

# Introduction To Hard Disk Drives

Mark E. Donaldson

## HARD DISK SPEED

The factors that effect the speed of a hard disk drive and the measurements used to describe hard-disk performance are:

- Rotation speed
- Number of sectors per track
- Access Time
- Seek time / Head switch time / cylinder switch time
- Rotational latency
- Command Overhead
- Data access time
- Cache on the HD
- How data is organized on the disk
- Transfer rates
- Interface (EIDE or SCSI)
- Sectors, Heads, & Cylinders

Hard-disk manufacturers measure speed in terms of access time, seek times latency, and transfer rate. These measurements, or benchmarks, often appear in hard-disk advertisements, in comparisons, and on specification sheets.

Here are some basics:

On a harddisk, data is stored in the magnetic coating of the disk. The so called head, held by an actor arm, is used to write and read data. This disk rotates with a constant turn time, measured in revolutions per minute (rpm). Data is organized on a disk in cylinders, tracks and sectors. Cylinders are concentric tracks on the surface of the disk. A track is divided into sectors. A harddisk has a head on each side of a disk. Nowadays, the actuator arm is moved by a servo-motor (not a step-motor which needs more time while swinging in after moving over the desired track). All harddisks have reserved sectors, which are used automatically by the drive logic if there is a defect in the media.

## Rotational Speed

Typical harddisks have a rotation speed from 4,500 to 7,200 rpm, and a 10,000 rpm drive just hit the market. The faster the rotation, the higher the transfer rate, but also the louder and hotter the HD. You may need to cool a 7200 rpm disk with an extra fan, or its life would be much shorter. Modern HDs read all sectors of a track in one turn (Interleave 1:1). The rotation speed is constant.

## Sectors Per Track

Modern harddisks use different track sizes. The outer parts of a disk have more space for sectors than the inner parts (zoned recording). Usually, HDs begin to write from the outside to the inside of a disk. Hence, data written or read at the beginning of a HD is accessed and transferred faster rate.

## Access time

Access time, a term frequently used in discussions of performance, is the interval of time between the moment a drive receives a request for data, and the moment the drive begins delivering the data. The access time for a hard disk is a combination of three factors: seek time, latency, and command overhead. Data access time is the combination of seek time, head switch time and rotational latency

# Introduction To Hard Disk Drives

Mark E. Donaldson

and is measured in ms. As you now know, the seek time only tells you about how fast the head is positioned over a wanted cylinder. Until data is read or written you will have to add the head switch time for finding the track and also the rotational latency time for finding the wanted sector.

## Seek Time & Head Switching Time

The fastest seek time occurs when moving from one track directly to the next. The slowest seek time is the so called full-stroke between the outer and inner tracks. Some harddisks (especially SCSI drives) don't execute the seek command correctly. These drives position the head somewhere close to the desired track or leave the head where it was. The seek time everyone is interested in is the **average seek time**, defined as the time it takes to position the drive's heads for a randomly located request. Yes, you are correct: seek time should be smaller if the disk is smaller (5 ¼ vs 3 ½)

All heads of a harddisk are carried on one actuator arm, so all heads are on the same cylinder. Head switch time measures the average time the drive takes to switch between two of the heads when reading or writing data.. **Cylinder switch time** is the average time it takes to move the heads to the next track when reading or writing data. All these times are measured in milliseconds (ms).

Seek time is the time it takes the read/write heads to move from their current location to the track where the desired information is located. Since the desired track could be located on the other side of the platter, or on an adjacent track, the seek time will vary for each individual seek transaction. In actuality, the average seek time for any arbitrary track is equal to the time required to seek across one-third of the tracks. Some drive manufacturers also cite **track-to-track seek time**, which is simply the amount of time to move from one track to an adjacent track. Today's hard drives have track-to-track seek times between three and five milliseconds and average seek times of less than 10 milliseconds.

## Rotational Latency

Each track on a hard drive contains multiple sectors. Once the read/write heads seek the correct track, the heads remain in place and idle until the correct sector passes under them. This waiting time is called latency. The average latency is equal to the time it takes the platter to make a one-half revolution; for example, on a platter spinning at 3600 RPM, one revolution takes 16.67 milliseconds, so the average latency is 8.3 milliseconds. The latency is identical on those drives that spin at the same speed. Some of today's hard-drive models have platters that spin at 5500 RPM or more, thus reducing latency.

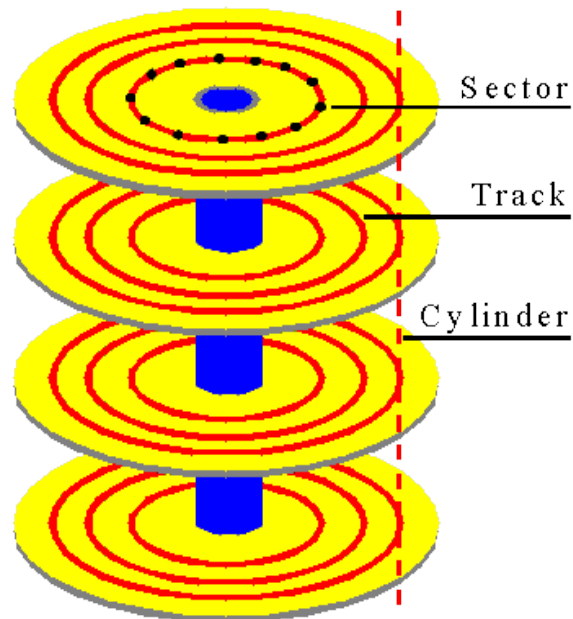
## Command Overhead

The command overhead is the time it takes the controller to process a data request. This includes determining the physical location of the data on the correct platter, directing the actuator to move the head stack assembly to the correct track, reading the data, and forwarding it to the computer. For today's hard drives, the disk overhead is relatively insignificant.

After the head is positioned over the desired track, it has to wait for the right sector. This time is called rotational latency and is measured in ms. The faster the drives spins, the shorter the rotational latency time. The average time is the time the disk needs to turn half way around, usually about 4ms (7200rpm) to 6ms (5400rpm).

# Introduction To Hard Disk Drives

Mark E. Donaldson



## Transfer Rate

Hard disks are also evaluated by their transfer rate, which generally refers to the rate at which data can be read from, or written to, the drive. The speed of the platters, density of the data bits, and access time affect the transfer rate. Transfer rates become particularly important when reading and writing large files. Today's drives have transfer rates ranging between five and thirty Megabits per second.

Most of today's hard drives include a small amount of RAM that is used to cache, or temporarily store, data. Some drive specifications refer to a burst transfer rate, or the speed at which data can be read from, or written to, the cache. The sustained transfer rate more accurately reflects the amount of data that can be accessed from the platters in a given amount of time.

