

HARD DISK DRIVES – CONCEPTS

Track density

This is a measure of how tightly the concentric tracks on the disk are packed: how many tracks can be placed down in inch of radius on the platters. For example, if we have a platter that is 3.74" in diameter, that's about 1.87 inches. The inner portion of the platter is where the spindle is, and the very outside of the platter can't be used either. About 1.2 inches of length along the radius is usable for storage. If in that amount of space the hard disk has 22,000 tracks, the track density of the drive would be approximately 18,333 tracks per inch (TPI).

Linear (recording) density

The density of bits recorded into track.

Areal density

Taking the product of these two values yields the drive's areal density, measured in bits per square inch. If the maximum linear density of the drive above is 300,000 bits per inch of track, its maximum areal density would be 5,500,000,000 bits per square inch, or in more convenient notation, 5.5 Gbits/in². The newest drives have areal densities exceeding 10 Gbits/in², and in the lab IBM in 1999 reached 35.3 Gbits/in²--524,000 BPI linear density, and 67,300 TPI track density! In contrast, the first PC hard disk had an areal density of about 0.004 Gbits/in²!

Areal density is a two-dimensional measure calculated by multiplying two linear measures: recording density (bit density) and track density. *Areal density*, sometimes also (imprecisely) called *bit density* or even just *density*, refers to the amount of data that can be stored in a given amount of **hard disk platter space**. It is one of the most important indicators of overall hard disk performance, though one that outside the PC enthusiast community is sadly under-discussed.

Areal density is strongly correlated to the **transfer rate specifications** of a drive. **The higher the drive's areal density**, in general, **the higher its transfer rates will be**, however, most of the improvement in transfer rate is due to increases in *bit* density, not track density. (When more bits are in a given length of track, the heads will read more data in a unit of time, assuming the spindle speed is constant.) If drive "A" has an areal density 5% lower than that of drive "B", but its bit density is 10% higher, it will have a higher transfer rate than drive "B".

Both bit density and track density have an impact on positioning performance. Increases in either one allow the data on the hard disk to be stored physically closer together on the disk. This reduces the distance that the read/write heads must seek to find different files on the disk, slightly improving seek time. Do keep in mind though that the improvements here are relatively small compared to the impact areal density has on transfer rates. Also, improvements only in track density don't do a lot to improve performance.

Areal density specifications are usually *maximum* specifications; look for the magic "M word" near the spec. The areal density will only be this high in certain regions of the disk. Modern drives use zoned bit recording to allow the areal density not to vary too greatly over the surface of the platter, but density will still be higher or lower in different parts of the disk.

There's also a "rough cut" areal density measure commonly used when talking about hard drives or comparing one generation of drives to another. Often, the total formatted capacity of the disk will be divided by the number of platters, and the density of the drive discussed in terms of "GB per platter". For example, the 30 GB Maxtor DiamondMax Plus 40 is a three-platter drive; its rough density then is 10 GB/platter, and that applies to all the members of that family. The IBM GXP75 family is 15 GB/platter, and so on.

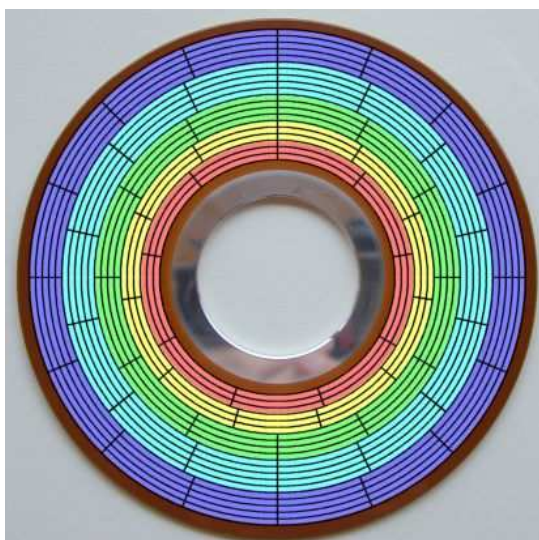
This is a convenient short-hand and is useful when discussing drives, just keep in mind its limitations. For starters, it's rather crude, so it's only good for contrasting different generations of drives with big differences in density. Second, implied in the "GB/platter" measure is the size of each platter. A 10 GB/platter drive with 2.5" platters has *much* higher density than a 10 GB/platter drive using 3.5" platters. Also, some drives use only one side of one their platters. The 15 GB DiamondMax Plus 40 for example uses two platters but only three of the four surfaces, so it is still a 10 GB/platter drive, not 7.5 GB/platter. (A better measure would be "GB per *surface*", but nobody seems to use that since most drives use both sides of each platter.)

The primary factors that influence areal density specifications are those that relate to data and recording: this means that *all* the factors discussed in this section are relevant. It is also influenced by the design and speed of the spindle motor; faster motors may require density to be reduced for reliability reasons.

