

[Figures are not included in this sample chapter]

A+ Certification Training Guide

- 3 -

Operating Systems

Upon completion of this chapter and its related LabExercises, you should be able to:

- Describe the series of events that occur when power is applied to the system.
- Define multi-user, multi-tasking, and multi-processor operations.
- Explain the function of the system BIOS.
- Discuss the sequence of the events in the POST test.
- Discuss naming conventions as they apply to various types of files.
- List the events that occur during the bootup process.
- Describe the function and purpose of DOS.
- Describe methods of bypassing and correcting inoperable DOS startup sequences.
- Install and configure operating systems to the basic operational level.
- Install and configure application software packages.
- Configure the system through CMOS Setup procedures.
- Create, delete and navigate through directories.
- Find, copy, rename, delete, and move, files.
- Manipulate file attributes in a DOS system.
- Use the AUTOEXEC.BAT and CONFIG.SYS files to optimize system performance.
- Load driver software for devices added to the system.
- Edit the AUTOEXEC.BAT and CONFIG.SYS files for troubleshooting purposes.

Introduction

The general responsibilities of an operating system were described in Chapter 1, "Microcomputer Fundamentals." In this chapter, you will investigate operating systems in greater depth. The first half of the chapter deals with the foundation of the operating system--the BIOS. This topic is discussed in four sections: power-on self-tests and system initialization, booting up to the operating system, system configuration, and BIOS functions.

The second half of the chapter deals with *Disk Operating Systems* (DOS). In this section, the structure of DOS systems is explored along with typical DOS disk organization. The commands and utilities available through the DOS command line are also presented.

Operating Systems

OBJECTIVE: Literally thousands of different operating systems are in use with microcomputers. The complexity of each operating system typically depends on the complexity of the application the microcomputer is designed to fill. The operating system for a fuel mixture controller in an automobile is relatively simple; an operating system for a multi-user computer system that controls many terminals is relatively complex.

The complete operating system for the fuel controller can be stored in a single, small ROM device. It takes control of the unit as soon as power is applied, resets the system, and tests it. During normal operation, the operating system monitors the sensor inputs for accelerator setting, humidity, and so forth, and adjusts the air/fuel mixing valves according to predetermined values stored in ROM. The fuel mixture controller is depicted in Figure 3.1.

FIGURE 3.1 *A simple air/fuel mixture controller.*

In the large, multiple-user system, the operating system is likely to be stored on disk and have sections loaded into RAM when needed. As illustrated in Figure 3.2, this type of operating system must control several pieces of hardware, manage files created and used by various users, provide security for each user's information, and manage communications between different stations. The operating system is also responsible for presenting each station with a user interface that can accept commands and data from the user. This interface can be a command line interpreter or a graphical user interface (GUI).

FIGURE 3.2 *A multi-user system.*

Complex operating systems typically contain several millions of lines of computer instruction. Due to this complexity, large operating systems are typically written in modules that handle the various responsibilities assigned to the system. The operating system for the fuel mixture controller is most likely a single module. However, the operating system for the multiple-user system probably consists of a core module, called the *kernel*, a task manager, a scheduler, a local file manager, and a host of other manager modules.

There are two basic types of operating systems:

- Single-process systems

- Multiple-process systems

In a single-process system, the operating system works with a single task only. These operating systems can operate in *batch mode* or *interactive mode*. In batch mode, the operating system runs one program until it is finished. In interactive mode, the operation of the program can be modified by input from external sources. The simple program presented with the \$1.98 Computer first introduced in Chapter 1, "Microcomputer Fundamentals," is an example of a batch mode operating system. If a mechanism is added to the \$1.98 so that program jumps can be caused by data from an external entry during the execution of the program, it then becomes an interactive system.

In multiple-process systems, the operating system is designed so that it can appear to work on several *tasks* simultaneously. A task is a portion of a program under execution. Computer programs are made up of several tasks that may work alone or as a unit. Tasks, in turn, can be made up of several *threads* that can be worked on separately. A thread is a section of programming that can be time-sliced by the operating system to run at the same time that other threads are being executed.

The multiple-process system breaks the tasks associated with a process into its various threads for execution. Typically, one thread might handle video output, another mouse input, and another output from the printer.

OBJECTIVE: Multiple-process operations can be organized in three different ways:

- Multi-user
- Multi-tasking
- Multi-processor

These three types of operating systems are described in Figure 3.3.

FIGURE 3.3 *Multiple-process operating systems.*

Multi-user and multi-tasking operations give the appearance of simultaneous operation by switching between different tasks in a predetermined order. The multi-user system switches between different users at multiple locations, while multi-tasking systems switch between different applications at a single location. In both cases, the information concerning the first task must be stored and information about the new task loaded each time a task switch occurs. The operating system's scheduler module is responsible for overseeing the switching function.

In a multi-processor operating system, tasks are divided between multiple microprocessors. This type of operation is referred to as *parallel processing*.

Although simple microcomputers store the entire operating system in ROM, most microcomputers use a *bootstrapping* process to load the operating system into RAM. Bootstrapping describes an arrangement in which the operating system is loaded into memory by a smaller program called the *bootstrap loader*. The operating system can be loaded from a ROM chip, a floppy disk, a hard-disk drive, or from another computer. The term *bootstrap* refers to the system pulling itself up by its own bootstraps, because in loading the more powerful operating system files from the disk, it has increased its on-board intelligence considerably. In personal computers, the bootstrap operation is one

of the functions of the ROM BIOS.

Basic Input/Output Systems

OBJECTIVE: PC system boards use one or two IC chips to hold the system's BIOS firmware. The system's memory map reserves memory locations from E0000h to FFFFFh for the system board BIOS routines. These chips contain the programs that handle startup of the system, the changeover to disk-based operations, video and printer output functions, and a *power-on self-test* (POST).

Post Tests and Initialization

OBJECTIVE: The POST test is actually a series of tests that are performed each time the system is turned on. The different tests check the operation of the microprocessor, the keyboard, the video display, the floppy- and hard-disk drive units, as well as both the RAM and ROM memory units.

When the system board is reset, or when power is removed from it, the system begins generating clock pulses when power is restored. This action applies a RESET pulse to the microprocessor, causing it to clear most of its registers to 0. However, it sets the Instruction Pointer register to 0FFF0h and the CS register to F0000h. The first instruction is taken from location FFFF0h. Notice that this address is located in the ROM BIOS program. This is not coincidental.

When a cold boot is performed on the system, the microprocessor must begin taking instructions from this ROM location to initialize the system for operation.

Initial POST Checks

The first instruction that the microprocessor executes causes it to jump to the POST tests where it performs standard tests such as, the *ROM BIOS checksum test* (that verifies that the BIOS program is accurate), the system's various *DRAM tests* (that verify the bits of the memory), as well as testing the system's CMOS RAM (to make certain that its contents have not changed due to a battery failure). During the memory tests, the POST displays a running memory count to show that it is testing and verifying the individual memory locations.

Sequentially, the system's interrupts are disabled, the bits of the microprocessor's flag register are set, and a Read/Write test is performed on each of its internal registers. The test program simply Writes a predetermined bit pattern into each register and then Reads it back to verify the register's operation. After verifying the operation of the microprocessor's registers, the BIOS program begins testing and initializing the rest of the system. It moves forward by inspecting the ROM BIOS chip itself. It does this by performing a checksum test of certain locations on the chip, and comparing the answer with a known value stored in another location.

A checksum test involves adding the values stored in the key locations together. The result is a rounded-off sum of the values. When the checksum test is performed, the sum of the locations is recalculated and compared to the stored value. If they match, no error is assumed to have occurred. If not, an error condition exists and an error message or beep code is produced.

At this point, the program checks to see whether the system is being started from an off condition, or being reset from some other state. When the system is started from an off condition, a *cold boot* is

being performed. However, simultaneously pressing the CTRL, ALT and DEL keys while the system is in operation generates a reset signal in the system and causes it to perform a shortened bootup routine. This operation is referred to as a *warm boot*, and enables the system to be shut down and restarted without turning it off. This function also enables the computer's operation to be switched to another operating system.

If power was applied to the system prior to the occurrence of the RESET signal, some of the POST's memory tests are skipped.

If a cold boot is indicated, the program tests the first 16KB of RAM memory by writing 5 different bit patterns into the memory, and reading them back, to establish the validity of each location. The BIOS startup steps are illustrated in Figure 3.4.

FIGURE 3.4 *The startup sequence.*

System Initialization

If the first 16KB of RAM successfully passes all five of the bit-pattern tests, the BIOS routine initializes the system's intelligent devices. During this part of the program, startup values stored in the ROM chip are moved into the system's programmable devices to make them functional. The BIOS initializes all the system's standard AT-compatible components, such as the interrupt, DMA, keyboard, and video controllers, along with its timer/counter circuits. The program checks the DMA controller by performing a R/W test on each of its internal registers, and then initializes them with startup values.

The program continues by setting up the system's Interrupt Controller. This includes moving the system's interrupt vectors into address locations 00000h through 003FFh. In addition, a R/W test is performed on each of the interrupt controller's internal registers. The routine then causes the controller to mask (disable) all its interrupt inputs, and tests each one to assure that no interrupts occur.

The programming of the interrupt controller is significant because most of the events in a PC-compatible system are interrupt driven. Its operation affects the operation of the computer in every phase from this point forward. Every peripheral or software routine that needs to get special services from the system makes use of the interrupt controller.

Following the initialization of the interrupt controller, the program checks the output of the system's Timer/Counter channels. It does this by counting pulses from the counters for a given period of time, to verify that the proper frequencies are being produced.

If the timer/counter frequencies are correct, the routine initializes and starts the video controller. The program obtains information about the type of display (monochrome, color, or both) being used with the system by reading configuration information from registers in the system's CMOS RAM. After this has been established, the program conducts R/W tests on the video adapter's RAM memory. If the video adapter passes all these tests, the program causes a cursor symbol to be displayed on the monitor. The steps of the initialization process are described in Figure 3.5.

FIGURE 3.5 *System initialization.*

Additional POST Checks

After the display adapter has been checked, the BIOS routine resumes testing the system's on-board memory. First, R/W testing is performed on all the additional RAM on the system board (beyond the first 16KB). In addition, the BIOS executes the system's built-in setup program to configure its Day/Time setting, its hard- and floppy-disk drive types, and the amount of memory actually available to the system.

Following the final memory test, the remaining I/O devices and adapters are tested. The program begins by enabling the keyboard circuitry and checking for a scan code from the keyboard. No scan code indicates that no key has been depressed. The program then proceeds to test the system's parallel printer and RS-232C serial ports. In each case, the test consists of performing R/W tests on each of the port's registers, storing the addresses of functional ports (some ports may not be installed, or in use), and storing time limitations for each port's operation. The steps of the POST process are described in Figure 3.6.

FIGURE 3.6 *Completion of the POST test.*

BIOS Extensions

After the initialization and POST tests are completed, the BIOS checks the area of memory between C0000h and DFFFFh for *BIOS extension* programs. IBM system designers created this memory area so that new or non-standard BIOS routines could be added to the basic BIOS structure. These extended firmware routines match software commands from the system to the hardware they support. Therefore, the software running on the system does not have to be directly compatible with the hardware.

