Chapter 14

Hard Disk Drives and Controllers

To most users, the hard disk drive is the most important, yet most mysterious, part of a computer system. A *hard disk drive* is a sealed unit that holds the data in a system. When the hard disk fails, the consequences usually are very serious. To maintain, service, and expand a PC system properly, you must fully understand the hard disk unit.

Most computer users want to know how hard disk drives work and what to do when a problem occurs. Few books about hard disks, however, cover the details necessary for the PC technician or sophisticated user. This chapter corrects that situation.

This chapter thoroughly describes the hard disk, from the drives to the cables and controllers that run them. In particular, the chapter examines the construction and operation of a hard disk drive. You will learn about the various disk interfaces that you can select, and the shortcomings and strengths of each type, as well as the procedures for configuring, setting up, and installing various drives. The chapter introduces the basic procedures necessary to integrate a hard disk drive into a PC system.

Definition of a Hard Disk

A hard disk drive contains rigid, disc-shaped platters usually constructed of aluminum or glass. Unlike floppy disks, the platters cannot bend or flex—hence the term *hard disk*. In most hard disk drives, the platters cannot be removed; for that reason, IBM calls them *fixed disk drives*. Although removable-platter hard disk drives are available, their non-standard nature, higher cost, and reliability problems make them unpopular.

Hard disk drives often are called *Winchester drives*. This term dates back to the 1960s, when IBM developed a high-speed hard disk drive that had 30MB of fixed-platter storage and 30MB of removable-platter storage. The drive had platters that spun at high speeds and heads that floated over the platters while they spun in a sealed environment. That drive, the *30-30 drive*, soon received the nickname "Winchester," after the famous

Winchester 30-30 rifle. After that time, drives that used a high-speed spinning platter with a floating head also became known as Winchester drives. The term has no technical or scientific meaning; it is a slang term, synonymous with *hard disk*.

Hard Drive Advancements

In the 10 or more years that hard disks have commonly been used in PC systems, they have undergone tremendous changes. To give you an idea of how far hard drives have come in that time, the following are some of the most profound changes in PC hard disk storage:

- Maximum storage capacities have increased from the 10MB 5.25-inch full-height drives available in 1982 to 3GB or more for drives in that form factor and well over 2GB for small 3.5-inch half-height drives.
- Data transfer rates have increased from 102KB per second for the original IBM XT in 1983 to nearly 10MB per second for some of the fastest drives today.
- Average seek times have decreased from more than 85 ms for the 10MB XT hard disk in 1983 to fewer than 8 ms for some of the fastest drives today.
- In 1982, a 10MB drive cost more than $1,500 ($150 per megabyte). Today, the cost of hard drives has dropped to less than $1 per megabyte.

Areal Density

Areal density has been used as a primary technology-growth-rate indicator for the hard disk drive industry. *Areal density* is defined as the product of the linear bits per inch (BPI), measured along the length of the tracks around the disk, multiplied by the number of tracks per inch (TPI) measured radially on the disk. The results are expressed in units of Mb/sq-inch and are used as a measure of efficiency in drive recording technology. Current high-end drives record at areal densities of about 160 Megabits per square inch (Mb/sq-inch). Prototype models with densities as high as 1 Gb/sq-inch have been constructed, allowing for capacities of more than 1GB on a single 3.5-inch platter.

Areal density (and, therefore, drive capacity) has been doubling approximately every two to three years, and disk drives are likely to reach areal densities of 10 Gb/sq-inch before the year 2000. A drive built with this technology would be capable of storing 1GB of data on a single 1.3-inch platter, and the entire drive would be about the size of a large wristwatch. New media and head technologies, such as ceramic or glass platters and fluid suspension, are being developed to support these higher areal densities. Manufacturing drive heads and disks to operate at closer tolerances is the primary challenge in achieving higher densities.

It seems almost incredible that computer technology improves by doubling performance or capacity every two to three years. If only other industries could match that growth and improvement rate!

Hard Disk Drive Operation

The basic physical operation of a hard disk drive is similar to that of a floppy disk drive: a hard drive uses spinning disks with heads that move over the disks and store data in tracks and sectors. In many other ways, however, hard disk drives are different from floppy disk drives.

Hard disks usually have multiple platters, each with two sides on which data can be stored. Most drives have at least two or three platters, resulting in four or six sides, and some drives have up to 11 platters. The identically positioned tracks on each side of every platter together make up a *cylinder*. A hard disk drive has one head per platter side, and all the heads are mounted on a common carrier device, or *rack*. The heads move in and out across the disk in unison; they cannot move independently because they are mounted on the same rack.

Hard disks operate much faster than floppy drives. Most hard disks spin at 3,600 rpm—approximately 10 times faster than a floppy drive. Until recently, 3,600 rpm was pretty much a constant among hard drives. Now, however, quite a few hard drives spin even faster. One drive that I have, for example, spins at 4,318 rpm; others spin as fast as 5,600, 6,400, and even 7,200 rpm. High rotational speed combined with a fast head-positioning mechanism and more sectors per track make one hard disk faster than another, and all these features combine to make hard drives much faster than floppy drives in storing and retrieving data.

The heads in a hard disk do not (and should not!) touch the platters during normal operation. When the heads are powered off, however, they land on the platters as they stop spinning. While the drive is on, a cushion of air keeps each head suspended a short distance above or below the platter. If the cushion is disturbed by a particle of dust or a shock, the head may come into contact with the platter spinning at full speed. When contact with the spinning platters is forceful enough to do damage, the event is called a *head crash*. The result of a head crash may be anything from a few lost bytes of data to a totally trashed drive. Most drives have special lubricants on the platters and hardened surfaces that can withstand the daily “takeoffs and landings” as well as more severe abuse.

Because the platter assemblies are sealed and nonremovable, track densities can be very high. Many drives have 3,000 or more tracks per inch of media. Head Disk Assemblies (HDAs), which contain the platters, are assembled and sealed in clean rooms under absolutely sanitary conditions. Because few companies repair HDAs, repair or replacement of items inside a sealed HDA can be expensive. Every hard disk ever made will fail eventually. The only questions are when the hard disk will fail and whether your data is backed up.

Many PC users think that hard disks are fragile, and generally, they are one of the most fragile components in your PC. In my weekly PC Hardware and Troubleshooting or Data Recovery seminars, however, I have run various hard disks for days with the lids off, and have even removed and installed the covers while the drives were operating. Those drives continue to store data perfectly to this day with the lids either on or off.

Of course, I do not recommend that you try this test with your own drives; neither would I use it on my larger, more expensive drives.

The Ultimate Hard Disk Drive Analogy!

I'm sure that you have heard the traditional analogy that compares the interaction of the head and media in a typical hard disk as being similar in scale to a 747 flying a few feet off the ground at cruising speed (500-plus mph). I have heard this analogy used over and over again for years, and I've even used it in my seminars many times without checking to see whether the analogy is technically accurate with respect to modern hard drives.

One highly inaccurate aspect of the 747 analogy has always bothered me: the use of an airplane of any type to describe the head-and-platter interaction. This analogy implies that the heads fly very low over the surface of the disk—but technically, this is not true. The heads do not fly at all, in the traditional aerodynamic sense; instead, they float on a cushion of air that's being dragged around by the platters. A much better analogy would use a hovercraft instead of an airplane; the action of a hovercraft much more closely emulates the action of the heads in a hard disk drive. Like a hovercraft, the drive heads rely somewhat on the shape of the bottom of the head to capture and control the cushion of air that keeps them floating over the disk. By nature, the cushion of air on which the heads float forms only in very close proximity to the platter and is called an *air bearing* by the disk drive industry.

I thought it was time to come up with a new analogy that more correctly describes the dimensions and speeds at which a hard disk operates today. I looked up the specifications on a specific hard disk drive, and then equally magnified and rescaled all the dimensions involved to make the head floating height equal to 1 inch. For my example, I used a Seagate model ST-12550N Barracuda 2 drive, which is a 2GB (formatted capacity), 3.5-inch SCSI-2 drive. In fact, I intend to install this very drive in the P75 Portable on which I am writing this book, replacing the current 1.2GB drive. Table 14.1 shows the specifications of this drive, as listed in the technical documentation:

Table 14.1 Seagate ST-12550N Barracuda 2 2G, 3.5-Inch, SCSI-2 Drive Specifications

Specification	Value	Unit of Measure
Linear density	52,187	Bits per inch (bpi)
Bit spacing	19.16	micro-inches (μ-in)
Track density	3,047	Tracks per inch (tpi)
Track spacing	328.19	micro-inches (μ-in)
Total tracks	2,707	tracks
Rotational speed	7,200	Revolutions per minute (rpm)
Average head linear speed	53.55	Miles per hour (mph)
Head slider length	0.08	inches
Head slider height	0.02	inches
Head floating height	5	micro-inches (μ-in)
Average seek time	8	milliseconds (ms)

By interpreting these specifications, you can see that in this drive. the heads are about 0.08 inch long and 0.02 inch high. The heads float on a cushion of air about 5 micro-inches (millionths of an inch) from the surface of the disk while traveling at an average speed of 53.55 mph (figuring an average track diameter of 2.5 inches). These heads read and write individual bits spaced only 19.16 micro-inches apart on tracks separated by only 328.19 micro-inches. The heads can move from one track to any other in only 8 ms during an average seek operation.

To create my new analogy, I simply magnified the scale to make the floating height equal to 1 inch. Because 1 inch is 200,000 times greater than 5 micro-inches, I scaled up everything else by the same amount.

Are you ready?

The heads of this "typical" hard disk, magnified to such a scale, would be more than 1,300 feet long and 300 feet high (about the size of the Sears Tower, lying sideways!), traveling at a speed of more than 10.7 million mph(178,500 miles per second!) only 1 inch above the ground, reading data bits spaced a mere 3.83 inches apart on tracks separated by only 5.47 feet. Additionally, each skyscraper-size head could move sideways to any track within a distance of 2.8 miles in an average 8 ms, resulting in an average sideways velocity of about 1,402 mph!

The speed of this imaginary head is difficult to comprehend, so I'll elaborate. The diameter of the Earth at the equator is 7,926 miles, which means a circumference of 24,900 miles. At 178,500 miles per second, this imaginary head would circle the Earth more than seven times per second!

This analogy should give you a new appreciation of the technological marvel that the modern hard disk drive actually represents. It makes the 747 analogy look rather pathetic (not to mention totally inaccurate), doesn't it?

Magnetic Data Storage

Learning how magnetic data storage works will help you develop a feel for the way that your disk drives operate and can improve the way that you work with disk drives and disks.

Nearly all disk drives in personal computer systems operate on magnetic principles. Purely optical disk drives often are used as a secondary form of storage, but the computer to which they are connected is likely to use a magnetic storage medium for primary disk storage. Due to the high performance and density capabilities of magnetic storage, optical disk drives and media probably never will totally replace magnetic storage in PC systems.

Magnetic drives, such as floppy disk drives and hard disk drives, operate by using *electromagnetism.* This basic principle of physics states that as an electric current flows through a conductor, a magnetic field is generated around the conductor. This magnetic field then can influence magnetic material in the field. When the direction of the flow of electric current is reversed, the magnetic field's polarity also is reversed. An electric motor uses electromagnetism to exert pushing and pulling forces on magnets attached to a rotating shaft.

Another effect of electromagnetism is that if a conductor is passed through a changing magnetic field, an electrical current is generated. As the polarity of the magnetic field changes, so does the direction of the electric current flow. For example, a type of electrical generator used in automobiles, called an *alternator*, operates by rotating electromagnets past coils of wire conductors in which large amounts of electrical current can be induced. The two-way operation of electromagnetism makes it possible to record data on a disk and read that data back later.

The read/write heads in your disk drives (both floppy and hard disks) are U-shaped pieces of conductive material. This U-shaped object is wrapped with coils of wire, through which an electric current can flow. When the disk drive logic passes a current through these coils, it generates a magnetic field in the drive head. When the polarity of the electric current is reversed, the polarity of the field that is generated also changes. In essence, the heads are electromagnets whose voltage can be switched in polarity very quickly.

When a magnetic field is generated in the head, the field jumps the gap at the end of the U-shaped head. Because a magnetic field passes through a conductor much more easily than through the air, the field bends outward through the medium and actually uses the disk media directly below it as the path of least resistance to the other side of the gap. As the field passes through the media directly under the gap, it polarizes the magnetic particles through which it passes so that they are aligned with the field. The field's polarity and, therefore, the polarity of the magnetic media are based on the direction of the flow of electric current through the coils.

The disk consists of some form of substrate material (such as Mylar for floppy disks or aluminum or glass for hard disks) on which a layer of magnetizable material has been deposited. This material usually is a form of iron oxide with various other elements added. The polarities of the magnetic fields of the individual magnetic particles on an erased disk normally are in a state of random disarray. Because the fields of the individual particles point in random directions, each tiny magnetic field is canceled by one that points in the opposite direction, for a total effect of no observable or cumulative field polarity.

Particles in the area below the head gap are aligned in the same direction as the field emanating from the gap. When the individual magnetic domains are in alignment, they no longer cancel one another, and an observable magnetic field exists in that region of the disk. This local field is generated by the many magnetic particles that now are operating as a team to produce a detectable cumulative field with a unified direction. The term *flux* describes a magnetic field that has a specific direction.

As the disk surface rotates below the drive head, the head can lay a magnetic flux over a region of the disk. When the electric-current flow through the coils in the head is reversed, so is the magnetic-field polarity in the head gap. This reversal also causes the polarity of the flux being placed on the disk to reverse. The *flux reversal* or *flux transition* is a change in polarity of the alignment of magnetic particles on the disk surface.

A drive head places flux reversals on a disk to record data. For each data bit (or bits) written, a pattern of flux reversals is placed on the disk in specific areas known as bit or transition cells. A *bit cell* or *transition cell* is a specific area of the disk controlled by the time and rotational speed in which flux reversals are placed by a drive head. The particular pattern of flux reversals within the transition cells used to store a given data bit or bits is called the *encoding method*. The drive logic or controller takes the data to be stored and encodes it as a series of flux reversals over a period of time, according to the encoding method used. Modified Frequency Modulation (MFM) and Run Length Limited (RLL) are popular encoding methods. All floppy disk drives use the MFM scheme. Hard disks use MFM or several variations of RLL encoding methods. These encoding methods are described in more detail later in this chapter.

During the write process, voltage is applied to the head, and as the polarity of this voltage changes, the polarity of the magnetic field being recorded also changes. The flux transitions are written precisely at the points where the recording polarity changes. Strange as it may seem, during the read process, a head does not output exactly the same signal that was written; instead, the head generates a voltage pulse or spike only when it crosses a flux transition. When the transition is from positive to negative, the pulse that the head would detect is negative voltage. When the transition changes from negative to positive, the pulse would be a positive voltage spike.

In essence, while reading the disk the head becomes a flux transition detector, emitting voltage pulses whenever it crosses a transition. Areas of no transition generate no pulse. Figure 14.1 shows the relationship between the read and write waveforms and the flux transitions recorded on a disk.

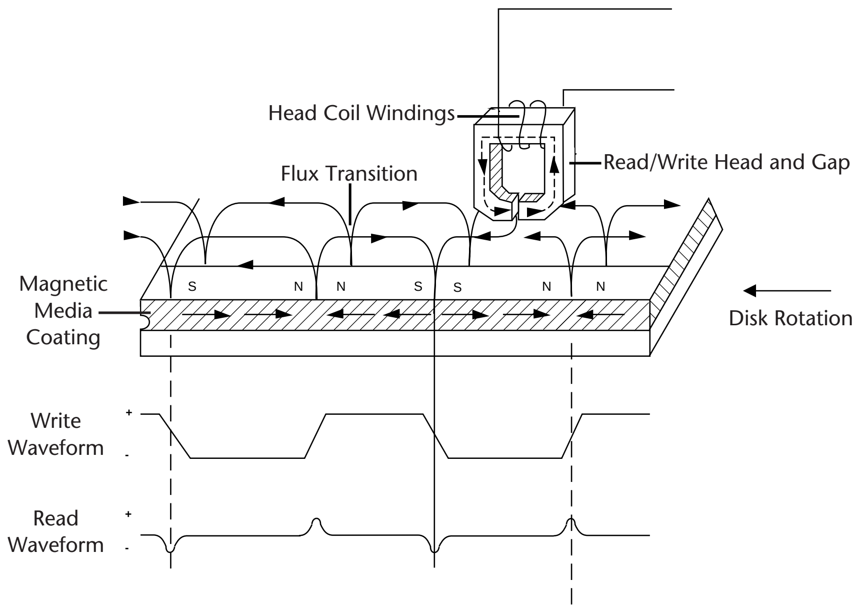


Fig. 14.1
Magnetic write and read processes.

You can think of the write pattern as being a square waveform that is at a positive or negative voltage level and that continuously polarizes the disk media in one direction or another. Where the waveform transitions go from positive to negative voltage, or vice versa, the magnetic flux on the disk also changes polarity. During a read, the head senses the flux transitions and outputs a pulsed waveform with a signal of zero volts unless a positive or negative transition is being detected. Pulses appear only when flux transitions occur on the disk media. By knowing the clock timing used, the drive or controller circuitry can determine whether a pulse (and therefore a flux transition) exists within a given transition cell.

The electrical pulse currents generated in the head while it is passing over a disk in read mode are very weak and can contain significant noise. Sensitive electronics in the drive and controller assembly then can amplify the signal above the noise level and decode the train of weak pulse currents back into data that is (theoretically) identical to the data originally recorded.

So as you now can see, disks are both recorded and read by means of basic electromagnetic principles. Data is recorded on a disk by passing electrical currents through an electromagnet (the drive head) that generates a magnetic field stored on the disk. Data on a disk is read by passing the head back over the surface of the disk; as the head encounters changes in the stored magnetic field, it generates a weak electrical current that indicates the presence or absence of flux transitions in the originally recorded signal.

Data Encoding Schemes

Magnetic media essentially is an analog storage medium. The data that we store on it, however, is digital information—that is, ones and zeros. When digital information is applied to a magnetic recording head, the head creates magnetic domains on the disk media with specific polarities. When a positive current is applied to the write head, the magnetic domains are polarized in one direction; when negative voltage is applied, the magnetic domains are polarized in the opposite direction. When the digital waveform that is recorded switches from a positive to a negative voltage, the polarity of the magnetic domains is reversed.

During a readback, the head actually generates no voltage signal when it encounters a group of magnetic domains with the same polarity, but it generates a voltage pulse every time it detects a switch in polarity. These magnetic-polarity switches are called *flux reversals*. Each flux reversal generates a voltage pulse in the read head; it is these pulses that the drive detects when reading data. A read head does not generate the same waveform that was written; instead, it generates a series of pulses, each pulse appearing where a magnetic flux transition has occurred.

To optimize the placement of pulses during magnetic storage, the raw digital input data is passed through a device called an *encoder/decoder (Endec),* which converts the raw binary information to a waveform that is more concerned with the optimum placement of the flux transitions (pulses). During a read operation, the Endec reverses the process and decodes the pulse train back into the original binary data. Over the years, several different schemes for encoding data in this manner have been developed; some are better or more efficient than others.

In any consideration of binary information, the use of timing is important. When interpreting a read or write waveform, the timing of each voltage transition event is critical. If the timing is off, a given voltage transition may be recognized at the wrong time, and bits may be missed, added, or simply misinterpreted. To ensure that the timing is precise, the transmitting and receiving devices must be in sync. This synchronization can be accomplished by adding a separate line for timing, called a *clock signal*, between the two devices. The clock and data signals also can be combined and then transmitted on a single line. This combination of clock and data is used in most magnetic data encoding schemes.

When the clock information is added in with the data, timing accuracy in interpreting the individual bit cells is ensured between any two devices. A *bit cell* is a window in time inside which a voltage transition will be placed to signify a bit. Clock timing is used to determine the start and end of each bit cell. Each bit cell is bounded by two clock cells where the clock transitions can be sent. First there is a clock transition cell, and then the data transition cell, and finally the clock transition cell for the data that follows. By sending clock information along with the data, the clocks will remain in sync, even if a long string of 0 bits are transmitted. Unfortunately, all the transition cells that are used solely for clocking take up space on the media that otherwise could be used for data.

Because the number of flux transitions that can be recorded on a particular medium is limited by the disk media and head technology, disk drive engineers have been trying various ways of encoding the data into a minimum number of flux reversals, taking into consideration the fact that some flux reversals, used solely for clocking, are required. This method permits maximum use of a given drive hardware technology.

Although various encoding schemes have been tried, only a few are popular today. Over the years, these three basic types have been the most popular:

- Frequency Modulation (FM)
- Modified Frequency Modulation (MFM)
- Run Length Limited (RLL)

The following section examines these codes, discussing how they work, where they have been used, and any advantages or disadvantages that apply to them.

FM Encoding. One of the earliest techniques for encoding data for magnetic storage is called *Frequency Modulation* (FM) encoding. This encoding scheme, sometimes called *Single Density* encoding, was used in the earliest floppy disk drives that were installed in PC systems. The original Osborne portable computer, for example, used these Single Density floppy drives, which stored about 80K of data on a single disk. Although it was popular until the late 70s, FM encoding no longer is used today.

FM represents one of the simplest ways to encode zeros and ones on a magnetic surface. In each bit cell, a flux reversal is recorded to indicate a 1 bit, and no flux reversal is recorded to indicate a 0 bit. With no other modifications, problems would occur if you were recording a long series of zeros, in which case no flux transitions would be recorded. If no transitions occurred for a long period, the controller easily could get out of sync with the drive, resulting in a possible misinterpretation of the data. To keep the devices in sync, a clock signal is written onto the drive along with the data.

In FM encoding, each bit actually requires two transition cells. A 1 bit is recorded as a clock flux reversal followed by a data flux reversal, which the drive simply would see as two consecutive flux reversals. A 0 bit also is recorded in two transition cells. Only the clock cell, however, contains a flux reversal; the data cell is empty (no reversal). Whether you are recording a 1 or a 0, the initial flux reversal represents the clock signal, and the second transition cell would carry a reversal only if the data were a 1 bit.

Although this method is simple and inexpensive, it has one major disadvantage: each data bit requires two flux reversals, which reduces potential disk capacity by half. Table 14.2 shows how each bit cell is encoded.

Table 14.2 FM Data to Flux Transition Encoding

Data Bit Value	Flux Encoding
1	TT
0	TN

T = flux transition
N = no flux transition

MFM Encoding. *Modified Frequency Modulation* (MFM) encoding was devised to reduce the number of flux reversals used in the original FM encoding scheme and, therefore, to pack more data onto the disk. In MFM encoding, the use of the clock transition cells is minimized, leaving more room for the data. Clock transitions are recorded only if a stored 0 bit is preceded by another 0 bit; in all other cases, a clock transition is not required. Because the use of the clock transitions has been minimized, the actual clock frequency can be doubled from FM encoding, resulting in twice as many data bits being stored in the same number of flux transitions as in FM.

Because it is twice as efficient as FM encoding, MFM encoding also has been called *Double Density* recording. MFM is used in virtually all PC floppy drives today and was used in nearly all PC hard disks for a number of years. Today, most hard disks use RLL (Run Length Limited) encoding, which provides even greater efficiency than MFM.

Because MFM encoding places twice as many data bits in the same number of flux reversals as FM, the clock speed of the data is doubled, so that the drive actually sees the same number of total flux reversals as with FM. This doubling means that data is read and written at twice the speed in MFM encoding, even though the drive sees the flux reversals arriving at the same frequency as in FM. This method allows existing drive technology to store twice the data and deliver it twice as fast.

The only caveat is that MFM encoding requires improved disk controller and drive circuitry, because the timing of the flux reversals must be more precise than in FM. As it turned out, these improvements were not difficult to achieve, and MFM encoding became the most popular encoding scheme for a long period.

Table 14.3 shows the data bit to flux reversal translation in MFM encoding:

Table 14.3 MFM Data to Flux Transition Encoding

Data Bit Value	Flux Encoding
1	NT
0 preceded by 0	TN
0 preceded by 1	NN

T = flux transition
N = no flux transition

RLL Encoding. Today's most popular encoding scheme, called *RLL (Run Length Limited),* packs up to 50 percent more information on a given disk than even MFM does and three times as much information as FM. In RLL encoding, groups of bits are taken as a unit and combined to generate specific patterns of flux reversals. By combining the clock and data in these patterns, the clock rate can be further increased while maintaining the same basic distance between the flux transitions on the disk. By optimizing the code to limit the minimum and maximum distance between two flux transitions, the clock rate (and, therefore, storage density) can be increased—typically by three times over FM and one and a half times over MFM encoding.

IBM invented RLL encoding and first used the method in many of its mainframe disk drives. During the late 1980s, the PC hard disk industry began using RLL encoding schemes to increase the storage capabilities of PC hard disks. Today, virtually every drive on the market uses some form of RLL encoding.

Instead of encoding a single bit, RLL normally encodes a group of data bits at a time. The term *Run Length Limited* is derived from the two primary specifications of these codes, which is the minimum number (the run length) and maximum number (the run limit) of transition cells allowed between two actual flux transitions. Several schemes can be achieved by changing the length and limit parameters, but only two have achieved any real popularity: RLL 2,7 and RLL 1,7.

Even FM and MFM encoding can be expressed as a form of RLL. FM can be called RLL 0,1, because there can be as few as zero and as many as one transition cell separating two flux transitions. MFM can be called RLL 1,3, because as few as one and as many as three transition cells can separate two flux transitions. Although these codes can be expressed in RLL form, it is not common to do so.

RLL 2,7 initially was the most popular RLL variation because it offers a high density ratio (1.5 times that of MFM) with a transition detection window the same relative size as that in MFM. This method allows for high storage density with fairly good reliability. In very high-capacity drives, however, RLL 2,7 did not prove to be reliable enough. Most of today's highest-capacity drives use RLL 1,7 encoding, which offers a density ratio 1.27 times that of MFM and a larger transition detection window relative to MFM. Compared with RLL 2,7 the storage density is a little less, but the reliability is much higher. Because of the larger relative window size within which a transition can be detected, RLL 1,7 is a more forgiving and more reliable code, which is required when media and head technology are being pushed to their limits. With the greater need for reliability in high-capacity disk storage, it seems that RLL 1,7 soon will be the most popular code in use.

Another little-used RLL variation called *RLL 3,9*—sometimes called *ARLL (Advanced RLL)*—allowed an even higher density ration than RLL 2,7. Unfortunately, reliability suffered too greatly under the RLL 3,9 scheme; the method was used by only a few controller companies that have all but disappeared.

It is difficult to understand how RLL codes work without looking at an example. Because RLL 2,7 still is the most popular form of RLL encoding, I will use it as an example. Even within a given RLL variation such as RLL 2,7 or 1,7, many different tables can be constructed to show what groups of bits are encoded as what sets of flux transitions. For RLL 2,7 specifically, thousands of different translation tables could be constructed, but for my examples, I will use the encoder/decoder (Endec) table used by IBM because it is the most popular variation that you will find.

According to the IBM conversion tables, specific groups of data bits 2, 3, and 4 bits long are translated into strings of flux transitions 4, 6, and 8 transition cells long, respectively. The selected transitions coded for a particular bit sequence are designed to ensure that flux transitions do not occur too close together or too far apart.

It is necessary to limit how close two flux transitions can be because of the basically fixed resolution capabilities of the head and disk media. Limiting how far apart these transitions can be ensures that the clocks in the devices remain in sync.

Table 14.4 shows the IBM-developed encoding scheme for 2,7 RLL.

Table 14.4 RLL 2,7 (IBM Endec) Data to Flux Transition Encoding

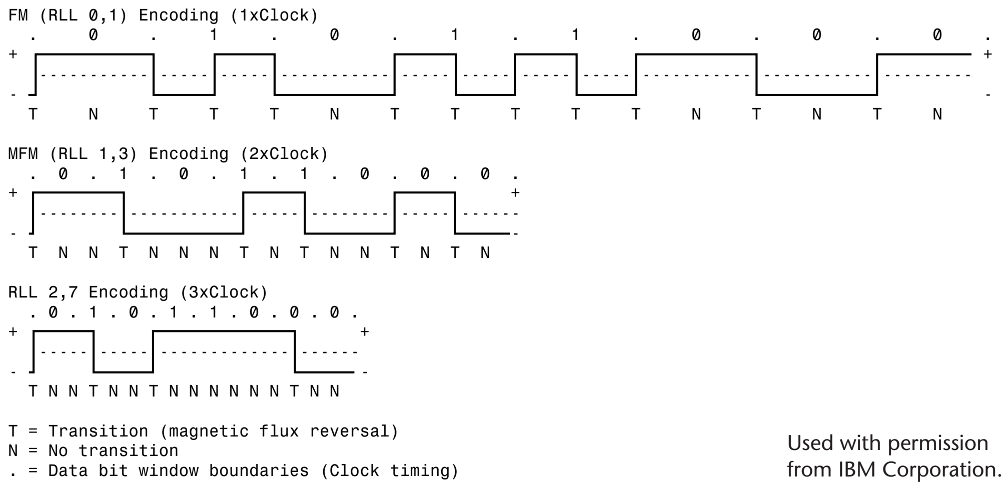
Data Bit Values	Flux Encoding
10	NTNN
11	TNNN
000	NNNTNN
010	TNNTNN
011	NNTNNN
0010	NNTNNTNN
0011	NNNNTNNN

T = flux transition
N = no flux transition

In studying this table, you may think that encoding a byte such as 00000001b would be impossible because no combinations of data bit groups fit this byte. Encoding this type of byte is not a problem, however, because the controller does not transmit individual bytes; instead, the controller sends whole sectors, making it possible to encode such a byte simply by including some of the bits in the following byte. The only real problem occurs in the last byte of a sector if additional bits are needed to complete the final group sequence. In these cases, the encoder/decoder (Endec) in the controller simply adds excess bits to the end of the last byte. These excess bits are truncated during any reads so that the last byte always is correctly decoded.

Encoding Scheme Comparisons

Figure 14.2 shows an example of the waveform written to store an *X* ASCII character on a hard disk drive under three different encoding schemes.


FM (RLL 0,1) Encoding (1xClock)
. 0 . 1 . 0 . 1 . 1 . 0 . 0 . 0 .
T N T T T N T T T T T N T N T N
MFM (RLL 1,3) Encoding (2xClock)
. 0 . 1 . 0 . 1 . 1 . 0 . 0 . 0 .
T N N T N N N T N T N N T N T N
RLL 2,7 Encoding (3xClock)
. 0 . 1 . 0 . 1 . 1 . 0 . 0 . 0 .
T N N T N N T N N N N N N T N N
T = Transition (magnetic flux reversal)
N = No transition
. = Data bit window boundaries (Clock timing)

Used with permission from IBM Corporation.

Fig. 14.2
ASCII *X* write waveforms using FM, MFM, and RLL 2,7 encoding.

In each of these encoding-scheme examples, the top line shows the individual data bits (01011000b) in their bit cells separated in time by the clock signal, which is shown as a period (.). Below that line is the actual write waveform, showing the positive and negative voltages as well as voltage transitions that result in the recording of flux transitions. The bottom line shows the transition cells, with T representing a transition cell that contains a flux transition and N representing a transition cell that is empty (no flux transition).

The FM encoding example is easy to explain. Each bit cell has two transition cells: one for the clock information and one for the data itself. All the clock transition cells contain flux transitions, and the data transition cells contain a flux transition only if the data is a 1 bit. No transition at all is used to represent a 0 bit. Starting from the left, the first data bit is 0, which decodes as a flux transition pattern of TN. The next bit is a 1, which decodes as TT. The next bit is 0, which decodes as TN, and so on. Using the FM encoding chart listed earlier, you easily can trace the FM encoding pattern to the end of the byte. Notice that in FM encoding, transitions can be written in adjacent transition cells, with no empty transition cells between them. This method is called a minimum run length of 0. Also, the maximum number of empty transition cells between any two flux transitions is 1, which is why FM encoding can be called RLL 0,1.

The MFM encoding scheme also has clock and data transition cells for each data bit to be recorded. As you can see, however, the clock transition cells carry a flux transition only when a 0 bit is stored after another 0 bit. Starting from the left, the first bit is a 0, and the preceding bit is unknown (assume 0), so the flux transition pattern is TN for that bit. The next bit is a 1, which always decodes to a transition-cell pattern of NT. The next bit is 0, which was preceded by 1, so the pattern stored is NN. Using the MFM encoding table listed earlier, you easily can trace the MFM encoding pattern to the end of the byte. You can see that the minimum and maximum number of transition cells between any two flux transitions is one and three, respectively; hence, MFM encoding also can be called RLL 1,3.

Notice that because half the total transitions used in FM are required, the clock rate can be doubled, so that the data takes up only half as much space. Also notice that even with the doubled clock rate, the minimum physical distance between any two flux transitions is exactly the same as with FM, meaning that the actual density of the write waveform is the same as with FM even though twice as much data is being encoded.

The RLL 2,7 pattern is more difficult to see because it relies on encoding groups of bits rather than each bit individually. Starting from the left, the first group that matches the groups listed in the encoder/decoder (Endec) table are the first three bits, 010. These bits are translated into a flux transition pattern of TNNTNN. The next two bits, 11, are translated as a group to TNNN; and the final group, 000 bits, is translated to NNNTNN to complete the byte. As you can see in this example, no additional bits were needed to finish the last group.

Notice that the minimum and maximum number of empty transition cells between any two flux transitions in this example are two and six, although a different example could show a maximum of seven empty transition cells. This is where the RLL 2,7 designation comes from. Because even fewer transitions are recorded than in MFM, the clock rate can be further increased to three times that of FM or one and a half times that of MFM, allowing more data to be stored in the same space on the disk. Notice, however, that the resulting write waveform itself looks exactly like a typical FM or MFM waveform in terms of the number and separation of the flux transitions for a given physical portion of the disk. In other words, the physical minimum and maximum distances between any two flux transitions remain the same in all three of these encoding-scheme examples.

Another new feature in high-end drives involves the disk read circuitry. Read channel circuits using Partial-Response, Maximum-Likelihood (PRML) technology allow disk drive manufacturers to increase the amount of data that can be stored on a disk platter by up to 40 percent. PRML replaces the standard “detect one peak at a time” approach of traditional analog peak-detect read/write channels with digital signal processing.

In digital signal processing, noise can be digitally filtered out, allowing flux change pulses to be placed closer together on the platter, achieving greater densities.

I hope that the examinations of these different encoding schemes and how they work have taken some of the mystery out of the way data is recorded on a drive. You can see that although schemes such as MFM and RLL can store more data on a drive, the actual density of the flux transitions remains the same as far as the drive is concerned.

Sectors

A disk track is too large to manage effectively as a single storage unit. Many disk tracks can store 50,000 or more bytes of data, which would be very inefficient for storing small files. For that reason, a disk track is divided into several numbered divisions known as *sectors*. These sectors represent slices of the track.

Different types of disk drives and disks split tracks into different numbers of sectors, depending on the density of the tracks. For example, floppy disk formats use 8 to 36 sectors per track, whereas hard disks usually store data at a higher density and can use 17 to 100 or more sectors per track. Sectors created by standard formatting procedures on PC systems have a capacity of 512 bytes, but this capacity may change in the future.

Sectors are numbered on a track starting with 1, unlike the heads or cylinders, which are numbered starting with 0. For example, a 1.44MB floppy disk contains 80 cylinders numbered from 0 to 79 and two heads numbered 0 and 1, and each track on each cylinder has 18 sectors numbered from 1 to 18.

When a disk is formatted, additional areas are created on the disk for the disk controller to use for sector numbering and identifying the start and end of each sector. These areas precede and follow each sector's data area, which accounts for the difference between a disk's unformatted and formatted capacities. For example, a 4MB floppy disk (3.5-inch) has a capacity of 2.88MB when it is formatted, and a 38MB hard disk has a capacity of only 32MB when it is formatted. All drives use some reserved space for managing the data that can be stored on the drive.

Although I have stated that each disk sector is 512 bytes in size, this statement technically is false. Each sector does allow for the storage of 512 bytes of data, but the data area is only a portion of the sector. Each sector on a disk typically occupies 571 bytes of the disk, of which only 512 bytes are usable for user data. The actual number of bytes required for the sector header and trailer can vary from drive to drive, but this figure is typical.

