Bangladesh Army University of Engineering & Technology (BAUET)
Department of Computer Science and Engineering

Microprocessors and Micro-controller

Handling Interrupts

Prof. Dr. Md. Rabiul Islam
Dept.of Computer Science & Engineering
Rajshahi University of Engineering & Technology
Rajshahi-6204, Bangladesh.
rabiul.cse@gmail.com

Outline

- Interrupt Responses of 8086 Microprocessor
- Overview of Priority Interrupt Controller
- System Overview of 8259A PIC
- Internal Block Diagram of 8259A PIC
- Working Procedure of 8259A PIC
- System Connections of 8259A PIC
- Responses with Various Higher and Lower Priority Interrupts
- Cascaded Connection of 8259A PIC
  - Importance
  - Circuit Diagram
  - Working Procedure
  - Address Table

References

Chapter 8, 8086 Interrupts and Interrupt Applications, Douglas V. Hall, "Microprocessors and Interfacing, Programming and Hardware", 2nd Edition, Tata McGraw Hill, 2002.

Interrupt Responses of 8086 Microprocessor

At the end of each instruction cycle, the 8086 microprocessor checks to see if any interrupts have been requested. If an interrupt has been requested, the 8086 responds to the interrupt by stepping through the following series of major actions:

1. It decrements the stack pointer by 2 and pushes the flag register on the stack.
2. It disable the 8086 NTR interrupt input by clearing the interrupt clag (IF) in the flag register.

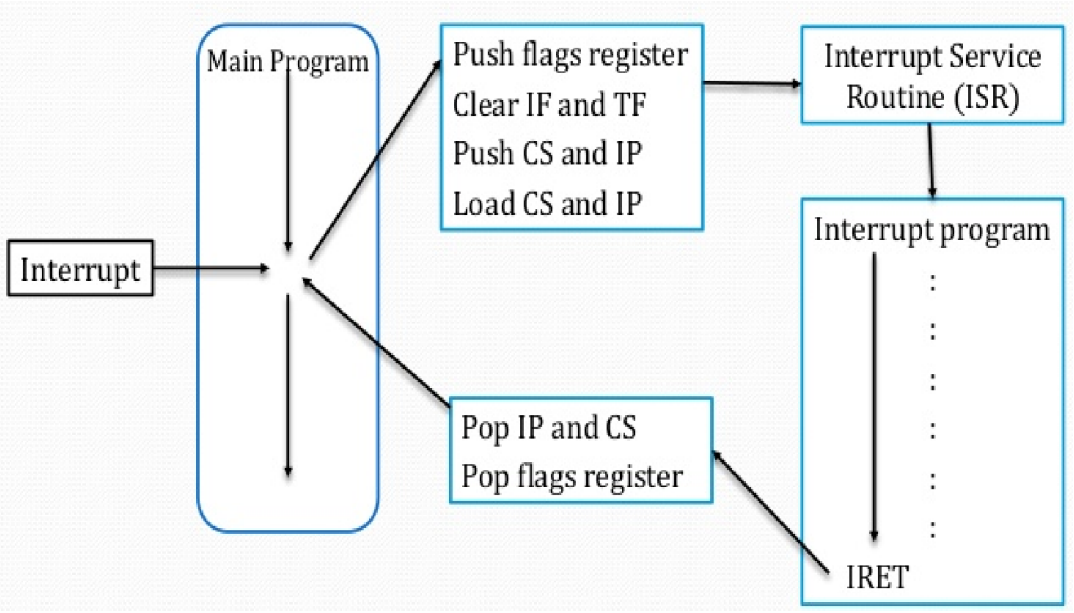


Figure: 8086 interrupt responses

3. It resets the trap flag (TF) in the flag register.
4. It decrements the stack pointer by 2 and pushes the current code segment register contents on the stack.
5. It decrements the stack pointer again by 2 and pushes the current instruction pointer contents on the stack.
6. It does an indirect far jump to the start of the procedure you wrote to respond to the interrupt.

Figure summaries these steps. The 80086 pushes the flag register on the stack, disables the INTR input and the single step function and does essentially an indirect far call to the interrupt service procedure. An IRET instruction at the end of the interrupt service procedure returns execution to the main program.

