

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

Introduction

This paper will assume the hard drive has already been correctly installed, with all the cables and power supplies attached where they were meant to be. It will also assume that the hard drive is on a newly purchased system, or one that has just received a new hard drive, or an additional (second) hard drive.

Disk Formatting

When formatting a hard disk, two distinct types of formatting must be considered:

- **Physical, or low level formatting (LLF)**
- **Logical, or high level formatting (HLF)**

When a floppy disk is formatted, the **DOS FORMAT** command performs both kinds of formats simultaneously. To format a hard disk, however, both operations must be performed separately. Additionally, the hard disk requires a third step which is performed between the two formatting processes. This step is called **partitioning**, where the partitioning information is written to the hard disk. Partitioning is required because a hard disk is designed to be used with more than one operating system. As you will also see in this paper, there are many other good reasons to partition a hard drive.

Separating the physical format in a way which is always the same, regardless of the operating system being used, and regardless of the high level format which would be different for each operating system installed on the hard disk, makes possible the use of multiple operating systems on one hard drive. The partitioning step allows more than one type of operating system to use a single hard drive or a single DOS to use the disk as several volumes or logical drives. A volume or logical drive is anything to which DOS assigns a drive letter. Consequently, formatting a hard disk involves three separate steps:

1. **Low Level Formatting (LLF)**
2. **Partitioning**
3. **High Level Formatting (HLF)**

The Low Level Format

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

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

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

drives can have practically any number of sectors per track, depending on what controller design was used.

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

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

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

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

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

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

Low Level Formatting is normally done by the manufacturer of the hard disk controller. Consequently, when a new hard drive is purchased, it has normally already been low level formatted, and is therefore not a necessary function for the purchaser to perform.

Legacy Hard Disk Drive Configuration

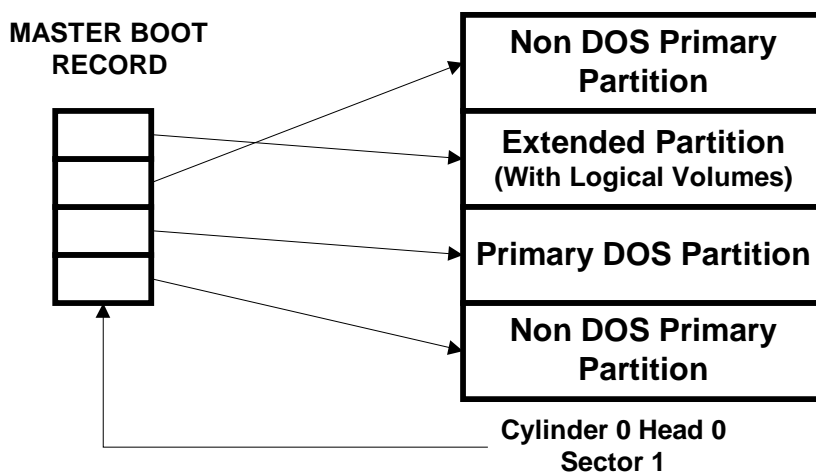
Mark E. Donaldson

Partitioning

Partitioning segments a hard disk into areas, called **partitions**, that can hold a particular operating system's file system. There are three commonly used operating system file systems used today:

- **FAT (File Allocation Table).** This system is the standard file system used by DOS, OS/2, and Windows NT. FAT partitions support the file names of 11 characters maximum (8 plus a 3 character extension), and a volume can be as large as 2G in size.
- **HPFS (High Performance File System).** This UNIX type file system is assessable only under OS/2 and Windows NT. DOS applications running under OS/2 or Windows NT can access files in HPFS partitions but straight DOS cannot. File names can be 256 characters long, and volume size is limited to 8G.
- **NTFS (Windows NT File System).** This UNIX type file system currently is accessible only under Windows NT., but drivers should be available for OS/2 to access NTFS as well. DOS cannot access these partitions, but DOS applications running under Windows NT can. File names can be 256 characters long, and volumes size is limited to 8G. NTFS capabilities may be added to Windows 95 in the future.

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



The term **cluster** was changed to allocation unit in DOS 4.0. The newer term is appropriate because a single cluster is the smallest unit of the disk that DOS can allocate when it writes a file. A cluster is equal to one or more sectors, and although a cluster can be a single sector in some cases (specifically in 1.2 M and 1.44 M floppies), it is usually more than one. Having more than one sector per cluster reduces the size and processing overhead of the FAT and enables DOS to run faster because it has fewer individual units of the disk to manage. The tradeoff is in wasted disk space.

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

Because DOS can manage space only in full cluster units, every file consumes space on the disk in increments of one cluster.

Smaller clusters generate less slack (space wasted between the actual end of each file and the end of the cluster). With larger clusters, the wasted space grows larger. For hard disks, the cluster size varies with the size of the partition. Table I shows the default cluster sizes DOS selects for a particular partition volume size.

| TABLE I DRIVE SIZES & CLUSTER SIZES | | | |
|----------------------------------------|-----------------|-------------------------|----------|
| Disk Type Partition Size | Sectors/Cluster | Cluster Size (Bytes) | FAT Type |
| Single Sided Floppy | 1 | 512 | |
| Double Side Floppy | 2 | 1024 | |
| 360 Mb Floppy (5.25) | 2 | 1024 | |
| 1.2 M Floppy (5.25) | 1 | 512 | |
| 720 Kb Floppy (3.5) | 2 | 1024 | |
| 1.44 Mb Floppy (3.5) | 1 | 512 | |
| 2.88 Mb Floppy (3.5) | 2 | 1024 | |
| 0 to 15 Mb Logical | 8 | 4096 | 12 bit |
| 16 to 127 Mb Logical | 4 | 2048 | 16 bit |
| 128 to 255 Mb Logical | 8 | 4096 | 16 bit |
| 256 to 512 Mb Logical | 16 | 8192 | 16 bit |
| 513 to 1024 Mb Logical | 32 | 16384 | 16 bit |
| 1025 to 2049 Mb Logical | 64 | 32768 | 16 bit |
| Compressed Logical | 16 | 8192 | 16 bit |

In most cases, these cluster sizes, which are selected by **DOS FORMAT** command, are the minimal possible for a given partition size. Therefore, 8K clusters are the smallest possible for a partition size greater than 256M. DOS creates a FAT using 12 bit numbers if the partition is 16M or less, while all other FATs are created using 16 bit numbers.

The effect of the larger cluster sizes on larger disks partitions can be substantial. A drive partition over 512M and up to 1G (16 clusters) containing about 5,000 files, with average slack of one half of the last cluster used for each file (one half of 16K), wastes about 40M [5000 x (.5 x 16K)] of file space on a disk set up with IBM or MS DOS. If you were to repartition the drive into two separate partitions of less than or equal to 512M each, then the cluster size would be cut in half as would be the wasted slack space. After restoring your files, you would end up with approximately 20M more disk space free. The tradeoff is that managing multiple partitions is not as convenient as a single partition. The only way you can control cluster or allocation unit sizing is by changing the sizes of the partitions.

Because the NTFS, HPFS, and the new FAT32 (32 bit FAT system) offer more allocation unit numbers, the allocation unit sizes are smaller, often a single sector. This dramatically reduces the slack space but increases file management overhead because many more allocation units must be managed.

