

4NT 6.01 and Take Command 6.01

*Published By
JP Software Inc.
P.O. Box 328
Chestertown, MD 21620 USA
(410) 810-8818
Fax (410) 810-0026*

Table of Contents

Part I Overview	2
Part II The Basics	3
1 Starting the Command Processor	3
Startup Command	4
Automatic Startup and Termination Programs	6
Exit Codes	8
Desktop Integration	8
2 The Command Line	10
Command-Line Editing	11
Command History & Recall	13
Command History Window	15
Local & Global History Lists	16
Command Names & Parameters	17
Filename Completion	17
Appending Backslashes to Directory Names	19
Customizing Filename Completion	19
Converting Between Long & Short Filenames	20
Filename Completion Window	21
Escape Character	21
Variable Completion	22
Automatic Directory Changes	22
Directory History Window	23
Multiple Commands	24
Command-Line Length Limits	25
Expanding & Disabling Aliases	25
Scrolling & History Keystrokes	25
Conditional Commands	26
Command Grouping	27
Starting Applications	28
Waiting for Applications to Finish	29
Command Parsing	29
3 Conditional Expressions	31
4 File Selection	36
Wildcards	36
Attribute Switches	39
Date, Time & Size Ranges	40
Size Ranges	42
Date Ranges	42
Time Ranges	44
File Exclusion Ranges	45
Multiple Filenames	45
Include Lists	46
Extended Parent Directory Names	47
LFN File Searches	47
Executable Extensions	48

@File Lists	49
Switches for File Selection	51
Using Internet URLs	51
FTP/HTTP Servers	52
5 Directory Navigation	54
Extended Directory Searches	56
CDPATH (feature)	59
6 Input / Output Manipulation	60
Redirection and Piping	60
Redirection.....	61
Piping	64
ANSI X3.64 Support	64
Keystack	65
Page and File Prompts	65
7 The Take Command Interface	66
The Take Command Window	66
Take Command Menus.....	68
File Menu	68
Edit Menu	69
Apps Menu	69
Options Menu	70
Utilities Menu	70
Help Menu	71
Take Command Dialogs.....	71
Run Program Dialog.....	72
Tool Bar Dialog	72
Find Files/Text Dialog.....	73
Edit Descriptions Dialog.....	74
Aliases Window.....	74
Environment Window.....	74
Functions Window.....	74
Tool Bar	75
Status Bar.....	75
Using the Scrollback Buffer.....	75
Highlighting & Copying Text.....	76
Resizing the Take Command Window.....	77
Using a Windows Command Line	77
Console Applications & the Console Window	78
Caveman (technical information)	79
Using Drag & Drop	80
DDE Support	81

Part III Configuration Options

81

1 Configuration Dialog	82
Startup Options Tab	82
Window Options Tab	83
Editing Options Tab	84
Colors Options Tab	85
History Options Tab	85
Syntax Options Tab	86
Batch Debugger Tab	86
Miscellaneous Options Tab	86

Internet Options Tab	87
Caveman Options Tab	87
2 Initialization Files	89
Initialization Directives	93
4StartPath.....	93
ConsoleColumns.....	94
ConsoleRows.....	94
DirHistory.....	94
DuplicateBugs.....	94
HideConsole.....	94
History	95
IBeamCursor.....	95
INIQuery	95
LocalAliases.....	96
LocalDirHistory.....	96
LocalFunctions.....	96
LocalHistory	96
PauseOnError.....	96
ScreenBufSize.....	96
TCStartPath.....	96
TimeServer.....	97
TreePath.....	97
UnicodeOutput.....	97
WindowState.....	97
WindowX, WindowY, WindowWidth, WindowHeight.....	97
Configuration Directives	97
AmPm	99
ANSI	99
AppendToDir.....	99
BatchEcho.....	100
BeepFreq.....	100
BeepLength.....	100
CDDWinLeft, CDDWinTop, CDDWinWidth, CDDWinHeight.....	100
CommandSep.....	100
CompleteHidden.....	101
CopyPrompt.....	101
CUA	101
CursorIns.....	101
CursorOver.....	101
DecimalChar.....	102
DelGlobalQuery.....	102
DescriptionMax.....	102
DescriptionName.....	102
Descriptions	102
EditMode.....	102
Editor	103
EscapeChar.....	103
EvalMax	103
EvalMin.....	103
ExecWait.....	103
FileCompletion (directive).....	103
FirewallHost.....	104
FirewallPassword.....	104
FirewallType.....	104

FirewallUser	104
FTPCFG	104
FuzzyCD (Extended Directory Search)	104
HistCopy	105
HistDups	105
HistLogName	105
HistLogOn	105
HistMin	105
HistMove	105
HistWrap	106
ListRowStart	106
LogErrors	106
LogName	106
LogOn	106
MailAddress	106
MailPassword	107
MailPort	107
MailServer	107
MailUser	107
NoClobber	107
ParameterChar	107
PassiveFTP	108
PathExt (directive)	108
PopupWinLeft, PopupWinTop, PopupWinWidth, PopupWinHeight	108
Proxy	109
ProxyPassword	109
ProxyPort	109
ProxyUser	109
RecycleBin	109
ScreenColumns	109
ScreenRows	109
ServerCompletion	109
SSLPort	110
SSLProvider	110
SSLStartMode	110
StatusBarText	110
StatusBarOn	110
SwapScrollKeys	111
TabStops	111
ThousandsChar	111
ToolBarOn	112
ToolBarText	112
UnixPaths	112
UpdateTitle	112
Win32SFNSearch	112
Color Directives	113
CDDWinColors	113
ColorDIR (directive)	113
InputColors	113
ListboxBarColors	113
ListColors	113
ListStatBarColors	114
PopupWinColors	114
SelectColors	114

SelectStatBarColors.....	114
StdColors.....	114
Key Mapping Directives	114
General Input Keys.....	115
Backspace	116
BeginLine	116
Copy (directive).....	116
Del (directive)	116
DelToBeginning.....	116
DelToEnd	116
DelWordLeft	116
DelWordRight	117
Down	117
EndLine	117
EraseLine	117
ExecLine	117
Ins	117
Left	117
NormalKey	117
Paste	118
Right	118
Up	118
WordLeft	118
WordRight	118
Command-Line Editing Keys.....	118
AddFile	119
AliasExpand	119
CommandEscape.....	119
DelHistory	119
EndHistory	119
Help (directive).....	119
HelpWord	120
LFNToggle	120
LineToEnd	120
NextFile	120
NextHistory	120
NormalEditKey.....	120
PopFile	120
PrevFile	121
PrevHistory	121
RepeatFile	121
SaveHistory	121
Scrollbar Buffer Keys.....	121
ScrollUp	121
ScrollDown	121
ScrollPgUp	122
ScrollPgDn	122
Popup Window Keys.....	122
DirWinOpen	122
HistWinOpen	122
NormalPopupKey.....	122
PopupWinBegin.....	123
PopupWinDel	123
PopupWinEdit	123

PopupWinEnd.....	123
PopupWinExec.....	123
LIST Keys.....	123
ListExit.....	124
ListFind.....	124
ListFindReverse.....	124
ListHex.....	124
ListHighBit.....	124
ListInfo.....	124
ListNext.....	124
ListOpen.....	124
ListPrevious.....	125
ListPrint.....	125
ListUnicode.....	125
ListWrap.....	125
NormalListKey.....	125
Advanced Directives	125
ClearKeyMap.....	126
Debug.....	126
Include.....	126
NextINIFile.....	126
SaveDirCase.....	126

Part IV Internal Commands 127

1 List by Name	127
2 List by Category	131
3 ? (command)	136
4 ACTIVATE	136
5 ALIAS	138
6 ASSOC	145
7 ATTRIB	146
8 BATCOMP	149
9 BDEBUGGER	149
10 BEEP	156
11 BREAK	157
12 CALL	157
13 CANCEL	158
14 CD / CHDIR	159
15 CDD	160
16 CHCP	162
17 CLS	163
18 COLOR	163
19 COPY	164
20 DATE	170
21 DDEEXEC	171
22 DEL / ERASE	172

23	DELAY	175
24	DESCRIBE	176
25	DETACH	178
26	DIR	179
27	DIRHISTORY	189
28	DIRS	190
29	DO	191
30	DRAWBOX	195
31	DRAWHLINE	196
32	DRAWVLINE	197
33	ECHO	198
34	ECHOERR	199
35	ECHOS	199
36	ECHOSERR	200
37	ENDLOCAL	200
38	ESET	201
39	EVENTLOG	203
40	EXCEPT	204
41	EXIT	205
42	FFIND	206
43	FOR	210
44	FREE	216
45	FTYPE	217
46	FUNCTION	218
47	GLOBAL	220
48	GOSUB	221
49	GOTO	222
50	HEAD	224
51	HELP	225
52	HISTORY	226
53	IF	227
54	IFF	228
55	IFTP	229
56	INKEY	231
57	INPUT	233
58	KEYBD	234
59	KEYS	235
60	KEYSTACK	236
61	LIST	237

62	LOADBTM	242
63	LOG	242
64	MD & MKDIR	243
65	MEMORY	245
66	MKLNK	245
67	MOVE	246
68	MSGBOX	251
69	ON	252
70	OPTION	253
71	PATH	254
72	PAUSE	256
73	PDIR	256
74	PLAYAVI	260
75	PLAYSOUND	261
76	POPD	261
77	PRINT	262
78	PROMPT	262
79	PUSHD	264
80	QUERYBOX	265
81	QUIT	266
82	RD	267
83	REBOOT	268
84	RECYCLE	269
85	REM	269
86	REN / RENAME	270
87	RETURN	272
88	SCREEN	273
89	SCRPUT	274
90	SELECT	275
91	SENDMAIL	279
92	SET	281
93	SETDOS	283
94	SETLOCAL	287
95	SHIFT	288
96	SHORTCUT	289
97	SHRALIAS	290
98	SMPP	290
99	SNPP	291
100	START	291

101	SWITCH	295
102	TAIL	296
103	TASKEND	298
104	TASKLIST	299
105	TCTOOLBAR	299
106	TEE	300
107	TEXT	301
108	TIME	302
109	TIMER	303
110	TITLE	304
111	TOUCH	304
112	TREE	307
113	TRUENAME	308
114	TYPE	308
115	UNALIAS	310
116	UNFUNCTION	311
117	UNSET	312
118	VER	313
119	VERIFY	314
120	VOL	314
121	VSCRPUT	314
122	WHICH	315
123	WINDOW	316
124	Y	317

Part V Aliases & Batch Files 318

1	Aliases	318
2	Batch Files	321
	.BAT, .CMD & .BTM Files	321
	Using .BAT Files Under 4NT	321
	Echoing in Batch Files	322
	Batch File Parameters	322
	Using Environment Variables	324
	Batch File Commands	325
	Interrupting a Batch File	327
	Detecting 4NT or Take Command	327
	Using Aliases in Batch Files	327
	Debugging Batch Files	329
	String Processing	329
	Batch File Line Continuation	331
	Batch File Compression	331
	Argument Quoting	332
	REXX Support	334
	EXTPROC Support	334

3 Special Character Compatibility	335
Part VI Variables & Functions	336
1 System Variables	339
CDPATH (variable)	339
CMDLINE	339
COLORDIR (variable)	339
COMSPEC	340
COPYCMD	340
DIRCMD	340
FILECOMPLETION (variable)	340
HISTORYEXCLUDE	341
PATH (variable)	341
PATHEXT (variable)	341
PROMPT (variable)	341
RECYCLEXCLUDE	341
TEMP	342
TITLEPROMPT	342
TMP	342
TREEEXCLUDE	342
2 Internal Variables	342
List by Category	343
? (variable)	346
_?	346
=	347
+	347
_4VER	347
_ACSTATUS	347
_ALT	348
_ANSI	348
_BATCH	348
_BATCHLINE	348
_BATCHNAME	348
_BATCHTYPE	348
_BATTERY	348
_BATTERYLIFE	348
_BATTERYPERCENT	348
_BG	349
_BOOT	349
_BUILD	349
_CAPSLOCK	349
_CHILDPID	349
_CI	349
_CMDLINE	349
_CMDPROC	349
_CMDSPEC	349
_CO	349
_CODEPAGE	350
_COLUMN	350
_COLUMNS	350
_COUNTRY	350
_CPU	350
_CPUUSAGE	350

_CTRL	350
_CWD	350
_CWDS	351
_CWP	351
_CWPS	351
_DATE	351
_DATETIME	351
_DAY	351
_DETACHPID	351
_DISK	351
_DNAME	351
_DOS	351
_DOSVER	352
_DOW	352
_DOWF	352
_DOWI	352
_DOY	352
_ECHO	352
_FG	352
_FTPERROR	352
_HLOGFILE	353
_HOST	353
_HOUR	353
_HWPROFILE	353
_IDOW	353
_IDOWF	353
_IMONTH	353
_IMONTHF	353
_ININAME	353
_IP	354
_ISODATE	354
_KBHIT	354
_LALT	354
_LASTDISK	354
_LCTRL	354
_LOGFILE	354
_LSHIFT	354
_MINUTE	354
_MONTH	354
_MONTHF	354
_NUMLOCK	354
_PID	354
_PIPE	354
_PPID	355
_RALT	355
_RCTRL	355
_ROW	355
_ROWS	355
_RSHIFT	355
_SCROLLLOCK	355
_SECOND	355
_SELECTED	355
_SHELL	355
_SHIFT	355

_SHRALIAS	355
_STARTPATH	356
_STARTPID	356
_SYSERR	356
_TIME	356
_TRANSIENT	356
_UNICODE	356
_WINDIR	356
_WINFGWINDOW	356
_WINNAME	356
_WINSYSDIR	356
_WINTICKS	356
_WINTITLE	356
_WINUSER	356
_WINVER	357
_XPIXELS	357
_YEAR	357
_YPIXELS	357
ERRORLEVEL	357
3 Variable Functions	357
List by Category	359
Date Formats	363
@ABS	364
@AGEDATE	364
@ALIAS	364
@ALTNAME	364
@ASCII	365
@ATTRIB	365
@CAPS	366
@CDROM	366
@CEILING	366
@CHAR	366
@CLIP	367
@CLIPW	367
@COLOR	367
@COMMA	368
@CONSOLE	368
@CONVERT	368
@CRC32	368
@CWD	368
@CWDS	369
@DATE	369
@DAY	369
@DEC	370
@DECIMAL	370
@DESCRIPT	370
@DEVICE	370
@DIGITS	370
@DIRSTACK	371
@DISKFREE	371
@DISKTOTAL	371
@DISKUSED	372
@DOMAIN	372
@DOSMEM	372

@DOW	372
@DOWF	372
@DOWI	373
@DOY	373
@ENUMSERVERS	373
@ENUMSHARES	373
@ERRTEXT	374
@EVAL	374
@EXEC	376
@EXECSTR	377
@EXETYPE	377
@EXPAND	378
@EXT	378
@FIELD	379
@FIELDS	379
@FILEAGE	380
@FILECLOSE	380
@FILEDATE	380
@FILENAME	381
@FILEOPEN	381
@FILEREAD	382
@FILES	382
@FILESEEK	383
@FILESEEKL	383
@FILESIZE	384
@FILETIME	384
@FILEWRITE	385
@FILEWRITEB	385
@FINDCLOSE	385
@FINDFIRST	386
@FINDNEXT	386
@FLOOR	387
@FORMAT	387
@FORMATN	388
@FSTYPE	388
@FULL	388
@FUNCTION	388
@GETDIR	389
@GETFILE	389
@GETFOLDER	389
@HISTORY	390
@IDOW	390
@IDOWF	390
@IF	390
@INC	390
@INDEX	391
@INIREAD	391
@INIWRITE	392
@INSERT	392
@INSTR	393
@INT	393
@IPADDRESS	394
@IPNAME	394
@ISALNUM	394

@ISALPHA	394
@ISASCII	395
@ISCNTRL	395
@ISDIGIT	395
@ISPRINT	395
@ISPUNCT	396
@ISSPACE	396
@ISXDIGIT	396
@LABEL	396
@LEFT	397
@LEN	397
@LFN	397
@LINE	398
@LINES	398
@LOWER	398
@LTRIM	399
@MAKEAGE	399
@MAKEDATE	399
@MAKETIME	400
@MAX	400
@MD5	400
@MIN	400
@MONTH	400
@NAME	401
@NUMERIC	401
@OPTION	401
@PATH	402
@PING	402
@RANDOM	402
@READSCR	403
@READY	403
@REGCREATE	403
@REGDELKEY	403
@REGEXIST	404
@REGQUERY	404
@REGSET	404
@REGSETENV	404
@REMOTE	404
@REMOVABLE	405
@REPEAT	405
@REPLACE	405
@REXX	405
@RIGHT	406
@RTRIM	406
@SEARCH	406
@SELECT	407
@SFN	407
@STRIP	407
@SUBSTR	407
@SUBST	408
@TIME	408
@TIMER	408
@TRIM	408
@TRUENAME	408

@TRUNCATE	408
@UNC	409
@UNICODE	409
@UNIQUE	409
@UPPER	409
@VERINFO	409
@WATTRIB	410
@WILD	410
@WINCLASS	411
@WINEXENAME	411
@WININFO	411
@WINMEMORY	411
@WINMETRICS	412
@WINSTATE	414
@WINSYSTEM	414
@WORD	416
@WORDS	417
@WORKGROUP	417
@YEAR	417

Part VII Setup & Troubleshooting 418

1 Registration	418
2 Troubleshooting, Service & Support	418
Technical Support	419
Contacting JP Software	421
3 Supported Platforms	421
4 ASCII & Unicode Versions	422
5 Help File	423
6 Error Messages	424

Part VIII Reference Information 430

1 CMD.EXE Comparison	430
2 Limits	437
3 File Systems & File Name Conventions	438
Drives & Volumes	439
File Systems	439
Directories & Subdirectories	440
File Names	441
File Attributes	442
Time Stamps	443
NTFS File Streams	444
4 ASCII, Key Codes & ANSI Commands	445
ASCII Tables	446
Key Codes and Scan Codes Explanation	451
Key Codes and Scan Codes Table	452
ANSI X3.64 Command Reference	456
5 Miscellaneous Reference Information	458
Executable Files & File Searches	458
Windows File Associations.....	461

Primary & Secondary Shells	461
Colors, Color Names & Codes	461
Keys & Key Names	464
Popup Windows	464
Windows System Errors	465
6 Glossary	468
Glossary - 4	468
Glossary - A	469
Glossary - B	470
Glossary - C	471
Glossary - D	474
Glossary - E	476
Glossary - F	477
Glossary - G	478
Glossary - H	479
Glossary - I	479
Glossary - K	480
Glossary - L	480
Glossary - M	481
Glossary - N	481
Glossary - O	481
Glossary - P	482
Glossary - Q	482
Glossary - R	483
Glossary - S	483
Glossary - T	484
Glossary - U	485
Glossary - V	485
Glossary - W	486
Glossary - X	486
Part IX What's New	487
Part X Order Form	489
Part XI Copyright & Version	491
Index	492

ACKNOWLEDGEMENTS

We couldn't produce products like 4NT and Take Command without the dedication and quality work of many people. Our thanks to: On-line Support: Blueberry Hill Communications (Palm Desert, CA). Beta Testers: We can't list all of our beta testers here! A special thanks to all of you who helped make 4NT and Take Command elegant, reliable, and friendly.

Copyright © 2004, JP Software Inc., All Rights Reserved. 4NT is JP Software Inc.'s trademark for its character-mode command processor. Take Command® is a registered trademark 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.

1 Overview



4NT 6.01 and TAKE COMMAND 6.01

Welcome to our help! We have designed this material to accompany our [current](#)^[49] Windows products: **4NT** and **Take Command**. The table below shows their availability on various platforms. For additional information, see [Supported Platforms](#)^[42] and [ASCII and Unicode Versions](#)^[42].

Platform	4NT		Take Command	
	ASCII	Unicode	ASCII	Unicode
Win 98 / 98SE / ME	s	n/a	s	n/a
Win NT/2000/XP/2003	u	s	u	s

n/a not available

u unsupported (may work, but responsibility is yours if something goes wrong)

s supported

Each of these programs is a **command interpreter** or **shell**. That means that they respond to commands you type at the prompt. They are also able to process [batch files](#)^[32] containing sequences of commands.

4NT is designed as a powerful alternative to **CMD.EXE** from Windows NT / 2000 / XP / 2003, and may also be used under Windows 98 and ME.

Take Command is a unique **graphical** command processor combining the power of the command line with the convenience of a GUI interface.

Our products are highly compatible with the default command interpreter that they replace or supplement. This 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.

- ▶ [The Basics](#)^[3]
- ▶ [Commands](#)^[12]
- ▶ [Aliases and Batch Files](#)^[31]
- ▶ [The Environment: Variables and Functions](#)^[33]
- ▶ [Setup and Troubleshooting](#)^[41]
- ▶ [Reference Information](#)^[43]
- ▶ [Copyright and Version](#)^[49]

JP Software Inc.
P.O. Box 328, Chestertown, MD 21620, USA
phone: (410) 810-8818
fax: (410) 810-0026
email: sales@jpsoft.com
web: <http://jpsoft.com/>

2 The Basics

- [Starting the Command Processor](#) ^[3]
- [The Command Line](#) ^[10]
- [File Selection](#) ^[36]
- [Directory Navigation](#) ^[54]
- [Other Features](#) ^[60]
- [The Take Command Interface](#) ^[66]

2.1 Starting the Command Processor

You will typically start the command processor from a Windows shortcut, located on the desktop or in the Quick Launch bar, or from the Programs section of the Start menu. The installation software will *optionally* create both a command processor folder or group and a desktop object which starts the command processor. Usually these are sufficient, but if you prefer you can create multiple desktop objects or items to start the command processor with different startup commands or options, or to run different batch files or other commands; see [Desktop Integration](#) ^[8] for details. You can use these items to run commonly-used commands and batch files directly from the desktop.

Each item or icon represents a different command processor window. You can set any necessary command line parameters for the command processor such as a command to be executed, any desired switches, and the name and path for the [.INI file](#) ^[89]. See [Command Line Options](#) ^[4] for more information on startup command line switches and options.

When you configure a command processor item, place the full path and name for the file in the Command Line field, and put any startup options that you want passed to the command processor (e.g., the name of a startup batch file) after the file name. For example:

```
Command Line:      C:\Program Files\JPSoft\4NT\4NT.EXE C:\GO.BAT
Working directory: C:\
```

You do not need to use the Change Icon button, because *TCMD.EXE* and *4NT.EXE* already contain icons.

When the command processor starts it automatically runs the optional [TCSTART](#) ^[6] or [4START](#) ^[6] program. You can use it to load aliases, functions, and environment variables, and to otherwise initialize the command processor environment.

You can also place the name of a batch file, internal or external command, or alias at the end of the Command Line field for any item (as shown in the example above). The batch file, command, or alias will be executed after *TCSTART* or *4START* but before the first prompt is displayed.

Each Windows program has a command line which can be used to pass information to the program when it starts. The command line is entered in the Command Line field for each shortcut or each item in a Program Manager group (or each item defined under another Windows shell), and consists of the name of the program to execute, followed by any startup options.

The command processor startup command line does not need to contain any information. When invoked with an empty command line, the command processor will configure itself from the [.INI file](#) ^[89], run [TCSTART](#) ^[6] or [4START](#) ^[6], and then display a prompt and wait for you to type a command. However, you may add information to the [startup command line](#) ^[4] that will affect the way the command processor operates.

2.1.1 Startup Command

Some of the options that the command processor recognizes are required in certain circumstances. Others are available if you want finer control over the way the program starts.

The line that starts the command processor will typically include the program name with drive and path, followed by any switches for the program, for example:

```
c:\4NT\4NT.exe @c:\4NT\4NT.ini
alternate format:
c:\4NT\4NT.exe /@c:\4NT\4NT.ini
```

Although the startup command line is usually very simple, you can add several options.

The complete syntax for the **4NT** startup command line is:

```
d:\path\program [d:\path] [[/]@d:\path\inifile] [//iniline...]
[/L][/LA][/LD][/LF][/LH][/Q][/S][/T:bf][/U][/X][/C] command
```

The complete syntax for the **Take Command** startup command line is:

```
d:\path\program [d:\path] [@d:\path\inifile] [//iniline...]
[/L][/LA][/LD][/LF][/LH][/T:bf][/X] [[/C] command]
```

Do not include the square brackets shown in the command line above. They are there to indicate that the items within the brackets are optional. Not all options are available in all products; see below for details.

If you include any of the options below, you should use them in the order that they are described. If you do not do so, you may find that they do not operate properly.

The following items can be included on the command line:

d:\path\program: The path and name of the executable program file (*TCMD.EXE* or *4NT.EXE*). It is required to start the command processor.

d:\path: This is the second **d:\path** in the command line above. It sets the drive and directory where the program is stored, called the **COMSPEC** path. This option is included for compatibility with character-mode command processors, but is not needed in normal use. 4NT and Take Command can find their own directory without a COMSPEC path, and usually the COMSPEC variable should be left pointing to the default character mode command processor in use on your system.

@d:\path\inifile: This option sets the path and name of the [.INI file](#)⁸⁹. You don't need this option if

1) your *.INI* file is named *TCMD32.INI* (**TC**) or *4NT.INI* (**4NT**),

and

2) it is either in the same directory as the command processor, or in the Windows directory (**TC**) or in the root directory of the boot drive (**4NT**).

This option is most useful if you want to start the program with a specific and unique *.INI* file. To get around a Windows limitation that causes the displayed command line of a shortcut to be truncated when an argument begins with "@", you can use the alternative syntax */@d:\path\inifile* with a leading sacrificial slash.

//iniline: This option tells the command processor to treat the text appearing between the **//** and the next space or tab as an *.INI* directive. The directive should be in the same format as a line in the *.INI* file, but may not contain spaces, tabs, or comments. Directives on the command line

override any corresponding directive in the `.INI` file. This option may be repeated. It is a convenient way to place a few simple directives on the startup line without having to modify or create a new `.INI` file.

/L, /LA, /LD, /LF, and /LH: These options force the command processor to use a local alias, function, directory history, and / or command history list. They can be used to override any [LocalAlias=No](#)^[96], [LocalFunctions=No](#)^[96], [LocalHistory=No](#)^[96], or [LocalDirHistory=No](#)^[96] settings in the `.INI` file. This allows you to use global lists as the default, but start a specific shell or session with local aliases, functions or histories. See the topics [ALIAS](#)^[138], [FUNCTION](#)^[218], and [Local and Global History Lists](#)^[16] for more details. **/LA** forces local aliases, **/LD** forces local directory history, **/LF** forces local functions, **/LH** forces local command history, and **/L** forces all four.

/Q: (4NT) This option has no effect. It is included only for compatibility with `CMD.EXE`.

/S: (4NT) This option tells 4NT that you do not want it to set up a Ctrl-C / Ctrl-Break handler. It is included for compatibility with `CMD.EXE`, but it may cause the system to operate incorrectly if you use this option without other software to handle Ctrl-C and Ctrl-Break. This option should be avoided by most users.

/T:bf: This option sets the foreground and background colors in the command processor command window. Both **b** and **f** are hexadecimal digits; **b** specifies the background color and **f** specifies the foreground color. This option is included only for compatibility with Windows NT's `CMD.EXE`; in most cases you should set default colors with the [StdColors](#)^[114] directive in the [.INI file](#)^[89], or the corresponding **Output Colors** option on the [Colors tab](#)^[85] of the [configuration dialogs](#)^[82]. If you use both, the **/T** switch overrides any StdColors setting.

/U: This option causes the output of internal commands to a pipe or redirected to a file to be in Unicode ([Unicode versions](#)^[422] only). Also see the [UnicodeOutput](#)^[97] directive.

/X: This option forces 4NT and Take Command to alter the operation of the MD and MKDIR command to automatically create all necessary intermediate directories when it creates a new subdirectory. Its effect is the same as adding a /S option to all [MD and MKDIR](#)^[243] commands. This option is included for compatibility with `CMD.EXE`, where it also enables other options. However, in 4NT and Take Command those options are already enabled by default.

/C | /K command: This option tells the command processor to run a specific command after starting. The command will be run after [TCSTART](#)^[6] or [4START](#)^[6], and before the prompt is displayed. The command can be any valid alias, internal or external command, or batch file. All other startup options must be placed before the command, because the command processor will treat characters after the command as part of the command and not as additional startup options.

When the command is preceded by a **/C**, the command processor will execute the command and then exit, returning to the parent program or the desktop without displaying a prompt. The **/K** switch has no effect; using it is the same as placing the command (without a **/C** or **/K**) at the end of the startup command line. It is included only for compatibility with `CMD.EXE`.

For example, the command line

```
c:\4NT\4NT.exe c:\4NT\start.btm
```

will:

1. start `c:\4NT\4NT.exe`;
2. initializes 4NT from `c:\4NT\4NT.INI` if it exists;
3. if `c:\4NT\4NT.INI` exists and it contains the directive `4StartPath=c:\start` and a `4START` program is found in directory `c:\start` it is executed by 4NT, ELSE if

`c:\4NT\4NT.INI` does not exist, or it exists but contains no `4StartPath` directive, or it exists and contains the directive `4StartPath=c:\4NT`, and a `4START` program is found in directory `c:\4NT`, it is executed by 4NT;

4. 4NT executes `c:\4NT\start.btm`;

5. Unless an `EXIT` command is executed in `c:\4NT\start.btm`, 4NT displays the command prompt.

The command line below, when executed by 4NT, TC, CMD.EXE, the RUN dialog, or a shortcut, will start 4NT, select local aliases, execute any `4START` file you have created, execute the file `PROCESS.BTM`, and exit when `PROCESS.BTM` is done. No prompt will be displayed by this session:

```
c:\4NT\4NT.exe /la /c c:\4NT\process.btm
```

Note: For the sake of "compatibility" with some CMD.EXE versions, 4NT and Take Command support a `"/V"` option. Since CMD, unlike 4NT/TC, doesn't support delayed expansion of variable references in the `"%varname%"` format, it introduced a special `"!varname!"` notation. Using `"/V"` simply allows 4NT/TC to handle that syntax as an alternative to `"%varname%"` or `"%varname"` but has no other benefit.

2.1.2 Automatic Startup and Termination Programs

Each command processor supports two programs, which run automatically without your intervention, as long as the command processor can find them, executed when the command processor is started and terminated, respectively.

Command Processor Startup Program

Each time the command processor starts, it looks for a program named `4START` (4NT) or `TCSTART` (Take Command). `4START` / `TCSTART` is normally a batch file (`.BAT`, `.BTM`, or `.CMD`), but it can be any executable file. The program must be in the directory specified in the `.INI file` directive `4StartPath` (4NT) or `TCStartPath` (TC) if the directive is used. If the directive is not used, the `4START` or `TCSTART` program, if any, in the same directory as your command processor is executed. Use of `4START` / `TCSTART` is optional, and the command processor will not display an error message if it cannot find the program. If you do not want to use a startup program, either set the start path directive to a directory which does not have one, OR leave it unspecified, and make sure that no matching executable file is in the command processor's directory.

`4START` / `TCSTART` is a convenient place to change the color or content of the prompt for each session, `LOG` the start of a session, or execute other special startup or configuration commands. It is also one way to set aliases, functions, and environment variables. See the section below on Pipes etc. about changing directories via `4START` / `TCSTART`.

PARAMETERS. With the exception of some initialization switches, the entire startup command line passed to the command processor is available to `4START` / `TCSTART` as batch file parameters (`%1`, `%2`, etc.). This can be useful if you want to see the command line passed to the command processor. For example, to pause if any parameters are passed, you could include this command in `4START` / `TCSTART`:

```
if %# GT 0 pause Starting %_cmdproc with parameters [%$]
```

(assuming the command processor's ParameterChar is "\$", the default for 4NT / TC).

Note: 4NT and Take Command do not look for or execute `AUTOEXEC.BAT` but the NT family of operating systems might attempt to parse `AUTOEXEC.BAT` for `SET` statements at system startup.

See your Windows documentation for details on that "feature".

Pipes, Transient Sessions/Processes, and 4START / TCSTART

When you set up the 4START / TCSTART program, remember that it is executed every time the command processor starts, *including* when running a [pipe](#)^[64] or when a transient copy of the command processor is started with the /C [startup option](#)^[4]. 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 your 4START / TCSTART program *changes to a different directory*, the output file will be written there — not in C:\DATA.

This is because the command processor you use (either 4NT and Take Command) starts a second copy (instance) of itself to run the commands on the right hand side of the pipe, and that new copy executes 4START / TCSTART before processing the commands from the pipe.

The same thing occurs if you use a transient session (one started with /C option) to run an individual command, then exit — the session will execute in the directory set by 4START / TCSTART, not the directory in which it was originally started (e.g., by specifying a working directory in a shortcut). For example, suppose you set up a desktop object with a command line like this, which starts a transient session:

```
Command:          d:\tc32\tcmd.exe /c list myfile.txt
Working Directory: c:\data
```

Normally this shortcut 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, aliases, or other settings in 4START / TCSTART will affect all copies of the command processor, including those used for pipes and transient sessions.

You can work around these potential problems with the [IF](#)^[227] or [IFF](#)^[228] commands and the [PIPE](#)^[354] and [TRANSIENT](#)^[356] internal variables. For example, to skip all 4START / TCSTART processing when running in a pipe or in a transient session, you could use a command like this at the beginning of 4START / TCSTART:

```
if %_pipe != 0 .or. %_transient != 0 quit
```

Command Processor Termination Program

Whenever a 4NT or Take Command session ends, it looks for a program named 4EXIT (4NT) or TCEXIT (Take Command). 4EXIT / TCEXIT is normally a batch file (.BAT, .BTM, or .CMD), but it can be any executable file. The location of this optional program is determined by the same rule as the location of the 4START / TCSTART program for the session, and is not necessary in most circumstances. However, it is a convenient place to put commands to save information from one session to another, such as a (command) history list before the command processor exit, or to [LOG](#)^[242] the end of the session. *You can use a termination program even if you have no startup program.*

PARAMETERS. No parameters are passed to the termination program.

2.1.3 Exit Codes

Exit Codes

If you start the command processor from another program (e.g. to run a batch file or internal command), it will return a numeric code to the other program when it is finished. This code is usually used to indicate whether the operation performed was successful or not, with 0 often indicating success and a non-zero value indicating a failure or other numeric result.

The command processor's exit code is normally the numeric exit code from the last internal or external command. However, for compatibility reasons and to avoid conflicts with external commands, only some internal commands set the exit code; others leave it unchanged from the most recent external command.

If you enter an unknown command the exit code will be 2, which is the internal command processor unknown command error number.

You can also use the `EXIT` ^[205] `n` command to explicitly set the exit code. If you do, this will override the rules above, and set the return code to the value in your `EXIT` command.

The normal rules described above may not return a code that indicates the success or failure of the specific operation that concerns you. Therefore, if you need to rely on the exit code from 4NT or Take Command, we recommend that you use a batch file or alias to create the exit code you want, and then set the code explicitly with `EXIT n`.

2.1.4 Desktop Integration

This section outlines how to integrate 4NT and Take Command into the Windows desktop for easier and simpler access to the command processor features. Use these instructions if you are running a fairly standard Windows configuration. If you have altered your Windows desktop properties substantially, you may need to take those changes into account.

Note: *No unusual procedure is required to create references to the command processor or batch files, and you should use the very same methods you would use to refer to any other application or file type. If you need detailed instructions, please consult your Windows documentation.*

- ▶ [Creating Explorer Shortcuts to the Command Processor](#) ^[8]
- ▶ [Creating Explorer Shortcuts to Batch Files](#) ^[8]
- ▶ [Creating Explorer Context Menu Entries](#) ^[8]

Creating Explorer Shortcuts to the Command Processor

The command processor installation software typically creates a program group which appears on the **Start Menu** under **Programs**, and includes items which start the command processor or its online help. If you want to create additional shortcuts elsewhere on the **Start menu**, or modify the **Programs** entries, click the right mouse button (*right click*) in an open area of the Task Bar, and select *Properties* on the popup menu. Select the *Advanced* tab, then the *Start Menu Programs* tab to modify or adjust the menus as required, as you would for any other Start Menu entry.

You can also create one or more shortcuts on the **desktop** to run the command processor. To do so *right click* in any open area of the desktop. On the popup menu click *New*, then *Shortcut*. Fill in the drive and path for `d:\path\TCMD.EXE` or `d:\path\4NT.EXE` (use the appropriate drive and path for your system). Add any other command line options you wish to set to the end of the line.

No additional settings are necessary. The only required item is the drive and path for **TCMD.EXE** or **4NT.EXE**. However, you can put command-line switches, a command, or the name of a batch file at the end of the command line for any shortcut. This allows you to run specific commands or set configuration options when you start the command processor from that shortcut. See [Command Line Options](#)^[4] for details.

For example, it might be useful to keep **4START.BTM**^[6] very short, and instead rely on global lists ([Aliases](#)^[138], [Functions](#)^[218], etc.) after launching **SHRALIAS**^[290] only once when you first start your system. This can be easily accomplished by creating a batch file (**startup.btm**) that loads all your desired aliases, functions and environment variables, handles any housekeeping tasks you wish, then starts **SHRALIAS**, such as:

```
:: STARTUP.BTM - initializes 4NT
*unalias *
set /r d:\path\myvariables.txt
alias /r d:\path\myaliases.txt
function /r d:\path\myfunctions.txt
shralias
...
```

Once you have done so, merely add a shortcut to that batch file in your Windows "Startup" directory which invokes the command line:

```
d:\path\4nt.exe /c d:\path\startup.btm
```

That shortcut could be created with Explorer as outlined above, or with the [SHORTCUT](#)^[289] command:

```
SHORTCUT "d:\path\4nt.exe", "/c d:\path\startup.btm", "", "Initialize
4NT",
"%allusersprofile%\start menu\programs\startup\4NTinit", 3
```

(all on one line), or any equivalent method appropriate for your configuration.

Creating Explorer Shortcuts to Batch Files

By default, Windows assumes that ***.BAT** and ***.CMD** files are to be processed with **COMMAND.COM** (Win9x/ME) or **CMD.EXE** (NT/2000/XP), and it knows nothing about ***.BTM** files. This means that a shortcut directly pointing to a ***.BAT** or ***.CMD** file will invoke the "wrong" command processor and that a shortcut pointing directly to a ***.BTM** file will generate a "Not a valid Win32 application" or similar error.

To remedy that situation, you have several options, including:

- Create shortcuts that invoke the specific command processor you want, e.g.
`d:\path\4nt.exe /c mybatch.cmd`
- Associate filename extensions ***.BAT**, ***.CMD** and/or ***.BTM** with the desired command processor (see [Windows File Associations](#)^[461], [ASSOC](#)^[145] and [FTYPE](#)^[217])

If you don't want to disturb the default associations for ***.BAT** and ***.CMD** batch files, simply create a new association for ***.BTM**, and use that extension for all your own batch files. Also see the [Using .BAT Files Under 4NT](#)^[321] topic.

Creating Explorer Context Menu Entries

Microsoft's **explorer** offers the ability to define new entries that will appear in the *right click* menu for various desktop objects such as files, directories and drives. Double-clicking on an icon triggers the **default** action (typically the *open* action) for the file type represented by that icon, but you can create additional entries and/or change the default to a different existing entry.

While the [FTYPE](#)^[217] command can be used to change the command associated with the default action, more extensive changes will require the "File Types" explorer dialog (under the "Tools/Folder Options" menu) or direct modification of the *Win32 Registry*. Select the method with which you are most comfortable.

For example, you could keep the default "execute with CMD.EXE" action for batch files, but add a new **"Execute with 4NT"** entry that invokes `d:\path\4nt.exe /c %*` or a similar command. That option would then become available for any batch file by selecting its name from the *right click* menu.

Many users find it convenient to add a **"4NT Prompt Here"** option for directories (folders). Some Windows configurations already include a "DOS Prompt Here" which is typically hard-coded to point to COMMAND.COM or CMD.EXE but can be readily adjusted to point to the command processor of your choice. For example, an entry could be set to execute `"d:\path\4nt.exe *cdd %*" (start a new 4NT session in this directory)`

Similar entries can be created at will for any file type. You could add a "LIST" entry to text files, or "Update JPSTREE Database" to drives, etc.

As usual when dealing with the Registry, **USE CAUTION!**

2.2 The Command Line

The command processor displays a `[c:\]` prompt when it is waiting for you to enter a command. *The actual text depends on the current drive and directory as well as your [PROMPT](#)^[262] 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 explains the features that will help you while you are typing in commands, how keystrokes are interpreted when you enter them at the command line, and how to transfer text between the command processor and other applications.

The keystrokes discussed here are the ones normally used by the command processor. If you prefer using different keystrokes to perform these functions, you can assign new ones with [key mapping directives](#)^[114] in the [INI file](#)^[89].

Some of the command line features documented in this section are:

- ▶ [Command-Line Editing](#)^[17]
- ▶ [Command History and Recall](#)^[13]
- ▶ [Command History Window](#)^[15]
- ▶ [Command Names and Parameters](#)^[17]
- ▶ [Filename Completion](#)^[17]
- ▶ [Filename Completion Window](#)^[27]
- ▶ [Variable Completion](#)^[22]
- ▶ [Automatic Directory Changes](#)^[22]
- ▶ [Directory History Window](#)^[23]
- ▶ [Multiple Commands](#)^[24]
- ▶ [Expanding and Disabling Aliases](#)^[25]

- ▶ [Command-Line Length Limits](#)^[25]
- ▶ [Scrolling and History Keystrokes](#)^[25]
- ▶ [Command Grouping](#)^[27]
- ▶ [Starting Applications](#)^[28]
- ▶ [Command Parsing](#)^[29]

Additional command-line features are documented under [File Selection](#)^[36] and under [Directory Navigation](#)^[54].

2.2.1 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**^[117] to execute it, or **Esc**^[117] to erase it.

The command line as typed can contain up to a maximum of 8,191 characters but can expand to a maximum of 16,383 characters after variable and alias substitution. See [Command-Line Length Limits](#)^[25].

You can use the following editing keys (among others) 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). The keystrokes listed here are merely default values, but most editing keys can be redefined via [Command-Line Editing Keys](#)^[118] or [General Input Keys](#)^[115] directives.

Cursor Movement Keys:

Left ^[117]	Move the cursor left one character.
Right ^[118]	Move the cursor right one character.
Ctrl-Left ^[118]	Move the cursor left one word.
Ctrl-Right ^[118]	Move the cursor right one word.
Home ^[116]	Move the cursor to the beginning of the line.
End ^[117]	Move the cursor to the end of the line.

Insert and Delete Keys:

Ins ^[117]	Toggle between insert and overstrike mode (cursor shape indicates mode).
Del ^[116]	Delete the character under (or to the right of) the cursor, or the highlighted text.
Bksp ^[116]	Delete the character to the left of the cursor, or the highlighted text.
Ctrl-L ^[116]	Delete the word or partial word to the left of the cursor.
Ctrl-R ^[117] or Ctrl-Bksp ^[117]	Delete the word or partial word to the right of the cursor.
Ctrl-Home ^[116]	Delete from the beginning of the line to the cursor.
Ctrl-End ^[116]	Delete from the cursor to the end of the line.
Esc ^[117]	Delete the entire line.
Ctrl-V ^[116]	Paste the first line of text from the clipboard at the current cursor position.
Ctrl-C	(TC) Paste all of the text from the clipboard at the current cursor position.
Ctrl-Shift-Ins	(TC) Insert the highlighted text (from anywhere in the window) at the current cursor position of the command line.

Highlighting (4NT):

Shift-Right	Highlight character right of cursor and move cursor
Shift-Left	Highlight character left of cursor and move cursor
Shift-Home	Highlight from cursor to beginning-of-line and move cursor
Shift-End	Highlight from cursor to end-of-line and move cursor
Ctrl-Shift-Right	Highlight word right of cursor and move cursor
Ctrl-Shift-Left	Highlight word left of cursor and move cursor
Ctrl-Y ^[116]	Copy highlighted text to the clipboard

Execution:

Ctrl-K ^[127]	Save the current command line in the history list without executing it, and then clear the command line
Ctrl-C or Ctrl-Break	Cancel the command line without saving in the history list. (Also see the CUA ^[107] directive).
Enter ^[117]	Execute the command line.

Miscellaneous:

F1 ^[225]	Command help.
Ctrl-F1 ^[126]	Word help.
Ctrl-F ^[119]	Expand alias.
Ctrl-A ^[126]	LFN Toggle.
Alt-PgUp, Alt-PgDn, Alt-Home, Alt-End, Alt-Up, Alt-Down	(4NT) Scroll the window within the console buffer. Use the cursor pad keys, not the numeric keypad keys.

To highlight text on the command line use the mouse **(TC)**, or hold down the **Shift** key and use any of the cursor movement keys listed above. You can select a complete word by placing the cursor

anywhere in the word and double-clicking with the mouse. 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 [Highlighting and Copying Text](#)^[76] for additional details). You can also use [Drag and Drop](#)^[80] to paste a filename from another application onto the command line.

Most of the command-line editing capabilities are also available when the command processor prompts you for a line of input. For example, you can use the command-line editing keys when [DESCRIBE](#)^[176] prompts for a file description, when [INPUT](#)^[233] prompts for input from an alias or batch file, or when [LIST](#)^[237] prompts you 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](#)^[85] of the [configuration dialogs](#)^[82], or the [InputColors](#)^[113] directive in the [.INI file](#)^[89].

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](#)^[103] is the caret "**^**", but in general, it's best to use the symbolic "**%="**" [EscapeChar](#)^[103] representation for portability. For example:

```
[c:\] echo The quick brown fox jumped over the lazy %=
More? 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](#)^[119]. You enter **Alt-255** by holding down the **Alt** key while you type **0255** on the numeric keypad, then releasing the **Alt** key. 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](#)^[114] in the [.INI file](#)^[89], and **Alt-255** itself can be reassigned with the [CommandEscape](#)^[119] directive.

Alternative Keyboard Input Method:

The method mentioned above for "Alt-255" can be used to generate other characters. *You must use the number keys on the numeric keypad, not the row of keys at the top of your keyboard.* When this "**Alt + keypad**" approach is used in a Unicode environment, the command processor will assume that a 3-digit decimal value means an ASCII character, while a 4-digit decimal value mean a Unicode glyph. Make sure that your hardware, character set, code page and font all support the desired combination. Use caution with this method if you plan on manipulating the generated character in other Windows components. See the section on [ASCII, Key Codes and ANSI X3.64 Commands](#)^[445] for some additional information.

2.2.2 Command History & 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](#)^[15]. You can choose to use either a [local or global command history](#)^[16].

Command History Keys:

Up (4NT) or Ctrl-Up (TC)	Recall the previous (or most recent) command, or the most recent command that matches a partial command line.
Down (4NT) or Ctrl-Down (TC)	Recall the next (or oldest) command, or the oldest command that matches a partial command line.
F3	Fill in the rest of the command line from the previous command, beginning at the current cursor position.
Ctrl-D	Delete the currently displayed history list entry, erase the command line, and display the previous (matching) history list entry.
Ctrl-E	Display the last entry in the history list.
Ctrl-K	Save the current command line in the history list without executing it, and then clear the command line.
Ctrl-Enter	Copy the current command line to the end of the history list even if it has not been altered, then execute it.
@	As the first character in a line: Do not save the current line in the history list when it is executed, nor store it in the CMDLINE ^[339] environment variable.

Note: The keystrokes shown above are the default values. See [Key Mapping Directives](#) ^[114] for details on how to assign different keystrokes.

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 **Up (4NT)** or **Ctrl-Up (TC)** 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](#) ^[137].

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. You can also view and manage the command history list with the [HISTORY](#) ^[226] command.

The history list is normally "circular". If you move to the last command in the list and then press **Up (4NT)** or **Ctrl-Up (TC)** once more, you'll see the first command in the list. Similarly, if you move to the first command in the list and then press **Up (4NT)** or **Ctrl-Up (TC)** once more, 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](#) ^[106] to No in the [.INI file](#) ^[89].

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 the **Up (4NT)** or **Ctrl-Up (TC)**. You only need to enter enough characters to identify the command that you want to find. For example, to find a DIR command, enter DI and then press the **Up (4NT)** or **Ctrl-Up (TC)**. If you press **Up (4NT)** or **Ctrl-Up (TC)** 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 **Up (4NT)** or **Ctrl-Up (TC)** again.

You can specify the size of the command history list on the "History" tab of the configuration dialog, or with the [History](#) ^[95] directive in the [.INI file](#) ^[89]. When the list is full, the oldest commands are discarded to make room for new ones. You can also use the [HistMin](#) ^[105] directive in the [INI file](#) ^[89] to enable or disable history saves and to specify the shortest command line that will be saved.

You can prevent any command line from being saved in the history by beginning it with an at sign [**@**] or by including in the contents of the [HistoryExclude](#)^[34] variable.

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](#)^[105] or [HistMove](#)^[105] to Yes in your [.INI file](#)^[89] or select Copy to End or Move to End on the "History" tab of the configuration dialogs. If you select either of these options, the list entry identified as "current" (the entry from which commands are retrieved when you press **Ctrl-Up**) is also adjusted to refer to the end of the history list after each recalled command is executed.

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 **Up (4NT)** or **Ctrl-Up (TC)**) 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 **Up (4NT)** or **Ctrl-Up (TC)** 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.

(TC) If you prefer to use the arrow keys to access the command history in (as in 4NT), without having to press **Ctrl**, see the [SwapScrollKeys](#)^[117] directive, or the corresponding option on the [History tab](#)^[85] of the configuration dialog. SwapScrollKeys switches the Take Command keystroke mapping so that the **Up**, **Down**, **PgDn** and **PgUp** keys manipulate the command history, and **Ctrl-Up**, **Ctrl-Down**, **Ctrl-PgDn** and **Ctrl-PgUp** are used to control the scrollbar buffer. For more details see [Scrolling and History Keystrokes](#)^[28].

2.2.3 Command History Window

You can view the command history in a scrollable command history window, and select the command to re-execute or modify from those displayed in the window.

Command History Window Keys:

Up ^[118]	Scroll the display up one line.
Down ^[117]	Scroll the display down one line.
Left ^[117]	Scroll the display left 4 columns.
Right ^[118]	Scroll the display right 4 columns.
PgUp ^[122]	Scroll the display up one page.
PgDn ^[122]	Scroll the display down one page.
Ctrl-PgUp ^[123] or Home ^[123]	Go to the beginning of the list.
Ctrl-PgDn ^[123] or End ^[123]	Go to the end of the list.
Ctrl-Enter ^[123]	Move the selected line to the command line for editing

Enter ^[123]	Execute the selected line
Ctrl-D ^[123]	Delete the selected line from the list
Esc ^[117]	Close the window without making a selection.

Note: The keystrokes shown above are the default values. See [Key Mapping Directives](#)^[114] for details on how to assign different keystrokes.

To activate the command history window press **PgUp** or **PgDn (4NT)** or **Ctrl-PgUp** or **Ctrl-PgDn (TC)** at the command line. A window will appear in the upper right corner of the screen, 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.)

Once you have selected a command in the history window, press **Enter** or double-click with the mouse to execute it immediately. Press **Ctrl-Enter** or hold down the Ctrl key while you double-click with the mouse to move the line 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 (4NT)** or **Ctrl-PgUp** or **Ctrl-PgDn (TC)**. Only those commands matching the typed characters will be displayed in the window.

You can control the position and size of the history window with [configuration directives](#)^[97] in the [.INI file](#)^[89] or the corresponding items on the [History tab](#)^[85] of the [configuration dialogs](#)^[82]. You can also change the keys used in the window with [key mapping directives](#)^[114] in the [INI file](#)^[89].

(**TC**) If you prefer to use the PgUp key to access the command history without having to press **Ctrl** (as in 4NT), see the [SwapScrollKeys](#)^[117] directive in *TCMD32.INI*, or the corresponding option on the [History tab](#)^[85] of the [configuration dialogs](#)^[82]. SwapScrollKeys switches the keystroke mapping so that the **Down**, **Up**, and **PgUp** keys manipulate the command history, and **Ctrl-Up**, **Ctrl-Down**, **Ctrl-PgUp**, and **Ctrl-PgDn** are used to control the scrollbar. For more details see [Scrolling and History Keystrokes](#)^[25].

2.2.4 Local & Global History Lists

The [command history](#)^[226] and [directory history](#)^[189] can be stored in either "local" or "global" lists.

With a local list, any changes made to the history will only affect the current copy of the command processor. They will not be visible in other shells, or other sessions.

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

You can control the type of history list with the [LocalHistory](#)^[96] and [LocalDirHistory](#)^[96] directives in the [.INI file](#)^[89], and with the **/L**, **/LD** and **/LH** options of the [START](#)^[291] command.

If you select a global history list for the command processor, you can share the history among all copies of the command processor running concurrently. When you close all command processor sessions, the memory for the global history list is released, and a new, empty history list is created the next time you start the command processor.

If you want the histories to be retained in memory even when no command processor session is running, see the [SHRALIAS](#)^[290] command, which retains the global alias, user-defined functions, command history, and directory history lists. SHRALIAS retains lists in memory, but cannot preserve it when Windows itself is shut down. To save your histories when restarting Windows, you must store them in a file and reload them after the system restarts. For details on how to do so, see the [HISTORY](#)^[226] and [DIRHISTORY](#)^[189] commands

2.2.5 Command Names & 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\4NT\4NT.exe" /LF
```

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 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 the [UnixPaths](#)^[112] directive to Yes in the [.INI file](#)^[89].

For more information on command entry see [Multiple Commands](#)^[24] and [Command Line Length Limits](#)^[25]. For details on how the command processor handles the various elements it finds on the command line see [Command Parsing](#)^[29].

2.2.6 Filename Completion

V6.01 ENHANCEMENTS NOT YET INCLUDED.

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 it. Filename completion can be used at the command line, which is explained here, and in a [filename completion window](#)^[21].

Filename Completion Keys:

F8 or Shift-Tab	Get the previous matching filename.
F9 or Tab	Get the next matching filename.
F10 (4NT)	Keep the current matching filename and display the next matching name immediately after the current one.
Ctrl-Shift-Tab (TC) or F11 (4NT)	Keep the current matching filename and display the next matching name immediately after the current one.
F12	Repeat the filename just returned from an F9 / Tab match.
Ctrl-A	Toggle between long and short filename.

Note: The keystrokes shown above are the default values. See [Key Mapping Directives](#)^[114] for details on how to assign different keystrokes.

For example, if you know the name of a file 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 match your pattern (in the above example, begins with *AU*). When there are no more filenames that match your pattern, the system 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**.

If you want to enter more than one matching filename on the same command line, press **Ctrl-Shift-Tab** or **F11 (TC)** or **F10 (4NT)** when each desired name appears. This will keep that name and place the next matching filename after it on the command line. You can then use **Tab** (or **F9**) and **Shift-Tab** (or **F8**) to move through the remaining matching files.

The pattern you use for matching may contain any valid filename characters, as well as wildcard characters and extended [wildcards](#)^[36]. 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, and *.* to the name on drives which only support short file names. It will also place a "*" after a partial extension. If you are typing a group of file names in an [include list](#)^[46], 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, and files with [executable extensions](#)^[48], 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](#)^[22]. 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. *Note* that you can also "execute" files whose extension has an association in the Windows Registry, but such files are not considered executable by the command processor, and only the method above using a **space** will work.

Filename completion occurs in the physical order in which matching filenames are stored in the directory, the same order in which `DIR /OU` would list them. That order is determined by the underlying file system.

The command processor also supports 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](#)^[109] directive in the [.INI file](#)^[89].

Filename completion will search the [PATH](#)^[254] for an executable filename if you are (1) at the beginning of the command line, (2) there

are no matching entries in the current directory, **and (3)** the name you are attempting to match doesn't contain a full or partial path specification. If all three conditions are met, filename completion will return the first matching executable found in the PATH.

Several topics are related to filename completion. See:

- ▶ [Converting Between Long and Short Filenames](#)^[20]
- ▶ [Appending Backslashes to Directory Names](#)^[19]
- ▶ [Customizing Filename Completion](#)^[19]
- ▶ [Filename Completion Window](#)^[21]
- ▶ [Variable Name Completion](#)^[22]

2.2.7 Appending Backslashes to Directory Names

If you set the [AppendToDir](#)^[99] directive in the [INI file](#)^[89], or the corresponding option in the configuration dialogs, the command processor will add a trailing backslash `[\]` to directory names. The character appended is a trailing slash `[/]` for directory names in [FTP URLs](#)^[52], or - *if you have set the directive [UnixPaths=yes](#)*^[112] - to all directory names.

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 (4NT)** or **Ctrl-Shift-Tab (TC)** 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\FINANCEMAPS.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.

2.2.8 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 [Editing tab](#)^[84] of the configuration dialogs, or set the [FileCompletion](#)^[103] directive manually in the [INI file](#)^[89]. You can also use the [FILECOMPLETION](#)^[340] environment variable. If you use both, the environment variable will override the settings made in the dialog or the [INI file](#)^[89]. 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 directive 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
RDONLY	Read-only files
HIDDEN	Hidden files
SYSTEM	System files
ARCHIVE	Files modified since the last backup
FILES	Everything that's not a directory

Note that if a file uses one of the reserved file type names shown above as its extension (e.g. "foo.hidden") , that file will be treated as if it were of that type.

The command name is the internal command, alias command, 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 in the configuration dialog:

```
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 pressed **Tab**, the command processor returns 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, the command processor checks 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.
```

2.2.9 Converting Between Long & Short Filenames

On LFN drives, the command processor will search for and display long filenames during filename completion. If you want to search for 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. The **LFNToggle** directive can be used to change the default "Ctrl-A" keystroke assigned to this feature.

