Embedded DOS™-ROM

Full-Featured ROMmable Disk Operating System

User's Manual
with Command Reference

IMPORTANT NOTICES

General Software, the GS logo, Embedded DOS, Embedded DOS 6-XL, Embedded BIOS, Embedded DOS-ROM, Embedded LAN, CodeProbe, The Snooper, and EtherProbe are trademarks of General Software, Inc. Other marks are the property of their respective holders.

<u>Important Licensing Information</u>

IMPORTANT -- READ CAREFULLY BEFORE OPENING THIS PACKAGE. BY OPENING THIS SEALED PACKAGE, INSTALLING OR OTHERWISE USING THE SOFTWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENSE AGREEMENT, DO <u>NOT</u> OPEN THE PACKAGE OR USE THIS SOFTWARE AND PROMPTLY RETURN THE UNOPENED SOFTWARE AND ANY ACCOMPANYING MATERIALS TO GENERAL SOFTWARE FOR A REFUND.

This License Agreement is a legal agreement between you and General Software, Inc. ("General") for the General software product identified above, which includes the computer software and any associated media and printed materials or electronic documentation (collectively, the "Software ").

GRANT OF LICENSE. In consideration of the license fees paid to General, and subject to the terms and conditions set forth herein, General hereby grants you a nonexclusive, nontransferable license (the "Adaptation License") to use the Software and the accompanying documentation solely for your internal use to evaluate and adapt the Software for use in products manufactured by you. Other than for the purpose of evaluating and adapting the Software for use with your products, you may not reproduce or install copies of the Software. Prior to distributing the Software in such products or installing the Software in the products for distribution, you must enter into a separate license agreement with General for such installation and distribution and pay the applicable royalties.

SOURCE CODE. Subject to the terms and conditions set forth herein, General grants you a nonexclusive, nontransferable license to use portions of the source code for the Software for your internal use only, for the sole purpose of adapting the Software to your products. You will keep the source code under restricted access in a safe and secure place and shall take reasonable precautions and actions to protect the source code from unauthorized use or disclosure. Such precautions and actions shall be at least as stringent as those you take to protect your own source code or other confidential information. You will not disclose the source code to any person or entity without the prior written consent of General, except that you may disclose the source code to your employees on a need-to-know basis, provided such employees are contractually obligated to maintain the confidentiality of the source code. The foregoing obligations will survive any termination of this Adaptation License.

PROPRIETARY RIGHTS. This Software is licensed to you, not sold, and is protected by copyright laws and international treaty provisions. All title, copyrights and other proprietary rights in and to the Software, the accompanying printed materials, and any copies thereof are owned by General. No title to the Software or any copy thereof or any associated proprietary rights are transferred to you by this license.

DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS. You may not copy, use or distribute all or any portion of the Software or the accompanying printed material except as expressly permitted herein. You may not reverse engineer, decompile, or disassemble the Software, nor may you modify the Software except as necessary to adapt the Software for use with your products. You may not sell, assign, rent, lease or lend the Software.

TERMINATION. Without prejudice to any other rights or remedies, General may terminate this Adaptation License if you breach any of the terms and conditions of this License Agreement. In such event, you must destroy and/or erase all copies of the Software and all of its component parts.

SUPPORT. For a period of thirty (30) days from the date you acquired this Software, General will provide, subject to availability, up to 5 hours of telephone support to assist you with questions regarding the installation and adaptation of the Software. If additional support is desired, a variety of support plans are available from General by entering into a separate support agreement. You acknowledge that except as may be expressly set forth in a separate written support agreement entered into by the parties, **GENERAL MAKES NO WARRANTIES OR REPRESENTATIONS OF ANY KIND WITH RESPECT TO THE RESULTS OR AVAILABILITY OF SUCH SUPPORT.**

DISCLAIMER OF WARRANTY. You acknowledge that the Software and any accompanying materials are being furnished solely for evaluation and adaptation purposes. Therefore, **THE SOFTWARE AND THE ACCOMPANYING MATERIALS ARE BEING PROVIDED "AS IS" AND GENERAL MAKES NO WARRANTIES OF ANY KIND WITH REGARD TO THE SOFTWARE, EITHER EXPRESS OR IMPLIED, AND THERE IS EXPRESSLY EXCLUDED ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE**.

INDEMNIFICATION. Without limiting the generality of any of the foregoing, you acknowledge that the Software is not designed or intended to be used in connection with any product, the operation, use or malfunction of which could result in death or serious bodily harm, and you agree to take full responsibility for the use of and results achieved from the incorporation of the Software into your products. You agree to defend, indemnify and hold harmless General, and its officers, directors, employees and agents, from and against any and all claims, actions, proceedings, liabilities, costs and expenses (including without limitation reasonable attorneys' fees) arising out of (a) any representation, warranty, act or omission made by you in connection with your products or the Software, or (b) the use of or inability to use your products or the Software incorporated therein or distributed therewith.

LIMITATION OF LIABILITY. IN NO EVENT SHALL GENERAL BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THIS ADAPTATION LICENSE OR THE USE OF OR INABILITY TO USE THE SOFTWARE, EVEN IF GENERAL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL GENERAL'S LIABILITY UNDER THIS ADAPTATION LICENSE OR RELATED TO THE SOFTWARE (WHETHER IN TORT, CONTRACTS OR OTHERWISE) EXCEED THE AMOUNTS PAID TO GENERAL FOR THIS ADAPTATION LICENSE.

FEDERAL GOVERNMENT ACQUISITION. By accepting delivery of this Software, the Government hereby agrees that this Software qualifies as "commercial computer software" as that term is used in the acquisition regulation applicable hereto. To the maximum extent possible under federal law, the Government will be bound by the commercial terms and conditions contained in this license. The following additional statement applies only to procurements governed by DFARS Subpart 227.4 (1988): Restricted Rights - Use, duplication and disclosure by the Government is subject to restriction as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 (1988).

MISCELLANEOUS. This License Agreement constitutes the entire agreement between you and General regarding the Software. This License Agreement is governed by the laws of the State of Washington and the United States of America, without reference to its choice of law rules. The provisions of the 1980 U.N. Convention on Contracts for the International Sale of Goods shall not apply.

TABLE OF CONTENTS

Chapter 1

EMBEDDED DOS-ROM OPERATING SYSTEM

Thank you for choosing General Software's EMBEDDED DOS-ROM operating system for use in your embedded system. EMBEDDED DOS-ROM offers an unparalleled combination of configurability, performance, functionality, and price that enables embedded application developers to use in applications that traditionally would run proprietary operating systems.

About EMBEDDED DOS-ROM

Your new EMBEDDED DOS-ROM Adaptation Kit includes the following parts:

* EMBEDDED DOS-ROM Adaptation Guide (this manual)
* SYSTEM DISK (contains INSTALL.BAT)

* Binary, ready-to-run files for DISK or ROM booting
* Optional sources if you've purchased the Source code version

* EMBEDDED DOS-ROM OEM License Agreement
* Technical Support Request (TSR) form
* Product Registration form

This manual will explain how to install the EMBEDDED DOS-ROM Adaptation Kit software on your development system, and how to create a bootable adaptation of EMBEDDED DOS-ROM for your embedded application, by configuring the binary version and the source versions. If you did not purchase the Source code version, then some portions of this manual relating to source code do not apply to you.

The purchase of the Source code version of this Adaptation Kit is really not a matter of price. We've priced the source code to be inexpensive enough that it can be purchased without reservation if needed. The Source code gives the developer the ability to customize

EMBEDDED DOS-ROM in many ways that the binary version cannot provide. Perhaps most importantly, the Source code version has source-level options that can disable codepaths that would go unused in a closed embedded system, making it possible to save valuable ROM space and use it for other purposes, such as the application. If you've purchased the binary version to start with, and find that it would be useful to have the configurability of the Source code version, call us if you need an upgrade.

The *SYSTEM Disk* contains the INSTALL.BAT file that can be used to install the EMBEDDED DOS-ROM Adaptation Kit software. If a README.TXT file is present, the up-to-the-minute instructions in that file should be used instead of the instructions in this Chapter.

The enclosed *OEM License Agreement* enables you to license binary adaptations of the EMBEDDED DOS-ROM software. Contact General Software for help with this form and for current pricing for the volume you are interested in. We suggest that you begin the licensing process early so that you can take advantage of current rates.

The enclosed *Technical Support Request (TSR)* form should be used to submit technical support requests to General Software by FAX. You may duplicate this form as needed to make multiple requests.

General has implemented a proprietary and sophisticated Product Support Database that allows tracking the nature, content, progress, and history of each call made to or from our Product Support Group. This allows any Support Technician to access the needed information should your situation require more than one call and the original Support Technician is not currently available to take your call or respond to your fax.

Also enclosed is a *Product Registration* form that should be completed and mailed immediately. This information is needed for technical support and also makes you eligible to receive upgrades and access General Software's on-line services.

Choice of Command Interpreters

Embedded DOS-ROM can be configured to run a mini-version of COMMAND.COM directly from ROM, as a part of Embedded DOS-ROM image itself. This eliminates the need for a ROM disk to load COMMAND.COM from, and allows the OEM to perform basic program loading and file management functions at the command line. It also provides AUTOEXEC.BAT support, to allow embedded applications to be loaded from a startup batch file.

If Embedded DOS-ROM is not configured to run its Mini-COMMAND.COM, then it can load its larger, run-from-RAM COMMAND.COM from a drive of your choosing. While the Mini-COMMAND.COM is quite limited in syntax and support of command options so that ROM space is preserved, the full COMMAND.COM supports a full set of DOS intrinsic commands and supports other desktop features such as piping and redirection.

Automatic ROM Disk Detection

If your system has a ROM disk image installed in ROM as a linear array (not paged), then the Embedded DOS-ROM kernel can automatically scan for it while booting, and present that disk as drive A:. This eliminates the need for extra ROM extension software to occupy valuable upper memory blocks that can be used for DOS application memory instead. The ROM disk image must be located on a 256KB address boundary. The entire 64MB address space is scanned for

ROM disk images. To be recognized, ROM disk images must have a valid boot record with a BIOS Parameter Block (BPB) and a 55h/aah signature at the end of the boot record.

DOS Repositioning

Embedded DOS-ROM can automatically move itself into VGA memory, UMB memory, or extended memory, to free-up low memory for application software. This is done with DOS=VGA, DOS=UMB, and DOS=HIGH statements in CONFIG.SYS, respectively. If more than one of these CONFIG.SYS commands is specified, then as many components of Embedded DOS-ROM are moved as possible into the auxiliary areas.

The DOS=VGA command provides a full 64KB of low DOS memory if the underlying chipset supports it.

The DOS=UMB command provides UMBs between segment C800h and F000h at the chipset's discretion.

The DOS=HIGH command provides a 64KB area at the start of extended memory that can be used to move the DOS kernel high. This area is not made available to application software, although if this command is not used, and HIMEM.SYS is used, it can be made available to application software through the XMS software API.

Chapter 2

CONFIG.SYS STATEMENTS

Embedded DOS-ROM allows the OEM to configure it at boot time through the CONFIG.SYS configuration file. The specific commands supported by Embedded DOS-ROM are described here.

2.1 General Format of the CONFIG.SYS File

The standard format for CONFIG.SYS files is a sequence of ASCII text lines, where the total size of the CONFIG.SYS file does not exceed 64KB. Blank lines, and lines starting with ';', are ignored during CONFIG.SYS processing. Each line must end with a carriage-return, a line-feed, or both. Control characters encountered on lines cause the remainder of the line to be unprocessed by Embedded DOS-ROM, except for ^I (TAB).

CONFIG.SYS is actually processed several times by Embedded DOS-ROM during system initialization. Each pass collects the proper information to initialize the system at its level. For example, all device drivers are loaded first before UMB statements can be processed, or memory managers would not be able to initialize memory before that memory was linked into the system's memory arena.

2.2 CONFIG.SYS Commands

? Command

The "?" command is used to prefix any line in CONFIG.SYS that is optional. If "?" is placed in front of any CONFIG.SYS command, Embedded DOS-ROM will display the line and prompt the user to press "Y" if the line should be executed, and "N" if it should be bypassed. This allows optional execution of CONFIG.SYS commands at the user's discretion.

BREAK=ON|OFF   The BREAK command sets the default value of the DOS break flag. The DOS break flag is used by Embedded DOS-ROM to determine if it should poll the console for a ^C from the operator when an INT 21h function is invoked. If BREAK=ON, then the console is polled, and if a ^C is pending, the program is terminated. If BREAK=OFF, then the console is not polled. Polling adds an extra trip through the I/O system for each INT 21h call, so performance is best with it off.

BUFFERS=*nnnn*   The BUFFERS command sets the default value for the maximum number of 512-byte file system buffers to be used in the system. Note that Embedded DOS-ROM's file system cache may age the buffers and discard them, making them available for other purposes during normal system operation. This background activity is transparent to the user. A minimum value for this parameter is 3. A good maximum value is 40, beyond which no perceptible performance improvements are realized.

                                    WARNING: The space used to allocate buffers is derived from the system pool, and the size of the system pool is governed by the build parameter, DOSDATA_SIZE_IN_KB. If you have reduced this parameter's value substantially in the system build process, make sure that enough space has been accounted for in the pool for the buffers to be allocated by the user.

CHAIN=*filename*   The CHAIN command transfers control to another configuration file. Because CONFIG.SYS is read in several passes, the CHAIN statement takes effect after all of the statements in CONFIG.SYS have been processed. Subsequent CHAIN commands override earlier ones. A common use of this command is to optionally execute several additional commands as an optional group by using the "?" character in front of the CHAIN command.

COMMENT=*any text*   The COMMENT command is used to add structured comments to the CONFIG.SYS file.