One way that capacity and speed have been improved on hard disks over time is by improving the utilization of the larger, outer tracks of the disk. The first hard disks were rather primitive affairs and their controllers couldn't handle complicated arrangements that changed between tracks. As a result, every track had the same number of sectors. The standard for the first hard disks was 17 sectors per track.

Of course, the tracks are concentric circles, and the ones on the outside of the platter are much larger than the ones on the inside--typically double the circumference or more. Since there is a constraint on how tight the inner circles can be packed with bits, they were packed as tight as was practically possible given the state of technology, and then the outer circles were set to use the same number of sectors by reducing their bit density. This means that the outer tracks were greatly underutilized, because in theory they could hold many more sectors given the same linear bit density limitations.

To eliminate this wasted space, modern hard disks employ a technique called *zoned bit recording* (ZBR), also sometimes called *multiple zone recording* or even just *zone recording*. With this technique, tracks are grouped into zones based on their distance from the center of the disk, and each zone is assigned a number of sectors per track. As you move from the innermost part of the disk to the outer edge, you move through different zones, each containing more sectors per track than the one before. This allows for more efficient use of the larger tracks on the outside of the disk.



Write Precompensation

This is a relic from the mid-80s. Older disks that use the same number of sectors for every track sometimes required an adjustment to be made while writing, beginning at a certain track number, and this setting was that value.

Modern IDE/ATA and SCSI drives have built-in intelligent controllers that take care of these sorts of adjustments (and many more) automatically. This setting should normally be set to -1, 0 or 65535 (the largest value it can support) or whatever value the autodetection sets. Which of these "bogus" values your BIOS uses to mean "there is no write precompensation value for this drive" depends on the BIOS. The number itself is ignored by the drive in any event.

Write precompensation

Using a stronger magnetic field to write data in sectors that are closer to the center of the disk. In CAV recording, in which the disk spins at a constant speed, the sectors closest to the spindle are packed tighter than the outer sectors.

One of the hard disk parameters stored in a PC's CMOS memory is the WPcom number, which is the track where precompensation begins. See BIOS setup.

Reduced Write Current

In older hard drives, there were the same number of sectors near the center of the platter or disk as there was in the outer tracks. To get the same number of bits in each sector, the bits had to be placed closer together in the center tracks. Using the same magnetic current to write information to the center tracks sometimes affected adjacent data. To solve this problem, a method was devised that reduced the current as the heads got closer to the center.

Cylinder and Head Skew

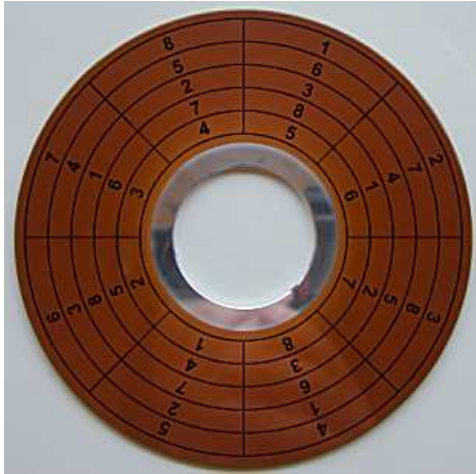
Sector interleaving was once used on older hard disks to ensure that the sectors were efficiently spaced on the track. This was needed to ensure that sector #2 didn't rotate past the head while sector #1 was being processed. The high-speed disk controllers on modern drives are now fast enough that they no longer are a performance-limiting factor in how the sectors on the disk are arranged. However, there are other delay issues within the drive that require spacing to be optimized in even the fastest drives, to maximize performance. And unlike the interleaving situation, these delays **are caused by electromechanical concerns** and are therefore likely to be with us for as long as hard drives use their current general design.

The first issue is the delay in time incurred when switching between cylinders on the hard disk, called appropriately enough, cylinder switch time. Let's imagine that we "lined up" all of the tracks on a platter so that the first sector on each track started at the same position on the disk. Now let's say that we want to read the entire contents of two consecutive tracks, a fairly common thing to need to do. We read all the sectors of track #1 (in sequence, since we can use a 1:1 interleave) and then switch to track #2 to start reading it at its first sector.

The problem here is that it takes time to physically move the heads to track #2. In fact, it often takes a millisecond or more. Let's consider a modern 10,000 RPM drive. The IBM Ultrastar 72ZX has a specification of only 0.6 milliseconds for seeking from one track to an adjacent one. That's actually quite fast by today's standards. But consider that in that amount of time, a 10,000 RPM drive will perform approximately 10% of a complete revolution of the platters! If sector #1 on track #2 is lined up with sector #1 on track #1, it will be long gone by the time we switch from track #1 to track #2. We'd have to wait for the remaining 90% of a revolution of the platters to do the next read, a big performance penalty. This problem isn't as bad as the interleave one was, because it occurs only when changing tracks, and not every sector. But it's still bad, and it's avoidable.