Despite the problem with slack space, the FAT file system is the most recommended for compatibility reasons. For example, few applications currently are compatible with the longer file names possible

Legacy Hard Disk Drive Configuration

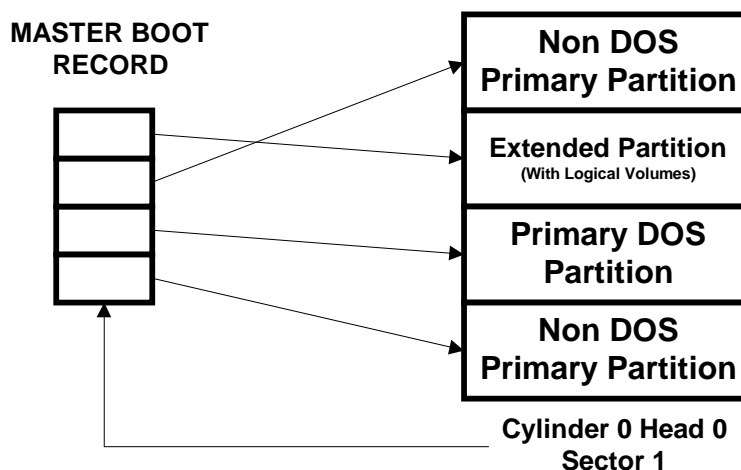
Mark E. Donaldson

in the HPFS and NTFS file systems. All the operating systems can access FAT volumes, and the file structures and data recovery procedures are well known. Data recovery can be difficult to impossible under the HPFS and NTFS systems.

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

NOTE: ALL DRIVES MUST BE PARTITIONED EVEN IF THERE IS ONLY ONE SINGLE PRIMARY DOS PARTITION.

For additional and more detailed information on hard disk partitioning, refer to the white paper I wrote entitled *FDISK*.



High Level Format

During the high level format, the operating system (such as DOS, OS/2, or Windows NT) writes the structures necessary for managing files and data. FAT partitions have a **Volume Boot Sector (VBS)**, a File Allocation Table (FAT), and a root directory on each formatted logical drive. These data structures enable the operating system to manage the space on the disk, keep track of files, and even manage defective areas so that they do not cause problems.

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

To format a DOS logical volume or partition, the following syntax would be used:

```
C:\FORMAT C: (or drive letter to be formatted) /S
```

Revised December 26, 2009

Page 5 of 27

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

Master Boot Record

When you turn on your PC, the processor attempts to begin the process of processing data. But, since the system memory is empty, the processor doesn't really have anything to execute, or even begin to know where to look for it. To ensure that the PC will always boot regardless of the BIOS code, both chip and BIOS manufacturers developed their code so that the processor, once turned on, always starts executing at the same place, FFFF0h.

Similarly, every hard disk must have a consistent "starting point" where key information is stored about the disk, such as the number of partitions and what type they are. There also must be someplace where the BIOS can load the initial boot program that starts the process of loading the operating system. The place where this information is stored is called the master boot record (MBR), also referred to as the master boot sector or even just the boot sector. Do not confuse the master boot sector with volume boot sectors, which are indeed different.

The master boot record is always located at cylinder 0, head 0, and sector 1, the first sector on the disk. This is the consistent starting point that the disk will always use. When a computer starts and the BIOS boots the machine, it will always look at this first sector for instructions and information on how to proceed with the boot process and load the operating system. The master boot record contains the following structures:

- **Master Partition Table:** This small bit of code that is referred to as a table contains a complete description of the partitions that are contained on the hard disk. When the developers designed the size of this master partition table, they left just enough room for the description of four partitions, hence the four partition (four physical partitions) limit. For this reason, and no other, a hard disk may only have four true partitions, also called primary or physical partitions. Any additional partitions must be logical partitions that are linked to (or are part of) one of the primary partitions. One of these partitions is marked as active, indicating that it is the one that the computer should use to continue the boot process.
- **Master Boot Code:** The master boot record is the small bit of computer code that the BIOS loads and executes to start the boot process. This code, when fully executed, transfers control to the boot program stored on the boot (active) partition to load the operating system.

If an active partition is found, regardless of where it is found, that partition's boot record is read into memory at 0000:7c00 and then the MBR code is added to 0000:7c00 pointing to the partition table entry that describes the partition being booted. The boot record program uses this data to determine the drive being booted from and the exact location of the partition on the disk. If no active partition table entry can be found, the boot process enters ROM BASIC via INT 18. All other errors will cause the system to hang or stop.

Notes

1. The first byte of an active partition table entry is 80. This byte is loaded into the DL register before INT 13 is called to read the boot sector. When INT 13 is called, DL is the BIOS device number. Because of this, the boot sector read by this MBR program can only be read from BIOS device number 80 (the first hard disk). This is one of the reasons why it is usually not possible to boot from any other hard disk. Of course, there are exceptions to every rule, but those are beyond the scope of this exercise.

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

- The MBR program uses the CHS based INT 13H AH=02H call to read the boot sector of the active partition. The location of the active partition's boot sector is in the partition table entry in CHS format. If the drive is 528MB, this CHS must be a translated CHS. No addresses in LBA form are used (another reason why LBA does not solve the 528MB problem).

Making A Boot Disk

There are three primary ways in which a boot disk, or bootable floppy disk, can be made. A boot disk can be made by using the following three commands:

- **FORMAT /S**
- **SETUP /F**
- **SYS**

Assuming the floppy disk that is to become the boot disk is in **Drive A:**, and the boot disk is to be made from **Drive C:**, the syntax for using these three commands would be as follows:

```
C:\FORMAT A:/S
C:\SETUP A:/F
C:\SYS A:
```

Table I below shows the files that will be copied (or transferred) to the new boot disk using the FORMAT /S and SETUP/F commands with the various sized floppy disk media.

| TABLE I MAKING A BOOT DISK | | | | | |
|-------------------------------|-----------------------|------------------|------------------|--------------------|---------------------|
| | FORMAT /S Any Size | SETUP /F 360k | SETUP /F 720K | SETUP /F 1.2 MB | SETUP /F 1.44 MB |
| IO.SYS | YES | YES | YES | YES | YES |
| MSDOS.SYS | YES | YES | YES | YES | YES |
| COMMAND.COM | YES | YES | YES | YES | YES |
| FDISK.EXE | | YES | YES | YES | YES |
| SYS.COM | | YES | YES | YES | YES |
| CHKDSK.EXE | | YES | YES | YES | YES |
| EXPAND.EXE | | YES | YES | YES | YES |
| ATTRIB.EXE | | YES | YES | YES | YES |
| DEBUG.EXE | | YES | YES | YES | YES |
| MEM.EXE | | YES | YES | YES | YES |
| FORMAT.COM | | YES | YES | YES | YES |
| MSD.COM | | | YES | YES | YES |
| MSD.EXE | | | YES | YES | YES |
| DBLSPACE.EXE | | | YES | YES | YES |
| XCOPY.EXE | | | YES | YES | YES |
| EDIT.COM | | | | YES | YES |
| QBASIC.EXE | | | | YES | YES |
| UNDELETE.EXE | | | | YES | YES |
| SCANDISK.EXE | | | | YES | YES |

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

Using SYS.COM

To repair DOS Volume Boot Sector

```
SYS C:  
format /mbr
```

Potential DOS Upgrade Problems

You already know that the DOS system files have special placement requirements on a hard disk. Sometimes these special requirements cause problems when you are upgrading from one version of DOS to another.

If you have attempted to upgrade a PC system from one version of DOS to another, you know that you use the **DOS SYS** command to replace old system files with new ones. The SYS command copies the existing system files (stored on a bootable disk with hidden, system, and read-only attributes) to the disk in the correct position and with the correct names and attributes. The **COPY** command does not copy hidden or system files (nor would it place the system files in the required positions on the destination disk if their other attributes had been altered so that they could be copied using COPY).

In addition to transferring the two hidden system files from one disk to another, SYS also updates the DOS volume boot sector on the destination disk so that it is correct for the new version of DOS. Common usage of the SYS command is as follows:

```
SYS C: (for drive C) or SYS A: (to make a floppy in drive A bootable)
```

The syntax of the command is as follows:

```
SYS[d.-][path]d:
```

In this command line, *dlpath* specifies an optional source drive and path for the system files. If the source drive specification is omitted, the boot drive is used as the source drive. This parameter is supported in DOS 4.0 and later versions only. Versions of DOS older than 4.0 automatically look for system files on the default drive (not on the boot drive). The *d* in the syntax specifies the drive to which you want to transfer the system files.