Overview of Priority Interrupt Controller

If we are working with an 8086, we have a problem here because the 8086 has only two interrupt inputs, NMI and INTR. If we save NMI for a power failure interrupt, this leaves only one interrupt input for all the other applications. For applications where we have interrupts from multiple sources, we use an external device called a *priority interrupt controller* (PIC) to "funnel" the interrupt signals into a single interrupt input on the processor.

System Overview of 8259A PIC

To show you how an 8259A functions in an 8086 system, we first need to review how the 8086 INTR input works. Remember from Figure and a discussion earlier in this chapter that if the 8086 interrupt flag is set and the INTR input receives a high signal, the 8086 will

1. Send out two interrupt acknowledge pulses on its $\overline{\text{INTA}}$ pin to the $\overline{\text{INTA}}$ pin of an 8259A PIC. The $\overline{\text{INTA}}$ pulses tell the 8259A to send the desired interrupt type to the 8086 on the data bus.
2. Multiply the interrupt type it receives from the 8259A by 4 to produce an address in the interrupt vector table.
3. Push the flags on the stack.
4. Clear IF and TF.
5. Push the return address on the stack.
6. Get the starting address for the interrupt procedure from the interrupt-vector table and load that address in CS and IP.
7. Execute the interrupt-service procedure.

Internal Block diagram of 8259A PIC

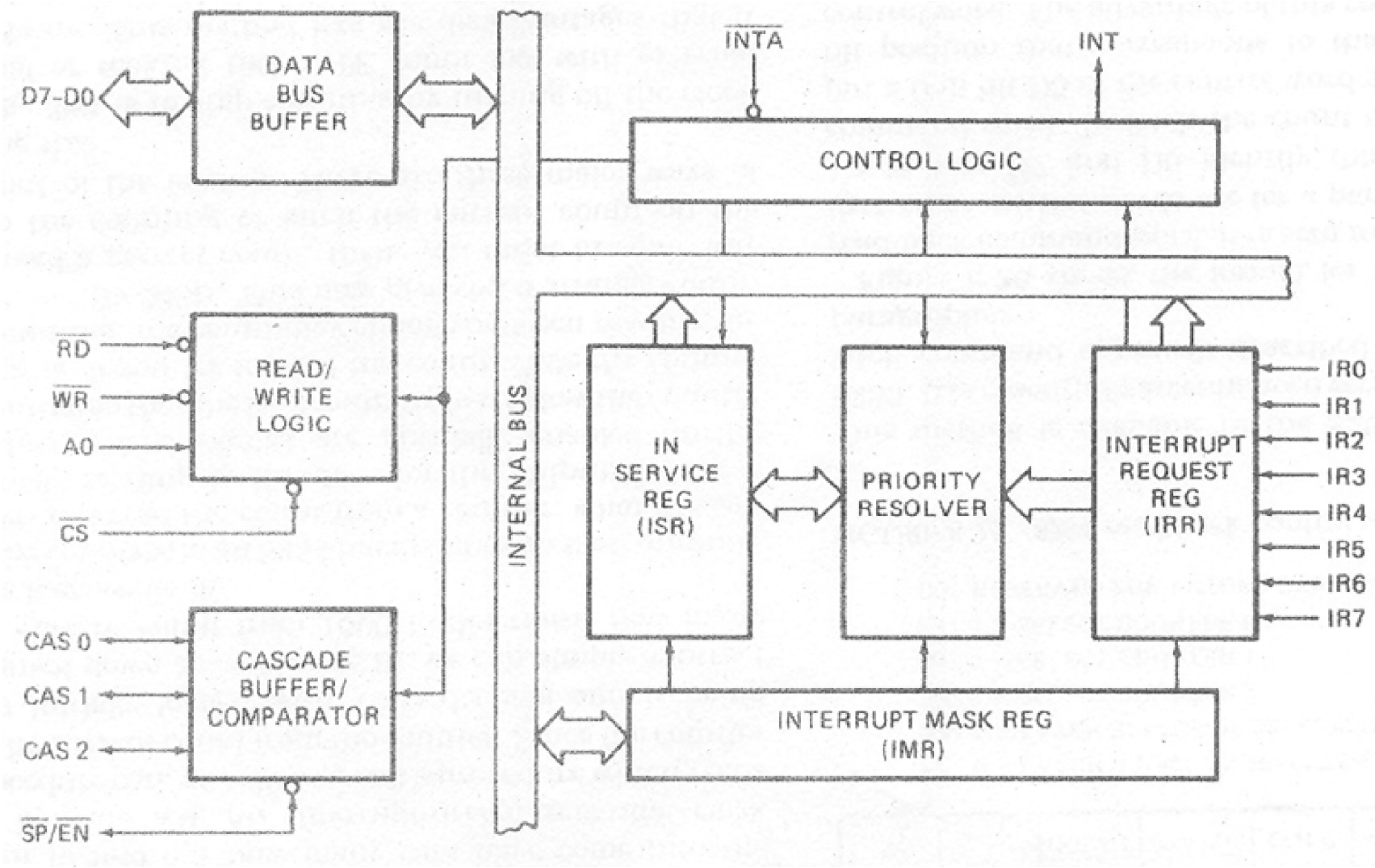


Figure: Internal block diagram of 8259A PIC

Working Procedure of 8259A PIC

in Figure observe the eight interrupt inputs labeled IR0 through IR7 on the right side of the diagram. If the 8259A is properly enabled, an interrupt signal applied to any one of these inputs will cause the 8259A to assert its INT output pin high. If this pin is connected to the INTR pin of an 8086 and if the 8086 interrupt flag is set, then this high signal will cause the previously described INTR response.

The $\overline{\text{INTA}}$ input of the 8259A is connected to the $\overline{\text{INTA}}$ output of the 8086. The 8259A uses the first $\overline{\text{INTA}}$ pulse from the 8086 to do some activities that depend on the mode in which it is programmed. When it receives the second $\overline{\text{INTA}}$ pulse from the 8086, the 8259A outputs an interrupt type on the 8-bit data bus, as shown in Figure The interrupt type that it sends to the 8086 is determined by the IR input that received an interrupt signal and by a number you send the 8259A when you initialize it. The point here is that the 8259A "funnels" interrupt signals from up to eight different sources into the 8086 INTR input, and it sends the 8086 a specified interrupt type for each of the eight interrupt inputs.

Working Procedure of 8259A PIC

