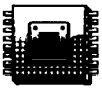
5 Basic Peripherals and Their Interfacing with 8086/88



INTRODUCTION

In the previous chapters, we have presented the architecture, instruction set and the art of programming with 8086/88. In this chapter, the general peripheral devices and their interfacing techniques with the microprocessor 8086/88 are discussed. In the minimal working system configuration of a general microprocessor, we consider a keyboard, display system, memory system and I/O ports along with the CPU. In general, all these devices are called peripheral devices. There are also some additional dedicated peripheral devices like PIC [Programmable Interrupt Controller], DMA [Direct Memory Access] controller, CRT controller, etc. All these dedicated peripherals are studied in the next chapter. The microprocessor may be seen as the heart of the system while all the peripheral circuits including memory system are built around the microprocessor. Since a processor without memory is not meaningful, memory (primary) may also be considered as an integral part of a microprocessor system.

Most of the peripheral devices are designed and interfaced with a CPU either to enable it to communicate with the user or an external process and to ease the circuit operations so that the microprocessor works more efficiently and effectively. The use of a special purpose peripheral integrated device simplifies both—the hardware circuits and the software, considerably. Each of these special purpose devices need a typical sequence of instructions to make it work. This instruction sequence appropriately initialises the peripheral and makes it work under the control of the microprocessor. Thus each dedicated peripheral device needs suitable initialisation. However memory, unlike the peripheral devices, does not need any initialisation and does not directly participate in the process of communication between CPU and the user. Rather, it acts as a media for the communication between a peripheral with the microprocessor. While memory may be treated as a peripheral, on the other hand, peripheral devices are often treated as memory locations. Thus as far as the CPU is concerned, there is no special difference in the method of handling memory or peripherals, except for the use of respective control signals. In this chapter, we will present interfacing techniques of semiconductor memories, I/O ports and a few other peripherals with 8086/8088.

5.1 SEMICONDUCTOR MEMORY INTERFACING

Semiconductor memories are of two types, viz. RAM (Random Access Memory) and ROM (Read Only Memory).

5.1.1 Static RAM Interfacing

The semiconductor RAMs are of broadly two types–static RAM and dynamic RAM. In this section, we will consider the interfacing of static RAM and ROM with 8086/8088. The semiconductor memories are organised as two dimensional arrays of memory locations. For example, 4K × 8 or 4K byte memory contains 4096 locations, where each location contains 8-bit data and only one of the 4096 locations can be selected at a time. Once a location is selected all the bits in it are accessible using a group of conductors called 'data bus'. Obviously, for addressing 4K bytes of memory, twelve address lines are required. In general, to address a memory location out of N memory locations , we will require at least n bits of address, i.e. n address lines where $n = \text{Log}_2 N$. Thus if the microprocessor has n address lines, then it is able to address at the most N locations of memory, where $2^n = N$. However, if out of N locations only P memory locations are to be interfaced, then the least significant p address lines out of the available n lines can be directly connected from the microprocessor to the memory chip while the remaining $(n–p)$ higher order address lines may be used for address decoding (as inputs to the chip selection logic). The memory address depends upon the hardware circuit used for decoding the chip select ($\overline{\text{CS}}$). The output of the decoding circuit is connected with the $\overline{\text{CS}}$ pin of the memory chip.

The general procedure of static memory interfacing with 8086 is briefly described as follows:

1. Arrange the available memory chips so as to obtain 16-bit data bus width. The upper 8-bit bank is called 'odd address memory bank' and the lower 8-bit bank is called 'even address memory bank', as described in memory organisation in Chapter 1.
2. Connect available memory address lines of memory chips with those of the microprocessor and also connect the memory $\overline{\text{RD}}$ and $\overline{\text{WR}}$ inputs to the corresponding processor control signals. Connect the 16-bit data bus of the memory bank with that of the microprocessor 8086.
3. The remaining address lines of the microprocessor, $\overline{\text{BHE}}$ and A_0 are used for decoding the required chip select signals for the odd and even memory banks. The $\overline{\text{CS}}$ of memory is derived from the O/P of the decoding circuit.

The procedure will be clearer while solving problems on memory interfacing with 8086/88.

As a good and efficient interfacing practice, the address map of the system should be continuous as far as possible, i.e. there should be no windows in the map and no fold back space should be allowed. A memory location should have a single address corresponding to it, i.e. absolute decoding should be preferred, and minimum hardware should be used for decoding. In a number of cases, linear decoding may be used to minimise the required hardware.

Let us now consider a few example problems on memory interfacing with 8086.

Program 5.1

Interface two 4K × 8 EPROMS and two 4K × 8 RAM chips with 8086. Select suitable maps.

Solution We know that, after reset, the IP and CS are initialised to form address FFFF0H. Hence, this address must lie in the EPROM. The address of RAM may be selected any where in the 1MB address space of 8086, but we will select the RAM address such that the address map of the system is continuous, as shown in Table 5.1.

Table 5.1 *Memory Map for Problem 5.1*

Address	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}
FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
						EPROM							8K × 8							

(*Contd.*)

Table 5.1 (*Contd.*)

Address	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}
FE000H	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
FDFFFH	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
						RAM						8K × 8								
FC000H	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

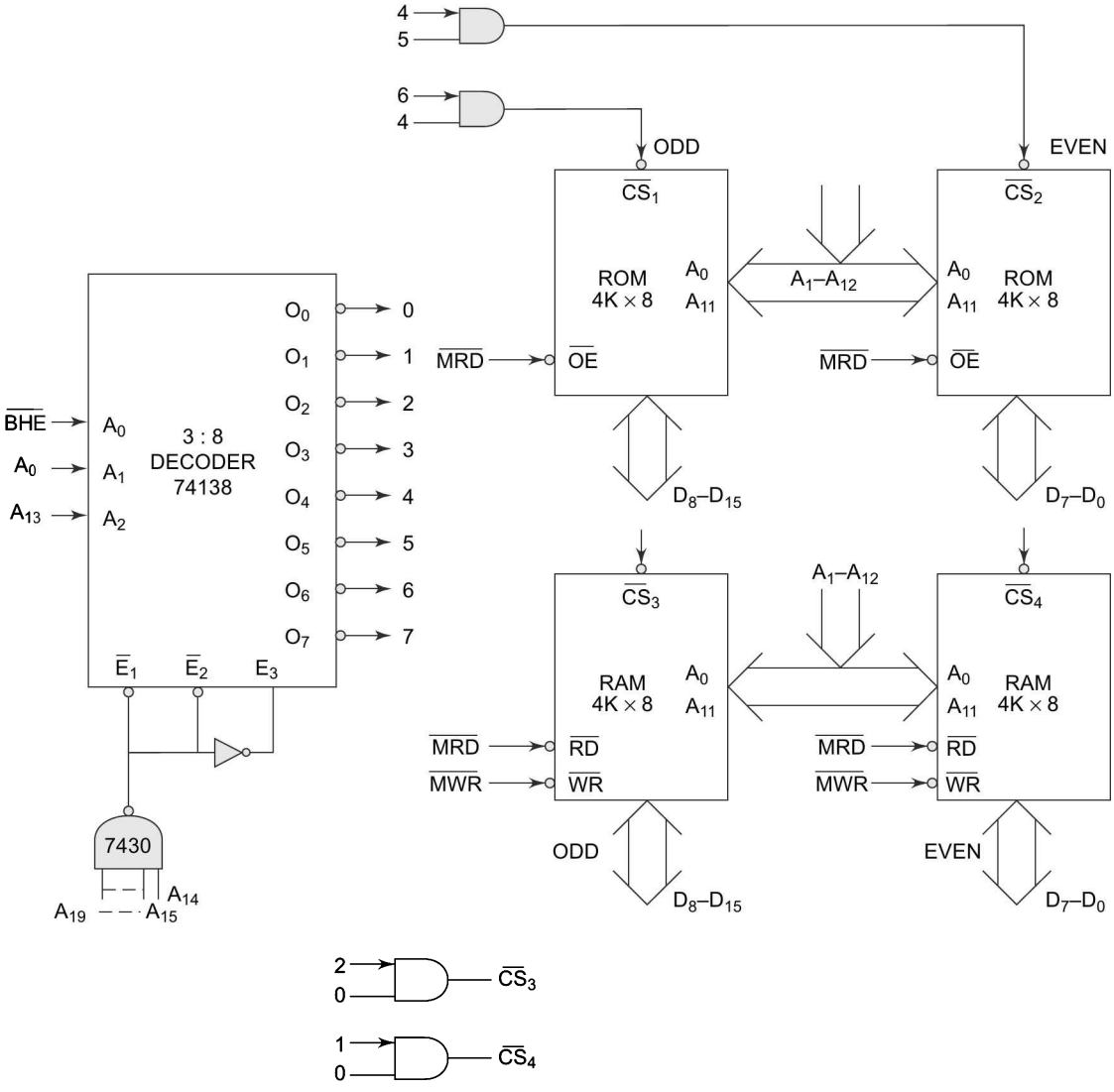


Fig. 5.1 *Interfacing Problem 5.1*

Total 8K bytes of EPROM need 13 address lines A_0–A_{12} (since $2^{13} = 8K$). Address lines A_{13}–A_{19} are used for decoding to generate the chip select. The $\overline{BHE}$ signal goes low when a transfer is at odd address or higher byte of data is to be accessed. Let us assume that the latched address, $\overline{BHE}$ and demultiplexed data lines are readily available for interfacing. Figure 5.1 shows the interfacing diagram for the memory system.

The memory system in this example contains in total four 4K × 8 memory chips.

The two 4K × 8 chips of RAM and ROM are arranged in parallel to obtain 16-bit data bus width. If A_0 is 0, i.e. the address is even and is in RAM, then the lower RAM chip is selected indicating 8-bit transfer at an even address. If A_0 is 1, i.e. the address is odd and is in RAM, the $\overline{BHE}$ goes low, the upper RAM chip is selected, further indicating that the 8-bit transfer is at an odd address. If the selected addresses are in ROM, the respective ROM chips are selected. If at a time A_0 and $\overline{BHE}$ both are 0, both the RAM or ROM chips are selected, i.e. the data transfer is of 16 bits. The selection of chips here takes place as shown in Table 5.2.

Table 5.2 *Memory Chip Selection for Problem 5.1*

Decoder I/P → *Address/* $\overline{BHE}$ →	A_2 A_{13}	A_1 A_0	A_0 $\overline{BHE}$	*Selection/ Comment*
Word transfer on $D_0 - D_{15}$	0	0	0	Even and odd addresses in RAM
Byte transfer on $D_7 - D_0$	0	0	1	Only even address in RAM
Byte transfer on $D_8 - D_{15}$	0	1	0	Only odd address in RAM
Word transfer on $D_0 - D_{15}$	1	0	0	Even and odd addresses in ROM
Byte transfer on $D_0 - D_7$	1	0	1	Only even address in ROM
Byte transfer on $D_8 - D_{15}$	1	1	0	Only odd address in ROM

Program 5.2

Design an interface between 8086 CPU and two chips of 16K × 8 EPROM and two chips of 32K × 8 RAM. Select the starting address of EPROM suitably. The RAM address must start at 00000H.

Solution The last address in the map of 8086 is FFFFFH. After resetting, the processor starts from FFFF0H. Hence this address must lie in the address range of EPROM. Figure 5.2 shows the interfacing diagram, and Table 5.3 shows complete map of the system.

Table 5.3 *Address Map for Problem 5.2*

Addresses	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}
FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
								32KB			EPROM									
F8000H	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0FFFFH	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
								64KB RAM												
00000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

It is better not to use a decoder to implement the above map because it is not continuous, i.e. there is some unused address space between the last RAM address (0FFFFH) and the first EPROM address (F8000H). Hence the logic is implemented using logic gates, as shown in Fig. 5.2.

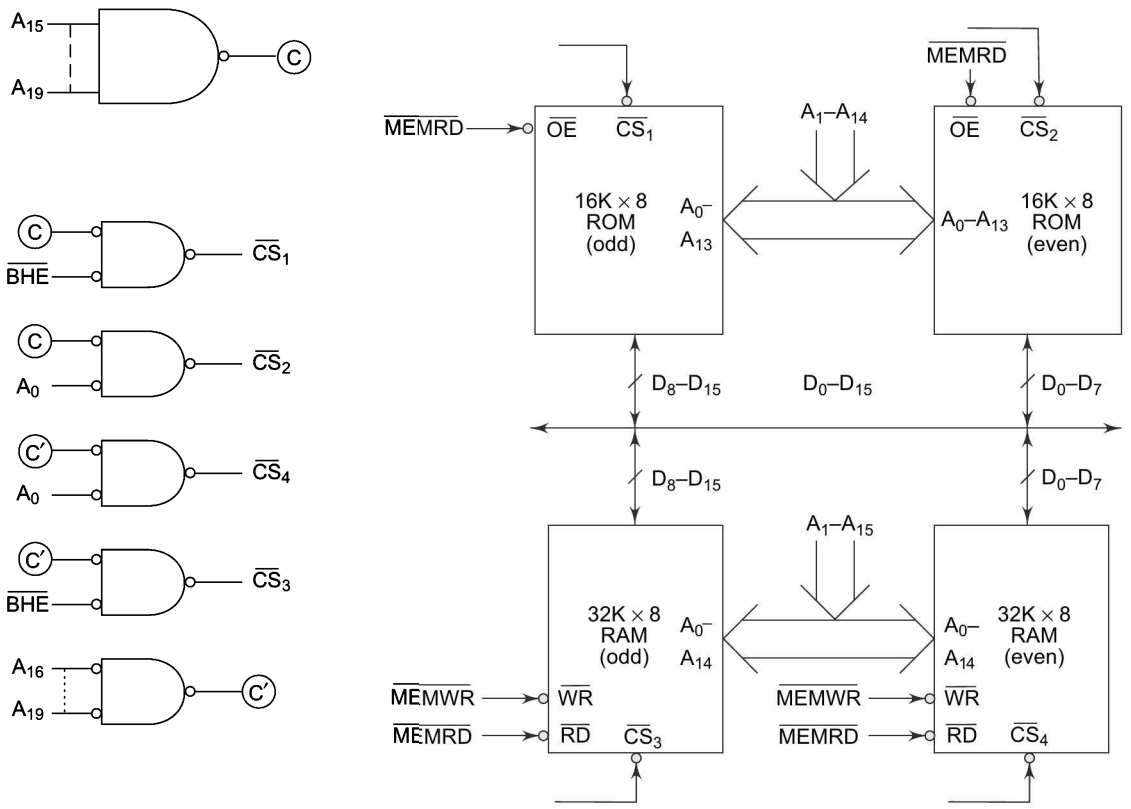


Fig. 5.2 *Interfacing Problem 5.2*

Let us select a variable C for memory address pulse, i.e. output of 6-input NAND gate. $\overline{BHE}$ is abbreviated as B. The chip selection logic can be designed as shown in Table 5.4.

Table 5.4

I/P		*O/P*	
A_0	$B\,(\overline{BHE})$	C_1	C_2
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

$C_2 = A_0$

$C_1 = \overline{BHE}$

To find out $\overline{CS}_1$ and $\overline{CS}_2$ we will have to combine C_1 and C_2 with C.

Table 5.5

I/P			*O/P*	
C_1	C_2	C	$\overline{CS_1}$	$\overline{CS_2}$
0	0	0	0	0
1	0	0	1	0
1	1	0	1	1
0	1	0	0	1

Table 5.5 shows that

$$\overline{CS_1} = C + C_1 = C + \overline{BHE} \text{ and } \overline{CS_2} = C + C_2 = C + A_0$$

Similarly we can find out CS_3 and CS_4.

Problem 5.3

It is required to interface two chips of 32K × 8 ROM and four chips of 32K × 8 RAM with 8086, according to the following map.

ROM 1 and 2 F0000H - FFFFFH, RAM 1 and 2 D0000H - DFFFFH

RAM 3 and 4 E0000H - EFFFFH

Show the implementation of this memory system.

Solution Let us write the memory map of the system as shown in Table 5.6.

The implementation of the above map is shown in Fig. 5.3 using the same technique as in Problem 5.1 and Problem 5.2. All the address, data and control signals are assumed to be readily available.

Table 5.6 *Address Map for Problem 5.3*

Address	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_0
F0000H	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ROM 1and2										64K										
FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
D0000H	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RAM 1and2										64K										
DFFFFH	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
E0000H	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RAM 3and4										64K										
EFFFFH	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5.1.2 Interfacing Memories with 8088

The techniques of interfacing semiconductor memory with 8088 is exactly like that of 8085. The processor 8088 externally has 8-bit data bus just like 8085. Also the 8088 memory system is not divided into even or odd memory banks. Unlike 8086, the memory in an 8088 system can be seen as a continuous block of memory locations. The control bus of 8088 is similar to 8085. Rather, 8088 was designed to be software compatible with 8086 but to work in the circuits which were designed around 8085.The following problems explain the memory interfacing techniques with 8088.

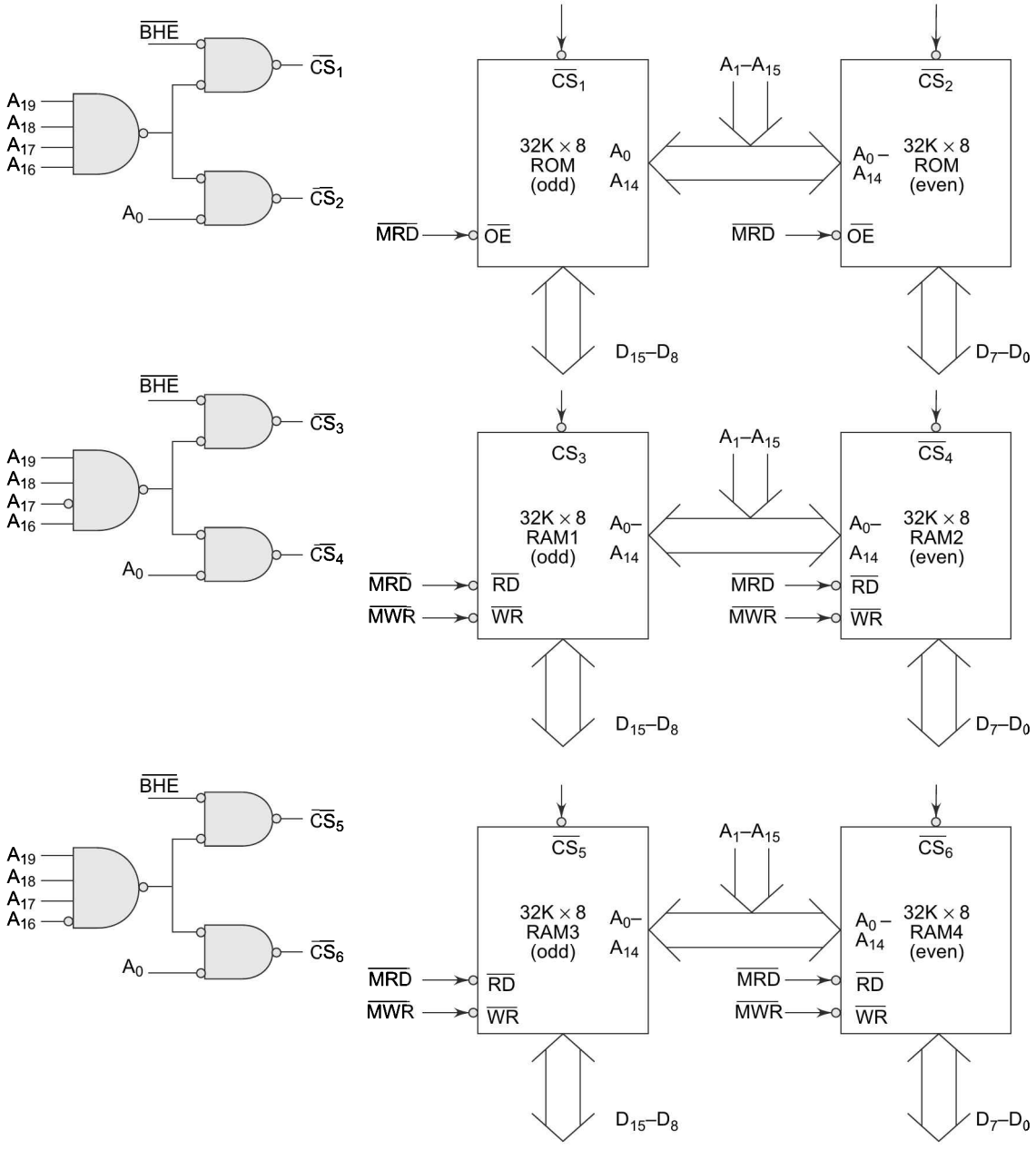


Fig. 5.3 *Interfacing Problem 5.3*

Problem 5.4

Design a memory system around 8088, that has total 16K × 8 EPROM and 32K × 8 RAM. The EPROM chips are available in modules of 8K × 8 and the RAM chips are available in modules of 8K × 8. The memory map should be specified below:

EPROM 1 - F0000H - F1FFFH

EPROM 2 - Decide suitably for a practical system.

RAM 1 - Contains Interrupt vector table

RAM 2 - 30000H - 31FFFH

RAM 3 - 40000H - 41FFFH

RAM 4 - 50000H - 51FFFH

Solution Like 8086, the processor 8088 also starts execution from the address FFFF0H, after reset. Hence EPROM 2 should be allotted the address space so as to accommodate the locations FFFF0H to FFFFFH, in it. Thus the starting address of EPROM 2 should be FE000H so that the last address is FFFFFH for an 8K × 8 EPROM chip.

The interrupt vector table of 8086/88 lies in the zeroth segment of memory system, i.e. RAM 1 must start at 00000H so that the last address is 01FFFH so as to accommodate interrupt vector table in it. Let us write the map of the system as shown in Table 5.7.

Table 5.7 *Address Map for Problem 5.4*

Address	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{0}
00000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(RAM 1)																				
01FFFH	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
30000H	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
											(RAM 2)									
31FFFH	0	0	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
40000H	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
											(RAM 3)									
41FFFH	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
50000H	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
											(RAM 4)									
51FFFH	0	1	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
F0000H	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
											(EPROM 1)									
F1FFFH	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
FE000H	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
											(EPROM 2)									
FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The available 8K memory chips has [213 = 8192(8K)] 13 address lines. Total 20 address lines (A_{19}–A_0) are available from the processor. The A_0–A_{12} of the available lines are directly connected to the pins A_0–A_{12} of the memory chips. The higher order address lines (A_{13}–A_{19}) are used for deriving chip selects as shown in Fig. 5.4. In this case, as the addresses accommodated by the RAM and ROM are quite distant in the overall system map, separate decoding circuits are used for deriving the chip selects of RAM and ROM.

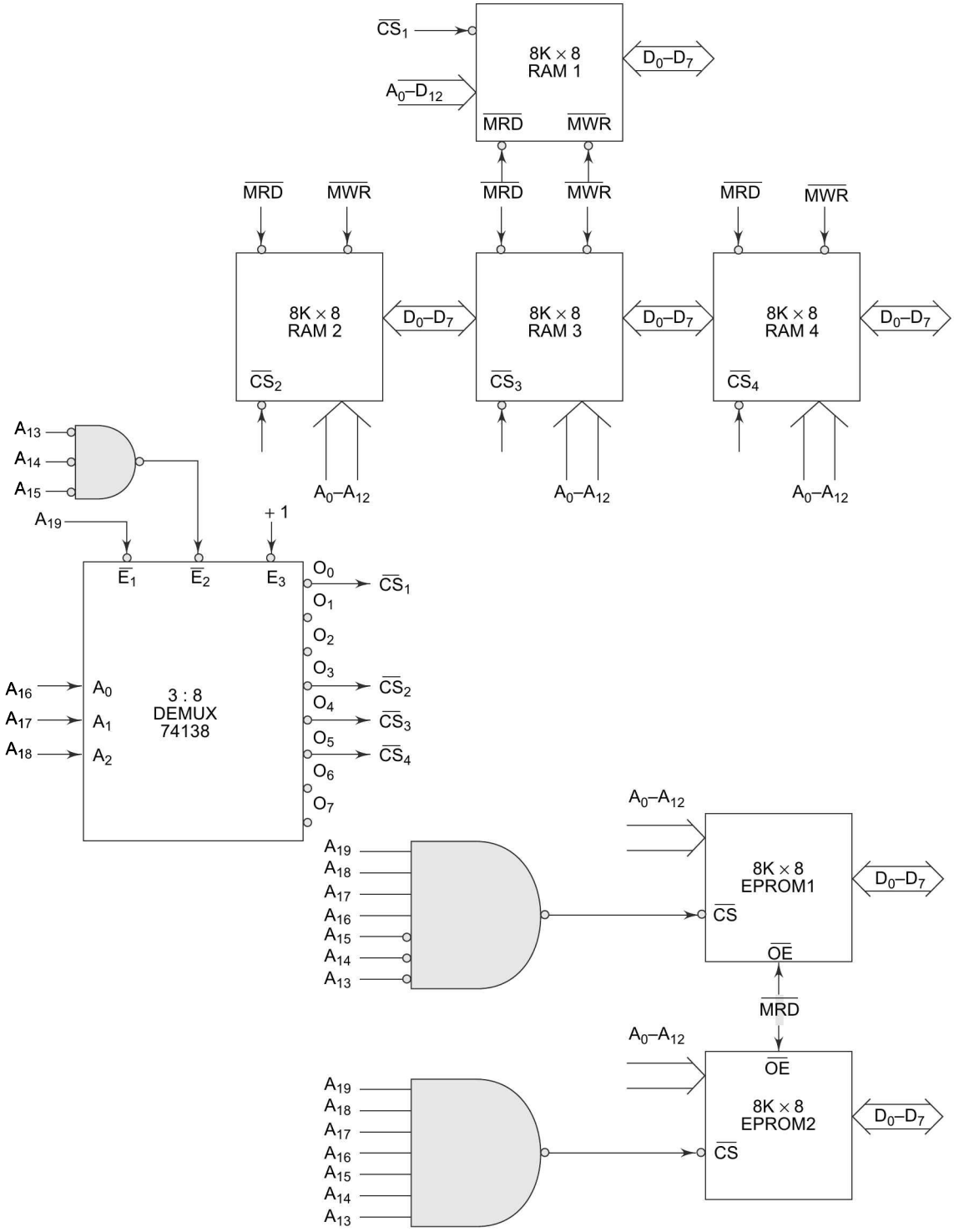


Fig. 5.4 *Interfacing Problem 5.4*

Problem 5.5

Interface a 4K × 8 EPROM, one chip of 8K × 8 RAM, two chips of 8K × 4 RAM with 8088. The map should be as given in Table 5.8.

EPROM - FF000H - FFFFFH

One 8K × 8 RAM - 00000H - 01FFFH

Two 8K × 4 RAM - 05000H - 06FFFH

Solution Let us write the complete map of the interfacing scheme.

Table 5.8 *Memory Map for Problem 5.5.*

Address	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_0
FF000H	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
(EPROM - 4K ↔ 8)																				
FFFFFH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
00000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(8K ↔ 8 RAM)																				
01FFFH	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
05000H	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
(8K ↔ 4 + 8K ↔ 4 RAM)																				
06FFFH	0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1

The above map can be implemented as shown in Fig. 5.5. 4K × 8 EPROM needs 12 address lines while 8K × 4 RAM need 13 address lines. Observe line A_{12} in the address map it needs to be inverted.

Note that A_{13} is abandoned here while interfacing 8K × 4 RAMs, because as it is varying, it cannot be used as input to the NAND gate for selecting the 8K × 4 RAMs. This is not a good interfacing practice. This problem can be avoided by using 3 to 8 demultiplexer for decoding the chip selects of the RAM chips as shown in Fig. 5.6.

In the maximum mode, the 8086 is prepared to operate in multiprocessor mode. All the control signals in this mode are derived by a chip called bus controller (8288). While interfacing memory or I/O devices in maximum mode, the control signals derived by the bus controller (8288) are used. Instead of $\overline{\text{RD}}$, for memory operations $\overline{\text{MRDC}}$ (Memory Read Command) output of 8288 is used and instead of $\overline{\text{WR}}$, $\overline{\text{AMWC}}$ (advanced memory write command) output of 8288 is used. The rest of the procedure in minimum and maximum mode is similar while interfacing the memories as well as peripherals.

Along with these $\overline{\text{MRDC}}$, $\overline{\text{IOWC}}$, $\overline{\text{AIOWC}}$, $\overline{\text{AMWC}}$ two more signals, namely $\overline{\text{IOWC}}$ and $\overline{\text{MWTC}}$, are provided by 8288. These signals are similar to $\overline{\text{AIOWC}}$ and $\overline{\text{AMWC}}$ in significance and operation but there activation is delayed by a clock cycle as compared to $\overline{\text{AIOWC}}$ and $\overline{\text{MWTC}}$.

5.2 DYNAMIC RAM INTERFACING

Whenever a large memory is required in a microcomputer system, the memory subsystem is generally designed using dynamic RAM as it has various advantages e.g. higher packaging density, lower cost and less power consumption.

A typical static RAM cell may require six transistors while the dynamic RAM cell requires only a transistor along with a capacitor. Hence it is possible to obtain higher packaging density and hence low cost units

are available. On the other side, there are some serious drawbacks of dynamic RAMs. The basic dynamic RAM cell uses a capacitor to store the charge as a representation of data. This capacitor is manufactured as a diode that is reverse-biased so that the storage capacitance comes into the picture. This storage capacitance is utilized for storing the charge representation of data but the reverse-biased diode has a leakage current that

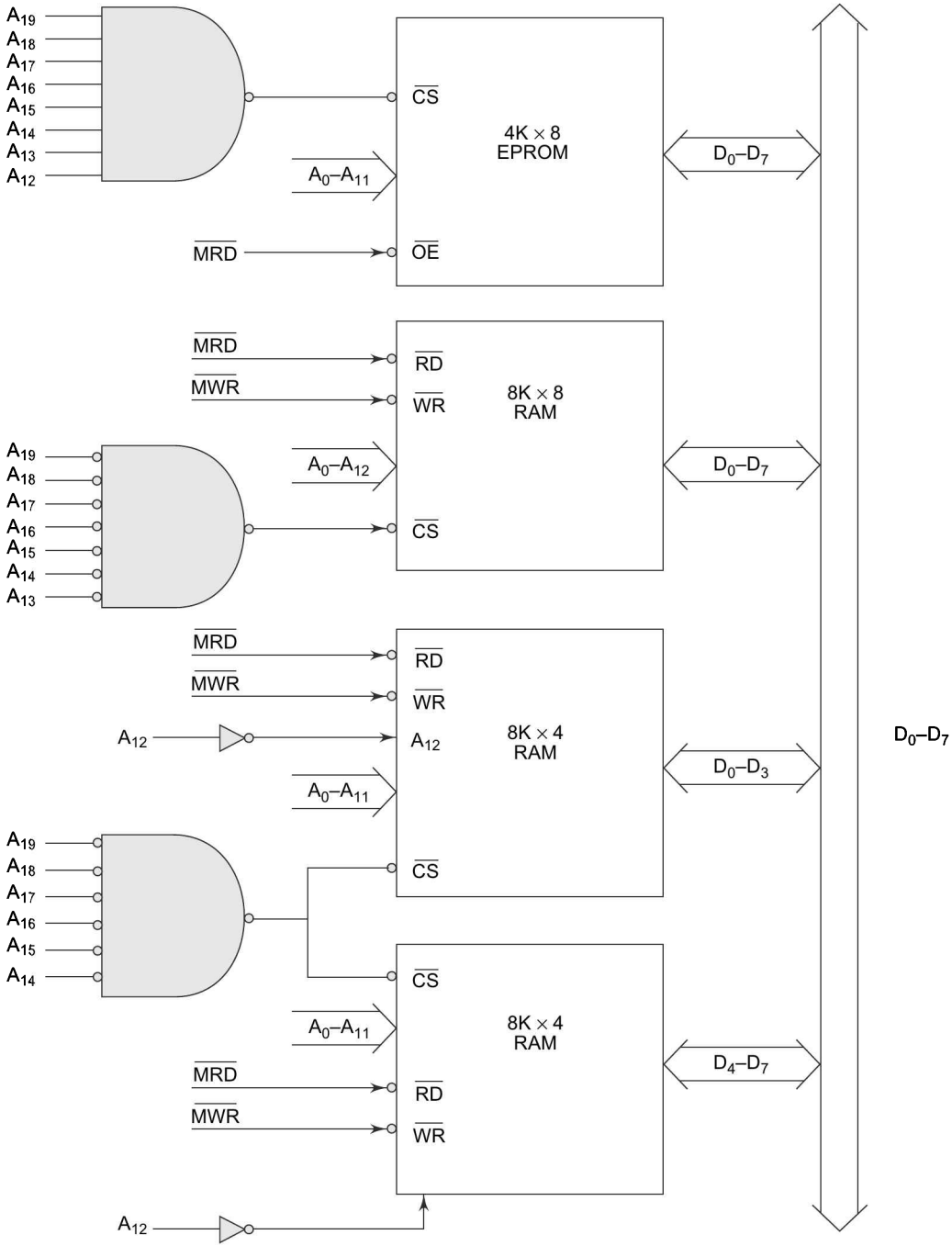


Fig. 5.5 *Interfacing Problem 5.5*

tends to discharge the capacitor giving rise to the possibility of data loss. To avoid this possible data loss, the data stored in a dynamic RAM cell must be refreshed after a fixed time interval regularly. The process of refreshing the data in the RAM is known as *refresh cycle*. This activity is similar to reading the data from each cell of the memory, independent of the requirement of microprocessor, regularly. During this refresh period all other operations (accesses) related to the memory subsystem are suspended. Hence the refresh activity

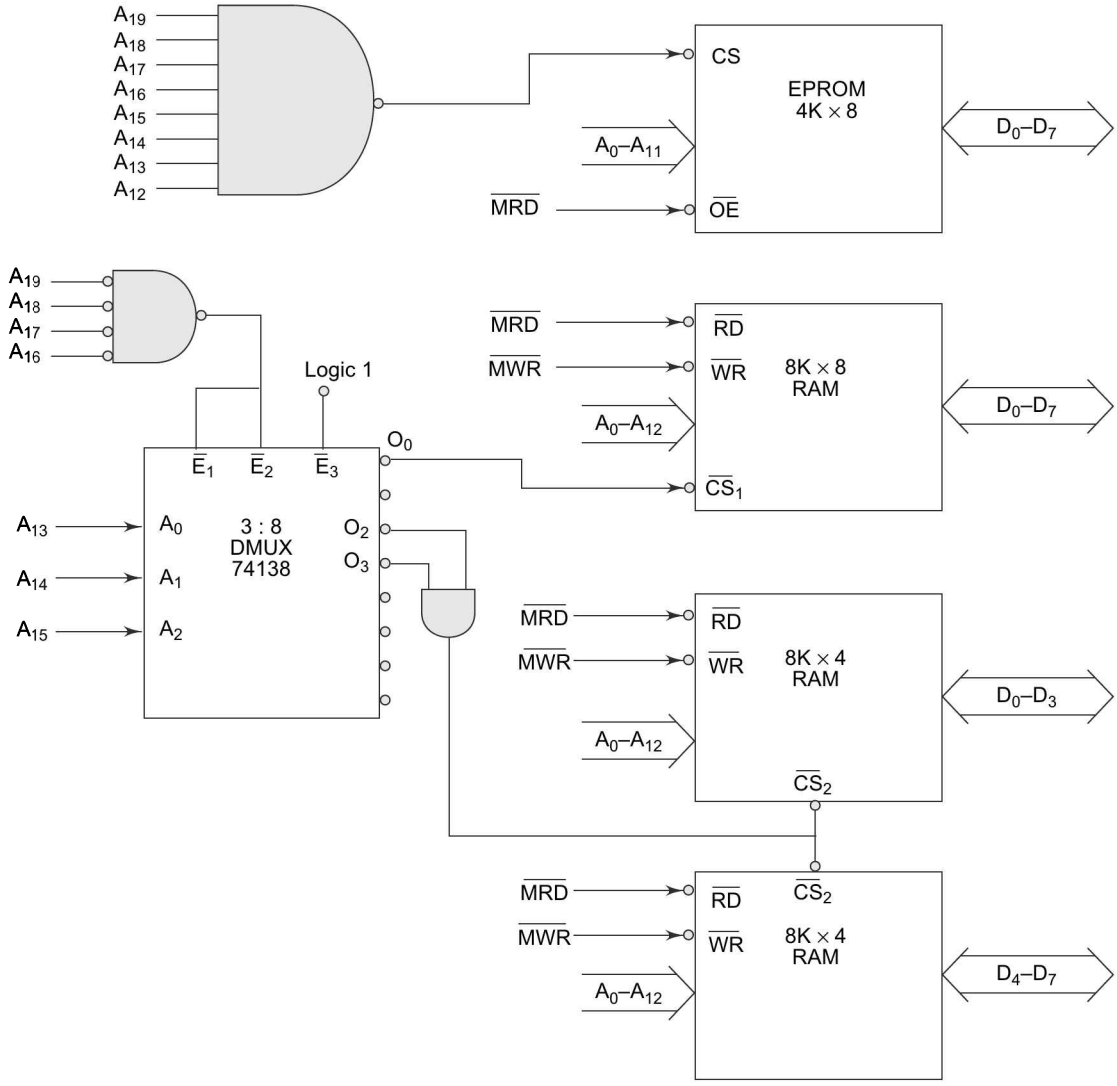


Fig. 5.6 *Alternative Implementation for Problem 5.5*

causes loss of time, resulting in reduced system performance. However, keeping in view the advantages of dynamic RAM, like low power consumption, higher packaging density and low cost, most of the advanced computer systems are designed using dynamic RAMs, of course, at the cost of operating speed. Also, the refresh mechanism and the additional hardware required makes the interfacing hardware, in case of dynamic RAM, more complicated, as compared to static RAM interfacing circuit. A dedicated hardware chip called as dynamic RAM controller is the most important part of the interfacing circuit.

The refresh cycle is different from the memory read cycle in the following aspects.

1. The memory address is not provided by the CPU address bus, rather, it is generated by a refresh mechanism counter known as refresh counter.
2. Unlike memory read cycle, more than one memory chip may be enabled at a time so as to reduce the number of total memory refresh cycles.
3. The data enable control of the selected memory chip is deactivated, and data is not allowed to appear on the system data bus during refresh, as more than one memory units are refreshed simultaneously. This is to avoid the data from the different chips to appear on the bus simultaneously.
4. Memory read is either a processor initiated or an external bus master initiated operation while memory refresh is an independent regular activity, initiated and carried out by the refresh mechanism.

Generally, dynamic RAM is available in units of several kilobits to even megabits of memory (note that it is not in terms of bytes or nibbles as in a static RAM). This memory is arranged internally in a two dimensional matrix array so that it will have *n* rows and *m* columns. The row address *n* and column address *m* are important for the refreshing operation. For example, a typical 4K bit dynamic RAM chip has an internally arranged bit array of dimension 64 × 64, i.e. 64 rows and 64 columns. Thus the row address and column address will require 6 bits each. These 6 bits for each row address and column address will be generated by the refresh counter, during the refresh cycles. A complete row of 64 cells is refreshed at a time to minimize the refreshing time. Thus the refresh counter needs to generate only row addresses. The row addresses are multiplexed, over lower order address lines. The refresh signals act to control the multiplexer, i.e. when refresh cycle is in process the refresh counter puts the row address over the address bus for refreshing. Otherwise, the address bus of the processor is connected to the address bus of DRAM, during normal processor initiated activities. A timer, called *refresh timer*, derives a pulse for refreshing action after each refresh interval, which can be qualitatively defined as the time for which a dynamic RAM cell can hold data charge level practically constant, i.e. no data loss takes place. Suppose the typical dynamic RAM chip has 64 rows, then each row should be refreshed after each refresh interval, or in other words, all the 64 rows are to be refreshed in a single refresh interval. This refresh interval depends upon the manufacturing technology of the dynamic RAM cell. It may range anywhere from 1 ms to 3 ms. Let us consider 2 ms as a typical refresh time interval. Hence, the frequency of the refresh pulses will be calculated as shown:

$$\text{Refresh Time (per row) } t_r = \frac{2\times10^{-3}}{64}$$

$$\text{Refresh Frequency } \quad f_r = \frac{64}{2\times10^{-3}} = 32\times10^{3} \text{ Hz}$$

The block diagram in Fig. 5.7 explains the refreshing logic and 8086 interfacing with dynamic RAM. Each of the used chips is a16K × 1-bit dynamic RAM cell array. The system contains two 16K byte dynamic RAM units. All the address and data lines are assumed to be available from an 8086 microcomputer system. The $\overline{OE}$ pin controls output data buffers of the memory chips. The CE pins are active high chip selects of memory chips. The refresh cycle starts, if the refresh output of the refresh timer goes high, $\overline{OE}$ and CE also tend to go high. The high CE enables the memory chip for refreshing, while high $\overline{OE}$ prevents the data from appearing on the data bus, as said in the description of the memory refresh cycle. The 16K × 1-bit dynamic RAM has an internal array of 128 × 128 cells, requiring 7 bits for row addresses. The lower order seven lines A_0–A_6 are multiplexed with the refresh counter output $A_{10} - A_{16}$. The logic is shown in Fig. 5.7.

The pin assignments for 2164 dynamic RAM is shown in Fig. 5.8 (a). The $\overline{RAS}$ and $\overline{CAS}$ are row and column address strobes and are driven by the dynamic RAM controller outputs. $A_0 - A_7$ lines are the row or column address lines, driven by $OUT_0 - OUT_7$ outputs of the controller. The $\overline{WE}$ pin indicates memory write cycles. The D_{IN} and D_{OUT} pins are data pins for write and read operations, respectively. In practical circuits, the refreshing logic is integrated inside dynamic RAM controller chips like 8203, 8202 and 8207, etc.

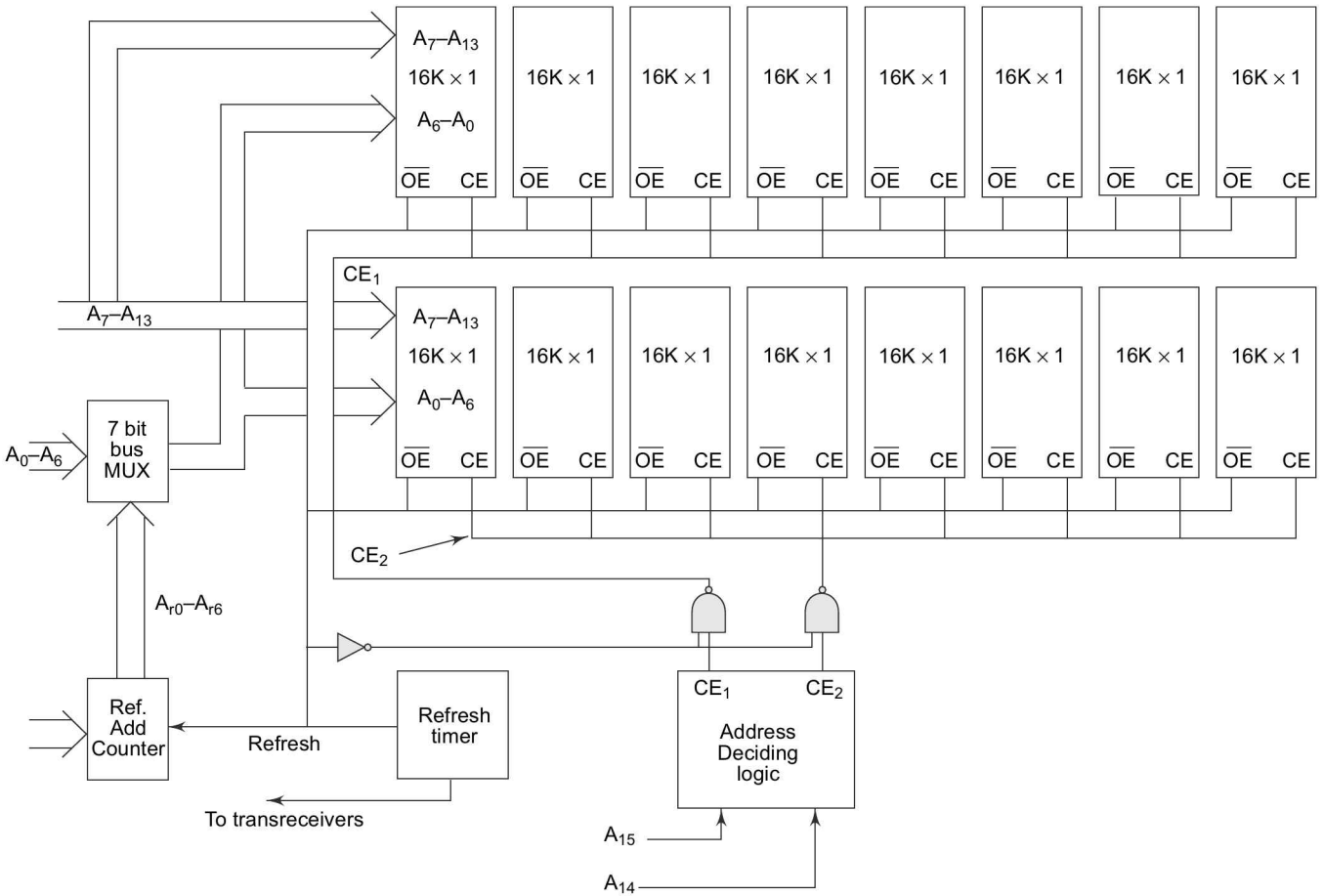


Fig. 5.7 *Dynamic RAM Refreshing Logic*

Intel's 8203 is a dynamic RAM controller (Fig. 5.8(b)) that supports 16K or 64K dynamic RAM chips. This selection is done using pin 16K/64K. If it is high, the 8203 is configured to control 16K dynamic RAM, else it controls 64K dynamic RAM. The address inputs of 8203 controller accept address lines A_1 to A_{16} on lines AL_0–AL_7 and AH_0–AH_7. The A_0 line is used to select the even or odd bank. The $\overline{RD}$ and $\overline{WR}$ signals decode whether the cycle is a memory read or memory write cycle and are accepted as inputs to 8203 from the microprocessor. The $\overline{WE}$ signal specifies the memory write cycle and is an output from 8203 that drives the $\overline{WE}$ input of dynamic RAM memory chip. The $\overline{OUT_0}$ – $\overline{OUT_7}$ set of eight pins is an 8-bit output bus that carries multiplexed row and column addresses of a bit location in a dynamic RAM chip. The $\overline{CAS}$ signal strobes the column address on $\overline{OUT_0}$ – $\overline{OUT_7}$. The $\overline{RAS_1}$ – $\overline{RAS_0}$ pins strobes row address on $\overline{OUT_0}$ – $\overline{OUT_7}$ for at the most two banks of 2164 dynamic RAM chips. Actually the row and column addresses are derived from the address lines A_1 – A_{16} accepted by the controller on its inputs AL_0–AL_7 and AH_0–AH_7. An external crystal may be applied between X_0 and X_1 pins, otherwise, with the OP_2 pin at +12V, a clock signal may be applied at the pin CLK. The $\overline{PCS}$ pin accepts the chip select signal derived by an address decoder.

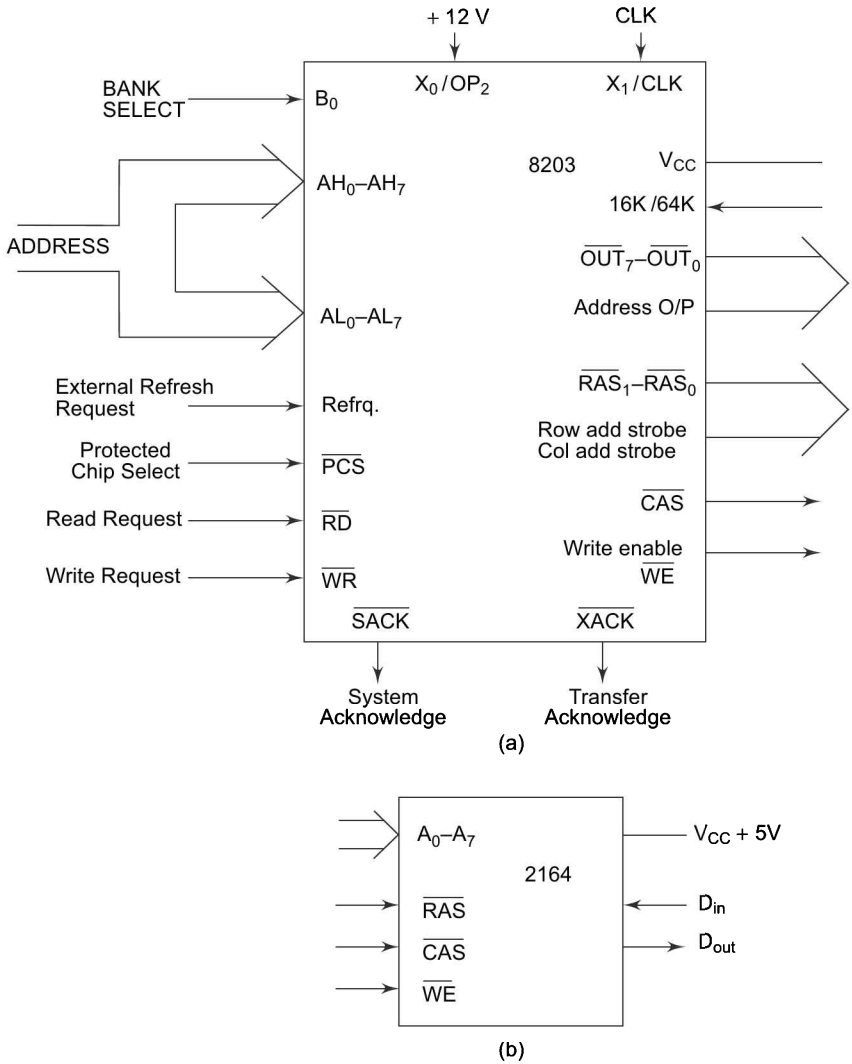


Fig. 5.8 *(a) Dynamic RAM controller (b) 1-bit Dynamic RAM*

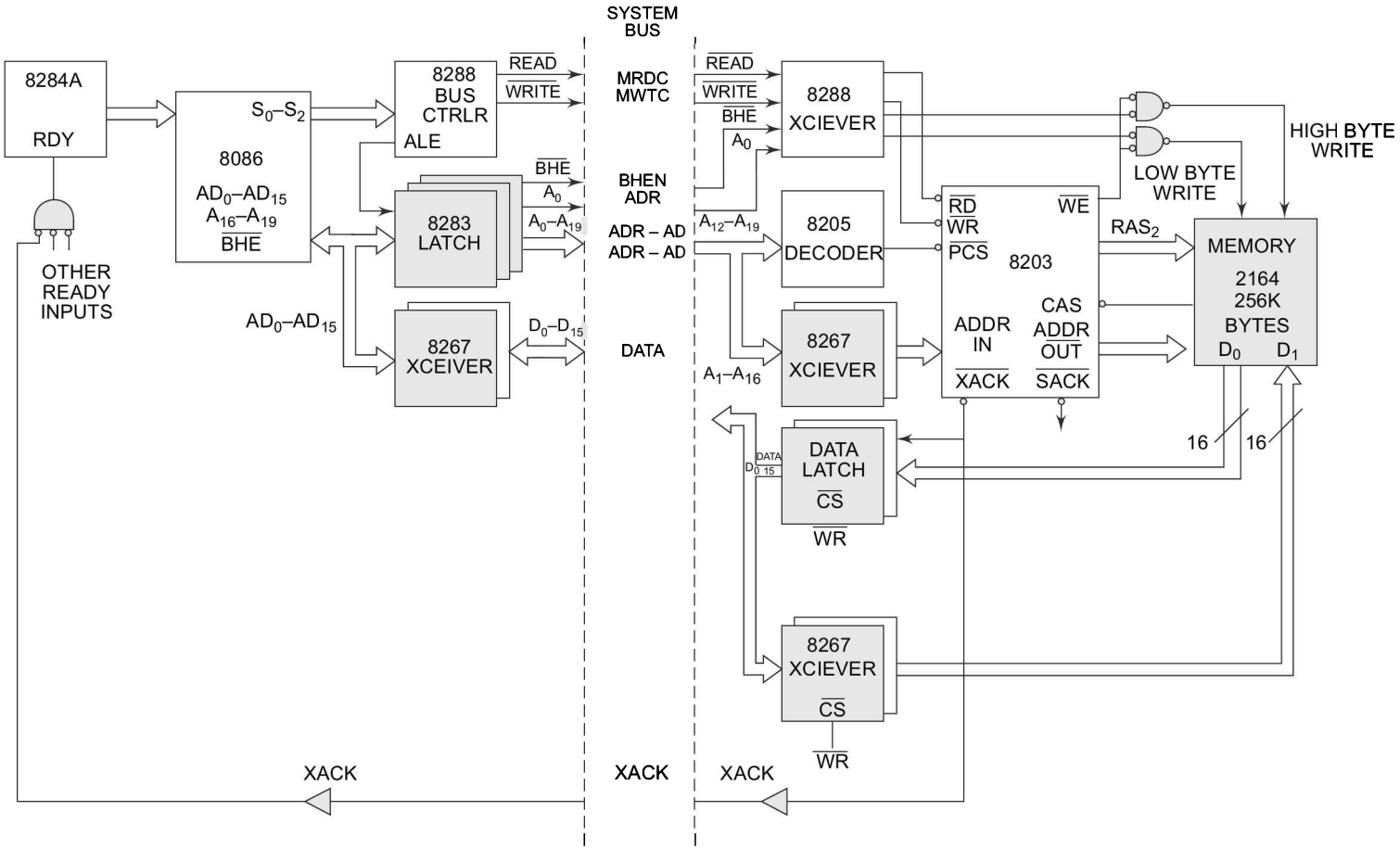


Fig. 5.9 *Interfacing 2164 Using 8203*

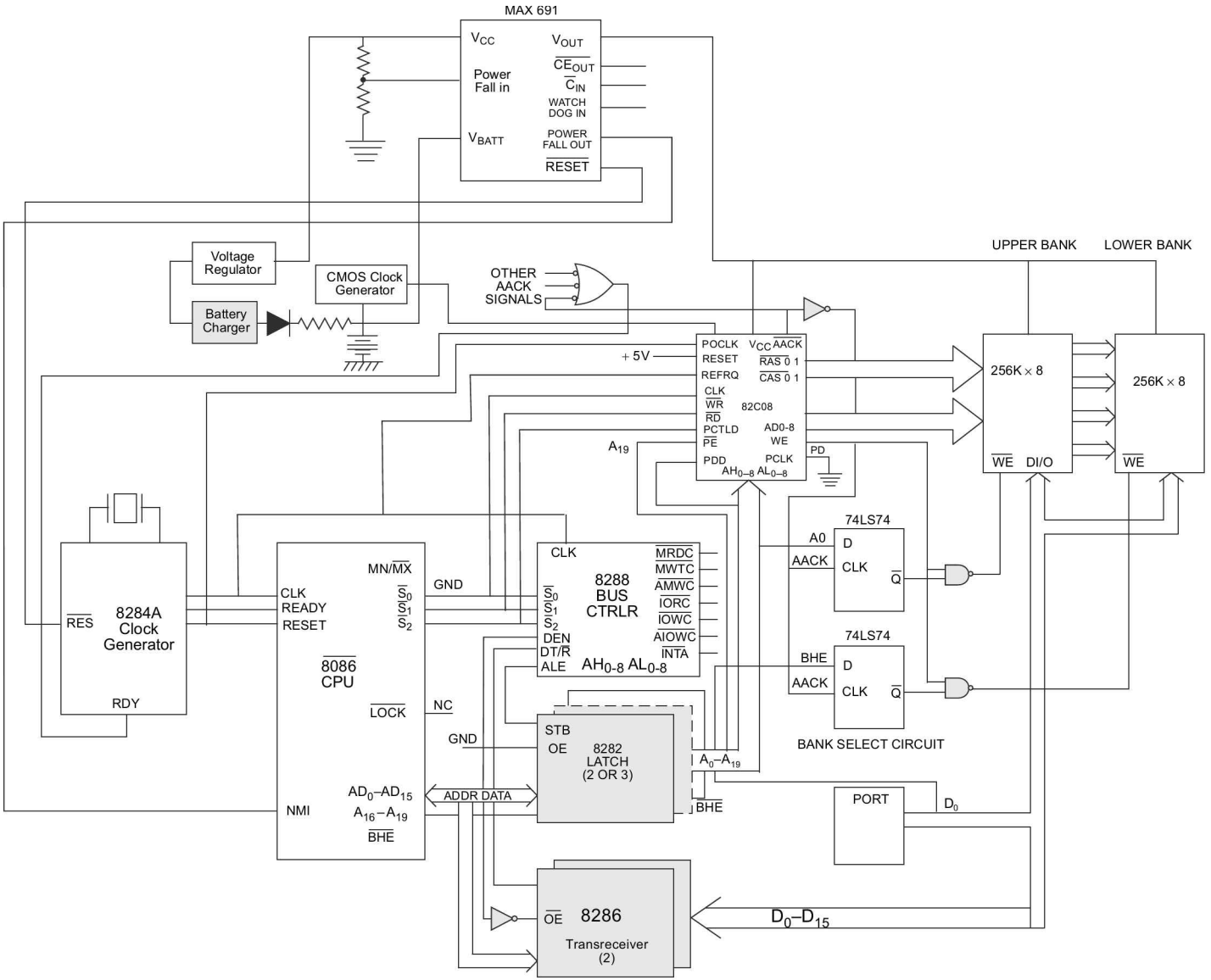


Fig. 5.10 *Interfacing 512 Kbyte of Dynamic RAM with 8086*

The REFREQ pin is used whenever the memory refresh cycle is to be initiated by an external signal. The $\overline{\text{XACK}}$ signal indicates that data is available during a read cycle or it has been written if it is a write cycle. It can be used as a strobe for data latches or as a ready signal to the processor. The $\overline{\text{SACK}}$ output signal marks the beginning of a memory access cycle. If a memory request is made during a memory refresh cycle, the $\overline{\text{SACK}}$ signal is delayed till the starting of memory read or write cycle. Figure 5.9 shows how the 8203 can be used to control a 256K bytes memory subsystem for a maximum mode 8086 microprocessor system. This design assumes that data and address busses are inverted and latched, hence the inverting buffers and inverting latches are used (8283-inverting buffer and 8287-inverting latch).

Figure 5.10 shows the interfacing of 512K byte dynamic RAM with 8086 using an advanced dynamic RAM controller 8208. Most of the functions of 8208 and 8203 are similar but 8208 can be used to refresh the dynamic RAM using DMA approach. The memory system is divided into even and odd banks of 256Kbyte each, as required for an 8086 system. The inverted $\overline{\text{AACK}}$ output of 8208 latches the A_0 and $\overline{\text{BHE}}$ signals required for selecting the banks. If the latched bank select signal and the $\overline{\text{WE}}$/PCLK output of 8208 both go low it indicates a write operation to the respective bank.

5.3 INTERFACING I/O PORTS

I/O ports or Input/Output ports are the devices through which the microprocessor communicates with other devices or external data sources/destinations. Input activity, as one may expect, is the activity that enables the microprocessor to read data from external devices , for example keyboards, joysticks, mouse, etc. These devices are known as input devices as they feed data into a microprocessor system.

Output activity transfers data from the microprocessor to the external devices, for example CRT display, 7-segment displays, printers, etc. The devices which accept the data from a microprocessor system are called output devices. Thus for a microprocessor the input activity is similar to read operation, while the output activity is similar to write operation. Note that an input device can only be read and an output device can only be written.

Hence $\overline{\text{IORD}}$ operation is related with reading data from an input device and not an output device and $\overline{\text{IOWR}}$ operation means writing data to an output device and not an input device. The control word and status word may be written and read respectively in both, input or output, devices, in case of programmable devices.

After executing an OUT operation, the data appears on the data bus and simultaneously a device select signal is generated from the address and control signals. Now, if the data is to be there, at the output of the

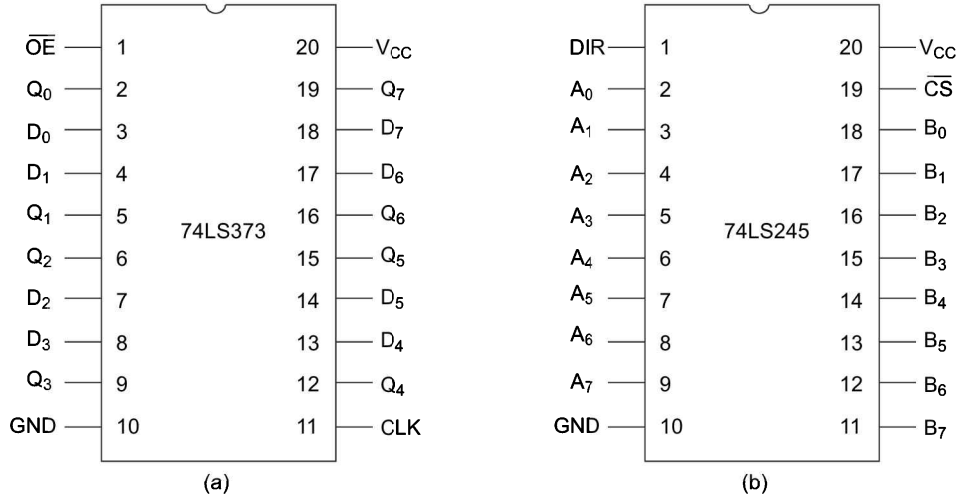


Fig. 5.11 *(a) Latch (O/P port) (b) Buffer (I/P port)*

device till the next change, it must be latched. Also if the output port is to source large currents the port lines must be bufferred. Hence the latch acts as a good output port. The chip 74LS373, contains eight bufferred latches and can be used as an 8-bit output port. While reading, an input device one must take care that much current should not be sourced or sunk from the data lines to avoid loading. To overcome this problem, one may use a tristate buffer as an input device. An input port may not be a latch as it reads the status of a signal at a particular instant. The chip 74LS245 contains eight buffers and may be used as an 8-bit input port. Actually, 74LS245 is a bidirectional buffer, but while using it as an input device, only one direction is useful. This direction of data transfer in 74LS245 is selected using its DIR pin. The pin diagrams of 74LS373 and 74LS 245 are shown in Figs 5.11 (a) and (b). The $\overline{OE}$ and $\overline{CS}$ are the chip selects of 74LS373 and 74LS245 respectively. Ds and Qs are corresponding latch inputs and outputs respectively.

The CLK pin is clock input for D flip-flops. If DIR is 1, then the direction is from A(I/Ps) to B(O/Ps), otherwise the data direction is from B(I/Ps) to A(O/Ps).

Steps in Interfacing an I/O Device The following steps are performed to interface a general I/O device with a CPU:

(i) Connect the data bus of the microprocessor system with the data bus of the I/O port.
(ii) Derive a device address pulse by decoding the required address of the device and use it as the chip select of the device.
(iii) Use a suitable control signal, i.e. $\overline{IORD}$ and/or $\overline{IOWR}$ to carry out device operations, i.e. connect $\overline{IORD}$ to $\overline{RD}$ input of the device if it is an input device, otherwise connect $\overline{IOWR}$ to $\overline{WR}$ input of the device. In some cases the $\overline{RD}$ or $\overline{WR}$ control signals are combined with the device address pulse to generate the device select pulse.

Methods of Interfacing I/O Devices There are two methods of interfacing general I/O devices.

(i) I/O mapped
(ii) Memory-mapped

The principal distinction in the two approaches is that in I/O mapped interfacings the devices are viewed as distinct I/O devices and are addressed accordingly. While *in memory-mapped scheme,* the devices are viewed as memory locations and are addressed likewise. In I/O mapped interfacing, all the available address lines of a microprocessor may not be used for interfacing the devices. The processor 8086 has 20 address lines. The I/O mapped scheme may use at the most 16 address lines $A_0 - A_{15}$ or even 8 address lines for address decoding. The unused higher order address lines are logic zero, while addressing the device. An I/O mapped device requires the use of IN and OUT instructions for accessing them. The I/O mapped method requires less hardware for decoding, as less number of address lines are used. In case of 8086, a maximum of 64K input and 64K byte output devices or 32K input and 32K word output devices can be interfaced. In addition to address and data busses, to address an input device, we require the $\overline{IORD}$ signal and to address an output device, we use $\overline{IOWR}$ signal for the respective operations. The $\overline{IOWR}$ and $\overline{IORD}$ signals are used for I/O mapped interfacing.

