APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

STUDY MATERIALS

Module - III

I/O organization: accessing of I/O devices –interrupts –direct memory access –buses –interface circuits – standard I/O interfaces (PCI, SCSI, USB)

One of the basic features of a computer is the ability to exchange the data with other devices.
Now a day, in almost all environments computer is an integral part.
So that I/O organization has an important role in computer organization.

1. Accessing of I/O devices

A simple arrangement to connect I/O devices to a computer is to use a single bus structure. It consists of three sets of lines to carry • Address • Data • Control Signals.

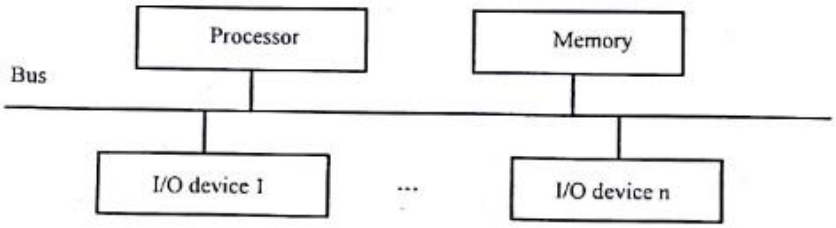


Fig 3.1: A single-bus structure

When the processor places a particular address on address lines, the devices that recognize this address responds to the command issued on the control lines.
The processor request either a read or write operation and the requested data are transferred over the data lines.

Based on the address space used by the I/O devices, there are two schemes for I/O.

Memory Mapped I/O:

- When I/O devices & memory share the same address space, the arrangement is called memory mapped I/O.
- Here same instructions are using for controlling memory and I/O.
- Example

Move DATAIN, R_o	→ Reads the data from DATAIN then into processor register R_o
Move R_o, DATAOUT	→ Send the contents of register Ro to location DATAOUT.
DATAIN	→ Input buffer associated with keyboard
DATAOUT	→ Output data buffer of a display unit / printer.

I/O Mapped I/O

- There are separate address spaces for I/O devices.
- CPU instructions are designed specifically to perform I/O.
- This doesn't means that I/O address lines are physically separate from memory address lines.
- A special signal on the bus indicates that the requested read/ write transfer is an I/0 operation.
- The different I/O devices connected to the bus will examine the address lines to see that whether they have to respond to this or not.

I/O Interface

This includes the hardware required to connect an I/O Device to the bus.

Interface circuit consists of address decoder, data registers, status registers and control circuitry.

Address decoder enables the device to recognize its address when this address appears on address lines.

The data register holds the data being transferred to or from the processor.

Status register contains the information relevant to the operation of the I/O device.

For an input device, SIN status flag in used.

For example, when SIN = 1 a character is entered at the keyboard.

When SIN=0, the character is read by the processor.

For an output device, SOUT status flag is used in a similar fashion. Status and data registers are connected to the data bus.

Program Controlled I/O

Here the processor repeatedly checks a status flag to achieve the required synchronization between Processor & I/O device that is, the processor polls the device.

Interrupt

Synchronization is achieved by having I/O device send special signal over the bus whenever it is ready for data transfer operation

DMA

Here the device interface transfers the data directly to or from the memory, without continuous involvement of the processor. It is a technique used for high speed I/O device.

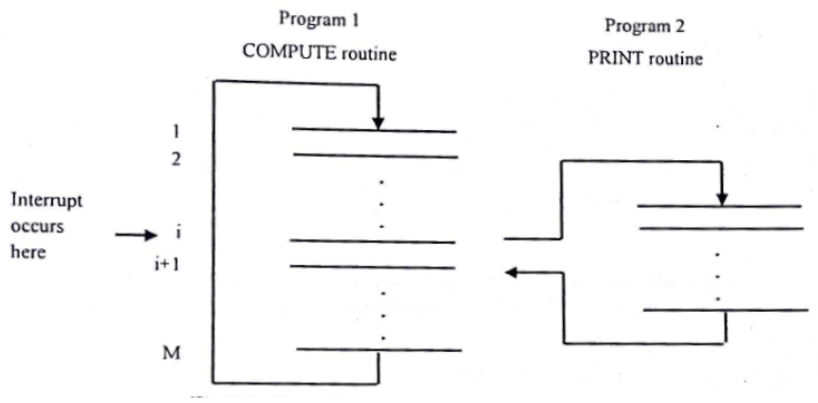
2. Interrupt

The processor doesn't have to continuously check the device status.

