Chapter-12

Interrupts

12.1 THE 8085 INTERRUPTS

The 8085 interrupt process is controlled by the Interrupt Enable flip-flop, which is internal to the processor and can be set or reset by using software instructions. If the flip-flop is enabled and the input to the interrupt signal INTR (pin 10) goes high, the microprocessor is interrupted. This is a maskable interrupt and can be disabled. The 8085 a nonmaskable and three additional vectored interrupt signal as well.

The 8085 interrupt process can be described in terms of eight steps:

Step 1: The interrupt process should be enabled by writing the instruction EI in the main program.

Instruction EI (Enable Interrupt)

- ❑ This is a 1-byte instruction.
- ❑ The instruction sets the Interrupt Enable flip-flop and enables the interrupt process.
- ❑ System reset or an interrupt disables the interrupt process.

Instruction DI (Disable Interrupt)

- This is a 1-byte instruction.
- The instruction resets the Interrupt Enable flip-flop and disables the interrupt process.
- It should be included in a program segment where an interrupt from an outside source cannot be tolerated.

Step 2: When the microprocessor is executing a 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 $\overline{\text{INTA}}$ – Interrupt Acknowledge (active low). The processor cannot accept any interrupt request until the interrupt flip-flop is enabled again.

Step 4: The signal $\overline{\text{INTA}}$ 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.

12.11 RST (Restart) Instructions

These are 1-byte Call instructions that transfer the program execution to a specific location. The RST instructions are executed in a similar way to that of Call instructions as listed in Table 12.1.

TABLE 12.1

Restart Instructions

	Binary Code									
Mnemonics	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Hex Code	Call Locations
RST 0	1	1	0	0	0	1	1	1	C7	0000
RST 1	1	1	0	0	1	1	1	1	CF	0008
RST 2	1	1	0	1	0	1	1	1	D7	0010
RST 3	1	1	0	1	1	1	1	1	DF	0018
RST 4	1	1	1	0	0	1	1	1	E7	0020
RST 5	1	1	1	0	1	1	1	1	EF	0028
RST 6	1	1	1	1	0	1	1	1	F7	0030
RST 7	1	1	1	1	1	1	1	1	FF	0038

In Figure 12.1, the instruction RST 5 is built using resistors and tri-state buffer.

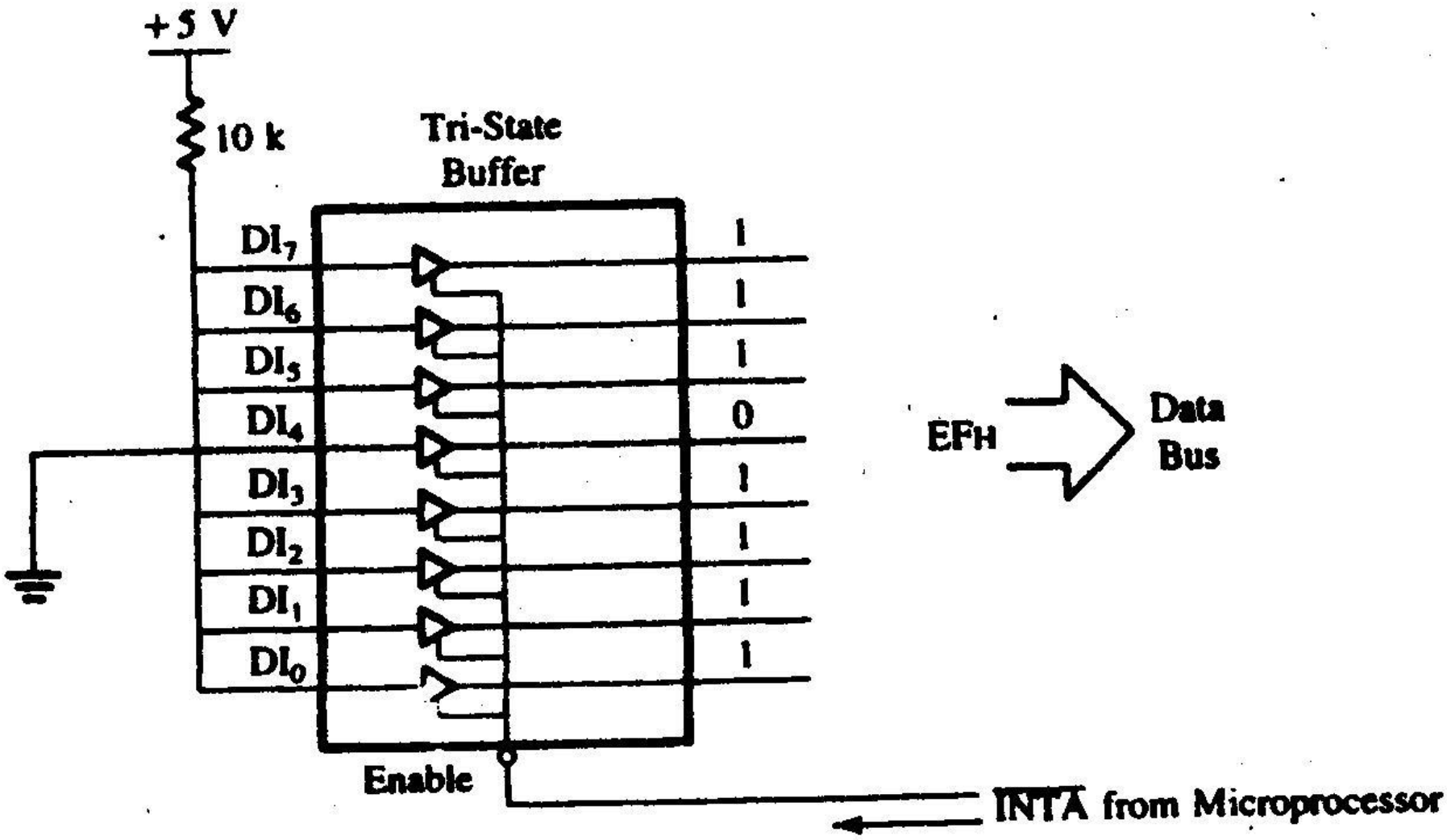


FIGURE 12.1
A Circuit to Implement the Instruction RST 5

12.1.2 Illustration: An Implementation of the 8085 Interrupt

PROBLEM SATEMENT

1. Write a main program to count continuously in binary with a one second delay between each count.

2. Write service routine at XX70H to flash FFH five times when the program is interrupted, with some appropriate delay between each count.

MAIN PROGRAM

```
            LXI SP, XX99H
            EI

            MVI A, 00H
NXTCNT:     OUT PORT1

            MVI C, 01H
            CALL DELAY

            INR A
            JMP NXTCNT
```

Service Routine

```
SERV:   PUSH B
        PUSH PSW

        MVI B, 0AH

        MVI A, 00H
FLASH:  OUT PORT1

        MVI C,01H
        CALL DELAY

        CMA

        DCR B
        JNZ FLASH

        POP PSW
        POP B
        EI
        RET
```

DESCRIPTION OF THE INTERRUPT PROCESS

1. The program will count continuously from 00H to FFH with a one second delay 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.

a.
b.
c.
d.
e. The program is transferred to memory location 0028H. The locations 0028-29-2AH should have the following Jump instructions 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 programs jumps to the service routine at XX70H.
5.
6.
7.

TESTING INTERRUPT ON A SINGLE-BOARD COMPUTER SYSTEM

In single-board microcomputers, some restart locations are usually reserved for users, and the system designer provides a Jump instruction at a restart location to jump somewhere in R/W memory. For example, in Intel's SDK-85 system, R/W memory begins at page 20H, and you may find the following instruction in the monitor program at memory location 0028H:

```
0028    JMP 20C2H
```

If instruction RST 5 is inserted as shown in Figure 12.3 , it transfers the program to location 0028H, and the monitor transfers the program from 0028H to location 20C2H. To implement the interrupt shown in Figure 12.3, you need to store the Jump instruction as shown below:

```
20C2    C3      JMP SERV
20C3    70
20C4    20
```

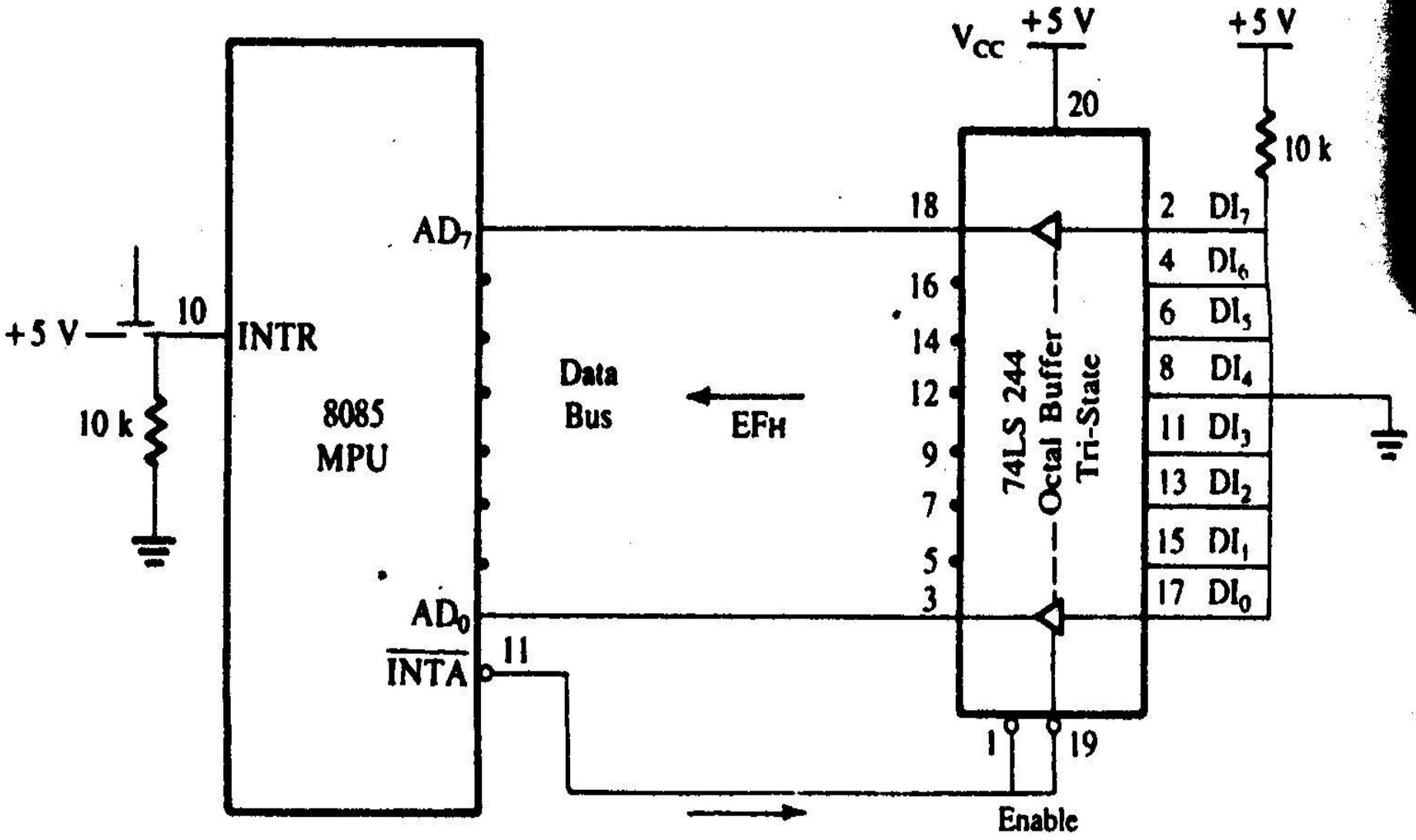


FIGURE 12.3
Schematic to Implement the 8085 Interrupt

ISSUES IN IMPLETING INTERRUPTS

In the above illustration, some questions remain unanswered:

1. *Is there a minimum pulse width required for the INTR signal?*

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 3 MHz clock frequency (such as the SDK-85 system), the input pulse to INTR should be at leas 5.8 μs long.

2. *How long can the INTR pulse stay high?*

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. In Figure 12.3, the manual push button will keep the INTR high for more than 20 ms; however, the service routine has a delay of 1 second, and the EI instruction is executed at the end of the service routine.

3. *Can the microprocessor be interrupted again before the completion of the first interrupt service routine?*

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.

12.2 8085 VECTORED INTERRUPTS

The 8085 has five interrupt inputs (Figure 12.5). One is called INTR, three are called RST 5.5, RST 6.5, and RST 7.5 respectively, and the fifth is called TRAP, a nonmaskable interrupt. These last four (RSTs and TRAP) are automatically vectored (transferred) to a specific locations on memory page 00H without any external hardware. The 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 highest priority, followed by RST 7.5, 6.5, 5.5, and INTR, in that order; however, the TRAP has lower priority than the Hold signal used for DMA.

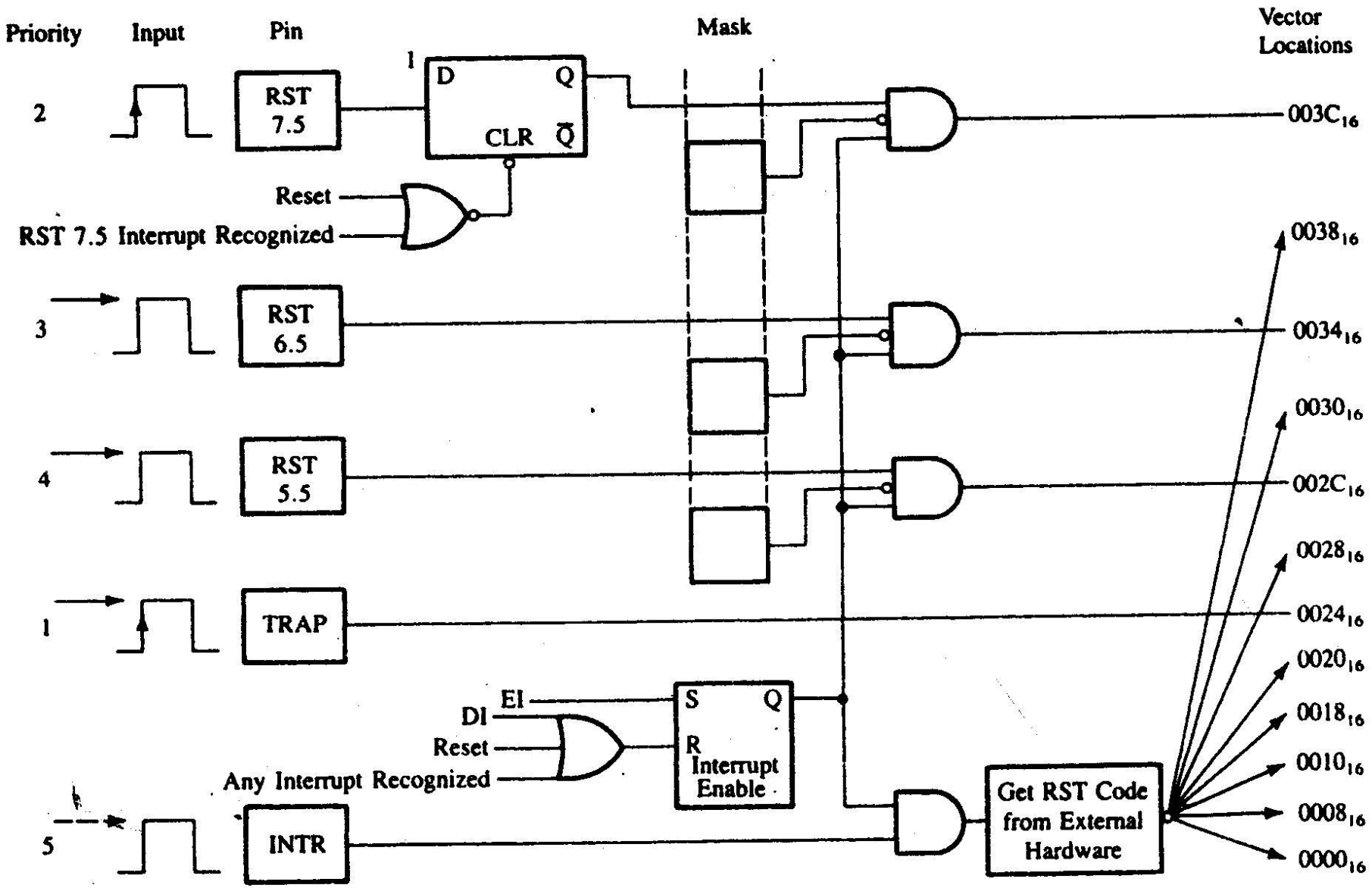


FIGURE 12.5
The 8085 Interrupts and Vector Locations

SOURCE: Intel Corporation, *MCS 80/85 Student Study Guide* (Santa Clara, Calif.: Author, 1979).

12.2.1 TRAP

TRAP, a nonmaskable interrupt known as NMI, is analogous to smoke detector. It has the highest priority among the interrupt signals, it need not to 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 transition from high to low to high.

Figure 12.5 shows that when this interrupt is triggered, the program control is transferred to location 0024H without any external hardware or the interrupt enable instruction EI. TRAP is generally used for such critical events as power failure and emergency shut-off.

12.2.2 RST 7.5, 6.5, and 5.5

These maskable interrupt are enabled under program control with two instructions: EI (Enable Interrupt), and SIM (Set Interrupt Mask) described below:

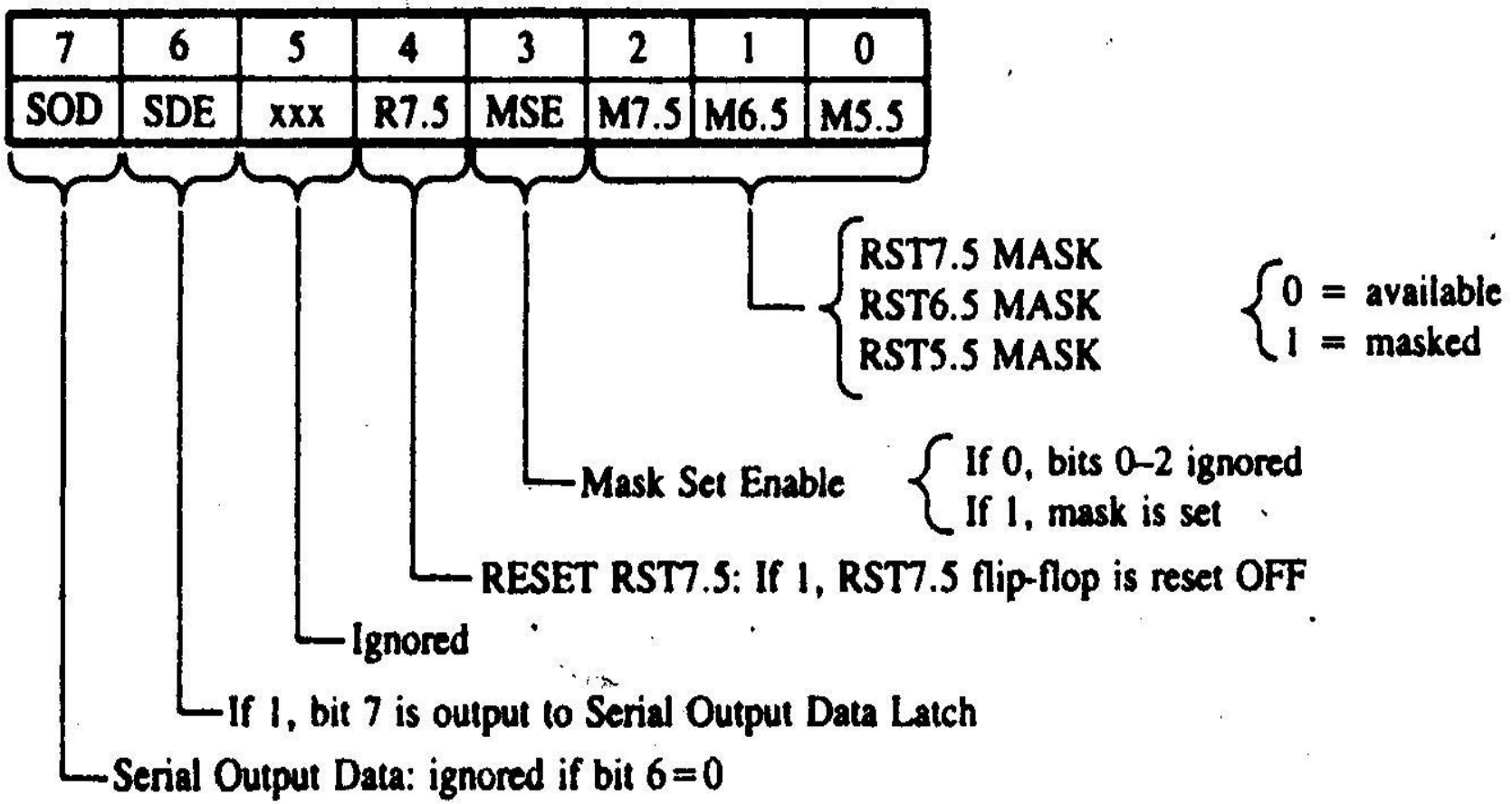


FIGURE 12.6

Interpretation of the Accumulator Bit Pattern for the SIM Instruction

SOURCE: Intel Corporation, *Assembly Language Programming Manual* (Santa Clara, Calif.: Author, 1979), pp. 3–59.