When the SYS command is executed, you usually are greeted by one of two messages:

```
System transferred or No room for system on destination disk
```

If a disk has data on it before you try to write the system files to it, the SYS command from DOS versions 3.3 and earlier probably will fail because they are not capable of moving other files out of the way. The SYS command in DOS 4.0 and higher versions rarely fail because they can and do move files out of the way.

Some users think that the cause of the No room message on a system that has an older version of DOS is that the system files in any newer version of DOS are always larger than the previous version, and that the new version files cannot fit into the space

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

allocated for older versions. Such users believe that the command fails because this space cannot be provided at the beginning without moving other data away. This belief is wrong. The SYS command fails in these cases because you are trying to install a version of DOS that has file names different from the names already on the disk. There is no normal reason for the SYS command to fail when you update the system files on a disk that already has them.

Although the belief that larger system files cannot replace smaller ones might be popular, it is wrong for DOS 3.0 and later versions. The system files can be placed virtually anywhere on the disk, except that the first clusters of the disk must contain the file IO.SYS (or its equivalent). After that requirement is met, the IO.SYS file might be fragmented and placed just about anywhere on the disk, and the SYS command implements it with no problems whatsoever. In version 3.3 or later, even the IO.SYS file can be fragmented and spread all over the disk, as long as the first cluster of the file occupies the first cluster of the disk (cluster 2). The only other requirement is that the names IO.SYS and IO.SYS (or their equivalents) must use the first and second directory entries.

DOS 4.0 and Later Versions

Under DOS 4.0 and later versions, the SYS command is much more powerful than under previous versions. Because the system files must use the first two entries in the root directory of the disk as well as the first cluster (cluster 2) of the disk, the DOS 4.0 and later versions' SYS command moves any files that occupy the first two entries but that do not match the new system file names to other available entries in the root directory; the SYS command also moves the portion of any foreign file occupying the first cluster to other clusters on the disk. Whereas the SYS command in older versions of DOS would fail and require a user to make adjustments to the disk,, the DOS 4.0 and later versions' SYS command automatically makes the required adjustments. For example, even if you are updating a Phoenix DOS 3.3 disk to IBM DOS 4.0, the IBM DOS SYS command relocates the Phoenix IO.SYS and MSDOS.SYS files so that the new IO.SYS and IO.SYS files can occupy the correct locations in the root directory as well as on the disk.

The SYS command in DOS versions 5.0 and 6.0 go one step farther: They replace old system files with the new ones. Even if the old system files had other names, DOS 5.0 and higher ensure that they are overwritten by the new system files. If you are updating a disk on which the old system file names match the new ones, the SYS command of any version of DOS overwrites the old system files with the new ones with no moving of files necessary. With the enhanced SYS command in DOS 4.0 and later versions, it is difficult to make a DOS upgrade fail.

Using FDISK /MBR

To repair Master Partition Boot Sector (Record)

Sys: C

DOS Structures

To manage files on a disk and enable all application programs to see a consistent disk interface no matter what type of disk is used, DOS uses several structures. The following list shows all the structures and areas that DOS defines and uses to manage a disk, in roughly the order in which they are encountered on a disk:

- Master and extended partition boot sectors

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

- DOS volume boot sector
- Root directory
- File Allocation Table (FAT)
- Clusters (allocation units)
- Data area
- Diagnostic read and write cylinder

A hard disk has all these DOS disk management structures allocated, and a floppy disk has all but the master and extended partition boot sectors and the diagnostic cylinder. These structures are created by the DOS FDISK program, which has no application on a floppy disk because floppy disks cannot be partitioned.

Each disk area has a purpose and function. If one of these special areas is damaged, serious consequences can result. Damage to one of these sensitive structures usually causes a domino effect and limits access to other areas of the disk or causes further problems in using the disk. For example, DOS cannot read and write files if the FAT is damaged or corrupted. You therefore should understand these data structures well enough to be able to repair them when necessary. Rebuilding these special tables and areas of the disk is essential to the art of data recovery.

Master Partition Boot Sectors

To share a hard disk among different operating systems, the disk might be logically divided into one to four master partitions. Each operating system, including DOS (through versions 3.2), might own only one partition. DOS 3.3 and later versions introduced the extended DOS partition, which allows multiple DOS partitions on the same hard disk. With the DOS **FDISK** program, you can select the size of each partition. The partition information is kept in several partition boot sectors on the disk, with the main table embedded in the **master partition boot sector**. *The master partition boot sector is always located in the first sector of the entire disk (cylinder 0, head 0, sector 1).* The extended partition boot sectors are located at the beginning of each extended partition volume.

Each DOS partition contains a DOS volume boot sector as its first sector. With the DOS **FDISK** utility, you might designate a single partition as active (or bootable). The master partition boot sector causes the active partitions volume boot sector to receive control when the system is started or reset. Additional master disk partitions can be set up for Novell NetWare, and for OS/2 HPFS, PCIX (UNIX), XENIX, CP/M-86, or other operating systems. Any of these foreign operating system partitions cannot be accessible under DOS, nor can any DOS partitions normally be accessible under other operating systems. (OS/2 and DOS share FAT partitions, the high-performance file system (HPFS) is exclusive to OS/2, and the NTFS is exclusive to Windows NT.)

A hard disk must be partitioned to be accessible by an operating system. You must partition a disk even if you want to create only a single partition. **Table 2** below shows the format of the **Master Boot Record (MBR)** with partition tables.

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

| TABLE 2 MASTER BOOT RECORD (PARTITION TABLE) | | |
|----------------------------------------------------|---------|-------------------------------------------------|
| Offset | Length | Description |
| Partition Table Entry #1 | | |
| 1BEh 446 | 1 byte | Boot Indicator Byte (80h = Active, else 00h) |
| 1BFh 447 | 1 byte | Starting Head (or Side) of Partition |
| 1C0h 448 | 16 bits | Starting Cylinder (10 bits) and Sector (6 bits) |
| 1C2h 450 | 1 byte | System Indicator Byte (see Table 3) |
| 1C3h 451 | 1 byte | Ending Head (or side) of Partition |
| 1C4h 452 | 16 bits | Ending Cylinder (10 bits) and Sector (6 bits) |
| 1C6h 454 | 1 dword | Relative Sector Offset of Partition |
| 1Cah 458 | 1 dword | Total Number of Sectors in Partition |
| Partition Table Entry #2 | | |
| 1CEh 462 | 1 byte | Boot Indicator Byte (80h = Active, else 00h) |
| 1CFh 463 | 1 byte | Starting Head (or Side) of Partition |
| 1D0h 464 | 16 bits | Starting Cylinder (10 bits) and Sector (6 bits) |
| 1D2h 466 | 1 byte | System Indicator Byte (see Table 3) |
| 1D3h 467 | 1 byte | Ending Head (or side) of Partition |
| 1D4h 468 | 16 bits | Ending Cylinder (10 bits) and Sector (6 bits) |
| 1D6h 470 | 1 dword | Relative Sector Offset of Partition |
| 1Dah 474 | 1 dword | Total Number of Sectors in Partition |
| Partition Table Entry #3 | | |
| 1Deh 478 | 1 byte | Boot Indicator Byte (80h = Active, else 00h) |
| 1DFh 479 | 1 byte | Starting Head (or Side) of Partition |
| 1E0h 480 | 16 bits | Starting Cylinder (10 bits) and Sector (6 bits) |
| 1E2h 482 | 1 byte | System Indicator Byte (see Table 3) |
| 1E3h 483 | 1 byte | Ending Head (or side) of Partition |
| 1E4h 484 | 16 bits | Ending Cylinder (10 bits) and Sector (6 bits) |
| 1E6 486 | 1 dword | Relative Sector Offset of Partition |
| 1Eah 490 | 1 dword | Total Number of Sectors in Partition |
| Partition Table Entry #4 | | |
| 1Eeh 494 | 1 byte | Boot Indicator Byte (80h = Active, else 00h) |
| 1EFh 495 | 1 byte | Starting Head (or Side) of Partition |
| 1F0h 496 | 16 bits | Starting Cylinder (10 bits) and Sector (6 bits) |
| 1F2h 498 | 1 byte | System Indicator Byte (see Table 3) |
| 1F3h 499 | 1 byte | Ending Head (or side) of Partition |
| 1F4h 500 | 16 bits | Ending Cylinder (10 bits) and Sector (6 bits) |
| 1F6h 502 | 1 dword | Relative Sector Offset of Partition |
| 1Fah 506 | 1 dword | Total Number of Sectors in Partition |

Table 3 Shows the standard values and meanings of the System Indicator Byte.

