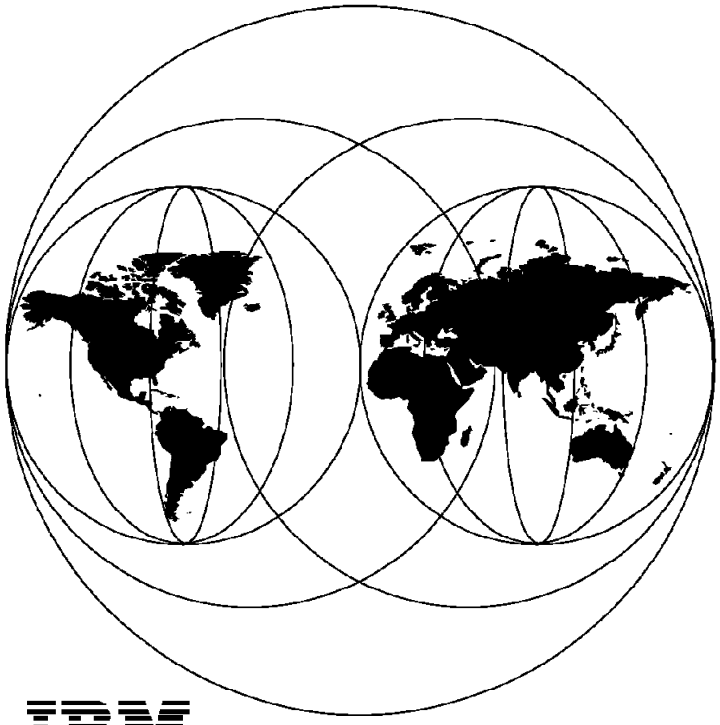
IBM Personal Computer
Disk Subsystem Considerations

SG24-2510-00

October 1995



IBM

International Technical Support Organization

Boca Raton Center

IBM Personal Computer
Disk Subsystem Considerations

SG24-2510-00

October 1995

Take Note!

Before using this information and the product it supports, be sure to read the general information under “Special Notices” on page xiii.

First Edition (October 1995)

This edition applies to IBM′s PC hardware and software products currently announced at the date of publication.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader′s feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. JLPC Building 14 Internal Zip 5220
1000 NW 51st Street
Boca Raton, Florida 33431-5220

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1995. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Abstract

This document provides detailed information on disk subsystem implementations in the IBM Personal Computer products and describes the differences between the technologies used.

This document is intended for IBM customers, dealers, system engineers, and consultants who need to know the capabilities and implications of the IBM Personal Computer disk subsystems.

This book assumes that the reader has a basic knowledge of personal computers and local area networks. The material presented in this book is not intended to give the reader an advanced skill level on the subject of PC disk subsystems.

(122 pages)

© Copyright IBM Corp. 1995

Contents

© Copyright IBM Corp. 1995

Figures

© Copyright IBM Corp. 1995

Tables

© Copyright IBM Corp. 1995

Special Notices

This publication is intended to help IBM customers, dealers, systems engineers and consultants to understand the disk subsystem technology used in IBM Personal Computer products. The information in this publication is not intended as the specification of any programming interfaces that are provided by OS/2, OS/2 LAN Server, NetWare, Windows NT or any other products mentioned in this publication. See the PUBLICATIONS section of the IBM Programming Announcement for these products for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (VENDOR) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

© Copyright IBM Corp. 1995

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM	OS/2
Personal System/2	PS/ValuePoint
PS/2	ThinkPad

The following terms are trademarks of other companies:

Windows is a trademark of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

C-bus is a trademark of Corollary, Inc.

Apple is a trademark of Apple Computer, Incorporated
Adaptec is a trademark of Adaptec Incorporated
Centronics is a trade mark of Centronics Data Computer Corporation
Compaq is a trademark of Compaq Computer Corporation
Microsoft is a trademark of Microsoft Corporation
NCR is a trademark of NCR Corporation
NetWare is a trademark of Novell, Incorporated
SAM is a trademark of Symantec Corporation
Seagate is a trademark of Seagate Technology, Incorporated
Western Digital is a trademark of Western Digital Corporation
Windows NT is a trademark of Microsoft Corporation
X/Open is a trademark of X/Open Company Limited

Other trademarks are trademarks of their respective companies.

Preface

This document is intended to provide an overview of disk subsystem technologies used in IBM Personal Computer products. It describes:

- Integrated Drive Electronics (IDE)
- Enhanced Integrated Drive Electronic (E-IDE)
- Small Computer System Interface (SCSI)
- Redundant Array of Independent Disks (RAID)

It also provides indepth information on Serial Storage Architecture (SSA), as well as a short overview of early disk subsystem implementations, ST506 and ESDI.

This document is intended to help IBM customers, dealers, system engineers, and consultants to understand the basic concepts and implications of a disk subsystem and how these basic concepts have been implemented in today's most popular disk subsystem technologies. Furthermore, it helps the reader to select the right implementation for a certain situation.

How This Document Is Organized

The document is organized as follows:

- Chapter 1, "Introduction"

  This chapter explains the vital role of a disk subsystem, provides a disk subsystem definition, details on the disk subsystem topology, information on the history of disk subsystem technology and an overview of the different disk subsystem implementations in current IBM PC products.

- Chapter 2, "Disk Subsystem Components"

  This chapter describes the components of a disk subsystem. Described are the system level interface and the hardware components (disk subsystem controller, direct access storage devices and the cabling).

- Chapter 3, "Disk Subsystem Implications"

  This chapter gives an overview of performance, availability and expandability implications of a disk subsystem.

- Chapter 4, "Device Level Disk Subsystems"

© Copyright IBM Corp. 1995

This chapter provides a description of the device-level based disk subsystems ST504 and the Enhanced Small Device Interface (ESDI).

- Chapter 5, “System Level Disk Subsystems”

  This chapter describes the system-level based disk subsystem technologies IDE, E-IDE and Small Computer System Interface (SCSI).

- Chapter 6, “Redundant Array of Independent Disks (RAID Technology)”

  This chapter explains the advantages of the different available Redundant Arrays of Independent Disks (RAID) definitions.

- Chapter 7, “Future Disk Subsystems”

  This chapter provides information on SCSI-3. Included in this chapter are descriptions of the Serial Storage Architecture (SSA), Fiber Channel (FC) and IEEE-1394 (also known as FireWire).

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- *Hardware Bible*, ISBN-N-1-56686-127-6
- *Introduction to Hard Disk Management*, ISBN-0-88022-897-0
- *The Book of SCSI*, ISBN-1-886411-02-6
- *The SCSI Bus and IDE Interface*, ISBN-0-201-42284-0
- *The RAID Book*, ISBN-1-879936-90-9

International Technical Support Organization Publications

- *IBM PC Server Disk Subsystem Configuration Examples,* SG24-4525
- *IBM Personal Computer 300 and 700 Series,* GG24-4496-00
- *ValuePoint Systems,* GG24-4298-00
- *IBM ThinkPad Systems,* GG24-4297-01

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

International Technical Support Organization Bibliography of Redbooks, GG24-3070.

To get a catalog of ITSO technical publications (known as “redbooks”), VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order ITSO Redbooks

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and MasterCard are accepted. Outside the USA, customers should contact their local IBM office.

Customers may order hardcopy ITSO books individually or in customized sets, called BOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain redbooks on a variety of products.

Acknowledgments

This project was designed and managed by:

Michael Koerner
International Technical Support Organization, Boca Raton Center

The authors of this document are:

Erwin Uit de Bos
IBM Netherlands

Michael Koerner
International Technical Support Organization, Boca Raton Center

This publication is the result of a residency conducted at the International Technical Support Organization, Boca Raton Center.

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Dan Bensimon
IBM Raleigh

John Dinwiddie
IBM Raleigh

Phil Horwitz
IBM Raleigh

Adge Hawes
IBM Havant, United Kingdom

David Laubscher
IBM Raleigh

Steven J. Livaccari
IBM Boca Raton

Jack McGuckian
IBM Havant, United Kingdom

Andy McNeill
IBM Raleigh

Tom Newsom
IBM Raleigh

Nelson Pesce
IBM Uruquay

Michael Ringel
IBM Germany

Howard Sykes
IBM Raleigh

Chapter 1. Introduction

Personal Computer systems are becoming increasingly more important to the day-to-day operation of both large and small businesses. Especially in server systems, the demand for fault-tolerant and high-performance disk subsystems is rapidly increasing. This book provides a detailed overview of disk subsystem technologies used in IBM PC systems. It explains the vital role of the disk subsystem for a server system and it takes a close look at general concepts of different disk subsystem technologies.

To know how the technology works is important, but knowing the effect on your business is even more important. So while explaining the most popular disk subsystem implementations, we will also look at the performance, availability and expandability of a disk subsystem.

To help you understand today's most popular disk subsystem implementations (E-IDE and SCSI), we will give a short overview of ST506 and ESDI. These two implementations were the most popular disk subsystems used for PCs during the 1980s. We will also make a comparison between the different disk subsystem implementations presented in this book, and present some guidelines for making the right choice of disk subsystem technology for particular environments.

Finally, we will describe Small Computer Systems Interface-3 (SCSI-3) and Serial Storage Architecture (SSA).

1.1 Disk Subsystem Definition

The disk subsystem provides permanent mass storage to computer systems and is responsible for maintaining the integrity of the data while it is stored on the mass storage devices.

1.2 The Importance of the Server Disk Subsystem

Important factors in a local area network (LAN) are performance, availability and expandability. For a LAN to be as productive as possible, excellent performance and availability is required. Expandability is also an important factor as you need to be able to expand the LAN when it reaches its maximum capacity.

© Copyright IBM Corp. 1995

LAN performance and availability are dependent on the performance and availability of all LAN components (client workstation, cabling, server system etc.). Every single component will have an impact on performance and availability. Lack of either performance or availability of a client workstation will, for example, only be recognized by the person working on that particular workstation. But lack of performance or availability of a server system will be recognized by everyone connected to that server system.

A server system is a functional unit that provides shared services to workstations over a network. Very often server systems are used as a file server to provide mass storage to client workstations. This mass storage is used very frequently by the clients. Usually, the amount of mass storage that is needed by the clients grows steadily and sometimes quite rapidly. The LAN will be faced with the need to expand in the mass storage area a couple of times. If you have chosen the wrong disk subsystem, you might have a hard time expanding the mass storage of your server in an easy and inexpensive way.

Conclusion: Server mass storage, provided by the server disk subsystem, is used by many people many times a day. This means that the server disk subsystem is mission-critical to your business and requires:

- Good performance
- High availability
- Adequate expandability

1.3 Disk Subsystem Topology

The disk subsystem is one of several peripheral interfaces within a computer. Examples of other peripheral interfaces are the keyboard, parallel/serial ports and networking subsystems.

Theoretically, each peripheral interface should have four layers to shield the rest of the system from details they should not have to know. An interface with these four levels is known as a *system level interface*.

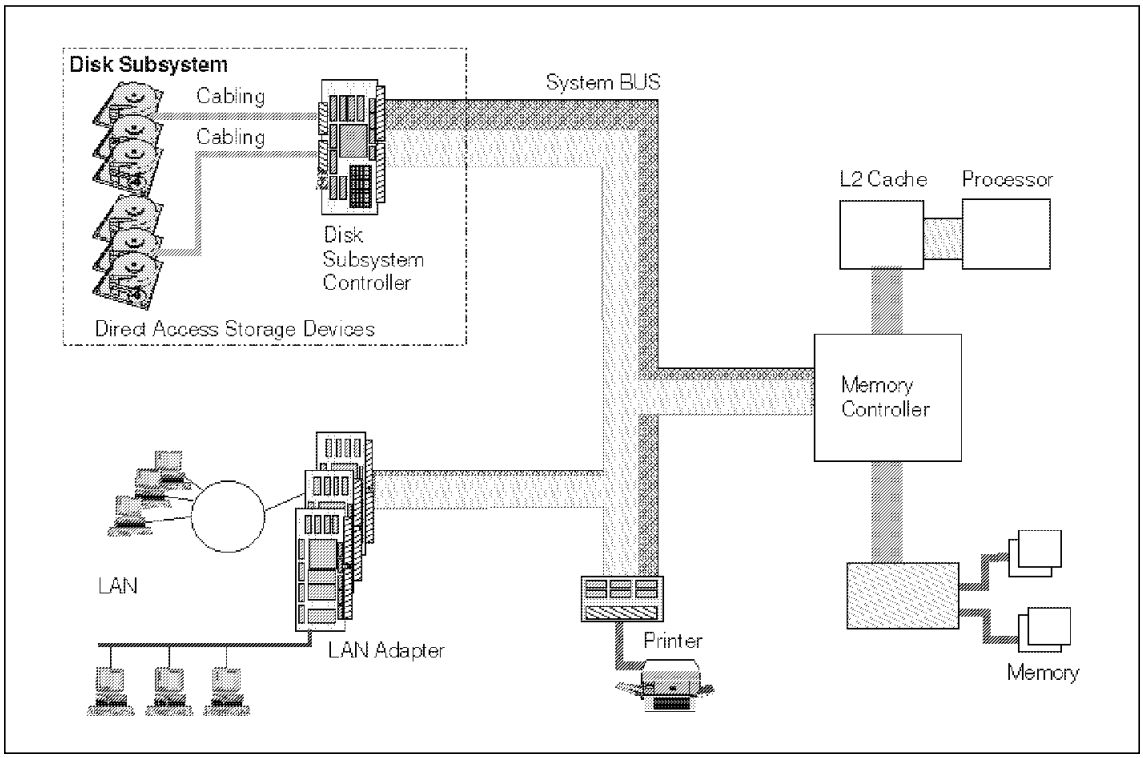


Figure 1. Disk Subsystem

Figure 1 displays an overview of the main components in a personal computer system, highlighting those parts that make up the disk subsystem. A typical disk subsystem consists of:

- Disk subsystem controller
- Direct access storage devices, such as hard disks and optical disks
- Cabling between storage devices and disk controller
- Software to configure and use the disk subsystem

The various technologies that are presented in this book have all implemented the disk subsystem interface differently into their components. These implementation differences are why these technologies have different figures for performance, availability, and expandability.

1.4 History

Since the introduction of the personal computer in the beginning of the 1980s, disk subsystem technology has gone through some significant changes. In the beginning only *device-level interfaces* were used. This type of interface does not implement the four layers of a system-level interface. Instead, it provides an interface that is very closely tied to the specific characteristics of certain devices. A device-level interface does not offer much flexibility because of this close bond between interface and device.

The two most popular device-level interfaces for hard disk subsystems are ST506 and ESDI described in Chapter 4, “Device Level Disk Subsystems” on page 29.

Disk devices grew rapidly in performance and capacity and became more reliable. Combined with the need for more flexibility, this led to the development of new disk subsystem technologies. From the origins of ST506 and ESDI, the IDE interface was developed which provided more performance and bigger capacities (see 5.1, “Integrated Drive Electronics (IDE and E-IDE)” on page 33). The need for more flexibility led to the development of the SCSI interface. SCSI provides interfaces for many peripherals as well as more performance and bigger capacities for disk devices (see 5.2, “Small Computer Systems Interface (SCSI)” on page 49).

1.5 IBM Disk Subsystems Overview

Table 1. Overview of Disk Subsystems Used in IBM PCs

Product	IDE	E-IDE	SCSI-2 Fast	SCSI-2 F/W	SCSI-2 F/W with RAID
PC 300 Vesa Local Bus	√	√	Opt		
PC 300 PCI	√	√	Opt	Opt	Opt
PC 700 IDE	√	√	Opt	Opt	Opt
PC 700 SCSI	√	√	√	Opt	Opt
PC Server 300 IDE	√	√	Opt	Opt	Opt
PC Server 300 SCSI	√		√	Opt	Opt
PC Server 320	√		Opt	√	Opt
PC Server 320 Array	√		Opt	Opt	√
PC Server 500			Opt	√	Opt
PC Server 500 Array			Opt	Opt	√
PC Server 720			Opt	√	Opt
PC Server 720 Array			Opt	Opt	√
ThinkPad	√	√	Opt		

Note: Opt means that the specified disk subsystem technology may be used in the system if an appropriate adapter is installed.

Chapter 2. Disk Subsystem Components

This chapter explains the functionality of the different components of a disk subsystem, which are:

- System-level interface
- Disk subsystem controller
- Direct access storage devices
- Cabling between the direct access storage devices and the disk subsystem controller
- Disk subsystem software

2.1 The System-Level Interface

The disk subsystem is one of the peripheral interfaces in a personal computer system. To get optimal performance, availability and expandability from a peripheral interface, the interface should be implemented with a system-level interface.

The advantage of using a system-level interface is that the complex details of the interface are hidden from the host system. This makes it much easier to implement the latest state-of-the-art technology and that ensures that the disk subsystem provides the most optimal performance, availability and expandability.

A system-level interface is divided into four layers. Table 2 shows these layers.

Table 2. System-Level Interface

Layer	Description
Command set layer	Lists all the commands that a device must be able to understand
Device model layer	Describes the general layout and behavior of types of devices
Protocol layer	Specifies how devices that are physically connected should communicate with each other
Physical interface layer	Describes the cabling, connectors and electrical characteristics

We will now look at the functionality of each of the four system-level interface layers and how they apply to a disk subsystem interface.

© Copyright IBM Corp. 1995

2.1.1 Physical Interface Layer

The physical interface layer is the lowest of the four layers and therefor the level that is the closest to the hardware implementation. It describes the physical and electrical characteristics of the components within the disk subsystem, such as:

- **Cable specifications**:
  - Physical characteristics of the cable
  - Number of lines required in the cable
  - Maximum length of a cable
- **Connector specifications**:
  - Maximum number of devices supported
  - Physical specifications for each connector
  - Pin layout for each connector, for example pin 1 through 16 for data, 17 through 25 for data control, etc.
- **Signal voltages**: for example, data pin 1 can have either 3.3 volts or no current.
- **Clock Speed**: defining the number of signals that can be transmitted in a certain time frame.

The physical interface layer does not define the physical and electrical characteristics of the interface between a disk subsystem and its host system. Those characteristics are specified in the technical documentation of the appropriate host system.

2.1.2 Protocol Layer

One layer above the physical interface layer is the protocol layer. The protocol layer defines how the disk subsystem components communicate with each other. It also describes the process used to ensure data integrity.

Following is a more detailed description of the responsibilities of the protocol layer:

- Signal interpretation. Defines how data and command signals are interpreted by the components. For example, a 3.3 volts current on a busy pin during a data transfer means that the drive is reading or writing data, while during the boot process of the system, 3.3 volts indicates that the device is performing its self-diagnostic tests.
- Message exchange procedure. Defines how messages are exchanged between the device and the disk subsystem controller. For example, if a device detects an error, an interrupt request signal will be sent out to the disk subsystem controller to ensure that the controller takes the appropriate actions.

- Error detection and/or error correction. Defines the method of determining whether data has been incorrectly stored on or retrieved from a device (for example, using a parity check), and the method for solving problems encountered during the transmission of data between the device and the host system.

The protocol layer does not define how the disk subsystem communicates with its host system. This procedure is specified in the technical documentation of the appropriate host system.

2.1.3 Device Model Layer

Above the protocol layer comes the device model layer. The device model layer contains one or more device models. Each device model describes the general structure and behavior of one type of devices in the peripheral interface.

Device types that have the same kind of structure and behavior use the same device model. For example, a re-writeable optical disk drive uses the same device model as a hard disk drive, but the device model of a printer will be totally different from the one used by the re-writeable optical disk and hard disk drive.

In most disk subsystems, only one device model is used to describe all direct access storage devices. The direct access storage device model describes at least the following device characteristics:

- Medium organization. Defines how data is structured on the device. Data can be structured, for example, in logical blocks or in physical blocks.
- Defect Management. Defines error handling and diagnostics management. For example, defective areas on a hard disk drive should be marked as no longer usable by the device.
- Performance. Specifies methods to achieve optimal performance. For example, a device can have a special caching mechanism to improve performance.

Many other characteristics, like power management facilities, can be specified in the device model as well, depending on the implementation of the device model layer.

Some disk subsystem implementations, such as SCSI and E-IDE, include multiple device models. This offers a peripheral interface that not only works with direct access storage device, but can work with other types of devices, such as printers and modems, as well. There are financial benefits to this, as there is no need to provide a separate peripheral interface for each type

of device. But the implementation needs a really good command set layer to use the full potential of each of the different device types.

The device model layer does not describe the general layout and behavior of device types to the host system. This is documented in the documentation for the appropriate host system.

2.1.4 Command Set Layer

The command set layer is the highest of the four system interface layers. It defines all the commands that a device of a certain type must be able to understand.

For a disk subsystem, the most important commands that need to be supported by the command set layer are as follows:

- **Read Data** from the device
- **Write Data** to the device
- **Query Device**
  - Geometry of the device (heads, tracks, cylinders, etc.)
  - Performance capabilities (buffers, caching methods, command streamlining, etc.)
  - Availability capabilities (error correction facilities, defect management, etc.)
- **Initialize Device**, for example, to format the device
- **Test Device**, for example, to run the diagnostics

The command set layer does not define the commands that the host system can use to communicate with the disk subsystem. Those commands are specified in the technical documentation of the appropriate host system.

2.2 Disk Subsystem Controller

The disk subsystem controller is always the first component of the disk subsystem encountered by the host system. All interaction between the host system and the disk subsystem goes through the disk subsystem controller.

Different hardware implementations of the disk subsystem assign different tasks to the disk subsystem controller, but generally, a controller has the following functionality:

- **Subsystem Manager**. The controller is usually the component that manages the whole disk subsystem. It acts as the focal point for all requests to the disk subsystem and controls the operation of all other subsystem components to fulfill these requests.

- **Host System Interface**. The controller takes care of translating the electrical signals between the format used by the host system and the format used by the disk subsystem. These signals are often different and often transmitted at different frequencies.

  To accommodate the difference in frequencies, the controller uses a small First-In-First-Out (FIFO) buffer. Sometimes the controller uses a larger buffer to apply caching techniques.

- **Error Detection and Correction**. The controller must be able to detect and correct errors in the data flow between a direct access storage device and the host system.

2.3 Direct Access Storage Devices

A storage device is able to store data and retrieve that data unchanged at a later time. The formal definition of a direct access storage device states that ″Direct Access Storage Devices (DASD) are devices in which access time to the data is effectively independent of the location of the data″.

Loosely translated into more practical terms, a direct access storage device is able to get to any data stored on the device without significant differences in access times and without the need for human intervention.

In a server disk subsystem, the function of direct access storage devices is to provide mass direct access storage to the client workstations on the LAN. Commonly used direct access storage devices are hard disks and optical disks.

2.3.1 Operational Overview

A direct access storage device is usually made up of:

- **Head Disk Assembly** (HDA). The HDA consists of:
  - A number of disks on which data is stored.
  - A motor to rotate the disks.
  - Heads to read and write data. The heads ″fly″ over the surface of a disk at a height of 2 to 3 micro inches (a human hair is 3000 micro inches).
  - An actuator to move the heads over the disks.
- **Printed Circuit Board** (PCB). The PCB consists of:
  - A Microprocessor Unit (MPU), to control the operation of the HDA and to communicate with the disk subsystem controller.
  - Servo circuitry, to position the actuator exactly.

- A read/write channel, to transform the electrical signals between a format that can be accepted by the host system and a format that can be accepted by the disk.
- A disk subsystem cable connector

Each disk is segmented radially into tracks. Each track is divided into sectors. A sector is the smallest addressable unit on a direct access storage device and usually contains 512 bytes of data. Figure 2 shows the layout of a disk. The disk has four tracks. Each track contains eight sectors.

