Unit-5

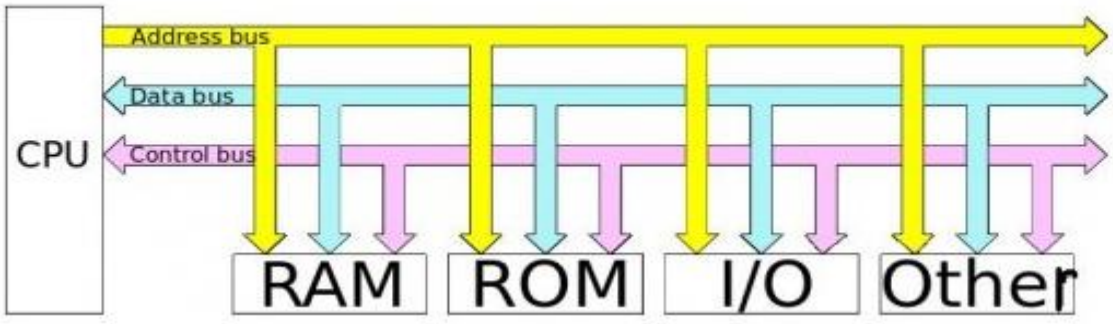
Bus Architecture

Types of Buses in Computer Architecture

Computers comprises of many internal components and in order for these components to communicate with each other, a *'bus'* is used for that purpose.

A bus is a common pathway through which information flows from one component to another. This pathway is used for communication purpose and can be established between two or more computer components. We are going to review different computer bus architectures that are used in computers.

Different Types of Computer Buses



Functions of Buses in Computers

The functions of buses can be summarized as below:
1. Data sharing - All types of buses found on a computer must be able to transfer data between the computer peripherals connected to it.

The data is transferred in in either serial or parallel, which allows the exchange of 1, 2, 4 or even 8 bytes of data at a time. (A byte is a group of 8 bits). Buses are classified depending on how many bits they can move at the same time, which means that we have 8-bit, 16-bit, 32-bit or even 64-bit buses.

2. Addressing - A bus has address lines, which match those of the processor. This allows data to be sent to or from specific memory locations.

3. Power - A bus supplies power to various peripherals that are connected to it.

4. Timing - The bus provides a system clock signal to synchronize the peripherals attached to it with the rest of the system.

The expansion bus facilitates the easy connection of additional components and devices on a computer for example the addition of a TV card or sound card.

Expansion Bus Types

These are some of the common expansion bus types that have ever been used in computers:

- ISA - Industry Standard Architecture
- EISA - Extended Industry Standard Architecture
- MCA - Micro Channel Architecture
- VESA - Video Electronics Standards Association
- PCI - Peripheral Component Interconnect
- PCMCIA - Personal Computer Memory Card Industry Association (Also called PC bus)
- AGP - Accelerated Graphics Port
- SCSI - Small Computer Systems Interface.

ISA Bus

This is the most common type of early expansion bus, which was designed for use in the original IBM PC.The IBM PC-XT used an 8-bit bus design. This means that the data transfers take place in 8 bit chunks (i.e. one byte at a time) across the bus. The ISA bus ran at a clock speed of 4.77 MHz.

For the 80286-based IBM PC-AT, an improved bus design, which could transfer 16-bits of data at a time, was announced. The 16-bit version of the ISA bus is sometimes known as the AT bus. (AT-Advanced Technology)

The improved AT bus also provided a total of 24 address lines, which allowed 16MB of memory to be addressed. The AT bus was backward compatible with its 8-bit predecessor and allowed 8-bit cards to be used in 16-bit expansion slots.

When it first appeared the 8-bit ISA bus ran at a speed of 4.77MHZ – the same speed as the processor. It was improved over the years and eventually the AT bus ran at a clock speed of 8MHz.

MCA (Micro Channel Architecture)

This bus was developed by IBM as a replacement for ISA when they designed the PS/2 PC which was launched in 1987.

The bus offered a number of technical improvements over the ISA bus. For instance, the MCA runs at a faster speed of 10MHz and can support either 16-bit or 32-bit data. It also supports bus mastering - a technology that placed a mini-processor on each expansion card. These mini-processors controlled much of the data transfer allowing the CPU to perform other tasks.

One advantage of MCA was that the plug-in cards were software configurable i.e. they required minimal intervention by the user when configuring.

The MCA expansion bus did not support ISA cards and IBM decided to charge other manufacturers royalties for use of the technology. This made it unpopular and it is now an obsolete technology.

EISA (Extended Industry Standard Architecture)

It was developed by a group of manufactures as an alternative to MCA. It was designed to use a 32-bit data path and provided 32 address lines giving access to 4GB of memory.