Instruction SIM:

Set Interrupt Mask. This is a 1–byte instruction and can be used for three different functions (Figure 12.6).

- 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 accumulator. Bit D_3 is a control bit and should = 1 for bits D_0, D_1, and D_2 to be effective. Logic 0 on D_0, D_1, and D_2 will enable the corresponding interrupts, and logic 1 will disable the interrupts.

- The second function is to reset RST 7.5 flip-flop (Figure 12.6). Bit D_4 is additional control for RST 7.5. If D_4=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 D_7 and D_6) and do not affect the interrupts.

The entire interrupt process (except TRAP) is disabled by resetting the Interrupt Enable flip-flop (Figure 12.5). The flip-flop can be reset in one of the three ways:

- Instruction DI
- System Reset
- Recognition of an Interrupt Request

Example: 12.1: Enable all the interrupts in an 8085 system.

```
Instruction:    EI              ; Enable interrupts
                MVI A, 08H      ; To enable RST 7.5, 6.5, and 5.5
                SIM             ; Enable RST 7.5, 6.5, and 5.5
```

Example: 12.2: Reset the 7.5 interrupt from Example 12.1.

```
Instruction:    MVI A, 18H      ; Set D4 = 1
                SIM             ; Reset 7.5 interrupt flip-flop
```

PENDING INTERRUPTS

When one interrupt request is being served, other interrupt request may occur and remain pending. The 8085 has an additional instruction call RIM (Read Interrupt Mask) to sense these pending interrupts (Figure 12.7).

Instruction RIM:

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

The RIM instruction loads the accumulator with the following information:

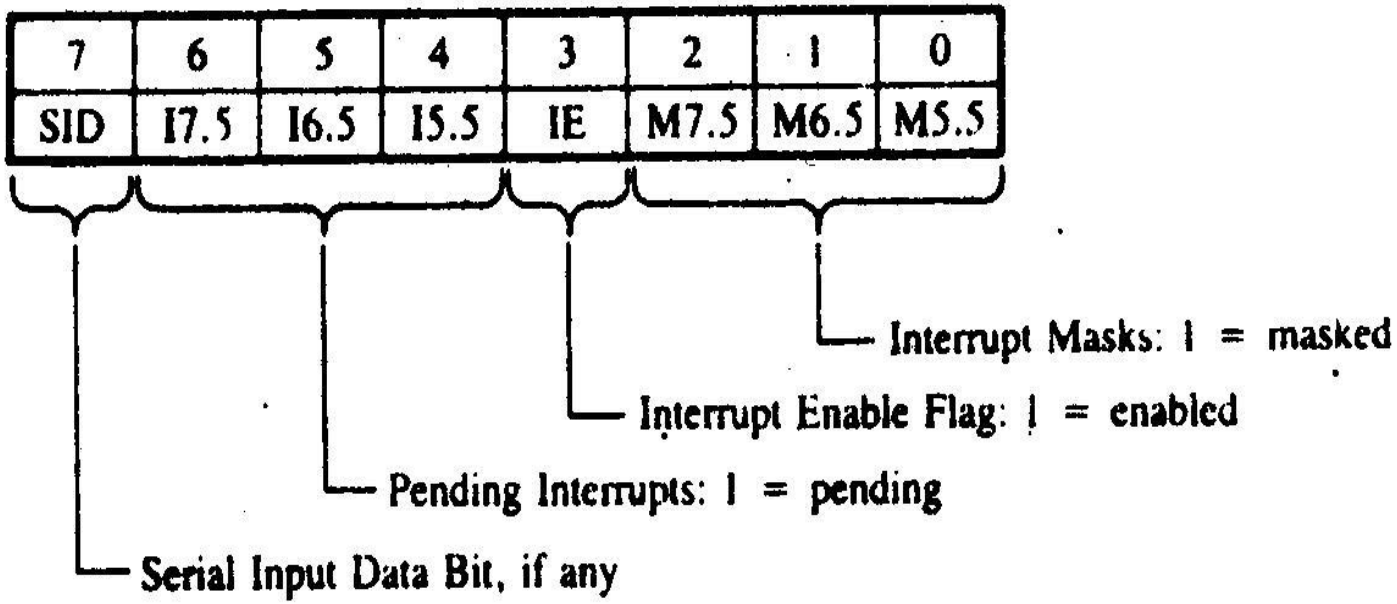


FIGURE 12.7

Interpretation of the Accumulator Bit Pattern for the RIM Instruction

SOURCE: Intel Corporation. *Assembly Language Programming Manual* (Santa Clara, Calif.: Author, 1979), pp. 3-49.

- To read interrupt masks, This instruction loads the accumulator with 8 bits indicating the current status of the interrupt masks (Figure 12.7).
- To identify pending interrupts. Bits D_4, D_5, and D_6 (Figure 12.7) identify the pending interrupts.
- To receive serial data bit D_7 is used (Figure 12.7).

Example: 12.3: Assuming the microprocessor is completing an RST 7.5 interrupt request, check to see if RST 6.5 is pending. If it is pending, enable RST 6.5 without affecting any other interrupts; otherwise return to the main program.

Instruction:

```
RIM
MOV B, A
ANI 20H          ; Check whether RST 6.5 is pending
JNZ NEXT
EI
RET
MOV A,B
ANI 0DH          ; Enable RST 6.5 by sending D1=0
ORI 08H          ; Enable SIM by setting D3=1
SIM
JMP SERV         ; Jump to service routine for RST 6.5
```

12.2.3 Illustration: Interrupt-Driven Clock

PROBLEM STATEMENT

Design a 1-minute timer using a 60 H_z 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.

HARDWARE DESCRIPTION

This 1-minute timer is designed with a 60 H_z AC line. The circuit (Figure 12.8) uses a step-down transformer; the 74121 monostable multivibrator, 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 H_z 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 service routine is executed.

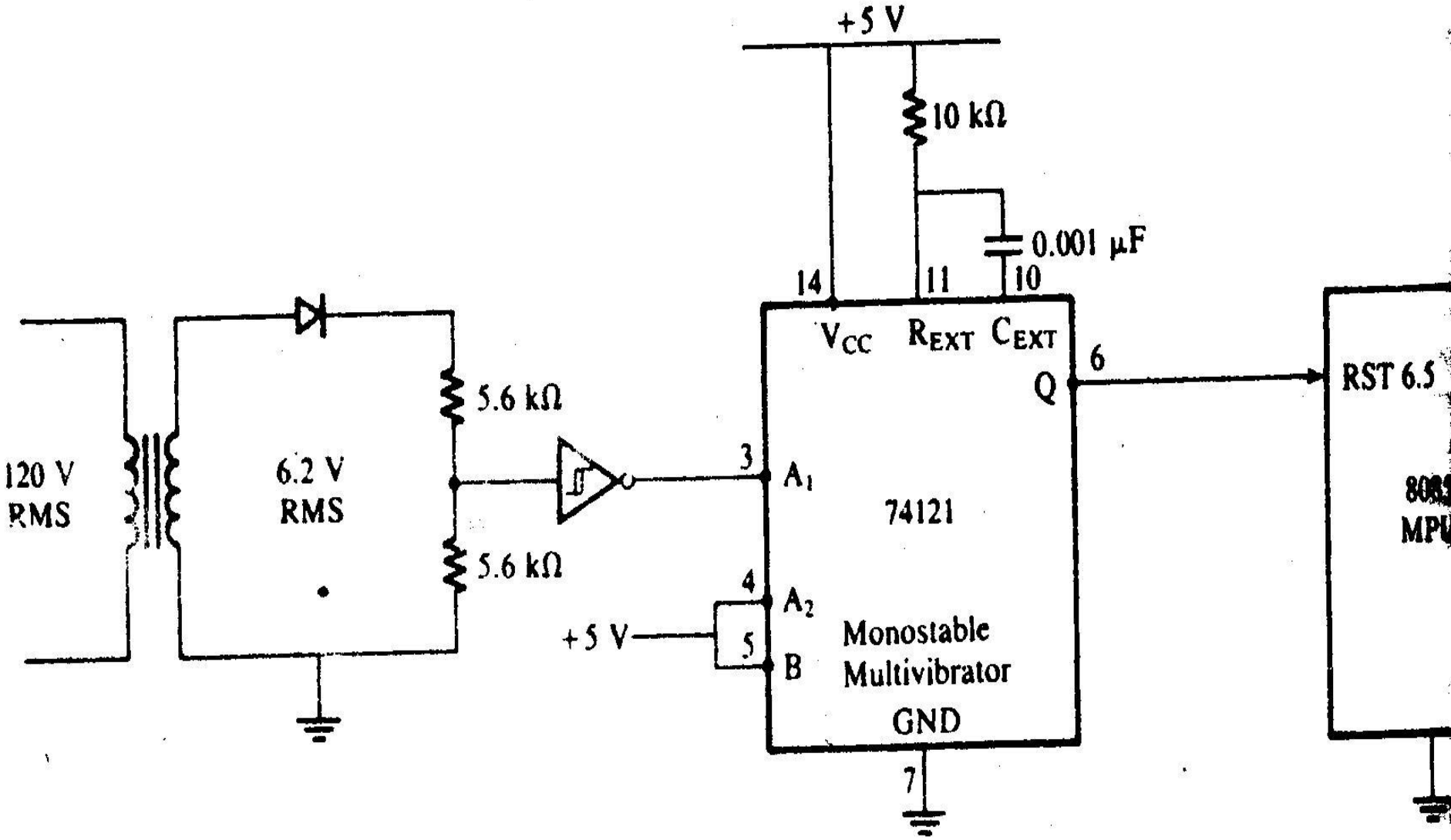


FIGURE 12.8
Schematic of Interrupt-Driven Timer Clock

Monitor Program

```
0034        JMP RWM
```

Main Program

```
            LXI SP, STACK
            RIM
            ORI 08H
            SIM
            LXI B,0000H     ; Set up B for minutes and C for seconds

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

RWM:        JMP TIMER       ; This is RST 6.5 vector location 0034H
```

Interrupt Service Routine

```
; Section I

TIMER:      DCR D          ;One interrupt occurred
            EI
            RNZ            ; Has 1 second elapsed? If not, return
;Section II
            DI
            MVI D,3CH      ;1 second is complete; load D to count again
            MOV A,C
            ADD 01H
            DAA            ; Decimal-adjust “Second”
            MOV C,A        ;Save “BCD” seconds
            CPI 60H
            EI
            RNZ

;Section III
            DI
            MVI C,00H      ;60 seconds complete; clear “Seconds” register
            INR B          ; Increment “Minutes” register
            RET
```

Chapter-13

Interfacing Data Converters

13.1 DIGITAL-TO-ANALOG (D/A) CONVERTERS

Digital-to-Analog converters can be classified in three categories:

- current output: provides current as output signal.
- voltage output: internally converts current signal into voltage signal.
- multiplying type: output current or voltage is product of input current or voltage.

Voltage output is slower than current output because of the delay in converting the current signal into voltage signal. However, in many applications, it is necessary to convert current into voltage by using an external operational amplifier.

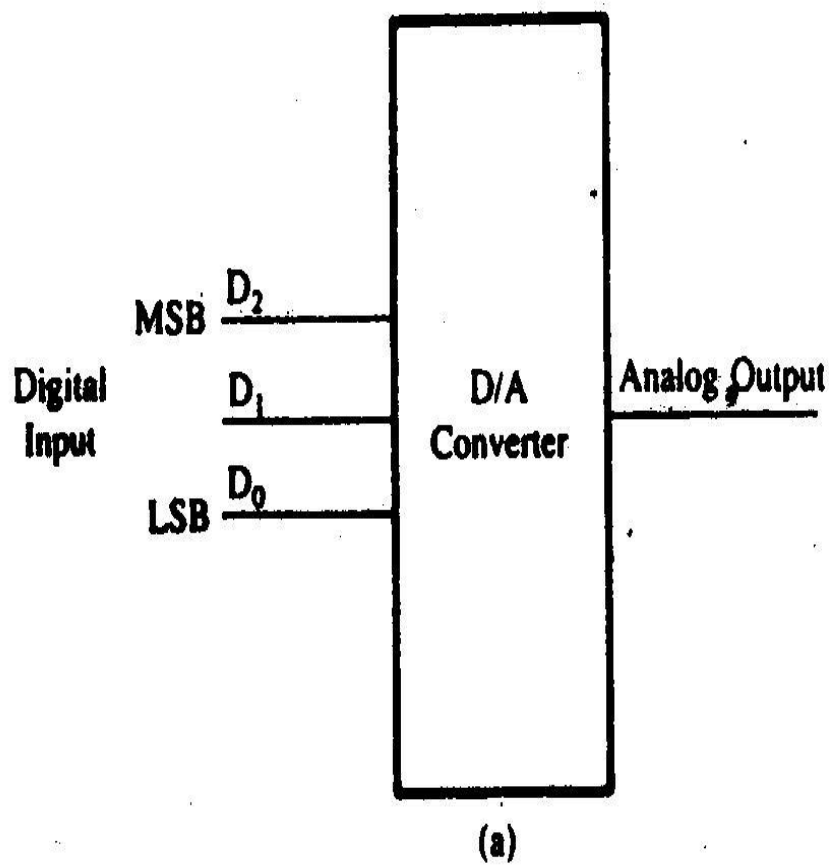
13.1.1 Basic Concepts

Figure 13.1(a) shows a block diagram of a 3-bit D/A converter. If the input ranges from 0 to 1 V, it can be divided into eight equal parts (1/8 V); each successive is 1/8 V higher than the previous combination, as shown in Figure 13.1(b).

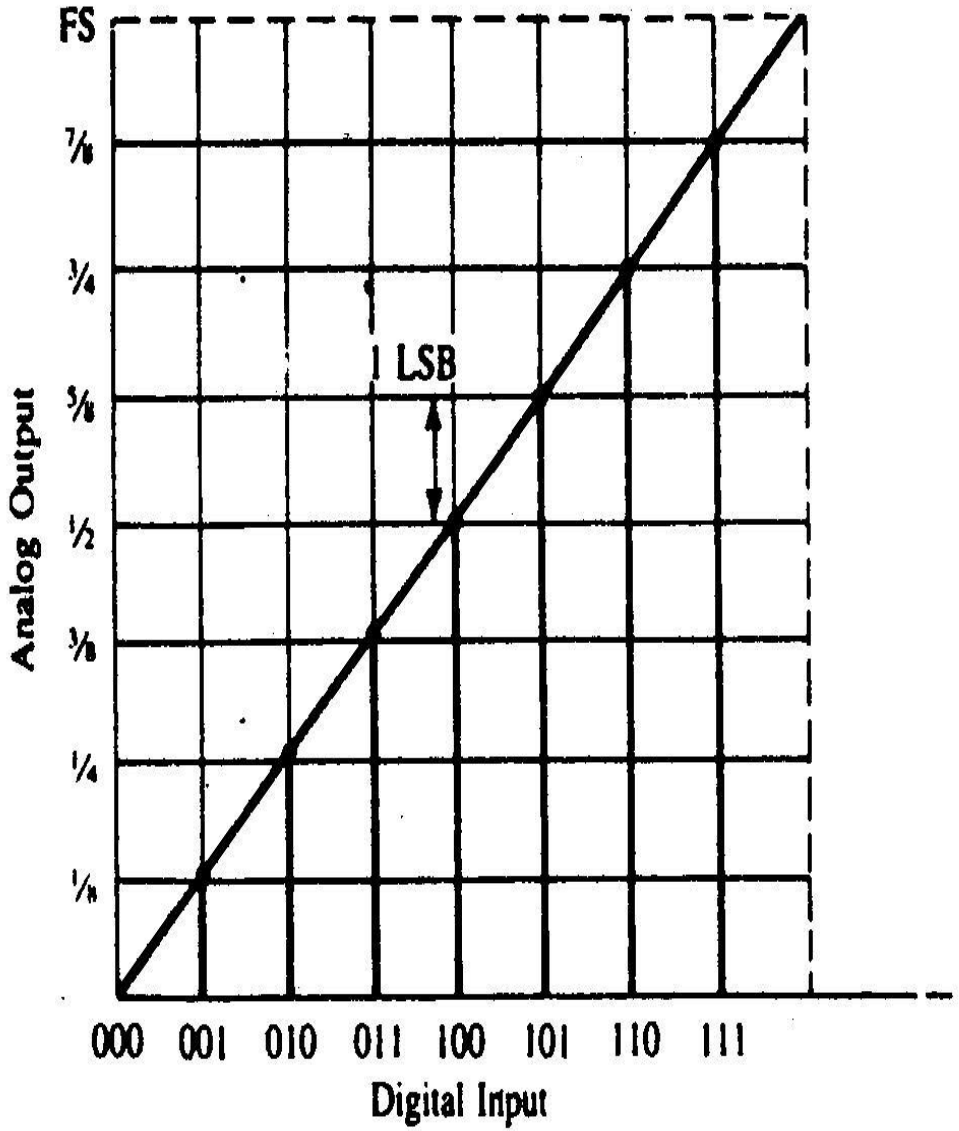
The following points can be summarized from the graph:

1. If a converter has n input lines, it can have 2^n input combinations.
2. If the full-scale analog voltage is 1 V, the smallest unit or the LSB (001_2) is equivalent to $1/2^n$ of 1 V. This is defined as resolution.
3. The MSB represents half of the full-scale value. In this example, the MSB (100_2)=1/2 V.
4. For the maximum input signal (111_2), the output signal is equal to the value of the full scale input signal minus the value of the 1 LSB input signal. In this example, the maximum input signal (111_2) represents 7/8 V.

Example 13.1: Calculate the values of the LSB, MSB, and full-scale output for an 8-bit DAC for the 0 to 10 V range.