The issue is avoided by offsetting the start sector of adjacent tracks to minimize the likely wait time (rotational latency) when switching tracks. This is called *cylinder skew*. Let's say that in the particular zone where tracks #1 and #2 are, there are 450 sectors per track. If 10% of the disk spins by on a track-to-track seek, 45 sectors go past. Allowing some room for error and controller overhead, perhaps the design engineers would shift each track so that sector #1 of track #2 was adjacent to sector #51 of track #1. Similarly, sector #1 of track #3 would be adjacent to sector #51 of track #2 (and hence, adjacent to sector #101 of track #1). And so on. By doing this, we can read multiple adjacent tracks virtually seamlessly, and with no performance hit due to unnecessary platter rotations.

The same problem, only to a lesser degree, occurs when we change heads within a cylinder. Here there is no physical movement, but it still takes time for the switch to be made from reading another, so it makes sense to offset the start sector of tracks within the same cylinder so that after reading from the first head/track in the cylinder, we can switch to the next one without losing our "pace". This is called *head skew*. Since switching heads takes much less time than switching cylinders, head skew usually means a smaller number of sectors being offset than cylinder skew does.



These two diagrams illustrate the concept of cylinder and head skew. Assume that these platters spin counter-clockwise (as seen from your vantage point) and that they are adjacent to each other (they might be the two surfaces of the same platter.) They each have a cylinder skew of three, meaning that adjacent tracks are offset by three sectors. In addition, the platter on the right has a head skew of one relative to the one on the left. (Of course, real drives have thousands of tracks with hundreds of sectors each.)

Both cylinder and head skew must be simultaneously "overlaid" onto all the tracks of the hard disk, resulting in a "two-dimensional pattern" of sorts, with different offsets being applied depending on the specific timing characteristics of the disk. The layout of the tracks is adjusted to account for cylinder skew and head skew, based on the way the designers intend the hard disk to store sequential data. All of the details are taken care of by the controller.

This is one reason why having integrated, dedicated drive electronics on the disk itself, is such a good idea. No universal, external controller could possibly know how to take all these hard disk characteristics and performance requirements into account.

Sector Format and Structure

The basic unit of data storage on a hard disk is the *sector*. The name "sector" comes from the mathematical term, which refers to a "pie-shaped" angular section of a circle, bounded on two sides by radii and the third by the perimeter of the circle. On a hard disk containing concentric circular tracks, that shape would define a sector of each track of the platter surface that it intercepted. This is what is called a *sector* in the hard disk world: a small segment along the length of a track. At one time, all hard disks had the same number of sectors per track, and in fact, the number of sectors in each track was fairly standard between models. **Today's advances have allowed the number of sectors per track ("SPT") to vary significantly (Zoned Bit Recording).**

In the PC world, each sector of a hard disk can store 512 bytes of user data. (There are some disks where this number can be modified, but 512 is the standard, and found on virtually all hard drives by default.) Each sector, however, actually holds much more than 512 bytes of information. Additional bytes are needed for control structures and other information necessary to manage the drive, locate data and perform other "support functions". The exact details of how a sector is structured depends on the drive model and manufacturer. However, the contents of a sector usually include the following general elements:

- **ID Information:** Conventionally, space is left in each sector to identify the sector's number and location. This is used for locating the sector on the disk. Also included in this area is status information about the sector. For example, a bit is commonly used to indicate if the sector has been marked defective and remapped.
- **Synchronization Fields:** These are used internally by the drive controller to guide the read process.
- **Data:** The actual data in the sector.
- **ECC:** Error correcting code is used to ensure data integrity.
- **Gaps:** One or more "spacers" added as necessary to separate other areas of the sector, or provide time for the controller to process what it has read before reading more bits.

Note: In addition to the sectors, each containing the items above, space on each track is also used for **servo information** (on embedded servo drives, which is the design used by all modern units).

The amount of space taken up by each sector for overhead items is important, because the more bits used for "management", the fewer overall that can be used for data. Therefore, hard disk manufacturers strive to reduce the amount of non-user-data information that must be stored on the disk. The term *format efficiency* refers to the percentage of bits on each disk that are used for data, as opposed to "other things". The higher the format efficiency of a drive, the better (but don't expect to find statistics on this for your favorite drive easy to find!)

One of the most important improvements in sector format was IBM's creation of the *No-ID Format* in the mid-1990s. The idea behind this innovation is betrayed by the name: the ID fields are removed from the sector format. Instead of labeling each sector within the sector header itself, **a format map is stored in memory and referenced when a sector must be located**. This map also contains information about what sectors have been marked bad and relocated, where the sectors are relative to the location of servo information, and so on. Not only does this improve format efficiency, allowing up to 10% more data to be stored on the surface of each platter, it also improves performance. Since this critical positioning information is present in high-speed memory, it can be accessed much more quickly. "Detours" in chasing down remapped sectors are also eliminated.