Like the MCA, EISA offered a disk-based setup for the cards, but it still ran at 8MHz in order for it to be compatible with ISA.

The EISA expansion slots are twice as deep as an ISA slot. If an ISA card is placed in an EISA slot it will use only the top row of connectors, whereas a full EISA card uses both rows. It offered bus mastering.

EISA cards were relatively expensive and were normally found on high-end workstations and network servers.

VESA Bus

Also known as the Local bus or the VESA-Local bus. VESA (Video Electronics Standards Association) was invented to help standardize PCs video specifications, thus solving the problem of proprietary technology where different manufacturers were attempting to develop their own buses.

The VL Bus provides 32-bit data path and can run at 25 or 33MHZ. It ran at the same clock frequency as the host CPU. But this became a problem as processor speeds increased because, the faster the peripherals are required to run, the more expensive they are to manufacture.

It was difficult to implement the VL-Bus on newer chips such as the 486s and the new Pentiums and so eventually the VL-Bus was superseded by PCI.

VESA slots have extra set of connectors and therefore the cards are larger. The VESA design was backward compatible with the older ISA cards.

Peripheral Component Interconnect

Peripheral Component Interconnect (PCI) is one of the latest developments in bus architecture and is the current standard for PC expansion cards. It was developed by Intel and launched as the expansion bus for the Pentium processor in 1993. It is a local bus like VESA i.e. it connects the CPU, memory and peripherals to wider, faster data pathway.

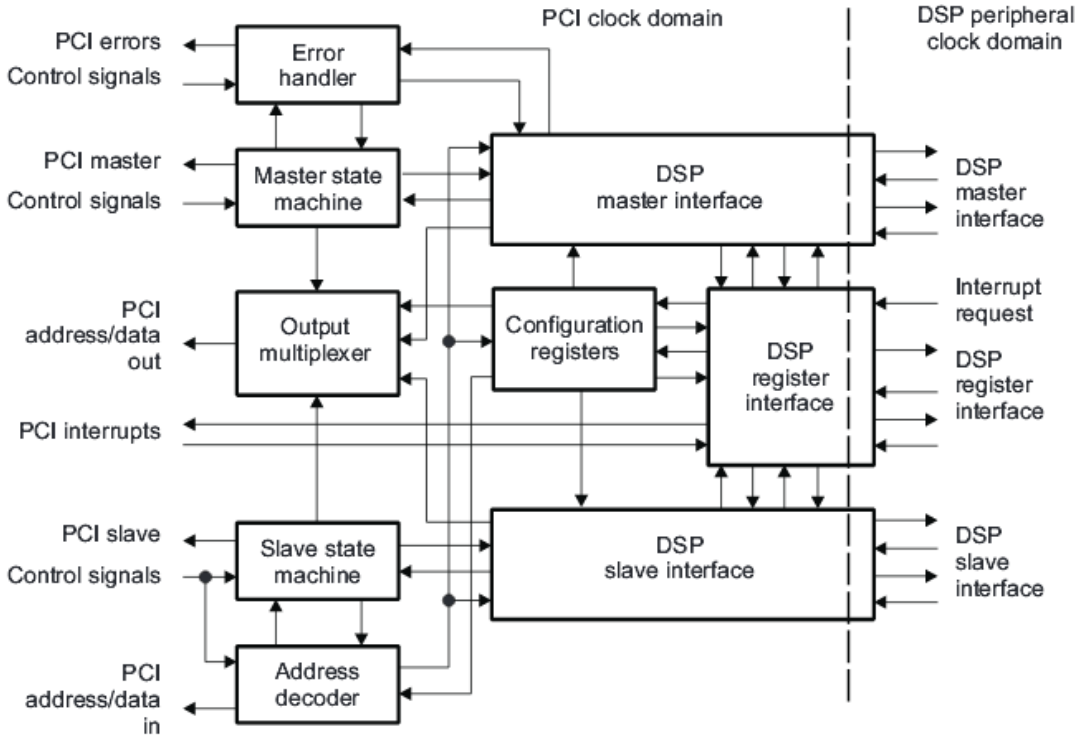
PCI supports both 32-bit and 64-bit data width; therefore it is compatible with 486s and Pentiums. The bus data width is equal to the processor, for example, a 32 bit processor would have a 32 bit PCI bus, and operates at 33MHz.

PCI was used in developing Plug and Play (PnP) and all PCI cards support PnP i.e. the user can plug a new card into the computer, power it on and it will “self identify” and “self specify” and start working without manual configuration using jumpers.

Unlike VESA, PCI supports bus mastering that is, the bus has some processing capability and therefore the CPU spends less time processing data. Most PCI cards are designed for 5v, but there are also 3v and dual-voltage cards, Keying slots are used to differentiate 3v and 5v cards and slots to ensure that a 3v card is not slotted into a 5v socket and vice versa.

