Computer Peripherals and Interfacing

Computer Peripherals and Interfacing

Past Question Solve

TABLE OF CONTENTS

4th Chapter
8085 Microprocessor Architecture and Memory Interfacing

1. What do you know about the term MPU? Is 8085 an MPU? Why or Why not?
[2019-2(b), 2014-3(b)]
Ans:
MPU: The term microprocessor unit (MPU) is defined as a group of devices (as a unit) that can communicate with peripherals, provide timing signals, direct data flow, and perform computing tasks as specified by the instructions in memory.

Using the above description, the 8085 microprocessor can almost qualify as an MPU, but with the following two limitations:

- The low-order address bus of the 8085 microprocessor is multiplexed (time-shared) with the data bus. The buses need to be demultiplexed.
- Appropriate control signals need to be generated to interface memory and I/O with the 8085.

2. What are the control and status signals of 8085?
[2014-3(a)]
Ans: Control and status signals indicate the nature of the operation. The group of signals includes two control signals, three status signals, and one special signal.
Control Signals:
$\overline{RD}$ - Read: This is a read control signal which is active low. This signal indicates that the selected I/O or memory device is to be read and data is available on the data bus.
$\overline{WR}$- Write: This is a write control signal which is active low. This signal indicates that the data on the data bus are to be written into a selected memory or an I/O device.
Status Signals:
IO/$\overline{M}$: This is a status signal to differentiate between I/O and memory operation. When it is high it indicates an I/O Operation. When it is low it indicates a memory operation. This signal is combined with $\overline{RD}$(Read) and $\overline{WR}$(write) to generate I/O and Memory Control Signals.
$S_1 and\ S_0$:These status signals, similarly to IO/M can identify various operations, but they are rarely used in small systems.

3. What is ALE? Discuss how data and address buses of 8085 processors are de-multiplexed.
[2017-3(a), 2016-2(a), 2015-1(b), 2014-3(c)]
Ans:
ALE - Address Latch Enable: This is a positive going pulse and generated every time 8085 begins an operation; it indicates that bits on AD7 - AD0 are address bits. This signal is used to latch the low-order address from the multiplexed bus and generate a separate set of eight address lines; A7 - A0.

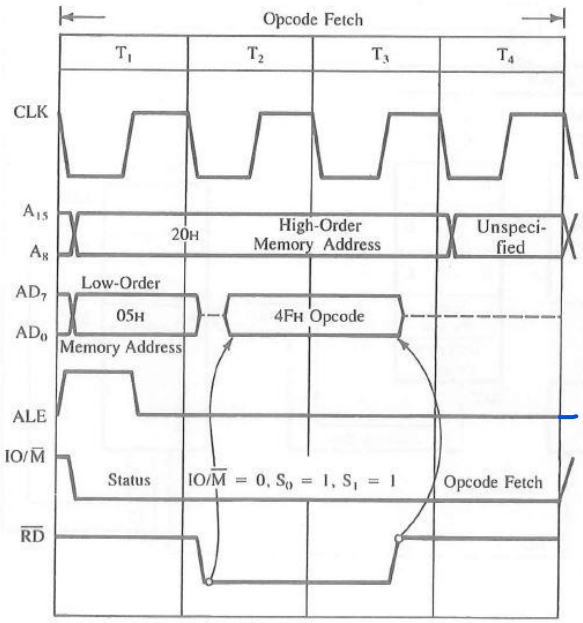


FIGURE 4.3
Timing: Transfer of Byte from Memory to MPU

The need for demultiplexing the bus AD_7–AD_0 becomes easier to understand after examining Figure 4.3. This figure shows that the address on the high-order bus (20H) remains on the bus for three clock periods. However, the low-order address (05H) is lost after the first clock period. This address needs to be latched and used for identifying the memory address. If the bus AD_7–AD_0 is used to identify the memory location (2005H), the address will change to 204FH after the first clock period.

Figure 4.4 shows a schematic that uses a latch and the ALE signal to demultiplex the bus. The bus AD_7–AD_0 is connected as the input to the latch 74LS373. The ALE signal is connected to the Enable (G) pin of the latch, and the Output control ($\overline{OC}$) signal of the latch is grounded.

Figure 4.3 shows that the ALE goes high during T_1. When the ALE is high, the latch is transparent; this means that the output changes according to input data. During T_1, the output of the latch is 05H. When the ALE goes low, the data byte 05H is latched until the next ALE, and the output of the latch represents the low-order address bus A_7–A_0 after the latching operation.

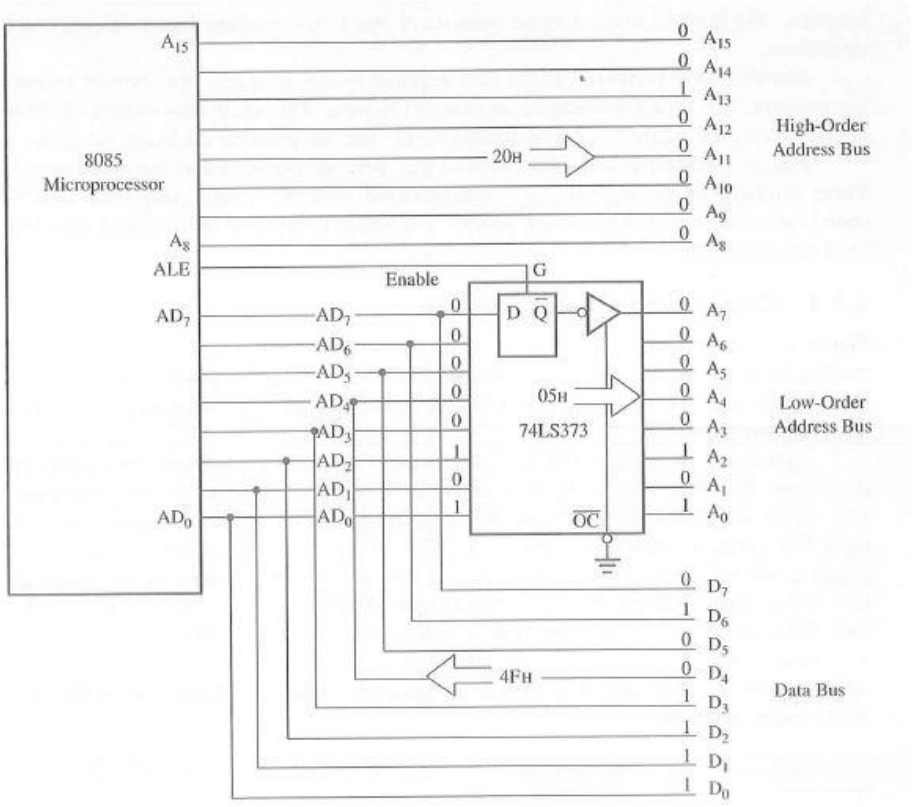


FIGURE 4.4
Schematic of Latching Low-Order Address Bus

Or,

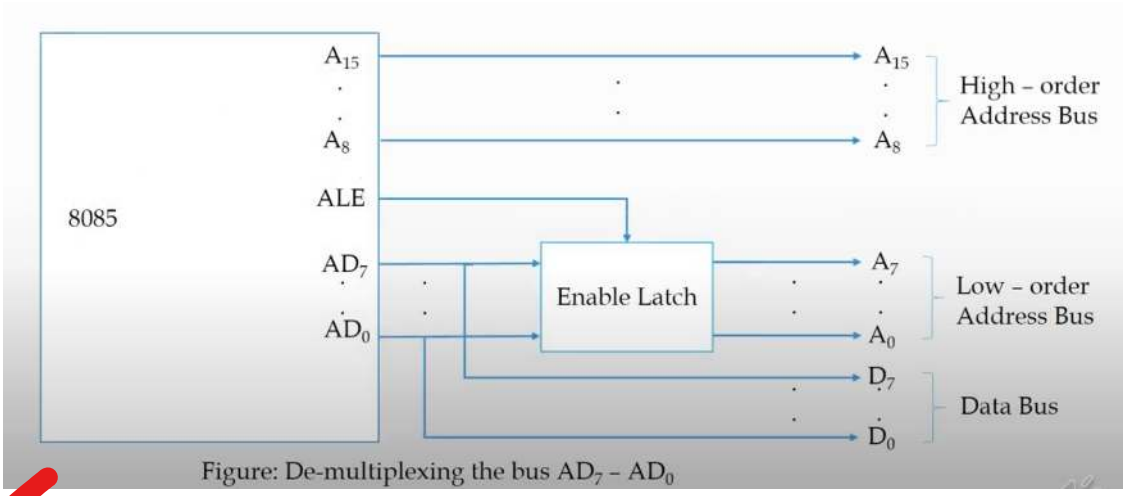


Figure: De-multiplexing the bus $AD_7 - AD_0$

4. Define instruction cycle, machine cycle, and T-state.
[2017-2(b)]
Ans:
Instruction Cycle: The instruction cycle is defined as the time required to complete the execution of an instruction.
Machine Cycle: The machine cycle is defined as the time required to complete one operation of accessing memory, I/O, or acknowledging an external request.
T-state: T-state is defined as one subdivision of the operation performed in one clock period.

5. **Mention the function of READY and HOLD signal of 8085 processor.**
[2015-2(c)]
Ans:
READY (Input): This signal is used to delay the microprocessor read or write cycles until a slow responding peripheral is ready to send or accept data.
HOLD(Input): This signal indicates that a peripheral is requesting to use the address and the data buses.

5th Chapter
Interfacing I/O Devices

1. **What is meant by computer peripheral interfacing? Explain with examples.**
[2019-1(a); 2018-1(a); 2017-1(a); 2016-1(a); 2015-1(a); 2014-1(a)]
Ans:
Computer Peripherals: Computer Peripherals are the components or devices which are connected to the processor and extend the work of the processor. All input and output devices are known as computer peripherals. For this reason, computer peripherals are sometimes called I/O devices.
In other words, Computer peripherals are the hardware that is connected to the CPU of the computer.

Interface: An interface is the concept of communication between two components, objects, elements, or two parts of a single object or component and it may occur on both hardware and software levels.
Simplifying, the interface is the interaction between two things.

Computer Peripherals and Interfacing: Computer Peripheral Interfacing is the interaction between computer processors and computer peripherals. Any input or output process that occurs through a peripheral device is known as computer peripheral interfacing.

Components of Computer Peripheral Interfacing: To interface among peripheral devices three components are needed.

- **I/O Processor**: I/O processor is the processor of the computer that means CPU. To interface between peripherals, there must be a central device that will control the interfacing.
- **I/O Module**: While computer peripheral interfacing there must be a mediator between peripherals and processors. Which is normally known as the I/O module.
- **I/O Device**: I/O device is the peripheral device with which the interface will occur. This can be any input or output device.

Examples of Computer Peripheral Interfacing:
- Taking input from a keyboard is peripheral interfacing. Here the keyboard is the peripheral device.
- Printing a document with a printer is peripheral interfacing where the printer is the peripheral device.
- Showing a video on the monitor is peripheral interfacing and this case monitor is the peripheral device.

. Discuss port-addressed I/O and memory-mapped I/O with control signals and instructions.

[2018-1(b); 2017-1(b); 2016-3(a); 2015-3(a); 2014-1(c)]

Ans:

Port-Addressed I/O:

In peripheral I/O an 8-bit port is used to interface I/O with the processor. In peripheral I/O the 8085 microprocessor uses two instructions for data transfer between the processor and the I/O devices. The instructions are IN and OUT. The instruction IN inputs data from an input device to the accumulator and the instructions OUT sends the contents of the accumulator to the output device. These are two bytes instructions with the second byte specifying the address or the port number of an I/O device.

For example, OUT instructions is described as follows:

Opcode	Operand	Description
OUT	8-bit Port Address	This is a two-byte instruction with the hexadecimal opcode D3 and the second byte is the port address of the output device.

Typically, to display the contents of the accumulator at an output device with the address, for example. 01H, the instruction will be written and stored in memory as follows:

Memory Address	Machine Code	Mnemonics	Machine Contents
2050	D3	OUT 01H	2050 → 11010011 = D3H
2051	01		2051 → 00000001 = 01H

(It is assumed here that the instruction is stored in memory locations 2050H, 2051H)

If the output port with the address 01H is designed as an LED display, the instruction OUT will display the contents of the accumulator at the port.

Similarly, IN instructions is described as follows:

Opcode	Operand	Description
IN	8-bit Port Address	This is a two-byte instruction with the hexadecimal opcode DB and the second byte is the port address of the input device.

Typically, to take input from an input device with the address, for example. 84H, the instruction will be written and stored in memory as follows:

Memory Address	Machine Code	Mnemonics	Machine Contents
2065	DB	IN 84H	2065 → 11011011 = DBH
2066	84		2066 → 10000100 = 84H

(It is assumed here that the instruction is stored in memory locations 2065H, 2066H)

If the input port with the address 84H is connected with any input device, the instruction IN will accept the data from the device.

The second byte of this IN/OUT instruction can be any 256 combinations of 8 bits from 00H to FFH. Therefore 8085 can communicate with 256 different input/output ports with device addresses ranging from 00H and FFH.

The Control Signals for input/output are IOR/IOW.

Memory-Mapped I/O:

In memory-mapped I/O, the input and output devices are assigned and identified by 16-bit addresses. To transfer data between the MPU and I/O devices, memory-related instructions (such as LDA, STA, etc.) and memory control signals MEMR and MEMW are used.
For example, STA instructions is described as follows:

Memory Address	Machine Code	Mnemonics	Comments
2050	32	STA 8000H	Store contents of accumulator in memory location 8000H
2051	00		
2052	80		

(It is assumed here that the instruction is stored in memory locations 2050H, 51H, 52H)

The STA is a three-byte instruction. The first byte is the opcode and the second and third bytes specify the memory address. However, the 16-bit address 8000H is entered in the reverse order. The low order byte 00H is stored in location 2051 and the high order byte 80H is stored in location 2052.
In this example, if an output device, instead of a memory register, is connected at this address, the contents of the accumulator will be transferred to the output device. This is called the memory-mapped I/O technique.