You may find it helpful to think of each sector as being a page in a book. In a book, each page contains text, but the entire page is not filled with text; rather, each page has top, bottom, left, and right margins. Information such as chapter titles (track and cylinder numbers) and page numbers (sector numbers) is placed in the margins. The "margin" areas of a sector are created and written to during the disk-formatting process. Formatting also fills the data area of each sector with dummy values. After the disk is formatted, the data area can be altered by normal writing to the disk. The sector header and trailer information cannot be altered during normal write operations unless you reformat the disk.

Each sector on a disk has a *prefix p*ortion, or header, that identifies the start of the sector and a sector number, as well as a *suffix portion*, or trailer, that contains a *checksum* (which helps ensure the integrity of the data contents). Each sector also contains 512 bytes of data. The data bytes normally are set to some specific value, such as F6h (hex), when the disk is physically (or low-level) formatted. (The following section explains low-level formatting.)

In many cases, a specific pattern of bytes that are considered to be difficult to write are written so as to flush out any marginal sectors. In addition to the gaps within the sectors, gaps exist between sectors on each track and also between tracks; none of these gaps contains usable data space. The prefix, suffix, and gaps account for the lost space between the unformatted capacity of a disk and the formatted capacity.

Table 14.5 shows the format for each track and sector on a typical hard disk with 17 sectors per track.

Table 14.5 Typical 17-Sector/17-Track Disk Sector Format

Bytes	Name	Description
16	POST INDEX GAP	All 4Eh, at the track beginning after the Index mark
		Sector data; repeated 17 times for an MFM encoded track
13	ID VFO LOCK	All 00h; synchronizes the VFO for the sector ID
1	SYNC BYTE	A1h; notifies the controller that data follows
1	ADDRESS MARK	FEh; defines that ID field data follows
2	CYLINDER NUMBER	A value that defines the actuator position
1	HEAD NUMBER	A value that defines the head selected
1	SECTOR NUMBER	A value that defines the sector
2	CRC	Cyclic Redundancy Check to verify ID data
3	WRITE TURN-ON GAP	00h written by format to isolate the ID from DATA
13	DATA SYNC VFO LOCK	All 00h; synchronizes the VFO for the DATA
1	SYNC BYTE	A1h; notifies the controller that data follows
1	ADDRESS MARK	F8h; defines that user DATA field follows
512	DATA	The area for user DATA
2	CRC	Cyclic Redundancy Check to verify DATA
3	WRITE TURN-OFF GAP	00h; written by DATA update to isolate DATA
15	INTER-RECORD GAP	All 00h; a buffer for spindle speed variation
693	PRE-INDEX GAP	All 4Eh, at track end before Index mark

571	*Total bytes per sector*
512	*Data bytes per sector*
10,416	*Total bytes per track*
8,704	*Data bytes per track*

Table 14.5 refers to a hard disk track with 17 sectors. Although this capacity is typical, more advanced hard disks place as many as 100 or more sectors per track, and the specific formats of those sectors may vary slightly from the example.

As you can see, the usable space on each track is about 16 percent less than the unformatted capacity. This example is true for most disks, although some may vary slightly.

The Post Index Gap provides a head-switching recovery period so that when switching from one track to another, sequential sectors can be read without waiting for an additional revolution of the disk. In some drives running 1:1 interleave controllers, this time is not enough; additional time can be added by skewing the sectors so that the arrival of the first sector is delayed.

The Sector ID data consists of the Cylinder, Head, and Sector Number fields, as well as a CRC field to allow for verification of the ID data. Most controllers use bit 7 of the Head Number field to mark the sector as bad during a low-level format or surface analysis. This system is not absolute, however; some controllers use other methods to indicate a marked bad sector. Usually, though, the mark involves one of the ID fields.

Write Turn on Gap follows the ID field CRC bytes and provides a pad to ensure a proper recording of the following user data area as well as to allow full recovery of the ID CRC.

The user Data field consists of all 512 bytes of data stored in the sector. This field is followed by a CRC field to verify the data. Although many controllers use two bytes of CRC here, the controller may implement a longer Error Correction Code (ECC) that requires more than two CRC bytes to store. The ECC data stored here provides the possibility of Data-field read correction as well as read error detection. The correction/detection capabilities depend on the ECC code chosen and on the controller implementation. A write turn-off gap is a pad to allow the ECC (CRC) bytes to be fully recovered.

The Inter-Record Gap provides a means to accommodate variances in drive spindle speeds. A track may have been formatted while the disk was running slower than normal and then write updated while the disk was running faster than normal. In such cases, this gap prevents accidental overwriting of any information in the next sector. The actual size of this padding varies, depending on the speed of disk rotation when the track was formatted and each time the Data field is updated.

The Pre-Index Gap allows for speed tolerance over the entire track. This gap varies in size, depending on the variances in disk-rotation speed and write-frequency tolerance at the time of formatting.

This sector prefix information is extremely important, because it contains the numbering information that defines the cylinder, head, and sector. All this information except the Data field, Data CRC bytes, and Write Turn-Off Gap is written only during a low-level format. On a typical non-servo-guided (stepper-motor actuator) hard disk on which thermal gradients cause mistracking, the data updates that rewrite the 512-byte Data area and the CRC that follows may not be placed exactly in line with the sector header information. This situation eventually causes read or write failures of the `Abort, Retry, Fail, Ignore` variety. You can correct this problem by reformatting the disk; this process rewrites the header and data information together at the current track positions. Then, when you restore the data to the disk, the Data areas are written in alignment with the new sector headers.

Disk Formatting. You usually have two types of formats to consider:

- Physical, or *low-level*
- Logical, or *high-level*

When you format a floppy disk, the DOS FORMAT command performs both kinds of formats simultaneously. To format a hard disk, however, you must perform the operations separately. Moreover, a hard disk requires a third step, between the two formats, in which the partitioning information is written to the disk. *Partitioning* is required because a hard disk is designed to be used with more than one operating system. Separating the physical format in a way that is always the same, regardless of the operating system being used and regardless of the high-level format (which would be different for each operating system), makes possible the use of multiple operating systems on one hard drive. The partitioning step allows more than one type of operating system to use a single hard disk or a single DOS to use the disk as several volumes or logical drives. A volume or logical drive is anything to which DOS assigns a drive letter.

Consequently, formatting a hard disk involves three steps:

1. Low-level formatting (LLF)
2. Partitioning
3. High-level formatting (HLF)

During a low-level format, the disk's tracks are divided into a specific number of sectors. The sector header and trailer information is recorded, as are intersector and intertrack gaps. Each sector's data area is filled with a dummy byte value or test pattern of values. For floppy disks, the number of sectors recorded on each track depends on the type of disk and drive; for hard disks, the number of sectors per track depends on the drive and controller interface.

The original ST-506/412 MFM controllers always placed 17 sectors per track on a disk. ST-506/412 controllers with RLL encoding increase the number of sectors on a drive to 25 or 26 sectors per track. ESDI drives can have 32 or more sectors per track. IDE drives simply are drives with built-in controllers, and depending on exactly what type of controller design is built in, the number of sectors per track can range from 17 to 100 or more. SCSI drives essentially are IDE drives with an added SCSI Bus Adapter circuit, meaning that they also have some type of built-in controller; and like IDE drives, SCSI drives can have practically any number of sectors per track, depending on what controller design was used.

Most IDE and SCSI drives use a technique called *Zoned Recording,* which writes a variable number of sectors per track. The outermost tracks hold more sectors than the inner tracks do, because they are longer. Because of limitations in the PC BIOS, these drives still have to act as though they have a fixed number of sectors per track. This situation is handled by translation algorithms that are implemented in the controller.

Multiple Zone Recording. One way to increase the capacity of a hard drive is to format more sectors on the outer cylinders than on the inner ones. Because they have a larger circumference, the outer cylinders can hold more data. Drives without Zoned Recording store the same amount of data on every cylinder, even though the outer cylinders may be twice as long as the inner cylinders. The result is wasted storage capacity, because the disk media must be capable of storing data reliably at the same density as on the inner cylinders. With ST-506/412 and ESDI controllers, unfortunately, the number of sectors per track was fixed; drive capacity, therefore, was limited by the density capability of the innermost (shortest) track.

In a Zoned Recording, the cylinders are split into groups called *zones*, with each successive zone having more and more sectors per track as you move out from the inner radius of the disk. All the cylinders in a particular zone have the same number of sectors per track. The number of zones varies with specific drives, but most drives have 10 or more zones.

Another effect of Zoned Recording is that transfer speeds will vary depending on what zone the heads are in. Since there are more sectors in the outer zones, but the rotational speed is always the same, the transfer rate will be highest.

Drives with separate controllers could not handle zoned recordings because there was no standard way to communicate information about the zones from the drive to the controller. With SCSI and IDE disks, it became possible to format individual tracks with different numbers of sectors, due to the fact that these drives have the disk controller built in. The built-in controllers on these drives can be made fully aware of the zoning that is used. These built-in controllers must then also translate the physical Cylinder, Head, and Sector numbers to logical Cylinder, Head, and Sector numbers so that the drive has the appearance of having the same number of sectors on each track. The PC BIOS was designed to only handle a single number of specific sectors per track throughout the entire drive, meaning that zoned drives always must run under a sector translation scheme.

The use of Zoned Recording has allowed drive manufacturers to increase the capacity of their hard drives by between 20 percent and 50 percent compared with a fixed-sector-per-track arrangement. Nearly all IDE and SCSI drives today use Zoned Recording.

Partitioning. Partitioning segments the drive into areas, called *partitions*, that can hold a particular operating system's file system. Today, PC operating systems use three common file systems:

- *FAT (File Allocation Table).* This system is the standard file system used by DOS, OS/2, and Windows NT. FAT partitions support file names of 11 characters maximum (eight plus a three-character extension), and a volume can be as large as 2GB.

- *HPFS (High Performance File System)*. This UNIX-style file system is accessible only under OS/2 and Windows NT. DOS applications running under OS/2 or Windows NT can access files in HPFS partitions, but straight DOS cannot. File names can be 256 characters long, and volume size is limited to 8GB.
- *NTFS (Windows NT File System)*. This UNIX-style file system currently is accessible only under Windows NT, but drivers should be available for OS/2 to access NTFS as well. DOS cannot access these partitions, but DOS applications running under Windows NT can. File names can be 256 characters long, and volume size is limited to 8GB.

Of these three file systems, the FAT file system still is by far the most popular (and recommended). The main problem with the FAT file system is that disk space is used in groups of sectors called *allocation units* or *clusters*. On large volumes, the larger cluster sizes required cause disk space to be used inefficiently. HPFS and NTFS always manage the disk space in sector increments, so there is no penalty of wasted disk space with large volumes.

The FAT file system is the most recommended for compatibility reasons. For example, few applications currently are compatible with the longer file names possible in the HPFS and NTFS file systems. All the operating systems can access FAT volumes, and the file structures and data-recovery procedures are well known. Data recovery can be difficult to impossible under the HPFS and NTFS systems; for those systems, good backups are imperative.

During partitioning, no matter what file system is specified, the partitioning software writes a special boot program and partition table to the first sector, called the Master Boot Sector (MBS). Because the term *record* sometimes is used to mean *sector*, this sector can also be called the Master Boot Record (MBR).

High-Level Format. During the high-level format, the operating system (such as DOS, OS/2, or Windows NT) writes the structures necessary for managing files and data. FAT partitions have a Volume Boot Sector (VBS), a file allocation table (FAT), and a root directory on each formatted logical drive. These data structures (discussed in detail in Chapter 15, "CD-ROM Drives") enable the operating system to manage the space on the disk, keep track of files, and even manage defective areas so that they do not cause problems.

High-level formatting is not really formatting, but creating a table of contents for the disk. In low-level formatting, which is the real formatting, tracks and sectors are written on the disk. As mentioned, the DOS FORMAT command can perform both low-level and high-level format operations on a floppy disk, but it performs only the high-level format for a hard disk. Hard disk low-level formats require a special utility, usually supplied by the disk-controller manufacturer.

Basic Hard Disk Drive Components

Many types of hard disks are on the market, but nearly all drives share the same basic physical components. Some differences may exist in the implementation of these components (and in the quality of materials used to make them), but the operational characteristics of most drives are similar. Following are the components of a typical hard disk drive (see fig. 14.3):

- Disk platters
- Read/write heads
- Head actuator mechanism
- Spindle motor
- Logic board
- Cables and connectors
- Configuration items (such as jumpers or switches)
- Bezel (optional)

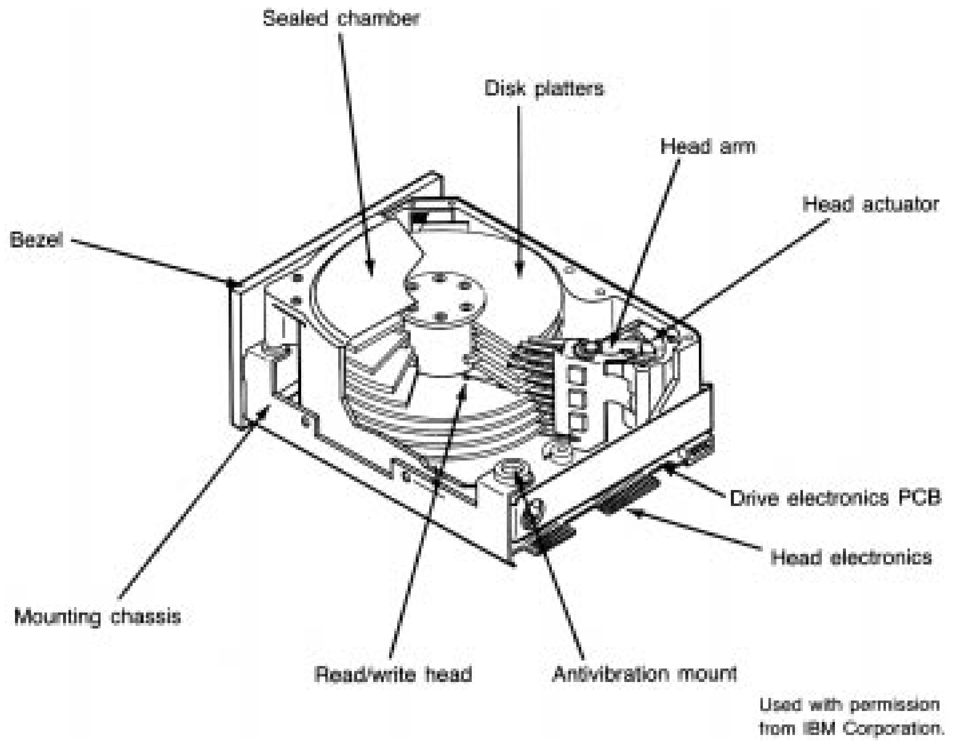


Fig. 14.3
Hard disk drive components.

The platters, spindle motor, heads, and head actuator mechanisms usually are contained in a sealed chamber called the *Head Disk Assembly* (HDA). The HDA usually is treated as a single component; it rarely is opened. Other parts external to the drive's HDA—such as the logic boards, bezel, and other configuration or mounting hardware—can be disassembled from the drive.

Hard Disk Platters (Disks)

A typical hard disk has one or more platters, or *disks*. Hard disks for PC systems have been available in a number of form factors over the years. Normally, the physical size of a drive is expressed as the size of the platters. Following are the most common platter sizes used in PC hard disks today:

- 5.25-inch (actually, 130mm, or 5.12 inches)
- 3.5-inch (actually, 95mm, or 3.74 inches)
- 2.5-inch
- 1.8-inch
- 1.3-inch

Larger hard drives that have 8-inch, 14-inch, or even larger platters are available, but these drives typically have not been associated with PC systems. Currently, the 3.5-inch drives are the most popular for desktop and some portable systems, whereas the 2.5-inch and smaller drives are very popular in portable systems. The 1.3-inch drives entered the market most recently, and they truly are amazing. These drives, which are about the size of a large wristwatch, currently can store 40MB. Capacities of 100MB or more are not far away: the 1.3-inch drives are expected to reach 1GB capacity by the year 2000! Due to their small size, these drives are extremely rugged; they can withstand shocks of 100 g's *while operating* and double that when powered off.

I thought it was interesting to note that the 3.5-inch drives actually use platters that are 95mm or 3.74 inches in diameter, whereas the 5.25-inch drives' platters actually are 130mm or 5.12 inches in diameter. (This information could prove to be useful in your next Computer Trivia contest!)

Most hard drives have two or more platters, although some of the smaller drives have only one. The number of platters that a drive can have is limited by the drive's physical size vertically. So far, the maximum number of platters that I have seen in any 5.25-inch full-height drive is 11; some 3.5-inch drives have as many as 10 platters.

Platters traditionally have been made from an aluminum alloy, for strength and light weight. With manufacturers' desire for higher and higher densities and smaller drives, many drives now use platters made of glass (or, more technically, a glass-ceramic composite). One such material is called MemCor, which is produced by the Dow Corning glass company. MemCor is composed of glass with ceramic implants, which resists cracking better than pure glass.

Glass platters offer greater strength (rigidity) and therefore can be machined to one-half the thickness of conventional aluminum disks, or less. Glass platters also are much more thermally stable than aluminum platters, which means that they do not change dimensions (expand or contract) very much with any changes in temperature. Several hard disks made by companies such as Seagate, Toshiba, Areal Technology, Maxtor, and Hewlett Packard currently use glass or glass-ceramic platters. For most manufacturers, glass disks will replace the standard aluminum substrate over the next few years, especially in high-performance 2.5- and 3.5-inch drives.

Recording Media

No matter what substrate is used, the platters are covered with a thin layer of a magnetically retentive substance called *media* in which magnetic information is stored. Two popular types of media are used on hard disk platters:

- Oxide media
- Thin-film media

Oxide media is made of various compounds, containing iron oxide as the active ingredient. A magnetic layer is created by coating the aluminum platter with a syrup containing iron-oxide particles. This media is spread across the disk by spinning the platters at high speed; centrifugal force causes the material to flow from the center of the platter to the outside, creating an even coating of media material on the platter. The surface then is cured and polished. Finally, a layer of material that protects and lubricates the surface is added and burnished smooth. The oxide media coating normally is about 30 millionths of an inch thick. If you could peer into a drive with oxide-media-coated platters, you would see that the platters are brownish or amber.

As drive density increases, the media needs to be thinner and more perfectly formed. The capabilities of oxide coatings have been exceeded by most higher-capacity drives. Because oxide media is very soft, disks that use this type of media are subject to head-crash damage if the drive is jolted during operation. Most older drives, especially those sold as low-end models, have oxide media on the drive platters. Oxide media, which has been used since 1955, remained popular because of its relatively low cost and ease of application. Today, however, very few drives use oxide media.

Thin-film media is thinner, harder, and more perfectly formed than oxide media. Thin film was developed as a high-performance media that enabled a new generation of drives to have lower head floating heights, which in turn made possible increases in drive density. Originally, thin-film media was used only in higher-capacity or higher-quality drive systems, but today, virtually all drives have thin-film media.

Thin-film media is aptly named. The coating is much thinner than can be achieved by the oxide-coating method. Thin-film media also is known as *plated*, or *sputtered*, media because of the various processes used to place the thin film of media on the platters.

Thin-film plated media is manufactured by placing the media material on the disk with an electroplating mechanism, much the way chrome plating is placed on the bumper of a car. The aluminum platter then is immersed in a series of chemical baths that coat the platter with several layers of metallic film. The media layer is a cobalt alloy about 3 micro-inches (millionths of an inch) thick.

Thin-film sputtered media is created by first coating the aluminum platters with a layer of nickel phosphorus and then applying the cobalt-alloy magnetic material in a continuous vacuum-deposition process called *sputtering*. During this process, magnetic layers as thin as 1 or 2 micro-inches are deposited on the disk, in a fashion similar to the way that silicon wafers are coated with metallic films in the semiconductor industry. The sputtering technique then is used again to lay down an extremely hard, 1 micro-inch protective carbon coating. The need for a near-perfect vacuum makes sputtering the most expensive of the processes described here.

The surface of a sputtered platter contains magnetic layers as thin as 1 millionth of an inch. Because this surface also is very smooth, the head can float closer to the disk surface than was possible previously; floating heights as small as 3 micro-inches above the surface are possible. When the head is closer to the platter, the density of the magnetic flux transitions can be increased to provide greater storage capacity. Additionally, the increased intensity of the magnetic field during a closer-proximity read provides the higher signal amplitudes needed for good signal-to-noise performance.

Both the sputtering and plating processes result in a very thin, very hard film of media on the platters. Because the thin-film media is so hard, it has a better chance of surviving contact with the heads at high speed. In fact, modern thin-film media is virtually uncrashable. Oxide coatings can be scratched or damaged much more easily. If you could open a drive to peek at the platters, you would see that the thin-film media platters look like the silver surfaces of mirrors.

The sputtering process results in the most perfect, thinnest, and hardest disk surface that can be produced commercially. The sputtering process has largely replaced plating as the method of creating thin-film media. Having a thin-film media surface on a drive results increased storage capacity in a smaller area with fewer head crashes—and in a drive that will provide many years of trouble-free use.

Read/Write Heads

A hard disk drive usually has one read/write head for each platter side, and these heads are connected, or *ganged*, on a single movement mechanism. The heads, therefore, move across the platters in unison.

Mechanically, read/write heads are simple. Each head is on an actuator arm that is spring-loaded to force the head into a platter. Few people realize that each platter actually is "squeezed" by the heads above and below it. If you could open a drive safely and lift the top head with your finger, the head would snap back into the platter when you released it. If you could pull down on one of the heads below a platter, the spring tension would cause it to snap back up into the platter when you released it.

Figure 14.4 shows a typical hard disk head-actuator assembly from a voice coil drive.

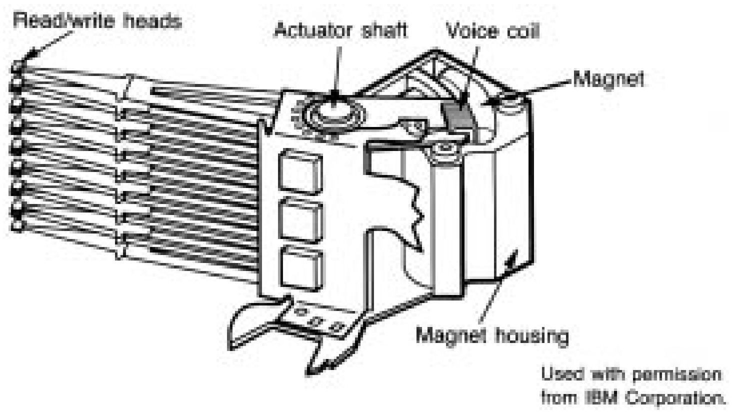


Fig. 14.4
Read/write heads and rotary voice coil actuator assembly.

When the drive is at rest, the heads are forced into direct contact with the platters by spring tension, but when the drive is spinning at full speed, air pressure develops below the heads and lifts them off the surface of the platter. On a drive spinning at full speed, the distance between the heads and the platter can be anywhere from 3 to 20 or more micro-inches (millionths of an inch).

In the early 1960s, hard disk drive recording heads operated at floating heights as large as 200 to 300 micro-inches; today's drive heads are designed to float as low as 3 to 5 micro-inches above the surface of the disk. To support higher densities in future drives, the physical separation between the head and disk is expected to be as little as 0.5 micro-inch by the end of the century.

Generally speaking, the older the drive and lower its capacity, the higher the heads float above the media. The small size of this gap is why the disk drive's Head Disk Assembly (HDA) should never be opened except in a clean-room environment: any particle of dust or dirt that gets into this mechanism could cause the heads to read improperly, or possibly even to strike the platters while the drive is running at full speed. The latter event could scratch the platter or the head.

To ensure the cleanliness of the interior of the drive, the HDA is assembled in a class-100 or better clean room. This specification is such that a cubic foot of air cannot contain more than 100 particles that measure up to 0.5 micron (19.7 micro-inch). A single person breathing while standing motionless spews out 500 such particles in a single minute! These rooms contain special air-filtration systems that continuously evacuate and refresh the air. A drive's HDA should not be opened unless it is inside such a room.

Although maintaining such an environment may seem to be expensive, many companies manufacture tabletop or bench-size clean rooms that sell for only a few thousand dollars. Some of these devices operate like a glove box; the operator first inserts the drive and any tools required, and then closes the box and turns on the filtration system. Inside the box, a clean-room environment is maintained, and a technician can use the built-in gloves to work on the drive.

In other clean-room variations, the operator stands at a bench where a forced-air curtain is used to maintain a clean environment on the bench top. The technician can walk in and out of the clean-room field simply by walking through the air curtain. This air curtain is much like the curtain of air used in some stores and warehouses to prevent heat from escaping in the winter while leaving a passage wide open.

Because the clean environment is very expensive to produce, few companies except those that manufacture the drives are prepared to service hard disk drives.

Read/Write Head Designs

As disk drive technology has evolved, so has the design of the Read/Write head. The earliest heads were simple iron cores with coil windings (essentially pure electromagnets). By today's standards the original head designs were enormous in physical size and operated at very low recording densities. Over the years many different head designs have evolved from the first simple Ferrite Core designs into several types and technologies available today. This section discusses the different types of heads found in PC type hard disk drives, including the applications and relative strengths and weaknesses of each.

Four types of heads have been used in hard disk drives over the years:

- Ferrite
- Metal-In-Gap (MIG)
- Thin Film (TF)
- Magneto-Resistive (MR)

Ferrite. *Ferrite heads*, the traditional type of magnetic-head design, evolved from the original IBM Winchester drive. These heads have an iron-oxide core wrapped with electromagnetic coils. A magnetic field is produced by energizing the coils; a field also can be induced by passing a magnetic field near the coils. This process gives the heads full read and write capability. Ferrite heads are larger and heavier than thin-film heads and therefore require a larger floating height to prevent contact with the disk.

Many refinements have been made in the original (monolithic) ferrite head design. A type of ferrite head called a *composite ferrite head* has a smaller ferrite core bonded with glass in a ceramic housing. This design permits a smaller head gap, which allows higher track densities. These heads are less susceptible to stray magnetic fields than are heads in the older monolithic design.

During the 1980s, composite ferrite heads were popular in many low-end drives, such as the popular Seagate ST-225. As density demands grew, the competing MIG and thin-film head designs were used in place of ferrite heads, which are virtually obsolete today. Ferrite heads cannot write to the higher coercivity media needed for high-density designs and have poor frequency response with higher noise levels. The main advantage of ferrite heads is that they are the cheapest type available.

Metal-In-Gap. *Metal-In-Gap* (MIG) heads basically are a specially enhanced version of the composite ferrite design. In MIG heads, a metal substance is sputtered into the recording gap on the trailing edge of the head. This material offers increased resistance to magnetic saturation, allowing higher-density recording. MIG heads also produce a sharper gradient in the magnetic field for a better-defined magnetic pulse. These heads permit the use of higher-coercivity thin-film disks and can operate at lower floating heights.

Two versions of MIG heads are available: single-sided and double-sided. Single-sided MIG heads are designed with a layer of magnetic alloy placed along the trailing edge of the gap. Double-sided MIG designs apply the layer to both sides of the gap. The metal alloy is applied through a vacuum-deposition process called sputtering. This alloy has twice the magnetization capability of raw ferrite and allows writing to the higher-coercivity thin-film media needed at the higher densities. Double-sided MIG heads offer even higher coercivity capability than the single-sided designs do.

Because of these increases in capabilities through improved designs, MIG heads for a time were the most popular head used in all but very high-capacity drives. Due to market pressures that have demanded higher and higher densities, however, MIG heads have been largely displaced in favor of Thin Film heads.

Thin Film. *Thin Film* (TF) heads are produced in the much the same manner as a semiconductor chip—that is, through a photolithographic process. In this manner, many thousands of heads can be created on a single circular wafer. This manufacturing process also results in a very small high-quality product.

TF heads offer an extremely narrow and controlled head gap created by sputtering a hard aluminum material. Because this material completely encloses the gap, this area is very well protected, minimizing the chance of damage from contact with the media. The core is a combination of iron and nickel alloy that is two to four times more powerful magnetically than a ferrite head core.

Thin film heads produce a sharply defined magnetic pulse that allows extremely high densities to be written. Because they do not have a conventional coil, TF heads are more immune to variations in coil impedance. The small, lightweight heads can float at a much lower height than the ferrite and MIG heads; floating height has been reduced to 2 micro-inches or less in some designs. Because the reduced height enables a much stronger signal to be picked up and transmitted between the head and platters, the signal-to-noise ratio increases, which improves accuracy. At the high track and linear densities of some drives, a standard ferrite head would not be able to pick out the data signal from the background noise. When Thin Film heads are used, their small size enables more platters to be stacked in a drive.

Until the past few years, TF heads were relatively expensive compared with older technologies, such as ferrite and MIG. Better manufacturing techniques and the need for higher densities, however, have driven the market to TF heads. The widespread use of these heads also has made them cost-competitive, if not cheaper than MIG heads.

Thin film heads currently are used in most high-capacity drives, especially in the smaller form factors. They have displaced MIG heads as the most popular head design being used in drives today. The industry is working on ways to improve TF head efficiency, so thin film heads are likely to remain popular for some time, especially in mainstream drives.

Magneto-Resistive. *Magneto-Resistive* (MR) heads are a relatively new technology. Invented and pioneered by IBM, magneto-resistive heads currently are the superior head design, offering the highest performance available. Most 3.5-inch drives with capacities in excess of 1G currently use MR heads. As areal densities continue to increase, the MR head eventually will become the head of choice for nearly all hard drives, displacing the popular MIG and TF head designs.

MR heads rely on the fact that the resistance of a conductor changes slightly when an external magnetic field is present. Rather than put out a voltage by passing through a magnetic-field flux reversal, as a normal head would, the MR head senses the flux reversal and changes resistance. A small current flows through the heads, and the change in resistance is measured by this sense current. This design enables the output to be three or more times more powerful than a thin film head during a read. In effect, MR heads are power-read heads, acting more like sensors than generators.

MR heads are more costly and complex to manufacture than other types of heads, because several special features or steps must be added. Among them:

- Additional wires must be run to and from the head to carry the sense current.
- Four to six more masking steps are required.
- Because MR heads are so sensitive, they are very susceptible to stray magnetic fields and must be shielded.

Because the MR principle can only read data and is not used for writing, MR heads really are two heads in one. A standard inductive thin film head is used for writing, and a magneto-resistive head is used for reading. Because two separate heads are built into one assembly, each head can be optimized for its task. Ferrite, MIG, and thin film heads are known as *single-gap heads* because the same gap is used for both reading and writing, whereas the MR head uses a separate gap for each operation.

The problem with single-gap heads is that the gap length always is a compromise between what is best for reading and what is best for writing. The read function needs a thin gap for higher resolution; the write function needs a thicker gap for deeper flux penetration to switch the media. In a dual-gap MR head, the read and write gaps can be optimized for both functions independently. The write (thin film) gap writes a wider track than the read (magneto-resistive) gap does. The read head then is less likely to pick up stray magnetic information from adjacent tracks.

Although MR heads have many good qualities, they do have some disadvantages. The primary disadvantage is cost. The cost actually is not too limiting, because these heads are primarily used in extremely high-capacity drives, for which cost is not as much of a concern. Another disadvantage is that MR heads are much more delicate than the other head designs. Handling MR heads during manufacturing requires more care due to their greater sensitivity to damage by ESD (Electro-Static Discharge). Finally, drives with MR heads require better shielding from stray magnetic fields, which can affect these heads more easily than they do the other head designs. All in all, however, the drawbacks are minor compared with the advantages that the MR heads offer.

Head Sliders

The term *slider* is used to describe the body of material that supports the actual drive head itself. The slider is what actually floats or slides over the surface of the disk, carrying the head at the correct distance from the media for reading and writing. Most sliders resemble a catamaran, with two outboard pods that float along the surface of the disk media and a central "rudder" portion that actually carries the head and read/write gap.

The trend toward smaller and smaller form factor drives has forced a requirement for smaller and smaller sliders as well. The typical mini-Winchester slider design is about .160 × .126 × .034 inch in size. Most head manufacturers now are shifting to 50 percent smaller nanosliders, which have dimensions of about .08 × .063 × .017 inch. The nanoslider is being used in both high-capacity and small-form-factor drives. Smaller sliders reduce the mass carried at the end of the head actuator arms, allowing for increased acceleration and deceleration, and leading to faster seek times. The smaller sliders also require less area for a landing zone, thus increasing the usable area of the disk platters. Further, the smaller slider contact area reduces the slight wear on the media surface that occurs during normal startup and spindown of the drive platters.

The newer nanoslider designs also have specially modified surface patterns that are designed to maintain the same floating height above the disk surface whether the slider is above the inner or outer cylinders. Conventional sliders increase or decrease their floating height considerably, according to the velocity of the disk surface traveling below them. Above the outer cylinders, the velocity and floating height are higher. This arrangement is undesirable in newer drives that use Zoned Recording, in which the same bit density is achieved on all the cylinders. Because the same bit density is maintained throughout the drive, the head floating height should be relatively constant as well for maximum performance. Special textured surface patterns and manufacturing techniques allow the nanosliders to float at a much more consistent height, making them ideal for Zoned Recording drives.

Head Actuator Mechanisms

Possibly more important than the heads themselves is the mechanical system that moves them: the *head actuator*. This mechanism moves the heads across the disk and positions them accurately above the desired cylinder. Many variations on head actuator mechanisms are in use, but all of them can be categorized as being one of two basic types:

- Stepper motor actuators
- Voice coil actuators

The use of one or the other type of positioner has profound effects on a drive's performance and reliability. The effect is not limited to speed; it also includes accuracy, sensitivity to temperature, position, vibration, and overall reliability. To put it bluntly, a drive equipped with a stepper motor actuator is much less reliable (by a large factor) than a drive equipped with a voice coil actuator.

The head actuator is the single most important specification in the drive. The type of head actuator mechanism in a drive tells you a great deal about the drive's performance and reliability characteristics. Table 14.6 shows the two types of hard disk drive head actuators and the affected performance parameters.

Table 14.6 Characteristics of Stepper Motor versus Voice Coil Drives

Characteristic	Stepper Motor	Voice Coil
Relative access speed	Slow	Fast
Temperature sensitive	Yes (very)	No
Positionally sensitive	Yes	No
Automatic head parking	Not usually	Yes
Preventive maintenance	Periodic format	None required
Relative reliability	Poor	Excellent

Generally, a stepper motor drive has a slow average access rating, is temperature-sensitive during read and write operations, is sensitive to physical orientation during read and write operations, does not automatically park its heads above a save zone during power-down, and usually requires annual or biannual reformats to realign the sector data with the sector header information due to mistracking. Overall, stepper motor drives are vastly inferior to drives that use voice coil actuators.

Some stepper motor drives feature automatic head parking at power-down. If you have a newer stepper motor drive, refer to the drive's technical reference manual to determine whether your drive has this feature. (Other than removing the lid and watching as you power-off—which is definitely *not* recommended—reading the documentation is the only reliable way to tell.) Sometimes, you hear a noise after power-down, but that can be deceptive; some drives use a solenoid-activated spindle break, which makes a noise as the drive is powered off and does not involve head parking.

Floppy disk drives position their heads by using a stepper motor actuator. The accuracy of the stepper mechanism is suited to a floppy drive, because the track densities usually are nowhere near those of a hard disk. Many of the less expensive, low-capacity hard disks also use a stepper motor system. Most hard disks with capacities of more than 40MB have voice coil actuators, as do all drives I have seen that have capacities of more than 100MB. In IBM's product line, a drive of 40MB or more always is a voice coil drive. On IBM systems with drives of less than 40MB (usually, 20MB or 30MB), both voice coil and stepper motor drives have been used.

This breakdown does not necessarily apply to other system manufacturers, but it is safe to say that hard disk drives with less than 80MB capacity may have either type of actuator and that virtually all drives with more than 80MB capacity have voice coil actuators. The cost difference between voice coil drives and stepper motor drives of equal capacity is marginal today, so there is little reason not to use a voice coil drive. Virtually no new stepper motor drives are being manufactured today.

Stepper Motor. A *stepper motor* is an electrical motor that can "step," or move from position to position, with mechanical detents or click stop positions. If you were to grip the spindle of one of these motors and spin it by hand, you would hear a clicking or buzzing sound as the motor passed each detent position with a soft click. The sensation is much like that of the volume control on some stereo systems which use a detented type control instead of something smooth and purely linear.

Stepper motors cannot position themselves between step positions; they can stop only at the predetermined detent positions. The motors are small (between 1 and 3 inches) and can be square, cylindrical, or flat. Stepper motors are outside the sealed HDA, although the spindle of the motor penetrates the HDA through a sealed hole. The stepper motor is located in one of the corners of the hard disk drive and usually is easy to see.

