# SUBJECT NAME: Computer Organization & Architecture

# SUBJECT CODE: 203105253

# UNIT 4

Prepared By

Trilok Suthar

**PERIPHERAL DEVICES AND THEIR CHARACTERISTICS:**:
Input-output subsystems, I/O device interface, I/O transfers-program controlled, interrupt driven and DMA, privileged and non-privileged instructions, software interrupts and exceptions. Programs and processes-role of interrupts in process state transitions, I/O device interfaces -SCSI, USB

# Input-output subsystems

- The I/O subsystem of a computer provides an efficient mode of communication between the central system and the outside environment. It handles all the input-output operations of the computer system

# Peripheral Devices

- Input or output devices that are connected to computer are called **peripheral devices**. These devices are designed to read information into or out of the memory unit upon command from the CPU and are considered to be the part of computer system. These devices are also called **peripherals**.
- For example: *Keyboards*, *display units* and *printers* are common peripheral devices.
- There are three types of peripherals:
- **Input peripherals** : Allows user input, from the outside world to the computer. Example: Keyboard, Mouse etc.
- **Output peripherals**: Allows information output, from the computer to the outside world. Example: Printer, Monitor etc
- **Input-Output peripherals**: Allows both input(from outside world to computer) as well as, output(from computer to the outside world). Example: Touch screen etc.

# I/O Interface

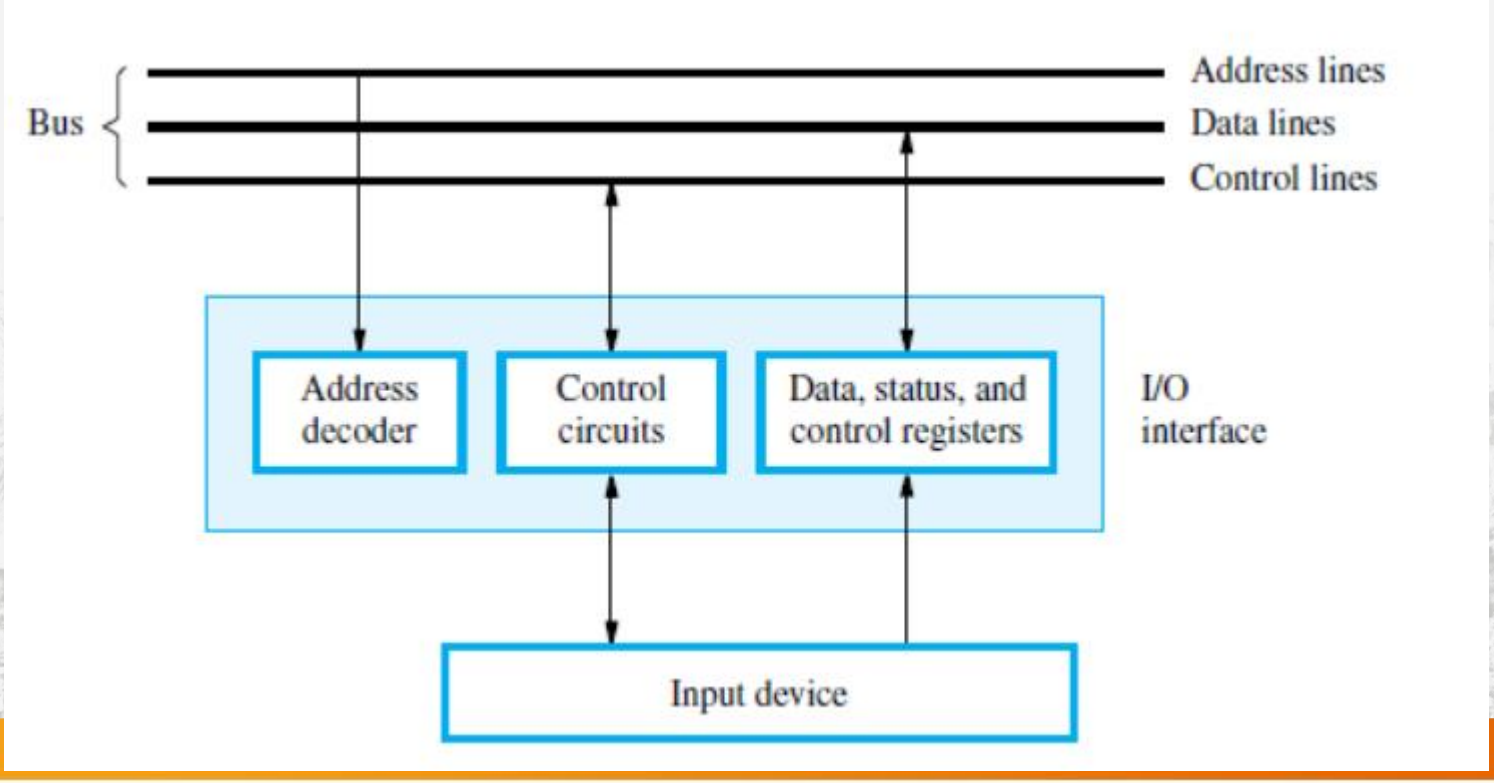
- The method that is used to transfer information between internal storage and external I/O devices is known as I/O interface.