We can arrange the I/O device to alert the processor when it becomes ready. It can do so by sending a hardware signal called an interrupt to the processor.

One of the bus control lines named as interrupt request line is dedicated for this purpose.

The routine executed in response to an interrupt request is called Interrupt Service Routine (ISR).



- The processor first completes the execution of instruction i.
- Then it loads the PC (Program Counter) with the address of the first instruction of the ISR (Interrupt Service Routine).
- After the execution of ISR, the processor has to come back to instruction i + 1.
- Therefore, when an interrupt occurs the current contents of PC which is pointing to i +1 is put in a temporary storage of known location. A return from interrupt instruction at the end of ISR reloads the PC from that temporary storage location, causing the execution to resume at instruction i+1.
- When the processor is handling the interrupts, it must inform the device that its request has been recognized so that it removes its interrupt requests signal. This may be accomplished by a special control signal called the interrupt acknowledge signal.
- The task of saving and restoring the information can be done automatically by the processor. The processor saves only the contents of program counter & status register.
- The delay between the time an interrupt request is received and the start of the execution of the ISR is called the Interrupt Latency.
- In real-time processing (where processing of certain routines must be accurately timed relative to external events) interrupt concept is used.

❖ **Enabling and Disabling Interrupts**

• **First Method**

- Processor hardware ignores the TRQ (Interrupt request) line until the execution of first instruction of ISR has been completed.
- DI (Disable interrupts).instruction will be used as first instruction in ISR (Interrupt Service Routine)
- Last instruction in ISR is EI, before RET instruction.

• **Second Method**

- Processor automatically disables interrupts before starting the execution of ISR.
- One bit in Processor Status Register called Interrupt Enable is used to indicate whether interrupts are enabled or not.
- When bit is 1, saves the contents of processor status register on the stack, then clears the bit to 0 so that further interrupts will be disabled. When RET from interrupt instruction is executed, the contents of PS are restored from stack, so setting the bit again to 1.

•**Third Method**

- Processor has a special IRQ line for which the interrupt-handling circuit responds only to the leading edge of the signal. Such a line is said to be edge-triggered.
- In this case, the processor will receive only one request, regardless of how long the line is activated.
-There is no chance for multiple interruptions and hence no need for explicit disable interrupts.

❖ **Events involved in handling an IRQ from a single device**

1. Device raises an IRQ
2. The processor interrupts the program currently being executed
3. Interrupts are disabled by changing the control bits in the Processor Status Register
4. Device is informed that its request has been recognized, and in response it deactivates the IRQ signal.
5. The action requested by the interrupt is performed by the ISR
6. Interrupts are enabled and execution of the interrupted program is resumed.

❖ **Handling Multiple Devices**

- When more than one device is connected to a single interrupt Request line, additional information is needed to identify the device that activates the IRQ line.

• **Polling**

-With the help of one bit in status register, it is able to identify the device who initiates the interrupt request.

- The simplest way to identify the interrupting device is poll all the I/O devices connected to the bus.

Advantage

It is easy to implement.

Disadvantage

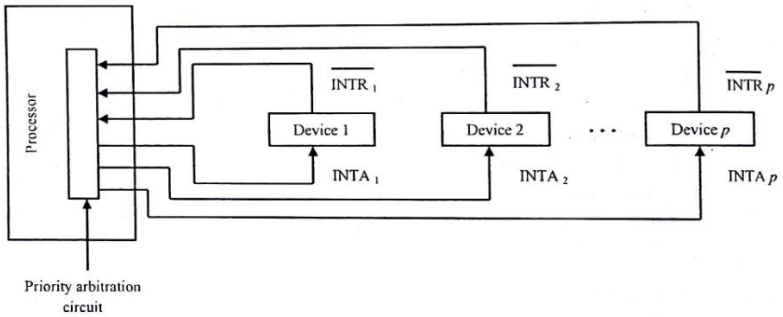
So much time is wasted by interrogating the IRQ bit of all devices that may not be requesting any service.

• Vectored Interrupts

- To reduce the time involved in the polling process, a device requesting an interrupt can identify itself directly by sending a special code to the processor over the bus.
- It enables the processor to identify the device that generated the interrupt.
- Code supplied by device may represent the starting address of ISR for that device
- Processor reads this address called **interrupt vector** & loads in to PC.
- Processor can immediately start the execution of the corresponding Interrupt Service Routine (ISR).

