CHAPTER - II

PC HARDWARE, SYSTEM SOFTWARE OVERVIEW & DIGITAL INTERFACING CONCEPTS

CHAPTER - TWO

PC HARDWARE, SYSTEM SOFTWARE OVERVIEW AND DIGITAL INTERFACING CONCEPTS

2.0 INTRODUCTION

A digital computer is a multiplexer, programmable machine that reads binary instructions from its memory, accepts binary data as input and process data according to the instructions and provides the results as output. The physical components of the computer are called **HARDWARE**. A set of instructions written for the computer to perform a task is called a **PROGRAM** and a group of programs is called **SOFTWARE**.

We are acquiring the digital information of the analog data with the help of ADC and making the analysis of the data with the help of PC. Once the PC acquires the data then there is no problem for manipulating the data as we wanted. Here the only tracking part is that to give the data to the PC. Therefore data interfacing techniques and the role of PC is very important in this DAS system. In discussing about the PC hardware overview and its principles of operation, we can say that it is common to all applications. Only difference is that the software differs from one application to another application. In most of the applications, the system software is same. The software supplied by the manufacturer of the computer is referred to as the system software. The basic system software supplied together with the computer consists of **OPERATING SYSTEM, PROGRAM DEVELOPMENT SOFTWARE and DIAGNOSTIC SOFTWARE**. For example, DOS (MS-DOS or PC-DOS) is a good system software. There is a Basic-Input-Output-System (BIOS), which is interface between the hardware and software. The BIOS is also a software but it is called the **FIRMWARE** due to its integration with hardware. We can say that, without BIOS, the PC is like a dead machine. The BIOS controls the hardware according to the requirements from the DOS.

2.1 IBM PC MOTHERBOARD

The fig.(2.1) shows the block diagram for the motherboard of the IBM PC. We see from the left side of the diagram, the 8080 CPU and the 8259A priority-interrupt controller. The next vertical line of devices to the right assist of the address bus buffers, the data bus buffers and the 8288 bus-controller chip. The bus controller chip is required because the 8088 is operated in maximum mode. The busses from these devices go across the drawing and connect to the 62 pin peripheral board connectors. The CPU then use these busses to communicate directly with the boards in the peripheral expansion slots. Now the ROM in the lower right, the keyboard logic etc. in the middle right, and the dynamic RAM in the upper right. Take a look at the column of devices which contains the 8237A-5 DMA controller. Starting at the bottom of this column we can see an 8253-5 programmable timer. Just above this a family of 8255A-5 programmable port device. Now we are left with just the three devices with DMA in their labels to ponder. The 8237A-5 is the DMA controller. The