BIOS Extensions are created in 512-byte blocks that must begin at a 2KB marker (for example, C8000h, C8200h, C8400h, C8800h, and so forth), as illustrated in Figure 3.7. A single extension can occupy multiple blocks; however, it can start only at one of the markers. When the main BIOS encounters the special 2-byte extension code at one of the 2KB markers, it tests the block of code and then turns control over to the extension. Upon completion of the extension code, control is passed back to the main BIOS which then checks for an extension marker at the next 2KB marker.

FIGURE 3.7 *BIOS extension blocks.*

Although the extension addresses are memory addresses, the extension code may be located anywhere in the system. In particular, BIOS extensions are often located on expansion cards. The system simply accesses them through the expansion bus.

Advanced video cards contain Video BIOS code, either in a ROM IC, or built directly into the video controller ASIC. The IBM EGA and VGA standards allow for on-board ROM that uses addresses between C0000h and C7FFFh.

Likewise, different types of HDD controller cards contain a BIOS extension IC. The HDD controllers in old XT units had BIOS extensions that used the address space between C8000h and C9FFFh. Some current HDD controllers, such as ESDI and SCSI adapters (described in Chapter 8) reserve memory blocks between C8000h and CBFFFh.

Another type of device that commonly uses the C000h-D000h blocks are network adapter cards. These cards enable the computer to be connected to other computers in the local area. The BIOS extension code on a network card may contain an *initial program load* (IPL) routine that causes the local computer to load up and operate from the operating system of a remote computer. Refer to the "Bootup" section of this chapter and the Networking information in Chapter 11 for more information about these BIOS extensions.

The system can accommodate as many extensions as fit mathematically within the allotted memory area. However, two extension programs cannot be located in the same range of addresses. With this in mind, peripheral manufacturers typically include some method of switching the starting addresses of their BIOS extensions so that they can be set to various markers.

CMOS Setup Routines

OBJECTIVE: Just prior to completing the bootup process, older PCs and PC-XTs checked a set of configuration switches on the system board to determine which types of options were being used with the system. On newer systems, the configuration information is stored on the system board in a battery-powered storage area called the *CMOS RAM*. Newer BIOS enable the user to have access to this configuration information through the Setup utility.

While performing its normal tests and bootup functions, the BIOS program displays a header on the screen and shows the RAM memory count as it is being tested. Immediately following the RAM test count, the BIOS program places a prompt on the monitor screen to tell the user that the CMOS setup program can be accessed by pressing a special key, or a key combination. Typical keys and combinations include: the DEL key, the ESC key, the F2 function key, the CTRL and ESC keys, and the CTRL-ALT-ESC key combination.

Arguably, the most popular BIOS in the world are those from *American Megatrends, Inc.* (AMI). This BIOS uses the DEL key. Other BIOS programs may use different keys, or key combinations, for accessing their setup menus. If the DEL key is not depressed within a predetermined amount of time, the BIOS program continues with the bootup process. However, if the DEL key is pressed during this time, the bootup routine is put on hold and the program displays a "CMOS Setup Selection" screen, similar to the one depicted in Figure 3.8.

FIGURE 3.8 A *CMOS Setup Selection* screen.

Every chipset variation has a specific BIOS designed for it. Therefore, functions specific to the design of system boards are using that chipset. Referring to the example in the figure, any one of three options can be selected from the screen: Return to the bootup process and continue normal operation, Select the Run CMOS Setup routine, or Run a built-in diagnostics program. This particular example is relatively simple. BIOS screens from other manufacturers or for other chipsets may have several options to consider. The example is also unusual in that it possesses a set of on-board diagnostic routines. These can be quite helpful when portions of the system are not functional, but they are not common in the industry.

If you are setting the computer up for the first time or adding new options to the system, it is necessary to run the *CMOS Configuration Setup* program. The values input through the setup utility are stored in the system's CMOS Setup registers. These registers are examined each time the system

is booted up to tell the computer which types of devices are installed.

The AMI Configuration Setup screen is shown in Figure 3.9. Through this screen, the user enters the desired configuration values into the CMOS registers. The cursor on the screen can be moved from item to item using the keyboard's cursor control keys.

FIGURE 3.9 *The CMOS Configuration Setup screen.*

When the cursor is positioned on top of a desired option, the PgUp and PgDn cursor keys can be used to change its value. When all of the proper options have been configured, pressing the ESC key causes the routine to exit the setup screen, update any changes made, and resume the bootup process.

Other BIOS Manufacturers

The Award BIOS from the Award Software company is another widely used BIOS. An Award BIOS Configuration selection menu screen is depicted in Figure 3.10. As the menu indicates, many user-configurable options are built into modern BIOS. Unlike the AMI BIOS, the Award firmware uses the + and - keys to manipulate the settings of menu items displayed on the screen.

FIGURE 3.10 *The Award BIOS Configuration Setup screen.*

The standard CMOS setup screen is very similar to the AMI screen in Figure 3.9. In both examples, the BIOS first presents a screen of basic configuration information. As both figures show, this screen typically includes information about the time and date, microprocessor, system memory organization, floppy-disk drives, hard-disk drives, and video configurations.

A third major BIOS is produced by Phoenix Technologies, Ltd. Its main features are identical to the AMI and Award BIOS. The main screen covers time and date, hard- and floppy-disk drives, and system memory. The Phoenix BIOS uses the F2 function key to enter the Setup function's Main menu, depicted in Figure 3.11. Notice that the select keys for manipulating the Setup program are identified at the bottom of the display.

FIGURE 3.11 *The Phoenix BIOS Configuration Setup screen.*

In most CMOS displays, the total memory does not equal the summation of the base and extended memory. This is because the BIOS reserves 384KB for shadowing purposes.

The other area in this screen that typically requires some effort to set up is the HDD parameters section. All BIOS come with a list of hard drive types that they can support directly. However, they also provide a position for user definable drive settings. Historically, this has been referred to as the "Type 47" entry, but this entry may be located at any number in the list.

Advanced CMOS Setup

A second CMOS Configuration screen, referred to as the "BIOS Features Setup" screen, or "Advanced CMOS Setup" screen provides extended user control over the configuration of the system. A relatively simple Award Features screen is illustrated in Figure 3.12. In this example, several bootup options can be enabled, such as the boot drive sequence and Password enabling. The bootup sequence allows the system to bootup without checking all of the drives in order. This setting may

need to be adjusted to include the A: floppy drive if it becomes impossible to boot the hard drive.

FIGURE 3.12 *The Award BIOS features setup screen.*

The password setting prevents users without the password from accessing the system. If the system has an unknown password, it will be necessary to clear the CMOS. Most system boards have a jumper block that can be shorted to reset the CMOS to its default settings. If this option is used, it will be necessary to reenter the original configuration information.

On Pentium-based system boards, the configuration jumpers and switches for enabling functions have been replaced by BIOS enabling settings. These settings usually include the disk drives, keyboard, and video options, as well as on-board serial and parallel ports. In addition, the user can turn certain sections of the system's RAM on or off for shadowing purposes, as well as establish parity or non-parity memory operation.

The complexity of modern system boards has created a huge number of configuration options for their BIOS. This is reflected in the complexity of their Advanced CMOS Configuration screens. Working in these screens, it is very easy to place the system in a condition where it is unable to respond. Because the problem is at the BIOS level, it is often difficult to get back into the CMOS to correct the problem. Therefore, system designers have included a couple of options to safeguard the system from this condition. In some BIOS, holding down the DEL key throughout the startup erases the CMOS contents and starts from scratch. There may also be jumpers placed on the system board that can be set to start the contents from a bare essentials setting. In either case, it is necessary to rebuild any advanced features in the CMOS configuration afterwards.

BIOS Entry and Exit

Even the Option selection pages for newer BIOS can be complex. A typical Options page is depicted in Figure 3.13. This screen serves as the main menu for entering and exiting the CMOS Setup, as well as for moving between its configuration pages.

FIGURE 3.13 *A complex entry menu.*

BIOS designers have built two options into newer BIOS to help users avoid the complexity of the advanced CMOS configuration settings. These options are *Auto Configuration* and *Default Settings*.

All newer system boards have an auto-configuration mode that takes over most of the setup decisions. This option works well in the majority of applications. Its settings produce an efficient, basic level of operation for standard devices in the system. However, they do not optimize the performance of the system. To do that, it is necessary to turn off the auto configuration feature and insert desired parameters into the configuration table. The auto configuration function typically has two options: Auto Configure with Power-On Defaults and Auto Configure with BIOS Defaults.

Using Power-On defaults for auto configuration loads the most conservative options possible into the system from the BIOS. These settings replace any entered configuration information in the CMOS Setup registers. Then any turbo speed mode is disabled, all memory caching is turned off, and all wait states are set to maximum. If these default values fail, it is an indication of hardware problems such as incorrect jumper settings or bad hardware components.

Using auto configuration with BIOS defaults provides a little more flexibility than the Power-On option. If you have entered an improper configuration setting and cannot determine which setting is causing the problem, this option is suggested. Like the Power-On option, this selection replaces the entered configuration settings with a new set of parameters from the BIOS. Choosing this option is likely to get you back into the CMOS Setup screen so that you can track down the problem. It is also the recommended starting point for optimizing the system's operation.

The many configuration options available in a modern BIOS requires the user to have a good deal of knowledge about the particular function that is being configured. Therefore, an extended discussion of the Advanced CMOS Setup options cannot be conducted at this point. However, such information is covered along with the system component it relates to as the book moves through various system components.

With older Award BIOS the CMOS Setup screen was accessed during bootup by pressing the ESC key. However, newer versions have adopted the same DEL-key strategy used with the AMI units. The exit routine is also different in that you can scroll between several exit options, or press the F10 key to save any changes, and exit the CMOS Setup. Older units required that the F5 key be pressed to confirm the exit selection. The newer units require a Y/N answer to exit.

Some BIOS may also offer a wide array of exit options. Typically, though, the options all involve writing the information away in CMOS and exiting, or not-writing the information to CMOS and exiting. One common mistake in working with CMOS configuration settings is not saving the new settings before exiting. When this happens, the new settings are not stored and the old settings are still in place when the system boots up.

BIOS Error Codes

If an error or setup mismatch is encountered, the BIOS issues an error code, either in message form on the display screen, or in beep-coded form through the system's speaker. Likewise, the Award BIOS produces display and beep-coded error messages when a bootup or configuration problem is encountered during the boot process.

In the case of Plug-and-Play systems, the BIOS must also communicate with the adapter cards located in the expansion slots to determine what their characteristics are. When the system is turned on, the PnP devices involved in the bootup process become active in their default configuration. Other logical devices, not required for bootup, start up in an inactive mode.

Before starting the bootup sequence, the PnP BIOS checks the devices installed in the expansion slots to see what types they are, how they are configured, and in which slots they are attached. It then assigns each adapter a *software handle* (name) and stores the names and configuration information in a RAM table. Next, the BIOS checks the adapter information against the system's basic configuration for resource conflicts. If no conflicts are detected, all the devices required for bootup are activated.

The devices not required for bootup may be configured and activated by the BIOS, or they may simply be configured and left in an inactive state. In either event, the operating system is left with the task of activating the remaining intelligent devices and resolving any resource conflicts that the BIOS detected and could not resolve. If the PnP option is not working for a particular device, or the operating system cannot resolve the remaining resource conflicts, then it will be necessary to use the

Manufacturer's setup instructions to perform manual configurations.