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 returns 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.

2.2.10 Filename Completion Window

You can 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 [Popup Windows](#)^[464] for details.

Filename Completion Window Keys:

F7 ^[120] or Ctrl-Tab ^[120]	(from the command line) Open the window.
Up ^[118]	Scroll the display up one line.
Down ^[117]	Scroll the display down one line.
Left ^[117]	Scroll the display left 4 columns.
Right ^[118]	Scroll the display right 4 columns.
PgUp ^[122]	Scroll the display up one page.
PgDn ^[122]	Scroll the display down one page.
Ctrl-PgUp ^[123] or Home ^[123]	Go to the beginning of the list.
Ctrl-PgDn ^[123] or End ^[123]	Go to the end of the list.
Enter ^[123] or Double Click	Insert the selected filename into the command line.

Note: The keystrokes shown above are the default values. See [Key Mapping Directives](#)^[114] for details on how to assign different keystrokes.

Also see [Filename Completion](#)^[17]

2.2.11 Escape Character

The command processor recognizes 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 X3.64 and printer control sequences.

The default escape character is a caret (^, ASCII: 94). If you don't like using the default escape character, you can pick another character using the [SETDOS](#)^[283] /E command, the configuration dialogs, or the [EscapeChar](#)^[103] directive in [.INI file](#)^[89]. If you plan to share aliases or batch files between several 4NT and Take Command configurations, use the [%=](#)^[347] pseudovisible, which is accepted in all of them, regardless of the actual value assigned to the escape character. See the section on [Special Character Compatibility](#)^[335] for details about choosing compatible escape characters for two or more products. Note that if you change the default, your batch file is unlikely to run as intended under CMD.EXE.

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](#)^[61] codes to the printer. The special characters which can follow the escape character are:

b backspace
c comma
e the ASCII `ESC` character (code 27)
f form feed
k back quote ```
n line feed
q quote mark `"`
r carriage return
s space
t tab character

If you follow the escape character with any other character, the escape character is removed and the second character is copied directly into 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 **ESCY** to the printer, you can use this command:

```
[c:\] echos %=f%=eY > prn
```

The escape character has an additional use when it is the last character on any line of batch file. The command processor recognizes this use of the escape character to signal **line continuation**: it removes the escape character and appends the next line to the current line before executing it.

2.2.12 Variable Completion

Variable name completion works like [filename completion](#).^[17] If the argument 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, and no other variables start with `p`, the sequence below may be used to display the `PATH`:

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

2.2.13 Automatic Directory Changes

Automatic directory changes are part of a set of comprehensive directory navigation features built into the command processor. For a summary of these features, and more information on the Extended Directory Searches and `CDPATH` features mentioned below, see the [Directory Navigation](#)^[54] section.

The automatic directory change feature lets you change directories quickly from the command prompt, without entering an explicit `CD`^[159] or `CDD`^[160] command. To do so, simply type the name of the directory you want to change to at the prompt, with a backslash `[\]` (either added manually, or automatically via the [AppendToDir](#)^[99] directive) at the end. For example:

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

This can make directory changes very simple when it's combined with [Extended Directory Searches](#)^[56] or [CDPATH](#)^[59]. 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](#)^[54] 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:\tcmd] 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 a UNC name (see [File Systems](#)^[439] for details on UNC names). Commands like "...\" can be used to move up the directory tree quickly (see [Extended Parent Directory Names](#)^[47]). 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\scanner\  
[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\SCANNER* subdirectory of the directory which is two levels "up" from the current directory in the tree.

2.2.14 Directory History Window

[The directory history window is part of a set of comprehensive directory navigation features built into the command processor. For a summary of these features, and more information on enhanced directory navigation features, see [Directory Navigation](#)^[54].]

Directory History Window Keys:

F6 ^[122]	Open the window from the command line (Take Command)
Ctrl-PgUp ^[122]	Open the window from the command line (4NT)
Up ^[118]	Scroll the display up one line.
Down ^[117]	Scroll the display down one line.
Left ^[117]	Scroll the display left 4 columns.
Right ^[118]	Scroll the display right 4 columns.
PgUp ^[122]	Scroll the display up one page.
PgDn ^[122]	Scroll the display down one page.
Ctrl-PgUp ^[123] or Home ^[123]	Go to the beginning of the list.
Ctrl-PgDn ^[123] or End ^[123]	Go to the end of the list.
Ctrl-Enter ^[123]	Move the selected line to the command line for editing
Enter ^[123]	Change to the selected drive/directory
Ctrl-D ^[123]	Delete the selected line from the list
Esc ^[117]	Close the window without making a selection.

Note: The keystrokes shown above are the default values. See [Key Mapping Directives](#)^[114] for details on how to assign different keystrokes.

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 the scrollable directory history window and change to any drive and directory on the list. To activate the directory history window, press **F6** at the command line. You can then select a new directory with the **Enter** key or by double-clicking with the mouse.

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 with the **DirHistory** directive in the **.INI file** or with the corresponding items on the **History tab** of the **configuration dialogs**. You can change the keys used in the window with **key mapping directives** in the **.INI file**.

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 below 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 the command processor.

You can also view and manage the directory history list with the **DIRHISTORY** command.

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 sessions.

Whenever you start a secondary shell 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 directory history 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 dialogs, with the **LocalDirHistory** directive in the **.INI file**, with the **/L** and **/LD** **command line options**, and with the **/L** and **/LD** options of the **START** command.

When you close all command processor sessions, the memory for the global directory history list is released, and a new, empty directory history list is created the next time you start the command processor. Under 4NT and Take Command, if you want the directory history list to be retained in memory even when no copy of the command processor is running, you need to execute the **SHRALIAS** command, which performs this service for the global command history, directory history, user-defined functions, and alias lists.

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 sessions or shells. We recommend that you start with a global directory history, then modify it if you find a situation where the default is not convenient.

2.2.15 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 an ampersand [&] or the "%+" pseudovisible reference. 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, 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 8,191 characters.

You can use multiple commands in [alias](#)^[138] definitions and [batch files](#)^[321] 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](#)^[283] /C command, the [CommandSep](#)^[100] directive in the [.INI file](#)^[89] or on the [Syntax tab](#)^[86] of the [configuration dialogs](#)^[82].

2.2.16 Command-Line Length Limits

When you first enter a command at the command prompt, in an alias or function definition, or in a batch file, it can be up to 8,191 characters long.

As the command processor scans the command line and substitutes the contents of aliases user defined functions, and environment variables for their names, the line usually gets longer. This expanded line is limited to 16,383 characters. If your use of aliases, user defined functions, or environment variables causes the command line to exceed the applicable one 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.

2.2.17 Expanding & 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 [Aliases](#)^[318] and the [ALIAS](#)^[138] command 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
```

Note: The leading asterisk is crucial in aliases that redefine existing commands, such as:

```
DIR=*dir /w
```

Without the asterisk, you would trigger an *alias loop error* whenever you try to use that alias since it would endlessly try to redefine itself.

2.2.18 Scrolling & History Keystrokes

In order to support the [scrollback buffer](#)^[75], some Take Command keystroke defaults are different from what you may be used to in 4NT. The differences are:

Command Line	4NT	Take Command
<i>Previous command</i>	Up	Ctrl-Up
<i>Next command</i>	Down	Ctrl-Down
<i>History window</i>	PgUp	Ctrl-PgUp
<i>Directory history</i>	Ctrl-PgUp, F6	F6

Screen Scrollback	4NT	Take Command
<i>Up one line</i>	Alt-Up	Up
<i>Down one line</i>	Alt-Dn	Down
<i>Up one page</i>	Alt-PgUp	PgUp
<i>Down one page</i>	Alt-PgDn	PgDn

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 4NT), set the [SwapScrollKeys](#)^[111]=yes directive in [TCMD32.INI](#)^[89], or the corresponding option on the [History tab](#)^[85] of the [configuration dialogs](#)^[82]. SwapScrollKeys switches the keystroke mapping so that the **Up**, **Down**, **PgUp** and **PgDn** keys manipulate the command history, and **Ctrl-Up**, **Ctrl-Down**, **Ctrl-PgUp**, and **Ctrl-PgDn** are used to control the scrollback buffer. [SwapScrollKeys](#)^[111] 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](#)^[114] in [TCMD32.INI](#)^[89]. The directives associated with the history and scrolling keys are:

DirWinOpen ^[122]	HistWinOpen ^[122]
NextHistory ^[120]	PrevHistory ^[120]
ScrollDown ^[120]	ScrollPgDn ^[122]
ScrollPgUp ^[122]	ScrollUp ^[120]

2.2.19 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 succeeds:

```
[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 fails, 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.

2.2.20 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**^[270] commands in all subdirectories of your hard disk. You could do it like this:

```
[c:\] global ren *.wxl *.wxo
[c:\] global ren *.txl *.txo
```

But with command grouping you can do the same thing in one command:

```
[c:\] global (ren *.wxl *.wxo %+ ren *.txl *.txo)
```

The two REN commands enclosed in the parentheses appear to **GLOBAL**^[220] as if they were a single command, so both commands are executed for every directory, but the directories are only scanned once, not twice.

This kind of command grouping is most useful with the **EXCEPT**^[204], **FOR**^[210], **GLOBAL**^[220], and **IF**^[227] 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 (1 line and 4 lines) will work properly, but the third (2 lines) 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)
```

If the above examples were typed at the command line, then the command processor would issue a **"More?"** prompt in response to each line until the command group is closed (i.e. the final parenthesis is recognized) as discussed below.

The second common use of command grouping is to redirect input or output for several commands without repeatedly using the **redirection**^[61] 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**^[198] 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:

```
(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](#)^[67] and [piping](#)^[64].

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:\]
```

Limitations

A group of commands in parentheses is like a long command line. The total length of the group may not exceed 8,191 characters, 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.

Each line you type at the normal prompt or the **More?** prompt, and each individual command within the line, must be within the usual [command line length limit](#)^[25].

2.2.21 Starting Applications

The command processor 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.

The command processor offers two methods to simplify and speed up access to your applications. One is to create an [alias](#)^[138], 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.

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

[executable extensions](#)^[48] or [Windows file associations](#)^[46].

For additional flexibility, you can also start applications with the [START](#)^[29] command. START provides a number of switches to customize the way an application is started.

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. The command processor also searches the Windows and Windows system directories; see the [PATH](#)^[25] command 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, the command processor 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. That search order may be altered (CAREFULLY!) via the advanced [PathExt](#)^[108] directive. If no such file is found, the command processor will move on to the next directory in 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 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. Under Windows 98 and ME, if you execute a .PIF file, the application will be opened in its own window. Applications run under [Take Command's Caveman](#)^[79] feature will run in the console window, but their output will be displayed in the Take Command window as well. For additional details, see [Console Applications and the Console Window](#)^[78].

2.2.22 Waiting for Applications to Finish

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 the command processor to wait for applications to finish before continuing by selecting the "Wait for External Apps" checkbox on the [Startup tab](#)^[82] of the configuration dialog, with the [ExecWait](#)^[103] directive in the [.INI file](#)^[89], or with the [START](#)^[29] command's **WAIT** switch (**START** can also control many other aspects of how your applications are started).

Regardless of the [ExecWait](#)^[103] setting, the command processor always waits for applications that are run from transient shells (with a **/C**), or from batch files before continuing with subsequent commands in the batch file. To start an application from a transient shell or a batch file and continue without waiting for the application to finish, use the [START](#)^[29] command (without the **WAIT** switch).

Due to the way Windows handles URLs, you cannot wait for the browser software to finish when you enter an "http:" URL at the prompt; in this situation, the command processor always displays the next prompt immediately.

2.2.23 Command Parsing

Whenever you type something at the command line and press the [Enter](#)^[11] 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](#)^[241]) to the history log file if history logging has been enabled with the [LOG /H](#)^[242] command and the command did not come from a batch file. Then, if the line contains multiple commands, the first command is isolated for processing. The following steps outline the basic processing required for each command. During that processing, additional parsing tasks may be triggered as noted and some steps may be repeated multiple times.

1. Separating the command from its command tail

The command processor begins by dividing the command into a command name and a command tail. The command name is the first word in the command, and 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". In some instances, the parser will be able to understand incorrect syntax such as "dir/w", but there should always be at least one space between the command name and its parameters.

2. Expanding aliases

Next, the command processor tries to match the command name against its list of [aliases](#)^[318]. 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](#)^[119] 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.

3. Expanding variables

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 (see "[Environment: Variables and Functions](#)"^[336]). This process is called variable expansion.

The variable expansion process is modified for certain internal commands, such as [EXCEPT](#)^[204], [IF](#)^[227], and [GLOBAL](#)^[220]. These commands are always followed by another command, so variable expansion takes place separately for the original command and the command that follows it.

4. Identifying an internal command

Once it has finished variable expansion, the command processor next tries to match the resulting command name with its list of [internal commands](#)^[127]. If it is unsuccessful, it knows that it will have to search for a batch file or external program to execute your command.

5. Displaying the command

When 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](#)^[106] (if command [logging](#)^[242] has been turned on).

6. Processing redirection and piping

Before it can actually execute your command, the command processor must scan the command tail to see if it includes [redirection](#)^[61] or [piping](#)^[64]. If so, the proper internal switches are set to send output to an alternate device or to a file instead of to the screen. A second process is started at this point, if necessary, to receive any piped output.

7. Processing escape characters

At this stage, remaining [Escape Characters](#)^[21] are processed. However, this might also already have taken place inside some of the variable functions (such as [@IF](#)^[390]) that are likely to pass escaped strings in their arguments. If you are referencing one of those in an [ECHO](#)^[198] or similar command, you need to escape twice ("^^") or use [SETDOS /X](#)^[283] to avoid premature evaluation. Carefully test those tricky situations to make sure the results are as you intended.

8. Executing the command

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 in [Executable Files and File Searches](#)^[458]).

9. Cleaning up

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.

Note: You can disable and re-enable several parts of command parsing (for example alias expansion, variable expansion, and redirection) with the [SETDOS /X](#)^[283] command.

2.3 Conditional Expressions

This topic is still under construction.

The commands [DO](#)^[191] (when used with the UNTIL or WHILE keyword), [IF](#)^[227], [IFF](#)^[228]/ELSEIFF, and the variable function [@IF](#)^[390] evaluate conditional expressions, and perform a different action based on whether or not the *expression* is TRUE. Most of the [examples](#)^[31] below use the [IF](#)^[227] command, but the conditional expressions could be used in the other cases above as well.

A conditional expression can be one of the following, as described below:

- ▶ [relational expression](#)^[31]
- ▶ [status test](#)^[31]
- ▶ [logical expression](#)^[31].

Relational Expressions

A relational expression compares two character strings, using one of the [relational operators](#)^[37] in the table below. Each of these two character strings can contain literal text, environment and internal variables, and variable functions, including user defined ones, in any combination. Note that quote marks are significant.

When comparing the two character strings, either a numeric or a string comparison will be used. A numeric comparison treats the strings as numeric values and tests them arithmetically. A string comparison treats the strings as text. The parser determines which kind of test to perform by examining the first character of each string *after expansion of variables and variable functions*. 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 quote marks around the values you are testing, as shown below. Because the quote mark is not a numeric character, string comparison is performed. Numeric comparison cannot be forced. To compare hexadecimal numbers, you must convert them to decimal numbers using [@CONVERT](#)^[36b].

The example below demonstrates the difference between numeric and string comparisons, as shown in the table below. 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:

expression	value	comparison type
<code>2 lt 19</code>	true	numeric
<code>"2" lt "19"</code>	false	string

The format of a relational expression is one of

`num1 relational operator[37] num2`
`string1 relational operator[37] string2`

Note: The correct syntax **requires** a space both before and after *operator* to separate it from its operands. Commonly seen constructs such as `%a==b` may or may not work depending on the specific parameters, but they are *never* recommended.

Relational operators

<u>operator</u>	numeric comparison: <i>expression</i> is true if	string comparison: <i>expression</i> is true if, when ignoring character case:
EQ or ==	<i>num1</i> equals <i>num2</i>	<i>string1</i> equals <i>string2</i> (case-insensitive)
NE or !=	<i>num1</i> does not equal <i>num2</i>	<i>string1</i> does not equal <i>string2</i>
LT	<i>num1</i> is less than <i>num2</i>	<i>string1</i> alphabetically precedes <i>string2</i>
LE	<i>num1</i> is less than or is equal to <i>num2</i>	<i>string1</i> alphabetically precedes or is equal to <i>string2</i>
GE	<i>num1</i> is greater than or is equal to <i>num2</i>	<i>string1</i> alphabetically succeeds or is equal to <i>string2</i>
GT	<i>num1</i> is greater than <i>num2</i>	<i>string1</i> alphabetically succeeds <i>string2</i>
EQC	tested as strings →	<i>string1</i> is identical to <i>string2</i> , including character case

Case differences are ignored in string comparisons (except by **EQC**). If two strings begin with the same text but one is shorter, the shorter string is considered to precede (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 quote marks 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 comparison, 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 leading sign (+ or -).

In order to maintain compatibility with *CMD.EXE*, the command processor recognizes the following additional names for conditions:

<i>CMD . EXE</i>	JPsoft command processors
EQL or EQU	EQ
NEQ	NE
LSS	LT
LEQ	LE
GTR	GT
GEQ	GE

[Internal variables](#)^[342] and [variable functions](#)^[357] 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 a condition test.

Status Tests

These conditions test operating system, file system or command processor status. In addition to the tests below, there are many internal variables and variable functions which allow you to test the status of many other parts of the system.

In the descriptions below of the various status tests, the status tests are true if and only if the specified condition is true.

DEFINED *variable*

If ***variable*** exists in the environment, the expression is true. This is equivalent to testing whether or not ***variable*** is nonempty, for example the following two commands are equivalent:

CMD.EXE has several internal variable names which have no special meaning under 4NT or Take Command: CD, CMDCMDLINE, CMDEXTVERSION, DATE, RANDOM, TIME. However, for the sole purpose of CMD.EXE "emulation", the **DEFINED** test of these variables will always be **true**.

Notes: [GOSUB variables](#)^[22] and [internal variables](#)^[34] always fail the **DEFINED** test.

ERRORLEVEL

[\[relational operator\]](#)^[37] ***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 non-zero number to indicate an error. The [relational operator](#)^[37] may be any of those listed above (e.g., **EQ**, **GT**). 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 ***filename*** matches a file which exists, the expression is true. You can use wildcards in the filename, in which case the expression is true if any file matching the wildcard name exists.

WARNING: In some operating systems the expression will be true if there is either a ***file*** or a ***directory*** named ***filename***. Use **ISFILE** or **ISDIR** instead.

ISALIAS *aliasname*

If ***aliasname*** is defined as an alias, the expression is true.

ISAPP *appname*

If ***appname*** matches the name of an application which is currently running, the expression is true. *To match a specific application, you must enter the full pathname of the application.* Partial names and wildcards will yield undependable results. Both the short and long filename forms of the name will be checked (see [LFN File Searches](#)^[47] for details on the correspondence between short and long filenames).

**ISDIR *path*
DIREXIST *path***

If the directory specified by ***path*** exists, the expression is true. ***Path*** may be either absolute or relative. For compatibility with Novell DOS / OpenDOS, **DIREXIST** may be used as a synonym for **ISDIR**.

ISFILE *filename*

If ***filename*** matches a file which exists, the expression is true. You can use wildcards in the filename, in which case the expression is true if any file matching the wildcard name exists. **ISFILE** matches only files, not directories.

ISFUNCTION *name*

If the user-defined function ***name*** is loaded, the expression is true.

Logical Expressions

A logical expression is one of the following:

- ▶ a [relational expression](#)^[3†]
- ▶ a [status test](#)^[3†]
- ▶ a [logical expression](#)^[3†] preceded by the unary logical operator NOT
- ▶ two [logical expression](#)^[3†]s connected by a binary logical operator

Logical operators

operator	type	usage	value is TRUE if
NOT	unary	NOT cond	cond is FALSE.
.AND.	binary	cond1 .AND. cond2	both cond1 and cond2 are TRUE.
.OR.	binary	cond1 .OR. cond2	at least one of cond1 and cond2 is TRUE.
.XOR.	binary	cond1 .XOR. cond2	one of cond1 and cond2 is TRUE, and the other one 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 order of testing by grouping conditions with parentheses, for example (enter this on one line):

```
if (%a == 1 .or. (%b == 2 .and. %c == 3)) echo something
```

Combining logical expressions

Parentheses can only be used when the portion of the **expression** inside the parentheses contains at least one of the **binary logical operators** **.and.**, **.or.**, or **.xor.**. Parentheses on a simple expression 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 tests is FALSE, the second is TRUE:

```
(a == a)
(a == a .and. b == b)
```

Parentheses may be nested.

Examples

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":

```
if "%@left[2,%@line[myfile,0]]" == "GO" call go.btm
```

This example tests whether there is more than 500 Kbytes of free memory:

```
c:\> if %@dosmem[K] gt 500 echo Over 500K free
```

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
```

2.4 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, or for accessing files on remote systems.

Most of the features explained in this section apply to the command processor commands only, and generally can not be used to pass file names to external programs unless those programs were specifically written to support these features.

The features discussed in this section are:

- ▶ [Wildcards](#) ^[36]
- ▶ [Attribute Switches](#) ^[39]
- ▶ [Date, Time, and Size Ranges](#) ^[40]
- ▶ [File Exclusion Ranges](#) ^[45]
- ▶ [Multiple Filenames](#) ^[45]
- ▶ [Include Lists](#) ^[46]
- ▶ [Extended Parent Directory Names](#) ^[47]
- ▶ [LFN File Searches](#) ^[47]
- ▶ [Executable Extensions](#) ^[48]
- ▶ [@File Lists](#) ^[49]
- ▶ [Command Switches for File Selection](#) ^[51]
- ▶ [Using Internet URLs](#) ^[51]
- ▶ [FTP/HTTP Servers](#) ^[52]

2.4.1 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](#) ^[45] (for internal commands), or [EXCEPT](#) ^[204] (for external commands). **The wildcard matching rules are the same.**

Most internal commands accept filenames with wildcards anywhere that a full filename can be used. There are two wildcard characters, the [asterisk](#) ^[36] "*" and the [question mark](#) ^[36] "?". Additionally, there is a method of specifying a [set of permissible characters](#) ^[36]. Note the issues about

[matching short file names](#)³⁶.

Asterisk "*" wildcard

An **asterisk** "*" in a file specification means "a set of any characters or no character in this position". For example, this command will display a list of all files (including directories, but excluding those files and directories with at least one of the attributes *hidden* and *system*) 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*
```

The command processor also lets you 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
```

Question mark "?" wildcard

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 Microsoft wildcard syntax, and are not likely to work properly with software other than Take Command and 4NT.

"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*.

Note: For historic reasons the question mark is occasionally but incorrectly referred to as "double quote".

Specific character set

In some cases, the question mark wildcard may be too general. In JP Software's command processors you can also specify the exact set of 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 accept a file that begins with an **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](#)^[46]. For example, if you have a file on an LFN drive 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, the command processor will attempt to delete *LETTER1* and then *V2*, because the semicolon indicates an [include list](#)^[46]. 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
```

Matching short file names

If the .INI directive Win32SFNSearch is either not used, or is set to YES (its default value), wildcard searches accept a match on either the LFN OR the SFN to match the behavior of CMD. This may cause some files to be found more than once, and for some files to be found because of SFN match only. In most situations this is not actually desirable, and can be avoided by setting the directive to NO.

Note: The wildcard expansion process will attempt to allow both CMD-style "*extension*" matching (only one extension, at the end of the word) and the advanced 4NT and Take Command *string* matching (allowing things like *.*.abc) when an asterisk is encountered in the destination of a [COPY](#)^[164], [MOVE](#)^[246] or [REN/RENAME](#)^[270] command.

2.4.2 Attribute Switches

Many commands in 4NT and Take Command include a /A: switch, which allows you to select files for the command to process based on their [attributes](#)^[442]. 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:

- R** Read-only
- H** Hidden
- S** System
- A** Archive
- D** Directory

On **NTFS** volumes, the extended attributes below are also available.

- E** Encrypted
- C** Compressed
- I** Not content-indexed
- J** Junction (reparse point)
- N** Normal (cannot be used for file selection)
- O** Offline
- P** Sparse file
- T** Temporary

The **N** (normal) attribute is not stored on disk. It is dynamically generated by the operating system if none of the other attributes is set. Its use for file selection is not supported in either commands or variable functions.

If no attributes are listed at all (*i.e.*, **/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 with that attribute will be excluded. 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.

When multiple attributes are preceded by **+**, only files which have at least one of these attributes will be selected. 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.

Example

The **dangerous** command below will make all hidden and/or read-only files in the default directory visible and writable, but not modify the attributes of files which are neither hidden nor read-only (thus not reporting files already in the desired state):

```
attrib /e /p /a:+r+h+s -r -h -a
```

2.4.3 Date, Time & Size Ranges

Most internal commands which accept wild cards 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 whether or not the file meets the range criteria that you have specified.

The command processor also supports [File Exclusion Ranges](#)⁴⁵ to exclude files from a command. These are similar to date, time, and size ranges, but have a slightly different purpose.

A range begins with the switch character (**/**), 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]**.

See the individual range types for details on specifying ranges:

- ▶ [Size Ranges](#)⁴²
- ▶ [Date Ranges](#)⁴²
- ▶ [Time Ranges](#)⁴⁴

General Rules for Using Ranges

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 10-27-97 and 10-30-97 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, to select files between 100 and 200 bytes long could also be entered as **/[s200,100]**.

If you combine two types of ranges, a file must satisfy both ranges to be included. For example, **/[d2-8-97,2-9-2004] /[s1024,2048]** means files last modified between February 8, 1997 and February 9, 2004, which are also between 1,024 and 2,048 bytes long.

When you use a 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. Any

exceptions are noted in the descriptions of individual commands.

For example, to get a directory of all the *.C files dated October 1, 2004, you could use this command:

```
[c:\] dir [/d10-1-2004,+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 that you changed yesterday or today to your floppy disk, you can use this command:

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

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](#)^[351] or [TIME](#)^[356]) and built-in variable functions (e.g. [@DATE](#)^[369], [@TIME](#)^[408], [@MAKEDATE](#)^[399], [@MAKETIME](#)^[400], [@FILEDATE](#)^[380], [@FILETIME](#)^[384], or [@EVAL](#)^[374]). These variables and functions allow you to perform arithmetic and date / time calculations. You may also define your own variable functions, to perform more complex manipulations repetitively.

The FAT file system maintains a single date and time for each file, reflecting the last time the file was written, and does not provide LFN support. This is the date and time used by the command processor on a FAT drive regardless of the operating system used to access the drive. For example, a virtual disk created under Win98 created with the built-in RAMDRIVE.SYS is a FAT drive, without LFN or creation/access date support.

File systems which support long filenames maintain 3 sets of dates and times for each file: creation, last access, and last modification. By default, date and time ranges work with the last modification (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 drives only the last access date is recorded; the last access time is always returned as 00:00. On NTFS drives last access information includes both actual date and actual time.

Date, time, and size ranges can be used with the [ATTRIB](#)^[146], [COPY](#)^[164], [DEL](#)^[172], [DESCRIBE](#)^[176], [DIR](#)^[179], [EXCEPT](#)^[204], [FFIND](#)^[206], [FOR](#)^[210], [LIST](#)^[237], [MOVE](#)^[246], [RD](#)^[267], [REN](#)^[270], [SELECT](#)^[275], and [TYPE](#)^[308] commands. They cannot be used with filename completion or in filename arguments for variable functions.

Do not use ranges with @file lists. See [@file lists](#)^[49] for details.

2.4.3.1 Size Ranges

Size ranges simply select files whose size is between the limits given. All ranges are inclusive. The second argument of a size range is optional. If you use a single argument, you will select all files of the specified 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.

Either or both values in a size range can be suffixed with a scale factor from the table below. Lower case letters denote a power of 1,000, upper case letters a power of 1,024 (2^{10}).

Code	Scale Factor		Code	Scale Factor		Unit Name
k	1,000	10^{**3}	K	1,024	2^{**10}	kilobyte
m	1,000,000	10^{**6}	M	1,048,576	2^{**20}	megabyte
g	1,000,000,000	10^{**9}	G	1,073,741,824	2^{**30}	gigabyte
t	1,000,000,000,000	10^{**12}	T	1,099,511,627,776	2^{**40}	terabyte

Notes

1) Disk manufacturers use the prefixes adopted from the metric system (kilo, mega, giga, tera) in their original, standard meaning (powers of 1,000), while memory manufacturers and Microsoft use the slightly larger powers of 1,024 (2^{10}).

2) The **scale code** is one of the few instances in which JP Software command processors are **case sensitive**.

Examples of size ranges:

Specification	Selects Files of Length
/[s0,0]	zero (empty)
/[s1M]	2^{**20} bytes or larger
/[s10k,+200]	between 10,000 and 10,200 bytes, inclusive
/[s10,153k]	between 10 and 153,000 bytes, inclusive

2.4.3.2 Date Ranges

Date ranges select files that were created or last modified at any time between the two dates. For example, /[d12-1-04,12-5-04] selects files that were last modified between December 1, 2004, and December 5, 2004.

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, /[d12-31-04,1-1-05] is equivalent to /[d12-31-2004,1-1-2005], and selects files modified on December 31, 2004 or January 1, 2005.

If either argument begins with a four digit year (which must be greater than 1900), it is assumed to be a date in the international format **yyyy-mm-dd**, otherwise it is assumed that the date elements are in the order appropriate for your **locale**. All non-ISO date examples in the HELP use the USA format: **mm-dd-yy**, unless otherwise stated explicitly.

By default, the time for the first date is assumed to be **00:00:00** (beginning of the day) and the time for the second date is assumed to be **23:59:59** (end of day). You can alter these defaults by including specific start and stop times inside the date range. The time is separated from the date with an at sign [**@**]. For example, the range /[d7-1-04@8:00a,7-3-04@6:00p] selects files that were modified at

any time between 8:00 am on July 1, 2004 and 6:00 pm on July 3, 2004. 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, `/[d10-1-04]` selects files dated between October 1, 2004 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>/[d1-27-2004,+3]</code>	modified between 1-27-2004 and 1-30-2004
<code>/[d1-27-2004,-3]</code>	modified between 1-24-2004 and 1-27-2004
<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 the command processor. It takes advantage of the facts 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, `/[d12-8-04@12:00,+2@12:00]` selects files that were last modified between noon on December 8 and noon on December 10, 2004. 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.

Windows keeps track of the date a file was created, the date it was last modified (written), and the date it was last accessed. You can specify which date and time is used in a date range by adding **a** (access), **c** (creation), or **w** (write) after the **d** in the range. For example, to select all files created between February 1, 2004 and February 7, 2004 you would use `/[dc2-1-04,2-7-04]`. If you don't specify which date and time to use, the command processor will use the date the file was last modified (written).

NOTE: On FAT drives which support long filenames, only the last access date is recorded; the last access time is always returned as 00:00. However, on NTFS drives, last access information includes both date and time.

Date and time ranges may not always work as you expect across a network, including on FTP or HTTP 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 or time ranges to yield the results you want on a remote system.

Defaults for Date Ranges

Start date:	Today
End date:	Today
Time of first argument:	Beginning of the day (00:00:00)
Time of second argument:	End of the day (23:59:59)
Missing second argument:	Current date and time

2.4.3.3 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:

Specification

`/[t12:00p,+120]`

`/[t-120,+120]`

`/[t0:00,11:59]`

Selects Files

modified between noon and 2:00 PM on any date

modified between two hours ago and the current time on any date

modified in the morning on any date

The separator character used in the time may vary depending upon your country information.

MS Windows (except Windows 3.11 or earlier) keeps track of the time a file was created, the time it was last modified (written), and the time it was last accessed. You can specify which time is used in a date range by adding `a` (access), `c` (creation), or `w` (write) after the `t` in the range specification. For example, to select all files *created* between noon and 2:00 pm, you would use `/[tc12:00p,2:00p]`. If you don't specify which date and time to use, the command processor will use the date the file was last modified (written).

NOTE: On FAT drives which support long filenames, only the last access date is recorded; the last access time is always returned as 00:00. However, on NTFS drives, last access information includes both date and time.

Time ranges may not always work as you expect across a network, including on FTP or HTTP 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.

Defaults

Start time:	Current time
End time:	Current time

2.4.4 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 the switch character (usually 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](#)^[36], 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](#)^[40]. 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, a unique feature of JP Software's command processors, work for internal commands. The [EXCEPT](#)^[204] command can be used to exclude files from processing by any external or internal command which ignores files with the hidden attribute. You can utilize the file exclusion range with external commands utilizing the [FOR](#)^[210] command, using the format `FOR /[!exclusionrule] ... DO externalcommand`; however, the performance may not be as good, since the external command is started separately for each match.

Note: File exclusion first checks to see if a file specification with embedded brackets exactly matches an existing file, and if no such file is found, then it interprets those brackets as wildcards.

When you use a file exclusion range in a command it should immediately follow the command name. See [Using Ranges](#) under [Date, Time, and Size Ranges](#)^[40] for additional details.

Also see [Include Lists](#)^[46].

2.4.5 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](#)^[36] 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](#)^[164] or [MOVE](#)^[246], 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](#)^[36] and [include lists](#)^[46], multiple filenames will work with internal commands but 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 [@File Lists](#)^[49] as well as the [DO](#)^[191], [FOR](#)^[210] and [SELECT](#)^[275] commands for other ways of passing multiple file names to a command.

2.4.6 Include Lists

Any internal command that accepts [multiple filenames](#)^[45] 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. See the rules to determine when a [semicolon is part of a file name](#)^[46] and when an include list separator below.

For example, you can shorten this command which uses multiple file names:

```
[c:\] copy a:\details\file1.txt a:\details\file1.doc c:
```

to this using an include list:

```
[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](#)^[164], 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](#)^[179] and [SELECT](#)^[275] 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](#)^[36] and [multiple filenames](#)^[45], the include list feature will work with internal commands, but not with external programs (unless they have been programmed especially to support them).

The maximum length of an include list is 2047 characters (same as the maximum length of a single file name).

Semicolons in filenames

Since a semicolon (";") is a valid (albeit unfortunate) character in a file name, you must quote any such name if you don't want the command processor to treat it as an include list.

Starting with Release 6.01, if a filename parameter includes a semicolon, the command processor first attempts to find a filename containing an embedded semicolon. If found, the filename is used. Only if no such file is found is the semicolon considered an include list separator.

Also see [File Exclusion Ranges](#)^[45].

2.4.7 Extended Parent Directory Names

The command processor allows you to extend the 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, i.e., in the parent directory, and `...\FILE.DAT` refers to a file two levels up, i.e., in the parent of the parent directory. If you are in the `C:\DATA\FINANCE\JANUARY` directory, you can copy the file `LETTERS.DAT` from the directory `C:\DATA` to drive A: with the command

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

Note: This extended notation may not be understood by external programs. Consider using the [@FULL](#)^[388] function to expand file and directory references when necessary:

```
[C:\DATA\FINANCE\JANUARY] myprog %@full[...\LETTERS.DAT]
```

2.4.8 LFN File Searches

This topic describes special considerations applicable to volumes which support long file names (including VFAT, FAT32, and NTFS volumes). All files on such volumes have a short (FAT-compatible 8.3) file name SFN. A file which was created (or renamed to) a name which contains lower case letters or other characters not compatible with SFN, or a name longer than 8 characters, or an extension longer than 3 characters, or more than one period (.) in its name will have both the long file name (LFN) specified, and an SFN automatically generated by the file system. The SFN associated with an LFN may change when the file is moved or copied even when the LFN is not changed.

When the command processor performs a wildcard search, **by default** it examines both forms of each file name to be compatible with the Microsoft command processors `CMD.EXE`. The long filenames are checked first, followed by the short file names. Matching files which have only a short filename will be found during the first search, because in that case the file system treats the SFN name as if it were a LFN.

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 *but will be reported with the LFN*.

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 generates short filenames that end "~1.", "~2.", etc. If you use a command such as

```
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 dialogs, or with the [Win32SFNSearch](#)^[112] directive in the [INI file](#)^[89]. Set Win32SFNSearch 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 *CMD.EXE*.

2.4.9 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* or *.LNK* to indicate that it contains information on how to execute a program under Windows; *.BTM*, *.BAT*, or *.CMD* to indicate a [batch file](#)^[321]. The exact list of default extensions for executable files varies slightly depending on which operating system and command processor you use, because each has its own rules for batch file extensions.

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.

Windows also includes the ability to associate file extensions with specific applications. See [Windows File Associations](#)^[461] for details on this feature and its relationship to executable extensions. Also see [Executable Files and File Searches](#)^[458].

You 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 for use only in the command processor, use the [SET](#)^[281] command to create a new environment variable. An environment variable is recognized as an executable extension if its name begins with a period.

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, alias, external program, 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.

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
```

If the command specified in an executable extension 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 including drive, path, filename, and extension. You can utilize other environment variables in the specification.

Once an executable extension is defined, any time you name a file with that extension as a command, it is equivalent to having typed the value of the extension variable, followed by the name of the file.

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](#)^[36], 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.

To remove an executable extension, use [UNSET](#)^[312] to remove the corresponding variable.

2.4.10 @File Lists

Some 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 name of the file containing the file list.

An @file list is simply a standard **ASCII** file containing the names of the files to process, one per line. The **Unicode** versions of 4NT and Take Command can also accept Unicode-encoded @file lists. This allows you to create a list of files for processing using output from [DIR](#)^[179] /B, [DIR](#) /F, or [FFIND](#)^[206], 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.

Commands supporting the @File syntax include:

ATTRIB^[146]
 COPY^[164]
 DEL / ERASE^[172]
 DESCRIBE^[176]
 DO^[191]
 EXCEPT^[204]
 FOR^[210]
 HEAD^[224]
 LIST^[237]
 MOVE^[246]
 RD / RMDIR^[267]
 REN / RENAME^[270]
 TAIL^[296]
 TOUCH^[304]
 TYPE^[308]

Note: Elements of an @file list are always processed exactly "as is". No further checking is done. This means that if a command allows options to restrict operations based on age (/U, /C), ranges (/d..., /t...), descriptions (/I), attributes (/A:), or location (/S), those options will **NOT** apply to the @file contents.

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 only 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**^[210]'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.

2.4.11 Switches for File Selection

Many of the file processing commands ([ATTRIB](#)^[146], [COPY](#)^[164], [DEL](#)^[172], [DESCRIBE](#)^[176], [MOVE](#)^[246], [REN](#)^[270], [TYPE](#)^[308], 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:[[-+]*rhsad*: Select files based on their attributes, for example **/A:rh** selects files which have the read-only and hidden attributes set. See [Attribute Switches](#)^[39] for details; see [File Attributes](#)^[442] for more information on attributes.

/I"text": Select files based on their description. [Wildcards](#)^[36] 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 **/I"[]"**. See [DESCRIBE](#)^[176] 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.

2.4.12 Using Internet URLs

If you type an Internet URL (Uniform Resource Locator) which begins with **http:** or **https:** at the prompt, the command processor will pass the URL to Windows. Normally 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:** or **https:** prefix and the browser software. While this association is standard for most browser installations, it may not be present on all systems.

The ability to "start" URLs in this way is restricted to those beginning with **http:** or **https:**. Other standard prefixes such as **ftp:**, **mail:**, and **news:** cannot be started directly from the prompt; you must enter these URLs directly into the browser software.

See [Waiting for Applications to Finish](#)^[29] for information on problems with waiting for the browser to finish after starting a URL.

2.4.13 FTP/HTTP Servers

The command processor allows direct access to remote servers from internal commands such as [COPY](#)^[164], [DEL](#)^[172], [DIR](#)^[179], [MOVE](#)^[246], [MD](#)^[243], [RD](#)^[267], [REN](#)^[270], and [SELECT](#)^[275] via several protocols:

- ▶ [FTP](#)^[52] (basic FTP)
- ▶ [TFTP](#)^[52] (Trivial FTP)
- ▶ [FTPS](#)^[52] (SSL FTP)
- ▶ [HTTP](#)^[52] (basic Web access)
- ▶ [HTTPS](#)^[52] (SSL HTTP)

The basic filename syntax for anonymous connections is:

```
"ftp://ftp.abc.com/..."
```

4NT and Take Command will look for your FTP user names and passwords in the file "**FTP.CFG**", which is kept in the 4NT or Take Command directory by default. You can specify another directory with the [FTPCFG](#)^[104] *.INI* directive. You must add entries to the FTP.CFG file manually. The format for each line is:

```
url username password
```

For example:

```
ftp://jpsoft.com fred secret
ftp://microsoft.com anyone mypassword
```

We recommend you encrypt this file if you're using NTFS. If FTP.CFG doesn't exist the first time 4NT or Take Command looks for it, it will be created as an encrypted file (NTFS only). **Note:** If you are using Win98 or FAT / VFAT, the file will *not* be encrypted and your user names and passwords will be unprotected in plain text.

Note: FTP, HTTPS, and SSL processing is provided through files *ipworks6.dll* and *ipwssl6.dll* included in the 4NT and Take Command distribution packages. Those files are installed by default in the command processor's directory.

- **FTP support:**

The quote marks around the directory/file name are required.

You can also specify a username and password:

```
"ftp://username:password@ftp.abc.com/..."
```

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 permission on server *ftp.abc.com* and a subdirectory of the root directory on that server is called *mydir*, you can display the files with this command (enter this on one line):

```
[c:\] dir "ftp://username:password@ftp.abc.com/mydir/*"
```

You can also use the internal [IFTP](#)^[229] command to start an FTP session with a server and then use a simplified syntax to manipulate files on the server.

The command processor normally connects to the FTP server on the default FTP port 21. If the FTP server you are connecting to uses a non-standard port, enter the port number (with a preceding colon) just after the server name, for example:

```
[c:\] dir "ftp://username:password@ftp.abc.com:8765/mydir/*"
```

To log on to a server which supports "anonymous" logins, enter the required user name (usually "anonymous") and password (usually your email address) using the syntax shown above, for example:

```
[c:\] dir "ftp://anonymous:email@domain.com@jpsoft.com/"
```

The command processor will distinguish between the "@" in the email address and the "@" before the server name in order to separate the parts of the URL properly.

If you use a partial file or path reference, such as

```
[c:\] dir "ftp:myfile.txt"
```

the command processor will attempt to build a fully qualified directory name in which to find the requested file or path, based on what the server reports as the current working directory. If an ftp file or path specification begins with a "~" (tilde, typically indicative of a path relative to the user's home directory), the command processor will instead pass the exact string directly to the remote server.

The command processor uses standard FTP commands to retrieve information about files and directories and manipulate those files and directories on FTP servers, and relies on the server's compliance with Internet FTP standards. If your server is not fully compliant, or does not operate in the manner that the command processor expects, 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 stored case of 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.

Time-related operations (e.g. switches like **COPY /C** or **/U**) may not always work reliably on FTP and HTTP Servers, due to differences in time zone and in the file time representations between your local system and the server. Be sure to experiment with the particular server in question before depending on commands which compare file times to yield the results you want.

Note: If you use a partial reference such as "ftp:mydir" outside the scope of an [IFTP](#) ^[229] command, the command processor will attempt to re-establish the last connection, if any. That new connection may or may not be logged to the last used directory on that sever. We recommend you always use a full reference (including server name) unless you are specifically taking advantage of an active [IFTP](#) ^[229] connection.

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. If you connect through a proxy server, you must use the [Proxy initialization directive](#) ^[109].

- **TFTP ("trivial FTP") support:**

See the [FTP](#) ^[52] section above for general notes and requirements.

TFTP is only available with [COPY](#) ^[164] (and with [MOVE](#) ^[246] when the source is a local file). The syntax is:

```
"tftp://server[:port]/filename"
```

For example:

```
copy update "tftp://190.189.188.0/update"
```

- **HTTP ("basic Web") support:**

See the [FTP](#)^[52] section above for general notes and requirements.

The **HTTP** syntax is:

```
"http://[user:password@]server[:port]/filename"
```

For example:

```
copy "http://jpsoft.com/"
```

- **FTPS ("SSL FTP") support:**

See the [FTP](#)^[52] section above for general notes and requirements.

The **FTPS** syntax is:

```
"ftps://[user:password@]server[:port]/filename"
```

For example:

```
copy "ftps://jpsoft.com/4nt/4nt600.exe"
```

- **HTTPS ("SSL HTTP") support:**

See the [FTP](#)^[52] section above for general syntax and requirements.

The **HTTPS** syntax is:

```
"https://[user:password@]server/filename"
```

For example:

```
copy "https://jpsoft.com/foo.htm"
```

2.5 Directory Navigation

The command processor and/or your operating system 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](#)^[159] command. JP Software command processors support these standard features, and offer a number of enhancements to make directory navigation much simpler and faster.

This section begins with a summary of all the command processor directory navigation features. It also

provides detailed documentation on the enhanced directory search features:

[Extended Directory Searches](#)^[56] and the [CDPATH](#)^[59].

The command processor 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. The command processor supports this method, and also offers two significant enhancements:

- ▶ [Extended Directory Searches](#)^[56] allow the command processor to search a "database" of all the directories on your system to find the one you want.
- ▶ The [CDPATH](#)^[59] allows you to enter a specific list of directories to be searched, rather than searching a database. Use [CDPATH](#)^[59] 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 Directory Change

The command processor supports the traditional methods of changing directories, and also offers several more flexible approaches:

- ▶ [Automatic directory changes](#)^[22] allow you to type a directory name at the prompt and switch to it automatically, without typing an explicit [CD](#)^[159] or similar command.
- ▶ The [CD](#)^[159] command can change directories on a single drive, and can return to the most recently used directory.
- ▶ The [CDD](#)^[160] command changes drive and directory at the same time, and can return to the most recently used drive and directory.
- ▶ The [PUSHD](#)^[264] command changes the drive and directory like [CDD](#)^[160], and records the previous directory in a directory "stack." You can view the stack with the [DIRS](#)^[190] command or the [@DIRSTACK](#)^[371] function, and return to the directory on the top of the stack with [POPD](#)^[261].

[CDD](#)^[160], [PUSHD](#)^[264], and [automatic directory changes](#)^[22] can also change to network drives and directories mapped to drive letters and to ones specified with UNC names (see [File Systems](#)^[439] for details).

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. JP Software command processors support three additional, simpler methods for returning to a previous directory:

- ▶ The [CD](#)^[159] and [CDD](#)^[160] commands 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](#)^[23] 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.

- ▶ The [POPD](#)^[267] command returns to the last directory saved by [PUSHD](#)^[264]. The directory stack holds 511 characters, enough for 20 to 40 typical drive and directory entries.

2.5.1 Extended Directory Searches

When you change directories with an [automatic directory change](#)^[22], [CD](#)^[159], [CDD](#)^[160], or [PUSHD](#)^[264] command, the command processor must find the directory you want to change to. To do so, it 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 relative or 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](#)^[59], then via an Extended Directory Search. This section covers only Extended Directory Searches, which are more flexible and more commonly used than [CDPATH](#)^[59].

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](#)^[59] 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 position and size of the popup directory search window from the [History tab](#)^[88] of the [configuration dialogs](#)^[82], or with the [CDDWinLeft](#), [CDDWinTop](#), [CDDWinWidth](#), [CDDWinHeight](#)^[100] directives in the [INI file](#)^[89]. You can also change the keys used in the popup window with [key mapping directives](#)^[114] in the [INI file](#)^[89].

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](#)^[160] command. When you create the database with [CDD /S](#), you can specify which drives 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](#). By default, the file is placed in the root directory of drive C:. You can specify a different location for this file on the [Misc tab](#)^[88] of the [configuration dialogs](#)^[82], or with the [TreePath](#)^[97] directive in the [INI file](#)^[89].

The same tree file is used by all JP Software command processors. If you are using two or more of our products on your computer and want to use different databases for each, provide unique values for the [TreePath](#)^[97] directive in their respective [INI file](#)^[89]s.

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](#) directive.

The [TREEEXCLUDE](#) variable can be used to specify which drives/directories should be excluded from inclusion in the directory database.

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 on the [Misc tab](#) of the [configuration dialogs](#), 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.

For example, suppose that you have a directory called `C:\DATA\MYDIR`, [CDPATH](#) is not set, and `C:\DATA` is not the current directory on drive C:. The following chart shows what [CDD](#) command you might use to change to this directory.

FuzzyCD	Type of extended search	Typical CDD Command
0	CDPATH only (default)	<code>cdd c:\data\mydir</code>
1	CDPATH or exact match	<code>cdd mydir</code>
2	CDPATH or leading match	<code>cdd myd</code>
3	CDPATH or any match	<code>cdd yd</code>

An extended directory search is not used if you specify a full directory path (one beginning with a backslash [N], 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](#)^[36] in the directory name. If you use a wildcard, an extended search will occur whether or not extended searches have been enabled.

When the command processor is changing directories and it finds *wildcards* in the directory name, it skips the explicit search and [CDPATH](#)^[59] 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 backslash.

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*`.

Disabling Extended Searches in Batch Files

When writing batch files you may want to use the [CD](#)^[159] or [CDD](#)^[160] command to switch directories without triggering an extended search. For example, you may need the search to fail (rather than search the extended search database) if a directory does not exist, or you may want to ensure that the extended search popup window does not appear in a batch file designed to run in unattended mode.

To disable extended searches, use the `/N` option of `CD` or `CDD`. When this option is used and a directory does not exist below the current directory or on the [CDPATH](#)^[59], the command will fail with an error message, and will not search the extended search database. For example this command might trigger an extended search:

```
cdd testdir
```

but this one will not:

```
cdd /n testdir
```

Note that this option is not available for [PUSHD](#)^[264]. To perform the same function when using `PUSHD`, save the current directory with `PUSHD` (without parameters) and then use `CDD /N` to change directories, for example:

```
pushd
cdd /n testdir
```

2.5.2 CDPATH (feature)

When you change directories with an [automatic directory change](#)^[22] or the [CD](#)^[159], [CDD](#)^[160], or [PUSHD](#)^[264] command, the command processor must find the directory you want to change to. To do so, it 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](#)^[56]. This section covers only the CDPATH.

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](#)^[339] is an environment variable, and is similar to the [PATH](#)^[341] 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 neither drive letter nor 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](#)^[281] command. The format of CDPATH is similar to that of 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.

If you often switch between "sibling" directories, i.e., between subdirectories of a common parent

directory. you can enter `..` as a search entry in your CDPATH.

2.6 Input / Output Manipulation

This section covers features to change how the command processor and some application programs handle input and output.

Internal commands and some external programs get their input from the computer's standard input device and send their output to the standard output device. Some programs, *including your command processor*, 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](#)^[179] 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](#)^[61], [Piping](#)^[64] and the [Keystack](#)^[65]. All three are explained in this section. In addition, 4NT and Take Command support ANSI X3.64 control sequences in displayed text. The last topic in this section explains [ANSI X3.64 support](#)^[64] in detail.

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 are most useful when they are combined with internal commands.

The [TEE](#)^[300] and [Y](#)^[317] commands are "pipe fittings" which add more flexibility to pipes.

- [Page and File Prompts](#)^[65]
- [Redirection and Piping](#)^[60]
- [Keystack](#)^[65]
- [ANSI X3.64 Support](#)^[64]

2.6.1 Redirection and Piping

This section covers redirection and piping. You can use these features to change how the command processor and some application programs handle input and output.

Internal commands and some external 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](#)^[179] 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. In addition, 4NT and Take Command support ANSI X3.64 control sequences in displayed text. The last topic in this section explains [ANSI X3.64 support](#) in detail.

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 are most useful when they are combined with internal commands.

The [TEE](#) and [Y](#) commands are "pipe fittings" which add more flexibility to pipes.

2.6.1.1 Redirection

Redirection can be used to reassign the standard input (stdin), standard output (stdout), and standard error (stderr) devices from their default settings (the keyboard and screen) to another device such as the printer or serial port, to a file, or to the Windows 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 whatever they were before the command.

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 standard redirection options supported by the command processor (see below for additional redirection options using numeric file handles):

- ▶ [Input redirection](#)
- ▶ [Output redirection](#)
- ▶ [NoClobber](#)
- ▶ [Multiple redirections](#)
- ▶ [Creating an empty file](#)
- ▶ [Redirection by handle](#)
- ▶ ["Here-document" redirection](#)

Input redirection

< filename To get input from a file or device instead of from the keyboard.

Output redirection

standard output	standard error	overwrite	append
yes	no	> filename	>> filename
no	yes	>&> filename	>>&> filename
yes	yes	>& filename	>>& filename

To use redirection, place the redirection symbol and **filename** at the end of the command line, 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).

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 after the closing parenthesis of a [command group](#)^[27] (e.g., **...**) **> report**), all of the output from the command group is redirected, and redirection ends when the command group is done.

WARNING: When you redirect the output of a command executed using **FOR** or **GLOBAL**,

Advanced redirection options (NoClobber, Multiple redirections, Creating an empty file, Redirection by handle, "here-document")

NoClobber

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](#)^[28] **/N1** command, the **Protect redirected output files** setting on the [Startup tab](#)^[82] of the configuration dialogs, or the [NoClobber](#)^[107] directive in the [INI file](#)^[89].

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
```

Multiple redirections

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.

Creating an empty file

You can use redirection to create an empty (zero-byte) file. To do so, enter **>filename** as a command, with no actual command before the **>** character. If you have enabled [NoClobber](#)^[107], use **>!filename**.

Redirection by handle

In addition to the redirection options above, the command processor also supports the Windows **CMD.EXE** syntax:

n>file	Redirect handle n to the named file
n>&m	Redirect handle n to the same place as handle m

Warning: You may not put any spaces between the *n* and the *>*, or between the *>*, *&*, and *m* in the second form. The values of *n* and *m* must be single decimal digits, and represent file handles. Windows defines 0, 1, and 2 as shown in the table below.

Handle	Assignment
0	standard input
1	standard output
2	standard error

The *n>file* syntax redirects output from handle *n* to *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 destination 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 by using standard command processor redirection features. The two examples above could be written as

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

and

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

"Here-document" redirection

Wherever input redirection is supported, you can use a UNIX-like "here-document" approach. The syntax is:

```
program << word
```

The current batch file is read up to the next occurrence of *word*, and the resulting text becomes standard input to *program*. For example:

```
c:\test\program.exe << endinput
input 1
input 2
input 3
endinput
echo This is the next line after "program.exe"
```

Special features of "here document":

- If the << is followed by a **hyphen** ("-"), the leading white space on the following lines will be removed before passing them to *program* (i.e. they will be effectively left-justified).
- The parser will perform **variable expansion** on each line, unless the word following << is enclosed in quote marks.

2.6.1.2 Piping

Piping is a special form of redirection, using an additional instance of the command processor for each instance of the *pip*ing specified in the command line.

You can create a *pipe* to send the *standard output* of a command (**command1**) to the *standard input* of another command (**command2**), and optionally also send the *standard error* as well:

standard output	standard error	command format
yes	no	command1 command2
yes	yes	command1 & command2

For example, to take the output of the [ALIAS](#)^[138] command (which displays a list of your aliases and their values) and pipe it to the external SORT utility to generate a sorted list, you would use the command:

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

To do the same thing and then pipe the sorted list to the internal [LIST](#)^[237] command for full-screen viewing:

```
[c:\] alias | sort | list /s
```

The [TEE](#)^[300] and [Y](#)^[317] commands 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.

WARNINGS: **4NT** and **Take Command** implement pipes by starting a new process for the receiving program. This process goes through the standard [secondary shell](#)^[467] start-up procedure, including execution of the [4START/TCSTART](#)^[67] file, for EACH receiving program. All of the sending and receiving programs run concurrently; the sending program writes to the pipe and the receiving program reads from the pipe. When the receiving program finds an End of File signal, it finishes reading and processing the piped data, and terminates. When you use pipes with **4NT** or **Take Command**, make sure you consider the possible consequences from using a separate process to run the receiving program, especially that it cannot create/modify/delete environment variables of the sending program, and inclusion of a command to change directories in the [4START/TCSTART](#)^[67] file may cause the new process to execute in a different directory. When you use more than one pipe in a single command, e.g. the second example above with LIST, each pipe adds another instance of the command processor.

2.6.2 ANSI X3.64 Support

There is no operating system support for ANSI X3.64 in Windows NT, 2000, XP, or 2003. For this reason, 4NT and Take Command contain their own limited ANSI X3.64 support (key substitutions are not supported). Even if you use these products on a Win98 system with ANSI.SYS loaded, only the command processor's internal version will be used. The command processor interprets only its own output, but not the output of external applications which will therefore not have the benefit of ANSI X3.64 support, except that in Take Command a [Caveman](#)^[79] application using "default colors" or "stdio" will also receive the benefit of ANSI X3.64 support. In some cases you can redirect the output of an application program to a temporary file, then send it through the command processor's ANSI X3.64 interpreter, e.g., by using the [TYPE](#)^[308] command. This will display ANSI X3.64 correctly, but will not

support an interactive application.

To utilize the 4NT or Take Command built-in ANSI X3.64 support you must enable it from the [Colors tab](#)^[85] of the [configuration dialogs](#)^[82], or with the [ANSI](#)^[99] directive in the [.INI file](#)^[89], or with the [SETDOS](#)^[283] /A command. You can determine whether or not ANSI X3.64 support is enabled with the [ANSI](#)^[348] internal variable.

Several commands in 4NT and Take Command provide replacements for ANSI X3.64 commands. For example, there are commands to set the screen colors and display text in specific colors and locations. These commands are easier to understand and use than the ANSI X3.64 control sequences.