On the other hand, the instruction LDA transfers the data from a memory location to the accumulator.
To use memory-related instructions for data transfer the control signals MEMR, MEMW should be connected to the I/O device instead of IOR and IOW signals.

Distinguish between port-addressed I/O and memory-mapped I/O.
[2019-1(b)]
Ans:
Port-Addressed I/O: In Port addressed I/O or Peripheral I/O or I/O mapped I/O an 8-bit port is used to interface I/O with the processor. Intel 8085 processor uses peripheral I/O for data transfer.

Memory-mapped I/O: In memory-mapped I/O a 16-bit memory register is used to interface I/O with the processor. Motorola 68000 processor uses memory-mapped I/O for data transfer.

Differences between port-addressed I/O and memory-mapped I/O are given below.

Port-Addressed I/O	Memory-Mapped I/O
It uses an 8-bit address.	It uses a 16-bit address.
Control Signals for input/output is IOR/IOW.	Control Signals for input/output is MEMR/MEMW.
Available instructions are IN and OUT.	Available instructions are STA, LDA, MOV M, R, ADD M, SUB M, etc.
Data transfer occurs between I/O devices and accumulators.	Data transfer occurs between I/O devices and registers.

The I/O map is independent of the memory map. 256 input devices and 256 output devices can be connected.	The memory map is shared between I/Os and system memory.
Execution speed is 10 T-states	Execution speed is 13 T-states for STA, LDA, and 7 T-states for MOV M, R.
Less Hardware is needed to decode 8-bit addresses.	More hardware is needed to decode 16-bit addresses.
Arithmetic and Logical Operations can not be directly performed with I/O data.	Arithmetic and Logical Operations can be directly performed with I/O data.

Comparison of Memory-Mapped I/O and Peripheral I/O

Characteristics	Memory-Mapped I/O	Peripheral I/O
1. Device address	16-bit	8-bit
2. Control signals for Input/Output	$\overline{MEMR}$/$\overline{MEMW}$	$\overline{IOR}$/$\overline{IOW}$
3. Instructions available	Memory-related instructions such as STA; LDA; LDAX; STAX; MOV M,R: ADD M; SUB M; ANA M: etc.	IN and OUT
4. Data transfer	Between any register and I/O	Only between I/O and the accumulator
5. Maximum number of I/Os possible	The memory map (64K) is shared between I/Os and system memory	The I/O map is independent of the memory map; 256 input devices and 256 output devices can be connected
6. Execution speed	13 T-states (STA,LDA) 7 T-states (MOV M,R)	10 T-states
7. Hardware requirements	More hardware is needed to decode 16-bit address	Less hardware is needed to decode 8-bit address
8. Other features	Arithmetic or logical operations can be directly performed with I/O data	Not available

4. Is it possible to interfacing IN 40H and OUT 40H? Explain Why and Why not?
Or, Is it possible to interfacing IN 30H and OUT 30H? Explain Why and Why not?
Or, Is it possible to interfacing IN 20H and OUT 20H? Explain Why and Why not?
[2019-1(c), 2018-1(c), 2016-2(b), 2015-1(c), 2014-2(a)]

Ans: Yes. It is possible to interfacing IN 40H and OUT 40H.

Since the instructions are IN, OUT. So it's a port-addressed I/O. And since port-addressed I/O uses an 8-bit address. So, the second byte of IN and OUT instructions can be any of the 256 combinations of eight bits from 00H to FFH. Since 40H falls between 00H and FFH, it is possible in interfacing IN 40H and OUT 40H.

5. Discuss an interfacing circuit to design a safety control system for home appliances using a decoder, a buffer as an input port, and a latch as an output port. Also, write the program to operate the controlling system.
[2019-3(b), 2018-3(b), 2016-3(b), 2014-4(b)]
Ans:

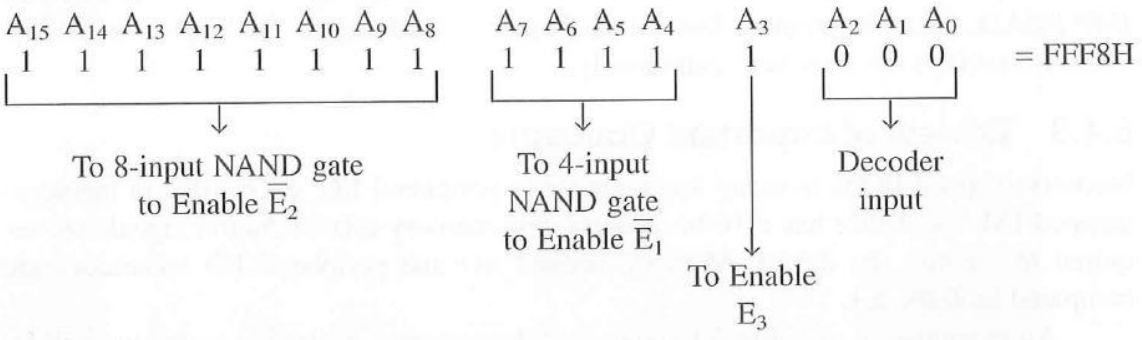
Figure 5.13 shows a schematic of interfacing I/O devices using the memory-mapped I/O technique. The circuit includes one input port with eight DIP switches and one output port to control various processes and gates, which are turned on/off by the microprocessor according to the corresponding switch positions. For example, switch S_7 controls the cooling system, and switch S_0 controls the exit gate. All switch inputs are tied high; therefore, when a switch is open (off), it has +5 V, and when a switch is closed (on), it has logic 0. The circuit includes one 3-to-8 decoder, one 8-input NAND gate, and one 4-input NAND gate to decode the address bus. The output O_0 of the decoder is combined with control signal $\overline{MEMW}$ to generate the device select pulse that enables the octal latch. The output O_1 is combined with the control signal $\overline{MEMR}$ to enable the input port. The eight switches are interfaced using a tri-state buffer 74LS244, and the solid state relays controlling various processes are interfaced using an octal latch (74LS373) with tri-state output.

OUTPUT PORT AND ITS ADDRESS

The various process control devices are connected to the data bus through the latch 74LS373 and solid state relays. If an output bit of the 74LS373 is high, it activates the corresponding relay and turns on the process; the process remains on until the bit stays high. Therefore, to control these safety processes, we need to supply an appropriate bit pattern to the latch.

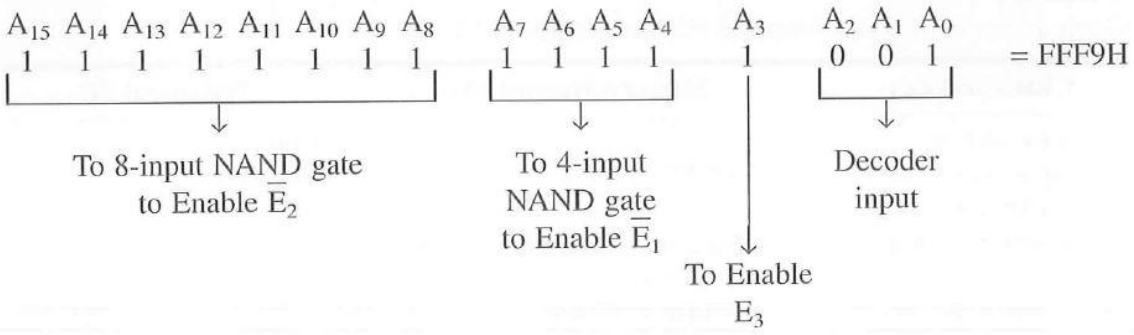
The 74LS373 is a latch followed by a tri-state buffer, as shown in Figure 5.13. The latch and the buffer are controlled independently by the Latch Enable (LE) and Output Enable ($\overline{OE}$). When LE is high, the data enter the latch, and when LE goes low, data are latched. The latched data are available on the output lines of the 74LS373 if the buffer is enabled by $\overline{OE}$ (active low). If $\overline{OE}$ is high, the output lines go into the high impedance state.

Figure 5.13 shows that the $\overline{OE}$ is connected to the ground; thus, the latched data will keep the relays on/off according to the bit pattern. The LE is connected to the device select pulse, which is asserted when the output O_0 of the decoder and the control signal $\overline{MEMW}$ go low. Therefore, to assert the I/O select pulse, the output port address should be FFF8H, as shown below:



INPUT PORT AND ITS ADDRESS

The DIP switches are interfaced with the 8085 using the tri-state buffer 74LS244. The switches are tied high, and they are turned on by grounding, as shown in Figure 5.13. The switch positions can be read by enabling the signal $\overline{OE}$, which is asserted when the output O_1 of the decoder and the control signal $\overline{MEMR}$ go low. Therefore, to read the input port, the port address should be



Instructions To control the processes according to switch positions, the microprocessor should read the bit pattern at the input port and send that bit pattern to the output port. The following instructions can accomplish this task:

```
READ: LDA FFF9H     ;Read the switches
      CMA           ;Complement switch reading, convert on-switch (logic 0)
                    ;   into logic 1 to turn on appliances
      STA FFF8H     ;Send switch positions to output port and turn on/off appli-
                    ;   ances
      JMP READ      ;Go back and read again
```

When this program is executed, the first instruction reads the bit pattern 1011 0111 (B7H) at the input port FFF9H and places that reading in the accumulator; this bit pattern represents the on-position of switches S_6 and S_3. The second instruction complements the

reading; this instruction is necessary because the on-position has logic 0, and to turn on solid state relays logic 1 is necessary. The third instruction sends the complemented accumulator contents (0100 1000 = 48H) to the output port FFF8H. The 74LS373 latches the data byte 0100 1000 and turns on the heating system and lights. The last instruction, JMP READ, takes the program back to the beginning and repeats the loop continuously. Thus, it monitors the switches continuously.

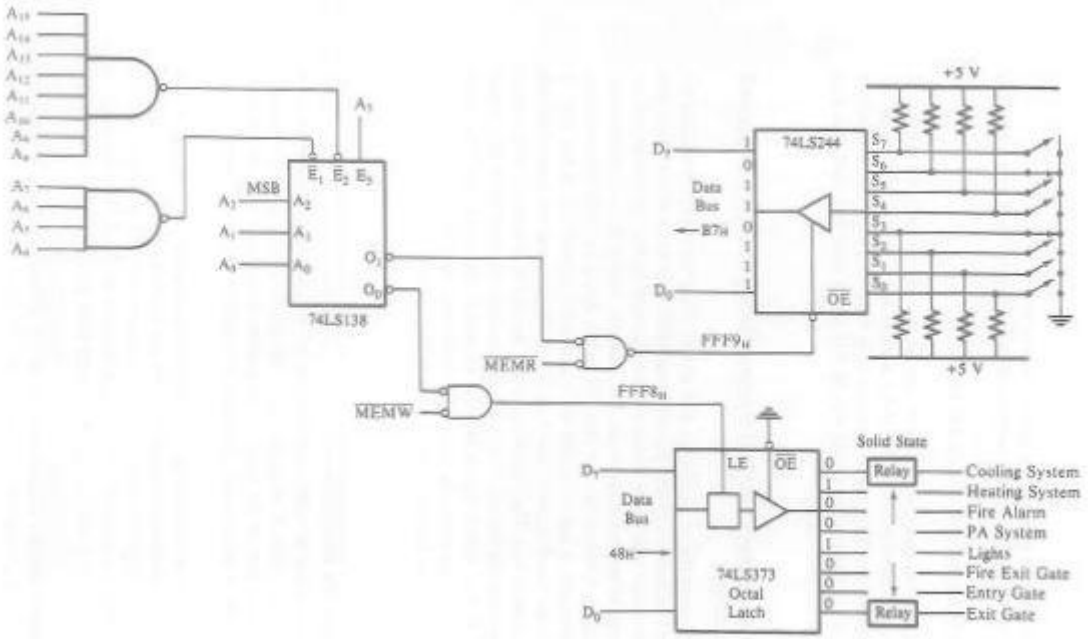


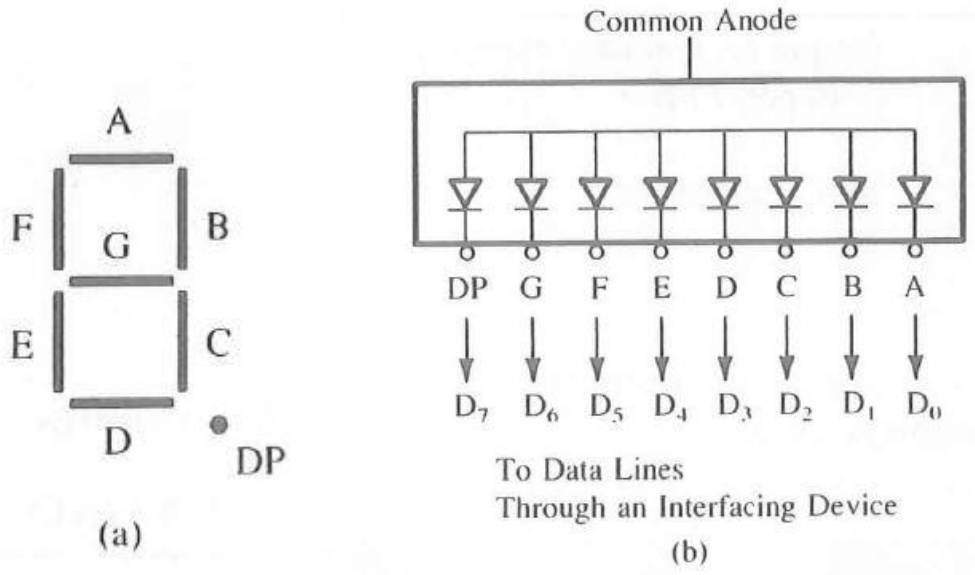
FIGURE 8.13
Memory-Mapped I/O Interfacing

6. Discuss how to interface common anode seven segment display LED with 8085 with proper circuit diagram. Also, Write the program to display sequentially digit 0-9 in the seven segments LED.