## Cache

All modern HDs have their own cache varying in size and organization. The cache is normally used for writing and reading. On SCSI HDs you may have to enable write caching, because often it is disabled by default. This varies from drive to drive. You will have to check the cache status with a program like [ftp://ftp.seagate.com/techsuppt/seagate\\_utils/aspidd15.zip](ftp://ftp.seagate.com/techsuppt/seagate_utils/aspidd15.zip). You may be surprised that it is not the cache size that is important, but the organization of the cache itself (write / read cache or look ahead cache).

With most EIDE drives, the PC's system memory is also used for storing the HD's firmware (e.g. software or BIOS). When the drive powers up, it reads the firmware from special sectors. By doing this, manufacturers save money by eliminating the need for ROM chips, but also give you the ability to easily update your drives BIOS if it is necessary (Like for the WD drives which had problems with some motherboard BIOS' resulting in head crashes!).

## Organization of Data On Disks

You now know, a harddisk has cylinders, heads and sectors. If you look in your BIOS you will find these 3 values listed for each harddisk in your computer. You learned that a harddisk doesn't have a

# Introduction To Hard Disk Drives

Mark E. Donaldson

fixed sector size as they had in earlier days. Today, these values are only used for compatibility with DOS, as they have nothing to do with the physical geometry of the drive. The harddisk controller calculates these values into a **logical block address (LBA)** and then this LBA value is converted into the real cylinder, head and sector values. Modern BIOS are able to use LBA, so limitations like the 504 MB barrier are now gone. Cylinder, heads and sectors are still used in DOS environments. SCSI drives have always used LBA to access data on the harddisk. Modern operating systems access data via LBA directly without using the BIOS.

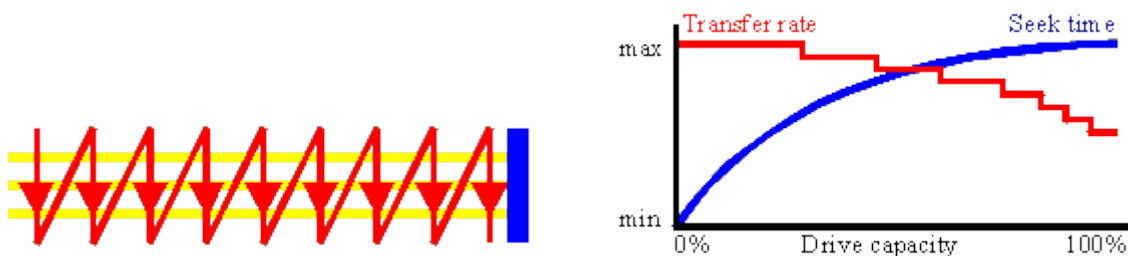
## Transfer Rates

Hard disks are also evaluated by their transfer rate, which generally refers to the rate at which data can be read from, or written to, the drive. The speed of the platters, density of the data bits, and access time affect the transfer rate. Transfer rates become particularly important when reading and writing large files. Today's drives have transfer rates ranging between five and thirty Megabits per second.

Most of today's hard drives include a small amount of RAM that is used to cache, or temporarily store, data. Some drive specifications refer to a burst transfer rate, or the speed at which data can be read from, or written to, the cache. The sustained transfer rate more accurately reflects the amount of data that can be accessed from the platters in a given amount of time.

In the pictures you can see the several ways how data can be stored physically on the harddisk. With a benchmark program that calculates the transfer rate or seek time of the whole harddisk you can see if your drive is using a 'vertical' or a 'horizontal' mapping. Depending on what kind of read/write heads and servo-motors (for positioning the actuator arm) are used it is faster to switch heads or to change tracks.

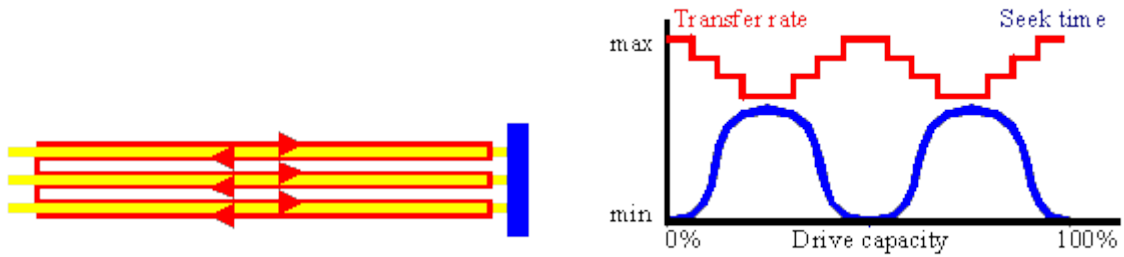
Traditional harddisks orders their capacity in **vertical mapping**. The data is read/written from one cylinder first, starting at the top track down to the bottom, before the heads are moved to the next cylinder.



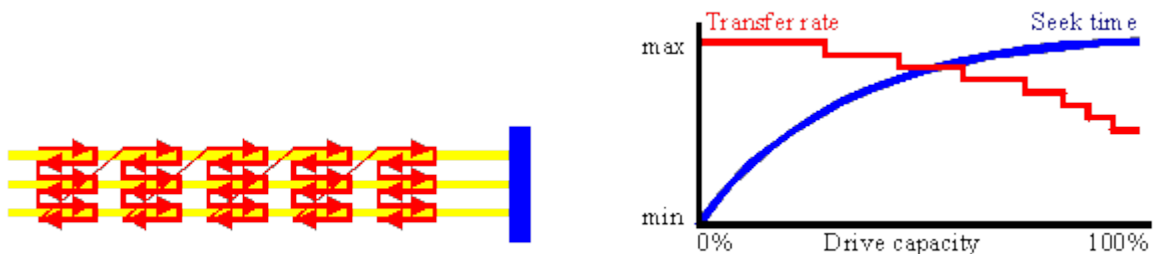
In **horizontal mapping**, the data is read/written starting from the outer cylinder to the inner cylinder, before switching the heads to the next track.

# Introduction To Hard Disk Drives

Mark E. Donaldson



Some harddisks use a combination of vertical and horizontal mapping. As you can see in the below pictures, transfer rate is higher when data is read or written to the outer parts of a disk. The reason is that there is more space for sectors. The number of sectors varies in steps. Usually on a disk there are 10 to 20 zones (called 'notches') with a constant sector number. That's the reason why you see the steps in the transfer rates.



Some harddisk use the combination of 'vertical' and 'horizontal' mapping. The 'horizontal' mapping is used in the zones, the 'vertical' mapping between the zones. However, transfer rate and seek time look the same to 'vertical' mapping. If you are going to buy a HD you have may need to know what kind of mapping the drive does. If you need constant transfer rates (for video, audio) you should get a drive which doesn't do the horizontal mapping. However, drives with horizontal mapping are not very common.

## Drive Interface (EIDE or SCSI)

Currently there are 2 different interfaces: EIDE and SCSI. You will find an EIDE controllers integrated with the motherboard and that EIDE harddisks are much cheaper than SCSI drives. For SCSI you need an extra controller, because there aren't a lot of motherboards with integrated SCSI controllers. Together with the higher price of a SCSI disk a SCSI system is more expensive than EIDE.

