8259-Programmable Interrupt Controller (8259-PIC)

Programmable Interface Device

A Programmable interface device is designed to perform various input/output functions. Such a device can be set up to perform specific functions by writing an instruction (or instructions) in its internal register, called the control word

Block Diagram

INTERRUPTS 397

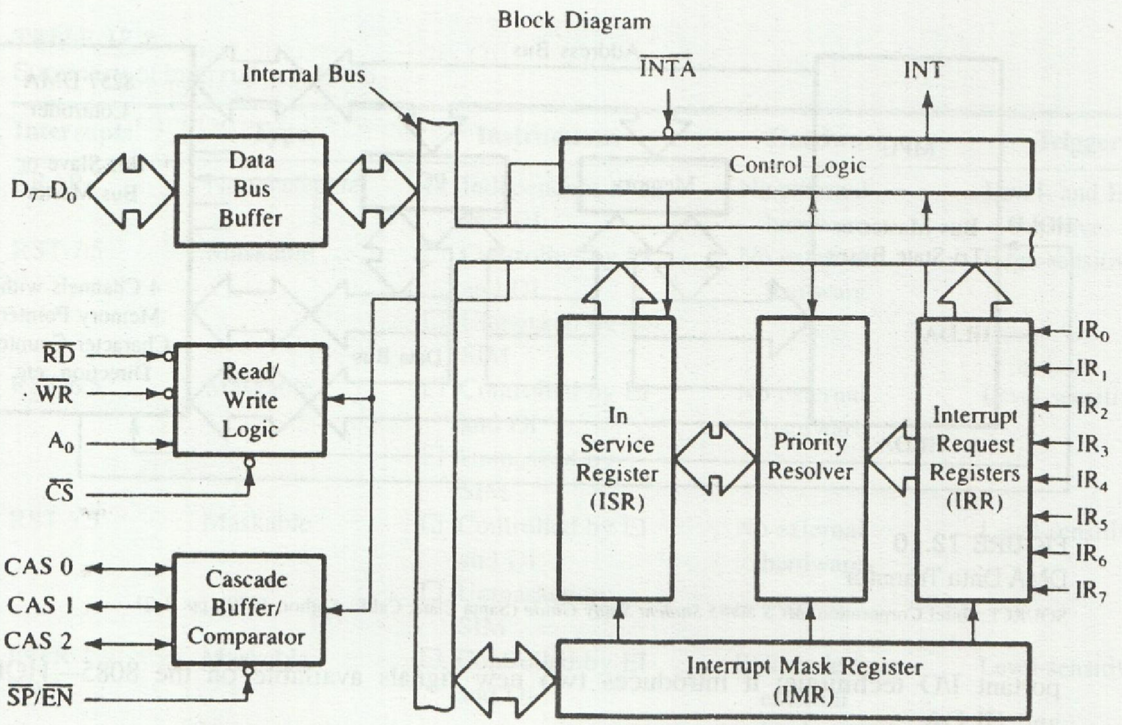


FIGURE 12.9
The 8259A Block Diagram
SOURCE: Intel Corporation, *Peripheral Components* (Santa Clara, Calif.: Author, 1993), pp. 3–171.

Features

The 8259 interrupt controller can:

- ✓ Manage eight interrupts according to the instructions written into its control registers. This is eqvt to providing eight interrupt pins on the processor in place of one INTR (8085) pin
- ✓ Vector an interrupt request anywhere in the memory map
- ✓ Resolve eight levels of interrupt priorities in a variety of modes, such as fully nested mode, automatic rotation mode, and specific rotation mode
- ✓ Mask each interrupt request individually
- ✓ Read the status of pending interrupts, in-service interrupts, and masked interrupts
- ✓ Be set up to accept either the level-triggered or the edge-triggered interrupt request
- ✓ Be expanded to 64 priority levels by cascading additional 8259s
- ✓ Be set up to work either with the 8085µP mode or the 8086/8088 µP mode

Description

The internal block diagram of the 8259 includes eight block

I. Control Logic

II. Read/Write Logic

III. Data Bus Buffer

IV. Three Registers IRR, ISR, and IMR

V. Priority Resolver and

VI. Cascade Buffer

Read/Write Logic

When the address line A0 is at logic 0, the controller is selected to write a command or read a status.