For information on the specific ANSI X3.64 commands supported by 4NT and Take Command see the [ANSI X3.64 Command Reference](#)^[456].

2.6.3 Keystack

The [KEYSTACK](#)^[236] command overcome two weaknesses of input redirection: 1) some programs ignore standard input and read the keyboard through the operating system, and 2) input redirection doesn't end until the program or command terminates. You can't, for example, use redirection to send the first few commands to a program and then type the rest of the commands yourself. But **Keystack** lets you do exactly that.

Keystack sends keystrokes to an application program. Once the Keystack is empty, the program will receive the rest of its input from the keyboard. 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.

To place the letters, digits, and punctuation marks you would normally type for your program into the keystack, enclose them in quote marks:

```
[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 [Keys and Key names](#)^[464] for details on how key names are entered. See the [KEYSTACK](#)^[236] command for information on using numeric key values along with or instead of key names, and other details about using the Keystack.

You must start or activate the window for the program that will receive the characters before you place them into the Keystack. See [KEYSTACK](#)^[236] for additional details; see [ACTIVATE](#)^[136] for information on activating a specific window.

2.6.4 Page and File Prompts

Page Prompts

Several command processor commands can generate prompts, which wait for you to press a key to view a new page or to perform a file activity. When the command processor is displaying information in page mode, for example with a [DIR](#)^[179] /P or [SET](#)^[281] /P command, it displays the message

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

At this prompt, you can press **Esc**, **Ctrl-C**, 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.

File Prompts

During file processing, if you have activated prompting with a command such as [DEL](#)^[172] /P, you will see a prompt similar to the following before processing every file:

(Y / N / A / R) ?

You can answer this prompt by pressing **Y** for "Yes, process this file", **N** for "No, do not process this file", **A** for "process All remaining files" or **R** for "process the Remainder of the files without further prompting. The "**R**" and "**A**" responses are equivalent (the latter was added for compatibility with CMD.EXE versions which display a "Yes/No/All" prompt) . 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 [Interrupting a Batch File](#)^[327].

2.7 The Take Command Interface

The Take Command Window

- ▶ [The Take Command Window](#)^[66]
- ▶ [Menus](#)^[68]
- ▶ [Dialogs](#)^[71]
- ▶ [Tool Bar](#)^[75]
- ▶ [Status Bar](#)^[75]
- ▶ [Scrollback Buffer](#)^[75]
- ▶ [Highlighting and Copying Text](#)^[76]
- ▶ [Using a Windows Command Line](#)^[77]

Starting Applications

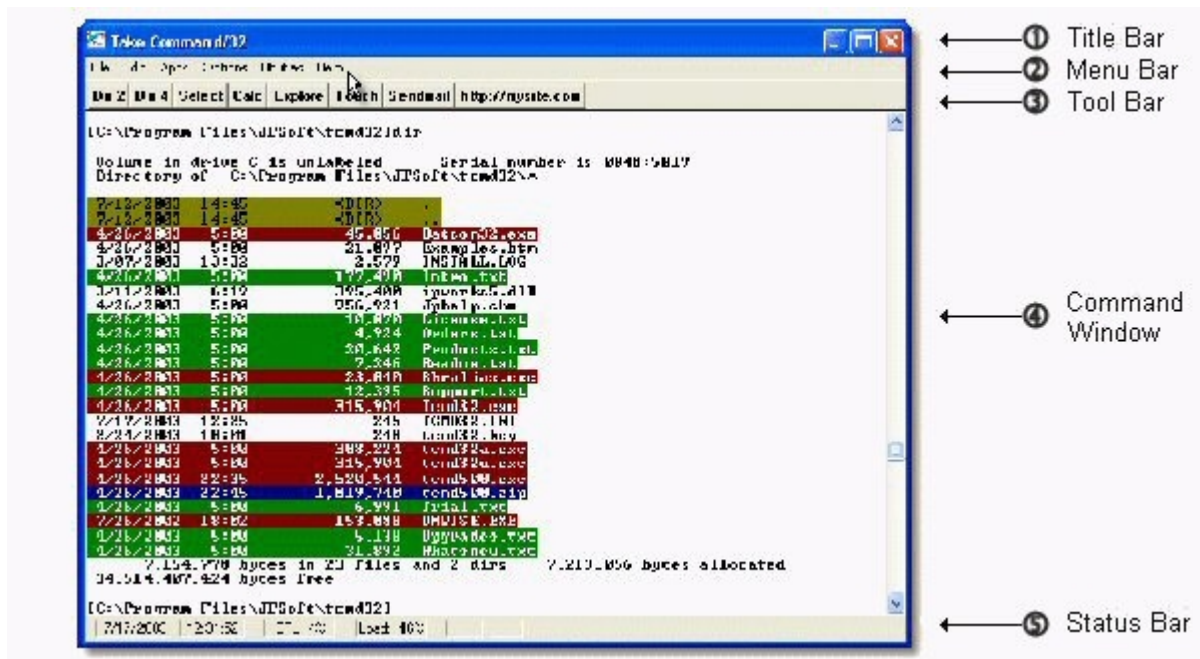
- ▶ [Starting Windows Applications](#)^[28]
- ▶ [Console Applications and the Console Window](#)^[78]

Take Command and the Windows Environment

- ▶ [Windows File Associations](#)^[467]
- ▶ [Using Drag and Drop](#)^[80]
- ▶ [Using DDE](#)^[81]

2.7.1 The Take Command Window

The Take Command window has **five** parts:



1. The **Title Bar** is the same as the one used in most Windows applications, with a control menu button on the left and the minimize, maximize, 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](#)^[316] command. You can also adjust the size of the Take Command window using standard window techniques, but see [Resizing the Take Command Window](#)^[77] for information about how Take Command's display changes when you do so.

2. See [Take Command Menus](#)^[68] for details about the **Menu Bar** and all of its menus.

3. The **Tool Bar** is used to execute internal or external commands, aliases, batch files, and applications with the click of a mouse. You can define up to 32 Tool Bar buttons; see [Tool Bar Dialog](#)^[72] for instructions. You can show or hide the Tool Bar with a choice on the [Options Menu](#)^[70], from [Window Option Tab](#)^[83] of the [Configuration Dialog](#)^[82], or with the [ToolBarOn](#)^[112] directive in [TCMD32.INI](#)^[89].

4. 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 scrollbar 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 [Highlighting and Copying Text](#)^[76] for information about saving and retrieving text in the Command Window and [The Command Line](#)^[10] for complete details about using the Command Line.

5. 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.

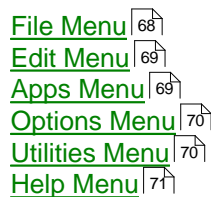
You can show or hide the Status Bar with a choice on the [Options Menu](#)^[70], with the [Window Option Tab](#)^[83] of the [Configuration Dialog](#)^[82], or with the [StatusBarOn](#)^[110] directive in [TCMD32.INI](#)^[89].

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 the [IBeamCursor = No](#)^[95] directive in the **[TakeCommand]** section of [TCMD32.INI](#)^[89] to force the use of an arrow cursor in all parts of the window.

2.7.1.1 Take Command Menus

Like most Windows applications, Take Command displays a menu bar along the top of the Take Command 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 **F**ile menu). You can also select a menu by pressing **Alt** or **F10** and then moving the highlight with the cursor keys.

The items on the menu bar allow you to select a variety of Take Command features:



2.7.1.1.1 File Menu

The File menu allows you to save or print the screen buffer, or exit Take Command.

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. This choice will also restart the desktop in Windows 98 and ME.

Shutdown

Exits Take Command and shuts down Windows.

Restart Windows

Exits Take Command, shuts down Windows and then reboots the system. This option does not

work in Windows 98 and ME.

Exit

Ends the current Take Command process.

2.7.1.1.2 Edit Menu

The Edit menu allows you to copy text between the Take Command window and the Windows clipboard. You can also access the clipboard with [redirection](#)^[61] to or from the CLIP: device, or with the [@CLIP](#)^[367] variable function.

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, below. If you hold down the right mouse button while you select a block of text, that block will be copied to the clipboard automatically when you release the button.

For more information on copying text see [Highlighting and Copying Text](#)^[76].

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.

Copy + Paste + Run

Copies the selected text from the Take Command scrollback buffer directly to the command line and executes the resulting command line.

Paste + Run

Copies text from the clipboard to the command line and executes the resulting command line.

Select All

Marks the entire contents of the Take Command scrollback buffer as selected text.

2.7.1.1.3 Apps Menu**Run**

Displays the [run dialog box](#)^[72] from which you can run an application or batch file. 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](#)^[78] to be visible or hidden. Take Command creates the console window to run character-mode applications. Many such applications can be run under Take Command's [Caveman](#)^[79] support, and will then display their output directly in the Take Command window. You can also control whether the console window remains hidden after an application finishes; see the Console **Hide** option on the [Windows tab](#)^[83] of the [configuration dialogs](#)^[82], or the [HideConsole](#)^[94] directive in the [.INI file](#)^[89].

2.7.1.1.4 Options Menu

Configure Take Command

Opens a [dialog](#)^[82] which you can use to change the configuration of Take Command.

Configure Tool Bar

Opens a dialog in which you can define up to 32 buttons for the Take Command [tool bar](#)^[75].

Command Log

Enables or disables command logging using the default log file (*TCLOG*) or the file you have chosen with the [LOG](#)^[242] command or the [LogName](#)^[106] directive in the [.INI file](#)^[89].

History Log

Enables or disables command history logging using the default log file (*TCHLOG*) or the file you have chosen with the [LOG /H](#)^[242] command or the [HistLogName](#)^[105] directive in the [.INI file](#)^[89].

Show Tool Bar

Enables or disables the Take Command [tool bar](#)^[75], 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 the Configure Tool Bar command, above.

Show Status Bar

Select this item to enable or disable the Take Command [status bar](#)^[75], which appears near the bottom of the Take Command window.

2.7.1.1.5 Utilities Menu

Find Files/Text

Opens the [Find Files Dialog Box](#)^[73] which lets you search for files or text interactively (see [FFIND](#)^[206] 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 the [Descriptions Dialog](#)^[74] edit window in which you can view and change the descriptions of files in any directory available on your system. See [DESCRIBE](#)^[176] for details on file descriptions.

Aliases

Opens the [Aliases](#)^[74] edit window in which you can view and change the list of current [aliases](#)^[138]. You can also use this window to import aliases from a file or to save the aliases to a file.

Environment

Opens the [Environment](#)^[74] edit window in which you can view and change the current [environment](#)^[336]. You can also use this window to import environment variables from a file or to

save the environment variables to a file.

Functions

Opens the [User Functions](#)^[74] edit window in which you can view and change the list of current [user-defined functions](#)^[218]. You can also use this window to import user functions from a file or to save the user functions to a file.

Editor

Starts the Windows Notepad editor or any other editor you have specified with the [Editor](#)^[103] directive in [TCMD32.INI](#)^[89] or on the [Misc tab](#)^[86] of the [configuration dialogs](#)^[82].

2.7.1.1.6 Help Menu

Also see the [Help File](#)^[423] topic.

Contents

Displays the **Table of Contents** of the Take Command help file (*jphelp.chm*) from which you can directly navigate to any topic. This is the same display you will see if you select the "**Contents**" tab from within the help system.

Search Topics

Displays the **Search dialog** of the Take Command help file (*jphelp.chm*) from which you can search for any topic. This is the same dialog you will see if you select the "**Search**" tab from within the help system.

Index

Displays the **Index dialog** of the Take Command help file (*jphelp.chm*) from which you can search for any keyword. This is the same dialog you will see if you select the "**Index**" tab from within the help system.

<http://jpsoft.com/>

A hyperlink to our World Wide Web site. Clicking it will attempt to display the JP Software home page in your default browser. Depending on your configuration, you may need to first establish an Internet connection.

Order from JP Software

A hyperlink to our Secure Online Store. Clicking it will attempt to display the store's first page in your default browser. Depending on your configuration, you may need to first establish an Internet connection.

About

Displays Take Command version, copyright, and license information.

2.7.1.2 Take Command Dialogs

The Take Command menus lead to several dialog boxes. Each is listed here for quick-reference, though in general you will find it easier to learn about each one from the context in which it is used (for example, the information referenced below on the tool bar dialog will be more useful after you have read the section on the [tool bar](#)^[75] itself).

Take Command 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 within this help system.

The reference in parentheses after certain dialogs listed below shows the name of the [menu](#)^[68] you

can use to access that dialog.

Run Program Dialog [72]	(Apps [69])
Properties Dialog [82]	(Options [70])
	Startup Tab [82]
	Windows Tab [83]
	Editing Tab [84]
	Colors Tab [85]
	History Tab [85]
	Syntax Tab [86]
	Misc Tab [86]
	Caveman Tab [87]
Tool Bar Dialog [72]	(Options [70])
Find Files/Text Dialog [73]	(Utilities [70])
Edit Descriptions Dialog [74]	(Utilities [70])
Aliases [74]	(Utilities [70])
Environment Variables [74]	(Utilities [70])
User Functions [74]	(Utilities [70])

2.7.1.2.1 Run Program Dialog

The Run Program dialog, started from the [Apps menu](#) [69], allows you to run a program by typing its name or browsing the disk.

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 a 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.

2.7.1.2.2 Tool Bar Dialog

This dialog, available from the Take Command [Options menu](#) [70], allows you to define or modify buttons on the [tool bar](#) [75].

The **Font Size** setting applies to the display text for all buttons on the Take Command tool bar.

Select the button you want to define, modify, or delete by clicking on it. A second dialog opens to let you define the label, command, directory, and mode for the button.

In the **Label** field, enter the text that you want to display on the button when it appears in Take Command's tool bar.

In the **Command** field, enter the command to be executed when you select the button on the tool bar. You can enter [multiple commands](#) [24] in the Command field by separating them with the command separator character [**&**]. You can use the **Browse** button to find a path and filename to be entered at the beginning of the Command field.

If the command line begins with an at-sign [@] it will not be added to the [command history](#)^[13] when the tool bar button is clicked. Otherwise, all commands executed from the tool bar are stored in the history.

Use the **Directory** field if a command must be started in a particular directory. If you leave this field blank, the command will be started in the current directory at the time the tool bar button is selected.

Use the radio buttons to select how you want the command to be executed:

- **Echo** means display the tool bar command on the command line, but do not execute it. You can add additional text to the line if you wish, then press Enter to execute the command. This is the default setting.
- **Echo & Execute** means display the tool bar command on the command line, then execute it immediately, without waiting for you to press Enter.
- **Execute w/o Echo** means execute the tool bar command immediately, without waiting for you to press Enter, and without displaying the command.

If you exit by choosing the **OK** button, any changes you have made will be saved in [TCMD32.INI](#)^[89], and reloaded automatically the next time you start Take Command. If you use the **Cancel** button, your changes will be discarded.

If you want to rearrange the order of the buttons on the tool bar, use a text editor (e.g. Windows' Notepad) to edit the [Buttons] section of [TCMD32.INI](#)^[89]. Simply rearrange the lines into the order you wish, and renumber the buttons accordingly.

Alternatively, the tool bar can be configured with the [TCTOOLBAR](#)^[299] command.

2.7.1.2.3 Find Files/Text Dialog

The Find Files/Text dialog box, available from the [Utilities menu](#)^[70], gives you the same features as the [FFIND](#)^[206] command, in dialog form.

Enter the file name or names you wish search in the **Files** field. You can use [wildcards](#)^[36] and [include lists](#)^[46] as part of the file name. To select files from previous searches in the same Take Command session, click on the down arrow beside the Files field, or press the up or down arrow while the input cursor is in the Files field. You can also use the **Browse** button to find files to include in the 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](#)^[36] and [include lists](#)^[46]. 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](#)^[36] 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. For example, to search for an A, followed by some number of other characters, followed by a B, enter the A*B as your search string. To search for the literal string A*B (A, followed by an asterisk, followed by B), enter `A*B` as your search string search string (the closing back-quote is optional).

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 [FFIND](#)^[206] command for more 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](#)^[442] 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. If you select **Dir Only** or **Dir & Below**, the search will begin in the current default directory, shown above the Files box. If you select **Entire Disk**, Take Command will use the drives that you specified in the **Disks** field. If you select **All Hard Disks**, Take Command will search all the hard disk drives it finds on your system.

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

If the search yields a list of matching files, you can save that list with the **Export** button or send it to the default printer with the **Print** button.

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 that dialog you can **Run** the file (if it is an executable file, a batch file, or has an [executable extension](#)^[461]), display the file with the **LIST** command, or **Edit** the file. If you choose **List**, the cursor will be placed on the first matching text within the file. When you exit from LIST or the editor, the original list of matching files will still be available.

2.7.1.2.4 Edit Descriptions Dialog

This dialog is available from the [Utilities menu](#)^[70]. Most of it 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](#)^[176] 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](#)^[179] and [SELECT](#)^[275] commands.

2.7.1.2.5 Aliases Window

This dialog is available from the [Batch Debugger](#)^[149], by using the [ESET](#)^[201] /W command, or from the [Utilities menu](#)^[70] (Take Command only). You can use this window to view, edit, add, and delete [aliases](#)^[138].

2.7.1.2.6 Environment Window

This dialog is available from the [Batch Debugger](#)^[149], by using the [ESET](#)^[201] /W command, or from the [Utilities menu](#)^[70] (Take Command only). You can use this window to view, edit, add, and delete [environment variables](#)^[336].

2.7.1.2.7 Functions Window

This dialog is available from the [Batch Debugger](#)^[149], by using the [ESET](#)^[201] /W command, or from the [Utilities menu](#)^[70] (Take Command only). You can use this window to view, edit, add, and delete user-defined [functions](#)^[218].

2.7.1.3 Tool Bar

The Take Command window has an optional Tool Bar that you can use to execute internal or external commands, aliases, or batch files with the click of a mouse.

To create buttons for the Tool Bar, select Configure Tool Bar from the [Options](#) ^[70] menu. This selection displays the [tool bar dialog](#) ^[72]. You can also configure the Tool Bar with the [TCTOOLBAR](#) ^[299] command.

You can define up to 32 Tool Bar buttons.

To enable or disable the Tool Bar, use the [ToolBarOn](#) ^[112] directive in [TCMD32.INI](#) ^[89], the Tool Bar Enable setting on the [Windows tab](#) ^[83] in the [configuration dialogs](#) ^[82], or the Show Tool Bar command in the [Options menu](#) ^[70].

The configuration dialog and *TCMD32.INI* settings are modified when you enable and disable the tool bar from the Options menu. This preserves the tool bar state when you close Take Command, and restores it the next time you start a Take Command session.

2.7.1.4 Status Bar

The Take Command window has an optional Status Bar that shows information about your system. To enable or disable the Status Bar, use the [StatusBarOn](#) ^[110] directive in the [TCMD32.INI](#) ^[89] file, the Status Bar Enable setting on the [Windows tab](#) ^[83] in the [configuration dialogs](#) ^[82], or the Show Statusbar command in the [Options menu](#) ^[70].

The configuration dialog and *TCMD32.INI* settings are modified when you enable and disable the status bar from the Options menu. This preserves the status bar state when you close Take Command, and restores it the next time you start a Take Command session.

The status bar displays the following information:

- ▶ The date and time, based on the Windows clock.
- ▶ The CPU usage percentage as reported by Windows.
- ▶ The percentage "memory load" as reported by Windows. If you are running a CPU monitor program or a program in the background that runs while the CPU is idle, the load may not be what you expect. This is due to the design of such programs, and is not a problem in Take Command.
- ▶ The state of the Caps Lock key on the keyboard.
- ▶ The state of the Num Lock key on the keyboard.

2.7.1.5 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 any Windows application.

You can also use the **Up** and **Down** 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 bottom of the buffer to display the text.

If you prefer to use the **Up/Down** and **PgUp** keys to access the command history (as in 4NT), see the [SwapScrollKeys](#)^[111] directive in [TCMD32.INI](#)^[89], or the corresponding option on the [History tab](#)^[85] of the [configuration dialogs](#)^[82]. SwapScrollKeys switches the keystroke mapping so that the **Up**, **Down**, and **PgUp** keys manipulate the command history, and **Ctrl-Up**, **Ctrl-Down**, **Ctrl-PgUp**, and **Ctrl-PgDn** are used to control the scrollback buffer. For more details see [Scrolling and History Keystrokes](#)^[25].

You can set the size of the scrollback buffer on the [Windows tab](#)^[83] of the [configuration dialogs](#)^[82], or with the [ScreenBufSize](#)^[96] directive in the [TCMD32.INI](#)^[89] file.

To clear the entire scrollback buffer, use the [CLS](#)^[163] /C command.

2.7.1.6 Highlighting & Copying Text

While you are working at the Take Command prompt you can use common Windows keystrokes to edit commands, and use the Windows clipboard to copy text between Take Command and other applications. You can also select all of the text in the Take Command Window 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](#)^[69].

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 movement 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 **Ctrl-V**, 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 keystrokes **Ctrl-Ins**, **Ctrl-Del**, and **Ctrl-Shift-Ins** for Copy, Cut, and Paste on the [Editing tab](#)^[84] of the [configuration dialogs](#)^[82], or with the [CUA](#)^[101] directive in the [TCMD32.INI](#)^[89] file.

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.

You can also use Windows' [Drag and Drop](#)^[80] facility to paste a filename from another application onto the command line, and you can access the clipboard with [redirection](#)^[61] to or from the **CLIP**: device, or with the [@CLIP](#)^[367] variable function.

2.7.1.7 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](#)^[179] (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 must check 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](#)^[109] directive in [TCMD32.INI](#)^[89], or the **Height** setting on the [Windows tab](#)^[83] of the [configuration dialogs](#)^[82]. The [_ROWS](#)^[355] internal variable 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](#)^[109] directive in the [TCMD32.INI](#)^[89] file, or the **Width** setting on the [Windows tab](#)^[83] of the [configuration dialogs](#)^[82]. The [_COLUMNS](#)^[355] internal variable 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 scrollbar 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 the configuration dialogs, or the current width of the window in the new font, whichever is larger.

2.7.2 Using a Windows Command Line

Take Command is a new environment that lets you perform tasks easily under Windows. You can use it to execute commands, start applications, and perform other work at the command line.

You may have accomplished some of these tasks by starting a Windows 98 or ME session to run 4DOS, or a Windows NT / 2000 / XP / 2003 session to run 4NT, JP Software's replacement character-mode command processors, or you may have used an "MS-DOS Prompt" or "Command Prompt" session to run the default command processor (*COMMAND.COM* or *CMD.EXE*). In either case, and especially if you are an experienced user of 4DOS or 4NT, you'll find plenty of familiar features in Take

Command. You'll also find a lot that's new and different.

What's new:

Take Command also offers a wide range of new Windows-related features which are not available in 4NT or *CMD.EXE*, including:

- ▶ A built-in [scrollback buffer](#)^[75] that lets you look back through the output from past commands.
- ▶ A standard Windows [menu bar](#)^[68] for access to many commonly-used Take Command features.
- ▶ A [status bar](#)^[75] showing the date, time, keyboard toggles, and the cpu and memory usage.
- ▶ A customizable [tool bar](#)^[75] that gives you quick access to commands and applications.
- ▶ Windows dialogs (accessible from the [Options](#)^[70] and [Utilities](#)^[70] menus) for editing environment variables, aliases, user functions, file descriptions, and startup parameters.
- ▶ High-speed, dialog-based file and text search (see "Find Files/Text" on the [Utilities menu](#)^[70]). The new [FFIND](#)^[206] command gives you the same capabilities at the Take Command prompt.
- ▶ A new technology, called "[Caveman](#)"^[79], which you can use to run many character-mode utilities in the Take Command window (see [Console Applications and the Console Window](#)^[78] for details).

What's Different:

While Take Command includes the command-line, batch file, and other capabilities provided by 4NT (and goes far beyond those provided by *CMD.EXE*), the Windows environment places a few limitations on how Take Command operates.

These limitations are minor — for example, some keystrokes are interpreted differently to conform more closely to Windows conventions, and there are some considerations when running batch files or aliases designed to work in character mode under a graphical program like Take Command.

There are also some differences between running under 4NT (or *CMD.EXE*) and running under Take Command. The primary differences are related to different methods for starting character-mode programs; this topic is covered in detail under [Console Applications and the Console Window](#)^[78].

In order to support the Take Command window scrollback buffer, some Take Command keystrokes are different from what you may be used to in our character-mode products. See [Scrolling and History Keystrokes](#)^[25] for more details.

Some [command-line editing](#)^[11] defaults have also been changed to conform more closely to Windows conventions. In Take Command the default editing mode is insert, not overstrike, and the default insert-mode cursor is a line, not a block. You can change these defaults with statements in [TCMD32.INI](#)^[89] or via the [Editing tab](#)^[84] of the [configuration dialogs](#)^[82].

2.7.3 Console Applications & the Console Window

The console session connected to Take Command is created when Take Command starts (this may cause a momentary flicker on your screen during the startup process). You can view this window at

any time with the **Alt-V** key or the **View Console** selection on the **Apps** menu. You can use **Alt-V** to return to the Take Command window, but only when the application in the console window has completed, and the input "focus" has returned to Take Command. You can control the width and height of the console window with the [ConsoleColumns](#)^[94] and [ConsoleRows](#)^[94] directives in the [TCMD32.INI](#)^[89] file, or the corresponding options on the [Windows tab](#)^[83] of the [configuration dialogs](#)^[82].

When you start a DOS or Windows character-mode application from Take Command, it is automatically run in this console session (to start a separate session for the application, use the [START](#)^[291] command). The console window automatically becomes visible when the application starts, and is hidden when the application exits. To leave the console window visible after the application exits, set the [HideConsole](#)^[94] directive to No in [TCMD32.INI](#), or set the corresponding option on the [Windows tab](#)^[83] of the configuration dialogs.

If you run a DOS or Windows character-mode program which does **not** exit immediately (for example, a DOS word processor or editor) you will be able to work in the console session, and return automatically to Take Command when you exit the application.

However, if you run a DOS or character-mode application from Take Command and the application exits quickly, without waiting for any input (for example, a utility like PKUNZIP), you may have to use **Alt-V** to return to the console window and view the output.

To make it easier to use character-mode applications from within Windows, Take Command also includes a technology called "Caveman." Caveman allows DOS and Windows 32-bit character-mode programs to display output and accept input within the Take Command window, and eliminates the requirement that you switch between the Take Command window and the console window. See the separate section on [Caveman](#)^[79] for complete details on enabling and using this approach.

If Take Command is running under Windows 98 or ME, and you execute a *.PIF* file instead of an executable *.COM* or *.EXE* file to start a DOS program, Take Command runs the program in a separate window instead of the console window. You can force Take Command to use the console window and ignore the *.PIF* file by using the full name of the file, including its extension, instead of just the base name. For example,

```
c : \wp
```

will open a new window if *WP.PIF* exists. But

```
c : \wp .exe
```

will ignore the *.PIF* file and run the executable in the console window.

You can also run a DOS or character-mode program in a separate window by using the [START](#)^[291] command.

2.7.4 Caveman (technical information)

(TC) The *Caveman feature* is only relevant to **Take Command**.

To make it easier to use character-mode applications from within Windows, Take Command includes a technology called "Caveman." **Caveman** allows DOS and Windows **character-mode** programs to run within the Take Command window.

Due to limitations in the way character-mode programs can operate under Windows, the techniques used by Caveman do not work well with all programs. This section tells you how to set up your system to make the best use of Take Command and Caveman.

When Caveman is running in Take Command, it continually scans the [console window](#)^[78] and updates the Take Command window with any changes it finds. It also sends keystrokes from the Take Command window to the console window. Caveman makes a character mode program appear to run in the Take Command window even though it is, in reality, running in the console session window.

Not all character-mode programs work well with Caveman. Simple utilities like XCOPY, FORMAT, and other programs which display basic "teletype-style" scrolling output, without significant use of popup windows, full-screen displays, or direct access to your system's hardware devices, work best. DOS programs which switch into graphics mode or an unusual text mode or which manipulate the video or keyboard hardware directly are likely to fail.

Unfortunately, Take Command cannot determine automatically which method is best for any given application. Therefore, you must enable Caveman for the applications you wish to use it with. While many applications will work properly, some experimentation on your part may be required.

Caveman also allows you to control whether the application's output is displayed using Take Command's screen colors or the colors set by the application, and whether Caveman should assume the application is a standard "tty-style" application which only writes to the "standard output" and "standard error" devices (if so Caveman can capture the output directly, rather than trying to mirror the application's screen).

When you install Take Command, Caveman is disabled. To enable it, use the [Caveman tab](#)^[87] from the Configure Take Command item on the Options menu. The same tab lets you list the applications you wish to run under Caveman, and select how colors are handled and whether the application uses standard output methods (see the [Caveman dialog help](#)^[87] for more details on these options). You can also use filenames and wildcards to include or exclude anything from an individual application to all programs on your system.

We recommend that you start by enabling all applications (use a "*" entry as described in the dialog documentation), then add exceptions for applications which do not work well with Caveman. You may want to add exceptions at the outset for full-screen programs like character-mode word processing or spreadsheet software, as these programs are less likely to behave properly under Caveman.

Once an application is listed in the dialog and enabled, it will run under Caveman, within the Take Command window, whenever you start it from the command line. However, if you start the application with the [START](#)^[291] command, the program will start in its own window.

You can also use START's **/CM** and **/CMSTDIO** switches to run an application under Caveman, even if it is not one of the applications listed implicitly or explicitly in the Caveman Applications Dialog.

Note: For compatibility reasons, commands on the right side of a [pipe](#)^[64] will not be executed under Caveman.

2.7.5 Using Drag & Drop

Take Command is compatible with Windows' **Drag-and-Drop** facility.

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 Window to other applications

using the clipboard; see [Highlighting and Copying Text](#)^[76] for details.

2.7.6 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](#)^[177] command.

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 you could use the following Visual Basic for Applications (VBA) fragment to send a DIR /W command to Take Command from within Word:

```
Dim TCMDChannel as Long
Shell ("d:\path\tcmd.exe")
TCMDChannel = DDEInitiate("TCMD32", "Execute")
DDEExecute TCMDChannel, "dir /w"
DDETerminate TCMDChannel
```

A similar sequence for a WinBatch script file (WBT) would be:

```
run ("d:\path\tcmd.exe", "")
channel = DDEInitiate("TCMD32", "Execute")
if channel != 0
    result = DDEExecute(channel, '*dir /w')
else
    Message("Error", "DDEInitiate failed!")
endif
exit
```

You could use the same approach to execute a batch file or any other Take Command command from within Word or WinBatch, 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](#)^[67] output from the commands you execute, because output on the Take Command window is not likely to be useful to the client program which invokes a command.

3 Configuration Options

4NT and Take Command offer a wide range of configuration options, allowing you to customize them for your needs and preferences.

- ▶ [Configuration Dialog](#)^[82]
- ▶ [Initialization Files](#)^[89]

Also see the [OPTION](#)^[253] and [SETDOS](#)^[283] commands.

3.1 Configuration Dialog

This dialog, available via the [OPTION](#)^[253] command, and also from the [Options menu](#)^[70] in Take Command, contains several "pages" or "tabs" of options that let you change the way the command processor looks and works. Each option sets a corresponding directive in the [.INI file](#)^[89] which was used to start the command processor.

Unless you select the **Cancel** button, any changes you make will take effect immediately. If you select **Apply**, the settings will only apply for the duration of that session. If you select **Save**, the settings will be recorded in the appropriate main section ("*[4NT]*" or "*[Take Command]*") of the [.INI file](#)^[89] and will be in effect each time you start that command processor. If you want to set configuration directives in the "*[Primary]*" or "*[Secondary]*" sections of the [.INI file](#), you must edit the file directly instead of using the dialog. Similarly, if you modified directives that originally resided in a separate [included](#)^[126] INI file, new directives will be saved to the main INI file but the included file itself will not be altered. It would be wise to verify that no "old" directive in included files override the changes you made into the main file.

For details about the [.INI file](#) and [.INI file directives](#), the allowable ranges for each, and the effect of each, see [TCMD32.INI](#)^[89] and [4NT.INI](#)^[89].

While you are using the dialog, you can move between sets of configuration options by clicking on the individual tabs. The sets of options available in this dialog are:

Startup ^[82]	Syntax ^[86]
Windows ^[83]	Miscellaneous ^[86]
Editing ^[84]	Internet ^[87]
Colors ^[85]	Batch Debugger ^[86]
History ^[85]	Caveman ^[87] (TC)

3.1.1 Startup Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[82] topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

Using the top entry, you can set the path to your [4START / 4EXIT](#)^[6] (4NT) or [TCSTART / TCEXIT](#)^[6] (Take Command) files if they aren't in the same directory as the command processor. This field sets the [4StartPath](#)^[93] or [TCStartPath](#)^[96] directive as appropriate.

Local History instructs the command processor to use a local or (if unchecked) global command history list and sets the [LocalHistory](#)^[96] directive.

Local Aliases instructs the command processor to use a local or (if unchecked) global alias list and sets the [LocalAliases](#)^[96] directive.

Local Directory History instructs the command processor to use a local or (if unchecked) global directory history and sets the [LocalDirHistory](#)^[96] directive.

Local Functions instructs the command processor to use a local or (if unchecked) global function list and sets the [LocalFunctions](#)^[96] directive.

SFN Search enables or disables a search of short file names after long filenames on LFN drives and sets the [Win32SFNSearch](#)^[112] directive.

PathExt enables or disables the use of the PATHEXT environment variable and sets the [PathExt](#)^[108] directive. Note that setting this variable but failing to create the PATHEXT environment variable will cause PATH searches to fail.

Delete to Recycle Bin determines whether deleted files are placed in the Recycle Bin and sets the [RecycleBin](#)^[109] directive.

Default batch echo sets the default batch echo state and sets the [BatchEcho](#)^[100] directive.

Copy prompt on overwrite determines whether the command processor will prompt in COPY or MOVE before overwriting an existing file if the command is being performed at the command prompt, and sets the [CopyPrompt](#)^[101] directive.

Protect redirected output files determines whether files are protected from being overwritten during [output redirection](#)^[61] and sets the [NoClobber](#)^[107] directive.

Wait for external apps determines whether or not the command processor waits for external applications to end and sets the [ExecWait](#)^[103] directive.

Update Titles prevents or allows updating of the command processor window title and sets the [UpdateTitle](#)^[112] directive.

Unix-style Paths enables or disables forward slashes in paths and sets the [UnixPaths](#)^[112] directive.

In the Logging section:

Errors determines whether error messages are written to the log file and sets the [LogErrors](#)^[106] directive.

Command turns command logging on by default and sets the [LogOn](#)^[106] directive. If you enter a file name in the File field, that file will be used for logging and the [LogName](#)^[106] directive will be set.

History turns history logging on by default and sets the [HistLogOn](#)^[105] directive. If you enter a file name in the File field, that file will be used for history logging that the [HistLogName](#)^[105] directive will be set.

3.1.2 Window Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[82] topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

In the Display section:

The Standard, Max, Min, and Custom radio buttons select the initial state for the command processor windows and set the [WindowState](#)^[97] directive.

The **X, Y, Width, and Height** fields set the initial size and position of the command processor window and set the [WindowX, WindowY, WindowWidth, and WindowHeight](#)^[97] directives. They are ignored unless the **Custom** radio button is also selected.

In the Text section:

The Tabs field sets the tabs stops for the [LIST](#)^[237] command's output and the [TabStops](#)^[111] directive.

(TC) The Width option sets the virtual screen width and the [ScreenColumns](#)^[109] directive.

(TC) The **Height** option sets the virtual screen height and the [ScreenRows](#)^[109] directive.

In the Window Configuration section:

(TC) **Enable Toolbar** sets the tool bar mode at startup and the [ToolBarOn](#)^[112] directive. The **Font Size** field sets the point size of tool bar text and the [ToolBarText](#)^[112] directive. The tool bar will only be visible if you have defined one or more [tool bar buttons](#)^[72].

(TC) **Enable Statusbar** sets the status bar mode at startup and the [StatusBarOn](#)^[110] directive. The **Font Size** field sets the point size of status bar text and the [StatusBarText](#)^[110] directive.

In the Cursor section:

(TC) The **Arrow** and **I-Beam** radio buttons let you select the type of cursor which Take Command will use and also set the [IBeamCursor](#)^[95] directive.

(TC) In the Screen Buffer section:

(TC) The **Buffer Size** field sets the number of bytes of memory allocated to the Take Command window buffer and also set the [ScreenBufSize](#)^[96] directive.

In the Console section:

(TC) The **Rows** and **Columns** options set the height and width of the console-mode screen buffer and the [ConsoleRows](#)^[94] and [ConsoleColumns](#)^[94] directives.

(TC) The **Hide** option determines whether the console window is hidden after a character-mode application ends and sets the [HideConsole](#)^[94] directive.

(TC) The **Font** button opens a standard Windows font dialog that lets you select the font, point size, and font style for the Take Command display. Font information is stored in a separate section of the *TCMD32.INI* file; therefore, the font option does not have a corresponding directive.

3.1.3 Editing Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[82] topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

In the Editing section:

The **Default** radio buttons set the start-up editing mode and set the [EditMode](#)^[102] directive. The "Insert" and "Overstrike" modes are returned to their original settings whenever you begin to edit a new line; the "InitInsert" and "InitOverstrike" modes are not.

(TC) The **Edit keys** select the keystrokes used for Cut, Copy and Paste, and set the [CUA](#)^[101] directive.

The **Cursor** fields set the cursor size for both overstrike and insert editing modes and set the [CursorOver](#)^[101] and [CursorIns](#)^[101] directives. (TC) These options have no effect if you have selected an Arrow cursor on the [Windows Options tab](#)^[83].

In the Filename Completion section:

The **Complete hidden files** option selects whether hidden and system files and directories can be returned by filename completion and sets the [CompleteHidden](#)^[101] directive.

The **Add '\' to Directories** option selects whether '\' is automatically appended to directory names (or '/' to FTP URLs) in filename completion, and sets the [AppendToDir](#)^[99] directive.

The **Options** field lets you select file types available for filename completion and sets the [FileCompletion](#)^[103] directive.

3.1.4 Colors Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[82] topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

The **Enable ANSI** option enables 4NT's and Take Command's ANSI X3.64 support and sets the [ANSI](#)^[99] directive.

The options in the **Colors** section let you select foreground and background colors for input and normal output in the command processor window. These options also set the [InputColors](#)^[113] and [StdColors](#)^[114] directives.

The options in the **List Colors** section select the foreground and background colors for the LIST window and set the [ListColors](#)^[113] directive.

The options in the **SELECT Colors** section set colors used by the [SELECT](#)^[275] command and also set the [SelectColors](#)^[114] directive.

The **DIR Colors** field lets you set the directory colors used by DIR and (in **4NT**), SELECT and also sets the [ColorDir](#)^[113] directive.

3.1.5 History Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[82] topic before continuing. If you need more details about a particular option, see the [Initialization Directives](#)^[93] related to that option.

1. The **Buffer Sizes** fields set the size of the **Command History** and **Directory History** buffers and also set the [History](#)^[95] and [DirHistory](#)^[94] directives.
2. The **Pop-Up Windows** fields set the size and position of :
 - the command and directory **History** search windows (sets the [PopupWinLeft](#), [PopupWinTop](#), [PopupWinWidth](#), [PopupWinHeight](#)^[108] directives), and
 - the special **Directory** popup window used by [extended directory searches](#)^[56] (sets the [CDDWinLeft](#), [CDDWinTop](#), [CDDWinWidth](#), [CDDWinHeight](#)^[100] directives).
3. In **Command History** section:
 - **(TC)** The **Scroll / History Keys** radio buttons select the set of keys used to manipulate the scrollbar buffer and the history list. Before changing these settings, be sure to read about the [SwapScrollKeys](#)^[111] directive, which is set by this option.
 - The **Minimum Saved** field sets the minimum command length to save in the command history and also sets the [HistMin](#)^[105] directive.
 - The **Copy to end**, **Move to end**, and **Wrap** options select how the command history operates. Before changing these settings, be sure to read about the [HistCopy](#)^[105], [HistMove](#)^[105] and

[HistWrap](#)^[106] directives, which are set by this option.

- The **Duplicates** option selects what the command history does with duplicate entries. See the [HistDups](#)^[105] directive, which is set by this option.

3.1.6 Syntax Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[82] topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

In the **Special Characters** section:

The **Separator**, **Escape**, and **Parameter** options let you select the command separator character, escape character, and parameter character. Please see the [CommandSep](#)^[100], [EscapeChar](#)^[103], and [ParameterChar](#)^[107] directives, which are set by these options, for details about these three characters, especially if you need to share batch files and aliases between 2 or more of our command processors.

The **Decimal** radio buttons select the character used as the decimal separator (Auto selects the separator designated by your country code) and sets the [DecimalChar](#)^[102] directive.

The **Thousands** radio buttons select the character used as the thousands separator (Auto selects the separator designated by your country code) and sets the [ThousandsChar](#)^[111] directive.

The radio buttons in the **Country** section select how time values are displayed by the command processor and set the [AmPm](#)^[99] directive.

The two **Default Beep** options set the length (in clock ticks) and frequency for the command processor's "error" beeps and also set the [BeepLength](#)^[100] and [BeepFreq](#)^[100] directives.

3.1.7 Batch Debugger Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[82] topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

The **Font** button selects the font style and size to use in the [debugger](#)^[149] window.

3.1.8 Miscellaneous Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[82] topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

In the **EVAL** section, the **Min** and **Max** fields set the minimum and maximum number of digits after the decimal point in values returned by [@EVAL](#)^[374] and set the [EvalMin](#)^[103] and [EvalMax](#)^[103] directives.

(TC) The **Editor** field selects the program to run for the "Editor" menu choice and sets the [Editor](#)^[103] directive.

In the **Descriptions** section, the **Enable** option enables or disables file descriptions and sets the [Descriptions](#)^[102] directive. The **Maximum Length** field sets the maximum text length of file descriptions and the [DescriptionMax](#)^[102] directive.

In the **Extended Directory Search** section, the **Search Level** radio buttons select the

[Extended Directory Search](#)^[56] mode and set the [FuzzyCD](#)^[104] directive. The **TreePath** field sets the path to the *JPSTREE.IDX* file which contains the extended directory search database and sets the [TreePath](#)^[97] directive.

3.1.9 Internet Options Tab

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[82] topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

In the **Firewall** section, **Host** is the server name of the firewall for FTP and HTTP commands. **User** is the user name if the firewall requires authentication. **Password** is the password if the firewall requires authentication. **Type** is the type of firewall.

In the **SMTP** section, **Server** is 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.) **Address** is 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.) **Port** is the port number to use for SMTP (the default is 25).

In the **HTTP Proxy** section, **Server** is the proxy server to use for HTTP calls. **Port** is the port number to use for HTTP calls.

Time Server is the internet time server to use to set the time. If no server is specified, TIME uses clock.psu.edu.

Passive FTP sets passive mode for FTP calls (sometimes required by a firewall).

3.1.10 Caveman Options Tab

(TC) The *Caveman Options Tab* is only available under **Take Command**.

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[82] topic before continuing. If you need more details about a particular option, see the Initialization directive related to that option.

This tab enables or disables Caveman support, and selects which character-mode applications should be run automatically under Caveman. See [Caveman](#)^[79] for complete information on Take Command's Caveman feature and the applications it supports.

The settings in this dialog let you enable or disable Caveman altogether, and specify the applications which are to run under Caveman. Information entered here is stored in the **[Caveman32]** section of *TCMD32.INI*.

Use the **Enable Caveman** checkbox at the top of the dialog to enable or disable Caveman globally. If this box is not checked, information in the application list below will be ignored (however, the application list can still be modified).

To add an application to the list, click the **Add** button to open the Caveman Apps dialog. To change an existing entry, select it and then click **Edit**, or double-click the entry, which will open the same dialog for that entry. **Delete** removes an entry from the list.

In the Caveman Apps dialog, enter either the name or the full path name of an executable file, e.g., **DISKCOPY.COM** or **C:\WINDOWS\COMMAND\DISKCOPY.COM**. If you use the full path name, the entry will apply only to that specific file. If you use just the file name, the entry will apply to any file of that name, regardless of its location. If you use both, the full path name entry will be used when you

execute that specific file, and the file name entry will be used for other files of the same name. Long filenames should **not** be quoted when they are entered in the **Command** field. You can use the **Browse...** button to open a standard Windows file open dialog to find a specific file.

You can also enter a [wildcard](#)³⁶ filename into the dialog, e.g., *.COM. This will enable Caveman for all character-mode programs which match the wildcard. For example, C:\WINDOWS\COMMAND*.COM would enable all .COM files in the named directory, and *.COM would enable all .COM files in all directories. [Extended wildcards](#)³⁶ may be used as well.

A single asterisk [*] wildcard matches all names, so an entry with "*" as the **Command** enables Caveman as the default for all character-mode programs.

The **Enable** checkbox enables the application to run under Caveman; if this box is not checked the application is disabled and will run in the console window.

The **Default Colors** checkbox tells Take Command to display the application's output using the Take Command window colors (if checked), or the colors set by the application (if unchecked). Check this box to create a more consistent display for applications which display scrolling output in default colors, and do not use full-screen displays or special colors of their own. If you check the **Default Colors** box for a wildcard name, it will apply to all programs which match the wildcard.

The **STDIO app** check box tells Take Command that this program uses standard (scrolling or "tty-style") output using the "standard output" (STDOUT) and "standard error" (STDERR) devices available to console applications. When selected, Caveman captures the application's output directly rather than reading and attempting to "mirror" the application's screen. This method is more reliable than mirroring the screen, but will result in lost output if the application uses any output method other than writing to STDOUT or STDERR. Some applications which appear to use only standard scrolling output may actually be using other or multiple output methods, so experimentation may be required to determine whether the STDIO selection is right for any particular application. If you check the **STDIO app** box for a wildcard name, it will apply to all programs which match the wildcard.

If you use wildcards to enable all programs (or a specific group of programs) for Caveman, you can create exceptions by entering specific filenames into the list and unchecking the **Enable** box for the programs you do not want enabled. You can use wildcards for the exceptions just as you would for programs to be enabled.

For example, you might choose to enable all character-mode programs for Caveman support using a "*" as described above, but also want to disable your DOS word processor called C:\WP\WP.EXE. To do so, add an entry for the full word processor file name (or one for WP.EXE, with no path), and make sure the **Enable** box is not checked when you add the entry. This would prevent WP.EXE from being run under Caveman despite the previous entry which enables all programs.

You can use the same technique to modify whether a particular application is to use **Default Colors** as described above. For example, suppose you enable all character-mode programs with default colors using a "*", and then want to disable default color output for the specific program COLTEST.EXE, so that it uses its own colors. To do so, add an entry for the full program file name (or one for COLTEST.EXE only, with no path), then uncheck the **Default Colors** box for that entry.

Take Command uses the last entry it finds which matches the name of the application you are starting. Therefore, to disable an application, to disable a group of applications with wildcards, or to change an application's Default Colors status, you must place the entry which disables the application(s) or changes their status **after** the entry which enables them. In the examples above, this would mean the "*" entry should come first, followed by the exception for WP.EXE (in the first example) or COLTEST.EXE (in the second).

3.2 Initialization Files

Part of the power of the command processor is its flexibility. You can alter its configuration to match your style of computing. Most of the configuration of the command processor is controlled through a file of initialization information:

Command Processor	Default initialization file
4NT	4NT.INI
TC	TCMD32.INI

This topic contains general information on command processor initialization. For information on specific directives see the separate topic for each type of directive:

- ▶ [Initialization Directives](#) ^[93]
- ▶ [Configuration Directives](#) ^[97]
- ▶ [Color Directives](#) ^[113]
- ▶ [Key Mapping Directives](#) ^[114]
- ▶ [Advanced Directives](#) ^[125]

These topics list the directives, with a one-line description of each, and a cross-reference which selects a full description of that directive. A few of the directives are simple enough that the one-line description is sufficient, but in most cases you should check for any additional information in the cross-reference topic if you are not already familiar with the directive.

We also discuss many ways of configuring the command processor in other parts of the online help:

- With aliases and user-defined functions you can set default options for internal commands and create new commands (see [Aliases](#) ^[318] and the [ALIAS](#) ^[138] and [FUNCTION](#) ^[218] commands).
- With [executable extensions](#) ^[48] you can associate data files with the applications you use to open them.
- With the [FILECOMPLETION](#) ^[346] environment variable and the [FileCompletion](#) ^[103] `.INI` directive, you can customize filename completion to match the command you are working with.
- With the [COLORDIR](#) ^[338] environment variable and the `ColorDir .INI` directive you can set the colors used by the DIR command.
- With the [SETDOS](#) ^[283] and [OPTION](#) ^[253] commands you can change most aspects of the command processor's operation "on the fly."
- With [command-line options](#) ^[4] you can specify where the command processor looks for its startup files and how it operates for a specific instance.

Modifying the `.INI` File

You can create, add to, and modify the `.INI` file in two ways: with the [configuration dialog](#) ^[82], available via the [OPTION](#) ^[253] command, or by editing the file with any ASCII editor.

You can also create, add to, and modify Take Command's `.INI` file with the [configuration dialog](#) ^[82], available on the **Configure Take Command** selection on the [Options menu](#) ^[70] or via the [OPTION](#) ^[253] command.

Most of the changes you make in the configuration dialog or with the `OPTION` command take effect

immediately. A few (e.g., those associated with the startup screen size) only take effect when you close the current process and start a new command processor session. See the online help for each individual dialog page if you are not sure when a change will take effect.

The dialogs handle most standard `.INI` file settings. The [Advanced directives](#)^[12b], the [Key Mapping directives](#)^[11d], and a few other individual directives (identified in the help topics for those directives) do not have corresponding fields in the configuration dialogs, and must be entered manually.

The command processor reads its `.INI` file when it starts, and configures itself accordingly. The `.INI` file is not reread when you change it manually. For manual changes to take effect, you must restart the command processor. 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.

Using the `.INI` File

Some settings in the `.INI` file are initialized when you install the command processor and 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.

The command processor searches for the `.INI` file in three places:

- ▶ If there is an "`@d:\path\inifile`" option on the startup command line, the command processor will use the path and file name specified there, and will not look elsewhere. See the [Command Line Options](#)^[4] details about the startup command line.
- ▶ If there is no `.INI` file name on the startup command line, the search proceeds to the same directory where the command processor program file is stored. This is the "normal" location for the `.INI` file. The command processor determines this directory automatically.
- ▶ 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 the command processor is loaded as a [secondary shell](#)^[46h], it does not search for the `.INI` file. Instead, it retrieves the [primary shell](#)^[46h]'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. You can override this behavior with the [NextINIFile](#)^[12b] directive, or by explicitly specifying the `.INI` file using the "`@d:\path\inifile`" syntax in the command processor startup command.

`.INI` File Sections

The `.INI` file has three possible sections: the first or **global** section, named after your command processor (**[4NT / Take Command]**); 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](#)^[46h]. In most cases, this is the only section you will need. Any changes you make to the `.INI` file with the [OPTION](#)^[25b] command are stored in the global section.

The **[Primary]** and **[Secondary]** sections include directives that are used only in primary and secondary shells, respectively. 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.

See [Primary and Secondary Shells](#)^[46] for an explanation of those terms

Sections that begin with any name other than the product name (**[4NT / Take Command]**), **[Primary]**, or **[Secondary]** are ignored, and permit you to add blocks of comments, such as an update history.

.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 must be within the [command line length limit](#)^[28].

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 dialogs](#)^[82] 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.

If you need to test different values for an `.INI` directive without repeatedly editing the `.INI` file, use the [OPTION](#)^[25] command or see the [INIQuery](#)^[98] directive.

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](#)^[12b] directive.

The [SETDOS](#)^[283] command can override several of the `.INI` file directives. For example, the cursor shape used by the command processor can be adjusted either with the `CursorIns` and `CursorOver` directives or the `SETDOS /S` command. The correspondence between a SETDOS option and a `.INI` directive is noted under both the individual help topic for that directive and under that option in the SETDOS help topic.

[Secondary shells](#)^[467] automatically inherit the configuration settings currently in effect in the previous shell. If values have been changed by [SETDOS](#)^[283] 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.

Types of Directives

There are various types of directives in the `.INI` file. The type of a directive is shown under the individual help topic for that directive. The types are distinguished by the kind of data, if any, that must be entered after the "=" (equal sign):

- ▶ **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 [Colors and Color Names](#)^[467] for the format of color names.
- ▶ **Name = Key (Default):** This directive takes a key specification. See [Keys and Keynames](#)^[464] 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:\TCMDI`) to avoid any possible ambiguities. A trailing backslash [`\`] at the end of the path name is accepted 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.**

- **Name:** This directive accepts NO parameters and a "=" is unnecessary (e.g. "ClearKeymap").

The command processor contains 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](#)^[419] for assistance.

Note: The directives are evaluated sequentially from top to bottom (within each section). When an [initialization](#)^[93], [configuration](#)^[97], or [color](#)^[113] directive appears more than once within a section, the last occurrence supersedes any previous one(s). Most [key mapping](#)^[114] and [advanced](#)^[125] directives are cumulative and may appear several times when several concurrent values are desired (such as when assigning several keystrokes to the same function).

3.2.1 Initialization Directives

The directives in this section control how the command processor starts and where it looks for its files. The initialization directives are:

4StartPath ^[93]	(4NT) Set location of 4START and 4EXIT
ConsoleColumns ^[94]	(TC) Sets the width of the console mode screen buffer
ConsoleRows ^[94]	(TC) Sets the height of the console mode screen buffer
DirHistory ^[94]	Set size of Directory History
DuplicateBugs ^[94]	Duplicate well-known CMD.EXE bugs
HideConsole ^[94]	(TC) Hide the console window after running a character-mode application
History ^[95]	Set size of Command History
IBeamCursor ^[95]	(TC) Set text cursor style
INIQuery ^[95]	Query for each line in the <i>.INI</i> file
LocalAliases ^[96]	Local vs. global aliases
LocalDirHistory ^[96]	Local vs. global directory history
LocalFunctions ^[96]	Local vs. global functions
LocalHistory ^[96]	Local vs. global history
PauseOnError ^[96]	(4NT) Force pause after displaying error message
ScreenBufSize ^[96]	(TC) Size of screen buffer
TCStartPath ^[96]	(TC) Path for TCSTART and TCEXIT
TimeServer ^[97]	URL of time server
TreePath ^[97]	Path for directory database, JPSTREE.IDX
UnicodeOutput ^[97]	Use Unicode for output redirection
WindowState ^[97]	Initial state for the command processor window
WindowX, WindowY, WindowWidth, WindowHeight ^[97]	Initial size and position of the command processor window

3.2.1.1 4StartPath

4StartPath = Path: **(4NT)** Sets the drive and directory where the *4START* and *4EXIT* batch files (if any) are located.

See other [Initialization Directives](#)^[93].

3.2.1.2 ConsoleColumns

ConsoleColumns = nnnn (80). **(TC)** Sets the width of the Take Command [console](#)^[78] screen buffer used for console applications. The range is 80 - 132 columns. This directive controls the text buffer used for the console window, **not** the actual width of the window on your screen. The window width is determined by the operating system, not Take Command, and may be smaller than the text buffer width (if it is, a horizontal scroll bar is provided to allow viewing of all text columns). Due to operating system limitations, under Windows 98 and ME this directive is ignored and the console window is fixed at 80 columns.

See other [Initialization Directives](#)^[93].

3.2.1.3 ConsoleRows

ConsoleRows = nnnn. **(TC)** Sets the height of the Command [console mode](#)^[78] screen buffer used for console applications. The default is 25 rows in Windows 98 and ME and 100 rows in Windows NT / 2000 / XP / 2003. The range is 25 - 4096 rows. This directive controls the text buffer used for the console window, **not** the actual height of the window on your screen. The window height is determined by the operating system, not Take Command, and may be smaller than the text buffer height (if it is, a vertical scroll bar is provided to allow viewing of all text rows). Due to operating system limitations, under Windows 98 and ME only values of 25 and 50 rows are accepted; if you use any other value, Take Command will revert to the default of 25 rows.

See other [Initialization Directives](#)^[93].

3.2.1.4 DirHistory

DirHistory = nnnn (1024): Sets the amount of memory allocated to the directory history list (in characters). The allowable range of values is 1,024 to 32767. If you use a global directory history list (see [Local and Global History Lists](#)^[16]), the DirHistory value is ignored in all sessions except that which first establishes the global list (you will need to close all current instances, including [SHRALIAS](#)^[290]).

See other [Initialization Directives](#)^[93].

3.2.1.5 DuplicateBugs

DuplicateBugs = Yes | NO: Tells the parser to duplicate certain well-known bugs in *CMD.EXE*. The only bug currently replicated is in the [IF](#)^[227] command.

See other [Initialization Directives](#)^[93].

3.2.1.6 HideConsole

HideConsole = YES | No: **(TC)** Normally Take Command will hide the [console window](#)^[78] after running a console mode program. If HideConsole is set to No, the console window will only be hidden until the first character-mode application you run. After that it remains on-screen. (The console window will be visible but will not remain "on top". You can make it the top window, or return the Take Command window to the top, by pressing **Alt-V**). 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 other [Initialization Directives](#)^[93].

3.2.1.7 History

History = nnnn (8192): Sets the amount of memory allocated to the command history list (in characters). The allowable range of values is from 4000 to 131071. If you use a [global history list](#)^[16], the History value is ignored in all shells except the shell which first establishes the global list (you will need to close all current instances, including [SHR_ALIAS](#)^[290]).

See other [Initialization Directives](#)^[93].

3.2.1.8 IBeamCursor

IBeamCursor = YES | No: **(TC)** 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**, the "*Arrow*" cursor is used in all areas of the window. This can be helpful on laptop systems where the I-Beam cursor is hard to see. You may need to restart Take Command to see the changes.

The cursors are those defined in your Windows configuration (generally under the "Pointers" tab of the "Mouse Properties" dialog).

IBeamCursor=YES uses the cursor referred to as "*I-Beam*" or "*Text Select*", and **IBeamCursor=No** uses the cursor referred to as "*Arrow*" or "*Normal Select*".

See other [Initialization Directives](#)^[93].

3.2.1.9 INIQuery

INIQuery = Yes | NO: If set to **Yes**, a prompt will be displayed before execution of each subsequent line in the current [INI file](#)^[89]. 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.

The dialog displayed by Take Command 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 <i>.INI</i> file (<i>i.e.</i> , cancels the INIQuery = Yes setting)

The 4NT prompt generated by INIQuery = Yes is:

[contents of the line] (Y/N/Q/R/E) ?

At this prompt, you may enter:

Y	Process this line and go on to the next.
N	Skip this line and go on to the next.
Q	Skip this line and all subsequent lines.
R	Execute this and all subsequent lines.
E	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.

See other [Initialization Directives](#)^[93].

3.2.1.10 LocalAliases

LocalAliases = Yes | **No**: The default value is **No**, which forces all copies of the command processor to share the same alias list. **Yes** keeps the lists for each shell separate. See [ALIAS](#)^[138] for more details on local and global alias lists.

See other [Initialization Directives](#)^[93].

3.2.1.11 LocalDirHistory

LocalDirHistory = Yes | **No**: The default value is **No**, which forces all copies of the command processor to share the same directory history list. **Yes** keeps the list for each session separate. See [Directory History Window](#)^[23] for an explanation of local and global directory histories.

See other [Initialization Directives](#)^[93].

3.2.1.12 LocalFunctions

LocalFunctions = Yes | **No**: The default value is **No**, which forces all copies of the command processor to share the same function list. **Yes** keeps the lists for each shell separate. See [FUNCTION](#)^[218] for more details on local and global function lists.

See other [Initialization Directives](#)^[93].

3.2.1.13 LocalHistory

LocalHistory = Yes | **No**: The default value is **No**, which forces all copies of the command processor to share the same history list. **Yes** keeps the lists for each shell separate. See [Local and Global History Lists](#)^[16] for more details on local and global history lists.

See other [Initialization Directives](#)^[93].

3.2.1.14 PauseOnError

PauseOnError = YES | No: **(4NT) 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](#)^[89]. **No** continues processing with no pause after an error message is displayed.

See other [Initialization Directives](#)^[93].

3.2.1.15 ScreenBufSize

ScreenBufSize = nnnn (200,000): **(TC)** Sets the size of the screen scrollbar buffer (in characters). The allowable range is from 50,000 to 1,000,000.

See other [Initialization Directives](#)^[93].

3.2.1.16 TCStartPath

TCStartPath = Path: **(TC)** Sets the drive and directory where TC finds the [TCSTART and TCEXIT](#)^[6] batch files (if any).

See other [Initialization Directives](#)^[93].

3.2.1.17 TimeServer

TimeServer = name: Specifies the URL of the internet time server to use for TIME /S. If no server is specified, the [TIME](#)^[302] command uses "clock.psu.edu".

Note: Not all "time servers" will respond properly to the **NTP** ("Network Time Protocol") query and it's usually best to use the default. Specifically, Microsoft's "**time.windows.com**" server used as default by Windows XP's "Internet Time" utility is typically not a suitable substitute. Some servers which seem to provide correct results as of this writing include (in no particular order) *time.nist.gov*, *finch.cc.ukans.edu*, *ntp.css.gov*, *ntp.lth.se*, *ntp.maths.tcd.ie*, *ntp0.cornell.edu*, *ntp-1.ece.cmu.edu*, *ntp-2.ece.cmu.edu*, *ntp2a.mcc.ac.uk*, *Rolex.PeachNet.EDU*, *salmon.maths.tcd.ie*, *sundial.columbia.edu*, *tick.wustl.edu*, *time.nrc.ca*, *timelord.uregina.ca*, *timex.cs.columbia.edu*, *Timex.PeachNet.EDU*.

See other [Initialization Directives](#)^[93].

3.2.1.18 TreePath

TreePath = Path: Sets the location of JPSTREE.IDX, the file used for the [extended directory search](#)^[56] database. By default, the file is placed in the root directory of drive C:\.

See other [Initialization Directives](#)^[93].

3.2.1.19 UnicodeOutput

UnicodeOutput = NO | Yes: Disable / enable Unicode output for redirection. This option will be overridden by a /U or /A startup [command line option](#)^[4].

3.2.1.20 WindowState

WindowState = STANDARD | Maximize | Minimize | Custom: Sets the initial state of the command processor 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](#)^[97] directives.

See other [Initialization Directives](#)^[93].

3.2.1.21 WindowX, WindowY, WindowWidth, WindowHeight

WindowX = nnnn
WindowY = nnnn
WindowWidth = nnnn
WindowHeight = nnnn

These 4 directives set the initial size and position of the command processor 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](#)^[97] is set to **Custom**.

See other [Initialization Directives](#)^[93].

3.2.2 Configuration Directives

These directives control the way that the command processor operates. Some can be changed with the [SETDOS](#)^[283] command while the command processor is running. Any corresponding SETDOS

command is listed in the description of each directive. The configuration directives are:

AmPm ^[99]	Time display format
ANSI ^[99]	Enables ANSI X3.64 support
AppendToDir ^[99]	Appends trailing "\" to directory names in filename completion
BatchEcho ^[100]	Default echo state for batch files
BeepFreq ^[100]	Default beep frequency
BeepLength ^[100]	Default beep length
CDDWinLeft, CDDWinTop, CDDWinWidth, CDDWinHeight ^[100]	Initial size and position of the extended directory search window
CommandSep ^[100]	Multiple command separator character
CompleteHidden ^[101]	Enable / disable return of hidden files from filename completion
CopyPrompt ^[101]	Enable or disable confirmation prompt for COPY and MOVE
CUA ^[101]	(TC) Key set used for cut, copy, and paste
CursorIns ^[101]	Cursor width in insert mode
CursorOver ^[101]	Cursor width in overstrike mode
DelGlobalQuery ^[102]	Enable or disable confirmation prompt with DEL /Q
DecimalChar ^[102]	Select decimal separator for @EVAL, etc.
DescriptionMax ^[102]	Maximum length of file descriptions
DescriptionName ^[102]	Name of file to hold file descriptions
Descriptions ^[102]	Enable / disable description processing
EditMode ^[102]	Editing mode (insert / overstrike)
Editor ^[103]	(TC) Program to run for "Editor" menu choice
EscapeChar ^[103]	Select escape character.
EvalMax ^[103]	Maximum precision displayed by @EVAL
EvalMin ^[103]	Minimum precision displayed by @EVAL
ExecWait ^[103]	Forces the command processor to wait for external programs to complete
FileCompletion ^[103]	Select files selected for file completion
FirewallHost ^[104]	Firewall server name
FirewallPassword ^[104]	Password for firewall authentication
FirewallType ^[104]	Type of firewall
FirewallUser ^[104]	Username for firewall authentication
FTPCFG ^[104]	Specifies the location of the file containing the ftp user names and passwords
FuzzyCD ^[104]	Enables or disables extended directory searches
HistCopy ^[105]	Copy recalled commands to end of history
HistDups ^[105]	Controls entry/retention of duplicate history entries
HistLogName ^[105]	History log file name
HistLogOn ^[105]	Turns on history logging when the command processor starts
HistMin ^[105]	Minimum command length to save
HistMove ^[105]	Move recalled commands to end of history
HistWrap ^[106]	Behavior of the command history list
ListRowStart ^[106]	Starting row number for LIST and FFIND
LogErrors ^[106]	Send error messages to the log file
LogName ^[106]	Name the log file
LogOn ^[106]	Turns on command logging when the command processor starts
MailAddress ^[106]	Email address of user
MailPassword ^[107]	Password for SMTP authentication
MailPort ^[107]	SMTP port number
MailServer ^[107]	Your SMTP server name
MailUser ^[107]	User name for SMTP authentication
NoClobber ^[107]	Overwrite protection for output redirection
ParameterChar ^[107]	Alias / batch file parameter character
PassiveFTP ^[108]	Enable or disable passive mode for FTP calls
PathExt ^[108]	Enable or disable the PATHEXT variable

PopupWinLeft, PopupWinTop, PopupWinWidth, PopupWinHeight ^[108]	Initial size and position of popup windows
Proxy ^[109]	Sets name of HTTP Proxy server
ProxyPassword ^[109]	HTTP proxy password for Basic authentication
ProxyPort ^[109]	Sets port number of HTTP Proxy server
ProxyUser ^[109]	HTTP proxy user name for Basic authentication
RecycleBin ^[109]	Sets whether deleted files go to the Recycle Bin
ScreenColumns ^[109]	(TC) Virtual screen width
ScreenRows ^[109]	(TC) Virtual screen height
ServerCompletion ^[109]	Sets server name completion options
SSLPort ^[110]	Sets the port number for FTP SLL service
SSLProvider ^[110]	Sets the name of the security provider
SSLStartMode ^[110]	Specifies how to start SLL negotiation
StatusBarText ^[110]	(TC) Point size of status bar text
StatusBarOn ^[110]	(TC) Set status bar mode at startup
SwapScrollKeys ^[111]	(TC) Switch to 4NT-style history and scrolling keys
TabStops ^[111]	Sets the tab positions for output.
ThousandsChar ^[111]	Thousands separator for @EVAL, etc.
ToolBarOn ^[112]	(TC) Set tool bar mode at startup
ToolBarText ^[112]	(TC) Point size of tool bar text
UnixPaths ^[112]	Enable or disable forward slash in command paths
UpdateTitle ^[112]	Prevents or allows updating of the command processor window title
Win32SFNSearch ^[112]	Search for short and long file names

3.2.2.1 AmPm

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 displays used by [DIR](#)^[179] and [SELECT](#)^[275], in [LOG](#)^[242] files, and the output of the [TIMER](#)^[303], [DATE](#)^[170], and [TIME](#)^[302] commands. It has no effect on [% TIME](#)^[356], [%@MAKETIME](#)^[400], the \$t and \$T options of [PROMPT](#)^[262], or date and time [ranges](#)^[40].

See other [Configuration Directives](#)^[97].

3.2.2.2 ANSI

ANSI = Yes | NO: Sets the initial state of [ANSI X3.64 support](#)^[64]. **Yes** enables ANSI X3.64 string processing of the outputs of 4NT / Take Command internal commands. **No** disables it. Note that ANSI X3.64 processing of the output of **external applications** is not available. Caveman output in Take Command will use the ANSI X3.64 default colors if the application is set to use "default colors" or "stdio". See the [ANSI X3.64 Commands Reference](#)^[456] for a list of the ANSI X3.64 commands supported by 4NT and Take Command. Also see the [_ANSI](#)^[348] internal variable.

See other [Configuration Directives](#)^[97].

3.2.2.3 AppendToDir

AppendToDir = Yes | NO: Yes appends a trailing "\" to directory names (or a trailing "/" to FTP URLs) when doing filename completion. (If you also have the [UnixPaths](#)^[112] .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](#)^[22].

See other [Configuration Directives](#)^[97].

3.2.2.4 BatchEcho

BatchEcho = YES | No: Sets the default batch echo mode. **Yes** enables echoing of all batch file commands unless [ECHO](#)^[198] is explicitly set off in the batch file. **No** disables batch file echoing unless ECHO is explicitly set on. Also see [SETDOS /V](#)^[283].

See other [Configuration Directives](#)^[97].

3.2.2.5 BeepFreq

BeepFreq = nnnn (440): Sets the default [BEEP](#)^[156] command frequency in cycles per second (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](#)^[100] 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.

This option is ignored in Windows 98 and ME (see the [BEEP](#)^[156] command for details).

See other [Configuration Directives](#)^[97].

3.2.2.6 BeepLength

BeepLength = nnnn (2): Sets the default [BEEP](#)^[156] 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). Also see [BeepFreq](#)^[100].

This option is ignored in Windows 98 and ME (see the [BEEP](#)^[156] command for details).

See other [Configuration Directives](#)^[97].

3.2.2.7 CDDWinLeft, CDDWinTop, CDDWinWidth, CDDWinHeight

CDDWinLeft = nnnn (3)
CDDWinTop = nnnn (3)
CDDWinWidth = nnnn (72)
CDDWinHeight = nnnn (16)

These values set the initial position and size of the popup window used by [extended directory searches](#)^[56], in characters, including the border. The defaults are shown above in parentheses, and specify 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 command processor 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.

See other [Configuration Directives](#)^[97].

3.2.2.8 CommandSep

CommandSep = c: This is the character used to separate multiple commands on the same line. It defaults to the ampersand [&]. You cannot use any of the [redirection](#)^[67] characters (| > <) or any of

the white space characters (space, tab, comma, or equal sign). The command separator is saved by [SETLOCAL](#)^[287] and restored by [ENDLOCAL](#)^[200]. Also see [SETDOS](#)^[283] /C. See the [%+](#)^[347] internal variable and the section on [Special Character Compatibility](#)^[335] for information on using compatible command separators for two or more products.

See other [Configuration Directives](#)^[97].

3.2.2.9 CompleteHidden

CompleteHidden = Yes | NO: **Yes** includes hidden and system files and directories in the list of names that can be returned for [filename completion](#)^[17].

See other [Configuration Directives](#)^[97].

3.2.2.10 CopyPrompt

CopyPrompt = Yes|NO: If set to **Yes**, the command processor will prompt in [COPY](#)^[164] and [MOVE](#)^[246] before overwriting an existing file if the command is being performed at the command prompt. This duplicates the behavior of later versions of CMD.EXE.

See other [Configuration Directives](#)^[97].

3.2.2.11 CUA

CUA = NO | Yes: **(TC)** With the default setting of "No", Take Command will use the command Windows (non-CUA) editing keys: **Ctrl-X** for cut, **Ctrl-C** for copy, and **Ctrl-V** for paste. 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.

Note: when CUA is set to "No", **Ctrl-C** is not available for interrupting commands. Use **Ctrl-Break** instead.

See other [Configuration Directives](#)^[97].

3.2.2.12 CursorIns

CursorIns = nnnn (15 **(TC)** 100 **(4NT)**): 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 you set CursorIns and CursorOver to -1, the cursor shape won't be modified at all. If you set them to 0, the cursor will be invisible. Also see [CursorOver](#)^[101], [SETDOS/S](#)^[283].

See other [Configuration Directives](#)^[97].

3.2.2.13 CursorOver

CursorOver = nnnn (100 **(TC)** 15 **(4NT)**): 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%. For more details see the [CursorIns](#)^[101] directive; also see [SETDOS](#)^[283] /S.

See other [Configuration Directives](#)^[97].

3.2.2.14 DecimalChar

DecimalChar = . | , | AUTO: Sets the character used as the decimal separator for [@EVAL](#)^[374], 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](#)^[111]) so that the two characters are different.** See also: [SETDOS /G](#)^[283].

See other [Configuration Directives](#)^[97].

3.2.2.15 DelGlobalQuery

DelGlobalQuery = YES | No: Enable or disable the confirmation prompt from DEL /Q when doing a wildcard-only or directory deletion. Use caution if you set DelGlobalQuery to No, as this will allow DEL /Q to delete an entire directory without prompting for confirmation. See [DEL](#)^[172] for additional details.

See other [Configuration Directives](#)^[97].

3.2.2.16 DescriptionMax

DescriptionMax = nnnn (511): Controls the description length limit for [DESCRIBE](#)^[176]. The allowable range is 20 to 511 characters.

See other [Configuration Directives](#)^[97].

3.2.2.17 DescriptionName

DescriptionName = File: Sets the file name in which to store file descriptions. The default file name is DESCRIPT.ION. Also see [SETDOS](#)^[283] /D.

See other [Configuration Directives](#)^[97].

3.2.2.18 Descriptions

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](#)^[283] /D.

See other [Configuration Directives](#)^[97].

3.2.2.19 EditMode

EditMode = Insert | Overstrike | InitOverstrike | InitInsert: This directive lets you start the command-line editor in either **Insert** (TC default) or **Overstrike** (4NT default) 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](#)^[283] /M.

EditMode defaults to Insert in Take Command, and to Overstrike in 4NT.

See other [Configuration Directives](#)^[97].

3.2.2.20 Editor

Editor = File: (TC) Specifies the path and filename of the program that Take Command will execute when you select "Editor" from the [Utilities menu](#)^[70]. The default is NOTEPAD.EXE.

See other [Configuration Directives](#)^[97].

3.2.2.21 EscapeChar

EscapeChar = c : Sets the character used to suppress the normal meaning of the following character. The default is a caret [^]. See [Escape Character](#)^[21] for a description of the special escape sequences. You cannot use any of the [redirection](#)^[61] characters (|, >, or <) or the white space characters (space, tab, comma, or equal sign) as the escape character. The escape character is saved by [SETLOCAL](#)^[287] and restored by [ENDLOCAL](#)^[200]. Also see [SETDOS](#)^[283] /E. See the [%=](#)^[347] internal variable and the section on [Special Character Compatibility](#)^[335] for information on using compatible escape characters for two or more products.

See other [Configuration Directives](#)^[97].

3.2.2.22 EvalMax

EvalMax = nnnn (10): Controls the maximum number of digits after the decimal point in values displayed by [@EVAL](#)^[374]. You can override this setting with the construct @EVAL[expression=n,n]. The allowable range is 0 to 10. Also see [SETDOS](#)^[283] /F.
See other [Configuration Directives](#)^[97].

3.2.2.23 EvalMin

EvalMin = nnnn (0): Controls the minimum number of digits after the decimal point in values displayed by [@EVAL](#)^[374]. The allowable range is 0 to 10. This directive will be ignored if EvalMin is larger than EvalMax. You can override this setting with the construct @EVAL[expression=n,n]. Also see [SETDOS](#)^[283] /F.

See other [Configuration Directives](#)^[97].

3.2.2.24 ExecWait

ExecWait = Yes | NO: Controls whether the command processor waits for an external program started from the command line to complete before redisplaying the prompt. See [Waiting for Applications to Finish](#)^[29] for details on the effects of this directive.

See other [Configuration Directives](#)^[97].

3.2.2.25 FileCompletion (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](#)^[340] environment variable. See [Customizing Filename Completion](#)^[19] for a detailed explanation of selective filename completion.

See other [Configuration Directives](#)^[97].

3.2.2.26 FirewallHost

FirewallHost = name: The server name of the firewall, if any, to use for FTP and HTTP access.

See [FirewallPassword](#)^[104], [FirewallType](#)^[104], [FirewallUser](#)^[104] and other [Configuration Directives](#)^[97].

3.2.2.27 FirewallPassword

FirewallPassword = name: The password for [FirewallUser](#)^[104] if required for authentication by the firewall.

Note: This directive stores the password in **plain text**, making it accessible to anyone who can read your [INI file](#)^[89].

See [FirewallHost](#)^[104], [FirewallType](#)^[104] and other [Configuration Directives](#)^[97].

3.2.2.28 FirewallType

FirewallType = [0 | 1 | 2 | 3]: The type of the firewall:

- 0 None (default)
- 1 Tunnel
- 2 SOCKS4
- 3 SOCKS5

See [FirewallPassword](#)^[104], [FirewallHost](#)^[104], [FirewallUser](#)^[104] and other [Configuration Directives](#)^[97].

3.2.2.29 FirewallUser

FirewallUser = name: The user name if the firewall requires authentication. A password can be defined via the [FireWallPassword](#)^[104] directive

See [FirewallHost](#)^[104], [FirewallType](#)^[104] and other [Configuration Directives](#)^[97].

3.2.2.30 FTPCFG

FTPCFG = filename. Specify the location and name of the file containing the FTP user names and passwords. The default is "FTP.CFG" in the 4NT or Take Command directory.

We recommend you encrypt this file if you're using NTFS. If FTP.CFG doesn't exist the first time 4NT or Take Command looks for it, it will be created as an encrypted file (NTFS only). If you are using Win98 or FAT/ VFAT, the file will be in plain unencrypted text.

See "[Using FTP/HTTP Servers](#)^[52]" for details.

3.2.2.31 FuzzyCD (Extended Directory Search)

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](#)^[56].

See other [Configuration Directives](#)^[97].

3.2.2.32 HistCopy

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](#)^[105] = **Yes**; otherwise, the HistCopy setting will override HistMove.

See other [Configuration Directives](#)^[97].

3.2.2.33 HistDups

HistDups = OFF | First | Last: Controls duplicate entry placement in the [history list](#)^[13].

Off	Add new entry unconditionally.
First	Add new entry only if it does not match any old entries
Last	Add new entry unconditionally. Delete matching older entries.

See other [Configuration Directives](#)^[97].

3.2.2.34 HistLogName

HistLogName = File: Sets the history log file name and path. If this directive is not used, the default name of **TCHLOG (TC)** or **4NTHLOG (4NT)** in the root directory of the boot drive will be used. Using HistLogName does not turn history logging on. To do so, you must use a [LOG](#)^[242] /H ON command or the [HistLogOn](#)^[105] directive. Also see [LogName](#)^[105].

See other [Configuration Directives](#)^[97].

3.2.2.35 HistLogOn

HistLogOn = yes | NO: If set to **Yes**, [history logging](#)^[242] is turned on when the command processor starts.

See other [Configuration Directives](#)^[97].

3.2.2.36 HistMin

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 **1,024**. You can prevent any command line from being saved in the history by beginning it with an "at" sign ("@"). Also see: [HISTORY](#)^[13] command.

See other [Configuration Directives](#)^[97].

3.2.2.37 HistMove

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](#)^[105] 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**.

See other [Configuration Directives](#)^[97].

3.2.2.38 HistWrap

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.

See other [Configuration Directives](#)^[97].

3.2.2.39 ListRowStart

ListRowStart = 1 | 0: Specifies whether [LIST](#)^[237] and [FFIND](#)^[206] consider the first line in a file line **1** or line **0**. The default is **1**.

See other [Configuration Directives](#)^[97].

3.2.2.40 LogErrors

LogErrors = yes | NO: If set to **Yes**, error messages will be written to the log file. See the [LOG](#)^[242] command for additional information.

See other [Configuration Directives](#)^[97].

3.2.2.41 LogName

LogName = File: Sets the log file name and path. If this directive is not used, the default name of **TCLOG (TC)** or **4NTLOG (4NT)** in the root directory of the boot drive will be used. Using LogName does not turn logging on; you must use a [LOG](#)^[242] ON command or the [LogOn](#)^[106] directive to do so. Also see [HistLogName](#)^[105].

See other [Configuration Directives](#)^[97].

3.2.2.42 LogOn

LogOn = yes | NO: If set to **Yes**, [command logging](#)^[242] is turned on when the command processor starts.

See other [Configuration Directives](#)^[97].

3.2.2.43 MailAddress

MailAddress = name: The email address of the current user, used in [SENDMAIL](#)^[279] for outgoing mail. If not set, SENDMAIL will attempt to get the address from the registry. This value is what will generally be transmitted in the "From:" header of messages you send.

See [MailPassword](#)^[107], [MailPort](#)^[107], [MailServer](#)^[107], [MailUser](#)^[107] and other [Configuration Directives](#)^[97].

3.2.2.44 MailPassword

MailPassword = string: The email password of the current user ([MailUser](#)^[107]), used in [SENDMAIL](#)^[279] for outgoing mail if the mail server requires it. If not set, SENDMAIL will attempt to get the password from the registry.

Note: This directive stores the password in **plain text**, making it accessible to anyone who can read your [INI file](#)^[89].

See [MailAddress](#)^[106], [MailPort](#)^[107], [MailServer](#)^[107], [MailUser](#)^[107] and other [Configuration Directives](#)^[97].

3.2.2.45 MailPort

MailPort = n: The SMTP port number (the default is 25) for use by [SENDMAIL](#)^[279].

See [MailAddress](#)^[106], [MailPassword](#)^[107], [MailServer](#)^[107], [MailUser](#)^[107] and other [Configuration Directives](#)^[97].

3.2.2.46 MailServer

MailServer =name: The local SMTP server name to use in [SENDMAIL](#)^[279] for outgoing mail. If not set, SENDMAIL will attempt to get the address from the registry.

See [MailAddress](#)^[106], [MailPassword](#)^[107], [MailPort](#)^[107], [MailUser](#)^[107] and other [Configuration Directives](#)^[97].

3.2.2.47 MailUser

MailUser = name: The email username of the current [SENDMAIL](#)^[279] user if your [SMTP server](#)^[107] requires it for authentication. If a password is also required, it can be defined by the [MailPassword](#)^[107] directive

See [MailAddress](#)^[106], [MailPassword](#)^[107], [MailPort](#)^[107], [MailServer](#)^[107] and other [Configuration Directives](#)^[97].

3.2.2.48 NoClobber

NoClobber = Yes | NO: If set to **Yes**, will prevent standard output [redirection](#)^[61] from overwriting an existing file, and will require that the output file already exist for append redirection. Also see [SETDOS](#)^[283] /N.

See other [Configuration Directives](#)^[97].

3.2.2.49 ParameterChar

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 [Batch File Parameters](#)^[322] and [ALIAS](#)^[138]). The default is the dollar sign [\$]. The parameter character is saved by SETLOCAL and restored by ENDLOCAL. Also see [SETDOS](#)^[283] /P. See [Special Character Compatibility](#)^[335] for information on using compatible parameter characters for two or more products.

See other [Configuration Directives](#)^[97].

3.2.2.50 PassiveFTP

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.

This setting applies to both the [IFTP](#)^[229] command and standalone [ftp references](#)^[62]. It can be temporarily changed when desired with the [OPTION](#)^[253] command:

```
OPTION //passiveftp=no
IFTP ...
...
OPTION //passiveftp=yes
```

See other [Configuration Directives](#)^[97].

3.2.2.51 PathExt (directive)

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](#)^[341] variable and the [PATH](#)^[254] command.

Caution: If you set PathExt = Yes in the .INI file and then fail to set the PATHEXT variable, path searches will fail as there will be no extensions for which to search!

PATHEXT is supported for compatibility reasons but should not generally be used as a substitute for the more flexible [executable extensions](#)^[48] feature.

See other [Configuration Directives](#)^[97].

3.2.2.52 PopupWinLeft, PopupWinTop, PopupWinWidth, PopupWinHeight

PopupWinLeft = nnnn (40)
PopupWinTop = nnnn (1)
PopupWinWidth = nnnn (36)
PopupWinHeight = nnnn (12)

These values set the initial position and size of the command-line and directory history windows, in characters, including the border. The defaults shown above in parentheses specify 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 command processor 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.

See other [Configuration Directives](#)^[97].

3.2.2.53 Proxy

Proxy = name: Specifies the proxy server address to use for HTTP calls.

See [ProxyPassword](#)^[109], [ProxyPort](#)^[109], [ProxyUser](#)^[109] and other [Configuration Directives](#)^[97].

3.2.2.54 ProxyPassword

ProxyPassword = password : Password for HTTP proxy if Basic authentication is to be used.

See [Proxy](#)^[109], [ProxyPort](#)^[109], [ProxyUser](#)^[109] and other [Configuration Directives](#)^[97].

3.2.2.55 ProxyPort

ProxyPort = port: Specifies the port number to use for HTTP calls.

See [Proxy](#)^[109], [ProxyPassword](#)^[109], [ProxyUser](#)^[109] and other [Configuration Directives](#)^[97].

3.2.2.56 ProxyUser

ProxyUser = name. A user name if Basic authentication is to be used for the HTTP proxy.

See [Proxy](#)^[109], [ProxyPassword](#)^[109], [ProxyPort](#)^[109] [Configuration Directives](#)^[97].

3.2.2.57 RecycleBin

RecycleBin = Yes | NO: If set to **Yes**, files deleted by the [DEL/ERASE](#)^[172] commands and by [RD/S](#)^[267] 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. The [RecycleExclude](#)^[341] internal variable can be used to exclude specific files.

See other [Configuration Directives](#)^[97].

3.2.2.58 ScreenColumns

ScreenColumns = nnnn: **(TC)** 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 [Resizing the Take Command Window](#)^[77] for more information on the virtual screen size.

See other [Configuration Directives](#)^[97].

3.2.2.59 ScreenRows

ScreenRows = nnnn (25): **(TC)** Sets the initial height of the Take Command window. See [Resizing the Take Command Window](#)^[77] for more information on screen size.

See other [Configuration Directives](#)^[97].

3.2.2.60 ServerCompletion

ServerCompletion = None | LOCAL | Global. Specifies how server name completion should proceed (see [Filename Completion](#)^[17] 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.

See other [Configuration Directives](#)^[97].

3.2.2.61 SSLPort

SSLPort = port The port number for the **FTP SSL** service. The default is **21**. For **implicit** SSL, use port **990**.

See [SSLProvider](#)^[110], [SSLStartMode](#)^[110] and other [Configuration Directives](#)^[97]

3.2.2.62 SSLProvider

SSLProvider = string - The name of the security provider to use for SSL authentication. The default value is "*", which selects the default SSL provider defined in the system. In current Windows implementations, this is "Microsoft Unified Security Protocol Provider".

See [SSLPort](#)^[110], [SSLStartMode](#)^[110] and other [Configuration Directives](#)^[97]

3.2.2.63 SSLStartMode

SSLStartMode = n - Specify how to start SSL negotiation:

- 0** (default) If the remote port is set to the standard plaintext port of the protocol, SSL will behave the same as if SSLStartMode is set to 2 (sslExplicit). In all other cases, SSL negotiation will be implicit (sslImplicit).
- 1** (sslImplicit) The SSL negotiation will start immediately after the connection is established.
- 2** (sslExplicit) Connection will first be in plaintext, and then SSL negotiation will be explicitly started through a protocol command such as STARTTLS.
- 3** (sslNone) No SSL negotiation, no SSL security. All communication will be in plaintext mode.

See [SSLPort](#)^[110], [SSLProvider](#)^[110] and other [Configuration Directives](#)^[97]

3.2.2.64 StatusBarText

StatusBarText = nnnn (8): **(TC)** Sets the size of the text on the status bar, in points. The allowable range is 4 to 16. Correct operation is not guaranteed for other values.

See other [Configuration Directives](#)^[97].

3.2.2.65 StatusBarOn

StatusBarOn = YES | No: **(TC) 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 menu](#)^[70]. The StatusBarOn setting is automatically updated to reflect the current state of the

status bar each time Take Command exits; this preserves the status bar state between sessions.

See other [Configuration Directives](#)^[97].

3.2.2.66 SwapScrollKeys

SwapScrollKeys = Yes | NO: **(TC)** Yes switches to 4NT-style keystrokes for manipulating the [scrollback buffer](#)^[75].

If SwapScrollKeys is set to Yes, the Up and Down arrow keys will scroll through the [command history](#)^[13] list and the PgUp key will pop up the [history window](#)^[15]. 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](#)^[25].

Note: Do not set SwapScrollKeys to Yes if you use [key mapping directives](#)^[114] 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 key mapping directives in the [.INI file](#)^[89] individually:

```
PrevHistory = Up
NextHistory = Down
HistWinOpen = PgUp
HistWinOpen = PgDn
ScrollUp = Ctrl-Up
ScrollDown = Ctrl-Down
ScrollPgUp = Ctrl-PgUp
ScrollPgDn = Ctrl-PgDn
```

See other [Configuration Directives](#)^[97].

3.2.2.67 TabStops

TabStops = nnnn (8): **(TC)** Sets the tab stops for the command processor's output (including the output from the [LIST](#)^[237] and [TYPE](#)^[308] commands). **(4NT)** Sets the tab stops for [LIST](#)^[237] output. The allowable range is 1 to 32.

See other [Configuration Directives](#)^[97].

3.2.2.68 ThousandsChar

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](#)^[102]) so that the two characters are different. See also: [SETDOS /G](#)^[283].

See other [Configuration Directives](#)^[97].

3.2.2.69 ToolBarOn

ToolBarOn = YES | No: **(TC) 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 menu](#)^[70]. The ToolBarOn setting is automatically updated to reflect the current state of the tool bar each time Take Command exits; this preserves the tool bar state between sessions.

See other [Configuration Directives](#)^[97].

3.2.2.70 ToolBarText

ToolBarText = nnnn (8): **(TC)** Sets the point size of text on the tool bar. The allowable range is 4 to 16.

See other [Configuration Directives](#)^[97].

3.2.2.71 UnixPaths

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
```

See other [Configuration Directives](#)^[97].

3.2.2.72 UpdateTitle

UpdateTitle = YES | No: The command processor normally changes the title in its 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.

See other [Configuration Directives](#)^[97].

3.2.2.73 Win32SFNSearch

Win32SFNSearch = YES | No: Set to **No** to disable the automatic search for short filenames after long filenames. See [LFN File Searches](#)^[47] for details.

See other [Configuration Directives](#)^[97].

3.2.3 Color Directives

These directives control the colors that the command processor uses for its displays. For complete details on color names and numbers, see [Colors and Color Names](#)^[46]. The color directives are:

CDDWinColors ^[113]	Colors for directory search window.
ColorDir ^[113]	Directory colors
InputColors ^[113]	Input colors
ListboxBarColors ^[113]	(4NT) Colors of popup list boxes.
ListColors ^[113]	Colors used in the LIST display
ListStatBarColors ^[114]	(4NT) Color for LIST status bar
PopupWinColors ^[114]	(TC) Colors for most popup windows
SelectColors ^[114]	Colors used in the SELECT display
SelectStatBarColors ^[114]	(4NT) Color used by SELECT status bar
StdColors ^[114]	Standard display colors

3.2.3.1 CDDWinColors

CDDWinColors = Color: Sets the default colors for the popup window used by [extended directory searches](#)^[56]. If this directive is not used, the colors will be reversed from the current colors on the screen.

See other [Color Directives](#)^[113].

3.2.3.2 ColorDIR (directive)

ColorDir = ext1 ext2 ...:colora;ext3 ext4 ... :colorb; ...: Sets the directory colors used by [DIR](#)^[179] and [SELECT](#)^[275] **(4NT)**. The format is the same as that used for the COLORDIR environment variable. See [Color-Coded Directories](#)^[179] for a detailed explanation.

See other [Color Directives](#)^[113].

3.2.3.3 InputColors

InputColors = Color: Sets the colors used for command-line input. This setting is useful for making your input stand out from the normal output.

See other [Color Directives](#)^[113].

3.2.3.4 ListboxBarColors

ListboxBarColors = Color: **(4NT)** Sets the color for the highlight bar in the popup list boxes (i.e., [command history window](#)^[15], [filename completion window](#)^[21], [@SELECT window](#)^[407], etc.).

See other [Color Directives](#)^[113].

3.2.3.5 ListColors

ListColors = Color: Sets the colors used by the [LIST](#)^[237] command. If this directive is not used, LIST will use the current default colors set by the [CLS](#)^[163] or [COLOR](#)^[163] command or by the [StdColors](#)^[114] directive.

See other [Color Directives](#)^[113].

3.2.3.6 ListStatBarColors

ListStatBarColors = Color: **(4NT)** Sets the colors used on the [LIST](#)^[237] 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](#)^[113]).

See other [Color Directives](#)^[113].

3.2.3.7 PopupWinColors

PopupWinColors = Color: **(TC)** Sets the default colors for the command-line, directory history, and filename completion windows, and most other popup windows (see [CDDWinColors](#)^[113] for the extended directory search window). If this directive is not used, the colors will be reversed from the current colors on the screen.

See other [Color Directives](#)^[113].

3.2.3.8 SelectColors

SelectColors = Color: Sets the colors used by the [SELECT](#)^[275] command. If this directive is not used, SELECT will use the current default colors set by the [CLS](#)^[163] or [COLOR](#)^[163] command or by the [StdColors](#)^[114] directive.

See other [Color Directives](#)^[113].

3.2.3.9 SelectStatBarColors

SelectStatBarColors = Color: **(4NT)** Sets the color used on the [SELECT](#)^[275] 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](#)^[114]).

See other [Color Directives](#)^[113].

3.2.3.10 StdColors

StdColors = Color: **(4NT)** Sets the standard colors to be used when [CLS](#)^[163] is used without a color specification. Using this directive is similar to placing a [COLOR](#)^[163] command in the [4START](#)^[6] file. With 4NT under Windows 98 and ME, if ANSI.SYS or a compatible driver is not loaded in CONFIG.SYS the colors will not be "sticky", i.e. you may lose them when you run an application.

StdColors = Color: **(TC)** Sets the standard colors to be used when [CLS](#)^[163] is used without a color specification. Using this directive is similar to placing a [COLOR](#)^[163] command in the [TCSTART](#)^[6] file.

See other [Color Directives](#)^[113].

3.2.4 Key Mapping Directives

These directives allow you to change the keys used for command-line editing and other internal functions. They cannot be entered via the [configuration dialogs](#)^[82]. You must enter them manually (see the [.INI file](#)^[89] topic for details).

They are divided into four types, depending on the context in which the keys are used. For a discussion

and list of directives for each type see:

- ▶ [General Input Keys](#)^[115]
- ▶ [Command-Line Editing Keys](#)^[118] and [Scrollbar Buffer Keys \(TC\)](#)^[121]
- ▶ [Popup Window Keys](#)^[122]
- ▶ [LIST Keys](#)^[123]

Using a key mapping 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.

See [Keys and Key Names](#)^[464] 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 de-assign the default keystroke for the same function. If you want to de-assign one of the default keys, use the [NormalKey](#)^[117], [NormalEditKey](#)^[120], [NormalPopupKey](#)^[122] or [NormalListKey](#)^[125] directive.

Note: if you assign the same key to two different functions, the **first** assignment found in the list will be used.

3.2.4.1 General Input Keys

These directives apply to all input. They are in effect whenever the command processor requests input from the keyboard, including during [command-line editing](#)^[11] and the [DESCRIBE](#)^[176], [ESET](#)^[201], [INPUT](#)^[233], [LIST](#)^[237], and [SELECT](#)^[275] commands. The general input keys are:

Backspace ^[116]	Deletes the character to the left of the cursor
BeginLine ^[116]	Moves the cursor to the start of the line
Copy ^[116]	Copies highlighted text to the keyboard
Del ^[116]	Deletes the character at the cursor
DelToBeginning ^[116]	Deletes from the cursor to the start of the line
DelToEnd ^[116]	Deletes from the cursor to the end of the line
DelWordLeft ^[116]	Deletes the word to the left of the cursor
DelWordRight ^[117]	Deletes the word to the right of the cursor
Down	Moves the cursor or scrolls the display down
EndLine ^[117]	Moves the cursor to the end of the line
EraseLine ^[117]	Deletes the entire line
ExecLine ^[117]	Executes or accepts a line
Ins ^[117]	Toggles insert / overstrike mode
Left ^[117]	Moves the cursor or scrolls the display left
NormalKey ^[117]	Deassigns a key

Paste 	Pastes line from clipboard
Right 	Moves the cursor or scrolls the display right
Up 	Moves the cursor or scrolls the display up
WordLeft 	Moves the cursor left one word
WordRight 	Moves the cursor right one word

3.2.4.1.1 Backspace

Backspace = Key (Bksp): Deletes the character to the left of the cursor.

See other [General Input Keys](#) .

3.2.4.1.2 BeginLine

BeginLine = Key (Home): Moves the cursor to the beginning of the line.

See other [General Input Keys](#) .

3.2.4.1.3 Copy (directive)

Copy = Key (Ctrl-Y): Copy the highlighted text to the clipboard.

See other [General Input Keys](#) .

3.2.4.1.4 Del (directive)

Del = Key (Del): Deletes the character at the cursor.

See other [General Input Keys](#) .

3.2.4.1.5 DelToBeginning

DelToBeginning = Key (Ctrl-Home): Deletes from the cursor to the start of the line.

See other [General Input Keys](#) .

3.2.4.1.6 DelToEnd

DelToEnd = Key (Ctrl-End): Deletes from the cursor to the end of the line.

See other [General Input Keys](#) .

3.2.4.1.7 DelWordLeft

DelWordLeft = Key (Ctrl-L): Deletes the word to the left of the cursor.

See other [General Input Keys](#) .

3.2.4.1.8 DelWordRight

DelWordRight = Key (Ctrl-R, Ctrl-Bksp): Deletes the word to the right of the cursor. See [ClearKeyMap](#)^[126] if you need to remove the default mapping of Ctrl-Bksp to this function.

See other [General Input Keys](#)^[115].

3.2.4.1.9 Down

Down = Key (Down): Scrolls the display down one line in LIST; moves the cursor down one line in [SELECT](#)^[275] and in the command-line history, directory history, or [@SELECT](#)^[407] window.

See other [General Input Keys](#)^[115].

3.2.4.1.10 EndLine

EndLine = Key (End): Moves the cursor to the end of the line.

See other [General Input Keys](#)^[115].

3.2.4.1.11 EraseLine

EraseLine = Key (Esc): Deletes the entire line.

See other [General Input Keys](#)^[115].

3.2.4.1.12 ExecLine

ExecLine = Key (Enter): Executes or accepts a line.

See other [General Input Keys](#)^[115].

3.2.4.1.13 Ins

Ins = Key (Ins): Toggles insert / overstrike mode during line editing.

See other [General Input Keys](#)^[115].

3.2.4.1.14 Left

Left = Key (Left): Moves the cursor left one character on the input line; scrolls the display left 8 columns in [LIST](#)^[237]; scrolls the display left 4 columns in the command-line, directory history, or [@SELECT](#)^[407] window.

See other [General Input Keys](#)^[115].

3.2.4.1.15 NormalKey

NormalKey = Key: Deassigns a general input key in order to disable the usual meaning of the key within the command processor 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](#)^[116] directive.

See other [General Input Keys](#)^[115].

3.2.4.1.16 Paste

Paste = Key (Ctrl-V): Paste the first line of the clipboard to the input line at the cursor position.

See other [General Input Keys](#)^[115].

3.2.4.1.17 Right

Right = Key (Right): Moves the cursor right one character on the input line; scrolls the display right 8 columns in [LIST](#)^[237]; scrolls the display right 4 columns in the command-line history, directory history, or [@SELECT](#)^[407] window.

See other [General Input Keys](#)^[115].

3.2.4.1.18 Up

Up = Key (Up): Scrolls the display up one line in [LIST](#)^[237]; moves the cursor up one line in [SELECT](#)^[275] and in the command-line history, directory history, or [@SELECT](#)^[407] window.

See other [General Input Keys](#)^[115].

3.2.4.1.19 WordLeft

WordLeft = Key (Ctrl-Left): Moves the cursor left one word; scrolls the display left 40 columns in [LIST](#)^[237].

See other [General Input Keys](#)^[115].

3.2.4.1.20 WordRight

WordRight = Key (Ctrl-Right): Moves the cursor right one word; scrolls the display right 40 columns in [LIST](#)^[237].

See other [General Input Keys](#)^[115].

3.2.4.2 Command-Line Editing Keys

These directives apply only to [command-line editing](#)^[11]. They are only effective at the prompt. The command-line editing keys are:

AddFile ^[119]	Keeps filename completion entry and adds another
AliasExpand ^[119]	Expands aliases on command line
CommandEscape ^[119]	Allows direct entry of a keystroke
DelHistory ^[119]	Deletes a history list entry

EndHistory ^[119]	Displays the last entry in the history list
Help ^[119]	Invokes this help system
HelpWord ^[120]	Invokes help for the word at the cursor
LFNToggle ^[120]	Toggles between long and short filenames
LineToEnd ^[120]	Copies a line to the end of the history, then executes it.
NextFile ^[120]	Gets the next matching filename
NextHistory ^[120]	Recalls the next command from the history
NormalEditKey ^[120]	Deassigns a command-line editing key
PopFile ^[120]	Opens the filename completion window
PrevFile ^[121]	Gets the previous matching filename
PrevHistory ^[121]	Recalls the previous command from the history
RepeatFile ^[121]	Repeats previous match during filename completion.
SaveHistory ^[121]	Saves the command line without executing it

3.2.4.2.1 AddFile

AddFile = Key (Ctrl-Shift-Tab (**TC**) or F10 (**4NT**)): Keeps the current filename completion entry and inserts the next matching name.

See other [Command-Line Editing Keys](#) ^[118].

3.2.4.2.2 AliasExpand

AliasExpand = Key (Ctrl-F): Expands all aliases in the current command line without executing them.

See other [Command-Line Editing Keys](#) ^[118].

3.2.4.2.3 CommandEscape

CommandEscape = Key (Alt-255): Allows direct entry of a keystroke that would normally be handled by the command line editor (e.g. Tab or Ctrl-D).

See other [Command-Line Editing Keys](#) ^[118].

3.2.4.2.4 DelHistory

DelHistory = Key (Ctrl-D): Deletes the displayed history list entry and displays the previous entry.

See other [Command-Line Editing Keys](#) ^[118].

3.2.4.2.5 EndHistory

EndHistory = Key (Ctrl-E): Displays the last entry in the history list.

See other [Command-Line Editing Keys](#) ^[118].

3.2.4.2.6 Help (directive)

Help = Key (F1): Displays the [Help File](#) ^[423] topic for the current command. Also see the [HELP](#) ^[225] command and the [HelpWord](#) ^[120] directive.

See other [Command-Line Editing Keys](#)^[118].

3.2.4.2.7 HelpWord

HelpWord = Key (Ctrl-F1): Invokes the [HELP](#)^[423] facility for the word at the cursor.

See other [Command-Line Editing Keys](#)^[118].

3.2.4.2.8 LFNToggle

LFNToggle = Key (Ctrl-A): Toggles filename completion between long filename and short filename modes on LFN drives.

See other [Command-Line Editing Keys](#)^[118].

3.2.4.2.9 LineToEnd

LineToEnd = Key (Ctrl-Enter): Copies the current command line to the end of the history list, then executes it.

See other [Command-Line Editing Keys](#)^[118].

3.2.4.2.10 NextFile

NextFile = Key (F9, Tab): Gets the next matching filename during filename completion. See [ClearKeyMap](#)^[126] if you need to remove the default mapping of Tab to this function.

See other [Command-Line Editing Keys](#)^[118].

3.2.4.2.11 NextHistory

NextHistory = Key (Down for **4NT**, Ctrl-Down for **TC**): Recalls the next command from the command history. (**TC**) Also see [Scrolling and History Keystrokes](#)^[25] and the [SwapScrollKeys](#)^[111] directive.

See other [Command-Line Editing Keys](#)^[118].

3.2.4.2.12 NormalEditKey

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. This will make the keystroke operate as a "normal" key with no special function. See [NormalKey](#)^[117] for an example.

3.2.4.2.13 PopFile

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](#) if you need to remove the default mapping of Ctrl-Tab to this function.

See other [Command-Line Editing Keys](#)^[118].

3.2.4.2.14 PrevFile

PrevFile = Key (F8, Shift-Tab): Gets the previous matching filename. See [ClearKeyMap](#) if you need to remove the default mapping of Shift-Tab to this function.

See other [Command-Line Editing Keys](#)^[118].

3.2.4.2.15 PrevHistory

PrevHistory = Key (Up for 4NT, Ctrl-Up for TC): Recalls the previous command from the command history. **(TC)** Also see [Scrolling and History Keystrokes](#)^[25] and the [SwapScrollKeys](#)^[117] directive.

See other [Command-Line Editing Keys](#)^[118].

3.2.4.2.16 RepeatFile

RepeatFile = Key (F12): Repeats the previous matching filename during filename completion.

See other [Command-Line Editing Keys](#)^[118].

3.2.4.2.17 SaveHistory

SaveHistory = Key (Ctrl-K): Saves the command line in the command history list without executing it.

See other [Command-Line Editing Keys](#)^[118].

3.2.4.2.18 Scrollback Buffer Keys

Scrollback Buffer Keys (TC)

The following keys are also part of the command line editing group. They are used to manipulate the [scrollback buffer](#)^[75] rather than to edit commands. For additional information see [Scrolling and History Keystrokes](#)^[25] and the [SwapScrollKeys](#)^[117] directive.

ScrollUp ^[121]	Scroll the buffer up one line.
ScrollDown ^[121]	Scroll the buffer down one line.
ScrollPgUp ^[122]	Scroll the buffer up one page.
ScrollPgDn ^[122]	Scroll the buffer down one page.

See other [Command-Line Editing Keys](#)^[121].

3.2.4.2.18.1 ScrollUp

ScrollUp = Key: Scrolls the command processor's [scrollback buffer](#)^[75] up one line.

See other [Scrollback Buffer Keys](#)^[121].

3.2.4.2.18.2 ScrollDown

ScrollDown = Key: Scrolls the command processor's [scrollback buffer](#)^[75] down one line.

See other [Scrollback Buffer Keys](#)^[121].

3.2.4.2.18.3 ScrollPgUp

ScrollPgUp = Key: Scrolls the command processor's [scrollback buffer](#)^[75] up one page.

See other [Scrollback Buffer Keys](#)^[121].

3.2.4.2.18.4 ScrollPgDn

ScrollPgDn = Key: Scrolls the command processor's [scrollback buffer](#)^[75] down one page.

See other [Scrollback Buffer Keys](#)^[121].

3.2.4.3 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](#)^[407] window.

DirWinOpen ^[122]	Opens the directory history window
HistWinOpen ^[122]	Opens the command history window
NormalPopupKey ^[122]	Deassigns a popup window key
PopupWinBegin ^[123]	Moves to the first line of the popup window
PopupWinDel ^[123]	Deletes a line from within the popup window
PopupWinEdit ^[123]	Moves a line from the popup window to the prompt
PopupWinEnd ^[123]	Moves to the last line of the popup window
PopupWinExec ^[123]	Selects the current item and closes the popup window

3.2.4.3.1 DirWinOpen

DirWinOpen = Key (Ctrl-PgUp, F6): Opens the directory history window while at the command line. **(TC)** Also see [Scrolling and History Keystrokes](#)^[25].

See other [Popup Window Keys](#)^[122].

3.2.4.3.2 HistWinOpen

HistWinOpen = Key (PgUp for **4NT**, Ctrl-PgUp for **TC**): Brings up the history window while at the command line. **(TC)** Also see [Scrolling and History Keystrokes](#)^[25] and the [SwapScrollKeys](#)^[111] directive.

See other [Popup Window Keys](#)^[122].

3.2.4.3.3 NormalPopupKey

NormalPopupKey = Key: Deassigns a popup window key in order to disable the usual meaning of the key within the popup window. This will make the keystroke operate as a "normal" key with no special function. See [NormalKey](#)^[117] for an example.

See other [Popup Window Keys](#)^[122].

3.2.4.3.4 PopupWinBegin

PopupWinBegin = Key (Ctrl-PgUp): Moves to the first item in the list when in the popup window.

See other [Popup Window Keys](#)^[122].

3.2.4.3.5 PopupWinDel

PopupWinDel = Key (Ctrl-D): Deletes a line from within the command history or directory history window.

See other [Popup Window Keys](#)^[122].

3.2.4.3.6 PopupWinEdit

PopupWinEdit = Key (Ctrl-Enter): Moves a line from the command history or directory history window to the prompt for editing.

See other [Popup Window Keys](#)^[122].

3.2.4.3.7 PopupWinEnd

PopupWinEnd = Key (Ctrl-PgDn): Moves to the last item in the list when in the popup window.

See other [Popup Window Keys](#)^[122].

3.2.4.3.8 PopupWinExec

PopupWinExec = Key (Enter): Selects the current item and closes the window.

See other [Popup Window Keys](#)^[122].

3.2.4.4 LIST Keys

These directives are effective only inside the [LIST](#)^[237] command.

ListExit ^[124]	Exits the current file
ListFind ^[124]	Prompts and searches for a string
ListFindReverse ^[124]	Prompts and searches backwards
ListHex ^[124]	Toggles hexadecimal display mode
ListHighBit ^[124]	Toggles LIST's "strip high bit" option
ListInfo ^[124]	Displays information about the current file
ListNext ^[124]	Finds the next matching string
ListOpen ^[124]	Displays the "open file" dialog
ListPrevious ^[125]	Finds the previous matching string
ListPrint ^[125]	Prints the file on LPT1
ListUnicode ^[125]	Toggles Unicode display mode
ListWrap ^[125]	Toggles LIST's wrap option
NormalListKey ^[125]	Deassigns a LIST key

3.2.4.4.1 ListExit

ListExit = Key (Esc): Exits from the [LIST](#)^[237] command.

See other [LIST Keys](#)^[123].

3.2.4.4.2 ListFind

ListFind = Key (F): Prompts and searches for a string.

See other [LIST Keys](#)^[123].

3.2.4.4.3 ListFindReverse

ListFindReverse = Key (Ctrl-F): Prompts and searches backward for a string.

See other [LIST Keys](#)^[123].

3.2.4.4.4 ListHex

ListHex = Key (X): Toggles hexadecimal display mode.

See other [LIST Keys](#)^[123].

3.2.4.4.5 ListHighBit

ListHighBit = Key (H): Toggles LIST's "strip high bit" option, which can aid in displaying files from certain word processors.

See other [LIST Keys](#)^[123].

3.2.4.4.6 ListInfo

ListInfo = Key (I): Displays information about the current file.

See other [LIST Keys](#)^[123].

3.2.4.4.7 ListNext

ListNext = Key (N): Finds the next matching string.

See other [LIST Keys](#)^[123].

3.2.4.4.8 ListOpen

ListOpen = Key (O): Opens the common Windows "open file" dialog to select a new file to [LIST](#)^[237].

See other [LIST Keys](#)^[123].

3.2.4.4.9 ListPrevious

ListPrevious = Key (Ctrl-B): Finds the previous matching string.

See other [LIST Keys](#)^[123].

3.2.4.4.10 ListPrint

ListPrint = Key (P): Prints the file on LPT1.

See other [LIST Keys](#)^[123].

3.2.4.4.11 ListUnicode

ListUnicode = Key (U): Toggles the [LIST](#)^[237] display mode between Unicode and ASCII.

See other [LIST Keys](#)^[123].

3.2.4.4.12 ListWrap

ListWrap = Key (W): Toggles [LIST](#)^[237]'s wrap option on and off. The wrap option wraps text at the right margin.

See other [LIST Keys](#)^[123].

3.2.4.4.13 NormalListKey

NormalListKey = Key: Deassigns a [LIST](#)^[237] key in order to disable the usual meaning of the key within **LIST**. This will make the keystroke operate as a "normal" key with no special function. See [NormalKey](#)^[117] for an example.

See other [LIST Keys](#)^[123].

3.2.5 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 dialogs](#)^[82]; you must enter them manually (see the [.INI file](#)^[89] for details).

- ▶ [ClearKeyMap](#)^[126] Clear default key mappings
- ▶ [Debug](#)^[126] Set debugging options
- ▶ [Include](#)^[126] Include text from another file in the current *.INI* file
- ▶ [NextINIFile](#)^[126] INI file for all secondary shells.
- ▶ [SaveDirCase](#)^[126] Control preservation of original upper/lower case name when changing directories.

3.2.5.1 ClearKeyMap

ClearKeyMap: Clears all current [key mappings](#)^[114]. 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 other 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.).

See other [Advanced Directives](#)^[125].

3.2.5.2 Debug

Debug = nnnn (default is 2): Controls certain debugging options which can assist you in tracking down unusual problems. Use the following values for **Debug** (to enable more than one option, add the desired values together):

- 0 Disabled.
- 1 During the startup process, display the complete command tail passed to the command processor, then wait for a keystroke.
- 2 (default) Include the product name with every error message displayed by the command processor. This may be useful if you are unsure of the origin of a particular error message.

Also see the [batch file debugger](#)^[149], a separate and unrelated facility for stepping through batch files.

See other [Advanced Directives](#)^[125].

3.2.5.3 Include

Include = File: Include the text from the named file at this point in the processing of the current [INI file](#)^[89]. Use this option to share a file of directives between several JP Software 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 dialogs modify the original .INI file only, and do not update included files.

See other [Advanced Directives](#)^[125].

3.2.5.4 NextINIFile

NextINIFile = File. The full path and name of the file must be specified. All subsequent shells will read the specified [INI file](#)^[89], and ignore any **[Secondary]** section in the original .INI file.

See other [Advanced Directives](#)^[125].

3.2.5.5 SaveDirCase

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.

See other [Advanced Directives](#)^[125].

4 Internal Commands

Our command processors give you instant access to more than **125** internal commands. By comparison, Microsoft's [CMD.EXE](#)^[430] provides fewer than 50 internal commands. The best way to learn about commands is to experiment with them. This section will help you find the one(s) that you need, categorized in the lists below by name and by category.

- ▶ [Commands By Name](#)^[127]
- ▶ [Commands By Category](#)^[131]

Note: Remember that you can replace any internal command with an [ALIAS](#)^[138], or disable a command altogether with [SETDOS /I](#)^[283].