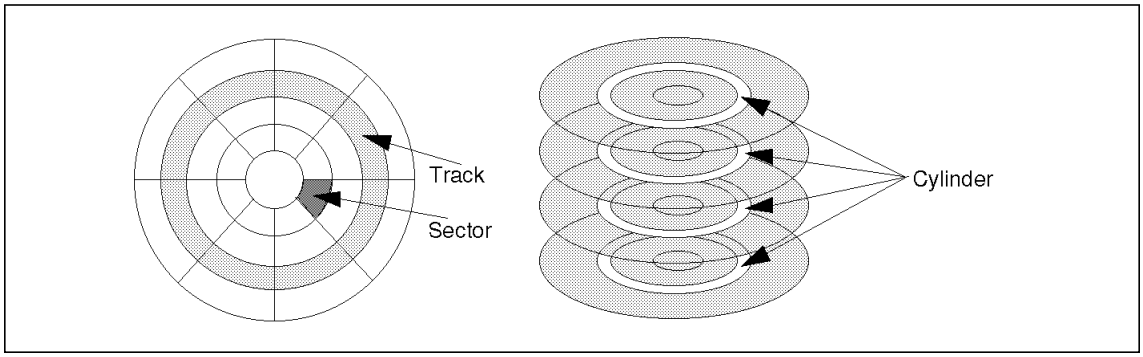


Figure 2. Example of a Disk Layout and a Cylinder

A cylinder is made up of all the tracks that are at the same location of each disk in the device, as shown in Figure 2. The figure shows a device that has four disks. The white rings in each disk indicate a specific track. All these white rings together make up a cylinder.

2.4 Cabling

The cabling is used to move data between the disk subsystem controller and the direct access storage devices.

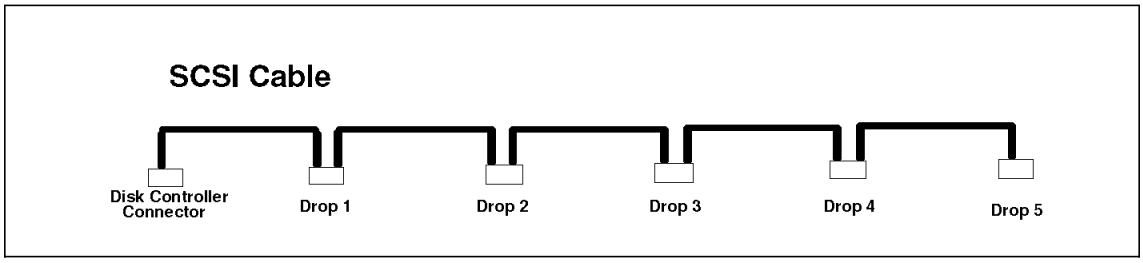


Figure 3. Example of a Disk Subsystem Cable (with 5 drops)

A disk subsystem cable usually has a controller connector and one or more device connectors. A device connector is often referred to as a *drop*; a five-drop cable has five device connectors (and one controller connector) as shown in Figure 3.

2.5 Software

The following software components are usually part of a disk subsystem:

- **Device Driver**. The operating system of a computer communicates with the disk subsystem through device driver software. The device driver is responsible for translating operating system commands into disk subsystem commands.
- **Configuration Utilities**. These are needed to set device specific parameters, such as block size, ID, priority, maximum data transfer rate, etc.

Chapter 3. Disk Subsystem Implications

This chapter describes the performance, availability and expandability implications of a disk subsystem on your personal computer.

3.1 Performance

The performance of the disk subsystem is determined by the time it takes the disk subsystem to complete a request to read or store data.

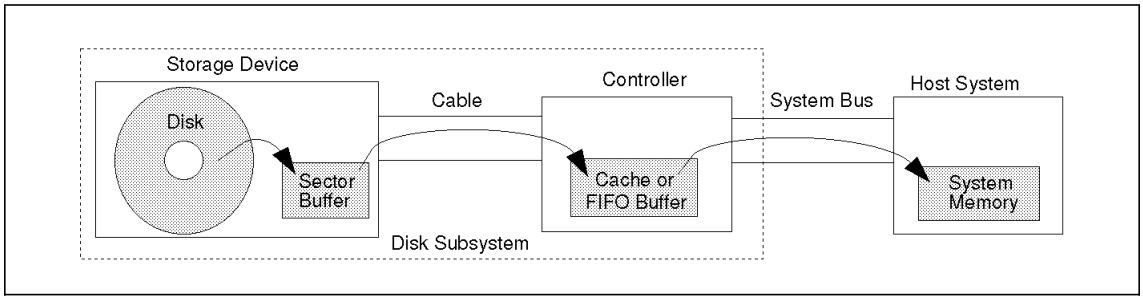


Figure 4. Overview of Data Transfer between Disk Subsystem and Host System

Figure 4 shows how data is usually moved between a disk subsystem and its host system. As you can see, each data transfer between the system memory and a direct access storage device takes three steps:

1. Moving the data between a direct access storage device and the sector buffer
2. Moving the data between the sector buffer and the controller cache or FIFO buffer
3. Moving the data between the controller cache or FIFO buffer and the system memory

Each step has its own performance characteristics, and the slowest step determines the overall performance of the disk subsystem.

Although it may seem that using three steps has a negative impact on the performance (it is usually quicker to do something in one single step), there are advantages to using a three-step procedure:

- **Frequency Independence**. Each of the steps can now be performed at its own maximum performance. For example, the maximum performance of a PCI system bus is 133 MBps while the maximum performance for

© Copyright IBM Corp. 1995

transferring data across an E-IDE cable is 20 MBps. If there were be no frequency independence then one part of the system would slow down the other parts.

- **Cost**. The system bus of the host system usually employs many more lines than are needed to control a disk subsystem. Using three steps means that a separate disk subsystem bus can be used that contains less lines, which makes the cabling less expensive.

When measuring the performance of a disk subsystem, the following terms are often used:

- **Average Access Time**. This is the average amount of time that it takes the hard disk to get to any location on the disk. It is the sum of the average seek time and the latency of the disk (seek time and latency will be explained later on in this section).
- **Data Transfer Rate**. This indicates how fast the disk subsystem can move data to or from the system memory of a computer. Usually, the rates given are *peak* rates that indicate the data transfer rate is at its maximum performance. *Sustained* transfer rates give a better indication of what can be expected from the disk subsystem under normal circumstances, because they indicate the performance that the disk subsystem can provide over a longer period of time.

It is important to understand that the data transfer rate does not define the performance of a disk subsystem. This is because the data transfer rate also depends on the capabilities and performance of other important system components, like the Central Processing Unit (CPU), the type of memory (low or high memory access time), or the bus architecture used by the host system. To correctly compare the data transfer rates of different disk subsystems, identical host systems must be used. For example, using a disk subsystem in a host system with a faster CPU will make the disk subsystem perform better.

The data transfer rate between the direct access storage device and the sector buffer on the disk subsystem controller is determined by the specifications of the system level interface and the implementation of the hardware components.

3.1.1 System-Level Interface

Some specifications of the system level interface for a disk subsystem have a big influence on the total performance of the disk subsystem. These specifications are usually written in stone and cannot be changed by a manufacturer:

- **Frequency**. The rate at which signals are sent through the disk subsystem cabling. Frequency is also referred to as *clock speed*. Measurements are usually in Mega Hertz (MHz). Frequency is one of the two items that determine the performance of step 2 on page 15.

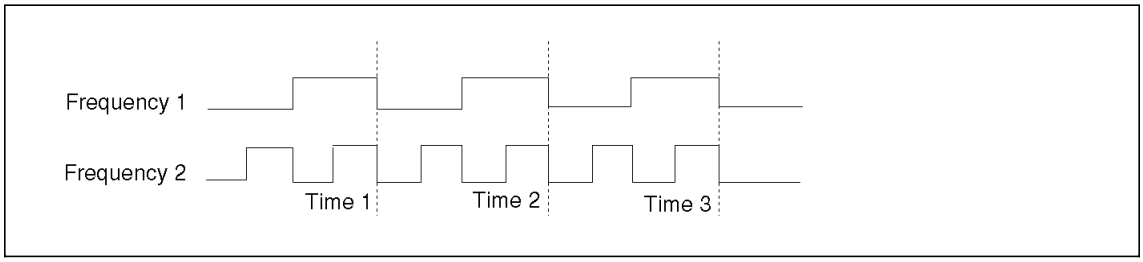


Figure 5. Impact on Performance by Different Frequencies

Figure 5 shows the effect of higher frequencies on the performance of the disk subsystem. Frequency 2 operates at a 50% higher frequency than frequency 1 and can therefore transmit 50% more data in the same amount of time.

- **Bandwidth**. The number of data bits that can be handled simultaneously by the disk subsystem. Bandwidth is one of the two items that determine the performance of step 2 on page 15.

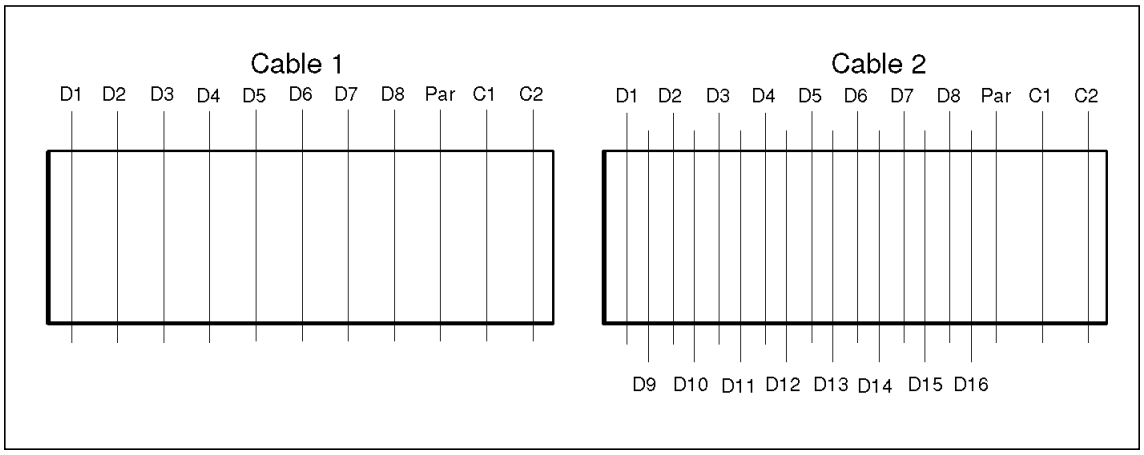


Figure 6. Impact on Performance by Different Bandwidths

Figure 6 shows two different cable layouts to demonstrate the effect bandwidth has on performance. Cable 1 has only 8 data lines so it can send 8 bits (1 byte) at the same time. Cable 2 has 16 data lines, so it can send 16 bits (2 bytes), doubling the performance of cable 1.

- **Protocol Efficiency**. The efficiency of the protocol layer can have a serious effect on the performance of a disk subsystem.

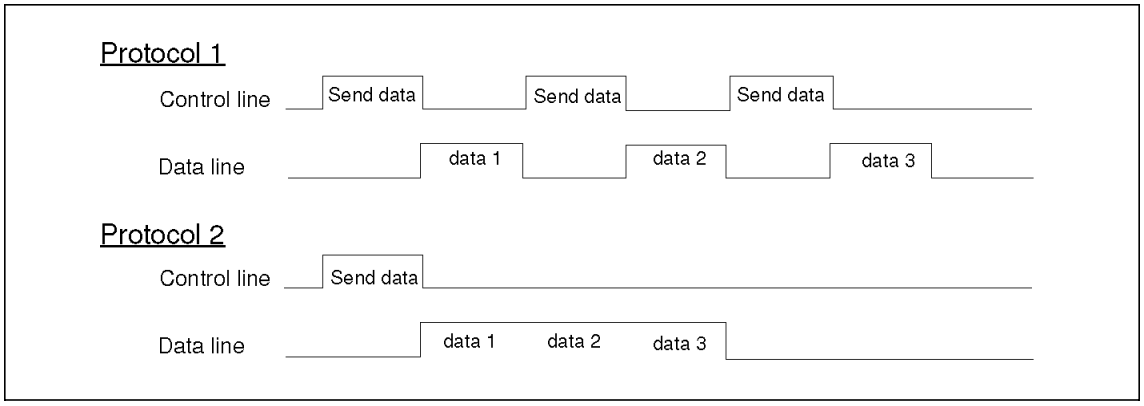


Figure 7. Impact on Performance by Different Protocols

Figure 7 shows the effect the protocol has on performance. With protocol 1, each time data is transmitted, a send data command has to be transmitted first. Protocol 2 only needs one send data command, and then all data items can be transmitted. The first protocol takes six cycles to transmit three data items whereas the second protocol only takes four cycles.

3.1.2 Hardware Components

The implementation of the hardware components of a disk subsystem can have a great effect on the performance. The hardware components are less bound to specifications, so this is where manufacturers can make a difference in their implementations.

Here is a list of performance related items and terms that are used by manufacturers to differentiate their products:

- **Seek Time**. The time required for the read/write head to be positioned on the correct track of the disk. Measurements are in milliseconds (ms). A faster seek time yields a better performance. Sometimes the term *track-to-track seek time* is used. It defines the time that it takes the device to move the read/write head from one track to another.

  Table 3 gives an overview of the evolution of seek times of some disk devices.

Table 3. Advancements in Average Seek Times for Storage Devices

Disk Device	Seek Time	Data Rate
IBM CD-ROM drive (1990)	380 ms	0.15 MBps
IBM Quad Speed CD-ROM drive (1995)	200 ms	0.59 MBps
Diskette drive (1994)	94 ms	0.061 MBps
IBM ImageStar 230MB Optical drive (1994)	40 ms	1.47 MBps
IBM PC-XT 10MB ST-506 hard disk (1982)	85 ms	0.625 MBps
IBM 120MB SCSI hard disk (1990)	23 ms	1.25 MBps
IBM 540MB IDE hard disk (1994)	12 ms	3.7 MBps
IBM Ultrastar 1.12GB SCSI hard disk (1994)	6.9 ms	7.4 MBps

- **Rotation Speed**. The faster a disk in a device rotates, the faster data can be moved to and from the disk. Measurements are in revolutions per minute (rpm). The rotation speed also determines the latency of the drive. The IBM UltraStar hard disk drives use a rotation speed of 7200 rpm.

  The maximum rotation speed of a device of a certain size is limited. For example, a 3.5″ device can run at about 7200 rpm at the most, before the reliability of the device becomes questionable. To get more rpm, a smaller device form factor is needed. A 2.5″ disk is able to have a much higher rpm.

- **Latency**. Latency is the time required for the right sector on a track to come under the read/write head. Latency is measured in ms. Usually, the figure for latency is the time it takes the drive to do half a rotation. A shorter latency is better for the performance. The latency for an IBM UltraStar hard disk drive is 4.17 ms.

  Latency is becoming more and more the factor that has the biggest influence on the performance. Today, seek times are almost at the same level as the latency. Latency occurs more often then seeks, so latency has a bigger influence on the total performance.

- **Media Transfer Rate** (MTR). Determines how fast a device can move data between the media and its sector buffer. The MTR depends on the rotation speed and on the density with which data is stored. If a sector takes up less space on the disk, it can be read faster. MTR is usually measured in Megabytes per second (MBps). The MTR for IBM UltraStar hard disk drives is 7.4 MBps.

  This rate is usually the bottleneck in the data transfer rate of todays disk subsystems. Compared to the speed of the system bus of the host system and the speed for transporting data across the cabling, this rate is the lowest.

- **Interleaving**. If a host system is not able to keep up with the pace of the disk subsystem, the data transfer rate from the disk subsystem needs to be decreased. Interleaving a disk will enforce this in the most efficient manner.

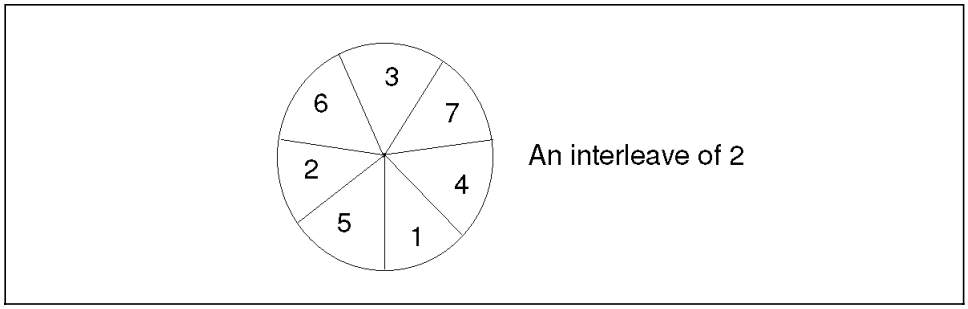


Figure 8. Example of Interleaving a Disk

As shown in Figure 8, the logical sector sequence differs from the physical sector sequence. The next logical sector on the disk is not the next physical sector. After reading or writing a sector, the disk subsystem will have to wait a few milliseconds before it can continue reading or writing the next logical sector. This should make it possible for the host system to catch up with the disk subsystem. Interleaving is measured in

the number of sectors that are inserted between two logically consecutive sectors.

Modern disk subsystems do not need to use interleaving anymore. Using larger sector buffers and double ported buffers, etc. has eliminated the need for interleaving.

- **Grouping large amounts of data**. Moving the head(s) of a device is often the most time consuming operation when reading or writing data. Table 4 lists the rules that should be followed to achieve minimum head movement in a device, to ensure optimal performance while reading or writing large amounts of data.

Table 4. Rules for Grouping Data on a Device

Data > Sector [1]	**Data > Track** [2]	**Data > Cylinder** [3]	**# of Disks**	**Grouping Rule**
no				Place data anywhere
yes	no			Place data in adjacent sectors of same track
yes	yes	no	1	Place data in adjacent tracks
yes	yes	no	multiple	Place data in the same track of different disks (no head movement is needed, just a head change)
yes	yes	yes	1	Place data in adjacent tracks
yes	yes	yes	multiple	Place data in adjacent cylinders

Note:

[1] Data > sector = data size larger than sector size.

[2] Data > sector = track size larger than track size.

[3] Data > sector = cylinder data size larger than cylinder size.

The grouping of large amounts of data is usually handled by the Operating System (OS). The disk subsystem does not know which sectors logically belong together, but the OS does. If the OS does not handle the grouping of data correctly, *fragmentation* of the data occurs. Fragmentation has a serious impact on the performance, since it causes unnecessary seeks and latency.

- **Track Skewing**. This only applies to devices that have multiple disks. When data that is larger than one track is read or written, a head change is required to continue reading or writing. Although, changing to another head can be done very quickly, the time required may be long enough to miss the first sector of the track on the next disk. If the first sector is

missed, the head will have to wait almost one complete rotation before it can continue. To avoid this situation a technique called track skewing can be implemented. See Figure 9 on page 22 for an example of a skewed disk.

Today's devices are fast enough to be able to use track skews of just one or two.

- **Cylinder Skewing**. When data that is larger than one cylinder is read or written, the read/write head(s) needs to be moved to an adjacent cylinder to continue reading or writing. Although moving to another cylinder can be done very quickly, it might be long enough to miss the first sector of the track on the next cylinder. If the first sector is missed, the head will have to wait almost one complete rotation before it can continue. To avoid this situation cylinder skewing can be implemented. See Figure 9 for an example of a skewed disk.

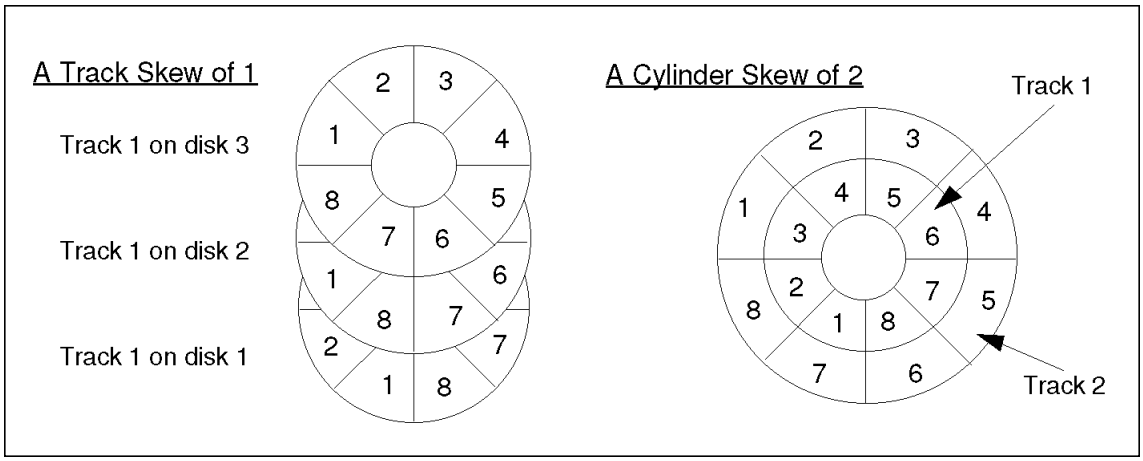


Figure 9. Example of Skewing a Disk

As shown in Figure 9, when track skewing is implemented, the first sector of the next disk is located at a further location on the disk. When sector 8 of track 1 on disk 1 has been read, the next sector to be read is sector 1 of track 1 on disk 2. In the example shown in the figure, the disk subsystem has as much time to perform the head change as it takes the disk to rotate 1 sector.

When cylinder skewing is used the first sector of the next cylinder is located at a further location on the disk. When sector 8 of track 1 has been read, the next sector to be read is sector 1 of track 2. In the example shown in the figure, the disk subsystem has as much time to move the head to track 2 as it takes the disk to rotate 2 sectors.

Today's devices are fast enough to be able to use cylinder skews of just 1 or 2.

- **Elevator seeking**. An elevator uses the most efficient way to fulfill all the requests to stop at certain floors. A device that uses elevator seeking is able to use the same elevator technique to sequence all read and write requests, giving the disk subsystem a better overall performance.
- **Double ported sector buffers**. Using double ported buffers increases the efficiency for moving data between the direct access storage device and the host system.

Table 5. Difference between a Single and a Double Ported Buffer

Single Ported Buffer	**Double Ported Buffer**
Device stores data in sector buffer	Device stores data in first sector buffer
Controller tells the host system it can transport the data from the sector buffer to the system memory	Controller tells the host system it can transport the data from the first sector buffer to the system memory
Device waits for the host system to completely transport the content of the buffer to the system memory	Device stores new data in second sector buffer
Host tells the controller that it has transported the data	Host tells the controller that it has transported the data (from the first sector buffer)
The device starts putting new data in the sector buffer	Controller tells the host system it can transport the data from the second sector buffer to the system memory

As you can see in Table 5, using a double ported buffer increases the performance of the disk subsystem significantly.

3.1.3 System Bus

The system bus of the host system also has an influence on the data transfer rate, because it defines the data transfer rate between the sector buffer and the system memory of the host (step 3 on page 15). If the performance of the system bus is low, the data transfer rate of your system will also be low.

More information on the performance of the system bus can be found in the technical information of your computer.

3.1.4 Software

The device driver (or the operating system) can really cripple the performance of the disk subsystem if it does not use the facilities provided by the hardware components or the system level interface efficiently.

3.2 Availability

Another very important requirement for a LAN is availability. Particularly the server disk subsystem plays an important role in the user′s perception of the availability of the LAN. For the end users at a client workstations, the LAN facilities, including mass storage, must be available for usage without any interrupts.

Availability is the ability of a functional unit to perform a required function under stated conditions for a stated period of time. Translated to a disk subsystem, availability defines the time that the data stored on the disk subsystem is available to be used by an end user.

Regarding availability, the following terms are often used:

- **Mean Time to Data Loss** (MTDL). MTDL defines the average time until data loss occurs. When data loss occurs, the lost data is not available to the end user anymore.
- **Mean Time Between Failures** (MTBF). MTBF is a term very often misinterpreted as a figure indicating the average lifetime of a component. This is not true. MTBF is the average time between consecutive failures of a component and measured in hours.
- **Fault Tolerance**. This is the ability of a system to continue operations even though one or more of its components are malfunctioning.
- **Redundancy**. This is the use of multiple components in the same system that are able to perform the same function. This is one of the most popular methods to achieve a certain degree of fault tolerance.

Statistics about failure rates of LAN components that cause loss of availability for the end user in a LAN environment, show that about 25% is caused by failures in direct access storage devices. Other causes include other failing hardware components, software errors, human errors, etc..

Optimal availability of a disk subsystem is influenced by several important factors that we will now investigate in more detail.

- **Frequency**. As a result of transmitting data at a very high frequency, intensive heat can develop in a cable. This heat needs to be cooled by the computer system, or the cable might melt, resulting in data loss. This has an effect on the cost of the system as well.

  High frequencies also increase the possibility of interference with other equipment outside of the computer system.

- **Grounding**. To minimize signal and environment interference, sufficient ground signals should be used in the cabling.

- **Error Detection and Correction Logic**. To prevent loss of data integrity, the disk subsystem should include error detection and correction capabilities. The device bus should for example include parity lines to check the validity of the data lines.