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

| Value | Description |
|-------|--------------------------------------------------|
| 00h | No allocated partition in this entry |
| 01h | Primary DOS, 12 bit FAT (Partition < 16M |
| 04h | Primary DOS, 16 bit FAT (16M <= Partition <= 32M |
| 05h | Extended DOS (Points to next Primary Partition) |
| 06h | Primary DOS, 16 bit FAT (Partition > 32M) |
| 07h | OS/2 HPFS |
| 02h | MS-XENIX Root Partition |
| 03h | MS-XENIX usr Partition |
| 08h | AIX File System Partition |
| 09h | AIX Boot Partition |
| 50h | Ontrack Disk Manager READ ONLY Partition |
| 51h | Ontrack Disk Manager READ/WRITE Partition |
| 56h | Golden Bow Vfeature Partition |
| 61h | Storage Dimensions Speedstor Partition |
| 63h | IBM 386/ix or UNIX System V/386 Partition |
| 64h | Novell NetWare Partition |
| 75h | IBM PCIX Partition |
| D8h | Digital Research Concurrent DOS/CPM-86 Partition |
| F2h | Some OEM's DOS 3.2+ second partition |
| FFh | UNIX Bad Block Table Partition |

Dos Volume Boot Sectors

The **volume boot sector** is the first sector on any area of a drive addressed as a volume (or logical DOS disk). On a floppy disk, for example, this sector is the first one on the floppy disk because DOS recognizes the floppy disk as a volume with no partitioning required. On a hard disk, the volume boot sector or sectors are located as the first sector within any disk area allocated as a nonextended partition, or any area recognizable as a DOS volume.

This special sector resembles the master partition boot sector in that it contains a program as well as some special data tables. The first volume boot sector on a disk is loaded by the system ROM BIOS for floppies or by the master partition boot sector on a hard disk. This program is given control; it performs some tests and then attempts to load the first DOS system file (**IO.SYS**). The volume boot sector is transparent to a running DOS system; it is outside the data area of the disk on which files are stored.

You create a volume boot sector with the DOS **FORMAT** command (high-level format). Hard disks have a volume boot sector at the beginning of every DOS logical drive area allocated on the disk, in both the primary and extended partitions. Although all the logical drives contain the program area as well as a data table area, only the program code from the volume boot sector in the active partition on a hard disk is executed. The others are simply read by the DOS system files during boot-up to obtain their data table and determine the volume parameters.

The volume boot sector contains program code and data. The single data table in this sector is called the **media parameter block** or **disk parameter block**. DOS needs the information it contains

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

to verify the capacity of a disk volume as well as the location of important features such as the FAT. The format of this data is very specific. Errors can cause problems with booting from a disk or with accessing a disk. Some non-IBM OEM versions of DOS have not adhered to the standards set by IBM for the format of this data, which can cause interchange problems with disks formatted by different versions of DOS. The later versions can be more particular, so if you suspect that boot sector differences are causing inability to access the disk, you can use a utility program such as DOS BEBUG or Norton Utilities to copy a boot sector from the newer version of DOS to a disk formatted by the older version. This step should enable the new version of DOS to read the older disk and should not interfere with the less particular older versions. This might occur in mixing different OEM versions.

Table 4 shows the format and layout of the **DOS Boot Record (DBR)**

| TABLE 4 DOS BOOT RECORD (DBR) FORMAT | | | |
|--------------------------------------------------------------------------------------------------|-----|--------------|---------------------------------------------------|
| Offset | | | |
| Hex | Dec | Field Length | Description |
| 00h | 0 | 3 bytes | Jump Instruction to Boot Program Code |
| 03h | 3 | 8 bytes | OEM Name and DOS Version |
| 0Bh | 11 | 1 word | Bytes/Sector (usually 512) |
| 0Dh | 13 | 1 byte | Sectors/Cluster (Must be a power of 2) |
| 0Eh | 14 | 1 word | Reserved Sectors (Boot Sectors, usually 1) |
| 10h | 16 | 1 byte | FAT Copies (usually 2) |
| 11h | 17 | 1 word | Maximum Root Directory Entries (usually 512) |
| 13h | 19 | 1 word | Total Sectors (If Partition <= 32M, else 0) |
| 15h | 21 | 1 byte | Media Descriptor Byte (F8h for Hard Disks) |
| 16h | 22 | 1 word | Sectors/FAT |
| 18h | 24 | 1 word | Sectors/Track |
| 1Ah | 26 | 1 word | Number of Heads |
| 1Ch | 28 | 1 dword | Hidden Sectors (If Partition <= 32M, 1 word only) |
| For DOS 4.0 or Higher Only, Else 00h | | | |
| 20h | 32 | 1 dword | Total Sectors (If Partition > 32M, else 0) |
| 24h | 36 | 1 byte | Physical Drive No. (00h=floppy, 80h=hard disk) |
| 25h | 37 | 1 byte | Reserved (00h) |
| 26h | 38 | 1 byte | Extended Boot Sector Signature (29h) |
| 27h | 39 | 1 dword | Volume Serial Number (32 bit random number) |
| 2Bh | 43 | 11 bytes | Volume Label ("NO NAME" stored if no label) |
| 36h | 54 | 8 bytes | File System ID ("FAT12" or "FAT16") |
| For All Versions of DOS | | | |
| 3Eh | 62 | 448 bytes | Boot Program Code |
| 1FEh | 510 | 2 bytes | Signature Bytes (55AAh) |
| A WORD is two bytes read in reverse order, and a DWORD is two WORDs read in reverse order | | | |

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

Root Directory

A *directory* is a simple database containing information about the files stored on a disk. Each record in this database is 32 bytes long, and no delimiters or separating characters are between the fields or records. A directory stores almost all the information that DOS knows about a file: name, attribute, time and date of creation, size, and where the beginning of the file is located on the disk. (The information a directory does not contain about a file is where the file continues on the disk and whether the file is contiguous or fragmented. The FAT contains that information.)

Two basic types of directories exist: the **root directory and subdirectories**. They differ primarily in how many there can be and in where they can be located. Any given volume can have only one root directory, and the root directory is always stored on a disk in a fixed location immediately following the two FAT copies. Root directories vary in size because of the varying types and capacities of disks, but the root directory of a given disk is fixed. After a root directory is created, it has a fixed length and cannot be extended to hold more entries. Normally, a hard disk volume has a root directory with room for 512 total entries. Subdirectories are stored as files in the data area of the disk and therefore have no fixed length limits.

Every directory, whether it is the root directory or a subdirectory, is organized in the same way. A directory is a small database with a fixed record length of 32 bytes. Entries in the database store important information about individual files and how files are named on a disk. The directory information is linked to the FAT by the starting cluster entry. In fact, if no file on the disk were longer than one single cluster, the FAT would be unnecessary. The directory stores all of the information needed by DOS to manage the file, with the exception of all the clusters that the file occupies other than the first one. The FAT stores the remaining information about other clusters the file uses.

To trace a file on a disk, you start with the directory entry to get the information about the starting cluster of the file and its size. Then go to that file allocation table. From there, you can go to a chain of clusters the file occupies until you reach the end of the file.

