UNIT 7 – INPUT/OUTPUT ORGANISATION

Structure

7.0 Introduction
7.1 Objectives
7.2 Input/Output (I/O) Devices
7.3 The Input/Output (I/O) Interface
7.3.1 System Bus and I/O Interface Modules
7.3.2 I/O and Memory Bus
7.3.3Isolated and Memory-Mapped I/O
7.4 Device Controllers
7.4.1 Device Controller and I/O Interface
7.5 Device Drivers
7.6 Asynchronous Data Transfer
7.6.1 Strobe Control
7.6.2 Handshaking
7.7 Input/Output (I/O) Techniques
7.7.1 Programmed I/O
7.7.2 Interrupt-Driven I/O
7.7.3 Interrupt Handling
7.7.4 Direct Memory Access (DMA)
7.8 Input Output Processor (IOP)
7.8.1 Characteristics of I/O Channels
7.9 External Communication Interfaces
7.10 Summary
7.11 Solutions /Answers

7.0 INTRODUCTION

In the preceding units, you have learned the concepts of the memory system for a computer system. The memory system of a computer includes primary, secondary and auxiliary, and high-speed memories. As discussed, the main memory of thecomputer system is used for storing the instructions and data of the programs, which are getting executed. To execute the program, the computer may need some data which is known as input.The program execution results in the creation of processed data, which is known as output. In addition to the memory system, another important component is the input and output system which is used to receive/send data from/to the external environment. This unit introduces you to various Input/Output (I/O) techniques and controllers, device drivers, structure ofI/O interface, and asynchronous data transfer.This unit also explains the I/O processor which is exclusively used for I/O operations.

7.1 OBJECTIVES

After going though this unit, you should be able to:

- define the structure of input/output (I/O) interface and I/O devices;
- explain the structure of controllers;
- explain different data transfermodes;
- explain various techniques used for Input/Output in a computer system;
- discuss the need of an input/output (I/O) processor;
- explain the role of external serial and parallel communication interfaces;
- explain the concepts of interrupt processing

7.2 INPUT/ OUTPUT (I/O) DEVICES

The input/output devices are used by a computer system to provide an interface with the external environment i.e., humans and other devices, which are connected to a computer. Themajor components of a typical microcomputer system are *microprocessor/CPU, memory* *system* and an *I/O interface*.These components are connected through buses. The primary function of the system bus is to transfer control information, addresses, instructions, and data between these components. Figure 7.1 depicts the block diagram of a typical microcomputer system.

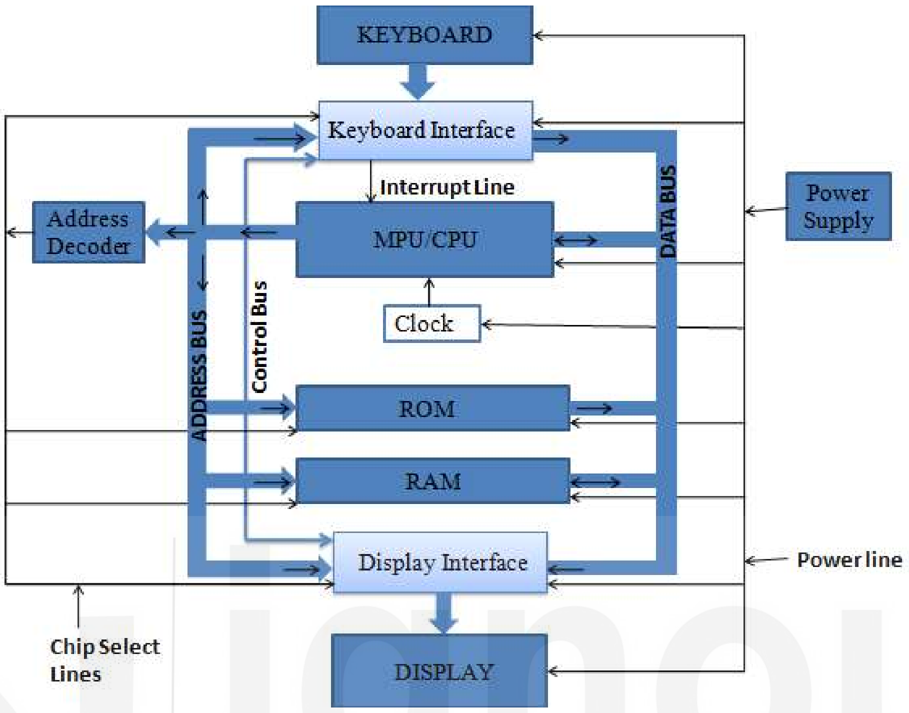


Figure 7.1: A Typical Microcomputer System's Block Diagram

The microcomputer, as shown in Figure 7.1, has a single micro processing unit (MPU). It also has RAM and ROM, which may be constructed by a number of RAM and ROM chips. The diagram also shows two basic interfaces, viz. keyboard interface and display interface, which are connected through the system bus. You may please note that other I/O interface may be connected in a similar way. Please also note that the system bus has been shown in the diagram as three distinct buses, viz. control bus, address bus, and data bus.

An Input/Output (I/O) subsystem includes all the input/output interfaces and connected Input/output devices.The basic objective of an I/O subsystem is to provide an efficient communication medium between the computer system and the external environment (humans and other devices).An I/O interface is used to connect an external I/O device with the computer system. The I/O interface interchanges control, status and data with the external device. The I/O interfaces can also be used to transfer instruction/data/control within the computer units, including processor registers and memory units.An I/O device attached to the computer is also known as **peripheral**or **external device**. The external devices may be categorized on the basis of their communication endpoints as:

- **Human readable**: These devices provide information in human readable form. Example-*display terminals*,*printers,* etc.
- **Machine-readable:**These devices provide information in machine readable form. Example-*magnetic disks*, *CD-RW,* etc.
- **Communication:**These devices provide information to communication devices such as *MODEM*.

7.3 THE INPUT/OUTPUT (I/O) INTERFACE

AnI/O interface (also known as **I/O Module**) is used totransfer information between internal memory and peripheral devices. Each peripheral device has its own set of characteristics. An I/O interface helps in resolving the differences between the processing unit components and peripherals. The following table lists major differences:

Table 7.1: Peripheral devices and processing unit components

Sr. No.	Processing unit components	Peripheral Devices	Remarks
1	Processor and memory are electronic devices.	Peripheral devices are electromagnetic devices.	Hence, they perform operations in a different manner
2	Data transfer rate of processor is very fast.	Data transfer rate of peripheral devices is slow.	Thus, a synchronization mechanism is required.
3	Processor uses word format.	Peripherals, in general, use bytes/blocks format.	
4	Processor may communicate either directly with different manners or indirectly using an interface in similar fashion.	Each peripheral device may have a different operating mode.	Thus, an interface to handle different operating modes is required.
5		Each peripheral device must be controlled in such a way that the operation of one does not disturb the operation peripheral device.	

To address the issues mentioned in Table 7.1, ahardware component between the peripheral devices and CPU is required to control and synchronize all input/output operations in the computer systems. These hardware components, which are used to provide an interface between the peripheral devices and the processor, are known as**interface units**.An I/O interface acts as a bridge between the processor and peripheral devices.
The I/O interface offers an interface which connects the internal components i.e.,processor and main memory as well as external components i.e., peripheral or external devices. In addition to data transfer between processor to I/O devices, it also establishes the coordination between them. Moreover, the I/O interface also hascomponents like **buffer system** and **error detection** mechanism to deal with the speed differences between processor and peripheral devices.