  A good example of this item is the *Predictive Failure Analysis* (PFA) feature on most of the IBM Hard disk drives. This feature detects when a failure of the drive is imminent and will then warn the user about this fact.

- **Defect List**. Information on defect disk areas of a device is stored in a defect list. The information will be used, for example, to by-pass the defect areas on a device by using available spare locations on the disk.

  A defect list is usually created during a low-level format of the drive, and it is expanded as new errors are found during the normal operation of the device. These errors are corrected (if possible) by using spare sectors on the device. Todays devices usually have one spare sector per track, and when a spare sector is needed, the nearest (in access time) free spare sector will be used.

- **Disk Subsystem Commands**. Commands should be implemented in the disk subsystem to enhance the availability of the disk subsystem, for example:

  - Diagnostics, for drive self-tests
  - Re-calibrate, to prevent misalignment of the read/write head, as this would result in loss of data integrity
  - Replace bad sector, as part of the defect list management
  - Show current status, to check the availability of the drive
  - Show error log, to investigate status on functionality of the drive

Correct implementation and use of these factors will help to increase the availability. However, the highest possible availability of a disk subsystem can be achieved by implementing additional techniques such as:

- **Redundant Components**. The existence of more than one, for a disk subsystem vital component (disk subsystem controller, storage device etc.) will increase availability significantly.
- **Hot Spare Device**. A hot spare device is a device that is installed in the disk subsystem defined for automatic use in the event of a device failure.
- **Hot Swap Device**. A hot swap device can be replaced while the system remains on line, without disruption to clients.

3.3 Expandability

The last part of this chapter deals with the expandability of the disk subsystem. Expandability of the disk subsystem is necessary when you need to increase the mass storage available on your computer system.

When looking at the expandability of your disk subsystem, you will need to pay attention to the following:

- **Standardization**. Standards usually describe the implementation of the four layers of the system level interface. Adding additional disk subsystem components that follow the standards of your existing disk subsystem should make it much easier for you to expand your server system. Having a disk subsystem that does not follow any standard will make upgrading your system a tough job or even impossible.
- **Maximum Number of Direct Access Storage Devices**. There is usually a limit to the number of devices that can be attached to a cable or that can be controlled by the disk subsystem controller. The higher the limit, the better the expandability. Note that if the disk subsystem implementation also allows other types of devices to be used, attaching these devices will take up spaces (bays) that could have been used by direct access storage devices.
- **Maximum Cable Length**. This impacts the maximum distance between a device and the disk controller. Especially if you need to attach external storage enclosures, the cable length can limit the expandability of your system.
- **Device Model Specifications**. The device model(s) enhances the expandability to all devices that are able to work according to the device model(s).
- **Command Set**. The command set enhances the expandability of the disk subsystem to all devices that are able to understand the command set layer.

- **Number of Disk Controllers**. Some disk subsystem implementations allow multiple disk controllers to be in the system at the same time. This significantly enhances the expandability, because you are able to use new disk controller technologies, more devices, etc.
- **Multiple Channels**. In some disk subsystem implementations, the number of devices that can be attached to a cable is less then the number of devices that the controller can handle. In this case, the controller could provide multiple channels, each with its own cable. A controller that has this feature offers more expandability. The IBM Streaming Raid controller and the IBM SCSI-2 Fast/Wide adapter/A provide two channels.
- **External Attachment**. A controller that is able to work with internal and external devices offers more expandability. The number of bays for internal devices is often limited. While most computer systems offer a maximum of seven bays, the IBM PC Server 500 and 720 offer 22 bays for internal devices.
- **Firmware Upgrade**. When the disk subsystem provides a facility for upgrading its firmware, it makes the disk subsystem more flexible. New firmware can increase the performance, availability and expandability. The IBM Streaming RAID controller and the IBM SCSI-2 Fast/Wide adapter/A have a facility to upgrade their firmware.

Chapter 4. Device Level Disk Subsystems

This chapter describes the two of the most popular device-level interfaces for hard disk subsystems, ST506 and ESDI.

4.1 Seagate Technology ST506

The Seagate Technology model ST506 was developed around 1980 and it was one of the first hard disks used in the Personal Computer. Because of its popularity, the interface for the ST506 was adopted by many other manufacturers as well, and they kept on using the name of the original drive as the name for the interface.

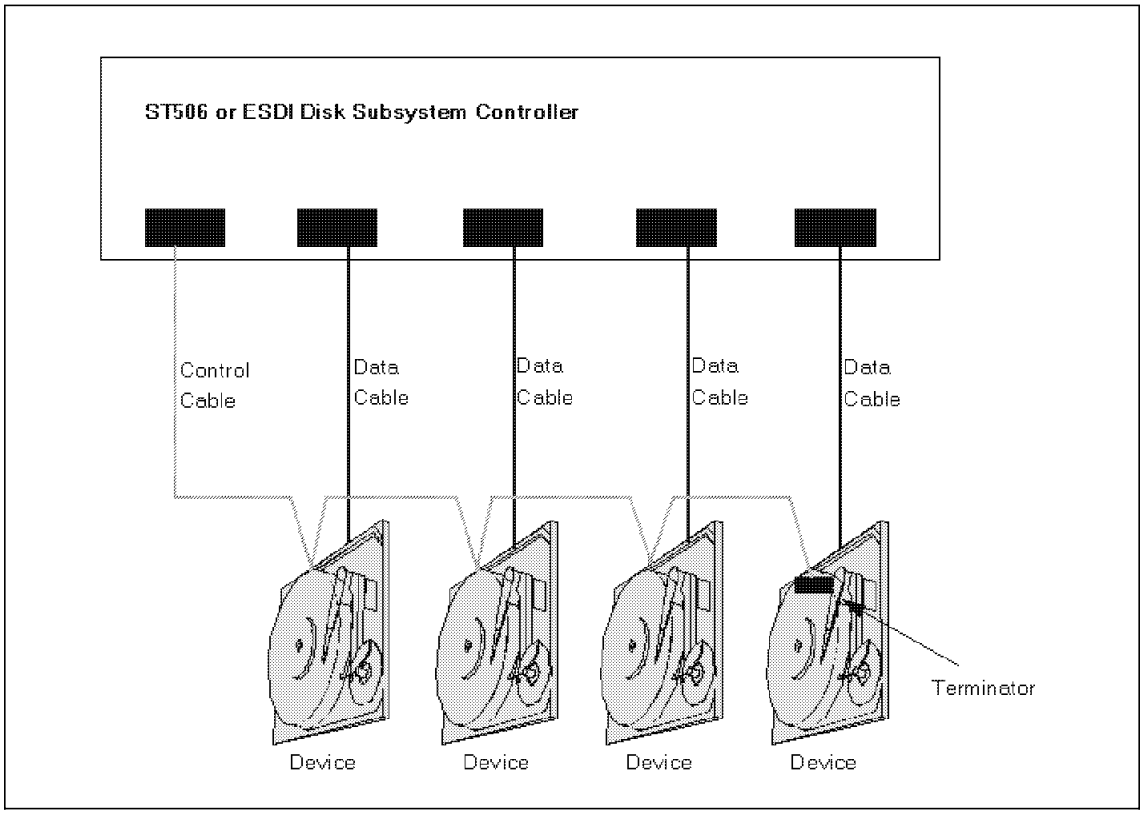


Figure 10. ST506 and ESDI Configuration Example

© Copyright IBM Corp. 1995

The physical interface provides connections for a maximum of four disk drives. Jumpers or switches on the drives are used to uniquely identify each drive. To pass control signals to each of the drives a single cable is used in a daisy-chain fashion. Each drive uses its own separate data cable. A terminating resistance is needed on the control cable after the last device in the chain.

ST506 is a serial interface, so data is send to the drives one bit at a time. This limits the performance of the interface, especially when used in faster PCs.

ST506 has only one device model, the hard disk, which is implemented in the controller. The specifications for the hard disk are either hard-coded into the controller or sometimes in non-volatile RAM. This limits the expandability, because only hard disks for which specifications are stored on the controller can be used.

The device model does not specify the way data should be encoded onto the rotating surfaces. Consequently, different adapters use different encoding techniques, with varying results. The main ones used are :

- Modified Frequency Modulation (MFM) encoding at 5 MHz (+/- 625 KB per second), 17 sectors per track, 512 bytes per track. This encoding scheme originates from the floppy disk drive. The first original IBM PC, was running at 4.77 MHz, which about matches the 5 MHz of MFM.
- Run Length Limited (RLL) encoding at 7.5 MHz. 50% improvement over MFM on available disk space and data transfer rate.
- Advanced Run Length Limited (ARLL) encoding at 9 MHz. 100% improvement over MFM on available disk space and data transfer rate.

The ST506 interface does not provide a protocol and command set layer.

4.2 Enhanced Small Device Interface

When faster single micro processors arrived, a new disk subsystem technology was required to be capable to run at higher speeds and to provide more flexibility. The Enhanced Small Device Interface (ESDI) had these capabilities. ESDI also has many similarities to ST506, making it easy to implement and to become a new standard for disk subsystems.

The physical interface uses the same cables and connectors as the ST506 interface (see Figure 10 on page 29). But the electrical characteristics of the signals are different. Because of this, you cannot mix ST506 controllers and devices with ESDI controllers and devices.

ESDI is, like ST506, a serial interface, but the encoding logic was moved from the controller to the drive. This allows for improved data transfer rate of maximum 3.125 MB/s and higher drive capacities.

Also important is the addition of *media defect management* as part of a basic protocol layer. This error checking and correction facility allows substitution of bad sectors with spare good sectors during a low-level format.

The device model for hard disks is nearly identical to the one used for the ST506 model. With ESDI, device specifications do not have to be stored on the controller anymore. This makes it much easier to connect different types of hard disks to an ESDI controller. Unfortunately, this dynamic configuration feature has not been used frequently by manufacturers.

ESDI also has a device model for tape-drives, but no ESDI tape devices have ever been marketed.

The addition of a basic protocol and basic command set layer provides ESDI with some more flexibility, enhancing its performance, availability and expandability.

Chapter 5. System Level Disk Subsystems

This chapter describes two of today's most popular implementations of disk subsystem technologies, IDE and SCSI.

5.1 Integrated Drive Electronics (IDE and E-IDE)

The IDE interface is a relatively inexpensive disk subsystem technology found in almost all of todays PC desktop and portable systems. The latest version of the IDE interface, Enhanced-IDE, will be able to fulfill the PC desktop and portable disk subsystem requirements for the next couple of years.

5.1.1 Introduction

The Integrated Drive Electronics (IDE) interface was developed around 1985. In order to save room in their portable computers, Compaq designed a disk subsystem that packaged the disk controller on the disk device. This package connected directly to the system bus on the systemboard, saving a precious adapter slot in the computer.

The design worked well and a very important side effect of the design was that it allowed great savings on the cost of the whole disk subsystem. Although including the controller on the disk raised the cost for the disk device, much more money was saved by not having a separate controller adapter.

Because of the savings in money and the fact that the IDE disk subsystem did not require an adapter slot on the system bus, many manufacturers adopted the design for portable computers as well as for desktop computers. This led of course, to incompatibilities between the IDE components of different manufacturers.

5.1.2 Standards

To resolve these incompatibilities, the Common Access Methods (CAM) committee started working on the AT Attachment (ATA) specifications in 1988. The name AT Attachment stems from the fact that it involved directly attaching a device to the system bus that was known at that time as the AT bus. The ATA specifications were approved by the American National Standards Institute (ANSI) in 1991.

© Copyright IBM Corp. 1995

Note: Throughout this book, we use the terms AT Attachment (ATA) and Integrated Drive Electronics (IDE) as synonyms. ATA-2 and Enhanced IDE are also used as synonyms.

5.1.3 Future

New device technologies and restrictions in the original IDE interface have led to the proposal of the ATA-2 standard. The ATA-2 proposal includes support for faster data transfer rates and larger device capacities, while maintaining compatibility with the original ATA specifications. Although the ATA-2 proposals have not yet been formalized, two variations on ATA-2 were already developed and brought to the marketplace:

- **Fast-ATA**. Developed by Seagate Technology. Fast-ATA adheres to the proposed ATA-2 specifications, providing better performance and larger capacities.
- **Enhanced-IDE** (E-IDE). Developed by Western Digital. E-IDE adheres to the proposed ATA-2 specifications but goes beyond those specifications as well. E-IDE supports up to four devices (instead of just two) and it also adds the AT Attachment Packet Interface (AT-API). Using AT-API, CD-ROM and tape devices are supported in E-IDE.

Work is already being done on the ATA-2 successor. ATA-3 will probably further enhance the performance of the IDE interface, but to achieve this the physical interface layer will have to be changed. To make the transition to ATA-3 somewhat easier to implement, the other ATA-3 layers will probably remain compatible with their ATA and ATA-2 counterparts. In more practical terms, this means that ATA-3 devices cannot be connected to ATA or ATA-2 devices, but you will be able to use the same software on the ATA-3 devices.

There is an ANSI working committee that constantly revises and expands the IDE specifications. The same ANSI committee also maintains the SCSI specifications.

5.1.4 Interface Overview

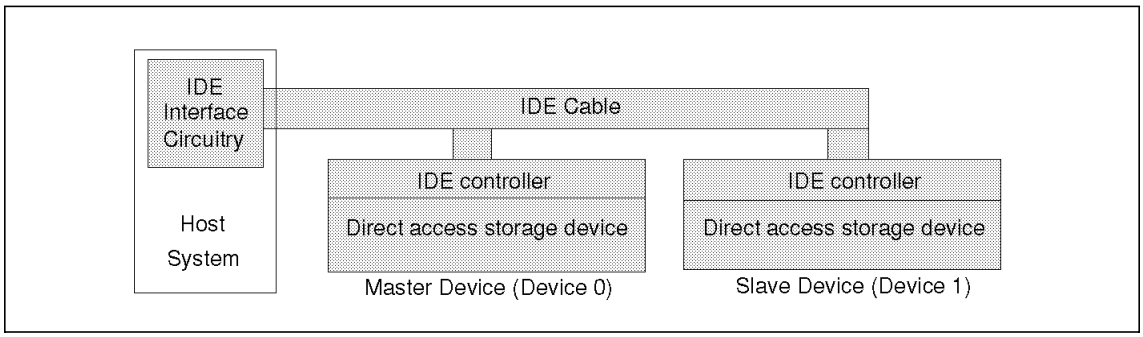


Figure 11. IDE Overview

Figure 11 shows a typical IDE configuration. The shaded areas in the figure are part of the IDE disk subsystem. As you can see, the IDE disk subsystem consists of:

- **IDE interface circuitry**, the interface between the IDE disk subsystem and the host system
- **IDE Devices**, consisting of direct access storage devices with dedicated controllers
- **IDE Cable**, used to transport data between the IDE devices and the host system

The IDE disk subsystem can be supported using the same software as the ST506 and ESDI interfaces. Most PC vendors included the ST506 and ESDI support software in the BIOS of the PC systems they sold, and they did not have to change anything to support IDE as well.

The IDE interface allows a maximum of two IDE devices to be connected. One device is called the master and the other is called the slave. These terms are not chosen very well, because they do not indicate the nature of the functionality of the devices. The only function of master and slave is to distinguish one device from the other. Each IDE device uses its own controller, independently from the other, so the slave device does not need the master device to function properly.

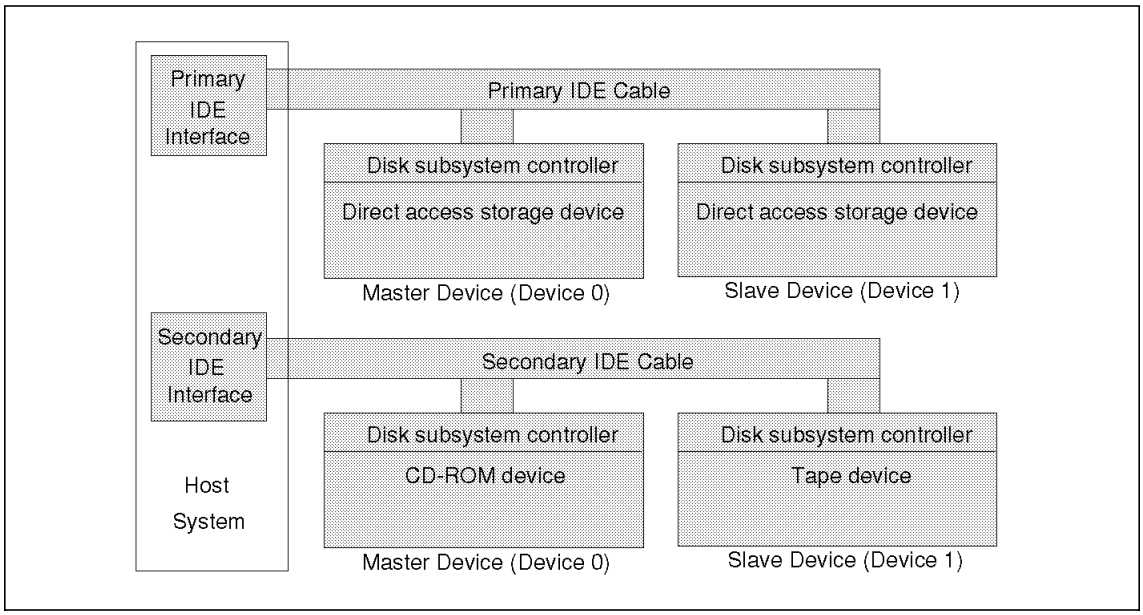


Figure 12. E-IDE Interface Overview

Figure 12 shows a typical E-IDE configuration. As you can see, E-IDE implements support for up to four devices, by duplicating a normal IDE configuration in the same host system. The figure also shows the two new device types (CD-ROM and tape drive) for which support has been added in E-IDE.

To fully exploit E-IDE, new software support is needed in the PC. Since IDE support was included in the system BIOS, it never required any special device driver software. Most vendors did the same with E-IDE and they included the new E-IDE support software in the system BIOS. In systems without E-IDE support in the system BIOS, special device drivers need to be loaded to support E-IDE.

E-IDE remains hardware and software compatible with the IDE interface, so components from both may be mixed together, although you will not be able to use all the E-IDE capabilities in systems that do not have E-IDE support.

5.1.5 System-Level Interface

The following sections explain how IDE implements the four layers of the system level interface. Unless stated otherwise, the E-IDE specifications are the same as the IDE specifications.

5.1.5.1 Physical Interface Layer

An IDE cable is a flat-ribbon cable with 40 lines. The maximum cable length supported is 18 inches (46 cm). Table 18 on page 91 shows the layout for the IDE cable and connector.

Because the IDE cable is directly connected to the system bus, the system bus has to run through the IDE cable. A system board has sufficient grounding to allow high frequencies in the system bus, but when a system bus runs through a cable, the grounding of the system board is lost. This makes it impossible to maintain a reliable connection over anything but a very short cable, especially if the cable is not shielded and grounded very well.

An IDE cable has one connector to attach to the IDE interface circuitry and two connectors to attach to IDE devices. All connectors on the cable have female plugs. All device and system board connectors have male plugs.

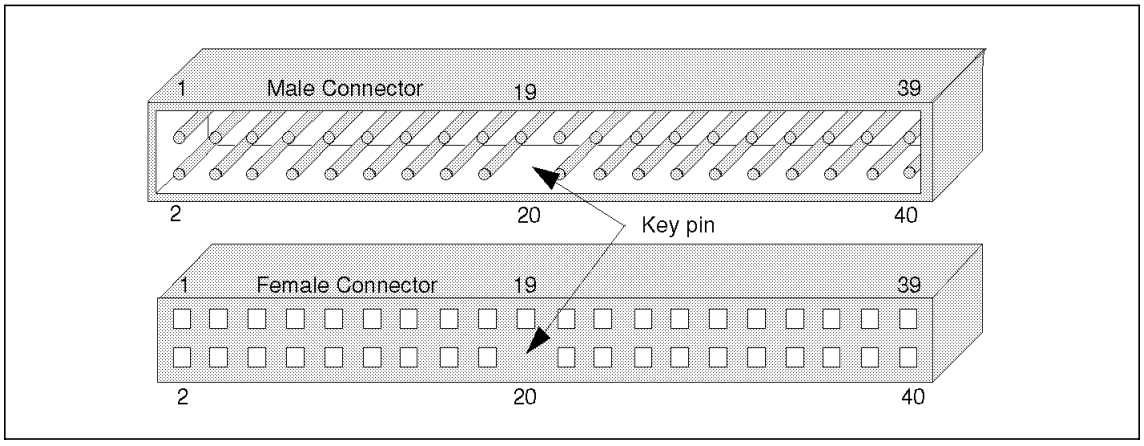


Figure 13. IDE Cable Connectors (with Key Pin)

Figure 13 shows the male and female IDE cable connectors. As you can see, each connector consists of two rows of 20 pins or pinholes. To prevent the cable from being inserted incorrectly, a *key pin* (pin 20) is used. The male part of the key pin is a pin that is not present at all, and the female part of the key pin does not have a pinhole.

The IDE interface is capable of working with either 5 volts or 3.3 volts of power. IDE Devices capable of running on 3.3 volts should also be able to work correctly on 5 volts. This is an important aspect of the IDE interface, since more and more PC systems are switching from 5 volts to 3.3 volts in order to reduce system power consumptions and temperatures.

IDE implements several frequencies to transfer data between the IDE devices and the IDE interface circuitry. These are described in the section on IDE performance (see Table 8 on page 45).

5.1.5.2 Protocol Layer

The IDE protocol layer defines the communication and error detection procedures between the host system and the IDE disk subsystem.

The following are two different methods for transferring data between an ID disk subsystem and its host system:

Programmed Input/Output (PIO)
: This is the usual method for transferring data between a direct access storage device, and the host system. The name originates from the fact that every sector of data that needs to be transferred has to be individually programmed by the host system CPU.

Direct Memory Access (DMA)
: This is the other method for transferring data. This method uses a DMA channel of the host system to transfer data between the IDE device and the host system.

The IDE interface defines five classes of commands. Each command class has its own protocol, but they are a lot alike. The following list shows the general protocol:

1. Host system sets up a command in the IDE controller
2. Host tells device to execute command
3. IDE device executes the command
4. IDE interrupts the host system to signal command completion
5. Host checks result of command

A more detailed description of the IDE protocols is given in A.2, “IDE Protocol Overview” on page 94.

A special protocol is used during power-up or hardware reset to determine the number of drives connected to the cable. If there is only one drive connected, then that drive will be the master drive. If there are two drives connected, then the master/slave relationship between the drives will have to

be determined. This is usually done by the end-user, who must correctly set jumpers or switches on the devices to determine the relationship.

The error detection and correction facilities of the IDE disk subsystem are discussed in 5.1.8, “Availability” on page 47.

5.1.5.3 Device Model Layer

The IDE device model layer contains only the direct access storage device device model. E-IDE adds device models for CD-ROM and Tape Drive devices.

An IDE direct access storage device is organized using the *Cylinder Head Sector* (CHS) addressing scheme. This addressing method was also used in the ST506 and ESDI disk subsystems, and uses the physical characteristics of the device to uniquely address each sector.

- **Head** identifies the disk in the device
- **Cylinder** identifies the track on the disk
- **Sector** identifies the sector within the track

Since many manufacturers included BIOS support for the ST506 and ESDI addressing scheme into their computers, no extra support was needed to be able to address IDE devices as well.

Table 6 shows the maximum values for the addressing parameters of the BIOS and the IDE specifications. As you can see in the table, the maximum values are not the same for IDE and BIOS. This is the cause for the 528MB IDE device capacity limit. Systems using BIOS support to address IDE devices need to restrict their addressing to those values which are supported by both BIOS and IDE. So to determine the maximum capacity for an IDE device, the highest common value for each parameter should be used.

Table 6. BIOS and IDE Addressing Capacities

Addressing scheme	Sectors	Heads	Cylinder	Capacity
BIOS	63	256	1024	8.5GB
IDE	255	16	65536	137GB
Highest common value	63	16	1024	528MB

In the 1990s, the maximum capacity of 528 MB became a real problem. To get around this problem, two solutions appeared in devices:

- **eXtended CHS** (XCHS). This solution translates a valid BIOS call into a valid IDE call. The translation is done by the device controller and makes

it possible to use the maximum BIOS values for heads, cylinders and sectors can be used.