- Peripherals connected to a computer need special communication links for interfacing with CPU.
- In computer system, there are special hardware components between the CPU and peripherals to control or manage the input-output transfers.
- These components are called **input-output interface units** because they provide communication links between processor bus and peripherals. They provide a method for transferring information between internal system and input-output devices.

- The I/O Bus consists of data lines, address lines and control lines. The I/O bus from the processor is attached to all peripherals interface.
- To communicate with a particular device, the processor places a device address on address lines. Each Interface decodes the address and control received from the I/O bus, interprets them for peripherals and provides signals for the peripheral controller.
- It is also synchronizes the data flow and supervises the transfer between peripheral and processor. Each peripheral has its own controller.

- For example, the printer controller controls the paper motion, the print timing. The control lines are referred as I/O command. The commands are as following:
  **Control command**- A control command is issued to activate the peripheral and to inform it what to do.
  **Status command**- A status command is used to test various status conditions in the interface and the peripheral.
  **Data Output command**- A data output command causes the interface to respond by transferring data from the bus into one of its registers.
  **Data Input command**- The data input command is the opposite of the data output.
- In this case the interface receives on item of data from the peripheral and places it in its buffer register. I/O Versus Memory Bus

- There are 3 ways that computer buses can be used to communicate with memory and I/O:
  1. Use two Separate buses, one for memory and other for I/O.
  2. Use one common bus for both memory and I/O but separate control lines for each.
  3. Use one common bus for memory and I/O with common control lines



# Modes of I/O Data Transfer

- Modes of I/O Data Transfer
- Data transfer between the central unit and I/O devices can be handled in generally three types of modes which are given below:
- Programmed I/O
- Interrupt Initiated I/O
- Direct Memory Access

# Programmed I/O

- Programmed I/O instructions are the result of I/O instructions written in computer program. Each data item transfer is initiated by the instruction in the program.
- Normally the transfer is from a CPU register to peripheral device or vice versa. Once the data is initiated the CPU starts monitoring the interface to see when next transfer can made. The instructions of the program keep close tabs on everything that takes place in the interface unit and the I/O devices.

- The transfer of data requires three instructions:
  - Read the status register.
  - Check the status of the flag bit and branch to step 1 if not set or to step 3 if set.
  - Read the data register.

# Drawback of the Programmed I/O:

The main drawback of the Program Initiated I/O was that the CPU has to monitor the units all the times when the program is executing.

Thus the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer.

This is a time consuming process and the CPU time is wasted a lot in keeping an eye to the executing of program.

# Interrupt Driven I/O

- In the programmed I/O method the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer. This is time consuming process because it keeps the processor busy needlessly.

- This problem can be overcome by using **interrupt initiated I/O**. In this when the interface determines that the peripheral is ready for data transfer, it generates an interrupt. After receiving the interrupt signal, the CPU stops the task which it is processing and service the I/O transfer and then returns back to its previous processing task.

# Direct Memory Access

- Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This technique is known as **DMA**.
- In this, the interface transfer data to and from the memory through memory bus. A DMA controller manages to transfer data between peripherals and memory unit.
- Many hardware systems use DMA such as disk drive controllers, graphic cards, network cards and sound cards etc. It is also used for intra chip data transfer in multicore processors. In DMA, CPU would initiate the transfer, do other operations while the transfer is in progress and receive an interrupt from the DMA controller when the transfer has been completed.

- DMA Controller
- Interface which allows I/O transfer directly between memory and Device, freeing CPU for Other tasks
- CPU initializes DMA Controller by sending memory address and the block size (number of words)

