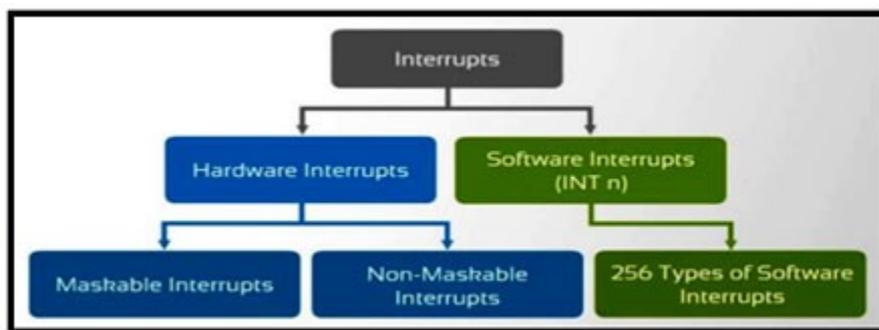


## UNIT - III

### 8086 INTERRUPTS:

An interrupt is the method of processing the microprocessor by peripheral device. An interrupt is used to cause a temporary halt in the execution of program. Microprocessor responds to the interrupt with an interrupt service routine, which is short program or subroutine that instructs the microprocessor on how to handle the interrupt.



There are two basic type of interrupt, maskable and non-maskable, nonmaskable interrupt requires an immediate response by microprocessor, it usually used for serious circumstances like power failure. A maskable interrupt is an interrupt that the microprocessor can ignore depending upon some predetermined condition defined by status register.

Interrupt can divide to five groups: 1. hardware interrupt

2. Non-maskable interrupts

3. Software interrupt

4. Internal interrupt

5. Reset

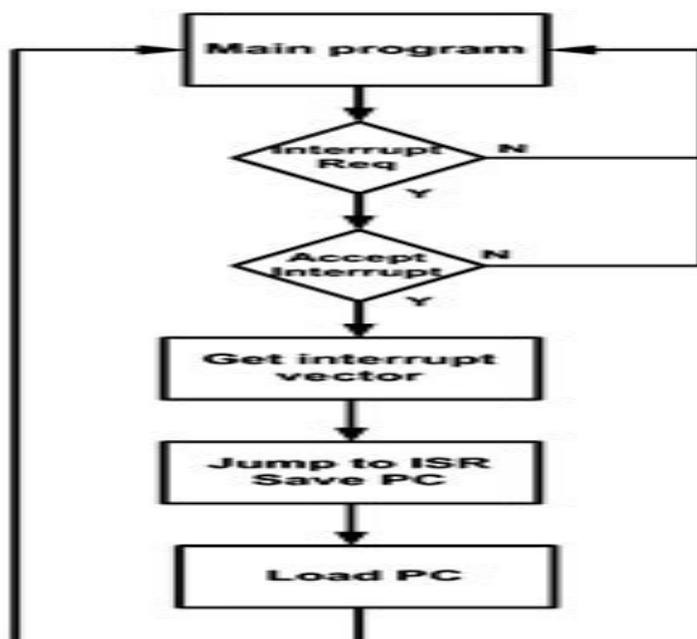
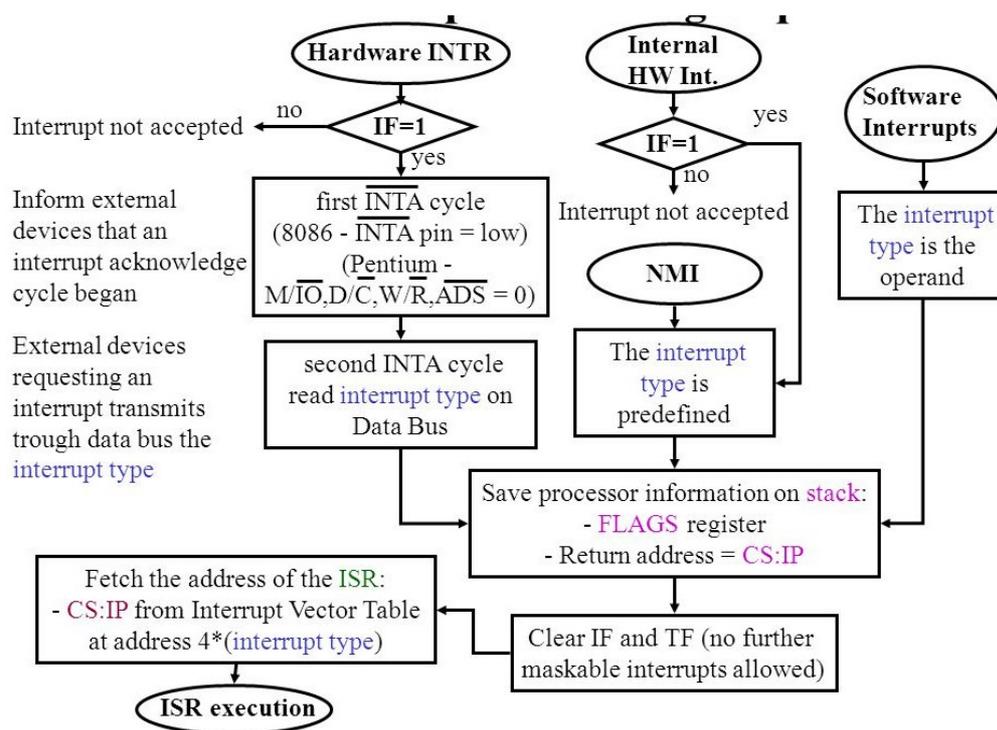
Hardware, software and internal interrupt are service on priority basis. Each interrupt is given a different priority level by assign it a type number. Type 0 identifies the highest-priority and type 255 identifies the lowest- priority interrupts. The 80x86 chips allow up to 256 vectored interrupts. This means that you can have up to 256 different sources for an interrupt and the 80x86 will directly call the service routine for that interrupt without any software processing. This is in contrast to no vectored interrupts that transfer control directly to a single interrupt service routine, regardless of the interrupt source.

The 80x86 provides a 256 entry interrupt vector table beginning at address 0:0 in memory. This is a 1K table containing 256 4-byte entries. Each entry in this table contains a segmented address that points at the interrupt service routine in memory. The lowest five types are dedicated to specific interrupts such as the divide by zero interrupt and the non maskable interrupt. The next 27 interrupt types, from 5 to 31 are High priority 3 reserved by Intel for use in future microprocessors. The upper 224 interrupt types, from 32 to 255, are available to use for hardware and software interrupts.

When an interrupt occurs (shown in figure 1), regardless of source, the 80x86 does the following:

1. The CPU pushes the flags register onto the stack.
2. The CPU pushes a far return address (segment: offset) onto the stack, segment value first.
3. The CPU determines the cause of the interrupt (i.e., the interrupt number) and fetches the four byte interrupt vector from address 0: vector\*4.
4. The CPU transfers control to the routine specified by the interrupt vector table entry

### PROCESSING OF AN INTERRUPT:



## Hardware and Software Interrupts:

### Hardware interrupts:

- ❖ The nonmaskable interrupt is generated by an external device, through a rising edge on the NMI pin
- ❖ An external device, through a high logic level on the INTR pin (the external device has to specify the interrupt number).

**Software interrupts:** Software interrupts (exceptions) using the INT instruction (followed by the interrupt number (type)).

### Interrupt Vector Table:

INTERRUPTS

	Type 32 — 255 User interrupt vectors
080H	Type 14 — 31 Reserved
040H	Type 16 Coprocessor error
03CH	Type 15 Unassigned
038H	Type 14 Page fault
034H	Type 13 General protection
030H	Type 12 Stack segment overrun
02CH	Type 11 Segment not present
028H	Type 10 Invalid task state segment
024H	Type 9 Coprocessor segment overrun
020H	Type 8 Double fault
01CH	Type 7 Coprocessor not available
018H	Type 6 Undefined opcode
014H	Type 5 BOUND
010H	Type 4 Overflow (INTO)
00CH	Type 3 1-byte breakpoint
008H	Type 2 NMI pin
004H	Type 1 Single-step
000H	Type 0 Divide error

(a)

An "**interrupt vector table**" (IVT) is a data structure that associates a list of interrupt handlers with a list of interrupt requests in a table of interrupt vectors. An entry in the interrupt vector is the address of the interrupt handler. While the concept is common across processor architectures, each IVT may be implemented in an architecture-specific fashion. For example, a dispatch table is one method of implementing an interrupt vector table.

The first 1Kbyte of memory of 8086 (00000 to 003FF) is set aside as a table for storing the starting addresses of Interrupt Service Procedures (ISP). Since 4-bytes are required for storing starting addresses of ISPs, the table can hold 256 Interrupt procedures. The starting address of an ISP is often called the Interrupt Vector or Interrupt Pointer. Therefore the table is referred as Interrupt Vector Table. In this table, IP value is put in as low word of the vector & CS is put in high vector.

### Dedicated interrupts of 8086:

The following are the various types of interrupts:  
 - Type 0 interrupts: This interrupt is also known as the divide by zero interrupt. For cases

where the quotient becomes particularly large to be placed / adjusted an error might occur.

- Type 1 interrupts: This is also known as the single step interrupt. This type of interrupt is primarily used for debugging purposes in assembly language.

- Type 2 interrupts: also known as the non-maskable NMI interrupts. These type of interrupts are used for emergency scenarios such as power failure.

- Type 3 interrupts: These type of interrupts are also known as breakpoint interrupts. When this interrupt occurs a program would execute up to its break point.

-Type 4 interrupts: Also known as overflow interrupts is generally existent after an arithmetic operation was performed.

**Interrupt Priority Structure**

Interrupt	Priority
Divide Error, INT(n),INTO	Highest
NMI	↓
INTR	
Single Step	

**Bio:**

to i use  
 Trac ers.  
 as t uch  
 boc hat  
 run nly  
 mu: her  
 BIO ling  
 CPU the  
 son igh  
 stag arly

isk,  
 key d in  
 ROM on the main board, takes control immediately after the processor is reset, including during power-up or when a hardware reset button is pressed. The BIOS initializes the hardware, finds, loads and runs the boot program (usually, but not necessarily, an OS loader), and provides basic hardware control to the operating system running on the machine, which is usually an operating system but may be a directly booting single software application.

For IBM's part, they provided all the information needed to use their BIOS fully or to directly utilize the hardware and avoid BIOS completely, when programming the early IBM PC models (prior to the PS/2). From the beginning, programmers had the choice of using BIOS or not, on a per-hardware-peripheral basis. Today, the BIOS in a new PC still supports most, if not all, of the BIOS interrupt function calls defined by IBM for the IBM

AT (introduced in 1984), along with many more newer ones, plus extensions to some of the originals (e.g. expanded parameter ranges). This, combined with a similar degree of hardware compatibility, means that most programs written for an IBM AT can still run correctly on a new PC today, assuming that the faster speed of execution is acceptable (which it typically is for all but games that use CPU-based timing). Despite the considerable limitations of the services accessed through the BIOS interrupts, they have proven extremely useful and durable to technological change.

Interrupt vector	Description																						
05h	Executed when Shift- <u>Print screen</u> is pressed, as well as when the BOUND instruction detects a bound failure.																						
<u>10h</u>	<p data-bbox="352 801 549 837">Video Services</p> <table border="1" data-bbox="387 880 1149 2141"> <thead> <tr> <th data-bbox="387 880 467 999">AH</th> <th data-bbox="467 880 1149 999">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="387 999 467 1117">00h</td> <td data-bbox="467 999 1149 1117">Set Video Mode</td> </tr> <tr> <td data-bbox="387 1117 467 1236">01h</td> <td data-bbox="467 1117 1149 1236">Set Cursor Shape</td> </tr> <tr> <td data-bbox="387 1236 467 1355">02h</td> <td data-bbox="467 1236 1149 1355">Set Cursor Position</td> </tr> <tr> <td data-bbox="387 1355 467 1473">03h</td> <td data-bbox="467 1355 1149 1473">Get Cursor Position And Shape</td> </tr> <tr> <td data-bbox="387 1473 467 1592">04h</td> <td data-bbox="467 1473 1149 1592">Get Light Pen Position</td> </tr> <tr> <td data-bbox="387 1592 467 1711">05h</td> <td data-bbox="467 1592 1149 1711">Set Display Page</td> </tr> <tr> <td data-bbox="387 1711 467 1830">06h</td> <td data-bbox="467 1711 1149 1830">Clear/Scroll Screen Up</td> </tr> <tr> <td data-bbox="387 1830 467 1948">07h</td> <td data-bbox="467 1830 1149 1948">Clear/Scroll Screen Down</td> </tr> <tr> <td data-bbox="387 1948 467 2067">08h</td> <td data-bbox="467 1948 1149 2067">Read Character and Attribute at Cursor</td> </tr> <tr> <td data-bbox="387 2067 467 2141">09h</td> <td data-bbox="467 2067 1149 2141">Write Character and Attribute at Cursor</td> </tr> </tbody> </table>	AH	Description	00h	Set Video Mode	01h	Set Cursor Shape	02h	Set Cursor Position	03h	Get Cursor Position And Shape	04h	Get Light Pen Position	05h	Set Display Page	06h	Clear/Scroll Screen Up	07h	Clear/Scroll Screen Down	08h	Read Character and Attribute at Cursor	09h	Write Character and Attribute at Cursor
AH	Description																						
00h	Set Video Mode																						
01h	Set Cursor Shape																						
02h	Set Cursor Position																						
03h	Get Cursor Position And Shape																						
04h	Get Light Pen Position																						
05h	Set Display Page																						
06h	Clear/Scroll Screen Up																						
07h	Clear/Scroll Screen Down																						
08h	Read Character and Attribute at Cursor																						
09h	Write Character and Attribute at Cursor																						

0Ah	Write Character at Cursor
0Bh	Set Border Color
0Ch	Write Graphics Pixel
0Dh	Read Graphics Pixel
0Eh	Write Character in TTY Mode
0Fh	Get Video Mode
10h	Set Palette Registers (EGA, VGA, SVGA)
11h	Character Generator (EGA, VGA, SVGA)
12h	Alternate Select Functions (EGA, VGA, SVGA)
13h	Write String
1Ah	Get or Set Display Combination Code (VGA, SVGA)
1Bh	Get Functionality Information (VGA, SVGA)
1Ch	Save or Restore Video State (VGA, SVGA)
4Fh	<a href="#">VESA BIOS Extension</a> Functions (SVGA)
11h	Returns equipment list
12h	Return <a href="#">conventional memory</a> size

13h

Low Level Disk Services

AH	Description
00h	Reset Disk Drives
01h	Check Drive Status
02h	Read Sectors
03h	Write Sectors
04h	Verify Sectors
05h	Format Track
08h	Get Drive Parameters
09h	Init Fixed Drive Parameters
0Ch	Seek To Specified Track
0D h	Reset Fixed Disk Controller
15h	Get Drive Type
16h	Get Floppy Drive Media Change Status
17h	Set Disk Type
18h	Set Floppy Drive Media Type

	41h	Extended Disk Drive (EDD) Installation Check
	42h	Extended Read Sectors
	43h	Extended Write Sectors
	44h	Extended Verify Sectors
	45h	Lock/Unlock Drive
	46h	Eject Media
	47h	Extended Seek
	48h	Extended Get Drive Parameters
	49h	Extended Get Media Change Status
	4Eh	Extended Set Hardware Configuration

14h	Serial port services											
	<table border="1"> <thead> <tr> <th>AH</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00h</td> <td>Serial Port Initialization</td> </tr> <tr> <td>01h</td> <td>Transmit Character</td> </tr> <tr> <td>02h</td> <td>Receive Character</td> </tr> <tr> <td>03</td> <td>Status</td> </tr> </tbody> </table>		AH	Description	00h	Serial Port Initialization	01h	Transmit Character	02h	Receive Character	03	Status
AH	Description											
00h	Serial Port Initialization											
01h	Transmit Character											
02h	Receive Character											
03	Status											

h	
---	--

15h

## Miscellaneous system services

AH	AL	Description
00h		Turn on cassette drive motor
01h		Turn off cassette drive motor
02h		Read data blocks from cassette
03h		Write data blocks to cassette
4Fh		Keyboard Intercept
83h		Event Wait
84h		Read Joystick
85h		Sysreq Key Callout
86h		Wait
87h		Move Block
88h		Get <a href="#">Extended Memory</a> Size
89h		Switch to Protected Mode
C0h		Get System Parameters
C1h		Get Extended BIOS Data Area Segment

C2h		Pointing Device Functions
C3h		Watchdog Timer Functions - PS/2 systems only
C4h		Programmable Option Select - <a href="#">MCA</a> bus PS/2 systems only
D8h		<a href="#">EISA</a> System Functions - EISA bus systems only
E8h	01h	Get Extended Memory Size (Newer function, since 1994). Gives results for memory size above 64 Mb.
E8h	20h	Query System Address Map. The information returned from <a href="#">E820</a> supersedes what is returned from the older AX=E801h and AH=88h interfaces.

<u>16h</u>	Keyboard services	
	<b>AH</b>	<b>Description</b>
	00h	Read Character
	01h	Read Input Status
	02h	Read Keyboard Shift Status
	05h	Store Keystroke in Keyboard Buffer
	10h	Read Character Extended
	11h	Read Input Status Extended

	<table border="1"> <tr> <td>h</td> <td></td> </tr> <tr> <td>12h</td> <td>Read Keyboard Shift Status Extended</td> </tr> </table>	h		12h	Read Keyboard Shift Status Extended				
h									
12h	Read Keyboard Shift Status Extended								
17h	<p>Printer services</p> <table border="1"> <thead> <tr> <th>AH</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00h</td> <td>Print Character to Printer</td> </tr> <tr> <td>01h</td> <td>Initialize Printer</td> </tr> <tr> <td>02h</td> <td>Check Printer Status</td> </tr> </tbody> </table>	AH	Description	00h	Print Character to Printer	01h	Initialize Printer	02h	Check Printer Status
AH	Description								
00h	Print Character to Printer								
01h	Initialize Printer								
02h	Check Printer Status								
18h	<p>Execute Cassette BASIC: Very early true IBM computers contain Microsoft Cassette BASIC in the ROM, to be started by this routine in the event of a failure to boot from disk (called by the BIOS). On virtually all clones and later models in the PC line from IBM, which lack BASIC in ROM, this interrupt typically displays a message such as "No ROM BASIC" and halts.</p>								
19h	<p>After POST this interrupt is used by BIOS to load the operating system. A program can call this interrupt to reboot the computer (but must ensure that hardware interrupts or DMA operations will not cause the system to hang or crash during either the reinitialization of the system by BIOS or the boot process).</p>								
1Ah	<p>Real Time Clock Services</p> <table border="1"> <thead> <tr> <th>AH</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00h</td> <td>Read RTC</td> </tr> </tbody> </table>	AH	Description	00h	Read RTC				
AH	Description								
00h	Read RTC								

	01 h	Set RTC
	02 h	Read RTC Time
	03 h	Set RTC Time
	04 h	Read RTC Date
	05 h	Set RTC Date
	06 h	Set <a href="#">RTC Alarm</a>
	07 h	Reset RTC Alarm

1Ah	<a href="#">PCI</a> Services - implemented by BIOSes supporting PCI 2.0 or later	
	<b>AX</b>	<b>Description</b>
	B101h	PCI Installation Check
	B102h	Find PCI Device
	B103h	Find PCI Class Code
	B106h	PCI Bus-Specific Operations
	B108h	Read Configuration Byte

	<table border="1"> <tr> <td>B109h</td> <td>Read Configuration Word</td> </tr> <tr> <td>B10Ah</td> <td>Read Configuration Dword</td> </tr> <tr> <td>B10Bh</td> <td>Write Configuration Byte</td> </tr> <tr> <td>B10Ch</td> <td>Write Configuration Word</td> </tr> <tr> <td>B10Dh</td> <td>Write Configuration Dword</td> </tr> <tr> <td>B10Eh</td> <td>Get IRQ Routine Information</td> </tr> <tr> <td>B10Fh</td> <td>Set PCI IRQ</td> </tr> </table>	B109h	Read Configuration Word	B10Ah	Read Configuration Dword	B10Bh	Write Configuration Byte	B10Ch	Write Configuration Word	B10Dh	Write Configuration Dword	B10Eh	Get IRQ Routine Information	B10Fh	Set PCI IRQ
B109h	Read Configuration Word														
B10Ah	Read Configuration Dword														
B10Bh	Write Configuration Byte														
B10Ch	Write Configuration Word														
B10Dh	Write Configuration Dword														
B10Eh	Get IRQ Routine Information														
B10Fh	Set PCI IRQ														
1Bh	Ctrl-Break handler - called by INT 09 when Ctrl- <a href="#">Break</a> has been pressed														
1Ch	Timer tick handler - called by INT 08														
1Dh	Not to be called; simply a pointer to the VPT (Video Parameter Table), which contains data on video modes														
1Eh	Not to be called; simply a pointer to the DPT (Diskette Parameter Table), containing a variety of information concerning the diskette drives														
1Fh	Not to be called; simply a pointer to the VGCT (Video Graphics Character Table), which contains the data for ASCII characters 80h to FFh														
41h	Address pointer: FDPT = Fixed Disk Parameter Table (1st hard drive)														
46h	Address pointer: FDPT = Fixed Disk Parameter Table (2nd hard drive)														
4Ah	Called by RTC for alarm														

## MEMORY AND I/O INTERFACING:

## I/O Interface

### Introduction:

Any application of a microprocessor based system requires the transfer of data between external circuitry to the microprocessor and microprocessor to the external circuitry. User can give information to the microprocessor based system using keyboard and user can see the result or output information from the microprocessor based system with the help of display device. The transfer of data between keyboard and microprocessor, and microprocessor and display device is called input/output data transfer or I/O data transfer. This data transfer is done with the help of I/O ports.