[2018-2(b), 2016-3(b), 2014-2(b)]

Ans:

Seven Segment LED: A seven-segment LED consists of seven light-emitting diodes and one segment for the decimal point. The LEDs are physically arranged as shown in Figure 4.9 (a). To display a number, the necessary segments are lit by sending appropriate signals for current flow through diodes.



The seven segments, A through G, are usually connected to data lines D_0 through D_6, respectively. If the decimal-point segment is being used, data line D_7 is connected to DP; otherwise it is left open. The binary code required to display a digit is determined by the type of the seven-segment LED (common cathode or common anode), the connections of the data lines, and the logic required to light the segment. For example, to display digit 7 at the LED in Figure 5.10, the requirements are as follows:

1. It is a common-anode seven-segment LED, and logic 0 is required to turn on a segment.
2. To display digit 7, segments A, B, and C should be turned on.
3. The binary code should be

Data Lines	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
Bits	X	1	1	1	1	0	0	0	= 78H
Segments	NC	G	F	E	D	C	B	A	

The code for each digit can be determined by examining the connections of the data lines to the segments and the logic requirements.

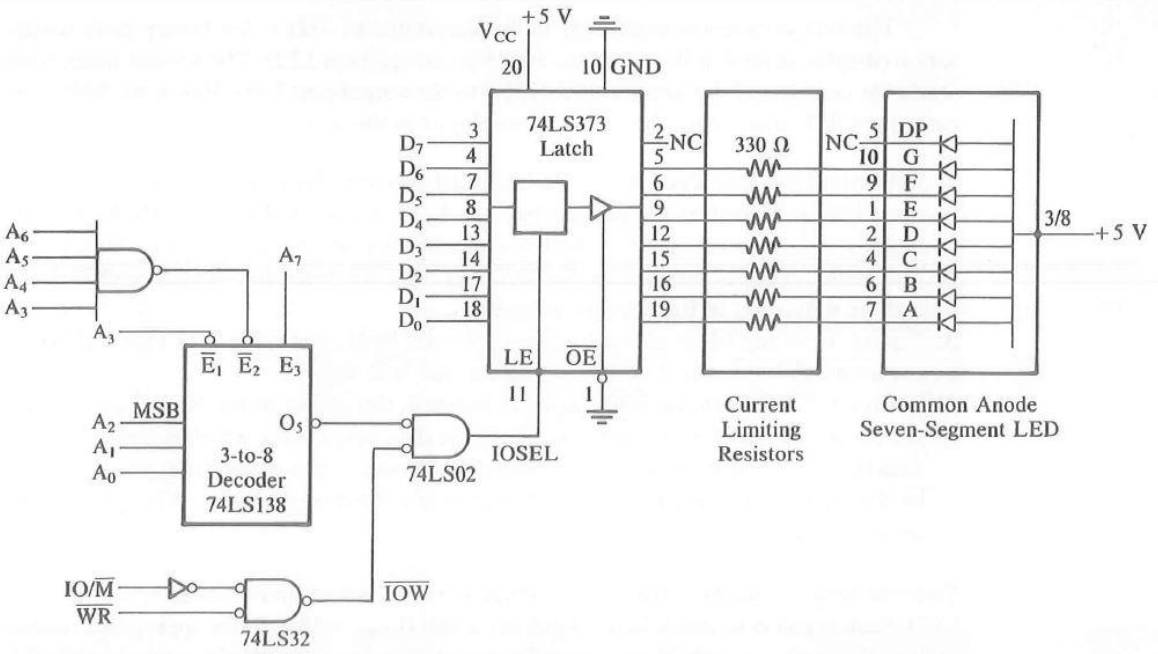


FIGURE 5.10
Interfacing Seven-Segment LED

INTERFACING CIRCUIT AND ITS ANALYSIS

To design an output port with the address F5H, the address lines A_7–A_0 should have the following logic:

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
1	1	1	1	0	1	0	1	= F5H

This can be accomplished by using A_2, A_1, and A_0 as input lines to the decoder. A_3 can be connected to active low enable $\overline{E}_1$, and the remaining address lines can be connected to $\overline{E}_2$ through the 4-input NAND gate. Figure 5.10 shows an output port with the address F5H. The output O_5 of the decoder is logically ANDed with the control signal $\overline{IOW}$ using the NOR gate (74LS02). The output of the NOR gate is the I/O select pulse that is used to enable the latch (74LS373). The control signal $\overline{IOW}$ is generated by logically ANDing $IO/\overline{M}$ and $\overline{WR}$ signals in the negative NAND gate (physically OR gate 74LS32).

Instructions:

The following instructions are necessary to display sequentially digit 0-9 at the seven segments LED.

```
MVI A,40H     ; Load code to display digit 0 in seven segments LED
OUT F5H       ; Display digit 0 at port F5H
MVI A,79H     ; Load code to display digit 1 in seven segments LED
OUT F5H       ; Display digit 1 at port F5H
MVI A,24H     ; Load code to display digit 2 in seven segments LED
OUT F5H       ; Display digit 2 at port F5H
MVI A,30H     ; Load code to display digit 3 in seven segments LED
OUT F5H       ; Display digit 3 at port F5H
MVI A,19H     ; Load code to display digit 4 in seven segments LED
OUT F5H       ; Display digit 4 at port F5H
```

```
MVI A,12H     ; Load code to display digit 5 in seven segments LED
OUT F5H       ; Display digit 5 at port F5H
MVI A,02H     ; Load code to display digit 6 in seven segments LED
OUT F5H       ; Display digit 6 at port F5H
MVI A,78H     ; Load code to display digit 7 in seven segments LED
OUT F5H       ; Display digit 7 at port F5H
MVI A,00H     ; Load code to display digit 8 in seven segments LED
OUT F5H       ; Display digit 8 at port F5H
MVI A,10H     ; Load code to display digit 9 in seven segments LED
OUT F5H       ; Display digit 9 at port F5H
```

7. **If the clock frequency is 5 MHz, how much time is required to execute an instruction of 18 T-states?**
[2018-2(a)]
Ans:
Given that,
Clock Frequency = 5 MHz
Instruction Execution Speed = 18 T-states

So,
T-states = (1/Clock Frequency) = (1/5) = 0.2 μs
Required time to execute an instruction of 18 T-states = (18 x 0.2) = 3.6 μs

8. **Define absolute address decoding and partial address decoding with example.**
[2017-4(a), 2016-2(c)2015-3(b)]
Ans:
Absolute Decoding: If all lines are decoded to generate one unique output pulse then it'll be called absolute decoding and good design practice. Such as in Fig 4.4 all eight lines are decoded to generate one unique output pulse. The device will be selected only with the address, 01H.

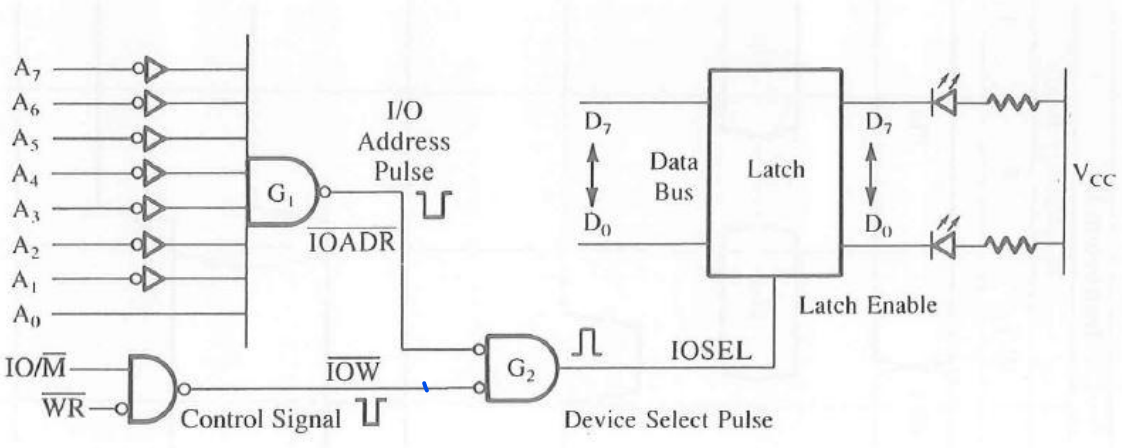


FIGURE 5.4
Decode Logic for LED Output Port
NOTE: To use this circuit with the 8085, the bus AD_7–A_0 must be demultiplexed.

Partial Decoding: To minimize the cost, the output port can be selected by decoding some of the address lines, this is called partial decoding. Such as in Fig 5.5 the address lines A_0

and A_1 are not connected, and they are replaced by IO/M and WR signals. Because the address lines A_0 and A_1 are at don't care logic level, they can be assumed 0 or 1. Thus this output port (latch) can be accessed by the addresses 00H, 01H, 02H, and 03H.

Partial decoding is a commonly used technique in small systems. Such multiple addresses will not cause any problem, provided these addresses are not assigned to any other output ports.

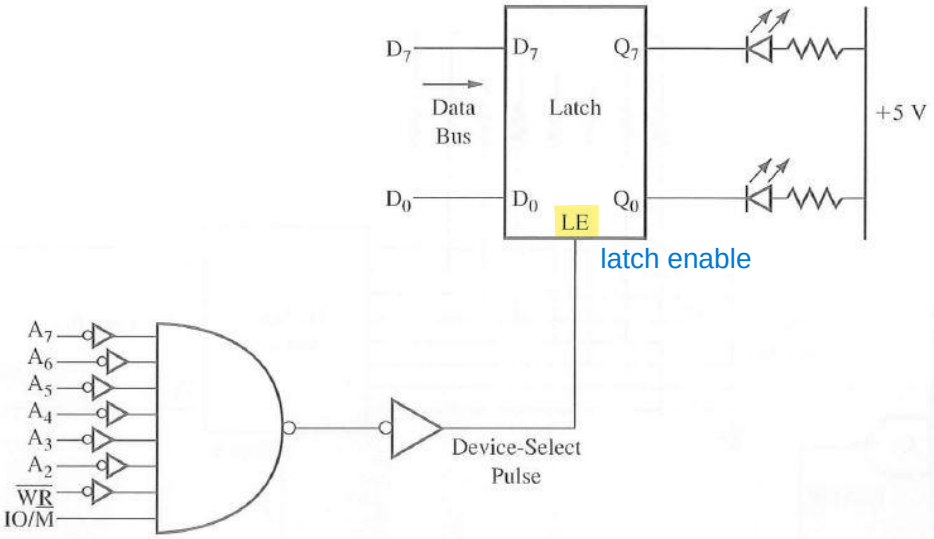


FIGURE 5.5
Partial Decoding: Output Latch with Multiple Addresses

9. Distinguish between absolute and partial address decoding.
[2014-1(b)]
Ans:

Absolute Address Decoding	Partial Address Decoding
All lines are decoded to generate one unique output pulse.	The output port can be selected by decoding some of the address lines.
The device has only one address.	The device has multiple addresses.
Can be used in any system.	Can be used in a small system where multiple addresses will not cause any problem.
Costly	Less Costly

10. Draw and discuss the timing-waveframe to execute the instruction OUT 01H.
[2018-3(a), 2017-3(b)]
Ans: Timing Waveframe to execute OUT 01H instruction:

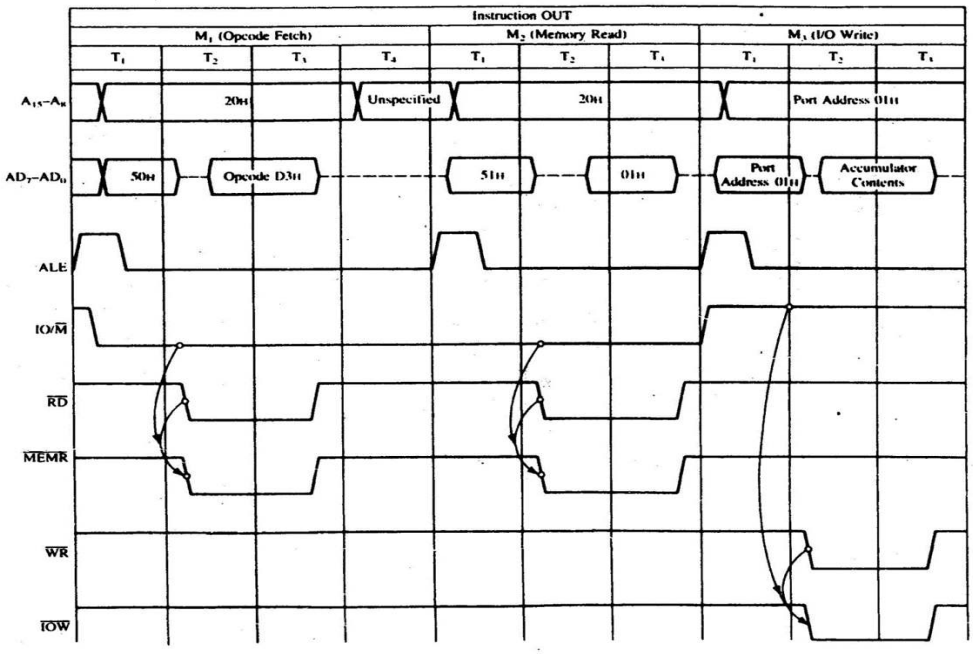


FIGURE 4.1
8085 Timing for Execution of OUT Instruction

OUT INSTRUCTION (8085)

In the first machine cycle, M_1 (Opcode Fetch, Figure 5.1), the 8085 places the high-order memory address 20H on A_{15}–A_8 and the low-order address 50H on AD_7–AD_0. At the same time, ALE goes high and IO/$\overline{M}$ goes low. The ALE signal indicates the availability of the address on AD_7–AD_0, and it can be used to demultiplex the bus. The IO/$\overline{M}$, being low, indicates that it is a memory-related operation. At T_2, the microprocessor sends the $\overline{RD}$ control signal, which is combined with IO/$\overline{M}$ (externally, see Chapter 4) to generate the $\overline{MEMR}$ signal, and the processor fetches the instruction code D3 using the data bus.