Bootup

OBJECTIVE: If the option to enter the Setup routine is bypassed, or if the routine has been exited, the BIOS begins the process of booting up to the operating system. A simple single operating system, single-disk bootup process is described in Figure 3.14. As you can see, it is a multiple-access operation that uses two different bootstrap routines to locate and load two different boot records.

FIGURE 3.14 *The bootstrap operation.*

The process starts when the BIOS begins looking through the system for a *master boot record*. This record can reside on drive A: or C:, or at any other location. The very first section on any logical DOS disk is called the *boot sector*. It contains information about how the disk is organized. It may also contain the small, optional master boot record that can access a larger, more powerful bootstrap loader program located elsewhere on the disk (normally in an area known as the root directory). In most systems, the master boot record is found at sector-0, head-0, and track-0 of the first logical hard drive. If the disk possesses a master boot record, it can boot up the hardware system to the operating system. The disk is then referred to as a *bootable disk*, or a *system disk*. If not, the disk is simply a *data disk* that can be used for storing information.

Traditionally, BIOS programs searches for the master boot record in floppy-disk drive A: first. If a bootable disk is in the floppy-disk drive, the BIOS executes the *primary bootstrap loader* routine to move the master boot record into RAM and then begin the process of loading the operating system. In the original IBM PC, the BIOS searched in the floppy-disk drive for the boot record. If it was not located there, the BIOS routine turned over control to a BASIC program located in the PC's ROM BIOS IC. In the PC-XT, the BIOS looked first in the floppy drive, or drives, and then in the hard-disk drive. If neither location contained the boot record, the system loaded up the ROM BASIC program. In clone systems, there was no ROM BIOS present to default to when no boot record was found. If the BIOS did not locate the boot record in the floppy or hard drive, it simply displayed a "Non-System Disk or Disk Error," or "ROM BASIC Interpreter Not Found" message on the screen.

In newer systems, the order in which the BIOS searches drives for the boot record is governed by information stored in the system's CMOS configuration RAM. The order can be set to check the floppy drive first and then the hard drive, or to check the hard drive first, or to check the hard drive only.

In a networked system, a bootstrap loader routine can also be located in the ROM extension of a network card as described earlier. When the system checks the BIOS extensions, the bootstrap routine redirects the bootup process to look for a boot record on the disk drive of another computer. Any boot record on the local drive is bypassed. Networking is covered in Chapter 11.

To accomplish the bootup, the BIOS enables the system's non-maskable interrupts and causes a single, short tone to be produced by the speaker circuitry. The single beep indicates that the POST portion of the bootup has been successfully completed.

The next BIOS instruction executes an Interrupt19 Disk Drive service routine. This interrupt routine carries out the *primary bootstrap loader* program, which looks for the master boot record in the first section of the floppy and hard disks. When located, it moves the master boot record into system RAM

to be executed.

The master boot record contains the *secondary bootstrap loader*, also called the *operating system loader*. This routine looks for an *operating system boot record*, typically located on the disk. When the routine finds the record, it loads the bigger boot record into RAM and begins executing it. This boot record brings special operating system files into memory so that they can control the operation of the system (for example, the operating system).

The operating system loader looks for a command processor file. The command processor can belong to any operating system, such as Microsoft MS-DOS, UNIX, IBM PC DOS, Novell Netware, and so forth. The default command processor for DOS is a system file called *COMMAND.COM*. This file interprets the input entered at the DOS prompt. When the bootstrap program finds the command processor, it moves it into system RAM along with the operating system support files.

In the case of Microsoft DOS, the special files in the OS boot record are the *IO.SYS* and *MSDOS.SYS* files. The BIOS recognizes these files by special extensions added to their names (.SYS and .COM).

In the original PC-DOS from IBM, the files were titled *IBMBIO.COM*, *IBMDOS.COM*, and *COMMAND.COM*. This step marks the end of the BIOS routine. The three system files must be found in the root directory (the starting point for any disk-based operations) in order for DOS to boot successfully. The total bootup process is described in Figure 3.15.

FIGURE 3.15 *The bootup process.*

If the system has performed a standard DOS bootup, without any modifications, it should print Date and Time prompts on the monitor screen, followed by the DOS command line "prompt" (A:\>> or C:\>>). The prompt indicates that DOS is operational and the currently active drive is the A: floppy drive, or the C:\ hard drive. Now the DOS software controls the movement of data, and overall operation of the system. DOS enables the basic bootup to be modified through two special utility files, called *CONFIG.SYS* and *AUTOEXEC.BAT*, discussed later in this chapter.

The operation of the system is now in the control of the operator, and whatever software is being used with the system. The system is waiting for the user to do something, such as enter commands and instructions, or run programs from the other two software categories. The user hasn't had anything to do with the operation of the system yet. This is why this type of software is referred to as system software.

BIOS Services

While the system is operating, the BIOS continues to perform several important functions. It contains routines that the operating system calls on to carry out basic services. These services include providing BIOS *interrupt CALLs* (software interrupt routines) for such operations as printer, video, and disk-drive accesses.

The ROM BIOS services are organized into groups identified by interrupt numbers. Each interrupt may cover several different services. When the microprocessor jumps to a particular interrupt, the software calling the interrupt must have already loaded the service number into the microprocessor to tell it which section of the interrupt handler to access.

The most notable BIOS interrupt calls include:

10h--Video Services (16)

13h--Hard and Floppy Drive Services (17 and 11)

14h--Serial Port Services (6)

16h--Keyboard Services (7)

17h--Parallel Printer Port Services (3)

18h--ROM BASIC (old systems)/Network Card Services (newer systems)

19h--Primary Bootstrap Loader 1Ah--Real Time Clock Services

The numbers in parenthesis refer to the number of different services available through the interrupt. For example, 10h--Video Services (16) indicates that 16 different services are available through interrupt call 10.

This list represents just a few of the more notable BIOS interrupts. The most important thing for a technician to remember about BIOS interrupt CALLs is that they form the backbone of the system's operation. The BIOS and DOS are constantly handing control of the system back and forth between themselves as normal system functions are carried out. These BIOS interrupt CALLs are also responsible for most of the drawbacks of the PC system. That is why so much effort is exerted in software to work around them. Advanced operating systems implement newer methods of handling system functions just to avoid handing control over to the BIOS interrupts.

Some older PCs have trouble supporting newer hardware because the BIOS does not support the new item. To correct this situation, it is usually necessary to load a software driver program to support the device. Another possibility is to replace the BIOS with an improved version. This operation is not performed often because an upgraded BIOS must be compatible with the older chip set.

MS-DOS

MS-DOS is a disk operating system for IBM PC-compatible computers. It is easily the most popular operating system in the world. As with any other operating system, its function is to oversee operation of the system by providing support for executing programs, controlling I/O devices, handling errors, and providing the user interface. MS-DOS is a disk-based, single user, single task operating system.

The main portions of MS-DOS are the IO.SYS, MSDOS.SYS, and COMMAND.COM mentioned earlier. The IO.SYS and MSDOS.SYS files are special, hidden system files that do not show up in a normal directory listing. The IO.SYS file implements the MS-DOS default control programs (referred to as device drivers) for various hardware components. These include:

- Boot disk drive

- Console display and keyboard
- System's time-of-day clock
- Parallel and serial communications port

Conversely, the MSDOS.SYS file provides default support features for software applications. These features include:

- Memory management
- Character input and output
- Real time clock access
- File and record management
- Execution of other programs

The COMMAND.COM command interpreter contains the operating system's most frequently used commands. When a DOS command is entered at the DOS prompt, the COMMAND.COM program examines it to see whether it is an *internal* DOS command, or an *external* DOS command. Internal commands are understood directly by COMMAND.COM, while external commands are stored in a directory called DOS. If it is one of the internal commands, the COMMAND.COM file can execute it immediately. If not, COMMAND.COM looks in the \DOS directory for the command program.

Likewise, when DOS runs an application, COMMAND.COM finds the program, loads it into memory, and then gives it control of the system. When the program is shut down, it passes control back to the command interpreter.

The remainder of the operating system is comprised of utility programs to carry out DOS operations such as formatting disks (Format), printing files (Print), and copying files (XCOPY).

MS-DOS Structure

It is important to consider that MS-DOS is a *disk* operating system. Therefore, you must understand how DOS organizes disks. The DOS organizational structure is typically described as being like a common office file cabinet, similar to the one depicted in Figure 3.16. Think of DOS as the filing cabinet structure. Our example has four drawers that can be opened. Think of these as *disk drives* labeled A, B, C/D, and E. Inside each drawer are hanging folders that can hold different types of items. Think of these as *directories*. The hanging folders may contain different types of items or other individual folders. Think of these individual folders as *sub-directories*. For organizational purposes, each hanging folder and each individual folder must have a unique label on it.

FIGURE 3.16 DOS organization.

Inside each hanging folder or individual folder are the items being stored. In a real filing cabinet, these items in the folders are usually documents of different types. However, pictures and tapes and other items related to the folder can also be stored in them.

Think of the items inside the folders as *files*. Disk-based systems manage data blocks by giving them filenames. Recall that a file is simply a block of logically related data, given a single name, and treated as a single unit. Like the contents of the folders, files can be programs, documents, drawings or other illustrations, sound files, and so on.

To find an item in the cabinet, you simply need to know which drawer, hanging folder and folder it is located in. This concept can be translated directly to the computer system. To locate a particular file, you simply need to know in which drive, directory, and sub-directory it is located. In MS-DOS the *path* to any file in the system can be written as a direction to the computer so that it knows where to find the file toward which it is being directed. This format for specifying a path is as follows:

```
C:\directory name\subdirectory name\file name
```

where the C: specifies the C disk drive. The directory, sub-directory, and file names would naturally be replaced by their real names. The *back slashes* (\) after each item indicate the presence of a directory or sub-directory. The first slash indicates a special directory, called the *root directory*, which is present on all DOS disks.

If the direction is to a file, the filename is always placed at the end of the path. MS-DOS allows for a basic file name of up to eight characters. It also allows for an extension of up to three characters. The extension is separated from the main portion of the file name by a period and is normally used to identify what type of file it is (that is the file name *file1.ltr* could be used to identify a letter created by a word processor).

NOTE: File name extensions are not actually required for most files. However, they become helpful in sorting between files in a congested system. You should be aware that the operating system reserves some three letter combinations, such as .COM and .SYS, for its own use. More information about file names and extensions is presented in the subsequent section concerning file-level DOS commands.

DOS Disk Structure

OBJECTIVE: It is also important to understand how DOS sees disks. In the earlier section on booting up, it was mentioned that the first area on each DOS disk is the boot sector. Although all DOS disks have this sector, they do not all have the optional master boot record located in the sector. Only those disks created to be bootable disks have this record.

File Allocation Tables

The second section of a DOS disk is an area referred to as the *File Allocation Table* (FAT). This area is a table of information about the condition of the disk. Basically, the system logs the use of the space on the disk in this table. In older versions of DOS, the amount of space dedicated to tracking the sectors on the disk was 16 bits. Therefore, only 65,536 sectors could be accounted for. This parameter limited the size of a DOS partition to 32MB (33,554,432 bytes).

To more effectively manage the space on the disk, newer versions of DOS divide the disk into groups of logically-related sectors, called *allocation units*, or *clusters*.