### Input port:

It is used to read data from the input device such as keyboard. The simplest form of input port is a buffer. The input device is connected to the microprocessor through buffer, as shown in the fig.1. This buffer is a tri-state buffer and its output is available only when enable signal is active. When microprocessor wants to read data from the input device (keyboard), the control signals from the microprocessor activates the buffer by asserting enable input of the buffer. Once the buffer is enabled, data from the input device is available on the data bus. Microprocessor reads this data by initiating read command.

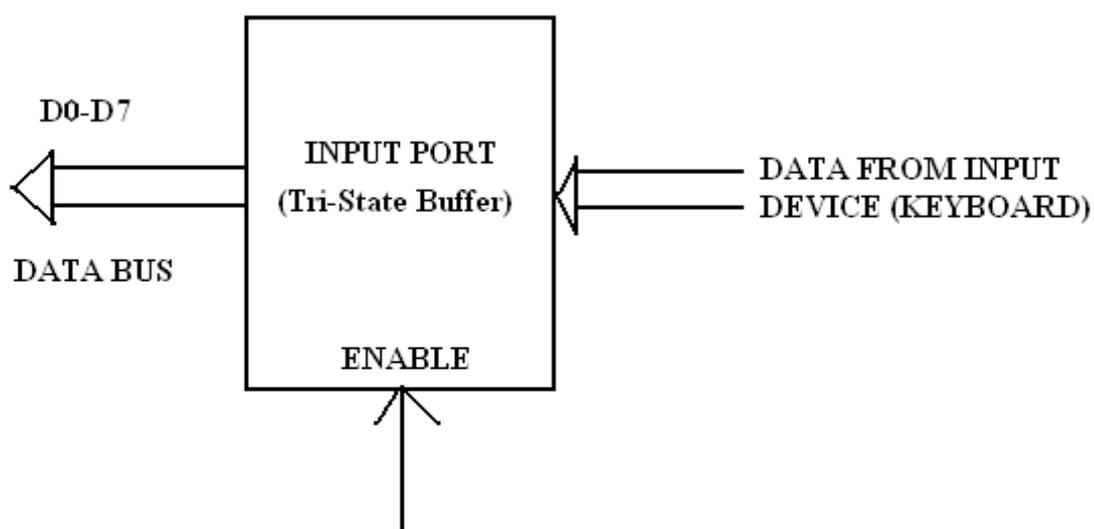


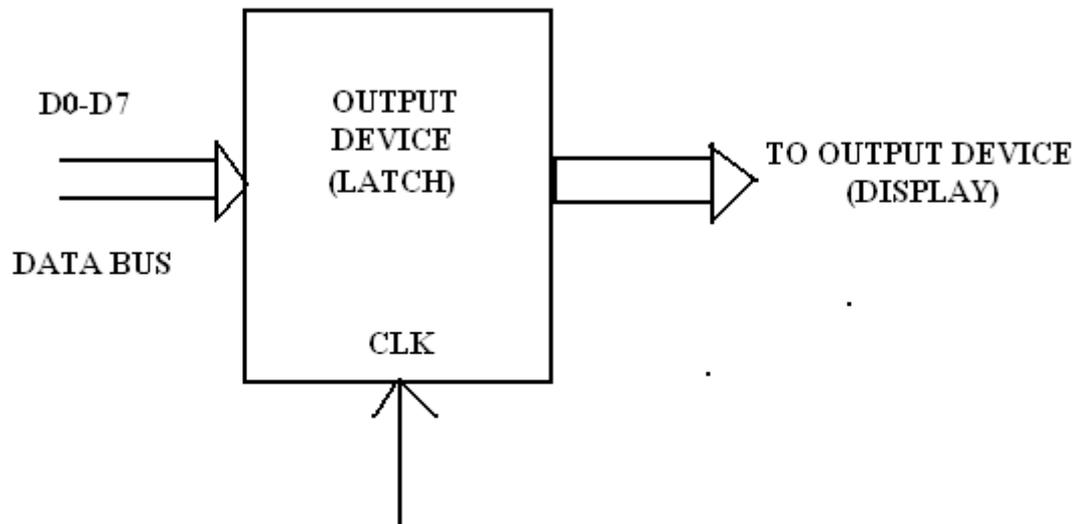
FIG.1 INPUT PORT

It is used to read data from the input device such as keyboard. The simplest form of input port is a buffer. The input device is connected to the microprocessor through buffer, as shown in the fig.1. This buffer is a tri-state buffer and its output is available only when enable signal is active. When microprocessor wants to read data from the input device (keyboard), the control signals from the microprocessor activates the buffer by asserting enable input of the buffer. Once the buffer is enabled, data from the input device is available on the data bus. Microprocessor reads this data by initiating read command.

### Output port:

It is used to send data to the output device such as display from the microprocessor. The simplest form of output port is a latch. The output device is connected to the microprocessor through latch, as shown in the fig.2. When microprocessor wants to send data to the output device is puts the data on the data bus and activates the clock signal of

the latch, latching the data from the data bus at the output of latch. It is then available at the output of latch for the output device.



**FIG.2 OUTPUT PORT**

There are three different ways that the data transfer can take place. They are

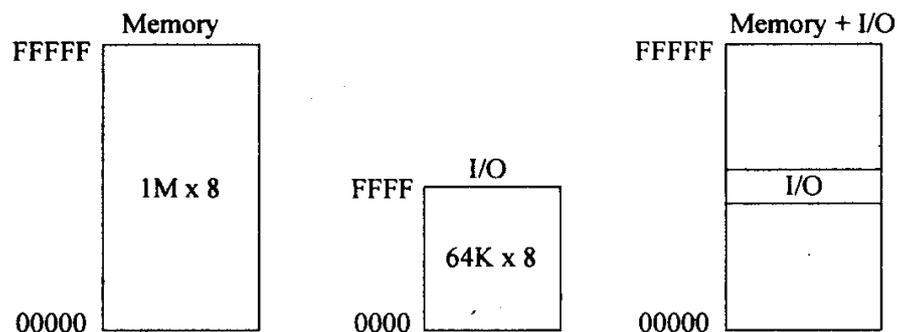
- (1) Program controlled I/O
- (2) Interrupt Program Controlled I/O
- (3) Hardware controlled I/O

In program controlled I/O data transfer scheme the transfer of data is completely under the control of the microprocessor program. In this case an I/O operation takes place only when an I/O transfer instruction is executed. In an interrupt program controlled I/O an external device indicates directly to the microprocessor its readiness to transfer data by a signal at an interrupt input of the microprocessor. When microprocessor receives this signal the control is transferred to ISS (Interrupt service subroutine) which performs the data transfer. Hardware controlled I/O is also known as direct memory access DMA. In this case the data transfer takes place directly between an I/O device and memory but not through microprocessors. Microprocessor only initializes the process of data transfer by indicating the starting address and the number of words to be transferred. The instruction .set of any microprocessor contains instructions that transfer information to an I/O device and to read information from an I/O device.

In 8086 we have IN, OUT instructions for this purpose. OUT instruction transfers information to an I/O device whereas IN instruction is used to read information from an I/O device. Both the instructions perform the data transfer using accumulator AL or AX. The I/O address is stored in register DX. The port number is specified along with IN or OUT instruction. The external I/O interface decodes to find the address of the I/O device. The 8 bit fixed port number appears on address bus A0 - A7 with A8 - A15 all zeros. The address connections above A15 are undefined for an I/O instruction. The 16 bit variable port number appears on address connections A0 - A15. The above notation indicates that first 256 I/O port addresses 00 to FF are accessed by both the fixed and variable I/O instructions. The I/O addresses from 0000 to FFFF are accessed by the variable I/O address.

I/O devices can be interfaced to the microprocessors using two methods. They are I/O mapped I/O and memory mapped I/O. The I/O mapped I/O is also known as isolated I/O

or direct I/O. In I/O mapped I/O the IN and OUT instructions transfer data between the accumulator or memory and I/O device. In memory mapped I/O the instruction that refers memory can perform the data transfer.



(a) Isolated I/O map  
Memory and I/O map of 8086

(b) Memory map I/O

I/O mapped I/O is the most commonly used I/O transfer technique. In this method I/O locations are placed separately from memory. The addresses for isolated I/O devices are separate from memory. Using this method user can use the entire memory. This method allows data transfer only by using instructions IN, OUT. The pins M/IO and W/R are used to indicate I/O read or an I/O write operations. The signals on these lines indicate that the address on the address bus is for I/O devices. Memory mapped I/O does not use the IN, OUT instruction it uses only the instruction that transfers data between microprocessor and memory.

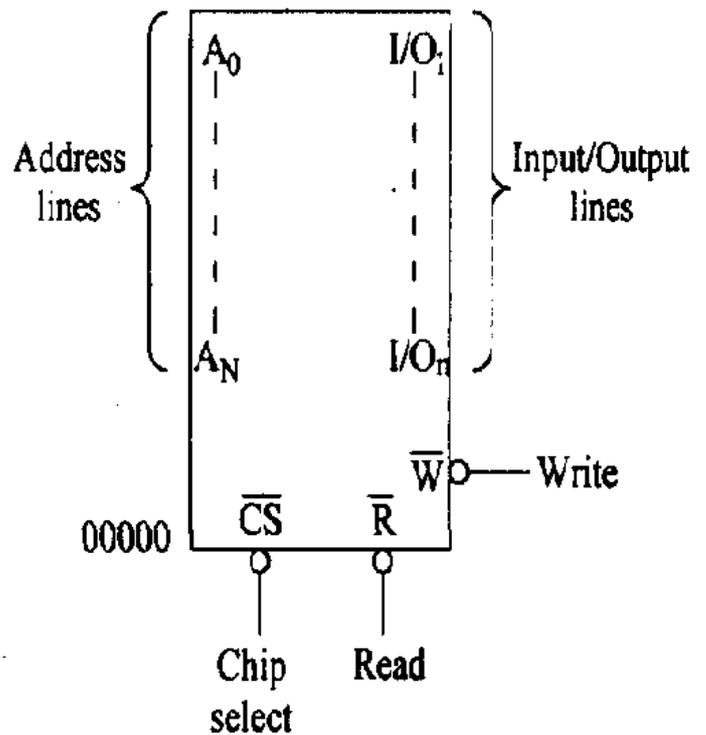
A memory mapped I/O device is treated as memory location. The disadvantage in this system is the overall memory is reduced. The advantage of this system is that any memory transfer instruction can be used for data transfer and control signals like I/O read and I/O write are not necessary which simplify the hardware.

### Memory interfacing

Memory is an integral part of a microcomputer system. There are two main types of memory.

- (i) **Read only memory (ROM)**: As the name indicates this memory is available only for reading purpose. The various types available under this category are PROM, EPROM, EEPROM which contain system software and permanent system data.
- (ii) **Random Access memory (RAM)**: This is also known as Read Write Memory. It is a volatile memory. RAM contains temporary data and software programs generally for different applications.

While executing particular task it is necessary to access memory to get instruction codes and data stored in memory. Microprocessor initiates the necessary signals when read or write operation is to be performed. Memory device also requires some signals to perform read and write operations using various registers. To do the above job it is necessary to have a device and a circuit, which performs this task is known as interfacing device and as this is involved with memory it is known as memory interfacing device. The basic concepts of memory interfacing involve three different tasks. The microprocessor should be able to read from or write into the specified register. To do this it must be able to select the required chip, identify the required register and it must enable the appropriate buffers.



### Simple memory device

Any memory device must contain address lines and Input, output lines, selection input, control input to perform read or write operation. All memory devices have address inputs that select memory location within the memory device. These lines are labeled as  $A_0$  .....  $A_N$ . The number of address lines indicates the total memory capacity of the memory device. A 1K memory requires 10 address lines  $A_0$ - $A_9$ . Similarly a 1MB requires 20 lines  $A_0$ - $A_{19}$  (in the case of 8086). The memory devices may have separate I/O lines or a common set of bidirectional I/O lines.

Using these lines data can be transferred in either direction. Whenever output buffer is activated the operation is read whenever input buffers are activated the operation is write. These lines are labeled as  $I/O_1$ ,.....  $I/O_m$  or  $DO$  ..... $D_n$ . The size of a memory location is dependent upon the number of data bits. If the number of data lines is eight  $D_0$  -  $D_7$  then 8 bits or 1 byte of data can be stored in each location. Similarly if numbers of data bits are 16 ( $D_0$  -  $D_{15}$ ) then the memory size is 2 bytes. For Example 2K x 8 indicates there are 2048 memory locations and each memory location can store 8 bits of data.

Memory devices may contain one or more inputs which are used to select the memory device or to enable the memory device. This pin is denoted by CS (Chip select) or CE (Chip enable). When this pin is at logic '0' then only the memory device performs a read or a write operation. If this pin is at logic '1' the memory chip is disabled. If there are more than one CS input then all these pins must be activated to perform read or write operation.

All memory devices will have one or more control inputs. When ROM is used we find OE output enable pin which allows data to flow out of the output data pins. To perform this

task both CS and OE must be active. A RAM contains one or two control inputs. They are R /W or RD and WR. If there is only one input R/W then it performs read operation when R/W pin is at logic 1. If it is at logic 0 it performs write operation. Note that this is possible only when CS is also active.

### **Memory Interface using RAMS, EPROMS and EEPROMS:**

#### **Semiconductor Memory Interfacing:**

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

#### **Static RAM Interfacing:**

The semiconductor RAMs are of broadly two types-static RAM and dynamic RAM. The semiconductor memories are organised as two dimensional arrays of memory locations. For example, 4K x 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. 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 ( CS ).

The output of the decoding circuit is connected with the 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'.

2. Connect available memory address lines of memory chips with those of the microprocessor and also connect the memory RD and 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 addresslines of the microprocessor, BHE and A0 are used for decoding the required chip select signals for the odd and even memory banks. CS of memory is derived from the O/P of the decoding circuit.

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. 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 minimize the required hardware. Let us now consider a few example problems on memory interfacing with 8086.

**Problem 5.1**

Interface two  $4K \times 8$  EPROMS and two  $4K \times 8$  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								$8K \times 8$												
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								$8K \times 8$												
FC000H	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

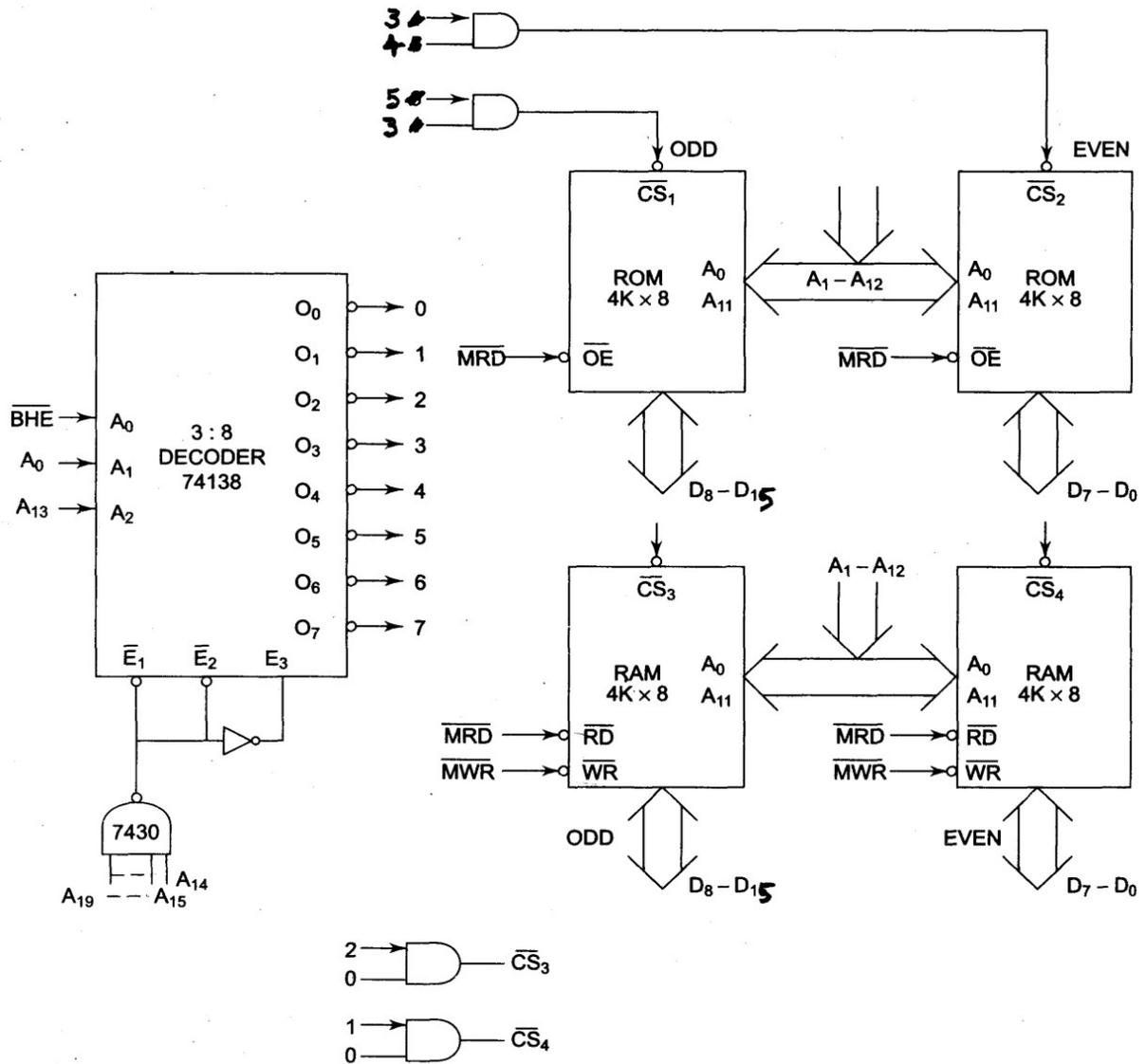
Total 8K bytes of EPROM need 13 address lines  $A_0 - A_{12}$  (since  $2^{13} = 8K$ ). Address lines  $A_{13} - A_{19}$  are used for decoding to generate the chip select. The  $\overline{BHE}$  signal goes low when a transfer is at odd address or higher byte of data is to be accessed. Let us assume 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  $4K \times 8$  memory chips.