COUNTRY=*nnn*   The COUNTRY command sets the country code so that the DOSCOUNTRYINFO service can return this value to application programs.

DEVICE=*devname*   The DEVICE command loads a specified file as a device driver (either character or block) in the system.

DEVICEHIGH=*devname*   The DEVICEHIGH command loads a specified file as a device driver in upper memory (such as a UMB), where it does not consume memory in the lower 640KB area.

DOS=VGA   The DOS=VGA command instructs Embedded DOS-ROM to request the underlying chipset hardware to enable shadow RAM at segment A000h so that it can be reclaimed for DOS or for application software.

DOS=UMB   The DOS=UMB command instructs Embedded DOS-ROM to request the underlying chipset hardware to enable shadow RAM in unused memory address space between segments C800h and E000h so that it can be reclaimed for DOS or for application software.

DOS=HIGH   The DOS=HIGH command instructs Embedded DOS-ROM to attempt to acquire the 64KB memory block at the 1MB address marker for its own data structures, freeing up low memory for application software. This statement cannot be used if HIMEM.SYS is to be loaded, as the two methods manage high memory differently.

ECHO=*any text*   The ECHO command is used to display a message to the console during CONFIG.SYS processing.

FCBS=*nnnn*   The FCBS command remains for compatibility with MS-DOS CONFIG.SYS files; it has no effect on operation of the system. Embedded DOS-ROM dynamically allocates system structures in response to application requests.

FILES=*nnnn*   The FILES command remains for compatibility with MS-DOS CONFIG.SYS files; it has no effect on operation of the system. Embedded DOS-ROM dynamically allocates system structures in response to application requests.

INSTALL=*progname*   The INSTALL command loads a specified file as a program as though it were executed in AUTOEXEC.BAT. This allows programs such as TSRs to be run without requiring COMMAND.COM or an AUTOEXEC.BAT file. SMARTDRV.EXE is commonly loaded with this command.

INSTALLHIGH=*progname*   The INSTALLHIGH command loads a specified file as a program in upper memory (such as a UMB), where it does not consume memory in the lower 640KB area.

LASTDRIVE=*d:*   The LASTDRIVE command specifies the last drive letter to be used by DOS for its own purposes. Drive letters after this one are typically managed by network redirectors such as Novel NetWare. The default last drive is E:.

REM=*any text*   The REM command is used to add structured comments to the CONFIG.SYS file.

STACKS=*nnnn*   The STACKS command sets the number of INT 21h stacks allocated by Embedded DOS-ROM during system initialization. By default, Embedded DOS-ROM allocates 5 stacks, each taking 2K bytes of memory.

       INT 21h stacks are used as a safe context to run INT 21h activities inside the kernel. Note that even the act of launching a program with INT 21h function 4bh uses a stack, so it may become

necessary to increase STACKS if many TSRs are to be loaded into the system.

If insufficient stacks remain in the system for an INT 21h service to be executed, the system might appear to stall.

SHELL=*progname*   The SHELL command specifies the initial program to load when all system initialization has completed. By default, the shell program name is d:\COMMAND.COM. If you wish to have your own program loaded in place of COMMAND.COM, use the SHELL command and specify the program name, including the full path and extension (COM or EXE).

UMB=*xxxx*   The UMB command specifies the hexadecimal segment address of an upper memory block (UMB) made available by a memory manager. Using the UMB command registers the memory block with the Embedded DOS-ROM arena manager, enabling programs to acquire the memory through standard DOS services, and C Library services that call the DOS services; i.e., malloc. This command should only be used with extended memory managers, and should not be used in conjunction with DOS=UMB statements.

VERIFY=ON|OFF   The VERIFY command sets the initial state of the DOS VERIFY flag, used by the FAT file system cache to <u>enable the write-behind algorithm</u> (delayed writes) if <u>VERIFY=OFF</u>. If VERIFY is set to ON, then the file system operates in a hardened mode (at the expense of reduced performance), making the file system more resilient to application crashes, since dirty file system cache blocks are flushed to disk before each write completes.

If VERIFY is set to OFF, then the automatic cache flushing on each I/O is disabled, causing the write-behind algorithm to write the data according to the CACHEFLUSH and CACHETTL parameter values.

VERSION=*n*   The VERSION command sets the version number that DOS reports to applications.

2.3 Device Driver Reference

Embedded DOS-ROM ships with installable device drivers that can be loaded with the DEVICE= and/or DEVICEHIGH= statements in CONFIG.SYS. This section documents their use.

ANSI.SYS Driver

Function: Emulates ANSI Escape Sequence Video Terminal.

Type: Character device driver

Syntax: DEVICE=ANSI.SYS

Parameters:

none.

Description:

The ANSI.SYS driver is loaded as a substitute for the default CON driver that is built into the Embedded DOS-ROM system. ANSI.SYS watches output requests and intercepts ANSI escape sequences emitted by programs that require an ANSI terminal in order to clear the screen, position the cursor, and perform other screen-oriented tasks. When these escape sequences are detected, they are translated by ANSI.SYS to the appropriate cursor positioning video BIOS requests.

EMM386.EXE Driver

Function:　　　Extended Memory Manager.

Type:　　　　　Character device driver

Syntax:　　　　DEVICE=EMM386.EXE [/V] [/X=aaaa-bbbb] [/I=aaaa-bbbb] [/N=aaaa-bbbb]

Parameters:

aaaa　Specifies a 16-bit segment address that is the lower bound of a range of addresses.

bbbb　Specifies a 16-bit segment address that is the upper bound of a range of addresses.

Description:

The EMM386.EXE driver is used to provide the system with backfilled memory in otherwise unused areas between 640KB and 1MB. EMM386.EXE does this as a virtual control program by switching the processor into V86 mode, so that DOS and applications run at ring 3 (least privileged mode of the processor), while EMM386 operates at ring 0 (most privileged mode of the processor.) The mechanism that maps the memory into the UMBs is the CPU's paging hardware, managed by EMM386.EXE. Note that because this mapping occurs within the CPU itself, it will not provide memory which can be available to DMA-based hardware; only to accesses made directly within the CPU itself.

This EMM386.EXE driver does not provide all the programming services that Microsoft's EMM386.EXE driver does. Instead, it only provides backfilling for UMB support, so that drivers and programs can be loaded high.

If the /V option is selected, then EMM386.EXE will provide verbose displays during initialization, such as the amount and location of XMS memory reserved for Upper Memory Blocks (UMBs) and 386 page tables.

If the /X option is selected, then EMM386.EXE will exclude the specified range from mapping UMBs. This is useful if a device or ROM resides at a segment address that would normally be mapped by EMM386.EXE (for example, D000h-DFFFh.)

If the /I option is selected, then EMM386.EXE will include the specified range in its UMB map. This will cause EMM386.EXE to map pages of extended memory into this segment address range, causing memory to appear there for use by DOS and application software.

If the /N option is selected, then EMM386.EXE will treat the specified range as a mappable range, but will not automatically give the memory to the DOS arena for use by application software. Instead, it will be up to the application software to manually create pointers to the area and use it without DOS's support.

Because mapping is accomplished with CPU paging hardware, and because the X86 family has 4KB pages, the segments are rounded to the nearest 4KB page boundary.

Example:

The following command loads EMM386.EXE and maps the 128KB at segment D000h, while excluding the region at A000-BFFF:

```
DEVICE=EMM386.EXE  /I=d000-dfff  /X=a000-bfff
```

HIMEM.SYS Driver

Function: XMS 2.0 Memory Manager.

Type: Character device driver

Syntax: DEVICE=HIMEM.SYS

Parameters:

none.

Description:

The HIMEM.SYS driver is used to provide the Embedded DOS-ROM XCOPY, and COMMAND.COM programs with uniform access to extended memory through the industry-standard XMS interface.

As Microsoft Windows 3.1 ships with its own version of HIMEM.SYS, you need to use the Microsoft HIMEM.SYS driver when running Windows 3.1.

Example:

The following command loads HIMEM.SYS:

```
DEVICE=HIMEM.SYS
```

POWER.SYS Driver

Function: Advanced Power Management Driver.

Type: Character device driver

Syntax: DEVICE=POWER.SYS

Parameters:

none.

Description:

The POWER.SYS driver provides power savings in the system in which it runs by monitoring the activity of Embedded DOS-ROM and its applications, and idling the processor when no computation or I/O activities are detected.

POWER.SYS can work with or without an underlying APM BIOS. If POWER.SYS detects that an underlying APM BIOS is available, then it uses the industry-standard APM BIOS calls to perform idles. If no APM BIOS is detected, then POWER.SYS idles the CPU by using the HLT instruction to stop the CPU's clock until the next interrupt arrives in the system. This has the most benefit when used with a static core CPU.

Example:

The following command loads POWER.SYS:

```
DEVICE=POWER.SYS
```

RAMDISK.SYS Driver

Function: Low Memory (Real-Mode) RAM Disk.

Type: Block device driver

Syntax: DEVICE=RAMDISK.SYS [/KBTOUSE=*nnn*]

Parameters:

nnn Specifies the number of kilobytes of main (low) memory to allocate for the RAM disk. This memory is taken away from the lower 640KB memory used for applications.

Description:

The RAMDISK.SYS driver provides emulation of a floppy disk drive of arbitrary size in low memory. Embedded DOS-ROM assigns the next available drive letter to the RAM disk.

Example:

The following command installs a 128KB RAM drive in the system:

```
DEVICE=RAMDISK.SYS  /KBTOUSE=128
```

VDISK.SYS Driver

Function: Extended Memory (Protected-Mode) RAM Disk.

Type: Block device driver

Syntax: DEVICE=VDISK.SYS [/KBTOUSE=*nnn*]

Parameters:

> *nnn*   Specifies the number of kilobytes of extended memory to allocate for the RAM
> disk. This memory is the same memory that HIMEM.SYS uses as well, so
> specifying a smaller number here results in more extended memory becoming
> available for XMS clients.

Description:

> The VDISK.SYS driver provides emulation of a floppy disk drive of arbitrary size in
> extended memory. Embedded DOS-ROM assigns the next available drive letter to the
> RAM disk.

Example:

> The following command installs a 512KB RAM drive in the system:

```
DEVICE=VDISK.SYS  /KBTOUSE=512
```

ROMDRIVE.SYS Driver

Function: Protected Mode ROM Disk.

Type: Block device driver

Syntax: DEVICE=ROMDRIVE.SYS

Parameters:

> *none.*

Description:

> The ROMDRIVE.SYS driver automatically scans the address space starting at 1MB
> (100000h) in 256KB increments looking for valid boot records. If found, this driver
> creates a new drive that corresponds with that ROM disk image in ROM.
>
> ROMDRIVE.SYS creates as many drives as it can find ROM images in the address space
> above 1MB, up to a maximum of 10 drives in the system. If the address space has double-
> mappings; i.e., the hardware maps the same ROM into several physical addresses, then
> multiple identical drives will be created; the extra ones can be ignored.

While ROMDRIVE.SYS is normally used in an embedded system to map a drive to an image of a disk in ROM, it can also be used in conjunction with ROMCLONE.SYS to test-out a ROM disk image directly on a PC without burning a ROM (see ROMCLONE.SYS for details).

To change the increment by which ROMDRIVE.SYS scans, the code can be found in the DRIVERS directory of the FULL SOURCE Adaptation Kit. Note that ROMDRIVE.SYS scans the full address space above 1MB on each processor. The 286 processor has a 16MB address space, while the 386 and above processors have a 4.2GB address space. ROMDRIVE.SYS automatically detects the processor type to determine how to scan the address space.

Example:

The following command scans for ROM images and creates drives for them automatically:

```
DEVICE=ROMDRIVE.SYS
```

ROMCLONE.SYS Driver

Function: ROM Emulator.

Type: Utility device driver

Syntax: DEVICE=ROMCLONE.SYS /FILE=*filename.ext*

Parameters:

   filename.ext   Specifies the name of the file containing a ROM image that is to be loaded
          into memory at the 1MB address boundary.

Description:

The ROMCLONE.SYS driver pre-loads the RAM memory starting at the 1MB address boundary (100000h) with the contents of the specified file. Once this RAM has been initialized with what appears to be a ROM image of a disk, it will be recognized as a ROM disk image by ROMDRIVE.SYS. This makes it easy to boot a Embedded DOS-ROM system on a PC and test a ROM disk image without burning Flash or EPROM.

ROMCLONE.SYS does not allow HIMEM.SYS to function with it, as it loads the file at the 1MB address boundary. ROMCLONE.SYS modifies the BIOS data area to indicate that all of the memory above 1MB is unavailable to protect the integrity of the ROM disk.

In order to function properly, ROMCLONE.SYS must be loaded *before* ROMDRIVE.SYS is loaded in CONFIG.SYS.

Example:

The following commands load a disk image from a file called DISK into RAM and then treats it as a ROM disk:

```
DEVICE=ROMCLONE.SYS  /FILE=DISK
DEVICE=ROMDRIVE.SYS
```

SERDRIVE.SYS Driver

Function: Remote Drive (Serial Port).

Type: Block device driver

Syntax: DEVICE=SERDRIVE.SYS [/CON]

Parameters:

/CON   Specifies that all keyboard and video BIOS calls are to be redirected over the same serial link. The corresponding INTERSVR.EXE program demultiplexes console I/O and disk I/O and handles them in tandem.

Description:

The SERDRIVE.SYS driver automatically attempts to communicate with a host over its COM1 (I/O port 3f8h) port, expecting to interact with INTERSVR.EXE, running on a host machine.

Once connected, SERDRIVE.SYS creates a new drive letter and redirects all I/O for that drive to the host via the serial communications line.