Control Logic

- ✓ This block has two pins INT (interrupt) as an output, $\overline{\textbf{INTA}}$ (Interrupt Acknowledge) as an input
- ✓ The INT is connected to the interrupt pin of the µP. Whenever a valid interrupt is asserted, this signal goes high. The $\overline{\textbf{INTA}}$ is the Interrupt Acknowledge signal from the µP

Interrupt Registers and Priority Resolver

- ✓ The Interrupt Request Register (IRR) has eight input lines (**IR_0-IR_7)** for interrupts. When these lines go high, the requests are stored in the register
- ✓ The In-Service Register (ISR) stores all the levels that are currently being serviced, and the Interrupt Mask Register (IMR) stores the masking bits of the interrupt lines to be masked
- ✓ The Priority Resolver (PR) examines these three registers and determines whether INT should be sent to the µP

Cascade Buffer/Comparator

- ✓ This block is used to expand the number of interrupt levels by cascading two or more 8259s

Interrupt Operation

To implement interrupts, the Interrupt Enable flip-flop in the µP should be enabled by writing the EI instruction, and the 8259 requires two types of control words; Initialization Command Words (ICWs) and Operational Command Words (OCWs)

ICWs:

ICWs are used to set up the proper conditions and specify RST vector addresses.

OCWs:

OCWs are used to perform functions such as masking interrupts, setting up status-read operations, etc.

After the initialization, the following sequence of events occurs when one or more interrupt request lines go high.

I. The IRR stores the requests

II. The priority resolver resloves the priority and sets the INT high when appropriate

III. The processor acknowledges the interrupt by sending $\overline{\text{INTA}}$

IV. After the $\overline{\text{INTA}}$ is received, the appropriate priority bit in the ISR is set to indicate which interrupt level is being serviced, and the corresponding bit in the IRR is reset to indicate that the request is accepted. Then, the opcode for the CALL instruction is placed on the data bus

V. When the processor decodes the CALL instruction, it places two more $\overline{\text{INTA}}$ signal on the data bus

VI. When the 8259 receives the second $\overline{\text{INTA}}$, it places the low order byte of the CALL address on the data bus. At the third $\overline{\text{INTA}}$, it places the high-order byte on the data bus. CALL address is the vector memory location for the interrupt; this is placed in the control register during the initialization.

VII. During the 3rd $\overline{\text{INTA}}$ pulse, the ISR bit is reset

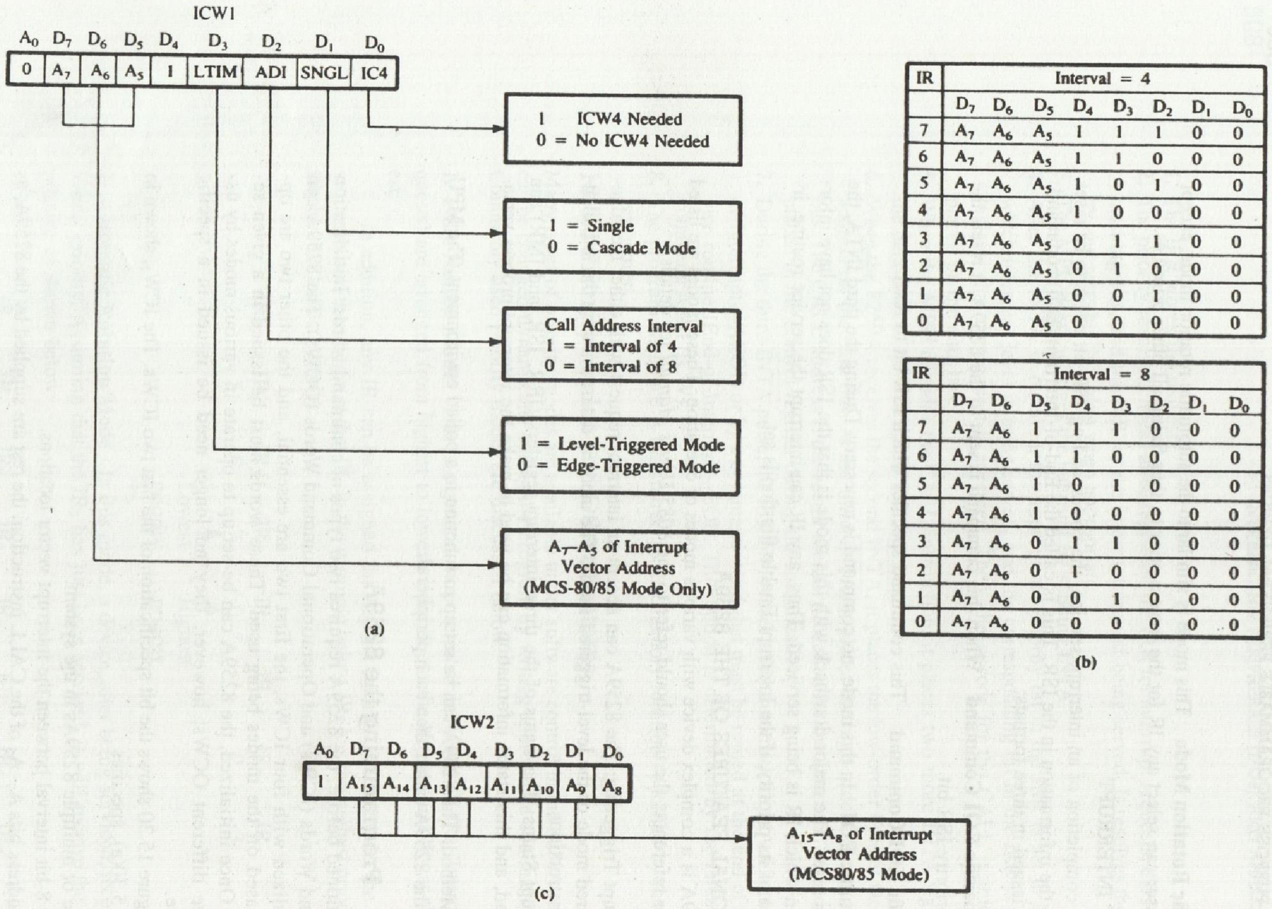
VIII. The program sequence is transferred to the memory location specified by the CALL