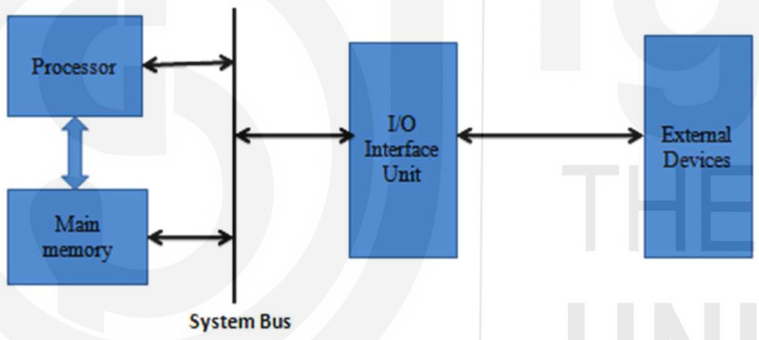


Figure 7.2: A Block Diagram of an I/OInterface connectingwith Processor and External Devices

Roles of I/O Interface

The major roles of I/O interfacesare:

1. Communication with the processor

To provide data transfer between processor and I/O interface, the control signals (Read/Write/Status), address of memory locations and actual data are required to exchange. The system bus works as a communication medium in order to facilitate the exchange of information/data between the processor and I/O interface. In system bus, the addressbus is used to send addresses; the control bus for control signals whereasthe data bus is employed for actual data transfer between processor and I/O interface.

2. Synchronization of control and timing signals

The I/O interface shares system bus with memory in order to provide data transfer. The timing and control signals are required in order to synchronize the flow of data from peripheral devices to processor/memory and vice versa. For instance-An I/O interface may request for control of data bus for data transfer from a peripheral device to the processor.

3. Communication with the I/O device

To complete an I/O operation, the exchange of data between I/O device and I/O interface is necessary. AnI/O operation may require sendingstatus signals, commands, or data.

4. Provision for data buffering

Due to the speed mismatch between the I/O devices and processor, the facility for data storage for a temporary period is required. The I/O interface stores information in registers temporarily which is referred to as data buffering. An I/O operation occurs in short bursts and the data are temporarily stored in a buffer area of I/O interfaceuntil the peripheral device/processoris ready to receive the data. Therefore, it is not required to hold the system bus for I/O operation due to slower I/O devices.

5. Error detection mechanism

The I/O interface also provides an in-built mechanism for error detection to check the communication errors along withmechanical errors. The errors are reported to the processor using parity bits and other mechanisms for further actions. Some mechanical errors may occur in devices such as mechanical and electrical failures, printer's paper jam etc.

In Figure 7.3, the internal block diagram of I/O interface unit consisting of different signals for processor as well as for I/O device is shown.

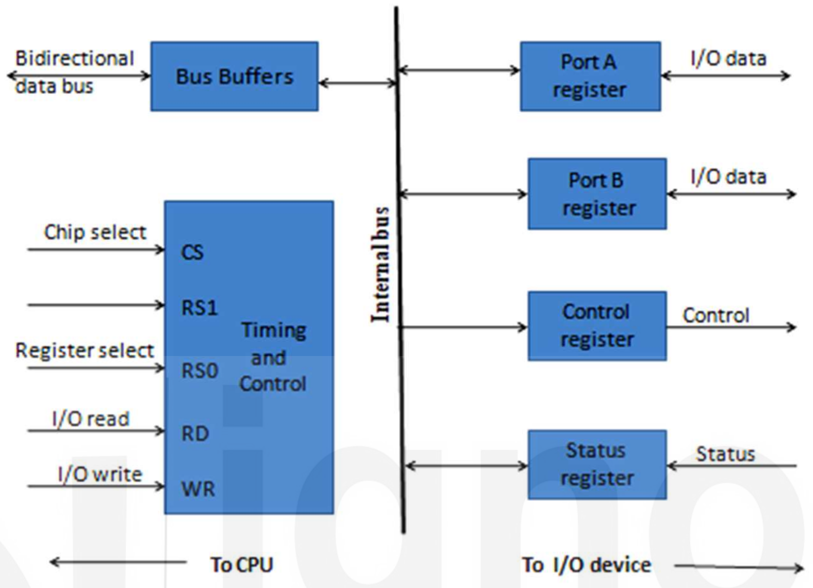


Figure 7.3: I/O Interface Unit's Block Diagram

7.3.1 System Bus and I/O Interface Modules

The control bus, data bus and address bus form a single bus which is known as system bus and it is used to establish the connection between the processor and I/O devices. Figure 7.4 depicts the communication links between the processor and several peripheral devices.

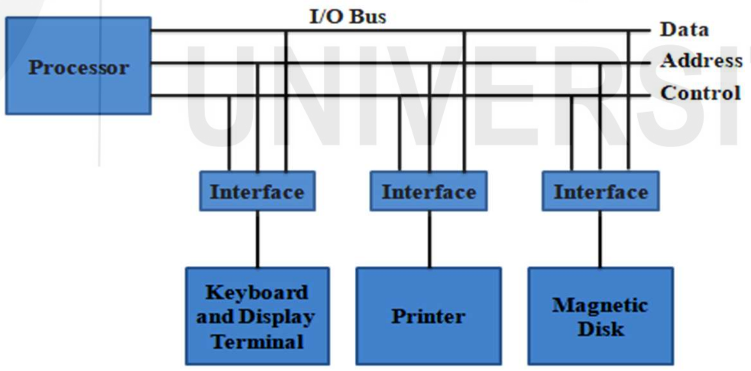


Figure7.4: Communication links between Processor and Peripheral devices

The I/O bus is connected to all peripheral devices through I/O interfaces. In order to communicate with an intended device, the processor sendsthe address of the device using address bus. Every I/O interface consistsof an address decoder to continuously monitor the content of address bus. When anI/O interface observes the address of its own peripheral device, it activates the associated device and the bus; otherwise, the peripheral device is disabled. A control signal (*I/O command*) is also provided simultaneously through the control bus. The four types of I/O commands that are arising out of the processor are as follows:

1. **Control Command:** This is the control signal code which is sent to the corresponding peripheral device and informs it about the action it has to perform.
2. **Status Command:** The is the statussignal which is used to test status conditions of the peripheral devices and interface. Some status commands are BUSY, DATA AVAILABLE, ERROR or NOT IN BUFFER etc.

3. **Data Output Command:** The signal/commandis utilized to activate the I/O interface for data transfer from the processor to buffer of I/O interface. The data from the buffer is ultimately sent to the peripheral device. The data is sent from the CPU to the buffer of interface after this command is provided.
4. **Data Input Command:** The processor sends this signalwhenever there is a need to read data from data any peripheral device. After the issuance of this command, the data from the intended peripheral device are extracted into the interface's buffer and this is followed by the data read operation by the processor.

7.3.2 I/O and Memory Bus

Aprocessor requires communicating with the I/O devices and memory system. The system bus (I/O Bus & Memory Bus) is used to control the exchangeof data among the processor, memory system and I/O devices. The I/O bus and memory bus; both busesconsist of data, address and control lines. To establish communication with the memory and I/O devices, the system bus can be used as follows:

i. One bus for each memory system and I/O.
ii. Shared data and addressbuses for both I/O devices and memory system but exclusive control bus for both.
iii. Shared system bus for both memory system and I/O devices.

7.3.3 Isolated and Memory-Mapped I/O

In case ofisolated I/O, the data and address buses are shared between I/O and memory but separate read/write control lines are used for I/O devices. Wheneverthe processor decodes instruction for an I/O device, it places the address on the address line and activates I/O read or write control line which causes data transfer between CPU and I/O device.
In other alternative, the computer employs the same set of read/write signals for I/O and memory and does not differentiate between I/O and memory addresses. This configuration is known as memory-mapped I/O.

7.4 DEVICE CONTROLLERS

The system bus is used to establish communication between the processor and all components including I/O devices. Some components are directly connected to the processor through system while some components are not directly connected to system bus.The intermediate electronic device, known as *device controller*, is used to connectthe I/O device and the system bus. The I/O device is connected at one end while it is connected with the system bus on another end. Thus, a device controller works as an interface to provide the communication medium between the system bus and I/O device.

Device Controller

It is not necessary that a device controller controls a single peripheral device. It can generally control more than peripheral devices. The device controller is available as an electronic circuit board, which can be directly plugged into the system bus. The device controller is also connected through a cable with a peripheral device. The connectors at the rear panel of a computer are in fact the end points of these cables. These connctors are called the ports and are used to plugin the external peripheral device to a computer. Figure 7.5 depicts the connection of computer system and I/O devices through device controllers.