As described in Chapter 1, "Microcomputer Fundamentals," the sectors on a DOS disk hold 512 bytes each. On the other hand, files can be any length. Therefore, a single file may occupy several sectors on the disk. The DOS disk routine breaks the file into sector-sized chunks and stores it in a cluster of sectors. In this manner, DOS uses the cluster to track files rather than sectors. Because the file allocation table has to handle information for a cluster only, rather than for each sector, the number of files that can be tracked in a given length table is greatly increased.

The organization of a typical FAT is described in Table 3.1. The first two entries are reserved for DOS information. Each sector after that holds a value. Each value may represent one of three conditions. A value of 0 indicates that the cluster is empty and can be used for storage. Any number besides 0 or FFFh indicates that the cluster contains data and the number provides the location of the next cluster in a chain of clusters. Finally, a value of FFFh (or FFFFh in a 16-bit entry) indicates the end of a cluster chain.

TABLE 3.1 FILE ALLOCATION TABLE STRUCTURE.

Cluster Number	Contents
Cluster 0	Reserved for DOS
Cluster 1	Reserved for DOS
Cluster 2	3 (contains data go to cluster 3)
Cluster 3	4 (contains data go to cluster 4)
Cluster 4	7 (contains data go to cluster 7)
Cluster 5	0 (free space)
Cluster 6	0 (free space)
Cluster 7	8 (contains data go to cluster 8)
Cluster 8	FFFh (end cluster chain)
Cluster 9	0 (free)
**	
Cluster x	0 (free)
Cluster y	0 (free)
Cluster z	0 (free)

On floppy disks, common cluster sizes are one or two sectors long. With hard disks, the cluster size may vary from 1 to 16 sectors in length. The FAT keeps track of which clusters are used and which ones are free. It contains a 12- or 16-byte entry for each cluster on the disk. The 12-byte entries are used with floppy disks and hard disks that are smaller than 17MB. The 16-byte entries are employed with hard-disk drives larger than 17MB. Obviously, the larger entries enable the FAT to manage more clusters.

In version-b of Windows 95, also referred to as *OSR2*, Microsoft supplied a 32-bit file allocation table system called *FAT32* to make efficient use of large hard drives (larger than 2GB). Under the previous FAT structure, large drives used large partitions, which, in turn, required large cluster sizes and wasted a lot of disk space.

The FAT32 format in OSR2 supports hard drives up to 2TB (terabytes) in size. FAT 32 uses 4KB

cluster sizes for partitions up to 8GB in size.

In free clusters, a value of zero is recorded. In used clusters, the cluster number is stored. In cases where the file requires multiple clusters, the FAT entry for the first cluster holds the cluster number for the next cluster used to store the file. Each subsequent cluster entry has the number of the next cluster used by the file. The final cluster entry contains an end-of-file marker code that tells the system that the end of the file has been reached.

These cluster *links* enable DOS to store and retrieve virtually any size file that fits on the disk. However, the loss of any link makes it impossible to retrieve the file and use it. If the FAT becomes corrupted, chained files can become *cross-linked* with each other, making them useless. For this reason, two complete copies of the FAT are stored consecutively on the disk under the DOS disk structure. The first copy is the normal working copy, while the second FAT is used as a backup measure in case the content of the first FAT becomes corrupted.

The Root Directory

The next section following the FAT tables is the disk's *Root Directory*. This is a special directory that is present on every DOS disk. It is the main directory of every logical disk, and serves as the starting point for organizing information on the disk. The location of every directory, subdirectory, and file on the disk is recorded in this table.

Each directory and subdirectory (including the root directory) can hold up to 512, 32-byte entries that describe each of the files in it. The first 8 bytes contain the file's name, followed by three bytes for its filename extension.

The next eleven bytes define the file's *attributes*. Attributes for DOS files include:

- Read Only
- System File
- Volume Label
- Subdirectory Entry
- Archive (backup) status

Two bytes are used to record the time the file was created or last modified. This is followed by two additional bytes that record the date the file was created or last modified.

The final four bytes are divided equally between the value for the *starting cluster number* and a *byte count number* for the file. Unlike the other information in the directory, the information associated with the last four bytes is not displayed when a directory listing is displayed on the screen.

Because each root directory entry is 32 bytes long, each disk sector can hold 16 entries. Consequently, the number of files or directories that can be listed in the root directory is dependent upon how many disk sectors are allocated to it. On a hard-disk drive, 32 sectors are normally set aside for the root directory. Therefore, the root directory for such a disk can accommodate up to 512

entries. Atypical 3-1/2" 1.44MB floppy has 16 sectors reserved for the root directory and can hold up to 224 entries.

Figure 3.17 describes the organization of a DOS disk and illustrates the position of the Boot Sector, File Allocation Tables, and the Root Directory. The remainder of the disk is dedicated to data storage. On a floppy disk the logical structure normally has a group of files located under the root directory. Directory structures can be created on a floppy, but due to their relatively small capacity, this is not normally done. However, a hard drive is another matter. With hard drives, it is normal to organize the disk into directories and subdirectories as described earlier in this chapter.

FIGURE 3.17 *DOS disk organization.*

Technically, every directory on a disk is a sub-directory of the root directory. All additional directories branch out from the root directory in a tree-like fashion. Therefore, a graphical representation of the disk drive's directory organization is called a *Directory Tree*. Figure 3.18 depicts the directory organization of a typical hard drive.

FIGURE 3.18 *The DOS directory tree structure.*

Under DOS, hard-disk drives can be divided into multiple *logical drives*. This operation is referred to as *partitioning* the drive. With earlier versions of DOS this became necessary as the capacity of hard drives exceeded the capability of the FAT to track all the possible sectors.

When a second logical drive is created on the hard disk, another boot sector, file allocation table and root directory is created. DOS sees this new structure on the hard drive as a completely new disk. Therefore it must have its own drive letter assigned to it.

In some applications, partitioning is popular because the system can be booted up to different operating systems. Because each partition contains its own boot sector, FAT, and Root Directory, each partition can be set up to hold and boot up a different operating system.

DOS Command Line

OBJECTIVE: The operating system is responsible for providing the user interface. The main user interface for DOS is the command line. The command line is the space immediately following the DOS prompt on the screen. The MS-DOS prompt for using the C: hard-disk drive as the active directory is displayed in Figure 3.19.

FIGURE 3.19 *The DOS prompt.*

From the DOS prompt, all DOS functions can be entered and executed. Many programs are capable of being started from this prompt. These files can be discerned by their file name extensions. Files with .COM, .EXE, or .BAT extensions can be started directly from the prompt. The .COM and .EXE file extensions are reserved by DOS and can be generated by only programs that can correctly configure them. .BAT files are simply ASCII text files that have been generated using DOS functions. Because they contain DOS commands mixed with .COM and .EXE files, DOS can execute .BAT files from the command line.

Programs with other types of extensions must be *associated* with one of these three file types to be

operated. The user can operate application software packages such as graphical user interfaces, word processors, business packages, data communications package, and user programming languages (such as QBASIC and DEBUG). As an example, the core component of a word processor could be a file called WORDPRO.EXE. Document files produced by word processors are normally given filename extensions of .DOC (for document) or .TXT (for text file). To view a document electronically, you first need to run the executable file and then use its features to load up, format, and display the document. Likewise, a BASIC file normally has an extension of .BAS assigned to it. To execute a file with this extension, it is necessary to run a BASIC interpreter, such as QBASIC.EXE, and then use it to load the .BAS file and then run it.

The user can also type DOS commands on the command line to perform DOS functions. These commands can be grouped into drive level commands, directory-level commands, and file-level commands. The format for using DOS commands is:

```
COMMAND (space) SOURCE location (space) DESTINATION location
```

```
COMMAND (space) location
```

```
COMMAND
```

The first example illustrates how DOS operations that involve a source and a final destination, such as moving a file from one place to another, are entered. The second example illustrates how single-location DOS operations, such as formatting a floppy disk in a particular disk drive, are specified. The final example applies to DOS commands that occur in a *default location*, such as obtaining a listing of the files on the current disk drive.

Many DOS commands can be modified by placing one or more software *switches* at the end of the basic command. A switch is added to the command by adding a space, a *fore-slash (/)*, and a single letter:

```
COMMAND (space) option /switch
```

NOTE: Common DOS command switches include /P for page, /W for wide format, and /S for system. Different switches are used to modify different DOS commands. In each case, the DOS User's Guide should be consulted for switch definitions available with each command.

Drives and Disks

OBJECTIVE: It is important to note that each disk drive in the system is identified by DOS with a single-letter name (such as A:), and that this name must be specified when giving the system commands, so that they are carried out using the proper drive. The format for specifying which drive is to perform a DOS operation calls for the presence of the drive's identifier letter in the command, followed by a colon (that is, A: or C:).

Figure 3.19 shows how the various disk drives are seen by a typical, stand-alone system. DOS assigns the letters A: and B: to the first and second floppy drives. Multiple hard-disk drive units can be installed in the system unit, along with the floppy drive(s). DOS recognizes a single hard-disk unit in the system as DRIVE C:. DOS utilities can also be used to divide a single, physical hard-disk drive

into two or more volumes that the system recognizes as *Logical Drives* C:, D:, and so forth. This is known as *partitioning* the drive.

NOTE: Figure 3.20 shows a CD-ROM drive as Drive D: because this is becoming the most common PC configuration. In the case of networked systems, logical drive letters may be extended to define up to Z drives. These drives are actually the hard drives located in remote computers. The operating system in the local machine treats them as additional logical drives (for example, F, G, and so forth).

FIGURE 3.20 *The system's disk drives.*

Conversely, a second hard-disk drive can be added to the system and set up as logical drive D:. It may also be partitioned into smaller logical drives that the system recognizes as drives E:, F:, and so on. Logical drives and disk partitioning are covered in Chapter 8.

Some DOS operations are simplified by enabling the system to choose the location for the command to be carried out through the use of *default settings* (special predetermined settings that are automatically used by the system when no specific directions are given to change the setting). These settings are remembered in DOS and used by the system when the operator does not specify a particular location for events to happen. The default setting in your system is the A: drive. In systems with two or more drives, it is imperative that the user specify exactly where the action called for is to occur.

The following DOS commands pertain to drive-level operations. They must be typed at the DOS prompt, and carry out the instruction along with any drive modifiers given.

DISKCOPY: This command is used to make a duplicate of a disk. The DISKCOPY operation is normally used to make backup disks, and is usually followed by a DISKCOMP operation:

```
C:\>DISKCOPY A: B:
```

DISKCOMP: This command is used to compare the contents of two disks. It compares the data on the disks not only to see that they are alike, but also to see that the data is located in the same place on both disks. The DISKCOMP operation is normally used to check backup disks, and usually follows a DISKCOPY operation:

```
C:\>DISKCOMP A: B:
```

FORMAT: This command is used to prepare a new floppy disk for use. Actual data locations are marked off on the disk for the tracks and sectors, and bad sectors are marked. In addition, the directory is established on the disk. New disks must be formatted before they can be used.

C:\>FORMAT B: is used even in a single-drive system. The system issues prompts to insert the proper disks at the correct times. A self-booting disk can be created by using a /S modifier (for system files) at the end of the normal FORMAT command.

C:\>FORMAT B: /S causes three system files (boot files) to be copied onto the disk after it has been formatted. The new disk now boots up without a DOS disk.

`C:\>FORMAT A: /Q` causes the system to perform a quick format operation on the disk. This amounts to removing the FAT and root directory from the disk.