(a)



(b)

FIGURE 13.1

A 3-Bit D/A Converter: Block Diagram (a) and Digital Input vs. Analog Output (b)

Solution:

1. LSB = $1/2^8 = 1/256$

   For 10 V, LSB = 10 V/256 = 39 mV

2. MSB = 1/2 full scale = 5V

3. Full-Scale Output = (Full-Scale Value – 1 LSB)
   = 10V – 0.039 V
   = 9.961 V

13.1.2 D/A Converter Circuits

The input resistors R_1, R_2, and R_3 are selected in binary weighted proportion; each has double the value of the previous resistor. If all three inputs are 1 V, the total output current is

$$I_o = I_T = I_1 + I_2 + I_3$$
$$= V_{in}/R_1 + V_{in}/R_2 + V_{in}/R_3$$
$$= V_{in}/1k(1/2 + 1/4 + 1/8)$$
$$= 0.875 \text{ mA}$$

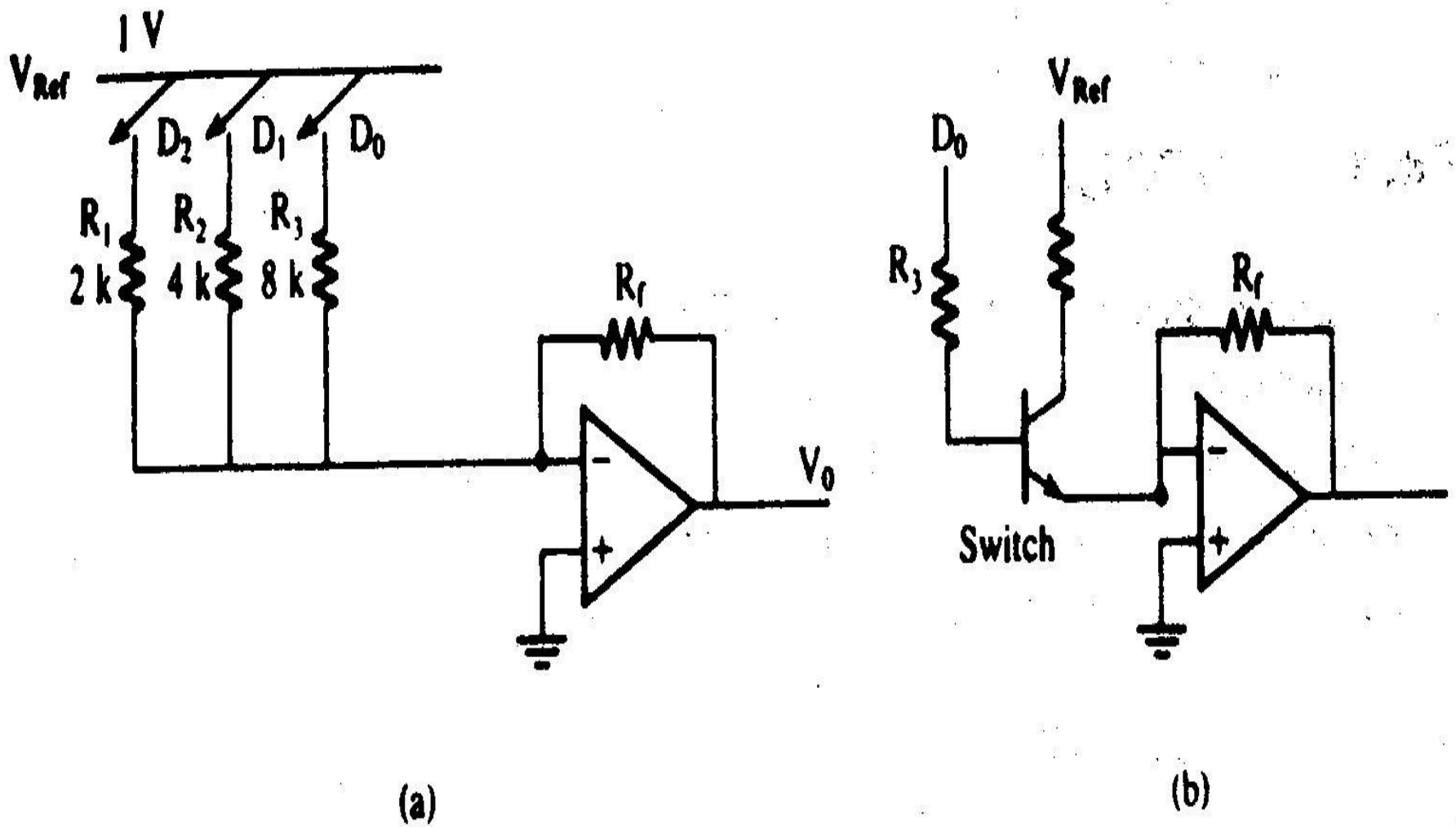


FIGURE 13.3
Simulated D/A Converter (a) and Transister Switch to Turn On/Off Bit D_0 (b)

The voltage output is

$$V_o = - R_f I_T$$
$$= - (1\ k)\ (0.875\ mA)$$
$$= |7/8|$$

This example shows that for the input = 111_2 , the output is equal to either 7/8 mA or 7/8 V, representing the D/A conversion process.

Now the output current I_o can be generalized for any number of bits as

$$I_o = V_{Ref} /R\ (A_1/2 + A_2/4 + - - - - - + A_n/2^n)$$

The following points can be inferred from the above example:

1. A D/A converter circuit requires three elements: resistor network with appropriate weighting, switches, and a reference source.
2. The output can be a currant signal or converted into a voltage signal using an operational amplifier.
3. The time required for conversion, called settling time, is dependent on the response of the switches and output amplifier (for a voltage output DAC).

13.1.3 Illustration: Interfacing an 8-Bit D/A Converter

PROBLEM SATEMENT

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; refer to Figure 13.5(a).
2. Write a program to generate a continuous ramp waveform.
3. Explain the operation of the 1408 in Figure 13.5(b) , which is calibrated for a bipolar range ±5V. Calculate the output V_O if the input is 10000000_2 .

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:

$$I_o = V_{Ref} /R_{14} (A_1/2 + A_2/4 + A_3/8 + A_4/16 + A_5/32 + A_6/64 + A_7/128 + A_8/256)$$

Where inputs A_1 through A_8 = 0 or 1.

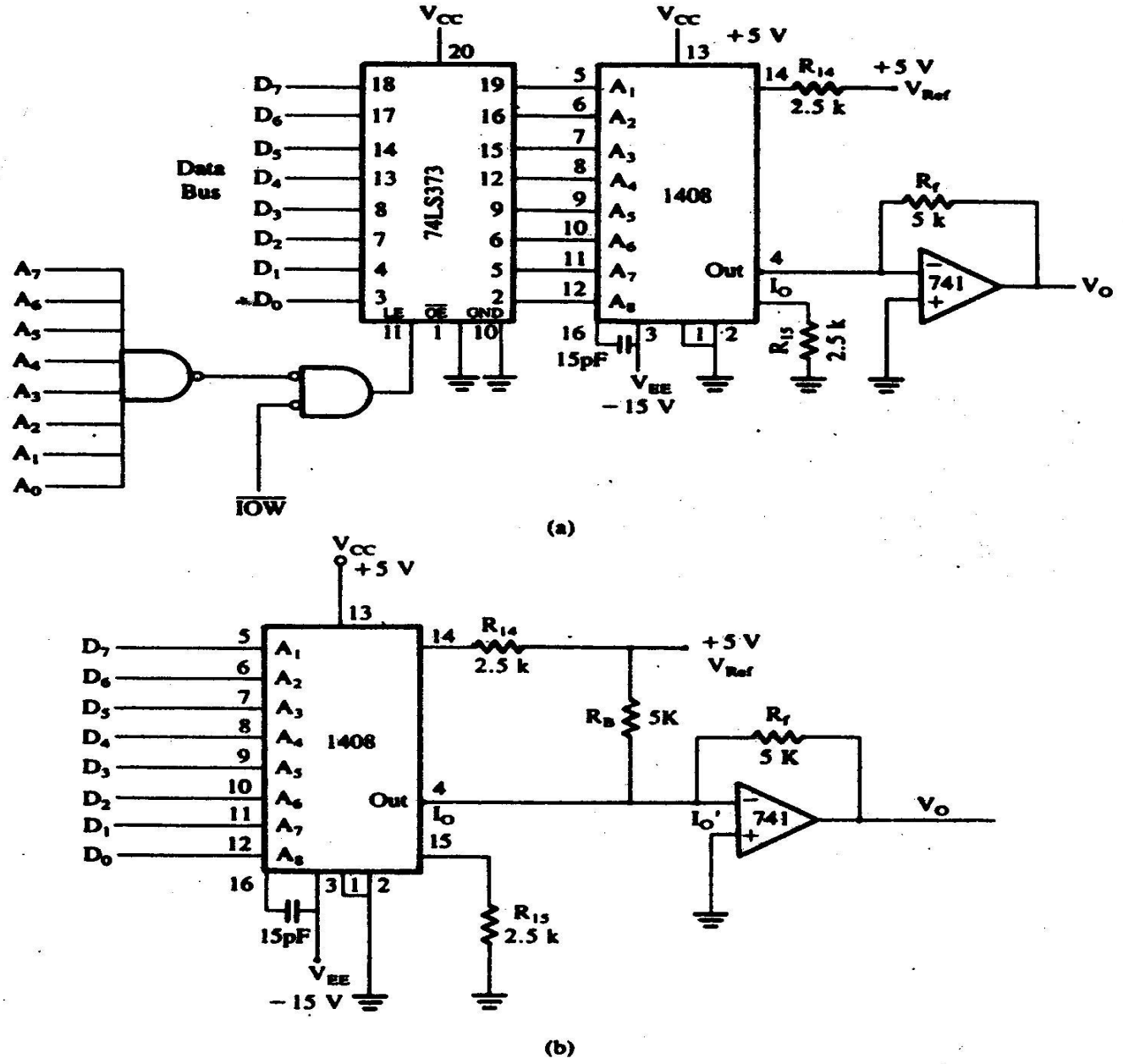


FIGURE 13.5
Interfacing the 1408 D/A Converter: Voltage Output in Unipolar Range (a) and in Bipolar Range (b)

This formula is an application of the generalized formula for the current I_o. For full scale input (D_7 through D_o = 1).

$$I_o = 5\ V / 2.5\ k\ (1/2 + 1/4 + 1/8 + 1/16 + 1/32 + 1/64 + 1/128 + 1/256)$$

$$= 2\ mA\ (255/256)$$

$$= 1.992\ mA.$$

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

$$= 2\ mA\ (255/256) \times 5\ k$$

$$= 9.961\ V$$

PROGRAM

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

```
        MVI A, 00H          ; First input 00H for DAC
DTOA:   OUT FFH             ; Output to DAC
        MVI B, COUNT        ; COUNT for delay
DELAY:  DCR B
        JNZ DELAY
        INR A               ; Next input for DAC
        JMP DTOA
```

Program Description:

The program outputs 00H to FFH as the digital inputs for DAC. The analog output of the DAC starts at 0 V 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 and that can displayed on oscilloscope.

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 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. The resistor R_B supplies 1 mA (V_{Ref}/R_B) current to the opposite direction of the current generated by the input signal. Therefore, the output current for the bipolar operation I_o is

$$I_o' = I_o - V_{Ref}/R_B$$

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

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

$$V_o = I_o' R_f$$
$$= (I_o - V_{Ref}/R_B)\, R_f$$
$$= (0 - 5\text{ V}/5\text{k})\,(5\text{k}) \qquad (I_o = 0 \text{ for input} = 0)$$
$$= -5\text{V}$$

When the input = 1000 0000, the output V_o is

$$V_o = (I_o - V_{Ref}/R_B) R_f$$

$$= (V_{Ref}/R_{14} \times A_1/2 - V_{Ref}/R_B) R_f \qquad (A_2 – A_8 = 0)$$

$$= (5\ V/2.5\ k \times 1/2 - 5\ V/5\ k)5k$$

$$= (1\ mA – 1\ mA)5\ k$$

$$= 0$$

13.2 ANALOG-TO-DIGITAL CONVERTERS

The A/D conversion is a quantization process whereby an analog signal is represented by equivalent binary states; this is opposite to the D/A conversion process. Analog-to-Digital converters can be classified in two general groups based on the conversion technique:

- Successive-approximation A/D converter: used in instrumentation.
- Integration-type A/D converter: used in digital meters, monitoring systems.

13.2.1 Basic Concepts

Figure 13.8(a) shows a block diagram of a 3-bit A/D converter. Figure 13.8(b) shows the graph of the analog input voltage (0 to 1 V) and the corresponding digital output signal. The resolution of the converter is 1/8 V.

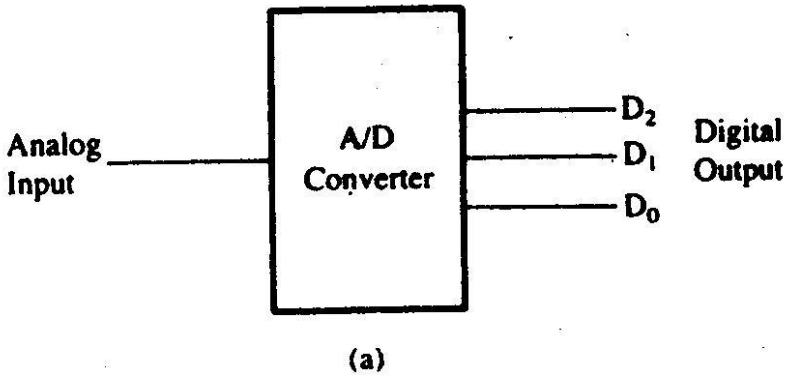
13.2.1 Successive-Approximation A/D Converter

Figure 13.9(a) shows the block diagram of a successive approximation A/D converter 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 V_o with the analog input signal V_{in}. 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 equivalent digital signal.

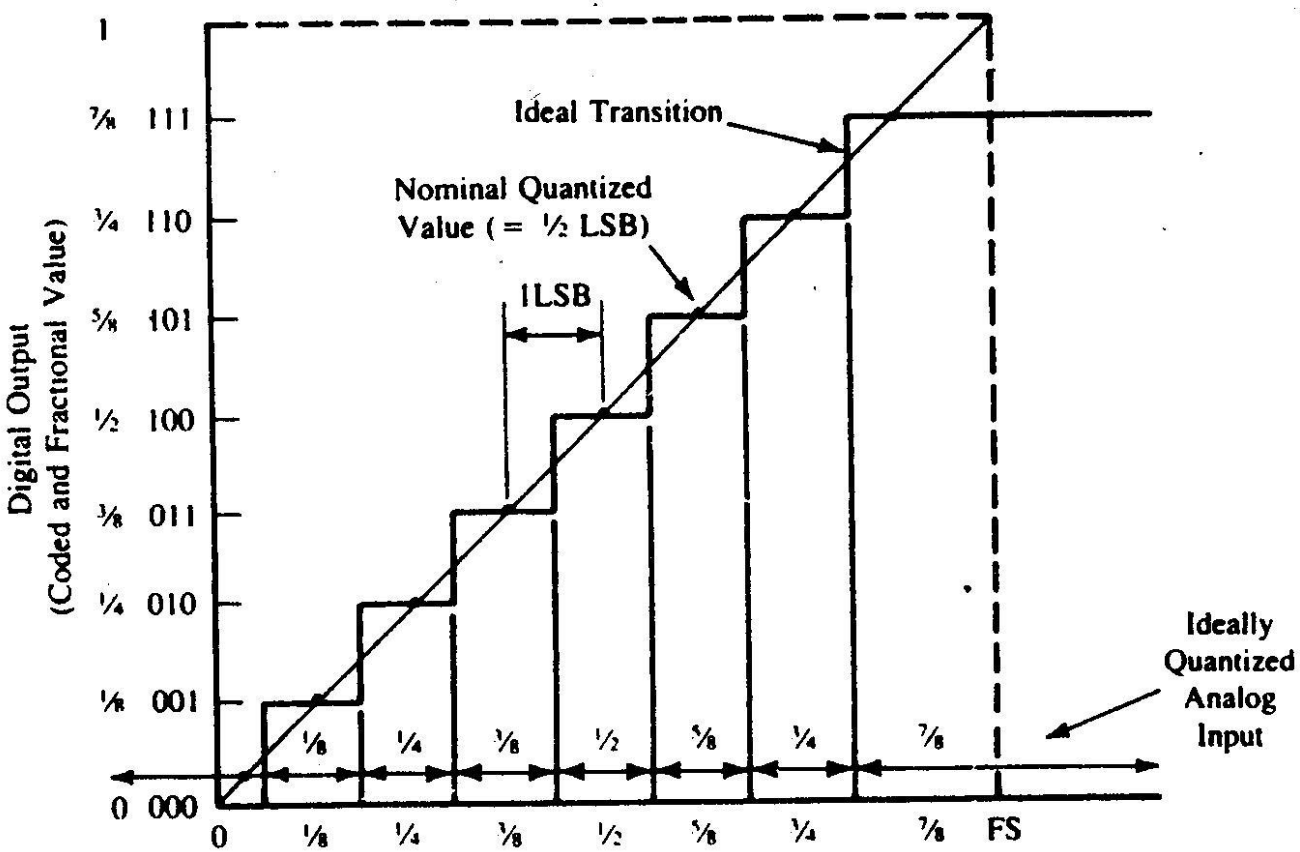
In the case of a 4-bit A/D converter, bit D_3 is turned on first and the output of the DAC is compared with an analog signal. If the comparator changes the state, indicating that the output generated by D_3 is larger than the analog signal, bit D_3 is turned off in the SAR and bit D_2 is turned on. The process continues until the input reaches bit D_0.

FIGURE 13.8
A 3-Bit A/D Converter: Block Diagram (a) and Analog Input vs. Digital Output (b)

SOURCE: Analog Devices, Inc., *Integrated Circuit Converters, Data Acquisition Systems, and Analog Signal Conditioning Components* (Norwood, Mass.: Author, 1979), pp. 1-18.



(a)



(b)

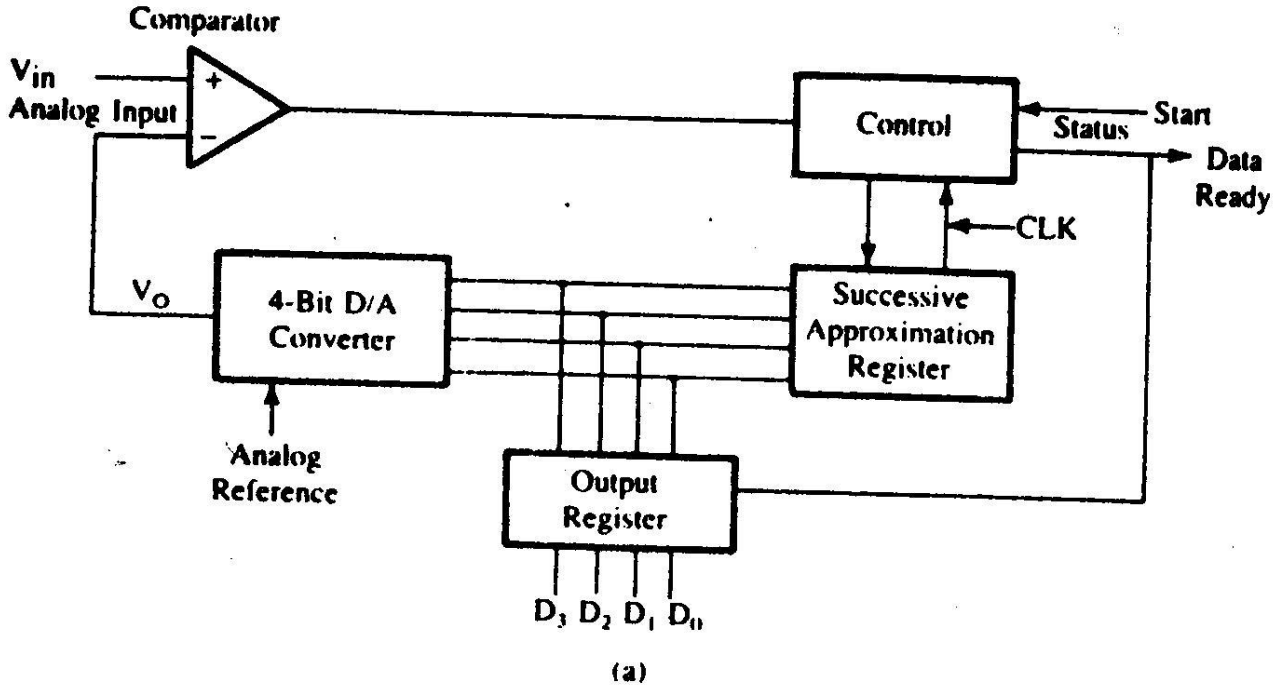


FIGURE 13.9
Successive-Approximation A/D Converter: Block Diagram (a) and Conversion Process for a 4-Bit Converter (b)

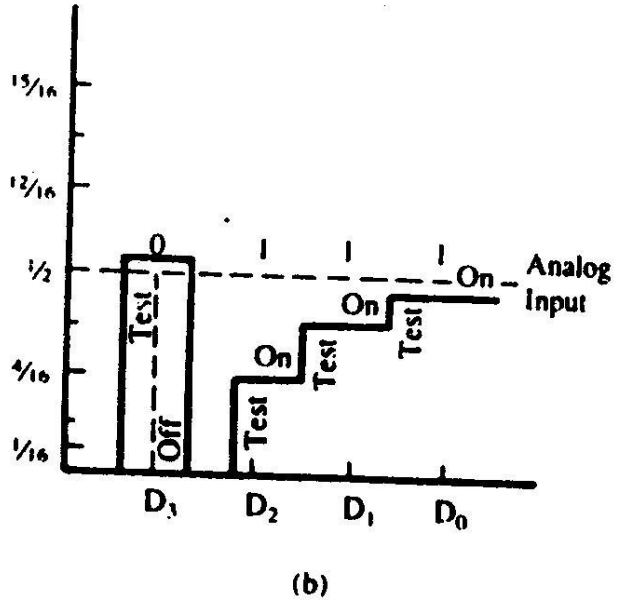


Figure 13.9(b) illustrates a 4-bit conversion process. When bit D_3 is turned on, the output exceeds the analog signal and therefore, D_3 is turned off. When the next three successive bits are turned on, the output becomes approximately equal to the analog signal.