In memory-mapped interfacing, all the available address lines are used for address decoding. Thus each memory-mapped I/O device with 8086 has a 20-bit address, i.e. 8086 can have as many as 1M memory-mapped input and as many byte output devices. Practically this is impossible, as memory-mapped I/O devices consume the addresses in the memory map of the CPU. 1M byte devices will require the complete 1Mbyte of the memory map and nothing will be left as program memory. Also the memory locations and the memory-mapped devices cannot have common addresses. The $\overline{MRDC}$ and $\overline{MRTC}$ signals are used for interfacing in memory-mapped I/O scheme. All the applicable data transfer instructions (e.g. MOV, LEA) can be used to communicate with memory-mapped I/O devices. However, I/O operations are much more sluggish

compared to the memory operations which are faster. Moreover, complex decoding hardware is required in this case since all the address lines are used for decoding.

In case of the 8086 systems, the memory-mapped method is seldom used. Hence all the peripheral devices in most of the practical systems are essentialy I/O mapped devices. In this book, only the I/O mapped method of interfacing is elaborated. 8086 has a 16-bit data bus, hence interfacing of 8-bit devices with 8086 need special consideration. Usually, 8-bit I/O devices are interfaced with lower order data bus of 8086, i.e. D_0–D_7. The 16-bit devices are interfaced directly with the 16-bit data bus, using A_0 and $\overline{BHE}$ pins of 8086. The following problems explain the actual method of interfacing I/O devices with 8086. The interfacing hardware always need supporting application programs to carry out the desired operations. Hence each interfacing example is accompanied with a supporting 8086 ALP.

Problem 5.6

Interface an input port 74LS245 to read the status of switches SW_1 to SW_8. The switches, when shorted, input a '1' else input a '0' to the microprocessor system. Store the status in register BL. The address of the port is 0740H.

Solution The hardware interface circuit is shown in Fig. 5.12. The address, control and data lines are assumed to be readily available at the microprocessor system.

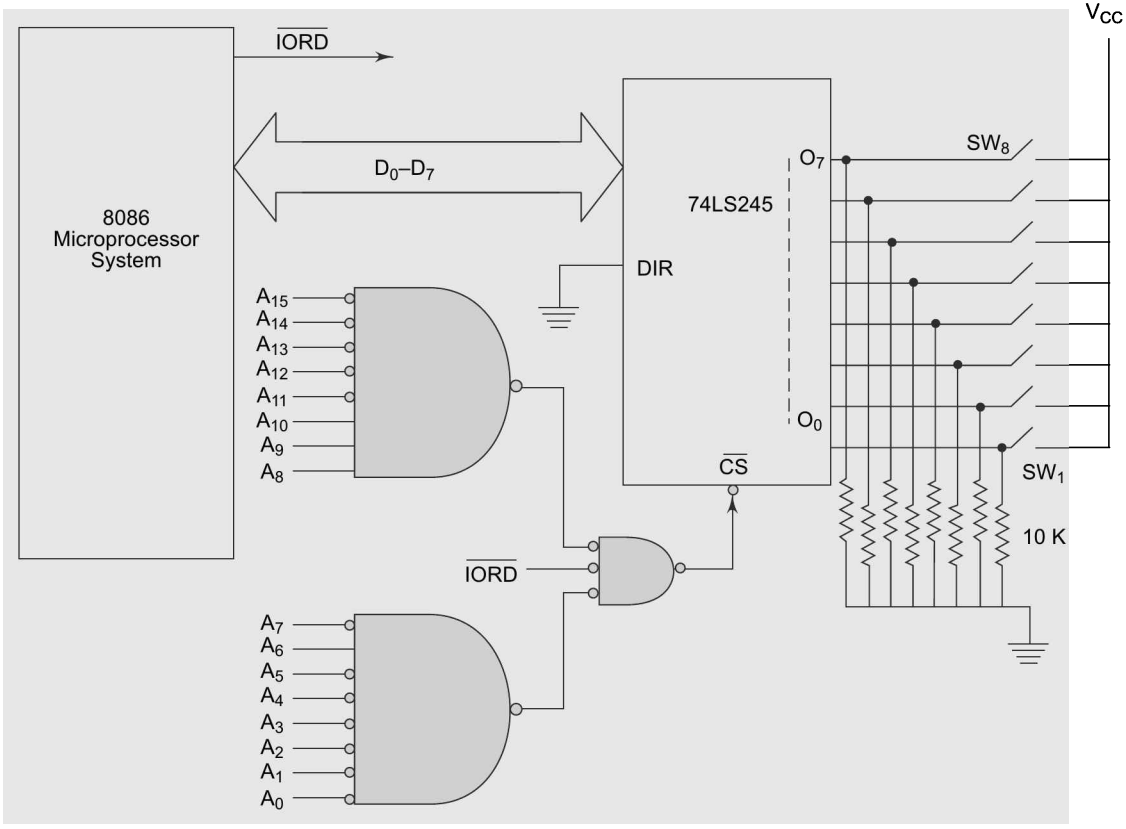


Fig. 5.12 *Interfacing Input Port 74LS245*

```
THE ALP IS GIVEN AS FOLLOWS:
MOV BL, 00H       ; Clear BL for status
MOV DX, 0740H     ; 16-bit port addrenss in DX
IN AL, DX         ; Read port 0740H for switch positions
MOV BL, AL        ; Store status of switches from AL into BL
HLT               ; Stop
```

Program 5.1 *ALP for Problem 5.6.*

Here LSB bit of BL corresponds to the status of SW_1 and likewise the MSB of BL corresponds to the status of SW_8. The '1' indicates 'on' or shorted switch and the '0' indicates an 'off' or opened switch. The pull-up registers in Fig. 5.12 are necessary because the open switches should input a '0' to the system but the TTL port 74LS245 will read the free input as '1'. (Free TTL inputs are always read as logic '1'.)

Problem 5.7

Design an interface of an input port 74LS245 to read the status of switches SW_1 to SW_8 (as in the previous problem), and an output port 74LS373 with 8086. Display the number of a key that is pressed, i.e. from 1 to 8 on a 7-seg display with help of the output port. Write an ALP for this task, assume that only one key is pressed at a time. Draw the schematic of the required hardware. The input port address is 0008H and the output port address is 000AH.

Solution In the previous problem, one might have noted that a lot of hardware is required to decode the port address absolutely. Thus instead of decoding the address completely, only a part of it may be decoded. For example, instead of using 16 address lines A_{15}–A_0, one may use only A_3–A_0. In this problem, the address 0008H may then be converted to xxx8H, where x denotes a don't care condition. Thus the port may have more than one address, for example 2358H, 1728H etc. Only the least significant nibble of the address needs to be 8H. The disadvantage of the scheme is that there are a number of addresses of the same port. Hence, the system must have only one port that has the lowest nibble address 8H, otherwise, the system may malfunction. Thus for smaller systems containing a few I/O ports, this scheme is suitable and advantageous as it requires less hardware.

The status of the switches is first read into the register AL. For displaying the shorted switch number in the 7-seg display, the bit corresponding to the switch is checked by rotating AL through carry and then checking the carry flag. If the carry flag is '1', after one left rotation, it means SW_1 is on. If the carry flag sets after two rotations, SW_2 is on and so on. Register CL is incremented after each rotation so that it contains the pressed switch number. The 8-bit contents of register CL are converted to 7-segment codes by a BCD to 7-seg decoder. The complete hardware (Fig. 5.13) and the ALP is given as shown. Note that both the ports are interfaced at even addresses, i.e. with lower order data bus $D_0 - D_7$.

Common cathode displays are used along with corresponding BCD to 7-seg decoder.

```
MOV BL, 00   ; Clear BL for switch status
MOV CL, 00   ; Clear CL for switch number
XOR AX, AX   ; Clear accumulators and flag
IN AL,08H    ; Read switch status
```

```
     INC CL        ; Increment CL for 1st switch
YY:  RCR AL        ; Rotate switch status
     JC XX         ; If carry, halt,
     INC CL        ; else increment CL for next switch
     JMP YY        ; number till carry is 1
XX:  MOV AL,CL     ; Take switch number into AL
     OUT OAH,AL    ; Out BCD switch number for display
     HLT           ; Stop
```

Program 5.2 *ALP for Problem 5.7*

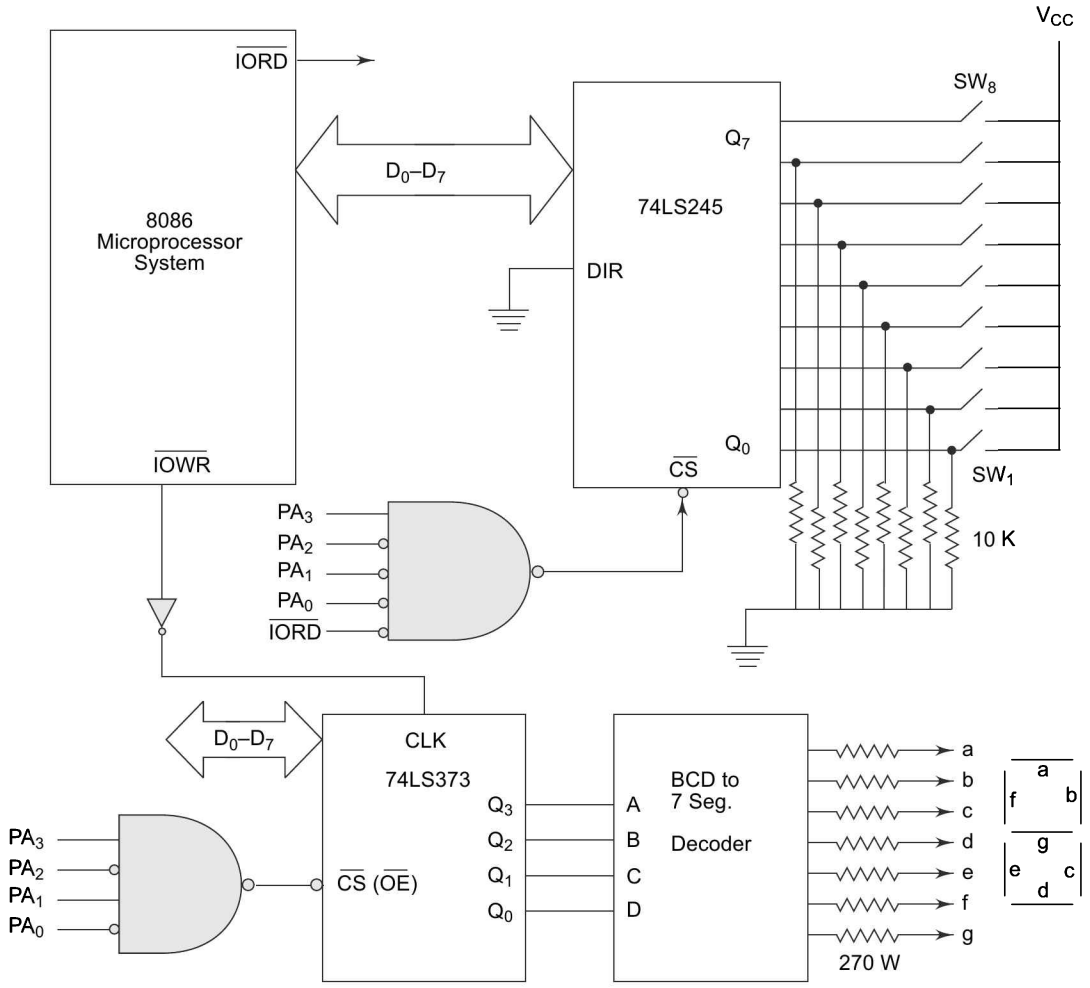


Fig. 5.13 *Interfacing Switches and Displays for Problem 5.7*

Problem 5.8

Using 74LS373 output ports and 7-segment displays, design a seconds counter that counts from 0 to 9. Draw the suitable hardware schematic and write an ALP for this problem. Assume that a delay of 1sec is available as a subroutine. Select the port address suitably.

Solution The counter hardware is shown in Fig. 5.14. Common cathode displays are used along with a suitable BCD to 7-segment decoder. The ALP calls the subroutine 'DELAY' that generates a delay program of 1sec. After counting from 0 to 9, it again starts from 0. The output port is interfaced at address 0008H.

The ALP for generating the seconds count is given below.

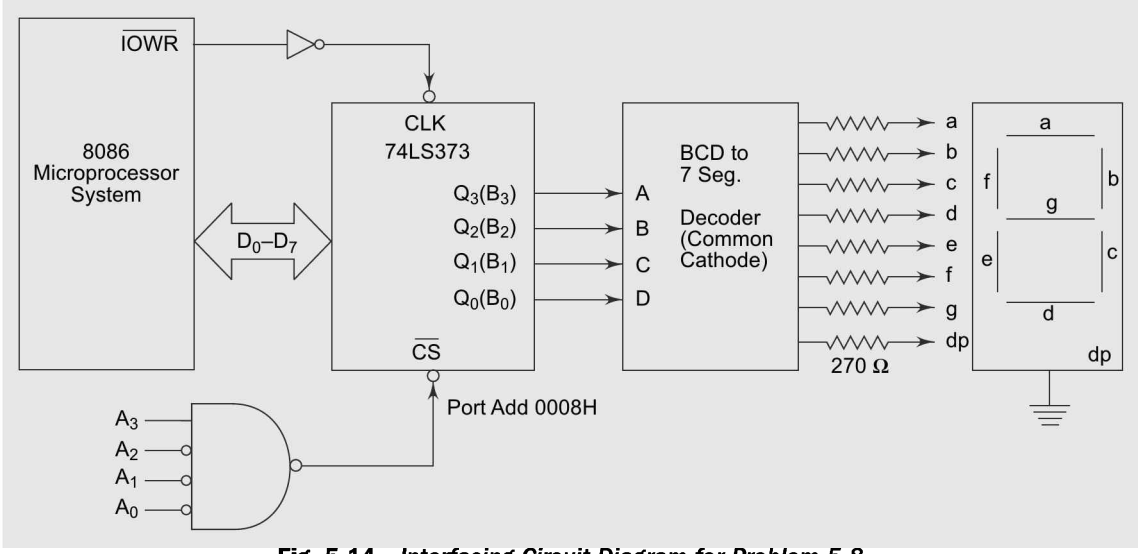


Fig. 5.14 *Interfacing Circuit Diagram for Problem 5.8*

```
XX : MOV AL,00H          ; Start from 00
YY : XOR AL,AL           ; Clear AL and flags
     OUT 08H,AL          ; Display 0H
     CALL DELAY          ; Wait for one second
     INC AL              ; Increment count for next
     CMP AL,0A H         ; display. Is it above 9H?
     JZ XX               ; If yes, start from 00,
     JMP YY              ; else continue.
```

Program 5.3 *ALP for Problem 5.8*

Multiplexed 7-seg Displays To display a single digit, at least one output port is required. Suppose one wants a 5-digit display for some practical application, Five such ports will be required, i.e. the hardware will be very complex and costly. To minimize the complexity and cost of hardware one may implement an almost visually identical display using only two ports. Suppose there are four 7-seg displays, numbered 1, 2, 3 and 4. One of the two ports, say port1 selects one of these displays, say display 1 at a time, while port 2 sends the data to be displayed (i.e. a,b,c,d,e,f,g and dp) to the first display for a fixed short duration. Port 1 next, selects the display 2 and port 2 sends the appropriate display data to it. Each of the display unit remains active for a short duration and the process continues in a loop. This is repeated at a high frequency so that the complete display containing more than one7-seg display appears to be stationary due to the persistence of vision. Instead of BCD to 7-seg decoder circuit, look up table technique may be used for converting BCD numbers to equivalent 7-seg codes.

Problem 5.9

Draw a schematic hardware circuit for interfacing five, 7-seg displays (common cathode) with 8086 using output ports. Display numbers 1 to 5 on them continuously. The 7-seg codes are stored in a look-up table serially at the address 2000:0000 H onwards starting from code for 1.

Solution Let us select the two port addresses 0004H and 0008H for the output ports. The first port 0004H outputs 7-seg code while the second output port 0008H selects the display by grounding the common cathode. The hardware is given in Fig. 5.15.

The 7-seg codes for C.C. displays can be decided as given. For a LED to be 'on', that particular anode should be 1 and the common cathode line should be grounded, using a port line that drives a transistor. Thus for the numbers to be displayed the code is calculated as shown.

Decimal no.	*a* A_7	*b* A_6	*c* A_5	*d* A_4	*e* A_3	*f* A_2	*g* A_1	*dp* A_0	
1—	1	1	0	0	0	0	0	0	= C0
2—	1	1	0	1	1	0	1	0	= DA
3—	1	1	1	1	0	0	1	0	= F2
4—	0	1	1	0	0	1	1	0	= 66
5—	1	0	1	1	0	1	1	0	= B6

These codes are stored in a look up table starting from 2000H:0000, as shown below.

2000 : 0000 → C0 H
2000 : 0001 → DA H
2000 : 0002 → F2 H
2000 : 0003 → 66 H
2000 : 0004 → B6 H

Only one display should be selected at a time, i.e. only the corresponding bit of port 2 should be high for selecting a common cathode display. All the other bits should be low to keep the other displays disabled. Thus to enable the least significant display, the LSB of the 8-bit selected port should remain '1'. Hence AL should have 01 or E1H in it to select the least significant display. The codes for the selection of displays and 7-segment codes directly depend upon the hardwared connections between them.

```
          MOV AX, 2000H          ; Initialize pointer to
          MOV DS, AX             ; Code table DS:BX
          MOV BX, 0000H
NEXT  :   MOV AL, 00H            ; Get 1st number from the table.
          MOV DH,AL
          MOV CL, 05H            ; Count for display
          MOV DL, E1H            ; Selection code for 1st display
AGAIN :   XLAT                   ;
          OUT 04H, AL            ; Out the code for the first
                                 ; number to port 04H.
          MOV AL,DL              ; Get to be enabled display code.
          OUT 08H,AL             ; Select 1st display.
          ROL DL                 ; decide code for selecting next
          INC DH                 ; display for next number
          MOV AL,DH              ; get next num. to be displayed.
          LOOP AGAIN             ; Repeat five times
          JMP NEXT               ; Continue the procedure
```

Program 5.4 *ALP for Problem 5.9*

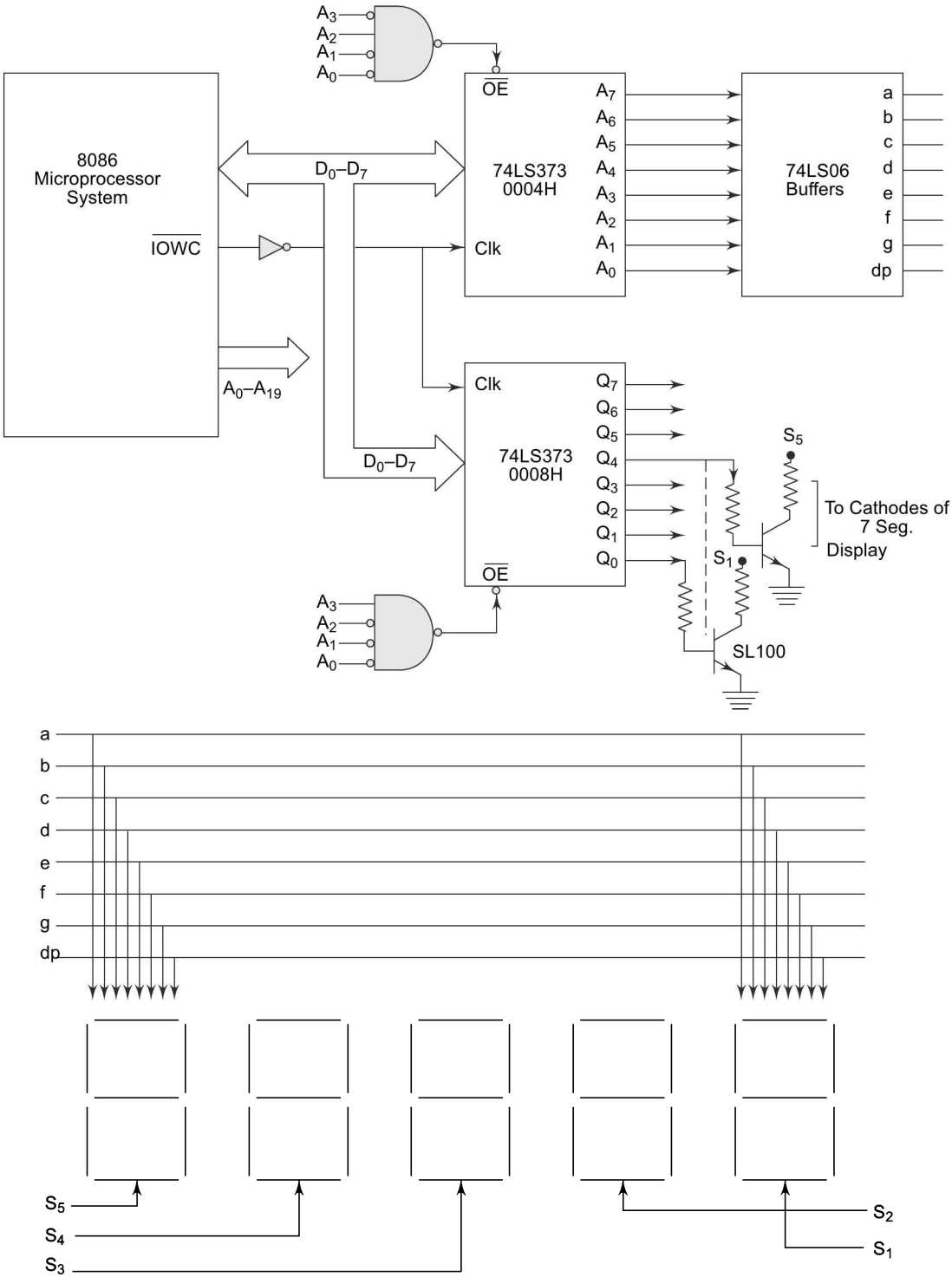


Fig. 5.15 *Interfacing Circuit Diagram of Problem 5.9*

For interfacing a 16-bit output port with 8086, we may use two 8-bit output ports to form a 16-bit port, with a single address, as shown in Fig. 5.16. Both the 8-bit ports are in this case addressed as a single 16-bit port with a single address.

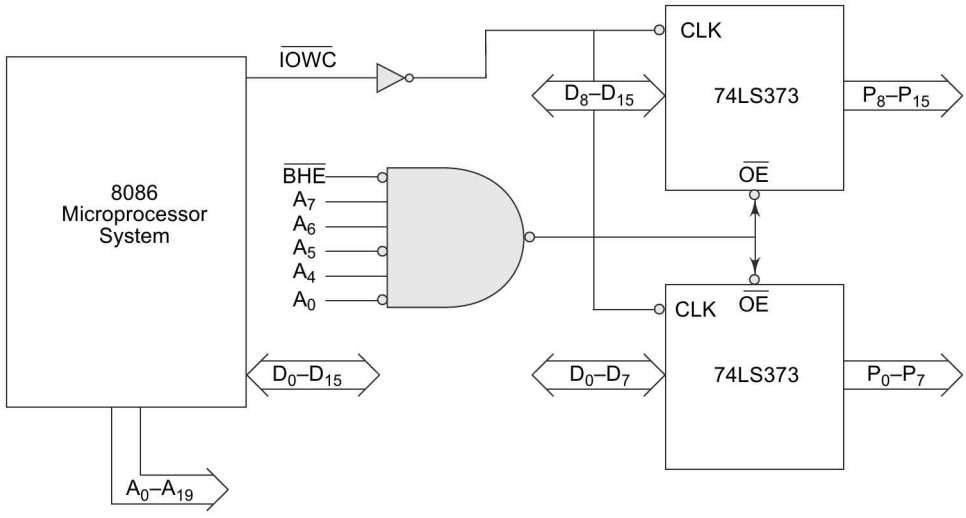


Fig. 5.16 *Interfacing a Circuit of 16 bit output port*

The OUT instruction of 8086 is able to output 16-bit data directly in a single cycle, and the programming technique is identical to that of the 8-bit port. The instruction uses 16-bit register AX as source operand as shown:

```
OUT Port_Add, AX
OUT [DX], AX
```

A 16-bit input port may also be interfaced similarly. Note the use of A0 and BHE signals in interfacing the 16-bit ports. One may find out address of the output port in Fig. 5.16.

Till now we have studied some common interfacing methods for I/O ports and have also solved some problems based on I/O ports. A few more problems will be discussed while studying the Programmable Peripheral Input/Output port chip 8255. The ports in this section were either input or output, but in 8255 the same port may be programmed either to work as input port or as output port. Hence the chip is called Programmable Input-Output (PIO) device.

5.4 PIO 8255 [PROGRAMMABLE INPUT-OUTPUT PORT]

The parallel input-output port chip 8255 is also known as programmable peripheral input-output port. The Intel's 8255 is designed for use with Intel's 8-bit, 16-bit and higher capability microprocessors. It has 24 input/output lines which may be individually programmed in two groups of twelve lines each, or three groups of eight lines. The two groups of I/O pins are named as Group A and Group B. Each of these two groups contain a subgroup of eight I/O lines called as 8-bit port and another subgroup of four I/O lines or a 4-bit port. Thus Group A contains an 8-bit port A along with a 4-bit port, C upper. The port A lines are identified by symbols PA_0–PA_7 while the port C lines are identified as PC_4–PC_7. Similarly, Group B contains an 8-bit port B, containing lines PB_0 – PB_7 and a 4-bit port C with lower bits PC_0–PC_3. The port C upper and port C lower can be used in combination as an 8-bit port C. Both the port Cs are assigned the same address. Thus one may have either three 8-bit I/O ports or two 8-bit and two 4-bit I/O ports

from 8255. All of these ports can function independently either as input or as output ports. This can be achieved by programming the bits of an internal register of 8255 called as Control Word Register (CWR). The internal block diagram and the pin configuration of 8255 are shown in Figs 5.17 (a) and (b).

The 8-bit data bus buffer is controlled by the read/write control logic. The read/write control logic manages all of the internal and external transfers of both data and control words. $\overline{RD}$, $\overline{WR}$, A_1, A_0 and RESET are the inputs, provided by the microprocessor to the READ/WRITE control logic of 8255. The 8-bit, 3-state bidirectional buffer is used to interface the 8255 internal data bus with the external system data bus. This buffer receives or transmits data upon the execution of input or output instructions by the microprocessor. The control words or status information is also transferred through the buffer.

The signal descriptions of 8255 are briefly presented as follows:

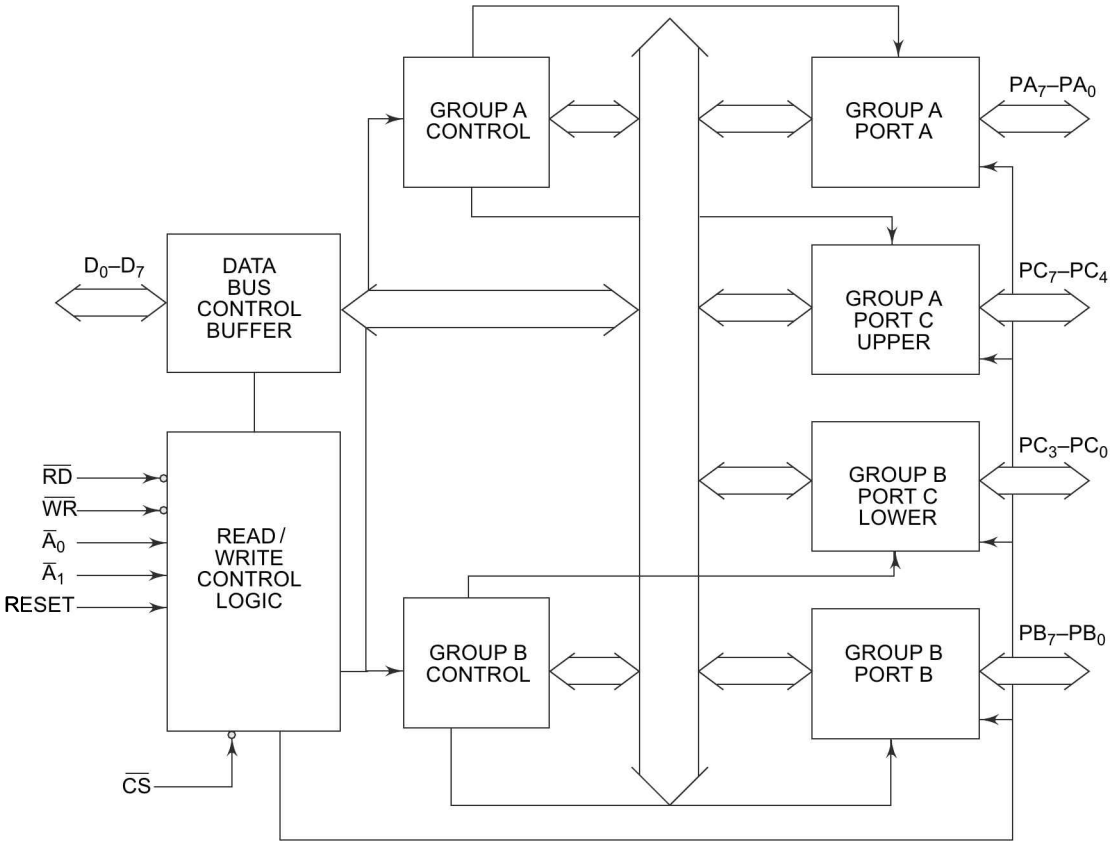


Fig. 5.17(a) *8255 Internal Architecture*

PA_7–PA_0 These are eight port A lines that act as either latched output or buffered input lines depending upon the control word loaded into the control word register.

PC_7–PC_4 Upper nibble of port C lines. They may act as either output latches or input buffers lines. This port also can be used for generation of handshake lines in mode1 or mode 2.

PC_3–PC_0 These are the lower port C lines, other details are the same as PC_7–PC_4 lines.

PB_0–PB_7 These are the eight port B lines which are used as latched output lines or buffered input lines in the same way as port A.

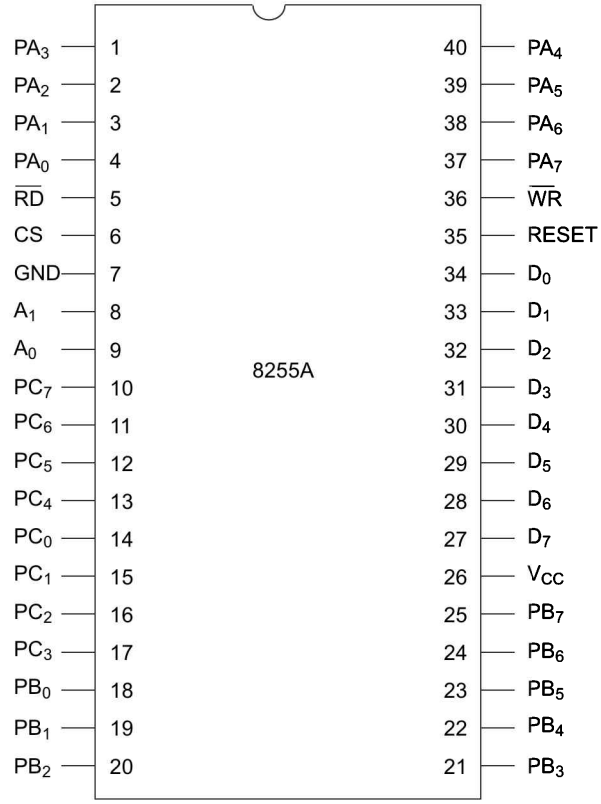


Fig. 5.17(b) *8255A Pin Configuration*

$\overline{RD}$ This is the input line driven by the microprocessor and should be low to indicate read operation to 8255.

$\overline{WR}$ This is an input line driven by the microprocessor. A low on this line indicates write operation.

$\overline{CS}$ This is a chip select line. If this line goes low, it enables the 8255 to respond to $\overline{RD}$ and $\overline{WR}$ signals, otherwise $\overline{RD}$ and $\overline{WR}$ signals are neglected.

A_1–A_0 These are the address input lines and are driven by the microprocessor. These lines ($A_1 - A_0$) with $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ form the following operations for 8255. These address lines are used for addressing any one of the four registers, i.e. three ports and a control word register as given in Tables 5.9 (a), (b) and (c).

In case of 8086 systems, if the 8255 is to be interfaced with lower order data bus, the A_0 and A_1 pins of 8255 are connected with A_1 and A_2 respectively.

Table 5.9(a)

$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	A_1	A_0	*Input (Read) cycle*
0	1	0	0	0	Port A to data bus
0	1	0	0	1	Port B to data bus
0	1	0	1	0	Port C to data bus
0	1	0	1	1	CWR to data bus

Table 5.9 (b)

$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	A_1	A_0	*Output (Write) cycle*
1	0	0	0	0	Data bus to Port A
1	0	0	0	1	Data bus to Port B
1	0	0	1	0	Data bus to Port C
1	0	0	1	1	Data bus to CWR

Table 5.9 (c)

$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	A_1	A_0	*Function*
X	X	1	X	X	Data bus tristated
1	1	0	X	X	Data bus tristated

D_0–D_7 These are the data bus lines those carry data or control word to/from the micro-processor.
RESET A logic high on this line clears the control word register of 8255. All ports are set as input ports by default after reset.

5.5 MODES OF OPERATION OF 8255

There are two basic modes of operation of 8255–I/O mode and Bit Set-Reset mode (BSR). In the I/O mode, the 8255 ports work as programmable I/O ports, while in BSR mode only port C(PC_0–PC_7) can be used to set or reset its individual port bits. Under the IO mode of operation, further there are three modes of operation of 8255, so as to support different types of applications, viz. *mode 0* , *mode 1* and *mode 2*. These modes of operation are discussed in significant details along with application problems in this section, so as to present a clear idea about 8255 operation and interfacing in different modes with 8086.

5.5.1 BSR Mode

In this mode, any of the 8-bits of port C can be set or reset depending on B_0 of the control word. The bit to be set or reset is selected by bit select flags B_3, B_2 and B_1 of the CWR as given in Table 5.10. The CWR format is shown in Fig. 5.18(a).

Table 5.10

B_3	B_2	B_1	*Selected Bits of port C*
0	0	0	B_0
0	0	1	B_1
0	1	0	B_2
0	1	1	B_3
1	0	0	B_4
1	0	1	B_5
1	1	0	B_6
1	1	1	B_7

5.5.2 I/O MODES

MODE 0 (Basic I/O mode) This mode is also known as *basic input/output mode*. This mode provides simple input and output capability using each of the three ports. Data can be simply read from and written to the input and output ports respectively, after appropriate initialisation.

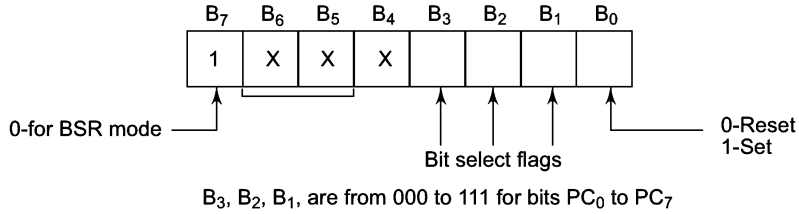


Fig. 5.18(a) *BSR Mode Control Word Register Format*

The salient features of this mode are as listed below:

(i) Two 8-bit ports (port A and port B) and two 4-bit ports (port C upper and lower) are available. The two 4-bit ports can be combinedly used as a third 8-bit port.
(ii) Any port can be used as an input or output port.
(iii) Output ports are latched. Input ports are not latched.
(iv) A maximum of four ports are available so that overall 16 I/O configurations are possible.

All these modes can be selected by programming a register internal to 8255, known as Control Word Register (CWR) which has two formats. The first format is valid for I/O modes of operation, i.e. modes 0, mode 1 and mode 2 while the second format is valid for bit set/reset (BSR) mode of operation. This format is shown in Fig. 5.18(b).

Now let us consider some interfacing problems so as to elaborate the hardware interfacing and I/O programming ideas using 8255 in mode 0.

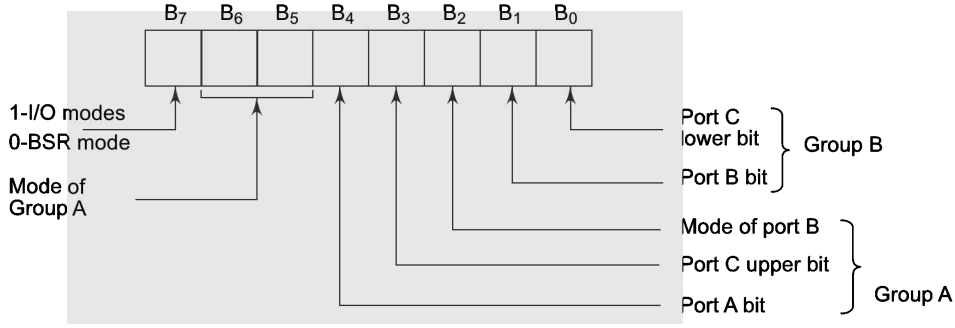
Problem 5.10

Interface an 8255 with 8086 to work as an I/O port. Initialize port A as output port, port B as input port and port C as output port. Port A address should be 0740H. Write a program to sense switch positions SW_0–SW_7 connected at port B. The sensed pattern is to be displayed on port A, to which 8 LEDs are connected, while the port C lower displays number of on switches out of the total eight switches.

Solution The control word is decided upon as follows:

B7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	Control word
1	0	0	0	0	0	1	0	= 82H
I/O	Port A		Port	Port	Port	Port	Port	
mode	in mode 0		A,o/p	C,o/p	B,mode 0	B,i/p	C,o/p	

Thus 82H is the control word for the requirements in the problem. The port address decoding can be done as given below. The 8255 is to be interfaced with lower order data bus, i.e. D_0–D_7. The A_0 and A_1 pins of 8255 are connected to A_{01} and A_{02} pins of the microprocessor respectively. The A_{00} pin of the microprocessor is used for selecting the transfer on the lower byte of the data bus. Hence any change in the status of A_{00} does not affect the port to be selected, rather A_{01} and A_{02} of the microprocessor decide the port to be selected as they are connected to A_0 and A_1 of 8255. The 8255 port addresses are tabulated as shown below.



Group A modes

B_6	B_5	Mode
0	0	mode 0
0	1	mode 1
1	0	mode 2
1	1	x

(i) Port B mode is either 0 or 1 depending upon B2 bit.

(ii) A port is an output port if the port bit is 0 else it is input port

Fig. 5.18(b) *I/O Mode Control Word Register Format*

8255	*I/O Address lines*																*Hex. Port Addresses*
Ports	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_{09}	A_{08}	A_{07}	A_{06}	A_{05}	A_{04}	A_{03}	A_{02}	A_{01}	A_{00}	
PortA	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0740H
Port B	0	0	0	0	0	1	1	1	0	1	0	0	0	0	1	0	0742H
Port C	0	0	0	0	0	1	1	1	0	1	0	0	0	1	0	0	0744H
CWR	0	0	0	0	0	1	1	1	0	1	0	0	0	1	1	0	0746H

Let us use absolute decoding scheme that uses all the 16 address lines for deriving the device address pulse. Out of $A_0 - A_{15}$ lines, two address lines A_{02} and A_{01} are directly required by 8255 for the three port and CWR address decoding. Hence only A_3 to A_{15} are used for decoding addresses. The complete hardware scheme is shown in Fig. 5.19. In the diagram, the 8086 is assumed to be in

the maximum mode so that $\overline{IORD}$ and $\overline{IOWR}$ are readily available. If the 8086 is in minimum mode, $\overline{RD}$ and $\overline{WR}$ of 8086 are to be connected accordingly to 8255 and M/ $\overline{IO}$ pin is combined with the chip select of above hardware suitably so as to select the 8255 when M/$\overline{IO}$ is low.

The ALP for the problem is developed as follows:

```
    MOV DX, 0746 H          ; Initialise CWR with
    MOV AL, 82 H            ; control word 82H
    OUT DX, AL              ;
    SUB DX,04               ; Get address of port B in DX
    IN AL, DX               ; Read port B for switch
    SUB DX,02               ; positions in to AL and get port A address
                            ; in DX.
    OUT DX, AL              ; Display switch positions on port A
    MOV BL, 00 H            ; Initialise BL for switch count
    MOV CH, 08H             ; Initialise CH for total switch number
YY: ROL AL                  ; Rotate AL through carry to check,
    JNC XX                  ; whether the switches are on or
    INC BL                  ; off, i.e. either 1 or 0
XX :DEC CH                  ; Check for next switch. If
    JNZ YY                  ; all switch are checked, the
    MOV AL, BL              ; number of on switches are
    ADD DX, 04              ; in BL.Display it on port C
    OUT DX,AL               ; lower.
    HLT                     ; Stop
```

Program 5.5 *ALP for Problem 5.10*

Problem 5.11

Interface a 4*4 Keyboard with 8086 using 8255. and write an ALP for detecting a key closure and return the key code in AL. The debouncing period for a key is 10 ms. Use software key debouncing technique. DEBOUNCE is an available 10 ms delay routine.

Solution Port A is used as output port for selecting a row of keys while port B is used as an input port for sensing a closed key. Thus the keyboard lines are selected one by one through port A and the port B lines are polled continuously till a key closure is sensed. Then routine DEBOUNCE is called for key debouncing. The key code is decided depending upon the selected row and a low sensed column. The hardware circuit diagram is shown in Fig. 5.21.

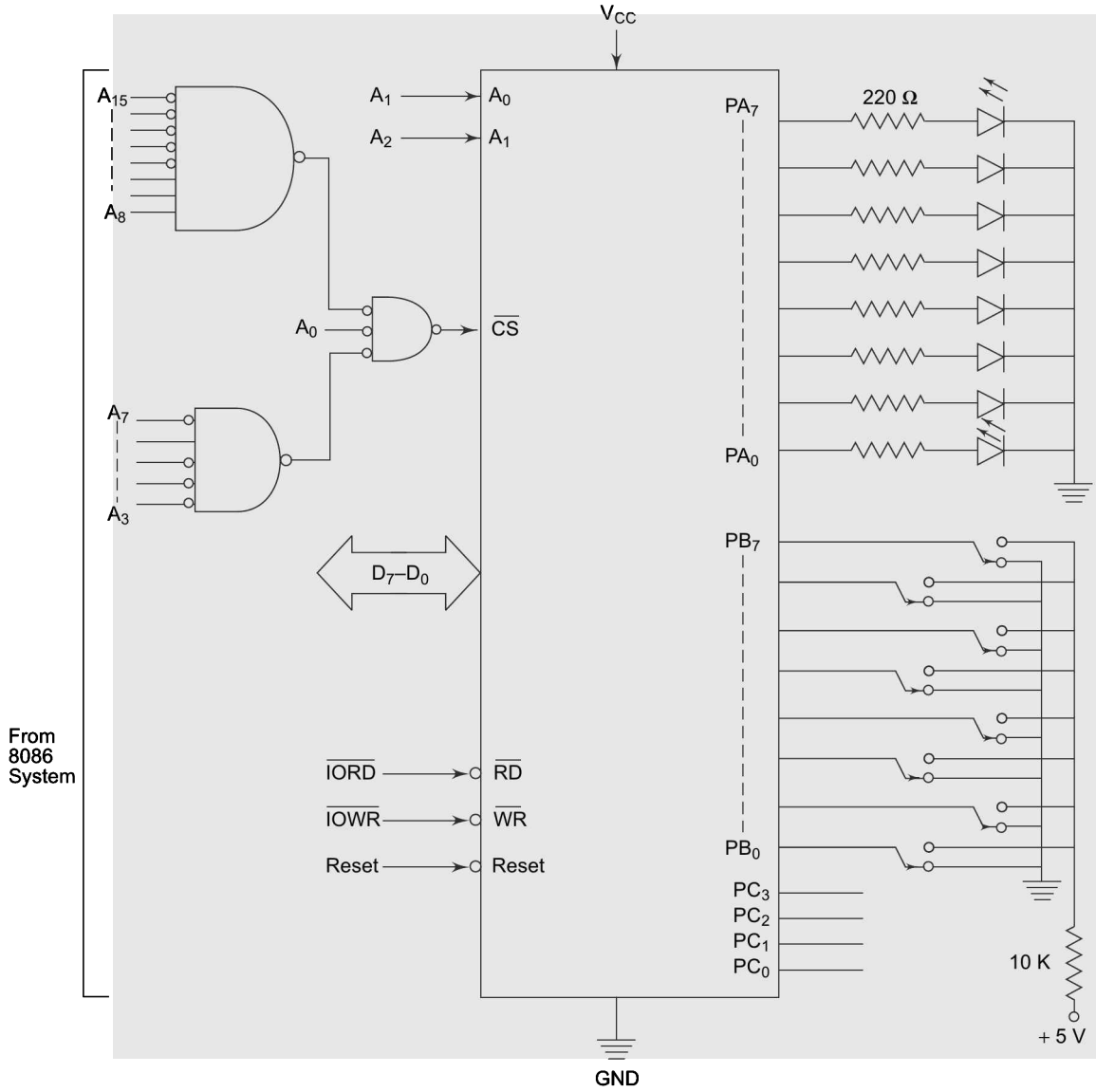


Fig. 5.19 *8255 Interfacing with 8086 for Problem 5.10*

The higher order lines of port A and port B are left unused. The addresses of port A and port B will be respectively 8000 H and 8002 H while the address of CWR will be 8006 H. The flow chart of the complete program is given in Fig. 5.22.

The ALP for the problem is given along with comments. The control word for this problem will be 82 H. Let us write this program using assembler directives. In this problem no major data is required hence only

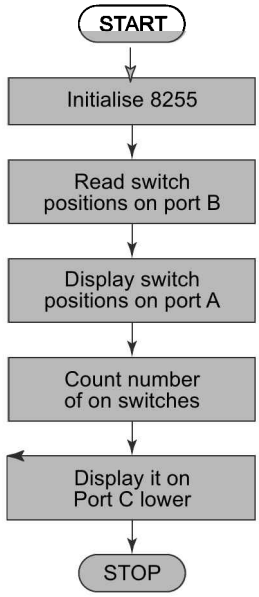


Fig. 5.20 *Flow chart for the ALP of Problem 5.10*

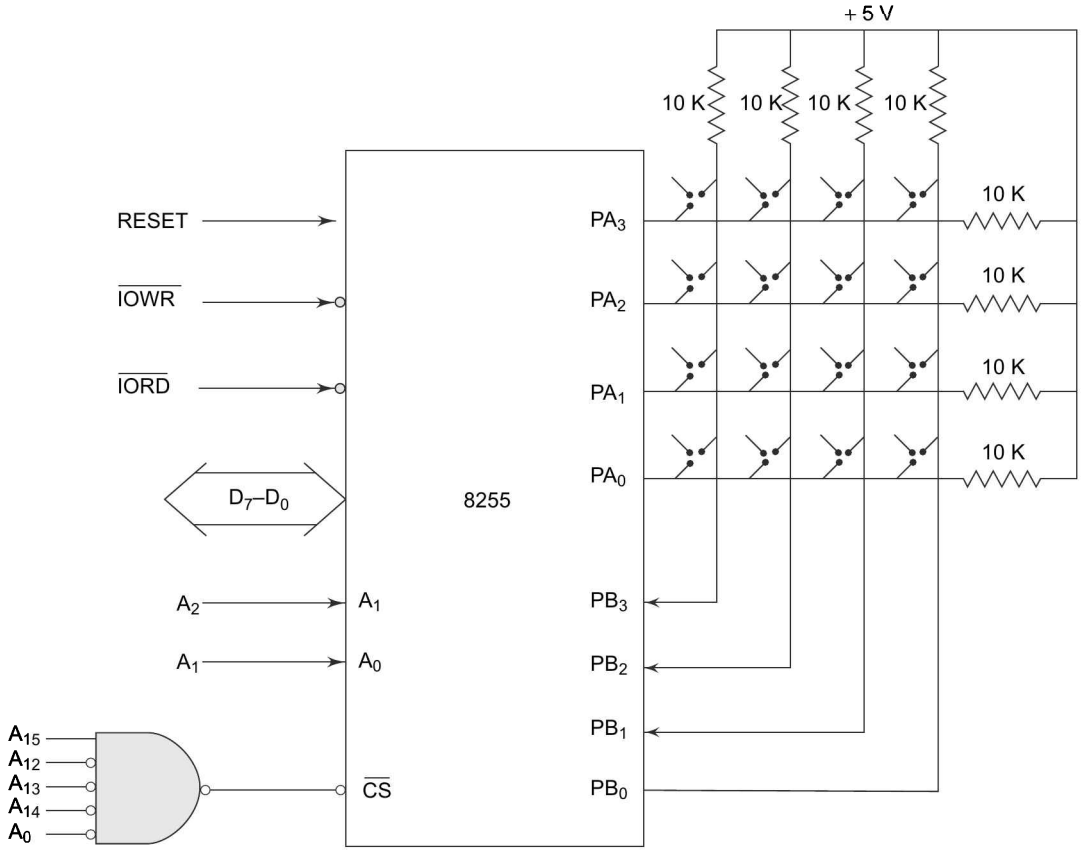


Fig. 5.21 *Interfacing 4 × 4 Keyboard for Problem 5.11*

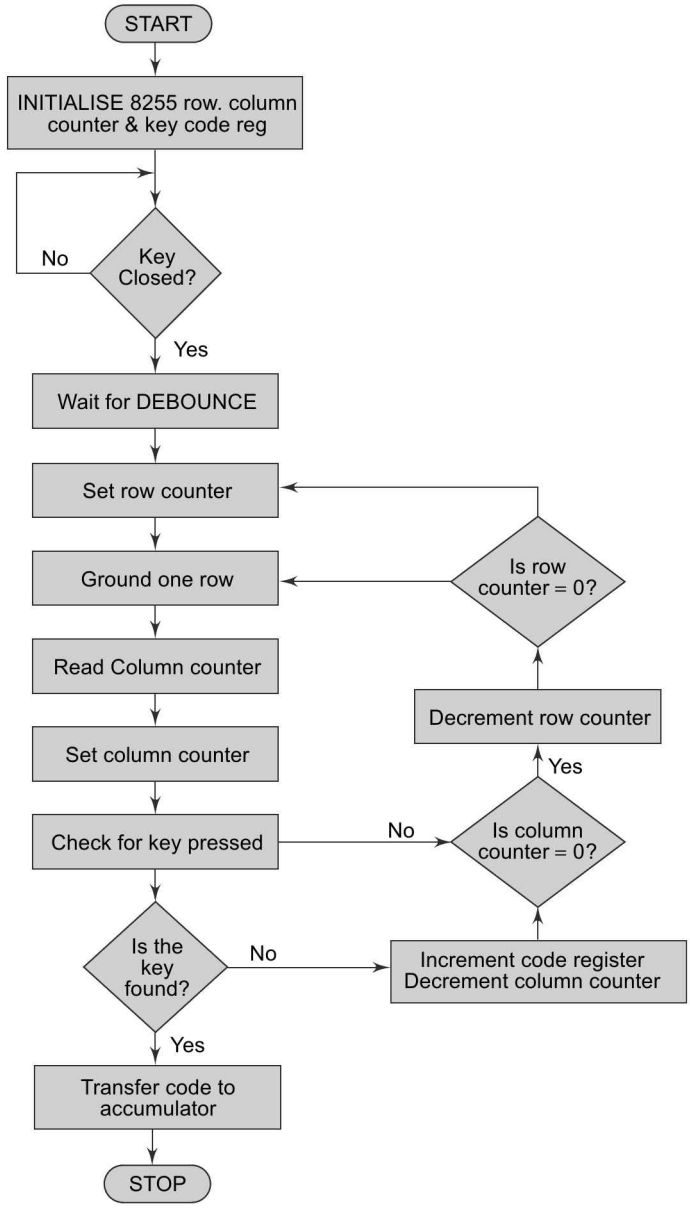


Fig. 5.22 *Flow chart for ALP of Problem 5.11*

one segment is used for storing the program code, i.e. code segment (CS). This program is written in MASM syntax. The 8255 is again interfaced to the lower byte of the 8086 data bus. Absolute decoding scheme is not used here to implement the circuit using minimum hardware.

```
CODE        SEGMENT
ASSUME      CS : CODE
START:      MOV AL, 82H         ; Load CWR with
```

```
            MOV DX, 8006H       ; control word
            OUT DX, AL          ; required
            MOV BL, 00H         ; Initialize BL for key code
            XOR AX, AX          ; Clear all flags
            MOV DX, 8000H       ; Port Address in AX.
            OUT DX, AL          ; Ground all rows.
            ADD DX,02           ; Port B address in DX.
WAIT :      IN AL, DX           ; Read all columns.
            AND AL, 0F H        ; Mask data lines D7-D4.
            CMP AL, 0F H        ; Any key closed?
            JZ WAIT             ; If not, wait till key
            CALL DEBOUNCE       ; closure else wait for 10 ms
            MOV AL, 7FH         ; Load data byte to ground
            MOV BH, 04H         ; a row and set row counter.
NXTROW :    ROL AL, 01          ; Rotate AL to ground next row.
            MOV CH, AL          ; Save data byte to ground next row.
            SUB DX,02           ; Output port address is in DX.
            OUT DX, AL          ; Ground one of the rows.
            ADD DX,02           ; Input port address is in DX.
            IN AL, DX           ; Read input port for key closure.
            AND AL, 0FH         ; Mask D4-D7.
            MOV CL, 04H         ; Set column counter.
NXTCOL :    ROR  AL, 01         ; Move D0 in CF.
            JNC CODEKY          ; Key closure is found, if CF=0.
            INC BL              ; Increment BL for next binary
                                ; key code.
            DEC CL              ; Decrement column counter,
                                ; if no key closure found.
            JNZ NXTCOL          ; Check for key closure in next column
            MOV AL,CH           ; Load data byte to ground next row.
            DEC BH              ; if no key closer found in column
                                ; get ready to ground next row.
            JNZ NXTROW          ; Go back to ground next row.
            JMP WAIT            ; Jump back to check for key.
                                ; closure again.
CODEKY :    MOV AL, BL          ; Key code is transferred to AL.
            MOV AH, 4CH         ; Return to DOS prompt.
            INT 21 H
```

This procedure generates 10 ms delay at 5 MHz operating frequency.

```
DEBOUNCE PROC NEAR
            MOV CL, 0E2H
BACK:       NOP
            DEC CL
            JNZ BACK
            RET
DEBOUNCE    ENDP
CODE        ENDS
            END START
```

Program 5.6 *ALP for Problem 5.11*

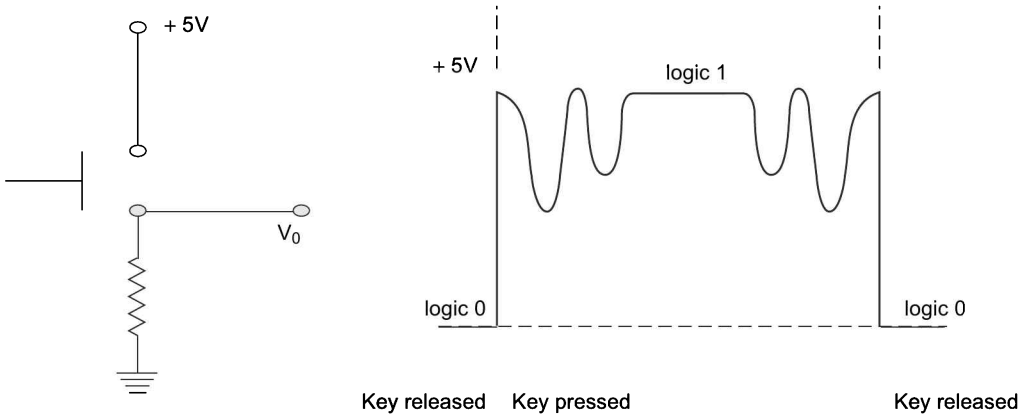


Fig. 5.23 *A Mechanical Key and Its Response*

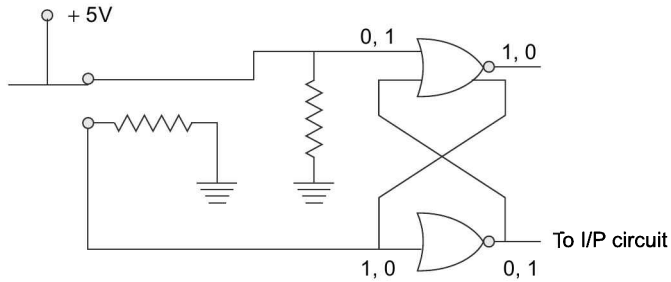


Fig. 5.24 *Hardware Debouncing Circuit*

Key Debounce Whenever a mechanical push-button is pressed or released once, the mechanical components of the key do not change the position smoothly, rather, it generates a transient response as shown in Fig. 5.23. These transient variations may be interpreted as the multiple key pressures and responded accordingly by the microprocessor system. To avoid this problem, two schemes are suggested: the first one utilizes a bistable multivibrator at the output of the key to debounce it as shown in Fig. 5.24. The other scheme suggests that the microprocessor should be made to wait for the transient period (usually 10 msec), so that the transient response settles down and reaches a steady state. A logic '0' will be read by the microprocessor when the key is pressed.

In a number of high precision applications, a designer may need to read or write more than 8-bits of data. In these cases, a system designer may have two options—the first is to have more than one 8-bit port, read(write) the ports one by one and then form the multibyte data; the second option allows forming 16-bit ports using two 8-bit ports and use 16-bit read or write operations. The following example elaborates interfacing of a 16-bit port using two 8-bit ports.

Problem 5.12

Interface 16-bit 8255 ports with 8086. The address of port A is F0H.

Solution To implement a 16-bit port two 8255s are required. One will act as the lower 8-bit port, i.e. D_0–D_7, while the other will act as the upper 8-bit port D_8–D_{15}. The overall scheme is as shown in Fig. 5.25. While initialising AL and AH (AX) both should be loaded with a suitable (common) control word. In this system, port A, port B and port C all may work as 16-bit ports.

Problem 5.13

Interface an 8255 with 8086 at 80H as an I/O address of port A. Interface five 7 segment displays with the 8255. Write a sequence of instructions to display 1, 2, 3, 4 and 5 over the five displays continuously as per their positions starting with 1 at the least significant position.

Solution The hardware scheme for the above problem is shown in Fig. 5.26. In this scheme, I/O port A is multiplexed to carry data for all the 7-segment displays. The port B selects (grounds) one of the displays at a time.

The displays used in the above hardware scheme are common cathod type. To glow a segment, logic 1 is applied on the corresponding line and the corresponding 7-segment display is selected by applying logic 1 on the port line that drives a transistor to ground the common cathode pin of the display. Thus the codes are decided as shown. For a common cathode display, a '1', applied to a segment glows it and a '0' blanks it.

Table 5.11

Number to be displayed	PA_7 *dp*	PA_6 *a*	PA_5 *b*	PA_4 *c*	PA_3 *d*	PA_2 *e*	PA_1 *f*	PA_0 *g*	*Code*
1	1	1	0	0	1	1	1	1	CF
2	1	0	0	1	0	0	1	0	92
3	1	0	0	0	0	1	1	0	86
4	1	1	0	0	1	1	0	0	CC
5	1	0	1	0	0	1	0	0	A4

All these codes, decided as above, are stored in a look up table starting at 2000:0001. The ALP along with comments is given as follows:

```
AGAIN:   MOV CL, 05H              ; Count for displays
         MOV BX, 2000H            ; Initialise data segment
         MOV DS, BX               ; for look up table
         MOV CH, 01H              ; 1st number to be displayed
         MOV AL, 80H              ; Load control word in the
         OUT 86H,AL               ; CWR
         MOV DL,01H               ; Enable code for Least significant
                                  ; 7-seg display
NXTDGT : MOV BX, 0000H            ; Set pointer to look up table
         MOV AL, CH               ; First no to display
                                  ; Store number to be displayed in AL.
         XLAT                     ; Find code from look up table
         OUT 80H,AL               ; Display the code
         MOV AL, DL               ; Enable the display
         OUT 81H,AL               ;
         ROL DL                   ; Go for selecting the next display
```

```
INC CH                    ; Next number to display
DEC CL                    ; Decrement count.
JNZ NXTDGT                ; Go for next digit display
JMP AGAIN                 ; Repeat the procedure
```

Program 5.7 *ALP for Problem 5.13*

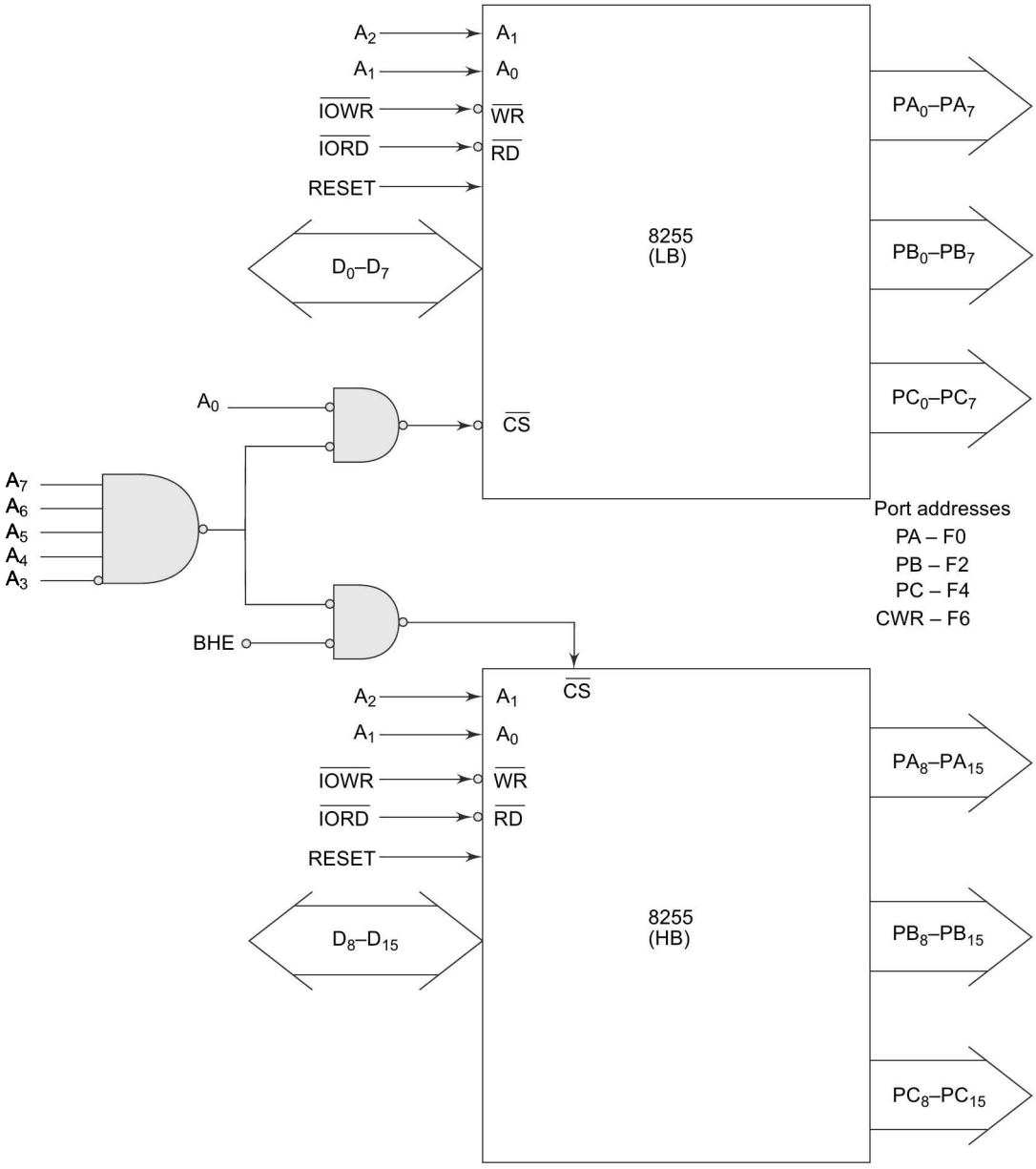


Fig. 5.25 *Interfacing 16-bit 8255 ports with 8086*

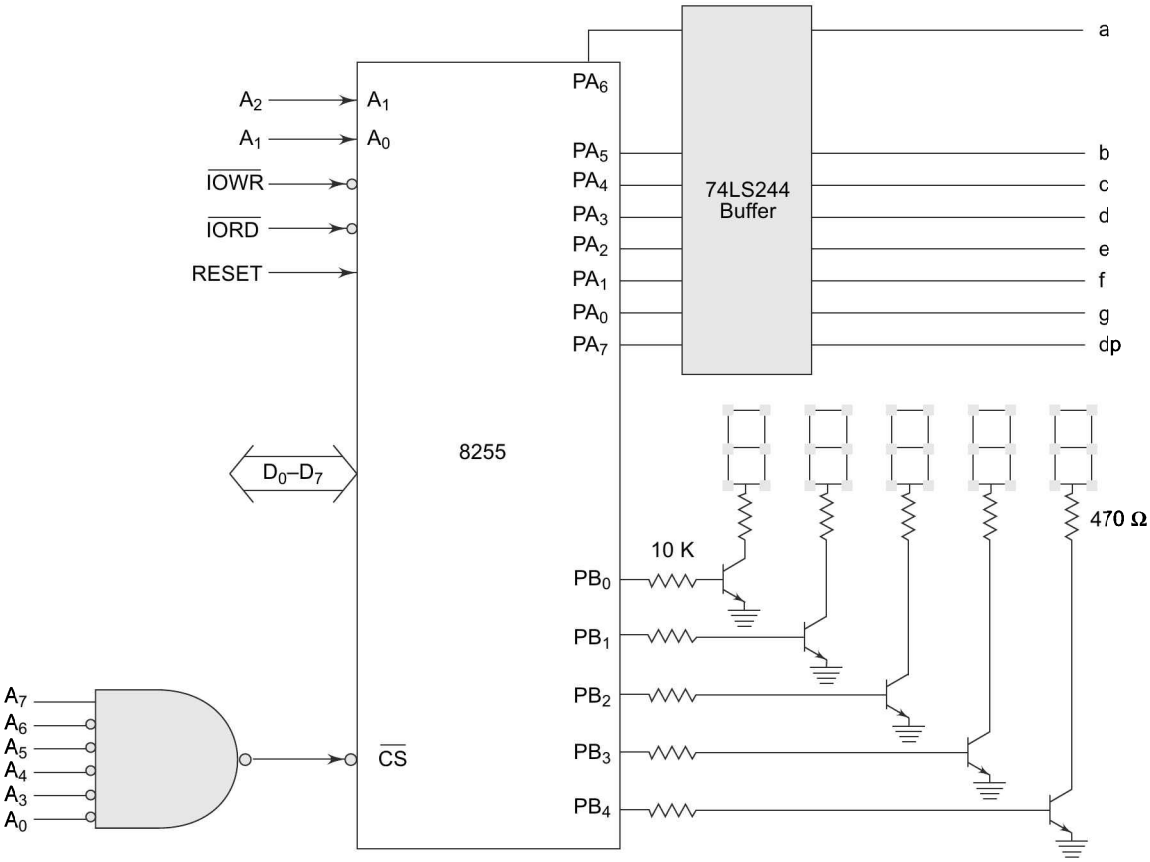


Fig. 5.26 *Interfacing Multiplexed 7-Segment Display*

MODE 1 (Strobed I/O mode) This mode is also called as strobed input/output mode. In this mode the handshaking signals control the input or output action of the specified port. Port C lines PC_0–PC_2, provide strobe or handshake lines for port B. This group which includes port B and PC_0–PC_2 is called as group B for strobed data input/output. Port C lines PC_3–PC_5 provide strobe lines for port A. This group including port A and PC_3–PC_5 forms group A. Thus port C is utilized for generating handshake signals. The salient features of mode 1 are listed as follows:

(i) Two groups—group A and group B are available for strobed data transfer.
(ii) Each group contains one 8-bit data I/O port and one 4-bit control/data port.
(iii) The 8-bit data port can be either used as input or an output port. Both the inputs and outputs are latched.
(iv) Out of 8-bit port C, P C_0–PC_2 are used to generate control signals for port B and PC_3–PC_5 are used to generate control signals for port A. The lines PC_6, PC_7 may be used as independent data lines.