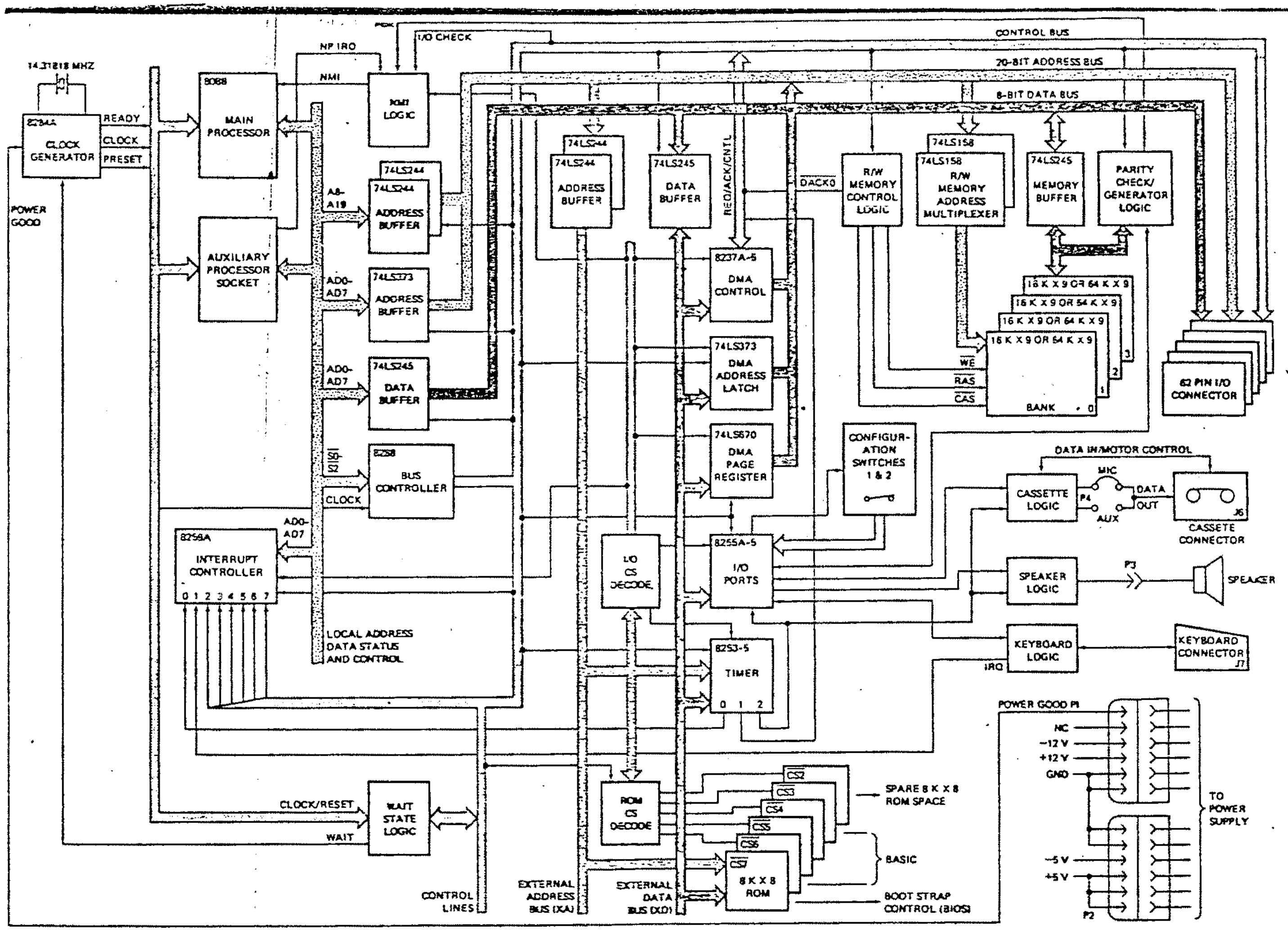


FIGURE 2.1 Block diagram of circuitry on IBM PC motherboard.

74LS373 under the DMA controller is used to grab the upper 8-bits of the DMA address sent out on the data bus by the 8237A-5 during a transfer. This device has the same function performed by octal latch. The 74LS670 is used to output bits A_{16}-A_{19} of the DMA transfer address, the same function performed by 8282 octal latch.

The clock generator continuously supplies free running clock to the microprocessor. When the power supplied is switched on, the clock generator resets 8088. The microprocessor initiates power-on sequence and it starts instruction fetch from the ROM location HEX FFFF0. The firmware begins from this location. The address latches use the trailing edge of ALE to latch the address sent on the local bus by the 8088 in T1. The bus controller decodes the bus status sent by the microprocessor and generates appropriate bus controller signals. The RAM decode logic, ROM decode logic and I/O port decode logic decodes the address on the address bus in order to enable the access of RAM, ROM or motherboard I/O ports by the microprocessor. The wait state logic decides the number of wait states to be introduced in different bus cycles of the CPU or DMA controller. The parity logic generates odd parity bit while writing data into RAM. During the read operation from RAM, if there is no odd parity, the parity logic generates parity error signal.

On receiving parity error signal, the NMI logic generates NMI which goes to the microprocessor. The NMI logic is generated under two more abnormal situations also:

i) Parity error while reading from daughterboard memory

ii) Malfunction defected by co-processor (co-processor interrupt)

The memory control logic has to be refreshed periodically. This is done by refreshing the rows one by one at an interval of 15 microseconds. For this purpose, the timer one is programmed by firmware to generate a request for every 15 microseconds which is issued to the DMA controller as DRQo. The DMA controller performs a bus cycle and sends the row address to be refreshed. All the DRAM chips are simultaneously enabled to refresh a particular row. This is achieved by bypassing the RAM decode logic.

During the normal reading from or writing into RAM, the row address and column address is sent to the RAM chip on the same input pins, through the row/column address multipliers. Row address is sent first and after a 60 ns delay column address is sent.

Since the address bus is common to 60th the microprocessor and the DMA needs to obtain bus sanction before starting a bus cycle. For this purpose the DMA controller issues activity by the microprocessor. On sensing the condition that the microprocessor does not need a bus request, HOLD, to "Bus Arbitration Logic" (BAL). The BAL constantly monitors the bus activity by the microprocessor. On sensing the condition that the microprocessor does not need a bus cycle immediately, the BAL issues a wait state request to the CPU logic and simultaneously BAL issues the bus sanction, HOLD ACK to the DMA controller.

When the bus is with the DMA, the CPU signals (address, data and other control signals) are prevented from entering into the system bus by tri-stating the appropriate outputs. Similarly, when the microprocessor has the bus control, the DMA outputs are disabled from interfacing with the system bus.

The interrupt logic receives different interrupt requests and generates a common interrupt request (INTR) which goes to the microprocessor. The microprocessor in turn, interrogates the interrupt controller by performing two chained interrupt acknowledgement cycles. In response, the interrupt controller sends a vector code on the data bus. The vector code contains information about the interrupt level for which the interrupt controller has sanctioned priority. The CPU accordingly branches to the corresponding interrupt service routine.

The keyboard interface receives serial data and clock from the keyboard and converts the data into parallel byte which contains the scancode for the keypressed. The keypressed interface rises an interrupt request (IRQ1) after assembling one scancode. The keyboard interrupt service routine (BIOS) converts this scancode into ASCII code. The configuration logic (mode switch input logic) senses the ON/OFF condition of the mode switches (DIP switch). This information is stored in the memory by the firmware for the future reference.

2.2 I/O PORT ADDRESSES

The port addresses in PCs are 16 bits considering the 8080's I/O mapped I/O scheme. Theoretically the PC can address 64 kilo input ports and 64 kilo output ports. The CPU addresses the I/O ports through IN (input) and OUT (output) instructions. The I/O addresses Hex 000 to 0FF are reserved for the motherboard. The address Hex 100 to 3FF are available for use in daughter boards. Here the address Hex 300 to 31F are available for I/O mapping in the expansion slot (i.e. for the use of Prototype Card).