In Figure 7.5, the following points are important to consider:

i. Each peripheral device is attachedtoa hardware interface known as I/O Port.
ii. A single-port device controller can controlonlyone peripheral device whereas the multi-port device controller controls multiple peripheral devices.
iii. In case of direct memory access (DMA), the communication between I/O controller and memory is established through system bus only, while the communication path passes through the CPU for non-DMA communication.

The followings are themajor benefits for connecting computer system with I/O devices using device controllers:

i. A single device controller may be used to connect multiple I/O devices with the computer system simultaneously.
ii. The device controllers allow I/O devices to upgrade/change without any update/change required in the current configuration of a computer system.
iii. The device controllers allow connecting the I/O devices ofdifferent configurations/manufacturers with the computer system. It allows computer usersto purchase I/O devices of different their configurations/manufacturers.

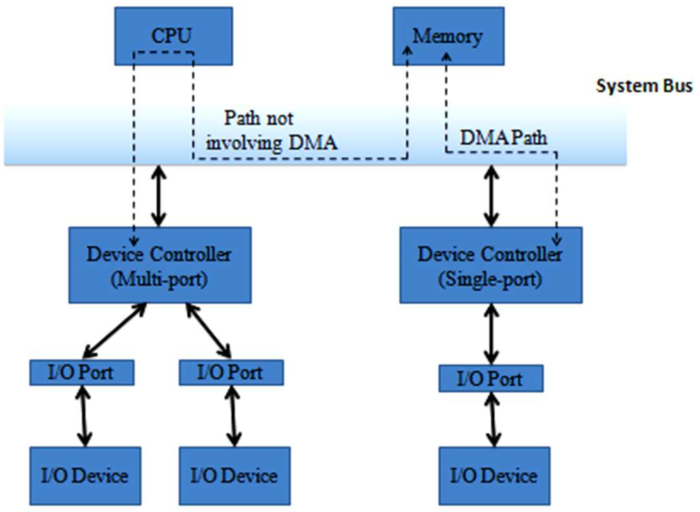


Figure 7.5: Device Controller connecting I/O Devices and Computer System

7.4.1 Device Controller and I/O Interface

Due to the different input/output needs and design complexities, there exist various peripheral deviceswith different structures. Thus,the common/standard structure of I/O interface does not exist. Figure 7.6 shows a generalblock diagram of an I/O interface. The common characteristics of the general structure of an I/O interface are as follows:

i. I/O logicis required in order to decode and execute thedata between the processor and I/O interface. Hence, control lines between the I/O interface and processors are required.
ii. Data lines between the system bus and I/O interface are necessary to facilitate the actual data transfer.
iii. The data registers may behave like a buffer between processor and I/O interface.
iv. To deal with each I/O device with different configuration,the I/O interface must have a separate logic specific.

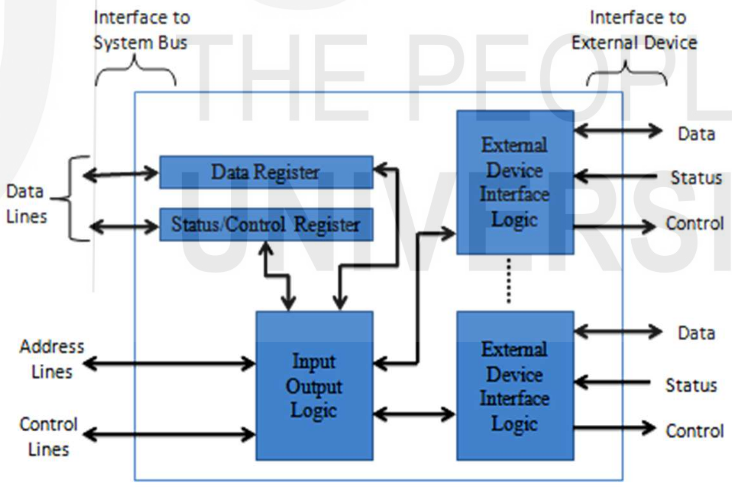


Figure 7.6: Block Diagram of an I/O Interface

7.5 DEVICE DRIVERS

The device driver is a software interface to handle communication with a specific peripheral device. The purpose of the device driver is to decodea logical request from the computer user into a series of specific actions performed by theperipheral device. For instance, a computer user may request to read a record from a disk. The device driver converts this logical request into a series of discrete commandsi.e.,check the status of the intended disk in the drive, locating the desired file, setting the position of R/E head, etc.

Device Drivers in Windows Systems

The device drivers are realized in the form of dynamic linked libraries (DLLs). The DLL consists of shareable code and thus, a single copy of the DLL code is loaded into the mainmemory. The hardware/software vendors can implement adevice driver for new

deviceswithout modifying or affecting the code of Windows operating system. Moreover, it allows multiple optional drivers tobe configured for differentperipheral devices.
For Windows based system, the device installation in the form of***Plug and Play*** is necessaryin order to add new peripheral devices. The objective of ***Plug and Play*** is to convertthe device connecting process from manual to automatic. In automatic device connecting process, the device will be attached which is followed by installation of driver software. After this, the installation process will be automatic and the settings will be changedaccording to the configuration of the host computer system.

Device Drivers for UNIX Systems
The device drivers are generally linked to the object code of the core of operating system which is known as kernel. To use a new device not included in operating system, the device driver object code has been re-linked with UNIX kernel. The advantages of this technique are simplicity and run-time efficiency while the main disadvantage is it requires regeneration of the kernel in order to attach a new device. Each entry of the /dev directory of the UNIX system is associated with a device driver which in turn is attached with the related device. Table 7.2 consists of information of some device as follows:

Table 7.2: Device Name in UNIX System

Device name	Description
/dev/console	console of a system
/dev/lp	line printer
/dev/tty01	user terminal 1
/dev/tty02	user terminal 2

Check Your Progress 1

1. What do you understand by I/O Interface?List major functions of I/O interface?

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

2. What is the need of device controller? What are the major benefits?

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

3. What is the need of a device driver? Give examples of device drivers.

……………………………………………………………………………………………….

……………………………………………………………………………………………….

………………………………………………………………………………………………..

7.6 ASYNCHRONOUS DATA TRANSFER

This section discusses the different data transfer modes.Thereexists two ways to transfer data from sender to receiver in a digital computer system. It is necessary tosynchronize the data transfer operationsusing clock pulses. When the internal registers of the two units i.e. I/O interface and CPU share a common clock, the mode of data transfer between them is known as synchronous data transfer. On the contrary, both of the units i.e., I/O interface and CPU are designed to work independent manner and each unit uses its own private clock in order to establish coordination. This mode of data transfer is knownas asynchronous data transfer.
The asynchronous data transfer faces some problems. Since there exists no fixed time slot to send or receive data. Therefore, there is no guarantee that whether the data is old or new. This problem can be solved using the following two methods:

i. Strobe Control Method
ii. Handshaking Method

7.6.1 Strobe Control method

In strobe control method a single control line is used each time for data transfer and this control signal is referred to as *Strobe*. The strobe may be activated in the following two different ways:

A. Source-initiated Strobe – Figure 7.7 depicts the block diagram and timing diagram for source-initiated strobe method. In this method, the data transfer is performed as:

i. Initially, source unit putsthe data on data bus and the strobe signal is switched ON.
ii. The destination unit reads data from the data bus.
iii. After reading data,the strobe gets OFF.

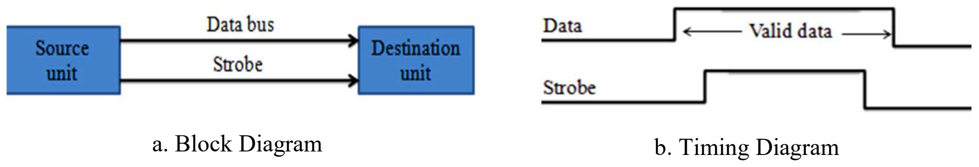


Figure 7.7: Source Initiated Strobe

B. **Destination-initiated Strobe** –Figure 7.8 depicts the block diagram and timing diagram for destination-initiated strobe method. The steps for data transfer are as follows:

i. First, the destination unit switches ON the strobe signal.
ii. After observingstrobe ON, the source unitputs data on the data bus.
iii. The destination unit reads the data from the data bus and the strobe signal gets OFF.