- **Logical Block Addressing** (LBA). LBA presents all the available sectors on the device as a contiguous list of blocks. Each block is addressed by the host system using its own logical block number. This method requires extra support in the BIOS of the host system.

  LBA uses the following formula to translate a logical address into a physical address:

  ```
  LBA = ((cylinder x heads_per_cylinder + heads) x sectors_per_track) + sector - 1
  ```

  This formula has been chosen so that two adjacent logical blocks translates into two adjacent tracks in a sector. This ensures the best performance for large amounts of data.

When one of these two translation modes is used, zone-bit recording techniques (see 5.2.5.3, “Device Model Layer” on page 56) may be applied to the devices, allowing much better use of the disk space in a device.

IDE does not specify rules for defect management. It is up to the manufacturer to deal with defects on the surface of a disk. Generally, two approaches are used to deal with defects:

- During a format of the drive, bad sectors will be marked, so they won’t be used during normal read/write operations.
- Bad sectors are re-allocated to spare sectors. This can only be done if logical block addressing is used.

The IDE device model also specifies power management features for the device. The Power Management Feature Set (PMFS) permits an IDE controller card to modify the behavior of a device in a manner which reduces the power required by the device to operate. PMFS consists of:

- A number of power saving modes (see Table 23 on page 104)
- Commands to set a device into a power saving mode (see Table 20 on page 98)
- The IDE Standby Timer, to let a device enter a certain power saving mode when the timer runs out (see Table 24 on page 104)

5.1.5.4 Command Set Layer

The IDE command set layer specifies mandatory commands and optional commands. Manufacturers are also allowed to implement their own commands. The IDE command set layer specifies the commands that the host system may use to communicate with the IDE disk subsystem.

The mandatory IDE commands deal with:

- Initialization of the device (format)
- Testing the drive (diagnostics)
- Reading data from the device
- Writing data to the device

The optional IDE commands deal with:

- Power management
- DMA transfers
- Device parameters
- Media change management

A complete list and explanation of the IDE Command set can be found in A.4, “E-IDE Command Set Description” on page 98.

5.1.6 Hardware Components

The three hardware components of the IDE interface are the IDE adapter, the IDE cabling and the IDE device. Each is described in the following sections.

5.1.6.1 IDE Interface Circuitry

The IDE interface circuitry is usually an integrated part on the motherboard of the host system, although some manufacturers market adapters with this interface circuitry. Sometimes, the circuitry is a part of a complex I/O chipset, which also handles other peripheral interfaces, like the keyboard, floppy and printers.

The reason for the existence of the IDE interface circuitry is found in a trade-off that was made during the design of the IDE disk subsystem. Two main goals for the IDE design were to save an adapter slot on the system bus and to use a forty line cable to attach the disk subsystem to the host system. To save an adapter slot, the IDE cable had to be directly connected to the system bus. But there are a few problems with directly connecting a disk subsystem cable to a system bus:

- A system bus is not usually designed to have a cable attached to it.
- A system bus usually has many more lines than necessary to control a disk subsystem, and certainly more than the forty lines available in the IDE cable
- The frequencies for the disk subsystem and the system bus have to be matched to each other, slowing one of them down.

To solve these problems, the IDE designers implemented some interface circuitry and a 40 pin cable connector on the system board of the host system. The main functions of the IDE interface circuitry are:

- Adapting the data flow between the Disk subsystem and the host system to the electrical specifications of each. The IDE interface is usually equipped with a buffer to accommodate the different frequencies used by the system bus and the IDE subsystem.
- Passing on commands from the host system to the disk subsystem. The register blocks on the IDE device are mapped into the I/O address space of the host system. The host system writes commands to the I/O addresses that represent the IDE device controller registers. The IDE interface circuitry is responsible for picking up the commands from the I/O space and sending them across the IDE cable to the IDE device controllers.

5.1.6.2 IDE Device

An IDE device implements an IDE controller and a direct access storage device in the same hardware package. Each IDE controller is responsible only for the DASD on which the controller is located. This allows for a simple (and thus inexpensive) controller implementation.

Each IDE controller is equipped with two register blocks containing a total of eleven. These registers are used to communicate with the host system and to exchange data between the IDE disk subsystem and the host system. Table 7 lists all the IDE registers and gives a short overview of their functions:

Table 7. Overview of IDE Registers

Data	Used to move data between the host system and the disk subsystem
Error	Contains the results of the last executed command
Features	Configure the mode of operation of the device
Sector Count	• Number of sectors involved in a data transfer operation • Number of sectors per track during a format
Sector Number	Sector address of first sector involved in a data transfer
Cylinder Low	Least significant 8 bits of the cylinder address of the first sector
Cylinder High	Most significant 8 bits of the cylinder address of the first sector
Drive/Head	Device and head address of first sector
Status	Report the status of the drive (reading register clears pending interrupts)
Command	Specifies the command that must be executed by the device
Alternate Status	Report the status of the drive (reading register does not clear pending interrupts)

Disk subsystem commands are sent to the controller registers through the IDE cable. When two drives are attached to the IDE cable, both drives will receive all commands from the host system. The value in the Drive/Head register on the IDE controllers designates the drive that should execute the command.

The implementation of an IDE direct access storage device is completely up to the manufacturer. As long as the controller on the device is able to understand the IDE command set layer, the device can be implemented in any way.

Currently, IDE direct access storage devices employing LBA or XCHS techniques are able to provide capacities of 1.6 GB.

5.1.6.3 Cabling

If only one device is attached to the IDE cable, it is recommended to attach that device to the device connector at the end of the cable.

The IDE cabling has already been described in 5.1.5.1, “Physical Interface Layer” on page 37.

5.1.7 Performance

The data transfer mode that is used by the IDE disk subsystems depends on the IDE devices that are attached to the IDE cable. The mode chosen is always the best performing mode that is supported by both devices. This means that when you attach a brand new PIO mode 4 capable IDE device to the same cable as your old Mode 0 device, the new device will not work any better than Mode 0.

This is especially important in E-IDE systems. In these systems there are two IDE cables available, so make sure that all fast devices are attached to the primary cable, while the slower devices are attached to the secondary cable. Note that you will probably have to rearrange the master and slave settings on the devices for this setup. Check the specifications of your IDE devices to see which transfer modes they support.

Table 8 lists the data transfer modes defined in the E-IDE standard.

Table 8. E-IDE Data Transfer Modes

Mode	Clock speed (ns)	DTR (MBps)	Comments
PIO Mode 0	600	3,33	Standard IDE data transfer mode
PIO Mode 1	383	5,22	Not used often
PIO Mode 2	240	8,33	Not used often
PIO Mode 3 (ATA-2)	180	11,11	Standard E-IDE data transfer mode
PIO Mode 4 (ATA-2)	120	16,66	Some recent devices are able to use this mode
PIO Mode 5 (ATA-2)	100	20	Proposed transfer mode
DMA Single-Word Mode 0	960	2,08	Not used often
DMA Single-Word Mode 1	480	4,16	Not used often
DMA Single-Word Mode 2	240	8,33	Defined in ATA-2, not used often
DMA Multi-Word Mode 0	480	4,16	Not used often
DMA Multi-Word Mode 1	150	13,33	Defined by ATA-2, Standard E-IDE data transfer mode
DMA Multi-Word Mode 2	120	16,66	Defined by ATA-2

Table 9 on page 46 lists the performance implications of PIO and DMA data transfer modes. In today's PCs, the performance of the host system CPU is so good, that in a single-tasking environment like DOS, it is no problem at all

for the processor to program the disk subsystem for every single sector that needs to be transferred. The difference in performance between a PIO and DMA transfer in this situation is very small.

In a heavily loaded multiprocess environment like the server environment, off-loading of processing to an intelligent disk controller frees CPU cycles for other tasks as for example handling more LAN attached client systems.

Table 9. Performance Considerations for PIO and DMA

Item	**PIO**	**DMA**	**Best performance**
Initialize Data Transfer	PIO initialize is quicker	DMA needs additional time to setup the DMA channel	PIO
Multi-tasking aspects	PIO needs the CPU to program every single sector transfer	DMA needs the CPU only once to initialize all transfers	DMA
Very small amounts of data		DMA channel setup overhead	PIO
Larger amounts of data	Large overhead, because PIO needs CPU to program every single sector transfer		DMA

In an IDE device the performance of the device is so that on the average, the time to get to any sector on the same track is shorter then the time it takes for the IDE subsystem to switch to another read/write head. The time to switch to another head is shorter than the the time that it takes the device to move the heads to another cylinder.

5.1.8 Availability

IDE employs two error detection and correction facilities:

- Data is stored on a disk using Error Correction Code (ECC) bytes. During a read of the data, the ECC bytes are used to check the integrity of the data.
- Physical disk addresses are verified by Cyclic Redundancy Code (CRC) bytes stored as a part of each sector. During a read of the data, the CRC bytes are used to verify if the correct data is read.

One line in the IDE cable is used during data transfers to signal to the host that the data being transferred has passed ECC tests and that the address of the sector was also correct. There are no lines in the IDE cable for parity or other error correction facilities, so while data is transported through the IDE cable, it is not checked or protected.

The IDE command set contains two mandatory commands and three optional commands that, when used properly by the operating system, can enhance the reliability of the disk subsystem:

- Mandatory commands:
  - Recalibrate, to make sure that the device stays within track boundaries
  - Read Verify sectors, to check if a sector of data can be read from the medium without errors
- Optional commands:
  - Read Drive Status, to determine if the drive is functioning correctly
  - Set Features, to configure availability features of the drive, such as error detection and correction facilities
  - Write Verify, to ensure that data that has just been stored on the medium can also be read without errors.

See A.4, “E-IDE Command Set Description” on page 98 for a more detailed description of each of these commands.

The ANSI ATA committee has recommended using shorter IDE cables when using the E-IDE transfer modes. The interference and heat that occurs in these transfer modes are likely to cause more failures when the maximum allowed length is used.

5.1.9 Expandability

The ATA standard has been ANSI approved for some years already. Although ATA-2 has not been officially approved by ANSI yet, no major changes are expected. This makes ATA and ATA-2 stable standards, so you should have no problems connecting new devices.

When mixing IDE components with E-IDE components, you will probably not be able to use all E-IDE features.

IDE allows a maximum of two devices to be attached, E-IDE allows a maximum of four devices to be attached.

The maximum IDE cable length limits the usage of IDE devices to system unit internal use only.

The IDE device model allows direct access storage device to be attached with a maximum capacity of 528 MB. Using the IDE maximum of two devices yields a maximum IDE capacity of 1056 MB. E-IDE allows a maximum capacity of 8.5 GB for each of the four devices it supports.

IDE allows propriety commands. Although these commands may enhance performance, availability and expandability of your disk subsystem, using these commands often forces you to buy all IDE components from the same vendor in order to use these propriety features.

IDE supports only one IDE adapter to which two IDE controllers may be attached. E-IDE supports four E-IDE controllers on the E-IDE adapter. The E-IDE adapter uses two channels, a primary and a secondary E-IDE channel.

Recent host systems allow BIOS upgrades to be applied. These BIOS upgrades are able to fix or enhance the IDE or E-IDE functionality.

5.2 Small Computer Systems Interface (SCSI)

SCSI is a very general and multi-purpose expansion bus. SCSI allows attachment of many different peripheral types and is used in many different computer system platforms. In the PC world, SCSI is primarily used in PC server systems, because SCSI has always allowed bigger device capacities and offered better performance, but is more expensive than other disk subsystems.

5.2.1 Introduction

SCSI was first developed around 1980 by disk drive manufacturer Shugart. Shugart wanted to create a device independent parallel interface in which data would be addressed logically. The logical data addressing and device independence would make it much easier to introduce new device technologies, while using a parallel interface would increase the performance and expandability, opening up the interface to devices that communicate in a parallel manner.

In the beginning of the 1980s, support for SCSI by other manufacturers grew slowly. Vendors saw the performance and expandability benefits of the SCSI interface, but because of its flexibility and many vendor options, SCSI was much more complicated to implement. From a user′s perspective, SCSI devices were not easy to install and configure, especially when the devices came from different vendors.

Around 1990, it was much easier to manufacture and use SCSI products. The benefits of SCSI then caused the PC industry to adopt SCSI and make it the standard disk subsystem implementation for servers and power user workstations.

5.2.2 Standards

When SCSI first appeared, the name being used was Shugart Associates Systems Interface (SASI). Shugart, and later on NCR, wanted other manufacturers to adopt their standard, so the SASI specifications were made public and presented to ANSI. Around 1982, an ANSI committee began working on what would become the SCSI interface. In 1986, the first SCSI release was finally approved.

To address the deficiencies in the first SCSI standard, a new SCSI standard (SCSI-2) was proposed in 1989 and approved in 1994 by ANSI. SCSI-2 allows for better performance and ensures better compatibility between devices. The most important additions provided by SCSI-2 are:

- Common Command Set (CCS). The CCS contains specifications for the software controlling the SCSI subsystem, making the SCSI devices more compatible with each other.
- Fast SCSI. Allows higher transfer rates to improve the performance.
- Wide SCSI. Allows a bigger bandwidth to improve the performance.
- Mandatory bus parity checking. Improves the availability by providing more error detection facilities.

A lot will happen with the SCSI interface before it is approved in its third incarnation. Chapter 7, “Future Disk Subsystems” on page 85 describes the work that is being done on enhancing the SCSI interface.

5.2.3 Interface Overview

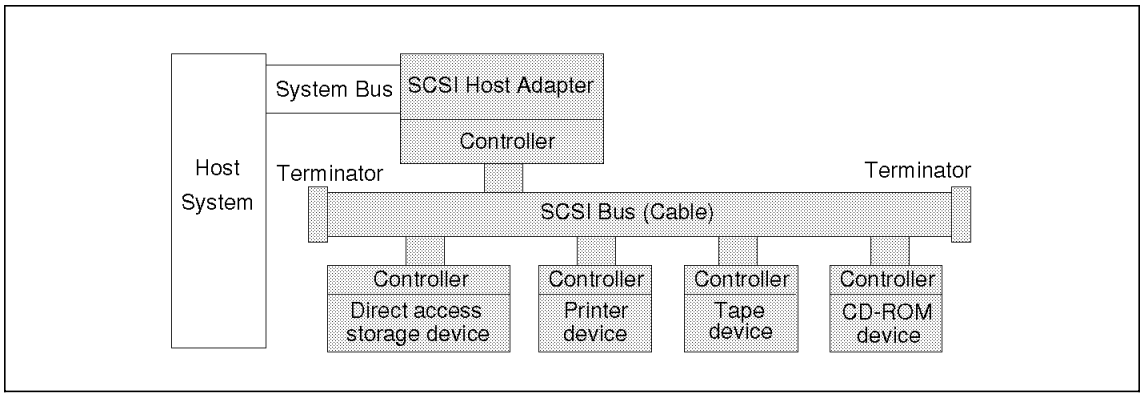


Figure 14. SCSI Interface Overview

Figure 14 shows an overview of a typical SCSI subsystem. A SCSI subsystem consists of a SCSI bus, terminated at each end, and a number of SCSI devices, all attached to the SCSI bus through dedicated controllers.

The SCSI subsystem uses a *host adapter* as the interface between the SCSI subsystem and its host system. This host adapter is usually a real adapter that plugs into one of the adapter slots of the host system, but it is sometimes implemented directly onto the system board of the host system.

Depending on the SCSI cable that is used, a total of eight or sixteen SCSI devices may be connected to a single SCSI bus. At least one of these devices must be a SCSI Host adapter. That leaves room for seven or fifteen other SCSI devices to be attached to the bus. A SCSI ID uniquely identifies each device.

5.2.4 SCSI Modes

SCSI defines many different modes of operation, including several different data transfer modes. Table 10 specifies these different SCSI modes.

Table 10. Different SCSI Modes

Mode	Bandwidth	Frequency	Max Data Transfer Rate	Max Cable Length Single Ended
SCSI-1	8-bit	5 MHz	5 MBps	6 meters
SCSI-2	8-bit	5 MHz	5 MBps	6 meters
Fast SCSI	8-bit	10 MHz	10 MBps	3 meters
Wide SCSI	16-bit	5 MHz	10 MBps	6 meters
Fast & Wide SCSI	16-bit	10 MHz	20 MBps	3 meters
32-bit SCSI	32-bit	5 MHz	20 MBps	6 meters
32-bit Fast SCSI	32-bit	10 MHz	40 MBps	3 meters
Fast-20 SCSI	8-bit	20 MHz	20 MBps	1.5 meters
Fast-20 Wide SCSI	16-bit	20 MHz	40 MBps	1.5 meters
Note: Fast-20 SCSI is also referred to as **Ultra SCSI** single ended				

5.2.5 System-Level Interface

The next sections describe the SCSI implementation of the system-level interface.

5.2.5.1 Physical Interface Layer

There are two types of electrical signaling defined in the SCSI standard:

- **Single-ended**, which allows a maximum cable length of 6 meters
- **Differential**, which allows a maximum cable length of 25 meters

It is not possible to mix these two types together. The PC world normally uses single-ended SCSI only, although some niche products are marketed that use differential SCSI. These products are usually taken from a non-PC environment into the PC world. An example of this is the IBM 3514 disk subsystem, which is also used in the IBM RS/6000 line of products. In this book we will only describe single-ended SCSI.

The PC world has seen a lot of official and unofficial SCSI cables. Table 11 lists the most commonly used cables. Internal cables are normally flat ribbon cables, external cables are Shielded Twisted Pair (STP).

Table 11. Common SCSI Cables

Cable Name	Type	# of lines	Connector Type	SCSI Mode
A Cable	I	50	IDC	8-bit
B Cable [1]	I + E	68	High Density	32-bit SCSI
P Cable	I + E	68 [2]	High Density	Wide SCSI
Q Cable [3]	I + E	68	High Density	32-bit SCSI
L Cable	I + E	110 [4]	High Density	32-bit SCSI
A Cable	E	50	Centronics	8-bit SCSI
SCSI-2	E	50	High Density	8-bit SCSI
IBM PS/2	E	60	High Density	8-bit SCSI

Note: Type I = internal cable, Type E = external cable

[1] Must be combined with the SCSI-2 cable. Will be removed in SCSI-3, has not been implemented much

[2] A simple adapter may be used to attach 50 pin devices

[3] The Q cable is only used in combination with the P cable to provide support for 32 bit SCSI

[4] A simple adapter may be used to attach 50 or 68 pin devices

The B cable will be removed from the SCSI standard. It has not been used very much. The centronics style external 50 pin cable was used quite often, but most new SCSI host adapters support the more convenient and smaller SCSI-2 cable.

Figure 15 shows some types of SCSI cable connectors. Note that these connectors are in scale with each other. The 68-pin High Density connector is really smaller than the other two connectors.

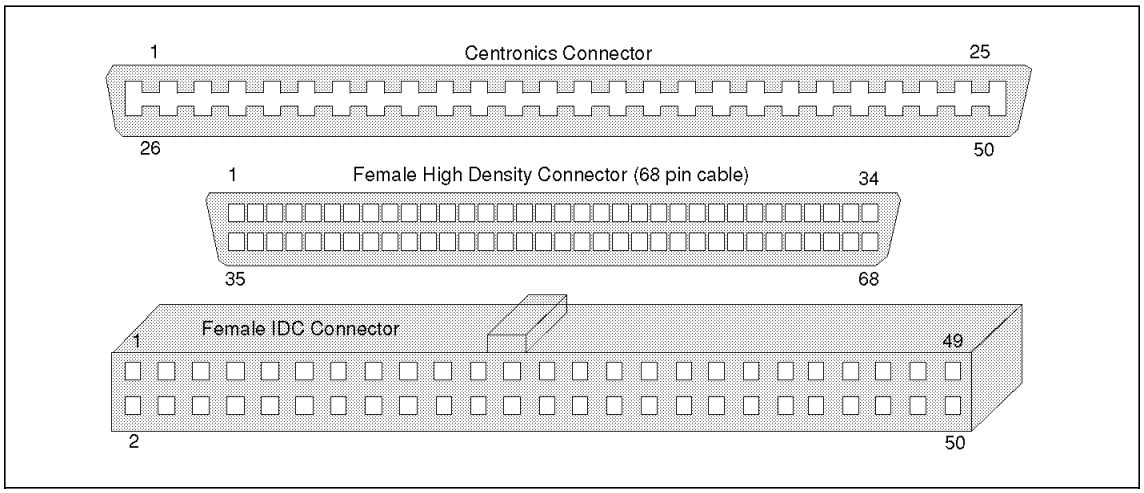


Figure 15. SCSI Cable Connectors

An internal cable normally has a connector for the host adapter and seven device drops, although there are also lots of internal cables which have less drops. The cable connector on the host adapter usually looks like a connector for a system bus. The cable connector for the host adapter looks like a system bus slot.

An external cable usually has two connectors. Some external devices have only one SCSI connector. Such a connector is called a *single-port tailgate*. These devices require a special cable connector that allows connection of a SCSI device and of another SCSI cable.

Figure 16 shows an internal and external cable connected to a SCSI Host adapter and to an internal and external SCSI device. Note the connection of the Terminator on the external device. It is connected to the top SCSI connector on the device. The terminator should be connected to the SCSI-OUT connector of the device. On some devices that may be the bottom connector.

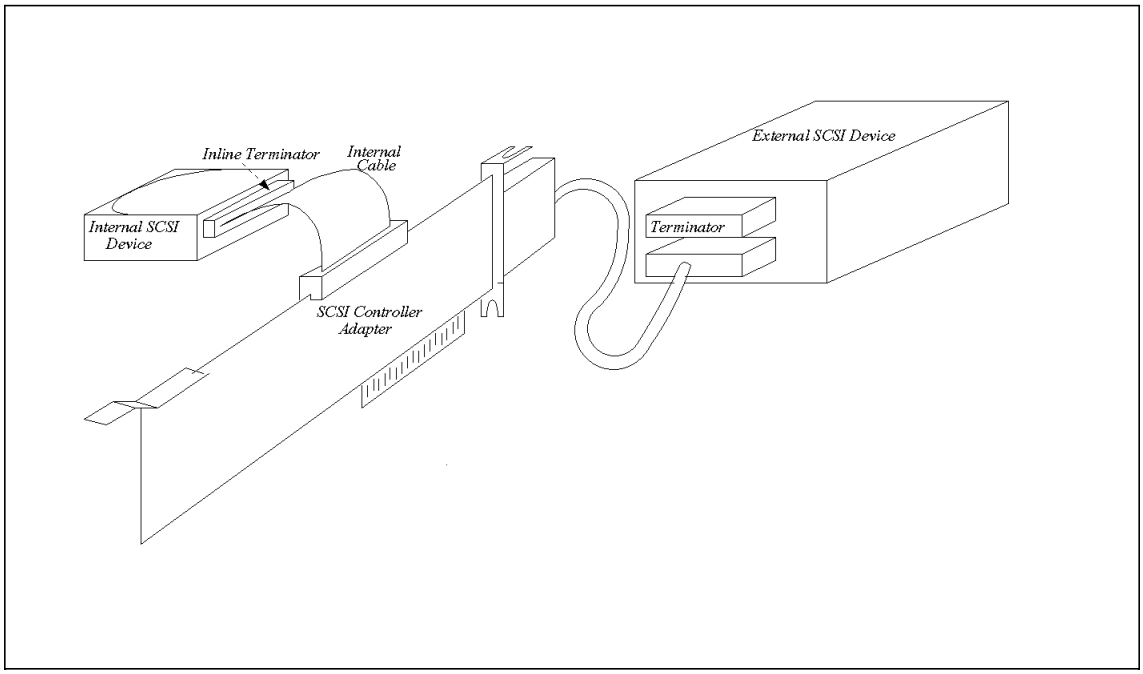


Figure 16. SCSI Internal and External Cable Connectors

There are quite a few different SCSI connectors. Table 12 explains the gender of device and cable connectors.

Table 12. SCSI Connector Gender

Connector	**Gender**
Connector on an internal cable	Female
Connector on an external cable	Male
Connector on an internal device	Male
Connector on an external device	Female

Each connector may use a stub of 10 cm from the main SCSI cable, to allow for easier connection of the connector. These stubs must be at least 30 cm apart from each other.

5.2.5.2 Protocol Layer

To communicate through the SCSI bus, a device (the *initiator*) should initiate a connection to the component that it wishes to communicate with (the *target*). When the connection has been established, the initiator may communicate with the target.