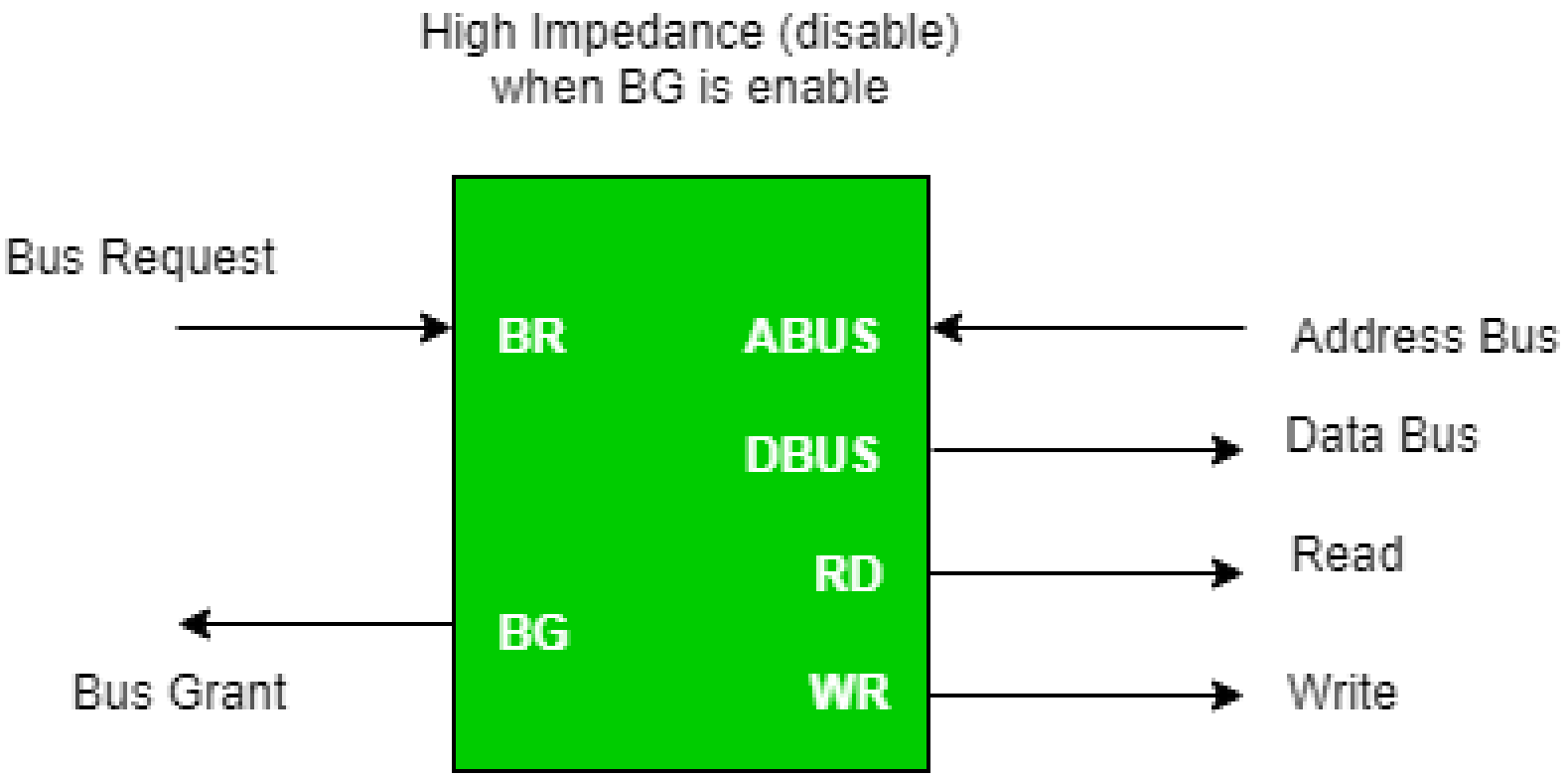
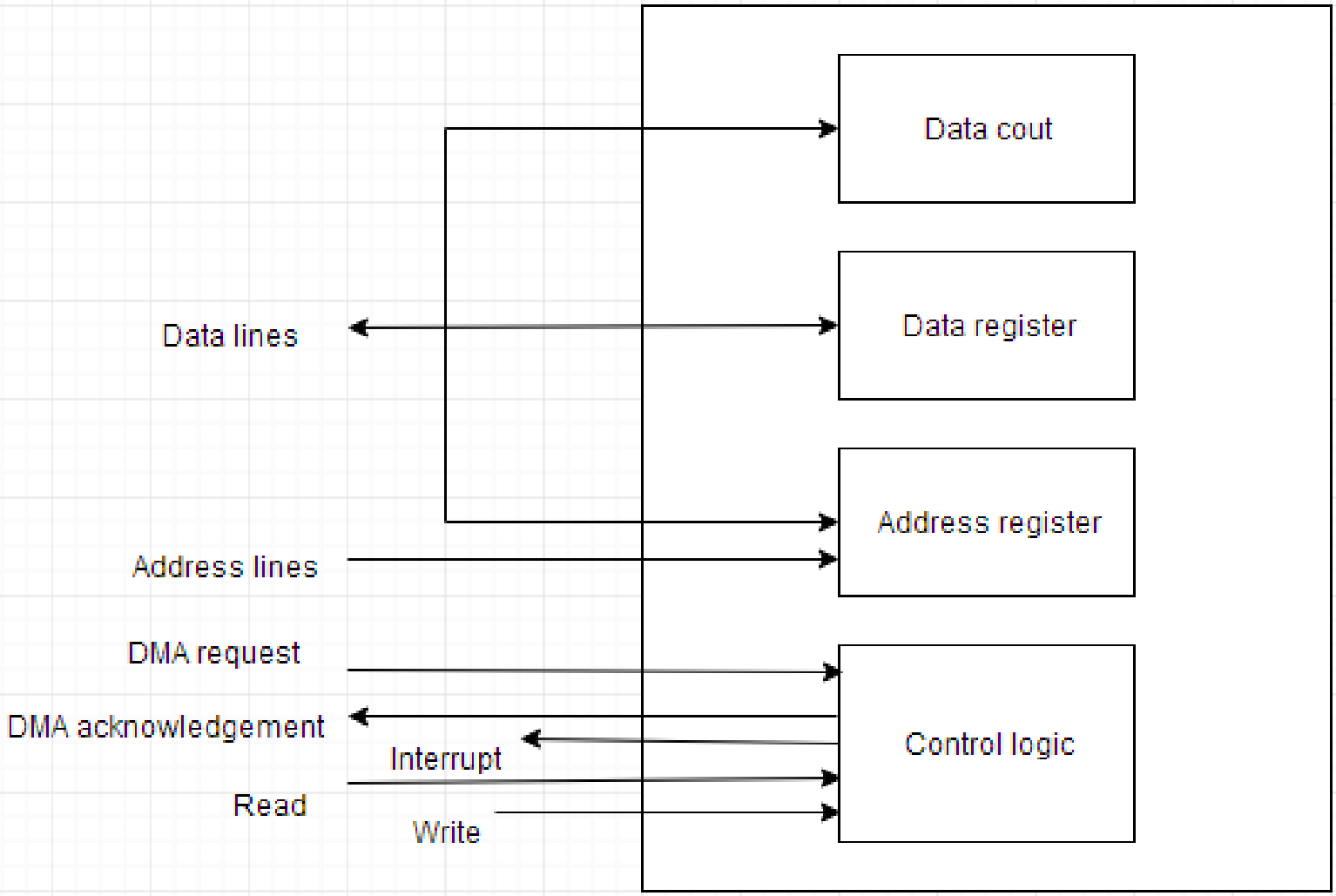