The EIDE interface has a primary and a secondary channel that will connect to two devices each, for a total of four. That could be a harddisk, CD-ROM or disk changers. Lately there have been tape backups with EIDE connectors, but you need special backup software. Scanners and CD-writers aren't available with EIDE interface, only with SCSI.. You can connect up to 7 devices to a SCSI bus or 15 devices to a Wide SCSI. In a standard environment, the performance of single harddisk won't improve much from the SCSI interface. Rather, the power of SCSI is that several devices can use the bus at the same time, not using the bus while they don't need it. So, we see the best benefit from SCSI when several devices are all used on the same bus. The SCSI interface comes in several types: 8-bit (50 wire data cable) or 16-bit (68 wire data cable, Wide SCSI). The clock can be 5 MHz (SCSI 1), 10 MHz (Fast SCSI), 20 MHz (Fast-20 or Ultra SCSI) or 40 MHz (Ultra-2 SCSI).

# Introduction To Hard Disk Drives

Mark E. Donaldson

On one EIDE channel, the 2 devices have to take turns controlling the bus. If there is a harddisk and a CD-ROM on the same channel, the harddisk has to wait until a request to the CD-ROM has finished. Because CD-ROM's are relatively slow, there is a degradation of performance. That's why everybody tells you to connect the CD-ROM to the secondary channel and your harddisk to the primary. The primary and secondary channels work more or less independently of one another (it's a matter of the EIDE controller chip). The theoretical transfer rate of EIDE is up to 16.6 MByte/s in PIO mode 4 or DMA mode 2.

Today's CD-ROM's often use PIO mode 3, while older device use PIO mode 0 to 2. Sometimes devices lie about the PIO mode they support. There are harddisks that say they are able to use PIO mode 2 but they only work reliably in PIO mode 1! Whenever you get errors accessing your harddisk, try to lower the PIO mode first!

It is not only the interface transfer rate that determines how fast a harddisk is. How fast the data can be written or read from the media, e.g. data density and rotation speed is more important. The fastest interface can't do anything faster than the 'inner values' of a harddisk are capable of. Today, most harddisks are still under 10 MByte/s transfer rate physically. A faster interface is advantageous on when data is read from or written to the cache in a multitasking environment with several devices accessed simultaneously. Multitasking environments especially benefit from SCSI, since simultaneous access occurs frequently. If you have a server or are working with large files like audio, video or disk-intense applications, you will benefit more from SCSI than EIDE. There are three reasons for this

- All modern operating systems now supports SCSI very well. Windows 3.x didn't!
- Busmastering really works better with a SCSI busmastering controller.
- The fastest harddisks with the best performance are SCSI.

If you need large capacities and the highest transfer rates available on the market you need SCSI. This is not because EIDE is incapable of this, it's because of the market. High-end disks with high capacities and high performance are intended to be used in servers and aren't build with EIDE interface. At the moment, EIDE disks are only built with up to a 5 Gigabyte capacity (there is a problem with a 4 GB barrier with some BIOS's again and for drives bigger than 8 GB you need a new BIOS that supports the INT 13 functions AH=41h bis 49h) and transfer rates of about 9 MBytes/s. If you need more, you'll have to use SCSI. Also, SCSI harddisks have larger cache RAM than EIDE harddisks.

## Performance

You need to know how a slow or fast harddisk affects your overall system performance in a standard environment. If your operating system isn't constantly swapping (e.g. you have enough memory) the speed of a harddisk is only a small part of a well balanced system. Let's say you have a harddisk that has 30% better performance than another older one; the benefit for standard applications would be from 2% up to 18%. Sometimes, you want or need the fastest components available. Other times, more capacity and reliability is needed. There are several programs available that test the performance of a harddisk. Some are crap, others are good. In any case, if you have one, you get numbers that tell you something. But do you have a point of comparison? Different benchmarks mean different numbers. Different environments mean different numbers. Modern benchmarks are

# Introduction To Hard Disk Drives

Mark E. Donaldson

independent from existing data on the harddisk (only read performance testing can be done). But a benchmark could be affected by several things:

- To which channel is the harddisk connected
- Is the harddisk alone or together with other devices connected to the controller
- Under which operating system is the harddisk tested and used
- Which drivers are loaded

## File Systems

If you purchase a new harddisk it is physically pre-formatted. This means that the cylinder, track and sector information is already written onto the disk. You now have to partition the disk to prepare it for the logical formatting and writing the partition information and boot sector to the disk. You don't have to use (or you aren't able!) the whole harddisk with one partition. You can divide it in several partitions. Depending on the operating system there are several file systems option. Famous file systems are FAT, (DOS, Windows), NTFS (Windows NT) and HPFS (OS/2).

## File Allocation Table

The FAT organizes several sectors into clusters and uses 12- or 16-Bit-cluster address numbers. The 16-Bit FAT is able to address up to 65526 clusters (some are used for special purposes). A cluster can be as big as 32 KBytes, which translates into a maximum partition size of 2 GBytes. The side effect is

that every file takes up at least 32 KBytes, even if it is only one kByte in size. That is the reason why power users like us divide a harddisk into slightly less then 512 Mbytes per partition--then 'only' 8 KBytes per cluster are used and you don't waste to much of your limited space on the harddisk. You can check your cluster size with `chkdsk.exe` at the DOS-prompt. Partition magic is a utility that lets you change partition size without loosing the data on your harddisk.

## FAT32

DOS 7.1, (shows as version 950b) now ships with a FAT size of 32 Bits. 4 Bits are reserved, hence 268,435,456 clusters can be addressed. This allows for partitions of up to 8 Gbytes with clusters of only 4 Kbytes in size. The maximum size of a partition is 2 TByte (2048 GByte). Compared to FAT16, FAT32 doesn't have a fixed size for the root entries, so you can now store as many directories and files you want in the root of your drive. Without changes, DOS applications can only access files which aren't bigger than 2 GBytes, and Win32 applications can work with files up to 4 GBytes. Wow! you think, that's great. Yes, I got really excited about these features, but don't stop reading here! FAT32 partitions are only accessible from this new Windows95 and the included DOS 7.1. No other operating system can access any data, including the really new Windows NT 4.0. All disk utilities that aren't written for FAT32 wont work correctly with the new file system. If you are lucky, your disk utilities just wont work. If you aren't lucky, they will destroy your data.

Windows95 OSR 2 ships with new versions of FDisk, Format, Scandisk and Defrag that can work with FAT32 partitions. However, the included DriveSpace3 doesn't! You can't use DriveSpace3 with FAT32 partitions! But that's not all: There are existing API's that don't work with FAT32 partitions, so some of your programs might not work as well. MS-DOS block device drivers like the often needed `ASPIDISK.SYS` for SCSI drives have to be revised to work with FAT32 drives! You now know how excited I was after I discovered that I can't access my SCSI harddisk in DOS 7.1 anymore, because I use it without BIOS support, only via a device driver.