At this point the question that may occur to you is, What happens if interrupt signals appear at, for example, IR2 and IR4 at the same time? In the *fixed-priority mode* that the 8259A is usually operated in, the answer to this question is quite simple. In this mode, the IR0 input has the highest priority, the IR1 input the next highest, and so on down to IR7, which has the lowest priority. What this means is that if two interrupt signals occur at the same time, the 8259A will service the one with the highest priority first, assuming that both inputs are unmasked (enabled) in the 8259A.

Now let's look again at the block diagram of the 8259A in Figure 8-27 so we can explain in more detail how the device will respond to multiple interrupt signals. In the block diagram note the four boxes labeled *interrupt request register* (IRR), *interrupt mask register* (IMR), *in-service register* (ISR), and *priority resolver*.

The interrupt mask register is used to disable (mask) or enable (unmask) individual interrupt inputs. Each bit in this register corresponds to the interrupt input with the same number. You unmask an interrupt input by sending a command word with a 0 in the bit position that corresponds to that input.

The interrupt request register keeps track of which interrupt inputs are asking for service. If an interrupt input has an interrupt signal on it, then the corresponding bit in the interrupt request register will be set.

Working Procedure of 8259A PIC

The in-service register keeps track of which interrupt inputs are currently being serviced. For each input that is currently being serviced, the corresponding bit will be set in the in-service register.

The priority resolver acts as a "judge" that determines if and when an interrupt request on one of the IR inputs gets serviced.

As an example of how this works, suppose that IR2 and IR4 are unmasked and that an interrupt signal comes in on the IR4 input. The interrupt request on the IR4 input will set bit 4 in the interrupt request register. The priority resolver will detect that this bit is set and check the bits in the in-service register (ISR) to see if a higher-priority input is being serviced. If a higher-priority input is being serviced, as indicated by a bit being set for that input in the ISR, then the priority resolver will take no action. If no higher-priority interrupt is being serviced, then the priority resolver will activate the circuitry which sends an interrupt signal to the 8086. When the 8086 responds with $\overline{\text{INTA}}$ pulses, the 8259A will send the interrupt type that was specified for the IR4 input when the 8259A was initialized. As we said before, the 8086 will use the type number it receives from the 8259A to find and execute the interrupt-service procedure written for the IR4 interrupt.