13.2.1 Interfacing 8-Bit A/D Converters

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.

PROGRAM

```
        OUT 82H
TEST:   IN 80H
        RAR
        JC TEST
        IN 81H
        RET
```

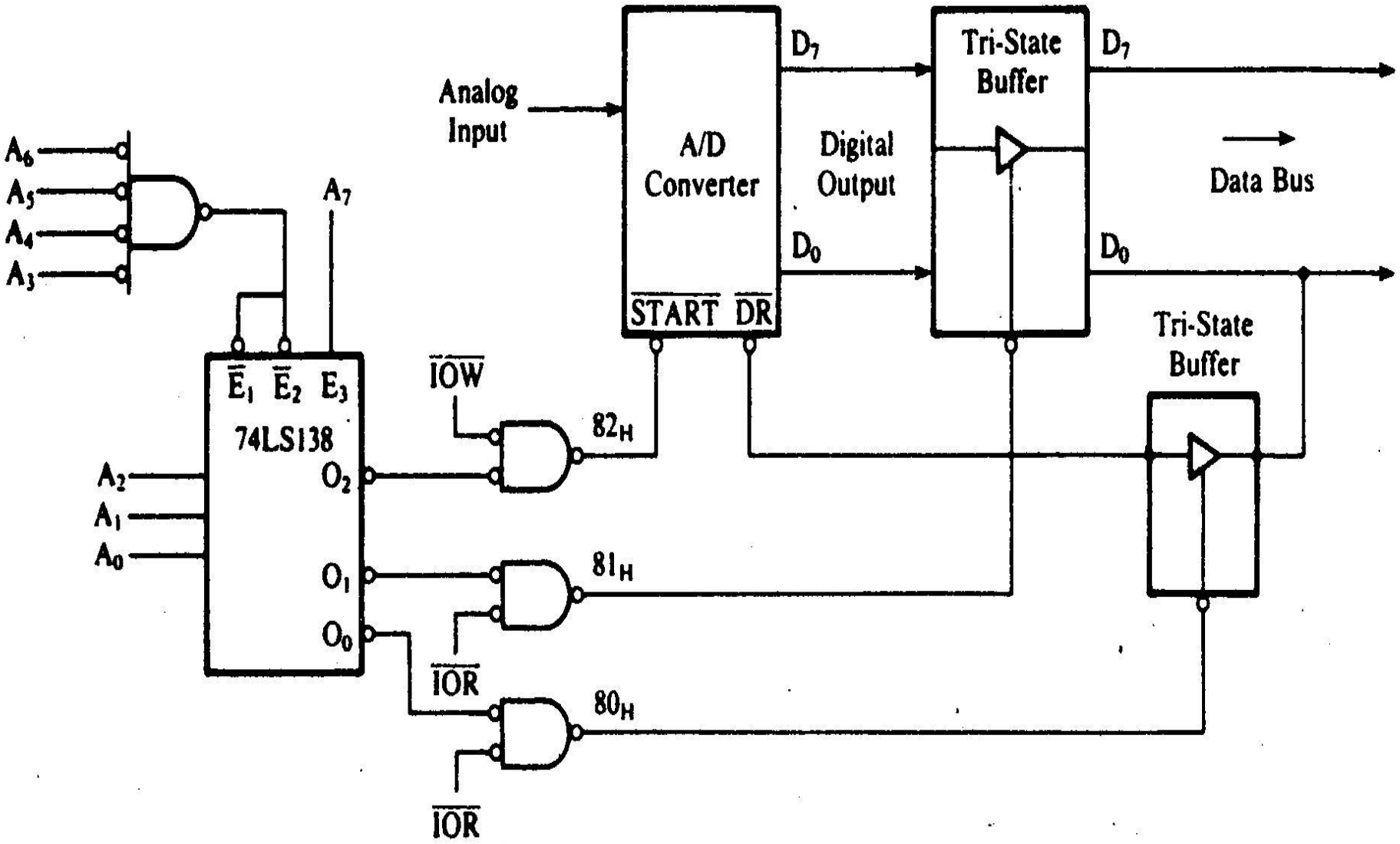


FIGURE 13.11
Interfacing an A/D Converter Using the Status Check

Chapter-14

Programmable Interface Devices:

- **8155 I/O and Timer**
- **8279 Keyboard/Display Interface**

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 **control register**.

14.1 BASIC CONCEPTS IN PROGRAMMABLE DEVICES

In Chapter 5 we assumed that the I/O devices were always ready for data transfer. But in the situation to interface a printer (a slower device) to a microprocessor (the fastest unit) there must be some signals exchanged between them prior to sending and receiving data. These signals are called handshake signals. Otherwise, processor will read the same data again and again for a slow responding input device and overwrite the previous data by the next data for a slow responding output device.

14.1.1 Making the 74LS245 Transceiver Programmable

The 74LS245 is a bidirectional tri-state octal buffer, and the direction of data flow is determined by the signal of DIR. When DIR is high, data flow from S to B, and when it is low, data flow from B to A.

In fact, this is a hard-wired programmable device; the direction of the data flow is programmed through DIR. However, we are interested in a device that can be programmed by writing an instruction through the MPU. This can be accomplished by adding a register called the control register, as shown in Figure 14.2, and by connecting the DIR signal to bit D_o of the control register. When $D_o = 1$, data flow from A to B as output, and when $D_o = 0$, data flow in the opposite direction as input.

Example 14.1: Write instructions to initialize the hypothetical chip (Figure 14.2) as an output buffer.

Solution: **Instructions**

```
MVI A,01H
OUT FFH
MVI A, BYTE1
OUT FEH
```

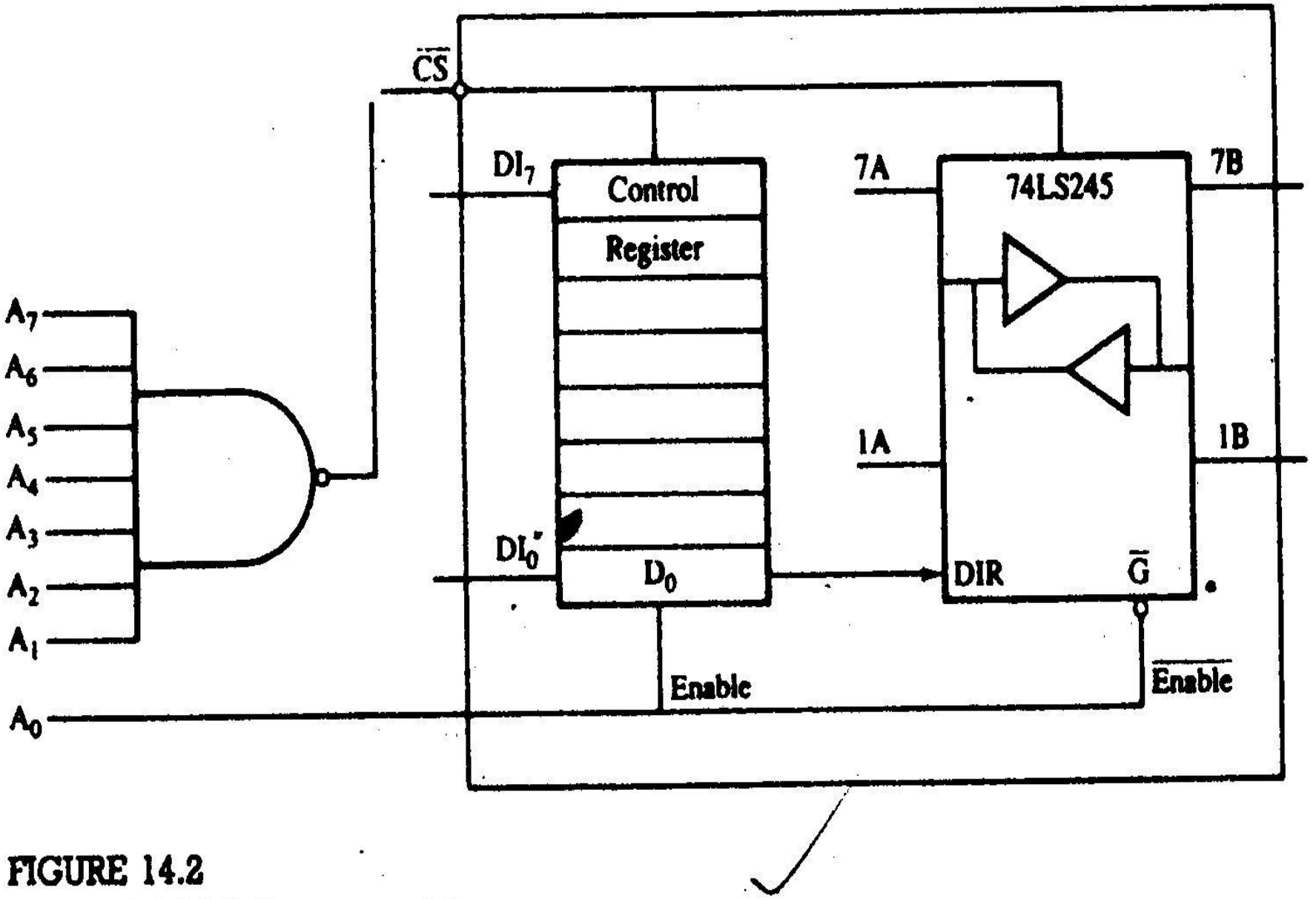


FIGURE 14.2
Making 74LS245 Programmable

14.1.2 Programmable Device with a Statues Register

Figure 14.3 shows a hypothetical programmable device with a status register. There are three registers and can be accessed as ports. The port addresses and the functions of these registers can be summarized as follows:

Control register (output only) = FFH (A_1 and A_0 = 1)
Status register (input only) = FFH (A_1 and A_0 = 1)
Data register (input or output) = FFH (A_1=1, A_0 = 0)

14.1.3 Programmable Device with a Handshake Signals

The MPU and peripherals operate at different speeds; therefore, signals are exchanged prior to data transfer between fast-responding MPU and slow-responding peripherals such as printers and data converters. These signals are called **handshake signals**. 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 (Strobe) and IBF (Input Buffer Full) and one interrupt signal (INTR). Figure 14.4(b) shows the programmable device in the output mode using the same handshake signals, except that they are labeled differently: OBF (Output Buffer Full) and ACK (Acknowledgement).

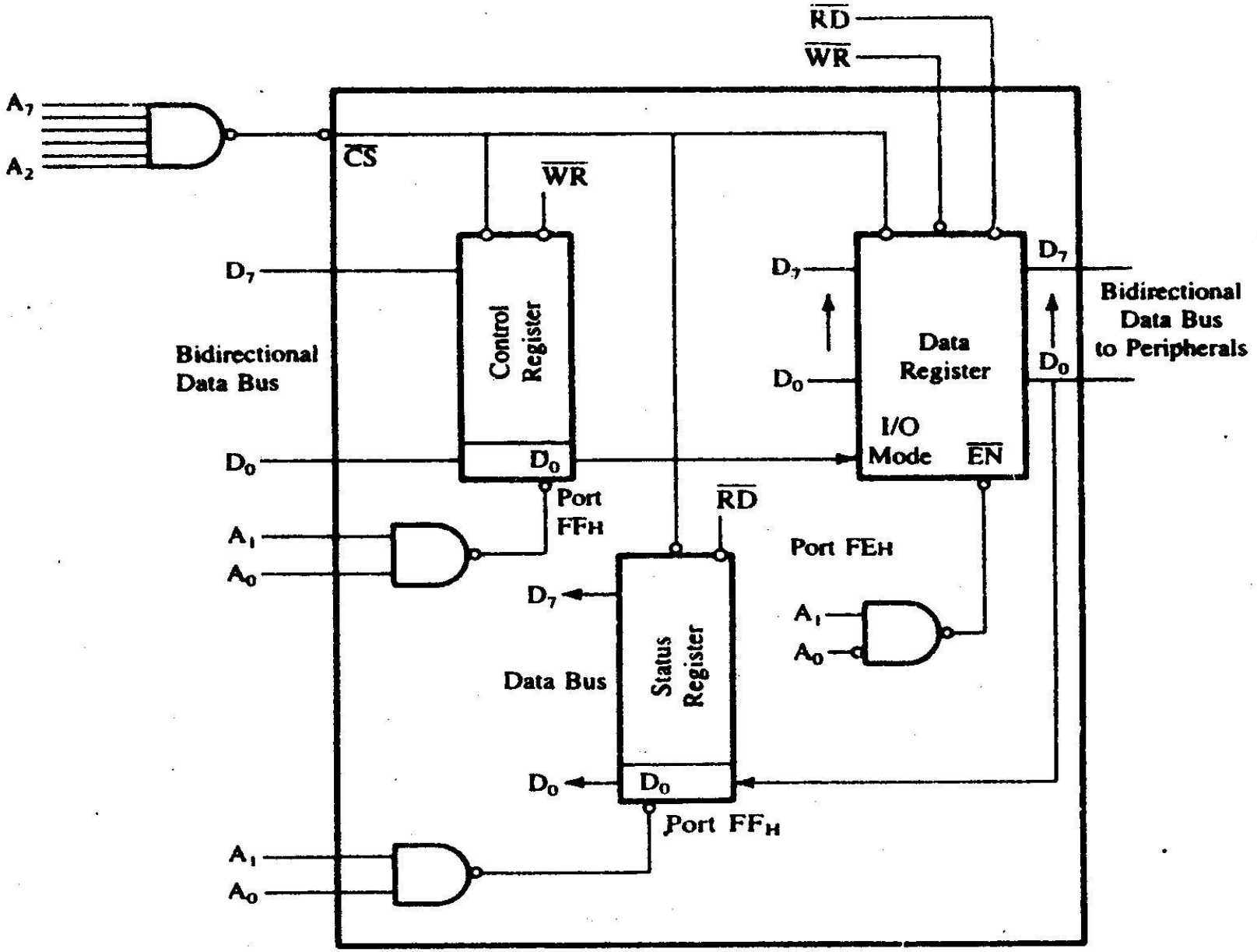


FIGURE 14.3
A Hypothetical Programmable Device with a Status Register

4. The MPU reads the byte by sending control signal RD.

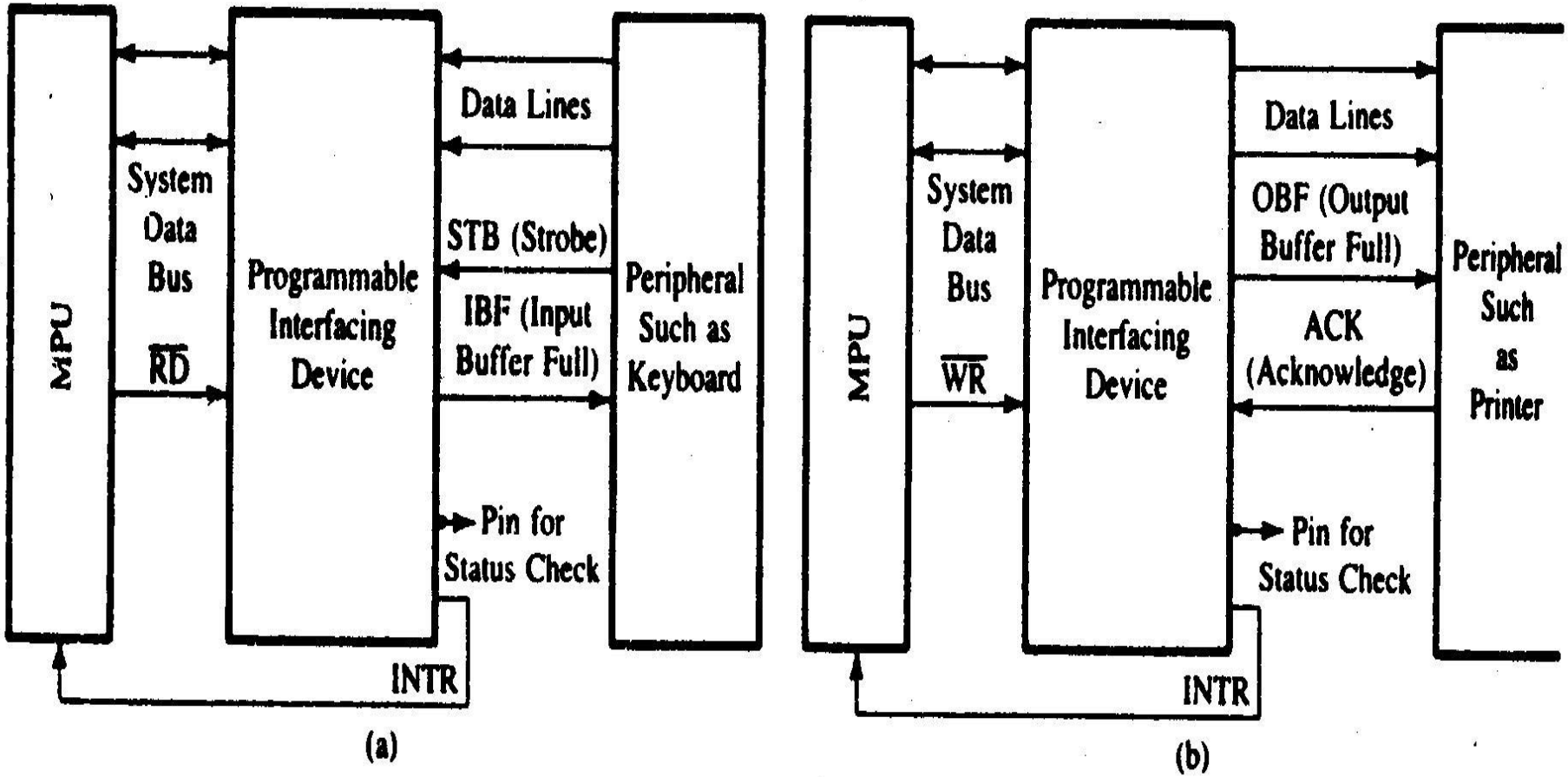


FIGURE 14.4
Interfacing Device with Handshake Signals for Data Input (a) and Data Output (b)

Examination of Figure 14.4 shows the following similarities among the handshake signals:

1. Handshake signals ACK and STB are input signals to the device and perform similar functions, although they are called by different names.
2. Handshake signals OBF and IBF are output signals from the device and perform similar functions (Buffer Full).

14.2 THE 8155: MULTIPURPOSE PROGRAMMABLE DEVICE

The 8155 is a multipurpose programmable device specially designed to be compatible with the 8085 microprocessor. The ALE, IO/$\overline{M}$, $\overline{RD}$, and $\overline{WR}$ signals from the 8085 can be connected directly to the device; this eliminates the need for external demultiplexing of the low-order bus $AD_7 – AD_0$ and generation of the control signals such as $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$, and $\overline{IOW}$.