4.1 List by Name

Also see: [Internal Commands Listed by Category](#)^[131]

	Description	4NT	TC
? ^[136]	Display internal commands	X	X
ACTIVATE ^[136]	Activate or set window state	X	X
ALIAS ^[138]	Define or display aliases	X	X
ASSOC ^[146]	Windows file associations	X	X
ATTRIB ^[146]	Change or display file attributes	X	X
BATCOMP ^[149]	Batch file compression	X	X
BDEBUGGER ^[149]	Batch file debugger	X	X
BEEP ^[156]	Beep the speaker	X	X
BREAK ^[157]	Define or display Ctrl-C state	X	X
CALL ^[157]	Call another batch file	X	X
CANCEL ^[158]	End batch file processing	X	X
CD ^[159]	Display or change directory	X	X
CDD ^[160]	Change drive and directory	X	X
CHCP ^[162]	Display or change code page	X	
CHDIR ^[159]	Display or change directory	X	X
CLS ^[163]	Clear the display window	X	X
COLOR ^[163]	Change the display colors	X	X
COPY ^[164]	Copy files and/or directories	X	X

<u>DATE</u> ^[170]	Display or change date	X	X
<u>DDEEXEC</u> ^[171]	Send DDE command		X
<u>DEL</u> ^[172]	Delete files and/or directories	X	X
<u>DELAY</u> ^[173]	Wait for specified time	X	X
<u>DESCRIBE</u> ^[174]	Display or change descriptions	X	X
<u>DETACH</u> ^[175]	Start app detached	X	X
<u>DIR</u> ^[176]	Display files and/or directories	X	X
<u>DIRHISTORY</u> ^[189]	Display directory history list	X	X
<u>DIRS</u> ^[190]	Display directory stack	X	X
<u>DO</u> ^[191]	Create batch file loops	X	X
<u>DRAWBOX</u> ^[195]	Draw a box	X	X
<u>DRAWHLINE</u> ^[196]	Draw a horizontal line	X	X
<u>DRAWVLINE</u> ^[197]	Draw a vertical line	X	X
<u>ECHO</u> ^[198]	Echo a message	X	X
<u>ECHOERR</u> ^[198]	Echo a message to STDERR	X	X
<u>ECHOS</u> ^[199]	Echo a message with no CR/LF	X	X
<u>ECHOSERR</u> ^[199]	Echo with no CR/LF to STDERR	X	X
<u>ENDLOCAL</u> ^[200]	Restore from a SETLOCAL	X	X
<u>ERASE</u> ^[172]	Delete files and/or directories	X	X
<u>ESET</u> ^[201]	Edit variables or aliases	X	X
<u>EVENTLOG</u> ^[203]	Write Windows event log	X	X
<u>EXCEPT</u> ^[204]	Exclude files from a command	X	X
<u>EXIT</u> ^[205]	Exit the command processor	X	X
<u>FFIND</u> ^[206]	Search for files or text	X	X
<u>FOR</u> ^[210]	Repeat a command	X	X
<u>FREE</u> ^[216]	Display disk space	X	X
<u>FTYPE</u> ^[217]	Display or edit file types	X	X
<u>FUNCTION</u> ^[218]	Create or edit user functions	X	X
<u>GLOBAL</u> ^[220]	Run command in subdirectories	X	X
<u>GOSUB</u> ^[221]	Call batch subroutines	X	X
<u>GOTO</u> ^[222]	Branch in a batch file	X	X
<u>HEAD</u> ^[224]	Display beginning of file	X	X
<u>HELP</u> ^[225]	Help for internal commands	X	X
<u>HISTORY</u> ^[226]	Display or change history	X	X

<u>IF</u> ^[227]	Conditional command execution	X	X
<u>IFF</u> ^[228]	Conditional command execution	X	X
<u>IFTP</u> ^[229]	Open FTP connection	X	X
<u>INKEY</u> ^[231]	Get a single keystroke	X	X
<u>INPUT</u> ^[233]	Get a text string	X	X
<u>KEYBD</u> ^[234]	Set keyboard toggles	X	X
<u>KEYS</u> ^[235]	Enable or disable history list	X	X
<u>KEYSTACK</u> ^[236]	Send keystrokes to app	X	X
<u>LIST</u> ^[237]	Display files	X	X
<u>LOADBTM</u> ^[242]	Load batch file as .BTM	X	X
<u>LOG</u> ^[242]	Save log of commands	X	X
<u>MD</u> ^[243]	Create subdirectories	X	X
<u>MEMORY</u> ^[245]	Display memory statistics	X	X
<u>MKDIR</u> ^[243]	Create subdirectories	X	X
<u>MKLNK</u> ^[245]	Create NTFS hard links	X	X
<u>MOVE</u> ^[246]	Move files or directories	X	X
<u>MSGBOX</u> ^[251]	Popup message box	X	X
<u>ON</u> ^[252]	Batch file error trapping	X	X
<u>OPTION</u> ^[253]	Configure command processor	X	X
<u>PATH</u> ^[254]	Set or display PATH	X	X
<u>PAUSE</u> ^[256]	Wait for input	X	X
<u>PDIR</u> ^[256]	User-formatted DIR	X	X
<u>PLAYAVI</u> ^[260]	Display an .AVI file	X	X
<u>PLAYSOUND</u> ^[261]	Play a sound file	X	X
<u>POPD</u> ^[261]	Restore from directory stack	X	X
<u>PRINT</u> ^[262]	Print a file	X	X
<u>PROMPT</u> ^[262]	Change command-line prompt	X	X
<u>PUSHD</u> ^[264]	Save directory to stack	X	X
<u>QUERYBOX</u> ^[265]	Popup input box	X	X
<u>QUIT</u> ^[266]	Exit batch file	X	X

<u>RD</u> ^[267]	Remove subdirectory	X	X
<u>REBOOT</u> ^[268]	Reboot system	X	X
<u>RECYCLE</u> ^[269]	Display or empty recycle bin	X	X
<u>REM</u> ^[269]	Remark	X	X
<u>REN</u> ^[270]	Rename files or directories	X	X
<u>RENAME</u> ^[270]	Rename files or directories	X	X
<u>RETURN</u> ^[272]	Return from GOSUB	X	X
<u>RMDIR</u> ^[267]	Remove subdirectory	X	X
<u>SCREEN</u> ^[273]	Position cursor	X	X
<u>SCRPUT</u> ^[274]	Write directly to screen	X	X
<u>SELECT</u> ^[275]	Select files for a command	X	X
<u>SENDMAIL</u> ^[279]	Send email	X	X
<u>SET</u> ^[281]	Set environment variables	X	X
<u>SETDOS</u> ^[283]	Set command processor variables	X	X
<u>SETLOCAL</u> ^[287]	Save environment and aliases	X	X
<u>SHIFT</u> ^[288]	Shift batch file arguments	X	X
<u>SHORTCUT</u> ^[289]	Create a Windows shortcut	X	X
<u>SHRALIAS</u> ^[290]	Share aliases	X	X
<u>SMPP</u> ^[290]	Simple message transfer	X	X
<u>SNPP</u> ^[291]	Send message to pager	X	X
<u>START</u> ^[291]	Start a new session	X	X
<u>SWITCH</u> ^[295]	Batch file switch / case	X	X
<u>TAIL</u> ^[296]	Display end of file	X	X
<u>TASKEND</u> ^[298]	End a task	X	X
<u>TASKLIST</u> ^[299]	Display Windows task list	X	X
<u>TCTOOLBAR</u> ^[299]	Edit Toolbar		X
<u>TEE</u> ^[300]	Pipe "tee-fitting"	X	X
<u>TEXT</u> ^[301]	Display text in batch file	X	X
<u>TIME</u> ^[302]	Set or display time	X	X
<u>TIMER</u> ^[303]	Stopwatch	X	X
<u>TITLE</u> ^[304]	Set window title	X	X
<u>TOUCH</u> ^[304]	Change file timestamps	X	X
<u>TREE</u> ^[307]	Display directory tree	X	X
<u>TRUENAME</u> ^[308]	Display true pathname	X	X
<u>TYPE</u> ^[308]	Display files	X	X

UNALIAS ^[310]	Remove aliases	X	X
UNFUNCTION ^[311]	Remove user-defined functions	X	X
UNSET ^[312]	Remove environment variable	X	X
VER ^[313]	Display version	X	X
VERIFY ^[314]	Display or set disk verification	X	X
VOL ^[315]	Display or set disk volume label	X	X
VSCRPUT ^[316]	Write text vertically	X	X
WHICH ^[317]	Display command information	X	X
WINDOW ^[318]	Window management	X	X
Y ^[319]	Pipe "y-fitting"	X	X
	Description	4NT	TC

4.2 List by Category

Also see: [Internal Commands Listed by Name](#) ^[127]

The best way to learn about commands is to experiment with them. The lists below categorize the available commands by topic and will help you find the one(s) you need.

- ▶ [File and directory management](#) ^[131]
- ▶ [Subdirectory management](#) ^[131]
- ▶ [Input and output](#) ^[131]
- ▶ [Window management commands](#) ^[131]
- ▶ [Commands primarily for use in or with batch files and aliases](#) ^[131]
- ▶ [Environment and path commands](#) ^[131]
- ▶ [System configuration and status](#) ^[131]
- ▶ [Other commands](#) ^[131]

File and directory management

	<u>4NT</u>	<u>TC</u>
<u>ATTRIB</u> ^[14b]	X	X
<u>COPY</u> ^[16b]	X	X
<u>DEL</u> ^[17b]	X	X
<u>DESCRIBE</u> ^[17b]	X	X
<u>ERASE</u> ^[17b]	X	X
<u>FFIND</u> ^[20b]	X	X
<u>HEAD</u> ^[22b]	X	X
<u>LIST</u> ^[23b]	X	X
<u>MOVE</u> ^[24b]	X	X
<u>RECYCLE</u> ^[26b]	X	X
<u>REN</u> ^[27b]	X	X
<u>RENAME</u> ^[27b]	X	X
<u>SELECT</u> ^[27b]	X	X
<u>TAIL</u> ^[29b]	X	X
<u>TOUCH</u> ^[30b]	X	X
<u>TREE</u> ^[30b]	X	X
<u>TRUENAME</u> ^[30b]	X	X
<u>TYPE</u> ^[30b]	X	X
<u>Y</u> ^[31b]	X	X

Subdirectory management

	<u>4NT</u>	<u>TC</u>
<u>CD</u> ^[15b]	X	X
<u>CDD</u> ^[16b]	X	X
<u>CHDIR</u> ^[15b]	X	X
<u>DIR</u> ^[17b]	X	X
<u>DIRS</u> ^[19b]	X	X
<u>MD</u> ^[24b]	X	X
<u>MKDIR</u> ^[24b]	X	X
<u>MKLNK</u> ^[24b]	X	X
<u>PDIR</u> ^[25b]	X	X
<u>POPD</u> ^[26b]	X	X
<u>PUSHD</u> ^[26b]	X	X
<u>RD</u> ^[26b]	X	X
<u>RMDIR</u> ^[26b]	X	X

Input and output

	4NT	TC
<u>DRAWBOX</u> <small>[195]</small>	X	X
<u>DRAWHLINE</u> <small>[196]</small>	X	X
<u>DRAWVLINE</u> <small>[197]</small>	X	X
<u>ECHO</u> <small>[198]</small>	X	X
<u>ECHOERR</u> <small>[198]</small>	X	X
<u>ECHOS</u> <small>[199]</small>	X	X
<u>ECHOSERR</u> <small>[199]</small>	X	X
<u>IFTP</u> <small>[229]</small>	X	X
<u>INKEY</u> <small>[231]</small>	X	X
<u>INPUT</u> <small>[233]</small>	X	X
<u>KEYSTACK</u> <small>[236]</small>	X	X
<u>MSGBOX</u> <small>[251]</small>	X	X
<u>PLAYAVI</u> <small>[260]</small>	X	X
<u>PLAYSOUND</u> <small>[261]</small>	X	X
<u>PRINT</u> <small>[262]</small>	X	X
<u>QUERYBOX</u> <small>[265]</small>	X	X
<u>SCREEN</u> <small>[273]</small>	X	X
<u>SCRPUT</u> <small>[274]</small>	X	X
<u>SENDMAIL</u> <small>[279]</small>	X	X
<u>SMPP</u> <small>[290]</small>	X	X
<u>SNPP</u> <small>[291]</small>	X	X
<u>VSCRPUT</u> <small>[314]</small>	X	X

Window management commands

	4NT	TC
<u>ACTIVATE</u> <small>[136]</small>	X	X
<u>TITLE</u> <small>[304]</small>	X	X
<u>WINDOW</u> <small>[316]</small>	X	X

Commands primarily for use in or with batch files and aliases
 (some work only in batch files; see the individual commands for details)

	<u>4NT</u>	<u>TC</u>
<u>ALIAS</u> ^[138]	X	X
<u>BATCOMP</u> ^[149]	X	X
<u>BDEBUGGER</u> ^[149]	X	X
<u>BEEP</u> ^[156]	X	X
<u>CALL</u> ^[157]	X	X
<u>CANCEL</u> ^[158]	X	X
<u>DDEEXEC</u> ^[177]		X
<u>DELAY</u> ^[178]	X	X
<u>DO</u> ^[197]	X	X
<u>ENDLOCAL</u> ^[208]	X	X
<u>FOR</u> ^[218]	X	X
<u>FUNCTION</u> ^[218]	X	X
<u>GLOBAL</u> ^[228]	X	X
<u>GOSUB</u> ^[227]	X	X
<u>GOTO</u> ^[222]	X	X
<u>IF</u> ^[227]	X	X
<u>IFF</u> ^[228]	X	X
<u>LOADBTM</u> ^[242]	X	X
<u>ON</u> ^[252]	X	X
<u>PAUSE</u> ^[256]	X	X
<u>QUIT</u> ^[266]	X	X
<u>REM</u> ^[269]	X	X
<u>RETURN</u> ^[272]	X	X
<u>SETLOCAL</u> ^[287]	X	X
<u>SHIFT</u> ^[288]	X	X
<u>SWITCH</u> ^[295]	X	X
<u>TEXT</u> ^[307]	X	X
<u>UNALIAS</u> ^[316]	X	X
<u>UNFUNCTION</u> ^[317]	X	X

Environment and path commands

	<u>4NT</u>	<u>TC</u>
<u>ESET</u> [207]	X	X
<u>PATH</u> [254]	X	X
<u>SET</u> [287]	X	X
<u>UNSET</u> [312]	X	X

System configuration and status

	<u>4NT</u>	<u>TC</u>
<u>ASSOC</u> [145]	X	X
<u>BREAK</u> [157]	X	X
<u>CLS</u> [163]	X	X
<u>CHCP</u> [162]	X	
<u>COLOR</u> [163]	X	X
<u>DATE</u> [170]	X	X
<u>DIRHISTORY</u> [189]	X	X
<u>EVENTLOG</u> [203]	X	X
<u>FREE</u> [216]	X	X
<u>FTYPE</u> [217]	X	X
<u>HISTORY</u> [226]	X	X
<u>KEYBD</u> [234]	X	X
<u>KEYS</u> [235]	X	X
<u>LOG</u> [242]	X	X
<u>MEMORY</u> [245]	X	X
<u>OPTION</u> [253]	X	X
<u>PROMPT</u> [262]	X	X
<u>REBOOT</u> [268]	X	X
<u>SETDOS</u> [283]	X	X
<u>SHORTCUT</u> [289]	X	X
<u>TASKEND</u> [298]	X	X
<u>TASKLIST</u> [299]	X	X
<u>TCTOOLBAR</u> [299]		X
<u>TIME</u> [302]	X	X
<u>VERIFY</u> [314]	X	X
<u>VER</u> [313]	X	X
<u>VOL</u> [314]	X	X

Other commands

	4NT	TC
? ^[136]	X	X
DETACH ^[176]	X	X
EXCEPT ^[204]	X	X
EXIT ^[205]	X	X
HELP ^[225]	X	X
SHRALIAS ^[296]	X	X
START ^[297]	X	X
TEE ^[306]	X	X
TIMER ^[303]	X	X
WHICH ^[315]	X	X

4.3 ? (command)

Purpose: Display a list of internal commands **or** prompt for a command.

Format: ? ["prompt" command]

Usage:

The "?" command has two separate meanings:

1. When you use the ? command by itself, it displays a list of internal commands. For help with any individual command, see the [HELP ^{\[225\]}](#) command. If you have disabled a command with [SETDOS /I ^{\[283\]}](#), it will not appear in the list.
2. The second function of ? is to prompt the user before executing a specific line. 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.

```
[c:\] ? "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)?
```

4.4 ACTIVATE

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 | "newtitle"]

window: Current title of window to work with

pos:	Move and resize window
topmost:	Set the topmost attribute
notopmost:	Remove the topmost attribute
top:	Move to the top of the window order
bottom:	Move to the bottom of the window order
hide:	Hide the window
newtitle:	New title for window

See also: [START](#)^[29], [TITLE](#)^[30], and [WINDOW](#)^[31].

Usage:

Both the current title and the new title, if any, must be enclosed in quote marks, which will not appear as part of the title text. If no options are specified, 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.

Note: ACTIVATE is designed to modify another session's window. It is **not** intended to modify the characteristics of the current session (use [TITLE](#)^[30] or [WINDOW](#)^[31] for that purpose).

The options are:

MAX	expands the window to its maximum size and activates it.
MIN	reduces the window to an icon.
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).
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 (to make the window visible again, use RESTORE).
"newtitle"	changes the window title.

You can specify only 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 first maximizes, and then renames the window originally called "Take Command":

```
[c:\] activate "Take Command" max
[c:\] activate "Take Command" "Take Command is Great!"
```

You can use [wildcards](#)^[36], including extended wildcards, 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.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

ACTIVATE is often used before [KEYSTACK](#)^[23] 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 / XP / 2003, 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 / 2003 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.

4.5 ALIAS

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 input files to read alias definitions from.

name: Name for an alias, or for the key to execute the alias.

value: Text to be substituted for the alias name or key.

[/P\(ause\)](#)^[138]

[/R\(ead file\)](#)^[138]

See also: [UNALIAS](#)^[310], [Aliases](#)^[318].

Usage:

- ▶ [Overview](#)^[138]
- ▶ [Multiple Commands and Special Characters in Aliases](#)^[138]
- ▶ [Nested Aliases](#)^[138]
- ▶ [Temporarily Disabling Aliases](#)^[138]
- ▶ [Partial \(Abbreviated\) Alias Names](#)^[138]
- ▶ [Keystroke Aliases](#)^[138]
- ▶ [Displaying Aliases](#)^[138]
- ▶ [Saving and Reloading Your Aliases](#)^[138]
- ▶ [Alias Parameters](#)^[138]
- ▶ [Expanding Aliases at the Prompt](#)^[138]
- ▶ [Local and Global Aliases](#)^[138]
- ▶ [Retaining Global Aliases with SHRALIAS](#)^[138]
- ▶ [The UNKNOWN_CMD Alias](#)^[138]

• Overview

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.

If there is only a single argument and it contains wildcards (* or ?), ALIAS will display all matching alias names.

(TC) You can also define or modify aliases with the [Alias Window](#)^[74]. 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 for Windows 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 for Windows 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 below 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`
```

This alias 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 "%+" [CommandSep](#)^[100] reference (or a literal ampersand "&" if the default value is used) indicates that the two commands are distinct and should be executed sequentially.

Aliases make extensive use of the [command separator](#)^[24], and the [parameter character](#)^[107], and may also use the [escape character](#)^[21].

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, 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 in the [configuration dialogs](#)^[82] or the [ExecWait](#)^[103] directive in the [INI file](#)^[89].

When you use the alias command at the command prompt or in a batch file, you must use back quotes ['] around the alias definition **if** it contains multiple commands, **or** parameters (discussed below), **or** environment variables, **or** 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. The examples above and below 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 for Windows in 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 POPD 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 *WINWORD.EXE* 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:

```
[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. 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](#)^[283] command.

• Partial (Abbreviated) 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 [Keys and Key Names](#)^[464] 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](#)^[117] or [NormalEditKey](#)^[120] directives in the [INI file](#)^[89].

You can also define a keystroke alias by using `"@"` or `"@@"` plus a scan code for one of the

permissible keys (see the [Key Code and Scan Code Tables](#)^[452] 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** currently defined aliases, type:

```
[c:\] alias
```

You can 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
```

You can also view the definitions for all aliases matching a specific pattern by specifying a single argument containing wildcards (* or ?). For example:

```
[c:\] alias *win*
```

will display all aliases containing the string "win".

- **Saving and Reloading Your Aliases**

You can save your aliases to a file ("ALIAS.LST") this way:

```
[c:\] alias > alias.lst
```

You can then reload all the alias definitions in the file the next time you start 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 the command processor 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 start the command processor.

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.

(TC) You can also save and reload your aliases using the [Aliases dialog](#)^[74]. 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](#)^[310] command.

- **Alias Parameters**

Aliases can use command-line arguments or parameters like those in batch files. The command-line arguments are numbered from %0 to %511. %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 [Argument Quoting](#)^[332] for details. Alias examples in this section assume the 4NT and Take Command default of "ParameterChar=\$". Adjust accordingly if you are using a different [ParameterChar](#)^[107].

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 only 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. The command processor interprets 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."

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). If an alias uses no parameters, all of the command-line arguments will be appended to the expanded command. A convenient way to prevent unwanted command-line arguments from being appended is to add a reference to **"%511"** within the alias.

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[%$]`
```

Now, if you enter:

```
[c:\] calc 5 * 6
```

The alias will display:

```
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.

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 other sessions.

With a global alias list, all copies of the command processor will share the same alias list, and any changes made to the aliases in one copy will affect all other copies. This is the default for 4NT and Take Command.

You can control the type of alias list from the [configuration dialogs](#)^[82], with the [LocalAliases](#)^[96] directive in the [.INI file](#)^[89], with the **/L** and **/LA** options of the [START](#)^[297] command, and with the **/L** and **/LA** [startup options](#)^[4].

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, then modify it if you find a situation where the default is not convenient.

Whenever you start a second copy of the command processor 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.

- **Retaining Global Aliases with SHRALIAS**

If you select a global alias list for the command processor you can share the aliases among all running copies of the command processor. When you close all command processor sessions, the memory for the global alias list is released, and a new, empty alias list is created the next time you start the command processor.

If you want the alias list to be retained in memory even when no command processor session is running, you need to execute the [SHRALIAS](#)^[290] command, which performs this service for the global alias list, the global user-defined functions list, the global command history list, and the global directory history list. You may find it convenient to execute SHRALIAS from your [TCSTART/4START](#)^[6] file.

SHRALIAS retains the alias list in memory, but cannot preserve it when Windows itself is shut down. To save your aliases when restarting Windows, you must store them in a file and reload them after the system restarts. For details on how to do so, see **Saving and Reloading Your Aliases** (above).

- **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 "%$" `
```

(using the 4NT/TC default of "[ParameterChar=\\$](#)^[107]").

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** (Pause) 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 and File Prompts](#)^[65]).
- /R** (Read 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 8,191 characters long. The definitions can span multiple lines in the file if each line of a definition, except the last, is terminated with an [escape character](#)^[21].

ALIAS /R will read from STDIN if no filename is specified and input is redirected:

```
[c:\] alias /r <
```

4.6 ASSOC

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 input files to read association definitions from.
.ext The file extension whose file type you want to display or set.
filetype A file type stored in the Windows registry.

[/P\(ause\)](#)^[145]

[/R\(ead\)](#)^[145]

See also: [FTYPE](#)^[217], and [Executable Extensions](#)^[48].

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 in more detail under [Windows File Associations](#)^[46]. 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 file types depend on the contents of your Windows registry. See the [FTYPE](#)^[217] 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 the **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** (Pause) Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[65].
- /R** (Read file) This option loads an association list from a file. The format of the file is the same as that of the ASSOC display:

```
.ext=filetype
```

where **.ext** is an extension, which is to be associated with **filetype**.

You can load multiple files with one ASSOC /R command by placing the names on the command line, separated by spaces:

```
[c:\] assoc /r assoc1.lst assoc2.lst
```

ASSOC /R will read from STDIN if no filename is specified and input is redirected.

4.7 ATTRIB

Purpose: Change or view file and subdirectory attributes.

Format: ATTRIB [/A:[-+]*rhsa*] /D /E /I"text" /P /Q /S] [+]*ADHORST*] [*@file*] *files* ...

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 [@file lists](#)^[49] for details).

[/A: \(Attribute select\)](#)^[146]

[/P\(ause\)](#)^[146]

[/D\(irectories\)](#)^[146]

[/Q\(quiet\)](#)^[146]

[/E \(No error messages\)](#)^[146]

[/S\(ubdirectories\)](#)^[146]

[/I"text" \(match description\)](#)^[146]

Attribute flags:

Clear	Set	Attribute affected
-A	+A	archive
-H	+H	hidden
-O	+O	offline
-T	+T	temporary
-R	+R	read-only
-S	+S	system

File Selection

Supports [attribute switches](#)^[39], extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46]. Use wildcards with caution on LFN volumes; see [LFN File Searches](#)^[47] for details.

Usage:

Every file and subdirectory has four 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 [File Attributes](#)^[44].

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 **[+]-[ADHORST]** part of the format), or with the [DIR /T](#)^[17] 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.

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), "J" (junction / reparse point) and "P" (sparse file) 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 **[+]-[ADHORST]** 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](#)^[36] 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*:

```
[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*
```

Options:

/A: (Attribute select) Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[39] for information on the attributes which can follow **/A:**. *Warning: the colon after /A is not optional.*

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 *.*
```

Do not use **/A:** with [@file lists](#)^[49] for details.

/D (Directories) If you use the **/D** option, ATTRIB will modify the attributes of directories in addition to files (yes, you can have a hidden directory):

```
[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, and when recursing through the directory hierarchy, where many directories have no files matching your selection criteria.

/I"text" (Match descriptions) Select files by matching text in their descriptions. The text can include [wildcards](#)^[36] 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"[]"**. Do not use **/I** with [@file lists](#)^[49] for details

/P (Pause) Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[63]

/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 [@file lists](#)^[49] for details.

4.8 BATCOMP

Purpose: Compress and optionally encrypt batch files. See [Batch File Compression](#)^[33] for additional details.

Format: BATCOMP [/Ekkkk /K /O /Q] InputFile [OutputFile]

Inputfile: A file to compress and/or encrypt.

OutputFile: A file to hold the output from the command

/E(ncryption key) **/O**(verwrite)

/K (remove comments) **/Q**(uiet)

File Selection

The single input file must be specified explicitly.

Usage:

BATCOMP is a batch file compressor and optionally allows for simple key-based encryption.

If you do not specify **OutputFile**, it defaults to the **InputFile** name with a **BTM** extension. If you specify **OutputFile**, you must give an extension or the outfile will not have one. In other words there is no BTM default if you just give a file name for outfile. If **OutputFile** already exists it will not be overwritten unless **/O** is used.

The output BTM file will not be legible, but it will run under the current versions of 4NT and Take Command. The behavior and performance of the file should be the same as if it were run in its original source form as a .BTM file.

Compression is not effective for very small files and may even result in a slightly larger file.

Options:

/E (Encryption key) Used to specify a key (string) to be used for encryption.

/K (Remove comments) By defaults, comments (i.e. lines starting with "REM" or "::" but not "REM >") are retained in the compressed file. /K forces those lines to be stripped for additional size reduction.

/O (Overwrite) Forces overwriting of any existing **OutputFile**.

/Q (Quiet) Suppresses all progress reports (stdout). Errors (stderr) are still shown.

Note: the BATCOMP internal command replaces the now obsolete external program BATCOM32.EXE distributed with prior versions.

4.9 BDEBUGGER

Purpose: Calls the internal batch file debugger. This is an extremely powerful command that includes support for *breakpoints*, *bookmarks*, *syntax coloring*, and *editing*

Format: BDEBUBGER [[/A] [/B] [/E] [/F] [/N] batchfilename [arguments]]

batchfilename: Full name of the batch file to debug.

arguments: Arguments for the batch file

[/A\(lias list window\)](#) ^[149]

[/F \(user-defined Functions window\)](#) ^[149]

[/B\(atch variable window\)](#) ^[149]

[/I\(nactive\)](#) ^[149]

[/E\(nvironment variable window\)](#) ^[149]

[/N \(do Not hide the debugger window\)](#) ^[149]

Usage:

BDEBUBGER opens a special window in which a batch file can be examined, modified, and executed. See the [Debugging Batch Files](#) ^[329] topic for other possible approaches.

The default settings for the BDEBUBGER (Batch File Debugger) command are kept in ASCII text file **BATCH.BCP** (in the program directory). Only modify the contents of that file if you're positive you know what you're doing. We strongly recommend that you keep a copy of the original, should your modifications prove unworkable or undesirable.

The batch debugger window toolbar has a number of icons to control debugging. Each has a tooltip for quick reference:

New	Create a new batch file. If you have an existing file open, you will be prompted to save any changes before it exits.
Open	Open an existing batch file.
Save	Save the current batch file.
Print	Print the current batch file.
Copy	Copy the highlighted selection to the clipboard.
Cut	Copy the highlighted selection to the clipboard and delete it from the file.
Paste	Copy the contents of the clipboard to the current cursor location.
Find	Search for text.
Replace	Replace the text with other text.
Undo	Undo the last edit.
Redo	Restore the last Undo.
Start Debugging	Starts the debugger. The cursor will be placed on the first line.
Stop Debugging	Stops the debugger.
Step	Execute the current line.
Skip	Skip the current line
Run to Breakpoint	Execute the batch file, stopping at the next breakpoint.
Toggle Breakpoint	Sets or turns off a breakpoint on the current line.
Clear Breakpoints	Turns off all breakpoints.
File Properties	Displays information on the current batch file.
Start New Shell	Start another copy of the command processor (this is useful if you need to perform some tasks while debugging a file.)

You can also set a breakpoint by moving the mouse cursor to the left margin of a line and left-clicking.

You can open popup windows to display and/or modify aliases, batch variables, environment variables, and user functions. The variable windows also have a toolbar, with a couple of additional buttons:

Update List	Save a modified list.
Import a File	Add the contents of a file to the list.

The text processing commands available in the batch debugger and variable windows are listed below. The default "hard coded" values used in the BATCH.BCP file are shown in parentheses after each

command name. The text commands can be classified into general categories:

- ▶ [Caret commands](#)
- ▶ [View commands](#)
- ▶ [Edit commands](#)
- ▶ [Mark / Clipboard commands](#)
- ▶ [Search commands](#)
- ▶ [File commands](#)
- ▶ [Bookmark commands](#)
- ▶ [Breakpoint commands](#)

- **Caret commands**

- | | | |
|-------------------------|------|--|
| Right | (1) | This command will move the caret one character to the right. When the caret is on the last position of the current line it is moved to the first position of the next line. |
| Shift-Right | (2) | In addition to the caret movement this command will also extend the current selection to the new caret position. |
| Left | (3) | This command will move the caret one character to the left. When the caret is on the first position of the current line it is moved to the last position of the previous line. |
| Shift-Left | (4) | In addition to the caret movement, this will also extend the current selection to the new position. |
| Up | (5) | This command will move the caret one line up. The caret column position will be set as close to its previous column position as possible. |
| Shift-Up | (6) | In addition to the caret movement this command will also extend the current selection to the new position. |
| Down | (7) | This command will move the caret one line down. The caret column position will be set as close to its previous column position as possible. When the caret is on the last line but not on the last column it will be moved to the last column. |
| Shift-Down | (8) | In addition to the caret movement this command will also extend the current selection to the new position. |
| End | (9) | This command will move the caret to the end of the line it is currently on. If the caret is already at the end nothing happens. |
| Shift-End | (10) | In addition to the caret movement this command will also extend the current selection to the new position. |
| Home | (11) | This command will move the caret to the start of the line it is currently on. If the caret is already at the start nothing happens. |
| Shift-Home | (12) | In addition to the caret movement this command will also extend the current selection to the new position. |
| Ctrl-Right | (13) | This command will move in one of the following ways: <ul style="list-style-type: none"> • When the caret is located on a delimiter character the caret is moved right until the first non-delimiter and non-white space is found. • When the caret is located on a non-delimiter character and not on a white space character the caret is moved to the start of the next word. • When the caret is located on a white space the caret is moved to the start of the next word or the next delimiter depending on what comes first. • When the caret is located on the last word, delimiters or white-space of the current line the caret is moved to the first word or delimiter of the next line. |
| Ctrl-Shift-Right | (14) | In addition to the caret movement this command will also extend the current selection to the new caret position. |
| Ctrl-Left | (15) | This command will move in one of the following ways: <ul style="list-style-type: none"> • When the caret is located on a delimiter character and it is preceded by delimiters the caret is moved left to the first delimiter. • When the caret is located on a delimiter character and it is not preceded by delimiters the caret is moved to the start of the previous word or delimiters. • When the caret is located on a non-delimiter character and not on a white-space character the caret is moved to the start of the current word. • When the caret is located on a white space the caret is moved to the start of the previous word or the previous delimiters depending on what comes first. • When the caret is located on the start of the first word, delimiters or white-space of the current line the caret is moved to the start of the last word or delimiters of the previous line. |
| Ctrl-Shift-Left | (16) | In addition to the caret movement this command will also extend the current selection to the new position. |
| Ctrl-Home | (17) | This command will move the caret to the beginning of the text. When the caret is already at this location nothing happens. |

- **View commands**

- | | | |
|------------------|------|--|
| Ctrl-Down | (80) | This command will scroll the view one line up. The command will always keep the caret inside the view. When it reaches the bottom of the view, the caret is automatically moved to the next line. |
| Ctrl-Up | (81) | This command will scroll the view one line down. The command will always keep the caret inside the view. When it reaches the top of the view, the caret is automatically moved to the previous line. |
| Ctrl-PgDn | (82) | This command will scroll the view one column to the left. The caret is not moved. |
| Ctrl-PgUp | (83) | This command will scroll the view one column to the right. The caret is not moved. |

- **Edit commands**

Ctrl-Z	(100) This command will undo the last change made to the edit control contents. You can undo any number of changes made to the control contents up to the maximum number of undo/redo hops.
Ctrl-Y	(101) This command will redo the last change you have un-done. You can re-do any number of changes up to the number of changes un-done.
Shift-Del	(103) This command will remove all characters to the right of the current caret position.
Backspace	(104) This command will remove the character to the left of the caret. When the caret is located at the start of the line, the characters right of the caret are appended to the previous line and the caret is moved to be positioned between the old line contents and the appended characters.
Delete	(105) This command removes the character to the right of the caret. When there are no characters to the right of the caret, the contents of the next line is appended to the current line.
Alt-Delete	(106) This command deletes the current line.
Return	(107) This command will split the current line and create a new line of the characters, if any, right of the caret. The caret is moved to the start of the newly created line.
Alt-U	(108) This command will convert all lower-case characters of the word under the caret to upper-case characters. If the caret is not located on a word, nothing happens.
Alt-L	(109) This command will convert all upper-case characters of the word under the caret to lower-case characters. If the caret is not located on a word, nothing happens.
Alt-S	(110) This command will convert all lower-case characters to upper-case characters and upper-case characters to lower-case characters of the word under the caret. If the caret is not located on a word, nothing happens.
Ctrl-Delete	(113) When the caret is located on a word, this command will delete all characters in the word right of the caret position.
Ctrl-Backspace	(114) When the caret is located on a word, this command will delete all characters in the word left of the caret position.
Tab	(115) This command does one of the two following things: <ul style="list-style-type: none"> • When there is a valid text selection, this command will indent the lines covered by the selection right by one tab-stop. • When there is no text selection, a tab is inserted at the current caret position.
Shift-Tab	(116) This command does one of the two following things: <ul style="list-style-type: none"> • When there is a valid text selection, this command will indent the lines covered by the selection left by one tab-stop. • When there is no text selection, the caret is moved back to the previous tab-stop position.
Shift-Alt-T	(117) This command will swap the line on which the caret is located with the previous line. When the caret is located on the first line, nothing will happen.
Shift-Ctrl-U	(118) When there is a valid selection, this command will convert all lower-case characters in the selection to upper-case characters. If there is no valid selection, nothing happens.
Ctrl-U	(119) When there is a valid selection, this command will convert all upper-case characters in the selection to lower-case characters. If there is no valid selection, nothing happens.
Ctrl-Alt-U	(120) When there is a valid selection, this command will convert all lower-case characters in the selection to upper-case characters and all upper-case characters in the selection to lower-case. If there is no valid selection, nothing happens.
Ctrl-D	(121) This command will insert the system date at the current caret location. The date output is formatted using the user's default format.
Shift-Ctrl-D	(122) This command will insert the system time at the current caret location. The time

- **Mark / Clipboard commands**

- Shift-Ctrl-W** (200) This command will select the word on which the caret is currently located. When the caret is not located on a word, nothing happens.
- Ctrl-A** (201) This command will select all the text.
- Shift-Ctrl-L** (203) This command will select the line on which the caret is currently located.
- Ctrl-V** (225) This command will, when there is text present in the clipboard, paste the clipboard contents at the current position.
- Ctrl-C** (226) This command will, when there is a selection, copy the selected text to the clipboard.
- Ctrl-X** (227) This command will, when there is a selection, copy the selected text to the clipboard and remove the selection from the text.
- Shift-Ctrl-C** (229) This command will copy the line on which the caret is located to the clipboard.
- Shift-Ctrl-X** (230) This command will copy the line on which the caret is located to the clipboard and remove the line from the text.
- Alt-C** (232) This command will, if there is a selection, append the selection to the current clipboard contents. The result is placed on the clipboard again.
- Alt-X** (233) This command will, if there is a selection, append the selection to the current clipboard contents. The result is placed on the clipboard again, and the selection is removed from the text.

- **Search commands**

- Ctrl-F3** (275) This command will find the next occurrence of the word under the caret. When the next occurrence is found, it is selected.
- F3** (277) This command will find the next occurrence of the current search pattern. When the search pattern is found, it is selected.
- Shift-F3** (278) This command will find the previous occurrence of the current search pattern. When the search pattern is found, it is selected.
- Ctrl-]** (279) This command will find the matching closing bracket if the caret is located on an opening bracket, or the matching open bracket if the caret is located on a closing bracket.
- Shift-Ctrl-]** (280) This command will find the matching closing bracket if the caret is located on an opening bracket, or the matching opening bracket if the caret is located on a closing bracket. In addition the text from the opening bracket up to and including the closing bracket is selected.
- Ctrl-G** (300) This command will show the *goto* dialog.
- Ctrl-F** (301) This command will show the *find* dialog.
- Ctrl-H** (302) This command will show the *replace* dialog.

- **File commands**

- Ctrl-S** (401) This command will save the control contents using its current name
- Shift-Ctrl-Del** (402) This command will delete all text. If the text has been modified the user will be prompted to save the contents first.
- Ctrl-P** (403) This command will open the printer properties dialog.

- **Bookmark commands**

- Ctrl-F2** (252) This command will clear the bookmark on the current line if it is set, or set the bookmark if it is cleared.
- Shift-Ctrl-F2** (253) This command will clear all bookmarks.
- F2** (254) This command will place the caret on the next line which has a bookmark set. When there is no next line with a bookmark, the text is searched starting at the first line.
- Shift-F2** (255) This command will place the caret on the previous line which has a bookmark set. When there is no previous line with a bookmark, the text is searched from the last line up again.

- **Breakpoint commands**

- Ctrl-B** (262) This command will toggle a breakpoint on the current line.
- Ctrl-Shift-F9** (263) This command will clear all breakpoints.

Options:

- /A** (Aliases window): start with the alias window open.
- /B** (Batch variables window): start with the batch variables window open.
- /E** (Environment variables window): start with the environment variables window open.
- /F** (Functions window): start with the user-defined functions window open.
- /I** (Inactive): load batch file, but do not start execution
- /N** (Do NOT hide window): by default, the debugger window hides itself while the debugged batch file is executing. "/N" keeps that window visible in the background (the main 4NT or Take Command window is brought to the foreground).

4.10 BEEP

Purpose: Beep the speaker or play simple music.

Format: BEEP [frequency duration ...]

frequency: The beep frequency in Hertz (cycles per second).

duration: The beep length in 1/18th second intervals.

See also: [BeepFreq](#)^[10b], [BeepLength](#)^[10b].

Usage:

BEEP generates a sound through your computer's speaker. You can use it in batch files to signal that an operation has been completed, or that the computer needs attention.

Because BEEP allows you to specify the frequency and duration of the sound, you can also use it to 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, allowing you to use BEEP as a way to create short delays. The default

value for *frequency* is 440 Hz; the default value for *duration* is 2.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Windows 98 / ME will ignore all **frequency** and **duration** values, and instead will produce only the system default sound. This is due to the design of Windows 98 / ME, and is not a bug in the command processor.

This batch file fragment runs a program called *DEMO*, then 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	2096
C# / Db	139	277	554	1108	2217
D	147	294	587	1175	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	1664	3322
A	220	440	880	1760	3520
A# / Bb	233	466	932	1866	3729
B	248	494	988	1973	3951

4.11 BREAK

Purpose: None (Compatibility Only)

Format: BREAK [ON | OFF]

Usage:

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 under DOS.

4.12 CALL

Purpose: Execute one batch file from within another.

Format: CALL file

file: The batch file to execute.

See also: [CANCEL](#)^[158] and [QUIT](#)^[268].

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 (without executing the CANCEL command), execution of the original batch file resumes at the next command.

WARNING! If you execute a batch file from inside another batch file without using CALL, the original batch file is terminated before the other one starts. This method of invoking a batch file from another is usually referred to as *chaining*. Note that if batch file **A** uses **CALL B**, and **B** chains to batch file **C**, on exit from **C** (without executing the CANCEL command) processing of batch file **A** is resumed as if it had used **CALL C**.

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 32 levels deep.

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 by using the [QUIT](#)^[268] command), or it should terminate batch file processing with [CANCEL](#)^[158]. 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 [%_?](#)^[346] or [%?%](#)^[346] environment variable, and use it with [conditional commands](#)^[26] (&& and ||).

See also [GOSUB](#)^[221] and [user-defined functions](#)^[218].

4.13 CANCEL

Purpose: Terminate batch file processing.

Format: CANCEL [value]

value: The numeric exit code to return to the command processor.

See also: [CALL](#)^[157] and [QUIT](#)^[268].

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](#) ^[227] command, and the [%?](#) ^[346] variable). Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

4.14 CD / CHDIR

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](#) ^[160], [MD](#) ^[243], [PUSHD](#) ^[264], [RD](#) ^[267], and [Directory Navigation](#) ^[54].

Internet: Can be used with [FTP Servers](#) ^[52].

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](#) ^[160] 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.

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 [Extended Parent Directory Names](#) ^[47]). 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](#) ^[59] and the [extended directory search](#) ^[56] database 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. Read the section on [Directory Navigation](#)^[54] for complete details on these and other directory navigation features. To disable extended directory searches for the current command (e.g. in a batch file) see the **/N** option below.

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](#)^[22] 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](#)^[23], which allows you to return quickly to a recently-used directory.

You can also use CD to display or change the current directory on an [FTP server](#)^[52] opened with [IFTP](#)^[229]. To do so, simply enter the desired pathname in quotes, for example:

```
[c:\]cd "ftp:"
ftp://jpsoft.com/
[c:\]cd "ftp:/pub"
```

Note: FTP directory changes do not use the [CDPATH feature](#)^[59] or [Extended Directory Searches](#)^[56] 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.

Options:

- /D** (Drive) Changes the current drive as well as directory. This option is included only for compatibility with the undocumented CD /D command available in *CMD.EXE*. In most cases you should use [CDD](#)^[160], which performs the same function.
- /N** (No extended search) Skips the standard [extended directory search](#)^[56] 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.

4.15 CDD

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(ll drives)

/S(earch tree)

/D(elete from JPSTREE.IDX)

/U(pdate tree)

/N(o extended search)

See also: [CD](#)^[158], [MD](#)^[243], [PUSHD](#)^[264], [RD](#)^[267], and [Directory Navigation](#)^[54].

Usage:

CDD is similar to the [CD](#)^[159] 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]
```

CDD can also be used to create and update the [Extended Directory Search](#)^[56] database (*JPSTREE.IDX*).

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.

CDD can also change to a network drive and directory specified with a UNC name (see [File Systems](#)^[439] for details).

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.

If CDD cannot change to the directory you have specified it will first search the [CDPATH](#)^[59], then the [extended directory search](#)^[56] database 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. Read the section on [Directory Navigation](#)^[54] 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](#)^[22] also modify the saved directory, so you can use **CDD -** to return to a directory that 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](#)^[23], which allows you to return quickly to a recently-used directory.

The operating system limits the permissible length of the full subdirectory name (see the [Directories and Subdirectories](#)^[44b] topic for information on directory names).

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](#)^[12b] directive.

Options:

- /A** (All 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** Removes the specified drives or directory trees from the [Extended Directory Search](#)^[56] database (*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** (No extended search) Skips the standard [extended directory search](#)^[56] 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 (Search tree) Builds or rebuilds the [Extended Directory Search](#)^[56] database (*JPSTREE.IDX*). 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](#)^[10] *.INI* directive is set. 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
```

/U (Update) Updates the [Extended Directory Search](#)^[56] database (*JPSTREE.IDX*) with the specified drives and directories instead of rebuilding the whole 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:\
```

Note: The [TREEEXCLUDE](#)^[342] variable can be used to specify which drives and directories should be ignored when updating the directory database.

4.16 CHCP

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.

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 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.

4.17 CLS

Purpose: Clear the window and move the cursor to the upper left corner; optionally change the default display colors.

Format: CLS [/C /S] [[BR]ight] *fg* ON [BR]ight] *bg*

fg: The new foreground color

bg: The new background color

/C(lear buffer)

/S(croll buffer) (**4NT**)

Usage:

CLS can be used to clear the window without changing colors, or 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 [Colors and Color Names](#)^[467] for details about colors.

(4NT) Under Windows 98 and ME, if you haven't enabled ANSI support, the colors will not be "sticky" -- you may lose them when you run an application.

Options:

/C (Clear buffer) Clears the entire scrollbar buffer. If **/C** is not used, only the visible portion of the window is cleared.

/S (**4NT**) (Scroll 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.

4.18 COLOR

Purpose: Change the default display colors.

Format: COLOR [BR]ight] *fg* ON [BR]ight] *bg*

fg: The new foreground color

bg: The new background color

See also: [CLS](#)^[163], and [Colors and Color Names](#)^[467] for details about using colors and the name and numeric codes for colors.

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
```

(4NT) Under Windows 98 and ME, if you haven't enabled ANSI support, the colors will not be "sticky" -- you may lose them when you run an application.

4NT also supports the syntax of *CMD.EXE* included with Windows NT 4.0 and later:

```
COLOR bf
```

In this syntax, **b** is a hexadecimal digit that specifies the background color and **f** is a hexadecimal digit that specifies the foreground color.

4.19 COPY

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] [/K] [/L] [/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, one per line (see [@file lists](#) ^[49] for details).

[/A\(SCII\)](#) ^[164]

[/A: \(Attribute select\)](#) ^[164]

[/B\(inary\)](#) ^[164]

[/C\(hanged\)](#) ^[164]

[/D \(copy encrypted files\)](#) ^[164]

[/E \(No error messages\)](#) ^[164]

[/G \(display percent copied/verified\)](#) ^[164]

[/H\(idden included\)](#) ^[164]

[/I"*text*" \(match description\)](#) ^[164]

[/J \(copy in restartable mode\)](#) ^[164]

[/K\(eep attributes\)](#) ^[164]

[/L \(ASCII FTP transfer\)](#) ^[164]

[/M\(odified\)](#) ^[164]

[/N\(othing\)](#) ^[164]

[/O \(target doesn't exist\)](#) ^[164]

[/P\(rompt\)](#) ^[164]

[/Q\(uiet\)](#) ^[164]

[/R\(eplace\)](#) ^[164]

[/S\(ubdirectories included\)](#) ^[164]

[/T\(otals\)](#) ^[164]

[/U\(pdate\)](#) ^[164]

[/V\(erify\)](#) ^[164]

[/X \(clear archive\)](#) ^[164]

[/Z \(overwrite\)](#) ^[164]

See also: [ATTRIB](#) ^[146], [MOVE](#) ^[246], and [REN](#) ^[276].

File Selection

Supports [attribute switches](#) ^[39], extended [wildcards](#) ^[36], [ranges](#) ^[40], [multiple file names](#) ^[45], and [include lists](#) ^[46]. Date, time, size or exclude ranges anywhere on the line apply to all *source* files. Use wildcards with caution on LFN volumes; see [LFN File Searches](#) ^[47] for details.

Internet: Can be used with [FTP/TFTP/HTTP/HTTPS Servers](#) ^[52].

Usage:

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. The following 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.

To emulate an approach used by some implementations of *CMD.EXE*, see the [COPYCMD](#)^[34b] topic.

• Copying Files

You can copy several files at once by using [wildcards](#)^[36]:

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

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](#)^[46], 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 a file to a device such as the printer, use the device name as the **destination**, for example:

```
[c:\] copy schedule.txt prn
```

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 [Redirection](#) ^[67] for more information on CLIP:.

• Appending Files

A plus **[+]** tells COPY to append two or more files to a single **destination** file. If you list several **source** files separated with **[+]** and don't specify a **destination**, COPY will use the name of 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:\MEMMEMO2* and *C:\MEMMEMO3* to *D:\DATA\MEMO1*, and leave the result in *C:\MEMMEMO1*:

```
[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.

You cannot append a file to itself.

• FTP Usage

If you have appropriate permissions, you can copy to and from Internet URLs (FTP, TFTP and HTTP). The URL **must** be enclosed in quote marks. For example:

```
[c:\] copy "ftp://ftp.abc.com/foo/index" index
```

Files copied to or from FTP/HTTP Servers are normally transferred in binary mode. To perform an ASCII transfer use the **/L** switch. File descriptions are not copied when copying files to an Internet URL.

COPY supports the special syntax

```
copy con: "ftp:..."
```

to directly copy text from the console to an ftp location.

Wildcard characters such as **[*]** and **[?]** will be treated as wildcards in FTP URLs, but will be treated as normal characters in HTTP URLs.

Note: The **/G** option (percent copied) may report erratic values during transfer of files larger than 4 Gb (an ftp limitation) and during http downloads.

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/HTTP Servers](#)^[52] and [IFTP](#)^[229].

• NTFS File Streams

COPY supports file streams on NTFS drives under Windows 2000 and above. 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](#)^[444] 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 it copied.

You can use [date, time, and size ranges](#)^[40] to further define the files that you want to copy. 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:

/A:^[164] Forces COPY to process **source** files with the attributes you specify after the ":", or to process all **source** files regardless of attributes (if **/A:**^[164] is used by itself).

/H^[164] Forces COPY to process hidden and system **source** files, as well as normal files. The hidden and system attributes from each source file will be preserved when creating the **destination** files.

/K^[164] Retains the read-only attribute from each **source** file when creating the **destination** file. See **/K**^[164] below for a special note if you are running under Novell NetWare.

/Z^[164] Forces COPY to overwrite an existing read-only **destination** file.

Use caution with **/A:**^[164], **/H**^[164], or **/K**^[164] when both the **source** and **destination** directories contain file descriptions. If the **source** file 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 file 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\
```

(If you remove the hidden attribute from the *DESCRIPT.ION* file the same caution applies even if you do not use **/A:**^[164], **/H**^[164], or **/K**^[164], as *DESCRIPT.ION* is then treated like any other file).

Note: The **wildcard expansion** process will attempt to allow both CMD-style "*extension*" matching (assumes only one extension, at the end of the word) and the advanced 4NT and Take Command *string* matching (allowing things like **.*.abc*) when an asterisk is encountered in the **destination** of a COPY command.

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 (**/A:**, **/C**, **/E**, **/H**, **/K**, **/M**, **/N**, **/P**, **/Q**, **/R**, **/S**, **/T**, **/U**, **/V**, **/X**, **/Z**) 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. If you use conflicting output options, like **/Q** and **/P**, COPY will generally take a "conservative" approach and give priority to the option which generates more prompts or more information.

/A (ASCII) If you use **/A** with a **source** filename, the file will be copied up to, but not including, the first Control-Z (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.

This option applies to the filename immediately preceding it, and to all subsequent filenames until the file name preceding the next **/A** or **/B** option.

/A: (Attribute select) Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[39] for information on the attributes which can follow **/A:**. See the cautionary note under **Advanced Features** above before using **/A:** when both **source** and **destination** directories contain file descriptions. You must include the colon with this option to distinguish it from the **/A**(SCII) switch, above. Do not use **/A:** with @file lists. See [@file lists](#)^[49] for details.

/B (Binary) 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 unless source files are appended to the target file, or the target is a device, e.g., NUL or PRN.

This option applies to the filename immediately preceding it, and to all subsequent filenames until the file name preceding the next **/A** or **/B** option.

- /C** (Changed files) Copy files only if the **destination** file exists and is older than the **source** (see also [/U](#)^[164]). 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](#)^[164]. Do not use **/C** with @file lists. See [@file lists](#)^[49] for details.
- /D** (XP+ Only) Force copy of an encrypted file even when the target will be decrypted (for CMD.EXE compatibility).
- /E** (No error messages) Suppress all non-fatal error messages, such as "File not found" or "Can't copy file to itself". 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 percentage copied. Useful when copying large files across a network or via FTP to ensure the copy is proceeding. When [/V](#)^[164] is also used, reports percentage verified.
- /H** (Hidden) Copy all matching files including those with the hidden and/or system attribute set. See the cautionary note under **Advanced Features** above before using **/H** when both **source** and **destination** directories contain file descriptions.
- /I"text"** (Match descriptions) Select **source** files by matching text in their descriptions. The text can include [wildcards](#)^[36] 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"[]"**. Do not use **/I** with @file lists. See [@file lists](#)^[49] for details.
- /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** (Keep attribute) To maintain compatibility with **CMD.EXE**, **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 will fail to maintain the read-only attribute. This is due to the way NetWare handles file attributes, and is not a problem in **COPY**.
- /L** Perform FTP transfers in ASCII mode, instead of the default binary mode.
- /M** (Modified) Copy only those files with the archive attribute set, *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](#)^[164] switch, or use [ATTRIB](#)^[146]. Do not use **/M** with @file lists. See [@file lists](#)^[49] for details.
- /N** (Nothing) Do everything except actually perform the copy. This option is useful for testing what the result of a complex **COPY** command will be. **/N** displays how many files would be copied. **/N** does **not** prevent creation of **destination** subdirectories when it is used with [/S](#)^[164].
- /O** Only copy the source file if the target file doesn't exist.
- /P** (Prompt) Ask the user to confirm each **source** file. Your options at the prompt are

explained in detail under [Page and File Prompts](#)^[63]. Note: the [CopyPrompt](#)^[10] directive can be used to force prompting at the command line only. Also see the [/Q](#)^[16] option below.

- /Q** (Quiet) Don't display filenames, percentage copied, total number of files copied, etc... When used in combination with the [/P](#)^[16] option above, it will prompt for filenames but will not display the totals. This option is most often used in batch files. See also [/T](#)^[16].
- /R** (Replace) Prompt the user before overwriting an existing file. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[63]. Also see the [CopyPrompt](#)^[10] directive.
- /S** (Subdirectories) 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](#)^[56] database.


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 [@file lists](#)^[49] for details.
- /T** (Totals) Turns off the display of filenames, like [/Q](#)^[16], but does display the total number of files copied.
- /U** (Update) 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](#)^[16]). This option is useful for keeping one directory matched with another with a minimum of copying. Do not use /U with @file lists. See [@file lists](#)^[49] for details. When used with file systems that have different time resolutions (such as FAT and NTFS), **/U** will attempt to use the "coarsest" resolution of the two.
- /V** (Verify) Verify each disk write by performing a true byte-by-byte comparison between the source and the newly-created target file. This option may significantly increase the time necessary to complete a COPY command.
- /X** (Clear archive) 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) 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.

4.20 DATE

Purpose: Display and optionally change the system date.

Format: DATE [/T] [*mm -dd -yy*]

- /T** (Display only)
- mm:** The month (1 - 12).
- dd:** The day (1 - 31).
- yy:** The year (80 - 99 or a 4- digit year).

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
Thu Sep 30, 2004 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 1-16-2005
```

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.


4.21 DDEEXEC

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 the [DDE Support](#)  topic for details.) For example, if you wanted to instruct Microsoft Internet Explorer to open JP Software's web site, you could use this command:

```
[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.

4.22 DEL / ERASE

Purpose: Erase one file, a group of files, or entire subdirectories.

Format: DEL [/A:[-+]*rhsad*] /E /F /I"*text*" /K /N /P /Q /R /S /T /W /X /Y /Z] [*@file*] *file*...
or
ERASE [/A:[-]*rhsad*] /E /F /I"*text*" /K /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 [@file lists](#)^[49] for details).

/A: (Attribute select)	/R (ecycle bin)
/E (No error messages)	/S (ubdirectories)
/F (orce delete)	/T (otal)
/I " <i>text</i> " (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)	

File Selection:

Supports [attribute switches](#)^[39], extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46]. Use wildcards with caution on LFN volumes; see [LFN File Searches](#)^[47] for details.

Internet: Can be used with [FTP/HTTP Servers](#)^[52].

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 impossible to recover without specialized utilities and a lot of work.

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.

To exclude files from a DEL command, use a [file exclusion range](#)^[43]. 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). NOTE: The Windows 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 the command processor to follow *CMD.EXE*'s approach and skip the confirmation prompt when **/Q** is used, set [DelGlobalQuery](#)^[102] to No in the [.INI file](#)^[89]. 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. This applies to the use of DEL with the Windows Recycle Bin, too - the description will be lost.

• Recycle Bin

When you delete files with DEL, 4NT and Take Command by default do **not** 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](#)^[109] directive in the [.INI file](#)^[89]. If you have disabled the recycle bin, you can override the setting and place deleted files in the recycle bin with the **/R** option:

```
[c:\] del /r letters.txt
```

If you have disabled Recycle Bin support, but want to override the default setting on a one-time basis, and delete some files without placing them in the recycle bin, use the **/K** option:

```
[c:\] del /k letters.txt
```

You can also exclude files from the Recycle bin (if RecycleBin=Yes is set, or if you use the **/R** option) with the [RecycleExclude](#)^[34] environment variable.