The two  $4K \times 8$  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 IP → Address/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



**Fig. 5.1** Interfacing Problem 5.1

**Problem 5.2**

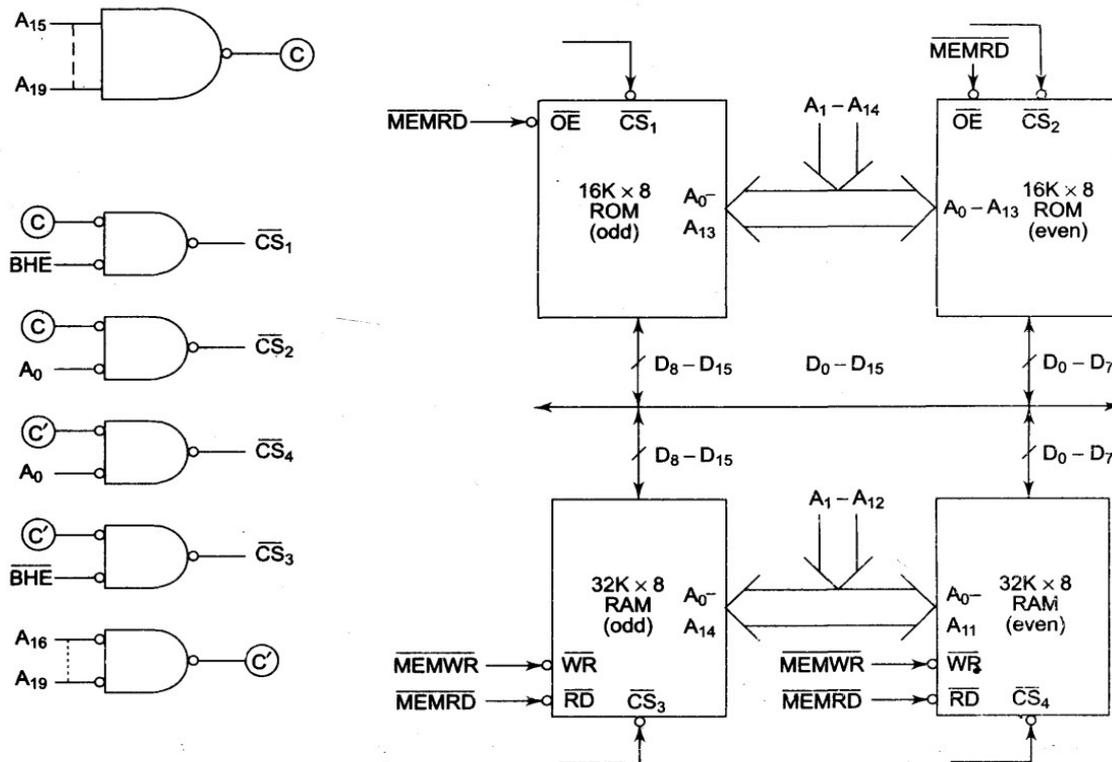
Design an interface between 8086 CPU and two chips of 16K × 8 EPROM and two chips of 32K × 8 RAM. Select the starting address of EPROM suitably. The RAM address must start at 00000H.

**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 <sub>19</sub>	A <sub>18</sub>	A <sub>17</sub>	A <sub>16</sub>	A <sub>15</sub>	A <sub>14</sub>	A <sub>13</sub>	A <sub>12</sub>	A <sub>11</sub>	A <sub>10</sub>	A <sub>09</sub>	A <sub>08</sub>	A <sub>07</sub>	A <sub>06</sub>	A <sub>05</sub>	A <sub>04</sub>	A <sub>03</sub>	A <sub>02</sub>	A <sub>01</sub>	A <sub>00</sub>
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

**Problem 5.3**

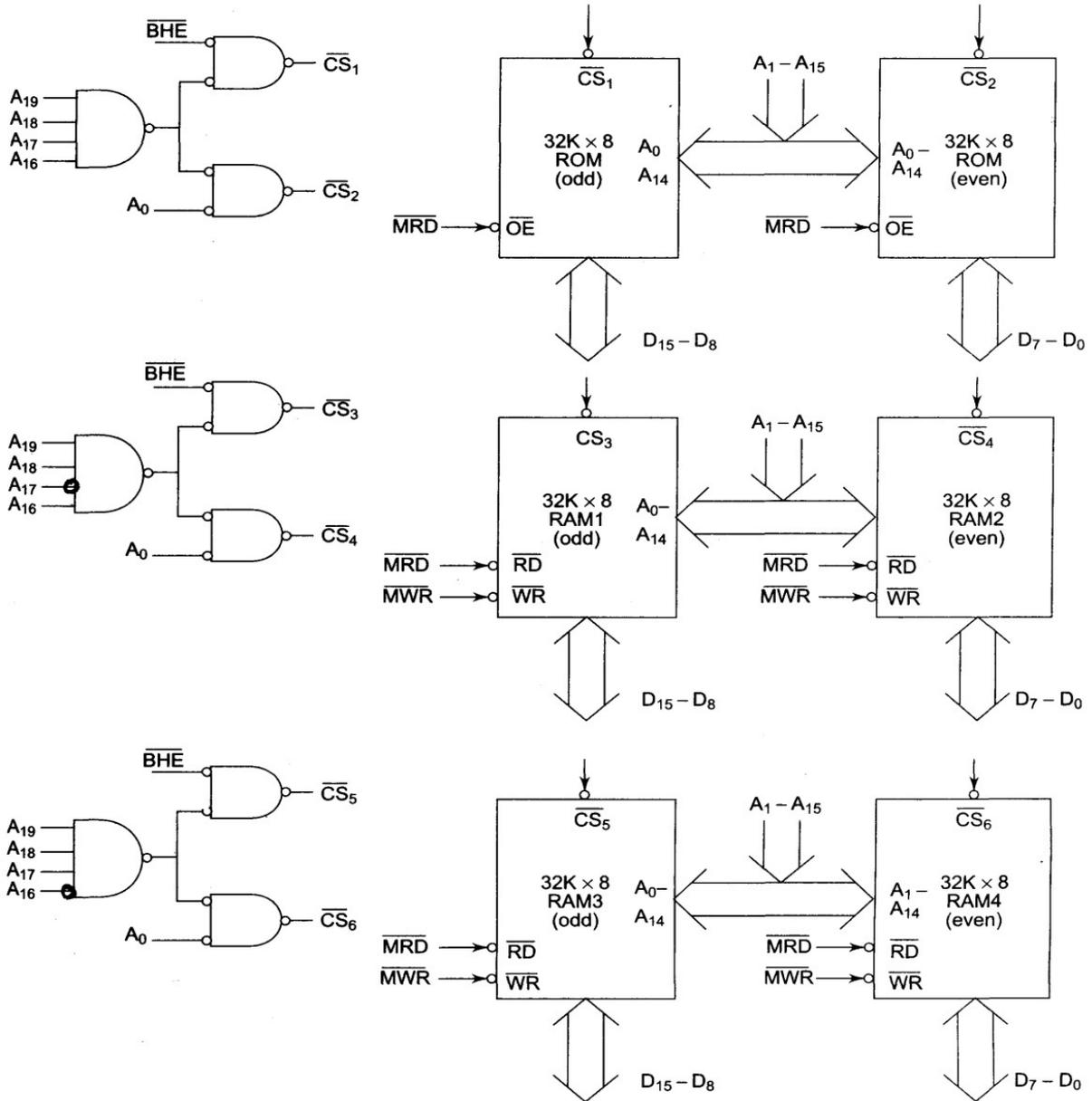
It is required to interface two chips of 32K × 8 ROM and four chips of 32K × 8 RAM with 8086, according to the following map.

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.



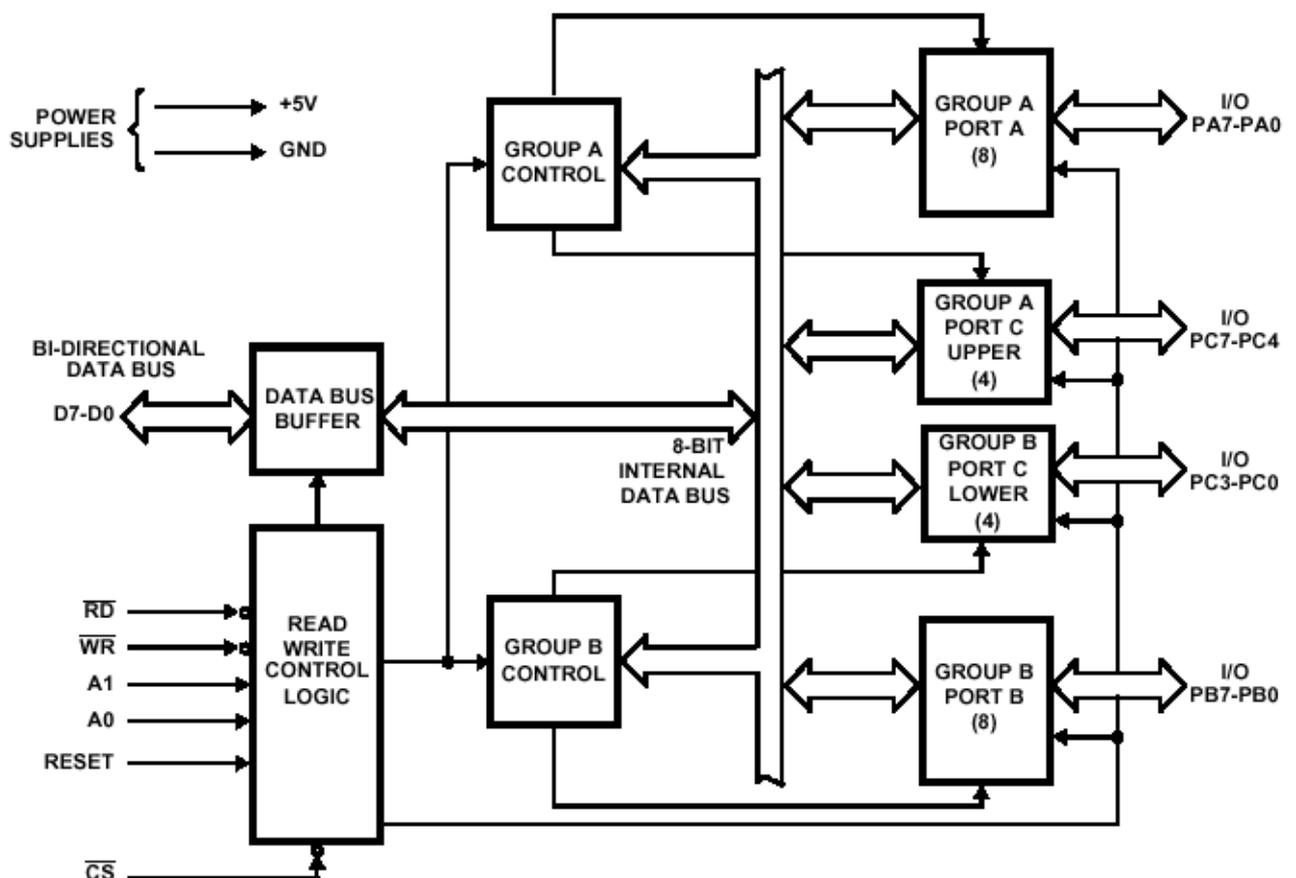
**Fig. 5.3** Interfacing Problem 5.3

## Features:

- 3 8-bit IO ports PA, PB, PC
- PA can be set for Modes 0, 1, 2. PB for 0,1 and PC for mode 0 and for BSR. Modes 1 and 2 are interrupt driven.
- PC has 2 4-bit parts: PC upper (PCU) and PC lower (PCL), each can be set independently for I or O. Each PC bit can be set/reset individually in BSR mode.
- PA and PCU are Group A (GA) and PB and PCL are Group B (GB)
- Address/data bus must be externally demux'd.
- TTL compatible.
- Improved dc driving capability

The 8255A is a programmable peripheral interface (PPI) device designed for use in Intel microcomputer systems. Its function is that of a general purposes I/O component to Interface peripheral equipment to the microcomputer system bush. The functional configuration of the 8255A is programmed by the systems software so that normally no external logic is necessary to interface peripheral devices or structures.

### ARCHITECTURE OF 8255 PPI:

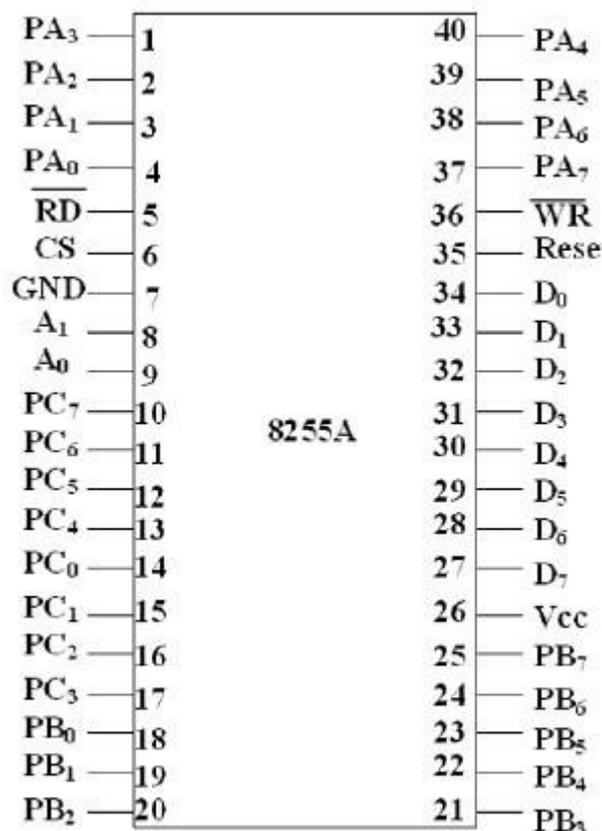


The port A lines are identified by symbols PA0-PA7 while the port C lines are identified as PC4-PC7 similarly. Group B contains an 8-bit port B, containing lines PB0- PB7 and a 4-bit port C with lower bits PC0-PC3. The port C upper and port C lower can be used in combination as an 8-bit port C. Both the port Cs is 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.

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 transfer of both data and control words. RD, WR, A1, A0 and RESET are the inputs, provided by the microprocessor to 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.

**Pin** .....



**8255A Pin Configuration**

- ↵ The port A lines are identified by symbols PA0-PA7 while the port C lines are
- ↵ Identified as PC4-PC7. Similarly, Group B contains an 8-bit port B, containing lines PB0-PB7 and a 4-bit port C with lower bits PC0- PC3. The port C upper and port C lower can be used in combination as an 8-bit port C.
- ↵ Both the port C is assigned the same address. Thus one may have either three
  - 8-bit I/O ports or two 8-bit and two 4-bit 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 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.
- ↵ RD,WR, A1, A0 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 description of 8255 is briefly presented as follows:

- ↵ **PA7-PA0:** These are eight port A lines that acts as either latched output or buffered input lines depending upon the control word loaded into the control word register.
- ↵ **PC7-PC4:** 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 mode2.
- ↵ **PC3-PC0:** These are the lower port C lines; other details are the same as PC7-PC4 lines.
- ↵ **PB0-PB7:** 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.
- ↵ **RD:** This is the input line driven by the microprocessor and should be low to indicate read operation to 8255.
- ↵ **WR:** This is an input line driven by the microprocessor. A low on this line indicates write operation.
- ↵ **CS:** This is a chip select line. If this line goes low, it enables the 8255 to respond to RD and WR signals, otherwise RD and WR signal are neglected.
- ↵ **D0-D7:** These are the data bus lines those carry data or control word to/from the
  - microprocessor.
- ↵ **RESET:**Logic high on this line clears the control word register of 8255. All ports

are set as input ports by default after reset.

- ↯ **A1-A0:** These are the address input lines and are driven by the microprocessor.
- ↯ These lines A1-A0 with RD, WR and CS from 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 table below.

In case of 8086 systems, if the 8255 is to be interfaced with lower order data bus, the A0 and A1 pins of 8255 are connected with A1 and A2 respectively.

$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{CS}}$	A <sub>1</sub>	A <sub>0</sub>	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

$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{CS}}$	A <sub>1</sub>	A <sub>0</sub>	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

$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{CS}}$	A <sub>1</sub>	A <sub>0</sub>	Function
X	X	1	X	X	Data bus tristated
1	1	0	X	X	Data bus tristated

### Control Word Register

### Operational modes of 8255

There are two basic operational modes of 8255:

- Bit set/reset mode (BSR mode).
- Input/Output mode (I/O mode).

The two modes are selected on the basis of the value present at the D<sub>7</sub> bit of the control word register. When D<sub>7</sub> = 1, 8255 operates in I/O mode, and when D<sub>7</sub> = 0, it operates in the BSR mode.

### Bit set/reset (BSR) mode

The Bit Set/Reset (BSR) mode is applicable to port C only. Each line of port C (PC<sub>0</sub> - PC<sub>7</sub>) can be set/reset by suitably loading the control word register. BSR mode and I/O mode are independent and selection of BSR mode does not affect the operation of other ports in I/O mode.



### 8255 BSR mode

- D<sub>7</sub> bit is always 0 for BSR mode.
- Bits D<sub>6</sub>, D<sub>5</sub> and D<sub>4</sub> are don't care bits.
- Bits D<sub>3</sub>, D<sub>2</sub> and D<sub>1</sub> are used to select the pin of Port C.
- Bit D<sub>0</sub> is used to set/reset the selected pin of Port C.

Selection of port C pin is determined as follows:

B3	B2	B1	Bit/pin of port C selected
0	0	0	PC <sub>0</sub>
0	0	1	PC <sub>1</sub>
0	1	0	PC <sub>2</sub>
0	1	1	PC <sub>3</sub>
1	0	0	PC <sub>4</sub>
1	0	1	PC <sub>5</sub>
1	1	0	PC <sub>6</sub>
1	1	1	PC <sub>7</sub>

As an example, if it is needed that PC<sub>5</sub> be set, then in the control word,

1. Since it is BSR mode, **D<sub>7</sub> = '0'**.
2. Since D<sub>4</sub>, D<sub>5</sub>, D<sub>6</sub> are not used, assume them to be '0'.
3. PC<sub>5</sub> has to be selected, hence, **D<sub>3</sub> = '1', D<sub>2</sub> = '0', D<sub>1</sub> = '1'**.
4. PC<sub>5</sub> has to be set, hence, **D<sub>0</sub> = '1'**.