# Introduction To Hard Disk Drives

Mark E. Donaldson

But that's not all! There is a noticeable decrease in performance with FAT32 drives compared to FAT16 drives! I first noticed this when I defraged my harddisk - the operation took much longer than usual. I wanted to be sure, so I tested the harddisk with Winbench. The DiskWinmark96 for an EIDE harddisk decreased by about 3% with Windows95 build 1111 FAT32 vs. FAT16. Under Windows95 build 1111 with FAT32 compared to Windows95 build 950a with a FAT16-drive of same size, the DiskWinmark96 decreased about 6%. The accessing of many small files was recognizably slower. Not only FAT32 is slower (which is due to the larger amount of clusters), the new build of Windows95 is slower with harddisk access in general! This doesn't mean that FAT32 is necessarily slower, this means that Microsoft did something with harddisk access in their new Windows95 OSR 2 that decreases harddisk performance in some way. It is not the EIDE driver; I also measured and saw performance decrease with other EIDE-drivers.

The application benchmark Winstone96 wasn't affected very much by this behavior. I always measured lower values with the newer build and also with FAT32, but that could have been from measuring errors. The difference was too small to say it is affected. Unfortunately, the BAPCo doesn't run on the new build of Windows95, but I guess it also won't show much difference in performance. That is due to the very small impact of the harddisk on these benchmarks. It would play a greater role if you only had 8 MB RAM.

## Conclusion

If you can, use build 950 (version 950a) of Windows95 and use partitions close to 512 MBytes with 8 KByte clusters. If you have build 1111 (version 950b) of Windows95 you can also use FAT32 drives with 4 KByte clusters to save space, especially with bigger partitions. Or to get rid of all these problems use another operating system which uses a different file system. The OSR2 DOS command 'FORMAT' has an undocumented switch '/Z:n' where  $n * 512$  bytes = cluster size, so you can set your own cluster size! The default cluster size is 4096 bytes = 4 kbyte. But be careful, smaller cluster size means millions of clusters, hence the management of them consumes more time.

Of course, there is a performance hit (look above!), which is measurable with benchmarks, but hardly recognizable by human. But it's getting really worse if you SCANDISK or DEFRAG (DISKDOCTOR or SPEEDDISK with Norton Utilities) your disk. I formatted a 2.5G EIDE disk with clusters of 1024 bytes size. (512 bytes is not possible). Yes, I dare nothing! ;) Now the 2.5G HDD has 2,480,338 (yes, nearly two and a half million) clusters of 1024 bytes. Even the FORMAT command warned me that this cluster size is very small and may result in a performance impact! Nice, but who trusts programs? I put back about 1.2 Gbyte of data and programs and was first impressed about the larger amount of free space. Then I run SPEEDDISK ... it is ridiculous! I am not a programmer and have no insight of how disk utilities works, but it seems that they hold the whole FAT and some more stuff in memory. The size of SPEEDDISK in memory swallowed up to 50 Mbytes while optimizing the disk! Currently my poor PC only have 32 MB of RAM. So Windows started to swap around. Similar problems with SCANDISK and DISKDOCTOR: They work for ages messing around with the millions of clusters. You better run these programs alone with nothing else loaded and don't disturb them in any way!

The DOS version of SCANDISK and DISKDOCTOR are running endless while checking the allocation table. DISKDOCTOR for DOS eventually ended up with a low memory error. Hmm, I wonder why the heck it wants to store everything in memory? Now I turned off the automatically

# Introduction To Hard Disk Drives

Mark E. Donaldson

check of the haddisks at bootup after a Windows95 crash. As stated before, the Windows version of these programs are slower now, but that's nothing compared to the DOS versions!

So what? All I want to say is: Use the /Z switch of the FORMAT command with care and use partitions! Microsoft better had build NTFS into Windows95. I've never seen such problems with NTFS. (I guess all Linux freaks are laughing now!).

# Introduction To Hard Disk Drives

Mark E. Donaldson

## TECHNICIANS' GUIDE TO PC HARD DISK SUBSYSTEMS

This booklet (now electronic) is published as a service of Data Recovery Labs. Its purpose is to provide the fundamental knowledge of concepts and terminology that is necessary to deal with the complexities of hard disk subsystems. It is not a technical reference guide and should not replace original documentation made available by manufacturers.

The principal author is Nick Majors. He has been in the industry since 1979, designing and developing hardware and operating system enhancements and performance tools. He is an experienced machine language programmer and has provided technical training to support personnel for some of Canada's leading banks, corporations, government departments and PC service organizations. He is Manager of Technical Services for Data Recovery Labs which he founded in 1989.

While this booklet deals primarily with PC and PC compatible platforms, there is much information to benefit support personnel with other hardware platforms.

### HISTORY AND OVERVIEW

The original IBM PC, introduced in 1981, did not support any type of hard drive. Program code in the BIOS did not recognize any such device and early versions of DOS precluded mass storage by limiting the maximum number of directory entries. This is not surprising when you consider that the original 4.77 MHz PC came with 16 K of RAM, expandable to 64 K on the motherboard. Even floppy drives and a disk operating system were options to upgrade the built in cassette recorder port and ROM basic.

To utilize a hard drive in a PC requires:

- Hardware IRQ (interrupt request)
- I/O port addresses for the controller
- DMA channel (now optional)
- A low level program code
- A physical interface to the bus (expansion slot or built-in)
- Operating system support
- Sufficient power and cooling

After DOS 2 introduced the sub-directory system and support for mass storage devices, companies started selling hard drives for the PC. These were external cabinets that contained the disk itself, a controller card, and a dedicated power supply (since the PC's 63.5 watts were insufficient). Migrating from other architectures, these units connected to the PC through cabling to an 8 bit adapter card that plugged into an available expansion slot. These subsystems were configured to use one of the available IRQ's, DMA channels, a range of I/O ports, and program code that was made available through a device driver loaded into memory after startup (booting from a floppy).

In 1983, the IBM XT (eXTended) was unveiled with its built-in 10 MB fixed disk. IBM worked with a company (Xebec, I think) to take the controller components normally located in the external cabinet

# Introduction To Hard Disk Drives

Mark E. Donaldson

and build them right onto a bus interface card, creating what we commonly call the "hard disk controller".

Program code was provided by a ROM chip on the controller card which supplemented subroutines in the BIOS, and the internal power supply was upgraded to 135 watts to provide power for the internally mounted drive.

The XT design utilized:

- IRQ 5
- I/O port addresses 320-32f
- DMA channel 3
- Program Code from adapter ROM at c8000
- DOS 2.0 or higher

Several companies started manufacturing and selling similar drive/controller packages with various improvements including higher capacities, superior performance and built-in floppy controller components on the same card (to save an expansion slot). These third party subsystems could even be added to an original PC, so long as the internal power supply was also upgraded.