Formatting and Capacity

Most PC users are familiar with the concept that a hard disk--in fact, all storage media--must be formatted before it can be used. There is usually some confusion, however, regarding exactly what formatting means and what it does. This is exacerbated by the fact that modern hard disks are not formatted in the same way that older ones were, and also the fact that the utilities used for formatting behave differently when acting on hard disks than when used for floppy disks.

This section takes a look at issues surrounding disk formatting and capacity, discusses unformatted and formatted hard disk capacity, and looks briefly at formatting utilities.

Two Formatting Steps

Many PC users don't realize that formatting a hard disk isn't done in a single step. In fact, three steps are involved:

1. **Low-Level Formatting:** This is the "true" formatting process for the disk. It creates the physical structures (tracks, sectors, control information) on the hard disk. Normally, this step begins with the hard disk platters "clean", containing no information.
2. **Partitioning:** This process divides the disk into logical "pieces" that become different hard disk volumes (drive letters). This is an operating system function (fdisk).
3. **High-Level Formatting:** This final step is also an operating-system-level command. It defines the logical structures on the partition and places at the start of the disk any necessary operating system files.

As you can see, two of the three steps are "formatting", and this dual use of the word is a big part of what leads to a lot of confusion when the term "formatting" is used. Another strange artifact of history is that the DOS "FORMAT" command behaves differently when it is used on a hard disk than when it is used on a floppy disk. Floppy disks have simple, standard geometry and cannot be partitioned, so the FORMAT command is programmed to automatically both low-level and high-level format a floppy disk, if necessary. For hard disks, however, FORMAT will only do a high-level format. Low-level formatting is performed by the controller for older drives, and at the factory for newer drives.

Low-Level Formatting

Low-level formatting is the process of outlining the positions of the tracks and sectors on the hard disk, and writing the control structures that define where the tracks and sectors are. This is often called a "true" formatting operation, because it really creates the physical format that defines where the data is stored on the disk. The first time that a low-level format ("LLF") is performed on a hard disk, the disk's platters start out empty. That's the last time the platters will be empty for the life of the drive. If an LLF is done on a disk with data on it already, the data is permanently erased (save heroic data recovery measures which are sometimes possible).

If you've explored other areas of this material describing hard disks, you have learned that modern hard disks are much more precisely designed and built, and much more complicated than older disks. Older disks had the same number of sectors per track, and did not use dedicated controllers. It was necessary for the external controller to do the low-level format, and quite easy to describe the geometry of the drive to the controller so it could do the LLF. Newer disks use many complex internal structures, including zoned bit recording to put more sectors on the outer tracks than the inner ones, and embedded servo data to control the head actuator. They also transparently map out bad sectors. Due to this complexity, all modern hard disks are low-level formatted at the factory for the life of the drive. There's no way for the PC to do an LLF on a modern IDE/ATA or SCSI hard disk, and there's no reason to try to do so.

Older drives needed to be re-low-level-formatted occasionally because of the thermal expansion problems associated with using stepper motor actuators. Over time, the tracks on the platters would move relative to where the heads expected them to be, and errors would result. These could be corrected by doing a low-level format, rewriting the tracks in the new positions that the stepper motor moved the heads to. This is totally unnecessary with modern voice-coil-actuated hard disks.

Warning: You should never attempt to do a low-level format on an IDE/ATA or SCSI hard disk. Do not try to use BIOS-based low-level formatting tools on these newer drives. It's

unlikely that you will damage anything if you try to do this (since the drive controller is programmed to ignore any such LLF attempts), but at best you will be wasting your time.

High-Level Formatting

After low-level formatting is complete, we have a disk with tracks and sectors--but nothing written on them. *High-level formatting* is the process of writing the file system structures on the disk that let the disk be used for storing programs and data. If you are using DOS, for example, the DOS format command performs this work, writing such structures as the master boot record and file allocation tables to the disk. High-level formatting is done after the hard disk has been partitioned, even if only one partition is to be used. See here for a full description of DOS structure, also used for Windows 3.x and Windows 9x systems.

The distinction between high-level formatting and low-level formatting is important. It is not necessary to low-level format a disk to erase it: a high-level format will suffice for most purposes; by wiping out the control structures and writing new ones, the old information is lost and the disk appears as new. (Much of the old data is still on the disk, but the access paths to it have been wiped out.)

Different operating systems use different high-level format programs, because they use different file systems. However, the low-level format, which is the real place where tracks and sectors are recorded, is the same.