Functional Block Diagram

Figure 1. PCI Block Diagram



The PCI module consists of the following blocks:

- **Address Decoder:** This block latches transaction control information from the PCI bus and decodes that information to determine if the transaction was targeted to the PCI. This block instructs the slave state machine to either accept or ignore slave transactions as they are presented on the PCI bus.
- **Slave State Machine:** This block generates and monitors all of the PCI signals necessary for accepting transactions on the bus. All of the slave PCI protocols handling functions are split between the address decoder and slave state machine blocks.
- **DSP Slave Interface:** This block accepts transactions from the slave state machine and passes those transactions to the targeted DSP resource (for example, DDR2 memory controller). This block performs the asynchronous decoupling between the PCI clock domain and the peripheral clock domain for slave transactions. It also implements the address translation control registers and performs address translation for slave transactions.
- **DSP Master Interface:** This block accepts bus transactions from DSP masters (for example, EDMA transfer controllers) and passes those transactions on to the master state machine. It performs the asynchronous decoupling between the peripheral clock domain and the PCI clock domain for master transactions. This block implements the address translation control registers and also performs the address translation for master transactions.
- **Master State Machine:** This block generates and monitors all of the PCI signals necessary for initiating transactions on the bus. The majority of the master PCI protocols handling functions are implemented in this block. This block responds to transfer requests that are presented to it from the DSP master block.

Signal Descriptions

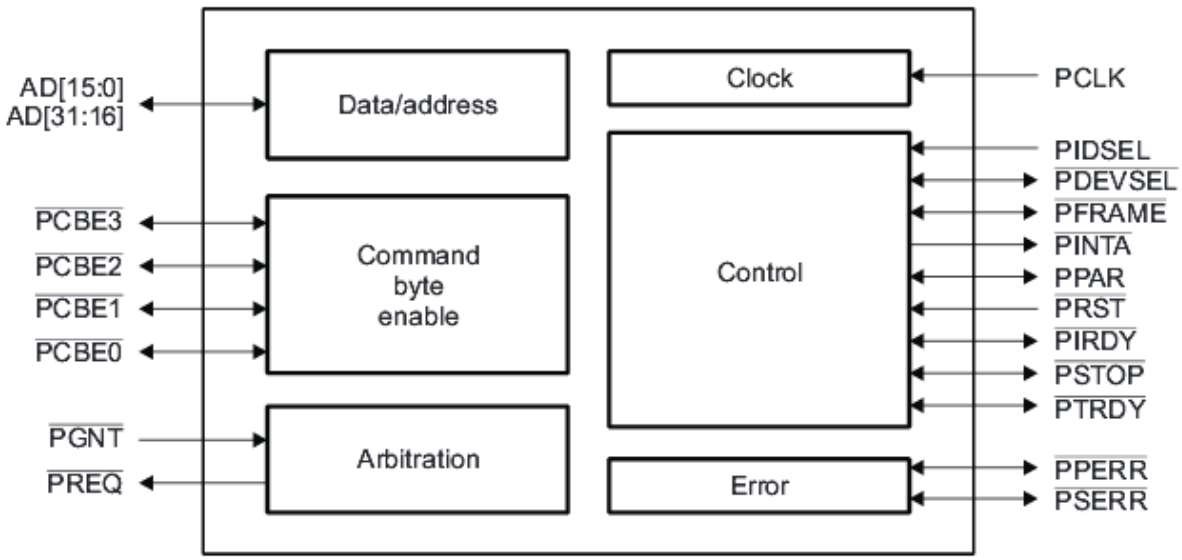


Table 1. PCI Pin Description

Pin Name	Type[1]	Description
PFRAME	I/O/Z	PCI Frame
PDEVSEL	I/O/Z	PCI Device Select
PSTOP	I/O/Z	PCI Transaction Stop Indicator
PCLK	I	PCI Clock
PCBE[3:0]	I/O/Z	PCI Command/Byte Enables
PPAR	I/O/Z	PCI Parity
PPERR	I/O/Z	PCI Parity Error
PSERR	I/O/Z	PCI System Error
PIRDY	I/O/Z	PCI Initiator Ready
PINTA	O/Z	PCI Interrupt A
PRST	I	PCI Reset
PIDSEL	I	PCI Initialization Select
PTRDY	I/O/Z	PCI Transmitter Ready
AD[31:16]	I/O/Z	PCI Data/Address bus [31:16]
AD[15:0]	I/O/Z	PCI Data/Address bus [16:0]
PREQ	O/Z	PCI Bus Request
PGNT	I	PCI Bus Grant

(1) I = Input, O = Output, Z = High impedance

PCI Address Spaces

The CPU and the PCI devices need to access memory that is shared between them. This memory is used by device drivers to control the PCI devices and to pass information between them. Typically the shared memory contains control and status registers for the device. These registers are used to control the device and to read its status. For example, the PCI SCSI device driver would read its status register to find out if the SCSI device was ready to write a block of information to the SCSI disk. Or it might write to the control register to start the device running after it has been turned on.