Following Table 2.1 shows an I/O address map for PC.

Table 2.1 : I/O address map :

HEX RANGE	USES
000 - 00F	DMA chip 8237A - 5
020 - 021	Interrupt 8259 - A
040 - 043	Timer 8253 -5
060 - 063	PPI 8255A - 5
080 - 083	DMA page registers
0AX	NMI Mask Register
0CX	Reserved
0EX	Reserved
200 - 20F	Game control
210 - 217	Expansion Unit
220 - 24F	Reserved
278 - 27F	Reserved
2F0 - 2F7	Reserved
2F8 - 2FF	Asynchronous communications
300 - 31F	Prototype card
320 - 32F	Fixed disk
378 - 37F	Parallel printer
380 - 38F	SDLC communications
3A0 - 3AF	Reserved

3B0 - 3BF	IBM monochrome display
3C0 - 3CF	Reserved
3D0 - 3DF	color/graphics
3E0 - 3E7	Reserved
3F0 - 3F7	Diskette
3F8 - 3FF	Asynchronous communications

2.3 I/O CHANNEL

The I/O channel is an extension of the microprocessor bus. It is however, de-multiplexed, repoured and enhanced by the addition of the interrupts and direct memory access (DMA) functions.

The I/O channel contains an 8 bit bidirectional data bus, 20 address lines, 6 levels of interrupts, control lines for memory and I/O read or write clock and timing lines, 3 channels of DMA control lines, memory control lines, a channel check line and power and ground for the adapters. Four voltage levels are provided for I/O cards +5 Vdc, -5 Vdc, +12 Vdc and -12 Vdc. These functions are provided in a 62 pin connector with 100 mil card tab spacing.

A ready line is available in the I/O channel to allow operation with slow I/O or memory devices. If the channels ready line is not activated by an addressed devices all processors generated memory read and write cycles take four 210 ns clock or 840 ns byte. All processors generated I/O read and write cycles require 5 clocks for a cycle time of 1.05 µs byte. All DMA transfers require five clocks for a cycle time of 1.05 µs byte. Refresh cycles occur once every 72 clocks (approximately 15 μs) and require four clocks or approximately equal to the bus bandwidth.

I/O devices are addressed using I/O mapped address space. The channel is designed so that 768 I/O device address are available to the I/O channel cards.

A channel check line exist for reporting error conditions to the processor. Activating this line result in a Non-Maskable Interrupt (NMI) to the 8Ø8Ø processor. Memory expansion options use this line to report parity errors.

The I/O channel is repoured to provide sufficient drive to power all eight (J1 through J8) expansion slots, assuming two low power schhottky (LS) loads per slot. The IBM I/O adapters typically use only one load.

Timing acquirement on slot J8 are much stricter than those on slots J1 through J7. Slot J8 also requires the card to provide a signal designating when the card is selected.

The fig.(2.2) shows the " system expansion slots " in the components layout diagram for the IBM PC motherboard.

2.4 I/O CHANNEL DESCRIPTION

The following is a desoription of the IBM Personal Computer XT I/O channel. All lines are IBM compatible. The fig.(2.3) shows the Pin names and numbers for peripheral slots on IBM PC motherboard.