When the 8085 decodes the machine code D3, it finds out that the instruction is a 2-byte instruction and that it must read the second byte.

In the second machine cycle, M_2 (Memory Read), the 8085 places the next address, 2051H, on the address bus and gets the device address 01H via the data bus.

In the third machine cycle, M_3 (I/O Write), the 8085 places the device address 01H on the low-order (AD_7–AD_0) as well as the high-order (A_{15}–A_8) address bus. The IO/$\overline{M}$ signal goes high to indicate that it is an I/O operation. At T_2, the accumulator contents are placed on the data bus (AD_7–AD_0), followed by the control signal $\overline{WR}$. By ANDing the IO/$\overline{M}$ and $\overline{WR}$ signals, the $\overline{IOW}$ (see Figure 4.5) signal can be generated to enable an output device.

Figure 5.1 shows the execution timing of the OUT instruction. The information necessary for interfacing an output device is available during T_2 and T_3 of the M_3 cycle. The data byte to be displayed is on the data bus, the 8-bit device address is available on the low-order as well as high-order address bus, and availability of the data byte is indicated by the $\overline{WR}$ control signal. The availability of the device address on both segments of the address bus is redundant information; in peripheral I/O, only one segment of the address bus (low or high) is sufficient for interfacing. The data byte remains on the data bus only for two T-states, then the processor goes on to execute the next instruction. Therefore, the data byte must be latched now, before it is lost, using the device address and the control signal (Section 5.13).

8th Chapter
Counter and Time Delays

1. **Describe the delay calculation process using register pair.**
[2016-4(a)]
Ans:

8.1.2 Time Delay Using a Register Pair

The time delay can be considerably increased by setting a loop and using a register pair with a 16-bit number (maximum FFFFH). The 16-bit number is decremented by using the instruction DCX. However, the instruction DCX does not set the Zero flag

The following set of instructions uses a register pair to set up a time delay.

Label	Opcode	Operand	Comments	T-states
	LXI	B,2384H	;Load BC with 16-bit count	10
LOOP:	DCX	B	;Decrement (BC) by one	6
	MOV	A,C	;Place contents of C in A	4
	ORA	B	;OR (B) with (C) to set Zero flag	4
	JNZ	LOOP	;If result ≠ 0, jump back to LOOP	10/7

In this set of instructions, the instruction LXI B,2384H loads register B with the number 23H, and register C with the number 84H. The instruction DCX decrements the entire number by one (e.g., 2384H becomes 2383H). The next two instructions are used only to set the Zero flag; otherwise, they have no function in this problem. The OR instruction sets the Zero flag only when the contents of B and C are simultaneously zero. Therefore, the loop is repeated 2384H times, equal to the count set in the register pair.

TIME DELAY

The time delay in the loop is calculated as in the previous example. The loop includes four instructions: DCX, MOV, ORA, and JNZ, and takes 24 clock periods for execution. The loop is repeated 2384H times, which is converted to decimals as

$$2384H = 2 \times (16)^3 + 3 \times (16)^2 + 8 \times (16)^1 + 4(16^0)$$
$$= 9092_{10}$$

If the clock period of the system = 0.5 µs, the delay in the loop T_L is

$$T_L = (0.5 \times 24 \times 9092_{10})$$
$$\approx 109 \text{ ms (without adjusting for the last cycle)}$$
$$\text{Total Delay } T_D = 109 \text{ ms} + T_O$$
$$\approx 109 \text{ ms (The instruction LXI adds only 5 µs.)}$$

2. **Discuss how time delay is calculated using a loop within a loop technique with a counter value of 38FFH. (Do it also for FFFH).**
[2018-4(a), 2017-6(a), 2014-5(a)]
Ans:

8.1.3 Time Delay Using a Loop within a Loop Technique

A time delay similar to that of a register pair can also be achieved by using two loops; one loop inside the other loop, as shown in Figure 8.3(a). For example, register C is used in the inner loop (LOOP1) and register B is used for the outer loop (LOOP2). The following instructions can be used to implement the flowchart shown in Figure 8.3(a).

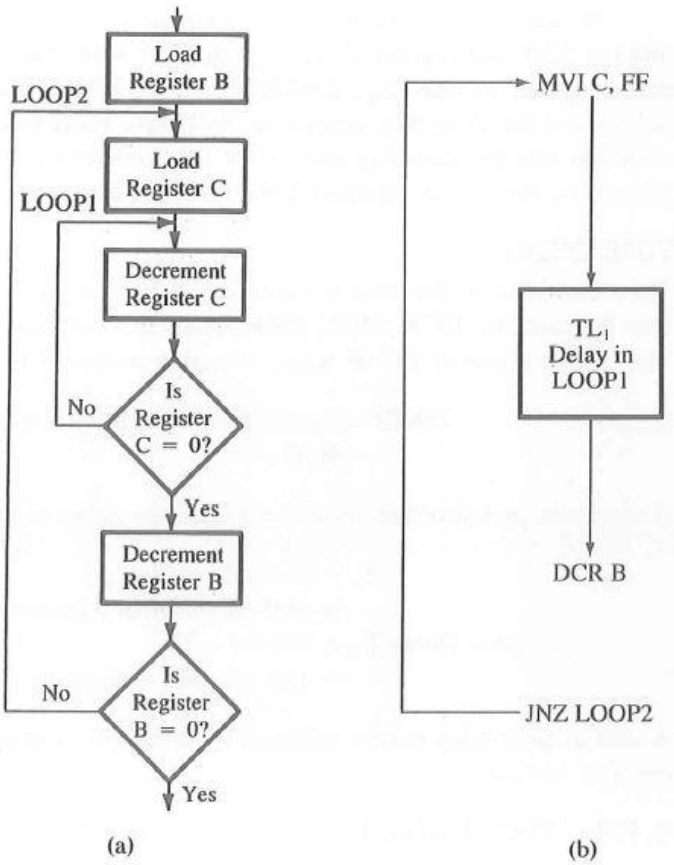
```
        MVI B,38H     7T
LOOP2:  MVI C,FFH     7T
LOOP1:  DCR C         4T
        JNZ LOOP1     10/7T
        DCR B         4T
        JNZ LOOP2     10/7T
```

DELAY CALCULATIONS

The delay in LOOP1 is $T_{L1} = 1783.5\ \mu s$. These calculations are shown in Section 8.1.1. We can replace LOOP1 by T_{L1}, as shown in Figure 8.3(b). Now we can calculate the delay in LOOP2 as if it is one loop; this loop is executed 56 times because of the count (38H) in register B:

$$\begin{aligned} T_{L2} &= 56(T_{L1} + 21\ \text{T-states} \times 0.5\ \mu s) \\ &= 56(1783.5\ \mu s + 10.5\ \mu s) \\ &= 100.46\ \text{ms} \end{aligned}$$

FIGURE 8.3
Flowchart for Time Delay with Two Loops



The total delay should include the execution time of the first instruction (MVI B,7T); however, the delay outside these loops is insignificant. The time delay can be increased considerably by using register pairs in the above example.

Similarly, the time delay within a loop can be increased by using instructions that will not affect the program except to increase the delay. For example, the instruction NOP (No Operation) can add four T-states in the delay loop. The desired time delay can be obtained by using any or all available registers.

12 Chapters
Interrupts

1. What happens when a microprocessor is interrupted?
[2018-2(c), 2015-4(a)]
Ans:
The 8085 interrupt process can be described in 8 steps.

Step 1: The interrupt process should be enabled by writing EI in the main program. The instruction EI sets the interrupted enabled flip flop. The instruction DI resets the flip flop and disables the interrupt process.

Step 2: When the microprocessor is executing the program, it checks the INTR line during the execution of each instruction.

Step 3: If the line INTR is high and the interrupt is enabled, the microprocessor completes the current instruction, disables the Interrupt Enable flip-flop, and sends a signal INTA – Interrupt Acknowledge (active low). The processor cannot accept any interrupt request until the interrupt flip-flop is enabled again.

Step 4: The signal INTA is used to insert a restart (RST) instruction (or a Call instruction) through external hardware. The RST instruction is a 1-byte call instruction that transfers the program control to a specific memory location.

Step 5: When the microprocessor receives an RST instruction, it saves the memory address of the next instruction on the stack. The program is transferred to the CALL location.

Step 6: Assuming that the task to be performed is written as a subroutine at the specified location, the processor performs the task. The subroutine is known as service routine.

Step 7: The service routine should include the instruction EI to enable the interrupt again.

Step 8: At the end of the subroutine, the RET instruction retrieves the memory address where the program was interrupted and continues the execution.

2. Write a main program to count continuously in binary with a one-second delay between each count and write service routine at XX70H to flash FFH five times when the program is interrupted, with some appropriate delay between each count.
(An Implementation of the 8085 interrupt)

[2019-4(c), 2018-4(c), 2016-7(b), 2015-5(b)]

Ans:

Main Program:

```
            LXI SP, XX99H       ; Initialize stack pointer
            EI                  ; Enable interrupt process
            MVI A,00H           ; Initialize counter
NXTCNT:     OUT PORT1           ; Display Count
            MVI C,01H           ; Parameter for 1-second Delay
            CALL Delay          ; Wait one second
            INR A               ; Next Count
            JMP NXTCNT          ; Continue
```

Delay Routine:

```
DELAY:      PUSH D              ; Save contents of DE
            PUSH PSW            ; Save contents of accumulator
SECOND:     LXI D, COUNT        ; Load register pair DE with a count for 1-second delay
LOOP:       DCX D               ; Decrement register pair DE
            MOV A,D
            ORA E               ; OR D and E to set Zero Flag
            JNZ LOOP            ; Jump to LOOP if delay count is not equal to 0
            DCR B               ; End of 1-second Delay. Decrement the counter
            JNZ SECOND          ; Is this the end of time needed? If not got back to
                                ; repeat 1 second delay
            POP PSW             ; Retrieved content of saved registers
            POP D
            RET                 ; Return to Main program
```

Service Routine:

```
SERV:       PUSH B              ; Save Contents
            PUSH PSW
            MVI B, 0AH          ; Load register B for five flashes and five blanks
            MVI A, 00H          ; Load 00 to blank display
FLASH:      OUT PORT1
            MVI C, 01H          ; Parameter for 1 second delay
            CALL DELAY
            CMA                 ; Complement Display Count
            DCR B               ; Reduce Count
            JNZ FLASH
            POP PSW
            POP B
            EI                  ; Enable Interrupt Process
            RET                 ; Return to Main program
```

DESCRIPTION OF THE INTERRUPT PROCESS

1. The main program initializes the stack pointer at XX99H and enables the interrupts. The program will count continuously from 00 to FFH, with a delay of one second between each count.
2. To interrupt the processor, push the switch. The INTR line goes high.
3. Assuming the switch is pushed when the processor is executing the instruction OUT at memory location XX06H, the following sequence of events occurs:*
   a. The microprocessor completes the execution of the instruction OUT.
   b. It senses that the line INTR is high, and that the interrupt is enabled.
   c. The microprocessor disables the interrupt, stops execution, and sends out a control signal $\overline{\text{INTA}}$ (Interrupt Acknowledge).
   d. The $\overline{\text{INTA}}$ (active low) enables the tri-state buffer and the instruction EFH is placed on the data bus.
   e. The microprocessor saves the address XX08H of the next instruction (MVI C,01H) on the stack at locations XX98H and XX97H, and the program is transferred to memory location 0028H. The locations 0028-29-2AH should have the following Jump instruction to transfer the program to the service routine.

   JMP XX70H

   (However, you do not have access to write at 0028H in the monitor program. See the next section.)
4. The program jumps to the service routine at XX70H.
5. The service routine saves the registers that are being used in the subroutine and loads the count ten in register B to output five flashes and also five blanks.
6. The service routine enables the interrupt before returning to the main program.
7. When the service routine executes the RET instruction, the microprocessor retrieves the memory address XX08H from the top of the stack and continues the binary counting.

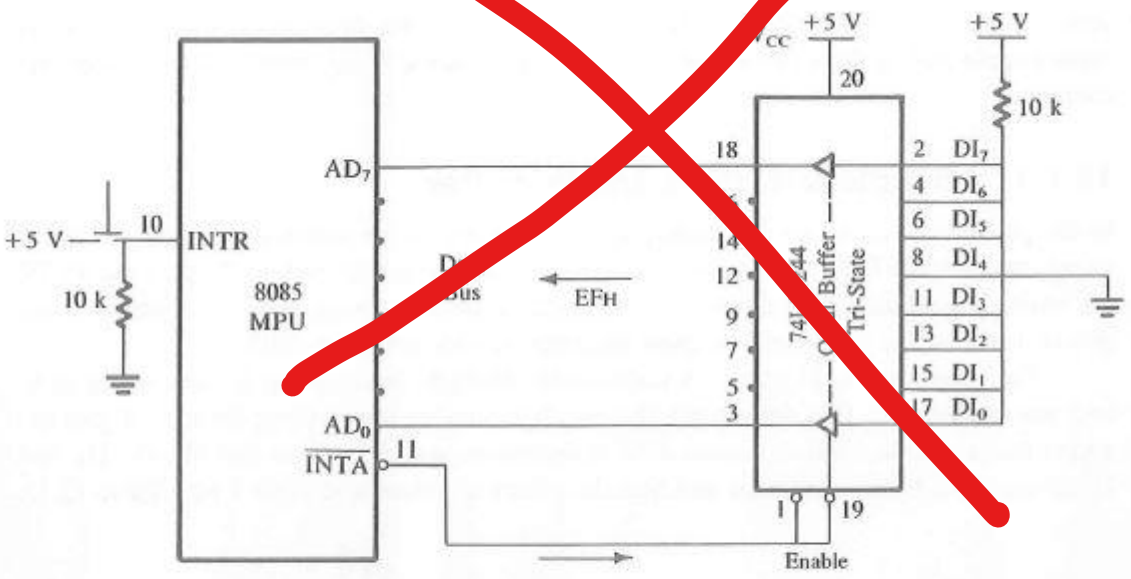


FIGURE 12.3
Schematic to Implement the 8085 Interrupt

3. **Is there a minimum pulse width required for the INTR signal?**
[2019-4(b)]
Ans: The microprocessor checks INTR, one clock period before the last T-states of an instruction cycle. In the 8085, the Call instructions require 18 T-states; therefore, the INTR pulse should be high at least for 17.5 T-states. In a system with a 3 MHz clock frequency (such as the SDK-85 system), the input pulse to INTR should be at least 5.8 μs long.