The CPU's system memory could be used for this shared memory but if it were, then every time a PCI device accessed memory, the CPU would have to stall, waiting for the PCI device to finish. Access to memory is generally limited to one system component at a time. This would slow the system down. It is also not a good idea to allow the system's peripheral devices to access main memory in an uncontrolled way. This would be very dangerous; a rogue device could make the system very unstable.

Peripheral devices have their own memory spaces. The CPU can access these spaces but access by the devices into the system's memory is very strictly controlled using DMA (Direct Memory Access) channels. ISA devices have access to two address spaces, ISA I/O (Input/Output) and ISA memory. PCI has three; PCI I/O, PCI Memory and PCI Configuration space. All of these address spaces are also accessible by the

CPU with the the PCI I/O and PCI Memory address spaces being used by the device drivers and the PCI Configuration space being used by the PCI initialization code within the Linux kernel.

Accelerated Graphics Port

The Accelerated Graphics Port (AGP or A.G.P.) is a high performance, component level interconnect targeted at 3D graphical display applications and is based on a set of performance extensions or enhancements to PCI. This document specifies the A.G.P. interface, and provides some design suggestions for effectively using it in high performance 3D graphics display applications

Relationship to PCI

The A.G.P. interface specification uses the 66 MHz PCI (Revision 2.1) specification as an operational baseline, and provides three significant erformance extensions or enhancements to the PCI specification which are intended to optimize the A.G.P. for high performance 3D graphics applications. These A.G.P. extensions are NOT described in, or required by, the PCI specification (Rev 2.1). These extensions are

- Deeply pipelined memory read and write operations, fully hiding memory access latency.
- Demultiplexing of address and data on the bus, allowing almost 100% bus efficiency.
- AC timing for 133 MHz data transfer rates, allowing for real data throughput in excess of 500 MB/sec.

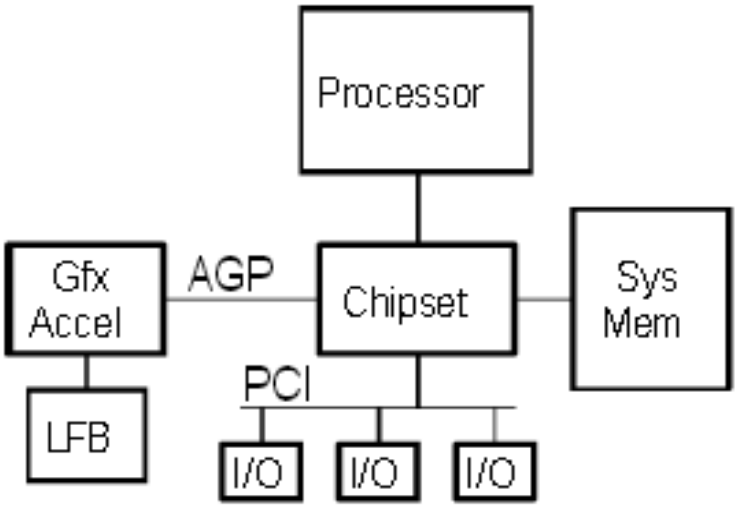


Figure 1-1 System Block Diagram: A.G.P. and PCI Relationship

The need for high quality and very fast performance of video on computers led to the development of the Accelerated Graphics Port (AGP). The AGP Port is connected to the CPU and operates at the speed of the processor bus. This means that video information can be sent more quickly to the card for processing.

The AGP uses the main PC memory to hold 3D images. In effect, this gives the AGP video card an unlimited amount of video memory. To speed up the data transfer, Intel designed the port as a direct path to the PC's main memory.

Data transfer rate ranges from 264 Mbps to 528mbps, 800 Mbps up to 1.5 Gbps. AGP connector is identified by its brown colour.

Personal Computer Memory Card Industry Association (PC Card)

The Personal Computer Memory Card Industry Association was founded to provide a standard bus for laptop computers. So it is basically used in the small computers.

Small Computer System Interface

Short for Small Computer System Interface, a parallel interface standard used by Apple Macintosh computers,PC's and Unix systems for attaching peripheral devices to a computer.

Universal Serial Bus (USB)

Universal Serial Bus (USB) is a set of interface specifications for high speed wired communication between electronics systems peripherals and devices with or without PC/computer. The USB was originally developed in 1995 by many of the industry leading companies like Intel, Compaq, Microsoft, Digital, IBM, and Northern Telecom.