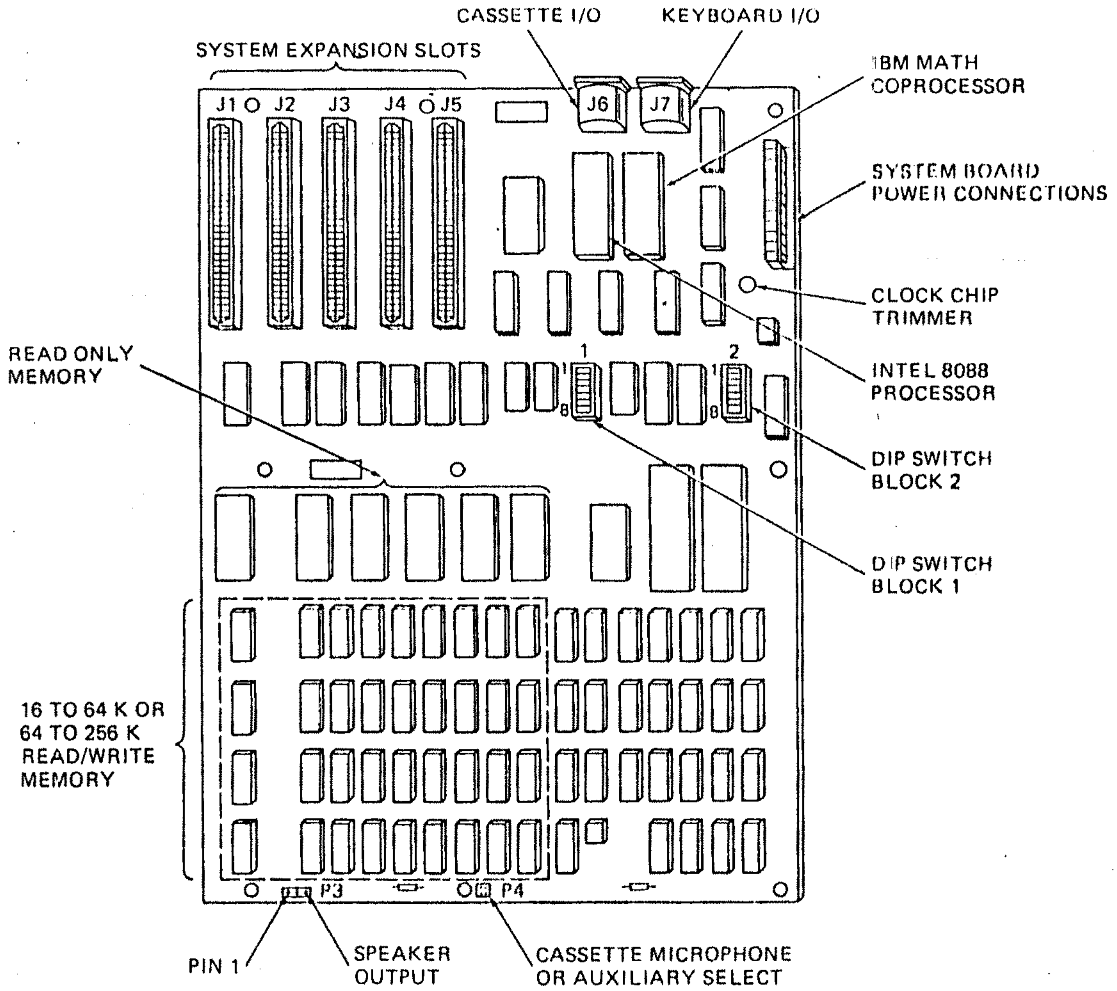
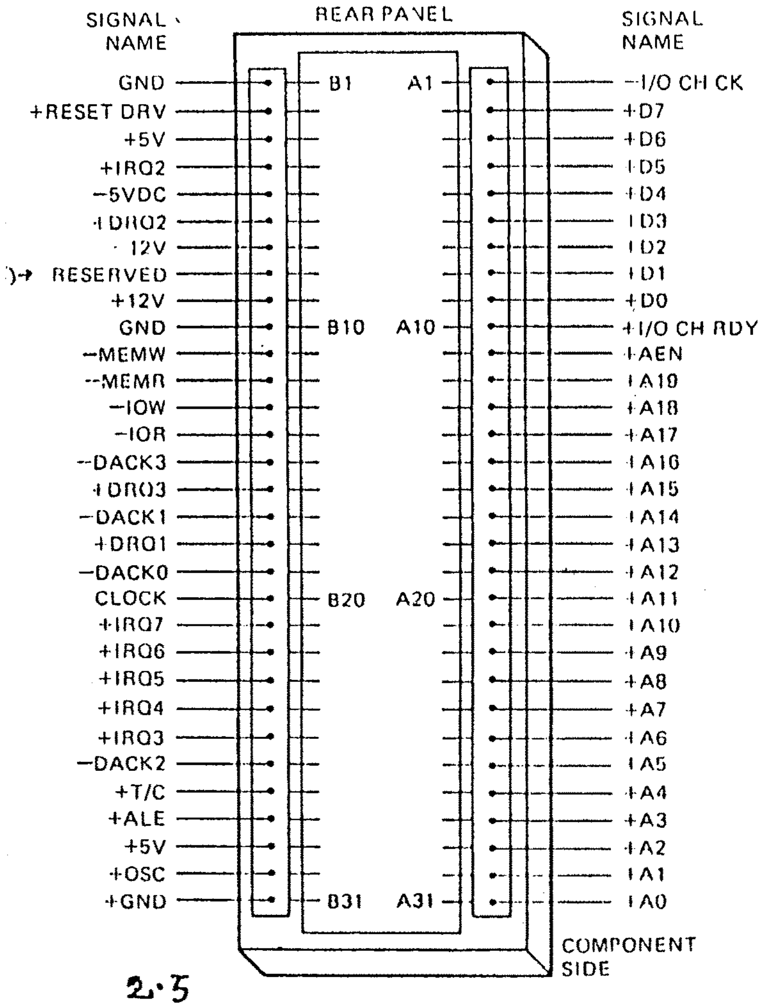


FIGURE 2.2 [redacted] Component layout diagram for IBM PC motherboard.



2·5

FIGURE [redacted] Pin names and numbers for peripheral slots on IBM PC motherboard.