Directories

OBJECTIVE: As mentioned earlier, in hard drive-based systems it is common to organize related programs and data into areas called *Directories*. This makes them easier to find and use, because modern hard drives are capable of holding large amounts of information. As described earlier, most directories can hold up to 512 directory or file name entries.

It would be difficult to work with directories if you could not know which one you were currently using. The DOS prompt can be set up to display which directory is being used. This particular directory is referred to as the *current* or *working directory* (for example, `C:\DOS\forms` indicates that you are working with programs located in a sub-directory of the DOS directory named forms). The first *back slash* (\) represents the root directory on the C hard drive. The presence of two dots (..) near the top of a directory listing acts to identify it as a sub-directory. These dots indicate the presence of a *parent directory* above the currently active subdirectory.

The following DOS commands are directory-based. The format for using them is identical to disk-related commands discussed earlier.

DIR: The *Directory* command gives a listing of the files on the disk that is in the drive indicated by the drive specifier.

`C:\>DIR` or `DIR B:` (If `DIR` is used without any drive specifier, the contents of the drive indicated by the prompt are displayed.) The command may also be used with modifiers to alter the way in which the directory is displayed.

`C:\>DIR/W` displays the entire directory at one time across the width of the display.

`C:\>DIR/P` displays the contents of the directory one page at a time. You must press a key to advance to the next display page.

MKDIR (MD): creates a new directory in an indicated spot in the directory tree structure.

`C:\>MD C:\DOS\XXX` creates a new subdirectory named XXX in the path that includes the ROOT directory (C:\) and the DOS directory.

CHDIR (CD): Changes the location of the active directory to a position specified with the command.

`C:\>CD C:\DOS` Changes the working directory from the C: root directory to the C:\DOS directory.

RMDIR (RD): Remove directory erases the directory specified in the command. You cannot remove a directory until it is empty and you cannot remove the directory if it is currently active.

`C:\>RD C:\DOS\forms` removes the DOS sub-directory "forms," provided it is empty.

PROMPT: Changes the appearance of the DOS prompt.

`C:\>PROMPT PG` causes the form of the prompt to change from simply `C:` to `C:\>` and causes the complete path from the main directory to the current directory to be displayed at the DOS prompt (for example, `C:\DOS>`).

TREE: Lists all the directory, and subdirectory, names on a specified disk.

`C:\>TREE C:` displays a graphical representation of the organization of the C hard drive.

DELTREE: Removes a selected directory and all the files and sub-directories below it.

`C:\>DELTREE C:\DOS\DRIVER\MOUSE` deletes the sub-directory "Mouse" and any sub-directories it may have.

Files and Filenames

OBJECTIVE: Disk-based systems store and handle related pieces of information in groups called *files*. The system recognizes and keeps track of the different files in the system by their filenames. Therefore, each file in the system is required to have a filename that is different from that of any other file in the directory. If two files having the same name were present in the system at the same time, the computer would become confused and fail to operate properly, because it could not tell on which file it was supposed to work. Each time you create a new file of information, you are required to give it a unique filename by which DOS can identify it.

Under DOS, you must remember a few rules when you create new filenames. The filename consists of two parts: a *name* and an *extension*. The filename is a combination of alphanumeric characters and is between one and eight characters in length. The extension is an optional addition to the name that begins with a period, and is followed by between one and three characters. Extensions are not required on filenames, but they often prove useful in describing the contents of a file, or in identifying different versions of the same file. If a filename that already exists is used to store another file, the computer writes the information in the new file over that of the old file, assuming that they are both the same. Therefore, only the new file still exists. The information in the old file is lost.

Many software packages automatically generate filename extensions for files they create. The software does this so that other parts of the program, which may work with the same file, can identify the file's source location or its form.

In any event, you should remember the following items when assigning and using filenames:

- All files must have a filename.
- All filenames must be different than any other filename in the system, or on the disk presently in use.
- Filenames are up to 8 characters long with an optional 3-character extension (separated from the basic filename by a period).
- When using a filename in a command, you must also use its extension, if one exists.

- Some special characters are not allowed in filenames. These are the brackets, colon, semicolon, plus sign, equals sign, back slash, fore-slash, and comma ([,], :, ;, +, =, \, /, and ,).
- When telling DOS where to carry out a command, you must tell it on which disk drive the operation is to be performed. The drive must be specified by its letter name followed by a colon (for example, A:, B:, C:, and so forth).
- The complete and proper way to specify a file calls for the drive specifier, the filename, and the filename extension, in that order (for example, B:filename.ext).

The following DOS commands are used to manipulate specific files. The format for using them is identical to the disk-related commands discussed earlier. However, the command must include the filename and its extension at the end of the directory path. Depending on the operation, the complete path may be required, or a default to the currently active drive is assumed.

COPY: The file copy command copies a specified file from one place (disk or directory) to another.

```
C:\>COPY A:filename.ext B:filename.ext
```

C:\>COPY A:filename.ext B: is used if the file is to have the same name in its new location; the second filename specifier can be omitted.

In a single-drive system, it is necessary to switch disks in the middle of the operation. (Notice that the drive B specifier is used even though only drive A: is present.) Fortunately, the DOS produces a prompt message to tell you when to put the target disk in the drive. This is not required in a two-drive system and no prompt is given. The transfer can be specified in any direction desired, as in:

```
C:\>COPY B:filename.ext A:
```

The only thing to keep in mind in this situation is to place the source disk in drive B and the target disk in drive A: before entering the command.

XCOPY: This command copies all the files in a directory, along with any sub-directories and their files. This command is particularly useful in copying files and directories between disks with different formats (i.e., from a 1.2MB disk to a 1.44MB disk:

```
C:\>XCOPY A: B: /s
```

This command copies all the files and directories from the disk in drive A: (except hidden and system files) to the disk in drive B:. The /s switch instructs the XCOPY command to copy directories and sub-directories.

DEL or ERASE: When this command is typed in at the DOS prompt, it enables the user to remove unwanted files from the disk:

```
C:\>DEL filename.ext  
C:\>ERASE B:filename.ext
```

A great deal of care should be taken when using this command. If a file is erased accidentally, it may not be retrievable.

REN: Enables the user to change the name or extension of a filename:

```
C:\>REN A:filename.ext newname.ext
```

Using this command does not change the contents of the file, only its name. The original filename (but not the file) is deleted. If you wish to retain the original file and filename, a copy command, using different filenames, can be used:

```
C:\>COPY A:filename.ext B:newname.ext
```

TYPE: Shows the contents of a designated file on the monitor screen.

```
C:\>TYPE AUTOEXEC.BAT displays the contents of the autoexec.bat file
```

FC: This file-compare command compares two files to see whether they are the same. This operation is normally performed after a file copy has been performed to ensure that the file was duplicated and located correctly:

```
C:\>FC A:filename.ext B:
```

If the filename was changed during the copy operation, the command would have to be typed as:

```
C:\>FC A:filename.ext B:newname.ext
```

ATTRIB: Changes file attributes such as read-only (+R or -R), archive (+A or -A), system (+S or -S), and hidden (+H or -H). The + and - signs are chosen to add or subtract the attribute from the file.

```
C:\>ATTRIB +R C:\DOS\memos.doc sets the file memos.doc as a read-only file.
```

Read-only attributes protect the file from accidentally being overwritten. Similarly, one of the main reasons for giving a file a Hidden attribute is to prevent it from accidentally being erased. The System attribute is reserved for use by the operating system and marks the file as a system file.

SETVER: This command sets the DOS version number that the system reports to an application. Programs designed for previous DOS versions may not operate correctly under newer versions unless the version has been set correctly:

```
C:\>SETVER C:
```

This entry causes all the files on the C: drive to be listed in the DOS version table. If the current DOS version is not known, typing VER at the DOS prompt displays it on the screen. These commands are particularly useful in networking operations where multiple computers are connected together to share information. In these applications, several versions of DOS may exist on different machines attached to the network.

DOS Shortcuts

OBJECTIVE: DOS provides some command line shortcuts through the keyboard's function keys. Some of the most notable are the F1 and F3 function keys. The F1 key brings the preceding command

back from the command line buffer, one character at a time. Likewise, the F3 key brings back the entire preceding command, through a single keystroke.

When using filenames in DOS command line operations, the filename appears at the end of the directory path in the source and destination locations. The * notation is called a *wild card* and enables operations to be performed with only partial source or destination information. Using the notation as *.* tells the software to perform the designated command on any file found on the disk using any filename and extension.

A question mark (?) can be used as a wild card, to represent a single character in a DOS name or extension. Multiple question marks can be used to represent multiple characters in a filename or extension.

Data from a DOS command can be modified to fit a prescribed output format, through the use of filter commands. The main filter commands are More, Find, and Sort. The filter command is preceded by a *pipe symbol* (|) on the command line, when output from another DOS command is to be modified. For example, to view the contents of a batch file that is longer than the screen display can present at one time, type **TYPE C:\xxx.bat |more**. If the information to be modified is derived from another file, the less than (<<) symbol is used.

The Find command searches through files and commands for specified characters. Likewise, the Sort command presents files in alphabetical order.

DOS I/O Commands

OBJECTIVE: The DOS mode command is used to configure the system's I/O devices. These devices include the parallel and serial ports as well as the monitor display and the keyboard.

DOS keeps track of its different parallel and serial ports by assigning them logical designations during the initialization phase of the system bootup. A parallel port is designated as an *LPT port* and can be assigned to the system as LPT1, LPT2, or LPT3. Likewise, serial ports are designated as *COM*, or *communications ports*. Any of the system's serial ports can be configured as COM1, COM2, COM3, or COM4. However, the serial ports cannot share the same COM port designation.

The format for using the mode command to configure the parallel printer port is as follows:

```
mode LPT1:n,m,P
```

where n is the number of characters per line across the page, m is the number of lines of print down the page, and the value of P sets up continuous retry on timeout errors (errors that occur when actions do not occur during a prescribed amount of time). The value of n can be set to 80 or 132 characters. Common values for m are 6 or 8 lines.

The mode command is also used to set up the serial ports. The format for the serial port is:

```
mode COMn:baud,parity,databits,stopbits,P
```

where n represents one of the four serial ports in the system. Baud is the transmission rate at which the port sends and receives data. Common values for this variable are 110, 150, 300, 600, 1200, 2400,

4800, 9600 and 19,200. Only the first two digits of the speed rating are placed in the command (for example, 9600 = 96).

Parity describes the type of error checking used by the port (error checking and parity are discussed in Chapter 7, "Input/Output," and Chapter 11, "Data Communications"). Parity can be set to E for even, O for odd, or N for none.

The data bit entry tells the receiver how many data bits to expect. The usual setting for data bits is 7; however, an 8-bit data word can also be selected. Likewise, different numbers of special stop bits can be used in serial communications to mark the end of a character or message. Typical stop bit values can be 1 or 2. The P value is used to indicate whether the port is being used with a serial printer or some other serial device. If a value is included for P, the system assumes that it is connected to a serial printer and performs continuous retries on timeout errors.

In addition to setting up the operation of the system's I/O ports, the mode command can be used to alter the output format of the video display. The format for using the mode command to alter the output on the video display is:

```
mode n,m,T
```

where n is the number of columns and color selection for the display. Typical values for this variable are 40, 80, BW40, BW80, CO40, CO80, and mono. The 40 and 80 values indicate the numbers of characters on a text line. The BW40 and BW80 options also indicate the number of characters per line, but include a reference to the color graphics adapter with color turned off. Conversely, the CO40 and CO80 values indicate the color graphics adapter with color enabled. Mono indicates a monochrome display adapter.