Working Procedure of 8259A PIC

Now, suppose that while the 8086 is executing the IR4 service procedure, an interrupt signal arrives at the IR2 input of the 8259A. This will set bit 2 of the interrupt request register. Since we assumed for this example that IR2 was unmasked, the priority resolver will detect that this bit in the IRR is set and make a decision whether to send another interrupt signal to the 8086. To make the decision, the priority resolver looks at the in-service register. If a higher-priority bit in the ISR is set, then a higher-priority interrupt is being serviced. The priority resolver will wait until the higher-priority bit in the ISR is reset before sending an interrupt signal to the 8086 for the new interrupt input. If the priority resolver finds that the new interrupt has a higher priority

Working Procedure of 8259A PIC

than the highest-priority interrupt currently being serviced, it will set the appropriate bit in the ISR and activate the circuitry which sends a new INT signal to the 8086. For our example here, IR2 has a higher priority than IR4, so the priority resolver will set bit 2 of the ISR and activate the circuitry which sends a new INT signal to the 8086. If the 8086 INTR input was reenabled with an STI instruction at the start of the IR4 service procedure, as shown in Figure 8-28*a*, then this new INT signal will interrupt the 8086 again. When the 8086 sends out a second $\overline{\text{INTA}}$ pulse in response to this interrupt, the 8259A will send it the type number for the IR2 service procedure. The 8086 will use the received type number to find and execute the IR2 service procedure.

Working Procedure of 8259A PIC

At the end of the IR2 procedure, we send the 8259A a command word that resets bit 2 of the in-service register so that lower-priority interrupts can be serviced. After that, an IRET instruction at the end of the IR2 procedure sends execution back to the interrupted IR4 procedure. At the end of the IR4 procedure, we send the 8259A a command word which resets bit 4 of the in-service register so that lower-priority interrupts can be serviced. An IRET instruction at the end of the IR4 procedure returns execution to the mainline program. This all sounds very messy, but it is really just a special case of nested procedures. Incidentally, if the IR4 procedure did not reenable the 8086 INTR input with an STI instruction, the 8086 would not respond to the IR2-caused INT signal until it finished executing the IR4 procedure, as shown in Figure *b*.

System Connections of 8259A PIC

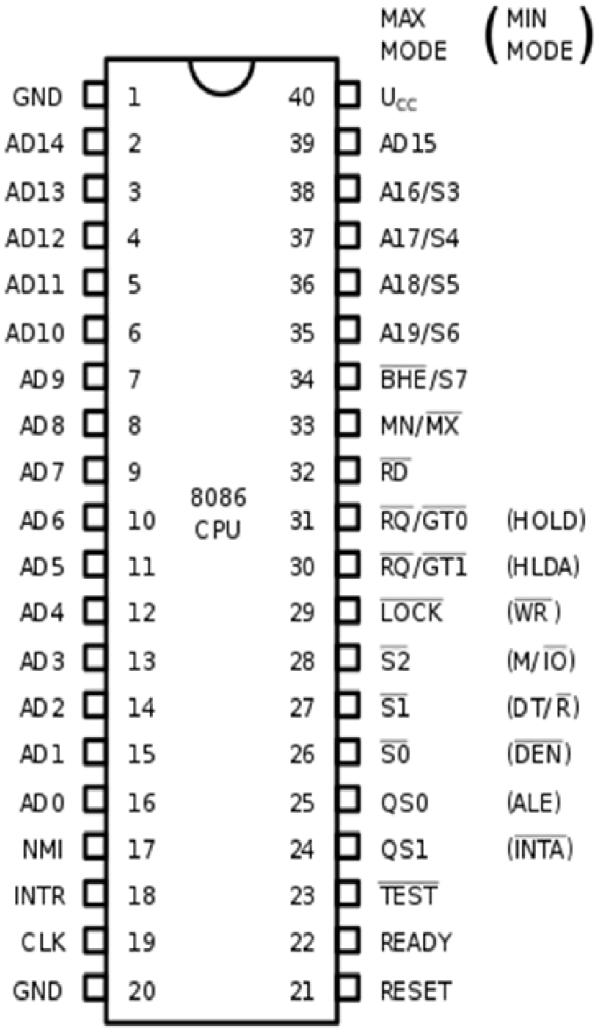
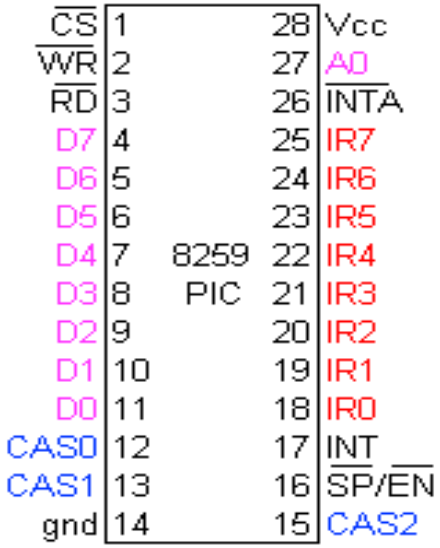


