

***4DOS***®

***4NT***™

***Take Command***®

# Reference Manual

**Published By**

JP Software Inc.

P.O. Box 328

Chestertown, MD 21620

U.S.A.

(410) 810-8818

Fax (410) 810-0026

## ACKNOWLEDGMENTS

We couldn't produce products like 4DOS, 4NT, and Take Command without the dedication and quality work of many people. Our thanks to:

**JP Software Staff:** Mike Bessy, Celeste Conn.

**On-line Support:** Blueberry Hill Communications (Palm Desert, CA) and Media3 Technologies (Pembroke, MA).

**Beta Testers:** We can't list all of our beta testers here! A special thanks to all of you who helped make 4DOS, 4NT, and Take Command elegant, reliable, and friendly.

Copyright © 2003, JP Software Inc., All Rights Reserved. 4DOS® and 4NT are JP Software Inc.'s trademarks for its family of character-mode command processors. 4DOS® and Take Command® are registered trademarks and JP Software, jpsoft.com, and all JP Software designs and logos are trademarks of JP Software Inc. Other product and company names are trademarks of their respective owners.

# CONTENTS

Introduction.....	1
How to Use This Manual.....	2
Registration and Upgrade Information .....	6
Technical Support.....	6
Chapter 1 / General Concepts .....	7
Operating Systems and Command Processors.....	7
Primary and Secondary Shells .....	9
Files and Directories.....	10
Drives and Volumes .....	10
File Systems.....	11
Network File Systems .....	12
Directories and Subdirectories .....	13
File Attributes and Time Stamps.....	16
NTFS File Streams.....	18
Internal and External Commands.....	18
Executable Files and File Searches .....	19
The Environment.....	23
Character Sets, ASCII, and Key Codes .....	24
The Keyboard.....	26
Video.....	28
ANSI Drivers .....	28
Chapter 2 / Conventions .....	30
Colors and Color Names.....	30
Blinking Text and Bright Background Colors .....	32
Keys and Key Names.....	34
Popup Windows.....	35
Chapter 3 / The Take Command Interface .....	37
The Take Command Window .....	37
Resizing the Take Command Window.....	39
Using the Scrollback Buffer .....	40
Highlighting and Copying Text .....	41
Using Drag and Drop .....	42
Take Command Menus.....	42
File Menu.....	43
Edit Menu .....	44
Apps Menu.....	44
Options or Setup Menu .....	45
Utilities Menu.....	46

Help Menu .....	47
Take Command Dialogs .....	47
Save To File Dialog.....	48
Print Dialog .....	48
Printer Setup Dialog .....	48
Run Program Dialog.....	48
Configuration Dialog.....	50
Tool Bar Dialog.....	51
Find Files / Text Dialog.....	52
Descriptions Dialog .....	54
Alias and Environment Dialogs.....	55
Take Command Help .....	55
Chapter 4 / Using 4DOS, 4NT, and Take Command .....	57
At the Command Line .....	57
Command-Line Editing.....	58
Command History and Recall.....	61
Command History Window .....	64
Local and Global Command History .....	65
Command Names and Parameters.....	66
Filename Completion .....	67
Filename Completion Window.....	72
Variable Name Completion.....	72
Automatic Directory Changes.....	72
Directory History Window .....	74
Multiple Commands .....	76
Expanding and Disabling Aliases.....	76
Command Line Help .....	77
Command-Line Length Limits .....	78
Starting Applications.....	78
Page and File Prompts .....	80
Directory Navigation .....	81
Extended Directory Searches.....	83
CDPATH .....	86
Input and Output.....	87
Redirection.....	88
Piping.....	91
Keystack.....	92
File Selection.....	93
Extended Parent Directory Names.....	94
Wildcards .....	94
Date, Time, and Size Ranges .....	97
File Exclusion Ranges .....	102

Multiple Filenames .....	103
Include Lists .....	104
LFN File Searches .....	105
Executable Extensions .....	106
@File Lists.....	109
Command Switches for File Selection .....	110
Windows File Associations.....	112
Using Internet URLs.....	112
Using FTP Servers .....	113
Using HTTP Servers .....	114
Waiting for Applications to Finish.....	114
Critical Errors.....	115
Advanced Features .....	116
Conditional Commands.....	116
Command Grouping .....	117
Escape Character .....	119
Scrolling and History Keystrokes.....	120
Date Handling.....	121
Chapter 5 / Aliases and Batch Files.....	122
Aliases .....	122
Batch Files .....	125
.BAT, .CMD, and .BTM Files.....	125
Echoing.....	127
Batch File Parameters .....	127
Using Environment Variables .....	128
Batch File Commands .....	129
Interrupting a Batch File.....	131
Automatic Batch Files.....	132
Detecting 4DOS, 4NT, and Take Command .....	134
Using Aliases in Batch Files.....	134
Debugging Batch Files .....	137
String Processing.....	140
Line Continuation .....	143
Batch File Compression .....	143
REXX Support .....	145
EXTPROC Support.....	147
DDE Support .....	147
Using the Environment .....	148
Configuration Variables.....	150
Internal Variables .....	153
Variable Functions .....	165
Special Character Compatibility.....	195

Command Parsing .....	197
Argument Quoting.....	199
Chapter 6 / Configuration.....	203
Modifying the .INI File.....	203
Using the .INI File.....	204
.INI File Sections .....	205
.INI File Directives .....	206
Types of Directives .....	208
Initialization Directives .....	209
Configuration Directives .....	217
Color Directives .....	228
Key Mapping Directives.....	230
Advanced Directives.....	235
Examples.....	238
Chapter 7 / Commands .....	241
Command Categories.....	241
How to Use the Command Descriptions.....	243
?.....	247
ACTIVATE.....	248
ALIAS.....	250
ASSOC.....	261
ATTRIB .....	263
BEEP .....	267
BREAK.....	268
CALL .....	269
CANCEL.....	270
CD / CHDIR .....	271
CDD .....	274
CHCP.....	277
CLS .....	278
COLOR.....	279
COPY .....	280
CTTY .....	289
DATE.....	290
DDEEXEC.....	291
DEL / ERASE.....	292
DELAY .....	298
DESCRIBE.....	299
DETACH .....	302
DIR .....	303
DIRHISTORY .....	319
DIRS .....	321

DO.....	322
DRAWBOX.....	326
DRAWHLIN.....	328
DRAWVLIN.....	329
ECHO and ECHOERR .....	330
ECHOS and ECHOSERR.....	332
ENDLOCAL .....	333
ESET .....	334
EVENTLOG .....	336
EXCEPT .....	337
EXIT .....	340
FFIND .....	341
FOR .....	347
FREE .....	357
FTYPE.....	358
FUNCTION.....	360
GLOBAL.....	361
GOSUB.....	363
GOTO .....	365
HEAD .....	367
HELP .....	369
HISTORY .....	370
IF .....	372
IFTP.....	382
INKEY .....	385
INPUT .....	388
KEYBD .....	390
KEYS .....	391
KEYSTACK.....	392
LFNFOR.....	397
LH / LOADHIGH .....	398
LIST.....	401
LOADBTM .....	408
LOCK.....	409
LOG .....	410
MD / MKDIR .....	412
MEMORY .....	414
MKLNK .....	415
MOVE.....	416
MSGBOX.....	423
ON.....	425
OPTION .....	427
PATH.....	429

PAUSE .....	432
PLAYAVI.....	433
PLAYSOUND.....	434
POPD.....	435
PRINT .....	436
PROMPT .....	437
PUSHD.....	441
QUERYBOX.....	443
QUIT.....	444
RD / RMDIR .....	445
REBOOT .....	447
RECYCLE .....	449
REM.....	450
REN / RENAME .....	451
RETURN .....	455
SCREEN.....	456
SCRPUT .....	458
SELECT .....	460
SENDMAIL.....	467
SET.....	468
SETDOS .....	472
SETLOCAL .....	480
SHIFT.....	481
SHORTCUT .....	483
SHRALIAS .....	485
SNPP .....	487
START .....	488
SWAPPING.....	493
SWITCH.....	494
TAIL .....	497
TASKEND.....	499
TASKLIST.....	500
TCTOOLBAR .....	501
TEE.....	503
TEXT .....	504
TIME .....	505
TIMER.....	506
TITLE.....	508
TOUCH.....	509
TREE .....	512
TRUENAME .....	514
TYPE .....	515
UNALIAS .....	517



UNFUNCTION .....	519
UNLOCK .....	521
UNSET .....	522
VER .....	524
VERIFY .....	525
VOL .....	526
VSCRPUT .....	527
WHICH.....	529
WINDOW .....	530
Y.....	531
Appendix A / Error Messages .....	533
Index .....	543



# INTRODUCTION

Welcome to our Reference Manual. We have designed this manual to accompany all three of our products: 4DOS (for DOS and Windows 95, Windows 98, and Windows Millenium or “ME”), 4NT (for Windows NT, Windows 2000, Windows XP, Windows 98, and Windows ME), and Take Command (for Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, and Windows ME).

Each of these programs is a command interpreter or “shell.” That means that they respond to commands you type at the prompt. 4DOS and 4NT are designed to replace the command interpreter that was supplied with your operating system:

4DOS replaces *COMMAND.COM* from DOS or from Windows 95, Windows 98, and Windows ME.

4NT replaces *CMD.EXE* from Windows NT / 2000 / XP, and may also be used under Windows 98 and Windows ME.

Each of our products is highly compatible with the command interpreter that it replaces. That means that you don't have to change your computing habits or unlearn anything to use any of these products. Each adds many new features and commands to its operating environment, making the operating system friendlier, easier to use, and much more powerful and versatile, without requiring you to learn a new program or a new style of work.

Each product is shipped with a smaller *Introduction and Installation Guide*. Please start with the smaller manual. It will tell you how to get started with the product you purchased and also contains some other information that applies specifically to that product. The *Introduction and Installation Guide* also contains details on how to contact JP Software for technical support, customer service, or other assistance.

Once you have installed 4DOS, 4NT, or Take Command, you can learn its new features at your own pace. Relax, enjoy the power of your new program, and browse through the manual occasionally. Press the **F1** key whenever you need help. The program will soon become an essential part of your computer, and you'll wonder how you ever got along without it.

We are constantly working to improve these programs. If you have suggestions for features or commands we should include in the next version, or any other way we could improve our product, please let us know (see your *Introduction and Installation Guide* for contact information). Many of the features in this version

were suggested by our users. We can't promise to include every suggestion, but we really do appreciate and pay attention to your comments.

### ***How to Use This Manual***

We have combined the reference information for 4DOS, 4NT, and Take Command in one manual because they offer very similar features and commands, so individual manuals would be almost identical.

Most of this manual describes commands and features which are available in all three products. When we need to discuss the features or behavior of a single product, we mention it specifically in the text. When an entire paragraph or section applies to a specific product, we use marginal text to identify that product:

- 4DOS** marks sections that apply only to 4DOS.
- 4NT** marks sections that apply only to 4NT.
- TC** marks sections that apply only to Take Command.

Occasionally, a feature will be marked with two of these marginal notations when it applies to two different products.

When we use the term **Windows** in this manual, it refers to Windows 95, Windows 98, Windows ME ("ME"), Windows NT, Windows 2000, Windows XP, and their successors, unless another meaning is obvious from the context. When we need to refer to a feature or concept in one of these operating systems specifically, we name it explicitly.

We have designed this manual to serve as an introduction to our products for novice users and as a reference manual for advanced users. You will likely find some parts of the manual too simple or too technical for your tastes. Unless you are convinced that one of those sections holds just the information you need for a specific task, feel free to skip to the next part of the manual that is more to your liking. You can use almost every feature of our products without having to worry about other features or commands.

Many of the examples in this manual show how to enter a command at the prompt. The examples use the default 4NT and Take Command prompt, which appears as [c:\] (in 4DOS, the prompt appears as c:\>). The prompt you see will also vary if you create a custom prompt with the PROMPT command.

You will see many references in this manual to “option dialogs.” These dialogs are part of 4NT and Take Command, and are used to control configuration options. In 4DOS the same controls are available, but there are no “option dialogs.” Instead, configuration options are controlled through the screens provided by the OPTION command, or by making entries in the *4DOS.INI* file. Additional details on configuration are in Chapter 6.

- ! As you read the manual, you will occasionally see an exclamation point [!] next to a paragraph. The exclamation point means that that paragraph contains a caution or warning you should observe when using the feature it discusses.

This manual is divided into seven chapters, an appendix, and an index. Here's a brief overview of what you'll find in each:

### ***Chapter 1 / General Concepts***

This chapter is an introduction to several terms and concepts that we use throughout the manual. If you're a novice, you might want to browse through the entire chapter. If you're a power user and all of the topics in this chapter seem simple, then go on to the next chapter. If you think you need to brush up a bit on the basics of a couple of terms or ideas, you'll probably find them here.

### ***Chapter 2 / Conventions***

This chapter contains information about three conventions that are used throughout 4DOS, 4NT, and Take Command: colors and color names, keys and key names, and popup windows. You'll find many references in the manual to the information in this chapter. You may choose to read through this chapter to see what is available, or refer to the topics here when you come to a cross reference later in the manual.

### ***Chapter 3 / The Take Command Interface***

This chapter covers the Take Command screen and the associated menus, dialogs, and other features related to working with Take Command as a Windows program. It will help you understand how Take Command functions simultaneously as both a command line product, and a GUI utility; for that reason it's important reading for those moving to Take Command from a character-mode command processor like 4DOS or *COMMAND.COM*. Some parts cover more advanced dialogs and other

features that you won't need as you get started, but everyone using Take Command should look through the basic screen elements and behaviors documented in the earlier parts of the chapter.

### ***Chapter 4 / Using 4DOS, 4NT, and Take Command***

This chapter is for everyone. It contains a description of 4DOS, 4NT, and Take Command command line features, plus lots of examples to help you learn to use each one. Even if you are a novice user and want to ignore some of these features until later, skim through this chapter to get an idea of what is available and where to find information you may eventually want. Our products offer both features which are not related to specific commands, and a complete set of over 120 internal commands. This chapter has reference information on most of the command line features. Reference information on commands is in Chapter 7.

### ***Chapter 5 / Aliases and Batch Files***

This chapter introduces two of the most powerful features in 4DOS, 4NT, and Take Command: Aliases and Batch Files. You can use both to automate much of your computing work. Even if you are a novice user, you should skim through this chapter to see what is available. This chapter also contains information about environment variables, and the internal variables and variable functions that make aliases and batch files extremely powerful and flexible.

### ***Chapter 6 / Configuration***

This chapter is for anyone who wants to personalize 4DOS, 4NT, or Take Command, and for advanced users who want to be sure that the command processor is running at peak efficiency on their systems. It includes detailed information on setting up your command processor and on changing its configuration.

### ***Chapter 7 / Command Reference Guide***

Our products offer over 120 internal commands. This chapter explains the purpose of each command and tells you how to use it. It has examples that will help you learn each command, and the technical details you will need to get the command to behave exactly as you wish.

## **Appendix**

**Appendix A** lists the error messages that can be generated by 4DOS, 4NT, and Take Command. Look here if you need an explanation of an error message and an idea how to address the problem that caused it.

## **Index**

If you can't find the information you need, this should help you find it.

## **Additional Information**

Files distributed with each of our products cover important additional information beyond what's included in this manual.

4DOS, 4NT, and Take Command include complete online help for all commands and other features. The online help provides much of the same information that is available in this manual and the *Introduction and Installation Guide*, but in an electronic form which you can access quickly. The online help also includes details about changes in the latest version of our products, compatibility with other products, and additional technical and reference information. See your *Introduction and Installation Guide* for more information about the online help.

When you receive your copy of 4DOS, 4NT, or Take Command, be sure to look through *README.TXT*, which contains general notes, highlights of the latest release, and brief installation instructions for those upgrading from a downloaded copy. Important additional information may be included in other files distributed with your product; if so, *README.TXT* will refer you to those files as well.

## **Introduction and Installation Guide**

In addition to this manual, we include an *Introduction and Installation Guide* with each of our products. Besides installation information, this guide contains information about features that are unique to each product. For example, the Guide contains detailed information about start-up options for 4DOS, 4NT, and Take Command. The *Introduction and Installation Guide* is as important to your use of your new command processor as this manual. Even if you are an experienced user, please browse through the Guide to see what information it contains.

### ***Registration and Upgrade Information***

If you purchased 4DOS, 4NT, or Take Command from a software dealer, your copy came with a registration card (packed with the literature accompanying your product). **Please fill out this card and return it promptly to JP Software.** It ensures that we have a record of your registration so that we can give you ongoing technical support and notices of upgrades. If you purchased 4DOS, 4NT, or Take Command directly from JP Software, you are already registered and no registration card is necessary.

4DOS, 4NT, and Take Command are upgraded through **maintenance releases**, designed to fix minor problems or improve compatibility, and **major upgrades** which contain enhancements and additional features. For complete details about obtaining upgrades and new releases, see your *Introduction and Installation Guide*.

### ***Technical Support***

Technical support is available our public electronic support forum, private electronic mail, telephone, fax, and mail. For complete details see the section on Technical Support in your *Introduction and Installation Guide* or the **Troubleshooting, Service and Support** topic in the online help.



## **CHAPTER 1 / GENERAL CONCEPTS**

You can start using 4DOS, 4NT, and Take Command as soon as you finish installing them, because each of our products is compatible with the traditional commands you're probably used to. But most users find that the more they know about their computer systems, the more power they get from our products. And, the more experience they gain, the more they want to know about their computer system as a whole.

This section of the manual explains some fundamental concepts about your computer, operating system, and 4DOS, 4NT, and Take Command. It should help you understand the terms and concepts in the pages that follow. If you find some of the concepts overwhelming, just remember that they are here when you need them. If you find this material too simple, skim over the topics and then go on to the next section. Each topic in this chapter is independent, so if you read it straight through you won't necessarily find a natural "flow" from one topic to another.

If you come across terms or concepts in this chapter that you are unsure about, refer to the Index, or to the Glossary in the online help.

### ***Operating Systems and Command Processors***

This section explains briefly what an operating system is, what a command processor is, and how the command processor works under DOS, Windows 95 / 98 / ME, and Windows NT / 2000 / XP.

An operating system is nothing more than a collection of software. However, unlike application software, which performs a specific, user-oriented task (such as creating and printing documents, or performing calculations on rows and columns of numbers), operating system software is designed to perform some special functions. The operating system typically:

- ✓ Starts the computer system. The operating system is the first software loaded when you turn on the computer.
- ✓ Provides services to other software. These include basic file access, assignment of your computer's memory for use by different programs, and the control of hardware devices like the keyboard, display, printer, and serial ports.
- ✓ Ensures that programs don't interfere with each other while they are running.

- ✓ Provides a way for you to start programs to do your work.
- ✓ Includes utilities to control and manage your system, for example to organize disks and files, display status information, or adjust your system for international conventions.

More complex operating systems may include many more functions, such as built-in network connections, the ability to switch rapidly between many tasks, support for high-quality sound output, and so on.

Our products run under the DOS, Windows 95, Windows 98, Windows ME, Windows NT, Windows 2000, and Windows XP operating systems.

Each of these operating systems includes a command processor: a program that accepts your instructions and carries them out. The basic operation of a command processor is very simple. It prompts you for a command, you type in the command, the command processor executes it, and the cycle repeats. This is sometimes called a “command line” interface, because commands are typically executed line by line.

Under DOS, the command-line interface is the only way to tell the operating system what you want it to do. Windows includes a graphical user interface that is intended to replace the command line as the primary method for providing instructions to the operating system. However, most Windows users find that the command line is required for some tasks, and much more efficient for many others. They therefore use both the graphical interface and the command line, depending on the task at hand.

The command processor shipped with DOS and Windows 95, 98, and ME is called *COMMAND.COM*. The command processor shipped with Windows NT / 2000 / XP is called *CMD.EXE*.

When you install 4DOS, *4DOS.COM* replaces *COMMAND.COM* as your command processor. 4NT replaces the Windows NT / 2000 / XP version of *CMD.EXE* with *4NT.EXE*. (The original command processor is not deleted or removed from your system, but you no longer need to use it once 4DOS or 4NT is available). All of these command processors are normal programs that know how to translate your commands into actions. 4DOS and 4NT are simply much more powerful command processors than those supplied with the operating system.

Take Command performs functions similar to 4DOS and 4NT, but it operates as a Windows “GUI” application (as opposed to a “text mode” application. Because Windows does not include a GUI command processor, Take

Command does not replace an existing operating system program as 4DOS and 4NT do. Instead, it is used as an additional utility run from the desktop.

**4DOS** Under DOS and Windows 95 / 98, the command processor is typically started automatically at the end of the boot process. If you don't want to use *COMMAND.COM* as your command processor, you use the **SHELL** statement in *CONFIG.SYS* to specify a different command processor.

**4NT, TC** Under Windows ME, NT, 2000, and XP, there is no *CONFIG.SYS* file, and no standard way to specify a default command processor. Instead, you normally just create a desktop shortcut for any command processor you want to use.

When you install 4DOS, 4NT, or Take Command, the installation program can make the appropriate changes to *CONFIG.SYS* and / or create the necessary desktop shortcuts.

## ***Primary and Secondary Shells***

Technically, the command processor is a **shell**: a program that understands your commands and makes the correct calls to the underlying operating system to perform various tasks, including running programs.

### ***4DOS***

Under DOS, the command processor that runs when the computer boots up is called the **primary shell**.

Any command processor that is started as a “child” to a primary shell is a **secondary shell**. Typically, secondary shells are started by application programs with a “shell to the command prompt” feature, and by multitasking environments like Windows. A secondary shell has most of the same features as a primary shell.

There are a few differences between primary and secondary shells under DOS. Generally, less memory is available when a secondary shell is running, because at least part of the program that started it is still in memory waiting to spring back to life when you exit. And normally only the primary shell automatically executes the instructions in your *AUTOEXEC.BAT* file.

Under DOS you can leave a secondary shell with the EXIT command, but there is no way to exit from the primary shell, because DOS requires a shell present to operate.

Under Windows 95 and 98, 4DOS shells started from the desktop are secondary shells because a primary shell is loaded before Windows starts. Under Windows ME, each 4DOS shell started from the desktop is a primary shell, because no 4DOS primary shell can be loaded before Windows starts.

Each time 4DOS starts as either a primary or secondary shell, it looks for and executes one or more “automatic” batch files. These files are explained on page 132.

### ***4NT and Take Command***

Each 4NT or Take Command process that is started from the desktop is a **primary shell**. **Secondary shells** are started by applications with a “shell to the command prompt” feature, and are also used for a special kind of communication between programs called a pipe (see page 91).

Under 4NT and Take Command, you can exit from both primary and secondary shells. When you exit from a primary shell, the command processor is closed and you are returned to the desktop.

Like 4DOS, each time 4NT or Take Command starts it looks for and executes the “automatic” batch files explained on page 132.

## ***Files and Directories***

You may have dozens, hundreds, or thousands of files stored on your computer's disks. Your operating system is responsible for managing all of these files. In order to do so, it uses a unique name to locate each file in much the same way that the post office assigns a unique address to every residence.

The unique name of any file is composed of a **drive letter**, a **directory path**, and a **filename**. Each of these parts of the file's name is case insensitive; you can mix upper and lower case letters in any way you wish.

### ***Drives and Volumes***

A **drive letter** designates which drive contains the file. In a file's full name, the drive letter is followed by a colon. Drive letters **A:** and **B:** are normally reserved for the floppy disk drives.

Normally, drive **C:** is the first (or only) hard disk drive. Most operating systems can divide or “partition” a large hard disk into multiple logical drives

or **volumes** that are usually called **C:**, **D:**, **E:**, etc. Network systems (LANs) may assign (“map”) additional drive letters to sections of the network file server drives. In addition, you can often access network drives via their name (*e.g.* `\\server\vol1\...`), without using a drive letter. See page 13 for more details.

Most recent systems also include a CD-ROM drive. The CD-ROM is also assigned a drive letter (or several letters, for CD-ROM changers), typically using letters beyond that used by the last hard disk in the system, but before any network drives.

Some systems may have “RAM disks” (sometimes called “virtual disks”), which are areas of memory set aside by software (a “RAM disk driver”) for use as fast but temporary storage.

For example, on a system with a large hard disk you might have **A:** and **B:** as floppy drives, **C:**, **D:**, and **E:** as parts of the hard disk, **F:** as a CD-ROM drive, **G:** as a RAM disk, and **H:** and **I:** as mapped network drives.

## ***File Systems***

Each disk volume is organized according to a **file system**. The file system determines how files are named and how they are organized on the disk.

As hard disk technology and operating systems have evolved, new file systems have been invented to support longer file names, larger drives, and higher disk performance. Several different and incompatible schemes have evolved. Which file systems you can use depends on which operating system you are using, and how the operating system and your hard disk are configured.

The operating systems under which our products run support four standard file systems: FAT, VFAT, FAT32, and NTFS. See **File Names** on page 15 for details on the rules for naming files under each file system.

- ✓ The **FAT File System** is the traditional file system used by all versions of DOS. Its name comes from the File Allocation Table DOS uses to keep track of the space allocated to each file. Windows 95, 98, ME, NT, and Windows 2000 and XP also support the FAT file system.
- ✓ The **VFAT File System** is a Windows extension of the FAT file system. This system maintains additional information about files on FAT drives, including long filenames (LFNs).

- ✓ The **FAT32 File System** is an additional extension to the VFAT file system. It is available in Windows 95 OEM Service Release 2 (“OEMSR2” or Windows 95 B) and later, in Windows 98, Windows ME, and Windows 2000 and XP, but not in Windows NT. It is similar to the VFAT file system, but supports larger disk drives. This file system is not supported under Windows NT 4.0 (pre-Windows 2000) or under DOS.
- ✓ The **Windows NT File System** or **NTFS** is a file system provided with all versions of Windows NT / 2000 / XP. NTFS supports long file names and offers improved performance and support for large drives. Other operating systems (DOS and Windows 3.x / 95 / 98 / ME) cannot normally access files on NTFS drives, although some third-party utilities (*e.g.* NTFSDOS) are available which allow access.

Throughout this manual, the term “LFN file system” is used to describe the VFAT, FAT32, and NTFS systems as a group (LFN stands for Long File Name).

Additional file systems may be installed under some operating systems to support CD-ROM or network drives.

The file system type (FAT, VFAT, FAT32, or NTFS) is determined when a hard disk volume is formatted and applies to the entire volume. For example, you might have a 60 GB hard disk partitioned into four 15000 MB volumes, with the first three volumes (C:, D:, and E:) formatted for the FAT32 file system, and the fourth formatted for NTFS.

4DOS, 4NT, and Take Command support any standard file system installed under your operating system. 4DOS can access files on FAT drives, and files on VFAT and FAT32 drives under Windows 95 / 98 / ME (with full long filename support). 4NT and Take Command can access all files supported by the operating system.

### ***Network File Systems***

A network file system allows you to access files stored on another computer on a network, rather than on your own system. 4DOS, 4NT, and Take Command support all network file systems which are compatible with the underlying operating system.

File and directory names for network file systems depend on both the “server” software running on the system that has the files on it, and the “client”

software running on your computer to connect it to the network. However, they usually follow the rules described here.

Most network software “maps” unused drive letters on your system to specific locations on the network, and you can then treat the drive as if it were physically part of your local computer.

Many networks also support the **Universal Naming Convention**, which provides a common method for accessing files on a network drive without using a “mapped” drive letter. Names specified this way are called UNC names. They typically appear as `\\server\volume\path\filename`, where **server** is the name of the network server where the files reside, **volume** is the name of a disk volume on that server, and the **path\filename** portion is a directory name and file name which follow the conventions described under **Directories** below. 4NT and Take Command also allow you to use UNC directory names when changing directories (see Directory Navigation on page 81 for more details).

When you use a network file system, remember that the naming conventions for files on the network may not match those on your local system. For example, your local system may support long filenames while the network server or client software does not, or vice versa. 4DOS, 4NT, and Take Command will usually handle whatever naming conventions are supported by your network software, as long as the network software accurately reports the types of names it can handle.

In some cases, 4DOS may not be able to report correct statistics on network drives (such as the number of bytes free on a drive). This is usually because the network file system does not provide complete or accurate information, or because the network drive is larger than drives which DOS normally supports, and the byte count therefore exceeds DOS's reporting limits.

## ***Directories and Subdirectories***

A file system is a method of organizing all of the files on an entire disk or hard disk volume. **Directories** or **folders** are used to divide the files on a disk into logical groups that are easy to work with. Their purpose is similar to the use of file drawers to contain groups of hanging folders, hanging folders to contain smaller manila folders, and so on. (The terms **directory** and **folder** are nearly synonymous -- we use **directory** throughout this manual.)

Every drive has a **root** or base directory, and most have one or more **subdirectories**. Subdirectories can also have subdirectories, extending in a branching **tree** structure from the root directory. The collection of all

directories on a drive is often called the **directory tree**, and a portion of the tree is sometimes called a **subtree**. The terms **directory** and **subdirectory** are typically used interchangeably to mean a single subdirectory within this tree structure.

Subdirectory names follow the same naming rules as files in each operating system (see below). However, under DOS it is best to use a name of 8 characters or less, without an extension, when naming subdirectories, because some application programs do not properly handle subdirectory names that have an extension.

The drive and subdirectory portion of a file's name are collectively called the file's **path**. For example, the file name *C:\DIR1\DIR2\MYFILE.DAT* says to look for the file *MYFILE.DAT* in the subdirectory *DIR2* which is part of the subdirectory *DIR1* which is on drive C. The path for *MYFILE.DAT* is *C:\DIR1\DIR2*. The backslashes between subdirectory names are required.

The total length of a file's path may not exceed 64 characters in DOS (this limit excludes the file name and extension, but includes the drive letter and colon). With 4DOS, the path and file name on LFN volumes must each be 255 characters or less in length, and in addition the total length of the path and file name together cannot exceed 260 characters. With 4NT and Take Command, the path and filename can be up to 2047 characters, though many Windows applications (including Explorer) have trouble with path and filename lengths exceeding 260 characters.

The operating system and command processor remember both a **current or default drive** for your system as a whole, and a **current or default directory** for every drive in your system. Whenever a program tries to create or access a file without specifying the file's path, the operating system uses the current drive (if no other drive is specified) and the current directory (if no other directory path is specified).

The root directory is named using the drive letter and a single backslash. For example, *D:\* refers to the root directory of drive *D*:. Using a drive letter with no directory name at all refers to the current directory on the specified drive. For example, *E:4NT.DOC* refers to the file *4NT.DOC* in the current directory on drive *E*:, whereas *E:\4NT.DOC* refers to the file *4NT.DOC* in the root directory on drive *E*..

There are also two special subdirectory names that are useful in many situations: a single period by itself [*.*] means “the current default directory.” Two periods together [*..*] means “the directory which contains the current default directory” (often referred to as the **parent directory**). These special



names can be used wherever a full directory name can be used. 4DOS, 4NT, and Take Command allow you to use additional periods to specify directories further “up” the tree (see page 94).

## ***File Names***

Finally, each file has a **filename**. Under the FAT file system, the filename consists of a **base name** of 1 to 8 characters plus an optional **extension** composed of a period plus 1 to 3 more characters. Traditional FAT filenames with an 8-character name and a 3-character extension are sometimes referred to as short filenames (SFNs) to distinguish them from long filenames (LFNs).

You can use alphabetic and numeric characters plus the punctuation marks ! # \$ % & ' ( ) - @ ^ \_ ` { } and ~ in both the base name and the extension of a FAT filename. Because the exclamation point [!], percent sign [%], caret [^], at sign [@], parentheses [()], and back-quote [`] also have other meanings to 4DOS, 4NT, and Take Command, it is best to avoid using them in filenames.

The VFAT and FAT32 file systems (which can be used under Windows 95, Windows 98, Windows ME, Windows 2000, and Windows XP), and the NTFS file system (supported in Windows NT / 2000 / XP) allow file names with a maximum of 255 characters, including spaces and other characters that are not allowed in a FAT system file name, but excluding some punctuation characters which are allowed in FAT file names. See your operating system documentation for details on the characters allowed. If you use file names which contain semicolons [;], see page 97 for details on avoiding problems with interpretation of those file names under 4DOS, 4NT, and Take Command.

FAT file names are always stored on the disk in upper case, and are displayed in upper or lower case depending on the options you select in 4DOS, 4NT, and Take Command. LFN file names are stored and displayed exactly as you entered them, and are not automatically shifted to upper or lower case. For example, you could create a file called *MYFILE*, *myfile*, or *MyFile*, and each name would be stored in the directory just as you entered it. However, case is ignored when looking for filenames, so you cannot have two files whose names differ only in case (*i.e.*, the three names given above would all refer to the same file). This behavior is sometimes described as “case-retentive but not case-sensitive” because the case information is retained, but does not affect access to the files.

Files stored on LFN volumes have “FAT-compatible” names: names which contain only those characters legal on a FAT volume, and which meet the 8-character name / 3-character extension limits. Programs which cannot handle long names generally can access files by using FAT-compatible names.

If an LFN-compatible file name includes spaces or other characters that would not be allowed in a FAT name, you **must** place double quotes around the name. The quotes should be around the entire name, though they may also work properly around the filename portion only, if that is the only part of the name that contains special characters. For example, suppose you have a file named *LET3* on a FAT volume, and you want to copy it to the *LETTERS* directory on drive F:, an LFN volume, and give it the name *Letter To Sara*. To do so, use either of these commands:

```
[c:\wp] copy let3 f:\LETTERS\ "Letter To Sara"  
[c:\wp] copy let3 "f:\LETTERS\Letter To Sara"
```

The LFN file systems do not explicitly define an “extension” for file names which are not FAT-compatible. However, by convention, all characters after the last period in the file name are treated as the extension. For example, the file name “*Letter to Sara*” has no extension, whereas the name “*Letter.to.Sara*” has the extension *Sara*.

You may occasionally encounter filenames which are not displayed the way you expect if you have used characters from outside the U.S. English character set in the name. These are generally due to problems in the way your operating system translates characters between the OEM and ANSI character sets. Correcting the problem may require experimentation with fonts, character sets, and code pages, and occasionally some such problems may not be readily correctable within 4DOS, 4NT, and Take Command. For more information on underlying issues related to fonts and character sets see page 24.

### ***File Attributes and Time Stamps***

Each file also has **attributes** and one or more **time stamps**. Attributes define characteristics of the file which may be useful to the operating system, to you, or to an application program. Time stamps can record when the file was created, last modified, or last accessed. Most 4DOS, 4NT, and Take Command file processing commands allow you to select files for processing based on their attributes and / or time stamp(s).

Each file on your system has four standard attributes. Every time a program modifies a file, the operating system sets the **Archive** attribute, which signals that the file has been modified since it was last backed up. This attribute can be used by 4DOS, 4NT, or Take Command to determine which files to COPY or MOVE, and by backup programs to determine which files to back up. When the **Read-only** attribute is set, the file can't be changed or erased accidentally; this can be used to help protect important files from damage. The **Hidden** and **System** attributes prevent the file from appearing in normal directory listings. (Two additional attributes, **Directory** and **Volume label**, are also available. These attributes are controlled by the operating system, and are not modified directly by 4DOS, 4NT, or Take Command.)

Windows 2000 and XP support additional attributes including **Encrypted**, **Compressed**, **Normal**, **Offline**, **Temporary**, **Sparse**, **Not content-indexed**, and **Reparse point**. These attributes can be accessed with the ATTRIB and DIR commands, and the @WATTRIB variable function (page 187).

Attributes can be set and viewed with the ATTRIB command (see page 263). The DIR command (see page 303) also has options to select filenames to view based on their attributes, to view the attributes themselves, and to view information about normally “invisible” hidden and system files.

When a file is created, and every time it is modified, the operating system records the system time and date in a **time stamp** in the file's directory entry. Several 4DOS, 4NT, and Take Command variable functions and commands, and many backup and utility programs, use this time stamp to determine the relative ages of files.

On FAT volumes, only the single time stamp described above is available. Files on LFN volumes have three sets of time and date stamps. The operating system records when each file was created, when it was last written or modified, and when it was last accessed. The “last write” time stamp matches the single time stamp used on traditional FAT volumes.

Several 4DOS, 4NT, and Take Command functions and commands let you specify which set of time and date stamps you want to view or work with on LFN volumes. These commands and functions use the letter “c” to refer to the creation time stamp, “w” for the last write time stamp, and “a” for the last access time stamp. Note that FAT32 and VFAT volumes, whether under Windows 95 / 98 / ME, Windows NT / 2000 / XP, store a date but no time in the “last access” time stamp; on these drives the time of last access will always be 00:00.

### **NTFS File Streams**

Microsoft's NTFS file system (in Windows NT 4.0 and above, including Windows 2000 and XP) allows each file to contain multiple “streams” or sets of data. For example a compiler could use streams to store a program's source code, object code, and other data, or a word processing program could use them to store multiple versions of the same document.

Streams are specified by entering a stream name following the file name, for example:

```
myfile.doc:version1  
myfile.doc:version2
```

Stream names must be spelled out, wildcards can not be used.

A file which includes multiple streams may also contain data stored as part of the base file, and not in any stream.

You can display the stream names with the **DIR /:** option. The file processing commands COPY, DEL, LIST, MOVE and TYPE support file streams when the stream name is explicitly specified. See the individual commands for additional details. Other file-related commands, such as ATTRIB, REN and TOUCH work with the file as a whole, and not with any particular stream or portion of the file data.

Variable functions which reference file contents, such as @FILEOPEN, @LINE, and @LINES will also accept stream names.

Other commands and functions which return file information (TREE, @FILESIZE, etc.) do not include stream data.

### **Internal and External Commands**

Whenever you type something at the command line prompt and press the **Enter** key, you have given a command to the command processor, which must figure out how to execute your command. If you understand the general process that 4DOS, 4NT, and Take Command use, you will be able to make the best use of them and their commands.

The command processor begins by dividing the line you typed into a **command name** and a **command tail**. The command name is the first word in the command; the tail is everything that follows the command name. For example, in this command line:

```
dir *.txt /2/p/v
```

the command name is “dir”, and the command tail is “ \*.txt /2/p/v.”

If the command name is not an alias or, in 4NT or Take Command, an Internet URL beginning with **http:** (see page 112), the command processor tries to find the name in its list of **internal commands**. An internal command is one that the command processor can perform itself, without running another program. DIR and COPY are examples of internal commands.

If the command name is not found in the command processor's list of internal commands, it assumes that it must find and execute an **external command**. This means that the command processor must load and run a separate program, either an executable program or a batch file. DOS and Windows are shipped with a number of external utility programs (such as FORMAT and DISKCOPY), and any program or application you install on your system becomes a new external command.

The advantage of internal commands is that they run almost instantly. When you type an internal command, the command processor interprets the command line and carries out the necessary activities without having to look for, load, and run another program.

The advantage of external commands is that they can be large, varied, and complex without taking space inside the system command processor. External commands can also be renamed or replaced easily. If you want to rename the external DOS command XCOPY to MYCOPY, for example, all you need to do is find the file called *XCOPY.EXE* in your DOS or Windows\Command directory and change its name to *MYCOPY.EXE*. If you want to replace XCOPY with a more efficient program of the same name, you can do so. 4DOS, 4NT, and Take Command add this flexibility to internal commands. You can rename or replace any internal command by using an alias (see pages 122 and 250), and you can enable or disable internal commands whenever you wish (see SETDOS /I on page 475).

## ***Executable Files and File Searches***

Once the command processor knows that it is supposed to run an external command, it tries to find an executable file whose name matches the command name. (Executable files are typically those with a *.COM* or *.EXE* extension, or with a *.PIF* extension under Windows.) It runs the executable file if it finds one. (See the previous section for more information on external commands.)

If the command processor cannot find a program to run, it next looks for a batch file (a file with one or more commands in it) whose name matches the command name. The command processor looks first for a *.BTM* file, then for a *.CMD* file (in 4NT and Take Command), then for a *.BAT* file, and finally for a *.REX* or *.REXX* file (in 4NT and Take Command). See page 125 for more information on these different types of batch files. If the command processor finds such a file, it then reads each line in the file as a new command.

**4NT, TC** Under 4NT and Take Command you can change the list of extensions that are considered “executable”, and the order in which they are searched, with the *PATHEXT* environment variable and the related *PathExt* directive in the *.INI* file; see page 152 for details. *PATHEXT* is supported for compatibility reasons but should not generally be used as a substitute for executable extensions (see below), which are more flexible.

If the search for an external program or batch file fails, the command processor checks to see if the command name matches the name of a file with an “executable extension.” (An executable extension associates a file extension with a specific program to process that type of file – for example if you have associated *.DOC* with your editor or word processor, and type the name of a *.DOC* file, the command processor can start the editor automatically. See page 106 for additional details.) If an executable extension is found, the command processor runs the program specified when the association was defined.

4DOS, 4NT, and Take Command first perform this search (for an executable program, a batch file, or a file with an executable extension) in the current directory. If that search fails, they repeat the search in every directory in your **search path**.

The search path is a list of directories that the command processor (and some applications) search for executable files. For example, if you wanted the command processor to search the root directory of the C: drive, the \WINUTIL subdirectory on the C: drive, and the \UTIL directory on the D: drive for executable files, your search path would look like this:

```
PATH=C:\;C:\WINUTIL;D:\UTIL
```

Notice that the directory names in the search path are separated by semicolons.

You can create or view the search path with the *PATH* command (see page 429). You can use the *ESET* command (see page 334) to edit the path. Many

programs also use the search path to find their own files. The search path is stored in the environment with the name PATH.

**TC** Under Windows 95, and 98, and ME, before searching the PATH Take Command searches the \WINDOWS directory then the \WINDOWS\SYSTEM directory; under Windows NT / 2000 / XP the order is reversed, and Take Command searches the \WINDOWS\SYSTEM32 directory followed by the \WINDOWS directory. (The actual directory names may be different on your system. Take Command will determine the correct names for the “Windows” and “Windows System” directories and use them.) These search procedures conform with the traditional search sequences used under each Windows operating system.

**4NT, TC** If the file is not found on the PATH, 4NT and Take Command also check for a corresponding App Paths entry in the Windows registry. App Paths entries are created by some applications during the installation process.

Remember, the command processor always looks for an executable file (or a file with an executable extension) in the current subdirectory, then in the Windows directories (Take Command only), and then in each directory in the search path. (You can change the search order so the current directory is not searched first; see the PATH command for details.)

If you include an extension as part of the command name, 4DOS, 4NT, and Take Command only search for a file with that extension. Similarly, if you include a path as part of the command name, the command processor will look only in the directory you specified, and ignore the usual search of the current directory and the PATH.

If your command name includes a path, the elements must be separated with backslashes (e.g. *C:\WP7\WP*). If you are accustomed to Unix syntax where forward slashes are used in command paths, and want the command processor to recognize this approach, you can set UnixPaths to Yes in the .INI file (see page 227).

Once the file is found, the command processor executes it based on its extension. *.EXE* and *.COM* files are executed by passing their names to the operating system. *.BTM*, *.BAT*, and (if applicable) *.CMD* files are executed by the command processor, which reads each line in the file as a new command. Files with executable extensions are executed by starting the associated application, and passing the name of the file on the command line.

**4NT, TC** If you specify a file name including extension, and the file exists in the current directory (or in the directory you specify), but the file does not have

an extension known to 4NT or Take Command (*.EXE*, *.COM*, *.BTM*, *.BAT*, *.CMD*, or an executable extension), then the file name will be passed to Windows to check for file associations defined in the Windows registry. This allows you to execute any file whose extension is known to Windows, simply by typing its name. For example, if you have no executable extension defined for *.PSP* files, but this is an extension known to Windows, at the prompt you can simply enter a command like this (the extension must be entered):

```
[c:\graphics] imagel.psp
```

and 4NT or Take Command will request that Windows start the application for you. See Using Windows File Associations on page 112 for additional details on how to control Windows file associations in 4NT and Take Command. Also see the ASSOC and FTYPE commands (pages 261 and 358).

The following table sums up the possible search options (the term “standard search” refers to the search of the current directory, the Windows directories in Take Command, and each directory in the search path):

<u>Command</u>	<u>4DOS, 4NT, and Take Command Search Sequence</u>
<b>WP</b>	Standard search for any executable file whose base name is <i>WP</i> .
<b>WP.EXE</b>	Standard search for <i>WP.EXE</i> ; will not find files with other extensions.
<b>C:\WP7\WP</b>	Looks in the <i>C:\WP7</i> directory for any executable file whose base name is <i>WP</i> . Does not check the standard search directories.
<b>C:\WP7\WP.EXE</b>	Looks only for the file <i>C:\WP7\WP.EXE</i> .
<b>LAB.DOC</b>	Standard search for <i>LAB.DOC</i> , if <i>.DOC</i> is defined as an executable extension. Runs the associated application if the file is found. If <i>.DOC</i> is not an executable extension, passes the name to Windows to check for a Windows file association.
<b>C:\LI\LAB.DOC</b>	Looks only for the file <i>C:\LI\LAB.DOC</i> , and only if <i>.DOC</i> is defined as an executable extension. Runs the associated application if the file is found. If <i>.DOC</i> is not an executable extension, passes the name to Windows to check for a Windows file association.



If the command processor cannot find an executable file, batch program, or a file with an executable extension in the current directory, a directory in the search path, or the directory you specified in the command, it then looks for an alias called `UNKNOWN_CMD` (see `ALIAS` on page 250 for details). If you have defined an alias with that name, it is executed (this allows you to control error handling for unknown commands). Otherwise, the command processor displays an “Unknown command” error message and waits for your next instruction.

## ***The Environment***

The operating system allows you to keep a list of information in memory. This list is called the **environment**. Every program receives a copy of the environment when it begins, and many programs use some of its information to configure themselves or to find files.

The environment is arranged as a series of **variables** and their related **values**. Each variable consists of a **name** followed by an equal sign [=] and some text (the **value**). For compatibility with traditional command processors, 4DOS stores the variable name in upper case, regardless of how you enter it. Like Windows NT's *CMD.EXE*, 4NT and Take Command store the name as you enter it, and do not shift it to upper case (however, case is ignored when looking for a variable; for example **MyVar**, **myvar**, and **MYVAR** all refer to the same variable).

You can view the environment by typing `SET`, and add new entries or edit existing entries with the `SET` and `ESET` commands. You can remove entries from the environment with both the `UNSET` and `SET` commands.

A typical environment entry looks like this:

```
LIB=c:\lib
```

In this example, the name of the variable is “LIB” and its value is “c:\lib”. The text string or value can include any character except a null (ASCII 0).

The format and meaning of each entry in the environment is up to the program that uses the particular variable. Environment variables can contain just about anything, and can be used for any purpose the author of a program desires. The “purpose” of the environment as a whole is simply to hold small amounts of text which programs can then access according to their own rules. Most environment variables are used by single programs for their own information; a few (like `PATH`) have well-defined meanings and are used by many different programs.

4DOS, 4NT, and Take Command use several environment variables to control their own behavior. They also provide a wide range of facilities for manipulating and managing the environment. See “Using the Environment” beginning on page 148 for details about these special variables other environment-related features.

When a program starts, it inherits a copy of the environment. Normally, any changes it makes are visible only to programs which it starts (see the ESET /M and SET /M commands for an exception to that rule under DOS).

If you use 4DOS, 4NT, or Take Command to start an application, the new program may inherit either the command processor's copy of the environment or Windows' default environment. Unless you are using the command processor to specifically set environment variables that you want to pass to an application, you probably won't care which version of the environment is inherited by the new program.

The rules governing environment inheritance are complex and depend on which specific version of Windows you are using, which command processor you are using, and what command you use to start the application. You may have to experiment in any specific situation to determine which environment a program receives from 4DOS, 4NT, or Take Command. The following general rules may help you select the correct way to start an application to make sure it receives the environment variables and values you want it to have.

- ✓ Applications started directly from Windows will normally inherit the default Windows environment and will not “see” environment variables set by 4DOS, 4NT, or Take Command.
- ✓ Applications started from 4DOS, 4NT, or Take Command will normally inherit the 4DOS, 4NT, or Take Command environment and will be able to see variables set within 4DOS, 4NT, or Take Command.
- ✓ Applications started with the START /I command from 4NT or Take Command will normally inherit the default Windows environment and will not “see” environment variables set by 4NT or Take Command.

## ***Character Sets, ASCII, and Key Codes***

The translation of a key you type on the keyboard to a displayed character on the screen depends on several related aspects of character handling. A

complete discussion of these topics is well beyond the scope of this manual. However, a basic picture of the steps in the keystroke and character translation process will help you understand how characters are processed in your system, and why they occasionally may not come out the way you expect.

Internally, computers use numbers to represent the keys you press and the characters displayed on the screen. To display the text that you type, your computer and operating system require five pieces of information:

- ✓ The numeric **key code** for the physical key you pressed;
- ✓ The specific character that key code represents based on your current keyboard layout or **country setting**;
- ✓ The **character set** currently in use on your system (see below);
- ✓ The international **code page** in use for that character set; and
- ✓ The **display font** used to display the character.

The numeric **key code** is determined by your physical hardware including the language for which your keyboard was manufactured. The **character set** is usually determined by the operating system. These items typically are not under your control. However, most systems do allow you to control the keyboard **country setting** and the **code page**. In Take Command, you can also control the **display font**.

If the key codes produced by your keyboard, the code page, and the font you choose are not fully compatible with each other, the characters displayed on the screen will not match what you type. The differences are likely to appear in line-drawing characters, “international” (non-English) characters, and special symbols, not in commonly-used alphabetic, numeric, or punctuation characters.

DOS, Windows 95, Windows 98, and Windows ME use a “single-byte” character set for keyboard and screen display. These sets define 256 characters or symbols, with a numeric representation for each. (“Double-byte” character sets, with up to 65,536 characters each, are used for languages with more than 256 symbols, and for some multi-lingual systems.) Most PC single-byte character sets are based on a code called ASCII, the American Standard Code for Information Interchange.

The original ASCII code was defined during the 1960s for use in mainframe and minicomputer systems, and has 128 character values. These include the

upper and lower case letters, numerals, and punctuation marks used in U.S. English, plus non-printing control codes (which can be entered on most keyboards by pressing the **Ctrl** key plus another character, or by pressing certain special keys like **Tab**, **Enter**, **Backspace**, and **Esc**). However, ASCII is not a complete character set, because it defines only 128 of the required 256 symbols.

IBM, in its original PC, created a complete 256-character set (called the Original Equipment Manufacturer or “OEM” character set) by defining an additional 128 **extended ASCII** codes for math symbols, “international” characters, the characters used to draw boxes and lines, and some miscellaneous symbols.

Some operating systems support other character sets; in particular, Windows uses the ANSI character set internally to store and display text, even though other parts of the system (*e.g.* the file system which stores file names on disk) use IBM's OEM character set. The ANSI character set is identical to the OEM character set for U.S. English printed characters, but may vary for “international” characters not used in U.S. English. In addition, Windows NT / 2000 / XP use Unicode, a character set that allows for more than 65,000 characters, in order to support languages which do not use the Latin alphabet.

4NT and Take Command are available in both ASCII (Windows 95 / 98 / ME) and Unicode (Windows NT / 2000 / XP) versions.

In most cases, Windows automatically translates characters from one set to another as needed, but problems can sometimes result in errors in displayed text (*e.g.*, differences between the appearance of accented characters in filenames in Windows and DOS applications).

See your operating system documentation for more information about character sets, code pages, and country and language.

## ***The Keyboard***

### ***4DOS and 4NT***

When you press a single key or a key combination on the keyboard, the computer translates your keystroke into two numbers. For all alphabetic, numeric, and punctuation keys, the **Tab**, **Enter**, **Backspace**, **Esc** keys and **Ctrl** plus an alphabetic key, these numbers are an ASCII (or, for 4NT, Unicode) code plus a scan code. The ASCII code represents the key's

meaning; the scan code identifies which specific key was pressed. For example, many keyboards have two plus [+] keys, one above the equal sign and one on the numeric keypad. Both generate the same ASCII code, but they generate different scan codes.

Keys which are not represented by ASCII codes are translated to an ASCII 0 plus an **extended key code**. These keys include the function keys, the cursor keys, and **Alt** plus a key. The extended key code for a key is often the same as the scan code for that key. Do not confuse **extended key codes** with the **extended ASCII** codes discussed in the previous section. The former are the special codes for function keys, cursor keys, and **Alt** keys. The latter are extensions to the ASCII character set for graphical and international characters; see the previous section for details.

The translation of keystrokes to internal codes is affected by your current keyboard layout (country setting). This allows you to work with keyboards with a variety of different layouts designed for different countries.

Some keys, like the **Alt**, **Ctrl**, and **Shift** by themselves or in combination with each other, plus **Print Screen**, **SysReq**, **Scroll Lock**, **Pause**, **Break**, **Num Lock**, and **Caps Lock**, do not have any code representations at all in normal use (these keys do have physical scan codes, but those codes are normally handled internally by the BIOS or operating system). The computer performs special actions automatically when you press these keys (for example, it switches your keyboard into Caps Lock mode when you press the Caps Lock key), and does not report the keystrokes to whatever program is running unless the program has been written specially to accept them.

See your operating system documentation for more information about keyboard layouts and country and language support. The online help system has complete tables of standard ASCII and key codes. For Take Command, refer to your operating system and / or font documentation for details on the full character set available in any particular font.

## ***Take Command***

When you press or release a key on the keyboard, Windows intercepts the keystroke and interprets its meaning. Most keystrokes are translated into messages which are sent to the application that has the "focus," that is, to the program whose window is highlighted. Some keystrokes and key combinations, such as Alt-Tab, are interpreted by Windows as direct commands to perform such actions as changing the focus to a new program, starting the Windows task manager, etc.

Windows always interprets **Alt** plus a key as an “accelerator” keystroke (used to open an application menu, or move to a specific field in a dialog). Such keystrokes are not passed to Take Command. Therefore, Alt-key combinations cannot be used for aliases and other special uses in Take Command.

When the operating system sends a keystroke to Take Command, it arrives as a combination of an ASCII (or Unicode) code and a scan or keyboard code. The online help has a complete list of ASCII and scan codes for the keys on most keyboards.

### **4DOS and 4NT Video**

4DOS and 4NT are “character-based” programs, which means they work in text mode on your computer. In text mode, the screen or window used by the program displays text in a single, fixed-pitch font (a font where every character is the same width), but cannot mix fonts or display graphics. Both command processors can run graphics programs which change your screen to graphics mode, but the screen must be returned to text mode whenever the command processor is active.

Original IBM PC video cards could display 80 columns and 25 rows of text. Newer, advanced video systems normally run in this 80 x 25 display mode but can also display more columns and rows. Each of the different possible screen configurations is a different text “mode”.

In addition to running in a variety of text modes, 4DOS and 4NT can run in “normal” full-screen mode or in a window inside a graphical environment such as the Windows desktop. When they run in such a window, they are still operating in text mode. The underlying graphical environment translates their text commands into the appropriate graphics for your video display.

4DOS and 4NT support whatever number of rows and columns of text you decide to use. They don't have commands to switch from one screen size to another – you will need to use the software that came with your computer or operating system to do that – but they can read and work with the number of rows and columns on your screen

### **ANSI Drivers**

**4DOS** Every version of DOS, Windows 95, and Windows 98 includes a program called *ANSI.SYS*. This program lets you use text colors other than drab

white on black, redefine keys, and control screen output. Commercial and “freeware” ANSI drivers are also available as replacements for *ANSI.SYS*.

In most cases, 4DOS automatically determines whether you have an ANSI driver installed (if the automatic determination does not work on your system you can set the ANSI state manually in *4DOS.INI*, see page 217 for details). If you do have an ANSI driver installed, 4DOS will use the driver to clear the screen and set screen colors.

If you use a 4DOS command or feature which sets screen colors and you do not have an ANSI driver installed the colors will not be “sticky” – you may lose them when you run an application.

**4NT,TC** There is no provision for ANSI support in Windows NT, 2000, or XP. 4NT and Take Command contain their own ANSI support, but it will only work with their own output, and not with external applications. (With the exception that in Take Command, a Caveman app using “default colors” or “stdio” will also use the ANSI support.)

Several 4DOS, 4NT, and Take Command commands provide replacements for traditional ANSI commands. For example, there are commands to set the screen colors and display text in specific colors. These commands are easier to understand and use than the traditional ANSI control sequences. Some of these commands manipulate screen colors directly. Others use an ANSI driver if one is installed (under 4DOS), but save you the work of figuring out ANSI control sequences.

## **CHAPTER 2 / CONVENTIONS**

This chapter contains information about three conventions that are used throughout 4DOS, 4NT, and Take Command: colors and color names, keys and key names, and popup windows. These topics are combined here so that they will be easy to find when you need to refer to them. You will find cross references to this chapter in several other parts of this manual.

### **Colors and Color Names**

You can use color names in several of the directives in the *.INI* file and in many commands. The general form of a color name is:

[BRiGht] [BLInk] *fg* ON [BRiGht] *bg* [BORder *bc*]

where *fg* is the foreground or text color, *bg* is the background color, and *bc* is the border color.

The available colors are:

Black	Blue	Green	Red
Magenta	Cyan	Yellow	White

Color names and BRiGht, BLInk, and BORder may be shortened to the first 3 letters. BLInk and BORder are valid only in full-screen 4DOS sessions.

You can also specify colors by number instead of by name. The numbers are most useful in potentially long *.INI* file directives like ColorDir , where using color names may take too much space. The following numbers are recognized:

0 - Black	8 - Gray ("bright black")
1 - Blue	9 - Bright blue
2 - Green	10 - Bright green
3 - Cyan	11 - Bright cyan
4 - Red	12 - Bright red
5 - Magenta	13 - Bright magenta
6 - Yellow	14 - Bright yellow
7 - White	15 - Bright white

Use one number to substitute for the [BRiGht] *fg* portion of the color name, and a second to substitute for the [BRiGht] *bg* portion. For example, instead



of **bright cyan on blue** you could use **11 on 1** to save space in a ColorDir specification.

There are several subtleties that complicate the use of colors and color names. In order to understand them, you will need to read through the restrictions described below. You may also want to review the Video section in the previous chapter (see page 28). These restrictions are due to the design of your PC video hardware, BIOS, and video drivers, and are not inherent in 4DOS, 4NT, or Take Command. Some of the restrictions are complex, so feel free to skip over those that do not apply to color combinations you use.

Some restrictions depend on the display “mode.” 4DOS and 4NT can run in either **full-screen** display mode, when the command processor is using the entire screen and has more direct control over the video hardware; or in **windowed** display mode, when the command processor appears in a window as part of a graphical display under Windows.

### ***Color Errors***

A standard color specification allows sixteen foreground and eight background colors (sixteen if bright backgrounds are enabled, see below). However, most video adapters and monitors do not provide true renditions of these colors which visually match the names used to describe them. For example, you may see normal “yellow” as brown and bright yellow as yellow, or see normal red as red and “bright red” as pink. Color errors are often worse when running in windowed mode (see above), because the graphical environment that created the window may not map the text-mode colors the way you expect. These problems are inherent in the monitor, video adapter, and driver software. They cannot be corrected using 4DOS, 4NT, or Take Command color specifications.

### ***Border Colors***

**4DOS** 4DOS can accept border colors in the CLS and COLOR commands, and in the StdColors directive in the .INI file. Border colors will be ignored, or will cause an error, if they are used elsewhere. Border colors do not work in windowed mode, and will be ignored if used in a window under Windows. They are not supported by many newer video adapters.

### ***Blinking Text and Bright Background Colors***

The interactions between blinking characters, bright backgrounds, and your display mode can be complex. You will need to understand them if you use either attribute in your color specifications.

The effects of blinking and bright background color specifications depend partly on whether you are in full-screen or windowed display mode.

#### ***Full-Screen Display Mode***

Full-screen display mode uses the entire screen for your command processor or application. This mode is the only one available in DOS; it is available as an option for text mode in Windows.

In full-screen display mode your video hardware can be configured via software commands to display either blinking text, or text with a bright background, but not both. This is due to the design of PC video hardware and is not a software restriction.

The memory on your video adapter includes a “flag” for each character on the screen. The flag controls blinking text and bright background colors. If the flag is off, the character is displayed with a normal background and does not blink. If the flag is “on,” the character either blinks or is displayed with a bright background, depending on which way your video adapter is currently configured.

#### ***4DOS***

In full-screen display mode, the configuration of your video adapter can be controlled by 4DOS with the BrightBG directive in the *4DOS.INI* file (see page 228) or the SETDOS /B command (page 473). If you set BrightBG = No or use the SETDOS /B0 command, 4DOS will configure the video adapter for blinking text, and all characters on the screen with the blink / bright background flag set will blink. If you set BrightBG = Yes or use SETDOS /B1, 4DOS will configure the video adapter for bright background colors, and the characters will be displayed with a bright background instead of blinking. If you don't use BrightBG or SETDOS /B, or you explicitly use a SETDOS /B2, 4DOS will not attempt to configure your video hardware. Most video adapters default to blinking text in full-screen mode, but this can be changed by application programs. If you use BrightBG, or SETDOS /B0 or /B1, 4DOS will configure the hardware each time it displays the prompt. BrightBG and

SETDOS /B are not available in 4NT and Take Command, because Windows always enables bright backgrounds and disables blinking text.

Because there is only one flag for each character to specify both blinking text and bright background color, it doesn't matter which attribute you use when you specify the color. Whether you specify blinking text or a bright background, you will see the same thing on your screen. For example, these two COLOR commands will always produce the same results:

```
color blink white on blue
color white on bright blue
```

If bright backgrounds are enabled, both commands will produce white text on a bright blue background. If blinking text is enabled, both commands will produce blinking white text on a blue background.

### ***Windowed Display Mode***

Windowed display mode uses a window on the screen for your command processor or text-mode application. It is available for the command processor and most applications running under Windows.

In windowed mode, the command processor cannot control your hardware to select blinking or bright backgrounds. Instead, Windows displays bright background colors, regardless of the BrightBG or SETDOS /B setting. It does not provide a way to display blinking characters in windowed mode. As an example, the two commands given above would both display white text on a bright blue background when run in windowed mode.

### ***Switching Modes***

Windows allows you to switch any DOS or other text-mode process between full-screen and windowed mode. For example, when running 4DOS or 4NT under Windows, you can switch modes by pressing **Alt-Enter**. Switching modes will usually alter color rendition, as you switch between direct interaction with the video board (in full-screen mode) and the color mapping provided by your graphical environment (in windowed mode). In addition, switching modes may alter the display of blinking text and bright background characters as described above.

**4DOS** Under 4DOS, if BrightBG is set to Yes, bright background colors will be preserved when you switch modes. However, if BrightBG is set to No, the display will shift from blinking text in the full-screen mode to bright background colors behind that text in the windowed mode. This effect can be

disturbing; you may need to take it into account if you write batch files or aliases which may be used in either mode.

### Keys and Key Names

Key names are used to define keystroke aliases, in several of the *.INI* file directives, and with the KEYSTACK command. The format of a key name is the same in all three uses:

[Prefix-]Keyname

The key prefix can be left out, or it can be one of the following:

Alt followed by A - Z, 0 - 9, F1 - F12, or Bksp

Ctrl followed by A - Z, F1 - F12, Bksp, Tab, Enter, Up, Down, Left, Right, PgUp, PgDn, Home, End, Ins, or Del

Shift followed by F1 - F12 or Tab.

The possible key names are:

A - Z	Tab	PgUp
0 - 9	Enter	PgDn
F1 - F12	Up	Home
Esc	Down	End
Left	Ins	Bksp
Right	Del	

All key names must be spelled as shown. Alphabetic keys can be specified in upper-case or lower-case. You cannot specify a punctuation key except by using its numeric value (see below).

The prefix and key name must be separated by a dash [-]. For example:

Ctrl-F10	This is okay
Ctrl F10	The space will cause an error

If you prefer, you can use a numeric value instead of a key name. Use the ASCII code for an ASCII, extended ASCII, or control character. Use the scan code preceded by an at sign [@] for extended key codes like **F1** or the cursor keys. For example, use 13 for **Enter**, or @59 for **F1**. In general, you will find it easier to use the names described above rather than key numbers. See the online help for an explanation and list of ASCII and key codes.

Some keys are intercepted by the operating system and are not passed on to 4DOS, 4NT, or Take Command. For example, under DOS **Ctrl-S** pauses

screen output temporarily, and on some systems **Ctrl-P** toggles Print Echo mode (where text displayed on the screen is automatically echoed to the printer). In Windows, Alt-Esc and Ctrl-Esc typically pop up a task list or are used in switching among multiple tasks. Keys which are intercepted by the operating system (including Windows menu accelerators, *i.e.* **Alt** plus another key) generally cannot be assigned to aliases or with key mapping directives, because the command processor never receives these keystrokes and therefore cannot act on them.

You also may not be able to use certain keys if your keyboard is not 100% PC-compatible, your keyboard driver does not support them, or, under DOS, if you have an ANSI driver which does not support them. For example, on some systems the **F11** and **F12** keys are not recognized; others may not support unusual combinations like **Ctrl-Tab**. These problems are rare; when they do occur, they are usually due to the operating system and/or your ANSI driver.

## ***Popup Windows***

Several features of 4DOS, 4NT, and Take Command display popup windows. A popup window may be used to display filenames, recently-executed commands, recently-used directories, the results of an extended directory search, or a list created by the **SELECT** command or the **@SELECT** internal function.

Popup windows always display a list of choices and a cursor bar. You can move the cursor bar inside the window until you find the choice that you wish to make, then press the **Enter** key to select that item.

Navigation inside any popup window follows the conventions described below. Additional information on each specific type of popup window is provided when that window is introduced, later in the manual.

You can control the color (in 4DOS and 4NT), position, and size of most popup windows from the “History” tab in the configuration dialog (in 4NT and Take Command), or with the **PopupWinLeft**, **PopupWinTop**, **PopupWinWidth**, and **PopupWinHeight** directives in the **.INI** file (see page 224). A few popup windows (*e.g.*, the extended directory search window) have their own specific **.INI** directives, and corresponding separate choices in the configuration dialog. You can also change the keys used in most popup windows with key mapping directives in the **.INI** file (see page 224).

Once a window is open, you can use these navigation keys to find the selection you wish to make:

<b>Up Arrow</b>	Move the selection bar up one line.
<b>Down Arrow</b>	Move the selection bar down one line.
<b>Left Arrow</b>	Scroll the display left (4 columns in 4DOS and 4NT; 1 column in Take Command if it is a scrolling display, <i>i.e.</i> if it has a horizontal scrollbar).
<b>Right Arrow</b>	Scroll the display right (4 columns in 4DOS and 4NT; 1 column in Take Command if it is a scrolling display, <i>i.e.</i> if it has a horizontal scrollbar).
<b>PgUp</b>	Scroll the display up one page.
<b>PgDn</b>	Scroll the display down one page.
<b>Ctrl-PgUp or Home</b>	Go to the beginning of the list.
<b>Ctrl-PgDn or End</b>	Go to the end of the list.
<b>Esc</b>	Close the window without making a selection.
<b>Enter</b>	Select the current item and close the window.

In addition to scrolling through a popup window, you can search the list using character matching. If you press a character, the cursor bar will move to the next entry that begins with that character. If you type multiple characters, the cursor will move to the entry that begins with the character string entered to that point (you can enter a search string up to 32 characters long). If no entry matches the character or string that you have typed, the command processor beeps and does not move the cursor bar. To reset the search string, press Backspace.

You can change the keys used in popup windows with key mapping directives in the *.INI* file (see page 230).

## CHAPTER 3 / THE TAKE COMMAND INTERFACE

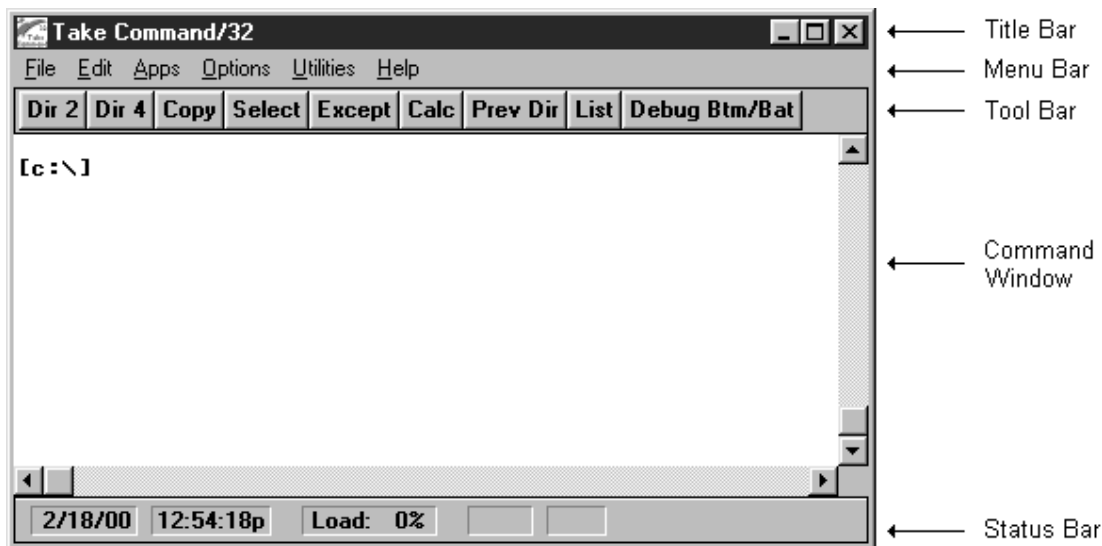
This chapter describes Take Command's Windows-related features, such as windows, menus, dialogs, and the tool bar and status bar. It does not apply to 4DOS or to 4NT (though 4NT's option dialogs are very similar to the Take Command configuration dialog described on page 50).

Most of the features described in this section are easy to use, but a few are more technical in nature. You may want to skip over any menu and dialog options which seem more technical than you need at the moment, and come back to them later as you start to use them.

Most of the features documented in this chapter are covered in equal or greater detail in the online help, in the section entitled "Using the Take Command Interface". In particular, this section includes basic information on most Take Command dialogs, but leaves more detailed descriptions about dialog fields and behavior to the online help system (each Take Command dialog has a **Help** button which will display the corresponding help information).

### *The Take Command Window*

The Take Command window has five parts:



The **Title Bar** is the same as the one used in most Windows applications, with a control menu button on the left and the maximize, minimize, and close buttons on the right. You can change the text that appears on the Title Bar,

and adjust the size of the Take Command window, with the **WINDOW** command (see page 530).

The **Menu Bar** and all of its menus are explained in detail starting on page 42.

The **Tool Bar** is used to execute internal or external commands, aliases, batch files, and applications with the click of a mouse. You can show or hide the Tool Bar with either a menu choice (see page 45) or with a configuration option (see page 50). You can define up to 32 Tool Bar buttons. See page 51 for instructions.

The **Command Window** accepts your input and displays Take Command's output. You can use the scroll bars or the **Up arrow** and **Down arrow** keys to view text that has scrolled through the window. You can also save the contents of the Command Window and scrollback buffer to a file, copy text from Command Window to the clipboard, and copy text from the clipboard or from the Command Window to the command line. See page 41 for information about saving and retrieving text in the Command Window and page 57 for complete details about using the Command Line.

Finally, the **Status Bar** at the bottom of the Take Command window displays information about your system:

- ✓ The date and time, based on the Windows clock.
- ✓ The percentage of CPU usage.
- ✓ The percentage “memory load” as reported by Windows.
- ✓ The state of the Caps Lock key on the keyboard.
- ✓ The state of the Num Lock key on the keyboard.

You can show or hide the Status Bar with either a menu choice (see page 45) or with a configuration option (see page 50).

If you find the “I-Beam” cursor in the Take Command window difficult to see, disable it from the **Startup** page of the configuration dialog, or set `IBeamCursor` to `No` in the `.INI` file (see page 212), to force the use of an arrow cursor in all parts of the window.



## ***Resizing the Take Command Window***

You can resize the Take Command window at any time with standard Windows techniques (*e.g.*, by dragging a corner with the mouse). Resizing the window changes the number of rows and columns of text which will fit in the command window (the actual number of rows and columns for any given window size depends on the font you are using). Take Command reacts to these changes using two sets of rules: one for the height and one for the width.

When the height of the command window changes, future commands simply use the new height as you see it on the screen. For example, if you reduce the window to three rows high and do a DIR /P (display a directory of files and pause at the bottom of each visual "page"), DIR will display two lines of output, a prompt ("Press any key to continue ..."), and then pause. If you expand the window to 40 lines high and repeat the same command, DIR will display 39 lines, a prompt, and then pause.

However, when the width of the window changes, Take Command may also adjust the current "virtual screen width". The virtual width is the maximum number of characters on each line in Take Command's internal screen buffer. You can think of it as the width of the data which can be displayed in the Take Command window, including an invisible portion to the right of the window's right-hand edge. When the virtual width is larger than the actual width, a standard horizontal scroll bar is displayed to allow you to see any hidden output.

The screen height normally starts at 25 lines; you can alter this default with the ScreenRows directive in the *.INI* file (see page 225), or the Height setting on the Display page of the configuration dialog. The `_ROWS` internal variable (see page 162) can be used to determine the current screen height.

The virtual screen width starts at 80 columns or the number of columns which fit into the startup Take Command window, whichever is larger. You can alter the default minimum width of 80 columns with the ScreenColumns directive (see page 225) in the *.INI* file, or the Width setting on the Display page of the configuration dialog. The `_COLUMNS` internal variable (see page 158) can be used to determine the current virtual screen width.

If you use keyboard commands or the mouse to expand the Take Command window beyond its previous virtual width, the virtual width is automatically increased. This ensures that the internal buffer can hold lines which will fill the newly enlarged window. If you contract the window, the virtual width is

not reduced because this might require removing output already on the screen or in the scrollback buffer.

As a result, widening the window will make future commands use the new enlarged size (for example, as the window is widened `DIR /W`, which displays a "wide" directory listing, will display additional columns of file names). However, if the window is narrowed future commands will still remember the enlarged virtual width, and display data to the right of the window edge. The horizontal scroll bar will make this data visible.

When the font is changed, Take Command will recalculate the virtual screen width. The new virtual width will be the width set by the Screen Columns directive or on the Display page of the configuration dialog, or the current width of the window in the new font, whichever is larger.

### ***Using the Scrollback Buffer***

Take Command retains the text displayed on its screen in a "scrollback buffer."

You can scroll through this buffer using the mouse and the vertical scroll bar at the right side of the Take Command window, just as you can in many Windows programs.

You can also use the **Up Arrow** [↑] and **Down Arrow** [↓] keys to scroll the display one line at a time from the keyboard, and the **PgUp** and **PgDn** keys to scroll one page at a time.

If you scroll back through the buffer to view previous output, and then enter text on the command line, Take Command will automatically return to the end of the buffer to display the text.

If you prefer to use the **Arrow** and **PgUp** keys to access the command history (as in 4DOS and 4NT), see the `SwapScrollKeys` .INI file directive (page 226), or the corresponding option on the History page of the configuration dialog (page 50). `SwapScrollKeys` switches the keystroke mapping so that the ↑, ↓, **PgUp**, and **PgDn** keys manipulate the command history, and **Ctrl-↑**, **Ctrl-↓**, **Ctrl-PgUp**, and **Ctrl-PgDn** are used to control the scrollback buffer. For more details see page 120.

You can set the size of the scrollback buffer on the Display page of the configuration dialog available from the Options menu, or with the `ScreenBufSize` .INI file directive.

To clear the entire scrollbar buffer, use the **CLS /C** command (see page 278).

## ***Highlighting and Copying Text***

While you are working at the Take Command prompt, you can use common Windows keystrokes to edit commands and use the clipboard to copy text between Take Command and other applications. You can also select all of the text in the Take Command screen buffer by using the Select All command on the Edit menu.

The right mouse button will pop up an “Edit” context menu.

To copy text from the Take Command window to the clipboard, first use the mouse to highlight the text, then press **Ctrl-C**, or use the Copy command on the Edit menu.

If you double-click on a word in the Take Command window, the entire word is highlighted or selected.

To highlight text on the command line, use the **Shift** key in conjunction with the **Left**, **Right**, **Ctrl-Left**, **Ctrl-Right**, **Home**, and **End** cursor keys. The **Del** key will delete any highlighted text on the command line, or you can type new text to replace the highlighted text.

While the Take Command window contains text, it is not a document window like those used by word processors and other similar software, and you cannot move the cursor throughout the window as you can in text processing programs. As a result, you cannot use the Windows shortcut keys like **Shift-Left** or **Shift-Right** to highlight text in the window. These keys work only at the command line; to highlight text elsewhere in the window you must use the mouse.

To copy text from the clipboard to the command line use **Shift-Ins**, or the Paste command on the Edit menu.

To paste text from elsewhere in the Take Command window directly onto the command line, highlight the text with the mouse and press **Ctrl-Shift-Ins**, or use the Copy+Paste command on the Edit menu. This is equivalent to highlighting the text and pressing **Ctrl-C** followed by **Ctrl-V**. It's a convenient way to copy a filename from a previous DIR or other command directly to the command line.

If you prefer, you can configure Take Command to use the “CUA” (Common User Access) keystrokes **Ctrl-Ins**, **Ctrl-Del**, and **Ctrl-Shift-Ins** for Copy,

Cut, and Paste on the Options 1 page of the configuration dialog (see page 50), or with the CUA *.INI* directive (see page 219).

You should use caution when pasting text containing carriage return or line feed characters onto the command line. If the text you insert contains one of these characters, the command line will be executed just as if you had pressed Enter. If you insert multiple lines, the text will be treated just like multiple lines of commands typed at the prompt.

### ***Using Drag and Drop***

Take Command is compatible with Windows' "Drag-and-Drop" facilities.

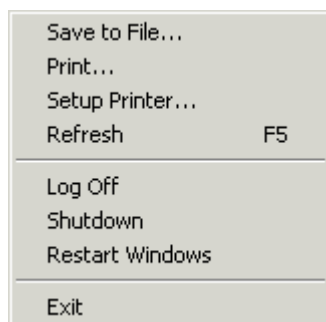
To add a filename to the command line using drag and drop, simply drag the file from another application using the mouse and release the mouse button with the file icon anywhere inside the Take Command window. The full name of the file will be pasted onto the command line at the current cursor position.

Take Command is a drag and drop "client," which means it can accept files dragged in from other applications and paste their names onto the command line as described above. It is not a drag and drop "server," so you cannot drag filenames from the Take Command window into other applications. However, you can copy filenames and other text from the Take Command screen to other applications using the clipboard; see above for details.

### ***Take Command Menus***

Like most Windows applications, Take Command displays a menu bar near the top of its window. To select a particular menu item, click once on the menu heading, or use **Alt-x** where "x" is the underlined letter on the menu bar (for example, **Alt-F** displays the File menu). You can also select a menu by pressing **Alt** or **F10** and then moving the highlight with the cursor keys.

## ***File Menu***



**Save to File** saves the contents of the Command Window and Scrollback Buffer to a file. A Save As dialog box appears in which you can enter the name of the file that you wish to use.

**Print** sends the contents of the Command Window and Scrollback Buffer to the printer. A Print dialog box appears in which you can choose the portion of the Screen Buffer you wish to print.

**Setup Printer** displays a standard printer setup dialog box. The options available in the dialog box depend on the printer driver(s) you are using.

**Refresh** redraws everything in the Take Command window (use this selection if the display appears incorrect, for example if it is not repainted properly after another application exits). You can also press F5 at the Take Command prompt to refresh the screen.

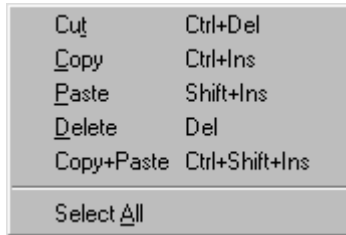
**Log Off** exits Take Command and logs off the current user in Windows NT / 2000 / XP. This choice will also restart the desktop in Windows 95 / 98 / ME.

**Shutdown** exits Take Command and shuts down Windows.

**Restart Windows** exits Take Command, shuts down Windows, and then reboots the system. This option only works in Windows NT / 2000 / XP, not in Windows 95 / 98 / ME.

**Exit** ends the current Take Command process.

### Edit Menu



**Cut** removes text from the Take Command command line and moves it to the clipboard.

**Copy** copies selected text from the Take Command command line or scrollback buffer to the clipboard.

**Paste** copies text from the clipboard to the Take Command command line. If the text you insert contains a line feed or carriage return, the command line will be executed just as if you had pressed Enter. If you insert multiple lines, each line will be treated like a command typed at the prompt.

**Delete** removes text from the Take Command command line.

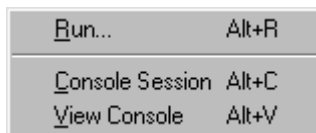
**Copy + Paste** copies the selected text from the Take Command scrollback buffer directly to the command line.

**Select All** marks the entire contents of the Take Command scrollback buffer as selected text.

To use the Cut, Copy, or Delete commands, you must first select a block of text with the mouse, the keyboard, or with the Select All command.

Note that you can also access the clipboard with redirection to or from the CLIP: device (see page 88), or with the @CLIP variable function (page 170).

### Apps Menu



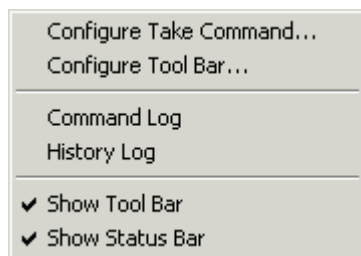
**Run** displays the run dialog box from which you can run an application or batch file (see page 48). Take Command remembers the commands you have run from this dialog in the current session.

To select from this list, click on the drop-down arrow to the right of the "Command Line" field or press the down-arrow.

**Console Session** starts a new console (text-mode) session, separate from Take Command, by running the default character-mode command processor.

**View Console** toggles the Take Command console window to visible or hidden. Take Command creates the console window to run character-mode applications. Most such applications can be run under Take Command's Caveman support, and will then display their output directly in the Take Command window. For complete details on character-mode applications, Caveman, and the console window, see **Console Applications and the Console Window** in the online help (under "Using the Take Command Interface"), or your *Introduction and Installation Guide*. You can also control whether the console window remains hidden after an application finishes; see the Startup page of the configuration dialog (page 50) or the HideConsole directive in *TCMD32.INI* (page 211).

## ***Options or Setup Menu***



**Configure Take Command** opens a series of dialogs which you can use to change the configuration of Take Command (see page 50).

**Configure Tool Bar** opens a dialog in which you can define up to 32 buttons for the Take Command tool bar (see page 51).

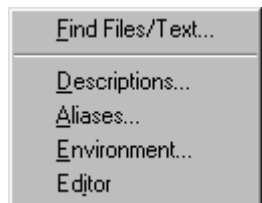
**Command Log** enables or disables command logging using the default log file (*TC32LOG*) or the file you have chosen with the LOG command or the LogName directive in the *.INI* file.

**History Log** enables or disables command history logging using the default log file (*TC32HLOG*) or the file you have chosen with the LOG /H command or the HistLogName directive in the *.INI* file.

**Show Tool Bar** enables or disables the Take Command tool bar (see page 51), which appears near the top of the Take Command window. The tool bar will not appear until you have defined at least one item for it with Configure Tool Bar, above.

**Show Status Bar** enables or disables the Take Command status bar, which appears near the bottom of the Take Command window.

### Utilities Menu



**Find Files/Text** opens the Find Files Dialog Box which lets you search for files or text interactively (see FFIND on page 341 to search from the command line).

Once Take Command has created a list of files based on your specifications, you can double-click on a file name and Take Command will display an information box about the file. From the information box, you can choose to list, edit, or run the file.

**Descriptions** opens an edit window in which you can view and change the descriptions of files in any directory available on your system. See DESCRIBE (page 299) for details on file descriptions.

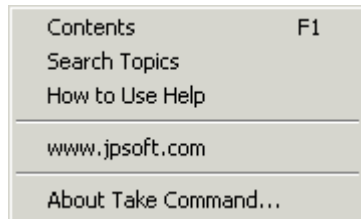
**Aliases** opens an edit window in which you can view and change the list of current aliases (see page 122 for more details on aliases). You can also use this window to import aliases from a file or save all current aliases in a file. Any changes you make will take effect as soon as you close the Aliases window.

**Environment** opens an edit window in which you can view and change the current environment. Any changes you make will be immediately recorded in Take Command's environment.



**Editor** starts the Windows Notepad editor, or any other editor you have specified with the Editor directive in the .INI file or on the Commands page of the configuration dialog.

## **Help Menu**



**Contents** displays the Table of Contents for Take Command Help.

**Search Topics** displays the Search dialog for Take Command Help. This is the same dialog you will see if you click on the **Search** button from within the help system.

**How to Use Help** displays the standard text that explains the Windows help system.

**jpsoft.com** is a hyperlink to our World Wide Web site. Clicking it will open your default browser if it isn't already running, and send a request to the browser to display our web site.

**About** displays Take Command version, copyright, and license information.

See page 55 for more information about how to use Take Command's online help.

## **Take Command Dialogs**

The Take Command menus lead to several dialog boxes. Each is described in this section.

The text below explains the purpose of each dialog, and, where appropriate, provides references within this manual for more information on the features associated with the dialog. For complete details on the fields in each Take Command dialog, see the online help under Using the Take Command Interface / The Take Command Screen / Dialogs, or use the **Help** button from within the dialog.

Take Command also uses standard Windows dialogs for tasks like printing, selecting a font, or browsing files and directories. These dialogs are provided by Windows, not Take Command, and are common to many different Windows programs; they are not documented in the Take Command help system.

### ***Save To File Dialog***

This standard Windows "Save As" dialog box is available from the File menu. From it, you can select the drive, directory, and file to use to save the contents of the command window and screen buffer.

### ***Print Dialog***

This standard Windows Print dialog is available from the File menu. You can select which part of the command window and scrollback buffer you want to print. You can choose settings such as the number of copies to print, whether you want collated copies, and the print quality. If you have already selected some text in the Take Command window, the dialog defaults to printing just the selected text. Otherwise, the default is to print the entire buffer.

### ***Printer Setup Dialog***

The options that appear in this dialog, which is available from the File menu and from the File/Print menu choice, depend on your printer and printer driver. The choices you make in this dialog affect future printing from Take Command and from your other applications.

### ***Run Program Dialog***

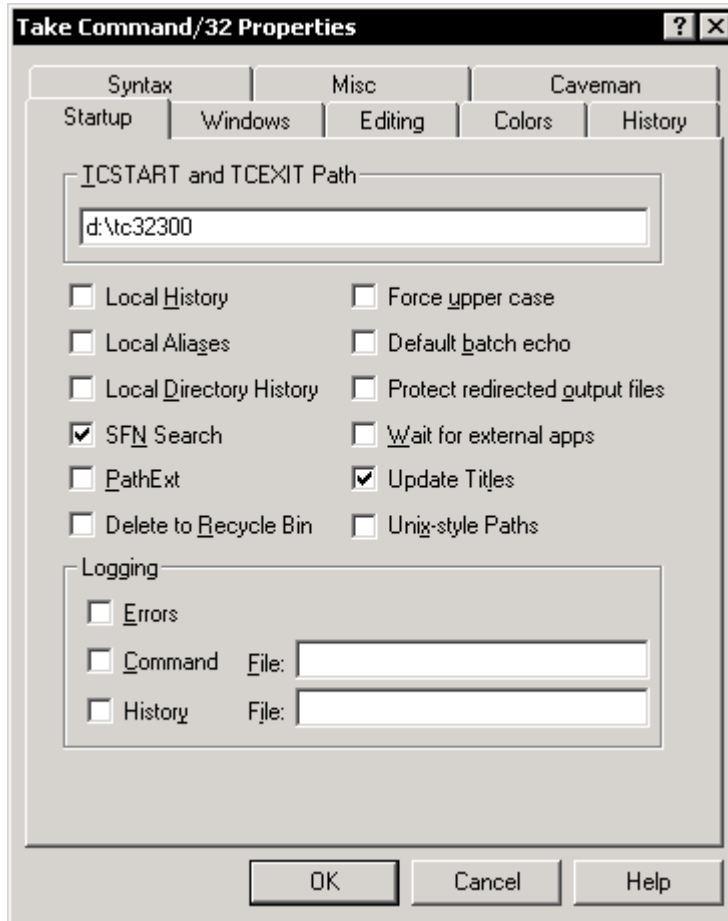


The Apps / Run menu leads to the Run Program dialog. In the Command Line edit box, you can enter the name of any executable program plus command-line parameters. If you click on the arrow to the right of the edit box, the dialog displays a list of previous commands you have entered during the current Take Command session.

The **Normal**, **Minimized**, and **Maximized** radio buttons determine the type of window that will be used for the program. If you select Minimized, the program will start as an icon on the Taskbar. Maximized starts the program in a full-screen window. The Normal button lets the operating system select the size and position of the program's window.

The **Browse** button leads to standard file browser from which you can select any executable program. Your choice will be placed in the Command Line edit box, and you can add parameters before selecting OK to run the program.

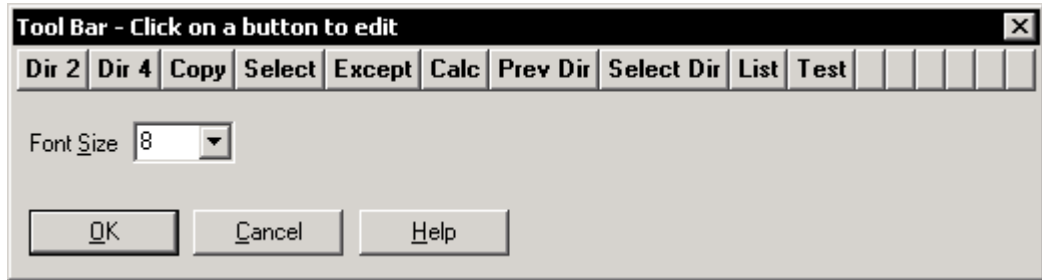
## **Configuration Dialog**



The Options / Configure Take Command menu choice leads to eight “pages” of dialogs that let you change the way Take Command looks and works.

For information about the purpose and choices for each item on each page, first select the tab at the top for the page you want, then click the Help button near the bottom of the dialogs. The online help also has cross references to information about other ways to set each option, either for the current session or for all Take Command sessions.

## *Tool Bar Dialog*



This dialog, available from the Options menu, allows you to define or modify buttons on the tool bar (the actual dialog has 32 buttons, but only 16 are shown above).

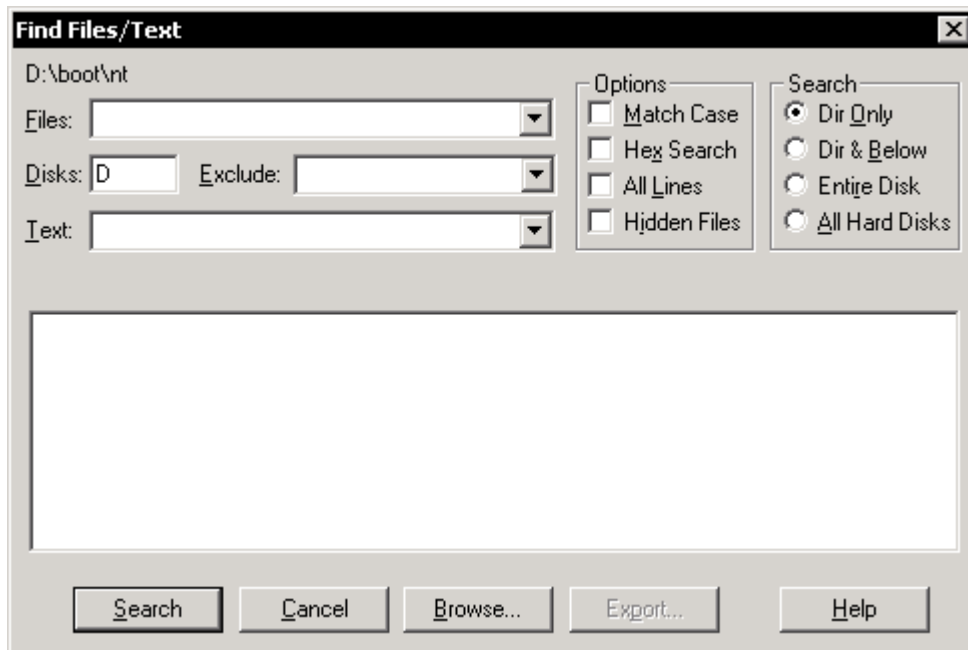
Select the button you want to define or modify by clicking on it. The Font Size setting applies to the display text for all buttons on the Take Command tool bar.

When you click on a button, a second dialog opens to let you define the label, command, directory, and mode for the button. You can use the **Browse** button to find a path and filename to be entered at the beginning of the Command field.

You can also control whether the command is executed immediately, and whether it is echoed before it is executed.

See the online help for additional details on the meaning of each field and button, and for information on starting programs in the correct working directory.

## **Find Files / Text Dialog**



The Find Files/Text dialog box gives you the same features as the FFIND command, in dialog form. It is available from the Utilities menu.

Enter the file name or names you wish to search for in the **Files** field. You can use wildcards and include lists (see page 94 and 104) as part of the file name. You can also use the **Browse** button to find specific files to examine for a text search.

Enter the drive(s) you want to search in the **Disks** field. This field is ignored unless **Entire Disk** is selected in the **Search** portion of the dialog. If you select **All Hard Disks**, this field is set automatically to include all hard disk drive letters Take Command finds on your system.

If you use wildcards to specify the files to search, you can narrow the search with the **Exclude** field by specifying files that you want to exclude from the search. Like the **Files** field, the **Exclude** field can contain wildcards and include lists (see page 94 and 104). For example, if you want to search all files with an extension beginning with "I" except for **.ICO** and **.INI** files, you could enter "\*.i\*" in the **Files** field and then "\*.ico;\*.ini" in the **Exclude** field.

Enter the text (or hexadecimal values) you are searching for in the **Text** field. You can use extended wildcards (see page 94) in the

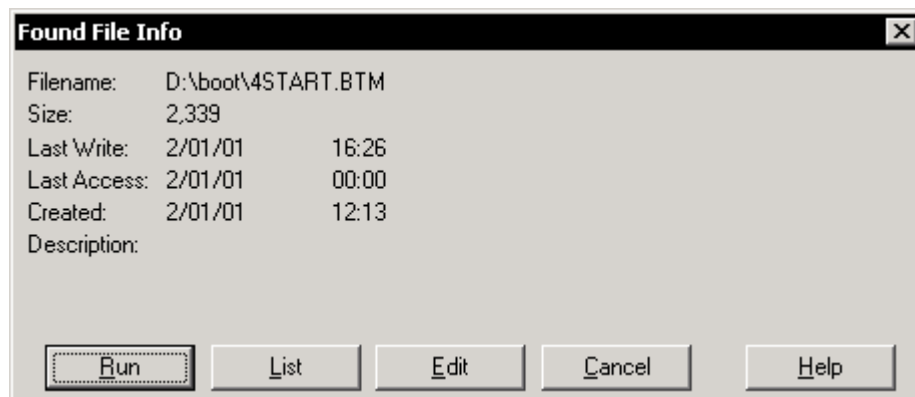
search string to increase the flexibility of the search. Use back-quotes [ ] around the text if you want to search for characters which would otherwise be interpreted as wildcards.

The **Match Case** box, when it is selected, makes the search case-sensitive. This option is ignored if **Hex Search** is selected. The **Hex Search** option signals that you are searching for hexadecimal values, not ASCII characters; see the online help for additional details.

If you enable **All Lines**, every matching line from every file will be displayed; otherwise only the first matching line from each file will be displayed. Unless you enable the **Hidden Files** option, files with the hidden and system attributes (see page 18) will not be included in the search. The radio buttons in the **Search** area let you specify where you want Take Command to look for files.

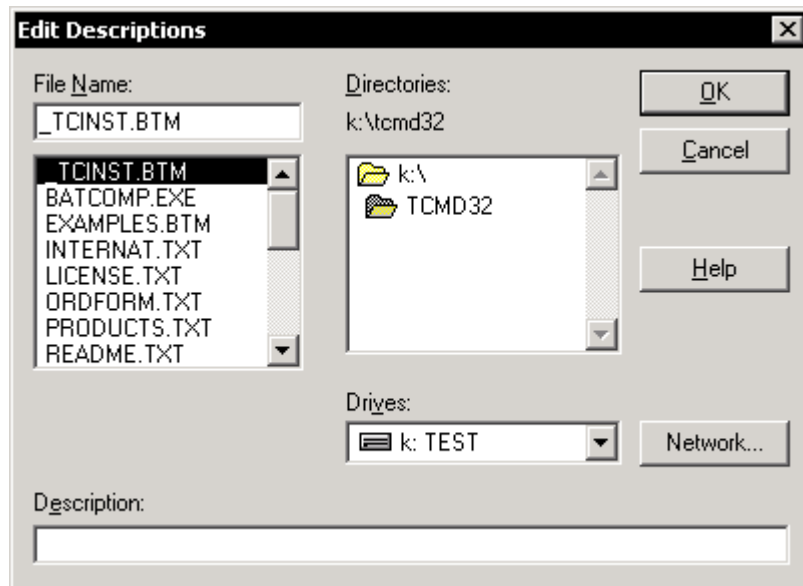
To start the search, press the **Find** button. Once the search has started the **Find** button changes to a **Stop** button, which you can use to interrupt the search before it is finished.

If you select one of the matching files in the list (by double-clicking on it, or selecting it with the cursor and pressing Enter), Take Command will display another dialog with complete directory information about the file:



From this dialog, you can **Run** the file (if it is an executable file, a batch file, or has an executable extension), display the file with the **LIST** command (see page 401), or **Edit** the file. **Edit** uses the Windows Notepad editor, or any other editor you have specified with the Editor directive in the *.INI* file or on the Commands page of the configuration dialog.

## ***Descriptions Dialog***

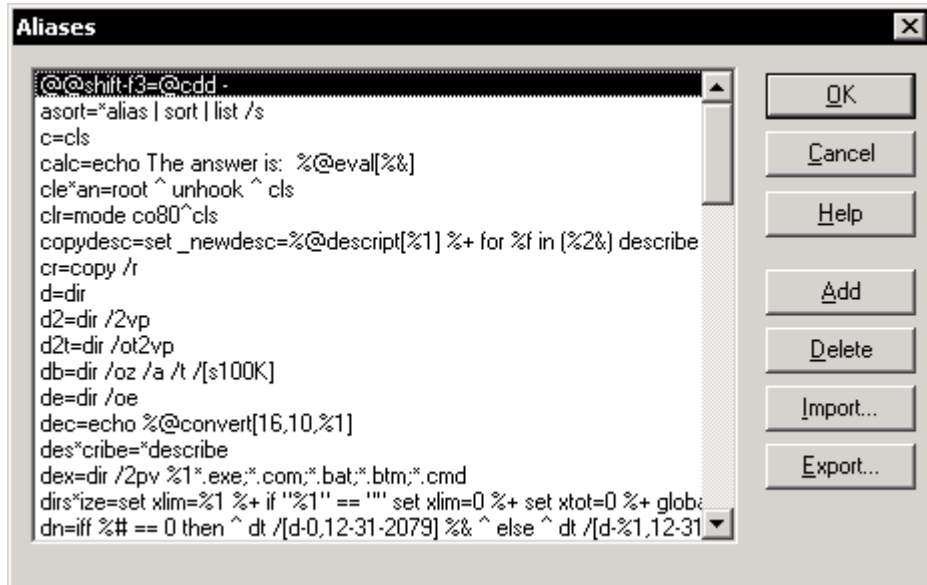


The Edit Descriptions dialog is available from the Descriptions item on the Utilities menu. Most of this dialog looks and works like a standard file browser. The pane at the bottom of the dialog lets you view, enter, or edit the description for any file. See DESCRIBE on page 299 for more information on file descriptions.

File descriptions can also be entered or changed with the DESCRIBE command, and are visible when you use the DIR and SELECT commands.



## Alias and Environment Dialogs



You can use the Aliases and Environment dialogs, available from the Utilities menu, to view, edit, add, and delete Take Command aliases and environment variables. (The two dialogs are the same except for the title and the data which appears within the fields. The Environment dialog is shown above.) For more information about aliases, see page 122; for more information about environment variables, see page 148.

The current list of aliases or environment variables is shown in the pane. You can double-click on any line of the display to edit that alias definition or environment variable. To add a new alias or variable to the list, use the **Add** button. The **Delete** button deletes the highlighted alias or environment variable.

The **Import** button reads a list of aliases or environment variables from a file; the **Export** button writes the current list to a file.

## Take Command Help

The online help system for Take Command covers all Take Command features and internal commands. It includes reference information to assist you in using Take Command and developing batch files, and it includes most — but not all — of the details which are included in the printed manuals. You can start the help system by pressing **F1**, by selecting the Help menu, or by using the techniques below.

If you type part or all of a command on the command line and then press **F1**, the help system will provide "context-sensitive" help by using the first word on the line as a help topic. If it's a valid topic, you will see help for that topic automatically; if not, you will see the Table of Contents for the help file, and you can pick the topic you want. You can also select help for the topic at the cursor (or immediately to the left if the cursor is at the end of a word) by pressing **Ctrl-F1**.

You can also display help for a topic by typing **HELP** followed by the topic name. You can use this feature to obtain help on any topic — not just on commands. For example, if you enter the command **HELP \_DISK** you will see help for the **\_DISK** internal variable.

Take Command uses the Windows help system to display help text. Once you've started the help system with **HELP** or **F1**, you can use standard keystrokes to navigate through the help files. For more information on using Windows help press **F1** again from within the help system, or select **How To Use Help** from the Take Command Help menu.

If you type the name of any internal command at the prompt, followed by a slash and a question mark [/?] like this:

```
copy /?
```

then you will see "quick help" for the command.

Finally, if you use a command incorrectly, omit a required parameter, or use an unrecognized option, Take Command will display a syntax summary for the command.

## **CHAPTER 4 / Using 4DOS, 4NT, and Take Command**

4DOS, 4NT, and Take Command are both a collection of commands and a set of features which make your computer easier to use. The commands are explained in the Command Reference section which begins on page 241. This chapter explains each of the features that are not directly related to individual commands.

Most of the features described in this section are easy to use, but a few are more technical in nature. Such advanced features are marked with a ♦ next to the feature name or the paragraph which describes the feature's operation.

As you read through this section, we urge you to experiment with the features that catch your interest and pass over any which seem too complicated. Come back to this section as you gain expertise, and you will probably discover that the more complex features will seem easy and very useful. You don't need to learn any more than you want, and even if you are a computer novice, you'll find some features that will interest you immediately.

If you come across terms or concepts in this chapter that you are unsure about, refer to Chapter 1 / General Concepts, the Index, or the Glossary in the online help system. If you have any questions related to Take Command's "GUI" interface, refer to the previous chapter, which covers that topic in detail.

### ***At the Command Line***

4DOS displays a `c:\>` prompt when it is waiting for you to enter a command. 4NT and Take Command display the similar `[c:\]` prompt. (In each case, the actual text depends on the current drive and directory as well as your PROMPT settings.) This is called the command line and the prompt is asking you to enter a command, an alias or batch file name, or the instructions necessary to begin an application program.

This section of the manual explains the features that will help you while you are typing in commands, and how keystrokes are interpreted when you enter them at the command line. The keystrokes discussed here are the ones normally used by 4DOS, 4NT, and Take Command. If you prefer using different keystrokes to perform these functions, you can assign new ones with key mapping directives in the `.INI` file (see page 203).

## **Command-Line Editing**

The command line works like a single-line word processor, allowing you to edit any part of the command at any time before you press **Enter** to execute it, or **Esc** to erase it.

The command line you enter can be up to 511 characters long under 4DOS, and 2047 characters long under 4NT and Take Command.

You can use the following editing keys when you are typing a command (the words **Ctrl** and **Shift** mean to press the Ctrl or Shift key together with the other key named):

### ***Cursor Movement:***

<b>←</b>	Move the cursor left one character.
<b>→</b>	Move the cursor right one character.
<b>Ctrl-←</b>	Move the cursor left one word.
<b>Ctrl-→</b>	Move the cursor right one word.
<b>Home</b>	Move the cursor to the beginning of the line.
<b>End</b>	Move the cursor to the end of the line.

### ***Insert and Delete:***

<b>Ins</b>	Toggle between insert and overstrike mode.
<b>Del</b>	Delete the character at the cursor, or the highlighted text.
<b>Backspace</b>	Delete the character to the left of the cursor or the highlighted text.
<b>Ctrl-L</b>	Delete the word or partial word to the left of the cursor.
<b>Ctrl-R or Ctrl-Bksp</b>	Delete the word or partial word to the right of the cursor.
<b>Ctrl-V</b>	Paste the first line from the clipboard to the cursor position.
<b>Ctrl-Home</b>	Delete from the beginning of the line to the cursor.
<b>Ctrl-End</b>	Delete from the cursor to the end of the line.
<b>Esc</b>	Delete the entire line.

<i>TC</i>	<b>Shift-Ins</b>	Insert the text from the clipboard at the current cursor position on the command line.
<i>TC</i>	<b>Ctrl-Shift-Ins</b>	Insert the highlighted text (from anywhere in the window) at the current cursor position on the command line.

***Execution:***

<b>Ctrl-C or Ctrl-Break</b>	Cancel the command line.
<b>Enter</b>	Execute the command line.

***Marking Text:***

<b>Shift-←</b>	Mark the character to the left.
<b>Shift-→</b>	Mark the character to the right.
<b>Shift-Home</b>	Mark from the beginning of the line to the cursor.
<b>Shift-End</b>	Mark from the cursor to the end of the line.
<b>Ctl-Shift-←</b>	Mark the word to the left.
<b>Ctl-Shift-→</b>	Mark the word to the right.
<b>Ctrl-Y</b>	Copy the marked text to the clipboard.

***Miscellaneous:***

<b>Ctrl-F5</b>	Toggles batch debug mode.
<b>Alt-PgUp, Alt-PgDn, Alt-Home, Alt-End, Alt-Up, Alt-↓</b>	Scroll the 4NT window within the console buffer. (Use the cursor pad keys, not the numeric keypad keys.)

To highlight text on the command line, use the keys under Marking Text. Once you have selected or highlighted text on the command line, any new text you type will replace the highlighted text. If you press **Bksp** or **Del** while there is text highlighted on the command line, the highlighted text will be deleted. While you are working at the prompt, you can use the clipboard to copy text between the command processor and other applications (see page 41 for additional details). Take Command can also use Drag and Drop to paste a filename from another application onto the command line (see page 42 for additional details).

**4DOS** To mark text with 4DOS, you must either have an ANSI driver loaded or set the StdColors directive in 4DOS.INI.

Most of the command-line editing capabilities are also available when 4DOS, 4NT, or Take Command prompts you for a line of input. For example, you can use the command-line editing keys when DESCRIBE (page 299) prompts for a file description, when INPUT (page 388) prompts for input from an alias or batch file, or when LIST (page 401) prompts for a search string.

If you want your input at the command line to be in a different color from the command processor's prompts or output, you can use the “Colors” tab of the configuration dialog, or the InputColors directive in your .INI file (option dialogs are available from the Option menu in Take Command or by typing OPTION at the 4DOS or 4NT prompt). See page 229 for more details. You **must** have an ANSI driver installed to use InputColors under 4DOS.

The command processor will prompt for additional command-line text when you include the escape character as the very last character of a typed command line. The default escape character is Ctrl-X (ASCII 24, displayed as an up arrow [↑]) in 4DOS, and a caret [^] in 4NT and Take Command. For example:

```
[c:\] echo The quick brown fox jumped over the^
More? lazy sleeping dog. > alphabet
```

- ❖ Sometimes you may want to enter one of the command line editing keystrokes on the command line, instead of performing the key's usual action. For example, suppose you have a program that requires a Ctrl-R character on its command line. Normally you couldn't type this keystroke at the prompt, because it would be interpreted as a “Delete word right” command.

To get around this problem, use the special keystroke **Alt-255**. You enter Alt-255 by holding down the **Alt** key while you type **0255** on the numeric keypad, then releasing the **Alt** key. (You must use the number keys on the numeric pad; the row of keys at the top of your keyboard won't work. Also, in Take Command the leading **0** before the **255** is required.) This forces the command processor to interpret the next keystroke literally and place it on the command line, ignoring any special meaning it would normally have as a command-line editing or history keystroke. You can use Alt-255 to suppress the normal meaning of command-line editing keystrokes even if they have been reassigned with key mapping directives in the .INI file (see page 203), and Alt-255 itself can be reassigned with the CommandEscape directive.

## **Command History and Recall**

Each time you execute a command, the entire command line is saved in a **command history list**. You can display the saved commands, search the list, modify commands, and rerun commands. The command history is available at the command prompt and in a special command history window.