Figure - CPU Bus Signals for DMA Transfer

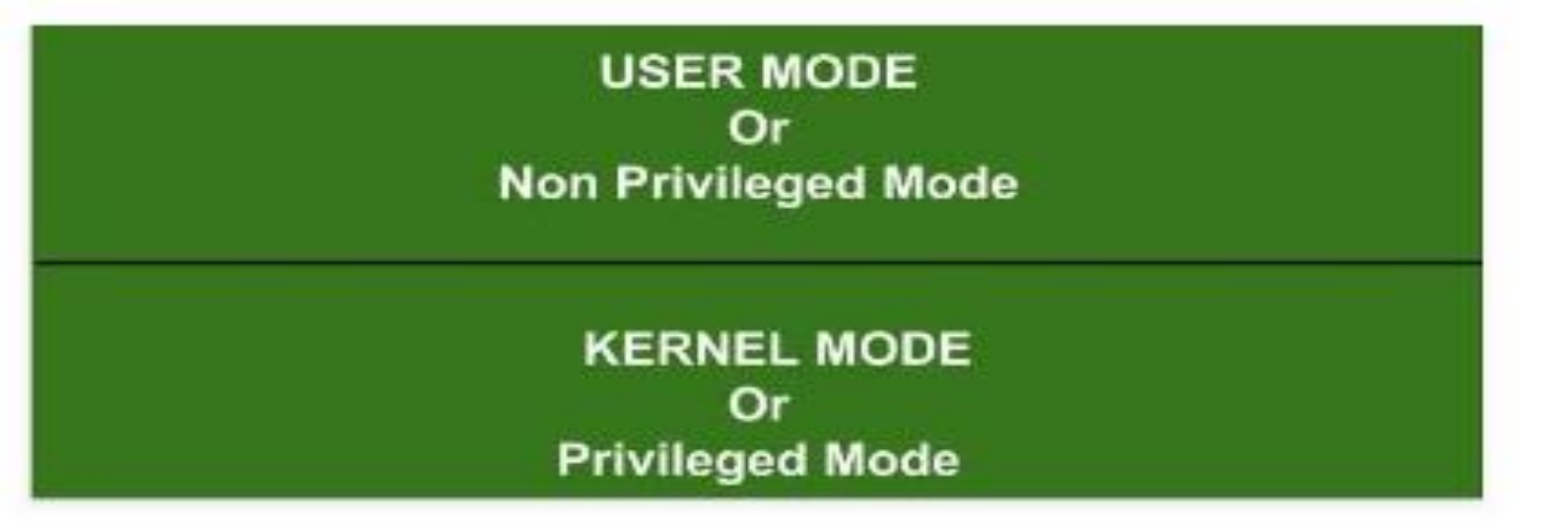
- **Bus Request :** It is used by the DMA controller to request the CPU to hand over the control of the buses.
- **Bus Grant :** It is activated by the CPU to Inform the external DMA controller that the buses are in high impedance state and the requesting DMA can take control of the buses. Once the DMA has taken the control of the buses it transfers the data. This transfer can take place in many ways.
- **Types of DMA transfer using DMA controller:**
- **Burst Transfer :**
  DMA returns the bus after complete data transfer. A register is used as a byte count, being decremented for each byte transfer, and upon the byte count reaching zero, the DMAC will release the bus. When the DMAC operates in burst mode, the CPU is halted for the duration of the data transfer.

# Block Diagram of DMA



# privileged and non-privileged instructions

- In any Operating System, it is necessary to have **Dual Mode Operation** to ensure protection and security of the System from unauthorized or errant users . This Dual Mode separates the User Mode from the System Mode or Kernel Mode.



# Privileged Instructions

- The Instructions that can run only in Kernel Mode are called Privileged Instructions
- Privileged Instructions possess the following characteristics :
- (i) If any attempt is made to execute a Privileged Instruction in User Mode, then it will not be executed and treated as an illegal instruction. The Hardware traps it to the Operating System.
- (ii) Before transferring the control to any User Program, it is the responsibility of the Operating System to ensure that the **Timer** is set to interrupt. Thus, if the timer interrupts then the Operating System regains the control.
  Thus, any instruction which can modify the contents of the Timer is a Privileged Instruction.
- (iii) Privileged Instructions are used by the Operating System in order to achieve correct operation.