4. **Can the microprocessor be interrupted again before the completion of the first interrupt service routine?**
[2018-4(b), 2017-7(b)]
Ans:
The answer to this question is determined by the programmer. After the first interrupt, the interrupt process is automatically disabled. In the Illustrative program in section 12.12, the service routine enables the interrupt at the end of the service routine; in this case, the microprocessor cannot be interrupted before the completion of this routine. If the instruction EI is written at the beginning of the routine, the microprocessor can be interrupted again during the service routine.

5. **How long can the INTR pulse stay high?**
Ans:
The INTR pulse can remain high until the interrupt flip-flop is set by the EI instruction in the service routine. If it remains high after the execution of the EI instruction, the processor will be interrupted again, as if it were a new interrupt.

6. **Define maskable, nonmaskable, vectored, and non-vectored interrupts of 8085.**
Or, What do you mean by maskable and non maskable interrupts?
[2019-4(a), 2017-7(a), 2015-5(a), 2016-4(c)]
Ans:
Maskable: Maskable Interrupts are those which can be disabled or ignored by the microprocessor. These interrupts are either edge-triggered or level-triggered, so they can be disabled. INTR, RST 7.5, RST 6.5, RST 5.5 are maskable interrupts in the 8085 microprocessor.

Non-Maskable: Non-Maskable Interrupts are those which cannot be disabled or ignored by the microprocessor. TRAP is a non-maskable interrupt. It consists of both levels as well as edge triggering and is used in critical power failure conditions.

Vectored: Vectored Interrupts are those which have fixed vector addresses (starting address of sub-routine) and after executing these, program control is transferred to that address. The vectored interrupts of 8085 are TRAP, RST 5.5, RST 6.5, and RST 7.5
Vector Addresses are calculated by the formula 8 * TYPE

Non-Vectored: Non-Vectored Interrupts are those in which the vector address is not predefined. The interrupting device gives the address of the sub-routine for these interrupts. INTR is the only non-vectored interrupt in the 8085 microprocessor.

7. **Differentiate the interrupts: (i) maskable and non-maskable (ii) vectored and non-vectored.**

[2018-5(a)]

Ans:

Maskable	Non-Maskable
Can be disabled or ignored by the instructions of the CPU.	Can be disabled or ignored by the instructions of the CPU.
Maskable interrupts help to handle lower priority tasks.	Non-maskable interrupts help to handle higher priority tasks.
In maskable interrupts, response time is high.	In non-maskable interrupts, response time is low.
It may be vectored or non-vectored.	All are vectored interrupts.

	Vectored	Non-Vectored
Definition	Have a fixed vector address.	Vector address is not predefined.
Functionality	Jump to specific interrupt handler routine	Iterate through interrupt sources based on priority
Handling Time	Faster and more efficient handling	Potential delays in handling due to iteration
Prioritization	Directly identifies highest-priority interrupt	Relies on predefined priority scheme
Implementation Complexity	Requires complex hardware and software mechanisms	Relatively simpler implementation

8. **Describe 8085 Vectored interrupts.**

[2016-1(b)]

Ans:

The 8085 has five interrupt inputs.

One is called INTR, three are called RST 5.5, RST 6.5, and RST 7.5 respectively, and the fifth is called TRAP. These last four (RSTs and TRAP) are automatically vectored (transferred) to a specific location on memory page 00H without any external hardware. They do not require the INTA signal or an input port; the necessary hardware is already implemented inside the 8085. These interrupts and their call locations are as follows:

Interrupts	Call Locations
1. TRAP	0024H
2. RST 7.5	003CH
3. RST 6.5	0034H
4. RST 5.5	002CH

The TRAP has the highest priority, followed by RST 7.5, 6.5, 5.5, and INTR.

TRAP: TRAP, a nonmaskable interrupt known as NMI. It has the highest priority among the interrupt signals, it need not be enabled, and cannot be disabled. It is level-and

edge-sensitive, meaning that the input should go high to be acknowledged. It cannot be acknowledged again until it makes a transition from high to low to high.

RST 7.5, 6.5, 5.5: These maskable interrupts are enabled under program control with two instructions: EI (Enable Interrupt), and SIM (Set Interrupt Mask).

9. What are TRAP or NMI and RST 7.5? Mention the differences between RST 7.5 and RST 7.

[2019-5(b), 2015-5(c)]

Ans:

TRAP/NMI: TRAP, a nonmaskable interrupt known as NMI, is analogous to a smoke detector. It has the highest priority among the interrupt signals, it need not be enabled, and cannot be disabled. It is level-and edge-sensitive, meaning that the input should go high to be acknowledged. It cannot be acknowledged again until it makes the transition from high to low to high.

RST 7.5: This is a maskable interrupt that can be enabled under program control with two instructions EI and SIM. It is positive edge sensitive and can be triggered with a short pulse.

Difference between RST 7.5 and RST 7:

RST 7.5	RST 7
It is a hardware interrupt.	It is a software interrupt.
The vectored address is 3CH	The vectored address is 38H
This interrupt generates from an external device or hardware.	This interrupt generates by any internal system of the computer.
It does not increment the program counter.	It increments the program counter.
It has the lowest priority than RST 7	It has the highest priority than RST 7.5
It is an asynchronous event.	It is a synchronous event.

10. Why SIM and RIM instructions are used?

[2019-5(a)]

Ans:

Instruction SIM: Set Interrupt Mask. This is a 1–byte instruction and can be used for three different functions.

- One function is to set for RST 7.5, 6.5, and 5.5 interrupts. This instruction reads the content of the accumulator and enables or disables the interrupts according to the contents of the accumulator. Bit D3 is a control bit and should = 1 for bits D0, D1, and D2 to be effective. Logic 0 on D0, D1, and D2 will enable the corresponding interrupts, and logic 1 will disable the interrupts.
- The second function is to reset the RST 7.5 flip-flop (Figure 12.6). Bit D4 is additional control for RST 7.5. If D4=1, RST 7.5 is reset. This is used to override (or ignore) RST 7.5 without servicing it.
- The third is to implement serial I/O (Bits D7 and D6) and do not affect the interrupts

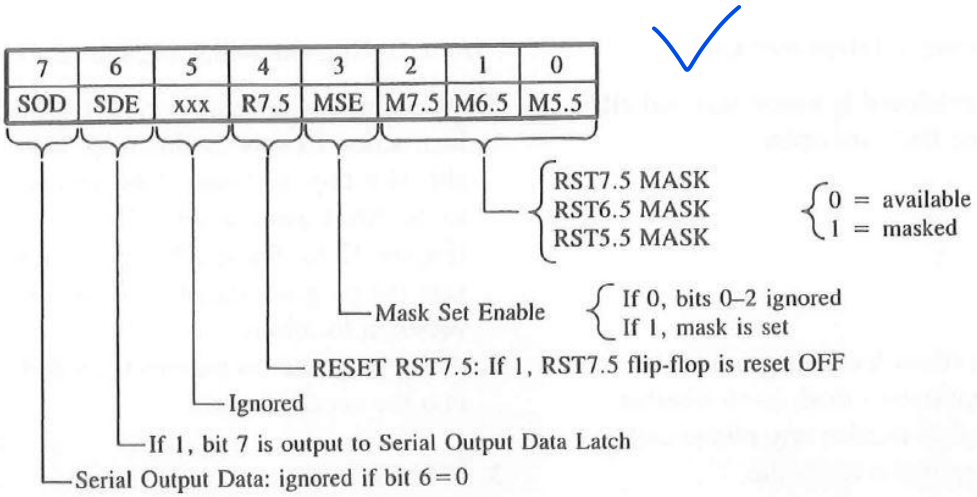


FIGURE 12.6
Interpretation of the Accumulator Bit Pattern for the SIM Instruction

Instruction RIM: Read Interrupt Mask. This is a 1–byte instruction that can be used for the following functions:

- To read interrupt masks, This instruction loads the accumulator with 8 bits indicating the current status of the interrupt masks.
- To identify pending interrupts. Bits D4, D5, and D6 identify the pending interrupts.
- To receive serial data bit D7 is used.

The RIM instruction loads the accumulator with the following information:

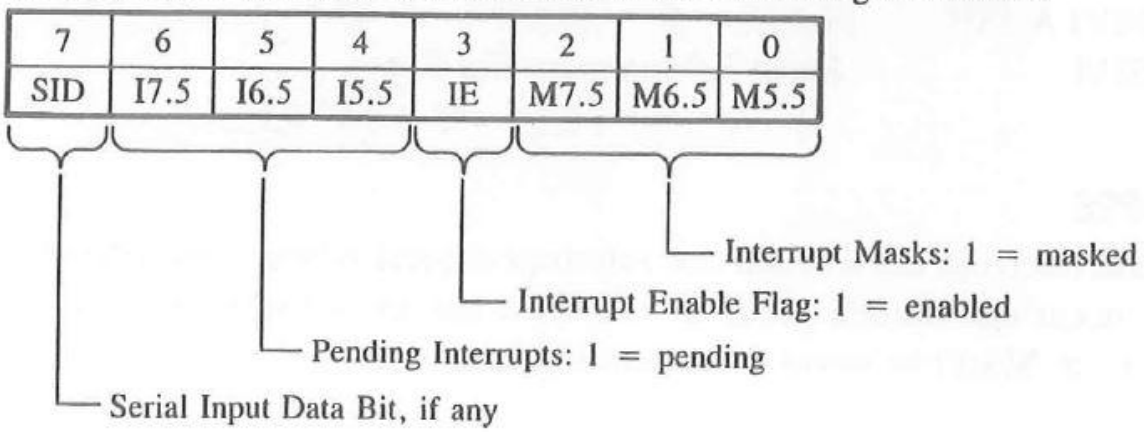


FIGURE 12.7
Interpretation of the Accumulator Bit Pattern for the RIM Instruction

11. Design a 1-minute timer using a 60 Hz power line as an interrupting source. The output ports should display minutes and seconds in BCD. At the end of the minute, the output ports should continue displaying one minute and zero seconds.
[2019-5(c), 2018-5(b), 2017-7(c)]

Ans:

Hardware Description:

This 1-minute timer is designed with a 60 Hz AC line. The circuit (Figure 12.8) uses a step-down transformer; the 74121 monostable multivibrators, and interrupt pin RST 6.5. After the interrupt, the program control is transferred to memory location 0034H in the monitor program.

The AC line with 60 Hz frequency has a period of 16.6 ms; that means it can provide a pulse every sixtieth of a second with 8.3 ms pulse width, which is too long for the interrupt. The interrupt flip-flop is enabled again before 6 μs in the service routine, therefore the pulse should be turned off before the EI instruction in the service routine is executed.

Monitor Program:

```
            JMP RWM          ; This is RST 6.5; go to location in user memory to
                             ; give Restart access to the user.
```

Main Program:

```
            LXI SP, STACK    ; Initialize stack pointer
            RIM              ; Read Mask
            ORI 08H          ; Bit pattern to enable RST 6.5
            SIM              ; Enable RST 6.5
            LXI B, 0000H     ; Set up B for minutes and C for seconds

            MVI D, 3CH       ; Set up D to count 60(10) interrupts
            EI               ; Enable Interrupts
DISPLAY:    MOV A, B
            OUT PORT1        ; Display Minitues at PORT1
            MOV A, C
            OUT PORT2        ; Display seconds at PORT2
            JMP DISPLAY

RWM:        JMP TIMER        ; This is RST 6.5 vector location 0034H; go to TIMER
                             ; routine to upgrade the clock
```

Interrupt Service Routine:

```
; Section I
TIMER:      DCR D            ;One interrupt occurred; reduce count by 1
            EI               ; Enable interrupts
            RNZ              ; Has 1 second elapsed? If not, return
;Section II
            DI               ; No other interrupts allowed
            MVI D,3CH        ;1 second is complete; load D to count 60 interrupts
            MOV A,C
            ADD 01H          ; increment second register
            DAA              ; Decimal-adjust “Seconds”
            MOV C,A          ;Save “BCD” seconds
            CPI 60H
            EI
            RNZ              ; Is time=60 seconds? If not return
```

```
;Section III
        DI              ; Disable interrupts
        MVI C,00H       ;60 seconds complete; clear "Seconds" register
        INR B           ; Increment "Minutes" register
        RET             ; 1 minute elapsed
```

PROGRAM DESCRIPTION

The main program clears registers B and C to store minutes and seconds, respectively; enables the interrupts; sets up register D to count 60 interrupts; and displays the starting time in minutes (00) and seconds (00). Instruction SIM enables RST 6.5 according to the bit pattern in the accumulator.