The m variable can be set to r (for right shift), to l (for left shift). If the T value is present in the command, a test pattern is presented on the screen so that it can be aligned properly.

To use the mode command to set the display mode, a device statement must be included in the CONFIG.SYS file for the ANSI.SYS device driver.

Finally, the mode command can be used to shift data from one output port to another. As an example, it is possible to shift data intended for a serial port to the parallel port. This is a quick and useful troubleshooting tool when working with ports. If data intended for a suspect port can be successfully re-directed to another port, then a hardware problem with the first port is indicated. An example of re-directing data from one port to another is:

```
mode LPT1:=COM2
```

This example re-directs data intended for the first parallel port to the second parallel port.

The mode command can be used inside the AUTOEXEC.BAT file to automatically reconfigure the system at startup.

DOS Utilities

OBJECTIVE: A subclass of system software, called *utilities*, can be used to perform some basic

system operations. These programs enable the system to be optimized for operations in particular functions or with different options.

In the DOS operating system, two of these utilities, called the CONFIG.SYS and AUTOEXEC.BAT files, can be included in the DOS bootup process. As the system moves through the bootup procedure, the BIOS checks in the root directory of the boot disk for the presence of a file named CONFIG.SYS. Afterwards, it searches for the COMMAND.COM interpreter, and finally looks in the root directory for the AUTOEXEC.BAT file. In particular, the CONFIG.SYS and AUTOEXEC.BAT files play key roles in optimizing the system's memory and disk drive usage. This operation can be summarized as follows:

1. BIOS performs INT19 to search drives for master boot record.
2. Primary Bootstrap Loader moves master boot record into memory.
3. System executes Secondary Bootstrap Loader from master boot record.
4. Secondary Bootstrap loader moves IO.SYS and MSDOS.SYS into memory.
5. DOS checks for CONFIG.SYS file in root directory.
6. If CONFIG.SYS is found, DOS reconfigures system.
7. DOS loads COMMAND.COM.
8. COMMAND.COM checks for the AUTOEXEC.BAT file in the root directory.
9. If the AUTOEXEC.BAT file is found, COMMAND.COM carries out the commands found in the file.
10. If no AUTOEXEC.BAT file is found, COMMAND.COM displays the DOS Time and Date prompt as describe earlier.

DOS Memory

OBJECTIVE: To understand how the CONFIG.SYS and AUTOEXEC.BAT files improve the performance of the system, you must understand how DOS views memory and why.

The original DOS version was constructed in two sections. The first 640KBs of memory was reserved for use by DOS and its programs. The remaining section was reserved for use by the BIOS and the system's peripherals (such as the video card, the hard drive controller card, and so forth). This arrangement utilized the entire 1MB addressing range of the 8088 microprocessor.

As more powerful microprocessors entered to market (80286 microprocessors can access up to 16MB of memory, and the 80386 and 80486 can handle up to 4GB of memory), DOS retained the limitations imposed on it by the original version to remain compatible with older machines and software.

Special add-on programs called *memory managers* have been created to enable DOS to access and

use the additional memory available to more powerful microprocessors.

Basic Memory Organization

Every computer has a memory organization plan called a *memory map*. A simplified memory map, showing RAM, ROM, and I/O address allocations, is shown in Figure 3.21.

FIGURE 3.21 *A computer memory map.*

When the original PC was designed, there were certain decisions made in dividing up the 8088's 1MB of memory address space. The Intel microprocessors have a separate memory map for I/O addresses. These decisions were implemented by the original DOS program. By necessity, these decisions carried over into the address allocations of all DOS-based PC-compatible's.

Basically, DOS can recognize the following classifications of memory: Conventional Memory, Upper Memory Blocks, High Memory Area, Expanded Memory, Extended Memory, and Virtual Memory.

Conventional Memory The conventional memory area is divided into two sections referred to as *Base Memory* and the *Upper Memory Area (UMA)*. These sections are illustrated in Figure 3.22. Base memory occupies the first 640KBs of addresses while the remaining 384KBs is referred to as Upper Memory.

FIGURE 3.22 *Conventional memory.*

Base memory (locations 00000h through 9FFFFh) is the standard memory area for all PC-compatible systems. It traditionally holds DOS, interrupt vector tables, and relocated ROM BIOS tables. The remaining space in the base memory area is referred to as DOS Program Memory. Programs written to operate under PC- or MS-DOS use this area for program storage and execution.

The Upper Memory Area occupies the 384KB portion of the PC's address space from A0000h to FFFFFh. This space is segmented into 64KB *Upper Memory Block* regions, as illustrated in Figure 3.23. Although the addresses are allocated, no actual memory is here. The Upper Memory Area was originally dedicated to different forms of video display memory and ROM-based functions. However, many advanced systems reserve space in this area to incorporate a memory-usage scheme called *Shadow RAM* to improve the overall performance of the computer.

FIGURE 3.23 *Upper memory blocks of the UMA.*

With this feature, the contents of the ROM BIOS and/or Video BIOS are rewritten (shadowed) into Upper Memory Area. This scheme enables the system to operate faster when application software makes use of any of the BIOS' CALL routines. Rather than accessing an IC ROM device, which takes up to 4 Wait States to complete, BIOS calls are redirected by the shadow feature to the same information located in fast, 0 Wait State DRAM devices. Some benchmark tests have shown performance increases between 300 and 400% in systems where the shadow feature is used.

Extended Memory With the advent of the 80286 microprocessor and its protected operating mode, it became possible to access physical memory locations beyond the 1MB limit of the 8088. Memory above this address is generally referred to as *Extended Memory*. With the 286 microprocessor, this adds up to an additional 15MB of RAM for a total of 16MB (24 bit address). Extended Memory is illustrated in Figure 3.24.

FIGURE 3.24 *Extended memory.*

Even though the 80286 could physically access this type of memory using its special addressing mode, it was impossible for application programs to access it at the time. This was due to the 640k DOS limit imposed by earlier architectures. Extended memory could range up to 4GB in 80386- and 80486-based computers (32 bit address). It was not that software couldn't access memory at these addresses, it was simply a matter of DOS not having the capability to enable it.

Applications programs can be written specifically to take advantage of these memory locations, but few are. Operating systems, such as Microsoft DOS versions beyond 4.0 and Windows versions beyond 3.0, as well as IBM's OS/2 operating system, can take full advantage of extended memory through the protected addressing modes of the more advanced microprocessors. This capability to manage higher memory enables the system to free up more of the base memory area for use by applications programs.

The DOS versions above 4.0 contain a memory management program called HIMEM.SYS that manages extended memory above the 1024k level. This utility operates under the Microsoft *Extended Memory Specification* (XMS). When the utility is loaded into memory, it shifts most of the operating system functions into an area known as the *High Memory Area* (HMA) of extended memory. The HMA takes up the first 64KB of addresses above the 1MB boundary and is a result of a quirk in the design of advanced Intel microprocessors.

The HIMEM function is activated by adding a line of instruction to the system's CONFIG.SYS file so that it is executed when the computer is booted. When the HIMEM utility is encountered, the program assumes control of the system's *A20 Interrupt Handler* routine. This function is part of the BIOS program and takes control of the system's A20 address line when activated.

The A20 Interrupt Handler is located at BIOS interrupt INT15 and is used to transfer data blocks of up to 64KBs in length between the system and extended memory. The INT15 function also supplies entries for the various microprocessor tables that are required for protected virtual addressing mode. **Expanded Memory (EMS)** Some publications may refer to memory above the 1MB limit as "expanded memory." However, the term *Expanded Memory* is generally reserved to describe another special memory option. In 1985, three companies (Lotus, Intel, and Microsoft) joined together to define a method of expanding the 8088's memory usage capabilities by switching banks of memory from outside the DOS memory area into the usable address ranges of the 8088. This method became known as the LIM EMS (for *Lotus, Intel, and Microsoft Expanded Memory Specification*) standard.

This idea of bank switching was not exactly new; it had been used with older computer systems before the advent of the IBM line. The LIM EMS standard simply defined how this technique should be applied to IBM PCs and their compatibles. The original standard defined specifications for both hardware and software elements of the EMS system. Figure 3.25 illustrates the basic principal behind the EMS standard.

FIGURE 3.25 *Expanded memory (EMS) operations.*

The specification allows four 16KB areas (*pages*) of memory between C0000h and EFFFFh to be used as windows into pre-defined RAM locations above the 1MB address limit. Originally, these RAM addresses were located on special EMS RAM cards that plugged into one of the system board's expansion slot connectors. Newer system boards, based on the 80486 and Pentium microprocessors, can use their advanced virtual memory paging capabilities to handle the EMS function directly on the

board.

Figure 3.25 depicts hex locations D0000h through DFFFFh being used as windows through which the Expanded Memory addresses are translated. In reality, the four 16KB windows can be selected from anywhere within the LIM EMS-defined address range, and can be relocated to anywhere within the 32MB physical address range.

The EMS software specifications consist of predetermined programs called *Expanded Memory Manager* (EMM) drivers that work with application software to control the bank-switching operations. These drivers contain special function calls that application programs can use to manipulate the expanded memory. Note, however, that the application software must be written to take advantage of the EMS function calls. EMS versions before 4.0 made provision for the expanded memory to be used only as data storage areas. Programs could not actually be executed in these areas. Versions 4.0, and later, support much larger bank-switching operations, as well as program execution and multitasking. **Virtual Memory** The term *virtual memory* is used to describe memory that isn't what it appears to be. Virtual Memory is actually disk drive space that is manipulated to seem like RAM. Software creates virtual memory by swapping files between RAM and the disk drive, as illustrated in Figure 3.26. Because the swapping represents a major transfer of information that involves the hard-disk drive, an overall reduction in speed is encountered with virtual memory operations.

FIGURE 3.26 *Virtual memory operations.*

CONFIG.SYS

OBJECTIVE: During installation, DOS versions from 5.0 forward create a system file called CONFIG.SYS. This particular filename is reserved by DOS for use with a special file that contains setup (configuration) instructions for the system. When DOS is loaded into the system, a portion of the boot-up program automatically searches in the default drive for a file named CONFIG.SYS. The commands in this file configure the DOS program for use with options devices and applications programs in the system.

The CONFIG.SYS program is responsible for: (1) setting up any memory managers being used, (2) configuring the DOS program for use with options devices and application programs, (3) loading up *device-driver software*, and (4) installing memory-resident programs. These activities are illustrated by the following sample CONFIG.SYS file:

```
1 Device=C:\DOS\HIMEM.SYS
Device=C:\DOS\EMM386.EXE 1024 RAM

2 FILES=30
BUFFERS=15
STACKS=64,500

3 DEVICE=C:\DOS\SMARTDRV.SYS 1024
DOS=HIGH, UMB
DEVICEHIGH=C:\MOUSE\MOUSE.SYS
DEVICEHIGH=C:\DOS\RAMDRIVE.SYS 4096/a
```

Memory Managers

In the first section, the system's memory-manager programs are loaded. In this case, the HIMEM.SYS command loads the DOS *extended memory driver*. This driver manages the use of *Extended Memory* (XMS) installed in the system. This memory manager should normally be listed in the CONFIG.SYS file before any other memory managers or device drivers.