Seek Time

The *seek time* of a hard disk measures the amount of time required for the read/write heads to move between tracks over the surfaces of the platters. Seek time is one of the most commonly discussed metrics for hard disks, and it *is* one of the most important positioning performance specifications. However, using this number to compare drives can be somewhat fraught with danger. Alright, that's a bit melodramatic; nobody's going to get hurt or anything. :^) Still, to use seek time properly, we must figure out exactly what it means.

Switching between tracks requires the head actuator to move the head arms physically, which being a mechanical process, takes a specific amount of time. The amount of time required to switch between two tracks depends on the distance between the tracks. However, there is a certain amount of "overhead" involved in track switching, so the relationship is not linear. It does not take double the time to switch from track 1 to track 3 that it does to switch from track 1 to track 2, much as a trip to the drug store 2 miles away does not take double the time of a trip to the grocery store 1 mile away, when you include the overhead of getting into the car, starting it, etc.

Seek time is normally expressed in milliseconds (commonly abbreviated "msec" or "ms"), with average seek times for most modern drives today in a rather tight range of 8 to 10 ms. Of course, in the modern PC, a millisecond is an *enormous* amount of time: your system memory has speed measured in nanoseconds, for example (one million times smaller). A 1 GHz processor can (theoretically) execute over one million instructions in a millisecond! Obviously, even small reductions in seek times can result in improvements in overall system performance, because the rest of the system is often sitting and waiting for the hard disk during this time. It is for this reason that seek time is usually considered one of the most important hard disk performance specifications. Some consider it the most important.



At one point many years ago seek times were difficult to use because manufacturers wouldn't agree on a standardized way of reporting them. Today, this has largely been corrected. While seek time is usually given as a single number, in fact there are three different seek time specifications you should examine for a drive, as they represent the drive's performance when doing different types of seeks:

- **Average:** As discussed, this is meant to represent an average seek time from one random track (cylinder) to any other. This is the most common seek time metric, and is usually 8 to 10 ms, though older drives had much higher numbers, and top-of-the-line SCSI drives are now down to as low as 4 ms!
- **Track-to-Track:** This is the amount of time that is required to seek between adjacent tracks. This is similar in concept (but not exactly the same as) the track switch time and is usually around 1 ms. (Incidentally, getting this figure without at least two significant digits is pretty meaningless; don't accept "1 ms" for an answer, get the number after the decimal point! Otherwise every drive will probably round off to "1 ms".)
- **Full Stroke:** This number is the amount of time to seek the entire width of the disk, from the innermost track to the outermost. This is of course the largest number, typically being in the 15 to 20 ms range. In some ways, combining this number with the average seek time represents the way the drive will behave when it is close to being full.

While I believe that seek time is a very important specification, I have become somewhat cynical in the last few years regarding the amount of attention paid to it. The reason is that there is so little difference between the seek time specs of most comparable drives in any given class or category. For example, almost all IDE/ATA 7200 RPM drives shipping in 2000 had an average seek time specification of 8.0, 8.5 or 9.0 milliseconds. This doesn't leave a lot to work with. However, at the same time, we must realize that of the four components that comprise the drive's access time, if you are comparing two drives of the same class and spindle speed, only seek time will differ much between them. So this small differential may be the only thing to distinguish drives; and small differences are what you are likely to see. (Larger discrepancies though, directly translate into often *very* noticeable differences in performance. A drive with a 5 ms seek time will generally blow the doors off one with a seek time of 8.0 to 9.0 ms in random positioning tasks, which is why these fast drives are preferred for servers and other multi-user environments.)

To really put seek time in proper context, it should be remembered that it is the largest component of access time, which is the composite metric that best represents positioning performance. However, it is only one component, and there is at one that is of at least equal importance. Also, bear in mind that seek times are *averages* that make certain assumptions of how the disk will be used.

Note: Some manufacturers include settle time as part of their seek time specification. Since settle time is relatively small this doesn't really change the seek time numbers much.

Settle Time

The *settle time* specification (sometimes called *settling time*) refers to the amount of time required, after the actuator has moved the head assembly during a seek, for the heads to

stabilize sufficiently for the data to begin to be read. Since it is a component of access time and therefore part of the time required to position for reading a random file, I include it here for completeness. However, since settle time is usually so short (typically less than 0.1 msec) it is dwarfed in importance by seek time and rotational latency, and differences between drives in this regard are not really significant. Some manufacturers do not even bother to specify settle time, and some just lump it in with seek time.

Settle time, like seek time, is a function of the drive's actuator characteristics.

Command Overhead Time

Command overhead refers to the time that elapses from when a command is given to the hard disk until something actually starts happening to fulfill the command. In a way, it's sort of like a "reaction time" for the disk. Consider when you're driving a car and a streetlight suddenly turns red; your "command overhead" is the time that elapses from when the light changes, until your foot starts to move toward the brake pedal.