Thus, as per the above values, 0B (Hex) will be loaded into the Control Word Register (CWR).

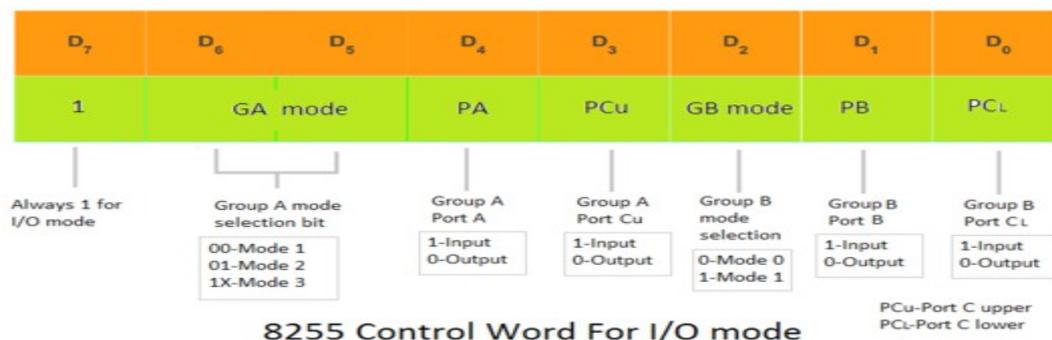
D	D	D	D	D	D	D	D0
7	6	5	4	3	2	1	
0	0	0	0	1	0	1	1

### Input/Output mode

This mode is selected when  $D_7$  bit of the Control Word Register is 1. There are three I/O modes.

1. Mode 0 - Simple I/O
2. Mode 1 - Strobed I/O
3. Mode 2 - Strobed Bi-directional I/O

### Control Word Format



### I/O Control Word Format

- $D_0, D_1, D_3, D_4$  are assigned for port C lower, port B, port C upper and port A respectively. When these bits are 1, the corresponding port acts as an input port. For e.g., if  $D_0 = D_4 = 1$ , then lower port C and port A act as input ports. If these bits are 0, then the corresponding port acts as an output port. For e.g., if  $D_1 = D_3 = 0$ , then port B and upper port C act as output ports.
- $D_2$  is used for mode selection of Group B (port B and lower port C). When  $D_2 = 0$ , mode 0 is selected and when  $D_2 = 1$ , mode 1 is selected.
- $D_5$  &  $D_6$  are used for mode selection of Group A ( port A and upper port C). The selection is done as follows:

$D_6$	$D_5$	Mode
0	0	0
0	1	1
1	X	2

- As it is I/O mode,  $D_7 = 1$ .

For example, if port B and upper port C have to be initialized as input ports and lower port C and port A as output ports (all in mode 0):

1. Since it is an I/O mode,  $D_7 = 1$ .
2. Mode selection bits,  $D_2, D_5, D_6$  are all 0 for mode 0 operation.

3. Port B and upper port C should operate as Input ports, hence,  $D_1 = D_3 = 1$ .
4. Port A and lower port C should operate as Output ports, hence,  $D_4 = D_0 = 0$ .

Hence, for the desired operation, the control word register will have to be loaded with "10001010" = 8A (hex).

#### A) Mode 0 - simple I/O

In this mode, the ports can be used for simple I/O operations without handshaking signals. Port A, port B provide simple I/O operation. The two halves of port C can be either used together as an additional 8-bit port, or they can be used as individual 4-bit ports. Since the two halves of port C are independent, they may be used such that one-half is initialized as an input port while the other half is initialized as an output port.

The input/output features in mode 0 are as follows:

1. Output ports are latched.
2. Input ports are buffered, not latched.
3. Ports do not have handshake or interrupt capability.
4. With 4 ports, 16 different combinations of I/O are possible.

#### **Mode 0 - input mode**

- In the input mode, the 8255 gets data from the external peripheral ports and the CPU reads the received data via its data bus.
- The CPU first selects the 8255 chip by making CS low. Then it selects the desired port using  $A_0$  and  $A_1$  lines.
- The CPU then issues an RD signal to read the data from the external peripheral device via the system data bus.

#### **Mode 0 - output mode**

- In the output mode, the CPU sends data to 8255 via system data bus and then the external peripheral ports receive this data via 8255 port.
- CPU first selects the 8255 chip by making CS low. It then selects the desired port using  $A_0$  and  $A_1$  lines.
- CPU then issues a WR signal to write data to the selected port via the system data bus. This data is then received by the external peripheral device connected to the selected port.

**b) Mode 1: ( Strobed input/output mode )** In this mode the handshaking control the input and output action of the specified port. Port C lines PC0-PC2, provide strobe or handshake lines for port B. This group which includes port B and PC0-PC2 is called as group B for Strobed data input/output. Port C lines PC3-PC5 provides strobe lines for portA. This group including port A and PC3-PC5 from group A. Thus port C is utilized for generating handshake signals.

The salient features of mode 1 are listed as follows:

1. Two groups – group A and group B are available for strobed data transfer.
2. Each group contains one 8-bit data I/O port and one 4-bit control/data port.
3. The 8-bit data port can be either used as input and output port. The inputs and outputs both are latched.
4. Out of 8-bit port C, PC0-PC2 are used to generate control signals for port B and PC3-PC5 are used to generate control signals for port A. the lines PC6, PC7 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):**

- **STB** (Strobe input) – If this lines 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 latches, i.e. it works as an acknowledgement. IBF is set by a low on STB and is reset by the rising edge of 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 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 PC4 (INTEA) or PC2 (INTEB) as shown in fig.
- INTR is reset by a falling edge of RD input. Thus an external input device can be request the service of the processor by putting the data on the bus and sending the strobe signal.

**Output control signal definitions (mode 1):**

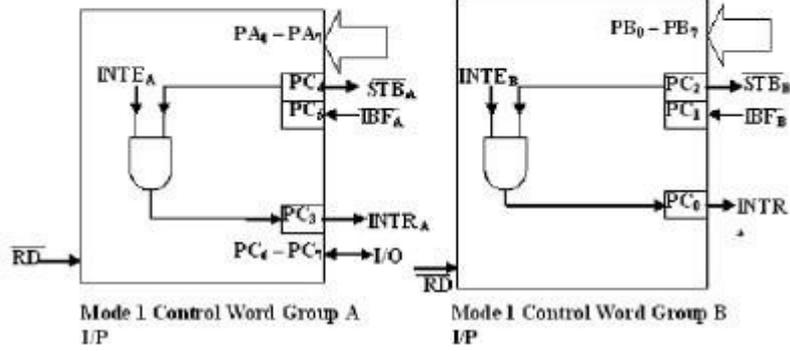
- **OBF** (Output buffer full) – This status signal, whenever falls to low, indicates that CPU has written data to the specified output port. The OBF flip-flop will beset by a rising edge of WR signal and reset by a low going edge at the ACK input.
- **ACK** (Acknowledge input) – ACK signal acts as an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

Input control signal definitions in Mode 1

1	0	1	0	10	X	X	X
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>

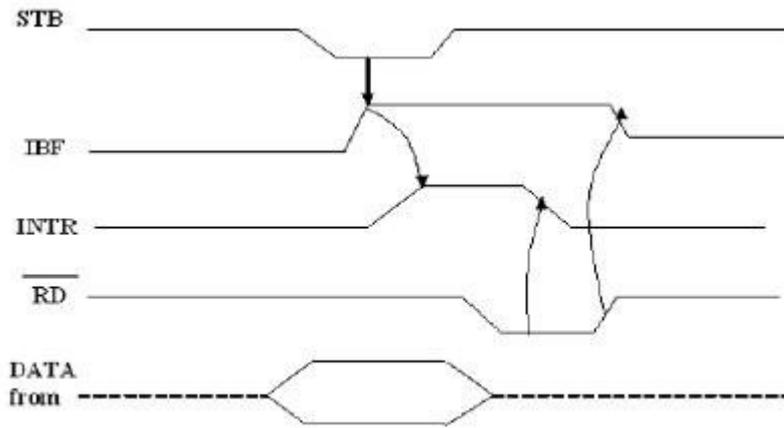
1	X	X	X	X	1	1	X
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>

1 - Input  
0 - Output  
For PC<sub>5</sub>-PC<sub>7</sub>

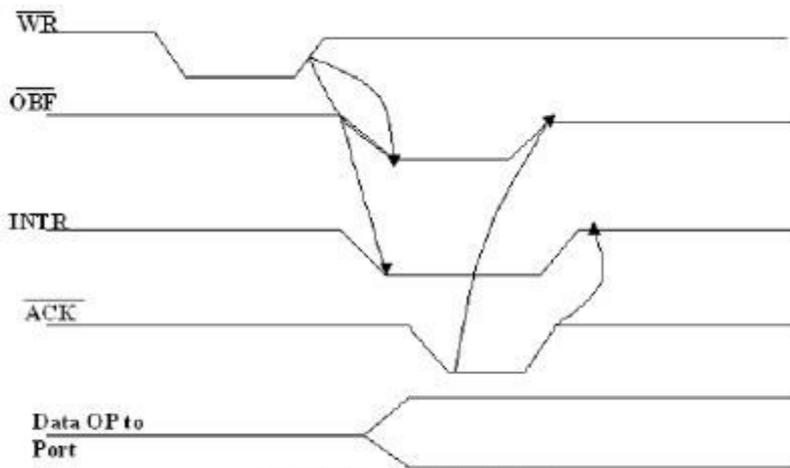


Mode 1 Control Word Group A  
I/P

Mode 1 Control Word Group B  
I/P



Mode 1 Strobed Input Data Transfer



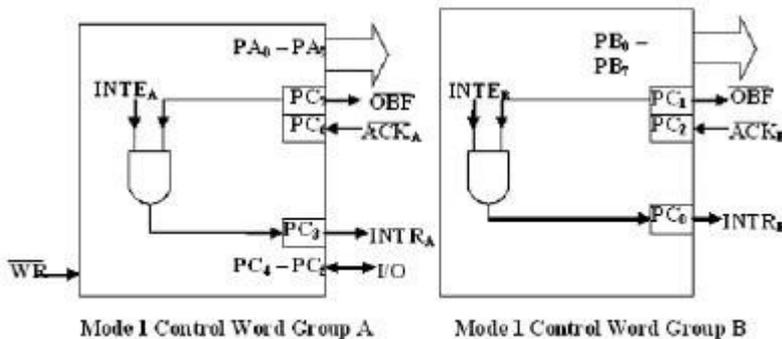
Mode 1 Strobed Data Output

Output control signal definitions Mode 1

1	0	1	0	1/0	X	X	X
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>

1	X	X	X	X	1	0	X
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>

1 - Input  
0 - Output  
For PC<sub>4</sub>-PC<sub>5</sub>



c) **Mode 2 (Strobed bidirectional I/O):** This mode of operation of 8255 is also called as strobed bidirectional I/O. This mode of operation provides 8255 with additional features for communicating with a peripheral device on an 8-bit databus. 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.

In this mode, 8255 is a bidirectional 8-bit port with handshake signals. The Rd and WR signals decide whether the 8255 is going to operate as an input port or output port.

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.( PC2 - PC0 )
4. Inputs and outputs are both latched.
5. The 5-bit control port C (PC3-PC7) 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:**

↵ **OBF** (Output buffer full) - This signal, when falls to low level, indicates that the CPU has written data to port A.

↵ **ACK** (Acknowledge) This control input, when falls to logic low level, Acknowledges that the previous data byte is received by the destination and next byte may be sent by the processor. This signal enables the internal tri state buffers to send the next data byte on port A.

↵ **INTE1** ( A flag associated with OBF ) This can be controlled by bit t/reset mode with PC6.

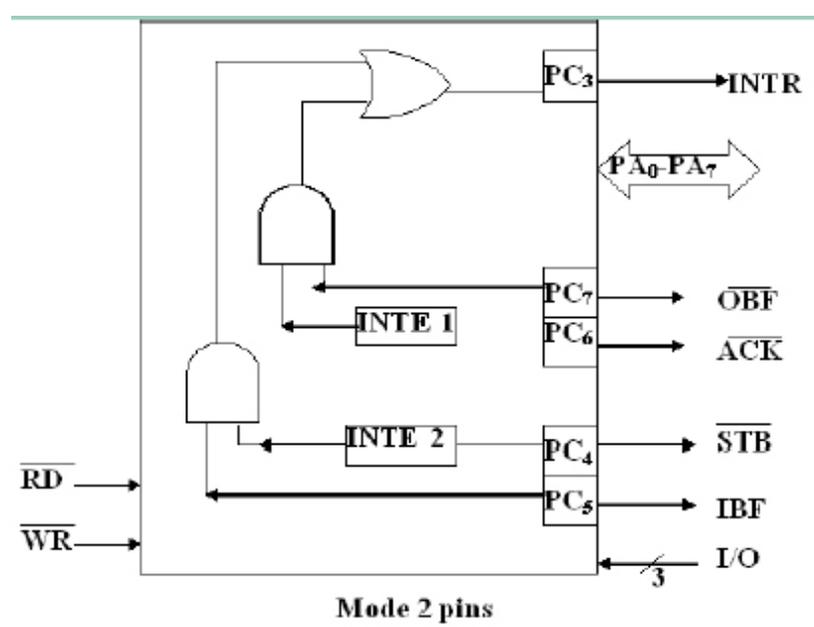
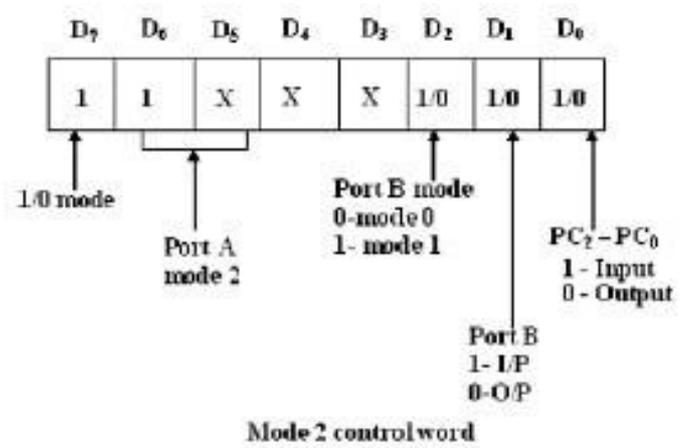
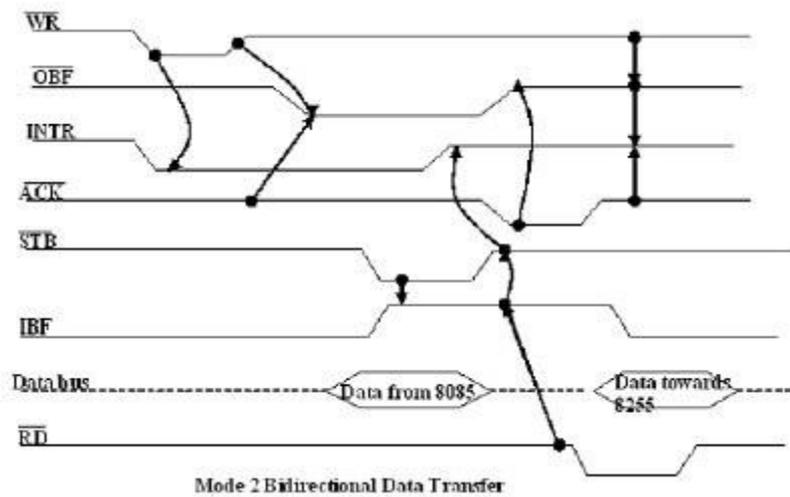
#### **Control signals for input operations:**

↵ **STB** (Strobe input)a low on this line is used to strobe in the data into the input Latches of 8255.

↵ **IBF** (Input buffer full) when the data is loaded into input buffer, this signal rises to logic „1“. This can be used as an acknowledge that the data has been received by the receiver.

↵ The waveforms in fig show the operation in Mode 2 for output as well as input port.

↵ Note: WR must occur before ACK and STB must be activated before RD.



**8253(PROGRAMMABLE INTERVAL TIMER/COUNTER):** The Intel 8253 and 8254 are Programmable Interval Timers (PITs), which perform timing and counting functions using three 16-bit counters.<sup>[1]</sup> They were primarily designed for the Intel 8080/8085-processors, but later used in x86-systems. They (or an equivalent circuit embedded in a larger chip) are found in all IBM PC compatibles. The Intel 82c54 variant handles up to 10 MHz clock signals.

The timer has three counters, called channels. Each channel can be programmed to operate in one of six modes. Once programmed, the channels can perform their tasks independently. The timer is usually assigned to IRQ-0 (highest priority hardware interrupt) because of the critical function it performs and because so many devices depend on it.

### Counters

There are 3 counters (or timers), which are labeled as "Counter 0", "Counter 1" and "Counter 2".<sup>[3]</sup> Each counter has 2 input pins - "CLK" (clock input) and "GATE" - and 1-pin, "OUT", for data output. The 3 counters are 16-bit down counters independent of each other, and can be easily read by the CPU.

In the original IBM PCs, the first counter (selected by setting A1=A0=0, see Control Word Register below) is used to generate a timekeeping interrupt. The second counter (A1=0, A0=1) is used to trigger the refresh of DRAM memory. The last counter (A1=1, A0=0) is used to generate tones via the PC speaker.

Besides the counters, a typical Intel 8253 microchip also contains the following components:

### Data/Bus Buffer

This block contains the logic to buffer the data bus to / from the microprocessor, and to the internal registers. It has 8 input pins, usually labelled as D7..D0, where D7 is the MSB.

### Read/Write Logic

The Read/Write Logic block has 5 pins, which are listed below. Notice that "/X" denotes an active low signal.

- /RD: read signal
- /WR: write signal
- /CS: chip select signal
- A0, A1: address lines

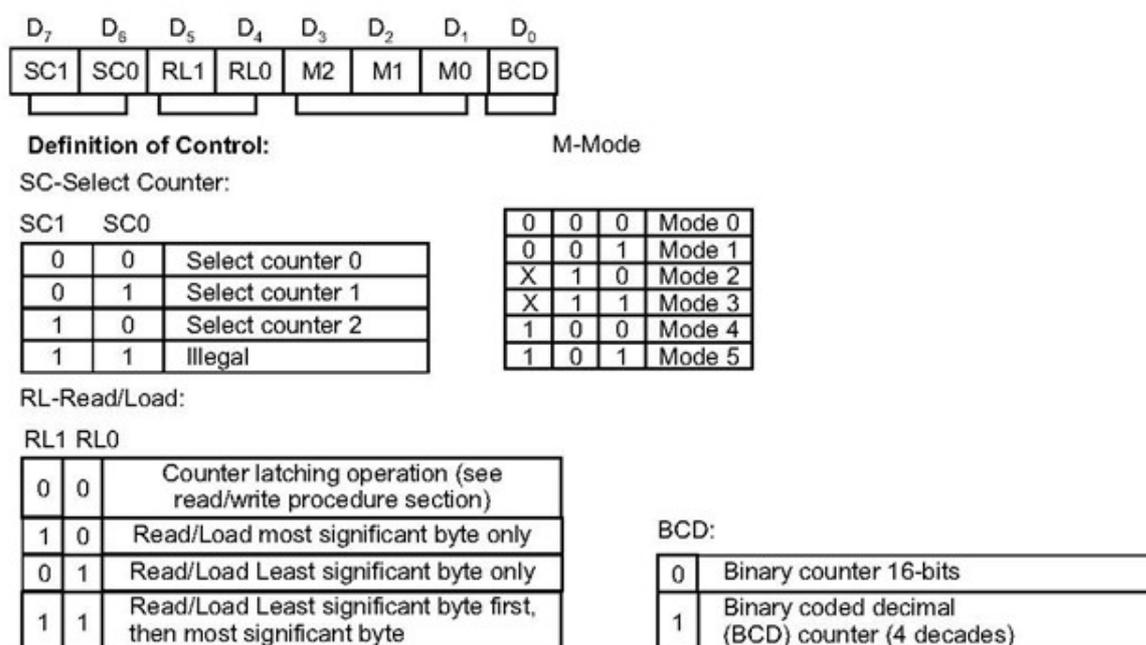
Operation mode of the PIT is changed by setting the above hardware signals. For example, to write to the Control Word Register, one needs to set  $\overline{CS}=0$ ,  $\overline{RD}=1$ ,  $\overline{WR}=0$ ,  $A1=A0=1$ .

