the address is dependent on the level of the interrupt, as detected by W_0-W_2). Since the INTR line is not used in this mode, more than one 8259 may be connected in the master mode; *it is thus possible to have more than sixty-four levels of interrupts in this mode.*

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I	–	–	–	–	W_2	W_1	W_0

Figure 13.16 Polled Mode Output Word
(Source: Intel Corporation)

13.5 8259 STATUS READ OPERATIONS

The status of the Interrupt Request Register, the In-Service Register, and the Interrupt Mask Register of the 8259 may be read by issuing appropriate Read commands as described below.