The simplest use of the command history list is to repeat a command exactly. For example, you might enter the command

```
[c:\] dir a:*.wks;*.doc
```

to see some of the files on drive A. You might move some new files to drive A and then want to repeat the DIR command. Just press the ↓Up Arrow key (in 4DOS or 4NT) or **Ctrl-Up** (in Take Command) repeatedly to scan back through the history list. When the DIR command appears, press **Enter** to execute it again. (You can also view the command history in a window — see page 64 for details.)

After you have found a command, you can edit it before pressing **Enter**. You will appreciate this feature when you have to execute a series of commands that differ only slightly from each other.

The history list is normally “circular”. If you move to the last command in the list and then press ↓ (4DOS / 4NT) or **Ctrl-↓** (Take Command) one more time, you'll see the first command in the list. Similarly, if you move to the first command in the list and then press ↑ (4DOS / 4NT) or **Ctrl-↑** (Take Command) one more time, you'll see the last command in the list. You can disable this feature and make command history recall stop at the beginning or end of the list by turning off History Wrap on the “History” tab of the configuration dialog, or setting HistWrap to No in the *.INI* file (see page 223).

You can search the command history list to find a previous command quickly using **command completion**. Just enter the first few characters of the command you want to find and press ↑ (4DOS / 4NT) or **Ctrl-↑** (**Take Command**). You only need to enter enough characters to identify the command that you want to find. For example, if you're using Take Command and want to find a DIR command, enter DI and then press **Ctrl-↑**. If you press **Ctrl-↑** key a second time, you will see the previous command that matches. The system will beep if there are no matching commands. The search process stops as soon as you type one of the editing keys, whether or not the line is changed. At that point, the line you're viewing becomes the new line to match if you press **Ctrl-↑** again.

You can specify the size of the command history list on the “History” tab of the configuration dialog, or with the History directive in the *.INI* file (see page 212). When the list is full, the oldest commands are discarded to make room for new ones. You can also use the HistMin directive in the *.INI* file to enable or disable history saves and to specify the shortest command line that will be saved (see page 222).

You can prevent any command line from being saved in the history by beginning it with an at sign [*@*].

When you execute a command from the history, that command remains in the history list in its original position. The command is not copied to the end of the list (unless you modify it). If you want each command to be copied or moved to the end of the list when it is re-executed, set HistCopy or HistMove to Yes in your *.INI* file (see page 222) or select Copy to End or Move to End on the “History” tab of the configuration dialog. If you select either of these options, the list entry identified as “current” (the entry from which commands are retrieved when you press **Ctrl-↑**) is also adjusted to refer to the end of the history list after each recalled command is executed. You can also have duplicates automatically removed from the history list with the HistDups directive.

### **Command History Keys:**

<b>↑</b> <b>Ctrl-↑</b>	(4DOS, 4NT) (Take Command)  Recall the previous (or most recent) command, or the most recent command that matches a partial command line.
<b>↓</b> <b>Ctrl-↓</b>	(4DOS, 4NT) (Take Command)  Recall the next (or oldest) command, or the oldest command that matches a partial command line.
<b>F3</b>	Fill in the rest of the command line from the previous command, beginning at the current cursor position.
<b>Ctrl-D</b>	Delete the currently displayed history list entry, erase the command line, and display the previous (matching) history list entry.
<b>Ctrl-E</b>	Display the last entry in the history list.



<b>Ctrl-K</b>	Save the current command line in the history list without executing it, and then clear the command line.
<b>Ctrl-Enter</b>	Copy the current command line to the end of the history list even it has not been altered, then execute it.
<b>@</b>	As the first character in a line: Do not save the current line in the history list when it is executed, or store it in the CMDLINE environment variable (see CMDLINE on page 151 for further details).

Use **F3** when your new command is different from your previous one by just a character or two at the beginning. For example, suppose you want to execute a DIR on several file names then use DEL to delete those same files. After the DIR is complete type DEL and press **F3**; the rest of the command line will be completed for you. Check that it's correct, and then press **Enter** to delete the files. **F3** also retrieves the entire previous command (like ↑ / **Ctrl**-↑) if nothing has been typed on the line.

Use **Ctrl-E** to “get your bearings” by returning to the end of the list if you've scrolled around so much that you aren't sure where you are any more.

Use **Ctrl-K** to save some work when you've typed a long command and then realize that you weren't quite ready. For example, if you forget to change directories and notice it after a command is typed or mostly typed, but before you press **Enter**, just press **Ctrl-K** to save the command without executing it. Use the CD or CDD command to change to the right directory, press ↑ / **Ctrl**-↑ twice to retrieve the command you saved, make any final changes to it, and press **Enter** to execute it.

Use **Ctrl-Enter** to organize the history list for repetitive tasks. Instead of searching through the command history for the next command in a sequence, you can place all of the necessary commands next to each other and make them easier to repeat.

If you prefer to use the arrow keys to access the command history in Take Command (as in 4DOS), without having to press **Ctrl**, see the SwapScrollKeys .INI directive, or the corresponding option on the “History” tab of the configuration dialog. SwapScrollKeys switches the Take Command keystroke mapping so that the ↑, ↓, **PgUp**, and **PgDn** keys manipulate the command history, and **Ctrl**-↑, **Ctrl**-↓, **Ctrl-PgUp**, and **Ctrl-PgDn** are used

to control the scrollbar buffer. For more details see Scrolling and History Keystrokes on page 120.

## ***Command History Window***

You can also view the command history in a scrollable **command history window**, and select the command to modify or re-execute from those displayed in the window. To activate the command history window press **PgUp** or **PgDn** (in 4DOS and 4NT) or **Ctrl-PgUp** or **Ctrl-PgDn** (in Take Command) at the command line. A window will appear with the command you most recently executed marked with a highlight. (If you just finished re-executing a command from the history, then the next command in sequence will be highlighted.)

See page 35 for general information about moving through the list in the window. The display is not circular as it is at the prompt; it has a fixed beginning and end. The **Ctrl-D** (delete from history) key works within the history window as it does at the command line.

Once you have selected a command in the history window, press **Enter** or double-click on it to execute it immediately. Press **Ctrl-Enter** or hold down **Ctrl** and double-click on the line to move it to the prompt for editing (you cannot edit the line directly in the history window).

You can view a “filtered” history window by typing some characters on the command line, then pressing **PgUp** or **PgDn** or **Ctrl-PgUp** or **Ctrl-PgDn**. Only those commands matching the typed characters will be displayed in the window.

### ***Command History Window Keys:***

(See page 35 for general popup window keys, and information on customizing window position, size, and color.)

<b>PgUp</b> or <b>PgDn</b>	(4DOS, 4NT)
<b>Ctrl PgUp</b> or <b>Ctrl-PgDn</b>	(Take Command)
	Open the command history window.
<b>Ctrl-D</b>	Delete the selected line from the history list.
<b>Enter</b> or <b>Double Click</b>	Execute the selected line.
<b>Ctrl-Enter</b> or <b>Ctrl Double Click</b>	Move the selected line to the command line for editing.

If you prefer to use the **PgUp** key to access the command history in Take Command, without having to press **Ctrl**, see the **SwapScrollKeys** directive on page 226, or the corresponding option on the History page of the configuration dialog (see page 50).

## ***Local and Global Command History***

The command history can be stored in either a “local” or “global” list.

With a local command history list, any changes made to the history will only affect the current command processor window. A local command history list is the default under 4DOS.

Whenever you start a secondary shell (see page 9) which uses a local history list, it inherits a copy of the command history from the previous shell. However, any changes to the history made in the secondary shell will affect only that shell.

With a global command history list, all copies of the command processor will share the same command history, and any changes made to the history in one session will affect all other copies. Global lists are the default for 4NT and Take Command.

You can control the type of history list from the “Startup” tab of the configuration dialog, with the **LocalHistory** directive in the **.INI** file (see page 213), with the **/L** and **/LH** startup options (see your *Introduction and Installation Guide*), and with the **/L** and **/LH** options of the **START** command (see page 488).

**4DOS**     4DOS can share a global command history list among a parent 4DOS shell and any child shells you start from that parent shell. For example, if you use 4DOS under DOS, start an application, and then “shell to DOS” from the application, the original copy of 4DOS and the child shell can share the global command history. If you want to share a global command history list among all copies of 4DOS in Windows 95 / 98, you **must** load a parent copy of 4DOS as the primary command processor, before starting Windows.

**4NT, TC**   ❖ If you select a global history list for 4NT or Take Command, you can share the history among all active copies of the command processor. When you close all 4NT or Take Command processes, the memory for the global history list is released, and a new, empty history list is created the next time you start 4NT or Take Command. If you want the history list to be retained in memory even when no command processor is running, you need to execute the **SHRALIAS** command, which loads a program to perform this service for

the global alias, function, command history and directory history lists (see page 485).

There is no fixed rule for deciding whether to use a local or global history list. Depending on your work style, you may find it most convenient to use one type, or a mixture of types in different shells. We recommend that you start with the default approach for your command processor, then modify it if you find a situation where the default is not convenient.

### **Command Names and Parameters**

When you enter a command you type its name at the prompt, followed by a space and any parameters for the command. For example, all of these could be valid commands:

```
c:\> dir
c:\> copy file1 file2 d:\
c:\> f:\util\mapmem /v
c:\> "c:\Program Files\jpsoft\take command\tcmd32.exe" /l
```

The last three commands above include both a command name, and one or more parameters. There are no spaces within the command name (except in quoted file names), but there is a space between the command name and any parameters, and there are spaces between the parameters.

Some commands may work when parameters are entered directly after the command (without an intervening space, *e.g.* **dir/p**), or when several parameters are entered without spaces between them (*e.g.* **dir /2/p**). A very few older programs may even require this approach. However leaving out spaces in this way is usually technically incorrect, and is not recommended as a general practice, as it may not work for all commands.

If the command name includes a path, the elements must be separated with backslashes (*e.g.* **F:\UTIL\MAPMEM**). If you are accustomed to Unix syntax where forward slashes are used in command paths, and want the command processor to recognize this approach, you can set **UnixPaths** to **Yes** in the **.INI** file or using the **OPTION** command or configuration dialog. If you set **UnixPaths**, you must have spaces between all of your parameters.

For more information on command entry see **Multiple Commands** on page 76 and **Command Line Length Limits** on page 78. For details on how the command processor handles the various elements it finds on the command line see **Command Parsing** on page 197.

## **Filename Completion**

Filename completion can help you by filling in a complete file name on the command line when you only remember or want to type part of the name. For example, suppose you want to copy a file. You know that its name begins *AU* but you can't remember the rest of the name. Type:

```
[c:\] copy au
```

and then press the **Tab** key or **F9** key. The command processor will search the current directory for filenames that begin *AU* and insert the first one onto the command line in place of the *AU* that you typed.

If this is the file that you want, simply complete the command. If the command processor didn't find the file that you were looking for, press **Tab** or **F9** again to substitute the next filename that begins with *AU* (file names are displayed in the physical order in which they occur in the disk directory). When there are no more filenames that match your pattern, the command processor will beep each time you press **Tab** or **F9**.

If you go past the filename that you want, press **Shift-Tab** or **F8** to back up and return to the previous matching filename. After you back up to the first filename, the system will beep each time you press **Shift-Tab** or **F8**.

**4DOS, 4NT** If you want to enter more than one matching filename on the same command line, press **F10** when each desired name appears. This will keep the previously displayed name and place the next matching filename after it on the command line. You can then continue to use **Tab** (or **F9**), **Shift-Tab** (or **F8**), and **F10** to move through the remaining matching files. Use **F10** each time you want to continue with add additional names. To repeat the previous filename use **F12**.

**TC** To enter more than one matching filename on the same command line in Take Command follow the procedure described above for 4DOS and 4NT, but use **Ctrl-Shift-Tab** or **F11** (rather than **F10**) when each desired name appears. To repeat the previous filename use **F12**.

If you want to select from a scrollable list of matching filenames, you can press the **F7** key instead of **Tab** for the initial match. A popup window will appear with all of the matching filenames. Select the desired filename and press **Enter** to insert it on the command line.

The pattern you use for matching may contain any valid filename characters, as well as wildcard characters and extended wildcards (see page 94). For example, you can copy the first matching *.TXT* file by typing

```
[c:\] copy *.txt
```

and then pressing **Tab**.

If you don't specify part of a filename before pressing **Tab**, the command processor will match all files. For example, if you enter the above command as "COPY ", without the "\*.TXT", and then press **Tab**, the first filename in the current directory is displayed. Each time you press **Tab** or **F9** after that, another name from the current directory is displayed, until all filenames have been displayed.

- ❖ If you type a filename without an extension, the command processor will add \*.\* to the name (\* on LFN drives). It will also place a "\*" after a partial extension. If you are typing a group of file names in an include list (see page 104), the part of the include list at the cursor will be used as the pattern to match.

When filename completion is used at the start of the command line, it will only match directories, executable files (see page 19), and files with executable extensions, since these are the only file names that it makes sense to use at the start of a command. If a directory is found, a "\" will be appended to it to enable an automatic directory change (see page 72). If you need to complete the name of any other file at the start of the command line, press **Space** before starting to type the name. Filename completion will then match any name, not just directory and executable names.

**4NT, TC** 4NT and Take Command also support network server and sharename completion. If the filename begins with \\, the completion routines will enumerate the network resources for matching server and/or share names. You can control the way server name completion functions with the ServerCompletion directive in the *.INI* file (see page 225). Be sure to review the ServerCompletion options if you experience performance problems with server name completion on large networks.

### **Converting Between Long and Short Filenames**

On LFN drives, the command processor will search for and display long filenames during filename completion. If you want to search for traditional 8.3 short filenames, press **Ctrl-A** before you start using filename completion.

This allows you to use filename completion on LFN drives with applications that do not support long filenames.

You can press **Ctrl-A** at any time prior to beginning filename completion. The switch to short filename format remains in effect for the remainder of the current command line. When the command processor begins a new command line it will return to long filename format until you press **Ctrl-A** again.

You can also press **Ctrl-A** just after a filename is displayed, and the name will be converted to short filename format. However, this feature only affects the most recently entered file or directory name (the part between the cursor and the last backslash [\] on the command line), and any subsequent entries. It will not automatically convert all the parts of a previously entered path.

**Ctrl-A** “toggles” the filename completion mode, so you can switch back and forth between long and short filename displays by pressing **Ctrl-A** each time you want to change modes.

### ❖ *Appending Slashes to Directory Names*

If you set the `AppendToDir .INI` directive (see page 218), or the corresponding option on the “Editing” tab of the configuration dialog, the command processor will add a trailing backslash [\] to all directory names (or a trailing slash [/] to directory names in FTP URLs). (If you have the `UnixPaths .INI` directive set, the command processor will add a trailing forward slash [/] to directory names instead of the backslash.) This feature can be especially handy if you use filename completion to specify files that are not in the current directory -- a succession of **Tab** (or **F9**) and **F10** (or, in Take Command, **F11 / Ctrl-Shift-Tab**) keystrokes can build a complete path to the file you want to work with.

The following example shows the use of this technique to edit the file `C:\DATA\FINANCE\MAPS.DAT`. The lines which include “<F9>” show where F9 (or Tab) is pressed; the other lines show how the command line appears after the previous F9 or Tab (the example is displayed on several lines here, but all appears at a single command prompt when you actually perform the steps):

```
1 [c:\] edit \da <F9>
2 [c:\] edit \data\
3 [c:\] edit \data\f <F9>
4 [c:\] edit \data\frank.doc <F9>
5 [c:\] edit \data\finance\
6 [c:\] edit \data\finance\map <F9>
```

```
7 [c:\] edit \data\finance\maps.dat
```

Note that F9 was pressed twice in succession on lines 3 and 4, because the file name displayed on line 3 was not what was needed — we were looking for the FINANCE directory, which came up the second time F9 was pressed. In this example, filename completion saves about half the keystrokes that would be required to type the name in full. If you are using long file or directory names, the savings can be much greater.

### ***Filename Completion Keys:***

<b>F8 or Shift-Tab</b>	<b>Get the previous matching filename.</b>
<b>F9 or Tab</b>	Get the next matching filename.
<b>F10</b>	(4DOS / 4NT) Keep the current matching filename and display the next matching name immediately after the current one.
<b>Ctrl-Shift-Tab or F11</b>	(Take Command) Keep the current matching filename and display the next matching name immediately after the current one.
<b>F12</b>	Repeat the filename just returned from an F9 / Tab match.
<b>Ctrl-Tab or F7</b>	Popup a scrollable window with all matching filenames.
<b>Ctrl-A</b>	On LFN drives, toggle between long filename and short filename format.

### ❖ ***Customizing Filename Completion***

You can customize filename completion for any internal or external command or alias. This allows the command processor to display filenames intelligently based on the command you are entering. For example, you might want to see only *.TXT* files when you use filename completion in the EDIT command.

To customize filename completion you can use the “Editing” tab of the configuration dialog, or set the FileCompletion directive manually in your *.INI* file. You can also use the FILECOMPLETION environment variable. If you use both, the environment variable will override the settings made in the dialog or the *.INI* file. You may find it useful to use the environment variable for experimenting, then create permanent settings with the configuration dialog or the FileCompletion directive.



The format for both the environment variable and the *.INI* file is:

```
cmd1:ext1 ext2 ...; cmd2: ...
```

where “cmd” is a command name and “ext” is a file extension (which may include wildcards) or one of the following file types:

DIRS	Directories
RONLY	Read-only files
HIDDEN	Hidden files
SYSTEM	System files
ARCHIVE	Files modified since the last backup
FILES	Everything that’s not a directory

Prefacing “ext” with a ! means “don’t match this extension”. For example, to make file completion for the EDIT command return all files with extensions starting with *.TX*, except *.TXZ*, you could use:

```
FileCompletion=edit:!txz tx?
```

The command name is the internal command, alias, or executable file name (without a path). For example, to have file completion return only directories for the CD command and only *.C* and *.ASM* files for a Windows editor called WinEdit, you would use this setting for filename completion:

```
FileCompletion=cd:dirs; winedit:c asm
```

To set the same values using the environment variable, you would use this line:

```
[c:\] set filecompletion=cd:dirs; winedit:c asm
```

With this setting in effect, if you type “CD ” and then press **Tab**, the command processor will return only directories, not files. If you type “WINEDIT ” and press **Tab**, you will see only names of *.C* and *.ASM* files.

When testing to see if customized filename completion should be used, 4DOS, 4NT, and Take Command check the actual command line you type, without expanding any aliases. For example, if you use the FileCompletion setting shown above and have “W” aliased to “WINEDIT,” and then enter a “W” command, the FileCompletion setting – which refers only to “WINEDIT” – will be ignored. To use customized filename completion for aliases you must enter the alias name in the FileCompletion setting:

```
FileCompletion=cd:dirs; winedit:c asm; w:c asm
```

## ***Filename Completion Window***

You can also view filenames in a **filename completion window** and select the file you want to work with. To activate the window, press **F7** or **Ctrl-Tab** at the command line. You will see a window in the upper-right corner of the screen, with a sorted list of files that match any partial filename you have entered on the command line. If you haven't yet entered a file name, the window will contain the name of all files in the current directory. You can search for a name by typing the first few characters; see page 35 for details.

(**Ctrl-Tab** will work only if your keyboard and BIOS or keyboard driver support it. If it does not work on your system, use **F7** instead.)

***Filename Completion Window*** (see page 35 for general popup window keys, and information on customizing window position, size, and color):

- |                              |  |
|------------------------------|--|
| <b>F7 or Ctrl-Tab</b>        | (from the command line) Open the filename completion window. |
| <b>Enter or Double Click</b> | Insert the selected filename into the command line.          |

## ***Variable Name Completion***

Variable name completion works like filename completion. If the text at the cursor begins with a %, the completion routines will scan the environment for matching variable names. For example, if the **PROMPT** and **PATH** variables are in the environment, in that order, this sequence might be used to display the **PATH**:

```
[c:\] echo %p<Tab>
[c:\] echo %PROMPT<Tab>
[c:\] echo %PATH<Enter>
```

## ***Automatic Directory Changes***

[Automatic directory changes are part of a set of comprehensive directory navigation features built into 4DOS, 4NT, and Take Command. For a summary of these features, and more information on the Extended Directory Searches and **CDPATH** features mentioned below, see the Directory Navigation section beginning on page 81.]

The automatic directory change feature lets you change directories quickly from the command prompt, without entering an explicit **CD** or **CDD**

command. To do so, simply type the name of the directory you want to change to at the prompt, with a backslash [\] at the end. For example:

```
[c:\] tcmd\  
[c:\tcmd]
```

This can make directory changes very simple when it is combined with Extended Directory Searches or CDPATH. If you have enabled either of those features, the command processor will use them in searching for any directory you change to with an automatic directory change (see Directory Navigation on page 81 for more information on CDPATH and Extended Directory Searches).

For example, suppose Extended Directory Searches are enabled, and the directory WIN exists on drive E:. You can change to this directory with a single word on the command line:

```
[c:\4dos] win\  
[e:\win]
```

(Depending on the way Extended Directory Changes are configured, and the number of subdirectories on your disk whose names contain the string *WIN*, when you execute such a command you may see an immediate change as shown above, or a popup window which contains a list of subdirectories named *WIN* to choose from.)

The text before the backslash can include a drive letter, a full path, a partial path, or (with 4NT and Take Command) a UNC name (see page 13). Commands like "...\" can be used to move up the directory tree quickly (see page 94). Automatic directory changes save the current directory, so it can be recalled with a "CDD -" or "CD -" command. For example, any of the following are valid automatic directory change entries:

```
[c:\] d:\data\finance\  
[c:\] archives\  
[c:\] ...\util\win95\  
[c:\] \\server\vol1\george\
```

The first and last examples change to the named directory. The second changes to the *ARCHIVES* subdirectory of the current directory, and the third changes to the *UTIL\WIN95* subdirectory of the directory which is two levels "up" from the current directory in the tree.

## ***Directory History Window***

[The directory history window is part of a set of comprehensive directory navigation features built into 4DOS, 4NT, and Take Command. For a summary of these features, and more information on enhanced directory navigation features, see the Directory Navigation section beginning on page 81.]

The current directory is recorded automatically in the **directory history list** just before each change to a new directory or drive.

You can view the directory history from a **directory history window** and change to any drive and directory on the list. To activate the directory history window, press **F6** (Take Command) or **Ctrl-PgUp** (4DOS and 4NT) at the command line. You can then select a new directory with the **Enter** key.

If the directory history list becomes full, old entries are deleted to make room for new ones. You can set the size of the list from the “History” tab of the configuration dialog, or with the DirHistory directive in the *.INI* file (see page 210). In order to conserve space, each directory name is recorded just once in the directory history, even if you move into and out of that directory several times. The directory history can be stored in either a “local” or “global” list; see the next section for details.

When you switch directories the original directory is saved in the directory history list, regardless of whether you change directories at the command line, from within a batch file, or from within an alias. However, directory changes made by external directory navigation utilities or other external programs are not recorded by 4DOS, 4NT, or Take Command.

***Directory History Window*** (see page 35 for general popup window keys, and information on customizing window position, size, and color):

<b>F6</b>	(from the command line) Open the directory history window in Take Command.
<b>Ctrl-PgUp</b>	(from the command line) Open the directory history window in 4DOS and 4NT.
<b>Ctrl-D</b>	Delete the selected line from the directory list.
<b>Enter</b>	Change to the selected drive and directory.
<b>Ctrl-Enter</b>	Move the selected line to the command line for editing.

## ***Local and Global Directory History***

The directory history can be stored in either a “local” or “global” list.

With a local directory history list, any changes made to the list will only affect the current copy of the command processor. They will not be visible in other shells. A local directory history list is the default under 4DOS.

Whenever you start a secondary shell (see page 9) which uses a local history list, it inherits a copy of the directory history from the previous shell. However, any changes to the history made in the secondary shell will affect only that shell.

With a global list, all copies of the command processor will share the same directory history, and any changes made to the list in one copy will affect all other copies. Global lists are the default for 4NT and Take Command.

You can control the type of history list from the “Startup” tab of the configuration dialog, with the LocalDirHistory directive in the *.INI* file (see page 213), with the */L* and */LD* startup options (see your *Introduction and Installation Guide*), and with the */L* and */LD* options of the *START* command (see page 488).

- 4DOS** 4DOS can share a global directory history list among a parent 4DOS shell and any child shells you start from that parent shell. For example, if you use 4DOS under DOS, start an application, and then “shell to DOS” from the application, the original copy of 4DOS and the child shell can share the global directory history. If you want to share a global directory history list among all copies of 4DOS in Windows, you **must** load a parent copy of 4DOS, usually as the primary command processor, before starting Windows. Under 4DOS, you can control where a global directory history list is stored with the *UMBDirHistory* directive in the *.INI* file (see page 216), or the corresponding setting available from the *OPTION* command.
- 4NT, TC** ❖ If you select a global directory list for 4NT or Take Command, you can share the list among all copies of the command processor. When you close all 4NT or Take Command processes, the memory for the global directory history list is released, and a new, empty list is created the next time you start 4NT or Take Command. If you want the list to be retained in memory even when no command processor is running, you need to execute the *SHRALIAS* command, which loads a program to perform this service for the global alias, function, command history and directory history lists (see page 485).

There is no fixed rule for deciding whether to use a local or global directory history list. Depending on your work style, you may find it most convenient to use one type, or a mixture of types in different shells. We recommend that you start with the default setting for your command processor, then modify it if you find a situation where the default is not convenient.

### ***Multiple Commands***

At times, you probably know the next two or three commands that you want to execute. Instead of waiting for each one to finish before you type the next, you can type them all on the same command line, separated by a caret [^] in 4DOS, or an ampersand [&] in 4NT and Take Command. For example, if you know you want to copy all of your *.TXT* files to drive A: and then run CHKDSK to be sure that drive A's file structure is in good shape, in 4NT or Take Command you could enter the following command:

```
[c:\] copy *.txt a: & chkdsk a:
```

You may put as many commands on the command line as you wish, as long as the total length of the command line does not exceed 511 characters in 4DOS or 2,047 characters in 4NT and Take Command.

You can use multiple commands in alias definitions (see page 122) and batch files (see page 125) as well as from the command line.

If you don't like using the default command separator, you can pick another character using the SETDOS /C command (see page 473), the CommandSep directive in the *.INI* file (see page 219), or on the "Syntax" tab of the configuration dialog. If you plan to share aliases or batch files between 4DOS, 4NT and Take Command, see page 195 for details about choosing compatible command separators for two or more products.

### **❖ *Expanding and Disabling Aliases***

A few command line options are specifically related to aliases, and are documented briefly here for completeness. If you are not familiar with aliases, see pages 122 and 250 for complete detail.

You can expand an alias on the command line and view or edit the results by pressing **Ctrl-F** before the command is executed. Doing so is especially useful when you are developing and debugging a complex alias or if you want to make sure that an alias that you may have forgotten won't change the intent of your command.

At times, you may want to temporarily disable an alias that you have defined. To do so, precede the command with an asterisk [\*]. For example, if you have an alias for DIR which changes the display format, you can use the following command to bypass the alias and display the directory in the standard format:

```
[c:\] *dir
```

## **Command Line Help**

All of our products include complete online help. You can start the help system at the command line by entering **HELP** or **HELP** plus a topic, or by pressing the **F1** key at any time.

If you have already typed part or all of a command on the line, the help system will provide “context-sensitive” help by using the first word on the line as a help topic. If it's a valid topic, you will see help for that topic automatically; if not, you will see a table of contents and you can then pick the topic you want. For example, if you press **F1** after entering each of the command lines shown below you will get the display indicated:

[c:\]	Topic list / table of contents
[c:\] copy *.* a:	Help on COPY
[c:\] c:\util\map	Topic list / table of contents

If you press **Ctrl-F1**, you will get help for the command at the cursor (or immediately to the left if the cursor is at the end of a word).

For quick help you can type the name of any internal command at the prompt, followed by a slash and a question mark [/?] like this:

```
[c:\] copy /?
```

This will show you help for the command in a “quick-reference” style (the output can be redirected; see page 88 for information on redirection).

/? will only access the help system when you use it with an internal command. If you use it with an external command name, the external command will be executed and will interpret the /? parameter according to its own rules. Some external commands do display help when run with a /? parameter, but this is a characteristic of these commands and does not depend on the command processor. Many external commands do not have this feature.

Please see the *Introduction and Installation Guide* for additional information on the online help system.

### ❖ **Command-Line Length Limits**

Under 4DOS, when you first enter a command at the prompt or in an alias or batch file, it can be up to 511 characters long. The 4NT and Take Command command lines can hold 2,047 characters.

As the command processor scans the command line and substitutes the contents of aliases and environment variables for their names, the line usually gets longer. This expanded line is stored in an internal buffer which allows each line to grow to 511 characters (in 4DOS) or 4,095 characters (in 4NT and Take Command) during the expansion process. If your use of aliases or environment variables causes the command line to exceed either of these limits as it is expanded, you will see a “Command line too long” error and the remainder of the line will not be executed. This can occur due to simple expansion of aliases and variables, or due to a “loop” where an alias or variable refers back to itself.

## **Starting Applications**

4DOS, 4NT, and Take Command offers several ways to start applications.

First, you can simply type the name of any application at the prompt. As long as the application's executable file is in one of the standard search directories (see below), the command processor will find it and start it. If you type the full path name of the executable file at the prompt the application will be started even if it is not in one of the standard search directories.

4DOS, 4NT, and Take Command offer two methods to simplify and speed up access to your applications. One is to create an alias (see page 250), for example:

```
[c:\] alias myapp d:\apps\myapp.exe
```

**TC**

In Take Command you can also use the Tool Bar to start frequently-used applications. For example, a tool bar button named **MyApp** which invokes the command **d:\apps\myapp.exe** would accomplish the same thing as the alias shown above.

You can use these methods together. For example, if you define the alias shown above you can set up a tool bar button called **MyApp** and simply use



the command **myapp** for the button, which would then invoke the previously-defined alias.

In 4NT and Take Command, you can also start an application by typing the name of a data file associated with the application. The command processor will examine the file's extension and run the appropriate application, based on Windows file associations (see page 112) or executable extensions (see page 106). If there is no executable extension for the file and you want Windows to start the appropriate application based on a Windows file association then you must type the full name with extension; if you type only the name, Windows will not be able to start the application for you, and you will get an "Unknown command" error.

For additional flexibility, you can also start applications with the **START** command. **START** provides a number of switches to customize the way an application is started.

For additional information on waiting for applications to finish see page 114.

### ***Searching for Applications***

When you start an application without specifying a path, the command processor searches for the application in the current directory, and then all directories on the **PATH**. Take Command also searches the Windows and Windows system directories; see the **PATH** command on page 429 for details. (If you do enter an explicit path, the command processor will only look in the directory you specified.)

If you enter a file name with no extension, 4DOS will search each directory for a matching **.COM**, **.EXE**, **.BTM**, or **.BAT** file, then for an executable extension. 4NT and Take Command will search each directory for a matching **.PIF**, **.COM**, **.EXE**, **.BTM**, **.BAT**, or **.CMD** file (and **.REX** and/or **.REXX** if a REXX interpreter is loaded), then for a file matching a Windows file association or executable extension. If no such file is found, the command processor will move on to the next directory in the search sequence.

See page 19 for additional details on the search sequence.

### ***TC Application Windows***

In most cases, Take Command starts each application in its own window. When the application exits, the window is closed.

However, Take Command runs DOS and Windows 32-bit character mode applications in a “console window” associated with the Take Command process. This window is opened automatically, and remains open as long as Take Command is running. The window is visible whenever an application is actually running within it. After an application exits, you can switch back to the console window and view the output with the **Alt-V** key, or the View Console Window selection on the **Apps** menu (see page 44). Under Windows 95 / 98 / ME, if you execute a *.PIF* file, the application will be opened in its own window. Applications run under Take Command's Caveman feature will run in an invisible Caveman console window, but their output will be displayed in the Take Command window as well. For more details, see your *Introduction and Installation Guide* or the online help. Additional information on Caveman is available in the online help.

### 4NT, TC *Page and File Prompts*

Several 4DOS, 4NT, and Take Command commands can generate prompts, which wait for you to press a key to view a new page or to perform a file activity.

When 4DOS, 4NT, or Take Command is displaying information in page mode, for example with a **DIR /P** or **SET /P** command, it displays the message

```
Press Esc to Quit or any other key to continue...
```

At this prompt, you can press **Esc**, or **Ctrl-Break** if you want to quit the command. You can press almost any other key to continue with the command and see the next page of information.

During file processing, if you have activated prompting with a command like **DEL /P**, you will see this prompt before processing every file:

```
Y/N/R ?
```

You can answer this prompt by pressing **Y** for “Yes, process this file;” **N** for “No, do not process this file;” or **R** for “process the **R**emainder of the files without further prompting.” You can also press **Esc**, **Ctrl-C**, or **Ctrl-Break** at this prompt to cancel the remainder of the command.

If you press **Ctrl-C** or **Ctrl-Break** while a batch file is running, you will see a “Cancel batch job” prompt. For information on responses to this prompt see page 131.

## ***Directory Navigation***

The operating system and command processor remember both a **current** or **default drive** for your system as a whole, and a **current** or **default directory** for every drive in your system. The current directory on the current drive is sometimes called the **current working directory**.

With traditional command processors, you change the current drive by typing the new drive letter plus a colon at the prompt, and you change the current working directory with the CD command. 4DOS, 4NT, and Take Command support those standard features, and offer a number of enhancements to make directory navigation much simpler and faster.

This section begins with a summary of all 4DOS, 4NT, and Take Command directory navigation features. It also provides detailed documentation on the enhanced directory search features: Extended Directory Searches (page 83) and the CDPATH (page 86).

The directory navigation features are in three groups: features which help the command processor find the directory you want, methods for initiating a directory change with a minimal amount of typing, and methods for returning easily to directories you've recently used. Each group is summarized below.

### ***Finding Directories***

Traditional command processors require you to explicitly type the name of the directory you want to change to. 4DOS, 4NT, and Take Command support this method, and also offer two significant enhancements:

- ✓ **Extended Directory Searches** (see page 72) allow the command processor to search a “database” of all the directories on your system to find the one you want.
- ✓ The **CDPATH** (see page 71) allows you to enter a specific list of directories to be searched, rather than searching a database. Use CDPATH instead of Extended Directory Searches if you find the extended searches too broad, or your hard drive has too many directories for an efficient search.

### ***Initiating a Directory Change***

4DOS, 4NT, and Take Command support the traditional methods of changing directories, and also offer several more flexible approaches:

- ✓ **Automatic directory changes** (see page 72) allow you to type a directory name at the prompt and switch to it automatically, without typing an explicit CD or similar command.
- ✓ The **CD command** (see page 271) can change directories on a single drive, and can return to the most recently used directory.
- ✓ The **CDD command** (see page 274) changes drive and directory at the same time, and can return to the most recently used drive and directory.
- ✓ The **PUSHD command** (see page 441) changes the drive and directory like CDD, and records the previous directory in a directory “stack.” You can view the stack with DIRS and return to the directory on the top of the stack with POPD (see the next section).

**4NT, TC** CDD, PUSHD, and automatic directory changes can also change to network drives and directories mapped to drive letters and to ones specified with UNC names (see page 13 for information on UNC names).

### ***Returning to a Previous Directory***

Traditional command processors do not remember previously-used directories, and can only “return” to a directory by changing back to it with a standard drive change or CD command. 4DOS, 4NT, and Take Command support three additional methods for returning to a previous directory:

- ✓ The **CD** - and **CDD** - commands (see pages 271 and 274) can be used to return to the previous working directory (the one you used immediately before the current directory). Use these commands if you are working in two directories and alternating between them.
- ✓ The **directory history window** (see page 74) allows you to select one of several recently-used directories from a popup list and return to it immediately. The window displays the contents of the directory history list (see the next section for details).
- ✓ The **POPD command** (see page 435) will return to the last directory saved by PUSHD (see the previous section). The directory stack holds 511 characters, enough for 20 to 40 typical drive and directory entries.

## ***Extended Directory Searches***

When you change directories with an automatic directory change, CD, CDD, or PUSH command, 4DOS, 4NT, or Take Command must find the directory you want to change to. To do so, the command processor first uses the traditional method to find a new directory: it checks to see whether you have specified either the name of an existing subdirectory below the current directory, or the name of an existing directory with a full path or a drive letter. If you have, the command processor changes to that directory, and does no further searching.

This traditional search method requires that you navigate manually through the directory tree, and type the entire name of each directory you want to change to. Extended Directory Searches speed up the navigation process dramatically by allowing the command processor to find the directory you want, even if you only enter a small part of its name.

When the traditional search method fails, the command processor tries to find the directory you requested via the CDPATH, then via an Extended Directory Search. This section covers only Extended Directory Searches, which are more flexible and more commonly used than CDPATH; for details on CDPATH see the next section.

Extended Directory Searches use a database of directory names to facilitate changing to the correct directory. The database is used only if Extended Directory Searches are enabled, and if the explicit directory search and CDPATH search fail to find the directory you requested.

An extended directory search automatically finds the correct path to the requested directory and changes to it if that directory exists in your directory database. If more than one directory in the database matches the name you have typed, a popup window appears and you can choose the directory you want.

You can control the color, position and size of the popup directory search window from the “History” tab of the configuration dialog, or with the CDDWinLeft, CDDWinTop, CDDWinWidth, and CDDWinHeight directives in the .INI file (see page 218). You can also change the keys used in the popup window with key mapping directives in the .INI file (see page 224).

To use extended directory searches, you must explicitly enable them (see below) and also create the directory database.

### ***The Extended Search Database***

To create or update the database of directory names, use the **CDD /S** command (see page 274 for details). When you create the database with **CDD /S**, you can specify which drives and directories should be included. If you enable Extended Directory Searches and do not create the database, it will be created automatically the first time it is required, and will include all local hard drives.

The database is stored in the file *JPSTREE.IDX*, which is placed in the root directory of drive C: by default. The same tree file is used by all JP Software command processors. You can specify a different location for this file on the “Misc” tab of the configuration dialog, or with the *TreePath .INI* directive (see page 215). If you are using two or more of our products on your computer and want to have different drives stored in the database for each, use the dialogs or the *TreePath* directive to place their database directories in different locations.

If you use an internal command to create or delete a directory, the directory database is automatically updated to reflect the change to your directory structure. The updates occur if the command processor can find the *JPSTREE.IDX* file in the root directory of drive C: or in the location specified by the *TreePath .INI* directive.

The internal commands which can modify the directory structure and cause automatic updates of the file are **MD**, **RD**, **COPY /S**, **DEL /X**, **MOVE /S**, and **REN**. The **MD /N** command can be used to create a directory without updating the directory database. This is useful when creating a temporary directory which you do not want to appear in the database.

### ***Enabling Extended Searches***

To enable extended directory searches and control their operation, you must set the **FuzzyCD** directive in the *.INI* file. You can set **FuzzyCD** either from the “Misc” tab of the configuration dialog, or by editing the *.INI* file manually.

If **FuzzyCD = 0**, extended searches are disabled, the *JPSTREE* database is ignored, and **CD**, **CDD**, **PUSHD**, and automatic directory changes search for directories using only explicit names and **CDPATH**. This is the default.

If **FuzzyCD = 1** and an extended search is required, then the command processor will search the *JPSTREE* database for directory names which **exactly match** the name you specified.

If **FuzzyCD = 2** and an extended search is required, the command processor will search the database for exact matches first, just as when **FuzzyCD = 1**. If the requested directory is not found, it will search the database a second time looking for directory names that **begin with** the name you specified.

If **FuzzyCD = 3** and an extended search is required, the command processor will search the database for exact matches first, just as when **FuzzyCD = 1**. If the requested directory is not found, it will search the database a second time looking for directory names that **contain** the name you specified anywhere within them.

An extended directory search is not used if you specify a full directory path (one beginning with a backslash [\], or a drive letter and a backslash). If you use a name which begins with a drive letter (*e.g.* **C:MYDIR**), the extended search will examine only directories on that drive.

#### ❖ ***Forcing an Extended Search with Wildcards***

Normally you type a specific directory name for the command processor to locate, and the search proceeds as described in the preceding sections. However, you can also force the command processor to perform an extended directory search by using wildcard characters (see page 94) in the directory name. If you use a wildcard, an extended search will occur whether or not extended searches have been enabled.

When 4DOS, 4NT, or Take Command is changing directories and it finds wildcards in the directory name, it skips the explicit search and CDPATH steps and goes directly to the extended search.

If a single match is found, the change is made immediately. If more than one match is found, a popup window is displayed with all matching directories.

Wildcards can only be used in the final directory name in the path (after the last backslash in the path name). For example you can find **COMM\\*A\*.\*** (all directories whose parent directory is COMM and which have an A somewhere in their names), but you cannot find **CO?M\\*A\*.\*** because it uses a wildcard before the last part of the name.

If you use wildcards in the directory name as described here, and the extended directory search database does not exist, it will be built automatically the first time a wildcard is used. You can update the database at any time with **CDD /S**.

Internally, extended directory searches use wildcards to scan the directory database. If FuzzyCD is set to 2, an extended search looks for the name you typed followed by an asterisk (*i.e.* *DIRNAME\**). If FuzzyCD is set to 3, it looks for the name preceded and followed by an asterisk (*i.e.* *\*DIRNAME\**).

These internal wildcards will be used in addition to any wildcards you use in the name. For example if you search for *ABC?DEF* (ABC followed by any character followed by DEF) and FuzzyCD is set to 3, the command processor will actually search the directory database for *\*ABC?DEF\**.

### ❖ **CDPATH**

When you change directories with an automatic directory change or the CD, CDD, or PUSHD command, 4DOS, 4NT, or Take Command must find the directory you want to change to. To do so, the command processor first uses the traditional method to find a new directory.

When the traditional search method fails, the command processor tries to find the directory you requested via the CDPATH, then via an Extended Directory Search. This section covers only the CDPATH; for details on Extended Directory Searches see the previous section.

Enabling both CDPATH and Extended Directory Searches can yield confusing results, so we recommend that you do not use both features at the same time. If you prefer to explicitly list where the command processor should look for directories, use CDPATH. If you prefer to have the command processor look at all of the directory names on your disk, use Extended Directory Searches.

CDPATH is an environment variable, and is similar to the PATH variable used to search for executable files: it contains an explicit list of directories to search when attempting to find a new directory. The command processor appends the specified directory name to each directory in CDPATH and attempts to change to that drive and directory. It stops when it finds a match or when it reaches the end of the CDPATH list.

CDPATH is ignored if a complete directory name (one beginning with a backslash [\\]) is specified, or if a drive letter is included in the name. It is only used when a name is given with no drive letter or leading backslash.

CDPATH provides a quick way to find commonly used subdirectories in an explicit list of locations. You can create CDPATH with the SET command. The format of CDPATH is the same as PATH: a list of directories separated by semicolons [;]. For example, if you want the directory change commands



to search the C:\DATA directory, the D:\SOFTWARE directory, and the root directory of drive E:\ for the subdirectories that you name, you should create CDPATH with this command:

```
[c:\] set cdpath=c:\data;d:\software;e:\
```

Suppose you are currently in the directory C:\WP\LETTERS\JANUARY, and you'd like to change to D:\SOFTWARE\UTIL. You could change directories explicitly with the command:

```
[c:\wp\letters\january] cdd d:\software\util
```

However, because the *D:\SOFTWARE* directory is listed in your CDPATH variable as shown in the previous example (we'll assume it is the first directory in the list with a UTIL subdirectory), you can simply enter the command

```
[c:\wp\letters\january] cdd util
```

or, using an automatic directory change:

```
[c:\wp\letters\january] util\
```

to change to D:\SOFTWARE\UTIL.

As it handles this request, the command processor looks first in the current directory, and attempts to find the *C:\WP\LETTERS\JANUARY\UTIL* subdirectory. Then it looks at CDPATH, and appends the name you entered, *UTIL*, to each entry in the CDPATH variable — in other words, it tries to change to *C:\DATA\UTIL*, then to *D:\SOFTWARE\UTIL*. Because this change succeeds, the search stops and the directory change is complete.

## ***Input and Output***

Internal commands and some external character-mode programs get their input from the computer's **standard input** device and send their output to the **standard output** device. Some programs also send special messages to the **standard error** device. Normally, the keyboard is used for standard input and the video screen for both standard output and standard error, but you can temporarily change these assignments for special tasks.

For example, suppose you want a printed list of the files in a directory. If you change the standard output to the printer and issue a DIR command, the task is easy. DIR's output goes to the standard output device, and you have redirected standard output to the printer, so the DIR command prints

filenames instead of displaying them on the screen. You can just as easily send the output of DIR (or any other command) to a file or a serial port.

We offer three methods of manipulating input and output: Redirection, Piping, and the Keystack. All three are explained in this section.

Redirection and piping affect the standard input, standard output, and standard error devices. They do not work with application programs which read the keyboard hardware directly, or which write directly to the screen. Because most Windows applications fall into that category, you will find that redirection and piping most useful when they are combined with internal commands.

### ***Redirection***

Redirection can be used to reassign the **standard input**, **standard output**, and **standard error** devices from their default settings (the keyboard and screen) to another device like the printer or serial port, to a file, or to the clipboard. You must use some discretion when you use redirection with a device; there is no way to get input from the printer, for example.

Redirection always applies to a specific command, and lasts only for the duration of that command. When the command is finished, the assignments for standard input, standard output, and standard error revert to their previous values.

In the descriptions below, **filename** means either the name of a file or of an appropriate device (PRN, LPT1, LPT2, or LPT3 for printers; COM1 to COM4 for serial ports; CON for the keyboard and screen; CLIP: for the clipboard; NUL for the “null” device, etc.).

Here are the redirection symbols supported by 4DOS, 4NT, and Take Command:

To get input from a file or device instead of from the keyboard:

```
< filename
```

To redirect standard output to a file or device:

```
> filename
```

To redirect standard output and standard error to a file or device:

```
>& filename
```

To redirect standard error only to a file or device:

```
>&> filename
```

If you want to append output to the end of an existing file, rather than creating a new file, replace the first “>” in the last three commands above with “>>” (*i.e.*, use >>, >>&, and >>&>).

To use redirection, place the redirection symbol and filename at the end of the command, after the command name and any parameters. For example, to redirect the output of the DIR command to a file called *DIRLIST*, you could use a command line like this:

```
[c:\] dir /b *.dat > dirlist
```

You can use both input and output redirection for the same command, if both are appropriate. For example, this command sends input to SORT from the file *DIRLIST*, and sends output from SORT to the file *DIRLIST.SRT*:

```
[c:\] sort < dirlist > dirlist.srt
```

You can redirect text to or from the Windows clipboard by using the pseudo-device name **CLIP:** (the colon is required). The clipboard device is always available in 4NT and Take Command. It can only be used in 4DOS when you are running under Windows.

If you redirect the output of a single internal command like DIR, the redirection ends automatically when that command is done. If you start a batch file with redirection, all of the batch file's output is redirected, and redirection ends when the batch file is done. Similarly, if you use redirection at the end of a command group (see page 117), all of the output from the command group is redirected, and redirection ends when the command group is done.

### ❖ *Advanced Redirection Options*

When output is directed to a file with >, >&, or >&>, if the file already exists, it will be overwritten. You can protect existing files by using the SETDOS /N1 command (see page 476), the “Protect redirected output files” setting on the “Startup” tab of the configuration dialog, or the NoClobber directive in the .INI file (see page 223).

When output is appended to a file with >>, >>&, or >>&>, the file will be created if it doesn't already exist. However, if NoClobber is set as described

above, append redirection will not create a new file; instead, if the output file does not exist a “File not found” or similar error will be displayed.

You can temporarily override the current setting of NoClobber by using an exclamation mark [!] after the redirection symbol. For example, to redirect the output of DIR to the file *DIROUT*, and allow overwriting of any existing file despite the NoClobber setting:

```
[c:\] dir >! dirout
```

Redirection is fully nestable. For example, you can invoke a batch file and redirect all of its output to a file or device. Output redirection on a command within the batch file will take effect for that command only; when the command is completed, output will revert to the redirected output file or device in use for the batch file as a whole.

You can also use redirection to create a zero-byte file. To do so, enter **>filename** as a command, with no actual command before the > character.

**4DOS** For another method of changing the standard input and output devices see CTTY on page 289.

**4NT, TC** In addition to the redirection options above, 4NT and Take Command also support the *CMD.EXE* syntax:

```
n>file
```

and

```
n>&m
```

where [n] and [m] are digits between 0 and 9. You may not put any spaces between the n and the >, or between the >, &, and m in the second form. The digits represent file handles; 4NT and Take Command define “0” as standard input, “1” as standard output, and “2” as standard error. Handles 3 to 9 will probably not be useful unless you have an application which uses those handles for a specific, documented purpose, or you have opened a file with the %@FILEOPEN variable function (see page 176) and the file handle is between 3 and 9. The **n>file** syntax redirects output from handle n to a file. You can use this form to redirect two handles to different places. For example:

```
[c:\] dir > outfile 2> errfile
```

sends normal output to a file called *OUTFILE* and any error messages to a file called *ERRFILE*.

The **n>&m** syntax redirects handle **n** to the same location as the previously assigned handle **m**. For example, to send standard error to the same file as standard output, you could use this command:

```
[c:\] dir > outfile 2>&1
```

Notice that you can perform the same operations more easily by using standard 4NT or Take Command redirection features. The two examples above could be written as:

```
[c:\] dir > outfile >&> errfile
```

and

```
[c:\] dir >& outfile
```

## ***Piping***

You can also create a “pipe,” which sends the standard output of one command to the standard input of another command. To send the standard output of *command1* to the standard input of *command2*:

```
command1 | command2
```

To send the standard output and standard error of *command1* to the standard input of *command2*:

```
command1 |& command2
```

For example, to take the output of the SET command (which displays a list of your environment variables and their values) and pipe it to the SORT utility to generate a sorted list, you would use the command:

```
[c:\] set | sort
```

To do the same thing and then pipe the sorted list to the internal LIST command for full-screen viewing (see page 401):

```
[c:\] set | sort | list
```

- ❖ The TEE and Y commands (see pages 503 and 531) are “pipe fittings” which add more flexibility to pipes.
- ❖ Like redirection, pipes are fully nestable. For example, you can invoke a batch file and send all of its output to another command with a pipe. A pipe on a command within the batch file will take effect for that command only;

when the command is completed, output will revert to the pipe in use for the batch file as a whole. You may also have 2 or more pipes operating simultaneously if, for example, you have the pipes running in different windows or processes.

**4DOS** ❖ 4DOS creates one or two temporary files to hold the output of a pipe. The files are given unique names. By default, these files are stored in the root directory of the boot drive, but you can override this with either the **TEMP4DOS**, **TEMP**, or **TMP** environment variable (see page 152).

**4NT, TC** 4NT and Take Command implement pipes by starting a new process for the receiving program instead of using temporary files. The sending and receiving programs run simultaneously; the sending program writes to the pipe and the receiving program reads from the pipe. When the receiving program finishes reading and processing the piped data, it ends automatically.

In most cases, you will not notice any differences between these different types of pipes, except perhaps some additional disk activity under 4DOS. But you may not get the results you expect if you use a pipe command like:

```
[c:\] echo test | input %%var
```

In 4DOS, this pipe will create an environment variable called VAR and set its value as “test.” You will be able to see the new variable by typing SET at the prompt. However, in 4NT and Take Command, VAR will be set in the environment that belongs to the receiving program. But that environment will be discarded when the pipe has been emptied and the process ends. You will never see VAR in the environment even though the command processor and the operating system are both operating correctly.

The same cautions apply to the “pipe-fitting” commands, TEE and Y. When you use pipes with 4NT and Take Command, make sure you think about any possible consequences that can occur from using a separate process to run the receiving program.

### **Keystack**

The Keystack overcomes two weaknesses of input redirection: some programs do not use standard input, and read the keyboard through the operating system or BIOS. Also, input redirection doesn't end until the program or command terminates. You can't, for example, use redirection to send the opening commands to a program and then type the rest of the commands yourself. But Keystack lets you do exactly that.

The Keystack sends keystrokes to an application program. Once the Keystack is empty, the program will receive the rest of its input from the keyboard. The Keystack is useful when you want a program to take certain actions automatically when it starts. It is most often used in batch files and aliases.

The Keystack is invoked with the KEYSTACK command (see page 392). To place the letters, digits, and punctuation marks you would normally type for your program into the keystack, enclose them in double quotes:

```
[c:\] keystack "myfile"
```

Many other keys can be entered into the Keystack using their names. This example puts the **F1** key followed by the **Enter** key in the keystack:

```
[c:\] keystack F1 Enter
```

See page 34 for details on how key names are entered. See the KEYSTACK command on page 392 for information on using numeric key values along with or instead of key names, and other details about using the Keystack.

**4NT, TC** In 4NT or Take Command, you must activate the window for the program that will receive the characters before you place them into the Keystack. See KEYSTACK for additional details; see ACTIVATE on page 248 for information on activating a specific window.

## ***File Selection***

Most internal commands (like COPY, DIR, etc.) work on a file or a group of files. Besides typing the exact name of the file you want to work with, you can use several shorthand forms for naming or selecting files and the applications associated with them: **Extended Parent Directory Names; Wildcards; Date, Time, and Size Ranges; File Exclusion Ranges; Multiple Filenames; Include Lists; Executable Extensions; @File Lists; certain Command Switches; and Windows File Associations.** These features are explained in this section. At the end of this section we also explain methods for accessing files on remote systems, including the use of **Internet URLs** and connections to **FTP Servers**.

Most of these features apply to 4DOS, 4NT, and Take Command commands only, and can not be used to pass file names to external programs. For example, the file name `..\FILE.DAT` uses an extended parent directory name ("..."). It can be used in an internal command like COPY or MOVE. However, your editor probably was not designed to support this extension to traditional

directory names, and is likely to give an error message if you try to pass it such a name.

Remember throughout this section that, if you are using the traditional FAT file system, a filename is a **base name** of 1 to 8 characters, optionally followed by an **extension** which is a period [.] and 1 to 3 more characters. On drives which support long filenames, the names can contain up to 255 characters, including spaces. If a long filename includes spaces, you must place it in quotation marks on the command line. (See page 15 for additional details on filename formats and restrictions.)

### ***Extended Parent Directory Names***

4DOS, 4NT, and Take Command allow you to extend the traditional “..” syntax for naming the parent directory, by adding additional [.] characters. Each additional [.] represents an additional directory level above the current directory. For example, `.\FILE.DAT` refers to a file in the current directory, `..\FILE.DAT` refers to a file one level up (in the parent directory), and `...\FILE.DAT` refers to a file two levels up (in the parent of the parent directory). If you are in the `C:\DATA\FINANCE\JANUARY` directory and want to copy the file `LETTERS.DAT` from the directory `C:\DATA` to drive A, you could use this command:

```
[C:\DATA\FINANCE\JANUARY] copy ...\LETTERS.DAT A:
```

### ***Wildcards***

Wildcards let you specify a file or group of files by typing a partial filename. The appropriate directory is scanned to find all of the files that match the partial name you have specified.

Wildcards are usually used to specify which files **should** be processed by a command. If you need to specify which files **should not** be processed see File Exclusion Ranges on page 102 (for internal commands), or EXCEPT on page 337 (for external commands).

Most internal commands accept filenames with wildcards anywhere that a full filename can be used. There are two wildcard characters, the asterisk [\*] and the question mark [?], plus a special method of specifying a range of permissible characters.



An asterisk [\*] in a filename means “any zero or more characters in this position.” For example, this command will display a list of all files in the current directory:

```
[c:\] dir *
```

If you want to see all of the files with a *.TXT* extension, you could type this:

```
[c:\] dir *.txt
```

If you know that the file you are looking for has a base name that begins with *ST* and an extension that begins with *.D*, you can find it this way. Filenames such as *STATE.DAT*, *STEVEN.DOC*, and *ST.D* will all be displayed:

```
[c:\] dir st*.d*
```

You can also use the asterisk to match filenames with specific letters somewhere inside the name. The following example will display any file with a *.TXT* extension that has the letters *AM* together anywhere inside its base name. It will, for example, display *AMPLE.TXT*, *STAMP.TXT*, *CLAM.TXT*, and *AM.TXT*:

```
[c:\] dir *am*.txt
```

A question mark [?] matches any single filename character. You can put the question mark anywhere in a filename and use as many question marks as you need. The following example will display files with names like *LETTER.DOC* and *LATTER.DAT*, and *LITTER.DU*:

```
[c:\] dir l?tter.d??
```

The use of an asterisk wildcard before other characters, and of the character ranges discussed below, are enhancements to the standard wildcard syntax, and may not work properly with software other than 4DOS, 4NT, and Take Command.

### ❖ **Advanced Wildcards**

“Extra” question marks in your wildcard specification are ignored if the file name is shorter than the wildcard specification. For example, if you have files called *LETTER.DOC*, *LETTER1.DOC*, and *LETTERA.DOC*, this command will display all three names:

```
[c:\] dir letter?.doc
```

The file *LETTER.DOC* is included in the display because the “extra” question mark at the end of *LETTER?* is ignored when matching the shorter name *LETTER*.

In some cases, the question mark wildcard may be too general. You can also specify what characters you want to accept (or exclude) in a particular position in the filename by using square brackets. Inside the brackets, you can put the individual acceptable characters or ranges of characters. For example, if you wanted to match *LETTER0.DOC* through *LETTER9.DOC*, you could use this command:

```
[c:\] dir letter[0-9].doc
```

You could find all files that have a vowel as the second letter in their name this way. This example also demonstrates how to mix the wildcard characters:

```
[c:\] dir ?[aeiouy]*.*
```

You can exclude a group of characters or a range of characters by using an exclamation mark [!] as the first character inside the brackets. This example displays all filenames that are at least 2 characters long **except** those which have a vowel as the second letter in their names:

```
[c:\] dir ?[!aeiouy]*.*
```

The next example, which selects files such as *AIP*, *BIP*, and *TIP* but not *NIP*, demonstrates how you can use multiple ranges inside the brackets. It will display files whose names begin with **A, B, C, D, T, U, or V**:

```
[c:\] dir [a-dt-v]ip
```

You may use a question mark character inside the brackets, but its meaning is slightly different than a normal (unbracketed) question mark wildcard. A normal question mark wildcard matches any character, but will be ignored when matching a name shorter than the wildcard specification, as described above. A question mark inside brackets will match any character, but will **not** be discarded when matching shorter filenames. For example:

```
[c:\] dir letter[?].doc
```

will display *LETTER1.DOC* and *LETTERA.DOC*, but not *LETTER.DOC*.

A pair of brackets with no characters between them [], or an exclamation point and question mark together [!?], will match only if there is no character in that position. For example:

```
[c:\] dir letter[.doc
```

will not display *LETTER1.DOC* or *LETTERA.DOC*, but will display *LETTER.DOC*. This is most useful for commands like:

```
[c:\] dir /I"[]" *.btm
```

which will display a list of all *.BTM* files which **don't** have a description, because the empty brackets match only an empty description string (DIR /I selects files to display based on their descriptions).

You can repeat any of the wildcard characters in any combination you desire within a single file name. For example, the following command lists all files which have an **A**, **B**, or **C** as the third character, followed by zero or more additional characters, followed by a **D**, **E**, or **F**, followed optionally by some additional characters, and with an extension beginning with **P** or **Q**. You probably won't need to do anything this complex, but we've included it to show you the flexibility of extended wildcards:

```
[c:\] dir ??[abc]*[def]*.[pq]*
```

You can also use the square bracket wildcard syntax to work around a conflict between long filenames containing semicolons [;], and the use of a semicolon to indicate an include list (see page 104). For example, if you have a file named *C:\DATA\LETTER1;V2* and you enter this command:

```
[c:\] del \data\letter1;v2
```

you will not get the results you expect. Instead of deleting the named file, 4DOS, 4NT, or Take Command will attempt to delete *LETTER1* and then *V2*, because the semicolon indicates an include list (see page 104). However, if you use square brackets around the semicolon it will be interpreted as a filename character, and not as an include list separator. For example, this command would delete the file named above:

```
[c:\] del \data\letter1[;]v2
```

## ***Date, Time, and Size Ranges***

Most internal commands which accept wildcards also allow date, time, and size ranges to further define the files that you wish to work with. The command processor will examine each file's size and timestamp (a record of when the file was created, last modified, or last accessed) to determine if the file meets the range criteria that you have specified.

A range begins with the switch character (usually a slash [/]), followed by a left square bracket ([) and a character that specifies the range type: **s** for a size range, **d** for a date range, or **t** for a time range. The **s**, **d**, or **t** is followed by a start value, and an optional comma and end value. The range ends with a right square bracket (]). For example, to select files between 100 and 200 bytes long you could use the range **/[s100,200]**.

All ranges are inclusive. For example, a size range which selects files from 10,000 to 20,000 bytes long will match files that are exactly 10,000 bytes and 20,000 bytes long, as well as all sizes in between; a date range that selects files last modified between 10-27-02 and 10-30-02 will include files modified on each of those dates, and on the two days in between.

If you reverse range start and end values the command processor will recognize the reversal, and will use the second (lower) value as the start point of the range and the first (higher) value as its end point. For example, the range above for files between 100 and 200 bytes long could also be entered as **/[s200,100]**.

### Size Ranges

Size ranges simply select files whose size is between the limits given. For example, **/[s10000,20000]** selects files between 10,000 and 20,000 bytes long.

Either or both values in a size range can end with “k” to indicate thousands of bytes, “K” to indicate kilobytes (1,024 bytes), “m” to indicate millions of bytes, or “M” to indicate megabytes (1,048,576 bytes). For example, the range above could be rewritten as **/[s10k,20k]**.

The second argument of a size range is optional. If you use a single argument, like **/[s10k]**, you will select files of that size or larger. You can also precede the second argument with a plus sign [+]; when you do, it is added to the first value to determine the largest file size to include in the search. For example, **/[s10k,+1k]** select files from 10,000 through 11,000 bytes in size.

Some further examples of size ranges:

<u>Specification</u>	<u>Selects Files of Length</u>
<b>/[s0,0]</b>	zero (empty)
<b>/[s1M]</b>	1 megabyte or larger
<b>/[s10k,+200]</b>	between 10,000 and 10,200 bytes

## Date Ranges

Date ranges select files that were created or last modified at any time between the two dates. For example, `/[d12-1-02,12-5-02]` selects files that were last modified between December 1, 2002, and December 5, 2002.

You can use hyphens, slashes, or periods to separate the month, day, and year. The year can be entered as a 2-digit or 4-digit value. Two-digit years between 80 and 99 are interpreted as 1980 - 1999; values between 00 and 79 are interpreted as 2000 - 2079. For example, `/[d12-31-02,1-1-03]` is equivalent to `/[d12-31-2002,1-1-2003]`, and selects files modified on December 31, 2002 or January 1, 2003.

If either argument begins with a four digit year greater than 1900, it is assumed to be a date in the international format **yyyy-mm-dd**.

The time for the starting date defaults to 00:00:00 and the time for the ending date defaults to 23:59:59. You can alter these defaults, if you wish, by including a start and stop time inside the date range. The time is separated from the date with an at sign `@`. For example, the range `/[d7-1-02@8:00a,7-3-02@6:00p]` selects files that were modified at any time between 8:00 am on July 1, 2002 and 6:00 pm on July 3, 2002. If you prefer, you can specify the times in 24-hour format (e.g., `@18:00` for the end time in the previous example).

If you omit the second argument in a date range, the command processor substitutes the current date and time. For example, `/[d10-1-00]` selects files dated between October 1, 2000 and today.

You can use an offset value for either the beginning or ending date, or both. An offset begins with a plus sign `+` or a minus sign `-` followed by an integer. If you use an offset for the second value, it is calculated relative to the first. If you use an offset for the first (or only) value, the current date is used as the basis for calculation. For example:

<u>Specification</u>	<u>Selects Files</u>
<code>/[d1-27-2003,+3]</code>	modified between 1-27-2003 and 1-30-2003
<code>/[d1-27-2003,-3]</code>	modified between 1-24-2003 and 1-27-2003
<code>/[d-0]</code>	modified today (from today minus zero days, to today)
<code>/[d-1]</code>	modified yesterday or today (from today minus one day, to today)
<code>/[d-1,+0]</code>	modified yesterday (from today minus one day, to zero days after that)

As a shorthand way of specifying files modified today, you can also use **/[d]**; this has the same effect as the **/[d-0]** example shown above.

To select files last modified **n** days ago or earlier, use **/[d-n,1/1/80]**. For example, to get a directory of all files last modified 3 days or more before today (*i.e.*, those files **not** modified within the last 3 days), you could use this command:

```
[c:\] dir /[d-3,1/1/80]
```

This reversed date range (with the later date given first) will be handled correctly by 4DOS, 4NT, and Take Command. It takes advantage of the fact that an offset in the **start** date is relative to today, and that the base or “zero” point for PC file dates is January 1, 1980, or earlier.

You cannot use offsets in the time portion of a date range (the part after an @ sign), but you can combine a time with a date offset. For example, **/[d12-8-02@12:00,+2@12:00]** selects files that were last modified between noon on December 8 and noon on December 10, 2002. Similarly, **/[d-2@15:00,+1]** selects files last modified between 3:00 pm the day before yesterday and the end of the day one day after that, *i.e.*, yesterday. The second time defaults to the end of the day because no time is given.

Date ranges may not always work as you expect across a network, including on FTP servers, due to differences in time zone and file time storage method between the local and remote systems. Be sure to do some non-destructive testing before depending on date ranges to yield the results you want on a remote system.

See *Using Ranges* below for information on selecting files based on creation or access time, rather than modification time.

### ***Time Ranges***

A time range specifies a file modification time without reference to the date. For example, to select files modified between noon and 2:00 pm on any date, use **/[t12:00p,2:00p]**. The times in a time range can either be in 12-hour format, with a trailing “a” for AM or “p” for PM, or in 24-hour format.

If you omit the second argument in a time range, you will select files that were modified between the first time and the current time, on any date. You can also use offsets, beginning with a plus sign **[+]** or a minus sign **[-]** for either or both of the arguments in a time range. The offset values are interpreted as minutes. Some examples:

<u>Specification</u>	<u>Selects Files</u>
<code>/[t12:00p,+120]</code>	modified between noon and 2:00 PM on any date
<code>/[t-120,+120]</code>	modified between two hours ago and the current time on any date
<code>/[t0:00,11:59]</code>	modified in the morning on any date

Time ranges may not always work as you expect across a network, including on FTP servers, due to differences in time zone and file time storage method between the local and remote systems. Be sure to do some non-destructive testing before depending on time ranges to yield the results you want on a remote system.

See the next section for information on selecting files based on creation or access time, rather than modification time.

### ***Using Ranges***

If you combine two types of ranges, a file must satisfy both ranges to be included. For example, `/[d10-8-02,10-9-02] /s1024,2048]` means files last modified on October 8 or October 9, 2002, which are also between 1,024 and 2,048 bytes long.

When you use a date, time, size, or file exclusion range in a command, it should immediately follow the command name, so that any additional switches for the command are after any range(s) used. If the range is placed later in the command it may be ignored, or cause an error. Unlike some command switches which apply to only part of the command line, the range usually applies to all file names specified for the command.

For example, to get a directory of all the \*.C files dated October 1, 2002, you could use this command:

```
[c:\] dir /[d10-1-02,+0] *.c
```

To delete all of the 0-byte files on your hard disk, you could use this command:

```
[c:\] del /s0,0] *.* /s
```

And to copy all of the non-zero byte files in the current directory that you changed yesterday or today to your floppy disk, you can use this command:

```
[c:\] copy /[d-1] /s1] *.* a:
```

Date, time, and size ranges can be used with the ATTRIB, COPY, DEL, DESCRIBE, DIR, EXCEPT, FOR, LIST, MOVE, RD, REN, SELECT, and TYPE commands. They cannot be used with filename completion or in filename arguments for variable functions.

- ❖ It can be complex to type all of the elements of a range, especially when it involves multiple dates and times. In this case you may find it easier to use aliases for common operations. For example, if you often wish to select from .DAT files modified over the last three days and copy the selected files to the floppy disk, you might define an alias like this:

```
alias workback `select /[d-2] copy (*.dat) a:`
```

For more complex requirements, you may want to use internal variables (*e.g.* \_DATE or \_TIME, see page 153) and variable functions (*e.g.* @DATE, @TIME, @MAKEDATE, @MAKETIME, @FILEDATE, @FILETIME, or @EVAL, see page 165). These variables and functions allow you to perform arithmetic and date / time calculations.

The FAT file system maintains a single date and time for each file, reflecting the last time the file was written. This is the date and time used by 4DOS on a FAT drive with no LFN support (*e.g.* under MS-DOS 6.22, or PC DOS 7).

File systems which support long filenames maintain 3 sets of dates and times for each file: creation, last access, and last write. By default, date and time ranges work with the last write time stamp. You can use the “last access” (a) or “created” (c) time stamp in a date or time range with the syntax:

```
/[da...] or /[dc...] or /[ta...] or /[tc...]
```

For example, to select files that were last accessed yesterday or today:

```
/[da-1]
```

(NOTE: On VFAT and FAT32 drives, only the last access date is recorded by the operating system; the last access time is always returned as 00:00. However, on NTFS drives, last access information includes both date and time.)

### ***File Exclusion Ranges***

Most internal commands which accept wildcards also accept file exclusion ranges to further define the files that you wish to work with. The command processor examines each file name and excludes files that match the names you have specified in a file exclusion range.



A file exclusion range begins with a slash, followed by a left square bracket and an exclamation mark (“[!”) The range ends with a right square bracket (“]”).

Inside the brackets, you can list one or more filenames to be excluded from the command. The filenames can include wildcards and extended wildcards, but cannot include path names or drive letters.

The following example will display all files in the current directory except backup files (files with the extension *.BAK* or *.BK!*):

```
[c:\] dir /[!*.bak *.bk!] *.*
```

You can combine file exclusion ranges with date, time, and size ranges (see page 97). This example displays all files that are 10K bytes or larger in size and that were created in the last 7 days, except *.C* and *.H* files:

```
[c:\] dir /[s10k] /[d-7] /[!*.c *.h] *
```

File exclusion ranges will only work for internal commands. The EXCEPT command (page 337) can be used to exclude files for many external commands.

When you use a file exclusion range in a command, it should immediately follow the command name. See Using Ranges on page 101 for additional details.

## **Multiple Filenames**

Most file processing commands can work with multiple files at one time. To use multiple file names, you simply list the files one after another on the command line, separated by spaces. You can use wildcards in any or all of the filenames. For example, to copy all *.TXT* and *.DOC* files from the current directory to drive A, you could use this command:

```
[c:\] copy *.txt *.doc a:
```

If the files you want to work with are not in the default directory, you must include the full path with each filename:

```
[c:\] copy a:\details\file1.txt a:\details\file1.doc c:
```

Multiple filenames are handy when you want to work with a group of files which cannot be defined with a single filename and wildcards. They let you be very specific about which files you want to work with in a command.

- ! When you use multiple filenames with a command that expects both a source and a destination, like COPY or MOVE, be sure that you always include a specific destination on the command line. If you don't, the command will assume that the last filename is the destination and may overwrite important files.

Like extended wildcards and include lists (see below), multiple filenames work with internal commands but may not with external programs, unless those programs have been written to handle multiple file names on the command line.

If you have a list of files to process that's too long to put on the command line or too time-consuming to type, see DO on page xxx, FOR on page 347 or SELECT on page 460 for other ways of passing multiple file names to a command.

### ***Include Lists***

Any internal command that accepts multiple filenames will also accept one or more include lists. An include list is simply a group of filenames, with or without wildcards, separated by semicolons [;]. All files in the include list must be in the same directory. You may not add a space on either side of the semicolon.

If you used an include list instead of multiple file names for the previous examples, they would look like this:

```
[c:\] copy *.txt;*.doc a:
[c:\] copy a:\details\file1.txt;file1.doc c:
```

Include lists are similar to multiple filenames, but have three important differences. First, you don't have to repeat the path to your files if you use an include list, because all of the included files must be in the same directory. Second, if you use include lists, you aren't as likely to accidentally overwrite files if you forget a destination path for commands like COPY, because the last name in the list will be part of the include list, and won't be seen as the destination file name. (Include lists can only be used as the *source* parameter – the location files are coming from – for COPY and other similar commands. They cannot be used to specify a destination for files.)

Third, multiple filenames and include lists are processed differently by the DIR and SELECT commands. If you use multiple filenames, all of the files matching the first filename are processed, then all of the files matching the second name, and so on. When you use an include list, all files that match

any entry in the include list are processed together, and will appear together in the directory display or SELECT list. You can see this difference clearly if you experiment with both techniques and the DIR command. For example:

```
[c:\] dir *.txt *.doc
```

will list all the *.TXT* files with a directory header, the file list, and a summary of the total number of files and bytes used. Then it will do the same for the *.DOC* files. However:

```
[c:\] dir *.txt;*.doc
```

will display all the files in one list.

Like extended wildcards and multiple filenames (see above), the include list feature will work with internal commands, but not with external programs (unless they have been specifically programmed to support them). The maximum length of an include list is 260 characters in 4DOS and 2047 characters in 4NT and Take Command.

## ***LFN File Searches***

Under Windows, files on volumes which support long file names (including VFAT, FAT32, and NTFS volumes) can have both a long file name (LFN) and a short FAT-compatible file name. 4DOS (under Windows 95, 98, and ME), 4NT, and Take Command normally examine both forms of each file name when searching for files. They do so in order to remain compatible with the default command processors: *CMD.EXE* and *COMMAND.COM*.

The long filename is checked first, and if it does not match then the short name is checked. Matching files which have only a short filename will be found during the first search, because in that case Windows treats the short name as if it were a long name.

For example, suppose you have two files in a directory with these names:

Long Name	Short Name
Letter Home.DOC	LETTER~1.DOC
Letter02.DOC	LETTER02.DOC

A search for *LETTER??.DOC* will find both files. The second file (LETTER02.DOC) will be found during the search of long filenames. The first file ("Letter Home.DOC") will be found during the search of short filenames.

- ! Take extra care when you use wildcards to perform operations on LFN volumes because you may select more files than you intended. For example, Windows often creates short filenames that end “~1.”, “~2.”, etc. If you use a command like

```
del *1.*
```

you will delete all such files, including most files with long filenames, which is probably **not** the result you intended!

You can change this default behavior with the “SFN Search” checkbox on the “Startup” tab in the configuration dialog, or with the Win95SFNSearch / Win32SFNSearch directive in the *.INI* file. Set Win95SFNSearch (in 4DOS) or Win32SFNSearch (in 4NT and Take Command) to No to disable the secondary short filename search. This will prevent the potential problem described above, but will make the command processor’s behavior inconsistent with that of *COMMAND.COM* and *CMD.EXE*.

### Executable Extensions

Normally, when you type a filename (as opposed to an alias or internal command name) as the first word on the command line, the command processor looks for a file with that name to execute. The file’s extension may be *.EXE* or *.COM* to indicate that it contains a program, *.PIF* to indicate that it contains information on how to execute a program under Windows, or *.BTM*, *.BAT*, or *.CMD* to indicate a batch file.

The exact list of default extensions for executable files varies slightly depending on which operating system you use, because each has its own rules for batch file extensions (see page 19 for details).

You can add to the default list of extensions, and have the command processor take the action you want with files that are not executable programs or batch files. The action taken is always based on the file’s extension. For example, you could start your text editor whenever you type the name of a *.DOC* file, or start your database manager whenever you type the name of a *.DAT* file.

- 4NT, TC** Windows also includes the ability to associate file extensions with specific applications. See the next section for details on Windows file associations and their relationship to 4NT or Take Command executable extensions.

You can use environment variables to define the internal command, external program, batch file, or alias to run for each defined file extension. To create

an executable extension, use the SET command to create a new environment variable. An environment variable is recognized as an executable extension if its name begins with a period.

For example, if you want to run a word processor called *EDITOR* whenever you type the name of a file that has an extension of *.EDT*, you could use this command:

```
[c:\] set .edt=c:\edit\editor.exe
```

The syntax for creating an executable extension is:

```
set .ext[;.ext;...]=command [options]
```

where *.EXT* is the executable file extension; *command* is the name of the internal command, external program, alias, or batch file to run; and *[options]* are any command-line startup options you want to specify for the program, batch file, or alias. You can specify multiple extensions for a single command by separating them with semicolons.

If the *command* is a batch file or external program, the command processor will search the PATH for it if necessary. However, you can make sure that the correct program or batch file is used, and speed up the executable extension, by specifying the full name of the program to run, including drive, path, filename, and extension.

Once an executable extension is defined, any time you name a file with that extension the corresponding program, batch file, or alias is started, with the name of your file passed to it as a parameter.

The next example defines *WORDPAD.EXE* (a Windows editor) as the processor for *.TXT* files:

```
[c:\] set .txt="c:\program files\accessories\wordpad.exe"
```

Now, if you have a file called *HELLO.TXT* and enter the command

```
[c:\source] hello
```

the command processor will execute the command:

```
"c:\program files\accessories\wordpad.exe"  
c:\source\hello.txt
```

Notice that the full pathname of *HELLO.TXT* is automatically included. If you enter parameters on the command line, they are appended to the end of the command. For example, if you changed the above entry to:

```
[c:\source] hello -w
```

the command processor would execute the command:

```
"c:\program files\accessories\wordpad.exe"  
c:\source\hello.txt -w
```

In order for executable extensions to work, the command, program, batch file, or alias must be able to interpret the command line properly. For example, if a program you want to run doesn't accept a file name on its command line as shown in these examples, then executable extensions won't work with that program.

- ❖ Executable extensions may include wildcards, so you could, for example, run your text editor for any file with an extension beginning with **T** by defining an executable extension called *.T\**. Extended wildcards (e.g., **DO[CT]** for *.DOC* and *.DOT* files) may also be used.

The search for executable files starts in the current directory, then proceeds to each subdirectory specified by the **PATH** environment variable (if a **“.”** is used in the **PATH** the current directory is **not** searched first). Take Command also searches the **\WINDOWS** and **\WINDOWS\SYSTEM** directories after the current directory and before any directories listed in your search path. See the **PATH** command on page 429 for details on the way **\WINDOWS** and **\WINDOWS\SYSTEM** are searched, and for additional information on using a **“.”** in the **PATH**.

You may need to take this search order into account when using executable extensions. Using the *.C* example above, if you had a file named *FORMAT.C* in the current directory and entered the command **FORMAT A:**, your command would run the **WORDPAD** program specified by the executable extension, instead of finding the standard **FORMAT** command as you perhaps intended. You can get around this by remembering that the **FORMAT** command is in the file *FORMAT.COM* or *FORMAT.EXE*. If you entered the command **FORMAT.COM A:** (or **FORMAT.EXE A:**) then the *.C* executable extension would not match, and the search would continue until it found the *FORMAT* program.

To delete an executable extension, use the **UNSET** command (see page 522) to remove the corresponding variable.

## **@File Lists**

Certain file processing commands allow you to specify the files you want to process in a file list instead of on the command line. We call these "**@file lists**" because the "@" sign is used in the command, preceding the list filename.

An @file list is simply a standard ASCII file containing the names of the files to process, one per line. This allows you to create a list of files for processing using output from DIR /B, DIR /F, FFIND, a text editor, or any other method that produces a file in the proper format. Paths may be included in the file; see below for details.

@File lists are supported by the ATTRIB, COPY, DEL / ERASE, DESCRIBE, DO, EXCEPT, FOR, HEAD, LIST, MOVE, RD / RMDIR, REN / RENAME, TAIL, TOUCH, and TYPE commands.

To use an @file list, precede its name with an "@" sign in the command. For example, to copy all of the files listed in *MYLIST.TXT* to *D:\SAVE\*:

```
[c:\] copy @mylist.txt d:\save\
```

If you use a drive and/or path specification the "@" sign can appear before the path or before the file name. For example, these are equivalent:

```
[c:\] copy @e:\lists\mylist.txt d:\save\  
[c:\] copy e:\lists\@mylist.txt d:\save\
```

To use appropriately formatted data on the Windows clipboard as an @file list use **@CLIP:** as the file name, for example:

```
[c:\] copy @clip: d:\save\
```

## **@File Lists, Paths, and Subdirectories**

The entries in @file lists may contain no path, a relative path, or an absolute path, for example:

```
file1  
mydir\file1  
d:\data\mydir\file1
```

If a filename has no path the command processor will look for the file in the directory that is current when the operation takes place. Similarly, if a

relative path is used it will be interpreted as relative to the directory that is current when the operation takes place.

@file lists should **not** be used with the subdirectory switches in file processing commands (COPY /S, DEL /S, etc.). To process files listed in a single @file list across multiple subdirectories use FOR's ability to read the list and handle each file name individually, for example:

```
for %file in (@flist) copy /s %file d:\target\
```

### @File Lists and "@" Signs in File Names

Note that the "@" sign is a rarely used, but legal filename character in some environments. If a file whose name begins with @ exists and you attempt to use an @file list with the same name, the file whose name begins with @ will take precedence. For example, if C:\ contains both a file named @MYLIST.TXT and another named MYLIST.TXT, this command:

```
[c:\] copy @mylist.txt d:\save\
```

will copy the single file @MYLIST.TXT to D:\SAVE\, and will not process the list of files in MYLIST.TXT. To avoid this confusion, use a different name for one of the files.

### Command Switches for File Selection

The file processing commands (COPY, DEL, MOVE, REN, TYPE, etc.) support several standard switches for selecting files to process. Be sure to see the individual commands for details on which switches are supported for each command and how they work, and for additional switches specific to each command.

The common file selection switches include:

**/A:[[-+]r]hsad:** Select files based on their attributes, for example **/A:rh** selects files which have the read-only and hidden attributes set. See below for additional details.

**/I"text":** Select files based on their description. Wildcards are supported. For example, **/I"\*agua"** selects all files with the string "agua" somewhere in the file description. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all



filenames that do not have a description with **/T[""]**. See **DESCRIBE** (page 299) for details on file descriptions.

**/N**: Don't actually process any files. This allows you to test what the results of a command would be, without actually performing the operation.

**/P**: Prompt for confirmation of each file individually.

**/S**: Process files in the current directory and all of its subdirectories.

### **Attribute Switches**

Many 4DOS, 4NT, and Take Command commands have a **/A** or **/A:** switch which allows you to select files for the command to process based on their attributes. These switches all use the format **/A:[-][+]**RHSAD**** (the colon after **/A** is optional in **DIR**, **FFIND**, and **SELECT**, but is required in all other cases). The characters after the **/A:** specify which attributes to select, as follows:

<b>R</b>	Read-only	<b>A</b>	Archive
<b>H</b>	Hidden	<b>D</b>	Subdirectory
<b>S</b>	System		

**4NT, TC** 4NT and Take Command support several additional attributes for files on NTFS drives (Windows 2000 / XP):

<b>T</b>	Temporary file	<b>O</b>	Offline
<b>P</b>	Sparse file	<b>I</b>	Not content indexed
<b>E</b>	Encrypted	<b>J</b>	Junction (reparse point)
<b>C</b>	Compressed file		

If no attributes are listed at all (e.g., **/A:**), the command will process all files and (where applicable) all subdirectories, including hidden and system files and directories.

If attributes are combined, all the specified attributes must match for a file to be selected. For example, **/A:RHS** will select only those files with all three attributes set.

If you precede an attribute with a hyphen **[-]**, files without that attribute will be selected. For example, **/A:RH-S** selects files which have the read-only and hidden attributes set and which do not have the system attribute set.

If you precede an attribute with a plus **[+]**, files will be selected which have that attribute turned on or off. For example, **/A:+H+S** will select files with

the hidden or system attribute, or both, but will not select files which have neither attribute set. **/A:R+H+S** will select files which are read-only, and also have the hidden or system attribute, or both.

You can combine the plus sign, hyphen, and unmarked attributes to build a specification as complex as you need. For example, the attribute specification **/A: +R+H-SA** selects files which have the archive attribute set, which do not have the system attribute set, and which have the read-only and hidden attributes turned either on or off.

See File Attributes and Time Stamps on page 18 for more information on attributes.

### **4NT, TC Windows File Associations**

Windows includes the ability to associate file extensions with specific applications; this feature is sometimes called “file associations”. For example, within Windows a graphics program might be associated with files with a *.BMP* extension, while Notepad could be associated with files with a *.TXT* extension

When you attempt to start an application from the command line or a batch file, the command processor first searches for an external program file with a standard extension (*.COM*, *.EXE*, etc.). It then checks executable extensions. If all of these tests fail, the command processor looks in the registry to see if there is an association for it.

4NT and Take Command offer two commands which provide control over file associations. Both should be used with caution to avoid creating errors in the registry or damaging existing file types. The **ASSOC** command (see page 261) modifies or displays the associations between extensions and file types in the Windows registry. The **FTYPE** command (page 358) modifies or displays the command used to open a file of a specified type.

Executable extensions defined in 4NT or Take Command always take precedence over file associations defined in Windows. For example, if you associate the *.TXT* extension with your own editor using a 4NT or Take Command executable extension, and Windows has associated *.TXT* with Notepad, your setting will have priority, and the association with Notepad will be ignored when you invoke a *.TXT* file from within 4NT or Take Command.

### **4NT, TC Using Internet URLs**

If you type an Internet URL (Uniform Resource Locator) at the prompt, 4NT and Take Command will pass the URL to Windows. For example, for URLs beginning with an **http:** or **https:**, Windows will start your web browser, and request that the browser retrieve the page pointed to by the URL. (This feature will only work if Windows can find the proper association between the **http:** prefix and the browser software. While this association is standard for most browser installations, it may not be present on all systems.)

Due to the way Windows handles Internet URLs, you cannot wait for the browser software to finish when you enter an internet URL at the prompt; in this situation, 4NT always displays the next prompt immediately.

### **4NT, TC** *Using FTP Servers*

4NT and Take Command offer FTP support in internal commands such as COPY, DEL, DIR, FFIND, HEAD, LIST, MOVE, MD, RD, REN, SELECT, TAIL, and TYPE. The basic filename syntax for anonymous connections is:

```
"ftp://jpsoft.com/..."
```

(The double quotes around the directory/file name are required.)

You can also specify a username and password:

```
"ftp://username:password@ftp.domain.com/..."
```

If you enter \* for the password, 4NT and Take Command will prompt you (with character echoing disabled) for the password.

For example, to get a directory of the JP Software FTP site, you could use this command:

```
[c:\] dir "ftp://jpsoft.com/*"
```

If you have FTP permissions on server *ftp.domain.com* and a subdirectory of your root directory on the server called *mydir*, you can display the files with this command (enter this on one line):

```
[c:\] dir "ftp://username:password@ftp.domain.com/  
mydir/*"
```

You can also use the internal IFTP command to start an FTP session with a server and then use a simplified syntax to manipulate files on the server. See IFTP on page 382 for details.

**!** 4NT and Take Command use standard FTP commands to retrieve information about files and directories and manipulate those files and directories on FTP servers, and rely on the server's compliance with Internet FTP standards. If your server is not fully compliant, or does not operate in the manner that 4NT and Take Command expect, the results may not be what you intend. For example, FTP servers are supposed to be case-insensitive; if the server you are connecting to is not, you may have to use the proper case for file and directory names when you use FTP commands. We urge you to test each server you use with nondestructive commands like DIR before you try to copy or delete files, create or remove directories, etc.

Before you can use the built-in FTP support or the IFTP command, you must establish the necessary connection to the Internet. For example, if you use Windows Dial-Up Networking to connect to the Internet, you must start your dial up connection first.

### **4NT, TC *Using HTTP Servers***

4NT and Take Command offer HTTP support in the internal COPY and MOVE commands. The basic filename syntax is:

`"http://jpsoft.com/..."`

## ***Waiting for Applications to Finish***

**4NT, TC** (This section applies only to 4NT and Take Command. You cannot control how 4DOS waits for applications started from the prompt.)

When you start a Windows application from the prompt, the command processor does not normally wait for the application to finish before returning to the prompt. This allows you to continue your work at the prompt while the application is running. You can force 4NT and Take Command to wait for applications to finish before continuing by selecting the "Wait for External Apps" checkbox on the "Startup" tab of the configuration dialog, with the ExecWait directive in the .INI file (see page 221), or with the START command's /WAIT switch (START can also control many other aspects of how your applications are started).

Regardless of the ExecWait setting, 4NT and Take Command always wait for applications which are run from batch files before continuing with subsequent commands in the batch file. To start an application from a batch file and continue with the batch file immediately, without waiting for the application to finish, use the START command (without the /WAIT switch).

**TC** Take Command always waits for character-mode applications run in the Take Command console window, or run under Caveman. To start such a character-mode application and continue without waiting for the application to finish, use the START command (without the /WAIT switch) to start the application in its own window, or create a .PIF file for the application (the existence of a .PIF file forces Take Command to run the application in its own window).

Due to the way Windows handles URLs, you cannot wait for the browser software to finish when you enter an Internet URL at the prompt; in this situation, 4NT always displays the next prompt immediately.

## **Critical Errors**

DOS and Windows watch for physical errors during input and output operations. Physical errors are those due to hardware problems, such as trying to read a floppy disk while the drive door is open.

These errors are called **critical errors** because the operating system, command processor, or application program may not be able to proceed until the error is resolved.

**4DOS** When a critical error occurs, you will see a message asking you to choose one of four error handling options. The message comes from the command processor and will vary slightly depending upon your operating system and (under Windows) whether you are in full-screen or windowed mode. However, the options and their meanings are similar in all cases:

**Retry:** Retry the operation. Choose this option if you have corrected the problem.

**Ignore:** Ignore the error and continue. Use caution when choosing this option. Ignoring critical errors, especially on the hard disk, can cause additional errors or damage data on the disk.

**Fail:** Tell the operation that the operation failed. This option returns an error code to the command processor or to the application program that was running when the error occurred. 4DOS generally stops the current command when an operation fails. This option is not available for all errors; if you don't see it, use Abort instead. Under DOS, you can force a Fail response to all critical errors with the CritFail directive in *4DOS.INI* (see page 184).

**Abort:** Abort the program. Choose this option to stop the program that was running when the error occurred. Choosing Abort after an error will abort the command, but not the command processor itself.

**4NT, TC** Under 4NT and Take Command, Windows may handle critical errors internally, in which case the command processor will simply display a normal error message. If you must respond to the error condition, you will typically see a dialog asking you to choose one of three options:

**Abort or Cancel:** Tell the program that the operation failed. This option returns an error code to the command processor or to the application program that was running when the error occurred. The command processor generally stops the current command when an operation fails.

**Retry or Try Again:** Retry the operation. Choose this option if you have corrected the problem.

**Ignore or Continue:** Ignore the error and continue. This option can be dangerous; it tells the command processor (or the application that was running when the error occurred) that the operation succeeded when it did not!

### ❖ *Advanced Features*

The next three features are designed for advanced users. If you are a novice user, you might want to skim over this section and return to it as your computing skills and needs progress.

### *Conditional Commands*

When an internal command or external program finishes, it returns a result called the exit code. Conditional commands allow you to perform tasks based upon the previous command's exit code. Many programs return a 0 if they are successful and a non-zero value if they encounter an error.

If you separate two commands by **&&** (AND), the second command will be executed only if the first returns an exit code of 0. For example, the following command will only erase files if the BACKUP operation reports success:

```
[c:\] backup c:\ a: && del c:\*.bak;*.lst
```

If you separate two commands by **||** (OR), the second command will be executed only if the first returns a non-zero exit code. For example, if the

following BACKUP operation reports an error, then ECHO will display a message:

```
[c:\] backup c:\ a: || echo Error in the backup!
```

All internal commands return an exit code, but not all external programs do. Conditional commands will behave unpredictably if you use them with external programs which do not return an explicit exit code. To determine whether a particular external program returns a meaningful exit code use an **ECHO %?** command immediately after the program is finished. If the program's documentation does not discuss exit codes you may need to experiment with a variety of conditions to see how the exit code changes.

## **Command Grouping**

Command grouping allows you to group a set of commands together logically by enclosing them in parentheses. The parentheses are similar in function to the BEGIN and END block statements in some programming languages.

There are two primary uses for command grouping. One is to execute multiple commands in a place where normally only a single command is allowed. For example, suppose you wanted to execute two different REN commands in all subdirectories of your hard disk. You could do it like this:

```
[c:\] global ren *.wx1 *.wxo  
[c:\] global ren *.tx1 *.txo
```

But with command grouping you can do the same thing in one command:

```
[c:\] global (ren *.wx1 *.wxo & ren *.tx1 *.txo)
```

The two REN commands enclosed in the parentheses appear to GLOBAL as if they were a single command, so both commands are executed for every directory, but the directories are only scanned once, not twice. (To use a command like this under 4DOS, replace the ampersand [&] with a caret [^].)

This kind of command grouping is most useful with the DO, EXCEPT, FOR, GLOBAL, IF, and IFF commands. When you use this approach in a batch file you must either place all of the commands in the group on one line, or place the opening parenthesis at the end of a line and place the commands on subsequent lines. For example, the first two of these sequences will work properly, but the third will not:

```
for %f in (1 2 3) (echo hello %f & echo goodbye %f)
```

```
for %f in (1 2 3) (  
    echo hello %f  
    echo goodbye %f  
)  
  
for %f in (1 2 3) (echo hello %f  
echo goodbye %f)
```

The second common use of command grouping is to redirect input or output for several commands without repeatedly using the redirection symbols. For example, consider the following batch file fragment which places some header lines (including today's date) and directory displays in an output file using redirection. The first ECHO command creates the file using >, and the other commands append to the file using >>:

```
echo Data files %_date > filelist  
dir *.dat >> filelist  
echo. >> filelist  
echo Text files %_date >> filelist  
dir *.txt >> filelist
```

Using command grouping, these commands can be written much more simply. Enter this example on one line (use a caret [^] as the command separator if you try this under 4DOS):

```
(echo Data files %_date & dir *.dat & echo. & echo Text  
files %_date & dir *.txt) > filelist
```

The redirection, which appears outside the parentheses, applies to all the commands within the parentheses. Because the redirection is performed only once, the commands will run slightly faster than if each command was entered separately. The same approach can be used for input redirection and for piping (see page 88 for more details on redirection and piping).

You can also use command grouping in a batch file or at the prompt to split commands over several lines. This last example is like the redirection example above, but is entered at the prompt. Note the “More?” prompt after each incomplete line. None of the commands are executed until the command group is completed with the closing parenthesis. This example does **not** have to be entered on one line:

```
[c:\] (echo Data files %_date  
More? dir *.dat  
More? echo.  
More? echo Text files %_date  
More? dir *.txt) > filelist
```



[c:\]

A group of commands in parentheses is like a long command line. The total length of the group may not exceed 511 characters in 4DOS, or 4,095 characters in 4NT and Take Command, whether the commands are entered from the prompt, an alias, or a batch file. The limit **includes** the space required to expand aliases and environment variables used within the group. In addition, each line you type at the normal prompt or the **More?** prompt, and each individual command within the line, must meet the usual length limits: 511 characters in 4DOS, or 2,047 characters in 4NT and Take Command.

## Escape Character

4DOS, 4NT, and Take Command recognize a user-definable escape character. This character gives the following character a special meaning; it is **not** the same as the ASCII ESC that is often used in ANSI and printer control sequences.

The default 4DOS escape character is Ctrl-X (ASCII 24), which will be displayed here – and usually on your screen – as an up arrow [↑]. (The appearance of control characters depends on the font you use.) In 4NT and Take Command, the default escape character is a caret [^] (for compatibility with CMD.EXE).

If you don't like using the default escape character, you can pick another character using the SETDOS /E command (see page 474), the “Syntax” tab of the configuration dialog, or the EscapeChar directive in your .INI file (see page 221). If you plan to share aliases or batch files between 4DOS, 4NT, and Take Command, see page 195 for details about choosing compatible escape characters for two or more products.

Ten special characters are recognized when they are preceded by the escape character. The combination of the escape character and one of these characters is translated to a single character, as shown below. These are primarily useful for redirecting codes to the printer; ^e is also useful to generate ANSI “escape sequences” in your PROMPT, ECHO, or other output commands. The special characters which can follow the escape character are:

<b>b</b>	backspace	<b>n</b>	line feed
<b>c</b>	comma	<b>q</b>	double quote
<b>e</b>	ASCII ESC character (27)	<b>r</b>	carriage return
<b>f</b>	form feed	<b>s</b>	space
<b>k</b>	back quote	<b>t</b>	tab character

If you follow the escape character with any other character, the escape character is removed and the second character is copied directly to the command line. This allows you to suppress the normal meaning of special characters (such as ? \* / \ | " ` > < and &). For example, to display a message containing a < symbol, which normally indicates redirection:

```
[c:\] echo 2 is ^<4
```

To send a form feed followed by the sequence ESC Y to the printer, you can use this command:

```
[c:\] echos ^f^eY > prn
```

(In 4DOS, replace the caret [^] in these examples with a Ctrl-X [↑].)

The escape character has an additional use when it is the last character on any line of a *.BAT* or *.BTM* batch file. 4DOS, 4NT, and Take Command recognize this use of the escape character to signal line continuation: the command processor removes the escape character and appends the next line to the current line before executing it.

### Scrolling and History Keystrokes

In order to support the scrollbar buffer, some Take Command keystrokes are different from what you may be used to in 4DOS and 4NT. The differences are:

<u>Command Line:</u>	<u>4DOS and 4NT</u>	<u>Take Command</u>
Previous command	<b>Up</b> [↑]	<b>Ctrl-Up</b>
Next command	<b>Down</b> [↓]	<b>Ctrl-Down</b>
Open history window	<b>PgUp</b>	<b>Ctrl-PgUp</b>
Directory history	<b>Ctrl-PgUp</b>	<b>F6</b>
<u>Screen Scrollback:</u>		
Up one line	N/A	<b>Up</b> [↑]
Down one line	N/A	<b>Down</b> [↓]
Up one page	N/A	<b>PgUp</b>
Down one page	N/A	<b>PgDn</b>

If you prefer to reverse this arrangement and use the arrow and **PgUp** keys to access the command history without having to press **Ctrl** (as in 4DOS), see the *SwapScrollKeys .INI* file directive, or the Command Line 1 page of the

configuration dialogs. **SwapScrollKeys** switches the keystroke mapping so that the **↑**, **↓**, and **PgUp** keys manipulate the command history, and **Ctrl-↑**, **Ctrl-↓**, **Ctrl-PgUp**, and **Ctrl-PgDn** are used to control the scrollback buffer. (**SwapScrollKeys** does not affect the use of **F6** for the directory history).

You can also change the way any individual key operates with the corresponding key mapping directive in the *.INI* file. The directives associated with the history and scrolling keys are:

<b>NextHistory</b>	<b>ScrollUp</b>
<b>PrevHistory</b>	<b>ScrollDown</b>
<b>HistWinOpen</b>	<b>ScrollPgUp</b>
<b>DirWinOpen</b>	<b>ScrollPgDn</b>

## **Date Handling**

The date functions in 4DOS, 4NT, and Take Command are designed to handle events occurring during the life of the IBM PC-compatible computer system on which the software was installed, or a similar predecessor or successor system, and in addition within the limits set by the operating systems on which our products operate. These dates are typically associated with file creation, modification, or access, or with short-term processes related to operation of the computer system, such as the number of days since a backup or the time remaining until a program should be started.

Most date-related commands and functions can accept either a 2-digit or 4-digit year as input. 2-digit years between 80 and 99 are assumed to refer to the years 1980 - 1999; those between 00 and 79 are assumed to refer to 2000 - 2079. A 2-digit year number cannot be used to refer to a date outside this range; references to years beyond 2079 must be entered with 4 digits.

Year numbers in output are displayed as either 2-digit values (using the 1980 - 2079 range described above), or in some cases as 4-digit values.

The date functions in 4DOS, 4NT, and Take Command are not designed to handle dates outside the range stated above, and generally will not do so correctly. As a result, they cannot be used to process dates of events which may fall outside that range, such as historical events, life events (birth, death, etc.), and other dates which may be part of the data stored or processed on your system.

## **CHAPTER 5 / ALIASES AND BATCH FILES**

This chapter introduces two of the most powerful features of 4DOS, 4NT, and Take Command: aliases and batch files. It also discusses the environment (a list of information available to all programs), along with the command processor's internal variables and variable functions. The discussion of the environment, variables, and variable functions is included in this chapter because they are most often used in aliases and batch files.

### ***Aliases***

Much of the power of 4DOS, 4NT, and Take Command comes together in **aliases**, which give you the ability to create your own commands. An alias is a name that you select for a command or group of commands. Simple aliases substitute a new name for an existing command. More complex aliases can redefine the default settings of internal or external commands, operate as very fast in-memory batch files, and perform commands based on the results of other commands.

This section of the manual will show you some examples of the power of aliases. See the ALIAS command (page 250) for complete details about writing your own aliases. You can create aliases either from the command line, as described in this section, or in Take Command with the Aliases dialog which is available from the Utilities menu.

The simplest type of alias gives a new name to an existing command. For example, you could create a command called **R** (for Root directory) to switch to the root directory this way:

```
[c:\] alias r = cd \
```

After the alias has been defined this way, every time you type the command **R**, you will actually execute the command **CD \**.

Aliases can also create customized versions of commands. For example, the **DIR** command can sort a directory in various ways. You can create an alias called **DE** that means “sort the directory by filename extension, and pause after each page while displaying it” like this:

```
[c:\] alias de = dir /oe /p
```

Aliases can be used to execute sequences of commands as well. The following command creates an alias called MUSIC which saves the current drive and directory, changes to the SOUNDS directory on drive C, runs the program *E:\MUSIC\PLAYER.EXE*, and, when the program terminates, returns to the original drive and directory (enter this on one line):

```
[c:\] alias music = `pushd c:\sounds & e:\music\player.exe  
& popd`
```

This alias is enclosed in back-quotes because it contains multiple commands. You must use the back-quotes whenever an alias contains multiple commands, environment variables, parameters (see below), redirection, or piping. See the ALIAS command for full details. Also, please note that throughout this section we use the 4NT and Take Command command separator, an ampersand [&], to separate multiple commands. If you are using 4DOS, substitute a caret [^] for the ampersand in the examples.

When an alias contains multiple commands, the commands are executed one after the other. However, if any of the commands runs an external Windows application (such as the fictitious *PLAYER.EXE* shown above), you must be sure the alias will wait for the application to finish before continuing with the other commands. See **Waiting for Applications to Finish** on page 114 for additional details.

Aliases can be nested; that is, one alias can invoke another. For example, the alias above could also be written as:

```
[c:\] alias play = e:\music\player.exe  
[c:\] alias music = `pushd c:\sounds & play & popd`
```

If you enter MUSIC as a command, the command processor will execute the PUSH command, detect that the next command (PLAY) is another alias, and execute the program *E:\MUSIC\PLAYER.EXE*, and — when the program exits — return to the first alias, execute the POP command, and return to the prompt.

You can use aliases to change the default options for both internal commands and external commands. Suppose that you always want the DEL command to prompt before it erases a file:

```
[c:\] alias del = *del /p
```

An asterisk [\*] is used in front of the second “del” to show that it is the name of an internal command, not an alias. See page 252 for more information about this use of the asterisk.

You may have a program on your system that has the same name as an internal command. Normally, if you type the command name, you will start the internal command rather than the program you desire, unless you explicitly add the program's full path on the command line. For example, if you have a program named *DESCRIBE.COM* in the *C:\WUTIL* directory, you could run it with the command *C:\WUTIL\DESCRIBE.EXE*. However, if you simply type *DESCRIBE*, the internal *DESCRIBE* command will be invoked instead. Aliases give you two ways to get around this problem.

First, you could define an alias that runs the program in question, but with a different name:

```
[c:\] alias desc = c:\winutil\describe.exe
```

Another approach is to rename the internal command and use the original name for the external program. The following example renames the *DESCRIBE* command as *FILEDESC* and then uses a second alias to run *DESCRIBE.EXE* whenever you type *DESCRIBE*:

```
[c:\] alias filedesc = *describe  
[c:\] alias describe = c:\winutil\describe.exe
```

You can also assign an alias to a key, so that every time you press the key, the command will be invoked. You do so by naming the alias with an at sign [*@*] followed by a key name. After you enter this next example, you will see a 2-column directory with paging whenever you press **Shift-F5**, then **Enter**:

```
[c:\] alias @Shift-F5 = *dir /2/p
```

This alias will put the *DIR* command on the command line when you press **Shift-F5**, then wait for you to enter file names or additional switches. You must press **Enter** when you are ready to execute the command. To execute the command immediately, without displaying it on the command line or waiting for you to press **Enter**, use two at signs at the start of the alias name:

```
[c:\] alias @@Shift-F5 = *dir /2/p
```

The next example clears the window whenever you press **Ctrl-F2**:

```
[c:\] alias @@Ctrl-F2 = cls
```

Aliases have many other capabilities as well. This example creates a simple command-line calculator. Once you have entered the example, you can type *CALC 4\*19*, for example, and you will see the answer:

```
[c:\] alias calc = `echo The answer is:  %@eval[%$]`
```

Our last example in this section creates an alias called IN. It will temporarily change directories, run an internal or external command, and then return to the current directory when the command is finished:

```
[c:\] alias in = `pushd %1 & %2$ & popd`
```

Now if you type:

```
[c:\] in c:\sounds play furelise.wav
```

you will change to the *C:\SOUNDS* subdirectory, execute the command *PLAY FURELISE.WAV*, and then return to the current directory.

The above example uses two parameters: %1 means the first argument on the command line, and %2& means the second and all subsequent arguments. Parameters are explained in detail under the *ALIAS* command.

Your copy of 4DOS, 4NT, or Take Command includes a sample alias file called *ALIASES* which contains several useful aliases and demonstrates many alias techniques. Also, see the *ALIAS* and *UNALIAS* commands on pages 250 and 517 for more information and examples. See page 134 for tips about using aliases inside your batch files.

## **Batch Files**

A batch file is a file that contains a list of commands to execute. The command processor reads and interprets each line as if it had been typed at the keyboard. Like aliases, batch files are handy for automating computing tasks. Unlike aliases, batch files can be as long as you wish. Batch files take up separate disk space for each file, and can't usually execute quite as quickly as aliases, since they must be read from the disk.

### **.BAT, .CMD, and .BTM Files**

A batch file can run in two different modes. In the first, traditional mode, each line of the batch file is read and executed individually. In the second mode, the entire batch file is read into memory at once. The second mode can be several times faster, especially if most of the commands in the batch file are internal commands. However, only the first mode can be used for self-modifying batch files (which are rare) or for batch files which install memory-resident utilities under DOS.

The batch file's extension determines its mode. Files with a *.BAT* extension or a *.CMD* extension (in 4NT and Take Command) are run in the slower, traditional mode. Files with a *.BTM* extension are run in the faster, more efficient mode. You can change the execution mode inside a batch file with the *LOADBTM* command (see page 408).

4NT and Take Command can handle *.BTM* files of any length. Under 4DOS, *.BTM* files must be less than 64K (65536) bytes long.

### **4NT ! Using .BAT Files Under 4NT**

In most cases under 4NT your batch files will be stored as *.CMD* or *.BTM* files. However, you may also choose to use some *.BAT* files, especially if you are moving from DOS to Windows NT / 2000 / XP . If you do, you need to be aware of the way 4NT executes *.BAT* files, which is slightly different from the method used by *CMD.EXE*.

*CMD.EXE* passes all *.BAT* files to the Windows NT / 2000 /XP **DOS** command processor, typically *COMMAND.COM*, for execution (yes, there is a DOS command processor in Windows NT / 2000 / XP!). *COMMAND.COM* handles a few DOS-related commands, but passes most internal commands to a second copy of *CMD.EXE* so that they are executed in the Windows NT environment. This convoluted system allows you to load memory-resident DOS programs (TSRs), and run other programs which use them, all from the same *.BAT* file. However, it reduces performance for all *.BAT* files in order to support those rare files which load DOS TSRs under Windows NT.

4NT does not use this system; it executes *.BAT* files in the normal way, just like *.CMD* and *.BTM* files. This works better for most files, but may render DOS TSRs loaded from a *.BAT* file ineffective because other commands in the file are not executed in DOS-based environment.

In most cases this difference will not affect your *.BAT* files, because you will not be loading DOS TSRs in Windows NT / 2000 / XP. If you do need to load TSRs from *.BAT* files, we recommend that you obtain a copy of our DOS command processor, 4DOS, start it from your Windows NT / 2000 /XP desktop, and run the *.BAT* files from 4DOS (you could also use *CMD.EXE*, but of course the *.BAT* files then cannot use 4DOS or 4NT features). While we do not generally recommend using 4DOS under Windows NT / 2000 / XP , it can work well in this specific situation.