The 8155 includes 256 bytes of R/W memory, three I/O ports, and a timer. Figure 14.5 shows the pin configuration and block diagram of the 8155 programmable device. The programmable I/O sections of this device are illustrated in the following sections:

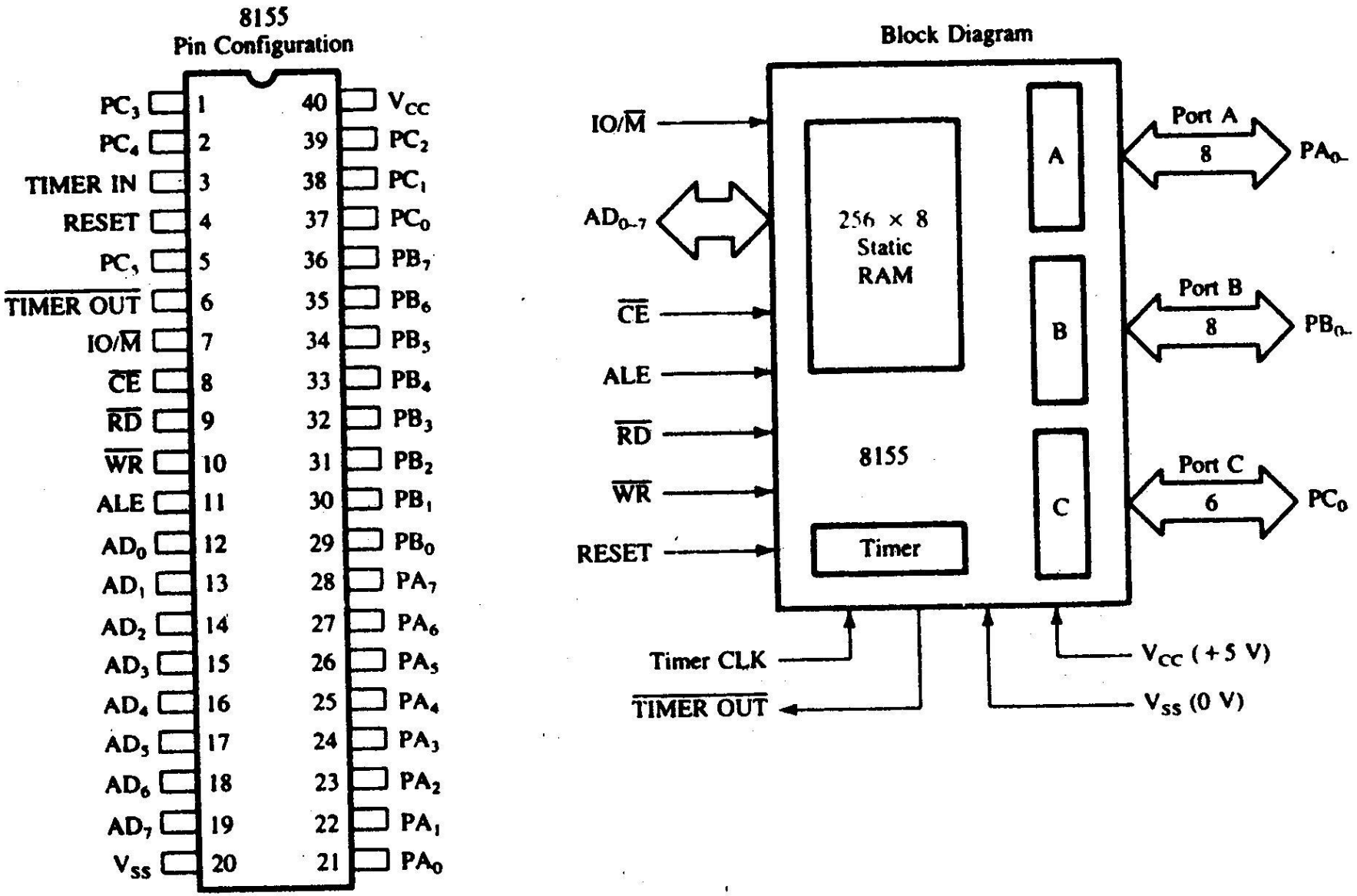


FIGURE 14.5
8155 Pin Configuration and Block Diagram

Source: Intel Corporation, *Embedded Microprocessors* (Santa Clara, Calif.: Author, 1994), pp. 1–31.

14.2 The 8155 Programmable I/O Ports and Timer

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.

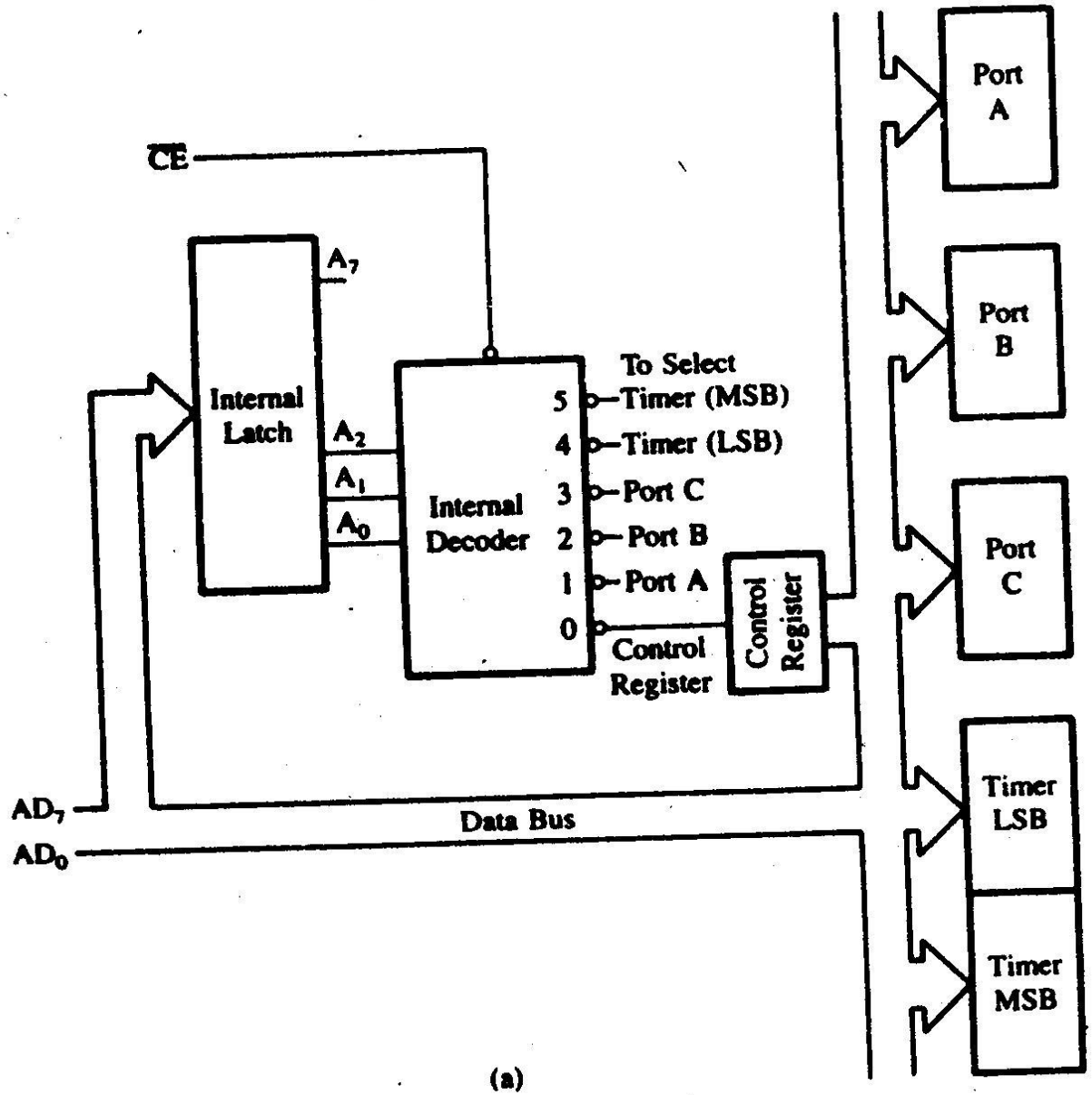
The I/O section includes two 8-bit parallel I/O ports (A and B), one 6-bit port (C), and a timer (Figure 14.5). All the ports can be configured simply as input/output ports. Ports A and B also can be programmed in the handshake mode, each port using three signals from port C. The timer is a 14-bit down-counter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

THE 8155 I/O PORTS

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 AD_0, AD_1, and AD_2.



A_2	A_1	A_0	Ports
0	0	0	Control/Status Register
0	0	1	Port A
0	1	0	Port B
0	1	1	Port C
1	0	0	LSB Timer
1	0	1	MSB Timer

(b)

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

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.

CHIP ENABLE LOGIC AND PORT ADDRESSES

Address lines $AD_2 – AD_0$ also shown as $A_2 – A_0$ after internal demultiplexing, select one of the registers, as shown in Figure 14.6(b). Address lines $A_3 – A_7$ are don't care lines; however, the logic levels on the corresponding high-order lines, $A_{11} – A_{15}$, will be duplicated on lines $A_3 – A_7$.

Example 14.2: Determine the addresses of the control/status register, I/O ports, and timer registers in Figure 14.7.

Solution: To select the chip, the output line of the 8205 (3-to-8) decoder (Figure 14.7) should go low:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}
0	0	1	0	0

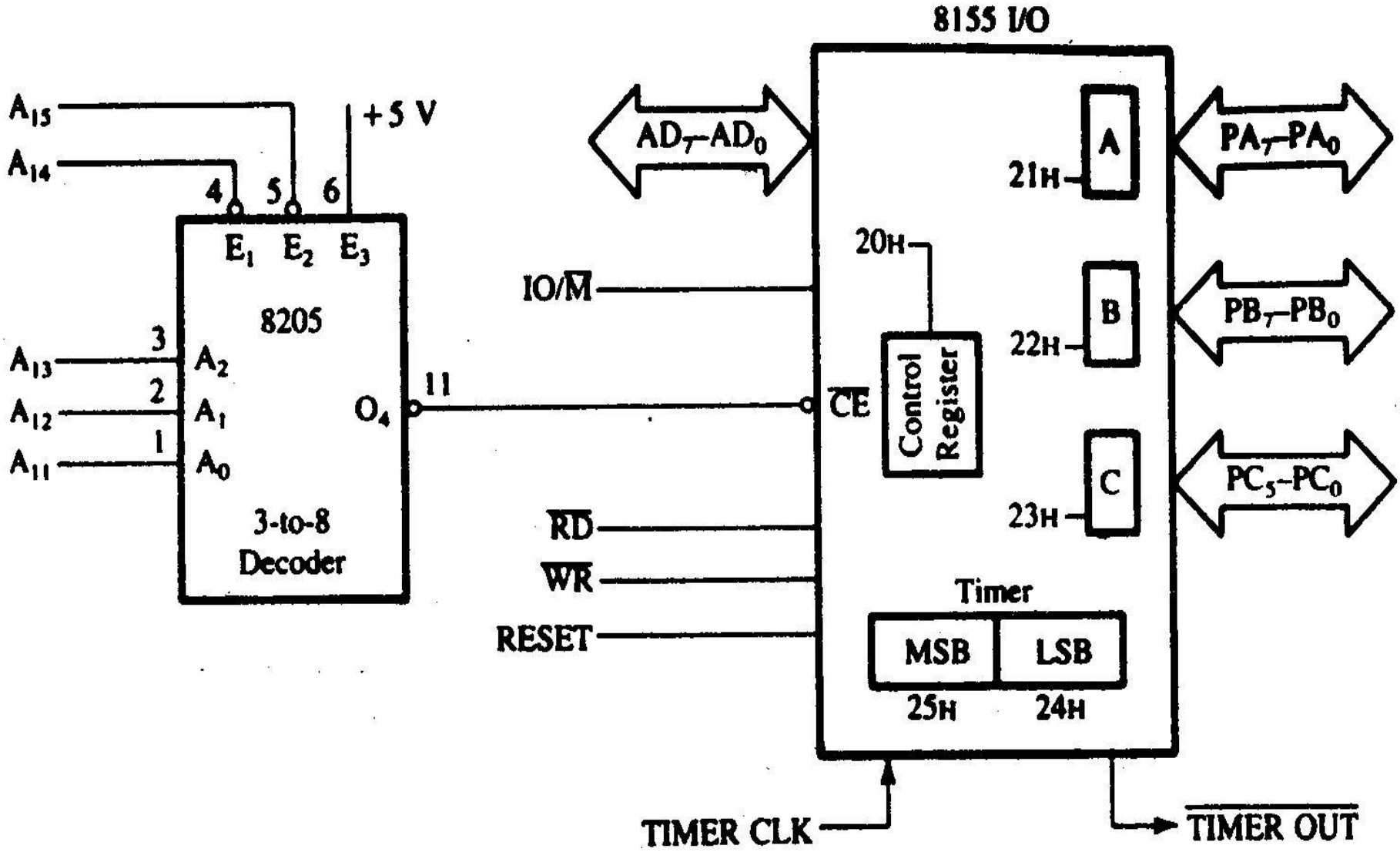


FIGURE 14.7
Interfacing 8155 I/O Ports (Schematic from the SDK-85 System)

By combining five high-order address lines with three low-order address lines (A2 – A0), the port numbers in Figure 14.7 will range from 20H to 25H shown below:

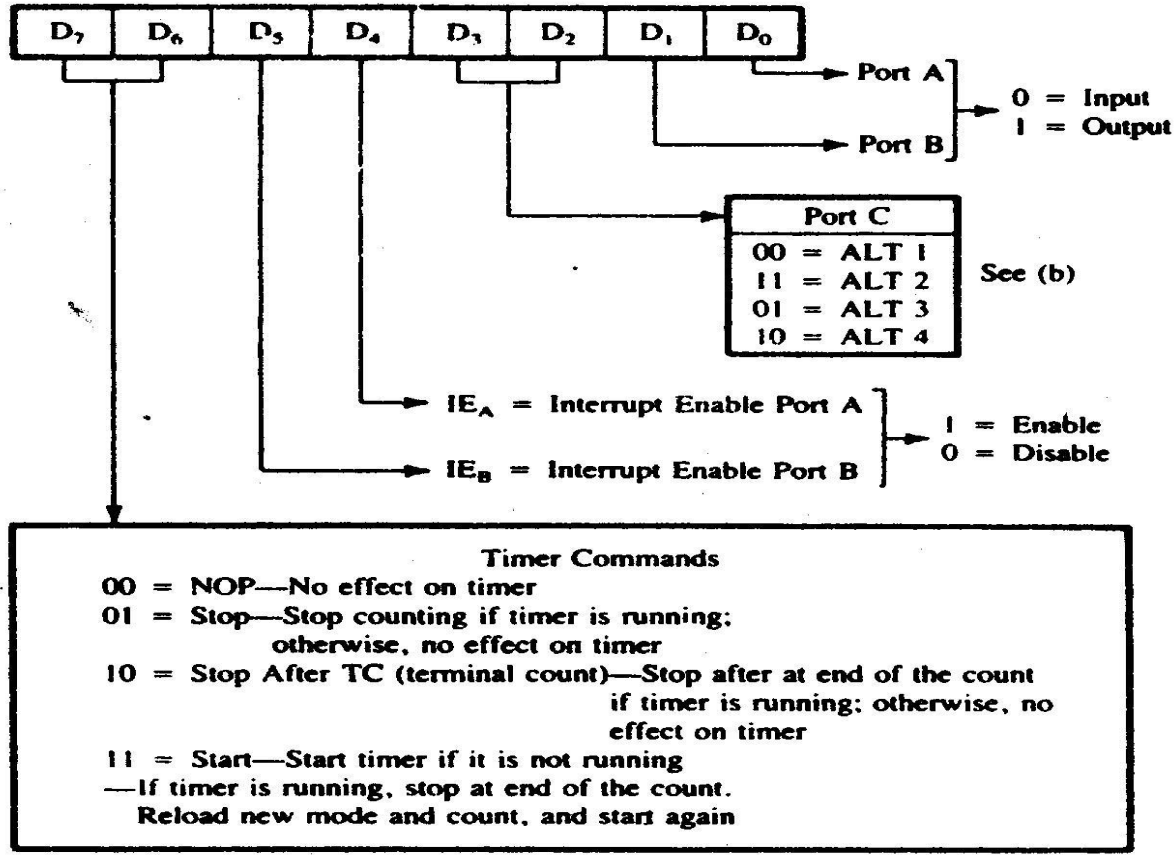
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	AD_2	AD_1	AD_0 = Address	Ports
0	0	1	0	0	0	0	0	= 20H – CR or SR
					0	0	1	= 21H – Port A
					0	1	0	= 22H – Port B
					0	1	1	= 23H – Port C
					1	0	0	=24H–Timer(LSB)
A_7	A_6	A_5	A_4	A_3	1	0	1	=25H–Timer(MSB)

2H

N.B. For I/O-mapped I/O (or port-addressed I/O or peripheral I/O) the low-order address and high-order address bus carry the same information.

CONTROL WORD

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.



(a)

Table: ALT 1–ALT 4: Port C Bit Assignments, Defined by Bits D_3 and D_2 in the Control Register

ALT	D_3	D_2	PC_5	PC_4	PC_3	PC_2	PC_1	PC_0
ALT 1	0	0	I	I	I	I	I	I
ALT 2	1	1	O	O	O	O	O	O
ALT 3	0	1	O	O	O	$\overline{STB}_A$	BF_A	$INTR_A$
ALT 4	1	0	$\overline{STB}_B$	BF_B	$INTR_B$	$\overline{STB}_A$	BF_A	$INTR_A$

I = Input, STB = Strobe, INTR = Interrupt Request
O = Output, BF = Buffer Full, Subscript A = Port A
B = Port B

(b)

FIGURE 14.8
Control Word Definition in the 8155 (a) and Table of Port C Bit Assignments (b)

Bit D_2 and D_3 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 D_4 and D_5 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.

14.2.2 Illustration: Interfacing Sven Segment LED Output Ports Using the 8155

PROBLEM STATEMENT

1. Design two seven-segment-LED displays using ports A and B of the 8155.
2. Write initialization instructions and display data bytes at each port.

HARDWARE DESCRIPTION

The decode logic is the same as that used in the previous discussion; therefore, the port addresses are as follows:

Control Register = 20H
Port A = 21H
Port B = 22H

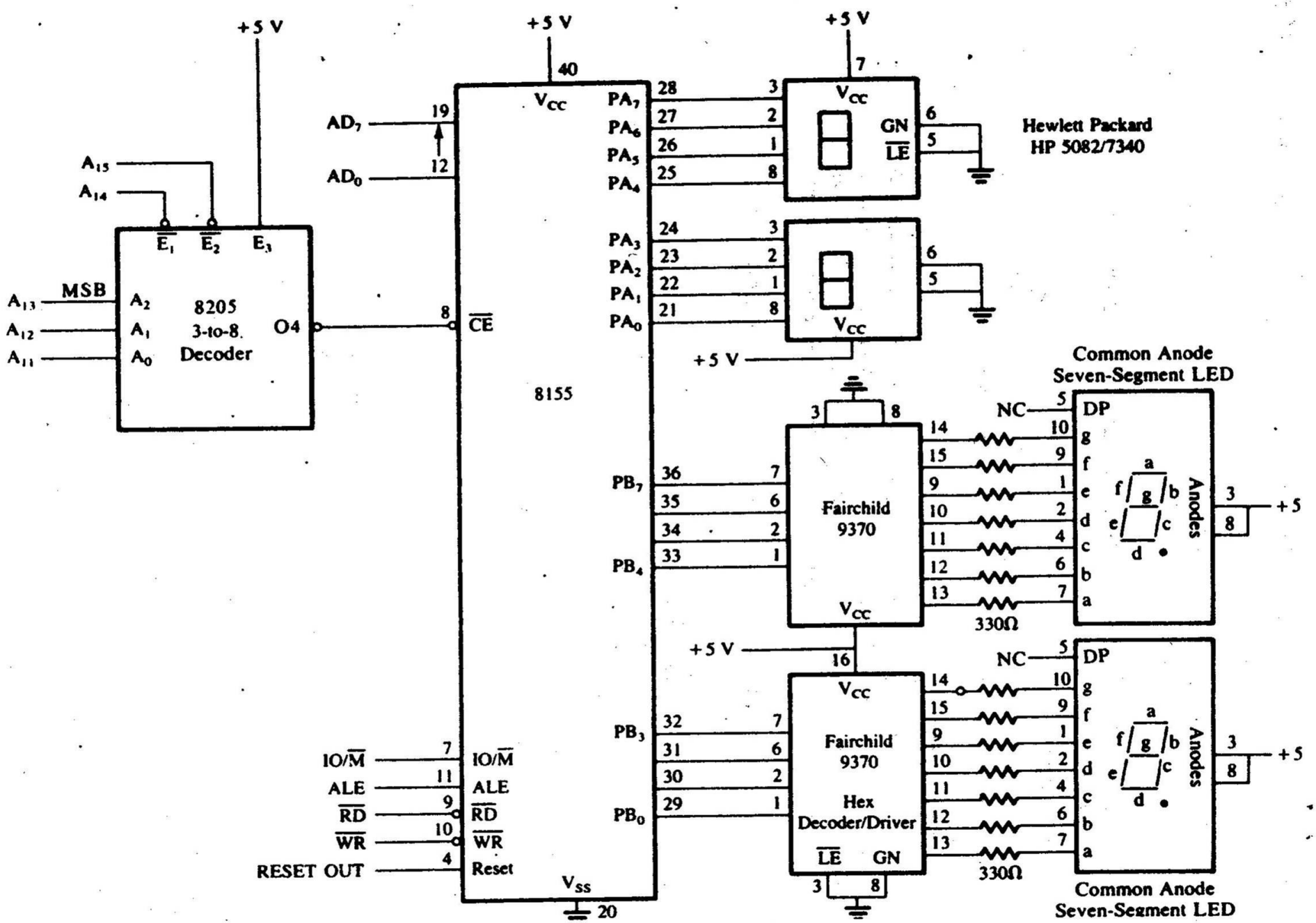


FIGURE 14:9
Interfacing 8155 I/O Ports with Seven-Segment LEDs

CONTROL WORD

To configure ports A and B as outputs, the control word is as follows:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	0	0	1	1	= 03H

PROGRAM

```
MVI A, 03H
OUT 20H
MVI A, BYTE1
OUT 21H
MVI A,BYTE2
OUT 22H
HLT
```

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 in Figure 14.(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).