DOS directory entries are 32 bytes long and are in the format shown in **Table 5**.

Table 6 describes the DOS Directory file attribute type.

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

| TABLE 5 DOS DIRECTORY) FORMAT | | | |
|----------------------------------|-----|--------------|------------------|
| Offset | Dec | Field Length | Description |
| Hex | | | |
| 00h | 0 | 8 bytes | File Name |
| 08h | 8 | 3 bytes | File Extension |
| 0Bh | 11 | 1 byte | File Attributes |
| 0Ch | 12 | 10 bytes | Reserved (00h) |
| 16h | 22 | 1 word | Time of Creation |
| 18h | 24 | 1 word | Date of Creation |
| 1Ah | 26 | 1 word | Starting Cluster |
| 1Ch | 28 | 1 dword | Size in bytes |

NOTE: File names and extensions are left justified and padded with spaces (32h). The first byte of the file name indicates the file status as follows:

| Hex | File Status |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 00h | Entry never used; entries past this point not searched. |
| 05h | Indicates first character of filename is actually E5h. |
| E5h | “Lower case Sigma”. Indicates file has been erased. |
| 2Eh | “.” (period) Indicates this entry is a directory. If the second byte is also 2Eh, the cluster field contains the cluster number of parent directory (0000h if the parent is the root). |

| TABLE 6 DOS DIRECTORY FILE ATTRIBUTE BYTE | | | | | | | | | |
|----------------------------------------------|---|---|---|---|---|---|---|-------|------------------------------------|
| Bit Position Hex | | | | | | | | Value | Description |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 01h | Read Only File |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 02h | Hidden File |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 04h | System File |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 08h | Volume Label |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 10h | Subdirectory |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 20h | Archive (updated since backup) |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 40h | Reserved |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 80h | Reserved |
| Examples | | | | | | | | | |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 21h | Read Only, archive |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 32h | Hidden, Subdirectory, archive |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 27h | Read only, hidden, system, archive |

File Allocation Tables (FATs)

The *file allocation table (FAT)* is a table of number entries describing how each cluster is allocated on the disk. The data area of the disk has a single entry for each cluster. Sectors in the nondata area on the disk are outside the range of the disk controlled by the FAT. The sectors involved in any of the boot sectors, file allocation table, and root directory are outside the range of sectors controlled by the FAT.

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

The FAT does not manage every data sector specifically, but rather allocates space in groups of sectors called **clusters** or **allocation units**. A cluster is one or more sectors designated by DOS as allocation units of storage. The smallest space a file can use on a disk is one cluster; all files use space on the disk in integer cluster units. If a file is one byte larger than one cluster, two clusters are used. DOS determines the size of a cluster when the disk is high-level formatted by the DOS **FORMAT** command.

You can think of the FAT as a sort of spreadsheet that controls the cluster use of the disk. Each cell in the spreadsheet corresponds to a single cluster on the disk; the number stored in that cell is a sort of code telling whether the cluster is used by a file, and if so where the next cluster of the file is located.

The numbers stored in the FAT are hexadecimal numbers that are either 12 or 16 bits long. The 16-bit FAT numbers are easy to follow because they take an even two bytes of space and can be edited fairly easily. The 12-bit numbers are 1 1/2 bytes long, which presents a problem when most disk sector editors show data in byte units. To edit the FAT, you must do some hex/binary math to convert the displayed byte units to FAT numbers. Fortunately, (unless you are using the DOS **DEBUG** program), most of the available tools and utility programs have a FAT editing mode that automatically converts the numbers for you. Most of them also show the FAT numbers in decimal form, which most people find easier to handle.

The DOS **FDISK** program determines whether a 12-bit or 16-bit FAT is placed on a disk, even though the FAT is written during the high-level format (**FORMAT**). All floppy disks use a 12-bit FAT, but hard disks can use either. On hard disk volumes with more than 16 megabytes (32,768 sectors), DOS creates a 16-bit FAT; otherwise, DOS creates a 12-bit FAT.

DOS keeps two copies of the FAT. Each one occupies contiguous sectors on the disk, and the second FAT copy immediately follows the first. Unfortunately, DOS uses the second FAT copy only if sectors in the first FAT copy become unreadable. If the first FAT copy is corrupted, which is a much more common problem, DOS does not use the second FAT copy. Even the DOS **CHKDSK** command does not check or verify the second FAT copy. Moreover, whenever DOS updates the first FAT, large portions of the first FAT automatically are copied to the second FAT. If, therefore, the first copy was corrupted and then subsequently updated by DOS, a large portion of the first FAT would be copied over the second FAT copy, damaging it in the process. After the update, the second copy is usually a mirror image of the first one, complete with any corruption. Two FATs rarely stay out of sync for very long. When they are out of sync and DOS writes to the disk and causes the first FAT to be updated, it also causes the second FAT to be overwritten by the first FAT. Because of all this, the usefulness of the second copy of the FAT is limited to manual repair operations, and even then it is useful only if the problem is caught immediately, before DOS has a chance to update the disk.

Clusters (Allocation Units)

The term **cluster** was changed to **allocation unit** in DOS 4.0. The newer term is appropriate because a single cluster is the smallest unit of the disk that DOS can handle when it writes or reads a file. A cluster is equal to one or more sectors, and although a cluster can be a single sector, it is usually more than one. Having more than one sector per cluster reduces the size and processing overhead of the enables DOS to run faster because it has fewer individual units of the disk to

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

manage. The trade-off is in wasted disk space. Because DOS can manage space only in full cluster units, every file consumes space on the disk in increments of one cluster.

Smaller clusters generate less slack (space wasted between the actual end of each file and the end of the cluster). With larger clusters, the wasted space grows larger. For hard disks, the cluster size can vary greatly among different versions of DOS and different disk sizes. **Table 1** shows the cluster sizes DOS selects for a particular volume size. In most cases, these cluster sizes, selected by the DOS **FORMAT** command, are the minimum possible for a given partition size. Therefore, 8K clusters are the smallest possible for a partition size of greater than 256M.

The effect of these larger cluster sizes on disk use can be substantial. A drive containing about 5,000 files, with average slack of one-half of the last cluster used for each file, wastes about 20M [5000*(.5*8)K] of file space on a disk set up with IBM DOS or MS-DOS. Using Compaq DOS 3.31, this wasted space doubles to 40M for the same 5,000 files. Someone using a system with Compaq DOS 3.31 could back up, repartition, and reformat with IBM DOS, and after restoring all 5,000 files, gain 20M of free disk space.

The reason that Compaq DOS 3.31 does not use the most efficient (or smallest) cluster size possible for a given partition size is because its makers were interested in improving the performance of the system at the expense of great amounts of disk space. Larger cluster sizes get you a smaller FAT, with fewer numbers to manage; DOS overhead is reduced when files are stored and retrieved, which makes the system seem faster. For example, the **CHKDSK** command runs much faster on a disk with a smaller FAT. Unfortunately, the trade-off for speed here is a tremendous loss of space on the disk. (Compaq DOS 4.0 and 5.0 use IBM DOS and MS-DOS conventions.)

Windows 95 will soon be available with a new 32-bit FAT system that allows for more than 64K clusters. Since there can be more clusters, the individual clusters can be smaller. This will alleviate the large cluster size problem for larger drives, and will extend the size of a FAT partition on a hard disk to beyond the current 2GB limit. Note that Windows NT and OS/2 already have more sophisticated file systems that do away with the FAT structure and that are not limited in the way that FAT partitions are.

The Data Area

The data area of a disk is the area that follows the boot sector, file allocation tables, and root directory on any volume. This area is managed by the FAT and the root directory. DOS divides it into allocation units sometimes called clusters. These clusters are where normal files are stored on a volume.