The following general protocol is used to setup a connection and then communicate on the SCSI bus:

Table 13. General SCSI Protocol

Step	Bus Phase	Description
1	**Bus Free**	Initiator waits until SCSI bus is free.
2	**Arbitration**	All devices that want to communicate, signal this by putting their SCSI ID on the bus. The component with the highest ID wins.
3	**Selection**	The winning initiator selects its target by putting its own SCSI ID and that of its target on the SCSI bus. The target acknowledges, and takes over the control of the SCSI bus.
4	**Message**	The initiator and target determine the protocol and communications mode they will use to communicate. Options that are determined at this stage are: • Asynchronous or synchronous transfers • Frequency to be used during the transfer (Fast SCSI) • Bandwidth to be used (Wide SCSI)
5	**Command**	The target tells the initiator to transmit the SCSI commands it needs to process.
6	**Data**	Data is exchanged between initiator and target.
7	**Status**	The initiator and target determine the success or failure of the exchange.
8	**Bus Free**	The SCSI bus returns to the bus free phase

In time consuming operations, involving complex or long running commands, it is possible to temporarily free the bus so other devices may use it. For example, when the host system wants to read data from a tape device, the SCSI bus will be freed while the tape drive spools the tape to the desired location.

Some SCSI devices are able to use *Tagged Command Queues*. A queue can contain up to 255 commands. There are several different ways to insert a

command into the queue, which allows the initiator to influence the sequence of the queue.

Error handling in the SCSI disk subsystem is discussed in 5.2.8, “Availability” on page 66.

5.2.5.3 Device Model Layer

SCSI supports the following device types:

- Direct Access devices
- Sequential Access devices
- Printer devices
- Processor devices
- Write-Once devices
- CD-ROM devices
- Scanner devices
- Optical memory devices
- Medium-Changer devices
- Communications devices

In this section we will only explain the DASD device model. We will first explain some terms that are used when talking about a SCSI direct access storage device.

Zone-bit recording
: Allows the device to better use the available disk space, by adding more sectors to the outer tracks of a device.

Extend
: A continuous sequence of blocks which all have the same sector size.

Notch
: A continuous sequence of tracks which all have the same amount of sectors per track.

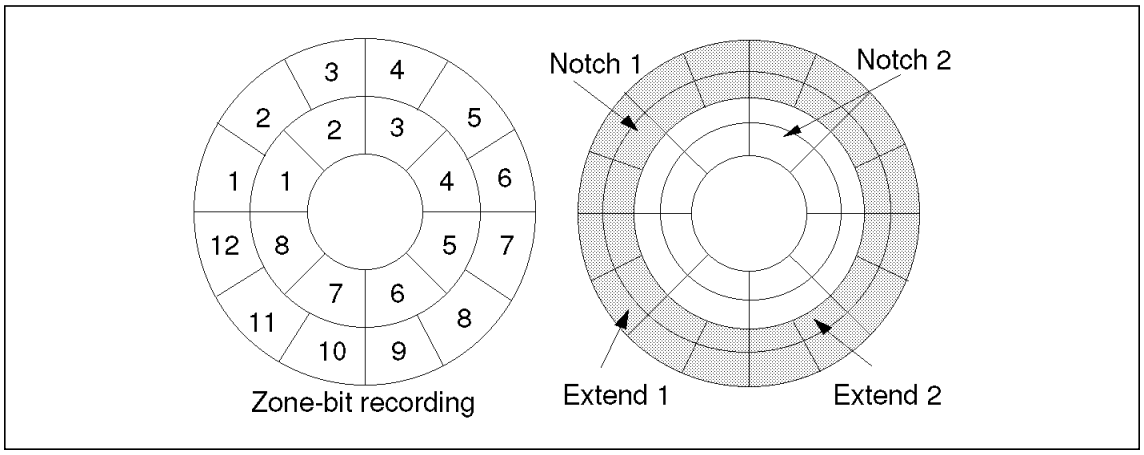


Figure 17. Zone-Bit Recording, Extends and Notches

On the left side of Figure 17, the zone-bit technique is shown. The outer circle contains more sectors per track then the inner one, allowing more data to be stored on the disk.

On the right side of the figure, the shaded area has a different number of sectors per track than the non-shaded area, making these areas part of different notches.

On the same right side, the outer shaded ring is larger and contains more bytes per track (when zone-bit recording is used) than the inner shaded ring, making these two rings part of different extends.

Data is structured on the SCSI direct access storage device in a logical manner. A sector (or logical block) is the smallest addressable unit. Sectors are typically 512 bytes, although SCSI allows each sector to have any size between 1 and 64K. Different tracks are allowed to have different number of sectors (zone-bit recording).

Every SCSI device uses a set of parameters to control its mode of operation. Three versions of the parameter sets are used by a SCSI device:

1. **Current set**. This set contains the values that are currently active for the device. The current set is not permanent. Its values are lost when the SCSI subsystem is reset.
2. **Default set**. This set contains values defined by the manufacturer. The set is permanently stored on the device and can not be modified, but may be copied to the current set at anytime.

3. **Saved set**. This set contains the values that are preferred by the user. These values are stored on the device, and are copied to the current set during a reset of the SCSI disk subsystem.

Each parameter set contains several groups of parameter pages. Each group deals with a certain configuration aspect of the device and each page deals with a more specific aspect within the group. Some parameter pages are common to all devices and other pages are specific to a certain device type. It is also possible for a vendor to implement their own parameter pages. The layout for the mandatory pages is fixed and there are SCSI commands to determine which pages are supported by a device and what the layout for each page (including vendor pages) is.

The following list briefly describes the parameter groups for an SCSI DASD:

- **Diagnostic** parameters, specifying diagnostic operations that the device should be able to perform
- **Log** parameters, specifying descriptions of events and counters for these events. For example, log events for a DASD include buffer overflow, medium errors and verify errors.
- **Mode** parameters, specifying the operational parameters that the device supports, such as the drive geometry, caching and defect support.
- **Vital Product Data** parameters, specifying information about the device and its vendor.

5.2.5.4 Command Set Layer

The SCSI command set is very extensive. It contains commands for all device models and also separate commands that apply to only one device model.

The SCSI command set layer specifies mandatory and optional commands, as well as allowing vendor-unique commands. Each device must implement the mandatory commands from the basic command set as well as the mandatory commands specified by its device class.

A SCSI command is usually a sequence of 6, 10 or 12 bytes, called a *Command Descriptor Block* (CDB). The following list shows the function of each field in the CDB:

- **CDB Length**. The length of the CDB. The 10 and 12 byte descriptor blocks are able to address more logical blocks. Using a 6 byte descriptor block on a device that uses 512 bytes per block, limits the maximum capacity of the device to 1 GB.

- **Command**. One byte is used to specify the actual command that has to be executed by the target. This allows for a total of 256 commands.
- **Target ID**. The target SCSI ID identifies the target device on the SCSI bus that has to execute the command.
- **LBA**. The Logical Block Address identifies the address of the first logical block involved in the command.
- **Transfer Size**. Specifies the number of sequential logical blocks involved in the command
- **Control Info**. Used to specify control info about the command. For example, control info will specify if a command is part of a linked chain of commands.
- **Reserved**. These bits will be used in future versions of SCSI

Table 14. SCSI Command Descriptor Block Fields

CDB Length	Command	Target ID	LBA	Transfer Size	Control Info	Reserved
6 bytes	8-bit	3-bit	21-bit	8-bit	8-bit	none
10 bytes	8-bit	3-bit	32-bit	16-bit	13-bit	8-bit
12 bytes	8-bit	3-bit	32-bit	32-bit	13-bit	8-bit

Table 14 shows the content of the SCSI Command Descriptor Blocks and the size of the fields in the CDB.

The mandatory basic SCSI Commands deal with:

- Query and Set Device Parameters
- Test Device
- Error Management

The mandatory DASD SCSI Commands deal with:

- Initialize Device (format)
- Read and Write Data

Appendix B.6, “Basic SCSI Command Set” on page 110 lists all basic SCSI commands. Appendix B.7, “SCSI Command Set for DASD” on page 111 lists the SCSI commands for direct access storage devices.

Because of the complexity and variety of SCSI devices, operating system support for the SCSI disk subsystem is usually implemented as one or more device drivers. In the early SCSI days, users had to load separate drivers for each device. To prevent this, several device driver standards have been

developed. These standards require just one device driver for each device type. The two most popular device driver standards are:

- **Advanced SCSI Programming Interface** (ASPI). Developed by Adaptec. ASPI is the most widely used standard for SCSI device drivers.
- **Common Access Method** (CAM). The standard that has been proposed to ANSI as the SCSI device drivers standard.

Device drivers following these standards are usually split into two categories:

1. A host adapter specific driver, which allows the SCSI host adapter of your system to talk ASPI or CAM.
2. A device type specific ASPI or CAM driver, which allows the host adapter to talk ASPI or CAM to the device type.

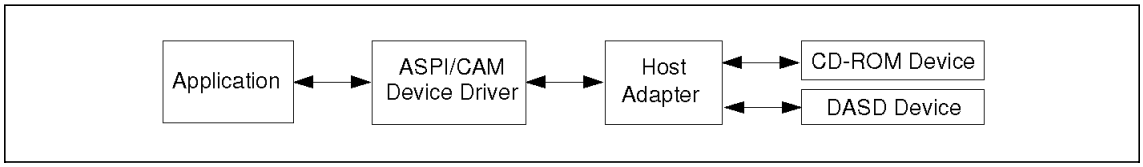


Figure 18. SCSI Device Driver Implementation

5.2.6 Hardware Components

The next sections describe the hardware components in a SCSI disk subsystem.

5.2.6.1 Host Adapter

The SCSI Host adapter is the interface between the host system and the SCSI disk subsystem. All requests from the host system are send to the SCSI host adapter, which is responsible for:

- Translating host commands into SCSI disk subsystem commands
- Sending the translated SCSI commands to the correct SCSI devices
- Transporting data between the SCSI devices and the host system

A SCSI host adapter is attached to the SCSI bus and to the system bus of the host system. Sometimes, the SCSI host adapter is integrated on the system board of the host system, but more often it is a separate adapter that may be installed in an adapter slot of the host system. Most host systems allow the installation of multiple SCSI adapters.

Usually, the frequencies of the system bus do not match those of the SCSI bus, so the host adapter has to adapt the data flow between the host system and the SCSI disk subsystem to these frequencies. Some host adapters use a small buffer to do this, while others employ larger buffers that are also able to perform some kind of caching.

Some SCSI host adapters also offer other types of functions, such as sound. There are often restrictions on the SCSI interfaces provided by these mixed function adapters.

The SCSI Host adapter is usually configured to use the highest SCSI ID. This allows the host adapter to always win during arbitration, ensuring that it can always control the bus when it needs to.

5.2.6.2 SCSI Direct Access Devices

SCSI direct access storage devices are packaged with their own controllers. The controller is responsible for executing the SCSI commands it receives. The controller must be able to recognize and execute all basic SCSI commands (see appendix B.6, “Basic SCSI Command Set” on page 110) and all SCSI DASD commands (see appendix B.7, “SCSI Command Set for DASD” on page 111).

Todays SCSI devices are able to provide capacities of 6 GB. In the near future, devices with capacities of 10 GB are expected.

Most SCSI DASD's are equipped with jumpers and/or switches to determine the SCSI ID of the device and the terminator function. Some devices also allow a delay in their start up sequence during power up to prevent a power supply overload.

More details on SCSI direct access storage devices can also be found in 5.2.5.3, "Device Model Layer" on page 56.

5.2.6.3 Cabling and Terminators

The cable specifications for the SCSI disk subsystem are specified in 5.2.5.1, "Physical Interface Layer" on page 52.

Terminators are needed at both ends of the SCSI bus to ensure the quality of the signals on the SCSI bus. Since most SCSI devices include terminator logic, the first and the last device on the SCSI bus should have their terminator logic activated. All other SCSI components on the bus should have their terminator logic deactivated.

Usually, the first device on a SCSI bus is the SCSI Host adapter, and the termination has already been taken care of. Some host adapters allow a SCSI bus to be partly internal and partly external to the host system unit. These adapters include software to correctly set the termination on the adapter, depending on the situation.

Most SCSI devices allow their termination logic to be activated using switches, jumpers or pull-out plugs. Consult the manual of the SCSI device for precise instructions on activating or deactivating the termination logic.

5.2.7 Performance

SCSI is a parallel interface, because it uses 8, 16 or 32 lines (depending on the SCSI mode) for data signals. This enhances the performance capabilities of SCSI enormously, compared to ST506 and ESDI.

All SCSI data transfer modes may be used in two different methods, asynchronous and synchronous. Synchronous mode is a bit faster, since less overhead is needed during the data transfer. But asynchronous mode may yield a better system performance in some situations.

All SCSI devices are able to negotiate which data transfer mode and method will be used. The method used to transfer data depends on the devices involved in the transfer. If a device is involved that has a relatively long response time or low operating rate, it is better to use the asynchronous mode to effectively use the SCSI bus. For example, a Printer device can only print at a certain speed, far below that of even the slowest SCSI transfer rate, so it makes sense to communicate with a SCSI printer asynchronously, allowing the SCSI bus to be used by other devices while printing.

To determine the fastest common mode, the device configuration parameters of the initiator and target are compared. These device configuration parameters can be programmed by the operating system (device driver) or the end user. Check the documentation of the device and/or the operating system device driver to see if your device operates at its most efficient mode.

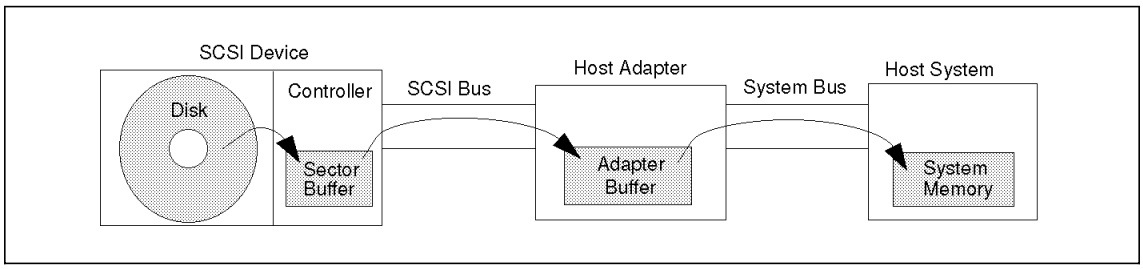


Figure 19. SCSI Data Transfer Overview

Figure 19 shows the data flow during the execution of a read command. As you can see, a SCSI data transfer involves three steps:

- **Device to Sector Buffer**. The data is moved from its physical location on the disk medium to a sector buffer in the controller part of the SCSI Device. During this step, the data is translated from its physical form into a digital format that is understandable by the host system. The performance of this step is determined by the implementation of the

manufacturer of the SCSI device. There are no specifications for this step in the SCSI standards.

- **Sector Buffer to Host Adapter Buffer**. The data is moved from the sector buffer on the SCSI controller to the buffer of the SCSI host adapter. The performance of this step is clearly defined in the SCSI specifications. Table 10 on page 51 lists all the SCSI data transfer modes.
- **Host Adapter Buffer to System Memory**. The data is moved from the buffer on the host adapter to the memory of the host system. The performance of this step is determined by the specifications of the system bus of the host system. See the technical specifications of your computer for more details on the performance of your system bus.

Most SCSI DASDs have a buffer which is able to store commonly or recently used blocks of data. The SCSI DASD device model and command set include commands and parameter pages to modify the content or behavior of the buffer. Sometimes the end user is allowed to use these commands or modify the parameters. The following list describes these commands and parameters:

- **Lock/Unlock Cache**. This command locks or unlocks blocks of data in the cache. When a block is locked in the cache, it may not be discarded from the cache. An unlocked block is allowed to be discarded. Locking a block of data is useful for data that is often retrieved.
- **Multiplication Factor**. This parameter defines the minimum and maximum size of the blocks of data in the cache. Setting these sizes correctly can enhance the performance enormously. For example, in an environment where mostly small amounts of data are transferred, setting the minimum and maximum size to values matching the small amounts of data will ensure that most data can be buffered.
- **Pre-Fetch Data**. This command copies data from the device to its buffer, allowing a better response time when the data is actually requested by the host system or another device.
- **Read Cache Disable**. This parameter forces bypassing of the buffer. Each request for data must be retrieved from the medium.
- **Retention Priorities**. The read and write retention priorities determine the order in which data is discarded from the buffe when the buffer is full.
- **Synchronize Cache**. This command forces the storage of data in the buffer that has not been written yet.
- **Write Cache Enable**. This parameter enables or disables write caching. When write caching is enabled, data specified in a write command may

be temporarily stored in the device buffer, allowing the write command to be executed much faster.

5.2.8 Availability

SCSI cables should have an impedance of at least 80 ohms to maintain the quality of the electrical signals. The higher the impedance, the better the signal quality. Better signal quality results in a higher availability.

SCSI direct access storage devices employ two error detection and correction facilities:

- Data is stored on a disk using Error Correction Code (ECC) bytes. During a read of the data, the ECC bytes are used to check the integrity of the data.
- Physical disk addresses are verified by Cyclic Redundancy Code (CRC) bytes stored as a part of each sector. During a read of the data, the CRC bytes are used to verify if the correct data is read.

SCSI DASD devices have four lists containing medium defects:

- **Permanent List**. This list contains the addresses of defective sectors found by the vendor. The list is permanent and can not be altered.
- **Certification List**. This list contains the addresses of defective sectors found during a format of the device. This list is refreshed each time the device is formatted.
- **Defect List**. This list is sent by an initiator prior to formatting a device and contains addresses of defective sectors. This feature may be used to prevent the usage of certain sectors on a device.
- **Grown Defect List**. This list contains the addresses of all defective sectors found during formatting and normal operation of the device.

When a defective sector is encountered during the normal operation of a SCSI device, the reassign block is used to fix the defect. The implementation of the command is defined by the vendor, but is usually implemented so that the closest spare sector takes the place of the defective sector. If the error occurs during a read operation, the data that can be recovered from the defective sector is copied to the spare sector. When the defective sector is encountered during a write operation, the write operation is repeated to the spare sector without loss of data integrity.

Another possibility to enhance the availability of SCSI devices is by employing *target routines* of the SCSI device. Each SCSI device is allowed to have a maximum of eight target routines. These routines are unique to the device and if they are implemented, they are usually implemented to perform diagnostics or statistics routines.

The SCSI bus contains one parity line for each 8 data lines. This provides a method for detecting data errors while the data is transported through the SCSI bus. The SCSI bus also contains many ground lines, reducing the interference and heat generation of the cable.

The SCSI command set for DASDs contains a lot of commands to ensure the availability of the SCSI disk subsystem. See appendix B.6, “Basic SCSI Command Set” on page 110 and appendix B.7, “SCSI Command Set for DASD” on page 111 for an overview of these commands.

5.2.9 Expandability

The number of SCSI devices that may be attached to a SCSI bus depends on two things:

- The number of data lines in the SCSI Cable
- The SCSI controllers on the devices

Each SCSI device identifies itself on the SCSI bus using its corresponding data line. For example, the SCSI device with ID 4 uses the line for Data Bit(4), while the host adapter uses Data Bit(7). In a typical SCSI system, the SCSI cable has only eight data lines, limiting the number of devices to a maximum of eight.

Wide SCSI is able to use 16 data lines. If a Wide SCSI cable is used that has 16 data lines, the SCSI bus is able to support 16 devices. But the support for wide SCSI must also be implemented in the SCSI controller. The total number of SCSI devices attachable to a SCSI Wide controller (16-bit) is:

- 16-bit wide devices only, fifteen devices plus one controller
- 8-bit narrow devices only, seven devices plus one controller
- Mix of 8-bit narrow devices and 16-bit wide devices, seven 8-bit plus seven 16-bit plus two controller. It is important to use the lower 8-bits addressing for the 8-bit narrow devices only.

Above is especially important in server systems. Pay attention when attaching non-wide devices, such as a CD-ROM or tape drive, to systems that are also using wide SCSI devices and host adapters.

A SCSI device usually consists of a SCSI controller and a single physical device. It is however possible to attach a maximum of eight physical devices to a single SCSI connector. This makes it possible to connect more then eight SCSI devices to a single SCSI bus.

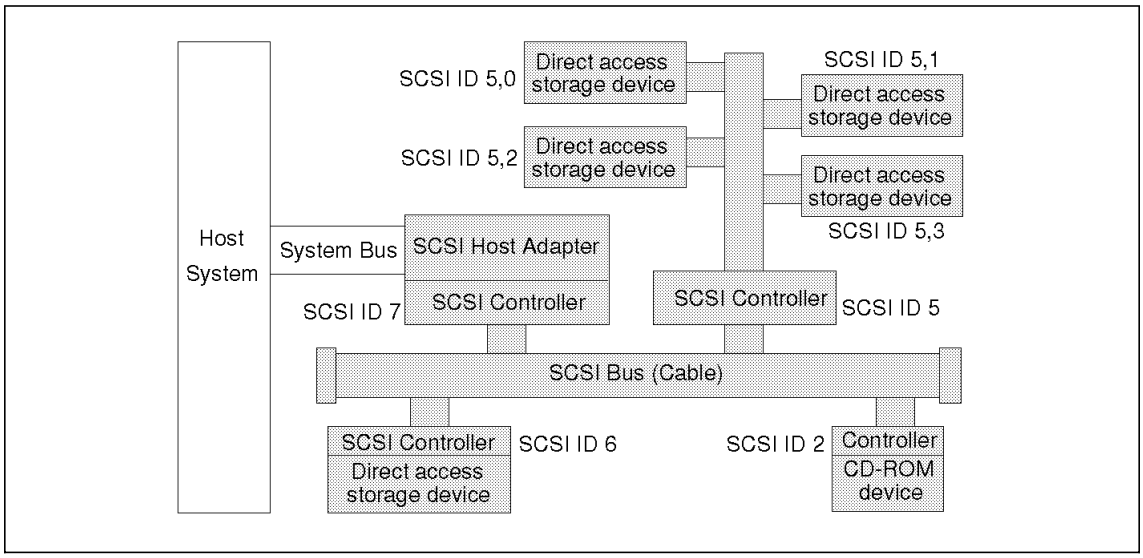


Figure 20. Multiple Physical Devices Attached to a Single SCSI Controller

Figure 20 shows a logical SCSI device which has multiple physical devices. Note that the SCSI ID's are different for these types of devices. The SCSI ID is now made up of two parts. The first number identifies the SCSI controller, and the second number uniquely identifies the physical device attached to that controller. Since the host adapter can not have multiple physical devices attached to it, the largest number of physical devices that may be attached to a single 8-bit SCSI bus (narrow) is 56 (seven controllers with eight physical devices each).

It is allowed to connect multiple host systems to the same SCSI bus (See Figure 21 on page 70). Since most existing cables and host adapters do not support this kind of configuration, attaching multiple host systems to the same SCSI bus is usually accomplished by using a SCSI switch (See Figure 22 on page 70).