(iv) Various examples of Privileged Instructions include:

- I/O instructions and Halt instructions
- Turn off all Interrupts
- Set the Timer
- Context Switching
- Clear the Memory or Remove a process from the Memory
- Modify entries in Device-status table

# Non-Privileged Instructions

- The Instructions that can run only in User Mode are called Non-Privileged Instructions .
- Various examples of Non-Privileged Instructions include:
- Reading the status of Processor
- Reading the System Time
- Generate any Trap Instruction
- Sending the final printout of Printer

- Also, it is important to note that in order to change the mode from Privileged to Non-Privileged, we require a Non-privileged Instruction that does not generate any interrupt.

# software interrupts and exceptions

- A condition or event that interrupts the normal flow of control in a program.
- Basically interrupt is a signal.
- Signal means signal can be generated from particular hardware or signal can be generated from particular program execution.
- Two types
- Hardware interrupts occur due to a change in state of some hardware.
- Software interrupts are triggered by the execution of a machine instruction.
- Signal used to send to a device to say “Stop doing what you are doing and listen to me. I have got something for you.”

# Software Interrupt

- A software interrupts is a type of interrupt generated by executing an instructions is called software interrupt. A software interrupt is invoked by software.
- Two types of software interrupt
- Normal interrupts: caused by the software instructions
- Exception: unplanned interrupts while executing a program is called exception.
- E.g. while executing a program if we got a value which should be divided by zero is called a exception.
- There are eight software interrupts in 8085 microprocessor
- RST0-RST7

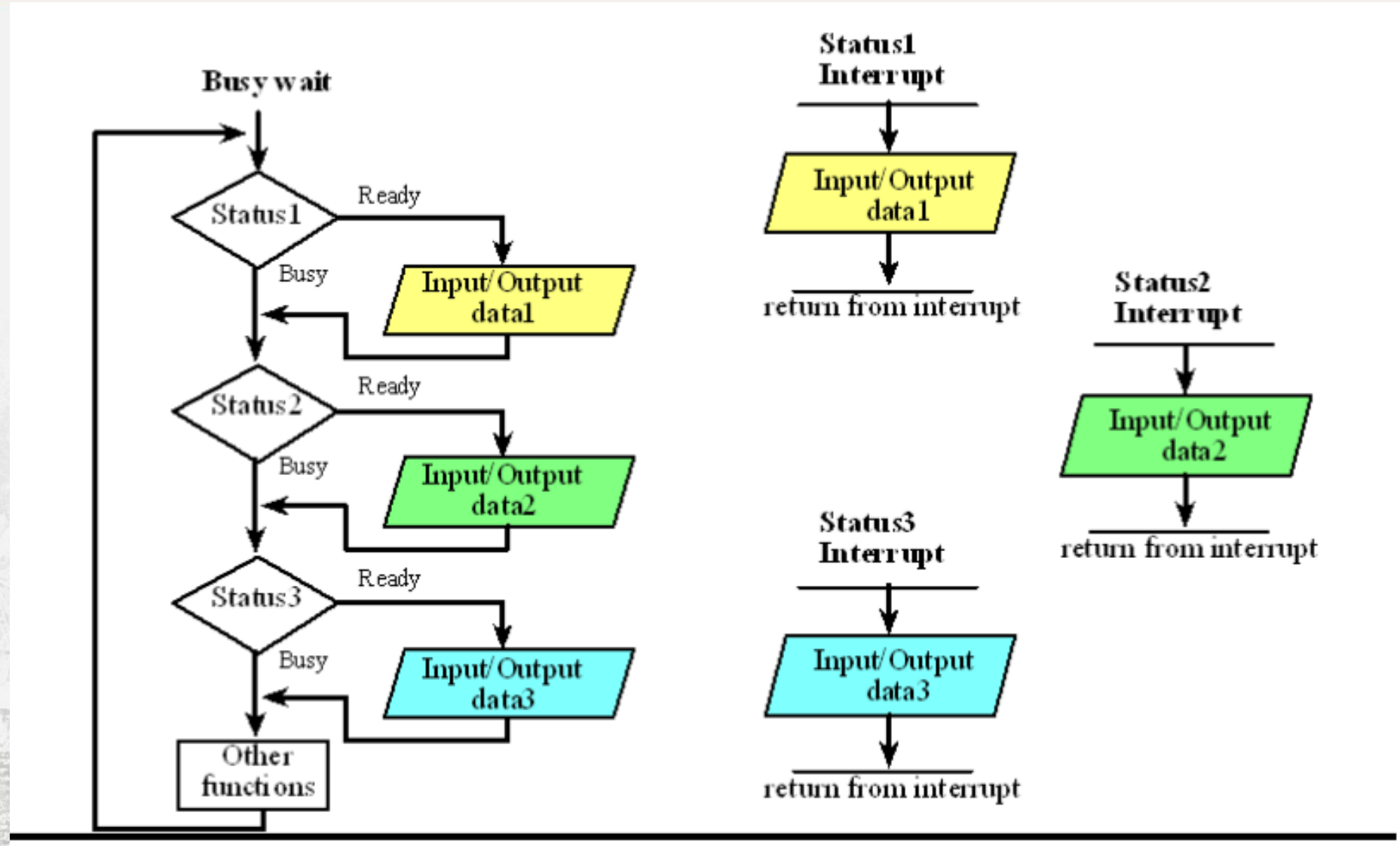
# Programs and processes-Role of Interrupts in process state transitions