### Control Word Register

This register contains the programmed information which will be sent (by the to the device. It defines how the PIT logically works. Each access to these ports takes about 1  $\mu$ s.

To initialize the counters, the microprocessor must write a control word (CW) in this register. This can be done by setting proper values for the pins of the Read/Write Logic block and then by sending the control word to the Data/Bus Buffer block.

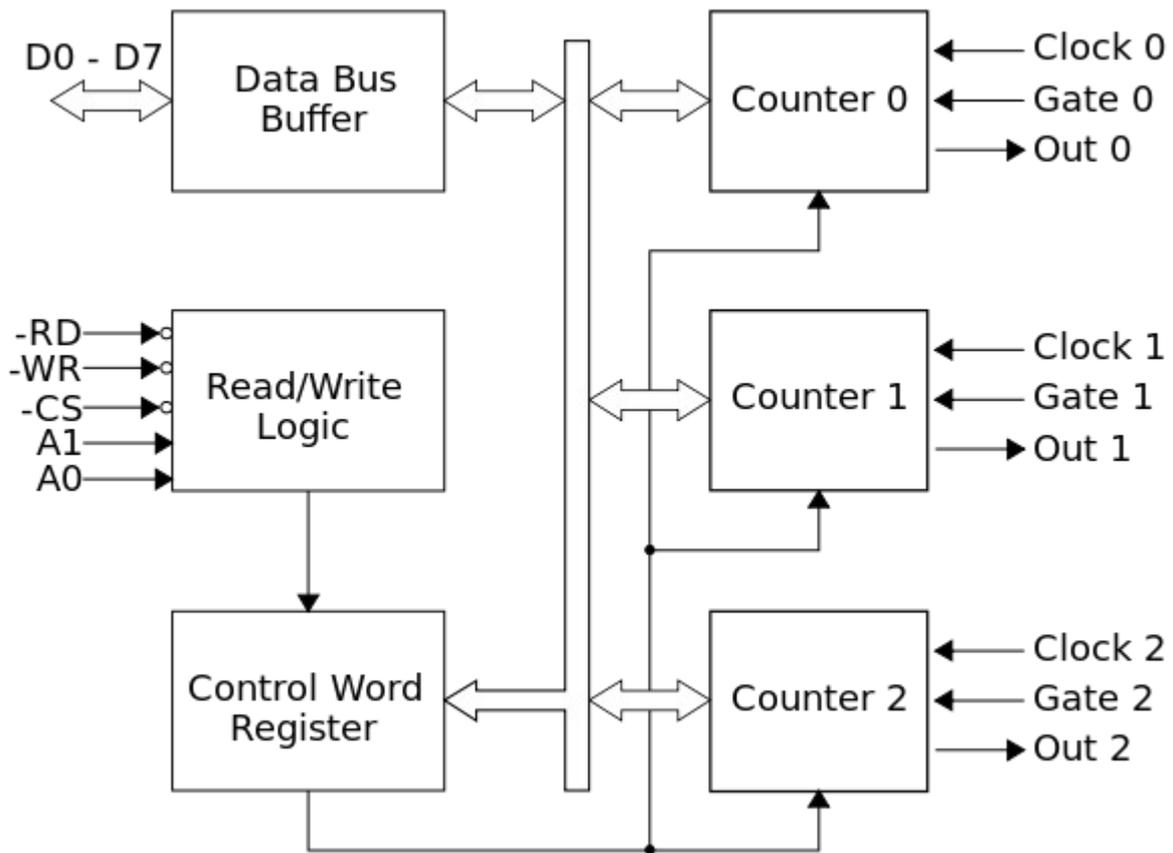
The control word register contains 8 bits, labeled D7..D0 (D7 is the MSB).



**Fig. 9g.3:** Control word format and mode definitions  
(Source: Intel Corporation)

When setting the PIT, the microprocessor first sends a control message, then a count message to the PIT. The counting process will start after the PIT has received these messages, and, in some cases, if it detects the rising edge from the GATE input signal.

On PCs the address for timer0 (chip) is at port 40h..43h like described and the second timer1 (chip) is at 50h..53h.



### Operation Modes

The D3, D2, and D1 bits of the Control Word set the operating mode of the timer. There are 6 modes in total; for modes 2 and 3, the D3 bit is ignored, so the missing modes 6 and 7 are aliases for modes 2 and 3. Notice that, for modes 0, 2, 3 and 4, GATE must be set to HIGH to enable counting. For mode 5, the rising edge of GATE starts the count. For details on each mode, see the reference links.

#### Mode 0 (000): Interrupt on Terminal Count

Mode 0 is used for the generation of accurate time delay under software control. In this mode, the counter will start counting from the initial COUNT value loaded into it, down to 0. Counting rate is equal to the input clock frequency.

The OUT pin is set low after the Control Word is written, and counting starts one clock cycle after the COUNT programmed. OUT remains low until the counter reaches 0, at which point OUT will be set high until the counter is reloaded or the Control Word is written. The Gate signal should remain active high for normal counting. If Gate goes low, counting gets terminated and current count is latched till Gate pulse goes high again. The first byte of the new count when loaded in the count register, stops the previous count.

#### Mode 1 (001): Programmable One Shot

In this mode 8253 can be used as Monostable Multivibrator. GATE input is used as trigger input. OUT will be initially high. OUT will go low on the Clock pulse following a trigger to begin the one-shot pulse, and will remain low until the Counter reaches zero. OUT will then go high and remain high until the CLK pulse after the next trigger.

After writing the Control Word and initial count, the Counter is armed. A trigger results in loading the Counter and setting OUT low on the next CLK pulse, thus starting the one-shot pulse. An initial count of N will result in a one-shot pulse N CLK cycles in duration.

The one-shot is retriggerable, hence OUT will remain low for N CLK pulses after any trigger. The one-shot pulse can be repeated without rewriting the same count into the counter. GATE has no effect on OUT. If a new count is written to the Counter during a oneshot pulse, the current one-shot is not affected unless the counter is retriggered. In that case, the Counter is loaded with the new count and the oneshot pulse continues until the new count expires.

### **Mode 2 (X10): Rate Generator**

In this mode, the device acts as a divide-by-n counter, which is commonly used to generate a real-time clock interrupt.

Like other modes, counting process will start the next clock cycle after COUNT is sent. OUT will then remain high until the counter reaches 1, and will go low for one clock pulse. OUT will then go high again, and the whole process repeats itself.

The time between the high pulses depends on the preset count in the counter's register, and is calculated using the following formula:

Note that the values in the COUNT register range from to 1; the register never reaches zero.

### **Mode 3 (X11): Square Wave Generator**

This mode is similar to mode 2. However, the duration of the high and low clock pulses of the output will be different from mode 2. Suppose N is the number loaded into the counter (the COUNT message), the output will be

- high for  $N / 2$  counts, and low for  $N / 2$  counts, if N is even.
- high for  $N + 1 / 2$  counts, and low for  $N - \frac{1}{2}$  counts, if N is odd.

### **Mode 4 (100): Software Triggered Strobe**

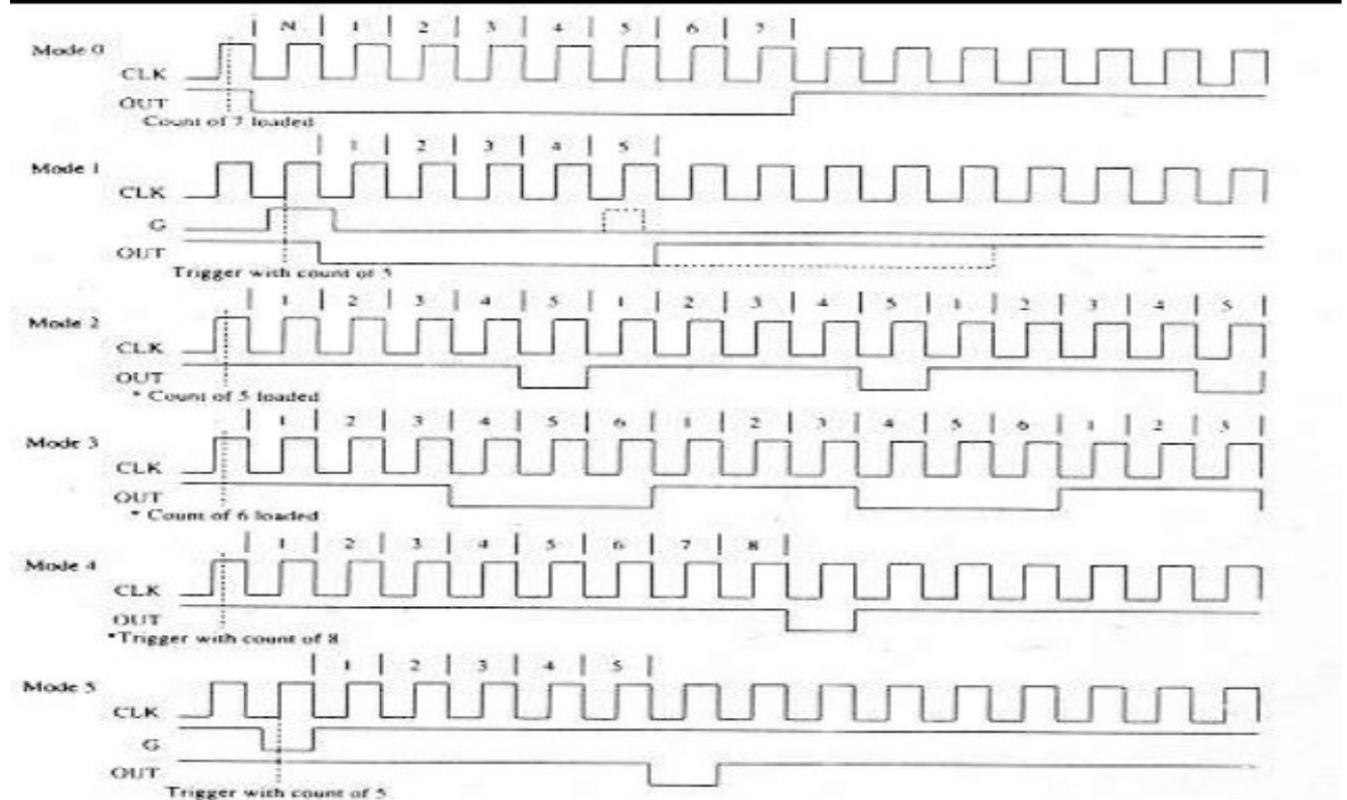
After Control Word and COUNT is loaded, the output will remain high until the counter reaches zero. The counter will then generate a low pulse for 1 clock cycle (a strobe) - after that the output will become high again.

### **Mode 5 (101): Hardware Triggered Strobe**

This mode is similar to mode 4. However, the counting process is triggered by the GATE input.

After receiving the Control Word and COUNT, the output will be set high. Once the device detects a rising edge on the GATE input, it will start counting. When the counter reaches 0, the output will go low for one clock cycle – after that it will become high again, to repeat the cycle on the next rising edge of GATE.

## Modes of Operation of 8253



**8259 (PIC):** 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 will relieve the processor from all these tasks. The programmable interrupt controller 8259A from Intel is one such device. The 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.

## **Architecture and Signal Descriptions of 8259A:**

The architectural block diagram of 8259A is shown in Fig. 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 INTA pulse. The IR0 has the highest priority while the IR7 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 puts. IMR operates on IRR at the direction of the Priority Resolver.

### **Interrupt Control Logic**

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

### **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 buffer during read or write operations.

### **Read write Control Logic**

This circuit accepts and decodes commands from the CPU. This also allows the status of the 8259A to be transferred on to the data bus.

## Cascade Buffer/Comparator

This block stores and compares the ID's 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 slave mode. The 8259A in 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 INTA pulse. Figure 1.2 shows the pin configuration of 8259A, followed by their functional description of each of the signals in brief.

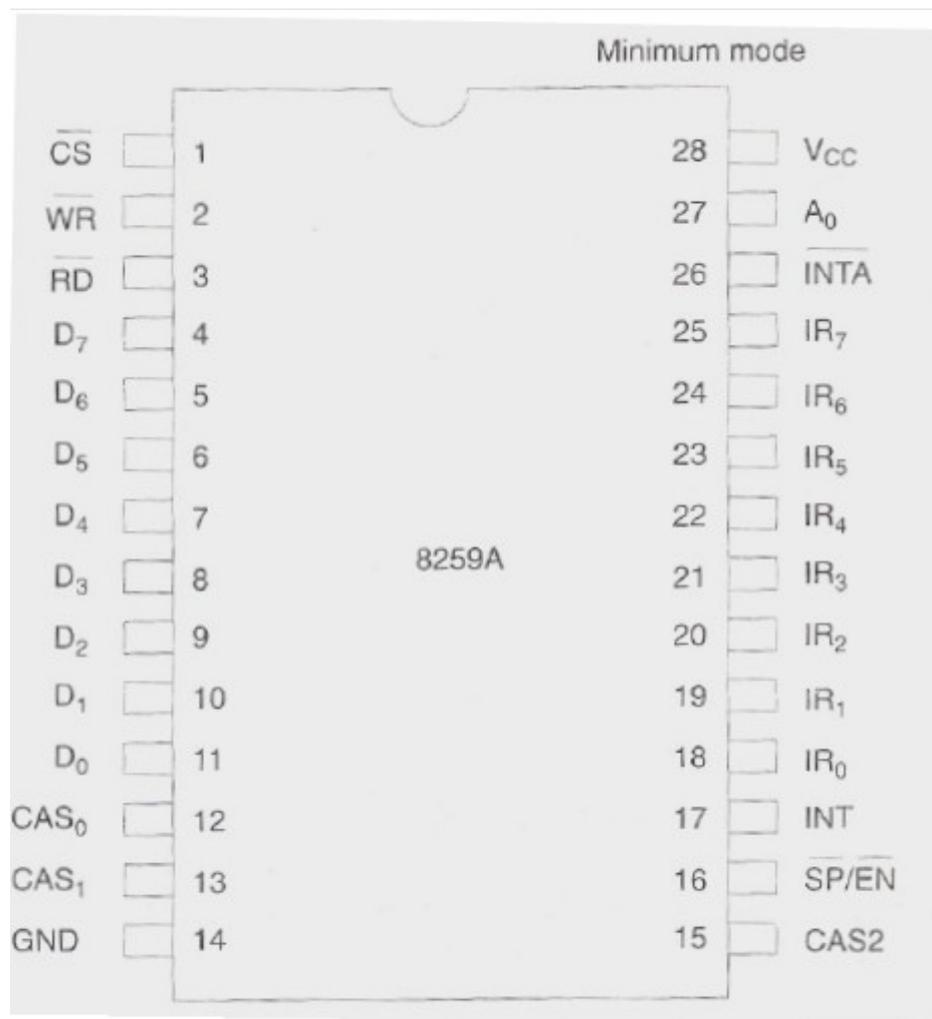


Fig.1.2. 8259 Pin Diagram

**CS** : This is an active-low chip select signal for enabling RD\* and WR\* operations of 8259A. INTA\* function is independent of CS\*.

**WR\*** : This pin is an active-low write enable input to 8259A. This enables it to accept command words from CPU.

**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. D7-D0 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.

CASo-CAS2: Cascade Lines A single 8259A provides eight vectored interrupts. If more interrupts are required, the 8259A is used in cascade mode. In cascade mode, a master 8259A along with eight slaves 8259A can provide up to 64 vectored interrupt lines. These three lines act as select lines for addressing the slaves 8259A.

PS\*/EN\*: This pin is a dual purpose pin. When the chip is used in buffered mode, it can be used as 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 (SP = 1) or a slave (EN = 0).

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.

IR0-IR7 (Interrupt requests): These pins act as inputs to accept interrupt requests to the CPU. In edge triggered mode, an interrupt service is requested by raising an IR pin from a low to a high state and holding it high until it is acknowledged, and just by latching it to high level, if used in level triggered mode.

INTA\* (Interrupt acknowledge): This pin is an input used to strobe-in 8259A interrupt vector data on to the data bus. In conjunction with CS, WR, and 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.

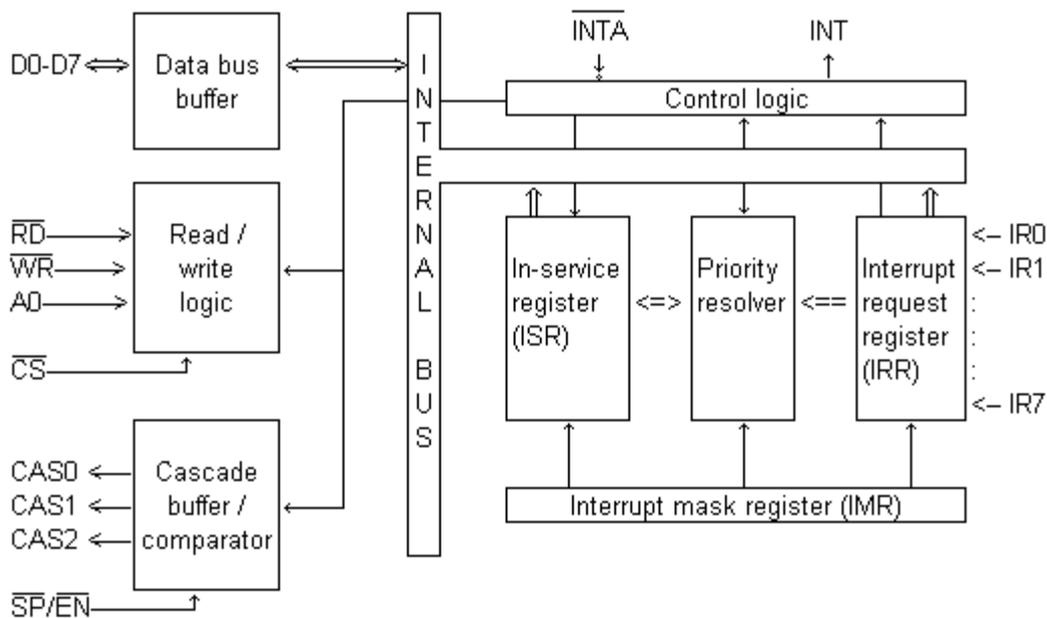
### **Interrupt Sequence in an 8086 System:**

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 INTA pulse.

4. Upon receiving an 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 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 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.