The control signals for both the groups in input and output modes are explained as follows:

Input control signal definitions (mode 1)

$\overline{STB}$ (Strobe input)—If this line falls to logic low level, the data available at 8-bit input port is loaded into input latches.

IBF(Input buffer full)—If this signal rises to logic 1, it indicates that data has been loaded into the latches, i.e. it works as an acknowledgement. IBF is set by a low on **$\overline{STB}$** and is reset by the rising edge of **$\overline{RD}$** input.

INTR (Interrupt request) This active high output signal can be used to interrupt the CPU whenever an input device requests the service. INTR is set by a high at **STB** pin and a high at IBF pin. INTE is an internal flag that can be controlled by the bit set/reset mode of either PC_4 ($INTE_A$) or PC_2 ($INTE_B$) as shown in Figs 5.27(a) and (b). INTR is reset by a falling edge on **RD** input. Thus an external input device can request the service of the processor by putting the data on the bus and sending the strobe signal. Figure 5.27 explains the signal definitions clearly.

The strobed data input cycle waveforms are shown in Fig. 5.28(a).

Output control signal definitions (mode 1)

OBF (Output buffer full)—This status signal, whenever falls to logic low, indicates that the CPU has written data to the specified output port. The OBF flip-flop will be set by a rising edge of **WR** signal and reset by a low going edge at the **ACK** input.

ACK (Acknowledge input)— **ACK** signal acts as an acknowledgement to be given by an output device. **ACK** signal, whenever low, informs the CPU that the data transferred by the CPU to the output device through the port is received by the output device.

INTR (Interrupt request)—Thus an output signal that can be used to interrupt the CPU when an output device acknowledges the data received from the CPU. INTR is set when **ACK**, **OBF** and INTE are '1'. It is reset by a falling edge on **WR** input. The INTEA and INTEB flags are controlled by the bit set-reset mode of PC_6 and PC_2, respectively.

The waveforms in Fig. 5.28 may help in understanding the handshake data transfers. The following Figs 5.29 (a) and (b) explains the signal definitions of mode1 in output mode clearly.

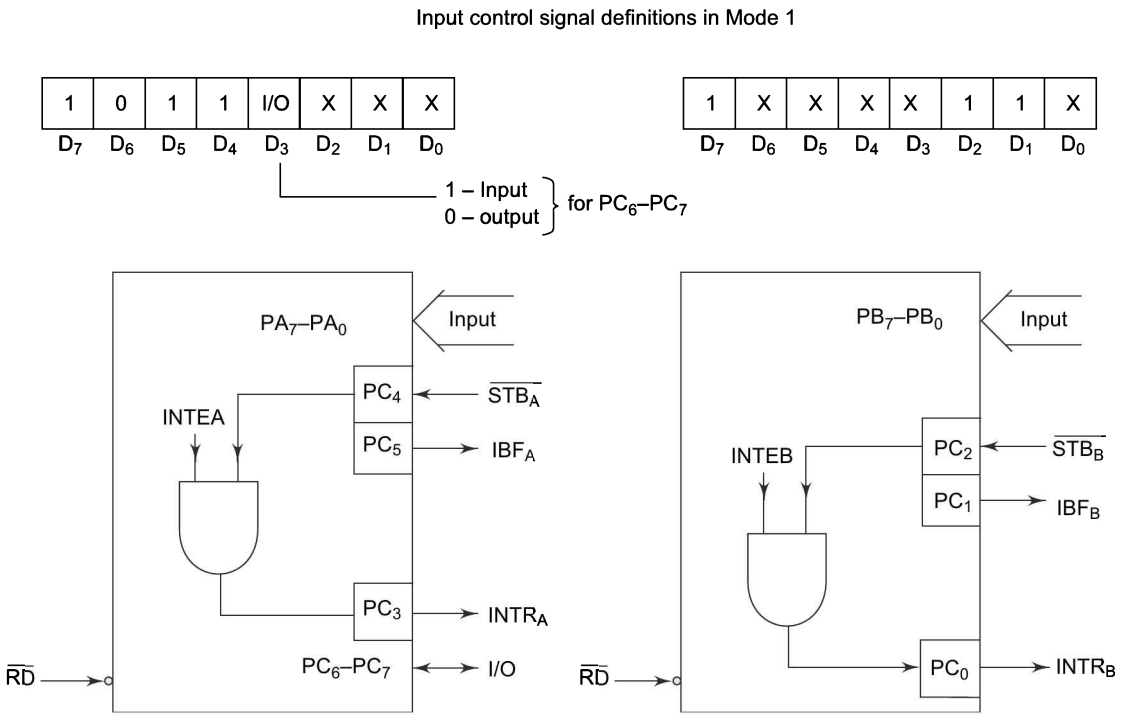


Fig. 5.27 *(a) Mode1 Control Word Group A I/P (b) Mode 1 Control Word Group B I/P*

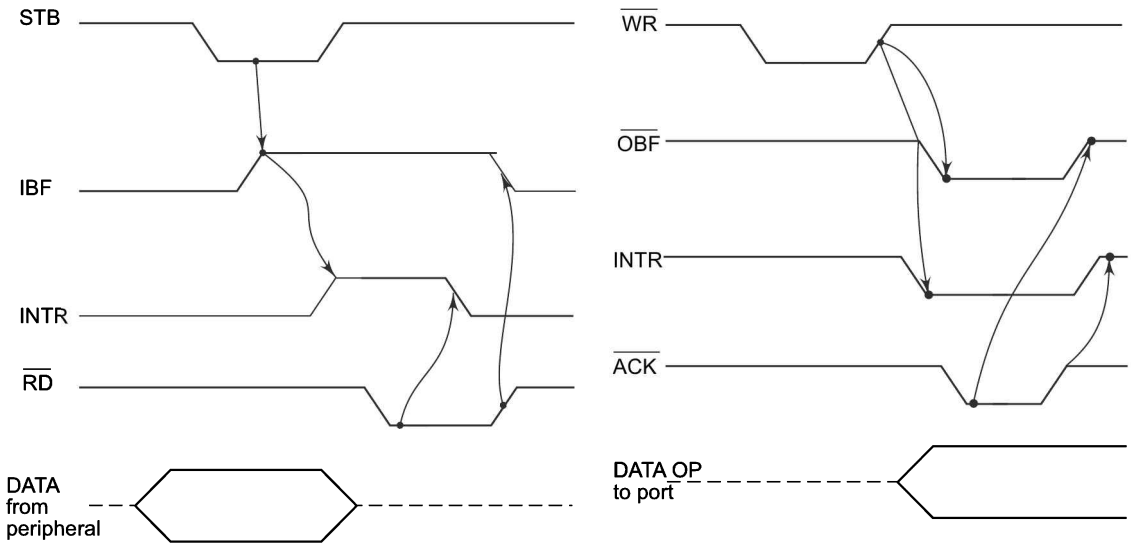


Fig. 5.28 *(a) Mode1 Strobed Input Data Transfer (b) Mode1 Strobed Data Output*

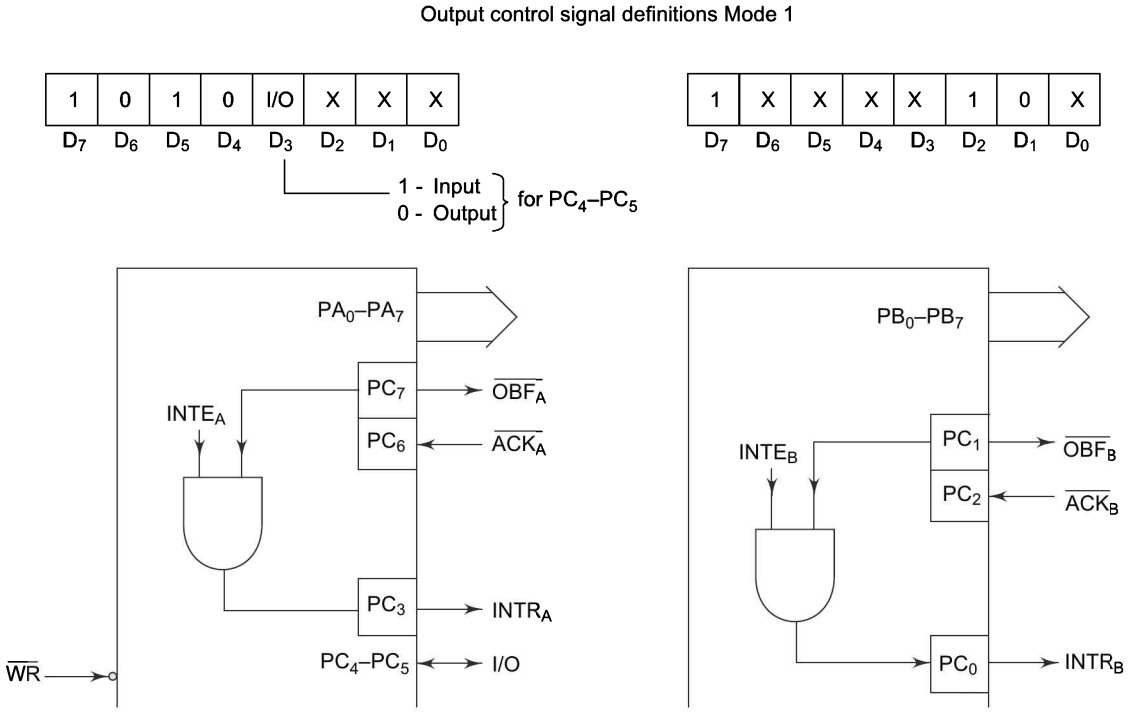


Fig. 5.29 *(a) Mode 1 Control Word Group A o/p (b) Mode 1 Control Word Group B o/p*

After discussing mode 1 in necessary details, let us now consider some hardware interfacing examples that utilize this mode of operation of 8255.

Problem 5.14

Interface a standard IEEE-488 parallel bus printer with 8086. Draw the necessary hardware scheme required for the same and write an ALP to print a character whose ASCII code is available in AL.

Solution Before going through this solution, one should refer to the standard Centronix, INB or EPSON printer pin configuration, given in Table 5.12. There are two types of parallel cables used to connect a microcomputer with a printer, viz. 25 pin cables and 36 pin cables. Basically the 25 pin and the 36 pin cables are similar except for the 11 extra pins for ground (GND) used as 'RETURN' lines for different signals.

The group A is used in mode 1 for handshake data transfer so that port A is used for data transfer and port C lines PC_3–PC_5 are used as handshake lines. Port B lines are used for checking the printer status, hence port B is used as input port in mode 0. Port C lower is used as output port for enabling the printer. The control words are shown in Fig. 5.30.

Table 5.12 *Pin Connections and Descriptions for Centronix-type Parallel Interface to IBM PC and EPSON FX-100 Printers*

Printer Controller				
Signal Pin No.	*Return Pin No.*	*Signal*	*Direction*	*Description*
1	19	$\overline{\text{STROBE}}$	IN	STROBE pulse to read data in. Pulse width must be more than 0.5 μs at receiving terminal. The signal level is normally "high"; read-in of data is performed at the "low" level of this signal.
2	20	DATA 1	IN	These signals represent information of the 1st to 8th bits of parallel data respectively. Each signal is at "high" level when data is logical "1" and "low" when logical "0".
3	21	DATA 2	IN	
4	22	DATA 3	IN	
5	23	DATA 4	IN	
6	24	DATA 5	IN	
7	25	DATA 6	IN	
8	26	DATA 7	IN	
9	27	DATA 8	IN	
10	28	$\overline{\text{ACKNLG}}$	OUT	Approximately 5 μs pulse; "low" indicates the data has been received and the printer is ready to accept other data.
11	29	BUSY	OUT	A "high" signal indicates that the printer cannot receive data. The signal becomes "high" in the following cases. 1. During data entry. 2. During printing operation. 3. In "outline" state. 4. During printer error status.
12	30	PE	OUT	A "high" signal indicates that the printer is out of paper.

(*Contd.*)

Table 5.12 (*Contd.*)

Signal Pin No.	*Return Pin No.*	*Signal*	*Direction*	*Description*
13	—	SLCT	OUT	This signal indicates that the printer is in the selected
14	—	$\overline{\text{AUTO}}$ $\overline{\text{FEED XT}}$	IN	With this signal being at "low" level, the paper is automatically fed one line after printing. (The signal level can be fixed to "low" with DIPSW Pin 2-3 provided on the control circuit board.)
15	—	NC		Not used.
16	—	0V		Logic GND Level.
17	—		CHASIS GND	Printer chasis GND. In the printer, the chasis GND and the logic GND are isolated from each other.
18	—	NC	—	Not used.
19–30	—	GND	—	"Twisted-Pair Return" signal; GND level.
31	—	$\overline{\text{INIT}}$	IN	When the level of this signal becomes "low" the printer controller is reset to its initial state and the print buffer is cleared. This signal is normally at "high" level, and its pulse width must be more than 50 μs at the receiving terminal.
32		$\overline{\text{ERROR}}$	OUT	The level of this signal becomes "low" when the printer is in "Paper End "state, "Offline" state and "Error" state.
33	—	GND	—	Same as with pin numbers 19 to 30.
34	—	NC	—	Not used.
35				Pulled up to + 5 Vdc through 4.7 k-ohms resistance.
36	—	$\overline{\text{SLCT IN}}$	IN	Data entry to the printer is possible only when the level of this signal is "low". (Internal fixing can be carried out with DIP SW 1-8. The condition at the time of shipment is set "low" for this signal.)

Note:
1. "Direction" refers to the direction of signal flow as viewed from the printer.
2. "Return" denotes "Twised-Pair Return" and is to be connected at signal-ground level. When wiring the interface, be sure to use a twisted-pair cable for each signal and never fail to complete connection on the return side. To prevent noise effectively, these cables should be shielded and connected to the chassis of the system unit.
3. All interface conditions are based on TTL, level. Both the rise and fall times of each signal must be less than 0.2 μs.
4. Data transfer must not be carried out by ignoring the $\overline{\text{ACKNLG}}$ or BUSY signal. (Data transfer to this printer can be carried out only after interfacing the $\overline{\text{ACKNLG}}$ signal or when the level of the BUSY signal is "low".)
5. Remaining pins on the connector are no connection pins.
6. x-not available in 25 pins connector.

Printer Operation The printer interface connections with 8255 and the printer connector shown in Fig. 5.31 and Fig. 5.32 respectively. First of all the printer should be initialised by sending a 50 μs (minimum) pulse on the $\overline{\text{INIT}}$ pin of the printer. Then the BUSY pin is to be checked to confirm if the printer is ready. If this signal is low, it indicates that the printer is ready to accept a character from the CPU. Port pins of 8255 may

not have sufficient drive capacity to drive the printer input signals so that the open collector buffers 74LSOY are used to enhance the drive capacity. When this happens the ASCII code of the character to be printed is sent on the eight parallel port lines. Once the data is sent on eight parallel lines, the $\overline{\text{STROBE}}$ signal is activated after at least 0.5 μs, to indicate that the data is available on the eight data lines. The falling edge of the $\overline{\text{STROBE}}$ signal causes the printer to make its BUSY pin high, indicating that the printer is busy. After a minimum period of 0.5 μs, the $\overline{\text{STROBE}}$ signal can be sent high. The data must be valid on the data lines for at least 0.5 μs after the $\overline{\text{STROBE}}$ signal goes high. After receiving the appropriate $\overline{\text{STROBE}}$ pulse, the printer starts the necessary electromechanical action to print the character and when it is ready to receive the next character, it asserts its $\overline{\text{ACKNLG}}$ signal low approximately for 5 ms. The rising edge of the $\overline{\text{ACKNLG}}$ signal indicates to the computer that it is ready to receive the next character. The rising edge of the $\overline{\text{ACKNLG}}$ signal also resets the BUSY signal from the printer. A low on the BUSY pin further indicates that the printer is ready to accept the next character. The $\overline{\text{ACKNLG}}$ and BUSY signals can be used interchangeably for handshaking purposes. The waveforms for the above printer operation are shown in Fig. 5.33.

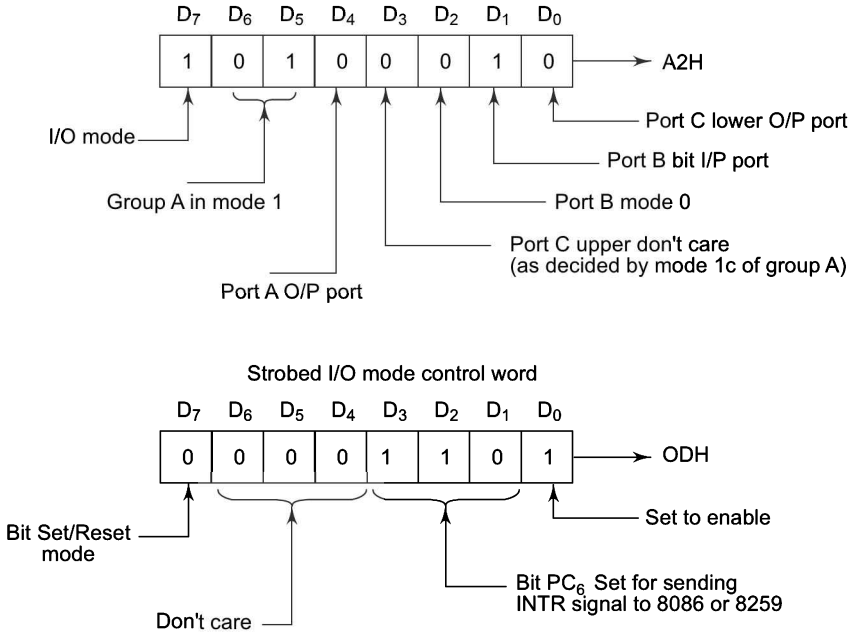


Fig. 5.30 *Bit Set/Reset Control Word*

```
        MOV BL,AL          ; Get the ASCII code in BL.
        MOV AL,0A2H        ; Control word for 8255
        OUT 0F6H,AL        ; Load CWR with the control word.
BUSY:   IN AL,0F2H         ; Read printer status from the BUSY pin.
        AND AL,08H         ; Mask all bits except PB3
        JNZ BUSY           ; If AL#0, printer is busy. Wait till
                           ; it becomes free.
        MOV AL, BL         ; Get the character for printer in AL
        OUT 0F0H,AL        ; Send it to the port for the printer
        NOP                ; Wait for some time
        MOV AL,08 H        ; Pull STROBE low
```

```
OUT OF6H            ; Reset PC4
NOP                 ; Wait
MOV AL,09 H         ; Raise STROBE high
OUT OF6H            ; SET PC4
HLT
```

Program 5.8 *ALP for Printing a Character for Problem 5.14*

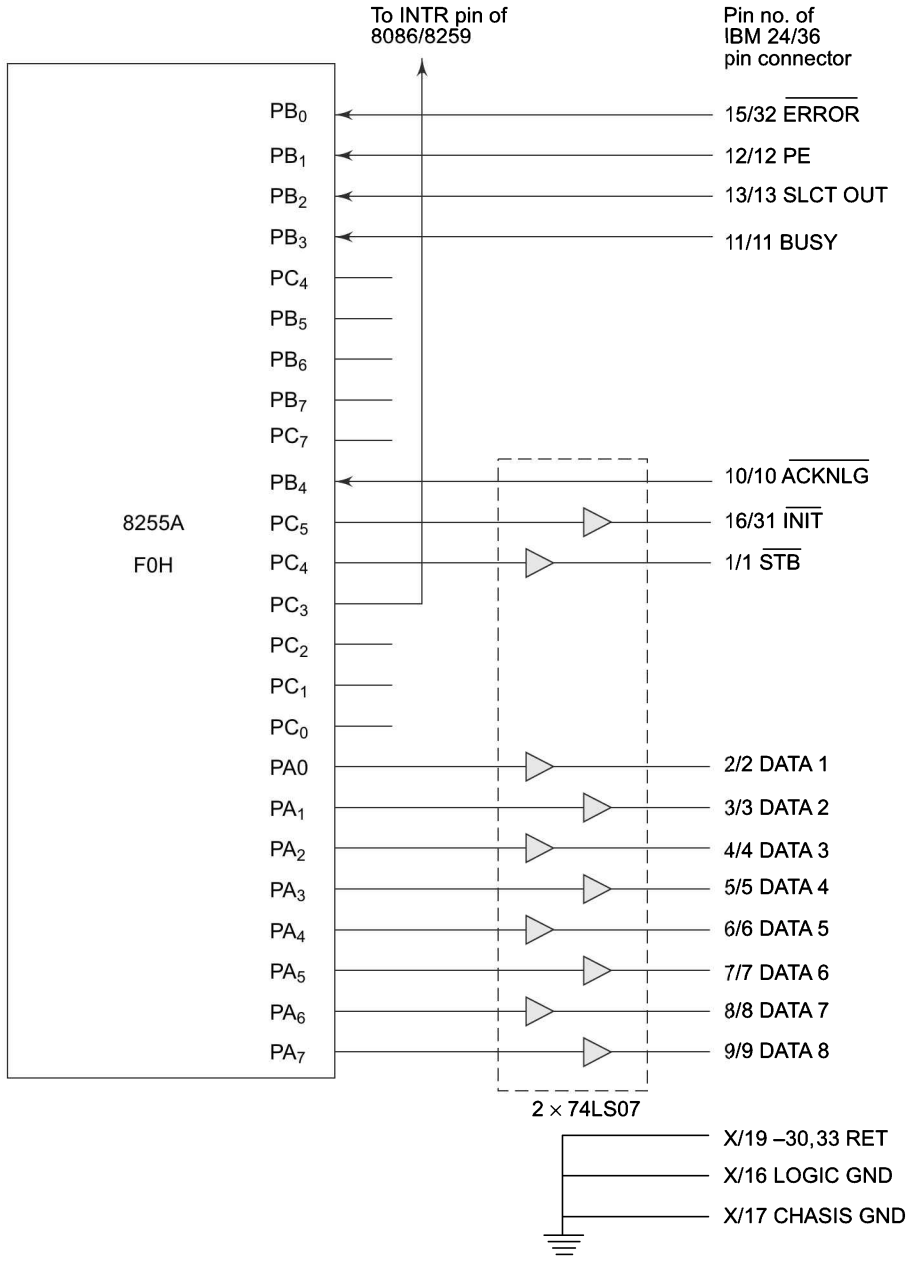


Fig. 5.31 *Printer Interface with 8255*

MODE 2 (Strobed bidirectional I/O) This mode of operation of 8255 is also known as *strobed bidirectional I/O* . This mode of operation provides 8255 with an additional feature for communicating with a peripheral device on an 8-bit data bus. Handshaking signals are provided to maintain proper data flow and synchronization between the data transmitter and receiver. The interrupt generation and other functions are similar to mode 1. Thus in this mode, 8255 is a bidirectional 8-bit port with handshake signals. The $\overline{RD}$ and $\overline{WR}$ signals decide whether the 8255 is going to operate as an input port or output port.

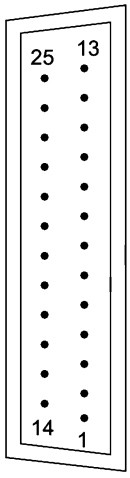


Fig. 5.32 *Centronics Printer Connector*

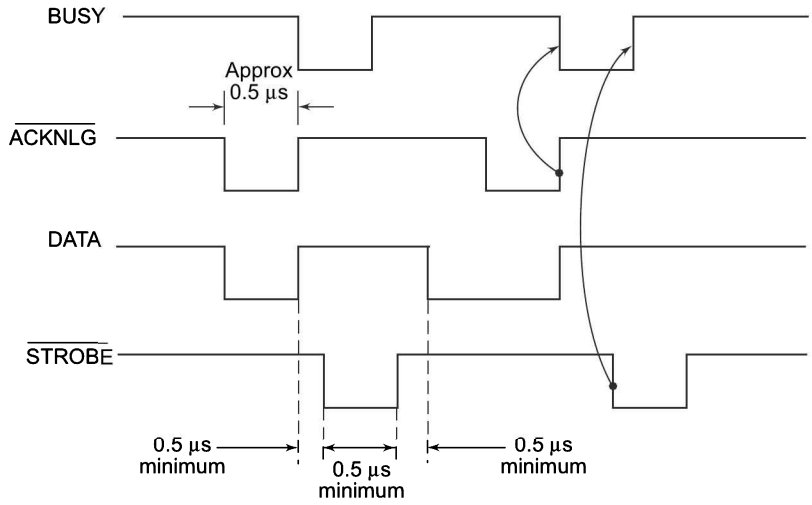


Fig. 5.33 *Timing Waveforms of Data Transfer to a Centronix Compatible Parallel Printer*

The salient features of mode 2 of 8255 are listed as follows:

1. The single 8-bit port in group A is available.
2. The 8-bit port is bidirectional and additionally a 5-bit control port is available.
3. Three I/O lines are available at port C,viz. PC_2–PC_0.
4. Inputs and outputs are both latched.
5. The 5-bit control port C (PC_3–PC_7) is used for generating/accepting handshake signals for the 8-bit data transfer on port A.

Control signal definitions in mode 2

INTR (Interrupt request) As in mode 1, this control signal is active high and is used to interrupt the microprocessor to ask for transfer of the next data byte to/from it. This signal is used for input (read) as well as output (write) operations.

Control signals for output operations

$\overline{OBF}$ (Output buffer full) This signal, when falls to logic low level, indicates that the CPU has written data to port A.

$\overline{ACK}$ (Acknowledge) This control input, when falls to logic low level, acknowledges that the previous data byte is received by the destination and the next byte may be sent by the processor. This signal enables the internal tristate buffers to send out the next data byte on port A.

INTE1 (A flag associated with $\overline{OBF}$) This can be controlled by bit set/reset mode with PC_6.

Control signals for input operations

$\overline{STB}$ (Strobe input) A low on this line is used to strobe in the data into the input latches of 8255.

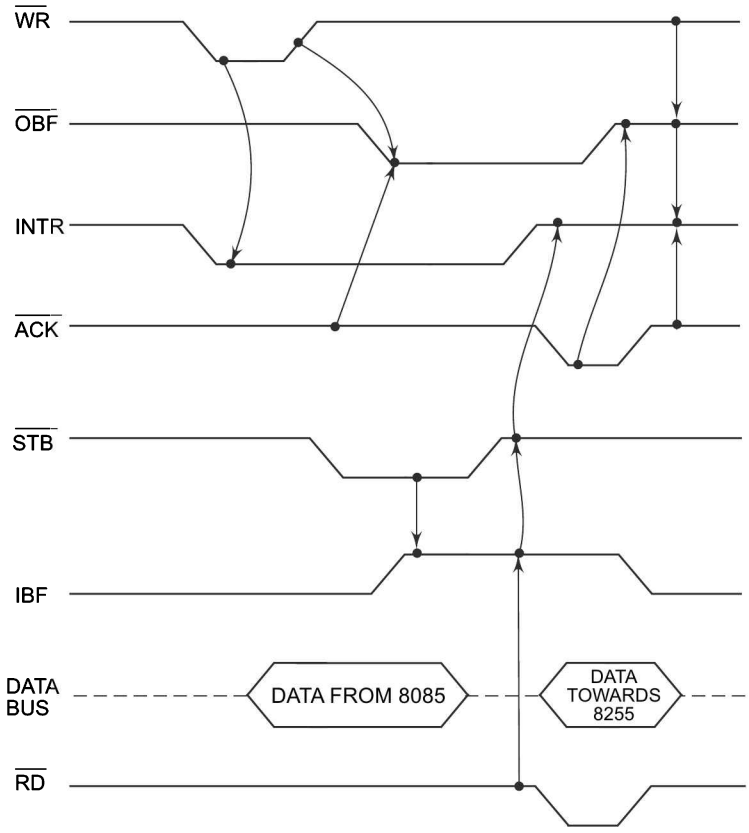


Fig. 5.34 *Mode 2 Bidirectional Data Transfer*

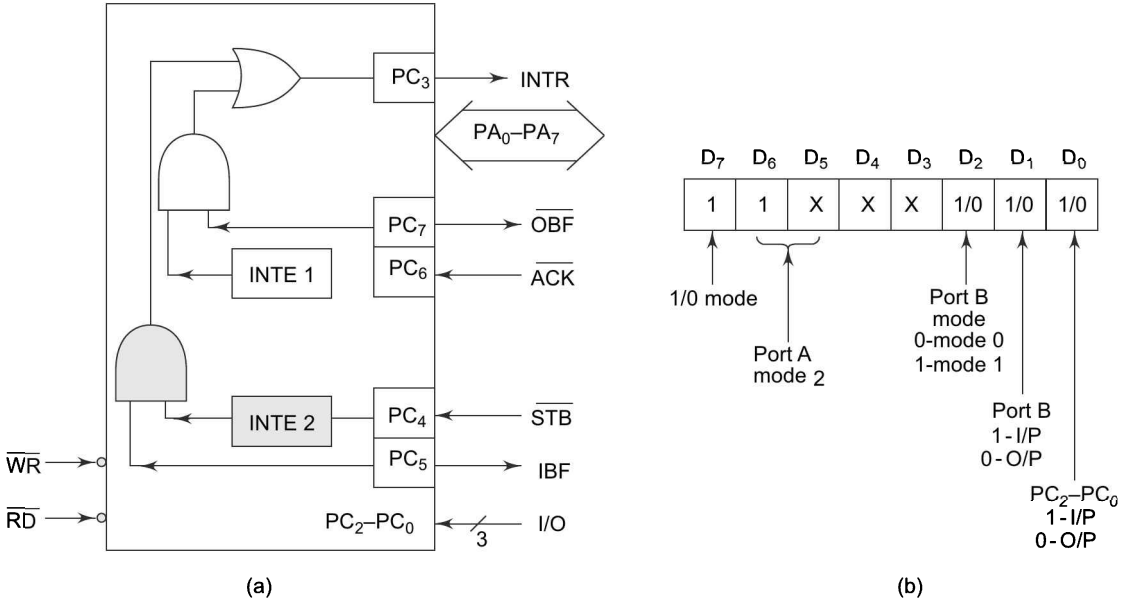


Fig. 5.35 *(a) Mode 2 pins (b) Mode 2 control word*

IBF (Input buffer full) When the data is loaded into the input buffer, this signal rises to logic '1'. This can be used as an acknowledgement that the data has been received by the receiver.

The waveforms in Fig. 5.34 show the operation in mode 2 for output as well as input port.

Note: $\overline{WR}$ must occur before $\overline{ACK}$ and $\overline{STB}$ must be activated before $\overline{RD}$.

Figure 5.35 (a) shows a schematic diagram containing an 8-bit bidirectional port, 5-bit control port and the relation of INTR with the control pins. Port B can either be set to mode 0 or mode 1 while port A (Group A) is in mode 2. Mode 2 is not available for port B. Figure 5.35 (b) shows the necessary control word.

The INTR goes high only if either IBF, INTE2, $\overline{STB}$ and $\overline{RD}$ go high or $\overline{OBF}$, INTE1, $\overline{ACK}$ and $\overline{WR}$ go high. The port C can be read to know the status of the peripheral device, in terms of the control signals, using the normal I/O instructions.

The following problem emphasizes the use of 8255 in mode 2.

The programming in assembly language is divided into two parts. The first is the transmitter or sender part and the second being the receiver part. The transmitter program sends data to the receiver 8086 system while the receiver program accepts the data from the transmitting 8086 system. To interchange the blocks of data between the two systems, each one will require the transmitter as well as the receiver program.

The interrupt vector address for NMI is 0000 : 0008H. At this location the IP and CS values of the transmitter program addresses are stored. The execution of the transmitter program on one system causes execution of the receiver program on the other system through NMI interrupt. Rather, the receiver is executed as the NMI interrupt service routine as a response to the execution of the transmitter program.

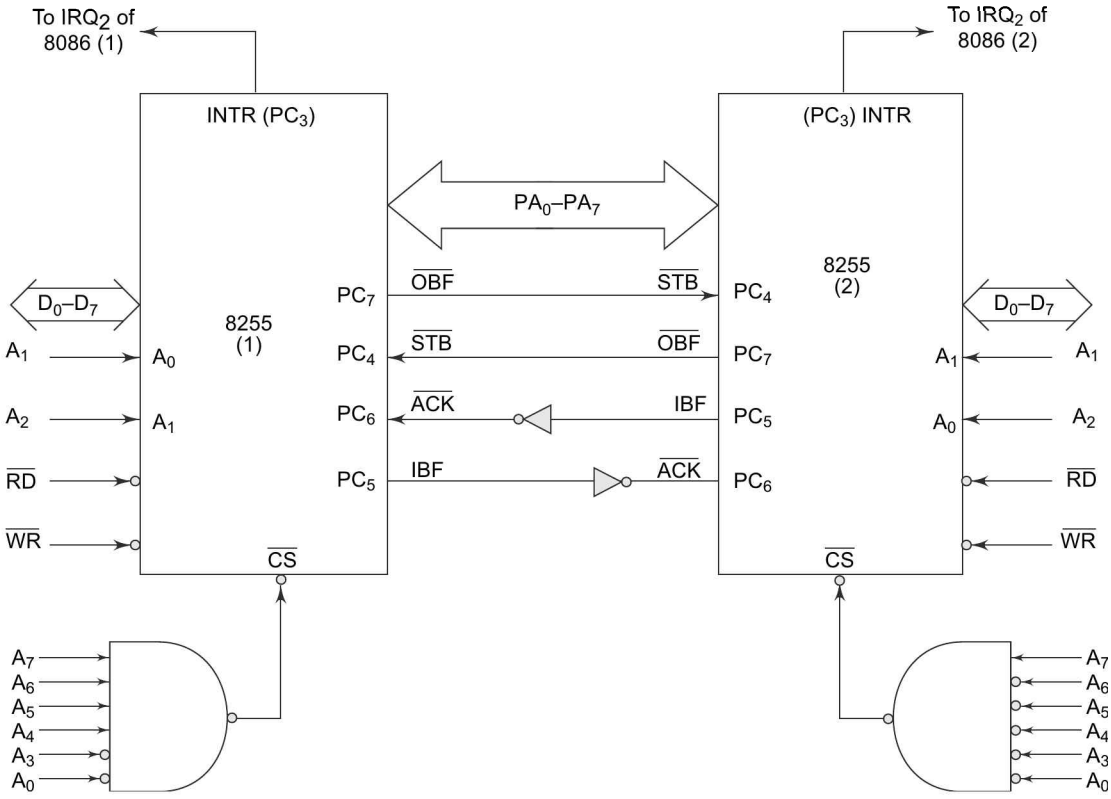


Fig. 5.36 *Interconnections between the two 8255s in Mode 2 for Problem 5.16*

Problem 5.15

An 8086 system with a 8255 interfaced at port A address F0H, has a block of 100 data bytes stored in it. Another 8086 system with another 8255 interfaced at port A address 80H has another block of 100 data bytes stored in it. Interchange these blocks of data bytes between the two 8086 systems. Draw the necessary hardware scheme and write the necessary sequence of instructions. Both systems run on the same CLK rate.

Solution The complete hardware schematic is shown in Fig. 5.36. The INTR pin of 8255 in mode 2 is applied to the respective 8086 processor at NMI pin. The inverted IBF signal of the first 8255 is connected to $\overline{\text{ACK}}$ of the second and the inverted IBF signal of the second 8255 is connected to $\overline{\text{ACK}}$ of the first 8255, so that input buffer full signal of one 8255 acknowledges the receipt of data byte sent by the other. The $\overline{\text{OBF}}$ signal of one 8255 is connected to $\overline{\text{STB}}$ signal of the other 8255 and vice versa, so that the output buffer full signal of an 8255 informs the other 8255 that the data is ready on the data bus for it. The 8-bit data bus, i.e. PA_0–PA_7 of the two 8255s are connected with each other.

```
            ; This program transmits parallel data byte by byte to another
              system through
            ; 8255 IN MODE 2.
DATA        SEGMENT
CW1         EQU COH
BLOCK1 DB   100D DUP (?)
DATA        ENDS
ASSUME      CS : CODE,DS : DATA
CODE        SEGMENT
            MOV AX, 0000H              ; Initialise interrupt
            MOV DS, AX                 ; vector table of
            MOV AX, OFFSET TRANS       ; 8086 at table
            MOV [0008H],AX ; TRANS.
            MOV [000AH],SEG TRANS
            MOV CL,101D                ; count CL (one additional)
            MOV AX, DATA               ; Initialise data segment
            MOV DS,AX
            MOV AL, CW1                ; Initialise 8255 in
            MOV DS, AX                 ; mode 2
            OUT F6H, AL                ;
            STI                        ;
            MOV [SI], OFFSET BLOCK1-1
TRANS :     INT 2
            CALL FAR PTR SYNCHRO       ; Wait for synchronization
            INC SI                     ; Pointer to block in SI
            DEC CL                     ; Decrement count
```

```
          JZ STOP                    ; If = 0, then stop else
          MOV AL,[SI]                ; Go for transfer of the next
          OUT FOH,AL                 ; byte and out it to port A
WAIT :    JMP WAIT                   ; Wait for acknowledgement
STOP :    HLT                        ; Stop if the complete block
CODE      ENDS                       ; is transferred
          END
```

Program 5.9(a) *Transmitter ALP for Problem 5.15*

```
          ; The receiver program receives data bytes
          ; transmitted by the other system and stores
          ; them in the array as asked in the program.
STACK           SEGMENT
          STACKD DB   500H
STACK           ENDS
DATA            SEGMENT
          CW2 EQU COH
          BLOCK2 DB 100D DUP (?)
DATA            ENDS
CODE            SEGMENT
ASSUME CS : CODE, DS : DATA ,SS: STACK
            MOV AX, STACK
            MOV SS, AX
            MOV AX, 0000H          ; Initialise interrupt
            MOV DS, AX             ; vector table
            MOV [0008H],OFFSET NEXT
            MOV [000AH],SEG NEXT
            MOV CL,101 D           ; count for bytes
            MOV AX,DATA            ; Initialise data segment
            MOV DS,AX
            MOV AL,CW2             ; Initialise 8255 in mode 2
            OUT 86H,AL             ; to receive data
            MOV SI,OFFSET BLOCK2-1 ; Point to block 2
WAIT :      JMP WAIT               ; to store received data and wait
NEXT :      INC SI                 ; Increment SI, point to start
            DEC CL                 ; of block 2 and decrement
                                   ; COUNTER
            JZ STOP
            IN 80H
            MOV [SI],AL
            JMP WAIT
STOP :      HLT
CODE        ENDS
      END
```

Program 5.9(b) *Receiver ALP for Problem 5.15*

After the transmitter transmits the data bytes, the receiver receives it and stores it in the array BLOCK 2 but acknowledge $\overline{\text{ACK}}$ is sent to the transmitter immediately. Hence the transmitter should wait before sending the next data byte to the receiver so as to give it the sufficient time to read, store and upgrade count for the received data. Transmitter runs its delay procedure SYNCHRO to solve this problem.

```
SYNCHRO    PROC   FAR
    INC DI              ; All the instructions in this routine are dummy
    DEC CH              ; instructions, just to cause the same delay as the
    JZ OK               ; requires takes, to be prepared for accepting
                              the next data
                        ; byte after getting interrupted by the
                                    transmitter.
OK :IN AL,0F3H
    MOV [DI] AL         ; These are comparable with the
    IRET                ; corresponding receiver program
SYNCHRO    ENDP         ; instructions.
```

Program 5.10 *ALP for Synchronization Delay*

It may be noted that for the execution of this program, procedure SYNCHRO must be entered with the transmitter program before the END statement.

These programs should be entered in both the 8086 systems. The procedure SYNCHRO is to be entered with the transmitter program as it is called by it. Thus after entering the complete set of these programs into the two 8086 systems, run the receiver program on the receiver 8086 kit. It will initialise the receiver 8086 IVT, its 8255 in receiver mode and wait for the interrupt from the transmitting terminal. Now run the program on the transmitter 8086 kit. This program initialises the transmitter 8086 IVT. Its 8255, in transmitter mode, goes on transmitting data byte by byte and waits for the delay caused by the SYNCHRO before each byte is transmitted so as to give sufficient time to the receiver 8086 to read data, store it in array and upgrade counters and pointers.

Note that the procedure SYNCHRO contains all the dummy instructions. They do not serve any purpose for the algorithm, but just provide their execution delay.The instructions in the procedure are the same as the instructions in the receiving program, from label NEXT to the JMP WAIT instruction. This is not merely a coincidence but an accurate way of providing the required delay for the receiver. As both the 8086 CPUs run at the same clock speed, each of the instructions, which from the procedure and correspondingly those from the receiver program will take the same time for execution. Thus the transmitter will provide the exact delay as required by the receiver.

5.6 INTERFACING ANALOG TO DIGITAL DATA CONVERTERS

This topic is aimed at the study of 8-bit and 12-bit analog to digital converters and their interfacing with 8086. In most of the cases, the PIO 8255 is used for interfacing the analog to digital converters with a microprocessor. We have already studied 8255 interfacing with 8086 as an I/O port, in the previous section. This section will only emphasize the interfacing techniques of analog to digital converters with 8255.

The analog to digital converter is treated as an input device by the microprocessor, that sends an initialising signal to the ADC to start the analog to digital data conversion process. The start of conversion signal is a pulse of a specific duration . The process of analog to digital conversion is a slow process, and the microprocessor has to wait for the digital data till the conversion is over. After the conversion is over, the ADC sends end of conversion (EOC) signal to inform the microprocessor about it and the result is ready at the output

buffer of the ADC. These tasks of issuing an SOC pulse to ADC, reading EOC signal from the ADC and reading the digital output of the ADC are carried out by the CPU using 8255 I/O ports.

The time taken by the ADC from the active edge of SOC pulse (the edge at which the conversion process actually starts) till the active edge of EOC signal is called as the *conversion delay* of the ADC. Or broadly speaking, the time taken by the converter to calculate the equivalent digital data output from the moment of the start of conversion is called conversion delay. It may range anywhere from a few microseconds, in case of fast ADCs, to even a few hundred milliseconds in case of slow ADCs. A number of ADCs are available in the market. The selection of ADC for a particular application is done, keeping in mind the required speed, resolution and the cost factor. The available ADCs in the market use different conversion techniques for the conversion of analog signals to digital signals. Successive approximation and dual slope integration techniques are the most popular techniques used in the integrated ADC chips. Whatever may be the technique used for conversion, a general algorithm for ADC interfacing contains the following steps.

1. Ensure the stability of analog input, applied to the ADC
2. Issue start of conversion (SOC) pulse to ADC
3. Read end of conversion (EOC) signal to mark the end of conversion process
4. Read digital data output of the ADC as equivalent digital output

It may be noted that the analog input voltage must be a constant at the input of the ADC right from the beginning to the end of the conversion to get correct results. This may be ensured by a sample and hold circuit which samples the analog signal and holds it constant for a specified time duration. The microprocessor may issue a hold signal to the sample and hold circuit. If the applied input changes before the complete conversion process is over, the digital equivalent of the analog input calculated by the ADC may not be correct.

In this section, we are going to study a few ADC chips and their interfacing techniques with 8086. The first is the ADC0808 an 8-bit ADC and the second is ICL7109, Intersil's 12-bit dual slope ADC. Before proceeding with the interfacing part each of the chip is discussed in significant details so that the interfacing circuits and the algorithms can clearly be understood.

5.6.1 ADC 0808/0809

The analog to digital converter chips 0808 and 0809 are 8-bit CMOS, *successive approximation converters*. Successive approximation technique is one of the fastest technique used for the process of analog to digital conversion. The conversion delay is 100 μs at a clock frequency of 640 kHz, which is quite low as compared to other converters. These converters do not need any external zero or full scale adjustments as they are already taken care of by internal circuits. These converters internally have a 3:8 analog multiplexer so that at a time eight different analog inputs can be connected to the chips. Out of these eight inputs only one can be selected for conversion by using address lines ADD A, ADD B and ADD C, as shown. Using these address inputs, multichannel data acquisition systems can be designed using a single ADC. The CPU may drive these lines using output port lines in case of multichannel applications. In case of single input applications, these may be hardwared to select the proper input.

Table 5.13

Analog I/P selected	*Address lines*		
	C	*B*	*A*
I/P 0	0	0	0
I/P 1	0	0	1
I/P 2	0	1	0
I/P 3	0	1	1

(*Contd.*)

Table 5.13 (*Contd.*)

Analog I/P selected	*Address lines*		
	C	*B*	*A*
I/P 4	1	0	0
I/P 5	1	0	1
I/P 6	1	1	0
I/P 7	1	1	1

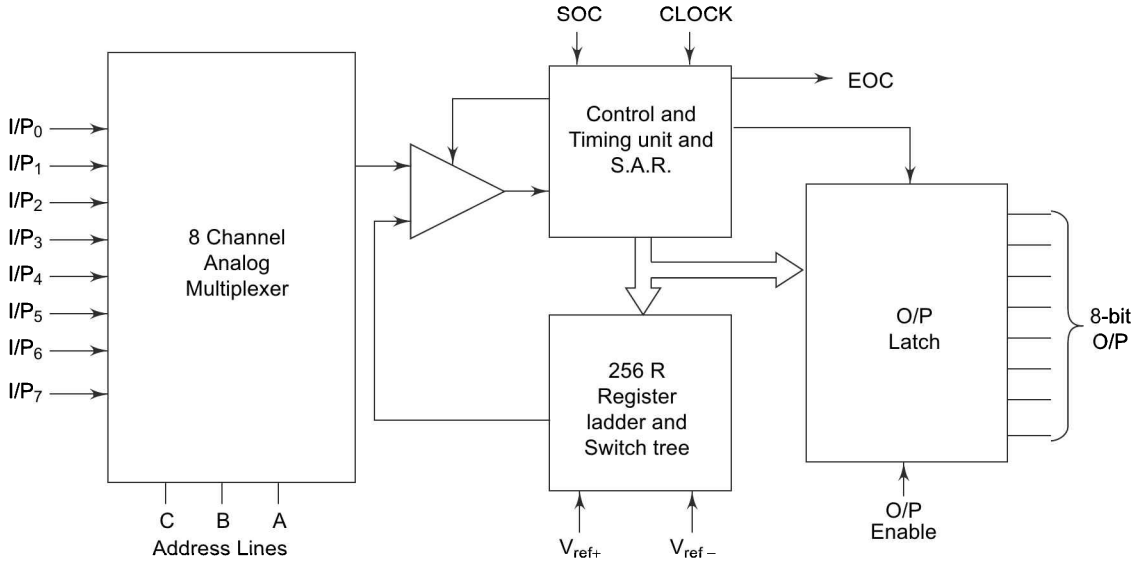


Fig. 5.37(a) *Block Diagram of ADC 0808/0809*

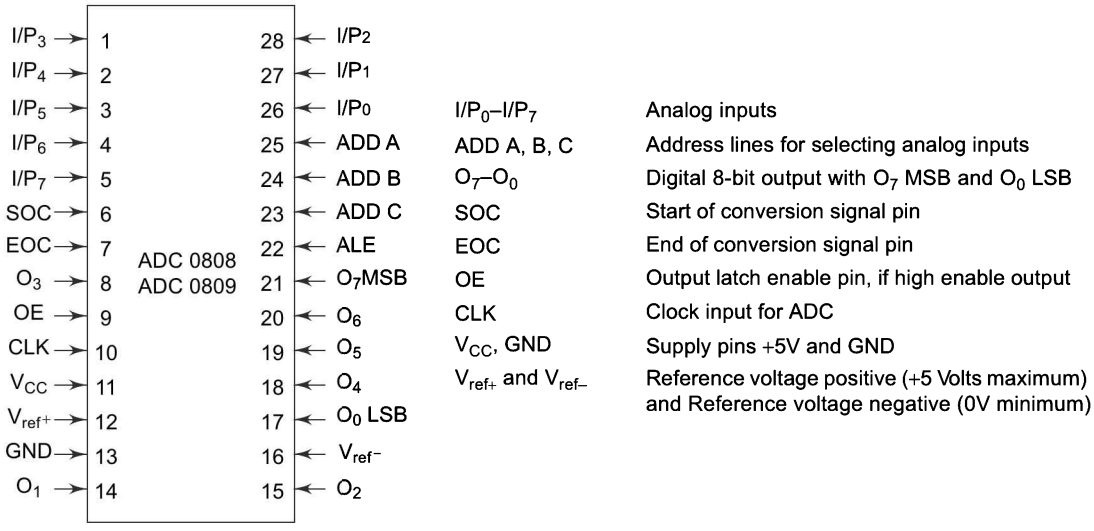


Fig. 5.37(b) *Pin Diagram of ADC 0808/0809*

These are unipolar analog to digital converters, i.e. they are able to convert only positive analog input voltages to their digital equivalents. These chips do not contain any internal sample and hold circuit. If one needs a sample and hold circuit for the conversion of fast signals into equivalent digital quantities, it has to be externally connected at each of the analog inputs. Figures 5.37(a) and (b) show the block diagrams and pin diagrams for ADC 0808/0809. Some electrical specifications of the ADC 0808/0809 are given in Table 5.14.

Table 5.14

Minimum SOC pulse width	100 ns
Minimum ALE pulse width	100 ns
Clock frequency	10 to 1280 kHz
Conversion time	100 ms at 640 kHz
Resolution	8-bit
Error	+/–1 LSB
V_{ref}^+	Not more than +5V
V_{ref}^-	Not less than GND
+ V_{cc} supply	+ 5 V DC
Logical 1 i/p voltage	minimum V_{cc} –1.5 V
Logical 0 i/p voltage	maximum 1.5 V
Logical 1 o/p voltage	minimum V_{cc}–0.4 V
Logical 0 o/p voltage	maximum 0.45 V

Till now we have studied the necessary details of the analog to digital converter chips 0808/0809. Now we consider some interfacing examples of these chips with 8086 so that the working of these ADCs will be absolutely clear along with the required algorithms for interfacing.

Problem 5.16

Interface ADC 0808 with 8086 using 8255 ports. Use Port A of 8255 for transferring digital data output of ADC to the CPU and Port C for control signals. Assume that an analog input is present at I/P_2 of the ADC and a clock input of suitable frequency is available for ADC. Draw the schematic and write required ALP.

The timing diagram of different signals of ADC0808 is shown in Fig. 5.38.

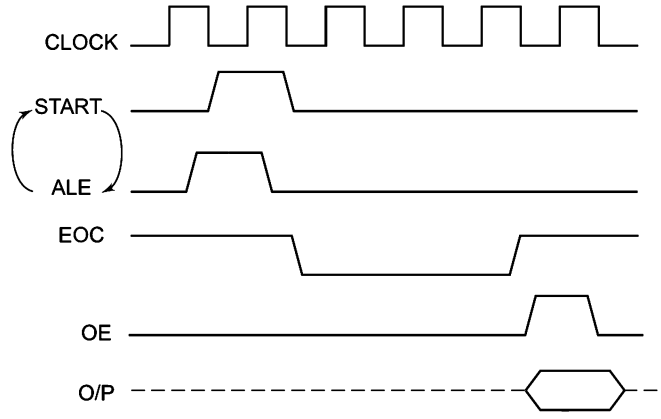


Fig. 5.38 *Timing Diagram of ADC 0808*

Solution Figure 5.39 shows the interfacing connections of ADC0808 with 8086 using 8255. The analog input I/P_2 is used and therefore address pins A,B,C should be 0,1,0 respectively to select I/P_2. The OE and ALE pins are already kept at +5V to select the ADC and enable the outputs. Port C upper acts as the input port to receive the EOC signal while port C lower acts as the output port to send SOC to the ADC. Port A acts as a 8-bit input data port to receive the digital data output from the ADC. The 8255 control word is written as follows:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Control word
1	0	0	1	1	0	0	0	= 98 H

The required ALP is given as follows:

```
        MOV AL,98 H             ; Initialise 8255 as
        OUT CWR,AL              ; discussed above
        MOV AL,02H              ; Select I/P2 as analog
        OUT PORT B,AL           ; input
        MOV AL,00H              ; Give start of conversion
        OUT PORT C,AL           ; pulse to the ADC.
        MOV AL,01 H             ;
        OUT PORT C,AL           ;
        MOV AL,00H              ;
        OUT PORT C,AL           ;
WAIT :  IN AL,PORTC             ; Check for EOC by
        RCL                     ; reading port C upper and
        JNC WAIT                ; rotating through carry.
        IN AL,PORTA             ; If EOC, read digital equivalent in
                                      AL
        HLT                     ; Stop
```

Program 5.11 *ALP for Problem 5.16*

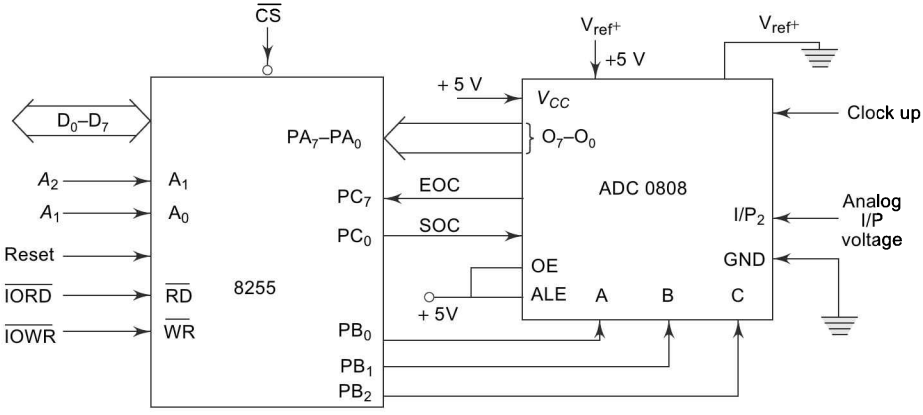


Fig. 5.39 *Interfacing 0808 with 8086*

5.6.2 ADC 7109—A Dual Slope 12-bit ADC

Intersil's ICL 7109 is a dual slope integrating analog to digital converter. This 12-bit ADC is designed to work with 8-bit and 16-bit or higher order microprocessors with great ease. As compared to other 12-bit ADCs, it is a very low cost option, useful for slow practical applications, as the method used for analog to digital conversion is dual slope integration.

The 12-bit data output, polarity and overrange signals can be directly accessed under the software control of two byte enable inputs $\overline{\text{LBEN}}$ and $\overline{\text{HBEN}}$. The RUN/$\overline{\text{HOLD}}$ and STATUS outputs allow monitoring and control of the conversion process. The salient features of the ADC 7109 are as given as follows:

1. 12-bit data output equivalent to analog input along with polarity, over range and under range outputs
2. It can be operated in parallel or serial output mode
3. Differential input and differential reference
4. Low noise (Typical 15 mV p-p)
5. Low input current (Typical 1pA)
6. Can operate up to 30 conversions per second, with an external crystal or RC circuit used to decide the operating clock frequency.

The pin diagram of the ADC is given here followed by brief signal descriptions, in Fig. 5.40 and Table 5.15 respectively.

Table 5.15 *Pin Assignment and Function Description*

Pin	*Symbol*	*Description*		
1.	GND	Digital Ground return for all digital logic		
2.	STATUS	Output High during integrate and deintegrate until data is latched Output Low when analog section is in Auto-Zero configuration		
3.	POL	Polarity-High for Positive input		
4.	OR	Over range-High if Overranged		
5.	B12	Bit 12	(Most Significant Bit)	
6.	B11	Bit 11		
7.	B10	Bit 10		All
8.	B9	Bit 9		three
9.	B8	Bit 8		state
10.	B7	Bit 7	High = true	output
11.	B6	Bit 6		data
12.	B5	Bit 5		bits
13.		B4		
14.		B3		
15.		B2		
16.		B1 (Least Significant Bit)		
17.	TEST	Input High—Normal Operation. Input Low—Forces all bit outputs high Note: This input is used for test purposes only. Tie high if not used.		

(*Contd.*)

Table 5.15 (*Contd.*)

Pin	*Symbol*	*Description*
18.	$\overline{\text{LBEN}}$	Low Byte Enable—With Mode (Pin 21) low, and $\overline{\text{CE / LOAD}}$ (Pin 20) low, taking this pin low activities low order byte outputs B1–B8. — With Mode (Pin 21) high, this pin serves as a low byte flag output used in handshake mode.
19.	$\overline{\text{HBEN}}$	High Byte Enable—With Mode (Pin 21) low, and $\overline{\text{CE / LOAD}}$ (Pin 20) low, taking this pin low activates high order byte outputs B9–B12. POL OR — With Mode (Pin 21) high. This pin serves as a high byte flag output used in handshake mode.
20.	$\overline{\text{CE / LOAD}}$	Chip Enable Load—With Mode (Pin 21) low. $\overline{\text{CE / LOAD}}$ serves as a master output enable. When high, B1–B12, POL OR outputs are disabled. —With Mode (Pin 21) high, this pin serves as a load strobe used in handshake mode.
21.	Mode	Input Low—Direct output mode where $\overline{\text{CE / LOAD}}$ (Pin 20). $\overline{\text{HBEN}}$ (Pin 19) and $\overline{\text{LBEN}}$ (Pin 18) act as inputs directly controlling byte outputs. Input Pulsed High—Causes Immediate entry into handshake mode and outputs data are available accordingly. Input High—Enables $\overline{\text{CE / LOAD}}$ (Pin 20). $\overline{\text{HBEN}}$ (Pin 19), and $\overline{\text{LBEN}}$ (Pin 18) as outputs, handshake mode will be entered and data output is available after conversion completion.
22.	OSC IN	Oscillator Input
23.	OSC OUT	Oscillator Output
24.	OSC SEL	Oscillatory Select—Input high configures OSC IN, OSC OUT, BUF OSC OUT as RC oscillator—clock will be of same phase and duty cycle as BUF OSC OUT. —Input low configures OSC IN, OSC OUT for crystal oscillator—clock frequency will be 1/58 of frequency at BUF OSC OUT.
25.	BUF OSC OUT	Buffered Oscillator Output
26.	RUN/ $\overline{\text{HOLD}}$	Input High—Conversion continuously performed every 8 192 clocks pulses. Input Low—converter will stop in Auto-Zero 7 counts before integrate.
27.	SEND	Input—Used in handshake mode to indicate ability of an external device to accept data. Connect to + 5V if not used.
28.	V^-	Analog Negative Supply—Nominally –5V with respect to GND (Pin 1).

(*Contd.*)

Table 5.15 (*Contd.*)

Pin	*Symbol*	*Description*
29.	REF OUT	Reference Voltage Output—Nominally 2.88 V down from V^- (Pin 40)
30.	BUFFER	Buffer Amplifier Output
31.	AUTO-ZERO	Auto-Zero Node—Inside foil of CAZ
32.	INTEGRATOR	Integrator Output—Outside foil of C_{INT}
33.	COMMON	Analog Common—System is Auto-Zeroed to COMMON
34.	INPUT LO	Differential Input Low Side
35.	INPUT HI	Differential Input High Side
36.	REF IN +	Differential Reference Input Positive
37.	REF CAP +	Reference Capacitor Positive
38.	REF CAP –	Reference Capacitor Negative
39.	REF IN	Differential Reference Input Negative
40.	V^-	Positive Supply Voltage—Nominally + 5V with respect to GND (Pin 1).

Note: All digital levels are positive true.

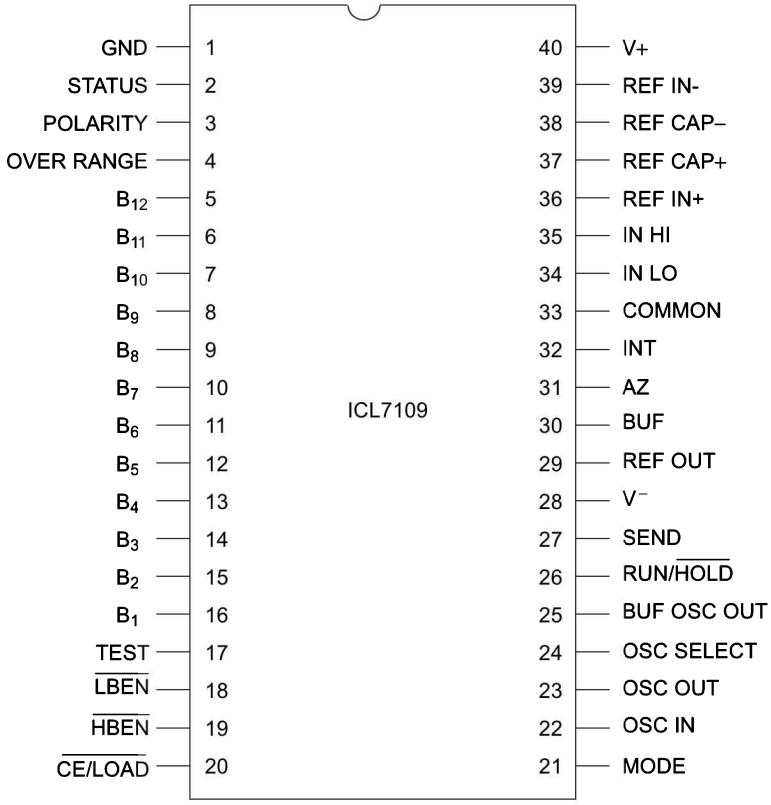


Fig. 5.40 *Pin Configuration of ICL 7109ADC*

The internal circuit of ICL 7109 is slightly complicated. Hence it is divided in two parts namely—analog section and digital section for better understanding. The following text explains the internal operation in terms of the separate sections.