Signal	I/O	Description
OSC	O	**Oscillator**: High speed clock with a 70-ns period (14.31818 MHz). It has 50% duty cycle.
CLK	O	**System clock**:It is divide by three of the oscillator and has a period of 210 ns (4.77 MHz). The clock has a 33% duty cycle.
RESET-DRV	O	This line is used to reset or initialize system logic upon power-up or during a low-line voltage outage.This signal is synchronized to the falling edge of clock and is active high.
A_0-A_{19}	O	**Address bits 0 to 19**: These lines are used to address memory and I/O devices within the system. The 20 address lines allow access of up to 1 megabyte of memory. A_0 is the least significant bit (LSB) and A_{19} is the most significant bit(MSB). These lines are are generated by either the processor or DMA controller. They are active high.
D_0-D_7	I/O	**Data bits 0 to 7** : These lines

		provide data bus bits Ø to 7 for the processor, memory, and I/O devices. $D_{\emptyset}$ is the least significant bit(LSB) and D_7 is the most significant bit(MSB). These lines are active high.
ALE	O	**Address latch enable**: This line is provided by the 8288 bus controller and is used on the system board to latch valid addresses from the processor. It is available to the I/O channel as an indicator of a valid processor address (when used with AEN). Processor addresses are latched with the falling edge of ALE.
$\overline{\text{I/O CH CK}}$	I	**-I/O channel check**: This lines provides the processor with parity (error) information on memory or devices in the I/O channel. When the signal is active low a parity error is indicated.
I/O CH RDY	I	**I/O channel ready**: This line, normally high(ready) is pulled

		low (not ready) by any memory or I/O device to lengthen I/O or memory cycles. It allows slower devices to attach to the I/O channel with minimum of difficulty. Any slow device using this line should drive it low immediately upon detecting a valid address and a read or write command . This line should never be held low longer than 10 clock cycles. Machine cycles(I/O or memory) are extended by an integral number of CLK cycles (210 ns).
IRQ2-IRQ7	I	**Interrupt request 2 to 7**: These lines are used to signal the processor that an I/O device requires attention.They are prioritized with IRQ2 as the highest priority and IRQ7 as the lowest priority. An Interrupt Request is generated by raising an IRQ line (low to high) & holding it high until it is acknowledged by the processor (interrupt service routine).
$\overline{\text{IOR}}$	O	**-I/O Read command** : This command

		line instructs an I/O device to drive its data onto the data bus. It may be driven by the processor or the DMA controller. This signal is active low.
$\overline{IOW}$	O	-**I/O Write command** : This command line instructs an I/O device to read the data on the data bus.It may be driven by the processor or the DMA controller. This signal is active low.
$\overline{MEMR}$	O	-**Memory read command** : This command line instructs the memory to drive its data bus. It may be driven by the processor or the DMA controller. This signal is active low.
$\overline{MEMW}$	O	-**Memory write command** : This command line instructs the memory to store the data present onto data bus. It may be driven by the processor or the DMA controller. This signal is active low.
AEN	O	**Address Enable**: This line is used to

		de-gate the processor and other devices from the I/O channel to allow transfer to take place. When this line is active (high), the DMA controller has control of the address bus, data bus, read command lines (memory and I/O), and the write command lines(memory & I/O).
T/C	O	**Terminal count** : This line provides a pulse when the terminal count for any DMA channel is reached. This signal is active high.
$\overline{\text{CARD}}$ $\overline{\text{SLCTD}}$	I	-**Card selected** : This line is activated by cards in expansion slot J8. It signals the system board that the card has been selected and that appropriate drivers on the system board should be directed to either read from , or write to expansion slot J8.Connectors J1 through J8 are tied together at this pin, but the system board does use their signal. This line should be driven by an open collector device.

The following voltages are available on the system -board I/O channels :

+5 Vdc +/- 5% ; located on 2 connector pins.

-5 Vdc +/-10% ; located on 1 connector pin.

+12 Vdc+/- 5% ; located on 1 connector pin.

-12 Vdc +/-10% ; located on 1 connector pin.

GND ; located on 3 connector pins.

2.5 SYSTEM SOFTWARE

The software supplied by the manufacturer of a computer is referred to as the system software and the program developed by the user are grouped under the application software. The basic system software supplied together with the computer consist of :

1) Operating system,

2) Program development software and

3) Diagnostic software.

2.5.1 OPERATING SYSTEM : An operating system is a collection of programs that link a central processing unit with the external world through a well defined set of commands for the development and execution of application programs. An operating system performs the following functions :

i) Manages the use of system resources,

ii) Allocates CPU resources to tasks (programs),

iii) Activates, suspends and destroys tasks,

iv) Perform memory management,

v) Perform movements of files among storage media and

vi) Handles I/O and interrupts.

Since the configuration of computer installations can vary from application to application, the facilities provided by the operating systems vary accordingly. However, the following parts of the operating system software are common to all computer installations :

a) Monitor or executive or supervisor,

b) Task scheduler,

c) Command interpreter and

d) Drivers.

It may be noted that operating systems on microcomputers are now generally used but in most applications microcomputers are dedicated to performing a single task or three to four task and a background/foreground mode of operation is performed.

The monitor is considered to be the heart of an operating system. Its main purpose is to control the processing of user programs and provides additional facilities designed to suit particular areas of applications.

The task scheduler which is used in multiprogramming environment determines the next program to be run when the provides one has been completed or suspended due to an I/O or errors. There is a great variety of scheduling algorithms based on queuing theory which are used in scheduler programs.

The command interpreter is a link between executive monitor and user. The user communicates with the computer system through a suitably defined set of commands e.g. shell is command interpreter in Unix.

Drivers are programs which operate peripheral devices through their hardware control units. The executive acts as the interface between the user program and the driver. A user can use the peripheral devices via the monitor which passes the request to the appropriate driver.