SERDRIVE.SYS automatically determines the host computer's communication speed, attempting 115Kbaud first, then 57600 baud, then 38400, 19.2K, and finally 9600. If the host does not respond at any of these baud rates, then I/O is not redirected and the device driver performs no function.

The baud rate for the communications line is established on the host side when running INTERSVR.EXE (see the command reference for details).

If the /CON parameter is specified on the DEVICE= command in CONFIG.SYS, then SERDRIVE.SYS also redirects any INT 10h and INT 16h video and keyboard BIOS requests over the same serial communications line, so that it can be handled on the host. The effect is that both disk and console I/O can be performed through the same serial communications link transparently.

Example:

The following command causes SERDRIVE.SYS to redirect disk I/O and console I/O to the host:

```
DEVICE=SERDRIVE.SYS  /CON
```

The following command optionally loads SERDRIVE.SYS without redirecting console I/O by using the '?' feature in CONFIG.SYS.

```
? DEVICE=SERDRIVE.SYS
```

Chapter 3

MINI-COMMAND REFERENCE
(not COMMAND.COM)

Embedded DOS-ROM can be configured to support a built-in command interpreter called mini-COMMAND. This command interpreter is a minimalistic application that recognizes common DOS commands, but does not provide the flexibility that the larger COMMAND.COM program does. The following is a summary of the syntax supported by the mini-COMMAND interpreter. The command descriptions for these commands are described in more detail in Chapter 4.

HELP Command

Syntax: HELP

Also: ?

BREAK Command

Syntax: BREAK [ON|OFF]

CD Command

Syntax: CD [drive:][pathname]

Also: CHDIR [drive:][pathname]

CLS Command

Syntax: CLS

COPY Command

Syntax: COPY [drive:]pathname [drive:]pathname

DATE Command

Syntax: DATE [mm-dd-yy]

DEL Command

Syntax: DEL [drive:]pathname

Also: DELETE [drive:]pathname
ERA [drive:]pathname
ERASE [drive:]pathname

DIR Command

Syntax: DIR [drive:][path][wildcard-filename]

ECHO Command

Syntax: ECHO [ON|OFF|string]

EXIT Command

Syntax: EXIT

GOTO Command

Syntax: GOTO label

IF Command

Syntax: IF [NOT] ERRORLEVEL n statement

Also: IF [NOT] EXIST filename statement

MD Command

Syntax: MD [drive:]path

Also: MKDIR [drive:]path

PATH Command

Syntax: PATH [path1[;path2][;path3][;...]

PAUSE Command

Syntax: PAUSE

PROMPT Command

Syntax: PROMPT string

RD Command

Syntax: RD [drive:]path

Also: RMDIR [drive:]path

REM Command

Syntax: REM any comment

REBOOT Command

Syntax: REBOOT

REN Command

Syntax: REN [drive:][path]filespec filespec

Also: RENAME [drive:][path]filespec filespec

SET Command

Syntax: SET [keyword=[string]]

SYNC Command

Syntax: SYNC

Also: SYNCH

TIME Command

Syntax: TIME [hh:mm:ss[.hh]]

TYPE Command

Syntax: TYPE [drive:][path]filespec

VER Command

Syntax: VER

VERIFY Command

Syntax: VERIFY [ON|OFF]

VOL Command

Syntax: VOL [drive:]

Chapter 4

COMMAND REFERENCE
LOADABLE COMMAND.COM

Embedded DOS-ROM comes with a COMMAND.COM command interpreter that processes intrinsic commands (such as COPY), external commands (such as CHKDSK), and batch files (such as AUTOEXEC.BAT). This section presents the set of intrinsic and external commands supported by COMMAND.COM and its associated utilities.

4.1 Basic User Concepts

Throughout the rest of this section, we will refer to various terms that have a specific meaning beyond their normal English meaning. You should be familiar with these terms and their associated concepts before using Embedded DOS-ROM commands.

4.1.1 Files, Directories, and Filenames

Embedded DOS-ROM allows you to create, delete, and manipulate objects called *files*. Files contain zero, one, or more bytes of information, and are stored electronically on floppy diskettes, hard disks, laserdisk, or other storage media. Every file has a name, or *filename*, and these names are stored in special system files called *directories*.

Files may be created and duplicated with the COPY command. The DEL and ERASE commands are used to delete files.

Directories can be created, deleted, and manipulated with the Embedded DOS-ROM command processor. The DIR command can be used to list the contents of a directory. The MKDIR command makes a new directory, and the RMDIR removes a directory.

Directories also have names, and these names are stored along with other files' names. The master directory is named "\", pronounced *root*. The root directory contains files and subdirectories, as do all of those subdirectories.

Filenames (and directory names) can be typed in uppercase or lowercase letters, and can also include special symbols such as '&' and '+', as well as numbers. For DOS-compatible file systems, filenames follow a rigid format that is a carry-over from the days of CP/M and MS-DOS:

```
filename.ext
```

where *filename* is a 1- to 8-character name, and *ext* is a 0- to 3-character extension to the name. Both parts are separated by a period (.) when the extension is specified. When no extension is specified, the period is not necessary, but may be specified.

4.1.2 Wildcarded Filenames

Normally, filenames (and directory names) refer to exactly one file; for example, when creating a directory with MKDIR and deleting files with DEL.

Some commands, such as COPY, DIR, and DEL, allow the use of a special form of a filename that lets you specify that some parts of the filename are "don't cares". These don't care symbols ('?' and '*') are called wildcards.

The '?' wildcard character can be used in place of any character in the name or extension parts of a filename, and tells Embedded DOS-ROM that you mean all files that match the filename without regard to that character. For example:

```
BILL.DOC
OPUS.DOC
BILLCAT.TXT
FDISK.COM
BASIC.EXE
BANANA
BILLFOLD.
```

are all valid filenames, and all of them but OPUS.DOC and FDISK.COM start with the letter 'B'. By referring to the wildcarded filename, "B???????.???", you can indicate all files that start with a letter 'B'. If you wanted all files that have an extension of "DOC", then you could use:

```
????????.DOC
```

Similarly, you could refer to all files (namely, BANANA and BILLFOLD.) that have no extensions by specifying the following wildcarded filename that has a period but no extension:

```
????????.
```

Because it is common to want to fill-out the remainder of either the name or extension parts of a wildcarded filename with the '?' wildcard character, Embedded DOS-ROM provides the '*' wildcard character to serve as a shorthand wildcard. This character instructs Embedded DOS-ROM to wildcard the remainder of the name or the extension (depending on which field contains

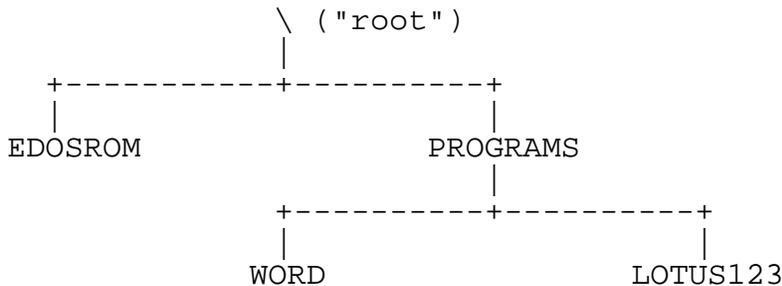the character). Thus, the following wildcarded filename specifies all files that begin with the letter 'B':

```
B*.*
```

Similarly, you could refer to all files that have a "DOC" extension with the following notation:

```
*.DOC
```

4.1.3 Pathnames

Embedded DOS-ROM stores filenames in directories, and those directories can contain subdirectories. Even so, the directory that contains all of the other files and subdirectories on a disk is the root directory, "\". This allows you to organize your files by using subdirectories that form a *tree* of directories and files. For example, if you have a word processor program, and a spreadsheet program, and your Embedded DOS-ROM program utilities, you might want to create a directory structure like the one below to organize your the files:

```
                    \ ("root")
                    |
   +----------+----------+
   |                     |
 EDOSROM             PROGRAMS
                        |
          +----------+----------+
          |                     |
        WORD                LOTUS123
```

This allows you to separate files into logical groups, and also allows you to create and maintain filenames that may by coincidence have the same exact filenames by keeping them in their own subdirectories. For example, your word processor and spreadsheet programs might both contain "READ-ME" files that could actually have the same filename. This subdirectory organization keeps them separate.

In our example, the PROGRAMS subdirectory contains two subdirectories, WORD and LOTUS123. In order to properly refer to the 123.EXE program in the LOTUS123 directory, you would need to specify a full *pathname* of that file that tells Embedded DOS-ROM where to find the file. The proper pathname for a file called 123.EXE located in the LOTUS123 subdirectory in our example is:

```
\PROGRAMS\LOTUS123\123.EXE
```

4.1.4 The Current Directory

Of course, with many nested subdirectories, specifying the full pathname of a file every time it is used could become tedious. To help make the job of specifying files easier, Embedded DOS-ROM maintains a *current directory* for each drive in your workstation.

The current directory for any drive can be displayed with the following command:

```
C> CD d:
d:\PROGRAMS
C>
```

where d: is the name of the drive to display the current directory for. To change the current directory, simply specify the pathname of the directory to change to. For example, to change into the LOTUS123 directory of our example as shown above (assuming drive C), you would type the following:

```
C> CD C:\PROGRAMS\LOTUS123
C>
```

Thereafter, you could refer to the 123.EXE file in that directory simply as 123.EXE, instead of specifying the full pathname.

Embedded DOS-ROM knows that you are referring to a filename in the current directory when you leave off the backslash at the beginning of a filename. If you start a pathname with a backslash, then Embedded DOS-ROM assumes that you are specifying a pathname that starts from the root directory.

4.1.5 The Current Drive

Because most users usually use files on one drive (a hard disk, for example), Embedded DOS-ROM maintains a *current drive* for you so that, if you do not specify a drive letter in a pathname, Embedded DOS-ROM assumes that you mean the default drive.

When Embedded DOS-ROM starts, it sets the default drive to the one that it boots from. For example, if you booted your workstation with a floppy diskette in drive A, then the current drive would also be A by default. To change the default drive to another drive letter, simply type that drive letter followed by a colon (:) at the prompt. For example, if you booted from drive A but wanted to change to drive C afterward, you would type the following:

```
A> C:
C>
```

4.1.6 Putting it All Together

You can use all of the concepts in this section in combination when you use Embedded DOS-ROM. When constructing a filename, you must think about which components you will need to specify so that Embedded DOS-ROM fully understands where the file is located. The full form of a file specification is shown below:

```
[d:][pathname][filename][.[ext]]
```

The *d:* is an optional drive letter, and need only be specified if the file is located on a drive that is not the default drive.

The *pathname* field may omitted, in which case Embedded DOS-ROM assumes that the file is in the default directory. If you specify a pathname, and it starts with a backslash (\), then Embedded

DOS-ROM knows that you are specifying a pathname that starts from the root directory. Otherwise, Embedded DOS-ROM treats the pathname as relative to the current directory.

The *filename* field is required; however, it may contain wildcard characters when the command supports it.

The *ext* field is optional, and if it omitted, only those files with no extensions will match the filename. When omitted, the period may be specified or omitted. This field, like the *filename* field, can be wildcarded with '?' and/or '*'.

4.1.7 Special Filenames

Embedded DOS-ROM reserves some filenames for itself. In all systems, the following filenames are actually names of *devices* in the system, and cannot be used as filenames or pathnames to name your own files:

```
CON  - Console keyboard and screen
AUX  - First serial port
PRN  - First printer port
NUL  - Null device
COM1 - First serial port
COM2 - Second serial port
COM3 - Third serial port
COM4 - Fourth serial port
LPT1 - First parallel port
LPT2 - Second parallel port
LPT3 - Third parallel port
CLOCK$ - The system clock device
```

These names may be used when referring to these devices; however, you cannot name your files by these names. For example, if you wanted to copy data from a file called README.TXT to the printer connected to the first parallel port, you could use the following Embedded DOS-ROM command:

```
C> COPY README.TXT LPT1
```

Similarly, if you wanted to type some data from the console's keyboard into a file called MYFILE.DAT, you could use the following sequence:

```
C> COPY CON MYFILE.DAT
The men of the town go to the little island
to find coal for their city.  They go down
to the dock at eight and fight for a cozy
spot on the prow of the boat.
^Z
C>
```

Afterwards, you could use the following command to display the contents of the file:

```
C> TYPE MYFILE.DAT
The men of the town go to the little island
```

```
        to find coal for their city.  They go down
        to the dock at eight and fight for a cozy
        spot on the prow of the boat.
        C>
```

Alternatively, you could use the special device name CON with the COPY command to achieve the same effect:

```
        C> COPY MYFILE.DAT CON
        The men of the town go to the little island
        to find coal for their city.  They go down
        to the dock at eight and fight for a cozy
        spot on the prow of the boat.
        C>
```

Embedded DOS-ROM reserves some special extensions for special purposes. Files having a "SYS" extension are device drivers and cannot be changed. Some other extensions are reserved to indicate what types of data are in the files, and have special meaning to Embedded DOS-ROM:

```
        Extension Type of File
        .BAT       Batch file, contains commands to run.
        .EXE       Executable program, BINARY format.
        .COM       CP/M compatible program, BINARY format.
        .SYS       Embedded DOS-ROM device driver or system file.
```

4.2 Starting an Embedded DOS-ROM Session

To begin using Embedded DOS-ROM, select "Embedded DOS-ROM" in the Advanced CMOS Configuration SETUP screen, and boot your target. Embedded DOS-ROM, when properly configured as the primary operating system, will load and display a sign-on banner similar to the following (note that OEM adaptations of Embedded DOS-ROM may display slightly different sign-on banners):

```
Starting Embedded DOS-ROM...

Current date is Fri  03-03-96
Enter new date (mm-dd-yy):_

Current time is 15:24:02.84
Enter new time:_

C>_
```

The system is now ready to accept commands. The "C>" is printed by the Embedded DOS-ROM command processor program, COMMAND.COM, and is called "*the prompt*." You can change the style of the prompt to include other information besides the current drive letter. To learn how to do this, consult the PROMPT command reference later in this chapter.