• Nested Interrupts

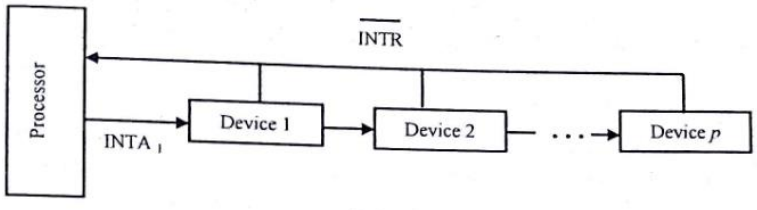
- I/O devices should be organized in a priority structure.
- That is an interrupt request from a higher priority device should be accepted while the processor is servicing another request from a lower priority device.
- A multiple priority scheme can be implemented easily by using separate interrupt request and interrupt acknowledge lines for each device.



- Each of the interrupt request line is assigned a different priority level.
- Interrupt requests received over these lines are sent to a priority arbitration circuit in the processor.
- A request is accepted only if it has higher priority level than that currently assigned to the processor.

•Simultaneous Requests

- If simultaneous request arrives from 2 or more devices, then processor must have some means of deciding which request to service first.
- A widely used scheme is to connect the devices to form a daisy chain, as in below figure.



Daisy chain

- Here all interrupt devices are serially connected.
- Higher priority device places first closed to the processor.
- Second device along the chain have second priority, third device along the chain have third priority and so on.
- Interrupt request line (INTR) is common to all devices.
- CPU responds to the interrupt via Interrupt Acknowledge line (INTA)
- When several devices raise INTR, processor responds by setting INTA line to 1.
- It is received by device 1. It passes to device 2 only if device 1 does not require any service.

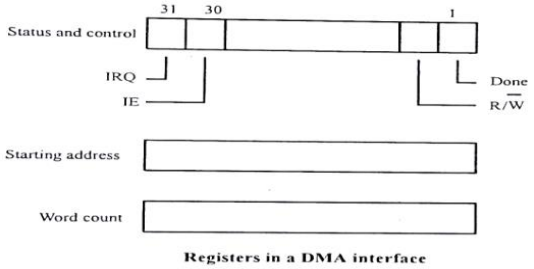
- **Advantage of priority** scheme is, it allows the processor to accept interrupt requests from some devices but not from others, depending upon priorities.

3. Direct Memory Access

- An important aspect governing the computer system performance is the transfer of data between memory and I/O devices.
- Polling and Interrupt driven I/O concentrates on data transfer between the processor and I/O devices.
- To transfer large blocks of data at high speed, an alternative approach is used.
- A special control unit may be provided to allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor. This is known as Direct Memory Access (DMA).
- DMA transfers are performed by a control circuit known as DMA Controller that is part of the I/O interface.
- DMA controller performs the functions that would normally be carried out by the processor when accessing the main memory.
- Although DMA Controller can transfer data without intervention by the processor, its operation must be under the control of a program executed by the processor.
- To initiate the transfer of a block of data, the processor sends the starting address, the number of words in the block, and direction of the transfer. Once information is received, the DMA Controller proceeds to perform the requested operation. When the entire block has been transferred, the controller informs the processor by raising an interrupt signal.

Registers in a DMA Interface

- Two registers are used for storing the starting address and the word count.
- Third register contains the status and control flags.



Registers in a DMA interface

- R/W determines the direction of transfer.
- Done flag will be set to one when the controller has completed the whole data transfer and is ready to receive another command.
- IRQ bit is set to when it has requested an interrupt.
- Interrupt Enable (IE) flag will be set to 1 means that it causes the controller to raise an interrupt after it has completed transferring a block of data.

Cycle stealing

- Requests by DMA devices for using the bus (for memory access) are always given higher priority than processor requests.
- Among different DMA devices, top priority is given to high-speed peripherals (disks, high-speed network interface, and graphics display device).
- It is often stated that DMA steals memory cycles from the processor (cycle stealing).
- If DMA controller is given exclusive access to the main memory to transfer a block of data without interruption, this is called block or burst mode.

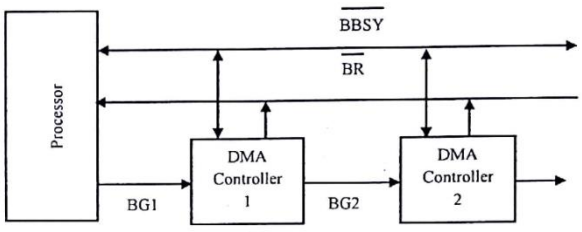
Bus Arbitration

- Bus Arbitration is needed to resolve conflicts with more than one device (2 DMA Controllers or DMA and processor, etc...) try to use the bus to access main memory.
- The device that is allowed to initiate bus transfers on the bus at any given time is called bus master.
- The process by which the selection of next device to become bus master and bus mastership is transferred to it is known as bus arbitration.