In 1984, the IBM AT (Advanced Technology) brought a complete overhaul to hard disk systems. Program code was included in the motherboard ROM BIOS, eliminating the need for a ROM chip on the new 16 bit controller card, and a higher quality drive improved access times. The system included use of newly added higher IRQ's, eliminated the use of DMA for hard drives, and changed the range of I/O addresses.

The AT design utilized:

- IRQ 14
- I/O port addresses 1f0-1f8
- DMA channel - none
- Program Code from motherboard ROM BIOS
- DOS 2.0 or higher

Hardware configuration details for the AT, including hard disk parameters, were stored in a low power CMOS chip connected to a small battery, eliminating many of the motherboard configuration dip switches and jumpers. The battery allowed information to be maintained while the computer was turned off and information could be changed by running a setup program from disk.

The original AT supported 14 different drive types, recognizing specific hard disks ranging from 10 to 112 MB. Any drive with physical parameters that did not match one of these types needed a ROM extension on the controller card or a device driver loaded during boot-up.

DOS versions prior to 4.0 (or 3.31) did not support partitions larger than 32 MB no matter how big a drive was. This was because of sector numbering that could not exceed 16 bit values (up to 65,536 sectors). To make a larger partition required the use of special software like Ontrack's Disk Manager. This software was so popular that many drive manufacturers shipped it with their product.

# Introduction To Hard Disk Drives

Mark E. Donaldson

Unfortunately, while this offered one of the better solutions, it did pose compatibility problems for many disk utilities because, in effect, you had a non-DOS partition. Many people opted instead to divide their drives into 32 MB partitions creating a C: D: E: etc. up to the physical capacity. Prior to DOS 3.3, even this was not possible, because DOS did not recognize extended partitions!

The number of drive types supported by CMOS has expanded to over 40 and most current BIOS's provide a user definable type to allow parameters to match any drive. Most PC's today still rely on this original CMOS drive type scheme to configure and control hard disks, though many architectures and BIOS's have changed how the information is stored and updated.

This inherently creates certain limiting factors, including a problem with more than two hard drives and BIOS/OS limits to recognizing more than 1024 cylinders, 16 heads and 63 sectors per track. Various technologies must be used to translate non-compliant physical geometry's into logical parameters recognized by the system to maintain compatibility with operating system and utility programs that are tied to these limits.

With each sector holding 512 bytes of data this allows for drives no larger than 504 MB.

**1024 X 16 X 63 = 1,032,192 sectors X 512 = 528,482,304 bytes**

To understand this limit, we have to examine how hard drives are accessed by a PC. Primarily, I/O commands are sent to the controller via a range of reserved port addresses. This is a very complicated and tedious process and the ROM BIOS includes a subroutine (INT 13) to perform these tasks for you. The DOS operating system then has certain functions that further simplify the process. They include INT 25/26 functions to read and write absolute sectors to the drive as well as higher level functions (INT 21) to Open files, Close files, Write data to files, etc. Most programs rely on these DOS functions to control access to hard drives and DOS in turn calls the INT 13 BIOS subroutines which do the actual I/O commands.

All of these subroutines are assembly language code that are loaded into memory from either your BIOS chip(s) or Operating System files at startup. Assembly language routines store and manipulate values in registers within the CPU. The lowest common denominator for INTEL based PC's is 16 bit registers. Any program or routine that wants to be backward compatible, must use these base registers whether run on an 8088 or a Pentium. This basic INT 13 is the backbone of PC compatibility and uses 16 bit registers in the following way:

- DX - 8 bits for head number - 8 bits for drive number
- CX - 10 bits for cylinder number and 6 bits for sector number

The largest 10 bit number possible is 1023 hence the limit on cylinder numbers (from 0-1023) and the largest 6 bit number is 63 (from 1-63) allowing 63 sectors per track. But note the DX register allows a maximum of 255 heads, not the 16 that original specifications called for. This is what allows various translation schemes to deal with drives up to 8 GB while remaining INT 13 compatible.

If the DX register allows 8 bits for drive number, shouldn't it be able to control 255 drives instead of only two?

# Introduction To Hard Disk Drives

Mark E. Donaldson

When the INT 13 sends commands to your controller, it must know the physical geometry of the drive in question. During initialization of the PC, values for disk types are read from CMOS and stored into an

area of RAM called the BIOS DATA AREA. Pointers to those entries are stored in the **Interrupt Vector Table** (at addresses 0:104h and 0:118h). The table only reserved pointers for two drives, so even if your CMOS held more values, the standard routines wouldn't know how to deal with them.

Why were the INT 13 routines so limited? They were written at a time when 10 and 20 MB drives were the norm, and 120 MB was unbelievable. They were designed to communicate with a specific controller

interface, the ST412/506 standard (WD1003 controller), and the specs for passing parameters included only 10 bits for cylinders number, 4 bits for head number and a single bit for drive number.

Some of these limitations have been overcome in the past by replacing the PC's INT 13 sub-routines with code that could recognize and deal with different devices. That's why virtually all SCSI adapter cards include a ROM chip and you must set the drive type to ZERO. In effect, no standard drives installed for the motherboard BIOS to control. NetWare and other advanced Operating Systems use the drive type only long enough to boot-up and then replace the INT 13 code with their own device drivers. You then need to load a different .DSK file written specifically for each type of controller.

In the past couple of years, many motherboard and controller BIOS's have been enhanced to deal with ever increasing demands, but the process of establishing new standards has led to continuing confusion and compatibility problems.

## HARD DRIVES AND CONTROLLERS

Before we consider how to install, configure and maintain hard drives, we need a basic understanding of drive construction and design concepts. This chapter examines in some detail the parts and functional components of hard drive subsystems. (Note : A number of acronyms are used throughout this chapter and the glossary for this booklet is not yet available. Therefore, I have attached a brief set of definitions for some of the terminology.)

A hard drive subsystem is comprised of the following components:

- The Hard Disk, with one or more boards (PCB) attached.
- A Controller Mechanism, either on the hard disk PCB or on the bus adapter within the PC.
- Bus Adapter for interfacing the controller to the host PC.
- Cables and Connectors to link it all together.

## THE HARD DISK

Within a sealed enclosure (Head Disk Assembly or HDA) are one or more rigid platters that are "fixed" or non-removable. These are coated with magnetically sensitized material and data can be written to and read from the surface by means of electromagnetic read/write heads. When powered up, the platters are constantly rotating (except for certain pre-programmed sleep modes) and the heads are moved back and forth across the surface to access different locations. This is a sealed unit which should not be opened, except by qualified personnel in a controlled, dust free environment.

The circuit board(s) attached to the outside of the HDA provide the electronics needed for physical control of the motors within the sealed unit. They interface the source of electrical power to the disk

# Introduction To Hard Disk Drives

Mark E. Donaldson

assembly through varied connectors and cables. Most boards have some jumpers, dip switches and/or resistors that are used for configuration purposes. Functionally, these PCB's are separate from the Hard Disk Controller, but many of the newer drives (IDE and SCSI) embed the controller chip directly onto this board (as opposed to having it on the Bus adapter).