Mechanical Links. The stepper motor is mechanically linked to the head rack by a split-steel band coiled around the motor spindle or by a rack-and-pinion gear mechanism. As the motor steps, each detent, or click-stop position, represents the movement of one track through the mechanical linkage.

Some systems use several motor steps for each track. In positioning the heads, if the drive is told to move to track 100, the motor begins the stepping motion, proceeds to the 100th detent position, and stops, leaving the heads above the desired cylinder. The fatal flaw in this type of positioning system is that due to dimensional changes in the platter-to-head relationship over the life of a drive, the heads may not be precisely placed above the cylinder location. This type of positioning system is called a *blind system*, because the heads have no true way of determining the exact placement of a given cylinder.

The most widely used stepper motor actuator systems use a *split-metal-band mechanism* to transmit the rotary stepping motion to the in-and-out motion of the head rack. The band is made of special alloys to limit thermal expansion and contraction as well as stretching of the thin band. One end of the band is coiled around the spindle of the stepper motor; the other is connected directly to the head rack. The band is inside the sealed HDA and is not visible from outside the drive.

Some drives use a *rack-and-pinion gear mechanism* to link the stepper motor to the head rack. This procedure involves a small pinion gear on the spindle of the stepper motor that moves a rack gear in and out. The rack gear is connected to the head rack, causing it to move. The rack-and-pinion mechanism is more durable than the split-metal-band mechanism and provides slightly greater physical and thermal stability. One problem, however, is *backlash*: the amount of play in the gears. Backlash increases as the gears wear and eventually renders the mechanism useless.

Temperature Fluctuation Problems. Stepper motor mechanisms are affected by a variety of problems. The greatest problem is temperature. As the drive platters heat and cool, they expand and contract, respectively; the tracks then move in relation to a predetermined track position. The stepper mechanism does not allow the mechanism to move in increments of less than a single track to correct for these temperature-induced errors. The drive positions the heads to a particular cylinder according to a predetermined number of steps from the stepper motor, with no room for nuance.

The low-level formatting of the drive places the initial track and sector marks on the platters at the positions where the heads currently are located, as commanded by the stepper motor. If all subsequent reading and writing occur at the same temperature as during the initial format, the heads always record precisely within the track and sector boundaries.

At different temperatures, however, the head position does not match the track position. When the platters are cold, the heads miss the track location because the platters have shrunk and the tracks have moved toward the center of the disk. When the platters are warmer than the formatted temperature, the platters will have grown larger and the track positions are located outward. Gradually, as the drive is used, the data is written inside, on top of, and outside the track and sector marks. Eventually, the drive fails to read one of these locations, and a DOS `Abort, Retry, Ignore` error message usually appears.

The temperature sensitivity of stepper motor drives also may cause the "Monday morning blues." When the system is powered up cold (on Monday, for example), a 1701, 1790, or 10490 Power-On Self Test (POST) error occurs. If you leave the system on for about 15 minutes, the drive can come up to operating temperature, and the system then may boot normally. This problem sometimes occurs in reverse when the drive gets particularly warm, such as when a system is in direct sunlight or during the afternoon, when room temperature is highest. In that case, the symptom is a DOS error message with the familiar `Abort, Retry, Ignore` prompt.

Temperature-induced mistracking problems can be solved by reformatting the drive and restoring the data. Then the information is placed on the drive at the current head positions for each cylinder. Over time, the mistracking recurs, necessitating another reformat-and-restore operation, which is a form of periodic preventive maintenance for stepper motor drives. An acceptable interval for this maintenance is once a year (or perhaps twice a year, if the drive is extremely temperature-sensitive).

Reformatting a hard drive, because it requires a complete backup-and-restore operation, is inconvenient and time-consuming. To help with these periodic reformats, most low-level-format programs offer a special reformat option that copies the data for a specific track to a spare location, reformats the track, and then copies the data back to the original track. When this type of format operation is finished, you don't have to restore your data, because the program took care of that chore for you one track at a time.

Caution

Never use a so-called nondestructive format program without first making a complete backup. This type of program *does* wipe out the data as it operates. "Destructive-reconstructive" more accurately describes its operation. If a problem occurs with the power, the system, or the program (maybe a bug that stops the program from finishing), all the data will not be restored properly, and some tracks may be wiped clean. Although such programs save you from having to restore data manually when the format is complete, they do not remove your obligation to perform a backup first.

Beware of programs whose advertising is filled with marketing hype and miracle claims for making a hard disk "better than new." One company even boasted in its advertisements that by using its program, you will "never have any problems" with your hard disk—an outrageous claim indeed! What the ads don't say is that any real low-level format program performs these same feats of "magic" without the misleading or exaggerated claims and unnecessary hype. Many of these nondestructive formatters really cannot format a large number of drives that are available today. Also, you should note that annual or biannual formatting is not necessary with voice coil actuator drives, because they do not exhibit these types of mistracking errors.

Voice Coil. A *voice coil actuator* is found in all higher quality hard disk drives, including most drives with capacities greater than 40MB and virtually all drives with capacities exceeding 80MB. Unlike the blind stepper motor positioning system, a voice coil actuator uses a feedback signal from the drive to accurately determine the head positions and to adjust them, if necessary. This system allows for significantly greater performance, accuracy, and reliability than traditional stepper motor actuators offered.

A voice coil actuator works by pure electromagnetic force. The construction of this mechanism is similar to that of a typical audio speaker, from which the term *voice coil* is derived. An audio speaker uses a stationary magnet surrounded by a voice coil connected to the speaker's paper cone. Energizing the coil causes the coil to move relative to the stationary magnet, which produces sound from the speaker cone. In a typical hard disk voice coil system, the electromagnetic coil is attached to the end of the head rack and placed near a stationary magnet. No contact is made between the coil and the magnet; instead, the coil moves by pure magnetic force. As the electromagnetic coils are energized, they attract or repulse the stationary magnet and move the head rack. Such systems are extremely quick and efficient, and usually much quieter than systems driven by stepper motors.

Unlike a stepper motor, a voice coil actuator has no click-stops, or detent positions; rather, a special guidance system stops the head rack above a particular cylinder. Because it has no detents, the voice coil actuator can slide the heads in and out smoothly to any position desired, much like the slide of a trombone. Voice coil actuators use a guidance mechanism called a *servo* to tell the actuator where the heads are in relation to the cylinders and to place the heads accurately at the desired positions. This positioning system often is called a *closed loop, servo-controlled mechanism. Closed loop* indicates that the index (or servo) signal is sent to the positioning electronics in a closed-loop system. This loop sometimes is called a *feedback loop*, because the feedback from this information is used to position the heads accurately. *Servo-controlled* refers to this index or the servo information that is used to dictate or control head-positioning accuracy.

A voice coil actuator with servo control is not affected by temperature changes, as a stepper motor is. When the temperature is cold and the platters have shrunk (or when the temperature is hot and the platters have expanded), the voice coil system compensates because it never positions the heads in predetermined track positions. Rather, the voice coil system searches for the specific track, guided by the prewritten servo information, and can position the head rack precisely above the desired track at that track's current position, regardless of the temperature. Because of the continuous feedback of servo information, the heads adjust to the current position of the track at all times. For example, as a drive warms up and the platters expand, the servo information allows the heads to "follow" the track. As a result, a voice coil actuator often is called a *track following* system.

Two main types of voice-coil positioner mechanisms are available:

- Linear voice-coil actuators
- Rotary voice-coil actuators

The types differ only in the physical arrangement of the magnets and coils.

A *linear actuator* (see fig. 14.5) moves the heads in and out over the platters in a straight line, much like a tangential-tracking turntable. The coil moves in and out on a track surrounded by the stationary magnets. The primary advantage of the linear design is that it eliminates the head azimuth variations that occur with rotary positioning systems. (*Azimuth* refers to the angular measurement of the head position relative to the tangent of a given cylinder.) A linear actuator does not rotate the head as it moves from one cylinder to another, thus eliminating this problem.

Although the linear actuator seems to be a good design, it has one fatal flaw: the devices are much too heavy. As drive performance has increased, the desire for lightweight actuator mechanisms has become very important. The lighter the mechanism, the faster it can be accelerated and decelerated from one cylinder to another. Because they are much heavier than rotary actuators, linear actuators were popular only for a short time; they are virtually nonexistent in drives manufactured today.

Rotary actuators (see fig 14.4) also use stationary magnets and a movable coil, but the coil is attached to the end of an actuator arm, much like that of a turntable's tone arm. As the coil is forced to move relative to the stationary magnet, it swings the head arms in and out over the surface of the disk. The primary advantage of this mechanism is light weight, which means that the heads can be accelerated and decelerated very quickly, resulting in very fast average seek times. The disadvantage is that as the heads move from the outer to inner cylinders, they are rotated slightly with respect to the tangent of the cylinders. This rotation results in an azimuth error and is one reason why the area of the platter in which the cylinders are located is somewhat limited. By limiting the total motion of the actuator, the azimuth error can be contained to within reasonable specifications. Virtually all voice coil drives today use rotary actuator systems.

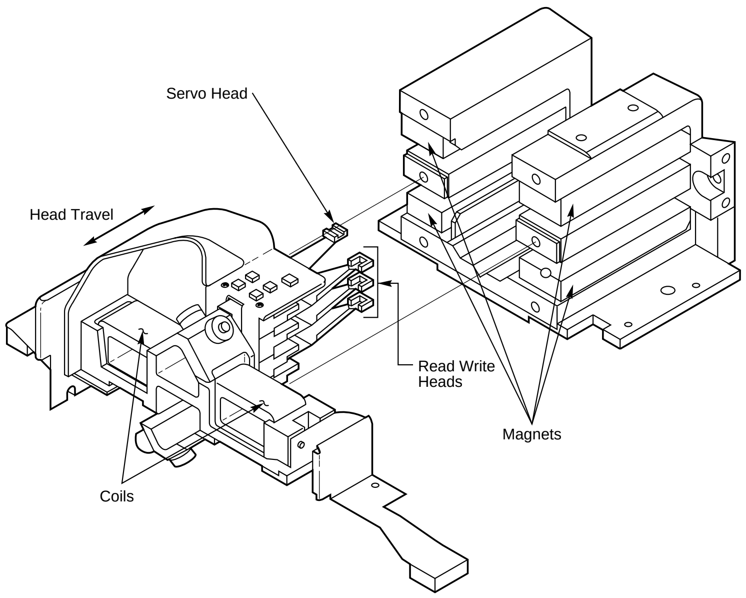


Fig. 14.5
A linear voice coil actuator.

Servo Control Mechanisms. Three servo mechanism designs have been used to control voice coil positioners over the years:

- Wedge servo
- Embedded servo
- Dedicated servo

These designs are slightly different, but they accomplish the same basic task: they enable the head positioner to adjust continuously so that it is precisely placed above a given cylinder in the drive. The main difference among these servo designs is where the gray code information is actually written on the drive.

All servo mechanisms rely on special information that is only written to the disk when the disk is manufactured. This information usually is in the form of a special code called a gray code. A *gray code* is a special binary notational system in which any two adjacent numbers are represented by a code that differs in only one bit place or column position. This system makes it easy for the head to read the information and quickly determine its precise position. This guidance code can be written only when the drive is manufactured; the code is used over the life of the drive for accurate positional information.

The servo gray code is written at the time of manufacture by a special machine called a *servowriter*: basically, a jig that mechanically moves the heads to a given reference position and then writes the servo information for that position. Many servowriters are themselves guided by a laser-beam reference that calculates its own position by calculating distances in wavelengths of light. Because the servowriter must be capable of moving the heads mechanically, this process is done with the lid of the drive off or through special access ports on the HDA. After the servowriting is complete, these ports usually are covered with sealing tape. You often see these tape-covered holes on the HDA, usually accompanied by warnings that you will void the warranty if you remove the tape. Because servowriting exposes the interior of the drive, it must be done in a clean-room environment.

A servowriter is an expensive piece of machinery, costing up to $50,000 or more, and often must be custom-made for a particular make or model of drive. Some drive-repair companies have servowriting capability, which means that they can rewrite the servo information on a drive if it becomes damaged. Lacking a servowriter, a drive with servo-code damage must be sent back to the drive manufacturer for the servo information to be rewritten.

Fortunately, it is impossible to damage the servo information through any normal reading and writing to a hard disk. Drives are designed so that servo information cannot be overwritten, even during low-level formatting of a drive. One myth that has been circulating (especially with respect to IDE drives) is that you can damage the servo information by improper low-level formatting. This is not true. An improper low-level format may compromise the performance of the drive, but the servo information is totally protected and cannot be overwritten.

The track-following capabilities of a servo-controlled voice coil actuator eliminates the positioning errors that occur over time with stepper motor drives. Voice coil drives simply are not affected by conditions such as thermal expansion and contraction of the platters. In fact, many voice coil drives today perform a special thermal-recalibration procedure at predetermined intervals while they run. This procedure usually involves seeking the heads from cylinder 0 to some other cylinder one time for every head on the drive. As this sequence occurs, the control circuitry in the drive monitors how much the track positions have moved since the last time the sequence was performed, and a thermal calibration adjustment is calculated and stored in the drive's memory. This information then is used every time the drive positions to ensure the most accurate positioning possible.

Most drives perform the thermal-recalibration sequence every five minutes for the first half-hour that the drive is powered on and then once every 25 minutes after that. With some drives (such as Quantum, for example), this thermal-calibration sequence is very noticeable; the drive essentially stops what it is doing, and you hear rapid ticking for a second or so. At this time, some people think that their drive is having a problem reading something and perhaps is conducting a read retry, but this is not true. Most of the newer intelligent drives (IDE and SCSI) employ this thermal-recalibration procedure for ultimate positioning accuracy.

While we are on the subject of automatic drive functions, most of the drives that perform thermal-recalibration sequences also automatically perform a function called a *disk sweep*. This procedure is an automatic head seek that occurs after the drive has been idle for a period of time (for example, nine minutes). The disk-sweep function moves the heads to a random cylinder in the outer portion of the platters, which is considered to be the high float-height area because the head-to-platter velocity is highest. Then, if the drive continues to remain idle for another period, the heads move to another cylinder in this area, and the process continues indefinitely as long as the drive is powered on.

The disk-sweep function is designed to prevent the head from remaining stationary above one cylinder in the drive, where friction between the head and platter eventually would dig a trench in the media. Although the heads are not in direct contact with the media, they are so close that the constant air pressure from the head floating above a single cylinder causes friction and excessive wear.

Wedge Servo. Some early servo-controlled drives used a technique called a *wedge servo*. In these drives, the gray-code guidance information is contained in a "wedge" slice of the drive in each cylinder immediately preceding the index mark. The index mark indicates the beginning of each track, so the wedge-servo information was written in the Pre-Index Gap, which is at the end of each track. This area is provided for speed tolerance and normally is not used by the controller. Figure 14.6 shows the wedge-servo information on a drive.

Some controllers, such as the Xebec 1210 that IBM used in the XT, had to be notified that the drive was using a wedge servo so that they could shorten the sector timing to allow for the wedge-servo area. If they were not correctly configured, these controllers would not work properly with such drives. Many people believed—erroneously—that the wedge-servo information could be overwritten in such cases by an improper low-level format. This is not the case, however; all drives using a wedge servo disable any write commands and take control of the head select lines whenever the heads are above the wedge area. This procedure protects the servo from any possibility of being overwritten, no matter how hard you try. If the controller tried to write over this area, the drive would prevent the write, and the controller would be unable to complete the format. Most controllers simply do not write to the Pre-Index Gap area and do not need to be configured specially for wedge-servo drives.

The only way that the servo information normally could be damaged is by a powerful external magnetic field (or perhaps by a head crash or some other catastrophe). In such a case, the drive would have to be sent in for repair.

One problem is that the servo information appears only one time every revolution, which means that the drive often needs several revolutions before it can accurately determine and adjust the head position. Because of these problems, the wedge servo never was a popular design; it no longer is used in drives.

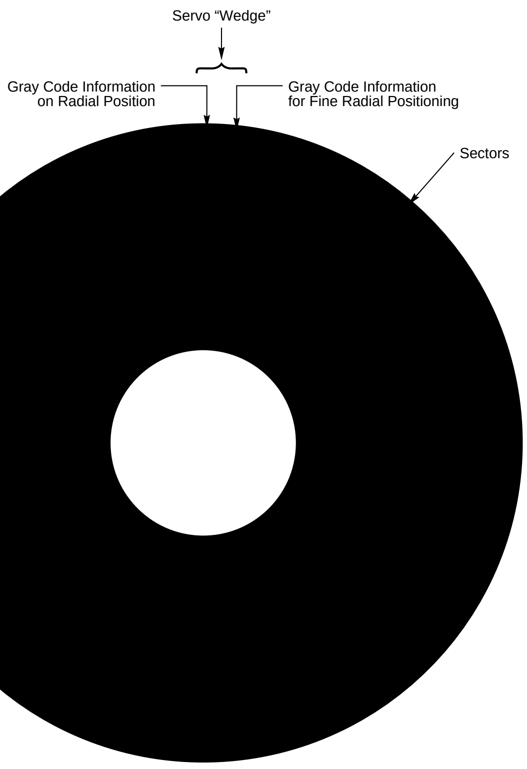

Servo "Wedge"
Gray Code Information on Radial Position
Gray Code Information for Fine Radial Positioning
Sectors

Fig. 14.6
A wedge servo.

Embedded Servo. An *embedded servo* (see fig. 14.7) is an enhancement of the wedge servo. Instead of placing the servo code before the beginning of each cylinder, an embedded servo design writes the servo information before the start of each sector. This arrangement enables the positioner circuits to receive feedback many times in a single revolution, making the head positioning much faster and more precise. Another advantage is that every track on the drive has this positioning information, so each head can quickly and efficiently adjust position to compensate for any changes in the platter or head dimensions, especially for changes due to thermal expansion or physical stress.

Most drives today use an embedded servo to control the positioning system. As in the wedge servo design, the embedded-servo information is protected by the drive circuits, and any write operations are blocked whenever the heads are above the servo information. Thus, it is impossible to overwrite the servo information with a low-level format, as many people incorrectly believed.

Although the embedded servo works much better than the wedge servo, because the feedback servo information is available several times in a single disk revolution, a system that offered continuous servo feedback information would be better.

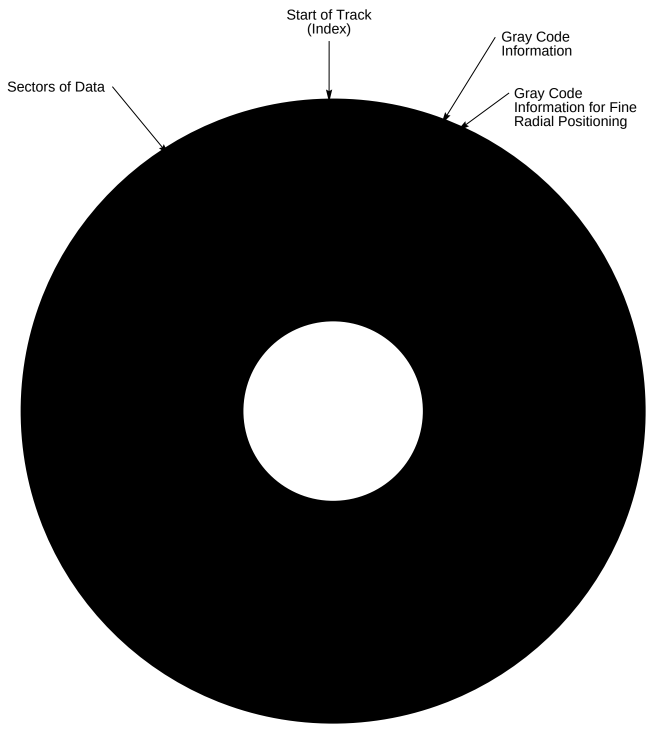


Fig. 14.7
An embedded servo.

Dedicated Servo. A *dedicated servo* is a design in which the servo information is written continuously throughout the entire track, rather than just one time per track or at the beginning of each sector. Unfortunately, if this procedure were performed on the entire drive, no room would be left for data. For this reason, a dedicated servo uses one side of one of the platters exclusively for the servo-positioning information. The term *dedicated* comes from the fact that this platter side is completely dedicated to the servo information and cannot contain any data. Although the dedicated-servo design may seem to be wasteful, none of the other platter sides carry any servo information, and you end up losing about the same amount of total disk real estate as with the embedded servo.

When a dedicated-servo drive is manufactured, one side of one platter is deducted from normal read/write usage; on this platter is recorded a special set of gray-code data that indicates proper track positions. Because the head that rests above this surface cannot be used for normal reading and writing, these marks can never be erased, and the servo information is protected, as in the other servo designs. No low-level format or other procedure can possibly overwrite the servo information.

When the drive is commanded to move the heads to a specific cylinder, the internal drive electronics use the signals received by the servo head to determine the position of the heads. As the heads are moved, the track counters are read from the dedicated servo surface. When the requested track is detected below the servo head, the actuator is stopped. The servo electronics then fine-tune the position so that before writing is allowed, the heads are positioned absolutely precisely above the desired cylinder. Although only one head is used for servo tracking, the other heads are attached to the same rigid rack, so if one head is above the desired cylinder, all the others will be as well.

One noticeable trait of dedicated servo drives is that they usually have an odd number of heads. For example, the Toshiba MK-538FB 1.2GB drive on which I am saving this chapter has eight platters but only 15 read/write heads; the drive uses a dedicated-servo positioning system, and the 16th head is the servo head. You will find that virtually all high-end drives use a dedicated servo because such a design offers servo information continuously, no matter where the heads are located. This system offers the greatest possible positioning accuracy. Some drives even combine a dedicated servo with an embedded servo, but this type of hybrid design is rare.

Automatic Head Parking. When a hard disk drive is powered off, the spring tension in each head arm pulls the heads into contact with the platters. The drive is designed to sustain thousands of takeoffs and landings, but it is wise to ensure that the landing occurs at a spot on the platter that contains no data. Some amount of abrasion occurs during the landing and takeoff process, removing just a "micro puff" from the media; but if the drive is jarred during the landing or takeoff process, real damage can occur.

One benefit of using a voice coil actuator is *automatic head parking*. In a drive that has a voice coil actuator, the heads are positioned and held by magnetic force. When power is removed from the drive, the magnetic field that holds the heads stationary over a particular cylinder dissipates, enabling the head rack to skitter across the drive surface and potentially cause damage. In the voice coil design, therefore, the head rack is attached to a weak spring at one end and a head stop at the other end. When the system is powered on, the spring normally is overcome by the magnetic force of the positioner. When the drive is powered off, however, the spring gently drags the head rack to a park-and-lock position before the drive slows down and the heads land. On many drives, you can actually hear the "ting…ting…ting…ting" sound as the heads literally bounce-park themselves, driven by this spring.

On a drive with a voice coil actuator, you can activate the parking mechanism simply by turning off the system; you do not need to run a program to park or retract the heads. In the event of a power outage, the heads park themselves automatically. (The drives unpark automatically when the system is powered on.)

Some stepper motor drives (such as the Seagate ST-251 series drives) park their heads, but this function is rare among stepper motor drives. The stepper motor drives that do park their heads usually use an ingenious system whereby the spindle motor actually is used as a generator after the power to the drive is turned off. The back EMF (Electro Motive Force), as it is called, is used to drive the stepper motor to park the heads.

Air Filters

Nearly all hard disk drives have two air filters. One filter is called the *recirculating filter*, and the other is called either a *barometric* or *breather filter*. These filters are permanently sealed inside the drive and are designed never to be changed for the life of the drive, unlike many older mainframe hard disks that had changeable filters. Many mainframe drives circulate air from outside the drive through a filter that must be changed periodically.

A hard disk on a PC system does not circulate air from inside to outside the HDA, or vice versa. The recirculating filter that is permanently installed inside the HDA is designed to filter only the small particles of media scraped off the platters during head takeoffs and landings (and possibly any other small particles dislodged inside the drive). Because PC hard disk drives are permanently sealed and do not circulate outside air, they can run in extremely dirty environments (see fig. 14.8).

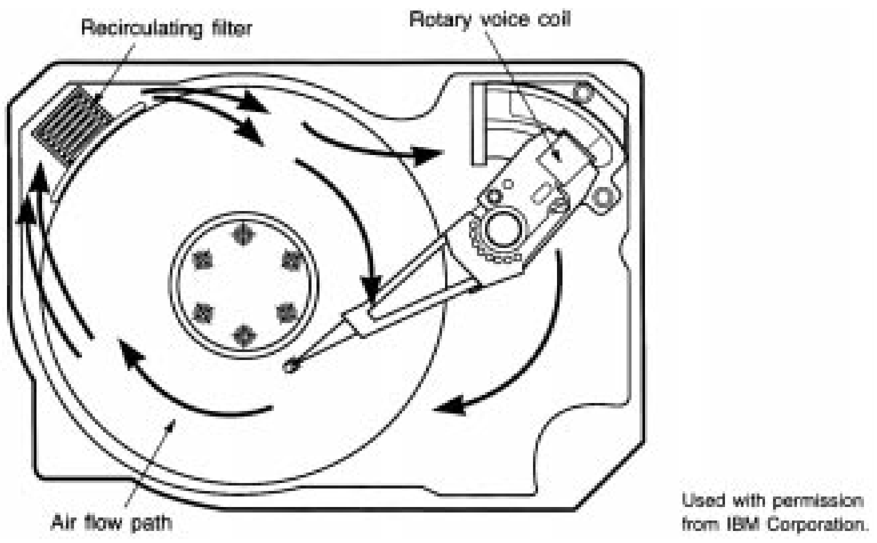


Fig. 14.8
Air circulation in a hard disk.

The HDA in a hard disk is sealed but not airtight. The HDA is vented through a barometric or breather filter element that allows for pressure equalization (breathing) between the inside and outside of the drive. For this reason, most hard drives are rated by the drive's manufacturer to run in a specific range of altitudes, usually from –1,000 to +10,000 feet above sea level. In fact, some hard drives are not rated to exceed 7,000 feet while operating, because the air pressure would be too low inside the drive to float the heads properly. As the environmental air pressure changes, air bleeds into or out of the drive so that internal and external pressures are identical. Although air does bleed through a vent, contamination usually is not a concern, because the barometric filter on this vent is designed to filter out all particles larger than 0.3 micron (about 12 micro-inches) to meet the specifications for cleanliness inside the drive. You can see the vent holes on most drives, which are covered internally by this breather filter. Some drives use even finer-grade filter elements to keep out even smaller particles.

I got a laugh when I read an article in one of the better known computer magazines, stating not only that hard drives are airtight, but also that the air is evacuated from the interior of the drive, and the heads and platters run in a vacuum! The person who wrote the article obviously does not understand even the most basic principles of hard disk operation. Air is required inside the HDA to float the heads, and this cushion of air (sometimes called an air bearing) is the primary principle in Winchester hard disk design.

I conducted a seminar in Hawaii several years ago, and several of the students were from the Mauna Kea astronomical observatory. They indicated that virtually all hard disks they had tried to use at the observatory site had failed very quickly, if they worked at all. This was no surprise, because the observatory is at the 13,800-foot peak of the mountain, and at that altitude, even people don't function very well! At the time, it was suggested that the students investigate solid-state (RAM) disks, tape drives, or even floppy drives as their primary storage medium. Since this time, IBM's Adstar division (which produces all IBM hard drives) introduced a line of rugged 3.5-inch drives that are in fact hermetically sealed (airtight), although they do have air inside the HDA. Because they carry their own internal air under pressure, these drives can operate at any altitude, and also can withstand extremes of shock and temperature. The drives are designed for military and industrial applications, such as aboard aircraft and in extremely harsh environments.

Caution

Airborne particulates such as cigarette smoke normally do not affect a PC hard disk drive, because any air that bleeds into the hard drive is filtered before entering the drive. However, many other components of the system (such as floppy drives, keyboards, connectors, and sockets) will sustain damage from contaminants such as cigarette smoke.

Hard Disk Temperature Acclimation

Although the HDA is sealed, it is not hermetically sealed, which means that it is not airtight and that there is air inside. To allow for pressure equalization, hard drives have a filtered port to bleed air into or out of the HDA as necessary.

This breathing also enables moisture to enter the drive, and after some period of time, it must be assumed that the humidity inside any hard disk is similar to that outside the drive. Humidity can become a serious problem if it is allowed to condense—and especially if the drive is powered up while this condensation is present. Most hard disk manufacturers have specified procedures for acclimating a hard drive to a new environment with different temperature and humidity ranges, especially for bringing a drive into a warmer environment in which condensation can form. This situation should be of special concern to users of laptop or portable systems with hard disks. If you leave a portable system in an automobile trunk during the winter, for example, it could be catastrophic to bring the machine inside and power it up without allowing it to acclimate to the temperature indoors.

The following text and Table 14.7 are taken from the factory packaging that Control Data Corporation (later Imprimis and eventually Seagate) used to ship its hard drives:

> If you have just received or removed this unit from a climate with temperatures at or below 50°F (10°C) do not open this container until the following conditions are met, otherwise condensation could occur and damage to the device and/or media may result. Place this package in the operating environment for the time duration according to the temperature chart.

Table 14.7 Hard Disk Drive Environmental Acclimation Table

Previous Climate Temp.	Acclimation Time
+40°F (+4°C)	13 hours
+30°F (–1°C)	15 hours
+20°F (–7°C)	16 hours
+10°F (–12°C)	17 hours
0°F (–18°C)	18 hours
–10°F (–23°C)	20 hours
–20°F (–29°C)	22 hours
–30°F (–34°C) or less	27 hours

As you can see from this chart, a hard disk that has been stored in a colder-than-normal environment must be placed in the normal operating environment for a specified amount of time to allow for acclimation before it is powered on.

Spindle Motors

The motor that spins the platters is called the *spindle motor* because it is connected to the spindle around which the platters revolve. Spindle motors in hard disks always are connected directly; no belts or gears are used. The motors must be free of noise and vibration; otherwise, they transmit to the platters a rumble that could disrupt reading and writing operations.

The motors also must be precisely controlled for speed. The platters on hard disks revolve at speeds ranging from 3,600 to 7,200 rpm or more, and the motor has a control circuit with a feedback loop to monitor and control this speed precisely. Because this speed control must be automatic, hard drives do not have a motor-speed adjustment. Some diagnostics programs claim to measure hard drive rotation speed, but all that these programs do is estimate the rotational speed by the timing at which sectors arrive.

There actually is no way for a program to measure hard disk rotational speed; this measurement can be made only with sophisticated test equipment. Don't be alarmed if some diagnostic program tells you that your drive is spinning at an incorrect speed; most likely the program is wrong, not the drive. Platter rotation and timing information is simply not provided through the hard disk controller interface. In the past, software could give approximate rotational speed estimates by performing multiple sector read requests and timing them, but this was valid only when all drives had the same number of sectors per track (17) and they all spun at 3,600 rpm. Zoned Recording combined with a variety of different rotational speeds found in modern drives, not to mention built-in buffers and caches means that these calculation estimates cannot be performed accurately.

On most drives, the spindle motor is on the bottom of the drive, just below the sealed HDA. Many drives today, however, have the spindle motor built directly into the platter hub inside the HDA. By using an internal hub spindle motor, the manufacturer can stack more platters in the drive, because the spindle motor takes up no vertical space. This method allows for more platters than would be possible if the motor were outside the HDA.

Note

Spindle motors, particularly on the larger form-factor drives, can consume a great deal of 12-volt power. Most drives require two to three times the normal operating power when the motor first spins the platters. This heavy draw lasts only a few seconds, or until the drive platters reach operating speed. If you have more than one drive, you should try to sequence the start of the spindle motors so that the power supply does not receive such a large load from all the drives at the same time. Most SCSI and IDE drives have a delayed spindle-motor start feature.

Spindle Ground Strap

Most drives have a special grounding strap attached to a ground on the drive and resting on the center spindle of the platter spindle motor. This device is the single most likely cause of excessive drive noise.

The *grounding strap* usually is made of copper and often has a carbon or graphite button that contacts the motor or platter spindle. The grounding strap dissipates static generated by the platters as they spin through the air inside the HDA. If the platters generate static due to friction with the air, and if no place exists for this electrical potential to bleed off, static may discharge through the heads or the internal bearings in the motor. When static discharges through the motor bearings, it can burn the lubricants inside the sealed bearings. If the static charge discharges through the read/write heads, the heads can be damaged or data can be corrupted. The grounding strap bleeds off this static buildup to prevent these problems.

Where the spindle of the motor contacts the carbon contact button (at the end of the ground strap) spinning at full speed, the button often wears, creating a flat spot. The flat spot causes the strap to vibrate and produce a high-pitched squeal or whine. The noise may come and go, depending on temperature and humidity. Sometimes, banging the side of the machine can jar the strap so that the noise changes or goes away, but this is not the way to fix the problem. *I am not suggesting that you bang on your system!* (Most people mistake this noise for something much more serious, such as a total drive-motor failure or bearing failure, which rarely occurs.)

If the spindle grounding strap vibrates and causes noise, you can remedy the situation in several ways:

- Dampen the vibration of the strap by attaching some foam tape or rubber to it.
- Lubricate the contact point.
- Tear off the strap (not recommended!).

On some drives, the spindle motor strap is easily accessible. On other drives, you have to partially disassemble the drive by removing the logic board or other external items to get to the strap.

Of these suggested solutions, the first one is the best. The best way to correct this problem is to glue (or otherwise affix) some rubber or foam to the strap. This procedure changes the harmonics of the strap and usually dampens vibrations. Most manufacturers now use this technique on newly manufactured drives. An easy way to do this is to place some foam tape on the back side of the ground strap.

You also can use a dab of silicone RTV (room-temperature vulcanizing) rubber or caulk on the back of the strap. If you try this method, be sure to use low-volatile (noncorrosive) silicone RTV sealer, which commonly is sold at auto-parts stores. The noncorrosive silicone will be listed on the label as being safe for automotive oxygen sensors. This low-volatile silicone also is free from corrosive acids that can damage the copper strap and is described as low-odor because it does not have the vinegar odor usually associated with silicone RTV. Dab a small amount on the back side of the copper strap (do not interfere with the contact location), and the problem should be solved permanently.

Lubricating the strap is an acceptable, but often temporary, solution. You will want to use some sort of conducting lube, such as a graphite-based compound (the kind used on frozen car locks). Any conductive lubricant (such as moly or lithium) will work as long as it is conductive, but do not use standard oil or grease. Simply dab a small amount of lubricant onto the end of a toothpick, and place a small drop directly on the point of contact.

The last solution is not acceptable. Tearing off the strap eliminates the noise, but it has several possible ramifications. Although the drive will work (silently) without the strap, an engineer placed it there for a reason. Imagine those ungrounded static charges leaving the platters through the heads, perhaps in the form of a spark—possibly even damaging the Thin Film heads. You should choose one of the other solutions.

I mention this last solution only because several people have told me that members of the tech-support staff of some of the hard drive vendors, and even of some manufacturers, told them to remove the strap, which—of course—I do not recommend.

Logic Boards

A disk drive, including a hard disk drive, has one or more logic boards mounted on it. The logic boards contain the electronics that control the drive's spindle and head actuator systems and that present data to the controller in some agreed-on form. In some drives, the controller is located on the drive, which can save on a system's total chip count.

Many disk drive failures occur in the logic board, not in the mechanical assembly. (This statement does not seem logical, but it is true.) Therefore, you can repair many failed drives by replacing the logic board, not the entire drive. Replacing the logic board, moreover, enables you to regain access to the data on the failed drive—something that replacing the entire drive precludes.

Logic boards can be removed or replaced because they simply plug into the drive. These boards usually are mounted with standard screw hardware. If a drive is failing and you have a spare, you may be able to verify a logic-board failure by taking the board off the known good drive and mounting it on the bad one. If your suspicions are confirmed, you can order a new logic board from the drive manufacturer. You also may be able to purchase a refurbished unit or even to trade in your old drive or logic board. The drive manufacturer will have details on what services it can offer.

To reduce costs further, many third-party vendors also can supply replacement logic-board assemblies. These companies often charge much less than the drive manufacturers for the same components. (See the "Vendor List" in Appendix B for vendors of drive components, including logic boards.)