There are two approaches to bus arbitration.

1. Centralized Arbitration

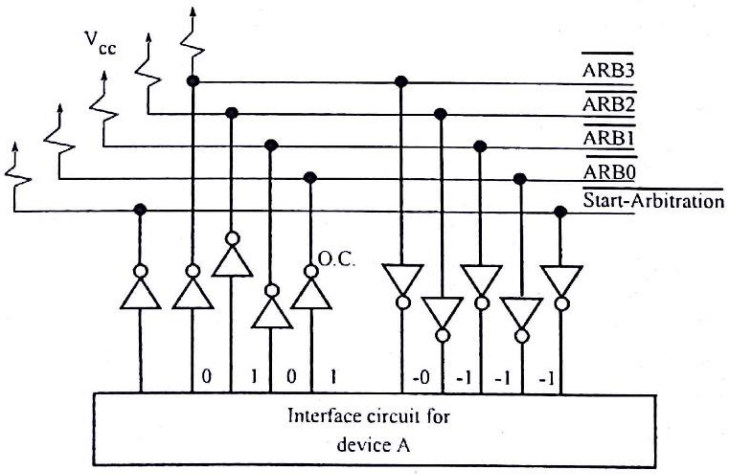
Here, a single arbiter performs the arbitration. Bus arbiter may be a processor or a separate unit connected to the bus.



A Simple arrangement for bus arbitration using a daisy chain

2. Distributed Arbitration

- Here all devices participate in the selection of the next bus master.
- No central arbiter is used.



A distributed arbitration scheme

- Each device on bus is assigned a 4-bit identification number. When one or more devices request the bus, they assert the Start-Arbitration signal and place their 4-bit ID number on ARB [3...0].
- The request that has the highest ID number ends up having the highest priority.
- Advantage is that it offers higher reliability (operation of the bus is not dependent on any one device).
- SCSI bus is an example of distributed (decentralized) arbitration.

4. Buses

- A bus protocol is the set of rules that govern the behaviour of various devices connected to the bus as to when to place information on the bus, assert control signals, and so on.
- Bus lines may be grouped into three types: **Address, data and control lines.**
- The control signals specify whether a read or write operation is to be performed.
- The bus control signal also carries timing information.
- They specify the times at which the processor and I/O devices may place on the bus or receive data from the bus.
- Many schemes have been there for the timing of data transfers over the bus. They can broadly be classified into two types:

1. Synchronous Bus

In a synchronous bus, all devices derive timing information from a common clock line. Equal spaced pulses on this line define equal time intervals. In the simplest form of a synchronous bus, each of these intervals constitutes a bus cycle during which one data transfer can take place.

2. Asynchronous Bus

An alternative scheme for controlling data transfers on the bus is based on the use of a handshake between the master and slave. The common clock is replaced by two timing control lines, Master-ready and Slave-Ready. The first is asserted by the master to indicate that it is ready for a transaction, and the second is a response from the slave.

5. Interface Circuits

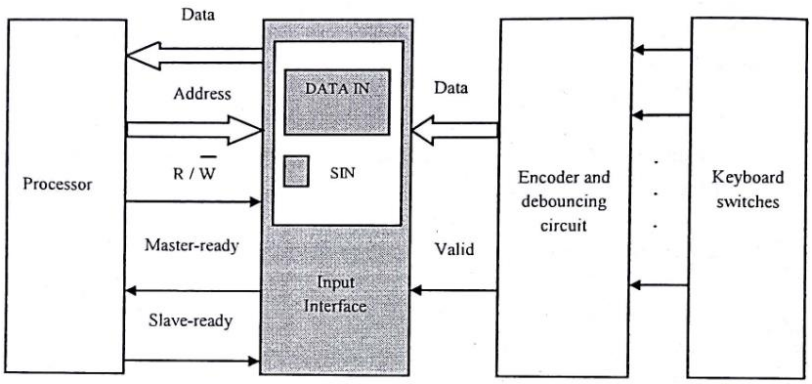
- An I/O interface consists of a circuitry required to connect an I/O device to a computer bus.
- One side of the interface consists of bus signals and the other side consists of the data path with its associated control (to transfer the data between the interface and 1/0 Device).This side is called a port.
- Ports can be classified into two types: serial and parallel port.