## INSIDE THE HDA - PARTS OF A HARD DISK:

- Disk Platter(s), separated by spacers and held together by a clamp.
- Spindle shaft onto which platters are mounted.
- Spindle motor for rotating the platters.
- Electromagnetic read/write heads (one per surface).
- Access arms or armatures from which the heads are suspended.
- Actuator for moving the arms (with heads attached).
- Preamplifier circuitry to maximize read/write signals.
- Air filter and pressure vent.

### The Platters:

Most platters or disks are made of an aluminum alloy, though ceramic or glass platters can also be found. The diameter is normally 2 1/2", 3 1/2" or 5 1/4" with a hole in the center for mounting onto the spindle shaft. Thickness of the media can vary from less than 1/32 of an inch to about 1/8 of an inch.

During manufacture the platters are coated with a magnetizable material. Older drives used a ferrite compound applied by squirting a solution onto the surface and rotating at high speeds to distribute the material by centrifugal force. This process left a rust colored ferrite layer which was then hardened, polished and coated with a lubricant. Newer drives apply the magnetic layer by plating a thin metal film onto the surface through galvanization or sputtering. These surfaces have a shiny chrome-like appearance.

### Spindle and Spindle Motors:

Most drives have several platters that are separated by disk spacers and clamped to a rotating spindle that turns the platters in unison. A direct drive, brushless spindle motor is built into the spindle or mounted directly below it. (Sometimes this motor is visible from outside of the sealed enclosure) The spindle, and consequently the platters, are rotated at a constant speed, usually 3,600 RPM, though newer models have increased that to 4800, 5400, or 7,200. The spindle motor receives control signals through a closed loop feedback system that stabilizes to a constant rotation speed. Control signals come from information written onto the surface(s) during manufacture or with older drives, from physical sensors.

### Read/Write Heads:

Since both sides of each platter are coated to provide separate surfaces, there is normally one electromagnetic read/write head for each side of each platter. Therefore, a drive with 4 platters would have 8 sides and 8 heads. Some drives use one side as a dedicated surface for control signals leaving an odd number (5,7,etc.) of heads for actual use. Each head is mounted onto the end of an access arm and these arms (one per surface) are moved in unison under the control of a single actuator mechanism. When one head is over track 143, all the heads on all other sides should be at the same location over their respective surfaces. Generally speaking, only one of the heads is active

# Introduction To Hard Disk Drives

Mark E. Donaldson

at any given time. There are some drives that can read or write from two or more heads at a time, but this represents a major design change and the technology is not yet widely used.

The spinning disk(s) create an air cushion over which the heads float. Depending on design, this air buffer ranges from 2 to 15 microns. By contrast, a smoke particle or finger print is about 30 microns in size! The heads are not supposed to come into contact with the surface during rotation. Only when powered off should the heads come to rest on the surface, but this should be over a specific area of the surface, reserved for that purpose. Most drives built since the late 1980's employ an automatic parking feature which moves the heads to this designated region and may even lock the heads there until powered up.

## Head Actuators:

The head actuator is the positioning mechanism used to move the arms and consequently the heads, back and forth over the surface. Once again, earlier drives used a different method than is now common. Originally, head positioning was controlled by a stepper motor that rotated in either direction by reacting to stepper pulses and moving the head assembly back and forth by means of a "rack and pinion" or by spooling and unspooling a band attached to the actuator arms. Each pulse moved the assembly over the surface in predefined steps or detents. Each step represented a track location and data was expected to be under the head. This design, still used for floppy drives, is not suitable for current drive densities and is prone to alignment problems caused by friction, wear and tear, heat deformation, and lack of feedback information needed for correcting positioning error.

The more common voice coil actuator controls the movement of a coil toward or away from a permanent magnet based upon the amount of current flowing through it. The armatures are attached to this coil and move in and out over the surface with it. This is a very precise method, but also very sensitive. Any variation in the current can cause the head assembly to change position and there are no pre-defined positions. Inherently this is an analog system, with the exact amount of movement controlled by the exact amount of current applied. The actual position of the coil is determined by servo (or indexing) information, which is written to the drive by the manufacturer. Location is adjusted to different tracks by reading and reacting to these control signals.

## Internal Electronics:

There is surprisingly little circuitry found within the sealed HDA. There are electrical and control wires for the spindle and head actuator motors and the head assembly has flex cables with a preamplifier chip often built onto it. This chip takes pulses from the heads (as close to the source as possible) and cleans up and amplifies these signals before transmission to components outside of the housing.

## Air Filtering and Ventilation:

Minor wear of internal components and occasional contact of the heads with the surface can cause microscopic particles to be loosened within the HDA. A permanent air filter is mounted within the air stream to remove these particles before they can cause damage to delicate mechanisms. Most drives also have a small vent to allow for minor air exchange from outside of the housing. This allows for equalization of air pressure so drives can be used in different environments without risk of imploding or exploding.

## CONTROLLERS AND BUS ADAPTERS

The hard disk controller provides the logical link between a hard disk unit and the program code within the host computer. It reacts to requests from the computer by sending seek, read, write, and

# Introduction To Hard Disk Drives

Mark E. Donaldson

control signals to the drive and must interpret and control the flow of data. Data moving to and from the drive includes sector ID's, positioning information and timing or clock signals. The controller must encode, decode and separate this control information from actual data written to or read from the drive. Also, data is sent to and from the drive serially, in bit format, but the smallest unit that a CPU can work with is a byte (8 bits). The controller must take bits (8 - 16 - or 32 at a time) and assemble them into bytes, words, and doublewords that can be transferred to/from the computer.

## OUR INDUSTRY MUST LOVE STANDARDS - WE HAVE THOUSANDS OF THEM!

And so it is with hard disk controllers. Controllers can be categorized in several different ways, by : Basic computer design (PC/XT vs AT-286-386-486,etc) as mentioned in the first chapter, standard AT controllers use different I/O addresses, IRQ and employ PIO as opposed to DMA.

### Bus Architecture (8-16 bit ISA, 32 bit MCA/EISA/VLB/PCI, etc.)

The adapter must be designed to interface with and use features of available expansion spots in the host computer.

### Controller Card vs Adapter

The expansion board that plugs into the PC is commonly referred to as a controller card, but for many drives (primarily IDE and SCSI) the controller mechanism is built directly onto the drive PCB and the expansion board in the PC (or built into motherboard) is actually a Host/Bus adapter.

## TROUBLESHOOTING TIP

If the BIOS reports "HDD CONTROLLER FAILURE" don't assume the problems is with your AT/IO board. It might well be the drive PCB that has failed.

### Controller/Drive Interface

Both drive and controller must communicate in the same 'language' and several different standards have been established. These include ST506/412, ESDI, SCSI, IDE(ATA/XTA) and EIDE(ATA2).