Analog Section The block diagram of the internal analog section of ICL 7109 is shown in Fig. 5.41. If RUN/$\overline{\text{HOLD}}$ is either left open or connected to $+V_{cc}$, the ADC starts conversion at the rate determined by the clock frequency, to be decided either by a crystal or by an RC circuit. The total conversion cycle is divided into three phases namely *autozero phase, signal integrate phase* and *deintegrate phase.*

Autozero phase During autozero phase, input high (IN HI) and input low (IN LO) are disconnected from the external input signal and shorted to analog common. Then the reference capacitor is charged to reference voltage. A feedback loop is closed around the system to charge the autozero capacitor CAZ to compensate for the offset voltages in the buffer amplifier, integrator and comparator.

Signal integrate phase In this phase the autozero capacitor CAZ is opened out. The external input signal is connected with the internal circuit. The differential input voltage between IN LO and IN HI pins is then integrated by the internal integrator for a fixed period of 2048 clock cycles.

De-integrate phase This is the final phase of the analog to digital conversion. The input low is internally connected to analog common, and input high is connected across the previously charged reference capacitor. The capacitor then discharges through the internal circuit of the chip. Hence integrator output returns to zero crossing with a fixed slope. This time taken by the integrator output to return to zero is proportional to the input signal.

Digital Section The digital section includes the clock oscillator, 12-bit binary counter, output latches, TTL compatible output drivers, polarity, overrange and their control logics and UART handshake logic as organised in Fig. 5.42.

The digital section uses a positive logic system wherein logical 'low' corresponds to a low voltage and logical 'high' corresponds to a high voltage. The actual ranges for logical 'low' and 'high' can be obtained from the data sheets.

Intersil's 'Component Data Catalogue may be referred for the detailed information about the chip and its functioning. This text is only aimed at giving some brief introduction of the chip before we proceed for its interfacing with 8086.

Typical component value selection For the proper working of ICL 7109, it is necessary that the component values of the practical circuit are properly chosen. All the component values are recommended in the data manual but, the three components, viz. Rint, Cint and CAz should be selected suitably as all of them directly determine the required input voltage range and the operating clock frequency. The full scale input voltage range is double the voltage between REF IN- and REF IN+. The clock frequency should be integral multiple of the power supply frequency (50Hz) to achieve the optimum power supply frequency rejection. The formulae for component values selection are given as follows:

$$\text{Rint} = \frac{\text{Full scale i / p voltage}}{20\mu\text{A}}$$

$$\text{Cint} = \frac{2048 * (\text{Clockperiod}) * 20\mu\text{A}}{\text{Integrator o / p voltage swing}}$$

The ADC 7109 can either be driven with a clock frequency, derived from a crystal or from an RC circuit. If the clock frequency is derived from the crystal then the actual operating frequency is given by *f.*

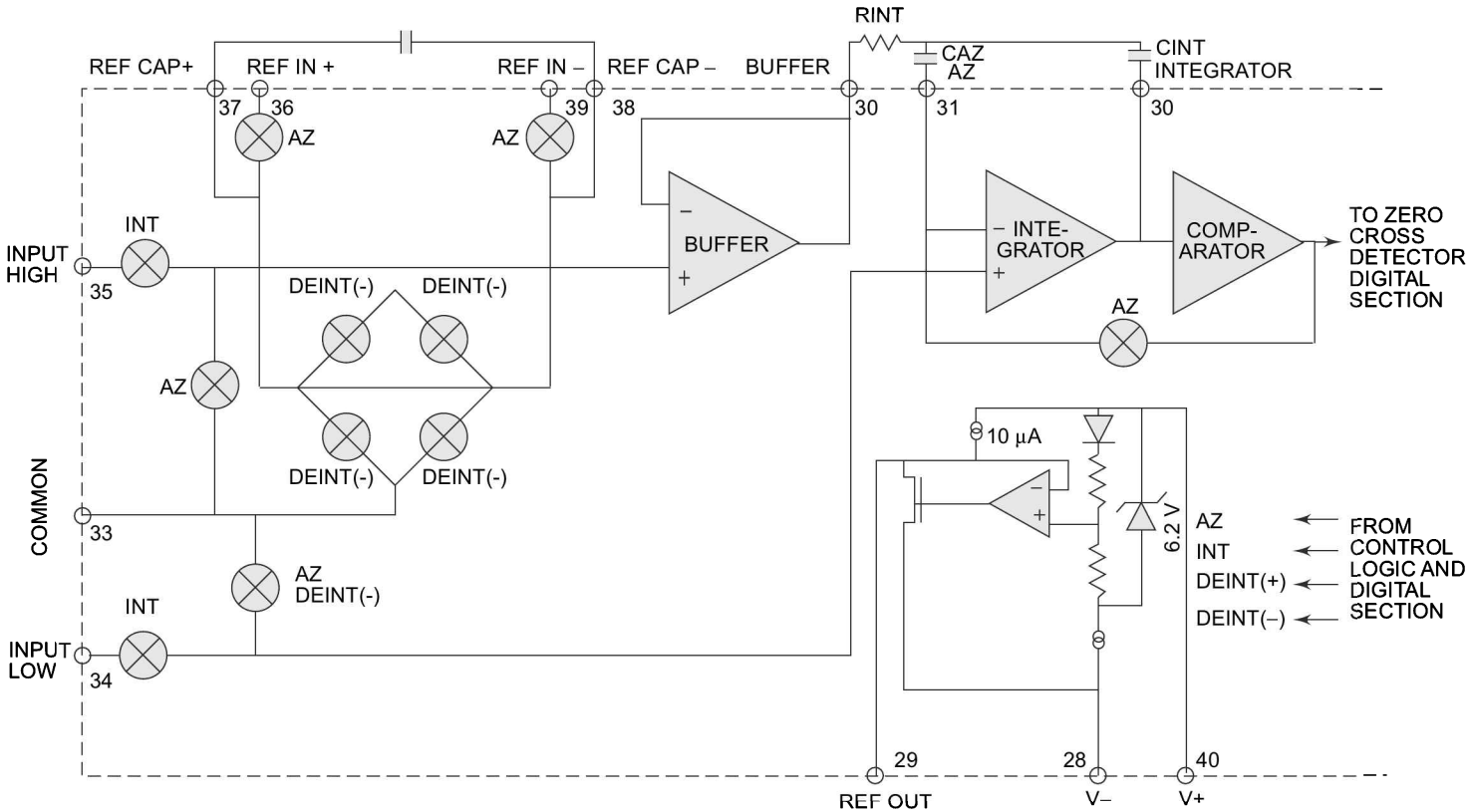


Fig. 5.41 *Analog Section of 7109*

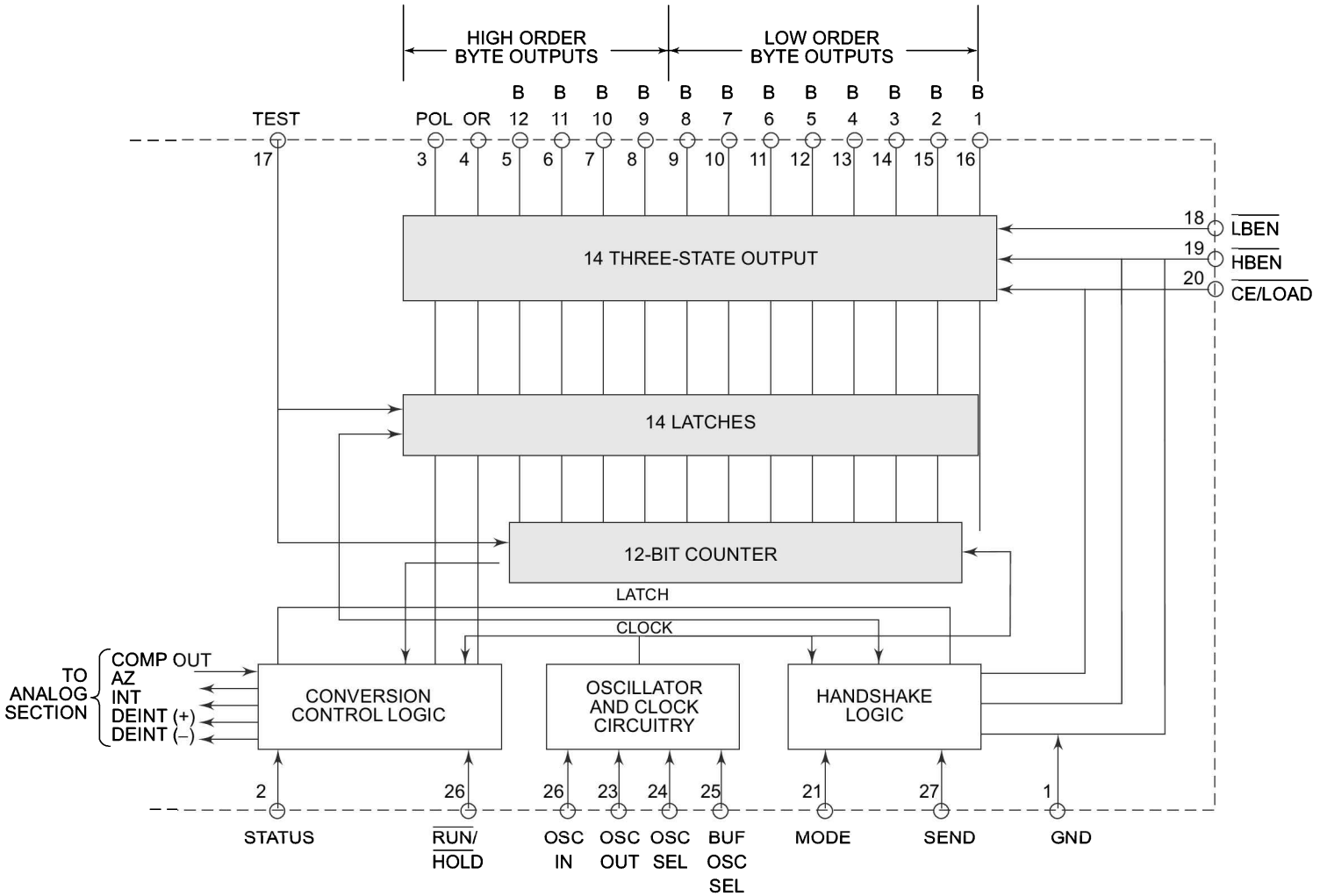


Fig. 5.42 *Digital Section of ICL 7109*

$$f = \frac{\text{Crystal freq}}{58}$$

Otherwise, if the ADC is driven by a clock frequency derived from an RC circuit it is given by the formula.

$$f = \frac{0.45}{RC}$$

5.6.3 Interfacing with 8086

Figure 5.43 shows the interfacing of ICL 7109 with 8086 using 8255. The assembly language program reads the digital equivalent output from ADC 7109 and stores it in the register CX. Note that the ADC gives only 12-bit output, hence the most significant nibble of CX must be masked.

The parallel I/O port chip 8255 is used to interface ICL 7109 with 8086. Being a 12-bit ADC, it will require 12 I/O lines for data outputs. Port A and Port C lower are used as input ports to read digital data output from the ADC. The pins $\overline{\text{LBEN}}$ and $\overline{\text{HBEN}}$ are permanently grounded to enable all the data lines from

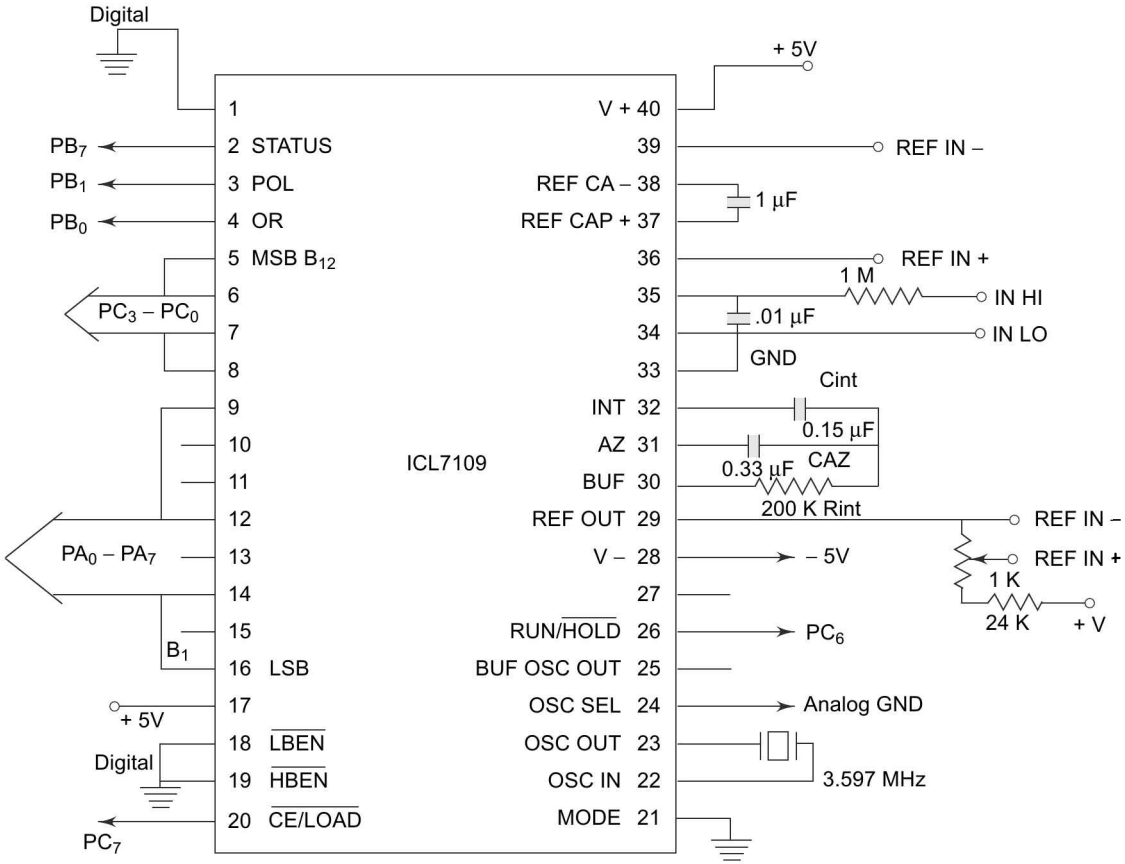


Fig. 5.43 *Interfacing 7109 with 8086 through 8255*

B_1 to B_{12} at a time. The lines $\overline{\text{CE/LOAD}}$ and RUN/ $\overline{\text{HOLD}}$ are controlled using port C upper of 8255, that is used as an output port and finally port B is used to read the STATUS, POL and OR signals and hence is used as an input port. The polarity (POL) and overrange data (OR) will be available in DL register with D_0 corresponding to OR and D_1 corresponding to POL. The digital data is read from the ports if STATUS goes low, i.e. conversion if the over. Figure 5.44 shows the algorithm for the analog to digital conversion with ADC 7109. After the execution of the program the digital equivalent of the analog input is in register CX, while the overrange and polarity information is in DL. From the above description 8255 control word comes out to be 93H.

The above circuit components are designed for 4096 mV full scale analog input range. As specified in the manual the Rint for this range is 200 K. The reference input voltage between REF IN- and REF IN+ will be half of full scale analog input voltage.

```
            ; This program issues a RUN signal to ICL7109,checks for
            ; STATUS(EOC) and reads digital output with POL and OR
                  outputs.
ASSUME      CS : CODE
CODE        SEGMENT
START:      MOV AL, 93H              ; Initialization of
            OUT CWR, AL              ; 8255
            MOV AL, 40 H             ; RUN (PC6) to go high and
            OUT PORT C, AL           ; CE(PC7) to go low, for start of
                                           conversion.
            WAIT : IN AL, PORTB      ; Read STATUS signal.
            RCL AL, 01               ; Check STATUS using carry flag.
            JC WAIT                  ; Wait till carry (STATUS) goes
            IN AL, PORTA             ; low i.e. conversion is over.
            MOV CL, AL
            IN AL, PORTC             ; Read digital data output and store
                                           the
            AND AL, OFH              ; lower byte in CL and higher byte
                                           in CH. Mask
            MOV CH, AL               ; higher bits of CH as of only 12
                                           bits are of interest.
            IN AL,PORTB              ; Store OR and POL in D0 and D1 in
            MOV DL,AL                ; DL register.
            MOV AH,4CH               ; Return to DOS.
            INT 21H
 CODE       ENDS
            END START
```

Program 5.12 *ALP for Interfacing ICL 7109 with 8086*

Readers may find a number of other 8-bit and 12-bit ADCs, their details and interfacing techniques from the respective data manuals or other textbooks. The discussion here is mainly aimed at explaining the general interfacing techniques of ADCs though the specific ADCs are discussed in detail.

5.7 INTERFACING DIGITAL TO ANALOG CONVERTERS

The digital to analog converters convert binary numbers into their analog equivalent voltages. The DAC find applications in areas like digitally controlled gains, motor speed controls, programmable gain amplifiers, etc. This text explains the generally available DAC integrated circuits and their interfacing techniques with the microprocessor.

5.7.1 AD 7523 8-bit Multiplying DAC

Intersil's AD 7523 is a 16 pin DIP, multiplying digital to analog converter, containing R-2R ladder (R = 10 K) for digital to analog conversion along with single pole double throw NMOS switches to connect the digital inputs to the ladder.

Pin diagram of AD7523 is shown in Fig. 5.45.

The supply range extends from +5V to +15V, while V_{ref} may be any where between –10V to +10V. The maximum analog output voltage will be +10V, when all the digital inputs are at logic high state. Usually a Zener is connected between OUT1 and OUT2 to save the DAC from negative transients. An operational amplifier is used as a current-to-voltage converter at the output of AD 7523 to convert the current output of AD7523 to a proportional output voltage. It also offers additional drive capability to the DAC output. An external feedback resistor acts to control the gain. One may not connect any external feedback resistor, if no gain control is required. The following example explains the interfacing of AD 7523 with 8086.

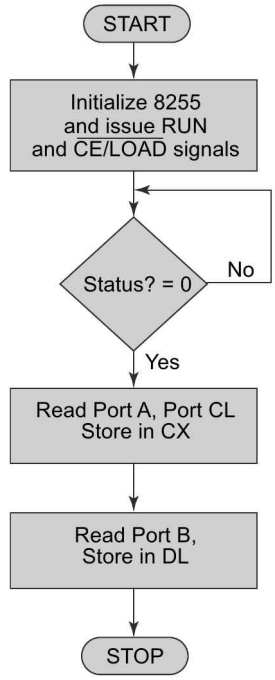


Fig. 5.44 *Algorithm for reading O/P of 7109*

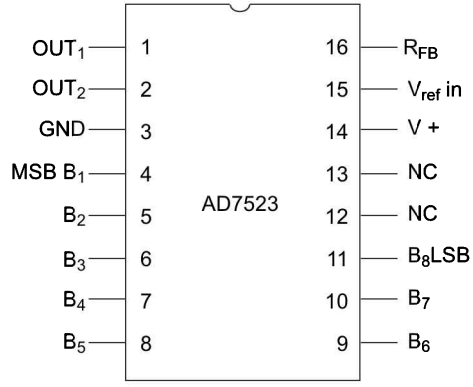


Fig. 5.45 *Pin Diagram of AD7523*

Problem 5.17

Interface DAC AD7523 with an 8086 CPU running at 8 MHz and write an assembly language program to generate a sawtooth waveform of period 1 ms with V_{max} 5V.

Solution Figure 5.46 shows the interfacing circuit of AD 7523 with 8086 using 8255. Program 5.13 gives an ALP to generate a sawtooth waveform using this circuit.

```
ASSUME        CS : CODE
```

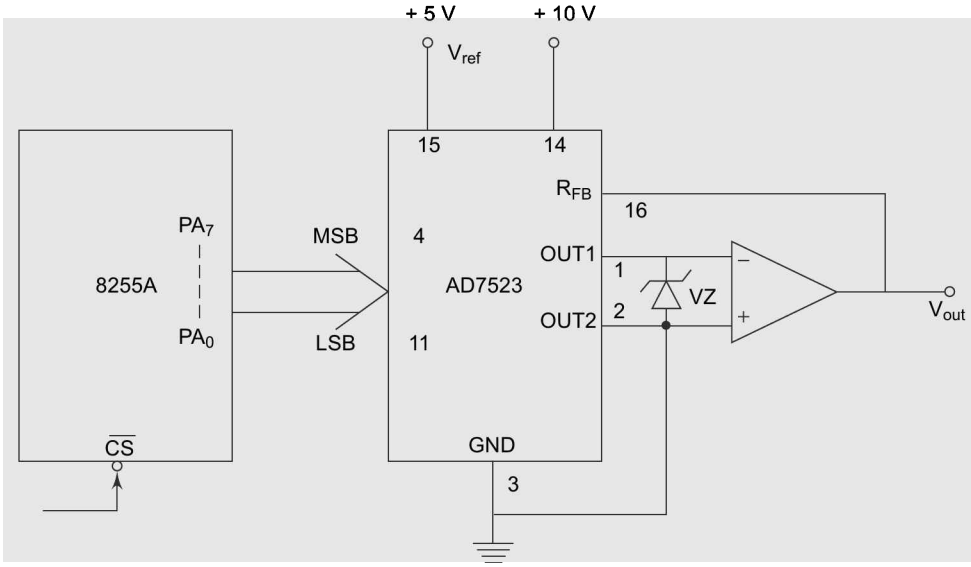


Fig. 5.46 *Interfacing of AD7523*

```
CODE        SEGMENT
START:      MOV AL,80 H          ; Initialise port A as output
            OUT CWR,AL           ; port
AGAIN:      MOV AL,00H           ; Start the ramp from 0V
BACK :      OUT PORTA,AL         ; Input 00H to DAC
            INC AL               ; Increment AL to increase ramp output
            CMP AL,0F2H          ; Is upper limit reached?
            JB BACK              ; If not, then increment the ramp
            JMP AGAIN            ; Else start again from 00H
CODE        ENDS
            END START
```

Program 5.13 *ALP for Generating Sawtooth Waveform Using AD 7523*

In the above program, port A is initialized as the output port for sending the digital data as input to DAC. The ramp starts from the 0V (analog), hence AL starts with 00H. To increment the ramp, the content of AL is incremented during each execution of the loop till it reaches F2H. After that the saw tooth wave again starts from 00H, i.e. 0V(analog), and the procedure is repeated. Note that the ramp period given by this program is precisely 1.000625 ms. Here the count F2H has been calculated by dividing the required delay of 1ms by the time required for the execution of the loop once. The ramp slope can be controlled (reduced) by calling a controllable delay after the OUT instruction. It may be noted here that meeting the frequency, i.e. time and amplitude requirement exactly is slightly difficult in such applications.

5.7.2 DAC0800 8-bit Digital to Analog Converter

The DAC 0800 is a monolithic 8-bit DAC manufactured by National Semiconductor. It has settling time around 100 ms and can operate on a range of power supply voltages, i.e. from 4.5 V to + 18 V. usually the supply V+ is 5 V or +12 V. The V-pin can be kept at a minimum of – 12 V. The pin diagram of DAC 0800 is shown in Fig. 5.47. The pin definitions are self explanatory. Figure 5.48 shows interfacing of 0800 DAC with 8086.

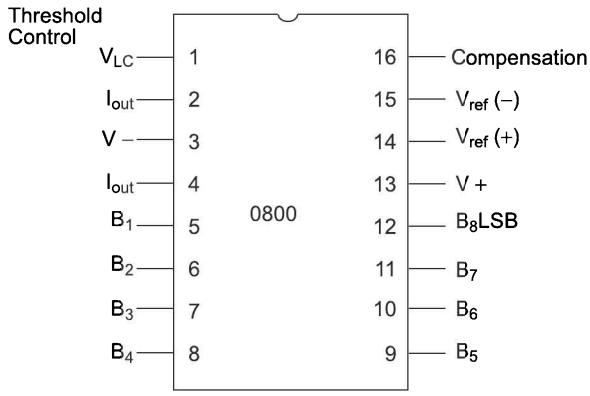


Fig. 5.47 *Pin Diagram of DAC 0800*

Problem 5.18

Write an assembly language program to generate a triangular wave of frequency 500 Hz using the interfacing circuit given in Fig. 5.48. The 8086 system operates at 8 MHz. The amplitude of the triangular wave should be +5 V.

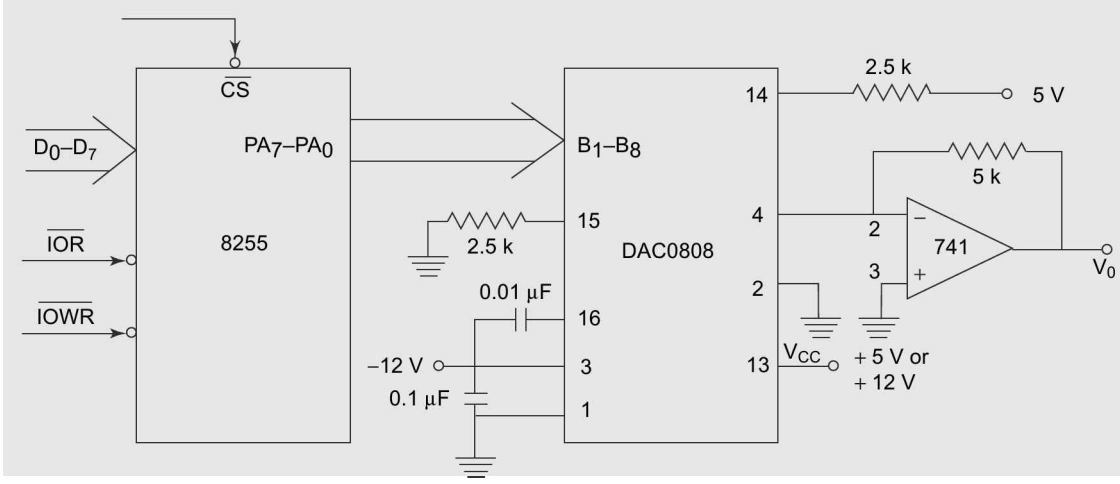


Fig. 5.48 *Interfacing DAC0800 with 8086*

Solution The V_{ref+} should be tied to +5 V to generate a wave of +5V amplitude. The required frequency of the output is 500 Hz, i.e. the period is 2 ms. Assuming the wave to be generated is symmetric, the waveform will rise for 1 ms and fall for 1 ms. This will be repeated continuously. In the previous program, we have already written an instruction sequence for period 1 ms. Using the same instruction sequence one can derive this triangular waveform. The ALP is given as follows:

```
ASSUME     CS : CODE
CODE       SEGMENT
START :    MOV AL,80 H          ; Initialise 8255 ports
           OUT CWR,AL           ; suitably.
           MOV AL,00H           ; Start rising ramp from
```

```
BACK :    OUT PORT A,AL      ; OV by sending OOH to DAC.
          INC AL             ; Increment ramp till 5V
          CMP AL,FFH         ; i.e. FFH.
          JB BACK            ; If it is FFH then,
BACK1 :   OUT PORT A,AL      ; Output it and start the falling
          DEC AL             ; ramp by decrementing the
          CMP AL,OO          ; counter till it reaches
          JA BACK1           ; zero. Then start again
          JMP BACK           ; for the next cycle.
CODE      ENDS
          END START
```

Program 5.14 *ALP for Generating a Triangular Wave Using DAC 0800*

The technique of interfacing 12-bit DACs with 8086 is also similar. If 8-bit ports are used for interfacing a 12-bit DAC, two successive 8-bit OUT instructions are required to apply input to the DAC. If a 16-bit port is used for interfacing a 12-bit DAC, a single OUT instruction is sufficient to apply the 12-bit input to the DAC. 12-bit DACs generate more precise analog voltages ($2^{12} = 4096$ steps in full output range) as compared to 8-bit DACs ($2^8 = 256$ steps in full output range).

5.8 STEPPER MOTOR INTERFACING

A stepper motor is a device used to obtain an accurate position control of rotating shafts. It employs rotation of its shaft in terms of steps, rather than continuous rotation as in case of AC or DC motors. To rotate the shaft of the stepper motor, a sequence of pulses is needed to be applied to the windings of the stepper motor, in a proper sequence. The number of pulses required for one complete rotation of the shaft of the stepper motor are equal to its number of internal teeth on its rotor. The stator teeth and the rotor teeth lock with each other to fix a position of the shaft. With a pulse applied to the winding input, the rotor rotates by one teeth position or an angle x. The angle x may be calculated as:

$$x = 360°/\text{no. of rotor teeth}$$

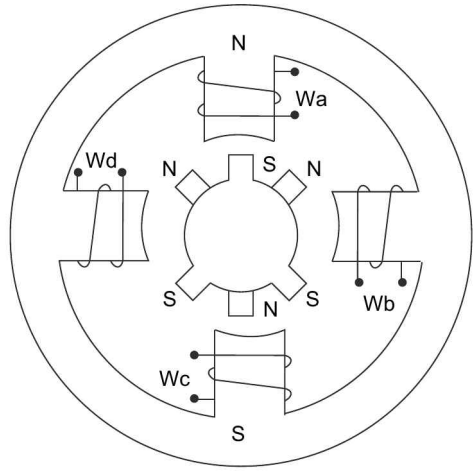


Fig. 5.49(a) *Internal Schematic of a Four Winding Stepper Motor*

After the rotation of the shaft through angle x, the rotor locks itself with the next tooth in the sequence on the internal surface of stator. The internal schematic of a typical stepper motor with four windings is shown in Fig. 5.49(a). The stepper motors have been designed to work with digital circuits. Binary level pulses of 0–5V are required at its winding inputs to obtain the rotation of shafts. The sequence of the pulses can be decided, depending upon the required motion of the shaft. Figure 5.49(b) shows a typical winding arrangement of the stepper motor. Figure 5.49(c) shows conceptual positioning of the rotor teeth on the surface of rotor, for a six teeth rotor.

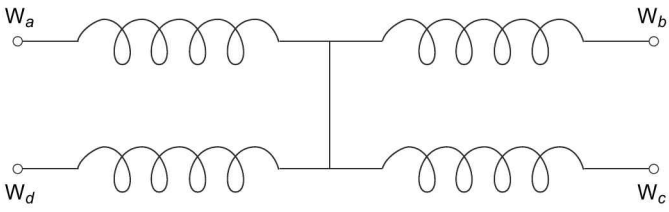


Fig. 5.49(b) *Winding Arrangement of a Stepper Motor*

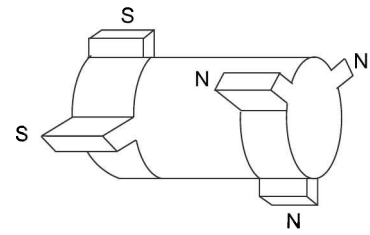


Fig. 5.49(c) *A Stepper Motor Rotor*

The circuit for interfacing a winding W_n with an I/O port is given in Fig. 5.50. Each of the windings of a stepper motor need this circuit for its interfacing with the output port. A typical stepper motor may have parameters like torque 3 kg-cm, operating voltage 12 V, current rating 0.2 A and a step angle 1.8°, i.e. 200 steps/revolution (number of rotor teeth).

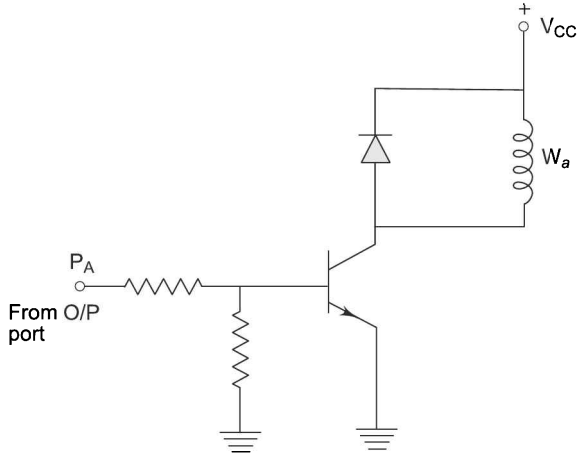


Fig. 5.50 *Interfacing Stepper Motor Winding W_a*

A simple scheme for rotating the shaft of a stepper motor is called a wave scheme. In this scheme, the windings W_a, W_b, W_c and W_d are applied with the required voltage pulses, in a cyclic fashion. By reversing the sequence of excitation, the direction of rotation of the stepper motor shaft may be reversed. Table 5.16(a) shows the excitation sequences for clockwise and anticlockwise rotations. Another popular scheme for rotation of a stepper motor shaft applies pulses to two successive windings at a time but these are shifted only by one position at a time. This scheme for rotation of stepper motor shaft is shown in Table 5.16(b).

Table 5.16(a) *Excitation Sequences of a Stepper Motor Using Wave Switching Scheme*

Motion	*Step*	*A*	*B*	*C*	*D*
Clockwise	1	1	0	0	0
	2	0	1	0	0
	3	0	0	1	0
	4	0	0	0	1
	5	1	0	0	0

(*Contd.*)

Table 5.16(a) (*Contd.*)

Motion	*Step*	*A*	*B*	*C*	*D*
Anticlockwise	1	1	0	0	0
	2	0	0	0	1
	3	0	0	1	0
	4	0	1	0	0
	5	1	0	0	0

The following problem elaborates stepper motor interfacing circuit and the required programming. A number of schemes are available for rotating shaft of a stepper motor. The above two schemes are generally used in practical applications.

Problem 5.19

Design a stepper motor controller and write an ALP to rotate shaft of a 4-phase stepper motor:

(i) in clockwise 5 rotations

(ii) in anticlockwise 5 rotations

Table 5.15(b) *An Alternative Scheme for Rotating Stepper Motor Shaft*

Motion	*Step*	*A*	*B*	*C*	*D*
Clockwise	1	0	0	1	1
	2	0	1	1	0
	3	1	1	0	0
	4	1	0	0	1
	5	0	0	1	1
Anticlockwise	1	0	0	1	1
	2	1	0	0	1
	3	1	1	0	0
	4	0	1	1	0
	5	0	0	0	0

The 8255 port A address is 0740H. The stepper motor has 200 rotor teeth. The port A bit PA_0 drives winding W_a, PA1 drives W_b and so on. The stepper motor has an inertial delay of 10 m sec. Assume that the routine for this delay is already available.

Solution The stepper motor connections for all the four windings are shown in Fig. 5.51. The ALP for rotating the shaft of the stepper motor is shown in Program 5.15.

```
ASSUME CS : CODE
CODE SEGMENT
START:    MOV AL, 80H
```

```
            OUT CWR, AL
            MOV AL, 88H          ; Bit pattern 10001000 to start
            MOV CX, 1000         ; the sequence of excitation
AGAIN1:     OUT PORT A,AL        ; from WA. Excite WA, WB,
            CALL DELAY           ; WC and WD in sequence with delay. For 5
                                       clockwise
            ROL AL, 01           ; rotations the count is 200*5 = 1000.
            DEC CX               ; Excite till count = 0.
            JNZ AGAIN1
            MOV AH, 88H          ; Bit pattern to excite WA.
            MOV CX, 1000         ; Count for 5 rotations
```

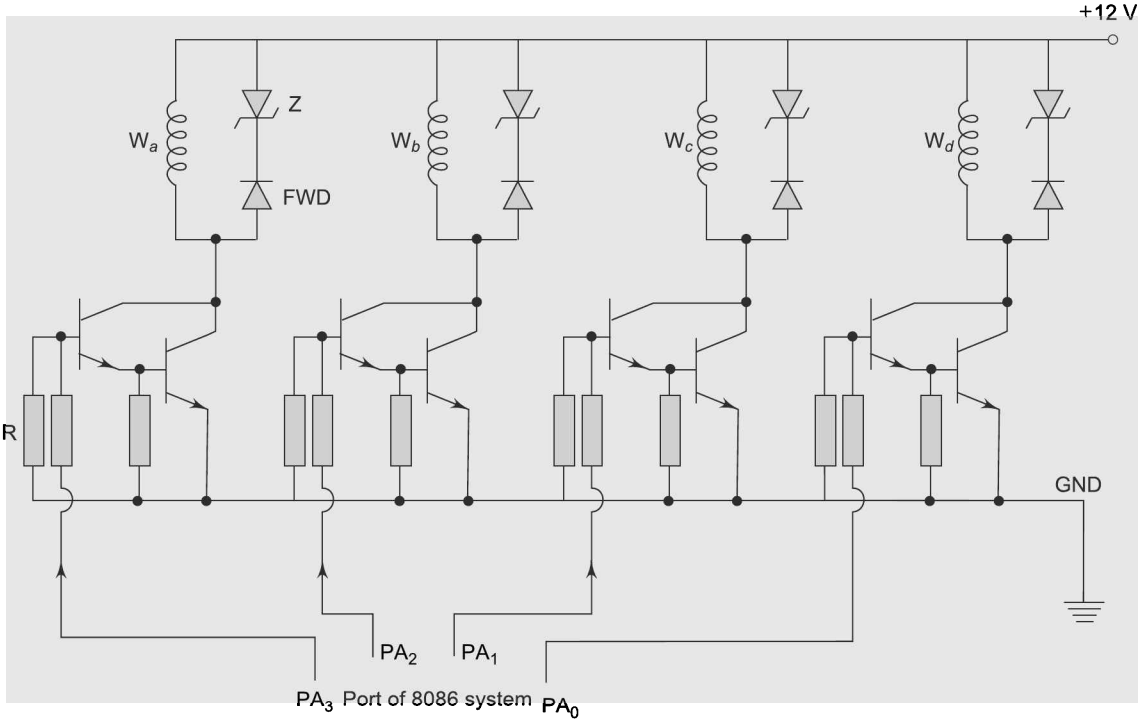


Fig. 5.51 *Stepper Motor Windings Connections*

```
AGAIN2:     OUT PORTA,AL         ; Excite WA, WB, WC, and WD.
            CALL DELAY           ; Wait.
            ROR AL, 01           ; Rotate bit pattern right to obtain
                                       anticlockwise
            DEC CX               ; motion of shaft. Decrement count.
            JNZ AGAIN2           ; If 5 rotations are completed
            MOV AH, 4CH          ; return to DOS else
            INT 21H              ; Continue
CODE        ENDS
            END START
```

Program 5.15 *ALP for Problem 5.19*

The count for rotating the shaft of the stepper motor through a specified angle may be calculated from the number of rotor teeth. The number of rotor teeth is equal to the count for one rotation, i.e. 360°. Hence for any specified angle $\theta°$ the count is calculated as:

$$C = \frac{\text{Number of rotor teeth}}{350} \times \theta°$$

5.9 CONTROL OF HIGH POWER DEVICES USING 8255

High power devices like motors, heating furnaces, temperature baths and other process equipments may be controlled using I/O ports of a microcomputer system. These are controlled using power electronics devices like SCRs, triacs, etc. These devices control the flow of energy to the processes to be controlled. This is obtained by controlling the firing angle of SCRs or Triacs. In the early days, the firing angles of thyristors were controlled using pulse generating circuits like relaxation oscillators but with the advancement in the field of microprocessors, it was observed that SCRs or triacs can be more accurately fired using a microprocessor. The main problem in this method lies in isolation between low power and high power circuits. A microprocessor circuit is a low power circuit, while an electrical circuit of a furnace is a high power circuit. Even any switching component of a high power circuit may be sufficient to damage the microprocessor system. Hence to protect this low power circuit, isolation is necessary. Isolation transformers are generally used for this purpose. Optoisolators also provide excellent isolation between high power and low power sides. Moreover, they are compact in size and cost effective as compared to the pulse transformers. The I/O signals generated by the microprocessor for these high power devices are applied through these isolating circuits as shown in Figs 5.52 (a) and (b).

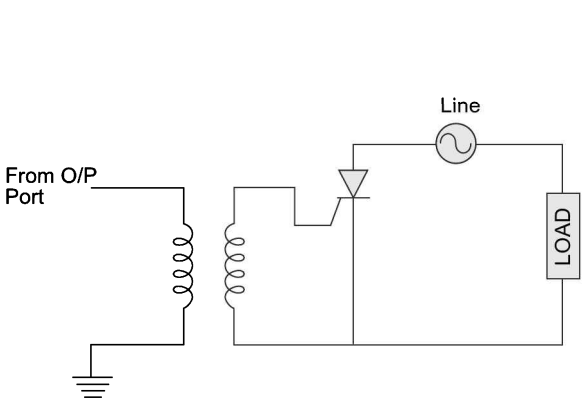


Fig. 5.52(a) *Isolation Using Transformers*

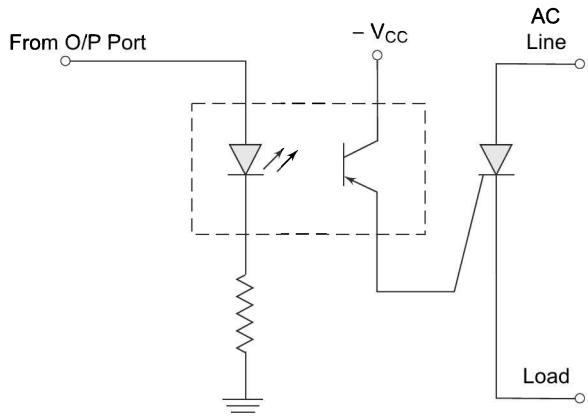
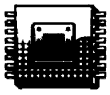


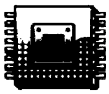
Fig. 5.52(b) *Isolation Using Opto-Isolators*



SUMMARY

In this chapter, we have presented some of the peripheral devices and their interfacing circuits. To start with, the interfacing of static and dynamic memories have been discussed with examples. Then general concepts of I/O devices and their interfacing with 8086 are briefly explained. The parallel programmable peripheral interface

8255 has been presented in significant details along with the interfacing examples and programs for each of its operating modes. Further, a few important devices like ADCs, DACs and stepper motor have been discussed with an emphasis on their interfacing with 8086. The discussion concludes with a brief note on control of power (control) electronics devices. A number of advanced I/O devices are available which can be interfaced with the CPU using I/O ports, and the interfacing techniques of all these devices to be interfaced using I/O ports are similar in principle. Besides this, a large family of special purpose programmable peripheral devices is also available. This contributes considerably to the development of the recently available advanced computer, communication and automatic control systems. Some of these peripherals are explained in the next chapter.



EXERCISES

5.1 Interface four 8K chips of static RAM and four 4K chips of EPROM with 8086. Interface two of the RAM chips in the zeroth segment so as to accommodate IVT in them. The remaining two RAM chips are to be interfaced at the end of the fifth segment. Two EPROM chips should be interfaced so that the system is restartable as usual, and the remaining two EPROM chips should be interfaced at the starting of A000th segment. Use absolute decoding scheme.

5.2 Interface eight 8K chips of RAM and four 8K chips of EPROM with 8086. Interface the RAM bank at a segment address 0B00H and the EPROM bank at a physical address F8000H. Do not allow any fold back space.

5.3 Interface eight 8K chips of RAM and four 8K × 4 chips of EPROM with 8086 at the further specified address map. You may use linear decoding scheme for minimising the required hardware.

Chips	Segment Address	Starting Offset Address
RAM1 and RAM2	0000H	0000H
RAM3 and RAM4	0500H	5000H
RAM5 and RAM6	7000H	2000H
RAM7 and RAM8	E000H	A000H
EPROM1 and EPROM2	F000H	0000H
EPROM3 and EPROM4	D000H	7000H

Will this system be practically useful? Explain. If not then what minimum change do you suggest in this address map?

5.4 Interface two 8K RAM chips and two 4K EPROM chips with 8088 so as to form a completely working system configuration.

5.5 Describe the procedure of interfacing static memories with a CPU? Bring out the differences between interfacing the memories with 8086 and 8088.

5.6 Bring out the differences between static and dynamic RAM.

5.7 Design a 2-digit seconds counter using 74373 output ports.

5.8 Design a 3-digit pulse counter using 74373 output ports to count TTL compatible pulses using an input line of a 74245 input port.

5.9 Generate a square wave of period 1sec using 74373 output ports.

5.10 Design a multiplexed display scheme to display seconds, minutes and hours counter using 8255 ports. Assume that a standard delay of 1 second is available.

5.11 Design a one unit 14-segment alphanumeric display and write a program to display an alphanumeric character of which the code is in AX.

5.12 Interface an 8255 with 8086 so as to have port A address 00, port B address 02, port C address 01 and CWR address 03.

5.13 Explain the different modes of operation of 8255.

5.14 Explain the control word format of 8255 in I/O and BSR mode.

5.15 Write a program to print message—'This is a printer test routine.' to a printer using 8255 port initialised in mode1.

5.16 Interface an 8´8 keyboard using two 8255 ports and write a program to read the code of a pressed key.

5.17 Interface a typical 12-bit DAC with 8255 and write a program to generate a triangular waveform of period 10 ms. The CPU runs at 5 MHz clock frequency.

5.18 Using a typical 12-bit DAC generate a step waveform of duration 1sec, maximum voltage 3 volts and determine duration of each step suitably.

5.19 Draw and discuss analog and digital section of 7109 in brief.

5.20 Design a 7109 circuit to convert the analog voltage to digital equivalent at a rate of 30 samples per second.

5.21 Draw a typical stepper motor interface with 8255.

5.22 Write ALPs to rotate a 200 teeth, 4 phase stepper motor as specified below.

(i) Five rotations clockwise and then five rotations anticlockwise.

(ii) Rotate through angle 135° in 2 sec.

(iii) Rotate the shaft at a speed of 10 rotations per minute.

5.23 Write ALPs to trigger a triac with a +5 V pulse of 200 msec as specified below.

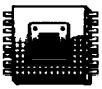
(i) At an angle 45^{o} in each positive and negative half cycle.

(ii) At an angle 30^{o} in each positive half cycle.

(iii) At an angle 20^{o} in each negative half.

Assume that after each zero crossing of a 50 Hz line signal an external zero crossing circuit generates an interrupt to the CPU which runs at a frequency of 5 MHz.

6 Special Purpose Programmable Peripheral Devices and Their Interfacing



INTRODUCTION

In the previous chapter, we discussed a few general purpose peripheral devices along with their interfacing techniques and example programs. More or less all the peripheral devices provide interface between the CPU and slow electromechanical processes or devices. Previously, all these interfaces were taken care of by the CPU using interrupt mechanism or polling techniques. However, due to the low speed of the processes a considerable amount of CPU time gets consumed in the interface and communication activities, resulting in reduced overall efficiency and low processing speed. To minimize the slow speed I/O communication overhead, a set of dedicated programmable peripheral devices have been introduced. Once initiated by the CPU, these programmable peripheral devices take care of all the interface activities for which they have been designed. Thus the CPU becomes free from the interface activities and can execute its main task more efficiently. In this chapter, we present several integrated programmable peripherals and their interfacing techniques. More advanced peripherals based on DMA controlled data transfer will be discussed in the next chapter.

6.1 PROGRAMMABLE INTERVAL TIMER 8254

In Chapter 4, we have shown that it is not possible to generate an arbitrary time delay precisely using delay routines. Intel's programmable counter/timer device (8254) facilitates the generation of accurate time delays. When 8254 is used as a timing and delay generation peripheral, the microprocessor becomes free from the tasks related to the counting process and can execute the programs in memory, while the timer device may perform the counting tasks. This minimizes the software overhead on the microprocessor.

6.1.1 Architecture and Signal Descriptions

The programmable timer device 8254 contains three independent 16-bit counters, each with a maximum count rate of 10 MHz. It is thus possible to generate three totally independent delays or maintain three independent counters simultaneously. All the three counters may be independently controlled by programming the three internal command word registers.

The 8-bit, bidirectional data buffer interfaces internal circuit of 8254 to microprocessor systems bus. Data is transmitted or received by the buffer upon the execution of IN or OUT instruction. The read/write logic

controls the direction of the data buffer depending upon whether it is a read or a write operation. It may be noted that IN instruction reads data while OUT instruction writes data to a peripheral. The internal block diagram and pin diagram of 8254 are shown in Figs 6.1(a) and 6.1(b), respectively.

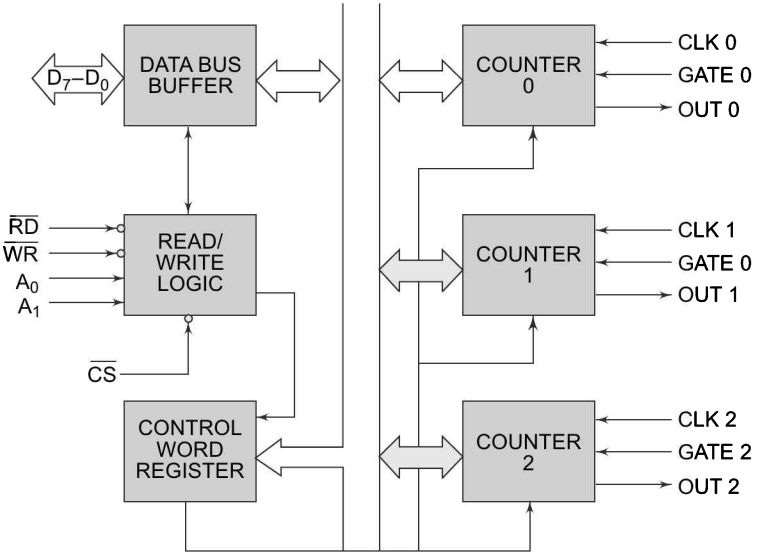


Fig. 6.1(a) *Internal Block Diagram of 8254*

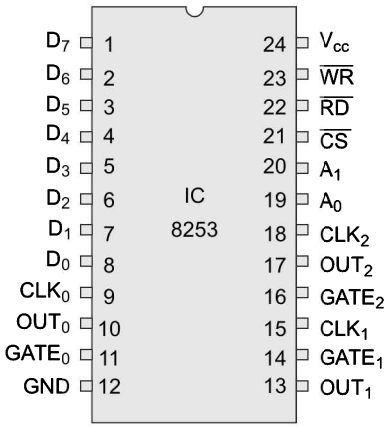


Fig. 6.1(b) *Pin Configuration of 8254*

The three counters available in 8254 are independent of each other in operation, but they are identical to each other in organization. These are all 16-bit presettable, down counters, able to operate either in BCD or in hexadecimal mode. The mode control word register contains the information that can be used for writing or reading the count value into or from the respective count register using the OUT and IN instructions. The speciality of the 8254 counters is that they can be easily read on line without disturbing the clock input to the counter. This facility is called as "on the fly" reading of counters, and is invoked using a mode control word.

A_0, A_1 pins are the address input pins and are required internally for addressing the mode control word registers and the three counter registers. A low on '$\overline{CS}$' line enables the 8254. No operation will be performed by 8254 till it is enabled. Table 6.1 shows the selected operations for various control inputs.

Table 6.1 *Selected Operations for Various Control Inputs of 8254*

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	*Selected Operation*
0	1	0	0	0	Write Counter 0
0	1	0	0	1	Write Counter 1
0	1	0	1	0	Write Counter 2
0	1	0	1	1	Write Control Word
0	0	1	0	0	Read Counter 0
0	0	1	0	1	Read Counter 1
0	0	1	1	0	Read Counter 2
0	0	1	1	1	No Operation (tristated)
0	1	1	×	×	No Operation (tristated)
1	×	×	×	×	Disabled (tristated)

A control word register accepts the 8-bit control word written by the microprocessor and stores it for controlling the complete operation of the specific counter. It may be noted that, the control word register can only be written and cannot be read as it is obvious from Table 6.1. The CLK, GATE and OUT pins are available for each of the three timer channels. Their functions will be clear when we study the different operating modes of 8254.

6.1.2 Control Word Register

The 8254 can operate in any one of the six different modes. A control word must be written in the respective control word register by the microprocessor to initialize each of the counters of 8254 to decide its operating mode. Each of the counter works independently depending upon the control word decided by the programmer as per the needs. In other words, all the counters can operate in any one of the modes or they may be even in different modes of operation, at a time. The control word format is presented, along with the definition of each bit, in Fig. 6.2.

While writing a count in the counter, it should be noted that, the count is written in the counter only after the data is put on the data bus and a falling edge appears at the clock pin of the peripheral thereafter. Any reading operation of the counter, before the falling edge appears may result in garbage data.

With this much information, on the general functioning of 8254, one may proceed further for the details of the operating modes of 8254. However, the concepts shall be clearer after one goes through the interfacing examples and the related assembly language programs.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

Control Word Format

SC_1	SC_0	OPERATION
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Illegal

SC-Select Counter Bit Definitions

RL_1	RL_0	OPERATION
0	0	Latch Counter for 'ON THE FLY' reading
0	1	Read/Load Least Significant Byte only
1	0	Read/Load MSB only
1	1	Read/Load LSB first then MSB

RL-Read/Load Bit Definitions

M_2	M_1	M_0	Selected Mode
0	0	0	Mode 0
0	0	1	Mode 1
x	1	0	Mode 2
x	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

$M_2M_1M_0$ Mode Select Bit Definitions

BCD	Operation
0	Hexadecimal Count
1	BCD Count

HEX/BCD Bit Definition

Fig. 6.2 *Control Word Format and Bit Definitions*

6.1.3 Operating Modes of 8254

Each of the three counters of 8254 can be operated in one of the following six modes of operation:

1. Mode 0 (Interrupt on terminal count)
2. Mode 1(Programmable monoshot)
3. Mode 2 (Rate generator)
4. Mode 3 (Square wave generator)
5. Mode 4 (Software triggered strobe)
6. Mode 5 (Hardware triggered strobe)

In this section, we will discuss all these modes of operation of 8254 in brief.

MODE 0 This mode of operation is generally called as *interrupt on terminal* count. In this mode, the output is initially low after the mode is set. The output remains low even after the count value(5) is loaded in the counter. The counter starts decrementing the count value after the falling edge of the clock, if the GATE input is high. The process of decrementing the counter continues at each falling edge of the clock till the terminal count is reached, i.e. the count becomes zero. When the terminal count is reached, the output goes high and remains high till the selected control word register or the corresponding count register is reloaded with a new mode of operation or a new count, respectively. This high output may be used to interrupt the processor whenever required, by setting a suitable terminal count. Writing a count register while the previous counting is in process, generates the following sequence of response.

The first byte of the new count when loaded in the count register, stops the previous count. The second byte when written, starts the new count, terminating the previous count then and there.

The GATE signal is active high and should be high for normal counting. When GATE goes low counting is terminated and the current count is latched till the GATE again goes high. Figure 6.3 shows the operational waveforms in mode 0.

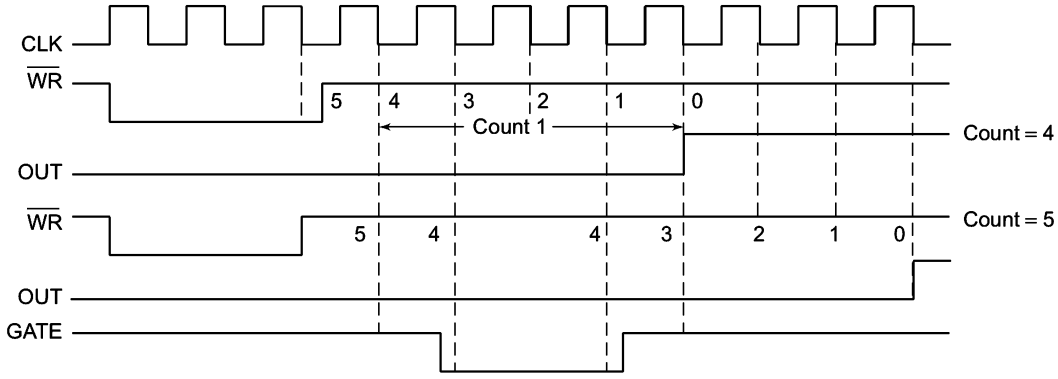


Fig. 6.3 *Waveforms of $\overline{WR}$, OUT and GATE in Mode 0*

MODE 1 This mode of operation of 8254 is called as *programmable one-shot mode*. As the name implies, in this mode, the 8254 can be used as a *monostable multivibrator*. The duration of the quasistable state of the monstable multivibrator is decided by the count loaded in the count register. The gate input is used as trigger input in this mode of operation. Normally the output remains high till the suitable count is loaded in the count register and a trigger is applied. After the application of a trigger (on the positive edge), the output goes low and remains low till the count becomes zero. If another count is loaded when the output

is already low, it does not disturb the previous count till a new trigger pulse is applied at the GATE input. The new counting starts after the new trigger pulse. Figure 6.4 shows the related waveforms for mode 1.

MODE 2 This mode is called either *rate generator* or *divide by N counter*. In this mode, if N is loaded as the count value, then, after $(N-1)$ cycles, the output becomes low only for one clock cycle. The count N is reloaded and again the output becomes high and remains so for $(N-1)$ clock pulses. The output is normally high after initialisation or even a low signal on GATE input can force the output to go high. If GATE goes high, the counter starts counting down from the initial value. The counter generates an active low pulse at the

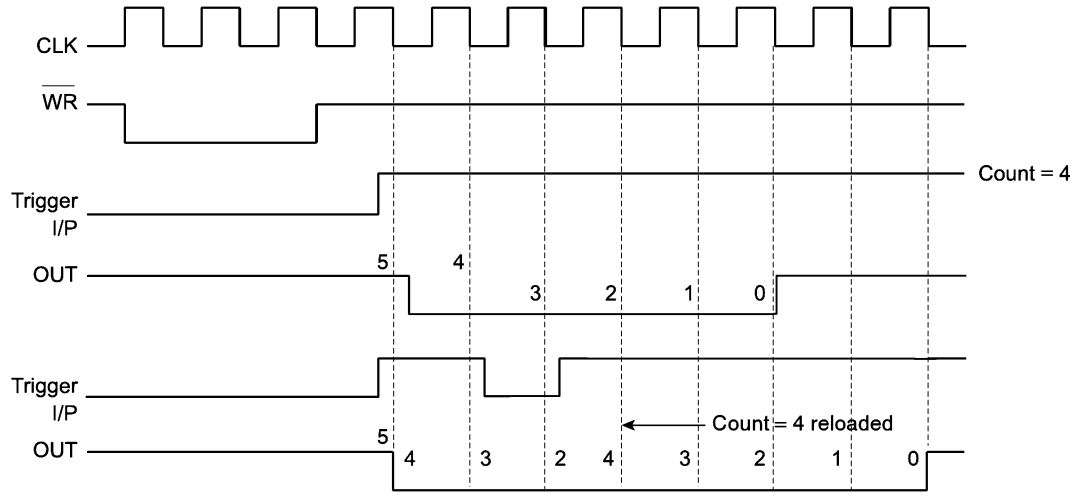


Fig. 6.4 *$\overline{WR}$, GATE and OUT Waveforms for Mode 1*

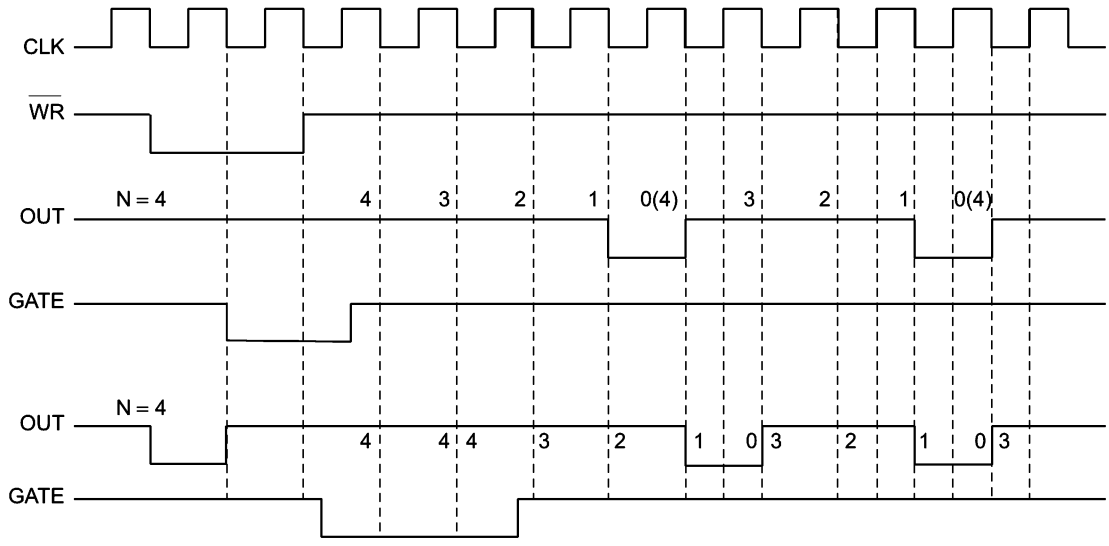


Fig. 6.5 *Waveforms at Pin $\overline{WR}$ and OUT In Mode 2*

output initially, after the count register is loaded with a count value. Then count down starts and whenever the count becomes zero another active low pulse is generated at the output. The duration of these active low pulses are equal to one clock cycle. The number of input clock pulses between the two low pulses at the output is equal to the count loaded. Figure 6.5 shows the related wave- forms for mode 2. Interestingly, the counting is inhibited when GATE becomes low.

MODE 3 In this mode, the 8254 can be used as a *square wave rate generator*. In terms of operation this mode is somewhat similar to mode 2. When, the count N loaded is even, then for half of the count, the output remains high and for the remaining half it remains low . If the count loaded is odd, the first clock pulse decrements it by 1 resulting in an even count value (holding the output high). Then the output remains high for half of the new count and goes low for the remaining half. This procedure is repeated continuously resulting in the generation of a square wave. In case of odd count , the output is high for longer duration and low for shorter duration. The difference of one clock cycle duration between the two periods is due to

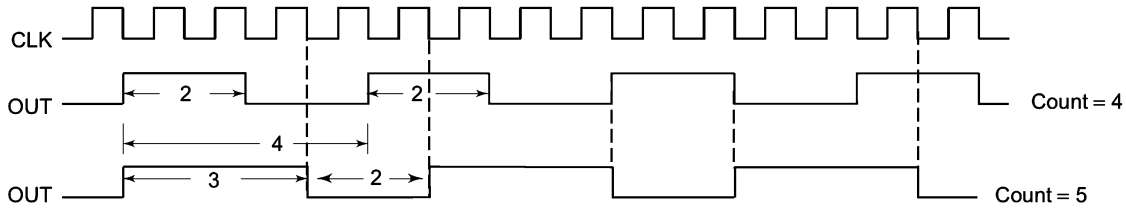


Fig. 6.6 *Waveforms for Mode 3*

the initial decrementing of the odd count. The waveforms for mode 3 are shown in Fig. 6.6. In general, if the loaded count value 'N' is odd, then for $(N+1)/2$ pulses the output remains high and for $(N-1)/2$ pulses it remains low.

MODE 4 This mode of operation of 8254 is named as *software triggerred strobe*. After the mode is set, the output goes high. When a count is loaded, counting down starts. On terminal count, the output goes low for one clock cycle,and then it again goes high. This low pulse can be used as a strobe, while interfacing the microprocessor with other peripherals. The count is inhibited and the count value is latched, when the GATE signal goes low. If a new count is loaded in the count register while the previous counting is in progress, it is accepted from the next clock cycle. The counting then proceeds according to the new count. The related waveforms are shown in Fig. 6.7.

MODE 5 This mode of operation also generates a strobe in response to the rising edge at the trigger input. This mode may be used to *generate a delayed strobe* in response to an externally generated signal. Once this mode is programmed and the counter is loaded, the output goes high. The counter starts counting after the rising edge of the trigger input (GATE). The output goes low for one clock period, when the terminal count is reached. The output will not go low until the counter content becomes zero after the rising edge of any trigger. The GATE input in this mode is used as trigger input. The related waveforms are shown in Fig. 6.8.