Parallel Port

- A parallel port transfer's data in the form of bits (may be 8 or 16).
- In Parallel Port, the connections between the device and the computer uses a multiple pin connector and a cable with as many wires, typically arranged in a flat configuration.
- The circuits at either end are relatively simple, as there is no need to convert between parallel and serial formats. This arrangement is suitable for devices that are physically close to the computer.

Interface design-Example

The following figure shows the hardware components needed for connecting a keyboard to a processor.

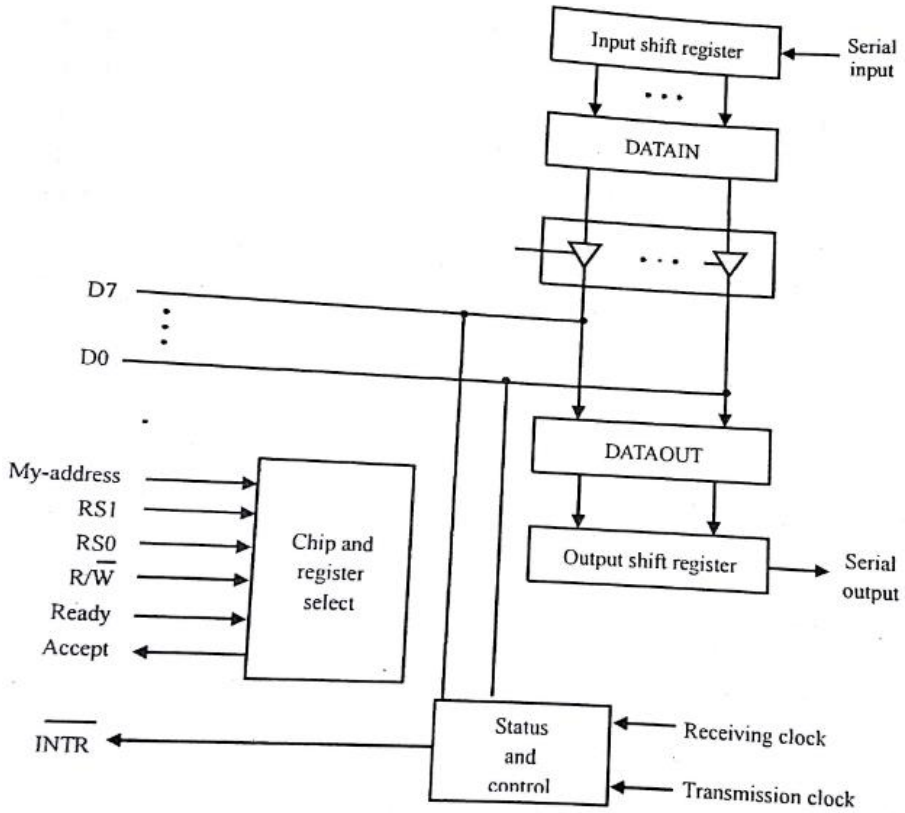


Keyboard to Processor connection

- The output of the encoder consists of the bits that represent the encoded character and one signal called valid, which indicates the key is pressed.
- The information is sent to the interface circuits, which contains a data register, DATAIN and a status flag SIN.
- When a key is pressed, the valid signal changes from 0 to l, causing the ASCII code to be loaded into DATAIN and SIN set to 1.
- The status flag SIN set to 0 when the processor reads the contents of the DATAIN register.
- The interface circuit is connected to the asynchronous bus on which transfers are controlled using the Handshake signals Master ready and Slave-ready.

Serial Port

- A serial port used to connect the processor to I/O device that requires transmission one bit at a time.
- It is capable of communicating in a bit serial fashion on the device side and in a bit parallel fashion on the bus side.
- The transformation between serial and parallel format is achieved with shift registers that have parallel access capacity.