### Data Encoding Method

Determines how densely data can be packed onto a track.

- **MFM (Modified Frequency Modulation)** encoding is sufficient for only 17 x 512 byte sectors per track.
- **RLL (Run Length Limited)** permits up to 27 and variations of.
- **ARLL (Advanced Run Length Limited)** allow 34 or more sectors per track. This recording density is a major determinant of storage capacity, and with rotation speed and interleave are critical factors for true data transfer capability.

### Support for Translation

Some controllers present different logical parameters to the PC than the actual physical geometry of the drive.

### Need for ROM Extension or Software Device Driver

Additional program code is used to provide support for hard drives when none exists (as in PC/XTs), to implement translation schemes (as in ST506/RLL and ESDI designs), allow for non-standard devices or features (SCSI), or for a combination of these (EIDE). Below is a quick list of the major

# Introduction To Hard Disk Drives

Mark E. Donaldson

combinations that have been used in PCs past and present. While I am sure many others could be added, these are the ones I have come across over the years.

<u>Computer</u>	<u>Bus Connection</u>	<u>Interface</u>	<u>Encoding</u>	<u>Translate</u>	<u>ROM</u>
PC/XT	8 bit ISA Controller	ST506/412	MFM	NO	YES
PC/XT	8 bit ISA Controller	ST506/412	RLL	OPTION	YES
AT	16 bit ISA Controller	ST506/412	MFM	NO	NO
AT	16 bit MCA Controller	ST506/412	MFM	NO	NO
AT	16 bit ISA Controller	ST506/412	RLL	OPTION	YES
AT	16 bit MCA Controller	ST506/412	RLL	YES	YES
AT	16 bit ISA Controller	ESDI (10 Mbps)	RLL	OPTION	YES
AT	16 bit ISA Controller	ESDI (24 Mbps)	ARLL	OPTION	YES
AT	16 bit MCA Controller	ESDI (PS/2)	RLL,ARLL	YES	YES
PC/XT	8 bit ISA Adapter	SCSI	RLL	YES	YES
AT	16 bit ISA Adapter	SCSI	RLL,ARLL	YES	YES
AT	16 bit MCA Adapter	SCSI	RLL,ARLL	YES	YES
AT	32 bit EISA Adapter	SCSI	RLL,ARLL	YES	YES
AT	32 bit VLB Adapter	SCSI	RLL,ARLL	YES	YES
AT	32 bit PCI Adapter	SCSI	RLL,ARLL	YES	YES
PC/XT	8 bit ISA Adapter	IDE/XTA	RLL	OPTION	YES
AT	16 Bit ISA Adapter	IDE/ATA	RLL,ARLL	OPTION	NO
AT	32 Bit VLB Adapter	EIDE/ATA2	ARLL	OPTION	YES
AT	32 Bit PCI Adapter	EIDE/ATA2	ARLL	OPTION	YES

ESDI drives have some of the controller logic built onto the hard drive PCB and some on the controller card. PS/2 ESDI uses the same physical interface as other ESDI devices, but supports additional features specific to their implementation. Not to mention hundreds of other combinations to support different interleaves, track buffers, hardware caching, bus mastering, error correction schemes, SCSI I-II-III, optional floppy control, ESDI to SCSI converters, ST506 to SCSI converters (etc., etc., etc.).

So, what does all this mean to you? Specifically, don't be surprised if the drive you have in your left hand, does not work correctly with the controller / adapter you have in your right hand. Also, if controllers are changed it may affect performance as well as the ability to access previously recorded data.

## ACRONYM DEFINITIONS

**IRQ (Interrupt Request)** - Lines on the bus used to signal hardware interrupts.

**I/O (Input Output)** - Peripherals accessible by the CPU through registers at specific I/O addresses (or I/O ports).

**PIO (Programmed Input Output)** - Exchange of data between memory and peripherals by means of Input Output commands.

**DMA (Dynamic Memory Access)** - Transferring data directly between memory and peripherals without going through the CPU.

# Introduction To Hard Disk Drives

Mark E. Donaldson

## **BUS ARCHITECTURES:**

**ISA (Industry Standard Architecture)** - 8 bit and 16 bit expansion slots used by PC, XT, and AT designs. Often called IBM Standard Architecture.

**EISA (Extended Industry Standard Architecture)** - Developed by several independent manufacturers (Compaq, AST, Zenith, Tandy, etc.) to standardize 32 bit operation and combat IBM's MCA.

**MCA (Micro Channel Architecture)** - Expansion bus introduced by IBM in 1987, used by some (but not all) PS/2 models.

**PCI (Peripheral Component Interconnect)** - High speed bus developed by Intel to support the demands of Pentium and 486 based computers.

**VLB (VESA Local Bus)** - High speed, 32 bit extension to the ISA bus promoted by the VESA (Video Electronics Standards Association).

## **DRIVE INTERFACES:**

**ST506/412** - Standard interface used on XT and AT drives and controllers. Originally developed by Seagate Technologies to support their ST506 (5 MB) and ST412 (10 MB) drives. The entire controller mechanism is located on a controller card and communications between the drive and controller flow over 2 ribbon cables - one for drive control and one for data.

**ESDI (Enhanced Small Device Interface)** - Developed by Maxtor in the early 1980's as an upgrade and improvement to the ST506 design. While the drive does not have an embedded controller, one of the most critical functions, encoding-decoding, is performed on the drive. This allows for faster communications and higher drive capacities. Uses the same cabling as ST506 interface, but carries different signals on each line.

**SCSI (Small Computer System Interface)** - Based on an original design by Shugart Associates, SCSI is not specifically a drive interface, but a method of allowing different devices to communicate with a PC. For hard drives the entire controller is built onto the drive PCB, allowing for very high speed transfers to and from the drive. Fully interpreted, parallel data is then transferred to and from the PC by way of a single cable through a bus interface that has configured the device as a hard drive.

**IDE (Integrated Drive Electronics)** - A technology pioneered by Compaq and Conner that embedded a controller onto the hard disk PCB while maintaining compatibility with the register level commands sent by the computer's INT 13 routines. IDE drives are configured and appear to the computer like standard ST506 drives.

**ATA (AT Attachment)** - Implementation of the IDE design with a 16 bit AT style controller on board the drive.

**XTA (XT Attachment)** - Rarely used implementation of IDE with an integrated 8 bit XT controller.

# Introduction To Hard Disk Drives

Mark E. Donaldson

**ATA-2** - Enhancement to the AT Attachment standard to provide for considerable performance improvement and more sophisticated drive identification.

EIDE (Enhanced IDE) and FAST-ATA - Various implementations of the ATA-2 standard as marketed by Western Digital (EIDE) and Seagate/Quantum (FAST-ATA).

## DATA ENCODING SCHEMES

**MFM (Modified Frequency Modulation)** - Common technique used to encode the magnetic fluxes recorded on a drive into data. Still used on floppy drives and most original XT and AT systems. Notice that most drive types supported in CMOS have 17 sectors per track. This is the standard density for MFM encoding.