- An **interrupt** is the automatic transfer of software execution in response to a hardware event that is asynchronous with the current software execution. This hardware event is called a **trigger**. The hardware event can either be a busy to ready transition in an external I/O device or an internal event

- When the hardware needs service, signified by a busy to ready state transition, it will request an interrupt by setting its trigger flag. A **thread** is defined as the path of action of software as it executes. The execution of the interrupt service routine is called a background thread. This thread is created by the hardware interrupt request and is killed when the interrupt service routine returns from interrupt
- A new thread is created for each interrupt request

# Role of Interrupts in process state transitions

- It is important to consider each individual request as a separate thread because local variables and registers used in the interrupt service routine are unique and separate from one interrupt event to the next interrupt.
- In a **multi-threaded** system, we consider the threads as cooperating to perform an overall task. Consequently we will develop ways for the threads to communicate (e.g., FIFO) and to synchronize with each other. Most embedded systems have a single common overall goal.
- A **process** is also defined as the action of software as it executes. Processes do not necessarily cooperate towards a common shared goal. Threads share access to I/O devices, system resources, and global variables, while processes have separate global variables and system resources. **Processes do not share I/O devices.**


Busy wait
Status1
Ready
Busy
Input/Output data1
Status2
Ready
Busy
Input/Output data2
Status3
Ready
Busy
Input/Output data3
Other functions
Status1 Interrupt
Input/Output data1
return from interrupt
Status2 Interrupt
Input/Output data2
return from interrupt
Status3 Interrupt
Input/Output data3
return from interrupt

# I/O device interfaces-

# SCSI, USB

# I/O device interfaces-SCSI

- The acronym SCSI stands for **Small Computer System Interface**. It refers to a standard bus defined by the American National Standards Institute (ANSI) under the designation X3.131 .
- In the original specifications of the standard, devices such as disks are connected to a computer via a **50-wire cable**, which can be **up to 25 meters** in length and can transfer data at rates up to **5 megabytes/s.**
- The SCSI bus standard has undergone many revisions, and its data transfer capability has increased very rapidly, almost doubling every two years. **SCSI-2 and SCSI-3** have been defined, and each has several options.

- A controller connected to a SCSI bus is one of two types – an initiator or a target. An initiator has the ability to select a particular target and to send commands specifying the operations to be performed. Clearly, the controller on the processor side, such as the SCSI controller, must be able to operate as an initiator. The disk controller operates as a target.
- It carries out the commands it receives from the initiator. The initiator establishes a logical connection with the intended target. Once this connection has been established, it can be suspended and restored as needed to transfer commands and bursts of data. While a particular connection is suspended, other device can use the bus to transfer information.
- This ability to overlap data transfer requests is one of the key features of the SCSI bus that leads to its high performance.

- Data transfers on the SCSI bus are always controlled by the target controller. To send a command to a target, an initiator requests control of the bus and, after winning arbitration, selects the controller it wants to communicate with and hands control of the bus over to it.

- The processor sends a command to the SCSI controller, which causes the following sequence of event to take place:

1. The SCSI controller, acting as an initiator, contends for control of the bus.
2. When the initiator wins the arbitration process, it selects the target controller and hands over control of the bus to it.

3. The target starts an output operation (from initiator to target); in response to this, the initiator sends a command specifying the required read operation.

4. The target, realizing that it first needs to perform a disk seek operation, sends a message to the initiator indicating that it will temporarily suspend the connection between them. Then it releases the bus.

- 5. The target controller sends a command to the disk drive to move the read head to the first sector involved in the requested read operation. Then, it reads the data stored in that sector and stores them in a data buffer. When it is ready to begin transferring data to the initiator, the target requests control of the bus. After it wins arbitration, it reselects the initiator controller, thus restoring the suspended connection.

- 6. The target transfers the contents of the data buffer to the initiator and then suspends the connection again. Data are transferred either 8 or 16 bits in parallel, depending on the width of the bus.

- 7. The target controller sends a command to the disk drive to perform another seek operation. Then, it transfers the contents of the second disk sector to the initiator as before. At the end of these transfers, the logical connection between the two controllers is terminated.
- 8. As the initiator controller receives the data, it stores them into the main memory using the DMA approach.
- 9. The SCSI controller sends as interrupt to the processor to inform it that the requested operation has been completed.

# USB

- The USB has been designed to meet several key objectives:

❖ Provide a simple, low-cost, and easy to use interconnection system that overcomes the difficulties due to the limited number of I/O ports available on a computer.

❖Accommodate a wide range of data transfer characteristics for I/O devices, including telephone and Internet connections.

❖Enhance user convenience through a “plug-and-play” mode of operation

# USB Structure

- A serial transmission format has been chosen for the USB because a serial bus satisfies the low-cost and flexibility requirements.