## **Echoing**

By default, each line in a batch file is displayed or “echoed” as it is executed. You can change this behavior, if you want, in several different ways:

Any batch file line that begins with an `[@]` symbol will not be displayed.

The display can be turned off and on within a batch file with the `ECHO OFF` and `ECHO ON` commands.

The default setting can be changed with the `SETDOS /V` command (see page 477), the “Default Batch Echo” checkbox on the “Startup” tab of the configuration dialog (see page 427), or the `BatchEcho` directive in the `.INI` file (see page 218).

For example, the following line turns off echoing inside a batch file. The `[@]` symbol keeps the batch file from displaying the `ECHO OFF` command:

```
@echo off
```

Your command processor also has a command line echo that is unrelated to the batch file echo setting. See the `ECHO` command on page 330 for details about both settings.

## **Batch File Parameters**

Like aliases and application programs, batch files can examine the command line that is used to invoke them. The command tail (everything on the command line after the batch file name) is separated into individual **parameters** (also called **arguments** or **batch variables**) by scanning for the spaces, tabs, and commas that separate them. A batch file can work with the individual parameters or with the command tail as a whole.

These parameters are numbered from `%1` to `%255` in 4DOS, or `%1` to `%511` in 4NT and Take Command.. `%1` refers to the first parameter on the command line, `%2` to the second, and so on. It is up to the batch file to determine the meaning of each parameter. You can use quotation marks to pass spaces, tabs, commas, and other special characters in a batch file parameter; see page 199 for details.

Parameters that are referred to in a batch file, but which are missing on the command line, appear as empty strings inside the batch file. For example, if you start a batch file and put two parameters on the command line, any

reference in the batch file to %3, or any higher-numbered parameter, will be interpreted as an empty string.

A batch file can also work with three special parameters: %0 contains the name of the batch file as it was entered on the command line, %# contains the number of command line arguments, and in 4NT and Take Command, %n\$ contains the complete command tail starting with argument number “n” (for example, %3\$ means the third parameter and all those after it). The default value of “n” is 1, so %\$ contains the entire command tail. The values of these special parameters will change if you use the SHIFT command (see page 481).

**4DOS** By default, 4DOS uses an ampersand [&] instead of a dollar sign [\$] to indicate the remainder of the command tail. For example, %& means all the parameters, and %2& means the second parameter and all those after it. If you want to share batch files or aliases between 4DOS and 4NT or Take Command, see page 195 for information on selecting compatible parameter characters for all products.

For example, if your batch file interprets the first argument as a subdirectory name then the following line would move to the specified directory:

```
cd %1
```

A friendlier batch file would check to make sure the directory exists and take some special action if it doesn't:

```
iff isdir %1 then
    cd %1
else
    echo Subdirectory %1 does not exist!
    quit
endiff
```

(see the IF and IFF commands on pages 372 and 380).

## ***Using Environment Variables***

Batch files can also use environment variables, internal variables, and variable functions. See pages 153 - 195 for a complete list of the internal variables and variable functions available. You can use these variables and functions to determine system status (*e.g.*, the type of CPU in the system), resource levels (*e.g.*, the amount of free disk space), file information (*e.g.*, the date and time a file was last modified), and other information (*e.g.*, the current date and time). You can also perform arithmetic operations

(including date and time arithmetic), manipulate strings and substrings, extract parts of a filename, and read and write files.

To create temporary variables for use inside a batch file, just use the **SET** command to store the information you want in an environment variable. Pick a variable name that isn't likely to be in use by some other program (for example, **PATH** would be a bad choice!), and use the **UNSET** command (page 522) to remove these variables from the environment at the end of your batch file. You can use **SETLOCAL** (page 480) and **ENDLOCAL** (page 333) to create a “local” environment so that the original environment will be restored when your batch file is finished.

Environment variables used in a batch file may contain either numbers or text. It is up to you to keep track of what's in each variable and use it appropriately; if you don't (for example, if you use **%@EVAL** to add a number to a text string), you'll get an error message.

## **Batch File Commands**

Some commands are particularly suited to batch file processing. Each command is explained in detail in the **Command Reference** section of this manual, beginning on page 241. Here is a list of some of the commands you might find most useful:

**4NT, TC**

**ACTIVATE** activates another window.

**BEEP** produces a sound of any pitch and duration through the computer's speaker.

**CALL** executes one batch file from within another.

**CANCEL** terminates all batch file processing.

**CLS** and **COLOR** set the display colors.

**DO** starts a loop. The loop can be based on a counter, or on a conditional test like those used in **IF** and **IFF**.

**DRAWBOX** draws a box on the screen.

**DRAWHLINE** and **DRAWVLINE** draw horizontal and vertical lines on the screen.

**ECHO** and **ECHOS** print text on the screen (the text can also be redirected to a file or device). **ECHOERR** and **ECHOSERR** print text to the standard error device.

**4NT, TC**

**EVENTLOG** writes a string to the Windows NT / 2000 / XP application event log.

**FOR** executes commands for each file that matches a set of wildcards, or each entry in a list.

**GOSUB** executes a subroutine inside a batch file. The **RETURN** command terminates the subroutine.

**GOTO** branches to a different location in the batch file.

**IF** and **IFF** execute commands based on a test of string or numeric values, program exit codes, or other conditions.

**INKEY** and **INPUT** collect keyboard input from the user and store it in environment variables.

**KEYSTACK** sends keystrokes to applications.

**LOADBTM** changes the batch file operating mode.

**4NT, TC**

**MSGBOX** displays a dialog box with standard buttons like Yes, No, OK, and Cancel, and returns the user's selection.

**ON** initializes error handling for Ctrl-C / Ctrl-Break, or for program and command errors.

**PAUSE** displays a message and waits for the user to press a key.

**4NT, TC**

**QUERYBOX** displays a dialog box for text input.

**QUIT** ends the current batch file and optionally returns an exit code.

**REM** places a remark in a batch file.

**SCREEN** positions the cursor on the screen and optionally prints a message at the new location.

**SCRPUT** displays a message in color.

**4NT, TC**

**SENDMAIL** sends an email message using SMTP.

**SETLOCAL** saves the current disk drive, default directory, environment, alias list, and special character settings.  
**ENDLOCAL** restores the settings that were saved.

**SHIFT** changes the numbering of the batch file parameters.

**4NT, TC**      **SNPP** send an message to an alphanumeric pager.

**4NT, TC**      **START** starts another program.

**SWITCH** selects a group of statements to execute based on the value of a variable.

**TC**      **TCTOOLBAR** changes the Take Command tool bar buttons.

**TEXT** displays a block of text. **ENDTEXT** ends the block.

**TIMER** starts or reads a stopwatch.

**TITLE** changes the window title.

**VSCRPUT** displays a vertical message in color.

These commands, along with the internal variables and variable functions, make the enhanced batch file language extremely powerful. Your copy of 4DOS, 4NT, or Take Command includes a sample batch file, *EXAMPLES.BTM*, that demonstrates some of the many things you can do with batch files.

## ***Interrupting a Batch File***

You can usually interrupt a batch file by pressing **Ctrl-C** or **Ctrl-Break**. Whether and when these keystrokes are recognized will depend on whether the command processor or an application program is running, how the application (if any) was written, whether **BREAK** is **ON** or **OFF** under DOS (see page 268), and whether the **ON BREAK** command is in use (see page 425).

If the command processor detects a **Ctrl-C** or **Ctrl-Break** (and **ON BREAK** is not in use), it will display a prompt, for example:

```
Cancel batch job C:\CHARGE.BTM ? (Y/N/A) :
```

Enter **N** to continue, **Y** to terminate the current batch file and continue with any batch file which called it, or **A** to end all batch file processing regardless

of the batch file nesting level. Answering **Y** is similar to the **QUIT** command (page 444); answering **A** is similar to the **CANCEL** command (page 270).

## ***Automatic Batch Files***

4DOS, 4NT, and Take Command support two “automatic” batch files, files that run without your intervention, as long as the command processor can find them.

Each time 4DOS, 4NT, or Take Command starts it looks for a file called *4START* (for 4DOS and 4NT), or *TCSTART* (for Take Command). *4START* and *TCSTART* are normally batch files (.BAT, .BTM, or .CMD), but can be any executable file. If the *4START* or *TCSTART* file is not in the same directory as your command processor, you should use the “Startup” tab of the configuration dialog, or the *4StartPath* / *TCStartPath* directive in your *.INI* file (see page 215) to specify its location. *4START* / *TCSTART* is optional, so the command processor will not display an error message if it cannot find the file.

*4START* / *TCSTART* is a convenient place to change the color or content of the prompt for each shell, LOG the start of a shell, or execute other special startup or configuration commands. It is also one way to set aliases and environment variables.

- ❖ With the exception of some initialization switches, the entire startup command line passed to the command processor is available to *4START* / *TCSTART* via batch file parameters (%1, %2, etc.). This can be useful if you want to see the command line passed to a secondary shell by an application. For example, to pause if any parameters are passed you could include this command in *4START* / *TCSTART* (enter this on one line):

```
if "%1" != "" pause Starting shell %_shell with parameters [%$]
```

**4DOS** Whenever it is started during the system boot process, 4DOS runs *AUTOEXEC.BAT* immediately after *4START*. On a DOS system, *AUTOEXEC.BAT* runs each time the computer boots up. (If *COMMAND.COM* cannot find *AUTOEXEC.BAT*, it asks you for the time and date. 4DOS skips that step and immediately displays a prompt.)

Normally, *AUTOEXEC.BAT* must be in the root directory of the boot drive. You can store it in a different location (and even give it a different name) by using the *4DOS.INI* directive *AutoExecPath* (see page 210). You can also

pass parameters to *AUTOEXEC.BAT* using the `AutoExecParms` directive in *4DOS.INI*.

4NT and Take Command do not execute *AUTOEXEC.BAT*.

Whenever 4DOS, 4NT, or Take Command end, they run a second file called *4EXIT* (for 4DOS and 4NT), or *TCEXIT* (for Take Command). *4EXIT* and *TCEXIT* are normally batch files (.BAT, .BTM, or .CMD). This file, if you use it, should be in the same directory as your *4START* / *TCSTART* file. Like *4START* or *TCSTART*, *4EXIT* and *TCEXIT* are optional. They are not necessary in most circumstances, but they are a convenient place to put commands to save information such as a history list before a shell ends, or LOG the end of the shell.

Under 4DOS, *4START* and *4EXIT* should not load any memory resident programs (TSRs). Otherwise, these files can include any commands that could be part of any batch file or any commands which you could type from the command line.

### ***Pipes, Transient Processes, and 4START / TCSTART***

When you set up the *4START* / *TCSTART* file, remember that it is executed **every** time the command processor starts, including when running a pipe in 4NT or Take Command (see page 92), or a transient copy of the command processor started with the `/C` startup option (see your *Introduction and Installation Guide* for details on `/C`).

For example, suppose you enter a command line like this, which uses a pipe:

```
[c:\data] myprog | sort > out.txt
```

Normally this command would create the output file *C:\DATA\OUT.TXT*. However, if you have a *4START* / *TCSTART* file which changes to a different directory, the output file will be written there — not in *C:\DATA*.

This is because both 4NT and Take Command start a second command processor to run the commands on the right hand side of the pipe, and that new copy runs *4START* before processing the commands from the pipe. If *4START* changes directories, the command from the pipe will be executed in the new directory. (This is not a problem in 4DOS because 4DOS does not start a second command processor to run a pipe.)

The same problem can occur if you use a transient shell started with `/C` to run an individual command, then exit — the shell will execute in the

directory set by 4START / TCSTART, not the directory in which it was originally started. For example, suppose you set up a desktop object with a command line like this, which starts a transient shell:

```
Command:      d:\4nt401\4nt.exe /c list myfile.txt
Working Directory:  c:\data
```

Normally this command would LIST the file *C:\DATA\MYFILE.TXT*. However, if 4START / TCSTART changes to a different directory, the command processor will look for *MYFILE.TXT* there — not in *C:\DATA*.

Similarly, any changes to environment variables or other settings in 4START / TCSTART will affect all copies of the command processor, including those used for pipes and transient shells.

You can work around these potential problems with the IF or IFF command and the internal variables `_PIPE` (page 162; only available in 4NT and Take Command) and `_TRANSIENT` (page 163). For example, to skip all 4START / TCSTART processing when running in a pipe or transient shell, you could use a command like this at the beginning of 4START / TCSTART:

```
if %_pipe != 0 .or. %_transient != 0 quit
```

We strongly recommend that you include a command like this at the beginning of 4START or TCSTART if you use a programmer's "MAKE" utility in your work. MAKE programs use transient shells to accomplish their work and can easily receive errors if the working directory or other parameters are changed each time a shell starts.

### ❖ **Detecting 4DOS, 4NT, or Take Command**

From a batch file, you can determine if 4DOS, 4NT, or Take Command is loaded by testing for the variable function `@EVAL`, with a test like this:

```
if "%@eval[2+2]" == "4" echo 4NT is loaded!
```

This test can never succeed in *COMMAND.COM* or *CMD.EXE*. Other variable functions could also be used for the same purpose.

### ❖ **Using Aliases in Batch Files**

One way to simplify batch file programming is to use aliases to hide unnecessary detail inside a batch file. For example, suppose you want a batch file to check for certain errors, and display a message and exit if one is



encountered. This example shows one way to do so (the command separator and parameter characters used here are those for 4NT and Take Command; change these characters appropriately if you are using 4DOS:

```
setlocal
unalias *
alias error `echo. & echo ERROR: %$ & goto dispmenu`
alias fatalerror `echo. & echo FATAL ERROR: %$ & quit`
alias in `pushd %1 & %2$ & popd`
if not exist setup.btm fatalerror Missing setup file!
call setup.btm
cls
:dispmenu
text
    1. Word Processing
    2. Spreadsheet
    3. Email
    4. Web
    5. Exit
endtext
echo.
inkey Enter your choice:  %%userchoice
switch %userchoice
case 1
    input Enter the file name:  %%fname
    if not exist d:\letters\%fname error No such file
    in d:\letters c:\wp\wp.exe %fname
case 2
    in d:\finance c:\quattro\q.exe
case 3
    c:\netscape\program\netscape.exe
case 4
    in e:\pmail c:\pegasus\winpm32.exe
case 5
    goto done
default
    error Invalid choice, try again
endswitch
goto dispmenu
:done
endlocal
```

The first alias, **ERROR**, simply displays an error message and jumps to the label **DISPMENU** to redisplay the menu. The “%\$” in the second **ECHO** command displays all the text passed to **ERROR** as the content of the message. The similar **FATALERROR** alias displays the message, then exits the batch file.

The last alias, *IN*, expects 2 or more command-line arguments. It uses the first as a new working directory and changes to that directory with a *PUSHD* command. The rest of the command line is interpreted as another command plus possible command line parameters, which the alias executes. This alias is used here to switch to a directory, run an application, and switch back. It could also be used from the command line.

The following 9 lines print a menu on the screen and then get a keystroke from the user and store the keystroke in an environment variable called *userchoice*. Then the *SWITCH* command is used to test the user's keystroke and to decide what action to take.

There's another side to aliases in batch files. If you're going to distribute your batch files to others, you need to remember that they may have aliases defined for the commands you're going to use. For example, if the user has aliased *CD* to *CDD* and you aren't expecting this, your file may not work as you intended. There are two ways to address this problem.

The simplest method is to use *SETLOCAL*, *ENDLOCAL*, and *UNALIAS* to clear out aliases before your batch file starts, and restore them at the end, as we did in the previous example. Remember that *SETLOCAL* and *ENDLOCAL* will save and restore not only the aliases but also the environment, the current drive and directory, and various special characters (see page 480 for details).

If this method isn't appropriate or necessary for the batch file you're working on, you can also use an asterisk [*\**] before the name of any command. The asterisk means the command that follows it should not be interpreted as an alias. For example the following command redirects a list of file names to the file *FILELIST*:

```
dir /b > filelist
```

However, if the user has redefined *DIR* with an alias this command may not do what you want. To get around this just use:

```
*dir /b > filelist
```

The same can be done for any command in your batch file. If you use the asterisk, it will disable alias processing, and the rest of the command will be processed normally as an internal command, external command, or batch file. Using an asterisk before a command will work whether or not there is actually an alias defined with the same name as the command. If there is no alias with that name, the asterisk will be ignored and the command will be processed as if the asterisk wasn't there.

## ❖ *Debugging Batch Files*

4DOS, 4NT, and Take Command include a built-in batch file debugger, invoked with the SETDOS /Y1 command (see page 478). The debugger allows you to “single-step” through a batch file line by line, with the file displayed in a popup window as it executes. You can execute or skip the current line, continue execution with the debugger turned off, view the fully-expanded version of the command line, or exit the batch file. The batch debugger can also pop up a separate window to view or edit current environment variables and aliases, and can pop up the LIST command to display the contents of any file.

To start the debugger, insert a SETDOS /Y1 command at the beginning of the portion of the batch file you want to debug, and a SETDOS /Y0 command at the end. You can also invoke SETDOS /Y1 from the prompt, but because the debugger is automatically turned off whenever the command processor returns to the prompt, you must enter the SETDOS command and the batch file name on the same line, for example:

```
[c:\] setdos /y1 & mybatch.btm
```

If you use the debugger regularly you may want to define a simple alias to invoke it, for example (in 4DOS, use %& rather than %\$):

```
[c:\] alias trace `setdos /y1 & %$`
```

This alias simply enables the debugger, then runs whatever command is passed to it. You can use the alias to debug a batch file with a command like this:

```
[c:\] trace mybatch.btm
```

You can also start or stop the debugger by pressing **Ctrl-F5** while a batch file is waiting for input, such as in an INPUT or INKEY statement. You must complete the input (press Enter during INPUT, or any other key during INKEY) before the debugger will start.

When the debugger is running you can control its behavior with keystrokes. Debugging continues after each keystroke unless otherwise noted:

<b>T</b> (race), <b>E</b> nter, or <b>F8</b>	Execute the current command. If it calls a subroutine with GOSUB, or another batch file with CALL, single-step into the called subroutine or batch file.
--	--

<b>S</b> (tep) or <b>F10</b>	Execute the current command, but execute any subroutine or CALLED batch file without single-stepping.
<b>J</b> (ump)	Skip the current command and proceed to the next command.
<b>X</b> (Expand)	Display the next command to be executed, after expansion of aliases and environment variables.
<b>L</b> (ist)	Prompt for a file name and then view the file with the LIST command.
<b>V</b> (ariables)	Open a popup window to display the current environment, in alphabetical order.
<b>A</b> (liases)	Open a popup window to display the current aliases, in alphabetical order.
<b>O</b> (ff) or <b>Esc</b>	Turn off the debugger and continue with the remainder of the batch file.
<b>Q</b> (uit)	Quit the debugger and the current batch file, without executing the remainder of the file.

The debugger highlights each line of the batch file as it is executed. It executes the commands on the line one at a time, so when a line contains more than one command, the highlight will not move as each command is executed. To see the individual commands, use the **X** key to expand each command before it is executed.

If you use a “prefix” command like **EXCEPT**, **FOR**, **GLOBAL**, or **SELECT**, the prefix command is considered one command, and each command it invokes is another. For example, this command line executes four commands — the **FOR** and three **ECHO** commands:

```
for %x in (a b c) do echo %x
```

You cannot use the batch debugger with **REXX** files (see page 145) or **EXTPROC** files (page 147). It can only be used with normal **4DOS**, **4NT**, and **Take Command** batch files.

The debugger gives you a detailed, step-by-step view of batch file execution, and will help solve particularly difficult batch file problems. However, in some cases you will find it easier to diagnose these problems with techniques that allow you to review what is happening at specific points in the batch file without stepping through each line individually.

There are several tricks you can use for this purpose. Probably the simplest is to turn ECHO on at the beginning of the file while you're testing it, or use SETDOS /V2 to force ECHO on even if an ECHO OFF command is used in the batch file. This will give you a picture of what is happening as the file is executed, without stopping at each line. It will make your output look messy of course, so just turn it off once things are working. You can also turn ECHO on at the beginning of a group of commands you want to "watch", and off at the end, just by adding ECHO commands at the appropriate spots in your file.

If an error occurs in a batch file, the error message will display the name of the file, the number of the line that contained the error, and the error itself. For example:

```
e:\test.bat [3] Invalid parameter "/d"
```

tells you that the file *E:\TEST.BAT* contains an error on line 3. The first line of the batch file is numbered 1.

Another trick, especially useful in a fast-moving batch file or one where the screen is cleared before you can read messages, is to insert PAUSE commands wherever you need them in order to be able to watch what's happening. You can also use an ON ERRORMSG command (see page 425) to pause if an error occurs, then continue with the rest of the file (the first command below), or to quit if an error occurs (the second command):

```
on errormsg pause
on errormsg quit
```

If you can't figure out how your aliases and variables are expanded, try turning LOG on at the start of the batch file. LOG keeps track of all commands after alias and variable expansion are completed, and gives you a record in a file that you can examine after the batch file is done. You must use a standard LOG command; LOG /H (the history log) does not work in batch files.

You may also want to consider using redirection to capture your batch file output. Simply type the batch file name followed by the redirection symbols, for example:

```
[c:\] mybatch >& testout
```

This records all batch file output, including error messages, in the file *TESTOUT*, so you can go back and examine it. If you have ECHO ON in the batch file you'll get the batch commands intermingled with the output, which

can provide a very useful trace of what's happening. Of course, output from full-screen commands and programs that don't write to the standard output devices can't be recorded, but you can still gain a lot of useful information if your batch file produces any output.

If you're using redirection to see the output, remember that any prompts for input will probably go to the output file and not to the screen. Therefore, you will need to know in advance the sequence of keystrokes required to get through the entire batch file, and enter them by hand or with KEYSTACK. Under 4NT and Take Command, you can also use the TEE command (see page 503) to both view the output while the batch file is running and save it in a file for later examination.

### ❖ **String Processing**

As you gain experience with batch files, you're likely to find that you need to manipulate text strings. You may need to prompt a user for a name or password, process a list of files, or find a name in a phone list. All of these are examples of string processing – the manipulation of readable text.

4DOS, 4NT, and Take Command include several features that make string processing easier. For example, you can use the INPUT, MSGBOX, and QUERYBOX commands for user input; the ECHO, SCREEN, SCRPUT, and VSCRPUT commands for output; and the FOR command or the @FILEREAD function to scan through the lines of a file. In addition, variable functions offer a wide range of string handling capabilities (see page 165 for full details, including a list of string-handling functions).

For example, suppose you need a batch file that will prompt a user for a name, break the name into a first name and a last name, and then run a hypothetical LOGIN program. LOGIN expects the syntax **/F:first /L:last** with both the first and last names in upper case and neither name longer than 8 characters. Here is one way to write such a batch file:

```
@echo off
setlocal
unalias *
input Enter your name (no initials):  %%name

set first=%@word[0,%%name]
set flen=%@len[%first]
set last=%@word[1,%%name]
set llen=%@len[%last]
```

```
iff %flen gt 8 .or. %llen gt 8 then
    echo First or last name too long
    quit
endiff

login /F:%@upper[%first] /L:%@upper[%last]
endlocal
```

The SETLOCAL command at the beginning of this batch file saves the environment and aliases. Then the UNALIAS \* command removes any existing aliases so they won't interfere with the behavior of the commands in the remainder of the batch file. The first block of lines ends with an INPUT command which asks the user to enter a name. The user's input is stored in the environment variable NAME.

The second block of lines extracts the user's first and last names from the NAME variable and calculates the length of each. It stores the first and last name, along with the length of each, in additional environment variables. Note that the @WORD function numbers the first word as 0, not as 1.

The IFF command in the third block of lines tests the length of both the first and last names. If either is longer than 8 characters, the batch file displays an error message and ends. (INPUT can limit the length of input text more simply with its /L switch. We used a slightly more cumbersome method above in order to demonstrate the use of string functions in batch files.)

Finally, in the last block, the batch file executes the LOGIN program with the appropriate parameters, then uses the ENDLOCAL command to restore the original environment and alias list. At the same time, ENDLOCAL discards the temporary variables that the batch file used (NAME, FIRST, FLEN, etc.).

When you're processing strings, you also need to avoid some common traps. The biggest one is handling special characters.

Suppose you have a batch file with these two commands, which simply accept a string and display it:

```
input Enter a string:  %%str
echo %str
```

Those lines look safe, but what happens if the user enters the string “some > none” (without the quotes). After the string is placed in the variable STR, the second line becomes

```
echo some > none
```

The “>” is a redirection symbol, so the line echoes the string “some” and redirects it to a file called NONE – probably not what you expected. You could try using quotation marks (see page 199) to avoid this kind of problem, but that won't quite work. If you use back-quotes (ECHO `%STR`), the command will echo the four-character string %STR. Environment variable names are not expanded (replaced by their contents, see page 197) when they are inside back-quotes.

If you use double quotes (ECHO "%STR"), the string entered by the user will be displayed properly, and so will the quotation marks. With double quotes, the output would look like this:

```
"some > none"
```

As you can imagine, this kind of problem becomes much more difficult if you try to process text from a file. Special characters in the text can cause all kinds of confusion in your batch files. Text containing back-quotes, double quotes, or redirection symbols can be virtually impossible to handle correctly.

One way to overcome these potential problems is to use the SETDOS /X command (see page 477) to temporarily disable redirection symbols and other special characters. The two-line batch file above would be a lot more likely to produce the expected results if it were rewritten this way:

```
setdos /x-15678
input Enter a string:  %%str
echo %str
setdos /x0
```

The first line turns off alias processing and disables several special symbols, including the command separator and all redirection symbols. Once the string has been processed, the last line re-enables the features that were turned off in the first line.

If you need advanced string processing capabilities beyond those provided by 4DOS, 4NT, and Take Command you may want to consider using the REXX language. Our products support external REXX programs for this purpose; see page 145 for additional details. You may also find other scripting or text processing languages such as AWK and PERL to be helpful for advanced string processing.



### ❖ *Line Continuation*

4DOS, 4NT, and Take Command will combine multiple lines in the batch file into a single line for processing when you include the escape character (see page 119) as the very last character of each line to be combined (except the last). For example, using the 4NT and Take Command escape character:

```
echo The quick brown fox jumped over the lazy^
sleeping^
dog. > alphabet
```

You cannot use this technique to extend a batch file line beyond the normal line length limit of 511 characters in 4DOS, or 2,047 characters in 4NT and Take Command.

### ❖ *Batch File Compression*

You can compress your batch files with a program called *BATCOMP.EXE*, distributed with 4DOS, and the equivalent *BATCOM32.EXE*, distributed with 4NT and Take Command (in the text below we refer to both as “BATCOMP”). This program condenses batch files by about a third and makes them unreadable with the LIST command and similar utilities. Compressed batch files run at approximately the same speed as regular *.BTM* files.

You may want to consider compressing batch files if you need to distribute them to others and keep your original code secret or prevent your users from altering them. The full syntax for the batch compression program is:

```
BATCOMP [/K /O /Q] input file [output file]
```

(under 4NT and Take Command, use *BATCOM32* instead of *BATCOMP*).

You must specify the full name of the input file, including its extension, on the *BATCOMP* command line. If you do not specify the output file, *BATCOMP* will use the same base name as the input file and add a *.BTM* extension. *BATCOMP* will also add a *.BTM* extension if you specify a base name for the output file without an extension. For example, to compress *MYBATCH.BAT* and save the result as *MYBATCH.BTM*, you can use any of these three commands from 4NT:

```
[c:\] batcom32 mybatch.bat
[c:\] batcom32 mybatch.bat mybatch
[c:\] batcom32 mybatch.bat mybatch.btm
```

If the output file (*MYBATCH.BTM* in the examples above) already exists, BATCOMP will prompt you before overwriting the file. You can disable the prompt by including */O* on the BATCOMP command line immediately before the input file name. Even if you use the */O* option, BATCOMP will not compress a file into itself.

The */K* option removes comments (blank lines and those starting with “*::*” or “*REM* ”) from the file prior to compression. Lines beginning “*REM >*” (a method used to create 0-byte files) will not be removed. Using */K* will make the compressed batch files smaller but will change line numbering, so that error messages which refer to batch file line numbers may be harder to decipher than they otherwise would be.

The */Q* option suppresses all output from BATCOMP, including error messages.

**4DOS** Under 4DOS, compressed *.BTMs* must be less than 64K bytes long. You can usually work around this limitation by breaking a very long batch file into two or more smaller files that *CALL* the other, and then compiling the shorter files separately.

**!** JP Software does not provide a decompression utility to uncompress batch files. If you use *BATCOMP.EXE* or *BATCOM32.EXE*, make sure that you also keep a copy of the original batch file for future inspection or modification.

You can adopt one of two strategies for keeping track of your original source files and compressed batch files. First, you may want to create the source files with a traditional *.BAT* or *.CMD* extension and reserve the *.BTM* extension for compressed batch files. The advantage of this approach is that you can modify and test the uncompressed versions at any time, although they will run in the slower, traditional mode unless they begin with a *LOADBTM* command (see page 408).

If you prefer, you can use a *.BTM* extension for both the source and compressed files. In this case you will have to use a different directory or a different base name for each file. For example, you might use *SOURCE\MYBATCH.BTM* for the source file and *COMP\MYBATCH.BTM* for the compressed version, or use *MYBATCHE.S.BTM* for the source file and *MYBATCH.BTM* for the compressed file (however, the latter approach may make it more difficult to keep track of the correspondence between the source file and the compressed file).

BATCOMP and BATCOM32 are character-mode applications designed to run in the environments where our command processors run. A batch file compressed with any copy of BATCOMP or BATCOM32 can be used with any current JP Software command processor (subject to 4DOS's 64K *.BTM* file size limit).

If you plan to distribute batch files to users of different platforms, be sure to read the compatibility discussion on page 195.

### ❖ **REXX Support**

REXX is a powerful file and text processing language developed by IBM, and available on many PC and other platforms. REXX is an ideal extension to the 4DOS, 4NT, and Take Command batch language, especially if you need advanced string processing capabilities.

The REXX language is not built into 4DOS, 4NT, or Take Command, and must be obtained separately. REXX support is built in to IBM PC DOS 7.0; you can also purchase add-on REXX software which runs under Windows, such as Enterprise Alternatives' Enterprise REXX, Quercus's Personal REXX (also available for DOS), IBM Object REXX, and Regina REXX (a freeware package).

**4DOS** Under 4DOS, REXX programs are stored either in *.BAT* or *.REX* files. *.REX* files are used for Quercus's Personal REXX; *.BAT* files are used for REXX programs under IBM PC\_DOS 7.0 and above, and can also be used with Personal REXX as described below.

To enable support for *.REX* files, you must define an executable extension (see page 106) that tells 4DOS to load REXX when you invoke a *.REX* file. For example:

```
set .rex=c:\prexx\rexx.exe
```

If you store REXX programs in *.BAT* files, the way you enable REXX support depends on whether you are running PC DOS 7 or above (which includes REXX), or another operating system such as MS-DOS, where native REXX support is not available and a third-party product must be used. The differences are:

- ✓ ✓ If you are using PC DOS 7 or above, 4DOS automatically checks each *.BAT* file to see if it contains a REXX program (see below). If a REXX program is found, 4DOS searches the PATH for *REXX.EXE*, the REXX interpreter included with the operating system. You can

use the Options 2 page of the OPTION dialogs or the REXXPath directive in *4DOS.INI* (see page 214) to specify the location of the REXX interpreter if *REXX.EXE* is not on your PATH, or if you wish to use a different REXX system whose interpreter is not named *REXX.EXE*.

- ✓ ✓If you are not using PC DOS 7 or above, 4DOS does not assume that it should check each *.BAT* file to see if it contains a REXX program. To enable this feature you **must** explicitly set the REXXPath directive in *4DOS.INI* to define the name and path for your REXX interpreter.

**4NT, TC** 4NT and Take Command support REXX programs stored in *.CMD*, *.REX*, or *.REXX* files. When the command processor loads, it asks Windows to locate the Enterprise REXX, Personal REXX, Object REXX, or Regina REXX libraries. If they are available, the command processor checks to see if you are running a REXX file, or if the first two characters on the first line of a *.CMD* file are *[/\*]*, the beginning of a REXX comment. If either of these tests succeeds, the command processor passes the file to the REXX interpreter for processing.

**TC** When working with a supported REXX interpreter (Enterprise REXX, Personal REXX, Object REXX, or Regina REXX), Take Command automatically handles all input and output for the REXX program, and all output appears in the Take Command window. If you need to run a REXX program inside your REXX processor's window, and not under Take Command, you should start the REXX processor's executable file explicitly, then load and run the REXX program from there.

All of the REXX processors described above also extend the interface between REXX and the command processor by allowing you to invoke 4DOS, 4NT, and Take Command commands from within a REXX program.

**4NT, TC** When you send a command from a REXX program back to the command processor to be executed (for example, if you execute a DIR command within a REXX script), the REXX software must use the correct "address" for the command processor. 4NT and Take Command both use the standard address **CMD**.

For details on communication between REXX and the command processor, or for more information on any aspect of REXX, see your REXX documentation.

## **4NT, TC❖ *EXTPROC* Support**

For compatibility with *CMD.EXE*, 4NT and Take Command offer an external processor (EXTPROC) option for batch files that lets you define an external program to process a particular *.CMD* file. To identify a *.CMD* file to be used with an external processor, place the string "EXTPROC" as the first word on the first line of the file, followed by the name of the external program that should be called. The command processor will start the program and pass it the name of the *.CMD* file and any command-line arguments that were entered.

For example, suppose *GETDATA.CMD* contains the following lines:

```
EXTPROC D:\DATAACQ\DATALOAD.EXE
OPEN PORT1
READ 4000
DISKWRITE D:\DATAACQ\PORT1\RAW
```

Then if you entered the command:

```
[d:\dataacq] getdata /p17
```

The command processor would read the *GETDATA.CMD* file, determine that it began with an EXTPROC command, read the name of the processor program, and then execute the command:

```
D:\DATAACQ\DATALOAD.EXE D:\DATAACQ\GETDATA.CMD /p17
```

The hypothetical *DATALOAD.EXE* program would then be responsible for reopening the *GETDATA.CMD* file, ignoring the EXTPROC line at the start, and interpreting the other instructions in the file. It would also have to respond appropriately to the command-line parameter entered (/p17).

Do not try to use 4NT or Take Command as the external processor named on the EXTPROC line in the *.CMD* file. They will interpret the EXTPROC line as a command to re-open themselves. The result will be an infinite loop that will continue until the computer runs out of resources and locks up.

## **TC ❖ *DDE* Support**

Take Command can communicate with other Windows applications by using Dynamic Data Exchange or DDE. To use Take Command as a "DDE client" and send a message from Take Command to another application, see the DDEEXEC command on page 291.

You can also use Take Command as a "DDE server" and send commands to it from another application. To do so, use an application or server name of "TCMD32", and a topic name of "Execute". The message can be any valid command that you could enter on the Command line: any alias, internal command, batch file, or external command. To send more than one line in a single DDE string, separate the lines with carriage return and line feed characters.

When using Take Command as a DDE Server you should start Take Command, execute the desired command, and exit Take Command, using the DDE commands of the "client" application. Take Command's DDE server feature is not intended to be used with a copy of Take Command running at the prompt; attempting to do so may result in unusual displays or display of the prompt in an incorrect location.

For example, if you use Microsoft Word for Windows you could use the following Visual Basic for Applications (VBA) fragment to start Take Command, then send it a DIR /W command, all from within Word:

```
Dim TCMDChannel
Shell ("d:\path\tcmd32.exe")
TCMDChannel = DDEInitiate("TCMD32", "Execute")
DDEExecute TCMDChannel, "dir /w"
DDETerminate TCMDChannel
```

You could use the same approach to execute a batch file or any other Take Command command from within Word, and similar approaches are available in other applications which offer DDE support. Consult your application manual for complete details.

In most cases, when you use Take Command as a DDE server you will want to redirect output from the commands you execute, because output on the Take Command screen is not likely to be useful to the client program which invokes a command.

## ***Using the Environment***

The **environment** is a collection of information about your computer that every program receives. Each entry in the environment consists of a variable name, followed by an equal sign and a string of text. You can automatically substitute the text for the variable name in any command. To create the substitution, include a percent sign [%] and a variable name on the command line or in an alias or batch file. For example, you can create a variable named BACKUP like this:

```
[c:\] set BACKUP=*.bak;*.bk!;*.bk
```

If you then type

```
[c:\] del %BACKUP
```

it is equivalent to the following command:

```
[c:\] del *.bak;*.bk!;*.bk
```

You can create, alter, view, and delete environment variables with the Environment dialog (available from the Utilities menu) as well as with the SET, ESET, and UNSET commands.

In 4DOS, the size of the environment can be specified on the Startup page of the OPTION dialogs, with the Environment and EnvFree directives in *4DOS.INI* (see page 211), or with the /E: startup switch (see your *Introduction and Installation Guide*). In 4NT and Take Command, the size of the environment is set automatically and expanded as necessary..

- ❖ Environment variable names may contain any alphabetic or numeric characters, the underscore character [], and the dollar sign [\$]. You can force acceptance of other characters by including the full variable name in square brackets, like this: %[AB##2]. You can also indirectly reference environment variables using square brackets. For example %[%var1] means “the contents of the variable whose name is stored in VAR1”. A variable referenced with this technique cannot contain more than 511 characters of information in 4DOS and 2047 characters in 4NT and Take Command.
- ❖ Environment variables may contain alias names. The command processor will substitute the variable value for the name, then check for any alias name which may have been included within the value. For example, the following commands would generate a 2-column directory of the .TXT files:

```
[c:\] alias d2 dir /2
[c:\] set cmd=d2
[c:\] %cmd *.txt
```

- ❖ The trailing percent sign that was traditionally required for environment variable names is not required in 4DOS, 4NT, or Take Command, which accept any character that cannot be part of a variable name (including a space) as the terminator. However, the trailing percent can be used to maintain compatibility.

The trailing percent sign **is** needed if you want to join two variable values. The following examples show the possible interactions between variables and

literal strings. First, create two environment variables called ONE and TWO this way:

```
[c:\] set ONE=abcd
[c:\] set TWO=efgh
```

Now the following combinations produce the output text shown:

%ONE%TWO	abcdTWO	( "%ONE%" + "TWO" )
%ONE%TWO%	abcdTWO	( "%ONE%" + "TWO%" )
%ONE%%TWO	abcdefgh	( "%ONE%" + "%TWO" )
%ONE%%TWO%	abcdefgh	( "%ONE%" + "%TWO%" )
%ONE%[TWO]	abcd[TWO]	( "%ONE%" + "[TWO]" )
%ONE%[TWO]%	abcd[TWO]	( "%ONE%" + "[TWO]%" )
%[ONE]%TWO	abcdefgh	( "%[ONE]" + "TWO" )
%[ONE]%TWO%	abcdefgh	( "%[ONE]" + "TWO%" )

- ❖ If you want to pass a percent sign to a command, or a string which includes a percent sign, you must use two percent signs in a row. Otherwise, the single percent sign will be seen as the beginning of a variable name and will not be passed on to the command. For example, to display the string “We're with you 100%” you would use the command:

```
echo We're with you 100%%
```

You can also use back-quotes around the text, rather than a double percent sign. See page 199 for details.

**4DOS** Each copy of the command processor maintains its own copy of the environment. The copy of the environment maintained by the primary shell is called the **master** environment. If 4DOS is running as both the primary shell and as a secondary shell, it will allow you to access the master environment from the secondary shell with the commands SET /M (page 468), UNSET /M (page 522), and ESET /M (page 334), and with the %@MASTER variable function (page 183). Master environment access is not applicable in 4NT or Take Command.

## ***Configuration Variables***

The following environment variables have special meanings in 4DOS, 4NT, and Take Command

**CDPATH** tells the command processor where to search for directories specified by the CD, CDD, and PUSHD commands and in automatic directory changes. (**\_CDPATH** can be used as an alternative to CDPATH



if you are using Microsoft Bookshelf, which uses a `CDPATH` variable for its own purposes.) `CDPATH` is composed of a list of directories, separated by semicolons [`;`]. See page 86 for more information about using `CDPATH`.

**CMDLINE** is the fully expanded text of the currently executing command line. `CMDLINE` is set just before invoking any *.PIF*, *.COM*, *.EXE*, *.BTM*, *.BAT*, or *.CMD* file. If a command line is prefaced with an “@” to prevent echoing (see page 63), it will not be put in `CMDLINE`, and any previous `CMDLINE` variable will be removed from the environment.

**COLORDIR** controls directory display colors used by `DIR` and `SELECT`. See page 310 for a complete description of the format of this variable.

**COMSPEC** contains the full path and name of the character-mode command processor.

**COPYCMD** is used by some versions of *COMMAND.COM* and *CMD.EXE* to hold default options for the `COPY` command. 4DOS, 4NT, and Take Command do not support this variable, but you can achieve the same effect with an alias. For example, if you want the `COPY` command to default to prompting you before overwriting an existing file, you could use this alias:

```
[c:\] alias copy = `*copy /r`
```

If you wish to use `COPYCMD` for compatibility with systems that do not use 4DOS, 4NT, or Take Command, you can define the alias this way:

```
[c:\] alias copy = `*copy %copycmd`
```

Then you could set `COPYCMD` to “/r” to achieve the same effect as the alias shown above.

**DIRCMD** is used by some versions of *COMMAND.COM* and *CMD.EXE* to hold default options for the `DIR` command. 4DOS, 4NT, and Take Command do not support this variable, but you can achieve the same effect with an alias. For example, if you want the `DIR` command to default to a 2-column display with a vertical sort and a pause at the end of each page, you could use this alias:

```
[c:\] alias dir = `*dir /2/p/v`
```

If you wish to use `DIRCMD` for compatibility with systems that do not use 4DOS, 4NT, or Take Command, you can define the alias this way:

```
[c:\] alias dir = `*dir %dircmd`
```

Then you could set DIRCMD to “/2/p/v” to achieve the same effect as the alias shown above.

**FILECOMPLETION** sets the files made available during filename completion for selected commands. See page 70 for a description of the format of this variable.

**PATH** is a list of directories that the command processor will search for executable files that aren't in the current directory. PATH may also be used by some application programs to find their own files. See page 19 and the PATH command on page 429 for a full description of this variable.

**PATHEXT** can be used to select the extensions to look for when searching the PATH for an executable file. It consists of a list of extensions, separated by semicolons. For example, to replicate the default extension list used by 4DOS, 4NT, and Take Command:

```
set pathext=.com;.exe;.btm;.bat
```

PATHEXT is ignored unless the PathExt setting is set to Yes in your *.INI* file (see page 224). Once PATHEXT is enabled the standard path search for *.COM*, *.EXE*, *.BTM*, and *.BAT*, files is replaced by a search for files with the extensions listed in PATHEXT, in the order listed there. Note that if you enable PATHEXT in Windows 200 / XP, the default value provided by Windows will exclude *.BTM* files from the executable file searches.

Enabling PATHEXT affects **only** the standard path search, it does not affect the subsequent searches for files with executable extensions. PATHEXT is supported for compatibility reasons but should not generally be used as a substitute for executable extensions, which are much more flexible. For more details on path searches, see the PATH command on page 429).

**PROMPT** defines the command-line prompt. It can be set or changed with the PROMPT command (see page 437).

- |             |  |
|-------------|--|
| <b>4DOS</b> | <b>TEMP</b> specifies the directory where 4DOS should store temporary pipe files if the TEMP4DOS variable doesn't exist. Some other programs also use TEMP to define where they should place their temporary files. Temporary pipe files are not used by 4NT and Take Command. |
| <b>4DOS</b> | <b>TEMP4DOS</b> specifies where 4DOS should store temporary pipe files. Temporary pipe files are not used in 4NT and Take Command.   |

**4NT, TC** 4NT and Take Command use the environment to keep track of the default directory on each drive or hard disk volume. DOS keeps track of the default directory for each drive letter internally; Windows does not. Like CMD.EXE, 4NT and Take Command overcome this by saving the default directory for each drive in the environment, using variable names that cannot be accessed by the user. Each variable begins with an equal sign followed by the drive letter and a colon (for example, =C:). You cannot view or change these variables with the SET command; they are only available for internal use by 4NT and Take Command.

### ❖ *Internal Variables*

**Internal variables** are special environment variables built into 4DOS, 4NT, and Take Command to provide information about your system. They are not actually stored in the environment, but can be used in commands, aliases, and batch files just like any other environment variable.

The values of these variables are stored internally in the command processor, and cannot be changed with the SET, UNSET, or ESET command. However, you can override any of these variables by defining a new variable with the same name.

These internal variables are often used in batch files and aliases to examine system resources and adjust to the current computer settings. You can examine the contents of any internal variable (except %= and %+) from the command line with a command like this:

```
[c:\] echo %variablename
```

The variables are listed below. The first list is by category, to assist you in locating the information you want quickly. The second list includes all the variables in alphabetical order, and defines the meaning of each one.

Some variables have constant values. These variables are included for compatibility, to make it easier to write batch files and aliases which work with all of our command processors.

On disk volumes which do not support long filenames, variables which return a path or file name will return their result in upper or lower case depending on the value of the SETDOS /U switch (see page 477) or the UpperCase directive in the .INI file (see page 228). On volumes which do support long filenames, these variables will return names as they are stored on the disk and no case shifting will be performed (see page 15 for more details).

Returned filenames are **not** quoted automatically; you must add quotes yourself if they are required for your use of the variable value.

Some variables return values based on information provided by your operating system. These variables will only return correct information if the operating system provides it. For example, `_BATTERY` will not return accurate results if your operating system and power management drivers do not provide correct information on battery status to the command processor.

A few examples of internal variable usage are located at the end of the lists, on page 164. For a more comprehensive set of examples see the online help, or the *EXAMPLES.BTM* file which came with your command processor.

### ***Internal Variable Categories***

Hardware status:

<code>_acstatus</code>	<code>_apmac</code>	<code>_apmbatt</code>	<code>_apmlife</code>
<code>_battery</code>	<code>_batterylife</code>	<code>_batterypercent</code>	<code>_cpu</code>
<code>_cpuusage</code>	<code>_kbhit</code>	<code>_monitor</code>	<code>_ndp</code>
<code>_video</code>			

Operating system and software status:

<code>_ansi</code>	<code>_boot</code>	<code>_codepage</code>	<code>_country</code>
<code>_dos</code>	<code>_dosver</code>	<code>_dpmi</code>	<code>_dv_host</code>
<code>_hwprofile</code>	<code>_ip</code>	<code>_mouse</code>	<code>_win</code>
<code>_windir</code>	<code>_winfgwindow</code>	<code>_winname</code>	<code>_winsysdir</code>
<code>_winticks</code>	<code>_wintitle</code>	<code>_winuser</code>	<code>_winver</code>

Command processor status:

<code>_4ver</code>	<code>_alias</code>	<code>_batch</code>	<code>_batchline</code>
<code>_batchname</code>	<code>_build</code>	<code>_cmdproc</code>	<code>_dname</code>
<code>_childpid</code>	<code>_detachpid</code>	<code>_echo</code>	<code>_env</code>
<code>_hlogfile</code>	<code>_kstack</code>	<code>_logfile</code>	<code>_pid</code>
<code>_pipe</code>	<code>_ppid</code>	<code>_shell</code>	<code>_startpid</code>
<code>_swapping</code>	<code>_transient</code>	<code>_wintitle</code>	

Screen, color, and cursor:

<code>_bg</code>	<code>_ci</code>	<code>_co</code>	<code>_column</code>
<code>_columns</code>	<code>_fg</code>	<code>_row</code>	<code>_rows</code>
<code>_selected</code>	<code>_xpixels</code>	<code>_ypixels</code>	

**Drives and directories:**

<code>_cwd</code>	<code>_cwds</code>	<code>_cwp</code>	<code>_cwps</code>
<code>_disk</code>	<code>_lastdisk</code>		

**Dates and times:**

<code>_date</code>	<code>_day</code>	<code>_dow</code>	<code>_dowf</code>
<code>_dowi</code>	<code>_doy</code>	<code>_hour</code>	<code>_idow</code>
<code>_idowf</code>	<code>_minute</code>	<code>_month</code>	<code>_second</code>
<code>_time</code>	<code>_year</code>		

**Error codes:**

<code>?</code>	<code>??</code>	<code>_?</code>	<code>errorlevel</code>
<code>_ftperror</code>	<code>_syserr</code>		

**Compatibility:**

<code>=</code>	<code>+</code>
----------------	----------------

*Internal Variable Details*

This alphabetical list includes all internal variables, and explains the value each variable returns. In the list below the possible values for most variables are shown in double quotes for ease of understanding. The actual values returned by the variables do not include the double quotes.

`?` contains the exit code of the last external command. Many programs return a “0” to indicate success and a non-zero value to signal an error. However, not all programs return an exit code. If no explicit exit code is returned, the value of `%?` is undefined.

**4DOS**    `??` returns a code which explains how the last program terminated:

- 0 – program terminated normally.
- 1 – program terminated by Ctrl-C or Ctrl-Break.
- 2 – program terminated due to a critical error.
- 3 – program terminated and stayed resident in memory (TSR).

`_?` contains the exit code of the last internal command. It is set to “0” if the command was successful, “1” if a usage error occurred, “2” if another command processor error or an operating system error occurred, or “3” if the command was interrupted by **Ctrl-C** or **Ctrl-Break**. You must use or save this value immediately, because it is set by every internal command.

**=** returns the current escape character. Use this variable, instead of the actual escape character, if you want your batch files and aliases to work regardless of how the escape character is defined. For example, if the escape character is a caret [^] (the default in 4NT and Take Command), both of the commands below will send a form feed to the printer. However, if the escape character has been changed, the first command will send the string “^f” to the printer, while the second command will continue to work as intended.

```
echos ^f > prn
echos %=f > prn
```

**+** returns the current command separator. Use this variable, instead of the actual command separator, if you want your batch files and aliases to work regardless of how the command separator is defined. %+ should always be surrounded by spaces. For example, if the command separator is an ampersand [&] (the default in 4NT and Take Command), both of the commands below will display “Hello” on one line and “world” on the next. However, if the command separator has been changed the first command will display “Hello & echo world”, while the second command will continue to work as intended.

```
echo Hello & echo world
echo Hello %+ echo world
```

**\_4VER** is the current 4DOS, 4NT, or Take Command version (for example, “7.50” or “5.00”). The current decimal character is used to separate the major and minor version numbers (see DecimalChar on page 220 for details). 4NT and Take Command also append a “U” for Unicode versions.

**4NT, TC** **\_ACSTATUS** returns the AC line status (0 = off-line, 1 = on-line, or “unknown”).

**4DOS** **\_ALIAS** contains the free space in the alias list, in bytes.

**\_ANSI** contains “1” if internal flags indicate that an ANSI compatible display driver is installed (4DOS) or ANSI support is enabled (4NT / Take Command); “0” if not.

**4DOS** In 4DOS, the internal flags which determine the value of **\_ANSI** depend on the SETDOS /A option (see page 473), and the ANSI directive in 4DOS.INI. If SETDOS /A is 0 or ANSI is set to Auto, 4DOS tests for the presence of an ANSI driver. In this case you may need to experiment to see if this variable works properly with your particular driver, because there is no standard and

100% reliable way to detect an ANSI driver. See page 28 for more information on ANSI drivers.

<u>SETDOS /A</u>	<u>ANSI Directive</u>	<u>ANSI Value</u>
0 (default)	Auto (default)	Result of test
1	Yes	1
2	No	0

**4NT, TC** In 4NT and Take Command, **\_ANSI** returns “1” if ANSI support is enabled, and “0” if it is not. To enable ANSI support use the Colors tab of the configuration dialog, the ANSI directive in the *4NT.INI* or *TCMD32.INI* file, or the SETDOS /A command. See the ANSI Codes Reference in the Reference section of the online help for information on the ANSI codes supported by 4NT and Take Command.

**4DOS** **\_APMAC** is the Advanced Power Management AC line status (“on-line”, “off-line”, or “unknown”). An empty string is returned if APM is not installed on your system.

**4DOS** **\_APMBATT** is the Advanced Power Management battery status (“high”, “low”, “critical”, “charging”, or “unknown”). An empty string is returned if APM is not installed.

**4DOS** **\_APMLIFE** is the Advanced Power Management remaining battery life (0 - 100 or “unknown”). An empty string is returned if APM is not installed.

**\_BATCH** is the current batch nesting level. It is “0” if no batch file is currently being processed.

**\_BATCHLINE** is the current line number in the current batch file. It is “-1” if no batch file is currently being processed.

**\_BATCHNAME** is the full path and file name of the current batch file. It is an empty string if no batch file is currently being processed.

**4NT, TC** **\_BATTERY** returns the battery charge status, which can be one or more of the values:

1	high
2	low
4	critical
8	charging
128	no battery
	unknown

**4NT, TC** **\_BATTERYLIFE** is the number of seconds of battery life remaining, or “unknown”.

**4NT, TC** **\_BATTERYPERCENT** is the percentage of battery charge remaining (0 – 100), or “unknown”.

**\_BG** is a string containing the first three letters of the current background screen output color (for example, “Bla”).

**\_BOOT** is the boot drive letter, without a colon.

**\_BUILD** is the internal 4DOS, 4NT, or Take Command build number.

**\_CI** is the insert mode cursor shape, as a percentage (see SETDOS /S on page 476 and CursorIns on page 219).

**4NT, TC** **\_CHILDPID** is the process ID of the most recent child process.

**\_CMDLINE** returns the current command line. This is most useful in key aliases. If you specify it on the command line, it will return the contents of the command line with the %\_cmdline name removed.

**\_CMDPROC** is the name of the current command processor (*i.e.*, “4DOS”, “4NT”, “TCMD32”).

**\_CO** is the overstrike mode cursor shape, as a percentage (see SETDOS /S on page 476 and CursorOver on page 220).

**\_CODEPAGE** is the current code page number.

**\_COLUMN** is the current cursor column (for example, “0” for the left side of the screen).

**\_COLUMNS** is the current number of virtual screen columns (for example, “80”). See page 39 for additional details on the virtual screen width.

**\_COUNTRY** is the current country code.

**\_CPU** is the CPU type:

86	8086 and 8088	386	i386
186	80186 and 80188	486	i486
200	NEC V20 and V30	586	Pentium
286	80286	686	Pentium Pro, II, III, or IV



Compatible AMD, Cyrix, or other processors will generally return the value corresponding to the Intel processor they most closely replicate. To determine the type, revision, stepping level, and other such details for advanced processors use the @WININFO variable function (see page 188).

**4NT, TC** **\_CPUUSAGE** is the current CPU usage, in percent (0 – 100).

**\_CWD** is the current working directory in the format *d:\pathname*.

**\_CWDS** has the same value as CWD, except it always ends the pathname with a backslash [\].

**\_CWP** is the current working directory in the format *\pathname*.

**\_CWPS** has the same value as CWP, except it always ends the pathname with a backslash [\].

**\_DATE** contains the current system date, in the format determined by your country settings.

**\_DAY** is the current day of the month (1 to 31).

**4NT, TC** **\_DETACHPID** is the process ID of the most recent DETACHed process.

**\_DISK** is the current disk drive, without a colon (for example, "C").

**\_DNAME** is the name of the file used to store file descriptions. It can be changed with the DescriptionName directive in the .INI file (see page 220), or with the SETDOS /D command (see page 472).

**\_DOS** is the operating system and command processor type. Each JP Software command processor returns a different value depending on the operating system, as follows:

	<b>4DOS</b>	<b>4NT</b>	<b>Take Command</b>
<b>DOS</b>	DOS		
<b>Windows 95</b>	DOS	WIN95C	WIN95
<b>Windows 98</b>	DOS	WIN98C	WIN98
<b>Windows ME</b>	DOS	WINMEC	WINME
<b>Windows NT</b>		NT	WIN32
<b>Windows 2000</b>		WIN2K	WIN32
<b>Windows XP</b>		WINXP	WIN32

This variable is useful if you have batch files running in more than one environment, and need to take different actions depending on the underlying operating environment or command processor. To differentiate between different versions of Windows within 4NT and Take Command, use the `_WIN` variable (see below); to differentiate between different command processors, use the `_CMDPROC` variable (above).

`_DOSVER` is the current operating system version. The current decimal character is used to separate the major and minor version numbers (see `DecimalChar` on page 220 for details).

`_DOW` is the first three characters of the current day of the week (“Mon”, “Tue”, “Wed”, etc.), in English. Also see `_IDOW`.

`_DOWF` is the full day of the week for the current date, in English. Also see `_IDOWF`.

`_DOWI` is the current day of the week as an integer (1 = Sunday, 2 = Monday, etc.).

`_DOY` is the day of the year (1 to 366).

**4DOS** `_DPMI` returns the DPMI version number, or “0” if DPMI isn’t present. (See the Glossary in the 4DOS online help for a short description of DPMI.)

**4DOS** `_DV` is “1” if DESQview is loaded or “0” otherwise.

`_ECHO` returns the current echo state (0=off, 1=on). There are two ECHO states, one for the command line and one for batch files.

**4DOS**    **\_ENV** is the free space in the environment, in bytes.

**4NT, TC**    **ERRORLEVEL** contains the exit code of the last external command. This variable is equivalent to the ? variable described on page 155. It is included only for compatibility with Windows NT / 2000 / XP *CMD.EXE*.

**\_FG** is a string containing the first three letters of the current foreground screen output color (for example, "Whi").

**4NT, TC**    **\_FTPERROR** is the error code of the last operating system error. The possible errors are:

101	You cannot change the remote host at this time
102	The remote host address is invalid
118	Firewall error
141	FTP protocol error
142	Communication error
143	Busy performing current action
144	Local file error
145	Can't open local file for reading
146	No remote file specified while uploading
147	Data interface error
301	Operation interrupted
302	Can't open local file
311	Accept failed for data connection
312	Asynchronous select failed for data connection
11001	Host not found
11002	Non-authoritative 'Host not found'
11003	Non-recoverable errors: FORMERR, REFUSED, NOTIMP
11104	Valid name, no data record (check DNS setup)

**\_HLOGFILE** returns the name of the current history log file (or an empty string if LOG /H is OFF).

**4NT, TC**    **\_HOST** is the hostname for the local computer.

**\_HOUR** is the current hour (0 - 23).

**4NT, TC**    **\_IDOW** is the 3-character abbreviation for the day of the week for the current date, in the current locale language.

**4NT, TC**    **\_IDOWF** is the full name for the day of the week for the current date, in the current locale language.

**4NT, TC**    **\_HWPROFILE** is the name of the current Windows hardware profile.

- 4NT, TC** **\_IP** is the IP address of the local computer. If there are more than one NIC in the computer, it returns a space-delimited list of all addresses.
- \_KBHIT** returns “1” if one or more keystrokes are waiting in the keyboard buffer, or “0” if the keyboard buffer is empty.
- 4DOS** **\_KSTACK** returns “1” if *KSTACK.COM* is loaded or “0” otherwise.
- \_LASTDISK** is the last valid drive letter, without a colon.
- \_LOGFILE** returns the name of the current log file (or an empty string if LOG is OFF). See LOG on page 410 for information on logging.
- \_MINUTE** is the current minute (0 - 59).
- 4DOS** **\_MONITOR** is the monitor type (“mono” or “color”).
- \_MONTH** is the current month of the year (1 to 12).
- \_MOUSE** is “1” if a mouse driver is loaded, and “0” otherwise.
- \_NDP** is the coprocessor type:
- |     |   |
|-----|---|
| 0   | no coprocessor is installed                   |
| 87  | 8087  |
| 287 | 80287   |
| 387 | 80387, 80486DX, 80487, or any type of Pentium |
- 4NT, TC** **\_PID** is the current process ID number.
- 4NT, TC** **\_PIPE** is “1” if the current process is running inside a pipe, and “0” otherwise.
- 4NT, TC** **\_PPID** is the process ID number of the parent process.
- \_ROW** is the current cursor row (for example, “0” for the top of the screen).
- \_ROWS** is the current number of screen rows (for example, “25”). See page 39 for additional details on screen size.
- \_SECOND** is the current second (0 - 59).
- TC** **\_SELECTED** returns the first line of text highlighted in the Take Command window. If no text has been highlighted, **SELECTED** returns an empty string.

**\_SHELL** is the current shell nesting level. The primary shell is level “0”, and each subsequent secondary shell increments the level by 1.

**4NT, TC** **\_STARTPID** is the process ID of the most recent START’ed process

**4DOS** **\_SWAPPING** returns the current swapping state. The return value is “OFF” if swapping has been disabled with the SWAPPING command, “EMS” if expanded memory is being used, “XMS” if extended memory is being used, or “Disk” if 4DOS is using disk swapping. The return value is “None” if swapping has been disabled with the *4DOS.INI* Swapping directive or if 4DOS failed to initiate memory or disk swapping during initialization (see your *Introduction and Installation Guide* for information about swapping).

**\_SYSERR** is the error code of the last operating system error. You will need a technical or programmer's manual to understand these error values.

**\_TIME** contains the current system time in the format hh:mm:ss. The separator character may vary depending upon your country information.

**\_TRANSIENT** is “1” if the current shell is transient (started with a /C, see your *Introduction and Installation Guide* for details), or “0” otherwise.

**4DOS** **\_VIDEO** is the video card type: “mono”, “cga”, “ega”, or “vga”.

**4DOS** **\_WIN** is the current Microsoft Windows mode. The possible values are:

- 0 Windows is not running
- 2 Windows 3.x in 386 enhanced mode
- 3 Windows 3.x in real or standard mode
- 40 Windows 95 / 98 / ME

**4NT, TC** **\_WINDIR** returns the pathname of the Windows directory.

**4NT, TC** **\_WINFGWINDOW** returns the title of the foreground window.

**4NT, TC** **\_WINNAME** returns the computer name of the current system.

**4NT, TC** **\_WINSYSDIR** returns the pathname of the Windows system directory.

**4NT, TC** **\_WINTICKS** returns the number of milliseconds since Windows was started.

**4NT, TC** **\_WINTITLE** returns the title of the current window.

**4NT, TC** **\_WINUSER** returns the name of the user currently logged on.

**4NT, TC** **\_WINVER** returns the current Windows version number. The current decimal character is used to separate the major and minor version numbers (see DecimalChar on page 220 for details).

**4NT, TC** **\_XPIXELS** returns the physical screen horizontal size in pixels.

**\_YEAR** is the current year (1980 to 2099).

**4NT, TC** **\_YPIXELS** returns the physical screen vertical size in pixels.

### ***Internal Variable Examples***

You can use these variables in a wide variety of ways depending on your needs. Here are just a few examples. For a more comprehensive set of examples see the online help, or the *EXAMPLES.BTM* file which came with your command processor.

Some of these examples rely on the IF command (page 372) or the IFF command (page 380) to test the value of a variable and perform different actions based on that value.

Store the current date and time in a file, then save the output of a DIR command in the same file:

```
echo Directory as of %_date %_time > dirsave
dir >> dirsave
```

Use the IFF command to check whether there are enough resources free before running an application:

```
iff %_GDIFREE lt 40 then
    echo Not enough GDI resources!
    quit
else
    d:\mydir\myapp
endiff
```

Call another batch file if today is Monday:

```
if "%_DOW" == "Mon" call c:\cleanup\weekly.bat
```

## ❖ *Variable Functions*

Variable functions are like internal variables, but they take one or more arguments (which can be environment variables or even other variable functions) and they return a value.

Like all environment variables, these variable functions must be preceded by a percent sign in normal use (%@EVAL, %@LEN, etc.). All variable functions must have square brackets enclosing their argument(s). The argument(s) to a variable function cannot exceed 511 characters in length in 4DOS, or 2047 characters in 4NT and Take Command, for all arguments taken as a group.

The variable functions are useful in aliases and batch files to check on available system resources, manipulate strings and numbers, and work with files and filenames.

A few examples of variable function usage are located at the end of the lists, on page 194. For a more comprehensive set of examples see the online help, or the *EXAMPLES.BTM* file which came with your command processor.

### *Notes on Specific Functions and Arguments*

Some variable functions, like @DISKFREE, are shown with “**b|k|m|g|t**” as one of their arguments. Those functions return a number of bytes, kilobytes, or megabytes based on the “**b|k|m|g|t**” argument:

- b** return the number of bytes
- K** return the number of kilobytes (bytes / 1,024)
- k** return the number of thousands of bytes (bytes / 1,000)
- M** return the number of megabytes (bytes / 1,048,576)
- m** return the number of millions of bytes (bytes / 1,000,000)
- g** return the number of billions of bytes (4NT and TC)
- G** return the number of gigabytes (4NT and TC)
- t** return the number of trillions of bytes (4NT and TC)
- T** return the number of terabytes (4NT and TC)

You can include commas (or the “thousands separator” character for your system) in the results from a “**b|k|m**” function by appending a “**c**” to the argument. For example, to add commas to a “**b**” or number of bytes result, enter “**bc**” as the argument. To set the thousands separator see the ThousandsChar directive on page 227.

Functions which accept a date as an argument use the date format and separators mandated by your country code (for example **dd.mm.yy** in

Germany, or **yy-mm-dd** in Japan). The year can be entered as a 4-digit or 2-digit value. Two-digit years between 80 and 99 are interpreted as 1980 - 1999; values between 00 and 79 are interpreted as 2000 - 2079. If a date begins with a four digit year greater than 1900, it is assumed to be in the international format **yyyy-mm-dd**.

Filenames passed as variable function arguments must be in quotes if they contain whitespace or special characters. Several functions also return filenames or parts of filenames. On LFN drives, the strings returned by these functions may contain white space or other special characters. To avoid problems which could be caused by these characters, quote the returned name before you pass it to other commands, for example (either of these methods would work):

```
set fname="%@findfirst[pro*.*]"
echo First PRO file contains:
type %fname
.....

set fname=%@findfirst[pro*.*]
echo First PRO file contains:
type "%fname"
.....
```

If you don't use the quotes in the SET or TYPE command in this example, TYPE will not interpret any white space or special characters in the name properly.

- ! In variable functions which take a drive letter as an argument, like @DISKFREE or @READY, the drive letter **must** be followed by a colon. The function will not work properly if you use the drive letter without the colon.
- ! The @FILEREAD, @FILEWRITE, @FILEWRITEB, @FILESEEK, @FILESEEKL, and @FILECLOSE functions allow you to access files based on their file handle. **These functions should only be used with file handles returned by @FILEOPEN**, unless otherwise noted under the individual functions. If you use them with any other file handle you may damage files or hang your system.

A number of functions accept a file attribute string to help determine which files to process (see page 18 for more information on file attributes). The string is typically shown as **-nrhsad** in the function parameter list. Wherever you see this argument, you can use any of the specified letters to refer to the desired file attributes, as follows:



<b>N</b>	Normal (no attributes set)	<b>S</b>	System
<b>R</b>	Read-only	<b>A</b>	Archive
<b>H</b>	Hidden	<b>D</b>	Directory

The rules for constructing the attribute string are the same as for attribute switches in commands (see page 111).

Many functions return values based on information provided by your operating system. Such functions will only return correct information if the operating system provides it. For example, @READY will not return accurate results if your operating system does not provide correct disk drive status information to the command processor.

The functions are listed below. The first list is by category, to assist you in locating the function you want. The second list includes all the functions in alphabetical order, and defines the arguments and return value for each one.

### ***Variable Function Categories***

#### **System status:**

@dosmem	@ems	@extended	@master
@readscr	@wininfo	@winmemory	@winmetrics
@winsystem	@xms		

#### **Drives and devices:**

@cdrom	@device	@diskfree	@disktotal
@diskused	@fstype	@label	@lpt
@ready	@remote	@removable	

#### **Files:**

@attrib	@crc32	@descript	@exetype
@fileage	@fileclose	@filedate	@fileopen
@fileread	@files	@fileseek	@fileseekl
@filesize	@filetime	@filewrite	@filewriteb
@findclose	@findfirst	@findnext	@line
@lines	@search	@truename	@unique
@verinfo	@wattrib		

#### **File names:**

@altname	@expand	@ext	@filename
@full	@lfn	@name	@path
@sfn			

**Strings and characters:**

@ascii	@caps	@char	@execstr
@field	@format	@index	@insert
@instr	@left	@len	@lower
@repeat	@replace	@right	@strip
@substr	@trim	@upper	@wild
@word	@words		

**Numbers and arithmetic:**

@abs	@ceiling	@comma	@convert
@dec	@decimal	@digits	@eval
@floor	@inc	@int	@max
@min	@numeric	@random	

**Dates and times:**

@day	@date	@dow	@dowf
@dowi	@doy	@idow	@idowf
@makeage	@makedate	@maketime	@month
@time	@year		

**Input Dialog Boxes**

@getdir	@getfile	@getfolder
---------	----------	------------

**Utility:**

@alias	@clip	@clipw	@errtext
@exec	@execstr	@function	@if
@iniread	@iniwrite	@option	@ping
@regcreate	@regquery	@regset	@regsetenv
@rexx	@select	@timer	@winclass
@winexename	@winstat		

***Variable Function Details***

**@ABS[n]:** Returns the absolute value of the number.

**@ALIAS[name]:** Returns the contents of the specified alias as a string, or a null string if the alias doesn't exist. When manipulating strings returned by @ALIAS you may need to disable certain special characters with SETDOS /X (see page 477). Otherwise, command separators, redirection characters, and other similar “punctuation” in the alias may be interpreted as part of the current command, rather than part of a simple text string.

**@ALTNAME[*filename*]**: Returns the alternate (short, “8.3” FAT-format) name for the specified file. If the *filename* is already in 8.3 format, returns the filename. If the file does not exist, returns an empty string. @ALTNAME will also return the shortened pathname if you provide a *path* in place of the *filename*.

**@ASCII[*c*]**: Returns the numeric value of the specified ASCII character as a string. For example %@ASCII[A] returns 65. If you enter more than one character, ASCII will return the numeric value for each character in a space delimited string. You can put an escape character [^] or [↑] before the actual character(s) to process. This allows quotes and other special characters as the argument (e.g., %@ASCII[↑]).

**@ATTRIB[*filename*[-nrhsad[,*p*]]]**: Returns a “1” if the specified file has the matching attribute(s); otherwise returns a “0”. See the note on page 166 for more information on the second argument. The “+” sign syntax used to select any of several attributes (e.g. +H+R for hidden or read-only files) will not work with @ATTRIB; use the optional third argument described below instead.

Without the optional *p* as a third argument, ATTRIB will only return a “1” if **all** of the attributes match. With the *p*, ATTRIB will return a “1” if **any** of the attributes match. For example, if *MYFILE.DAT* has R, H, and A attributes set:

%@attrib[myfile.dat,r]	returns 0 because there is not an exact match
%@attrib[myfile.dat,r,p]	returns 1 because there is a partial match.

If you do not specify any attributes, @ATTRIB will return the attributes of the specified file in the format **RHSAD** (or rather than a “0” or “1”. Attributes which are not set will be replaced with an underscore. For example, if *SECURE.DAT* has the read-only, hidden, and archive attributes set, %@ATTRIB[*SECURE.DAT*] would return “RH\_A\_” (without the quotes). If the file does not exist, @ATTRIB will return an empty string.

**@CAPS[“xxx”,text]**: Capitalizes the first letter of each word in the string. The first argument specifies the word separators you wish to use.

**@CDROM[*d*:]**: Returns “1” if the drive is a CD-ROM or “0” otherwise.

**4NT, TC @CEILING[*n*]**: Returns a value representing the smallest integer that is greater than or equal to the specified floating point number.

**@CHAR[*n*]**: Returns the character corresponding to an ASCII numeric value. If the argument is a set of numeric values, CHAR will return a string. For example **%@CHAR[65]** returns A; **@CHAR[65 66 67]** returns ABC.

**@CLIP[*n*]**: Returns line *n* from the clipboard. The first line is numbered 0. **\*\*\*EOC\*\*** is returned for all line numbers beyond the end of the clipboard. In 4DOS, this function will only work in Windows.

**@CLIPW[*text*]**: Writes the string to the clipboard. In 4DOS, this function will only work in Windows.

**@COMMA[*n*]**: Inserts commas, or the “thousands separator” character for your system, into a numeric string. To set the thousands separator see the ThousandsChar directive on page 227.

**@CONVERT[*input, output, value*]**: Converts a numeric string (*value*) from one number base (*input*) to another (*output*). Valid bases range from 2 to 36. In 4DOS, the *value* must be between 0 and  $2^{32}-1$  (2,147,483,647). In 4NT and Take Command, the *value* must be between 0 and  $2^{64}-1$ . No error is returned if *value* is outside that range. For example, to convert “1010101” from binary to decimal, use this command:

```
%@convert[2,10,1010101]
```

**@CRC32[*filename*]**: Returns a CRC32 value (compatible with that returned by PKZIP) for the specified *filename*, or -1 if the file doesn’t exist or can’t be opened.

**@DAY[*mm-dd-yy*]**: Returns the numeric day of the month for the specified date. See page 165 for information on acceptable date formats.

**@DATE[*mm-dd-yy*]**: Returns the number of days since January 1, 1980 for the specified date. See page 165 for information on acceptable date formats.

**@DEC[%*var*]**: Returns the same value as **@EVAL[%*var* - 1]**. That is, it retrieves and decrements the value of a variable. The variable itself is not changed; to do so, use a command like this:

```
set var=%@dec[%var]
```

**@DECIMAL[*n*]**: Returns the decimal portion of the number

**@DESCRIPT[*filename*]**: Returns the file description for the specified *filename* (see the DESCRIBE command on page 299).

**@DEVICE[*name*]**: Returns “1” if the specified *name* is a character device (such as a printer or serial port), or “0” if not.

**@DIGITS[*n*]**: Returns 1 if the string is digits only (including no decimal point, sign character, or thousands separator). Also see @NUMERIC.

**@DISKFREE[*d*;*b*|*k*|*m*]**: Returns the amount of free disk space on the specified drive. DOS networks with large server disk drives (over 2 GB) may report disk space values that are too small when @DISKFREE is used. This is because the network software does not report the proper values to 4DOS.

**@DISKTOTAL[*d*;*b*|*k*|*m*]**: Returns the total disk space on the specified drive. See the note about large DOS network drives under @DISKFREE above.

**@DISKUSED[*d*;*b*|*k*|*m*]**: Returns the amount of disk space in use by files and directories on the specified drive. See the note about large DOS network drives under @DISKFREE above.

**@DOSMEM[*b*|*k*|*m*]**: In 4DOS, returns the amount of free base memory. In 4NT and Take Command, returns the amount of free physical memory.

**@DOW[*mm-dd-yy*]**: Returns the first three characters of the day of the week for the specified date (“Mon”, “Tue”, “Wed”, etc.), in English. See page 165 for information on acceptable date formats. Also see @IDOW.

**@DOWF[*mm-dd-yy*]**: Returns the full day of the week for the specified date (“Monday”, “Tuesday”, etc.), in English. See page 165 for information on acceptable date formats. Also see @IDOWF.

**@DOWI[*mm-dd-yy*]**: Returns the day of the week for the specified date as an integer (1 = Sunday, 2 = Monday, etc.). See page 165 for information on acceptable date formats.

**@DOY[*mm-dd-yy*]**: Returns the day of year for the specified date (1-366). See page 165 for information on acceptable date formats.

**4DOS @EMS[*b*|*k*|*m*]**: Returns the amount of free EMS memory.

**@ERRTEXT[*n*]**: Returns the operating system error text for the specified code.

**@EVAL[*expression*]**: Evaluates an arithmetic *expression*. The *expression* can contain environment variables and other variable functions, and may use any of the operators listed below. @EVAL also supports parentheses,

commas, and decimals. Parentheses can be nested. @EVAL will strip leading and trailing zeros from the result. The valid operators are:

+/-	(with one value) negation or numeric sign ( <i>e.g.</i> -1, +3)
+	(with two values) addition
-	(with two values) subtraction
*	multiplication
/	division
\	integer division (returns the integer part of the quotient)
%%	modulo (returns the remainder when the first value is divided by the second)
**	exponentiation (in 4DOS, the exponent must be an integer)
AND	bitwise <b>and</b> (returns a 1 for each bit where the corresponding bits in both values are 1)
&	same as AND
OR	bitwise <b>or</b> (returns a 1 for each bit where the corresponding bit in either value is 1)
	same as OR
XOR	bitwise <b>exclusive or</b> (returns a 1 for each bit where the corresponding bit in one value is 1, and in the other value is 0)
^	same as XOR
<<	arithmetic shift left by the number of bits in the second value
>>	arithmetic shift right by the number of bits in the second value

**4NT, TC** 4NT and Take Command also support a number of functions:

log(x)	natural logarithm
log10(x)	log 10
exp(x)	exponential
sin(x)	sine
asin(x)	arcsine
sinh(x)	hyperbolic sine
cos(x)	cosine
acos(x)	arccosine
cosh(x)	hyperbolic cosine
tan(x)	tangent
atan(x)	arctangent
tanh(x)	hyperbolic tangent

When evaluating expressions, the order of precedence is: first functions, then exponentiation; then multiplication, division, and modulo; then addition and subtraction; then AND, OR, XOR, <<, and >>. For example,  $3 + 4 * 2$  will be interpreted as  $3 + 8$ , not as  $7 * 2$ . To change this order of evaluation, use parentheses to specify the order you want.

To ensure that your @EVAL expressions are interpreted correctly, spaces should be placed on both sides of each operator, for example:

```
%@eval[(20 %% 3) + 4]
```

```
%@eval[12 and 65]
```

You can enter hexadecimal numbers up to 8 (*4DOS*) or 16 (*4NT*, *TC*) digits long by preceding the number with **0x**. The result will always be returned in decimal; to convert it to hexadecimal use **@CONVERT**. For example:

```
[c:\] echo %@eval[0x10 + 0x10]  
32
```

```
[c:\] echo %@convert[10, 16, %@eval[0x10 + 0x10]]  
20
```

The maximum precision is 20 digits to the left of the decimal point and 10 digits to the right of the decimal point. You can alter the default precision to the right of the decimal point on the “Misc” tab of the configuration dialog, with the EvalMax and EvalMin *.INI* file directives (see page 221), or with the SETDOS /F command (see page 472). You can alter the decimal character from the “Syntax” tab of the configuration dialog, with the DecimalChar directive (see page 220), or with the SETDOS /G command.

You can alter the precision for a single evaluation with the construct **@EVAL[expression=x.y]**. The *x* value specifies the minimum decimal precision (the minimum number of decimal places displayed); the *y* value sets the maximum decimal precision. Use **=x,y** instead of **=x.y** if the comma is your decimal separator. You can specify either or both values. If *x* is greater than *y*, it is ignored; if only *x* is specified, *y* is set to the same value (e.g. **=2** is equivalent to **=2.2**). For example,

@eval[3 / 6=2.4]	returns 0.50
@eval[3 / 6=4.4]	returns 0.5000
@eval[3 / 7]	returns 0.4285714286
@eval[3 / 7=.4]	returns 0.4286
@eval[3 / 7=2.2]	returns 0.43
@eval[3 / 7=2]	also returns 0.43

Also see @DEC and @INC.

**@EXEC[[@]*command*]**: Execute the *command*. The *command* can be an alias, internal command, external command, *.BTM*, *.BAT*, or *.CMD* file.

@EXEC is primarily intended for running a program from within the PROMPT. It is a “back-door” entry into command processing and should be used with extreme caution. Incorrect or recursive use of @EXEC may or hang your system. By default, @EXEC returns the result code from the command; if you preface the command name with an '@' then @EXEC will return an empty string.

**@EXECSTR[*command*]**: Runs the specified *command* and returns the first line written to STDOUT by that *command*. Be sure to read the cautionary note under @EXEC above. @EXECSTR is useful for retrieving a result from an external utility — for example, if you have an external utility called *NETTIME.EXE* which retrieves the time of day from your network server and writes it to standard output, you could save it in an environment variable using a command like this:

```
[c:\] set server_time=%@execstr[d:\path\nettime.exe]
```

If the same utility returned a result properly formatted for the TIME command you could also use it to set the time on your system:

```
[c:\] time %@execstr[d:\path\nettime.exe]
```

**4NT, TC @EXETYPE[*filename*]**: Returns the application type:

0	Unknown	5	OS/2
1	DOS app	6	Win32 GUI
2	PIF file	7	Win32 console
3	Win16	8	Posix
4	Win 3.x VxD		

**@EXPAND[*filename*[-*nrhsad*]]**: Returns, on a single line, the names of all files and directories that match the *filename*, which may contain wildcards and include lists. Returns an empty string if no files match. If the file list is longer than the allowed command line length, it will be truncated without an error message. EXPAND will only insert a path in the returned filename if one was specified in the argument.

The second argument, if included, defines the attributes of the files that will be included in the search. See the note on page 166 for more information on this argument. If the attribute argument is not used, hidden files, system



files, and directories will be excluded from the returned list; all other files which match the **filename** will be included.

**@EXT[filename]**: Returns the extension from a **filename**, without a leading period. On volumes which support long file names, the extension can be up to 64 characters long. On traditional FAT drives it can be up to 3 characters long.

**4DOS @EXTENDED[b|k|m]**: Returns the amount of extended memory. Most memory managers convert extended memory to XMS and / or EMS memory, and make it unavailable as extended memory. Therefore, when a memory manager is loaded this function will usually return 0.

**@FIELD[["xxx"],n,string]**: Returns the *n*th field in the **string**. The first field is numbered 0. If *n* is negative, fields are returned from the end of the **string**. You can use the first argument, "**xxx**", to specify the separators that you wish to use. If you want to use a double quote as a separator, prefix it with an escape character (see page 119). If you don't specify a list of separators, @FIELD will consider only spaces, tabs, and commas as field separators. @FIELD works like @WORD, but it will not skip past multiple separator characters. (This allows you to read empty fields in a string.)

**@FILEAGE[filename]**: Returns the date and time of the file as a single numeric value, but can not be used for date and time calculations as it is not returned in identifiable units. The number can be used to compare the relative ages of two or more files. Also see @MAKEAGE.

**@FILECLOSE[n]**: Closes the file whose handle is *n*. You cannot close handles 0, 1 or 2. Returns "0" if the file closed OK or "-1" if an error occurred. Be sure to read the cautionary note about file functions on page 166.

**@FILEDATE[filename[,acw][,n]]**: Returns the date a file was last modified, in the default country format (mm-dd-yy for the US). The optional second argument selects which date field is returned for files on an LFN drive: *a* means the last access date, *c* means the creation date, and *w* means the last modification (write) date, which is the default. The optional third argument specifies the date format:

- 0 use system default
- 1 USA (mm/dd/yy)
- 2 European (dd/mm/yy)
- 3 Japan (yy/mm/dd)
- 4 International (yyyy-mm-dd)

**@FILENAME[*filename*]**: Returns the name and extension of a file, without a path.

**@FILEOPEN[*filename*, *read* | *write* | *append*, *b* | *t*]**: Opens the file in the specified mode and returns the file handle as an integer. The second parameter can be shorted to a single letter (“r”, “w”, or “a”). The optional third parameter controls whether the file is opened in binary or text mode. Text mode (the default) should be used to read text using @FILEREAD **without** a “length”, and to write text using FILEWRITE. Binary mode should be used to read binary data with @FILEREAD **with** a “length”, and to write binary data with @FILEWRITEB. To open a file for both reading and writing, open it in **append** mode, then use @FILESEEK to seek to the start of the file (or any other desired location) before performing additional operations. Returns “-1” if the file cannot be opened. Be sure to read the cautionary note about file functions on page 166.

**4NT, TC** @FILEOPEN can also open named pipes. The pipe name must begin with \\.\pipe\. @FILEOPEN first tries to open an existing pipe; if that fails, it tries to create a new pipe. Pipes are opened in blocking mode, duplex access, byte-read mode, and inheritable. For more information on named pipes see your Windows NT / 2000 / XP documentation.

**4NT, TC** @FILEOPEN can open file streams on NTFS drives under Windows 2000 and XP if the stream name is specified. See NTFS File Streams on page 18 for additional details on file streams.

**@FILEREAD[*n*, *length*]**: Reads data from the file whose handle is *n*. Returns “\*\*EOF\*\*” if you attempt to read past the end of the file. If *length* is not specified, @FILEREAD will read until the next CR or LF (end of line) character. If *length* is specified, @FILEREAD will read *length* bytes regardless of any end of line characters. Be sure to read the cautionary note about file functions on page 166.

**@FILES[*filename* [, *-nrhsad*]]**: Returns the number of files that match the *filename*, which may contain wildcards and include lists. Returns “0” if no files match. The *filename* must consist of a single file or directory name (with wildcards if desired); to check several directories or filenames use @FILES once for each , and add the results together with @EVAL. The second argument, if included, defines the attributes of the files that will be included in the search; see the note on page 166 for details.

**@FILESEEK[*n*, *offset*, *start*]**: Moves the file pointer *offset* bytes in the file whose handle is *n*. Returns the new position of the pointer, in bytes from the start of the file. Set *start* to 0 to seek relative to the beginning of the file, 1

to seek relative to the current file pointer, or 2 to seek relative to the end of the file. The **offset** value may be negative (seek backward), positive (seek forward), or zero (return current position, but do not change it). Be sure to read the cautionary note about file functions on page 166.

**@FILESEEKL[*n,line*]**: Moves the file pointer to the specified **line** in the file whose handle is **n**. The first line in the file is numbered 0. Returns the new position of the pointer, in bytes from the start of the file. Be sure to read the cautionary note about file functions on page 166. **@FILESEEKL** must read each line of the file up to the target line in order to position the pointer, and will therefore cause significant delays if used in a long loop or on a large file.

**@FILESIZE[*filename,b|k|m,a*]**: Returns the size of a file, or “-1” if the file does not exist. If the **filename** includes wildcards or an include list, returns the combined size of all matching files. The optional third argument **a** (allocated), if used, instructs **@FILESIZE** to return the amount of space allocated for the file(s) on the disk, rather than the amount of data in the file. Network drives and compressed drives may not always report allocated sizes accurately, depending on the way the network or disk compression software is implemented.

**@FILETIME[*filename,[acw],[s]*]**: Returns the time a file was last modified, in hh:mm format. The optional second argument selects which time field is returned for files on an LFN drive: **a** means the last access time, **c** means the creation time, and **w** means the last modification (write) time, which is the default. Times are normally returned with hours and minutes only; to retrieve seconds as well, add an **s** as the third argument. The last access time is always returned as 00:00 on LFN drives (see page 17 for additional details).

**@FILEWRITE[*n,text*]**: Writes a line to the file whose handle is **n**. Returns the number of bytes written, or “-1” if an error occurred. **n** must be a handle returned by **@FILEOPEN**; or 1 (for standard output) or 2 (for standard error). Be sure to read the cautionary note about file functions on page 166 for additional details.

**@FILEWRITEB[*n,length,string*]**: Writes the specified number of bytes from the **string** to the file whose handle is **n**. Returns the number of bytes written, or “-1” if an error occurred. Be sure to read the cautionary note about file functions on page 166.

**@FINDCLOSE[*filename*]**: Signals the end of a **@FINDFIRST** / **@FINDNEXT** sequence. You must use this function to release the directory search handle used for **@FINDFIRST** / **@FINDNEXT**.

**@FINDFIRST[*filename* [-*nrhsad*]]:** Returns the name of the first file that matches the *filename*, which may contain wildcards and include lists. The second argument, if included, defines the attributes of the files that will be included in the search. Returns an empty string if no files match. Be sure to read the notes on page 166 about quoting returned long filenames, and the note on page 166 for more information on the attribute argument.

After the last @FINDFIRST or @FINDNEXT, you must use @FINDCLOSE to avoid running out of directory search handles.

**@FINDNEXT[[*filename* [-*nrhsad*]]:** Returns the name of the next file that matches the filename passed to @FINDFIRST. @FINDNEXT should only be used after a successful call to @FINDFIRST. The first argument is included for compatibility with previous versions, but is ignored; it can be omitted if the second argument is not used (e.g. %@FINDNEXT[]). The second argument, if included, defines the attributes of the files that will be included in the search (see the note on page 166 for details). Returns an empty string when no more files match.

**4NT, TC @FLOOR[*n*]:** Returns a value representing the largest integer that is less than or equal to the specified floating point number.

After the last @FINDFIRST or @FINDNEXT, you must use @FINDCLOSE to avoid running out of directory search handles. Be sure to read the note on page 166 about quoting returned long filenames.

**@FORMAT[[-][[0]*x*][.*y*],*string*]:** Reformats a *string*, truncating it or padding it with spaces as necessary. If you use the minus [-], the *string* is left-justified; otherwise, it is right-justified. The *x* value is the minimum number of characters in the result. The *y* value is the maximum number of characters in the result. You can combine the options as necessary. For example:

```
"%@format[12,JPSsoftware]"    returns "  JPSsoftware"  
"%@format[.3,JPSsoftware]"    returns "JPS"
```

@FORMAT will add leading or trailing spaces if necessary to pad the result to the minimum width specified by *x*. If a leading zero is used before *x*, the padding will be with 0's instead, for example:

```
"%@format[4,5]"    returns "    5"  
"%@format[04,5]"   returns "0005"  
"%@format[-04,5]"  returns "5000"
```

**4NT, TC @FSTYPE[d:]:** Returns the file system type for the specified drive. FSTYPE will return "NTFS" for a drive that uses the Windows NT file system. It will return "FAT" for FAT, FAT32, and VFAT drives.

**@FULL[filename]:** Returns the full path and filename of a file. Be sure to read the notes on page 166 about quoting returned long filenames.

**@FUNCTION[name]:** Returns the contents of the specified user-defined function as a string, or a null string if the function doesn't exist. When manipulating strings returned by @FUNCTION you may need to disable certain special characters with SETDOS /X (see page 477). Otherwise, command separators, redirection characters, and other similar "punctuation" in the alias may be interpreted as part of the current command, rather than part of a simple text string.

**4NT, TC @GETDIR[d:\path]:** Pops up a dialog box to select a directory. **d:\path** specifies the initial directory; if it is not specified, GETDIR defaults to the current directory. Returns the chosen directory as a string, or an empty string if the user selects "Cancel" or presses Esc. The argument must be in quotes if it contains white space or special characters. Be sure to read the notes on page 166 about quoting returned long filenames.

**4NT, TC @GETFILE[d:\path\filename,filter]:** Pops up a dialog box to select a file. **d:\path\filename** specifies the initial directory and filename shown in the dialog, and may include wildcards. If it is not specified, GETFILE defaults to \*.\* in the current directory. Returns the full path and name of the selected file or an empty string if the user selects "Cancel" or presses Esc. The optional second argument specifies the file extension to use. You can specify multiple extensions by separating them with a semicolon. For example, **%@getfile[c:\windows,\*.exe;\*.btm]** lets the user select from .EXE and .BTM files only. The arguments must be in quotes if they contain white space or special characters. Be sure to read the notes on page 166 about quoting returned long filenames.

**4NT, TC @GETFOLDER[startdir]:** Returns a folder selected from a tree view of available directories. If you don't specify a start directory, GETFOLDER starts at "My Computer".

**4NT, TC @IDOW[date]:** Returns the 3-character abbreviation for the day of the week for the specified date, in the current locale language.

**4NT, TC @IDOWF[date]:** Returns the full name for the day of the week for the specified date, in the current locale language.

**@IF[*condition*, *true*, *false*]**: Evaluates the *condition* and returns a string based on the result. The condition can include any of the tests allowed in the IF command (see page 372). If the condition is true, @IF returns the first result string; if it is false, @IF returns the second string. For example, **%@IF[2 == 2,Correct!,Oops!]** returns "Correct!".

**@INC[%*var*]**: Returns the same value as %@EVAL[%*var* + 1]. That is, it retrieves and increments the value of a variable. The variable itself is not changed; to do so, use a command like this:

```
set var=%@inc[%var]
```

**@INDEX[*string1*,*string2*[,*offset*]]**: Returns the position of *string2* within *string1*, or "-1" if *string2* is not found. The first position in *string1* is numbered 0. The optional third parameter specifies the offset of additional matches. For example:

```
%@index[abcdeabcde,cd,2]
```

returns 7 for the offset of the second match. An offset of 0 will return the total number of matches, and a negative offset will search from right to left.

**@INIREAD[*filename*,*section*,*entry*]**: Returns an *entry* from the specified .INI file or an empty string if the *entry* does not exist. For example:

```
%@iniread[c:\tcmd\tcmd.ini,TakeCommand,history]
```

returns the size of the command history if it is specified in *TCMD.INI*. If you don't specify a path for the .INI file, @INIREAD will look in the \WINDOWS and \WINDOWS\SYSTEM directories. The section name and key name are each limited to 255 characters.

**@INIWRITE[*filename*,*section*,*entry*,*string*]**: Creates or updates an *entry* in the specified .INI file. For example:

```
%@iniwrite[c:\tcmd\tcmd.ini,TakeCommand,history,2048]
```

will set the size of the command history to 2,048 bytes the next time the Take Command is started. @INIWRITE returns "0" for success or "-1" for failure. The section name and key name are each limited to 255 characters.

**@INSERT[*n*, *string1*, *string2*]**: Inserts *string1* into *string2* starting at position *n*. The first position in the string is position 0. For example, **%@insert[1,arm,wing]** returns the string "warming."

**@INSTR[*start*, *length*, *string*]**: Returns a substring, starting at the position *start* and continuing for *length* characters. If the *length* is omitted, it will default to the remainder of the *string*. If the *length* is negative, the start is relative to the right side of the *string*. The first character in the *string* is numbered 0; if the *length* is negative, the last character is numbered 0. For example, %@INSTR[0,2,%\_TIME] gets the current time and extracts the hour; %@INSTR[1,-2,%\_TIME] extracts the seconds. If the *string* includes commas, it must be quoted with double quotes ["] or back-quotes [']. The quotes **do** count in calculating the position of the substring. @SUBSTR (page 186) is an older version of the same function.

**@INT[*n*]**: Returns the integer part of the number *n*.

**@LABEL[*d*:]**: Returns the volume label of the specified disk drive.

**@LEFT[*n*,*string*]**: Returns the leftmost *n* characters from the *string*. If *n* is greater than the length of the *string*, returns the entire *string*. For example, %@LEFT[2,jpsoft] returns the string "jp." A negative value for *n* will return all but the rightmost *n* characters.

**@LEN[*string*]**: Returns the length of a *string*.

**@LFN[*filename*]**: Returns the long filename for a short ("8.3") *filename*. In 4DOS, this function will only work if you are running under Windows 95 / 98 / ME. The *filename* may contain any valid filename element including drive letter, path, filename and extension; the entire name including all intermediate paths will be returned in long name format. Be sure to read the notes on page 166 about quoting returned long filenames

**@LINE[*filename*,*n*]**: Returns line *n* from the specified file. The first line in the file is numbered 0. "\*\*\*EOF\*\*\*" is returned for all line numbers beyond the end of the file.

The @LINE function must read each line of the file to find the line you request, and will therefore cause significant delays if used in a long loop or on a large file. For a more effective method of processing each line of a file in sequence use the FOR command (page 347), or @FILEOPEN and a sequence of @FILEREADS.

You can retrieve input from standard input if you specify CON as the filename. If you are redirecting input to @LINE using this feature, you must use command grouping (see page 117) or the redirection will not work

properly (you can pipe to @LINE without a command group; this restriction applies only to input redirection). For example:

```
(echo %@line[con,0]) < myfile.dat
```

**4NT, TC** @LINE can retrieve data from file streams on NTFS drives under Windows 2000 and XP if the stream name is specified. See NTFS File Streams on page 18 for additional details on file streams.

**@LINES[filename]:** Returns the line number of the last line in the file, or “-1” if the file is empty. The first line in the file is numbered 0, so (for example) @LINES will return 0 for a file containing one line. To get the actual number of lines, use %@INC[%@LINES[filename]]. @LINES must read each line of the file in order to count it, and will therefore cause significant delays if used on a large file.

**4NT, TC** @LINES can count lines in file streams on NTFS drives under Windows 2000 and XP if the stream name is specified. See NTFS File Streams on page 18 for additional details on file streams.

**@LOWER[string]:** Returns the *string* converted to lower case.

**4DOS** **@LPT[n]:** Returns a “1” if the specified printer is ready; otherwise, returns “0”. n=1 checks the printer connected to LPT1, n=2 checks LPT2, and n=3 checks LPT3. The value returned from @LPT may reflect the status of the printer port, or the printer itself, depending on your BIOS and printer port hardware. You may need to experiment to determine what values are returned on your system.

**@MAKEAGE[date[,time]]:** Returns the *date* and *time* (if included) as a single value in the same format as @FILEAGE. @MAKEAGE can be used to compare the time stamp of a file with a specific date and time, for example:

```
if %@fileage[myfile] lt %@makeage[1/1/85] echo OLD!
```

The value returned by @MAKEAGE can be used for comparisons, but can not be used for date and time calculations because it is not in identifiable units.

The @MAKEAGE value changes every two seconds, so, for example, %@MAKEAGE[01-01-2001,12:00:00] and %@MAKEAGE[01-01-2001,12:00:01] will return the same result.

**@MAKEDATE[n[,d]]:** Returns a date (formatted according to the current country settings). *n* is the number of days since 1/1/80. This is the inverse of @DATE. The optional second argument specifies the date format:



- |   |                     |   |                            |
|---|---------------------|---|----------------------------|
| 0 | use system default  | 3 | Japan (yy/mm/dd)           |
| 1 | USA (mm/dd/yy)      | 4 | International (yyyy-mm-dd) |
| 2 | European (dd/mm/yy) |   |                            |

**@MAKETIME[*n*]**: Returns a time (formatted according to the current country settings). *n* is the number of seconds since midnight. This is the inverse of @TIME.

**4DOS @MASTER[*varname*]**: Returns the value of a variable from the master (DOS) environment.

**@MAX[*a,b,c,...*]**: Returns the largest integer in the list.

**@MIN[*a,b,c,...*]**: Returns the smallest integer in the list.

**@MONTH[*mm-dd-yy*]**: Returns the month number for the specified date (1-12). See page 165 for information on acceptable date formats.

**@NAME[*filename*]**: Returns the base name of a file, without the path or extension. Be sure to read the notes on page 166 about quoting returned long filenames.

**@NUMERIC[*string*]**: Returns “1” if the *string* is composed entirely of digits (0 to 9), signs (+ or -), and the thousands and decimals separators. Otherwise, returns “0”. If the *string* begins with a decimal separator it is not considered numeric unless the next character is a digit, and there are no more decimal separators within the string. For example, “.07” is numeric, but “.a” and “.07.01” are not. Also see @DIGITS.

**4NT, TC @OPTION[*string*]**: Returns the current value for the specified .INI file directive. See page 205 for details on .INI file directives. All directives are supported except the key mapping directives.

**4NT, TC @PING[*host[,timeout[,packetsize]]*]**: Returns the response time in milliseconds for the specified host. *Host* is the IP address, *timeout* is the maximum number of seconds to wait, and *packetsize* is the size of the data packet sent to the host in the ping request. *timeout* defaults to 5 seconds, and *packetsize* defaults to 64 bytes. The minimum packet size is 8 bytes. If the request times out, @PING returns -1.

**@PATH[*filename*]**: Returns the path from a *filename*, including the drive letter and a trailing backslash but not including the base name or extension. Be sure to read the notes on page 166 about quoting returned long filenames.

**@RANDOM[*min*, *max*]:** Returns a random value between *min* and *max*, inclusive. *Min*, *max*, and the returned value are all integers between 0 and  $2^{31}$ . The random number generator is initialized from the system clock the first time it is used after the command processor starts, so it will produce a different sequence of random numbers each time you use it.

**@READSCR[*row*, *col*, *length*]:** Returns the text displayed at the specified location. The upper left corner of the window is location 0,0. The *row* and *column* can also be specified as an offset from the current cursor location by preceding either value with a [+] or [-]. For example:

```
%@readscr[-2,+2,10]
```

returns 10 characters from the screen, starting 2 rows above and 2 columns to the right of the current cursor position.

**@READY[*d*]:** Returns "1" if the specified drive is ready; otherwise returns "0".

**4NT, TC @REGCREATE[HKEY...\subkey]:** Create a new registry subkey. The argument starts with the root key, which can be abbreviated:

HKEY\_CLASSES\_ROOT or HKCR  
HKEY\_CURRENT\_USER or HKCU  
HKEY\_LOCAL\_MACHINE or HKLM  
HKEY\_USERS or HKU  
HKEY\_CURRENT\_CONFIG or HKCC

The remainder of the argument (after the backslash) specifies the new subkey. The entire name must be quoted if it contains any whitespace or special characters, for example:

```
@REGCREATE["HKLM\Software\My Company\My Product\User"]
```

REGCREATE will create any intermediate keys necessary. For example, @REGCREATE[HKCU\key1\key2\key3] will create all three keys (if they do not already exist). REGCREATE returns "0" if the subkey was created or "2" if an error occurred. An error message will be displayed for some errors (e.g. an invalid root key).

**4NT, TC @REGQUERY[HKEY...\subkey\value]:** Read a value from the registry. REGQUERY supports keys of type REG\_DWORD, REG\_QWORD, REG\_EXPAND\_SZ, REG\_SZ, and REG\_DWORD\_LITTLE\_ENDIAN. If the key is of type REG\_EXPAND\_SZ, the value is returned without further expansion. If the value name is not supplied, REGQUERY returns the

unnamed value for the specified key (the first value with a NULL name). See @REGCREATE (above) for information on the format of the key name. To retrieve an unnamed value, add a trailing \ after the value.

**4NT, TC @REGSET[HKEY...\subkey\value,type,data]:** Write a value to the registry. REGSET supports keys of type REG\_DWORD, REG\_QWORD, REG\_SZ, REG\_EXPAND\_SZ, and REG\_DWORD\_LITTLE\_ENDIAN. "Type" is the value type (REG\_DWORD, REG\_QWORD, REG\_EXPAND\_SZ, or REG\_SZ). "Data" is the data to set. If this argument is not supplied, @REGSET will remove the value. REGSET returns "0" if the value was written or "2" if an error occurred. See @REGCREATE (above) for information on the format of the key name.

**4NT, TC @REGSETENV[HKEY...\subkey\value,type,data]:** The same as REGSET, but a broadcast message is sent to all applications when the change is made, so that any application monitoring such messages can respond to the change immediately if it is designed to do so. REGSETENV returns "0" if the value was written or "2" if an error occurred. See @REGCREATE (above) for information on the format of the key name.

**@REMOTE[d:]:** Returns "1" if the specified drive is a remote (network) drive; otherwise returns "0".

**@REMOVABLE[d:]:** Returns "1" if the specified drive is removable (*i.e.*, a floppy disk or removable hard disk); otherwise returns "0".

**@REPEAT[c,n]:** Returns the character *c* repeated *n* times.

**@REPLACE[string1, string2, text]:** Replaces all occurrences of *string1* in the *text* string with *string2*. For example, %@replace[w,ch,warming] returns the string "charming." The search is case-sensitive.

**4NT, TC @REXX[ [=]expr]:** Calls the REXX interpreter to execute the *expression*. Returns the numeric result code from REXX, or the string result if *expr* is prefaced with an equal sign [=]. Console output from the REXX interpreter is suppressed while executing the expression. See page 145 for more information on REXX support.

**@RIGHT[n,string]:** Returns the rightmost *n* characters from the *string*. If *n* is greater than the length of the *string*, returns the entire *string*. For example, %@right[4,jpsoft] returns the string "soft." A negative value for *n* will return all but the leftmost *n* characters.

**@SEARCH[filename,path]:** Searches for the *filename* using the PATH environment variable or the specified *path*, appending an extension if one

isn't specified. (See page 19 for details on the default extensions used when searching the PATH, the order in which the search proceeds, and the search of the `\WINDOWS` and `\WINDOWS\SYSTEM` directories in Take Command.) Returns the fully-expanded name of the file including drive, path, base name, and extension, or an empty string if a matching file is not found. If wildcards are used in the **filename**, **@SEARCH** will search for the first file that matches the wildcard specification, and return the drive and path for that file plus the wildcard filename (e.g., `E:\UTIL\*.COM`).

**@SELECT[filename,top,left,bottom,right,title,1]**: Pops up a selection window with the lines from the specified file, allowing you to display menus or other selection lists from within a batch file. You can move through the selection window with the mouse or the standard popup window navigation keystrokes, including character matching (see page 35 for details; to change the navigation keys see page 224). **@SELECT** returns the text of the line the scrollbar is on if you press **Enter**, or an empty string if you press **Esc**. The **filename** must be in quotes if it contains white space or special characters. The file size is limited only by available memory. To select from lines passed through input redirection or a pipe, use **CON** as the **filename**. If you use the optional **1** argument after the window title, the list will be sorted alphabetically.

**@SFN[filename]**: Returns the fully expanded short ("8.3") filename for a long **filename**. (In 4DOS, this function will only work if you are running under Windows 95 / 98 / ME.) The **filename** may contain any valid filename element including drive letter, path, filename and extension; the entire name including all intermediate paths will be returned in short name format.

**@STRIP[chars,string]**: Removes the characters in **chars** from the **string** and returns the result. For example, `%@STRIP[AaEe,All Good Men]` returns "ll Good Mn". The test is case sensitive. To include a comma in the **chars** string, enclose the entire first argument in quotes. **@STRIP** will remove the quotes before processing the **string**.

**@SUBSTR[string,start,length]**: An older version of **@INSTR** (see page 181). The **string** parameter is at the start of the **@SUBSTR** argument list, and therefore cannot contain commas (because any commas in the string would be taken as argument separators). **@INSTR**, which has the **string** argument last, does not have this restriction.

**@TIME[hh:mm:ss]**: Returns the number of seconds since midnight for the specified time. The time must be in 24-hour format; "am" and "pm" cannot be used.

**@TIMER[*n*]**: Returns the current “split” (elapsed) time for a stopwatch started with the TIMER command (see page 506). The value of *n* specifies the timer to read and can be 1, 2, or 3.

**@TRIM[*string*]**: Returns the string with the leading and trailing white space (space and tab characters) removed.

**@TRUENAME[*filename*]**: Returns the true, fully-expanded name for a file. TRUENAME will “see through” a JOIN or SUBST. Wildcards may not be used in the filename. @TRUENAME can handle simple drive substitutions such as those created by JOIN, SUBST, or most network drive mappings. However, it may not be able to correctly determine the true name if you use “nested” JOIN or SUBST commands, or a network which does not report true names properly.

**@UNIQUE[*d:\path*]**: Creates a zero-length file with a unique name in the specified directory, and returns the full name and path. If no *path* is specified, the file will be created in the current directory. The file name will be FAT-compatible (8 character name and 3-character extension) regardless of whether the file is created on a FAT or LFN drive. This function allows you to create a temporary file without overwriting an existing file.

**4NT, TC** Rapid, repeated, consecutive invocations of @UNIQUE may occasionally return a non-unique file name (the same name twice, for example), due to a long-standing timing bug in several versions of Microsoft Windows. If you experience this problem you may need to use DELAY, DELAY /M, or BEEP (with a frequency less than 20 Hz) to provide a short delay between invocations. You may also be able to work around the problem by performing some disk I/O activity between invocations, as this can force physical creation of the file on the disk before @UNIQUE is invoked again.

**@UPPER[*string*]**: Returns the *string* converted to upper case.

**4NT, TC** **@VERINFO[*filename*,*info*,*language*]]**: Returns the version information for the specified file. The optional second parameter specifies the desired information. If it is not specified, @VERINFO defaults to “FileVersion”. The optional third parameter specifies the language/codepage pair (in hex). The default is the first code page for the default user language.

**4NT, TC** **@WATTRIB[*filename*,*-attributes*,*p*]]**: This function is the same as @ATTRIB, but supports the following additional attributes available under Windows 2000 and Windows XP:

E Encrypted                      C Compressed

<b>N</b>	Normal	<b>O</b>	Offline
<b>T</b>	Temporary	<b>I</b>	Not content-indexed
<b>S</b>	Sparse file	<b>J</b>	Junction (reparse point)

See @ATTRIB on page 169 for additional details.

**@WILD[*string1*,*string2*]:** Performs a comparison of the two strings, and returns "1" if they match or "0" if they don't match. The second argument, *string2*, may contain wildcards or extended wildcards; the first argument, *string1*, may not. The test is not case sensitive. The following example tests whether the \UTIL directory (or any directory that begins with the characters UTIL) is included in the PATH:

```
if %@wild[%path,*\UTIL*] == 1 command
```

**4NT, TC @WINCLASS[classname]:** Returns the window title of the first window with the specified class name.

**4NT, TC @WINEXENAME[title]:** Returns the executable name for the window with the specified title (you can use wildcards for the title matching). If there is more than one window that matches the title, WINEXENAME will return the topmost one.