Fig: Pin-out diagram of 8086 Microprocessor



PIN	DESCRIPTION
D7 - D0	Data Bus (Bidirectional)
$\overline{RD}$	Read Input
$\overline{WR}$	Write Input
A0	Command Select Address
$\overline{CS}$	Chip Select
CAS 2 - CAS 0	Cascade Lines
$\overline{SP/EN}$	Slave Program Input Enable
INT	Interrupt Output
$\overline{INTA}$	Interrupt Acknowledge Input
IR0 - IR7	Interrupt Request Inputs

Fig: Pin-out diagram and pin function of 8259A PIC

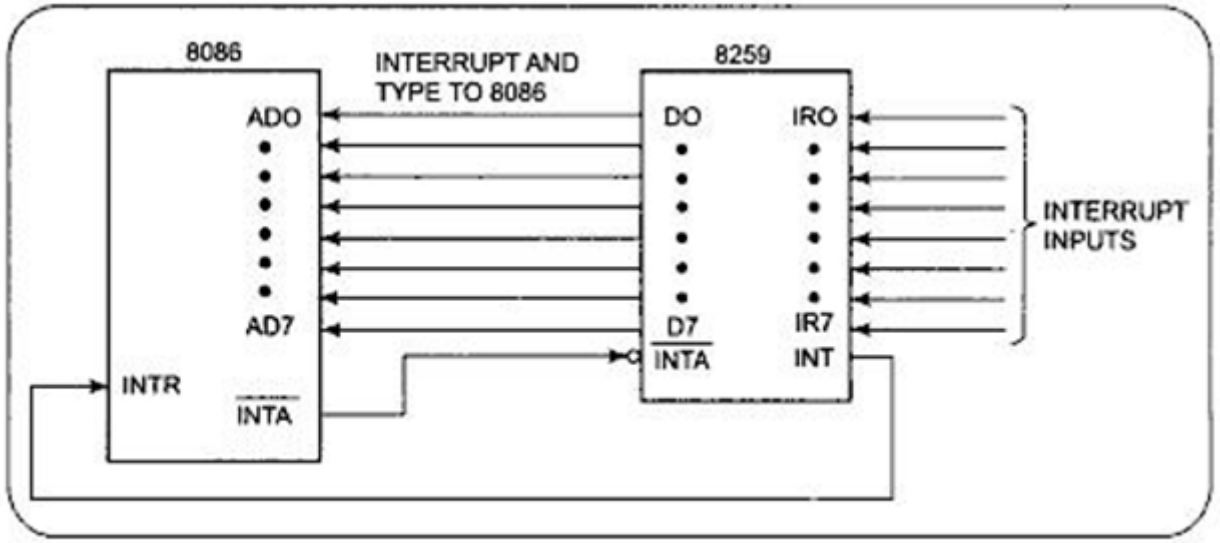
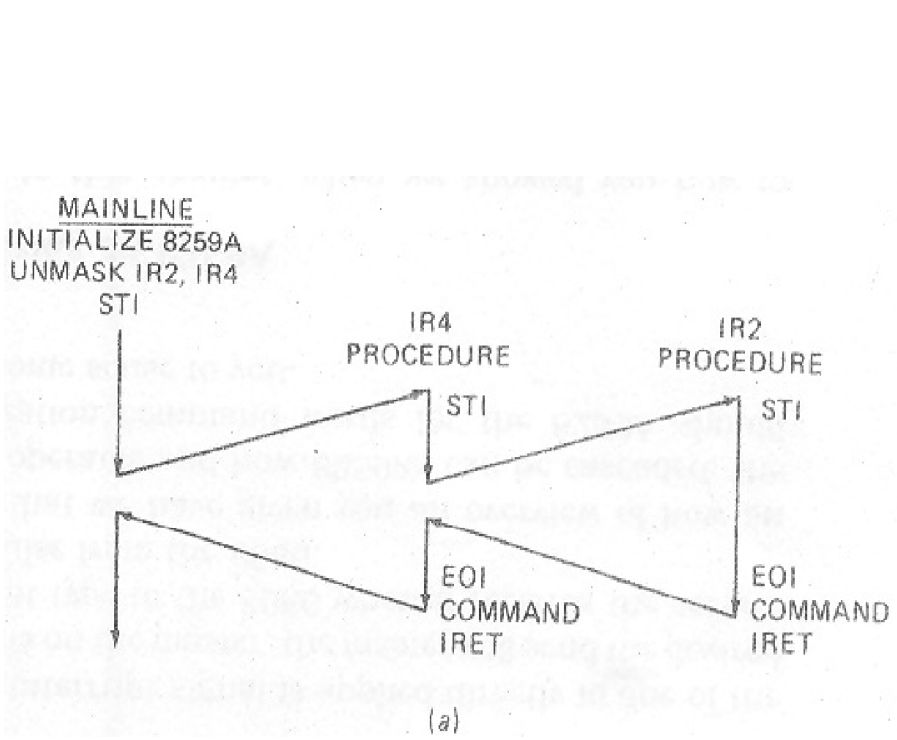