6.1.4 Programming and Interfacing 8254

As it is evident from the previous discussion, there may be two types of write operations in 8254, viz. (i) writing a control word into a control word register and (ii) writing a count value into a count register. The control word

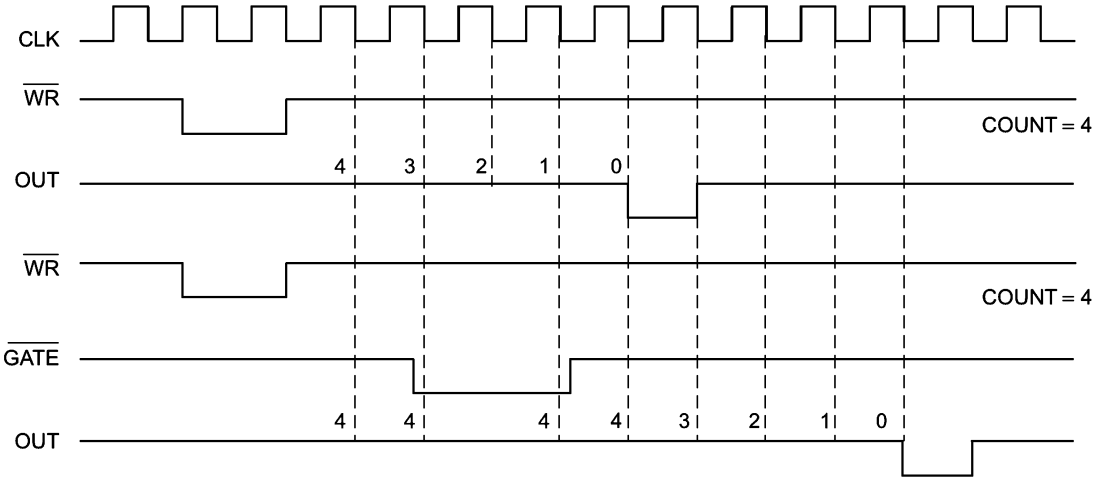


Fig. 6.7 *$\overline{WR}$, GATE and OUT Waveforms for Mode 4*

register, accepts data from the data buffer and initialises the counters, as required. The control word register contents are used for (a) initialising the operating modes (mode0–mode4) (b) selection of counters (counter0–counter2) (c) choosing binary/BCD counters (d) loading of the counter registers. The mode control register is a write only register and the CPU cannot read its contents.

One can directly write the mode control word for counter 2 or counter 1 prior to writing the control word for counter 0. Mode control word register has a separate address, so that it can be written independently. A count register must be loaded with the count value, in the same byte sequence that was

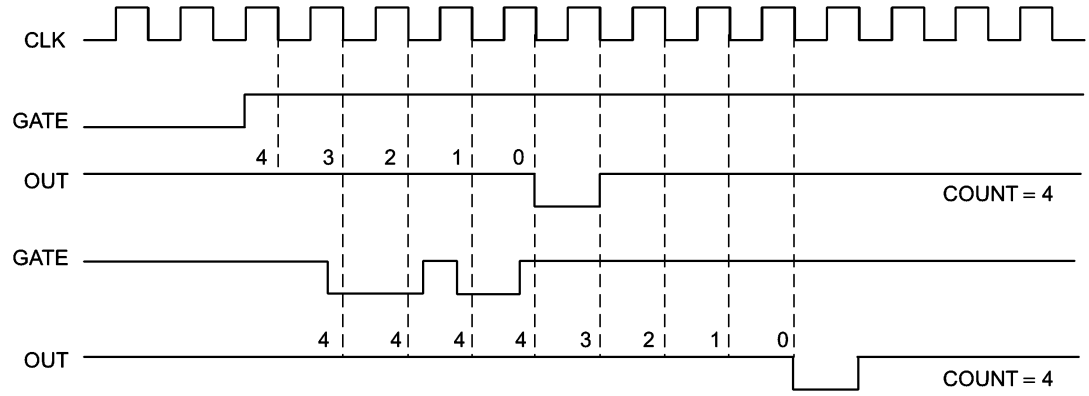


Fig. 6.8 *Waveforms in Mode 5*

programmed in the mode control word of that counter, using the bits RL_0 and RL_1. The loading of the count registers of different counters is again sequence independent. One can directly write the16-bit count register for count 2 before writing count 0 and count 1, but the two bytes in a count must be written in the byte sequence programmed using RL_0 and RL_1 bits of the mode control word of the counter. All the counters in 8254 are down counters, hence their count values go on decrementing if the CLK input pin is applied with a valid clock signal. A maximum count is obtained by loading all zeros into a count register, i.e. 2^{16} for binary counting and 10^4 for BCD counting. The 8254 responds to the negative clock edge of the clock input. The maximum operating clock frequency of 8254 is 2.6 MHz. For higher frequencies one

can use timer 8254, which operates up to 10 MHz, maintaining pin compatibility with 8254. The following Table 6.2 shows the selection of different mode control words and counter register bytes depending upon address lines A_0 and A_1.

Table 6.2 *Selection of Count Registers and Control Word Register with A_1 and A_0*

Selected Register	A_1	A_0
Mode Control Word Counter 0	1	1
Mode Control Word Counter 1	1	1
Mode Control Word Counter 2	1	1
Counter Register Byte Counter 2 LSB	1	0
Counter Register Byte Counter 2 MSB	1	0
Counter Register Byte Counter 1 LSB	0	1
Counter Register Byte Counter 1 MSB	0	1
Counter Register Byte Counter 0 LSB	0	0
Counter Register Byte Counter 0 MSB	0	0

In most of the practical applications, the counter is to be read and depending on the contents of the counter a decision is to be taken. In case of 8254, the 16-bit contents of the counter can simply be read using successive 8-bit IN operations. As stated earlier, the mode control register cannot be read for any of the counters. There are two methods for reading 8254 counter registers. In the first method, either the clock or the counting procedure (using GATE) is inhibited to ensure a stable count. Then the contents are read by selecting the suitable counter using A_0, A_1 and executing using IN instructions. The first IN instruction reads the least significant byte and the second IN instruction reads the most significant byte. Internal logic of 8254 is designed in such a way that the programmer has to complete the reading operation as programmed by him, using RL_0 and RL_1 bits of control word.

In the second method of reading a counter, the counter can be read while counting is in progress. This method, as already mentioned is called as *reading on fly*. In this method, neither clock nor the counting needs to be inhibited to read the counter. The content of a counter can be read 'on fly' using a newly defined control word register format for on-line reading of the count register. Writing a suitable control word, in the mode control register internally latches the contents of the counter. The control word format for 'read on fly' mode is given in Fig. 6.9 along with its bit definitions. After latching the content of a counter using this method, the programmer can read it using IN instructions, as discussed before.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	0	0	X	X	X	X

D_7–D_6 = SC1, SC0—Specify the counter to be selected as in Fig. 6.2

D_5–D_4 = 00—Designate counter latching operation

X = Don't Care—All other bits are neglected

Fig. 6.9 *Mode Control Word for Latching Count*

Problem 6.1

Design a programmable timer using 8254 and 8086. Interface 8254 at an address 0040H for counter 0 and write the following ALPs. The 8086 and 8254 run at 6 MHz and 1.5 MHz respectively.

(i) To generate a square wave of period 1 ms.

(ii) To interrupt the processor after 10 ms.

(iii) To derive a monoshot pulse with quasistable state duration 5 ms.

Solution

Neglecting the higher order address lines (A_{16}–A_8), the interfacing circuit diagram is shown in Fig. 6.10. The 8254 is interfaced with lower order data bus (D_0–D_7), hence A_0 is used for selecting the even bank. The A_0 and A_1 of the 8254 are connected with A_1 and A_2 of the processor. The counter addresses can be decoded as given below. If A_0 is 1, the 8254 will not be selected at all.

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	1	0	0	0	0	0	0	= 40H Counter 0
					0	1	0	= 42H Counter 1
					1	0	0	= 44H Counter 2
					1	1	0	= 46H Control word Reg.

(i) For generating a square wave, 8254 should be used in mode 3.

Let us select counter 0 for this purpose, that will be operated in BCD mode (may even be operated in HEX mode). Now suitable count is to be calculated for generating 1 ms time period.

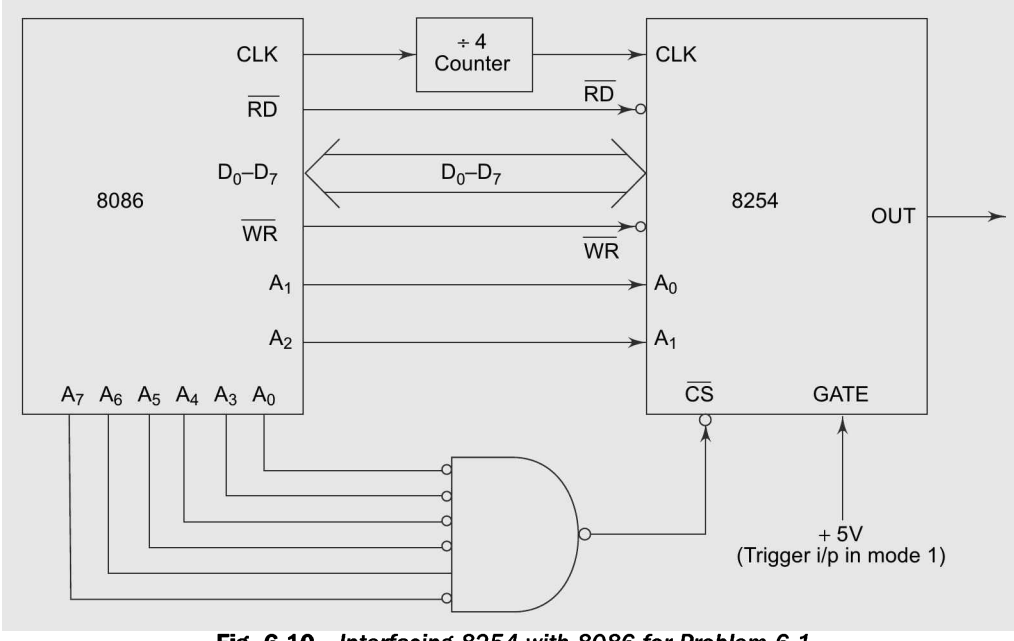


Fig. 6.10 *Interfacing 8254 with 8086 for Problem 6.1*

$f = 1.5$ MHz,

$$\Rightarrow \qquad T = \frac{1}{1.5 \times 10^{-6}} = 0.66\,\mu s$$

If N is the number of T states required for 1ms,

$$N = \frac{1 \times 10^{-3}}{0.66 \times 10^{-6}} = 1.5 \times 10^{3}$$
$$= 1500 \text{ states}$$

The control word is decided as below:

SC1	SC0	RL1	RL0	M2	M1	M0	BCD	
0	0	1	1	0	1	1	1	= 37 H

The ALP is given in Program 6.1.

```
CODE        SEGMENT
ASSUME      CS : CODE
START:      MOV AL,37H          ; Initialize 8254,
            OUT 46H,AL          ; counter 0 in mode3.
            MOV AL, 00          ; Write 00 decimal
            OUT 40H, AL         ; in LSB of count reg. and
            MOV AL, 15          ; 15 decimal in MSB as a
            OUT 40 H, AL        ; count.
            MOV AH,4CH
            INT 21H
 CODE       ENDS
            END START
```

Program 6.1 *ALP For Problem 6.1(a)*

(ii) For generating interrupt to the processor after 10 ms, the 8254 is to be used in mode 0. The OUT1 pin of 8254 is connected to interrupt input of the processor. Let us use counter 1 for this purpose, and operate the 8254 in HEX count mode.

No. of T states required for 10 ms delay $= \dfrac{10 \times 10^{-3}}{0.66 \times 10^{-6}} = 15 \times 10^{3}$
$= 15000$
$= 3A98$ H

The Control word is written below:

SC1	SC0	RL1	RL0	M2	M1	M0	BCD	
0	1	1	1	0	0	0	0	= 70 H

The ALP is written in Program 6.2.

```
CODE        SEGMENT
ASSUME      CS : CODE
START:      MOV AL, 70 H        ; Initialize 8254 with
            OUT 46H, AL         ; Counter1 in mode 0.
            MOV AL, 98H         ; Load 98H as LSB of count
            OUT 42H, AL         ; in count reg of counter1
            MOV AL, 3AH         ; then load 3AH in MSB
```

```
          OUT 42H, AL          ; of counter1
          MOV AH,4CH           ; RETURN TO DOS
          INT 21H              ;
CODE      ENDS
          END START
```

Program 6.2 *ALP for Problem 6.1(b)*

(iii) For generating a 5 ms quasistable state duration, the count required is calculated first. The counter 2 of 8254 is used in mode 1, to count in binary. The OUT2 signal normally remains high after the count is loaded, till the trigger is applied. After the application of a trigger signal, the output goes low in the next cycle, count down starts and whenever the count goes zero the output again goes high.

Number of T states required for 05 ms $= \dfrac{5 \times 10^{-3}}{0.66 \times 10^{-6}} = 7500$ states

$= 1\ D4C\ H$

The Control word is written below:

SC1	SC0	RL1	RL0	M2	M1	M0	BCD
1	0	1	1	0	0	1	0

$= B2\ H$

The ALP for the above purpose is written in Program 6.3.

```
CODE      SEGMENT
ASSUME    CS : CODE
START:    MOV AL, B2 H         ; Initialize 8254 with
          OUT 46H, AL          ; Counter 2 in mode 1
          MOV AL, 4CH          ; Load 4CH (LSB of count)
          OUT 44H, AL          ; into count register
          MOV AL, 1D           ; Load 1 DH (MSB of count)
          OUT 44H, AL          ; into count register
          MOV AH, 4CH          ; Stop
          INT2IH
CODE      ENDS
          END START
```

Program 6.3 *ALP for Problem 6.1 (c)*

Problem 6.2

Interface 8254 with 8086 at counter 0 address 7430 H and write a program to call subroutine after 100 ms. Assume that the system clock available is 2 MHz.

Solution

Address Decoding table for 8254 PIT.

Port		*HEX. address*			*Binary address*			
		A_{15} - - - - - - - - - - - - -			A_3	A_2	A_1	A_0
Counter 0	7430 H	0111	0100	0011	0	0	0	0
Counter 1	7432 H	0111	0100	0011	0	0	1	0
Counter 2	7434 H	0111	0100	0011	0	1	0	0
CWR	7436 H	0111	0100	0011	0	1	1	0

Description:

1. Input circuit signal to counter 0 is 2 MHz, Counter 0 is utilized in mode 3 to generate a square wave of 1 kHz.
2. Output square wave of counter 0 is given as circuit signal to counter 1, counter 1 is utilized in mode 0, t_d i.e. interrupt on terminal count.

Counter 1: For generating hardware delay of 100 ms counter 1 is used in mode 0 (i.e. interrupt on terminal count).

$$t_d = n \times t_c$$

where t_d = delay time

n = no. of count loaded in counter 1

t_c = time period of applied circuit

$$\therefore \quad t_c = \frac{1}{1\text{kHz}} = 1\text{ms}$$

Given $t_d = 100$ ms.

$$\therefore \quad n = \frac{t_d}{t_c} = \frac{100\text{ms}}{1\text{ms}} = 00$$

$$n = (100)_{\text{Decimal}}$$

Counter 0: For generating a square wave of 1 kHz counter 0 is used in mode 3 (i.e. square wave generation mode).

In mode 3
$$n = \frac{\text{input frequencey}}{\text{output frequency}} = \frac{2\text{MHz}}{1\text{kHz}} = \frac{2000\text{kHz}}{1\text{kHz}}$$

$$\therefore \quad n = (2000)_{\text{Decimal}}$$

2. Output square wave of counter 0 is given as Clk signal to Counter 1, Counter 1 is utilized in mode 0, i.e. interrupt on terminal count.

Counter 1: For generating hardware delay of 100 msec Counter 1 is used in mode 0 (i.e. interrupt on terminal count).

$$t_d = n \times t_c$$

Where t_d = delay time

n = count loaded in Counter 1

t_c = time period of applied circuit.

$$\therefore \quad t_c = \frac{1}{1\text{kHz}} = 1\text{ms}$$

Given $t_d = 100$ ms

$$\therefore \quad n = \frac{t_d}{t_c} = \frac{100\text{ms}}{1\text{ms}} = 100$$

$$\therefore \quad n = (100)\text{ Decimal}$$

CWR:

SC 1	SC 0	RL 1	RL 0	M 2	M 1	M 0	BCD

For Counter 0:

0	0	1	0	0	1	1	1	→ 27 H

Counter 0 Loading Only MSB Mode 3 Counter as a BCD,

$\therefore$ $n = (2000)_{\text{Decimal}}$

We have to load only MSB, i.e. $(20)_D$, whereas Counter is automatically initialized to $(00)_D$

For Counter 1:

0	1	1	0	0	0	0	1	→ 61 H

Counter 1 Loading only MSB Mode 0 BCD Counter.

$n = (0100)_D$

$\therefore$ We have to load only MSB, i.e. $(01)_D$

```
ASSUME CS : CODE, SS : STACK
STACK SEGMENT
TOP DW 100 DUP (?)
STACK ENDS
CODE SEGMENT
START  :  MOV AX, STACK
            MOV      SS, AX
            LEA        SP, TOP + 200
    MOV    DX, 7436 H          ; CWR address is transferred to DX
register.
    MOV    AL, 27H             ; Initialization of  counter 0.
    OUT    DX, AL.
    MOV    AL, 61H             ; Initialization of counter 1
    OUT    DX, AL
    MOV    AL, 20H
    MOV    DX, 7430H           ; Count 20 loaded in BCD counter 0 MSBs.
    OUT    DX, AL
    MOV    AL, 01H
    MOV    DX, 7432 H          ; Count 01 is loaded in MSBs of  counter 1
    OUT    DX, AL
    STI                        ; Set IF Flag.
    MOV    AH, 4 CH
    INT    21H
    CODE   ENDS
    END    START
```

Program 6.4 *ALP for Problem 6.2*

The circuit arrangment for this interfacing problem is as shown below:

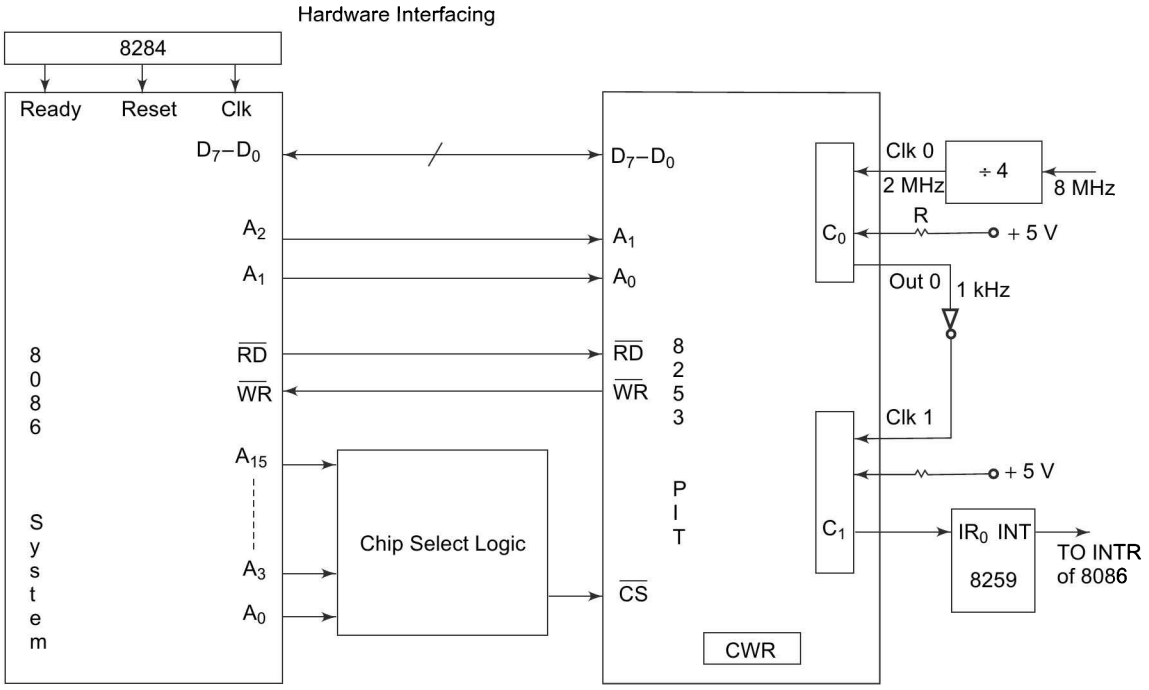


Fig. 6.11 *Interfacing 8254 with 8086 for Problem 6.2*

In all the above programs, the counting down starts as soon as the writing operation to the count register is over, and the $\overline{WR}$ pin goes high after writing. In case of 'read on fly' operation, the READ ON FLY control word for the specific counter is to be loaded in the control word register followed by the successive read operations. The 8254 and 8254 are the most widely used peripherals to generate accurate delays for industrial or laboratory purposes. The 8254 is similar to 8254 in architecture, programming and operation, hence 8254 is not discussed in this text.

6.2 PROGRAMMABLE INTERRUPT CONTROLLER 8259A

The processor 8085 had five hardware interrupt pins. Out of these five interrupt pins, four pins were allotted fixed vector addresses but the pin INTR was not allotted any vector address, rather an external device was supposed to hand over the type of the interrupt, i.e. (Type 0 to 7 for RST0 to RST7), to the microprocessor. The microprocessor then gets this type and derives the interrupt vector address from that. Consider an application, where a number of I/O devices connected with a CPU desire to transfer data using interrupt driven data transfer mode. In these types of applications, more number of interrupt pins are required than available in a typical microprocessor. Moreover, in these multiple interrupt systems, the processor will have to take care of the priorities for the interrupts, simultaneously occurring at the interrupt request pins.

To overcome all these difficulties, we require a programmable interrupt controller which is able to handle a number of interrupts at a time. This controller takes care of a number of simultaneously appearing interrupt requests along with their types and priorities. This relieves the processor from all these tasks. The programmable interrupt controller 8259A from Intel is one such device. Its predecessor 8259 was designed to operate only with 8-bit processors like 8085. A modified version, 8259A was later introduced that is compatible with 8-bit as well as 16-bit processors.

6.2.1 Architecture and Signal Descriptions of 8259A

The architectural block diagram of 8259A is shown in Fig. 6.12. The functional explanation of each block is given in the following text in brief:

Interrupt Request Register (IRR) The interrupts at IRQ input lines are handled by Interrupt Request Register internally. IRR stores all the interrupt requests in it in order to serve them one by one on the priority basis.

In-Service Register (ISR) This stores all the interrupt requests those are being served, i.e. ISR keeps a track of the requests being served.

Priority Resolver This unit determines the priorities of the interrupt requests appearing simultaneously. The highest priority is selected and stored into the corresponding bit of ISR during $\overline{\text{INTA}}$ pulse. The IR_0 has the highest priority while the IR_7 has the lowest one, normally in fixed priority mode. The priorities however may be altered by programming the 8259A in rotating priority mode.

Interrupt Mask Register (IMR) This register stores the bits required to mask the interrupt inputs. IMR operates on IRR at the direction of the Priority Resolver.

Interrupt Control Logic This block manages the interrupt and the interrupt acknowledge signals to be sent to the CPU for serving one of the eight interrupt requests. This also accepts the interrupt acknowledge (INTA) signal from CPU that causes the 8259A to release vector address on to the data bus.

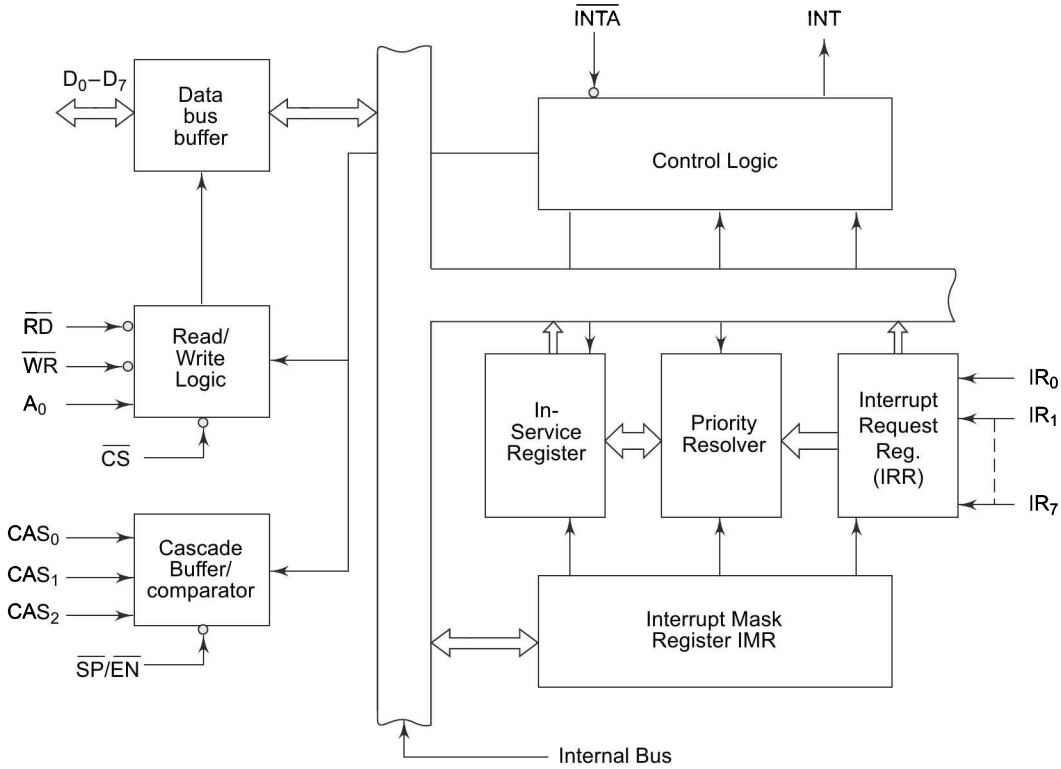


Fig. 6.12 *8259A Block Diagram*

Data Bus Buffer This tristate bidirectional buffer interfaces internal 8259A bus to the microprocessor system data bus. Control words, status and vector information pass through data buffer during read or write operations.

Read/Write Control Logic This circuit accepts and decodes commands from the CPU. This block also allows the status of the 8259A to be transferred on to the data bus.

Cascade Buffer/Comparator This block stores and compares the IDs of all the 8259As used in the system. The three I/O pins CAS0-2 are outputs when the 8259A is used as a master. The same pins act as inputs when the 8259A is in the slave mode. The 8259A in the master mode, sends the ID of the interrupting slave device on these lines. The slave thus selected, will send its pre-programmed vector address on the data bus during the next $\overline{\text{INTA}}$ pulse.

Figure 6.13 shows the pin configuration of 8259A, followed by their functional description of each of the signals in brief.

CS This is an active-low chip select signal for enabling $\overline{\text{RD}}$ and $\overline{\text{WR}}$ operations of 8259A. $\overline{\text{INTA}}$ function is independent of $\overline{\text{CS}}$.

$\overline{\text{WR}}$ This pin is an active-low write enable input to 8259A. This enables it to accept command words from CPU.

$\overline{\text{RD}}$ This is an active-low read enable input to 8259A. A low on this line enables 8259A to release status onto the data bus of CPU.

D_7–D_0 These pins form a bidirectional data bus that carries 8-bit data either to control word or from status word registers. This also carries interrupt vector information.

CAS_0–CAS_2 Cascade Lines A single 8259A provides eight vectored interrupts. If more interrupts are required, the 8259A is used in the cascade mode in which a master 8259A along with eight slaves 8259A can provide up to 64 vectored interrupt lines. These three lines act as select lines for addressing the slaves 8259A.

$\overline{\text{PS}}$ / $\overline{\text{EN}}$ This pin is a dual purpose pin. When the chip is used in buffered mode, it can be used as a buffer enable to control buffer transreceivers. If this is not used in buffered mode then the pin is used as input to designate whether the chip is used as a master ($\overline{\text{SP}} = 1$) or a slave ($\overline{\text{EN}} = 0$).

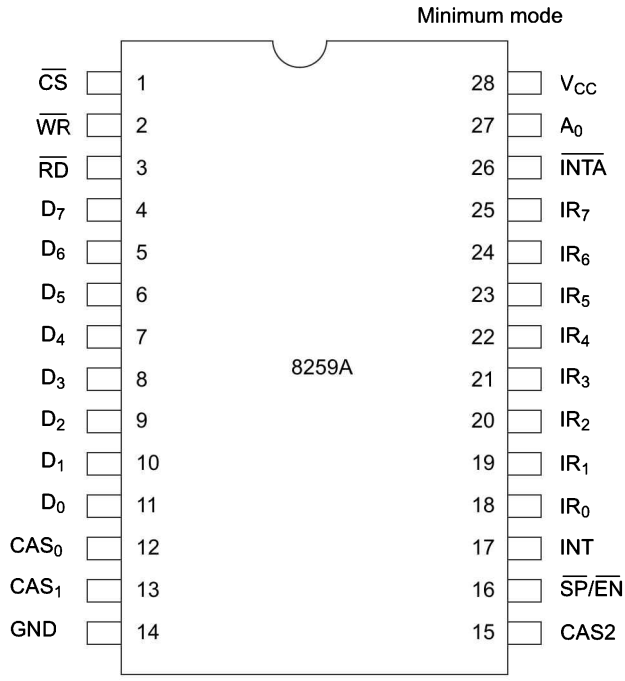


Fig. 6.13 *8259 Pin Diagram*

INT This pin goes high whenever a valid interrupt request is asserted. This is used to interrupt the CPU and is connected to the interrupt input of CPU.

IR_0–IR_7(Interrupt requests) These pins act as inputs to accept interrupt requests to the CPU. In the edge triggered mode, an interrupt service is requested by raising an IR pin from a low to a high state. It is held high until it is acknowledged, and just by latching it to high level, if used in the level triggered mode.

$\overline{INTA}$ (Interrupt acknowledge) This pin is an input used to strobe-in 8259A interrupt vector data on to the data bus. In conjunction with $\overline{CS}$, $\overline{WR}$, and $\overline{RD}$ pins, this selects the different operations like, writing command words, reading status word, etc.

The device 8259A can be interfaced with any CPU using either polling or interrupt. In polling, the CPU keeps on checking each peripheral device in sequence to ascertain if it requires any service from the CPU. If any such service request is noticed, the CPU serves the request and then goes on to the next device in sequence. After all the peripheral devices are scanned as above the CPU again starts from the first device. This type of system operation results in the reduction of processing speed because most of the CPU time is consumed in polling the peripheral devices.

In the interrupt driven method, the CPU performs the main processing task till it is interrupted by a service requesting peripheral device. The net processing speed of these type of systems is high because the CPU serves the peripheral only if it receives the interrupt request. If more than one interrupt requests are received at a time, all the requesting peripherals are served one by one on priority basis. This method of interfacing may require additional hardware if number of peripherals to be interfaced is more than the interrupt pins available with the CPU.

6.2.2 Interrupt Sequence in an 8086 System

The interrupt sequence in an 8086-8259A system is described as follows:

1. One or more IR lines are raised high that set corresponding IRR bits.
2. 8259A resolves priority and sends an INT signal to CPU.
3. The CPU acknowledges with $\overline{INTA}$ pulse.
4. Upon receiving an $\overline{INTA}$ signal from the CPU, the highest priority ISR bit is set and the corresponding IRR bit is reset. The 8259A does not drive data bus during this period.
5. The 8086 will initiate a second $\overline{INTA}$ pulse. During this period 8259A releases an 8-bit pointer on to data bus from where it is read by the CPU.
6. This completes the interrupt cycle. The ISR bit is reset at the end of the second $\overline{INTA}$ pulse if automatic end of interrupt (AEOI) mode is programmed. Otherwise ISR bit remains set until an appropriate EOI command is issued at the end of interrupt subroutine.

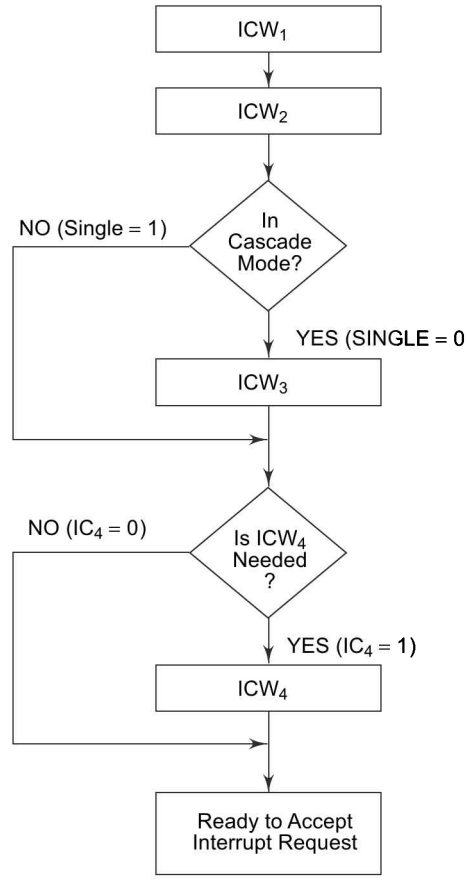


Fig. 6.14 *Initialization Sequence of 8259A*

6.2.3 Command Words of 8259A

The command words of 8259A are classified in two groups, viz. Initialization Command Words (ICWs) and Operation Command Words (OCWs).

Initialization Command Words (ICWs) Before it starts functioning, the 8259A must be initialized by writing two to four command words into the respective command word registers. These are called as Initialization Command Words (ICWs). If $A_0 = 0$ and $D_4 = 1$, the control word is recognized as ICW_1. It contains the control bits for edge/level triggered mode, single/cascade mode, call address interval and whether ICW_4 is required or not, etc. If $A_0 = 1$, the control word is recognized as ICW_2. The ICW_2 stores details regarding interrupt vector addresses. The initialisation sequence of 8259A is described in from of a flow chart in Fig. 6.14. The bit functions of the ICW_1 and ICW_2 are self explanatory as shown in Fig. 6.15.

Once ICW_1 is loaded, the following initialization procedure is carried out internally.

(a) The edge sense circuit is reset, i.e. by default 8259A interrupts are edge sensitive
(b) IMR is cleared
(c) IR7 input is assigned the lowest priority
(d) Slave mode address is set to 7
(e) Special mask mode is cleared and the status read is set to IRR
(f) If $IC_4 = 0$, all the functions of ICW_4 are set to zero. Master/slave bit in ICW_4 is used in the buffered mode only.

In an 8085 based system, $A_{15} - A_8$ of the interrupt vector address are the respective bits of ICW_2. In 8086/88 based system, five most significant bits of the interrupt type byte are inserted in place of $T_7 - T_3$ respectively and the remaining three bits (A_8, A_9 and A_{10}) are inserted internally as 000 (as if they are pointing

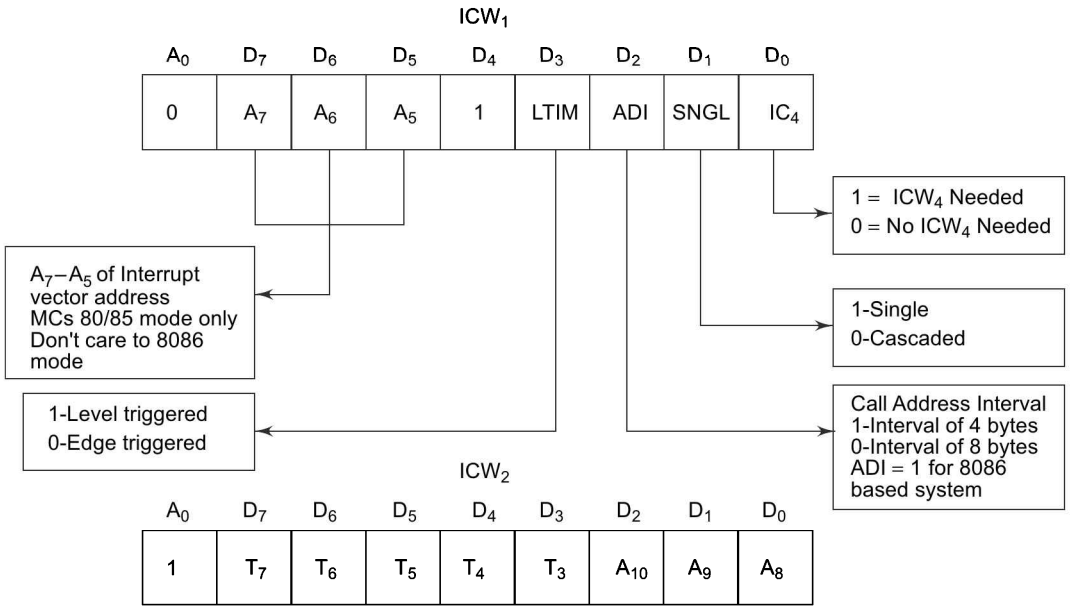


Fig. 6.15 *Initialization Command Words ICW_1 and ICW_2*

to IR_0). Other seven interrupt levels' vector addresses are internally generated automatically by 8259 using IR_0vector. Address interval is always four in an 8086 based system.

ICW_1 and ICW_2 are compulsory command words in initialization sequence of 8259A as is evident from Fig. 6.14, while ICW_3 and ICW_4 are optional. The ICW_3 is read only when there are more than one 8259As in the system, i.e. cascading is used (SNGL = 0). The SNGL bit in ICW_1 indicates whether the 8259A is in the cascade mode or not. The ICW_3 loads an 8-bit slave register. Its detailed functions are as follows:

In the master mode (i.e. $\overline{SP}$ = 1 or in buffer mode M/S = 1 in ICW_4), the 8-bit slave register will be set bit-wise to '1' for each slave in the system, as shown in Fig. 6.16. The requesting slave will then release the second byte of a CALL sequence.

In slave mode (i.e. $\overline{SP}$ = 0 or if BUF = 1 and M/S = 0 in ICW_4) bits D_2 to D_0 identify the slave, i.e. 000 to 111 for slave1 to slave8. The slave compares the cascade inputs with these bits and if they are equal, the second byte of the CALL sequence is released by it on the data bus.

ICW_4 The use of this command word depends on the IC_4 bit of ICW_1. If IC_4 = 1, ICW_4 is used, otherwise it is neglected. The bit functions of ICW_4 are described as follows:

SFNM Special fully nested mode is selected, if SFNM = 1.

BUF If BUF = 1, the buffered mode is selected. In the buffered mode, SP/EN acts as enable output and the master/slave is determined using the M/S bit of ICW_4.

M/S If M/S = 1, 8259A is a master. If M/S = 0, 8259A is a slave. If BUF = 0, M/S is to be neglected.

Master mode ICW_3

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

Sn = 1-IRn Input has a slave
= 0-IRn Input does not have a slave

Slave mode ICW_3

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	0	0	ID_2	ID_1	ID_0

$D_2D_1D_0$ - 000 to 111 for IR_0 to IR_7 or slave 1 to slave 8

Fig. 6.16 *ICW_3 in Master and Slave Mode*

AEOI If AEOI = 1, the automatic end of interrupt mode is selected.

mPM If the mPM bit is 0, the Mcs-85 system operation is selected and if mPM =1, 8086/88 operation is selected.

Figure 6.17 shows the ICW_4 bit positions:

ICW_4

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	SFNM	BUF	M/$\overline{S}$	AEOI	mPM

Fig. 6.17 *ICW_4 Bit Functions*

Operation Command Words Once 8259A is initialized using the previously discussed command words for initialisation, it is ready for its normal function, i.e. for accepting the interrupts but 8259A has its own ways of handling the received interrupts called as modes of operation. These modes of operations

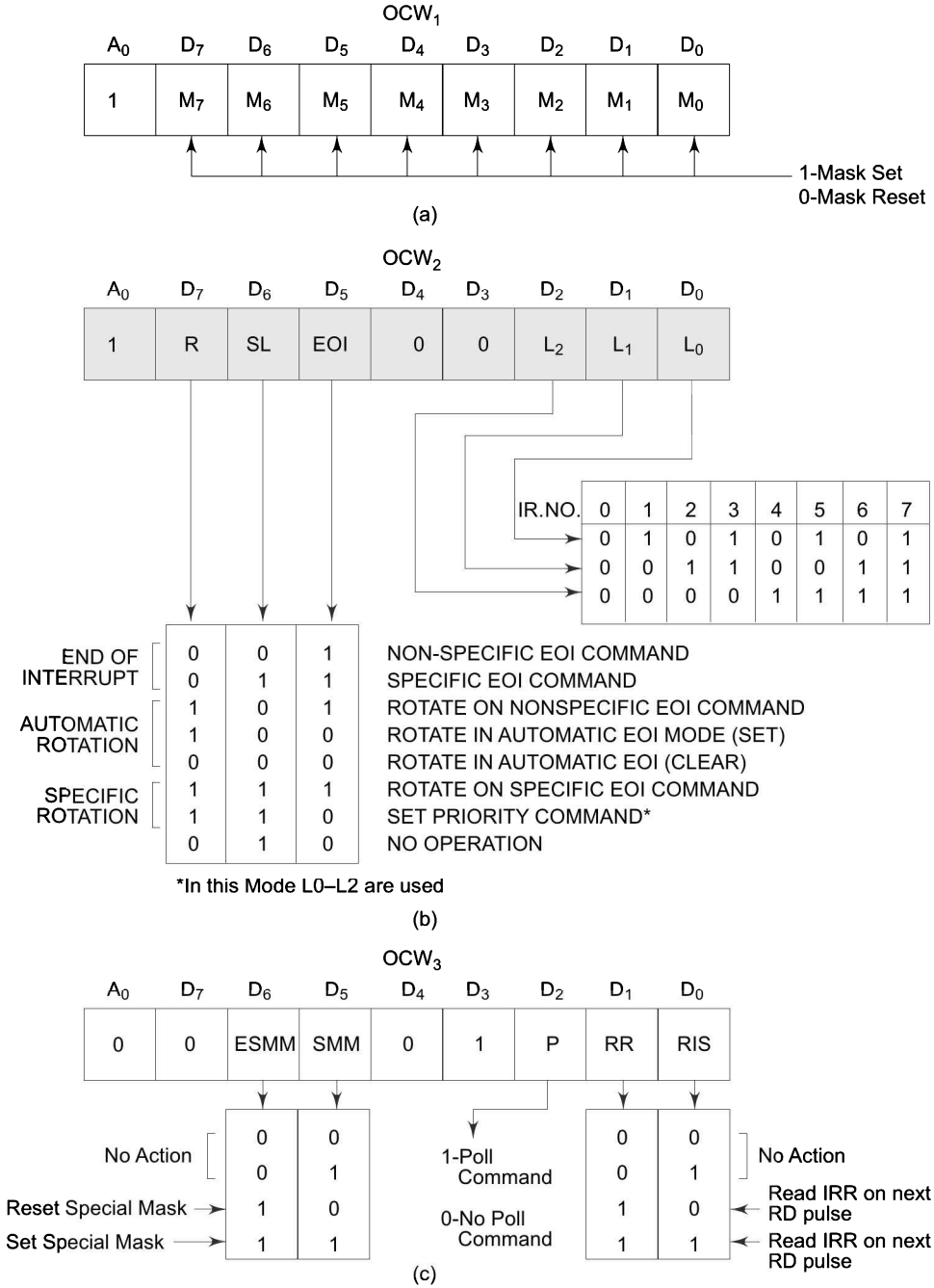


Fig. 6.18 *Operation Command Words*

can be selected by programming, i.e. writing three internal registers called as *operation command word registers*. The data written into them (bit pattern) is called as *operation command words*. In the three operation command words OCW_1, OCW_2 and OCW_3, every bit corresponds to some operational feature of the mode selected, except for a few bits those are either '1' or '0'. The three operation command words are shown in Fig. 6.18 (a), (b) and (c) with the bit selection details. OCW_1 is used to mask the unwanted interrupt requests. If the mask bit is '1', the corresponding interrupt request is masked, and if it is '0', the request is enabled. In OCW_2 the three bits, viz. R,SL and EOI control the end of interrupt, the rotate mode and their combinations as shown in Fig. 6.18 (b). The three bits L_2, L_1 and L_0 in OCW_2 determine the interrupt level to be selected for operation , if the SL bit is active, i.e. '1'. The details of OCW_2 are shown in Fig. 6.18(b).

In operation command word 3 (OCW3), if the ESMM bit, i.e. *Enable Special Mask Mode* bit is set to '1', the SMM bit is enabled to select or mask the *Special Mask Mode*. When ESMM bit is '0', the SMM bit is neglected. If the SMM bit, i.e. *Special Mask Mode* bit is '1', the 8259A will enter special mask mode provided ESMM = 1.

If ESMM = 1 and SMM = 0, the 8259A will return to the normal mask mode. The details of bits of OCW_3 are given in Fig. 6.18 (c) along with their bit definitions.

6.2.4 Operating Modes of 8259

The different modes of operation of 8259A can be programmed by setting or resting the appropriate bits of the ICWs or OCWs as discussed previously. The different modes of operation of 8259A are explained in the following text:

Fully Nested Mode This is the default mode of operation of 8259A. IR_0 has the highest priority and IR_7 has the lowest one. When interrupt requests are noticed, the highest priority request amongst them is determined and the vector is placed on the data bus. The corresponding bit of ISR is set and remains set till the microprocessor issues an EOI command just before returning from the service routine or the AEOI bit is set. If the ISR (In Service) bit is set, all the same or lower priority interrupts are inhibited but higher levels will generate an interrupt, that will be acknowledged only if the microprocessor's Interrupt enable Flag (IF) is set. The priorities can afterwards be changed by programming the rotating priority modes.

End of Interrupt (EOI) The ISR bit can be reset either with AEOI bit of ICW_1 or by EOI command, issued before returning from the interrupt service routine. There are two types of EOI commands specific and non-specific. When 8259A is operated in the modes that preserve fully nested structure, it can determine which ISR bit is to be reset on EOI. When non-specific EOI command is issued to 8259A it will automatically reset the highest ISR bit out of those already set.

When a mode that may disturb the fully nested structure is used, the 8259A is no longer able to determine the last level acknowledged. In this case a specific EOI command is issued to reset a particular ISR bit. An ISR bit that is masked by the corresponding IMR bit, will not be cleared by a non-specific EOI of 8259A, if it is in special mask mode.

Automatic Rotation This is used in the applications where all the interrupting devices are of equal priority. In this mode, an Interrupt Request (IR) level receives lowest priority after it is served while the next device to be served gets the highest priority in sequence. Once all the devices are served like this, the first device again receives highest priority.

Automatic EOI Mode Till AEOI = 1 in ICW_4, the 8259A operates in AEOI mode. In this mode, the 8259A performs a non-specific EOI operation at the trailing edge of the last $\overline{INTA}$ pulse automatically. This mode should be used only when a nested multilevel interrupt structure is not required with a single 8259A.

Specific Rotation In this mode a bottom priority level can be selected, using L_2, L_1 and L_0 in OCW_2 and R = 1, SL = 1, EOI = 0. The selected bottom priority fixes other priorities. If IR_5 is selected as a bottom priority, then IR_5 will have least priority and IR_4 will have a next higher priority. Thus IR_6 will have the highest priority. These priorities can be changed during an EOI command by programming the rotate on specific EOI command in OCW_2.

Special Mask Mode In the special mask mode, when a mask bit is set in OCW_1, it inhibits further interrupts at that level and enables interrupt from other levels, which are not masked.

Edge and Level Triggered Mode This mode decides whether the interrupt should be edge triggered or level triggered. If bit LTIM of $ICW_1 = 0$, they are edge triggered, otherwise the interrupts are level triggered.

Reading 8259 Status The status of the internal registers of 8259A can be read using this mode. The OCW_3 is used to read IRR and ISR while OCW_1 is used to read IMR. Reading is possible only in no polled mode.

Poll Command In the polled mode of operation, the INT output of 8259A is neglected, though it functions normally, by not connecting INT output or by masking INT input of the microprocessor. The poll mode is entered by setting P = 1 in OCW_3. The 8259A is polled by using software execution by microprocessor instead of the requests on INT input. The 8259A treats the next $\overline{RD}$ pulse to the 8259A as an interrupt acknowledge. An appropriate ISR bit is set, if there is a request. The priority level is read and a data word is placed on to data bus, after $\overline{RD}$ is activated. The data word is shown in Fig. 6.19.

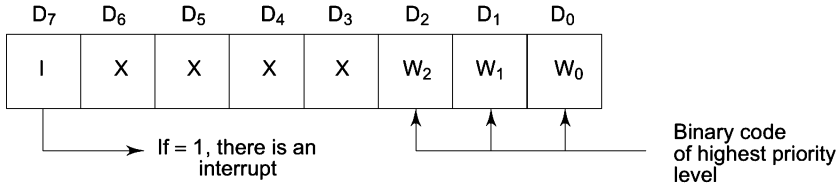


Fig. 6.19 *Data Word of 8259*

A poll command may give you more than 64 priority levels. Note that this has nothing to do with the 8086 interrupt structure and the interrupt priorities.

Special Fully Nested Mode This mode is used in more complicated systems, where cascading is used and the priority has to be programmed in the master using ICW_4. This is somewhat similar to the normal nested mode. In this mode, when an interrupt request from a certain slave is in service, this slave can further send requests to the master, if the requesting device connected to the slave has higher priority than the one being currently served. In this mode, the master interrupts the CPU only when the interrupting device has a higher or the same priority than the one currently being served. In normal mode, other requests than the one being served are masked out.

When entering the interrupt service routine the software has to check whether this is the only request from the slave. This is done by sending a non-specific EOI command to the slave and then reading its ISR and checking for zero. If its zero, a non-specific EOI can be sent to the master, otherwise no EOI should be sent. This mode is important, since in the absence of this mode, the slave would interrupt the master only once and hence the priorities of the slave inputs would have been disturbed.

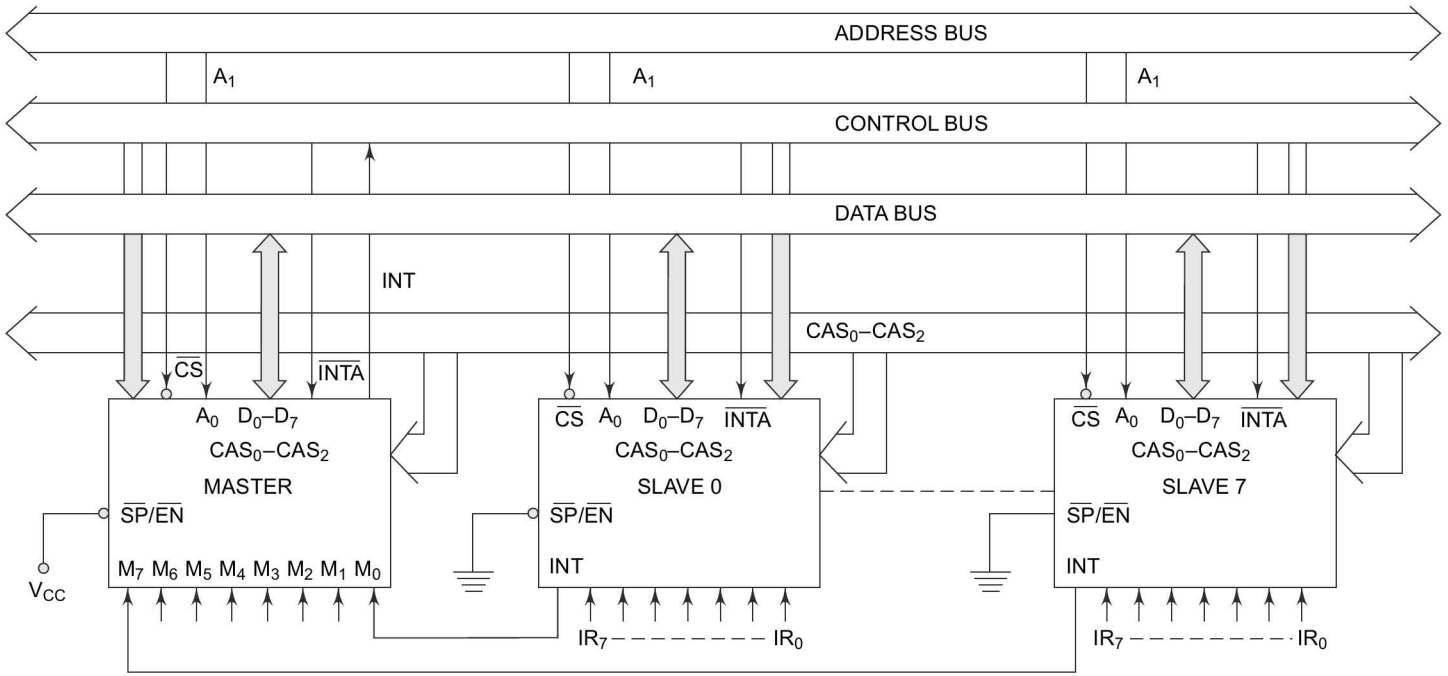


Fig. 6.20 *8259A in Cascaded Mode*

Buffered Mode When the 8259A is used in the systems in which bus driving buffers are used on data buses (e.g. cascade systems), the problem of enabling the buffers arises. The 8259A sends a buffer enable signal on $\overline{SP}/\overline{EN}$ pin, whenever data is placed on the bus.

Cascade Mode The 8259A can be connected in a system containing one master and eight slaves (maximum) to handle upto 64 priority levels. The master controls the slaves using CAS_0–CAS_2 which act as chip select inputs (encoded) for slaves. In this mode, the slave INT outputs are connected with master IR inputs. When a slave request line is activated and acknowledged, the master will enable the slave to release the vector address during the second pulse of $\overline{INTA}$ sequence. The cascade lines are normally low and contain slave address codes from the trailing edge of the first $\overline{INTA}$ pulse to the trailing edge of the second $\overline{INTA}$ pulse. Each 8259A in the system must be separately initialized and programmed to work in different modes. The EOI command must be issued twice, one for master and the other for the slave. A separate address decoder is used to activate the chip select line of each 8259A. Figure 6.20 shows the details of the circuit connections of 8259As in cascade scheme.

6.2.5 Interfacing and Programming 8259

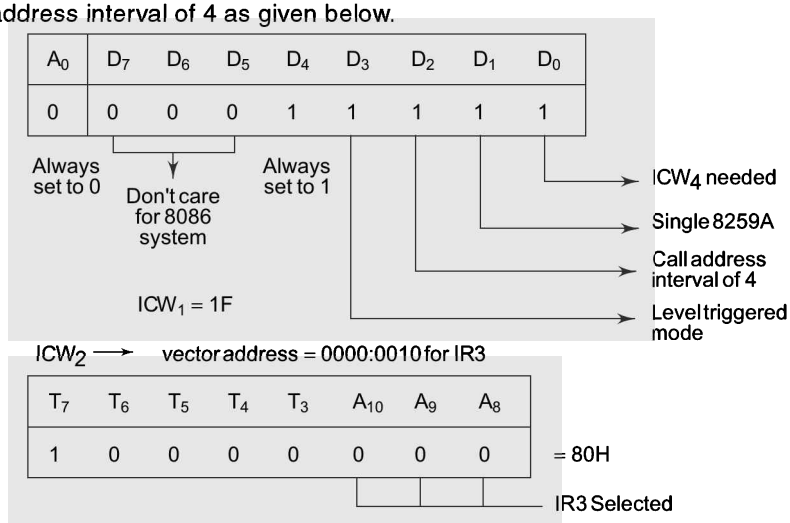
The following example elucidates the interfacing and programming of 8259 in an 8086 based system.

Problem 6.3

Show 8259A interfacing connections with 8086 at the address 074x. Write an ALP to initialize the 8259A in single level triggered mode, with call address interval of 4, non-buffered, no special fully nested mode. Then set the 8259A to operate with IR6 masked, IR4 as bottom priority level, with special EOI mode. Set special mask mode of 8259A. Read IRR and ISR into registers BH and BL respectively. IR_0 of 8259 will have type 80H.

Solution

Let the starting vector address is 0000:0200H (80H X 4 in segment 0000H). The interconnections of 8259A with 8086 are shown in Fig. 6.21. The 8259 is interfaced with lower byte of the 8086 data bus, hence A_0 line of the microprocessor system is abandoned and A_1 of the microprocessor system is connected with A_0 of the 8259A. Before going for an ALP, all the initialization command words (ICWs) and Operation Command Words (OCWs) must be decided. ICW_1 decides single level triggered, address interval of 4 as given below.



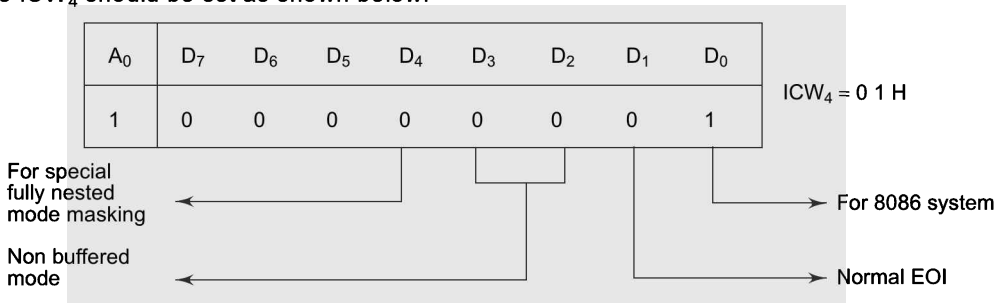
There is no slave hence the ICW_3 is as given below:

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
11	0	1	0	0	0	0	0	0

$ICW_3 = 0\ 0\ H$

Actually ICW_3 is not at all needed, because in ICW_1 the 8259 A is set for single mode.

The ICW_4 should be set as shown below:



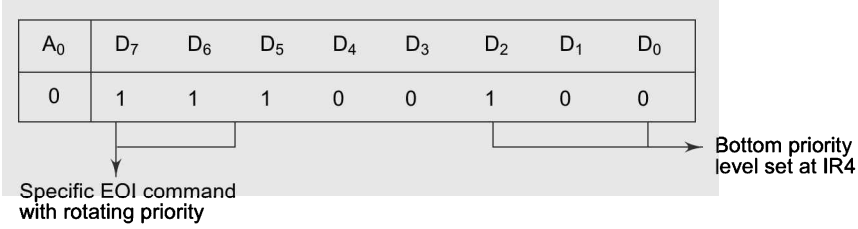
The OCW_1 sets the mask of IR6 as below:

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	1	0	0	0	0	0	0

IR6 masked

OCW1 = 40 H

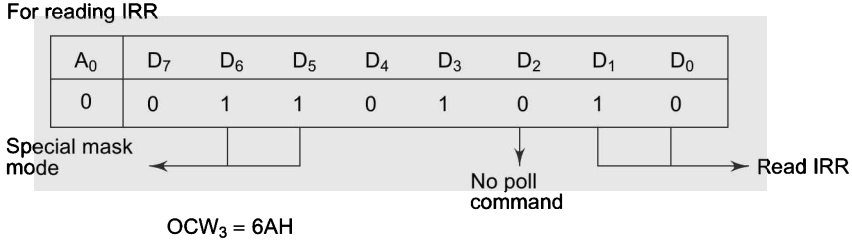
The OCW_2 sets the modes and rotating priority as shown below:

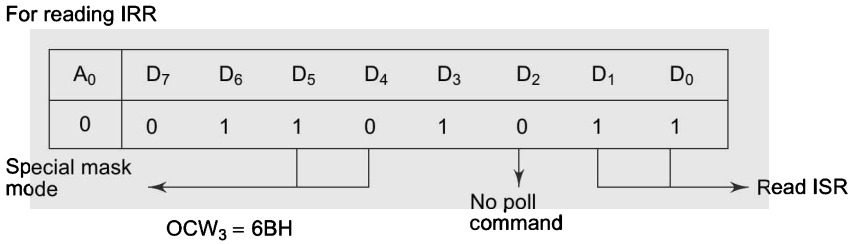


$OCW_2 = E4H$

The OCW_3 sets the special mask mode and reads ISR and IRR using the following control commands:

For reading IRR -





The following ALP writes these commands to initialize the operation of the 8259A as required in the problem. Program 6.5 gives an ALP for the required initialization of 8259A.

```
CODE      SEGMENT
ASSUME    CS : CODE
START:    MOV AL, 1FH          ; Set the 8259A in single, level
          MOV DX, 0740H
          OUT DX, AL           ; triggered mode with call address of
                                        interval of 4
          MOV DX, 0742H
          MOV AL, 80H          ; Select vector address 0000:0200H
          OUT DX, AL           ; for IR3 (ICW2)
          MOV AL, 01H          ; ICW4 for 8086 system, normal
          OUT DX, AL           ; EOI, non-buffered, special fully nested
                                        mode masked
          MOV AL, 40H          ;
          OUT DX, AL           ; OCW1 for IR6 masked,
          MOV AL, 0E4H         ; Specific EOI with rotating
          MOV DX, 0740H
          OUT DX, AL           ; Priority and bottom level of IR4 with
                                     OCW2
          MOV AL, 6AH          ; Write OCW3 for reading
          OUT DX, AL           ; IRR and store in BH
          IN AL, DX            ;
          MOV BH, AL
          MOV AL, 6BH          ; Write OCW3 to read
          OUT DX, AL           ; ISR and store in
          IN AL, DX            ; BL
          MOV BL, AL
          MOV AH, 4CH          ; Return to DOS
          INT 21H
CODE      ENDS
          END START
```

Program 6.5 *ALP to Initialize 8259A for Problem 6.3*

The interfacing circuit for Problem 6.3 has been shown below in Fig. 6.21.

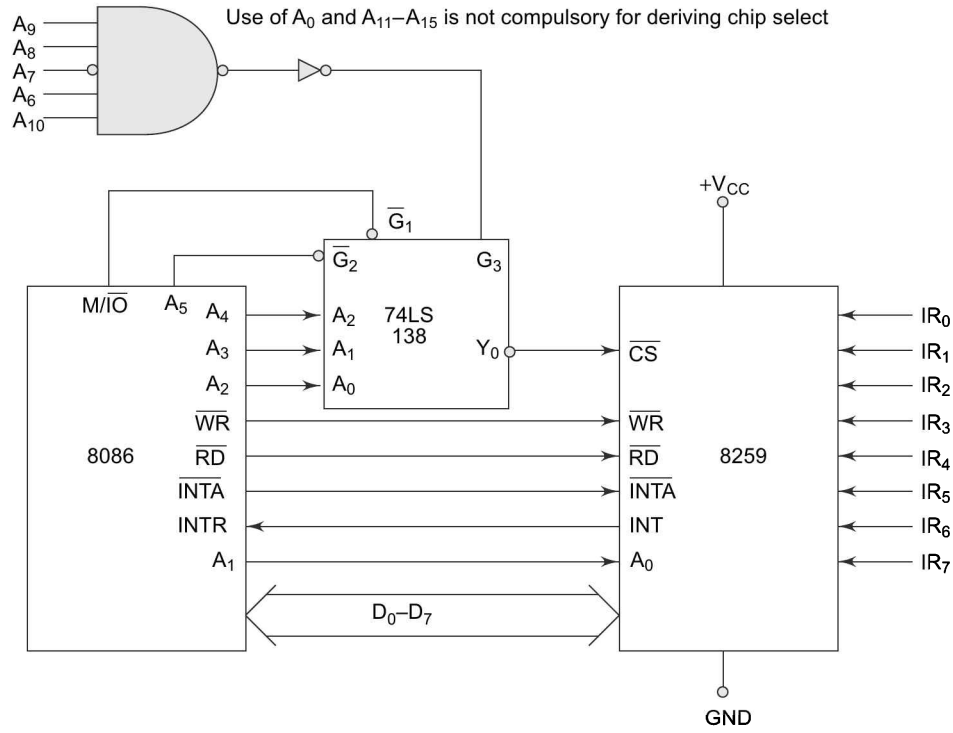


Fig. 6.21 *Interfacing 8259A with 8086*

Problem 6.4

Interface 3 ICS of 8259 PIC with 8086 system in such a way that one is master and rest two are slaves connected at IR_3 and IR_6 interrupt request level of the master. The 8259s are having vector address 60H, 70H and 80H. Write a program to initialize 8259 PIC so that IR_2 and IR_7 levels of master are masked. Initialize master in AEOI mode and automatic rotation mode in minimum mode of operation.

Solution

Address Decoding table for 8259 interfaced at even addresses.