The major goal of USB was to define an external expansion bus to add peripherals to a PC in easy and simple manner. The new external expansion architecture, highlights,

1. PC host controller hardware and software
2. Robust connectors and cable assemblies
3. Peripheral friendly master-slave protocols
4. Expandable through multi-port hubs.

USB offers users simple connectivity. It eliminates the mix of different connectors for different devices like printers, keyboards, mice, and other peripherals. That means USB-bus allows many peripherals to be connected using a single standardized interface socket. Another main advantage is that, in USB environment, DIP-switches are not necessary for setting peripheral addresses and IRQs. It supports all kinds of data, from slow mouse inputs to digitized audio and compressed video.

USB also allows hot swapping. The "hot-swapping" means that the devices can be plugged and unplugged without rebooting the computer or turning off the device. That means, when plugged in, everything configures automatically. So the user needs not worry about terminations, terms such as IRQs and port addresses, or rebooting the computer. Once the user is finished, they can simply unplug the cable out, the host will detect its absence and automatically unload the driver. This makes the USB a plug-and-play interface between a computer and add-on devices.

USB sends data in serial mode i.e. the parallel data is serialized before sends and de-serialized after receiving.

The benefits of USB are low cost, expandability, auto-configuration, hot-plugging and outstanding performance. It also provides power to the bus, enabling many peripherals to operate without the added need for an AC power adapter.

Various versions USB:

As USB technology advanced the new version of USB are unveiled with time. Let us now try to understand more about the different versions of the USB.

USB1.0: Version 0.7 of the USB interface definition was released in November 1994. But USB 1.0 is the original release of USB having the capability of transferring 12 Mbps, supporting up to 127 devices. And as we know it was a combined effort of some large players on the market to define a new general device interface for computers. This USB 1.0 specification model was introduced in January1996. The data transfer rate of this version can accommodate a wide range of devices, including MPEG video devices, data gloves, and digitizers. This version of USB is known as full-speed USB.

Since October-1996, the Windows operating systems have been equipped with USB drivers or special software designed to work with specific I/O device types. USB got integrated into Windows 98 and later versions. Today, most new computers and peripheral devices are equipped with USB.

USB1.1: USB 1.1 came out in September 1998 to help rectify the adoption problems that occurred with earlier versions, mostly those relating to hubs.

USB 1.1 is also known as full-speed USB. This version is similar to the original release of USB; however, there are minor modifications for the hardware and the specifications. USB version 1.1 supported two speeds, a full speed mode of 12Mbits/s and a low speed mode of 1.5Mbits/s. The 1.5Mbits/s mode is slower and less susceptible to EMI, thus reducing the cost of ferrite beads and quality components.

USB2.0: Hewlett-Packard, Intel, LSI Corporation, Microsoft, NEC, and Philips jointly led the initiative to develop a higher data transfer rate than the 1.1 specifications. The USB 2.0 specification was released in April 2000 and was standardized at the end of 2001. This standardization of the new device-specification made backward compatibility possible, meaning it is also capable of supporting USB 1.0 and 1.1 devices and cables.

Supporting three speed modes (1.5, 12 and 480 megabits per second), USB 2.0 supports low-bandwidth devices such as keyboards and mice, as well as high-bandwidth ones like high-resolution Web-cams, scanners, printers and high-capacity storage systems.

USB 2.0, also known as hi-speed USB. This hi-speed USB is capable of supporting a transfer rate of up to 480 Mbps, compared to 12 Mbps of USB 1.1. That's about 40 times as fast! Wow!

USB3.0: USB 3.0 is the latest version of USB release. It is also called as Super-Speed USB having a data transfer rate of 4.8 Gbit/s (600 MB/s). That means it can deliver over 10x the speed of today's Hi-Speed USB connections.

The USB 3.0 specification was released by Intel and its partners in August 2008. Products using the 3.0 specifications are likely to arrive in 2009 or 2010. The technology targets fast PC sync-and-go transfer of applications, to meet the demands of Consumer Electronics and mobile segments focused on high-density digital content and media.

USB 3.0 is also a backward-compatible standard with the same plug and play and other capabilities of previous USB technologies. The technology draws from the same architecture of wired USB. In addition, the USB 3.0 specification will be optimized for low power and improved protocol efficiency.

A physical USB device may consist of several logical sub-devices that are referred to as device functions. A single device may provide several functions, for example, a web-cam (video device function) with a built-in microphone (audio device function). In short, the USB specification recognizes two kinds of peripherals: stand-alone (single function units, like a mouse) or compound devices like video camera with separate audio processor.
The logical channel connection host to peripheral-end is called pipes in USB. A USB device can have 16 pipes coming into the host controller and 16 going out of the controller.