Fig. *Block diagram showing an 8259 connected to an 8086.*

Responses with Various Higher and Lower Priority Interrupts



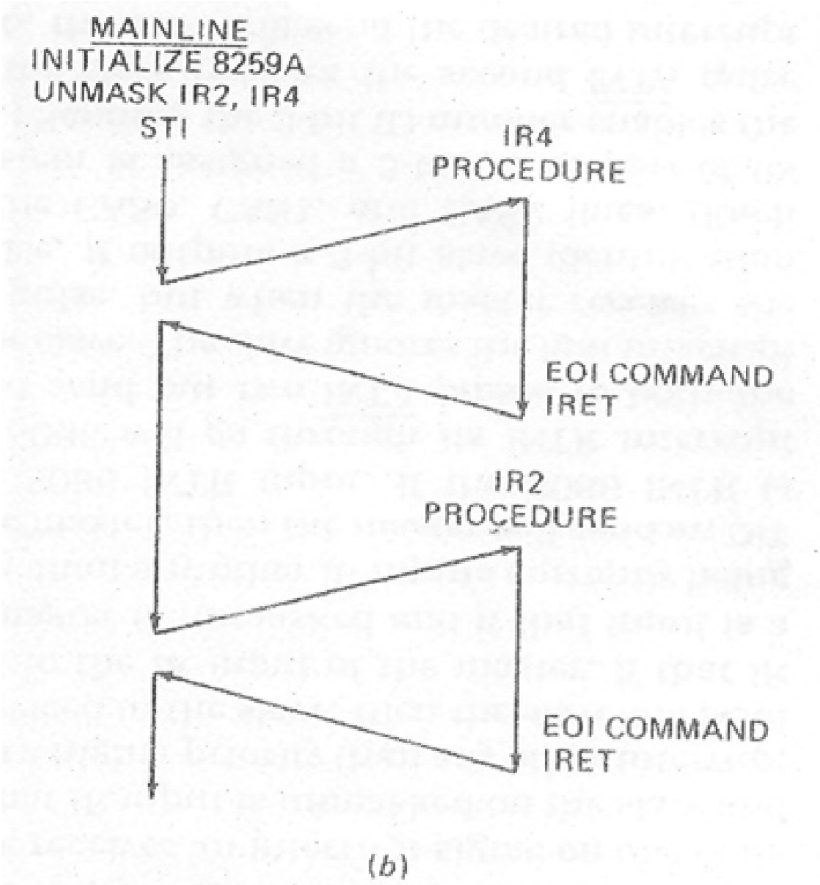


FIGURE 8259A and 8086 program flow for IR4 interrupt followed by IR2 interrupt. (*a*) Response with INTR enabled in IR4 procedure. (*b*) Response with INTR not enabled in IR4 procedure.

Cascaded Connections of 8259A PIC

Importance:

A signal 8259A provides eight vectored interrupts. If more interrupts are required, the 8259A is used in cascade mode. In cascade mode, a master 8259A along with eight slaves 8259A can provide upto 64 vectored interrupt lines. These three lines act as select lines for addressing the slave 8259A.