When the first pulse interrupts the processor, program control is transferred to memory location 0034H, as mentioned earlier. (Check location 0034H in your monitor program; you may find a Jump instruction to transfer the control to a memory location in R/W memory. Write a Jump instruction at that location to locate the service routine labeled TIMER.)

In the service routine (Section I), register D is decremented every second, the interrupt is enabled, and the program is returned to the main routine. This is repeated 60 times. After the sixtieth interrupt, counter D goes to zero and the program enters Section II. In this section, counter D is reloaded, the "second" register is incremented and adjusted for BCD, and the program is returned to the main routine. In this section, instruction DI is used as a precaution to avoid any interrupts from other sources. For the next 60 interrupts, the program remains in Section I. When Section II is repeated 60 times, the program goes to Section III, where the "minute" register is incremented and the "second" register is cleared. To avoid further interrupts, the interrupt is disabled and the program is returned to the main routine where one minute and zero seconds are displayed continuously.

In this particular program, the service routine does not save any register contents by using PUSH instructions before starting the service routine. However, in most service routines, register contents must be saved because the interrupt is asynchronous and can occur at any time.

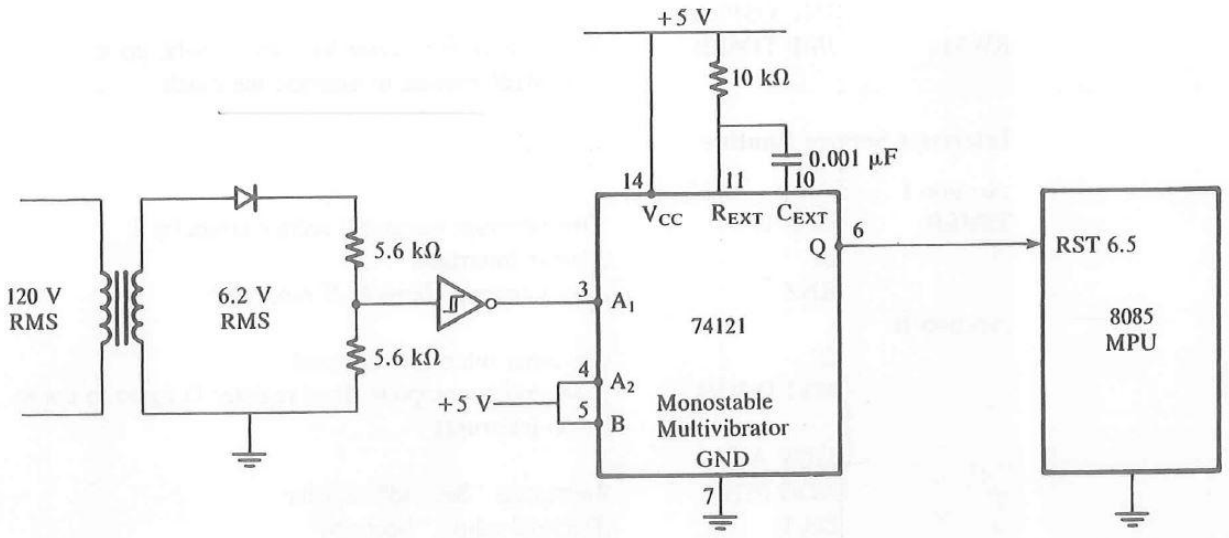


FIGURE 12.8
Schematic of Interrupt-Driven Timer Clock

13th Chapter
Interfacing Data Converters

1. **Design an output port with the address FFH to interface the 1408 D/A converter that is calibrated for a 0 to 10 V range. Write a program to generate a continuous ramp waveform. Explain the operation of 1408 for the calibration of a bipolar range ±5V. Calculate the output VO if the input is $(10000000)_2$.**

[2018-6(b), 2017-8(b), 2016-5(a)]

Ans:

Hardware Description:

The total reference current source is determined by the resistor R_{14} and the voltage V_{Ref}. The resistor R_{15} is generally equal to R_{14} to match the input impedance of the reference source. The output I_O is calculated as follows:

$$I_O = \frac{V_{Ref}}{R_{14}}\left(\frac{A_1}{2} + \frac{A_2}{4} + \frac{A_3}{8} + \frac{A_4}{16} + \frac{A_5}{32} + \frac{A_6}{64} + \frac{A_7}{128} + \frac{A_8}{256}\right)$$

where inputs A_1 through $A_8 = 0$ or 1.

This formula is an application of the generalized formula for the current I_O. For full-scale input (D_7 through $D_0 = 1$),

$$I_O = \frac{5\text{ V}}{2.5\text{ k}}\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256}\right)$$
$$= 2\text{ mA}\ (255/256)$$
$$= 1.992\text{ mA}$$

The output is 1 LSB less than the full-scale reference source of 2 mA. The output voltage V_O for the full-scale input is

$$V_O = 2\text{ mA}\ (255/256) \times 5\text{ k}$$
$$= 9.961\text{ V}$$

V0 = I0 + Rf

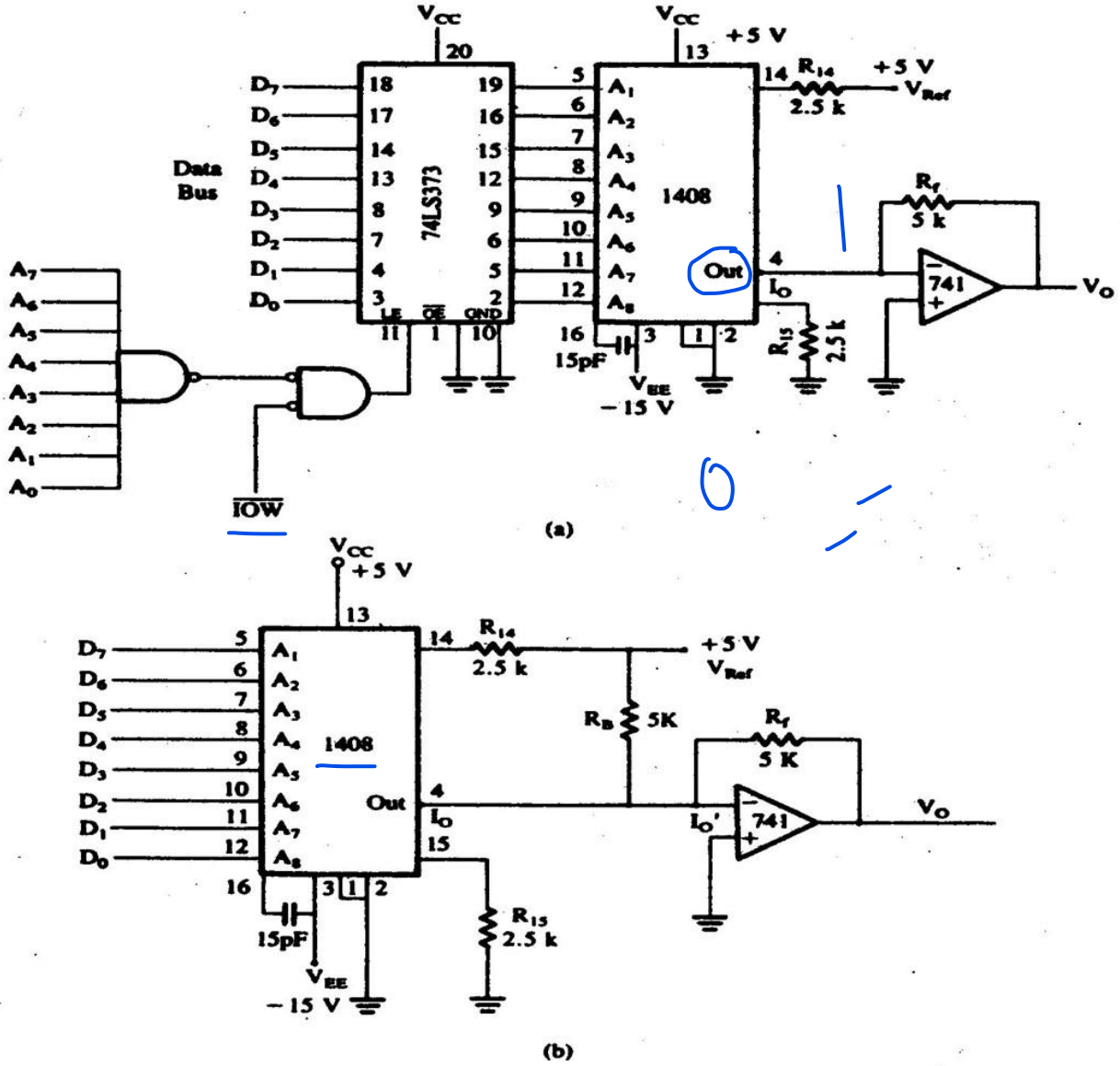


FIGURE 13.5
Interfacing the 1408 D/A Converter: Voltage Output in Unipolar Range (a) and in Bipolar Range (b)

PROGRAM

To generate a continuous waveform, the instructions are as follows:

```
        MVI A,00H       ;Load accumulator with the first input
DTOA:   OUT FFH         ;Output to DAC
        MVI B,COUNT     ;Set up register B for delay
DELAY:  DCR B
        JNZ DELAY
        INR A           ;Next input
        JMP DTOA        ;Go back to output
```

Program Description This program outputs 00 to FF continuously to the D/A converter. The analog output of the DAC starts at 0 and increases up to 10 V (approximately) as a ramp. When the accumulator contents go to 0, the next cycle begins; thus the ramp signal is generated continuously. The ramp output of the DAC can be observed on an oscilloscope with an external sync.

The delay in the program is necessary for two reasons:

1. The time needed for a microprocessor to execute an output loop is likely to be less than the settling time of the DAC.
2. The slope of the ramp can be varied by changing the delay.

OPERATING THE D/A CONVERTER IN A BIPOLAR RANGE

The 1408 in Figure 13.5(b) is calibrated for the bipolar range from −5 V to +5 V by adding the resistor R_B (5.0 k) between the reference voltage V_{Ref} and the output pin 4. The resistor R_B supplies 1 mA (V_{Ref}/R_B) current to the output in the opposite direction of the current generated by the input signal. Therefore, the output current for the bipolar operation I_O' is

$$I_O' = I_O - \frac{V_{Ref}}{R_B}$$

$$= \frac{V_{Ref}}{R_{14}}\left(\frac{A_1}{2} + \frac{A_2}{4} + \frac{A_3}{8} + \frac{A_4}{16} + \frac{A_5}{32} + \frac{A_6}{64} + \frac{A_7}{128} + \frac{A_8}{256}\right) - \frac{V_{Ref}}{R_B}$$

When the input signal is equal to zero, the output V_O is

$$V_O = I_O'R_f$$

$$= \left(I_O - \frac{V_{Ref}}{R_B}\right)R_f$$

$$= \left(0 - \frac{5\text{ V}}{5\text{ k}}\right)(5\text{ k}) \qquad (I_O = 0 \text{ for input} = 0)$$

$$= -5\text{ V}$$

When the input = 1000 0000, output V_O is

$$V_O = \left(I_O - \frac{V_{Ref}}{R_B}\right)R_f$$
$$= \left(\frac{V_{Ref}}{R_{14}} \times \frac{A_1}{2} - \frac{V_{Ref}}{R_B}\right)R_f \qquad (A_2 - A_8 = 0)$$
$$= \left(\frac{5\text{ V}}{2.5\text{ k}} \times \frac{1}{2} - \frac{5\text{ V}}{5\text{ k}}\right)5\text{ k}$$
$$= (1\text{ mA} - 1\text{ mA})5\text{ k} = 0\text{ V}$$

2. Discuss about Successive-Approximation A/D converter.

[2017-8(c), 2016-5(b), 2014-5(b)]

Ans: Figure 13.9(a) shows the block diagram of a successive approximation A/D converter that includes three major elements: the D/A converter, the successive approximation register (SAR), and the comparator. The conversion technique involves the output of the D/A converter Vo with the analog input signal Vin. The digital input to the DAC is generated using the successive approximation method (explain below). When the DAC output matches the analog signal, the input to the DAC is an equivalent digital signal.

The successive-approximation method of generating input to the DAC is similar to weighing an unknown material (e.g., less than 1 gram) on a chemical balance with a set of such fractional weights as ½ g, ¼ g, ⅛ g, etc. The weighing procedure begins with the heaviest weight (½ g), and subsequent weights (in decreasing order) are added until the balance is tipped. The weight that tips the balance is removed, and the process is continued until the smallest weight is used. In the case of a 4-bit A/D converter, bit D_3 is turned on first and the output of the DAC is compared with an analog signal. If the comparator changes the state, indicating that the output generated by D_3 is larger than the analog signal, bit D_3 is turned off in the SAR and bit D_2 is turned on. The process continues until the input reaches bit D_0.

Figure 13.9(b) illustrates a 4-bit conversion process. When bit D3 is turned on, the output exceeds the analog signal, and therefore, D3 is turned off. When the next three successive bits are turned on, the output becomes approximately equal to the analog signal.