When the Embedded DOS-ROM command processor is loaded, it automatically scans your boot drive for the file, AUTOEXEC.BAT. If found, it reads commands from that file first, and when all of those commands are finished executing, it will display the prompt and read commands from

the keyboard. If there is no AUTOEXEC.BAT file, then the command processor just asks for the date and time, and then prints the prompt right away.

4.3 Ending an Embedded DOS-ROM Session

To end your session with Embedded DOS-ROM, you must first exit the program that you are running **before** turning off your machine. This causes all files used by the currently running program to be closed properly.

When you exit the program, you should see the command prompt displayed by the Embedded DOS-ROM command processor:

```
C>_
```

4.4 The Embedded DOS-ROM Command Shell

The program that Embedded DOS-ROM runs automatically after booting is COMMAND.COM, the Embedded DOS-ROM command shell. You may substitute another program for the command shell by specifying the SHELL= statement in the CONFIG.SYS file.

The command shell, or command processor, is responsible for reading commands from the keyboard or from a file called a batch file and executing the commands by recognizing the verb of each command as a specific directive to do something. Most of this chapter is an alphabetical list of command names and a description of each command's functions.

The command processor has many features that help you use the resources of your target more efficiently. While most commands read their input keyboard and display output on the screen, the input and output functions can be redirected to files and even to other programs. COMMAND.COM also provides an *environment space* that allows you to configure the operation of the command shell to meet your specific needs. Finally, the command shell can read its input from a file, either through running COMMAND.COM as a program and redirecting its input and/or output, or by executing a *batch file*. These facilities are the subject of the next few sections.

4.4.1 Redirection

When a program or internal command is executed, the command shell routes the input of the executed program to the keyboard, and its output to the console screen. You can redirect either or both the standard input and output streams to files, so that a program reads its input from a file, and/or writes its output to a file.

To redirect a program's output file, simply follow the program's name and arguments with a '>' symbol, and the pathname to which output should be written. For example, if we wanted to save a directory listing to a file called SAVEDIR.TXT, we could do this:

```
C> DIR > SAVEDIR.TXT
C>
```

Similarly, we could use the SORT command to collect lines of text from the keyboard, and write the sorted results to a file called SORTED.TXT:

```
C> SORT > SORTED.TXT
This is the first line.
The second line.
The last line.
^Z
One file copied.
C>
```

To redirect the input of a program (such as SORT for example), just follow the command with a '<' symbol, and follow it with the name of the file from which input should be read. Suppose we wanted to sort our directory we created by redirecting DIR's output into SAVEDIR.TXT. Let's redirect SORT's input to that file and let the output go to the screen:

```
C> SORT < SAVEDIR.TXT


        8 File(s)   5230592 bytes free
 Directory of  C:\SRC\EDOSROM\DOC
 Volume in drive C has no label
 .            <DIR>         1-23-96    4:08p
 ..           <DIR>         1-23-96    4:08p
 ARCH     DOC    483840  11-07-95    1:09p
 NORMAL   BAK      1024   2-03-96    4:34p
 NORMAL   STY      1024   2-03-96    4:35p
 SAVEDIR  TXT       123   2-03-96   10:52p
 USER     BAK    160256   2-03-96   10:35p
 USER     DOC    162816   2-03-96   10:52p
```

Finally, both the input and output streams of a program may be redirected. Suppose we wanted to sort the same file, but write the sorted output to a file called SORTDIR.TXT, for later editing with a text editor:

```
C> SORT < SAVEDIR.TXT > SORTDIR.TXT
C>
```

Redirection is a powerful tool. Together with the pipe construct, redirection can be used as "glue" to bind modular programs together to create new, custom programs.

4.4.2 Piping

The following command sequence lists the contents of a directory into a temporary file, and then sorts the file, copying its output to the console screen:

```
C> DIR > FILE.TMP
C> SORT < FILE.TMP
```

The same effect can be achieved by "gluing" both programs together with a *pipe*. Using a pipe, the command processor automatically builds the temporary file and executes both commands for you. Here is the same sequence using piping:

```
C> DIR | SORT
```

Any number of programs may be piped together to form a pipeline. The command processor executes the commands sequentially; no multitasking actually occurs. Temporary files are created in the current directory of the default drive, and are removed when they are no longer needed.

4.4.3 The Environment Space

The command processor maintains an area of memory that contains variables and associated values. You can examine and change these variables with the SET command, detailed later in this chapter.

Application programs can also access this area, and commonly require setup information to be stored in the environment prior to being run. See your application program's owner's manual for a list of any environment variables that are supported by the application.

There are two very important variables used by the command shell itself; namely, COMSPEC and PATH.

The COMSPEC variable is assigned the full pathname, including boot drive, of the copy of the command processor that was loaded. This allows COMMAND.COM to reload itself under certain circumstances. Other programs may examine this variable to determine the drive on which utility programs might be located, for example.

The PATH variable is used to store the program search path, described next.

4.4.4 Program Search Path

When you run an application program, the command processor attempts to run the program from the current directory on the default drive. If it cannot be found in that directory, then it searches the PATH variable for other directories to try. The PATH variable is normally set to the following general format:

```
PATH=dir1;dir2;dir3;...;dirn
```

where *dir1* is the first directory to scan for the program, *dir2* is the second, and so on, until the last directory scanned is *dirn*.

In order for the command processor to always find its external utility programs, you may want to copy them to a directory called C:\GS. Then, you could set up the PATH variable to cause COMMAND.COM to scan that directory whenever it can't find a program name in the current directory:

```
C> SET PATH=C:\GS
```

4.4.5 Batch Files

Embedded DOS-ROM lets you store one or more commands in files with a special extension, BAT. These files are called *batch files*. By entering the filename of the batch file as a command, the command processor can run the commands in the file for you automatically.

Batch files can be very simple or extremely complex. A very simple batch file might delete all of the temporary files in the current directory that have a TMP extension:

```
C> COPY CON DELTEMP.BAT
del *.tmp
^Z
One file copied.
C> DELTEMP              (the batch file is run)
C>
```

To provide more flexibility, you can pass parameters to batch files on the command line. These parameters are made available in the form of special variable names; namely, %1, %2, %3, %4, %5, %6, %7, %8, and %9.

For example, if you wanted to implement a batch file that would move a file across directories or drives (the RENAME command cannot do this, as it is not strictly a renaming process), you might code the following lines in a batch file called MOVE.BAT:

```
COPY %1 %2
DEL %1
```

Sometimes you need to process a variable number of arguments on a batch file's command line. This requires a way to loop through all of the arguments, and processing them until an empty one is found.

For example, if you wanted to implement a batch file that would produce a directory listing of every argument that you pass on the command line (a "super directory" batch file), you might code the following lines:

```
:BIGLOOP
IF "%1"=="" GOTO ALLDONE
DIR %1
SHIFT
GOTO BIGLOOP

:EXIT
```

Another way of doing this might be:

```
FOR %%I IN (%1 %2 %3 %4 %5 %6 %7 %8 %9) DO DIR %%I
```

However the process is done, batch files can be powerful tools that automate daily chores. Once properly debugged, batch files can eliminate the human errors involved in backing up data, and many other tedious chores.

Chapter 7

ALPHABETICAL LIST OF COMMANDS

The remainder of this chapter describes each Embedded DOS-ROM command in great detail. Commands are categorized into two classes: External and Internal. External commands are actually separate utility programs supplied with Embedded DOS-ROM. They are loaded and run by the command processor when you type their name. Internal commands are built into the command processor and do not run any programs.

: Command

Function: Define Batch File Label.

Type: Internal

Syntax: *:label*

Parameters:

> *label*   Batch file label name

Description:

> The : command marks a point of execution in a batch file with a name, or *label*, that can be used by the GOTO command to transfer control to the point in the batch file.
>
> Looping and conditional execution with the IF command are possible through the use of labels defined with this command.
>
> Lines containing a label definition may *not* contain another command itself; label definitions must occupy their own command line.

In interactive mode, label definitions are simply ignored.

Example:

The following command language fragment might be used in a batch file to loop repeatedly through all of the batch file's arguments.

```
:MAINLOOP
IF "%1"=="" GOTO EXIT
IF EXIST %1 ECHO The file %1 exists.
SHIFT
GOTO MAINLOOP

:EXIT
```

@ Command

Function: Disables Echo on Command Execution.

Type: Internal

Syntax: *@command*

Parameters:

  command Embedded DOS-ROM command to be executed

Description:

The @ command executes the specified command in a batch file without echoing the prompt or the command itself, even when the ECHO flag is OFF.

The everyday use of this command is to disable the echoing of the ECHO OFF command at the start of a batch file. The @ command does not change the status of the ECHO flag itself, however.

Example:

The following command might be placed at the beginning of a batch file to disable all echo output within the batch file:

```
@ECHO OFF
```

ASK Command

Function: Prompts for and accepts operator input interactively.

Type: Internal

Syntax: ASK *varname=string*

Parameters:

      varname   Environment variable to set
      string   Prompt string

Description:

The ASK command displays the specified string on the console and waits for the operator to type a line of input at the prompt. When the operator presses the carriage return, ASK stores the typed-in text into the specified environment variable so that it may be inspected by an IF statement, or a program.

Example:

```
C> ASK COLOR=What color do you want?
What color do you want? RED
C> SET
COMSPEC=C:\COMMAND.COM
PATH=C:\DOS
COLOR=RED
C>
```

ATTRIB Command

Function: Displays, sets, or resets file attributes.

Type: External

Syntax: ATTRIB [+R | -R] [+A | -A] [*d:*][*pathname*]

Parameters:

      +R   The file is marked read-only
      -R   The file is marked read/write
      +A   The file is marked for archiving
      -A   The file is marked as unmodified
      d: Drive letter on which the specified files reside
      pathname Wildcarded pathname of files to affect or display attributes for

Description:

The ATTRIB utility program displays the attributes for a set of files, or can be used to change the read-only and archive attributes of the files.

If none of the +R, -R, +A, or -A parameters are specified, then the attributes of the specified files are displayed.

If the +R parameter is specified, then the files are marked read-only, so that they cannot be deleted inadvertently, and so that they are protected from accidental writing by application

programs. Caution: Programs may modify the read-only attribute themselves, effectively reversing this protection.

If the -R parameter is specified, then any read-only marks are removed from the files' directory entries. This has the effect of allowing the files to be deleted with the DEL or ERASE commands, and the files can be overwritten by application programs.

If the +A parameter is specified, then the files are marked archivable, causing some backup programs to automatically select the program for backup.

If the -A parameter is specified, then the archivable marks are removed from the files, causing some backup programs to skip over the files when performing a full backup.

If the drive letter is specified, then ATTRIB searches that drive for the specified files. If no drive letter is specified, then ATTRIB uses the default drive.

ATTRIB requires you to specify an (optionally wildcarded) pathname that indicates which files are to be affected. Both the * and ? wildcard characters can be used.

Example:

```
C> ATTRIB +R *.WKS
C>
```

Batch Files

Function: Execute pre-recorded list of commands.

Type: External

Syntax: [*d:*][*pathname*] [*parameter1*] [*parameter2*] [*...*]

Parameters:

>   *d:*   Drive letter on which the specified batch file resides
>   *pathname*   Pathname of a file containing list of commands
>   *parameter1, parameter2* - optional parameters

Description:

A batch file is a text file that contains zero, one, or more lines, each containing a valid Embedded DOS-ROM command as it would be typed manually at the system prompt. Once these commands are recorded in a batch file (with a text editor, for example), the entire sequence of commands may be executed by simply using to the batch file's name as a command name.

Batch files must always have a .BAT extension. When invoking the batch file, the extension is not used as a part of the command.

Up to ten parameters, may be passed to a batch file by simply including them as parameters to the command name. Embedded DOS-ROM automatically assigns them

special numeric names that can be used in commands that are inside the batch file. When
the commands in the batch file are executed by the Embedded DOS-ROM command
processor, the special names are automatically expanded to the names you specify on the
command line. The names of the parameters consist of a percent sign, followed by a
number, starting from 1 for the first one. For example:

```
C> COPY CON COPYIT.BAT
copy %1 %2
^Z
C> COPYIT thisfile.dat theother.dat
One file copied.
C>
```

Environment variables may also be passed to batch files through a similar syntax. To refer
to the *value* of an environment variable, simply enclose the *name* of the variable (in upper
case) with percent signs (%). For example:

```
C> SET DESSERT=CAKE
C> ECHO We are eating %DESSERT% tonight.
We are eating CAKE tonight.
C>
```

To stop a currently-executing batch file, you can press either CTRL-C or CTL-BRK.
Embedded DOS-ROM will display the message:

```
Terminate batch job (Y/N)?
```

If you type a Y, then the batch file will stop executing and the Embedded DOS-ROM
command processor will issue a prompt for you to input the next command from the
keyboard. If you type N, then the interrupted program will be aborted, but the batch file
will continue with the next command.

BREAK Command

Function: Displays or changes the status of the BREAK flag.

Type: Internal

Syntax: BREAK [ON | OFF]

Parameters:

> ON   Turns on ^C and CTL-BRK checking
> OFF   Turns off ^C and CTL-BRK checking

Description:

> The BREAK command changes or displays how Embedded DOS-ROM handles break-ins
> by the console user with ^C and CTL-BRK key sequences. If BREAK is ON, then
> Embedded DOS-ROM will break out of a running program or batch file when the ^C or

CTL-BRK keys are pressed. If BREAK is OFF, then Embedded DOS-ROM will not break out, but will instead pass the keys pressed to the program.