Table 6.3

	Port	*Hex. Address*	A_{15}	A_{14}	A_1		A_0
Master	Port 0	C000H	1100	0000	0000	000	0
8259	Port 1	C002H	1100	0000	0000	001	0
$Slave_3$	Port 0	0004H	1100	0000	0000	010	0
	Port 1	C006H	1100	0000	0000	011	0
$Slave_6$	Port 0	C008H	1100	0000	0000	100	0
	Port 1	C00AH	1100	0000	0000	101	0

* A_1 Pin of 8086 system is connected with A_0 pin of 8259 PIC master as well as slave.
* A_3, A_2 are used to generate chip select for three 8259 PIC.
* $A_{15} - A_4$ used as shown in the decoder diagram
* A_0 is used to generate chip select for even bank

Control Words

ICW_1:

	X	X	X	1	LTM	X	SNGL	ICW_4	
Master	X	X	X	1	1	X	0	1	
					Or				
	0	0	0	1	1	1	1	1	19 H
Slave 3	0	0	0	1	1	0	0	1	19 H
Slave 6	0	0	0	1	1	0	0	1	19 H

ICW_2:

T_7	T_6	T_5	T_4	T_3	0	0	0	
0	1	1	0	0	0	0	0	60 H for Masks
0	1	1	1	0	0	0	0	70 H for Slave 3
1	0	0	0	0	0	0	0	80 H for Slave 6

ICW_3:

For Master

SL_7	SL_6	SL_5	SL_4	SL_3	SL_2	SL_1	SL_0	
0	1	0	0	1	0	0	0	48 H for Master

For Slave

	0	0	0	0	0	ID_2	ID_1	ID_0	
Slave 3	0	0	0	0	0	0	1	1	03 H
Slave 6	0	0	0	0	0	1	1	0	06 H

ICW_4:

	0	0	0	SFNM	BUF	M/S	AEOI	UPM	
Master	0	0	0	1	0	0	1	1	13 H
Slave 3	0	0	0	0	0	0	0	1	01 H
Slave 6	0	0	0	0	0	0	0	1	01 H

Master is initialized in SFNM in Cascade mode

OCW_1:

	M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0	
Master	1	0	0	0	0	1	0	0	84 H

OCW_2:

	R	SL	EOI	0	0	L_2	L_1	L_0	
Master	1	0	0	0	0	0	0	0	80 H

For rotating priority in AEOI mode.

Program for this problem is presented below.

```
          ASSUME CS : CODE, DS : DATA
          STACK SEGMENT
          TOP DW 100 DUP (?)        Stack segment of 200 by ks
          STACK ENDS                are reserved
          CODE SEGMENT
START :   MOV AX, STACK
          MOV SS, AX                Initialization of SS 3 SP
          LEA SP, TOP+200
          CLI
          MOV DX, 0C000H            Port 0 address of Master in DX
          MOV AL, 19H               ICWI of Master
          OUT DX, AL                Out to Port 0 of Master
          ADD DX, 02H               Port 1 address in Dx
          MOV AL, 60H               Vector address of Master for IR0
                                          (ICW2)
          OUT DX, AL                ICW2 is out to Port 1
          MOV AL, 48H               ICW3 for Master
          OUT DX, AL                ICW3 is out to Port 1
          MOV AL, 13H               ICW4  is out to Port 1
          OUT DX, AL                OCWI for Master
          MOV AL, 84H               OCWI is Out to Port 1
          OUT DX, AL
          MOV AL, 80H               OCW2 for Master
          OUT DX, AL.               OCW2 is out to Port 1
                                    Now initialization of Slave 3.
          MOV DX, 0C004H            Port address of Slave 3 in DX
          MOV AL, 19H               ICWI
          OUT DX, AL
          ADD DX, 02H               Port 1 address of Slave 3 in Dx
          MOV AL, 70H               ICW2 for Slave 3
          OUT DX, AL
          MOV AL, 03H               ICW3 for Slave 3
          OUT DX, AL
          MOV AL, 01H               ICW4 for Slave 3
          OUT DX, AL                Now initialization of Slave 6
          MOV DX, 0C008H            Port 0 address of Slave 6 in Dx
          MOV AL, 19H               ICW1 for Slave 6
          OUT DX, AL
```

```
ADD DX, 02H              Port 1 address of Slave 6 in Dx
MOV AL, 80H              ICW2 for Slave 6
OUT DX, AL
MOV AL, 06H              ICW3 for Slave 6
OUT DX, AL
MOV AH, 01H              ICW4 FOR SLAVE 6
OUT DX, AL
MOV AH, 4CH
INT 21H
CODE ENDS
END START
```

Program 6.6 *ALP to Initialize Cascaded 8259A for Problem 6.4*

Interfacing circuit for this problem is presented below in Fig. 6.22.

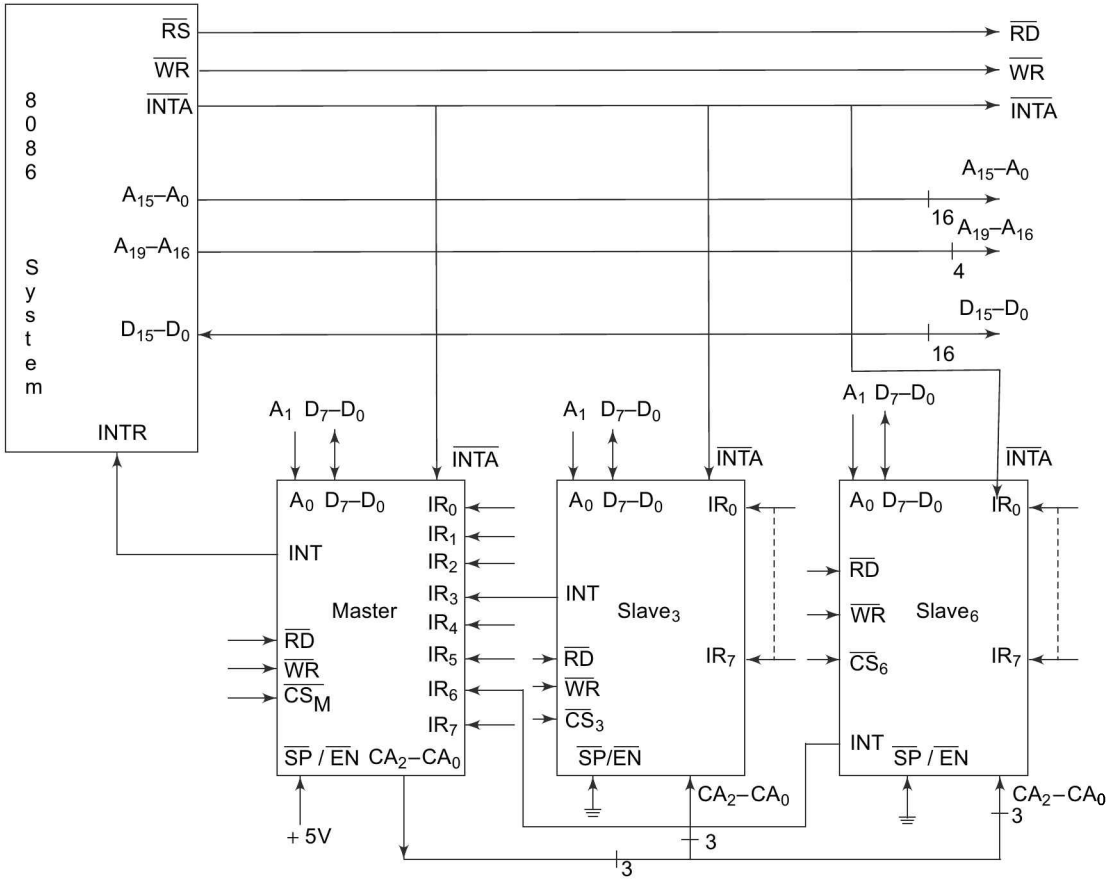


Fig. 6.22 *Cascaded 8259 Interfacing for Problem 6.4*

6.3 THE KEYBOARD/DISPLAY CONTROLLER 8279

In Chapter 5, while studying 8255, we have explained the use of 8255 in interfacing keyboards and displays with 8086. The disadvantage of this method is that the processor has to refresh the display and check the status of the keyboard periodically using polling technique. Thus a considerable amount of CPU time is wasted, reducing the system operating speed and hence the throughput.

Intel's 8279 is a general purpose keyboard display controller that simultaneously drives the display of a system and interfaces a keyboard with the CPU, leaving it free for its routine task. The keyboard-display interface scans the keyboard to identify if any key has been pressed and sends the code of the pressed key to the CPU. It also transmits the data received from the CPU, to the display device. Both of these functions are performed by the controller in repetitive fashion without involving the CPU. The keyboard is interfaced either in the interrupt or the polled mode. In the interrupt mode, the processor is requested service only if any key is pressed, otherwise the CPU can proceed with its main task. In the polled mode, the CPU periodically reads an internal flag of 8279 to check for a key pressure. The keyboard section can interface an array of a maximum of 64 keys with the CPU. The keyboard entries (key codes) are debounced and stored in an 8-byte FIFO RAM, that is further accessed by the CPU to read the key codes. If more than eight characters are entered in the FIFO (i.e. more than eight keys are pressed), before any FIFO read operation, the overrun status is set. If a FIFO contains a valid key entry, the CPU is interrupted (in interrupt mode) or the CPU checks the status (in polling) to read the entry. Once the CPU reads a key entry, the FIFO is updated, i.e. the key entry is pushed out of the FIFO to generate space for new entries. The 8279 normally provides a maximum of sixteen 7-seg display interface with CPU. It contains a 16-byte display RAM that can be used either as an integrated block of 16 × 8-bits or two 16 × 4-bit blocks of RAM. The data entry to RAM block is controlled by CPU using the command words of the 8279.

6.3.1 Architecture and Signal Descriptions of 8279

The keyboard display controller chip 8279 provides (a) a set of four scan lines and eight return lines for interfacing keyboards (b) a set of eight output lines for interfacing display. Figure 6.23 shows the functional block diagram of 8279 followed by its brief description.

I/O Control and Data Buffers The I/O control section controls the flow of data to/from the 8279. The data buffers interface the external bus of the system with internal bus of 8279. The I/O section is enabled only if $\overline{D}$ is low. The pins A_0, $\overline{RD}$ and $\overline{WR}$ select the command, status or data read/write operations carried out by the CPU with 8279.

Control and Timing Register and Timing Control These registers store the keyboard and display modes and other operating conditions programmed by CPU. The registers are written with $A_0 = 1$ and $\overline{WR} = 0$. The timing and control unit controls the basic timings for the operation of the circuit. Scan counter divide down the operating frequency of 8279 to derive scan keyboard and scan display frequencies.

Scan Counter The scan counter has two modes to scan the key matrix and refresh the display. In the encoded mode, the counter provides a binary count that is to be externally decoded to provide the scan lines for keyboard and display (four externally decoded scan lines may drive up to 16 displays). In the decoded scan mode, the counter internally decodes the least significant 2 bits and provides a decoded 1 out of 4 scan on SL_0–SL_3 (four internally decoded scan lines may drive up to 4 displays). The keyboard and display both are in the same mode at a time.

Return Buffers and Keyboard Debounce and Control This section scans for a key closure row-wise. If it is detected, the keyboard debounce unit debounces the key entry (i.e. wait for 10 ms). After the

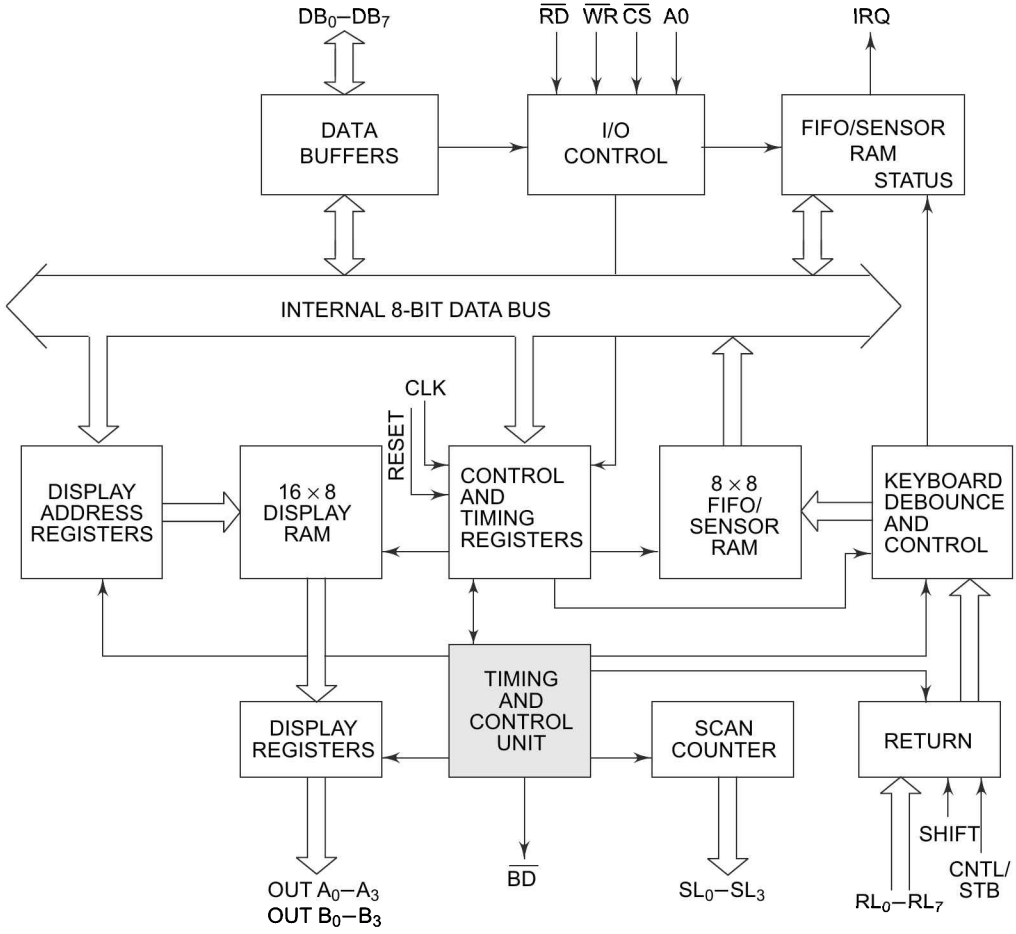


Fig. 6.23 *8279 Internal Architecture*

debounce period, if the key continues to be detected. The code of the key is directly transferred to the sensor RAM along with SHIFT and CONTROL key status.

FIFO/Sensor RAM and Status Logic In keyboard or strobed input mode, this block acts as 8-byte first-in-first-out (FIFO) RAM. Each key code of the pressed key is entered in the order of the entry, and in the meantime, read by the CPU, till the RAM becomes empty. The status logic generates an interrupt request after each FIFO read operation till the FIFO is empty. In scanned sensor matrix mode, this unit acts as sensor RAM. Each row of the sensor RAM is loaded with the status of the corresponding row of sensors in the matrix. If a sensor changes its state, the IRQ line goes high to interrupt the CPU.

Display Address Registers and Display RAM The display address registers hold the address of the word currently being written or read by the CPU to or from the display RAM. The contents of the registers are automatically updated by 8279 to accept the next data entry by CPU. The 16-byte display RAM contains the 16-bytes of data to be displayed on the sixteen 7-seg displays in the encoded scan mode.

Pin diagram of 8279 is shown below in Fig. 6.24.

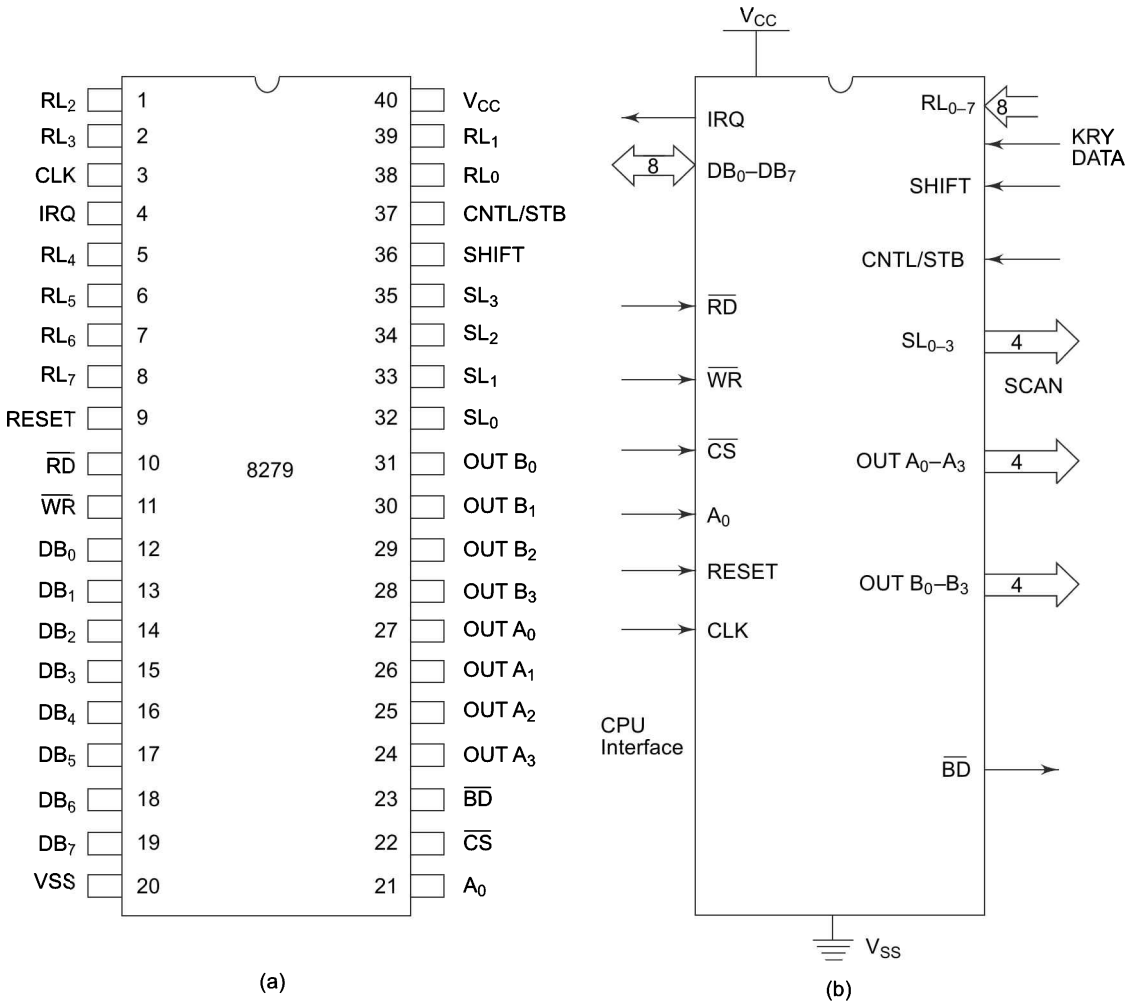


Fig. 6.24 *8279 Pin Configuration and Logic Diagram*

The signal discription of each of the pins of 8279 is presented below in brief:

DB_0–DB_7 These are bidirectional data bus lines. The data and command words to and from the CPU are transferred on these lines.

CLK This is a clock input used to generate internal timings required by 8279.

RESET This pin is used to reset 8279. A high on this line resets 8279. After resetting 8279, its in sixteen 8-bit display, left entry encoded scan, 2-key lock out mode. The clock prescaler is set to 31.

$\overline{CS}$ Chip Select: A low on this line enables 8279 for normal read or write operations. Otherwise, this pin should remain high.

A_0 A high on the A_0 line indicates the transfer of a command or status information. A low on this line indicates the transfer of data. This is used to select one of the internal registers of 8279.

$\overline{RD}$, $\overline{WR}$ (Input/Output) READ/WRITE input pins enable the data buffers to receive or send data over the data bus.

IRQ This interrupt output line goes high when there is data in the FIFO sensor RAM. The interrupt line goes low with each FIFO RAM read operation. However, if the FIFO RAM further contains any key-code entry to be read by the CPU, this pin again goes high to generate an interrupt to the CPU.

V_{ss}, V_{cc} These are the ground and power supply lines for the circuit.

SL_0–SL_3-Scan Lines These lines are used to scan the keyboard matrix and display digits. These lines can be programmed as encoded or decoded, using the mode control register.

RL_0–RL_7-Return Lines These are the input lines which are connected to one terminal of keys, while the other terminal of the keys are connected to the decoded scan lines. These are normally high, but pulled low when a key is pressed.

SHIFT The status of the shift input line is stored along with each key code in FIFO in the scanned keyboard mode. Till it is pulled low with a key closure it is pulled up internally to keep it high.

CNTL/STB-CONTROL/STROBED I/P Mode In the keyboard mode, this line is used as a control input and stored in FIFO on a key closure. The line is a strobe line that enters the data into FIFO RAM, in the strobed input mode. It has an internal pull up. The line is pulled down with a key closure.

$\overline{BD}$ -Blank Display This output pin is used to blank the display during digit switching or by a blanking command.

$OUTA_0$–$OUTA_3$ and $OUTB_0$–$OUTB_3$ These are the output ports for two 16×4 (or one 16×8) internal display refresh registers. The data from these lines is synchronized with the scan lines to scan the display and keyboard. The two 4-bit ports may also be used as one 8-bit port.

6.3.2 Modes of Operation of 8279

The modes of operation of 8279 are next discussed briefly:

(i) Input (Keyboard) modes
(ii) Output (Display) modes

Input (Keyboard) Modes 8279 provides three input modes which are discussed below in brief:

1. Scanned Keyboard Mode This mode allows a key matrix to be interfaced using either encoded or decoded scans. In the encoded scan, an 8×8 keyboard or in decoded scan, a 4×8 keyboard can be interfaced. The code of key pressed with SHIFT and CONTROL status is stored into the FIFO RAM.

2. Scanned Sensor Matrix In this mode, a sensor array can be interfaced with 8279 using either encoded or decoded scans. With encoded scan 8×8 sensor matrix or with decoded scan 4×8 sensor matrix can be interfaced. The sensor codes are stored in the CPU addressable sensor RAM.

3. Strobed input In this mode, if the control line goes low, the data on return lines, is stored in the FIFO byte by byte.

Output (Display) Modes 8279 provides two output modes for selecting the display options. These are discussed briefly:

1. Display Scan In this mode, 8279 provides 8 or 16 character multiplexed displays those can be organized as dual 4-bit or single 8-bit display units.

2. Display Entry (right entry or left entry mode) 8279 allows options for data entry on the displays. The display data is entered for display either from the right side or from the left side. The concepts regarding these modes will be more clear when the commands are discussed later in this chapter.

All these modes may be selected by programming the 8279 suitably. Further, these modes are discussed in significant details.

6.3.3 Details of Modes of Operation

Keyboard Modes

(i) Scanned Keyboard Mode with 2 Key Lockout In this mode of operation, when a key is pressed, a debounce logic comes into operation. During the next two scans, other keys are checked for closure and if no other key is pressed the first pressed key is identified. The key code of the identified key is entered into the FIFO with SHIFT and CNTL status, provided the FIFO is not full, i.e. it has at least one byte free. If the FIFO does not have any free byte, naturally the key data will not be entered and the error flag is set. If FIFO has at least one byte free, the above code is entered into it and the 8279 generates an interrupt (on IRQ line) to the CPU to inform about the previous key closures. If another key is found closed during the subsequent two scans, no entry to FIFO is made. If all the other keys are released before the first key, the keycode is entered into FIFO. If the first pressed key is released before the others, the first will be ignored. A keycode is entered to FIFO only once for each valid depression, independent of other keys pressed along with it, or released before it. If two keys are pressed within a debounce cycle (simultaneously), no key is recognized till one of them remains closed, and the other is released. The last key, that remains depressed is considered as single valid key depression.

(ii) Scanned Keyboard with N-Key Rollover In this mode, each key depression is treated independently. When a key is pressed, the debounce circuit waits for 2 keyboard scans and then checks whether the key is still depressed. If it is still depressed, the code is entered in FIFO RAM. Any number of keys can be pressed simultaneously and recognized in the order, the keyboard scan recorded them. All the codes of such keys are entered into FIFO. Note that, in this mode, the first pressed key need not be released before the second is pressed. All the keys are sensed in the order of their depression, rather in the order the keyboard scan senses them, and independent of the order of their release.

(iii) Scanned Keyboard Special Error Mode This mode is valid only under the N-Key rollover mode. This mode is programmed using *end interrupt/error mode set* command. If during a single debounce period (two keyboard scans) two keys are found pressed, this is considered a simultaneous depression and an error flag is set. This flag, if set, prevents further writing in FIFO but allows generation of further interrupts to the CPU for FIFO read. The error flag can be read by reading the FIFO status word. The error flag is set by sending normal clear command with CF = 1.

(iv) Sensor Matrix Mode In the Sensor Matrix mode, the debounce logic is inhibited. The 8-byte FIFO RAM now acts as 8×8-bit memory matrix. The status of the sensor switch matrix is fed directly to sensor RAM matrix. Thus the sensor RAM bits contains the row-wise and column-wise status of the sensors in the sensor matrix. The IRQ line goes high, if any change in sensor value is detected at the end of a sensor matrix scan or the sensor RAM has a previous entry to be read by the CPU. The IRQ line is reset by the first data read operation, if AI = 0, otherwise, by issuing the end interrupt command. AI is a bit in read sensor RAM word.

Display Modes There are various options of data display. For example, the command number of characters can be 8 or 16, with each character organised as single 8-bit or dual 4-bit codes. Similarly there are two display formats. The first one is known as left entry mode or type writer mode, since in a type writer the first character typed appears at the left-most position, while the subsequent characters appear successively to the right of the first one. The other display format is known as right entry mode, or calculator mode, since in a calculator the first character entered appears at the rightmost position and this character is shifted one position left when the next character is entered. Thus all the previously entered characters are shifted left by one position when a new character is entered.

(i) Left Entry Mode In the left entry mode, the data is entered from the left side of the display unit. Address 0 of the display RAM contains the leftmost display character and address 15 of the RAM contains the right most display character. It is just like writing in our note books, i.e. from left to write. If the 8279 is in autoincrement mode, the display RAM address is automatically updated with successive reads or writes. The first entry is displayed on the leftmost display and the sixteenth entry on the rightmost display. The seventeenth entry is again displayed at the leftmost display position.

(ii) Right Entry Mode In the right entry mode, the first entry to be displayed is entered on the rightmost display. The next entry is also placed in the right most display but after the previous display is shifted left by one display position. The leftmost character is shifted out of that display at the seventeenth entry and is lost, i.e. it is pushed out of the display RAM.

6.3.4 Command Words of 8279

All the command words or status words are written or read with $A_0 = 1$ and $\overline{CS} = 0$ to or from 8279. This section describes the various commands available in 8279.

(a) Keyboard Display Mode Set The format of the command word to select different modes of operation of 8279 is given below with its bit definitions.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_0
0	0	0	D	D	K	K	K	1

D	*D*	*Display modes*
0	0	Eight 8-bit character Left entry
0	1	Sixteen 8-bit character Left entry (Default after reset)
1	0	Eight 8-bit character Right entry
1	1	Sixteen 8-bit character Right entry

K	*K*	*K*	*Keyboard modes*
0	0	0	Encoded scan, 2 key lockout (Default after reset)
0	0	1	Decoded scan, 2 key lockout
0	1	0	Encoded scan N-Key roll over
0	1	1	Decoded scan N-Key roll over
1	0	0	Encoded scan sensor matrix
1	0	1	Decoded scan sensor matrix
1	1	0	Strobed Input Encoded scan
1	1	1	Strobed Input Decoded scan

(b) Programmable Clock The clock for operation of 8279 is obtained by dividing the external clock input signal by a programmable constant called prescaler.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_0
0	0	1	P	P	P	P	P	1

PPPPP is a 5-bit binary constant. The input frequency is divided by a decimal constant ranging from 2 to 31, decided by the bits of an internal prescaler, PPPPP.

(c) Read FIFO/Sensor RAM The format of this command is given as shown below:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_0
0	1	0	AI	X	A	A	A	1

X — don't care
AI — Auto Increment flag
AAA — Address pointer to 8 bit FIFO RAM

This word is written to set up 8279 for reading FIFO/sensor RAM. In scanned keyboard mode, AI and AAA bits are of no use. The 8279 will automatically drive data bus for each subsequent read, in the same sequence, in which the data was entered. In sensor matrix mode, the bits AAA select one of the 8 rows of RAM. If AI flag is set, each successive read will be from the subsequent RAM location.

(d) Read Display RAM This command enables a programmer to read the display RAM data.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_0
0	1	1	AI	A	A	A	A	1

The CPU writes this command word to 8279 to prepare it for display RAM read operation. AI is auto increment flag and AAAA, the 4-bit address, points to the 16-byte display RAM that is to be read. If AI = 1, the address will be automatically, incremented after each read or write to the display RAM. The same address counter is used for reading and writing.

(e) Write Display RAM

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_0
1	0	0	AI	A	A	A	A	1

AI — Auto Increment flag
AAAA — 4-bit address for 16-bit display RAM to be written
Other details of this command are similar to the 'Read Display RAM Command'.

(f) Display Write Inhibit/Blanking

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_0
1	0	1	X	IW	IW	BL	BL	1
Output nibbles		→		A	B	A	B	

The IW (Inhibit Write flag) bits are used to mask the individual nibble as shown in the above command. The output lines are divided into two nibbles ($OUTA_0$–$OUTA_3$ and $OUTB_0$–$OUTB_3$), those can be masked by setting the corresponding IW bit to 1. Once a nibble is masked by setting the corresponding IW bit to 1, the

entry to display RAM does not affect the nibble even though it may change the unmasked nibble. The blank display bit flags (BL) are used for blanking A and B nibbles. Here D_0 and D_2 corresponds to $OUTB_0$–$OUTB_3$ while D_1 and D_3 corresponds to $OUTA_0$–$OUTA_3$ for blanking and masking respectively.

If the user wants to clear the display, blank (BL) bits are available for each nibble as shown in the format. Both BL bits will have to be cleared for blanking both the nibbles.

(g) Clear Display RAM

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_0
1	1	0	CD_2	CD_1	CD_0	CF	CA	1

The CD_2, CD_1, CD_0 is a selectable blanking code to clear all the rows of the display RAM as given below. The characters A and B represent the output nibbles.

CD_2	CD_1	CD_0	
1	0	x	All zeros (x don't care) AB = 00
1	1	0	A_3–A_0 = 2 (0010) and B_3–B_0 = 00 (0000)
1	1	1	All ones (AB = FF), i.e. clear RAM

CD_2 must be 1 for enabling the clear display command. If $CD_2 = 0$, the clear display command is invoked by setting CA = 1 and maintaining CD_1, CD_0 bits exactly same as above. If CF = 1, FIFO status is cleared and IRQ line is pulled down. Also the sensor RAM pointer is set to row 0. If CA = 1, this combines the effect of CD and CF bits. Here, CA represents Clear All and CF represents Clear FIFO RAM.

End Interrupt/Error Mode Set

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_0
1	1	1	E	x	x	x	x	1

x—do not care

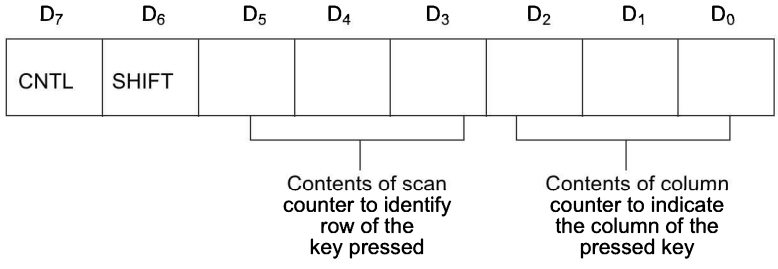
For the sensor matrix mode, this command lowers the IRQ line and enables further writing into the RAM. Otherwise, if a change in sensor value is detected, IRQ goes high that inhibits writing in the sensor RAM.

For N-key roll over mode, if the E bit is programmed to be '1', the 8279 operates in Special Error mode. Details of this mode are described in scanned keyboard special error mode.

6.3.5 Key-code and Status Data Formats

This section briefly describes the formats of the key-code/sensor data in their respective modes of operation and the FIFO Status Word formats of 8279.

Key-code Data Formats After a valid key closure, the key code is entered as a byte code into the FIFO RAM, in the following format, in scanned keyboard mode. The data format of the keycode in scanned



keyboard mode is given below. The keycode format contains 3-bit contents of the internal row counter, 3-bit contents of the column counter and status of the SHIFT and CNTL keys.

In the sensor matrix mode, the data from the return lines is directly entered into an appropriate row of sensor RAM, that identifies the row of the sensor that changed its status. The SHIFT and CNTL keys are ignored in this mode. RL bits represent the return lines. Rn represents the sensor RAM row number that is equal to the row number of the sensor array in which the status change was detected.

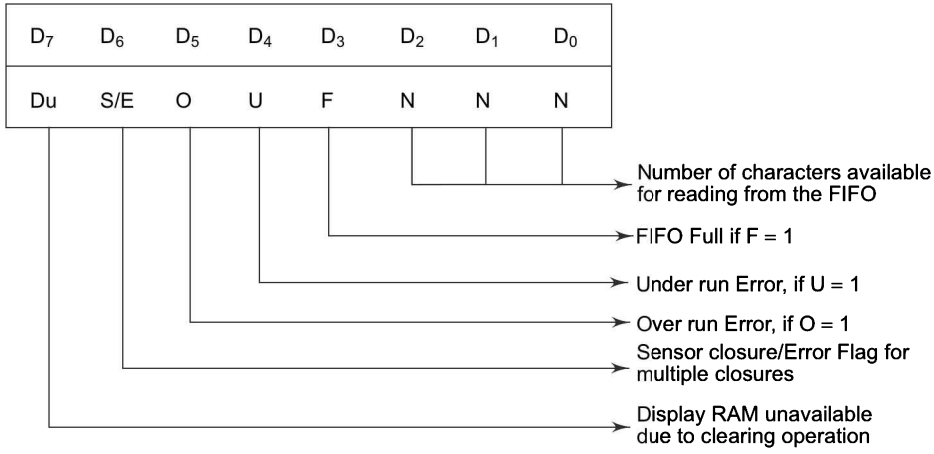
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Rn	RL_7	RL_6	RL_5	RL_4	RL_3	RL_2	RL_1	RL_0

Data Format of the Sensor Code in sensor matrix mode is given above.

In strobed input mode, data is entered to the FIFO RAM at the rising edges of CNTL/STB line, in the same format as in the sensor matrix mode.

FIFO Status Word The FIFO status word is used in keyboard and strobed input mode to indicate the error. Overrun error occurs, when an already full FIFO is attempted an entry. Underrun error occurs when an empty FIFO read is attempted. FIFO status word also has a bit to show the unavailability of FIFO RAM because of the ongoing clearing operation. In sensor matrix mode, a bit is reserved to show that at least one sensor closure indication is stored in the RAM. The S/E bit shows the simultaneous multiple closure error in special error mode.

The status word contains FIFO status, error and display unavailable signals. This is read, when $A_0 = 1$, $\overline{RD} = 0$ and $\overline{CS} = 0$. Data is read when A_0, $\overline{CS}$, $\overline{RD}$ are low. The source of data is specified by the read FIFO or read display command. The address of RAM being read is automatically incremented, if AI = 1. FIFO read always increments the address independent of AI. Data is written to the display RAM, with A_0, $\overline{WR}$ and $\overline{CS}$ tied low. The address is specified by the previous read display or write display command. The address is auto-incremented for subsequent write operations, if AI is set to. The FIFO status word format is as shown below:



6.3.6 Interfacing and Programming 8279

The following problem on 8279 interfacing and programming explains the interfacing and programming of 8279 with an 8086 microprocessor system.

Problem 6.5

Interface keyboard and display controller 8279 with 8086 at address 0080H. Write an ALP to set up 8279 in scanned keyboard mode with encoded scan, N-key rollover mode. Use a 16-character display in right entry display format. Then clear the display RAM with zeros. Read the FIFO for key closure. If any key is closed, store it's code to register CL. Then write the byte 55 to all the displays, and return to DOS. The clock input to 8279 is 2 MHz, operate it at 100 kHz.

Solution

The 8279 is interfaced with lower byte of the data bus, i.e. D_0–D_7. Hence the A_0 input of 8279 is connected with address line A_1. The data register of 8279 is to be addressed as 0080H, i.e. $A_0 = 0$. As already discussed, the data is either read from or written to this address ($A_0 = 0$). For addressing the command or status word A_0 input of 8279 should be 1 (the address line A_1 of 8086 should be 1), i.e. the address of the command word should be 0082H. Figure 6.25 shows the interfacing schematic.

The next step is to write all the required command words for this problem.

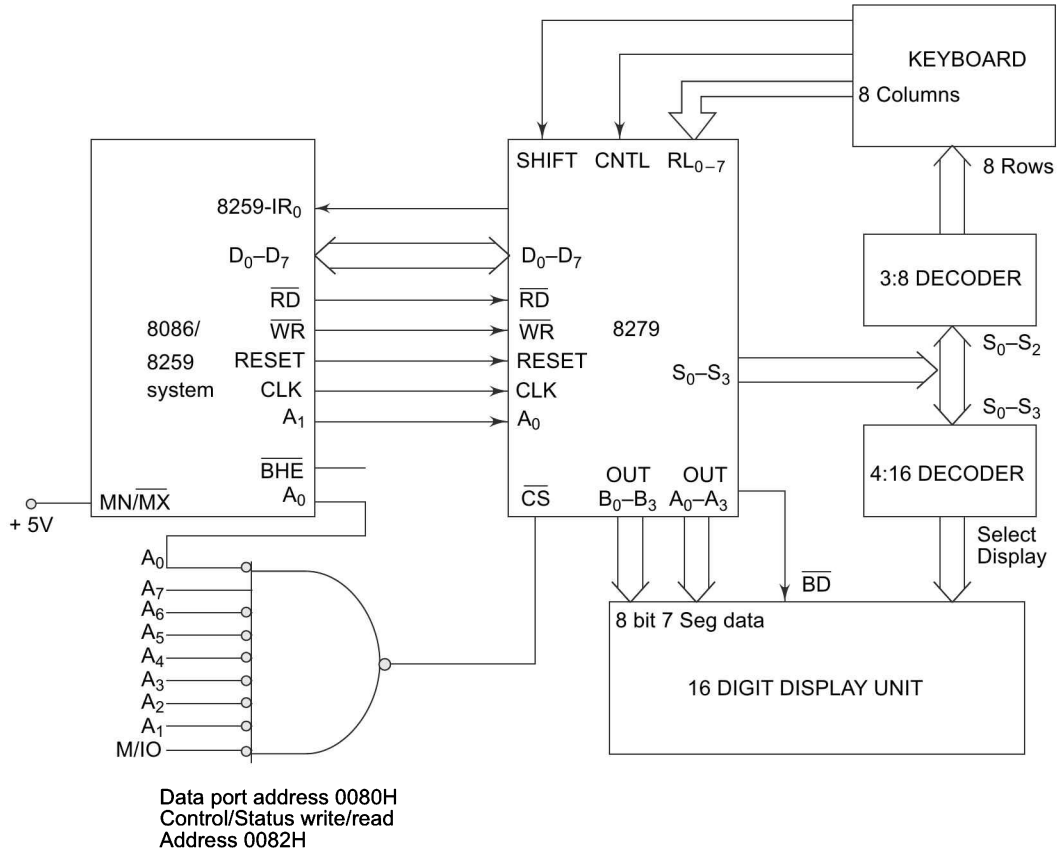
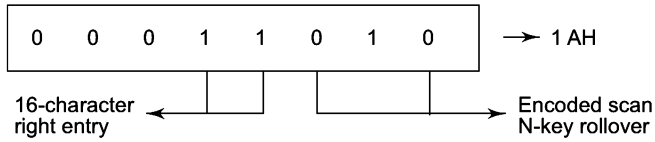
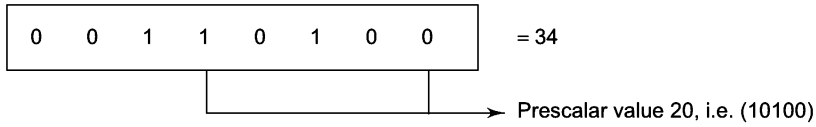


Fig. 6.25 *8279 Interfacing with 8086*

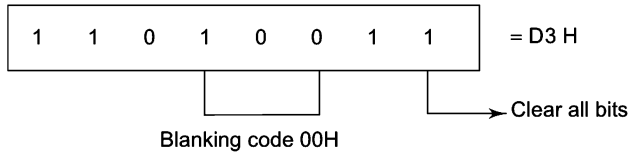
Keyboard/Display Mode Set CW This command byte sets the 8279 in 16-character right entry and encoded scan N-key rollover mode.



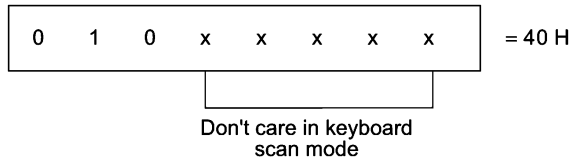
Program Clock Selection The clock input to 8279 is 2 MHz, but the operating frequency is to be 100 kHz, i.e. the clock input is to be divided by 20 (10100). Thus the prescalar value is 10100 and the command byte is set as given.



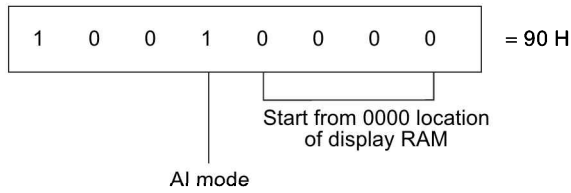
Clear Display RAM This command clears the display RAM with the programmed blanking code.



Read FIFO This command byte enables the programmer to read a key code from the FIFO RAM.



Write Display RAM This command enables the programmer to write the addressed display locations of the RAM as presented below.



Program 6.6 gives the ALP required to initialise the 8279 as required.

```
ASSUME     CS : CODE
CODE       SEGMENT
START:     MOV AL, 1AH    ; SEt 8279 in Encoded scan,
           OUT 82H, AL    ; N key rollover, 16 display, Right entry mode.
           MOV AL, 34H    ; Set clock prescalar to
```

```
            OUT 82H, AL          ; 100 kHz
            MOV AL, 0D3H         ; Clear display ram
            OUT 82H, AL          ; command
WAIT:       MOV AL, 40H          ; Read FIFO command
            OUT 82H, AL          ; for checking display RAM
            IN AL, 82H           ; Wait for clearing of
            AND AL, 80H          ; Display RAM by reading
            CMP AL, 80H          ; FIFO Du bit of the status word i.e.
            JNZ WAIT             ; If DU bit is not set wait, else proceed
            IN AL, 82H           ; Read FIFO status
            AND AL, 07H          ; Mask all bits except the
            CMP AL, 00           ; number of characters bits
            JNZ KEYCODE          ; If any key is pressed, take
WRAM:       MOV AL, 90H          ; required action, otherwise
            OUT 82H, AL          ; proceed to write display
            MOV AL, 55H          ; RAM by using write display
            MOV CH, 10H          ; command.Write the byte
NEXT:       OUT 80H, AL          ; 55H TO ALL DISPLAY RAM
            DEC CH               ; locations
            JNZ NEXT             ;
            JMP STOP             ;
KEYCODE:    CALL READCODE        ; Call routine to read the key
            MOV CL, AL           ; store the keycode in CL.
            JMP WRAM             ; code of the pressed key is assumed available
READCODE:   MOV AL, 40H
            OUT 82H, AL
            IN AL, 80H
            RET
STOP:       MOV AH, 4CH          ; stop
            INT21H
CODE        ENDS
            END START
```

Program 6.7 *Initialisation of 8279 using an 8086 ALP for Problem 6.6*

6.4 PROGRAMMABLE COMMUNICATION INTERFACE 8251 USART

Intel's 8251A is a *universal synchronous asynchronous receiver and transmitter* compatible with Intel's Processors. This may be programmed to operate in any of the serial communication modes built into it. This chip converts the parallel data into a serial stream of bits suitable for serial transmission. It is also able to receive a serial stream of bits and convert it into parallel data bytes to be read by a microprocessor.

Before presenting the detailed account of 8251, a brief look into data communication methods will be useful to the readers.

6.4.1 Methods of Data Communication

The data transmission between two points involves unidirectional or bidirectional transmission of meaningful digital data through a medium. There are basically three modes of data transmission:

(a) Simplex
(b) Duplex
(c) Half Duplex

In simplex mode, data is transmitted only in one direction over a single communication channel. For example, a computer (CPU) may transmit data for a CRT display unit in this mode. In duplex mode, data may be transferred between two transreceivers in both directions simultaneously. In the half duplex mode, on the other hand, data transmission may take place in either direction, but at a time data may be transmitted only in one direction. For example, a computer may communicate with a terminal in this mode. When the terminal sends data (i.e. terminal is sender), the message is received by the computer (i.e. the computer is receiver). However, it is not possible to transmit data from the computer to the terminal and from terminal to the computer simultaneously.

6.4.2 Architecture and Signal Descriptions of 8251

The architectural block diagram of 8251A is shown in Fig. 6.26, followed by the functional description of each block.

The data buffer interfaces the internal bus of the circuit with the system bus. The read write control logic controls the operation of the peripheral depending upon the operations initiated by the CPU. This unit also selects one of the two internal addresses those are control address and data address at the behest of the C/$\overline{D}$ signal. The modem control unit handles the modem handshake signals to coordinate the communication between the modem and the USART. The transmit control unit transmits the data byte received by the data buffer from the CPU for further serial communication. This decides the transmission rate which is controlled by the $\overline{TXC}$ input frequency. This unit also derives two transmitter status signals namely

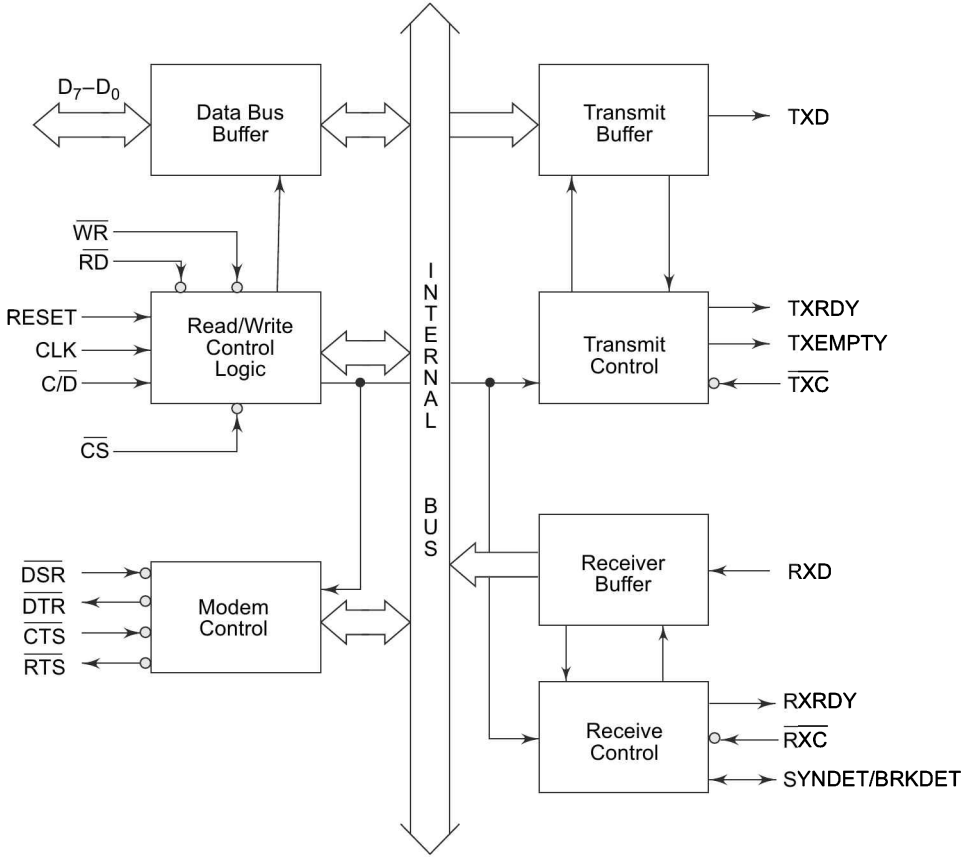


Fig. 6.26 *8251A Internal Architecture*

TXRDY and TXEMPTY. These may be used by the CPU for handshaking. The transmit buffer is a parallel to serial converter that receives a parallel byte for conversion into a serial signal and further transmission onto the communication channel. The receive control unit decides the receiver frequency as controlled by the $\overline{RXC}$ input frequency. This unit generates a receiver ready (RXRDY) signal that may be used by the CPU for handshaking. This unit also detects a break in the data string while the 8251 is in asynchronous mode. In synchronous mode, the 8251 detects SYNC characters using SYNDET/BD pin.

The pin configuration of 8251A is shown in Fig. 6.27. The following text describes the signal descriptions of 8251A:

D_0–D_7 This is an 8-bit data bus used to read or write status, command word or data from or to the 8251A.

C/$\overline{D}$–Control Word/Data This input pin, together with $\overline{RD}$ and $\overline{WR}$ inputs, informs the 8251A that the word on the data bus is either a data or control word/status information. If this pin is 1, control/status is on the bus, otherwise data is on the bus.

$\overline{RD}$ This active-low input to 8251A is used to inform it that the CPU is reading either data or status information from its internal registers.

$\overline{WR}$ This active-low input to 8251A is used to inform it that the CPU is writing data or control word to 8251A.

$\overline{CS}$ This is an active-low chip select input of 8251A. If it is high, no read or write operation can be carried out on 8251. The data bus is tristated if this pin is high.

CLK This input is used to generate internal device timings and is normally connected to clock generator output. This input frequency should be at least 30 times greater than the receiver or transmitter data bit transfer rate.

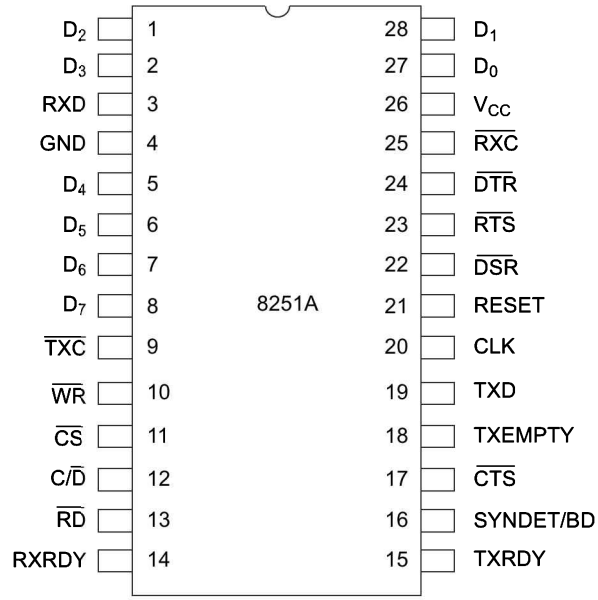


Fig. 6.27 *8251A Pin Configuration*

RESET A high on this input forces the 8251A into an idle state. The device will remain idle till this input signal again goes low and a new set of control word is written into it. The minimum required reset pulse width is 6 clock states, for the proper reset operation.

$\overline{TXC}$ Transmitter Clock Input This transmitter clock input controls the rate at which the character is to be transmitted. The baud rate (1x) is equal to the $\overline{TXC}$. frequency in synchronous transmission mode. In asynchronous mode, the baud rate is one of the three fractions, i.e. 1, 1/16 or 1/64 of the $\overline{TXC}$. The serial data is shifted out on the successive negative edge of the $\overline{TXC}$.

TXD Transmitted Data Output This output pin carries serial stream of the transmitted data bits along with other information like start bit, stop bits and parity bit, etc.

$\overline{\text{RXC}}$ Receiver Clock Input This receiver clock input pin controls the rate at which the character is to be received. In synchronous mode, the baud rate is equal to the $\overline{\text{RXC}}$ frequency. In asynchronous mode, the baud rate is one of the three fractions, i.e. 1, 1/16 and 1/64th of the $\overline{\text{RXC}}$ frequency. The received data is read into the 8251 on rising edge of $\overline{\text{RXC}}$. In most of the systems, the $\overline{\text{RXC}}$ and $\overline{\text{RXC}}$ frequencies are equal.

RXD-Receive Data Input This input pin of 8251A receives a composite stream of the data to be received by 8251A.

RXRDY-Receiver Ready Output This output indicates that the 8251A contains a character to be read by the CPU. The RXRDY signal may be used either to interrupt the CPU or may be polled by the CPU. To set the RXRDY signal in asynchronous mode, the receiver must be enabled to sense a start bit and a complete character must be assembled and then transferred to the data output register. In synchronous mode, to set the RXRDY signal, the receiver must be enabled and a character must finish assembly and then be transferred to the data output register. If the data is not successfully read from the receiver data output register before assembly of the next data byte, the overrun condition error flag is set and the previous byte is over written by the next byte of the incoming data and hence it is lost.

TXRDY-Transmitter Ready This output signal indicates to the CPU that the internal circuit of the transmitter is ready to accept a new character for transmission from the CPU. The TXRDY signal is set by a leading edge of write signal if a data character is loaded into it from the CPU. In the polled operation, the TXRDY status bit will indicate the empty or full status of the transmitter data input register.

$\overline{\text{DSR}}$-Data Set Ready This input may be used as a general purpose one bit inverting input port. Its status can be checked by the CPU using a status read operation. This is normally used to check if the data set is ready when communicating with a modem.

$\overline{\text{DTR}}$ -Data Terminal Ready This output may be used as a general purpose one bit inverting output port. This can be programmed low using the command word. This is used to indicate that the device is ready to accept data when the 8251 is communicating with a modem.

$\overline{\text{RTS}}$-Request to Send Data This output also may be used as a general purpose one bit inverting output port that can be programmed low to indicate the modem that the receiver is ready to receive a data byte from the modem. This signal is used to communicate with a modem.

$\overline{\text{CTS}}$ -Clear to Send If the clear to send the input line is low, the 8251A is enabled to transmit the serial data, provided the enable bit in the command byte is set to '1'. If a Tx disable or $\overline{\text{CTS}}$ divsable command occurs, while the 8251A is transmitting data, the transmitter transmits all the data written to the USART prior to disabling the $\overline{\text{CTS}}$ or Tx. If the $\overline{\text{CTS}}$ disable or Tx disable command occurs just before the last character appears in the serial bit string, the character will be retransmitted again whenever the $\overline{\text{CTS}}$ is enabled or the Tx enable occurs.

TXE-Transmitter Empty If the 8251A, while transmitting, has no characters to transmit, the TXE output goes high and it automatically goes low when a character is received from the CPU, for further transmission. In synchronous mode, a 'high' on this output line indicates that a character has not been loaded and

the SYNC character or characters are about to be or are being transmitted automatically as 'fillers'. The TXE signal can be used to indicate the end of a transmission mode.

SYNDET/BD-Synch Detect/Break Detect This pin is used in the synchronous mode for detecting SYNC characters (SYNDET) and may be used as either input or output. This can be programmed using the control word. After resetting, it is in the output mode. When used as an output, the SYNDET pin will go high to indicate that the 8251A has located a SYNC character in the receive mode. The SYNDET signal is automatically reset upon a following status read operation. When this is used as input, a positive going signal will cause the 8251A to start assembling a data character on the rising edge of the next RXC.

In the asynchronous mode, the pin acts as a break detect output. This goes high whenever the RXD pin remains low through two consecutive stop bit sequences. A stop bit sequence contains a stop bit, a start bit, data bits and parity bits. This is reset only with master chip reset or the RXD returning high. If the RXD returns to '1', during the last bit of the next character after the break, the break detect is latched up. The 8251A can now be cleared only with chip reset.

6.4.3 Description of 8251A Operating Modes

The 8251A can be programmed to operate in its various modes using its mode control words. A set of control words is written into the internal registers of 8251A to make it operate in the desired mode.

Once the 8251A is programmed as required, the TXRDY output is raised 'high' to signal the CPU that the 8251A is ready to receive a data byte from it that is to be further converted to serial format and transmitted. This automatically goes low when CPU writes a data byte into 8251A. In receiver mode, the 8251A receives a serial data byte from a modem or an I/O device. After receiving the entire data byte, the RXRDY signal is raised high to inform the CPU that the 8251A has a character ready for it. The RXRDY signal is automatically reset after the CPU reads the received byte from the 8251A . The 8251A cannot initiate transmission until the TX enable bit in the command word is set and a $\overline{\text{CTS}}$ signal is received from the modem or receiving I/O device.

The control words of 8251A are divided into two functional types:

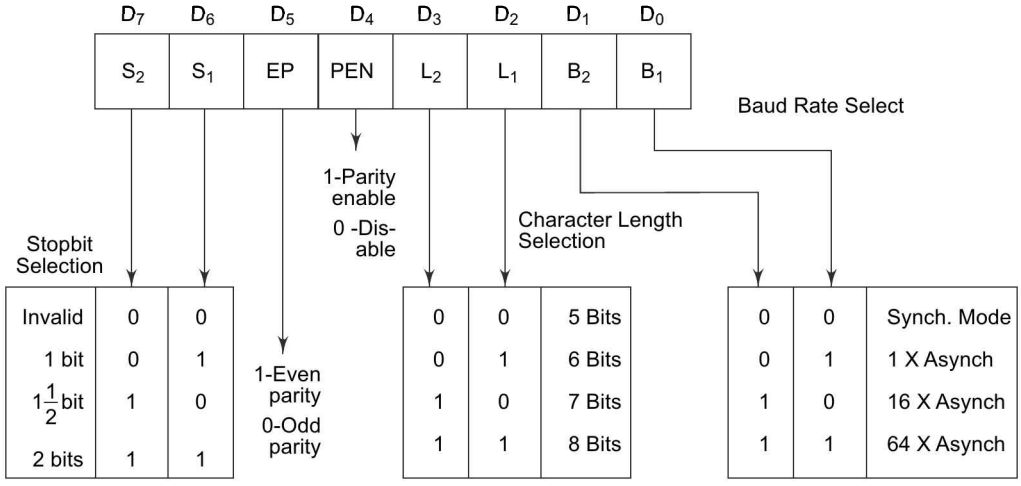
1. Mode Instruction control word
2. Command Instruction control word

Asynchronous Mode

Mode Instruction Control Word This defines the general operational characteristics of 8251A. After internal (reset command) or external (reset input pin) reset, this must be written to configure the 8251A as per the required operation. Once this has been written into 8251A, SYNC characters or command instructions may be programmed further as per the requirements. To change the mode of operation from synchronous to asynchronous or vice-a-versa, the 8251A has to be reset using master chip reset.

Figure 6.28 shows asynchronous mode instruction control word format.

Asynchronous Mode (Transmission) When a data character is sent to 8251A by the CPU, it adds start bits prior to the serial data bits, followed by optional parity bit and stop bits using the asynchronous mode instruction control word format. This sequence is then transmitted using TXD output pin on the falling edge of $\overline{\text{TXC}}$. When no data characters are sent by the CPU to 8251A the TXD output remains 'high', if a 'break' has not been detected.



N.B.—Stop bit selection as above is only for transmitter. Receiver never requires more than one stop bit

Fig. 6.28 *Mode Instruction Format Asynchronous Mode*

Asynchronous Mode (Receive) A falling edge on RXD input line marks a start bit. At baud rates of 16x and 64x, this start bit is again checked at the center of start bit pulse and if detected low, it is a valid start bit which starts counting. The bit counter locates the data bits, parity bit and stop bit. If a parity error occurs, the parity error flag is set. If a low level is detected as the stop bit, the framing error flag is set. The receiver requires only one stop bit to mark end of the data bit string, regardless of the stop bit programmed at the transmitting end. This 8-bit character is then loaded into parallel I/O buffer of 8251A. RXRDY pin is then raised high to indicate to the CPU that a character is ready for it. If the previous character has not been read by the CPU, the new character replaces it, and the overrun flag is set indicating that the previous character is lost. These error flags can be cleared using an error reset instruction. Figure 6.29 shows asynchronous mode transmission and receiver data formats. If character length is 5 to 7 bits then the remaining bits are set to zero.

Synchronous Mode Figure 6.30 shows the synchronous mode instruction format with its bit definitions.

Synchronous Mode (Transmission) The TXD output is high until the CPU sends a character to 8251A which usually is a SYNC character. When $\overline{\text{CTS}}$ line goes low, the first character is serially transmitted out. All the characters are shifted out on the falling edges of $\overline{\text{TXC}}$. Data is shifted out at the same rate as $\overline{\text{TXC}}$, over TXD output line. If the CPU buffer becomes empty, the SYNC character or characters are inserted in the data stream over TXD output. The TXEMPTY pin is raised high to indicate that the 8251A is empty (i.e. it does not have any byte to transmit) and is transmitting SYNC characters. The TXEMPTY pin is reset, automatically when a data character is written to 8251A by the CPU. Figure 6.31 shows the relation between TXEMPTY and SYNC character insertion.

Synchronous Mode (Receiver) In this mode, the character synchronization can be achieved internally or externally. If this mode is programmed, then 'ENTER HUNT' command should be included in the first command instruction word written into the 8251A. The data on RXD pin is sampled on rising edge of the $\overline{\text{RXC}}$.

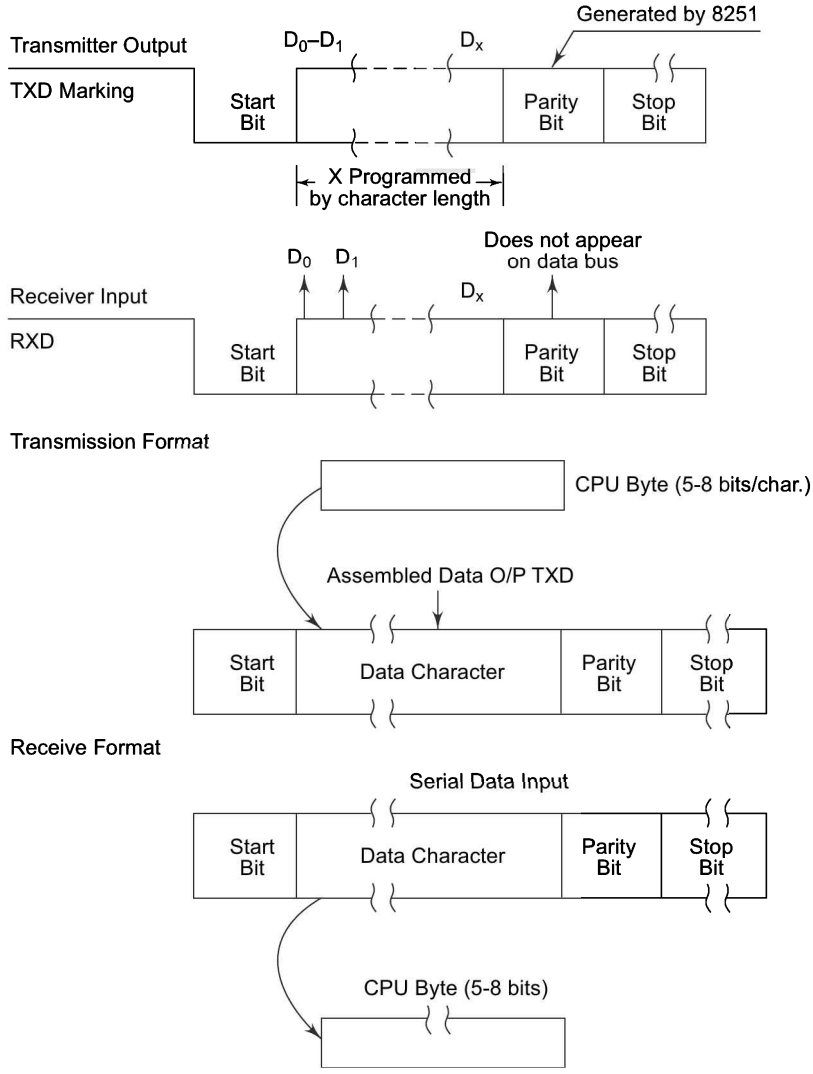
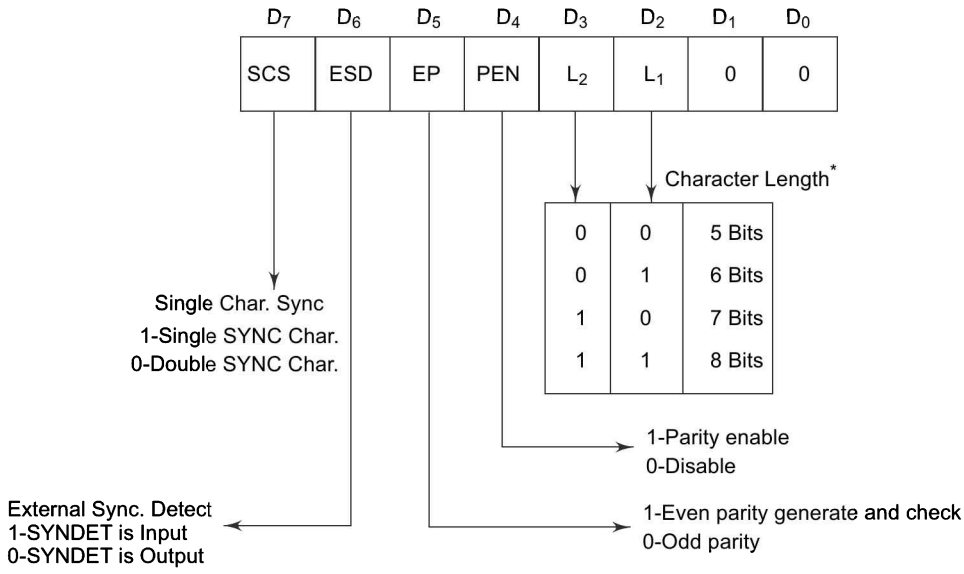


Fig. 6.29 *Asynchronous Mode Transmit and Receive Formats*

The content of the receiver buffer is compared with the first SYNC character at every edge until it matches. If 8251A is programmed for two SYNC characters, the subsequent received character is also checked. When both the characters match, the hunting stops. The SYNDET pin is set high and is reset automatically by a status read operation. If a parity bit is programmed, the SYNDET signal will not go as high as the middle of parity bit, or till the middle of the last data bit.