Programming the 8259

- ✓ The 8259 requires two types of command words, Initialization Command Words (ICWs) and Operational Command Words (OCWs)
- ✓ The 8259 can be initialized with four ICWs; the first two are essential, and the other two are optional based on the modes being used
- ✓ These words must be issued in a given sequence
- ✓ Once initialized, the 8259 can be set up to operate in various modes by using three different OCWs; however, they no longer need be issued in a specific sequence

(Please see page 510 in text book for ICW1 and ICW2)

- ✓ The ICW1 specifies; single or multiple 8259s in the system
- ✓ 4-or 8-bit interval between the interrupt vector location
- ✓ The address bits $\mathbf{A_7}$**-**$\mathbf{A_5}$ of the CALL instructions; the rest are supplied by the 8259


ICW1
A0 D7 D6 D5 D4 D3 D2 D1 D0
0 A7 A6 A5 1 LTIM ADI SNGL IC4
1 ICW4 Needed
0 = No ICW4 Needed
1 = Single
0 = Cascade Mode
Call Address Interval
1 = Interval of 4
0 = Interval of 8
1 = Level-Triggered Mode
0 = Edge-Triggered Mode
A7–A5 of Interrupt Vector Address (MCS-80/85 Mode Only)
(a)
ICW2
A0 D7 D6 D5 D4 D3 D2 D1 D0
1 A15 A14 A13 A12 A11 A10 A9 A8
A15–A8 of Interrupt Vector Address (MCS80/85 Mode)
(c)

IR	Interval = 4							
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
7	A_7	A_6	A_5	1	1	1	0	0
6	A_7	A_6	A_5	1	1	0	0	0
5	A_7	A_6	A_5	1	0	1	0	0
4	A_7	A_6	A_5	1	0	0	0	0
3	A_7	A_6	A_5	0	1	1	0	0
2	A_7	A_6	A_5	0	1	0	0	0
1	A_7	A_6	A_5	0	0	1	0	0
0	A_7	A_6	A_5	0	0	0	0	0

IR	Interval = 8							
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
7	A_7	A_6	1	1	1	0	0	0
6	A_7	A_6	1	1	0	0	0	0
5	A_7	A_6	1	0	1	0	0	0
4	A_7	A_6	1	0	0	0	0	0
3	A_7	A_6	0	1	1	0	0	0
2	A_7	A_6	0	1	0	0	0	0
1	A_7	A_6	0	0	1	0	0	0
0	A_7	A_6	0	0	0	0	0	0

(b)

FIGURE 15.30
Initialization Command Words for the 8259A

SOURCE: Intel Corporation, *Peripheral Components* (Santa Clara, Calif.: Author, 1993), p. 3-181.

NEXT CLASS

8251

THANK YOU