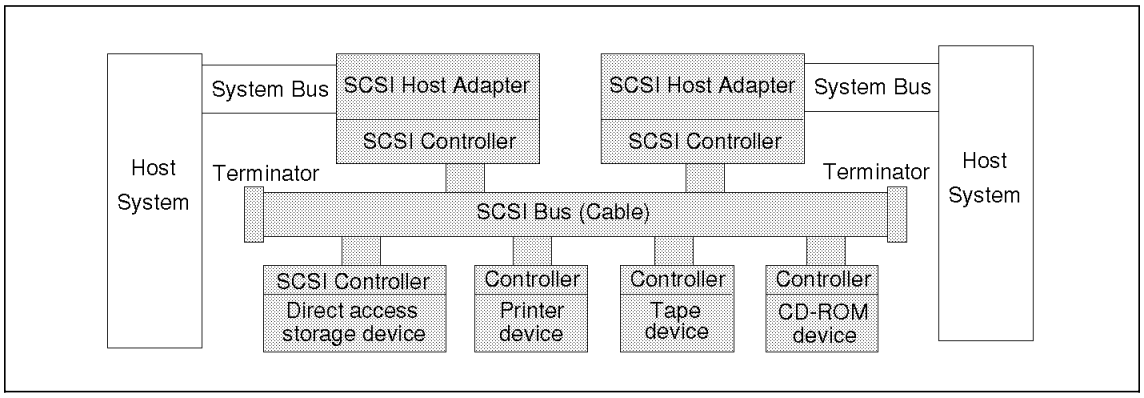


Figure 21. SCSI Bus with Multiple Controllers

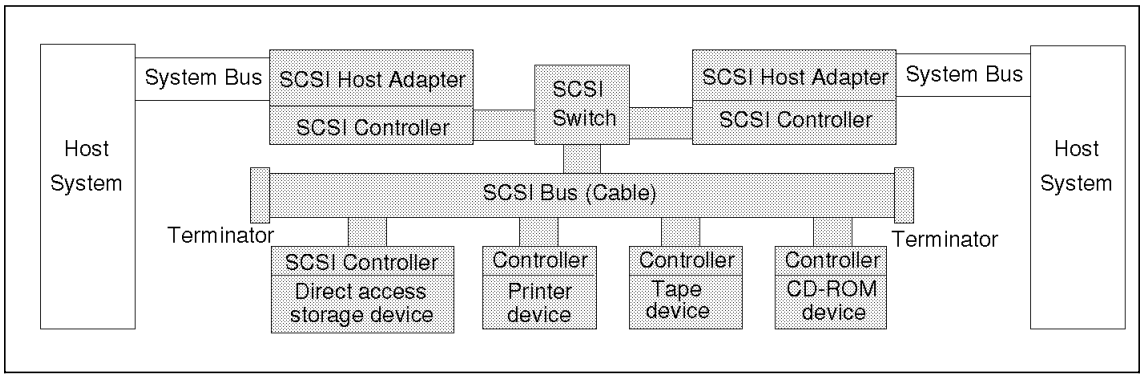


Figure 22. Multiple Host Systems Attached to One SCSI Bus Using a SCSI Switch

Most host systems allow multiple SCSI controllers to be used at the same time (see Figure 23 on page 71).

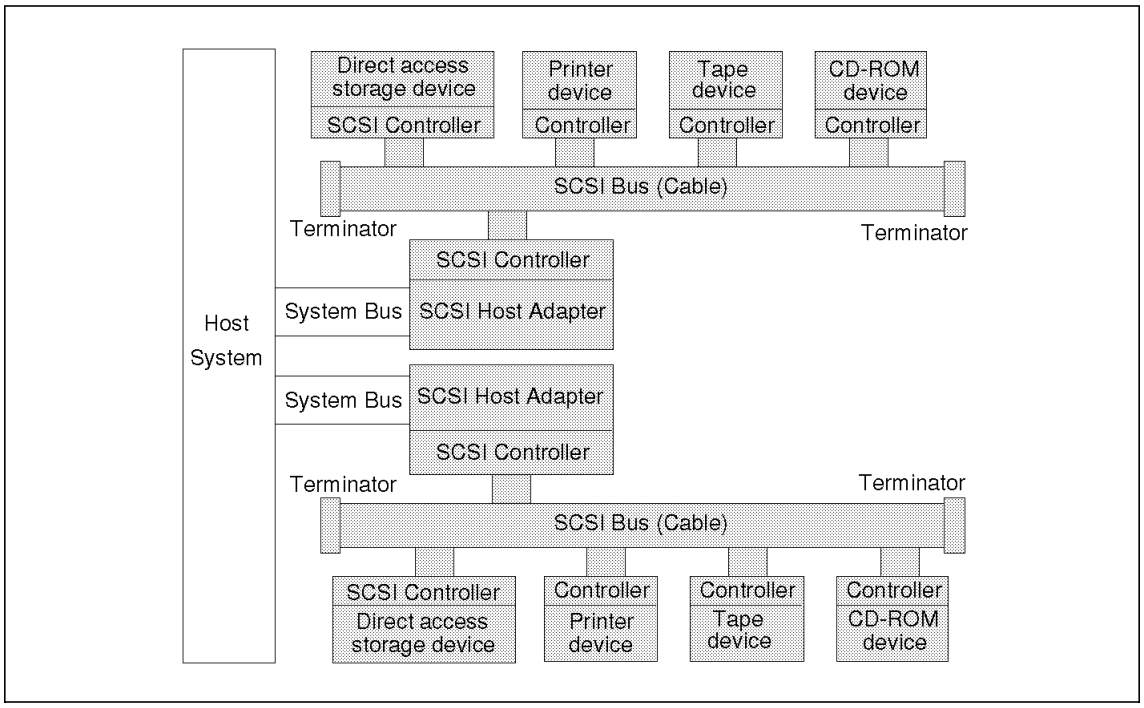


Figure 23. Multiple SCSI Controllers in One Host System

Most SCSI host adapters are able to support internal and external SCSI buses. Some are even capable of supporting internal and external SCSI devices on the same SCSI bus.

The maximum cable length of a normal SCSI bus is six meters. If fast SCSI is used, that length is halved to three meters. If Fast-20 (or Ultra) SCSI is used, the length is again halved to only 1.5 meters. If you use external storage enclosures, make sure you also count the length of the SCSI cable inside of the enclosure.

The maximum capacity allowed of a SCSI device depends on the method being used to address the blocks on the SCSI device. A 10 byte SCSI command reserves 32-bit for block addressing. Using a 512 bytes block size, the capacity for SCSI devices is in the terabytes. Some SCSI device drivers depend on the BIOS of the host system to address SCSI devices. These drivers limit the capacity for each drive to 8.5GB.

SCSI supports propriety commands. Although these commands may improve the performance, availability and expandability of your disk subsystem, they

often force you to buy all SCSI components from the same vendor in order to use these extra features.

Chapter 6. Redundant Array of Independent Disks (RAID Technology)

Drive array technology can improve mass store performance and increase the reliability of the disk subsystem. The purpose of this chapter is to explain the advantages of the different available RAID definitions.

6.1 Disk Arrays

The capacity of single large disks has grown rapidly, but the performance improvements have been modest, when compared to the advances made in the other subsystems that make up a computer system. The reason for this is that disks are mechanical devices, affected by delays in seeks and the rotation time of the media.

In addition, disks are often among the least reliable components of the computer systems, yet the failure of a disk can result in the unrecoverable loss of vital business data, or the need to restore a tape backup with consequent delays.

The use of arrays of independent disks can offer a solution to these concerns.

There is nothing unusual about connecting several disks to a computer to increase the amount of storage. Mainframes and minicomputers have always had banks of disks. The disk subsystem is called a disk array when several disks are connected and accessed by the disk controller in predetermined patterns designed to optimize performance and/or reliability.

The driving force behind disk array technology is the observation that it is cheaper to provide a given storage capacity or data rate with several small disks connected together than with a single disk.

6.2 RAID

© Copyright IBM Corp. 1995

RAID stands for Redundant Arrays of Independent[1] Disks, and provides a method of classifying the different ways of using multiple disks to increase availability and performance.

6.2.1 RAID Classifications

Disk arrays seem to have been invented independently by a variety of groups. For example, the Computer Architecture group at the University of California, Berkeley invented the term Redundant Arrays of Independent Disks (RAID).

The original RAID classification described five levels of RAID (RAID-1 through 5). To these have been added RAID-0 (data-striping), RAID-1 Enhanced (data stripe mirroring) and Orthogonal RAID-5 (which includes extra redundancy of components such as disk adapters). RAID-0 is not a pure RAID type, since it does not provide any redundancy.

Different designs of arrays perform optimally in different environments. The two main environments are those where high transfer rates are very important, and those where a high I/O rate is needed, that is, applications requesting short length random records. For more information an this we recommend the ITSO redbook IBM PC Server Disk Subsystem Configuration and Sizing, SG24-4525-00.

Table 15 shows the RAID array classifications, and is followed by brief descriptions of their designs and capabilities.

Table 15 (Page 1 of 2). RAID Classifications

RAID Level	**Description**
RAID-0	Block Interleave Data Striping without Parity
RAID-1	Disk Mirroring/Duplexing
RAID-1 (Enhanced)	Data Stripe Mirroring
RAID-2	Bit Interleave Data Striping with Hamming Code
RAID-3	Bit Interleave Data Striping with Parity Disk
RAID-4	Block Interleave Data Striping with One Parity Disk
RAID-5	Block Interleave Data Striping with Skewed Parity

[1] formerly called ″Inexpensive″.

Table 15 (Page 2 of 2). RAID Classifications	
RAID Level	**Description**
Orthogonal RAID-5	RAID-5 with Additional Redundancy (such as disk adapters)

6.2.1.1 RAID-0 - Block Interleave Data Striping without Parity

Striping of data across multiple disk drives without parity protection is a disk data organization technique sometimes employed to maximize DASD subsystem performance (for example, Novell NetWare's "data scatter" option).

An additional benefit of this data organization is that of "drive spanning". With data striped across multiple drives in an array, the logical drive size is the sum of the individual drive capacities. The maximum file size may be limited by the operating system.

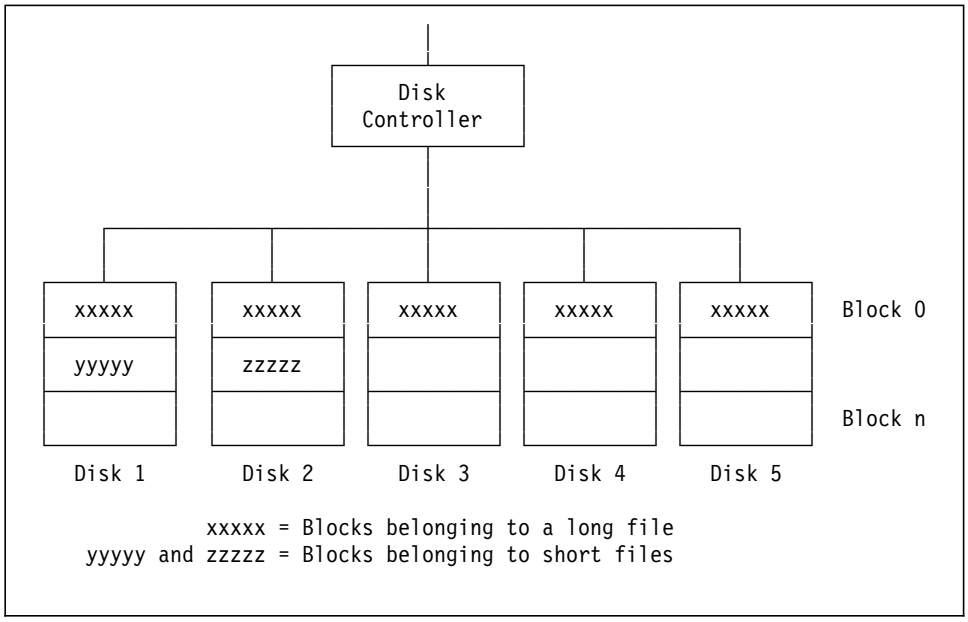


Figure 24. RAID-0 (Block Interleave Data Striping without Parity)

Data striping improves the performance with large files since reads/writes are overlapped across all disks. However, reliability is decreased as the failure of one disk will result in a complete failure of the disk subsystem according to the formula:

$$\text{Mean Time to Failure} = \frac{\text{Mean Time to Failure of a single disk}}{\text{Number of Disks in the array}}$$

6.2.1.2 RAID-1 - Disk Mirroring/Duplexing

This approach keeps two complete copies of all data, so whenever the computer makes an update to a disk, it can arrange to duplicate that update to a second disk, thus mirroring the original. Either disk can fail, and the data is still accessible. Additionally, because there are two disks, a read request can be satisfied from either device, thus leading to improved performance and throughput. Some implementations optimize this by keeping the two disks 180 degrees out of phase with each other, thus minimizing latency.

However, mirroring is an expensive way of providing protection against data loss, because it doubles the amount of disk storage needed (as only fifty percent of the installed disk capacity is available for data storage)

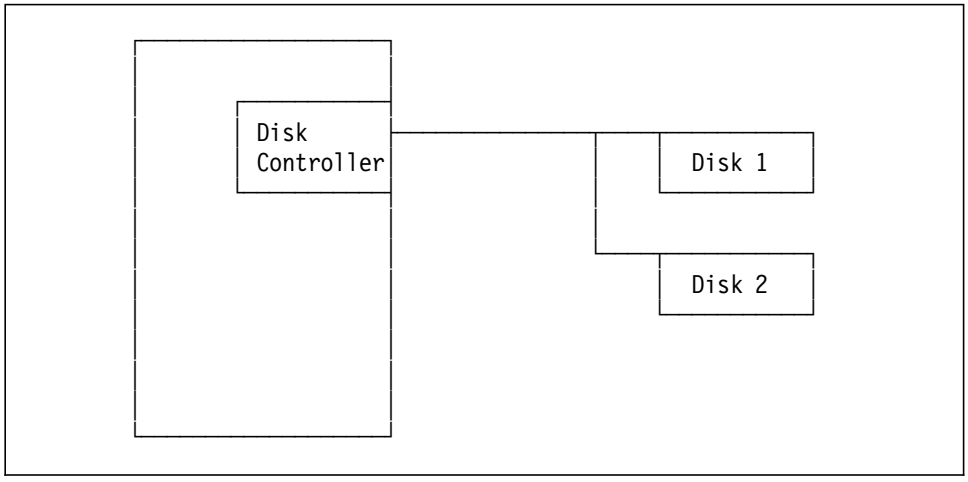


Figure 25. RAID-1 (Disk Mirroring)

Disk mirroring involves duplicating the data from one disk onto a second using a single controller.

Disk duplexing is the same as mirroring in all respects, except that the disks are attached to separate controllers. The server can now tolerate the loss of one disk controller, as well as or instead of a disk, without loss of the disk subsystem′s availability or the customers′ data. Since each disk is attached

to a separate controller, performance and throughput may be further improved.

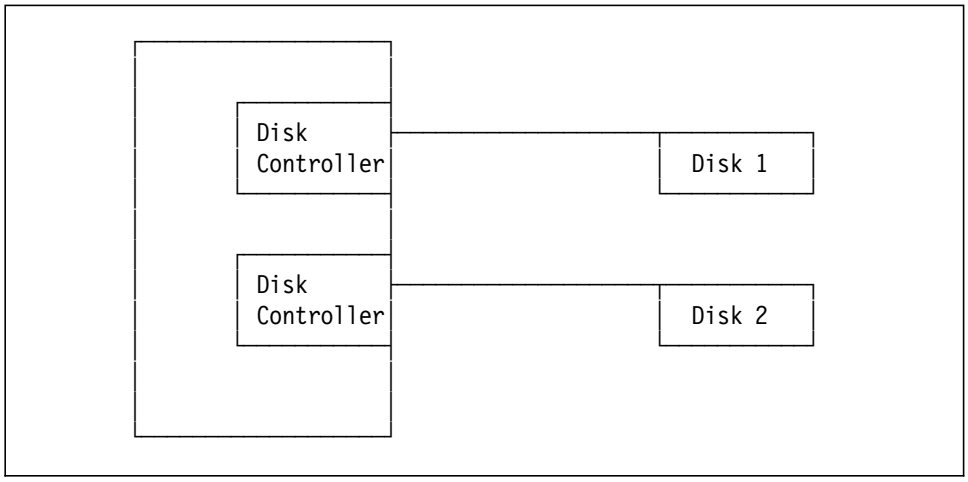


Figure 26. RAID-1 (Disk Duplexing)

6.2.1.3 RAID-1 Enhanced - Data Strip Mirroring

RAID level 1 supported by the IBM Server 95 Array system provides an enhanced feature for disk mirroring that stripes data and copies of the data across all the drives of the array. The first stripe is the data stripe; the second stripe is the mirror (copy) of the first data stripe, but shifted one drive. Because the data is mirrored, the capacity of the logical drive, when assigned to RAID 1 Enhanced, is fifty percent of the physical capacity of hard disk drives in the array. Some vendors have also referred to this as RAID-10, the combining of RAID-0 and RAID-1 while other vendors use RAID-6, the next available RAID number.

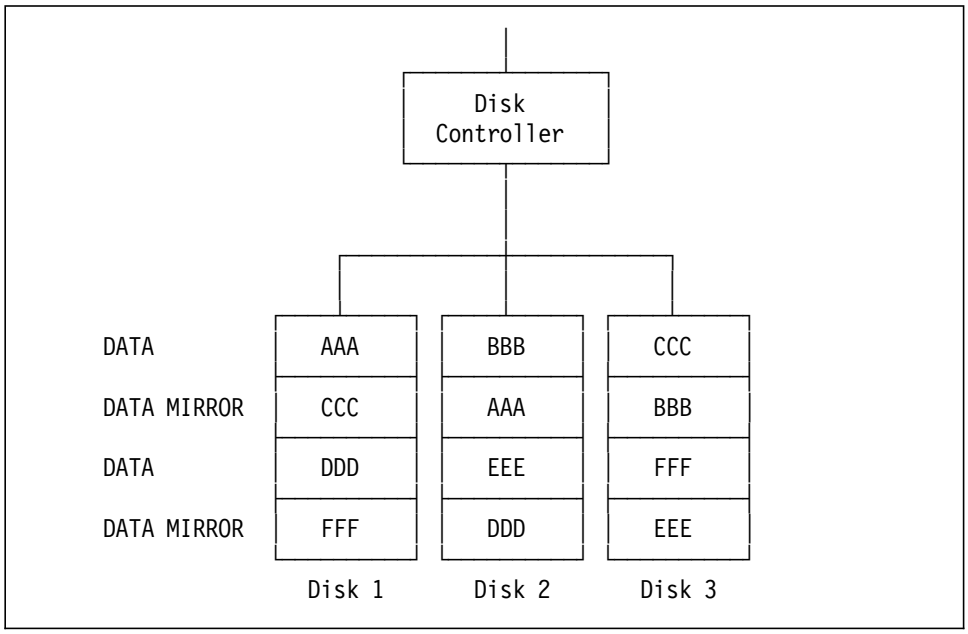


Figure 27. RAID-1 (Enhanced, Data Strip Mirroring)

6.2.1.4 RAID-2 - Bit Interleave Data Striping with Hamming Code

This type of array design is another form of data striping, and spreads the data across the disks one bit or one byte at a time in parallel. It is called bit (or byte) interleaving.

Thus, if there were five disks in the array, a sector on the first drive will contain bits 0, 5 and so on of the data block; the same sector of the second drive will contain bits 1 and 6, and so on as shown in Figure 28 on page 79.

RAID-2 tries to get around the fifty percent disk overhead in RAID-1 and provides redundancy by using the Hamming Code. The error correction bit scan be associated with from 4 to 32 or more bits of data. However, the overhead degenerates to fifty percent with fewer bits. For example, if data were grouped into bytes, eleven drives in total would be required, eight for data and three for Hamming Code (Note: for clarity the Hamming Code drives are not shown in Figure 28 on page 79). The 8-3 configuration reduces the overhead to twenty-seven percent.

An array of this design will perform optimally when large data transfers are being done. The host will see the array as a single, or “logical” drive, and

the data transfer rate will be the product of the number of drives in the array and the transfer rate of the individual drives.

This design is unable to handle multiple, simultaneous, small requests for data, unlike the previous design, so it is unlikely to satisfy the requirements for a transaction processing system that needs a high transaction rate.

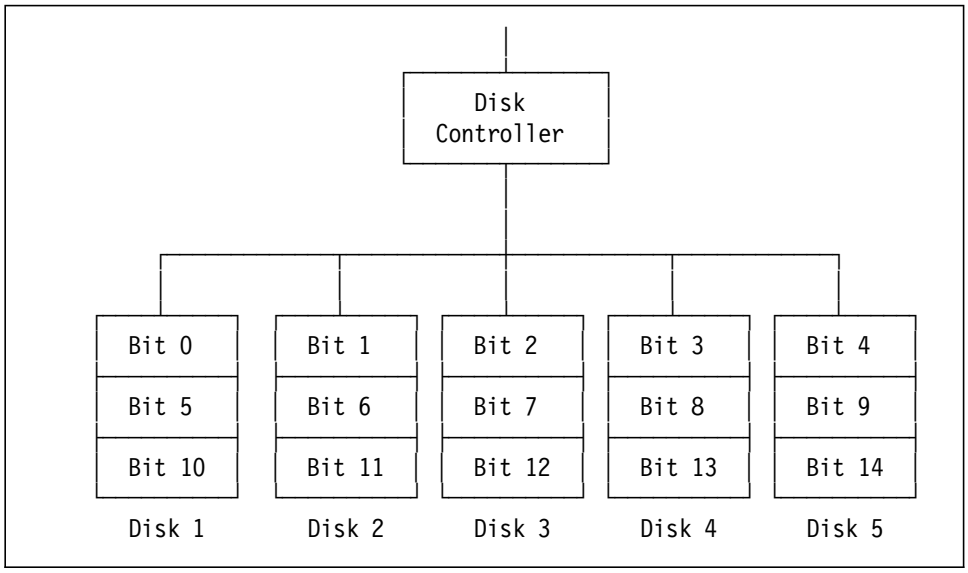


Figure 28. RAID-2 (Bit Interleave Data Striping with Hamming Code)

6.2.1.5 RAID-3 - Bit Interleave Data Striping with Parity Disk

The use of additional disks to redundantly encode customer's data and guard against loss is referred to as checksum, disk parity or error correction code (ECC). The principle is the same as memory parity, where the data is guarded against the loss of a single bit. The number of drives shown in Figure 29 on page 80 is to be considered a an example only. The actual number of drives can vary from 3 to n.

Four of the disks hold data, and can be accessed independently by the processor, while the fifth is hidden from the processor and stores the parity of the other four. Writing data to any of disks 1, 2, 3 or 4 causes the parity to be recomputed and written to disk 5. If any of the data disks were to subsequently fail, the data can still be accessed by using the information from the other data disks and the parity disk to reconstruct it.

Since the data is held on individually addressable disks, this design will offer a high I/O rate. Compared to a single disk of similar capacity, this array has more actuators for the same amount of storage. These actuators will work in parallel, as opposed to the sequential operation of the single actuator, thus reducing average access times (in read mode).

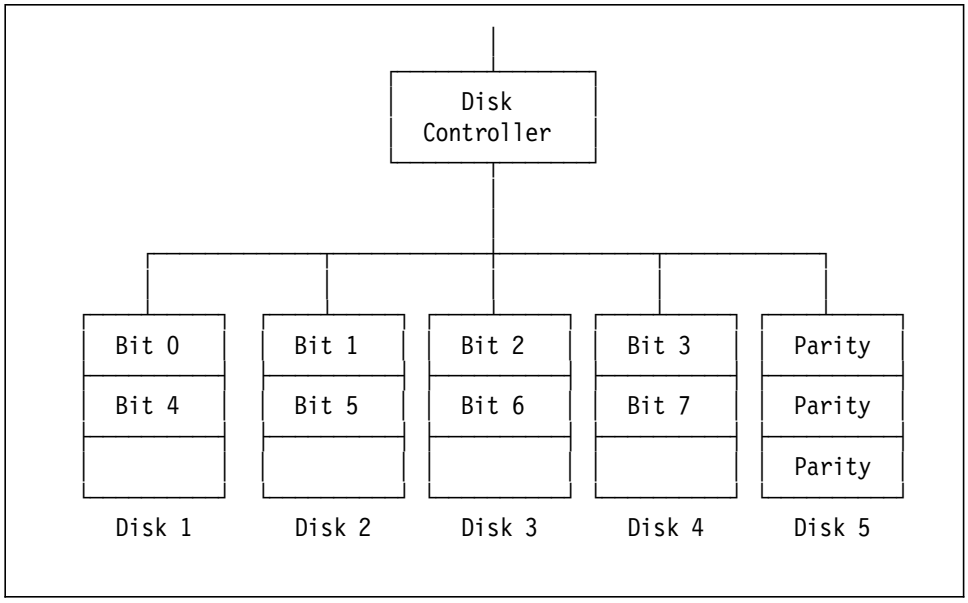


Figure 29. RAID-3 (Bit Interleave Data Striping with Parity Disk)

Multiple disks are used with the data scattered across them. One disk is used for parity checking for increased fault tolerance.

6.2.1.6 RAID-4 - Block Interleave Data Striping with One Parity Disk

The performance of bit-interleaved arrays in a transaction processing environment, where small records are being simultaneously read and written, is very poor. This can be compensated for by altering the striping technique, such that data is striped in block sizes that correspond to the record size being read. This will vary in different environments.
Super-computer type applications may require a block size of 64KB, while for most DOS applications 4KB will suffice.