- Clock and data information are encoded together and transmitted as a single signal. Hence, there are no limitations on clock frequency or distance arising from data skew.

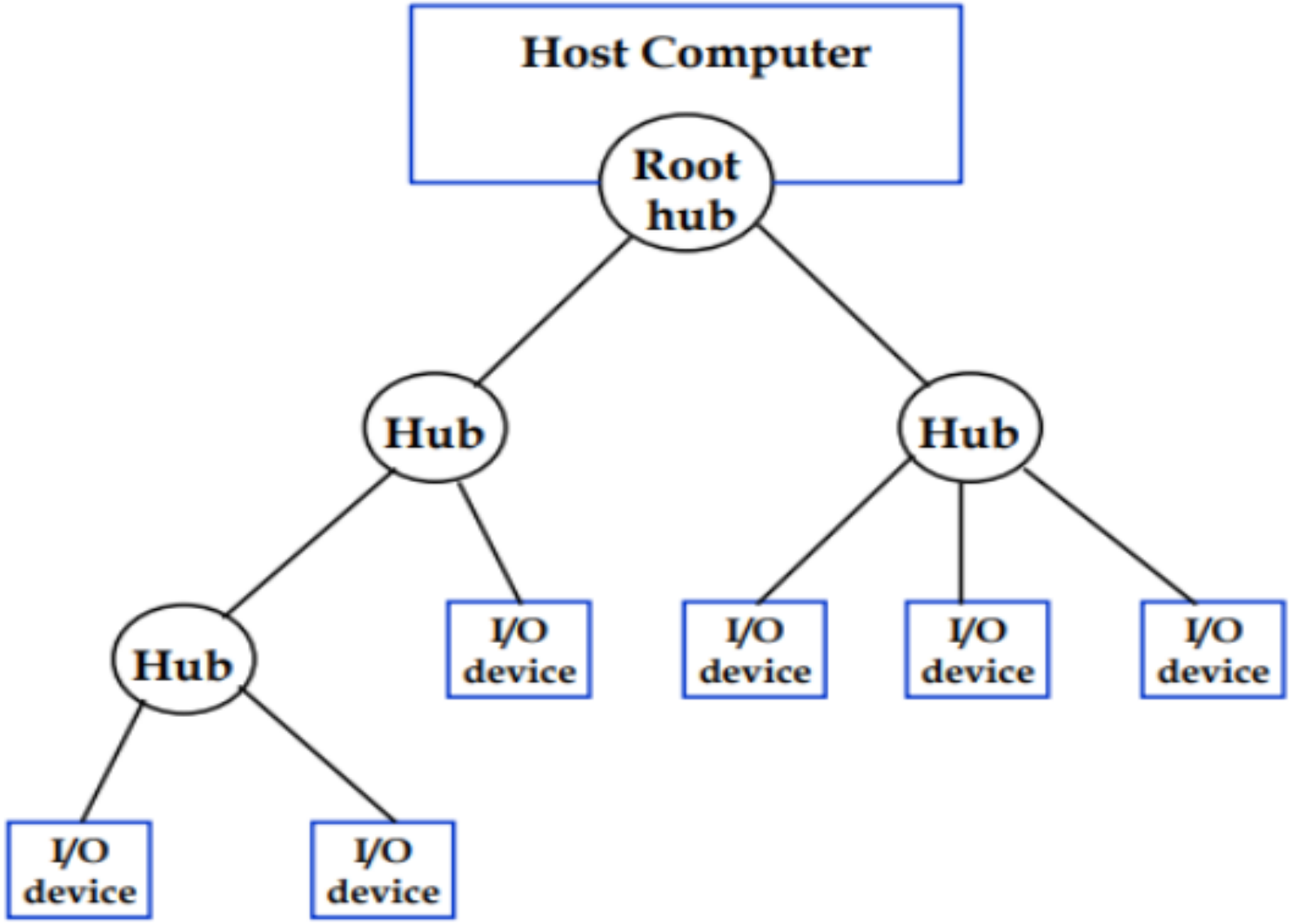
- To accommodate a large number of devices that can be added or removed at any time, the USB has the tree structure. Each node of the tree has a device called a hub, which acts as an intermediate control point between the host and the I/O device. At the root of the tree, a root hub connects the entire tree to the host computer.

- The tree structure enables many devices to be connected while using only simple point-to-point serial links.

Each hub has a number of ports where devices may be connected, including other hubs.

- In normal operation, a hub copies a message that it receives from its upstream connection to all its downstream ports. As a result, a message sent by the host computer is broadcast to all I/O devices, but only the addressed device will respond to that message.

- A message sent from an I/O device is sent only upstream towards the root of the tree and is not seen by other devices. Hence, USB enables the host to communicate with the I/O devices, but it does not enable these devices to communicate with each other.


Host Computer
Root hub
Hub
Hub
Hub
I/O device
I/O device
I/O device
I/O device
I/O device
I/O device

# USB Protocols:

All information transferred over the USB is organized in 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.

**Control** packets perform such tasks as addressing a device to initiate data transfer, acknowledging that data have been received correctly, or indicating an error.