Cables and Connectors

Most hard disk drives have several connectors for interfacing to the system, receiving power, and sometimes grounding to the system chassis. Most drives have at least these three types of connectors:

- Interface connector(s)
- Power connector
- Optional ground connector (tab)

Of these, the interface connectors are the most important, because they carry the data and command signals from the system to and from the drive. In many drive interfaces, the drive interface cables can be connected in a *daisy chain*, or bus-type configuration. Most interfaces support at least two drives, and SCSI (Small Computer System Interface) supports up to seven in the chain. Some interfaces, such as ST-506/412 or ESDI (Enhanced Small Device Interface), use a separate cable for data and control signals. These drives have two cables from the controller interface to the drive. SCSI and IDE (Integrated Drive Electronics) drives usually have a single data and control connector. With these interfaces, the disk controller is built into the drive (see fig. 14.9).

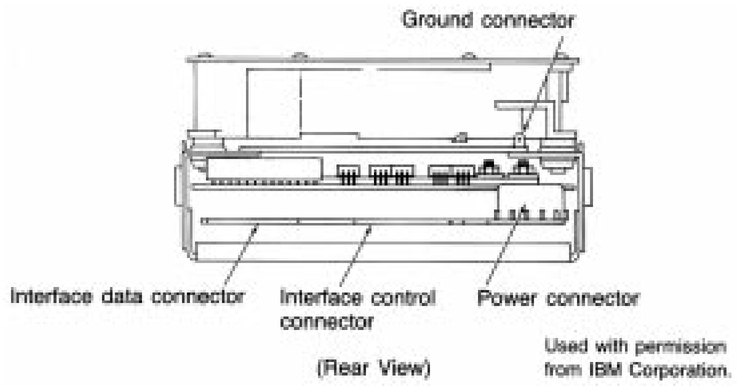

Ground connector
Interface data connector
Interface control connector
Power connector
(Rear View)
Used with permission from IBM Corporation.

Fig. 14.9
Typical hard disk connections (ST-506/412 or ESDI shown).

The different interfaces and cable specifications are covered in the sections on drive interfaces later in this chapter. You also will find connector pinout specifications for virtually all drive interfaces and cable connections in this chapter.

The *power connector* usually is the same type that is used in floppy drives, and the same power-supply connector plugs into it. Most hard disk drives use both 5- and 12-volt power, although some of the smaller drives designed for portable applications use only 5-volt power. In most cases, the 12-volt power runs the spindle motor and head actuator, and the 5-volt power runs the circuitry. Make sure that your power supply can supply adequate power for the hard disk drives installed in your PC system; most hard drives draw quite a bit more power than a floppy drive.

The 12-volt-power consumption of a drive usually varies with the physical size of the unit. The larger the drive is and the more platters there are to spin, the more power is required. Also, the faster the drive spins, the more power will be required. For example, most of the 3.5-inch drives on the market today use roughly one-half to one-fourth the power (in watts) of the full-size 5.25-inch drives. Some of the very small (2.5-, 1.8-, or 1.3-inch) hard disks barely sip electrical power and actually use 1 watt or less!

Ensuring an adequate power supply is particularly important with some systems, such as the original IBM AT. These systems have a power supply with three disk drive power connectors, labeled P10, P11, and P12. The three power connectors may seem to be equal, but the technical-reference manual for these systems indicates that 2.8 amps of 12-volt current is available on P10 and P11 and that only 1 amp of 12-volt current is available on P12. Because most full-height hard drives draw much more power than 1 amp, especially at startup, the P12 connector can be used only by floppy drives or half-height hard drives. Some 5.25-inch drives draw as much as 4 amps of current during the first few seconds of startup. These drives also can draw as much as 2.5 amps during normal operation.

Sometimes, you can solve random boot-up failures simply by plugging the hard drive into a suitable power connector (P10 or P11 on the IBM AT). Most IBM-compatible PC systems have a power supply with four or more disk drive power connectors that provide equal power, but some use power supplies designed like those of the IBM AT.

A *grounding tab* provides a positive ground connection between the drive and the system's chassis. In a typical IBM PC or IBM XT system, because the hard disk drive is mounted directly to the chassis of the PC using screws, the ground wire is unnecessary. On AT-type systems from IBM and other manufacturers, the drives are installed on plastic or fiberglass rails, which do not provide proper grounding. These systems must provide a grounding wire, plugged into the drive at this grounding tab. Failure to ground the drive may result in improper operation, intermittent failure, or general read and write errors.

Configuration Items

To configure a hard disk drive for installation in a system, several jumpers (and, possibly, terminating resistors) usually must be set or configured properly. These items will vary from interface to interface and often from drive to drive as well. A complete discussion of the configuration settings for each interface appears in "Hard Disk Installation Procedures" later in this chapter.

The Faceplate or Bezel

Most hard disk drives offer as an option a front faceplate, or *bezel* (see fig. 14.10). A bezel usually is supplied as an option for the drive rather than as a standard item.

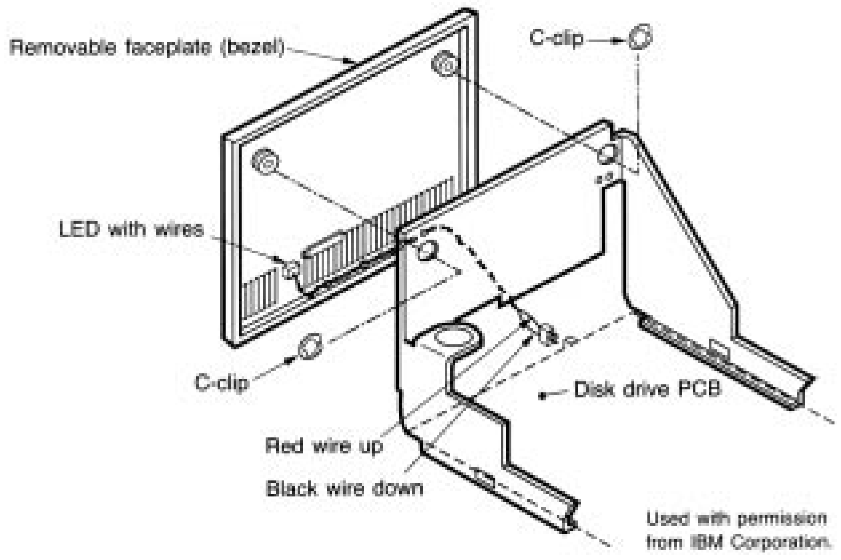

Removable faceplate (bezel)
C-clip
LED with wires
C-clip
Disk drive PCB
Red wire up
Black wire down
Used with permission from IBM Corporation.

Fig. 14.10
A typical hard drive faceplate (bezel).

Bezels often come in several sizes and colors to match various PC systems. For standard full-height, 5.25-inch, form-factor drives, you have only one choice of bezel. For half-height drives, bezels come in half-height and full-height forms. Using a full-height bezel on a half-height drive enables you to install a single drive in a full-height bay without leaving a hole in the front of the system. To add a second half-height drive, you may want to order the half-height bezels so that you can stack the old and new drives. Many faceplate configurations for 3.5-inch drives are available, including bezels that fit 3.5-inch drive bays as well as 5.25-inch drive bays. You even have a choice of colors (usually, black, cream, or white).

Some bezels feature a light-emitting diode (LED) that flickers when your hard disk is in use. The LED is mounted in the bezel; the wire hanging off the back of the LED plugs into the drive or perhaps the controller. In some drives, the LED is permanently mounted on the drive, and the bezel has a clear or colored window so that you can see the LED flicker while the drive is accessed.

One type of LED problem occurs with some AT-type-system hard disk installations: if the drive has an LED, the LED may remain on continuously, as though it were a “power-on” light rather than an access light. This problem happens because the controller in the AT has a direct connection for the LED, thus altering the drive LED function. Some controllers have a jumper that enables the controller to run the drive in what is called latched or unlatched mode. *Latched mode* means that the drive is selected continuously and that the drive LED remains lighted; in *unlatched mode* (to which we are more accustomed), the LED lights only when the drive is accessed. Check to see whether your controller has a jumper for changing this function; if so, you may be able to control the way that the LED operates.

In systems in which the hard disk is hidden by the unit’s cover, a bezel is not needed. In fact, using a bezel may prevent the cover from resting on the chassis properly, in which case the bezel will have to be removed. If you are installing a drive that does not have a proper bezel, frame, or rails to attach to the system, check Appendix B of this book; several listed vendors offer these accessories for a variety of drives.

Hard Disk Features

To make the best decision in purchasing a hard disk for your system, or to understand what differentiates one brand of hard disk from another, you must consider many features. This section examines the issues that you should consider when you evaluate drives:

- Actuator mechanism
- Media
- Head parking
- Reliability
- Speed
- Shock mounting
- Cost

Actuator Mechanism

A drive with high performance and reliability has two basic physical properties:

- Voice coil actuator mechanism
- Thin-film media

Drives with stepper motor actuators should be used only when cost far outweighs other considerations. You should not use these drives in portable systems or in systems that must operate under extreme temperature, noise, or vibration conditions. Don't use these drives where preventive maintenance cannot be provided, because they require periodic reformats to maintain data integrity. Finally, you should not use these drives in demanding situations, such as in a network file server. Drives with stepper motor actuators perform adequately in low-volume-usage systems, as long as you provide preventive maintenance at least annually or semiannually and the environment can be controlled. Fortunately, stepper motor drives are virtually out of production today; nearly all new drives use voice coil actuators.

Voice coil actuator drives should be used wherever possible, especially if any real demands are placed on the drive. These drives are ideal for portable systems or for systems that suffer extreme temperatures, noise, or vibration. These drives are ideal when a fast drive is necessary. A voice coil drive requires little or no preventive maintenance, so the first low-level format usually is the only low-level format ever required. Less maintenance (no reformatting) enables this type of drive to be used for high-volume situations in which a single support person maintains many PC systems. Fortunately, virtually all drives manufactured today are voice coil drives, and they should be! Voice coil actuator technology is the only way to build a reliable, high-performance, low-maintenance drive.

Head Parking

Head parking is an often-misunderstood issue with hard disks. When a hard disk comes to a stop (actually, before it stops), the heads land on the media. This contact occurs as the drive slows, and in some drives, the distance that the heads skid before the platters stop may be many linear feet. The same skidding occurs when the drive is powered on and the platters begin to turn. In some drives, the heads land on whatever cylinder they were last positioned above—usually, an area of the disk that contains data.

Most drives today move the heads to a nondata area called the *landing zone* before the platters slow enough for the heads to come into contact with them. This procedure is called *automatic head parking*. Drives that do not park their heads automatically still can be parked, but it requires the manual execution of a program to move the heads before the system is powered down.

Drives with voice coil actuators offer automatic head parking. While the drive is running, an electric coil overcomes the spring tension and moves the head around the disk. When power is lost, the spring automatically pulls the head rack away from data areas of the disk to a special landing zone.

Most stepper motor drives do not have an automatic-parking function; instead they must be parked manually. To find out whether your drive autoparks, contact the drive manufacturer and ask for the technical or specification manual for the drive, which will contain the answer.

Some newer stepper drives do incorporate a parking mechanism. One example is the Seagate ST-251 series. This popular stepper drive autoparks the heads by using an ingenious system in which the drive spindle motor is used as a generator, powering the stepper motor to park the heads. When the drive is powered off, you hear the stepper motor drive the heads to the landing zone. Seagate seems to be using this type of mechanism in many of its newer stepper motor drives as well.

Software is available that enables you to park the heads of drives that lack the automatic-parking feature. The software is not quite as reliable as automatic parking, however, because it does not park the heads if the power goes off unexpectedly. If your system requires a head-parking program, the program usually comes on the configuration or setup utility disk that goes with the system. For example, IBM supplied a head-parking routine on the diagnostics disk supplied with its original XT and AT systems, as well as on the Advanced Diagnostics disk supplied with the hardware-maintenance service manual. Simply boot these disks and select the option Prepare System for Moving. This procedure invokes a program on the disk called SHIPDISK.COM. Different SHIPDISK.COM files exist for XT and AT systems. The heads of all attached disks are parked. Then you shut down the system.

As a note, several years ago IBM issued a warning to its dealers, recommending that they not run SHIPDISK.COM from the DOS prompt. IBM said that a slight chance exists that you can lose data, because the program can accidentally write random data on the drive. The memo indicated that SHIPDISK.COM should be run only from the menu. Apparently, the problem was that SHIPDISK.COM parks the disks and then executes a software interrupt to return to the diagnostics-disk menu, and unpredictable things can happen (including stray disk writes) if the program was not run from the diagnostics-disk menu.

For AT systems, IBM supplied a separate program, SHUTDOWN.EXE, that is designed to be run from the DOS prompt. This program is on the AT diagnostics and advanced-diagnostics disks. You can copy this program to the hard disk and enter the SHUTDOWN command at the DOS prompt. You see a graphic of a switch, which turns off as the heads are parked. The program then halts the system, requiring a complete power-down. This program works only on AT-type systems.

Note

It usually is not a good idea to run a hard disk parking program that is not designed for your system. Although no physical harm will occur, the heads may not be parked in the correct landing zone. (Many programs improperly position the heads above Cylinder 0—the last place where you want them!)

If your hard drive is a stepper motor actuator drive without automatic parking, it should come with a parking program. If you have an IBM system, this program comes on the diagnostics and setup disks that came with the system. If you have an IBM-compatible, you probably received such a program on your setup disk. Additionally, some public-domain programs park a stepper motor hard disk.

Should you park the heads every time you shut down the drive? Some people think so, but IBM says that you do not have to park the heads on a drive unless you are moving the drive. My experiences are in line with IBM's recommendations, although a more fail-safe approach is to park the heads at every shutdown. Remember that voice coil drives park their heads automatically every time and require no manual parking operations. I only park a stepper motor drive if I am moving the unit, but there is nothing wrong with parking the heads at every shutdown. The procedure is simple and cannot hurt.

Reliability

When you shop for a drive, you may notice a feature called the *mean time between failures* (MTBF) described in the brochures. MTBF figures usually range from 20,000 hours to 500,000 hours or more. I usually ignore these figures, because they usually are just theoretical—not actual—statistical values. Most drives that boast these figures have not even been manufactured for that length of time. One year of 5-day work weeks with 8-hour days equals 2,080 hours of operation. If you never turn off your system for 365 days and run the full 24 hours per day, you operate your system 8,760 hours each year; a drive with a 500,000-hour MTBF rating is supposed to last (on average) 57 years before failing! Obviously, that figure cannot be derived from actual statistics, because the particular drive probably has been on the market for less than a year.

Statistically, for the MTBF figures to have real weight, you must take a sample of drives, measure the failure rate for at least twice the rated figure, and measure how many drives fail in that time. To be really accurate, you would have to wait until all the drives fail and record the operating hours at each failure. Then you would average the running time for all the test samples to arrive at the average time before a drive failure. For a reported MTBF of 500,000 hours (common today), the test sample should be run for at least 1 million hours (114 years) to be truly accurate, yet the drive carries this specification on the day that it is introduced.

Manufacturers and vendors sometimes play with these numbers. For example, several years ago, CDC rated a Wren II half-height drive at 20,000 hours MTBF (this drive was one of the most reliable in the world at the time), but I saw a reseller rate the same unit at 50,000 hours. Some of the worst drives that I have used boasted high MTBF figures, and some of the best drives have lower ones. These figures do not necessarily translate to reliability in the field, and that is why I generally place no importance on them.

Performance

When you select a hard disk, an important feature to consider is the performance (speed) of the drive. Hard disks come in a wide range of performance capabilities. As is true of many things, one of the best indicators of a drive's relative performance is its price. An old saying from the automobile-racing industry is appropriate here: "Speed costs money. How fast do you want to go?"

You can measure the speed of a disk drive in two ways:

- Average seek time
- Transfer rate

Average seek time, normally measured in milliseconds, is the average amount of time it takes to move the heads from one cylinder to another cylinder a random distance away. One way to measure this specification is to run many random track-seek operations and then divide the timed results by the number of seeks performed. This method provides an average time for a single seek.

The standard way to measure average seek time used by many drive manufacturers involves measuring the time that it takes the heads to move across one-third of the total cylinders. Average seek time depends only on the drive; the type of interface or controller has little effect on this specification. (In some cases, the setup of the controller to the drive can affect seek times; this subject is discussed later in this chapter.) The rating is a gauge of the capabilities of the head actuator.

Be wary of benchmarks that claim to measure drive seek performance. Most IDE and SCSI drives use a scheme called sector translation, so any commands to the drive to move the heads to a specific cylinder do not actually cause the intended physical movement. This situation renders such benchmarks meaningless for those types of drives. SCSI drives also require an additional command, because the commands first must be sent to the drive over the SCSI bus. Even though these drives can have the fastest access times, because the command overhead is not factored in by most benchmarks, the benchmark programs produce poor performance figures for these drives.

I don't put too much faith in the benchmarks, and the drive manufacturers have been very honest in reporting their true performance figures over the years. The bottom line is that if you want to know the true seek performance of your drive, the most accurate way to find it is simply to look it up in the drive specification manual.

A slightly different measurement, called average access time, involves another element, called latency. *Latency* is the average time (in milliseconds) that it takes for a sector to be available after the heads have reached a track. On average, this figure is half the time that it takes for the disk to rotate one time, which is 8.33 ms at 3,600 rpm. A drive that spins twice as fast would have half the latency. A measurement of average access time is the sum of the average seek time and latency. This number provides the average amount of time required before a randomly requested sector can be accessed.

Latency is a factor in disk read and write performance. Decreasing the latency increases the speed of access to data or files, accomplished only by spinning the drive platters faster. I have a drive that spins at 4,318 rpm, for a latency of 6.95 ms. Some drives spin at 7,200 rpm or faster, resulting in an even shorter latency time of only 4.17 ms. In addition to increasing performance where real-world access to data is concerned, spinning the platters faster also increases the data-transfer rate after the heads arrive at the desired sectors.

The transfer rate probably is more important to overall system performance than any other specification. *Transfer rate* is the rate at which the drive and controller can send data to the system. The transfer rate depends primarily on the drive's HDA and secondarily on the controller. Transfer rate used to be more bound to the limits of the controller, meaning that drives that were connected to older controllers often outperformed those controllers. This situation is where the concept of interleaving sectors came from. *Interleaving* refers to the ordering of the sectors so that they are not sequential, enabling a slow controller to keep up without missing the next sector.

A following section discusses interleaving in more detail. For now, the point I am trying to make is that modern drives with integrated controllers are fully capable of keeping up with the raw drive transfer rate. In other words, they no longer have to interleave the sectors to slow the data for the controller.

Another performance issue is the raw interface performance, which, in IDE or SCSI drives, usually is far higher than any of the drives themselves are able to sustain. Be wary of quoted transfer specifications for the interface, because they may have little effect on what the drive can actually put out. The drive interface simply limits the maximum theoretical transfer rate; the actual drive and controller place the real limits on performance.

In older ST-506/412 interface drives, you sometimes can double or triple the transfer rate by changing the controller, because many of the older controllers could not support a 1:1 interleave. When you change the controller to one that does support this interleave, the transfer rate will be equal to the drive's true capability.

To calculate the true transfer rate of a drive, you need to know several important specifications. The two most important specifications are the true rotational speed of the drive (in RPM) and the average number of physical sectors on each track. I say "average" because most drives today use a Zoned Recording technique that places different numbers of sectors on the inner and outer cylinders. The transfer rate on Zoned Recording drives always is fastest in the outermost zone, where the sector per track count is highest. Also be aware that many drives (especially Zoned Recording drives) are configured with sector translation, so that the BIOS reported number of sectors per track has little to do with physical reality. You need to know the true physical parameters, rather than what the BIOS thinks.

When you know these figures, you can use the following formula to determine the maximum transfer rate in millions of bits per second (Mbps):

> Maximum Data Transfer Rate (Mbps) = SPT × 512 bytes × RPM / 60 seconds / 1,000,000 bits

For example, the ST-12551N 2GB 3.5-inch drive spins at 7,200 RPM and has an average 81 sectors per track. The maximum transfer rate for this drive is figured as follows:

> 81 × 512 × 7,200 / 60 / 1,000,000 = 4.98Mbps

Using this formula, you can calculate the true maximum sustained transfer rate of any drive.

Cache Programs and Caching Controllers. At the software level, disk cache programs, such as SMARTDRV or PCKwik, can have a major effect on disk drive performance. These cache programs hook into the BIOS hard drive interrupt and then intercept the read and write calls to the disk BIOS from application programs and the device drivers of DOS.

When an application program wants to read data from a hard drive, the cache program intercepts the read request, passes the read request to the hard drive controller in the usual way, saves the data that was read in its cache buffer, and then passes the data back to the application program. Depending on the size of the cache buffer, numerous sectors are read into and saved in the buffer.

When the application wants to read more data, the cache program again intercepts the request and examines its buffers to see whether the data is still in the cache. If so, the data is passed back to the application immediately, without another hard drive operation. As you can imagine, this method speeds access tremendously and can greatly affect disk drive performance measurements.

Most controllers now have some form of built-in hardware buffer or cache that doesn't intercept or use any BIOS interrupts. Instead, the caching is performed at the hardware level and is invisible to normal performance-measurement software. Track read-ahead buffers originally were included in controllers to allow for 1:1 interleave performance. Some controllers have simply increased the sizes of these read-ahead buffers; others have added intelligence by making them a cache instead of a simple buffer.

Many IDE and SCSI drives have cache memory built directly into the drive. For example, the Toshiba MK-538FB 1.2GB drive on which I am saving this chapter has 512KB of built-in cache memory. Other drives have even more built-in caches, such as the Seagate Barracuda 2 2GB with 1MB of integral cache memory. I remember when 640KB was a lot of memory; now, tiny 3.5-inch hard disk drives have more than that built right in! These integral caches are part of the reason why most IDE and SCSI drives perform so well.

Although software and hardware caches can make a drive faster for routine transfer operations, a cache will not affect the true maximum transfer rate that the drive can sustain.

Interleave Selection. In a discussion of disk performance, the issue of interleave always comes up. Although traditionally, this was more a controller performance issue than a drive issue, most modern hard disks now have built-in controllers (IDE and SCSI) that are fully capable of taking the drive data as fast as the drive can send it. In other words, virtually all modern IDE and SCSI drives are formatted with no interleave (sometimes expressed as a 1:1 interleave ratio).

When a disk is low-level formatted, numbers are assigned to the sectors. These numbers are written in the Sector ID fields in the sector header and can be written or updated only by a low-level format. With older drive interfaces that used discrete controllers such as ST-506/412 or ESDI, you often had to calculate the best interleave value for the particular controller and system that you were using so you could low-level format the drive and number the sectors to offer optimum performance.

Notice that nearly all IDE and SCSI drives have their interleave ratios fixed at 1:1 and built-in controllers that can handle these ratios with no problem. In these drives, there no longer is a need to calculate or specify an interleave ratio, but knowing about interleaving can give you some further insight into the way that these drives function.

Many older ST-506/412 controllers could not handle the sectors as quickly as the drive could send them. Suppose that you have a standard ST-5096/412 drive with 17 sectors on each track and that you low-level formatted the drive, you specified a 1:1 interleave, which is to say that you numbered the sectors on each track consecutively.

Now suppose that you want to read some data from the drive. The controller commands the drive to position the heads at a specific track and to read all 17 sectors from that track. The heads move and arrive at the desired track. The time it takes for the heads to move is called the seek time. When the heads arrive at the desired track, you have to wait for an average half-revolution of the disk for the first sector to arrive. This wait is called latency. After an average latency of half of a disk revolution, the sector numbered 1 arrives below the heads. While the disk continues to spin at 3,600 rpm (60 revolutions per second), the data is read from sector 1, and as the data is being transferred from the controller to the system board, the disk continues to spin. Finally, the data is completely moved to the motherboard, and the controller is ready for sector 2. Figure 14.11 shows what is happening.

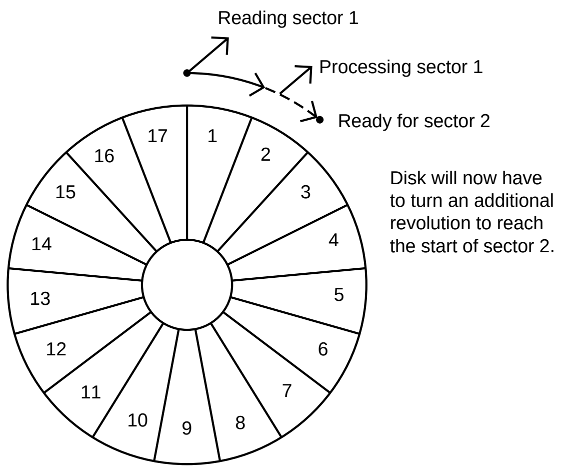

Reading sector 1
Processing sector 1
Ready for sector 2
Disk will now have to turn an additional revolution to reach the start of sector 2.
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

Fig. 14.11
A hard disk interleave ratio too low for the controller.

But wait—there's a problem here. Because the disk continues to spin at such a high rate of speed, the sector 2 passed below the head while the controller was working, and by the time that the controller is ready again, the heads will be coming to the start of sector 3. Because the controller needs to read sector 2 next, however, the controller must wait for the disk to spin a full revolution, or until the start of sector 2 comes below the heads. After this additional disk revolution, sector 2 arrives below the heads and is read. While the controller is transferring the data from sector 2 to the motherboard, sector 3 passes below the heads. When the controller finally is ready to read sector 3, the heads are coming to the start of sector 4, so another complete revolution is required, and the controller will have to get sector 3 on the next go-around. This scenario continues, with each new revolution allowing only one sector to be read and missing the next sector because that sector passes below the heads before the controller is ready.

As you can see, the timing of this procedure is not working out very well. At this pace, 17 full revolutions of the disk will be required to read all 17 sectors. Because each revolution takes 1/60 of 1 second, it will take 17/60 of 1 second to read this track, or almost one-third of a second—a very long time, by computer standards.

Can this performance be improved? You notice that after reading a specific sector from the disk, the controller takes some time to transfer the sector data to the motherboard. The next sector that the controller can catch in this example is the second sector away from the first one. In other words, the controller seems to be capable of catching every second sector.

I hope that you now can imagine the perfect solution to this problem: simply *number* the sectors out of order. The new numbering scheme takes into account how fast the controller works; the sectors are numbered so that each time the controller is ready for the next sector, the sector coming below the heads is numbered as the next sector that the controller will want to read. Figure 14.12 shows this new sector-numbering scheme.

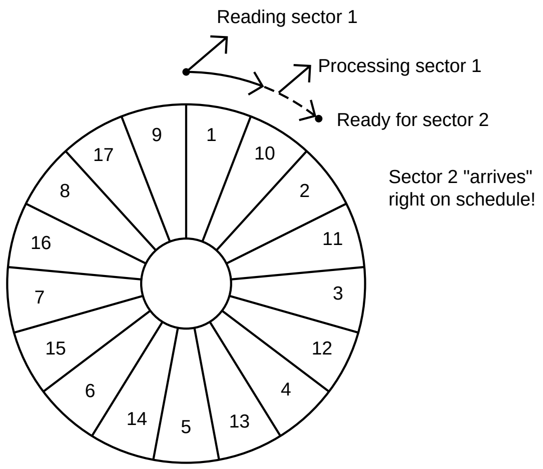


Fig. 14.12
A hard disk interleave ratio matching the controller's capabilities.

The new numbering system eliminates the extra disk revolution that previously was required to pick up each sector. Under the new scheme, the controller will read all 17 sectors on the disk in only two complete revolutions. Renumbering the sectors on the disk in this manner is called *interleaving*, which normally is expressed as a ratio. The interleave ratio in this example is 2 to 1 (also written as 2:1), which means that the next numbered sector is 2 sectors away from the preceding one; if the controller is capable of handling this, only two complete revolutions are needed to read an entire track. Thus, reading the track takes only 2/60 of one second at the 2:1 interleave, rather than the 17/60 of one second required to read the disk at the 1:1 interleave—an improvement of *800 percent* in the data transfer rate.

This example depicts a system in which the ideal interleave is 2:1. I used this example because most controllers that came in older AT systems worked in exactly this manner. If you set a controller for a 1:1 interleave, you likely would make the drive eight times slower than it should be. This situation, however, has changed with newer controller technology. Most disk controllers sold in the past couple of years support a 1:1 interleave on any AT-class system, even the slowest 6-MHz 286 versions. If you are purchasing or upgrading a system today, consider a disk subsystem with a 1:1 interleave controller to be a standard requirement. Fortunately, this is pretty much a non-issue; virtually all IDE or SCSI drives today have built-in controllers that easily handle a 1:1 interleave, and all these types of drives are preformatted at the factory in that manner. In many of these drives, it is not even possible to change the interleave to any value except 1:1.

The correct interleave for a system depends primarily on the controller and secondarily on the speed of the system into which the controller is plugged. A controller and system that can handle a *consecutive sector interleave* (a 1:1 interleave) must transfer data as fast as the disk drive can present it; this used to be quite a feat but now is commonplace.

Advances in controller technology have made a 1:1 interleave not only possible, but also affordable. Any 286 or faster system easily can handle the 1:1 interleave data transfer rate. The only types of systems that truly are too slow to handle a 1:1 interleave effectively are the original 4.77-MHz PC- and XT-type systems. Those systems have a maximum throughput to the slots of just under 400KB per second—not fast enough to support a 1:1 interleave controller.

Even with a 1:1 interleave, however, performance can vary significantly, and this is where the drive itself comes into play. A 1:1 interleave with a 17-sector disk is one thing, but with ESDI or SCSI drives spinning at 7,200 rpm and containing 81 or more sectors per track, the result is *nearly 5MB* of data transfer *each second.*

The interleave used in standard-issue IBM XT systems with hard drives was 6:1, whereas in IBM AT systems, it was 3:1. The best interleave for these systems actually is one lower than what was set up as standard in each case: In other words, the best interleave for the Xebec 1210 controller in a 4.77-MHz IBM PC or IBM XT is 5:1, and the best interleave for the Western Digital 1002 and 1003 controllers in a 6-MHz or 8-MHz IBM AT system is 2:1. If you redo the low-level format on these systems to the lower interleave number, you gain about 20 to 30 percent in data-transfer performance without changing any hardware and at no cost except some of your time.

Table 14.8 shows data transfer rates that were calculated for a variety of drives at a variety of different interleaves. The rows in this table represent a particular drive-and-controller combination with respect to the speed of revolution and the number of sectors per track. In all but the lowest-end setups, the interleave is 1:1, but I listed the other interleaves to show what the effect would be if they were used.

Table 14.8 Data-Transfer Rates (in K per Second) at Various Interleaves, Spindle Speeds, and Sector Densities

Speed	Sectors			Interleaves			
(RPM)	/Track	1:1	2:1	3:1	4:1	5:1	6:1
3,600	17	510	255	170	128	102	85
3,600	25	750	375	250	188	150	125
3,600	26	780	390	260	195	156	130
3,600	27	810	405	270	203	162	135
3,600	32	960	480	320	240	192	160
3,600	33	990	495	330	248	198	165
3,600	34	1,020	510	340	255	204	170
3,600	35	1,050	525	350	263	210	175
3,600	36	1,080	540	360	270	216	180
3,600	37	1,110	555	370	278	222	185
3,600	38	1,140	570	380	285	228	190
3,600	39	1,170	585	390	293	234	195
4,500	50	1,875	938	625	469	375	313
3,600	81	2,430	1,215	810	608	486	405
4,500	70	2,625	1,313	875	656	525	438
5,400	70	3,150	1,575	1,050	788	630	525
6,300	90	4,725	2,363	1,575	1,181	945	788
7,200	81	4,860	2,430	1,620	1,215	972	810

Compare the 85KB-per-second transfer rate of the original IBM XT drive and controller (3,600 rpm, 17 sectors, and a 6:1 interleave standard) with the 4,860KB-per-second transfer rate of the Seagate Barracuda 2 drive (7,200 rpm, 81 sectors, and a 1:1 interleave). As you can see, we have come a long way in 10 years of disk and controller technology!

If you want to find out the current interleave setting of your drive, I usually recommend using Norton Utilities for performing hard disk drive interleave testing. The Calibrate program included with Norton Utilities can check, and possibly even change, the interleave on ST-506/412 interface drives through a nondestructive low-level format. Notice that in this case, nondestructive actually means that the format is performed one track at a time, with the data for that track being backed up and restored all the while. A full backup of the entire drive beforehand still is recommended, because if something goes wrong, one or more tracks can be wiped out.

Calibrate (and other utility programs like it) can change the interleave only on ST-506/412 and possibly some ESDI drives, and there can be problems even on drives that they can handle. If you really want to change the interleave of a drive through a new low-level format, I usually recommend whatever low-level format program the controller manufacturer specifies for the best possible job. In most modern drives (ESDI, IDE, or SCSI), it usually is not possible (or desirable) to change the interleave with a generic low-level format program such as Calibrate. With an ESDI-type drive, for example, you normally would need to use the controller's built-in (or supplied on disk) low-level format (LLF) program to re-set the interleave (most ESDI controllers support 1:1, so there should be little reason to change). IDE and SCSI drives have the disk controller built in, and most have the interleave permanently set (nonchangeable) to the best possible choice, eliminating any reason to change. Virtually all modern drive/controller combinations have a default 1:1 interleave anyway, so changing would not be beneficial.

Head and Cylinder Skewing. Most controllers today are capable of transferring data at a 1:1 sector interleave. This is especially true of controllers that are built in to IDE and SCSI drives. With a 1:1 interleave controller, the maximum data transfer rate can be maintained when reading and writing sectors to the disk. Although it would seem that there is no other way to further improve efficiency and the transfer rate, many people overlook two important factors that are similar to interleave: head and cylinder skewing.

When a drive is reading (or writing) data sequentially, first all of the sectors on a given track are read; then the drive must electronically switch to the next head in the cylinder to continue the operation. If the sectors are not skewed from head to head within the cylinder, no delay occurs after the last sector on one track and before the arrival of the first sector on the next track. Because all drives require some time (although a small amount) to switch from one head to another, and because the controller also adds some overhead to the operation, it is likely that by the time the drive is ready to read the sectors on the newly selected track, the first sector will already have passed by. By skewing the sectors from one head to another—that is, rotating their arrangement on the track so that the arrival of the first sector is delayed relative to the preceding track—you can ensure that no extra disk revolutions will be required when switching heads. This method provides the highest possible transfer rate when head switching is involved.

In a similar fashion, it takes considerable time for the heads to move from one cylinder to another. If the sectors on one cylinder were not skewed from those on the preceding adjacent cylinder, it is likely that by the time the heads arrive, the first sector will already have passed below them, requiring an additional revolution of the disk before reading of the new cylinder can begin. By skewing the sectors from one cylinder to the next, you can account for the cylinder-to-cylinder head-movement time and prevent any additional revolutions of the drive.

Head Skew. *Head skew* is the offset in logical sector numbering between the same physical sectors on two tracks below adjacent heads of the same cylinder. The number of sectors skewed when switching from head to head within a single cylinder is to compensate for head switching and controller overhead time. Think of it as the surface of each platter being rotated as you traverse from head to head. This method permits continuous read or write operation across head boundaries without missing any disk revolutions, thus maximizing system performance.

To understand head skew, you first need to know the order in which tracks and sectors are read from a disk. If you imagine a single-platter (two-head) drive with 10 cylinders and 17 sectors per track, the first sector that will be read on the entire drive is Cylinder 0, Head 0, Sector 1. Following that, all the remaining sectors on that first track (Cylinder 0, Head 0) will be read until Sector 17 is reached. After that, two things could take place: one is that the heads could be moved so that the drive could continue reading the next track on the same side of the platter; or the second head could be selected, and therefore another entire track could be read with no head movement. Because head movement takes much longer than electronically selecting another head, all disk drives will select the subsequent heads on a cylinder before physically moving the heads to the next cylinder. Thus, the next sector to be read would be Cylinder 0, Head 1, Sector 1. Next, all the remaining sectors on that track are read (2 through 17), and then in our single platter example it is time to switch heads. This sequence continues until the last sector on the last track is read—in this example, Cylinder 9, Head 1, Sector 17.

If you could take the tracks off a cylinder in this example and lay them on top of one another, the tracks might look like this:

```
Cyl. 0, Head 0:    1- 2- 3- 4- 5- 6- 7- 8- 9-10-11-12-13-14-15-16-17
Cyl. 0, Head 1:    1- 2- 3- 4- 5- 6- 7- 8- 9-10-11-12-13-14-15-16-17
```

After reading all the sectors on head 0, the controller switches heads to head 1 and continues the read (looping around to the beginning of the track). In this example, the sectors were not skewed at all between the heads, which means that the sectors are directly above and below one another in a given cylinder.