Turning the BREAK flag ON makes it much easier to terminate a run-away batch file or program; however, some programs may not provide adequate safeguards against break-ins by the console user. Turning the flag OFF improves the protection of running batch files and programs, but makes it very difficult to terminate them if they run away.

Examples:

If no parameters are specified on the BREAK command, then the status of the BREAK flag is displayed. For example:

```
C> BREAK
BREAK is ON.
C>
```

If ON or OFF is specified, then the BREAK flag is set to that value. For example:

```
C> BREAK OFF
C> BREAK
BREAK is OFF.
C>
```

CALL Command

Function: Executes pre-recorded list of commands as a subroutine.

Type: Internal

Syntax: CALL [*d:*][*pathname*] [*parameter1*] [*parameter2*] [*...*]

Parameters:

d: Drive letter on which the specified batch file resides
pathname   Pathname of a file containing list of commands
parameter1, parameter2 - optional parameters

Description:

The CALL command invokes a batch file, but unlike simply specifying the batch file's name as the command, CALL instructs the command processor to treat the batch file as a subroutine that will return to the currently-executing batch file upon return. Normally, when batch files are invoked, they "chain" but do not call each other in this way.

The following example illustrates how the CALL command can be used within a batch file to call another batch file to perform some work, and then return to the original batch file:

```
C> COPY CON COPYIT.BAT
copy %1 %2
call REPORT.BAT
```

```
^Z

C> COPY CON REPORT.BAT
echo We're done!
^Z

C> COPYIT thisfile.dat theother.dat
One file copied.
We're done!
C>
```

CHDIR Command

Function: Displays or changes a drive's current directory.

Type: Internal

Syntax: CHDIR [*d:*][*path*]

Parameters:

> *d:*   Drive letter to change/display the current directory for
> *path*   New current directory for the specified drive

Description:

> The CHDIR command (abbreviated CD) displays the current directory of the specified drive, or can change the current directory of the specified drive. If no drive is specified, then the default drive is used.

> Each volume in the system is assigned a drive letter by Embedded DOS-ROM at system initialization time. Each drive letter has associated with it a "current" directory pathname, which is used internally by Embedded DOS-ROM to interpret pathnames that are not fully specified. When the system starts, the current directory of each drive is initialized to the root directory (\).

> The current directory of a drive may be changed without changing to that drive itself by specifying a drive letter.

> The current directory of the default drive may be changed by not specifying the drive letter.

> If no parameters are specified, then the current directory is displayed for the default drive. If only a drive letter is specified, then the current directory is displayed for the specified drive.

Examples:

> The following command displays the current directory of the default drive:

```
C> CHDIR
```

```
C:\
C>
```

The following command changes the current directory of the current drive to the
WINDOWS subdirectory under the root directory:

```
C> CHDIR WINDOWS
C>
```

The following command changes the current directory of drive E to the root (top-level)
directory:

```
C> CHDIR E:\
C>
```

CHKDSK Command

Function: Analyzes and maintains file system integrity.

Type: External

Syntax: CHKDSK [*d:*][*pathname*] [/F] [/V]

Parameters:

d:	Drive letter containing file system to inspect
pathname	Wildcarded pathname of files to analyze for contiguity
/F	Fixes, or corrects, any file system errors
/V	Displays each file as it is analyzed by CHKDSK

Description:

The CHKDSK utility program analyzes the file system residing on a drive to determine if it
contains structural errors. At the user's option, CHKDSK can be instructed to correct the
structural errors.

If no parameters are specified, then CHKDSK inspects the default drive, but does not
check any files' contiguity, nor does it fix any structural problems. CHKDSK displays a
list of structural errors as they are found, and performs no corrective action.

If a drive letter is specified, then CHKDSK will operate on the specified drive. Because
Embedded DOS-ROM may assign drive letters to non-DOS file system types (including
NetWare and OS/2 HPFS, for example), the file system must be supported by CHKDSK
before it can correctly analyze the file system's structure. If the file system is not one of
the supported types, CHKDSK will display a message to that effect, and will not check the
file system.

If a pathname with optional wildcard characters (* and ?) is specified, then those files are
checked by CHKDSK to determine if they are stored contiguously on the storage media.
Files that are stored in one chunk can be processed by Embedded DOS-ROM more
efficiently than those that are fragmented into many pieces. This option can be used to

determine if the performance of a program such as a database manager can be improved simply by copying the file to a new location in the file system.

If the /F option is specified, then CHKDSK will correct any structural damage to the file system. Because this action is highly file system-specific, a list of actions is not given here. Generally, CHKDSK will determine that files do not accidentally share allocated storage blocks, and that their sizes as recorded in the directory are consistent with the allocated storage blocks for those files.

If the /V option is specified, then CHKDSK will display the name of each file as it is analyzed. This allows the user to observe the progress of CHKDSK as its operations commence.

Before terminating, CHKDSK reports general statistics about the file system's storage capacity, and the amount of space used by different types of file system objects such as directories, files, hotfix areas, and so on.

CLS Command

Function: Clears the screen.

Type: Internal

Syntax: CLS

Parameters:

   none.

Description:

   The CLS command clears the workstation's screen and resets the cursor position to the upper-left hand corner of the screen. The next prompt is issued on the top line of the screen.

COMMAND Command

Function: Runs the Embedded DOS-ROM command shell.

Type: External

Syntax: COMMAND [/E:*nnnn*] [/P] [/X] [/C *string*]

Parameters:

   nnnnn Maximum environment size in bytes.
   string Command with arguments to execute.

Description:

The COMMAND program utility is a command shell that actually runs as an application program. When Embedded DOS-ROM boots, it runs COMMAND as the default shell, unless the user places a SHELL= statement in the CONFIG.SYS file that directs Embedded DOS-ROM to another shell program.

COMMAND reads commands, one per typed line, from standard input, and writes its messages to standard output.

If the /E option is specified, then the user can change the environment size to be used while the command processor is executing. The environment should be greater than 128 bytes, but less than 32,768 bytes.

If the /X option is specified, then the user can cause COMMAND.COM to not swap itself to extended memory when attempting to run a program. This may be desired when using SHELL= for the primary command interpreter, if TSRs are to be loaded from within batch files. Without specifying this option in this scenario, a new copy of COMMAND.COM is loaded after each TSR is invoked, wasting space unnecessarily.

If the /P option is specified, then COMMAND will not allow the EXIT statement to terminate the shell. Instead, the EXIT command does nothing when typed on the keyboard. This option is not intended for user application; instead, it is used by Embedded DOS-ROM to tell the command shell to remain in memory when EXIT is typed.

If the /C option is specified, then COMMAND loads into memory and executes the specified string as a command. The prompt is not displayed, and COMMAND terminates as soon as the specified command string is finished executing.

Example:

The following example runs the CHKDSK utility within another process and then returns.

```
C> COMMAND /C CHKDSK B:
```

COPY Command

Function: Copies one or more files or directories.

Type: Internal

Syntax: COPY [*d:*][*path1*] [*d:*][*path2*] [/V] [/A] [/B] [/R]

Parameters:

d:	Drive to copy files from/to.
path1	File(s) or directory to copy from.
path2	Target pathname(s) or directory.
/V	Disables the write-behind cache during the copy.
/A	Copies ASCII files up to end-of-file markers.
/B	Copies BINARY files without interpreting EOF markers.

　　　　/R　　　　　　　　Copies recursively into subdirectories.

Description:

　　　The COPY command copies one or more files (and optionally directories, with the /R option) to a new destination. If the destination path names a file, then all of the source files are written to the target file, concatenated together.

　　　If *path2* is not specified, the copy is created in the current directory on the disk in the default drive. If the source file (*path1*) is also on the default drive, and *path2* is not specified, then an error message is displayed, indicating that a file cannot be copied to itself.

　　　If the /V option is specified, then COPY temporarily turns VERIFY mode ON, so that the data will be written directly to the disk, and not into the cache, where it remains volatile until written by the system. By default, COPY leaves VERIFY in the state specified by the user, which defaults to OFF. This normally has the effect of improving COPY's performance by writing to the cache instead of the disk.

　　　If the /A option is specified, then the files will be copied in cooked mode; that is, they are assumed to contain ASCII text, and will only be copied to the first EOF mark (an embedded control-Z character). This is the default when files *are* concatenated.

　　　If the /B option is specified, then the files will be copied in raw mode; the entire files are copied without regard to their contents. This is the default when files are *not* concatenated.

　　　If the /R option is specified, then COPY recursively descends into the source file list, copying all of the subdirectories and their contents to the target. This option works similarly to the XCOPY command except that it works without loading another program.

Examples:

　　　The following command copies the Embedded DOS-ROM system file DOS.SYS from drive A to the root directory of drive C:

```
C> COPY A:DOS.SYS C:\DOS.SYS
One file copied.
C>
```

　　　The following command copies the Embedded DOS-ROM COMMAND.COM shell from the current directory of drive A to the current directory of drive C:

```
C> COPY A:COMMAND.COM
One file copied.
C>
```

DATE Command

Function:　　　Displays or changes the date.

Type: Internal

Syntax: DATE [*mm-dd-yy*]

Parameters:

> *mm-dd-yy* New date to set the workstation's clock to.

Description:

> The DATE command displays the current date (month, day, date, and year) on the screen. If a user specifies a new date on the command line, then DATE will change the date to the one specified.
>
> Embedded DOS-ROM checks the date to ensure that it is reasonably correct. For example, February 31 does not exist; therefore, Embedded DOS-ROM would reject 02-31-96 because February does not contain 31 days.
>
> This command actually sets the real-time clock in the workstation, if one exists. In AT-class machines, it updates the battery-maintained clock so that the new date will be remembered across power-downs.

Example:

> The following example displays the current date and prompts the user for a new date. The user can press the ENTER key to keep the date the way it is:

```
C> DATE
Current date is Fri  03-03-96
Enter new date (mm-dd-yy):_
```

DEL Command

Function: Deletes one or more files.

Type: Internal

Syntax: DEL [*d:*][*path*]

Parameters:

> *d:* Drive on which the files are to be deleted.
> *path* Wildcarded pathname of files to be deleted.
> /R Recursively deletes files in subdirectories

Description:

> The DEL (synonym ERASE) command deletes one or more files from a file system on a specified drive. If the specified path is a directory, all files in that directory will be deleted.

If path contains wildcards, then all files that match the wildcarded specification will be deleted.

If the /R option is specified, then DEL will recursively descend into each subdirectory specified in the path and delete *all* files in the subdirectories. The subdirectories themselves are also deleted when this option is specified.

If DEL is asked to delete a whole directory of files, then it asks the user to verify that it is okay to do so with the prompt:

```
Are you sure? (Y/N):
```

Warning: You should be very careful when issuing this command with wildcard path specifications, unless you are very familiar with the rules for expanding wildcards. Many files can be accidentally erased with a misplaced '*' character, for example.

Examples:

The following example deletes all files in the current directory of drive E:

```
C> DEL E:*.*
Are you sure? (Y/N): Y
C>
```

The following example deletes the file named FINANCE.WKS in the current directory of the default drive:

```
C> DEL FINANCE.WKS
C>
```

The following example deletes all of the files ending in the extension, ".BAK" in the current directory:

```
C> DEL *.BAK
C>
```

DELAY Command

Function: Delays a batch file for a specific machine-independent interval of time.

Type: Internal

Syntax: DELAY *seconds*

Parameters:

    seconds   Number of seconds to delay

Description:

The DELAY command pauses a batch file for a machine-independent amount of time. The system's time of day clock is queried to determine the time so that it will work on machines with different CPU speeds.

Examples:

The following example delays for 10 seconds before returning to the next command:

```
C> DELAY 10
C>
```

DELTREE Command

Function: Deletes one or more directories and their contents, including subdirectories.

Type: External

Syntax: DELTREE [/Y] [/V] [*d:*][*path*] [[path] [path]…]

Parameters:

/Y	Automatically confirms you want to delete the directory.
/V	Verbose method. Shows each file in path as it is being deleted.
d:	Drive on which the files are to be deleted.
path	Pathname or directory to be deleted.

Description:

The DELTREE command deletes one or more directories from a file system on a specified drive. If the specified path contains files and/or subdirectories, all of those files will be deleted.

If path contains wildcards, then all files that match the wildcarded specification will be deleted.

Warning: You should be very careful when issuing this command with wildcard path specifications, unless you are very familiar with the rules for expanding wildcards. Many files can be accidentally erased with a misplaced '*' character, for example.

Examples:

The following example deletes the specified directory on drive E:

```
C> DELTREE E:\STUFF
Really delete STUFF\*.*? [yn]: y
C>
```

The following example deletes multiple directories in the current directory of the default drive:

```
C> DELTREE STUFF THINGS
C>
```

DIR Command

Function: Lists the files in a directory or subdirectory.

Type: Internal

Syntax: DIR [*d:*][*path*] [/P] [/W] [/S] [/V] [/L] [/B] [/O[[-]NEDSG]

Parameters:

d:	Drive on which the files are to be deleted.
path	Wildcarded pathname of files to be deleted.
/P	Pauses the display after each screenful of information.
/W	Selects a wide, 5-up display of filenames.
/S	Search subdirectories.
/V	Select verbose listing.
/B	Select terse listing.
/L	Select lower-case listing.
/O	Sort listing.

Description:

The DIR command displays the files and subdirectories in a directory on the specified drive. If the drive is not specified, then the default drive is assumed.

The DIR command can derive its operands either from the command line, or from the DIRCMD environment variable, or both. The default switches are taken from the DIRCMD variable if it exists, and the user may override those defaults with command line switches.