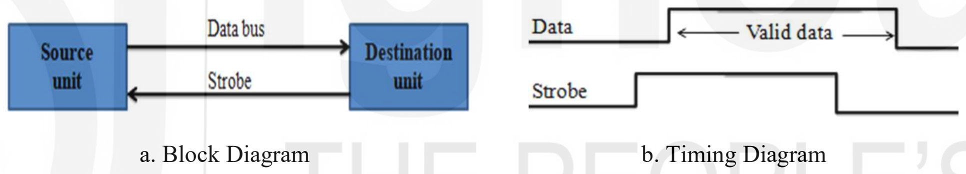


Figure 7.8: Destination Initiated Strobe

Disadvantages of Strobe Method:

The disadvantages of strobe methods are as follows:

- In source-initiatedstrobe method, it is not possible toconfirm the status of data at destination unit i.e., whether the data received is valid or not.
- In destination-initiatedstrobe method,itis not possible toconfirm the status of data at the data bus i.e., whether the data extracted from the data bus is valid or not.

7.6.2 Handshaking method

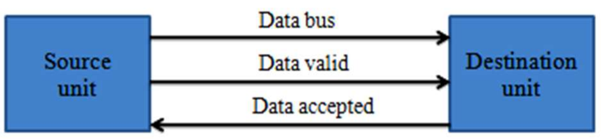
In handshaking method of asynchronous data transfer, the problemsof strobe control method can be handled. The handshaking method can be realized in the following two different ways:

A. Source-Initiated Handshaking – It consists of the following signals:
Data Valid: When activates, itspecifies that data on the data bus is validotherwise invalid.
Data Accepted: When activates, it specifies that data is acceptedotherwise not accepted.

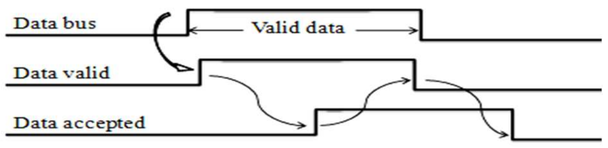
The steps of data transfer in Source Initiated Handshaking method are as follows:

i. First, source unitcopiesdata on the data bus and enables*Data Valid* signal.
ii. The destination unitreceives data from the data bus and enables*Data Accepted* signal.
iii. After this, *Data Valid*signalisdisabled whichimplies that data on data bus is invalid now.
iv. Finally, *Data Accepted* signal is disabled which indicates that the process of data transferhas been completed.

Now,it can be ensured that destination unit has read the valid data from the data bus using*Data Accepted* signal.Figure 7.9 depicts the blockdiagram and timing diagram of thesource-initiated handshaking method.



a. Block Diagram



b. Timing Diagram

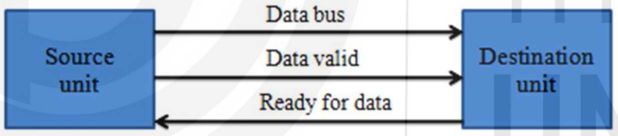
Figure 7.9: Source Initiated Handshaking

B. Destination-Initiated Handshaking – It employs the following signals:
***Request for Data*:**When activated, it requests for putting data on the data bus. ***Data Valid*:** When activated, itspecifies data is valid on the data bus otherwise invalid data.

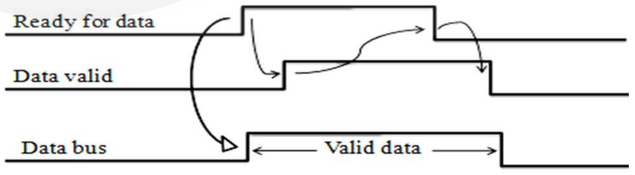
The steps of data transfer in Source Initiated Handshaking method are as follows:

i. When destination unit is ready to receive data, the appropriate signal i.e.,***Request for Data*** gets activated.
ii. In response, the sourceunit places data on the data bus as well asit also enables***Data valid*** signal.
iii. Next, the destination unit accepts data from the data bus. After receiving data, it disables***Request for Data*** signal.
iv. Lastly, the ***Data valid*** signal is disabled implies that the data bus consists of invalid data now.

Using ***data valid*** signal in **Destination Initiated Handshaking**, it is possible to ensure that source has put the data on the data bus.*F*igure 7.10 shows the block diagram and timing signal in destination-initiated handshaking. It consists of ***Request for Data***as well as ***Data valid*** signals.



a. Block Diagram



b. Timing Diagram

Figure 7.10: Destination Initiated Handshaking

7.7 INPUT/OUTPUT (I/O) TECHNIQUES

This section explains the methods to use I/O interface to support data transfer (input/output) from peripheral devices.The data transfer (I/O operation) between the computer system and peripheral devices may be performed using three techniques given as follows:

- Programmed I/O
- Interrupt-driven I/O
- Direct memory access (DMA)

Table 7.3presents important characteristics of the above three techniques.

Table 7.3: Three I/O Techniques

I/O Technique	Data Transfer Path through	Processor Intervention	Interrupt
Programmed I/O	Processor	Continuous	Not-Required
Interrupt-driven I/O	Processor	Discontinuous	Required
Direct memory access (DMA)	Direct to Memory	At the beginning and end	Required

7.7.1 Programmed I/O

In this technique, the processor is completely responsible for managing all I/O operations. The program which requests for I/O operation directly and continuously controls the I/O operation through the processor.The processor runsprogram that starts, continuously monitors and endsan I/O operation.The I/O operation may involve the following data transfer operations:

a. Data transfer from processor registers to I/O device.
b. Data transfer from I/O device to main memory through processor registers.

The major steps in programmed I/O are as follows:

i. The processor issues READ/WRITE command.
ii. The processor requests for I/O operation to I/O interface.
iii. I/O interface sets status bits.
iv. The processor continuously checks the status bits.
v. The processor waits for I/O device.
vi. When I/O device is ready, I/O operation (Read/Write data) is performed.

Figure 7.11 depicts the diagram of programmed I/O technique. When the CPU issues a command for an Input or an output to the I/O interface, it must wait until the I/O operation is completed. This process is known as polling.

Disadvantage: With the programmed I/O method, the processor continuously checks the status of the I/O device whether it is ready or not. The processor has to wait for slower I/O operations to complete. Therefore, this technique is wasteful of processor's time.

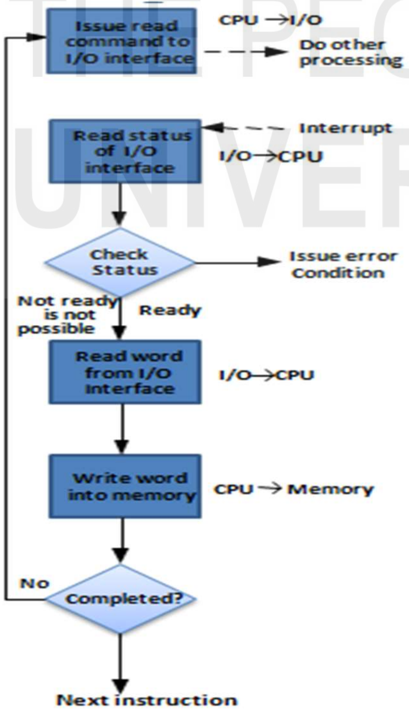


Figure 7.11: Programmed I/O

I/O Commands

Wheneverthe processor addresses an I/O interface, it sends an address and I/O command. There are four I/O commands which a processor may send to I/O interface for I/O operation.The I/O commands are given as follows:

- **Control:**Control commands are used to activate the device and also specify what operation is to perform. For example- aUSE command to make a specific device as current device for read/write operation.
- **Test:** The Test I/O command is used to check the status of a device. For instance- Whether the device is in **error condition**, **ready state**, or **notreadystate**.
- **Read:** This command is used to receiveone item of input data from the I/O device which is in communication.
- **Write:**This command is used to sendone item of output data to the respective output device.