2.5.2 PROGRAM DEVELOPMENT SOFTWARE

The program development software is needed by the user to help him in coding, updating and debugging the programs. The set of programs contribute to a program development software consists of the following :

i) Text editor, ii) Language translator (assembler and compiler), and iii) Debugging routines.

i) Text editor - The text editor program allows the text to be entered, stored and corrected. The editor provides facilities to delete or replace some of the pieces of a text, insert new sections and so on. A file manager program allows the text to be filled and retrieved by name. There are two kinds of editor programs; linear and contextual. The first one works on a line by line basis. To change a character in a line, the whole line must be deleted and the correct one retyped. The contextual editor permits the handling of a string of character at any place in the program.

ii) Language translator - A programing language (for e.g. Basic, Pascal, C etc.) is an interface between a users problem and a competing system. It has the following three attributes:

SYNTAX - A set of rules that define the writing of correct statement and expression of a language.

SEMANTICS - The interpretation of statements and assigning meaning to them.

DATA STRUCTURE - The data can be handled by the basic operations of a language.

Programming language differ in all three types. Assembly language programming allows a user to write his programs in symbolic code which corresponds to CPU hardware instructions. Assembly language-level programming is more convenient way of programming than machine level programming since it uses "mnemonics" to denote the operation code which is in octal or hexadecimal and "symbolic addresses" to denote the operands.

An assembler program translates the symbolic instructions into the binary machine code; the "object-code". A compiler program is used for high level translation. After being stored, the object code is executed by the computer, one statement at a time. It may be noted that different assemblers or compilers accepting the same source program can produce object codes with differing memory utilization, speed, number and type of listing. Assemblers are very important since in some process control application only an assembler can offer an acceptable solution. Almost, most of the system software, (e.g. monitor, scheduler etc.) is written in assembly level programming as well as driver routines of various peripheral devices.

A compiler is a program that translates another program written in high level language to the assembly language or machine language of specific microcomputer. The compiler program is written in the assembly language of the particular microprocessor. Examples of a compiler languages are Pascal, Cobol etc.

An interpreter on the other hand, is a program that translates and execute each source element individually. No object code is stored and loaded into the microcomputer memory for execution, as is done in the case of compilers. A disadvantage of interpretive language is that they result in slower program execution. However, with their interactive ability, the program development time is required considerably. Basic is the most commonly used interpretive language.

iii) Debugging routines - The debugging allows suspending the execution of the object code at several break-point specified prior and displaying the contents of the chosen processor resistor and memory locations. In some advanced debugging program it may be possible to change some locations and then test the influence of the change.

Figure 2.4 shows the elements of system software.

2.6 THREE-TIER COMMUNICATION IN A PC

Generally, the PC can be used for most of the applications in the following ways -

1) For running higher level languages like BASIC, COBOL, PASCAL etc.

2) For running programs written in application packages like DBASE, LOTUS 1-2-3 etc.

3) For simple operations through packages like WORDSTAR, LETTER EDITOR, GRAPHER, FLOW, PRINT-MASTER etc.

For above each uses appropriate SYSTEM SOFTWARE should be loaded into the PC. Some examples are -

i) For running of COBOL program the COBOL compiler should be loaded into the memory.

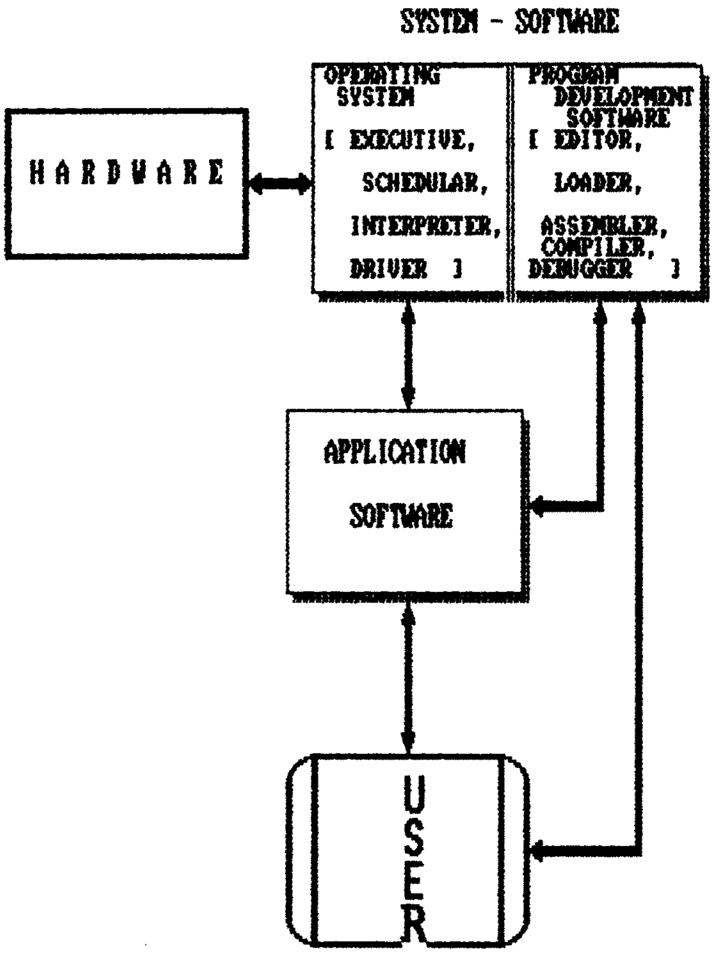


FIG.(2.4)SYSTEM SOFTWARE ELEMENTS.

ii) For running DBASE program, the DBASE package should be loaded into the memory.

iii) For preparing graphs in various forms, the GRAPHER, LOTUS 1-2-3 packages should loaded previously.

iv) For preparing letters LETTER EDITOR ,WORD STAR should be used.