DIR uses the path operand to determine which files to list. If the path is not specified, then the current directory is assumed. If the specified path is a directory name, then all files in that directory are listed. If the specified path is a wildcarded filename, then all files matching the path specification are listed.

The /P option can be used to pause the display after each screenful of directory listing. This can be useful for displaying the contents of very large directories with hundreds of files.

The /W option is used to generate a wide listing that omits the date, time, and size of each file's listing. This makes room for several files to be listed on each line, enabling 5 directory entries to be printed on each line.

The /S option is used to search all subdirectories in the directory specified.

The /V option is used to display the listing in a verbose format.

The /B option is used to display the listing in a terse format.

The /L option is used to display directory entries in lower-case.

The /O option is used to sort the directory listing according to certain criteria. If the N suboption is selected, entries will be sorted by filename. If the D suboption is selected, entries will be sorted by date/time. If the E suboption is selected, entries will be sorted by filename extension. If the S suboption is selected, entries will be sorted by file size. If the '-' is used in front of the suboptions, then the sort will be performed in reverse (descending) order. If the G suboption is specified, then subdirectory entries will be grouped.

Examples:

The following example displays a list of all of the files in the current directory of drive E:

```
C> DIR E:

 Volume in drive C has no label
 Directory of  C:\SRC\GS\DOC

.              <DIR>        1-23-96    4:08p
..             <DIR>        1-23-96    4:08p
ARCH    DOC    483840  11-07-95    1:09p
USER    DOC    162816   2-03-96   10:52p
NORMAL  STY      1024   2-03-96    4:35p
USER    BAK    160256   2-03-96   10:35p
NORMAL  BAK      1024   2-03-96    4:34p
        7 File(s)    5230592 bytes free
C>
```

The following example displays a list of files that have ".DOC" extensions in the current directory of the default drive:

```
C> DIR *.DOC

 Volume in drive C has no label
 Directory of  C:\SRC\GS\DOC

.              <DIR>        1-23-96    4:08p
..             <DIR>        1-23-96    4:08p
ARCH    DOC    483840  11-07-95    1:09p
USER    DOC    162816   2-03-96   10:52p
        2 File(s)    5230592 bytes free
C>
```

DISKCOMP Command

Function: Compares two diskettes, track-for-track.

Type: External

Syntax: DISKCOMP [*d:*] [*d:*] [/1] [/8]

Parameters:

  d: Specifies the drives to compare.
  /1   Compares one side of the disks only.
  /8   Compares only the first 8 sectors of each track.

Description:

   The DISKCOMP utility program compares the contents of two diskettes, track-by-track,
   until all of the tracks have been compared. If the disks are identical, then DISKCOMP
   displays the following message:

```
Diskettes compare OK
```

   Otherwise, DISKCOMP displays a compare-error message that indicates which tracks do
   not match.

   If the diskettes are not the same size or if they do not have the same capacity, then the
   following message is displayed:

```
Diskette types incompatible
```

   DISKCOMP will ask if more diskettes are to be compared after each compare, enabling it
   to compare a batch of diskettes without re-running the utility program.

   DISKCOMP won't work on diskettes that contain odd-sized sectors or unusual sector
   address marks (such as those employed by copy-protection schemes). DISKCOMP only
   copies 512-byte sectors.

   A return code is stored in ERRORLEVEL upon exit, based on the compare status. The
   following values are returned:

         0   Successful comparison.
         1   Compare error.
         2   Operator abort.
         3   Hard error (disk unformatted, etc.)
         4   Invalid parameters, or insufficient memory available.

Example:

   The following example compares the diskette in drive A with the diskette in drive B:

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

DISKCOPY Command

Function: Copies one floppy diskette to another diskette.

Type: External

Syntax: DISKCOPY [*d1:*] [*d2:*] [/1]

Parameters:

> *d1:*   Specifies the drive to copy from.
> *d2:*   Specifies the drive to copy to.
> /1   Copies one side only.
> /F   Forces reformatting of target.

Description:

> The DISKCOPY utility program copies the contents of one diskette to another, track-by-track, until all of the tracks have been copied. The second diskette is formatted as the copy proceeds, if it is determined to be of an incorrect format, or if it is unformatted.
>
> If the /1 option is selected, then only one side (head 0) of the source diskette will be copied to the destination diskette. If the second diskette requires formatting, then only one side of the diskette will be formatted.
>
> If the /F option is selected, then the target diskette will be formatted, even if it is determined to be preformatted. This is useful when a target disk has lost part of its formatting or has been incorrectly or partially formatted.
>
> DISKCOPY asks the user if more disks are to be copied, enabling it to be used in copying batches of diskettes from the same master without reloading the utility program for every diskette.
>
> A return code is stored in ERRORLEVEL upon exit, based on the compare status. The following values are returned:

> > 0   Successful copy.
> > 1   Recoverable read/write error occurred.
> > 2   Operator abort.
> > 3   Hard error (disk unformatted, etc.)
> > 4   Invalid parameters, or insufficient memory available.

Example:

> The following example copies the diskette in drive A to the diskette in drive B:

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

ECHO Command

Function: Displays/changes ECHO mode, displays messages.

Type: Internal

Syntax: ECHO [ON | OFF | . | *message*]

Parameters:

        ON   ECHO mode should be turned on.
        OFF   ECHO mode should be turned off.
        .   "ECHO." as a command prints a blank line.
        message   Message to be displayed.

Description:

The ECHO command has two functions; namely, control of the ECHO flag, and displaying messages in batch files.

ECHO mode controls the command processor's echoing of commands in batch files. If ECHO mode is on, then commands read from batch files are automatically echoed to the screen before they are executed. If ECHO mode is off, then commands are not echoed as they are executed. To display the current ECHO flag status, use the ECHO command without any parameters.

To display a message from a batch file, use the ECHO command with a non-empty string to be displayed. The special form of the ECHO command with a period ('.') immediately following the word ECHO (no intervening space) causes a blank line to be echoed.

Examples:

The following example displays the current setting of the ECHO flag:

```
C> ECHO
ECHO is ON.
C>
```

The following example displays the user message "Starting file copies . . ." on the screen when executed in a batch file:

```
ECHO Starting file copies . . .
```

ERASE Command

Function: Deletes one or more files.

Type: Internal

Syntax: ERA[SE] [*d:*][*path*]

Parameters:

d:	Drive on which the files are to be deleted.
path	Wildcarded pathname of files to be deleted.
/R	Recursively deletes files in subdirectories

Description:

The ERASE (synonym DEL) command deletes one or more files from a file system on a specified drive. If the specified path is a directory, all files in that directory will be deleted.

If path contains wildcards, then all files that match the wildcarded specification will be deleted.

If the /R option is specified, then ERASE will recursively descend into each subdirectory specified in the path and delete *all* files in the subdirectories. The subdirectories themselves are also deleted when this option is specified.

If ERASE is asked to delete a whole directory of files, then it asks the user to verify that it is okay to do so with the prompt:

```
Are you sure? (Y/N):
```

Warning: You should be very careful when issuing this command with wildcard path specifications, unless you are very familiar with the rules for expanding wildcards. Many files can be accidentally erased with a misplaced '*' character, for example.

Examples:

The following example deletes all files in the current directory of drive E:

```
C> ERASE E:*.*
Are you sure? (Y/N): Y
C>
```

The following example deletes the file named FINANCE.WKS in the current directory of the default drive:

```
C> ERASE FINANCE.WKS
C>
```

The following example deletes all of the files ending in the extension, ".BAK" in the current directory:

```
C> ERASE *.BAK
C>
```

EXIT Command

Function: Exits the command shell utility program.

Type: Internal

Syntax: EXIT

Parameters:

    none.

Description:

    The EXIT command terminates the current command shell and reverts control to the previous shell, provided that the current command shell is not the first one loaded in the system. The very first shell cannot be terminated with EXIT.

    If executed from a batch file, EXIT will terminate the batch file in a controlled manner, causing control to be transferred to the keyboard user.

FDISK Command

Function: Partitions a hard disk into one or more volumes.

Type: External

Syntax: FDISK [/R]

Parameters:

    /R   Read-only mode; partitions are *not* affected.

Description:

    The FDISK utility program is used during the installation or reconfiguration of a hard disk to view, install, or modify the disk's partition structure. In read-only mode, FDISK is inhibited from changing disks, allowing the user to view the status of the disk partitions.

    Unlike its MS-DOS counterpart, FDISK can create and manage many different partition types, even non-DOS ones. The following types are supported:

        _ 12-bit FAT (MS-DOS compatible)
        _ 16-bit FAT (MS-DOS compatible)
        _ Extended DOS (MS-DOS compatible)
        _ Huge FAT (MS-DOS 4, 5, and 6 compatible)
        _ OS/2 FSD (HPFS and others)
        _ OS/2 mirror partitions
        _ NetWare 286 (Novel compatible)
        _ NetWare 386 (Novel compatible)
        _ Xenix partitions (SCO-Xenix compatible)

    FDISK is a very simple menu-driven program that operates in a similar manner to its MS-DOS cousin.

Caution: When using FDISK to create or delete partitions, you risk loosing vast quantities of storage on your disk(s) with a slip of your finger. Be very careful when preparing your fixed disks with FDISK.

Be aware that FDISK numbers your hard drives starting with the number 0, and then 1, and so on. Be careful that you properly select the correct disk before editing its partition table.

FIND Command

Function: Scans one or more files for a text string.

Type: External

Syntax: FIND "*string*" [*d:*][*path*] [/V] [/C] [/N]

Parameters:

d:	Drive on which the files to be searched reside.
path	Wildcarded pathname of files to be scanned.
/V	Displays every line that does not contain the string.
/C	Counts the number of lines that contain the string.
/N	Numbers lines that are displayed.

Description:

The FIND utility program is used to scan a set of files for a text string line-by-line. If no extra options (/V, /C, or /N) are specified, then FIND simply displays each line that matches, as they are found.

If the /V option is specified, then FIND searches for lines that do *not* contain the specified text string.

If the /C option is specified, then FIND counts the number of matches (or mismatches, if /V is specified) that occur, and the number is displayed instead of the matching lines.

If the /N option is specified, then FIND displays line numbers in front of each matching (or mismatching) line as they are printed.

FIND is a member of a family of utilities known as "filters", that accept volumes of data and transform the data somehow, producing modified, reduced, or resulting, data on output. Filters are frequently connected together in the form of a "pipeline", as discussed in the section on piping and redirection, in this user's guide.

Examples:

The following example displays all the lines in the file PHONE.TXT that contain the string, George Bush:

```
C> FIND "George Bush" PHONE.TXT
```

The following example counts the lines in the file HOTSELL.LST on drive F that contain the string "DOS":

```
C> FIND /C "DOS" F:HOTSELL.LST
```

FOR Command

Function: Repeatedly executes a command over a range of arguments.

Type: Internal

Syntax: FOR *var* IN (*set*) DO *command*

Parameters:

var	Name of a variable to use.
set	Set of values to assign to *var*.
command	Command to execute for each value in *set*.

Description:

The FOR command executes a command several times, each time assigning a value from a set into a variable name. The variable name may optionally be used in the command as desired.

The variable name must start with a percent sign (%) if typed directly from the keyboard, or by two percent signs if executed from a batch file. Variable names are restricted to a single alphabetic letter.

The set consists of zero, one, or more (optionally wildcarded) pathnames. The FOR command automatically expands any wildcarded pathnames as it executes.

Any occurrences of the variable in the command will be substituted with the current value of the variable before it is run. The command may be an internal, external, or batch file command.

Examples:

The following example types out all of the files in the current directory of the default drive:

```
C> FOR %I IN (*.*) DO TYPE %I
```

This example copies all of the files having a DOC prefix to a BACKUP directory:

```
C> FOR %I IN (*.DOC) DO COPY %I \BACKUP
```

FORMAT Command

Function: Formats a diskette or hard disk partition for file system use.

Type: External

Syntax: FORMAT [*d:*] [/1] [/4] [/8] [/T:*tracks*] [N:*sectors*] [/V] [/S]

Parameters:

d:	Drive or partition to format.
/1	Diskette is to be formatted single-sided.
/4	Diskette is to be formatted double-density.
/8	Diskette is to be formatted 8 sectors per track.
/T:*tracks*	Number of tracks to be formatted.
/N:*sectors*	Number of sectors per track to format.
/V	Installs a volume label on the new file system.
/S	Installs system files in the new file system.

Description:

The FORMAT utility program prepares a hard disk partition, or a floppy diskette, for use with Embedded DOS-ROM. While hard disks are typically low-level formatted at the factory, floppy diskettes are usually shipped completely degaussed and the track structure must be established on the media. FORMAT initializes the track structure on floppy diskette media.

FORMAT also installs the skeleton of a file system (albeit an empty one) on the partition or floppy diskette so that it can be used to store files. The directory structure is initialized, and the disk blocks are all freed and readied to accept file data.

If the /S option is specified, then FORMAT copies the system files, DOS.SYS and COMMAND.COM, to the file system. Note that this option is only useful for disk-based systems; ROM-based systems run the DOS kernel from ROM.

If the /V option is specified, then FORMAT also prompts the user for an 11-character volume name, which it installs in the file system as an electronic marker that labels file system in the same way that the LABEL command does.

If the drive is not specified, then the default drive will be formatted. Otherwise, the specified drive will be formatted..

If /1 is specified, then FORMAT will only format one side of the floppy diskette. This option is not supported for hard disk partitions.

If /8 is specified, then FORMAT assumes that it should format each track with eight sectors per track. This option is not supported for hard disk partitions.

If /T:*tracks* is specified, then FORMAT only formats the specified number of tracks, enabling a portion of a diskette to be formatted.

If /N:*sectors* is specified, then FORMAT will organize each track into the specified number of sectors each. This option is designed to support new diskette media whenever

it becomes available. Not all values are supported for all media types. This option is not supported for hard disk partitions.