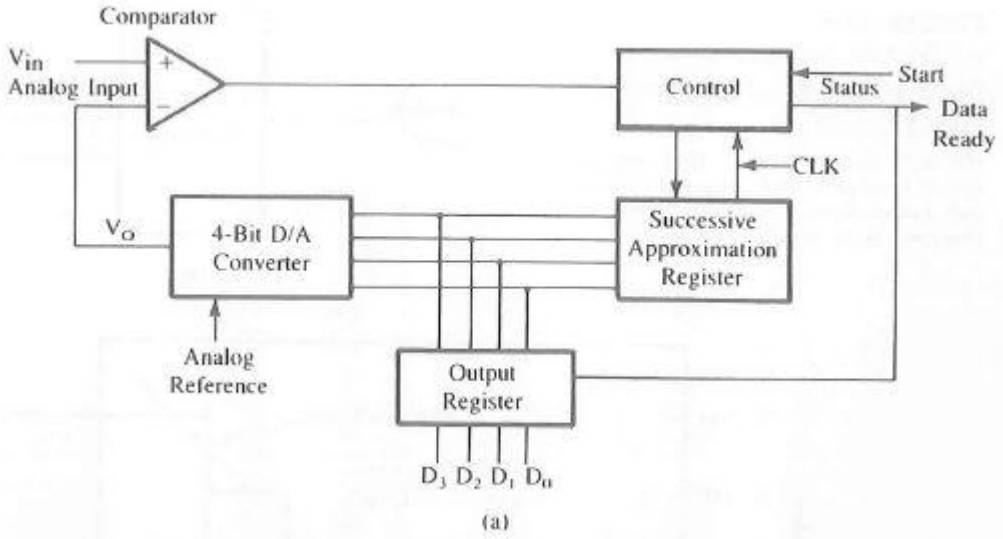
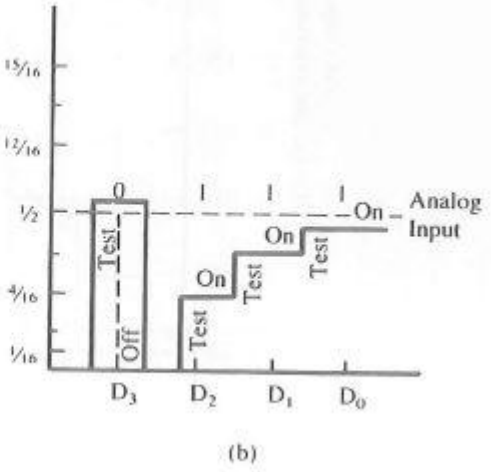


FIGURE 13.9
Successive-Approximation A/D Converter: Block Diagram (a) and Conversion Process for a 4-Bit Converter (b)



3. Discuss how an 8-bit A/D converter is interfaced with the processor using the status check I/O technique.
[2019-7(c), 2018-6(a)]
Ans:

INTERFACING AN 8-BIT A/D CONVERTER USING STATUS CHECK

Figure 13.11 shows a schematic of interfacing a typical A/D converter using status check. The A/D converter has one input line for analog signal and eight output lines for converted digital signals. Typically, the analog signal can range from 0 to 10 V, or ±5 V. In addition, the converter shows two lines START and DR (Data Ready), both active low (the active logic level of these lines can be either low or high depending upon the design of a particular converter). When an active low pulse is sent to the START pin, the DA

goes high and the output lines go into the high impedance state. The rising edge of the START pulse initiates the conversion. When the conversion is complete, the DR goes low, and the data are made available on the output lines that can be read by the microprocessor. To interface this converter, we need one output port to send a START pulse and two input ports: one to check the status of the DR line and the other to read the output of the converter.

In Figure 13.11, the address decoding is performed by using the 3-to-8 decoder (74LS138), the 4-input NAND gate, and inverters. Three output lines of the decoder are combined with appropriate control signals (IOW and IOR) to assign three port addresses from 80H to 82H. The output port 82H is used to send a START pulse by writing the OUT instruction; in this case, we are interested in getting a pulse from the microprocessor, and the contents of the accumulator are irrelevant to start the conversion. However, for some converters the IOW pulse from the microprocessor may not be long enough to start the conversion. When the conversion begins, the DR (Data Ready) goes high and stays high until the conversion is completed. The status of the DR line is monitored by connecting the line to bit D_0 of the data bus through a tri-state buffer with the input port address 80H. The processor will continue to read port 80H until bit D_0 goes low. When the DR goes active, the data are available on the output lines of the converter, and the processor can access that data by reading the input port 81H. The subroutine instructions, to initiate the conversion and to read output data, and the flowchart are shown in Figure 13.12.

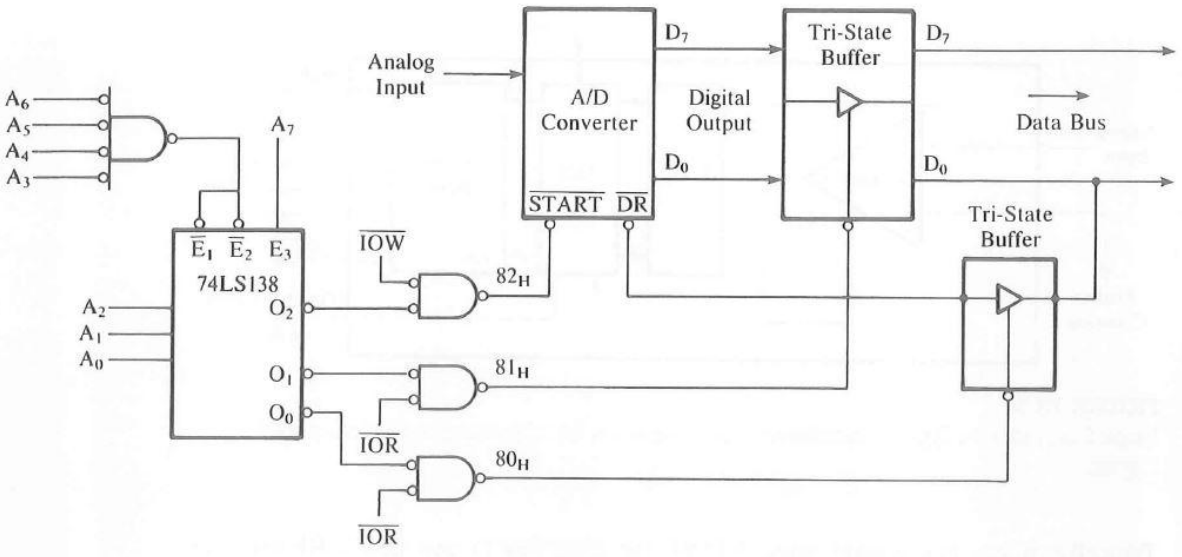
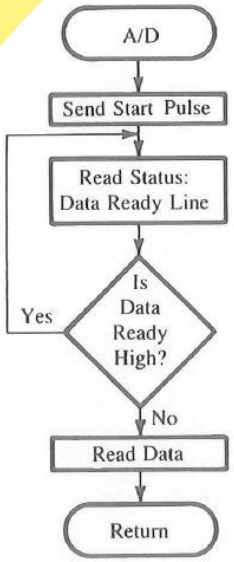


FIGURE 13.11
Interfacing an A/D Converter Using the Status Check




```
        OUT 82H   ;Start conversion
TEST:   IN 80H    ;Read Data Ready Status
        RAR       ;Rotate D0 into carry
        JC TEST   ;If D0 = 1, conversion is
                  ;  not yet complete; go
                  ;  back and check
        IN 81H    ;Read output and save it in
                  ;  accumulator
        RET
```

FIGURE 13.12
Flowchart of A/D Conversion Process

14th Chapter
Programmable Interface Device

1. What is meant by PPI?
Or, What is programmable device?

[2019-7(a), 2018-7(a)]

Ans: 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 a control register.

2. What do you know about the I/O Ports and Timer 8155?

[2019-6(a), 2016-6(a), 2014-6(a)]

Ans: The 8155 is a device with two sections: the first is a 256 bytes R/W memory, and the second is a programmable I/O. Functionally, these two sections can be viewed as two independent chips.

I/O Ports:

The I/O section includes two 8-bit parallel I/O ports (A and B), one 6-bit port (C), and a timer (Figure 14.5). All the ports can be configured simply as input/output ports. Ports A and B also can be programmed in the handshake mode, each port using three signals from port C. The timer is a 14-bit down-counter and has four modes.

The I/O section of the 8155 includes a control register, three I/O ports, and two registers for the timer. (Figure 14.6).

To communicate with the peripherals through the 8155 the following steps are necessary:

1. Determine the address (port numbers of the registers and I/Os) based on the Chip Enable logic and address lines AD0, AD1, and AD2.

2. Write control word in the control register to specify I/O functions of the ports

and the timer characteristics.

3. Write I/O instructions to port addresses to communicate with peripherals.

4. Read the status register, if necessary, to verify the status of the I/O ports and the timer. In simple applications, this step is not necessary.

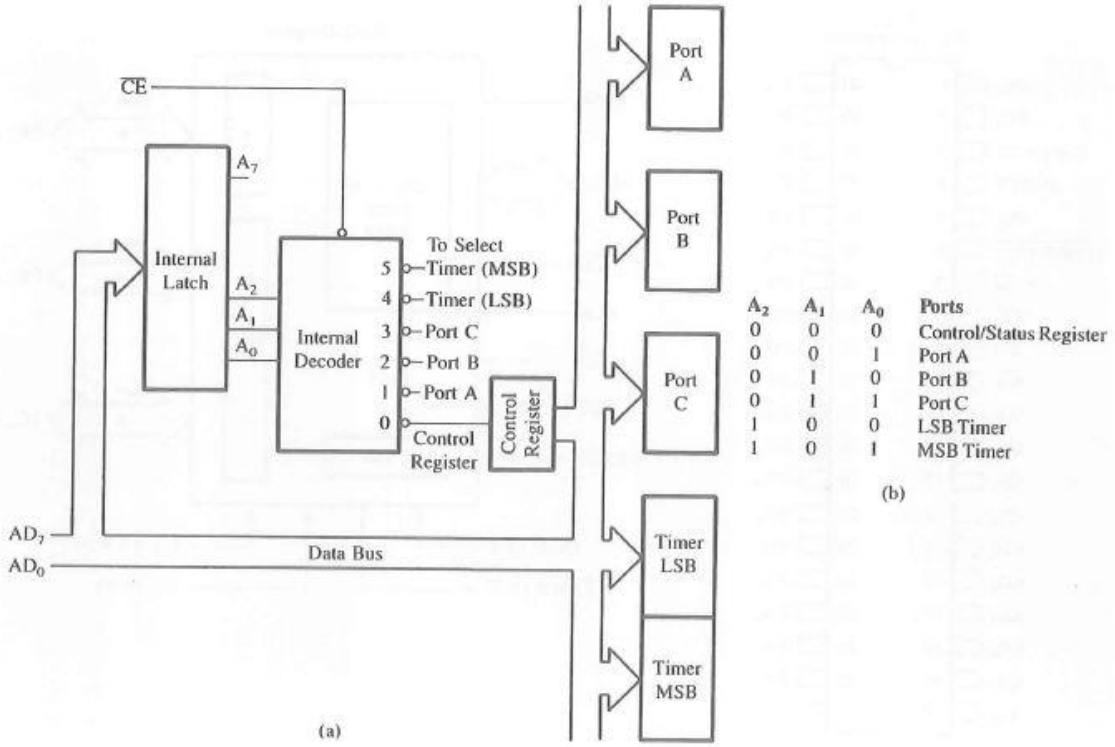


FIGURE 14.6
Expanded Block Diagram of the 8155 (a) and Its I/O Address: Selection (b)

Timer 8155:

14.2.3 The 8155 Timer

The timer section of the 8155 has two 8-bit registers; 14 bits are used for the counter, two bits for the timer mode, and it requires a clock as an input. This 14-bit down-counter provides output in four different modes, as described below.

Figure 14.10(a) shows two registers for a 14-bit count, one for LSB (low significant byte) and one for MSB (most significant byte). The most significant bits M_2 and M_1 are used to specify the timer mode. To operate the timer, a 14-bit count and mode bits are loaded in the registers. An appropriate control word starts the counter, which decrements the count by two at each clock pulse. The timer outputs vary according to the mode specified; see Figure 14.10(b).

The timer can be stopped either in the midst of counting or at the end of a count (applicable to Modes 1 and 3). In addition, the actual count at a given moment can be obtained by reading the status register. These details will be described later.

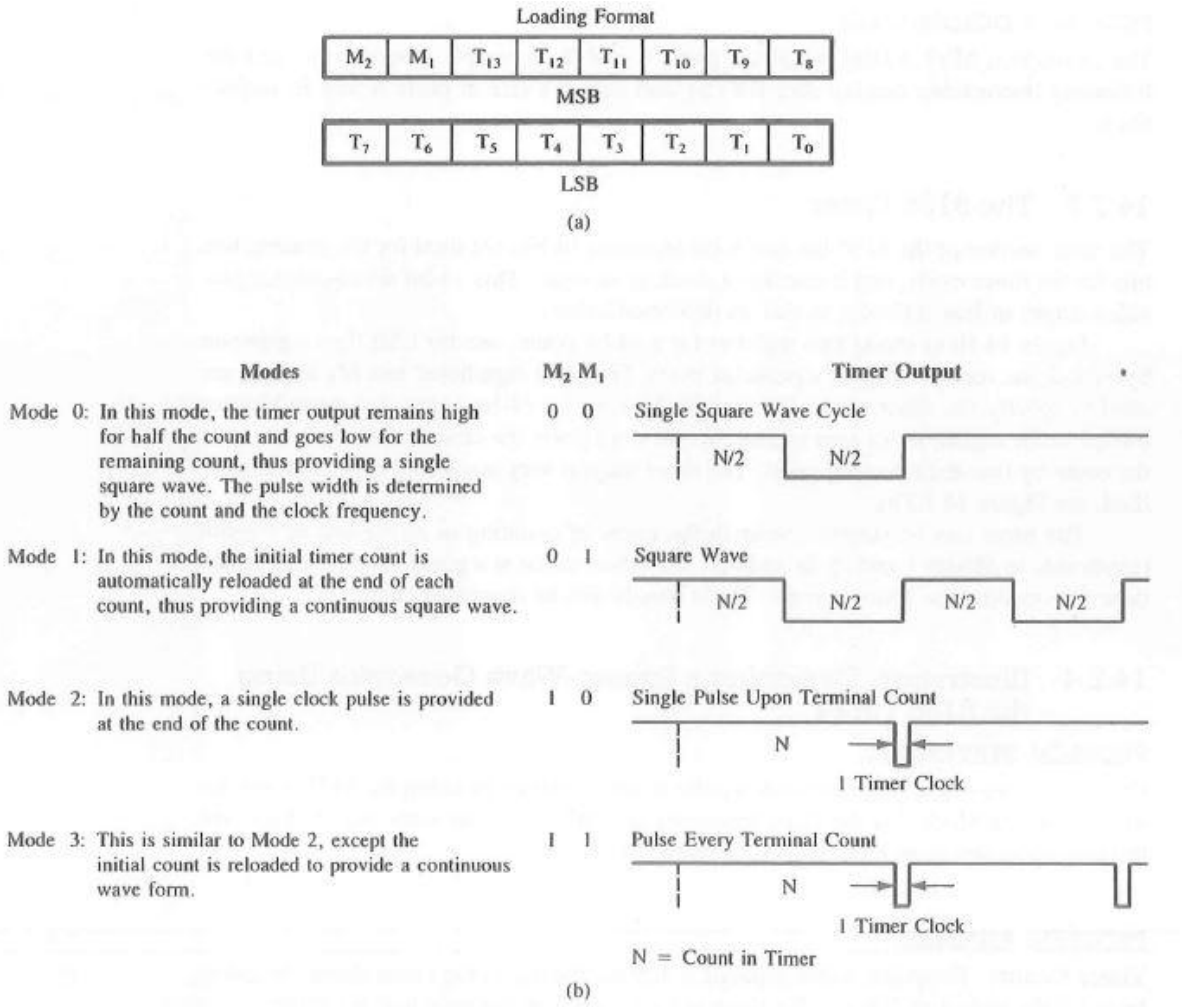


FIGURE 14.10
Timer Loading Format (a) and Modes (b)

Discuss the control word format of 8155.