Now the platters in this example are spinning at 3,600 rpm, so one sector is passing below a head once every 980 millionths of a second! This obviously is a very small timing window. It takes some time for the head switch to occur (usually, 15 millionths of a second), plus some overhead time for the controller to pass the head-switch command. By the time the head switch is complete and you are ready to read the new track, sector 1 has already gone by! This problem is similar to interleaving when the interleave is too low. The drive is forced to wait while the platter spins around another revolution so that it can begin to pick up the track, starting with Sector 1.

This problem is easy to solve: simply offset the sector numbering on subsequent tracks from those that precede them sufficiently to account for the head-switching and controller overhead time. That way, when Head 0, Sector 17 finishes and the head switches, Head 1, Sector 1 arrives right on time. The result looks something like this:

```
Cyl. 0, Head 0:     1- 2- 3- 4- 5- 6- 7- 8- 9-10-11-12-13-14-15-16-17
Cyl. 0, Head 1:    16-17- 1- 2- 3- 4- 5- 6- 7- 8- 9-10-11-12-13-14-15
```

Shifting the second track by two sectors provides time to allow for the head-switching overhead and is the equivalent to a head-skew factor of 2. In normal use, a drive switches heads much more often than it switches physical cylinders, which makes head skew more important than cylinder skew. Throughput can rise dramatically when a proper head skew is in place. Different head skews can account for different transfer rates among drives that have the same number of sectors per track and the same interleave.

A nonskewed MFM drive, for example, may have a transfer rate of 380KB per second, whereas the transfer rate of a drive with a head skew of 2 could rise to 425KB per second.

Notice that different controllers and drives have different amounts of overhead, so real-world results will be different in each case. In most cases, the head-switch time is very small compared with the controller overhead. As with interleaving, it is better to be on the conservative side to avoid additional disk revolutions.

Cylinder Skew. *Cylinder skew* is the offset in logical sector numbering between the same physical sectors on two adjacent tracks on two adjacent cylinders.

The number of sectors skewed when switching tracks from one cylinder to the next is to compensate for track-to-track seek time. In essence all of the sectors on adjacent tracks are rotated with respect to each other. This method permits continuous read or write operations across cylinder boundaries without missing any disk revolutions, thus maximizing system performance.

Cylinder skew is a larger numerical factor then head skew because more overhead exists. It takes much longer to move the heads from one cylinder to another than simply to switch heads. Also, the controller overhead in changing cylinders is higher as well.

The following is a depiction of our example drive with a head-skew factor of 2 but no cylinder skew.

```
Cyl. 0, Head 0:     1- 2- 3- 4- 5- 6- 7- 8- 9-10-11-12-13-14-15-16-17
Cyl. 0, Head 1:    16-17- 1- 2- 3- 4- 5- 6- 7- 8- 9-10-11-12-13-14-15
Cyl. 1, Head 0:     8- 9-10-11-12-13-14-15-16-17- 1- 2- 3- 4- 5- 6- 7
```

In this example, the cylinder-skew factor is 8. Shifting the sectors on the subsequent cylinder by eight sectors gives the drive and controller time to be ready for sector 1 on the next cylinder and eliminates an extra revolution of the disk.

Calculating Skew Factors. You can derive the correct head-skew factor from the following information and formula:

Head skew = (head-switch time/rotational period) × SPT + 2

In other words, the head-switching time of a drive is divided by the time required for a single rotation. The result is multiplied by the number of sectors per track, and 2 is added for controller overhead. The result should then be rounded up to the next whole integer (for example, 2.3 = 2, 2.5 = 3).

You can derive the correct cylinder-skew factor from the following information and formula:

Cylinder skew = (track-to-track seek time/rotational period) × SPT + 4

In other words, the track-to-track seek time of a drive is divided by the time required for a single rotation. The result is multiplied by the number of sectors per track, and 4 is added for controller overhead. Round the result up to a whole integer (for example, 2.3 = 2, 2.5 = 3).

The following example uses typical figures for an ESDI drive and controller. If the head-switching time is 15 µs (micro-seconds), the track-to-track seek is 3 ms, the rotational period is 16.67 ms (3,600 rpm), and the drive has 53 physical sectors per track:

Head skew = (0.015/16.67) × 53 +2 = 2 (rounded up)

Cylinder Skew = (3/16.67) × 53 + 4 = 14 (rounded up)

If you do not have the necessary information to make the calculations, contact the drive manufacturer for recommendations. Otherwise, you can make the calculations by using conservative figures for head-switch and track-to-track access times. If you are unsure, just as with interleaving, it is better to be on the conservative side, which minimizes the possibility of additional rotations when reading sequential information on the drive. In most cases, a default head skew of 2 and a cylinder skew of 16 work well.

Because factors such as controller overhead can vary from model to model, sometimes the only way to figure out the best value is to experiment. You can try different skew values and then run data-transfer rate tests to see which value results in the highest performance. Be careful with these tests, however; many disk benchmark programs will only read or write data from one track or one cylinder during testing, which totally eliminates the effect of skewing on the results. The best type of benchmark to use for this testing is one that reads and writes large files on the disk.

Most real (controller register level) low-level format programs are capable of setting skew factors. Those programs that are supplied by a particular controller or drive manufacturer usually already are optimized for their particular drives and controllers, and may not allow you to change the skew. One of the best general-purpose register-level formatters on the market that gives you this flexibility is the Disk Manager program by Ontrack. I highly recommend this program.

I normally do not recommend programs such as Norton Calibrate and Gibson Spinrite for re-interleaving drives, because these programs work only through the BIOS INT 13h functions rather than directly with the disk controller hardware. Thus, these programs cannot set skew factors properly, and using them actually may slow a drive that already has optimum interleave and skew factors.

Notice that most IDE and SCSI drives have their interleave and skew factors set to their optimum values by the manufacturer. In most cases, you cannot even change these values; in the cases in which you can, the most likely result is a slower drive. For this reason, most IDE drive manufacturers recommend against low-level formatting their drives. With some IDE drives, unless you use the right software, you might alter the optimum skew settings and slow the drive. IDE drives that use Zoned Recording cannot ever have the interleave or skew factors changed, and as such, they are fully protected. No matter how you try to format these drives, the interleave and skew factors cannot be altered. The same can be said for SCSI drives.

Shock Mounting

Most hard disks manufactured today have a *shock-mounted* HDA, which means that a rubber cushion is placed between the disk drive body and the mounting chassis. Some drives use more rubber than others, but for the most part, a shock mount is a shock mount. Some drives do not have a shock-isolated HDA due to physical or cost constraints. Be sure that the drive you are using has adequate shock-isolation mounts for the HDA, especially if you are using the drive in a portable PC system or in a system in which environmental conditions are less favorable than in a normal office. I usually never recommend a drive that lacks at least some form of shock mounting.

Cost

The cost of hard disk storage recently has fallen to the magic $1-per-megabyte barrier (or lower). You can purchase 2GB drives for $2,000, 1GB drives for $1,000, 500MB drives for $500, and so on. That places the value of the 10 MB drive that I bought in 1983 at about $10. (Too bad I paid $1,800 for it at the time!)

Of course, the cost of drives continues to fall, and eventually, even $1 per megabyte will seem expensive. Because of the low costs of disk storage today, not many drives that are less than 200MB are even being manufactured.

Capacity

Four figures commonly are used in advertising drive capacity:

- Unformatted capacity, in millions of bytes (MB)
- Formatted capacity, in millions of bytes (MB)
- Unformatted capacity, in megabytes (Meg or MB)
- Formatted capacity, in megabytes (Meg or MB)

Most manufacturers of IDE and SCSI drives now report only the formatted capacities, because these drives are delivered preformatted. Most of the time, advertisements refer to the unformatted or formatted capacity in millions of bytes (MB), because these figures are larger than the same capacity expressed in megabytes (Meg). This situation generates a great deal of confusion when the user runs FDISK (which reports total drive capacity in megabytes) and wonders where the missing space is. This question ranks as one of the most common questions that I hear during my seminars. Fortunately, the answer is easy; it only involves a little math to figure it out.

Perhaps the most common questions I get are concerning "missing" drive capacity. Consider the following example: "I just installed a new Western Digital AC2200 drive, billed as 212MB. When I entered the drive parameters (989 cylinders, 12 heads, 35 sectors per track), both the BIOS Setup routine and FDISK report the drive as only 203MB! What happened to the other 9MB?"

The answer is only a few calculations away. By multiplying the drive specification parameters, you get this result:

Cylinders:	989
Heads:	12
Sectors per track:	35
Bytes per sector:	512

Total bytes:	212.67MB
Total megabytes:	202.82Meg

The result figures to a capacity of 212.67MB (million bytes) or 202.82Meg. Drive manufacturers usually report drive capacity in millions of bytes, whereas your BIOS and FDISK usually report the capacity in megabytes. 1Meg equals 1,048,576 bytes (or 1,024KB, wherein each KB is 1,024 bytes). So the bottom line is that this 212.67MB drive also is a 202.82Meg drive! What is really confusing is that there is no industry wide accepted way of differentiating binary megabytes from decimal ones. Officially they are both abbreviated as "MB", so it is often hard to figure which one is being reported. Usually drive manufacturers will always report metric MBs, since they result in larger, more impressive sounding numbers! One additional item to note about this particular drive is that it is a Zoned Recording drive and that the actual physical parameters are different. Physically, this drive has 1,971 cylinders and 4 heads; however, the total number of sectors on the drive (and, therefore, the capacity) is the same no matter how you translate the parameters.

Although Western Digital does not report the unformatted capacity of this particular drive, unformatted capacity usually works out to be about 19 percent larger than a drive's formatted capacity. The Seagate ST-12550N drive, for example, is advertised as having the following capacities:

Unformatted capacity:	2,572.00MB
Unformatted capacity:	2,452.85Meg
Formatted capacity:	2,139.00MB
Formatted capacity:	2,039.91Meg

Each of these four figures is a correct answer to the question "What is the storage capacity of the drive?" As you can see, however, the numbers are very different. In fact, yet another number could be used. Divide the 2,039.91Meg by 1,024, and the drive's capacity is 1.99GB! So when you are comparing or discussing drive capacities, make sure that you are working with a consistent unit of measure, or your comparisons will be meaningless.

To eliminate confusion in capacity measurements. I have been using the abbreviation "Meg" in this section, which is not really industry standard. The true industry standard abbreviations for these figures are shown in table 14.9:

Table 14.9 Standard Abbreviations and Meanings

Abbreviation	Description	Decimal Meaning	Binary Meaning
Kb	Kilobit	1,000	1,024
KB	Kilobyte	1,000	1,024
Mb	Megabit	1,000,000	1,048,576
MB	Megabyte	1,000,000	1,048,576

(continues)

Table 14.9 Continued

Abbreviation	Description	Decimal Meaning	Binary Meaning
Gb	Gigabit	1,000,000,000	1,073,741,824
GB	Gigabyte	1,000,000,000	1,073,741,824
Tb	Terabit	1,000,000,000,000	1,099,511,627,776
TB	Terabyte	1,000,000,000,000	1,099,511,627,776

Unfortunately there are no differences in the abbreviations when used to indicate metric verses binary values. In other words MB can be used to indicate both Millions of Bytes and Megabytes. In general, memory values are always computed using the binary derived meanings, while disk capacity goes either way. Unfortunately this often leads to confusion in reporting disk capacities. Note that bits and Bytes are distinguished by a lower- or uppercase "B" character. For example, Millions of bits are indicated by using a lowercase *b*, resulting in the abbreviation of Mbps for Million bits per second, while MBps indicates Million Bytes per second.

Specific Recommendations

If you are going to add a hard disk to a system today, I can give you a few recommendations. Starting with the actual, physical hard disk itself, you should demand the following features in a hard disk drive to ensure that you are getting a quality unit:

- Voice coil head actuator
- Thin-film media

For the drive interface, there really are only two types to consider today:

- IDE (Integrated Drive Electronics)
- SCSI (Small Computer System Interface)

Of these, I prefer SCSI because of its great expandability, cross-platform compatibility, high capacity, performance, and flexibility. IDE offers a very high-performance solution, but expansion, compatibility, capacity, and flexibility are severely limited compared with SCSI.

Hard Disk Interfaces

A variety of hard disk interfaces are available today. As time has passed, the number of choices has increased, and many of the older designs no longer are viable in newer systems. You need to know about all these interfaces, from the oldest to the newest designs, because you will encounter all of them whenever upgrading or repairing systems is necessary.

The interfaces have different cabling and configuration options, and the setup and format of drives will vary as well. Special problems may arise when you are trying to install more than one drive of a particular interface type or (especially) when you are mixing drives of different interface types in one system.

This section completely covers the different hard disk drive interfaces, giving you all the technical information you need to deal with them in any way: troubleshooting, servicing, upgrading, and even mixing the different types.

This section examines the standard controllers and describes how you can work with these controllers, as well as replace them with much faster units. Also discussed are the different types of drive interfaces: ST-506/412, ESDI, IDE, and SCSI. Choosing the proper interface is important, because your choice also affects your disk drive purchase and the ultimate speed of the disk subsystem.

The primary job of the hard disk controller or interface is to transmit and receive data to and from the drive. The different interface types limit how fast data can be moved from the drive to the system and offer different levels of performance. If you are putting together a system in which performance is a primary concern, you need to know how these different interfaces affect performance and what you can expect from them. Many of the statistics that appear in technical literature are not indicative of the real performance figures that you will see in practice. I will separate the myths presented by some of these overoptimistic figures from the reality of what you will actually see.

With regard to disk drives, and especially hard disk drives, the specification on which people seem to focus the most is the drive's reported average seek time: the (average) time it takes for the heads to be positioned from one track to another. Unfortunately, the importance of this specification often is overstated, especially in relation to other specifications, such as the data-transfer rate.

The transfer rate of data between the drive and the system is more important than access time, because most drives spend more time reading and writing information than they do simply moving the heads around. The speed at which a program or data file is loaded or read is affected most by the data-transfer rate. Specialized operations such as sorting large files, which involve a lot of random access to individual records of the file (and, therefore, many seek operations), are helped greatly by a faster-seeking disk drive, so seeking performance is important in these cases. Most normal file load and save operations, however, are affected most by the rate at which data can be read and written to and from the drive. The data-transfer rate depends on both the drive and the interface.

Several types of hard disk interfaces have been used in PC systems over the years:

- ST-506/412
- ESDI
- IDE
- SCSI

Of these interfaces, only ST-506/412 and ESDI are what you could call true disk-controller-to-drive interfaces. SCSI and IDE are system-level interfaces that usually incorporate a chipset-based variation of one of the other two types of disk controller interfaces internally. For example, most SCSI and IDE drives incorporate the same basic controller circuitry used in separate ESDI controllers. The SCSI interface adds another layer of interface that attaches the controller to the system bus, whereas IDE is a direct bus-attachment interface.

In data recovery, it helps to know the disk interface you are working with, because many data-recovery problems involve drive setup and installation problems. Each interface requires a slightly different method of installation and drive configuration. If the installation or configuration is incorrect or accidentally altered by the system user, it may prevent access to data on a drive. Accordingly, anyone who wants to become proficient in data recovery must be an expert on installing and configuring various types of hard disks and controllers.

IBM's reliance on industry-standard interfaces such as those listed here was a boon for everybody in the IBM-compatible industry. These standards allow a great deal of cross-system and cross-manufacturer compatibility. The use of these industry-standard interfaces allows us to pick up a mail-order catalog, purchase a hard disk for the lowest possible price, and be assured that it will work with our system. This *plug-and-play* capability results in affordable hard disk storage and a variety of options in capacities and speed.

The ST-506/412 Interface

The ST-506/412 interface was developed by Seagate Technologies around 1980. The interface originally appeared in the Seagate ST-506 drive, which was a 5MB formatted (or 6MB unformatted) drive in a full-height, 5.25-inch form factor. By today's standards, this interface is a tank! In 1981 Seagate introduced the ST-412 drive, which added a feature called *buffered seek* to the interface. This drive was a 10MB formatted (12MB unformatted) drive that also qualifies as a tank by today's standards. Besides the Seagate ST-412, IBM also used the Miniscribe 1012 as well the International Memories, Inc. (IMI) model 5012 drive in the XT. IMI and Miniscribe are long gone, but Seagate remains today as one of the largest drive manufacturers. Since the original XT, Seagate has supplied drives for numerous IBM systems, including the AT and many PS/2 models.

Most drive manufacturers that made hard disks for PC systems adopted the Seagate ST-506/412 standard, a situation that helped make this interface popular. One important feature is the interface's plug-and-play design. No custom cables or special modifications are needed for the drives, which means that virtually any ST-506/412 drive will work with any ST-506/412 controller. The only real compatibility issue with this interface is the level of BIOS support provided by the system.

When introduced to the PC industry by IBM in 1983, ROM BIOS support for this hard disk interface was provided by a BIOS chip on the controller. Contrary to what most believed, the PC and XT motherboard BIOS had no inherent hard disk support. When the AT system was introduced, IBM placed the ST-506/412 interface support in the motherboard BIOS and eliminated it from the controller. Since then, any system that is compatible with the IBM AT (which includes most systems on the market today) has an enhanced version of the same support in the motherboard BIOS as well. Because this support was somewhat limited, especially in the older BIOS versions, many disk controller manufacturers also placed additional BIOS support for their controllers directly on the controllers themselves. In some cases, you would use the controller BIOS and motherboard BIOS together; in other cases, you would disable the controller or motherboard BIOS and then use one or the other. These issues will be discussed more completely later in this chapter, in the section called System Configuration.

The ST-506/412 interface does not quite make the grade in today's high-performance PC systems. This interface was designed for a 5MB drive, and I have not seen any drives larger than 152MB (MFM encoding) or 233MB (RLL encoding) available for this type of interface. Because the capacity, performance, and expandability of ST-506/412 are so limited, this interface is obsolete and generally unavailable in new systems. However, many older systems still use drives that have this interface.

Encoding Schemes and Problems. As indicated earlier in this chapter in the section on Data Encoding Schemes, encoding schemes are used in communications for converting digital data bits to various tones for transmission over a telephone line. For disk drives, the digital bits are converted, or *encoded*, in a pattern of magnetic impulses, or *flux transitions* (also called flux reversals), which are written on the disk. These flux transitions are decoded later, when the data is read from the disk.

A device called an Endec (encoder/decoder) accomplishes the conversion to flux transitions for writing on the media and the subsequent reconversion back to digital data during read operations. The function of the Endec is very similar to that of a modem (modulator/demodulator) in that digital data is converted to an analog waveform, which then is converted back to digital data. Sometimes, the Endec is called a *data separator*, because it is designed to separate data and clocking information from the flux-transition pulse stream read from the disk.

One of the biggest problems with ST-506/412 was that this Endec resided on the disk controller (rather than the drive), which resulted in the possibility of corruption of the analog data signal before it reached the media. This problem became especially pronounced when the ST-506/412 controllers switched to using RLL Endecs to store 50 percent more data on the drive. With the RLL encoding scheme, the actual density of magnetic flux transitions on the disk media remains the same as with MFM encoding, but the timing between the transitions must be measured much more precisely.

The original ST-506/412 controllers used MFM encoding, for which the interface was designed. A few years after this interface had been out, several controller manufacturers started using RLL Endec circuits in their controllers. These newer controllers were fully compatible with the ST-506/412 interface drives, but they could put 50 percent more data on the drives, and transfer data to and from the drives 50 percent faster than the older MFM-variety controllers. Unfortunately, this situation caused many problems with the cheaper drives that were on the market at the time.

In RLL encoding, the intervals between flux changes are approximately the same as with MFM, but the actual timing between them is much more critical. As a result, the transition cells in which signals must be recognized are much smaller and more precisely placed than with MFM. RLL encoding places more stringent demands on the timing of the controller and drive electronics. With RLL encoding, accurately reading the timing of the flux changes is paramount. Additionally, because RLL encodes variable-length groups of bits rather than single bits, a single error in one flux transition can corrupt two to four bits of data. For these reasons, an RLL controller usually has a more sophisticated error-detection and error-correction routine than an MFM controller.

Most of the cheaper disk drives on the market did not have data-channel circuits that were designed to be precise enough to handle RLL encoding without problems. RLL encoding also is much more susceptible to noise in the read signal, and the conventional oxide media coatings did not have a sufficient signal-to-noise ratio for reliable RLL encoding. This problem often was compounded by the fact that many drives of the time used stepper motor head positioning systems, which are notoriously inaccurate, further amplifying the signal-to-noise ratio problem.

At this time, manufacturers starting RLL-certifying drives for use with RLL Endec controllers. This stamp of approval essentially meant that the drive had passed tests and was designed to handle the precise timing requirements that RLL encoding required. In some cases, the drive electronics were upgraded between a manufacturer's MFM and RLL drive versions, but the drives are essentially the same. In fact, if any improvements were made in the so-called RLL-certified drives, the same upgrades usually also were applied to the MFM version.

The bottom line is that other than improved precision, there is no real difference between an ST-506/412 drive that is sold as an MFM model and one that is sold as an RLL model. If you want to use a drive that originally was sold as an MFM model with an RLL controller, I suggest that you do so only if the drive uses a voice coil head actuator and thin-film media. Virtually any ST-506/412 drive with these qualities is more than good enough to handle RLL encoding with no problems.

Using MFM encoding, a standard ST-506/412 format specifies that the drive will contain 17 sectors per track, with each sector containing 512 bytes of data. A controller that uses an RLL Endec raises the number of sectors per track to 25 or 26.

The real solution to reliability problems with RLL encoding was to place the Endec directly on the drive rather than on the controller. This method reduces the susceptibility to noise and interference that can plague an ST-506/412 drive system running RLL encoding. ESDI, IDE, and SCSI drives all have the Endec (and, often, the entire controller) built into the drive by default. Because the Endec is attached to the drive without cables and with an extremely short electrical distance, the propensity for timing-and noise-induced errors is greatly reduced or eliminated. This situation is analogous to a local telephone call between the Endec and the disk platters. This local communication makes the ESDI, IDE, and SCSI interfaces much more reliable than the older ST-506/412 interface; they share none of the reliability problems that once were associated with RLL encoding over the ST-506/412 interface. Virtually all ESDI, IDE, and SCSI drives use RLL encoding today with tremendously increased reliability over even MFM ST-506/412 drives.

ST-506/412 Configuration and Installation. The ST-506/412 interface is characterized by a two- or three-cable arrangement, depending on whether one or two drives are connected. One 34-connector control cable is daisy-chained between up to two drives. The daisy-chain arrangement is much like that used for floppy drives. Each drive on the daisy chain is jumped to respond to a particular Drive Select (DS) line. In the controller implementation used in all PC systems, there are two available lines, called Drive Select 1 (DS1) and Drive Select 2 (DS2). Some drives support as many as four DS lines, but only the first two are usable. Although it may appear that you could string four drives on a single daisy-chain cable, the design of the PC system and controllers uses only the first two.

The control cable usually has lines 25 through 29 twisted between the drive D and C connectors. The first drive (drive C) normally is plugged into the last control cable connector at the end of the cable opposite the controller; an optional second drive (D) can be installed in the middle control-cable connector. The twist in the lines serves to reroute the Drive Select lines so that the drive plugged into the last cable position appears to the controller to be attached to Drive Select 1, even though the jumper on the drive is set for DS2. This arrangement is very similar to the one used for floppy drives; if the cable is twisted, both drives must be set to the DS2 jumper position. If the cable does not have the twisted lines, the drive at the end of the cable (C) must be set to DS1.

Another configuration item is the terminating resistor, which must be installed on the drive at the end of the cable (C) and must be removed from the optional second drive (D) attached to the middle control-cable connector. The controller has a permanently installed terminating resistor that never has to be adjusted. Although the control cable is similar in function and appearance to the 34-pin cable used for floppy drives, the cables generally are not interchangeable because different lines are twisted. Whereas pins 25 through 29 are inverted on the hard disk control cable, pins 10 through 16 are inverted on the floppy cable, rendering them incompatible.

The other two cables, called *data cables*, are 20-connector cables, each of which runs from the controller to a single drive, because this cable is not daisy-chained. A two-drive system therefore has one control cable from the controller to each of two drives in a daisy chain, plus two separate data cables—one for each drive. The controller has three connectors to support the two-drive maximum limit. As its name suggests, the data cable carries data to and from the drive.

If you are using a single drive, only the data cable connector closest to the control cable connector is used; the other is left unattached. Most ST-506/412 controllers also have an on-board floppy controller, which also will have a 34-pin connector for the floppy drives. Figures 14.13 and 14.14 show the control and data cable connectors on a typical combination ST-506/412 hard disk and floppy disk controller. Notice that some of these combination controllers allow the floppy controller portion to be disabled and others do not, which may cause a conflict if you have any other floppy controller in the system.

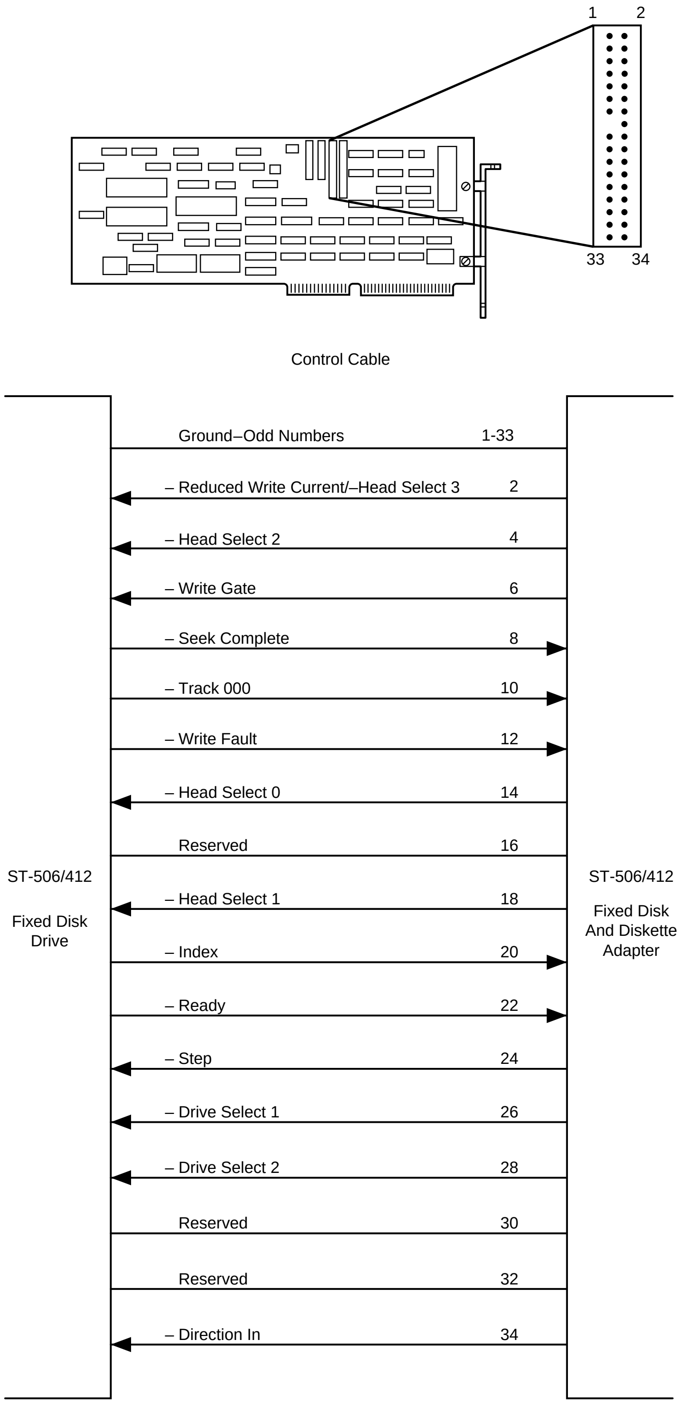


Fig. 14.13
ST-506/412 controller control cable connector.

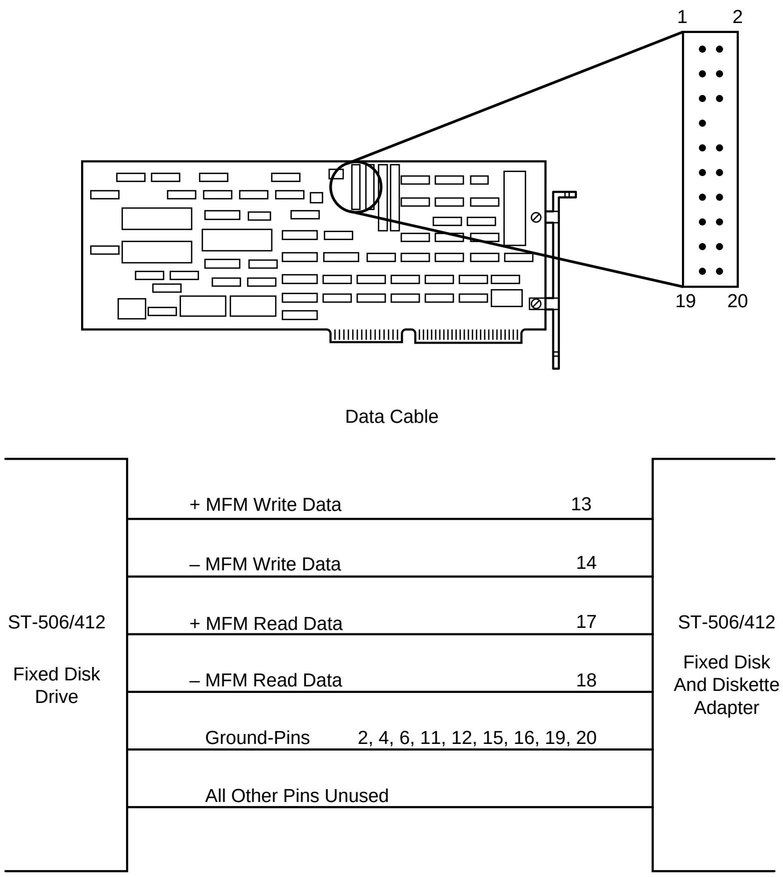

1
2
19
20
Data Cable
+ MFM Write Data 13
– MFM Write Data 14
+ MFM Read Data 17
– MFM Read Data 18
Ground-Pins 2, 4, 6, 11, 12, 15, 16, 19, 20
All Other Pins Unused
ST-506/412 Fixed Disk Drive
ST-506/412 Fixed Disk And Diskette Adapter

Fig. 14.14
ST-506/412 controller data cable connectors.

ST-506/412 Drive Configuration. With ST-506/412 and ESDI drives, you have to configure the following items on each drive:

- Drive Select (DS) jumpers
- Terminating resistor

These configuration items usually are located near the rear of the drive on the disk drive logic board.

Drive Select Jumpers. The *Drive Select jumper* selects the Drive Select (DS) signal to which the drive should respond. The drive controller sends control signals on two DS lines, one for each drive. Because each drive must be set to respond to a different DS signal, you can use only two drives per controller.

The DS jumpers must be set so that each drive responds to a different DS line from the controller (DS1 or DS2). If the 34-pin control cable has a twist in lines 25 through 29, both drives should be set to DS2. If the control cable is a straight-through design (no twist), the drive at the end of the cable opposite the controller (C) should be set to DS1, and a second drive attached to the middle control-cable connector (D) should be set to DS2. Notice that some drives label the DS jumpers starting with 0, so that DS1 would be labeled DS0 and DS2 would be labeled DS1.

Terminating Resistors. An ST-506/412 drive always is shipped from the factory with a terminating resistor installed. When you install these drives, you must ensure that the drive plugged into the end of the control-cable daisy chain has this terminator installed. Additionally, this terminator must be removed (or disabled with a jumper, in some cases) from the secondary drive installed in the center control-cable connector.

The functions of the terminating resistor are the same as those discussed for floppy drives. The idea is to provide electrical-signal termination so that the control signals to and from the drive and controller do not reflect back or echo along the cable. The terminating resistor provides the proper signal-to-noise ratio and the proper electrical load for the controller. Improper drive termination results in drives that do not function (or that do so only with excessive problems). Improper termination also may damage the controller because of improper electrical loads.

Control and Data Cables. The control cable connects to the controller and daisy chains to the secondary and primary drives, and separate data cables (20-pin) run from the controller to each drive. The data cable connector closest to the control-cable connector on the controller is used for the primary (C) drive.

When connecting cables, you should observe the proper pin-1 orientation from end to end. On ribbon cables, the pin-1 line usually is a different color from the other lines. (In most ribbon cables, for example, the pin-1 line is red or blue, and the rest of the cable is gray). You need to ensure that this pin-1 line is plugged into pin-1 of the controller and drive connectors. Both the controller and drive should have the pin-1 position on each connector marked. Sometimes the mark is the number 1; other times, it is a dot or some other symbol that is silkscreened on the circuit board. The cable connectors at the controller end may be keyed, in which case pin-15 will be missing from the control-cable connector on the controller connector, and the corresponding hole will be plugged in the control cable. The data cable connectors will be missing pin-8, and the corresponding hole also will be plugged in the data cables. The edge connectors used at the drive end normally are keyed to a notch in the drive connectors. The side of the connector with the notch cut out indicates the pin-1 orientation at the drive end.

Notice that the 34-pin control cable is very similar to the 34-pin control/data cable used for floppy drives; these cables, however, usually are not interchangeable. The ST-506/412 control cable has lines 25 through 29 twisted between the secondary and primary drive connectors, whereas the 34-pin floppy cable has lines 10 through 17 twisted. As a result, the cables are incompatible and therefore noninterchangeable.

ST-506/412 Interface Connectors. The ST-506/412 Interface uses two connections, a 34-pin Control connector and a 20-pin Data connector. Tables 14.10 and 14.11 show the pinouts for these connectors.

Table 14.10 ST-506/412 Hard Disk Interface 34-Pin Control Connector Pinout

Signal Name	Pin	Pin	Signal Name
GROUND	1	2	-HD SLCT 3
GROUND	3	4	-HD SLCT 2
GROUND	5	6	-WRITE GATE
GROUND	7	8	-SEEK CMPLT
GROUND	9	10	-TRACK 0
GROUND	11	12	-WRITE FAULT
GROUND	13	14	-HD SLCT 0
KEY (no pin)	15	16	Not Connected
GROUND	17	18	-HD SLCT 1
GROUND	19	20	-INDEX
GROUND	21	22	-READY
GROUND	23	24	-STEP
GROUND	25	26	-DRV SLCT 0
GROUND	27	28	-DRV SLCT 1
GROUND	29	30	Not Connected
GROUND	31	32	Not Connected
GROUND	33	34	-DIRECTION IN

Table 14.11 ST-506/412 Hard Disk Interface 20-Pin Data Connector Pinout

Signal Name	Pin	Pin	Signal Name
-DRV SLCTD	1	2	GROUND
Not Connected	3	4	GROUND
Not Connected	5	6	GROUND
Not Connected	7	8	KEY (no pin)
Not Connected	9	10	Not Connected
GROUND	11	12	GROUND
+MFM WRITE	13	14	-MFM WRITE
GROUND	15	16	GROUND
+MFM READ	17	18	-MFM READ
GROUND	19	20	GROUND

Power Cables. To complete the required cable connections to the hard drive, you need a spare power connector (table 14.12 shows the pinouts for the connector). Some older power supplies have only two-drive power connectors. Several companies sell a *power splitter cable*, or Y cable, that can adapt one cable from the power supply so that it powers two drives. If you add a power splitter to a system, make sure that the power supply can handle the load of the additional drive or drives.

Table 14.12 Disk Drive Power Connector Pinout

Pin	Wire Color	Signal
1	Yellow	+12v
2	Black	Gnd
3	Black	Gnd
4	Red	+5v

If the original power supply is not adequate, purchase an aftermarket unit that can supply adequate power. Most better aftermarket supplies have four-drive power connectors, eliminating the need for the splitter cables. Power splitter cables are available from several of the cable and accessory vendors listed in Appendix B, as well as from electronic supply stores such as Radio Shack.

Historical Notes. The following sections list some information on the original ST-506/412 controllers used in the PC environment. These were the controllers that IBM supplied in the XT and AT systems. At the time of introduction, these controllers set standards that, especially in the case of the AT controller, we still live with today. In fact, the entire IDE interface standard is based on the controller that IBM designed and used in the AT. All the conventions and standards for the hard disk interfaces that we use today started with these controllers.