7.7.2 Interrupt-driven I/O

For an I/O operation inprogrammed I/O, the processor continuously waits for the operation to be completed and the data transfer occurs when the device is ready. This process is known as polling which leads to slow performance of the processor. Interrupt-driven I/O can reduce the polling time efficiently.In interrupt-driven I/O, the processor issues a read/write command and it starts the execution of some other program/instructions. Whenever the desired device is ready or has completed the assigned I/O task, an interrupt signal is sent to processor for further actions. Figure 7.12 depicts the procedure of interrupt-driven I/O technique. The complete list of steps in interrupt-driven I/O is as follows:

i. The processor issues read /write command
ii. The processor executes some other program/instructions
iii. The I/O interface reads data from the desiredI/O device
iv. I/O interface interrupts the processor
v. The processor checksthe interrupt after completingeach instruction cycle
vi. The processor saves the context of the program in execution
vii. The processor requests for the desired data
viii. The I/O interface transfers data
ix. The processor resumes the previous program it had been executing before the interrupt.

Advantage: This technique reduces the overall waiting time of the processor.

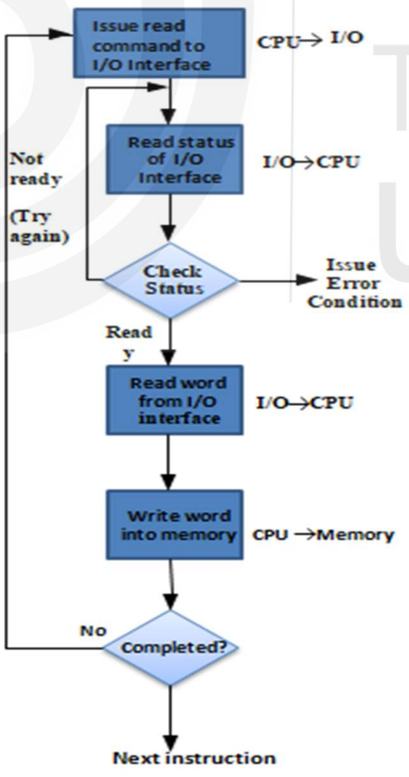


Figure 7.12: Interrupt Driven I/O

7.7.3 Interrupt Handling

An interrupt is an event which may occur to receive some service by a device or due to some error. After an interrupt occurs, it causes multiple tasks in the software as well as in the hardware.

Figure 7.13 represents the series of hardware tasks which occurred in order to I/O operation using an I/O device. The complete lists of tasks required for interrupt processing are as follows:

1. An I/O device sends an interrupt signal to processor.
2. The processor completes the execution of currently running instruction and next, it checks and takes action accordingly.
3. An acknowledgement signal is also sent by the processor to the corresponding I/O device.
4. The processor saves the context of the program that is being executed currently. It saves the following registers:
   (a) The status of the processor: program status word (PSW) register
   (b) The next instruction to be executed: program counter (PC) register.
5. In addition to thecontents of PC & PSW for the interrupted program, it is required to store the contents of the registers of processor on the stack. Figure 7.14 depicts an example of interrupt occurrence. According to theFigure 7.14, a user running program is interrupted after the execution of the instruction at location N. To handle the interrupt, the processor saves the contents of all general-purpose registers, PSW, PC along with the address of the next instruction at location N+1.
6. The processor loads the PC with the address of first location of the interrupt and then, the processor starts executing instruction from the interrupt-handler program. It means that the interrupt handler handles the interrupt.
7. When the processing of the interruptis completed. The context of the previous program is restored i.e.contents stored at stack are restored in the processor registers, which is shown in Figure 7.15.
8. Finally, the values of PC and PSW are retrieved from the stack and restored on the corresponding registers. It leadsto execution of the instruction whichwas interrupted from the previously interrupted program.

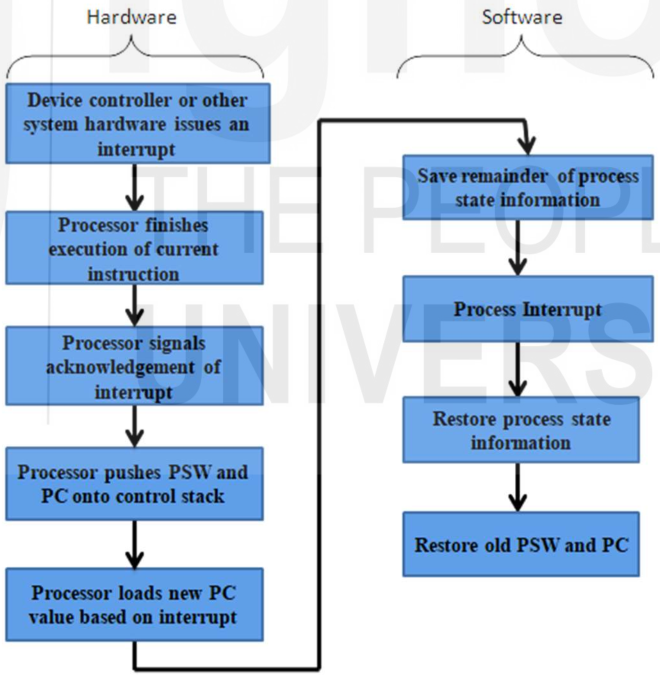

Hardware
Software
Device controller or other system hardware issues an interrupt
Processor finishes execution of current instruction
Processor signals acknowledgement of interrupt
Processor pushes PSW and PC onto control stack
Processor loads new PC value based on interrupt
Save remainder of process state information
Process Interrupt
Restore process state information
Restore old PSW and PC

Figure 7.13: The sequence of steps in Interrupt-Processing

In simplewords, the interrupt handling consists of the followings:

i. Interruptthe current program
ii. Saves the context of the interrupted program
iii. Transfer control to interrupt servicing program
iv. Execute the interrupt servicing program
v. Restore the context of interrupted program
vi. Resume interrupted program from the point where it was interrupting.

Design Issues

To realize the interrupt-driven I/O, there exist two design issues given as follows:

1) How does the processor identify the I/O device issuing the interrupt?
2) How does the processor handlemultiple interrupts occurred simultaneously?

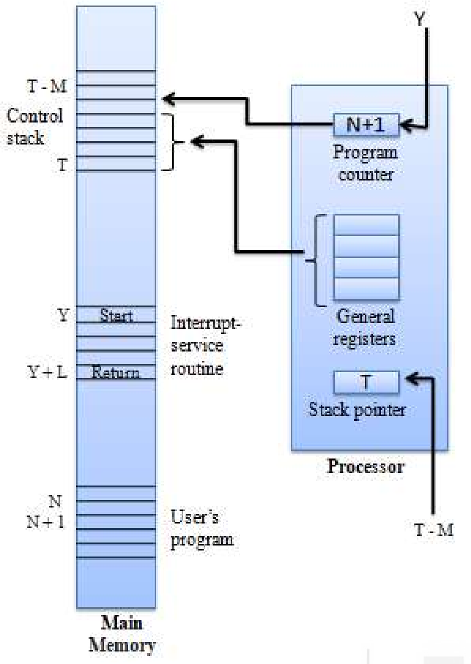


Figure 7.14: Interrupt occur for instruction at memory location N

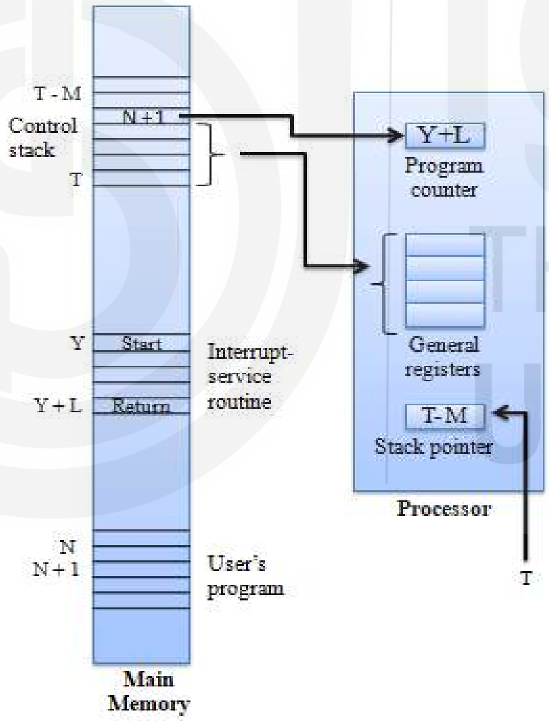


Figure 7.15: Return after handling an Interrupt

The following four general techniques can be used to solve the above design issues:

i. **Multiple Interrupt Lines:** Multiple interrupt lines can be used to handle multiple interrupts. In this technique, the priorities are assigned to various interrupts. Whenever an interrupt occurs, the highest priority interruptwill be handled first among all interrupts. However, the practical realization of this technique is difficult because computer system provides only a few lines for the interrupt.