Diagnostic Read And Write Cylinder

The **FDISK** partitioning program always reserves the last cylinder of a hard disk for use as a special diagnostic read-and-write test cylinder. That this cylinder is reserved is one reason **FDISK** always reports fewer total cylinders than the drive manufacturer states are available. DOS (or any other operating system) does not use this cylinder for any normal purpose, because it lies outside the partitioned area of the disk.

On systems with IDE, SCSI, or ESDI disk interfaces, the drive and controller might allocate an additional area past the logical end of the drive for a bad-track table and spare sectors. This situation may account for additional discrepancies between **FDISK** and the drive manufacturer.

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

The diagnostics area enables diagnostics software such as the manufacturer-supplied Advanced Diagnostics disk to perform read-and-write tests on a hard disk without corrupting any user data. Low-level format programs for hard disks often use this cylinder as a scratch-pad area for running interleave tests or preserving data during nondestructive formats. This cylinder is also sometimes used as a head landing or parking cylinder on hard disks that do not have an automatic parking facility.

Disk Speeds and Sector Lengths

There has been discussion of speed advantages on disks with fixed length sectors. Such disks have more sectors on the outermost tracks (or cylinders) than they do in the inner tracks, so data can be read faster from the outer tracks. On a 3.5" diameter disk, the innermost track is about 3" long and the outermost about 11" long. So you can read almost 4 times as much data from one revolution of the outermost track. Sounds impressive, but what does it really mean?

If the disk spins at 6000 RPM, it takes ten milliseconds (ms) to make one revolution. This means that, regardless of any other factors, after you have moved the read/write head to a sector, it's going to take an average of 5 ms for the disk to rotate and bring the data to the head (this is called "latency"). Sometimes you'll luck out and the data will be right there, sometimes it will take almost a full revolution; on average, it will be half a revolution, or 5 ms. Only then will the actual reading or writing of the data begin.

A sector typically holds 512 bytes. The number of sectors per track varies widely; a 1GB disk may have 500 sectors per track, a 2.5GB may have 1500. For a simple example, let's assume 1000 sectors per track average. That means we'll have 400 sectors (200KB) on the inner track, and 1600 (800KB) on the outer.

If you are going to read a large amount of data, more than will fit on a track (more than 200KB in this example), then having the data on the outer tracks will save time -- but not more than 5 or 10 ms. The vast majority of disk I/Os are a lot less than 200KB, so most sites will never benefit from this speed difference. In addition, you may have some bad sectors, which will have been replaced with spare sectors. The Logical Cluster Number (LCN) will be in sequence with the other clusters in the file you are reading, but the physical location could be anywhere on the disk.

The bottom line is that some sites, especially those which regularly read hundreds of KB at a time, may benefit a little from the speed difference between inner and outer tracks. But for most sites, the disk latency and head travel time will completely overshadow any speed gains; it's just too small a percentage of the total access time to be worth spending your time on.

FAT or NTFS for the Boot/System Partition

FAT format has been recommended for the boot partition, so that in the event of a catastrophic failure, you can recover by booting from a bootable DOS floppy and then repairing Windows NT. The disadvantage, which many have protested, is that you lose all of the security benefits of NTFS. An alternative, which so far has worked for me, is to have a second boot partition on another disk drive, and install Windows NT into that. This secondary Windows NT can be a minimal installation to save disk space. Should your primary boot partition become corrupted, you can boot to the secondary and then repair the primary. Thus you can use NTFS on all partitions, and still have a relatively easy recovery path.

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

Specialized Partitions - What if You Run Out of Space

There have been comments on the specialized partition scheme in the article "Configuring Partitions", the objection being that you would run out of free space sooner. At some sites, this is true. While I believe the plan we laid out is best for most sites, no single plan for partitioning is going to be suitable for all sites. I know of one case where a single 4.5GB partition is the most efficient plan.

If you are going to break up a disk into smaller partitions, you need to plan ahead and leave room for expansion, especially on the system partition. However, virtually every partition is eventually going to fill up, and a new or larger partition will be needed. I favor creating a completely new partition and copying the data over, but not deleting the original files until you are sure the copies are good. Once you are sure you have lost no data, then you can delete the original files and use the old partition for some other purpose.

Cluster Sizes

When you format an NTFS partition, you usually are given a choice of cluster sizes: 512 bytes, or the default. The default sizes are:

- Partitions up to 512MB default to 512 bytes.
- Partitions 513 to 1024MB default to 1KB.
- Partitions 1025 to 2048MB default to 2KB.
- Partitions of more than 2048MB default to 4KB.

When you format an NTFS partition using Disk Administrator, you have a choice of the default (as shown above) or 512 bytes. If you want some other cluster size, you must run `format/d:` from the command line (where "d" is the partition's drive letter). Your choices are limited to the four sizes listed above, plus 8KB, 16KB, 32KB and 64KB. However, compression is not supported on partitions with a cluster size greater than 4KB. Now, how do you decide which size to use?

Disk Space Wasted by Clusters

Data is written to disks in units of the cluster size. If you have a 1024 byte cluster size, the actual space used by any file will always be a multiple of 1KB. So the difference between the actual amount of data in the file and the amount of space used on the disk will always be between 0 and 1023 bytes. And that means that for each file on the disk, you will have an average of 512 bytes of wasted space. For any disk or partition, multiply the number of files by one-half the cluster size to get the approximate amount of unused space lost due to the disk cluster size.

Suppose you are going to have 50,000 files on a partition. The approximate amount of wasted space by cluster size will be:

| Cluster Size | Wasted Bytes |
|--------------|--------------|
| 512 bytes | 12,800,000 |
| 1KB | 25,600,000 |
| 2KB | 51,200,000 |
| 4KB | 102,400,000 |

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

It seems like a lot, but bear in mind that the 100MB wasted with 4KB clusters is only 5% of a 2GB partition; 25MB wasted with 1KB clusters is only 1.25% of a 2GB partition. It's probably not significant.

But if you have a lot more files or a smaller partition, it could be important.

MFT Records

In the Master File Table (MFT), there is a record for each file, including directories, on the partition. This record is called the File Record Segment. The FRS is always 1KB in size. If you are using a 512 byte cluster, a FRS can be fragmented into two pieces. On the other hand, if you use a 2 or 4KB cluster size, more than one FRS will fit in each cluster, wasting no disk space at all.

I/O Speed

The smaller the cluster size, the more fragmented a file can become, and the more fragmented the file, the longer I/O will take. But if the file is contiguous, cluster size will not matter. If you run Diskkeeper regularly, this will not be a factor.

The MFT is another matter. Since the MFT is always open for exclusive use by Windows NT, Diskkeeper cannot defragment it. If you are using 512 byte clusters, the MFT will fragment, performance will suffer, and there is nothing you can do about it except backup, reformat, restore.

The upper limit on a physical I/O from a Windows NT application is 64KB. This is completely independent of cluster size, and thus will not impact I/O speed.

Internet Use

Web pages run between three and four KB in size. If you use a 4KB cluster size, web pages will always be contiguous. If your work is mainly with web pages (lots of browsing), this could be significant.

Recommendations

I recommend using a 1KB cluster size. While 512 byte clusters will save disk space, the amount, as shown above, is not enough to matter. Nor is the size limit, since an NTFS partition with a 512 byte cluster size can be up to 2 terabytes (TB). The only exceptions would be:

- If you will have a large number of files (say, 100,000 or more), because this will extend the MFT and cause it to fragment. If you use a 4KB cluster size, the MFT fragmentation will be only a quarter of what you would have with 1KB clusters.
- If you will be using a volume set larger than 4TB. A 1KB cluster can only handle 4TB, a 2KB cluster can handle 8TB, and so on.

A Note About FAT Partitions

Clusters on FAT partitions are much the same as on NTFS, with a few important exceptions. First, you should know that partitions under 16MB use FAT-12, with 4KB clusters. Larger partitions use FAT-16. The default cluster sizes are as follows:

- Partitions 16 to 127MB default to 2KB.
- Partitions 128 to 255MB default to 4KB.