Like settle time, command overhead is a component of access time and thus part of the overall equation of random positioning performance. Also like settle time, it is generally very small and not highly variable between drive designs; it is generally around 0.5 ms for pretty much all modern drives and therefore not something that requires a lot of attention. Also like settle time, it is sometimes not even specified separately from seek time but rather "lumped in" with it. It is dominated by seek time and rotational latency in the overall positioning performance picture.

Command overhead is **influenced primarily by the design of the disk's integrated controller**, and to some extent, the nature of the interface used (which of course is a major influence on the design of the controller!)

Latency

The hard disk platters are spinning around at high speed, and the spin speed is not synchronized to the process that moves the read/write heads to the correct cylinder on a random access on the hard disk. Therefore, at the time that the heads arrive at the correct cylinder, the actual sector that is needed may be anywhere. After the actuator assembly has completed its seek to the correct track, the drive must wait for the correct sector to come around to where the read/write heads are located. This time is called *latency*. Latency is directly related to the spindle speed of the drive and such is influenced solely by the drive's spindle characteristics.

Conceptually, latency is rather simple to understand; it is also easy to calculate. The faster the disk is spinning, the quicker the correct sector will rotate under the heads, and the lower latency will be. Sometimes the sector will be at just the right spot when the seek is completed, and the latency for that access will be close to zero. Sometimes the needed sector will have just passed the head and in this "worst case", a full rotation will be needed before the sector can be read. On average, latency will be half the time it takes for a full rotation of the disk. This table shows the latency for the most common hard disk spindle speeds:

Spindle Speed (RPM)	Worst-Case Latency (Full Rotation) (ms)	Average Latency (Half Rotation) (ms)
3,600	16.7	8.3
4,200	14.2	7.1
4,500	13.3	6.7
4,900	12.2	6.1
5,200	11.5	5.8
5,400	11.1	5.6
7,200	8.3	4.2
10,000	6.0	3.0
12,000	5.0	2.5
15,000	4.0	2.0

The "average" value is almost always the one provided as a specification for the drive; sometimes the "worst case" number is also mentioned. Sometimes latency is not even mentioned specifically at all, but it can always be calculated using this formula:

$$1 \text{ min} = 60 \text{ s} = 60\,000 \text{ ms}$$

Full rotation:

$$60\,000 : 3\,600 = 16,7 \text{ ms}$$

$$\text{Half rotation} : \text{Full rotation} : 2 = 8,3$$

Which factors down to this formula:

$$30000 / \text{SpindleSpeed}$$

The result is a value in milliseconds.

In looking at the table above, notice that the first increases in spindle speed yielded the greatest percentage improvements in performance. As speeds continue to increase, there are diminishing returns for the extra RPMs. Going from 5,400 RPM to 7,200 RPM shaved 1.4 milliseconds off the average latency, but going from 7,200 to 10,000 (which is a bigger jump in both absolute and percentage terms) only reduces it 1.2 milliseconds. At some point companies will likely "max out" on spindle speeds because there won't be any point in increasing further, especially considering the problems that are created when speeds are increased. The 12,000 speed introduced by the Hitachi Pegasus, while very fast, never really caught on as an industry standard. It looks like 15,000 RPM will be the next standard spindle speed for top-of-the-line SCSI drives. It has yet to be seen what price will be paid for jumping to such a high spindle speed; the improvement in latency over standard 10,000 RPM drives is "only" 1.0 milliseconds. As with seek time, figures in milliseconds are big numbers when dealing with computer system performance, but to shave another 1.0 ms off latency from 15,000 RPM would require going to 30,000 RPM, which would be a very significant

engineering challenge probably not justified by shaving 1.0 ms off the total access time for the drive.

Again, as with seek times, latency is most relevant only to certain types of accesses. For multiple, frequent reads of random sectors on the disk, it is an important performance-limiting factor. For reading large continuous blocks of data, latency is a relatively minor factor because it will only happen while waiting to read the first sector of a file. The use of cylinder and head skewing on modern drives is intentionally designed to reduce latency considerations when switching between consecutive heads or cylinders on long sequential reads or writes.

Access Time

Access time is the metric that represents the composite of all the other specifications reflecting random performance positioning in the hard disk. As such, it is the best figure for assessing overall positioning performance, and you'd expect it to be the specification most used by hard disk manufacturers and enthusiasts alike. Depending on your level of cynicism then, you will either be very surprised, or not surprised much at all, to learn that it is rarely even discussed.

Perhaps the problem is that access time is really a derived figure, comprised of the other positioning performance specifications. The most common definition is:

Access Time = Command Overhead time + Seek Time + Seek Settle + Latency

Unfortunately, this definition is not universal, and is made complicated by the fact that manufacturers refuse to standardize on even what access time's subcomponents mean. Some companies incorporate settle time into seek time, some don't, for example. And to make matters worse, some companies use the term "access time" to mean "seek time"! They really are not the same thing at all.