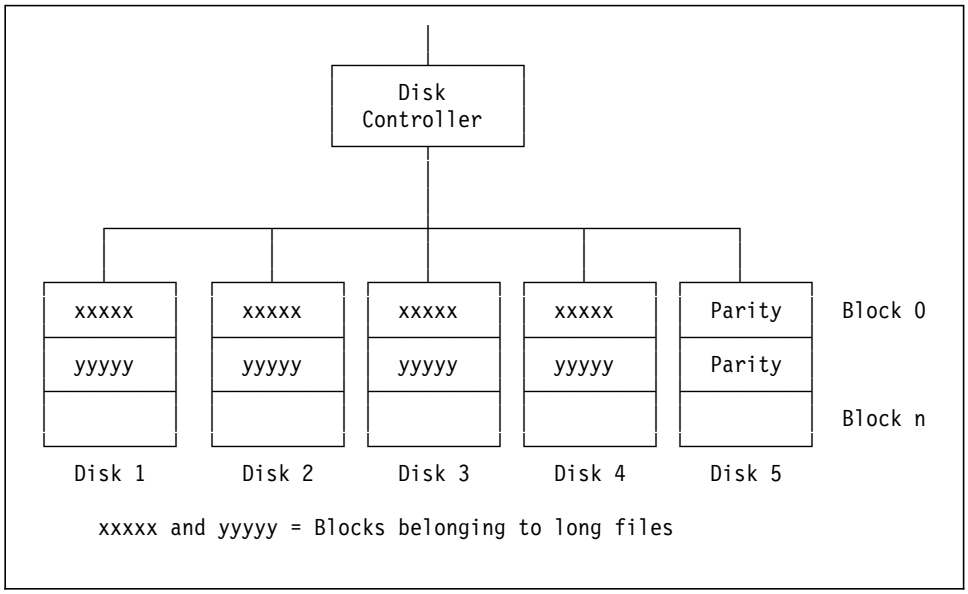


Figure 30. RAID-4 (Block Interleave Data Striping with One Parity Disk)

6.2.1.7 RAID-5 - Block Interleave Data Striping with Skewed Parity

The RAID-5 design tries to overcome all the problems seen in the previous RAID implementations. Data is striped across the disks to ensure maximum read performance when accessing large files, while having the data striped in blocks improves the array's performance in a transaction processing environment. Parity information is stored on the array to guard against data loss, and skewing is used to remove the bottleneck that would be created by having all the parity information stored on a single drive.

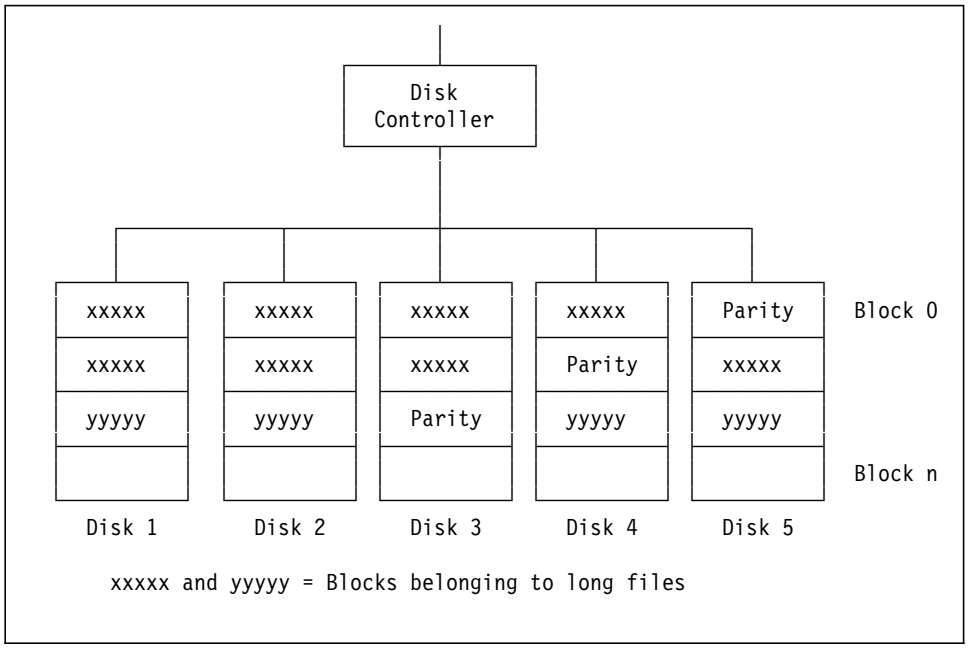


Figure 31. RAID-5 (Block Interleave Data Striping with Skewed Parity)

6.2.1.8 Orthogonal RAID-5

Orthogonal RAID-5 is an enhancement of RAID-5, which has been implemented in the PS/2 Server 195/295 internal disk array.

The performance of a disk subsystem depends on more than just the underlying performance of the disks. Multiple requests to one disk or across one adapter will typically take longer to satisfy than the same number of requests to multiple disks across multiple adapters.

In addition, the overall reliability of a standard RAID-5 system is dependent on the reliability of the one disk adapter to which all of the disks are connected.

Orthogonal RAID-5 solves both of these concerns by grouping the disk arrays orthogonally to the disk adapters, SCSI buses and power cables. This would be best implemented as a four-drive orthogonal RAID-5 array, where each disk would be connected via a different adapter and SCSI bus.

The result of this is that any one component of the disk subsystems, not just a disk drive, can fail with no loss of data and no interruption to system operation.

6.2.2 Summary of RAID Performance Characteristics

RAID-0: Block Interleave Data Striping without Parity

- Fastest data-rate performance
- Allows seek and drive latency to be performed in parallel
- Significantly out-performs single large disk

RAID-1: Disk Mirroring/Disk Duplexing and Data Strip Mirroring (RAID-1, Enhanced)

- Fast and reliable, but requires 100% disk space overhead
- Data copied to each set of drives
- With NetWare, performance almost as fast as RAID-0 due to split seeks. Split seeks send half of the operation to one side of the mirrored drives, and the other half to other side. This is valid for software implementations only.
- Split seeks are not implemented with OS/2 LAN Server
- No performance degradation with a single disk failure
- RAID-1 enhanced provides mirroring with odd number of drives

RAID-2: Bit Interleave Data Striping with Hamming Code

- Very fast for sequential applications, such as graphics modelling
- Almost never used with PC-based systems

RAID-3: Bit Interleave Data Striping with Parity

- Access to all drives to retrieve one record
- Best for large sequential reads
- Very poor for random transactions
- Poor for any write operations
- Faster than a single drive, but much slower than RAID-0 or RAID-1 in random environments

RAID-4: Block Interleave Data Striping with One Parity Disk

- Best for large sequential I/O
- Very poor write performance
- Faster than a single drive, but usually much slower than RAID-0 or RAID-1

RAID-5: Block Interleave Data Striping with Skewed Parity

- Best for random transactions
- Poor for large sequential reads if request is larger than block size
- Better write performance than RAID-3 and RAID-4

- Block size is key to performance, must be larger than typical request size
- Performance degrades in recovery mode that is when a single drive has failed

Orthogonal RAID-5: RAID-5 with Multiple Orthogonal Disk Adapters

- All the benefits of RAID-5
- Improved performance (due to load being spread across disk adapters)
- Improved reliability due to redundancy of disk adapters as well as disks

Table 16. Summary of RAID Performance Characteristics

RAID Level	Capacity	Large Transfers	High I/O Rate	Data Availability
Single Disk	Fixed (100%)	Good	Good	1
RAID-0	Excellent	Very Good	Very Good	Poor 2
RAID-1	Moderate (50%)	Good	Good	Good
RAID-2	Very Good	Good	Poor	Good
RAID-3	Very Good	Very Good	Poor	Good
RAID-4	Very Good	Very Good	Poor	Good
RAID-5	Very Good	Very Good	Good	Good
Orthogonal RAID-5	Very Good	Very Good	Good	Very Good

Note:

1 The MTBF (mean time before failure) for single disks can range from 10,000 to 800,000 hours.
2 Availability = MTBF of one disk divided by the number of disks in the array.

Chapter 7. Future Disk Subsystems

This chapter looks at the near future of disk subsystem technologies. We will describe those technologies that have the promise of becoming as important to the PC marketplace as IDE and SCSI-2 are today.

The following disk subsystem technologies will be examined :

- SCSI-3
- Fibre Channel (FC)
- IEEE-1394 (also known as FireWire)
- Serial Storage Architecture (SSA)

ATA-2 and/or ATA-3 will very likely also be major players in the PC marketplace in the near future. ATA-2 is already described in this book (see Chapter 5, “System Level Disk Subsystems” on page 33). Not enough is known yet about ATA-3 to grant it its own section in this book.

7.1 SCSI-3

SCSI-3 will be great step forward in the development of disk subsystems. SCSI-3 will further enhance the SCSI-2 interface in the following ways:

- SCSI-3 provides three new physical interface layers, SSA, FC and FireWire. These new layers provide better performance, higher availability and more expandability to SCSI.
- SCSI-3 is broken down into more then 15 standards, each dealing with a separate part. Because SCSI had become a very large standard, the separation makes the SCSI standard easier to maintain and better to work with. It also allows parts of SCSI-3 to be formalized much sooner.

7.2 Overview of SCSI-3 Standards

As you can see in Figure 32 on page 86, SCSI-3 is broken down into a lot of standards.

© Copyright IBM Corp. 1995

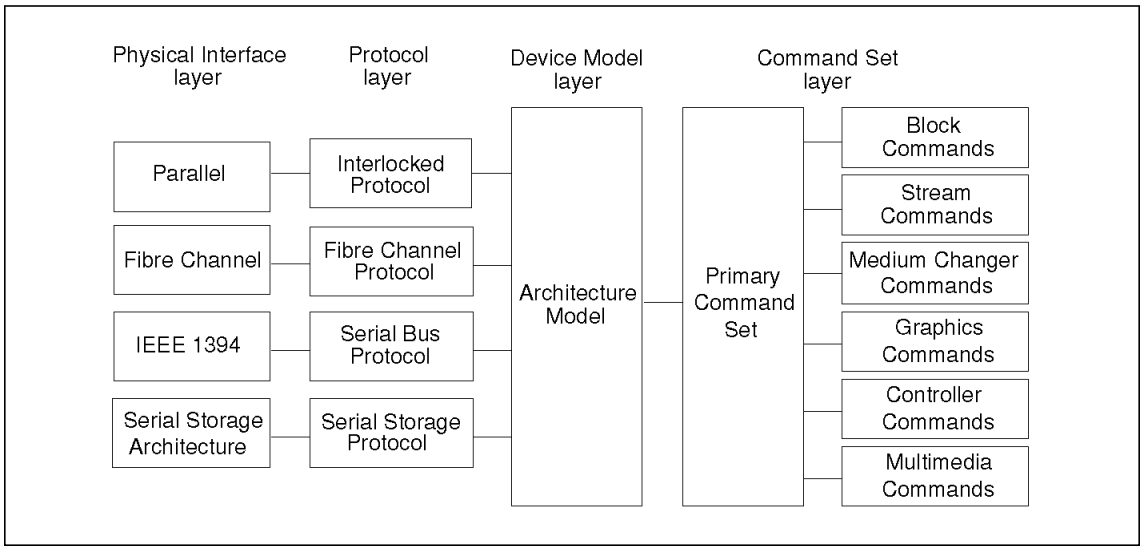


Figure 32. SCSI-3 Standards Overview

A short description of the function of each standard is given in Table 17. Most of the SCSI-3 enhancements take place in the physical interface layer. These enhancements are described in more detail in the following sections.

Table 17 (Page 1 of 2). Overview of SCSI-3 Standards

Standard	**Abbrev**	**Function**
SCSI-3 Interlocked Protocol	SIP	Describes the protocols used on the SPI bus.
Fibre Channel Protocol	FCP	Describes the protocols used on the FC bus.
Serial Bus Protocol	SBP	Describes the protocols used on the IEEE 1394 bus.
Serial Storage Protocol	SSP	Describes the protocols used on the SSA bus.
SCSI-3 Architecture Model	SAM	Describes the architecture of the SCSI-3 model (includes the SCSI-3 device models).
SCSI-3 Primary Commands	SPC	Describes the commands that all SCSI devices must implement.
SCSI-3 Block Commands	SBC	Describes the commands used to transmit *blocks* of data.
SCSI-3 Stream Commands	SSC	Describes the commands used to transmit *streams* of data.
SCSI-3 Medium Changer Commands	SMC	Describes the commands used to change a medium in a device.

Table 17 (Page 2 of 2). Overview of SCSI-3 Standards

Standard	Abbrev	Function
SCSI-3 Graphic Commands	SGC	Describes the commands that involve graphics.
SCSI-3 Controller Commands	SCC	Describes the commands used to configure and test the controller of a device.
SCSI-3 Multimedia Commands	MMC	Describes commands that involve multimedia data, such as audio and video.

7.3 SCSI-3 Parallel Interface (SPI)

SPI describes the physical interface layer for the SCSI-3 parallel interface. This is essentially a new version of the current SCSI-2 bus. Enhancements provided by SPI include:

- The SCSI P, Q and L cables (see Table 11 on page 52)
- Connection of more than eight SCSI devices to a single SCSI bus
- Fast-20 SCSI
- 3.3 volts version for portable computers

7.4 Fiber Channel (FC)

FC describes the physical interface layer for SCSI-3 fiber channel. FC is capable of supporting 128 devices, data transfer rates of 200 MBps and a maximum cable length of 20 meters for copper cables and 10 kilometers for fiber optic cables.

Of the three new physical interfaces provided by SCSI-3, FC currently promises the best performance. Another advantage of the FC bus is that it is also capable of transmitting other types of data. FC is not only restricted to transporting disk subsystem data, but can transport network subsystem data as well.

FC will probably have the highest cost of all SCSI physical interface layers. This will probably restrict its initial usage to high-end servers.

One of the more popular FC implementations is Fiber Channel - Arbitrated Loop (FC-AL). FC-AL provides a relatively inexpensive form of FC by implementing functions on the FC-AL device that are otherwise provided by separate products.

FC-AL connects devices in an arbitrated loop, somewhat similar to the traditional SCSI bus or a token-ring network. Devices in FC-AL need to arbitrate in order to be able to use the bus and communicate with other devices. When only two FC-AL devices are used, they will operate in a point-to-point like fashion.

7.5 IEEE 1394 High Performance Serial Bus (SBP)

SBP describes the physical interface layer for the IEEE 1394 high performance serial bus. This standard was first developed by Apple and is also knows as FireWire. IEEE 1394 is capable of supporting 64 devices at a maximum data transfer rate of 12.5 MBps and using a maximum cable length of 4.5 meters.

SBP provides some other enhancements on SCSI-2, some of which are not provided by the other physical interface layers. These improvements include:

- No terminators are required
- Automatic SCSI ID configuration
- Isochronous services that guarantee delivery of data at a certain time

SBP is a relatively inexpensive solution. Its specifications make SBP a very good solution for multimedia workstation environments, where video and audio equipment need to be connected to a computer system.

7.6 Serial Storage Architecture (SSA)

SSA describes the physical interface layer for the Serial Storage Architecture. SSA was first developed by IBM. SSA is capable of supporting 128 devices at a maximum data transfer rate of 80 MBps over a distance of 20 meters on a copper wire and 600 meters using fiber optics.

SSA is a serial link designed especially for low-cost high-performance connections between I/O devices. SSA uses a cable containing two differential-pair wires providing a two-signal connection. One pair may be used for sending data, while the other pair may be used to receive data, making SSA a true providing full-duplex architecture.

SSA features some specifications that make it well suited for high-availability environments:

- Auto-configuration of devices, including a flexible addressing scheme that allows connections as strings, loops, and switched loops. Hot swapping,

the insertion or removal of nodes without switching the network, is also permitted.
- No single point of failure.
- High level error detection and error handling protocols.

Of the four SCSI physical interface layers, SSA provides the best trade-off between cost, performance, availability and expandability. This will probably make SSA the most suitable physical interface layer for PC servers.

Appendix A. Details on the IDE Interface

This appendix contains details on the (E-)IDE disk subsystem technology.

A.1 IDE Connector Layout

Table 18 shows the layout for the IDE connector and cable.

Table 18 (Page 1 of 2). IDE 40-Pin Connector (and IDE SFF 44-Pin Connector)

Pin	I/O	Signal name	Description
1	I	-RESET	Reset all IDE devices
2	N/A	Ground	
3	I/O	DD7	Device Data bit 7
4	O	DD8	Device Data bit 8
5	I/O	DD6	Device Data bit 6
6	O	DD9	Device Data bit 9
7	I/O	DD5	Device Data bit 5
8	O	DD10	Device Data bit 10
9	I/O	DD4	Device Data bit 4
10	O	DD11	Device Data bit 11
11	I/O	DD3	Device Data bit 3
12	O	DD12	Device Data bit 12
13	I/O	DD2	Device Data bit 2
14	O	DD13	Device Data bit 13
15	I/O	DD1	Device Data bit 1
16	O	DD14	Device Data bit 14
17	I/O	DD0	Device Data bit 0
18	O	DD15	Device Data bit 15
19	NA	Ground	
20	NA	(key pin)	Key pin, to force correct insertion of the cable.
21	O	DMARQ	DMA Request. Used for handshaking during DMA transfers.
22	NA	Ground	

© Copyright IBM Corp. 1995

Table 18 (Page 2 of 2). IDE 40-Pin Connector (and IDE SFF 44-Pin Connector)

Pin	I/O	Signal name	Description
23	I	-DIOW	Device I/O Write. Used for handshaking during PIO transfers.
24	NA	Ground	
25	I	-DIOR	Device I/O Read. Used for handshaking during PIO transfers.
26	NA	Ground	
27	O	IORDY	I/O Ready. Used to allow/disallow access to the device registers.
28	I/O	SPSYNC or CSEL	Used for Spindle Synchronization (to synchronize two identical drives connected to the same IDE cable) or Cable Select (to have the IDE cable determine the device ID).
29	I	DMACK	DMA Acknowledge. Used for handshaking during DMA transfers.
30	NA	Ground	
31	O	INTRQ	Interrupt Request. Used by the device to tell the host system it should read the device registers.
32	O	-IOCS16	Used during 16 bit transfers
33	I	DA1	Device Address bit 1. Used to select a register.
34	I/O	-PDIAG	Device has Passed Diagnostics.
35	I	DA0	Device Address bit 0. Used to select a register.
36	I	DA2	Device Address bit 2. Used to select a register.
37	I	-CS0	Chip Select 0. Used to select a register block.
38	I	-CS1	Chip Select 1. Used to select a register block.
39	O	-DASP	Device Active, Device Present. Used to indicate the presence of a slave drive during startup. Otherwise it is used to indicate drive activity.
40	NA	Ground	
41	PWR	+5 V	**1** Power for the logic of the device.
42	PWR	+5 V	**1** Power for the motor of the device.
43	NA	Ground	**1**
44	I	-TYPE	**1** The type of the connected device (0 = ATA).

Note: I/O Field: I = input to device, O = output from device.

1 This pin is only used in the 44-pin IDE SFF connector, which is used in 2 1/2″ (and smaller) devices for portable computers.

The recommended connector and cable part numbers are :

- **Connector** 3M 3417-7000 or equivalent
- **Strain relief** 3M 3448-2040 or equivalent
- **Flat cable** 3M 3365-40 or 3M 3517-40 or equivalent (Max. length 46 cm)

A.2 IDE Protocol Overview

This section provides an overview of the IDE protocols.

- **PIO Read**

  1. The host sets up the register block on the controller.
  2. The host sends a command to the controller to begin execution.
  3. A sector is copied from the medium to the sector buffer on the controller.
  4. The host checks the status of the transfer.
  5. The host copies the sector buffer to system memory.
  6. If all sectors have been read, the command finishes. If more sectors need to be read, the procedure continues with step 3.

- **PIO Write**

  1. The host sets up the register block on the controller.
  2. The host sends a command to the controller to begin execution.
  3. The host copies data from the system memory to the device sector buffer.
  4. Data is read from the sector buffer and written on the device.
  5. The host system checks the status of the write operation.
  6. If all data has been written, the command finishes. If more sectors need to be written to the device, the procedure continues with 3.

- **Commands without data transfer**

  1. The host sets up the register block on the controller.
  2. The host sends a command to the controller to begin execution.
  3. The controller executes the command.

- **DMA Commands**. The procedure for DMA commands is about the same as for the PIO commands. The differences are :

  – The host system needs to set up a DMA channel first.
  – The host system only checks the status of the data transfer after all data has been transferred.

- **Other Commands**. The protocol for these commands depends on the command. Mostly, the same procedure will be used as for the commands without data transfer.

When the IDE controller needs to send a message to the host system, the following procedure is used :

1. The controller sets up the register blocks on the controller to indicate the message.
2. The controller sends an interrupt signal to the host system.

3. The host system examines the register blocks on the controller to determines what is has to do.

A.3 IDE Registers

ATA/IDE Addressing is controlled by the use of ten registers: eight *Command Block* registers and two *Control Block* registers. Command block registers are used for sending commands to the drive or receiving status from the drive. Control block registers are used for controlling the drive or giving alternate status. A short description of each register follows.

Two chip-select lines (-CS0 and -CS1) and three address lines (A0, A1, and A2) are used to select registers. The I/O address is decoded by the system board to determine which chip select signal is driven active.

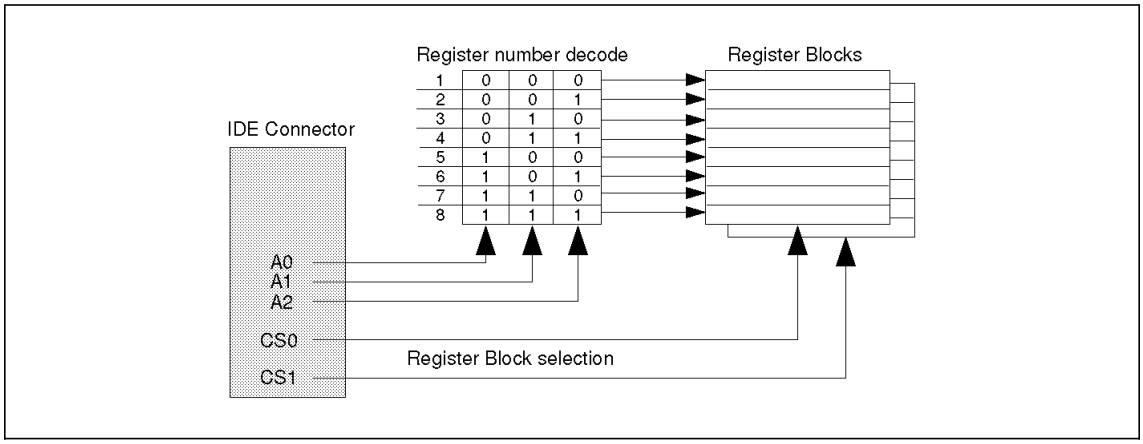


Figure 33. IDE Addressing Scheme

Figure 33 shows how this addressing method works.

Table 19 shows the exact relationship between the I/O address, the state of the chip select signals, and the register that is selected.

Table 19. IDE Register Overview

Register	**Read/ Write**	**I/O Addr. (Hex)**	**Chip Select Signals**		**Address Signals**		
			-CS0	**-CS1**	**A2**	**A1**	**A0**
Data	R/W	01F0	L	H	L	L	L
Error	R	01F1	L	H	L	L	H
Features	W	01F1	L	H	L	L	H
Sector count	R/W	01F2	L	H	L	H	L
Sector number	R/W	01F3	L	H	L	H	H
Cylinder low	R/W	01F4	L	H	H	L	L
Cylinder high	R/W	01F5	L	H	H	L	H
Drive/head	R/W	01F6	L	H	H	H	L
Status	R	01F7	L	H	H	H	H
Command	W	01F7	L	H	H	H	H
Alternate status	R	03F6	H	L	H	H	L
Device control	W	03F6	H	L	H	H	L
Drive address	R	03F7	H	L	H	H	H

Note:

L = Low level signal
H = High level signal

A.4 E-IDE Command Set Description

Table 20 lists all the commands in the E-IDE command set.