Cascaded Connections of 8259A PIC

Circuit Diagram

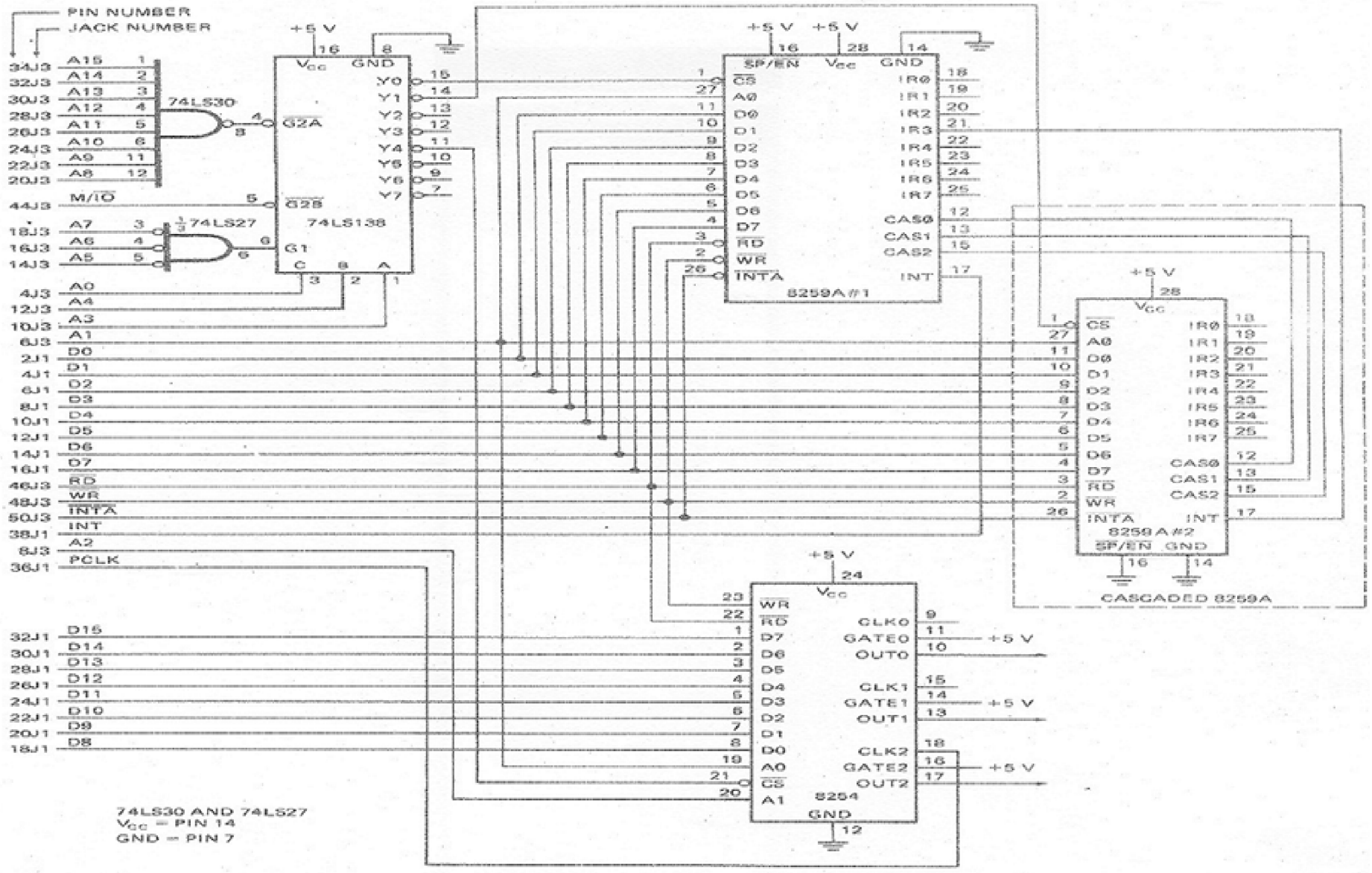


Figure: Circuit diagram of how to add 8259A PIC with cascaded connection

Cascaded Connections of 8259A PIC

Working Procedure

Figure shows how an 8259A can be added to an SDK-86 board. As shown by the truth table in Figure the 74LS138 address decoder will assert the $\overline{CS}$ input of the 8259A when an I/O base address of FF00H is on the address bus. The A0 input of the 8259A is used to select one of two internal addresses in the device. This pin is connected to system address line A1, so the system addresses for the two internal addresses of the 8259A are FF00H and FF02H. The eight data lines of the 8259A are always connected to the lower half of the 8086 data bus because the 8086 expects to receive interrupt types on these lower eight data lines. $\overline{RD}$ and $\overline{WR}$ are connected to the system $\overline{RD}$ and $\overline{WR}$ lines. $\overline{INTA}$ from the 8086 is connected to $\overline{INTA}$ on the 8259A.