**4NT, TC @WININFO[n]:** Returns information about the current system. "n" is a number determining which information to return:

- 1 Processor architecture (0=INTEL, 1=MIPS, 2=ALPHA, 3=PPC, 4=SHX, 5=ARM, 6=IA64, 7=ALPHA64)
- 2 Processor bit mask (set of configured processors)
- 3 Number of processors
- 4 Type of processor (i486 returns "486", all Pentiums return "586")
- 5 Processor level (Windows NT / 2000 / XP only)
- 6 Processor revision (Windows NT / 2000 / XP only)
- 7 Page size, in bytes
- 8 Allocation granularity for virtual memory, in bytes

**4NT, TC @WINMEMORY[n]:** Returns the requested Windows memory information. All values except memory load are returned in bytes. "n" is a number determining what to return:

- 0 Memory load (0 to 100)
- 1 Total physical RAM
- 2 Available physical RAM
- 3 Total that can be stored in the page file
- 4 Available page file

- 5 Total virtual memory for process
- 6 Total free virtual memory for process

**4NT, TC @WINMETRICS[n]:** Returns the requested Windows system metric. All screen dimension metrics are returned in pixels. “n” is a number determining which metric to return:

- 0 Width of screen
- 1 Height of screen
- 2 Width of arrow bitmap on horizontal scroll bar
- 3 Height of arrow bitmap on horizontal scroll bar
- 4 Height of title bar
- 5 Width of window border
- 6 Height of window border
- 7 Width of dialog box border
- 8 Height of dialog box border
- 9 Height of thumb box on vertical scroll bar
- 10 Width of thumb box on horizontal scroll bar
- 11 Width of icon
- 12 Height of icon
- 13 Width of cursor
- 14 Height of cursor
- 15 Height of single line menu bar
- 16 Width of client area for full-screen window
- 17 Height of client area for full-screen window
- 18 Height of Kanji window
- 19 Mouse present flag (0=no, 1=yes)
- 20 Height of arrow bitmap on vertical scroll bar
- 21 Width of arrow bitmap on vertical scroll bar
- 22 Debug version of Windows (0=no, 1=yes)
- 23 Left and right mouse buttons swapped (0=no, 1=yes)
- 28 Minimum width of a window
- 29 Minimum height of a window
- 30 Width of bitmaps in title bar
- 31 Height of bitmaps in title bar
- 32 Width of window frame that can be sized
- 33 Height of window frame that can be sized
- 34 Minimum tracking width of window
- 35 Minimum tracking height of window
- 41 Non-zero if Pen Windows is installed
- 42 Non-zero if DBCS version of USER.EXE is installed
- 43 Number of buttons on mouse
- 44 (Win9x) 1=security present; 0=no security

- 63 (Win9x) Non-zero if a network is installed
- 67 (Win9x) How system was started (0=normal, 1=fail-safe, 2=fail-safe w/network)
- 70 Windows will display visual info in place of audible info (0=no, 1=yes)
- 73 Computer has a slow processor (0=no, 1=yes)
- 74 Windows is set up for Arabic/Hebrew (0=no, 1=yes)
- 75 Mouse has a wheel (0=no, 1=yes)
- 76 (Win98/ME, Win200/XP) Coordinate of left side of virtual screen
- 77 (Win98/ME, Win200/XP) Coordinate of top of virtual screen
- 78 (Win98/ME, Win200/XP) Width in pixels of virtual screen
- 79 (Win98/ME, Win200/XP) Height in pixels of virtual screen
- 80 (Win98/ME, Win200/XP) Number of monitors on desktop

**4NT, TC @WINSTATE[title]:** Returns the window state of the first window with a matching title (which can include wildcards). The return values are:

- 0 Hidden
- 1 Normal
- 2 Minimized
- 3 Maximized

**4NT, TC @WINSYSTEM[parameter[,n]]:** Sets or returns the requested Windows system-wide parameters. The optional second argument “n” is the value to set. Where the selection is a state, the values are 0 for off, and 1 for on. The available parameters are:

- 1 Get beep state (0=OFF, 1=ON)
- 2 Set beep state
- 5 Get border width
- 6 Set border width
- 10 Get keyboard repeat speed (0 to 31)
- 11 Set the keyboard repeat speed
- 13 Get the width, in pixels, of an icon cell
- 13 Set the width, in pixels, of an icon cell
- 14 Get the screen saver time-out value, in seconds.
- 15 Set the screen saver time-out value, in seconds.
- 16 Get the state of the screen saver
- 17 Set the state of the screen saver
- 22 Get the keyboard repeat delay setting (0 - 3).
- 23 Set the keyboard repeat delay setting.
- 24 Get the height, in pixels, of an icon cell
- 24 Set the height, in pixels, of an icon cell
- 25 Get the icon-title wrapping state.



- 26 Set the icon title wrapping state.
- 27 Get the pop-up menu alignment (1=left aligned, 0=right aligned).
- 28 Set the pop-up menu alignment (1=right aligned, 0=left aligned).
- 29 Set the width of the double-click rectangle.
- 30 Set the height of the double-click rectangle.
- 32 Set the double-click time for the mouse, in milliseconds.
- 33 Swap or restore the meaning of the left and right mouse buttons. If n=0, the buttons are restored; if n=1, the buttons are swapped.
- 37 Get the full-window dragging state.
- 38 Set the full-window dragging state.
- 56 Get the Show Sounds accessibility flag
- 57 Set the Show Sounds accessibility flag
- 68 Get the keyboard preference state (if the user relies on the keyboard instead of the mouse).
- 69 Set the keyboard preference state.
- 70 Returns 1 if a screen reviewer utility is running.
- 71 Sets whether a screen reviewer utility is running.
- 74 Get the font smoothing feature state.
- 75 Set the font smoothing feature state.
- 76 Set the width, in pixels, of the rectangle used to detect the start of a drag operation.
- 77 Set the height, in pixels, of the rectangle used to detect the start of a drag operation.
- 79 Get the time-out value for the low-power phase of screen saving.
- 80 Return time-out value for the power-off phase of screen saving.
- 81 Set timeout value (in seconds) for the low-power phase of screen saving.
- 82 Set the time-out value for power-off phase of screen saving.
- 83 Get whether low-power phase of screen saving is enabled.
- 84 Get whether the power-off phase of screen saving is enabled.
- 85 Set the low-power phase of screen saving
- 86 Set the state of the power-off phase of screen saving.
- 87 Reload the system cursors. Set "n" to 0.
- 89 Get the locale identifier for the system default input language.
- 90 Set the default input language for the system shell and applications.
- 93 Get the Mouse Trails feature state. If the returned value is 0 or 1, the feature is disabled. If the value is greater than 1, the feature is enabled and the value indicates the number of cursors drawn in the trail.
- 94 Set the Mouse Trails feature state.
- 95 Get the snap-to-default-button feature state.
- 96 Set the snap-to-default-button feature state.

- 98 Get the width, in pixels, of the of the rectangle in which the mouse pointer has to stay to generate a WM\_MOUSEHOVER message.
- 99 Set the width, in pixels, of the of the rectangle in which the mouse pointer has to stay to generate a WM\_MOUSEHOVER message.
- 100 Get the height, in pixels, of the of the rectangle in which the mouse pointer has to stay to generate a WM\_MOUSEHOVER message.
- 101 Set the height, in pixels, of the of the rectangle in which the mouse pointer has to stay to generate a WM\_MOUSEHOVER message.
- 102 Get the time, in milliseconds, that the mouse has to stay in the hover rectangle to generate a WM\_MOUSEHOVER message.
- 103 Set the time, in milliseconds, that the mouse has to stay in the hover rectangle to generate a WM\_MOUSEHOVER message.
- 104 Get the number of lines to scroll when the mouse wheel is rotated.
- 105 Set the number of lines to scroll when the mouse wheel is rotated.
- 106 Get the time, in milliseconds, that the system waits before displaying a shortcut menu when the mouse cursor is over a submenu item.
- 107 Set the time, in milliseconds, that the system waits before displaying a shortcut menu when the mouse cursor is over a submenu item.
- 110 Get whether the IME status window is visible (per-user).
- 111 Set whether the IME status window is present.
- 112 Get the current mouse speed (1 to 20).
- 113 Set the current mouse speed (1 to 20).
- 114 Returns 1 if a screen saver is currently running.
- 4096 Get active window tracking state.
- 4097 Set active window tracking state.
- 4098 Get the menu animation feature state.
- 4099 Set the menu animation feature state.
- 4100 Get the combo box animation state.
- 4101 Set the combo box animation state.
- 4102 Get the list box smooth-scrolling effect state.
- 4103 Set the list box smooth-scrolling effect state.
- 4104 Get the gradient effect for window title bars.
- 4105 Set the gradient effect for window title vars.
- 4106 Get the menu access keys underline state.
- 4107 Set the menu access keys underline state.
- 4108 Get the active window tracking Z-order state.
- 4109 Set the active window tracking Z-order state.
- 4110 Get the hot-tracking state.
- 4111 Set the hot-tracking state.
- 4114 (Windows 2000, XP) Get the menu fade animation state.
- 4115 (Windows 2000, XP) Set the menu fade animation state.
- 4116 (Windows 2000, XP) Get the selection fade effect state.
- 4117 (Windows 2000, XP) Set the selection fade effect state.

- 4118 (Windows 2000, XP) Get the ToolTip animation state.
- 4119 (Windows 2000, XP) Set the ToolTip animation state.
- 4120 (Windows 2000, XP) Get type of tooltip animation (1 for fade, 0 for slide).
- 4121 (Windows 2000, XP) Set the type of tooltip animation.
- 4122 (Windows 2000, XP) Get the cursor shadow state.
- 4123 (Windows 2000, XP) Set the cursor shadow state.
- 4124 (Me, XP) Get the state of the Mouse Sonar feature
- 4125 (Me, XP) Set the state of the Mouse Sonar feature
- 4126 (Me, XP) Get mouse clicklock state
- 4127 (Me, XP) Set the mouse clicklock state
- 4128 (Me, XP) Get the mouse vanish feature state
- 4129 (Me, XP) Set the mouse vanish state
- 4130 (XP) Get whether native User menus have flat menu appearance.
- 4131 (XP) Set whether native User menus have flat menu appearance.
- 4132 (XP) Get the drop shadow effect state.
- 4133 (XP) Set the drop shadow effect state.
- 4158 (Windows 2000, XP) Get the state of all UI effects.
- 4159 (Windows 2000, XP) Set the UI effects state.
- 8192 Get the time following user input, in milliseconds, during which the system will not allow applications to force themselves into the foreground.
- 8193 Set the time following user input, in milliseconds, during which the system will not allow applications to force themselves into the foreground.
- 8194 Get the active window tracking delay, in milliseconds.
- 8195 Set the active window tracking delay, in milliseconds.
- 8196 Get the number of times SetForegroundWindow will flash the taskbar button when rejecting a foreground switch request.
- 8197 Set the number of times SetForegroundWindow will flash the taskbar button when rejecting a foreground switch request.
- 8198 (Windows 2000, XP) Get the caret width in edit controls (in pixels).
- 8199 (Windows 2000, XP) Set the caret width in edit controls.
- 8200 (Me, XP) Get time delay before the primary mouse button is locked.
- 8201 (Me, XP) Set time delay before the primary mouse button is locked.
- 8202 (XP) Get the type of font smoothing.
- 8203 (XP) Set the type of font smoothing (32769=standard anti-aliasing, 32770=ClearType).
- 8204 (XP) Get the contrast value used in ClearType smoothing.
- 8205 (XP) Set the contrast value (1000 to 2200).
- 8206 (XP) Get the width in pixels of the left and right edges of the focus rectangle.
- 8207 (XP) Set the width of the left and right edges of the focus rectangle.

8208 (XP) Get the height in pixels of the top and bottom edges of the focus rectangle.

8209 (XP) Set the height of the top and bottom edges of the focus rectangle.

**@WORD[["*xxx*"],*n*,*string*]**: Returns the *n*th word in the *string*. The first word is numbered 0. If *n* is negative, words are returned from the end of the *string*. You can use the first argument, "*xxx*", to specify the separators that you wish to use. If you want to use a double quote as a separator, prefix it with an escape character (see page 119). If you don't specify a list of separators, @WORD will consider only spaces, tabs, and commas as word separators. If the *string* argument is enclosed in quotation marks, you **must** enter a list of separators. For example:

```
%@word[2,NOW IS THE TIME]    returns "THE"
%@word[-0,NOW IS THE TIME]   returns "TIME"
%@word[-2,NOW IS THE TIME]   returns "IS"
%@word["=",1,2 + 2=4]        returns "4"
```

**@WORDS[["*xxx*"],*string*]**: Returns the number of words in the *string*. The optional list of delimiters follows the same format as @WORD. If the *string* argument is enclosed in quotation marks, you **must** enter a list of delimiters as well.

**4DOS**    **@XMS[*b*|*k*|*m*]**: Returns the amount of free XMS memory.

**@YEAR[*mm-dd-yy*]**: Returns the year for the specified date. See page 165 for information on acceptable date formats. The year can be specified as two digits or four digits; @YEAR returns the same number of digits included in its argument.

### ***Variable Function Examples***

You can use variable functions in a wide variety of ways depending on your needs. We've included a few examples below to give you an idea of what's possible. For a more comprehensive set of examples see the online help, or the *EXAMPLES.BTM* file which came with your command processor.

To set the prompt to show the amount of free memory (see the PROMPT command, page 437, for details on including variable functions in your prompt):

```
[c:\] prompt (%%@dosmem[K]K) $p$g
```

Set up a simple command-line calculator. The calculator is used with a command like `CALC 3 * (4 + 5)`:

```
[c:\] alias calc `echo The answer is: %@eval[%$]`
```

The following batch file uses variable functions to implement “once a day” execution of a group of commands. It works by constructing a 6-digit number “yymmdd” from today's date, and comparing that to a number of the same type stored in the file `C:\ONCEADAY.DAT`. If today's date is numerically larger than the saved date, and the time is after 6:00 AM, then the “once a day” commands are run, and today's date is saved in the file as the new date for comparison. Otherwise, no action is taken. You can make this file simpler using the `%@DATE` and `%@TIME` functions instead of using `%@INSTR` to extract substrings of the `%_DATE` and `%_TIME` variables; we used the approach shown to demonstrate the use of `%@INSTR`.

```
rem Temporary variables used to shorten example lines:
rem DD is _date, DY is yymmdd date, TM is _time
set dd=%_date
set dy=%@instr[6,2,%dd] %@instr[0,2,%dd] %@instr[3,2,%dd]
set lastdate=0
iff exist c:\onceaday.dat then
    set lastdate=%@line[onceaday.dat,0]
endiff
iff %dy gt %lastdate then
    set tm=%_time
    iff "%@instr[0,2,%tm] %@instr[3,2,%tm]" gt "0600" then
        rem Commands to be executed once a day go here
        echo %dy > c:\onceaday.dat
    endiff
endiff
```

## **Special Character Compatibility**

If you use two or more of our products, or if you want to share aliases and batch files with users of different products, you need to be aware of the differences in three important characters: the Command Separator (see page 76), the Escape Character (page 119), and the Parameter Character (page 127).

The default values of each of these characters in each product is shown in the following table:

<u>Product</u>	<u>Separator</u>	<u>Escape</u>	<u>Parameter</u>
----------------	------------------	---------------	------------------

4NT, Take Command	&	^	\$
4DOS	^	↑	&

(In this section, an up-arrow [↑] is used for the ASCII Ctrl-X character, numeric value 24. The actual appearance of control characters on your screen depends on the font you use.)

In your batch files and aliases, and even at the command line, you can smooth over these differences in three ways:

- ✓ Select a consistent set of characters on the “Syntax” tab of the configuration dialog (see page 50), or with *.INI* file directives (page 204). For example, to set the 4DOS characters to match 4NT and Take Command, use these lines in *4DOS.INI*:

```
CommandSep = &
EscapeChar = ^
ParameterChar = $
```

- ✓ Use internal variables that contain the current special character, rather than using the character itself (see page 153). For example, this command:

```
if "%1" == "" (echo Argument missing! ^ quit)
```

will only work if the command separator is a caret. However, this version works regardless of the current command separator:

```
if "%1" == "" (echo Argument missing! %+ quit)
```

- ✓ In a batch file, use the SETLOCAL command (see page 480) to save the command separator, escape character, and parameter character when the batch file starts. Then use SETDOS as described below to select the characters you want to use within the batch file. Use an ENDLOCAL command (page 333) at the end of the batch file to restore the previous settings.

You can also use the SETDOS command (see page 472) to change special characters on the command line. However, when setting new special character values on the command line you must take into account the possibility that one of your new values will have a current meaning that causes problems with the setting. For example, this command:

```
[c:\] setdos /e^
```

would **not** set the escape character to a caret [^] in 4DOS if the standard 4DOS special characters were currently in effect. The ^ would be seen as a command separator, and would terminate the SETDOS command before the escape character was set. To work around this, use the escape character variable %= before each setting to ensure that the following character is not treated with any special meaning.

For example, the following sequence of commands in a batch file will always set the special characters correctly to their standard 4NT and Take Command values, no matter what their current setting, and will restore them when the batch file is done:

```
setlocal
setdos /c%=& /e%=^ /p%=$
.....
endlocal
```

A similar sequence can be used to select the standard 4DOS characters, regardless of the current settings:

```
setlocal
setdos /c%=^ /e%=↑ /p%=&
.....
endlocal
```

## ***Command Parsing***

Whenever you type something at the command line and press the **Enter** key, or include a command in a batch file, you have given a command to the command processor which must figure out how to execute it. If you understand the general process that is used, you will be able to make the best use of the commands. Understanding these steps can be especially helpful when working with complex aliases or batch file commands.

To decide what activity to perform, the command processor goes through several steps. Before it starts, it writes the entire command line (which may contain multiple commands) to the history log file if history logging has been enabled with the LOG /H command (see page 410), and the command did not come from a batch file. Then, if the line contains multiple commands, the first command is isolated for processing.

4DOS, 4NT, and Take Command begin by dividing the command into a **command name** and a **command tail**. The command name is the first

word in the command; the tail is everything that follows the command name. For example, in the command line:

```
dir *.txt /2/p/v
```

the command name is “dir”, and the command tail is “ \*.txt /2/p/v”.

Next the command processor, tries to match the command name against its list of aliases. If it finds a match between the command name and one of the aliases you've defined, it replaces the command name with the contents of the alias. This substitution is done internally and is not normally visible to you; however, you can view a command line with aliases expanded by pressing **Ctrl-F** after entering the command at the prompt.

If the alias included parameters (%1, %2, etc.), the parameter values are filled in from the text on the command line, and any parameters used in this process are removed from the command line. The process of replacing a command name that refers to an alias with the contents of the alias, and filling in the alias parameters, is called **alias expansion**.

This expansion of an alias creates a new command name: the first word of the alias. This new command name is again tested against the list of aliases, and if a match is found the contents of the new alias is expanded just like the first alias. This process, called **nested alias expansion**, continues until the command name no longer refers to an alias.

Once it has finished with the aliases, the command processor next tries to match the command name with its list of internal commands. If it is unsuccessful, it knows that it will have to search for a batch file or external program to execute your command.

The next step is to locate any batch file or alias parameters, environment variables, internal variables, or variable functions in the command, and replace each one with its value. This process is called **variable expansion**.

The variable expansion process is modified for certain internal commands, like EXCEPT, IF, and GLOBAL. These commands are always followed by another command, so variable expansion takes place separately for the original command and the command that follows it.

Once all of the aliases and environment variables have been expanded, the command processor will echo the complete command to the screen (if command-line echo has been enabled) and write it to the log file (if command logging has been turned on).



Before it can actually execute your command, the command processor must scan the command tail to see if it includes redirection or piping. If so, the proper internal switches are set to send output to an alternate device or to a file, instead of to the screen. In 4NT and Take Command, a second process is started at this point, if necessary, to receive any piped output.

Finally, it is time to execute the command. If the command name matches an internal command, the command processor will perform the activities you have requested. Otherwise, the command processor searches for an executable (*.COM* or *.EXE*) file, a batch file, or a file with an executable extension that matches the command name (see the detailed description of this search on page 19).

Once the internal command or external program has terminated, the command processor saves the result or exit code that the command generated, cleans up any redirection that you specified, and then returns to the original command line to retrieve the next command. When all of the commands in a command line are finished, the next line is read from the current batch file, or if no batch file is active, the prompt is displayed.

You can disable and re-enable several parts of command parsing (for example alias expansion, variable expansion, and redirection) with the SETDOS /X command (see page 477).

## ***Argument Quoting***

As it parses the command line, the command processor looks for command separators (carets [^] in 4DOS, or ampersands [&] in 4NT and Take Command), conditional commands [| | or &&), white space (spaces, tabs, and commas), percent signs [%] which indicate variables to be expanded, and redirection and piping characters (>, <, or |).

Normally, these special characters cannot be passed to a command as part of an argument. However, you can include any of the special characters in an argument by enclosing the entire argument in back-quotes [`] or double quotes ["]. Although both back-quotes and double quotes will let you build arguments that include special characters, they do not work the same way.

No alias or variable expansion is performed on an argument enclosed in back-quotes. Redirection symbols inside the back-quotes are ignored. The back-quotes are removed from the command line before the command is executed.

No alias expansion is performed on expressions enclosed in double quotes. Redirection symbols inside double quotes are ignored. However, variable

expansion is performed on expressions inside double quotes. The double quotes themselves will be passed to the command as part of the argument.

For example, suppose you have a batch file *CHKNAME.BTM* which expects a name as its first parameter (%1). Normally the name is a single word. If you need to pass a two-word name with a space in it to this batch file you could use the command:

```
[c:\] chkname `MY NAME`
```

Inside the batch file, %1 will have the value MY NAME, including the space. The back-quotes caused the command processor to pass the string to the batch file as a single argument.

For a more complex example, suppose the batch file *QUOTES.BAT* contains the following commands:

```
@echo off
echo Arg1 = %1
echo Arg2 = %2
echo Arg3 = %3
```

and the environment variable FORVAR has been defined with this command:

```
[c:\] set FORVAR=for
```

Now, if you enter the command:

```
[c:\] quotes `Now is the time %forvar` all good
```

the output from *QUOTES.BAT* will look like this:

```
Arg1 = Now is the time %forvar
Arg2 = all
Arg3 = good
```

But if you enter the command

```
[c:\] quotes "Now is the time %forvar" all good
```

the output from *QUOTES.BAT* will look like this:

```
Arg1 = "Now is the time for"
Arg2 = all
Arg3 = good
```

Notice that in both cases, the quotes keep characters together and reduce the number of arguments in the line.

The following example has 7 command-line arguments, while the examples above only have 3:

```
[c:\] quotes Now is the time %%forvar all good
```

(The double percent signs are needed because the argument is parsed twice, once when passed to the batch file and again in the ECHO command.)

When an alias is defined in a batch file or from the command line, its argument can be enclosed in back-quotes to prevent the expansion of replaceable parameters, variables, and multiple commands until the alias is invoked. See ALIAS on page 250 for details.

You can disable and re-enable back-quotes and double quotes with the SETDOS /X command (see page 477).



## CHAPTER 6 / CONFIGURATION

Part of the power of 4DOS, 4NT, and Take Command is their flexibility. You can alter their configuration to match your style of computing. Most of the configuration of 4DOS, 4NT, and Take Command is controlled through a file of initialization information with the extension *.INI*, which is discussed in this chapter. This file is called *4DOS.INI* in 4DOS, *4NT.INI* in 4NT, and *TCMD32.INI* in Take Command. (For brevity, we refer to all three files as **.INI files** throughout this manual.) We also discuss many ways of configuring your command processor in other parts of this manual, for example:

- ✓ With **aliases** you can set default options for internal commands and create new commands (see pages 122 and 250).
- ✓ With **executable extensions** you can associate data files with the applications you use to open them (see page 106).
- ✓ With the FILECOMPLETION environment variable, and the FileCompletion *.INI* directive (explained below) you can customize filename completion to match the command you are working with.
- ✓ With the COLORDIR environment variable and the ColorDir *.INI* directive you can set the colors used by the DIR (in 4DOS, 4NT, and Take Command) and SELECT (in 4DOS and 4NT) commands.
- ✓ With the SETDOS command (see page 472), you can change some aspects of the command processor's operation "on the fly."
- ✓ With command-line options (see the *Introduction and Installation Guide*) you can specify where 4DOS, 4NT, or Take Command looks for its startup files and how the command processor operates for a specific instance.

### Modifying the .INI File

You can create, add to, and modify the *.INI* file in two ways: with the built-in configuration dialog or the OPTION command, or by editing the file with any ASCII editor.

The configuration dialog allows you to modify the settings that are used most often. It is accessible via the OPTION command, or, in Take Command, from the **Configure Take Command** selection on the **Options** menu. When you exit from the dialog you can select the **OK** button to save your changes in the

*.INI* file for use in the current shell and all future shells, or discard the changes you have made by selecting the **Cancel** button.

Most of the changes you make in the configuration dialog take effect immediately. A few (*e.g.*, those associated with the startup screen size, or initialization of the alias list) only take effect when you reboot your system (under DOS), or close the current process and start a new Take Command, 4NT, or 4DOS process (under Windows). See the online help for each individual dialog page if you are not sure when a change will take effect.

The configuration dialog handles most standard *.INI* file settings. The Advanced directives, the Key Mapping directives, and a few other individual directives noted below do not have corresponding fields in the configuration dialog, and must be entered manually. For more details on the dialog, see page 50. For details about the **OPTION** command, which can be used to set individual options or to open the configuration dialog, see page 427.

You can also create, add to, and edit the *.INI* file “manually” with any ASCII text editor. Each command processor reads its *.INI* file when it starts, and configures itself accordingly. The *.INI* file is not re-read when you change it manually. For manual changes to take effect, you must reboot your system (under DOS), or start a new command processor session (under Windows). If you edit the *.INI* file manually, make sure you save the file in ASCII format.

Each item that you can include in the *.INI* file has a default value. You only need to include entries in the file for settings that you want to change from their default values.

**TC** The *TCMD32.INI* file has several sections. All of the directives described here go into the **[TakeCommand]** section, which is usually first in the file. You can edit this section manually. Take Command uses other sections to record information you set while you are using it, including its window size and position, the font you are using, and the buttons you create on the tool bar. You should use Take Command's menu commands to change the settings in these other sections of the *.INI* file instead of editing them directly.

### **Using the *.INI* File**

Some settings in the *.INI* file are initialized when you install 4DOS, 4NT, and Take Command. Others (such as window size and position) are modified as you use the command processor, so you will probably have an *.INI* file even if you didn't create one yourself. You should not delete this file.

4DOS, 4NT, and Take Command search for the *.INI* file in three places:

- ✓ If there is an “@d:\path\inifile” option on the SHELL= line in *CONFIG.SYS* (for 4DOS) or on the startup command line (for all products), the command processor will use the path and file name specified there, and will not look elsewhere. See the *Introduction and Installation Guide* for details about the startup command line.
- ✓ If there is no *.INI* file name on the SHELL= line or startup command line, the search proceeds to the same directory where the command processor program file (*4DOS.COM*, *4NT.EXE* or *TCMD32.EXE*) is stored. This is the “normal” location for the *.INI* file. 4NT and Take Command determine this directory automatically. 4DOS determines it from the COMSPEC directory name on the SHELL= line in *CONFIG.SYS*. See your *Introduction and Installation Guide* for further details on setting the COMSPEC directory.
- ✓ If the *.INI* file is not found in the directory where the program file is stored, a final check is made in the root directory of the boot drive.

When 4DOS, 4NT, or Take Command is loaded as a secondary shell, it does not search for the *.INI* file. Instead, it retrieves the primary shell's *.INI* file data, processes the **[Secondary]** section of the original *.INI* file if necessary, and then processes any “@d:\path\inifile” option on the secondary shell command line (see your *Introduction and Installation Guide* for details). You can override this behavior with the NextINIFile directive (see page 236).

## **.INI File Sections**

The *.INI* file has three possible sections: the first or **global** section, named after your command processor (**[4DOS]**, **[4NT]**, or **[TakeCommand]**); the **[Primary]** section; and the **[Secondary]** section. Each section is identified by the section name in square brackets on a line by itself.

Directives in the global section are effective in all shells. In most cases, this is the only section you will need, particularly for 4NT and Take Command. Any changes you make to the *.INI* file with the OPTION command are stored in the global section.

The **[Primary]** and **[Secondary]** sections include directives that are used only in primary and secondary shells, respectively (see page 9 for more information on primary and secondary shells). You don't need to set up these sections unless you want different directives for primary and secondary shells.

Directives in the **[Primary]** section are used for the first or primary shell. The values are passed automatically to all secondary shells, unless overridden by a directive with the same name in the **[Secondary]** section.

Directives in the **[Secondary]** section are used in secondary shells only, and override any corresponding primary shell settings. For example, these lines in the *.INI* file:

```
[Primary]
ScreenRows = 25
[Secondary]
ScreenRows = 50
```

mean to assume that you have 25 rows on the screen in the primary shell and 50 lines in all secondary shells.

Sections that begin with any name other than the product name (**[4DOS]**, **[4NT]**, or **[TakeCommand]**), **[Primary]**, or **[Secondary]** are ignored.

### **.INI File Directives**

Most lines in the *.INI* file consist of a one-word **directive**, an equal sign [=], and a **value**. For example, in the following line, the word “History” is the directive and “2048” is the value:

```
History = 2048
```

Any spaces before or after the equal sign are ignored.

If you have a long string to enter in the *.INI* file (for example, for the ColorDir directive), you must enter it all on one line. Strings cannot be “continued” to a second line. Each line may be up to 511 characters long in 4DOS, or 2047 characters long in 4NT and Take Command.

The format of the **value** part of a directive line depends on the individual directive. It may be a numeric value, a single character, a choice (like “Yes” or “No”), a color setting, a key name, a path, a filename, or a text string. The value begins with the first non-blank character after the equal sign and ends at the end of the line or the beginning of a comment.

Blank lines are ignored in the *.INI* file and can be used to separate groups of directives. You can place comments in the file by beginning a line with a semicolon [;]. You can also place comments at the end of any line except one containing a text string value. To do so, enter at least one space or tab after the value, a semicolon, and your comment, like this:



```
History = 2048           ;set history list size
```

If you try to place a comment at the end of a string value, the comment will become part of the string and will probably cause an error.

If you use the configuration dialog to modify the *.INI* file, comments on lines modified from within the dialogs will not be preserved when the new lines are saved. To be sure *.INI* file comments are preserved, put them on separate lines in the file.

When the command processor detects an error while processing the *.INI* file, it displays an error message and prompts you before processing the remainder of the file. This allows you to note any errors before the startup process continues. The directive in error will retain its previous or default value. Only the most catastrophic errors (like a disk read failure) will terminate processing of the remainder of the *.INI* file. If you don't want a pause after each error, use a "PauseOnError = No" directive at the beginning of the *.INI* file.

If you need to test different values for an *.INI* directive without repeatedly editing the *.INI* file, use the **OPTION** command or see the **INIQuery** directive on page 212.

If you want to include the text of one *.INI* file within another (for example, if you have a set of common directives used by several JP Software products), see the **Include** directive on page 236.

The **SETDOS** command can override several of the *.INI* file directives (see page 472). For example, the cursor shape used by 4DOS, 4NT, and Take Command can be adjusted either with the **CursorIns** and **CursorOver** directives or the **SETDOS /S** command. The correspondence between **SETDOS** options and *.INI* directives is noted under each directive below, and under each option of the **SETDOS** command.

Secondary shells automatically inherit the configuration settings currently in effect in the previous shell. If values have been changed by **SETDOS** since the primary shell started, the current values will be passed to the secondary shell. If the previous shell's *.INI* file had a **[Secondary]** section, it will then be read and processed. If not, the previous shell's settings will remain in effect.

For example, you might set **BatchEcho** to **Yes** in the *.INI* file, to enable batch file echo. If you then use **SETDOS /V0** to turn off batch file echoing in the primary shell, then any secondary shells will inherit the **SETDOS** setting,

rather than the original value from the *.INI* file; *i.e.*, batch files in the secondary shell will default to no echo.

If you want to force secondary shells to start with a specific value for a particular directive, regardless of any changes made with SETDOS in a previous shell, repeat the directive in the **[Secondary]** section of the *.INI* file.

**4DOS** When you start a secondary shell (*e.g.* from the Windows desktop) you can specify an alternate location and name for *4DOS.INI* by passing the “@d:\path\inifile” option to 4DOS as a command-line parameter (see your *4DOS Introduction and Installation Guide* for details). In this case, the configuration settings in the alternate *4DOS.INI* file will supersede any settings inherited from the previous shell. Any values which are not explicitly set in the alternate file will retain the value they had in the previous shell. The first section below lists the different types of directives. Subsequent sections list all the individual directives, divided by function and then alphabetically by directive name.

At the end of this chapter, we have included a few examples of how to use the *.INI* file directives (see page 238).

### ***Types of Directives***

There are 8 types of directives in the *.INI* file. The different types of directives are shown in the lists below as follows:

- ✓ **Name = nnnn (1234):** This directive takes a numeric value which replaces the “nnnn.” The default value is shown in parentheses.
- ✓ **Name = c (X):** This directive accepts a single character as its value. The default character is shown in parentheses. You must type in the actual character; you cannot use a key name.
- ✓ **Name = CHOICE1 | Choice2 | ... :** This directive takes a choice value. The possible choices are listed, separated by vertical bars. The default value is shown in all upper case letters in the directive description, but in your file any of the choices can be entered in upper case or lower case. For example, if the choices were shown as “YES | No” then “YES” is the default.
- ✓ **Name = Color:** This directive takes a color specification. See page 30 for the format of color names.

- ✓ **Name = Key (Default):** This directive takes a key specification. See page 34 for the format of key names.
- ✓ **Name = Path:** This directive takes a path specification, but not a filename. The value should include both a drive and path (e.g., *C:\TCMD*) to avoid any possible ambiguities. A trailing backslash [*\*] at the end of the path name is acceptable but not required. Any default path is described in the text.
- ✓ **Name = File:** This directive takes a filename. We recommend that you use a full filename including the drive letter and path to avoid any possible ambiguities. Any default filename is described in the text.
- ✓ **Name = String:** This directive takes a string in the format shown. The text describes the default value and any additional requirements for formatting the string correctly. **No comments are allowed.**

4DOS, 4NT, and Take Command each contain a fixed-length area for storing strings entered in the *.INI* file, including file names, paths, and other strings. This area is large and is unlikely to overflow; if it does, you will receive an error message. If this occurs, reduce the complexity of your *.INI* file or contact our technical support department for assistance.

Each directive is listed below with its name, type, and default value. In some cases, no default is shown because the default value varies between 4DOS, 4NT, and Take Command; in these cases, the default is described in the text.

## Initialization Directives

The directives in this section control how your command processor starts and where it looks for its files.

4DOS, 4NT	<b>4StartPath</b> = Path: Sets the drive and directory where the <i>4START</i> and <i>4EXIT</i> batch files (if any) are located.
4DOS	<b>Alias</b> = nnnn (1024): Sets the amount of memory in bytes allocated for the alias list. This directive is not needed in 4NT or Take Command because both adjust the size of the alias list dynamically.
4DOS	<b>AutoExecParms</b> = String: Sets the parameter or parameters to be passed to <i>AUTOEXEC.BAT</i> , or the file specified with the <i>AutoExecPath</i> directive, above, when 4DOS is started as a primary

shell with the /P option in *CONFIG.SYS* (see your *Introduction and Installation Guide*). The parameters will be available in your *AUTOEXEC.BAT* file as %1, %2, etc.

### 4DOS

**AutoExecPath** = Path | File: Sets the path used to find *AUTOEXEC.BAT* if 4DOS is started as a primary shell with the /P option in *CONFIG.SYS* (see your *Introduction and Installation Guide*). If you include only a path, 4DOS will look for *AUTOEXEC.BAT* in the specified directory. If you include a complete file name, 4DOS will look for the specified file, and will not look for *AUTOEXEC.BAT*. The default is the file *AUTOEXEC.BAT* in the root directory of the boot drive.

Using AutoExecPath with a complete file name lets you store multiple startup files in a single directory. You may find this useful under DOS if you have multiple boot software and use different startup files under different configurations.

### TC

**ConsoleColumns** = nnnn (80); **ConsoleRows** = nnnn (100): These directives set the width and height of the screen buffer used for the Take Command console window (see your Take Command *Introduction and Installation Guide* or the online help for more information on the console window). The width range is 80 - 132 columns; the height range is 25 - 4096 rows. These directives control the text buffer used for the console window, **not** the actual dimensions of the window on your screen. The window dimensions are determined by the operating system, not Take Command, and may be smaller than the text buffer (if they are, scroll bars are provided to allow viewing of all text columns). Due to operating system limitations under Windows 95 and 98, **ConsoleColumns** is ignored and the window is fixed at 80 columns, and **ConsoleRows** is ignored unless it is set to 25 or 50.

**DirHistory** = nnnn (1024): Sets the amount of memory allocated to the directory history in bytes. The allowable range of values is 256 to 2048 bytes in 4DOS, and 256 to 32767 bytes in 4NT and Take Command. If you use a global directory history list (see page 75), the DirHistory value is ignored in all shells except the shell which first establishes the global list.

### 4NT, TC

**DuplicateBugs** = Yes | NO: Tells the parser to duplicate certain well-known bugs in *CMD.EXE*. The only bug currently replicated is in the IF command (see page 372 for details).

<b>4DOS</b>	<b>Environment</b> = nnnn (512): Sets the amount of memory allocated to the environment in bytes. The allowable range of values is 160 to 32767 bytes. This directive is not needed in 4NT or Take Command because both adjust the size of the environment dynamically.
<b>4DOS</b>	<b>EnvFree</b> = nnnn (128): Sets the minimum amount of memory in bytes that will be available in the environment for secondary shells. 4DOS will enlarge the environment for each secondary shell, if necessary, so that there is at least this much free environment space when the shell starts. The allowable range of values is 128 to 32767 bytes.
<b>4NT, TC</b>	<b>FirewallHost=name</b> : The server name of the firewall for FTP and HTTP commands.
<b>4NT, TC</b>	<b>FirewallPassword=name</b> : The password if the firewall requires authentication.
<b>4NT, TC</b>	<b>FirewallType=0   1   2   3</b> : The type of the firewall: <div><ul style="list-style-type: none"><li>0 None</li><li>1 Tunnel</li><li>2 SOCKS4</li><li>3 SOCKS5</li></ul></div>
<b>4NT, TC</b>	<b>FirewallUser=name</b> : The user name if the firewall requires authentication.
<b>4DOS</b>	<b>Function</b> = nnnn (1024): Sets the amount of memory in bytes allocated for the user-defined function list. This directive is not needed in 4NT or Take Command because both adjust the size of the function list dynamically.
<b>4DOS</b>	<b>HelpOptions</b> = String: Sets default options for the 4DOS help system. For the available options, see the <b>Help Reference</b> topic in the 4DOS online help.
<b>TC</b>	<b>HideConsole</b> = YES   No: Normally Take Command will hide the console window after running a DOS or Win32 console mode program. If HideConsole is set to <b>No</b> , the console window will only be hidden until the first character-mode application you run; after that it remains on-screen. This option works on all screens, but is primarily intended for high-resolution screens where you can resize the console and Take Command windows to run side by side. See

your Take Command *Introduction and Installation Guide* or the online help for more information on the console window.

**History** = nnnn (1024): Sets the amount of memory allocated to the command history list in bytes. The allowable range of values is 256 to 32767 bytes. If you use a global history list (see page 65), the History value is ignored in all shells except the shell which first establishes the global list.

TC

**IBeamCursor** = YES | No: If set to **Yes**, Take Command will display the standard “I-Beam” cursor in text areas of its window. If IBeamCursor is set to **No**, an arrow is used in all areas of the window (this can be helpful on systems where the I-Beam cursor is hard to see).

**INIQuery** = Yes | NO: If set to **Yes**, a dialog box will be displayed before execution of each subsequent line in the current *.INI* file. This allows you to modify certain directives when you start the command processor in order to test different configurations. INIQuery can be reset to **No** at any point in the file. Normally INIQuery = Yes is only used during testing of other *.INI* file directives.

4DOS, 4NT

The prompt generated by INIQuery = Yes is:

```
[contents of the line] (Y/N/Q/R/E) ?
```

At this prompt, you may enter:

```
Y = Yes:   Process this line and go on to the next.
N = No:    Skip this line and go on to the next.
Q = Quit:  Skip this line and all subsequent lines.
R = Rest:  Execute this and all subsequent lines.
E = Edit:  Prompt for a new value for this entry.
```

If you choose E for Edit, you can enter a new value for the directive, but not a new directive name.

TC

The dialog displayed when INIQuery = Yes gives you three options:

```
Yes       Executes the directive
No        Skips the directive
Cancel    Executes the directive and all remaining
          directives in the [TakeCommand] section of
```

the *.INI* file (*i.e.*, cancels the `INIQuery = Yes` setting)

**4DOS**

**InstallPath = Path:** Sets the path used to find the help system, *OPTION.EXE*, and other 4DOS files. This directive is normally set by the installation program, and should not be changed unless you move the 4DOS files to a different directory.

**LocalAliases = Yes | No:** The default value is **Yes** in 4DOS and **No** in 4NT and Take Command. **No** forces all copies of the command processor to share the same alias list. **Yes** keeps the lists for each shell separate. See page 259 for more details on local and global alias lists.

**LocalDirHistory = Yes | No:** The default value is **Yes** in 4DOS and **No** in 4NT and Take Command. **No** forces all copies of the command processor to share the same directory history. **Yes** keeps the directory histories for each shell separate. See page 75 for more details on local and global directory histories.

**LocalFunctions = Yes | No:** The default value is **Yes** in 4DOS and **No** in 4NT and Take Command. **No** forces all copies of the command processor to share the same user-defined function list. **Yes** keeps the lists for each shell separate.

**TC**

**LocalHistory = Yes | No:** The default value is **Yes** in 4DOS and **No** in 4NT and Take Command. **No** forces all copies of the command processor to share the same history list. **Yes** keeps the lists for each shell separate. See page 65 for more details on local and global history lists.

**4NT, TC**

**MailAddress=name:** The email address of the current user, used in SENDMAIL for outgoing mail . If not set, SENDMAIL will attempt to get the address from the registry.

**4NT, TC**

**MailPort=n:** The port number to use for SMTP (the default is 25).

**4NT, TC**

**MailServer=name:** The local SMTP server name to use in SENDMAIL for outgoing mail . If not set, SENDMAIL will attempt to get the address from the registry.

**4NT, TC**

**PassiveFTP=YES | no:** Passive FTP mode is usually required if you have a firewall. You should only set PassiveFTP to **NO** if you

are having FTP connection problems, or if you must use the PORT command instead of the PASV command.

**PauseOnError** = YES | No: **Yes** forces a pause with the message “Error in *filename*, press any key to continue processing” after displaying any error message related to a specific line in the *.INI* file. **No** continues processing with no pause after an error message is displayed.

**4NT, TC**      **Proxy** = name: Specifies the proxy server to use for HTTP calls. HTTP can be used with COPY (page 280) and MOVE (page 416).

**4NT, TC**      **ProxyPort** = n: Specifies the port number to use for the proxy server to use. The default is 80.

**4DOS**      **REXXPath** = File: Specifies the program that 4DOS will use to execute *.BAT* files that begin with the characters *[/\*]*. Specify a full path and filename if the program is not in your PATH. Under PC DOS 7 and above, REXXPath defaults to *REXX.EXE*, the REXX interpreter included with the operating system. If *REXX.EXE* is not in your PATH under PC DOS 7, you must use this directive to specify the location of the REXX interpreter. See page 145 for more information on REXX support.

**4DOS**      **Swapping** = swap type [, swap type] ...: Sets the type of swapping 4DOS should use. 4DOS runs in two parts, a resident portion that is always in memory and a transient portion that is “swapped” to EMS memory, XMS memory, a RAM disk, or your hard disk while application programs are running. The swap area for the transient portion normally requires about 200K bytes of memory or disk space for the primary shell, and 50K bytes for each secondary shell.

The *swap type* can be **EMS** to swap to EMS expanded memory, **XMS** to swap to XMS extended memory, **d:\path** to swap to the specified drive and path, or **None** for no swapping.

If you use **d:\path**, 4DOS will create the file *4DOSSWAP.nnn* in the specified directory. The file name changes if UniqueSwapName is set to Yes (see page 238). If you use **None**, the transient portion of 4DOS will remain in memory at all times; this will reduce memory available for application programs by about 250K.

You can specify multiple swap types and 4DOS will try them in the order listed. Swap type “None” is always appended to your list of



possible swap types as a “last resort,” even if you don't include it explicitly. The default Swapping specification is **EMS, XMS, x:\, None**, where **x** is the boot drive. In most cases you will find that the default setting works without modification.

Before changing the Swapping specification, see the **Miscellaneous Reference** section of the 4DOS online help for more detail on swapping types and the way 4DOS uses memory.

If you use disk swapping in the primary shell under Windows 95 / 98, 4DOS shells run from inside Windows will not inherit aliases, *.INI* settings, the command history, or the directory history from the primary shell. This is due to a restriction in Windows 95/98.

**TC**                      **ScreenBufSize** = nnnn (200000): Sets the size of the screen scrollbar buffer in bytes. The allowable range is from 50000 to 1000000 bytes.

**TC**                      **TCStartPath** = Path: Sets the drive and directory where the *TCSTART* and *TCEXIT* automatic batch files (if any) are located. See page 132 for details about these batch files.

**4NT, TC**                **TimeServer** = name: Specifies the name of the internet time server to use for TIME /S. The default is “clock.psu.edu”.

**TreePath** = Path: Sets the location of *JPSTREE.IDX*, the file used for extended directory searches (see pages 81 and 83 for information about extended searches). By default, the file is placed in the root directory of drive C:.

**4DOS**                      **UMBAlias** = Yes | NO | 1 | 2 ... | 8: **Yes** attempts to load global alias list storage into a UMB (Upper Memory Block). If you use a specific region number (**1** through **8**), 4DOS will attempt to reserve room for the global alias list in that UMB region.

Region numbers can be used under MS-DOS / PC DOS 5.0 and above. To use them, you must enable DOS UMB management with the *DOS=UMB* or *DOS=HIGH,UMB* directive in *CONFIG.SYS*. See the **Miscellaneous Reference** section of the 4DOS online help for complete details on UMBs and UMB regions, and the hardware and software required to support them.

If you use an invalid region number, or if region numbers have not been enabled on your system, 4DOS will load the global alias list

into the first available region. If no upper memory is available, space for the global alias list will be reserved in low memory.

UMBAlias applies to global aliases only, and is only used in the first shell which establishes the global alias area (see page 259 for more information on local and global alias lists). If you specify LocalAliases = Yes, or if the previous shell already created a global alias area, any UMBAlias setting is ignored.

- 4DOS**      **UMBDirHistory**= Yes | NO | 1 | 2 ... | 8: **Yes** attempts to load global directory history list storage into a UMB (Upper Memory Block). See page 75 for more information on local and global directory history lists. See UMBAlias (above) for information on the use of UMBs and region numbers. Like UMBAlias, UMBDirHistory will be ignored if a global directory history list is not used.
- 4DOS**      **UMBEnvironment** = Yes | NO | 1 | 2 ... | 8: **Yes** attempts to load the master environment into a UMB (Upper Memory Block). This reduces 4DOS's base memory requirements but may cause problems with some programs that try to access the master environment directly. See UMBAlias (above) for information on the use of UMBs and region numbers.
- 4DOS**      **UMBFunction** = Yes | NO | 1 | 2 ... | 8: **Yes** attempts to load the user-defined function list into a UMB (Upper Memory Block). This reduces 4DOS's base memory requirements but may cause problems with some programs that try to access the master environment directly. See UMBAlias (above) for information on the use of UMBs and region numbers.
- 4DOS**      **UMBHistory** = Yes | NO | 1 | 2 ... | 8: **Yes** attempts to load global history list storage into a UMB (Upper Memory Block). If you use a specific region number (1 through 8), 4DOS will attempt to reserve room for the global history list in that UMB region. See page 65 for more information on local and global history lists. See UMBAlias (above) for information on the use of UMBs and region numbers. Like UMBAlias, UMBHistory will be ignored if a global history list is not used.
- 4DOS**      **UMBLoad** = Yes | NO | 1 | 2 ... | 8: **Yes** attempts to load the resident portion of 4DOS into a UMB (Upper Memory Block). This reduces the size of the resident portion in base memory from about

3K bytes to 256 bytes. See **UMBAlias** (above) for information on the use of UMBs and region numbers.

- 4NT, TC**      **WindowState** = STANDARD | Maximize | Minimize | Custom: initial state of the 4NT or Take Command window. **Standard** puts the window in the default position on the Windows desktop, and is the default setting. **Maximize** maximizes the window; **Minimize** minimizes it, and **Custom** sets it to the position specified by the **WindowX**, **WindowY**, **WindowWidth**, **WindowHeight** directives.
- 4NT, TC**      **WindowX** = nnnn, **WindowY** = nnnn, **WindowWidth** = nnnn, **WindowHeight** = nnnn: These 4 directives set the initial size and position of the 4NT or Take Command window. The measurements are in pixels or pels. **WindowX** and **WindowY** refer to the position of the top left corner of the window relative to the top left corner of the screen. These directives will be ignored unless **WindowState** is set to **Custom**.

## Configuration Directives

These directives control the way that 4DOS, 4NT, and Take Command operate. Some can be changed with the SETDOS command while the command processor is running. Any corresponding SETDOS command is listed in the description of each directive; information on SETDOS begins on page 472.

**AmPm** = Yes | NO | Auto: **Yes** displays times in 12-hour format with a trailing “a” for AM or “p” for PM. The default of **No** forces a display in 24-hour time format. **Auto** formats the time according to the country code set for your system. **AmPm** controls the time format used by DIR and SELECT, in LOG files, and in the output of the TIMER, DATE, and TIME commands. It has no effect on %\_TIME, %@MAKETIME, the \$t and \$T options of PROMPT, or date and time ranges.

- 4DOS**      **ANSI** = AUTO | Yes | No: Tells 4DOS whether an ANSI driver is installed and should be used for the CLS and COLOR commands. 4DOS normally determines this itself, but if you are using a non-standard ANSI driver or your loading sequence is unusual, you may need to explicitly inform 4DOS. Also see SETDOS /A.
- 4NT,TC**      **ANSI** = Yes | NO: Sets the initial state of ANSI support. **Yes** enables ANSI string processing in the 4NT / Take Command window. **No** disables ANSI strings. Note that enabling ANSI will

only affect 4NT and Take Command output, not the output of external applications. Caveman output in Take Command will use the ANSI default colors if the application is set to use “default colors” or “stdio”. See the ANSI Codes Reference in the Reference section of the online help for a reference list of the ANSI sequences supported by 4NT and Take Command. Also see SETDOS /A, and the `_ANSI` internal variable on page 156.

**AppendToDir** = Yes | NO: If set to **Yes**, a trailing backslash [`\`] will be appended to directory names when doing filename completion. (If you also have the UnixPaths `.INI` directive set, the command processor will add a trailing forward slash to directory names.) Regardless of the setting of this directive, a trailing backslash is always appended to a directory name at the beginning of the command line to enable automatic directory changes.

**BatchEcho** = YES | No: Sets the default batch ECHO mode. **Yes** enables echoing of all batch file commands unless ECHO is explicitly set off in the batch file. **No** disables batch file echoing unless ECHO is explicitly set on. Also see SETDOS /V.

**BeepFreq** = nnnn (440): Sets the default BEEP command frequency in Hz. This is also the frequency for “error” beeps (for example, if you press an illegal key). To disable all error beeps, set this or BeepLength to 0. If you do, the BEEP command will still be operable, but will not produce sound unless you explicitly specify the frequency and duration.

**BeepLength** = nnnn (2): Sets the default BEEP length in system clock ticks (approximately 1/18 of a second per tick). BeepLength is also the default length for “error” beeps (for example, if you press an illegal key).

**CDDWinLeft, CDDWinTop, CDDWinWidth, CDDWinHeight** = nnnn: These values set the initial position and size of the popup window used by extended directory searches (see page 83), in characters, including the border. The defaults are 3, 3, 72, and 16, respectively (*i.e.*, a window beginning in column 3, row 3, 72 columns wide and 16 rows high). The position is relative to the top left corner of the screen. The width and height values include the space required for the window border. The window cannot be smaller than 10 columns wide by 5 rows high (including the border). The values you enter will be adjusted if necessary to keep a minimum-size window visible on the screen. In 4DOS and 4NT,

the window is normally displayed with a shadow, but if you specify a window starting at column 0 and extending to the right margin, the shadow is eliminated; this may be useful to prevent speech software from reading text in the shadow area while viewing the window.

**CommandSep** = c: This is the character used to separate multiple commands on the same line. The default is the caret [^] in 4DOS, and the ampersand [&] in 4NT and Take Command. You cannot use any of the redirection characters (| > < ) or any of the white space characters (space, tab, comma, or equal sign). The command separator is saved by SETLOCAL and restored by ENDLOCAL. Also see SETDOS /C, the %+ internal variable on page 156, and page 195 for information on using compatible command separators for two or more products.

**CompleteHidden** = Yes | NO: If set to **Yes**, filename completion will return hidden and system files and directories.

**CopyPrompt** = Yes | NO: If set to **Yes**, a COPY or MOVE will prompt before overwriting an existing file if the command is being performed at the command prompt. This duplicates the behavior of later versions of COMMAND.COM and CMD.EXE.

TC

**CUA** = Yes | NO: If set to **Yes**, Take Command will use the Common User Access (CUA) standard keys for cut and paste: Ctrl-Del for cut, Ctrl-Ins for copy, and Shift-Ins for paste. If set to the default value of "No," Take Command will use the Windows non-CUA editing keys: Ctrl-X for cut, Ctrl-C for copy, and Ctrl-V for paste. Note that if set to "No," these keys will not be available for their normal usage (e.g., Ctrl-C for interrupting commands).

**CursorIns** = nnnn (100 in 4DOS and 4NT, 15 in TC): This is the shape of the cursor for insert mode during command-line editing and all commands which accept line input (DESCRIBE, ESET, etc.). The size is a percentage of the total character cell size, between 0% and 100%. Because of the way video drivers map the cursor shape, you may not get a smooth progression in cursor shapes as **CursorIns** and **CursorOver** change. If **CursorIns** or **CursorOver** is set to -1, the command processor will not attempt to modify the cursor shape at all; you can use this feature to give another program full control of the cursor shape. If you set them to 0 in 4DOS or 4NT, the cursor will be invisible. You can retrieve the

current cursor shape values with the **%\_CI** and **%\_CO** internal variables (see page 158). Also see **SETDOS /S**.

**CursorOver** = nnnn (15 in 4DOS and 4NT, 100 in Take Command): This is the shape of the cursor for overstrike mode during command-line editing and all commands which accept line input. The size is a percentage of the total character cell size, between 0% and 100%. Also see **SETDOS /S**.

**DecimalChar** = . | , | **AUTO**: Sets the character used as the decimal separator for **@EVAL**, numeric **IF** and **IFF** tests, version numbers, and other similar uses. The only valid settings are period **[.]**, comma **[,]**, and **Auto** (the default). A setting of **Auto** tells the command processor to use the decimal separator associated with your current country code. If you change the decimal character you must also adjust the thousands character (with **ThousandsChar**, see below) so that the two characters are different. Also see **SETDOS /G**.

**4NT, TC**

**DelGlobalQuery** = **YES** | **No**: Enable or disable the confirmation prompt from **DEL /Q** when doing a wildcard-only or directory deletion. See **DEL** on page 292 for additional details. **Use caution if you set DelGlobalQuery to No**, as this will allow **DEL /Q** to delete an entire directory without prompting for confirmation.

**DescriptionMax** = nnnn (511): Controls the description length limit for **DESCRIBE** (see page 299). The allowable range is 20 to 511 characters.

**DescriptionName** = File: Sets the file name in which to store file descriptions. The default file name is **DESCRIPT.ION**. **Use this directive with caution** because changing the name from the default will make it difficult to transfer file descriptions to another system. Also see **SETDOS /D**.

**Descriptions** = **YES** | **No**: Turns description handling on or off during the file processing commands **COPY**, **DEL**, **MOVE**, and **REN**. If set to **No**, the command processor will not update the description file when files are moved, copied, deleted or renamed. Also see **SETDOS /D**.

**EditMode** = **INSERT** | **Overstrike** | **InitOverstrike** | **InitInsert**: This directive lets you start the command-line editor in either insert or overstrike mode. If you specify **InitOverstrike** or

**InitInsert**, the command line editor will start in the specified state, but if you toggle insert mode while editing a line, the editor will continue to use the new mode on subsequent lines. Also see SETDOS /M.

**TC**

**Editor** = File: Specifies the path and filename of the program that Take Command will execute when you select "Editor" from the Utilities menu. The default is *NOTEPAD.EXE*.

**EscapeChar** = c : Sets the character used to suppress the normal meaning of the following character. The default is Ctrl-X [↑] in 4DOS and a caret [^] in 4NT and Take Command. See page 119 for a description of the escape character and special escape sequences. You cannot use any of the redirection characters (|, >, or <) or the white space characters (space, tab, comma, or equal sign) as the escape character. The escape character is saved by SETLOCAL and restored by ENDLOCAL. Also see SETDOS /E, the %= internal variable on page 156, and page 195 for information on using compatible escape characters for two or more products.

**EvalMax** = nnnn (10): Controls the maximum number of digits after the decimal point in values returned by @EVAL (see page 171). This setting can be overridden with the construct @EVAL[expression=x.y]. The allowable range is 0 to 10. Also see SETDOS /F.

**EvalMin** = nnnn (0): Controls the minimum number of digits after the decimal point in values returned by @EVAL (see page 171). The allowable range is 0 to 10. EvalMin will be ignored if it is larger than EvalMax. This setting can be overridden with the construct @EVAL[expression=x.y]. Also see SETDOS /F.

**4NT, TC**

**ExecWait** = Yes | NO: Controls whether 4NT and Take Command wait for an external program started from the command line to complete before redisplaying the prompt. See **Waiting for Applications to Finish** on page 114 for details on the effects of this directive.

**FileCompletion** = cmd1: ext1 ext2 ...; cmd2: ext3 ext4 ... Sets the files made available during filename completion for selected commands. The format is the same as that used for the FILECOMPLETION environment variable. See page 70 for a detailed explanation of selective filename completion.

**FuzzyCD** = 0 | 1 | 2 | 3: Enables or disables extended directory searches, and controls their behavior. A setting of 0 (the default) disables extended searches. For complete details on the meaning of the other settings see Extended Directory Searches on page 83.

**HistCopy** = Yes | NO: Controls what happens when you re-execute a line from the command history. If this option is set to **Yes**, the line is appended to the end of the history list. By default, or if this option is set to **No**, the command is not copied. The original copy of the command is always retained at its original position in the list, regardless of the setting of HistCopy. Set this option to No if you want to use HistMove = Yes; otherwise, the HistCopy setting will override HistMove.

**HistDups** = OFF | First | Last: Controls the history list duplicate removal. If this option is set to **First**, and a new entry matches an older one, the old entry is preserved and the new entry discarded. If set to **Last**, the new entry is saved and the old entry deleted. If set to **Off**, everything is saved to the history.

**HistLogName** = File: Sets the history log file name and / or path. If only a path is given, the default log file name (4DOSHLOG, 4NTHLOG, TC32HLOG) will be used. Using HistLogName does not turn history logging on; you must use a LOG /H ON command to do so.

**HistLogOn** = Yes | NO: If set to **Yes**, history logging is turned on when the command processor starts.

**HistMin** = nnnn (0): Sets the minimum command-line size to save in the command history list. Any command line whose length is less than this value will not be saved. Legal values range from 0, which saves everything, to the maximum command-line length plus 1 (*i.e.*, 256 in 4DOS, or 2,047 in 4NT and Take Command), which disables all command history saves.

**HistMove** = Yes | NO: If set to **Yes**, a recalled line is moved to the end of the command history. The difference between this directive and HistCopy, above, is that HistCopy = Yes copies each recalled line to the end of the history but leaves the original in place. HistMove = Yes places the line at the end of history and removes the original line. This directive has no effect if HistCopy = Yes.



**HistWrap** = YES | No: Controls whether the command history “wraps” when you reach the top or bottom of the list. The default setting enables wrapping, so the list appears “circular”. If HistWrap is set to No, history recall will stop at the beginning and end of the list rather than wrapping. This setting affects history recall at the prompt only; the command history window never wraps.

**4DOS**

**LineInput** = Yes | NO: This directive controls how 4DOS gets its input from the command line. **Yes** forces 4DOS to perform line-by-line input, just as *COMMAND.COM* does, instead of character-by-character input. This will disable command-line editing, history recall, the directory history window, and filename completion. It is normally used only for rare memory-resident programs (TSRs) or applications which do not work properly unless the command processor uses line input. If you have a particular program that requires line input, you can use SETDOS /L to temporarily change modes. See the **Compatibility** section of the online help for information on any programs which require this option.

**ListRowStart** = 1 | 0: Specifies whether LIST and FFIND consider the first line in a file line "1" or line "0". The default is "1".

**LogErrors** = Yes | NO: If set to **Yes**, error messages will be written to the log file when logging is enabled (see LOG on page 410 for information on logging).

**LogName** = File: Sets the log file name and / or path. If only a path is given, the default log file name (*4DOSLOG*, *4NTLOG*, *TC32LOG* etc.) will be used. Using LogName does not turn logging on; you must use a LOG ON command to do so. See LOG on page 410 for information on logging.

**4DOS**

**Mouse** = AUTO | yes | no: Specifies whether a mouse is available for popup boxes. If you don't have a mouse, or if the driver takes a long time for the query/reset call, you should explicitly set this value to No.

**NoClobber** = Yes | NO: If set to **Yes**, will prevent standard output redirection (see page 88) from overwriting an existing file, and will require that the output file already exist for append redirection. Also see SETDOS /N.

**PathExt** = NO | Yes: Determines whether the command processor will use the PATHEXT environment variable. If set to **No** (the default), the PATHEXT variable is ignored. If set to **Yes**, the PATHEXT variable will be used to determine extensions to look for when searching the PATH for an executable file. For details, see the PATHEXT variable (page 152) and the PATH command (page 429).

PATHEXT is supported for compatibility reasons but should not generally be used as a substitute for the more flexible executable extensions feature (see page 106).

**ParameterChar** = c: Sets the character used after a percent sign to specify all or all remaining command-line arguments in a batch file or alias (e.g., %\$ or %n\$; see pages 128 and 257). The default is the ampersand [&] for 4DOS and the dollar sign [\$] for 4NT and Take Command. The parameter character is saved by SETLOCAL and restored by ENDLOCAL. Also see SETDOS /P. See page 195 for information on using compatible parameter characters for two or more products.

**PopupWinLeft, PopupWinTop, PopupWinWidth, PopupWinHeight** = nnnn: These values set the initial position and size of the command-line, directory history, and filename completion windows, and most other popup windows (see **CDDWinLeft** etc. for the extended directory search window). The values are in characters, and include the border. The defaults are 40, 1, 36, and 12, respectively (*i.e.*, a window beginning in column 40, row 1, 36 columns wide and 12 rows high). The position is relative to the top left corner of the window. The width and height values include the space required for the window border. The window cannot be smaller than 10 columns wide by 5 rows high (including the border). The values you enter will be adjusted if necessary to keep a minimum-size window visible on the screen. In 4DOS and 4NT, the window is normally displayed with a shadow, but if you specify a window starting at column 0 and extending to the right margin, the shadow is eliminated; this may be useful to prevent speech software from reading text in the shadow area while viewing the window.

#### **4DOS**

**Printer** = devicename: Sets the output device that the LIST command will print to. By default, LPT1 is used. The device can be PRN, LPT1 to 3, COM1 to 4, NUL (which will disable printed output) or any other installed character device.

- 4NT, TC**      **RecycleBin** = YES | No: If set to **Yes**, files deleted by the DEL and ERASE commands (see page 292), and by RD /S (page 445) are placed in the Windows Recycle Bin by default. If set to **No**, the files are deleted without being placed in the Recycle Bin. DEL's and RD's /K and /R switches allow you to override this setting for individual commands.
- 4DOS**      **ScreenColumns** = nnnn: Sets the number of columns used by the video display. Normally the screen size is determined automatically, but if you have a non-standard display you may need to set it explicitly. Systems which need to set OutputBIOS to Yes (see page 237) may also need to use this directive.
- TC**      **ScreenColumns** = nnnn (80): Sets the number of virtual screen columns used by the Take Command window. If the virtual screen width is greater than the physical window width, Take Command will display a horizontal scrollbar at the bottom of the window. See page 39 for more information on the virtual screen size.
- 4DOS**      **ScreenRows** = nnnn: Sets the number of screen rows used by the video display. Normally the screen size is determined automatically, but if you have a non-standard display you may need to set it explicitly. This value does not affect screen scrolling, which is controlled by your operating system, or (under DOS) your video BIOS or ANSI driver. ScreenRows is used only by the LIST and SELECT commands, the paged output options of other commands (*e.g.*, TYPE /P), and error checking in the screen output commands. Also see SETDOS /R.
- TC**      **ScreenRows** = nnnn (25): Sets the initial height of the Take Command window. See page 39 for more information on screen size.
- 4NT, TC**      **ServerCompletion** = None | LOCAL | Global: Specifies how server name completion should proceed (see Filename Completion on page 67 for information on how to use server name completion). The default of **Local** lists only local servers (*i.e.*, those in your "network neighborhood"). **Global** will enumerate the entire network. **None** will disable server completion; this may be necessary to prevent "hanging" if you start typing a server name and accidentally press Tab, and your local domain is very large or slow to respond.

**TC**                    **StatusBarText** = nnnn (8): Sets the point size of the text on the status bar. The allowable range is 4 to 16.

**TC**                    **StatusBarOn** = YES | No: **Yes** enables the status bar when Take Command starts. **No** disables it. The status bar can be enabled or disabled while Take Command is running by using the **Options** or **Setup** menu. The **StatusBarOn** setting is automatically updated to reflect the current state of the status bar each time Take Command exits.

**TC**                    **SwapScrollKeys** = Yes | NO: Yes switches to 4DOS/4NT-style keystrokes for manipulating the scrollbar buffer.

If **SwapScrollKeys** is set to Yes, the **Up** and **Down** arrow keys will scroll through the command history list and the **PgUp** key will pop up the history window. The **Ctrl-Up**, **Ctrl-Down**, **Ctrl-PgUp**, and **Ctrl-PgDn** keys will scroll the text in the screen buffer.

If **SwapScrollKeys** is set to No, these keys will assume their default meanings. The **Up** and **Down** arrow keys and the **PgUp** and **PgDn** keys will scroll the text in the screen buffer. The **Ctrl-Up** and **Ctrl-Down** keys will scroll through the command history list and the **Ctrl-PgUp** key will pop up the history window.

For additional details see Scrolling and History Keystrokes on page 120.

Do **not** set **SwapScrollKeys** to Yes if you use key mapping directives to reassign the scrolling or history keys individually. **SwapScrollKeys** takes effect before other key mappings, and using both methods at the same time will be confusing at best. Setting **SwapScrollKeys** to Yes has essentially the same effect as including the following individual key mapping directives in the *.INI* file:

```
PrevHistory = Up
NextHistory = Down
HistWinOpen = PgUp
HistWinOpen = PgDn
ScrollUp = Ctrl-Up
ScrollDown = Ctrl-Down
ScrollPgUp = Ctrl-PgUp
ScrollPgDn = Ctrl-PgDn
```

**TabStops** = nnnn (8): Sets the tab stops for Take Command's output, or for LIST in 4DOS and 4NT. The allowable range is 1 to 32.

**ThousandsChar** = . | , | AUTO: Sets the character used as the thousands separator for numeric output. The only valid settings are period [.] , comma [,] , and **Auto** (the default). A setting of Auto tells the command processor to use the thousands separator associated with your current country code. If you change the thousands character you must also adjust the decimal character (with DecimalChar, see above) so that the two characters are different. Also see SETDOS /G.

**TC** **ToolBarOn** = YES | No: **Yes** enables the tool bar when Take Command starts. **No** disables it. The tool bar can be enabled or disabled while Take Command is running by using the **Options** or **Setup** menu. The ToolBarOn setting is automatically updated to reflect the current state of the tool bar each time Take Command exits.

**TC** **ToolBarText** = nnnn (8): Sets the point size of text on the tool bar. The allowable range is 4 to 16.

**UnixPaths** = Yes | NO: Enables the forward slash as a path separator in the command name (the first item on the command line). This allows you to enter a command like:

```
[c:\] /bin/programs/foo.exe
```

without having the forward slashes interpreted as switch characters. Note that setting UnixPaths to Yes does **not** change the command processor or operating system switch character, it's still '/'. It simply allows you to put forward slashes in the command name without problems.

When UnixPaths is set to Yes command switches beginning with a forward slash must be preceded by a space to avoid confusion (this is a good general practice regardless of the setting of UnixPaths). For example:

```
[c:\] \bin\foo.exe /c      OK
[c:\] /bin/foo.exe /c      OK
[c:\] \bin\foo.exe/c       Error
[c:\] /bin/foo.exe/c       Error
```

### 4NT, TC

**UpdateTitle** = YES | No: 4NT and Take Command normally change the title in the title bar to include the command or batch file name each time a new command is executed. If you prefer a static title bar which does not change with each command, set UpdateTitle to **No** to prevent these updates.

**UpperCase** = Yes | NO: **Yes** specifies that file and directory names should be displayed in the traditional upper-case by internal commands like COPY and DIR. **No** allows the normal 4DOS, 4NT, and Take Command lower-case style. This directive does not affect the display of filenames on drives which support long filenames; see page 15 for more details. Also see SETDOS /U.

**Win95SFNSearch** | **Win32SFNSearch** = YES | No: Set to **No** to disable the automatic search for short filenames after long filenames in Windows. This directive is called Win95SFNSearch in 4DOS, and Win32SFNSearch in 4NT and Take Command. See page 105 for details.

### Color Directives

These directives control the colors that 4DOS, 4NT, and Take Command use for their displays. Screen border colors (only supported in 4DOS full-screen) can only be set in StdColors. “BORDER” color specifications included in other directives will be ignored. For complete details on color names see page 30.

### 4DOS

**BrightBG** = Yes | No: If set to **Yes**, 4DOS will enable bright background colors. If set to **No**, bright backgrounds will be disabled, but blinking foreground characters will be enabled. If BrightBG is not used, 4DOS will not adjust the bright background / blinking foreground switch at all. Most color video boards default to a blinking foreground with bright background colors disabled. See also SETDOS /B.

Using BrightBG requires careful attention to interactions of display type, mode, and color. For a detailed explanation, see page 32.

**CDDWinColors** = Color: Sets the default colors for the popup window used by extended directory searches (see page 83). If this directive is not used, the colors will be reversed from the current colors on the screen

**ColorDir** = ext1 ext2 ...:colora;ext3 ext4 ... :colorb; ...: Sets the directory colors used by DIR (and SELECT with 4DOS and 4NT).

The format is the same as that used for the COLORDIR environment variable. See page 310 for a detailed explanation of color-coded directories.

**InputColors** = Color: Sets the colors used for command-line input. This setting is useful for making your input stand out from the normal output. InputColors will **not** work properly under 4DOS unless you have an ANSI driver loaded (see page 28 for more information on ANSI).

**4DOS, 4NT**

**ListboxBarColors** = Color: Sets the color for the highlight bar in the popup list boxes (*i.e.*, command history window, filename completion window, @SELECT window, etc.).

**ListColors** = Color: Sets the colors used by the LIST command. If this directive is not used, LIST will use the current default colors set by the CLS or COLOR command or by the StdColors directive.

**4DOS, 4NT**

**ListStatBarColors** = Color: Sets the colors used on the LIST status bar. If this directive is not used, LIST will set the status bar to the reverse of the screen color (the screen color is controlled by ListColors, above).

**PopupWinColors** = Color: Sets the default colors for the command-line, directory history, and filename completion windows, and most other popup windows (see **CDDWinColors** for the extended directory search window). If this directive is not used, the colors will be reversed from the current colors on the screen

**SelectColors** = Color: Sets the color used by the SELECT command. If this directive is not used, SELECT will use the current default colors set by the CLS or COLOR command or by the StdColors directive.

**4DOS, 4NT**

**SelectStatBarColors** = Color: Sets the color used on the SELECT status bar. If this directive is not used, SELECT will set the status bar to the reverse of the screen color (the screen color is controlled by SelectColors, above).

**StdColors** = Color: Sets the standard colors to be used when CLS is used without a color specification, and for LIST and SELECT if ListColors and SelectColors are not used. Using this directive is similar to placing a COLOR command in *AUTOEXEC.BAT* (in 4DOS) or *4START/ TCSTART*. StdColors takes effect the first

time CLS, LIST, or SELECT is used after 4DOS, 4NT or Take Command starts, but will not affect the color of error or other messages displayed during the loading and initialization process.

Under 4DOS, and when using 4NT under Windows 95/98/ME, if *ANSI.SYS* or a compatible driver is not loaded the colors will not be “sticky” – you may lose them when you run an application. See page 28 for additional details.

### ❖ *Key Mapping Directives*

These directives allow you to change the keys used for command-line editing and other internal functions. They take effect only inside 4DOS, 4NT or Take Command and do not affect other programs or the help system. They cannot be entered via the configuration dialog; you must enter them manually (see page 203 for details).

The description of each directive below explains the function of the corresponding key. Using the directive allows you to assign a different or additional key to perform the function described. For example, to use function key **F3** to invoke the HELP facility (normally invoked with **F1**):

```
Help = F3
```

Any directive can be used multiple times to assign multiple keys to the same function. For example:

```
ListFind = F      ;F does a find in LIST
ListFind = F4     ;F4 also does a find in LIST
```

- ! Use some care when you reassign keystrokes. If you assign a default key to a different function, it will no longer be available for its original use. For example, if you assign **F1** to the AddFile directive (a part of filename completion), the **F1** key will no longer invoke the help system, so you will probably want to assign a different key to Help.

Please read the information on key names beginning on page 34 before using the key mapping directives.

Key assignments are processed before looking for keystroke aliases. For example, if you assign **Shift-F1** to HELP and also assign **Shift-F1** to a key alias, the key alias will be ignored.

Assigning a new keystroke for a function does not deassign the default keystroke for the same function. If you want to deassign one of the default



keys, use the **NormalKey** directive described below or the corresponding directive for keys in the other key groups (**NormalEditKey**, **NormalHWinKey**, or **NormalListKey**).

### **General Input Keys**

This first set of Key Mapping Directives applies to all input. These directives are in effect whenever 4DOS, 4NT, or Take Command requests input from the keyboard, including during command-line editing and the **DESCRIBE**, **ESET**, **INPUT**, **LIST**, and **SELECT** commands. See page 58 for more information about command-line editing.

**Backspace** = Key (Bksp): Deletes the character to the left of the cursor.

**BeginLine** = Key (Home): Moves the cursor to the beginning of the line.

**Del** = Key (Del): Deletes the character at the cursor.

**DelToBeginning** = Key (Ctrl-Home): Deletes from the cursor to the start of the line.

**DelToEnd** = Key (Ctrl-End): Deletes from the cursor to the end of the line.

**DelWordLeft** = Key (Ctrl-L): Deletes the word to the left of the cursor.

**DelWordRight** = Key (Ctrl-R, Ctrl-Bksp): Deletes the word to the right of the cursor. See **ClearKeyMap** on page 236 if you need to remove the default mapping of **Ctrl-Bksp** to this function.

**Down** = Key (Down): Scrolls the display down one line in **LIST**; moves the cursor down one line in **SELECT** and in the command-line history, directory history, or @**SELECT** window.

**EndLine** = Key (End): Moves the cursor to the end of the line.

**EraseLine** = Key (Esc): Deletes the entire line.

**ExecLine** = Key (Enter): Executes or accepts a line.

**Ins** = Key (Ins): Toggles insert / overstrike mode during line editing.

**Left** = Key (Left): Moves the cursor left one character on the input line; scrolls the display left 8 columns in **LIST**; scrolls the display left 4 columns in the command-line history, directory history, or @**SELECT** windows.

**NormalKey = Key:** Deassigns a general input key in order to disable the usual meaning of the key within 4DOS, 4NT or Take Command, and / or make it available for keystroke aliases. This will make the keystroke operate as a “normal” key with no special function. For example:

```
NormalKey = Ctrl-End
```

will disable Ctrl-End, which is the standard “delete to end of line” key. Ctrl-End could then be assigned to a keystroke alias. Another key could be assigned the “delete to end of line” function with the DelToEnd directive (above).

**Right = Key (Right):** Moves the cursor right one character on the input line; scrolls the display right 8 columns in LIST; scrolls the display right 4 columns in the command-line history, directory history, or @SELECT window.

**Up = Key (Up):** Scrolls the display up one line in LIST; moves the cursor up one line in SELECT and in the command-line history, directory history, or @SELECT window.

**WordLeft = Key (Ctrl-Left):** Moves the cursor left one word; scrolls the display left 40 columns in LIST.

**WordRight = Key (Ctrl-Right):** Moves the cursor right one word; scrolls the display right 40 columns in LIST.

### ***Command-Line Editing Keys***

The following directives apply only to command-line editing. They are only effective at the 4DOS, 4NT, or Take Command prompt.

**AddFile = Key (F10 in 4DOS and 4NT, Ctrl-Shift-Tab in Take Command):** Keeps the current filename completion entry and inserts the next matching name.

**AliasExpand = Key (Ctrl-F):** Expands all aliases in the current command line without executing them.

**CommandEscape = Key (Alt-0255):** Allows direct entry of a keystroke that would normally be handled by the command line editor (*e.g.* **Tab** or **Ctrl-D**).

**DelHistory = Key (Ctrl-D):** Deletes the displayed history list entry and displays the previous entry.

**EndHistory** = Key (Ctrl-E): Displays the last entry in the history list.

**Help** = Key (F1): Invokes the HELP facility.

**LFNToggle** = Key (Ctrl-A): Toggles filename completion between long filename and short filename modes on LFN drives.

**LineToEnd** = Key (Ctrl-Enter): Copies the current command line to the end of the history list, then executes it.

**NextFile** = Key (F9, Tab): Gets the next matching filename during filename completion. See **ClearKeyMap** on page 236 if you need to remove the default mapping of **Tab** to this function.

**NextHistory** = Key (Ctrl-Down): Recalls the next command from the command history. Also see **Scrolling and History Keystrokes** on page 120 and **SwapScrollKeys** on page 226.

**NormalEditKey** = Key: Deassigns a command-line editing key in order to disable the usual meaning of the key while editing a command line, and / or make it available for keystroke aliases. For details see the **NormalKey** directive on page 232.

**PopFile** = Key (F7, Ctrl-Tab): Opens the filename completion window. You may not be able to use **Ctrl-Tab**, because not all systems recognize it as a keystroke. See **ClearKeyMap** on page 236 if you need to remove the default mapping of **Ctrl-Tab** to this function.

**PrevFile** = Key (F8, Shift-Tab): Gets the previous matching filename during filename completion. See **ClearKeyMap** on page 236 if you need to remove the default mapping of **Shift-Tab** to this function.

**PrevHistory** = Key (Ctrl-Up): Recalls the previous command from the command history. Also see **Scrolling and History Keystrokes** on page 120 and **SwapScrollKeys** on page 226.

**RepeatFile** = Key (F12): Repeats the previous matching filename during filename completion.

**SaveHistory** = Key (Ctrl-K): Saves the command line in the command history list without executing it.

### ***Scrollbar Buffer Keys in Take Command***

The following keys are also part of the command line editing group. They are used to manipulate the scrollbar buffer rather than to edit commands. For additional information see *Scrolling and History Keystrokes* on page 120 and *SwapScrollKeys* on page 226. (To deassign one of the scrollbar keys use the *NormalEditKey* directive above.)

**ScrollUp** = Key (Up): Scrolls the Take Command scrollbar buffer up one line.

**ScrollDown** = Key (Down): Scrolls the Take Command scrollbar buffer down one line.

**ScrollPgUp** = Key (PgUp): Scrolls the Take Command scrollbar buffer up one page.

**ScrollPgDn** = Key (PgDn): Scrolls the Take Command scrollbar buffer down one page.

### ***Popup Window Keys***

The following directives apply to popup windows, including the command history window, the directory history window, the filename completion window, the extended directory search window, and the @SELECT window.

**DirWinOpen** = Key (F6 in Take Command, Ctrl-PgUp in 4NT and 4DOS): Opens the directory history window while at the command line.

**HistWinOpen** = Key (PgUp in 4NT and 4DOS, Ctrl-PgUp in Take Command): Brings up the history window while at the command line. Also see *Scrolling and History Keystrokes* on page 120 and *SwapScrollKeys* on page 226.

**NormalPopupKey** = Key: Deassigns a popup window key in order to disable the usual meaning of the key within the popup window. For details see the *NormalKey* directive on page 232.

**PopupWinBegin** = Key (Ctrl-PgUp): Moves to the first item in the list when in the popup window.

**PopupWinDel** = Key (Ctrl-D): Deletes a line from within the command history or directory history window.

**PopupWinEdit** = Key (Ctrl-Enter): Moves a line from the command history or directory history window to the prompt for editing.

**PopupWinEnd** = Key (Ctrl-PgDn): Moves to the last item in the list when in the popup window.

**PopupWinExec** = Key (Enter): Selects the current item and closes the window.

### ***LIST Keys***

The keys in this group are effective only inside the LIST command.

**ListExit** = Key (Esc): Exits from the LIST command.

**ListFind** = Key (F): Prompts and searches for a string.

**ListFindReverse** = Key (Ctrl-F): Prompts and searches backward for a string.

**ListHex** = Key (X): Toggles hexadecimal display mode.

**ListHighBit** = Key (H): Toggles LIST's "strip high bit" option, which can aid in displaying files from certain word processors.

**ListInfo** = Key (I): Displays information about the current file.

**ListNext** = Key (N): Finds the next matching string.

**ListPrevious** = Key (Ctrl-N): Finds the previous matching string.

**ListPrint** = Key (P): Prints the file (to LPT1 in 4DOS, to the selected printer in 4NT and Take Command).

**ListWrap** = Key (W): Toggles LIST's wrap option on and off. The wrap option wraps text at the right margin.

**NormalListKey** = Key: Deassigns a LIST key in order to disable the usual meaning of the key within LIST. For details see the NormalKey directive on page 232.

### **❖ *Advanced Directives***

These directives are generally used for unusual circumstances, or for diagnosing problems. Most often they are not needed in normal use. They

cannot be entered via the configuration dialog; you must enter them manually.

### ***Advanced Directives For 4DOS, 4NT, and Take Command***

The directives in this section work in 4DOS, 4NT and Take Command.

**ClearKeyMap:** Clears all current key mappings. ClearKeyMap is a special directive which has no value or “=” after it. Use ClearKeyMap to make one of the keys in the default map (**Tab**, **Shift-Tab**, **Ctrl-Tab**, or **Ctrl-Bksp**) available for a keystroke alias. ClearKeyMap should appear before any key mapping directives. If you want to clear some but not all of the default mappings, use ClearKeyMap, then recreate the mappings you want to retain (e.g., with “NextFile=Tab”, etc.).

**Debug = nnnn (0):** Controls certain debugging options which can assist you in tracking down unusual problems. See the online help for details.

**Include = File:** Include the text from the named file at this point in the processing of the current *.INI* file. Use this option to share a file of directives between several products. The text in the named file is processed just as if it were part of the original *.INI* file. When the include file is finished, processing resumes at the point where it left off in the original file. The included file may contain any valid directive for the current section, but may not contain a section name. Includes may be nested up to three levels deep (counting the original file as level 1). You must maintain include files manually — the configuration dialog modify the original *.INI* file only, and do not update included files.

**NextINIFile = File:** The full path and name of the file must be specified. All subsequent shells will read the specified *.INI* file, and ignore any **[Secondary]** section in the original *.INI* file. Under 4DOS, if you have a diskless or floppy-based workstation, NextINIFile will allow you to shift *4DOS.INI* to a network drive for secondary shells, and avoid all access to the original boot drive.

### ***Advanced Directive For 4NT and Take Command Only***

**SaveDirCase = YES | No:** Enables or disables special processing to maintain the original case of each path element when changing directories. This processing is necessary for a few programs which are case-sensitive in their use of directory names. If you do not use such a program, disabling this case preservation will speed up directory changes slightly.

## Advanced Directives For 4DOS Only

### 4DOS

The following advanced directives apply to 4DOS only, and will not work in 4NT or Take Command.

**CritFail** = Yes | NO: This is the same as /F on the SHELL= line in *CONFIG.SYS*; it forces a Fail response to all DOS critical errors. We do **not** recommend this on most systems, because you will not be able to react to a critical error and correct the problem that caused it.

**DiskReset** = Yes | NO: Enables disk resets after file processing commands, before DIR, and when starting 4DOS. Set to **Yes** if you have problems with disk change detection on floppy disk drives, or with network software which doesn't always properly flush data to the disk.

**DVCleanup** = YES | No: Controls the cleanup of 4DOS resources when you close a 4DOS window from the DESQview menu.

**FullINT2E** = YES | No: Enables support for the *COMMAND.COM* “back door” (interrupt 2E) which some programs use to execute commands.

**Inherit** = YES | No: *.INI* file data, aliases, and history lists are normally passed to secondary shells. **No** disables this feature.

**MessageServer** = YES | No: Enables the “message server” included for compatibility with *COMMAND.COM* in MS-DOS 4.x and above.

**NetWareNames** = Yes | NO. Set to **Yes** to include strings in the resident portion of 4DOS which Novell NetWare searches for when it loads. You **must** use this option on most NetWare systems to avoid problems with destroyed environment variables during LOGIN.

**OutputBIOS** = Yes | NO: Determines whether 4DOS uses the BIOS for all screen displays. If set to **No**, 4DOS will use direct screen writes for color-coded directories and commands like DRAWBOX, LIST, SELECT, and SCRPUT. If set to **Yes**, 4DOS will perform these commands using BIOS calls. When OutputBIOS is set to **Yes** you may also need to set ScreenColumns appropriately (see page 225).

**SDFlush** = Yes | NO: Determines whether 4DOS instructs Microsoft SMARTDRV (and other compatible disk caching programs) to flush any “write-behind” buffers to the disk before the 4DOS prompt is displayed. Setting SDFlush to Yes will ensure that SMARTDRV is instructed to write all modified data to disk before the prompt is displayed, but cannot guarantee that SMARTDRV actually performs this task.

**StackSize** = nnnn: Set the 4DOS internal stack size. The allowable range of values is 8192 to 16384. The default is 8192 bytes on standard DOS systems, and 10240 bytes under Windows 95 / 98 / ME. If you use complex combinations of “prefix” commands like DO, EXCEPT, FOR, GLOBAL, IF, IFF, and SELECT on the same line, and especially if you use these commands in multiple nested batch files or GOSUBs, you may encounter a “4DOS internal stack overflow” error. If you do, use this directive to increase the stack space available.

**SwapReopen** = Yes | NO: Set to **Yes** to enable reopening of the 4DOS swap file if it is closed by another program. This is required when swapping 4DOS to Novell NetWare drives or when using other applications which close 4DOS's swap file. In all other circumstances, it is only useful for diagnostic purposes. Setting SwapReopen to **Yes** also disables the reduced swapping size used in 4DOS secondary shells.

**UniqueSwapName** = Yes | No: Set to **Yes** to change the disk swap file name from *4DOSSWAP.nnn* to a unique, random name generated by 4DOS, with an extension of “4SW” (e.g., *A1CD6B11.4SW*).

**Win95LFN** = Yes | No: Under Windows 95, 98, or ME the default is **Yes**. Set Win95LFN to **No** to disable long filename support in these environments. This directive affects a wide range of internal filename and file handling features, but does not prevent 4DOS from detecting that it is running under Windows 95/98/ME. It generally should be used only for debugging or diagnostic purposes.

In other environments the default is **No**. You can set Win95LFN to **Yes** to force 4DOS to try to use long filenames in non-standard environments (e.g., where a driver or memory-resident program provides long filename support that is not built into the operating system). **Setting Win95LFN to Yes does not guarantee that non-standard LFN support will work with 4DOS**, so you should do some testing before using this method extensively or for critical data or procedures.

### Examples

The following examples will give you an idea of the types of things that can be done with the *.INI* file. The comments on each directive explain what it does.

First, a very simple 4DOS example that just sets up swapping and environment size, leaving everything else at its default value:



```
[4DOS]
InstallPath = c:\4dos700
Swapping = ems, c:\           ;try EMS, then C: root
Environment = 1024            ;set environment size
```

Here's another 4DOS example, for a system which supports Upper Memory blocks (UMBs). Several settings take advantage of UMBs, and others modify the 4DOS configuration to match the user's preferences. Note that the comment for the Swapping directive is on separate lines before the directive itself, as no comments are allowed in string directives:

```
[4DOS]
InstallPath = c:\4dos700
Environment = 3072           ;expand environment to 3K
Alias = 6144                 ;expand aliases to 6K
LocalHistory = No           ;use a global history
    ;for swapping try XMS, then RAM disk H:, then C:
Swapping = xms, h:\, c:\
UMBLoad = Yes               ;res. part of 4DOS in UMB
UMBEnvironment = Yes       ;master environment in UMB
UMBHistory = Yes           ;global history in UMB
BatchEcho = No             ;default is ECHO OFF
CursorOver = 100           ;overstrike cursor 100%
CursorIns = 10             ;insert cursor 10%
```

This example for 4NT configures certain special characters to match 4DOS, and changes other default settings to suit the user's preferences

```
[4NT]
PauseOnError = No          ;don't stop on INI errors
CommandSep = ^             ;4DOS command separator
EscapeChar = ↑            ;4DOS escape character
ParameterChar = &         ;4DOS parameter character
BatchEcho = No            ;default to ECHO OFF
History = 2048             ;expand history to 2K bytes
BeepFreq = 880            ;make beep higher pitch
EditMode = Insert         ;insert mode for cmd edit
ListFind = F5             ;F5 does a find in LIST
ListNext = F6             ;and F6 does a find next
StdColors = bri cya on blu ;default colors
colordir = DIRS:bri yel;com exe bat btm cmd:bri whi
```



## CHAPTER 7 / COMMAND REFERENCE

The following pages are a complete guide and reference to the 4DOS, 4NT, and Take Command commands that are available from the command line, in aliases, and in batch files. Most of these commands are **internal**, which means that the command processor performs the activity you have requested without running another program. (See page 18 for more information on internal and external commands.)

We offer over 120 internal commands, many more than any version of *COMMAND.COM* or *CMD.EXE*. These neither replace nor interfere with external commands like BACKUP, DISKCOPY, SCANDISK, or XCOPY. You can continue to use those utilities like you always have. Also, each of our command processors has been designed to be compatible with virtually all traditional internal commands, and to enhance most of those commands with additional options and capabilities. Once you have installed your new command processor, you can continue using the commands that you already know and get the same results.

We have made no attempt to document external DOS, Windows 95 / 98 / ME, or Windows NT / 2000 / XP commands in this reference, partly because they are explained in your operating system documentation, and partly because external commands — and the options available with each command — vary widely from one operating system to another and from one version to another. The 4DOS help system can be configured to start the DOS help program, if it is available.

Most of these commands are either enhanced traditional commands or are entirely new (a few are the same as traditional commands). If you are comfortable using traditional commands, you can switch to your new command processor without making any changes in your habits. But you will be missing a lot of the power of these enhancements and new commands unless you take a few minutes to see what's available here. Make sure you don't skip a section of this reference just because you already know how to use a traditional command with the same name.

If you come across terms or concepts in this chapter that you are unsure about, please refer to Chapter 1 / General Concepts, the Index, or the Glossary which is part of your online help.

### ***Command Categories***

The best way to learn commands is to use them and experiment with them. We urge you to browse through this chapter occasionally and look for commands that might help simplify your computing life. The following lists

categorize the available commands by topic and will help you find the ones that you need. Commands listed below with an asterisk [\*] are not available in all command processors; see the list at the end for details.

**System configuration:**

BREAK	CHCP *	CLS	COLOR
CTTY *	DATE	DIRHISTORY	FREE
HISTORY	KEYS *	KEYBD	LH/LOADHIGH*
LOG	MEMORY	OPTION	PROMPT
REBOOT	SETDOS	SWAPPING *	TIME
VER	VERIFY	VOL	

**File and directory management:**

ATTRIB	COPY	DEL/ERASE	DESCRIBE
FFIND	HEAD	LIST	MOVE
RECYCLE *	REN/RENAME	SELECT	TAIL
TOUCH	TREE	TRUENAME	TYPE

**Subdirectory management:**

CD/CHDIR	CDD	DIR	DIRS
MD/MKDIR	MKLNK *	POPD	PUSHD
RD/RMDIR			

**Input and output:**

DRAWBOX	DRAWHLIN	DRAWVLIN	ECHO
ECHOERR	ECHOS	ECHOSERR	INKEY
INPUT	KEYSTACK	MSGBOX*	QUERYBOX*
SCREEN	SCRPUT	TEXT	VSCRPUT

**Batch files and aliases** (these commands are primarily for batch files and aliases — some may also be useful at the prompt, others work only in batch files; see the individual commands for details):

ALIAS	BEEP	CALL	CANCEL
DELAY	DO	ENDLOCAL	FOR
FUNCTION	GLOBAL	GOSUB	GOTO
IF	IFF	IFTP *	LOADBTM
ON	PAUSE	QUIT	REM
RETURN	SETLOCAL	SHIFT	SNPP *
SWITCH	TEXT	UNALIAS	UNFUNCTION

**Environment and path:**

ESET	PATH	SET	UNSET
------	------	-----	-------

**Window management:**

ACTIVATE \*      TITLE \*      WINDOW \*

**Other commands:**

?                      DETACH \*              EXCEPT              EXIT  
HELP                      SHRALIAS\*              START              TEE  
TIMER                      Y

\* Some of these commands are not included in one or more of our command processors. Usually a command is left out of a product when it is not useful or cannot be implemented in that environment (for example, MSGBOX displays a pop-up Windows message box, and is not available under 4DOS because DOS applications cannot access Windows features like message boxes). The commands that are specific to particular products are:

ACTIVATE	4NT, TC	PLAYSOUND	4NT, TC
ASSOC	4NT, TC	PRINT	4NT, TC
CHCP	4DOS, 4NT	QUERYBOX	4NT, TC
CTTY	4DOS	RECYCLE	4NT, TC
DDEEXEC	TC	SENDMAIL	4NT, TC
DETACH	4NT, TC	SHORTCUT	4NT, TC
EVENTLOG	4NT, TC	SNPP	4NT, TC
FTYPE	4NT, TC	SHRALIAS	4NT, TC
IFTP	4NT, TC	START	4NT, TC
KEYS	4NT, TC	SWAPPING	4DOS
LFNFOR	4DOS	TASKEND	4NT, TC
LH / LOADHIGH	4DOS	TASKLIST	4NT, TC
LOCK	4DOS	TCTOOLBAR	TC
MKLNK	4NT, TC	TITLE	4NT, TC
MSGBOX	4NT, TC	UNLOCK	4DOS
PLAYAVI	4NT, TC	WINDOW	4NT, TC

## ***How to Use the Command Descriptions***

Each of the internal commands is described in detail on the following pages. The descriptions are arranged alphabetically, and each includes examples that will help you learn to use the commands.

Each description begins with the **name** of the command on the left side of the page. If the command is only available in some of our products, those products are listed next to the command name and in the page header. Commands marked “New” on the right side of the page are unique to 4DOS,

4NT, and Take Command. Those marked “Enhanced” are similar to traditional commands but add new features and options. The commands marked “Compatible” follow the syntax and features of the traditional command with the same name.

The name is followed by a sentence or two that briefly describes the command's **purpose** or major function. That sentence should help you determine quickly whether you have found the command you are seeking.

The next part of each description shows the command's **format** or syntax. The format line uses certain conventions to describe how the command should be entered and to create reference points for the text describing the command:

- ✓ Words in **UPPER CASE** must be spelled exactly as they are shown (although you can type them in either upper or lower case, or a combination). If a word is shown partly in upper case (for example **BRight**), only the upper case portion is required, the rest is optional.
- ✓ Words shown in *italics* (for example *source* or *filename*) are meant to be replaced by other words or values. Each of these words is explained directly beneath the format line and discussed in more detail in the text description of the command. When the word stands for a file name, you can use a simple name like *MYFILE.TXT*, or include a drive letter and / or a full path, like *C:\MYDIR\MYFILE.TXT*.
- ✓ Items followed by an ellipsis (three periods [...]) may be repeated. For example, *filename ...* means you may enter one or more file names at this point in the command.
- ✓ Text shown in **[square brackets]** is optional. Text outside of square brackets must be entered literally (if it is capitalized) or replaced by other words or values (if it is in italics). For example, in this hypothetical command:  
  

```
DOIT [/A /W] filename [NOW]
```
- ✓ the switches /A and /W and the keyword NOW are optional. The *filename* should be replaced by the appropriate name for the operation you want to perform. The switches and the keyword NOW at the end must be entered as shown, if they are used. For example, you could use “NOW”, “Now”, or “now”, but you could not abbreviate NOW to NO.

- ✓ Vertical bars [ | ] represent a choice; you can pick one option or another but not both. For example, the following format shows that the command may be followed by the word ON or the word OFF, but not both:

```
COMMAND [ ON | OFF ]
```

- ✓ A slash followed by a letter, like [/X], is an “option” or “switch” which controls the effect of a command. Many commands have several switches, and you are usually free to use none, one, or several to make a command behave as you wish. If you use a single switch, you must precede it with a slash. If you use several switches, in most cases you can put them together with one slash or use separate slashes. For example, if you wanted to use switches X, Y, and Z for a command, you could type them three different ways:

```
command /x /y /z  
command /x/y/z  
command /xyz
```

- ✓ A few switches, particularly in the DIR, SELECT, and START commands, use two or more characters. If you need to follow a multi-letter switch with another switch, the second switch must have its own slash to avoid ambiguity. For example, you could use DIR /oa /d to force an alphanumeric sort (/oa) and suppress directory colors (/d). However, if you try to put all the switches together (DIR /oad) DIR will not do what you want because the “d” will be interpreted as part of the /o switch, where it would mean sort by date and time. The second slash eliminates this ambiguity.

Included in the format section is an explanation of each replaceable argument and a one or two word explanation of each switch. Many descriptions also list related commands to help you find the exact command you want.

For file handling commands, a section called **File Selection** appears immediately after the format section. This section lists the file-handling features that the command supports. The list may include mention of extended wildcards (see page 94); multiple file names (page 103); include lists (page 104); and date, time, and size ranges (see page 97) and file exclusion ranges (see page 102).

Next, you'll find a description of the command's **usage**. This description normally starts with the basic functions of a command and gradually adds more details. We've also included many examples to help you see the

command in action. In the examples, characters in **bold** type represent input from the user. Characters in normal type represent prompts or responses from the command processor or lines in a batch file.

The last part of each description is a detailed explanation of the **options** or switches available for each command, in alphabetical order. Occasionally, we've included more examples in this section to demonstrate how a switch is used or how multiple switches interact.

Most of the commands and their switches have the same use in 4DOS, 4NT, and Take Command. When a command, feature, or switch applies to a single product, we mention it specifically in the text. When an entire paragraph applies to a specific product, we use marginal text to identify that product. When an entire command is only available in one product, the product is listed next to the command name, and in the page headings.

- 4DOS** marks information that applies only to 4DOS.
- 4NT** marks information that applies only to 4NT.
- TC** marks information that applies only to Take Command.

An exclamation point [!] to the left of a paragraph means that paragraph contains a caution or warning you should observe when using the feature it discusses.

In the Usage and Options sections you may see the symbol ❖ This indicates a more in-depth discussion or an advanced topic. You can skip this section if you are new to the command, but you may want to come back to it later for more details, or if you're having trouble with the command. In most cases the remainder of the section after such a symbol is devoted to similar information. The ❖ doesn't mean that only advanced users will need the information – you may find it useful even if you're relatively new to computers or to our products. But it does mean that you can skip the marked section and still understand and use the basic features of the command. If a ❖ appears before the “Usage” heading, it indicates that the entire command is generally used only in unusual situations or by more advanced users.

When you see a ❖ in the list of options, remember that the options are listed alphabetically, so there may be more basic options discussed later in the list, after a more complex or advanced option marked with ❖ Don't stop reading the option list the first time you see the mark.



?

(New)

**Purpose:** Display a list of internal commands, or prompt for a command.

**Format:** ? ["prompt" command]

***prompt:*** Prompt text about whether to execute the ***command***.

***command:*** Command to be executed if user answers Y.

**Usage:** ? has two functions.

When you use the ? command by itself, it displays a list of internal commands. The ? display looks like this (the partial list below is for 4DOS; you will see a slightly different list for 4NT or Take Command:

```
c:\> ?
? ALIAS          ATTRIB          BEEP
BREAK           CALL            CANCEL          CASE
.....
UNSET           VER             VERIFY          VOL
VSCRPUT         Y
```

If you have disabled a command with SETDOS /I, it will not appear in the list.

The second function of ? is to prompt the user before executing a specific line in a batch file. If you add a ***prompt*** and a command, ? will display the prompt followed by “(Y/N)?” and wait for the user’s response. If the user presses “Y” or “y”, the line will be executed. If the user presses “N” or “n”, the line will be ignored.

For example, the following command might be used in a batch file:

```
? "Load the network" call netstart.btm
```

When this command is executed, you will see the following prompt; if you answer “Y”, the CALL command will be executed:

```
Load the network (Y/N)?
```

**ACTIVATE** [4NT, TC]

(New)

**Purpose:** Activate a window, set its state, or change its title.

**Format:** ACTIVATE "*window*" [MAX | MIN | RESTORE | CLOSE |  
POS=left,top,width,height | TOPMOST | NOTOPMOST | TOP |  
BOTTOM | HIDE | "*title*"]

***window*:** Current title of window to work with.

***pos*:** Move or resize window.

***topmost*:** Set the topmost attribute.

***notopmost*:** Remove the topmost attribute.

***top*:** Move to the top of the Z-order.

***bottom*:** Move to the bottom of the Z-order.

***hide*:** Hide the window.

***title*:** New title for window.

See also: START, TITLE, and WINDOW.

**Usage:** Both the current title and the new title, if any, must be enclosed in double quotes. The quotes will not appear as part of the title text.

If no options are used, the window named in the command will become the active window (the window with a highlighted title bar), and will be able to receive input. The window will also be activated with some options, as noted below.

The options are:

**MAX** expands the window to its maximum size and activates it.

**MIN** reduces the window to an icon and activates it.

**RESTORE** returns the window to its default size and location and activates it.

**CLOSE** sends a "close" message to close the window, and end the process running in the window.

**POS** sets the window position and size (in pixels) and activates the window.

**TOPMOST** keeps the window on top of all other windows until it closes, or NOTOPMOST is used.

**NOTOPMOST** allows other windows to overlay the window (this is the normal state for most windows).

**TOP** moves the window to the top of the window order, above all other non-TOPMOST windows.

**BOTTOM** moves the window to the bottom of the window order.

**HIDE** makes the window invisible and activates it (to make the window visible again, use RESTORE).

**“title”** changes the window title.

You can not specify more than one option at a time. For example, if you change the title you cannot also maximize the window. To perform multiple operations, use multiple ACTIVATE commands.

This example maximizes and then renames the window called “Take Command”:

```
[c:\] activate "Take Command" max  
[c:\] activate "Take Command" "TCMD"
```

You can use wildcards (see page 94) in the **window** name if you only know the first part of the title. This is useful with applications that change their window title to reflect the file currently in use.

ACTIVATE is often used before KEYSTACK (page 392) to make sure the proper window receives the keystrokes. ACTIVATE works by sending messages to the named **window**. If the window ignores or misinterprets the messages, ACTIVATE may not have the effect you want.

If ACTIVATE is used in a batch file under Windows 2000 or XP, and the batch file is not itself running in the active window (the window with its title bar highlighted), then ACTIVATE may not activate the desired window. This is because under Windows 2000 / XP you cannot make another window active except when the window which issues the command is itself active already. This is an operating system feature which helps to prevent windows which are not in the foreground from “grabbing” input destined for other windows.

**ALIAS**

(New)

**Purpose:** Create new command names that execute one or more commands or redefine default options for existing commands; assign commands to keystrokes; load or display the list of defined alias names.

**Format:** ALIAS [/P /R *file...*] [*name*[=][*value*]]

***file:*** One or more files to read for alias definitions.

***name:*** Name for an alias, or for the key to execute the alias.

***value:*** Text to be substituted for the alias name.

**/P**(ause)

**/R**(ead file)

See also: UNALIAS, and Aliases on page 122.

**Usage:** The ALIAS command lets you create new command names or redefine internal commands. It also lets you assign one or more commands to a single keystroke. An alias is often used to execute a complex series of commands with a few keystrokes or to create “in memory batch files” that run much faster than disk-based batch files.

For example, to create a single-letter command **D** to display a wide directory, instead of using the longer **DIR /W**, you could use the command:

```
[c:\] alias d = dir /w
```

Now when you type a single **d** as a command, it will be translated into a **DIR /W** command.

**TC** *You can also define or modify aliases with the Alias dialog (see page 55). The dialog allows you to enter the alias name and value into separate fields in a dialog box, rather than using the ALIAS command. All of the information in this section also applies to aliases defined via the dialog, unless otherwise noted.*

If you define aliases for commonly used application programs, you can often remove the directories they're stored in from the PATH. For example, if you use Microsoft Word and had the **C:\WINWORD** directory in your path, you could define the following alias:

```
[c:\] alias ww = c:\winword\winword.exe
```

With this alias defined, you can probably remove *C:\WINWORD* from your path. Word will now load more quickly than it would if the command processor had to search the PATH for it. In addition, the PATH can be shorter, which will speed up searches for other programs.

If you apply this technique for each application program, you can often reduce your PATH to just two or three directories containing utility programs, and significantly reduce the time it takes to load most software on your system. Before removing a directory from the PATH, you will need to define aliases for all the executable programs you commonly use which are stored in that directory.

Aliases are stored in memory, and are not saved automatically when you turn off your computer or end your current command processor session. See page 256 for information on saving and reloading your aliases.

### ***Multiple Commands and Special Characters in Aliases***

An alias can represent more than one command. For example:

```
[c:\] alias letters = `cd \letters & tedit`
```

creates a new command called LETTERS. The command first uses CD to change to a subdirectory called *\LETTERS* and then runs a program called *TEDIT*. The ampersand [&] is the 4NT and Take Command command separator and indicates that the two commands are distinct and should be executed sequentially. (The 4DOS command separator is normally a caret [^].)

Aliases make extensive use of the command separator (see page 76), and the parameter character (see page 224), and may also use the escape character (see page 119). These characters differ between 4DOS and 4NT / Take Command. In the text and examples below, we use the 4NT and Take Command characters. The difference is also explained the first time each character is used. Be sure to insert the correct characters for the command processor you are using. See page 195 for more details on the different characters used under different command processors.

When an alias contains multiple commands, the commands are executed one after the other. However, if any of the commands run an external Windows application, you must be sure the alias will wait for the application to finish before continuing with the other commands.

This behavior is controlled by the **Wait for completion** setting on the “Startup” tab of the configuration dialog (see page 50) or the ExecWait directive in the *.INI* file (see page 221).

When you type alias commands at the command line or in a batch file, you **must** use back-quotes [ ``` ] around the definition if it contains multiple commands, parameters (discussed below), environment variables, redirection, or piping. The back-quotes prevent premature expansion of these arguments. You **may** use back-quotes around other definitions, but they are not required. (You do not need back-quotes when your aliases are loaded from an ALIAS /R file; see below for details. You also do not need back quotes when entering an alias in the Aliases dialog box.) The examples in this section include back-quotes only when they are required.

### ***Nested Aliases***

Aliases may invoke internal commands, external commands, or other aliases. (However, an alias may not invoke itself, except in special cases where an IF or IFF command is used to prevent an infinite loop.) The two aliases below demonstrate alias nesting (one alias invoking another). The first line defines an alias which runs Word from the *E:\WINWORD\* subdirectory. The second alias changes directories with the PUSH command, runs the WP alias, and then returns to the original directory with the POP command:

```
[c:\] alias wp = e:\winword\winword.exe  
[c:\] alias w = `pushd c:\wp & wp & popd`
```

The second alias above could have included the full path and name of the *WINWORD.EXE* program instead of calling the WP alias. However, writing two aliases makes the second one easier to read and understand, and makes the first alias available for independent use. If you rename the *WINWORD.EXE* program or move it to a new directory, only the first alias needs to be changed.