14.2.4 Illustration: Designing a Square-Wave Generator Using the 8155 Timer

PROBLEM STATEMENT

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. Use the same decode logic and the port addresses as in Example 14.2 (Figure 14.7).

PROBLEM ANALYSIS

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.

Loading Format

M_2	M_1	T_{13}	T_{12}	T_{11}	T_{10}	T_9	T_8

MSB

T_7	T_6	T_5	T_4	T_3	T_2	T_1	T_0

LSB

(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 N/2 N/2
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 N/2 N/2 N/2 N/2
Mode 2: In this mode, a single clock pulse is provided at the end of the count.	1 0	Single Pulse Upon Terminal Count N 1 Timer Clock
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 N 1 Timer Clock

N = Count in Timer

(b)

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

Clock Period = 1/f = 1/3 x 10^6 = 330 ns

Timer Count = Pulse Period / Clock Period = 200 x 10^{-6} / 330 x 10^{-9} = 606

Count = 025EH

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

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

CONTROL WORD

Ports A and B are used as output port; therefore, the control word is:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
1	1	0	0	0	0	1	1	=	C3H

Initialization Instructions

```
MVI A, 5EH
OUT 24H
MVI A,42H
OUT 25H
MVI A, C3H
OUT 20H
HLT
```

14.2.5 The 8155 I/O Ports in Handshake Mode

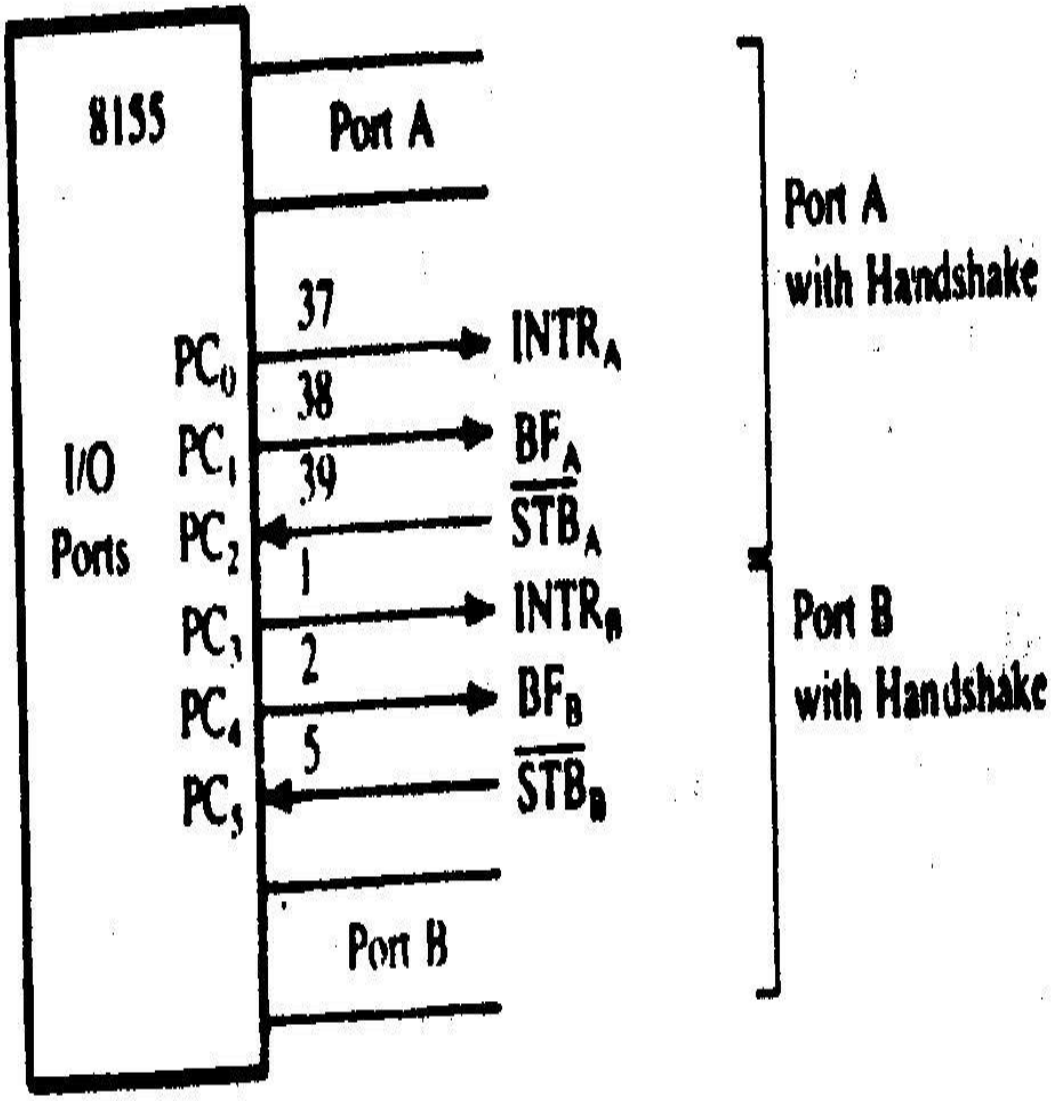
In the handshake mode, data transfer occurs between the MPU and peripherals using control signals called handshake signals. Two I/O ports of the 8155, A and B, can be configured in the handshake mode; each uses three signals from port C as control signals (Figure 14.11). Another alternative (ALT3 in the table in Figure 14.8) available in the 8155 is to configure port A in the handshake mode with three control signals from port C, configure port B as simple I/O, and configure the remaining three bits from port C as outputs.

CONTROL SIGNALS IN HANDSHAKE MODE

When both ports A and B are configured in the handshake mode, port A uses the lower three signals of port C (PC_0, PC_1, and PC_2), and port B uses the upper three signals (PC_3, PC_4, and PC_5), as shown in Figure 14.11. The function of these signals are:

- **STB (Strobe Input):** The low signal on this pin informs the 8155 that data are strobed into the input port.
- **BF (Buffer Full):** The high signal on this pin indicates the presence of a data byte in the port.

FIGURE 14.11
8155 with Handshake Mode



- **INTR (Interrupt Request):** This signal interrupts the processor to read an existing data from its (8155) input port or write a new data to its output port.

- **INTE (Interrupt Enable) :** This is an internal flip-flop used to enable or disable the interrupt capability of the 8155. The interrupts for port A and B are controlled by bits D_4 and D_5 respectively, in the control register.

These control signals can be used to implement either interrupt I/O or status check I/O.

INPUT

Figure 14.12 (a) shows the sequence of events and timing in data input to the 8155.

1.
2.
3.
4.

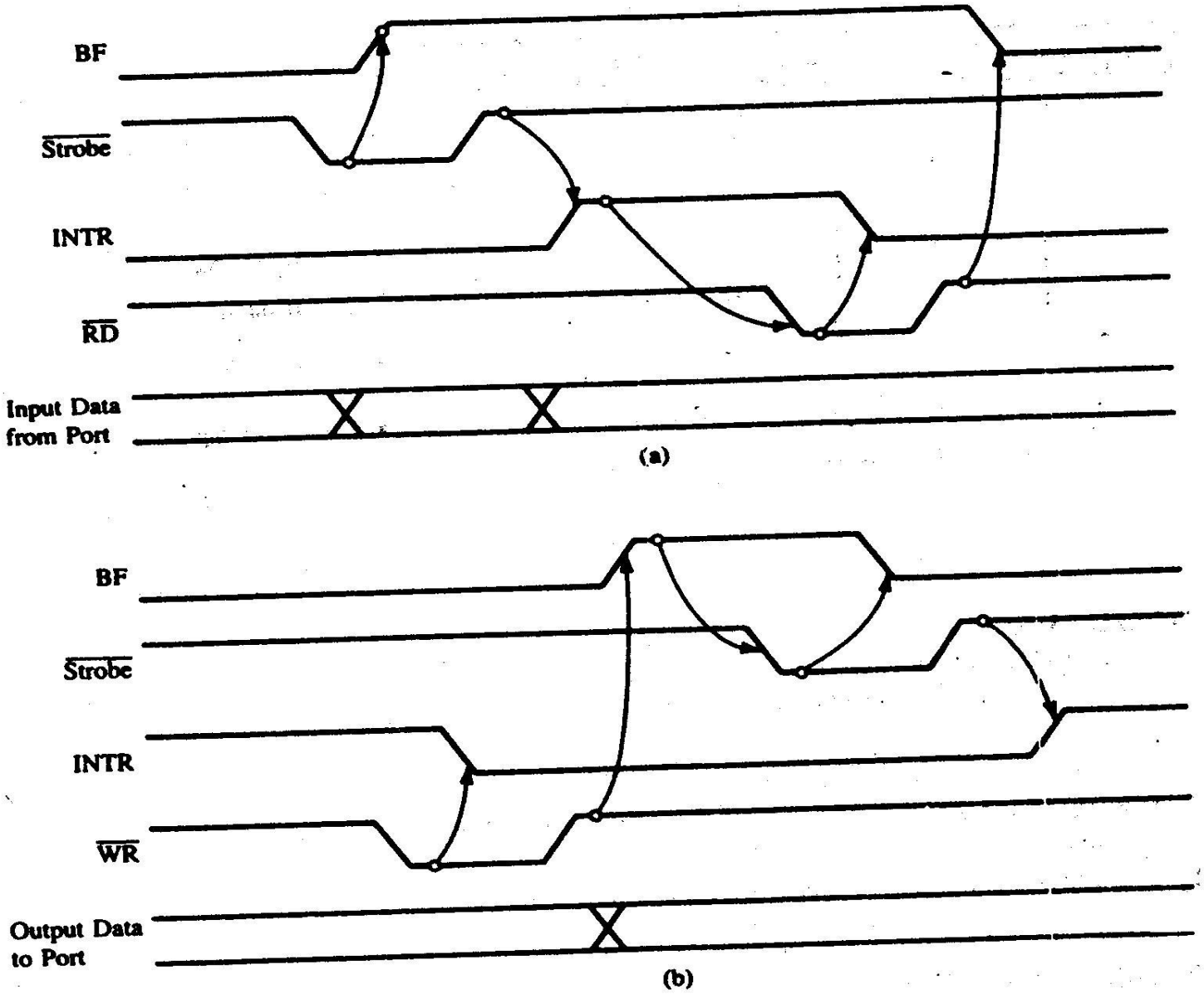


FIGURE 14.12
Timing Waveforms of the 8155 I/O Ports with Handshake: Input Mode (a) and Output Mode (b)

(Santa Clara, Calif.: Author, 1994), pp. 1–43.

OUTPUT

Figure 14.12 (b) shows the sequence of events and timing in data output to the 8155.

1.
2.
3.
4.

STATUS WORD

The MPU can read the status register to check of the timer. The control register and the status register have the same port address; they are differentiated only by RD and WR signals. The status register bits are defined in Figure 14.13.

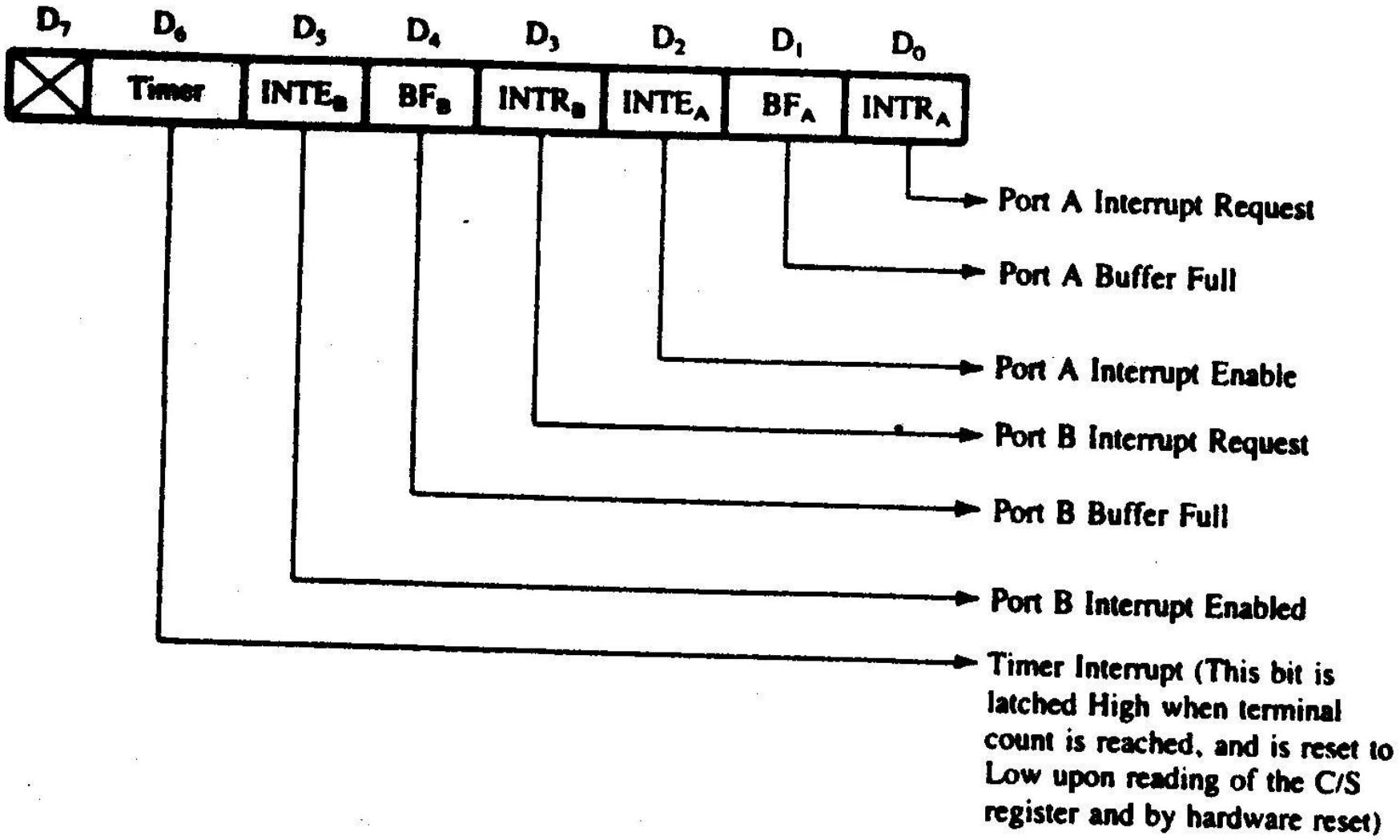


FIGURE 14.13
Status Word Definition

Source: Intel Corporation, *MCS–80/85 Family User's Manual* (Santa Clara, Calif.: Author, 1979), pp. 6–20.

14.2.6 Illustration: Interfacing I/O Ports in Handshake Mode using the 8155

PROBLEM STATEMENT

Design an interfacing circuit using the 8155 read and display form an A/D converter to meet the following requirements:

1. Set up port A in the handshake mode to read data from A/D converter.
2. Set up port B as an output port to display data at seven-segment LEDs.
3. Use line PC_3 from port C to initiate a conversion.

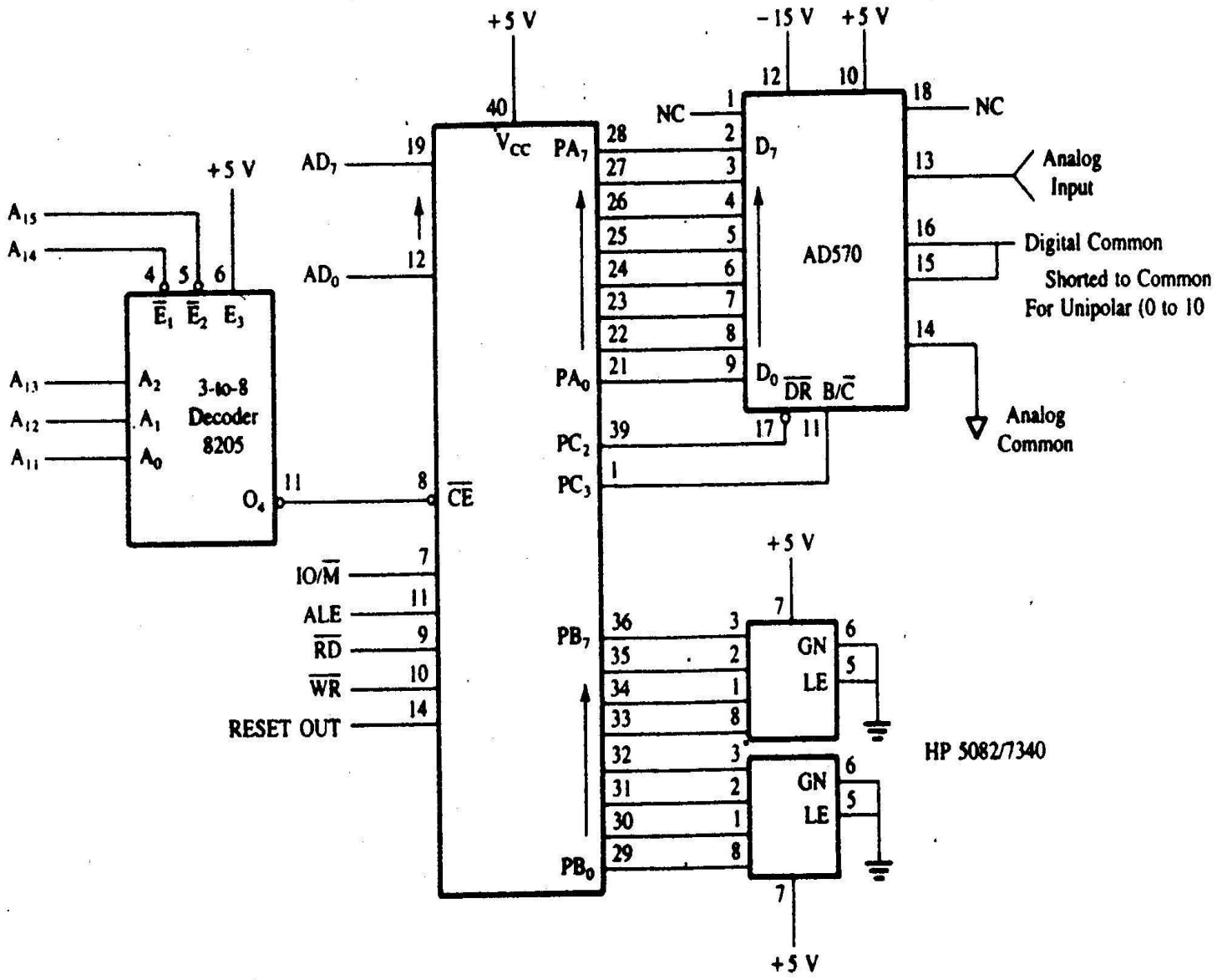


FIGURE 14.14
Interfacing the A/D Converter AD570 in the Handshake Mode

PROBLEM ANALYSIS

The circuit shows that the INTR signal (bit PC_o) is not being used. This suggests that port A is configured for status check and not for interrupt I/O. Therefore, the control word:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
0	0	0	0	0	1	1	0	=	06H

8155 Timer

To start the timer, the control word:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
1	1	0	0	0	1	1	0	=	C6H

To stop the timer, the control word:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
0	1	0	0	0	1	1	0	=	46H