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 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 [%_?](#)^[34b] internal variable, and use it with [conditional commands](#)^[26] (**&&** and **||**).

Use caution when using wildcards with DEL on LFN drives, because the command processor's wildcard matching will match both short and long filenames. This can delete files you did not expect; see [LFN File Searches](#)^[47] for additional details.

• FTP Usage

If you have appropriate permissions, you can delete files on [FTP servers](#)^[52]. The FTP name must be enclosed in quote marks so the forward slashes won't be interpreted as switches. For example:

```
[c:\] del "ftp://ftp.abc.com/index"
```

You can also use the [IFTP](#)^[229] command to start an FTP session on a server and then use one of the following syntax examples:

```
[c:\] del "ftp:path/*.txt"
[c:\] del "ftp:/path/*.txt"
```

The first syntax will normally be interpreted by the server as relative to the path you specified when you used the IFTP command to start the FTP session. The second syntax, with a slash before the path name, is interpreted as starting from the root.

• NTFS File Streams

DEL supports file streams on NTFS drives under Windows 2000 and above. 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](#)^[444] for additional details.

Options:

- /A:** (Attribute select) Delete only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[39] for information on the attributes which can follow **/A:**. Do not use **/A:** with @file lists. See [@file lists](#)^[49] for details.
- /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.
- /F** (Force 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"** (Match descriptions) Select files by matching text in their descriptions. The text can include [wildcards](#)^[36] 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"[]"**. Do not use **/I** with @file lists. See [@file lists](#)^[49] for details
- /K** Physically delete files instead of sending them to the Windows Recycle Bin. This option overrides the default "[RecycleBin=Yes](#)^[109]" *.INI* setting.
- /N** (Nothing) Do everything except actually delete the file(s). This is useful for testing what the result of a DEL would be.
- /P** (Prompt) Prompt the user to confirm each erasure. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[68].
- /Q** (Quiet) Don't display filenames as they are deleted, or the number of files deleted or bytes freed. If [DelGlobalQuery](#)^[102] is set to No in the *.INI file*^[89] then **/Q** also disables the normal confirmation prompt when performing wildcard deletions (e.g. DEL *.*), for compatibility with the Windows 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**.

- /R** Delete files to the Windows Recycle Bin. This option overrides a "[RecycleBin=No](#)^[109]" *.INI* setting.
- /S** (Subdirectories) Delete the specified files in this directory and all of its subdirectories. This is like a GLOBAL DEL, and can be used to delete all the files in a subdirectory tree or even a whole disk. Do not use /S with @file lists. See [@file lists](#)^[49] for details.
- /T** (Total) 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** (Wipe) 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 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. **/W** will override a **/R** or a "[RecycleBin = Yes](#)^[109]" *.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](#)^[56].
- /Y** (Yes) The reverse of **/P** — it assumes a **Y** response to everything, including deleting an entire subdirectory tree. The command processor normally prompts 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** (Zap) 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](#)^[204] works by hiding files, **/Z** will override an EXCEPT command. However, files specified in a [file exclusion range](#)^[43] 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\
```

4.23 DELAY

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 batch file checks the battery status (as reported by your Advanced Power Management drivers) every 15 seconds, and gives a warning when battery life falls below 30%:

```
do forever
  iff %_apmlife 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 about 1 billion seconds (34 years!).

The command processor uses 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** (Break) Allows terminating a DELAY by pressing a key.
- /M** (Milliseconds) Count by milliseconds instead of seconds. Normally only used for delays of less than 1 second.

4.24 DESCRIBE

Purpose: Create, modify, or delete file and subdirectory descriptions.

Format: DESCRIBE [/A:[-][+]*rhsad*] /I"text" [*@file* *file* [/D]"description"] ...] [/U
[[d:\path\descript.ion] ...]]

file: The file or files to operate on.

@file: A text file containing the names of the files to describe, one per line (see [@file lists](#)^[49] for details).

"description": The description to attach to the file.

/A: (Attribute select)

/I (match description)

/D(escription follows)

/U(pdate) descriptions file

File Selection

Supports [attribute switches](#)^[39], extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46]. Use wildcards with caution on LFN volumes; see [LFN File Searches](#)^[47] for details.

Usage:

DESCRIBE adds descriptions to files and subdirectories. The descriptions are 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 with the [Descriptions dialog](#)^[74]. 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.

When you use DESCRIBE on an LFN drive, you must quote the **file** name if it contains white space or special characters.

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 "Misc" tab of the configuration dialogs, or with the [DescriptionMax](#)^[102] directive in the [.INI file](#)^[89]. 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](#)^[146] 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 rehidden by the command processor if you remove the hidden attribute.

You can change the description file name with the [DescriptionName](#)^[102] directive in the [.INI file](#)^[89] or the [SETDOS /D](#)^[283] command, and retrieve it with the [% DNAME](#)^[357] internal variable. 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](#)^[164], [MOVE](#)^[246], [DEL](#)^[172], or [RENAME](#)^[276]), 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 dialogs](#)^[82], with the [Descriptions](#)^[102] directive in the [.INI file](#)^[89], or with [SETDOS /D](#)^[283].

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](#)^[164] and [MOVE](#)^[246] for additional details.

Options:

- /A:** (Attribute select) Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[39] for information on the attributes which can follow **/A:**. Do not use **/A:** with @file lists. See [@file lists](#)^[49] for details.
- /D** (Description 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"** (Match descriptions) Select files by matching text in their descriptions. The text can include [wildcards](#)^[36] 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"[]"**. Do not use **/I** with @file lists. See [@file lists](#)^[49] for details.
- /U** (Update DESCRIPT.ION file) Update the DESCRIPT.ION file (or the file specified by the [DescriptionName](#)^[102] directive), deleting the entries for any nonexistent files. If no filename is supplied, DESCRIBE will process DESCRIPT.ION in the current directory. Otherwise, DESCRIBE will process DESCRIPT.ION in the specified path(s). This option may not be used in conjunction with other DESCRIBE options.

4.25 DETACH

Purpose: Start a console (character-mode) 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](#)^[29] and [TASKEND](#)^[298].

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](#)^[61] symbols. 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, the command processor returns to the prompt immediately. It does not wait for a detached program to finish.

The Process ID of the detached program is returned in the [DETACHPID](#)^[351] variable.

You can use the [TASKEND](#)^[298] command to stop a detached program which does not terminate on its own..

4.26 DIR

Purpose: Display information about files and subdirectories.

Format: DIR [/1] [/2] [/4] [/A[[:][:]]rhsad]] [/B] [/C] [/D] [/E] [/F] [/G] [/H] [/I"text"] [/J] [/K] [/L] [/M] [/N] [/O[[:][:]]adeginrsu]] [/P] [/Q] [/R] [/S] [/T[:acw]] [/U] [/V] [/W] [/X] [/Z] [:] [file...]

file: The file, directory, or list of files or directories to display.

/1 (one column)	/L (ower case)
/2 (two columns)	/M (suppress footer)
/4 (four columns)	/N (ew format)
/: (show streams)	/O (rder)
/A (ttribute select)	/P (ause)
/B (are)	/Q (show owner)
/C (ompression)	/R (disable wRap)
/D (isable color coding)	/S (ubdirectories)
/E (upper case)	/T (aTtribute) or (Time)
/F (ull path)	/U (sUmmary information)
/G (allocated size)	/V (ertical sort)
/H (ide dots)	/W (ide)
/I"text" (match description)	/X (display short names)
/J (ustify names)	/Z (use FAT format)
/K (suppress header)	

See also: [PDIR](#)^[256], [ATTRIB](#)^[146], [DESCRIBE](#)^[176], [SELECT](#)^[275], and [SETDOS](#)^[283].

File Selection

Supports extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46].

Internet: Can be used with [FTP servers](#)^[52].

Usage:

DIR can be used to display information about files from one or more of your directories (local or remote), 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, or 5 columns, sort the files several different ways, use color to distinguish file types, and pause after each full screen.

If you want to produce customized output that will be subsequently parsed by another program or batch file, or if you need a special-purpose directory display, see the [PDIR](#)^[256] command. DIR and PDIR are related, but they are not the same command. They do not have identical switches and they are not

intended to produce identical output.

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](#)^[138] 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.

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](#)^[179]
- ▶ [Default DIR Output Format](#)^[179]
- ▶ [Switching Formats](#)^[179]
- ▶ [Multiple Column Displays](#)^[179]
- ▶ [Color-Coded Directories](#)^[179]
- ▶ [Redirected Output](#)^[179]
- ▶ [Other Notes](#)^[179]
- ▶ [Options](#)^[179]
- ▶ [FTP usage](#)^[179]

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 the normal wildcard characters (* and ?) and the [extended wildcards](#)^[36]. 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 LFN drives, and *.* on drives which do not support long file names. 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](#)^[46] 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 [DESCRIBE](#)^[17b] 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 wild cards 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](#)^[40] 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 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 D is APPS      Serial ...
```

```
Directory of D:\4NT\4NT\*
```

```
8-24-2004 12:17      <DIR>      .
8-24-2004 12:17      <DIR>      ..
8-21-2004 18:08      212,854    4NT.EXE
8-02-2004 10:08              45    4NT.INI
```

(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:\*. *

.                <DIR>          8-24-04  12:17
..               <DIR>          8-24-04  12:17
TEST             <DIR>          8-01-04  16:21
jpstree.idx      967          8-28-04  17:57 JP fuzzy directory index
```

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.

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 describe 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** together, 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
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.

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 FAT-format display, with long names truncated to fit in the available space. If you use **/X**, the 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 when using the command processor on an LFN drive:

	default columns	/2 or /4 columns	/W (wide)
Normal	1 column, LFN plus date, time, size	2 - 4 columns, LFNs only	No. of columns based on longest LFN
/Z (FAT)	1 column, truncated LFN plus date, time, size	2 - 4 columns, truncated LFN plus date, time, size	5+ columns, truncated LFNs only
/X (alt. name)	1 column, SFN and LFN, plus date, time, size	2 - 4 columns, SFNs plus date, time, size	5+ columns, SFNs only
/Z /X	1 column, SFN plus date, time, size	(Same as /X)	(Same as /X)

Color-Coded Directories

The DIR command can display each file name and the associated file information in a different color, depending on the file's extension.

To choose the display colors, you must either use the [SET](#) ^[28] command to create an environment variable called [COLORDIR](#) ^[33], use the [configuration dialogs](#) ^[82], or use the [ColorDir directive](#) ^[11] in the [.INI file](#) ^[89]. If you use neither the COLORDIR variable nor 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](#) ^[89]. 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 is:

```
ext ... :ColorName; ...
```

where "ext" is either a file extension (which may include wildcards) or one of the following file types:

type	files affected
ARCHIVE	Files with archive attribute set (usu. modified since the last backup)
COMPRESSED	Compressed files
DIRS	Directories
ENCRYPTED	Encrypted files
HIDDEN	Hidden files
JUNCTION	Files which are junctions
NORMAL	File with no attribute set
NOTINDEXED	Files whose content is not indexed
OFFLINE	Offline files
RDONLY	Read-only files
SPARSE	Sparse files
SYSTEM	System files
TEMPORARY	temporary files

and "ColorName" is any valid color name (see [Colors and Color Names](#) ^[46] for information on color names).

Note that if a file uses one of the reserved file type names shown above as its extension (e.g.

"foo.hidden") , that file will receive the color defined for the file type.

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:

```
[c:\] set colordir=com exe:red; c asm:bright cyan; ronly:green on white
```

[Extended wildcards](#)^[36] can be used in directory color specifications. 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](#)^[67] 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:
```

FTP Usage

You can display directories on [FTP servers](#)^[52]. The URL must be enclosed in quote marks. For example:

```
[c:\] dir "ftp://jpsoft.com/4nt"
```

You can also use the [IFTP](#)^[229] command to start an FTP session on a server, and then use a simplified syntax to specify the files and directories you want.

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 with the configuration dialogs or in the [.INI file](#)^[89].

DIR can generally display any file date between January 1, 1980 and December 31, 2099 if the date is supplied properly by the operating system.

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. Only compressed NTFS drives are supported. **/C** and **/O:c** will be ignored for other compression programs, and on uncompressed drives. **/C** will not display compression ratios on drives that support long file names unless you also use **/Z** to switch to the old-style short filename format.

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** above 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** above for more details.
- /:** Display file stream names and sizes on NTFS volumes. When combined with the **/B** or **/F** options, the size is omitted.

When **/:** is used in conjunction with **/B** (Bare), the file name is displayed on the first line, then any streams, indented two spaces, on subsequent lines:

```
c:\test\myfile.dat
  foo:$DATA
  bar:$DATA
```

When **/:** is used in conjunction with **/F** (Full path), the file name is displayed on the first line, then any streams are appended to the filename on subsequent lines:

```
c:\test\myfile.dat
c:\test\myfile.dat:foo
c:\test\myfile.dat:bar
```

- /A[:]** **(Attribute select)** Display only those files that have the specified attribute(s) set. See [Attribute Switches](#) ³⁹¹ for information on the attributes which can follow **/A:**.
- /B** **(Bare)** 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. **/B** also sets **/H**.
- /C** **(Compression)** 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 unless you use **/Z** to switch to the short filename format.

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 8KB). 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.

- /D** (**Disable** 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.
- /F** (**Full** path) Display each filename with its drive letter and path in a single column, without other information. If you use **/F** with **/X**, the "short" version of the entire path is displayed.
- /G** (Allocated space) Display the allocated disk space instead of the actual size of each file.
- /H** (Hide dots) Suppress the display of the "." and ".." directories.
- /I"text"** (Match descriptions) Select filenames by matching text in their descriptions. The text can include [wildcards](#)^[36] 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** (**Justify** names) Justify (align) filename extensions and display them in the FAT 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** (**Lower** case) Display file and directory names in lower case.
- /M** Suppress the footer (file and byte count totals) display.
- /N** (**New** format) 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** (**Order**) Set the sorting order. You may use any combination of the sorting options below. 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:
 - n** Sort by filename and extension, unless **e** is explicitly included. This is the default.
 - Reverse the sort order for the next option
 - a** Sort names and extensions in standard ASCII order, rather than sorting by magnitude when numeric substrings 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 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. For information on supported compression systems see **/C** above.
- d** Sort by date and time (oldest first); also see **/T:acw**
- e** Sort by extension
- g** Group subdirectories first, then files
- i** Sort by file description (ignored if **/C** or **/O:c** is also used).
- r** Reverse the sort order for all options
- s** Sort by size
- u** Unsorted

/P (**Pause**) Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail under [Page and File Prompts](#) ^[65].

/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 (**Subdirectories**) 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. DIR will display hidden subdirectories for compatibility with CMD.EXE.

/T (aTtribute display) Display the filenames and attributes in the format *RHSADENTPCOI*, regardless of volume type or Windows version:

R Read-only	A Archive
H Hidden	D Subdirectory
S System	
E Encrypted	C Compressed
N Normal	O Offline
T Temporary	I Not content-indexed
P Sparse file	J Junction (reparse point)

Attributes which are set are represented by their letter, unset attributes by the _ (underscore) character.

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:

```
[c:\] dir /tz          incorrect, will display error
[c:\] dir /t/z        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:

- a** Last access date and time (access time is only saved on NTFS volumes).

- c** Creation date and time.
- w** Last write date and time (default).

- /U** (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** (**Vertical** sort) Display the filenames sorted vertically rather than horizontally (use with the **/2**, **/4** or **/W** options).
- /W** (**Wide**) 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** above 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 LFN drives in the old-style format, with the filename on the left and the description (when available) 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" symbol to show that one or more characters have been truncated. If a [description file](#)^[102] exists, **/Z** defaults to using the name of the "." and ".." directories as description for those entries

4.27 DIRHISTORY

Purpose: Display, add to, clear, or read the directory history list.

Format: DIRHISTORY [/A *directory* /F /N /P /R *filename*]

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)
/N (o duplicates)	

See also: [HISTORY](#)^[226].

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 the [directory history window](#)^[23], which allows you to use the list to return to a previous directory. Also see [directory navigation](#)^[54].

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 (see the [History Options Tab](#)^[85] or the [DirHistory](#)^[94] directive in the [INI file](#)^[89]). The default size is 1,024 characters.

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 history lists](#)^[16].

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 [TCSTART.BTM](#)^[6] (**TC**) or [4START.BTM](#)^[6] (**4NT**) or other startup file, and the second in [TCEXIT.BTM](#)^[6] (**TC**) or [4EXIT.BTM](#)^[6] (**4NT**):

```
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** (Add): Add a directory to the directory history list.
- /F** (Free): Erase all entries in the directory history list.
- /N** (No duplicates): Removes duplicate entries (oldest first) from the directory history list.
- /P** (Prompt): Wait for a key after displaying each page of the list. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[65].
- /R** (Read): Read the directory history from the specified file and append it to the list currently held in memory.

4.28 DIRS

Purpose: Display the current directory stack.

Format: DIRS

See also: [PUSHD](#)^[264], [POPD](#)^[261], [@DIRSTACK](#)^[37] and [Directory Navigation](#)^[54].

Usage:

The PUSH command adds the current default drive and directory to the directory stack, a list that the command processor maintains in memory. The POP 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 POP 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.

4.29 DO

Purpose: Create loops in batch files.

Format: DO *loop_control*
commands
 [ITERATE^[19h]]
commands
 [LEAVE^[19h]]
commands
 ENDDO

Loop_control formats:

```
DO [19h]count  

DO FOREVER [19h]  

DO [19h]varname = start TO end [BY step]  

DO WHILE [19h]condition  

DO UNTIL [19h]condition  

DO UNTIL DATETIME [19h]date time  

DO [19h]varname IN [range] [/I:"text"] [/A:[-][+][19h]rhsad] filesset  

DO [19h]varname IN /L stringset  

DO [19h]varname IN @file
```


count	Integer in the range [0, 2 147 483 647], or an internal variable or variable function that evaluates to such a value, specifying the number of times the loop is executed.
varname	The environment variable containing the current value of the loop index, or the current filename or string, or the current line from a file. Do not prefix with %.
start, end, step:	Integers in the range [-2 147 483 647, 2 147 483 647] or internal variable or variable functions that evaluate to such values, controlling the number of times the loop is executed.
condition	A conditional expression ^[37] to determine whether or not the loop should be executed
fileset:	A filename or list of filenames, possibly using wildcards ^[36]
stringset	An arbitrary set of strings. Wildcards are not interpreted.
file	A file each line of which contains a string the loop is to be executed for
range ^[40]	A date, time, size or exclusion range. At most one of each, in any order.
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.
date	The loop termination date in ISO 9601 format
time	The loop termination time in 24-h hh:mm:ss format
/A: ^[19]	(Attribute select)
/I"text" ^[19]	(match description)
/L(iteral) ^[19]	members of set are strings, not filenames

Supports extended [wildcards](#)^[36], [ranges](#)^[40], and [include lists](#)^[46] for the **set**. Use wildcards with caution on LFN volumes; see [LFN File Searches](#)^[47] for details.

Usage:

DO can only be used in [batch files](#)^[32].

Types of DO Loops

DO can be used to create several different kinds of loops.

- **DO count**, is a *counted* loop. The batch file lines between **DO** and **ENDDO** are repeated **count** times. *The command processor does not provide the user with the count of how many times the loop has been executed, though it is possible for the user to create a such a mechanism.* **Example:**

```
set ct=0
do 5
    beep
    set ct=%@inc[%ct]
enddo
```

- **DO FOREVER** creates an endless loop. You must use [LEAVE](#)^[19] or [GOTO](#)^[22] to exit such a loop.
- **DO varname = start TO end [BY step]** 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 **step** is positive or not specified, and **varname** is less than or equal to **end**, **DO** executes the batch file lines up to the **ENDDO**. Next, **DO** adds to the value of **varname** either the value of **step** if **BY step** is specified, or 1, 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 **count** is negative, **varname** will be decreased by the absolute value of **count** 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
```

- **DO WHILE condition** evaluates **condition** each time through the loop as a [conditional expression](#)^[37] **before** executing the loop, and will execute it only if it is true. If **condition** is FALSE when the DO is first executed, the loop will never be executed.
- **DO UNTIL condition** evaluates **condition** as a [conditional expression](#)^[37] each time **after** execution of the loop, and repeats the loop only if it is FALSE. Therefore, the statements within the loop will always be executed at least once.
- **DO UNTIL DATETIME date time** executes the loop until the current date and time is equal to or greater than the specified date (ISO format) and time (24-hour format).
- **DO varname IN fileset** executes the commands between DO and ENDDO by creating an environment variable, **varname**, and setting it equal to every filename in the **fileset**, IGNORING items not matching filenames. This is similar to the **set** used in the [FOR](#)^[21b] command, but it can only include filenames, not simple text strings. If **varname** already exists in the environment, *it will be overwritten* (unlike the control variable in [FOR](#)^[21b]). For example:

```
do x in *.txt
  ...
enddo
```

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. The order of matches is dependent on the file system, and is totally unrelated to any characteristics of the filenames matched.

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.

To use date, time, size, or file exclusion [ranges](#)^[40] for the **set** place them just before the filename(s), for example:

```
do x in /[d9-1-2004,9-31-2004] *.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 [Redirection](#)^[67] for more information on **CLIP:**.

- **DO varname IN /L stringset** executes the commands between DO and ENDDO once for every string literal in **stringset**, setting **varname** to each in turn.
- **DO varname IN @file** executes the commands between DO and ENDDO once for every line in

file, setting *varname* to the content of each one in turn. **Beware of characters with special meaning to the command processor, such as redirection and piping symbols, within the file.** Use [SETDOS](#) [283] /X with appropriate codes as needed.

Special DO keywords: ITERATE and LEAVE

Two special keywords, **ITERATE** and **LEAVE**, may (but need not) be used inside a **DO** / **ENDDO** loop. **ITERATE** ignores the remaining commands 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 command following its **ENDDO**. Both keywords may be repeated as often as desired. Both **ITERATE** and **LEAVE** are most often used in an [IF](#) [227] or [IFF](#) [228] command (group):

```
do while "%var" != "%val1"
...
  if "%var" == "%val2" leave
enddo
```

Usage Notes

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

DO loops can be nested, i.e. you can have a **DO/ENDDO** loop within another **DO/ENDDO** loop.

The **DO** and **ENDDO** commands must be on separate lines, and cannot be placed within a [command group](#) [271], or on the same line as other commands. This is the reason **DO** loops cannot be used in aliases. However, commands within the **DO** loop can use command groups or the command separator in the normal way. For example, the following command will not work properly, because the **DO** and **ENDDO** are inside a command group and are not on separate lines:

```
if "%a" == "%b" (do i = 1 to 10 %+ echo %i %+ enddo)
```

However, this batch file fragment uses multiple commands and command grouping within the **DO**, and will work properly:

```
do i = 1 to 10
...
  if "%x1" == "%x2" (echo Done! %+ leave)
...
enddo
```

You can exit from all **DO** / **ENDDO** loops by using [GOTO](#) [222] to a line past the corresponding **ENDDO**. However, be sure to read the cautionary notes about [GOTO](#) [222] and **DO** under the [GOTO](#) [222] command before using **GOTO** in any other way inside any **DO** loop.

You cannot use [RETURN](#) [272] to return from a [GOSUB](#) [221] while inside a **DO** loop.

Options:

/A: (Attribute select) Select the files in a [DO](#) [191]X **IN** ... by their specified attribute(s). See [Attribute Switches](#) [39] for information on the attributes which can follow **/A:**.

/I"text" (Match descriptions) Select files in a [DO](#) [191]X **IN** ... by matching *text* in their descriptions. The text can include [wildcards](#) [36] 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 (Literal) The arguments following `DO`^[197] `x` `IN` `/L` are strings, not filenames. Each argument will be assigned in sequence, from left to right, to the loop control variable on consecutive passes through the loop.

Note: Do not confuse the `DO` command with the unrelated optional `do` keyword of the `FOR`^[210] command.

4.30 DRAWBOX

Purpose: Draw a box on the screen.

Format: `DRAWBOX ulrow ulcol lrow lcol style [BRlght] fg ON [BRlght] bg [FILL [BRlght] 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:
 0 No lines (box is drawn with blanks)
 1 Single line
 2 Double line
 3 Single line on top and bottom, double on sides
 4 Double line on top and bottom, single on sides
fg: Foreground character color
bg: Background character color
bgfill: Background fill color (for the inside of the box)

See also: [DRAWHLINE](#)^[196] and [DRAWVLINE](#)^[197].

Usage:

`DRAWBOX` is useful for creating attractive screen displays in batch files.

For example, to draw a box around 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 [Colors and Color Names](#)^[467] for details about 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 standard 25 line by 80 column display, valid rows are 0 - 24 and valid columns are 0 - 79.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format

("0x" followed by a sequence of 0-F hex digits). DRAWBOX checks for valid row and column values, and displays a "Usage" error message if any values are out of range.

(TC) The maximum **row** value is determined by the current height of the Take Command window. The maximum **column** value is determined by the current virtual screen width (see [Resizing the Take Command Window](#)^[77] 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.

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.

DRAWBOX uses 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.

(TC) They will only appear correctly if you have configured Take Command to use a font, such as Terminal, which contains standard extended ASCII characters.

4.31 DRAWHLINE

Purpose: Draw a horizontal line on the screen.

Format: DRAWHLINE *row column len style* [BRlght] *fg* ON [BRlght] *bg*

row: Starting row
column: Starting column
len: Length of line
style: Line drawing style:
 1 Single line
 2 Double line
fg: Foreground character color
bg: Background character color

See also: [DRAWBOX](#)^[195] and [DRAWVLINE](#)^[197].

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.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). If either value is out of range, DRAWHLINE displays a "Usage" error message.

(TC) The maximum *row* value is determined by the current height of the Take Command window. The maximum *column* value is determined by the current virtual screen width (see [Resizing the Take Command Window](#)^[77] 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 [Colors and Color Names](#)^[461] for details about colors.

DRAWHLINE uses 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.

(TC) They will only appear correctly if you have configured Take Command to use a font, such as Terminal, which contains standard extended ASCII characters.

4.32 DRAWVLINE

Purpose: Draw a vertical line on the screen.

Format: DRAWVLINE *row column len style* [BRlght] *fg* ON [BRlght] *bg*

row: Starting row
column: Starting column
len: Length of line
style: Line drawing style:
 1 Single line
 2 Double line
fg: Foreground character color
bg: Background character color

See also: [DRAWBOX](#)^[195] and [DRAWHLINE](#)^[196].

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. Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). If either value is out of range, DRAWVLINE displays a "Usage" error message.

(TC) The maximum *row* value is determined by the current height of the Take Command window. The maximum *column* value is determined by the current virtual screen width (see [Resizing the Take Command Window](#)^[77] for more information).

See [Colors and Color Names](#)^[461] for details about colors.

DRAWVLINE uses 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.

(TC) They will only appear correctly if you have configured Take Command to use a font, such as Terminal, which contains standard extended ASCII characters.

4.33 ECHO

Purpose: Display a message, enable or disable batch file or command-line echoing, or display the echo status.

Format: `ECHO [ON | OFF | message]`

message: Text to display.

See also: commands [ECHOS](#)^[199], [ECHOSERR](#)^[200], [ECHOERR](#)^[199], [SCREEN](#)^[273], [SCRPUT](#)^[274], [TEXT](#)^[301], and [VSCRPUT](#)^[314], and internal variable [_ECHO](#)^[352].

Usage:

The command processor has 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. Alternately, you can examine the value of the internal variable [_ECHO](#)^[352].

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 both at the command prompt and in a batch file to display a message on the current output device regardless of the applicable **ECHO** state. A command line that begins with the @ character will not be displayed regardless of the applicable **ECHO** state. To turn off batch file echoing, without displaying the **ECHO** command itself, use this line:

```
@echo off
```

This line will display a message in a batch file:

```
echo Processing your print files...
```

The first space after the **ECHO** command and trailing spaces in *message* are ignored. To display them, use back quotes. For example:

```
echo    This text is indented 3 spaces    ``
```

will display 3 leading and 3 trailing spaces.

If you want to echo a blank line from within a batch file, enter:

```
echo ``
```

(two consecutive back quotes), or

```
echo.
```

(special syntax for *compatibility with CMD.EXE*).

You cannot use the [command separator](#)^[100] character [**&**], or the [redirection](#)^[61] symbols [**>** **<**] in *message*, unless you enclose them in quotes (see [Argument Quoting](#)^[332]) or precede them with the [escape character](#)^[21], or use the **/X** option of the [SETDOS](#)^[283] command.

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](#)^[283] command, the [configuration dialogs](#)^[82], or the [BatchEcho](#)^[322] directive in the [.INI file](#)^[89].

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.

See [Redirection](#)^[61] for more information about the standard output and standard error devices.

4.34 ECHOERR

Purpose: Display a message to the standard error device (stderr).

Format: ECHOERR *message*

message: Text to display.

See also: [ECHO](#)^[198], [ECHOS](#)^[199], [ECHOSERR](#)^[200].

Usage:

ECHOERR acts like [ECHO](#)^[198] 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 [Redirection](#)^[61] for more information about the standard output and standard error devices.

4.35 ECHOS

Purpose: Display a message without a trailing carriage return and line feed.

Format: ECHOS *message*

message: Text to display.

See also: [ECHO](#)^[198], [ECHOERR](#)^[199], [ECHOSERR](#)^[200], [SCREEN](#)^[273], [SCRPUT](#)^[274], [TEXT](#)^[301], and [VSCRPUT](#)^[314].

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 (**%=**^[347] is translated to the command processor escape character, and **%=e**^[347] to an ASCII Esc).

```
[c:\] echos %=eP > lpt1:
```

You cannot use the [command separator](#)^[100] character [**&**], or the [redirection](#)^[61] symbols [**>** **<**] in an ECHOS message, unless you enclose them in quotes (see [Argument Quoting](#)^[332]) or precede them with the [escape character](#)^[21].

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).

See [Redirection](#)^[61] for more information about the standard output and standard error devices.

4.36 ECHOSERR

Purpose: Display a message to the standard error device (stderr) without a trailing carriage return and line feed.

Format: ECHOSERR *message*

message: Text to display.

See also: [ECHO](#)^[198], [ECHOS](#)^[199], [ECHOERR](#)^[199]..

Usage:

ECHOSERR acts like [ECHOS](#)^[199] 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 [Redirection](#)^[61] for more information about the standard output and standard error devices.

4.37 ENDLOCAL

Purpose: Restore the saved disk drive, directory, environment, alias list, and special characters. Permit selected variables to be exported.

Format: ENDLOCAL [exportvar ...] [/D]

See also: [SETLOCAL](#)^[287].

Usage:

The [SETLOCAL](#)^[287] 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. It does NOT save the function list. ENDLOCAL restores everything that was saved by the previous SETLOCAL command, except as described below.

For example, this batch file fragment saves everything, removes all aliases so that user aliases will not affect batch file commands, 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 / ENDLOCAL may be nested within a single batch file up to 16 levels of nesting. You can also have multiple, separate SETLOCAL / ENDLOCAL pairs within a batch file, and nested batch files can each have their own SETLOCAL / ENDLOCAL.

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. If you invoke one batch file from another without using CALL, the first batch file is terminated, and an automatic ENDLOCAL is performed; the second batch file inherits the settings as they were prior to any SETLOCAL.

- **Exporting environment variables**

The environment variables whose names are specified in the ENDLOCAL command are exported. This means that their names and values from inside the SETLOCAL / ENDLOCAL will be placed into the restored environment, either adding variables, or possibly modifying them. In the example below, the variable TEST will have the value **abcd** after the ENDLOCAL is executed, regardless of what its value was, or even if it had not been previously defined:

```
setlocal
set test=abcd
endlocal test
```

The list of variables to export may contain wildcards. All variables matching the requested pattern will be exported.

- **Exporting current working directory**

See option /D below.

Options:

/D (Don't restore directory) Export the current directory: the original drive and directory saved by SETLOCAL will **not** be restored.

4.38 ESET

Purpose: Edit environment variables and aliases.

Format: ESET [/A /D /F /M /S /U /V /W] variable name...

variable name: The name of an environment variable or alias to edit.

/A (lias)	/S (ystem variable) in Registry
/D (efault environment) in Registry	/U (ser variable) in Registry
/F (unction)	/V (olatile variable) in Registry
/M (aster environment)	/W (indowed Editing)

See also: [ALIAS](#)^[138], [UNALIAS](#)^[310], [SET](#)^[284], and [UNSET](#)^[312].

Usage:

ESET allows you to edit environment variables, aliases, and functions using line editing commands (see [Command-Line Editing](#)^[11]).

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 and 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.

To edit variables defined in the Windows Registry or to edit functions, you must use the appropriate option switch.

Environment variable and alias names are limited to 80 characters. The total length of the name and value combined is limited by the maximum line length of the command processor (8,191 characters). If you use special techniques to create a longer environment variable, ESET will edit it, provided it contains no more than 8,191 characters.

Note: You cannot use ESET with [GOSUB variables](#)^[224].

If you have enabled global aliases (see [ALIAS](#)^[138]), 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. Similarly, if you have enabled global functions (see [FUNCTION](#)^[218]), any changes made to a function with ESET /F will immediately affect all other copies of the command processor which are using the same function list.

Registry Variables: **Default**, **System**, **User**, and **Volatile** registry variables can be manipulated with the ESET command's **/D**, **/S**, **/U** and **/V** switches respectively. For example, to edit volatile variable "myvar" from the registry, use:

```
[c:\] eset /v myvar
```

Use caution when directly modifying registry variables as they may be essential to various Windows processes and applications.

Options:

/A (Alias) 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 be able to edit the alias.

- /F** (Function) – Edit a user-defined function.
- /D** (Default environment) : Edit a "default" variable in the registry (HKU\DEFAULT\Environment).
- /S** (System) : Edit a "system" variable in the registry (HKLM\System\CurrentControlSet\Control\Session Manager\Environment).
- /U** (User) : Edit a "user" variable in the registry (HKCU\Environment).
- /V** (Volatile) : Edit a "volatile" variable in the registry (HKCU\Volatile Environment).
- /W** (Windowed) : Edit the variable, alias, or function list in a popup window like that used by the [Batch File Debugger](#)^[149]. Note that any variable name passed to ESET will be ignored when this option is used. Non-environment variables (/D, /S, /U, /V) may not be edited with this option.

4.39 EVENTLOG

Purpose: Write a string to the Windows NT / 2000 / XP / 2003 event log.

Format: EVENTLOG [/E] [/I] [/S source] [/W] message

message: the text to write.

source: the source for this message.

/E(rror)

/I(nformational)

/S(ource)

/W(arning)

See also: [HISTORY](#)^[226] and [LOG](#)^[242].

Usage:

EVENTLOG posts messages to the Windows NT / 2000 / XP / 2003 application event log. Each message can be a maximum of 8,191 characters long. You cannot use the [command separator](#)^[100] character ([&]) or the [redirection](#)^[60] symbols (| > <) in an EVENTLOG message, unless you enclose the message in [quotes](#)^[332] or precede the special characters with the [escape character](#)^[21].

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 Event Log viewer.

If you do not have proper registry permissions when you execute the EVENTLOG command and/or the key cannot be created, EVENTLOG will fail and display an error. EVENTLOG is primarily intended for use by users with "Administrator" status.

Options:

/E (Error): Store the message as an error entry in the event log.

/I (Informational): Store the message as an informational entry in the event log. This is the default if no switch is used.

/S (Source) Specify the event log entry source. (If the source contains white space, it must be double-quoted). For example:

```
eventlog /sCompiling /I Your message here.
```

/W (Warning): Store the message as a warning entry in the event log.

4.40 EXCEPT

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 [@file lists](#)^[49] for details).

command: The command to execute, including all appropriate arguments and switches.

/I (match description)

See also: [ATTRIB](#)^[146] and [File Exclusion Ranges](#)^[45].

File Selection

Supports extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46]. Date, time, size, or file exclusion ranges must appear immediately after the EXCEPT keyword.

Use wildcards with caution on LFN volumes; see [LFN File Searches](#)^[47] 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.

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 example copies all the files and subdirectories on drive C to drive D except those in *C:\WSC* and *C:\DOS*, using the COPY command:

```
[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. 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).

Caution: EXCEPT will not work with programs or commands that ignore the hidden attribute or which

work explicitly with hidden files, including [DEL](#)^[172] /Z, and the /H (process hidden files) switch available in some command processor file processing commands.

[File exclusion ranges](#)^[45] provide a faster and more flexible method of excluding files from internal commands, and do not manipulate file attributes, as EXCEPT does. However, exclusion ranges can only be used with internal commands; you must use EXCEPT for external 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](#)^[27] 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:

```
[c:\data] except (*.dat) (copy *.da* d:\save %+ ren *.da* *.sa*)
```

If you use filename completion (see [Filename Completion](#)^[17]) to enter the filenames inside the parentheses, type a space after the open parenthesis before entering a partial filename or pressing Tab. Otherwise, the command-line editor will treat the open parenthesis as the first character of the filename to be completed.

Option:

/I"text" (Match descriptions) Select files by matching text in their descriptions. The text can include [wildcards](#)^[36] 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"[]". Do not use /I with @file lists. See [@file lists](#)^[49] for details.

4.41 EXIT

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.

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 - 4,294,967,295.

Option:

/B Exit the current batch file in Windows 2000 and above, rather than the shell. This switch is for compatibility with *CMD.EXE*. The [CANCEL](#)^[158] and [QUIT](#)^[266] commands are generally more flexible for use in batch files.

4.42 FFIND

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] [/+n] [/n] file...`

***list*:** A list of disk drive letters (without colons).

***file*:** The file, directory, or list of files or directories to display.

/A (ttribute select)	/N (ot)
/B (are)	/O (rder)
/C (ase sensitive)	/P (ause)
/D (rive)	/R (everse search order)
/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 (verbose)
/K (no headers)	/X["xx"] (hex display / search string)
/L (ine numbers)	/Y (prompt to stop after match)
/M (no footers)	/[+ -] skip matches

File Selection

Supports extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46].

Internet: Can be used with [FTP Servers](#)^[52].

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 [Find Files / Text dialog](#)^[73], 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.

For example, 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" *v.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 the command processor's [extended wildcards](#)^[36] 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*. It will also find *Juny* and *Jule*. 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, for example, finds all .TXT files that contain a question mark:

```
[c:\] ffind /i/t"?" *.txt
```

At times, 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 (in the example below, 41 63 65 is the hex code for "Ace"):

```
[c:\] ffind /b"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.

FFIND can also find files on FTP servers. The URL must be enclosed in quote marks. 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/HTTP Servers](#)^[52] and [IFTP](#)^[229].

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: (Attribute select) Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[39] for information on the attributes which can follow **/A:**.

- /B** (Bare) 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** (Case 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** (Drive) 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, this command displays all the *.BTM* files on C: and E:, but only the *.BAT* files on D:
- ```
[c:\] ffind /s /dce *.btm d:\*.bat
```
- /E** (Upper case display) Display filenames in upper case.
- /F** Stops the search after the first match.
- /I** (Ignore wildcards) Only meaningful when used in conjunction with the **/T "text"** option. Suppresses the recognition of wildcard characters in the search text. This option is useful if you need to search for characters that would normally be interpreted as wildcards: *, ?, [, and].
- /I"text"** (Match descriptions) Select filenames by matching text in their descriptions. The text can include [wildcards](#)^[36] 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** (Line numbers) Include the line number for each text line displayed. FFIND numbers lines beginning with 1, unless [ListRowStart](#)^[106] is set to 0 in the [INI file](#)^[89]. 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 the [command line length limit](#)^[23], 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** (Not) Reverse the meaning of the search, i.e., report only files which contain no match. Setting **/N** will also set **/B**, i.e. searches are on a file-by-file basis; FFIND cannot search for all lines without match.
- /O** (Order) Set the sort order for the files that FFIND displays. 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)
- 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 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** (Pause) Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail under [Page and File Prompts](#).
- /R** (Reverse) 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** (Subdirectories) 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 quote marks (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** Display only the summary.
- /V** (All matching lines) Show every matching line. FFIND's default behavior is to show only the first matching line, then 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 in conjunction 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: 1A

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, and don't want to display all files which include a particular search string.
- /[+|-]n** **"/+n"** causes FFIND to skip the first **n** matches. **"/-n"** causes FFIND to stop after **n** matches.

4.43 FOR

Purpose: Repeat a command for several values of a variable.

Format: FOR [/A:[-][+]*rhsad*] /F [*options*] /H /I "*text*" /L /R [*path*] %*var* IN ([@]*set* | *start*, *step*, *end*) [DO] [(*command* ... [LEAVEFOR] [])]

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)

/F(ile parsing)

/L (counted loop)

/H(ide dots)

/R(ecursive)

File Selection

Supports [attribute switches](#)^[39], extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46]. Ranges must appear immediately after the FOR keyword.

Use wildcards with caution on LFN volumes; see [LFN File Searches](#)^[47] 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 [/]).

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

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*.

The first three sections below ([Working with Files](#)^[21b], [Working with Text](#)^[21b], and [Retrieving Text from Files](#)^[21b]) describe the FOR command and the enhancements to it which are included in the command processor. The sections on [Parsing Text from Files](#)^[21b] and [Counted FOR Loops](#)^[21b] describe features added for compatibility with *CMD.EXE*. The sections [Directory Recursion](#)^[21b] and [Output Redirection](#)^[21b] warn of special considerations. The section entitled [Other Notes](#)^[21b] 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) 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 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) type %x
```

FOR supports [wildcards and extended wildcards](#)^[36], as well as [extended parent directory](#)^[47] 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 command processor internal commands, or other commands which require quoting filenames with white space. FOR does not quote returned names automatically, even if you included quotes in the **set**.

If the set includes filenames, the file list can be further refined by using [date, time, size and file exclusion ranges](#)^[40]. 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 December 4, 2004:

```
for /[d12-4-2004,+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) 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 the general discussion of the [Environment](#)^[336] for details on the use of square brackets when expanding environment variables):

```
for %var in (path prompt comspec) 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) 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. If you use **@CLIP:** as the filename, FOR will read any text available from the Windows clipboard.

See [Redirection and Piping](#)^[60] for more information on these features.

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](#)^[38] as the **set** to retrieve each line from the file, as described in the previous section. Then use variable functions like [@INSTR](#)^[39], [@LEFT](#)^[39], [@RIGHT](#)^[40], and [@WORD](#)^[41] to parse the line (see [Variable Functions](#)^[35] 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) 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":

```
for /f "tokens=1,2,3,4" %a in ("this is a test") echo %a
```

"Counted" FOR Loop

The "counted FOR" loop is included only for compatibility with *CMD.EXE*. In most cases, you will find the [DO](#)^[19] command 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) echo %val
```

This example will print the odd numbers from 1 to 10 (1, 3, 5, 7, and 9):

```
for /L %val in (1,2,10) echo %val
```

The **step** value can be negative. If it is, the loop will end when the variable is less than the **end** value.

Directory Recursion

Option switch **/R** specifies that the search should recursively process subdirectories. If no directory is specified after the **/R**, search starts in the current default directory. If a directory is specified after the **/R** option switch, its name must be enclosed in quote marks if it includes any special characters, for example, if it is specified with the aid of an environment variable (e.g., %windir\command).

When FOR **/R** is used for recursing directories, it actually makes each eligible directory the current working directory in turn, and the loop is there executed. Consequently, the loop control variable will NOT reflect the current execution path. To make sure that files of the same name in different directories are correctly handled, you must utilize one of the variable functions which convert a relative path to an absolute one, e.g., @truenam[], @full[], etc.

Output Redirection

The default output redirection (i.e., `for ... > filename`) creates a new output file in each iteration. If `filename` does not include an absolute filepath, it will be created relative to the then current default directory. The simplest way to force a single target file is to enclose the whole command in parentheses, e.g.,:

```
(for %x in (set) command) > filename
```

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 (*CMD.EXE* requires 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.

Do not confuse the optional (and unnecessary) **DO** keyword that can be placed between the **"(set)"** and **"command"** portions of a **FOR** statement with the completely unrelated **DO**^[19] command.

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 variable that FOR uses is created in the environment and then erased when the FOR command is done. For compatibility with *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.

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:

```
for %p in (a: b:) 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.

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):

```
for %file in (*.txt) 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:

```
for %file in (*.txt) del "%@name[%file]".bak"
```

You can use [command grouping](#)^[27] 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:

```
[c:\text] for %file in (*.wkq) (copy %file d:\wksave\ %+ ren %file *.sav)
```

or (in a batch file):

```
for %file in (*.wkq) (
    copy %file d:\wksave\
    ren %file *.sav
)
```

In a batch file you can use [GOSUB](#)^[22] 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:

```
[c:\] for %cmd in (filetest fileform fileprnt) %cmd datafile
```

This line will expand to three separate commands:

```
filetest datafile
fileform datafile
fileprnt datafile
```

FOR statements can be nested.

Special FOR keywords: LEAVEFOR

The special keyword **LEAVEFOR** can be used inside a FOR command group. **LEAVEFOR** terminates the current FOR processing and continues with the line following the FOR command, in a manner similar to that of the LEAVE keyword in a [DO](#)^[194] command.

```
for %i in (*) (
    ...
    if "%i" == "foo.bar" leavefor
    ...
)
```

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](#)^[39] 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!
```

/F (File 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 (Hide dots) Suppress the assignment of the "." and ".." directories to the FOR variable.

/I"text" (Match descriptions) Select filenames by matching text in their descriptions. The text can include [wildcards](#)^[36] 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 [path] (Recursive) 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. **Warning:** if the directory name includes special characters, including "%" to indicate an environment variable, it must be enclosed in quote marks (").

4.44 FREE

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](#)^[245].

Usage:

A colon [:] is required after each drive letter. This example displays the status of drives A and C:

```
[c:\] free a: c:
```

If the volume serial number is available, it will appear after the drive label or name.

4.45 FTYPE

Purpose: Modify or display the command used to open a file of a type specified in the Windows registry.

Format: FTYPE [/P] [/R [filename] | filetype=[command]]

filename: One or more input files to read file type definitions from.

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](#)^[145], and [Executable Extensions](#)^[48].

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 "indirect" Windows file associations stored under the registry handle HKEY_CLASSES_ROOT, and discussed in more detail under [Windows File Associations](#)^[46]. 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 help file. 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 (Pause) Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail under

[Page and File Prompts](#)^[65].

/R (Read 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.

4.46 FUNCTION

Purpose: Create or display user-defined variable functions

Format: FUNCTION [/P] [/R [file...]] | name=[value]

file: One or more input files to read function definitions from.

name: The name of the function you want to display or set.

value: The value or definition of what the function should return.

/P(ause)

/R(ead from file)

See also: [UNFUNCTION](#)^[31].

Usage:

FUNCTION allows you to create or display user-defined variable functions that can be used everywhere [Variable Functions](#)^[357] can be used. User-defined functions can be used as powerful alternatives to [subroutines](#)^[22].

If you invoke the FUNCTION command with no parameters, it will display the current function list.

If you include a **name**, which may include wildcards (* or ?), with no equal sign and no **value**, FUNCTION will display the current values, if any, of all functions matching **name**.

If you include the equal sign and **value**, FUNCTION will create or update the function referred to by **name**.

A function can optionally use references to arguments numbered from %0 to %511 which will be replaced with the matching argument value when the function is called. %0 refers to the function name, %1 to the first argument, etc. For example, the function

```
function leftmost=`%@left[1,%1]`
```

will return the leftmost character in its argument, e.g. "%@leftmost[xyz]" will return "x".

The special variable reference "%#" expands to the number of arguments passed to the function.

A function value need not refer to any arguments at all. For example:

```
function tomorrow=`%@makedate[%@inc[%@date[%_date]]]`
```

could be simply invoked as "%@tomorrow[]".

The [Colors, Color Names and Codes](#)^[46] topic shows a simple example of use of a function in a batch file.

(TC) You can also define or modify user functions with the [Functions Window](#)^[74]. All of the information in this section also applies to functions defined via the dialog, unless otherwise noted.

Local and Global Functions

Functions can be stored in either a "local" or "global" list.

With a local function list, any changes made to the functions will only affect the current copy of the command processor. They will not be visible in other shells or other sessions.

With a global function list, all copies of the command processor will share the same function list, and any changes made to the functions in one copy will affect all other copies. This is the default in 4NT and Take Command. The use

You can control the type of function list from the [configuration dialogs](#)^[82], with the [LocalFunction](#)^[96] directive in the [.INI file](#)^[89], with the **/L** and **/LF** options of the [START](#)^[297] command, and with the **/L** and **/LF** [startup options](#)^[4].

There is no fixed rule for determining whether to use a local or global function 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, then modify it if you find a situation where the default is not convenient.

Whenever you start a second copy of the command processor which uses a local function list, it inherits a copy of the functions from the previous shell. However, any changes to the functions 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 function list in both shells.

Displaying Functions

If you want to see a list of all current FUNCTIONS commands, type:

```
[c:\] function
```

You can also view the definition of a single function. For example, if you want to see the definition of a MYFUNCTION function you created, you can type:

```
[c:\] function myfunction
```

Saving and Reloading Your Functions

You can save your functions to a file ("*FUNCTIONS.LST*") this way:

```
[c:\] function > function.lst
```

You can then reload all the function definitions in the file the next time you start up with the command:

```
[c:\] function /r function.lst
```

This is much faster than defining each function individually in a batch file. If you keep your function definitions in a separate file which you load when the command processor starts, you can edit them with a text editor, reload the edited file with FUNCTION /R, and know that the same function list will be loaded the next time you start the command processor.

When you define functions in a file that will be read with the FUNCTION /R command, you do not need back quotes around the value, even if back quotes would normally be required when defining the same function at the command line or in a batch file.

To remove a function, use the [UNFUNCTION](#)^[317] command.

Options:

- /P** (Pause): Wait for a key to be pressed after each screen page before continuing the display.
- /R** (Read 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

where *name* is the name of the function 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 FUNCTION /R command by placing the names on the command line, separated by spaces:

```
c:\> function /r func1.lst func2.lst
```

Each definition in a FUNCTION /R file can be up to 4095 characters long. The definitions can span multiple lines in the file if each line, except the last, is terminated with an [Escape Character](#)^[27]. If there is no filename argument and input is redirected, FUNCTION /R will read from STDIN.

4.47 GLOBAL

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 **command** first in the current directory and then it actually makes every subdirectory under the current directory the current working directory in turn, and there performs **command**. **Command** can be an internal command, an alias, an external command, or a batch file. When **command** is executed, it may be necessary to utilize one of the variable functions which convert a relative path to an absolute one, e.g., @truename[], @full[], etc to make sure that files of the same name in different directories are correctly handled.

This example copies the files in every directory on drive A to the directory C:\TEMP:

```
[a:\] global copy *.* c:\temp
```

If a specific filename is found in more than one directory on A:, assuming COPY is the default internal command, the one found last will be left in c:\temp. Which of identically named files is found last is UNPREDICTABLE!

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](#)^[27] 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)
```

Output Redirection

The default output redirection (i.e., `global ... > filename`) creates a new output file as each directory is visited. Unless `filename` includes an absolute filepath, it will be created relative to the currently visited directory. The simplest way to force a single target file is to enclose the whole command in parentheses, e.g.,:

```
(global command) > filename
```

Options:

- /H** (Hidden directories) Forces GLOBAL to look for hidden directories. If you don't use this switch, hidden directories are ignored.
- /I** (Ignore exit codes) If this option is not specified, GLOBAL will terminate if **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.

Without using this option, if GLOBAL is unable to change to a directory, e.g., the user does not have access rights, it will stop with an error message. With this option set, GLOBAL will ignore that directory, and all of its subdirectories, and continue in the next accessible directory.
- /P** (Prompt) Forces GLOBAL to prompt with each directory name before it performs the command. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[65].
- /Q** (Quiet) Do not display the directory names as each directory is processed.

4.48 GOSUB

Purpose: Execute a subroutine in the current batch file.

Format: GOSUB label [variables]

label: The batch file label at the beginning of the subroutine.

variables: Optional GOSUB variables.

See also: [CALL](#)^[157], [GOTO](#)^[222], and [RETURN](#)^[272].

Usage:

GOSUB can only be used in batch files.

The command processor allows 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](#)^[272] 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

```

GOSUB begins its search for the *label* on the line of the batch file immediately 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 still is not found, 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](#)^[272] call.

GOSUB calls with variables are limited to a maximum of 22 levels deep. (There is no numerical limit on normal GOSUB calls.)

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, cannot be modified with the [SET](#)^[281], [ESET](#)^[201], or [UNSET](#)^[312] commands, and cannot be used with the "DEFINED" test of [IF](#)^[227], [IFF](#)^[228], or [@IF](#)^[390].

You cannot use [SET](#)^[281] 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](#)^[272] from a GOSUB while inside a [DO](#)^[191] 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](#)^[272] command had been included as the last line of the file.

Subroutines can be nested.

Also see [user-defined functions](#)^[218].

4.49 GOTO

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](#)^[221], [CALL](#)^[157].

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 - quitting.
quit
:C_EXISTS
copy *.* a:\
```

GOTO begins its search for the *label* on the line of the batch file immediately after the GOTO command. If the label is not found between that position and the end of the file, GOTO will restart the search at the beginning of the file. If the label is still not found, 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](#)^[228] statements and [DO](#)^[191] / 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.

Option:

/I (IFF 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](#)^[138] alias) on a subsequent ENDDO, ELSE, ELSEIFF, or ENDIFF statement.

4.50 HEAD

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 [@file lists](#)^[49] for details).

/A: (Attribute select)

/P(ause)

/C (number of bytes)

/Q(uiet)

/I"*text*" (match description)

/V(erbose)

/N(umber of lines)

See also: [LIST](#)^[237], [TAIL](#)^[296], [TYPE](#)^[308].

File Selection

Supports [attribute switches](#)^[39], extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46].

Internet: Can be used with [FTP/HTTP Servers](#)^[52], e.g.