### ***Temporarily Disabling Aliases***

If you put an asterisk [\*] immediately before a command in the **value** of an alias definition (the part after the equal sign), it tells the command processor not to attempt to interpret that command as another (nested) alias. An asterisk used this way must be preceded by

a space or the command separator and followed immediately by an internal or external command name.

By using an asterisk, you can redefine the default options for any internal or external command. For example, suppose that you always want to use the DIR command with the /2 (two column) and /P (pause at the end of each page) options. The following line will do just that:

```
[c:\] alias dir = *dir /2/p
```

If you didn't include the asterisk, the second DIR on the line would be the name of the alias itself, and the command processor would repeatedly re-invoke the DIR alias, rather than running the DIR command. This would cause an "Alias loop" or "Command line too long" error. The asterisk forces interpretation of the second DIR as a command, not an alias.

An asterisk also helps you keep the names of internal commands from conflicting with the names of external programs. For example, suppose you have a program called *DESCRIBE.EXE*. Normally, the internal DESCRIBE command will run anytime you type DESCRIBE. But two simple aliases will give you access to both the *DESCRIBE.EXE* program and the DESCRIBE command:

```
[c:\] alias describe = c:\winutil\describe.exe
[c:\] alias filedesc = *describe
```

The first line above defines DESCRIBE as an alias for the *DESCRIBE.EXE* program. If you stopped there, the external program would run every time you typed DESCRIBE and you would not have easy access to the internal DESCRIBE command. The second line renames the internal DESCRIBE command as FILEDESC. The asterisk is needed in the second command to indicate that the following word means the internal command DESCRIBE, not the DESCRIBE alias which runs your external program.

Another way to understand the asterisk is to remember that a command is always checked for an alias first, then for an internal or external command, or a batch file (see page 197). The asterisk at the beginning of a command name simply skips over the usual check for aliases when processing that command, and allows the command processor to go straight to checking for an internal command, external command, or batch file.

You can also use an asterisk before a command that you enter at the command line or in a batch file. If you do, that command won't be interpreted as an alias. This can be useful when you want to be sure you are running the true, original command and not an alias with the same name, or temporarily defeat the purpose of an alias which changes the meaning or behavior of a command. For example, above we defined an alias for DIR which made directories display in 2-column paged mode by default. If you wanted to see a directory display in the normal single-column, non-paged mode, you could enter the command \*DIR and the alias would be ignored during that one command.

You can also disable aliases temporarily with the SETDOS /X command (see page 477).

### ***Partial Alias Names***

You can also use an asterisk in the ***name*** of an alias. When you do, the characters following the asterisk are optional when you invoke the alias command. (Use of an asterisk in the alias ***name*** is unrelated to the use of an asterisk in the alias ***value*** discussed above.) For example, with this alias:

```
[c:\] alias wher*eis = dir /sp
```

the new command, WHEREIS, can be invoked as WHER, WHERE, WHEREI, or WHEREIS. Now if you type:

```
[c:\] where myfile.txt
```

The WHEREIS alias will be expanded to the command:

```
dir /sp myfile.txt
```

### ***Keystroke Aliases***

If you want to assign an alias to a keystroke, use the key name on the left side of the equal sign, preceded by an at sign [@]. For example, to assign the command DIR /W to the F4 key, type

```
[c:\] alias @F4 = dir /w
```

See page 34 for a complete listing of key names and a description of the key name format.



**TC** *Windows always interprets Alt plus a key as a menu accelerator key. Such keystrokes are not passed to Take Command, and therefore cannot be used for keystroke aliases.*

If you define a keystroke alias with a single at sign as shown above, then, when you press the **F4** key, the value of the alias (DIR /W above) will be placed on the command line for you. You can type additional parameters if you wish and then press **Enter** to execute the command. With this particular alias, you can define the files that you want to display after pressing **F4** and before pressing **Enter** to execute the command.

If you want the keystroke alias to take action automatically without waiting for you to edit the command line or press **Enter**, you can begin the definition with two at signs [@@]. The command processor will execute the alias “silently,” without displaying its text on the command line. For example, this command will assign an alias to the **F11** key that uses the CDD command to take you back to the previous default directory:

```
[c:\] alias @@f11 = cdd -
```

When you define keystroke aliases, the assignments will only be in effect at the command line, not inside application programs. Be careful not to assign aliases to keys that are already used at the command line (like **F1** for Help). The command-line meanings take precedence and the keystroke alias will never be invoked. If you want to use one of the command-line keys for an alias instead of its normal meaning, you must first disable its regular use with the NormalKey or NormalEditKey directives in your .INI file. See page 230 for instructions.

- ❖ You can also define a keystroke alias by using “@” or “@@” plus a scan code for one of the permissible keys (see the Key Codes table in the online help for a list of scan codes). In most cases, it will be easier to use key names. Scan codes should only be used with unusual keyboards where a key name is not available for the key you are using.

### ***Displaying Aliases***

If you want to see a list of all current ALIAS commands, type:

```
[c:\] alias
```

You can also view the definition of a single alias. For example, if you want to see the definition of the alias LIST, you can type:

```
[c:\] alias list
```

### ***Saving and Reloading Your Aliases***

You can save your aliases to a file called *ALIAS.LST* this way:

```
[c:\] alias > alias.lst
```

You can then reload all the alias definitions in the file the next time you boot up with the command:

```
[c:\] alias /r alias.lst
```

This is much faster than defining each alias individually in a batch file. If you keep your alias definitions in a separate file which you load when your system starts, you can edit them with a text editor, reload the edited file with ALIAS /R, and know that the same alias list will be loaded the next time you boot your computer.

When you define aliases in a file that will be read with the ALIAS /R command, you do not need back-quotes around the value, even if back-quotes would normally be required when defining the same alias at the command line or in a batch file.

You can also save and reload your aliases using the Aliases dialog (see page 55). The Export button in the dialog box is equivalent to the ALIAS > filename command shown above, and the Import button is equivalent to the ALIAS /R command.

To remove an alias, use the UNALIAS command, or the Delete button in the Aliases dialog.

### ***Alias Parameters***

Aliases can use command-line arguments or parameters like those in batch files. The command-line arguments are numbered from %0 to %255. %0 contains the alias name. It is up to the alias to determine the meaning of the other parameters. You can use quotation marks to pass spaces, tabs, commas, and other special characters in an alias parameter; see page 199 for details.

Parameters that are referred to in an alias, but which are missing on the command line, appear as empty strings inside the alias. For example, if you put two parameters on the command line, any reference in the alias to %3 or any higher-numbered parameter will be interpreted as an empty string.

The parameter %n\$ has a special meaning. 4NT and Take Command interpret it to mean “the entire command line, from argument n to the end.” If n is not specified, it has a default value of 1, so %\$ means “the entire command line after the alias name.” 4DOS normally uses an ampersand [&] instead of a dollar sign [\$] to indicate the remainder of the command tail (for example, use %& to refer to all parameters, or %2& to refer to the second parameter and all those after it). The special parameter %# contains the number of command-line arguments.

For example, the following alias will change directories, perform a command, and return to the original directory:

```
[c:\] alias in `pushd %1 & %2$ & popd`
```

When this alias is invoked as:

```
[c:\] in c:\comm mycomm /zmodem /56K
```

the first parameter, %1, has the value *c:\comm*. %2 is *mycomm*, %3 is */zmodem*, and %4 is */56K*. The command line expands into these three separate commands:

```
pushd c:\comm
mycomm /zmodem /56K
popd
```

- ❖ This next example uses the IFF command to redefine the defaults for SET. It should be entered on one line:

```
[c:\] alias set = `iff %# == 0 then & *set /p
& else & *set %$ & endiff`
```

This modifies the SET command so that if SET is entered with no arguments, it is replaced by SET /P (pause after displaying each page), but if SET is followed by an argument, it behaves normally. Note the use of asterisks (\*set) to prevent alias loops.

- ❖ If an alias uses parameters, command-line arguments will be deleted up to and including the highest referenced argument. For example, if

an alias refers only to %1 and %4, then the first and fourth arguments will be used, the second and third arguments will be discarded, and any additional arguments beyond the fourth will be appended to the expanded command (after the value portion of the alias). For example, the second and third parameters will be discarded by this alias:

```
[c:\] alias show=`echo %1 %4`  
[c:\] show 1 2 3 4 5 6 7  
1 4 5 6 7
```

If an alias uses no parameters, all of the command-line arguments will be appended to the expanded command. For example:

```
[c:\] alias show=echo  
[c:\] show 1 2 3 4 5 6 7  
1 2 3 4 5 6 7
```

- ❖ Aliases also have full access to all variables in the environment, internal variables, and variable functions. For example, you can create a simple command-line calculator this way:

```
[c:\] alias calc=`echo The answer is: %@eval[%&]`
```

To use the calculator, simply enter the alias name and the expression to be calculated, for example:

```
[c:\] calc 5 * 6  
The answer is: 30
```

### ❖ *Expanding Aliases at the Prompt*

You can expand an alias on the command line and view or edit the results by pressing **Ctrl-F** after typing the alias name, but before the command is executed. This replaces the alias with its contents, and substitutes values for each alias parameter, just as if you had pressed the **Enter** key. However, the command is not executed; it is simply redisplayed on the command line for additional editing.

**Ctrl-F** is especially useful when you are developing and debugging a complex alias, or if you want to make sure that an alias that you may have forgotten won't change the effect of your command.

## ❖ **Local and Global Aliases**

The aliases can be stored in either a “local” or “global” list. You can control the type of alias list on the “Startup” tab on the configuration dialog or with the LocalAliases directive in the *.INI* file (see page 213), with the /L and /LA options of the START command (see page 488), and (in 4NT and Take Command) with the /L and /LA startup options (see your *Introduction and Installation Guide*).

With a local alias list, any changes made to the aliases will only affect the current copy of the command processor. They will not be visible in other shells or sessions. This is the default under 4DOS.

With a global alias list, all copies of the command processor will share the same alias list, and any changes made in one copy will affect all other copies. This is the default for 4NT and Take Command.

There is no fixed rule for determining whether to use a local or global alias list. Depending on your work style, you may find it most convenient to use one type, or a mixture of types in different sessions or shells. We recommend that you start with the default approach for your command processor, then modify it if you find a situation where the default is not convenient.

Whenever you start a secondary shell (see page 9) which uses a local alias list, it inherits a copy of the aliases from the previous shell. However, any changes to the aliases made in the secondary shell will affect only that shell. If you want changes made in a secondary shell to affect the previous shell, use a global alias list in both shells.

### **4NT, TC    Retaining Global Aliases with SHRALIAS**

If you select a global alias list for 4NT or Take Command, you can share the aliases among all copies of the command processor running in any session. When you close all 4NT or Take Command sessions, the memory for the global alias list is released, and a new, empty alias list is created the next time you start 4NT or Take Command.

If you want the alias list to be retained in memory even when no command processor session is running, use the SHRALIAS command (see page 485), which loads a program to perform this service for global aliases, user-defined functions, command history (page 65), and

directory history (page 75). You may find it convenient to execute SHRALIAS from your *4START* / *TCSTART* file (see page 132).

❖ **The UNKNOWN\_CMD Alias**

If you create an alias with the name **UNKNOWN\_CMD**, it will be executed any time the command processor would normally issue an “Unknown command” error message. This allows you to define your own “handler” for unknown commands. When the **UNKNOWN\_CMD** alias is executed, the command line which generated the error is passed to the alias for possible processing. For example, to display the command that caused the error:

```
alias unknown_cmd `echo Error in command "%&"`
```

- ! If the **UNKNOWN\_CMD** alias contains an unknown command, it will call itself repeatedly. If this occurs the command processor will loop up to 10 times, then display an “**UNKNOWN\_CMD** loop” error.

Options: **/P(ause)**: This option is only effective when **ALIAS** is used to display existing definitions. It pauses the display after each page and waits for a keystroke before continuing (see page 80).

**/R(ead file)**: This option loads an alias list from a file. The format of the file is the same as that of the **ALIAS** display:

```
name=value
```

where **name** is the name of the alias and **value** is its value. You can use an equal sign [=] or space to separate the name and value. Back-quotes are not required around the value. You can add comments to the file by starting each comment line with a colon [:]. You can load multiple files with one **ALIAS /R** command by placing the names on the command line, separated by spaces:

```
[c:\] alias /r alias1.lst alias2.lst
```

Each definition in an **ALIAS /R** file can be up to 511 characters long in 4DOS, or 4095 characters in 4NT and Take Command. The definitions can span multiple lines in the file if each line or a definition, except the last, is terminated with an escape character (see page 119).

**ALIAS /R** will read from **STDIN** if no filename is specified and input is redirected.

## ASSOC [4NT, TC] (Enhanced)

**Purpose:** Modify or display relationships between file extensions and file types stored in the Windows registry.

**Format:** ASSOC [/P /R *file...*] [*.ext*]=[*filetype*]]

***file*:** One or more files containing entries to be added to the registry.

***.ext*:** The file extension whose file type you want to display or set.

***filetype*:** A file type stored in the Windows registry.

**/P(ause)**

**/R(ead)**

See also: FTYPE, and Executable Extensions on page 106.

**Usage:** ASSOC allows you to create, modify, or display associations between file extensions and file types stored in the Windows registry.

ASSOC manages Windows file associations stored under the registry handle HKEY\_CLASSES\_ROOT, and discussed on page 112. If you are not familiar with file associations be sure to read about them before using ASSOC.

If you invoke ASSOC with no parameters, it will display the current associations. If you include a ***.ext***, with no equal sign or ***filetype***, ASSOC will display the current association for that extension.

If you include the equal sign and ***filetype***, ASSOC will create or update the association for extension ***.ext*** to refer to the specified file type. The valid filetypes depend on the contents of your Windows registry. See the FTYPE command or your Windows documentation for additional details.

ASSOC cannot delete an extension from the registry. However, you can create a similar effect by associating the extension with an empty file type using ASSOC ***.ext=***, without a ***filetype*** parameter.

- ! ASSOC should be used with caution, and only after backing up the registry. Improper changes to file associations can prevent applications and / or the operating system from working properly.

**Options:** **/P(ause)**: Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail on page 80.

**/R**(ead file): This option loads a list of associations from a file. The format of the file is the same as that of the ASSOC display:

.ext=filetype

ASSOC /R will read from STDIN if no filename is specified and input is redirected.



## **ATTRIB**

(Enhanced)

**Purpose:** Change or view file and subdirectory attributes.

**Format:** ATTRIB [/A:[[-+]rhsad] /D /E /I"text" /P /Q /S] [+ | -[ADHNORST]]  
[@file] file ...

**files:** A file, directory, or list of files or directories on which to operate.

**@file:** A text file containing the names of the files on which to operate, one per line (see page 109 for details).

**/A:** (Attribute select)

**/P**(ause)

**/D**(irectories)

**/Q**(uiet)

**/E** (No error messages)

**/S**(ubdirectories)

**/I"text"** (match description)

Attribute flags:

**+A** Set the archive attribute

**-A** Clear the archive attribute

**+D** Modify directory attributes (same as **/D**)

**-D** Modify directory attributes (same as **/D**)

**+H** Set the hidden attribute

**-H** Clear the hidden attribute

**+R** Set the read-only attribute

**-R** Clear the read-only attribute

**+S** Set the system attribute

**-S** Clear the system attribute

**4NT, TC** NTFS attribute flags:

**+N** Set the normal attribute

**-N** Clear the normal attribute

**+O** Set the offline attribute

**+O** Clear the offline attribute

**+T** Set the temporary attribute

**-T** Clear the temporary attribute

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104). Use extended wildcards with caution on LFN volumes; see page 105 for details.

**Usage:** Every file and subdirectory has 4 attributes that can be turned on (set) or turned off (cleared): **Archive**, **Hidden**, **Read-only**, and **System**.

Windows 2000 and Windows XP support additional attributes including **Normal**, **Offline**, and **Temporary**. For details on the meaning of each attribute, see page 18.

The ATTRIB command lets you view, set, or clear attributes for any file, group of files, or subdirectory.

You can view file attributes by entering ATTRIB without specifying new attributes (*i.e.*, without the [+|-][ADHNORST]) part of the format). (You can also view file attributes with the **DIR /T** command.) The primary use of ATTRIB is to set attributes. For example, you can set the read-only and hidden attributes for the file *MEMO*:

```
[c:\] attrib +rh memo
```

Attribute options apply to the file(s) that follow the options on the ATTRIB command line. The example below shows how to set different attributes on different files with a single command. It sets the archive attribute for all *.TXT* files, then sets the system attribute and clears the archive attribute for *TEST.COM*:

```
[c:\] attrib +a *.txt +s -a test.com
```

When you use ATTRIB on an LFN drive, you must quote any file names which contain white space or special characters. See page 15 for additional details.

To change directory attributes, use the **/D** switch. If you give ATTRIB a directory name instead of a file name, and omit **/D**, it will append “\\*.\*” to the end of the name and act on all files in that directory, rather than acting on the directory itself.

- ❖ Your operating system also supports “D” (subdirectory) and “V” (volume label) attributes, and Windows 2000 and Windows XP support the “E” (encrypted), “C” (compressed), “I” (not content-indexed), “P” (sparse file) and “J” (junction / reparse point) attributes. These attributes will be displayed by ATTRIB, but cannot be altered; they are designed to be controlled only by the operating system itself.
- ❖ ATTRIB will ignore underlines in the new attribute (the [+|-][ADHNORST]) part of the command). For example, ATTRIB sees these 2 commands as identical:

```
[c:\] attrib +a filename
```

```
[c:\] attrib +__A_ filename
```

This allows you to use a string of attributes from either the @ATTRIB variable function or from ATTRIB itself (both of which use underscores to represent attributes that are not set) and send that string back to ATTRIB to set attributes for other files. For example, to clear the attributes of *FILE2* and then set its attributes to match those of *FILE1* (enter this on one line):

```
[c:\] attrib -arhs file2 & attrib +%@attrib[file1]  
file2
```

- ❖ When ATTRIB encounters a +D or -D in the attribute string it treats it as equivalent to the /D switch, and allows modification of the attributes of a directory. When combined with @ATTRIB, or with ATTRIB's output, both of which return a D to signify a directory, this feature allows you to transfer attributes from one directory to another. For example, to clear the attributes of all files and directories beginning with *ABC* and then set their attributes to match those of *FILE1* (enter this on one line):

```
[c:\] attrib -arhs abc* & attrib +%@attrib[file1] abc*
```

Note that +D and -D can **not** be used to **change** the directory attribute itself (*i.e.* to convert a file into a directory, or vice versa). They are only useful when passing the output of @ATTRIB (or of ATTRIB itself) to ATTRIB in order to transfer attributes from one directory to another.

- Options: ❖/A: (Attribute select): Select only those files that have the specified attribute(s) set. See **Attribute Switches** on page 111 for information on the attributes which can follow /A:.

This switch specifies which files to select, **not** which attributes to set. For example, to remove the archive attribute from all hidden files, you could use this command:

```
[c:\] attrib /a:h -a *.*
```

/D(irectories): If you use the /D option, ATTRIB will modify the attributes of subdirectories in addition to files (yes, you can have a hidden subdirectory):

```
[c:\] attrib /d +h c:\mydir
```

If you use a directory name instead of a file name, and omit **/D**, **ATTRIB** will append "**\\*.\***" to the end of the name and act on all files in that directory, rather than acting on the directory itself.

**/E** (No error messages): Suppress all non-fatal error messages, such as "File Not Found." Fatal error messages, such as "Drive not ready," will still be displayed. This option is most useful in batch files and aliases.

**/I"text"**: Select files by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**/P(ause)**: Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail on page 80.

**/Q(quiet)**: This option turns off **ATTRIB**'s normal screen output. It is most useful in batch files.

**/S(subdirectories)**: If you use the **/S** option, the **ATTRIB** command will be applied to all matching files in the current or named directory and all of its subdirectories. Do not use **/S** with @file lists; see page 109 for details.

**BEEP**

(New)

**Purpose:** Beep the speaker or play simple music.

**Format:** BEEP [*frequency duration* ...]

***frequency***: The beep frequency in Hertz.

***duration***: The beep length in .055 second intervals.

**Usage:** You can use BEEP in batch files to signal that an operation has been completed, or that the computer needs attention, and you can play simple music or to create different kinds of signals for the user.

You can include as many frequency and duration pairs as you wish. No sound will be generated for frequencies less than 20 Hz. The default value for ***frequency*** is 440 Hz; the default value for ***duration*** is 2.

**4NT, TC** Windows 95/98 will ignore all ***frequency*** and ***duration*** values, and will produce only the system default sound. This is due to the design of Windows, and is not a bug in 4NT or Take Command.

This batch file fragment runs a program called *DEMO*, plays a few notes and waits for you to press a key:

```
demo & beep 440 4 600 2 1040 6
pause Finished with the demo - hit a key...
```

The following table gives the ***frequency*** values for a five octave range (middle C is 262 Hz):

C	131	262	523	1046	2093
C#/Db	139	277	554	1108	2217
D	147	294	587	1174	2349
D#/Eb	156	311	622	1244	2489
E	165	330	659	1318	2637
F	175	349	698	1397	2794
F#/Gb	185	370	740	1480	2960
G	196	392	784	1568	3136
G#/Ab	208	415	831	1662	3322
A	220	440	880	1760	3520
A#/Bb	233	466	932	1864	3729
B	248	494	988	1976	3951

**BREAK**

(Compatible)

**Purpose:** Display, enable, or disable Ctrl-C and Ctrl-Break checking for external applications.

**Format:** BREAK [ON | OFF]

**Usage:** The Ctrl-C and Ctrl-Break keys are used by many programs (including the command processor) as a signal to interrupt the current operation. BREAK controls how often DOS checks to see if you've entered one of these keystrokes.

**4NT, TC** Ctrl-C and Ctrl-Break checking cannot actually be enabled or disabled under Windows. 4NT and Take Command support BREAK as a “do-nothing” command, for compatibility with *CMD.EXE*. This avoids errors in batch files which use the BREAK command. The additional discussion below applies only to 4DOS, not to 4NT or Take Command.

**4DOS** Normally, BREAK is turned off, and DOS only checks for Ctrl-C and Ctrl-Break keystrokes during DOS input or output operations involving the screen, keyboard, serial port, and printer. However, many programs don't use DOS for these operations, and it can be difficult to interrupt them. When BREAK is turned on, DOS checks for Ctrl-C and Ctrl-Break every time a program calls DOS. Since most programs use DOS to access files and perform other functions, turning BREAK on makes it much more likely that a Ctrl-C or Ctrl-Break will be noticed. If you turn BREAK on, programs will run slightly slower than normal (the difference is not usually noticeable).

Turning BREAK on or off only affects when DOS detects Ctrl-C and Ctrl-Break and notifies the program you're running. Any program can choose to ignore these signals. Also, any program can change the BREAK setting on its own.

Type BREAK plus ON or OFF to set the BREAK status, or BREAK by itself to display the current BREAK status. For example:

```
[c:\] break on
[c:\] break
BREAK is ON
```

BREAK is off by default. You can change the default by adding a BREAK=ON command to your *CONFIG.SYS* file.

**CALL**

(Compatible)

**Purpose:** Execute one batch file from within another.

**Format:** CALL *file*

***file*:** The batch file to execute.

See also CANCEL and QUIT.

**Usage:** CALL allows batch files to call other batch files (batch file nesting). The calling batch file is suspended while the called (second) batch file runs. When the second batch file finishes, the original batch file resumes execution at the next command. If you execute a batch file from inside another batch file without using CALL, the first batch file is terminated before the second one starts.

The following batch file fragment compares an input line to “wp” and calls another batch file if it matches:

```
input Enter your choice: %%option
if "%option" == "wp" call wp.bat
```

Batch files may be nested up to 16 levels deep in 4DOS and 32 levels deep in 4NT and Take Command.

The current ECHO state is inherited by a called batch file.

- ❖ The called batch file should always either return (by executing its last line, or using the QUIT command), or terminate batch file processing with CANCEL. Do not restart or CALL the original batch file from within the called file as this may cause an infinite loop or a stack overflow.
- ❖ CALL returns an exit code which matches the batch file return code. You can test this exit code with the %\_? or %? environment variable (see page 155), and use it with conditional commands (see page 116).

# **CANCEL**

(New)

**Purpose:** Terminate batch file processing.

**Format:** CANCEL [*value*]

***value*:** The numeric exit code to return to the command processor.

See also: CALL and QUIT.

**Usage:** The CANCEL command ends all batch file processing, regardless of the batch file nesting level. Use QUIT to end a nested batch file and return to the previous batch file.

You can CANCEL at any point in a batch file. If CANCEL is used from within an alias it will end execution of both the alias and any batch files which are running at the time.

The following batch file fragment compares an input line to “end” and terminates all batch file processing if it matches:

```
input Enter your choice: %%option
if "%option" == "end" cancel
```

- ❖ If you specify a ***value***, CANCEL will set the ERRORLEVEL or exit code to that value (see the IF command, and the %? variable on page 155).



**CD / CHDIR**

(Enhanced)

**Purpose:** Display or change the current directory.

**Format:** CD [/D /N] [ *path* | - ]

or

CHDIR [/D /N] [ *path* | - ]

***path*:** The directory to change to, including an optional drive name.

**/D**(rive)

**/N**(o extended search)

See also: CDD, MD, PUSHD, RD, and Directory Navigation on page 81.

**Internet:** Can be used with FTP servers.

**Usage:** CD and CHDIR are synonyms. You can use either one.

CD lets you navigate through a drive's subdirectory structure by changing the current working directory. If you enter CD and a directory name, the named directory becomes the new current directory. For example, to change to the subdirectory *C:\FINANCE\MYFILES*:

```
[c:\] cd \finance\myfiles  
[c:\finance\myfiles]
```

Every disk drive on the system has its own current directory. Specifying both a drive and a directory in the CD command will change the current directory on the specified drive, but will not change the default drive. For example, to change the default directory on drive A:

```
[c:\] cd a:\utility  
[c:\]
```

Notice that this command does not change to drive A:. Use the CDD command to change the current drive and directory at the same time.

When you use CD to change to a directory on an LFN drive, you must quote the ***path*** name if it contains white space or special characters. See page 15 for additional details.

You can change to the parent directory with **CD ..**; you can also go up one additional directory level with each additional **[.]**. For example, **CD ....** will go up three levels in the directory tree (see page 94 for additional details). You can move to a sibling directory – one that branches from the same parent directory as the current subdirectory – with a command like **CD ../newdir**.

If you enter **CD** with no argument or with only a disk drive name, it will display the current directory on the default or named drive.

If **CD** cannot change to the directory you have specified it will attempt to search the **CDPATH** (see page 86) and the extended directory search database (see page 83) in order to find a matching directory and switch to it. You can disable this default extended search with **/N**. You can also use wildcards in the *path* to force an extended directory search (see page 83). Read the section on Directory Navigation beginning on page 81 for complete details on these and other directory navigation features.

**CD** saves the current directory before changing to a new directory. You can switch back to the previous directory by entering **CD -**. (There must be a space between the **CD** command and the hyphen.) You can switch back and forth between two directories by repeatedly entering **CD -**. The saved directory is the same for both the **CD** and **CDD** commands. Drive changes and automatic directory changes (see page 72) also modify the saved directory, so you can use **CD -** to return to a directory that you exited with an automatic directory change.

Directory changes made with **CD** are recorded in the directory history list and can be displayed in the directory history window (page 82), which allows you to return quickly to a recently-used directory.

**4NT, TC** You can also use **CD** to change the current directory on an FTP server opened with **IFTP**. To do so, simply enter the desired pathname in quotes, for example:

```
[c:\] cd "ftp:/etc"
```

FTP directory changes do not use the **CDPATH** or extended directory search database.

- ❖ **CD** never changes the default drive. If you change directories on one drive, switch to another drive, and then enter **CD -**, the directory will be restored on the first drive but the current drive will not be changed.

The operating system limits the length of the full subdirectory name. See page 13 for more information on directory names.

**4NT, TC** When changing directories, 4NT and Take Command normally maintain the original case of each path element. This is necessary for a few programs which are case-sensitive in their use of directory names. If you do not use such a program, disabling this case preservation will speed up directory changes slightly; to do so, see the SaveDirCase directive on page 236.

**Options:**    **/D(rive):** Changes the current drive as well as directory. This option is included only for compatibility with the undocumented CD /D command command available in *CMD.EXE*. In most cases you should use CDD, which performs the same function.

**/N(o extended search):** Skips the standard extended directory search (see page 83) when the directory is not found. This option is useful in batch files to force an error – rather than an extended search – if a directory is not found.

**CDD**

(New)

**Purpose:** Change the current disk drive and directory.

**Format:** CDD [/A /D[drive ...] /N /S[drive ...] /U[drive ...]] [*path* | -]

***path*:** The name of the directory (or drive and directory) to change to.

***drive*:** A drive or list of drives to include in the extended directory search database.

/A(all drives)

/S(earch tree)

/D(etele from tree)

/U(pdate tree)

/N(o extended search)

See also: CD, MD, PUSH, RD, and Directory Navigation on page 81.

**Usage:** CDD is similar to the CD command, except that it also changes the default disk drive if one is specified. CDD will change to the directory and drive you name. To change from the root directory on drive A to the subdirectory *C:\WP*:

```
[a:\] cdd c:\wp  
[c:\wp]
```

You can change to the parent directory with **CDD ..**; you can also go up one additional directory level with each additional **[.]**. For example, **CDD ....** will go up three levels in the directory tree (see page 94 for additional details).

CDD can also change to a network drive and directory specified with a UNC name (see page 13 for information on UNC names).

When you use CDD to change to a directory on an LFN drive, you must quote the ***path*** name if it contains white space or special characters. See page 15 for additional details.

If CDD cannot change to the directory you have specified it will attempt to search the CDPATH (see page 86) and the extended directory search database (see page 83) in order to find a matching directory and switch to it. You can disable this default extended search with /N. You can also use wildcards in the ***path*** to force an extended directory search (see page 85). Read the section on Directory Navigation beginning on page 81 for complete details on these and other directory navigation features.

CDD saves the current drive and directory before changing to a new directory. You can switch back to the previous drive and directory by entering **CDD -** (there must be a space between the CDD command and the hyphen). You can switch back and forth between two drives and directories by repeatedly entering **CDD -**. The saved directory is the same for both the CD and CDD commands. Drive changes and automatic directory changes (see page 72) also modify the saved directory, so you can use **CDD -** to return to a directory you exited with a drive change or an automatic directory change.

Directory changes made with CDD are recorded in the directory history list, and can be displayed in the directory history window (see page 74), which allows you to return quickly to a recently-used directory.

The operating system limits the permissible length of the full subdirectory name (see page 13 for information on directory names).

**4NT, TC** When changing directories, 4NT and Take Command normally maintain the original case of each path element. This is necessary for a few programs which are case-sensitive in their use of directory names. If you do not use such a program, disabling this case preservation will speed up directory changes slightly; to do so, see the *SaveDirCase* directive on page 236.

**Options:** **/A**(ll drives): When CDD is used with this option, it displays the current directory on all drives from C: to the last drive in the system. You cannot move to a new drive and directory and use **/A** in the same command.

**/D**(elete): Removes the specified drives or directory trees from *JPSTREE.IDX*. Uses the same syntax for drive and directory names as **/S**. For example, to delete the directories under *F:\MYDIR* from *JPSTREE.IDX*:

```
cdd /d f:\mydir
```

**/N**(o extended search): Skips the standard extended directory search (see page 83) when the directory is not found. This option is useful in batch files to force an error – rather than an extended search – if a directory is not found.

**/S**(earch tree): Builds or rebuilds the Extended Directory Search database, *JPSTREE.IDX* (see page 83). You cannot move to a new drive and directory and use **/S** in the same command.

To include all local hard drives in the database, use the command:

```
cdd /s
```

To limit or add to the list of drives included in the database, list the drives and network volume names after the **/S** switch. For example, to include drives C, D, E, and the network volume `\\server\dir1` in the database, use this command:

```
cdd /s cde \\server\dir1
```

All non-hidden directories on the listed drives will be indexed. (CDD **/S** will also index the hidden directories if the CompleteHidden .INI directive is set.) You cannot restrict the database to certain directories within a drive. Each time you use **/S**, everything in the previous directory database is replaced by the new database that is created. To update the database see **/U** below.

You can index specific subdirectories rather than an entire drive. For example, to index all directories on drive C but only the MSSDK directory tree on drive D:

```
cdd /s c:\ d:\mssdk
```

Note that if you use this format, you must provide a complete drive specification (i.e., “c:\”) for each drive to be indexed.

**/U(update):** Updates *JPSTREE.IDX* with the specified drives and directories instead of rebuilding the directory database. Uses the same syntax for drive and directory names as **/S**. For example, to update the *D:\MSSDK* tree and all of drive E:

```
cdd /u d:\mssdk e:\
```

**CHCP** [4DOS, 4NT] (Compatible)

**Purpose:** Display or change the current system code page.

**Format:** CHCP [*n*]

***n*:** A system code page number.

❖**Usage:** Code page switching allows you to select different character sets for language support. CHCP is not available in Take Command.

If you enter CHCP without a number, the current code page is displayed:

```
[c:\] chcp
Active code page: 437
```

If you enter CHCP plus a code page number, the code page is changed. For example, to set the code page to multilingual:

```
[c:\] chcp 850
```

When you use CHCP under DOS, it affects the entire system. When you use CHCP under Windows it only affects the current process, and any new programs started from within that process; the active code page in other processes remains unchanged.

**4DOS** Before using CHCP under DOS, you must first load the device drivers (in *CONFIG.SYS*), make sure the information file (*COUNTRY.SYS*) is available, load national language support (using the NLSFUNC command), and prepare the specified code page for the devices (using the MODE command with the CODEPAGE PREPARE option). CHCP accepts one of the prepared system code pages. An error message is displayed if a code page is selected that has not been prepared for the system.

See your DOS or Windows documentation for more information on CHCP.

**CLS**

(Enhanced)

**Purpose:** Clear the window and move the cursor to the upper left corner; optionally change the default display and border (DOS only) colors.

**Format:** CLS [/C /S] [[BRiGht] [BLInk] *fg* ON [BRiGht] *bg*] [BORder *bc*]

***fg*:** The new foreground color (**BLI** only valid in full-screen 4DOS)

***bg*:** The new background color

***bc*:** The new border color (DOS only)

**/C**(lear buffer)

**/S**(croll buffer)

**Usage:** CLS can be used to clear the window without changing colors, to clear the window and change the colors simultaneously, or to clear the entire scrollbar buffer. These two examples show how to clear the window to the default colors, and to bright white letters on a blue background:

```
[c:\] cls
```

```
[c:\] cls bright white on blue
```

CLS is often used in batch files before displaying text.

See page 30 for details about colors and color names, and notes on the use of blinking characters and bright background colors.

**4DOS** ❖In 4DOS if *ANSI.SYS* or a compatible driver is not loaded the colors will not be “sticky” – you may lose them after you run an application. See page 28 for additional details. If your display accommodates more than 25 rows by 80 columns and CLS doesn't clear the whole screen, your ANSI driver probably does not support the large display properly.

**Options:** **/C**(lear buffer): Clear the entire 4NT or Take Command scrollbar buffer. If **/C** is not used, only the visible portion of the screen is cleared.

**4NT** **/S**(croll buffer): Clear the screen by scrolling the buffer, rather than filling the screen with blanks (the default method ). This saves the text on the screen into the scrollbar buffer if it is larger than the visible window. This switch may not give the expected results when the buffer size is less than twice the window size.



**COLOR**

(New)

**Purpose:** Change the default display colors.

**Format:** COLOR [BRiGht] [BLInk] *fg* ON [BRiGht] *bg* [BORder *bc*]

***fg*:** The new foreground color

***bg*:** The new background color

***bc*:** The new border color (DOS only)

See also: CLS.

**Usage:** COLOR is normally used in batch files before displaying text. For example, to set screen colors to bright white on blue, you can use this command:

```
[c:\] color bright white on blue
```

The **BLInk** keyword for blinking text is only valid in full-screen 4DOS sessions. It does not work in 4NT, Take Command, or in 4DOS running in a window under Windows 95 / 98 / ME.

**4DOS** If you have an ANSI driver (such as *ANSI.SYS*) installed, COLOR will not change anything on the screen. It will only set the default colors for subsequent screen displays. If you are not using an ANSI driver, COLOR will change the display colors of every character on the screen. However, the colors will not be “sticky” – you may lose them after you run an application. See page 28 for additional details on ANSI drivers.

**4DOS** ❖If you see odd characters like “[44;37m” when you set the screen colors, 4DOS thinks you have an ANSI driver loaded when you don't. Use SETDOS /A2, the Display page of the OPTION dialogs, or ANSI = No in *4DOS.INI*, to tell 4DOS you have no ANSI driver.

**4NT, TC** ❖4NT and Take Command also support the same syntax as CMD.EXE:

COLOR bf

Where **b** is a hexadecimal digit that specifies the background color and **f** is a hexadecimal digit that specifies the foreground color. See your Windows documentation for more information.

**COPY**

(Enhanced)

**Purpose:** Copy data between disks, directories, files, or physical hardware devices (such as your printer or serial port).

**Format:** COPY [/A:[-][+]rhsad] /C /D /E /G /H /I"text" /J /L /K /M /N /O /P /Q /R /S /T /U /V /X /Z] [*@file*] *source*[+] ... [/A /B] *destination* [/A /B]

***source*:** A file or list of files or a device to copy **from**.

***destination*:** A file, directory, or device to copy **to**.

***@file*:** A text file containing the names of the ***source*** files to copy, one per line (see page 109 for details).

/A(SCII)	/N(othing)
/A: (Attribute select)	/O (copy if not exist)
/B(inary)	/P(rompt)
/C(hanged)	/Q(quiet)
/E (no error messages)	/R(eplace)
/G (percent copied)	/S(ubdirectories)
/H(idden)	/T(otals)
/I"text" (match description)	/U(pdate)
/J (copy in restartable mode)	/V(erify)
/K(eep attributes)	/X (clear archive)
/L (ASCII FTP transfer)	/Z (overwrite)
/M(odified)	

See also: ATTRIB, MOVE, and REN.

**Files:** Supports extended wildcards, ranges, multiple file names, include lists (see pages 94 - 104), and the clipboard device (CLIP:). Date, time, size or exclude ranges anywhere on the line apply to all ***source*** files. Use extended wildcards with caution on LFN volumes; see page 105 for details.

**Internet:** Can be used with FTP and HTTP servers.

**Usage:** The COPY command accepts all traditional syntax and options and adds many new features.

The simplest use of COPY is to make a copy of a file, like this example which makes a copy of a file called *FILE1.ABC*:

```
[c:\] copy file1.abc file2.def
```

You can also copy a file to another drive and / or directory. This command copies *FILE1* to the *\MYDIR* directory on drive E:

```
[c:\] copy file1 e:\mydir
```

When you COPY files to or from an LFN drive, you must quote any file names which contain white space or special characters. See page 15 for additional details.

### ***Copying Files***

You can copy several files at once by using wildcards:

```
[c:\] copy *.txt e:\mydir
```

(See page 94 for an explanation of the wildcard characters \* and ?.)

You can also list several ***source*** files in one command. The following command copies 3 files from the current directory to the *\MYDIR* directory on drive E:

```
[c:\] copy file1 file2 file3 e:\mydir
```

COPY also understands include lists (see page 104), so you can specify several different kinds of files in the same command. This command copies the *.TXT*, *.DOC*, and *.BAT* files from the *E:\MYDIR* directory to the root directory of drive A:

```
[c:\] copy e:\mydir\*.txt;*.doc;*.bat a:\
```

If there is only one argument on the line, COPY assumes it is the ***source***, and uses the current drive and directory as the ***destination***. For example, the following command copies all the *.DAT* files on drive A to the current directory on drive C:

```
[c:\data] copy a:*.dat
```

If there are two or more arguments on the line, separated by spaces, then COPY assumes that the last argument is the ***destination*** and copies all ***source*** files to this new location. If the ***destination*** is a drive, directory, or device name then the ***source*** files are copied individually to the new location. If the ***destination*** is a file name, the first ***source*** file is copied to the ***destination***, and any additional ***source*** files are then appended to the new ***destination*** file.

For example, the first of these commands copies the *.DAT* files from the current directory on drive A individually to *C:\MYDIR* (which must already exist as a directory); the second appends all the *.DAT* files together into one large file called *C:\DATA* (assuming *C:\DATA* is not a directory):

```
[c:\] copy a:*.dat c:\mydir\  
[c:\] copy a:*.dat c:\data
```

When you copy to a directory, if you add a backslash [\] to the end of the name as shown in the first example above, COPY will display an error message if the name does not refer to an existing directory. You can use this feature to keep COPY from treating a mistyped **destination** directory name as a file name and attempting to append all your **source** files to a **destination** file, when you really meant to copy them individually to a **destination** directory.

To copy text to or from the clipboard, use **CLIP:** as the device name. Using CLIP: with non-text data will produce unpredictable results. See page 88 for additional information about CLIP:, including details on when you can use the clipboard under 4DOS.

### Appending Files

- ❖ A plus [+] tells COPY to append two or more files to a single **destination** file. If you have several **source** files separated with [+] and don't specify a **destination**, COPY will use the first **source** file as the destination, and append each subsequent file to the first file.

For example, the following command will append the contents of *C:\MEMO2* and *C:\MEMO3* to *C:\MEMO1* and leave the combined contents in the file named *C:\MEMO1*:

```
[c:\] copy memo1+memo2+memo3
```

To append the same three files but store the result in *BIGMEMO*:

```
[c:\] copy memo1+memo2+memo3 bigmemo
```

If no **destination** is specified, the destination file will always be created in the current directory even if the first **source** file is in another directory or on another drive. For example, this command will append *C:\MEM\MEMO2* and *C:\MEM\MEMO3* to *D:\DATA\MEMO1*, and leave the result in *C:\MEM\MEMO1*:

```
[c:\mem] copy d:\data\memo1+memo2+memo3
```

- ❖ You cannot append files to a device (such as a printer); if you try to do so, COPY will ignore the [+] signs and copy the files individually. If you attempt to append several **source** files to a **destination** directory or disk, COPY will append the files and place the copy in the new location with the same name as the first **source** file.

#### **4NT, TC** *FTP and HTTP Usage*

If you have appropriate permissions, you can copy to and from Internet FTP and HTTP URLs. The URL must be enclosed in double quotes so the forward slashes won't be interpreted as switches. For example:

```
[c:\] copy "ftp://jpsoft.com/foo/index" index
```

Files copied to or from FTP servers are normally transferred in binary mode. To perform an ASCII transfer use the /L switch.

Wildcard characters like [\*] and [?] will be treated as wildcards in FTP URLs, but will be treated as normal characters in HTTP URLs.

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see Using FTP Servers on page 113, and the IFTP command on page 382.

#### **4NT, TC** *NTFS File Streams*

COPY supports file streams on NTFS drives under Windows NT / 2000 / XP. You can copy an individual stream by specifying the stream name, for example:

```
[c:\] copy streamfile:s1 file2
```

If no stream name is specified the entire file is copied, including all streams. However, if you copy a file to a drive or device which does not support streams, only the file's primary data is copied; any additional streams are not processed.

See NTFS File Streams on page 18 for additional details.

**❖ Advanced Features**

If your **destination** has wildcards in it, COPY will attempt to match them with the **source** names. For example, this command copies the *.DAT* files from drive A to *C:\MYDIR* and gives the new copies the extension *.DX*:

```
[c:\] copy a:*.dat c:\mydir\*.dx
```

This feature can give you unexpected results if you use it with multiple **source** file names. For example, suppose that drive A contains *XYZ.DAT* and *XYZ.TXT*. The command:

```
[c:\] copy a:\*.dat a:\*.txt c:\mydir\*.dx
```

will copy *A:\XYZ.DAT* to *C:\MYDIR\XYZ.DX*. Then it will copy *A:\XYZ.TXT* to *C:\MYDIR\XYZ.DX*, overwriting the first file.

You can use date, time, and size ranges to further define the files that you want to copy (see page 97 for information on ranges). This example copies every file in the *E:\MYDIR* directory, which was created or modified yesterday, and which is also 10,000 bytes or smaller in size, to the root directory of drive A:

```
[c:\] copy /[d-1] /[s0,10000] e:\mydir\*.* a:\
```

You can also use file exclusion ranges to restrict the list of files that would normally be selected with wildcards. This example copies every file in the *E:\MYDIR* directory except backup (*.BAK* or *.BK!*) files:

```
[c:\] copy /[!*.bak;*.bk!] e:\mydir\*.* a:\
```

COPY will normally process **source** files which do not have the hidden or system attribute, and will ignore the read-only and archive attributes. It will always set the archive attribute and clear the read-only attribute of **destination** files. In addition, if the **destination** is an existing file with the read-only attribute, COPY will generate an "Access Denied" error and refuse to overwrite the file. You can alter some of these behaviors with switches (see the individual switch descriptions below for complete details):

**/A:** Forces COPY to process *source* files with the attributes you specify (see page 111 for more information about specifying attributes).

**/H** Forces COPY to process hidden and system *source* files, and preserves the hidden and system attributes when creating *destination* files.

**/K** Retains the read-only attribute from each *source* file when creating the *destination* file.

**/Z** Forces COPY to overwrite an existing read-only *destination* file.

- ! Use caution with **/A:**, **/H**, or **/K** when both the *source* and *destination* directories contain file descriptions. If the *source* specification matches the description file name (normally *DESCRIPT.ION*), and you use a switch which tells COPY to process hidden files, the *DESCRIPT.ION* file itself will be copied, overwriting any existing descriptions in the *destination* directory. For example, if the *\DATA* directory contains file descriptions this command would overwrite any existing descriptions in the *\SAVE* directory:

```
[c:\data] copy /h d*.* \save\
```

**Options:** The **/A(SCII)** and **/B(inary)** options apply to the preceding filename and to all subsequent filenames on the command line until the file name preceding the next **/A** or **/B**, if any. The other options apply to all filenames on the command line, no matter where you put them. For example, either of the following commands could be used to copy a font file to the printer in binary mode:

```
[c:\] copy /b myfont.dat prn  
[c:\] copy myfont.dat /b prn
```

Some options do not make sense in certain contexts, in which case COPY will ignore them. For example, you cannot prompt before replacing an existing file when the *destination* is a device such as the printer – there's no such thing as an “existing file” on the printer.

Options used in less common situations have been marked with ❖ below. Remember that the options are in alphabetical order, so more basic options are interspersed with those marked with ❖

- ❖ **/A(SCII):** If you use **/A** with a *source* filename, the file will be copied up to, but not including, the first Ctrl-Z (Control-Z or ASCII 26)

character in the file (some application programs use the Ctrl-Z to mark the end of a file). If you use **/A** with a *destination* filename, a Ctrl-Z will be added to the end of the file. **/A** is the default when appending files, or when the *destination* is a device like NUL or PRN, rather than a disk file. Also see **/B**.

- ❖ **/A:** (Attribute select): Select only those files that have the specified attribute(s) set. See **Attribute Switches** on page 111 for information on the attributes which can follow **/A:**. You **must** include the colon with this option to distinguish it from the **/A(SCII)** switch, above. See the cautionary note under **Advanced Features** above before using **/A:** when both *source* and *destination* directories contain file descriptions.
- ❖ **/B**(inary): If you use **/B** with a *source* filename, the entire file is copied; Ctrl-Z characters in the file do not affect the copy operation. Using **/B** with a *destination* filename prevents addition of a Ctrl-Z to the end of the *destination* file. **/B** is the default for normal file copies. Also see **/A**.

**/C**(hanged files): Copy files only if the *destination* file exists and is older than the *source* (see also **/U**). This option is useful for updating the files in one directory from those in another without copying any newly created files. (Before using **/C** in a network environment be sure to read the note under **/U**.)

- ❖ **/E** (No error messages): Suppress all non-fatal error messages, such as "File not found." Fatal error messages, such as "Drive not ready," will still be displayed. This option is most useful in batch files and aliases.
- ❖ **/G:** Displays the percent copied. This is useful when copying large files across a network or via FTP to ensure the copy is proceeding.
- ❖ **/H**(idden): Copy all matching files including those with the hidden and / or system attribute set (see page 18). See the cautionary note under **Advanced Features** above before using **/H** when both *source* and *destination* directories contain file descriptions.

**/I"text"**: Select *source* files by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.



**4NT, TC** **/J**: Copy the file in restartable mode. The copy progress is tracked in the destination file in case the copy fails. The copy can be restarted by specifying the same source and destination file names.

- ❖ **/K**(eep attributes): To maintain compatibility with CMD.EXE, *COMMAND.COM*, and network operating systems, COPY normally maintains the hidden and system attributes, sets the archive attribute, and removes the read-only attribute on the target file. **/K** tells COPY to also maintain the read-only attribute on the *destination* file. However, if the *destination* is on a Novell NetWare volume, this option may fail to maintain the read-only attribute. This is due to the way NetWare handles file attributes, and is not a problem in COPY.

**4NT, TC** **/L**: Perform FTP transfers in ASCII mode, instead of the default binary mode.

**/M**(odified): Copy only those files with the archive attribute set (see page 18), *i.e.*, those which have been modified since the last backup. The archive attribute of the *source* file will **not** be cleared after copying; to clear it use the **/X** switch, or use ATTRIB (page 263).

**/N**(othing): Do everything except actually perform the copy. This option is useful for testing what the result of a complex COPY command will be. **/N** does **not** prevent creation of *destination* subdirectories when it is used with **/S**.

**/O**: Only copy the source file if the target file doesn't exist.

**/P**(rompt): Ask the user to confirm each *source* file. Your options at the prompt are explained in detail on page 80.

**/Q**(uiet): Don't display filenames or the total number of files copied. This option is most often used in batch files. See also **/T**.

**/R**(eplace): Prompt the user before overwriting an existing file. Your options at the prompt are explained in detail on page 80.

**/S**(ubdirectories): Copy the subdirectory tree starting with the files in the *source* directory plus each subdirectory below that. The *destination* must be a directory; if it doesn't exist, COPY will attempt to create it. COPY will also attempt to create needed subdirectories on the tree below the *destination*, including empty *source* directories. If COPY **/S** creates one or more destination directories, they will be added automatically to the extended directory search database; see page 83 for details.

If you attempt to use **COPY /S** to copy a subdirectory tree into part of itself, **COPY** will detect the resulting infinite loop, display an error message, and exit.

Do not use **/S** with **@file** lists; see page 109 for details.

**/T**(otals): Turns off the display of filenames, like **/Q**, but does display the total number of files copied.

**/U**(pdate): Copy each **source** file only if it is newer than a matching **destination** file or if a matching **destination** file does not exist (see also **/C**). This option is useful for keeping one directory matched with another with a minimum of copying.

The time comparisons used with **/U** (and **/C**) may not always work reliably across a network, including on FTP servers, due to differences in time zone and in the file time storage method between the local and remote systems. Be sure to do some non-destructive testing (*e.g.* with **/N**) before depending on this option to yield the results you want in a network environment.

- ❖ **/V**(erify): Verify disk writes. This is the same as executing the **VERIFY ON** command, but is only active during the **COPY**. **/V** does **not** read back the file and compare its contents with what was written; it only verifies that the data written to disk is physically readable.

**/X**: Clear the archive attribute from the source file after a successful copy. This option is most useful if you are using **COPY** to maintain a set of backup files.

- ! **/Z**: Overwrite read-only **destination** files. Without this option, **COPY** will fail with an “Access denied” error if the **destination** file has its read-only attribute set. This option allows **COPY** to overwrite read-only files without generating any errors.

**CTTY** [4DOS] (Compatible)

**Purpose:** Change the default console device.

**Format:** CTTY *device*

***device:*** The new console device.

❖ **Usage:** Normally, 4DOS uses the keyboard as the standard input device and the display as the standard output device. Together, the keyboard and display are known as the console or CON. The CTTY command allows you to substitute another device that can perform standard character I/O for the console.

For example to change the console to the first serial port:

```
c:\> cttty com1
```

Change the console back to the standard keyboard and display (this command must be entered from the current console, *e.g.*, a terminal attached to COM1, or from a batch file):

```
c:\> cttty con
```

CTTY works only for programs and commands that use standard DOS input and output functions. This includes all 4DOS internal commands except DRAWBOX, DRAWHLINE, DRAWVLINE, LIST, SCREEN, SCRPUT, SELECT, and VSCRPUT. In addition, if you use color-coded directories you should disable them with DIR /D when using CTTY. Otherwise, directories will not be displayed correctly.

**DATE**

(Enhanced)

**Purpose:** Display and optionally change the system date.

**Format:** DATE [/T] [*mm-dd-yy*]

***mm***: The month (1 - 12).

***dd***: The day (1 - 31).

***yy***: The year (80 - 99, or a 4-digit year).

**/T** (Display only)

See also: TIME.

**Usage:** If you simply type DATE without any parameters, you will see the current system date and time, and be prompted for a new date. Press ENTER if you don't wish to change the date. If you type a new date, it will become the current system date, which is included in the directory entry for each file as it is created or altered:

```
[c:\] date
Mon Dec 23, 2002 9:30:06
Enter new date (mm-dd-yy):
```

You can also enter a new system date by typing the DATE command plus the new date on the command line:

```
[c:\] date 12-16-2002
```

You can use hyphens, slashes, or periods to separate the month, day, and year entries. The year can be entered as a 2-digit or 4-digit value. Two-digit years between 80 and 99 are interpreted as 1980 - 1999; values between 00 and 79 are interpreted as 2000 - 2079.

DATE adjusts the format it expects depending on your country settings. When entering the date, use the correct format for the country setting currently in effect on your system.

You can also use the international date format **yyyy-mm-dd**.

**Option:** **/T** (Display only): Displays the current date but does not prompt you for a new date. If a new date is specified in the same command as **/T** the new date will be ignored.

## **DDEEXEC**    [TC]

(New)

**Purpose:**    Send a DDE command to another application.

**Format:**    DDEEXEC *server, topic, command*

***server***:    The DDE name of the program that will receive the command.

***topic***:    The server's topic name for receiving the command.

***command***:    The command string to send to the server.

**Usage:**    Windows supports a form of communication between programs called Dynamic Data Exchange or DDE. Using DDE, one program can send a command or data to another. The receiving program is usually called the DDE server; the sending program is usually called the DDE client. When you use DDEEXEC, Take Command acts as a DDE client and the program which receives the command is the server. (Take Command can also act as a DDE server. See page 147 for details.)

For example, if you wanted to instruct Microsoft Internet Explorer to open JP Software's web site, you could use this command (enter this on one line):

```
[c:\] ddeexec iexplore, WWW_OpenURL,  
"http://jpsoft.com/"
```

In this example, the ***server*** name is **iexplore**, the ***topic*** name is **WWW\_OpenURL**, and the ***command*** is **"http://jpsoft.com/"**. (Internet Explorer must be running, and the version on your system must support this syntax, for this example to work.)

The server name, topic name, and possible commands are defined by the server application. See the documentation included with the programs to which you want to send DDE messages for details about the names and commands to use.

**DEL / ERASE**

(Enhanced)

**Purpose:** Erase one file, a group of files, or entire subdirectories.

**Format:** DEL [/A:[-+]rhsad] /E /F /I"text" /N /P /Q /R /S /T /W /X /Y /Z] [*@file*]  
*file...*

or

ERASE [/A:[-+]rhsad] /E /F /I"text" /N /P /Q /R /S /T /W /X /Y /Z] [*@file*]  
*file...*

***file*:** The file, subdirectory, or list of files or subdirectories to erase.

***@file*:** A text file containing the names of the files to delete, one per line (see page 109 for details).

/A: (Attribute select)	/R(ecycle bin)
/E (No error messages)	/S(ubdirectories)
/F(orce delete)	/T(otal)
/I"text" (match description)	/W(ipe)
/K (no Recycle Bin)	/X (remove empty subdirectories)
/N(othing)	/Y(es to all prompts)
/P(rompt)	/Z(ap hidden and read-only files)
/Q(uiet)	

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104). Use extended wildcards with caution on LFN volumes; see page 105 for details.

**Internet:** Can be used with FTP servers.

**Usage:** DEL and ERASE are synonyms, you can use either one.

- ! Use the DEL and ERASE commands with caution; the files and subdirectories that you erase may be very difficult or impossible to recover once they are deleted.

To erase a single file, simply enter the file name:

```
[c:\] del letters.txt
```

You can also erase multiple files in a single command. For example, to erase all the files in the current directory with a *.BAK* or *.PRN* extension:

```
[c:\] del *.bak *.prn
```

When you use DEL on an LFN drive, you must quote any file names which contain white space or special characters. See page 15 for additional details.

To exclude files from a DEL command, use a file exclusion range (see page 102). For example, to delete all files in the current directory except those whose extension is *.TXT*, use a command like this:

```
[c:\] del /[*.*.TXT] *.*
```

When using exclusion ranges or other more complex options you may want to use the */N* switch first, to preview the effects of the DEL without actually deleting any files.

If you enter a subdirectory name, or a filename composed only of wildcards (*\** and */* or *?*), DEL asks for confirmation (*Y* or *N*) unless you specified the */Y* option. If you respond with a *Y*, DEL will delete all the files in that subdirectory (hidden, system, and read-only files are only deleted if you use the */Z* option).

**4NT, TC** By default, 4NT and Take Command ask for confirmation of wildcard-only and directory deletions, as described above. The traditional Windows NT / 2000 / XP command processor, *CMD.EXE*, behaves the same way but does not ask for confirmation if you use */Q* to delete files quietly. If you want 4NT and Take Command to follow *CMD.EXE*'s approach and skip the confirmation prompt when */Q* is used, set *DelGlobalQuery* to No in the .INI file (see page 220). **Use caution if you set *DelGlobalQuery* to No**, as this will allow DEL */Q* to delete an entire directory without prompting for confirmation.

DEL displays the amount of disk space recovered, unless the */Q* option is used (see below). It does so by comparing the amount of free disk space before and after the DEL command is executed. This amount may be incorrect if you are using a deletion tracking system which stores deleted files in a hidden directory, or if another program performs a file operation while the DEL command is executing.

Remember that DEL removes file descriptions along with files. Most deletion tracking systems will not be able to save or recover a file's description, even if they can save or recover the data in a file.

**4NT, TC** When you delete files with DEL, 4NT and Take Command will not by default move the deleted files to the Windows Recycle Bin. You can

change this default on the “Startup” tab of the configuration dialog, or with the RecycleBin directive in the *.INI* file (see page 225). If you want to override the default setting on a one-time basis, and place the files in the Recycle Bin, use the **/R** option:

```
[c:\] del /r letters.txt
```

If you have enabled the Recycle Bin support, you can override this setting and delete files without placing them in the Recycle Bin, with the **/K** option:

```
[c:\] del /k letters.txt
```

- ! When a file is deleted, its disk space is returned to the operating system for use by other files. However, the contents of the file remain on the disk until they are overwritten by another file. If you wish to obliterate a file or wipe its contents clean, use **DEL /W**, which overwrites the file with zeros before deleting it. Use this option with caution — once a file is obliterated, it is usually impossible to recover.

DEL returns a non-zero exit code if no files are deleted, or if another error occurs. You can test this exit code with the `%_?` internal variable (see page 155), and use it with conditional commands (`&&` and `||`; see page 116).

#### **4NT, TC** *FTP Usage*

If you have appropriate permissions, you can delete files on FTP servers. The FTP name must be enclosed in double quotes so the forward slashes won't be interpreted as switches. For example:

```
[c:\] del "ftp://ftp.somedomain.com/index"
```

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see *Using FTP Servers* on page 113, and the IFTP command on page 382.

#### **4NT, TC** *NTFS File Streams*

DEL supports file streams on NTFS drives under Windows NT / 2000 / XP. You can delete an individual stream by specifying the stream name, for example:

```
[c:\] del streamfile:s1
```



If no stream name is specified the entire file is deleted, including all streams.

See NTFS File Streams on page 18 for additional details.

**Options:**    **/A:** (Attribute select): Delete only those files that have the specified attribute(s) set. See **Attribute Switches** on page 111 for information on the attributes which can follow **/A:**.

**/E** (No error messages): Suppress all non-fatal error messages, such as "File Not Found." Fatal error messages, such as "Drive not ready," will still be displayed. This option is most useful in batch files and aliases.

**4NT, TC**    **/F**(orce delete): This option has the same effect as **/Z** (see below): it deletes read-only, hidden, and system files as well as normal files. It is included for compatibility with *CMD.EXE*.

**/I"text":** Select files by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**4NT, TC**    **/K:** Physically delete files instead of sending them to the Windows Recycle Bin. This option overrides the default "RecycleBin=Yes" *.INI* setting.

**/N**(othing): Do everything except actually delete the file(s). This is useful for testing what the result of a DEL would be.

**/P**(rompt): Prompt the user to confirm each erasure. Your options at the prompt are explained in detail on page 80.

**/Q**(uiet): Don't display filenames as they are deleted, or the number of files deleted or bytes freed. If *DelGlobalQuery* is set to No in the *.INI* file then **/Q** also disables the normal confirmation prompt when performing wildcard deletions (*e.g.* **DEL \*.\***), for compatibility with the traditional Windows NT / 2000 / XP command processor, *CMD.EXE*. **Use caution if you set *DelGlobalQuery* to No**, as this will allow **DEL /Q** to delete an entire directory without prompting for confirmation. See also **/T**.

**4NT, TC /R:** Delete files to the Windows Recycle Bin. This option overrides a "RecycleBin=No" .INI setting.

- !** **/S**(ubdirectories): Delete the specified files in this directory and all of its subdirectories. This can be used to delete all the files in a subdirectory tree or even a whole disk. It should be used with caution!

The **/S** switch cannot be used when the files are on an FTP server; see page **Error! Bookmark not defined.** for details. Also, do not use **/S** with @file lists; see page 109 for details.

**/T**(otal): Don't display filenames as they are deleted, but display the total number of files deleted plus the amount of free disk space recovered. Unlike **/Q**, the **/T** option will not speed up deletions under DOS.

- !** **/W**(ipe): Clear the file to zeros before deleting it. Use this option to completely obliterate a file's contents from your disk. Once you have used this option it is usually impossible to recover the file even if you are using an undelete utility, because the contents of the file are destroyed before it is deleted. **/W** overwrites the file only once; it does **not** adhere to security standards which require multiple overwrites with varying data when destroying sensitive information. In 4NT and Take Command, **/W** will override a **/R** or a RecycleBin = Yes .INI setting.
- ❖ **/X** (remove empty subdirectories): Remove empty subdirectories after deleting (only useful when used with **/S**). If DEL deletes one or more directories, they will be removed automatically from the extended directory search database; see page 83 for details.
- !** ❖ **/Y**(es): The reverse of **/P** – it assumes a **Y** response to everything, including deleting an entire subdirectory tree. 4DOS, 4NT, and Take Command normally prompt before deleting files when the name consists only of wildcards or a subdirectory name (see above); **/Y** overrides this protection, and should be used with extreme caution!
- !** ❖ **/Z**(ap): Delete read-only, hidden, and system files as well as normal files. Files with the read-only, hidden, or system attribute set are normally protected from deletion; **/Z** overrides this protection, and should be used with caution. Because EXCEPT works by hiding files, **/Z** will override an EXCEPT command. However, files specified in a file exclusion range (see page 102) will not be deleted by DEL **/Z**.

For example, to delete the entire subdirectory tree starting with C:\UTIL, including hidden and read-only files, without prompting (use this command with CAUTION!):

```
[c:\] del /sxyz c:\util\
```

**DELAY**

(New)

**Purpose:** Pause for a specified length of time.

**Format:** DELAY [/B /M *time*]

***time*:** The number of seconds or milliseconds to delay.

**/B**(reak enabled)

**/M**(illiseconds)

**Usage:** DELAY is useful in batch file loops while waiting for something to occur. To wait for 10 seconds:

```
delay 10
```

DELAY is most useful when you need to wait a specific amount of time for an external event, or check a system condition periodically. For example, this 4NT / Take Command batch file checks the battery status every 15 seconds, and gives a warning when battery life falls below 30%:

```
do forever
  iff %_batterypercent lt 30 then
    beep 440 4 880 4 440 4 880 4
    echo Low Battery!!
  endiff
  delay 15
enddo
```

The ***seconds*** value can be as large as 3600 (one hour) in 4DOS, and about 1 billion seconds (34 years!) in 4NT and Take Command.

4DOS, 4NT, and Take Command use the minimum possible processor time during a DELAY, in order to allow other applications full use of system resources.

You can cancel a delay by pressing **Ctrl-C** or **Ctrl-Break**.

**Options:** **/B**(reak): Allows terminating a DELAY by pressing a key.

❖**/M**(illiseconds): Count by milliseconds instead of seconds. Normally only used for delays of less than 1 second. Due to the limitations of DOS-based timers, the minimum resolution in 4DOS is about 55 ms.

## DESCRIBE

(New)

**Purpose:** Create, modify, or delete file and subdirectory descriptions.

**Format:** DESCRIBE [/A:[-][+]*rhsad*] /I"text" [*@file*] *file* [/D] "*description*" ...

***file*:** The file, directory, or list of files and directories to operate on.

***@file*:** A text file containing the names of the files on which to operate, one per line (see page 109 for details).

***"description"*:** The description to attach to the file.

**/A:** (Attribute select)

**/I** (match description)

**/D**(escription follows)

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104). Use extended wildcards with caution on LFN volumes; see page 105 for details.

**Usage:** DESCRIBE adds descriptions to files and subdirectories. The descriptions can be displayed by DIR in single-column mode, and by SELECT. Descriptions let you identify your files in much more meaningful ways than you can in a filename alone.

**TC** *You can also enter or modify descriptions in Take Command with the Descriptions dialog (see page 54). The dialog allows you to select a single file and modify its description using a dialog box, rather than using the DESCRIBE command. The information in this section also applies to descriptions created via the dialog, unless otherwise noted.*

You enter a description on the command line by typing the DESCRIBE command, the filename, and the description in quotation marks, like this:

```
[c:\] describe memo.txt "Memo to Bob about party"
```

If you don't put a description on the command line, DESCRIBE will prompt you for it:

```
[c:\] describe memo.txt
Describe "memo.txt" : Memo to Bob about party
```

If you use wildcards or multiple filenames with the DESCRIBE command and don't include the description text, you will be prompted to enter a description for each file. If you do include the description on the command line, all matching files will be given the same description.

If you use DESCRIBE on an LFN drive, you must quote the *file* name if it contains white space or special characters. See page 15 for additional details.

If you enter a quoted description on the command line, and the text matches the name of a file in the current directory, the command processor will treat the string as a quoted file name, not as description text as you intended. To resolve this problem use the /D switch immediately prior to the quoted description (with no intervening spaces). For example, if the current directory contains the files *DATA.TST* and *“Test File”*, the first of these commands will work as intended, but the second will not (in the second example the string “test file” will be treated as a second file name, when it is intended to be description text):

```
[c:\] describe data.tst /D"test file"
[c:\] describe data.tst "test file"
```

On drives which support long file names you will not see file descriptions in a normal DIR display, because DIR must leave space for the long filenames. To view the descriptions, use **DIR /Z** to display the directory in FAT format. See the DIR command for more details.

Each description can be up to 511 characters long. You can change this limit on the “Misc” tab of the configuration dialog, or with the DescriptionMax directive in the .INI file (see page 220). In order to fit your descriptions on a single line in a standard DIR display, keep them to 40 characters or less (longer descriptions are wrapped in the DIR output). DESCRIBE can edit descriptions longer than DescriptionMax (up to a limit of 511 characters), but will not allow you to lengthen the existing text.

The descriptions are stored in each directory in a hidden file called *DESCRIPT.ION*. Use the ATTRIB command to remove the hidden attribute from this file if you need to copy or delete it. *DESCRIPT.ION* is always created as a hidden file, but will not be re-hidden by 4DOS, 4NT, or Take Command if you remove the hidden attribute.

You can change the description file name with the DescriptionName directive in the .INI file (page 220) or the SETDOS /D command (page 472), and retrieve it with the %\_DNAME variable (page 159). Use caution when changing the description file name, as changing the name from the default will make it difficult to transfer file descriptions to another system.

The description file is modified appropriately whenever you perform an internal command which affects it (such as COPY, MOVE, DEL, or RENAME), but not if you use an external program (such as XCOPY or the Windows Explorer). You can disable description processing on the "Misc" tab of the configuration dialog, with the Descriptions directive in the .INI file, or with SETDOS /D.

When you COPY or MOVE files between two directories, both of which have descriptions, and you use switches which enable processing of hidden files (or you have removed the hidden attribute from *DESCRIPT.ION*), you must **use caution** to avoid overwriting existing file descriptions in the **destination** directory with the *DESCRIPT.ION* file from the **source** directory. See the notes under the **Advanced Features** sections of COPY and MOVE for additional details.

Options:   ❖**/A:** (Attribute select): Select only those files that have the specified attribute(s) set. See **Attribute Switches** on page 111 for information on the attributes which can follow **/A:**.

**/D**(escription follows): The quoted string immediately following this switch is a description, not a file name. Use **/D** to avoid any ambiguity in the meaning of quoted strings. See the **Usage** section above for details.

**/I"text"**: Select files by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**DETACH** [4NT, TC]

(Compatible)

**Purpose:** Start a DOS application or a Win32 console application program in detached mode.

**Format:** DETACH *command*

***command*:** The name of a command to execute, including an optional drive and path specification and any arguments. The name must be enclosed in quotation marks if it contains any spaces.

See also: START and TASKEND.

**Usage:** When you start a program with DETACH, that program cannot use the keyboard, mouse, or video display. It is “detached” from the normal means of user input and output. However, you can redirect the program's standard I/O to other devices if necessary, using redirection symbols (see page 88). In most cases, you should only DETACH text-mode programs, since most graphical applications cannot run without a screen or keyboard, or have their input and output redirected.

The ***command*** can be an internal command, external command, alias, or batch file. If it is not an external command, the command processor will detach a copy of itself to execute the command.

For example, the following command will detach a copy of the command processor to run the batch file *XYZ.BTM*:

```
[c:\] detach xyz.btm
```

You can also include any parameters or command line switches which the command knows how to interpret:

```
[c:\] detach "xyz.btm Monday Nebraska"
```

Once the program has started, 4NT or Take Command returns to the prompt immediately. It does not wait for a detached program to finish.

You can use the TASKEND command (see page 499) to stop a detached program which does not terminate on its own.



# DIR

(Enhanced)

**Purpose:** Display information about files and subdirectories.

**Format:** DIR [/1 /2 /4 /: /A[:][**-**]rhsad] /B /C[HP] /D /E /F /G /H /I"text" /J /K /L /M /N /O[:][**-**]acdeginrsu] /P /Q /R /S /T[:acw] /U[n] /V /W /X /Z] [*file...*]

**file:** The file, directory, or list of files or directories to display.

<b>/1</b> (one column)	<b>/L</b> (ower case)
<b>/2</b> (two columns)	<b>/M</b> (suppress footer)
<b>/4</b> (four columns)	<b>/N</b> (ormal display) or (New format)
<b>/:</b> (show streams)	<b>/O</b> (rder)
<b>/A</b> (ttribute select)	<b>/P</b> (ause)
<b>/B</b> (are)	<b>/Q</b> (show owner)
<b>/C[HP]</b> (Compression)	<b>/R</b> (disable wRap)
<b>/D</b> (isable color coding)	<b>/S</b> (ubdirectories)
<b>/E</b> (upper case)	<b>/T</b> (aTtribute) or (Time)
<b>/F</b> (ull path)	<b>/U</b> (sUmmary information)
<b>/G</b> (allocated size)	<b>/V</b> (ertical sort)
<b>/H</b> (ide dots)	<b>/W</b> (ide)
<b>/I"text"</b> (match description)	<b>/X</b> (display short names)
<b>/J</b> (ustify names)	<b>/Z</b> (use FAT format)
<b>/K</b> (suppress header)	

See also: ATTRIB, DESCRIBE, SELECT, and SETDOS.

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104).

**Internet:** Can be used with FTP servers.

**Usage:** DIR can be used to display information about files from one or more of your disk directories, in a wide range of formats. Depending on the options chosen, you can display the file name, attributes, and size; the time and date of the last change to the file; the file description; and the file's compression ratio. You can also display information in 1, 2, 4, 5, or more columns, sort the files several different ways, use color to distinguish file types, and pause after each full screen.

DIR can also display information about files elsewhere on your network, and on remote sites accessible through Internet FTP.

The various DIR displays are controlled through options or switches. The best way to learn how to use the many options available with the DIR command is to experiment. You will soon know which options you want to use regularly. You can select those options permanently by using the ALIAS command.

For example, to display all the files in the current directory, in 2 columns, sorted vertically (down one column then down the next), and with a pause at the end of each page:

```
[c:\] dir /2/p/v
```

To set up this format as the default, using an alias:

```
[c:\] alias dir=*dir /2/p/v
```

When you use DIR on an LFN drive, you must quote any file names which contain white space or special characters. See page 15 for additional details.

The following sections group DIR's features together in several categories. Many of the sections move from a general discussion to more technical material. If you find some of the information in a category too detailed for your needs, feel free to skip to the beginning of the next section. The sections are:

- Selecting Files (below)
- Default DIR Output Format (page 306)
- Switching Formats (page 308)
- Multiple Column Displays (page 309)
- Color-Coded Directories (page 310)
- Redirected Output (page 311)
- FTP Usage (page 312)
- Other Notes (page 312)
- Options (page 313)

### **Selecting Files**

DIR can display information about a single file or about several, dozens, hundreds, or thousands of files at once. To display information about a single file, just add the name of the file to the DIR command line:

```
[c:\] dir january.wks
```

The simplest way to view information about several files at once is to use wildcards. DIR can work with traditional wildcard characters (\* and ?) and the extended wildcards described on page 94. For example to display all of the *.WKS* files in the current directory:

```
[c:\] dir *.wks
```

To display all *.TXT* files whose names begin with A, B, or C:

```
[c:\] dir [abc]*.txt
```

If you don't specify a filename, DIR defaults to \*.\* on traditional FAT drives, and \* on LFN drives. This default displays all non-hidden files and subdirectories in the current directory. If you specify a filename for a **non-LFN** drive which includes some wildcards, and does not include an extension, DIR will append ".\*" to it to match all extensions.

If you link two or more filenames together with spaces, DIR will display all of the files that match the first name and then all of the files that match the second name. You may use a different drive and path for each filename. This example lists all of the *.WKS* and then all of the *.WK1* files in the current directory:

```
[c:\] dir *.wks *.wk1
```

If you use an include list (see page 104) to link multiple filenames, DIR will display the matching filenames in a single listing. Only the first filename in an include list can have a path; the other files must be in the same path. This example displays the same files as the previous example, but the *.WKS* and *.WK1* files are intermixed:

```
[c:\] dir *.wks;*.wk1
```

You can include files in the current or named directory plus all of its accessible subdirectories by using the /S option. This example displays all of the *.WKS* and *.WK1* files in the D:\DATA directory and each of its subdirectories:

```
[c:\] dir /s d:\data\*.wks;*.wk1
```

You can also select files by their attributes by using the /A option. For example, this command displays the names of all of the subdirectories of the current directory:

```
[c:\] dir /a:d
```

Finally, with the **/I** option, DIR can select files to display based on their descriptions (see the DESCRIBE command on page 299 for more information on file descriptions). DIR will display a file if its description matches the text after the **/I** switch. The search is not case sensitive. You can use wildcards and extended wildcards as part of the text. For example, to display any file described as a “Test File” you can use this command:

```
[c:\] dir /i"test file"
```

If you want to display files that include the words “test file” anywhere in their descriptions, use extended wildcards like this:

```
[c:\] dir /i"*test file*"
```

To display only those files which do **not** have descriptions, use:

```
[c:\] dir /I"[]"
```

In addition, you can use ranges (see page 97) to select or exclude specific sets of files. For example, to display all files modified in the last week, all files except those with a **.BAK** extension, and all files over 500 KB in size:

```
[c:\] dir /[d-7]
[c:\] dir /[!*.bak]
[c:\] dir /[s500K]
```

You can, of course, mix any of these file selection techniques in whatever ways suit your needs.

### ***Default DIR Output Format***

DIR’s output varies based on the type of volume or drive on which the files are stored. If the volume supports long file names, the default DIR format contains 4 columns: the date of the last file modification or write, the time of last write, the file size in bytes, and the file name. The name is displayed as it is stored on the disk, in upper, lower, or mixed case. DIR will wrap filenames from one line to the next if they are too long to fit the width of the display. The standard output format is:

```
Volume in drive C is C - BOOTUP      Serial ...
Directory of  C:\4DOS700\*.*
```

```

5/25/2001  17:46      <DIR>      .
5/25/2001  17:46      <DIR>      ..
6/07/2001   7:00    254,356  4DOS.COM
6/07/2001   7:00    744,092  4DOS.HLP
6/07/2001   7:00    221,313  4DOS.TXT

```

(See **Switching Formats** below for information on changing the standard long filename format to allow room for file descriptions.)

On FAT volumes which do not support long file names, the default DIR format contains 5 columns: the file name, the file size in bytes, the date of the last write, the time of the last write, and the file's description. File names are listed in lower-case; directory names in upper case:

```

Volume in drive C is C - BOOTUP      Serial ...
Directory of  C:\4DOS700\*. *

.                <DIR>          5/25/01  17:46
..               <DIR>          5/25/01  17:46
4DOS.COM         254356    6/07/01   7:00  4DOS Program
4DOS.HLP         744092    6/07/01   7:00  4DOS Help ...
4DOS.TXT         221313    6/07/01   7:00  4DOS Doc ...

```

DIR's output is normally sorted by name, with directories listed first. You can change the sort order with the **/O** option. For example, these two commands sort the output by date — the first command lists the oldest file first; the second command lists the oldest file last:

```

[c:\] dir /o:d
[c:\] dir /o:-d

```

When displaying file descriptions, DIR wraps long lines to fit on the screen. DIR displays a maximum of 40 characters of text in each line of a description, unless your screen width allows a wider display. If you disable description wrapping with the **/R** option, the description is truncated at the right edge of the screen, and a right arrow [►] is added at the end of the line to alert you to the existence of additional description text.

Regardless of the volume type, DIR's default output is sorted. It displays directory names first, with "<DIR>" inserted instead of a file size, and then filenames. DIR assumes that sequences of digits should be sorted numerically (for example, the file *DRAW2* is listed before *DRAW03* because 2 is numerically smaller than 03), rather than strictly alphabetically (where *DRAW2* would come second because "2" is after "0" in alphanumeric order). You can change the sort order with

the **/O** option. When DIR displays file names in a multi-column format, it sorts file names horizontally unless you use the **/V** option to display vertically sorted output.

DIR's display can be modified in many ways to meet different needs. Most of the following sections describes the various ways you can change DIR's output format.

### ***Switching Formats***

On volumes which support long file names, you can force DIR to use a FAT-like format (file name first, followed by file information) with the **/Z** option. If necessary, DIR **/Z** truncates long file names on LFN drives, and adds a right arrow [►] to show that the name contains additional characters.

The standard LFN output format does not provide enough space to show descriptions along with file names. Therefore, if you wish to view file descriptions as part of the DIR listing on a volume which supports long file names, you **must** use the **/Z** option.

DIR will display the alternate, short file names for files with long file names if you use the **/X** option. Used alone, **/X** causes DIR to display names in 2 columns after the size, time, and date: one column for alternate or short file names and the other for long file names. If a file does not have a short or alternate name which is different from the long filename, the first filename column is empty

If you use **/X** and **/Z**, DIR will display the short or alternate file names in the FAT-style display format.

If you use the **/B** option, DIR displays just file names and omits the file size, time stamp, and description for each file, for example:

```
[c:\] dir w* /b
WINDOWS
WINNT
win311
WINALIAS
WINENV.BTM
.....
```

There are several ways to modify the display produced by **/B**. The **/F** option is similar to **/B**, but displays the full path and name of each file, instead of just its name. To view the same information for a directory

and its subdirectories use **/B /S** or **/F /S**. You can use **/B /X** to display the short name of each file, with no additional information. To display the short version of both the path and file name for each file, use **/F /X**. For example:

```
[c:\] dir /x/f/s *.pif
C:\MACH64\INSTALL.PIF
C:\PROGRA~1\WINZIP\WZ.PIF
C:\WINDOWS\DOSPRMPT.PIF
C:\WINDOWS\STARTM~1\APPS&U~1\INFOSE~1.PIF
C:\WINDOWS\STARTM~1\PROMPTS\4DOS.PIF
C:\WINDOWS\STARTM~1\PROMPTS\TOCP.PIF
C:\WINDOWS\STARTM~1\PROMPTS\SPECIAL\4DOS(R~1.PIF
.....
```

### ***Multiple Column Displays***

DIR has three options, **/2**, **/4**, and **/W**, that create multi-column displays.

The **/2** option creates a 2-column display. On drives which support long filenames, only the name of each file is displayed, with directory names placed in square brackets to distinguish them from file names. On drives which do not support long filenames, or when **/Z** or **/X** is used (see below), the display includes the short name, file size, and time stamp for each file.

The **/4** option is similar to **/2**, but displays directory information in 4 columns. On drives which do not support long filenames, or when **/Z** or **/X** is used (see below), the display shows the file name and the file size in kilobytes (KB) or megabytes (MB), with "<D>" in the size column for directories.

The **/W** option displays directory information in 5 or more columns, depending on your screen width. Each entry in a **DIR /W** display contains either the name of a file or the name of a directory. Directory names are placed in square brackets to distinguish them from file names.

If you use one of these options on a drive that supports long file names, and do not select an alternate display format with **/Z** or **/X**, the actual number of columns will be based on the longest name to be displayed and your screen width, and may be less than the number you requested (for example, you might see only three columns even though you used **/4**). If the longest name is too long to fit in on a single line the display

will be reduced to one column, and each name will be wrapped, with “extra” blank lines added so that each name takes the same number of lines.

On LFN drives you can use **/Z** with any of the multi-column options to create a traditional FAT-format display, with long names truncated to fit in the available space. If you use **/X**, the traditional FAT-format display is also used, but short names are displayed (rather than truncated long names). The following table summarizes the effects of different options on LFN drives:

Format	Display Columns		
	Normal	/2 or /4	/W
Normal	1 column, long names plus size, date, time	2 - 4 columns, long names only	Columns based on longest name, long names only
/Z	1 column, truncated long names plus size, date, time, description	2 - 4 columns, truncated long names plus other info	5+ columns, truncated long names only
/X	1 column, both names plus size, date, time	2 - 4 columns, short names plus other info	5+ columns, short names only
/Z /X	1 column, short names plus size, date, time, description	(Same as /X alone)	(Same as /X alone)

### ***Color-Coded Directories***

The DIR command can display each file's name and associated information in a different color, depending on the file's extension.

To choose the display colors, you must either use the SET command to create an environment variable called COLORDIR, use the “Colors” tab of the configuration dialog, or use the ColorDir directive in your *.INI* file. If you do not use the COLORDIR variable or the ColorDir directive, DIR will use the default screen colors for all files.

If you use both the COLORDIR variable and the ColorDir directive, the environment variable will override the settings in your *.INI* file. You



may find it useful to use the COLORDIR variable for experimenting, then to set permanent directory colors with the ColorDir directive.

The format for both the COLORDIR environment variable and the ColorDir directive in the .INI file is:

```
ext ... :ColorName; ...
```

where “ext” is a file extension (which may include wildcards) or one of the following file types:

DIRS	Directories
RDONLY	Read-only files
HIDDEN	Hidden files
SYSTEM	System files
ARCHIVE	Files modified since the last backup

and “ColorName” is any valid color name (see page 30 for information on color names).

Unlike most color specifications, the background portion of the color name may be omitted for directory colors. If you don't specify a background color, DIR will use the current screen background color.

For example, to display the .COM and .EXE files in red on the current background, the .C and .ASM files in bright cyan on the current background, and the read-only files in green on white (this should be entered on one line):

```
[c:\] set colordir=com exe:red; c asm:bright cyan;  
rdonly:green on white
```

Extended wildcards can be used in directory color specifications (see page 94 for more information on extended wildcards). For example, to display .BAK, .BAX, and .BAC files in red:

```
[c:\] set colordir=BA[KXC]:red
```

### ***Redirected Output***

The output of the DIR command, like that of most other internal commands, can be redirected to a file, printer, serial port, or other device. However, you may need to take certain DIR options into account when you redirect DIR's output.

DIR wraps both long file names and file descriptions at the width of your display. Its redirected output will also wrap at the screen width. Use the **/R** option if you wish to disable wrapping of long descriptions.

If you redirect a color-coded directory to a file, DIR will remove the color data as it sends the directory information to a file. It will usually do the same if you redirect output to a character device such as a printer or serial port. However, it is not always possible for DIR to tell whether or not a device is a character device. If you notice that non-colored lines are being sent to the output device and colored lines are appearing on your screen, you can use the **/D** option to temporarily disable color-coding when you redirect DIR's output.

To redirect DIR output to the clipboard, use **CLIP:** as the output device name, for example:

```
[c:\] dir *.exe > clip:
```

#### **4NT, TC** *FTP Usage*

You can display directories on FTP servers. The URL must be enclosed in double quotes so the forward slashes won't be interpreted as switches. For example:

```
[c:\] dir "ftp://jpsoft.com/data"
```

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see *Using FTP Servers* on page 113, and the IFTP command on page 382.

#### ❖ **Other Notes**

If you have selected a specific country code for your system, DIR will display the date in the format for that country. The default date format is U.S. (mm-dd-yy). The separator character in the file time will also be affected by the country code. Thousands and decimal separators in numeric displays are affected by the country code, and by the ThousandsChar and DecimalChar settings selected on the "Syntax" tab of the configuration dialog, or in the *.INI* file (see pages 220 and 227).

If you are using a disk compression program, you can use the **/C** switch to view the amount of compression achieved for each file. When you do,

the compression ratio is displayed instead of the file's description. You can also sort the display by compression ratios with the **/O:c** switch. Details for both switches are in the Options section, below.

**4DOS** DIR can handle directories of any size, limited only by available memory. Under 4DOS, each short filename requires 64 bytes of memory, plus the size of the description (if any). For example, a system with just 128K of free memory can display up to about 2,000 files per directory. Long filenames require additional space roughly equal to the length of the long names to be displayed. Additional space is also required when using DIR /S, particularly on LFN drives with long directory names and/or deeply-nested directories. Memory requirements for DIR are generally not a concern under 4NT and Take Command, because of the virtual memory available under these operating systems.

**4DOS** DOS networks with large server volumes (over 2 GB) may report incorrect free disk space values at the end of the DIR display. If this occurs, it is because the network software does not report the correct values to 4DOS.

**Options:** Options on the command line apply only to the filenames which follow the option, and options at the end of the line apply to the preceding filename only. This allows you to specify different options for different groups of files, yet retains compatibility with the traditional DIR command when a single filename is specified.

**/1:** Single column display – display the filename, size, date, and time; also displays the description on drives which do not support long filenames.. This is the default. If **/T** is used the attributes are displayed instead of the description; if **/C** or **/O:c** is used the compression ratio is displayed instead of the description. This option is most useful if you wish to override a default **/2**, **/4**, or **/W** setting stored in an alias.

**/2:** Two column display – display just the name (on LFN drives), or display the filename, size, date, and time on other drives. See **Multiple Column Displays** on page 309 for more details.

**/4:** Four column display – display just the name (on LFN drives); or display the filename and size, in K (kilobytes) or M (megabytes) on other drives, with files between 1 and 9.9 megabytes in size displayed in tenths (*i.e.*, "2.4M"). See **Multiple Column Displays** on page 309 for more details.

**4NT, TC** **/:** Display file stream names and sizes on NTFS volumes.

- ❖ **/A(tribute select)**: Display only those files that have the specified attribute(s) set. See **Attribute Switches** on page 111 for information on the attributes which can follow **/A**.
- ❖ **/B(are)**: Suppress the header and summary lines, and display file or subdirectory names only, in a single column. This option is most useful when you want to redirect a list of names to a file or another program. If you use **/B** with **/S**, DIR will show the full path of each file (the same display as **/F**) instead of simply its name and extension. If you use **/B** with **/X** on an LFN drive, DIR will display the short name of each file instead of the long name.

**/C(ompression)**: Display per-file and total compression ratio on compressed drives. The compression ratio is displayed instead of the file description or attributes. The ratio is left blank for directories and files with a length of 0 bytes, and for files on non-compressed drives. **/C** only works in single-column mode; it is ignored if **/2**, **/4**, or **/W** is used. The compression ratios will not be visible on LFN drives unless you use **/Z** to switch to the traditional short filename format.

Under 4DOS, only the compression programs distributed with MS-DOS and Windows 95 / 98 / ME (DRVSPACE and DBLSPACE) are supported. Under 4NT and Take Command, only compressed NTFS drives are supported.

The numerator of the displayed compression ratio is the amount of space which would be allocated to the file if the compression utility were not in use, based on the compressed drive's cluster size (usually 8K bytes). The denominator is the space actually allocated for the compressed file. For example, if a file is allocated 6,144 bytes when compressed, and would require 8,192 bytes if uncompressed, the displayed compression ratio would be 8,192 / 6,144, or 1.3 to 1.

**4DOS** Using **/CH** displays compression ratios like **/C**, but bases the calculation on the host drive's cluster size. This gives a more accurate picture of the space saved through compression than is given by **/C**. This option will occasionally display compression ratios slightly less than 1.0 for files which have actually expanded when stored on the compressed drive.

**4DOS** If **/CP** is used instead of **/C**, the compression is displayed as a percentage (*e.g.*, 33%) instead of a ratio (*e.g.*, 3 to 1). If **/CHP** is used

instead of **/CH**, the host compression is displayed as a percentage. The **/CHP** option must be entered as shown; you can **not** use **/CPH**.

- ❖ **/D**(isable color coding): Temporarily disable directory color coding. May be required when color-coded directories are used and DIR output is redirected to a character device like the printer (*e.g.*, PRN or LPT1) or serial port (*e.g.*, COM1 or COM2). **/D** is not required when DIR output is redirected to a file.

**/E**: Display filenames in upper case; also see SETDOS **/U** (page 477) and the UpperCase *.INI* file directive (page 228).

- ❖ **/F**(ull path): Display each filename with its drive letter and path in a single column, without other information. If you use **/F** with **/X** on a volume which supports long filenames, the “short” version of the entire path is displayed.

**/G**: Display the allocated disk space instead of the actual size of each file.

**/H**(ide dots): Suppress the display of the “.” and “..” directories.

**/I"text"**: Select filenames by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

The **/I** option may be used to select files even if descriptions are not displayed (for example, if **/2** is used). However, **/I** will be ignored if **/C** or **/O:c** is used.

**/J**(ustify names): Justify (align) filename extensions and display them in the traditional format. If on an LFN drive, you must also specify the **/X** and **/Z** options.

- ❖ **/K**: Suppress the header (disk and directory name) display.

**/L**(ower case): Display file and directory names in lower case; also see SETDOS **/U** (page 477) and the UpperCase *.INI* file directive (page 228).

- ❖ **/M**: Suppress the footer (file and byte count totals) display.

**4DOS**    **/N:** Reset the DIR options to the default values. This is useful when you want to display some files in one format, and then change back to the defaults for another set of files.

**4NT, TC** **/N:** Use the long filename display format, even if the files are stored on a volume which does not support long filenames. See also **/Z**.

**/O(rder):** Set the sorting order. You may use any combination of the following sorting options; if multiple options are used, the listing will be sorted with the first sort option as the primary key, the next as the secondary key, and so on:

- Reverse the sort order for the next option
- a** Sort names and extensions in standard ASCII order, rather than sorting numerically when digits are included in the name or extension.
- c** Sort by compression ratio (the least compressed file in the list will be displayed first). For single-column directory displays in the traditional short filename format, the compression ratios will be used as the basis of the sort and will also be displayed. For wider displays (**/2**, **/4**, and **/W**) and displays in LFN format, the compression ratios will be used to determine the order but will not be displayed. Under 4DOS, if **/O:c** is used with **/CH** or **/CHP** the sort will be based on the host-drive compression ratios. For information on supported compression systems see **/C** above.
- d** Sort by date and time (oldest first); for drives which support long filenames, also see **/T:acw**.
- e** Sort by extension.
- g** Group subdirectories first, then files.
- i** Sort by the file description (ignored if **/C** or **/O:c** is also used).
- n** Sort by filename (this is the default).
- r** Reverse the sort order for all options.
- s** Sort by size.
- u** Unsorted.

**/P(ause):** Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail on page 80.

**4DOS**    **/Q:** Display the file or directory owner.

**/R** (disable wRap): Forces long descriptions to be displayed on a single line, rather than wrapped onto two or more lines. Use **/R** when output is redirected to a character device, such as a serial port or the printer; or when you want descriptions truncated, rather than wrapped, in the on-screen display.

**/S**(ubdirectories): Display file information from the current directory and all of its accessible subdirectories. DIR will only display headers and summaries for those directories which contain files that match the filename(s), ranges, and attributes that you specify on the command line. In 4DOS, DIR processes only those subdirectories without the Hidden or System attributes. To view hidden or system subdirectories use **/A** along with **/S**. In 4NT and Take Command, DIR will display hidden subdirectories (for compatibility with CMD.EXE).

- ❖ **/T** (aTtribute display): Display the filenames and attributes. **/T** is ignored if **/C** or **/O:c** is also used. The attributes are displayed in the format RHSAD, where:

<b>R</b>	Read-only	<b>A</b>	Archive
<b>H</b>	Hidden	<b>D</b>	Subdirectory
<b>S</b>	System		

**4NT, TC** Under Windows 2000 and XP the following additional attributes will be displayed, in the format ENTPCOI:

<b>E</b>	Encrypted	<b>C</b>	Compressed
<b>N</b>	Normal	<b>O</b>	Offline
<b>T</b>	Temporary	<b>I</b>	Not content-indexed
<b>S</b>	Sparse file	<b>J</b>	Junction (reparse point)

On drives which support long file names, if you wish to add another option after **/T**, you must start the next option with a forward slash. If you don't, the command processor will interpret the **/T** as the **/T:acw** time display switch (see below) and the following character as a time selector. For example:

<code>[c:\] dir /tz</code>	incorrect, will display error
<code>[c:\] dir /t/z</code>	correct

**/T:acw** (Time display): Specify which of the date and time fields on a drive which supports long filenames should be displayed and used for sorting (see page 17):

- a** Last access date and time (access time will always be 00:00 on VFAT and FAT32 volumes).
- c** Creation date and time.
- w** Last write date and time (default).

See page 17 for more information on file dates and times on FAT, FAT32, VFAT, and NTFS drives.

**/U[n]** (sUmmary information): Only display the number of files, the total file size, and the total amount of disk space used. Information on individual files is not displayed. **/U1** will display summaries for each directory, but no total summary for each parent directory. **/U2** displays the grand total only.

**/V**(ertical sort): Display the filenames sorted vertically rather than horizontally (use with the **/2**, **/4** or **/W** options).

**/W**(ide): Display filenames only, horizontally across the screen. On drives which do not support long filenames, or when used with **/Z** or **/X**, **/W** displays as many columns as it can fit into the command processor window, using 16 characters in each column. Otherwise (*i.e.*, when long filenames are displayed) the number of columns depends on the width of the longest name in the listing. See **Multiple Column Displays** on page 309 for more details.

**/X**: Display both the short name (8-character name plus 3-character extension) and the long name of each file on an LFN drive. In normal single-column output the short name is displayed first, followed by the long name. The short name column is left blank if the short name and long name are the same. **/X** also selects short filenames in the **/2**, **/4**, **/B**, **/W**, and **/Z** displays, and short file and path names in the **/F** display.

**/Z**: Display filenames on an LFN drive in the old-style format, with the filename on the left and the description on the right. Long names will be truncated to 12 characters unless **/X** is also used; if the name is longer than 12 characters, it will be followed by a right arrow [►] to show that one or more characters have been truncated.



## **DIRHISTORY**

(New)

**Purpose:** Display, add to, clear, or read the directory history list.

**Format:** DIRHISTORY [/A *directory* /F /P /R *file...*]

***file*:** One or more files containing entries to be added to the directory history.

***directory*:** The name of a directory to be added to the directory history.

***filename*:** The name of a file containing entries to be added to the directory history.

/A(dd)

/P(ause)

/F(ree)

/R(ead)

See also: HISTORY.

**Usage:** Every time you change to a new directory or drive, the command processor records the current directory in an internal directory history list. See page 82 for information on directory history window, which allows you to use the list to return to a previous directory. See page 81 for general information on directory navigation.

The DIRHISTORY command lets you view and manipulate the directory history list directly. If no parameters are entered, DIRHISTORY will display the current directory history list:

```
[c:\] dirhistory
```

With the options explained below, you can clear the list, add new directories to the list without changing to them, save the list in a file, or read a new list from a file.

The number of directories saved in the directory history list depends on the length of each directory name. The list size can be specified at startup from 256 to 2048 characters in 4DOS, or 256 to 32767 characters in 4NT and Take Command by using the DirHistory directive in the .INI file (see page 210) or the configuration dialog in 4NT and Take Command. The default size is 256 characters in 4DOS and 1024 characters in 4NT and Take Command

Your directory history list can be stored either locally (a separate history list for each copy of the command processor) or globally (all

copies of the command processor share the same list). For details see the discussion of local and global directory history lists beginning on page 75.

You can save the directory history list by redirecting the output of **DIRHISTORY** to a file. This example saves the history to a file called **DIRHIST** and reads it back again.

```
[c:\] dirhistory > dirhist
.....
[c:\] dirhistory /r dirhist
```

Because the directory history stores each name only once, you don't have to delete its contents before reading back the file unless you want to delete the directories that were visited by the intervening commands.

If you need to save your directory history at the end of each day's work, you might use the first of these commands in your **4START.BTM** / **TCSTART.BTM** or other startup file, and the second in **4EXIT.BTM** / **TCEXIT.BTM**:

```
if exist c:\dirhist dirhistory /r c:\dirhist

dirhistory > c:\dirhist
```

This restores the previous history list if it exists, and saves the history when the command processor exits.

Options: ❖ **A(dd)**: Add a directory to the directory history list.

**/F(ree)**: Erase all entries in the directory history list.

**/P(rompt)**: Wait for a key after displaying each page of the list. Your options at the prompt are explained on page 80.

❖ **/R(ead)**: Read the directory history from the specified file and append it to the list currently held in memory.

**DIRS**

(New)

**Purpose:** Display the current directory stack.

**Format:** **DIRS**

See also: **PUSHD**, **POPD**, and Directory Navigation on page 81.

**Usage:** The **PUSHD** command adds the current default drive and directory to the directory stack, a list that the command processor maintains in memory. The **POPD** command removes the top entry of the directory stack and makes that drive and directory the new default. The **DIRS** command displays the contents of the directory stack, with the most recent entries on top (*i.e.*, the next **POPD** will retrieve the first entry that **DIRS** displays).

For example, to change directories and then display the directory stack:

```
[c:\] pushd c:\database
[c:\database] pushd d:\wordp\memos
[d:\wordp\memos] dirs
c:\database
c:\
```

The directory stack holds 511 characters, enough for 20 to 40 typical drive and directory entries.

**DO**

(New)

**Purpose:** Create loops in batch files.

**Format:** DO [*n* | FOREVER]

or

DO varname = start TO end [BY *n*]

or

DO [WHILE | UNTIL] *condition*

or

DO *varname* IN [/A[:][-]rhsad /I"text" /L] [@]*fileset*

commands

**[ITERATE]**

**[LEAVE]**

commands

**ENDDO**

***varname*:** The environment variable that will hold the loop counter, filename, or line from a file.

***n*, *start*, *end*:** Integers between 0 and 2,147,483,647 inclusive, or an internal variable or variable function that evaluates to such a value.

***condition*:** A test to determine if the loop should be executed.

***fileset*:** A filename or list of filenames.

***commands*:** One or more commands to execute each time through the loop. If you use multiple commands, they must be separated by command separators or be placed on separate lines.

**/A:** (Attribute select)

**/L** (text arguments)

**/I** (match descriptions)

**Files:** Supports extended wildcards, ranges, and include lists for the ***fileset*** (see pages 94 - 104).

Usage: DO can only be used in batch files.

DO can be used to create 4 different kinds of loops. The first, introduced by **DO n**, is a counted loop. The batch file lines between DO and ENDDO are repeated **n** times. For example:

```
do 5
    beep
enddo
```

You can also specify “forever” for **n** if you wish to create an endless loop (you can use LEAVE or GOTO to exit such a loop; see below for details).

The second type of loop is similar to a “for loop” in programming languages like BASIC. DO creates an environment variable, **varname**, and sets it equal to the value **start** (if **varname** already exists in the environment, it will be overwritten). DO then begins the loop process by comparing the value of **varname** with the value of **end**. If **varname** is less than or equal to **end**, DO executes the batch file lines up to the ENDDO. Next, DO adds 1 to the value of **varname**, or adds the value **n** if BY **n** is specified, and repeats the compare and execute process until **varname** is greater than **end**. This example displays the even numbers from 2 through 20:

```
do i = 2 to 20 by 2
    echo %i
enddo
```

DO can also count down, rather than up. If **n** is negative, **varname** will decrease by **n** with each loop, and the loop will stop when **varname** is less than **end**. For example, to display the even numbers from 2 through 20 in reverse order, replace the first line of the example above with:

```
do i = 20 to 2 by -2
```

The third type of loop is called a “while loop” or “until loop.” DO evaluates the **condition**, which can be any of the tests supported by the IF command (see page 372), and executes the lines between DO and ENDDO as long as the condition is true. The loop ends when the condition becomes false.

WHILE tests the condition at the start of the loop. Therefore, if the condition is false when the loop starts, the statements within the loop

will never be executed, and the batch file will continue with the statement after the ENDDO.

UNTIL tests the condition at the end of the loop. Therefore, the statements within the loop will always be executed at least once.

The fourth type of loop executes the lines between DO and ENDDO once for every filename in the *fileset*, or once for each argument (if you use the /L option). For example:

```
do x in *.txt
```

will execute the loop once for every *.TXT* file in the current directory; each time through the loop the variable *x* will be set to the name of the next file that matches the file specification.

- ❖ If, between DO and ENDDO, you create a new file that *could* be included in the list of files, it may or may not appear in an iteration of the DO loop. Whether the new file appears depends on its physical location in the directory structure, a condition over which the command processor has no control.

You can also execute the loop once for each line of text in a file by placing an *@* in front of the file name (in this case only a single filename can be specified). If you have a file called *DRIVES.TXT* that contains a list of drives on your computer, one drive name per line, you can execute the loop once for each drive this way:

```
do x in @drives.txt
```

To use date, time, size, or file exclusion ranges for the *fileset* place them just before the filename(s), for example:

```
do x in /[d1-1-2001,1-31-2001] *.txt
```

To execute the loop once for each line of text in the clipboard, use **CLIP:** as the file name (e.g. DO X IN @CLIP:). CLIP: will not return any data unless the clipboard contains text. See page 88 for additional information on CLIP:, including details on when you can use the clipboard under 4DOS.

Two special commands, ITERATE and LEAVE, can be used inside a DO / ENDDO loop. ITERATE ignores the remaining lines inside the loop and returns to the beginning of loop for another iteration (unless DO determines that the loop is finished). LEAVE exits from the

current DO loop and continues with the line following ENDDO. Both ITERATE and LEAVE are most often used in an IF or IFF command (see pages 372 and 380):

```
do while "%var" != "%val1"
    ...
    if "%var" == "%val2" leave
enddo
```

You can nest DO loops up to 15 levels deep.

The DO and ENDDO commands must be on separate lines, and cannot be placed within a command group (see page 117), or on the same line as other commands (this is the reason DO cannot be used in aliases). However, commands within the DO loop can use command groups or the command separator in the normal way.

**4DOS !** If you receive a stack overflow error when using DO in complex, nested command sequences, see the notes under the StackSize directive on page 238.

**! ❖** You can exit from all DO / ENDDO loops by using GOTO to a line past the last ENDDO. However, be sure to read the cautionary notes about GOTO and DO under the GOTO command (page 365) before using a GOTO in any other way inside any DO loop.

You cannot use RETURN to return from a GOSUB while inside a DO loop.

**Option:** **/A**(ttribute select): Find only those files that have the specified attribute(s) set. See **Attribute Switches** on page 111 for information on the attributes which can follow **/A**.

**/I"text"**: Select filenames by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**/L**: The arguments in a "DO x IN ..." statement are text, not filenames. The arguments will be assigned (from left to right) to the DO variable on each pass through the loop.

**DRAWBOX**

(New)

**Purpose:** Draw a box on the screen.

**Format:** `DRAWBOX ulrow ulcol lrow lcol style [BRiGht] [BLInk] fg  
ON [BRiGht] bg [FILl [BRiGht] bgfill] [ZOOm] [SHAdow]`

***ulrow*:** Row for upper left corner

***ulcol*:** Column for upper left corner

***lrow*:** Row for lower right corner

***lcol*:** Column for lower right corner

***style*:** Box drawing style:

- |          |  |
|----------|--|
| <b>0</b> | No lines (box is drawn with blanks)            |
| <b>1</b> | Single line                                    |
| <b>2</b> | Double line                                    |
| <b>3</b> | Single line on top and bottom, double on sides |
| <b>4</b> | Double line on top and bottom, single on sides |

***fg*:** Foreground character color (**BLI** only valid in full-screen 4DOS)

***bg*:** Background character color

***bgfill*:** Background fill color (for the inside of the box)

See also: **DRAWHLINE** and **DRAWVLINE**.

**Usage:** **DRAWBOX** is useful for creating attractive screen displays in batch files.

For example, to draw a box around the the edge of an 80x25 window with bright white lines on a blue background:

```
drawbox 0 0 24 79 1 bri whi on blu fill blu
```

See page 30 for details about colors and color names, and notes on the use of bright background colors.

If you use **ZOOM**, the box appears to grow in steps to its final size. The speed of the zoom operation depends on the speed of your computer and video system.

If you use **SHADOW**, a drop shadow is created by changing the characters in the row under the box and the 2 columns to the right of



the box to normal intensity text with a black background (this will make characters displayed in black disappear entirely).

The **row** and **column** values are zero-based, so on a 25 line by 80 column display, valid **rows** are 0 - 24 and valid **columns** are 0 - 79. In Take Command, the maximum **row** value is determined by the current height of the Take Command window, and the maximum **column** value is determined by the current virtual screen width (see page 39 for more information).

If **ulrow** is set to 999, **lrow** is assumed to be the desired height, and the box will be centered vertically. If **ulcol** is set to 999, **lcol** is assumed to be the desired width, and the box will be centered horizontally.

DRAWBOX checks for valid row and column values, and displays a "Usage" error message if any values are out of range.

Unlike DRAWHLINE and DRAWVLINE, DRAWBOX does **not** automatically connect boxes to existing lines on the screen with the proper connector characters. If you want to draw lines inside a box and have the proper connectors drawn automatically, draw the box first, then use DRAWHLINE and DRAWVLINE to draw the lines.

- 4DOS**     ❖DRAWBOX normally writes text directly to the screen. If you have an unusual display adapter which does not support direct video output, see the OutputBIOS directive on page 237.
- TC**        DRAWBOX, DRAWHLINE, and DRAWVLINE use the standard line and box drawing characters in the U.S. English extended ASCII character set. If your system is configured for a different country or language, or if you use a font which does not include these line drawing characters, the box or lines may not appear on your screen as you expect. They will only appear correctly if you have configured Take Command to use a font such as Terminal, which contains standard extended ASCII characters.

### **DRAWHLINE**

(New)

**Purpose:** Draw a horizontal line on the screen.

**Format:** **DRAWHLINE** *row column len style* [BRiGht] [BLInk]fg ON [BRiGht] bg

***row***: Starting row

***column***: Starting column

***len***: Length of line

***style***: Line drawing style: **1** for single line, **2** for double line.

***fg***: Foreground character color (**BLI** only valid in full-screen 4DOS)

***bg***: Background character color

See also: DRAWBOX and DRAWVLINE.

**Usage:** DRAWHLINE is useful for creating attractive screen displays in batch files. It detects other lines and boxes on the display, and creates the appropriate connector characters when possible (not all types of lines can be connected with the available characters).

For example, the following command draws a double line along the top row of the display with green characters on a blue background:

```
drawhline 0 0 80 2 green on blue
```

The ***row*** and ***column*** values are zero-based, so on a 25 line by 80 column display, valid ***rows*** are 0 - 24 and valid ***columns*** are 0 - 79. If either value is out of range, DRAWHLINE displays a “Usage” error message. In Take Command, the maximum ***row*** value is determined by the current height of the Take Command window, and the maximum ***column*** value is determined by the current virtual screen width (see page 39 for more information).