Examples:

The following example formats a double-sided, double-density floppy disk with a standard, MS-DOS compatible, 360Kb capacity file system:

```
C> FORMAT A: /4
```

GOTO Command

Function:　　Jumps to a label in a batch file.

Type:　　Internal

Syntax:　　GOTO *label*

Parameters:

label　　　　Name of a label to jump to.

Description:

The GOTO command causes the command processor to start executing commands that follow the specified label, in the current batch file. Labels can be inserted anywhere in batch files, and take the following form:

```
:label
```

Example:

For example, to create an infinite loop in a batch file that continuously scans for the existence of a file in a directory on a network drive N, you could use the following batch commands:

```
:MYLOOP
IF NOT EXIST TESTFILE.DAT GOTO MYLOOP
ECHO The file exists!  We are all done.
```

HELP Command

Function:　　Displays a list of available commands.

Type:　　Internal

Syntax:　　HELP

Parameters:

none.

Description:

The HELP command displays a list of the commands that are supported by the command interpreter. This is useful to a user that must use several different versions of DOS that support extensions provided by OEMs.

Examples:

The following example displays a list of commands supported by COMMAND.COM:

```
C> HELP
Available command list:

break   cd      chdir   cls     copy    date    del     dir
era     erase   exit    md      mkdir   path    prompt  ren
rename  rd      rmdir   set     time    type    ver
verify  vol     help    rem     ask     echo    goto
switch  for     if      pause   shift   call    delay

C>
```

IF Command

Function: Executes a command based on a condition.

Type: Internal

Syntax: IF [NOT] *condition command*

Parameters:

condition	Condition that must be satisfied.
command	Command to be executed if condition is TRUE.

Description:

The IF command causes a command to be executed if (or if NOT) a condition is TRUE.

The condition can take any of three forms:

- Testing the value of ERRORLEVEL:

    ERRORLEVEL *level*

    This condition is TRUE when the last external command returned the specified status. See DISKCOPY and DISKCOMP for examples of two commands that do this.

- Testing the existence of a file:

    ```
    EXIST [d:][path]filename[.ext]
    ```

    This condition is TRUE only if the specified file exists.

- Testing equivalence of two strings:

    string1 == string2

    This condition is TRUE when *string1* is lexically the same as *string2*. If multiple words (separated by spaces, tabs, or other separators) are needed, then you should place the strings in quotes, as follows:

    "*string1*" == "*string2*"

Examples:

The following command prints "The file exists" if the file TEST.DAT exists:

```
C> IF EXIST TEST.DAT ECHO The file exists
The file exists
C>
```

The following command tests if the previous DISKCOPY command was successful or not:

```
C> IF ERRORLEVEL 0 ECHO The copy was successful
The copy was successful
C>
```

This command compares the first batch file argument in a batch file with a switch "/P" and if specified, transfers to the EXIT label in a batch file:

```
IF "%1"=="/P" GOTO EXIT
```

INTERSVR Command

Function: Enables disk, and optionally console, redirection through a serial communications line to a target computer. This allows the target computer access to a selected drive of the host computer, and optionally routes all keyboard and screen activity from the target computer's keyboard and screen to the host computer's keyboard and screen.

Type: External

Syntax: INTERSVR [/X=*d*:] [/BAUD=*baud*] [/COM=*port*] [/WRITE]

Parameters:

d:	Drive to allow target to access.
baud	115200, 57600, 38400, 19200, or 9600.
port	1 for COM1, or 2 for COM2.
/WRITE	Specifies write access to drive is permitted.

Description:

The INTERSVR utility enables full-duplex, multiplexed disk and console I/O requests captured by the SERDRIVE.SYS device driver on a target computer to be serviced on a host computer. There are two benefits of this linkage:

1. The target has access to the host's hard drive.
2. The host can be used as the target's console.

Once INTERSVR.EXE is loaded, it waits for requests to be routed to it by the target computer. At any time, the host user may press the CTL and ALT keys down together for a couple seconds to terminate the communications link.

INTERSVR.EXE provides read-only access to the host's drive by default. If write access is desired, add the /WRITE option to INTERSVR's command line.

By default, communications are established at 57K baud, which is half the possible speed of the link. With proper cabling and matched UARTs on the host and target, it is possible to use the link at 115,200 baud, but this does not work under all circumstances. If you find that your cabling or host/target combination does not support communications at 57K baud, you may wish to try communications at 9600 baud first, and work up until the highest baud rate is determined.

While the host sets the baud rate, the target's SERDRIVE.SYS tries all of the baud rates at the time it connects to the host, from highest to lowest. If you switch baud rates on the host, it will be necessary to reboot the target so that it can find the new baud rate you have selected.

Example:

The following example establishes a link with a target on COM1, shares drive C:, and makes it writable:

```
C> INTERSVR  /X=C: /PORT=1  /WRITE
```

LABEL Command

Function: Creates or changes a volume label on a diskette or hard disk partition.

Type: External

Syntax: LABEL [*d:*] [*label*]

Parameters:

d: Drive or partition to assign a label to.
label   New volume label to assign to the drive or partition.

Description:

The LABEL utility program deletes any existing label associated with the specified drive and optionally installs a new label.

Volume labels can be from one to 11 characters in length, and cannot contain any of the following special characters:

```
* ? / \ | . , ; : + = < > [ ] ( ) & ^
```

If no label is specified, then the user is asked for the label. If no label is given, then the current label is simply removed from the diskette or partition, and a new one is not added. If a label is given, then the specified label replaces the old one.

If a drive is specified, then that drive is affected. If no drive is specified, then the default drive is affected.

Example:

The following example labels drive C with the label, MYDISK:

```
C> LABEL C: MYDISK
C>
```

LOADHI Command

Function: Runs a program in upper memory.

Type: Internal

Syntax: LOADHI [*filename*]

Parameters:

filename   Name of program to run in upper memory.

Description:

The LOADHI command causes the specified program to be run in upper memory, so that in the event the program is a TSR, it can occupy memory away from the 640 KB main memory area. This leaves more contiguous memory for other applications.

There is no advantage to running a program in upper memory that does not terminate and stay resident in memory..

Example:

The following example runs SMARTDRV in an upper memory block (note that an extended memory manager must be running in order for a UMB to exist):

```
C> LOADHI SMARTDRV
General Software SMARTDRV Disk Cache Version 1.0

C>
```

MEM Command

Function: Displays memory usage statistics.

Type: Internal

Syntax: MEM [/C] [/F] [/S] [/P]

Parameters:

/C	Classifies programs by memory usage.
/F	Displays information about free memory.
/S	Displays system kernel memory pool.
/P	Pauses after each screenful of information.

Description:

The MEM command displays statistics about the quantity and usage of RAM in the running system, including low memory, extended memory, and operating system memory pool.

Examples:

The following example shows a basic MEM display:

```
C> MEM
Memory Type        Total  =   Used  +   Free
----------------   -------    -------    -------
Operating System     57K        12K        45K
Conventional        583K       108K       475K
Upper                 0K         0K         0K
Extended (XMS)      912K         0K        912K
INT 15h Extended      0K         0K         0K
----------------   -------    -------    -------
Total memory        1552K       120K       1432K

Total Under 1MB      640K       120K        520K

Largest executable program size     452K
Largest free upper memory block       0K

The high memory area is available for use.
Largest free XMS block size = 912K.
```

```
    Number of XMS handles available = 40927.

    C>
```

MKDIR Command

Function: Makes a subdirectory.

Type: Internal

Syntax: MKDIR [*d:*]*path*

Parameters:

> *d:*   Drive to create the directory on.
> *path*   Pathname of the directory to be created.

Description:

> The MKDIR command (abbreviated MD) creates a subdirectory of a root directory or a
> subdirectory. By using the MKDIR command, tree-structured file systems can be created
> that can effectively manage thousands of files by grouping them hierarchically.

> If a drive is specified, then the directory is created on the specified drive. Otherwise, it is
> created on the default drive.

> A pathname of the directory to be created must be given. The name must be unique and
> cannot name an existing file or subdirectory, as duplicate files are normally not supported
> by file systems.

Examples:

> The following example creates a subdirectory of the current directory on the default drive
> called JOE:

```
C> MKDIR JOE
C>
```

> The following command creates a directory called FINANCE in the root directory of drive
> A:

```
C> MKDIR A:\FINANCE
C>
```

MODE Command

Function: Sets or displays operational modes for devices.

Type: External

Syntax: MODE LPT*n*:[*pagewidth*][,[*linesperinch*][,P]]
                     MODE COM*x*:*baudrate*,[*parity*],[*databits*],[*stopbits*],[P | -]
                     MODELPT*n*:=COM*x*:
                     MODE *display*

Parameters:

n	Parallel port number (1-3).
x	Asynchronous communications port number (1-4).
pagewidth	Columns per printed line.
linesperinch	Lines per vertical inch.
baudrate	Transmission baudrate for async devices.
parity	Parity type (even, odd, or none).
databits	Data bits per transmitted character.
stopbits	Stop bits per transmitted character.
P	Infinite retry should be enforced.
-	Infinite retry should be terminated.
display	Display mode type to change to.

Description:

The MODE utility program displays the status of devices in the system, or changes their mode of operation. Supported devices are the parallel printers, the asynchronous communications ports, and the displays. If no parameters are specified, then the status of the devices in the system is displayed.

The first form listed above changes the settings for the default pagewidth and number of lines that can be printed in one inch (vertically) associated with a parallel printer. This allows listings to be wrapped correctly on smaller paper. The parallel printers are named LPT1, LPT2, LPT3, and LPT4.

The second form listed above changes the settings for a communications port. The ports are named COM1, COM2, COM3, and COM4. The system initializes each COM port to 2400 baud, even parity, 7 data bits, and 1 stop bit.

The supported baud rates are: 110, 150, 300, 600, 1200, 2400, 4800, 9600, and 19,200.

The supported parity types are: E (even), O (odd), and N (none).

The supported stop bits are 1 or 2. For most communications applications, if the baud rate is 110, then 2 should be selected, otherwise 1 should be used.

If P is specified, then MODE uses infinite retry on the parallel or serial device, so that I/O doesn't time-out due to temporary device problems, such as out-of-paper.

The third form re-routes I/O destined to the specified parallel printer through an asynchronous COM port. This allows printers attached to serial ports to be used transparently.

The fourth form changes the display mode, as follows:

- MONO   Switches to monochrome (TTL) display.
- BW40   Switches to black-and-white, 40 column display.
- BW80   Switches to black-and-white, 80-column display.
- CO40   Switches to color, 40-column display.
- CO80   Switches to color, 80-column display.
- 40   Switches to 40-column display.
- 80   Switches to 80-column display.

Examples:

The following example shows how a dual-monitor system can switch back-and-forth between two monitors:

```
C> MODE MONO                 (switch to monochrome screen)
C> MODE CO80                 (switch to color screen)
```

The following example re-routes LPT1 through the second COM port:

```
C> MODE LPT1:=COM2:
C>
```

The following example disables any rerouting of LPT1:

```
C> MODE LPT1:
C>
```

The following example sets up the second communications port for 9600 baud, no parity, 1 stop bit, and 8 data bits. Infinite retry is installed:

```
C> MODE COM2:9600,N,8,1,P
C>
```

MORE Command

Function: Displays output one screenful at a time.

Type: External

Syntax: MORE < *filepath*
progpath | MORE

Parameters:

filepath   Pathname of a file to be paged through.
progpath   Name of a program to run.

Description:

The MORE utility program is a filter that can be used by itself or in a pipeline (see examples above) to view a file or the output of a program one screenful at a time. After

each screenful, MORE pauses and waits for the user to press a key, after which the next screenful of information is displayed.

Examples:

The following examples show how the contents of a file may be typed out, pausing between each screenful of 24 lines. Either command will achieve the same effect:

```
C> TYPE MYFILE.DAT | MORE

C> MORE < MYFILE.DAT
```

The following example shows how you can combine DIR and MORE to achieve the same effect as the DIR/P command:

```
C> DIR | MORE
```

The following example shows how a fictitious user report generator program might be run, piping the results into the MORE filter.

```
C> MYREPORT | MORE
```

PATH Command

Function: Displays or sets the current path.

Type: Internal

Syntax: PATH [*pathlist*]

Parameters:

> *pathlist*   List of directory paths to search.

Description:

The PATH command displays or changes the current search path that is used by the command processor, COMMAND.COM, to locate user programs and batch files.

If no pathlist parameter is specified, then the current path is displayed. If a pathlist parameter is specified, then the path will be changed to the one specified.

The current path is actually stored as an environment variable, in the form:

```
PATH=path1;path2;path3;path4;...;pathn
```

where *path1* is the first directory to search for programs and batch files in, *path2* is the next directory, and so on. When COMMAND.COM tries to run a program or batch file, it first looks in the current directory of the default drive, and then looks through all of the

named directories in the PATH environment variable until the program or batch file is located.

You may also change your PATH variable with the SET command, documented later in this chapter. Both commands have the same effect, although using the PATH command without operands to display the current path is easier than scanning through the output from the SET command to accomplish the same thing.

Examples:

The following example shows the current path:

```
C> PATH
PATH=C:\DOS;D:\UTILS
C>
```

The following example sets the current path to first look in the DOS directory of drive C, and then in the PROGRAMS directory on the default drive:

```
C> PATH C:\DOS;\PROGRAMS
C>
```

PAUSE Command

Function: Pauses to allow for user intervention.

Type: Internal

Syntax: PAUSE [*message*]

Parameters:

message An optional message to be displayed before pausing.

Description:

The PAUSE command is typically used in batch files to suspend execution of the batch file, print a message on the screen, and wait for the user to press a key after some action has been performed. This is useful in instances where, for example, diskettes must be changed before continuing.