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

- Partitions 256 to 511MB default to 8KB.
- Partitions 512 to 1023MB default to 16KB.
- Partitions 1024 to 2047MB default to 32KB.
- Partitions 2048 to 4095MB default to 64KB.
- Partitions 4095 to 8191MB default to 128KB.
- Partitions 8192 to 16383MB default to 256KB.

NOTE: Windows NT 3.51 and earlier do not support FAT clusters larger than 64KB.

Since 256KB is the largest possible cluster size for a FAT partition, partitions of more than 16384MB simply waste space; FAT partitions are limited to 65,535 clusters, so they just can't see more than 16384MB. Note that, due to this limit in the number of clusters, the defaults listed above are also minimum sizes. For example, if you format a 511MB FAT partition with 4KB clusters, you won't be able to access all of the disk. 65,535 4KB clusters total 268,431,360 bytes, or just under 256MB. Of course, none of the data about the MFT applies to FAT partitions.

Configuring Windows Nt Partitions

This begins a series of articles on setting up and administering disks and partitions, and discusses how to distribute your disk space. Some of this information is based on our own experience, so you may disagree with some points. That's fine, your system and needs may require different methods. This system does work, however, and will at least supply you with some information to work with.

You should have a boot/system partition, an application partition and one or more data partitions. The boot partition will hold Windows NT and the primary pagefile; the application partition will hold all other programs, except those which must be installed on the system partition. All other files, whenever possible, go to your data partitions.

If you have the disk space, it is a good idea to have at least three data partitions: one for archive data, one for volatile data, one for other data. This saves backup time - the boot and program partitions only need to be backed up after initial installation and on those rare occasions when you install a Windows NT Service Pack or upgrade, or install a new application. The same is true of the archive partition, because that is where you would put data that is not likely to change (lookup tables and such). By volatile data I mean temp files and work files, which change very frequently, and do not need to be backed up anyway. So the only partitions that require frequent backups are the ones for "other data". And the easier and faster the backup procedure is, the more frequently you can back up.

The other reason for this scheme is that when data loss or corruption occurs, it is almost always because of a bad write to the partition. If 99% of the writing is being done to one partition, then the chance of corrupting the other partitions is reduced by 99%. If this one partition only contains data (which has been backed up), and no program files, then losing the partition or a file is not very damaging or important; it is not going to make your machine unbootable, so recovery is easy. Of course, you can have more than one partition of any of these types.

Now, as to size, the boot/system partition should be at least 400MB, plus the size of your pagefile. You can put the pagefile in a partition of its own, but it is not vital. I have run systems both ways, and have not noticed any difference, but the advantage to putting the pagefile on the system partition is that you can move it to another partition if you need more space for the system. (I had a 250MB

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

system partition and could not install Windows NT 4.0 Service Pack 3 because I had only 50MB free; the install said "insufficient free space". I had to reformat and reinstall the entire system.)

The size of the application partition depends on your applications. Figure the approximate size of your application files, not counting data files which you can put on data partitions, or applications which must install on the system partition. Include space for any new applications you plan to add. Now add free space by multiplying your total by some value between $1 \frac{1}{3}$ and 2, so that the partition will be around 50% to 75% full. Plenty of free space means faster I/O, so if you can spare the disk space, go for 50% free.

The data partitions use all the rest of your disk space. How you configure them (which format, what size, what volume type) depends on several factors:

- Amount of data
- Type of format
- Data recoverability
- Security
- Disk space available

To divide your data as suggested above, you need to figure out how much space will be used by each type of data. Always leave a generous amount of free space, preferably 50%, since speed of access is most important on data files, especially those that are most frequently accessed.

The boot/system partition should be FAT format, but the other partitions can be FAT or NTFS. (See the article "FAT VS. NTFS: WHICH DO YOU CHOOSE, AND WHEN?" in Volume 2, Issue 6 of this eLetter for data on selecting which format to use).

NTFS has its own built-in sector recovery, in the event of a disk sector going bad, but data recovery depends on fault-tolerance. Fault-tolerance requires NTFS format, and more than one physical disk, and increases the amount of disk space needed. For example, a mirror set keeps multiple copies of each file, one on each partition in the set.

To implement the Windows NT security features, you must use the NTFS format. The features just don't exist for FAT; aside from the disk format, security is not a factor in configuring partitions.

Probably the most important factor is disk space - you only have a certain amount to work with. But don't make the mistake of not considering alternatives just because they require adding hard drives. The prices of IDE disks are dropping rapidly. SCSI is more expensive, but even so, mass storage really is cheap. When you consider the expense of recovering corrupted data (How many man-hours to restore the data? How much income lost because the data or system is unavailable?), one single recovery event would probably cost more than the additional disk that would have prevented the data loss in the first place.

There are a few other minor points worth mentioning. Bear in mind that, aside from the boot/system partition, you can change partitions pretty easily. A FAT partition can be converted to NTFS without having to backup/restore the data (an NTFS partition cannot, however, be converted to FAT without full backup/restore). But you should always do a full backup in any case before converting, or doing

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

anything that could corrupt data. Although Windows NT itself cannot change a partition size without erasing all of the data, there are programs that allow you to change partition sizes *without* losing the data. Be warned that these programs are dangerous because they are powerful; if you make a mistake, you can wipe out the whole disk. Back up first! The point here is, whatever configuration you choose can be changed if needed.

Aside from details applying to special cases (like applications that require their data files to be in the same folder as the program), that's about all there is to configuring your disks. It's really pretty simple, given the basic data.

Efficient NTFS Partitions

How you set up and use an NTFS partition can have a great deal of effect on performance. All partitions are not created equal! In fact, two brand-new but differently set up partitions can yield drastically different performances. Response time can degrade over time, even if you keep the partition defragmented.

The Partition Itself

Partitions should be created NTFS, not converted from FAT. On a newly formatted NTFS partition, the Master File Table (MFT) is created at the beginning of the disk, and about 12 1/2% of the disk is reserved for the MFT. But when you convert a FAT partition to NTFS, the MFT is placed wherever there is free space. It almost always ends up badly fragmented.

Large partitions should be avoided. They take a lot longer to back up and to restore, data corruption becomes more likely because there is so much more writing going on, and access to the disk becomes slower (it takes longer to find, read and write files). Of course, there are valid reasons for very large partitions - if you have a 5 GB database or you work with large video files, you'll need big partitions. Just don't make them big if you can avoid it; 1 to 2 GB is about right. It's also a good idea to have specialized partitions: System, Applications, Data, Temp, etc. This will increase safety and performance.

Directories

It's nice to have deep, multi-branched directory trees; I like the logical organization, keeping separate types of files neatly sorted. The bad news: Deep trees can really slow things down. The good news: It's easy to tidy up deep trees so they don't slow things down. Here are the details:

Under NTFS, each directory is a file just like any other, but with a special internal structure called a "B+ Tree". It's fairly complicated, but for our purposes it's enough to say that it is a very good structure for a directory tree, but can be weak on handling changes. In other words, the more changes you make, the more complicated it gets internally, so the longer it takes to locate your file. Since files are listed in the directory file alphabetically by file name, adding new files (or directories) can require changes in the middle of the tree structure. Many such changes can make the structure quite complex, and more complexity means less speed.

Files are located by searching through the directories. If you are looking for a file in a tree that is ten levels deep, you have to locate ten directories before you get to the one that points to the file itself. That takes a lot longer than locating a file that is only three levels deep. Plus, if the directories have

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

been changed a lot so that their internal structure has become complex, finding files can become very slow.

Directories tend to grow, but rarely shrink. Sometimes when you add a new file or directory, it can be fitted into the space left by a deleted file, but often it uses a new space. The directory grows and can fragment, slowing down access even more.