The pipes are unidirectional. Each interface is associated with single device function and is formed by grouping endpoints.

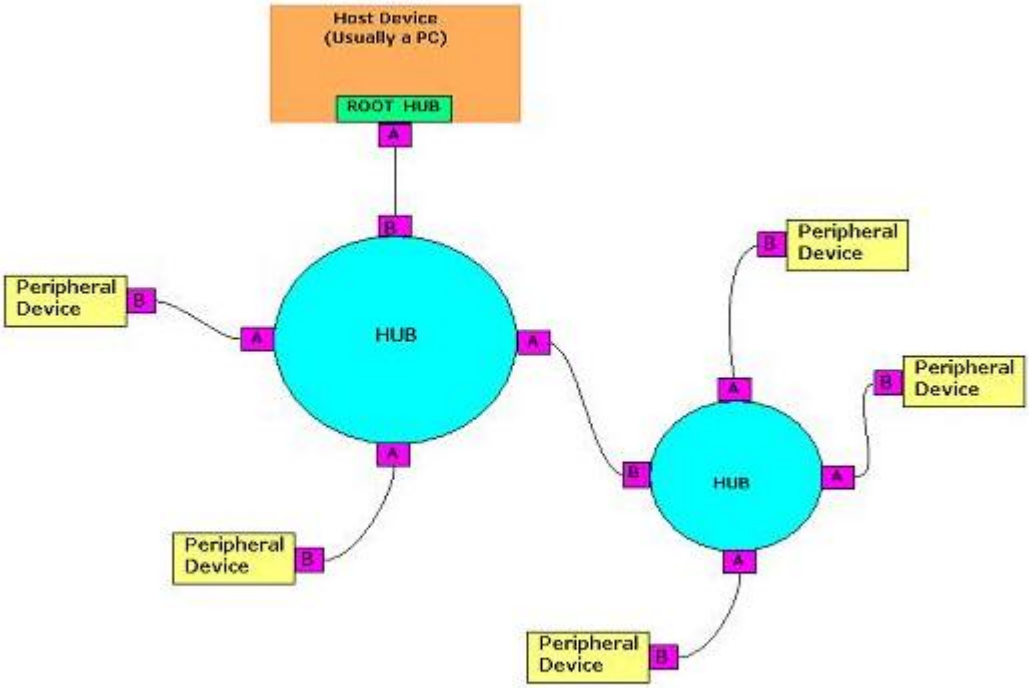


Fig2: The USB "tiered star" topology

USB connectors & the power supply:
Connecting a USB device to a computer is very simple -- you find the USB connector on the back of your machine and plug the USB connector into it. If it is a new device, the operating system auto-detects it and asks for the driver disk. If the device has already been installed, the computer activates it and starts talking to it.

The USB standard specifies two kinds of cables and connectors. The USB cable will usually have an "A" connector on one end and a "B" on the other. That means the USB devices will have an "A" connection on it. If not, then the device has a socket on it that accepts a USB "B" connector.

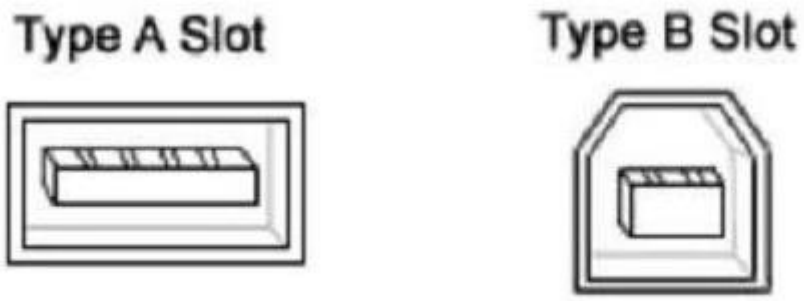


Fig 3: USB Type A & B Connectors

The USB standard uses "A" and "B" connectors mainly to avoid confusion:

1. "A" connectors head "upstream" toward the computer.
2. "B" connectors head "downstream" and connect to individual devices.

By using different connectors on the upstream and downstream end, it is impossible to install a cable incorrectly, because the two types are physically different.

Individual USB cables can run as long as 5 meters for 12Mbps connections and 3m for 1.5Mbps. With hubs, devices can be up to 30 meters (six cables' worth) away from the host. Here the high-speed cables for 12Mbps communication are better shielded than their less expensive 1.5Mbps counterparts. The USB 2.0 specification tells that the cable delay to be less than 5.2 ns per meter

Inside the USB cable there are two wires that supply the power to the peripherals-- +5 volts (red) and ground (brown)-- and a twisted pair (yellow and blue) of wires to carry the data. On the power wires, the computer can supply up to 500 milliamps of power at 5 volts. A peripheral that draws up to 100ma can extract all of its power from the bus wiring all of the time. If the device needs more than a half-amp, then it must have its own power supply. That means low-power devices such as mice can draw their power directly from the bus. High-power devices such as printers have their own power supplies and draw minimal power from the bus. Hubs can have their own power supplies to provide power to devices connected to the hub.