Original IBM 8-Bit Controllers. The first ST-506/412 controller standard sold for PC systems was the hard disk controller used in the original 10MB IBM XT. This controller actually was made for IBM by Xebec Corporation and also was sold under the Xebec name as the Xebec 1210 controller. The Xebec 1210 is an ST-506/412 controller that uses Modified Frequency Modulation (MFM) encoding to record data on a drive. This controller's ROM, produced by IBM, contains an 8KB hard disk BIOS with an internal table that had entries for four different drives. Each drive was selected by jumpers on the controller, which actually were soldered in the early IBM units. If you purchased the controller from Xebec, you got a slightly different but completely compatible ROM, and the jumpers were not soldered, so you easily could select one of the four BIOS table entries. Xebec also allowed system integrators to copy its ROM to modify the built-in drive tables for a specific drive.

Later IBM XT systems with a 20MB hard disk still used the Xebec 1210, but it had a new 4KB ROM that contained different drive tables, as well as jumpers like those found on the versions also sold separately by Xebec. Xebec never sold an autoconfigure version of this controller, which would have made integrating different drives easier.

The Xebec 1210 is one of the slowest ST-506/412 controllers ever made, supporting at best a 5:1 interleave on a stock IBM PC or IBM XT system. If you use the IBM Advanced

Diagnostics program for the IBM PC or IBM XT, the low-level formatter produces a standard 6:1 interleave, which results in a paltry 85KB-per-second data-transfer rate. By changing the interleave to 5:1, you can wring 102KB per second from this controller—still unbelievably slow by today's standards.

Xebec also made a Model 1220 that combined a hard disk and floppy disk controller, was hardware-compatible with the 1210, and works with the IBM or standard Xebec ROM. The separate floppy controller then could be removed from the system, and you could save a slot.

I recommend replacing this controller with an autoconfigure controller whenever you get the chance. Most other controllers also are significantly faster than the Xebec.

Original IBM 16-Bit Controllers. For the AT, IBM used two controllers made by Western Digital: the WD1002-WA2 and the WD1003A-WA2. The WD1003 is an upgraded WD1002 with a much lower chip count. The WD1003 also was shorter than the WD1002 to fit into the IBM XT 286.

The WD1002 is used in the IBM AT as a combination hard disk and floppy disk controller. The WD1002 and the WD1003 are standard ST-506/412 controllers that supply MFM encoding to the drive. Neither controller contains a ROM BIOS; instead, BIOS support is built into the motherboard ROM. Both controllers support a 2:1 interleave, even on a standard 6-MHz IBM AT system. The IBM Advanced Diagnostics low-level formatter can put down a 2:1 interleave, but the default is 3:1. Most users of these controllers can realize a performance gain if they simply reformat to the lower interleave.

The ESDI Interface

ESDI, or *Enhanced Small Device Interface*, is a specialized hard disk interface established as a standard in 1983, primarily by Maxtor Corporation. Maxtor led a consortium of drive manufacturers to adopt its proposed interface as a high-performance standard to succeed ST-506/412. ESDI later was adopted by the ANSI (American National Standards Institute) organization and published under the ANSI X3T9.2 Committee. The latest version of the ANSI ESDI document is known as X3.170a-1991. You can obtain this document, and other ANSI-standard documents, from ANSI itself or from Global Engineering Documents. These companies are listed in Appendix B.

Compared with ST-506/412, ESDI has provisions for increased reliability, such as building the Endec (encoder/decoder) into the drive. ESDI is a very-high-speed interface, capable of a maximum 24-megabits-per-second transfer rate. Most drives running ESDI, however, are limited to a maximum 10 or 15 megabits per second. Unfortunately, compatibility problems between different ESDI implementations, combined with pressure from low-cost, high-performance IDE interface drives have served to make the ESDI interface obsolete. Few, if any, new systems today include ESDI drives, although ESDI has become somewhat popular in high-end systems during the late '80s.

Enhanced commands enabled some ESDI controllers to read a drive's capacity parameters directly from the drive, as well as to control defect mapping, but several manufacturers had different methods for writing this information on the drive. When you install an ESDI drive, in some cases the controller automatically reads the parameter and defect information directly from the drive. In other cases, however, you still have to enter this information manually, as with ST-506/412.

The ESDI's enhanced defect-mapping commands provide a standard way for the PC system to read a defect map from a drive, which means that the manufacturer's defect list can be written to the drive as a file. The defect-list file then can be read by the controller and low-level format software, eliminating the need for the installer to type these entries from the keyboard and enabling the format program to update the defect list with new entries if it finds new defects during the low-level format or the surface analysis.

Most ESDI implementations have drives formatted to 32 sectors per track or more (80 or more sectors per track are possible)—many more sectors per track than the standard ST-506/412 implementation of 17 to 26 sectors per track. The greater density results in two or more times the data-transfer rate, with a 1:1 interleave. Almost without exception, ESDI controllers support a 1:1 interleave, which allows for a transfer rate of 1MB per second or greater.

Because ESDI is much like the ST-506/412 interface, it can replace that interface without affecting software in the system. Most ESDI controllers are register-compatible with the older ST-506/412 controllers, which enables OS/2 and other non-DOS operating systems to run with few or no problems. The ROM BIOS interface to ESDI is similar to the ST-506/412 standard, and many low-level disk utilities that run on one interface will run on the other. To take advantage of ESDI defect mapping and other special features, however, use a low-level format and surface-analysis utility designed for ESDI (such as the ones usually built into the controller ROM BIOS and called by DEBUG).

During the late '80s, most high-end systems from major manufacturers were equipped with ESDI controllers and drives. More recently, manufacturers have been equipping high end systems with SCSI. The SCSI interface allows for much greater expandability, supports more types of devices than ESDI does, and offers equal or greater performance. I no longer recommend installing ESDI drives unless you are upgrading a system that already has an ESDI controller.

ESDI Interface Connectors. ESDI (Enhanced Small Device Interface) uses two connections, a 34-pin Control connector and a 20-pin Data connector. Tables 14.13 and 14.14 show the pinouts for these connectors.

Table 14.13 ESDI (Enhanced Small Device Interface) 34-Pin Control Connector Pinout

Signal Name	Pin	Pin	Signal Name
GROUND	1	2	-HD SLCT 3
GROUND	3	4	-HD SLCT 2
GROUND	5	6	-WRITE GATE
GROUND	7	8	-CNFG/STATUS
GROUND	9	10	-XFER ACK

Signal Name	Pin	Pin	Signal Name
GROUND	11	12	-ATTENTION
GROUND	13	14	-HD SLCT 0
KEY (no pin)	15	16	-SECTOR
GROUND	17	18	-HD SLCT 1
GROUND	19	20	-INDEX
GROUND	21	22	-READY
GROUND	23	24	-XFER REQ
GROUND	25	26	-DRV SLCT 0
GROUND	27	28	-DRV SLCT 1
GROUND	29	30	Reserved
GROUND	31	32	-READ GATE
GROUND	33	34	-CMD DATA

Table 14.14 ESDI (Enhanced Small Device Interface) 20-Pin Data Connector Pinout

Signal Name	Pin	Pin	Signal Name
-DRV SLCTD	1	2	-SECTOR
-CMD COMPL	3	4	-ADDR MK EN
GROUND	5	6	GROUND
+WRITE CLK	7	8	-WRITE CLK
GROUND	9	10	+RD/REF CLK
-RD/REF CLK	11	12	GROUND
+NRZ WRITE	13	14	-NRZ WRITE
GROUND	15	16	GROUND
+NRZ READ	17	18	-NRZ READ
GROUND	19	20	-INDEX

ESDI Drive Configuration. The ESDI interface was modeled after the ST-506/412 interface and shares virtually all the same types of configuration items and procedures. The 34-pin control and 20-pin data cables are identical to those used in an ST-506/412 installation, and all the configuration procedures with regard to Drive Select jumpers, twisted cables, and terminating resistors are the same as with ST-506/412.

Follow the configuration procedures for ST-506/412 drives when configuring ESDI drives. Since ESDI was developed from the ST-506/412 interface it shares many characteristics including the type of cabling used as well as how it is configured.

The IDE Interface

Integrated Drive Electronics (IDE) is a generic term applied to any drive with an integrated (built-in) disk controller. The first drives with integrated controllers were Hardcards; today, a variety of drives with integrated controllers are available. In a drive with IDE, the disk controller is integrated into the drive, and this combination drive/controller assembly usually plugs into a bus connector on the motherboard or on a bus adapter card. Combining the drive and controller greatly simplifies installation, because there are no separate power or signal cables from the controller to the drive. Also, when the controller and the drive are assembled as a unit, the number of total components is reduced, signal paths are shorter, and the electrical connections are more noise-resistant, resulting in a more reliable design than is possible when a separate controller, connected to the drive by cables, is used.

Placing the controller (including Endec) on the drive gives IDE drives an inherent reliability advantage over interfaces with separate controllers. Reliability is increased because the data encoding, from digital to analog, is performed directly on the drive in a tight noise-free environment; the timing-sensitive analog information does not have to travel along crude ribbon cables that are likely to pick up noise and to insert propagation delays into the signals. The integrated configuration allows for increases in the clock rate of the encoder, as well as the storage density of the drive.

Integrating the controller and drive also frees the controller and drive engineers from having to adhere to the strict standards imposed by the earlier interface standards. Engineers can design what essentially are custom drive and controller implementations, because no other controller would ever have to be connected to the drive. The resulting drive and controller combinations can offer higher performance than earlier stand-alone controller and drive setups. IDE drives sometimes are called drives with *embedded controllers*.

The IDE connector on motherboards in many systems is nothing more than a stripped-down bus slot. In ATA IDE installations, these connectors contain a 40-pin subset of the 98 pins that would be available in a standard 16-bit ISA bus slot. The pins used are only the signal pins required by a standard-type XT or AT hard disk controller. For example, because an AT-style disk controller uses only interrupt line 14, the motherboard AT IDE connector supplies only that interrupt line; no other interrupt lines are needed. The XT IDE motherboard connector supplies interrupt line 5 because that is what an XT controller would use.

Many people who use systems with IDE connectors on the motherboard believe that a hard disk controller is built into their motherboard, but the controller really is in the drive. I do not know of any PC systems that have hard disk controllers built into the motherboard.

When IDE drives are discussed, the ATA IDE variety usually is the only kind mentioned, because it is so popular. But other forms of IDE drives exist, based on other buses. For example, several PS/2 systems came with Micro Channel (MCA) IDE drives, which plug directly into a Micro Channel Bus slot (through an angle adapter or Interposer card).

An 8-bit ISA form of IDE also exists but was never very popular. Most IBM-compatible systems with the ISA or EISA Bus use AT-Bus (16-bit) IDE drives. The ATA IDE interface is by far the most popular type of drive interface available.

The primary advantage of IDE drives is cost. Because the separate controller or host adapter is eliminated and the cable connections are simplified, IDE drives cost much less than a standard controller-and-drive combination. These drives also are more reliable, because the controller is built into the drive. Therefore, the Endec or *data separator* (the divisor between the digital and analog signals on the drive) stays close to the media. Because the drive has a short analog-signal path, it is less susceptible to external noise and interference.

Another advantage is performance. IDE drives are some of the highest-performance drives available—but they also are among the lowest-performance drives. This apparent contradiction is because all IDE drives are different. You cannot make a blanket statement about the performance of IDE drives, because each drive is unique. The high-end models, however, offer performance that is equal or superior to that of any other type of drive on the market for a single-user, single-tasking operating system.

The greatest drawbacks to the IDE interface are capacity and expandability. IDE drives are not suited for large, high-performance systems that require large-capacity, high-performance drives. Incompatibilities among different manufacturers' standards make it difficult to install more than one IDE drive on a system. Because the controller is mounted on the drive, to add a second drive you must disable its controller and have it use the controller on the first drive. This process can be difficult because of the many different kinds of controllers on the drives. In many cases, to add a second IDE drive, you must use one made by the same manufacturer as the first, for compatibility purposes.

An additional drawback to IDE drives is that they normally are locked into a specific type of bus. You cannot easily move IDE drives of one type to a system that does not have the corresponding type of bus. Also, IDE drives are specific to IBM-compatible systems and cannot be used in foreign environments, such as Apple Macintosh systems, UNIX systems, and other computing environments. Because of these drawbacks, I normally use SCSI drives myself, but in many situations, IDE is the logical choice for other people.

IDE Origins. Technically, the first IDE drives were Hardcards. Companies such as the Plus Development division of Quantum took small 3.5-inch drives (either ST-506/412 or ESDI) and attached them directly to a standard controller. The assembly then was plugged into a bus slot as though it were a normal disk controller. Unfortunately, the mounting of a heavy, vibrating hard disk in an expansion slot with nothing but a single screw to hold it in place left a lot to be desired—not to mention the possible interference with adjacent cards due to the fact that many of these units were much thicker than a controller card alone.

Several companies got the idea that you could redesign the controller to replace the logic-board assembly on a standard hard disk and then mount it in a standard drive bay just like any other drive. Because the built-in controller in these drives still needed to plug directly into the expansion bus just like any other controller, a cable was run between the drive and one of the slots.

These connection problems were solved in different ways. Compaq was the first to incorporate a special bus adapter in its system to adapt the 98-pin AT bus edge connector on the motherboard to a smaller 40-pin header style connector that the drive would plug into. The 40-pin connectors were all that were needed, because it was known that a disk controller never would need more than 40 of the bus lines.

In 1987, IBM developed its own MCA IDE drives and connected them to the bus through a bus adapter device called an interposer card. These bus adapters (sometimes called paddle boards) needed only a few buffer chips and did not require any real circuitry, because the drive-based controller was designed to plug directly into the bus. The paddle board nickname came from the fact that they resembled game paddle or joystick adapters, which do not have much circuitry on them. Another 8-bit variation of IDE appeared in 8-bit ISA systems such as the PS/2 Model 30. The XT IDE interface uses a 40-pin connector and cable that is similar to, but not compatible with, the 16-bit version.

IDE Bus Versions. Three main types of IDE interfaces are available, with the differences based on three different bus standards:

- AT Attachment (ATA) IDE (16-bit ISA)
- XT IDE (8-bit ISA)
- MCA IDE (16-bit Micro Channel)

The XT and ATA versions have standardized on 40-pin connectors and cables, but the connectors have slightly different pinouts, rendering them incompatible with one another. MCA IDE uses a completely different 72-pin connector and is designed for MCA bus systems only.

In most cases, you must use the type of IDE drive that matches your system bus. This situation means that XT IDE drives work only in XT-class 8-bit ISA slot systems, AT IDE drives work only in AT-class 16-bit ISA or EISA slot systems, and MCA IDE drives work only in Micro Channel systems (such as the IBM PS/2 Model 50 or higher). A company called Silicon Valley offers adapter cards for XT systems that will run ATA IDE drives. Other companies, such as Arco Electronics and Sigma Data, have IDE adapters for Micro Channel systems that allow ATA IDE drives to be used on these systems. (You can find these vendors in Appendix B.) These adapters are very useful for XT or PS/2 systems, because there is a very limited selection of XT or MCA IDE drives, whereas the selection of ATA drives is virtually unlimited.

In most modern ISA and EISA systems, you will find an ATA connector on the motherboard. If your motherboard does not have one of these connectors and you want to attach an AT IDE drive to your system, you can purchase an adapter card that changes your 98-pin slot connector to the 40-pin IDE connector. These adapter cards are nothing more than buffered cables; they are not really controllers. The controller is built into the drive. Some of the cards offer additional features, such as an on-board ROM BIOS or cache memory.

ATA IDE. CDC, Western Digital, and Compaq actually created what could be called the first ATA type IDE interface drive and were the first to establish the 40-pin IDE connector pinout. The first ATA IDE drives were 5.25-inch half-height CDC 40M units (I believe that they had a green activity LED) with integrated WD controllers sold in the first Compaq 386 systems way back in '86. After that, Compaq helped found a company called Conner Peripherals to supply Compaq with IDE drives. Conner originally made drives only for Compaq but later, Compaq sold much of its ownership of Conner.

Eventually, the 40-pin IDE connector and drive interface method was placed before one of the ANSI standards committees, which, in conjunction with drive manufacturers, ironed out some deficiencies, tied up some loose ends, and published what is known as the CAM ATA (Common Access Method AT Attachment) interface. The CAM Committee was formed in October 1988, and the first working document of the AT Attachment interface was introduced in March 1989. Before the CAM ATA standard, many companies that followed CDC, such as Conner Peripherals, made proprietary changes to what had been done by CDC. As a result, many older ATA drives are very difficult to integrate into a dual-drive setup that has newer drives.

Some areas of the ATA standard have been left open for vendor-specific commands and functions. These vendor-specific commands and functions are the main reason why it is so difficult to low-level format IDE drives. To work properly, the formatter that you are using usually must know the specific vendor-unique commands for rewriting sector headers and remapping defects. Unfortunately, these and other specific drive commands differ from OEM to OEM, clouding the "standard" somewhat.

It is important to note that only the ATA IDE interface has been standardized by the industry. The XT IDE and MCA IDE never were adopted as industrywide standards and never became very popular. These interfaces no longer are in production, and no new systems of which I am aware come with these nonstandard IDE interfaces.

The CAM ATA Specification. The ATA specification was introduced in March 1989 by the ANSI CAM committee. You can obtain the current working-draft version of this standard, X3T9.2/791D Rev 4a or later, from Global Engineering Documents, which is listed in Appendix B. This standard has gone a long way toward eliminating incompatibilities and problems with interfacing IDE drives to ISA and EISA systems. The ATA specification defines the signals on the 40-pin connector, the functions and timings of these signals, cable specifications, and so on. The following section lists some of the elements and functions defined by the CAM ATA specification.

Dual-Drive Configurations. Dual-drive ATA installations can be problematic, because each drive has its own controller and both controllers must function while being connected to the same bus. There has to be a way to ensure that only one of the two controllers will respond to a command at a time.

The ATA standard provides the option of operating on the AT Bus with two drives in a daisy-chained configuration. The primary drive (drive 0) is called the *master*, and the secondary drive (drive 1) is the *slave*. You designate a drive as being master or slave by setting a jumper or switch on the drive or by using a special line in the interface called the Cable Select (CSEL) pin.

When only one drive is installed, the controller responds to all commands from the system. When two drives (and, therefore, two controllers) are installed, all commands from the system are received by both controllers. Each controller then must be set up to respond only to commands for itself. In this situation, one controller then must be designated as the master and the other as the slave. When the system sends a command for a specific drive, the controller on the other drive must remain silent while the selected controller and drive are functioning. You handle discrimination between the two controllers by setting a special bit (the DRV bit) in the Drive/Head Register of a command block.

ATA I/O Connector. The ATA interface connector is a 40-pin header-type connector that should be keyed to prevent the possibility of installing it upside down. A key is provided by the removal of pin 20, and the corresponding pin on the cable connector should be plugged to prevent a backward installation. The use of keyed connectors and cables is highly recommended, because plugging an IDE cable in backward can damage both the drive and the bus adapter circuits (although I have done it myself many times with no smoked parts yet!).

Table 14.15 shows the ATA-IDE interface connector pinout.

Table 14.15 ATA (AT Attachment) IDE (Integrated Drive Electronics) Connector Pinout

Signal Name	Pin	Pin	Signal Name
-RESET	1	2	GROUND
Data Bit 7	3	4	Data Bit 8
Data Bit 6	5	6	Data Bit 9
Data Bit 5	7	8	Data Bit 10
Data Bit 4	9	10	Data Bit 11
Data Bit 3	11	12	Data Bit 12
Data Bit 2	13	14	Data Bit 13
Data Bit 1	15	16	Data Bit 14
Data Bit 0	17	18	Data Bit 15
GROUND	19	20	KEY (pin missing)
DRQ 3	21	22	GROUND
-IOW	23	24	GROUND
-IOR	25	26	GROUND
I/O CH RDY	27	28	SPSYNC:CSEL
-DACK 3	29	30	GROUND
IRQ 14	31	32	-IOCS16
Address Bit 1	33	34	-PDIAG
Address Bit 0	35	36	Address Bit 2
-CS1FX	37	38	-CS3FX
-DA/SP	39	40	GROUND
+5 Vdc (Logic)	41	42	+5 Vdc (Motor)
GROUND	43	44	-TYPE (0=ATA)

ATA I/O Cable. A 40-conductor ribbon cable is specified to carry signals between the bus adapter circuits and the drive (controller). To maximize signal integrity and to eliminate potential timing and noise problems, the cable should not be longer than 0.46 meters (18 inches).

ATA Signals. The ATA interface signals and connector pinout are listed in Appendix A. This section describes some of the most important signals in more detail.

Pin 20 is used as a key pin for cable orientation and is not connected through the interface. This pin should be missing from any ATA connectors, and the cable should have the pin-20 hole in the connector plugged off to prevent the cable from being plugged in backward.

Pin 39 carries the Drive Active/Slave Present (DASP) signal, which is a dual-purpose, time-multiplexed signal. During power-on initialization, this signal indicates whether a slave drive is present on the interface. After that, each drive asserts the signal to indicate that it is active. Early drives could not multiplex these functions and required special jumper settings to work with other drives. Standardizing this function to allow for compatible dual-drive installations is one of the features of the ATA standard.

Pin 28 carries the Cable Select or Spindle Synchronization signal (CSEL or SPSYNC), which is a dual-purpose conductor; a given installation, however, may use only one of the two functions. The CSEL (Cable Select) function is the most widely used and is designed to control the designation of a drive as master (drive 0) or slave (drive 1) without requiring jumper settings on the drives. If a drive sees the CSEL as being grounded, the drive is a master; if CSEL is open, the drive is a slave.

You can install special cabling to ground CSEL selectively. This installation normally is accomplished through a Y-cable arrangement, with the IDE bus connector in the middle and each drive at opposite ends of the cable. One leg of the Y has the CSEL line connected through, indicating a master drive; the other leg has the CSEL line open (conductor interrupted or removed), making the drive at that end the slave.

ATA Commands. One of the best features of the ATA IDE interface is the enhanced command set. The ATA IDE interface was modeled after the WD1003 controller that IBM used in the original AT system. All ATA IDE drives must support the original WD command set (eight commands), with no exceptions, which is why IDE drives are so easy to install in systems today. All IBM-compatible systems have built-in ROM BIOS support for the WD1003, which means that essentially, they support ATA IDE as well.

In addition to supporting all the WD1003 commands, the ATA specification added numerous other commands to enhance performance and capabilities. These commands are an optional part of the ATA interface, but several of them are used in most drives available today and are very important to the performance and use of ATA drives in general.

Perhaps the most important is the *Identify Drive* command. This command causes the drive to transmit a 512-byte block of data that provides all details about the drive. Through this command, any program (including the system BIOS) can find out exactly what type of drive is connected, including the drive manufacturer, model number, operating parameters, and even the serial number of the drive. Many modern BIOSes use this information to automatically receive and enter the drive's parameters into CMOS memory, eliminating the need for the user to enter these parameters manually during system configuration. This arrangement helps prevent mistakes that later can lead to data loss when the user no longer remembers what parameters he or she used during setup.

The Identify Drive data can tell you many things about your drive, including the following:

- Number of cylinders in the recommended (default) translation mode
- Number of heads in the recommended (default) translation mode
- Number of sectors per track in the recommended (default) translation mode
- Number of cylinders in the current translation mode
- Number of heads in the current translation mode
- Number of sectors per track in the current translation mode
- Manufacturer and model number
- Firmware revision
- Serial number
- Buffer type, indicating sector buffering or caching capabilities

Several public-domain programs can execute this command to the drive and report the information on-screen. I use the IDEINFO program (which can be downloaded from the IBM Hardware Forum on CompuServe) or the IDEDIAG utility (which can be downloaded from the Western Digital BBS). Phone numbers for these information services appear in Appendix B. I find these programs especially useful when I am trying to install IDE drives and need to know the correct parameters for a user-definable BIOS type. These programs get the information directly from the drive itself.

Two other very important commands are the Read Multiple and Write Multiple commands. These commands permit multiple-sector data transfers and, when combined with block-mode Programmed I/O (PIO) capabilities in the system, can result in incredible data-transfer rates many times faster than single-sector PIO transfers.

Tip

If you want the ultimate in IDE performance and installation ease, make sure that your motherboard BIOS goes beyond supporting just the WD1003 command set and also supports block-mode Programmed I/O (PIO) and the Identify Drive commands. This support will allow your BIOS to execute data transfers to and from the IDE drive several times faster than normal, and also will make installation and configuration easier because the BIOS will be able to detect the drive-parameter information automatically. Block PIO and automatic detection of the drive type are included in the latest versions of most PC BIOSs.

There are many other enhanced commands, including room for a given drive manufacturer to implement what are called vendor-unique commands. These commands often are used by a particular vendor for features unique to that vendor. Often, features such as low-level formatting and defect management are controlled by vendor-unique commands. This is why low-level format programs can be so specific to a particular manufacturer's IDE drives and why many manufacturers make their own LLF programs available.

ATA IDE Drive Categories. ATA-IDE drives can be divided into three main categories. These categories separate the drives by function (such as translation capabilities) and design (which can affect features such as low-level formatting). The three drive categories are:

- Non-Intelligent ATA-IDE drives
- Intelligent ATA-IDE drives
- Intelligent Zoned Recording ATA-IDE drives

The following sections describe these categories.

Non-Intelligent IDE. As I stated earlier, the ATA standard requires that the built-in controller respond exactly as though it were a Western Digital WD1003 controller. This controller responds to a command set of eight commands. Early IDE drives supported these commands and had few, if any, other options. These early drives actually were more like regular ST-506/412 or ESDI controllers bolted directly to the drive than the more intelligent drives that we consider today to be IDE.

These drives were not considered to be intelligent IDE drives; an intelligent drive is supposed to have several capabilities that these early IDE drives lacked. The drives could not respond to any of the enhanced commands that were specified as (an optional) part of the ATA IDE specification, including the Identify Drive command. These drives also did not support sector translation, in which the physical parameters could be altered to appear as any set of logical cylinders, heads, and sectors. Enhanced commands and sector-translation support are what make an IDE drive an intelligent IDE drive, and these features were not available in the early IDE drives.

These drives could be low-level formatted in the same manner as any normal ST-506/412 or ESDI drive. They were universally low-level formatted at the factory, with factory-calculated optimum interleave (usually, 1:1) and head- and cylinder-skew factors. Also, factory defects were recorded in a special area on the drive; they no longer were written on a sticker pasted to the exterior. Unfortunately, this arrangement means that if you low-level format these drives in the field, you most likely will alter these settings (especially the skew factors) from what the factory set as optimum, as well as wipe out the factory-written defect table.

Some manufacturers released special low-level format routines that would reformat the drives while preserving these settings, but others did not make such programs available. Because they did not want you to overwrite the defect list or potentially slow the drive, most manufacturers stated that you should never low-level format their IDE drives.

This statement started a myth that the drives could somehow be damaged or rendered inoperable by such a format, which truly is not the case. One rumor was that the servo information could be overwritten, which would mean that you would have to send the drive back to the manufacturer for re-servoing. This also is not true; the servo information is protected and cannot be overwritten. The only consequence of an improper low-level format of these drives is the possible alteration of the skew factors and the potential loss of the factory defect maps.

The Disk Manager program by Ontrack is the best special-purpose format utility to use on these drives for formatting, because it is aware of these types of drives and often can restore the skew factors and preserve the defect information. If you are working with a drive that already has had the defect map overwritten, Disk Manager can perform a very good surface analysis that will mark off any of these areas that it finds. Disk Manager allows you to specify the skew factors and to mark defects at the sector level so that they will not cause problems later. Other general-purpose diagnostics that work especially well with IDE drives such as this include the Microscope program by Micro 2000.

Intelligent IDE. Later IDE drives became known as intelligent IDE drives. These drives support enhanced ATA commands, such as the Identify Drive command, and sector-translation capabilities.

These drives can be configured in two ways: in raw physical mode or in translation mode. To configure the drive in raw physical mode, you simply enter the CMOS drive parameters during setup so that they match the true physical parameters of the drive. For example, if the drive physically has 800 cylinders, 6 heads, and 50 sectors per track, you enter these figures during setup. To configure the drive in translation mode, you simply enter any combination of cylinders, heads, and sectors that adds up to equal or less than the true number of sectors on the drive.

In the example that I just used, the drive has a total of 240,000 sectors ($800 \times 6 \times 50$). All I have to do is figure out another set of parameters that adds up to equal or less than 240,000 sectors. The simplest way to do this is to cut the number of cylinders in half and double the number of heads. Thus, the new drive parameters become 400 cylinders, 12 heads, and 50 sectors per track. This method adds up to 240,000 sectors and enables the drive to work in translation mode.

When these drives are in translation mode, a low-level format cannot alter the interleave and skew factors; neither can it overwrite the factory defect-mapping information. A low-level format program can, however, perform additional defect mapping or sector sparing while in this mode.

If the drive is in true physical mode, a low-level format will rewrite the sector headers and modify the head and cylinder skewing. If done incorrectly, the format can be repaired by a proper low-level format program that allows you to set the correct head and cylinder skew. This task can be accomplished automatically by the drive manufacturer's recommended low-level format program (if available) or by other programs, such as Disk Manager by Ontrack. When you use Disk Manager, you have to enter the skew values manually; otherwise, the program uses predetermined defaults. To get the correct skew values, it is best to contact the drive manufacturer's technical-support department. You can calculate the skew values if the manufacturer cannot provide them.

To protect the skew factors and defect information on intelligent IDE drives, all you have to do is run them in translation mode. In translation mode, this information cannot be overwritten.

Intelligent Zoned Recording IDE. The last and most sophisticated IDE drives combine intelligence with Zoned Recording. With Zoned Recording, the drive has a variable number sectors per track in several zones across the surface of the drive. Because the PC BIOS can handle only a fixed number of sectors on all tracks, these drives always must run in translation mode. Because these drives are always in translation mode, you cannot alter the factory-set interleave and skew factors or wipe out the factory defect information.

You still can low-level format these drives, however, and use such a format to map or spare additional defective sectors that crop up during the life of the drive. To low-level format intelligent Zoned Recording drives, you need either a specific utility from the drive manufacturer or an IDE-aware program, such as Disk Manager by Ontrack or Microscope by Micro 2000.

IDE Drive Configuration. IDE drives can be both simple and troublesome to configure. Single-drive installations usually are very simple, with few, if any, special jumper settings to worry about. Multiple-drive configurations, however, can be a problem. Jumpers have to be set on both drives; and the names, locations, and even functions of these jumpers can vary from drive to drive.

Because the CAM ATA (Common Access Method AT Attachment) IDE specification was ironed out only after many companies were already making and selling drives, many older IDE drives have problems in dual-drive installations, especially when the drives are from different manufacturers. In some cases, two particular drives may not function together at all. Fortunately, most of the newer drives follow the CAM ATA specification, which clears up this problem. Drives that follow the specification have no problems in dual-drive installations.

Cable Configuration. The cable connection to IDE drives usually is very simple. There is a single 40-pin cable that normally has three pin-header style connectors on it. One of the connectors plugs into the IDE interface connector; the other two plug into the primary and secondary drives. The cable normally runs from the IDE connector to both drives in a daisy-chain arrangement. On one end, this cable plugs into the IDE interface connector, which is located on the motherboard in many systems but also may be located on an IDE interface adapter card. The cable then connects to the secondary (D) and primary (C) drives in succession, with the primary drive usually (but not always) being at the end of the cable opposite the IDE interface connector.

There are no terminating resistors to set with IDE drives; instead a distributed termination circuit is built into all IDE drives. The last drive on the cable need not be the primary drive, so you actually may find the primary or secondary drive at either connector. Jumpers on the drives themselves normally control whether a drive responds as primary or secondary.

You may see a different arrangement of cable connections in some IDE installations. In some installations, the middle connector is plugged into the motherboard, and the primary and secondary drives are at opposite ends of the cable in a Y arrangement. If you see this arrangement, be careful; in some of these Y-cable installations, the cable, rather than jumpers on the drives, actually controls which drive is primary and which is secondary.

Controlling master/slave selection via the cable rather than jumpers on the drive is performed via a special signal on the IDE interface called *CSEL (Cable SELect),* which is on pin 28 of the interface. If the CSEL line is connected through from the drive to the IDE interface connector, the drive automatically is designated as primary. If the CSEL line is open between a drive and the IDE interface connector, that drive automatically is designated as secondary.

In the Y-cable approach, the IDE interface connector is in the middle of the cable, and a separate length of cable goes to each drive. Study this type of cable closely. If one of the ends of the Y has line 28 open (usually, a hole in the cable through that wire), only the secondary drive can be plugged into that connector. HP Vectra PC systems use exactly this type of IDE cable arrangement. This type of setup eliminates the need to set jumpers on the IDE drives to configure them for primary or secondary operation, but the setup can be troublesome if you do not know about it.

IDE Drive Jumper Settings. Configuring IDE drives can be simple, as is the case with most single-drive installations, or troublesome, especially where it comes to mixing two drives from different manufacturers on a single cable.

Most IDE drives come in three configurations:

- Single-drive (master)
- Master (dual-drive)
- Slave (dual-drive)

Because each IDE drive has its own controller, you must specifically tell one drive to be the master and the other to be the slave. There's no functional difference between the two, except that the drive that's specified as the slave will assert the DASP signal after a system reset that informs the master that a slave drive is present in the system. The master drive then pays attention to the Drive Select line, which it otherwise ignores. Telling a drive that it's the slave also usually causes it to delay its spinup for several seconds, to allow the master to get going and thus to lessen the load on the system's power supply.

Until the ATA IDE specification, no common implementation for drive configuration was in use. Some drive companies even used different master/slave methods for different models of drives. Because of these incompatibilities, some drives work together only in a specific master/slave or slave/master order. This situation affects mostly older IDE drives that were introduced before the ATA specification.

Most drives that fully follow the ATA specification now need only one jumper (Master/Slave) for configuration. A few also need a Slave Present jumper as well. Table 14.16 shows the jumper settings required by most ATA IDE drives.

Table 14.16 Jumper Settings for Most ATA IDE-Compatible Drives

Jumper Name	Single-Drive	Dual-Drive Master	Dual-Drive Slave
Master (M/S)	On	On	Off
Slave Present (SP)	Off	On	Off

The Master jumper indicates that the drive is a master or a slave. Some drives also require a Slave Present jumper, which is used only in a dual-drive setup and then installed only on the master drive, which is somewhat confusing. This jumper tells the master that a slave drive is attached. With many ATA IDE drives, the Master jumper is optional and may be left off. Installing this jumper doesn't hurt in these cases and may eliminate confusion, so I recommend that you install the jumpers listed here.

Conner Peripherals Drives. Because they were introduced before the ATA IDE specification was formalized, Conner Peripherals drives often are different in configuration from many other-brand drives. When you mix and match IDE hard drives from different manufacturers, the drives are not always fully compatible. Table 14.17 shows the jumper settings that are correct for most Conner IDE drive installations.

Table 14.17 Jumper Settings for Conner Peripherals IDE Drives

Jumper Name	Single Drive	Dual-Drive Master	Dual-Drive Slave
Master or Slave (C/D)	On	On	Off
Drive Slave Present (DSP)	Off	On	Off
Host Slave Present (HSP)	Off	Off	On
Drive Active (ACT)	On	On	Off

The C/D jumper is used to determine whether the drive is a master (drive C) or a slave (drive D). The drive is configured as master when this jumper is on. The DSP jumper indicates that a slave drive is present. The HSP jumper causes the drive to send the Slave Present signal to the master drive. The ACT jumper enables the master drive to signal when it is active.

Some Conner drives are not set up to support the industry-standard CAM ATA interface by default. The problems show up when you attempt to connect another manufacturer's drive to some Conner drives in either a master or slave role. Fortunately, you can correct many of these situations by changing the configuration of the drive.

You can make this change in two ways. One way is to use a special program to semipermanently change the mode of the drive. A special file available on the Conner BBS, called *FEATURE.COM*, contains a program that displays the current ISA/CAM setting and allows the setting to be changed. The change actually is stored in a feature byte in the firmware of the drive, and after this byte is changed, most other manufacturers' drives will work with the Conner drives. The program also can be used to reset the feature byte back to its original configuration, which is best when you are connecting to other Conner drives.