Cascaded Connections of 8259A PIC

Working Procedure

The interrupt request signal, INT, from the 8259A is connected to the INTR input of the 8086. The multipurpose $\overline{SP}/\overline{EN}$ pin is tied high because we are using only one 8259A in this system. When just one 8259A is used in a system, the cascade lines (CAS0, CAS1, and CAS2) can be left open. The eight IR inputs are available for interrupt signals. Unused IR inputs should be tied to ground so that a noise pulse cannot accidentally cause an interrupt. In a later section we will show you how to initialize this 8259A, but first we need to show you how more than one 8259A can be added to a system.

The dashed box on the right side of Figure 8-14 shows how another 8259A could be added to the SDK-86 system to give a total of 15 interrupt inputs. If needed, an 8259A could be connected to each of the eight IR inputs of the original 8259A to give a total of 64 interrupt inputs. Note that since the 8086 has only one INTR input, only one of the 8259A INT pins is connected to the 8086 INTR pin. The 8259A connected directly into the 8086 INTR pin is referred to as the *master*. The INT pin from the other 8259A connects into an IR input on the master. This secondary, or *cascaded*, device is referred to as a *slave*. Note that the $\overline{INTA}$ signal from the 8086 goes to both the master and the slave devices.

Cascaded Connections of 8259A PIC

Working Procedure

Briefly, here is how the master and the slave work when the slave receives an interrupt signal on one of its IR inputs. If that IR input is unmasked on the slave and if that input is a higher priority than any other interrupt level being serviced in the slave, then the slave will send an INT signal to the IR input of the master. If that IR input of the master is unmasked and if that input is a higher priority than any other IR inputs currently being serviced in the master, then the master will send an INT signal to the 8086 INTR input. If the 8086 INTR is enabled, the 8086 will go through its INTR interrupt procedure and send out two $\overline{\text{INTA}}$ pulses to both the master and the slave. The slave ignores the first interrupt acknowledge pulse, but when the master receives the first $\overline{\text{INTA}}$ pulse, it outputs a 3-bit slave identification number on the CAS0, CAS1, and CAS2 lines. (Each slave in a system is assigned a 3-bit ID as part of its initialization.) Sending the 3-bit ID number enables the slave. When the slave receives the second $\overline{\text{INTA}}$ pulse from the 8086, the slave will send the desired interrupt type number to the 8086 on the lower eight data bus lines.

If an interrupt signal is applied directly to one of the IR inputs on the master, the master will send the desired interrupt type to the 8086 when it receives the second $\overline{\text{INTA}}$ pulse from the 8086.

Cascaded Connections of 8259A PIC

Address Table

The cascade pins (CAS0, CAS1, and CAS2) from the master are connected to the corresponding pins of the slave. For the master, these pins function as outputs, and for the slave device, they function as inputs. A further difference between the master and the slave is that on the slave the $\overline{SP}/\overline{EN}$ pin is tied low to let the device know that it is a slave.

A8–A15	A5–A7	A4	A3	A2	A1	A0	M/$\overline{IO}$	Y OUTPUT SELECTED	SYSTEM BASE ADDRESS	DEVICE
1	0	0	0	X	X	0	0	0	F F 0 0	8259A #1
1	0	0	1	X	X	0	0	1	F F 0 8	8259A #2
1	0	1	0	X	X	0	0	2	F F 1 0	
1	0	1	1	X	X	0	0	3	F F 1 8	
1	0	0	0	X	X	1	0	4	F F 0 1	8254
1	0	0	1	X	X	1	0	5	F F 0 9	
1	0	1	0	X	X	1	0	6	F F 1 1	
1	0	1	1	X	X	1	0	7	F F 1 9	
ALL OTHER STATES								NONE		

FIGURE Truth table for 74LS138 address decoder

Thanks

For any additional query please contact with me through

rabiul_cse@yahoo.com