A Serial Interface

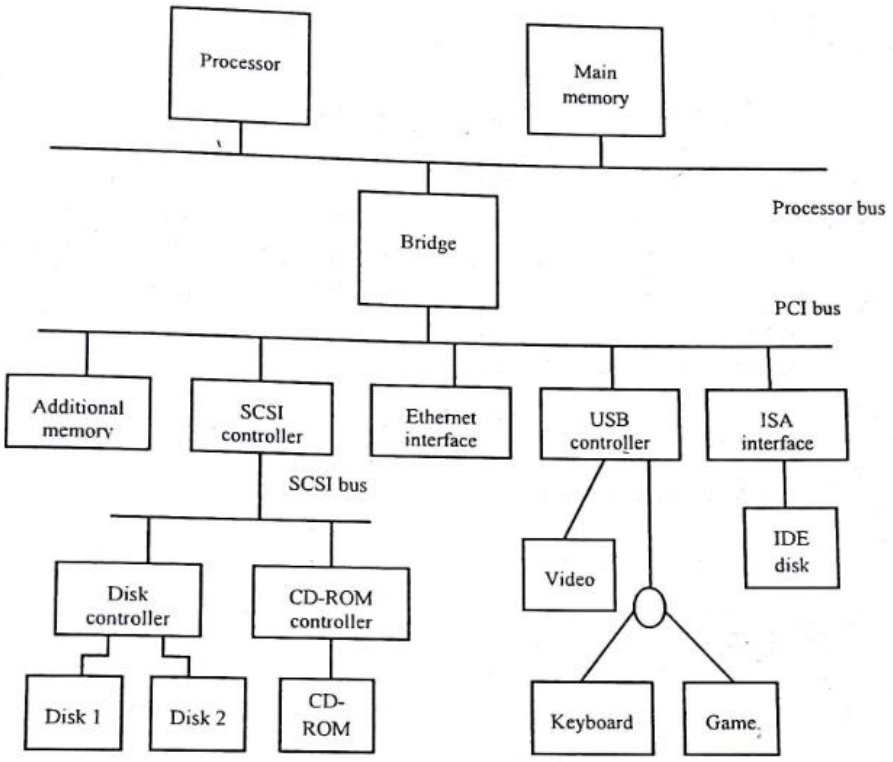
- It includes DATAIN and DATAOUT registers. The input shift register accepts bit serial input from the I/O device.
- When all 8 bits of data have been received, the contents of this shift register are loaded in parallel into the DATAIN register.
- Similarly, output data in the DATAOUT register are loaded into the output shift register, from which the bits are shifted out and sent to the I/O device.

6. Standard I/O Interfaces (PCI, SCSI, USB)

- A standard I/O Interface is required to fit the I/O device with an Interface circuit.
- The processor bus is the bus defined by the signals on the processor chip itself .
- The devices that require a very high speed connection to the processor such as the main memory may be connected directly to this bus.
- The motherboard usually provides another bus that can support more devices.
- The two buses are interconnected by a circuit call as a **bridge**.
- The bridge connects two buses, which translates the signals and protocols of one bus into another. The bridge circuit introduces a small delay in data transfer between processor and the devices.

Widely used bus standards are:

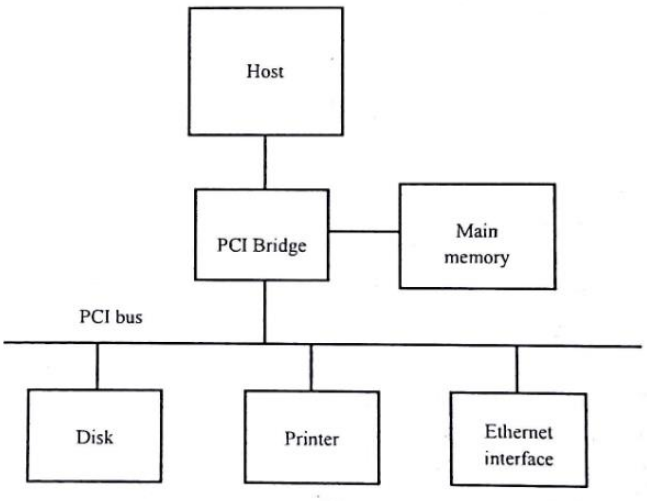
1. PCI (Peripheral Component Inter Connect)
2. SCSI (Small Computer System Interface)
3. USB (Universal Serial Bus)



An example of a computer system using different interface standards

1. Peripheral Component Interconnect Bus (PCI)

- PCI is developed as a low cost bus that is truly processor independent.
- It supports high speed disk, graphics and video devices.
- PCI has plug and play capability for connecting I/O devices. To connect new devices, the user simply connects the device interface board to the bus.



Use of a PCI bus in a computer system

Data Transfer

- A read / write operation involving a single word is treated as a burst of length one.
- PCI has three address spaces. They are:
  - Memory address space
  - I/O address space
  - Configuration address space

- I/O address space is intended for use with processors, which have separate I/O address space.
- Configuration space is intended to give the PC1 its plug and play capability.
- PCI Bridge provides a separate physical connection to main memory.
- The master maintains the address information on the bus until data transfer is completed.
- At any time, only one device acts as bus master.
- A master is called 'initiator' in PCI which is either processor or DMA controller.
- The addressed device that responds to read and write commands is called a target.
- A complete transfer operation on the bus, involving an address and a burst of data is called a transaction. Individual word transfers within the transaction are called phases.