Pin No:	Signal	Color of the cable
1	+5V power	Red
2	- Data	White / Yellow
3	+Data	Green / Blue
4	Ground	Black/Brown

Table - 1: USB pin connections

\

Plug and Play

Plug and Play UPnP technology defines an architecture for pervasive peer to-peer network connectivity of intelligent appliances, wireless devices, and PCs of all form factors. It is designed to bring easy to use, flexible, standards based connectivity to ad-hoc or nmanaged networks whether in the home, in a small business, public spaces, or attached to the Internet.

UPnP technology provides a distributed, open networking architecture that leverages TCP/IP and the Web technologies to enable seamless proximity networking in addition to control and data transfer among networked devices.

To configure ISA adapters, including sound adapters, SCSI host adapters, multiport serial boards, and others that conform to the Plug and Play ISA Specification Version 1.0A, use the SCOadmin PnP Configuration Manager:

- immediately after you install the system, if you have any devices to configure
- immediately after you install an ISA PnP adapter
- any time you want to modify device configuration on a previously installed card

When you configure a adapter with the PnP Configuration Manager, you set the adapter's resources to match values compatible with existing UNIX ISA drivers.

In most cases, you should configure audio and network devices with their respective configuration managers; this ensures that the drivers and hardware are set to the same values. However, there are situations where you must use the PnP Configuration Manager for these devices:

- If there are other devices on the audio adapter (for example, IDE or SCSI ports), they are not recognized by the Audio Configuration Manager and must be configured with the PnP Configuration Manager.
- ISA Plug and Play network adapters are not recognized by the Network Configuration Manager. You can enable these devices using the PnP Configuration Manager, then configure them like regular ISA devices using the Network Configuration Manager.
- If the Audio Configuration Manager does not recognize a new Plug and Play audio device that is compatible with an older model, you can enable the device with the PnP Configuration Manager. Then configure its audio driver by selecting the compatible device listed in the Audio Configuration Manager, using the same values you used with the PnP manager.

To enable a Plug and Play device

1. Open Device Manager.
2. Right-click the device you want, and then click Enable.

   Enable will only be listed if the device is disabled.

You can also enable a device at the device's Properties page. At the bottom of the General tab, if Change Settings is present, click it. Then on the Driver tab, click Enable.

If you are prompted to restart the computer, the device is not enabled until the computer is restarted.

To disable a plug and play device

When you disable a device, the physical device stays connected to your computer, but the device driver is disabled. The drivers are available again when you enable the device. It can be useful to disable devices if you want to have more than one hardware configuration for your computer, or if you have a portable computer that you use at a docking station.

If you are prompted to restart the computer, the device will not be disabled and will continue to function until the computer is restarted.

After you disable a device and restart your computer (if necessary), the resources allocated to the device are free and can be allocated to another device.

To disable a Plug and Play device

1. Open Device Manager.
2. Right-click the specific device you want, and then click Disable.

The benefits of a pluggable system are

- extensibility: the application can be dynamically extended to include new features.
- parallel development: since features can be implemented as separate components, they can be developed in parallel by different teams.
- clear development direction: since the plugin framework ideally provides a well-defined interface and documentation for plugin writers, developers have a clear roadmap for development.
- simplicity: a plugin typically has one function, and so developers have a single focus

SCSI:

Small Computer System Interface (**SCSI**)is a set of standards for physically connecting and transferring data between computers and peripheral devices.

The SCSI standards define commands, protocols, electrical and optical interfaces.

SCSI is most commonly used for hard disk drives and tape drives, but it can connect a wide range of other devices, including scanners and CD drives, although not all controllers can handle all devices.

Recent physical versions of SCSI—Serial Attached SCSI (SAS), SCSI-over-Fibre Channel Protocol (FCP), and USB Attached SCSI (UAS)—break from the traditional parallel SCSI standards and perform data transfer via serial communications.

Although much of the SCSI documentation talks about the parallel interface, all modern development efforts use serial interfaces. Serial interfaces have a number of advantages over parallel SCSI, including higher data rates, simplified cabling, longer reach, and improved fault isolation.

The primary reason for the shift to serial interfaces is the clock skew issue of high speed parallel interfaces, which makes the faster variants of parallel SCSI susceptible to problems caused by cabling and termination.