PAUSE displays the following message on the screen before accepting a keypress from the user:

```
Strike any key when ready . . .
```

Example:

The following example prints a message and waits for the user to press a key before continuing:

```
C> PAUSE Insert your disk in drive A now.
Insert your disk in drive A now.
Strike any key when ready . . . _
C>
```

PROMPT Command

Function: Changes the current command shell prompt.

Type: Internal

Syntax: PROMPT [*string*]

Parameters:

     string   Prompt string to be used by COMMAND.COM.

Description:

The PROMPT command maintains the PROMPT environment variable that is used by COMMAND.COM to display something before the user is asked to type-in a command. The prompt string contains metacharacters that form a simple language, shown in a table below.

The default PROMPT variable is ng. This has the effect of showing the current drive letter followed by a "greater-than" sign:

```
A>
```

If no string is specified, then the prompt is reset to the default prompt, above. Otherwise, the prompt string is changed for the next command prompting.

The metacharacters supported by Embedded DOS-ROM are the same ones that are supported by MS-DOS. They are shown in Table 3.1:

Metacharacter	Displayed as . . .
$$	Dollar sign
$t	System time
$d	System date
$p	Current path of default drive
$v	MS-DOS compatibility version number
$n	Default drive letter without the colon
$g	Greater-than sign (>)
$l	Less-than sign (<)
$b	Vertical bar (	)
$q	Equal sign (=)
$e	Escape character (0x1b)
$_	Carriage-return/Line-feed pair

Table 7.1. Metacharacters supported by PROMPT.

Example:

The following example changes the prompt to display the date and time on one line, and then the current directory on the following line:

```
C> PROMPT $d $t $_$p $g
```

REM Command

Function: Provides a remark in a batch file.

Type: Internal

Syntax: REM [*remark*]

Parameters:

remark Any optional comment in any format whatsoever.

Description:

The REM command provides a simple way of entering a free-form comment in a batch file, that has no side effects. The REM command may be used interactively, but has no effect.

Example:

```
C> REM this is a remark in a batch file!
C>
```

RENAME Command

Function: Renames a file or group of files.

Type: Internal

Syntax: REN[AME] [*d:*]*path newpath*

Parameters:

d: Drive on which files are to be renamed.
path Wildcarded pathname of file(s) to be renamed.
newpath Wildcarded new pathname of files.

Description:

The RENAME command (abbreviated REN) renames a file or group of files. Files cannot be moved in the directory structure with this command; instead, only their filenames are altered within the directory in which they reside.

Wildcards may be used in the second pathname to indicate that the characters in that component of the first filename are to be kept as-is. For example, to rename the file JANUARY.RPT to JANUARY.SAV, you could use the following command to avoid typing JANUARY twice:

```
C> RENAME JANUARY.RPT *.SAV
C>
```

Example:

The following example renames all of the files with .WKS extensions in the current directory of the default drive to have .BAK extensions instead:

```
C> REN *.WKS *.BAK
C>
```

RMDIR Command

Function: Removes a subdirectory.

Type: Internal

Syntax: RMDIR [*d:*]*path*

Parameters:

d:	Drive to remove the directory from.
path	Pathname of the directory to be removed.

Description:

The RMDIR command (abbreviated RD) removes a subdirectory of a root directory or of a subdirectory. This command can only be used to delete directories, and cannot be used to delete files, even if they are inside the directory to be removed. Conversely, the DEL command cannot delete directories; only the files they contain.

If a drive is specified, then the directory on the specified drive is removed. Otherwise, the default drive is assumed. A pathname of the directory to be removed must be given.

Examples:

The following example removes a subdirectory of the current directory on the default drive called JOE:

```
C> RMDIR JOE
C>
```

The following command removes a directory called FINANCE in the root directory of drive A:

```
C> RMDIR A:\FINANCE
C>
```

SET Command

Function: Displays or changes the environment strings.

Type: Internal

Syntax: SET [*variable*=[*string*]]

Parameters:

> *variable*   Variable name to change the value of.
> *string*   String to be assigned to the variable.

Description:

> The SET command displays the entire environment space (one variable per line), or changes the assignment of one variable in the environment space.
>
> If no operands are specified, then the SET command simply displays all of the environment variables in the environment space.
>
> If a variable name and an equal sign is given, but no string is specified, then the variable name is removed from the environment space. If the string is specified, then the previous definition of the variable is deleted, and the new one is installed in the environment.
>
> Common commercial software packages use the environment space to hold their configuration parameters. Batch files may also access the environment variable assignments with the following syntax:
>
> %*variable*%
>
> The use of percent signs around a text string tells COMMAND.COM to substitute the actual string assignment of the variable name with the variable in percents.
>
> When a program is run, the current environment is cloned and the program runs with its own copy of the environment space. Thus, if it changes any variable values in its copy of the environment space, it will not affect the environment space of the command interpreter when the program terminates.
>
> The size of the environment space may be specified when invoking the COMMAND.COM command processor. See the COMMAND utility program documentation in this chapter for details.

Examples:

> The following command sets the PROMPT variable to a new string that displays the current directory and a '>' sign:

```
C> SET PROMPT=$p$g
C:\PROGRAMS>_
```

SHARE Command

Function: Does nothing in EDOS-ROM. Added for record locking support for MS-DOS.

Type: External

Syntax: SHARE

Parameters:

> none.

Description:

> It serves as a placeholder for the SHARE command, required in MS-DOS systems to add
> support for record locking, as it is not built-in to the base kernel of MS-DOS. All Starlite
> systems have file sharing and record locking built-in, so this command is provided solely
> to make installation batch files work if they test for the existence of SHARE.

SHIFT Command

Function: Allows access to multiple batch file arguments.

Type: Internal

Syntax: SHIFT

Parameters:

> none.

Description:

> The SHIFT command shifts the contents of the 9 batch file arguments so that %2 is
> copied into %1, %3 is copied into %2, and so on. The %9 variable is reset to the empty
> string.

SORT Command

Function: Sorts lines of text.

Type: External

Syntax: SORT [/R] [/+*n*]

Parameters:

/R	Input is to be sorted in reverse order.
/+*n*	Starting column of field to sort on.

Description:

The SORT utility program is a filter that reads lines of text from standard input, sorts them, and writes the sorted file to standard output. Piping or redirection is needed to make use of this utility.

If the /R option is specified, then a descending rather than ascending sort is performed.

If the /+n option is specified, then the lines will be sorted on the field starting at column 'n' and ending at the last column, of each line read from standard input. By default, the file is sorted on column 1.

The ability of SORT to handle large files is dependent on the amount of conventional memory available at the time SORT runs.

Examples:

The following example sorts a directory listing:

```
C> DIR | SORT
```

The following command sorts a report by column 27:

```
C> SORT /+27 < JANUARY.RPT
```

SWITCH Command

Function: Displays or changes the option switch character.

Type: Internal

Syntax: SWITCH [*character*]

Parameters:

character   Switch character to be used.

Description:

The SWITCH command allows the user to display or change the option switch character. The default value for this character is the forward slash (/), and is the one documented as the switch separator in all of the commands in this chapter.

For compatibility with other operating systems, this character can be changed to some other character (for example, a minus sign). Because all of the utility programs use the operating system's switch character rather than a hard-coded slash character, they will all function with whatever switch character the user selects.

If no character is specified, then the current switch character is displayed.

Examples:

The following example displays the current switch character:

```
C> SWITCH
SWITCH is '/'.
C>
```

The following command changes the switch character to a minus sign:

```
C> SWITCH '-'
C>
```

SYNCH Command

Function: Flushes all system file buffers for orderly shutdown.

Type: Internal

Syntax: SYNCH

Parameters:

none.

Description:

The SYNCH command provides a synchronization checkpoint feature that enables a batch file to flush the file system's buffers to disk before doing something that would otherwise cause a disorderly shutdown (such as executing a PARK program that parks the disk heads, or physically turning it off.)

After the SYNCH command executes, all buffers in the file system are guaranteed to be flushed until the next write, caused by opening, closing, writing, or otherwise affecting any file name space in the system.

SYNCH can only be used to synchronize local file systems, and may or may not flush buffers on other workstations in a distributed file system.

Examples:

The following example flushes the file system so that the workstation can be turned off:

```
C> SYNCH
C> (the workstation can now be turned off)
```

SYS Command

Function: Copies the system files to the destination disk.

Type: External

Syntax: SYS *d:*

Parameters:

  d: Specifies the drive letter to transfer the system to.

Description:

  The SYS command is used in systems that use the disk-loadable version of Embedded
  DOS-ROM to copy the system files to a disk so that it becomes bootable media.

  The system files include DOS.SYS (the operating system kernel), COMMAND.COM (the
  command interpreter), and the PBR (partition boot record, or first sector on the disk.)

Examples:

  The following example makes the hard drive C: bootable:

```
C> SYS C:
Copying DOS.SYS...
Copying COMMAND.COM...
System files transferred.
```

TIME Command

Function: Displays or changes the system time.

Type: Internal

Syntax: TIME [*hh:mm:ss*]

Parameters:

  hh:mm:ss   System time in hours, minutes, and seconds.

Description:

  The TIME command displays or changes the system time. If no parameter is specified,
  then the current system time is displayed, and the user is queried for the new system time.
  If the user just presses the ENTER key, the system time is not changed. If the user enters
  a new time, then the system's real-time-clock is updated.

If a parameter is specified, then it must be in the form: hh:mm:ss. It is acceptable to omit the seconds. The hours, minutes, and seconds are checked by TIME to ensure that they are reasonably correct. Thus, it is invalid to enter the time: 99:88:77, since hours cannot be larger than 23, minutes cannot be greater than 59, and seconds cannot exceed 59.

Example:

The following example displays the current time and changes it interactively:

```
A> TIME
Current time is  2:12:14.16
Enter new time: 13:12:14
A>
```

TREE Command

Function: Displays directory structure of drive.

Type: External

Syntax: TREE [*d:*] [/F]

Parameters:

d:	Drive to display tree for.
/F	Displays filenames as well as directory names.

Description:

The TREE utility program displays the directory structure of a drive by recursively descending into each subdirectory, until all of the directories are listed.

If the /F option is specified, then filenames of files in the directories are displayed as they are traversed.

If no drive letter is specified, then the default drive is assumed; otherwise, the specified drive is scanned.

Example:

The following example displays the directory structure for drive D:

```
A> TREE D:
```

TYPE Command

Function: Displays the contents of a file.

Type: Internal

Syntax: TYPE [*d:*]*path*

Parameters:

> *d:*   Drive that the file resides on.
> path   Pathname of the file to display.

Description:

> The TYPE command copies the contents of the specified file to standard output (usually, the screen). If the drive letter is not specified, then the default drive is assumed.

Example:

> The following example displays the contents of the CONFIG.SYS file on drive C:

```
A> TYPE C:\CONFIG.SYS
FILES=20
BUFFERS=20
A>
```

VER Command

Function: Displays the version information about the operating system.

Type: Internal

Syntax: VER

Parameters:

> none.

Description:

> The VER command displays version of the Embedded DOS-ROM operating system that is running.

VERIFY Command

Function: Displays or changes the status of the VERIFY flag.

Type: Internal

Syntax: VERIFY [ON | OFF]

Parameters:

> ON   Disables lazy-writing in the file system & disk drivers.

OFF   Enables lazy-writing in the file system & disk drivers.

Description:

The VERIFY command changes or displays how Embedded DOS-ROM handles I/O to disk files and directory structures. If VERIFY is ON, then Embedded DOS-ROM verifies immediately that disk I/O is completed successfully before telling the user that it was. This is accomplished by writing data directly to disk, without temporarily storing it in a file system or disk driver cache.

If VERIFY is OFF, then Embedded DOS-ROM caches writes to files and defers the actual writing to disk, enabling multiple writes to the same sectors to be served much faster. The cache is automatically written to disk in the background during "dead time", when the disk is not busy. This is accomplished with the multitasking threads and semaphores that the Embedded DOS-ROM kernel supports.

Examples:

If no parameters are specified on the VERIFY command, then the status of the VERIFY flag is displayed. For example:

```
C> VERIFY
VERIFY is ON.
C>
```

If ON or OFF is specified, then the VERIFY flag is set to that value. For example:

```
C> VERIFY OFF
C> VERIFY
VERIFY is OFF.
C>
```

VOL Command

Function: Displays the volume label on a drive.

Type: Internal

Syntax: VOL [*d:*]

Parameters:

d: Drive or partition to display the volume label for.

Description:

The VOL command displays the volume label of a diskette or a hard disk partition, much like the external LABEL utility program allows. VOL does not allow the user to change the volume label.

If the drive letter is not specified, then the default drive is assumed.

XCOPY Command

Function: Copies groups of files and/or subdirectories.

Type: External

Syntax: XCOPY *source* [*destination*] [/A | /M] [/P] [/S] [/E] [/V] [/W]

Parameters:

> *source* Specifies the file(s) to be copied
> *destination* Specifies the location of the copies
> /A Copies files with the archive bit set without changing the bit
> /M Copies files with the archive bit set, clearing while copying
> /P Prompts before copying each file
> /S Copies subdirectories except empty ones
> /E Copies subdirectories including empty ones
> /V Verifies each new file
> /W Prompts for a key press before copying the files

Description:

> The XCOPY command provides enhanced functionality over the COPY command supported by COMMAND.COM. It allows the copying of whole directory trees of files, and makes use of XMS memory (provided by HIMEM.SYS) to buffer the files, thereby optimizing performance.

Examples:

> The following example copies the entire contents of a diskette in drive A: to a directory called DISK on drive C:

```
C> XCOPY  A:*.*  C: /E
```

INDEX