2. SCSI (Small Computer System Interface)

- It refers to a standard bus defined by the American National Standards Institute (ANSI).
- The SCSI bus may be used to connect a variety of devices to a computer.
- It is particularly well-suited for use with disk drives.
- It is often found in installations such as institutional databases or email systems where many disks drives are used.
- Many versions including SCSI-2, SCSI-3 (also known as Ultra SCSI or fast 20 SCSI), and different Ultra versions are there.
- A SCSI may be narrow or wide. In narrow it may have 8 data lines, so that one byte of data can be transfer at a time. In a wide SCSI, it has 16 data lines and transfer 16 bit of data at a time.
- The maximum capacity of the bus is 8 devices for a narrow bus and 16 devices for a wide bus.

Data Transfer

- A SCSI bus may be connected directly to the processor bus, or more likely to another standard I/O bus such as PCI, through a SCSI controller.
- Data and commands are transferred in the form of multi-byte messages called packets.
- To send commands or data to a device, the processor assembles the information in the memory then instructs the SCSI controller to transfer it to the device.
- Similarly, when data arc read from a device, the controller transfers the data to the memory and then informs the processor by raising an interrupt.

There are two types of SCSI controllers.

1. Initiator:

- It has the ability to select a particular target and to send commands specifying the operations to be performed.
- Controller on the processor side must be an initiator type.

2. Target:

- Disk controller operates as a target. It carries out the commands received from the initiator.
- Data transfer on SCSI bus is always controlled by the target controller.

- To send a command to the target, an initiator requests control of the bus and after winning arbitration. selects the controller it wants to communicate with and hands over the control of the bus over to it.
- Then the controller starts a data transfer operation to receive a command from the initiator.

3. Universal Serial Bus (USB)

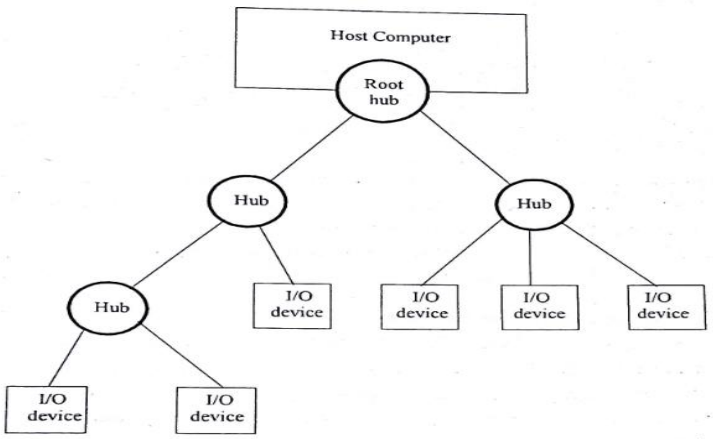
- The Universal Serial Bus (USB) is the most widely used interconnection standard.
- A large variety of devices are available with a USB connector, including mice, memory keys, disk drives, printers, cameras, and many more.
- The commercial success of the USB is due to its simplicity and low cost.
- The original USB specification supports two speeds of operation, called low-speed (1.5 Megabits/s) and full-speed (12 Megabits/s).
- Later, USB 2, called High-Speed USB, was introduced. It enables data transfers at speeds up to 480 Megabits/sec. USB 3 (called Superspeed) was developed, it supports data transfer rates up to 5 Gigabits/s.

The USB has been designed to meet several key objectives:

- Provide a simple, low-cost, and easy to use interconnection system.
- Accommodate a wide range of I/O devices and bit rates, including Internet connections, and audio and video applications.
- Enhance user convenience through a "plug-and-play" mode of operation.

USB Architecture

- The USB uses point-to-point connections and a serial transmission format.
- When multiple devices are connected, they are arranged in a tree structure as shown in below figure.



Universal Serial Bus tree structure

- Each node of the tree has a device called a hub, which acts as an intermediate transfer point between the host computer and the I/O devices.
- At the root of the tree, a root hub connects the entire tree to the host computer.
- The leaves of the tree are the I/O devices: a mouse, a keyboard, a printer, an Internet connection, a camera, or a speaker.