In the external SYNC mode, synchronization is achieved by applying a high level on the SYNDET input pin, that forces 8251A out of HUNT mode. The high level can be removed after one $\overline{\text{RXC}}$ cycle. An ENTER HUNT command has no meaning in asynchronous mode. The parity and overrun error both are checked in the same way as in asynchronous mode. Figure 6.32 shows synchronous mode transmit and receive data formats.



* If the character size less than 8-bits, the remaining bits are set to 'O'.

Fig. 6.30 *Synchronous Mode Instruction Format*

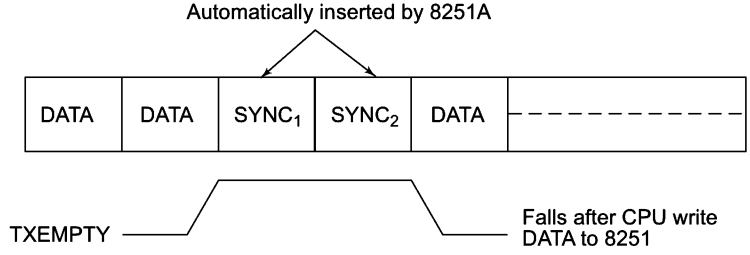


Fig. 6.31 *TXEMPTY Signal and SYNC Characters*

Command Instruction Definition The command instruction controls the actual operations of the selected format like enable transmit/receive, error reset and modem control. Once the mode instruction has been written into 8251A and the SYNC characters are inserted internally by 8251A, all further control words written with C/$\overline{D}$ = 1 will load a command instruction. A reset operation returns 8251A back to mode instruction format. The command instruction format is shown in Fig. 6.33, with its bit definitions.

Status Read Definition This definition is used by the CPU to read the status of the active 8251A to confirm if any error condition or other conditions like the requirement of processor service has been detected, during the operation.

A read command is issued by processor with C/$\overline{D}$ = 1 to accomplish this function. Some of the bits in the definition have the same significances as those of the pins of 8251A. These are used to interface the 8251A in a polled configuration, besides the interrupt controlled mode. The pin TXRDY is an exception. The status 'read format' is shown in Fig. 6.34, with its bit definitions.

* N.B.—If the character size is less than 8-bits, the remaining bits are set to '0'.

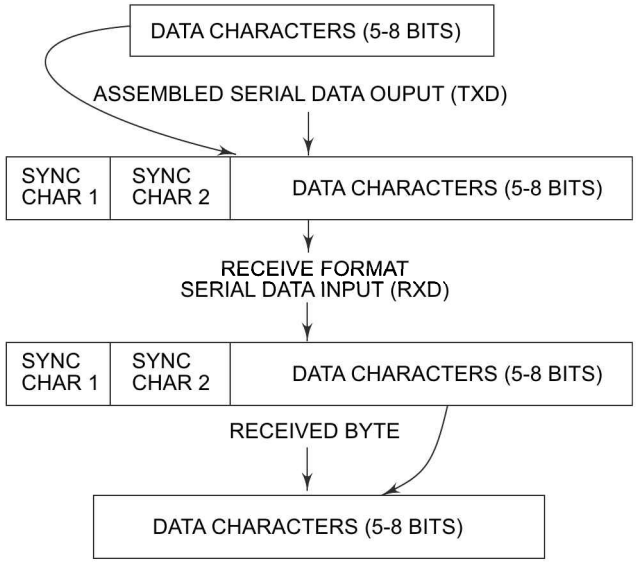


Fig. 6.32 *Data Formats of Synchronous Mode*

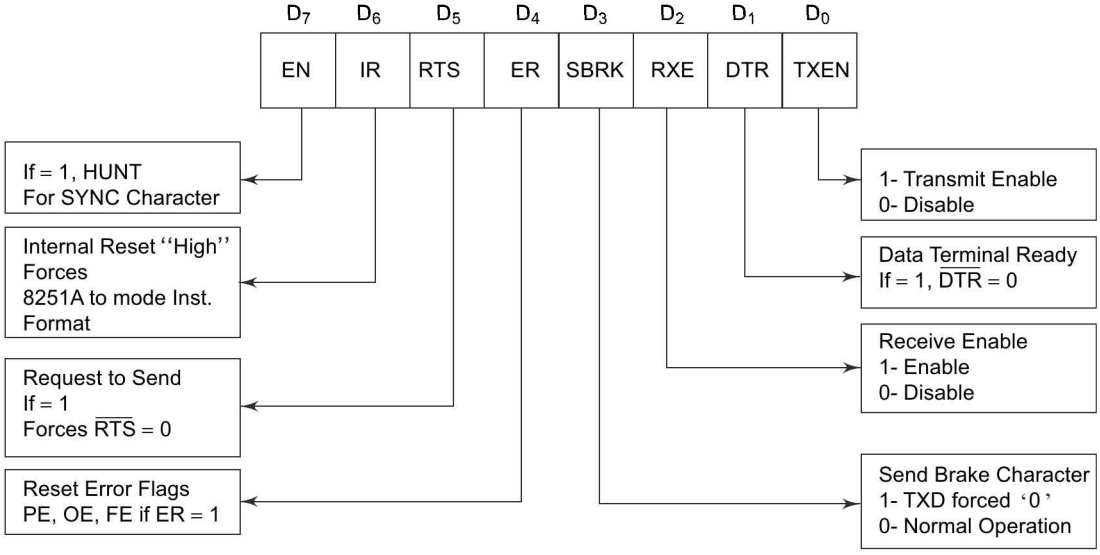


Fig. 6.33 *Command Instruction Format*

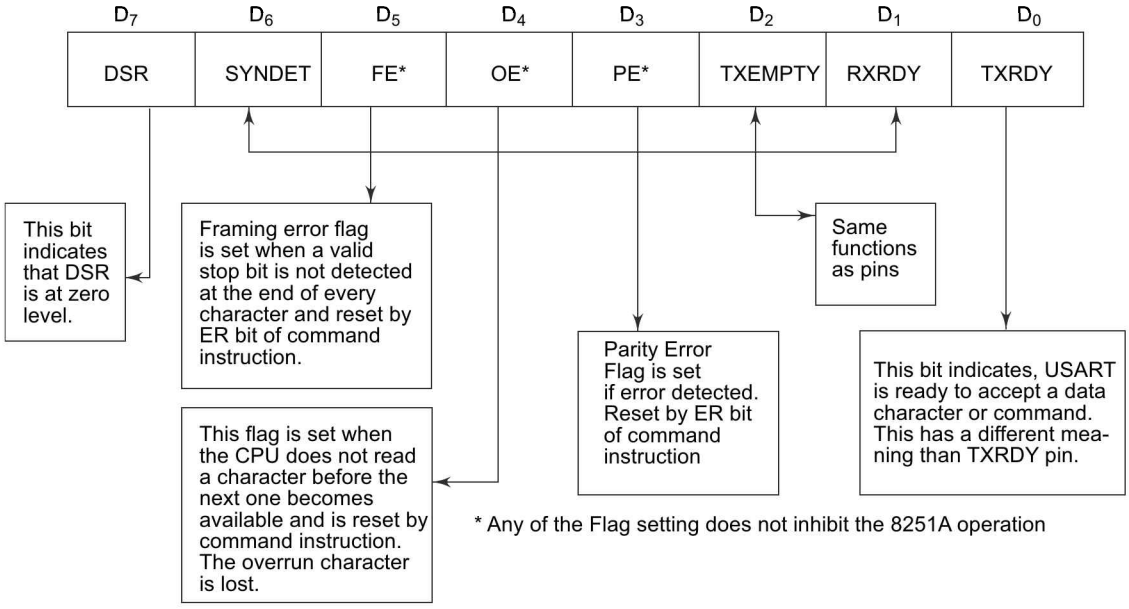


Fig. 6.34 *Status Read Instruction Format*

6.4.4 Interfacing and Programming 8251 with 8086

The following problem explains the interfacing and programming of 8251A in an 8086 system.

Problem 6.7

Design the hardware interface circuit for interfacing 8251 with 8086. Set the 8251A in asynchronous mode as a transmitter and receiver with even parity enabled, 2 stop bits, 8-bit character length, frequency 160 kHz and baud rate 10 K.

(a) Write an ALP to transmit 100 bytes of data string starting at location 2000:5000H.

(b) Write an ALP receive 100 bytes of data string and store it at 3000:4000H.

Solution

The interfacing connections of 8251A with 8086 are shown in Fig. 6.35.

Asynchronous mode control word for Problem 6.6 (a)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	1	1	1	1	1	0	= 0FE H
2 stop bits		Even parity enabled		8-bit format		CLK scaled by 16		

(a) ALP to initialize 8251 and transmit 100 bytes of data

```
ASSUME      CS : CODE
CODE        SEGMENT
START:      MOV AX, 2000H       ;
            MOV DS, AX          ; DS points to byte string segment
            MOV SI, 5000H       ; SI points to byte string
            MOV CL, 64H         ; length of the string in CL(hex)
            MOV AL, 0FEH        ; Mode control word out to
            OUT 0FEH, AL        ; D0-D7.
            MOV AX, 11H         ; Load command word
            OUT 0FEH, AL        ; to transmit enable and error reset
WAIT:       IN AL, 0FEH         ; Read status,
            AND AL, 01H         ; check transmitter enable
            JZ WAIT             ; bit,if zero wait for the transmitter to
                                  be ready
            MOV AL, [SI]        ; If ready, first byte of string data
            OUT 0FCH, AL        ; is transmitted.
            INC SI              ; Point to next byte.
            DEC CL              ; Decrement counter.
            JNZ WAIT            ; If CL is not zero, go for next byte.
            MOV AH, 4CH         ; If CX is zero, return to DOS
            INT 21H
CODE        ENDS
            END START
```

Program 6.8 *ALP to Transmit 100 Bytes of Data*

For Problem 6.6 (b), the command instruction word can be calculated as 14H.

(b) An ALP to initialize 8251 and receive 100 bytes of data.

```
ASSUME      CS : CODE
CODE        SEGMENT
START:      MOV AX, 3000H       ;
            MOV DS, AX          ; Data segment set to 3000H
            MOV SI, 4000H       ; Pointer to destination offset
            MOV CL, 64H         ; Byte count in CL
            MOV AL, 7EH         ; Only one stop bit for
            OUT 0FEH, AL        ; receiver is set
            MOV AL, 14H         ; Load command word to enable
            OUT 0FEH, AL        ; the receiver and disable
                                  transmitter
NXTBT:      IN AL, 0FEH         ; Read status
            AND 38H             ; Check FE, OE and PE,
            JZ READY            ; If zero, jump to READY
            MOV AL, 14H         ; If not zero, clear them
            OUT 0FEH, AL        ;
READY:      IN AL, 0FEH         ; Check RXRDY.If the
            AND 02H             ; receiver is not ready,
            JZ READY            ; wait
            IN AL, 0FCH         ; If it is ready,
            MOV [SI], AL        ; receive the character
            INC SI              ; Increment pointer to next byte
```

```
        DEC CL              ; Decrement counter
        JNZ NXTBT           ; Repeat,if CL is not zero
        MOV AH,4CH          ; If CL is 0, return to DOS
        INT 21H
CODE    ENDS
        END START
```

Program 6.9 *ALP to Receive 100 Bytes*

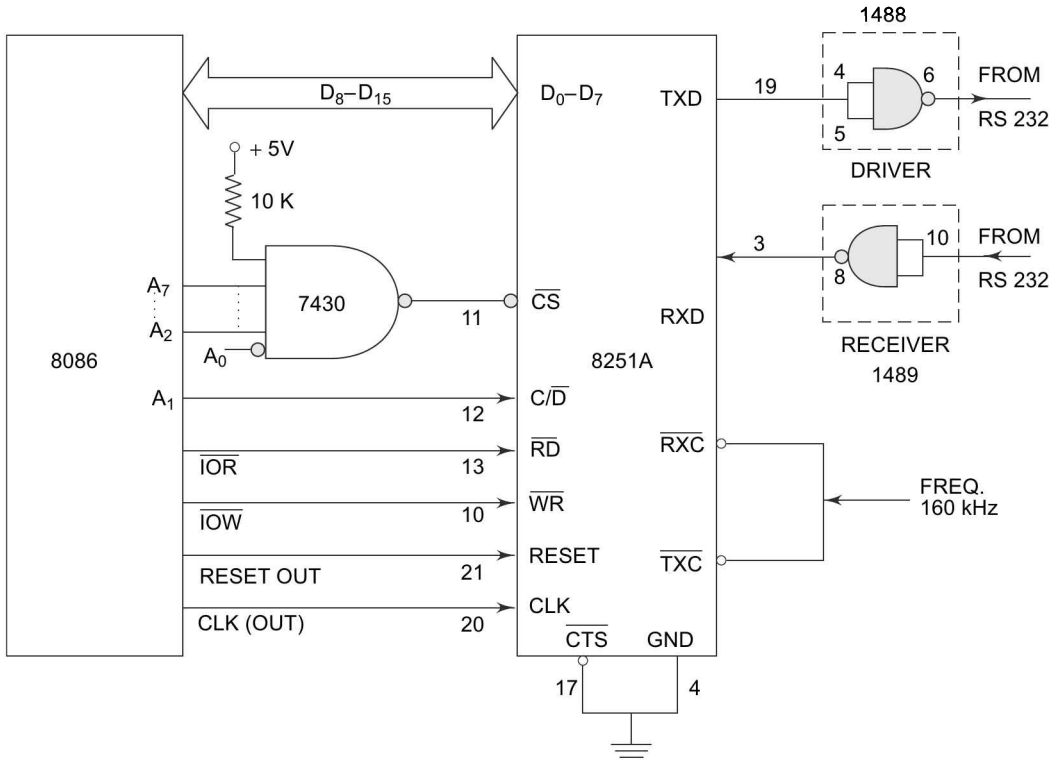


Fig. 6.35 *Interfacing of 8251A with 8086*

6.4.5 High-Speed Communication Standard; Universal Serial Bus (USB) Functional Description

Universal serial bus is the most popular serial communication standard introduced by USB Implementers Forum (USB-IF). RS-232c had been used for long and has a few serous disadvantages like limited speed of communication, high-voltage level signaling and big-size communication adapters. The USB standards are available with speeds from a few hundred kilobytes per second (USB1.x) to around 5 GB per second (USB3.x). The logic levels used for representing the bits use voltages less than 5 volts and currents less than 500 mA. A USB uses comparatively small size of connectors for establishing connection with compatible

devices. In fact, in addition to the standard USB connector available with PCs, many small-size versions are introduced specially for embedded products. Recent USB ports are used for many other activities like battery charging, communication with PC and even device-to-device communications.

USB implements an asynchronous serial communication. Initially, it used to be half-duplex. However, the new USB standard supports full-duplex communication at the cost of additional two conductors. The data is transmitted in the form of packets containing start, data, parity and stop bits. The attached devices are detected and configured automatically, provided they are USB compatible. USB standards provide automatic detection and correction of errors. Design and implementation of a USB compatible system requires design certification and licensing from the USB-IF.

In its original form, USB provides a serial bus standard for connecting peripherals such as mouse, keyboard, gamepads and joysticks, scanners, cameras, printers, external memory devices and other networking components. USB 1.0 was introduced in 1995 and 1996 followed by its advanced version, USB1.1, in 1998 for communication up to the speed of 1.5 MB/second. USB 2.0 was introduced in 2000 and supported data transfers up to 12 MB/second. USB 2.0 was revised in 2002 to provide three different speeds of communications up to 480 MB/second to support a wide variety of peripherals.

A USB communication system consists of three basic units: 1) Host, 2) Cable, and 3) Device. The host detects a connected device, and manages flow of control information and actual data between the system and the device. USB cable is a metallic medium of communication. It consists of four conductors.

- The VBUS conductor is a power supply pin and carries voltage from 4.2 V to 5.25 V.
- The Gnd connector extends the system ground to the device.
- There are two data pins D+ and D- for differential signaling of the logic levels.

A USB device monitors device address in each communication and prepares itself for communication if selected. It responds to all the requests from the host. It adds bits to the packets being set to the host for detecting errors. It detects and corrects errors in the data received from the host.

USB transfers are of four types.

- The interrupt USB transfer is meant for low-volume data transfer like keyboards, mouse and touch pads. Bulk data transfers are implemented in terms of block data transfers and are intended for devices requiring bigger data volumes like printers and scanners.
- Isochronous mode of transfers are meant for devices like speakers and microphones that require real-time synchronization for reproducing the original signal. These transfers do not require error detection and correction.
- The control transfers are used to configure and control the connected devices. The control transfers are handled as highest priority, automatic error protection and high data rate transfers.
- The USB asynchronous communication transfers four types of bit packets over the cable.
    - The first type of packet is called 'Handshake Packet'. The handshake packet consists of a packet identification, i.e., PID byte, and are used for reliable communication of data packets.
    - The second type of packet is called 'Token Packet'. The token packets are always sent by the host and contain 2 bytes including 11-bit address and 5 bits of cyclic redundancy code. The token packet commands the device either to receive data or transmit data in response to specific demands from the host.
    - Data packets contain actual bytes of data up to 1024 bytes. It also includes a 16-bit CRC code.

- A fourth type of packets called PRE packets are only of importance to low-speed USB devices. They contain a special PID value allotted to low-speed devices. High-speed devices neglect the PRE packets. However, the PRE values are used by network hubs while communicating with USB devices over a network. Over networks, the hubs control the data flow rate even to the high-speed devices subject to the instantaneous network traffic.

Signaling in USB Systems Available USB standards support four signaling rates.

- Low-speed USB 1.0 rate of 1.5 Mbps is suitable for human interface devices.
- Full-speed USB 1.0 rate of 12 Mbps is suitable for communication with network hubs using networks like LAN.
- High-speed USD 2.0 rate of 480 Mbps is suitable for higher speed devices but it is also compatible with full-speed devices with USB 1.0 standard.
- Super-speed USB 3.0 rates of up to 5.0 Gbps or 596 Mbps support full-duplex operation. However, it is backward compatible with the earlier USB standards though they require an additional pair of D+ and D– connectors for full-duplex communication.

USB signaling is implemented in differential for using 0 to 0.3 volts for logic 0 and 2.6 to 3.6 volts for logic 1 in low- and full-speed options. In higher speed modes, it uses –10 mV to +10 mV for logic 1 and 360 mV to 400 mV for logic 1. The host pulls down the D+ and D– lines using a 15 K resistor indicating no device is connected with the port. If a device is connected, it pulls one of the data lines high with a pull up of 1.5 K indicating that a device is connected with the port.

It must be noted that the D+ and D– conductors carry a valid bit only when their state is different and thus the name 'differential coding'. If both have zero state, it indicates end of the packet. To mark the next start of the packet, the D+ line must go high while D– remains low. Serial bus states are described in terms of a 'J' and a 'K' state. Duration of each state for full-speed communication in 1/12 MHz = 83 ns, i.e., bit-clock rate duration. In the 'J' state , the D+ line is high (2.6 to 3.6 V)and D– line is low (0 to 0.3 V) as shown in Fig. 6.36(a). In the 'K' state , the D+ line is low (0 to 0.3 V) and D– line is high (2.6 to 3.6 V) as shown in Fig. 6.36(b). A toggling between the two states represents a logical 0 bit. A repetition of either 'J' or 'K' states represents a logical 1 bit. The successive 'J' and 'K' states are shown in Figs 6.36 (c) and (d). A bit stream '0100100' is shown coded in Fig. 6.36(e). In the idle state, both lines are low. A starting of a data packet is marked by the D+ line going high while the D– line remains low. In the next bit period, the lines enter the 'K' state. The states toggle six times to mark a valid start of packet and then the 'K' state continues for one bit duration to mark start of a data bit stream as shown in Fig. 6.36(e). The next state is now 'J', i.e., there is a toggling, so the bit in the 'J' state is '0'. Further, there is again a 'J' state, i.e., a state continues, it represents a '1' bit. Thus, if there is a toggling between J to K or K to J, it represents a '0' bit and if there is a continuation of states, either J to J or K to K, it represents a '1' bit. A data packet may contain up to 1024 bits followed by a 16-bit CRC code. The end of the packet is marked by both the lines D+ and D– being pulled low for more than one-bit duration. If there are more than six successive '1' bits in the stream, a zero is inserted by the host or a USB compatible device after the sixth '1'. This is called bit stuffing. Thus, received seven consecutive '1' are considered an error. In case of an error, the same data packet may be requested again by the device using a special type of the handshake packet.

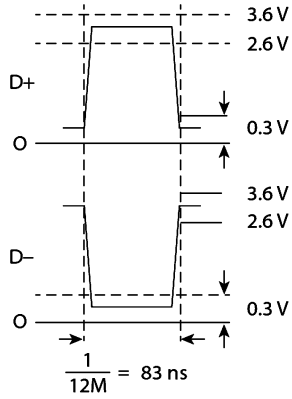


Fig. 6.36(a) *J State*

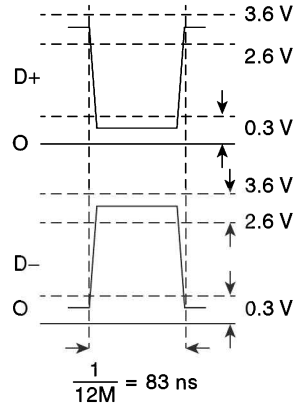


Fig. 6.36(b) *K State*

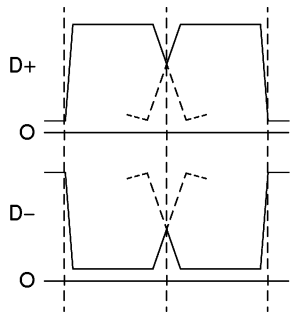


Fig. 6.36(c) *Successive J States*

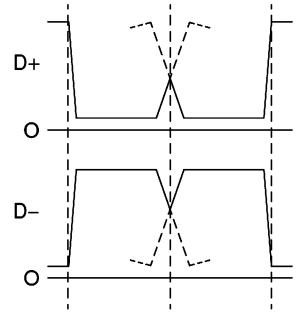


Fig. 6.36 (d) *Successive K States*

- Toggling between J & K states indicates a logics
- Successive states either J or K indicate logic 1

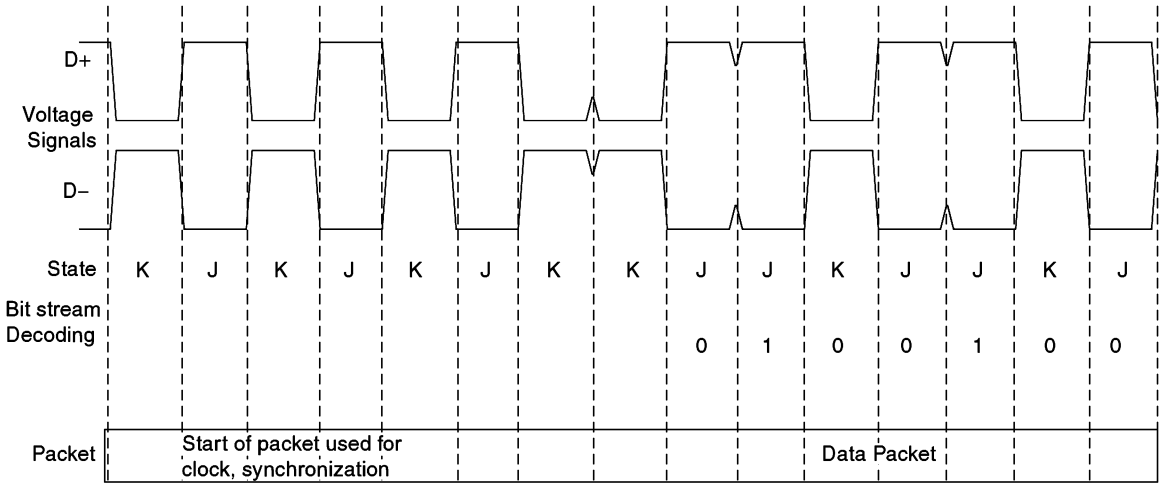
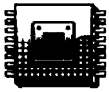


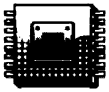
Fig. 6.36(e) *Coding of a Bit Stream Using the USB Signalling Scheme.*



SUMMARY

In this chapter, we have presented a detailed account of the functioning of some of the important Intel peripherals. With the recent advances in the field, the Intel family of the peripherals, has been continuously increasing. Those, which are frequently used in the industrial and general systems are discussed in this chapter, in significant details. This chapter has been started with the discussion on the programmable timer 8254. The necessary functional details of 8254 have been discussed along with an interfacing example and supporting programs. Further, the peripherals like programmable interrupt controller 8259A, programmable keyboard display controller 8279A, programmable communication interface 8251A have been discussed along with their architectures, signal descriptions, interfacing and programming examples.

Thus this chapter has provided an insight into the operations, programming and interfacing of the dedicated peripherals.



EXERCISES

6.1 Draw and discuss the internal architecture of 8254.

6.2 Draw and discuss the different modes of operation of 8254.

6.3 Explain the significances of different bits of the control word register format of 8254.

6.4 Design a real time clock using 8254 interfaced with 8086. The CLK input to the 8254 is of 1.5 MHz frequency. Assume suitable addresses for 8254. Further, display the time using a 6-digit 7-segment multiplexed display unit which is interfaced with the 8086 using 8255. The 7-segment data port address of 8255 is 00 and the digit select port address of 8255 is 01. Draw the hardware schematic and write a program.

6.5 Design an 8086 microprocessor based stop-watch using 8254 and 8255. The stop-watch counts up to 100 seconds in the steps of 10 ms and displays the time on a four-digit 7-segment multiplexed display. The CLK input frequency to 8254 is 2.4 MHz. Draw the required hardware scheme and write the required ALP. Select suitable addresses for 8254 and 8255.

6.6 A flow transducer, which generates number of TTL compatible pulses proportional to the volume of the liquid passed through it, is available. It generates a pulse, if 5 ml volume of the liquid passes through it. Design an 8086 based system using 8254 for measuring up to 1000 litres of the liquid at a precision of 10 ml. Assume the required addresses suitably.

6.7 Draw and discuss the internal architecture of 8259A.

6.8 Explain the functions of the following pins of 8259A.
(i) CAS_0–CAS_2 (ii) $\overline{SP}/\overline{EN}$

6.9 Describe an interrupt request response of an 8086 system.

6.10 What is the difference between 8259 and 8259A?

6.11 Explain the initialisation sequence of 8259A.

6.12 How will you provide more than eight interrupt input lines to an 8086 based system?
Design an interrupt system which provides twenty nine interrupt inputs to the 8086 system.

6.13 Explain the following terms in relation to 8259A.
(i) EOI (ii) Automatic rotation

(iii) Automatic EOI (iv) Specific rotation
(v) Special mask mode (vi) Edge and level triggered mode
(vii) Cascading (viii) Special fully nested mode
(ix) Buffered mode (x) Polling

6.14 Show interfacing schematic for connecting 13 interrupting devices to 8086 using 8259. Connect the slave 8259 at IR4 of the master 8259. The master should use the vectors 10H to 17H and the slave should use the vectors 30H to 34H. The master and slave PICs are selected at addresses 90H and 94H respectively. Write an ALP to initialise the 8259s in fixed priority, level triggered, normal EOI and special mask mode.

6.15 How does 8259A differentiates between an 8-bit and 16-bit processors?

6.16 How do you interface 8259A with 8086 in maximum mode? Draw the schematic.

6.17 Elaborate the need of a dedicated keyboard display controller.

6.18 Draw and discuss architecture of 8279.

6.19 Explain the functions of the following signals of 8279.

(i) IRQ (ii) SL_0–SL_3
(iii) RL_0-RL_7 (iv) SHIFT
(v) CNTL/STB (vi) $\overline{BD}$
(vii) $OUTA_0$–$OUTA_3$ and $OUTB_0$–$OUTB_3$

6.20 Will it be possible to interface more than sixteen 7-segment display units using 8279? If yes, explain how.

6.21 What is the sensor matrix mode of 8279? Explain the function of the 8×8-bit RAM in this mode.

6.22 Explain the following terms in relation to 8279.

(i) Two key lock out (ii) N-key roll-over
(iii) Right entry (iv) Left entry
(v) FIFO (vi) Display RAM

6.23 Explain the different commands of 8279 in brief.

6.24 Explain the key-code format of 8279.

6.25 Explain the FIFO status word of 8279.

6.26 Explain the mode set register of 8279.

6.27 Draw the schematic of an 8279 keyboard controller interfaced to 8086. An 11 key keyboard and an 8-digit 7-segment display is to be driven by the system so that by reading the FIFO we should directly get the number of the key pressed (the number corresponding to the key, i.e. 0 to 9 must be same as the keycode formed by 8279). The first 10 keys are allotted to the numbers 0 to 9 and the eleventh key is a 'CLEAR', which should clear the contents of the display RAM. Program the 8279 in left entry, 2-key lock out mode.

6.28 Design an 8086 and 8279 based system to interface sixteen 7-segment display units using any four port lines of 8279 $OUTA_0$–$OUTA_3$. You may use any additional hardware if required. Write a program to display the hex numbers 0 to FH on these sixteen displays using right entry mode. For example, 0 will be displayed at the LSB position of the display for half second , then 0 will be shifted to the next left display and 1 will be displayed at the LSB position for half second and so on. After all the numbers up to FH are displayed, the display should be blanked by glowing all the segments of all the 7-segment units for half second and then the same procedure should be repeated continuously.

6.29 Interface a 26-keys keyboard with 8279. The keys represent the alphabets 'a' to 'z'. Write an 8086 ALP to find out the ASCII equivalent of the alphabet corresponding to the pressed key. The 8086 system runs at 6 MHz while the 8279 should work at 200 kHz. Will the internal prescalar reduce 6 MHz to 200 kHz? If any external prescalar is required, design it with minimum hardware.

6.30 Draw and discuss internal architecture of USART 8251.

6.31 Explain the following signal descriptions of 8251.

(i) $C/\overline{D}$ (ii) $\overline{TXC}$ (iii) TXD
(iv) $\overline{RXC}$ (v) RXD (vi) RXRDY
(vii) TXRDY (viii) $\overline{DSR}$ (ix) $\overline{DTR}$
(x) $\overline{RTS}$ (xi) $\overline{CTS}$ (xii) TXEMPTY
(xiii) SYNDET/BD

6.32 Explain the mode instruction control word format of 8251.

6.33 Draw and discuss the asynchronous mode transmitter and receiver data formats of 8251.

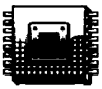
6.34 Draw and discuss the status word format of 8251.

6.35 Draw and discuss the synchronous mode transmit and receive data formats of 8251.

6.36 Interface 8251 with 8086 at an address 80H. Initialise it in asynchronous transmit mode, with 7-bits character size, baud factor 16, one start bit, one stop bit, even parity enabled. Further transmit a message 'HAPPY NEW YEAR' in ASCII coded form to a modem.

6.37 Write a program to initialise 8251 in synchronous mode with even parity, single SYNCH character, 7-bit data character. Then receive FFH bytes of data from a remote terminal and store it in the memory at address 5000H:2000H.

7 DMA, and High Storage Capacity Memory Devices



INTRODUCTION

In the previous chapter, we studied some of the dedicated peripherals and their interfacing techniques with 8086. In this chapter, we will further discuss a few advanced peripherals and their interfacing techniques with 8086. In the applications where the CPU is to transfer bulk data, it may be a waste of time to transfer the data from source to destination using program controlled data transfer or interrupt driven data transfer. The alternate way of transferring the bulk data is the Direct Memory Access (DMA) technique in which the data is transferred under the control of a DMA controller, after it is properly initialised by the CPU. A DMA controller is designed to complete the bulk data transfer task much faster than the CPU. One such application which involves bulk data transfer is the storage of programs or data into secondary memories. At the end we have presented a brief overview of high capacity memory devices like old days floppy, Compact disk, Digital video disk and Hard disk drive.

7.1 DMA CONTROLLER 8257

The *Direct Memory Access* or DMA mode of data transfer is the fastest amongst all the modes of data transfer. In this mode, the device may transfer data directly to/from memory without any interference from the CPU. The device requests the CPU (through a DMA controller) to hold its data, address and control bus, so that the device may transfer data directly to/from memory. The DMA data transfer is initiated only after receiving HLDA signal from the CPU. For facilitating DMA type of data transfer between several devices, a DMA controller may be used.

Intel's 8257 is a four channel DMA controller designed to be interfaced with their family of microprocessors. The 8257, on behalf of the devices, requests the CPU for bus access using local bus request input i.e. HOLD in minimum mode. In maximum mode of the microprocessor $\overline{RQ}/GT$ pin is used as bus request input. On receiving the HLDA signal (in minimum mode) or $\overline{RQ}/\overline{GT}$ signal (in maximum mode) from the CPU, the requesting device gets the access of the bus, and it completes the required number of DMA cycles for the data transfer and then hands over the control of the bus back to the CPU.

7.1.1 Internal Architecture of 8257

The internal architecture of 8257 is depicted in Fig. 7.1. The chip supports four DMA channels, i.e. four peripheral devices can independently request for DMA data transfer through these channels at a time. The DMA controller has 8-bit internal data buffer, a read/write unit, a control unit, a priority resolving unit along with a set of registers. We will discuss each one of them in the following sections. Next, we describe the register organisation of 8257.

Register Organisation of 8257 The 8257 performs the DMA operation over four independent DMA channels. Each of the four channels of 8257 has a pair of two 16-bit registers, viz. DMA *address register* and *terminal count register*. Also, there are two common registers for all the channels, namely, *mode set register* and *status register*. Thus there are a total of ten registers. The CPU selects one of these ten registers using address lines A_0–A_3. Table 7.1 shows how the A_0–A_3 bits may be used for selecting one of these registers. We will now describe each register as follows:

DMA Address Registers Each DMA channel has one DMA address register. The function of this register is to store the address of the starting memory location, which will be accessed by the DMA channel. Thus the starting address of the memory block which will be accessed by the device is first loaded in the DMA address register of the channel. Naturally, the device that wants to transfer data over a DMA channel, will access the block of memory with the starting address stored in the DMA Address Register.

Terminal Count Register As in the previous case, each of the four DMA channels of 8257 has one terminal count register (TC). This 16-bit register is used for ascertaining that the data transfer through a DMA channel ceases or stops after the required number of DMA cycles. Thus this register should be appropriately written before the actual DMA operation starts. The low order 14-bits of the terminal count register are initialised with the binary equivalent of the number of required DMA cycles minus one. After each DMA cycle, the terminal count register content will be decremented by one and finally it becomes zero after the required number of DMA cycles are over.

The bits 14 and 15 of this register indicate the type of the DMA operation (transfer). If the device wants to write data into the memory, the DMA operation is called DMA write operation. Bit 14 of the register in this case will be set to one and bit 15 will be set to zero. Table 7.2 gives details of DMA operation selection and the corresponding bit configuration of the bits 14 and 15 of the TC register.

Mode Set Register The mode set register is used for programming the 8257 as per the requirements of the system. The function of the mode set register is to enable the DMA channels individually and also to set the various modes of operation. A DMA channel should not be enabled till the DMA address register and the terminal count register contain valid information, otherwise, an unwanted DMA request may initiate a DMA cycle, probably destroying the valid memory data.

Table 7.1 *8257 Register Selection*

Register	*Byte*	*Address Inputs*			*F/L*		*BI-Directional Data Bus*							
		A_3	A_2	A_1	A_0		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CH-0 DMA Address	LSB	0	0	0	0	0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	MSB	0	0	0	0	1	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8
CH-0 Terminal Count	LSB	0	0	0	1	0	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

(*Contd.*)

Table 7.1 *(Contd.)*

Register	*Byte*	*Address Inputs*			*F/L*	*BI-Directional Data Bus*							
		A_3	A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	MSB	0	0	0	1	1	Rd	Wr	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8
CH-1 DMA Address	LSB	0	0	1	0	0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	MSB	0	0	1	0	1	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8
CH-1 Terminal Count	LSB	0	0	1	1	0	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
	MSB	0	0	1	1	1	Rd	Wr	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8
CH-2 DMA Address	LSB	0	1	0	0	0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	MSB	0	1	0	0	1	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8
CH-2 Terminal Count	LSB	0	1	0	1	0	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
	MSB	0	1	0	1	1	Rd	Wr	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8
CH-3 DMA Address	LSB	0	1	1	0	0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	MSB	0	1	1	0	1	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8
CH-3 Terminal Count	LSB	0	1	1	1	0	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
	MSB	0	1	1	1	1	Rd	Wr	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8
MODE SET	—	1	0	0	0	0	AL	TCS	EW	RP	EN3	EN2	EN1	EN0
(Programme only)														
STATUS (Read only)	—	1	0	0	0	0	0	0	0	UP	TC3	TC2	TC1	TC0

A_{10}–A_{15} DMA Starting Address, C_0–C_{13} Terminal Count Value (N-1), Rd & Wr-DMA verify (00), Write (01) or Read (10) cycle selection. AL Auto Load, TCS-TC STOP, EW-Extended Write, RP-Rotating Priority, EN3-EN0-Channel Mask Enable, UP-Update Flag, TC3-TC0-Terminal Count Status Bits.

The mode set register format is shown in Fig. 7.2. It is thus extremely important that the mode set register should be programmed by the CPU for enabling the DMA channels only after initializing the DMA address register and terminal count register appropriately.

Table 7.2 *DMA Operation Selection Using A_{15}/RD and A_{14}/WR*

Bit 15	*Bit 14*	*Type of DMA Operation*
0	0	Verify DMA Cycle
0	1	Write DMA Cycle
1	0	Read DMA Cycle
1	1	(Illegal)

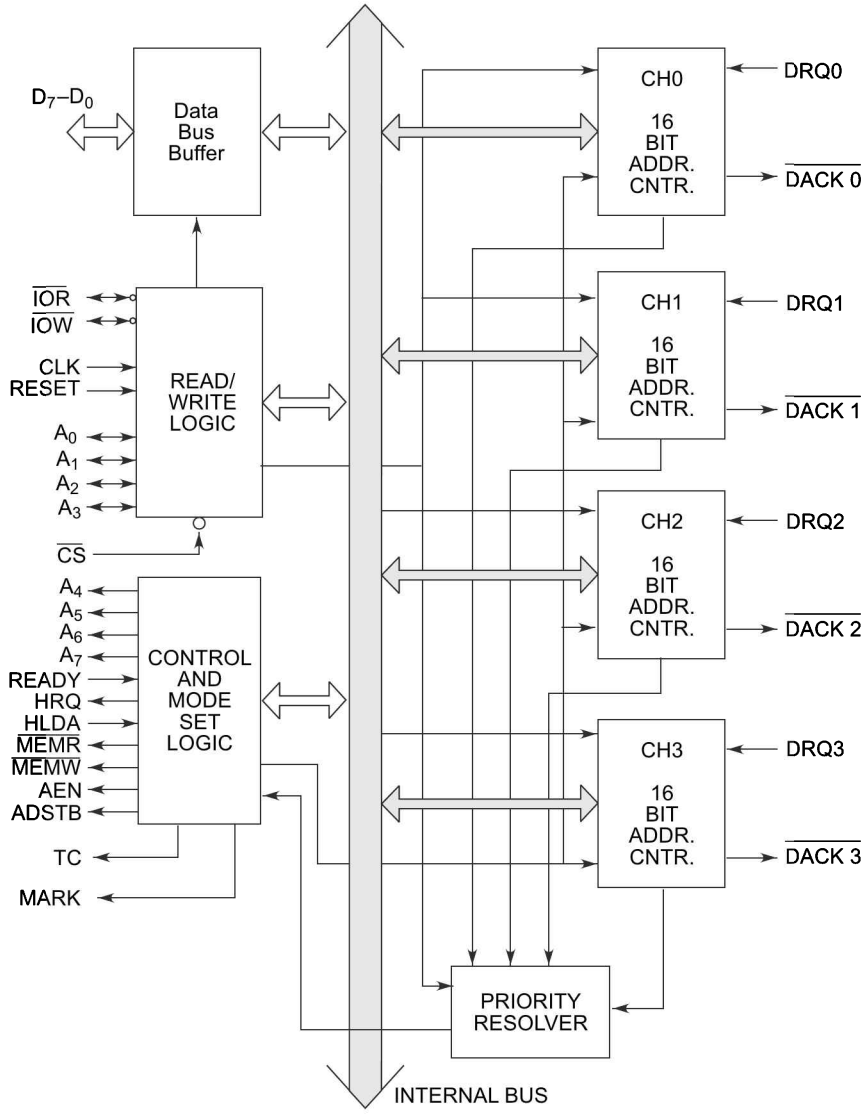


Fig. 7.1 *Internal Architecture of 8257*

The bits D_0–D_3 enable one of the four DMA channels of 8257. For example, if D_0 is '1', channel 0 is enabled. If bit D_4 is set, rotating priority is enabled, otherwise, the normal, i.e. fixed priority is enabled. The normal and rotating priorities will be explained later in this text.

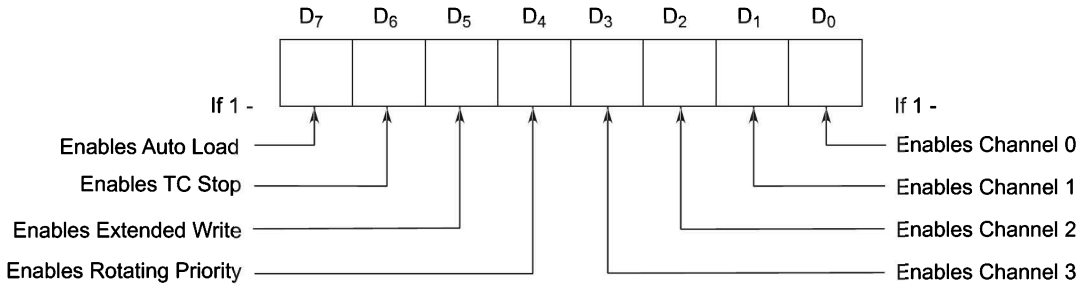


Fig. 7.2 *Bit Definitions of the Mode Set Register*

If the TC STOP bit is set, the selected channel is disabled after the *terminal count* condition is reached, and it further prevents any DMA cycle on the channel. To enable the channel again, this bit must be reprogrammed. If the TC STOP bit is programmed to be zero, the channel is not disabled, even after the count reaches zero and further requests are allowed on the same channel.

The auto load bit, if set, enables channel 2 for the repeat block chaining operations, without immediate sofware intervention between the two successive blocks. The channel 2 registers are used as usual, while the channel 3 registers are used to store the block reinitialisation parameters, i.e. the DMA starting address and the terminal count. After the first block is transferred using DMA, the channel 2 registers are reloaded with the corresponding channel 3 registers for the next block transfer, if the *Update* flag is set.

The extended write bit, if set to '1', extends the duration of $\overline{\text{MEMW}}$ and $\overline{\text{IOW}}$ signals by activating them earlier. This is useful in interfacing the peripherals with different access times. If the peripheral is not accessed within the stipulated time, it is expected to give the 'NOT READY' indication to 8257, to request it to add one or more wait states in the DMA cycle. The mode set register can only be written into.

Status Register The status register of 8257 is shown in Fig. 7.3. The lower order 4-bits of this register contain the terminal count status for the four individual channels. If any of these bits is set, it indicates that the specific channel has reached the terminal count condition. These bits remain set till either the status is read by the CPU or the 8257 is reset. The update flag is not affected by the read operation. This flag can only be cleared by resetting 8257 or by resetting the auto load bit of the mode set register. If the update flag is set, the contents of the channel 3 registers are reloaded to the corresponding registers of

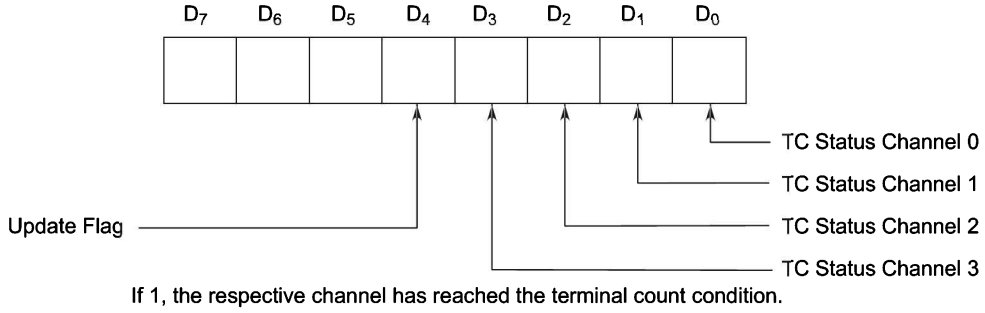


Fig. 7.3 *Bit Definitions of Status Register of 8257*

channel 2, whenever the channel 2 reaches a terminal count condition, after transferring one block and the next block is to be transferred using the autoload feature of 8257. The update flag is set every time, the channel 2 registers are loaded with contents of the channel 3 registers. It is cleared by the completion of the first DMA cycle of the new block. This register can only be read.

7.1.2 Data Bus Buffer, Read/Write Logic, Control Unit and Priority Resolver

The 8-bit, tristate, bidirectional buffer interfaces the internal bus of 8257 with the external system bus under the control of various control signals. In the slave mode, the read/write logic accepts the I/O Read or I/O Write signals, decodes the A_0–A_3 lines and either writes the contents of the data bus to the addressed internal register or reads the contents of the selected register depending upon whether $\overline{IOW}$ or $\overline{IOR}$ signal is activated. In master mode, the read/write logic generates the $\overline{IOR}$ and $\overline{IOW}$ signals to control the data flow to or from the selected peripheral. The control logic controls the sequences of operations and generates the required control signals like AEN, ADSTB, $\overline{MEMR}$, $\overline{MEMW}$, TC and MARK along with the address lines A_4–A_7, in master mode. The priority resolver resolves the priority of the four DMA channels depending upon whether normal priority or rotating priority is programmed.

7.1.3 Signal Descriptions of 8257

Figure 7.4 shows pin configuration of 8257, followed by the functional description of each signal.

DRQ_0–DRQ_3 These are the four individual channel DMA request inputs, used by the peripheral devices for requesting DMA services. The DRQ_0 has the highest priority while DRQ_3 has the lowest one, if the fixed priority mode is selected.

$\overline{DACK_0}$ – $\overline{DACK_3}$ These are the active-low DMA acknowledge output lines which inform the requesting peripheral that the request has been honoured and the bus is relinquished by the CPU. These lines may act as strobe lines for the requesting devices.

D_0–D_7 These are bidirectional, data lines used to interface the system bus with the internal data bus of 8257. These lines carry command words to 8257 and status word from 8257, in slave mode, i.e. under the control of CPU. The data over these lines may be transferred in both the directions. When the 8257 is the bus master (master mode, i.e. not under CPU control), it uses D_0–D_7 lines to send higher byte of the generated address to the latch. This address is further latched using ADSTB signal. The address is transferred over D_0–D_7 during the first clock cycle of the DMA cycle. During the rest of the period, data is available on the data bus.

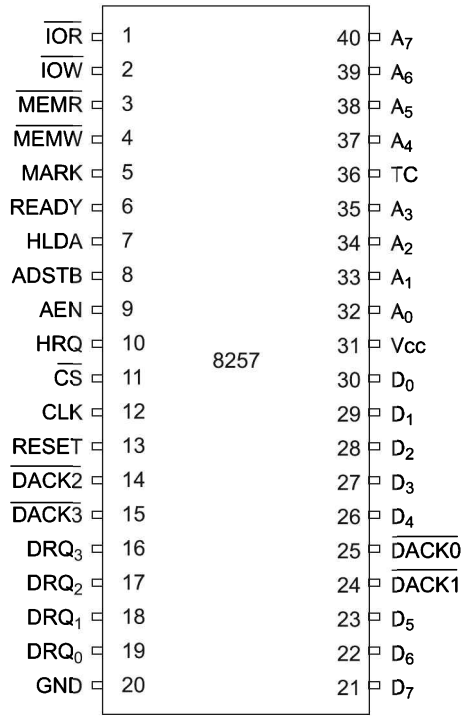


Fig. 7.4 *Pin Diagram of 8257*

$\overline{IOR}$ This is an active-low bidirectional tristate input line that acts as an input in the slave mode. In slave mode, this input signal is used by the CPU to read internal registers of 8257. This line acts as output in master mode. In master mode, this signal is used to read data from a peripheral during a memory write cycle.

$\overline{IOW}$ This is an active-low, bidirectional tristate line that acts as input in slave mode to load the contents of the data bus to the 8-bit mode register or upper/lower byte of a 16-bit DMA address register or terminal count register. In the master mode, it is a control output that loads the data to a peripheral during DMA memory read cycle (write to peripheral).

CLK This is a clock frequency input required to derive basic system timings for the internal operation of 8257.

RESET This active-high asynchronous input disables all the DMA channels by clearing the mode register and tristates all the control lines.

A_0–A_3 These are the four least significant address lines. In slave mode, they act as input which select one of the registers to be read or written. In the master mode, they are the four least significant memory address output lines generated by 8257.

$\overline{CS}$ This is an active-low chip select line that enables the read/write operations from/to 8257, in slave mode. In the master mode, it is automatically disabled to prevent the chip from getting selected (by CPU) while performing the DMA operation.

A_4–A_7 This is the higher nibble of the lower byte address generated by 8257 during the master mode of DMA operation.

READY This is an active-high asynchronous input used to stretch memory read and write cycles of 8257 by inserting wait states. This is used while interfacing slower peripherals.

HRQ The hold request output requests the access of the system bus. In the non-cascaded 8257 systems, this is connected with HOLD pin of CPU. In the cascade mode, this pin of a slave is connected with a DRQ input line of the master 8257, while that of the master is connected with HOLD input of the CPU.

HLDA The CPU drives this input to the DMA controller high, while granting the bus to the device. This pin is connected to the HLDA output of the CPU. This input, if high, indicates to the DMA controller that the bus has been granted to the requesting peripheral by the CPU.

$\overline{MEMR}$ This active-low memory read output is used to read data from the addressed memory locations during DMA read cycles.

$\overline{MEMW}$ This active-low three state output is used to write data to the addressed memory location during DMA write operation.

ADSTB This output from 8257 strobes the higher byte of the memory address generated by the DMA controller into the latches.

AEN This output is used to disable the system data bus and the control the bus driven by the CPU. This may be used to disable the system address and data bus by using the enable input of the bus drivers to inhibit the non-DMA devices from responding during DMA operations. This also may be used to transfer the higher byte of the generated address over the data bus. If the 8257 is I/O mapped, this should be used to disable the other I/O devices, when the DMA controller address is on the address bus.

TC Terminal count output indicates to the currently selected peripheral that the present DMA cycle is the last for the previously programmed data block. If the TC STOP bit in the mode set register is set, the selected channel will be disabled at the end of the DMA cycle. The TC pin is activated when the 14-bit content of the terminal count register of the selected channel becomes equal to zero. The lower order 14 bits of the terminal count register are to be programmed with a 14-bit equivalent of (n-1), if n is the desired number of DMA cycles.

MARK The modulo 128 mark output indicates to the selected peripheral that the current DMA cycle is the 128th cycle since the previous MARK output. The mark will be activated after each 128 cycles or integral multiples of it from the beginning of the data block (the first DMA cycle), if the total number of the required DMA cycles (n) is completely divisible by 128.

V_{cc} This is a +5V supply pin required for operation of the circuit.

GND This is a return line for the supply (ground pin of the IC).

7.2 DMA TRANSFERS AND OPERATIONS

The 8257 is able to accomplish three types of operations, viz. verify DMA operation, write operation and read operation. The complete operational sequence of 8257 is described using a state diagram in Fig. 7.5 for a single channel.

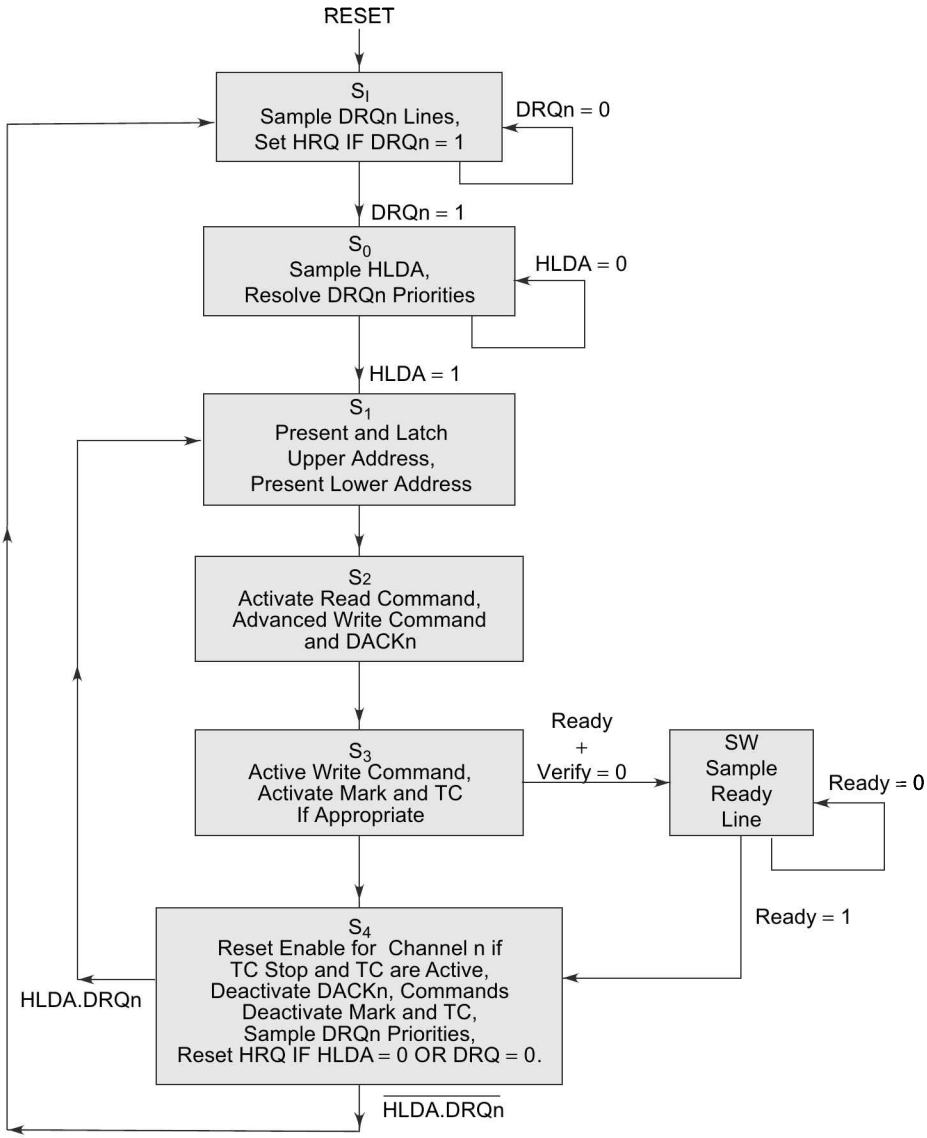
A single byte transfer using 8257 may be requested by an I/O device using any one of the 8257 DRQ inputs. In response, the 8257 sends HRQ signal to the CPU at its HLD input and waits for acknowledgement at the HLDA input. If the HLDA signal is received by the DMA controller, it indicates that the bus is available for the transfer. The $\overline{\text{DACK}}$ line of the used channel is pulled down by the DMA controller to indicate the I/O device that its request for the DMA transfer has been honoured by the CPU. The DMA controller generates the read and write commands to transfer the byte from/to the I/O device. The $\overline{\text{DACK}}$ line is pulled transfer is over, to indicate the DMA controller that the transfer, as requested by the device, is over. The HRQ line is lowered by the DMA controller to indicate the CPU that it may regain the control of the bus. The DRQ must be high until acknowledged and must go low before S_4 state of the DMA operation state diagram to avoid another unwanted transfer.

If more than one channel requests service simultaneously, the transfer will occur as a *burst transfer*. This will be discussed further in case of 8237. No overhead is required in switching from one channel to another. In each S_4, the DRQ lines are sampled and the highest priority request is recognized during the next transfer. Once the higher priority transfer is over; the lower priority transfer requests may be served, provided their DRQ lines are still active. The HRQ line is maintained active till all the DRQ lines go low.

The burst or continuous transfer, described above may be interrupted by an external device by pulling down the HLDA line. After each transfer, the 8257 checks the HLDA line. If it is found active, it completes the current transfer and releases the HRQ line (i.e. sends it low) and returns to its idle state. If the DRQ line is still active, the 8257 will again activate HRQ and proceed as already described. The 8257 uses four clock cycles to complete a transfer. The 8257 has a READY input to interface it with low speed devices. The READY pin status is checked in S_3 of the state diagram. If it is low, the 8257 enters a wait state. The status is sampled in every state till it goes high. Once the READY pin goes high, the 8257 continues from state S_4 to complete the transfer. The 8257 can be interfaced as a memory mapped device or an I/O mapped device. If it is connected as memory mapped device, proper care must be taken while programming Rd/A_{15} and Wr/A_{14} bits in the terminal count register.

7.2.1 Priorities of the DMA Requests

The 8257 can be programmed to select any of the two priority schemes using the command register. The first is the *fixed priority scheme*, while the second is the *rotating priority scheme*. In the fixed priority scheme, each device connected to a channel is assigned a fixed priority. In this scheme, the $DREQ_3$ has the lowest priority followed by DRQ_2 and DRQ_1. DRQ_0 has the highest priority. In the rotating priority mode, the priorities assigned to the channels are not fixed. At any point of time, suppose DRQ_0 has highest priority and DRQ_3 the lowest, then after the device at channel 0 gets the service, its priority goes down and the channel 0 becomes the lowest priority channel. Channel1 now becomes the highest priority channel, and remains the highest priority channel till it gets the service. Once channel 1 is served, it becomes the lowest priority channel and the channel 2 now becomes the highest priority channel. If you select the rotating priority, in a single chip DMA system any device requesting the service is guaranteed to be recognized after no more than three higher priority requests, thus avoiding dominance of any one channel. The priority allotment in the rotating priority mode is as shown in Fig. 7.6.



DRQn refers to anv DRQ line of an enabled DMA channel

Fig. 7.5 *DMA Operation State Diagram*

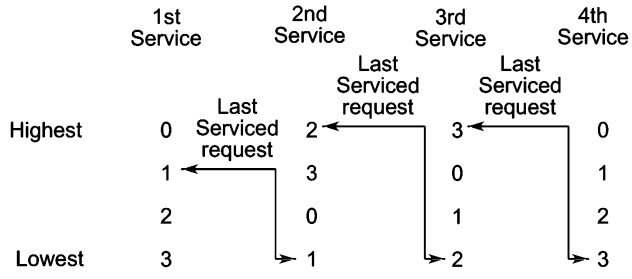


Fig. 7.6 *Priority Allotment in Rotating Priority Mode*

7.2.2 Programming and Reading the 8257 Registers

The selected register may be read or written depending upon the instruction executed by the CPU but the mode set register can only be written in, while the status register can only be read. The 16-bit register pair of each channel is read or written in two successive read or write operations. The Least Significant Byte (LSB) and the Most Significant Byte (MSB) of each register for a specific channel has the same address, but they are differentiated by an internal First/Last (F/L) flip-flop. If the F/L flip-flop is 0, it indicates the first operation, i.e. the LSB is to be read or written, otherwise, it is the last operation, i.e. the MSB is to be read or written. The F/L flip-flop can be cleared by resetting 8257. Thus the first operation after RESET will always be a LSB operation and the successive one for the same register address will be a MSB operation. The least significant three address bits A_0–A_2 indicate the specific register for a specific channel. The A_3 address line is used to differentiate between all the channel registers and the common registers, i.e. mode set and status registers. The higher order address lines A_4–A_{15} may be used to derive the chip select signal $\overline{CS}$ of 8257. All the accesses to any of the terminal count registers and DMA address registers must be in pairs, i.e. the LSB accesses must be followed by the MSB accesses. In verify transfer mode, no actual data transfer takes place. In this mode, the 8257 acts in the same way as read or write transfer to generate addresses, but no control lines are activated.

7.2.3 Interfacing 8257 with 8086

Once a DMA controller is initialised by a CPU properly, it is ready to take control of the system bus on a DMA request, either from a peripheral or itself (in case of memory-to-memory transfers). The DMA controller sends a HOLD request to the CPU and waits for the CPU to assert the HLDA signal. The CPU relinquishes the control of the bus before asserting the HLDA signal. Once the HLDA signal goes high, the DMA controller activates the $\overline{DACK}$ signal to the requesting peripheral and gains the control of the system bus. The DMA controller is the sole master of the bus, till the DMA operation is over. The CPU remains in the HOLD status (all of its signals are tristated except HOLD and HLDA), till the DMA controller is the master of the bus. In other words, the DMA controller interfacing circuit implements a switching arrangement for the address, data and control busses of the memory and peripheral subsystem from/to the CPU to/from the DMA controller. A conceptual implementation of the system is shown in Fig. 7.7(a).

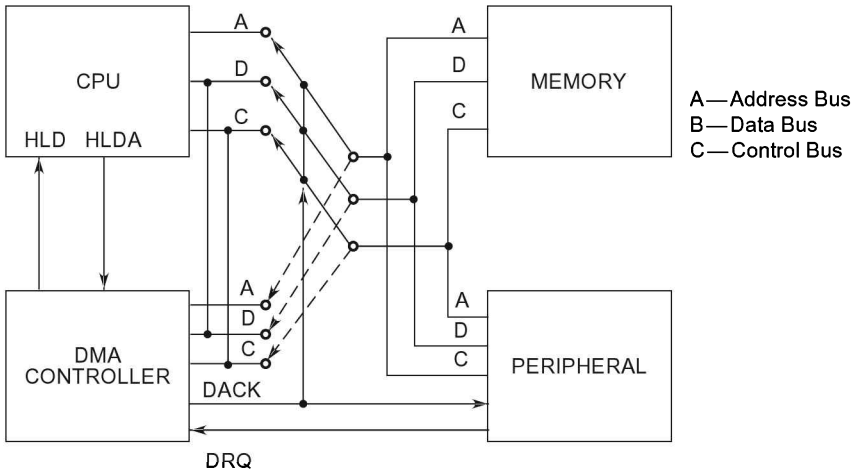


Fig. 7.7(a) *Interfacing a Typical DMA Controller with a System*

To explain the interfacing of 8257 with 8086 let us consider an interfacing example.

Problem 7.1

Interface DMA controller 8257 with 8086 so that the channel 0 DMA address register has an I/O address 80H and the mode set register has an address 88H. Initialize the 8257 with normal priority, TC stop and non-extended write. Autoload is not required. The transfer is to take place using channel 0. Write an ALP to move 2KB of data from a peripheral device to memory address 2000:5000H, with the above initialisation.