**Data** packets carry information that is delivered to a device. For example, input and output data are transferred inside data packets

# USB Speed

1.5 Mbs – Low Speed – USB 1.0

12 Mbs – Full Speed – USB 1.0

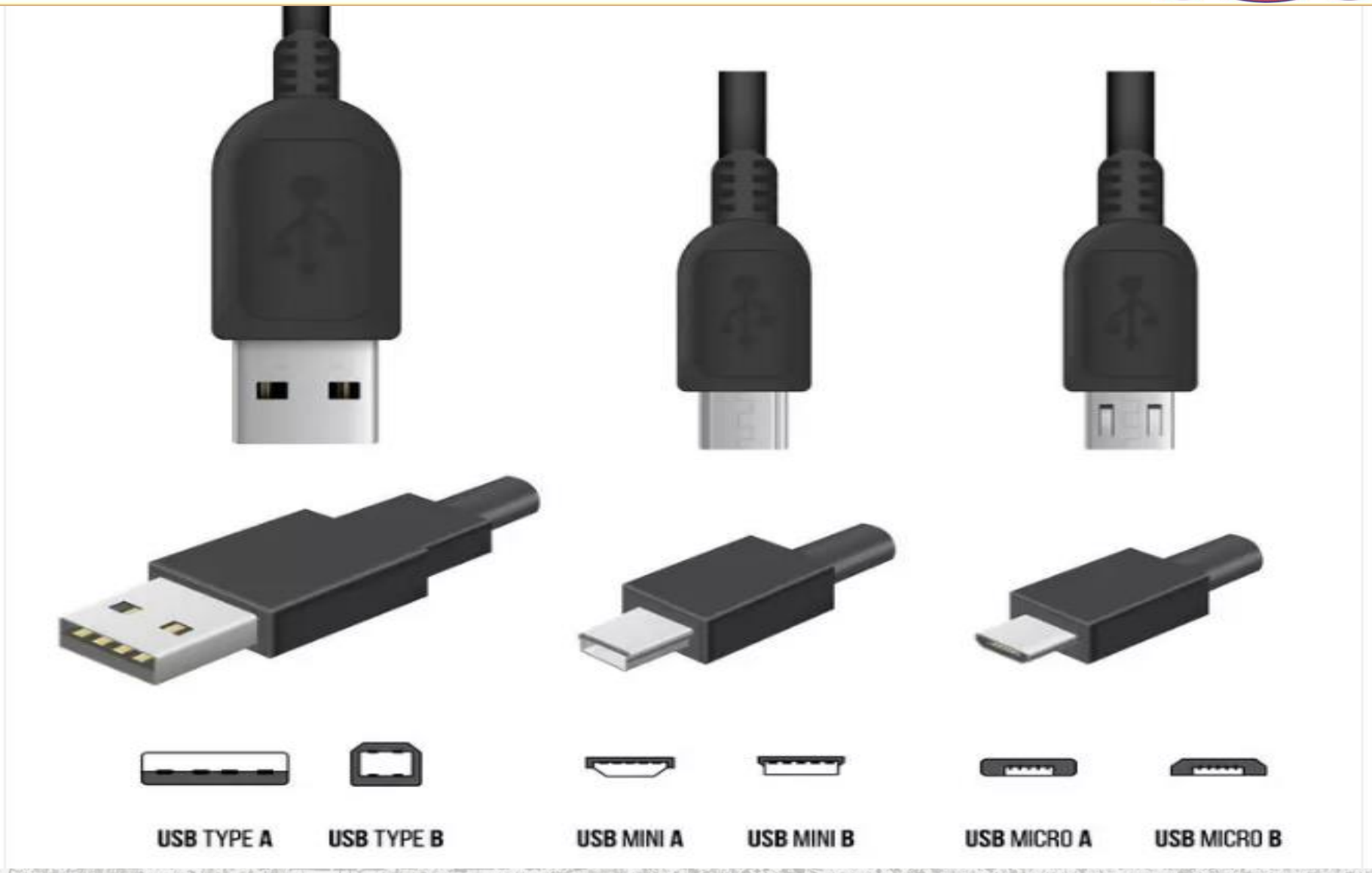
480 Mbs – High Speed – USB 2.0

5 Gbs – Super Speed – USB 3.0

10 Gbs – Super Speed+ – USB 3.1

## USB REVISIONS

| | Top speed | Max power output | Power direction | Cable configuration | Availability |
|---|---|---|---|---|---|
| USB 1.1 | 12Mbps | 2.5V, 500mA | Host to peripheral | Type-A to Type-B | 1998 |
| USB 2.0 | 480Mbps | 2.5V, 1.8A | Host to peripheral | Type-A to Type-B | 2000 |
| USB 3.0 | 5Gbps | 5V, 1.8A | Host to peripheral | Type-A to Type-B | 2008 |
| USB 3.1 | 10Gbps | 20V, 5A | Bi-directional | Type-C both ends, reversible plug orientation | 2015 |


USB TYPE A
USB TYPE B
USB MINI A
USB MINI B
USB MICRO A
USB MICRO B


USB TYPE A
USB TYPE B
USB TYPE C
USB MICRO B
USB MINI B