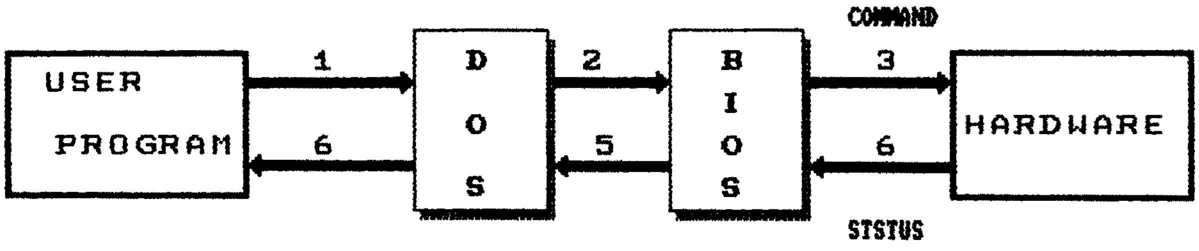
From all the above examples ; it is seen that the common requirement is the loading of the "OPERATING SYSTEM". The IBM-PC uses PC-DOS or MS-DOS. These both operating systems are identical in many aspects and developed by a company called MICROSOFT. With DOS, there is a no use of PC to the common user. DOS communicates with the hardware units and application programs. But the communication of the DOS with the hardware is through the BIOS program which are stored in ROM. With out BIOS ROM, there is a no use of operating system for the PC. This communication process between "USER PROGRAM-DOS-BIOS-HARDWARE" is called " THREE-TIRE-COMMUNICATION " in a PC.

Fig.(2.5) shows the THREE-TIRE-COMMUNICATION in a PC. This communication in a PC is not usually visible to the application programs.

Procedure for a THREE-TIRE-COMMUNICATION :

1) An application program calls DOS for performing specific functions like reading from the keyboard, printing some lines, displaying some message on the CRT monitor etc.

2) The DOS finds out the details and requirements of the application program, e.g. which file has to be printed or displayed. Then the DOS calls the respective I/O drivers in BIOS. At the



1 , 2 , 3 ➡ CALL; 4 , 5 , 6 ➡ RETURN

FIG. (2.5) THREE-TIRE-COMMUNICATION-IN A PC.

same time the DOS supplies parameters like memory address, number of bytes etc. to the BIOS.

3) The BIOS in turn issues appropriate commands to the hardware.

Thus there is a three-tire-communication in a PC.

2.7 DIGITAL INTERFACING

The primary function of computer is to accept data from input devices such as keyboards and A/D converters, read instructions from memory, process data according to the instructions and send the result to the output devices such as printer, video monitor etc. These input and output devices are called either peripherals or I/Os. Here the memory can be considered as a special type of I/O. Designing logic circuits and writing instructions to enable the computer to communicate with these peripheral is called interfacing and the logic circuit are called I/O ports or interfacing devices. The computer communicates with the peripherals in either two formats : asynchronous or synchronous. Similarly, it transfers data in either two modes: Parallel I/O and Serial I/O. Computer identifies peripherals either as memory-mapped I/O or peripheral I/O based on their interfacing logic circuits. Data transfer between the computer and its peripherals can ~~be~~ take place under various conditions. The modes, the techniques, the instructions and the conditions of data transfer are summarized in the fig.(2.6).

2.7.1 BASIC CONCEPTS : The approach an designing an interfacing circuit for an I/O device is determined primarily by the instructions to be used for data transfer. An I/O device can be interfaced with the computer either as a peripheral I/O or as a