One additional thing: Long file names can cause directories and the MFT to fragment. The way the file names are stored, each character requires two bytes. For computer efficiency, the DOS 8-dot-3 format is best. On the other hand, for human efficiency, 20 to 30 character names are much better. Of course, there are exceptions, such as files on a CD-ROM or an archive partition where they won't be re-written, but in general, don't go over thirty characters.

Cluster Size

New data regarding the MFT and its internal functions leads me to recommend 4096KB as the best cluster size, especially if you will have a very large number of files or will be using compression. Never use less than 1024KB, as this will allow MFT records to fragment, and never exceed 4096KB, as compression will not work.

Definitions

Allocation Unit - Same as Cluster.

ARLL - Advanced Run Length Limited. More complex yet powerful derivatives of the **RLL (Run Length Limited)** scheme. Include 1,7 and 3,9 encoding. Allows 34 or more sectors per track.

Bad Sector - A disk sector that cannot hold data reliably because of a media flaw or damaged format markings.

Bad Track Table - A label affixed to the casing of a hard disk drive that tells which tracks are flawed and cannot hold data. The listing is entered into the low level formatting program.

Bit Density - Expressed as **Bits Per Inch (BPI)**, bit density defines how many bits can be written onto one linear inch of a track. Sometimes also called **linear density**.

Cluster - Also called **allocation unit**. A group of sectors on a disk that forms a fundamental unit of storage to the operating system.. Clusters or allocation units size is determined by DOS when the disk is high level formatted. There are one or more sectors on a track.

Cylinder - The set of tracks on a disk that are on each side of all of the platters in a stack and are the same distance from the center of the disk. The total number of tracks that can be read without moving the heads. A floppy drive with two heads usually has 160 tracks, which are accessible as 80 cylinders. A typical 4G hard disk has 10 platters with 20 heads (19 for data and one servo head) and 4,000 cylinders, in which each cylinder is composed of 19 tracks.

DEBUG - The name of a utility program included with DOS that is for specialized purposes such as altering memory locations, tracing program execution, patching programs and disk sectors, and performing other low level tasks.

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

Defragmentation - The process of rearranging disk sectors so that files are stored on consecutive sectors in adjacent tracks.

Extended Partition - A nonbootable DOS partition containing DOS volumes. Starting with DOS V3.3, the DOS **FDISK** program can create two partitions that serve DOS. One ordinary bootable partition (called the primary partition) and an extended partition, which may contain as many as 23 volumes from D: through Z:.

FAT - File Allocation Table. A table held near the outer edge of the disk that tells which sectors are allocated to each file and in what order. The standard file system used by DOS, OS/2, and Windows NT. FAT partitions support the file names of 11 characters maximum (8 plus a 3 character extension), and a volume can be as large as 2G in size.

FDISK - The name of the disk partitioning program under several operating systems used to create the Master Boot Record and allocate partitions for the operating system's use.

FORMAT - The DOS format program that performs both low and high level formatting on floppy disks but only high level formatting on hard disks.

High Level Formatting - Formatting performed by the DOS **FORMAT** program. Among other things, it creates the root directory and file allocation tables.

High Performance File System (HPFS). A UNIX type file system assessable only under OS/2 and Windows NT. DOS applications running under OS/2 or Windows NT can access files in HPFS partitions but straight DOS cannot. File names can be 256 characters long, and volume size is limited to 8G.

Interleave Ratio - The number of sectors that pass beneath the read/write heads before the next numbered sector arrives. When the interleave ration is 3:1, for example, a sector is read, two pass by, and then the next is read. A proper interleave ratio, laid down during low level formatting, enables the disk to transfer information without excessive revolutions due to missed sectors.

Logical Drive - A drive as named by a DOS drive specifier , such as C: or D:. Under DOS 3.3 or later, a single physical drive can act as several logical drives, each with its own specifier. Same as Volume. Anything that DOS assigns a drive letter to in an extended DOS partition.

Lost Clusters - Clusters that have been marked accidentally as unavailable in the file allocation table even though they don't belong to any file listed in the directory.

Low Level Formatting - Formatting that divides tracks into sectors on the platter surfaces. Places sector identifying information before and after each sector and fills each sector with null data (usually hex F6). Specifies the sector interleave and marks defective tracks by placing invalid checksum figures in each sector on a defective track. The dividing of the disk's tracks into a specific number of sectors. The sector's header and trailer information is recorded, as are the intersector and intertrack gaps. Each sector's data area is filled with a dummy byte value or test pattern of values. For floppy disks, the number of sectors recorded on each track depends on the type of disk and drive. For hard disks, the number of sectors per track depends on the drive and controller interface.

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

Master Boot Record (MBR) - Same as Master Boot Table. Always found on the first sector, of the first track, of the first surface.

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 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.

Partition A section of a hard disk devoted to a particular operating system. Most hard disks have only one partition which is devoted to DOS. A hard disk can have as many as four partitions, each occupied by a different operating system. DOS V3.3 or higher can occupy two of these four partitions.

Partitioning - The dividing of a hard disk in to individual segments called **partitions**. When a disk is partitioned, the information is recorded in a record called the **Master Boot Record (MBR)**, **Master Boot Sector**, or **Master Boot Table**. DOS allows for up to four partition tables, or partitions, with three of them being designated as Primary Partitions, and one of them as an Extended Partition. **Primary Partitions** are used for the placement of separate operating systems. **Logical Drives** can only be placed in the Extended Partition.

Primary Partition - An ordinary, single volume bootable partition.

Root Directory - The main directory on any hard or floppy disk. Has a fixed size and location for a particular disk volume and cannot be resized dynamically the way subdirectories can.

RLL - Run Length Limited. A method used on some hard disks to encode data into magnetic pulses. RLL requires more processing, but stores almost 50 percent more data per disk than the older **MFM (modified frequency modulation)** method. Actually accomplished by recording 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. Can support up to 17 x 512 byte sectors per track.

Sector - The smallest unit into which a track is divided during the low level formatting process. A sector consists of 512 bytes of data space. A section of one track defined with identification markings and an identification number. Most sectors hold 512 bytes.

ST506/412 - Seagate Technologies 506/412. 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 the controller flow over 2 ribbon cables, one for drive control and one for data.

Starting Cluster - The number of the first cluster occupied by a file. Listed in the directory entry of every file.

TPI - Tracks Per Inch. Used as a measurement of magnetic track density. Standard 5 1/4 inch 360K floppy disks have a density of 48 TPI, and the 1.2M disks have 96 TPI density. All 3 1/2 inch disks have a 135.4667 TPI density, and hard disks can have densities greater than 3,000 TPI.

Legacy Hard Disk Drive Configuration

Mark E. Donaldson

Track - One of many concentric circles that holds data on a disk surface. Consists of a single line of magnetic flux changes and is divided into some number of 513 byte sectors.

Track Density - Expressed as Tracks Per Inch (TPI), track density defines how many tracks are recorded in one inch of space measured radially from the center of the disk. Sometimes also called radial density.

Virtual Memory - A technique by which operating systems load more programs and data to memory than they can hold. Parts of the programs and data are kept on disk and constantly swapped back and forth into system memory. The applications software programs are unaware of this setup and act as though a large amount of memory is available.

Volume - Same as a Logical Drive. Anything that DOS assigns a drive letter to in an extended DOS partition. A portion of a disk signified by a single drive specifier. Under DOS V3.3 and later, a single hard disk can be partitioned into several volumes, each with its own logical drive specifier (C:, D:, and so on).

Volume Boot Sector -

Windows NT File System (NTFS). A UNIX type file system currently accessible only under Windows NT., but drivers should be available for OS/2 to access NTFS as well. DOS cannot access these partitions, but DOS applications running under Windows NT can. File names can be 256 characters long, and volumes size is limited to 8G. NTFS capabilities may be added to Windows 95 in the future.

Zoned Recording - The splitting of cylinders into groups called **zones**, with each successive zone having more and more sectors per track as you move out from the inner radius of the disk. All the cylinders in a particular zone have the same number of sectors per track.