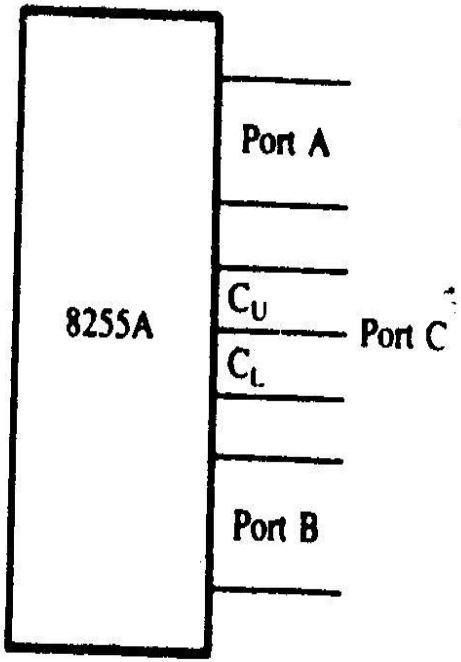
Chapter-15

General Purpose Programmable Peripheral Devices

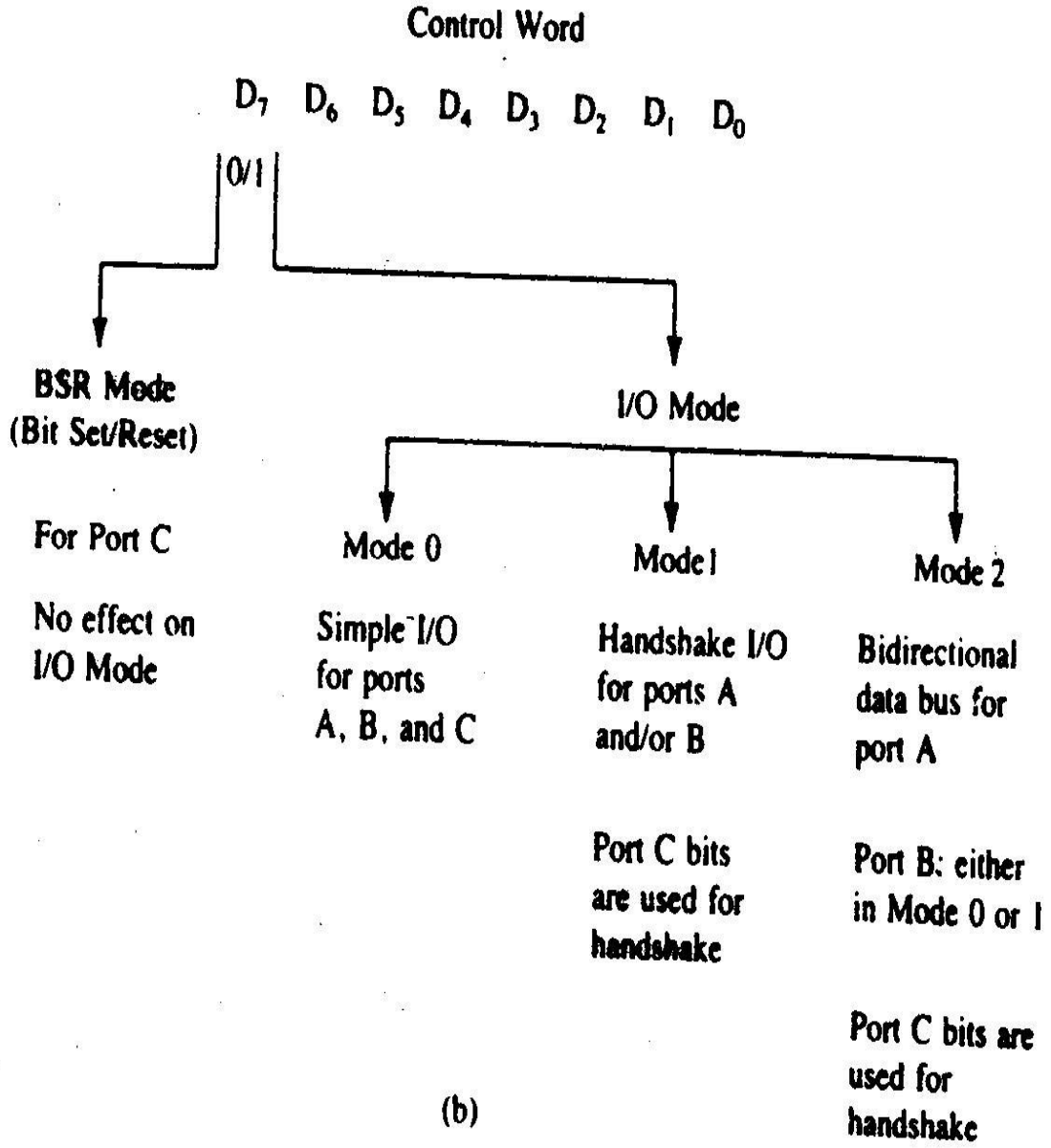
15.1 THE 8255A PROGRAMMABLE PERIHERAL INTERFACE

The 8255A is a widely used, programmable, parallel I/O device. It can be programmed to transfer data under various conditions, from simple I/O to interrupt I/O. It is flexible, versatile, and economical (when multiple I/O ports are required), but somewhat complex. It is an important general-purpose I/O device that can be used with almost any microprocessor.

The 8255A has 24 I/O pins that can be grouped primarily in two 8-bit parallel ports: A and B, with the remaining 8-bits as port C. The eight bits of port C can be used as individual bits or grouped in two 4-bit ports: C_{UPPER} (C_U) and C_{LOWER} (C_L), as in Figure 15.1(a). The functions of these ports are defined by writing control word in the control register.



(a)



(b)

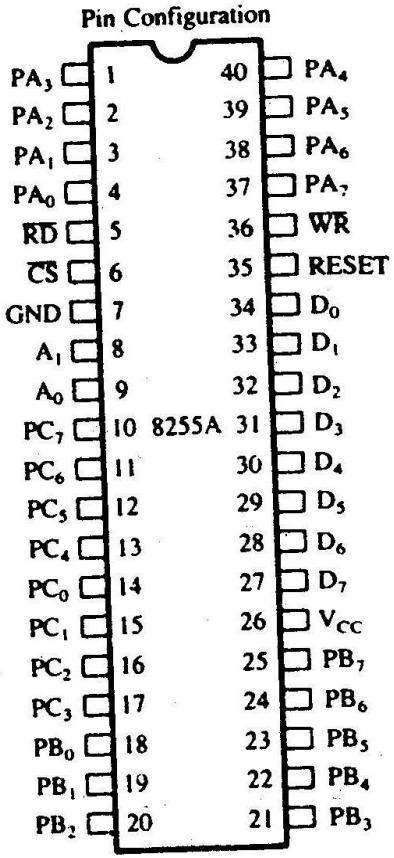
FIGURE 15.1
8255A I/O Ports (a) and Their Modes (b)

Figure 15.1(b) shows all the functions of the 8255A, classified according to two modes: the Bit Set/Reset (BSR) mode and the I/O mode. The BSR mode is used to set or reset the bits in port C. The I/O mode is further divided into three modes: Mode 0, Mode 1, and Mode 2.

- **Mode 0:** all ports function as simple I/O ports.
- **Mode 1:** handshake mode; ports A and/or B use bits from port C as handshake signals.
- **Mode 2:** port A for bidirectional data transfer using handshake signals from port C, and port B can be either Mode 0 or Mode 1.

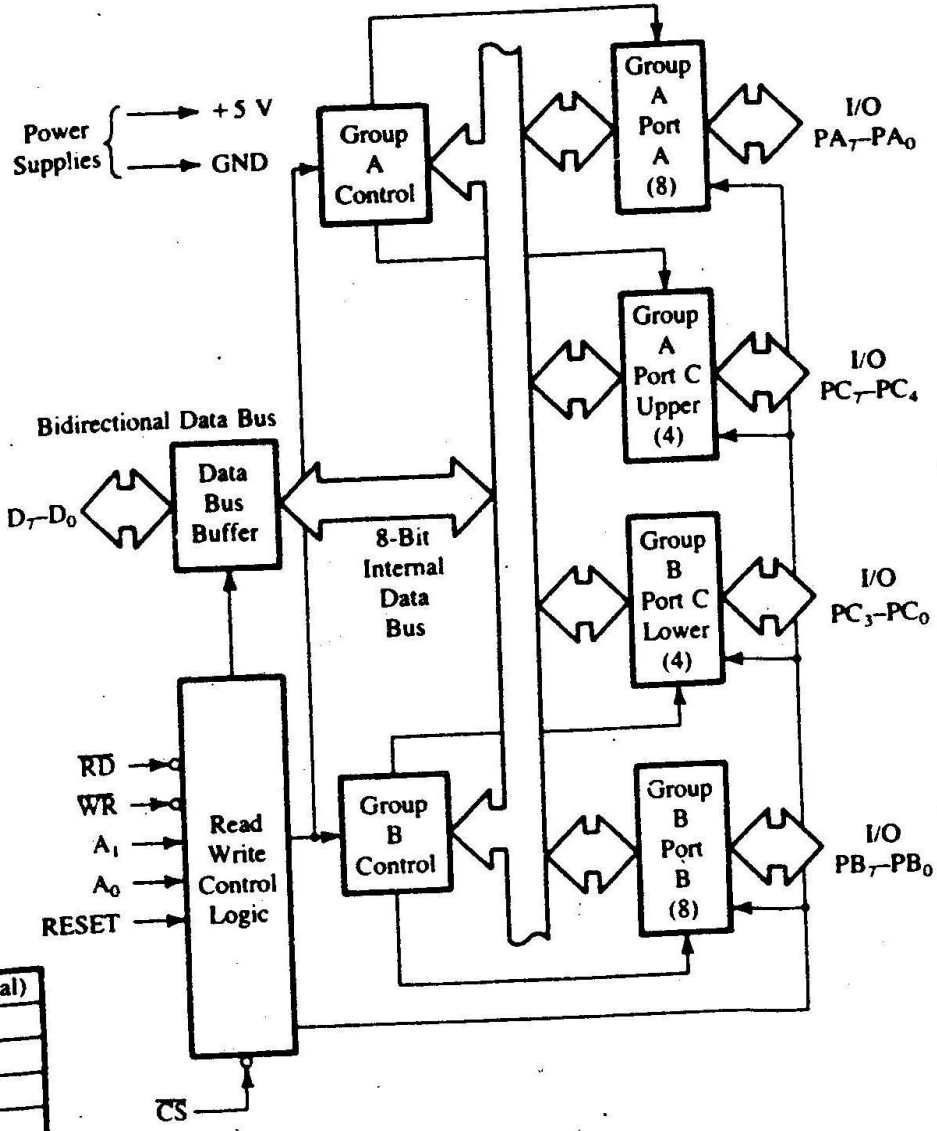
15.1.1 Block Diagram of the 8255A

The block diagram in Figure 15.2(a) shows two 8-bit ports (A and B), two 4-bit ports (C_U and C_L).

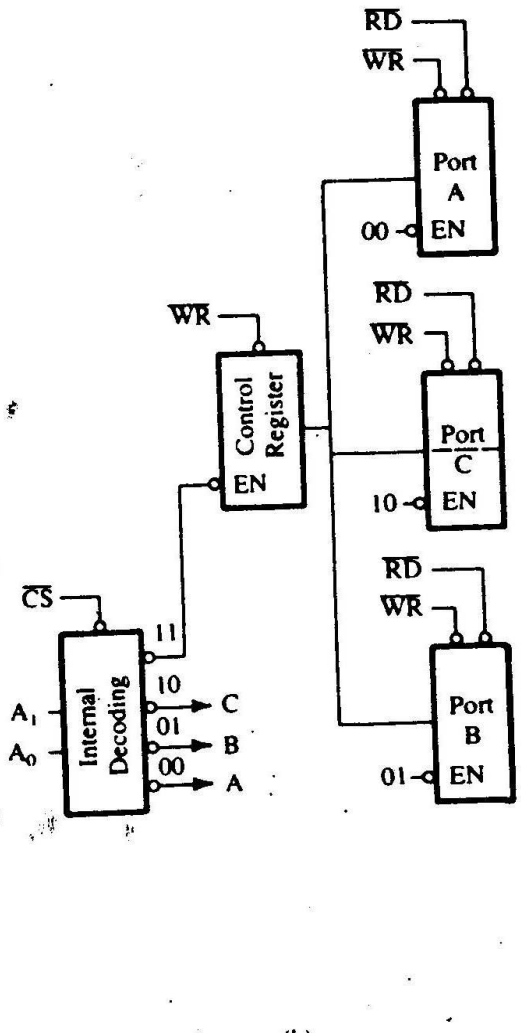


Pin Names

D_7–D_0	Data Bus (Bidirectional)
RESET	Reset Input
$\overline{CS}$	Chip Select
$\overline{RD}$	Read Input
$\overline{WR}$	Write Input
A_0, A_1	Port Address
PA_7–PA_0	Port A (Bit)
PB_7–PB_0	Port B (Bit)
PC_7–PC_0	Port C (Bit)
V_{CC}	+5 Volts
GND	0 Volts

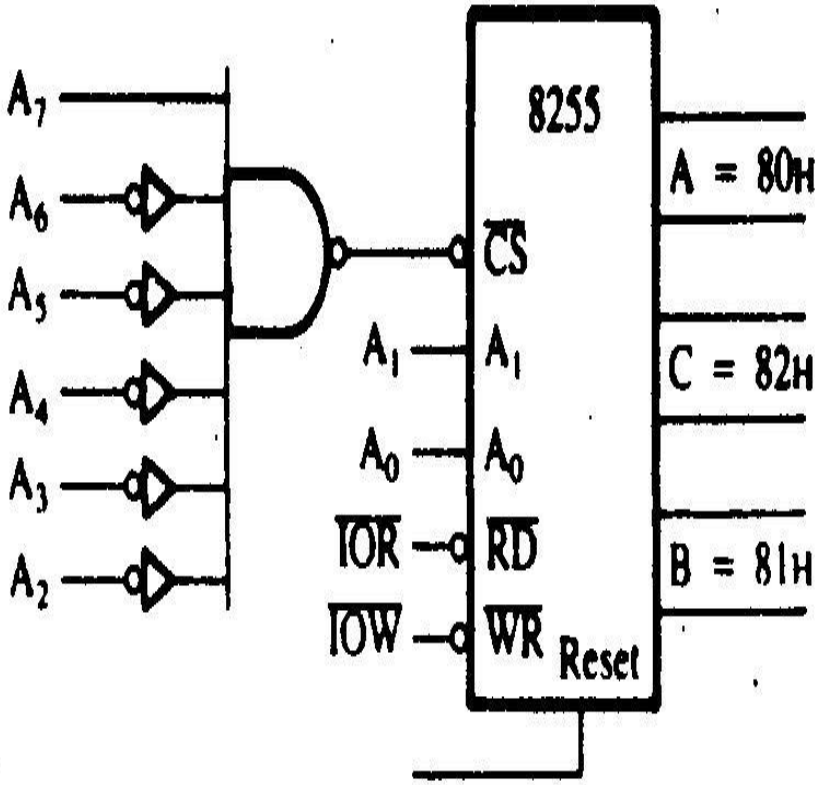


(a)



(b)

FIGURE 15.2 ... Functional Version of the Control Logic and I/O Ports (b)



(a)

$\overline{CS}$		Hex Address	Port
A_7 A_6 A_5 A_4 A_3 A_2	A_1 A_0		
1 0 0 0 0 0	0 0	= 80H	A
	0 1	= 81H	B
	1 0	= 82H	C
	1 1	= 83H	Control Register

(b)

FIGURE 15.3

8255A Chip Select Logic (a) and I/O Port Addresses (b)

CONTROL WORD

The bit D_7 specifies either the I/O function or the Bit Set/Reset function as classified in Figure 15.1(b). Figure 15.4 shows the control word format for I/O Mode.

To communicate with the peripherals through the 8255A, three step are necessary through the 8255A, three steps are necessary:

1. Determine the addresses of ports A, B, and C and of the control register according to the Chip Select logic and A_0 and A_1.
2. Write a control word in the control register.
3. Write I/O instructions to communicate through ports A, B, and C.

15.1.2 Mode 0: Simple Input or Output

In this mode, ports A and B are used as two simple 8-bit I/O ports and port C as two 4-bit ports. Each port (or half-port, in case of C) can be programmed to function as simply an input port or an output port. The input/output features in Mode 0 are as follows:

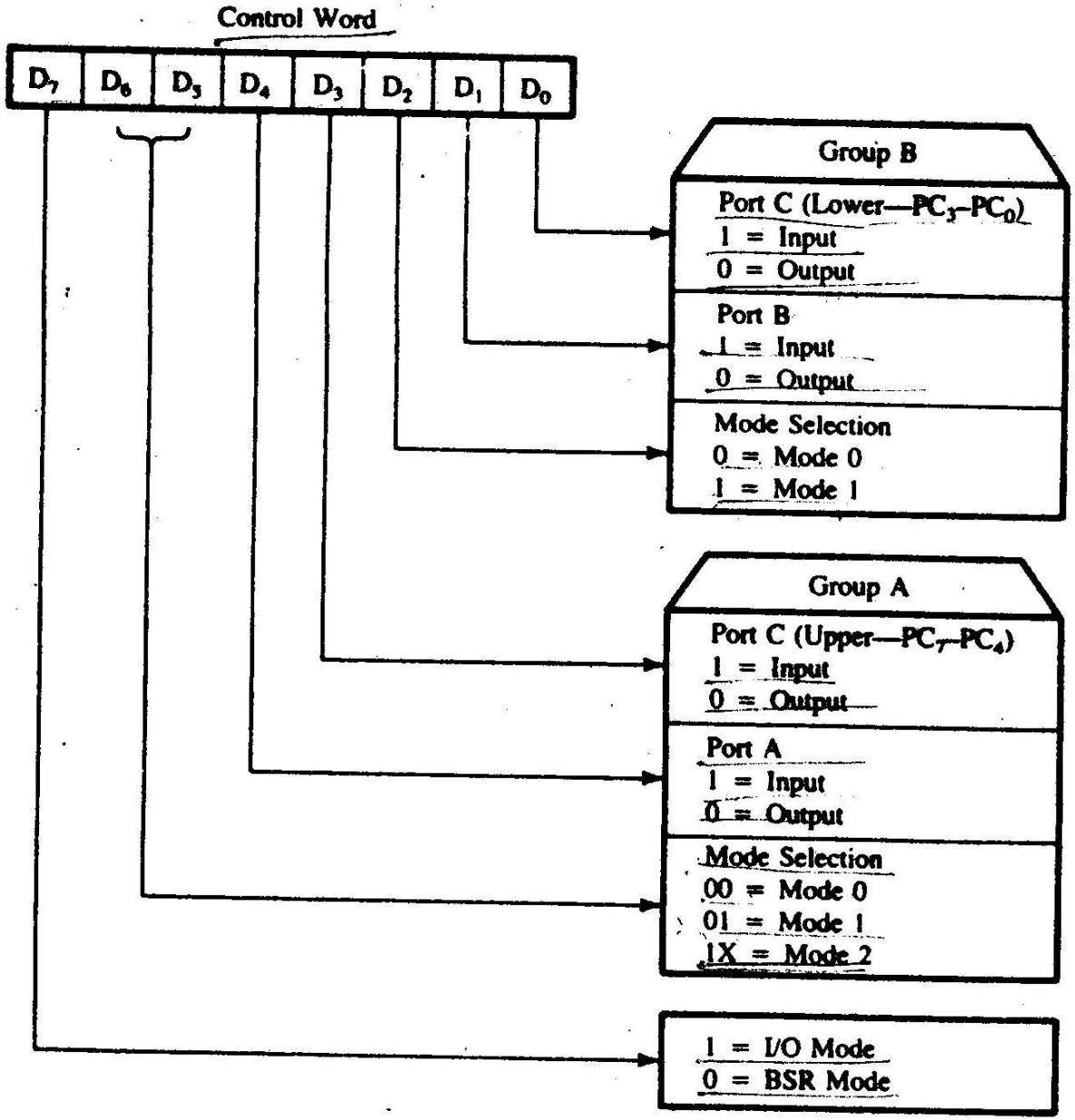


FIGURE 15.4
8255A Control Word Format for I/O Mode

SOURCE: Adapted from Intel Corporation, *Peripheral Components* (Santa Clara, Calif.: Author, 1993), p. 3–104.