ii. **Software Poll:** The processor jumps to an interrupt service routine after an interrupt has occurred. The processor polls at I/O interface by observing the status register in order to identify the I/O interface which caused the interrupt. The software poll technique is time consuming due to polling process.

iii. **Daisy Chain or Hardware Poll:** In this technique, the interrupt request line is shared by all I/O interfaces. Whenever an interrupt occurs, the processor sends an acknowledgement of interrupt. This acknowledgement goes to all the I/O devices through I/O interface. When it reaches to the I/O device which sends interrupt, the I/O device sends a response by specifying an address or unique identifier through the data bus. This unique identifier helps in deciding the appropriate interrupt servicing program. The hardware poll consists of an in-built priority mechanism and this priority is based on the sequence of devices on interrupt acknowledge line.

iv. **Bus Arbitration**: The bus arbitration technique allows only one I/O interface to control the bus and this I/O interface is known as bus master. It means only one interrupt request can be fulfilled at the same time. After acknowledgement of interrupt by processor, the I/O interface sends the interrupt vector to processor through data lines. The interrupt is handled according to the number in interrupt vector.

7.7.4 Direct Memory Access (DMA)

The programmed I/O as well as interrupt-driven I/O techniques require the processor's interventions for data transfer to/from the main memory. Since processor may involve in the execution of multiple programs due to multiprogramming, which restricts the processor time for testing the I/O device and servicing the I/O request by transferring I/O data over the system bus. Is it possible to store/retrieve data to/from main memory without involving the processor in the data transfer? Yes! There exists an alternative approach which is referredto as **direct memory access (DMA)**. In DMA, the main memory andI/O interface exchange data directly without the active involvement of processor. Figure 7.16 depicts the block diagram of direct memory access (DMA) technique. Now, an obvious question arises: What is the use of DMA interface? The DMA interface is mainly used to transfer a large quantity of data which isexchanged between I/O device andmainmemory.

Figure 7.16 depicts the block diagram of DMA technique. The steps of the DMA technique aregiven as follows:

i. Processor issues read/write command to DMA module for transferring the block of data
ii. The processor sends the following information to DMA interface-
    a. To perform read/write operation, the Read /Write signal is sent using control lines
    b. I/O Device address is communicated through a data bus
    c. Beginning address of memory block using data bus
    d. "The number of words to be transferred" is communicated through data bus. This information is stored in a count register.
iii. The processor executes some other program.
iv. Now, DMA controller performs the data transfer
v. When DMA controller finishes the data transfer, it sends an interrupt signal to processor

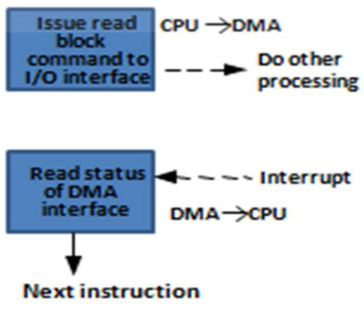

Issue read block command to I/O interface
CPU → DMA
Do other processing
Read status of DMA interface
Interrupt
DMA → CPU
Next instruction

Figure 7.16: DMA Technique

Figure 7.17 shows the block diagram of DMA module along with its different signal lines.

The DMA module handles data transfers to send the entire block of data and one word at a time is transferred directly to or from memory without going through the registers of processor.After transferring the entire block, an interrupt signal is sent to processor by DMA module.Therefore, the processor involvement is limited only at the start and end of the I/O operation.

In DMA, the processor intervention is minimized but it must use the path through system bus. Thus, DMA interface requests for system bus to transfer one data word at a time and the control of the system bus is returned to the processor after transferring on word of data. This process is known as **cycle stealing**. The CPU cycle stealing causes the delay to the operation

of processor for one memory cycle. In DMA technique, the active involvement of processor can be restricted at the beginning and at the completion of the I/O operation.
Figure 7.18 depicts the five cycles of typical instruction execution. In Figure 7.18, three points are marked where a processor can respond to DMA request, and; a point is also marked where the processor can respond tointerrupt request. The point at which the interrupt request can be acknowledged is called the interrupt cycle.

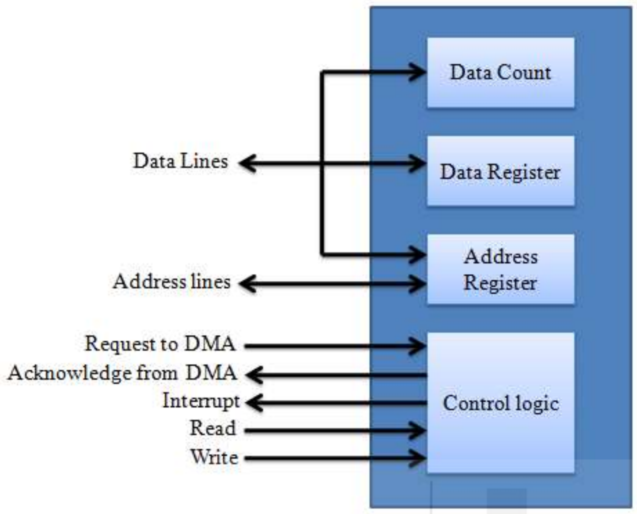


Figure 7.17: DMA's Block Diagram

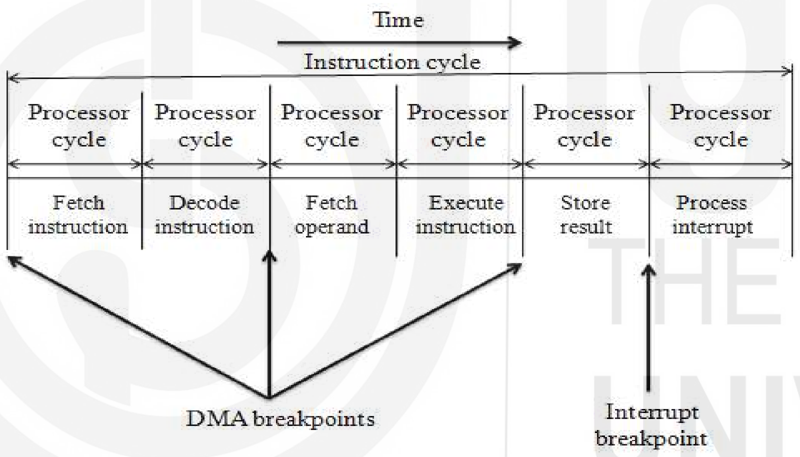
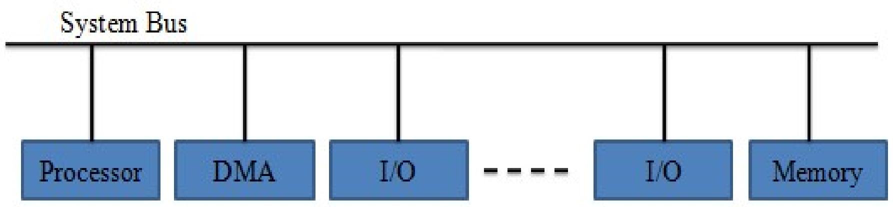
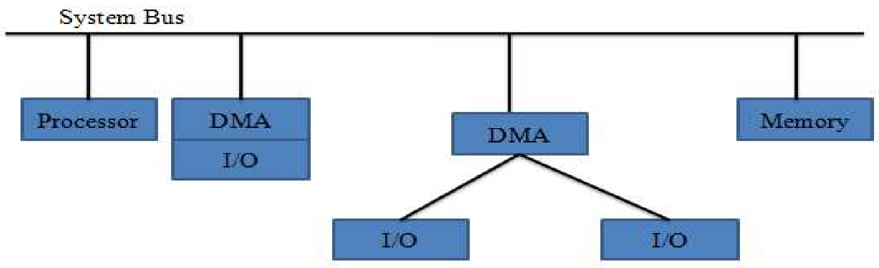