8259 internal block diagram



## 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  $A0 = 0$  and  $D4 = 1$ , the control word is recognized as ICW1. It contains the control bits for edge/level triggered mode, single/cascade mode, call address interval and whether ICW4 is required or not, etc. If  $A0 = 1$ , the control word is recognized as ICW2. The ICW2 stores details regarding interrupt vector addresses. The initialisation sequence of 8259A is described in form of a flow chart in Fig. 1.3. The bit functions of the ICW1 and ICW2 are self explanatory as shown in Fig. 1.4. Once ICW1 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 status read is set to IRR.
- (f) If  $IC4 = 0$ , all the functions of ICW4 are set to zero. Master/slave bit in ICW4 is used in the buffered mode only.

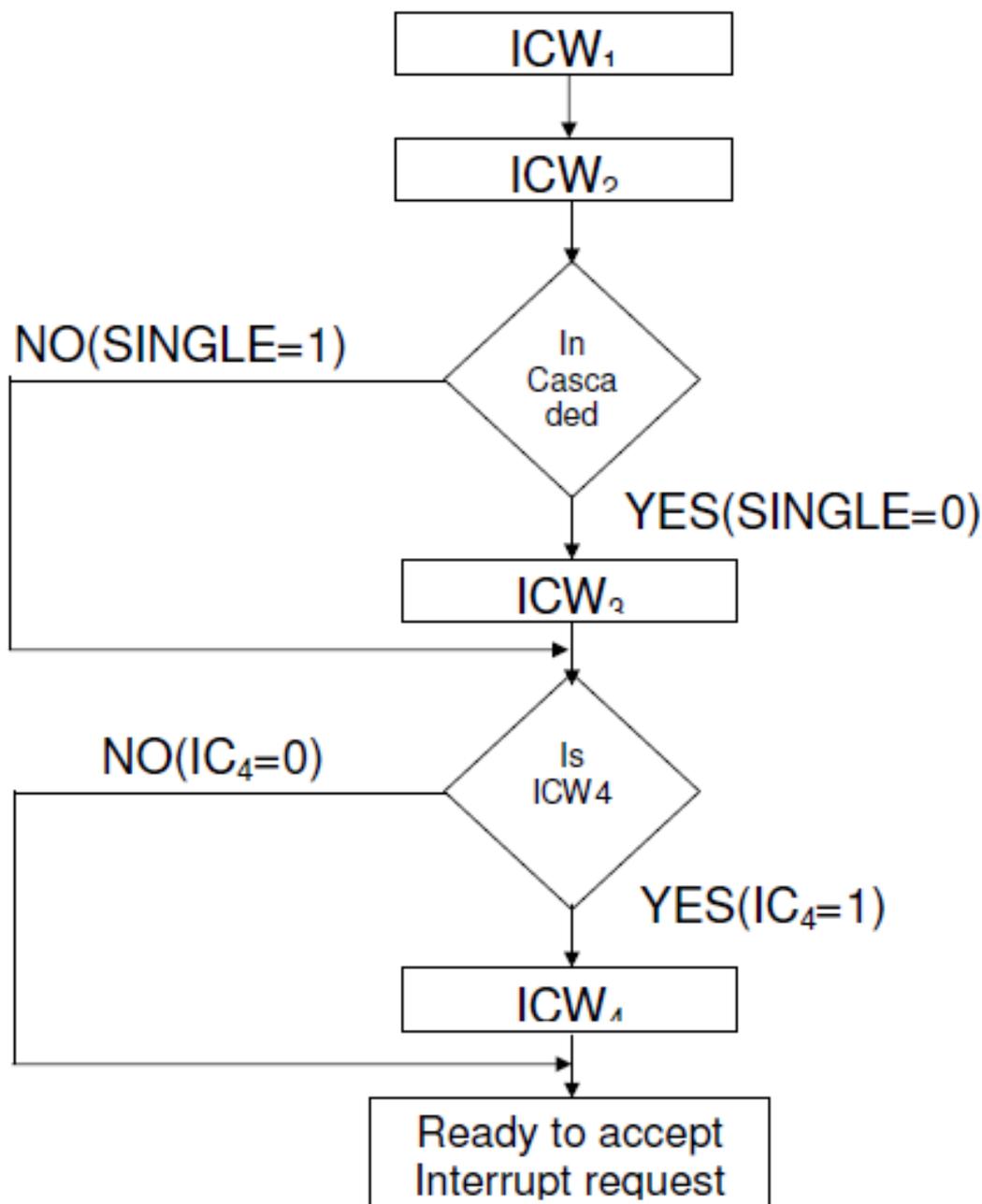


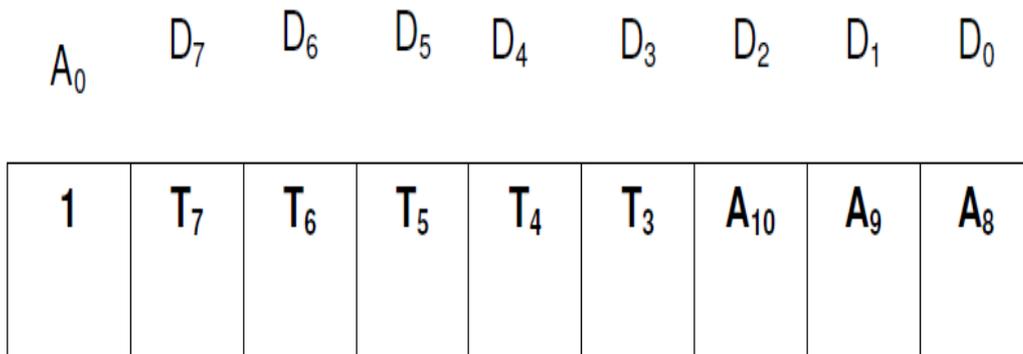
Fig1.3. Initialisation Sequence of 8259A

In an 8085 based system, A15 – A8 of the interrupt vector address are the respective bits of ICW2. In 8086/88 based system A15 – A11 of the interrupt vector address are inserted in place of T7 - T3 respectively and the remaining three bits (A8, A9 and A10) are selected depending upon the interrupt level, i.e. from 000 to 111 for IR0 to IR7.

ICW1 and ICW2 are compulsory command words in initialization sequence of 8259A as is evident from Fig. 1.3, while ICW3 and ICW4 are optional. The ICW3 is read only when there are more than one 8259As in the system, i.e. cascading is used (SNGL = 0). The SNGL bit in ICW1 indicates whether the 8259A is in cascade mode or not. The ICW3 loads an 8-bit slave register. Its detailed functions are as follows.



## ICW<sub>2</sub>



$T_7-T_3$  are  $A_3-A_0$  of Interrupt vector address

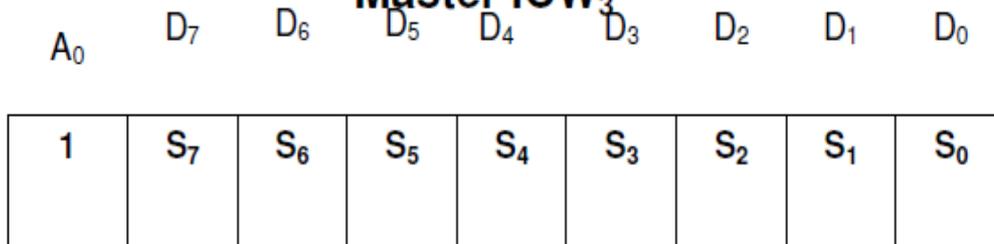
$A_{10}-A_9, A_8$  - Selected according to Interrupt request level.

They are not the address lines to microprocessor

$A_0 - 1$  Selects ICW<sub>2</sub>

Fig1.4. Initialisation Command Words ICW<sub>1</sub> and ICW<sub>2</sub>

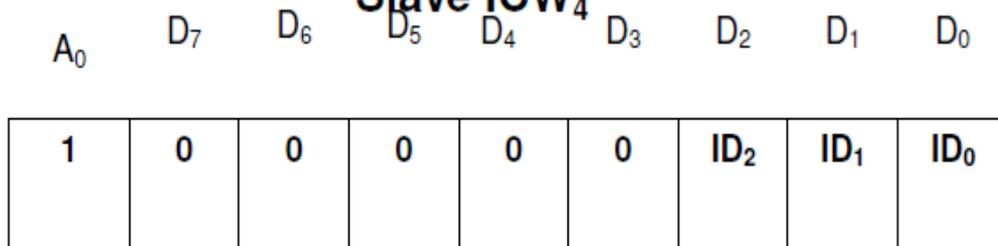
## Master ICW<sub>3</sub>



$S_n = 1 - IR_n$  Input has a slave

$= 0 - IR_n$  Input does not have a slave

## Slave ICW<sub>4</sub>



$D_2 D_1 D_0 - 000$  to  $111$  for  $IR_0$  to  $IR_7$  or slave 1 to slave 8

Fig1.5. ICW<sub>3</sub> in Master and Slave Mode

ICW4 The use of this command word depends on the IC4 bit of ICW1. If IC4= 1, ICW4 is used, otherwise it is neglected. The bit functions of ICW4 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 ICW4.

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.

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

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

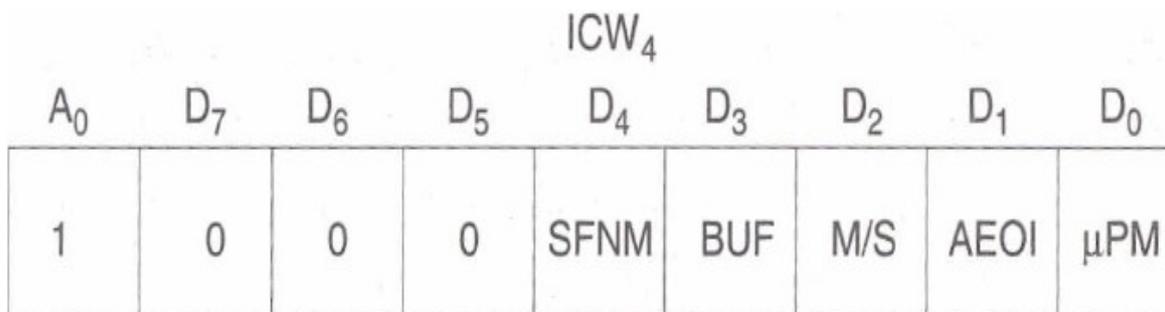


Fig1.6 shows the ICW<sub>4</sub> bit positions

## Operation Command Words

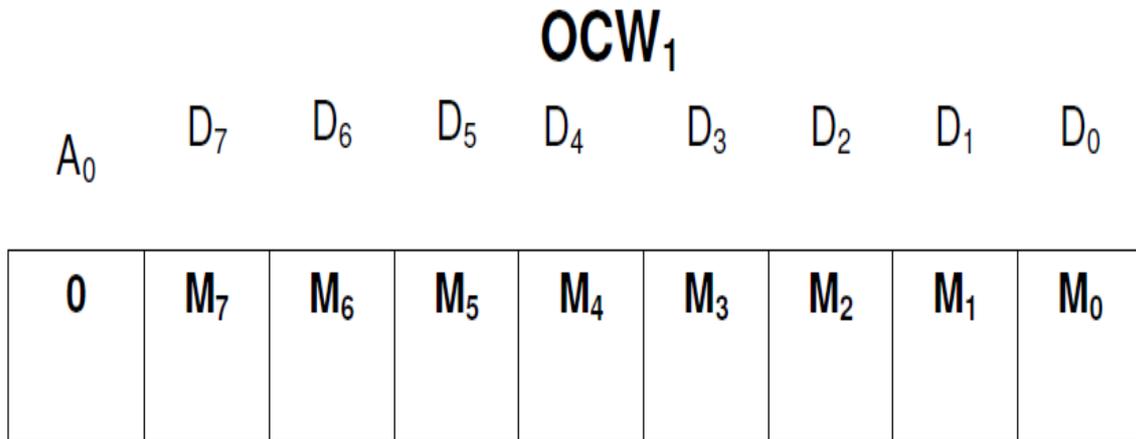
Once 8259A is initialized using the previously discussed command words for initialisation, it is ready for its normal function, i.e. for accepting the interrupts but 8259A has its own ways of handling the received interrupts called as modes of operation. These modes of operations can be selected by programming, 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 OCW1, OCW2 and OCW3 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. 1.7 (a), (b) and (c) with the bit selection details. OCW1 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 OCW2 the three bits, viz. R, SL and EOI control the end of interrupt, the rotate mode and their combinations as shown in Fig. 1.7

(b), The three bits L2, L1 and L0 in OCW2 determine the interrupt level to be selected for operation, if the SL bit is active, i.e. '1'. The details of OCW2 are shown in Fig. 1.7(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 OCW3 are given in Fig. 1.7 (c) along with their bit definitions.



M<sub>0</sub>-M<sub>7</sub>    1 - Mask Set  
                   0 - Mask Deassert

Fig1.7.a.

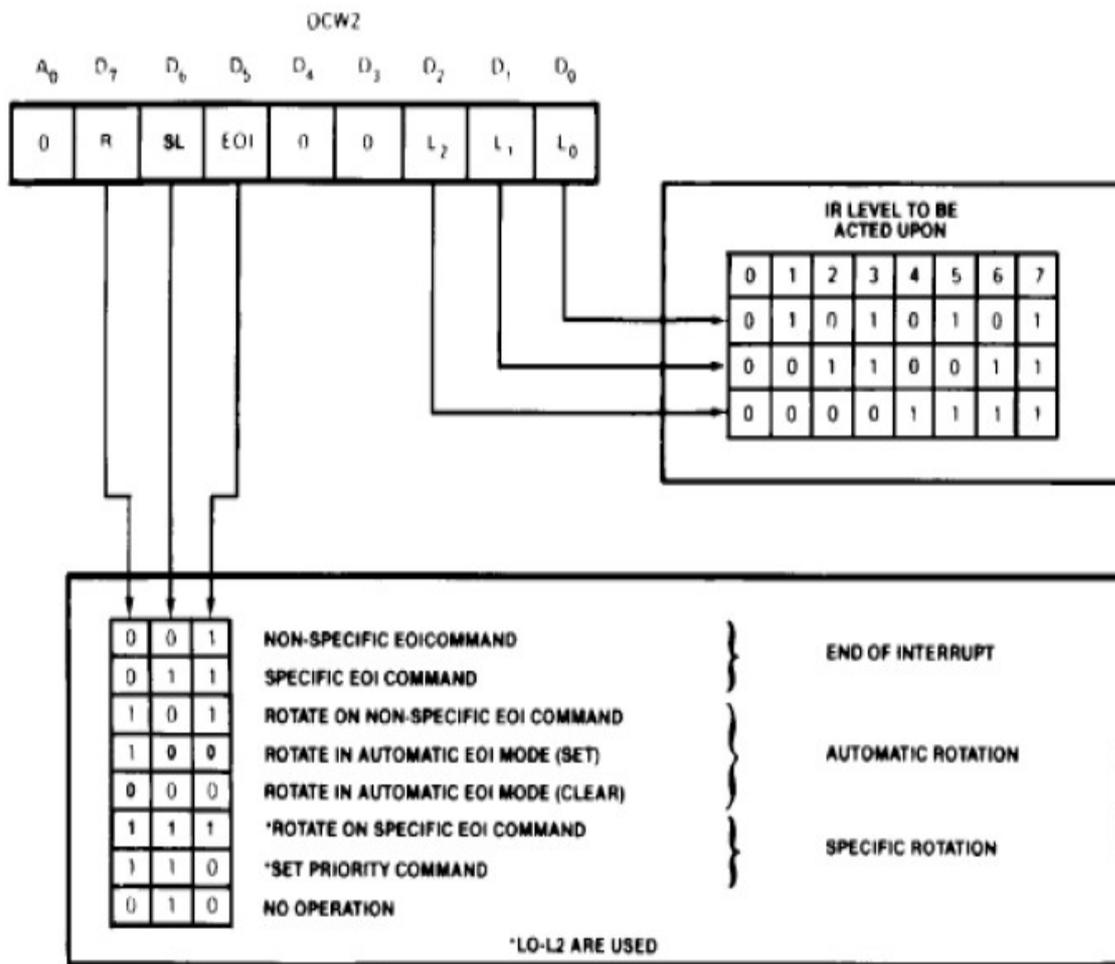


Fig1.7.b

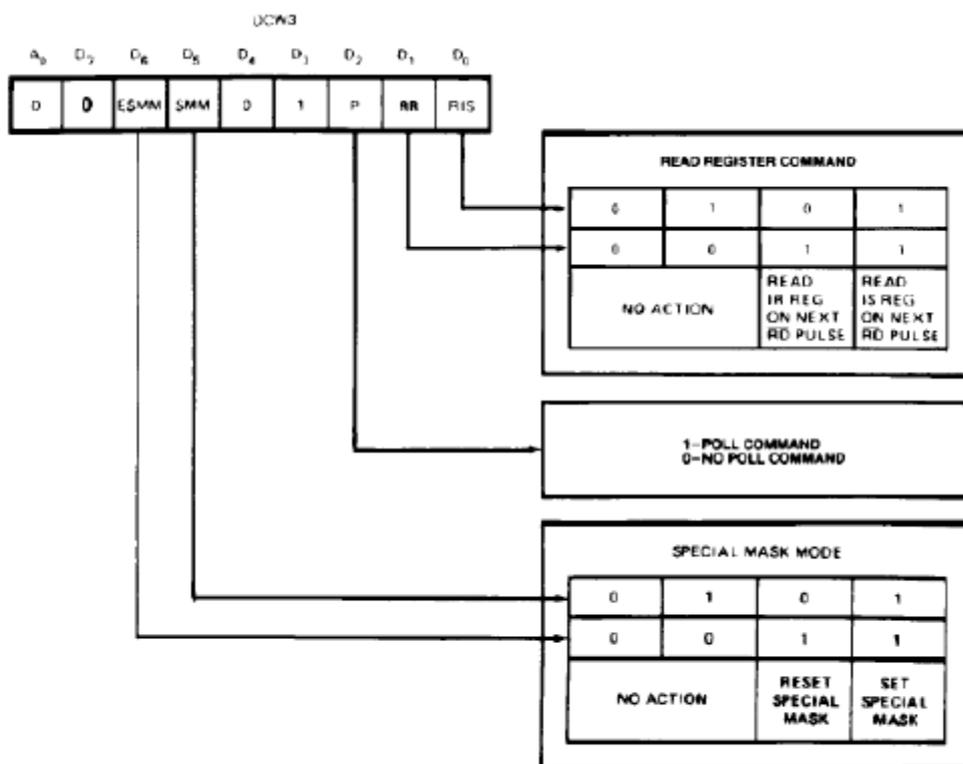


Fig1.7.c.

## SERIAL COMMUNICATION USING 8251

8251 is a Universal Synchronous and Asynchronous Receiver and Transmitter compatible with Intel's processors. 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.

Basic Modes of data transmission

- a) Simplex
- b) Duplex
- c) Half Duplex

a) Simplex mode

Data is transmitted only in one direction over a single communication Channel. For example, the processor may transmit data for a CRT display unit in this mode.