1. Outputs are latched.
2. Inputs are not latched.
3. Ports do not have handshake or interrupt capability.

Example 15.1:

1. Identify the port addresses in Figure 15.5.
2. Identify the Mode 0 control word to configure port A and port C_U as output ports and port B and port C_L as input ports.
3. Write a program to read the DIP switches and display the reading from port B port A and from port C_L at port C_U.

Solution:

1. Port Addresses

Port A = 8000H (A_1=0, A_0=0)
Port B = 8001H (A_1=0, A_0=1)
Port C = 8002H (A_1=1, A_0=0)
Control Register = 8003H (A_1=1, A_0=1)

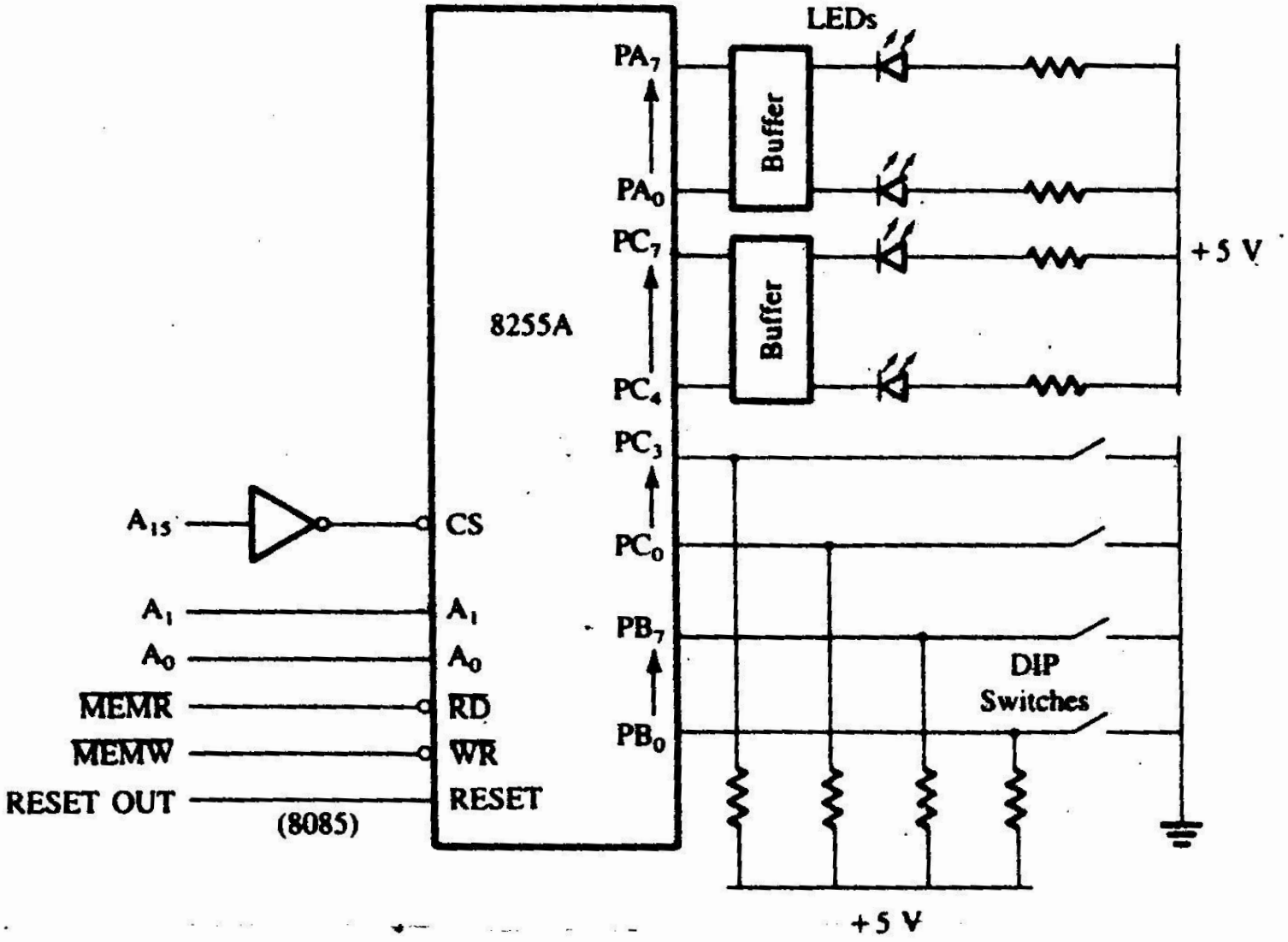


FIGURE 15.5
Interfacing 8255A I/O Ports in Mode 0

2. Control Word

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	0	0	0	1	1	= 83H

3. Program

```
MVI A,83H
STA 8003H
LDA 8001H
STA 8000H
LDA 8002H
ANI 0FH
RLC
RLC
RLC
RLC
STA 8002H
HLT
```

15.1.3 BSR (Bit Set/Reset) Mode

The BSR mode is concerned only with the eight bits of port C, which can be set or reset by writing an appropriate control word in the control register. A control word with bit D_7=0 is recognize as a BSR control word, and it does not alter any previously transmitted control word with bit D_7=1; thus the I/O operations of ports A and B are not affected by a BSR control word. In the BSR mode, individual bits of port C can be used for applications such as an on/off switch.

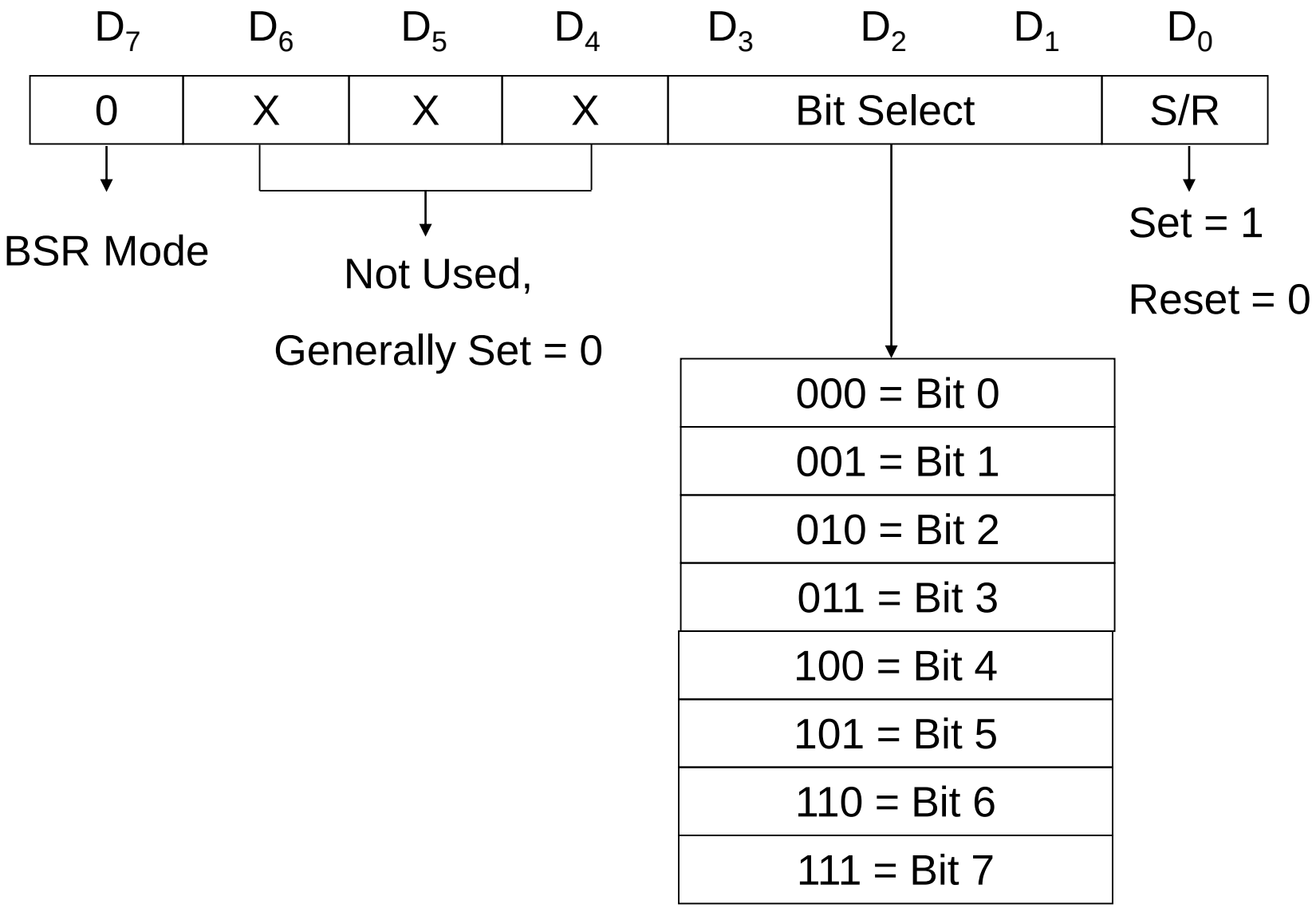
BSR CONTROL WORD

This control word, when written in the control register, sets or resets one bit at a time, as specified in Figure 15.6.

Example 15.2 :

Write BSR control word subroutine to set bits PC_7 and PC_3 and reset them after 10 ms. Use the schematic in Figure 15.3 and assume that a delay subroutine is available.

Figure 15.6 : 8255A Control Word Format in the BSR Mode



BSR Control Words

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
To set bit PC_7 =	0	0	0	0	1	1	1	1	= 0FH
To reset bit PC_7 =	0	0	0	0	1	1	1	0	= 0EH
To set bit PC_3 =	0	0	0	0	0	1	1	1	= 07H
To reset bit PC_3 =	0	0	0	0	0	1	1	0	= 06H

PORT ADDRESS

Control register address = 83H; refer to Figure 15.3(b).

SUBROUTINE

```
BSR:    MVI A, 0FH
        OUT 83H
        MVI A,07H
        OUT 83H
        CALL DELAY
        MVI A,06H
        OUT 83H
        MVI A,0EH
        OUT 83H
        RET
```

15.4 THE 8254 (8253) PROGRAMMABLE INTERVAL TIMER

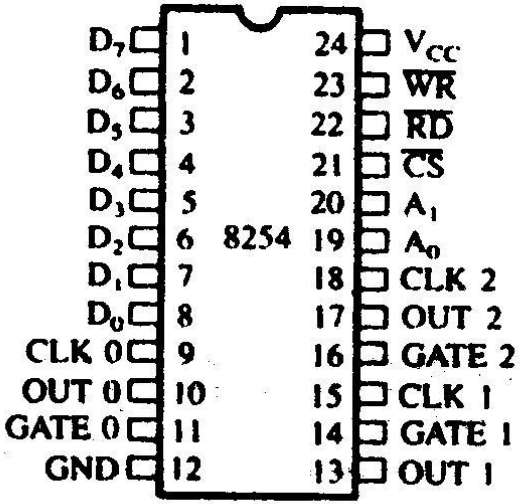
The 8254 programmable interval timer/counter is functionally similar to the software designed counters and timers described in chapter 8. It generates accurate time delay and can be used for applications such as a real-time clock, an event counter, a square-wave generator, and a complex waveform generator.

The 8254 includes three identical 16-bit counters that can operate independently in any one of the six modes. It is packaged in a 24-pin DIP and requires a single +5v power supply. To operate a counter, a 16-bit count is loaded in its register and, on command, begins to decrement the count until it reaches 0. At the end of the count, it generates a pulse that can be used to interrupt the MPU. The counter can count in binary or BCD. In addition, count can be read by the MPU while counter is decrementing.

15.4.1 Block Diagram of the 8254

Figure 15.23 is the block diagram of the 8254; it includes three counters (0, 1, 1 and 2), a data bus buffer, Read/Write control logic, and a control register. Each counter has two input signals-Clock (CLK) and GATE-and output signal-OUT.

Pin Configuration



Pin Names

D_7–D_0	Data Bus (8 Bit)
CLK N	Counter Clock Inputs
GATE N	Counter Gate Inputs
OUT N	Counter Outputs
$\overline{RD}$	Read Counter
$\overline{WR}$	Write Command or Data
$\overline{CS}$	Chip Select
A_0–A_1	Counter Select
V_{CC}	+5 Volts
GND	Ground

Block Diagram

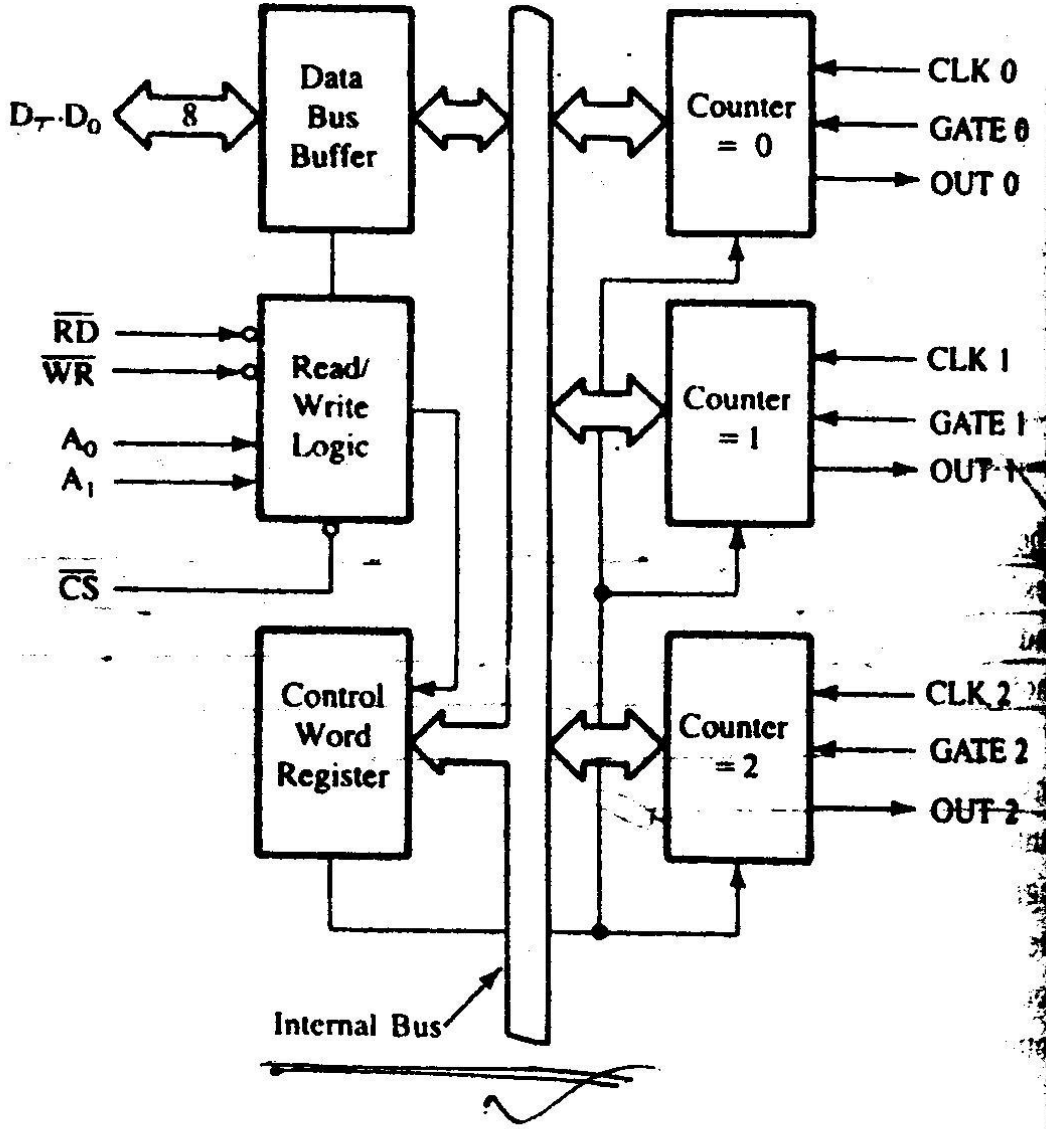


FIGURE 15.23
8254 Block Diagram

SOURCE: Intel Corporation, *Peripheral Components* (Santa Clara, Calif.: Author, 1993), p. 3–62.

CONTROL LOGIC

A_0	A_1	Selection
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Register

CONTROL WORD REGISTER

The control word format is shown in Figure 15.24.

MODE

8254 Control Word Format

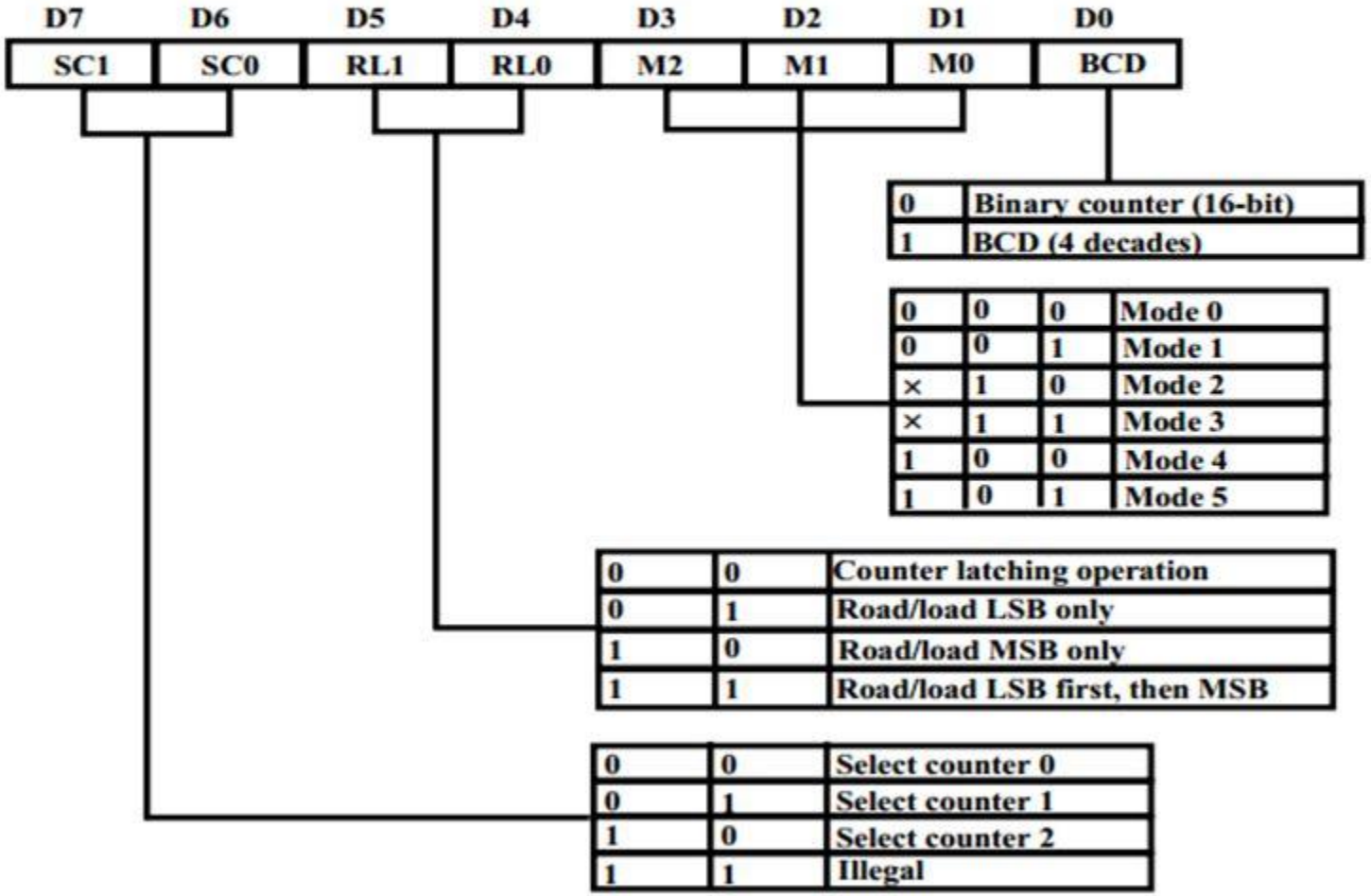


Fig. 14.9 Control Register