The EMM386.EXE program provides the system's microprocessor with access to the upper memory area. Operating together with the HIMEM.SYS program, this enables the system to conserve conventional memory by moving device drivers and memory resident programs into the UMA. This concept is described in Figure 3.27.

FIGURE 3.27 Loading memory managers.

HIMEM.SYS also creates a 64KB area of memory just above the 1MB address space called the *High Memory Area* (HMA). With this, the DOS=HIGH statement is used to shift portions of DOS from conventional memory into the HMA.

Similarly, the EMM386.EXE command loads the DOS *Expanded Memory* simulator driver. A file called LIM EMS 4.0 is another commonly used expanded memory manager that you might encounter in a CONFIG.SYS file set up for expanded memory operations.

Files, Buffers, and Stacks

In the second section of the file are the commands that define DOS for operation with optional devices and applications. The *FILES* command causes the DOS program to establish the number of files that DOS can handle at any one time at 30. This just happens to be the minimum number required to load Windows for operation. The *BUFFERS* command sets aside 15 blocks of RAM memory space for storing data being transferred to and from disks. Similarly, the *STACKS* command establishes the number and length of some special RAM memory storage operations at 64 memory stacks, with each being 500 bytes long.

Device Drivers

Device drivers are loaded in the third part of the file. Device drivers are programs that tell DOS how to control specific devices. DEVICEHIGH=C:\MOUSE\MOUSE.SYS is a command that loads a third-party device driver supporting the particular mouse being used with the system. The order in which device drivers appear in the CONFIG.SYS file is important. The recommended order for listing device drivers is (1) HIMEM.SYS, (2) the expanded memory manager if installed, (3) the EMM386.EXE command, and then (4) any other device drivers being used.

The SMARTDRV.SYS driver establishes a disk cache in an area of extended memory as a storage space for information read from the hard-disk drive. A *cache* is a special area of memory reserved to hold data and instructions recently accessed from another location. A *disk cache* holds information recently accessed from the hard-disk drive. Information stored in RAM can be accessed much more quickly than that stored on the hard drive. When a program or DOS operation requests more data, the SMARTDRV program redirects the request to check in the cache memory area to see whether the requested data is there. If SMARTDRV finds the information in the cache, it operates on it from

there. If the requested information is not in the cache, the system accesses the hard drive for it.

Using this technique, the overall operating speed of the system is improved. When the system is shut down, SMARTDRV copies the most current information onto the hard drive. Therefore, no data is lost due to it being stored in RAM. The idea behind SMARTDRV operations is described by Figure 3.28.

FIGURE 3.28 *How SMARTDRV works.*

The 1024 modifier establishes a memory cache size of 1MB (1024k of memory) in extended memory. This is a typical cache size for SMARTDRV; 2MB (2048k), however, is probably the most efficient size for the cache. This is because the larger the cache size, the greater the chance that the requested information is in the cache. So there is no need to go to the hard drive for the information. If the command is modified further by an /a extension, the cache is established under an expanded memory operation rather than extended memory. Extended memory is the default for SMARTDRV operations.

The *RAMDRIVE.SYS* driver simulates the organization of a hard-disk drive in RAM memory. This type of drive is called a *Virtual Disk*. In this case, the `DEVICEHIGH=` command loads the *RAMDRV* into the upper-memory area rather than the base-memory area, where a simple `DEVICE=` command would run it. Likewise, the `DOS=HIGH,UMB` command shifts the operation of DOS into the high-memory area, and gives the application access to the upper-memory area.

The operation of both the *SMARTDRV.SYS* and *RAMDRIVE.SYS* device drivers is governed by the *HIMEM.SYS* memory manager. This is normal only because both programs involve the use of memory beyond the 1MB conventional-memory level. Likewise, the `DEVICEHIGH=` and `DOS=HIGH` commands that move programs into the upper memory area perform under the guidance of the *HIMEM.SYS* manager.

The fourth portion of the file sets the system up to use special keyboard shortcuts available in the *DOSKEY* program. *DOSKEY* is a type of program referred to as a *memory-resident* program. Memory-resident programs are programs that run in the background of other programs.

The DOS `INSTALL` command is placed in the *CONFIG.SYS* file to load memory-resident files into memory when DOS starts up. These files remain in memory as long as the system is on. A common install command is: `INSTALL=C:\DOS\SHARE.EXE`. The *SHARE* program provides the capability to share files in a networked, or multitasking, environment.

Other common *CONFIG.SYS* commands include:

- `BREAK`
- `COUNTRY`
- `DRIVPARM`
- `LASTDRIVE`
- `NUMLOCK`

- REM
- SET
- SHELL
- INCLUDE
- MENUCOLOR
- MENUDEFAULT
- SUBMENU

The definitions and usage of these commands are covered in detail in the MS-DOS User's Guide. The DOS installable device drivers are also defined in that publication.

Altering CONFIG.SYS Steps

The operation of the CONFIG.SYS file can be altered or bypassed by pressing selected keyboard keys during the bootup process. Holding the SHIFT key, or pressing the F5 key while the MS-DOS message "Starting DOS..." is on the screen, causes the bootup process to skip all the commands in the CONFIG.SYS file. This action also bypasses all the steps of the AUTOEXEC.BAT file (discussed in the next section).

When this option is used, the system boots up with a complete set of default settings. No installable device drivers are installed, the current directory is set to C:\DOS, and you may receive a "Bad or missing command interpreter" message. If this message is received, the system asks you to manually enter the path to the COMMAND.COM file.

OBJECTIVE: Similarly, pressing the F8 function key while the DOS message is on the screen causes the system to stop between each CONFIG.SYS command, and ask for verification before proceeding. This can be very helpful in troubleshooting configuration and bootup problems. This action also causes the system to ask the user whether the AUTOEXEC.BAT file should be run or skipped. Placing a question mark after a CONFIG.SYS command (before the = sign) causes the system to automatically seek verification whenever the system is booted up.

DOS comes with several other standard device driver programs. These include:

- KEYBOARD.SYS
- DISPLAY.SYS
- ANSI.SYS
- DRIVER.SYS
- PRINTER.SYS

KEYBOARD.SYS is the DOS default keyboard definition file. The DISPLAY.SYS driver supports code-page switching for the monitor type in use by the system. A *code page* is the set of 256 characters that DOS can handle at one time, when displaying, printing, and manipulating text. ANSI.SYS supports ANSI escape-code sequences used to modify the function of the system's display and keyboard. This file is also required to display colors on the monitor in DOS. DRIVER.SYS creates the logical drive assignments for the system (such as A: and C:). Finally, the PRINTER.SYS driver supports code-page switching for parallel ports. All these drivers are normally found in the DOS directory.

POWER.EXE

A special, power-saving program called POWER.EXE is designed for use in notebook computers. When it is loaded in the last line of the CONFIG.SYS file, and the system hardware meets the Advanced Power Management specification, the power savings can be as high as 25%. This is an important savings when you are discussing the operation of a battery, and its length of operation before it needs to be recharged. The POWER.EXE file must be available in the C:\DOS directory.

AUTOEXEC.BAT

After completing the CONFIG.SYS operation, DOS searches for the presence of a file called the AUTOEXEC.BAT file. This file contains a *batch* of DOS commands that are automatically carried out when DOS is loaded into the system. This file can also be re-executed from the DOS prompt if you simply type the command "AUTOEXEC." This is not true of the CONFIG.SYS file however. The system must be restarted to perform the commands in this file.

Refer to the following sample AUTOEXEC.BAT file:

```
DATE
TIME
PROMPT=$P$G
SET TEMP=C:\TEMP
PATH=C:\;C:\DOS;C:\MOUSE
DOSKEYSMARTDRV.EXE 2048 1024
CD\
DIR
```

The first two commands cause DOS to prompt you for the date and time (because DOS does not automatically do this when an AUTOEXEC.BAT file is present). The *PROMPT=\$P\$G* command causes the active drive and directory path to be displayed on the command line. The *SET TEMP=* line sets up an area for holding data temporarily in a directory named TEMP.

The *PATH* command creates a specific set of paths that DOS is to use to search for program files. In this case, DOS searches for executable files first in the root directory, followed by the DOS directory, and finally through the MOUSE directory. This statement effectively lets the Mouse program be executed from anywhere in the system. Upon receiving the command, the operating system looks through all of the directories in the path until it finds the specified filename.

The *syntax* (punctuation and organization) of the PATH command is very important. Each entry must be complete from the root directory and must be separated from the previous entry by a semicolon. There should be no spaces in the PATH command.

The *DOSKEY* command loads the DOSKEY program into memory. Following this, the SMARTDRV.EXE 2048 1024 command configures the system for a 2MB disk cache in DOS and a 1MB cache for Windows. After the cache has been established, the CD\ command causes the DOS default directory to change to the root directory. The last line causes a DOS DIR command to be performed automatically at the end of the operation.

The execution of the AUTOEXEC.BAT file can be interrupted by pressing the *Pause* key on the keyboard. The program can be restarted by pressing any key. With DOS version 6.2, the F8 interactive bypass procedure, described for use with the CONFIG.SYS file, was extended to include the AUTOEXEC.BAT file.

You can use the DOS batch file commands to construct elaborate startup procedures. Other programs designed to test ports and peripherals can be constructed using these commands. These test files can be named using the DOS filename conventions. They must be stored with a .BAT extension to be executable from the DOS prompt, but the extension does not need to be entered in order for the program to run.

Neither of these two special files are required for normal operation of the computer with DOS. However, they can prove to be very useful in tailoring the operation of the system to your personal use, or to the requirements of different software applications packages. To determine whether either of these files already exist on your DOS disk, simply type the DIR command at the DOS prompt (with the disk in the default drive).

Refer to the MS-DOS® User's Guide manual for more information about the creation and use of the CONFIG.SYS and AUTOEXEC.BAT files. Other DOS utilities for disk management are covered in the disk drive and preventive maintenance chapters (Chapters 8 & 13).

DOS Editor

Later versions of DOS contain a small text editor program that enables users to easily modify text files. This package is started by typing **EDIT** and the filename at the DOS prompt. The DOS editor's working screen appears. The editor is particularly useful in modifying the CONFIG.SYS and AUTOEXEC.BAT files. The DOS editor is an editor for unformatted text files. It does not introduce formatting codes, such as underlining and italics, into the text in the manner that more powerful word processors do. This is an important consideration when dealing with DOS utility files. Formatting codes can introduce errors in these files, because DOS is not able to recognize them.

DOS Versions

Although DOS has remained compatible with its original design, this does not mean that it has not changed significantly since its original version.

In July of 1981, Microsoft purchased the rights to a personal computer DOS from the Seattle Computer Products and promptly named it MS-DOS. A month later IBM began shipping a private labeled version of the Microsoft package that it named PC-DOS 1.0.

In May of 1982, Microsoft released MS-DOS version 1.1 to IBM for its units, and released its own brand name DOS product, MS-DOS 1.25, for PC-compatible computers. This version added support

for 360KB double-sided floppy-disk drives.

In March of 1983, MS-DOS 2.0 was announced. It added support for 10MB hard drives, a directory tree structure, and 360KB floppy drive support to the operating system. A minor revision, titled MS-DOS 2.11, added foreign language and date features to the operating system in March of 1984.