Solution Figure 7.7(b) shows interfacing connections of DMA controller 8257 with 8086.
As the DMA controller can generate only 16-bit address, while the CPU generates 20-bit address, the four upper address bits are generated and latched on the bus externally using a latch 8212. The 8086 uses three more 8212 latches and two 74245 buffers to demultiplex the address and data buses. These latches are controlled using the AEN signal of the DMA controller. If the DMA controller is in master mode, these will be automatically disabled and if the DMA controller is in slave mode, i.e. the buses are in the control of the CPU, these latches are enabled, using the $\overline{DS_1}$ signal. The data bus D_0–D_{15} is the general system data bus while the bus DD_0–DD_7 is a data bus to be used by an 8-bit peripheral that works under the control of the DMA controller. The 8-bit data bus DD_0–DD_7 is generated from the system data bus D_0–D_{15} using two additional data buffers which enable the 8-bit peripheral to access the even as well as odd addressed memory locations over the 8-bit data bus DD_0–DD_7. The $\overline{MEMRD}$ and $\overline{MEMWD}$ signals decide the direction of data flow under the control of the DMA controller. The A_0 and $\overline{DACK_0}$ signals enable the two buffers only for the DMA controlled data transfer on channel 0. The DMA controller requires an additional latch to demultiplex the address bus A_8–A_{15} from the data bus D_0–D_7, generated by it. This latch is enabled by the ADSTB signal generated by the DMA controller. An additional 74245 is used to read and write the DMA controller registers under the control of the CPU. The inverted clock delays the DMA controller operation as compared to the CPU. Note that the upper data bus D_8–D_{15} is used for initialising the DMA controller in the slave mode. Also note the corresponding 16-bit instructions used to initialise the 8-bit peripheral using the upper 8-bit data bus D_8–D_{15}. Note the circuit arrangements made for accessing the even as well as odd addresses using D_0–D_7. All the initialisation command words should be derived before writing the program.

[redacted] As per the problem specification, we need the following: enable TC stop, enable channel 0, disable auto-load, Disable extended-write, disable rotating priority, disable all other channels. As already discussed, the individual bits of the mode set register are set or reset as shown below.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	0	0	0	0	0	1	= 41 H

[redacted] The DMA address register contains the starting address of the memory block which is to be accessed using DMA, i.e. 5000H.

[redacted] As has been mentioned in Section 7.1.1, the last significant 14-bits of this register will contain the binary equivalent of the required number of the DMA cycles, i.e. number of bytes to be transferred minus one. As per the requirement, 2Kbytes of data have to be transferred from the device to memory. Therefore, the low order 14-bits of TC register will contain 7FFH. Moreover, the DMA operation in this case is going to be a memory write operation, hence A_{15} and A_{14} of this register should be 0 and 1. The TC register contents for these specifications are shown below.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	= 47FFH

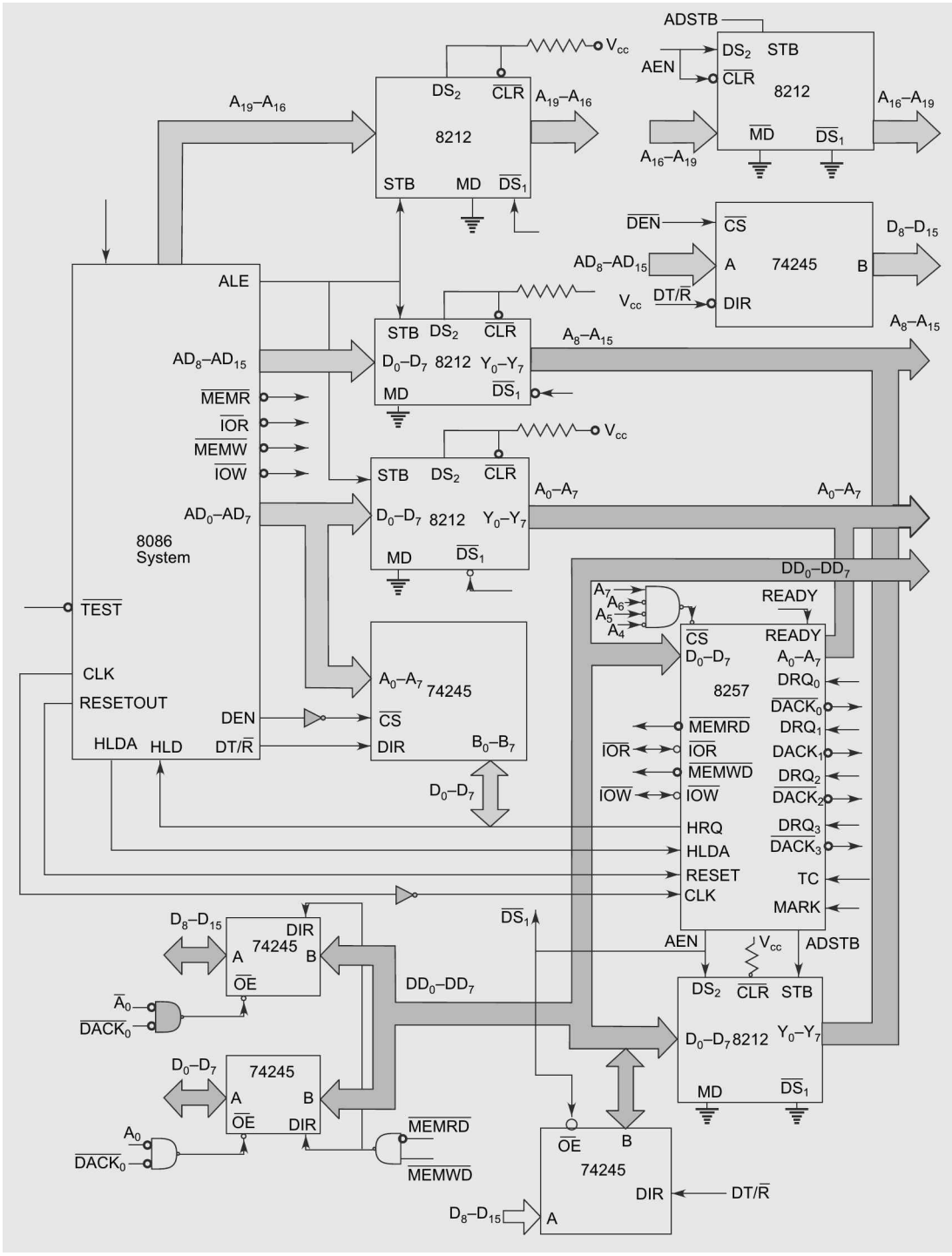


Fig. 7.7(b) *Interfacing 8257 with 8086*

The ALP for Problem 7.1 is given as follows.

```
ASSUME      CS:CODE,DS:DATA
CODE        SEGMENT
START:      MOV AX, DATA          ; Initialize Data Segment
            MOV DS, AX            ;
            MOV AX, DMAL          ; Load DMA address register with
            OUT 80H, AX           ; lower byte of DMA address
            MOV AX, DMAH          ; Load higher byte of DMA address
            OUT 80H, AX           ; register of Channel 0
            MOV AX, TCL           ; Load lower byte TC register of
            OUT 81H, AX           ; channel 0
            MOV AX, TCH           ; Load higher byte of TC register
            OUT 81H, AX           ;
            MOV AX, MSR           ; MODE SET Register
                                  ; Initialization, The F/L flip-flop
            OUT 88H, AX           ; is assumed to be cleared
            MOV AH, 4CH           ; Latch segment address on A16-A19
            INT 21H               ; externally, i.e. 0010(2H) and wait
                                    for the
                                  ; DMA request
CODE ENDS                         ; After the request is served, the
                                  ; CPU may continue the execution
DATA SEGMENT
MSR EQU 4141H                     ; Mode Set Register content
DMAL EQU 0000H                    ; DMA Address lower byte
DMAH EQU 5050H                    ; DMA Address higher byte
TCL EQU FFFFH                     ; Terminal count lower byte
TCH EQU 4747H                     ; Terminal count higher byte
DATA    ENDS
        END START
```

Program 7.1 *ALP for Problem 7.1*

7.3 PROGRAMMABLE DMA INTERFACE 8237

We have described the DMA controller 8257 in the previous section. Now, we will discuss an advanced Programmable DMA Controller 8237, which provides a better performance, compared to 8257. This is capable of transferring a byte or a bulk of data between system memory and peripherals in either direction. Memory to memory data transfer facility is also available in this peripheral. As in the case of 8257, the 8237 also supports four independent DMA channels which may be expanded to any number by cascading more number of 8237. But the distinctive feature of this chip is that it provides, many programmable control and dynamic reconfigurability features which enhance the data transfer rate of the system remarkably. Since, there are architectural differences between 8257 and 8237, we will describe this chip also with adequate details.

7.3.1 Internal Architecture of 8237

The internal block diagram of 8237 is shown in Fig. 7.8. The 8237 contains three basic blocks of its operational logic. The timing and control block generates the internal timings and external control signals. The program command control block decodes the various commands given to the 8237 by the CPU before servicing a DMA request. It also decodes the mode control word used to select the type of the programmed DMA

transfer. The Priority Encoder block resolves priority between the DMA channels requesting the services simultaneously. The timing and control block derives necessary timings from the CLK input.

7.3.2 Register Organisation of 8237

8237 houses a set of twelve types of registers. Some of these registers are present in each of the four channels while the remaining are common for all the channels. Considering the multiple existence of the registers, there are 25 registers in 8237 which are described in detail as follows:

Current Address Register Each of the four DMA channels of 8237 has a 16-bit current address register that holds the current memory address, being accessed during the DMA transfer. The address is automatically incremented or decremented after each transfer and the resulting address value is again stored in the current address register. This can be byte-wise programmed by the CPU, i.e. lower byte first and the higher byte later. This may be reinitialized by an auto-initialization command to its original value after EOP.

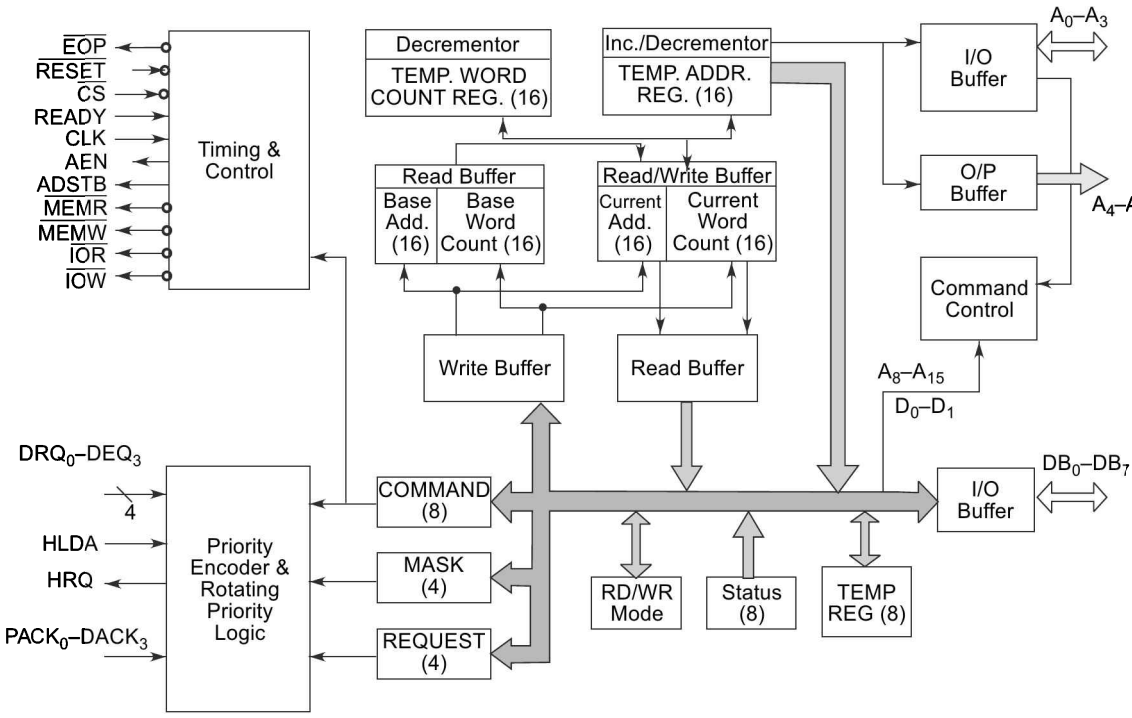


Fig. 7.8 *Block Diagram of 8237*

Current Word Register Each channel has a 16-bit current word register that holds the number of (counts) data byte transfers to be carried out. The word count is decremented after each transfer and the new value is again stored back to the current word register. When the count becomes zero an EOP (end of process) signal will be generated. This can be written in successive bytes by the CPU, in program mode. After the EOP, this may be reinitialized using autoinitialize command.

Base Address and Base Word Count Registers Each channel has a pair of these registers. These maintain an original copy of the respective initial current address register and current word register (before

incrementing or decrementing), respectively. These are automatically written along with the current registers. These cannot be read by the CPU. The contents of these registers are used internally for auto-initialization.

Command Register This 8-bit register controls the complete operation of 8237. This can be programmed by the CPU and cleared by a reset operation. Figure 7.9 shows the definition of the command register.

Mode Register Each of the DMA channel has an 8-bit mode register. This is written by the CPU in program mode. Bits 0 and 1 of the mode register determine which of the four channel mode registers is to be written. The bits 2 and 3 indicate the type of DMA transfer. As we have discussed earlier, there are three types of DMA transfer, viz. memory read, memory write and verify transfer. Bit 4 of the mode register indicates whether auto-initialization is selected or not, while bit 5 indicates whether address increment or address decrement mode is selected. The definition of the mode register is presented in Fig. 7.10.

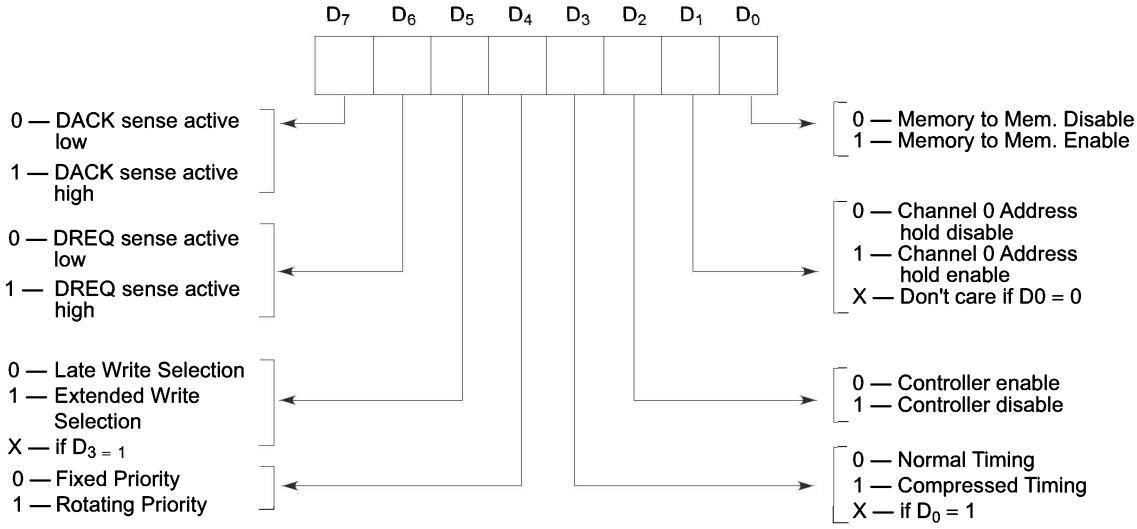


Fig. 7.9 *Command Register Definition*

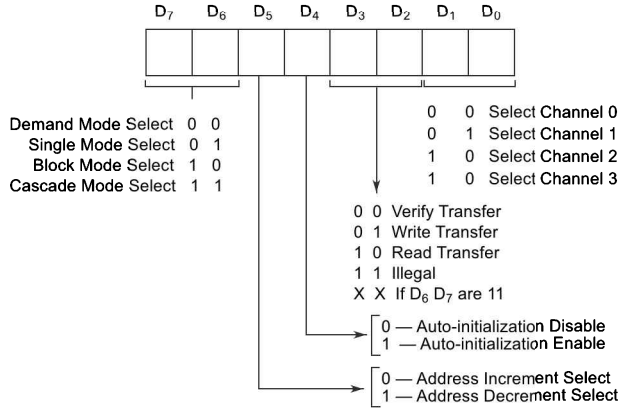


Fig. 7.10 *Mode Register Definition*

Request Register Each channel has a request bit associated with it, in the request register. These are nonmaskable and subject to prioritization by the priority resolving network of 8237. Each bit is set or reset under program control or is cleared upon generation of a TC or an external EOP. This register is cleared by a reset. The bit definitions of the request register are shown in Fig. 7.11.

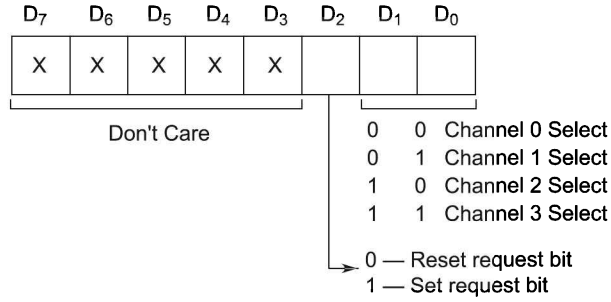


Fig. 7.11 *Request Register Definition*

Mask Register Sometimes it may be required to disable a DMA request of a certain channel. Each of the four channels has a mask bit which can be set under program control to disable the incoming DREQ requests at the specific channel. This bit is set when the corresponding channel produces an EOP signal, if the channel is not programmed for auto-initialization. The register is set to FFH after a reset operation. This disables all the DMA requests till the mask register is cleared. The bit definitions of the mask register are shown in Figs 7.12(a) and (b) respectively. Interestingly, all these mask bits may be cleared using a software command to enable the devices at the respective channels to proceed further for DMA access.

Temporary Register The temporary register holds data during memory-to-memory data transfers. After the completion of the transfer operation, the last word transferred remains in the temporary register till it is cleared by a reset operation.

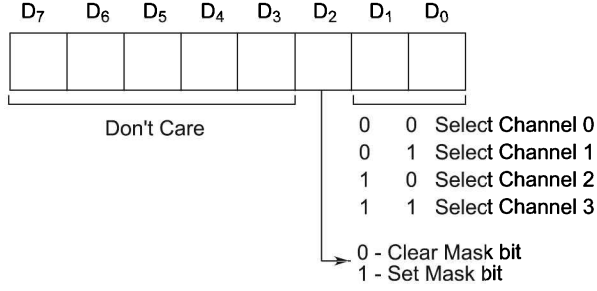


Fig. 7.12(a) *Mask Register Definition to Program the Mask Bits Individually*

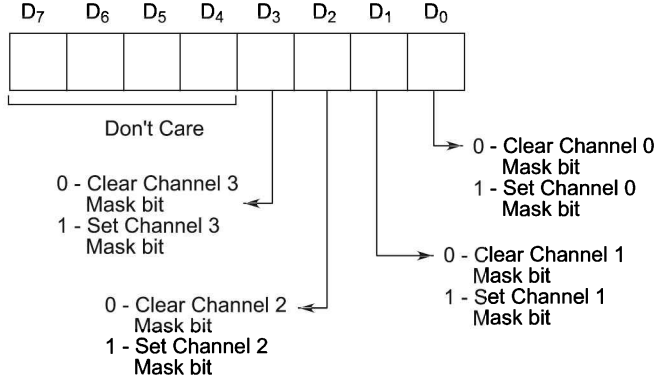


Fig. 7.12(b) *Mask Register Definition to Program all the Mask Bits Simultaneously*

Status Register The status register keeps the track of all the DMA channel pending requests and the status of their terminal counts. The bits D_0-D_3 are updated (set) every time, the corresponding channel reaches TC or an external $\overline{EOP}$ occurs. These are cleared upon reset and also on each status read operation. Bits D_4-D_7 are set, if the corresponding channels request services. Figure 7.13 shows the definition of the status register.

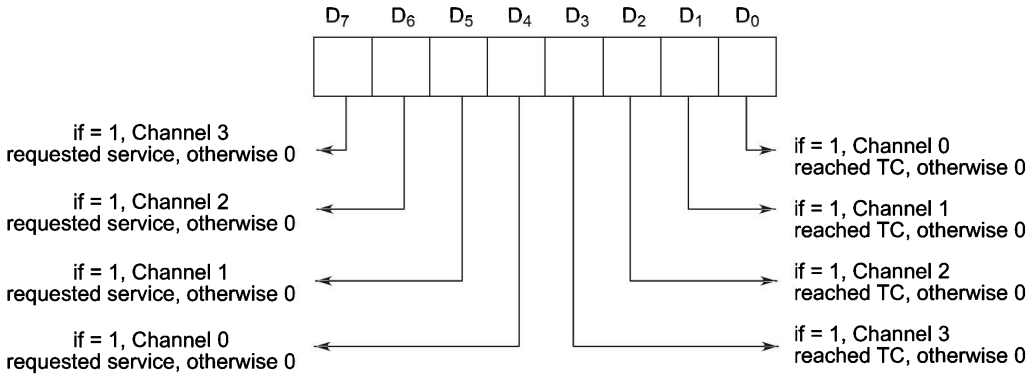


Fig. 7.13 *Status Register Definition*

7.3.3 Signal Descriptions of 8237

Figure 7.14 shows the pin diagram of 8237. The functional signal description of each pin is discussed in brief as follows:

V_{cc} This is a +5V supply pin required for operation of the circuit.

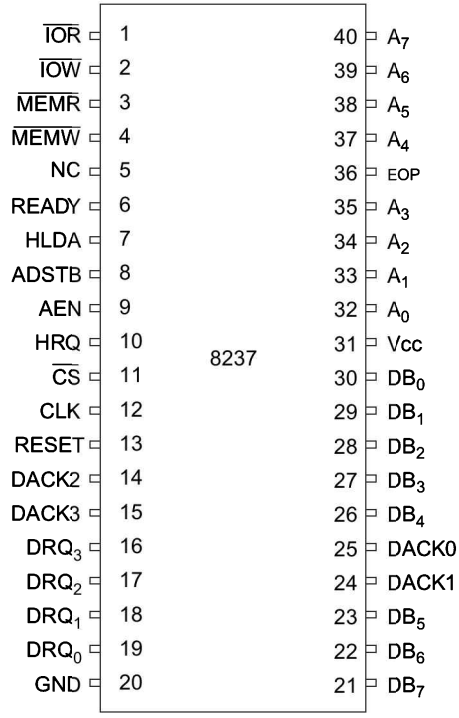
GND This is a return line for the supply (ground pin of the IC).

CLK This is the internally required CLK signal for deriving the internal timings required for the circuit operation.

$\overline{CS}$ This is an active-low chip select input of the IC.

RESET A high on this input line clears the command, status, request and temporary registers. It also clears the internal first/last flip-flop and sets the mask register. The 8237 remains stuck to reset till the RESET pin is high.

READY This active-high input is used to match the read or write speed of 8237 with slow memories or I/O devices.



* Pin 5 should be always at logic high. An internal pull-up pulls it up when floating else it should be tied to V_{CC}.

Fig. 7.14 *Pin Diagram of 8237*

HLDA(Hold Acknowledge) This active-high input signal is to be connected with HLDA pin of the CPU, to indicate to the 8237 that the CPU has relinquished the control of the bus, as a response to a bus request.

$DREQ_0$–$DREQ_3$(DMA Request Inputs) These active-high input lines are the individual channel request inputs driven by peripheral devices to request a DMA service. $DREQ_0$ has the highest priority while $DREQ_3$ has the lowest one. The priorities of the DREQ lines is programmable. After reset, these lines become active-high. The active level of these lines is also programmable. The DREQ must be maintained high until the corresponding DACK pin goes high.

DB_0–DB_7-(Data Bus) These are bidirectional lines used to transfer data to/from I/O or memory. During I/O read, the contents of address register, status register, temporary register or word count register are sent to the CPU over these lines. During I/O write, the CPU sends the data to any of the above registers. In memory to memory operation, the 8237 fetches data from memory using these lines during read-memory transfer cycle. Also the data is transferred to the memory on these eight lines during write-memory transfer cycle.

$\overline{\text{IOR}}$ I/O read is a bidirectional active-low line. In idle cycle or slave mode, this is an input control signal used by the CPU to read its registers. In the active cycle or master mode, it is an output control signal used by 8237 to access data from a peripheral.

$\overline{\text{IOW}}$ I/O write is a bidirectional active-low line. In an idle cycle, it is an input control signal used by CPU to load information into the 8237. In the active cycle, it is an output control signal used by 8237 to transfer data to peripherals.

A_0–A_3 These four least significant address lines are bidirectional three state signals. In idle cycle, they are inputs and are used by C.P.U. to address the control registers to be read or written. In an active cycle, they provide the lowest 4 bits of the output addresses generated by 8237.

A_4–A_7 These are the four most significant address lines activated only during DMA services to generate the respective address bits.

HRQ (Hold Request) The HRQ is an output pin used to request the control of the system bus from CPU. If the corresponding mask bit is not set, every valid $\overline{\text{DREQ}}$ to 8237 will issue a HRQ signal to the CPU. This is connected either to HLD (minimum mode) or to $\overline{\text{RQ}}$ / $\overline{\text{GT}}$ (maximum mode) pin of the CPU.

$DACK_0$–$DACK_3$-(DMA acknowledge) These DMA acknowledge output pins are used to indicate the individual peripheral that it has been granted a DMA cycle by the CPU, in coordination with 8237. After reset, these are active-low, but may be programmed as required.

AEN (Address Enable) This active-high output enables the 8-bit latch that drives the upper 8 bit address bus. The AEN pin is used to disable other bus drivers during DMA transfers.

ADSTB (Address Strobe) This output line is used to strobe the upper address byte generated by 8237, in master mode into, an external latch.

$\overline{\text{MEMR}}$ This active-low output is used to access data from the selected memory location, during DMA read or a memory to memory transfer.

$\overline{\text{MEMW}}$ This active-low output signal is used to write data to the selected memory location during DMA write or a memory to memory transfer.

$\overline{\text{EOP}}$ (End of Process) This is an active low-bidirectional (input or output) pin, used to indicate the completion of a DMA operation. Also, an external signal can terminate a DMA operation by driving this pin low. The 8237 generates a pulse when terminal count is reached, i.e. the transfer byte count reaches zero. This generates EOP signal at this pin. The reception of EOP, either internal or external, will cause 8237 to terminate the service, reset the request and to copy the base registers into the current registers, if auto-initialize is enabled. During memory to memory transfers, EOP signal will be generated, when the terminal count for channel 1 occurs. The $\overline{\text{EOP}}$ pin should be pulled high, if it is not in use, to avoid erratic end of process.

7.3.4 DMA Operations with 8237

The 8237 operates in two cycles, viz. idle or *passive cycle* and *active cycle*. Each cycle contains a fixed number of states. The 8237 can assume six states, when it is in active cycle. During idle cycle, it is in state SI (Idle State).

The 8237 is initially in a state SI, i.e. an idle state where the 8237 does not have any valid pending DMA request. During this time, although the 8237 may be idle, the CPU may program it in this state. Once there is a DMA request, the 8237 enters state S_0, which is the first state of the DMA operation. When the 8237 requests the CPU for a DMA operation, and the CPU has not acknowledged the request, the 8237 waits in S_0 state. The acknowledge signal from the CPU indicates that the data transfer may now begin. The S_1, S_2, S_3 and S_4 are the working states of DMA operation, in which the actual data transfer is carried out. If more time is required to complete a transfer (by the peripheral) than that is allowed (by the controller), wait states (S_W) may be inserted between S_2 and S_3 or S_3 and S_4 using the READY pin of 8237. From the above discussion, it is clear that a memory read or a memory write DMA operation actually requires four states, i.e. S_1 to S_4.

A memory-to-memory transfer is a two cycle operation, and requires a read-from and a write-to memory cycle to complete each DMA transfer. Each of these two types of cycles, require four states for its completion. Thus eight states may be required for a single memory-to-memory DMA transfer. Each of these four states use two digit numbers for its identification. For example, for a memory read DMA cycle, the four states required may be represented as S_{11}, S_{12}, S_{13} and S_{14}. The first four states (S_{11}, S_{12}, S_{13} and S_{14}) are used for read-from-memory cycle and the next four states (S_{21}, S_{22}, S_{23} and S_{24}) are used for write-to-memory cycle. A memory to memory transfer cycle is explained in more details later in this section.

7.3.5 Transfer Modes of 8237

The 8237 is in the idle cycle if there is no pending request or the 8237 is waiting for a request from one of the DMA channels. Once a channel requests a DMA service, the 8237 sends the HOLD request to the CPU using its HRQ pin. If the CPU acknowledges the hold request on HLDA, the 8237 enters an active cycle. In the active cycle, the actual data transfer takes place in one of the following transfer modes, as is programmed.

Single Transfer Mode In this mode, the device transfers only one byte per request. The word count is decremented and the address is decremented or incremented (depending on programming) after each such transfer. The Terminal Count (TC) state is reached, when the count becomes zero. For each transfer, the DREQ must be active until the DACK is activated, in order to get recognized. After TC, the bus will be relinquished for the CPU. For a new DREQ to 8237, it will again activate the HRQ signal to the CPU and the HLDA signal from the CPU will push the 8237 again into the single transfer mode. This mode is also called as 'cycle stealing'.

Block Transfer Mode In this mode, the 8237 is activated by DREQ to continue the transfer until a TC is reached, i.e. a block of data is transferred. The transfer cycle may be terminated due to $\overline{\text{EOP}}$ (either internal or external) which forces Terminal Count (TC). The DREQ needs to be activated only till the DACK signal is activated by the DMA controller. Auto-initialization may be programmed in this mode.

Demand Transfer Mode In this mode, the device continues transfers until a TC is reached or an external $\overline{\text{EOP}}$ is detected or the DREQ signal goes inactive. Thus a transfer may exhaust the capacity of data transfer of an I/O device. After the I/O device is able to catch up, the service may be re-established activating the DREQ signal again. Only the EOP generated by TC or external EOP can cause the auto-initialization, and only if it is programmed for.

Cascade Mode In this mode, more than one 8237 can be connected together to provide more than four DMA channels. The HRQ and HLDA signals from additional 8237s are connected with DREQ and DACK

pins of a channel of the host 8237 respectively. The priorities of the DMA requests may be preserved at each level. The first device is only used for prioritizing the additional devices (slave 8237s), and it does not generate any address or control signal of its own. The host 8237 responds to DREQ generated by slaves and generates the DACK and the HRQ signals to coordinate all the slaves. All other outputs of the host 8237 are disabled.

Memory to Memory Transfer To perform the transfer of a block of data from one set of memory address to another one, this transfer mode is used. Programming the corresponding mode bit in the command word, sets the channel 0 and 1 to operate as source and destination channels, respectively. The transfer is initialized by setting the $DREQ_0$ using software commands. The 8237 sends HRQ (Hold Request) signal to the CPU as usual and when the HLDA signal is activated by the CPU, the device starts operating in block transfer mode to read the data from memory. The channel 0 current address register acts as a source pointer. The byte read from the memory is stored in an internal temporary register of 8237. The channel 1 current address register acts as a destination pointer to write the data from the temporary register to the destination memory location. The pointers are automatically incremented or decremented, depending upon the programming. The channel 1 word count register is used as a counter and is decremented after each transfer. When it reaches zero, a TC is generated, causing EOP to terminate the service.

The 8237 also responds to external EOP signals to terminate the service. This feature may be used to scan a block of data for a byte. When a match is found the process may be terminated using the external EOP.

Under all these transfer modes, the 8237 carries out three basic transfers namely, write transfer, read transfer and verify transfer. In write transfer, the 8237 reads from an I/O device and writes to memory under the control of $\overline{\text{IOR}}$ and $\overline{\text{MEMW}}$ signals. In read transfer, the 8237 reads from memory and writes to an I/O device by activating the $\overline{\text{MEMR}}$ and $\overline{\text{IOW}}$ signals. In verify transfers, the 8237 works in the same way as the read or write transfer but does not generate any control signal.

7.3.6 Address Generation in DMA

The 8237 multiplexes the eight higher order address bits (A_8–A_{15}) on the data lines. The state S_1(idle) is used to transfer the higher order address bits to a latch from which they are to be placed on the address bus. The falling edge of the Address Strobe (ADSTB) signal is used to load these higher address bits from data lines to the latch while that of AEN is used to place the same to the system address bus through a tristate buffer. The lower order address A_0–A_7 is directly generated by 8237 on its A_0–A_7 pins.

7.3.7 8237 Commands and Programming

There are several commands which can be executed by 8237 when it is in program condition. Some of these commands are discussed as follows:

Clear First/Last Flip-Flop As we have discussed in case of 8257, there exists an internal flip-flop in 8237 also which is called First/Last flip-flop (F/L ff). This flip-flop output decides whether the lower byte or the upper byte of the selected 16-bit register will be read or written. Here, the selected register means the current address register or the current word count register. Thus by clearing the first/last flip-flop by this command, the CPU will address the higher or lower byte in an appropriate sequence.

Clear Mask Register As has been described earlier, a mask set register when set, may disable the DMA channels, so that the DMA requests are not entertained. A clear mask register command will clear the bits of the mask register individually or collectively, so that the DMA channels are enabled for accepting DMA requests.

Master Clear Command Using this command, all the internal registers of 8237 are cleared, while all the bits of the mask register are set. This means after executing this command , the DMA controller disables all the DMA channels and enters an idle cycle.

Along with these commands, a few others have been tabulated in Fig. 7.15(a), while Fig. 7.15(b) tabulates the command codes for manipulating the current address registers and the current word count registers. These commands may be executed while the 8237 is under the control of the CPU. If $\overline{CS}$ is low and the HLDA is inactive, the 8237 enters the program condition. The lines A_0 to A_3, $\overline{IORD}$ and $\overline{IOWD}$ are used to select and program the registers of 8237. Note that the address lines A_1 and A_2 select one of the four DMA channels, while the line A_0 selects one of the two registers for a channel.

Problem 7.2

Design an interface between the programmable DMA controller 8237 and 8086. The command register address of the 8237 is F8H. Select the corresponding addresses for all the other registers of 8237.

Solution The interface between 8086 and 8237 is shown in Fig. 7.16. The interesting part of the circuit lies in the transfer of data to/from the odd addresses on the 8-bit data bus. Being an 8-bit device, the 8237 is able to transfer data on D_0–D_7 but for an odd address, the 8086 memory system transfers data on D_8–D_{15}. Hence, the higher byte of the data bus is imposed on the lower byte, using two 74LS245 buffers, if the 8237 is in master mode. Note that the data transfer at an even address is carried out necessarily on D_0–D_7, while the transfer at an odd address is carried out on D_8–D_{15} data lines of the memory system. The transfers at odd addresses will receive/send data from/to AX (the unused AL will be don't care), during initialization of 8237.

Signals						Operations
A_3	A_2	A_1	A_0			
1	0	0	0	0	1	Read Status Register
1	0	0	0	1	0	Write Command Register
1	0	0	1	0	1	Illegal
1	0	0	1	1	0	Write Request Register
1	0	1	0	0	1	Illegal
1	0	1	0	1	0	Write Single Mask Register Bit
1	0	1	1	0	1	Illegal
1	0	1	1	1	0	Write Mode Register
1	1	0	0	0	1	Illegal
1	1	0	0	1	0	Clear Byte Pointer Flipflop
1	1	0	1	0	1	Read Temporary Register
1	1	0	1	1	0	Master Clear
1	1	1	0	0	1	Illegal
1	1	1	0	1	0	Clear Mask Register
1	1	1	1	0	1	Illegal
1	1	1	1	1	0	Write All Mask Register Bit

Fig. 7.15(a) *Software Command Codes*

Channel	Register	Operation	Signals							Internal Flip-Flop	Data Bus DB_0-DB_7
			$\overline{CS}$	$\overline{IOR}$	$\overline{IOW}$	A_3	A_2	A_1	A_0		
0.	Bass and Current Address	Write	0	1	0	0	0	0	0	0	A_0–A_7
			0	1	0	0	0	0	0	1	A_8–A_{15}
	Current Address	Read	0	0	1	0	0	0	0	0	A_0–A_7
			0	0	1	0	0	0	0	1	A_8–A_{15}
	Base and Current Word Count	Write	0	1	0	0	0	0	1	0	W_0–W_7
			0	1	0	0	0	0	1	1	W_8–W_{15}
	Current Word Count	Read	0	0	1	0	0	0	1	0	W_0–W_7
			0	0	1	0	0	0	1	1	W_8–W_{15}
1.	Base and Current Address	Write	0	1	0	0	0	1	0	0	A_0–A_7
			0	1	0	0	0	1	0	1	A_8–A_{15}
	Current Address	Read	0	0	1	0	0	1	0	0	A_0–A_7
			0	0	1	0	0	1	0	1	A_8–A_{15}
	Base and Current Word Count	Write	0	1	0	0	0	1	1	0	W_0–W_7
			0	1	0	0	0	1	1	1	W_8–W_{15}
	Current Word Count	Read	0	0	1	0	0	1	1	0	W_0–W_7
			0	0	1	0	0	1	1	1	W_8–W_{15}
2.	Bass and Current Address	Write	0	1	0	0	1	0	0	0	A_0–A_7
			0	1	0	0	1	0	0	1	A_8–A_{15}
	Current Address	Read	0	0	1	0	1	0	0	0	A_0–A_7
			0	0	1	0	1	0	0	1	A_8–A_{15}
	Base and Current Word Count	Write	0	1	0	0	1	0	1	0	W_0–W_7
			0	1	0	0	1	0	1	1	W_8–W_{15}
	Current Word Count	Read	0	0	1	0	1	0	1	0	W_0–W_7
			0	0	1	0	1	0	1	1	W_8–W_{15}
3.	Base and Current Address	Write	0	1	0	0	1	1	0	0	A_0–A_7
			0	1	0	0	1	1	0	1	A_8–A_{15}
	Current Address	Read	0	0	1	0	1	1	0	0	A_0–A_7
			0	0	1	0	1	1	0	1	A_8–A_{15}
	Base and Current Word Count	Write	0	1	0	0	1	1	1	0	W_0–W_7
			0	1	0	0	1	1	1	1	W_8–W_{15}
	Current Word Count	Read	0	0	1	0	1	1	1	0	W_0–W_7
			0	0	1	0	1	1	1	1	W_8–W_{15}

Fig. 7.15(b) *Word Count and Address Register Commands*

The addresses of the internal registers of the 8237 are listed as follows:

Command Register FEH
Mode Register FBH
Request Register F9H
Mask Register FA/FFH Individual/ Common Mask
Status Register F8H
Temporary Register FDH
Byte Pointer Flip-Flop FCH
Base and Current Address Registers
Channel 0 F0H
Channel 1 F2H
Channel 2 F4H
Channel 3 F6H
Base and Current Word Count Register
Channel 0 F1H
Channel 1 F3H
Channel 2 F5H
Channel 3 F7H

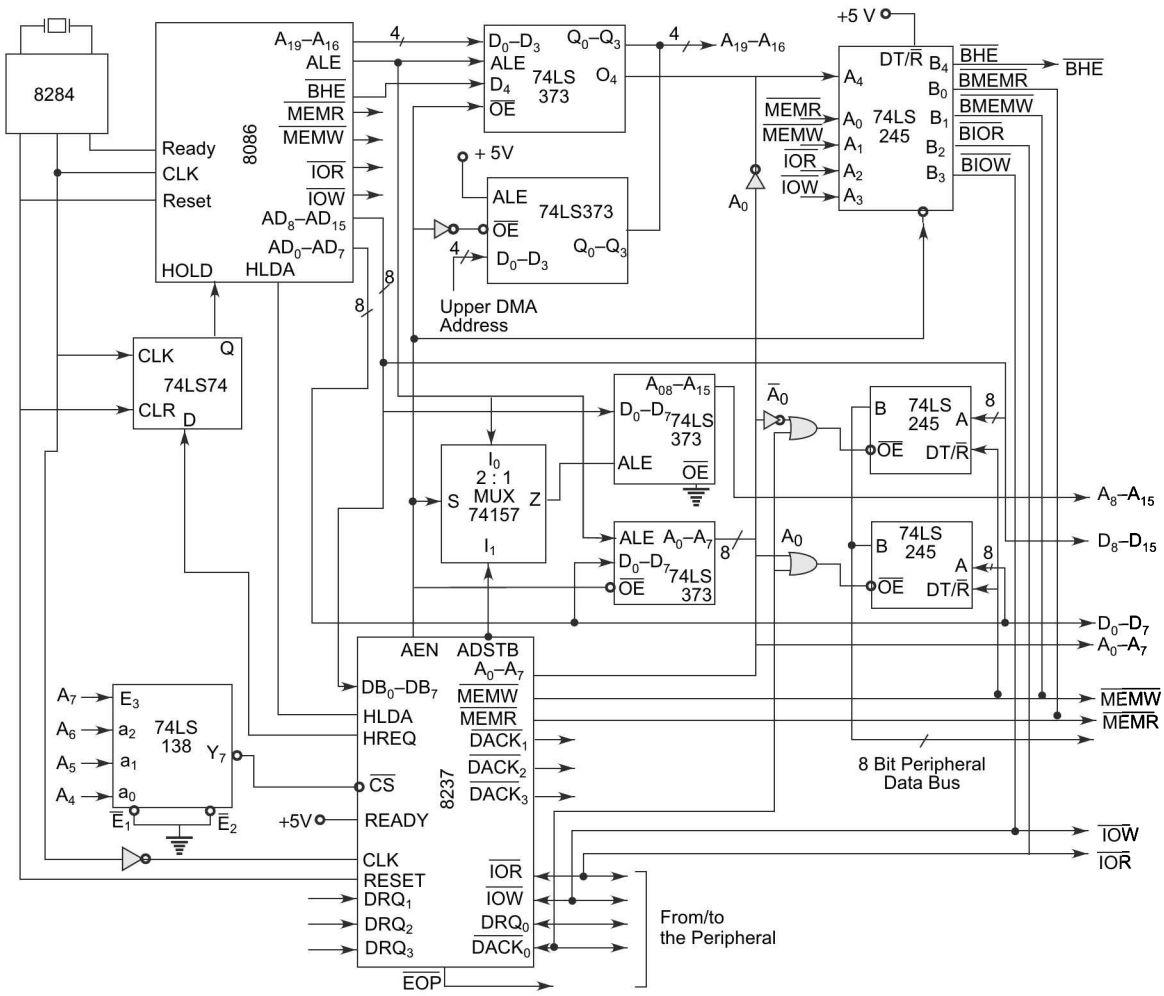


Fig. 7.16 *Interfacing 8237 with 8086*

Problem 7.3

With the above hardware system of Problem 7.2, initialize the 8237 for memory-to-memory DMA transfer mode using channel 0, masking all other channels. Initialize the 8237 for normal timings, fixed priority, extended write with DREQ active high and DACK active-high. The 8237 should work in auto-initialization mode with address increment, block mode select with read transfer on channel 0. Further, write a program to transfer a data block of size 4KB available at 5000:0000H to 5000:1000H.

Solution Different programmable register contents for the initialization as per the problem specifications are given. Note that the upper data bus D_8-D_{15} drives the data lines of the DMA controller. The reader may refer to the bit patterns for each register presented in the register organisation section.

Command Word Register

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	1	0	0	0	0	1	= A1 H

Mode Register Word

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	1	1	0	0	0	= 98 H

Request Register

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	0	0	0	0	1	0	0	= 04 H

Mask Register

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

In memory to memory transfer, channel 0 acts as source, i.e. the channel 0 current address register contains source address and the channel 1 current address register contains destination address. The channel 1 current word count register contains the block length count. Program 7.2 gives the ALP for this problem.

```
ASSUME      CS : CODE
CODE        SEGMENT
START:      MOV AX,0A100H       ; Out command word A1
            OUT 0F8H,AX         ; to port F8 for initialization
            MOV AX, 9800H       ; Out mode word to port FB
            OUT 0FBH,AX         ; for mode programming
            MOV AX,0E00H        ; Mask all requests
            OUT 0FFH,AX         ; except channel 0 (DREQ0).
            MOV AX, 00H         ; Clear byte pointer
            MOV 0FCH,AX         ; flip-flop
            MOV AX, 00H         ; Write base and current address
            OUT 0F0H,AX         ; Register of channel 0 in
            OUT 0F0H,AX         ; successive byte transfers
            MOV AX, 00H         ; Clear byte pointer flip-flop
            OUT 0FCH,AX         ;
            MOV AX, 00H         ; Write base and current address
            OUT 0F2H,AX         ; register of channel 1 in
            MOV AX, 1000H       ; successive byte transfers
            OUT 0F2H, AX        ;
            MOV AX, 00H         ; Clear byte pointer flip-flop
            OUT 0FCH,AX         ;
            MOV AX,0FF00H       ; Load word count 0FFFH
            OUT 0F3H,AX         ; in channel 1 word count
            MOV AX,0F00H        ; register in successive bytes
            OUT 0F3H,AX         ;
            MOV AX,0400H        ; Initialize the transfer
            OUT 0F9H,AX         ; by setting DREQ request
            NOP                 ; Wait for transfer to start
            MOV AH,4CH          ; Return to DOS
            INT 21H
CODE        ENDS
            END START
```

Program 7.2 ALP for Problem 7.3

Note that the segment addresses cannot be handled by 8237. Hence in a single DMA operation only 64K bytes of data can be transferred, in block transfer mode.

7.4 HIGH STORAGE CAPACITY MEMORY DEVICES

7.4.1 Floppy Disks

Floppies are the most commonly used secondary memory devices, which store data by the virtue of the magnetic material coated on their surface. The floppies are available in three sizes, viz. standard 8", $5^{1/4}$" mini-floppy and $3^{1/2}$" microfloppy. Previously, the 8" standard size was mostly in use, but its large size, low memory capacity and low mechanical strength made it obsolete. Whatever their physical sizes and storage formats, all the floppies incorporate the basic principles of magnetic data recording and reading.

The data is stored on a set of concentric rings known tracks, on their surfaces, which is further divided into sectors. A sector is supposed to contain either 512 or 1024 bytes of data at maximum. However, the sector size may vary from 128 bytes to the entire size of the track. The concentric tracks are subdivided into sectors using radial logical separations, as shown in Fig. 7.17(a).

The circular disk type flexible media, coated with the magnetic material is enclosed in a plastic jacket for its protection. The jacket has a small circular hole near the central circular (big) hole that is used to drive (rotate) the circular disk inside the jacket. This rotation of the disk enables a drive head to scan a complete circular track. The small hole is called the index hole. This enables the drive to identify the beginning of a track and its first sector. The track 00 is the outermost track on the disk surface and the sector 00 is the first sector on the track. The track number and sector number go on increasing till the numbering reaches the innermost track, and its last sector. The head slot allows the drive head to move radially over the disk surface to refer to the different tracks. The write protect notch is to be covered by a sticker to inhibit writing to the disk. Figure 7.17(b) shows a typical $5^{1/4}$" floppy with its details.

The floppy media is rotated at the speed of 300 RPM (Revolution per minute) inside its jacket. A recent development has been the recording of data on both sides of the disk. These types of disks are called double sided disks. The two tracks on the two surfaces beneath each other are collectively referred to as cylinders. Thus a cylinder contains two tracks. The cylinders are numbered in the same way as tracks. The Double Density Double Sided (DSDD) disks are organised with 40 tracks on each side of the disk. A Double Density (DD) disk track is divided into nine sectors each containing 512 bytes of data. Thus the disk contains 40 (tracks) × 2 (surfaces) × 9 (sectors) × 512 (bytes per sector), i.e. 360 Kbytes of data.

The high density diskettes have 80 tracks per side with 8 sectors per track and 1024 bytes per sector. Thus these diskettes can store up to $80 \times 2 \times 8 \times 1024$, i.e. 1.2M bytes. The magnetic recording technique used for storing data onto the disks is called as Non-Return to Zero (NRZ) recording. In this technique, the magnetic flux on the disk surface never returns to zero, i.e. no erase operation is carried out.

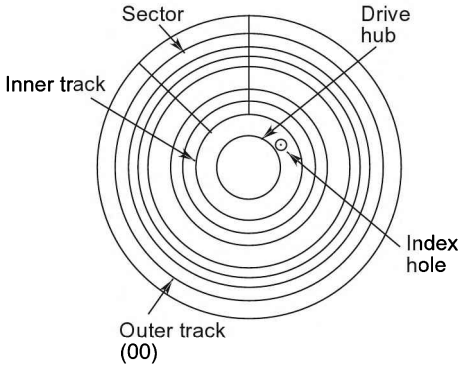


Fig. 7.17(a) *Format of a $5^{1/4}$" Floppy*

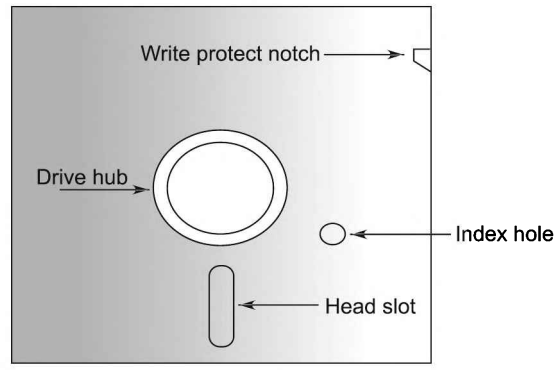


Fig. 7.17(b) *$5^{1/4}$" Mini Floppy*

The most recent form of a floppy disk is its $3^{1/2}$" microfloppy format. This is a much improved version of the $5^{1/4}$" floppy disk. This is small in size and packed in a strong plastic jacket that does not bend easily. Thus a microfloppy is easy to handle and is more durable. The other problem with $5^{1/4}$" floppy was its permanently open head slot that exposed the media to other contaminants and dust, reducing the media life due to wear problems. The microfloppy has a head slot covered with a spring controlled door that only opens at the time of media access. The write protect window is located at the corner of the jacket. The sticker or the tape used for write protection in $5^{1/4}$" disk is now replaced by an unremoveable sliding plastic square disk, that is only to be slided for write protecting the disk. Previously, the sticker or tapes used for write protections used to get dislodged inside the floppy drives, causing problems.

The index hole is replaced by a slightly different drive mechanism. The $3^{1/2}$" floppy has a drive mechanism that fits the drive hub only in a unique position inside the floppy drive. Thus, the $3^{1/2}$" floppy has gotten rid of the index hole, that used to create problems due to dirt or dust.

The DSDD $3^{1/2}$" floppy is organized in 80 tracks per side, containing nine sectors each. Each of the sectors can store 512 bytes of data. Thus the disk can store $80 \times 2 \times 9 \times 512$, i.e. 720 KB of data. The $3^{1/2}$" DSHD floppy is organized in 80 tracks per side, each containing 18 sectors. Each of the sectors can store upto 512 bytes. Thus the disk can store $80 \times 2 \times 18 \times 512$, i.e. 1.44MB of the data. Figure 7.17(c) shows the $3^{1/2}$" floppy diskette.

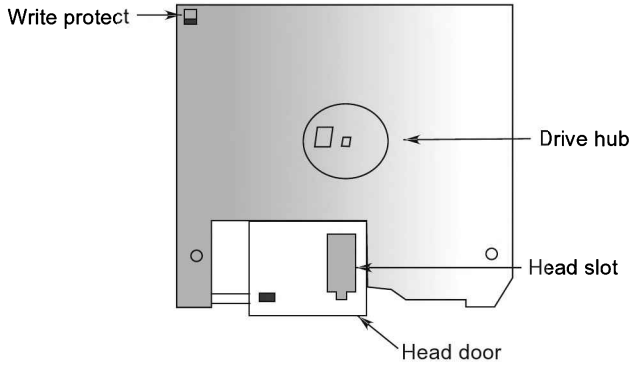


Fig. 7.17(c) *$3^{1/2}$" Micro Floppy*

7.4.2 Compact Disc (CD)

The Compact Disc (CD) is an optical disk used to store digital data. It was originally developed to store and playback sound recordings exclusively, but the format was later adapted for storage of data (CD-ROM), write-once audio and data storage (CD-R), rewritable media (CD-RW), Video Compact Discs (VCD), etc. Standard CDs have a diameter of 120 millimeters (4.7 in) and can hold up to 80 minutes of uncompressed audio or 700 MB (700×2^{20} bytes) of data. Mini CDs have various diameters ranging from 60 to 80 millimeters (2.4 to 3.1 in). They are sometimes used for CD delivering device drivers. CDs in various forms are widely used in computer industry.

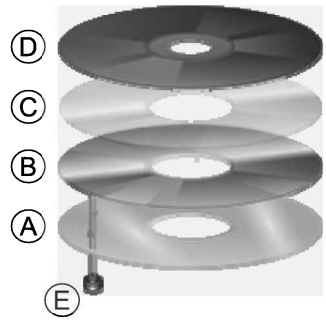


Fig. 7.18 *A Typical CD Structure*

The CD technology is an advanced form of the Laser Disc technology. A typical CD is a circular disk with a holding ring at the center. The Drive arrangement holds the CD tightly at the central ring and can rotate it. A movable laser beam can scan the disk radially due to its own movement as well as along the circular concentric tracks similar to those on a floppy disk due to rotation. While recording digital data on the polycarbonate disk, the laser beam creates pits for logical '1' and no pits for logical '0' using a non-return to zero type of coding scheme. While reading the data, a laser beam scans the pits or no-pits tracks generated by the recording beam and interprets the pits or no-pits track in terms of the recorded sequences of '1's and '0's using the reflected optical laser beam from the tracks. A typical CD structure is shown in Fig.7.18.

A polycarbonate disk layer has the recorded data using pits or no-pits. A shining layer below the data layer reflects the layer for reading. A lacquer layer below the shining protects the shining layer. A label or artwork graphics is screen printed on the top of the lacquer layer of the disk. A laser beam scans the CD along the tracks and reflects it back to a sensor, which is further converted into a bit sequence waveform and interpreted in terms of '1's and '0's. Different standards and formats have been used for storing different types of data

like audio, video, or computer files on the CDs. Though CDs were initially introduced as read only (CD-ROM)disks, very soon they were advanced to Read/Write CD(RD/WR). CDs have typical data transfer rates of a few MB per second. The writing however is very slow.

7.4.3 Digital Video Disk

A DVD represents a family of CD type of disks for digital video signal storage. Amongst other similar video storages, **DVDR** is for recordable DVD and **DVDRW** for rewriteable DVD. Using recent DVDs, it is also possible to have up to 4.6 GB ordinary data on a recordable DVD. Due to its high storage capacity and density, DVDs have also become popular as portable computer data storage devices. DVD data transfer rates of up to 30 Mbps have been achieved so far.

A DVD is also basically an optical disk that uses a red laser for reading the disks. DVDs offer higher storage capacity than compact disks while having the same dimensions. Pre-recorded DVDs are produced in large quantities using molds that physically impress data onto the DVDs. Such disks are called DVD ROM. DVD ROMs are not re-recordable. Blank one-time recordable DVDs (DVD-R and DVD+R) can be recorded using a DVD recorder. Rewritable DVDs (DVD-RW or DVD-RAM) can be erased and then written many times. Standard formats are available to write video data onto the disks. DVDs containing other data may follow other recording standards for storing non-video data. Unlike CDs, DVDs can store data in multiple layers. Some DVD specifications (e.g. for DVD-Video) are openly available and have to be purchased from the DVD Format/Logo Licensing Corporation by paying the license fees.

The disk is held by a centrally located hub with rotating mechanism. A very fine laser beam (635 nm) and an optical system can scan the underside of the disk by rotating it and moving the reading laser beam radially. In fact, the CD and DVD driving mechanisms are very complex and are not the topic of discussion here.

A dual-layer disk incorporates a second physical layer within the disk itself. The dual-layer disk accesses the second layer by penetrating the laser beam through the first semitransparent layer. The dual-layer disks are costlier than single-layer disks. There are two modes for dual-layer orientation. Dual-layer disk data transfer rates are slow especially when a change of layer is required.

7.4.4 Blue Ray Disk

Blue ray Disk (BD) is also an optical medium that outperforms the DVDs. Physically, it is similar to a CD or DVD, but it can store data up to 25 GB per layer and is being popularly used for storing long-duration videos like movies. BDs are different compared to CDs and DVDs in terms of hardware specifications and multimedia storage formats. They can provide better resolution compared to DVDs. The Blue-ray Disk uses a high-frequency blue laser with a very small wavelength to read the disk. Thus, BDs can store data at much greater density than DVDs. In course of time, they have been named 'Blu-ray' disks. With advancements in BD technology, BDs have outperformed DVDs and have become more popular in movie markets. Due to huge storage capacities, BDs also require high-speed motors (around 10000 rpm) to access them. However, writing of disks at this high speed causes improper writing due to wobbling. Many formats for storing data and multimedia on to BDs are available. A few formats store data at high speed but at low resolution, and the remaining formats store data at low speed but at higher resolution. BD standards like BDXL are able to store up to 128 GB data on the disks. Re-recordable BDs are also available. Audio, video, and other synchronization streams are multiplexed and stored on the disks in a container format like data packets. Different BD formats are available for audio, video and other data-storage applications. BDs can support reading rates of 24 frames per second for a resolution of 1920 x 1080 pixels. The data-transfer rates achieved till date are around 54 Mbits/second.

7.4.5 Hard Disk Drives

A Hard Disk Drive (HDD) is the main and largest secondary data storage in a computer. The operating system, device drivers, system software, user application programs and most other files are stored in the hard

disk drives. The HDD is an electromechanical arrangement to handle and access the magnetic storage media available in the form of single or multiple coaxial thick cylindrical disks. The cylindrical disk surfaces are coated with magnetic media and the bits are stored on the circular tracks similar to floppy disks. A single head or multiple heads can access the surface areas of the disks. A read/write head can radially scan the surface while rapid rotation of the disks facilitates complete surface access in terms of concentric rings called tracks. Tracks are further divided into sectors. Thus, each cylinder has tracks and sectors on its upper and lower circular surface. The common axis of the cylinders is driven by an electric motor to obtain the rotations. However, all the cylinders rotate even if the data is being accessed on only one cylinder surface due to the common axis. An electronic drive mechanism precisely controls the position of the head, disks and other electrical and mechanical parts of the drive. The data bit stream reading and writing is basically an electromagnetic phenomenon. The digital data bits are stored in the form of orientations of electromagnetic dipoles along the tracks and sectors on the surface of cylindrical disks. Thus, the data is permanently available even in the absence of power supply. The HDD is also called hard drive, hard disk, fixed drive, fixed disk or fixed disk drive. The currently available HDDs have huge data storing capacities like 1000 to 1200 GB. The physical dimensions of HDDS have reduced from several feet to a few inches and the weight has reduced from several hundred kilograms to a few hundred grams in the last fifty years. On the other hand, their data-storage capacity has been increased millions of times. Currently available HDDs have data-transfer rates of up to 1 Gbps with a disk rotation system of 7200 rpm. A typical hard disk drive component are shown in Fig. 7.19. The top left part of the image shows the first part of HHD chassis that mounts the electronic control mechanism. The top right part shows the second part of the metal chassis that holds the coaxial cylindrical disks. The bottom left image shows the read/write head carrying the lever and the electronic control card. The bottom right part of the image shows the magnetic storage media disks and the disk holding bay.

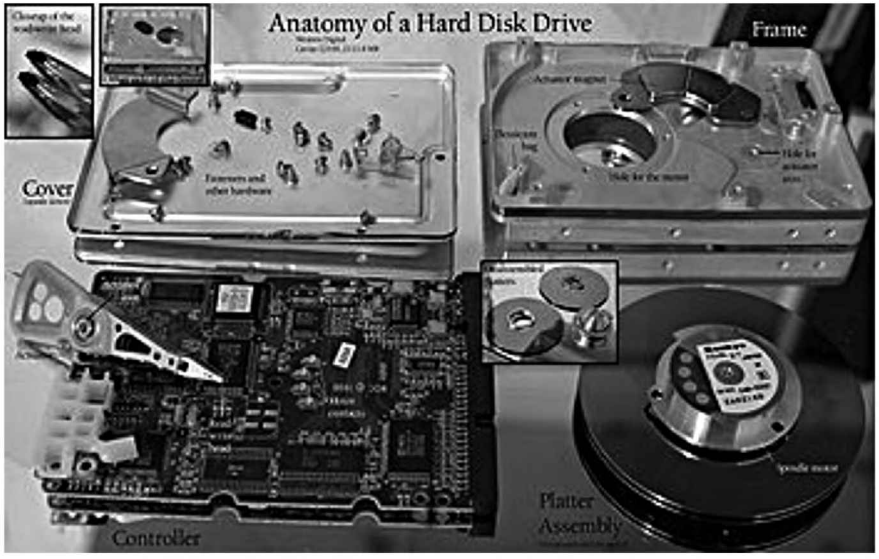
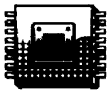


Fig. 7.19 *Internal Components of an HDD Drive*

Before actually using a hard disk drive, it must be formatted using an appropriate operating system. A HDD can, however, have multiple partitions to store more than one operating system. The hard disk drive can be logically partitioned into many drives that can virtually act as physically separate drives. Hard disk drives spare some of their memory capacity for defect management and error correction. Resident programs like operation systems and device drivers also consume some portion of the memory capacity. The remaining memory capacity is used for storing user data. Access time of an HDD is the time duration from the instant of issuing a read or write command to the instant at which the byte becomes available or the write operation is complete. *Seek time* is a measure

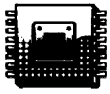
of how long it takes the head assembly to travel to the track of the disk radially that contains data. Rotational latency causes some duration to pass before the desired disk sector comes under the head when data transfer is requested. These two delays are of the order of a few milliseconds each. Once the head reaches the appropriate position, the bit rate causes a delay depending upon the read-write speed of the media and the size of the block to be read. Additional delay may also be introduced if the drive disks are stopped in between for some reason.



SUMMARY

This chapter presents a detailed account of the operation of some of the DMA based advanced peripherals. With the recent advances in the field, the Intel family of the peripherals, has been continuously enhanced with a number of new peripherals. Of course, all of them cannot be discussed here, but a few of them, which are frequently used in the industrial and general systems are discussed in this chapter, in significant details. Some of the advanced peripherals like the floppy disk controllers and the CRT controllers are covered here in details, since they are the unavoidable parts of the advanced microprocessor based systems.

This chapter starts with the discussion on a DMA controller 8257. The necessary functional details of 8257 have been discussed along with an interfacing example and the supporting program. Then, the advanced DMA controller 8237 has been studied in significant details. At the end, a brief introduction to high capacity memory devices has been presented.



EXERCISES

7.1 What is the advantage of DMA controlled data transfer over interrupt driven or program controlled data transfer? Why are DMA controlled data transfers faster?

7.2 Draw and discuss the architecture of 8257.

7.3 Draw and discuss the mode set register of 8257.

7.4 Explain the functions of the following signals of 8257.

(i) $\overline{\text{IOR}}$ (ii) $\overline{\text{IOW}}$ (iii) HRQ (iv) HLDA

(v) $\overline{\text{MEMR}}$ (vi) $\overline{\text{MEMW}}$ (vii) TC (viii) AEN

(ix) ADSTB (x) MARK

7.5 Draw and discuss the status register of 8257.

7.6 What are the registers available in 8257? What are their functions?

7.7 Discuss the priorities of DMA request inputs of 8257.

7.8 An 8086 system has a DMA controller 8257 interfaced such that address of its mode set register is F8H and address of its DMA address register of channel 0 is F0H. Write an ALP to read 2K bytes of data from location 5000H : 2000H in the system memory to a peripheral on channel of the DMA controller. Disable all other channels, program TC stop, no autoload is required, normal priority.

7.9 Bring out the advances in 8237 over 8257.

7.10 Discuss the functions of different registers of 8237.

7.11 Discuss the formats of the following registers of 8237.

(i) Command Register (ii) Mode Register

(iii) Request Register (iv) Mask Register

(v) Status Register

7.12 Discuss the function of EOP signal of 8237.

7.13 Discuss different states of operation of 8237 during different types of transfers.

7.14 Discuss the following modes of DMA transfer.

(i) Signal transfer mode (ii) Block transfer mode

(iii) Demand transfer mode (iv) Memory to memory transfer

7.15 What do you mean by the cascade operation of 8237? Why is it required?

7.16 Discuss the address generation by 8237 during DMA operation.

7.17 Discuss the different commands supported by 8237.

7.18 What do you mean by cycle stealing?

7.19 Write a program to initialise 8237 for all channels enabled, rotating priority, non-extended write, DREQ active low and cycle stealing. The 8237 need not be in autoinitialization mode. The DACK output should be active high. The DMA controller should wait for a DMA request on any of the channels and transfer 32 Kbytes of data to the first requesting channel.

7.20 Write short notes on.

(i) CD (ii) BD (iii) DVD (iv) HDD