```
head "http://jpsoft.com/notfound.htm"
```

Usage:

The HEAD command displays the first part of a file or files. It is normally only useful for displaying ASCII text files (i.e. alphanumeric characters arranged in lines separated by CR/LF). Executable files (.COM and .EXE) and many data files may be unreadable when displayed with HEAD because they include non-alphanumeric characters or unusual line separators.

You can press **Ctrl-S** to pause HEAD's display and then any key to continue.

The following example displays 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 [Highlighting and Copying Text](#)^[76] for additional information on CLIP:.

FTP Usage

HEAD can also display files on [FTP/HTTP Servers](#)^[52]. The URL must be enclosed in quote marks. For example:

```
[c:\] head "ftp://jpsoft.com/index"
```

You can also use the [IFTP](#)^[229] command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want.

NTFS File Streams

HEAD supports file streams on NTFS drives under Windows 2000 / XP / 2003. 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.

Options:

- /A:** (Attribute select): Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[39] for information on the attributes which can follow **/A:**. Do not use **/A:** with @file lists. See [@file lists](#)^[49] for details.
- /C:** Display the specified number of bytes. **/C** accepts a b, k, or m modifiers at the end of the number. **b** is the number of 512-byte blocks, **k** is the number of kilobytes, and **m** the number of 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"[]"**. Do not use **/I** with @file lists. See [@file lists](#)^[49] for details.
- /N n** Display **n** lines. The default is 10.
- /P** (Pause): Pause and prompt after displaying each page.
- /Q** (Quiet): Do not display a header for each file. This is the default behavior, but an explicit **/Q** may be needed to override an alias that forces **/V**.
- /V** (Verbose): Display a header for each file.

4.51 HELP

Purpose: Display help for internal commands.

Format: HELP [topic]

topic: A help topic (internal command, variable or function).

See also: [The Online Help System](#)^[423].

Usage:

Online help is available for all JP Software command processors.

The **4NT** and **Take Command** help system (*jphelp.chm*) uses Microsoft's HTML Help Viewer (*HH.EXE*) included in all their current Windows bundles. If you are using both command processors under the same Windows configuration, you may find it more convenient to keep a single copy of *jphelp.chm* in your Windows HELP directory ("c:\windows\help") instead of several copies in product-specific directories.

If you type the command **HELP** by itself (or press **F1**^[119] when the command line is empty), an introductory page ("Overview") 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. All internal commands and most internal variables, functions and directives have their own topic.

You can also invoke help for the word immediately above (or immediately to the left of) the cursor by pressing **Ctrl-F1** (this can be useful when you need the syntax for a variable function).

4.52 HISTORY

Purpose: Display, add to, clear, or read the history list.

Format: HISTORY [/A *command* /F /N /P /R *filename*]

command: A command to be added to the history list.

filename: The name of a file containing entries to be added to the history list.

/A(dd)

/P(ause)

/F(ree)

/R(ead)

/N(o duplicates)

See also: [DIRHISTORY](#), [HistoryExclude](#) and [LOG](#).

Usage:

The command processor keeps a list of the commands you have entered on the command line. See [Command History and Recall](#) 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 8192 to 131071 characters (see the [History](#) directive). The default size is 8192 characters.

Your 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 full details see [local and global history](#).

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 with the Take Command [configuration dialogs](#) or the [HistMin](#) directive in the [.INI file](#). You can prevent any command line from being saved in the history by beginning it with an "at" sign ("@").

You can exclude specific commands from the History List with the [HistoryExclude](#) variable.

You can control whether duplicate entries will be saved in the history list with the [HistDups](#) directive

in the [.INI file](#)^[89].

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, and saves the history when the command processor exits.

Options:

- /A** (Add) Add a command to the history list. This performs the same function as the Ctrl-K key at the command line (see [Command History and Recall](#)^[13]).
- /F** (Free) Erase all entries in the command history list.
- /N** (No duplicates): Removes duplicate entries (oldest first) from the history list.
- /P** (Prompt) Wait for a key after displaying each page of the list. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[65].
- /R** (Read) 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 limit](#)^[25].

If you are creating a HISTORY /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](#)^[21]. However, you cannot use this method to exceed the command-line length limit.

4.53 IF

Purpose: Execute a command if a condition or set of conditions is true.

Format: `IF [/I] condition command`
`IF [/I] (condition (command1) ELSE (command2))`

condition: A [conditional expression](#)^[31]
command The command to execute if **condition** is TRUE.
command1 The command to execute if **condition** is TRUE.
command2 The command to execute if **condition** is FALSE.
[/I\(ignore case\)](#)^[22]

See also: [Conditional expressions](#)^[31], [IFF](#)^[228], [@IF](#)^[390].

Usage:

IF is most often used only aliases and batch files. It is always followed by a *condition* (see [Conditional expressions](#)^[31]), and then a *command*. First [condition](#)^[31] is evaluated, and if it is TRUE, *command* is executed. Otherwise, *command* is ignored.

ELSE

The **IF ... ELSE ...** syntax of **CMD.EXE** is also supported:

```
IF [/I] condition (command1) ELSE (command2)
```

The commands to be executed must be enclosed in parentheses (as in a [command group](#)^[27]). If *condition* is TRUE, *command1* is executed, if FALSE, *command2* is executed. **Note:** this syntax is much less powerful than the [IFF](#)^[228] command, which is recommended.

Option:

/I (Ignore case) This option is included only for compatibility with **CMD.EXE**. It normally has no effect, since all string comparisons are case-insensitive unless you specify a case-sensitive test (EQC).

4.54 IFF

Purpose: Perform conditional execution of sets of commands.

Format:

```
IFF condition1 THEN
    commandset1
[ELSEIFF condition2 THEN
    commandset2 ] ...
[ELSE
    commandset3 ]
ENDIFF
```

condition1,2,3: [Conditional expressions](#)^[31]

commandset1: One or more commands to execute if *condition1* is TRUE

commandset2: One or more commands to execute if *condition1* is FALSE, but *conditions2* is TRUE.

commandset3: One or more commands to execute if both *condition1* and *conditions2* are FALSE.

See also: [IF](#)^[227] and [@IF](#)^[390].

Usage:

IFF is similar to [IF](#)^[227], but it can perform one *commandset* when a [conditional expression](#)^[31] is true and a different *commandset* when it is false. Repeated use of the optional **ELSEIFF** clause permits **IFF** to sequentially evaluate multiple, independent [conditional expression](#)^[31]s, and execute the *commandset* associated with the first TRUE [conditional expression](#)^[31], or, if none are true, the *commandset* associated with the optional **ELSE** clause. After execution of any one of the *commandsets* the command after the **ENDIFF** clause will be executed.

You must **start a new line OR** include a [command separator](#)^[100]

- after each **THEN**

- **before** each **ELSEIFF**
- both **before** *and* **after** the **ELSE**.

The individual commands in each *commandset* may be *separate lines* of a batch file, or they may be separated by *command separator*^[100]s, in any combination. A *commandset* may also be empty. The individual commands in a *commandset* may include any internal command, alias, external command, or batch file.

IFF statements can be **nested**, i.e., a *commandset* may include another **IFF** / **ENDIFF** group. You must make sure that each individual command / *commandset* is syntactically correct. If an "inner" **IFF** / **ENDIFF** group is in error, it may not be detected until *after* the "outer" **ENDIFF** has been executed.

Notes

Be sure to read the cautionary notes about *GOTO*^[222] and **IFF** under the *GOTO*^[222] command before using a *GOTO*^[222] inside an **IFF** statement.

If you *pipe*^[60] data to an **IFF**, the data will be passed to the command(s) following the **IFF**, not to **IFF** itself.

Example

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 %1
is not a directory! %+ endiff`
```

4.55 IFTP

Purpose: Open or close an FTP/FTPS session

Format: IFTP [/S command] [/C /N /Q /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)
/N(o paths) **/V**(erbose)
/Q(uiet)

Usage:

Most file processing commands and functions in the command processor 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 or it is closed by the remote server. There are several advantages to using IFTP: the FTP connection remains open so commands normally 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 such as this:

```
[c:\] iftp "ftp://user:pwd@jpsoft.com/dir1"
```

For an FTPS connection, use something similar to:

```
[c:\] iftp "ftps://user:pwd@jpsoft.com/dir1"
```

This command tells IFTP to open an FTP/FTPS session with the server **jpsoft.com**, send **user** as the login username and **password** 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 asterisks).

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.

Note: If you enter IFTP with no arguments while a connection is active, the current server name and directory will be displayed.

If you enter IFTP with only the /Q or /V switch, you change the amount of information displayed without disturbing the existing connection (if any).

Once you have established an FTP session with IFTP, you can refer to files on the server by using "ftp:" (or "ftps:") 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 four 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.

Note: If an ftp file or path specification begins with a "~" (tilde), the command processor will not attempt to build a full directory name but will instead pass the entire string to the remote server.

You can only have **one** IFTP connection open at a time within a 4NT or Take Command session. However, while you have an IFTP connection 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/*"
```

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. The command processor 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 the command processor 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 the command processor expects, 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.

Options:

- /C** (Close): Use this switch, with no URL, to close an IFTP session (see the example above).
- /N** (no paths): Pass both source and target names to the server "as is" without any attempt at expanding the paths. This option should be used with caution and only for "non standard" servers (on older mainframes?) for which the default processing fails to build a suitable name.
- /Q** (Quiet): Turn off the display of the conversation with the FTP server.
- /S** (send): Allows you to send commands directly to an FTP server. The connection must have already been opened by a previous IFTP command.
- /V** (Verbose): Display the dialog with the FTP server while opening the connection. This can be useful for debugging connection problems.

See [FTP Servers](#)^[52] for additional information on formatting and usage of FTP and FTPS references.

4.56 INKEY

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 buttons)

See also: [INPUT](#)^[233].

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.

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](#)^[23b]. You can supply a list of valid keystrokes with the **/K** option.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Standard keystrokes 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 [ASCII, Key Codes, and ANSI X3.64 Commands](#)^[44b] for lists of ASCII and extended key codes.

To test for a non-printing ASCII keystroke returned by INKEY use the [@ASCII](#)^[36b] function 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. A batch file can handle Ctrl-C and Ctrl-Break itself with the [ON BREAK](#)^[25b] command.

Options:

- /C** (Clear 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** (Digits only) Prevents INKEY from accepting any keystroke except a digit from 0 to 9.
- /K"keys"** (Keystrokes). Specify the permissible keystrokes. The list of valid keystrokes should be enclosed in quote marks. 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), for example:

```
[c:\] inkey /k"ab[Ctrl-F9]" Enter A, B, Ctrl-F9 %%var
```

See [Keys and Key Names](#)^[46b] 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** (Mouse buttons) Returns @240 for the left button, @241 for the right button, and @242 for the middle button.
- /P** (Password) Prevents INKEY from echoing the character.

/W (Wait) Time-out period, in seconds, to wait for a response. If no keystroke is entered by the end of the time-out 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)? %%netmon
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.

4.57 INPUT

Purpose: Get a string from the keyboard and save it in 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)

/D(igits only)

/E(dit)

/L(ength)

/N(o colors)

/P(assword)

/W(ait)

/X (no carriage return)

See also: [SET](#)^[281], [INKEY](#)^[231], [KEYSTACK](#)^[236], [MSGBOX](#)^[251], and [QUERYBOX](#)^[265].

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](#)^[231]).

INPUT works within the command line window. If you prefer to use a dialog for user input, see the [MSGBOX](#)^[251] and [QUERYBOX](#)^[265] 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 redirected file or from the [KEYSTACK](#)^[236].

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

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. A batch file can handle Ctrl-C and Ctrl-Break itself with the [ON BREAK](#)^[252] command.

You can [pipe](#)^[64] text to INPUT, but it will set the variable in the "child" process used to handle the right hand side of the pipe. This variable will not be available in the original copy of the command processor used to start the pipe.

Options:

- /C** (Clear buffer) Discard any keystrokes pending in the keyboard buffer before INPUT begins accepting characters.
- /D** (Digits only) Prevents INPUT from accepting any keystrokes except digits from 0 to 9.
- /E** (Edit) 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 (you will still be able to edit the characters typed before the limit was reached).
- /N** (No color) Disables the use of input colors defined in the [InputColors](#)^[113] directive in the [INI file](#)^[89], and forces INPUT to use the default display colors.
- /P** (Password) Tells INPUT to echo asterisks, instead of the characters you type.
- /W** (Wait) Time-out period, in seconds, to wait for a response. If no keystroke is entered by the end of the time-out 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 time-out period, INPUT will wait indefinitely for the remainder of the line. You can specify **/W0** to return immediately if there are no keys waiting in the keyboard buffer.
- /X** (No carriage return) Prevents INPUT from adding a carriage return and line feed after the user's entry.

4.58 KEYBD

Purpose: Set the state of the keyboard toggles: Caps Lock, Num Lock, and Scroll Lock.

Format: KEYBD [/Cn /Nn /Sn]

n can be either 0 to turn off the toggle or 1 to turn on the toggle.

/C(aps lock)

/S(croll lock)

/N(um lock)

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.

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 (Caps lock) Turn the Caps Lock key on or off.

/N (Num lock) Turn the Num Lock key on or off.

/S (Scroll lock) Turn the Scroll Lock key on or off.

4.59 KEYS

Purpose: Enable, disable, or display the history list.

Format: KEYS [ON | OFF | LIST]

See also: [HISTORY](#)^[226].

Usage

This command is provided for compatibility with KEYS command in *CMD.EXE*, which controls the history list in Windows. The same functions are available by setting the [HistMin](#)^[105] directive in the [INI file](#)^[89], and by using the [HISTORY](#)^[226] command. (*CMD.EXE*'s KEYS command has no effect in Windows 2000 and above, because command line editing is always enabled. However, the command processor's KEYS command functions as described here under Windows NT.)

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, the command processor will show you the current status of the history list.

4.60 KEYSTACK

Purpose: Feed keystrokes to a program or command automatically.

Format: KEYSTACK [**!**] [**/Wx**] [**"abc"**] [**keyname**[**n**]] ...

!: Signal to clear the Keystack and the keyboard buffer.
/Wx: Delay in clock ticks.
"abc": Literal characters to be placed in the Keystack.
keyname: Name of a key whose code is to be placed in the Keystack.
[n]: Number of times to repeat the immediately preceding 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 will send the keystrokes to the currently active window. If you want to send keystrokes to another program (rather than have them function with the command processor itself), you must start the program or [ACTIVATE](#)^[136] its window so it can receive the keystrokes. You must do this before executing the KEYSTACK command.

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](#)^[291] 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](#)^[103] is set. 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. ***This includes the batch file debugger.***

Characters entered within quote marks (**"abc"**) will be sent to the target program "as is". The only items allowed outside quote marks are key names, the **!** and **/W** options, and a repeat count.

See [Keys and key names](#)^[464] 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 the command:

```
[d:\doc] start winword %+ keystack /w54 F10 Right 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), right arrow (go to the File menu), 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.

You can store a maximum of 8192 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.

Option:

/W (Wait) 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 quote marks. 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.

4.61 LIST

Purpose: Display a text file, with forward and backward paging and scrolling.

Format: LIST [/A:[-+]rhsad] [/B[-]n] [/H] [/I] [/I"text"] [/L[-]n] [/R] [/S] [/T"text"] [/W] [/X] [@file] [file...]

file: A file or list of files to display.

@file: A text file containing the names of the files to view, one per line (see [@file lists](#)^[49] for details).

/A: (Attribute select)	/R (everse)
/B (yte offset)	/S (tandard input)
/H (igh bit off)	/T (search for Text)
/I (gnore wildcards)	/W (rap)
/I"text" (match descriptions)	/X (heX display mode)
/L (ine offset)	

See also: [HEAD](#)^[22], [TAIL](#)^[29], [TYPE](#)^[30].

File Selection

Supports [attribute switches](#)^[39], extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46].

Internet: Can be used with [FTP/HTTP Servers](#)^[52].

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
```

Note: LIST is primarily intended for displaying the contents of ASCII text files (i.e. alphanumeric characters arranged in lines separated by CR/LF). It can be used for other files which contain non-alphabetic characters or unusual line separators, but you may need to use hex mode (see below) to display or search these files. Lines longer than 16,383 characters will be truncated unless you're in Wrap or Hex modes.

LIST displays files in the command processor window. In Take Command, 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 either with the mouse (on the tool bar and scrollbars) or from the keyboard. LIST recognizes the following keys and buttons:

<u>Key (Button)</u>	<u>Meaning</u>
Home	Display the first page of the file.
End	Display the last page of the file.
Esc (ListExit) ⁽¹²⁴⁾	Exit the current file.
Ctrl-C (Quit)	Quit LIST.
Ctrl-PgUp	Display previous file.
Ctrl-PgDn	Display next file.
Up Arrow	Scroll up one line.
Down Arrow	Scroll down one line.
Left Arrow	Scroll left 8 columns.
Right Arrow	Scroll right 8 columns.
Ctrl Left Arrow	Scroll left 40 columns.
Ctrl Right Arrow	Scroll right 40 columns.
Del	Prompt whether to delete the file.
Ins	Prompt whether to save the pipe or file to a new name.
Tab	Prompt for a new default tab size.
F1	Display online help.
B (ListPrevious) ⁽¹²⁵⁾	Go back to the previous file in the current group of files.
F (Find)	Prompt and search for a string or a sequence of hexadecimal values.
Ctrl-F	Prompt and search for a string, searching backward from the end of the file.
G (Goto)	Go to a specific line or, in hex mode, to a specific hexadecimal offset.
H (High)	Toggle the "strip high bit" (/H) option.
I (Info)	Display information on the current file (the full name, size, date, and time).
N (ListNext) ⁽¹²⁴⁾	Find next matching string.
Ctrl-N	Find previous matching string in the file.
O (ListOpen) ⁽¹²⁴⁾	Open a new file.
Ctrl-O	Open a new file.
P (Print)	Print selected pages or the entire file (make your selection in the Windows "Print" dialog).
U (ListUnicode) ⁽¹²⁵⁾	Toggle the Unicode display mode.
W (Wrap)	Toggle the "line wrap" (/W) option.
X (Hex)	Toggle the hex-mode display (/X) option.

Text searches performed with **F**, **N**, **Ctrl-F**, and **Ctrl-N**, or with the corresponding buttons, are not case-sensitive unless you check the "Match case" box in the search dialog. LIST remembers the search strings you have used in the current session; to select a previous string, use 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).

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.

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** ([ASCII](#)^[44b] values for the string "Ace"). Hexadecimal searches are case-sensitive, and search for exactly the string you enter.

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.

You can use [extended wildcards](#)^[36] 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". 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.

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](#)^[10b] is set to 0 in the [.INI file](#)^[89]. 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 16,383 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 output 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](#)^[64] automatically, but if it does not, use **/S** to explicitly specify piped input. 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.

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 [Redirection](#)^[67] for more information on CLIP:.

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. Regardless of the display mode, LIST in Take Command will bring up a standard Windows print dialog which allows you to print selected text, the current page, or the entire file. LIST in 4NT will ask you whether you wish to print the entire file or the current display page.

• FTP/HTTP Usage

LIST can display files on [FTP servers](#)^[52] as well as the contents of HTTP/HTTPS URLs. The URL must be enclosed in quote marks. For example:

```
[c:\] list "ftp://jpsoft.com/index"
[c:\] list "http://jpsoft.com/notfound.htm"
```

You can also use the [IFTP](#)^[229] 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/HTTP Servers](#)^[52] and [IFTP](#)^[229].

• NTFS File Streams

LIST supports file streams on NTFS drives under Windows 2000 and above. 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](#)^[444] 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.

If no filename is specified (and STDIN is not redirected), LIST will open the common Windows "open file" dialog.

Most of the LIST keystrokes can be reassigned with [key mapping](#)^[123] directives in the [.INI file](#)^[89].

You can set the colors used by LIST with the [ListColors](#)^[113] directive in the [.INI file](#)^[89], or the LIST Colors selection on the [Colors tab](#)^[83] of the [configuration dialogs](#)^[82]. If ListColors is not used, the LIST display will use the current default colors.

By default, LIST sets tab stops every 8 columns. You can change this behavior with the [TabStops](#)^[111] directive.

Options:

- /A:** (Attribute select) Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[39] for information on the attributes which can follow **/A:**. Do not use **/A:** with [@file lists](#)^[49] for details.
- /B[-]n** (Byte offset) Start at byte "n". If "n" is negative, start "n" bytes from the end of the file. The **/B** option will only display the file from the offset to the end; you cannot go back to a point before the offset.
- /H** (High 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 tool bar.
- /I** (Ignore wildcards) Only meaningful when used in conjunction with the **/T "text"** option. Directs 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" (Match descriptions) Select files by matching text in their descriptions. The text can include [wildcards](#)^[36] 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"[!]*"**. Do not use **/I** with @file lists. See [@file lists](#)^[49] for details.

/L[-]n (Line offset) Start at line "n". If "n" is negative, start "n" lines from the end of the file. The **/L** option only affects the initial page display; it does not prevent you from subsequently scrolling back to the start of the file.

/R (Reverse) 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 (Standard 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 (Text) 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](#)^[36]. 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 (Wrap) 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 tool bar.

/X (Hex mode) Display the file in hexadecimal (hex) mode. This option is useful when displaying executable files and other files that contain non-text characters. Each byte of the file 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 tool bar.

4.62 LOADBTM

Purpose: Switch a batch file to or from BTM mode.

Format: LOADBTM [ON | OFF]

Usage:

The command processor recognizes three kinds of [batch files](#)^[321]: *.CMD*, *.BAT*, and *.BTM*. Batch files executing in BTM mode run two to ten times faster than in CMD or BAT mode. Batch files automatically start in the mode indicated by their extension.

The LOADBTM command turns BTM mode on and off. It can be used to switch modes in either a *.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.

4.63 LOG

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(rrors)

/H(istory log)

/W(rite to)

See also: [HISTORY](#)^[226].

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 process ID in hexadecimal, along with the actual command after any alias or variable expansion. You can use the log file as a record of your daily activities. The [LogOn](#)^[106] directive can be used to enable command logging by default when the command processor starts.

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 [HistLogOn](#)^[105] directive can be used to enable history logging by default when the command processor starts.

By default, LOG writes to the file *TCLOG (TC) or 4NTLOG (4NT)* in the root directory of the boot drive. The default file name for LOG /H is *TCHLOG (TC) or 4NTHLOG (4NT)*. See the [LogName](#)^[106] and [HistLogName](#)^[105] directives to change the default names and/or locations of the log files.

Entering LOG or LOG /H with no parameters displays the log status (ON or OFF) and the name of the LOG file:

```
[c:\] log
LOG (C:\4NTLOG) 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 process ID.

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 be turned into a batch file that performs the same commands with little or no editing.

Options:

/E (Errors) This option saves all error messages to the log file. Also see the [LogErrors](#)^[108] directive.

/H (History 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 (Write) 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.

4.64 MD & MKDIR

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](#)^[267].

Internet: Can be used with FTP servers. See [FTP Servers](#)^[52].

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 [Directories and Subdirectories](#)^[440] for details.

When creating a directory on an LFN drive, you must quote any *path* which contains white space or special characters.

If MD creates one or more directories, they will be added automatically to the [extended directory search](#)^[56] database unless the **/N** option is specified.

You can create directories on FTP servers. The URL must be enclosed in quote marks. For example:

```
[c:\] md "ftp://ftp.abc.com/data/index"
```

Options:

- /N** (No update) Do not update the [extended directory search](#)^[56] database, *JPSTREE.IDX*. This is useful when creating a temporary directory which you do not want to appear in the extended search database.
- /S** (Subdirectories) 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
```

For compatibility with *CMD.EXE*, **/S** becomes the default if you enable command processor extensions with the **/X** switch on the command processor startup command line. See [Command Line Options](#)^[4] for details on **/X**.

4.65 MEMORY

Purpose: Display the amount and status of system RAM.

Format: MEMORY

Usage:

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.

4.66 MKLNK

Purpose: Create an NTFS hard or soft link.

Format: MKLNK [/D] arg1 arg2.

arg1: Name of an existing file (for hard link) or directory (for soft link).

arg2: Name of the new directory entry (a file or directory reference) to be created.

/D(delete a link)

Usage:

Due to Operating and File Systems restrictions, this command requires an NTFS 5 volume (Win2000, WinXP, etc.) and is not available under Win98 or WinME.

The file/directory names in **arg1** and **arg2** can be fully or partially qualified but may not contain wildcards.

• Hard Links

If the first argument ("**arg1**") is a file name, MKLNK will create a hard link.

NTFS 5 gives you the ability to create "hard links", which are multiple references to a single file. Creating a hard link causes the system to create an additional directory entry that points to the same file (unlike shortcuts, which actually create an additional file). Files can have multiple hard links, and therefore a single file can appear in multiple directories, with multiple names in each.

Once you create a hard link, you can access the file by any of its directory entries. Hard links are file-system level directory entries and each file on an NTFS volume has at least one hard link to itself. 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. MKLNK (and the underlying Windows API) cannot be used if the current directory is on a **subst** or **net use** drive, or a UNC volume.

• Soft Links

If the first ("**arg1**") argument is a directory, MKLNK will create a soft link (aka "*directory junction*" or "*reparse point*").

A soft link is an indirect or symbolic reference ("**arg2**") to a directory that physically resides in another location ("**arg1**"). Note: deleting files from a soft link is equivalent to deleting the files from the original directory.

Note: Other operating systems, such as Unix, may also support "hard links" and "soft links", but keep in mind that the Windows implementations of those concepts may not behave in the same manner even those the names might be similar.

Option:

/D Remove an existing hard link.

4.67 MOVE

Purpose: Move files to a new directory and drive.

Format: MOVE [/A:[-]rhsad] [/C] [/D] [/E] [/G] [/H] [/I"text"] [/J] [/L] [/M] [/N] [/O] [/P] [/Q] [/R] [/S] [/T] [/U] [/V] [/W] [/Y] [/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 [@file lists](#)^[49] for details).

/A: (Attribute select)	/O (don't move if target exists)
/C (hanged)	/P (rompt)
/D (irectory)	/Q (uiet)
/E (No error messages)	/R (eplace)
/G (display percent copied)	/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)	/Y (force move of encrypted files)
/N (othing)	/Z (overwrite)

Note: MOVE is a complex command. When source and destination are on the same volume and the destination doesn't exist, it's equivalent to a simple [REN](#)^[27b], but when the destination exists or two volumes are involved, it becomes a two-step command: a [COPY](#)^[16b] to the target followed, if successful, by a [DEL](#)^[17b] of the source. In this topic, references to "move" may apply to the entire process or only to one of the above steps specifically depending on context.

See also [COPY](#)^[16b], [DEL](#)^[17b] and [RENAME](#)^[27b].

File Selection

Supports [attribute switches](#)^[3b], extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46]. Date, time, size, or file exclusion ranges anywhere on the line apply to all source files. Use wildcards with caution on LFN volumes; see [LFN File Searches](#)^[47] for details.

Internet: Can be used with [FTP/TFTP/HTTP/HTTPS Servers](#)^[52].

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:\FINANCEMYFILES 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 [Extended Directory Searches](#)^[56] for details.

Be careful when you use MOVE with the SELECT^[27] 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).

- **FTP Usage:**

If you have appropriate permissions, you can move files to and from Internet URLs (FTP, TFTP and HTTP). The URL must be enclosed in quote marks. For example:

```
[c:\] move "ftp://ftp.abc.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 such as **[*]** and **[?]** will be treated as wildcards in FTP URLs, but will be treated as normal characters in HTTP URLs.

Note: The **/G** option (percent moved) may report erratic values during transfer of files larger than 4 Gb (an ftp limitation) and during http downloads.

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/HTTP Servers](#)^[52] and [IFTP](#)^[229].

- **NTFS File Streams:**

MOVE supports file streams on NTFS drives under Windows 2000 and above. 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. However, 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](#)^[444] 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.)

Note: The wildcard expansion process will attempt to allow both CMD-style "*extension*" matching (only one extension, at the end of the word) and the advanced 4NT/TC *string* matching (allowing things like *.*.abc) when an asterisk is encountered in the **destination** of a MOVE command.

Options:

- /A:** (Attribute select) Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[39] 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. Do not use **/A:** with @file lists. See [@file lists](#)^[49] for details.
- /C** (Changed 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. Do not use **/C** with @file lists. See [@file lists](#)^[49] for details.
- /D** (Directory) 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.
- /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** (Hidden) 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"** (Match descriptions) Select **source** files by matching text in their descriptions. The text can include [wildcards](#)^[36] 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"[]"**. Do not use **/I** with @file lists. See [@file lists](#)^[49] for details.
- /J** Copy the file in restartable mode. The copy progress is tracked in the destination file in case the move fails. The copy can be restarted by specifying the same source and destination file names.
- /L** Perform FTP transfers in ASCII mode, instead of the default binary mode.
- /M** (Modified) Move only files that have the archive bit set. The archive bit will remain set after the MOVE; to clear it use [ATTRIB](#)^[146]. Do not use **/M** with @file lists. See [@file lists](#)^[49] for details.
- /N** (Nothing) Do everything except actually move the file(s). This option is most useful for testing what a complex MOVE command will do. **/N** displays how many files would be moved. **/N** does **not** prevent creation of **destination** subdirectories when it is used with **/S**.

- /O** Don't move the file(s) unless the target doesn't exist, i.e. do not overwrite an existing target..
- /P** Prompt) Prompt the user to confirm each move. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[63]. Note: the [CopyPrompt](#)^[10] directive can be used to force prompting at the command line only.
- /Q** (Quiet) Don't display filenames, the total number of files moved, the percentage moved, or the amount of disk space recovered, if any. When used in combination with the **/P** option above, it will prompt for filenames but will not display the totals. This option is most often used in batch files. See also **/T**.
- /R** (Replace) Prompt for a **Y** or **N** response before overwriting an existing *destination* file. Also see the [CopyPrompt](#)^[10] directive.
- /S** (Subdirectories) 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](#)^[56] database. 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 [@file lists](#)^[49] for details.
- /T** (Total) 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** (Update) 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. Do not use **/U** with @file lists. See [@file lists](#)^[49] for details. When used with file systems that have different time resolutions (such as FAT and NTFS), **/U** will attempt to use the "coarsest" resolution of the two.
- /V** (Verify) Verify each disk write by performing a true byte-by-byte comparison between the source and the newly-created target file. This option may significantly increase the time necessary to complete a MOVE command.
- /W** (Wipe): If the MOVE is to a different drive, after the move overwrite the source file with 0's before deleting it (like DEL **/W**).
- /Y** (XP+ Only) Force copy of an encrypted file even when the target will be decrypted (for CMD.EXE compatibility).
- /Z** (Overwrite) 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.

To emulate an approach used by some implementations of *CMD.EXE*, see the [COPYCMD](#)^[34] topic.

4.68 MSGBOX

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 (st button)

/Q(uestion)

/2 (nd button)

/S(top)

/3 (rd button)

/T(imeout)

/I(con)

/W(arning)

See also: [INKEY](#)^[231], [INPUT](#)^[233], and [QUERYBOX](#)^[265].

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. The command processor will 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 internal variable [%_?](#)^[346]. 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:

Yes or OK	10
No	11
Cancel	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 in its documentation (see [?_?](#)). 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 the [INKEY](#)^[231] and [INPUT](#)^[233] commands.

Options:

/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.

/Tn: (Time out) 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.

/W(arning): Display an exclamation point icon in the message box.

4.69 ON

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, operating system error (such as a disk write error) or internal command error (such as a COPY command that fails to copy any files, or the use of an invalid command 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 together, 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 batch file, the first batch 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](#)^[222] or [GOSUB](#)^[221] 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 and 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](#)^[27] 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 next command in the batch file (the command after the one that was interrupted by the break or error). To avoid continuing the batch file after a break or error is for the *command* to transfer control with [GOTO](#)^[222], end the batch file with [QUIT](#)^[266] or [CANCEL](#)^[158], or start another batch file (without CALLing it).

When handling an error condition with ON ERROR, you may find it useful to use [internal variables](#)^[342], particularly [% ?](#)^[346] and [% SYSERR](#)^[356], 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.

Caution: 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.

4.70 OPTION

Purpose: Modify the command processor configuration.

Format: OPTION [/optname=value ...]
OPTION @filename
OPTION optname

optname: Name of an INI file directive to set, modify, or display.

value: A new value for that directive.

filename: A file containing initialization directives to be immediately activated.

See also: [.INI file](#)^[89], [SETDOS](#)^[283]

Usage:

OPTION displays a property sheet which allows you to modify most of the configuration options stored in the [.INI file](#)^[89].

When you exit from the property sheet, you can select **Save** to save your changes in the *.INI* file for use in the current session and all future sessions or select **Cancel** to discard the changes. See

[Configuration Dialogs](#)^[82] for more information.

In most cases, changes you make in the **Startup** section of the OPTION dialogs will only take effect when you restart your command processor. 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](#)^[114] and [Advanced Directives](#)^[125] cannot be modified with the OPTION dialogs. These settings must be inserted or modified in the *.INI* file manually.

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 [.INI file](#)^[89] for details).

• 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:

```
[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 most useful for testing settings quickly, and in aliases or batch files which depend on certain options being in effect.

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".

• Using the "OPTION @filename" syntax

The OPTION command allows you to set/modify many directives with a single command on the fly. The file must be in the same format as an [.INI file](#)^[89].

• Displaying an option value

Specifying an option name alone will display the value of that option; e.g.:

```
C:\>option LocalHistory  
localHistory=Yes
```

Also see the [@OPTION](#)^[401] function.

4.71 PATH

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](#)^[207] and [SET](#)^[287] (the **PATH** command is syntactically equivalent to **SET PATH**).

Usage:

When the command processor is asked to execute an external command (a **.COM**, **.EXE**, **.BTM**, **.BAT**, or **.CMD** file, or an executable extension), 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.

(TC) Under Windows 98 and ME, the command processor 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 / 2003 the order is reversed, and the command processor 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. The command processor 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.

(TC) For example, after the following **PATH** command, Take Command will search for an executable file in six directories: the current directory, the two Windows directories, the root directory on drive C, then the **BIN** subdirectory on C, and then the **UTIL** subdirectory on C:

(4NT) For example, after the following **PATH** command, 4NT will search for an executable file in four directories: the current directory, 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
```

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](#)^[347] environment variable, and the related [PathExt](#)^[108] **.INI** directive, can be used to select the extensions to look for when searching the **PATH** for an executable file.

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](#)^[287] or [ESET](#)^[207] 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.

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 4,090 characters (the internal buffer limit, with room for "PATH "). It is usually more efficient to use aliases to load application programs than to create a long PATH. See [ALIAS](#)^[138] 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.

4.72 PAUSE

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:

```
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. In a batch file, you can handle **Ctrl-C** and **Ctrl-Break** yourself with the [ON BREAK](#)^[252] command.

4.73 PDIR

Purpose: Display information about files and subdirectories in user-definable fields. It is a "programmable DIR" command.

Format: PDIR [*ranges*] [/I"*text*"] [/A:[[:]][-]*rhsda*] [/H] [/K] [/M] [/O:[[:]][-]*adeginrsu*] [/P] [/S] [/T[:*a|c|w*]] [/(...)] [*file...*]

<u>/A</u> ^[256]	Attribute select	<u>/O</u> ^[256]	Order by subkeys
<u>/H</u> ^[256]	do not Hide . and ..	<u>/P</u> ^[256]	Pause after each screen page
<u>/I</u> ^[256] "tex t"	Include files that match description	<u>/S</u> ^[256]	include Subdirectories
<u>/K</u> ^[256]	show header	<u>/T</u> ^[256]	select by Type of filetime
<u>/M</u> ^[256]	show footer	<u>/(...)</u> ^[256]	include only the specified fields in the output

See also: DIR^[179], ATTRIB^[146], DESCRIBE^[176], SELECT^[275].

File Selection

Supports attribute switches^[39], extended wildcards^[36], ranges^[40], multiple file names^[45], and include lists^[46].

Internet: Can be used with FTP/HTTP Servers^[52].

Usage:

PDIR is an extremely flexible command allowing you to display information about files and directories from one or more local or remote volume or directories in a wide array of user-defined formats. For a simpler version, see the DIR^[179] command.

PDIR and DIR^[179] are related, but they are not the same command. They do not have identical switches and they are not intended to produce identical output. **PDIR** is primarily intended to produce output that will be subsequently parsed by another program (or batch file), or (more rarely) for a special-purpose directory display. Its options and output are geared towards those applications.

The various **PDIR** displays are controlled through options or switches. The best way to learn how to use the many options available with the **PDIR** command is to experiment. You will soon know which options you want to use regularly. You can then select those options permanently by using the ALIAS^[138] command.

Example:

To display the CRC, the full LFN and the owner of each file:

```
pdire /r ffn q) *
```

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.

Most options are used to **select** the desired files/directories. This is in contrast to the DIR^[179] command. The special option /(...)^[256] is used to determine which characteristics of the selected files or directories should be displayed in which sequence and format.

/A[:] (Attribute select) Display only those files that have the specified attribute(s) set. See Attribute Switches^[39] for information on the attributes which can follow **/A: .**

- /H** Show the "." and ".." directory names (normally suppressed).
- /I"text"** (Match descriptions) Select filenames by matching text in their descriptions. The text can include [wildcards](#)^[36] 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.
- /K** Show the header (disk and directory name) display.
- /M** Show the footer (file and byte count totals) display.
- /O** (Order) 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:
- n** Sort by filename (this is the default)
 - 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 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. For information on supported compression systems see **/C** above.
 - d** Sort by date and time (oldest first); also see **/T:acw**
 - e** Sort by extension
 - g** Group subdirectories first, then files
 - i** Sort by file description (ignored if **/C** or **/O:c** is also used).
 - r** Reverse the sort order for all options
 - s** Sort by size
 - u** Unsorted
- /P** (Pause) Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[63].
- /S** (Subdirectories) Display file information from the current directory and all of its accessible subdirectories.
- /T:type** (Time display) Specifies which single one of the date and time fields below, available on a drive which supports long filenames, should be displayed and used for sorting:
- a** Last access date and time (access time is only saved on NTFS volumes).
 - c** Creation date and time.
 - w** Last write date and time (default).
- DEFAULT:** If **/T** is not specified, **/T:w**
- Note:** If more than one time type is specified, the first one specified is used, and all subsequent ones ignored.
- /(...)** (Fields and Formats) Use this option to define the various fields and display formats you wish to use for each selected entry. The fields may be in any order, and may be

repeated. If this option is not used, the output format is identical to that of the [DIR](#)^[179] command.

- a** Attributes
- c** Compression: Display the compression ratio on compressed drives. The ratio is left blank for directories and files with a length of 0 bytes and for files on non-compressed drives. 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 8KB). 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.
- d[...]** Date.
 - d** day
 - m** month
 - y** year
- f[...]** File or Directory name (case sensitive)
 - P** SFN path
 - p** LFN path
 - N** SFN filename
 - n** LFN filename
- i** Description
- m** MD5 hash value (see the [@MD5](#)^[40b] function)
- q** File or directory owner (NTFS only)
- r** CRC32 hash value (see the [@CRC32](#)^[36b] function)
- s[p]** stream names (NTFS only). If "**sp**" is specified, prefix the pathname to the filename+stream.
- t[...]** Time
 - d** milliseconds
 - s** seconds
 - m** minutes
 - h** hours
- z[...]** Size.
 - a** allocated size (this will usually be more than the physical size unless the file is compressed.)
 - c** the size will be formatted using the thousands separator (default is a comma)
 - k, K, m, M, g, G, t, T** (case sensitive) format as kilobytes, megabytes, gigabytes, or terabytes, as used in variable functions. (see [Memory Size / Disk Space / File Size Units and Report Format](#)^[357]). Note that the size will be truncated, not rounded.

@function[*] call the specified variable [function](#)^[357] (internal or user-defined). To specify the current filename, use ***** as the argument. For example, "**pdir / (f @md5 [*])**" displays the filename and the MD5 hash. Note that the %

prefix of the function name is NOT used with the symbolic * argument. If the argument of the function is NOT the symbolic * or it is an "inner" function the % prefix must be **doubled**, e.g., `@function1[%@function2[*]]`

"..." Literal string (in quotes). Characters are displayed "as-is". Note that the string can include [escaped](#) characters. For example, assuming `EscapeChar=^` (the default), use `"^r^n"` (with quotes) to insert a CR/LF sequence.

Note: You can also specify the formatting for a field by prefacing the field character with:

`[-]width.precision`

where the "-" will left-justify the field (default is right-justify); **width** specifies the minimum width, and **precision** specifies the maximum width of the field.

4.74 PLAYAVI

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.

Note: This command relies on the capabilities of your Windows configurations, including access to the proper codec. See your Windows documentation for details.

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. Under Take Command, you can change this default behavior with the **/A** switch, described below.

Options:

/A (Asynchronous): **(TC)** 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 (Center): Displays the AVI viewer in the middle of the screen. Without this option, the viewer appears in the upper-left corner of the screen.

/S (Synchronous): **(TC)** 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.

4.75 PLAYSOUND

Purpose: Play MP3, .WAV, Midi, and other sound files.

Format: PLAYSOUND [/A /S] *filename*

filename: The file to play.

/A(synchronous)

/S(ynchronous)

Usage:

PLAYSOUND "plays" MP3, .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 an unknown file extension.

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.

Options:

/A (Asynchronous): 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 (Synchronous): 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.

4.76 POPD

Purpose: Return to the disk drive and directory at the top of the directory stack..

Format: POPD [*]

See also: [DIRS](#)^[190], [PUSHD](#)^[264], [@DIRSTACK](#)^[371] and [Directory Navigation](#)^[54].

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](#)^[23]. Read the section on [Directory Navigation](#)^[54] for complete details on this and other 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.

4.77 PRINT

Purpose: Print the specified file(s) using the application associated with each file's extension.

Format: PRINT filename ...

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 most 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 [Windows File Associations](#)^[461] settings in the 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.

4.78 PROMPT

Purpose: Change the command-line prompt.

Format: PROMPT [text]

text: Text to be used as the new command-line prompt.

See also: [ESET](#)^[201] and [SET](#)^[281] (the **PROMPT** command is syntactically equivalent to **SET 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. Unless otherwise specified, those meta characters are case-independent.

- a** The ampersand character[&].
- b** The vertical bar character [|].
- c** The open parenthesis [(].
- d** Current date, in the format: *Thu 12-12-04* (the month, day, and year are formatted according to your current country settings).
- D** Current date, in the format: *Thu Dec 12, 2004*.
- e** The ASCII **ESC** character (decimal 27), very useful for [ANSI](#)^[64] commands.
- f** The close parenthesis [)].
- g** The > character.
- h** Backspace over the previous character.
- j** Current date in ISO 9601 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.
- 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.)
- Xd:** Current directory on drive **d**: in upper case, including the drive letter.
- z** Current shell nesting level. The first copy of the command processor is shell level 0.
- +** Display one + character for each directory on the PUSHHD 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
Thu Sep 30, 2004 10:29:19 >
```

The command processor prompt can be set in [4START \(4NT\) or TCSTART \(TC\)](#)^[6] or in any batch file that runs when the command processor starts.

If you enter PROMPT with no arguments, the prompt will be reset to its default value.

You can include literal text and special characters as well as the value of any [environment](#)^[336] variable,

[internal variable](#)^[342], or [variable function](#)^[357] in a prompt. For example, if you want to include the size of the largest free memory block 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 at 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](#)^[376] variable function 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`
```

You can include [ANSI](#)^[64] escape sequences in the PROMPT by using the built-in ANSI X3.64 support in **4NT** / **TC**. This example uses ANSI X3.64 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
```

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](#)^[6] (**4NT**) or [TCSTART](#)^[6] (**TC**) file and use IF, IFF, or SWITCH statements to set the appropriate prompt for different shells.

4.79 PUSHD

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](#)^[190], [POPD](#)^[261], [@DIRSTACK](#)^[371] and [Directory Navigation](#)^[54].

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:\WORDPWEMOS`, 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.

`PUSHD` can also change to a network drive and directory specified with a UNC name (see [File Systems](#)^[439] for details).

If `PUSHD` cannot change to the directory you have specified it will attempt to search the [CDPATH](#)^[59] and the [extended directory search](#)^[56] database. You can also use [wildcards](#)^[36] in the **path** to force an extended directory search. Read the section on [Directory Navigation](#)^[54] 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](#)^[23].

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.

4.80 QUERYBOX

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 /P /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) **/P**(assword)

/E(dit existing value) **/T**(imeout)

/L (maximum Length)

See also: [INKEY](#)^[234], [INPUT](#)^[233], and [MSGBOX](#)^[254].

Usage:

`QUERYBOX` displays a dialog box with a prompt, an optional title, and a string 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.

`QUERYBOX` is similar to `INPUT`, except it appears as a popup dialog box. If you prefer to work within the command line window, see the `INKEY` and `INPUT` commands.

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 a string and store it in the variable NAME:

```
querybox "File Name" Enter a name: %%name
```

If you press **Ctrl-C** or **Ctrl-Break** while QUERYBOX 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. A batch file can handle **Ctrl-C** and **Ctrl-Break** itself with [ON BREAK](#) ^[252].

QUERYBOX returns a value of zero in the internal variable [%_?](#) ^[346] after a successful operation, and a non-zero value otherwise (timeout, cancel, etc.). Be sure to save the return value in another variable or test it immediately; because the value of [%_?](#) changes with every internal command.

Options:

- /D** (Digits): Only accepts numeric values.
- /E** (Edit): 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".
- /P** (Password) Tells QUERYBOX to echo asterisks, instead of the characters you type.
- /Tn** Wait for a maximum of "n" seconds for a response.

4.81 QUIT

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](#) ^[158] and [EXIT](#) ^[205].

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](#) ^[227] command, and the [%?](#) ^[346] variable. Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

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.

4.82 RD

Purpose: Remove one or more subdirectories.

Format: RD [/I"text" /K /R /Q /S] [@file] path...
or
RMDIR [/I"text" /K /R /Q /S] [@file] path...

path: The name of one or more subdirectories to remove.

@file: A text file containing the names of the directories to remove, one per line (see [@file lists](#)^[49] for details).

/I (match descriptions)

/R(ecycle bin)

/K (no Recycle Bin)

/S(ubdirectories)

/Q(uiet)

See also: [MD](#)^[243].

File Selection

Supports extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46]. Use wildcards with caution on LFN volumes; see [LFN File Searches](#)^[47] for details.

Internet: Can be used with [FTP Servers](#)^[52].

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](#)^[172] 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.

If RD deletes one or more directories, they will be deleted from the [extended directory search](#)^[56] database.

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.

RD will delete hidden directories, for compatibility with CMD.EXE.

You can remove directories on [FTP servers](#)^[52]. The URL must be enclosed in quote marks. For example:

```
[c:\] rd "ftp://ftp.abc.com/data"
```

You can also use the [IFTP](#)^[229] command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want

Options:

- /I"text"** (Match descriptions) Select directories by matching text in their descriptions. The text can include [wildcards](#)^[36] 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"[]"**. Do not use **/I** with @file lists. See [@file lists](#)^[49] for details.
- /K** (No Recycle Bin) When used with the **/S** option, this will physically delete files instead of sending them to the Windows Recycle Bin. This option overrides the default [RecycleBin](#)^[109] **/INI** option.
- /Q** (Quiet) When used with the **/S** option, this will suppress the prompt before deleting the directories.
- /R** (Recycle Bin) When used with the **/S** option, this will send the deleted files to the Windows Recycle Bin. This option overrides the default [RecycleBin](#)^[109] **/INI** option.
- /S** (Subdirectories) This option is included only for compatibility with *CMD.EXE*, and should be used with **EXTREME CAUTION!** It deletes all files (including hidden and system files) in the named directory and all of its subdirectories, then removes all subdirectories. **It can potentially erase all files on a drive with a single command.**

Note: Do **not** use **/S** with @file lists; see [@File Lists](#)^[49] for details.

4.83 REBOOT

Purpose: Reboot the computer, log off Windows, or shut down.

Format: REBOOT [/L] [/P] [/S] [/V]

/L(ogoff)

/S(hutdown)

/P(ower off)

/V(erify)

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.

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
```

The command processor issues the standard commands to shut down other applications and the operating system before rebooting. Windows may prompt you for additional actions or even ignore the request altogether depending on which processes are running.

Options:

- /L** (Logoff) 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 may not work properly in Windows 98 and ME, due to the way Windows responds to "logoff" requests.

- /P** (Poweroff) Log off Windows and turn off the computer.
- /S** (Shutdown) Shut down the system, but do not reboot. This is equivalent to selecting Shutdown from the Start menu, then selecting "Shut down the computer" in the shutdown dialog.
- /V** (Verify) Prompt for confirmation (**Y** or **N**) before acting.

4.84 RECYCLE

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** (Delete): Empty the recycle bin for the specified drive(s).
- /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** (Prompt): Prompt the user to confirm each delete operation.
- /Q** (Quiet): Don't display the name of the recycle bin(s). This option is most often used in batch files.

Note: This feature requires some APIs which were not formally introduced by Microsoft until Windows 2000.

4.85 REM

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 2,043 characters long. 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 and *CMD.EXE*. A simpler method for creating a zero-byte file with the command processor is to use **>filename** as a command, with no actual command before the [>] redirection character.)

4.86 REN / RENAME

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 [@file lists](#)^[49] for details).

/A: (Attribute select)	/P (rompt)
/E (No error messages)	/Q (uiet)
/I"text" (match description)	/S (ubdirectory)
/N (othing)	/T (otal)

See also: [COPY](#)^[164] and [MOVE](#)^[246].

File Selection:

Supports [attribute switches](#)^[39], extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46]. Use wildcards with caution on LFN volumes; see [LFN File Searches](#)^[47] for details.

Internet: Can be used with [FTP/HTTP Servers](#)^[52] and HTTP/HTTPS 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.

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*:

```
[c:\] ren config.nt autoexec.nt tcstart.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](#)^[56] database will be automatically updated to reflect the change.

You can also rename a subdirectory to a new location in the directory tree on the same physical drive (sometimes called "prune and graft"). 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**.

If you have appropriate permissions, you can rename files on FTP, HTTP, and HTTPS servers. The URL must be enclosed in quote marks. For example:

```
[c:\] ren "ftp://ftp.abc.com/file1.txt" file2.txt
```


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/HTTP Servers](#)^[52] and [IFTP](#)^[229].

Note: The wildcard expansion process will attempt to allow both CMD-style "*extension*" matching (assumes only one extension, at the end of the word) and the advanced 4NT/TC *string* matching (allowing things like `*.*.abc`) when an asterisk is encountered in the **destination** of a REN command.

Options:

- /A:** (Attribute select) Rename only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[39] for information on the attributes which can follow **/A:**. Do not use **/A:** with `@file lists`. See [@file lists](#)^[49] for details.
- /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"** (Match descriptions) Select files by matching text in their descriptions. The text can include [wildcards](#)^[36] 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"[]"**. Do not use **/I** with `@file lists`. See [@file lists](#)^[49] for details.
- /N** (Nothing) Do everything except actually rename the file(s). **/N** displays how many files would be renamed. This option is useful for testing what a REN command will actually do.
- /P** (Prompt) Prompt the user to confirm each rename operation. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[65].
- /Q** (Quiet) Don't display filenames or the number of files renamed. When used in combination with the **/P** option above, it will prompt for filenames but will not display the totals. This option is most often used in batch files. See also **/T**.
- /S** (Subdirectory) 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](#)^[220] REN.
- /T** (Total) Don't display filenames as they are renamed, but report the number of files renamed. See also **/Q**.

4.87 RETURN

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](#)^[227].

Usage:

The command processor allows 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](#)^[197] loop.

The following batch file fragment calls a subroutine which displays the files in the current directory:

```
echo Calling a subroutine
gosub subrl
echo Returned from the subroutine
quit
:subrl
dir /a/w
return
```

If you specify a **value**, RETURN will set the internal exit code to that value. That exit code should be tested immediately upon return from the subroutine and before it is reset by another command. For information on exit codes from internal commands, see the [?](#)^[346] variable.

4.88 SCREEN

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 and ECHOERR](#)^[198], [ECHOS and ECHOSERR](#)^[199], [SCRPUT](#)^[274], [TEXT](#)^[301], and [VSCRPUT](#)^[314].

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](#)^[274] or [VSCRPUT](#)^[314]. 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.

(**TC**) 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 [Resizing the Take Command Window](#)^[77] 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!
```

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 command processor window:

```
screen 999 999 Hello, World
```

4.89 SCRPUT

Purpose: Position text on the screen and display it in color.

Format: SCRPUT *row col [BRlght] fg ON [BRlght] bg text*

row: Starting row
col: Starting column
fg: Foreground character color
bg: Background character color
text: The text to display

See also: [ECHO and ECHOERR](#)^[198], [ECHOS and ECHOSERR](#)^[199], [SCREEN](#)^[273], [TEXT](#)^[301], and [VSCRPUT](#)^[314].

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 [Colors and Color Names](#)^[461] for details.

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. (**TC**) The maximum **row** value is determined by the current height of the Take Command window. The maximum **column** value is determined by the current virtual screen width (see [Resizing the Take Command Window](#)^[77] for more information).

(**TC**) SCRPUT checks for a valid **row** and **column**, and displays a "Usage" error message if either value is out of range.

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 does not move the cursor when it displays the **text**.

The following batch file fragment displays part of 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
```

4.90 SELECT

Purpose: Interactively select files for a command.

Format: SELECT [/1 /A[:][-][+]*rhsad*] /C /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.

/1 One selection only	/J (ustify names)
/A (ttribute select)	/L (ower case)
/C (ompression)	/O (rder)
/D (isable color coding)	/T (ime)
/E (use upper case)	/X (display short names)
/H (ide dots)	

File Selection

Supports extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46]. Ranges **must** appear immediately after the SELECT keyword.

Internet: Can be used with FTP servers. See [Using FTP/HTTP Servers](#)^[52].

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.

SELECT can also select files on FTP servers. The URL must be enclosed in quote marks. 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/HTTP Servers](#)^[52] and [IFTP](#)^[229].

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 through the file list. You can also use character matching to find specific files, just as you can in any [popup window](#)^[464]. 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:

space	Select a file, or unselect a marked file.
+	Select a file (all products), or unselect a marked file (4NT) .
-	Unselect a marked file.
*	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.
/	Unselect all files.
Ctrl-L	(4NT) View the current highlighted file with LIST ^[237] . When you exit from LIST, the SELECT screen will be restored.
Enter	Execute the command with the marked files, or with the currently highlighted file if no files have been marked.
Esc	Skip the files in the current display and go on to the next file specification inside the parentheses or brackets (if any).
Ctrl-C or Ctrl-Break	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.

(4NT) You can set the default colors used by SELECT on the [Colors tab](#)^[83] of the configuration dialogs, or with the [SelectColors](#)^[114] and [SelectStatBarColors](#)^[114] directives in the [.INI file](#)^[89]. 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](#)^[46] inside the parentheses by inserting a semicolon:

```
[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](#)^[25]. The current line length is displayed by SELECT while you are marking files to help you to stay within that limit.

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. 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](#)^[40]. 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:

```
[c:\] alias utils select (c:\util\*.com;*.exe;*.btm;*.bat)
```

If you want to use [filename completion](#) ^[17] 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 wild cards 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.

Options:

- /I** Only allow one selection.
- /A[:]** (Attribute select) Select only those files that have the specified attribute(s) set. See [Attribute Switches](#) ^[39] for information on the attributes which can follow **/A:**.
- /C** (Compression) 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 short filename format. Only compressed NTFS drives are supported. See [DIR /C](#) ^[179] for more details on how compression ratios are calculated.
- /D** (Disable color coding) **(4NT)** Temporarily turn off directory color coding.
- /E** Display filenames in upper case.
- /H** (Hide dots) Suppress the display of the "." and ".." directories.
- /I"text"** (Match descriptions) Display filenames by matching text in their descriptions. The text can include [wildcards](#) ^[36] 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** (Justify names) Justify (align) filename extensions and display them in the FAT format.
- /L** (Lower case): Display file and directory names in lower case.
- /O** (Order) Set the sort order for the files. The order can be any combination of the following options:
 - n** Sort by filename (this is the default)
 - 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 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).
- 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:

- a** Last access date and time (access time is not saved on VFAT and FAT32 volumes).
- c** Creation date and time.
- w** Last write date and time (default).

/X Display short filenames in 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.

4.91 SENDMAIL

Purpose: Send an email message.

Format: SENDMAIL [/A file] [/D] [/In] [/Pn] /V] "address [bcc:address]" subject [text | @msgfile]

file: The attachment file.

address:^[279] The destination email address.

subject^[279]: The subject line.

text^[279]: The message to send.

msgfile^[279] The file containing the message body.

/A(attachment)^[279]

/P(riority)^[279]

/D(elivery Confirmation)^[279]

/V(erbose)^[279]

/I(mportance)^[279]

See also: [SNPP](#)^[291], [SMPP](#)^[290].

Usage:

SENDMAIL sends an email message from Take Command or 4NT via SMTP. The text of the message can be entered either on the command line or read from a text file.

Before you can use SENDMAIL, you must either set the [MailServer](#)^[107] and [MailUser](#)^[107] .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](#)^[279], a [subject](#)^[279], and [text](#)^[279]. Optionally it may also have [attachments](#)^[279].

1. The **address** field contains one or more standard Internet email addresses:

```
[c:\] sendmail abc@xyz.com ...
```

If the **address** contains white space, the entire address field must be surrounded by quotes. You can

specify multiple destinations by separating the addresses with **commas** and enclosing the entire string in quotes (all addresses will appear in the "To:" header sent to all recipients). You can add **BCC** ("blind copy") addresses by prefacing the desired target(s) with "**bcc:**". For example:

```
[c:\] sendmail "bob@bob.com bcc:joe@joe.com" Test Hello!
```

will send the text "Hello!" with subject "Test" to "bob@bob.com" with a blind copy to "joe@joe.com".

2. The **subject** will appear as the subject line in the message. If it contains white space, it must be surrounded by quotes.

3. The **message** may either be entered on the command line or be placed in a text file. If it is in a text file, the file must be less than 32,768 bytes long. 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. You can use the same approach to send the text content of the clipboard (@clip:) or the console (@con:):

```
[c:\] sendmail abc@xyz.com Party @c:\messages\invitation.txt
[c:\] sendmail abc@xyz.com Party @clip:
[c:\] type myfile.txt | sendmail abc@xyz.com Party @con:
```

SENDMAIL uses the current values for [MailAddress](#)^[106], [MailPassword](#)^[107], [MailPort](#)^[107], [MailServer](#)^[107] and [MailUser](#)^[107]. If you wish to temporarily alter those values, you can do so with the [OPTION](#)^[253] command, e.g:

```
OPTION //MailAddress="Bill Shakespeare <bard@globe.com>"
SENDMAIL ...
```

Note: SENDMAIL processing is provided through files *ipworks6.dll* and *ipwssl6.dll* included in the 4NT and Take Command distribution packages. These files are installed by default in the command processor's directory, but if you use both 4NT and Take Command, and they are installed in different directories, as recommended, you may find it more convenient to only keep **one** copy of *ipworks6.dll* and *ipwssl6.dll* in a common location, such as your "c:\windows\system32" directory.

Options:

/A file (Attachment): Attach **file** to the email message. The **/A** switch and the name of the file to attach must appear *before address*. Any file name that contains spaces or special characters must be quoted. You can send multiple files by repeating the **/A** switch and the additional file(s) to send. For example:

```
sendmail /a file1 /a "d:\path\My file2" abc@xyz.com ...
```

Note: Only send "attached" data when you are sure the intended recipient can and will accept that type of material.

/D (Delivery Notification): Request Delivery Notification.

/In (Importance): Set the Importance where **n** is:

1	High
2	Normal (default)
3	Low

/Pn (Priority): Set the Priority where **n** is:

0	Unspecified (default)
1	Normal

2 Urgent
3 Non Urgent

/V (Verbose): Show all the interaction with the server, except the message header and message body text.

4.92 SET

Purpose: Display, create, modify, or delete environment variables.

Format: SET [/A] [/D] [/P] [/R [*file...*]] [/S] [/U] [/V] [*name*[=*value*/*prompt*]]

file: One or more input files to read variable definitions from.
name: The name of the environment variable to define or modify.
value The new value for the variable.
prompt Optional input prompt for the "/P name=" option

/A(rithmetic)^[28†] /R(ead from file)^[28†]
/D(efault)^[28†] /S(ystem)^[28†]
/E(nvironemnt, too)^[28†] /U(ser)^[28†]
/P(ause or Prompt)^[28†] /V(olatile)^[28†]

See also: ESET^[20†] and UNSET^[31‡].

Usage:

Every program and command inherits an environment^[33‡], which is a list of variable *names*, each of which is followed by an equal sign and some text representing its **value**. Many programs use entries in the environment to modify their own actions. The command processor itself uses several environment variables^[33‡].

(TC) You can also create or modify environment variables with the Environment Window^[74†]. 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:\UTIL
CMDLINE=C:\TCMD\TCSTART.CMD
```

If you enter a partial name, SET will display the environment variables whose names begin with the specified string. For example, to display the variables whose names start with "pa":

```
[c:\] set pa
```

If there is only a single argument and it contains wildcards (* or ?), SET will display all matching environment variables. You cannot use wildcards to display the registry variables (**/D**, **/S**, **/U**, and **/V**).