If ***row*** is set to 999, the line will be centered vertically. If ***column*** is set to 999, the line will be centered horizontally.

See page 30 for details about colors and color names.

Be sure to read the notes on page 327, at the end of the DRAWBOX command, for important information on non-standard video adapters (in 4DOS), and the way your country configuration and default font can affect the appearance of lines (in Take Command).

**DRAWVLINE**

(New)

**Purpose:** Draw a vertical line on the screen.

**Format:** DRAWVLINE *row column len style*[BRIght][BLInk] *fg* ON [BRIght] *bg*

***row*:** Starting row

***column*:** Starting column

***len*:** Length of line

***style*:** Line drawing style: **1** for single line, **2** for double line.

***fg*:** Foreground character color (**BLI** only valid in full-screen 4DOS)

***bg*:** Background character color

See also: DRAWBOX and DRAWHLINE.

**Usage:** DRAWVLINE is useful for creating attractive screen displays in batch files. It detects other lines and boxes on the display, and creates the appropriate connector characters when possible (not all types of lines can be connected with the available characters).

For example, to draw a double width line along the left margin of the display with bright red characters on a black background:

```
drawvline 0 0 25 2 bright red on black
```

The ***row*** and ***column*** values are zero-based, so on a 25 line by 80 column display, valid ***rows*** are 0 - 24 and valid ***columns*** are 0 - 79. If either value is out of range, DRAWVLINE displays a “Usage” error message. In Take Command, the maximum ***row*** value is determined by the current height of the Take Command window, and the maximum ***column*** value is determined by the current virtual screen width (see page 39 for more information).

If ***row*** is set to 999, the line will be centered vertically. If ***column*** is set to 999, the line will be centered horizontally.

See page 30 for details about colors and color names.

Be sure to read the notes on page 327, at the end of the DRAWBOX command, for important information on non-standard video adapters (in 4DOS), and the way your country configuration and default font can affect the appearance of lines (in Take Command).

## **ECHO and ECHOERR**

(Enhanced / New)

**Purpose:** Display a message, enable or disable batch file or command-line echoing, or display the echo status.

**Format:** ECHO [ON | OFF | *message*]

ECHOERR *message*

***message*:** Text to display.

See also: ECHOS / ECHOSERR, SCREEN, SCRPUT, SETDOS and TEXT.

**Usage:** 4DOS, 4NT, and Take Command have a separate echo capability for batch files and for the command line. The command line ECHO state is independent of the batch file ECHO state; changing ECHO in a batch file has no effect on the display at the command prompt, and vice versa.

To see the current echo state, use the ECHO command with no arguments. This displays either the batch file or command-line echo state, depending on where the ECHO command is performed.

In a batch file, if you turn ECHO on, each line of the file is displayed before it is executed. If you turn ECHO off, each line is executed without being displayed. ECHO can also be used in a batch file to display a message on the screen. Regardless of the ECHO state, a batch file line that begins with the [@] character will not be displayed. To turn off batch file echoing, without displaying the ECHO command, use this line:

```
@echo off
```

ECHO commands in a batch file will send messages to the screen while the batch file executes, even if ECHO is set OFF. For example, this line will display a message in a batch file:

```
echo Processing your print files...
```

Leading and trailing spaces in the ***message*** are normally ignored. To display them, use backquotes. For example:

```
echo ` ` `This text is indented 3 spaces
```

If you want to echo a blank line from within a batch file, enter:

echo.

You cannot use the command separator character ([^] in 4DOS or [&] in 4NT and Take Command) or the redirection symbols (| > <) in an ECHO message, unless you enclose them in quotes (see page 199) or precede them with the escape character (see page 119).

ECHO defaults to ON in batch files. The current ECHO state is inherited by called batch files. You can change the default setting to ECHO OFF with the SETDOS /V0 command, on the “Startup” tab of the configuration dialog, or with the BatchEcho directive in the *.INI* file (see page 218).

If you turn the command-line ECHO on, each command will be displayed before it is executed. This will let you see the command line after expansion of all aliases and variables. The command-line ECHO is most useful when you are learning how to use advanced features. This example will turn command-line echoing on:

```
[c:\] echo on
```

ECHO defaults to OFF at the command line.

ECHOERR acts like ECHO but sends its output to the standard error device STDERR (usually the screen) instead of the standard output device. If the standard output of a batch file is redirected to a file or another device with >, ECHOERR will still generate a screen message. See page 88 for more information about the standard output and standard error devices and redirection.

## **ECHOS and ECHOSERR**

(New)

**Purpose:** Display a message without a trailing carriage return and line feed.

**Format:** ECHOS message

ECHOSERR *message*

See also: ECHO / ECHOERR, SCREEN, SCRPUT, TEXT, and VSCRPUT.

**Usage:** ECHOS is useful for text output when you don't want to add a carriage return / linefeed pair at the end of the line. For example, you can use ECHOS when you need to redirect control sequences to your printer; this example sends the sequence **Esc P** to the printer on LPT1 (%= is translated to the default escape character, and %=e to an ASCII Esc; see pages 119 and 156 for additional details):

```
[c:\] echos %=eP > lpt1:
```

You cannot use the command separator character ([^] in 4DOS or [&] in 4NT and Take Command) or the redirection symbols [|><] in an ECHOS message, unless you enclose them in quotes (see page 199) or precede them with the escape character (see page 119).

- ❖ ECHOS does not translate or modify the message text. For example, carriage return characters are not translated to CR/LF pairs. ECHOS sends only the characters you enter (after escape character and back-quote processing). The only character you cannot put into an ECHOS message is the NUL character (ASCII 0).

ECHOSERR acts like ECHOS but sends its output to the standard error device (usually the screen) instead of the standard output device. If the standard output of a batch file is redirected to a file or another device with >, ECHOSERR will still generate a screen message. See page 88 for more information about the standard output and standard error devices and redirection.

## **ENDLOCAL**

(New)

**Purpose:** Restore the saved disk drive, directory, environment, alias list, and special characters.

**Format:** ENDLOCAL [exportvar ...]

See also: SETLOCAL.

**Usage:** The SETLOCAL command in a batch file saves the current disk drive, default directory, all environment variables, the alias list, and the command separator, escape character, parameter character, decimal separator, and thousands separator. ENDLOCAL restores everything that was saved by the previous SETLOCAL command. For an example, see SETLOCAL on page 480.

SETLOCAL and ENDLOCAL can be nested up to 16 levels (4NT and Take Command) or 8 levels (4DOS) deep. You cannot use SETLOCAL and ENDLOCAL in an alias or at the command line.

An ENDLOCAL is performed automatically at the end of a batch file if you forget to do so, or when you invoke one batch file from another without using CALL.

You can “export” environment variables from inside a SETLOCAL by specifying the variable names to be preserved following the ENDLOCAL. For example:

```
setlocal
set test=abcd
endlocal test
```

**ESET**

(New)

**Purpose:** Edit environment variables and aliases.

**Format:** ESET [/A /D /F /M /S /U /V] *variable name...*

***variable name:*** The name of an environment variable or alias to edit.

/A(lias)	/S(ystem variable)
/D(efault environment)	/U(ser variable)
/F(unction)	/V(olatile)
/M(aster environment)	

See also: ALIAS, UNALIAS, SET, and UNSET.

**Usage:** ESET allows you to edit environment variables, aliases, and user-defined functions using line editing commands (see page 58 for information on line editing).

For example, to edit the executable file search path:

```
[c:\] eset path
path=c:\;c:\dos;c:\util
```

To create and then edit an alias:

```
[c:\] alias d = dir /d/j/p
[c:\] eset d
d=dir /d/j/p
```

ESET will search for environment variables first, then aliases. If you have a variable and an alias with the same name, ESET will edit the variable and ignore the alias unless you use the /A option.

The total length of the name and value combined is limited by the maximum line length of the command processor: 511 characters in 4DOS, or 2047 characters in 4NT and Take Command. If you use special techniques to create a longer environment variable, ESET will edit it provided it contains no more than 511 characters of text in 4DOS, or 4,095 characters in 4NT and Take Command.

**4NT, TC** If you have enabled global aliases (see page 259), any changes made to an alias with ESET will immediately affect all other copies of the command processor which are using the same alias list.



- Option:**     **/A(lias):** Edit the named alias even if an environment variable of the same name exists. If you have an alias and an environment variable with the same name, you must use this switch to edit the alias.
- 4NT, TC**   **/D(efault environment):** Edit a “default” variable in the registry (HKU\DEFAULT\Environment).
- /F(unction):** Edit the named user-defined function.
- 4DOS**       **/M(aster environment):** Edit an environment variable in the master environment rather than the local environment. This option is only useful from a secondary command shell (for example, when an application has “shelled to DOS”). **/M** only works for environment variables; it cannot be used to edit the primary shell's aliases.
- 4NT, TC**   **/S(ystem):** Edit a “system” variable in the registry (HKLM\System\CurrentControlSet\Control\Session Manager\Environment).
- 4NT, TC**   **/U(ser):** Edit a “user” variable in the registry (HKCU\Environment).
- 4NT, TC**   **/V(olatile):** Edit a “volatile” variable in the registry (HKCU\Volatile Environment).

**EVENTLOG** [4NT, TC]

(New)

**Purpose:** Write a string to the Windows NT / 2000 / XP event log.

**Format:** EVENTLOG [/E /I /Stext /W] *message*

***message:*** The text to write.

**/E**(rror)

**/S**(ource)

**/I**(nformational)

**/W**(arning)

See also: HISTORY and LOG.

**Usage:** EVENTLOG posts messages to the Windows NT / 2000 / XP application event log. Each message can be a maximum of 2047 characters long. You cannot use the command separator character ([&]) or the redirection symbols (| > <) in an EVENTLOG message, unless you enclose the message in quotes (see page 199) or precede the special characters with the escape character (see page 119).

By default, the text written with EVENTLOG is stored in the event log as informational messages. You can store warning and error messages by using the /W and /E switches.

Messages in the log can be reviewed with the Windows NT / 2000 / XP Event Log viewer.

EVENTLOG must be able to create a registry key the first time it is run. If you do not have proper registry permissions when you run EVENTLOG for the first time, and the key cannot be created, EVENTLOG will fail, and display an error. If this occurs, run EVENTLOG once while logged in as Administrator (to create the key) , then run it from your normal user account.

**Options:** **/E**(rror): Store the message as an error entry in the event log.

**/I**(nformational): Store the message as an informational entry in the event log. This is the default if no switch is used.

**/S**(ource): Specify the eventlog entry source. (If the source contains whitespace, it must be double-quoted). For example:

eventlog /sCompiling /I Your message here.

**/W**(arning): Store the message as a warning entry in the event log.

## **EXCEPT**

(New)

**Purpose:** Perform a command on all available files except those specified.

**Format:** EXCEPT [/I"text"] (*@file*) (*file*) *command*

***file*:** The file or files to exclude from the command.

***@file*:** A text file containing the names of the files to exclude, one per line (see page 109 for details).

***command*:** The command to execute, including all appropriate arguments and switches.

**/I** (match description)

See also: ATTRIB, and File Exclusion Ranges (page 102).

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104). Ranges **must** appear immediately after the EXCEPT keyword. Use extended wildcards with caution on LFN volumes; see page 105 for details.

**Usage:** EXCEPT provides a means of executing a command on a group of files and / or subdirectories, and excluding a subgroup from the operation. The ***command*** can be an internal command or alias, an external command, or a batch file.

File exclusion ranges (see page 102) provide a faster and more flexible method of excluding files from internal commands, and do not need to manipulate file attributes, as EXCEPT does. However, exclusion ranges can only be used with internal commands; you must use EXCEPT for external commands.

You may use wildcards to specify the files to exclude from the command. The first example erases all the files in the current directory except those beginning with *MEMO*, and those whose extension is *.WKS*. The second copies all files and subdirectories on drive C to drive D, except those in *C:\MSC* and *C:\DOS*:

```
[c:\] except (memo*.* *.wks) erase *.*  
[c:\] except (c:\msc c:\dos) copy c:\*.* d:\ /s
```

When you use EXCEPT on an LFN drive, you must quote any file names inside the parentheses which contain white space or special

characters. See page 15 for additional details. For example, to copy all files except those in the “*Program Files*” directory to drive E:\:

```
[c:\] except ("Program Files") copy /s *.* e:\
```

EXCEPT will assume that the files to be excluded are in the current directory, unless another directory is specified explicitly.

- ! EXCEPT prevents operations on the specified file(s) by setting the hidden attribute, performing the command, and then clearing the hidden attribute. If the command is aborted in an unusual way, you may need to use the ATTRIB command to remove the hidden attribute from the file(s).
- ! EXCEPT will not work with programs or commands that ignore the hidden attribute or which work explicitly with hidden files, including /A:H, DEL /Z, and the /H (process hidden files) switch available in some 4DOS, 4NT, and Take Command commands.
- ❖ Date, time, and size ranges can be used immediately after the word EXCEPT to further qualify which files should be excluded from the **command**. If the **command** is an internal command that supports ranges, an independent range can also be used in the **command** itself. You can also use a file exclusion range within the EXCEPT command; however, this will select files to be **excluded** from EXCEPT, and therefore **included** in execution of the **command**.
- ❖ You can use command grouping (see page 117) to execute multiple **commands** with a single EXCEPT. For example, the following command copies all files in the current directory whose extensions begin with .DA, except the .DAT files, to the D:\SAVE directory, then changes the first two characters of the extension of the copied files to .SA. This example should be entered on one line:

```
[c:\data] except (*.dat) (copy *.da* d:\save &  
ren *.da* *.sa*)
```

If you use filename completion (see page 67) to enter the filenames inside the parentheses, type a space after the open parenthesis. Otherwise, the command-line editor will treat the open parenthesis as the first character of the filename.

**Options:** /I"text": Select files by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the /I immediately,

with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**EXIT**

(Enhanced)

**Purpose:** Exit the current command processor session.

**Format:** EXIT [/B] [*value*]

***value*:** The numeric exit code to return.

**/B** (exit from batch file)

**Usage:** EXIT terminates the current copy of the command processor. Use it to return to an application when you have “shelled out” to work at the prompt, or to end a command-line session under Windows.

To close the session, or to return to the application that started the command processor, type:

```
[c:\] exit
```

If you specify a ***value***, EXIT will return that value to the program that started the command processor. For example:

```
[c:\] exit 255
```

The ***value*** is a number you can use to inform the program of some result, such as the success or failure of a batch file. It can range from 0 - 255 in 4DOS, or 0 - 4,294,967,295 in 4NT and Take Command.

**4DOS** ❖You cannot EXIT from the primary 4DOS shell under DOS. If EXIT does not seem to have any effect, you are probably in the primary shell.

**Options:** **/B:** Exit the current batch file in Windows 2000 and XP, rather than the shell. This switch is for compatibility with *CMD.EXE* in

**4NT, TC** Windows 2000 and XP. The CANCEL and QUIT commands are generally a more flexible way to exit batch files.

**FFIND**

(New)

**Purpose:** Search for files by name or contents.

**Format:** FFIND [/A[:][-]rhsad] /B /C /D[*list*] /E /F /I /I"text" /K /L /M /N  
/O[:][-]acdeginrsu] /P /R /S /T | X]"xx" /U /V /Y] file...

***list*:** A list of disk drive letters (without colons).

***file*:** The file, directory, or list of files or directories to display.

/A(tribute select)	/N(ot)
/B(are)	/O(rder)
/C(ase sensitive)	/P(ause)
/D(rive)	/R(everse)
/E (upper case display)	/S(ubdirectories)
/F (stop after match)	/T"xx" (text search string)
/I(gnore wildcards)	/U (summary only)
/I"text" (match description)	/V(erbose)
/K (no headers)	/X]"xx"] (hex display / search string)
/L(ine numbers)	/Y (prompt to stop after match)
/M (no footers)	

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104).

**Internet:** Can be used with FTP servers.

**Usage:** FFIND is a flexible search command that looks for files based on their names and their contents. Depending on the options you choose, FFIND can display filenames, matching text, or a combination of both in a variety of formats.

**TC** *Most of the functions provided by FFIND are also available in the Take Command Find Files / Text dialog, accessible from the Utilities menu. You can use the FFIND command, the dialog, or both, depending on your needs.*

If you want to search for files by name, FFIND works much like the DIR command. For example, to generate a list of all the .BTM files in the current directory, you could use the command:

```
[c:\] ffind *.btm
```

The output from this command is a list of full pathnames, followed by the number of files found.

If you want to limit the output to a list of *.BTM* files which contain the string *color*, you could use this command instead:

```
[c:\] ffind /t"color" *.btm
```

The output from this command is a list of files that contain the string *color* along with the first line in each file that contains that string. By default, FFIND uses a case-insensitive search, so the command above will include files that contain *COLOR*, *Color*, *color*, or any other combination of upper-case and lower-case letters.

If you would rather see the **last** line of each file that contains the search string, use the **/R** option, which forces FFIND to search from the end of each file to the beginning. This option will also speed up searches somewhat if you are looking for text that will normally be at the end of a file, such as a signature line:

```
[c:\] ffind /r /t"Sincerely," *.txt
```

You can use extended wildcards in the search string to increase the flexibility of FFIND's search. For example, the following command will find *.TXT* files which contain either the string *June* or *July*. The **/C** option makes the search case-sensitive:

```
[c:\] ffind /c/t"Ju[nl][ey]" *.txt
```

If you want to search for text that contains wildcard characters (**\***, **?**, **[**, or **]**), you can use the **/I** option to force FFIND to interpret these as normal characters instead of wildcards. The following command finds all *.TXT* files that contain a question mark:

```
[c:\] ffind /i/t"?" *.txt
```

You may need to search for data that cannot be represented by ASCII characters. You can use FFIND's **/X** option to represent the search string in hexadecimal format (this option also changes the output to show hexadecimal offsets rather than text lines). With **/X**, the search must be represented by pairs of hexadecimal digits separated by spaces; a search of this type is always case-sensitive (in the example below, 41 63 65 is the hex code for "Ace"):

```
[c:\] ffind /x"41 63 65" *.txt
```



You can use FFIND's other options to further specify the files for which you are searching and to modify the way in which the output is displayed.

When you use FFIND on an LFN drive, you must quote any file names which contain white space or special characters. See page 15 for additional details.

**4NT, TC** FFIND can also find files on FTP servers. The URL must be enclosed in double quotes so the forward slashes won't be interpreted as switches. For example:

```
[c:\] ffind /t"4nt" "ftp://jpsoft.com/index"
```

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see Using FTP Servers on page 113, and the IFTP command on page 382.

Note that searching for text in files on FTP servers (as in the command above) will be slow as the data from each file searched must be retrieved from the server and transferred to your computer to be checked for the search string.

**Options:** **/A(tribute select):** Find only those files that have the specified attribute(s) set. See **Attribute Switches** on page 111 for information on the attributes which can follow **/A**.

**/B(are):** Display file names only and omit the text that matches the search. This option is only useful in combination with **/T** or **/X**, which normally force FFIND to display file names and matching text.

**/C(ase sensitive):** Perform a case-sensitive search. This option is only valid with **/T**, which defaults to a case-insensitive search. It is not needed with a **/X** hexadecimal search, which is always case-sensitive.

**/D(rive):** Search all files on one or more drives. If you use **/D** without a list of drives, FFIND will search the drives specified in the list of files. If no drive letters are listed, FFIND will search all of the current drive. You can include a list of drives or a range of drives to search as part of the **/D** option. For example, to search drives C:, D:, E:, and G:, you can use either of these commands:

```
[c:\] ffind /dcdeg ...  
[c:\] ffind /dc-eg ...
```

- ❖ Drive letters listed after **/D** will be ignored when processing *file* names which also include a drive letter. For example, to display all the *.BTM* files on C: and E:, but only the *.BAT* files on D:

```
[c:\] ffind /s /dce *.btm d:\*.bat
```

**/E:** Display filenames in the traditional upper case; see SETDOS /U (page 477) and the UpperCase *.INI* file directive (page 228).

**/F:** Stops the search after the first match.

**/I**(gnore wildcards): Suppresses the recognition of wildcard characters in the search text. (Only meaningful when used in conjunction with the **/T"text"** option.) This is useful if you need to search for characters that would normally be interpreted as wildcards: \*, ?, [, and ].

**/I"text"**: Select filenames by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**/K** (No headers): Suppress the display of the header or filename for each matching text line.

**/L**(ine numbers): Include the line number for each text line displayed. FFIND numbers lines beginning with 1, unless ListRowStart is set to 0 in the *.INI* file. A new line is counted for every CR or LF character (FFIND determines automatically which character is used for line breaks in each file), or when line length reaches 511 characters, whichever comes first.

**/M** (No footers): Suppress the footer (the number of files and number of matches) at the end of FFIND's display.

**/N**(ot): Reverse the meaning of the search. Setting **/N** will also set **/B**, i.e. searches are on a file-by-file basis.

**/O**(rder): Set the sort order for the files that FFIND searches. You can use any combination of the following sorting options; if multiple options are used, the listing will be sorted with the first sort option as the primary key, the next as the secondary key, and so on:

- Reverse the sort order for the next option
- a** Sort names and extensions in standard ASCII order, rather than sorting numerically when digits are included in the name or extension.
- c** Sort by compression ratio (the least compressed file in the list will be displayed first).
- d** Sort by date and time (oldest first); for drives which support long file names.
- e** Sort by extension.
- g** Group subdirectories first, then files.
- i** Sort by the file description (ignored if **/O:c** is also used).
- n** Sort by filename (this is the default).
- r** Reverse the sort order for all options.
- s** Sort by size.
- u** Unsorted.

**/P(ause)**: Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail on page 80.

**/R(everse)**: Only meaningful when used in conjunction with the **/T "text"** or **/X** options. Searches each file from the end backwards to the beginning. This option is useful if you want to display the last occurrence of the search string in each file instead of the first (the default). It may also speed up searches for information that is normally at the end of a file, such as a signature.

**/S(ubdirectories)**: Display matches from the current directory and all of its subdirectories. By default, FFINd processes only those subdirectories without the Hidden or System attributes. To view hidden or system subdirectories use **/A** along with **/S**.

**/T"text"** (Text search): Specify the text search string. **/T** must be followed by a text string in double quotes (*e.g.*, **/t"color"**). FFINd will perform a case-insensitive search unless you also use the **/C** option. For a hexadecimal search and / or hexadecimal display of the location where the search string is found, see **/X**. You can specify a search string with either **/T** or **/X**, but not both.

**/U**: Only display the summary.

**/V(erbose):** Show every matching line. FFIND's default behavior is to show only the first matching line and then go on to the next file. This option is only valid with **/T** or **/X**.

**/X["xx"]** (Hexadecimal display / search): Specify hexadecimal display and an optional hexadecimal search string.

If **/X** is followed by one or more pairs of hexadecimal digits in quotes (e.g., **/x"44 63 65"**), FFIND will search for that exact sequence of characters or data bytes without regard to the meaning of those bytes as text. If those bytes are found, the offset is displayed (in both decimal and hexadecimal). A search of this type will always be case-sensitive.

If **/X** is **not** followed by a hexadecimal search string it must be used with **/T**, and will change the output format to display offsets (in both decimal and hexadecimal) rather than actual text lines when the search string is found. For example, this command uses **/T** to display the first line in each *.BTM* file containing the word "hello":

```
[c:\] ffind /t"hello" *.btm
---- c:\test.btm
echo hello

1 line in 1 file
```

If you use the same command with **/X**, the offset is displayed instead of the text:

```
[c:\] ffind /t"hello" /x *.btm
---- c:\test.btm
Offset: 26 (1Ah)

1 line in 1 file
```

You can specify a search string with either **/T** or **/X**, but not both.

**/Y:** Prompt to stop searching after each match. This option is most useful when you are using FFIND to search for one specific file.

## FOR

(Enhanced)

**Purpose:** Repeat a command for several values of a variable.

**Format:** FOR [/A:[-][+]*rhsad*] /D /F [*options*] /H /I"text" /L /R [*path*] %*var* IN ([@]*set* | *start, step, end*) [DO] *command* ...

***options*:** Parsing options for a "file parsing" **FOR**.

***path*:** The starting directory for a "recursive" **FOR**.

**%*var*:** The variable to be used in the command ("FOR variable").

***set*:** A set of values for the variable.

***start*:** The starting value for a "counted" **FOR**.

***step*:** The increment value for a "counted" **FOR**.

***end*:** The limit value for a "counted" **FOR**.

***command*:** A command or group of commands to be executed for each value of the variable.

/A: (Attribute select)

/I (match descriptions)

/D(isable "/")

/L (counted loop)

/F(file parsing)

/R(ecursive)

/H(ide dots)

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104). Ranges **must** appear immediately after the FOR keyword. Use extended wildcards with caution on LFN volumes; see page 105 for details.

**Usage:** FOR begins by creating a set. It then executes a command for every member of the set. The command can be an internal command, an alias, an external command, or a batch file. The members of the set can be a list of file names, text strings, a group of numeric values, or text read from a list of files.

When the ***set*** is made up of text or several separate file names (not an include list), the elements must be separated by spaces, tabs, commas, or the switch character (normally a slash [/]).

FOR includes a large number of options, some of which duplicate functions available in other internal commands, and / or do not follow conventions you may find in our other commands. Most of these extra options are included for compatibility with *CMD.EXE* in Windows NT /

2000 / XP. However, we make them available in all three of our products so that aliases and batch files which use them can work under all products.

The first three sections below (*Working with Files*, *Working with Text*, and *Retrieving Text from Files*) describe the traditional FOR command, and the enhancements to it which are included in 4DOS, 4NT, and Take Command. The sections on *Parsing Text from Files* and *Counted FOR Loops* describe features added for compatibility with Windows NT 4.0, Windows 2000, and Windows XP. The section entitled *Other Notes* contains information you may need if you use any aspect of the FOR command extensively.

### ***Working with Files***

Normally, the *set* is a list of files specified with wildcards. For example, if you use this line in a batch file:

```
for %x in (*.txt) do list %x
```

then LIST will be executed once for each file in the current directory with the extension *.TXT*. The FOR variable %x is set equal to each of the file names in turn, then the LIST command is executed for each file. (You could do the same thing more easily with a simple LIST \*.TXT. We used FOR here so you could get a feel for how it operates, using a simple example. Many of the other examples in this section are constructed in the same way.)

The *set* can include multiple files or an include list, like this:

```
for %x in (d:\*.txt;*.doc;*.asc) do type %x
```

FOR supports wildcards and extended wildcards (see page 94), as well as extended parent directory names (*e.g.*, ...\*\*.txt* to process all of the *.TXT* files that are contained in the directory 2 levels above the current directory).

When you use FOR on an LFN drive, you must quote any file names within the *set* which contain white space or special characters. The same restriction applies to names returned in the FOR variable, if you pass them to internal commands or other commands which require quoting filenames with white space. FOR does not quote returned names automatically, even if you include quotes in the *set*. See page 15 for additional details on file name quoting.

For example, suppose that the program **myedit** can accept a quoted long filename on its command line. The first command below will pass the names of all files which begin with the string “long name” to **myedit**, but the names passed to **myedit** will not be quoted so the command will fail. The second command will quote the names passed to **myedit** and should therefore work as expected:

```
[c:\] for %f in ("long name*") edit %f
[c:\] for %f in ("long name*") edit "%f"
```

If the **set** includes filenames, the file list can be further refined by using date, time, size and file exclusion ranges (see pages 97 and 102). The range or ranges must be placed immediately after the word **FOR**. Ranges will be ignored if no wildcards are used inside the parentheses. For example, this **set** is made up of all of the **.TXT** files that were created or updated on October 4, 2002:

```
for [/d2002-10-4,+0] %x in (*.txt) do ...
```

If the **command** is an internal command that supports ranges, an independent range can also be used in the **command** itself.

You can also refine the list by limiting it with the **/A:** option to select only files that have specific attributes.

By default, **FOR** works only with files in the current directory or a specified directory. With the **/R** option, **FOR** will also search for files in subdirectories. For example, to work with all of the **.TXT** files in the current directory and its subdirectories:

```
for /r %x in (*.txt) do ...
```

If you specify a directory name immediately after **/R**, **FOR** will start in that directory and then search each of its subdirectories. This example works with all of the **.BAK** files on drive D:

```
for /r d:\ %x in (*.bak) do ...
```

- ❖ When you use wildcards to specify the **set**, **FOR** scans the directory and finds each file which matches the wildcard name(s) you specified. If, during the processing of the **FOR** command, you create a file that could be included in the **set**, it may or may not appear in a future iteration of the same **FOR** command. Whether the new file appears depends on its physical location in the directory structure. For example, if you use **FOR** to execute a command for all **.TXT** files, and the command also

creates one or more new *.TXT* files, those new files may or may not be processed during the current **FOR** command, depending on where they are placed in the physical structure of the directory. This is an operating system constraint over which the command processor has no control. Therefore, in order to achieve consistent results you should construct **FOR** commands which do not create files that could become part of the *set* for the current command.

### ***Working with Text***

The *set* can also be made up of text instead of file names. For example, to create three files named file1, file2, and file3, each containing a blank line:

```
for %suffix in (1 2 3) do echo. > file%suffix
```

You can also use the names of environment variables as the text. This example displays the name and content of several variables from the environment (see page 149 for details on the use of square brackets when expanding environment variables). Enter this on one line:

```
for %var in (path prompt comspec) do echo %var=%[%var]
```

### ***Retrieving Text from Files***

**FOR** can extract text from files in two different ways. The first method extracts each line from each file in the *set* and places it in the variable. To use this method, place an **[@]** at the beginning of the *set*, in front of the file name or names.

For example, if you have a file called *DRIVES.TXT* that contains a list of drives on your computer, one drive name per line (with a “:” after each drive letter), you can print the free space on each drive this way:

```
for %d in (@drives.txt) do free %d > prn
```

Because the **[@]** is also a valid filename character, **FOR** first checks to see if the file exists with the **[@]** in its name (*i.e.*, a file named *@DRIVES.TXT*). If so, the filename is treated as a normal argument. If it doesn't exist, **FOR** uses the filename (without the **[@]**) as the file from which to retrieve text.

If you use **@CON** as the filename, **FOR** will read from standard input (a redirected input file) or from a pipe; see pages 88 and 91 for more



information. If you use @CLIP: as the filename, FOR will read any text available from the Windows clipboard (in 4DOS, you can only access the clipboard when running under Microsoft Windows.).

### ❖ **Parsing Text from Files**

The second method of working with text from files is to have FOR parse each line of each file for you. To begin a "file-parsing" FOR, you must use the /F option and then include one or more file names in the *set*. When you use this form of FOR, the variable must be a single letter, for example, %a.

This method of parsing, included for compatibility with *CMD.EXE*, can be cumbersome and inflexible. For a more powerful method, use FOR with @filename as the *set* to retrieve each line from the file, as described in the previous section. Then use variable functions like @INSTR, @LEFT, @RIGHT, and @WORD to parse the line (see page 165 for information on variable functions).

By default, FOR will extract the first word or *token* from each line and return it in the variable. For example, to display the first word on each line in the file *FLIST.TXT*:

```
for /f %a in (flist.txt) do echo %a
```

You can control the way FOR /F parses each line by specifying one or more parsing options in a quoted string immediately after the /F. The available options are:

**skip=n:** FOR /F will skip "n" lines at the beginning of each file before parsing the remainder of the file.

**tokens=n, m, ...:** By default, FOR /F returns just the first word or "token" from each parsed line in the variable you named. You can have it return more than one token in the variable, or return tokens in several variables, with this option.

This option is followed by a list of numbers separated by commas. The first number tells FOR /F which token to return in the first variable, the second number tells it which to return in the second variable, etc. The variables follow each other alphabetically starting with the variable you name on the FOR command line. This example returns the first word of each line in each text file in %d, the second in %e, and the third in %f:

```
for /f "tokens=1,2,3" %d in (*.txt) do ...
```

You can also indicate a range of tokens by separating the numbers with a hyphen [-].

**eol=c:** If FOR /F finds the character “c” in the line, it will assume that the character and any text following it are part of a comment and ignore the rest of the line.

**delims=xxx..:** By default, FOR /F sees spaces and tabs as word or token delimiters. This option replaces those delimiters with all of the characters following the equal sign to the end of the string. This option must therefore be the last one used in the quoted options string.

You can also use FOR /F to parse a single string instead of each line of a file by using the string, in quotes, as the *set*. For example, this command will assign variable A to the string “this”, B to “is”, etc., then display “this” (enter the command on one line):

```
for /f "tokens=1,2,3,4" %a in ("this is a test") do  
echo %a
```

### ❖ “Counted” FOR Loop

The “counted FOR” loop is included for compatibility with *CMD.EXE*. In most cases, you will find the DO command (page 322) more useful for performing counted loops.

In a counted FOR command, the *set* is made up of numeric values instead of text or file names. To begin a counted FOR command, you must use the /L option and then include three values, separated by commas, in the *set*. These are the *start*, *step*, and *end* values. During the first iteration of the FOR loop, the variable is set equal to the *start* value. Before each iteration, the variable is increased by the *step* value. The loop ends when the variable exceeds the *end* value. This example will print the numbers from 1 to 10:

```
for /l %val in (1,1,10) do echo %val
```

This example will print the odd numbers from 1 to 10:

```
for /l %val in (1,2,10) do echo %val
```

The **step** value can be negative. If it is, the loop will end when the variable is less than the **end** value.

### ❖ Other Notes

You can use either % or %% in front of the variable name. Either form will work, whether the FOR command is typed from the command line or is part of an alias or batch file (some traditional command processors require a single % if FOR is used at the command line, but require %% if FOR is used in a batch file). The variable name can be up to 80 characters long. The word DO is optional.

If you use a single-character FOR variable name, that name is given priority over any environment variable which starts with the same letter, in order to maintain compatibility with the traditional FOR command. For example, the following command tries to add a: and b: to the end of the PATH, but will not work as intended:

```
[c:\] for %p in (a: b:) do path %path;%p
```

The “%p” in “%path” will be interpreted as the FOR variable %p followed by the text “ath”, which is not what was intended. To get around this, use a different letter or a longer name for the FOR variable, or use square brackets around the variable name (see page 149).

FOR variables can be referenced like normal environment variables, but are not stored in the same way, and cannot be modified with the SET, ESET, or UNSET commands.

The following example uses FOR with variable functions to delete the .BAK files for which a corresponding .TXT file exists in the current directory (this should be entered on one line):

```
[c:\docs] for %file in (*.txt) do del
    %@name[%file].bak
```

The above command may not work properly on an LFN drive, because the returned FILE variable might contain white space. To correct this problem, you need two sets of quotes, one for DEL and one for %@NAME (enter this on one line):

```
[c:\docs] for %file in (*.txt) do del
    "%@name["%file"]".bak"
```

You can use command grouping (see page 117) to execute multiple commands for each element in the *set*. For example, the following command copies each *.WKQ* file in the current directory to the *D:\WKSAVE* directory, then changes the extension of each file in the current directory to *.SAV*. This should be entered on one line:

```
[c:\text] for %file in (*.wkq) do (copy %file
      d:\wksave\ & ren %file *.sav)
```

(Use a caret [**^**] as the separator character if you are using 4DOS).

In a batch file you can use GOSUB to execute a subroutine for every element in the *set*. Within the subroutine, the FOR variable can be used just like any environment variable. This is a convenient way to execute a complex sequence of commands for every element in the *set* without CALLing another batch file.

One unusual use of FOR is to execute a collection of batch files or other commands with the same parameter. For example, you might want to have three batch files all operate on the same data file. The FOR command could look like this (enter this on one line):

```
[c:\] for %cmd in (filetest fileform fileprnt)
      do %cmd datafile
```

This line will expand to three separate commands:

```
filetest datafile
fileform datafile
fileprnt datafile
```

The variable that FOR uses (the **%CMD** in the example above) is created in the environment and then erased when the FOR command is done. For compatibility with *COMMAND.COM* and *CMD.EXE*, a single-character FOR variable is created in a special way that does not overwrite an existing environment variable with the same name. When using a multi-character variable name you must be careful not to use the name of one of your environment variables as a FOR variable. For example, a command that begins

```
[c:\] for %path in ...
```

will write over your current path setting, then erase the path variable completely when FOR is done.

**4DOS** FOR statements can be nested. Under 4DOS, the permissible nesting level depends on the amount of free space in 4DOS's internal stack. If you receive a stack overflow error when using FOR in complex, nested command sequences, see the notes under the StackSize directive on page 238.

**Options:** **/A:** (Attribute select): Process only those files that have the specified attribute(s). **/A:** will be used only when processing wildcard file names in the *set*. It will be ignored for filenames without wildcards or other items in the *set*. See **Attribute Switches** on page 111 for information on the attributes which can follow **/A:**.

For example, to process only those files with the archive attribute set:

```
for /a:a %f in (*.*) echo %f needs a backup!
```

**4DOS** ❖ **/D**(isable "/): Disables the special processing of the forward slash [/] character in the FOR *set*. For compatibility with certain versions of *COMMAND.COM* (in those prior to Windows 95), 4DOS normally treats a forward slash inside the *set* as an "escape" character, discards the slash, and returns the character after the slash, followed by the remainder of the string. This behavior can be used in batch files to separate a string into individual characters (although 4DOS provides a much easier method with the @INSTR and @LEFT variable functions).

The **/D** option must follow the FOR keyword and come before the variable name. These examples show the effects of **/D**:

```
[c:\] for %s in (/abcdef) do echo %s
a
bcdef
```

```
[c:\] for /d %s in (/abcdef) do echo %s
/abcdef
```

**/F**(ile parsing): Return one or more words or tokens from each line of each file in the set. The **/F** option can be followed by one or more options in a quoted string which control how the parsing is performed. See the details under Parsing Text From Files, above.

**/H**(ide dots): Suppress the assignment of the "." and ".." directories to the FOR variable.

**/I"text"**: Select filenames by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text

must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**/L** ("Counted loop"): Interpret the three values in the **set** as the **start**, **step**, and **end** values of a counted loop. See the details under "Counted FOR" Loop, above.

**/R**(ecursive): Look in the current directory and all of its subdirectories for files in the **set**. If the **/R** is followed by a directory name, look for files in that directory and all of its subdirectories. If you are passing a variable as the directory name, you must enclose it in double quotes so it won't be interpreted as the FOR variable name.

**FREE**

(New)

**Purpose:** Display the total disk space, total bytes used, and total bytes free on the specified (or default) drive(s).

**Format:** FREE [*drive: ...*]

***drive:*** One or more drives to include in the report.

See also: MEMORY.

**Usage:** FREE provides the same disk information as the external command CHKDSK, but without the wait, since it does not check the integrity of the file and directory structure of the disk.

A colon [:] is required after each drive letter. This example displays the status of drives A and C:

```
[c:\] free a: c:
Volume in drive A: is unlabeled
 1,213,952 bytes total disk space
 1,115,136 bytes used
   98,816 bytes free
Volume in drive C: is DEVELOPMENT
242,496,000 bytes total disk space
236,851,712 bytes used
 5,644,288 bytes free
```

**4DOS** DOS networks with large server disk drives (over 2 GB) may report disk space values that are too small when FREE is used. If this occurs, it is because the network software does not report the proper values to 4DOS.

**FTYPE** [4NT, TC] (Enhanced)

**Purpose:** Modify or display the command used to open a file of a type specified in the Windows registry.

**Format:** FTYPE [/P /R *file...*] [*filetype*=[*command*]]

***file*:** One or more files containing entries to be added to the registry.

***filetype*:** A file type stored in the Windows registry.

***command*:** The command to be executed when a file of the specified type is opened.

**/P**(ause)

**/R**(ead from file)

See also: ASSOC, and Executable Extensions on page 106.

**Usage:** FTYPE allows you to display or update the command used to open a file of a specified type listed in the Windows registry.

FTYPE modifies the behavior of Windows file associations stored under the registry handle HKEY\_CLASSES\_ROOT, and discussed in more detail on page 112. If you are not familiar with file associations be sure to read about them before using FTYPE.

The entry modified by FTYPE is the **Shell\Open\Command** entry for the specified file type, which defines the application to execute when a file of that type is opened. The open action is generally invoked by selecting **Open** on the popup menu for a file from the Windows Explorer. Note that opening a file and double-clicking its icon (or selecting the icon and pressing Enter) may not be the same thing — double-clicking or pressing Enter invokes the default action for the file type, which may or may not be “Open”.

If you invoke FTYPE with no parameters, it will display the current file types and associated shell open commands. Use the **/P** switch to pause the display at the end of each page. If you include a ***filetype***, with no equal sign or ***command***, FTYPE will display the current command for that file type.

If you include the equal sign and ***command***, FTYPE will create or update the shell open command for the specified file type. The ***command*** generally includes an application name, including full path, plus parameters. The specific syntax required depends on the internal operation of both Windows and the application involved, and is beyond



the scope of this manual. You can learn about typical syntax by reviewing appropriate Windows and application documentation, and / or by checking through the current contents of your registry.

To remove the shell open command for a file type, use a command like **FTYPE filetype=**, with no ***command*** parameter. This will not delete the shell open command entry from the registry; it simply sets the command to an empty string.

- ! FTYPE should be used with caution, and only after backing up the registry. Improper changes to file associations can prevent applications and / or the operating system from working properly.

Options: /**P**(ause): Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail on page 80.

/**R**(ead file): This option loads a list of file types and associated shell open commands. If no filename is specified and the input is redirected, FTYPE will read from STDIN. The format of the file is the same as that of the FTYPE display.

## FUNCTION

(New)

**Purpose:** Create or display user-defined variable functions.

**Format:** FUNCTION [/P /R *file*...] name=value

***file*:** One or more files containing entries to be added to the function list.

***name*:** The name of the function you want to display or set.

***value*:** Variable function to be substituted for the variable name.

**/P(ause)**

**/R(ead from file)**

See also: UNFUNCTION.

**Usage:** FUNCTION allows you to create or display user-defined variable functions that can be used everywhere variable functions are used.

If you invoke FUNCTION with no parameters, it will display the current function list. If you include a ***name***, with no equal sign or ***value***, FUNCTION will display the current association for that extension.

If you include the equal sign and ***value***, FUNCTION will create or update the function referred to by ***name***.

Variables in the function are numbered from %0 to %255, and are replaced with the matching argument when the function is called. %0 is the function name; %1 is the first argument. For example, the function:

```
function leftmost=`%@left[1,%1]
```

will return the leftmost character in a string.

**Options:** **/P(ause):** Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail on page 80.

**/R(ead file):** This option loads a list of functions from a file. If no filename is specified and input is redirected, /R will read from STDIN. The format of the file is the same as that of the FUNCTION display:

```
name=value
```

**GLOBAL**

(New)

**Purpose:** Execute a command in the current directory and its subdirectories.

**Format:** GLOBAL [/H /I /P /Q] *command*

***command:*** The command to execute, including arguments and switches.

**/H**(idden directories)

**/P**(rompt)

**/I**(gnore exit codes)

**/Q**(uiet)

**Usage:** GLOBAL performs the command first in the current directory and then in every subdirectory under the current directory. The command can be an internal command, an alias, an external command, or a batch file.

This example copies the files in every directory on drive A to the directory *C:\TEMP*:

```
[a:\] global copy *.* c:\temp
```

If you use the **/P** option, GLOBAL will prompt for each subdirectory before performing the command. You can use this option if you want to perform the command in most, but not all subdirectories of the current directory.

- ❖ You can use command grouping (see page 117) to execute multiple ***commands*** in each subdirectory. For example, the following command copies each *.TXT* file in the current directory and all of its subdirectories to drive A. It then changes the extension of each of the copied files to *.SAV*:

```
[c:\] global (copy *.txt a: & ren *.txt *.sav)
```

**4DOS !** If you receive a stack overflow error when using GLOBAL in complex, nested command sequences, see the notes under the StackSize directive on page 238.

**Options:** ❖**H**(idden directories): Forces GLOBAL to look for hidden directories. If you don't use this switch, hidden directories are ignored.

**/I**(gnore exit codes): If this option is not specified, GLOBAL will terminate if the command returns a non-zero exit code. Use **/I** if you want the command to continue in additional subdirectories even if it

returns an error in one subdirectory. Even if you use **/I**, GLOBAL will normally halt execution if the command processor receives a **Ctrl-C** or **Ctrl-Break**.

**/P(rompt)**: Forces GLOBAL to prompt with each directory name before it performs the command. Your options at the prompt are explained in detail on page 80.

**/Q(uiet)**: Do not display the directory names as each directory is processed.

## **GOSUB**

(New)

**Purpose:** Execute a subroutine in the current batch file.

**Format:** GOSUB label [arguments]

***label:*** The batch file label at the beginning of the subroutine.

See also: CALL, GOTO and RETURN.

**Usage:** GOSUB can only be used in batch files.

4DOS, 4NT, and Take Command allow subroutines in batch files. A subroutine must start with a ***label*** (a colon [:] followed by a label name) which appears on a line by itself. Case differences are ignored when matching labels. The subroutine must end with a RETURN statement.

The subroutine is invoked with a GOSUB command from another part of the batch file. After the RETURN, processing will continue with the command following the GOSUB command. For example, the following batch file fragment calls a subroutine which displays the directory and returns:

```
echo Calling a subroutine
gosub subr1
echo Returned from the subroutine
quit
:subr1
dir /a/w
return
```

4DOS searches the entire batch file for the ***label***, starting at the beginning of the file. For compatibility with *CMD.EXE*, 4NT and Take Command begin their search on the next line of the batch file (after the GOSUB command). If the label is not found between the current position and the end of the file, GOSUB will restart the search at the beginning of the file.

If the label does not exist, the batch file is terminated with the error message "Label not found."

You can define GOSUB variables by placing them after the label name on the GOSUB line. For example:

```
Gosub Sub1 abc 15 "Hello World"
```

The variable names are defined on the label line. For example:

```
:Sub1 [str n world]
```

defines three variables - %str (set to "abc"), %n (set to 15), and %world (set to "Hello World"). Note that the square brackets are required on the label line. GOSUB variables are only defined for the duration of the subroutine. They are not inherited by nested GOSUBs, and are destroyed by the RETURN call.

GOSUB variables are placed in the environment in a special form for the duration of the subroutine, and will "mask" any environment variables of the same name that existed before the subroutine was called. GOSUB variables can be referenced like normal environment variables, but are not stored in the same way, and cannot be modified with the SET, ESET, or UNSET commands.

You cannot use SET within a subroutine to change the value of a GOSUB variable. If you attempt to do so, the SET command will set the standard environment variable of the same name, not the GOSUB variable, but this value will be "masked" by the GOSUB variable and will remain inaccessible until the subroutine ends.

GOSUB saves the IFF and DO states, so IFF and DO statements inside a subroutine won't interfere with statements in the part of the batch file from which the subroutine was called.

You cannot RETURN from a GOSUB while inside a DO loop.

If the command processor reaches the end of the batch file while inside a subroutine, it will automatically return to the command after the GOSUB, just as if an explicit RETURN command had been included as the last line of the file.

- ❖ Subroutines can be nested. Under 4DOS, the permissible nesting level depends on the amount of free space in 4DOS's internal stack. If you receive a stack overflow error when using GOSUB in complex, nested command sequences, see the notes under the StackSize directive on page 238.

**GOTO**

(Enhanced)

**Purpose:** Branch to a specified line inside the current batch file.

**Format:** GOTO [/I] *label*

***label*:** The batch file label to branch to.

/I(FF and DO continue)

See also: GOSUB.

**Usage:** GOTO can only be used in batch files.

After a GOTO command in a batch file, the next line to be executed will be the one immediately after the ***label***. The ***label*** must begin with a colon [:] and appear on a line by itself. The colon is required on the line where the label is defined, but is not required in the GOTO command itself. Case differences are ignored when matching labels.

This batch file fragment checks for the existence of the file *CONFIG.SYS*. If the file exists, the batch file jumps to C\_EXISTS and copies all the files from the current directory to the root directory on A:. Otherwise, it prints an error message and exits:

```
if exist config.sys goto C_EXISTS
echo CONFIG.SYS doesn't exist - exiting.
quit
:C_EXISTS
copy *.* a:\
```

4DOS searches the entire batch file for the ***label***, starting at the beginning of the file. For compatibility with *CMD.EXE*, 4NT and Take Command begin their search on the next line of the batch file (after the GOTO command). If the label is not found between the current position and the end of the file, GOTO will restart the search at the beginning of the file.

If the label does not exist, the batch file is terminated with the error message "Label not found."

- ! ❖ To avoid errors in the processing of nested statements and loops, GOTO cancels all active IFF statements and DO / ENDDO loops unless you use /I. This means that a normal GOTO (without /I) may not

branch to any label that is between an IFF and the corresponding ENDIFF or between a DO and the corresponding ENDDO.

- ❖ For compatibility with *CMD.EXE*, the command:

```
GOTO :EOF
```

will end processing of the current batch file if the label :EOF does not exist. However, this is less efficient than using the QUIT or CANCEL command to end a batch file.

- Options: ❖**/I**(FF and DO continue): Prevents GOTO from canceling IFF statements and DO loops. Use this option only if you are absolutely certain that your GOTO command is branching entirely within any current IFF statement **and** any active DO / ENDDO block. Using **/I** under any other conditions will cause an error later in your batch file.

You cannot branch into another IFF statement, another DO loop, or a different IFF or DO nesting level, whether you use the **/I** option or not. If you do, you will eventually receive an “unknown command” error (or execution of the UNKNOWN\_CMD alias) on a subsequent ENDDO, ELSE, ELSEIFF, or ENDIFF statement.



**HEAD**

(New)

**Purpose:** Display the beginning of the specified file(s).

**Format:** HEAD [/A:[-][+]*rhsad*] /C*n* /I"text" /N*n* /P /Q /V] [*@file*] *file...*

***file*:** The file or list of files that you want to display.

***@file*:** A text file containing the names of the files to display, one per line (see page 109 for details).

/A: (Attribute select)	/P(ause)
/C (number of bytes)	/Q(quiet)
/I"text" (match description)	/V(erbose)
/N(umber of lines)	

See also: LIST, TAIL, TYPE.

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104).

**Internet:** Can be used with FTP servers.

**Usage:** The HEAD command displays the first part of a file. It is normally only useful for displaying ASCII text files. Executable files (.COM and .EXE) and many data files may be unreadable when displayed with HEAD because they include non-alphanumeric characters. You can press **Ctrl-S** to pause HEAD's display and then any key to continue.

To display the first 15 lines of the files *MEMO1* and *MEMO2*:

```
[c:\] head /n15 memo1 memo2
```

To display text from the clipboard use **CLIP:** as the file name. CLIP: will not return any data if the clipboard does not contain text. See page 88 for additional information on CLIP:.

#### **4NT, TC *FTP Usage***

HEAD can also display files on FTP servers. The URL must be enclosed in double quotes so the forward slashes won't be interpreted as switches. For example:

```
[c:\] head "ftp://jpsoft.com/index"
```

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see Using FTP Servers on page 113, and the IFTP command on page 382.

#### **4NT, TC NTFS File Streams**

HEAD supports file streams on NTFS drives under Windows NT / 2000 / XP. You can type an individual stream by specifying the stream name, for example:

```
[c:\] head streamfile:s1
```

If no stream name is specified the file's primary data is displayed.

See NTFS File Streams on page 18 for additional details.

**Options:** **/A:** (Attribute select): Select only those files that have the specified attribute(s) set. See **Attribute Switches** on page 111 for information on the attributes which can follow **/A:**.

**/C:** Display the specified number of bytes. **/C** accepts a **b**, **k**, or **m** at the end of the number. **B** is the number of 512-byte blocks, **k** is thousands of bytes, **K** is kilobytes, **m** is millions of bytes, and **M** is megabytes.

**/I"text":** Select files by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**/N:** The number of lines to display. The default is 10.

**/P(ause):** Prompt after displaying each page. Your options at the prompt are explained in detail on page 80.

**/Q(quiet):** Don't display a header for each file.

**/V(erbose):** Display a header for each file.

**HELP**

(Enhanced, External command)

**Purpose:** Display help for internal commands.

**Format:** HELP [*topic*]

***topic:*** A help topic, internal command, or external command.

**Usage:** Online help is available for 4DOS, 4NT, and Take Command. 4DOS uses its own help program; 4NT and Take Command use the Windows help facility.

See page 55 for more details on getting help at the command line. See the *Introduction and Installation Guide* for a more thorough explanation of the online help available.

If you type the command HELP by itself (or press **F1** when the command line is empty), the table of contents is displayed. If you type HELP plus a topic name, that topic is displayed. For example:

```
help copy
```

displays information about the COPY command and its options.

**4DOS** Under 4DOS, your system's DOS help program can be started from within the help system to get help on DOS external commands. For details see your *Introduction and Installation Guide*.

## HISTORY

**(New)**

**Purpose:** Display, add to, clear, or read the history list.

**Format:** HISTORY [/A *command* /F /N /P /R *file...*]

**command:** A command to be added to the history list.

**file:** One or more files containing entries to be added to the history list.

/A(dd)	/P(ause)
/F(ree)	/R(ead)
/N(o duplicates)	

**See also:** DIRHISTORY and LOG.

**Usage:** 4DOS, 4NT, and Take Command keep a list of the commands you have entered on the command line. See page 61 for information on command recall, which allows you to use the history list to repeat or edit commands you have previously executed.

The **HISTORY** command lets you view and manipulate the command history list directly. If no parameters are entered, **HISTORY** will display the current command history list:

```
[c:\] history
```

With the options explained below, you can clear the list, add new commands to the list without executing them, save the list in a file, or read a new list from a file.

The number of commands saved in the history list depends on the length of each command line. The history list size can be specified at startup from 256 to 8192 characters in 4DOS, or 256 to 32767 characters in 4NT and Take Command (see page 212). The default size is 1024 characters in 4DOS and 2048 characters in 4NT and Take Command.

**4NT, TC** The history list can be stored either locally (a separate history list for each copy of 4NT or Take Command) or globally (all copies of the command processor share the same list). For full details see the discussion of local and global history lists beginning on page 65.

- ❖ You can use the **HISTORY** command as an aid in writing batch files by redirecting the **HISTORY** output to a file and then editing the file

appropriately. However, it is easier to use the LOG /H command for this purpose.

- ❖ You can disable the history list or specify a minimum command-line length to save from the “History” tab of the configuration dialog, or with the HistMin directive in the *.INI* file.

You can control whether duplicate entries will be saved in the history list with the HistDups directive in the *.INI* file.

- ❖ You can save the history list by redirecting the output of HISTORY to a file. This example saves the command history to a file called *HISTFILE* and reads it back again immediately. If you leave out the HISTORY /F command on the second line, the contents of the file will be appended to the current history list instead of replacing it:

```
[c:\] history > histfile
[c:\] history /f
[c:\] history /r histfile
```

If you need to save your command history at the end of each day's work, you might use the first of these commands in your *4START.BTM* / *TCSTART.BTM* or other startup file, and the second in *4EXIT.BTM* / *TCEXIT.BTM*:

```
if exist c:\histfile history /r c:\histfile
history > c:\histfile
```

This restores the previous history list if it exists, then defines an alias which will allow you to save the history before exiting.

Options: ❖ **A(dd)**: Add a command to the history list. This performs the same function as the **Ctrl-K** key at the command line (see page 61).

**/F(ree)**: Erase all entries in the command history list.

**/P(rompt)**: Wait for a key after displaying each page of the list. Your options at the prompt are explained in detail on page 80.

**/N(o duplicates)**: Removes duplicate entries (oldest first) from the history list.

- ❖ **/R(ead)**: Read the command history from the specified file and append it to the history list currently held in memory. Each line in the file must fit within the command-line length (see page 78).

**IF**

(Enhanced)

**Purpose:** Execute a command if a condition or set of conditions is true.

**Format:** IF [/I] [NOT] *condition* [.AND. | .OR. | .XOR. [NOT] *condition* ...]  
*command*

***condition*:** A test to determine if the command should be executed.

***command*:** The command to execute if the condition is true.

/I(ignore case)

See also: IFF; @IF on page 180.

**Usage:** IF is normally used only in aliases and batch files. It is always followed by one or more ***conditions*** and then a ***command***. First, the ***conditions*** are evaluated. If they are true, the ***command*** is executed. Otherwise, the ***command*** is ignored. If you add a NOT before a ***condition***, the ***command*** is executed only when the ***condition*** is false.

You can link ***conditions*** with **.AND.**, **.OR.**, or **.XOR.**, and you can group conditions with parentheses (see Combining Tests below). You can also nest IF statements.

The ***conditions*** can test strings, numbers, the existence of a file or subdirectory, the exit code returned by the preceding external command, and the existence of aliases and internal commands.

The ***command*** can be an alias, an internal command, an external command, or a batch file. The entire IF statement, including all ***conditions*** and the ***command***, must fit on one line.

Some examples of IF conditions and commands are included below; additional examples may be found in the *EXAMPLES.BTM* file which came with your command processor.

- ❖ You can use command grouping (see page 117) to execute multiple ***commands*** if the ***condition*** is true. For example, the following command tests if any *.TXT* files exist. If they do, they are copied to drive A: and their extensions are changed to *.TXO*:

```
if exist *.txt (copy *.txt a: & ren *.txt *.txo)
```

(Change the command separator to a caret [^] to use a command like this under 4DOS. Also, note that the IFF command provides a more structured method of executing multiple commands if a condition or set of conditions is true.)

**4NT,TC** When an IF test fails, the remainder of the command is discarded, and the command processor normally continues with the next command on the line, or the next line. This behavior is not compatible with *CMD.EXE*, which discards all remaining commands on the line when an IF test fails, including those after a command separator or pipe character. To change the behavior so that IF affects all commands on the line, as in *CMD.EXE*, set DuplicateBugs to Yes in the *.INI* file (see page 210).

For example, if DuplicateBugs is set to Yes (the default), the following command will display nothing, because the second ECHO command is discarded along with the first when the **condition** fails. If DuplicateBugs is set to No, it will display “hello”:

```
[c:\] if 1 == 2 echo Wrong! & echo hello
```

## Conditions

The conditional tests listed in the following sections are available in both the IF and IFF commands. They fit into two categories: string and numeric tests, and status tests. The tests can use environment variables, internal variables and variable functions, file names, literal text, and numeric values as their arguments.

### String and Numeric Tests

Six test conditions can be used to test character strings. The same conditions are available for both numeric and normal text strings (see below for details). In each case you enter the test as:

*string1* operator *string2*

The **operator** defines the type of test (equal, greater than or equal, and so on). You should always use spaces on both sides of the **operator**. The operators are:

<u>Operator</u>	<u>Tests</u>
<b>EQ</b> or <b>==</b>	<i>string1</i> equal to <i>string2</i>
<b>NE</b> or <b>!=</b>	<i>string1</i> not equal to <i>string2</i>
<b>LT</b>	<i>string1</i> less than <i>string2</i>
<b>LE</b>	<i>string1</i> less than or equal to <i>string2</i>
<b>GE</b>	<i>string1</i> greater than or equal to <i>string2</i>
<b>GT</b>	<i>string1</i> greater than <i>string2</i>

When IF compares two character strings, it will use either a **numeric** comparison or a **string** comparison. A numeric comparison treats the strings as numeric values and tests them arithmetically. A string comparison treats the strings as text.

The difference between numeric and string comparisons is best explained by looking at the way two values are tested. For example, consider comparing the values 2 and 19. Numerically, 2 is smaller, but as a string it is “larger” because its first digit is larger than the first digit of 19. So the first of these **conditions** will be true, and the second will be false:

```
if 2 lt 19 ...  
if "2" lt "19" ...
```

IF determines which kind of test to do by examining the first character of each string. If both strings begin with a numeric character (a digit, sign, or decimal point), a numeric comparison is used. (If a string begins with a decimal separator it is not considered numeric unless the next character is a digit, and there are no more decimal separators within the string. For example, “.07” is numeric, but “.a” and “.07.01” are not.) If either value is non-numeric, a string comparison is used. To force a string comparison when both values are or may be numeric, use double quotes around the values you are testing, as shown above. Because the double quote is not a numeric character, IF performs a string comparison.

Case differences are ignored in string comparisons. If two strings begin with the same text but one is shorter, the shorter string is considered to be “less than” the longer one. For example, “a” is less than “abc”, and “hello there” is greater than “hello”.

When you compare text strings, you may need to enclose the arguments in double quotes in order to avoid syntax errors which can



occur if one of the argument values is empty (*e.g.*, due to an environment variable which has never been assigned a value). This technique will not work for numeric comparisons, as the quotes will force a string compare, so with numeric tests you must be sure that all variables are assigned values before the test is done.

Numeric comparisons work with both integer and decimal values. The values to be compared must contain only numeric digits, decimal points, and an optional sign (+ or -). The number may contain up to 16 digits to the left of the decimal point, and 8 digits to the right.

**4NT, TC** In order to maintain compatibility with *CMD.EXE*, 4NT and Take Command recognize the following additional names for conditions:

**EQU** is the same as **EQ** and **==**

**NEQ** is the same as **NE** and **!=**

**LSS** is the same as **LT**

**LEQ** is the same as **LE**

**GTR** is the same as **GT**

**GEQ** is the same as **GE**

Internal variables (page 153) and variable functions (page 165) are very powerful when combined with string and numeric comparisons. They allow you to test the state of your system, the characteristics of a file, date and time information, or the result of a calculation. You may want to review the variables and variable functions when determining the best way to set up an IF test.

This batch file fragment runs a program called *WEEKLY* if today is Monday:

```
if "%_dow" == "mon" weekly
```

This batch file fragment tests for a string value:

```
input "Enter your selection : " %%cmd
if "%cmd" == "WP" goto wordproc
if "%cmd" NE "GRAPHICS" goto badentry
```

This example calls *GO.BTM* if the first two characters in the file *MYFILE* are "GO" (enter this example on one line):

```
if "%@left[2,%@line[myfile,0]]" == "GO"
```

```
call go.btm
```

The next two examples test whether there is more than 500 KBytes of free memory or more than 2 MBytes of free EMS memory (the EMS example only applies to 4DOS):

```
c:\> if @@dosmem[K] gt 500 echo Over 500K free
c:\> if @@ems[M] gt 2 echo Over 2 MB EMS free
```

### **Status Tests**

These conditions test the system or command processor status. You can use internal variables and variable functions to test many other parts of the system status.

#### **DEFINED "variable"**

If the variable exists in the environment, the condition is true. This is equivalent to testing whether the variable is not empty, for example the following two commands are equivalent:

```
if defined abc echo Hello
if "%abc" != "" echo Hello
```

#### **ERRORLEVEL [operator] n**

This test retrieves the exit code of the preceding external program. By convention, programs return an exit code of 0 when they are successful and a number between 1 and 255 to indicate an error (depending on the program you are running, the maximum return value may be larger in 4NT and Take Command). The condition can be any of the operators listed above (**EQ**, **!=**, **GT**, etc.). If no operator is specified, the default is **GE**. The comparison is done numerically.

Not all programs return an explicit exit code. For programs which do not, the behavior of **ERRORLEVEL** is undefined.

#### **EXIST filename**

If the file exists, the condition is true. You can use wildcards (see page 94) in the filename, in which case the condition is true if any file matching the wildcard name exists.

Do not use **IF EXIST** to test for existence of a directory (use **IF ISDIR** instead). Due to variations in operating system internals,

**IF EXIST** will not return consistent results when used to test for the existence of a directory.

**ISALIAS** aliasname

If the name is defined as an alias, the condition is true.

**4NT, TC**     **ISAPP** appname

If the application is running, the condition is true. You must enter the full pathname of the application. Both the short and long filename forms of the name will be checked (see LFN File Searches on page 105 for details on the correspondence between short and long filenames).

**ISDIR** | **DIREXIST** path

If the subdirectory exists, the condition is true. For compatibility with Novell DOS / OpenDOS, **DIREXIST** may be used as a synonym for **ISDIR**.

**ISFUNCTION** path

If the user-defined function is loaded, the condition is true.

**ISINTERNAL** command

If the specified command is an active internal command, the condition is true. Commands can be activated and deactivated with the **SETDOS /I** command.

**ISLABEL** label

If the specified label exists in the current batch file, the condition is true. Labels may be one or more words long.

**4NT, TC**     **ISWINDOW** "title"

If a window which matches the title exists, the condition is true. Double quotes must be used around the title, which may contain wildcards and extended wildcards (see page 94 for details on wildcards).

The first batch file fragment below tests for the existence of **A:\JAN.DOC** before copying it to drive C (this avoids an error message if the file does not exist):

```
if exist a:\jan.doc copy a:\jan.doc c:\
```

This example tests the exit code of the previous program and stops all batch file processing if an error occurred:

```
if errorlevel == 0 goto success
echo "External Error -- Batch File Ends!"
cancel
```

### ❖ **Combining Tests**

You can negate the result of any test with **NOT**, combine tests of any type with **.AND.**, **.OR.**, and **.XOR.**, and force the desired test order by using parentheses.

When two tests are combined with **.AND.**, the result is true if both individual tests are true. When two tests are combined with **.OR.**, the result is true if either (or both) individual tests are true. When two tests are combined with **.XOR.**, the result is true only if one of the tests is true and the other is false.

This example runs a program called *DATALOAD* if today is Monday or Tuesday (enter this on one line):

```
if "%_dow" == "Mon" .or. "%_dow" == "Tue" dataload
```

Test conditions are always scanned from left to right – there is no implied order of precedence, as there is in some programming languages. You can, however, force a specific test order by grouping conditions with parentheses, for example (enter this on one line):

```
if (%a == 1 .or. (%b == 2 .and. %c == 3)) echo
something
```

Parentheses can only be used when the portion of the **condition** inside the parentheses contains at least one “.and.”, “.or.”, or “.xor.”.

Parentheses on a simple condition which does not combine two or more tests will be taken as part of the string to be tested, and will probably make the test fail. For example, the first of these IF tests would fail; the second would succeed:

```
if (a == a) ...
if (a == a .and. b == b) ...
```

Parentheses can be nested. Under 4DOS, the permissible nesting level depends on the amount of free space in 4DOS's internal stack; if you

receive a stack overflow error when using nested parentheses, see the notes under the StackSize directive on page 238.

**Options:**    **/I**(gnore case): This option, which is only available in 4NT and Take Command, is included only for compatibility with *CMD.EXE*. It has no effect, since all string comparisons in IF are case-insensitive.

**IFF**

(New)

**Purpose:** Perform IF / THEN / ELSE conditional execution of commands.

**Format:** IFF [NOT] *condition* [.AND. | .OR. | .XOR. [NOT]

*condition* ...] THEN & commands

[ELSEIFF *condition* THEN & *commands*] ...

[ELSE & *commands*]

ENDIFF

**Note:** Change the command separator [&] to [^] in the syntax above if you are using 4DOS.

***condition:*** A test to determine if the command(s) should be executed.

***commands:*** One or more commands to execute if the condition(s) is true. If you use multiple commands, they must be separated by command separators or be placed on separate lines of a batch file.

See also: IF, and @IF on page 180.

**Usage:** IFF is similar to the IF command, except that it can perform one set of ***commands*** when a condition or set of ***conditions*** is true and a different set of ***commands*** when the ***conditions*** are false.

IFF can also execute multiple commands when the ***conditions*** are true or false; IF normally executes only one command. IFF imposes no limit on the number of commands and is generally a “cleaner” and more structured command than IF.

IFF is always followed by one or more ***conditions***. If they are true, the ***commands*** that follow the word THEN are executed. Additional ***conditions*** can be tested with ELSEIFF. If none of these ***conditions*** are true, the ***commands*** that follow the word ELSE are executed. After the selected ***commands*** (if any) are executed, processing continues after the word ENDIFF.

If you add a NOT before the condition, the THEN ***commands*** are executed only when the ***condition*** is false and the ELSE ***commands*** are executed only when the ***condition*** is true.

The **commands** may be separated by command separators, or may be on separate lines of a batch file. You must include a command separator or a line break after a THEN, before an ELSEIFF, and before and after an ELSE. Note that the syntax above and the examples below use the default 4NT and Take Command command separator, an ampersand [&]. Replace the ampersand with a caret [^] if you are using 4DOS.

You can link **conditions** with **.AND.**, **.OR.**, or **.XOR.**, and you can group conditions with parentheses. You can nest IFF statements up to 15 levels deep. The **conditions** can test strings or numbers, the existence of a file or subdirectory, the errorlevel returned from the preceding external command, and the existence of alias names and internal commands.

See the IF command for a list of the possible **conditions**, and details on using **.AND.**, **.OR.**, **.XOR.**, and parentheses.

The **commands** can include any internal command, alias, external command, or batch file. The alias in this example checks to see if the argument is a subdirectory. If so, the alias deletes the subdirectory's files and removes it (enter this on one line):

```
[c:\] alias prune `iff isdir %1 then &
    del /sxz %1 & else &
    echo Not a directory! & endiff`
```

- ! ❖ Be sure to read the cautionary notes about GOTO and IFF under the GOTO command (page 365) before using a GOTO inside an IFF statement.
- ❖ If you pipe data to an IFF, the data will be passed to the command(s) following the IFF, not to IFF itself.

**IFTP** [4NT, TC]

(New)

**Purpose:** Open or close an FTP session.

**Format:** IFTP [/C /Q /S /V] ["ftp://[user[:password]@]server[/path]"]

**user:** The user name to login to the FTP site.

**password:** The password to login to the FTP site.

**server:** The FTP server name.

**path:** The default directory on the server for this session.

/C(lose)

/S(end)

/Q(quiet)

/V(erbose)

**Usage:** Most file processing commands and functions in 4NT and Take Command can access files on FTP servers in the same manner as files on local hard drives and a local network. Normally, each time you use the FTP feature of one of these commands or functions, it starts an FTP session, performs its task, and then closes the FTP session.

IFTP starts an FTP session which remains open until you close it. There are several advantages to using IFTP: the FTP connection remains open so commands execute more quickly, the syntax for accessing files on the server is shorter, and you can specify a default directory on the server for file operations.

To open an FTP connection using IFTP, use syntax like this:

```
[c:\] iftp "ftp://user:pwd@jpsoft.com/dir1"
```

This command tells IFTP to open an FTP session with the server **jpsoft.com**, send **user** as the login username and **pwd** as the login password, and to establish the directory **/dir1** as the default directory for this session. The user name and password are optional; if they are not used, IFTP will attempt to log in anonymously. The quotation marks are required. If you specify a password of "\*", you will be prompted to enter the password (which will be appear on the screen as \*'s).

Note that in the example above **dir1** is a subdirectory of the FTP "root" directory – the home directory for the named FTP user. In most server configurations this is not the same as the FTP server's physical root directory.



If you enter IFTP with no arguments, the current server name and directory will be displayed.

Once you have established an FTP session with IFTP, you can refer to files on the server by using "ftp:" but leaving out the user name, password, and URL of the server. On most servers, file and path names which begin "ftp:" are relative to the default directory, if any, that you specified when you opened the IFTP session; file and path names which begin "ftp:/" are relative to the root directory for the login name.

The difference can be seen in these 4 DIR commands, assuming the IFTP session started above:

1. [c:\] dir "ftp:\*.txt"
2. [c:\] dir "ftp:dir2/\*.txt"
3. [c:\] dir "ftp:/\*.txt"
4. [c:\] dir "ftp:/dir2/\*.txt"

The first command lists the *.TXT* files in the default session directory, **dir1**. The second command lists the *.TXT* files in **/dir1/dir2** because it interprets the path *dir2/\*.txt* to be relative to the default directory. The quotes could be omitted from example 1 because it contains no forward slash that could be mistaken as an option switch

The third and fourth commands above, because they include a [/] immediately following the "ftp:" designator, are relative to the root directory. Command 3 lists the *.TXT* files in the root directory and command 4 lists the files in the **dir2** subdirectory of the root directory.

You can only have one IFTP session open at a time. However, while you have an IFTP session open, you can still use a complete FTP URL to perform an operation on a different server. For example, while the session above is open, you can use this command to display all files in the root directory of jpsoft.com:

```
[c:\] dir "ftp://jpsoft.com/*"
```

If you have an open connection created with IFTP, you can determine the server to which you are connected by entering the IFTP command with no arguments.

An IFTP session remains open until you explicitly close it with this command:

```
[c:\] iftp /c
```

Most FTP servers “time out” after a period of inactivity. 4NT and Take Command cannot detect this timeout, and will simply display errors if you try to use a connection that has been closed by the server. You should not assume that an IFTP connection will continue to function if you leave it open but unused for a significant period of time.

- ! IFTP and the other FTP features of 4NT and Take Command rely on the server's compliance with Internet FTP standards. If your server is not fully compliant, or does not operate in the manner that 4NT and Take Command expect, commands may not work as you intend. We urge you to test each server you use with nondestructive commands like DIR before you try to copy or delete files, create or remove directories, etc.

Before you can use IFTP, you must establish the necessary connection to the server. For example, if you use Dial-Up Networking to connect to the server, you must start your dial up connection first.

- ❖ If you connect through a proxy server, you must use the Proxy initialization directive (see page 214).

**Option:**     **/C(lose):** Use this switch, with no URL, to close an IFTP session (see the example above).

**/Q(quiet):** Turn off the display of the conversation with the FTP server.

**/S(end):** Allows you to send commands directly to an FTP server. The connection must have already been opened by a previous IFTP command.

**/V(erbose):** Enable the display of the conversation with the FTP server. This can be useful for debugging connection problems.

## INKEY

(New)

**Purpose:** Get a single keystroke from the user and store it in an environment variable.

**Format:** INKEY [/C /D /K"keys" /M /P /Wn /X] [*prompt*] %%varname

***prompt*:** Optional text that is displayed as a prompt.

***varname*:** The variable that will hold the user's keystroke.

/C(lear buffer)

/P(assword)

/D(igits only)

/W(ait)

/K (valid keystrokes)

/X (no carriage return)

/M(ouse button)

See also: INPUT, KEYSTACK, MSGBOX, and QUERYBOX.

**Usage:** INKEY optionally displays a prompt. Then it waits for a specified time or indefinitely for a keystroke, and places the keystroke into an environment variable. It is normally used in batch files and aliases to get a menu choice or other single-key input. Along with the INPUT command, INKEY allows great flexibility in reading input from within a batch file or alias.

If ***prompt*** text is included in an INKEY command, it is displayed while INKEY waits for input.

**4NT, TC** INKEY works within the command line window. If you prefer to use a dialog for user input, see the MSGBOX and QUERYBOX commands.

The following batch file fragment prompts for a character and stores it in the variable *NUM*:

```
inkey Enter a number from 1 to 9: %%num
```

INKEY reads standard input for the keystroke, so it will accept keystrokes from a redirected file or from the KEYSTACK. You can supply a list of valid keystrokes with the /K option.

Standard keystrokes with ASCII values between 1 and 255 are stored directly in the environment variable. Extended keystrokes (for example, function keys and cursor keys) are stored as a string in decimal format, with a leading @ (for example, the F1 key is @59). The **Enter** key is stored as an extended keystroke, with the code @28. See the online help for a list of the ASCII and extended key codes.

To test for a non-printing ASCII keystroke returned by INKEY use the **@ASCII** function (see page 169) to get the numeric value of the key. For example, to test for **Esc**, which has an ASCII value of 27:

```
inkey Enter a key:  %%key
if "%@ascii[%key]" == "27" echo Esc pressed
```

If you press **Ctrl-C** or **Ctrl-Break** while INKEY is waiting for a key, execution of an alias will be terminated, and execution of a batch file will be suspended while you are asked whether to cancel the batch job (see page 131). A batch file can handle **Ctrl-C** and **Ctrl-Break** itself with the ON BREAK command (see page 425).

**Options:** **/C**(lear buffer): Clears the keyboard buffer before INKEY accepts keystrokes. If you use this option, INKEY will ignore any keystrokes which you type, either accidentally or intentionally, before it is ready to accept input.

**/D**(igits only): Prevents INKEY from accepting any keystroke except a digit from 0 to 9.

**/K"keys"**: Specify the permissible keystrokes. The list of valid keystrokes should be enclosed in double quotes. For alphabetic keys the validity test is not case-sensitive. You can specify extended keys by enclosing their names in square brackets (within the quotes). Enter this example on one line:

```
inkey /k"ab[Ctrl-F9]" Enter A, B, or Ctrl-F9 %%var
```

See page 34 for a complete listing of the key names you can use within the square brackets, and a description of the key name format.

If an invalid keystroke is entered, the command processor will echo the keystroke if possible, beep, move the cursor back one character, and wait for another keystroke.

**/M**(ouse button): Returns @240 if you click the left mouse button, @241 for the right button, and @242 for the middle button.

**/P**(assword): Prevents INKEY from echoing the character.

**/W**(ait): Timeout period, in seconds, to wait for a response. If no keystroke is entered by the end of the timeout period, INKEY returns with the variable unchanged. This allows you to continue the batch file if the user does not respond in a given period of time. You can specify

**/W0** to return immediately if there are no keys waiting in the keyboard buffer.

For example, the following batch file fragment waits up to 10 seconds for a character, then tests to see if a “Y” was entered:

```
set netmon=N
inkey /K"YN" /w10 Network monitor (Y/N)? %%net
iff "%netmon" == "Y" then
    rem Commands to load the monitor program
endiff
```

**/X** (no carriage return): Prevents INKEY from displaying a carriage return and line feed after the user's entry.

**INPUT**

(New)

**Purpose:** Get a string from the keyboard and save it as an environment variable.

**Format:** INPUT [/C /D /E /Ln /N /P /Wn /X] [*prompt*] %%*varname*

***prompt*:** Optional text that is displayed as a prompt.

***varname*:** The variable that will hold the user's input.

/C(lear buffer)

/N(o colors)

/D(igits only)

/P(assword)

/E(dit)

/W(ait)

/L(ength)

/X (no carriage return)

See also: INKEY, KEYSTACK, MSGBOX, and QUERYBOX.

**Usage:** INPUT optionally displays a prompt. Then it waits for your entry, and places any characters you type into an environment variable. INPUT is normally used in batch files and aliases to get multi-character input (for single-keystroke input, see INKEY).

**4NT, TC** INPUT works within the command line window. If you prefer to use a dialog for user input, see the MSGBOX and QUERYBOX commands.

If ***prompt*** text is included in an INPUT command, it is displayed while INPUT waits for input. Standard command-line editing keys may be used to edit the input string as it is entered. If you use the /P password option, INPUT will echo asterisks instead of the keys you type.

All characters entered up to, but not including, the carriage return are stored in the variable.

The following batch file fragment prompts for a string and stores it in the variable FNAME:

```
input Enter the file name: %%fname
```

INPUT reads standard input, so it will accept text from a re-directed file or from the KEYSTACK.

If you press **Ctrl-C** or **Ctrl-Break** while INPUT is waiting for input, execution of an alias will be terminated, and execution of a batch file will be suspended while you are asked whether to cancel the batch job

(see page 131). A batch file can handle **Ctrl-C** and **Ctrl-Break** itself with the **ON BREAK** command (see page 425).

- ❖ You can pipe text to **INPUT**; if you do, it will set the variable to the first line it receives. This approach should only be used in **4DOS**. In **4NT** and **Take Command**, it will set the variable in the “child” process used to handle the right hand side of the pipe, but this variable will not be available in the original copy of **4NT** or **Take Command** used to start the pipe. See page 91 for more information on pipes.

**Options:**    **/C**(lear buffer): Discard any keystrokes pending in the keyboard buffer before **INPUT** begins accepting characters.

**/D**(igits only): Prevents **INPUT** from accepting any keystrokes except digits from 0 to 9.

**/E**(dit): Allows you to edit an existing value. If there is no existing value for **varname**, **INPUT** proceeds as if **/E** had not been used, and allows you to enter a new value.

**/Ln** (Length): Sets the maximum number of characters which **INPUT** will accept to “n”. If you attempt to enter more than this number of characters, **INPUT** will beep and prevent further input (though you will still be able to edit characters already typed).

**/N**(o colors): Disables the use of input colors defined in the **InputColor** directive in the **.INI** file, and forces **INPUT** to use the default display colors.

**/P**(assword): Tells **INPUT** to echo asterisks, instead of the characters you type.

**/W**(ait): Timeout period, in seconds, to wait for a response. If no keystroke is entered by the end of the timeout period, **INPUT** returns with the variable unchanged. This allows you to continue the batch file if the user does not respond in a given period of time. If you enter a key before the timeout period, **INPUT** will wait indefinitely for the remainder of the line. You can specify **/W0** to return immediately if there are no keys waiting.

**/X** (no carriage return): Prevents **INPUT** from displaying a carriage return and line feed after the user’s entry.

**KEYBD**

(New)

**Purpose:** Set the state of the keyboard toggles: Caps Lock, Num Lock, and Scroll Lock.

**Format:** KEYBD [/Cn /Nn /Sn]

/C(aps lock)

/S(croll lock)

/N(um lock)

**n** can be either 0 to turn off the toggle or 1 to turn on the toggle.

**Usage:** Most keyboards have 3 toggle keys, the Caps Lock, Num Lock, and Scroll Lock. The toggle key status is usually displayed by three lights at the top right corner of the keyboard.

This command lets you turn any toggle key on or off. It is most useful in batch files and aliases if you want the keys set a particular way before collecting input from the user.

For example, to turn off the Num Lock and Caps Lock keys, you can use this command:

```
[c:\] keybd /c0 /n0
```

If you use the KEYBD command with no switches, it will display the present state of the toggle keys.

**4DOS** ❖ In 4DOS, KEYBD works by performing a BIOS setting. Some memory resident programs that monitor the physical keyboard rather than BIOS settings may not recognize that the state of the toggle keys has changed after a KEYBD command.

The toggle key state is typically the same for all sessions, and changes made with KEYBD in one session will therefore affect all other sessions.

**Options:** /C(aps lock): Turn the Caps Lock key on or off.

/N(um lock): Turn the Num Lock key on or off.

/S(croll lock): Turn the Scroll Lock key on or off.



**KEYS**    *[4NT, TC]*

(Enhanced)

**Purpose:**    Enable, disable, or display the history list.

**Format:**    KEYS [ON | OFF | LIST]

See also: HISTORY.

**Usage:**    This command is provided only for compatibility with the KEYS command in *CMD.EXE*, which controls the history list in Windows NT. The same functions are available by setting the HistMin directive in the *.INI* file, and by using the HISTORY command. (*CMD.EXE*'s KEYS command has no effect in Windows 2000 and XP, because command line editing is always enabled. However, the 4NT and Take Command KEYS command functions as described here under Windows NT, Windows 2000, and Windows XP.)

The history list collects the commands you type for later recall, editing, and viewing. You can view the contents of the list through the history list window or by typing any of the following commands:

```
[c:\] history
[c:\] history /p
[c:\] keys list
```

The first command displays the entire history list. The second displays the entire list and pauses at the end of each full screen. The third command produces the same output as the first, except that each line is numbered.

You can disable the collection and storage of commands in the history list by typing:

```
[c:\] keys off
```

You can turn the history back on with the command:

```
[c:\] keys on
```

If you issue the KEYS command without any parameters, it will show you the current status of the history list.

**KEYSTACK**

(New)

**Purpose:** Feed keystrokes to a program or command automatically.

**Format:** KEYSTACK [/] [/Wx] [ "*abc*" ] [*keyname*[*n*]] ...

**/:** Signal to clear the Keystack and the keyboard buffer.

**x:** Delay in clock ticks.

**"*abc*"**: Literal characters to be placed in the Keystack.

***keyname***: Name or code for a key to be placed in the Keystack.

***n***: Number of times to repeat the named key.

**/W(ait)**

**Usage:** KEYSTACK takes a series of keystrokes and feeds them to a program or command as if they were typed at the keyboard. When the program has used all of the keystrokes in the keystack buffer, it will begin to read the keyboard for input, as it normally would.

KEYSTACK works differently in 4DOS and 4NT / Take Command. The differences are explained in the two sections below, followed by important additional information which applies to both products.

**4DOS**

Under 4DOS, KEYSTACK places keystrokes into a buffer. When an application program (or 4DOS itself) requests another keystroke, the "stacked" keystroke is retrieved from the buffer. The KEYSTACK command must be executed **before** running the program which is going to receive the keystrokes in order to put the keystrokes into the buffer first, so the program can find them when it runs.

KEYSTACK will only work if the memory-resident program *KSTACK.COM* has been loaded. *KSTACK* is usually loaded from the *AUTOEXEC.BAT* file (see page 132). If *KSTACK* is not loaded, the KEYSTACK command will display an error message. If you are using Windows 95 / 98 / ME, see your *Introduction and Installation Guide* for information on loading *KSTACK* within a window.

Programs that bypass DOS and the BIOS for keyboard input cannot read keystrokes entered with KEYSTACK. If you use KEYSTACK and then run such a program, the keystrokes will not appear in the

program, but may appear at the prompt when you exit the program and return to 4DOS.

### ***4NT and Take Command***

KEYSTACK will send the keystrokes to the currently active window. If you want to send keystrokes to another program (rather than have them function with 4NT or Take Command), you must start the program or ACTIVATE its window (see page 248) so it can receive the keystrokes. You must do this **before** executing the KEYSTACK command. You cannot use ACTIVATE and KEYSTACK by entering the commands separately at the prompt, because when you return to the 4NT or Take Command window to enter the KEYSTACK command it will become the active window again, defeating the purpose of the previous ACTIVATE. For this reason the ACTIVATE / KEYSTACK sequence is normally entered using an alias or batch file.

KEYSTACK is most often used for programs started from batch files. In order for KEYSTACK to work in a batch file, you must start the program with the START command, then use the KEYSTACK command. If you start the program directly — without using START — the batch file will wait for the application to complete before continuing and running the KEYSTACK command, and the keystrokes will not appear in the target program.

If you use KEYSTACK in an alias executed from the prompt, the considerations are essentially the same, but depend on whether ExecWait is set (see page 114 for details). If ExecWait is **not** set, you can use KEYSTACK immediately after an application is started. However, if ExecWait **is** set, the KEYSTACK command will not be executed until the program has finished, and the keystrokes will not be sent to the target program.

You may not be able to use KEYSTACK effectively if you have programs running in the background which change the active window (for example, by popping up a dialog box). If a window pops up in the midst of your KEYSTACK sequence, keystrokes stored in the KEYSTACK buffer may go to that window, and not to the application you intended.

**4NT**      KEYSTACK will only work if the file *KEYSTACK.EXE* is in the same directory as *4NT.EXE*, or a directory listed in your PATH. If

*KEYSTACK.EXE* cannot be found, the KEYSTACK command will display an error message.

### **Additional Information**

The remainder of the information on KEYSTACK applies to 4DOS, 4NT, and Take Command, unless otherwise noted.

Characters entered within double quotes ("*abc*") will be sent to the target program "as is". The only items allowed outside double quotes are key names, the ! and /W options, and a repeat count.

See page 34 for a complete listing of key names and a description of the key name and numeric key code format. If you want to send the same key name or numeric code several times, you can follow it with a repeat count in square brackets. For example, to send the Enter key 4 times, you can use this command:

```
keystack enter [4]
```

The repeat count works only with individual keystrokes, or numeric keystroke or character values. It cannot be used with quoted strings.

An exclamation mark [!] will clear all pending keystrokes in the KEYSTACK buffer.

For example, to start Microsoft Word and open the last document you worked on, you could use this command from Take Command (enter this on one line):

```
d:\doc] start winword & keystack /W54 F10 Down "1"
```

This runs Word, delays about three seconds (54 clock ticks at about 1/18 second each) for Word to get started, places the keystrokes for F10 (change to the menu bar), down arrow (display the File menu), and "1" (open the most recently used file) into the buffer. Word receives these keystrokes and performs the appropriate actions.

**4DOS** You can store a maximum of 511 text or special characters in the 4DOS KEYSTACK buffer. A delay takes two character slots in the buffer. A repeated character takes one character slot per repetition.

**4NT, TC** You can store a maximum of 2,038 characters in the KEYSTACK buffer. The count is determined by the number of characters on the KEYSTACK command line, not by the actual number of characters

sent to the application. Each time the KEYSTACK command is executed, it will clear any remaining keystrokes stored by a previous KEYSTACK command.

You may need to experiment with your programs and insert delays (see the /W option) to find the window activation and keystroke sequence that works for a particular program.

### ❖ **4DOS Advanced Options**

KEYSTACK treats the number 0 as a special case; it is used with programs that flush the keyboard buffer. When KEYSTACK processes a key value of 0, it tells the program the buffer is clear, so subsequent keystrokes will be accepted normally. Some programs will require several "0"s before they will accept input; you may need to experiment to determine the correct number.

For example, the following batch file starts a spreadsheet program and loads the file specified on the command line when the batch file is invoked (the KEYSTACK command should be entered on one line):

```
pushd c:\finance
keystack 0 Enter 0 Enter 0 Enter 0 Enter 0 Enter
"/FR" 0 "%1" Enter
spread
popd
```

The sequence of "0 Enter" pairs tells the program that the keyboard buffer is empty, then passes a carriage return, repeating this sequence five times. (You must determine the actual sequence required by your software through experimentation. Few programs require as long a startup sequence as is shown here.) This gets the program to a point where an empty spreadsheet is displayed. The rest of the KEYSTACK line issues a File Retrieve command (/FR), simulates an empty keyboard buffer once more, enters the file name passed on the batch command line (%1), and finally enters a carriage return to end the file name.

Here's the same command defined as an alias (enter this on one line):

```
alias sload `pushd c:\finance ^ keystack 0 Enter 0
Enter 0 Enter 0 Enter 0 Enter "/FR" 0 "%1" Enter ^
spread ^ popd`
```

KEYSTACK mimics the BIOS by stacking both an ASCII code and a scan code for each key. It does so by calculating the code for each character, whether it is entered as part of a quoted string, as a key name, or as an ASCII value less than 128. However, if you are stacking keys for a program which distinguishes between keys with the same symbol, like the plus on the keyboard and the gray plus, you will have to calculate the codes for the keys on the numeric keypad yourself. Calculate the value  $((256 * \text{scan code}) + \text{ASCII code})$  and enter that numeric value as an argument for KEYSTACK.

For example, for the Enter key on the numeric keypad, the scan code is 224 and the ASCII code is 13, so to stack both values use  $((256 * 224) + 13)$  or KEYSTACK 57357. Try this approach if a “normal” KEYSTACK command does not work (for example, if you use “KEYSTACK Enter” for the Enter key and the program doesn't see the correct character). To stack such combined key codes you must use the numeric value, not the key name. See the online help for a complete list of ASCII codes and scan codes.

**Options:**    **❖/W(ait):** Delay the next keystroke in the KEYSTACK buffer by a specified number of clock “ticks”. A clock tick is approximately 1/18 second. The number of clock ticks to delay should be placed immediately after the **W**, and must be between 1 and 65,535 (65,535 ticks is about 1 hour). You can use the **/W** option as many times as desired and at any point in the string of keystrokes except within double quotes. Some programs may need the delays provided by **/W** in order to receive keystrokes properly from KEYSTACK. The only way to determine what delay is needed is to experiment.

**LFNFOR** *[4DOS]*

(Compatible)

**Purpose:** Enable and disable LFN support for FOR wildcards in Windows 9x.

**Format:** LFNFOR [ON | OFF]

See also: FOR and the @ALTNAME and @LFN variable functions.

**Usage:** If you use wildcards in the **set** of a FOR command, 4DOS returns long file names on an LFN volume by default. For example, the command:

```
c:\> for %f in (*.*) do echo %f
```

will, by default, display the long version of each filename in the current directory.

You can alter this behavior with LFNFOR. After the command

```
c:\> lfnfor off
```

the same command will return the short version of each filename in the current directory. You can restore the default behavior with the command

```
c:\> lfnfor on
```

If you enter LFNFOR without either ON or OFF, 4DOS will report the current state of LFNFOR.

LFNFOR is included for compatibility with *COMMAND.COM*. Under 4DOS, you may find it easier to use @ALTNAME (page 169) and @LFN (page 181) to convert between long and short filenames returned by FOR.

LFNFOR only affects the filenames and pathnames in a FOR command, and only when wildcards are used in the set. It has no effect on DIR, SELECT, or any other command which returns file and path names.

**LH / LOADHIGH** [4DOS]

(Compatible)

**Purpose:** Load a memory resident program into an Upper Memory Block (UMB).

**Format:** LH [/L:r1,n1;r2,n2;... /S] *filename*

or

LOADHIGH [/L:r1,n1;r2,n2;... /S] *filename*

***filename:*** The name of the program to load into high memory.

**/L**(oad region)

**/S**(hrink)

**Usage:** LH and LOADHIGH are synonyms. You can use either one.

LOADHIGH requires version 5.0 or later of MS-DOS, PC DOS or DR-DOS, or Windows 95 / 98 / ME.

If you load memory-resident programs into UMBs, you will have more room in conventional memory for application programs. If your system has no UMBs, or if the program is larger than the largest UMB, then LOADHIGH will load the program into conventional base memory.

For example, to load the program *C:\UTIL\CACHE.EXE* into high memory:

```
c:\> loadhigh c:\util\cache.exe
```

If you are running MS-DOS / PC DOS / DR-DOS 5.0 or above, or Windows 95 / 98 / ME, LOADHIGH requires the DOS=UMB command in your *CONFIG.SYS* file.

- ! If you use a memory manager like 386MAX or QEMM to manage your UMBs, rather than the DOS or Windows memory manager and the DOS=UMB directive, then LOADHIGH will not work, and you must use the equivalent command supplied with your memory manager in order to load programs high.

See the **Miscellaneous Reference** section of the online help for complete details on UMBs and UMB regions, and the hardware and software required to support them.

- ❖ The **/L** and **/S** switches are designed to aid in optimizing the use of upper memory by selecting one or more UMB regions for a particular memory-resident program to use. They are fully compatible with the



MS-DOS 6.0 and above memory optimizer, MEMMAKER, and may also be used under MS-DOS 5.0.

While a complete discussion of memory optimization techniques is well beyond the scope of this manual, the basic technique is to load each memory-resident program into a specific region to free up the maximum possible amount of base memory.

Optimizing UMB usage requires a detailed knowledge of the memory requirements for your memory-resident programs. This information can be very difficult to obtain, because some programs require more memory when starting and less when running, and each program is unique. However, if you know these memory requirements, you may be able to use /L and /S to optimize the use of upper memory, or to improve on the optimization done by MEMMAKER or another memory optimizer.

**Options:** **❖/L:r1,n1;r2,n2;...** (Load region): Specifies which region(s) should be used for the program, and optionally the minimum free space required in a region before the program is allowed access to that region.

If /L is not used, all upper memory regions are available to the program. If /L is used, you must specify one or more regions (**r1**, **r2**, etc.); only those regions will be made available. Upper memory regions are numbered sequentially beginning with 1 (region 0 refers to low memory, and is normally used only by MEMMAKER).

If only a region number is given, the entire region is made available to the program (assuming there is free space in the region). If a size is given for a particular region (**n1**, **n2**, etc.), then the region is only made available if the free space in the region is equal to or greater than that size. All sizes are in bytes. Any region not available to a particular program is “locked out” while that program is loading and made available again once the program is loaded. If the combination of /L values you use does not provide sufficient upper memory space for the program to load, it will be loaded in low memory.

You can combine /L values in several ways. In each of the following examples, if the requested space is not available or is insufficient the program is loaded into low memory:

```
lh /l:2 filename
```

The program can use region 2 only.

lh /1:2;3 filename

The program can use regions 2 and 3 only.

lh /1:2,2048 filename

The program can use region 2 only, and only if it has 2048 or more bytes free.

lh /1:2,2048;3 filename

The program can use region 2 if it has 2048 or more bytes free, and region 3 regardless of how many bytes it has free

**/S(hrink):** Shrinks each region selected by /L to the minimum size used in /L before loading the program. This prevents memory-resident programs which take all available memory from using more than you want them to. This switch is primarily intended for use by MEMMAKER.

## LIST

(New)

**Purpose:** Display a file, with forward and backward paging and scrolling.

**Format:** LIST [/A:[-][+]*rhsad*] /H /I /I"*text*" /R /S /T /W /X] [*@file*] *file*...

***file*:** A file or list of files to display.

***@file*:** A text file containing the names of the files to display, one per line (see page 109 for details).

**/A:** (Attribute select)

**/S**(tandard input)

**/H**(igh bit off)

**/T** (search for Text)

**/I**(gnore wildcards)

**/W**(rap)

**/I"*text*"** (match descriptions)

**/X** (heX display mode)

**/R**(everse)

See also: TYPE

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104).

**Internet:** Can be used with FTP servers.

**Usage:** LIST provides a fast and flexible way to view a file, without the overhead of loading and using a text editor.

For example, to display a file called *MEMO.DOC*:

```
[c:\] list memo.doc
```

LIST is most often used for displaying ASCII text files. It can be used for other files which contain non-alphabetic characters, but you may need to use hex mode (see below) to read these files.

**TC** *LIST displays files in the Take Command window. The standard tool bar and scroll bars are replaced with the LIST tool bar and scroll bars. Use the scroll bars or cursor pad to scroll through the file. You can select the LIST commands with the mouse (on the tool bar and scrollbars) or from the keyboard.*

LIST recognizes the following keys and (in Take Command) buttons:

<u>Key</u>	<u>Button</u>	<u>Meaning</u>
Home		Display the first page of the file.
End		Display the last page of the file.

<b>Esc</b>	<b>Cont</b>	Exit the current file.
<b>Ctrl-C</b>	<b>Quit</b>	Quit LIST.
<b>Ctrl-PgUp</b>		Display previous file.
<b>Ctrl-PgDn</b>		Display next file.
<b>Up Arrow</b>		Scroll up one line.
<b>Down Arrow</b>		Scroll down one line.
<b>Left Arrow</b>		Scroll left 8 columns.
<b>Right Arrow</b>		Scroll right 8 columns.
<b>Ctrl-Left Arrow</b>		Scroll left 40 columns.
<b>Ctrl-Right Arrow</b>		Scroll right 40 columns.
<b>Del</b>		Prompt whether to delete the file.
<b>Ins</b>		Prompt whether to save the file (or pipe) to a new name.
<b>Tab</b>		Prompt for a new default tab size.
<b>F1</b>		Display online help.
<b>Ctrl-F1</b>		Display online help for the command at the cursor.
<b>B</b>	<b>Back</b>	Go back to the previous file in the current group of files.
<b>F</b>	<b>Find</b>	Prompt and search for a string or a sequence of hexadecimal values.
<b>Ctrl-F</b>		Prompt and search for a string, searching backward from the end of the file.
<b>G</b>	<b>Goto</b>	Go to a specific line or, in hex mode, to a specific hex offset.
<b>H</b>	<b>High</b>	Toggle the “high bit” (/H) option.
<b>I</b>	<b>Info</b>	Display information on the current file (the full name, size, date, and time).
<b>N</b>	<b>Next</b>	Find next matching string.
<b>Ctrl-N</b>		Find previous matching string in the file.
<b>O</b>	<b>Open</b>	Open a new file.
<b>P</b>	<b>Print</b>	Print selected text, the current page, or the entire file.
<b>W</b>	<b>Wrap</b>	Toggle the “line wrap” (/W) option.

<b>X</b>	<b>heX</b>	Toggle the hex-mode display (/X) option.
----------	------------	--

Text searches performed with **F**, **N**, **Ctrl-F**, **Ctrl-N**, or with the corresponding buttons, are not case-sensitive unless (in Take Command) you check the “Match case” box in the search dialog.

When the search string is found LIST displays the line containing the string at the top of the window, and highlights the string it found. Any additional occurrences of the string on the same display page are also highlighted. Highlighting is intended for use with text files; in binary files the search string will be found, but may not be highlighted properly.

**4DOS, 4NT** If the display is currently in hexadecimal mode and you press **F** or **Ctrl-F**, you will be prompted for whether you want to search in hexadecimal mode. If so, you should then enter the search string as a sequence of 2-digit hexadecimal numbers separated by spaces, for example 41 63 65 (these are the ASCII values for the string “Ace”). Hex searches are case-sensitive, and search for the exact string.

**TC** If you want to search for specific hexadecimal values check the “Hex search” box, and enter the search string as a sequence of 2-digit hexadecimal numbers separated by spaces, for example 41 63 65 (these are the ASCII values for the string “Ace”; see the online help for a complete list of standard ASCII codes). Hexadecimal searches are case-sensitive, and search for exactly the string you enter.

You can use extended wildcards in the search string. For example, you can search for the string “to\*day” to find the next line which contains the word “to” followed by the word “day” later on the same line, or search for the numbers “101” or “401” with the search string “[14]01”. See page 94 for information on wildcards. If you begin the search string with a back-quote [```], or enclose it in back-quotes, wildcard characters in the string will be treated as normal text with no special wildcard meaning.

LIST saves the search string used by **F**, **N**, **Ctrl-F**, and **Ctrl-N** so you can LIST multiple files and search for the same string simply by pressing **N** in each file, or repeat your search the next time you use LIST. In Take Command, you can also select a previous string by using the drop-down arrow to the right of the string entry field (the **N** key and the **Next** button search for the top item in this drop-down list).

You can use the **/T** switch to specify search text for the first *file*. When you do so, LIST begins a search as soon as the file is loaded. Use **/I** to ignore wildcards in the initial search string, and **/R** to make the initial search go backwards from the end of the file. When you LIST multiple files with a single LIST command, these switches affect only the first file; they are ignored for the second and subsequent files.

You can use the **G** key to go to a specific line number in the file (or to a specified hexadecimal offset in hex mode). LIST numbers lines beginning with 1, unless ListRowStart is set to 0 in the *.INI* file. A new line is counted for every CR or LF character (LIST determines automatically which character is used for line breaks in each file), or when line length reaches 511 (4DOS) or 4095 (4NT / Take Command) characters, whichever comes first.

LIST normally allows long lines in the file to extend past the right edge of the screen. You can use the horizontal scrolling keys (see above) to view text that extends beyond the screen width. If you use the **W** command or **/W** switch to wrap the display, each line is wrapped when it reaches the right edge of the screen, and the horizontal scrolling keys are disabled.

To view text from the clipboard, use **CLIP:** as the file to be listed. CLIP: will not return any data unless the clipboard contains text. See page 88 for additional information on CLIP:, including details on when you can use the clipboard under 4DOS.

To view text from another command simply pipe the output of the command to LIST, for example:

```
[c:\] dir | list
```

Normally LIST will detect input from a pipe automatically, but if it does not, use **/S** to explicitly specify piped input. (In 4NT and Take Command your ability to navigate backward through the displayed output – *e.g.* with PgUp – may be limited when viewing a large amount of data through a pipe, due to the way Windows handles piped output.)

If you print the file which LIST is displaying, the print format will match the display format. If you have switched to hexadecimal or wrapped mode, that mode will be used for the printed output as well. If you print in wrapped mode, long lines will be wrapped at the width of the display. If you print in normal display mode without line wrap, long lines will be wrapped or truncated by the printer, not by LIST.

Take Command will display a standard print dialog which allows you to print selected text, the current page, or the entire file, and to select the printer. 4DOS and 4NT will ask you whether you wish to print the entire file or the current display page. (Printed output in 4DOS and 4NT normally goes to device LPT1. If you wish to send the printed output to another device, use the Printer directive (see page 224) in the *.INI* file).

#### **4NT, TC   *FTP Usage***

LIST can also display files on FTP servers. The URL must be enclosed in double quotes so the forward slashes won't be interpreted as switches. For example:

```
[c:\] list "ftp://jpsoft.com/index"
```

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see Using FTP Servers on page 113, and the IFTP command on page 382.

#### **4NT, TC   *NTFS File Streams***

LIST supports file streams on NTFS drives under Windows NT / 2000 / XP. You can list an individual stream by specifying the stream name, for example:

```
[c:\] list streamfile:s1
```

If no stream name is specified the file's primary data is displayed. See NTFS File Streams on page 18 for additional details.

### **❖   *Advanced Features***

If you specify a directory name instead of a filename as an argument, LIST will display each of the files in that directory.

Most of the LIST keystrokes can be reassigned with *.INI* file directives (see pages 230 and 235).

You can set the colors used by LIST with the ListColors directive in the *.INI* file, or the LIST Colors selection on the "Colors" tab of the configuration dialog. If ListColors is not used, the display will use the current default colors. If ListStatBarColors (in 4DOS and 4NT) is not used, the status bar will use the reverse of the LIST colors.

By default, LIST sets tab stops every 8 columns. You can change this behavior on the "Windows" tab on the configuration dialog, or with the TabStops .INI file directive (see page 227).

**4DOS**    ❖LIST normally writes text directly to the screen. If you have an unusual display adapter which does not support direct video output, see the OutputBIOS directive on page 237.

Options:   ❖**/A:** (Attribute select): Select only those files that have the specified attribute(s) set. See **Attribute Switches** on page 111 for information on the attributes which can follow **/A:**.

❖    **/H**(igh bit off): Strip the high bit from each character before displaying. This is useful when displaying files created by some word processors that turn on the high bit for formatting purposes. You can toggle this option on and off from within LIST with the **H** key or the **High** button on the Take Command tool bar.

**/I**(gnore wildcards): Only meaningful when used in conjunction with the **/T "text"** option. Direct LIST to interpret characters such as \*, ?, [, and ] as literal characters instead of wildcard characters. **/I** affects only the initial search started by **/T**, not subsequent searches started from within LIST.

**/I"text"**: Select files by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**/R**(everse): Only meaningful when used in conjunction with the **/T "text"** option. Directs LIST to search for text from the end of the file instead of from the beginning of the file. Using this switch can speed up searches for text that is normally near the end of the file, such as a signature. **/R** affects only the initial search started by **/T**, not subsequent searches started from within LIST.

**/S**(tandard input): Read from standard input rather than a file. This allows you to redirect command output and view it with LIST. Normally, LIST will detect input from a redirected command and adjust automatically. However, you may find circumstances when **/S** is required. For example, to use LIST to display the output of DIR you could use either of these commands:



```
[c:\] dir | list
[c:\] dir | list /s
```

**/T(ext):** Search for text in the first *file*. This option is the same as pressing **F**, but it allows you to specify the search text on the command line. The text must be contained in quotation marks if it contains spaces, punctuation, or wildcard characters. For example, to search for the string Take Command in the file *README.DOC*, you can use this command:

```
[c:\] list /t"Take Command" readme.doc
```

The search text may include wildcards and extended wildcards (see page 94). For example, to search for the words Hello and John on the same line in the file *LETTER.DAT*:

```
[c:\] list /t"Hello*John" letter.dat
```

When you LIST multiple files with a single LIST command, **/T** only initiates a search in the first file. It is ignored for the second and subsequent files. Also see **/I** and **/R**.

**/W(rap):** Wrap the text at the right edge of the screen. This option is useful when displaying files that don't have a carriage return at the end of each line. The horizontal scrolling keys do not work when the display is wrapped. You can toggle this option on and off from within LIST with the **W** key or the **Wrap** button on the Take Command tool bar.

**/X (hex mode):** Display the file in hexadecimal (hex) mode. This is useful when displaying files that contain non-text characters. Each byte is shown as a pair of hex characters. The corresponding text is displayed to the right of each line of hexadecimal data. You can toggle this mode on and off from within LIST with the **X** key or the **heX** button on the Take Command tool bar.

**LOADBTM**

(New)

**Purpose:** Switch a batch file to or from BTM mode.

**Format:** LOADBTM [ON | OFF]

**Usage:** 4DOS, 4NT, and Take Command recognize two kinds of batch files: *.BAT* or *.CMD*, and *.BTM*. Batch files executing in BTM mode are loaded into memory once rather than line-by-line, and run faster than in BAT or CMD mode. (However, BTM mode should not be used to load memory-resident programs in DOS, nor should BTM mode be used for self-modifying batch files.) Batch files automatically start in the mode indicated by their extension. For more information on *.BTM*, *.BAT*, and *.CMD* files, see page 125.

The LOADBTM command turns BTM mode on and off. It can be used to switch modes in either a *.BAT* / *.CMD* or *.BTM* file. If you use LOADBTM with no argument, it will display the current batch mode: LOADBTM ON or LOADBTM OFF.

Using LOADBTM to repeatedly switch modes within a batch file is not efficient. In most cases the speed gained by running some parts of the file in BTM mode will be more than offset by the speed lost through repeated loading of the file each time BTM mode is invoked.

LOADBTM can only be used within a batch file. It is most often used to convert a *.BAT* or *.CMD* file to BTM mode without changing its extension.

**LOCK**    *[4DOS]*    (Compatible)

**Purpose:**    Lock a disk drive to allow exclusive access under Windows 95 / 98 / ME.

**Format:**    LOCK [*drive*: ...]

*drive*: The drive or list of drives to lock.

      See also: UNLOCK.

**Usage:**    LOCK is only available when 4DOS is running under Windows 95 / 98 / ME. LOCK locks the specified drive(s) so that the 4DOS session, and the programs you run in it, have exclusive access to those drive(s). This allows you to run programs (such as disk maintenance utilities) which access the disk drive directly, without rebooting in DOS.