[2019-7(b), 2018-7(b), 2915-7(a)]

Ans: The I/O ports and the timer can be configured by writing a control word in the control register. The control register bits are defined as shown in Figure 14.8.

Bit D2 and D3 determine the functions of port C; their combination specifies one of the four alternatives, from simple I/O to interrupt I/O, as shown in Figure 14.8(b). Bits D4 and D5 are used only in the interrupt mode to enable or disable internal flip-flops of the 8155. These bits do not have any effect on the Internal Enable (INTE) flip-flop of the MPU.

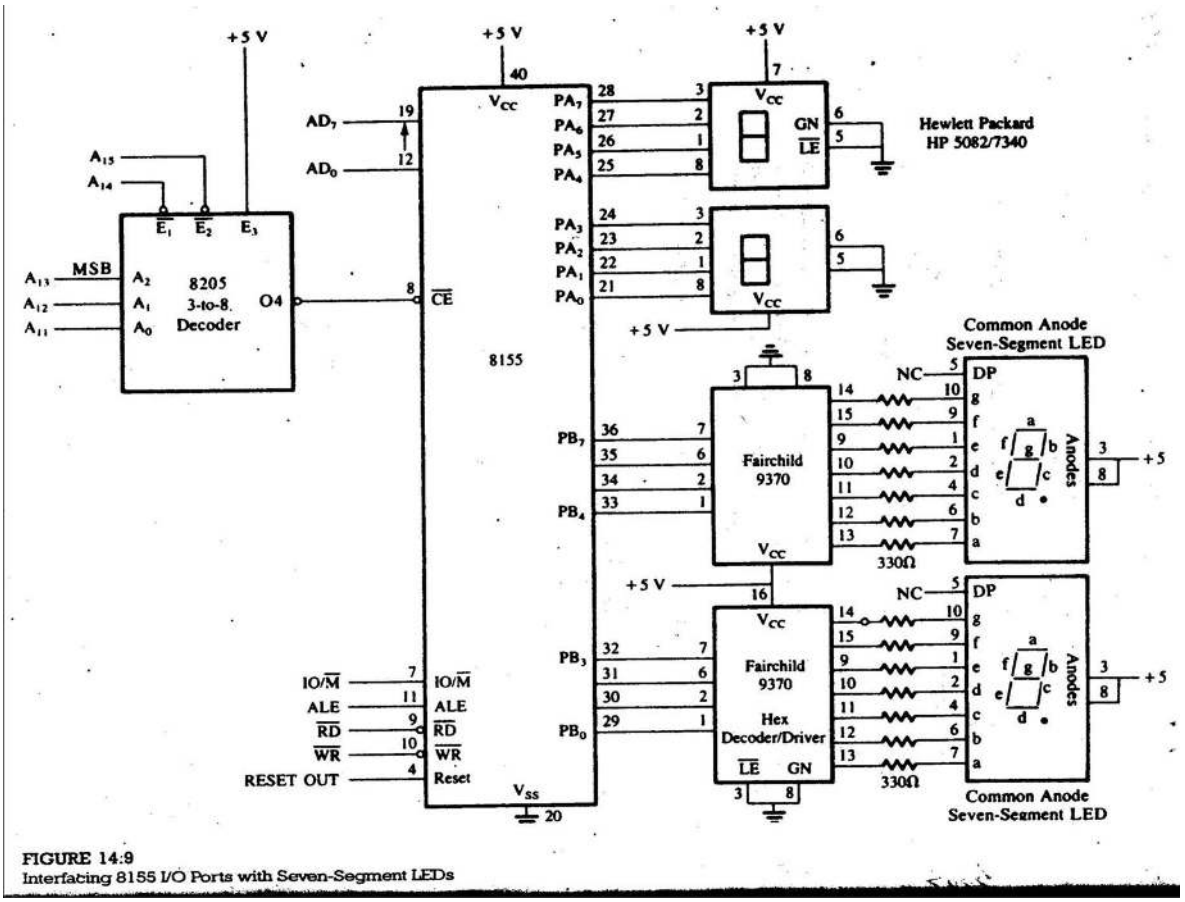


FIGURE 14:9
Interfacing 8155 I/O Ports with Seven-Segment LEDs

4. Handshake signals are used to interface I/O devices with processors. Can you explain Why and How these signals are used?

[2019-2(a), 2017-4(b)]

Ans:

The MPU and peripherals operate at different speeds; therefore, signals are exchanged prior to data transfer between the fast-responding MPU and slow-responding peripherals such as printers and data converters. These signals are called **handshake signals.** The exchange of handshake signals prevents the MPU from writing over the previous data before a peripheral has had a chance to accept it or from reading the same data before a peripheral has had time to send the next data byte. These signals are generally provided by programmable devices. Figure 14.4(a) shows a programmable device in the input mode, with two handshake signals (STB and IBF) and one interrupt signal (INTR). Now the MPU has two ways of finding out whether a peripheral is ready: either by checking the status of a handshake signal or through the interrupt technique, as explained below.

DATA INPUT WITH HANDSHAKE

The steps in data input from a peripheral such as a keyboard are as follows:

1. A peripheral strobes or places a data byte in the input port and informs the interfacing device by sending handshake signal STB (Strobe).
2. The device informs the peripheral that its input port is full—do not send the next byte until this one has been read. This message is conveyed to the peripheral by sending handshake signal IBF (Input Buffer Full).
3. The MPU keeps checking the status until a byte is available. Or the interfacing device informs the MPU, by sending an interrupt, that it has a byte to be read.
4. The MPU reads the byte by sending control signal $\overline{RD}$.

DATA OUTPUT WITH HANDSHAKE

Figure 14.4(b) shows the programmable device in the output mode using the same two handshake signals, except that they are labeled differently. The steps in data output to a peripheral such as a printer are as follows:

1. The MPU writes a byte into the output port of the programmable device by sending control signal $\overline{WR}$.
2. The device informs the peripheral, by sending handshake signal OBF (Output Buffer Full), that a byte is on the way.
3. The peripheral acknowledges the byte by sending back the ACK (Acknowledge) signal to the device.
4. The device interrupts the MPU to ask for the next byte, or the MPU finds out that the byte has been acknowledged through the status check.

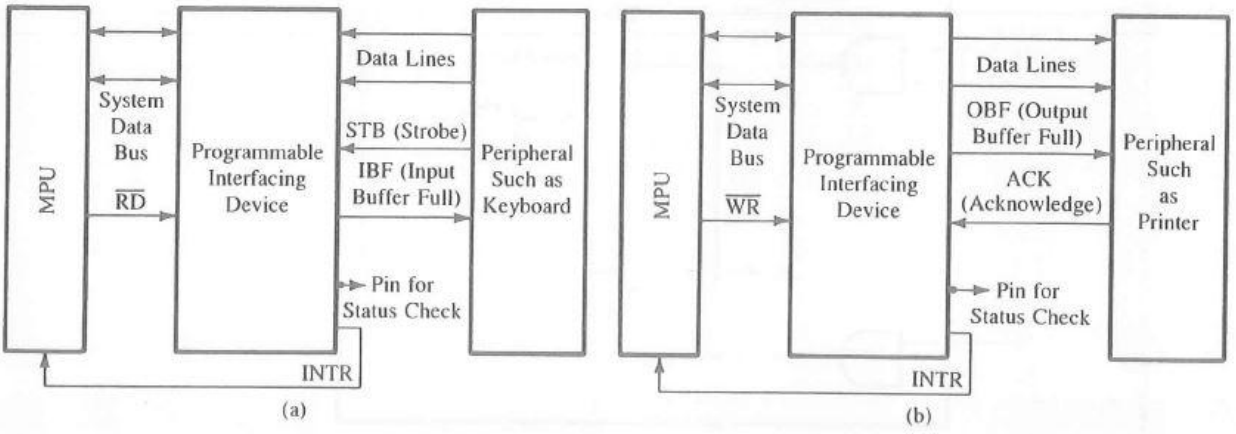


FIGURE 14.4
Interfacing Device with Handshake Signals for Data Input (a) and Data Output (b)

5. Design a square-wave generator with a pulse width of 100 µs by using the 8155 timer. Set up timer in Mode 1 if the clock frequency is 3 MHz.
[2019-6(b), 2017-6(b), 2016-6(b), 2015-7(b), 2014-6(b)]
Ans:

Timer Count The pulse width required is 100 μs; therefore, the count should be calculated for the period of 200 μs. The timer output stays high for only half the count.

$$\text{Clock Period} = 1/f = 1/3 \times 10^6 = 330 \text{ ns}$$

$$\text{Timer Count} = \frac{\text{Pulse Period}}{\text{Clock Period}} = \frac{200 \times 10^{-6}}{330 \times 10^{-9}} = 606$$

$$\text{Count} = 025\text{EH}$$

Assuming the same decode logic for the 8155 Chip Enable line as in Example 14.2, the port addresses for the timer registers are

Timer LSB = 24H
Timer MSB = 25H

The least significant byte, 5EH (of the count 025EH), should be loaded in the timer register with address 24H. The most significant byte is determined as follows:

M_2	M_1	T_{13}	T_{12}	T_{11}	T_{10}	T_9	T_8	
0	1	0	0	0	0	1	0	= 42H
↓		↓						
Timer Mode 1		MSB						

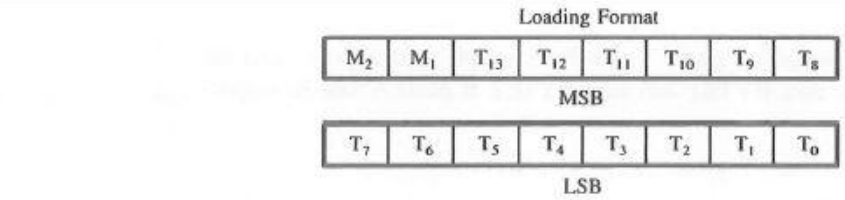
Therefore, 42H should be loaded in the timer register with the address 25H.

Control Word Assuming the same configuration for ports A and B as before, only bits D_7 and D_6 should be set to 1 to start the counter (see control word definition in Figure 14.8).

Therefore, Control Word: 1100 0011 = C3H

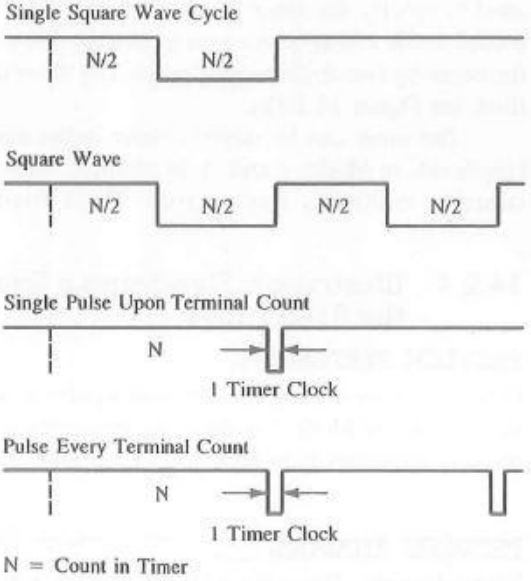
Initialization Instructions

```
MVI A,5EH     ;LSB of the count
OUT 24H       ;Load the LSB timer register
MVI A,42H     ;MSB of the count
OUT 25H       ;Load the MSB timer register
MVI A,C3H
OUT 20H       ;Start the timer
HLT
```



(a)

Modes	M_2 M_1	Timer Output
Mode 0: In this mode, the timer output remains high for half the count and goes low for the remaining count, thus providing a single square wave. The pulse width is determined by the count and the clock frequency.	0 0	Single Square Wave Cycle
Mode 1: In this mode, the initial timer count is automatically reloaded at the end of each count, thus providing a continuous square wave.	0 1	Square Wave
Mode 2: In this mode, a single clock pulse is provided at the end of the count.	1 0	Single Pulse Upon Terminal Count
Mode 3: This is similar to Mode 2, except the initial count is reloaded to provide a continuous wave form.	1 1	Pulse Every Terminal Count



(b)

FIGURE 14.10
Timer Loading Format (a) and Modes (b)