SCSI Parallel Interface

Internal parallel SCSI cables are usually ribbons, with two or more 50–, 68–, or 80–pin connectors attached. External cables are typically shielded (but may not be), with 50– or 68–pin connectors at each end, depending upon the specific SCSI bus width supported. The 80–pin Single Connector Attachment (SCA) is typically used for hot-pluggable devices

Fibre Channel

Fibre Channel can be used to transport SCSI information units, as defined by the Fibre Channel Protocol for SCSI (FCP). These connections are hot-pluggable and are usually implemented with optical fiber.

iSCSI

iSCSI (Internet Small Computer System Interface) usually uses Ethernet connectors and cables as its physical transport, but can run over any physical transport capable of transporting IP.

USB Attached SCSI

USB Attached SCSI allows SCSI devices to use the Universal Serial Bus.

SCSI command protocol

In addition to many different hardware implementations, the SCSI standards also include an extensive set of command definitions. The SCSI command architecture was originally defined for parallel SCSI buses but has been carried forward with minimal change for use with iSCSI and serial SCSI. Other technologies which use the SCSI command set include the ATA Packet Interface, USB Mass Storage class and FireWire SBP-2.

In SCSI terminology, communication takes place between an initiator and a target. The initiator sends a command to the target, which then responds. SCSI commands are sent in a Command Descriptor Block (CDB). The CDB consists of a one byte operation code followed by five or more bytes containing command-specific parameters.

At the end of the command sequence, the target returns a status code byte, such as 00h for success, 02h for an error (called a Check Condition), or 08h for busy. When the target returns a Check Condition in response to a command, the initiator usually then issues a SCSI Request Sense command in order to obtain a key code qualifier (KCQ) from the target. The Check Condition and Request Sense sequence involves a special SCSI protocol called a Contingent Allegiance Condition.

There are 4 categories of SCSI commands: N (non-data), W (writing data from initiator to target), R (reading data), and B (bidirectional). There are about 60 different SCSI commands in total, with the most commonly used being:

- **Test unit ready:** Queries device to see if it is ready for data transfers (disk spun up, media loaded, etc.).
- **Inquiry:** Returns basic device information.
- **Request sense:** Returns any error codes from the previous command that returned an error status.
- **Send diagnostic and Receive diagnostic results:** runs a simple self-test, or a specialised test defined in a diagnostic page.
- **Start/Stop unit:** Spins disks up and down, or loads/unloads media (CD, tape, etc.).
- **Read capacity:** Returns storage capacity.
- **Format unit:** Prepares a storage medium for use. In a disk, a low level format will occur. Some tape drives will erase the tape in response to this command.
- **Read (four variants):** Reads data from a device.
- **Write (four variants):** Writes data to a device.
- **Log sense:** Returns current information from log pages.
- **Mode sense:** Returns current device parameters from mode pages.
- **Mode select**: Sets device parameters in a mode page.

The SCSI architectural model provides an abstract view of the way that SCSI devices communicate. It is intended to show how the different SCSI standards are inter-related. The main concepts and terminology of the SCSI architectural model are:

- Only the externally observable behavior is defined in SCSI standards.
- The relationship between SCSI devices is described by a client-server service-delivery model. The client is called a SCSI initiator and the server is called a SCSI target.
- A SCSI domain consists of at least one SCSI device, at least one SCSI target and at least one SCSI initiator interconnected by a service delivery subsystem.
- A SCSI device has one or more SCSI ports, and a SCSI port may have an optional SCSI port identifier (SCSI ID or PID).
- A SCSI device can have an optional SCSI device name which must be unique within the SCSI domain in which the SCSI device has SCSI ports. This is often called a World Wide Name. Note that the "world" may only consist of a very small number of SCSI devices.
- A SCSI target consists of one or more logical units (LUNs), which are identified by logical unit numbers.
- A LUN may have dependent LUNs embedded within it. This can recur up to a maximum nesting depth of four addressable levels.
- There are three type of SCSI ports: initiator ports, target ports and target/initiator ports. A SCSI device may contain any combination of initiator ports, target ports and target/initiator ports.
- SCSI distributed objects are considered to communicate in a three layer model:
    - The highest level of abstraction is the SCSI Application Layer (SAL) where an initiator and a target are considered to communicate using SCSI commands sent via the SCSI application protocol.
    - The SCSI Transport Protocol Layer (STPL) is where an initiator and a target are considered to communicate using a SCSI transport protocol.

- Examples of SCSI transport protocols are Fibre Channel, SSA, SAS, UAS, iSCSI and the SCSI Parallel Interface.
- The lowest level is the SCSI Interconnect Layer (SIL) where an initiator and a target are considered to communicate using an interconnect. It consists of the services, signaling mechanism and interconnect subsystem used for the physical transfer of data from an initiator to a target

- SCSI Architecture Model (SAM)