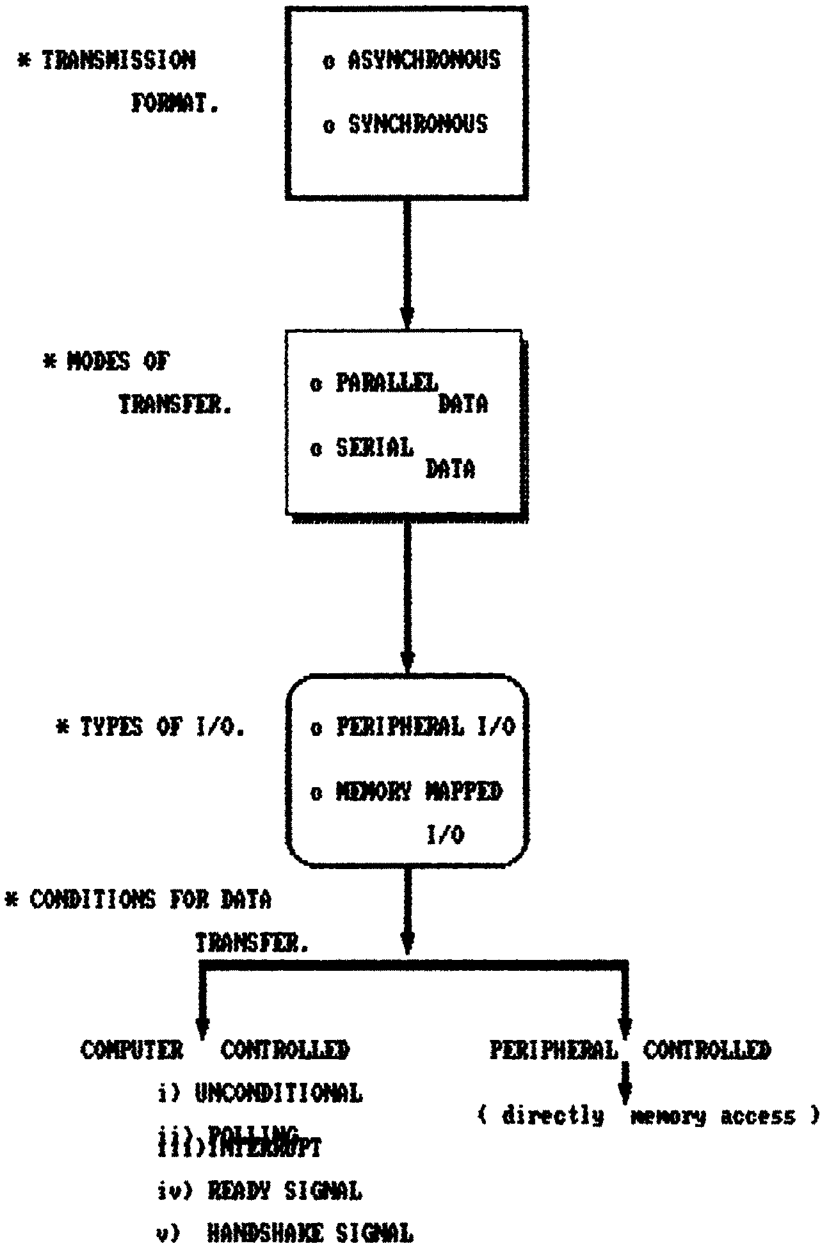


FIG. (2.6) PROCESS OF DATA TRANSFER BETWEEN THE COMPUTER & PERIPHERALS.

memory mapped I/O. In the peripheral I/O, the instructions like IN/OUT are used for data transfer, and the device is identified by an 8-bit address. In the memory mapped I/O, memory related instructions are used for data transfer and the device is identified by a 16-bit-address. THe basic concepts in interfacing I/O devices are similar in both methods.

In the peripheral I/O the instruction IN inputs data from an input device into the accumulator and the instruction OUT send the contents of the accumulator to an output device. The memory mapped I/O technique uses the memory related data transfer instructions and memory control signals to transfer data between the accumulator and an I/O device.The memory mapped I/O technique is similar in many ways to the peripheral I/O technique. For example, in memory mapped I/O for data transfer LDA, STA and for memory control signals memory read & memory write (MEMR & MEMW) instructions are used. In memory I/O there is a necessity of decoding all the bits of address.

2.7.2 DEVICE SELECTION : In general, peripherals are connected in parallel on the data and address buses. To select an appropriate peripheral, the device address on the address bus and the control logic can be used as follows:

i) Decodes the address bus to generate a unique pulse corresponding to the device address on the bus; this is called device address pulse.

ii) Combine (AND) the device address pulse with the control signal to generate a device select pulse that is generated only when both signals are asserted.

iii) Use the device select pulse to activate the interfacing device (I/O port).

2.7.3 ABSOLUTE VERSUS LINEAR-SELECT DECODING : In the decoding concept, all the address lines are decoded to generate one unique output pulse and the device will selected. This concept is called absolute decoding. Absolute decoding is a good design practice, but it is costly to decode the all lines.

To minimize the decoding cost (component cost) one address line and control signal are combined to generate the device select pulse. This concept is called the linear-select decoding. As a result, the device has multiple address.

As interfacing device, a latch is used for an output port and a tri-state buffer is used for an input port. The address bus can be decoded by using either the absolute or the linear select decoding technique. A decoder or discrete gate can be used for absolute decoding and address line can be used directly for linear select decoding. The linear select decoding technique reduces the component cost, but the I/O device ends up with multiple addresses.

Some examples of interfacing peripherals (I/Os) to the PCs are : i) Interfacing input keyboard,

ii) Interfacing output displays (e.g. LED display for binary data) and

iii) Interfacing memory.

REFERENCES :

1) **Microprocessor Architecture, Programming And Applications with the 8085/8080A,**

-R.S.Gaonkar

Wiley Eastern Limited, 1986.

2) **Microprocessor And Interfacing Programming and Hardware,**

-Douglas V. Hall

Mcgraw-Hill International Editions

Computer science series, 1986.

3) **Appel Instrumentation and Control Circuits and Software,**

-J.E.Olesky

A Reston book, Prentice-Hall, INC.Eglewood Ciffs,

New Jersey, 1986.

4) **IBM PC and Clones, hardware, Trobleshooting and Maintenance**

-B. Govindarajalu

TataMcGraw-Hill Company Ltd.New Delhi, third edition.

5) **Personal Computer XT systems, Technical Reference.**

6) **Designing Microprocessor Based Instrumentation,**

- J.J. Carr