      If no drives are specified, 4DOS will attempt to lock all drives.

!    **USE EXTREME CAUTION** with programs which access the disk drive directly (and therefore require LOCK) and which were not written for Windows. If such programs are unaware of Windows' changes to the FAT directory structure, they can damage or destroy files, filenames, and system data. This damage can be caused whether you run such programs under Windows or from a DOS boot without the Windows GUI.

**LOG**

(New)

**Purpose:** Save a log of commands to a disk file.

**Format:** LOG [/E /H /W *file*] [ON | OFF | *text*]

***file*:** The name of the file to hold the log.

***text*:** An optional message that will be added to the log.

**/E**(rror log)

**/W**(rite to)

**/H**(istory log)

See also: HISTORY.

**Usage:** LOG keeps a record of all internal and external commands you use, whether they are executed from the prompt or from a batch file. Each entry includes the current system date and time and the session ID (for 4DOS, the shell level; for 4NT and Take Command the process ID, in hexadecimal), along with the actual command after any alias or variable expansion (see page 197 for more information on alias and variable expansion). You can use the log file as a record of your daily activities.

LOG with the /H option keeps a similar record called a “history log”. The history log records only commands entered at the prompt; it does not record batch file commands. In addition, the history log does not record the date and time for each command, and it records commands before aliases and variables are expanded.

The two logging options are independent. You can have both a regular log and a history log enabled simultaneously.

By default, LOG writes to the file *4DOSLOG*, *4NTLOG*, or *TC32LOG*, in the root directory of the boot drive. The corresponding default file names for the history log are *4DOSHLOG*, *4NTHLOG*, and *TC32HLOG*. You can set the default log file names from the “Startup” tab of the configuration dialog, or with the LogName and HistLogName directives in the *.INI* file.

Entering LOG or LOG /H with no parameters displays the name of the log file and the log status (ON or OFF):

```
[c:\] log
LOG (C:\TCMDLOG) is OFF
```

To enable or disable logging, add the word “ON” or “OFF” after the LOG command:

```
[c:\] log on
```

or

```
[c:\] log /h on
```

Entering LOG or LOG /H with *text* writes a message to the log file, even if logging is set OFF. This allows you to enter headers in the log file:

```
[c:\] log "Started work on the database system"
```

The LOG file format looks like this:

```
[date time][id] command
```

where the date and time are formatted according to the country code set for your system, and **id** is the session ID as described above.

The LOG /H output can be used as the basis for writing batch files. Start LOG /H, then execute the commands that you want the batch file to execute. When you are finished, turn LOG /H off. The resulting file can often be turned into a batch file that performs the same commands with little or no editing.

**Options:**    **/E**(rror log): This option saves all error messages to the error log.

**/H**(istory log): This option makes the other options on the command line (after the **/H**) apply to the history log. For example, to turn on history logging and write to the file C:\LOG\HLOG:

```
[c:\] log /h /w c:\log\hlog
```

**/W**(rite): This switch specifies a different filename for the LOG or LOG /H output. It also automatically performs a LOG ON command. For example, to turn logging on and write the log to C:\LOG\LOGFILE:

```
[c:\] log /w c:\log\logfile
```

Once you select a new file name with the LOG /W or LOG /H/W command, LOG will use that file until you issue another LOG /W or LOG /H/W command, or until you reboot your computer. Turning LOG or LOG /H off or on does not change the file name.

**MD / MKDIR**

(Enhanced)

**Purpose:** Create a subdirectory.

**Format:** MD [/N /S] *path...*

or

MKDIR [/N /S] *path...*

***path*:** The name of one or more directories to create.

/N(o update)

/S(ubdirectories)

See also: RD.

**Internet:** Can be used with FTP servers.

**Usage:** MD and MKDIR are synonyms. You can use either one.

MD creates a subdirectory anywhere in the directory tree. To create a subdirectory from the root, start the ***path*** with a backslash [\]. For example, this command creates a subdirectory called *MYDIR* in the root directory:

```
[c:\] md \mydir
```

If no path is given, the new subdirectory is created in the current directory. This example creates a subdirectory called *DIRTWO* in the current directory:

```
[c:\mydir] md dirtwo
```

To create a directory from the parent of the current directory (that is, to create a sibling of the current directory), start the pathname with two periods and a backslash [..\].

The operating system limits the permissible length of the full subdirectory name. See page 14 for details.

When creating a directory on an LFN drive, you must quote any ***path*** which contains white space or special characters. See page 15 for additional details.

If MD creates one or more directories, they will be added automatically to the extended directory search database unless the /N option is

specified. See page 83 for more information on extended directory searches.

- 4NT, TC** ❖ You can create directories on FTP servers. The URL must be enclosed in double quotes so the forward slashes won't be interpreted as switches. For example:

```
[c:\] md "ftp://jpsoft.com/data/index"
```

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see Using FTP Servers on page 113, and the IFTP command on page 382.

**Options:** /N(o update): Do not update the extended directory search database, *JPSTREE.IDX* (see page 83 for more details on extended searches). This is useful when creating a temporary directory which you do not want to appear in the extended search database.

/S(ubdirectories): Allows you to create more than one directory at a time. For example, if you need to create the directory *C:\ONE\TWO\THREE* and none of the named directories exist, you can use /S to have MD create all of the necessary subdirectories in a single command (without the /S, this command will fail because the parent directory *C:\ONE\TWO* does not exist):

```
[c:\] md /s \one\two\three
```

- 4NT, TC** For compatibility with *CMD.EXE*, /S becomes the default if you enable command processor extensions with the /X switch on the 4NT or Take Command startup command line. See your *Introduction and Installation Guide* for details on /X.

### MEMORY

(New)

**Purpose:** Display the amount and status of system RAM.

**Format:** **MEMORY**

**Usage:** MEMORY displays information about the RAM in your system. The display varies slightly depending on the specific command processor you are using; this example is from 4NT:

```
60 % Memory load

133,300,225 bytes total physical RAM
12,734,464 bytes available physical RAM

248,348,672 bytes total page file
194,465,792 bytes available page file

2,143,289,344 bytes total virtual RAM
2,129,985,536 bytes available virtual RAM

65,536 bytes total alias
30,180 bytes free

4,096 bytes total history
```

**4DOS** In 4DOS, MEMORY lists the amount of total RAM in your system and the amount available for applications after DOS, 4DOS, and memory-resident programs have been loaded; the amount of EMS expanded memory, XMS extended memory, and non-XMS extended memory; the HMA status; and the amount of memory 4DOS is using for environment variable space, alias space, and history space.

**4NT, TC** In 4NT and Take Command, MEMORY lists the percentage “memory load” as reported by Windows, the total and available physical RAM, the total and available page file size, the total and available virtual memory, the total and free alias space, and the total history space. The memory load is a figure returned by the operating system which gives an overall sense of memory utilization. It is not a precise indicator of system load or memory usage. The total page file figure shows the total number of bytes that can be stored in the file, but may not reflect the actual size of the current file on disk.



**MKLNK** [4NT, TC] (New)

**Purpose:** Create an NTFS hard link (Windows 2000 and Windows XP only).

**Format:** MKLNK [/D] *source target*.

***source:*** Name of the existing file.

***target:*** Name of the new directory entry to be created.

**/D**(elete a link)

**Usage:** NTFS 5 give you the ability to create “hard links”, which are multiple links to a single file. Creating a hard link causes the system to add the additional path to the file’s NTFS directory (unlike shortcuts, which actually create an additional file). Files can have multiple hard links, so a single file can appear in multiple directories, or with different names in the same directory.

Once you create a hard link, you can use it like any other file name. A file is deleted from the file system only after all links to it have been deleted.

Hard links must be created on the same NTFS volume as the file.

**Options:** **/D**(elete): Remove an existing hard link.

**MOVE**

(Enhanced)

**Purpose:** Move files to a new directory and drive.

**Format:** MOVE [/A:[-][+]rhsad] /C /D /E /F /G /H /I"text" /J /L /M /N /O /P /Q /R /S /T /U /V /W /Z] [*@file*] *source... destination*

***source:*** A file or list of files to move.

***destination:*** The new location for the files.

***@file:*** A text file containing the names of the ***source*** files to move, one per line (see page 109 for details).

/A: (Attribute select)	/N(othing)
/C(hanged)	/O (don't move if target exists)
/D(irectory)	/P(rompt)
/E (No error messages)	/Q(quiet)
/F(orce delete)	/R(eplace)
/G (display percent moved)	/S(ubdirectory tree)
/H(idden and system)	/T(otal)
/I"text" (match description)	/U(pdate)
/J (copy in restartable mode)	/V(erify)
/L (ASCII FTP transfer)	/W(ipe)
/M(odified files)	/Z (overwrite)

See also: COPY and RENAME.

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104). Ranges anywhere on the line apply to all ***source*** files. Use extended wildcards with caution on LFN volumes; see page 105 for details.

**Internet:** Can be used with FTP and HTTP servers.

**Usage:** The MOVE command moves one or more files from one directory to another, whether the directories are on the same drive or not. It has the same effect as copying the files to a new location and then deleting the originals. Like COPY and RENAME, MOVE works with single files, multiple files, and sets of files specified with an include list.

The simplest MOVE command moves a single ***source*** file to a new location and, optionally, gives it a new name. These two examples both move one file from drive C: to the root directory on drive A:

```
[c:\] move myfile.dat a:\
```

```
[c:\] move myfile.dat a:\savefile.dat
```

In both cases, *MYFILE.DAT* is removed from drive *C:* after it has been copied to drive *A:*. If a file called *MYFILE.DAT* in the first example, or *SAVEFILE.DAT* in the second example, already existed on drive *A:*, it would be overwritten. (This demonstrates the difference between **MOVE** and **RENAME**. **MOVE** will move files between drives and will overwrite the destination file if it exists; **RENAME** will not.)

When you move a single file, the **destination** can be a directory name or a file name. If it is a directory name, and you add a backslash [\] to the end of the name, **MOVE** will display an error message if the name does not refer to an existing directory. You can use this feature to keep **MOVE** from treating a mistyped **destination** directory name as a file name, and attempting to move the **source** file to that name.

If you **MOVE** multiple files, the **destination must** be a directory name. **MOVE** will move each file into the **destination** directory with its original name. If the **destination** is not a directory, **MOVE** will display an error message and exit. For example, if *C:\FINANCE\MYFILES* is not a directory, this command will display an error; otherwise, the files will be moved to that directory:

```
[c:\] move *.wks *.txt c:\finance\myfiles
```

The **/D** option can be used for single or multiple file moves; it checks to see whether the **destination** is a directory, and will prompt to see if you want to create the **destination** directory if it doesn't exist.

If **MOVE** creates one or more destination directories, they will be added automatically to the extended directory search database; see page 83 for details.

- ! Be careful when you use **MOVE** with the **SELECT** command. If you **SELECT** multiple files and the **destination** is not a directory (for example, because of a misspelling), **MOVE** will assume it is a file name. In this case each file will be moved in turn to the **destination** file, overwriting the previous file, and then the original will be erased before the next file is moved. At the end of the command, all of the original files will have been erased and only the last file will exist as the **destination** file.

You can avoid this problem by using square brackets with **SELECT** instead of parentheses (be sure that you don't allow the command line to get too long – watch the character count in the upper left corner

while you're selecting files). MOVE will then receive one list of files to move instead of a series of individual filenames, and it will detect the error and halt. You can also add a backslash [\] to the end of the **destination** name to ensure that it is the name of a subdirectory (see above).

### 4NT, TC **FTP Usage**

If you have appropriate permissions, you can move files to and from Internet FTP and HTTP URLs. The URL must be enclosed in double quotes so the forward slashes won't be interpreted as switches. For example:

```
[c:\] move "ftp://jpsoft.com/fl.txt" c:\text\
```

Files moved to or from FTP servers are normally transferred in binary mode. To perform an ASCII transfer use the /L switch. File descriptions are not copied when moving files to an Internet URL.

Wildcard characters like [\*] and [?] will be treated as wildcards in FTP URLs, but will be treated as normal characters in HTTP URLs.

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see Using FTP Servers on page 113, and the IFTP command on page 382.

### 4NT, TC **NTFS File Streams**

MOVE supports file streams on NTFS drives under Windows NT / 2000 / XP. You can move an individual stream by specifying the stream name, for example:

```
[c:\] move streamfile:s1 file2
```

If no stream name is specified the entire file is moved, including all streams.

- ! If you move a file to a drive or device which does not support streams, only the file's primary data is moved; any additional streams are not processed and their data will be lost.

See NTFS File Streams on page 18 for additional details.

### ❖ **Advanced Features and Options**

MOVE first attempts to rename the file(s), which is the fastest way to move files between subdirectories on the same drive. If that fails, (e.g., because the **destination** is on a different drive or already exists), MOVE will copy the file(s) and then delete the originals. If MOVE must physically copy the files and delete the originals, rather than renaming them (see above), then some disk space may be freed on the **source** drive. The free space may be the result of moving the files to another drive, or of overwriting a larger **destination** file with a smaller **source** file. MOVE displays the amount of disk space recovered unless the /Q option is used (see below). It does so by comparing the amount of free disk space before and after the MOVE command is executed. However, this amount may be incorrect if you are using a deletion tracking system which retains deleted files for later recovery, or if another program performs a file operation while the MOVE command is executed.

When physically copying files, MOVE preserves the hidden, system, and read-only attributes of the **source** files, and sets the archive attribute of the **destination** files. However, if the files can be renamed, and no copying is required, then the **source** file attributes are not changed.

- ! Use caution with the /A: and /H switches (both of which can allow MOVE to process hidden files) when you are physically moving files, and both the **source** and **destination** directories contain file descriptions. If the **source** file specification matches the description file name (normally *DESCRIPT.ION*), and you tell MOVE to process hidden files, the *DESCRIPT.ION* file itself will be moved, overwriting any existing file descriptions in the **destination** directory. For example, if the *C:\DATA* directory contains file descriptions, this command would overwrite any existing descriptions in the *D:\SAVE* directory:

```
[c:\data] move /h d*.* d:\save\
```

(If you remove the hidden attribute from the *DESCRIPT.ION* file the same caution applies even if you do not use /A: or /H, as *DESCRIPT.ION* is then treated like any other file.)

- Options: ❖ **A:** (Attribute select): Select only those files that have the specified attribute(s) set. See **Attribute Switches** on page 111 for information on the attributes which can follow /A:. See the cautionary note under

**Advanced Features and Options** above before using **/A**: when both *source* and *destination* directories contain file descriptions.

**/C**(hanged files): Move files only if the *destination* file exists and is older than the *source* (see also **/U**). This option is useful for updating the files in one directory from those in another without moving any newly-created files. (Before using **/C** in a network environment be sure to read the note under **/U**.)

**/D**(irectory): Requires that the *destination* be a directory. If the *destination* does not exist, MOVE will prompt to see if you want to create it. If the *destination* exists as a file, MOVE will fail with an "Access denied" error. Use this option to avoid having MOVE accidentally interpret your *destination* name as a file name when it's really a mistyped directory name.

**/E** (No error messages): Suppress all non-fatal error messages, such as "File Not Found." Fatal error messages, such as "Drive not ready," will still be displayed. This option is most useful in batch files and aliases.

**4DOS** **/F**: (Force delete) This option is only for use when running in an OS/2 DOS session. It forces deletion of the source file without saving it to the DELDIR directory (if DELDIR is not in use, **/F** has no effect). **/F** is only effective when MOVE must copy the source file(s) and delete the originals (i.e., if the destination is on a different drive or the destination file already exists). If the files are simply renamed, **/F** has no effect.

**/G**: Displays the percentage of the file moved. This is useful when copying large files across networks or via FTP to show whether the move is proceeding.

❖ **/H**(idden): Move all files, including hidden and system files. See the cautionary note under **Advanced Features and Options** above before using **/H** when both *source* and *destination* directories contain file descriptions.

**/I"text"**: Select *source* files by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**4NT, TC** **/J**: Move the file in restartable mode. The move progress is tracked in the destination file in case the move fails. The move can be restarted by specifying the same source and destination file names.

**4NT, TC** **/L**: Perform FTP transfers in ASCII mode, instead of the default binary mode.

**/M(odified files)**: Move only files that have the archive bit set. The archive bit will remain set after the MOVE; to clear it use ATTRIB (page 263).

**/N(othing)**: Do everything except actually move the file(s). This option is most useful for testing what a complex MOVE command will do. **/N** does **not** prevent creation of **destination** subdirectories when it is used with **/S**.

**/O**: Don't MOVE the source unless the target does not exist.

**/P(rompt)**: Prompt the user to confirm each move. Your options at the prompt are explained in detail on page 80.

**/Q(quiet)**: Don't display filenames, the total number of files moved, or the amount of disk space recovered, if any. This option is most often used in batch files. See also **/T**.

**/R(eplace)**: Prompt for a **Y** or **N** response before overwriting an existing **destination** file.

**/S(ubdirectories)**: Move an entire subdirectory tree to another location. MOVE will attempt to create the **destination** directories if they don't exist, and will remove empty subdirectories after the move. When **/D** is used with **/S**, you will be prompted if the first **destination** directory does not exist, but subdirectories below that will be created automatically by MOVE. If MOVE **/S** creates one or more destination directories, they will be added automatically to the extended directory search database; see page 83 for details.

If you attempt to use **/S** to move a subdirectory tree into part of itself, MOVE will detect the resulting infinite loop, display an error message, and exit.

Do not use **/S** with @file lists; see page 109 for details.

**/T(otal):** Don't display filenames as they are moved, but display the total number of files deleted and the amount of free disk space recovered, if any.

**/U(pdate):** Move each **source** file only if it is newer than a matching **destination** file or if a matching **destination** file does not exist (also see **/C**). This option is useful for moving new or changed files from one directory to another.

The time comparisons used with **/U** may not always work reliably across a network, including on FTP servers, due to differences in time zone and in the file time storage method between the local and remote systems. Be sure to do some non-destructive testing (*e.g.* with **/N**) before depending on this option to yield the results you want in a network environment.

**/V(erify):** Verify each disk write. This is the same as executing the VERIFY ON command, but is only active during the MOVE. **/V** does **not** read back the file and compare its contents with what was written; it only verifies that the data written to disk is physically readable.

**/W(ipe):** If the MOVE is to a different drive, after the move overwrite the source file with 0's before deleting it (like DEL **/W**).

- !** **/Z:** Overwrite read-only **destination** files. Without this option, MOVE will fail with an "Access denied" error if the **destination** file has its read-only attribute set. This option allows MOVE to overwrite read-only files without generating any errors.



**MSGBOX**    [4NT, TC]

(New)

**Purpose:**    Display a message box and collect the user's response.

**Format:**    MSGBOX [/1 /2 /3 /I /Q /S /Tn /W] OK | OKCANCEL | YESNO | YESNOCANCEL ["*title*"] *prompt*

***title*:** Text for the title bar of the message box.

***prompt*:** Text that will appear inside the message box.

/1 (first button)                      /Q(uestion)

/2 (second button)                    /S(top)

/3 (third button)                     /T(imeout)

/I(con)                                /W(arning)

See also: INKEY, INPUT, and QUERYBOX.

**Usage:**    MSGBOX can display one of four kinds of message boxes and wait for the user's response. You can use ***title*** and ***prompt*** to display any text you wish. 4NT and Take Command automatically size and center the box on the screen.

The message box may have 1, 2, or 3 response buttons. The command MSGBOX OK creates a single-button box; the user must simply acknowledge the prompt text.

The OKCANCEL and YESNO forms have 2 buttons each. The YESNOCANCEL form has 3 buttons. The button the user chooses is returned in the variable %\_?. Be sure to save the return value in another variable or test it immediately, because the value of %\_? changes with every internal command.

The following list shows the value returned for each selection:

<b>Yes</b>	10	<b>No</b>	11
<b>OK</b>	10	<b>Cancel</b>	12

If you exit the message box without selecting one of these options, MSGBOX will set %\_? to 0. If there is an error in the MSGBOX command itself, %\_? will be set as described on page 155. If /T is used and the time limit expires, %\_? will be set to 20.

For example, to display a Yes or No message box and take action depending on the result, you could use commands like this:

```
msgbox yesno "Copy" Copy all files to A:?  
if %_? == 10 copy *.* a:
```

MSGBOX creates a popup dialog box. If you prefer to retrieve input from inside the command line window, see INKEY and INPUT.

**Option:**     **/1:** The first button is the default.

**/2:** The second button is the default.

**/3:** The third button is the default.

**/I(con):** Display an icon consisting of a lower case “i” in a circle in the message box.

**/Q(uestion):** Display a question mark icon in the message box.

**/S(top):** Display a stop sign icon in the message box.

**/W(arning):** Display an exclamation point icon in the message box.

**/Tn:** MSGBOX will wait a maximum of “n” seconds for a response. If the time limit expires, %\_? will be set to 20. This switch may not work properly with a **YESNO** message box, due to a Windows bug. This bug does not affect any of the other options.

**ON**

(New)

**Purpose:** Execute a command in a batch file when a specific condition occurs.

**Format:** ON BREAK [*command*]

or

ON ERROR [*command*]

or

ON ERRORMSG [*command*]

**Usage:** ON can only be used in batch files.

ON sets a “watchdog” that remains in effect for the duration of the current batch file. Whenever a BREAK or ERROR condition occurs after ON has been executed, the corresponding ***command*** is automatically executed.

ON BREAK will execute the ***command*** if the user presses **Ctrl-C** or **Ctrl-Break**. ON ERROR and ON ERRORMSG will execute the ***command*** after any critical error (see page 115), operating system error, or internal command error (such as a command that fails to process any files, or the use of an invalid option).

ON ERROR executes the ***command*** immediately after the error occurs, without displaying any command processor error message (operating system errors may still be displayed). ON ERRORMSG displays the appropriate error message, then executes the ***command***. If both are specified, ON ERROR will take precedence, and ON ERRORMSG will be ignored. The remainder of this section discusses both settings, using the term “ON ERROR[MSG]”.

ON BREAK and ON ERROR[MSG] are independent of each other. You can use either one, or both, in any batch file.

Each time ON BREAK or ON ERROR[MSG] is used, it defines a new ***command*** to be executed for a break or error, and any old ***command*** is discarded. If you use ON BREAK or ON ERROR[MSG] with no following ***command***, that type of error handling is disabled. Error handling is also automatically disabled when the batch file exits.

ON BREAK and ON ERROR[MSG] only affect the current batch file. If you CALL another file, the first file's error handling is suspended, and the CALLED file must define its own error handling. When control returns to the first batch file, its error handling is reactivated.

The **command** can be any command that can be used on a batch file line by itself. Frequently, it is a GOTO or GOSUB command. For example, the following fragment traps any user attempt to end the batch file by pressing **Ctrl-C** or **Ctrl-Break**. It scolds the user for trying to end the batch file, then continues:

```
on break gosub gotabreak
do i = 1 to 1000
    echo %i
enddo
quit
:gotabreak
echo Hey! Stop that!!
return
```

You can use a command group (see page 117) as the **command** if you want to execute multiple commands, for example:

```
on break (echo Oops, got a break! & quit)
```

ON BREAK and ON ERROR[MSG] always assume that you want to continue executing the batch file. After the **command** is executed, control automatically returns to the command after the one that was interrupted by the break or error. To avoid continuing the batch file after a break or error the **command** can transfer control with GOTO, end the batch file with QUIT or CANCEL, or start another batch file (without CALLing it).

When handling an error condition with ON ERROR[MSG], you may find it useful to use internal variables (see page 153), particularly %\_? and %\_SYSERR, to help determine the cause of the error.

The ON ERROR[MSG] command will **not** be invoked if an error occurs while reading or writing redirected input, output, or a pipe.

- ! If a break or error occurs while the **command** specified in ON BREAK or ON ERROR[MSG] is executing, the command will be restarted. This means you must use caution to avoid (or handle) any possible errors in the commands invoked by ON ERROR[MSG], since such errors can cause an infinite loop and / or a stack overflow (in 4DOS).

**OPTION**

(New)

**Purpose:** Modify the command processor configuration.

**Format:** OPTION [/optname=value ...][@infile]

**optname:** An INI file directive to set or modify.

**value:** A new value for that directive.

**infile:** An INI file to read (4NT and Take Command).

See also: The *.INI* file on page 203 and the Take Command configuration dialog on page 50.

**Usage:** OPTION displays a set of dialogs (in 4DOS) or a property sheet (in 4NT and Take Command) which allows you to modify most of the configuration options stored in the *.INI* file.

**4DOS** The normal way to exit from the dialogs is to select **Save** on the **Exit** menu. This will save your changes for use in the current session, and also save them to the *.INI* file (along with any prior changes made in the current session) for use in future sessions. You can also select **OK** to use your changes in the current session only, or **Cancel** to discard the changes.

**4NT, TC** When you exit from the dialogs or property sheet using the **OK** button, changes are saved in the *.INI* file and will be in effect for in the current session and all future sessions. To discard the changes, use the **Cancel** button.

In most cases, changes you make in the **Startup** section or tab of the OPTION dialogs will only take effect when you reboot your system (under DOS), or restart your command processor (under Windows). Other changes take effect as soon as you exit the dialogs with **Save** or **OK**. However, not all option changes will appear immediately, even if they have taken effect. For example, some color changes will only appear after a CLS command.

OPTION handles most standard *.INI* file settings. More advanced settings, including all those listed under Key Mapping Directives (page 230) and Advanced Directives (page 235) cannot be modified with the OPTION dialogs. These settings must be inserted or modified in the *.INI* file manually. For more details see Chapter 5.

OPTION does not preserve comments when saving modified settings in the *.INI* file. To be sure *.INI* file comments are preserved, put them on separate lines in the file (see page 206 for details).

Under 4DOS, OPTION runs the external program *OPTION.EXE* to display the dialogs or notebook. If the command processor cannot find this file it will display an error message. This program must be run with the OPTION command, and will not work if you invoke it directly. Under 4NT and Take Command, OPTION runs internally, and no external program is required.

### ❖ *Setting Individual Options*

If you follow the OPTION command with one or more sequences of a double slash mark `[//]` followed by an **option=value** setting, the OPTION dialogs will not appear. Instead, the new settings will take effect immediately, and will be in effect for the current session only. This example turns off batch file echo and changes the input colors to bright cyan on black (enter this all on one line):

```
[c:\] option //BatchEcho = No //InputColors = bri cya  
on bla
```

Option values may contain white space. However, you cannot enter an option value which contains the `“//”` string.

This feature is useful for testing settings quickly, and in aliases or batch files which depend on certain options being in effect. This use of OPTION is less efficient in 4DOS than it is in 4NT or Take Command, because 4DOS must load the external *OPTION.EXE* program for each **OPTION //...** command. 4NT and Take Command process `// options` internally.

Changes made with `//` are temporary. They will not be saved in the *.INI* file, even if you subsequently load the option dialogs and select **Save** or **OK**.

**PATH**

(Enhanced)

**Purpose:** Display or alter the list of directories that the command processor will search for executable files, batch files, and files with executable extensions that are not in the current directory.

**Format:** PATH [*directory*[:*directory*...]]

***directory*:** The full name of a directory to include in the path setting.

See also: ESET and SET.

**Usage:** When the command processor is asked to execute an external command, it first looks for the file in the current directory. If it fails to find an executable file in the current directory, it will search each of the *directories* specified in the PATH setting. See page 19 for more details on how the path is searched.

**TC** Under Windows 95 / 98 / ME, Take Command searches the \WINDOWS and \WINDOWS\SYSTEM directories, in that order, after the current directory and before any directories listed in your search path. Under Windows NT / 2000 / XP, the order is reversed, and Take Command searches the \WINDOWS\SYSTEM32 directory followed by the \WINDOWS directory before any directories listed in your search path. (The actual directory names may be different on your system. Take Command will determine the correct names for the “Windows” and “Windows System” directories and use them.) These search procedures conform to the traditional search sequences used under each version of the Windows operating system.

For example, after the following PATH command, the command processor will search for an executable file in four directories: the current directory, then the root directory on drive C, then the *BIN* subdirectory on C, and then the *UTIL* subdirectory on C:

```
[c:\] path c:\;c:\bin;c:\util
```

(In Take Command, the \WINDOWS and \WINDOWS\SYSTEM directories will be searched as well.)

The list of ***directories*** to search is stored as an environment string, and can also be set or viewed with SET, and edited with ESET.

The PATHEXT environment variable, and the related PathExt .INI directive, can be used to select the extensions to look for when

searching the PATH for an executable file. See PATHEXT on page 152 for additional details.

Directory names in the path must be separated by semicolons [;]. Each directory name is shifted to upper case to maintain compatibility with programs which can only recognize upper case directory names in the path. If you modify your path with the SET or ESET command, you may include directory names in lower case. These may cause trouble with some programs, which assume that all path entries have been shifted to upper case.

If you enter PATH with no parameters, the current path is displayed:

```
[c:\] path
PATH=C:\;C:\BIN;C:\UTIL
```

Entering PATH and a semicolon clears the search path so that only the current directory is searched for executable files (this is the default at system startup).

Some applications also use the PATH to search for their data files.

- ❖ If you include an explicit file extension on a command name (for example, *WP.EXE*), the search will find files with that name and extension in the current directory and every directory in the path. It will not locate other executable files with the same base name (e.g., *WP.COM*).
- ❖ If you have an entry in the path which consists of a single period [.] , the current directory will **not** be searched first, but instead will be searched when the command processor reaches the “.” in the path. This allows you to delay the search of the current directory for executable files and files with executable extensions. In rare cases, this feature may not be compatible with applications which use the path to find their files; if you experience a problem, you will have to remove the “.” from the path while using any such application.

- 4DOS**
- ❖ In normal use, 4DOS can create a path as long as 506 characters (the command-line limit is 511 characters, and “PATH ” takes five). However, some DOS applications expect a path no longer than the traditional limit of 123 characters. If you extend your path beyond this limit and experience problems with applications, see the **Compatibility** section of the online help for tips on resolving the difficulty.



- ❖ To create a path longer than the command-line length limit, use **PATH** repeatedly to append additional directories to the path:

```
path [first list of directories]
path %path:[second list of directories]
...
```

You cannot use this method to extend the path beyond 506 characters in 4DOS or 4090 characters in 4NT and Take Command (the internal buffer limits, with room for “PATH ”). It is usually more efficient to use aliases to load application programs than to create a long **PATH**. See **ALIAS** on page 250 for details.

If you specify an invalid directory in the path, it will be skipped and the search will continue with the next directory in the path.

### **PAUSE**

(Enhanced)

**Purpose:** Suspend batch file or alias execution.

**Format:** PAUSE [*text*]

***text*:** The message to be displayed as a user prompt.

**Usage:** A PAUSE command will suspend execution of a batch file or alias, giving you the opportunity to change disks, turn on the printer, etc.

PAUSE waits for any key to be pressed and then continues execution. You can specify the ***text*** that PAUSE displays while it waits for a keystroke, or let it use the default message:

```
Press any key when ready...
```

For example, the following batch file fragment prompts the user before erasing files (the PAUSE command should be entered on one line):

```
pause Press Ctrl-C to abort, any other key to  
erase all .LST files  
erase *.lst
```

If you press **Ctrl-C** or **Ctrl-Break** while PAUSE is waiting for a key, execution of an alias will be terminated, and execution of a batch file will be suspended while you are asked whether to cancel the batch job (see page 131). In a batch file you can handle **Ctrl-C** and **Ctrl-Break** yourself with the ON BREAK command (see page 425).

## **PLAYAVI**    [4NT, TC]

(New)

**Purpose:**    Play Windows .AVI (video clip) files.

**Format:**    PLAYAVI [/A /C /S] *filename*

***filename:*** The file to play.

**/A(synchronous)**

**/S(ynchronous)**

**/C(enter)**

**Usage:**    PLAYAVI "plays" an .AVI or Windows video clip file. For example, to play the DragDrop demo file, you can use the command:

```
[c:\] playavi c:\windows\help\dragdrop.avi
```

By default, PLAYAVI operates in synchronous mode, which means the command processor waits for the AVI file to complete and its window to close before continuing with the next command in a batch file or alias, or prompting you for a new command. You can change this default behavior with the /A switch, described below.

**Option:**    **/A(synchronous):** Plays the .AVI file in asynchronous mode. Control returns to the command processor prompt immediately for a new command or to execute the next command in the current batch file or alias.

**/C(enter):** Displays the AVI viewer in the center of the command processor window. Without this option, the viewer appears in the upper-left corner of the window.

**/S(ynchronous):** Plays the .AVI file in synchronous mode (this is the default). The command processor pauses until the file has finished playing and its window closes.

**PLAYSOUND** [4NT, TC]

(New)

**Purpose:** Play .WAV, Midi, and other sound files.

**Format:** PLAYSOUND [/A /S] *filename*

***filename*:** The audio file to play.

**/A(synchronous)**

**/S(ynchronous)**

**Usage:** PLAYSOUND "plays" .WAV, Midi (.MID) and other types of sound files for which Windows has an appropriate decoder or "codec" installed. It determines the file type automatically from its contents, not its file extension, so it can play sound files which have non-standard file extensions.

By default, PLAYSOUND operates in synchronous mode, which means the command processor waits for the sound file to complete and its window to close before continuing with the next command in a batch file or alias, or prompting you for a new command. You can change this default behavior with the /A switch, described below.

You can cancel the playing of a synchronous sound file by pressing Ctrl-C or Ctrl-Break while it is playing.

**Option:** /A(synchronous): Plays the sound file in asynchronous mode. Control returns to the command processor prompt immediately for a new command or to execute the next command in the current batch file or alias.

/S(ynchronous): Plays the sound file in synchronous mode (this is the default). The command processor pauses until the file has finished playing and its window closes.

**POPD**

(New)

**Purpose:** Return to the disk drive and directory at the top of the directory stack.

**Format:** POPD [\*]

See also: DIRS, PUSHHD, and Directory Navigation on page 81

**Usage:** Each time you use the PUSHHD command, it saves the current disk drive and directory on the internal directory stack. POPD restores the last drive and directory that was saved with PUSHHD and removes that entry from the stack. You can use these commands together to change directories, perform some work, and return to the starting drive and directory.

Directory changes made with POPD are recorded in the directory history list and can be displayed in the directory history window (see page 74). See the section on Directory Navigation beginning on page 81 for details on directory navigation features.

This example saves and changes the current disk drive and directory with PUSHHD, and then restores it. The current directory is shown in the prompt:

```
[c:\] pushd d:\database\test
[d:\database\test] pushd c:\wordp\memos
[c:\wordp\memos] pushd a:\123
[a:\123] popd
[c:\wordp\memos] popd
[d:\database\test] popd
[c:\]
```

You can use the DIRS command to see the complete list of saved drives and directories (the directory stack).

The POPD command followed by an asterisk [\*] clears the directory stack without changing the current drive and directory.

- ❖ If the directory on the top of the stack is not on the current drive, POPD will switch to the drive and directory on the top of the stack without changing the default directory on the current drive.

**PRINT** [4NT, TC]

(New)

**Purpose:** Print the specified file(s) using the application associated with each file's extension.

**Format:** PRINT *filename ...*

**Files:** Supports multiple file names.

**Usage:** Except for plain text files, few Windows files can be successfully printed without sending them to an associated application for interpretation and formatting. The PRINT command does just that. Using the extension for each file you want to print, it begins by determining if a Print action has been defined for that file type. If so, it executes the Print action and sends the file to the application for processing.

For example, if you use the command:

```
[c:\] print myletter.doc
```

PRINT looks up the Print command for .DOC files in the registry (and, on many computers, will find that it is associated either with WordPad or Microsoft Word). It will execute the associated program and send it the file along with the necessary command to print the file, and then quit.

If PRINT cannot find a Print command for a file, it displays an error message and skips that file. If there are additional files in the list you gave it to print, it will go on to the next file in the list.

PRINT depends on proper settings in the Windows registry and proper behavior of the program associated with each file type in order to print the file. If the registry entries or the application associated with a particular file type are not configured correctly, PRINT may not work as you expect.

## **PROMPT**

(Enhanced)

**Purpose:** Change the command-line prompt.

**Format:** PROMPT [*text*]

***text:*** Text to be used as the new command-line prompt.

**Usage:** You can change and customize the command-line prompt at any time. The prompt can include normal text and system information such as the current drive and directory, the time and date, and the amount of memory available. You can create an informal “Hello, Bob!” prompt or an official-looking prompt full of impressive information.

The prompt ***text*** can contain special commands in the form \$?, where ? is one of the characters listed below:

- b** The vertical bar character [|].
- c** The open parenthesis [(].
- d** Current date, in the format: *Fri 12-14-01* (the month, day, and year are formatted according to your current country settings).
- D** Current date, in the format: *Fri Dec 14, 2001*.
- e** The ASCII ESC character (decimal 27).
- f** The close parenthesis [)].
- g** The > character.
- h** Backspace over the previous character.
- j** Current date, in the format *yyyy-mm-dd*.
- l** The < character.
- m** Time in hours and minutes using 24-hour format.
- M** Time in hours and minutes using the default country format.
- n** Current drive letter.
- p** Current drive and directory (lower case).
- P** Current drive and directory (upper case on drives which do not support long filenames; directory names shown in mixed case as stored on the disk on LFN drives).
- q** The = character.
- r** The numeric exit code of the last external command.

- s** The space character.
- t** Current 24-hour time, in the format *hh:mm:ss*.
- T** Current 12-hour time, in the format *hh:mm:ss[a/p]*.
- u** The current user. (In 4DOS, it displays the value of the environment variable LOGINNAME.)
- v** Operating system version number, in the format *3.10*.
- w** Current directory, in a shortened format. If the current directory is the root or a first-level subdirectory, it is displayed as-is. If it is second level or deeper, the path is truncated (i.e., "c:\...\config"). This does not work with UNC names. \$W and \$w behave like \$P and \$p for displaying upper/lower case.
- xd:** Current directory on drive *d:*, in lower case, including the drive letter. (Uses the actual case of the directory name as stored on the disk for LFN drives; see page 15 for details.)
- Xd:** Current directory on drive *d:*, in upper case, including the drive letter.
- z** Current shell nesting level; the primary command processor is shell 0.
- +** Display one + character for each directory on the PUSHD directory stack.
- \$** The \$ character.
- \_** CR/LF (go to beginning of a new line).

For example, to set the prompt to the current date and time, with a ">" at the end:

```
[c:\] prompt $d $t $g
Fri Jun 8, 2001 10:29:42 >
```

The prompt can be set in *4START* or *TCSTART* (see page 132), or in any batch file that runs when the command processor starts.

The default 4DOS prompt is **\$n\$g** (drive name followed by ">") on floppy drives, and **\$p\$g** (current drive and directory followed by ">") on all other drives. The 4NT and Take Command default prompt is **[ \$n ]** (drive name in square brackets) on floppy drives, and **[ \$p ]** (current drive and directory in square brackets) on all other drives.

If you enter PROMPT with no arguments, the prompt will be reset to its default value. The PROMPT command sets the environment



variable **PROMPT**, so to view the current prompt setting use the command:

```
[c:\] set prompt
```

(If the prompt is not set at all, the **PROMPT** environment variable will not be used, in which case the **SET** command above will give a “Not in environment” error.)

- ❖ Along with literal text and special characters, you can include the text of any environment variable, internal variable, or variable function (see pages 153 and 165) in a prompt. For example, if you want to include the amount of free memory in the command prompt, plus the current drive and directory, you could use this command:

```
[c:\] prompt [(%%@dosmem[K]K) $p]  
[(31043K) c:\data]
```

Notice that the **@DOSMEM** function is shown with two leading percent signs [%]. If you used only one percent sign, the **@DOSMEM** function would be expanded once when the **PROMPT** command was executed, instead of every time the prompt is displayed. As a result, the amount of memory would never change from the value it had when you entered the **PROMPT** command. You can also use back-quotes to delay expanding the variable function until the prompt is displayed:

```
[c:\] prompt `[(@dosmem[K]K) $p]`
```

- ❖ You can use this feature along with the **@EXEC** variable function (see page 174) to create a complex prompt which not only displays information but executes commands. For example, to execute an alias which checks battery status each time the prompt is displayed (enter the alias on one line):

```
[c:\] alias cbatt `if %_apmlife lt 30 beep 440 4 880 4  
440 4 880 4`  
[c:\] prompt `%@exec[@cbatt]$p$g`
```

**PROMPT** will also look for the environment variable **TITLEPROMPT**, and use its contents to build a prompt in the Windows title bar. **TITLEPROMPT** can contain any of the \$ metacharacters or variables that are valid in **PROMPT**. Note that **4DOS** is limited to 79 characters; **4NT** and **TCMD** are limited to around 127 characters.

- ❖ If you have an ANSI-compatible driver installed under 4DOS, or you have enabled 4NT or Take Command's built-in ANSI support, you can include ANSI escape sequences in the PROMPT *text*. (See your online help for information on ANSI escape sequences). This example uses ANSI sequences to set a prompt that displays the shell level, date, time and path in color on the top line of the screen (enter the command as one line):

```
[c:\] prompt $e[s$e[1;1f$e[41;1;37m$e[K[$z] $d  
Time: $t$h$h$h$h Path: $p$e[u$e[0;32m$n$g
```

- ❖ A few older batch files use the PROMPT command to transmit *ANSI.SYS* control sequences to the screen (for example, to redefine function keys). This technique will not work with 4DOS, because it doesn't display a prompt within batch files; hence, the characters in the PROMPT string are never sent to *ANSI.SYS*. To send ANSI sequences in 4DOS, use the ECHO command, substituting an escape character followed by an **e** for **\$e** in the PROMPT string (see page 119 for more information).

**4DOS** ❖ You may find it helpful to define a different prompt in secondary shells, perhaps including **\$z** in the prompt to display the shell level. To do so, place a PROMPT command in your *4START* file and use IF or IFF statements to set the appropriate prompt for different shells.

## **PUSHD**

(New)

**Purpose:** Save the current disk drive and directory, optionally changing to a new drive and directory.

**Format:** PUSHD [*path*]

***path*:** The name of the new default drive and directory.

See also: DIRS, POPD and Directory Navigation on page 81.

**Usage:** PUSHD saves the current drive and directory on a “last in, first out” directory stack. The POPD command returns to the last drive and directory that was saved by PUSHD. You can use these commands together to change directories, perform some work, and return to the starting drive and directory. The DIRS command displays the contents of the directory stack.

To save the current drive and directory, without changing directories, use the PUSHD command by itself, with no ***path***.

If a ***path*** is specified as part of the PUSHD command, the current drive and directory are saved and PUSHD changes to the specified drive and directory. If the ***path*** includes a drive letter, PUSHD changes to the specified directory on the new drive without changing the current directory on the original drive.

This example saves the current directory and changes to C:\WORDP\MEMOS, then returns to the original directory:

```
[c:\] pushd \wordp\memos
[c:\wordp\memos] popd
[c:\]
```

When you use PUSHD to change to a directory on an LFN drive, you must quote the ***path*** name if it contains white space or special characters. See page 15 for additional details.

PUSHD can also change to a network drive and directory specified with a UNC name (see page 13 for information on UNC names).

If PUSHD cannot change to the directory you have specified it will attempt to search the CDPATH (see page 86) and the extended directory search database (see page 83). You can also use wildcards in the ***path*** to force an extended directory search (see page 85). Read the

section on Directory Navigation beginning on page 81 for complete details on these and other directory navigation features.

Directory changes made with PUSHD are also recorded in the directory history list and can be displayed in the directory history window (see page 74).

- ❖ The directory stack can hold up to 511 characters, or between 20 and 40 typical entries (depending on the length of the names). If you exceed this limit, the oldest entry is removed before adding a new entry.

**QUERYBOX**    [4NT, TC]

(New)

**Purpose:**    Use a dialog box to get an input string from the user and save it in an environment variable.

**Format:**    **QUERYBOX** [/D /E /Ln /Tn] ["title"] *prompt* %%varname

**title:** Text for the title bar of the dialog box.

**prompt:** Text that will appear inside the dialog box.

**varname** : Variable name where the input will be saved

**/D**(igits only)

**/L** (maximum Length)

**/E**(dit existing value)

**/T**(imeout)

See also: INKEY, INPUT, and MSGBOX.

**Usage:**    **QUERYBOX** displays a popup window with a prompt, an optional title, and an input field. Then it waits for your entry, and places any characters you type into an environment variable. **QUERYBOX** is normally used in batch files and aliases to get string input. If you prefer to work within the command line window, see the **INKEY** and **INPUT** commands (page xxx).

Standard command-line editing keys may be used to edit the input string as it is entered. All characters entered up to, but not including, the carriage return are stored in the variable.

For example, to prompt for the variable **NAME**:

```
querybox "File Name" Enter a name: %%name
```

If you press **Ctrl-C** or **Ctrl-Break** while **QUERYBOX** is waiting for input, aliases will be terminated, and batch files will be suspended while you are asked whether to cancel the batch job (see page 131).

**Options:**    **/D**(igits): Only accepts numeric values.

**/E**(dit): Allows you to edit an existing value. If there is no existing value for **varname**, **QUERYBOX** allows you to enter a new value.

**/Ln** (Length): Sets the maximum number of characters which **QUERYBOX** will accept to "n".

**/Tn**: Wait for a maximum of "n" seconds for a response.

### QUIT

(New)

**Purpose:** Terminate the current batch file.

**Format:** QUIT [*value*]

***value*:** The numeric exit code to return to the command processor or to the previous batch file.

See also: CANCEL and EXIT.

**Usage:** QUIT provides a simple way to exit a batch file before reaching the end of the file. If you QUIT a batch file called from another batch file, you will be returned to the previous file at the line following the original CALL.

This example batch file fragment checks to see if the user entered “quit” and exits if true:

```
input Enter your choice : %%option
if "%option" == "quit" quit
```

QUIT only ends the current batch file. To end all batch file processing, use the CANCEL command.

- ❖ If you specify a ***value***, QUIT will set the ERRORLEVEL or exit code to that value. For information on exit codes see the IF command, and the %? variable on page 155.
- ❖ You can also use QUIT to terminate an alias. If you QUIT an alias while inside a batch file, QUIT will end both the alias and the batch file and return you to the command prompt or to the calling batch file.

**RD / RMDIR**

(Enhanced)

**Purpose:** Remove one or more subdirectories.

**Format:** RD [/I"text" /K /Q /R /S] [@file] *path...*

or

RMDIR [/I"text" /S] [@file] *path...*

***path:*** The name of a subdirectory to remove.

***@file:*** A text file containing the names of the directories to remove, one per line (see page 109 for details).

**/I** (match descriptions)

**/R**(ecycle bin)

**/K** (no undelete)

**/S**(ubdirectories)

**/Q**(uiet)

See also: MD.

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104).

**Internet:** Can be used with FTP servers.

**Usage:** RD and RMDIR are synonyms. You can use either one.

RD removes directories from the directory tree. For example, to remove the subdirectory *MEMOS* from the subdirectory *WP*:

```
[c:\] rd \wp\memos
```

Before using RD, you must delete all files and subdirectories (and their files) in the ***path*** you want to remove. Remember to remove hidden and read-only files as well as normal files (you can use DEL /Z to delete hidden and read-only files).

You can use wildcards in the ***path***. When removing a directory on an LFN drive, you must quote any ***path*** which contains white space or special characters (see page 15 for details).

If RD deletes one or more directories, they will be deleted from the extended directory search database; see page 83 for details.

You cannot remove the root directory, the current directory (.), any directory above the current directory in the directory tree, or any directory in use by another process.

**4NT, TC** RD will delete hidden directories, for compatibility with CMD.EXE.

**4NT, TC** ❖ You can remove directories on FTP servers. The URL must be enclosed in double quotes so the forward slashes won't be interpreted as switches. For example:

```
[c:\] rd "ftp://ftp.somedomain.com/data"
```

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see Using FTP Servers on page 113, and the IFTP command on page 382.

**Options:** **/I"text":** Select directories by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**4NT, TC** ! ❖ **/S(subdirectories):** This option is included for compatibility with *CMD.EXE*, and **must be used with extreme caution!**

**4NT, TC** It deletes all files (including hidden and system files) in the named directory and all of its subdirectories, then removes all subdirectories. **It can erase all files on a drive with a single command.**

If you use **/S** you will have three additional switches available: **/Q** will not prompt whether to delete the directory tree, **/R** will send the deleted files to the Recycle Bin, and **/K** will not, regardless of the RecycleBin *.INI* setting. For more details see the corresponding **/K** and **/R** options for DEL (page 292).

Do not use **/S** with @file lists; see page 109 for details.



## REBOOT

(New)

**Purpose:** Do a warm or cold system reboot.

**Format:** REBOOT [/C /L /S /V]

/C(old reboot)

/S(hutdown)

/L(ogoff)

/V(erify)

/P(oweroff)

❖ **Usage:** REBOOT will log off or shut down the operating system, or completely restart your computer. It normally performs a warm reboot, which is comparable to pressing Ctrl-Alt-Delete under DOS, or to a shutdown and restart under Windows. Under DOS, REBOOT can also perform a cold reboot, which is comparable to turning the power off and back on or pressing the reset button.

A reboot is necessary to activate any changes to your *CONFIG.SYS* under DOS or Windows 95 / 98, and may also be used if you wish to restart DOS with an altered *4START* or *AUTOEXEC.BAT* file.

REBOOT defaults to performing a warm boot, with no prompting. The following example prompts you to verify the reboot, then does a warm boot:

```
[c:\] reboot /v
```

4NT and Take Command issue the standard commands to shut down other applications and Windows before rebooting.

**4DOS** ! Under DOS, some system BIOSes, memory managers, multitaskers, or memory-resident programs (TSRs) may intercept attempts to reboot your system and defeat them entirely, convert a cold boot request to a warm boot or vice versa, or in very rare cases, hang the system – requiring a reboot! As a result you may need to experiment with which reboot options work best for your system hardware and software configuration, and under rare circumstances REBOOT may not be usable on your system. REBOOT will **not** work properly from 4DOS while running under Windows (at most, it will shut down the 4DOS session, but not Windows itself).

**Options:** /C(old): Do a “cold” reboot. This is similar to turning the power off and back on, and may be necessary to properly initialize the system.

- 4DOS** REBOOT /C may not physically reset all hardware devices as thoroughly as actually turning off the power; its effect depends on the internal design of each hardware device and on your system configuration. This option will not work under Windows.
- 4NT, TC** /L(ogoff): Log off Windows, but do not reboot. This option is equivalent to selecting Shutdown from the Start menu, then selecting "Close all programs and log on as a different user" in the shutdown dialog. This option will not work properly in Windows 95 / 98 / ME, due to the way Windows responds to "logoff" requests.
- 4NT, TC** /P(oweroff): Log off Windows and turn off the computer.
- 4NT, TC** /S(hutdown): Shut down the system, but do not reboot. In Windows, this is equivalent to selecting Shutdown from the Start menu, then selecting "Shut down the computer" in the shutdown dialog.
- /V(erify): Prompt for confirmation (Y or N) before rebooting.

## RECYCLE [4NT, TC]

(New)

**Purpose:** Delete files in the recycle bin or display the recycle bin status.

**Format:** RECYCLE [/D /E /Q /P] [drives ...]

**drives:** Local fixed and removable (non CD-ROM / DVD) drives

**/D(elete)**                      **/P(rompt)**

**/E** (no error messages)                      **/Q**(uiet)

**Usage:** If you don't specify any drives (or paths), RECYCLE will delete (or display) everything in the recycle bin for all local drives.

Options: **/D(elete):** Empty the recycle bin for the specified drives.

**/E (No error messages):** Suppress all non-fatal error messages, such as "File Not Found." Fatal error messages, such as "Drive not ready," will still be displayed. This option is most useful in batch files.

**/P(rompt):** Prompt the user to confirm each delete operation. Your options at the prompt are explained in detail on page 80.

**/Q(quiet):** Don't display filenames or the number of files deleted. This option is most often used in batch files.

**REM**

(Compatible)

**Purpose:** Put a comment in a batch file.

**Format:** REM [*comment*]

***comment:*** The text to include in the batch file.

**Usage:** The REM command lets you place a remark or comment in a batch file. Batch file comments are useful for documenting the purpose of a batch file and the procedures you have used. For example:

```
rem This batch file provides a
rem menu-based system for accessing
rem word processing utilities.
rem
rem Clear the screen and get selection
cls
```

REM must be followed by a space or tab character, then the comment. Comments can be up to 507 characters long in 4DOS, or 2,043 characters in 4NT and Take Command. The command processor will ignore everything on the line following the “REM”, including quotes, redirection symbols, and other commands (see below for the exception to this rule).

If ECHO is ON, the comment is displayed. Otherwise, it is ignored. If ECHO is ON and you don't want to display the line, preface the REM command with an at sign [@].

You can also place a comment in a batch file by starting the comment line with two colons [::]. In essence this creates a batch file “label” without a valid label name. Such comments are processed slightly faster than those entered with REM.

- ❖ You can use REM to create a zero-byte file if you use a redirection symbol immediately after the REM command. For example, to create the zero-byte file *C:\FOO*:

```
[c:\] rem > foo
```

(This capability is included for compatibility with *COMMAND.COM* and *CMD.EXE*. A simpler method for creating a zero-byte file with 4DOS, 4NT, or Take Command is to use **>filename** as a command, with no actual command before the [>] redirection character.)

## **REN / RENAME**

(Enhanced)

**Purpose:** Rename files or subdirectories.

**Format:** REN [/A:[-][+]*rhsad*] /E /I"text" /N /P /Q /S /T] [*@file*] *old\_name...*  
*new\_name*

or

RENAME [/A:[-][+]*rhsad*] /E /I"text" /N /P /Q /S /T] [*@file*] *old\_name...*  
*new\_name*

***old\_name*:** Original name of the file(s) or subdirectory.

***new\_name*:** New name to use, or new path on the same drive.

***@file*:** A text file containing the names of the source files to rename, one per line (see page 109 for details).

<b>/A:</b> (Attribute select)	<b>/P</b> (rompt)
<b>/E</b> (No error messages)	<b>/Q</b> (uiet)
<b>/I"text"</b> (match description)	<b>/S</b> (ubdirectory)
<b>/N</b> (othing)	<b>/T</b> (otal)

See also: COPY and MOVE.

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104). Use extended wildcards with caution on LFN volumes; see page 105 for details.

**Internet:** Can be used with FTP servers.

**Usage:** REN and RENAME are synonyms. You may use either one.

REN lets you change the name of a file or a subdirectory, or move one or more files to a new subdirectory on the same drive. (If you want to move files to a different drive, use MOVE.)

In its simplest form, you give REN the ***old\_name*** of an existing file or subdirectory and then a ***new\_name***. The ***new\_name*** must not already exist – you can't give two files the same name (unless they are in different directories). The first example renames the file *MEMO.TXT* to *MEM.TXT*. The second example changes the name of the *\WORD* directory to *\WP*:

```
[c:\] rename memo.txt mem.txt
```

```
[c:\] rename \word \wp
```

When you rename files or directories on an LFN drive, you must quote any names which contain white space or special characters. See page 15 for additional details.

You can also use REN to rename a group of files that you specify with wildcards, as multiple files, or in an include list. When you do, the **new\_name** must use one or more wildcards to show what part of each filename to change. Both of the next two examples change the extensions of multiple files to **.SAV** (enter the first command on one line):

```
[c:\] ren config.sys autoexec.bat 4start.btm *.sav  
[c:\] ren *.txt *.sav
```

REN can move files to a different subdirectory on the same drive. When it is used for this purpose, REN requires one or more filenames for the **old\_name** and a directory name for the **new\_name**:

```
[c:\] ren memo.txt \wp\memos\  
[c:\] ren oct.dat nov.dat \data\save\
```

The final backslash in the last two examples is optional. If you use it, you force REN to recognize the last argument as the name of a directory, not a file. The advantage of this approach is that if you accidentally mistype the directory name, REN will report an error instead of renaming your files in a way that you didn't intend.

REN can also move files to a new directory and change their name at the same time if you specify both a path and file name for **new\_name**. In this example, the files are renamed with an extension of **.SAV** as they are moved to a new directory:

```
[c:\] ren *.dat \data\save\*.sav
```

If you use REN to rename a directory, the **new\_name** must normally be specified explicitly, and cannot contain wildcards. You can override this restriction with **/S**. When you rename a directory the extended directory search database will be automatically updated to reflect the change. See page 83 for details.

- ❖ You can also rename a subdirectory to a new location in the directory tree on the same physical drive (sometimes called “prune and graft”). This feature works under Windows 98 / ME and Windows NT / 2000 /

XP, but not under DOS or Windows 95, which do not support it internally. You must specify the new name explicitly, not just give the path. For example, if the *D:\4NT* directory contains a subdirectory *TEST*, you can rename *TEST* to be a subdirectory of the root directory like this:

```
[d:\4nt] ren TEST \TEST\
```

REN does not change a file's attributes. The *new\_name* file(s) will have the same attributes as *old\_name*.

### **4NT, TC FTP Usage**

If you have appropriate permissions, you can rename files on FTP servers. The URL must be enclosed in double quotes so the forward slashes won't be interpreted as switches. For example (enter this on one line):

```
[c:\] ren "ftp://jpsoft.com/file1.txt" file2.txt
```

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see Using FTP Servers on page 113, and the IFTP command on page 382.

Options: **/A:** (Attribute select): Rename only those files that have the specified attribute(s) set. See **Attribute Switches** on page 111 for information on the attributes which can follow **/A:**.

**/E** (No error messages): Suppress all non-fatal error messages, such as "File Not Found." Fatal error messages, such as "Drive not ready," will still be displayed. This option is most useful in batch files.

**/I"text"**: Select files by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**/N(othing)**: Do everything except actually rename the file(s). This option is useful for testing what a REN command will actually do.

**/P(rompt)**: Prompt the user to confirm each rename operation. Your options at the prompt are explained in detail on page 80.

**/Q(quiet):** Don't display filenames or the number of files renamed. This option is most often used in batch files. See also **/T**.

**/S(ubdirectory):** Normally, you can rename a subdirectory only if you do not use any wildcards in the **new\_name**. This prevents subdirectories from being renamed inadvertently when a group of files is being renamed with wildcards. **/S** will let you rename a subdirectory even when you use wildcards. **/S** does **not** cause REN to process files in the current directory and all subdirectories as it does in some other file processing commands. To rename files throughout a directory tree, use a GLOBAL REN (see page 361 for information on GLOBAL).

**/T(otal):** Don't display filenames as they are renamed, but report the number of files renamed. See also **/Q**.



## **RETURN**

(New)

**Purpose:** Return from a GOSUB (subroutine) in a batch file.

**Format:** RETURN [*value*]

***value*:** The numeric exit code to return to the command processor or to the previous batch file.

See also: GOSUB.

**Usage:** 4DOS, 4NT, and Take Command allow subroutines in batch files.

A subroutine begins with a label (a colon followed by one or more words) and ends with a RETURN command.

The subroutine is invoked with a GOSUB command from another part of the batch file. When a RETURN command is encountered the subroutine terminates, and execution of the batch file continues on the line following the original GOSUB. If RETURN is encountered without a GOSUB, the command processor will display a “Missing GOSUB” error.

You cannot execute a RETURN from inside a DO loop.

The following batch file fragment calls a subroutine which displays the files in the current directory:

```
echo Calling a subroutine
gosub subr1
echo Returned from the subroutine
quit

:subr1
dir /a/w
return
```

- ❖ If you specify a ***value***, RETURN will set the internal error level to that value. For information on exit codes see the IF command, and the internal %\_? variable on page 161.

**SCREEN**

(New)

**Purpose:** Position the cursor on the screen and optionally display a message.

**Format:** SCREEN *row column* [*text*]

***row*:** The new row location for the cursor.

***column*:** The new column location for the cursor.

***text*:** Optional text to display at the new cursor location.

See also: ECHO, SCRPUT, TEXT, and VSCRPUT.

**Usage:** SCREEN allows you to create attractive screen displays in batch files. You use it to specify where a message will appear on the screen. You can use SCREEN to create menus and other similar displays. For example, the following batch file fragment displays a menu:

```
@echo off
cls
screen 3 10  Select a number from 1 to 4:
screen 6 20  1 - Word Processing
screen 7 20  2 - Spreadsheet
screen 8 20  3 - Telecommunications
screen 9 20  4 - Quit
```

SCREEN does not change the screen colors. To display text in specific colors, use SCRPUT or VSCRPUT. SCREEN always leaves the cursor at the end of the displayed text.

The ***row*** and ***column*** values are zero-based, so on a 25 line by 80 column display, valid ***rows*** are 0 - 24 and valid ***columns*** are 0 - 79. SCREEN checks for a valid ***row*** and ***column***, and displays a “Usage” error message if either value is out of range. In Take Command, the maximum ***row*** value is determined by the current height of the Take Command window, and the maximum ***column*** value is determined by the current virtual screen width (see page 39 for more information).

You can also specify the ***row*** and ***column*** as offsets from the current cursor position. Begin the value with a plus sign [+] to move the cursor down or to the right, or with a minus sign [-] to move the cursor up or to the left. This example prints a string 3 lines above the current position, in absolute column 10:

```
screen -3 10 Hello, World!
```

If you specify 999 for the *row*, SCREEN will center the text vertically on the display. If you specify 999 for the *column*, SCREEN will center the text horizontally. This example prints a message at the center of the window:

```
screen 999 999 Hello, World
```

**SCRPUT**

(New)

**Purpose:** Position text on the screen and display it in color.

**Format:** SCRPUT *row col* [BRIght][BLInk] *fg* ON [BRIght] *bg text*

***row***: Starting row

***col***: Starting column

***fg***: Foreground text color (**BLI** only valid in full-screen DOS)

***bg***: Background text color

***text***: The text to display

See also: ECHO, SCREEN, TEXT, and VSCRPUT.

**Usage:** SCRPUT allows you to create attractive screen displays in batch files. You use it to specify where a message will appear on the screen and what colors will be used to display the message text. You can use SCRPUT to create menu displays, logos, etc.

SCRPUT works like SCREEN, but requires you to specify the display colors. See page 30 for details about colors and color names, and notes on the use of blinking bright background colors.

The ***row*** and ***column*** values are zero-based, so on a 25 line by 80 column display, valid ***rows*** are 0 - 24 and valid ***columns*** are 0 - 79. SCRPUT checks for a valid ***row*** and ***column***, and displays a “Usage” error message if either value is out of range. In Take Command, the maximum ***row*** value is determined by the current height of the Take Command window, and the maximum ***column*** value is determined by the current virtual screen width (see page 39 for more information).

You can also specify the ***row*** and ***column*** as offsets from the current cursor position. Begin the value with a plus sign [+] to move down the specified number of rows or to the right the specified number of columns, or with a minus sign [-] to move up or to the left. If you specify 999 for the ***row***, SCRPUT will center the text vertically in the command processor window. If you specify 999 for the ***column***, SCRPUT will center the text horizontally.

SCRPUT normally does not move the cursor when it displays the ***text***. However, under 4DOS if you have set OutputBIOS to Yes in *4DOS.INI* (see below), SCRPUT will leave the cursor at the end of the displayed ***text***.

The following batch file fragment displays a menu in color:

```
cls white on blue
scrput 3 10 bri whi on blu Select an option:
scrput 6 20 bri red on blu 1 - Word Processing
scrput 7 20 bri yel on blu 2 - Spreadsheet
scrput 8 20 bri gre on blu 3 - Communications
scrput 9 20 bri mag on blu 4 - Quit
```

**4DOS**    ❖If you have an unusual display adapter which does not support the direct video output used by SCRPUT, see the OutputBIOS directive on page 237.

**SELECT**

(New)

**Purpose:** Interactively select files for a command.

**Format:** SELECT [/A[:][-][+]*rhsad*] /C[HP] /D /E /H /I"text" /J /L  
/O[:][-]*acdeginrsu* /T:acw /X /Z] [*command*] ... (*files*)...

***command*:** The command to execute with the selected files.

***files*:** The files from which to select. File names may be enclosed in either parentheses or square brackets. The difference is explained below.

/A(tribute select)	/J(ustify names)
/C[HP] (Compression)	/L(ower case)
/D(isable color coding)	/O(rder)
/E (use upper case)	/T(ime)
/H(ide dots)	/X (display short names)
/I"text" (match description)	/Z (use FAT format)

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104). Ranges **must** appear immediately after the SELECT keyword. Use extended wildcards with caution on LFN volumes; see page 105 for details.

**Internet:** Can be used with FTP servers.

**Usage:** SELECT allows you to select files for internal and external commands by using a "point and shoot" display. You can have SELECT execute a command once for each file you select, or have it create a list of files for a command to work with. The ***command*** can be an internal command, an alias, an external command, or a batch file.

If you use parentheses around the ***files***, SELECT executes the ***command*** once for each file you have selected. During each execution, one of the selected files is passed to the ***command*** as an argument. If you use square brackets around ***files***, the SELECTed files are combined into a single list, separated by spaces. The command is then executed once with the entire list presented as part of its command-line arguments.

**4NT, TC** SELECT can also select file names on FTP servers. The URL must be enclosed in double quotes so the forward slashes won't be interpreted as switches. For example:

```
[c:\] select del ("ftp://ftp.domain.com/")
```

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see *Using FTP Servers* on page 113, and the IFTP command on page 382.

### ***Using the SELECT File List***

When you execute the SELECT command, the file list is displayed in a full-window format which includes a top-line status bar and shows the command to be executed, the number of files marked, and the number of Kbytes in those files.

SELECT supports the mouse for selecting and scrolling the list. You can also use the cursor up, cursor down, PgUp, and PgDn keys to scroll. You can also use character matching to find specific files, just as you can in any popup window; see page 35 for details. While the file list is displayed you can enter any of the following keys to select or unselect files, display files, execute the command, or exit:

<b>space</b>	Select a file, or unselect a marked file.
<b>+</b>	Select a file (all products), or unselect a marked file (4DOS / 4NT only).
<b>-</b>	Unselect a marked file.
<b>*</b>	Reverse all of the current marks (except those on subdirectories). If no files have been marked you can use * to mark all of the files.
<b>/</b>	Unselect all files.
<b>Ctrl-L</b>	In 4DOS and 4NT, view the current highlighted file with LIST (see page 401). When you exit from LIST, the SELECT screen will be restored.
<b>Enter</b>	Execute the command with the marked files, or with the currently highlighted file if no files have been marked.
<b>Esc</b>	Skip the files in the current display and go on to the next file specification inside the parentheses or brackets (if any).
<b>Ctrl-C or Ctrl-Break</b>	Cancel the current SELECT command entirely.

On FAT drives the file list is shown in standard FAT directory format, with names at the left and descriptions at the right. On LFN drives the format is similar but more space is allowed for the name, and the description is not shown. In this format long names are truncated if they do not fit in the allowable space. For a short-name format (including descriptions) on long filename drives, use the **/X** and **/** or **/Z** switches.

When displaying descriptions in the short filename format, **SELECT** adds a right arrow **►** at the end of the line if the description is too long to fit on the screen. This symbol will alert you to the existence of additional description text. You can use the left and right arrow keys to scroll the description area of the screen horizontally and view the additional text.

**4DOS**, You can set the default colors used by **SELECT** on the “Colors” tab

**4NT** of the configuration dialog, or with the **SelectColors** and **SelectStatBarColors** directives in the **.INI** file (see page 229). If **SelectColors** is not used, the **SELECT** display will use the current default colors. If **SelectStatBarColors** is not used, the status bar will use the reverse of the **SELECT** colors.

### ***Creating SELECT Commands***

In the simplest form of **SELECT**, you merely specify the command and then the list of files from which you will make your selection(s). For example:

```
[c:\] select copy (*.com *.exe) a:\
```

will let you select from among the **.COM** files on the current drive, and will then invoke the **COPY** command to copy each file you select to drive A:. After the **.COM** files are done, the operations will be repeated for the **.EXE** files.

If you want to select from a list of all the **.COM** and **.EXE** files mixed together, create an include list inside the parentheses by inserting a semicolon (see page 104 for information on include lists):

```
[c:\] select copy (*.com;*.exe) a:\
```



Finally, if you want the SELECT command to send a single list of files to COPY, instead of invoking COPY once for each file you select, put the file names in square brackets instead of parentheses:

```
[c:\] select copy [*.com;*.exe] a:\
```

If you use brackets, you have to be sure that the resulting command (the word COPY, the list of files, and the destination drive in this example) does not exceed the command line length limit: 511 characters for internal commands or 126 characters for external commands under 4DOS; 2,047 characters for all commands under 4NT and Take Command. The current line length is displayed by SELECT while you are marking files to help you to conform to these limits.

The parentheses or brackets enclosing the file name(s) can appear anywhere within the command; SELECT assumes that the first set of parentheses or brackets it finds is the one containing the list of files from which you wish to make your selection.

When you use SELECT on an LFN drive, you must quote any file names inside the parentheses which contain white space or special characters. See page 15 for additional details. For example, to copy selected files from the *“Program Files”* directory to the *E:\SAVE* directory:

```
[c:\] select copy ("Program Files\*.*) e:\save\
```

File names passed to the **command** will be quoted automatically if they contain white space or special characters.

The list of files from which you wish to select can be further refined by using date, time, size, and file exclusion ranges (see page 97). The range(s) must be placed immediately after the word SELECT. If the **command** is an internal command that supports ranges, an independent range can also be used in the **command** itself.

You cannot use command grouping to make SELECT execute several commands, because SELECT will assume that the parentheses are marking the list of files from which to select, and will display an error message or give incorrect results if you try to use parentheses for command grouping instead. (You can use a SELECT command **inside** command grouping parentheses, you just can't use command grouping to specify a group of commands for SELECT to execute.)

**❖ Advanced Topics**

If you don't specify a command, the selected filename(s) will become the command. For example, this command defines an alias called **UTILS** that selects from the executable files in the directory **C:\UTIL**, and then executes them in the order marked (enter the alias on one line):

```
[c:\] alias utils select  
      (c:\util\*.com;*.exe;*.btm;*.bat)
```

If you want to use filename completion (see page 67) to enter the filenames inside the parentheses, type a space after the opening parenthesis. Otherwise, the command-line editor will treat the open parenthesis as the first character of the filename.

With the **/I** option, you can select files based on their descriptions. **SELECT** will display files if their description matches the text after the **/I** switch. The search is not case sensitive. You can use wildcards and extended wildcards as part of the text.

When sorting file names and extensions for the **SELECT** display, the command processor normally assumes that sequences of digits should be sorted numerically (for example, the file **DRAW2** would come before **DRAW03** because 2 is numerically smaller than 03), rather than strictly alphabetically (where **DRAW2** would come second because "2" comes after "0"). You can defeat this behavior and force a strict alphabetic sort with the **/O:a** option.

**4DOS**     **SELECT** normally writes text directly to the screen. If you have an unusual display adapter which does not support direct video output, see the **OutputBIOS** directive on page 237.

**4DOS**     If you receive a stack overflow error when using **SELECT** in complex, nested command sequences, see the notes under the **StackSize** directive on page 238.

Options:   **❖A**(ttribute select): Display only those files that have the specified attribute set. See **Attribute Switches** on page 111 for information on the attributes which can follow **/A**.

**/C**(ompression): Display per-file and total compression ratios on compressed drives. The compression ratio is displayed instead of the file description. The ratio is left blank for directories and files with a length of 0 bytes, and for files on non-compressed drives. The

compression ratios will not be visible on LFN drives unless you use **/Z** to switch to the traditional short filename format.

Under 4DOS, only the compression programs distributed with MS-DOS and Windows 95 / 98 / ME (DRVSPACE and DBLSPACE) are supported. Under 4NT and Take Command, only compressed NTFS drives are supported.

**4DOS** Using **/CH** displays compression ratios like **/C**, but bases the calculation on the host drive's cluster size. This gives a more accurate picture of the space saved through compression than is given by **/C**.

**4DOS** If **/CP** is used instead of **/C**, the compression is displayed as a percentage (e.g., 33%) instead of a ratio. If **/CHP** is used instead of **/CH**, the host compression is displayed as a percentage. The **/CHP** option must be entered as shown; you can **not** use **/CPH**.

See the **DIR /C** documentation on page 464 for more details on how compression ratios are calculated.

**4DOS, 4NT** ❖ **/D**(isable color coding): Don't colorize the directory.

**/E** (use upper case): Display filenames in upper case; also see SETDOS /U (page 477) and the UpperCase directive in the .INI file.

**/H**(ide dots): Suppress the display of the "." and ".." directories.

**/I"text"**: Display filenames by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**/J**(ustify names): Justify (align) filename extensions and display them in the traditional format.

**/L**(ower case): Display file and directory names in lower case; also see SETDOS /U (page 477) and the UpperCase directive in the .INI file (page 228).

**/O**(rder): Set the sort order for the files. The order can be any combination of the following options:

- Reverse the sort order for the next option.
- a** Sort names and extensions in standard ASCII order, rather than sorting numerically when digits are included in the name or extension.
- c** Sort by compression ratio (the least compressed file in the list will be displayed first). Under 4DOS, if **/O:c** is used with **/CH** or **/CHP** the sort will be based on the host-drive compression ratios. For information on supported compression systems see **/C** above.
- d** Sort by date and time (oldest first).
- e** Sort by extension.
- g** Group subdirectories together.
- i** Sort by the file description (ignored if **/C** or **/O:c** is also used).
- n** Sort by filename (this is the default).
- r** Reverse the sort order for all options.
- s** Sort by size.
- u** Unsorted.

**/T:acw** (Time display): Specify which of the date and time fields on an LFN drive should be displayed and used for sorting (see page 17 for more information on date and time fields):

- a** Last access date and time (access time will always be 00:00 on VFAT and FAT32 volumes).
- c** Creation date and time.
- w** Last write date and time (default).

**/X:** Display short filenames, in the traditional FAT format (like **/Z**), on LFN drives.

**/Z:** Display a directory on an LFN drive in the old-style format, with the filename at the left and the description at the right. Long names will be truncated to 12 characters; if the name is longer than 12 characters, it will be followed by a right arrow [►].

## **SENDMAIL**    [4NT, TC]

(New)

**Purpose:**    Send an email message.

**Format:**    SENDMAIL [/A file][/V] *address subject text*

***address:*** The email address.

***subject:*** The subject line.

***text:*** The message to send.

/A(ttachment)            /V(erbose)

**Usage:**    SENDMAIL sends an email message via SMTP. The message text can either be entered on the command line or can be place in a text file.

Before you can use SENDMAIL, you must either set the MailServer and MailUser .INI directives, or have a default account in the registry. Depending on your system configuration, you may also need to start an Internet connection before you use SENDMAIL.

A SENDMAIL message has three required parts: an address, a subject, and text, and an optional attachment.

The ***address*** is a standard Internet email address:

```
[c:\] sendmail abc@xyz.com ...
```

If the ***address*** contains white space, the entire address must be surrounded by double quotes.

The ***subject*** will appear as the subject line in the message. If it contains white space, it must be surrounded by double quotes.

The ***message*** may either be entered on the command line or be placed in a text file. To tell SENDMAIL to send the contents of a file as the message text, enter the ***message*** portion of the command as an @ sign, followed by the filename

```
[c:\] sendmail ab@yz.com Party @c:\invitation.txt
```

**Options:**    /A(ttachment): Attach the specified file to the email message. The /A switch and the file to attach is must appear before the ***address***.

/V(erbose): Show all the interaction with the server, except for the message header and message body text.

**SET**

(Enhanced)

**Purpose:** Display, create, modify, or delete environment variables.

**Format:** SET [/A /D /M /P /R *file*... /U /V] [*name*[=][*value*]]

***file*:** One or more files containing variable definitions.

***name*:** The name of the environment variable to define or modify.

***value*:** The new value for the variable.

/A(rithmetic)	/R(ead from file)
/D(efault)	/S(ystem)
/M(aster)	/U(ser)
/P	/V(olatile)

See also: ESET and UNSET.

**Usage:** Every program and command inherits an *environment*, which is a list of variable ***names***, each of which is followed by an equal sign and some text. Many programs use entries in the environment to modify their own actions. 4DOS, 4NT, and Take Command use several environment variables (see page 150). See pages 23 and 148 for more information on the environment.

You can also create or modify environment variables with the Environment dialog (see page 55). The dialog allows you to enter the variable ***name*** and ***value*** into separate fields in a dialog box, rather than using the SET command. All of the information in this section also applies to variables defined via the dialog, unless otherwise noted.

If you simply type the SET command with no options or arguments, it will display all the names and values currently stored in the environment. Typically, you will see an entry called PATH, an entry called CMDLINE, and whatever other environment variables you and your programs have established:

```
[c:\] set
PATH=C:\;C:\Win98;C:\Win98\SYSTEM;C:\UTIL
CMDLINE=C:\TCMD200\TCSTART.CMD
```

To add a variable to the environment, type SET, a space, the variable name, an equal sign, and the text:

```
[c:\] set mine=c:\finance\myfiles
```

The variable name is converted to upper case by 4DOS. (For compatibility with *CMD.EXE*, 4NT and Take Command store the name as you enter it, and do not shift it to upper case (however, case is ignored when looking for a variable; for example **MyVar**, **myvar**, and **MYVAR** all refer to the same variable). The text after the equal sign will be left just as you entered it. If the variable already exists, its value will be replaced with the new text that you entered.

- ! Normally you should not put a space on either side of the equal sign. A space before the equal sign will become part of the **name**; a space after the equal sign will become part of the **value**.
- ! If you use SET to create a variable with the same name as one of the 4DOS, 4NT, or Take Command internal variables (see page 153), you will disable the internal variable. If you later execute a batch file or alias that depends on that internal variable, it may not operate correctly.

To display the contents of a variable, type SET plus the variable name:

```
[c:\] set mine
```

You can edit environment variables with the ESET command. To remove variables from the environment, use UNSET, or type SET plus a variable name and an equal sign:

```
[c:\] set mine=
```

The variable **name** is limited to a maximum of 80 characters. Under 4DOS, the name and **value** together cannot be longer than 511 characters. In 4NT and Take Command, the maximum length for the name and value together is 2,047 characters.

In 4DOS, the size of the environment can be specified on the Startup page of the OPTION dialogs, with the Environment and EnvFree directives in *4DOS.INI* (see page 211), or with the /E: startup switch (see your *Introduction and Installation Guide*). In 4NT and Take Command, the size of the environment is set automatically, and increased as necessary as you add variables.

Options:   ❖A(rithmetic): Evaluate the argument to the right of the equal sign, place the result in the environment, and display it. You can use

**4NT, TC** @EVAL to perform the same task; SET /A is included only for compatibility with *CMD.EXE*. For example, this command adds 2 and 2, and puts 4 in the variable VAR:

```
[c:\] set /a var=2+2
```

In addition, /A interprets alphabetic strings to the right of the equal sign as environment variable names even if they are not preceded by a percent sign. For example, this sequence will set Y to 4:

```
[c:\] set x=2  
[c:\] set /a y=x+2
```

**4NT, TC** /D(efault environment): Create/modify/delete a “default” variable in the registry (HKU\DEFAULT\Environment).

**4DOS** /M(aster): Display or modify the master environment rather than the local environment. This option is only useful in secondary shells.

/P(ause): Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail on page 80.

- ❖ /R(ead): Read environment variables from a file. This is much faster than loading variables from a batch file with multiple SET commands. Each entry in the file must fit within the command-line length limit for your command processor: 511 bytes for 4DOS and 4,095 bytes for 4NT and Take Command. The file is in the same format as the SET display (*i.e.*, **name=value**), so SET /R can accept as input a file generated by redirecting SET output.

For example, the following commands will save the environment variables to a file, and then reload them from that file:

```
set > varlist  
set /r varlist
```

You can load variables from multiple files by listing the filenames individually after the /R. You can add comments to a variable file by starting the comment line with a colon [:].

If you are creating a SET /R file by hand, and need to create an entry that spans multiple lines in the file, you can do so by terminating each line, except the last, with an escape character (see page 119). However, you cannot use this method to exceed the command-line length limit.



If you do not specify a filename and input is redirected, SET /R will read from STDIN.

**4NT, TC** /S(ystem): Create/modify/delete a “system” variable in the registry (HKLM\System\CurrentControlSet\Control\Session Manager\Environment).

**4NT, TC** /U(ser): Create/modify/delete a “user” variable in the registry (HKCU\Environment).

**4NT, TC** /V(olatile): Create/modify/delete a “volatile” variable in the registry (HKCU\Volatile Environment).

**SETDOS**

(New)

**Purpose:** Display or set the 4DOS, 4NT, or Take Command configuration.

**Format:** SETDOS [/A?/B? /C? /D? /E? /Fn.n /G?? /I+ | - *command* /L? /M?  
/N? /P? /R? /S?:? /U? /V? /X[+ | -]n /Y?]

/A(NSI)	/M(ode for editing)
/B(right background)	/N(o clobber)
/C(ompound)	/P(arameter character)
/D(escriptions)	/R(ows)
/E(scape character)	/S(hape of cursor)
/F(ormat for @EVAL)	/U(pper case)
/G (numeric separators)	/V(erbose)
/I(internal commands)	/X (expansion, special characters)
/L(ine)	/Y (debug batch file)

**Usage:** SETDOS allows you to customize certain aspects of 4DOS, 4NT, and Take Command to suit your personal tastes or the configuration of your system. Each of these options is described below.

You can display the value of all SETDOS options by entering the SETDOS command with no parameters.

Most of the SETDOS options can be initialized when the command processor executes the directives in the *.INI* file (see page 217), and can also be changed from the configuration dialog. The name of the corresponding directive is listed in square brackets [ ] with each option below; if none is listed, that option cannot be set from the *.INI* file. You can also define the SETDOS options in your *AUTOEXEC.BAT*, *4START*, *TCSTART* or other startup file (see page 132), in aliases, or at the command line.

Secondary shells automatically inherit most configuration settings currently in effect in the previous shell. If values have been changed by SETDOS since 4DOS, 4NT, or Take Command started, the new values will be passed to the secondary shell. For details on inheritance of SETDOS values by secondary shells and their relationship to the *.INI* file, see page 207.

SETDOS /I settings are not inherited by secondary shells. If you want to use SETDOS /I- to disable commands in all shells, place the

SETDOS command(s) in your *4START* / *TCSTART* file (see page 132), which is executed when any shell starts.

Many of the options below are marked with ❖ If you are a new user, skip these and read the /M, /S, and /U options, which are more common.

- Options: ❖/A(NSI) [ANSI]: This option determines whether 4DOS will attempt to use ANSI escape sequences for the CLS and COLOR
- 4DOS** commands. 4DOS normally determines this itself, but if you are using a non-standard ANSI driver or your loading sequence is unusual, you may need to explicitly inform 4DOS. /A0 allows 4DOS to determine automatically whether an ANSI driver is installed (the default). /A1 forces 4DOS to assume an ANSI driver is installed. /A2 forces 4DOS to assume an ANSI driver is not installed. See page 28 for more information on ANSI drivers.
- 4NT,TC** The /A(NSI) option determines whether 4NT's or Take Command's ANSI support is enabled. /A1 enables ANSI string processing; the default of /A0 disables ANSI strings. See the ANSI Codes Reference in the Reference section of the online help for a list of the ANSI sequences supported by 4NT and Take Command. Also see the \_ANSI internal variable on page 156.
- 4DOS** /B(right background) [BrightBG]: This option determines whether 4DOS configures the video adapter for blinking text (/B0, the default) or bright background colors (/B1), or leaves the video bright / blink configuration unchanged (/B2). See page 30 for a detailed discussion of this option.
- ❖ /C(ompound character) [CommandSep]: This option sets the character used for separating multiple commands on the same line. The default is the caret [^] in 4DOS and the ampersand [&] in 4NT and Take Command. You cannot use any of the redirection characters (| > <), or the blank, tab, comma, or equal sign as the command separator. The command separator is saved by SETLOCAL and restored by ENDLOCAL. The following example changes the separator character to a tilde [~]:

```
[c:\] setdos /c~
```

If you want to share batch files or aliases among 4DOS, 4NT, and Take Command, see page 156 for information on the %+ variable, which

retrieves the current command separator, and page 195 for details on using compatible command separators for all the products you use.

- ❖ **/D(Descriptions) [Descriptions and DescriptionName]:** This option controls whether file processing commands like COPY, DEL, MOVE, and REN process file descriptions along with the files they belong to. **/D1** turns description processing on, which is the default. **/D0** turns description processing off.

You can also use **/D** to set the name of the hidden file in each directory that contains file descriptions. To do so, follow **/D** with the filename in quotes:

```
[c:\] setdos /d"files.bbs"
```

- ! **Use this option with caution** because changing the name of the description file will make it difficult to transfer file descriptions to another system.
- ❖ **/E(scape character) [EscapeChar]:** This option sets the character used to suppress the normal meaning of the following character. Any character following the escape character will be passed unmodified to the command. The default escape character is a Ctrl-X (ASCII 24, which appears on screen as an up-arrow [↑]) in 4DOS, and the caret [^] in 4NT and Take Command. You cannot use any of the redirection characters (**| > <**) or the blank, tab, comma, or equal sign as the escape character. The escape character is saved by SETLOCAL and restored by ENDLOCAL. Certain characters (**b, c, e, f, k, n, q, r, s,** and **t**) have special meanings when immediately preceded by the escape character. See page 119 for additional details.

If you want to share batch files or aliases among 4DOS, 4NT, and Take Command, see page 156 for information on the **%=** variable, which retrieves the current escape character, and page 195 for details on using compatible escape characters for all the products you use.

**/F(ormat for @EVAL) [EvalMax, EvalMin]:** This option sets the default decimal precision for the @EVAL variable function (see page 171). The maximum precision is 10 digits to the right of the decimal point.

The general form of this option is **/Fx.y**, where the **x** value sets the minimum number of digits to the right of the decimal point and the **y** value sets the maximum number of digits. You can use **=x,y** instead of

=**x.y** if the comma is your decimal separator. Both values can range from 0 to 10. You can specify either or both values: **/F2.5**, **/F2**, and **/F.5** are all valid entries. If **x** is greater than **y**, it is ignored; if only **x** is specified, **y** is set to the same value (e.g. **/F2** is equivalent to **/F2.2**). See the **@EVAL** function on page 171 if you want to set the precision for a single computation.

- ❖ **/G** (Numeric separators) [DecimalChar, ThousandsChar]: This option sets the decimal and thousands separator characters. The format is **/Gxy** where “x” is the new decimal separator and “y” is the new thousands separator. Both characters must be included. The only valid settings are **/G.**, (period is the decimal separator, comma is the thousands separator); **/G,** (the reverse); or **/G0** to remove any custom setting and use the default separators associated with your current country code (this is the default).

The decimal separator is used for **@EVAL**, numeric **IF** and **IFF** tests, version numbers, and other similar uses. The thousands separator is used for numeric output, and is skipped when performing calculations in **@EVAL**.

- ❖ **/I**(nternal): This option allows you to disable or enable internal commands. To disable a command, precede the command name with a minus [-]. To re-enable a command, precede it with a plus [+]. For example, to disable the internal **LIST** command to enable the use of an external command with the same name:

```
[c:\] setdos /i-list
```

To re-enable all disabled commands use **/I\***.

- 4DOS**    ❖ **/L**(ine) [LineInput]: This option controls how 4DOS gets its input from the command line. **/L0** tells the command processor to use character input (the default). **/L1** tells it to use line input (like **COMMAND.COM**). **/L1** will disable command-line editing, history recall, filename completion, and the directory history window. It should only be used if it is needed for compatibility with a specific program. If you have a program that requires line input, you can use the following line in an alias or batch file to change the line input option just for that single program:

```
setdos /L1 ^ program %& ^ setdos /L0
```

See the **Compatibility** section of the 4DOS online help for information on programs which require this option.

**/M(ode) [EditMode]:** This option controls the initial line editing mode. To start in overstrike mode at the beginning of each command line, use **/M0** (the default in 4DOS and 4NT). To start in insert mode, use **/M1** (the default in Take Command).

- ❖ **/N(o clobber) [NoClobber]:** This option controls output redirection (see page 88). **/N0** means existing files will be overwritten by output redirection (with **>**) and that appending (with **>>**) does not require the file to exist already. This is the default. **/N1** means existing files may not be overwritten by output redirection, and that when appending, the output file must exist. A **/N1** setting can be overridden with the **!** character.
- ❖ **/P(arameter character) [ParameterChar]:** This option sets the character used after a percent sign to specify all or all remaining command-line arguments in a batch file or alias (e.g., **%\$** or **%n\$**; see pages 128 and 257). The default is the ampersand **[&]** for 4DOS and the dollar sign **[\$]** for 4NT and Take Command. The parameter character is saved by **SETLOCAL** and restored by **ENDLOCAL**.

If you want to share batch files or aliases among 4DOS, 4NT, and Take Command, see page 195 for details on selecting compatible parameter characters for all the products you use.

**4DOS**    ❖ **/R(ows) [ScreenRows]:** This option sets the number of screen rows used by the video display. Normally 4DOS detects the screen size, but if you have a non-standard display you may need to set it explicitly. This option does not affect screen scrolling (that is controlled by your video driver, the BIOS, or *ANSI.SYS*). It also does not set the screen size; it is used only to specify the screen height for **LIST**, **SELECT**, paged output options (i.e., **TYPE /P**), and error checking in screen output commands.

**/S(hape) [CursorOver, CursorIns]:** This option sets the cursor shape. The format is **/So:i** where **o** is the cursor size for overstrike mode, **i** the cursor size for insert mode. The size is entered as a percentage of the total character height (in 4DOS and 4NT) or width (in Take Command). The default values for 4DOS and 4NT are 10:100 (a 10% underscore cursor for overstrike mode, and a 100% block cursor for insert mode). The default values for Take Command are 100:15 (a 100% or block cursor for overstrike mode, and a 15% or thin line cursor for insert mode). Because of the way video drivers remap the cursor shape, you may not get a smooth progression in the cursor size from 0% - 100%. To disable the cursor in 4DOS or 4NT, enter **/S0:0**.

If either value is -1, , the command processor will not attempt to modify the cursor shape at all. You can use this feature to give another program full control of the cursor shape. You can retrieve the current cursor shape values with the `%_CI` and `%_CO` internal variables (see page 158).

**/U(pper) [UpperCase]:** This option controls the default case (upper or lower) for file and directory names displayed by internal commands like `COPY` and `DIR`. **/U0** displays file names in lower case (the default). **/U1** displays file names in the traditional upper case. The **/U** setting is ignored for filenames on LFN drives. Names on such drives are always displayed in the case in which they are stored; see page 15 for more details.

- ❖ **/V(erbose) [BatchEcho]:** This option controls the default for command echoing in batch files. **/V0** disables echoing of batch file commands unless `ECHO` is explicitly set ON. **/V1**, the default setting, enables echoing of batch file commands unless `ECHO` is explicitly set OFF.

**/V2** forces echoing of all batch file commands, even if `ECHO` is set OFF, or if the line begins with an `@`. This allows you to turn echoing on for a batch file without editing the batch file and removing the `ECHO OFF` command(s) within it. **/V2** is intended for debugging, and can be set with SETDOS, but not with the configuration dialog or the `BatchEcho` directive in the `.INI` file. For more information on batch file debugging see page 137, and **/Y** (below).

- ❖ **/W (switch character):** This option sets the DOS switch character (normally a slash [/]). **It will not work** in most current versions of MS-DOS or PC DOS, or under Windows 95, 98, or ME, as all of these operating systems ignore switch character changes. It is included only for compatibility with older versions of DOS, DR-DOS, and some special-purpose DOS-compatible systems which allow switch character changes, and even in those environments the DOS switch character setting may be ignored by application programs.
- ❖ **/X[+|-]n (expansion and special characters):** This option enables and disables alias and environment variable expansion, and controls whether special characters have their usual meaning or are treated as text. It is most often used in batch files to process text strings which may contain special characters. See page 140 for further details on string processing in batch files, and page 197 for details on alias expansion, variable expansion, and special characters.

The features enabled or disabled by **/X** are numbered. All features are enabled when the command processor starts, and you can re-enable all features at any time by using **/X0**. To disable a particular feature, use **/X-n**, where **n** is the feature number from the list below. To re-enable the feature, use **/X+n**. To enable or disable multiple individual features, list their numbers in sequence after the + or - (e.g. **/X-345** to disable features 3, 4, and 5).

The features are:

- 1 All alias expansion
- 2 Nested alias expansion only
- 3 All variable expansion (includes environment variables, batch file parameters, and alias parameters)
- 4 Nested variable expansion only
- 5 Multiple commands, conditional commands, and piping (affects the command separator, **|**, **&&**, **|**, and **|&**)
- 6 Redirection (affects **<**, **>**, **>&**, **>&>**, etc.)
- 7 Quoting (affects back-quotes **[ ]** and double quotes **["]**) and square brackets
- 8 Escape character
- 9 User-defined functions

If nested alias expansion is disabled the first alias of a command is expanded, but any aliases it invokes are not expanded. If nested variable expansion is disabled, each variable is expanded once, but variables containing the names of other variables are not expanded further.

For example, to disable all features except alias expansion while you are processing a text file containing special characters:

```
setdos /x-3456789
... [perform text processing here]
setdos /x0
```

- ❖ **/Y** (debug batch file): **/Y1** enables the built-in batch file debugger. The debugger allows you to “single-step” through a batch file line by line, with the file displayed in a popup window as it executes. For complete details on using the debugger see Debugging Batch Files on page 137



(this topic also covers additional debugging techniques which do not require stepping through each line individually).

To start the debugger, insert a SETDOS /Y1 command at the beginning of the portion of the batch file you want to debug, and a SETDOS /Y0 command at the end. You cannot use the batch debugger with REXX files (see page 145) or EXTPROC files (page 147).

You can also invoke SETDOS /Y1 from the prompt, but because the debugger is automatically turned off whenever the command processor returns to the prompt, you must enter the SETDOS command and the batch file name on the same line, for example:

```
[c:\] setdos /y1 & mybatch.btm
```

**SETLOCAL**

(New/Enhanced)

**Purpose:** Save a copy of the current disk drive, directory, environment, alias list, and special characters.

**Format:** SETLOCAL

See also: ENDLOCAL.

**Usage:** SETLOCAL is used in batch files to save the default disk drive and directory, the environment, the alias list, and the command separator, escape character, parameter character, decimal separator, and thousands separator. You can then change their values and later restore the original values with ENDLOCAL.

For example, this batch file fragment saves everything, removes all aliases, changes the disk and directory, changes the command separator, runs a program, and then restores the original values:

```
setlocal
unalias *
cdd d:\test
setdos /c~
program ~ echo Done!
endlocal
```

SETLOCAL and ENDLOCAL are nestable up to 16 (4NT and TCMD) or 8 (4DOS) levels deep. You cannot use SETLOCAL in an alias or at the command line.

An ENDLOCAL is performed automatically at the end of a batch file if you forget to do so, or when you invoke one batch file from another without using CALL.

❖ You can “export” modified variable variables from inside a SETLOCAL / ENDLOCAL block; see ENDLOCAL for details.

**4DOS !** Do not load memory-resident programs (TSRs) from a batch file while SETLOCAL is in effect. If you do, when ENDLOCAL is executed memory may become fragmented (this is not usually harmful, but wastes memory).

## SHIFT

(Enhanced)

**Purpose:** Allows the use of more than 10 batch file parameters in batch files.

**Format:** SHIFT [*n* | /*n*]

***n*:** Number of positions to shift.

**Usage:** SHIFT is provided for compatibility with COMMAND.COM and CMD.EXE, where it is used to access more than 10 parameters. 4DOS, 4NT, and Take Command support 256 parameters (%0 to %255), so you may not need to use SHIFT for batch files running exclusively under JP Software command processors.

SHIFT moves each of the batch file parameters *n* positions to the left. The default value for *n* is 1. SHIFT 1 moves the parameter in %1 to position %0, the parameter in %2 becomes %1, etc. You can reverse a SHIFT by giving a negative value for *n*.

SHIFT also affects the special parameters %*n*\$ (command-line tail; %*n*& in 4DOS) and %# (number of command arguments).

For example, create a batch file called *TEST.BAT*:

```
echo %1 %2 %3 %4
shift
echo %1 %2 %3 %4
shift 2
echo %1 %2 %3 %4
shift -1
echo %1 %2 %3 %4
```

Executing *TEST.BAT* produces the following results:

```
[c:\] test one two three four five six seven
one two three four
two three four five
four five six seven
three four five six
```

- ❖ If you add a slash before the value *n*, the value determines the position at which to begin the shift. For example:

```
shift /2
```

leaves parameters %0 and %1 unchanged, and moves the value of %3 to position %2, %4 to %3, etc. The value after the slash cannot be negative, and shifts performed with the slash cannot be undone later in the batch file.

**SHORTCUT    [4NT, TC]**

(New)

**Purpose:**    Create a shortcut.

**Format:**    SHORTCUT command [[args] [dir] [desc] link mode]

***command:*** Command to execute or link file to display

***args:*** Command line arguments

***dir:*** Working directory

***desc:*** Link description

***link:*** Filename of the *.LNK* or *.PIF* file.

***mode:*** How to display the window (1=normal, 2=minimized, 3=maximized)

**Usage:**    SHORTCUT creates Windows shortcuts and places them in any folder on your system. You can run any Windows shortcut from Take Command or 4NT by entering the name of the *.LNK* or *.PIF* file on the command line.

If you only provide one argument (a link file name), SHORTCUT will display the values for that link.

Otherwise, SHORTCUT requires 6 arguments; to leave an argument blank, enter 2 double quotes ["] in its place. Other arguments must be enclosed in double quotes if they include white space or other special characters.

The ***command*** is the full path of the the executable file to start, or the data file or folder to open. If the ***command*** is a data file, its extension must be associated with an executable command (see ASSOC) for the shortcut to work.

The ***args*** argument lists any command-line arguments which you want to include when the command is executed. For example, if the command points to a batch file, you might want to include "%c" in the ***args*** so that the command processor exits immediately when the batch file is completed.

The ***dir*** argument is the path of the directory which you want Windows to switch to when the command starts. If you don't care which directory is used, you can omit this argument by entering "" in its place.

The ***desc*** provides a description that is stored internally in the shortcut. If you omit the description, enter two double quotes ["" ] in its place.

The ***link*** argument is the full path and filename of the resulting shortcut. If you include a filename but no path, the shortcut will be created in the current directory. The file extension must be ***.LNK***, unless you are creating a shortcut to a DOS command, in which case the extension must be ***.PIF***.

If you want the shortcut to appear on the Windows desktop, you should include the full path to the desktop folder in the command. In Windows 95 / 98 / ME, the desktop is usually stored in ***C:\WINDOWS\DESKTOP***; in Windows NT / 2000 / XP, the desktop is usually stored in ***C:\WINNT\DESKTOP***. To store the shortcut on the desktop regardless of the computer configuration or operating system, you can normally use ***%\_windir\desktop\*** as the path.

The final argument, ***mode***, determines how Windows will display the application or folder when you run the shortcut. It must be 1 for a normal window, 2 for a minimized window (normally placed on the taskbar), or 3 for a maximized window.

**SHRALIAS**    *[4NT, TC]*

(New)

**Purpose:**     Retains global alias, user-defined functions, command history, and directory history in memory when the command processor is not running.

**Format:**     SHRALIAS [/U]  
  
                  /U(nload)

**Usage:**     When you close all 4NT or Take Command sessions, the memory for the global alias, function, command history, and directory history lists are released. If you want the lists to be retained in memory even when no command processor session is running, you need to execute SHRALIAS.

The SHRALIAS command starts and initializes *SHRALIAS.EXE*, a small program which remains active and retains global lists when 4NT or Take Command is not running. *SHRALIAS.EXE* must be stored in the same directory as 4NT or Take Command, or in a directory on your PATH. You cannot run *SHRALIAS.EXE* directly, it must be run by the SHRALIAS command.

Once SHRALIAS has been executed, the global lists will be retained in memory until you use **SHRALIAS /U** to unload the lists, or until you shut down your operating system.

SHRALIAS will not work unless you have at least one copy of 4NT or Take Command running with global alias, command history, and directory history lists enabled. If the required global lists are not found, SHRALIAS will display an error.

If you start SHRALIAS from a temporary 4NT or Take Command session which exits after starting SHRALIAS, the 4NT or Take Command session may terminate and discard the shared lists before SHRALIAS can attach to them. In this case *SHRALIAS.EXE* will not be loaded. If you experience this problem, add a short delay with the DELAY command after SHRALIAS is loaded and before your session exits.

**4NT**            SHRALIAS does not work properly in detached sessions (e.g. those started with DETACH, or with Windows NT / 2000 / XP 's AT utility), due to security issues within Windows NT / 2000 / XP . The SHRALIAS command will be ignored in detached sessions.

For more information about global lists, see Local and Global Command History (page 65), Local and Global Directory History (page 75), and local and global alias lists on page 259.

**Option:**    **/U(nload):** Shuts down *SHRALIAS.EXE*. If *SHRALIAS* is not loaded again, the memory used by global alias, function, command history and directory history lists will be released when the last copy of 4NT or Take Command exits.



**SNPP** [4NT, TC] (New)

**Purpose:** Send messages to alphanumeric pagers.

**Format:** SNPP server pagerid message

***server***: The SNPP server name

***pagerid***: The ID of the pager to receive the message

***message***: The message to send

**Usage:** SNPP sends messages to alphanumeric pagers through standard Internet Paging Gateways.

**START** [4NT, TC]

(Enhanced)

**Purpose:** Start a program in another session or window.

**Format:** The format for START depends on which command processor you are using (see the Options section below for additional differences in valid options under 4NT and Take Command).

**TC** START ["*program title*"] [/ABOVENORMAL /BELOWNORMAL /C /CM /CMSTDIO /D*path* /FS /HIGH /I /INV /K /L /LD /LF /LH /LOW /MAX /MIN /NORMAL /PGM "*programe*" /POS=*x,y,width,height* /REALTIME /SEPARATE /SHARED /SIZE=*rows,cols* /WAIT] [*command*]

**4NT** START ["*program title*"] [/B /C /D*path* /FS /HIGH /I /INV /K /L /LA /LD /LF /LH /LOW /MAX /MIN /NORMAL /PGM *programe* /POS=*x,y,width,height* /REALTIME /SEPARATE /SHARED /SIZE=*rows,columns* /WAIT] [*command*]*program*

***title*:** Title to appear on title bar.

***path*:** Startup directory.

***programe*:** Program name (not the session name).

***command*:** Command to be executed.

<b>/ABOVENORMAL</b>	<b>/LD</b> (local directory history)
<b>/B</b> (no new console)	<b>/LF</b> (local function list)
<b>/BELOWNORMAL</b>	<b>/LH</b> (local history list)
<b>/C</b> (lose when done)	<b>/LOW</b> (low priority)
<b>/CM</b> (Caveman)	<b>/MAX</b> (imized)
<b>/CMSTDIO</b> (Caveman stdio)	<b>/MIN</b> (imized)
<b>/D</b> (irectory)	<b>/NORMAL</b> (priority)
<b>/FS</b> (full screen)	<b>/PGM</b> (program name)
<b>/HIGH</b> (priority)	<b>/POS</b> (ition of window)
<b>/I</b> (nherit environment)	<b>/REALTIME</b> (priority)
<b>/INV</b> (isible)	<b>/SEPARATE</b> (separate session)
<b>/K</b> (eep when done)	<b>/SHARED</b> (shared session)
<b>/L</b> (ocal lists)	<b>/SIZE</b> (screen buffer size)
<b>/LA</b> (local aliases)	<b>/WAIT</b> (for session to finish)

See also: DETACH.

**Usage:** **START** is used to begin a new session and optionally run a program in that session. If you use **START** with no parameters, it will begin a new 4NT or Take Command session. If you add a ***command***, **START** will begin a new session and execute that command.

**START** will return to the 4NT or Take Command prompt immediately (or continue a batch file), without waiting for the program to complete, unless you use **/CM** or **/CMSTDIO** (Take Command only), or **/WAIT**.

The ***program title***, if it is included, will appear on the task list and Alt-Tab displays. The ***program title*** must be enclosed in quotation marks and cannot exceed 80 characters. If the ***program title*** is omitted, the program name will be used as the title.

**START** always assumes that the first quoted string on the command line is the ***program title***; if there is a second quoted string it is assumed to be the ***command***. If the name of the program you are starting contains white space (and must therefore be quoted), you cannot simply place it on the command line. If you do, as the first quoted string it will be interpreted as the ***program title***, not the ***command***. To address this, use the **/PGM** switch to indicate explicitly that the quoted string is the program name, or include a title before the program name. For example, to start the program "*C:\Program Files\Proc.Exe*" you could use either of the first two commands below, but the third command would not work:

```
[c:\] start /PGM "C:\Program Files\Proc.Exe"  
[c:\] start "test" "C:\Program Files\Proc.Exe"  
[c:\] start "C:\Program Files\Proc.Exe"
```

**START** will look for the program name in the "App Paths" registry key, and will insert the "Path" value if it exists at the beginning of the **PATH** in the environment inherited by the program.

**START** offers a large number of switches to control the session you start. The list below summarizes the most commonly used **START** options, and how to use them to control the way a session is started:

**/MAX**, **/MIN**, and **/POS** allow you to start a character-mode windowed session in a maximized window, a minimized window, or a window with a specified position and size. The default is to let the operating environment choose the position and size of the window.

**/C** allows you to close the session when the command is finished (the default for GUI sessions); **/K** allows you to keep the session open and go to a prompt (the default for character mode sessions).

If **progrname** is the name of a directory, the command processor will start Windows Explorer in the specified directory. (Explorer must be in the PATH, the `\WINDOWS` directory, or the `\WINDOWS\SYSTEM` directory for this feature to work correctly.)

**Options:**    The options below are not in strictly alphabetical order. Instead, they are divided first by product, then alphabetically.

The following options are available in both 4NT and Take Command:

**/ABOVENORMAL:** (Windows 2000 and XP) Set the priority above normal.

**/BELOWNORMAL:** (Windows 2000 and XP) Set the priority below normal.

**/C(lose):** Close the session when the application ends.

**/D (Directory):** Specifies the startup directory. Include the directory name immediately after the **/D**, with no intervening spaces or punctuation. Due to limitations in the way Windows starts DOS programs, **/D** is ignored when starting DOS applications.

**/FS:** Start the console application in full-screen mode.

**/HIGH:** Start the window at high priority.

**/I(nherit environment):** Inherit the system default environment, if any, rather than the current environment.

**/INV(isible):** Start the session or window as invisible. No icon will appear and the session will only be accessible through the Task Manager or Window List.

**/K(eep session or window at end):** Continue the session or window after the application program ends. Use the EXIT command to end the session.

**/L(ocal lists):** Start 4NT or Take Command with local alias and history lists. This option combines the effects of **/LA**, **/LD**, **/LF**, and **/LH** (below).

**/LA** (Local Alias list): Start 4NT or Take Command with a local alias list. See page 259 for information on local and global aliases.

**/LD** (Local directory history list): Start 4NT or Take Command with a local directory history list. See page 75 for information on local and global directory history.

**/LF** (Local function list): Start 4NT or Take Command with a local user-defined function list.

**/LH** (Local History list): Start 4NT or Take Command with a local history list. See page 65 for information on local and global history lists.

**/LOW**: Start the window at low priority.

**/MAX(imized)**: Start the session or window maximized.

**/MIN(imized)**: Start the session or window minimized.

**/NORMAL**: Start the window at normal priority.

**/REALTIME**: Start the window at real-time priority.

**/PGM** (program name): The quoted string following this option is the program name. Any additional text beyond the quoted string is passed to the program as its arguments, so to use other **START** switches you must place them before **/PGM**. You can use **/PGM** to allow **START** to differentiate between a quoted long filename and a quoted title for the session.

**/POS(ition)**: Start the window at the specified screen position. The syntax is **/POS=x, y, width, height** where the values are specified in pixels or pels. **X** and **Y** refer to the position of the top left corner of the window relative to the top left corner of the screen.

**/SEPARATE**: Start a 16-bit Windows application in a separate virtual machine. Normally, all 16-bit Windows applications are started in the same virtual machine.

**/SHARED**: Start a 16-bit Windows application in the shared virtual machine (the opposite of **/SEPARATE**). This is the default; it is included for compatibility with *CMD.EXE*.

**/SIZE:** Start the window with the specified screen buffer size. The full syntax is **/SIZE=rows, columns**, where **rows** is the number of text rows and **columns** is the number of text columns.

**/WAIT:** Wait for the new session or window to finish before continuing. Under Windows 95 / 98 / ME, Take Command always waits for DOS programs run in its console window (programs that are not started from *.PIF* files). See the *Introduction and Installation Guide* or the online help for details.