In the end though, when you are looking at the ability of a drive to randomly position, access time is the number you want to look at. Since command overhead and settle time are both relatively small and relatively similar between drives, that leaves the sum of seek time and latency as the defining characteristic between drives. Seek time and latency are a result of very different drive performance factors--seek time being primarily a matter of the actuator and latency the spindle motor --resulting in the possibility of some drives being better in one area and worse in another. In practice, high-end drives with faster spindles usually have better seek times as well since these drives are targeted to a performance-sensitive market that wouldn't buy a drive with slow seek time.

Let's compare a high-end, mainstream IDE/ATA drive, the Maxtor DiamondMax Plus 40, to a high-end, mainstream SCSI drive, the IBM Ultrastar 72ZX. (When I say "high end" I mean that the drives are good performers, but neither drive is the fastest in its interface class at the time I write this.) The Maxtor is a 7200 RPM drive with a seek time spec of "< 9.0 ms", which means 9 ms. Its sum of its seek time and latency is about 13.2 ms. The IBM is a 10,000 RPM drive with a seek time spec of 5.3 ms. Its sum of seek time and latency is about 8.3 ms. This difference of 5 ms represents an enormous performance difference between these two drives, one that would be readily apparent to any serious user of the two drives.

As you can see, the Cheetah beats the DiamondMax on both scores, seek time and latency. When comparing drives of a given class, say, IDE/ATA 7200 RPM drives, they will all have

the same latency, which means, of course that the only number to differentiate them is seek time. Comparing the Maxtor above to say, the Seagate Barracuda ATA II with its 8.2 ms seek time shows a difference of 0.8 ms, or around 10%. But the proper comparison includes the other components of access time. So the theoretical access time of the Maxtor drive is about 13.7 ms (including 0.5 ms for command overhead) and that of the Seagate Barracuda drive 12.9. The difference now is about 6%. Is that significant? Only you can judge, but you also have to remember that even access time is only one portion of the overall performance picture.

Remember that access time is an average figure, comprised of other averages. In fact, access time on any particular read or write can vary greatly. For an illustration, let's consider the IBM 34GXP drive, look at its minimums and maximums, and see how they translate into access time minimums and maximums:

Attribute	Best-Case Figure (ms)	Worst-Case Figure (ms)
Command Overhead	0.5	0.5
Seek Time	2.2	15.5
Settle Time	<0.1	<0.1
Latency	0.0	8.3
Total	2.8	28.4

As you can see, there's quite a range! In the real world these extremes will rarely occur, and over time will be "averaged out" anyway, which is the reason that average figures are used. However, it's important to remember that this wide range can occur on any given access, and random perturbations can affect benchmarks and other performance tests.

Transfer Performance Specifications

Since the obvious objective in using a hard disk is to transfer data to the hard drive and onto the disks, or off the disks and out of the drive, the rate of data transfer is of paramount importance. Traditionally, real transfer rate metrics have been very underrated and given almost no attention compared to positioning specifications like seek time. The only transfer specification that is really commonly mentioned is the speed of the interface, which is actually the *least* important indicator of overall disk performance.

Before we look at transfer specifications, we need to have a short word about terminology. :^) Transfer rates are confusing in part because of the phrase "transfer rate" can mean so many different things. Data transfer occurs in two main steps. For a read, data is first read from the disk platters by the heads and transferred to the drive's internal buffer; then it is moved from the buffer, over the interface, to the rest of the system. For a write, the process is reversed. The rate that transfer occurs within the disk is of course the *internal* transfer rate; the rate that transfer occurs over the interface is the *external* transfer rate. They are usually not the same, and in some cases can differ by an order of magnitude.

Internal transfer rates are further broken down into the media transfer rate and the sustained transfer rate, and further complicating things is the fact that transfer rates are not constant



over the surface of the drive. It sounds impossible to get a handle on, but it's not that bad once you place it all in the proper context and perspective, and that's exactly what we will do in this section.

Tip: Whenever you are reading a spec sheet, or discussing transfer rates with someone, be sure to find out exactly *what* transfer rate is being discussed. By itself the term "transfer rate" is meaningless.

Internal Media Transfer Rate

The *internal media transfer rate* of a drive (often just called the *media transfer rate* or the *media rate*) refers to the actual speed that the drive can read bits from the surface of the platter, or write bits to the surface of the platter. It is normally quoted in units of megabits per second, abbreviated Mbit/sec or Mb/s. Typical values for today's drives are in the hundreds of Mb/s, with a maximum media rate of about 500 Mb/s being high-end at the time of this writing.