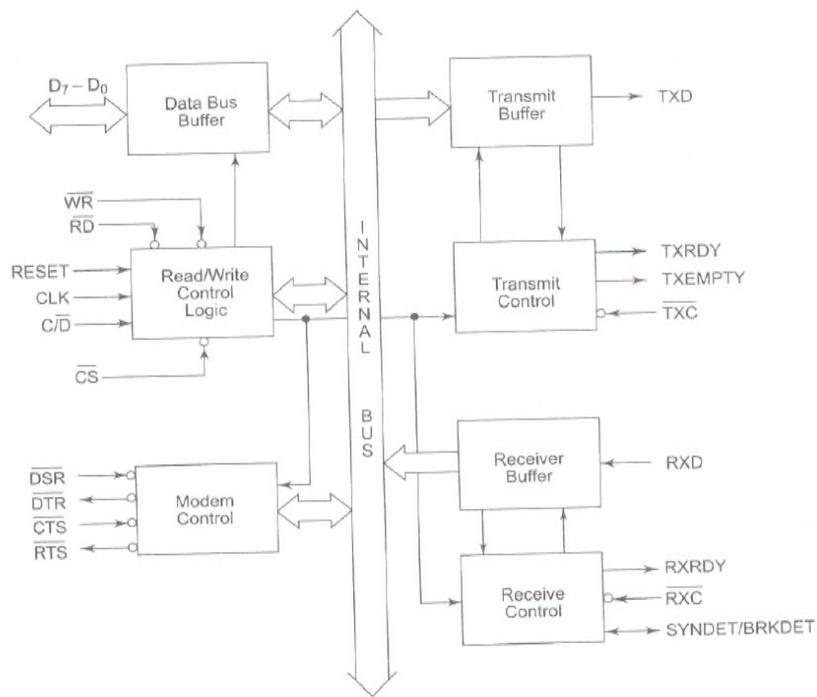
b) Duplex Mode

In duplex mode, data may be transferred between two transreceivers in both directions Simultaneously.

c) Half Duplex mode

In this mode, data transmission may take place in either direction, but at a time data may be transmitted only in one direction. A computer may communicate with a terminal in this mode. It is not possible to transmit data from the computer to the terminal and terminal to computer simultaneously.

**ARCHITECTURE OF 8251 USART:**

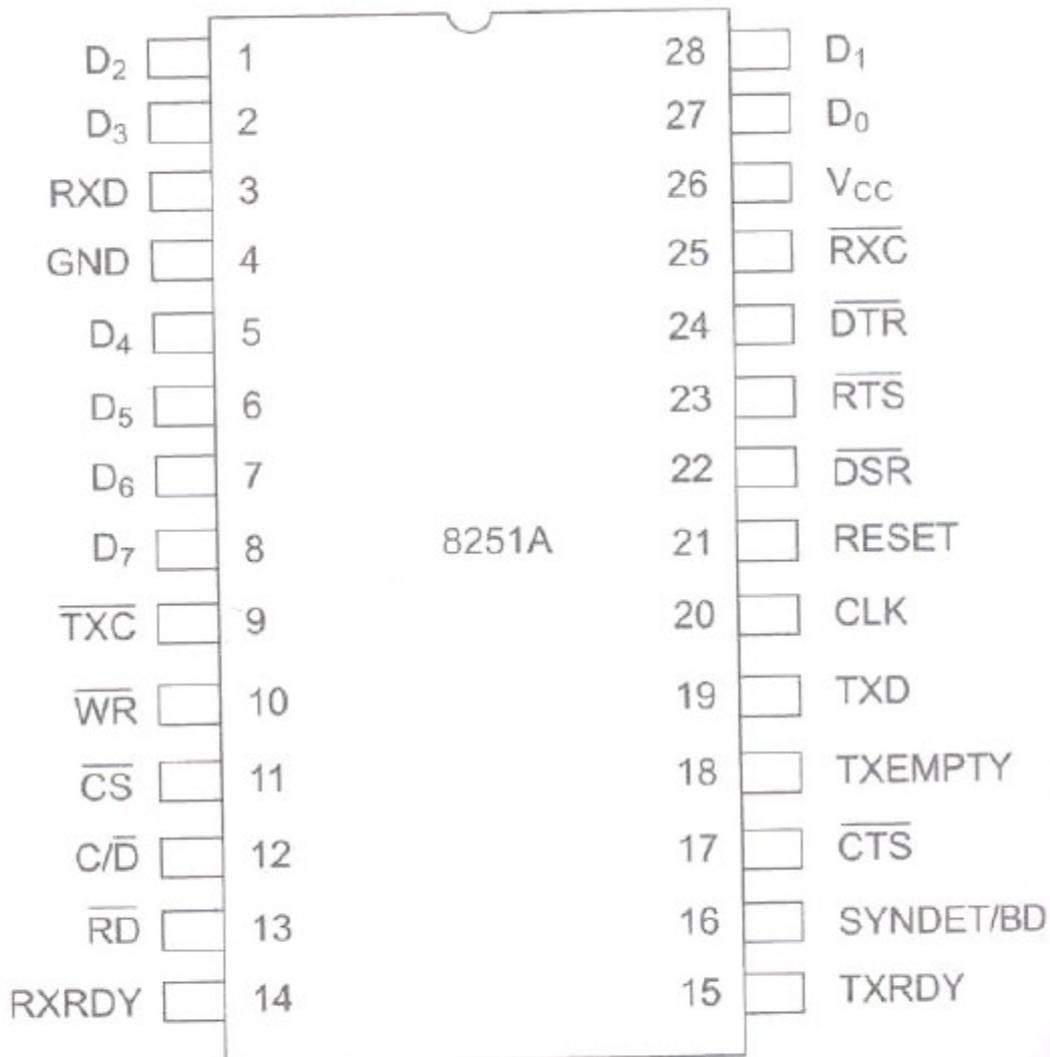


**Fig. 8.1** Internal architecture of 8251

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.  $C / \overline{D}$  decides whether the address on internal data bus is control address / data address. The modem control unit handles the modem handshake signals to coordinate the communication between modem and USART. The transmit control unit transmits the data byte received by the data buffer from the CPU for serial communication. The transmission rate is controlled by the input frequency. Transmit control unit also derives two transmitter status signals namely TXRDY and TXEMPTY which 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 for further transmission.

The receive control unit decides the receiver frequency as controlled by the RXC input frequency. The receive control 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.

**Signal Description of 8251:**



**Fig. 8.2** Pin Configuration of 8251

### Operating Modes of 8251

1. Asynchronous mode
2. Synchronous mode

#### 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 TXC.

#### Asynchronous Mode (Receive)

A falling edge on RXD input line marks a start bit. 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. The 8-bit character is then loaded into the parallel I/O buffer of 8251.

RXRDY pin is 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.

Mode instruction format for Asynchronous mode

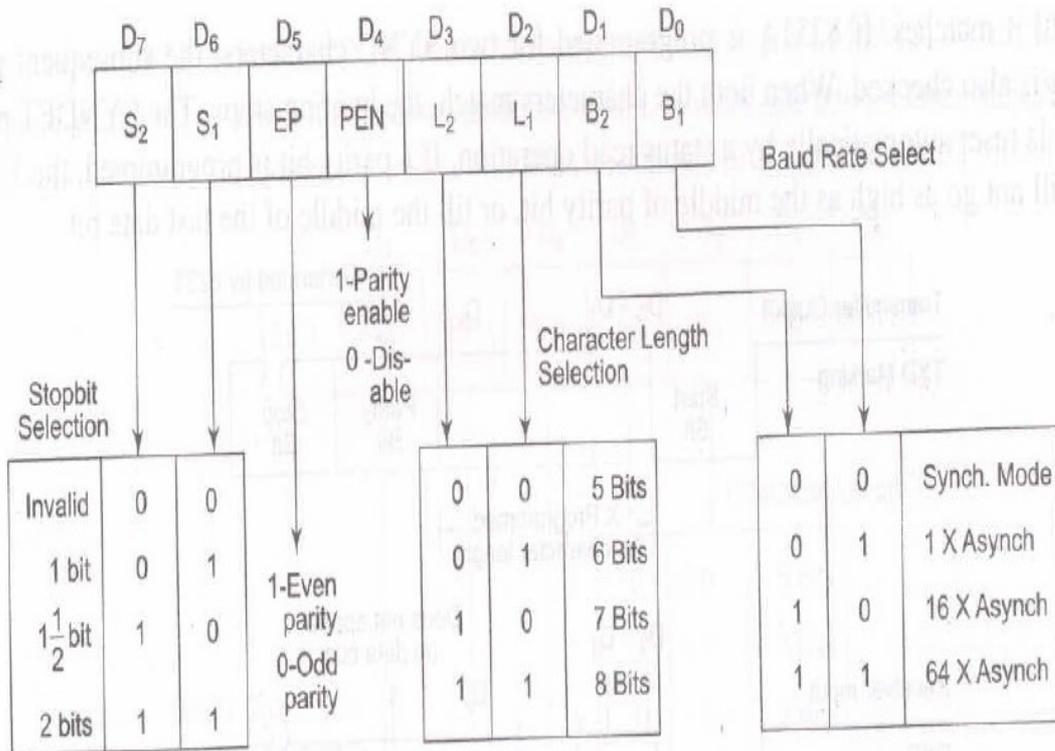


Fig. 8.3 Mode Instruction Format Asynchronous Mode

Asynchronous Mode Transmit and Receive Formats

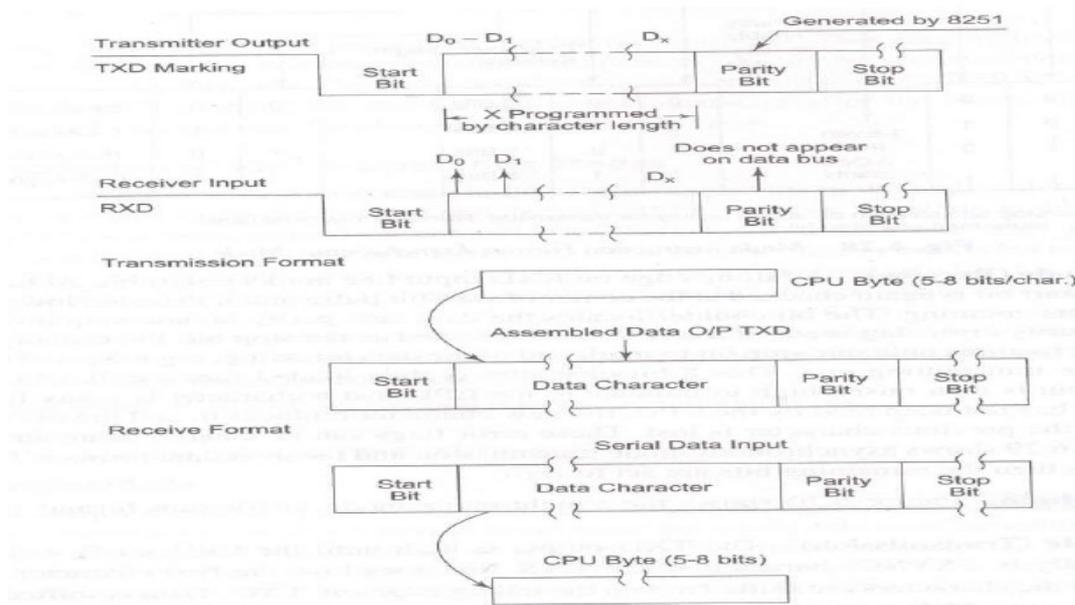


Fig. 8.4 Asynchronous Mode Transmit and Receive Formats

## Synchronous mode

### Synchronous Mode Instruction Format

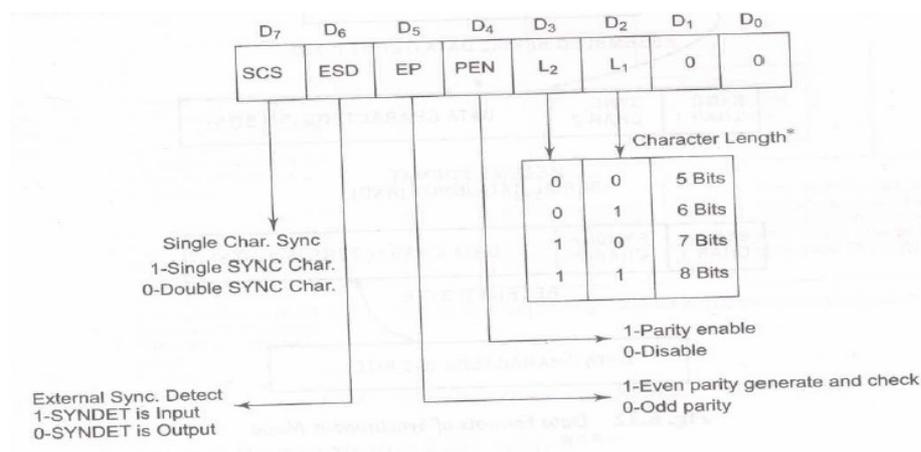


Fig. 8.5 Synchronous Mode Instruction Format

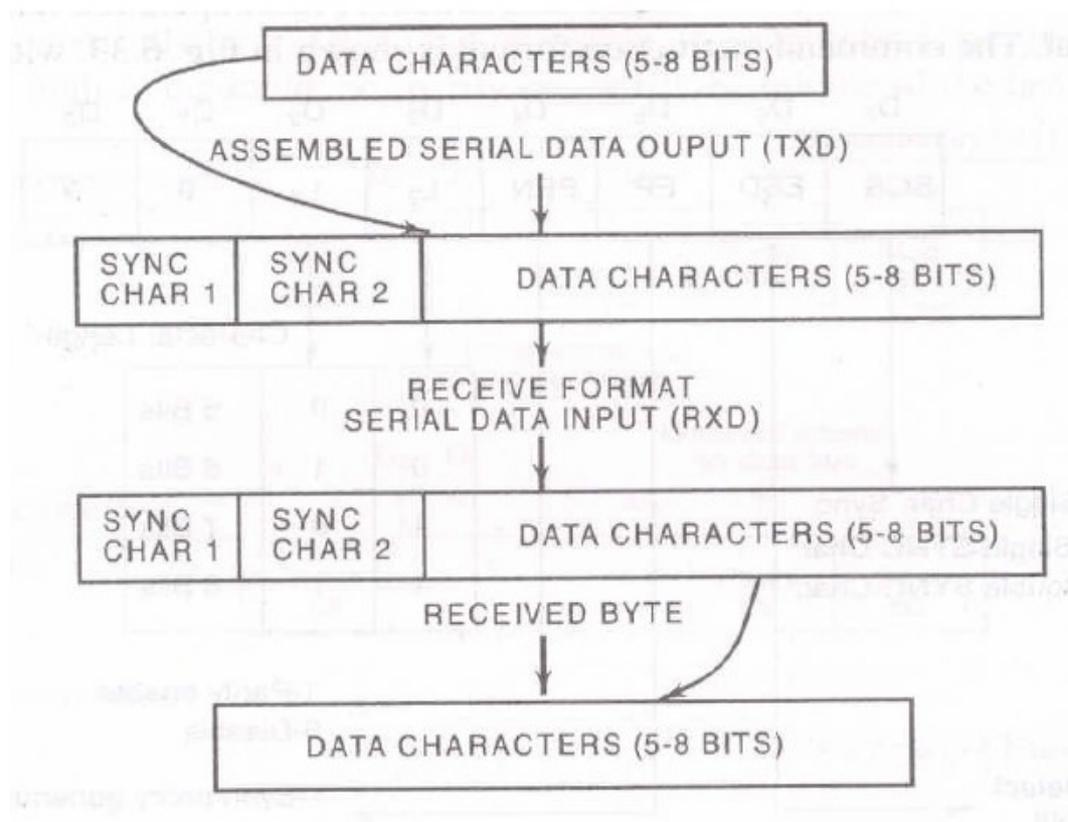
### Synchronous Mode (Transmission)

The TXD output is high until the CPU sends a character to 8251 which usually is a SYNC character. When  $\overline{\text{CTS}}$  line goes low, the first character is serially transmitted out. Characters are shifted out on the falling edge 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.

### Synchronous Mode (Receiver)

In this mode, the character synchronization can be achieved internally or externally. The data on RXD pin is sampled on rising edge of the  $\overline{\text{RXC}}$ . The content of the receiver buffer is compared with the first SYNC character at every edge until it matches. If 8251 is programmed for two SYNC characters, the subsequent received character is also checked. When the characters match, the hunting stops. The SYNDET pin set high and is reset automatically by a status read operation. In the external SYNC mode, the synchronization is achieved by applying a high level on the SYNDET input pin that forces 8251 out of HUNT mode. The high level can be removed after one  $\overline{\text{RXC}}$  cycle. The parity and overrun error both are checked in the same way as in asynchronous mode.

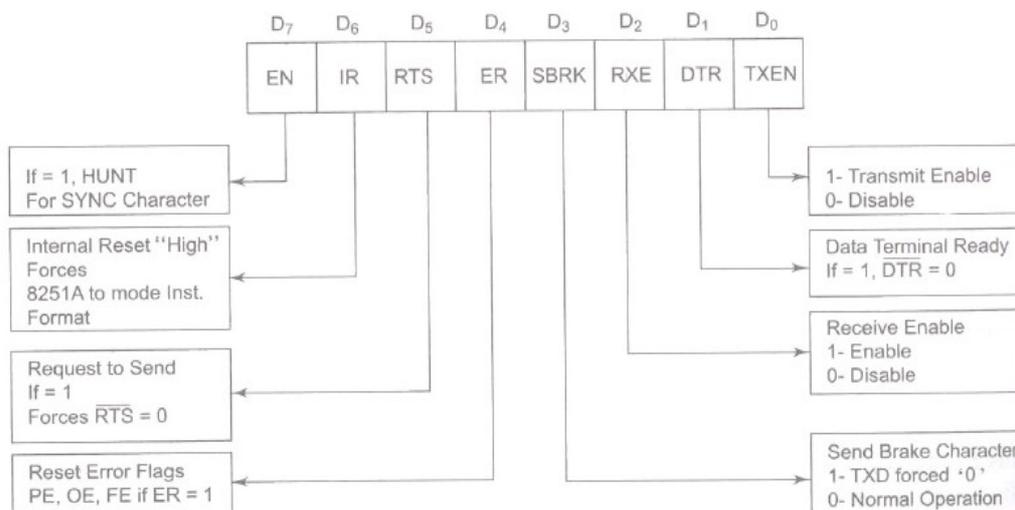
### Synchronous mode Transmit and Receive data format



**Fig. 8.6** Data Formats of Synchronous Mode

### Command Instruction Definition

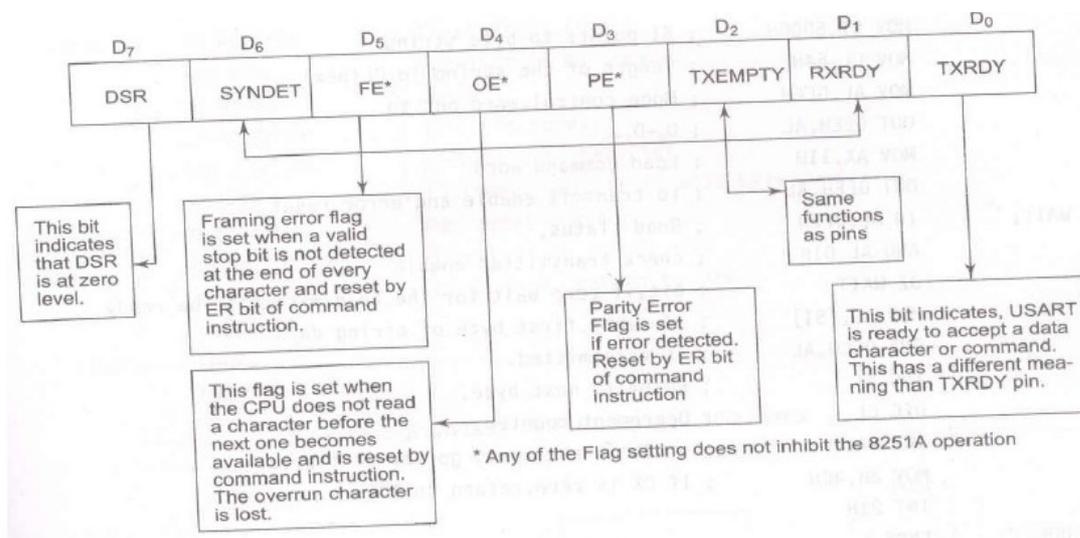
The command instruction controls the actual operations of the selected format like enable transmit/receive, error reset and modem control. A reset operation returns 8251 back to mode instruction format.