**TC**        The following options are only available in Take Command:

**/CM** (Caveman): Run a DOS or character-mode application under Caveman. Use this option to force an application to run under Caveman even if Caveman is not specifically enabled for that application. For more details on Caveman see the online help. **/CM** will be ignored if Caveman is not enabled (via the Configure Caveman Apps dialog on the **Options** menu). Take Command always waits for the application to finish when **/CM** is used, even if you do not use **/WAIT**.

**/CMSTDIO:** Run a console application under Caveman, using STDIO mode. **/CMSTDIO** will be ignored if Caveman is not enabled (via the Configure Caveman Apps dialog on the **Options** menu). Take Command always waits for the application to finish when **/CMSTDIO** is used, even if you do not use **/WAIT**.

**4NT**        The following option is only available in 4NT:

**/B:** The program is started without creating a new window or console, *i.e.* in the 4NT window. Normally, the application is started in its own window. For compatibility with *CMD.EXE*, **/B** also disables Ctrl-C processing for the program.

**SWAPPING** *[4DOS]*

(New)

- Purpose:** Enable or disable 4DOS swapping, or display the swapping state.
- Format:** SWAPPING [ON | OFF]
- ❖ **Usage:** SWAPPING temporarily disables or enables the swapping of the transient portion of 4DOS to expanded memory, to XMS extended memory, or to disk (see page 214).

Setting **SWAPPING OFF** may be useful for speeding up batch files (including *AUTOEXEC.BAT*) when 4DOS is using disk swapping. When you are running several small programs from a batch file, disk swapping can sometimes cause a noticeable delay. However, if you disable swapping, there will be about 200K less memory available for large application programs.

The following batch file fragment disables swapping, runs several programs, and then re-enables swapping:

```
swapping off
c:\util\mouse
c:\video\ansi.com
cls bright white on blue
c:\bin\cache.com
swapping on
```

If you enter **SWAPPING** with no arguments, 4DOS displays the current swapping type (XMS, EMS, Disk, or None) and state:

```
c:\> swapping
SWAPPING (XMS) is ON
```

Setting **SWAPPING OFF** does not close the disk swap file or release any reserved EMS or XMS memory.

You may have trouble if you load memory-resident programs (TSRs) with **SWAPPING OFF** and unload them with **SWAPPING ON**, or vice versa. Many TSRs expect the system to be in the same state when they unload that it was in when they loaded, and variation from this norm may cause the TSR to unload improperly or hang your system.

**SWITCH**

(New)

**Purpose:** Select commands to execute based on a value.

**Format:** SWITCH *expression*

CASE *value1* [.OR. *value2*] ...

    commands

CASE *value3*

    commands

[DEFAULT

    commands]

ENDSWITCH

***expression:*** An environment variable, internal variable, variable function, text string, or a combination of these elements, that is used to select a group of commands.

***value1, value2, etc.:*** A value to test or multiple values connected with .OR.

***commands:*** One or more commands to execute if the expression matches the value. If you use multiple commands, they must be separated by command separators or placed on separate lines of a batch file.

See also: IF and IFF.

**Usage:** SWITCH can only be used in batch files. It allows you to select a command or group of commands to execute based on the possible values of a variable or a combination of variables and text.

The SWITCH command is always followed by an ***expression*** created from environment variables, internal variables, variable functions, and text strings, and then by a sequence of CASE statements matching the possible ***values*** of that ***expression***. If one of the ***values*** in a CASE statement matches the ***expression***, the commands following that CASE statement are executed, and all subsequent CASE statements and the commands which follow them are ignored.



If no matches are found, the commands following the optional **DEFAULT** statement are executed. If there are no matches and there is no **DEFAULT** statement, no commands are executed by **SWITCH**.

After all of the commands following the **CASE** or **DEFAULT** statement are executed, the batch file continues with the commands that follow **ENDSWITCH**.

You must include a command separator or new line after the **expression**, before each **CASE** or **DEFAULT** statement, before each command, and before **ENDSWITCH**. You can link values in a **CASE** statement only with **.OR.** (but not with **.AND.** or **.XOR.**).

For example, the following batch file fragment displays one message if the user presses **A**, another if user presses **B** or **C**, and a third if the user presses any other key:

```
inkey Enter a keystroke: %%key
switch %key
case A
    echo It's an A
case B .or. C
    echo It's either B or C
default
    echo It's not A, B, or C
endswitch
```

In the example above, the value of a single environment variable was used for the **expression**. You will probably find that this is the best method to use in most situations. However, you can use other kinds of expressions if necessary. The first example below selects a command to execute based on the length of a variable, and the second bases the action on a quoted text string stored in an environment variable:

```
switch %@len[%var1]
case 0
    echo Missing var1
case 1
    echo Single character
...
endswitch

switch "%string1"
case "This is a test"
    echo Test string
case "The quick brown fox"
```

```
    echo It's the fox
    ...
endswitch
```

The SWITCH and ENDSWITCH commands must be on separate lines, and cannot be placed within a command group (see page 117), or on the same line as other commands (this is the reason SWITCH cannot be used in aliases). However, commands within the SWITCH block can use command groups or the command separator in the normal way.

SWITCH commands can be nested. Under 4DOS, the permissible nesting level depends on the amount of free space in 4DOS's internal stack; if you receive a stack overflow error when using SWITCH in complex, nested command sequences, see the notes under the StackSize directive on page 238. You can exit from all SWITCH / ENDSWITCH processing by using GOTO to a line past the last ENDSWITCH.

**TAIL**

(New)

**Purpose:** Display the end of the specified file(s).

**Format:** TAIL [/A:[-][+]*rhsad*] /C*n* /I"text" /N*n* /P /Q /V] [*@file*] *file*...

***file*:** The file or list of files that you want to display.

***@file*:** A text file containing the names of the files to display, one per line (see page 109 for details).

/A: (Attribute select)	/P(ause)
/C (number of bytes)	/Q(quiet)
/I"text" (match description)	/V(erbose)
/N(umber of lines)	

See also: HEAD, LIST, TYPE.

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104).

**Internet:** Can be used with FTP servers.

**Usage:** The TAIL command displays the last part of a file. It is normally only useful for displaying ASCII text files. Executable files (.COM and .EXE) and many data files may be unreadable when displayed with TAIL because they include non-alphanumeric characters. You can press **Ctrl-S** to pause TAIL's display and then any key to continue.

To display the last 15 lines of the files *MEMO1* and *MEMO2*:

```
[c:\] tail /n15 memo1 memo2
```

To display text from the clipboard use **CLIP:** as the file name. CLIP: will not return any data if the clipboard does not contain text. See page 88 for additional information on CLIP:.

#### **4NT, TC FTP Usage**

TAIL can also display files on FTP servers. The URL must be enclosed in double quotes so the forward slashes won't be interpreted as switches. For example:

```
[c:\] tail "ftp://jpsoft.com/index"
```

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see Using FTP Servers on page 113, and the IFTP command on page 382.

#### **4NT, TC NTFS File Streams**

TAIL supports file streams on NTFS drives under Windows NT / 2000 / XP. You can type an individual stream by specifying the stream name, for example:

```
[c:\] tail streamfile:s1
```

If no stream name is specified the file's primary data is displayed.

See NTFS File Streams on page 18 for additional details.

Options:     ❖**A:** (Attribute select): Select only those files that have the specified attribute(s) set. See **Attribute Switches** on page 111 for information on the attributes which can follow **/A:**.

**/C:** Display the specified number of bytes. **/C** accepts a **b**, **k**, or **m** at the end of the number. **B** is the number of 512-byte blocks, **k** is thousands of bytes, **K** is kilobytes, **m** is millions of bytes, and **M** is megabytes.

**/I"text":** Select files by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**/N:** The number of lines to display. The default is 10.

**/P(ause):** Prompt after displaying each page. Your options at the prompt are explained in detail on page 80.

**/Q(quiet):** Don't display a header for each file.

**/V(erbos):** Display a header for each file.

## **TASKEND**    [4NT, TC]

(New)

**Purpose:**    End the specified process.

**Format:**    TASKEND [/F] pid | name | "title"

***pid*:** Process ID

***name*:** Process name

***title*:** Window title

**/F**(orce).

**Usage:**    Windows applications (and Windows itself) run as one or more processes or tasks. You can use the TASKLIST command to display a list of currently-running tasks. TASKEND can be used to end a task.

When you use TASKEND, you must specify the task you want to end by process ID number, by name (usually the name of the executable file that started the task) or by window title. If you use the Window title to specify the task, you must enclose it in double quotes. You can use wild cards and extended wildcards in the window title (see page 94).

If you use TASKEND without the /F option, the effect is much the same as closing a window by clicking the close button. The application is notified of the request to end the task and has an opportunity to save data, prompt whether you mean to shut down, and perform other normal "close" operations.

- !    If you use the /F option with TASKEND, the application is shut down abruptly and has no chance to save data. Use of the /F option is only recommended for unusual circumstance and advanced users because of the possibility of data loss.

**Option:**    **/F**(orce): Forces the task or application to end immediately, with no opportunity to save data, prompt the user, etc. Use this option with caution; it can possibly lead to system instability and data loss or corruption.

**TASKLIST** [4NT, TC]

(New)

**Purpose:** Display a list of currently running processes.

**Format:** TASKLIST [/P] [name]

***name:*** Process name or window title.

**/P(ause)**

**Usage:** Windows applications (and Windows itself) run as one or more processes or tasks. You can use the TASKLIST command to display a list of currently-running tasks.

TASKLIST displays the process ID number for each running task, the name of the executable program that started the task, and, when available, the window title.

You can limit the output of TASKLIST by specifying the taskname that you wish to see. The name can contain wildcards and extended wildcards (see page 94). Only those tasks whose name matches the name you give will be displayed. For example, to list all the tasks with the string “office” anywhere in their name:

```
[c:\] tasklist *office*
```

**Option:** **/P(ause):** Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail on page 80.

## TCTOOLBAR [TC]

(New)

**Purpose:** Change the tool bar buttons.

**Format:** TCTOOLBAR [/U] button [, *flags*, *text*, *command*]

***button*:** The button number (1 – 32)

***flags*:** 0=Echo, 1=Echo and Execute, 2=Execute without echo

***text*:** The button text

***command*:** The command line to execute.

**/U**(pdate)

**Usage:** TCTOOLBAR lets you configure the Take Command tool bar buttons (you can also use the Configure Tool Bar dialogs available from the Options menu). The changes you make can be temporary or, with the /U option, written to the *TCMD32.INI* file so that they will be loaded the next time Take Command starts.

There are a maximum of 32 buttons on the tool bar. The ***button*** argument must be a number from 1 to 32 to select the button you want to work with.

If you enter a command with no other parameters, for example:

```
[c:\] tctoolbar 1
```

the button with that number, if it is currently visible, will be removed from the tool bar. If you want to add or modify a button, you must include the ***flags***, ***header***, and ***command*** parameters in the command.

The ***flags*** parameter specifies what happens when you click the button. If it is 0, the button's command is added to the command line at the current cursor position, but the command line is not executed immediately. You can then edit the command, add additional options and parameters, etc., before you press Enter to execute it (or Esc to cancel the action). If the ***flags*** parameter is set to 1, the command text is added to the command line and then the entire command line is immediately executed. If it is 2, Take Command adds the button's command to its in-memory copy of the command line and then executes that, without actually displaying the text.

The ***text*** parameter specifies the text that appears on the button. If the text contains white space or other special characters, it must be enclosed in double quotes.

The ***command*** parameter contains the text that is placed on the command line and / or executed when the button is clicked. It may be any internal or external command, an alias name, batch file name, or any other text that you want added to the command line. The first part of the command – the command or program name -- should be quoted if it contains white space or other special characters.

Option: /U(pdate): Write the changed button definition to the *TCMD32.INI* file so that it will be reloaded automatically the next time Take Command starts.



**TEE**

(New)

**Purpose:** Copy standard input to both standard output and a file.

**Format:** TEE [/A] *file*...

***file*:** One or more files that will receive the “tee-d” output.

/A(ppend)

See also: Y, and the redirection options (page 88).

❖ **Usage:** TEE is normally used to “split” the output of a program so that you can see it on the display and also save it in a file. It can also be used to capture intermediate output before the data is altered by another program or command.

TEE gets its input from standard input (usually the piped output of another command or program), and sends out two copies: one goes to standard output, the other to the ***file*** or ***files*** that you specify. TEE is not likely to be useful with programs which do not use standard output, because these programs cannot send output through a pipe. See page 91 for more information on pipes.

For example, to search the file *DOC* for any lines containing the string “Take Command”, make a copy of the matching lines in *TC.DAT*, sort the lines, and write them to *TCS.DAT* (enter this on one line):

```
[c:\] find "Take Command" doc | tee tc.dat | sort >
tcs.dat
```

If you are typing at the keyboard to produce the input for TEE, you must enter a **Ctrl-Z** to terminate the input.

**4DOS** When using TEE with a pipe under 4DOS, the previous command writes its output to a temporary file. When that command finishes, TEE reads the temporary file, displays the output, and writes it to the file(s) named in the TEE command.

**4NT, TC** When using TEE with a pipe under 4NT or Take Command, the programs on the two ends of the pipe run simultaneously. See page 92 for details.

**Option:** /A(ppend): Append to the file(s) rather than overwriting them.

**TEXT**

(New)

**Purpose:** Display a block of text in a batch file.

**Format:** **TEXT**

.

.

**ENDTEXT**

See also: ECHO, SCREEN, SCRPUT, and VSCRPUT.

**Usage:** TEXT can only be used in batch files.

The TEXT command is useful for displaying menus or multi-line messages. TEXT will display all subsequent lines in the batch file until terminated by ENDTEXT. Both TEXT and ENDTEXT must be entered as the only command on the line.

To redirect the entire block of text, use redirection on the TEXT command itself, but not on the actual text lines or the ENDTEXT line. No environment variable expansion or other processing is performed on the lines between TEXT and ENDTEXT; they are displayed exactly as they are stored in the batch file.

If you are running 4DOS with an ANSI driver loaded, or if you have enabled ANSI support in 4NT or Take Command, you can change screen colors by inserting ANSI escape sequences anywhere in the text block. You can also use a CLS or COLOR command to set the screen color before executing the TEXT command.

The following batch file fragment displays a simple menu:

```
@echo off
cls
screen 2 0
text
Enter one of the following:
1 - Spreadsheet
2 - Word Processing
3 - Utilities
4 - Exit
endtext
inkey /k"1234" Enter your selection: %%key
```

**TIME**

(Enhanced)

**Purpose:** Display or set the current system time.

**Format:** TIME [/S [server] /T] [*hh*[:*mm*[:*ss*]]] [AM | PM]

***hh***: The hour (0 - 23).

***mm***: The minute (0 - 59).

***ss***: The second (0 - 59).

**/S**(erver time)

**/T** (Display only)

See also: DATE

**Usage:** If you don't enter any parameters, TIME will display the current system time and prompt you for a new time. Press **Enter** if you don't wish to change the time; otherwise, enter the new time:

```
[c:\] time
Thu Oct 31, 2002 9:30:06
New time (hh:mm:ss):
```

TIME defaults to 24-hour format, but you can optionally enter the time in 12-hour format by appending “a”, “am”, “p”, or “pm” to the time you enter.

For example, to enter the time as 9:30 am:

```
[c:\] time 9:30 am
```

The operating system adds the system time and date to the directory entry for every file you create, modify, or access. If you keep both the time and date accurate, you will have a record of when you last updated each file.

**Option:** **/T**: Displays the current time but does not prompt you for a new time. You cannot specify a new time on the command line with /T. If you do, the new time will be ignored.

**4NT, TC** **/S**: Sets the date and time from the specified internet time server. If no server is specified, TIME uses the server specified in the TimeServer .INI entry (the default is **clock.psu.edu**).

**TIMER**

(New)

**Purpose:** TIMER is a system stopwatch.

**Format:** TIMER [ON] [/1 /2 /3 /S]

**ON:** Force the stopwatch to restart

**/1** (stopwatch #1)

**/3** (stopwatch #3)

**/2** (stopwatch #2)

**/S**(plit)

**Usage:** The TIMER command turns a system stopwatch on and off. When you first run TIMER, the stopwatch starts:

```
[c:\] timer
Timer 1 on: 12:21:46
```

When you run TIMER again, the stopwatch stops and the elapsed time is displayed:

```
[c:\] timer
Timer 1 off: 12:21:58
Elapsed time: 0:00:12.06
```

There are three stopwatches available (1, 2, and 3) so you can time multiple overlapping events. By default, TIMER uses stopwatch #1.

TIMER is particularly useful for timing events in batch files. For example, to time both an entire batch file, and an intermediate section of the same file, you could use commands like this:

```
rem Turn on timer 1
timer
rem Do some work here
rem Turn timer 2 on to time the next section
timer /2
rem Do some more work
echo Intermediate section completed
rem Display time taken in intermediate section
timer /2
rem Do some more work
rem Now display the total time
timer
```

The smallest interval TIMER can measure depends on the operating system you are using, your hardware, and the interaction between the two. However, it should never be greater than .06 second.

**Options:**    **/1:** Use timer #1 (the default).

**/2:** Use timer #2.

**/3:** Use timer #3.

**/S(plit):** Display a split time without stopping the timer. To display the current elapsed time but leave the timer running:

```
[c:\] timer /s  
Timer 1 elapsed: 0:06:40.63
```

**ON:** Start the timer regardless of its previous state (on or off).

Otherwise the **TIMER** command toggles the timer state (unless **/S** is used).

**TITLE** [4NT, TC] (Compatible)

**Purpose:** Change the window title.

**Format:** TITLE *title*

***title:*** The new window title

See also: ACTIVATE and WINDOW.

**Usage:** TITLE changes the text that appears in the caption bar at the top of the 4NT or Take Command window. It is included for compatibility with *CMD.EXE*. You can also change the window title with the WINDOW command or the ACTIVATE command.

The title text should not be enclosed in quotes unless you want the quotes to appear as part of the actual title.

To change the title of the current window to "Take Command Test":

```
[c:\] title Take Command Test
```

## TOUCH

(New)

**Purpose:** Change a file's date and/or time stamps.

**Format:** TOUCH [/A:[-][+]rhsad] /C /D[acw][mm-dd-yy] /E /F /I"text" /N /Q  
/R[:acw] *reffile* /S /T[:acw][hh:mm]] [*@file*] *file...*

***file*:** One or more files whose date and / or time stamps are to be changed.

***reffile*:** A file whose date and / or time stamps are to be transferred to one or more other files.

***@file*:** A text file containing the names of the files to touch, one per line (see page 109 for details).

/C(reate file)

/N(othing)

/D(ate)

/Q(quiet)

/E (No error messages)

/R(eference file)

/F(orce read-only files)

/S(ubdirectories)

/I (match descriptions)

/T(ime)

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104). Use extended wildcards with caution on LFN volumes; see page 105 for details.

**Usage:** TOUCH is used to change the date and / or time of a file. You can use it to be sure that particular files are included or excluded from an internal command, backup program, compiler MAKE utility , or other program that selects files based on their time and date stamps, or to set a group of files to the same date and time for consistency.

- ! TOUCH should be used with caution, and in most cases should only be used on files you create. Many programs depend on file dates and times to perform their work properly. In addition, many software manufacturers use file dates and times to signify version numbers. Indiscriminate changes to date and time stamps can lead to confusion or incorrect behavior of other software.

TOUCH normally works with existing files, and will display an error if the ***file*** you specify does not exist, or has the read-only attribute set. To create the ***file*** if it does not already exist, use the /C switch. To force a date and time change for read-only files, use the /F switch.

TOUCH displays the date, time, and full name of each file whose timestamp is modified. To disable this output, use **/Q**.

If you don't specify a date or a time, TOUCH will default to the current date and time. For example, to set the time stamp of all **.C** files in the current directory to the current date and time:

```
[d:\source] touch *.c
6-12-2002 11:13:58 D:\SOURCE\MAIN.C
6-12-2002 11:13:58 D:\SOURCE\INIT.C
...
```

If you specify a date but not a time, the time will default to the current time from your system clock. Similarly, if you specify a time but not a date, the date will be obtained from the system clock.

To transfer the date and time from one file to another use **/R**; see below for details.

**4NT, TC** TOUCH can also set the date and time for directories if you use the **/A:d** switch and specify a directory name. However, due to operating system limitations this works only under Windows NT / 2000 / XP; it will not work under Windows 95 / 98 / ME.

On LFN files, TOUCH sets the “modified” or “last write” date and time by default. By adding an **a**, **c**, or **w** to the **/D** or **/T** switch, you can set the last access, creation, or last write date and time stamps that are maintained for each file; see page 17 and the **Options** section below for additional details.

Options: **\*/A:** (Attribute select): Select only those files that have the specified attribute(s) set. See **Attribute Switches** on page 111 for information on the attributes which can follow **/A:**.

**/C**(reate file): Create the **file** (as a zero-byte file) if it does not already exist. You cannot use wildcards with **/C**, but you can create multiple **files** by listing them individually on the command line.

**/D**(ate): Specify the date that will be set for the selected files. If the date is not specified, TOUCH will use the existing file date. For LFN files use **/Da**, **/Dc**, or **/Dw**, followed by the date, to specify the last access, creation, or last write date stamp. The date must be entered using the proper format for your current country settings. You can also use the international date format **yyyy-mm-dd**.



- ❖ **/E** (No error messages): Suppress all non-fatal error messages, such as "File not found." Fatal error messages, such as "Drive not ready," will still be displayed.

**/F**(orce read-only files): Remove the read-only attribute from each file before changing the date and time, and restore it afterwards. Without **/F**, attempting to change the date and time on a read-only file will usually cause an error.

**/I"text"**: Select files by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**/Q**(uiet): Do not display the new date and time and the full name for each file.

**/R**: The date and time for the second (and any additional) file will be set to the current date and time for the reference file whose name immediately follows the **/R**. For LFN files, you can use **/R:a**, **/R:c**, or **/R:w**, followed by the file name, to specify the last access, creation, or last write time stamp. However, files on FAT, VFAT and FAT32 volumes do not have a last access time, so **TOUCH /R:a** will have no effect on such files.

If **/D** or **/T** is used after **/R**, the specified date or time will override the date or time from the reference file. For example, to set the date for file X2 to match the date for X1, and also set the time for X2 to 11:42 AM, you could use:

```
[c:\] touch /r x1 /t11:42 x2
```

**/S**(ubdirectories): **TOUCH** all matching files in the specified directory and its subdirectories.

**/T**(ime): Specify the time that will be set for the selected files in hh:mm format. If the time is not specified, **TOUCH** will use the existing file time. For LFN files, you can use **/Ta**, **/Tc**, or **/Tw**, followed by the time, to specify the last access, creation, or last write time stamp. However, files on FAT, VFAT and FAT32 volumes do not have a last access time, so **TOUCH /Ta** will have no effect on such files.

**TREE**

(New)

**Purpose:** Display a graphical directory tree.

**Format:** TREE [/A /B /F /H /P /S /T[:acw]] *dir...*

***dir*:** The directory to use as the start of the tree. If more than one directory is specified, TREE will display a directory tree for each.

/A(SCII)

/P(ause)

/B(are)

/S (file size)

/F(iles)

/T(ime and date)

/H(idden directories)

**Files:** Supports extended wildcards, ranges, multiple directory names, and include lists (see pages 94 - 104).

**Usage:** The TREE command displays a graphical representation of the directory tree using standard or extended ASCII characters. For example, to display the directory structure on drive C:

```
[c:\] tree c:\
```

TREE uses the standard line drawing characters in the U.S. English extended ASCII character set. If your system is configured for a different country or language, or if you use a font which does not include these line drawing characters, the connecting lines in the tree display may not appear correctly on your screen. To correct the problem, use /A, or configure the command processor to use a font, such as Terminal, which contains standard extended ASCII characters.

You can print the display, save it in a file, or view it with LIST by using standard redirection symbols (see page 88). Be sure to review the /A option before attempting to print the TREE output. The options, discussed below, specify the amount of information included in the display.

**Options:** /A(SCII): Display the tree using standard ASCII characters. You can use this option if you want to save the directory tree in a file for further processing or print the tree on a printer which does not support the graphical symbols that TREE normally uses.

/B(are): Display the full pathname of each directory, without any of the line-drawing characters.

**/F(iles):** Display files as well as directories. If you use this option, the name of each file is displayed beneath the name of the directory in which it resides.

**/H(idden):** Display hidden as well as normal directories. If you combine **/H** and **/F**, hidden files are also displayed.

**/P(ause):** Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail on page 80.

**/S(ize):** Display the size of each file. This option is only useful when combined with **/F**.

**/T(ime and date):** Display the time and date for each directory. If you combine **/T** and **/F**, the time and date for each file will also be displayed. For LFN files, the time and date of the last write will be shown by default. You can select a specific time and date stamp by using the following variations of **/T** (see page 17 for more details on date and time stamps):

**/T:a** Last access date and time (access time will always be 00:00 on VFAT and FAT32 volumes).

**/T:c** Creation date and time.

**/T:w** Last write date and time (default).

**TRUENAME**

(New)

**Purpose:** Find the full, true path and file name for a file.

**Format:** TRUENAME *file*

***file*:** The file whose name TRUENAME will report.

See also: @TRUENAME variable function on page 187.

**Usage:** Default directories, as well as the JOIN and SUBST external commands, can obscure the true name of a file. TRUENAME “sees through” these obstacles and reports the fully qualified name of a file.

The following example uses TRUENAME to get the true pathname for a file:

```
[c:\] subst d: c:\util\test
[c:\] truenam d:\test.exe
c:\util\test\test.exe
```

**4DOS** To use TRUENAME under 4DOS you must be running MS-DOS or PC DOS 3.0 or above, DR-DOS 6.0 or above, or Windows 95 / 98 / ME.

On LFN drives TRUENAME returns the short name for the file, for example:

```
c:\> truenam "Program Files"
C:\PROGRA~1
```

TRUENAME can handle simple drive substitutions such as those created by JOIN, SUBST, or most network drive mappings. However, it may not be able to correctly determine the true name if you use “nested” JOIN or SUBST commands, or a network which does not report true names properly.

**TYPE**

(Enhanced)

**Purpose:** Display the contents of the specified file(s).

**Format:** TYPE [/A:[-][+]*rhsad*] /I"text" /L /P] [*@file*] *file...*

***file*:** The file or list of files that you want to display.

***@file*:** A text file containing the names of the files to type, one per line (see page 109 for details).

**/A:** (Attribute select)                      **/L**(line numbers)

**/I"text"** (match description)              **/P**(ause)

See also: LIST.

**Files:** Supports extended wildcards, ranges, multiple file names, and include lists (see pages 94 - 104).

**Internet:** Can be used with FTP servers.

**Usage:** The TYPE command displays a file. It is normally only useful for displaying ASCII text files. Executable files (.COM and .EXE) and many data files may be unreadable when displayed with TYPE because they include non-alphanumeric characters. You can press **Ctrl-S** to pause TYPE's display and then any key to continue.

To display the files *MEMO1* and *MEMO2*:

```
[c:\] type /p memo1 memo2
```

To display text from the clipboard use **CLIP:** as the file name. CLIP: will not return any data if the clipboard does not contain text. See page 88 for additional information on CLIP:.

You will probably find LIST to be more useful for displaying files on the screen. The TYPE /L command used with redirection (see page 88) is useful if you want to add line numbers to a file, for example:

```
[c:\] type /l myfile > myfile.num
```

**4NT, TC      *FTP Usage***

TYPE can also display files on FTP servers. The URL must be enclosed in double quotes so the forward slashes won't be interpreted as switches. For example:

```
[c:\] type "ftp://jpsoft.com/index"
```

You can also use the IFTP command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want. For more information, see *Using FTP Servers* on page 113, and the IFTP command on page 382.

**4NT, TC      *NTFS File Streams***

TYPE supports file streams on NTFS drives under Windows NT / 2000 / XP. You can type an individual stream by specifying the stream name, for example:

```
[c:\] type streamfile:s1
```

If no stream name is specified the file's primary data is displayed.

See *NTFS File Streams* on page 18 for additional details.

**Options:**      ♦ **/A:** (Attribute select): Select only those files that have the specified attribute(s) set. See **Attribute Switches** on page 111 for information on the attributes which can follow **/A:**.

**/I"text":** Select files by matching text in their descriptions. The text can include wildcards and extended wildcards. The search text must be enclosed in quotation marks, and must follow the **/I** immediately, with no intervening spaces. You can select all filenames that have a description with **/I"[?]\*"**, or all filenames that do not have a description with **/I"[]"**.

**/L(line numbers):** Display a line number preceding each line of text.

**/P(ause):** Prompt after displaying each page. Your options at the prompt are explained in detail on page 80.

## UNALIAS

(New)

**Purpose:** Remove aliases from the alias list.

**Format:** UNALIAS [/Q /R *file...*] *alias...*

or

UNALIAS \*

***alias*:** One or more aliases to remove from memory.

***file*:** One or more files to read for alias names to remove.

**/Q**(uiet)

**/R**(ead file)

See also: ALIAS and ESET.

**Usage:** 4DOS, 4NT, and Take Command maintain a list of the aliases that you have defined. The UNALIAS command will remove aliases from that list. UNALIAS supports wildcards in the alias name, and you can delete the entire alias list by using the command **UNALIAS \***.

For example, to remove the alias DDIR:

```
[c:\] unalias ddir
```

To remove all the aliases:

```
[c:\] unalias *
```

To remove all the aliases that begin with "DD":

```
[c:\] unalias dd*
```

If you keep aliases in a file that can be loaded with the ALIAS /R command (see page 256), you can remove the aliases by using the UNALIAS /R command with the same file name:

```
[c:\] unalias /r alias.lst
```

This is much faster than removing each alias individually in a batch file, and can be more selective than using UNALIAS \*.

**TC** You can also modify aliases with the Alias dialog (see page 55).

**Options:**    **/Q(quiet):** Prevents UNALIAS from displaying an error message if one or more of the aliases does not exist. This option is most useful in batch files, for removing a group of aliases when some of the aliases may not have been defined.

**/R(ead):** Read the list of aliases to remove from a file. The file format should be the same format as that used by the ALIAS /R command (see page 256). You can use multiple files with one UNALIAS /R command by placing the names on the command line, separated by spaces:

```
[c:\] unalias /r alias1.lst alias2.lst
```



## UNFUNCTION

(New)

**Purpose:** Remove user-defined functions from the function list.

**Format:** UNFUNCTION [/Q /R *file...*] *function...*

or

UNFUNCTION \*

***file:*** One or more files to read for function definitions to remove.

***function:*** One or more functions to remove from memory.

**/Q**(uiet)

**/R**(ead file)

See also: FUNCTION.

**Usage:** 4DOS, 4NT, and Take Command maintain a list of the functions that you have defined. The UNFUNCTION command will remove functions from that list. UNFUNCTION supports wildcards in the function name, and you can delete the entire function list by using the command **UNFUNCTION \***.

For example, to remove the function ZZ3:

```
[c:\] unfunction zz3
```

To remove all the user-defined functions:

```
[c:\] unfunction *
```

To remove all the functions that begin with "ZZ":

```
[c:\] unfunction zz*
```

If you keep functions in a file that can be loaded with the FUNCTION /R command (see page 256), you can remove the functions by using the UNFUNCTION /R command with the same file name:

```
[c:\] unfunction /r function.lst
```

This is much faster than removing each function individually in a batch file, and can be more selective than using UNFUNCTION \*.

**Options:** **/Q**(uiet): Prevents UNFUNCTION from displaying an error message if one or more of the aliases does not exist. This option is most useful in

batch files, for removing a group of functions when some of the functions may not have been defined.

**/R(ead):** Read the list of functions to remove from a file. The file format should be the same format as that used by the **FUNCTION /R** command (see page 256). You can use multiple files with one **UNFUNCTION /R** command by placing the names on the command line, separated by spaces:

```
[c:\] unfunction /r function1.lst function2.lst
```

## **UNLOCK**    *[4DOS]*

(Compatible)

**Purpose:**    Unlock a disk drive to disable exclusive access under Windows 95 / 98 / ME.

**Format:**    UNLOCK [*drive*: ...]

*drive*: The drive or list of drives to unlock.

**See also:** LOCK.

**Usage:**    UNLOCK is only available when 4DOS is running under Windows 95 / 98 / ME.

UNLOCK unlocks the specified drive(s), reversing the effects of LOCK and allowing other sessions to access the drive(s). See the warning under LOCK before using these commands.

If no drives are specified, 4DOS will attempt to unlock all drives.

**UNSET**

(New)

**Purpose:** Remove variables from the environment or registry.

**Format:** UNSET [/D /M /Q /R *file...* /S /U /V] *name...*

or

UNSET \*

***file:*** One or more files containing variable definitions to remove.

***name:*** One or more variables to remove or file types to disable.

**/D**(efault)

**/S**(ystem)

**/M**(aster environment)

**/U**(ser)

**/Q**(uiet)

**/V**(olatile)

**/R**(ead from file)

See also: ESET and SET.

**Usage:** UNSET removes one or more variables from the environment. UNSET supports wildcards in the variable name.

For example, to remove the environment variable CMDLINE:

```
[c:\] unset cmdline
```

If you use the command **UNSET \***, all of the environment variables will be deleted:

```
[c:\] unset *
```

**TC**

You can also remove individual variables from the environment with the Environment dialog (see page 55).

UNSET can be used in a batch file, in conjunction with the SETLOCAL and ENDLOCAL commands, to clear the environment of variables that may cause problems for applications run from that batch file.

For more information on environment variables, see the SET command and the general discussions on pages 23 and 148.

**!**

Use caution when removing environment variables, and especially when using **UNSET \***. Many programs will not work properly without

certain environment variables; for example, 4DOS, 4NT, and Take Command depend on PATH.

**Options:**     **/D(efault environment):** Delete a “default” variable in the registry

**4NT, TC**   **(HKU\DEFAULT\Environment).**

**4DOS**     **/M(aster):** Remove the variable from the master environment rather than the local environment. This option is only useful in secondary shells

**/Q(quiet):** Prevents UNSET from displaying an error message if one or more of the variables or associations does not exist. This option is most useful in batch files, for removing a group of variables when some of the variables may not have been defined.

**/R(ead):** Read environment variables to UNSET from a file. This much faster than using multiple UNSET commands in a batch file, and can be more selective than UNSET \*. The file format should be the same format as that used by the SET /R command (see page 468 for more details). UNSET /R will read from STDIN if there is no filename specified and input is redirected.

**4NT, TC**   **/S(ystem):** Delete a “system” variable in the registry (HKLM\System\CurrentControlSet\Control\Session Manager\Environment).

**4NT, TC**   **/U(ser):** Delete a “user” variable in the registry (HKCU\Environment).

**4NT, TC**   **/V(olatile):** Delete a “volatile” variable in the registry (HKCU\Volatile Environment).

**VER**

(Enhanced)

**Purpose:** Display the current command processor and operating system versions.

**Format:** VER [/R]

**/R**(evision level)

**Usage:** Version numbers consist of a one-digit major version number, a separator, and a one- or two-digit minor version number. VER uses the default decimal separator defined by the current country information. The VER command displays both version numbers:

```
[c:\] ver  
4NT 5.00A (Unicode)   Windows XP 5.10
```

**Option:** **/R**(evision level): Display the command processor and operating system internal revision level (if any), plus your 4DOS, 4NT, or Take Command serial number and registered name.

**4DOS** Under 4DOS, **/R** also displays whether DOS is loaded into the high memory area (HMA), is resident in ROM, or is in normal base memory.

## **VERIFY**

(Compatible)

**Purpose:** Enable or disable disk write verification or display the verification state.

**Format:** VERIFY [ON | OFF]

**Usage:** The operating system maintains an internal verify flag. When the flag is on, the operating system attempts to verify each disk write by making sure that the data written to the disk can be read back successfully. It does **not** compare the data in memory with the data actually placed on disk to fully verify the disk write process.

**4NT, TC** Disk write verification cannot actually be enabled or disabled under Windows. 4NT and Take Command support VERIFY as a “do-nothing” command, for compatibility with *CMD.EXE*. This avoids “unknown command” errors in batch files which use the VERIFY command. The additional discussion below applies only to 4DOS.

If used without any parameters, VERIFY will display the state of the verify flag:

```
[c:\] verify
VERIFY is OFF
```

VERIFY is off when the system boots up. Once it is turned on with the VERIFY ON command, it stays on until you use the VERIFY OFF command or until you reboot.

Verification will slow your disk write operations slightly (the effect is not usually noticeable).

**VOL**

(Enhanced)

**Purpose:** Display disk volume label(s).

**Format:** VOL [*d:*] ...

***d:*** The drive or drives to search for labels.

**Usage:** Each disk may have a volume label, created when the disk is formatted or with the external LABEL command. Also, every floppy disk formatted with DOS version 4.0 or above, or with Windows has a volume serial number.

The VOL command will display the volume label and, if available, the volume serial number of a disk volume. If the disk doesn't have a volume label, VOL will report that it is "unlabeled." If you don't specify a drive, VOL displays information about the current drive:

```
[c:\] vol
Volume in drive C: is MYHARDDISK
```

If available, the volume serial number will appear after the drive label or name.

To display the disk labels for drives A and B:

```
[c:\] vol a: c:
Volume in drive A: is unlabeled
Volume in drive C: is WINSYS
```

**4NT, TC** 4NT and Take Command will also return volume information for UNC names.



## VSCRPUT

(New)

**Purpose:** Display text vertically in the specified color.

**Format:** VSCRPUT *row col* [BRIght][BLInk] *fg* ON [BRIght] *bg text*

***row***: Starting row

***col***: Starting column

***fg***: Foreground text color (**BLI** is only valid in full-screen DOS)

***bg***: Background text color

***text***: The text to display

See also: SCRPUT.

**Usage:** VSCRPUT writes text vertically on the screen rather than horizontally. It can be used for simple graphs and charts generated by batch files.

Like the SCRPUT command, VSCRPUT uses the colors you specify to write the text. See page 30 for details about colors and color names, and notes on the use of bright background colors.

The ***row*** and ***column*** values are zero-based, so on a 25 line by 80 column display, valid ***rows*** are 0 - 24 and valid ***columns*** are 0 - 79. VSCRPUT checks for a valid ***row*** and ***column***, and displays a "Usage" error message if either value is out of range. In Take Command, the maximum ***row*** value is determined by the current height of the Take Command window, and the maximum ***column*** value is determined by the current virtual screen width (see page 39 for more information).

You can also specify the ***row*** and ***column*** as offsets from the current cursor position. Begin the value with a plus sign [+] to move down the specified number of rows or to the right the specified number of columns before displaying text, or with a minus sign [-] to move up or to the left. If you specify 999 for the ***row***, VSCRPUT will center the text vertically. If you specify 999 for the ***column***, VSCRPUT will center the text horizontally.

VSCRPUT does not move the cursor when it displays the text. However, under 4DOS if you have set OutputBIOS to Yes in 4DOS.INI (see below), VSCRPUT will leave the cursor at the end of the displayed ***text***.

The following batch file fragment displays an X and Y axis and labels them:

```
cls bright white on blue
drawhline 20 10 40 1 bright white on blue
drawvline 2 10 19 1 bright white on blue
scrput 21 20 bright red on blue X axis
vscrput 8 9 bright red on blue Y axis
```

**4DOS**    ❖VSCRPUT normally writes text directly to the screen. If you have an unusual display adapter which does not support direct video output, see the OutputBIOS directive on page 237.

**WHICH**

(New)

**Purpose:** Display the command type and what it would execute.

**Format:** WHICH command ...

**command:** One or more commands or files to display information about.

**Usage:** WHICH displays information about internal and external commands, aliases, files, executable extensions, and (in 4NT and Take Command) Windows file associations. The information it reports depends on the type of command or file you specify. For example:

```
[c:\] which cdd buildtree test.btm test.exe *.doc
donothing
CDD is an internal command
buildtree is an alias : cdd /s
test.btm is a batch file : C:\test.btm
test.exe is an external : C:\test.exe
*.doc is associated with : C:\Program
Files\Office2000\Office\WINWORD.EXE
donothing is an unknown command
```

When you use WHICH under Take Command or 4NT, it can also recognize REXX files, EXTPROC files, and associated files.

If a filename includes white space or special characters, it must be enclosed in double quotes.

If the command is an abbreviated alias, WHICH will display the full name. For example:

```
[c:\] alias opt*ions *options
[c:\] which opt
opt*ions is an alias : *options
```

**WINDOW** [4NT, TC]

(New)

**Purpose:** Minimize or maximize the current window, restore the default window size, or change the window title.

**Format:** WINDOW [MIN | MAX | RESTORE | TOPMOST | NOTOPMOST | TOP | BOTTOM | HIDE | FS | WIN | TRAY | | /POS=left,top,width,height | /SIZE=rows,columns | "*title*"]

***title*:** A new title for the window.

**/POS(ition)**

**/SIZE** (of screen buffer)

See also: ACTIVATE and TITLE.

**Usage:** WINDOW is used to control the appearance and title of the current window. MIN reduces the window to an icon, MAX enlarges it to its maximum size, and RESTORE returns it to its default size and location. TOPMOST keeps the window on top of all other windows, and NOTOPMOST allows other windows to overlay it. TOP moves the window to the top of the Z-order, above all other non-TOPMOST windows, and BOTTOM moves it to the bottom. HIDE makes the window invisible. TRAY makes the window invisible and moves the icon from the taskbar to the system tray. FS and WIN switch the 4NT window between full screen and windowed mode.

You can use the **/POS** option to set the location and size of the window on the desktop.

If you specify a new title, the title text must be enclosed in double quotes. The quotes will not appear as part of the actual title.

You can only specify one WINDOW option at a time. The different options cannot be combined in a single WINDOW command. To perform multiple operations, use multiple WINDOW commands.

**Options:** **/POS(ition):** Set the window screen position and size. The values are specified in pixels. **left** and **top** refer to the position of the top left corner of the window relative to the top left corner of the screen.

**4NT** **/SIZE** (of screen buffer): Specify the screen buffer size. The full syntax is **/SIZE=rows, columns**, where **rows** is the number of text rows and **columns** is the number of text columns. Due to the design of Windows console sessions, you cannot use **/SIZE** to reduce the size of the screen buffer; it can only be increased.

## Y

(New)

**Purpose:** Copy standard input to standard output, and then copy the specified file(s) to standard output.

**Format:** Y *file* ...

***file*:** The file or list of files to send to standard output.

See also: TEE.

❖ **Usage:** The Y command copies input from standard input (usually the keyboard) to standard output (usually the screen). Once the input ends, the named files are appended to standard output.

For example, to get text from standard input, append the files *MEMO1* and *MEMO2* to it, and send the output to *MEMOS*:

```
[c:\] y memo1 memo2 > memos
```

The Y command is most useful if you want to add redirected data to the beginning of a file instead of appending it to the end. For example, this command copies the output of DIR, followed by the contents of the file DIREND, to the file DIRALL:

```
[c:\] dir | y dirend > dirall
```

If you are typing at the keyboard to produce input text for Y, you must enter a **Ctrl-Z** to terminate the input.

**4NT, TC** When using Y with a pipe under 4NT or Take Command, you must take into account that the programs on the two ends of the pipe run simultaneously, not sequentially. See page 92 for more information.



## APPENDIX A / ERROR MESSAGES

This appendix lists error messages generated by 4DOS, 4NT, and Take Command, and includes a recommended course of action where appropriate. If you are unable to resolve the problem, look through your *Introduction and Installation Guide* for any additional troubleshooting recommendations, then contact JP Software for technical support (see your *Introduction and Installation Guide*).

Error messages relating to files are generally errors returned by the operating system. You may find some of these messages (for example, "Access denied") too vague to be helpful. 4DOS, 4NT, and Take Command include the file name in file error messages, but are often unable to determine a more accurate explanation of these errors. The message shown is the best information available based on the error codes returned by the operating system.

For some errors you are instructed to "restart the session or reboot the system." This means that you should attempt to correct the error by closing and restarting the current session under Windows. Under DOS, you will probably have to reboot the system to correct the problem. The following list includes all error messages, in alphabetical order:

- 4DOS**     **4DOS internal stack overflow:** You attempted to nest batch files or commands like DO, EXCEPT, FOR, IF, IFF, GLOBAL, or SELECT too deep, and 4DOS ran out of stack space. Restructure your command, alias, or batch file, or use the OPTION command or the StackSize directive in *4DOS.INI* to increase the internal stack size.
- 4DOS**     **4DOS initialization error --:** An error occurred during the 4DOS startup process. Look up the rest of the message in this list for a more specific explanation.
- 4DOS**     **4DOS server error --:** An error occurred in communication between 4DOS's resident and transient portions. A more specific error message follows (the additional error message can be looked up in this list) .
- 4DOS**     **4DOS swapping failed, loading in non-swapping mode:** None of the swapping options worked, so 4DOS loaded in non-swapping mode, which requires about 200K of additional DOS memory. Check your Swapping specification with the OPTION command or in *4DOS.INI*, and/or free some XMS or EMS memory or disk space.
- 4DOS**     **4DOS unrecoverable error XX:** An error occurred in the resident portion of 4DOS. These errors will terminate secondary shells and, and may require you to reboot the system or restart the session if they occur during a primary shell or if 4DOS cannot continue. The error codes are:
- BI    Bad function code. Contact JP Software.
  - DI    Same as Disk swap file corrupted (see page 536).
  - DR    Same as Swap file read error (see page 541)..
  - DS    Same as Swap file seek error (see page 541)..
  - EI    Same as EMS mapping error (see page 536)..

- NS No number for new shell. You have started too many 4DOS secondary shells without properly exiting some of them. Clean up any work in process and reboot the system or restart the session.
- PT Illegal process termination. Contact JP Software.
- TS Terminated inactive shell. Contact JP Software.
- XI Same as XMS move failed (see page 542).

**Access denied:** You tried to write to or erase a read-only file, rename a file or directory to an existing name, create a directory that already exists, remove a read-only directory or a directory with files or subdirectories still in it, or access a file in use by another program.

**4DOS Address table missing:** Your *4DOS.COM* file is invalid. If you used an executable file compression program on *4DOS.COM*, the compression may not be compatible with 4DOS. Re-install *4DOS.COM* from diskette, or download a new copy. **Alias loop:** An alias refers back to itself either directly or indirectly (*i.e.*,  $a = b = a$ ), or aliases are nested more than 16 deep. Correct your alias list.

**Already excluded files:** You used more than one exclude range in a command. Combine the exclusions into a single range.

**4DOS Attempt to exit from root shell:** Another program has probably destroyed a portion of 4DOS's memory. Reboot the system or restart the session; if the error persists, contact JP Software.

**Bad disk unit:** Generally caused by a disk drive hardware failure.

**Batch file missing:** The command processor can't find the batch (*.BAT* or *.CMD*) file it was running. It was either deleted, renamed, moved, or the disk was changed. Correct the problem and rerun the file.

**Can't COPY or MOVE file to itself:** You cannot COPY or MOVE a file to itself. The command processor attempts to perform full path and filename expansion before copying to help ensure that files aren't inadvertently destroyed.

**Can't create:** The command processor can't create the specified file. The disk may be full or write protected, or the file already exists and is read-only, or the root is full.

**Can't delete:** The command processor can't delete the specified file or directory. The disk is probably write protected.

**Can't get directory:** The command processor can't read the directory. The disk drive is probably not ready.

**Can't make directory entry:** The command processor can't create the filename in the directory. This is usually caused by a full root directory. Create a subdirectory and move some of the files to it.

**Can't open:** The command processor can't open the specified file. Either the file doesn't exist or the disk directory or File Allocation Table is damaged.



**Can't remove current directory:** You attempted to remove the current directory, which the operating system does not allow. Change the directory and try again.

**4DOS Can't set up disk swap file:** The disk swap file you specified cannot be opened. The path or drive is invalid, the disk is full, DOS is out of file handles, or there is a hardware problem. Use the **OPTION** command or check *4DOS.INI* to be sure your Swapping directive is correct.

**CD-ROM door open or CD-ROM not ready:** The CD-ROM drive door is open, the power is off, or the drive is disconnected. Correct the problem and try again.

**CD-ROM not High Sierra or ISO-9660:** The CD-ROM is not recognized as a data CD (it may be a music CD). Put the correct CD in the drive and try again.

**Clipboard is in use by another program:** The command processor could not access the Windows clipboard because another program was using it. Wait until the clipboard is available, or complete any pending action in the other program, then try again.

**Clipboard is empty or not text format:** You tried to retrieve some text from the Windows clipboard, but there is no text available.

**Command line too long:** A command line exceeded 511 characters in 4DOS, or 4095 characters in 4NT or Take Command, during alias and variable expansion. Reduce the complexity of the command or use a batch file. Also check for an alias which refers back to itself either directly or indirectly.

**Command only valid in batch file:** You have tried to use a batch file command, like **DO** or **GOSUB**, from the command line or in an alias. A few commands can only be used in batch files (see the individual commands for details).

**4DOS Command tail too long, truncated:** A program attempted to pass a command in an improper format or a command longer than 126 characters to a 4DOS secondary shell. This is probably a bug in the program from which 4DOS was loaded. Contact the author of the program or JP Software for technical assistance.

**Contents lost before copy:** **COPY** was appending files, and found one of the source files is the same as the destination. That source file is skipped, and appending continues with the next file.

**Data error:** The operating system can't read or write properly to the device. On a floppy drive, this error is usually caused by a defective floppy disk, dirty disk drive heads, or a misalignment between the heads on your drive and the drive on which the disk was created. On a hard drive, this error may indicate a drive that is too hot or too cold, or a hardware problem. Retry the operation; if it fails again, correct the hardware or diskette problem.

**DDE [error message]:** A DDE transaction could not be completed by the **DDEEXEC** command. The error message explains the reason. Consult the documentation for the DDE server you are using for additional details if necessary.

**Directory stack empty:** POPD or DIRS can't find any entries in the directory stack.

**Disk is write protected:** The disk cannot be written to. Check the disk and remove the write-protect tab or close the write-protect window if necessary.

**4DOS Disk swap file corrupted:** The 4DOS disk swapping file (*4DOSSWAP.nnn* or *xxxxxxx.4SW*) has been moved, deleted, or damaged by another program. Reboot the system or restart the session.

**Drive not ready -- close door:** The removable disk drive door is open. Close the door and try again.

**Duplicate redirection:** You tried to redirect standard input, standard output, or standard error more than once in the same command. Correct the command and try again.

**4DOS EMS deallocation failed:** 4DOS can't deallocate EMS memory when exiting from a secondary shell. The EMS map has been corrupted or the memory area used by 4DOS or the EMS driver has been destroyed by a program. Clean up any work in process and reboot the system or restart the session.

**4DOS EMS map save or restore failed:** 4DOS cannot save or restore the EMS page map. The EMS map has been corrupted, memory has been destroyed by a program, or you have an incompatible EMS driver. If this error recurs, try another swapping method, update your EMS driver, or contact JP Software.

**4DOS EMS mapping failed:** 4DOS can't map EMS pages when swapping to or from EMS. The EMS map has been corrupted or the memory area used by the loader or the EMS driver has been destroyed. Reboot the system or restart the session.

**Environment already saved:** You have already saved the environment with a previous SETLOCAL command. You cannot nest SETLOCAL / ENDLOCAL pairs.

**Error in command-line directive:** You used the **//inline** option to place an *.INI* directive on the startup command line, or, under 4DOS, on the SHELL= line in *CONFIG.SYS*, but the directive is in error. Usually a more specific error message follows, and can be looked up in this list.

**Error on line [nnnn] of [filename]:** There is an error in your *.INI* file. The following message explains the error in more detail. Correct the line in error and restart the command processor for your change to take effect.

**Error reading:** The operating system experienced an I/O error when reading from a device. This is usually caused by a bad disk, a device not ready, or a hardware error.

**Error writing:** The operating system experienced an I/O error when writing to a device. This is usually caused by a full disk, a bad disk, a device not ready, or a hardware error.

**Exceeded batch nesting limit:** You have attempted to nest batch files more than 16 levels deep in 4DOS, or 32 levels deep in 4NT and Take Command.

**Exceeded SETLOCAL nesting limit:** You have attempted to nest SETLOCAL more than 8 levels deep in 4DOS or 16 levels deep in 4NT and Take Command.

**4DOS Fatal error -- please reboot:** 4DOS cannot continue due to the previous error. Reboot the system or restart the session.

**4DOS Fatal error, some directives may not have been processed:** An I/O error occurred while reading your *4DOS.INI* file. There may be a physical problem with data on the disk or a sharing error on a multitasking system.

**File Allocation Table bad:** The operating system can't access the FAT on the specified disk. This can be caused by a bad disk, a hardware error, or an unusual software interaction.

**4NT, TC File association not found:** The ASSOC command could not find a file association for the specified extension in the Windows registry.

**File exists:** The requested output file already exists, and the command processor won't overwrite it.

**File is empty:** You attempted to LIST a file with no data (a zero-byte file).

**File not found:** The command processor couldn't find the specified file. Check the spelling and path name.

**4NT, TC File type not found:** The FTYPE command could not find the specified file type in the Windows registry.

**General failure:** This is usually a hardware problem, particularly a disk drive failure or a device not properly connected to a serial or parallel port. Try to correct the problem, or reboot and try again. Also see **Data error** above.

**Include file not found:** You used the Include directive in the *.INI* file, but the file you specified was not found or could not be opened.

**Include files nested too deep:** You used the Include directive in the *.INI* file, and attempted to nest include files more than three levels deep.

**Infinite COPY or MOVE loop:** You tried to COPY or MOVE a directory to one of its own subdirectories and used the /S switch, so the command would run forever. Correct the command and try again.

**Insufficient disk space:** COPY or MOVE ran out of room on the destination drive. Remove some files and retry the operation.

**4DOS Insufficient load space:** There is not enough room in 4DOS's internal memory areas to include all of the options you requested in *4DOS.INI*. Contact JP Software.

**4DOS Internal DOS error:** DOS encountered an internal bug and failed. Reboot the system.

- 4DOS**     **Invalid AUTOEXEC filename:** You specified an invalid path or filename for the AUTOEXEC file with the /P: startup switch. The default name will be used instead.
- Invalid character value:** You gave an invalid value for a character directive in the .INI file.
- Invalid choice value:** You gave an invalid value for a “choice” directive (one that accepts a choice from a list, like “Yes” or “No”) in the .INI file.
- Invalid color:** You gave an invalid value for a color directive in the .INI file.
- Invalid count:** The character repeat count for KEYSTACK is incorrect.
- Invalid date:** An invalid date was entered. Check the syntax and reenter.
- Invalid directive name:** The command processor can't recognize the name of a directive in your .INI file.
- 4DOS**     **Invalid DOS version:** You need a newer version of DOS to execute the specified command.
- Invalid drive:** A bad or non-existent disk drive was specified.
- 4DOS**     **Invalid INI file path or name, file not processed:** The path or name for the initialization file on the SHELL= line in *CONFIG.SYS*, or on the startup command line, is invalid. Correct the @d:\path\inifile option to name the correct file.
- Invalid key name:** You tried to make an invalid key substitution in the .INI file, or you used an invalid key name in a keystroke alias or command.
- Invalid numeric value:** You gave an invalid value for a numeric directive in the .INI file.
- Invalid parameter:** The command processor didn't recognize a parameter. Check the syntax and spelling of the command you entered.
- Invalid path:** The specified path does not exist. Check the disk specification and / or spelling.
- Invalid path or file name:** You used an invalid path or filename in a directive in the .INI file.
- 4DOS**     **Invalid startup switch, ignored:** You passed 4DOS an invalid option on the SHELL= line in *CONFIG.SYS* or on the startup command line for a secondary shell.
- 4DOS**     **Invalid Swapping option or path:** The swap type or disk swap path in the *4DOS.INI* Swapping directive is invalid. 4DOS ignores the bad swap type or path and attempts to scan the rest of the Swapping specification for a valid option. Multiple errors in the Swapping directive will cause this message to repeat.

**Invalid time:** An invalid time was entered. Check the syntax and reenter.

**Keystroke substitution table full:** The command processor ran out of room to store keystroke substitutions entered in the *.INI* file. Reduce the number of key substitutions or contact JP Software or your dealer for assistance.

**4DOS KSTACK.COM not loaded:** You attempted to execute a KEYSTACK command without loading *KSTACK.COM*. See KEYSTACK for more information.

**Label not found:** A GOTO or GOSUB referred to a non-existent label.

**4DOS Memory [allocation | deallocation] error:** 4DOS can't allocate or deallocate memory while loading, or while reserving or releasing memory for internal use. DOS memory allocation has been corrupted, or another application has reserved memory incorrectly. Reboot the system or restart the session.

**4DOS Memory destroyed:** The DOS memory control blocks have been corrupted. Reboot the system or restart the session.

**Missing ENDTEXT:** A TEXT command is missing a matching ENDTEXT

**Missing GOSUB:** Your batch file attempted to RETURN from a subroutine, but no subroutine had been called with GOSUB.

**Missing SETLOCAL:** An ENDLOCAL was used without a matching SETLOCAL.

**No aliases defined:** You tried to display aliases but no aliases have been defined.

**No closing quote:** The command processor couldn't find a second matching back-quote ['] or double-quote ["] on the command line.

**No expression:** The expression passed to the @EVAL variable function is empty.

**4DOS No file handle available:** This is an internal 4DOS disk swapping error. Change to another swapping method if possible, or contact JP Software.

**4DOS No room for INI file name:** 4DOS does not have enough space to pass the name of your *.INI* file to secondary shells; see **String area overflow** for more details. Any [Secondary] section in *4DOS.INI* will be ignored in secondary shells.

**4DOS No UMBs; loading low:** The LOADHIGH (or LH) command can't find any UMBs for your program. The program is loaded into base memory. LH and LOADHIGH only work with MS-DOS 5.0 and above, when the DOS=UMB directive is included in *CONFIG.SYS* and sufficient upper memory space is available for the program.

**4DOS No upper memory available, low memory will be used for ...:** You asked 4DOS to load the block of memory named in the message into a UMB via the corresponding directive in *4DOS.INI* (UMBLoad, UMBEnvironment, etc.), but no UMB was available. Check that your XMS driver is properly installed and/or free up some UMB space in use by another program.

- 4DOS**     **Non-DOS disk:** DOS can't read the disk. Either the disk is bad, or it has been formatted by a different operating system. Reformat it as a DOS disk. Also see **Data error** above; the problems described there can sometimes cause this error.
- 4NT, TC**   **No shared memory found:** The SHRALIAS command could not find any global alias list, function list, history list, or directory history list to retain, because you executed the command from a session with local lists. Start 4NT or Take Command with at least one global list, then invoke SHRALIAS.
- 4NT, TC**   **No SMTP server address:** The SENDMAIL command could not find a default mail server in the registry. Specify a server name with the MailServer .INI directive.
- 4NT, TC**   **No SMTP user address:** The SENDMAIL command could not find a default mail user name in the registry. Specify a user name with the MailAddress .INI directive.
- Not a directory:** The command requires a directory name, but you gave a file name, or a name which does not exist at all.
- Not an alias:** The specified alias is not in the alias list.
- Not in environment:** The specified variable is not in the environment.
- 4DOS**     **Not in swapping mode:** You attempted to turn swapping on or off with the SWAPPING command, but 4DOS is loaded in non-swapping mode.
- Not ready:** The specified device can't be accessed.
- Not same device:** This error usually appears in RENAME. You cannot rename a file to a different disk drive.
- 4DOS**     **Out of environment/alias space:** 4DOS has run out of space for environment variables or aliases. Edit the SHELL line in *CONFIG.SYS* or the Environment directive in *4DOS.INI* to increase the environment size, or the Alias directive in *4DOS.INI* to increase the alias list size.
- Out of memory:** The command processor or the operating system had insufficient memory to execute the last command, or under DOS, the memory control blocks have been destroyed. If this error occurs in a 4DOS secondary shell, return to the primary shell before running the command. Otherwise, try to free some memory by removing memory-resident programs (under DOS), or closing other sessions (Windows). If the error persists, contact JP Software for assistance.
- 4DOS**     Under 4DOS, if the base memory (DOS RAM) figures reported by MEMORY are unreasonable, the memory control blocks have probably been destroyed and you must reboot the system or restart the session.
- Out of paper:** The operating system detected an out-of-paper condition on one of the printers. Check your printer and add paper if necessary.

**Overflow:** An arithmetic overflow occurred in the @EVAL variable function. Check the values being passed to @EVAL. @EVAL can handle 20 digits to the left of the decimal point and 10 to the right.

**Read error:** The operating system encountered a disk read error; usually caused by a bad or unformatted disk. Also see **Data error** above.

**4DOS**    **Region unavailable, using first available region for ...:** You used a *4DOS.INI* directive to load the block of memory named in the message into a specific UMB region, but that region was unavailable. Check the use of upper memory for device drivers and other programs loaded before 4DOS, or change the requested region.

**Sector not found:** Disk error, usually caused by a bad or unformatted disk. Also see **Data error** above.

**Seek error:** The operating system can't seek to the proper location on the disk. This is generally caused by a bad disk or drive. Also see **Data error** above.

**Sharing error or Sharing violation:** You tried to access a file in use by another program, or modify or delete an executable file while it was running. Wait for the file to become available, or change your method of operation so that another program does not have the file open while you are trying to use it.

**4DOS**    **Specified INI file not found:** The file specified with the @inifile option on the 4DOS command line does not exist.

**4NT, TC**    **SHRALIAS already loaded:** You used the SHRALIAS command to load *SHRALIAS.EXE*, but it was already loaded. This message is informational and generally does not indicate an error condition.

**4NT, TC**    **SHRALIAS not loaded:** You used the SHRALIAS /U command to unload *SHRALIAS.EXE*, but it was never loaded. This message is informational and may not indicate an error condition.

**Startup failed, contact JP Software:** The command processor could not initialize and start operation. Contact JP Software or your dealer for assistance.

**4DOS**    **String area overflow:** The command processor ran out of room to store the text from string directives in the .INI file. Reduce the complexity of the .INI file or contact JP Software for assistance.

**4DOS**    **Swap file [seek | read | write] failed:** 4DOS encountered an I/O error while accessing the disk swap file (*4DOSSWAP.nnn* or *xxxxxxx.4SW*). The disk was changed, the file has been destroyed by a program, or 4DOS's memory area has been overwritten by another program. Reboot the system or restart the session.

**Syntax error:** A command or variable function was entered in an improper format. Check the syntax and correct the error.

- 4DOS Syntax error in region number or size:** You specified an invalid region number or size in the LH or LOADHIGH command.
- 4DOS Too many SETs in CONFIG.SYS:** The SET commands in your Novell DOS / OpenDOS *CONFIG.SYS* file exceeded the size of 4DOS's buffer area. Reduce the length of the commands or contact JP Software for assistance.
- 4DOS Too many open files:** DOS or Windows has run out of file handles. Try increasing the FILES setting in *CONFIG.SYS*.
- 4DOS Transient memory [allocation | deallocation] error:** 4DOS could not reserve or release memory for its transient portion (probably in a SWAPPING OFF or SWAPPING ON command). The memory control blocks have been destroyed, or a program has fragmented memory. Reboot the system or restart the session.
- Unbalanced parentheses:** The number of left and right parentheses did not match in an expression passed to the @EVAL variable function. Correct the expression and retry the operation.
- UNKNOWN\_CMD loop:** The UNKNOWN\_CMD alias (see page 260) called itself more than ten times. The alias probably contains an unknown command itself, and is stuck in an infinite loop. Correct the alias.
- Unknown command:** A command was entered that 4DOS, 4NT, or Take Command didn't recognize and couldn't find in the current search path. Check the spelling or PATH specification. You can handle unknown commands with the UNKNOWN\_CMD alias (see page 260)..
- Variable loop:** A nested environment variable refers to itself, or variables are nested more than 16 deep. Correct the error and retry the command.
- 4NT, TC Window title not found:** The ACTIVATE command could not find a window with the specified title. Correct the command or open the appropriate window and try again.
- Write error:** The operating system encountered a disk write error; usually caused by a bad or unformatted disk. Also see **Data error** above.
- 4DOS XMS deallocation failed:** 4DOS could not deallocate XMS memory when exiting a secondary shell. XMS memory has been destroyed; reboot the system or restart the session.
- 4DOS XMS move failed:** 4DOS could not move data between base memory and XMS memory while swapping itself. XMS memory has been destroyed; reboot the system or restart the session.



# INDEX

- ! warning mark, 3, 246
- !!!❖advanced topic mark, 57, 246
- \$ (dollar sign), as parameter
  - character, 129, 224, 478
- % sign
  - and alias parameters, 257
  - and batch file parameters, 128
  - and environment variables, 150, 151
  - in file names, 15
  - use with variable functions, 166
- %# and %n&
  - in aliases, 258
  - in batch files, 129, 483
- %0 to %255
  - in aliases, 257
  - in batch files, **128**, 483
- & (ampersand)
  - as command separator, 219, 475
  - as parameter character, 129, 224, 478
- && (and) in conditional commands, 117
- \* (asterisk)
  - as file name “wildcard”, 95
  - to temporarily disable an alias, 137
- /, in command switches, 245
- ;(semicolon)
  - in include lists, 105
  - in long file names, 15, 98
- ? as file name “wildcard”, 95
- ? command, 248
- ? variable, 156
- ?? variable, 157
- @ sign
  - for key codes, 34, 256
  - in file names, 15
  - in INKEY results, 388
  - to define a keystroke alias, 255
  - to override batch file echo, **128**, 332, 452
  - to override command history save, 62
  - to read a file in DO, 326
  - to read a file in FOR, 352
- ^ (caret), 219, 221, 475, 476
- \_? variable, 157
- || (or) in conditional commands, 117
- + variable, 157
- = variable, 157
- 4DOS
  - and DESQview, 237
  - and disk caching software, 237
  - and DR-DOS, 400
  - and MS-DOS 5.0 and above, 400
  - memory usage, 217
  - resident portion, loading in upper memory, 217
  - REXX support in, 146
  - stack size, 238
- 4DOS and 4NT, compatibility with Take Command, 40, 197, 226
- 4NT
  - REXX support in, 147
- 4START and 4EXIT, 133
  - and MAKE utilities, 135
  - location of, 209
- 4VER variable~\_~, 157
- ABS function~@~, 169
- ACSTATUS variable~\_~, 158
- ACTIVATE, 130, 249
  - and KEYSTACK, 94, 395
- Advanced Power Management status, 158
- ALIAS command, 19, 251
- ALIAS function~@~, 170
- ALIAS variable~\_~, 158
- Aliases, 123, 251
  - and executable extensions, 107
  - and QUIT, 446

- 
- back-quotes in, 202, **253**, 257, 261
  - dialog, 46
  - disabling, 77, 137, **253**, 479
  - editing, 55, 336
  - examples, 123, 136
  - expanding, on command line, 199, 233, **259**
  - expansion, 78, **199**
    - and batch file debugging, 140
    - command line, 77
    - disabling, 479
    - nested, 200
  - global, 213, **260**
    - in upper memory, 215
  - in batch files, 136
  - in upper memory, 215
  - inheritance, in secondary shells, 237
  - internal variables in, 259
  - keystroke, 125, **255**
    - and key assignments, 231
  - local, 213, **260**
  - local and global, 493
  - multiple commands in, 252
  - nested, 124, 200, **253**
  - order of execution, 199
  - parameters, 200, **257**, 259
  - reading from a file, 261
  - removing, 519
  - retrieving, 170, 180
  - saving and restoring, 257, 482
  - setting default command options with, 254, 258
  - sharing with 4DOS, 4NT, and Take Command, 197
  - size of list, 158, **210**, **211**, 416
  - suspending execution of, 434
  - testing for existence, 379
  - variable functions in, 196, **259**
- Alt key, 42
- ALTNAME function~@~, 170
- And (&&) in conditional commands, 117
- AND., in IF and IFF~., 374, **380**, 382
- ANSI driver, 29
  - and CLS, 279
  - and COLOR, 280
  - and PROMPT, 442
  - and screen colors, 230
  - and scrolling, 478
  - and TEXT, 506
  - detecting, 158
  - escape sequences, 120, 442, 506
  - use by 4DOS, 218
- ANSI support, 442
  - enabling, 218
- ANSI variable~\_~, 158
- APMAC variable~\_~, 158
- APMBATT variable~\_~, 158
- APMLIFE variable~\_~, 158
- Appending files, 281
- Applications
  - listing running programs, 502
  - searching for, **20**, 79, 109, 431
  - shortcuts for, 485
  - starting, 44, **78**, 303, 384, 399, 436, 490
    - for sound files, 436
    - for video files, 435
    - to print files, 438
  - terminating, 501
  - testing for, 379
  - waiting for, **115**, 124, 222
  - windows for, 80
- AppPaths registry entry, 21
- Argument quoting, 201
- Arithmetic, 169, 171, **173**, 181, 471
- ASCII, **25**, 170, 171
- ASCII function~@~, 170
- ASSOC, 262
  - reading from a file, 263
- Attachments, 469
- ATTRIB, 264
- ATTRIB function~@~, 170
-

---

AUTOEXEC.BAT, 133  
    location of, 210  
    parameters for, 210  
    speeding up, 495  
    starting KSTACK.COM, 394

Automatic batch files, 133

Automatic directory changes, 73

Backspace character, 120

BAT files~., 20, **126**, 410

Batch files, 126  
    "case" statements in, 496  
    aliases in, 136, 137  
    and executable extensions, 107  
    and redirection, 90  
    automatic, 133  
    based on command history, 373  
    branching in, 367  
    calling, 270  
    capturing output of, 141  
    chaining, 270  
    commands for, 130, 242  
    comments in, 452  
    compression, 144  
    Ctrl-Break handling, 427  
    debugging, **138**, 481  
    detecting 4DOS, 4NT, or Take  
        Command from, 135  
    displaying messages in, **332**, 334,  
        458, 460, 506, 529  
    echoing of commands, **128**, 218,  
        332, 479  
    environment variables in, 129  
    error handling, 427  
    internal variables in, 129  
    interrupting, 132  
    keyboard input in, 387, 390, 425,  
        445  
    labels, 365, 367  
        testing for existence, 379  
    line continuation in, 144  
    line numbers in, 140, 159  
    loops in, 324  
    modes, **126**, 410  
    names of, 129, 159  
    nesting, 158, 270  
    parameters, **128**, 200, 483  
    sample, 132  
    sharing among 4DOS, 4NT, and  
        Take Command, 197  
    starting applications from, 222  
    string processing in, 141, 479  
    subroutines, **365**, 457  
        passing parameters to, 365  
    suspending execution of, 434  
    temporary variables in, 130  
    terminating, 271, 446  
    timing events in, 508  
    variable functions in, **129**, 196

BATCH variable~\_, 158

BATCHLINE variable~\_, 159

BATCHNAME variable~\_, 159

BATCOMP and BATCOM32, 144

BATTERY variable~\_, 159

BATTERYLIFE variable~\_, 159

BEEP, 130, 218, **268**

BG variable~\_, 159

BIOS output  
    and DRAWBOX, 329  
    and LIST, 408  
    and SCRPUT, 461  
    and SELECT, 466  
    and VSCRPUT, 530  
    in 4DOS, 238

BOOT variable~\_, 159

Boxes, drawing, on screen, 328

BREAK, 269

BTM files~., 20, **126**, 410

BUILD variable~\_, 159

CALL, 130, **270**

CANCEL, 130, 270, **271**, 446

CAPS function~@~, 171

Carriage return character, 120

Case (upper / lower)  
    conversion, 183, 189

PROMPT, 439

---

---

Case" statements, in batch files~"~, 496  
CASE, in SWITCH, 496  
Caveman, 45, 80, 494  
    waiting for applications, 115  
CD, 272  
CDD, 275  
CDFS file system, 180  
CDPATH, 73, **86**, 273, 275, 443  
    and extended directory searches, 86  
    environment variable, 152  
CDROM function~@~, 171  
CEILING function~@~, 171  
Changing directories, 73, **81**, 272, 275, 437, 443  
    automatically, 73  
    CDPATH, 86  
    extended directory searches, 83  
    local and global history, 75  
    preserving case, 237, 273, 274, 276  
    returning to previous directory, 74, **82**, 273, 276, 443  
CHAR function~@~, 171  
Character sets, 25  
CHCP, 278  
CHDIR, 272  
CHILDPID variable~\_~, 159  
CI variable~\_~, 159  
CLIP function~@~, 171  
Clipboard, 41, 44, 59  
    CLIP  
        device, 89  
        copying, 283  
        redirecting to, 44, 89  
        retrieving text from, 171, 326, 353  
        viewing, 369, 406, 499, 517  
        writing text to, 171  
CLIPW function~@~, 171  
CLS, 41, 130, **279**  
CMD files~.~, 20, **126**, 410

CMDLINE environment variable, 152  
CMDLINE variable~\_~, 159  
CMDPROC variable~\_~, 159  
CO variable~\_~, 159  
CODEPAGE variable~\_~, 159  
Cold reboot, 449  
COLOR, 130, **280**  
COLORDIR environment variable, 152  
Colors  
    and ANSI driver, **29**, 230  
    blinking text, 32  
    border, 31  
    boxes, 328  
    bright background, **32**, 228, 475  
    errors in, 31  
    in directory displays, 229, **311**  
    in directory search window, 229  
    in LIST, 229, 408  
    in popup window, 229  
    in PROMPT, 442  
    in SELECT, 230  
    lines, 330, 331  
    names of, 30  
    numbers for, 30  
    setting defaults, 228, 279, 280  
    standard, 230  
    testing, 159, 162  
    text, 460, 506, 529  
    typed input, 60, 229  
COLUMN variable~\_~, 160  
COLUMNS variable~\_~, 160  
COMMA function~@~, 171  
Command grouping, 90, **118**  
Command line, 57  
    and clipboard, 59  
    automatic directory changes, 73  
    cut and paste, 59  
    deleting text, 44  
    echoing of, 332, 333  
    editing, **58**, 231  
        keys, 121, 226, 233  
        **mode**, 58, 221, 478

- 
- expanded, viewing, 333
  - filename completion, 67
  - help, 56, 77
  - history, 61
  - history window, 64
  - input method, 223, 477
  - length of, 58, **78**, 119
  - multiple commands, 76
  - pasting text, 41, 44
  - processing, 19, 199
  - prompt, 439
  - Command processor, 7
    - determining name of, 159
  - Command Reference Guide, 241
  - Command separator, **76**, 157, 219, 475
    - compatibility between products, 197
    - disabling, 479
    - in aliases, 252
    - in IFF, 383
    - saving and restoring, 482
  - Command tail, 19, 199
  - Command window, 38, 43, 44
  - COMMAND.COM message server, 238
  - Commands
    - conditional execution of, 382
    - determining type of, 531
    - executing, in prompt, 175
    - external, **18**, 241, 431
      - and /? help, 78
      - exit codes for, 156, 162
      - waiting for, **115**, 124, 222
    - format of, 243
    - help for, 56, 77, 371
    - internal, **18**, 241
      - disabling, 248, **477**
      - testing for existence, 379
    - length of, 58, 76, **78**
    - list of, 248
    - order of execution, 199
    - parameters, 66
    - processing, 199
    - prompting before execution, 248
    - unknown, handling, 261
    - viewing output, 406
  - Comments
    - .INI file, 207
    - alias files, 261
    - batch files, 452
    - environment variable files, 472
  - Compressed drives, 315, 466
  - Computer, name of, 164
  - COMSPEC environment variable, 152, 205, 525
  - Conditional commands, 117
    - disabling, 479
  - Conditions, evaluating, 181, 374, 382
  - CONFIG.SYS, 9
  - Configuration, 152, **203**, 429
    - using dialogs, 45, **50**
  - CONsole device, 290
  - Console session, 45, **80**
  - Conventions, 30
  - CONVERT function~@~, 171
  - COPY, 281
  - COPYCMD environment variable, 152
  - Copying from scrollback buffer, 41, 44
  - CopyPrompt, 219
  - Country code, 159, 160, 278, 291
  - COUNTRY variable~\_~, 160
  - CPU type, 160
  - CPU variable~\_~, 160
  - CPUUSAGE variable~\_~, 160
  - CRC32 function~@~, 171
  - Critical errors, 116, 157
    - automatic Fail response, 237
    - detecting, 427
  - Ctrl-Break
    - at page and file prompts, 80
    - checking, 269, 427
    - during DELAY, 299
    - during INKEY, 388
    - during INPUT, 390
    - during PAUSE, 434
    - during QUERYBOX, 445
    - during SELECT, 463
-

---

in batch files, 132, 427  
in external programs, 157  
Ctrl-Z end of file mark, and COPY,  
287  
CTTY, 90, **290**  
Current drive and directory, 14  
changing, on FTP servers, 273  
displaying, 273, 276  
retrieving, 160  
saving and restoring, 443, 482  
Cursor  
determining position, 160, 163  
mouse, shape of, 212  
positioning, 458  
text, shape of, 220, 478  
retrieving, 159  
CWD variable~\_~, 160  
CWDS variable~\_~, 160  
CWP variable~\_~, 160  
CWPS variable~\_~, 160  
Daily execution of batch file, 196  
DATE command, 291  
DATE function~@~, 171  
DATE variable~\_~, 160  
Dates, 122  
converting, 171, 172, 184, 196  
formatting, 184  
in variable function parameters,  
167  
international format, 167  
ranges, 98  
retrieving, 160, 161, 163, 165  
Day  
of month, 160, 171  
of week, 161, 163, 172, 181  
of year, 161, 172  
DAY function~@~, 171  
DAY variable~\_~, 160  
DDE, 149  
Take Command as client, 292  
Take Command as server, 149  
DDEEXEC, 292  
DEC function~@~, 171

Decimal character, 174, **220**, 477  
saving and restoring, 482  
DECIMAL function~@~, 172  
DEFAULT, in SWITCH, 496  
DEL, 293  
and recycle bin, 225, **295**, 296,  
297  
controlling prompts, 220  
DELAY, 299  
Deleting files, 293  
DESCRIBE, 231, **300**  
DESCRIPT function~@~, 172  
DESQview  
and 4DOS, 237  
detecting, **162**  
DETACH, 303  
DETACHPID variable~\_~, 160  
Detecting 4DOS, 4NT, and Take  
Command, from a batch file, 135  
DEVICE function~@~, 172  
Devices, detecting, 172  
Dialogs  
user input, 425, 445  
Dialogs, in Take Command, 47  
Aliases, 55  
Configuration, 45, **50**  
Descriptions, 54  
directory name, 180  
Environment, 55  
file name, 180  
Find files/text, 52  
folder selection, 181  
Run, 48  
DIGITS function~@~, 172  
DIR, 304  
and include lists, 105  
color-coded displays, 229, **311**  
and CTTY, 290  
directory size limits, 314  
file descriptions, and redirection,  
318  
multi-column displays, 310  
options, 314

---

- output format, 307
  - redirected output, 313, 316
  - selecting files, 305
- DIRCMD environment variable, 153
- Directory history, 74, 321
  - in upper memory, 216
  - local and global, 75, 213, 493
  - size of, 210
  - window, 74
- Directory navigation, 81
- Directory stack, 437, 443
  - clearing, 437
  - displaying, 323
- Directory tree, 13
- Directory tree, displaying, 514
- DIRHISTORY, 321
- DIRS, **323**, 437, 443
- Disk drives, 10
  - compressed, 315, 466
  - remote, 186
  - removable, 186
  - space on, 172, 359
  - swapping to, 214, 495
  - testing status of, 185
  - volume label, 182, 528
  - write verification on, 527
- DISK variable~\_~, 160
- DISKFREE function~@~, 172
- DISKTOTAL function~@~, 172
- DISKUSED function~@~, 172
- DNAME variable~\_~, 160
- DO, 130, **324**
  - and GOTO, 327
  - and RETURN, 327, 457
- DOS
  - commands, help for, 77
  - default command processor, 8
- DOS variable~\_~, 161
- DOS/V (Japan), 238
- DOSMEM function~@~, 172
- DOSVER variable~\_~, 161
- DOW function~@~, 172
- DOW variable~\_~, 161
- DOWF function~@~, 172
- DOWF variable~\_~, 161
- DOWI function~@~, 172
- DOWI variable~\_~, 161
- DOY function~@~, 172
- DOY variable~\_~, 161
- DPMI variable~\_~, 161
- DRAWBOX, 130, **328**
- DRAWHLINE, 130, **330**
- DRAWVLINE, 130, **331**
- DV variable~\_~, 162
- ECHO, 332
  - ANSI sequences in, 120
  - batch files, **128**, 131, 140, 452
    - default state, 218, 479
    - inheritance, 270
  - command line, 333
- ECHO variable~\_~, 162
- ECHOERR, 332
- ECHOS, 131, **334**
- ECHOSERR, 334
- Editing aliases and environment variables, 336
- Editor, 47, 221
- ELSE and ELSEIFF, in IFF, 382
- Email, sending, 469
- EMS function~@~, 172
- ENDDO, in DO, 324
- ENDIFF, in IFF, 382
- ENDLOCAL, 130, 137, **335**, 482, 524
- ENDSWITCH, in SWITCH, 496
- ENDTEXT, 132, 506
- ENV variable~\_~, 162
- Environment, **23**, 150, 470
  - commands for, 243
  - dialog, 46, 55
  - in upper memory, 216
  - master, 151, 184, 337, 525
  - saving and restoring, 482
  - size of, 150, **211**, 471
    - displaying, 416
    - testing, 162

---

---

Environment variables, **23**, 150, 470, 471  
    and FOR, 355  
    and INKEY, 387  
    and INPUT, 390  
    and QUERYBOX, 445  
    characters in name, 150  
    editing, 55, 336  
    expansion, 78, **200**  
        and batch file debugging, 140  
        disabling, 479  
    for current directory, in Windows NT, 154  
    name completion on command line, 72  
    nesting, 150  
    reading from a file, 472  
    referencing, 150  
    removing, 524  
    spaces in, 471  
    testing for existence, 378  
ERASE, 293  
Error messages, 535  
ERRORLEVEL variable, 162  
Errors  
    handling, in batch files, 427  
    input and output, 116  
    system  
        message text, 173  
        numeric codes, 162, 164  
ERRTEXT function~@~, 173  
Escape character, 120, 157, 221, 476  
    compatibility between products, 197  
    saving and restoring, 482  
ESET, 151, 231, **336**, 471  
    and PATH, 431  
EVAL function~@~, **173**, 221, 476  
EVENTLOG, 131, 338  
EXCEPT, 118, **339**  
    and DEL /Z, 298  
EXEC function~@~, 175

EXECSTR function~@~, 175  
Executable extensions, 80, **107**, 200  
    and filename completion, 68  
    and Personal REXX, 146  
    and Windows file associations, 113  
    wildcards in, 109  
EXETYPE function~@~, 175  
EXIT, 342  
Exit codes, 200  
    and CALL, 270  
    and CANCEL, 271  
    and conditional commands, 117  
    and DEL, 295  
    and EXIT, 342  
    and GLOBAL, 363  
    and IF tests, 378  
    and PROMPT, 439  
    and QUIT, 446, 457  
    retrieving, 156, 162  
Exiting, from Take Command, 43, 342  
EXPAND function~@~, 175  
EXT function~@~, 176  
Extended  
    key codes, 27  
Extended directory searches, **83**, 273, 275, 443  
    and CDPATH, 86  
    and wildcards, 85  
    database, **84**, 277  
        automatic updates, 289, 297, 414, 419, 423, 447, 454  
    options, **85**, 222  
    popup window, 219, 229  
EXTENDED function~@~, 176  
Extended parent directory names, 94  
EXTPROC support, 148  
FFIND, 52, **343**  
FG variable~\_~, 162  
FIELD function~@~, 176  
File attributes, **16**, 264  
    and COPY, 285, 287, 289, 424  
    and DEL, 297



- 
- and DIR, 304, 315, 318
  - and EXCEPT, 340
  - and FOR, 357
  - and GLOBAL, 363
  - and MOVE, 422
  - and REN, 455
  - and SELECT, 466
  - of subdirectories, 267
  - selecting files by, 111
  - setting, 264
  - switches for**, 112
  - testing, 170, 189
  - viewing, 264
  - Windows 2000/XP, 17, 189, 265, 318
- File descriptions**, 300
- controlling processing of, 221, 476
  - dialog, 46
  - displaying, and redirection, 318
  - editing, 54, 300
  - file name for, 160, **221**, **301**, 476
  - length of, 220, **301**, 308, 464
  - retrieving, 172
  - selecting files by, 111, 307, 466
  - sorting by, 318, 347, 468
- File exclusion ranges**, 103
- File lists~@~**, 109
- File names**, 10, **15**, 94
- case of, **15**, 155, 228, 479
  - FAT-compatible, 16
  - in variable function parameters, 167
  - length of, 15
  - long, **15**, 94
    - alternate (short) names, 170
    - and wildcards, 106
    - converting, 69, 182, 187
    - disabling, 239
    - displaying, 309, 310, **320**, 464, 468
    - extension, **16**, 176
    - quoting, **16**, 167
    - searching for, 106
    - semicolons in, 15, 98
    - parts of, **10**, 176, 177, 180, 184, 185
    - selecting, 180
    - short
      - converting, 69, 182, 187
      - displaying, 309, 310, **319**, 464, 468
    - shorthand for, 94
    - UNC, 13
    - unique, 188
- File processing prompts**, 80
- File streams**, NTFS, 18
- and COPY, 284
  - and DEL, 295
  - and HEAD, 370
  - and LIST, 407
  - and MOVE, 420
  - and TAIL, 500
  - and TYPE, 518
- File systems**, 11
- date and time stamps, 17, 103, 176, 178, 319, 468
  - detecting, 180
  - network, 12
  - path length, 14
- FILEAGE function~@~**, 176
- FILECLOSE function~@~**, 176
- FILECOMPLETION environment variable**, 153
- FILEDATE function~@~**, 176
- Filename completion**, 67
- customizing, **70**, 219, 222
  - window, 72
- FILENAME function~@~**, 177
- FILEOPEN function~@~**, 177
- FILEREAD function~@~**, 177
- Files**, 10
- adding line numbers to, 517
  - allocated size of, 178, 316
  - commands for, 242
  - copying, 281
  - counting, 177

---

counting lines in, 183  
date stamp, 17, 176, 183, 291, 304  
    copying, 512  
    modifying, 511  
deleting, 293  
destroying, 297  
displaying, 403, 517  
displaying the end, 499  
displaying the head, 369  
excluding, from commands, 103, 339  
executable, 20  
    types of, 175  
finding, 46, 52, 179, 343  
moving, 418, 453  
reading and writing, 176, 177, 178, 179, 182  
    with DO, 326  
    with FOR, 352  
renaming, 453  
searching for, 187  
selecting, 94, 462  
    date, time, and size ranges, 98  
    exclusion ranges, 103  
    extended parent directory names, 94  
    include lists, 105  
    multiple filenames, 104  
    switches for, 111  
    wildcards, 95  
selecting lines from, 187  
size of, 178, 304  
testing for existence, 379  
time stamp, 17, 176, 178, 183, 304, 507  
    and networks, 289, 424  
    copying, 512  
    modifying, 511  
true path of, 188, 516  
**FILES** function~@~, 177  
**FILESEEK** function~@~, 178

**FILESEEKL** function~@~, 178  
**FILESIZE** function~@~, 178  
**FILETIME** function~@~, 178  
**FILEWRITE** function~@~, 178  
**FILEWRITEB** function~@~, 179  
**FINDCLOSE** function~@~, 179  
**FINDFIRST** function~@~, 179  
Finding files or text, 46, 52, 343  
**FINDNEXT** function~@~, 179  
**FirewallHost**, 211  
**FirewallPassword**, 211  
**FirewallUser**, 211  
**FLOOR** function~@~, 179  
**Fonts**  
    and box and line drawing, 329  
    status bar, 226  
    tool bar, 227  
**FOR**, 118, 131, 349  
    and **GOSUB**, 356  
    environment variables, 355  
Form feed character, 120  
**FORMAT** function~@~, 179  
**FREE**, 359  
**FSTYPE** function~@~, 180  
**FTP**, 114  
    and **COPY /U**, 289  
    and date ranges, 101  
    and **MOVE /U**, 424  
    and specific commands, 114  
    and time ranges, 101  
    changing current directory, 273  
    opening server connection, 384  
**FTYPE**, 360  
    reading from a file, 361  
**FULL** function~@~, 180  
**FUNCTION**, 362  
    reading from a file, 362  
**Functions**  
    removing, 521  
    testing for existence, 379  
**General Concepts**, 7  
**GETDIR** function~@~, 180  
**GETFILE** function~@~, 180

---

GETFOLDER function~@~, 181

GLOBAL, 118, **363**

GOSUB, 131, **365**, 457  
    and FOR, 356

GOTO, 131, **367**  
    and DO, 327  
    and IFF, 383  
    and SWITCH, 498

HEAD, 369

Help

- at command line, 77
- in Take Command, 47
- location of files, in 4DOS, 213
- on Take Command, **55**
- options, in 4DOS, 212
- quick or “TTY” style, 56
- system, 5

HELP command, 77, 371

HISTORY, 372

History list, **61**, 372

- and secondary shells, 260
- controlling, 223, 373
- copying commands to end, 62, 222
- global, 213
  - in upper memory, 217
- inheritance, in secondary shells, 237
- local, 213
- local and global, **65**, 493
- moving commands to end, 62, 223
- reading from a file, 373
- size of, 212
- size of list, 416

History windows

- command line, 64
- directory, 74

HLOGFILE variable~\_~, 162

HOST variable~\_~, 162

HOURL variable~\_~, 162

HTTP, 115

- and specific commands, 115

HTTP servers

- and COPY, **284**, 289

- and MOVE, **420**, 424

- and proxy server, 214

HTTP URLs

- on command line, 113

- waiting for browser, 114, 116

HWPFILE variable~\_~, 163

IDOW function~@~, 181

IDOW variable~\_~, 163

IDOWF function~@~, 181

IDOWF variable~\_~, 163

IF, 118, 131, **374**

- conditions, 375

IF function~@~, 181

IFF, 131

- and GOSUB, 366

- and GOTO, 383

- conditions, 375

IFTP, 384

INC function~@~, 181

Include lists, 105

INDEX function~@~, 181

INI file~.~, **203**, 429

- advanced directives, 236

- and installation, 204

- and SETDOS, 207, 217, 474

- changing, for secondary shells, 237

- color directives, 228

- comments in, 207

- configuration directives, 217

- directives, 206

- errors in, 207, 212, 214

- global section, 205

- initialization directives, 209

- key mapping directives, 230

- location of, and multitaskers, 208

- primary section, 206

- prompts during execution, 212

- querying settings, 184

- secondary section, 206

INIREAD function~@~, 181

INIWRITE function~@~, 181

---

INKEY, 131, **387**  
INPUT, 131, 231, **390**  
Input and output, commands for, 242  
INSERT function~@~, 182  
INSTR function~@~, 182  
INT 2E support (4DOS), 237  
INT function~@~, 182  
Internal variables, 154  
    in aliases, 259  
    in batch files, 129  
International character sets, 278, 329, 330  
Introduction and Installation Guide, 5  
IP variable~\_~, 163  
ITERATE, in DO, 324  
KBHIT variable~\_~, 163  
KEYBD, 392  
Keyboard, 27  
    checking buffer, **163**, 389, 391  
    clearing buffer, 388, 391, 396  
Keys  
    Alt, 42  
    codes, 25  
    F11 and F12, using, 35  
    mapping, 230  
        clearing default key map, 236  
    names, **34**, 255  
KEYS command, 393  
KEYSTACK, 93, 131, 141, **394**  
    and ACTIVATE, 94, 395  
Keystrokes, sending to applications, 394  
KSTACK variable~\_~, 163  
KSTACK.COM, 394  
    detecting, 163  
LABEL function~@~, 182  
Labels, in batch files, 365, 367  
LASTDISK variable~\_~, 163  
LEAVE, in DO, 324  
LEFT function~@~, 182  
LEN function~@~, 182  
LFN function~@~, 182  
LFNFOR, 399  
LH, 400  
Line feed character, 120  
LINE function~@~, 182  
LINES function~@~, 183  
Lines, drawing, on screen, 330, 331  
LIST, 53, 403  
    default colors, 229  
    keys used with, 231, 235  
    print device for, 225  
    screen size, 226, 478  
LOADBTM, 127, 131, **410**  
LOADHIGH, 400  
LOCK, 411  
LOG, 140, **412**  
LOGFILE variable~\_~, 163  
Logging, 412  
    file names, 162, 163, 222, 224  
    of errors, 223  
    Windows NT log, 338  
Loops, in batch files, 324  
LOWER function~@~, 183  
LPT function~@~, 183  
MailAddress, 214  
MailPort, 214  
MailServer, 214  
MAKE utilities, 135  
MAKEAGE function~@~, 183  
MAKEDATE function~@~, 184  
MAKETIME function~@~, 184  
MASTER function~@~, 184  
MAX function~@~, 184  
MD, 414  
Memory  
    amount of, 172  
    expanded (EMS), 172  
    extended, 176  
        XMS, 196  
    location of DOS, 526  
    status, 416  
    types, and 4DOS swapping, **214**, 495

---

- 
- upper memory blocks (UMBs),  
215, 216, 217, 400
  - MEMORY, 416
  - Menus, in Take Command, 38, 42
    - Apps, 44
    - Edit, 44
    - File, 43
    - Help, 47
    - Options, 45
    - Utilities, 46
  - Messages, displaying, 332, 334
  - Microsoft SMARTDRV, and 4DOS,  
238
  - Microsoft Windows
    - and screen colors, 33
    - and secondary shells, 208
    - default command processor  
(Windows NT / 2000 / XP), 8
    - detecting, 161, 164
    - directories
      - retrieving, 164, 165
      - searching for executable  
files, 20, 21, 22, 79, 109,  
431
    - event log, 338
    - Explorer, starting, 492
    - file associations, 113, 262, 360
      - and executable extensions,  
113
      - displaying and modifying,  
262
      - starting applications, 79, 80,  
113
    - file types, 113, 360
      - displaying and modifying,  
360
    - keystrokes, for cut, copy, and  
paste, 41, 219
    - mode, 164
    - restarting, 43
    - retrieving memory information,  
190
    - retrieving system information,  
189
    - retrieving system metrics, 190
    - run time, 165
    - shortcuts, 485
    - shutting down, 43
    - shutting down, Windows 95 and  
Windows NT, 450
    - user name, 165
    - version, 165
  - MIN function~@~, 184
  - MINUTE variable~\_~, 163
  - MKDIR, 414
  - MKLNK, 417
  - MONITOR variable~\_~, 163
  - MONTH function~@~, 184
  - MONTH variable~\_~, 163
  - Mouse
    - cursor, shape of, 212
    - in 4DOS popup windows, 224
    - reading button responses, 388
  - MOUSE variable~\_~, 163
  - MOVE, 418
    - and SELECT, 419
  - MSGBOX, 131, 425
  - Multiple commands, 76
    - disabling, 479
    - in aliases, 252
  - NAME function~@~, 184
  - NDP variable~\_~, 163
  - Networks
    - disk swapping on, 238
    - file systems, 12
    - file time stamps on, 289, 424
  - NOT., in IF and IFF~.~, 374, 380, 382
  - Novell NetWare, and 4DOS, 238
  - Numbers
    - absolute value, 169
    - calculating, 173, 471
    - comparing, 375
    - converting bases, 171
    - formatting, 171, 179
    - maximum, 184
-

---

    minimum, 184  
    parts of, 172, 182  
    testing strings for, 172, 184  
Numeric coprocessor, 163  
NUMERIC function~@~, 184  
ON, 131, 427  
Operating system, 7  
    detecting, 161  
OPTION, 429  
    querying settings, 184  
OPTION function~@~, 184  
Options, in commands, 245  
Or (| |) in conditional commands, 117  
OR., in IF and IFF~., 374, **380**, 382  
Paged output, 80, 226  
Parameter character, 224, 478  
    compatibility between products, 197  
    in aliases, 258  
    in batch files, 129  
    saving and restoring, 482  
Parameters  
    expansion, 200  
    in aliases, 257, 259  
    in batch files, 128, 483  
Parent directory, 15, 94, 273, 275, 414  
PassiveFTP, 214  
Password entry, 388, 391  
Pasting text, on command line, 41, 44  
PATH command, 431  
PATH environment variable, 153  
    "." in, 432  
    and @SEARCH variable function, 187  
    changing, 336, **431**  
    format of, 432  
    invalid directory in, 433  
    length of, 432  
    searching for executable files, **20**, 79, 109, 431  
    viewing, 432  
PATH function~@~, 185  
Path, of a file, 10, 14  
    and aliases, 125  
    extracting from full file name, 185  
    finding full path name, 180  
    finding true path, 188, 516  
    if omitted, 14  
    in executable extensions, 108  
    in include lists, 105  
    length of, 14  
PATHEXT environment variable, 153  
PAUSE, 131, 140, **434**  
PID variable~\_~, 163  
PING function~@~, 185  
PIPE variable~\_~, 163  
Pipes, **91**, 505  
    and LIST, 409  
    disabling, 479  
    in 4NT and Take Command, 92  
    named, 177  
    nested, 92  
    testing for, 163  
PLAYAVI, 435  
PLAYSOUND, 436  
POPD, 323, **437**, 443  
Popup windows, 35, 224  
    and mouse, in 4DOS, 224  
    color of, 229  
PPID variable~\_~, 163  
Previous working directory, returning to, 273, 276  
PRINT, 438  
Printer  
    checking status of, 183  
    sending control codes to, 121  
    sending files to, 281, **438**  
    setup, 43  
    use in LIST, 407  
Printing scrollbar buffer, 43  
Process ID number, 163, 501, 502  
    child process, 159  
    detached process, 160  
    in log files, 412  
    parent process, 163

---

---

STARTed process, 164  
 PROMPT command, 439  
     ANSI sequences in, 120, 442  
 PROMPT environment variable, 154  
 Prompting  
     before command execution, 248  
     during file processing, 80  
     during paged output, 80  
 Proxy, 214  
 ProxyPort, 214  
 PUSH, 323, 437, **443**  
 QUERYBOX, 131, **445**  
 QUIT, 131, 270, 271, **446**  
 Quoting  
     arguments, 201  
     file names, 16  
 RAM disk, 11  
 RANDOM function~@~, 185  
 Ranges  
     date, time, and size, 98  
     file exclusion, 103  
 RD, 447  
     and recycle bin, 225, 448  
 READSCR function~@~, 185  
 READY function~@~, 185  
 REBOOT, 449  
 RECYCLE, 451  
 Recycle bin  
     and DEL, 225, **295**, 296, 297  
     and RD, 225, 448  
 Redirection, 88  
     alternative methods, 505, 534  
     and batch files, 90  
         capturing output, 141  
     and command grouping, 90, 119  
     disabling, 479  
     nested, 90  
     numeric file handles, 90  
     preventing file overwrites, 90,  
         224, 478  
 REGCREATE function~@~, 185  
 Registration, 6  
 Registry  
     creating keys, 185  
     querying values, 186  
     setting values, 186  
 REGQUERY function~@~, 186  
 REGSET function~@~, 186  
 REM, 131, **452**  
 REMOTE function~@~, 186  
 REMOVABLE function~@~, 186  
 REN, 418, **453**  
 RENAME, 418, **453**  
 REPEAT function~@~, 186  
 REPLACE function~@~, 186  
 RETURN, 131, 365, **457**  
     and DO, 327, 457  
 REXX function~@~, 187  
 REXX support, 144, **146**, 187  
 RIGHT function~@~, 187  
 RMDIR, 447  
 Root directory, 14  
 ROW variable~\_~, 163  
 ROWS variable~\_~, 164  
 Screen  
     clearing, 279  
     copying to clipboard, 41, 44  
     reading text from, 185  
     size, 39, 225, 226, 478  
         detecting, 160, 164, 165  
     size, in 4DOS and 4NT, 28  
     virtual width, **39**, 160, 225, 226  
 SCREEN, 131, **458**  
 Scrollback buffer, 40  
     clearing, 41, **279**  
     copying to clipboard, 41, 44  
     keys, 40, 121, 226, 234  
     printing, 43  
     saving, 43  
     size of, 40, 215  
 SCRPUT, 131, **460**  
 SEARCH function~@~, 187  
 Searching  
     for executable files, 20  
     for files or text, 46, 52, 343  
 SECOND variable~\_~, 164

---

---

**SELECT**, 462  
     and command grouping, 465  
     and include lists, 105  
     and MOVE, 419  
     color-coded displays, 229, 467  
     default colors, 230, 464  
     keys used with, 231  
     screen size, 226, 478  
**SELECT function**~@~, 187  
**SELECTED variable**~\_~, 164  
**Selecting**  
     files, 462  
     text, in command window, 44  
**Sending email**, 469  
**SENDMAIL**, 132, 469  
     attachments, 469  
**SET**, 151, 470  
**SETDOS**, 474  
     /A(NSI), 218, 280, **475**  
     /B(rightBG), 32, 229, **475**  
     /C(ompound character), 197, 219, **475**  
     /D(escriptions), 221, **476**  
     /E(scape character), 197, 221, **476**  
     /G (separators), 477  
     /I(nternal), 248, 379, **477**  
     /L(ine), 223, **477**  
     /M(ode), 221, **478**  
     /N(o clobber), 224, **478**  
     /P(arameter character), 197, 224, **478**  
     /R(ows), 226, **478**  
     /S(hape), 220, **478**  
     /U(ppper), 155, 228, 316, 346, **479**  
     /V(erbose), 128, 218, **479**  
     /W (Switch character), 479  
     /X (expansion), 143, 170, 180, 201, 202, 255, **479**  
     /Y (debug), **481**  
         and .INI file, 207, 474  
**SETLOCAL**, 130, 132, 137, **482**, 524  
**SFN function**~@~, 187  
**Shell**, 1  
     level, 164, 440  
     primary, 9  
     secondary, 9  
         inheritance, 207, 237, 474  
**SHELL variable**~\_~, 164  
**SHIFT**, 129, 132, **483**  
**SHORTCUT**, 485  
**SHRALIAS**, 66, 76, 260, **487**  
**Size ranges**, 98  
**SNPP**, 132, 489  
**Sound files**, playing, 436  
**Special characters**, compatibility  
     between products, 197  
**Stack size**, in 4DOS, 238  
**Standard error**, **88**, 333, 334  
**Standard input**, **88**, 505, 534  
     and LIST, 409  
**Standard output**, **88**, 505, 534  
**START**, 132, 490  
**Starting applications**, 44, 78, 490  
**STARTPID variable**~\_~, 164  
**Startup command line**, viewing, in 4START / TCSTART, 133  
**Status bar**, 38  
     enabling, 46, 226  
     font, 226  
**Status tests**, in IF and IFF, 378  
**Strings**  
     comparing, 375  
     counting words, 195  
     extracting substrings, 182, 187, 188  
     finding words, 176, 195  
     formatting, 171, 179  
     input of, 390  
     inserting text, 182  
     length, 182  
     removing blanks, 188  
     removing characters, 187  
     repeating characters, 186  
     replacing characters, 186  
     searching, 181

---



---

testing for numeric digits, 172, 184

STRIP function~@~, 187

Subdirectories, 13

- attributes of, 267
- changing. *see* Changing directories
- commands for, 242
- copying, 288
- creating, 414
- deleting files from, 297
- descriptions for, 300
- displaying tree, 514
- executing commands in, 363
- hidden, 363
- moving, 418, 423
- name, maximum length of, 14
- removing, 297, 423, **447**
- renaming, 453
- selecting, 180, 181
- testing for existence, 379

Subdirectory attribute. *see* File attributes

Subroutines. *see* Batch files, subroutines

SUBSTR function~@~, 188

Support, 6

Swapping, 214

- enabling and disabling, 495
- file names used by, 215, 238
- types of, 214

SWAPPING command, 495

SWAPPING variable~\_~, 164

SWITCH, 132, 496

- and GOTO, 498

Switches

- for file attributes**, 112
- for file selection, 111
- in commands, 245
- switch character, 479

SYSERR variable~\_~, 164

System

- attribute. *see* File attributes

- configuration, commands for, 242
- name, 164
- rebooting, 449

System errors. *see* Errors

Tab character, 120

Tab stops, 227

TAIL, 499

Take Command

- as DDE client, 292
- as DDE server, 149
- compatibility with 4DOS and 4NT, 40, **197**, 226
- window, 37. *See also* Command window.

TASKEND, 501

TASKLIST, 502

Tasks

- listing, 502
- terminating, 501

TCSTART and TCEXIT, 133

- and MAKE utilities, 135
- location of, 215

TCTOOLBAR, 132, 503

Technical support, 6

TEE, 92, **505**

TEMP environment variable, 154

TEMP4DOS environment variable, 154

Text

- finding, 46, 52, 343

TEXT, 132, **506**

THEN, in IFF, 382

Thousands character, **227**, 477

- saving and restoring, 482

Time

- converting, 188
- display format, 217
- file. *see* Files, time stamp
- formatting, 184
- ranges, 98
- retrieving, 162, 163, 164
- setting, 507

TIME, 507

---

TIME function~@~, 188  
TIME variable~\_~, 164  
TIMER, 132, 188, **508**  
TIMER function~@~, 188  
TimeServer, 215  
TITLE, 132, 510  
Tool bar, 38, 45, **51**  
    configuring, 503  
    enabling, 46, 227  
    font, 227  
TOUCH, 511  
TRANSIENT variable~\_~, 164  
TREE, 514  
TRIM function~@~, 188  
TRUENAME, 516  
TRUENAME function~@~, 188  
TSRs  
    and .BTM files, 410  
    and SETLOCAL, 482  
    and swapping state, 495  
TYPE, 517  
UNALIAS, 137, **519**  
UNC names, 13  
UNFUNCTION, **521**  
Unicode. *see* Character sets  
UNIQUE function~@~, 188  
Unknown commands, handling, 261  
UNKNOWN\_CMD alias, 23, **261**  
UNLOCK, 523  
UNSET, 130, 151, **524**  
UNTIL, in DO, 324  
Upgrades, 6  
Upper case. *see* Case  
UPPER function~@~, 189  
URLs, 113  
Variable expansion. *see* Environment  
    variables, expansion  
Variable functions, 166. *See also*  
    names of individual functions.  
    in aliases, 196, **259**  
    in batch files, **129**, 196  
VER, 526  
VERIFY, 289, 424, **527**

VERINFO function~@~, 189  
Version numbers, 157, 161, 165, 526  
Vertical text display, 529  
Video files, playing, 435  
Video hardware, 28  
    detecting, 163, 164  
VIDEO variable~\_~, 164  
View console, 45  
Virtual screen width. *see* Screen  
VOL, 528  
Volume. *see* Disk drives  
Volume label attribute. *see* File  
    attributes  
VSCRPUT, 132, **529**  
Warm reboot, 449  
Warning mark. *see* !  
WATTRIB function~@~, 189  
WHICH, 531  
WHILE, 324  
WILD function~@~, 189  
Wildcards, 95  
    and @SEARCH variable function,  
        187  
    and filename completion, 68  
    and renaming subdirectories, 456  
    comparing, 189  
    expanding, 175  
    extended, 96  
    in executable extensions, 109  
    in FFIND search text, 344  
    in include lists, 105  
    in LIST search text, 405  
    in multiple filenames, 104  
WIN variable~\_~, 161, 164  
WINCLASS function~@~, 189  
WINDIR variable~\_~, 164  
WINDOW, 532  
Windows  
    commands for, 243  
    console, 80  
    controlling state of, 217, 249, 532  
    for applications, 80  
    name of program in, 189

---

retrieving text from, 164  
state of, 191  
Take Command, 37. *See also*  
    Command window.  
        size of, 39  
testing for existence, 379  
title, 164, 165, 189  
title of, 249, 510, 532  
title updates, 4NT and Take  
    Command, 228  
Windows 95, 98, NT, 2000, or XP. *see*  
    Microsoft Windows.  
WINEXENAME function~@~, 189  
WINFGWINDOW variable~\_~, 164  
WININFO function~@~, 189  
WINMEMORY function~@~, 190  
WINMETRICS function~@~, 190  
WINNAME variable~\_~, 164  
WINSTATE function~@~, 191  
WINSYSDIR variable~\_~, 165  
WINSYSTEM function~@~, 191  
WINTICKS variable~\_~, 165  
WINTITLE variable~\_~, 165  
WINUSER variable~\_~, 165  
WINVER variable~\_~, 165  
WORD function~@~, 195  
WORDS function~@~, 195  
XMS function~@~, 196  
XOR., in IF and IFF~.~, 374, **380**, 382  
XPIXELS variable~\_~, 165  
Y, 92, **534**  
YEAR function~@~, 196  
YEAR variable~\_~, 165  
YPIXELS variable~\_~, 165  
Zero-length files, creating, 90, 452,  
    512