The second method for changing this configuration is available on some Conner drives. These drives also have a special jumper called ATA/ISA. This jumper almost always should be installed in the ATA position to provide compatibility with the ATA standard. If you are using only Conner drives, you can leave this jumper in ISA mode if you want. Some Conner drives have a separate jumper (E1) that can delay startup of the drive to minimize the load on the power supply. This jumper should be enabled on any drive that is configured as a slave. Most other drives automatically delay startup of the slave drive for a few seconds.

Most Conner drives also have a special 12-pin connector that is used to drive an optional LED (pin 1, LED +5v; and pin 2, ground), as well as to connect to special factory equipment for low-level formatting and configuration. A company called TCE (see Appendix B) sells a device called The Conner, which connects to this port and permits full factory-level initialization, formatting, and testing of Conner drives. I consider this piece of gear to be essential to anybody who services or supports a large number of Conner Peripherals drives. Notice that Compaq uses Conner drives in most of its systems.

For more information on any specific Conner drive, you can use the company's FaxBack system at (800) 4CONNER. Through this system, you can get drive information and jumper settings that are specific to Conner drives.

XT-Bus (8-Bit) IDE. Many systems with XT ISA bus architecture used XT IDE hard drives. The IDE interface in these systems usually is built into the motherboard. The IBM PS/2 Model 25, 25-286, 30, and 30-286 systems used an 8-bit XT IDE interface. These 8-bit XT IDE drives are difficult to find; few manufacturers other than IBM, Western Digital, and Seagate made them; and none of these drives were available in capacities beyond 40MB.

Since the ATA IDE interface is a 16-bit design, it could not be used in 8-bit (XT type) systems, so some of the drive manufacturers standardized on an XT-Bus (8-bit) IDE interface for XT class systems. These drives were never very popular, and were usually only available in capacities from 20MB to 40MB. Table 14.18 shows the industry standard 8-bit IDE connector pinout.

Table 14.18 XT-Bus IDE (Integrated Drive Electronics) Connector Pinout

Signal Name	Pin	Pin	Signal Name
-RESET	1	2	GROUND
Data Bit 7	3	4	GROUND
Data Bit 6	5	6	GROUND
Data Bit 5	7	8	GROUND
Data Bit 4	9	10	GROUND
Data Bit 3	11	12	GROUND
Data Bit 2	13	14	GROUND
Data Bit 1	15	16	GROUND
Data Bit 0	17	18	GROUND
GROUND	19	20	KEY (pin missing)
AEN	21	22	GROUND
-IOW	23	24	GROUND
-IOR	25	26	GROUND
-DACK 3	27	28	GROUND
DRQ 3	29	30	GROUND
IRQ 5	31	32	GROUND
Address Bit 1	33	34	GROUND
Address Bit 0	35	36	GROUND
-CS1FX	37	38	GROUND
-Drive Active	39	40	GROUND

Note that IBM used a custom version of the XT-Bus IDE interface in the PS/2 Model 25 and Model 30 systems. The pinout for the custom IBM XT-Bus IDE connector is shown in table 14.19.

Table 14.19 IBM Unique XT-Bus (PS/2 Model 25 and 30) IDE Connector Pinout

Signal Name	Pin	Pin	Signal Name
-RESET	1	2	-Disk Installed
Data Bit 0	3	4	GROUND
Data Bit 1	5	6	GROUND
Data Bit 2	7	8	GROUND
Data Bit 3	9	10	GROUND
Data Bit 4	11	12	GROUND
Data Bit 5	13	14	GROUND
Data Bit 6	15	16	GROUND
Data Bit 7	17	18	GROUND

(continues)

Table 14.19 Continued

Signal Name	Pin	Pin	Signal Name
-IOR	19	20	GROUND
-IOW	21	22	GROUND
-CS1FX	23	24	GROUND
Address Bit 0	25	26	GROUND
Address Bit 1	27	28	GROUND
Address Bit 2	29	30	+5 Vdc
RESERVED	31	32	+5 Vdc
-DACK 3	33	34	GROUND
DRQ 3	35	36	GROUND
IRQ 5	37	38	GROUND
I/O CH RDY	39	40	+12 Vdc
Spare	41	42	+12 Vdc
Spare	39	44	+12 Vdc

The newer PS/1, PS/Valuepoint, and PS/2 systems with 16-bit ISA architecture use ATA IDE drives. Because nearly all hard disk manufacturers make a multitude of drives with the ATA IDE interface, these systems are easy to upgrade or repair. ATA IDE drives are available in capacities up to and beyond 1GB.

MCA IDE. The IBM PS/2 Models 50 and higher come with Micro Channel Architecture (MCA) bus slots. Although most of these systems now use SCSI drives, for some time IBM used a type of MCA IDE drive in these systems. MCA IDE is a form of IDE interface, but it is designed for the MCA bus and is not compatible with the more industry-standard ATA IDE interface. Few companies other than IBM and Western Digital make replacement MCA IDE drives for these systems. I recommend replacing these drives with ATA IDE drives, using adapters from Arco Electronics or Sigma Data, or switching to SCSI drives instead. The IBM MCA IDE drives are expensive for the limited capacity that they offer.

The pinout of the MCA IDE connector is shown in table 14.20.

Table 14.20 MCA (Micro Channel Architecture) IDE Connector Pinout

Signal Name	Pin	Pin	Signal Name
-CD SETUP	A1	B1	Address Bit 15
Address Bit 13	A2	B2	Address Bit 14
GROUND	A3	B3	GROUND
Address Bit 11	A4	B4	OSC (14.3 MHz)
Address Bit 10	A5	B5	GROUND
Address Bit 9	A6	B6	Address Bit 12
+5 Vdc	A7	B7	-CMD
Address Bit 8	A8	B8	-CD SFDBK

Signal Name	Pin	Pin	Signal Name
Address Bit 7	A9	B9	GROUND
Address Bit 6	A10	B10	Data Bit 1
+5 Vdc	A11	B11	Data Bit 3
Address Bit 5	A12	B12	Data Bit 4
Address Bit 4	A13	B13	GROUND
Address Bit 3	A14	B14	CHRESET
+5 Vdc	A15	B15	Data Bit 8
Address Bit 2	A16	B16	Data Bit 9
Address Bit 1	A17	B17	GROUND
Address Bit 0	A18	B18	Data Bit 12
+12 Vdc	A19	B19	Data Bit 14
-ADL	A20	B20	Data Bit 15
-PREEMPT	A21	B21	GROUND
-BURST	A22	B22	Data Bit 0
+5 Vdc	A23	B23	Data Bit 2
ARB 0	A24	B24	Data Bit 5
ARB 1	A25	B25	GROUND
ARB 2	A26	B26	Data Bit 6
+12 Vdc	A27	B27	Data Bit 7
ARB 3	A28	B28	Data Bit 10
+ARB/-GRANT	A29	B29	GROUND
-TC	A30	B30	Data Bit 11
+5 Vdc	A31	B31	Data Bit 13
-S0	A32	B32	-SBHE
-S1	A33	B33	GROUND
+M/-IO	A34	B34	-CD DS 16
GROUND	A35	B35	-IRQ 14
CD CHRDY	A36	B36	GROUND

Introduction to SCSI

SCSI (pronounced "scuzzy") stands for *Small Computer System Interface*. This interface has its roots in SASI, the *Shugart Associates System Interface*. SCSI is not a disk interface, but a systems-level interface. SCSI is not a type of controller, but a bus that supports as many as eight devices. One of these devices, the *host adapter*, functions as the gateway between the SCSI bus and the PC system bus. The SCSI bus itself does not talk directly with devices such as hard disks; instead, it talks to the controller that is built into the drive.

A single SCSI bus can support as many as eight *physical units*, usually called *SCSI IDs*. One of these units is the adapter card in your PC; the other seven can be other peripherals. You could have hard disks, tape drives, CD-ROM drives, a graphics scanner, or other devices (up to seven total) attached to a single SCSI host adapter. Most systems support up to four host adapters, each with seven devices, for a total 28 devices!

When you purchase a SCSI hard disk, you usually are purchasing the drive, controller, and SCSI adapter in one circuit. This type of drive usually is called an *embedded SCSI drive*; the SCSI interface is built into the drive. Most SCSI hard drives actually are IDE drives with SCSI bus adapter circuits added. You do not need to know what type of controller is inside the SCSI drive, because your system cannot talk directly to the controller as though it were plugged into the system bus, like a standard controller. Instead, communications go through the SCSI host adapter installed in the system bus. You can access the drive only with the SCSI protocols.

Apple originally rallied around SCSI as being an inexpensive way out of the bind in which it put itself with the Macintosh. When the engineers at Apple realized the problem in making the Macintosh a closed system (with no slots), they decided that the easiest way to gain expandability was to build a SCSI port into the system, which is how external peripherals can be added to the slotless Macs. Because PC systems always have been expandable, the push toward SCSI has not been as urgent. With eight bus slots supporting different devices and controllers in IBM and IBM-compatible systems, it seemed as though SCSI was not needed.

SCSI now is becoming popular in the IBM-based computer world because of the great expandability that it offers and the number of devices that are available with built-in SCSI. One thing that stalled acceptance of SCSI in the PC marketplace was the lack of a real standard; the SCSI standard was designed primarily by a committee. No single manufacturer has led the way, at least in the IBM arena; each company has its own interpretation of how SCSI should be implemented, particularly at the host-adapter level.

SCSI is a standard, in much the same way that RS-232 is a standard. The SCSI standard (like the RS-232 standard), however, defines only the hardware connections, not the driver specifications required to communicate with the devices. Software ties the SCSI subsystem into your PC, but unfortunately, most of the driver programs work only for a specific device and a specific host adapter. For example, a graphics scanner comes with its own SCSI host adapter to connect to the system; a CD-ROM drive comes with another (different) SCSI host adapter and driver software that works only with that SCSI adapter. On a system with those two SCSI adapters, you would need a third SCSI host adapter to run SCSI hard disk drives, because the host adapters supplied by the scanner and CD-ROM companies do not include a built-in, self-booting BIOS that supports hard disk drives.

SCSI has become something of a mess in the IBM world because of the lack of a host-adapter standard, a software interface standard, and standard ROM BIOS support for hard disk drives attached to the SCSI bus. Fortunately, some simple recommendations can keep you from living this compatibility nightmare!

The lack of capability to run hard disks off the SCSI bus, and to boot from these drives and use a variety of operating systems, is a problem that results from the lack of an interface standard. The standard IBM XT and AT ROM BIOS software was designed to talk to ST-506/412 hard disk controllers. The software easily was modified to work with ESDI because ESDI controllers are similar to ST-506/412 controllers at the register level.

(This similarity at the register level enabled manufacturers to easily design self-booting, ROM-BIOS-supported ESDI drives.). The same can be said of IDE, which completely emulates the WD1003 ST-506/412 controller interface and works perfectly with the existing BIOS as well. SCSI is so different from these other standard disk interfaces that a new set of ROM BIOS routines are necessary to support the system so that it can self-boot. The newer IBM PS/2 systems that come with SCSI drives have this support built into the motherboard BIOS or as an extension BIOS on the SCSI host adapter.

Companies such as Adaptec and Future Domain have produced SCSI cards with built-in ROM BIOS support for several years, but these BIOS routines were limited to running the drives only under DOS. The BIOS would not run in the AT-protected mode, and other operating systems included drivers for only the standard ST-506/412 and ESDI controllers. Thus, running SCSI was impossible under many non-DOS operating systems. This situation has changed significantly, however; IBM now supports many third-party SCSI host adapters in OS/2, especially those from Adaptec and Future Domain. For compatibility reasons, I usually recommend using SCSI adapters from these two companies, or any other adapters that are fully hardware-compatible with the Adapted and Future Domain adapters.

Because of the lead taken by Apple in developing systems software (operating systems and ROM) support for SCSI, peripherals connect to Apple systems in fairly standard ways. Until recently, this kind of standard-setting leadership was lacking for SCSI in the IBM world. This situation changed on March 20, 1990, when IBM introduced several "standard" SCSI adapters and peripherals for the IBM PS/2 systems, with complete ROM BIOS and full operating-system support.

IBM has standardized on SCSI for nearly all its high-end systems. In these systems, a SCSI host adapter card is in one of the slots, or the system has a SCSI host adapter built into the motherboard. This arrangement is similar in appearance to the IDE interface, because a single cable runs from the motherboard to the SCSI drive, but SCSI supports as many as seven devices (some of which may not be hard disks), whereas IDE supports only two devices, which must be either a hard disk or a tape drive. PS/2 systems with SCSI drives are easy to upgrade, because virtually any third-party SCSI drive will plug in and function.

The example set by IBM is causing other manufacturers to supply systems with either SCSI host adapters or SCSI interfaces integrated into the motherboards. As SCSI becomes more and more popular in the PC world, SCSI peripheral integration will be easier, due to better operating-system and device-driver support.

ANSI SCSI standards. The SCSI standard defines the physical and electrical parameters of a parallel I/O bus used to connect computers and peripheral devices in daisy-chain fashion. The standard supports devices such as disk drives, tape drives, and CD-ROM drives. The original SCSI standard (ANSI X3.131-1986) was approved in 1986; SCSI-2 was approved in January of 1994, and a new revision called SCSI-3, is being developed.

The SCSI interface is defined as a standard by ANSI (American National Standards Institute), an organization that approves and publishes standards. The X3 Task Group operates as an ASC (Accredited Standards Committee) under ANSI to develop Information Processing System standards. X3T9 is the I/O Interfaces group, and X3T9.2 specifically is in charge of low-level interfaces such as SCSI and ATA-IDE (among others). The original SCSI-1 standard was published by the X3T9 ANSI group in 1986 and is officially published by ANSI as X3.131-1986.

One problem with the original SCSI-1 document was that many of the commands and features were optional, and there was little or no guarantee that a particular peripheral would support the expected commands. This problem caused the industry as a whole to define a set of 18 basic SCSI commands called the Common Command Set (CCS), which would become the minimum set of commands supported by all peripherals. This Common Command Set became the basis for what is now the SCSI-2 specification.

In addition to formal support for the CCS, SCSI-2 provided additional definitions for commands to access CD-ROM drives (and their sound capabilities), tape drives, removable drives, optical drives, and several other peripherals. In addition, an optional higher speed called FAST SCSI-2 and a 16-bit version called WIDE SCSI-2 were defined. Another feature of SCSI-2 is command queuing, which enables a device to accept multiple commands and execute them in the order that the device deems to be most efficient. This feature is most beneficial when you are using a multitasking operating system that could be sending several requests on the SCSI bus at the same time.

The X3T9 group approved the SCSI-2 standard as X3.131-1990 in August 1990, but the document was recalled in December 1990 for changes before final ANSI publication. Final approval for the SCSI-2 document was finally made in January of 1994(!), although it has changed little from the original 1990 release. The SCSI-2 document is now called ANSI X3.131-1994. The official document is available from Global Engineering Documents or the ANSI committee, which are listed in Appendix B. You can also download working drafts of these documents from the NCR SCSI BBS, listed in the vendor list under NCR Microelectronics.

Most companies indicate that their host adapters follow both the ANSI X3.131-1986 (SCSI-1) as well as the x3.131-1994 (SCSI-2) standards. Note that since virtually all parts of SCSI-1 are supported in SCSI-2, virtually any SCSI-1 device is also considered SCSI-2 by default. Many manufacturers advertise that their devices are SCSI-2, but this does not mean that they support any of the additional *optional* features that were incorporated in the SCSI-2 revision.

For example, an optional part of the SCSI-2 specification includes a fast synchronous mode that doubles the standard synchronous transfer rate from 5MB per second to 10MB per second. This Fast SCSI transfer mode can be combined with 16-bit Wide or 32-bit Wide SCSI for transfer rates of up to 40MB per second. Currently, there are no 32-bit peripherals to speak of, and only a few 16-bit Wide SCSI devices and host adapters exist. Most SCSI implementations are 8-bit standard SCSI or Fast SCSI. Even devices which support none of the fast or wide modes can still be considered SCSI-2.

Table 14.21 shows the maximum transfer rates for the SCSI bus at various speeds and widths, as well as the cable type required for specific transfer widths.

Table 14.21 SCSI Data-Transfer Rates

Bus Width	Std. SCSI	Fast SCSI	Cable Type
8-bit	5 MB/s	10 MB/s	A
16-bit	10 MB/s	20 MB/s	P
32-bit	20 MB/s	40 MB/s	P&Q

MB/s = Megabytes per second.

Note

The A cable is the standard 50-pin SCSI cable, whereas the P and Q cables are 68-pin cables designed for 16-bit and 32-bit Wide SCSI. For 32-bit SCSI, P and Q cables must be used together. Pinouts for these cable connections are listed in this chapter.

So-called SCSI-1 adapters have no problems with SCSI-2 peripherals. In fact, as was stated earlier, virtually any SCSI-1 device can also legitimately be called SCSI-2. You can't take advantage of Fast or Wide transfer capabilities, but the extra commands defined in SCSI-2 can be sent by means of a SCSI-1 controller. In other words, nothing is different between SCSI-1 and SCSI-2 compliant hardware. For example, I am running a Toshiba MK-538FB 1.2GB Fast SCSI-2 drive with my IBM SCSI host adapter, and it runs fine. Most adapters are similar, in that they actually are SCSI-2 compatible, even if they advertise only SCSI-1 support. Since the SCSI-2 standard was not actually approved before January 1994, any devices which claimed to be SCSI-2 before that time were not officially in compliance with the standard. This is really not a problem, however since the SCSI-2 document had not changed appreciably since it was nearly approved in 1990.

SCSI Hard Disk Evolution and Construction. SCSI is not a disk interface, but a bus that supports SCSI bus interface adapters connected to disk and other device controllers. The first SCSI drives for PCs simply were standard ST-506/412 or ESDI drives with a separate SCSI bus interface adapter (sometimes called a bridge controller) that converted the ST-506/412 or ESDI interfaces to SCSI. This interface originally was in the form of a secondary logic board, and the entire assembly often was mounted in an external case.

The next step was to build the SCSI bus interface "converter" board directly into the drive's own logic board. Today, we call these drives *embedded* SCSI drives, because the SCSI interface is built in.

At that point, there was no need to conform to the absolute specifications of ST-506/412 or ESDI on the internal disk interface, because the only other device that the interface ever would have to talk to was built in as well. Thus, the disk-interface and controller-chipset manufacturers began to develop more customized chipsets that were based on the ST-506/412 or ESDI chipsets already available but offered more features and higher performance. Today, if you look at a typical SCSI drive, you often can identify the chip or chipset that serves as the disk controller on the drive as being exactly the same kind that would be used on an ST-506/412 or ESDI controller or as being some evolutionary customized variation thereof.

Consider some examples. An ATA (AT Attachment interface) IDE drive must fully emulate the system-level disk-controller interface introduced with the Western Digital WD1003 controller series that IBM used in the AT. These drives must act as though they have a built-in ST-506/412 or ESDI controller; in fact, they actually do. Most of these built-in controllers have more capabilities than the original WD1003 series (usually in the form of additional commands), but they must at least respond to all the original commands that were used with the WD1003.

If you follow the hard drive market, you usually will see that drive manufacturers offer most of their newer drives in both ATA-IDE and SCSI versions. In other words, if a manufacturer makes a particular 500MB IDE drive, you invariably will see that the company also make a SCSI model with the same capacity and specifications, which uses the same HDA (Head Disk Assembly) and even looks the same as the IDE version. If you study these virtually identical drives, the only major difference you will find is the additional chip on the logic board of the SCSI version, called a SCSI Bus Adapter Chip (SBIC).

Figures 14.15 and 14.16 show the logic-block diagrams of the WD-AP4200 (a 200MB ATA-IDE drive) and the WD-SP4200 (a 200MB SCSI drive), respectively. These drives use the same HDA; they differ only in their logic boards, and even the logic boards are the same except for the addition of an SBIC (SCSI Bus Adapter Chip) on the SCSI drive's logic board.

Notice that even the circuit designs of these two drives are almost identical. Both drives use an LSI (Large Scale Integrated circuit) chip called the WD42C22 Disk Controller and Buffer manager chip. In the ATA drive, this chip is connected through a DMA control chip directly to the AT bus. In the SCSI version, a WD33C93 SCSI bus interface controller chip is added to interface the disk-controller logic to the SCSI bus. In fact, the logic diagrams of these two drives differ only in the fact that the SCSI version has a complete subset of the ATA drive, with the SCSI bus interface controller logic added. This essentially is a very condensed version of the separate drive and bridge controller setups that were used in the early days of PC SCSI!

To top off this example, study the following logic diagram for the WD 1006V-MM1, which is an ST-506/412 controller (see fig. 14.17).

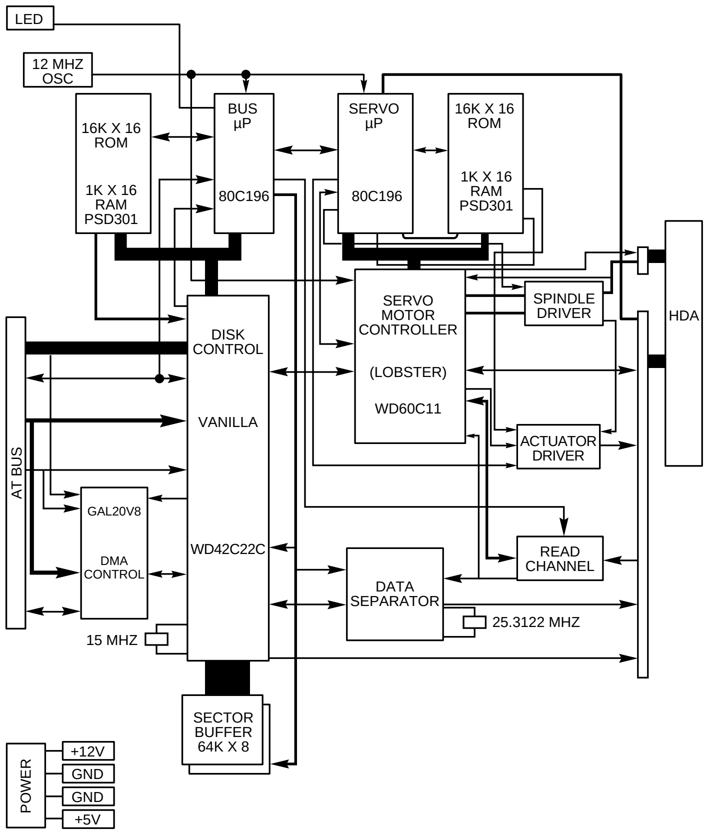


Fig. 14.15
Western Digital WD-AP4200 200MB ATA-IDE drive logic-board block diagram.

You can clearly see that the main LSI chip on board is the same WD42C22 disk controller chip used in the IDE and SCSI drives. Here is what the technical reference literature says about that chip:

> The WD42C22 integrates a high performance, low cost Winchester controller's architecture. The WD42C22 integrates the central elements of a Winchester controller subsystem such as the host interface, buffer manager, disk formatter/controller, encoder/decoder, CRC/ECC (Cyclic Redundancy Check/Error Correction Code) generator/checker, and drive interface into a single 84-pin PQFP (Plastic Quad Flat Pack) device.

The virtually identical design of ATA-IDE and SCSI drives is not unique to Western Digital. Most drive manufacturers design their ATA-IDE and SCSI drives the same way, often using these very same Western Digital chips as well as disk controller and SCSI bus interface chips from other manufacturers. You now should be able to understand that most SCSI drives simply are "regular" ATA-IDE drives with SCSI bus logic added. This fact will come up again later in this chapter, in the section called SCSI versus IDE that discusses performance and other issues differentiating these interfaces.

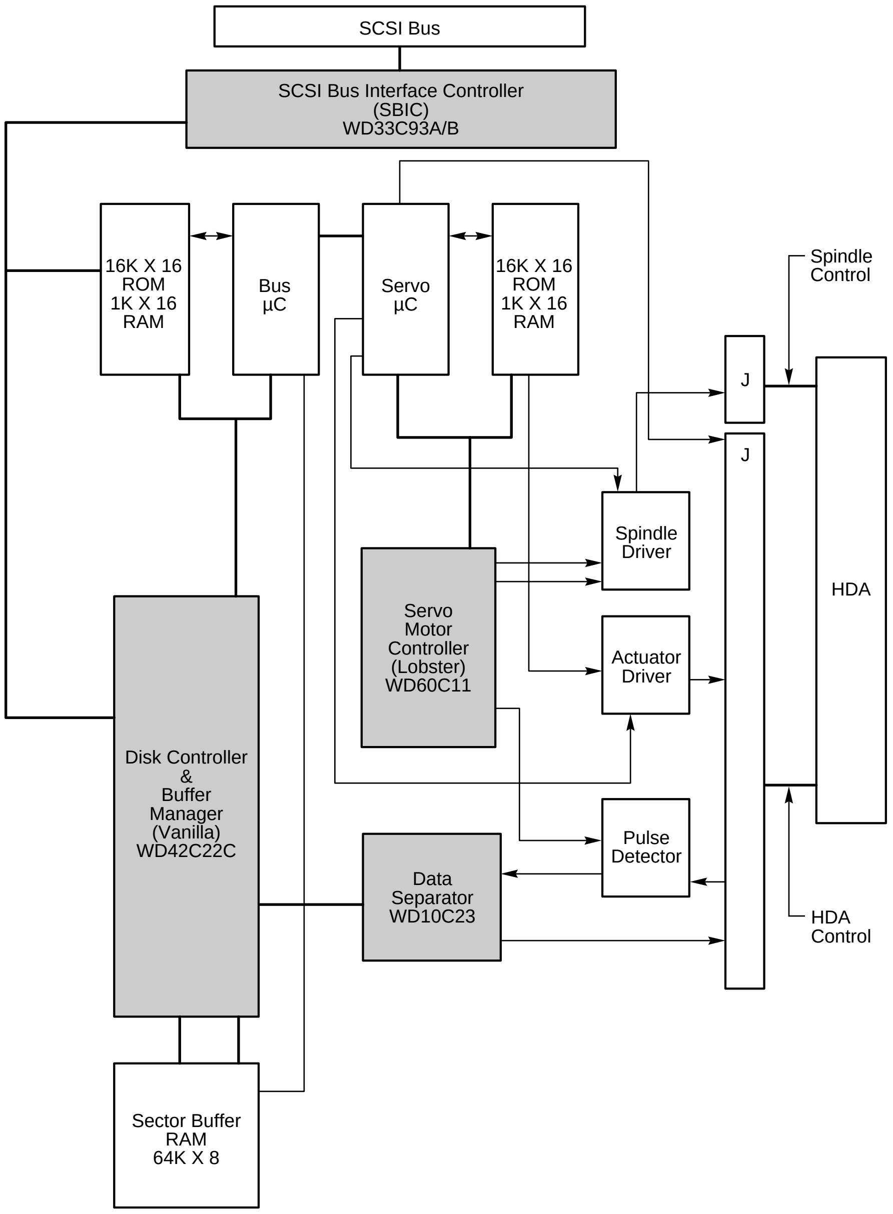


Fig. 14.16
Western Digital WD-SP4200 200MB SCSI drive logic-board block diagram.

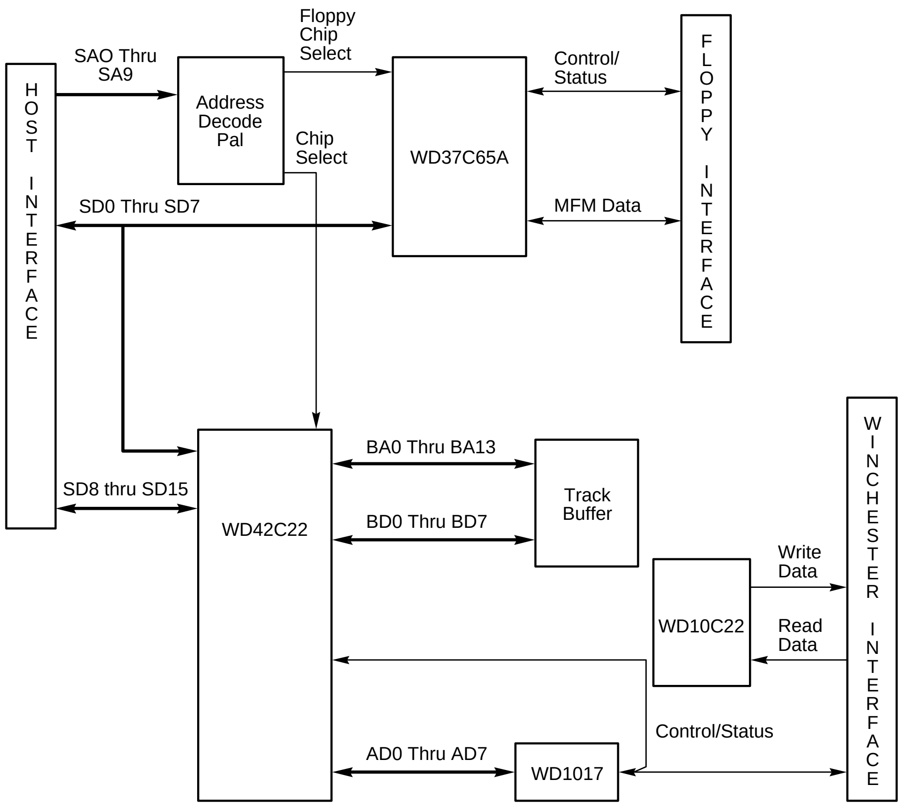


Fig. 14.17
Western Digital WD1006V-MM1 ST-506/412 Disk Controller block diagram.

For another example, I have several IBM 320MB and 400MB embedded SCSI-2 hard disks; each of these drives has on board a WD-10C00 Programmable Disk Controller in the form of a 68-pin PLCC (Plastic Leaded Chip Carrier) chip. The technical literature states, "This chip supports ST412, ESDI, SMD and Optical interfaces. It has 27-Mbit/sec maximum transfer rate and an internal, fully programmable 48- or 32-bit ECC, 16-bit CRC-CCITT or external user defined ECC polynomial, fully programmable sector sizes and 1.25 micron low power CMOS design."

In addition, these particular embedded SCSI drives include the 33C93 SCSI Bus Interface Controller chip, which also is used in the other SCSI drive that I mentioned. Again, there is a distinctly separate disk controller, and the SCSI interface is added on.

So again, most embedded SCSI drives have a built-in disk controller (usually based on previous ST-506/412 or ESDI designs) and additional logic to interface that controller to the SCSI bus (a built-in bridge controller, if you like). Now think about this from a performance standpoint. If virtually all SCSI drives really are ATA-IDE drives with a SCSI Bus Interface Controller chip added, what conclusions can you draw?

First, no drive can perform sustained data transfers faster than the data can actually be read from the disk platters. In other words, the HDA (Head Disk Assembly) limits performance to whatever it is capable of achieving. Drives can transmit data in short bursts at very high speeds, because they often have built-in cache or read-ahead buffers that store data. Many of the newer high-performance SCSI and ATA-IDE drives have 1MB or more of cache memory on board! No matter how big or intelligent the cache is, however, sustained data transfer still will be limited by the HDA.

Data from the HDA must pass through the disk controller circuits, which, as you have seen, are virtually identical between similar SCSI and ATA-IDE drives. In the ATA-IDE drive, this data then is presented directly to the system bus. In the SCSI drive, however, the data must pass through a SCSI Bus Interface adapter on the drive, travel through the SCSI bus itself, and then pass through another SCSI Bus Interface controller in the SCSI host adapter card in your system. The longer route that a SCSI transfer must take makes this type of transfer slower than the much more direct ATA-IDE transfer.

The conventional wisdom has been that SCSI always is much faster than IDE; unfortunately, this wisdom usually is wrong! This incorrect conclusion was derived by looking at the raw SCSI and ISA bus performance capabilities. An 8-bit Fast SCSI-2 bus can transfer data at 10MB (million bytes) per second, whereas the 16-bit ISA bus used directly by IDE drives can transfer data at rates ranging from 2MB to 8MB per second. Based on these raw transfer rates, SCSI seems to be faster, but the raw transfer rate of the bus is not the limiting factor. Instead, the actual HDA and disk-controller circuitry place the limits on performance. Another point to remember is that unless you are using a VL-Bus, EISA, or 32-bit MCA SCSI adapter, the SCSI data-transfer speeds will be limited by the host bus performance as well as by the drive performance.

Single-Ended or Differential SCSI. "Normal" SCSI also is called *single-ended* SCSI. For each signal that needs to be sent across the bus, a wire exists to carry it. With differential SCSI, for each signal that needs to be sent across the bus, a pair of wires exists to carry it. The first in this pair carries the same type of signal that the single-ended SCSI carries. The second in this pair, however, carries the logical inversion of the signal. The receiving device takes the difference of the pair (hence the name *differential*), which makes it less susceptible to noise and allows for greater cable length. Because of this, differential SCSI can be used with cable lengths up to 25 meters, whereas single-ended SCSI is good only for 6 meters with standard asynchronous or synchronous transfers or for only 3 meters for Fast SCSI.

You cannot mix single-ended and differential devices on a single SCSI bus; the result would be catastrophic. (That is to say that you probably will see smoke!) Notice that the cables and connectors are the same, so it's entirely possible to make this mistake. This usually is not a problem, however, because very few differential SCSI implementations exist. Especially with SCSI in the PC environment, single-ended is about all you will ever see. If, however, you come upon a peripheral that you believe might be differential, there are a few ways to tell. One way is to look for a special symbol on the unit; the industry has adopted different universal symbols for single-ended and differential SCSI. Figure 14.18 shows these symbols.

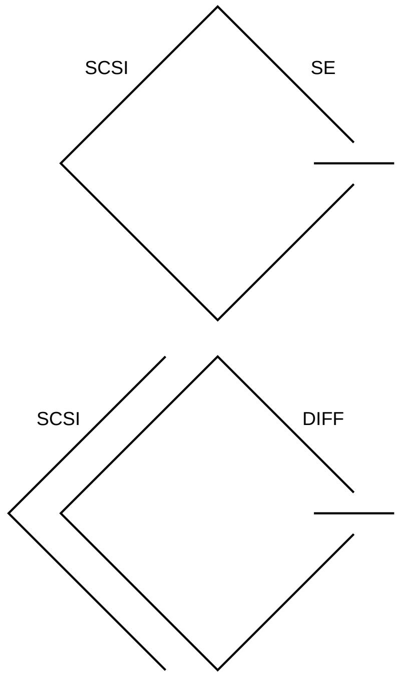


Fig. 14.18
Single-ended and differential SCSI universal symbols.

If you do not see such symbols, you can tell whether you have a differential device by using an ohm meter to check the resistance between pins 21 and 22 on the device connector. On a single-ended system, the pins should be tied together and also tied to Ground. On a differential device, the pins should be open or have significant resistance between them. Again, this generally should not be a problem, because virtually all devices used in the PC environment are single-ended.

SCSI-1 and SCSI-2. The SCSI-2 specification essentially is an improved version of SCSI-1 with some parts of the specification tightened and with several new features and options added. Normally, SCSI-1 and SCSI-2 devices are compatible, but SCSI-1 devices ignore the additional features in SCSI-2.

Some of the changes in SCSI-2 are very minor. For example, SCSI-1 allowed SCSI Bus parity to be optional, whereas parity must be implemented in SCSI-2. Another requirement is that initiator devices, such as host adapters, provide terminator power to the interface; most devices already did so.

SCSI-2 also has several optional features:

- Fast SCSI
- Wide SCSI
- Command queuing

- High-density cable connectors
- Improved Active (Alternative 2) termination

These features are not required; they are optional under the SCSI-2 specification. If you connect a standard SCSI host adapter to a Fast SCSI drive, for example, the interface will work, but only at standard SCSI speeds.

Fast SCSI. *Fast SCSI* refers to high-speed synchronous transfer capability. Fast SCSI achieves a 10MB per second transfer rate on the standard 8-bit SCSI cabling. When combined with a 16-bit or 32-bit Wide SCSI interface, this configuration results in data-transfer rates of 20 to 40MB per second.

Wide SCSI. Wide SCSI allows for parallel data transfer at bus widths of 16 and 32 bits. These wider connections require new cable designs. The standard 50-conductor 8-bit cable is called the A-cable. SCSI-2 originally defined a special 68-conductor B-cable that was supposed to be used in conjunction with the A cable for wide transfers, but the industry ignored this specification in favor of a newer 68-conductor P-cable that was introduced as part of the SCSI-3 specification. The P-cable superseded the A- and B-cable combination because the P-Cable can be used alone (without the A cable) for 16-bit Wide SCSI.