Figure 7.18: Breakpoints for DMA and Interrupt in an Instruction Cycle

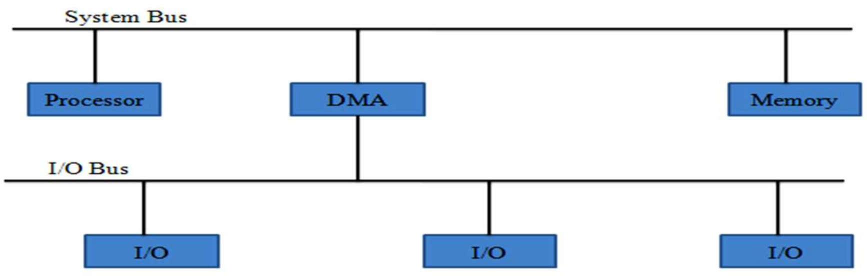
There exist different configurations to realize the DMA mechanism. Figure 7.19(a) shows one of the several configurations. In this configuration, the system bus is shared by all interfaces. The DMA works as a supportive component and may employ programmed I/O fordata transfer between memory and I/O interface using DMA interface. The advantage of this configuration and programmed I/O is that DMA does not require extra system bus cycles to transfer information from DMA to/from I/O interface as well as from main memory to/from DMA.



(a) Single Shared System Bus with detached DMA



(b) Single Shared SystemBus with integrated DMA-I/O



(c) I/O bus

Figure 7.19: Three Different DMA Configurations

Figure 7.19(b) depicts one more DMA configuration which has some advantages over the first configuration shown in Figure 7.19(a). In this configuration, adirect path is provided between the DMA interface and I/O interface and this path is different than the system bus. Furthermore, DMA logic may actas aconstituent of I/O interface and single or multiple I/O interfaces may be controlled by it. One more flexible configuration is shown in Figure 7.19(c) in which the DMA interface is connected to I/O bus. This configuration isextendable in an easy manner. The additional benefit in both configurations is that the data transfer can be performed from DMA interface to I/O interface without the involvement of the system bus.

Check Your Progress 2

1. What do you understand by Programmed I/O technique?

   ……………………………………………………………………………………………

   ……………………………………………………………………………………………

   ……………………………………………………………………………………………

2. Explain interrupt?What processing is performed on the occurrence of an interrupt?

   ……………………………………………………………………………………………

   ……………………………………………………………………………………………

   ……………………………………………………………………………………………..

3. Why is DMA needed? What are its advantage and disadvantages?

   ……………………………………………………………………………………………

   ……………………………………………………………………………………………

   ……………………………………………………………………………………………

7.8 INPUT-OUTPUT PROCESSOR (IOP)

The evolution of I/O techniques passes through many development stages and this evolution can be helpful to understand the idea of I/O processors. Various stages of evolutionaresummarizedas follows:

1. Direct Control: The processor directly controls I/O device.
2. Control using I/O controllers without interrupts: In this configuration, processor and I/O interfaces are different with each other and programmed I/O without interrupts was employed by I/O interface for I/O data transfer.

3. Control using I/O controllers with interrupts: In this configuration, the processor need not to wait for an I/O operation to be completed which leads to improved efficiency of the processor.
4. Control transferred to DMA: In this configuration, the involvement of the processor is restricted at the beginning and on completion of DMA operation. The I/O interface and DMA module control the data transfer directly without the involvement of the processor.
5. Control transferred to I/O processor: On the request of main processor, the I/O processor takes control of I/O operation and performs I/O operation with the intervention of the main processor. This approach allows controllingvarious I/O devices with minimum involvement of processor.

Through various evolution stages, it is evident that the responsibility of I/O tasks has been shifted from CPU to I/O-processor and it leads to improvedperformance of the processor.
It is also evident from the recent developments that a major shift occurs due to the introduction of I/O interface which is capable to executethe I/O instructions. Hence, the *'I/O interface'* is often called an**I/O processor** or **I/O channel**.

Input Output Processor (IOP)
An IOP is a processor that handles only the I/O processing.The block diagram of a computer system with input output processor is shown in Figure 7.20. The computer system with I/O processor consists of three major components: (a) *a memory unit* (ii) *the processor* and (iii) *the IOP*. The IOP controls and manages the input-output tasks. On the command of CPU, the IOP can fetch, decode and execute I/O instructions which are loaded to perform the I/O operations.

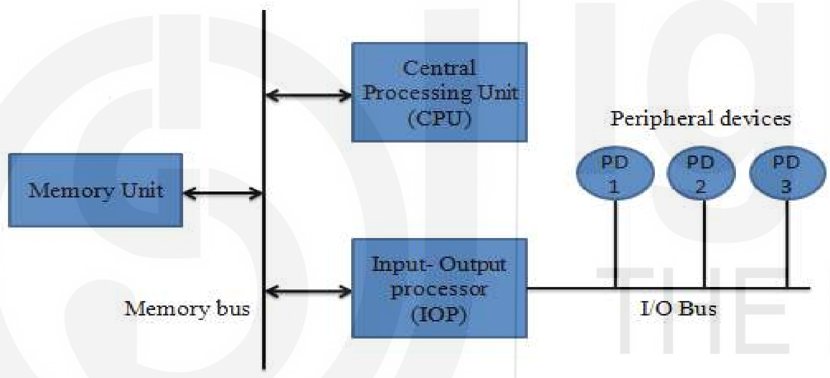
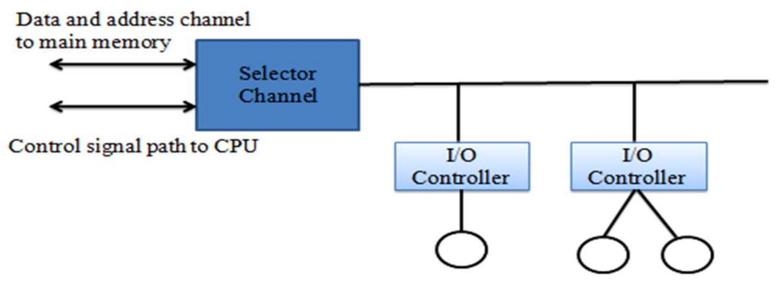


Figure 7.20: Block Diagram of a Computer System with I/O Processor

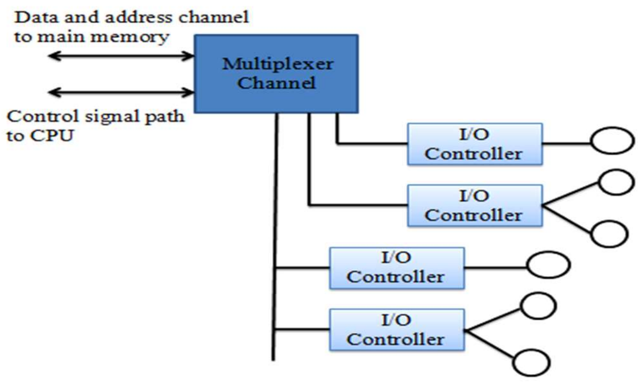
7.8.1 I/O Processor's Characteristics
In simple words, anI/O processor (or I/O channel)is theextension of the concept of DMA technique. An I/O channel holds complete control of the I/O operation and is also capable to execute I/O instructions. The CPU initiates the I/O operations andhandovers the execution of I/O commands to I/O processor. In response, the I/O processor executes the I/O instructions stored into the main memory.