**Fig. 8.7** Command Instruction Format

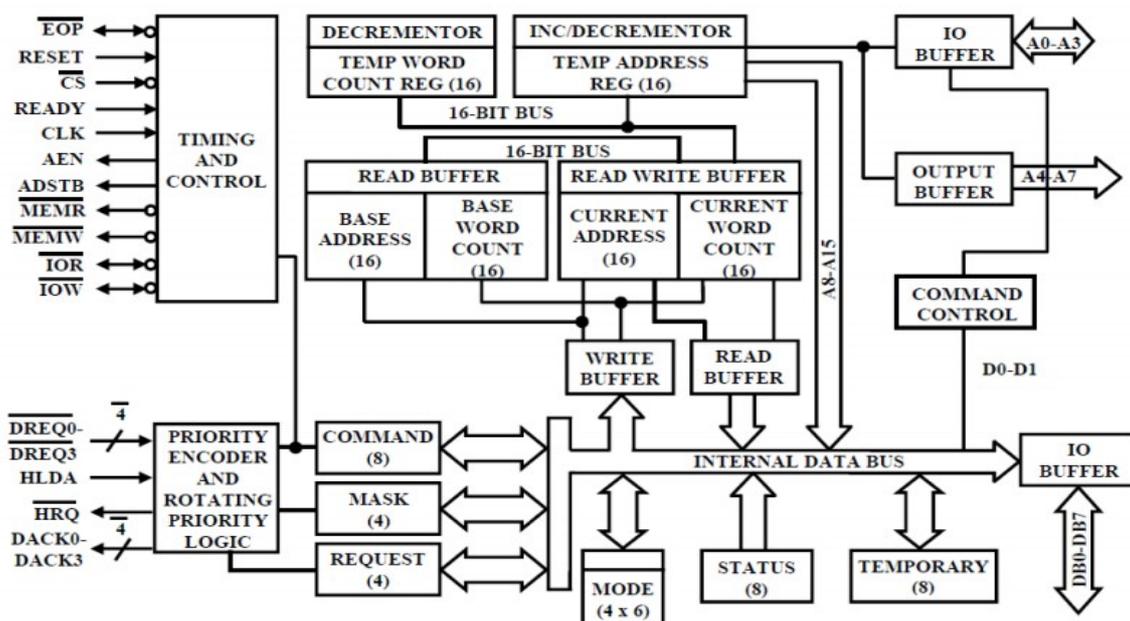
## Status Read Definition

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



**Fig. 8.8** Status Read Instruction Format

## Block Diagram of 8237



### 8237 Internal Registers

#### CAR

- The current address register holds a 16-bit memory address used for the DMA transfer.
- each channel has its own current address register for this purpose.
- When a byte of data is transferred during a DMA operation, CAR is either incremented or decremented. depending on how it is programmed

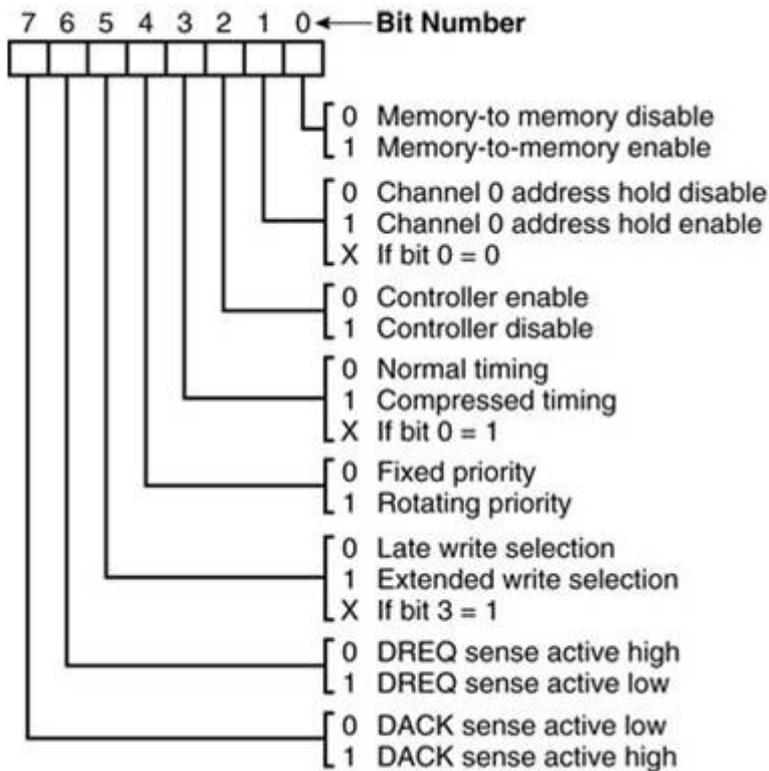
#### CWCR

- The current word count register programs a channel for the number of bytes to transferred during a DMA action.

#### CR

- The command register programs the operation of the 8237 DMA controller.
- The register uses bit position 0 to select the memory-to-memory DMA transfer mode.

1. Memory-to-memory DMA transfers use DMA channel
2. DMA channel 0 to hold the source address
3. DMA channel 1 holds the destination address

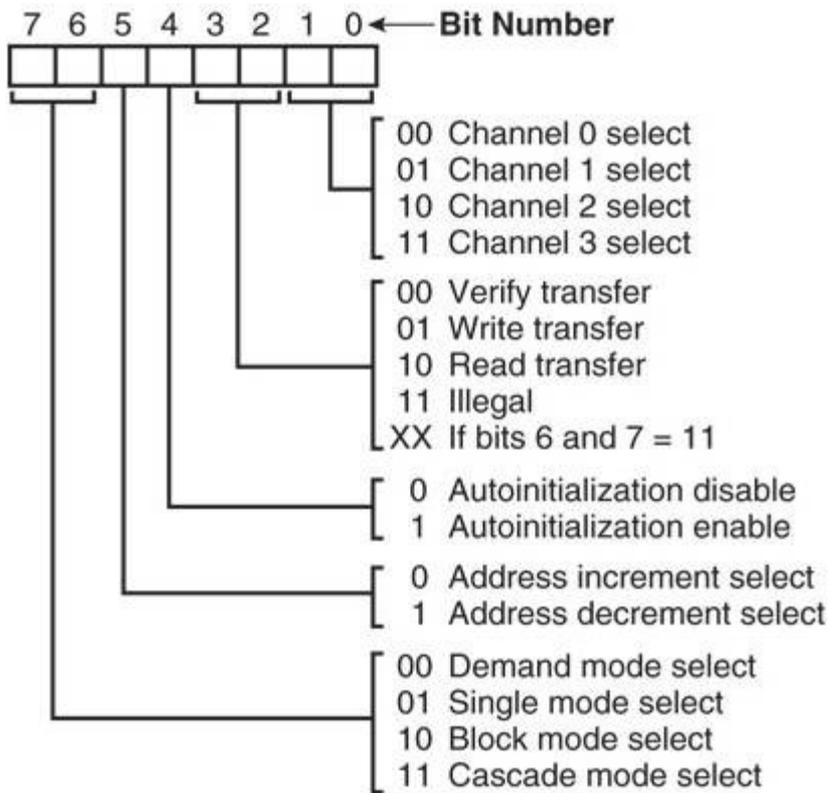


### **BA and BWC**

- The base address (BA) and base word count (BWC) registers are used when auto-initialization is selected for a channel.
- In auto-initialization mode, these registers are used to reload the CAR and CWCR after the DMA action is completed.

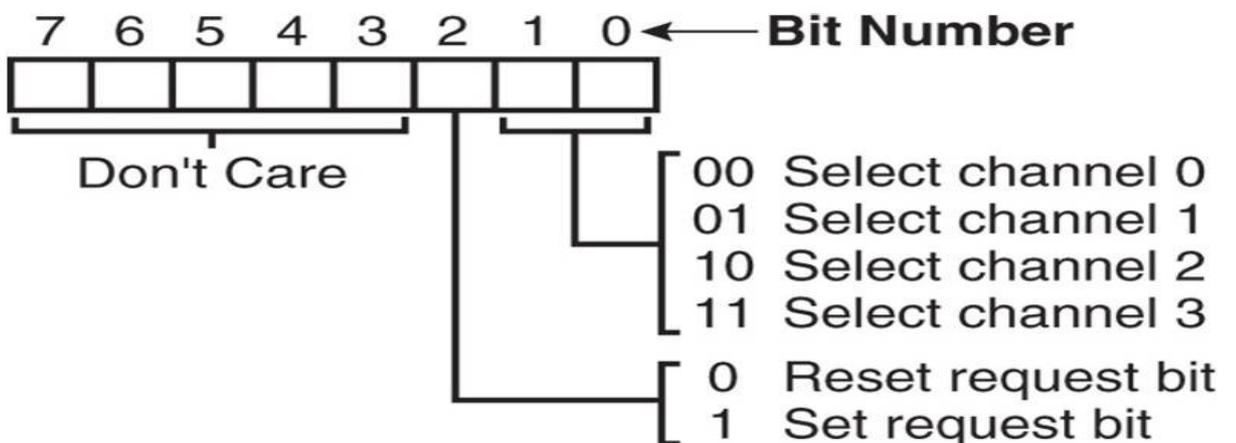
### **MR**

- The mode register programs the mode of operation for a channel.
- Each channel has its own mode register as selected by bit positions 1 and 0.
  1. Remaining bits of the mode register select operation, auto-initialization, increment/decrement, and mode for the channel



**BR**

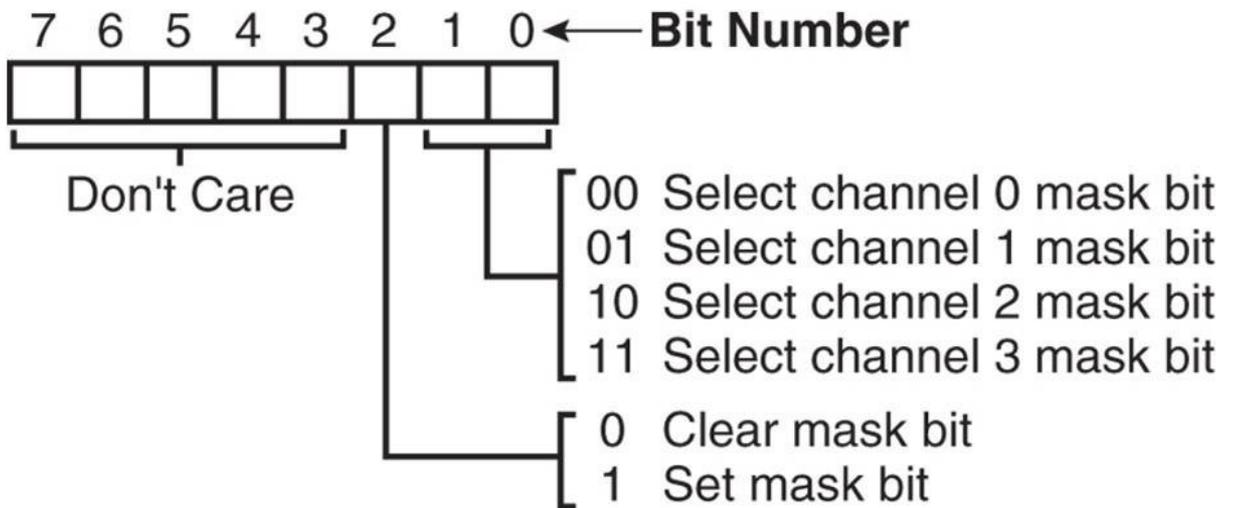
- The bus request register is used to request a DMA transfer via software.
  1. very useful in memory-to-memory transfers, where an external signal is not available to begin the DMA transfer



**MRSR**

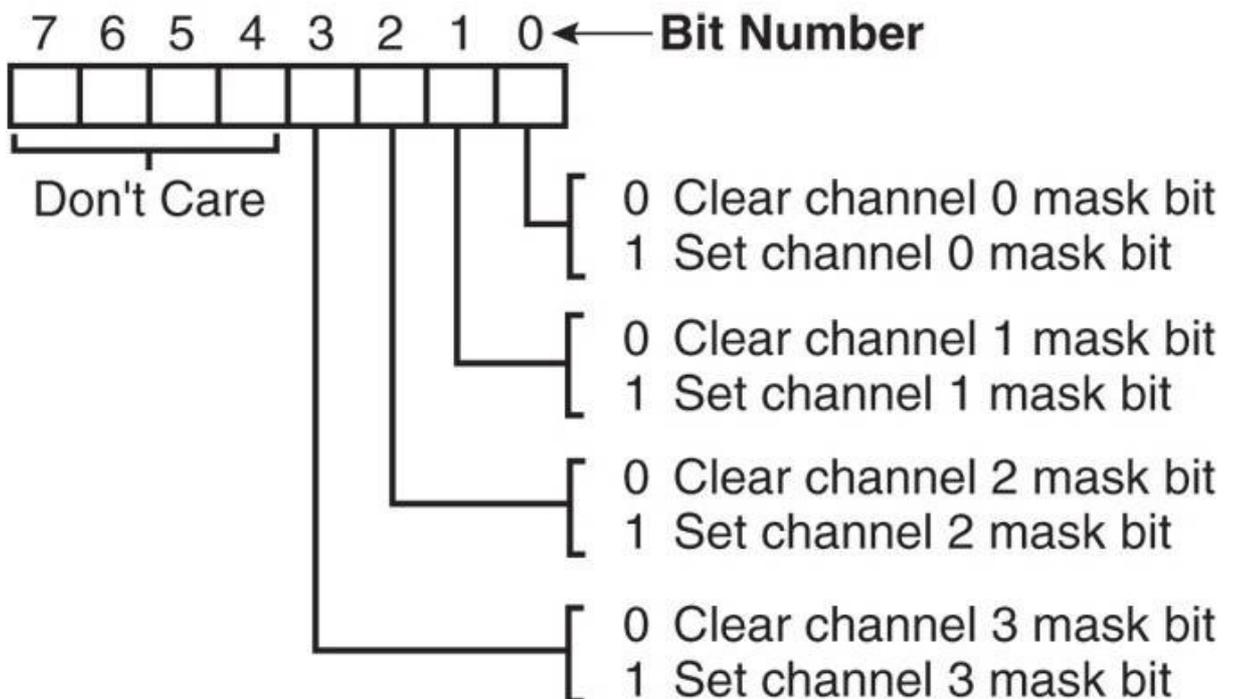
- The mask register set/reset sets or clears the channel mask.

1. if the mask is set, the channel is disabled
2. the RESET signal sets all channel masks to disable them



### MSR

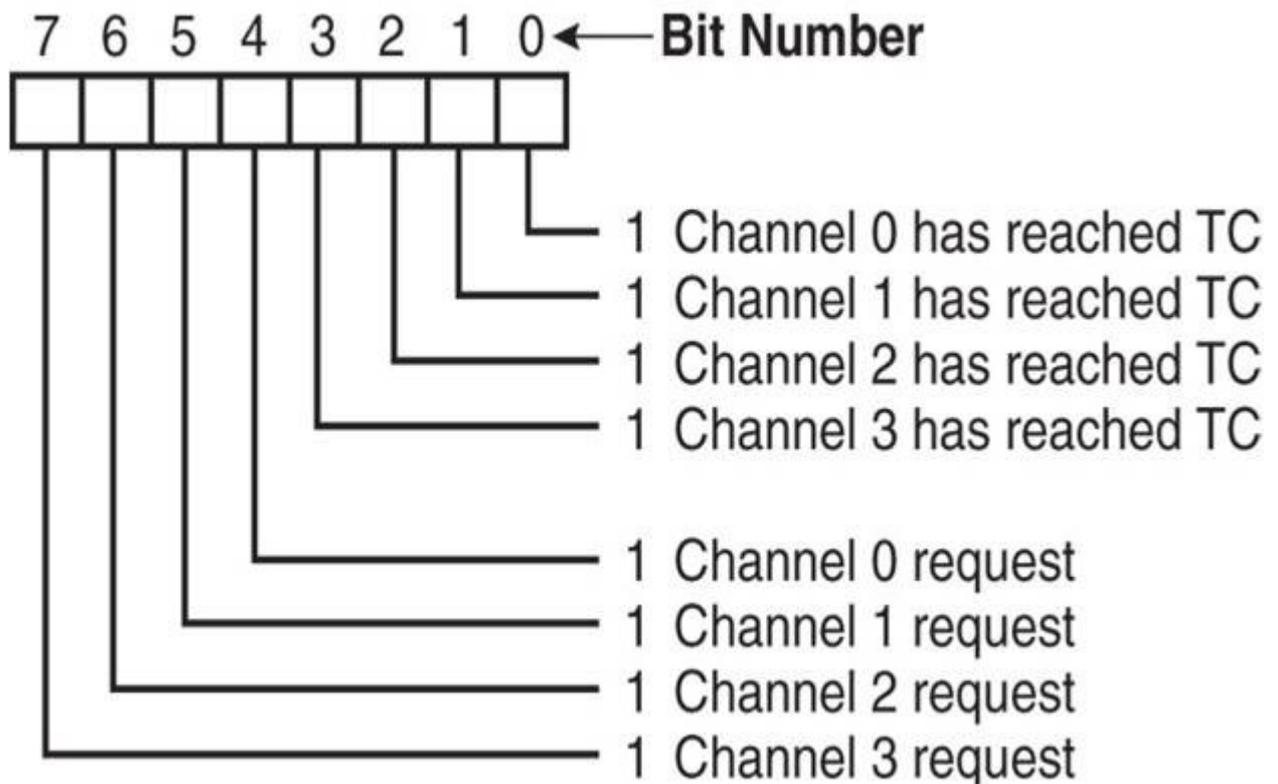
- The mask register clears or sets all of the masks with one command instead of individual channels, as with the MRSR.



### SR

- The status register shows status of each DMA channel. The TC bits indicate if the channel has reached its terminal count (transferred all its bytes).

- When the terminal count is reached, the DMA transfer is terminated for most modes of operation.
- The request bits indicate whether the DREQ input for a given channel is active.



## 8237 Software Commands

### ***Master clear***

Acts exactly the same as the RESET signal to the 8237. As with the RESET signal, this command disables all channels

### ***Clear mask register***

Enables all four DMA channels.

### ***Clear the first/last flip-flop***

Clears the first/last (F/L) flip-flop within 8237. The F/L flip-flop selects which byte (low or high order) is read/written in the current address and current count registers.

if F/L = 0, the low-order byte is selected if F/L = 1, the high-order byte is selected Any read or write to the address or count register automatically toggles the F/L flip-flop.

## UNIT-V

### 8051 MICROCONTROLLER