Table 20 (Page 1 of 2). IDE Command Set

Command Name	Type	Description
Acknowledge Media Change	O	Used by the OS to acknowledge a media change in the device
Boot (Post-Boot)	O	Signals that the host system has finished its booting sequence
Boot (Pre-Boot)	O	Signals that the host system is going to start its boot sequence
Check Power Mode	O	Determine the current power savings mode of the device
Door Lock	O	Locks the door of the removable media device
Door Unlock	O	Unlocks the door of a removable media device
Download Microcode	O	Permanently or temporarily download a new version of the microcode to the device
Execute Device Diagnostic	M	Perform internal diagnostic tests
Format Track	V	Initialize the media for usage
Identify Device	M	Retrieve device configuration information (see Table 21 on page 100)
Idle	O	Set device into Idle power savings mode, after a specified amount of time (see Table 24 on page 104)
Idle Immediate	O	Immediately set device into idle power savings mode
Initialize Device Parameters	M	Specify device geometry
Media Eject	O	Eject the media in the removable media device
Nop	O	Do nothing, allows host system to check device status
Read Buffer	O	Read the contents of the sector buffer
Read DMA	O	Read data from the device and transport the data to the host using a DMA channel
Read Long	O	Read a sector of data from the device, including its ECC codes
Read Multiple	O	Read sectors from the device and interrupt the host after all sectors have been read
Read Sector(s)	M	Read sectors from the device and interrupt the host after each sector has been read
Read Verify Sector(s)	M	Check the integrity and readability of sectors on the device

Table 20 (Page 2 of 2). IDE Command Set

Command Name	Type	Description
Recalibrate	O	Move the read/write heads to a predetermined location
Seek	M	Move the read/write heads to a specified location
Set Features	O	Set device configuration
Set Multiple Mode	O	Specify the number of sectors involved in Read Multiple and Write Multiple commands
Sleep	O	Set the drive to sleep power savings mode
Standby	O	Set the drive to standby power savings mode, after a specified amount of time
Standby Immediate	O	Immediately set the drive to standby power savings mode
Write Buffer	O	Write data to the sector buffer
Write DMA	O	Write data to the device, using a DMA channel to transport the data from the host system to the device
Write Long	O	Write data to the device, including the given ECC codes
Write Multiple	O	Write multiple sectors of data, interrupting the host only after all sectors have been written
Write Same	O	Write the same data multiple times
Write Sector(s)	M	Write data to the device, interrupting the host after each sector has been written
Write Verify	O	Write data to the device and verify if the written data is readable

Note:

Type M = Mandatory command
Type O = Optional command
Type V = Vendor Specific

A.5 E-IDE Device Identification Information

Table 21 lists the information that is send to the host system as the result of a Identify Device command. Notice that 255 words of information are involved. Since each word contains two bytes, a total of 512 bytes are transmitted to the host system.

Table 21 (Page 1 of 3). E-IDE Device Identification Information

Word	Description
0	General configuration information: **bit 15** 0=ATA device, 1=ATAPI device **bit 7** 1=removable media device **bit 6** 1=device can not be removed **others** obsolete or reserved
1	Number of logical cylinders on device
2	Reserved
3	Number of logical heads on device
4-5	Vendor unique
6	Number of logical sectors per logical track on device
7-9	Vendor unique
10-19	Device serial number (20 ASCII characters)
20-21	Vendor unique
22	# of ECC bytes on Read Long and Write Long commands
23-26	Device firmware revision (8 ASCII characters)
27-46	Device model number (40 ASCII characters)
47	Device capabilities: **bits 15-8** Vendor unique **bits 7-0** 00h = Read Multiple and Write Multiple commands not implemented xxh = Max # of sectors that can be transferred per interrupt on Read Multiple and Write Multiple. commands
48	Vendor unique

Table 21 (Page 2 of 3). E-IDE Device Identification Information

Word	Description
49	Device capabilities: **15-14** Reserved **13** 0 = Vendor specific implementation of Standby Timer implemented 1 = standard implementation **12** Reserved (for advanced transfer mode support) **11** 0 = IORDY may be supported 1 = IORDY supported **10** 1 = IORDY can be disabled **9** 1 = LBA addressing scheme supported **8** 1 = DMA data transfers supported **7-0** Vendor unique
50	Reserved
51	**15-8** PIO data transfer cycle timing mode **7-0** Vendor unique
52	**15-8** DMA data transfer cycle timing mode **7-0** Vendor unique
53	**15-2** Reserved **1** 1 = Fields reported in words 64-70 are valid **0** 1 = Fields reported in words 54-58 are valid
54	Number of current logical cylinders
55	Number of current logical heads
56	Number of current logical sectors per logical track
57-58	Current capacity in logical sectors
59	**15-9** Reserved **8** 1 = Multiple sector setting is valid (for R/W multiple commands) **7-0** Max # sectors transferred per R/W multiple commands
60-61	Total number of user addressable sectors (LBA mode only)
62	Single-Word DMA data transfer mode specifications: **15-8** Indicate which mode is active **7-0** Indicate which modes are supported by the device
63	Multi-Word DMA data transfer mode specifications: **15-8** Indicate which mode is active **7-0** Indicate which modes are supported by the device
64	PIO data transfer mode specifications: **15-8** Reserved **7-0** Indicate which modes are supported by the device
65	Minimum Multiword DMA Transfer Cycle Time Per Word (in ns)

Table 21 (Page 3 of 3). E-IDE Device Identification Information

Word	Description
66	Manufacturer's Recommended Multiword DMA Transfer Cycle time (in ns)
67	Minimum PIO Transfer Cycle Time Without Flow Control (in ns)
68	Minimum PIO Transfer Cycle Time With IORDY Flow Control (in ns)
69-72	Reserved (for advanced PIO mode support)
73	Indicate which major ATA versions are supported by the device
74	Indicate which minor ATA versions (revisions) are supported by the device
75-127	Reserved
128-159	Vendor unique
160-255	Reserved

A.6 Set Features Command

Table 22 lists the features that can be set using the set features command.

Set Features Specifications

Table 22. IDE

Value	Description
01h	Enable 8-bit data transfers
02h	Enable write cache [1]
03h	Set transfer mode based on value in Sector Count register
33h	Disable retry [1]
44h	Vendor unique length of ECC on READ LONG/WRITE LONG commands
54h	Set cache segments to Sector Count register value [1]
55h	Disable read look-ahead feature
66h	Disable reverting to power on defaults
77h	Disable ECC [1]
81h	Disable 8-bit data transfers
82h	Disable write cache [1]
88h	Enable ECC [1]
99h	Enable retries [1]
AAh	Enable read look-ahead feature
ABh	Set maximum prefetch using Sector Count register value [1]
BBh	4 bytes of ECC apply on READ LONG/WRITE LONG commands
CCh	Enable reverting to power on defaults
Note: [1] This feature is vendor-specific	

A.7 IDE Power Management

Table 23 lists all the modes that are specified by the IDE standard.

Table 23. IDE Power Management Modes

Mode	Description
Active	The whole drive is powered as normal.
Idle	Some electronic parts of the device are turned off, allowing quick recovery to active mode.
Standby	Most parts of the device (including the device motor) are turned off, requiring a longer recovery to active mode.
Sleep	Almost all parts of the device do not receive power. A reset of the disk subsystem is needed to re-activate the device.

Table 24 lists the meaning of the values that can be programmed into the IDE timer.

Table 24. IDE Timer Values

Timer Value	Description
0	Timeout Disabled
1 - 240	(value * 5) seconds
241 - 251	((value - 240) * 30) minutes
252	21 minutes
253	Vendor unique period between 8 and 12 hours
254	Reserved
255	21 minutes 15 seconds

Appendix B. Details on the SCSI interface

This appendix contains details on the SCSI disk subsystem technology.

B.1 SCSI Bus Control Signals

Table 25 explains the function of the control signals in the SCSI cables.

Table 25. SCSI Bus Control Signals

Signal	Description
Acknowledge	Used for handshaking during data transfer
Attention	Used by initiator to indicate to the target that it needs attention
Busy	Indicates that the SCSI bus is being used
Control / Data	Indicates the type of data on the bus
Input / Output	Indicates the direction of the data flow
Message	Used to indicate that a device has a message for another device
Request	Used for handshaking during data transfer
Restart	Used to reset the disk subsystem
Select	Used by an initiator to select a device as its target

© Copyright IBM Corp. 1995

B.2 SCSI Internal A Cable Connector Layout

The layout of the internal SCSI A cable with IDC connector is given in Table 26. The same layout is used for the high density 50-pin SCSI-2 cable connector.

Table 26. Internal SCSI A Cable Pin Layout

Pin	Signal Name	Pin	Signal Name
1	Ground	**26**	+5V Terminator
2	Data Bit (0)	**27**	Reserved
3	Ground	**28**	Reserved
4	Data Bit (1)	**29**	Ground
5	Ground	**30**	Ground
6	Data Bit (2)	**31**	Ground
7	Ground	**32**	-Attention
8	Data Bit (3)	**33**	Ground
9	Ground	**34**	Ground
10	Data Bit (4)	**35**	Ground
11	Ground	**36**	-Busy
12	Data Bit (5)	**37**	Ground
13	Ground	**38**	-Acknowledge
14	Data Bit (6)	**39**	Ground
15	Ground	**40**	-Restart
16	Data Bit (7)	**41**	Ground
17	Ground	**42**	-Message
18	Data Bit (P)	**43**	Ground
19	Ground	**44**	-Select
20	Ground	**45**	Ground
21	Ground	**46**	-Control / Data
22	Ground	**47**	Ground
23	Reserved	**48**	-Request
24	Reserved	**49**	Ground
25	(open)	**50**	-Input / Output

B.3 SCSI External A Cable Connector Layout

Table 27 lists the connector and cable layout for the external centronics-style SCSI A cable. The same layout is used for the high density 50-pin SCSI-2 cable connector.

Table 27. External SCSI A Cable Pin Layout

Pin	Signal Name	Pin	Signal Name
1	Ground	**26**	Data Bit (0)
2	Ground	**27**	Data Bit (1)
3	Ground	**28**	Data Bit (2)
4	Ground	**29**	Data Bit (3)
5	Ground	**30**	Data Bit (4)
6	Ground	**31**	Data Bit (5)
7	Ground	**32**	Data Bit (6)
8	Ground	**33**	Data Bit (7)
9	Ground	**34**	Data Bit (P)
10	Ground	**35**	Ground
11	Ground	**36**	Ground
12	Reserved	**37**	Reserved
13	(open)	**38**	+5V Terminator
14	Reserved	**39**	Reserved
15	Ground	**40**	Ground
16	Ground	**41**	-Attention
17	Ground	**42**	Ground
18	Ground	**43**	-Busy
19	Ground	**44**	-Acknowledge
20	Ground	**45**	-Restart
21	Ground	**46**	-Message
22	Ground	**47**	-Select
23	Ground	**48**	-Control / Data
24	Ground	**49**	-Request
25	Ground	**50**	-Input / Output

B.4 External 60-pin IBM Cable Connector Layout

The 60 pin SCSI connector was used by IBM on early SCSI implementations. During the early stages of the SCSI-2 development, this connector layout was used. Later SCSI-2 revisions did not specify this cable anymore. Table 28 lists the layout of this cable connector.

Table 28. IBM SCSI 60-Pin Cable Layout

Pin	Signal Name	Pin	Signal Name
1	Ground	**31**	Ground
2	Data Bit (0)	**32**	-Attention
3	Ground	**33**	Ground
4	Data Bit (1)	**34**	Ground
5	Ground	**35**	Ground
6	Data Bit (2)	**36**	-Busy
7	Ground	**37**	Ground
8	Data Bit (3)	**38**	-Acknowledge
9	Ground	**39**	Ground
10	Data Bit (4)	**40**	-Restart
11	Ground	**41**	Ground
12	Data Bit (5)	**42**	-Message
13	Ground	**43**	Ground
14	Data Bit (6)	**44**	-Select
15	Ground	**45**	Ground
16	Data Bit (7)	**46**	-Control / Data
17	Ground	**47**	Ground
18	Data Bit (Parity 0)	**48**	-Request
19	Ground	**49**	Ground
20	Ground	**50**	-Input / Output
21	Ground	**51**	Ground
22	Ground	**52**	Reserved
23	Ground	**53**	Reserved
24	Ground	**54**	Reserved
25	(open)	**55**	Reserved
26	+5V Terminator	**56**	Reserved
27	Ground	**57**	Reserved
28	Ground	**58**	Reserved
29	Ground	**59**	Reserved
30	Ground	**60**	Reserved

B.5 SCSI P Cable Connector Layout

Table 29 lists the pin layout for the SCSI P cable. Note that the same layout is used for internal and external connectors.

Table 29. SCSI P Cable Pin Layout

Pin	Signal Name	Pin	Signal Name
1	Ground	**35**	Data Bit (12)
2	Ground	**36**	Data bit (13)
3	Ground	**37**	Data bit (14)
4	Ground	**38**	Data Bit (15)
5	Ground	**39**	Data Bit (Parity 1)
6	Ground	**40**	Data Bit (0)
7	Ground	**41**	Data Bit (1)
8	Ground	**42**	Data Bit (2)
9	Ground	**43**	Data Bit (3)
10	Ground	**44**	Data Bit (4)
11	Ground	**45**	Data Bit (5)
12	Ground	**46**	Data Bit (6)
13	Ground	**47**	Data Bit (7)
14	Ground	**48**	Data Bit (Parity 0)
15	Ground	**49**	Ground
16	Ground	**50**	Ground
17	+5V Terminator	**51**	+5V Terminator
18	+5V Terminator	**52**	+5V Terminator
19	Reserved	**53**	Reserved
20	Ground	**54**	Ground
21	Ground	**55**	-Attention
22	Ground	**56**	Ground
23	Ground	**57**	-Busy
24	Ground	**58**	-Acknowledge
25	Ground	**59**	-Reset
26	Ground	**60**	-Message
27	Ground	**61**	-Select
28	Ground	**62**	-Control/Data
29	Ground	**63**	-Request
30	Ground	**64**	-Input/Output
31	Ground	**65**	Data Bit (8)
32	Ground	**66**	Data Bit (9)
33	Ground	**67**	Data Bit (10)
34	Ground	**68**	Data Bit (11)

B.6 Basic SCSI Command Set

Table 30 lists and briefly describes all commands that are a part of the basic SCSI command set.

Table 30. Basic SCSI Command Set

Command Name	Type	Description
Change Definition	O	Set device operating mode to a certain SCSI version
Compare	O	Compare two blocks of data
Copy	O	Copy data from one device to another
Copy and Verify	O	Copy data from one device to another, then compare the copied data with the original to verify the correctness and integrity
Inquiry	M	Query information about the target device, such as : • Device Type • Device Vendor • Ability to support Wide transfers
Log Select	O	Retrieve statistics from device
Log Sense	O	Retrieve information about the statistics kept by the device
Mode Select (6 and 10)	O	Configure the target device (6 and 10 byte versions)
Mode Sense (6 and 10)	O	Query target device configuration (6 and 10 byte versions)
Read Buffer	O	Used for diagnostics of device and SCSI bus
Receive Diagnostic Results	O	Request diagnostics results from the target
Request Sense	M	Retrieve information about an error condition
Send Diagnostic	M	Target should perform self-diagnostics
Test Unit Ready	M	Check if the target device is ready to accept commands
Write Buffer	O	Used for diagnostics of device and SCSI bus and for down-loading microcode

Note:

Type M = Mandatory command
Type O = Optional command

B.7 SCSI Command Set for DASD

A short description of all commands that should be or may be implemented by SCSI direct access storage devices is given in Table 31.

Table 31. SCSI Command Set for DASD

Command Name	Type	Description
Format Unit	M	Initialize the device
Lock-Unlock Cache	O	Allow or disallow blocks to be removed from the device cache
Pre-Fetch	O	Move requested data to device cache
Prevent-Allow Medium Removal	O	Allow or disallow removal of the medium of the device
Read (6 and 10)	M	Transfer data from target to initiator (6 and 10 byte versions)
Request Capacity	M	Query device capacity (returns # of blocks and bytes per block)
Read Defect Data	O	Request the medium defect lists from the target
Read Long	O	Read data including its ECC
Reassign Blocks	O	Relocate data in defective sectors to spare sectors
Release	M	Allow access to the target by other devices
Reserve	M	Disallow access to the target by other devices
Rezero Unit	O	Return device to a specific state
Search Data	O	Try to find matches or mismatches in data
Seek (6 and 10)	O	Move the head assembly to the specified logical block address
Set Limits	O	Specify limits for linked commands
Start-Stop Unit	O	Enable or disable media access operations on the target
Synchronize Cache	O	Ensure that the cache contents are equal to the medium contents
Verify	O	• Verify integrity of stored data • Check if sector is defective
Write (6 and 10)	O	Transfer data from initiator to target
Write and Verify	O	Perform Write and then perform Verify
Write Long	O	Write data, including the given ECC
Write Same	O	Write the same data multiple times

Note:

Type M = Mandatory command
Type O = Optional command

Glossary

A

actuator. The part of a DASD that moves the read/write head over the medium.

arbitration. A method with which multiple devices attached to a single bus can bid to get control of that bus.

asynchronous mode. A mode of data transfer across the SCSI Bus where each byte of data transferred must be acknowledged as received by the target before the next byte can be sent.

average access time. The average amount of time that it takes the hard disk to get to any location on the disk. It is the sum of the average seek time and the latency of the disk.

B

block size. The size of the smallest amount of data on a device.

buffer. Memory used to temporarily store data

buffer fill rate. The rate at which a device can fill its buffer.

bus master. An intelligent device that can bid for and gain control of the host system bus to perform a specific task.

C

cache. A large buffer that is used to enhance the performance of devices with relatively slow response times

CD-ROM. A Compact Disk Read-Only-Media is a disc that:

- requires optical technology to be read
- can not be written to

clock speed. The number of signals that can be transmitted in a certain amount of time.

command set layer. The part of the system-level interface that lists all the commands that a device must be able to understand

Common Command Set. A set of SCSI commands that is specified in the ANSI standard that all SCSI device must be able to use in order to be fully compatible with the ANSI standard.

connect. A part of the SCSI protocol that is used to establish a link between two devices.

cylinder. The collection of all the tracks that are at the same location of each disk in a device.

D

Data Transfer Rate (DTR). Indicates how fast the disk subsystem can move data to or from the system memory of a computer.

device model layer. The part of the system-level interface that describes the general layout and behavior of types of devices

defect management. The methods used by a device to handle medium defects.

device-level interface. An interface that is very dependent on the specifications of the device (or peripheral) attached to the interface.

diagnostics. A set of procedures to verify the functioning of a device.

disconnect. A part of the SCSI protocol that allows a device to disconnect from the SCSI Bus while it is processing data or a command, to enable other devices to use the SCSI Bus.

© Copyright IBM Corp. 1995

disk geometry. A description of the physical characteristics of a drive (number of read/write heads, number of cylinders, number of tracks, etc.)

disk subsystem. The disk subsystem provides permanent mass storage to computer system and is responsible for maintaining the integrity of the data while it is stored on the mass storage devices.

disk subsystem controller. The part of a disk subsystem that is responsible for the actions of the whole disk subsystem.

direct access storage device (DASD). A device in which access time to the data is effectively independent of the location of the data.

Direct Memory Access (DMA). A method used to transfer data directly from device to system memory without using the main system processor.

drop. A device connector

F

fault-tolerance. The ability of a system to continue operating even though one ore more of its components is malfunctioning

FIFO Buffer. A buffer from which the data that has been stored first in the buffer is also the data that will leave the buffer first.

firmware. Software that is a part of a device

format. The process used by a device to initialize its medium

H

hard disk. A device that uses magnetic methods for storing and retrieving data from its medium.

head. The part of a device that actually reads and writes data from the medium.

hot spare device. A device that is installed in a computer system, but which will not be used until another device fails.

hot swap device. A device that may be removed from or inserted into a computer system without having to shutdown the computer.

I

initiator. A device attached to the SCSI Bus that wants to communicate with another device (the target).

J

jumper. A small device to connect two pins together. Jumpers are used to configure certain characteristics of the device, such as the device ID.

L

latency. The time required for the right sector on a track to come under the read/write head.

M

Mean-Time-Between-Failure (MTBF). The average time between consecutive failures of a component.

Mean-Time-to-Data-Loss (MTDL). Defines the average time until data loss occurs. When data loss occurs, the lost data is not available anymore to the end user.

medium. The part of a storage device on which data is physically stored.

O

optical device. A device that uses optical (usually laser) methods to store and/or retrieve data from its medium.

P

parity check. Used to check the integrity of data

peripheral. A device that attaches to a computer system to enhance the capabilities of the system.

physical interface layer. The part of the system-level interface that describes the cabling, connectors and electrical characteristics.

power management. The ability of a device to control the amount of power used by the device.

power management feature set. The set of features used by an IDE device for power management.

protocol layer. The part of the system-level interface that specifies how devices that are physically connected should communicate with each other.

R

reconnect. A part of the SCSI protocol that is used by a target device to re-establish the connection with its initiator.

redundancy. The use of multiple components in the same system that are able to perform the same function.

rotation speed. Rotation speed of the disk drive, measured in revolutions-per-minute SCSI Bus (rpm).

S

SCSI bus. The means of transporting data between several SCSI devices.

SCSI ID. A number given to a device to be able to uniquely identify that device on the SCSI Bus.

sector. The smallest addressable unit of data on a device

sector buffer. Buffer used by a device to temporarily store data from one or more sectors on the medium of the device.

seek. To selectively position the access mechanism of a direct access device.

seek time. Time required to selectively position the access mechanism of a direct access device.

server. A computer system that provides shared services to workstations over a network

subsystem. A part of the whole computer system.

switch. A mechanical part of a device used to configure certain aspects of the device. To configure an aspect, the switch is switched into one of several possible settings.

synchronous. A mode of data transfer across the SCSI Bus where each byte of data transferred does not have to be acknowledged as received by the target device before the next byte can be sent.

system-level interface. An interface that uses several layers to provide the system some degree of independence from the peripheral attached to the interface.

T

target. A device attached to the SCSI Bus that receives and processes commands sent from another device on the SCSI Bus. The device that sends the command is known as an Initiator.

terminator. A resistance used to dampen resonance in a cable, so the quality of the signals in the cable remain at a high level.

track. The collection of sectors on one disk that can be accessed without moving the head of the device.

W

workstation. A computer system that uses shared services over a network.

List of Abbreviations

APA	All Points Addressable
ANSI	Ammerican National Standards Institute
ARLL	Advanced Run Length Limited
ASPI	Advanced SCSI Programming Interface
CAM	Common Access Methods
CPU	Central Processing Unit
DASD	Direct Access Storage Device
DTR	Data Transfer Rate
E-IDE	Enhanced-Integrated Drive Electronics
ESDI	Enhanced Small Device Interface
FC	Fiber Channel
FCP	Fiber Channel Protocol
FC-AL	Fiber Channel-Arbitrated Loop
Hz	Herz
IBM	International Business Machines Corporation
ICA	Industry Standard Architecture
IDE	Intergrated Drive Electronics
ITSO	International Technical Support Organization
LAN	Local Area Network
MCA	Micro Channel Architecture
MFM	Modified Frequency Modulation
MMC	SCSI-3 MultiMedia Commands
MTBF	Mean Time Between Failures
MTDL	Mean Time to Data Loss
MTR	Media Transfer Rate
PC	Personal Computer
PCI	Peripheral Component Interconnect
PIO	Programmed Input/Output
PMFS	Power Management Feature Set
PROFS	Professional Office System
RAID	Redundant Array of Inexpensive Disks
RAM	Random Access Memory
RLL	Run Length Limited
SAM	SCSI-3 Architecture Model
SBC	SCSI-3 Block Commands
SBP	Serial Bus Protocol
SCC	SCSI-3 Controller Commands
SCSI	Small Computer System Interface
SGC	SCSI-3 Graphics Commands
SIP	SCSI-3 Interlocked Protocol
SMC	SCSI-3 Medium Changer Commands
SPC.	SCSI-3 Primary Commands
SSA	Serial Storage Architecture
SSC	SCSI-3 Stream Commands
SSP	Serial Storage Protocol
ST506	Seagate Technology ST506

© Copyright IBM Corp. 1995

Index

© Copyright IBM Corp. 1995

E

F

H

I

K

L

M

N

O

P

R

S

T

W

Z

Printed in U.S.A.