The I/O channels can be divided into two categories:

i. Selector I/O channels
ii. Multiplexer I/O channels.


Data and address channel to main memory
Selector Channel
Control signal path to CPU
I/O Controller
I/O Controller

(a) Selector I/O Channel



(b) Multiplexer I/O Channel

Figure 7.21: Structures of I/O Channels

A. **Selector I/O channel:** The selector I/O channel isa dedicated channel to perform data transfer exclusively with a high speed I/O device connected to it. Each I/O controller may controla single or multiple I/O devices. Thus, I/O channelsreduce the burden ofthe CPU by controlling the I/O controllers. The selector I/O channel is shown in Figure 7.21(a).

B. **Multiplexer I/O channel:** Figure 7.21(b) depicts themultiplexer I/O channel. This I/O channel is capable to handle I/O operations with more than oneI/O device altogether. For example-if, there are three slow devices interested to send the following bytes given as:

Ist I/O device:	A1	A2	A3	A4	A5 ……
IInd I/O device:	B1	B2	B3	B4	B5……
IIIrd I/O device:	C1	C2	C3	C4	C5……

Then, the byte multiplexer I/O channel may send the bytes as follows:
C1 B1 A1 C2 B2 A2 C3 B3 A3……

For high-speed devices, block multiplexer I/O channels can be used which transfers data in form of blocks.

7.9 EXTERNAL COMMUNICATION INTERFACES

The external communication interface is used to provide an interface between the peripheral devices and I/O interface. There exist two main categories of external communication interfaces:
(a) Serial Communication Interface
(b) Parallel Communication Interface

The serial communication interfaceemploys only one line for transferring one bit of data at a time. Serial interfaces are used for *terminals*and*printers*.
Theparallel communication interfacecan transfer several bits at the same time. The parallel interfacesareusuallyutilized for I/O deviceshigh-speed.For instance, *disks* and *tapes* use parallel interfaces. The communication that takes place across the communicationinterface includes the exchange of data and control information.
In both serial and parallel communication, the I/O interface must engage in communication with the peripheral. The communication steps for the read/write operation are given as follows:

- The I/O interface sends a control signal to I/O device to requestpermission to receive or send data.
- TheI/O device accepts the Read/Write request and sends an acknowledgement.
- The data transfer operation either from I/O device to I/O interface (Read operation) or from I/O interface to I/O device (Write operation) is performed.
- The I/O device sends an acknowledgement signal after the data transfer operation.

Point-to-Point andMultipoint Configuration
There are two types of connection between the external I/O devices and I/O interface in a computer system: (i) point-to-point (ii) multipoint. In a point-to-point interface, a dedicated line is provided between the external device and I/O interface in the computer system. The

examples of use of point-to-point interfaces point are: keyboard, printer and external modems. Some common examples of point-to-point interfaces are **EAI-232** and **RS-232C**.
On the other hand, the multipoint external interfacesare used to support external multimedia devices (such as audio, video, CD-ROM) and mass storage devices (such as tape drives and disk). Some common examples of multipoint external interfaces are **Infini Band** and **FireWire**.

Check Your Progress 3

1. List majorcharacteristics of I/O processor.

   ……………………………………………………………………………………………

   ……………………………………………………………………………………………

   ……………………………………………………………………………………………

2. Differentiate selector and multiplexer I/O channels.

   ……………………………………………………………………………………………

   ……………………………………………………………………………………………

   ……………………………………………………………………………………………

3. Compare serial and parallel external interfaces.

   ……………………………………………………………………………………………

   ……………………………………………………………………………………………

   ……………………………………………………………………………………………

7.10 SUMMARY

This unit explains exclusively the I/Oorganisation of a computer system. This unit presents the description of I/O devices, the concept of I/O interface, the description and structure of device controllers, and asynchronous data transfer modes. It also discusses three I/O techniques i.e.,programmed I/O, interrupt-driven I/O, and direct memory access (DMA). The threeI/O techniques involve processor at different levels and help processor to behave differently. The interrupt processing is also discussed in detail.The evolution of I/O processor is also discussed along with the brief explanation of the input/output processor as well as the external communication interfaces. The I/O processors are the most recent I/O interfaces that are capable to execute the I/O instructions without the involvement of the processor.

7.11 SOLUTIONS /ANSWERS

Check Your Progress 1

1. An I/O interface acts as a bridge between the I/O devices and processor. It provides a communication interface to connect the computer with the external environment. The I/O interfaces are used to send/receive data from the external environment using external device or peripheral. The I/O interface offers the following major functions:
   - Synchronization of control and Timing signals
   - Establish communication between peripheral devices and processor
   - Data buffering to due to the speed mismatch between memory and processor
   - A mechanism for error detection
2. Device controllers work as a connecting mediumto connect the peripheral devices to a computer system rather than connecting them directly to the system bus. On device controller, the I/O device is connected at one end while it is connected with the system bus on another end. Thus, a device controller works as an interface between the system bus and the I/O device. The major benefits for connecting I/O devices to a computer system using device controllers are as follows:

- A single device controller can be used to connect multiple I/O devices with the computer system simultaneously.
- The device controllers allow I/O devices to upgrade/change without any update/change required in the current configuration of computer system.
- The device controllers allow connecting the I/O devices with different configurations/manufacturers with the computer system. This feature offers more flexibility to the computer users in order to buy I/O devices of different configurations/manufacturers

3. A device driver is a software module which manages the communication with a specific I/O device. It provides a software interface to hardware devices.Some examples of different device drivers are printer driver, sound card driver, graphics driver, network card driver, USB driver etc.

Check Your Progress 2

1. Programmed I/O technique is used to perform the I/O operation and it does not need an interrupt for I/O operation.The main purpose of the programmed I/O is to perform data transfer between processor and external environment. This technique is very inefficient, especially when multiprogramming environment is used, as Programmed I/O requires continuous involvement of the processor for the I/O to complete. This wasted time could have been used for exection of other programs.
2. An interrupt is an event which may occur to receive some service by a device or due to some error. After an interrupt occurs, it causes multiple tasks in the software as well in the hardware. The abstract steps for the interrupt handling are as follows:
    - Interrupt the current program
    - Saves the context of the interrupted program
    - Transfer control to interrupt servicing program
    - Execute the interrupt servicing program
    - Restore the context of the interrupted program
    - Resume interrupted program from the point of interruption.
3. DMA is needed to enable a device to transfer data without exposing the CPU which results in much less CPU overhead.The advantages of DMA are:
    - DMA allows a peripheral device to read and write to/from memory without passing through the CPU.
    - DMA allows for faster processing as the processor can be working on something else while the peripheral can perform memory related work.

The disadvantages of DMA are:

- requires a DMA controller to carry out the operation, which increases the cost of the system
- problem of cache coherence

Check Your Progress 3

1. Some of the characteristics of I/O channels are:
    - An I/O processor (I/O channel)is the extension of the DMA concept.
    - An I/O channel holds complete control of the I/O operation and is also capable to execute I/O instructions.
    - The CPU initiates the I/O operations and handovers the execution I/O commands to I/O channel. In response, the I/O channel executes the I/O instructions stored into the main memory.
2. The difference between Selector Channel and Multiplexer Channel is that the Selector channel communicates the data with only one dedicated high-speed device at one time while the multiplexer channel is capable to handle I/O with multiple devices at the same time.
3. The difference between serial and parallel interfaces is given below:
    - The serial communication interface employs only one line to transfer one bit of data at a time. The parallel communication interface can transfer several bits at the same timeis usually utilized for I/O devices high-speed.
    - Serial interfaces are used for terminals and printers while high-speed peripherals such as disks and tapes use parallel interfaces.