32-bit Wide SCSI has not found popularity and probably never will in the PC environment. Theoretically, 32-bit SCSI implementations require two cables: a 68-conductor P-cable and a 68-conductor Q-cable.

Termination. The single-ended SCSI bus depends on very tight termination tolerances to function reliably. Unfortunately, the original 132-ohm passive termination defined in the SCSI-1 document was not designed for use at the higher synchronous speeds now possible. These passive terminators can cause signal reflections to cause errors when transfer rates increase or when more devices are added to the bus. SCSI-2 defines an active (voltage-regulated) terminator that lowers termination impedance to 110 ohms and improves system integrity.

Command Queuing. In SCSI-1, an initiator device, such as a host adapter, was limited to sending one command per device. In SCSI-2, the host adapter can send as many as 256 commands to a single device, which will store and process those commands internally before responding on the SCSI bus. The target device even can resequence the commands to allow for the most efficient execution or performance possible. This feature is especially useful in multitasking environments, such as OS/2 and Windows NT, that can take advantage of this feature.

New Commands. SCSI-2 took the Common Command Set that was being used throughout the industry and made it an official part of the standard. The CCS was designed mainly for disk drives and did not include specific commands designed for other types of devices. In SCSI-2, many of the old commands are reworked and several new commands have been added. New command sets have been added for CD-ROMs, optical drives, scanners, communications devices, and media changers (jukeboxes).

SCSI-3. Even though the SCSI-2 specification has only recently been approved (although it has remained stable for some time), the SCSI-3 specification is already being developed. SCSI-3 will have everything that SCSI-2 has and definitely will add new commands, features, and implementations. For example, SCSI-3 will provide support for up to 32 devices on the bus instead of only 8.

One of the most exciting things about SCSI-3 is the proposed Serial SCSI, a scheme that may use only a six-conductor cable and that will be able to transfer data at up to 100MB per second! The switch to serial instead of parallel is designed to control the delay, noise, and termination problems that have plagued SCSI-2, as well as to simplify the cable connection. Serial SCSI will be capable of transferring more data over 6 wires than 32-bit Fast Wide SCSI-2 can over 128 wires! The intention is that Serial SCSI be implemented on the motherboard of future systems, giving them incredible expansion and performance capabilities.

Although Serial SCSI may not make the older host adapters and cables obsolete overnight, it does make future cabling possibilities even more of a puzzle. Serial SCSI offers the possibility of longer cable lengths; less electromagnetic interference; and easier connections on laptops, notebooks, and docking stations. Expect SCSI-3 to offer almost pain-free installations with automatic plug-and-play SCSI ID setup and termination schemes.

In any practical sense, SCSI-3 is still some ways away from being approved. Because the standard exists in draft documents before being officially approved, if the portions of the standard become stable, we may very well see products claiming SCSI-3 compatibility well before the standard truly exists. Since SCSI-3 actually incorporates all of what is in SCSI-2, technically anybody can call any SCSI-1 or SCSI-2 device a SCSI-3 device as well. Beware of product hype along these lines. Some of the new SCSI-3 features will likely be incompatible with previous SCSI implementations, and may take a while to appear on the market.

SCSI Cables and Connectors. The SCSI standards are very specific when it comes to cables and connectors. The most common connectors specified in this standard are the 50-position unshielded pin header connector for internal SCSI connections and the 50-position shielded Centronics latch-style connectors for external connections. The shielded Centronics style connector also is called Alternative 2 in the official specification. Passive or Active termination (Active is preferred) is specified for both single-ended and differential buses. The 50-conductor bus configuration is defined in the SCSI-2 standard as the A-cabled.

The SCSI-2 revision added a high-density, 50-position, D-shell connector option for the A-cable connectors. This connector now is called Alternative 1. The Alternative 2 Centronics latch-style connector remains unchanged from SCSI-1. A 68-conductor B-cable specification was added to the SCSI-2 standard to provide for 16- and 32-bit data transfers; the connector, however, had to be used in parallel with an A-cable. The industry did not widely accept the B-cable option, which has been dropped from the SCSI-3 standard.

To replace the ill-fated B-cable, a new 68-conductor P-cable was developed as part of the SCSI-3 specification. Shielded and unshielded high-density D-shell connectors are specified for both the A-cable and P-cable. The shielded high-density connectors use a squeeze-to-release latch rather than the wire latch used on the Centronics-style connectors. Active termination for single-ended buses is specified, providing a high level of signal integrity.

SCSI Cable and Connector Pinouts. The following section details the pinouts of the various SCSI cables and connectors. There are two electrically different version of SCSI, Single Ended and Differential. These two versions are electrically incompatible, and must not be interconnected or damage will result. Fortunately, there are very few Differential SCSI applications available in the PC industry, so you will rarely (if ever) encounter it. Within each electrical type (Single Ended or Differential), there are basically three SCSI cable types:

- A-Cable (Standard SCSI)
- P-Cable (16- and 32-bit Wide SCSI)
- Q-Cable (32-bit Wide SCSI)

The A-Cable is used in most SCSI-1 and SCSI-2 installations, and is the most common cable you will encounter. SCSI-2 Wide (16-bit) applications use a P-Cable instead, which completely replaces the A-Cable. You can intermix standard and wide SCSI devices on a single SCSI bus by interconnecting A- and P-Cables with special adapters. 32-bit wide SCSI-3 applications use both the P- and Q-Cables in parallel to each 32-bit device. Today there are virtually no PC applications for 32-bit Wide SCSI-3, and because of the two cable requirement, it is not likely to catch on.

The A-Cables can have Pin Header (Internal) type connectors or External Shielded connectors, each with a different pinout. The P- and Q-Cables feature the same connector pinout on either Internal or External cable connections.

The following tables show all of the possible interface, cable, and connector pinout specifications. A hyphen preceding a signal name indicates the signal is Active Low. The RESERVED lines have continuity from one end of the SCSI bus to the other. In an A-Cable bus, the RESERVED lines should be left open in SCSI devices (but may be connected to ground), and are connected to ground in the bus terminator assemblies. In the P- and Q-Cables, the RESERVED lines are left open in SCSI devices as well as in the bus terminator assemblies.

Single Ended SCSI Cables and Connectors. The single ended electrical interface is the most popular type for PC systems. The following tables show all of the possible single ended cable and connector pinouts. The A-Cable is available in both internal unshielded as well as external shielded configurations. Both are shown in tables 14.22 and 14.23.

Table 14.22 A-Cable (Single Ended) Internal Unshielded Header Connector Pinout

Signal Name	Pin	Pin	Signal Name
GROUND	1	2	-DB(0)
GROUND	3	4	-DB(1)
GROUND	5	6	-DB(2)
GROUND	7	8	-DB(3)
GROUND	9	10	-DB(4)
GROUND	11	12	-DB(5)
GROUND	13	14	-DB(6)
GROUND	15	16	-DB(7)
GROUND	17	18	-DB(Parity)
GROUND	19	20	GROUND
GROUND	21	22	GROUND
RESERVED	23	24	RESERVED
Open	25	26	TERMPWR
RESERVED	27	28	RESERVED
GROUND	29	30	GROUND
GROUND	31	32	-ATN
GROUND	33	34	GROUND
GROUND	35	36	-BSY
GROUND	37	38	-ACK
GROUND	39	40	-RST
GROUND	41	42	-MSG
GROUND	43	44	-SEL
GROUND	45	46	-C/D
GROUND	47	48	-REQ
GROUND	49	50	-I/O

Table 14.23 A-Cable (Single Ended) External Shielded Connector Pinout

Signal Name	Pin	Pin	Signal Name
GROUND	1	26	-DB(0)
GROUND	2	27	-DB(1)
GROUND	3	28	-DB(2)
GROUND	4	29	-DB(3)
GROUND	5	30	-DB(4)
GROUND	6	31	-DB(5)
GROUND	7	32	-DB(6)
GROUND	8	33	-DB(7)

(continues)

Table 14.23 Continued

Signal Name	Pin	Pin	Signal Name
GROUND	9	34	-DB(Parity)
GROUND	10	35	GROUND
GROUND	11	36	GROUND
RESERVED	12	37	RESERVED
Open	13	38	TERMPWR
RESERVED	14	39	RESERVED
GROUND	15	40	GROUND
GROUND	16	41	-ATN
GROUND	17	42	GROUND
GROUND	18	43	-BSY
GROUND	19	44	-ACK
GROUND	20	45	-RST
GROUND	21	46	-MSG
GROUND	22	47	-SEL
GROUND	23	48	-C/D
GROUND	24	49	-REQ
GROUND	25	50	-I/O

IBM has standardized on the SCSI interface for virtually all PS/2 systems introduced since 1990. These systems use a Micro Channel SCSI adapter or have the SCSI Host Adapter built into the motherboard. In either case, IBM's SCSI interface uses a special 60-pin mini-Centronics type external shielded connector that is unique in the industry. A special IBM cable is required to adapt this connector to the standard 50-pin Centronics-style connector used on most external SCSI devices. The pinout of the IBM 60-pin mini-Centronics-style External Shielded connector is shown in table 14.24. In the following table, notice that although the pin arrangement is unique, the pin number to signal designations correspond with the standard unshielded internal pin header type of SCSI connector.

Table 14.24 IBM PS/2 SCSI External Shielded 60-Pin Connector Pinout

Signal Name	Pin	Pin	Signal Name
GROUND	1	60	Not Connected
-DB(0)	2	59	Not Connected
GROUND	3	58	Not Connected
-DB(1)	4	57	Not Connected
GROUND	5	56	Not Connected
-DB(2)	6	55	Not Connected
GROUND	7	54	Not Connected
-DB(3)	8	53	Not Connected

Signal Name	Pin	Pin	Signal Name
GROUND	9	52	Not Connected
-DB(4)	10	51	GROUND
GROUND	11	50	-I/O
-DB(5)	12	49	GROUND
GROUND	13	48	-REQ
-DB(6)	14	47	GROUND
GROUND	15	46	-C/D
-DB(7)	16	45	GROUND
GROUND	17	44	-SEL
-DB(Parity)	18	43	GROUND
GROUND	19	42	-MSG
GROUND	20	41	GROUND
GROUND	21	40	-RST
GROUND	22	39	GROUND
RESERVED	23	38	-ACK
RESERVED	24	37	GROUND
Open	25	36	-BSY
TERMPWR	26	35	GROUND
RESERVED	27	34	GROUND
RESERVED	28	33	GROUND
GROUND	29	32	-ATN
GROUND	30	31	GROUND

The P-Cable (Single-Ended) and connectors are used in 16-bit wide SCSI-2 applications (see table 14.25 for the pinout). This cable is used also in combination with the Q-cable for 32-bit SCSI applications.

Table 14.25 P-Cable (Single Ended) Internal or External Shielded Connector Pinout

Signal Name	Pin	Pin	Signal Name
GROUND	1	35	-DB(12)
GROUND	2	36	-DB(13)
GROUND	3	37	-DB(14)
GROUND	4	38	-DB(15)
GROUND	5	39	-DB(Parity 1)
GROUND	6	40	-DB(0)
GROUND	7	41	-DB(1)
GROUND	8	42	-DB(2)
GROUND	9	43	-DB(3)

(continues)

Table 14.25 Continued

Signal Name	Pin	Pin	Signal Name
GROUND	10	44	-DB(4)
GROUND	11	45	-DB(5)
GROUND	12	46	-DB(6)
GROUND	13	47	-DB(7)
GROUND	14	48	-DB(Parity 0)
GROUND	15	49	GROUND
GROUND	16	50	GROUND
TERMPWR	17	51	TERMPWR
TERMPWR	18	52	TERMPWR
RESERVED	19	53	RESERVED
GROUND	20	54	GROUND
GROUND	21	55	-ATN
GROUND	22	56	GROUND
GROUND	23	57	-BSY
GROUND	24	58	-ACK
GROUND	25	59	-RST
GROUND	26	60	-MSG
GROUND	27	61	-SEL
GROUND	28	62	-C/D
GROUND	29	63	-REQ
GROUND	30	64	-I/O
GROUND	31	65	-DB(8)
GROUND	32	66	-DB(9)
GROUND	33	67	-DB(10)
GROUND	34	68	-DB(11)

The Q-Cable (Single Ended) and connector is used only on 32-bit SCSI implementations, which also require a P-Cable as well (see table 14.26 for the pinout). 32-bit SCSI applications are rare to virtually nonexistant.

Table 14.26 Q-Cable (Single Ended) Internal or External Shielded Connector Pinout

Signal Name	Pin	Pin	Signal Name
GROUND	1	35	-DB(28)
GROUND	2	36	-DB(29)
GROUND	3	37	-DB(30)

Signal Name	Pin	Pin	Signal Name
GROUND	4	38	-DB(31)
GROUND	5	39	-DB(Parity 3)
GROUND	6	40	-DB(16)
GROUND	7	41	-DB(17)
GROUND	8	42	-DB(18)
GROUND	9	43	-DB(19)
GROUND	10	44	-DB(20)
GROUND	11	45	-DB(21)
GROUND	12	46	-DB(22)
GROUND	13	47	-DB(23)
GROUND	14	48	-DB(Parity 2)
GROUND	15	49	GROUND
GROUND	16	50	GROUND
TERMPWRQ	17	51	TERMPWRQ
TERMPWRQ	18	52	TERMPWRQ
RESERVED	19	53	RESERVED
GROUND	20	54	GROUND
GROUND	21	55	TERMINATED
GROUND	22	56	GROUND
GROUND	23	57	TERMINATED
GROUND	24	58	-ACKQ
GROUND	25	59	TERMINATED
GROUND	26	60	TERMINATED
GROUND	27	61	TERMINATED
GROUND	28	62	TERMINATED
GROUND	29	63	-REQQ
GROUND	30	64	TERMINATED
GROUND	31	65	-DB(24)
GROUND	32	66	-DB(25)
GROUND	33	67	-DB(26)
GROUND	34	68	-DB(27)

Differential SCSI Signals. Differential SCSI is not normally used in a PC environment, but is very popular with minicomputer installations due to the very long bus lengths that are allowed. Although not popular in PC systems, the interface connector specifications are shown here for reference.

The A-Cable (Differential) Connector is available in both an internal unshielded form as well as an external shielded form. Tables 14.27 and 14.28 show the pinouts for both.

Table 14.27 A-Cable (Differential) Internal Unshielded Header Connector Pinout

Signal Name	Pin	Pin	Signal Name
GROUND	1	2	GROUND
+DB(0)	3	4	-DB(0)
+DB(1)	5	6	-DB(1)
+DB(2)	7	8	-DB(2)
+DB(3)	9	10	-DB(3)
+DB(4)	11	12	-DB(4)
+DB(5)	13	14	-DB(5)
+DB(6)	15	16	-DB(6)
+DB(7)	17	18	-DB(7)
+DB(Parity)	19	20	-DP(Parity)
DIFFSENS	21	22	GROUND
RESERVED	23	24	RESERVED
TERMPWR	25	26	TERMPWR
RESERVED	27	28	RESERVED
+ATN	29	30	-ATN
GROUND	31	32	GROUND
+BSY	33	34	-BSY
+ACK	35	36	-ACK
+RST	37	38	-RST
+MSG	39	40	-MSG
+SEL	41	42	-SEL
+C/D	43	44	-C/D
+REQ	45	46	-REQ
+I/O	47	48	-I/O
GROUND	49	50	GROUND

Table 14.28 A-Cable (Differential) External Shielded Header Connector Pinout

Signal Name	Pin	Pin	Signal Name
GROUND	1	26	GROUND
+DB(0)	2	27	-DB(0)
+DB(1)	3	28	-DB(1)
+DB(2)	4	29	-DB(2)
+DB(3)	5	30	-DB(3)
+DB(4)	6	31	-DB(4)
+DB(5)	7	32	-DB(5)
+DB(6)	8	33	-DB(6)

Signal Name	Pin	Pin	Signal Name
+DB(7)	9	34	-DB(7)
+DB(Parity)	10	35	-DP(Parity)
DIFFSENS	11	36	GROUND
RESERVED	12	37	RESERVED
TERMPWR	13	38	TERMPWR
RESERVED	14	39	RESERVED
+ATN	15	40	-ATN
GROUND	16	41	GROUND
+BSY	17	42	-BSY
+ACK	18	43	-ACK
+RST	18	44	-RST
+MSG	20	45	-MSG
+SEL	21	46	-SEL
+C/D	22	47	-C/D
+REQ	23	48	-REQ
+I/O	24	49	-I/O
GROUND	25	50	GROUND

The P-Cable (Differential) and Connector is used for 16-bit and 32-bit wide SCSI connections. The pinout is shown in Table 14.29.

Table 14.29 P-Cable (Differential) Internal or External Shielded Connector Pinout

Signal Name	Pin	Pin	Signal Name
+DB(12)	1	35	-DB(12)
+DB(13)	2	36	-DB(13)
+DB(14)	3	37	-DB(14)
+DB(15)	4	38	-DB(15)
+DB(Parity 1)	5	39	-DB(Parity 1)
GROUND	6	40	GROUND
+DB(0)	7	41	-DB(0)
+DB(1)	8	42	-DB(1)
+DB(2)	9	43	-DB(2)
+DB(3)	10	44	-DP(3)
+DB(4)	11	45	-DB(4)
+DB(5)	12	46	-DB(5)
+DB(6)	13	47	-DB(6)
+DB(7)	14	48	-DB(7)
+DB(Parity 0)	15	49	-DB(Parity 0)

(continues)

Table 14.29 Continued

Signal Name	Pin	Pin	Signal Name
DIFFSENS	16	50	GROUND
TERMPWR	17	51	TERMPWR
TERMPWR	18	52	TERMPWR
RESERVED	19	53	RESERVED
+ATN	20	54	-ATN
GROUND	21	55	GROUND
+BSY	22	56	-BSY
+ACK	23	57	-ACK
+RST	24	58	-RST
+MSG	25	59	-MSG
+SEL	26	60	-SEL
+C/D	27	61	-C/D
+REQ	28	62	-REQ
+I/O	29	63	-I/O
GROUND	30	64	GROUND
+DB(8)	31	65	-DB(8)
+DB(9)	32	66	-DB(9)
+DB(10)	33	67	-DB(10)
+DB(11)	34	68	-DB(11)

The Q Cable (Differential) and connector is used only with 32-bit wide SCSI implementations, and in that case would also require a 16-bit wide P-cable. Table 14.30 shows the Q-Cable pinout.

Table 14.30 Q-Cable (Differential) Internal or External Shielded Connector Pinout

Signal Name	Pin	Pin	Signal Name
+DB(28)	1	35	-DB(28)
+DB(29)	2	36	-DB(29)
+DB(30)	3	37	-DB(30)
+DB(31)	4	38	-DB(31)
+DB(Parity 3)	5	39	-DB(Parity 3)
GROUND	6	40	GROUND
+DB(16)	7	41	-DB(16)
+DB(17)	8	42	-DB(17)
+DB(18)	9	43	-DB(18)
+DB(19)	10	44	-DP(19)
+DB(20)	11	45	-DB(20)

Signal Name	Pin	Pin	Signal Name
+DB(21)	12	46	-DB(21)
+DB(22)	13	47	-DB(22)
+DB(23)	14	48	-DB(23)
+DB(Parity 2)	15	49	-DB(Parity 2)
DIFFSENS	16	50	GROUND
TERMPWRQ	17	51	TERMPWRQ
TERMPWRQ	18	52	TERMPWRQ
RESERVED	19	53	RESERVED
TERMINATED	20	54	TERMINATED
GROUND	21	55	GROUND
TERMINATED	22	56	TERMINATED
+ACKQ	23	57	-ACKQ
TERMINATED	24	58	TERMINATED
TERMINATED	25	59	TERMINATED
TERMINATED	26	60	TERMINATED
TERMINATED	27	61	TERMINATED
+REQQ	28	62	-REQQ
TERMINATED	29	63	TERMINATED
GROUND	30	64	GROUND
+DB(24)	31	65	-DB(24)
+DB(25)	32	66	-DB(25)
+DB(26)	33	67	-DB(26)
+DB(27)	34	68	-DB(27)

Termination. All busses need to be electrically terminated at each end, and the SCSI bus is no exception. Improper termination still is one of the most common problems in SCSI installations. Three types of terminators typically are available for the SCSI bus:

- Passive
- Active (also called Alternative 2)
- Forced Perfect Termination (FPT)

Typical passive terminators (a network of resistors) allow signal fluctuations in relation to the terminator power signal on the bus. Usually, passive terminating resistors suffice over short distances, such as 2 or 3 feet, but for longer distances, active termination is a real advantage. Active termination is required with Fast SCSI.

An active terminator actually has one or more voltage regulators to produce the termination voltage, rather than resistor voltage dividers. This arrangement helps ensure that the SCSI signals always are terminated to the correct voltage level. The SCSI-2 specification recommends active termination on both ends of the bus and requires active termination whenever Fast or Wide SCSI devices are used.

A variation on active termination is available: Forced Perfect Termination. Forced Perfect Termination is an even better form of active termination, in which diode clamps are added to eliminate signal overshoot and undershoot. The trick is that instead of clamping to +5 and Ground, these terminators clamp to the output of two regulated voltages. This arrangement enables the clamping diodes to eliminate signal overshoot and undershoot, especially at higher signaling speeds and over longer distances.

Several companies make high-quality terminators for the SCSI bus, including Aeronics and the Data Mate division of Methode. Both of these companies make a variety of terminators, but Aeronics is well noted for some unique FPT versions that are especially suited to problem configurations that require longer cable runs or higher signal integrity. One of the best investments that you can make in any SCSI installation is in high-quality cables and terminators.

SCSI Drive Configuration. SCSI drives are not too difficult to configure, especially compared with IDE drives. The SCSI standard controls the way that the drives must be set up. You need to set two or three items when you configure a SCSI drive:

- SCSI ID setting (0 through 7)
- Terminating resistors

SCSI ID Setting. The SCSI ID setting is very simple. Up to eight SCSI devices can be used on a single SCSI bus, and each device must have a unique SCSI ID address. The host adapter takes one address, and up to seven SCSI peripherals take the others. Most SCSI host adapters are factory-set to ID 7, which is the highest-priority ID. All other devices must have unique IDs that do not conflict with one another. Some host adapters boot only from a hard disk set to a specific ID. In my system, for example, the IBM SCSI host adapter requires the boot drive to be set to ID 6. Newer IBM host adapters and systems enable you to boot from a hard disk at any SCSI ID. Older Adaptec host adapters required the boot hard disk to be ID 0; newer ones can boot from any ID.

Setting the ID usually involves changing jumpers on the drive itself. If the drive is installed in an external chassis, the chassis may have an ID selector switch that is accessible at the rear. This selector makes ID selection a simple matter of pressing a button or rotating a wheel until the desired ID number appears. If no external selector is present, you must open the external device chassis and set the ID via the jumpers on the drive.

Three jumpers are required to set the SCSI ID; the particular ID selected actually is derived from the binary representation of the jumpers themselves. For example, setting all three ID jumpers off results in a binary number of 000b, which translates to an ID of 0. A binary setting of 001b equals ID 1, 010b equals 2, 011b equals 3, and so on. Notice that as I list these values, I append a lowercase *b* to indicate binary numbers.

Unfortunately, the jumpers can appear either forward or backward on the drive, depending on how the manufacturer set them up. To keep things simple, I have recorded all the different ID jumper settings in Tables 14.31 and 14.32. One table is for drives that order the jumpers, with the Most Significant Bit (MSB) to the left; the other is for drives with the jumpers ordered so that the MSB is to the right.

Table 14.31 SCSI ID Jumper Settings with the Most Significant Bit to the Left

SCSI ID	Jumper Settings		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

1 = Jumper On, 0 = Jumper Off

Table 14.32 SCSI ID Jumper Settings with the Most Significant Bit to the Right

SCSI ID	Jumper Settings		
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1
5	1	0	1
6	0	1	1
7	1	1	1

1 = Jumper On, 0 = Jumper Off

Termination. SCSI termination is very simple. Termination is required at both ends of the bus; there are no exceptions. If the host adapter is at one end of the bus, it must have termination enabled. If the host adapter is in the middle of the bus, and if both internal and external bus links are present, the host adapter must have its termination disabled, and the devices at each end of the bus must have terminators installed. Several types of terminators are available, differing both in quality and in appearance. Active terminators are the minimum recommended, and Forced Perfect Terminators (FPT) are considered the best available. For more information on the different types, see the previous section on Terminators in this chapter.

The rules are simple: use the best terminators possible, and make sure that only the ends of the SCSI bus are terminated. The majority of problems that I see with SCSI installations are the result of improper termination. Some devices have built-in termination resistors that are enabled or disabled through a jumper or by being physically removed. Other devices do not have built-in terminating resistors; these devices instead rely on external terminator modules for termination.

When installing an external SCSI device, you usually will find the device in a storage enclosure with both input and output SCSI connectors, so that you can use the device in a daisy chain. If the enclosure is at the end of the SCSI bus, an external terminator module most likely will have to be plugged into the second (outgoing) SCSI port to provide proper termination at that end of the bus (see fig. 14.19).

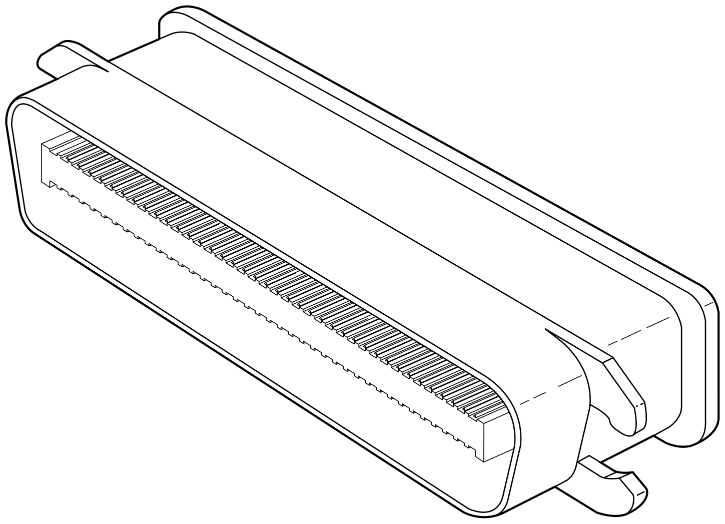


Fig. 14.19
External SCSI device terminator.

External terminator modules are available in a variety of connector configurations, including pass-through designs, which are needed if only one port is available. Pass-through terminators also are commonly used in internal installations in which the device does not have built-in terminating resistors. Many hard drives use pass-through terminators for internal installations to save space on the logic-board assembly (see fig. 14.20).

The pass-through models are required when a device is at the end of the bus and only one SCSI connector is available.

Remember to stick with high-quality active or Forced Perfect terminators at each end of the bus, and you will eliminate most common termination problems.

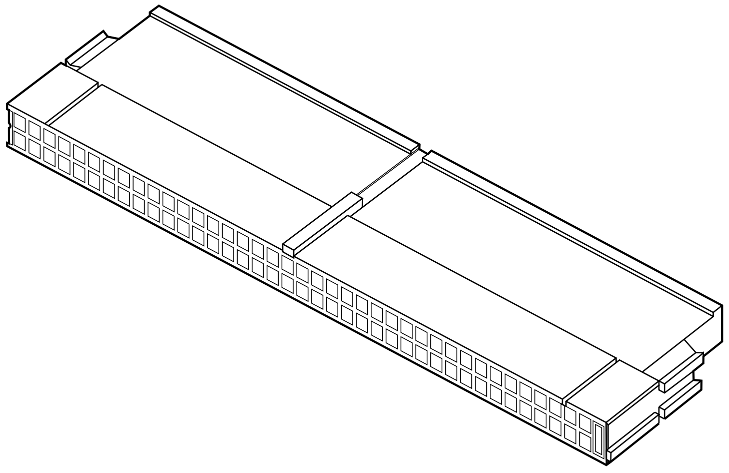


Fig. 14.20
Internal pin-header connector pass-through SCSI terminator.

Other Settings. Other configuration items on a SCSI drive can be set via jumpers. The following are several of the most common additional settings that you will find:

- Start on command (delayed start)
- SCSI parity
- Terminator power
- Synchronous negotiation

These configuration items are described in the following sections.

Start on Command (Delayed Start). If you have multiple drives installed in a system, it is wise to set them up so that all the drives do not start to spin immediately when the system is powered on. A hard disk drive can consume three or four times more power during the first few seconds after power-on then during normal operation. The motor requires this additional power to get the platters spinning quickly. If several drives are drawing all this power at the same time, the power supply may be overloaded, which can cause the system to hang or to have intermittent startup problems.

Nearly all SCSI drives provide a way to delay drive spinning so that this problem does not occur. When most SCSI host adapters initialize the SCSI bus, they send out a command called Start Unit to each of the ID addresses in succession. By setting a jumper on the hard disk, you can prevent the disk from spinning until it receives the Start Unit command from the host adapter. Because the host adapter sends this command to all the ID addresses in succession, from the highest-priority address (ID 7) to the lowest (ID 0), the higher-priority drives can be made to start first, with each lower-priority drive spinning up sequentially. Because some host adapters do not send the Start Unit command, some drives may simply delay spinup for a fixed number of seconds rather than wait for a command that never will arrive.

If drives are installed in external chassis with separate power supplies, you need not implement the delayed-start function. This function is best applied to internal drives that must be run from the same power supply that runs the system. For internal installations, I recommend setting Start on Command (delayed start) even if you have only one SCSI drive; this setting will ease the load on the power supply by spinning the drive up after the rest of the system has full power. This method is especially good for portable systems and other systems in which the power supply is limited.

SCSI Parity. SCSI parity is a limited form of error checking that helps ensure that all data transfers are reliable. Virtually all host adapters support SCSI parity checking, so this option should be enabled on every device. The only reason why it exists as an option is that some older host adapters do not work with SCSI parity, so the parity must be turned off.

Terminator Power. The terminators at each end of the SCSI bus require power from at least one device on the bus. In most cases, the host adapter supplies this terminator power; in some cases, however, it does not. For example, parallel port SCSI host adapters typically do not supply terminator power. It is not a problem if more than one device supplies terminator power, since each source is diode protected. For simplicity's sake, many will configure all devices to supply terminator power. If no device supplies terminator power, the bus will not be terminated correctly and will not function properly.

SCSI Synchronous Negotiation. The SCSI bus can run in two modes: asynchronous (the default) and synchronous. The bus actually switches modes during transfers through a protocol called *synchronous negotiation*. Before data is transferred across the SCSI bus, the sending device (called the *initiator*) and the receiving device (called the *target*) negotiate how the transfer will take place. If both devices support synchronous transfers, they will discover this fact through the negotiation, and the transfer will take place at the faster synchronous rate.

Unfortunately, some older devices do not respond to a request for synchronous transfer and actually can be disabled when such a request is made. For this reason, both host adapters and devices that support synchronous negotiation often have a jumper that can be used to disable this negotiation so that it can work with older devices. By default, all devices today should support synchronous negotiation, and this function should be enabled.

SCSI Drivers. Each SCSI peripheral that you add to your SCSI bus (other than hard disk drives) requires an external driver to make the device work. Hard disks are the exception; driver support for them normally is provided as part of the SCSI host adapter BIOS. These external drivers are specific not only to a particular device, but also to the host adapter.

Recently, two types of standard host adapter interface drivers have become popular, greatly reducing this problem. By having a standard host adapter driver to write to, peripheral makers can more quickly create new drivers that support their devices and then talk to the universal host adapter driver. This arrangement eliminates dependence on one particular type of host adapter. These primary or universal drivers link the host adapter and the operating system.

The Advanced SCSI Programming Interface (ASPI) currently is the most popular universal driver, with most peripheral makers writing their drivers to talk to ASPI. The *A* in ASPI used to stand for Adaptec, the company that introduced it, but other SCSI device vendors have licensed the right to use ASPI with their products. DOS does not support ASPI directly, but it does when the ASPI driver is loaded. OS/2 2.1 and later versions provide automatic ASPI support for several SCSI host adapters.

Future Domain and NCR have created another interface driver called the Common Access Method (CAM). CAM is an ANSI-approved protocol that enables a single driver to control several host adapters. In addition to ASPI, OS/2 2.1 and later versions currently offer support for CAM. Future Domain also provides a CAM-to-ASPI converter in the utilities that go with its host adapters.

SCSI Configuration Tips. When you are installing a chain of devices on a single SCSI bus, the installation can get complicated very quickly. Here are some tips for getting your setup to function quickly and efficiently.

- *Start by adding one device at a time.* Rather than plug numerous peripherals into a single SCSI card and then try to configure them at the same time, start by installing the host adapter and a single hard disk. Then you can continue installing devices one at a time, checking to make sure that everything works before moving on.
- *Keep good documentation.* When you add a SCSI peripheral, write down the SCSI ID address as well as any other switch and jumper settings, such as SCSI Parity, Terminator Power, and Remote Start. For the host adapter, record the BIOS addresses, Interrupt, DMA channel, and I/O Port addresses used by the adapter, as well as any other jumper or configuration settings (such as termination) that might be important to know later.
- *Use proper termination.* Each end of the bus must be terminated, preferably with active or Forced Perfect (FPT) terminators. If you are using any Fast SCSI-2 device, you must use active terminators rather than the cheaper passive types. Even with standard (slow) SCSI devices, active termination is highly recommended. If you have only internal or external devices on the bus, the host adapter and last device on the chain should be terminated. If you have external and internal devices on the chain, you generally will terminate the first and last of these devices but not the SCSI host adapter itself (which is in the middle of the bus).
- *Use high-quality shielded SCSI cables.* Make sure that your cable connectors match your devices. Use high-quality shielded cables, and observe the SCSI bus-length limitations. Use cables designed for SCSI use, and if possible, stick to the same brand of cable throughout a single SCSI bus. Different brands of cables have different impedance values; this situation sometimes causes problems, especially in long or high-speed SCSI implementations.

Following these simple tips will help minimize problems and leave you with a trouble-free SCSI installation.

IDE versus SCSI

When you compare the performance and capabilities of IDE (Integrated Drive Electronics) and SCSI (Small Computer System Interface) interfaced drives, you need to consider several factors. These two types of drives are the most popular drives used in PC systems today, and a single manufacturer may make identical drives in both interfaces. Deciding which drive type is best for your system is a difficult decision that depends on many factors.

In most cases, you will find that an IDE drive outperforms an equivalent SCSI drive at a given task or benchmark and that IDE drives usually cost less than SCSI drives, thus offering better value. In some cases, however, SCSI drives have significant performance and value advantages over IDE drives.

Performance. ATA (AT Attachment) IDE drives currently are used in most PC configurations on the market today, because the cost of an IDE-drive implementation is low and the performance capabilities are high. In comparing any given IDE and SCSI drive for performance, you have to look at the capabilities of the HDAs (Head Disk Assemblies) that are involved.

To minimize the variables in this type of comparison, it is easiest to compare IDE and SCSI drives from the same manufacturer that also use the identical HDA. You will find that in most cases, a drive manufacturer makes a given drive available in both IDE and SCSI forms. For example, Seagate makes the ST-3600A (ATA-IDE) and ST-3600N (Fast SCSI-2) drives, both of which use identical HDAs and which differ only in the logic board. The IDE version has a logic board with a built-in disk controller and a direct AT bus interface. The SCSI version has the same built-in disk controller and bus interface circuits, and also a SCSI Bus Interface Controller (SBIC) chip. The SBIC chip is a SCSI adapter that places the drive on the SCSI bus. What you will find, in essence, is that virtually all SCSI drives actually are IDE drives with the SBIC chip added.

The HDAs in these example drives are capable of transferring data at a sustained rate of 2.38MB to 4MB per second. Because the SCSI version always has the additional overhead of the SCSI bus to go through, in almost all cases the directly attached IDE version performs faster.

SCSI versus IDE: Advantages and Limitations. IDE drives have much less command overhead for a given sector transfer than SCSI drives do. In addition to the drive-to-controller command overhead that both IDE and SCSI must perform, a SCSI transfer involves negotiating for the SCSI bus; selecting the target drive; requesting data; terminating the transfer over the bus; and finally converting the logical data addresses to the required cylinder, head, and sector addresses.

This arrangement gives IDE an advantage in sequential transfers handled by a single-tasking operating system. In a multitasking system that can take advantage of the extra intelligence of the SCSI bus, SCSI can have the performance advantage.

SCSI drives offer significant architectural advantages over IDE and other drives. Because each SCSI drive has its own embedded disk controller that can function independently