**RLL (Run Length Limited)** - Encoding method that allows 50% more information to be recorded on a track than MFM. Actually accomplished by recording more fluxes for every byte, but packing them more tightly onto the surface. Often called 2,7 RLL because the recording scheme involves patterns with no more than 7 successive zeros and no less than two.

**ARLL (Advanced Run Length Limited)** - More complex yet powerful derivatives of the RLL scheme. Include 1,7 and 3,9 encoding.

## RAID

RAID (Redundant Array of Inexpensive Disks) is a storage technology that groups multiple hard drives into what appears to be one logical volume. The term RAID was introduced in a late-1987 paper by Patterson, Gibson, and Katz of the University of California-Berkley entitled "A Case for Redundant Arrays of Inexpensive Disks (RAID)." The paper compared RAID to SLED (Single Large Expensive Disk) and described five-disk array architectures, or levels. RAID technology is currently the hottest mass storage topic in the literature.

Disk arrays generally improve system performance by supporting multiple simultaneous read and/or write operations as well as by increasing capacity and providing fault tolerance. The use of multiple drives in an array actually increases the chances that a drive failure will occur. However, the data redundancy of RAID allows the array to tolerate a drive failure. A basic description of each of the levels of RAID follows:

### RAID 0

This form of RAID is not RAID as described in the Berkley paper because there is no data redundancy. Most disk arrays use striping, or distribution of data across multiple drives. RAID 0 implements striping without redundancy and is, therefore, less reliable than a single drive. The only advantage is increased speed.

### RAID 1

RAID 1 implements "mirroring," or shadowing, of disks. Each drive in the system has a copy, or "mirror, of itself. If a drive falls, the duplicate drive keeps working with no lost data or downtime. Since there are two sources of data, the average access time for a read request will be faster than that for a single drive. For a write request, which is almost always preceded by a read, the decrease in read seek time of RAID 1 is offset by the increase in write seek time (since the data has to be

# Introduction To Hard Disk Drives

Mark E. Donaldson

written to two disks). A read and two writes takes the same time as a read/write for a single drive. RAID 1, with an optimized controller, has slightly lower overall access times than a single drive.

The main advantage of RAID 1 over other RAID architectures is simplicity. It only requires a dual channel controller or a minimal device driver using one or two controllers to implement. No change to the operating system is needed. RAID 1 is relatively expensive to implement because only half the available disk space is used for data storage. In addition, the necessity of duplicate drives requires more power and more space for the same storage capacity.

**RAID 0/1 (sometimes also called 10) is a hybrid of RAID 0 and RAID 1.** The data is striped across the drives in the array as in RAID 0. In addition, each striped drive group is "mirrored" by a duplicate drive group attached to a second drive controller.

## RAID 2

RAID 2 is an architecture that succeeds in reducing disk overhead (the cost of storage space lost to redundancy) by using Hamming codes to detect and correct errors. Check disks are required in addition to the data disks. The data is striped across the disks along with an Interleaved Hamming code. Because all of the data disks must seek before a read starts, and because for a write the data disks must seek, read data, all drives (including check disks) must seek again, and then the data is written, seek times are very slow compared to a single drive. However, once the seek is completed, data transfer rates are very high. For an array with 8 data drives, the drives will transmit data in parallel. The transfer rate of the array will be 8 times that of a single drive.

RAID 2 is best for reading and writing large data blocks at high data transfer rates. In the microcomputer environment the existing error detection/ correction features result in redundant error isolation data for RAID 2 and make RAID 2 impractical for microcomputers. By letting the drives manage error detection, it is possible to implement RAID requiring only one check disk for error correction.

## RAID 3

By assuming that each disk drive in the array can detect and report errors, the RAID system only has to maintain redundancy in the data necessary to correct errors. RAID 3 employs a single check disk (parity disk) for each group of drives. Data is striped across each of the data disks. The check disk receives the XOR (exclusive OR) of all the data written to the data drives. Data for a failed drive can be reconstructed by computing the XOR of all the remaining drives. This approach reduces disk overhead from RAID 1 and 2. For a five-disk array, four of the drives store data; providing 4 GB of data storage in a 5 GB array. RAID 3 also has the same high transfer rates as RAID 2. However, because every data drive is involved in every read or write, a penalty is paid.

RAID 3 can process only one I/O transaction at a time. In addition, the minimum amount of data that can be written or read from a RAID 3 array is the number of data drives multiplied by the number of bytes per sector, referred to as a transfer unit. A typical five-drive array would have four data disks, one parity disk, and might have a 512-byte sector size on each disk. The transfer unit would be 2048 bytes (4 x 512). When a data read is smaller than the transfer unit, the entire unit is read anyway, increasing the length of a read operation. For a data write smaller than the transfer unit, although only a portion of a sector of each disk needs to be modified, the array must still deal with complete transfer units. A complete unit must be read from the array; the data must be rewritten where

# Introduction To Hard Disk Drives

Mark E. Donaldson

necessary; and the modified data must be written back to the data disks and the check disk updated. RAID 3 works well in applications that process large chunks of data.

## RAID 4

RAID 4 addresses the problems associated with bit-striping a transfer block of data across the array. As in RAID 3, one drive in the array is reserved as the check disk. This architecture, however, utilizes block or sector striping to the drives, resulting in read transactions involving only one drive and timing comparable to a single drive. In addition, multiple read requests can be handled at the same time. However, since every write accesses the parity disk, only one write at a time is possible. RAID 4 is most useful in an environment where the ratio of reads to writes is very high.

## RAID 5

Because RAID levels 2 through 4 each use a dedicated check disk, only one write transaction is possible at any time. RAID 5 overcomes the write bottleneck by distributing the error correcting codes (ECC) across each of the disks in the array. Therefore, each disk in the array contains both data and check-data. Distributing check-data across the array allows reads and writes to be done in parallel. Data recovery and seek times are comparable to RAID 4.

## Disk Array Implementations

Actual disk array implementations are not always as simple or straightforward as described above. Some manufacturers combine features of different RAID levels to create a hybrid, as in RAID 0/1. RAID implementations that are extremely fault-tolerant provide redundancy beyond that of the drives. Additional redundancy is accomplished by providing a system with redundant drive controllers, redundant power supplies, redundant SCSI controllers, and so on. Some manufacturers offer "hot swappability," the ability to replace a failed drive (or other hardware units) without shutting the system down. Other manufacturers offer a spare drive that is automatically put into use rebuilding the failed drive as soon as the system senses a failure.

Still other RAID manufacturers offer only software-based RAID. The RAID architecture is contained in software that the customer implements with his own hardware. And finally, some RAID systems offer more than one level of RAID in the same package to handle mixed applications more efficiently. A listing of RAID vendors and information about the products they offer appears at the end of this section in 9.5 RAID Vendors Database., although not all information is 'on-line.'