Version 3.0 of MS-DOS was released in August of 1984. It was released along with IBM's AT model and added support for 1.2MB floppy-disk drives and larger hard-disk drives. In November of the same year, version 3.1 added support for Microsoft networks. By January of 1986, version 3.2 had entered the market and brought support for 3-1/2 inch, 720KB floppy-disk drives to the operating system.

In August of 1987, version 3.3 delivered 1.44MB floppy-disk drive and multiple 32MB partitions for hard drives.

Version 4.0 of MS-DOS was released in June of 1988. This new version introduced a graphical shell for DOS, a mouse interface, and expanded memory drivers. By November, version 4.01 was being shipped to clean up problems with the 4.0 version.

The next version of MS-DOS didn't appear until June of 1991. Version 5.0 brought a full screen editor, task-swapping capabilities, *Undelete* and *Unformat* commands, and QBASIC to the operating system. In addition, it included support for upper memory blocks, larger hard disk partition sizes (up to 2GB), support for 2.88MB floppies, and the capability to load DOS into the HMA, as well as to load device drivers into the UMBs.

Microsoft began shipping the 6.0 version of MS-DOS in March of 1993. This new version included a DoubleSpace disk compression utility that enabled users to double the storage capacity of their hard-disk drives. Over 1 million copies of this version sold within the first month and a half. An enhanced version, MS-DOS 6.2, was released in November of the same year. By February of 1994, legal problems over the compression utility caused Microsoft to release version 6.21 with the utility removed. However, by June, version 6.22 appeared with the compression software back in the operating system under the name DriveSpace.

In April of 1994, IBM released a new version of PC-DOS. This was version 6.3. In January of 1995, they followed with PC-DOS 7, which included data compression for hard disk doubling. This marked the last release of a major command line-based DOS operating system. Table 3.2 summarizes the development of DOS products.

TABLE 3.2 DOS DEVELOPMENT TIMELINE

Year	Version	Features
1981	V1.0	First operating system for IBM-PC.
	V1.25	Double-sided disk support and bug fixes added, widely distributed by OEM's.
1983	V2.0	Hierarchical file support and hard disk support.
	V2.01	International support added.
	V2.11	V2.01 with bug fixes.
1984	V3.0	Introduced with AT model. Support for 1.2MB floppy, and larger hard disk sizes.

- V3.1 Support for Microsoft Networks added.
- 1986 V3.2 Enhanced support for new media types added.
- 1987 V3.3 Support for 1.44MB floppy, support for 4 serial ports, hard disk partitions > 32MB, improved national language support.
- V4.0 Dosshell, support for TSR's, expanded memory drivers, install program (select), mem command.
- V4.01 V4.0 with bug fixes.
- 1992 V5.0 Support for upper memory blocks, larger partition sizes (< 2GB), loading device drivers in UMB, improved dosshell and on-line help, support for 2.88MB drives, Qbasic, improved system editor (edit).
- 1994 V6.0 DoubleSpace Disk compression introduced.
- V6.22 DriveSpace Disk compression, replaces DoubleSpace.

Installing DOS

OBJECTIVE: In the earliest versions of DOS, the operating system was contained on two disks, the System Disk and the Supplemental Disk. Because the early PCs operated from floppy-disk drives, they booted directly to the system disk. The most-used DOS utility functions were loaded into RAM. For advanced DOS functions, the Supplemental disk was inserted in the drive when requested by the system.

When the PCs moved to hard drive operations, the main DOS files were placed on the hard disk as part of its formatting process. The other DOS files were typically copied into a C:\DOS directory when the unit was set up.

The installation of newer DOS versions (after 5.0) is a relatively easy process. It is controlled by the file *SETUP.EXE*. The only response required by this program is to tell it whether DOS should be installed on the hard-disk drive. The installation process is so automated in newer versions that it runs the Setup.exe program automatically on new systems. This is accomplished simply by starting the system with the first DOS disk (1 of 3) in the floppy drive.

The setup program's installation screen defaults to the C:\DOS sub-directory to install the DOS files. The files on the installation floppies are compressed, so they cannot simply be copied onto the hard drive. As the DOS files are installed, the setup program automatically determines the amount of memory, the number of drives, and other configuration information about your computer.

With versions of MS-DOS from 5.0 forward, the setup program also creates the files *AUTOEXEC.BAT* and *CONFIG.SYS*. Once setup has determined the configuration, the findings are presented for the user to check. If the setup program incorrectly determines the configuration, it can be changed during the setup.

DOS Shell

Versions of DOS from 4.0 forward offer an option to install the *DOS shell*. The DOS shell is a multiple window user-interface that simplifies performing DOS operations. The DOS shell interface is depicted in Figure 3.29.

FIGURE 3.29 *The DOS shell.*

As illustrated in the figure, the DOS shell divides the screen into four basic areas:

- Title bar
- Directory tree window
- File list window
- Program list window

The drive icons under the title bar are used to select the disk drive in which to carry out any DOS functions. The drive icon is selected by highlighting it with the cursor and then pressing the Enter key. When the drive is selected, its directory structure appears in the directory tree window.

The directory tree window shows the organization of the selected disk's directory structure. As with drive selection, a particular directory or sub-directory can be selected using the cursor. When this is done, the contents of the directory appear in the file list window.

The program list window shows programs that can be run directly from the shell such as QBASIC and DOS Editor.

Beginning with version 4.0 MS-DOS enabled *Task Swapping*. By enabling the Task Swapper utility in the Title Bar's Options menu, multiple programs could be run simultaneously. The execution of the programs was not simultaneous, but there was no need to exit from one program to start another. All the active programs appear in the Active Task window. To switch between them is a simple operation of getting to the shell and clicking on the desired program in the window.

The shell makes it easy to locate and start files on the disk. Any program can be located by scrolling through the different windows. When the desired file is located, it can be run by simply highlighting it with the cursor and pressing the Enter key, or by clicking a mouse button. There are no cryptic commands to remember and no long paths to type.

Summary

This chapter has looked at operating systems in depth. It has concentrated on the ROM BIOS and disk operating system. At this point, you should be able to discuss the duties of the BIOS and disk operating system. You should also be able to describe the events involved in booting up a computer to the operating system. In addition, you should be able to describe the general use of the disk operating system after it has been loaded. The next chapter examines the Microsoft Windows operating environment and the Windows 95 Operating System. These systems differ from the disk operating system that we have covered in this chapter in that they were designed to be graphical in nature and use.

Lab Exercise

A hands-on lab procedure corresponds to the theory materials presented in this chapter. Refer to the lab manual and perform Procedure 2, "Introduction to DOS."

Review Questions

1. What function does the POWER.EXE file perform and where is it commonly found?
2. Under what conditions is it normally necessary to run the *CMOS Configuration Setup* program?
3. What type of operating system breaks the tasks associated with a process into its various threads for execution?
4. Which DOS command prepares a disk to function as a self-booting disk?
5. Describe the DOS prompt.
6. In terms of managing processes, what type of operating system is DOS?
7. For which file is the operating system loader looking during the bootup process?
8. What advantage does the DOS shell offer to the system?
9. What is the purpose of placing a SMARTDRV.EXE command in the CONFIG.SYS file?
10. What is the first event that occurs when the system is turned on or reset?
11. Which memory manager should always be listed before any other memory managers or device drivers?
12. If the system locks up while the message "Starting DOS" is on the screen, what two actions should be performed?
13. Pressing the F8 key while the "Starting DOS" message is on the screen has what effect on the system?
14. Where do you find the system's memory managers listed?
15. From the system startup point of view, how do a cold and a warm boot differ?
16. What does the "*" character stand for when used in a DOS filename?
17. How does the XCOPY command differ from the copy or diskcopy command?
18. List the three files that must be located in the root directory in order for you to successfully boot MS-DOS.
19. How are multi-tasking and multi-user systems different?

20. Write a DOS command that can be inserted in the AUTOEXEC.BAT file to cause the active path and directory to be shown on the DOS command line.
21. Which filename extensions enable programs to be started directly from the DOS prompt?
22. What does HIMEM.SYS do?
23. Write a DOS command that can be used to make a duplicate of another disk.
24. What condition is indicated by the presence of an A:\> prompt on the monitor screen?
25. What does the BUFFERS= command in the CONFIG.SYS file do?

Review Answers

1. POWER.EXE is a special, power-saving program designed for use in notebook computers. When it is loaded in the last line of the CONFIG.SYS file, it can provide savings as high as 25%. For more information, see the section "POWER.EXE."
2. When setting up a computer for the first time, or when adding new options to the system. For more information, see the section "CMOS Setup Routines."
3. Multiple-process operating systems. For more information, see the section "Operating Systems."
4. Format /s. For more information, see the section "Drives and Disks."
5. The DOS prompt is a screen symbol that marks a point where DOS commands can be entered. The prompt includes the designation of the active disk drive, and may contain directory and subdirectory path information (for example, C:\dos\mouse). For more information, see the section "DOS Command Line."
6. DOS is a single-process operating system that operates in *interactive mode*. For more information, see the section "Operating Systems."
7. The BIOS begins looking through the system for a Master Boot Record. For more information, see the section "Bootup."
8. The DOS shell is a multiple-window user interface that simplifies performing DOS operations. For more information, see the section "DOS Shell."
9. To establish a disk cache in the extended memory area for data read from the hard drive. For more information, see the section "Device Drivers."
10. The system is reset to its starting condition. For more information, see the section "Post Tests and Initialization."
11. The HIMEM.SYS command. For more information, see the section "Memory Managers."

- 12.** Press the F5 key to skip the CONFIG.SYS and AUTOEXEC.BAT portions of the bootup to see whether information in these files is causing conflicts. If the system boots up without these files, reboot and use the F8 key to single-step through the two files until the conflict is located. For more information, see the section "Altering CONFIG.SYS Steps."
- 13.** It causes the system to move through the CONFIG.SYS and AUTOEXEC.BAT files one command at a time. For more information, see the section "Altering CONFIG.SYS Steps."
- 14.** In the system's CONFIG.SYS file. For more information, see the section "CONFIG.SYS."
- 15.** In a cold boot situation, the system executes the entire BIOS startup routine. In a warm boot, only some of the POST tests are performed on the system. This makes a warm boot much quicker. For more information, see the section "Initial POST Checks."
- 16.** The "*" symbol is used as wild card character that can be substituted for an unknown character or character string in DOS commands. For more information, see the section "DOS Shortcuts."
- 17.** The XCOPY command copies all the files in a directory, along with any sub-directories and their files; the file copy command copies a specified file from one place (disk or directory) to another. For more information, see the section "Files and Filenames."
- 18.** MSDOS.SYS, IO.SYS, and COMMAND.COM. For more information, see the section "Bootup."
- 19.** The multi-user system switches between different users at multiple locations; multi-tasking systems switch between different applications at a single location. For more information, see the section "Operating Systems."
- 20.** PROMPT=\$P\$G. For more information, see the section "Directories."
- 21.** .EXE, .COM, and .BAT For more information, see the section "DOS Command Line."
- 22.** It governs the use of extended memory. For more information, see the section "Extended Memory."
- 23.** DISKCOPY A: For more information, see the section "Drives and Disks."
- 24.** DOS is operational and the currently active drive is the A: floppy drive. For more information, see the section "Bootup."
- 25.** It sets aside blocks of RAM memory space for storing data being transferred to and from disks. For more information, see the section "Files, Buffers, and Stacks."