Media transfer rate can be confusing to understand even for the serious hard disk enthusiast; it's equally difficult to describe. :^) For starters, let's explain what it is *not*. It is only related to what is going on inside the hard disk, and therefore has nothing directly to do with the interface transfer rate. It refers *only* to the speed of reading or writing bits to a *single* track of one surface of the disk. Nothing else is included--no positioning, no track or head switching. A track holds a relatively small amount of data--under 0.25 MB with current technology. This means that almost no real-world reads or writes occur on a single track except for very short files, and the performance when reading those is primarily limited by positioning, not transfer. The end result of this is that the media transfer rate does not have much relevance to real-world use of a drive. It is primarily a "theoretical" specification that illustrates the state of the drive's technology. It is used almost exclusively for comparing drives against each other.

Media transfer rates are not constant across the entire surface of a platter. Let's recall for a moment the fact that modern disk drives use zoned bit recording. This is done because the length of the inner tracks on the disk is much shorter than that of the outer tracks. ZBR allows the outer tracks to have more sectors per track than the inner tracks. However, since every track is spinning at the same speed, this means that when reading the outer tracks, the disk is transferring more data per second when reading the inner tracks. For this reason, the media transfer rate decreases as you move from the outer tracks of the disk to the inner ones.

The explanation above is the reason that there is no single "media transfer rate" figure for a modern hard disk. They are typically stated as a range, from minimum to maximum (with the maximum figure given alone, of course, if only one number is provided). For example, the IBM Deskstar 34GXP (model DPTA-373420) has a media transfer rate of between approximately 171 Mb/s and 284 Mb/s depending where on the disk you are reading: that drive has 12 different zones. This drive has 272 sectors in its innermost zone, and 452 sectors on its outside tracks.

Another important thing to remember about the media transfer rate (and another reason why it is a theoretical measure only) is that it includes *all* bits read or written to the disk, not just user data. As discussed in detail here, , some of the data storage space in a sector is reserved for overhead. This means that you cannot assume that the media rate represents the rate at which


user data can be read from the disk. Taking the IBM drive above again as an example, its maximum media transfer rate is 284 Mb/s, but the maximum rate that the drive can read user data is about 222 Mb/s in the outside zone.

It's not really feasible to calculate the media transfer rate from other drive specifications, because manufacturers typically do not publish details of their sector format and other pertinent overhead characteristics. The best that you can do is approximate the value by looking at the rate at which user data can be streamed from a given part of the disk. To do so, we need to know how much data is able to pass under the read/write heads in one second. This is dependent on the density of the data (how tightly packed the data is into each linear inch of disk track), and also how fast the disk is spinning. The density of the data can be calculated easily if we know how many sectors are on the track, since we know how many bytes of user data there are in a sector (512). The speed of the disk is calculated in RPM, so we divide it by 60 to get revolutions per second. This gives us a calculation of the data transfer rate in megabits per second as follows (to get the result in *megabytes* per second, simply divide by 8):

Conclusion:

The media transfer rate of the drive is primarily affected by all of the various **data recording and encoding factors**, as well as the **size of the platters**, and the drive's **spindle speed**. In addition, the drive's controller must be fast enough to be able to handle the fastest rate that the disk can read or write, but manufacturers ensure that this is never an issue by beefing up their controllers where necessary.

Head Switch Time

 Each cylinder contains a number of tracks, each accessible by one of the heads on the drive (one head per surface). To improve efficiency, the drive will normally use all of the tracks in a cylinder before going to the next cylinder when doing a sequential read or write; this saves the time required to physically move the heads to a new cylinder. Switching between heads is a purely electronic process instead of a mechanical one. However, switching between heads within a cylinder still requires a certain amount of time, called the *head switch time*. This is usually less than the track switch time, and is usually on the order of 1 to 2 milliseconds. (Seems kind of slow for an electronic process, doesn't it? The reason is that this time includes all of the overhead of the switch as well; it is all of the time that passes between when the read stops on one head and when it actually starts again on the next one.)

Cylinder Switch Time

Cylinder switch time is the time that elapses when the drive finishes reading (or writing) all the data on a given cylinder and needs to switch to the next one. This normally only occurs during fairly long reads or writes, since the drive will read all the tracks in a cylinder before switching cylinders. Cylinder switch time is slower than head switch time because it involves a mechanical process: moving the actuator assembly. It is usually somewhere around 2 to 3 milliseconds.

Note: You might think that cylinder switch time would be the same as track-to-track, after all, it's the same thing, isn't it? They aren't the same however, because cylinder switch time includes all of the overhead time that passes from the time the read stops on one track until it

starts again on the next one. This is why cylinder switch times are typically double those of track-to-track seeks.

Cylinder switch time is influenced by the characteristics of the hard disk's controller as well as its actuator mechanics. It does not vary greatly from drive model to model or between manufacturers.