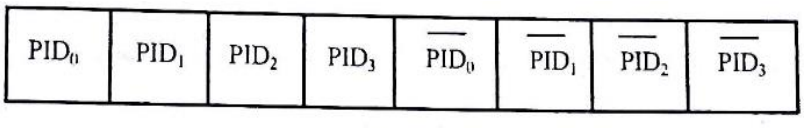
- The tree structure makes it possible to connect many devices using simple point-to-point serial links. If 110 devices are allowed to send messages at any time, two messages may reach the hub at the same time and interfere with each other. For this reason, the USB operates strictly on the basis of polling.
- A device may send a message only in response to a poll message from the host processor. Hence, no two devices can send messages at the same time. This restriction allows hubs to be simple, low-cost devices.
- The above specified mode of operation is same for all devices operating at either high speed or low speed.

Addressing

- Each device on the USB, whether it is a hub or an I/O device, is assigned a 7-bit address.
- This address is local to the USB tree and is not related in any way to the processor's address space.
- The root hub of the USB, which is attached to the processor, appears as a single device.
- The host software communicates with individual devices by sending information to the root hub, which it forwards to the appropriate device in the USB tree.
- When a device is firstly connected to a hub, or when it is powered on, it has the address 0. Periodically, the host polls each hub to collect status information and learn about new devices that may have been added or disconnected.
- When the host is informed that a new device has been connected, it reads the information in a special memory in the device's USB interface to learn about the device's capabilities.
- It then assigns the device a unique USB address and writes that g address in one of the device's interface registers. It is this initial connection procedure that gives the USB its plug-and-play capability.

USB Protocols

- All information transferred over the USB is organized into packets, where a packet consists of one or more bytes of information.
- The information transferred on the USB can be divided into two broad categories: control and data.
- A packet consists of one or more fields containing different kinds of information.
- The first field of any packet is packet identifier PID, which identifies the type of packet. There are four bits of information in this field, but they are transmitted twice. The first time they are sent with their true values, and the second time with each bit complemented. This enables the receiving device to verify that the PID byte has been received correctly. Format is given in the following figure.



Packet identifier field

try it now

A KTU STUDENTS PLATFORM

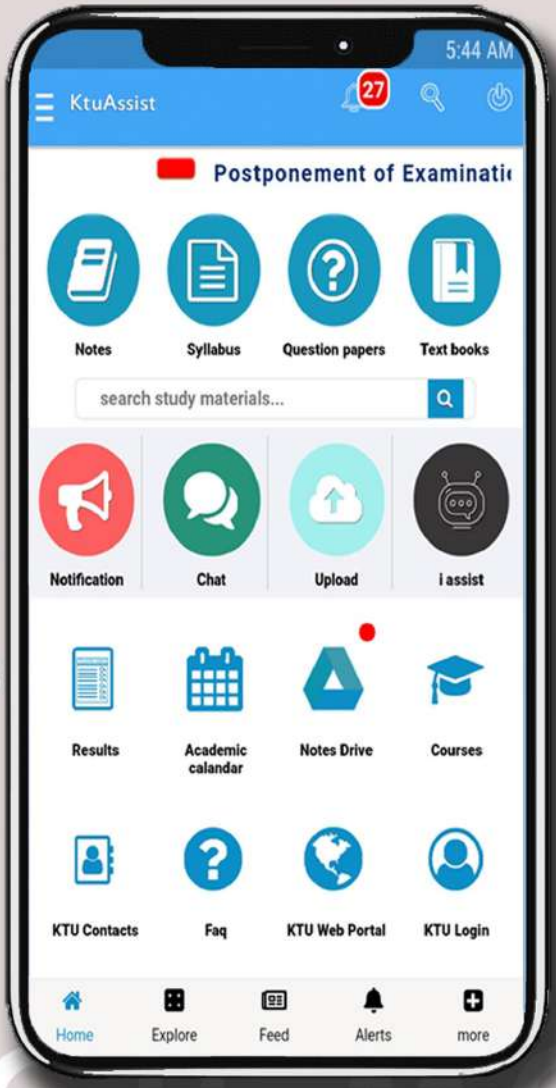
SYLLABUS

NOTES

TEXT BOOKS

QUESTION PAPERS

KTU NOTIFICATION



DOWNLOAD IT FROM GOOGLE PLAY

CHAT

FAQ

LOGIN

CALENDAR

MUCH MORE

DOWNLOAD APP

ktuassist.in

instagram.com/ktu_assist

facebook.com/ktuassist