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 and the text after the equal sign will be left just as you entered it (however, case is ignored when looking for a variable; for example **MyVar**, **myvar**, and **MYVAR** all refer to the same variable). 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 command processor [internal variables](#) ^[342], 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](#) ^[207] command. To remove variables from the environment, use [UNSET](#) ^[312], or type SET plus a variable name and an equal sign:

```
[c:\] set mine=
```

The variable's **name** is limited to a maximum of 80 characters. The **name** and **value** together cannot be longer than 8,191 characters.

Note: You cannot use SET to modify [GOSUB variables](#) ^[227].

The size of the environment is set automatically, and increased as necessary as you add variables.

Registry Variables: Windows stores some of its own variables in the registry. This includes Default, System, User, and Volatile variables. Those variables can be manipulated with the SET command's [/D](#) ^[287], [/S](#) ^[287], [/U](#) ^[287] and [/V](#) ^[287] switches respectively. For example, to display the contents of volatile variable **clientname**, use:

```
[c:\] set /v clientname
```

Note that setting a registry variable will set the variable in the local environment only if you also use the [/E](#) ^[287] option together with one of the switches [/D](#) ^[287], [/S](#) ^[287], [/U](#) ^[287], or [/V](#) ^[287].

User variables are user-specific, and **volatile** variables are only valid for the current Windows session. Use caution when directly modifying registry variables as they may be essential to various Windows processes and applications.

Options:

/A (Arithmetic) Evaluate the argument to the right of the equal sign, place the result in the environment, and display it. You can use [@EVAL](#) ^[374] to perform the same task; SET **/A** is included for compatibility with *CMD.EXE*. For example, this command adds 2 and 2, and places 4 in the environment 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
```

- /D** (Default variable): Create/modify/delete a "default" variable in the registry ("HKU\DEFAULT\Environment").
- /E** (Environment, too). When used together with one of [/D](#)^[28], [/S](#)^[28], [/U](#)^[28], or [/V](#)^[28], set both the registry variable and the local environment variable.
- /P** (Pause or Prompt)

When used without a variable name, wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[65].

When used with a variable name and an optional prompt string, e.g. "**set /p myvar=Enter value**", emulates the CMD.EXE behavior of allowing entry of a value for the variable. This is provided for compatibility reasons only. For more flexibility, use the [INPUT](#)^[233] command.

- /R** (Read) 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](#)^[25] for the command processor. 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:

```
[c:\] set > varlist
[c:\] 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](#)^[27]. 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.

- /S** (System): Create/modify/delete a "system" variable in the registry ("HKLM\System\CurrentControlSet\Control\Session Manager\Environment").
- /U** (User variable): Create/modify/delete a "user" variable in the registry ("HKCU\Environment").
- /V** (Volatile): Create/modify/delete a "volatile" variable in the registry ("HKCU\Volatile Environment").

4.93 SETDOS

Purpose: Display or set the command processor configuration.

Format: SETDOS [/A?] [/C?] [/D?] [/E?] [/Fn.n] [/G??] [/I[+|-] *command*] [/M?] [/N?] [/P?] [/S?:?] [/V?] [/X[+|-]n]

/A (NSI)	/M (ode for editing)
/C (ompound)	/N (o clobber)
/D (escriptions)	/P (arameter character)
/E (scape character)	/S (hape of cursor)
/F (ormat for @EVAL)	/V (erbose)
/G (numeric separators)	/X (expansion, special characters)
/I (nternal)	

See also: [OPTION](#) ^[253].

Usage:

SETDOS allows you to customize certain aspects of the command processor 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 [configuration directives](#) ^[97] in the [.INI file](#) ^[89], and can also be changed from the [configuration dialogs](#) ^[82].. The name of the corresponding directive is listed 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 [TCSTART \(TC\)](#) or [4START \(4NT\)](#) or other startup file (see [Automatic Batch Files](#) ^[64]), in aliases, or at the command line.

Note: The functionality of the "/Y" option ("debug", no longer supported) of previous versions has been moved to the new [BDEBUGGER](#) ^[149] command.

Options:

- /A** (TC) (ANSI) [[ANSI](#) ^[99]] This option determines whether [ANSI X3.64 support](#) ^[64] is enabled. **/A1** enables ANSI X3.64 string processing in the Take Command window. The default of **/A0** disables ANSI X3.64 strings. See the [ANSI X3.64 Commands Reference](#) ^[456] for a list of the ANSI X3.64 sequences supported by Take Command. Also see the [ANSI](#) ^[99] directive and the [_ANSI](#) ^[346] internal variable.
- /C** (Compound character): [[CommandSep](#) ^[100]] This option sets the character used for separating multiple commands on the same line. The default value is the ampersand [**&**]. You cannot use any of the [redirection](#) ^[67] characters (**| > <**), or a space, tab, comma, or equal sign as the command separator. The command separator is saved by [SETLOCAL](#) ^[287] and restored by [ENDLOCAL](#) ^[206]. The following example changes the separator character to a tilde [**~**]:


```
[c:\] setdos /c~
```
- /D** (Descriptions) [[Descriptions](#) ^[102] and [DescriptionName](#) ^[102]] This option controls whether file processing commands like [COPY](#) ^[164], [DEL](#) ^[172], [MOVE](#) ^[246], and [REN](#) ^[276] process file descriptions along with the files they belong to. **/D1** turns description processing on, which is the default. **/D0** turns description processing off. Also see the [Descriptions](#) ^[102] directive.

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** (Escape character) [[EscapeChar](#)^[103]] This option sets the character used to suppress the normal meaning of the following character. Any character following the [escape character](#)^[24] will be passed unmodified to the command. The default escape character is a caret (^). You cannot use any of the [redirection](#)^[64] characters (| > <) or a space, tab, comma, or equal sign as the escape character. The escape character is saved by [SETLOCAL](#)^[287] and restored by [ENDLOCAL](#)^[208]. Certain characters (**b**, **c**, **e**, **f**, **k**, **n**, **q**, **r**, **s**, and **t**) have special meanings when immediately preceded by the escape character.
- /F** (Format for @EVAL) [[EvalMax](#)^[103], [EvalMin](#)^[103]] This option lets you set default decimal display precision for the [@EVAL](#)^[374] variable function. The maximum precision is 20 digits to the left of the decimal point and up to 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 [EvalMax](#)^[103] and [EvalMin](#)^[103] directives to set the precision when the command processor starts; see the [@EVAL](#)^[374] function if you want to set the display precision for a single computation.

- /G** (Numeric separators) [[DecimalChar](#)^[102], [ThousandsChar](#)^[114]] This option sets the [decimal](#)^[102] and [thousands](#)^[114] 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](#)^[374], numeric [IF](#)^[227] and [IFF](#)^[228] tests, version numbers, and other similar uses. The thousands separator is used for numeric output, and is skipped when performing calculations in [@EVAL](#).

- /I** (Internal) 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 force the command processor to use an external command:

```
[c:\] setdos /i-list
```

To re-enable all disabled commands use **/I***.

- /M** (Mode)[[EditMode](#)^[102]] 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 4NT). To start in insert mode, use **/M1** (the default in TC). Also see the [EditMode](#)^[102] directive.
- /N** (No clobber) [[NoClobber](#)^[107]] This option controls output [redirection](#)^[64]. **/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. Also see the [NoClobber](#)^[107] directive.
- /P** (Parameter character) [[ParameterChar](#)^[107]] This option sets the character used after a

percent sign to specify all or all remaining command-line arguments in a [batch file](#)^[32h] or [alias](#)^[13h]. The default value is the dollar sign [\$]. The parameter character is saved by [SETLOCAL](#)^[28h] and restored by [ENDLOCAL](#)^[20h].

/S (**4NT**) (Shape) [[CursorOver](#)^[10h], [CursorIns](#)^[10h]] The size is entered as a percentage of the total character **height**. The default values are 10:100 (a 10% underscore cursor for overstrike mode, and a 100% block 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 1% - 100%. (You cannot disable the cursor; if you specify a size of 0, the function will fail.)

(**TC**) (Shape) [[CursorOver](#)^[10h], [CursorIns](#)^[10h]] The size is entered as a percentage of the total character **width**. The default values are 100:15 (a 100% or block cursor for overstrike mode, and a 15% or thin line cursor for insert mode).

If either value is -1, the command processor will not attempt to modify the cursor shape at all. You can retrieve the current cursor shape values with the **%_CI** and **%_CO** internal variables. See also the [CursorOver](#)^[10h] and [CursorIns](#)^[10h] directives.

/V (Verbose) [[BatchEcho](#)^[10h]] This option controls the default for command echoing in batch files.

/V0 disables echoing of batch file commands unless [ECHO](#)^[19h] is explicitly set ON.

/V1, the default setting, enables echoing of batch file commands unless [ECHO](#)^[19h] is explicitly set OFF. Also see the [BatchEcho](#)^[10h] directive.

/V2 forces echoing of all batch file commands, even if [ECHO](#)^[19h] is set OFF or 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](#)^[19h] OFF command(s) within it. **/V2** is intended for debugging, and can be set with SETDOS, but not with the configuration dialogs or the [BatchEcho](#)^[10h] directive in the [.INI file](#)^[8h]. For more information on batch file debugging, see the [BDEBUGGER](#)^[14h] and [Debugging Batch Files](#)^[32h] topics.

/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.

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, variable function evaluation, 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 quote marks ["] and square brackets).

8 Escape character

If nested alias expansion is disabled (/X-2), the first alias of a command is expanded but any aliases it invokes are not expanded. If nested variable expansion is disabled (X-4), 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-35678
... [perform text processing here]
setdos /x0
```

4.94 SETLOCAL

Purpose: Save a copy of the current disk drive, directory, environment, alias list, and special characters.

Format: SETLOCAL

See also: [ENDLOCAL](#) ^[200].

Usage:

SETLOCAL is valid only in batch files to save

- the default disk drive and directory,
- the environment,
- the alias list, and
- the special character set (command separator, escape character, parameter character, decimal separator, and thousands separator).

After using SETLOCAL, you can change the values of any or all of the above, and later restore the original values with a simple [ENDLOCAL](#) ^[200] command, or just by exiting the batch file.

SETLOCAL saves neither history, nor function definitions.

For example, this batch file fragment saves everything, removes all aliases so that user aliases will not affect batch file commands, 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 may be nested up to 16 levels deep in each batch file. You can also have multiple, separate SETLOCAL / ENDLOCAL pairs within a batch file, and nested batch files can each have their own SETLOCAL / ENDLOCAL pairs.

SETLOCAL does not override the LocalAliases=No setting. Consequently changing aliases inside a SETLOCAL/ENDLOCAL pair affect the definition of aliases of other concurrently executing sessions of the current command processor.

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. If you invoke one batch file from another without using CALL, the first batch file is terminated, and an automatic ENDLOCAL is performed; the second batch file inherits the settings as they were prior to any SETLOCAL.

You can "export" modified variables and aliases from inside a SETLOCAL / ENDLOCAL block. See [ENDLOCAL](#) ^[200] for details.

4.95 SHIFT

Purpose: Allows the use of more than 512 parameters in a batch file.

Format: SHIFT [*n* | /*n*]

n: Number of positions to shift

Usage:

SHIFT is provided for compatibility with older batch files, where it was used to access more than 10 parameters. 4NT and Take Command support 512 parameters (%0 to %511), 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* (i.e., after SHIFT -1, the former %0 is restored, %0 becomes %1, %1 becomes %2, etc.).

SHIFT also affects the special command-line tail parameters %*n*\$ 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.

Note: For compatibility with CMD.EXE, the SHIFT command does not alter the contents or order of the parameters returned by %*. See [Batch File Parameters](#)^[322] for details.

4.96 SHORTCUT

Purpose: Create a shortcut.

Format: SHORTCUT *command [args dir desc link mode]*

command: Command to execute
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 directory or folder on your system. You can run any Windows shortcut from the command processor by entering the name of the .LNK or .PIF file on the command line.

If you provide a single argument (a link file name), SHORTCUT will display the values for that link.

SHORTCUT requires 6 arguments; to leave an argument blank, enter 2 quote marks [""] in its place. Other arguments must be enclosed in quote marks if they include white space or other special characters.

The **command** is the full path of 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](#)^[145]) 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 "" 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.

Note: If you want the shortcut to appear on the Windows desktop, you should include the full path to the desktop directory in the command. In most Windows configurations, that directory can be referenced symbolically as %userprofile\desktop\.

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.

See [Desktop Integration](#)^[8] for additional information on shortcuts use.

4.97 SHRALIAS

Purpose: Retains global command history, directory history, alias and user function lists in memory when the command processor is not running.

Format: SHRALIAS [/U]

/U(nload)

Usage:

When you close all command processor sessions, the memory for the global command history, global directory history, global alias and global function lists is released. If you want the lists to be retained in memory even when the command processor is not 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 the command processor is not running. *SHRALIAS.EXE* must be stored in the same directory as the command processor or in a directory on your PATH. You cannot run **SHRALIAS.EXE** directly, it must be invoked internally 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 the command processor running with global alias, global function, global command history, or global directory history enabled. If no global list is found, SHRALIAS will display an error.

If you start SHRALIAS from a temporary command processor session which exits after starting SHRALIAS, the command processor 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](#)^[175] command after SHRALIAS is loaded and before your session exits.

SHRALIAS would not work properly in detached sessions (e.g. those started with [DETACH](#)^[178], or with the AT utility), due to security issues within Windows. Therefore the SHRALIAS command is ignored in detached sessions.

For more information about global histories, function and alias lists, see [Local and Global History Lists](#)^[161], [Local and Global Functions](#)^[218], [Local and Global Aliases](#)^[138].

Note: SHRALIAS shares the global lists (alias, history, directory history, and user functions) between both **4NT** and **Take Command**.

Option:

/U (Unload) Shuts down *SHRALIAS.EXE*. All global command history, directory history, function and alias lists will be released from memory when the last copy of the command processor exits unless SHRALIAS is loaded again before that time.

4.98 SMPP

Purpose: Send simple text (**SMS**) messages, typically to text-enabled cellular phones and similar devices.

Format: SMPP server username password recipient message

server	SMS server name
username	User name for the SMS server
password	Password for the SMS server
recipient	Phone number or dotted IP of an SMS-enabled device
message:	The message to send

See also: [SENDMAIL](#)^[279], [SNPP](#)^[291].

Usage:

SMPP sends **message** through standard Internet Paging Gateways. Depending on your system configuration, you may need to start an Internet connection before you use SMPP. See your service provider for specific requirements.

Note: SMPP processing is provided through files *ipworks6.dll* and *ipwssl6.dll* included in the 4NT and Take Command distribution packages. Those files are installed by default in the command processor's directory.

4.99 SNPP

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

See also: [SENDMAIL](#)^[279], [SMPP](#)^[290].

Usage:

SNPP sends **message** to alphanumeric pagers through standard Internet Paging Gateways. Depending on your system configuration, you may need to start an Internet connection before you use SNPP.

Note: SNPP processing is provided through files *ipworks6.dll* and *ipwssl6.dll* included in the 4NT and Take Command distribution packages. Those files are installed by default in the command processor's directory.

4.100 START

Purpose: Start a program in another session or window.

Format: START ["*window title*"] [/AFFINITY=*n*] [/ABOVENORMAL] [/B] [/BELOWNORMAL] [/C] [/CM] [/CMSTDIO] [/D*path*] [/FS] [/HIGH] [/I] [/INV] [/K] [/L] [/LA] [/LD] [/LF] [/LH] [/LOW] [/MAX] [/MIN] [/NORMAL] [/POS=*x,y,width,height*] [/REALTIME] [/SEPARATE] [/SHARED] [/SIZE=*rows,cols*] [/WAIT] [/WIN] [/PGM] "*progrname*" [*command*]

window title:	Title to appear on title bar.
path:	Startup directory.
progrname	Program name (not the session name).
command:	Command to be executed.

/ABOVENORMAL	/LD (local directory history)
/AFFINITY (multiple CPUs)	/LF (local functions)
/B (no new console)	/LH (local history list)
/BELOWNORMAL	/LOW (low priority)
/C (lose when done)	/MAX (imized)
/CM (Caveman)	/MIN (imized)
/CMSTDIO (Caveman stdio)	/NORMAL (priority)
/D (irectory)	/PGM (program name)
/FS (full screen)	/POS (ition of window)
/HIGH (priority)	/REALTIME (priority)
/I (nherit environment)	/SEPARATE (separate session)
/INV (isible)	/SHARED (shared session)
/K (eep when done)	/SIZE (screen buffer size)
/L (ocal lists)	/WAIT (for session to finish)
/LA (local aliases)	/WIN (indowed session)

See also: [DETACH](#)^[178].

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 command processor session. If you add a **command**, START will begin a new session or window and execute that command.

START will return to the command processor 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 **window title**, if it is included, will appear on the task list and Alt-Tab displays. The **window title** must be enclosed in quotation marks and cannot exceed 80 characters. If the **window 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 **window title**; if there is a second quoted string it is assumed to be the **command**. As a result, 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 **window 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"
```

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

START offers a large number of switches to control the session you start. In most cases you need only a few switches to accomplish what you want. The list below summarizes the most commonly used START options, and how you can 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 Windows sessions); **/K** allows you to keep the session open and go to a prompt (the default for character mode sessions).

If the **progrname** is in the "App Paths" registry, its associated "Path" value (if it exists) is inserted into the beginning of the PATH in the environment inherited by the program.)

If the **progrname** is the name of a directory instead of an executable program, 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.)

The Process ID of the detached session or program is returned in the **STARTPID** [356] variable.

Options:

/ABOVENORMAL (Windows 2000 and above) Set the priority above normal.

/AFFINITY=n On multiple-CPU machines, set the processor affinity for this process.
Acceptable values are **0** to **n-1** (where **n** is the number of available CPUs)

/BELOWNORMAL (Windows 2000 and above) Set the priority below normal.

/B (**4NT**) 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.

/C (Close) Close the session or window when the application ends.

/CM (Caveman) (**TC**) Run a DOS or character-mode application under **Caveman** [79]. Use this option to force an application to run under Caveman even if Caveman is not specifically enabled for that application. For more details see [Console Applications and the Console Window](#) [78]. **/CM** will be ignored if Caveman is not enabled via the [Caveman Applications Dialog](#) [87] (accessible via the **Options** menu). Take Command always waits for the application to finish when **/CM** is used, even if you do not use **/WAIT**.

/CMSTDIO (Caveman with Standard I/O) (**TC**) Run a DOS or character-mode application under Caveman as with **/CM**, but using Caveman's STDIO mode. For more details on STDIO mode see [Console Applications and the Console Window](#) [78] and the [Caveman Applications Dialog](#) [87].

/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 (Full Screen) Start the console application in full-screen mode.

/HIGH Start the window at high priority.

/I (Inherit environment) Inherit the default environment, if any, rather than the current environment.

/INV (Invisible) 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** (Keep session or window at end) The session or window continues after the application program ends. Use the [EXIT](#)^[20b] command to end the session.
- /L** (Local lists) Start the command processor with local alias, function, and history lists. This option combines the effects of **/LA**, **/LD**, **/LF**, and **/LH** (below).
- /LA** (Local Alias list) Start the command processor with a local alias list. See [ALIAS](#)^[13b] for information on local and global aliases.
- /LD** (Local directory history list) Start the command processor with a local directory history list. See [Local and Global History Lists](#)^[16] for information on local and global directory history.
- /LF** (Local Function list) Start the command processor with a local function list. See [FUNCTION](#)^[21b] for information on local and global functions.
- /LH** (Local History list) Start the command processor with a local history list. See [Local and Global History Lists](#)^[16] for information on local and global history lists.
- /LOW** Start the window at low priority.
- /MAX** (Maximized) Start the session or window maximized.
- /MIN** (Minimized) Start the session or window minimized.
- /NORMAL** Start the window at normal 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** *which must be the last option for START*. You can use **/PGM** to allow **START** to differentiate between a quoted long filename and a quoted title for the session.
- /POS** (Position) Start the window at the specified screen position. The syntax is **/POS=x, y, width, height** where the values are specified in pixels. **X** and **Y** refer to the horizontal and vertical positions, respectively, of the top left corner of the window relative to the top left corner (0,0) of the screen.
- /REALTIME** Start the window at realtime priority.
- /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; the switch is included only for compatibility with *CMD.EXE*.
- /SIZE=rows,columns** Specify the screen buffer size. **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. **(TC)** Under Windows 98 and ME, Take Command always waits for DOS programs run in its console window (programs that are not started from *.PIF* files). See [Console Applications and the Console Window](#)^[78] for details.

/WIN Start the new console session as a window.

4.101 SWITCH

Purpose: Select commands to execute in a batch file 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,: 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](#) ^[227], and [IFF](#) ^[228].

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 none of 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](#)^[27], 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.

You can exit from all SWITCH / ENDSWITCH processing by using GOTO to a line past the last ENDSWITCH.

4.102 TAIL

Purpose: Display the end of the specified file(s).

Format: TAIL [/A:[-][+]*rhsad*] /C*n* /F /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 [@file lists](#)^[49] for details).

/A: (Attribute select)	/N (umber of lines)
/C (number of bytes)	/P (ause)
/F (ollow)	/Q (uiet)
/I" <i>text</i> " (match description)	/V (erbose)

See also: [HEAD](#)^[224], [LIST](#)^[237], [TYPE](#)^[308].

File Selection

Supports extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46].

Internet: Can be used with [FTP servers](#)^[52], including HTTP/HTTPS files, e.g.

```
tail "http://jpsoft.com/notfound.htm"
```

Usage:

The TAIL command displays the last part of a file or files. It is normally only useful for displaying ASCII text files (i.e. alphanumeric characters arranged in lines separated by CR/LF). Executable files (.COM and .EXE) and many data files may be unreadable when displayed with TAIL because they include non-alphanumeric characters or unusual line separators.

You can press **Ctrl-S** to pause TAIL's display and then any key to continue.

The following example displays 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 [Highlighting and Copying Text](#)^[76] for additional information on CLIP:.

- **FTP Usage**

TAIL can also display files on [FTP servers](#)^[52]. The URL must be enclosed in quote marks. For example:

```
[c:\] tail "ftp://jpsoft.com/index"
```

You can also use the [IFTP](#)^[229] command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want.

- **NTFS File Streams**

TAIL supports file streams on NTFS drives. 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.

- **Pipes**

TAIL can optionally be used with an input [pipe](#)^[64]. For example:

```
[c:\] dir | tail /n2
```

This is not ordinarily feasible in Windows because pipes can't be "rewound", and therefore the pipe has to be written to a temporary memory buffer and the TAIL taken from there. Consequently, this limits the amount you can actually display in TAIL to < 1Mb when the input is piped.

Options:

/A: (Attribute select): Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[39] for information on the attributes which can follow **/A:**. Do not use **/A:** with @file lists. See [@file lists](#)^[49] for details.

/C: Display the specified number of bytes. /C accepts a b, k, or m modifiers at the end of the

number. **b** is the number of 512-byte blocks, **k** is the number of kilobytes, and **m** the number of megabytes.

- /F** Continue monitoring the file and display new lines until the file is closed.
- /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"[]"**. Do not use **/I** with @file lists. See [@file lists](#)^[49] for details.
- /N n** Display **n** lines. The default is **10**.
- /N+x** Start displaying with the **x**th line from the beginning of the file (the default is to end with the last line). For example, "TAIL /N+5 file" will display starting at the 5th line in the file.
- /P** (Pause): Pause and prompt after displaying each page.
- /Q** (Quiet): Do not display a header for each file. This is the default behavior, but an explicit **/Q** may be needed to override an alias that forces **/V**.
- /V** (Verbose): Display a header for each file.

4.103 TASKEND

Purpose: End the specified process.

Format: TASKEND [/F] pid | name | "title"

pid: The process ID
name: The process name
title: Window title

/F(orce)

Also see: [TASKLIST](#)^[299], [PID](#)^[354], [DETACHPID](#)^[351], [WINTITLE](#)^[356]

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 quote marks. You can use wild cards and extended wildcards in the window title.

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 (Force) 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.

4.104 TASKLIST

Purpose: Display a processes list

Format: TASKLIST [/O] [/P] [name]

name: Process name or window title

/O(rder by PID)

/P(ause)

Also see: [TASKEND](#)^[298].

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 task name that you wish to see. The name can contain [wildcards and extended wildcards](#)^[36].

Options:

/O (Order): Sort the output by Process ID (PID).

/P (Pause) Wait for a key to be pressed after each screen page before continuing the display.

4.105 TCTOOLBAR

Purpose: Change the tool bar buttons.

Format: TCTOOLBAR [/U] button [, flags, text, command]

button: The button number (1 – 32)

flags: 0=Echo, 1=Echo & 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 dialog](#)^[72] available from the [Options](#)^[70] 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 like

```
[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 quote marks.

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.

Option:

/U (Update): Write the changed button definition to the *TCMD32.INI* file so that it will be included the next time Take Command starts.

4.106 TEE

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](#)^[317], and the [piping](#)^[64] and [redirection](#)^[61] options.

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.

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 the output file *TCS.DAT*:

```
[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.

Remember that when using TEE with a pipe under 4NT and Take Command, the programs on the two ends of the pipe run simultaneously, not sequentially as they did in the DOS world.

See [Piping](#)^[64] for more information on pipes.

Option:

/A (Append) Append to the file(s) rather than overwriting them.

4.107 TEXT

Purpose: Display a block of text in a batch file.

Format: **TEXT**

.
.
.
ENDTEXT

See also: [ECHO and ECHOERR](#)^[198], [SCREEN](#)^[273], [SCRPUT](#)^[274], and [VSCRPUT](#)^[314].

Usage:

TEXT can only be used in batch files. Both **TEXT** and **ENDTEXT** must be entered as the only commands on their respective lines.

The **TEXT** command is useful for displaying menus, tables, special characters, or multiline messages. **TEXT** will display all subsequent lines in the batch file until terminated by **ENDTEXT**.

In the lines between **TEXT** and **ENDTEXT** *special characters are not parsed*. As a consequence, no environment variable expansion or other processing is performed, and all lines are displayed exactly as they are stored in the batch file, subject only to the choice of font and codepage differences, if any, between the program which created the file and that in effect during its execution. However, if the ANSI X3.64 interpretation option is enabled, you can change screen colors by inserting ANSI X3.64 escape sequences anywhere in the text block. The **ENDTEXT** command itself will not be displayed.

You can also use the [CLS](#)^[163] or the [COLOR](#)^[163] command to set the default screen colors before executing **TEXT**.

Redirecting TEXT output

To redirect or pipe the entire block of text, use [redirection](#)^[61] or [piping](#)^[64] on the **TEXT** command itself as shown in the Examples below, but not on the actual text lines or the **ENDTEXT** line. As with any other command, this redirection is not affected by redirection of all output of the batch file in the command which starts the batch file.

Warning: If the **TEXT** command is redirected or piped, and the redirection/piping **fails**, the lines of the batch file following the **TEXT** command are **executed** as if they were commands, causing potential harm. The simplest way to avoid trouble this may cause is to use the **ON ERROR** command before **TEXT**. See the second example below.

Examples

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
```

The example below uses **TEXT** to display or append to a file (specified as the optional parameter of the batch file) non-ASCII characters:

```
@echo off
setlocal
setdos /x-6
set dest=%@if[%# GT 0,>> %1,]
setdos /x+6
set repeat=0
on error (unset dest %+ goto PROBLEM)
:PROBLEM
iff %repeat GT 1 then
    echo Repeated problems - quitting
    quit
endiff
set repeat=%@inc[%repeat]
text %dest
+-----+
| Logical Drives |
+-----+
endtext
subst %dest
echo. %dest
if %_transient eq 1 .and. %# EQ 0 pause
endlocal
```

4.108 TIME

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](#) ^[170].

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 Sep 30, 2004 9:30:06
Enter new date (mm-dd-yy):
```

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.

Options:

/S server (Set from server) Sets the date and time from the specified internet time server. If no server is specified, TIME uses the server defined in the [TimeServer](#)^[97] .INI file entry (the default is clock.psu.edu).

/T (Display only) 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.

4.109 TIMER

Purpose: TIMER is a system stopwatch.

Format: TIMER [ON | OFF] [/1 /2 /3 /S]

ON: Force the stopwatch to start/restart

OFF: Force the stopwatch to stop

/1 (stopwatch #1)

/2 (stopwatch #2)

/3 (stopwatch #3)

/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 0.06 second.

Options:

- /1** Use timer #1 (the default).
- /2** Use timer #2.
- /3** Use timer #3.
- /S** (Split) 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).
- OFF** Stops the timer.

4.110 TITLE

Purpose: Change the window title.

Format: TITLE title

title: The new window title.

See also: the [TITLEPROMPT](#)^[342] variable and the [ACTIVATE](#)^[136] and [WINDOW](#)^[316] commands.

Usage:

TITLE changes the text that appears in the caption bar at the top of the command processor 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 "Title Test":

```
[c:\] title Title Test
```

4.111 TOUCH

Purpose: Change a file's date and [time stamps](#)^[443].

Format: TOUCH [/A:[-][+]*rhsad*] [/C] [/D[*acw*][*date*] [/E] [/F] [/I"text"] [/N] [/Q] [/R[:*acw*] *reffile*] [/S] [/T[:*acw*][hh:mm[:ss]]] [*@file*] *file*...

reffile: A file whose date and / or time stamps are to be copied to one or more other files.

file: One or more files whose date and/or time stamps are to be changed.

@file: A text file containing the names of the files to touch, one per line (see [@file lists](#)^[49] for details).

/A: (Attribute select)	/N (othing)
/C (reate file)	/Q (uiet)
/D (ate) [<i>date</i>]	/R (epeat) <i>reffile</i>
/E (No error messages)	/S (ubdirectories)
/F (orce read-only files)	/T (ime) [<i>time</i>]
/I (match descriptions)	

File Selection

Supports [attribute switches](#)^[39], extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46]. Use wildcards with caution on LFN volumes; see [LFN File Searches](#)^[47] 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 from your system clock. 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
12-12-04 11:13:58 D:\SOURCE\MAIN.C
12-12-04 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 retain the date but modify the time, use the **/D** switch without a parameter.

To modify the date but retain the time, use the **/T** switch without a parameter.

To transfer the date and / or time from one file to another use **/R**, see below for details.

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 NT/W2000/XP but will not work under Win9x/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 / or time stamps that are maintained for each file.

Note: The timestamp resolution on FAT volumes is limited by the operating system. In general the create time resolution is 10 milliseconds, the write time resolution is 2 seconds, and the access time resolution is one day.

Options:

- /A:** (Attribute select) Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[39] for information on the attributes which can follow **/A:**. Do not use **/A:** with @file lists. See [@file lists](#)^[49] for details.

- /C** (Create 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** (Date) Specify the date that will be set for the selected files. If the date is not specified, TOUCH will use the current date. 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**.

Use **/Da**, **/Dc**, or **/Dw**, followed by the date, to specify the last access, creation, or last write date.

Use **/D** without a parameter to retain the existing value.

- /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.

- /F** (Force 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"** (Match descriptions) Select files by matching text in their descriptions. The text can include [wildcards](#)^[36] 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"[]"**. Do not use **/I** with @file lists. See [@file lists](#)^[49] for details.

- /N** (No action) Display what would occur without actually doing it.

- /Q** (Quiet) Do not display the new date and time and the full name for each file.

- /R** (Repeat / Reference file) The time for the second (and any additional) file will be set to the current time for the first file. For LFN files, you can use **/R:a**, **/R:c**, or **/R:w**, followed by the time, to specify the last access, creation, or last write time stamp. However, files on FAT-based volumes do not have a last access time, so **TOUCH /R:a** will have no effect on such files.

- /S** (Subdirectories): TOUCH all matching files in the specified directory and its subdirectories. Do not use /S with @file lists. See [@file lists](#)^[49] for details.
- /T** (Time) Specify the time that will be set for the selected files in hh:mm format. If this option is omitted, TOUCH will use the [current time](#)^[356] as if you had specified "/T%_time".

Use **/Ta**, **/Tc**, or **/Tw**, followed by the time, to specify the last access, creation, or last write time. However, since files on FAT/VFAT/FAT32 volumes do not have a last access time, **TOUCH /Ta** will have no effect on such files.

Use **/T** without a parameter to retain the existing value.

4.112 TREE

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 one or more directories are specified, **TREE** will display a tree for each specified directory. If none are specified, the tree for the current working directory is displayed.

[/A](#)^[307] (SCII)

[/P](#)^[307] (ause)

[/B](#)^[307] (are)

[/S](#)^[307] (file size)

[/F](#)^[307] (iles)

[/T](#)^[307] (ime and date)

[/H](#)^[307] (idden directories)

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 (or not appear at all) on your screen. To correct the problem, use [/A](#)^[307], or configure the command processor to use a font which can display standard extended ASCII characters.

You can print the display, save it in a file, or view it with [LIST](#)^[237] by using standard [redirection](#)^[67] symbols. Be sure to review the [/A](#)^[307] option before attempting to print the **TREE** output. The options discussed below specify the amount of information included in the display.

Options:

- /A** (ASCII) 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** (Bare) Display the full pathname of each directory, without any of the line-drawing characters.

- /F** (Files) 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** (Hidden) Display hidden as well as normal directories. If you combine **/H** and [/F](#)^[307], hidden files are also displayed.
- /P** (Pause) Wait for a key to be pressed after each screen page before continuing the display. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[65].
- /S** (Size) Display the size of each file. This option is only useful when combined with [/F](#)^[307].
- /T** (Time and date) Display the time and date for each directory. If you combine **/T** and [/F](#)^[307], the time and date for each file will also be displayed.

By default, the time and date shown will be of the last modification. You can select a specific time and date stamp by using the following variations of **/T**:

- /T:a** Last access date and time (access time is not displayed on [VFAT](#)^[485] and [FAT32](#)^[477] volumes).
- /T:c** Creation date and time.
- /T:w** Last modification ("write") date and time (default).

4.113 TRUENAME

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: The [@TRUENAME](#)^[408] variable function.

Usage:

Default directories, as well as the MKLNK command, 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:\] truname d:\test.exe
c:\util\test\test.exe
```

4.114 TYPE

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 display, one per line (see [@file lists](#)^[49] for details).

/A: (Attribute select) **/L**(ine numbers)
/I"text" (match description) **/P**(ause)

See also: [HEAD](#)^[224], [TAIL](#)^[296], [LIST](#)^[237].

File Selection

Supports [attribute switches](#)^[39], extended [wildcards](#)^[36], [ranges](#)^[40], [multiple file names](#)^[45], and [include lists](#)^[46].

Internet: Can be used with [FTP and HTTP servers](#)^[52], e.g.

```
type "http://jpsoft.com/notfound.htm"
```

Usage:

The TYPE command displays a file. It is normally only useful for displaying ASCII **text** files (i.e. alphanumeric characters arranged in lines separated by CR/LF). Executable files (.COM and .EXE) and many data files may be unreadable when displayed with TYPE because they include non-alphanumeric characters or unusual line separators.

To display the files *MEMO1* and *MEMO2*:

```
[c:\] type /p memo1 memo2
```

You can press **Ctrl-S** to pause TYPE's display and then any key to continue.

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 [Redirection](#)^[61] for more 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](#)^[61] is useful if you want to add line numbers to a file, for example:

```
[c:\] type /l myfile > myfile.num
```

• **NTFS File Streams**

TYPE supports file streams on NTFS drives under Windows 2000 and above. 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](#)^[444] for additional details.

Options:

/A: (Attribute select) Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[39] for information on the attributes which can follow **/A:**. Do not use **/A:** with @file lists. See [@file lists](#)^[49] for details.

/I"text" (Match descriptions) Select files by matching text in their descriptions. The text can include [wildcards](#)^[36] 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"[]"`. Do not use `/I` with `@file` lists. See [@file lists](#)^[49] for details.

/L (Line numbers) Display a line number preceding each line of text.

/P (Pause) Prompt after displaying each page. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[65].

4.115 UNALIAS

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 from which to read the aliases to be undefined.

/Q(uiet)

/R(ead file)

See also: [ALIAS](#)^[138] and [ESET](#)^[207].

Usage:

The command processor maintains 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 remove one or more aliases by name, or 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](#)^[138] command, 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 *. UNALIAS /R accepts all of the alias definition formats you can use in a file for ALIAS /R.

(TC) You can also delete individual aliases with the [Alias dialog](#)^[74].

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 (Read) 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](#)^[138] command. 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
```

UNALIAS /R will read from STDIN if no filename is present and input is redirected.

4.116 UNFUNCTION

Purpose: Remove user-defined functions from the function list.

Format: UNFUNCTION [/Q /R file...] function...
or
UNFUNCTION *

function: One or more functions to remove from memory.

file: One or more files from which to read functions to be undefined.

/Q(uiet)

/R(ead file)

See also: [FUNCTION](#)^[218] and [ESET](#)^[207].

Usage:

The command processor maintains 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 remove one or more functions by name, or you can delete the entire function list by using the command **UNFUNCTION ***.

For example, to remove the function DDIR:

```
[c:\] unfunction ddir
```

To remove all the functions:

```
[c:\] unfunction *
```

To remove all the functions that begin with "DD":

```
[c:\] unfunction dd*
```

If you keep functions in a file that can be loaded with the [FUNCTION /R](#)^[218] command, 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 *.

(TC) You can also delete individual functions from the [Functions Dialog](#)^[74].

Options:

/Q (Quiet) Prevents UNFUNCTION from displaying an error message if one or more of the functions 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 (Read) 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](#) ^[218] command. 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
```

UNFUNCTION /R will read from STDIN if no filename is present and input is redirected.

4.117 UNSET

Purpose: Remove variables from the environment or the registry.

Format: UNSET [/D /Q /R file.../S /U /V] name...
or
UNSET *

name: One or more variables to removed.

file: One or more files from which to read variables to be removed.

[/D\(efault\)](#) ^[312]

[/S\(ystem\)](#) ^[312]

[/E\(nvironment, too\)](#) ^[312]

[/U\(ser\)](#) ^[312]

[/Q\(quiet\)](#) ^[312]

[/V\(olatile\)](#) ^[312]

[/R\(ead\)](#) ^[312]

See also: [ESET](#) ^[207] and [SET](#) ^[287].

Usage:

UNSET removes one or more variables from the environment or from the Windows Registry.

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](#) ^[74].

UNSET can be used in a batch file, in conjunction with the [SETLOCAL](#) ^[287] and [ENDLOCAL](#) ^[206] 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](#) ^[287] command and the general discussion of the [environment](#) ^[336].

Note: You cannot use UNSET with [GOSUB variables](#) ^[227].

Use caution when removing environment variables, and especially when using UNSET *. Many programs will not work properly without certain environment variables; for example, the command processor depends on PATH.

Registry Variables: **Default**, **System**, **User**, and **Volatile** registry variables can be manipulated with

the UNSET command's [/D](#)^[312], [/S](#)^[312], [/U](#)^[312] and [/V](#)^[312] switches, respectively. To remove the variable from both the registry and from the local environment, use *both* the [/E](#)^[312] switch *and* the registry variable selection switch together. For example, to remove volatile variable **myvar** from both the registry and the local environment, use:

```
[c:\] unset /v /e myvar
```

Use caution when directly removing registry variables as they may be essential to various Windows processes and applications.

Options:

- /D** (Default environment): Delete a "default" variable from the registry (HKU\DEFAULT\Environment).
- /E** (Environment, too). When used together with one of [/D](#)^[281], [/S](#)^[281], [/U](#)^[281], or [/V](#)^[281], unset both the registry variable and the local environment variable.
- /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** (Read) 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 [SET](#)^[281] for more details).

UNSET /R will read from STDIN if no filename is present and input is redirected.
- /S** (System) Delete a "system" variable from the registry (HKLM\System\CurrentControlSet\Control\Session Manager\Environment).
- /U** (User) Delete a "user" variable from the registry (HKCU\Environment).
- /V** (Volatile) Delete a "volatile" variable from the registry (HKCU\Volatile Environment)

4.118 VER

Purpose: Display the 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 6.01U   Windows XP 5.1
```

Option:

- /R** (Revision level) Display the command processor and operating system internal revision

level (if any), plus your command processor serial number and registered name.

4.119 VERIFY

Purpose: Enable or disable disk write verification or display the verification state.

Format: VERIFY [ON | OFF]

Usage:

Disk write verification cannot actually be enabled or disabled under Windows. The command processor supports **VERIFY** as a "do-nothing" command, for compatibility with CMD.EXE. This avoids "*unknown command*" errors in batch files which use the VERIFY command.

If used without any parameters, VERIFY will display the state of the verify flag:

```
[c:\] verify
VERIFY is ON
```

4.120 VOL

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 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: b:
Volume in drive A: is unlabeled
Volume in drive B: is BACKUP_2
```

VOL will also return volume information for UNC names.

See also: [@LABEL](#) ³⁹⁶.

4.121 VSCRPUT

Purpose: Display text vertically in the specified color.

Format: VSCRPUT row col [BRlght] fg ON [BRlght] bg text

row: Starting row
col: Starting column
fg: Foreground text color
bg: Background text color
text: The text to display

See also: [SCRPUT](#)^[274].

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, it uses the colors you specify to write the text. See [Colors and Color Names](#)^[467] 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.

(TC) The maximum **row** value is determined by the current height of the Take Command window. The maximum **column** value is determined by the current virtual screen width (see [Resizing the Take Command Window](#)^[77] 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**.

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
```

4.122 WHICH

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](#)^[318] (including keystroke aliases), and files. When a file matches an applicable [Executable Extension](#)^[48] or [Windows File Association](#)^[467], that data will be displayed. The exact information reported depends on the type of command or file you specify. For example:

```
[c:\] which cdd buildtree notepad test.btm test.exe test.xyz test.doc
```

```

donothing
CDD is an internal command
buildtree is an alias : cdd /s
notepad is an external: C:\windows\notepad.exe
test.btm is a batch file : C:\test.btm
test.exe is an external : C:\test.exe
test.xyz is an executable extension : C:\path\mybatch.btm C:\test.xyz
test.doc is associated with : C:\Program
Files\Office97\Office\WINWORD.EXE
donothing is an unknown command

```

If the command is an abbreviated alias, WHICH will display the full name; i.e.:

```

[c:\] alias opt*ions=*option
[c:] which opt
opt*ions is an alias : *option

```

WHICH can also recognize [REXX](#)^[334] files, [EXTPROC](#)^[334] files, and associated files.

Note: WHICH does not support wildcard specifications. Arguments must be actual commands or file names (including variable and function references as in "which %comspec"). If a filename includes white space or special characters, it must be enclosed in quote marks.

See [Executable Files and File Searches](#)^[458] for details on the order in which various locations are searched.

Also see [@SEARCH](#)^[406], [ASSOC](#)^[145], [FTYPE](#)^[217].

4.123 WINDOW

Purpose: Minimize or maximize the current window, restore the default window size, or change the window title.

Format: WINDOW [MAX | MIN | RESTORE | HIDE | TRAY | FS | WIN | TOPMOST | NOTOPMOST | TOP | BOTTOM | /POS=x,y,width, height | /SIZE=rows,columns | "newtitle"]

newtitle: A new title for the window

/POS(ition) **/SIZE** (of screen buffer)

See also: [ACTIVATE](#)^[136] and [TITLE](#)^[304].

Usage:

WINDOW is used to control the appearance and title of the current window. Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

4NT supports the MAX, MIN, RESTORE, HIDE, TRAY, FS, WIN, /POS, /SIZE, and "newtitle" options.

Take Command supports the MAX, MIN, RESTORE, HIDE, TRAY, TOPMOST, NOTOPMOST, TOP, BOTTOM, /POS, and "newtitle" options.

Note: You can specify only one WINDOW option at a time. The different options cannot be combined in a single WINDOW command. To perform multiple operations you must use multiple

WINDOW commands.

Options:

For both 4NT and Take Command:

/POS	(Position) Set the window position and size on the desktop. The values are specified in pixels. x and y refer to the position of the top left corner of the window relative to the top left corner (0,0) of the screen. The x and y values of the /POS option select the window's origin while the width and height values determine its size.
"newtitle"	Changes the window title. The title text must be enclosed in quote marks. The quotes will not appear as part of the actual title as displayed
MAX	Expands the window to its maximum size.
MIN	Reduces the window to an icon.
RESTORE	Returns the window to its default size and location.
HIDE	Makes the window invisible (to make the window visible again, use RESTORE).
TRAY	Moves the window to the taskbar tray.

For 4NT only:

/SIZE	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 <i>reduce</i> the size of the screen buffer; it can only be <i>increased</i> .
FS	Switches a 4NT window to full-screen mode.
WIN	Switches a 4NT window to windowed mode.

For Take Command only:

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.

4.124 Y

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](#)^[300], and the [piping](#)^[64] and [redirection](#)^[61] options.

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.

When using Y with a pipe you must take into account that the programs on the two ends of the pipe run simultaneously, not sequentially.

See [Piping](#)^[64] for more information on pipes.

5 Aliases & Batch Files

Whenever you have a command (internal or external) that you need to execute often, one that's too complex to be dependably typed manually at the [Command Line](#)^[10], one that needs to be part of an exact sequence of other commands, or one that you want to be able to easily repeat from another location or share with others, or you repeat very often and therefore want to have a very short name, you can store that command as part of a convenient ALIAS and/or batch file. This section discusses that process in detail.

- ▶ [Aliases](#)^[318]
- ▶ [Batch Files](#)^[321]
- ▶ [Special Character Compatibility](#)^[335]

5.1 Aliases

Much of the power of the command processor 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 shows you some examples of the power of aliases. See the [ALIAS](#)^[138] command for complete details about writing your own aliases. You can create aliases either from the command line, as described in this section, or **(TC)** with the [Aliases dialog](#)^[74] which is available from the [Utilities menu](#)^[70].

The simplest type of alias gives a new name to an existing command. For example, you could create a command called **R** (for **R**oot 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](#)^[138] command for full details.

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](#)^[29] 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 executes the [PUSHD](#)^[264] command, detects that the next command (**PLAY**) is another alias and executes the program *E:\MUSIC\PLAYER.EXE*, and — when that program exits — returns to the first alias, executes the [POPD](#)^[261] command, and returns 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](#)^[172] 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 [Temporarily Disabling Aliases](#)^[138] 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.EXE* 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](#)^[176] command will be executed instead. Aliases give you two simple ways to get around this problem.

First, you could define an alias that runs the program in question, but using a different name:

```
[c:\] alias desc=c:\winutil\describe.exe
```

Another approach is to use an alias to rename the internal command and use its original name for the

external program. The following example creates the alias **FILEDESC** for the **DESCRIBE**^[176] command, 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** followed by **Enter**:

```
[c:\] alias @Shift-F5=*dir /2/p
```

This alias will put the **DIR**^[179] 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, neither displaying it on the command line, nor waiting for you to press **Enter**, use two **@** 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. The next 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 temporarily changes directories, runs an internal or external command, and then returns to the current directory when that 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 arguments: **%1** means the first argument on the command line, and **%2\$** means the second and all subsequent arguments.

NEXT PARAGRAPH UNDER CONTRUCTION.

Aliases cannot use the indirect access to command parameters, e.g., **%[*n*]** (where *n* is an argument number) does not return the selected argument. This behavior is different from batch file behavior.

PREVIOUS PARAGRAPH UNDER CONTRUCTION.

See the **ALIAS**^[138] and **UNALIAS**^[310] commands for more information and examples.

5.2 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](#)^[138], 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.

Some of the topics included in this section are:

- ▶ [.BAT, .CMD, and .BTM](#)^[321]
- ▶ [Using .BAT Files Under 4NT](#)^[321]
- ▶ [Echoing in Batch Files](#)^[322]
- ▶ [Batch File Line Continuation](#)^[331]
- ▶ [Batch File Parameters](#)^[322]
- ▶ [Using Environment Variables](#)^[324]
- ▶ [Batch File Commands](#)^[325]
- ▶ [Interrupting a Batch File](#)^[327]
- ▶ [Automatic Batch Files](#)^[6]
- ▶ [Detecting the command processor](#)^[327]
- ▶ [Using Aliases in Batch Files](#)^[327]
- ▶ [Debugging Batch Files](#)^[329]
- ▶ [String Processing](#)^[329]
- ▶ [Batch File Compression](#)^[331]
- ▶ [Argument Quoting](#)^[332]
- ▶ [REXX Support](#)^[334]
- ▶ [EXTPROC Support](#)^[334]

5.2.1 .BAT, .CMD & .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, and the file is opened and closed to read each line. In the second mode the batch file is opened once, the entire file is read into memory, and the file is closed. Only the first mode (.BAT/.CMD) can be used for self-modifying batch files (which are rare).

The batch file's extension determines its mode. Files with a .BAT or .CMD extension 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](#)^[242] command.

5.2.2 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 Windows 98 to Windows NT/W2000/XP/2003. 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 Windows' DOS command processor, COMMAND.COM, for execution (yes, there is a skeletal DOS command processor in Windows). 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 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.

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 a DOS-based environment.

In most cases this difference will not affect your *.BAT* files, because you will not be loading DOS TSRs in Windows. 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 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 / 2003, it works well in this specific situation.

When invoking DOS programs from a 4NT batch file, we recommend that you enable the CONFIG.NT directive **NTCMDPROMPT** without which Windows tends to "forget" to return control to a calling 32-bit program (such as 4NT) and may leave you at an unexpected COMMAND.COM prompt. CONFIG.NT typically resides in the Windows "SYSTEM32" directory. See your Windows documentation for additional information.

5.2.3 Echoing in Batch Files

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](#)^[198] OFF and ECHO ON commands.
- ▶ The default setting can be changed with the [SETDOS](#)^[283] /V command, the "Default Batch Echo" checkbox on the [Startup tab](#)^[82] of the configuration dialogs, or the [BatchEcho](#)^[100] directive in the [INI File](#)^[89].

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
```

The command processor also has a command line echo that is unrelated to the batch file echo setting. See [ECHO](#)^[198] for details about both settings.

5.2.4 Batch File Parameters

Like [aliases](#)^[138], user-defined [functions](#)^[218] 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 or alias name) is separated into individual positional parameters (also called arguments or batch variables) by scanning for the spaces, tabs, and commas that separate them. For aliases and functions, a forward slash ("/") triggers the beginning of a new parameter (e.g. the string "foo/bar" is separated into parameters "foo" and "/bar").

These parameters are numbered from **%1** to **%511**. **%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 [Argument Quoting](#)^[332] 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 (or alias) as it was entered on the command line, **%#** contains the number of command line arguments, and **%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](#) ^[288] command. While these examples assume a [ParameterChar](#) ^[107] of "\$" (4NT and Take Command default value), your specific configuration may use another token (see [Special Character Compatibility](#) ^[335]).

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](#) ^[227] and [IFF](#) ^[228] commands.)

Batch files can also use [environment variables](#) ^[336], [internal variables](#) ^[342], and [variable functions](#) ^[357].

Microsoft's CMD.EXE compatibility:

For compatibility with Microsoft's *CMD.EXE*, the command processor also supports additional syntax to qualify references to the parameter of a batch file. Those alternatives, however, are usually best replaced by the more flexible [Variable Functions](#) ^[357].

CMD-style syntax	Expands to	Suggested replacement
%*	All parameters.	%\$
%~n	unquoted ("")	%@replace[%="",,%n]
%~fn	Fully qualified name of "%n"	%@full[%n]
%~dn	Drive letter portion of "%n"	%@left[2,%@full[%n]]
%~pn	Full path (no drive letter) of "%n"	%@right[-2,%@path[%@full[%n]]]
%~nn	Root name (no extension) of "%n"	%@name[%n]
%~xn	File extension of "%n"	.%@ext[%n]
%~sn	Fully qualified short name of "%n"	%@sfn[%@full[%n]]
%~an	File attributes of "%n"	%@attrib[%n]
%~tn	File date and time of "%n"	%@filedate[%n] %@filetime[%n]
%~zn	File size of "%n" in bytes	%@filesize[%n]
%~\$PATH:n	Full name of the first match for "%n" in the PATH	%@search[%n]

Notes:

As under CMD.EXE, the [SHIFT](#)^[288] command does not alter the contents or order of the parameters returned by %*.

In the special case where the argument to a "%~" variable is "0" (e.g. "%~f0"), the returned file name will always include the extension, as it does under CMD.EXE.

A "%~\$PATH:n" will return an empty string if the file referenced by %n is not found in the path.

Note that the tilde ("~") qualified references will trigger an error message when used improperly, e.g. if attempting to display the size of a string argument which is **not** the name of a file.

5.2.5 Using Environment Variables

Batch files can use [environment variables](#)^[336], [internal variables](#)^[342], [variable functions](#)^[357], or [user-defined functions](#)^[218]. 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](#)^[287] 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](#)^[312] command to remove these variables from the environment at the end of your batch file. You can use [SETLOCAL](#)^[287] and [ENDLOCAL](#)^[200] 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](#)^[374] to add a number to a text string), you'll get an error message.

5.2.6 Batch File Commands

Some commands are particularly suited to batch file processing. Each command is explained in detail in the [Command Reference](#)^[13]. Here is a list of some of the commands you might find most useful:

<u>ACTIVATE</u> ^[136]	activates another window.
<u>BEEP</u> ^[156]	produces a sound of any pitch and duration through the computer's speaker.
<u>CALL</u> ^[157]	executes one batch file from within another.
<u>CANCEL</u> ^[158]	terminates all batch file processing.
<u>CLS</u> ^[163]	clears the command processor screen
<u>COLOR</u> ^[163]	sets the command processor display colors.
<u>DO</u> ^[197]	starts a loop. The loop can be based on a counter, or on a conditional test like those used in IF and IFF. ENDDO terminates the loop.
<u>DRAWBOX</u> ^[198]	draws a box on the screen.
<u>DRAWHLINE</u> ^[198]	draws horizontal lines on the screen.
<u>DRAWVLINE</u> ^[197]	draws vertical lines on the screen.
<u>ECHO</u> ^[198]	sends text to the standard output device.
<u>ECHOS</u> ^[199]	sends text to the standard output device.
<u>ECHOERR</u> ^[198]	sends text to the standard error device.
<u>ECHOSERR</u> ^[199]	sends text to the standard error device.
<u>ENDLOCAL</u> ^[200]	restores the settings that were saved and allows specific variables to be exported (see <u>SETLOCAL</u> ^[287]).
<u>ENDTEXT</u> ^[307]	ends the block of text started with <u>TEXT</u> ^[307] .
<u>EVENTLOG</u> ^[203]	writes a string to the Windows application event log.
<u>FOR</u> ^[210]	executes commands for each file that matches a set of wildcards, or each entry in a list.
<u>GOSUB</u> ^[227]	executes a subroutine inside a batch file (see <u>RETURN</u> ^[272]).
<u>GOTO</u> ^[222]	branches to a different location in the batch file.
<u>IF</u> ^[227]	execute commands based on a test of string or numeric values, program exit codes, or other conditions.
<u>IFF</u> ^[228]	
<u>INKEY</u> ^[237]	collects keyboard input and store it in environment variables.
<u>INPUT</u> ^[233]	collects keyboard input and store it in environment variables.
<u>KEYSTACK</u> ^[236]	sends keystrokes to applications.
<u>LOADBTM</u> ^[242]	changes the batch file operating mode.
<u>MSGBOX</u> ^[257]	displays a dialog box with standard buttons like Yes, No, OK, and Cancel, and returns the user's selection.
<u>ON</u> ^[252]	initializes error handling for Ctrl-C / Ctrl-Break, or for program and command errors.
<u>PAUSE</u> ^[256]	displays a message and waits for the user to press a key.
<u>PDIR</u> ^[258]	creates a customized DIR-like display of directory contents.
<u>PLAYAVI</u> ^[260]	plays Windows .AVI files.
<u>PLAYSOUND</u> ^[267]	plays Windows sound files.
<u>QUERYBOX</u> ^[265]	displays a dialog box for text input.
<u>QUIT</u> ^[266]	ends the current batch file and optionally returns an exit code.
<u>REM</u> ^[269]	places a remark in a batch file.
<u>RETURN</u> ^[272]	terminates a subroutine (see <u>GOSUB</u> ^[227]).
<u>SCREEN</u> ^[273]	positions the cursor on the screen and optionally prints a message at the new location.
<u>SCRPUT</u> ^[274]	displays a message in color.
<u>SENDMAIL</u> ^[279]	sends an email message.

These commands, along with the internal variables and variable functions, make the enhanced batch file language extremely powerful. Your copy of the command processor includes a sample batch file, in the file *EXAMPLES.BTM*, that demonstrates some of the things you can do with batch files.

5.2.7 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**^[157] is ON or OFF, and whether the **ON BREAK**^[252] command is in use.

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**^[266] command; answering **A** is similar to the **CANCEL**^[158] command.

Note: (TC) When the **CUA**^[101] directive is set to "No", Ctrl-C is not available for interrupting commands. Use Ctrl-Break instead.

5.2.8 Detecting 4NT or Take Command

From a batch file, you can determine if 4NT or Take Command is loaded by testing for the variable function **@EVAL**^[374], with a test like this:

```
if "%@eval[2 + 2]%" == "4" echo %_cmdproc is loaded!
```

This test can never succeed in *CMD.EXE*. Other variable functions could also be used for the same purpose. The **_CMDPROC**^[349] internal variable tells you which specific command processor is running.

5.2.9 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:

```
setlocal
unalias *
setdos /e=%^ /c=%& /p=%$
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. Solitaire
    3. Internet
    4. Exit
endtext
echo.
inkey Enter your choice: %%userchoice
```



```
switch %userchoice
case 1
    input Enter the file name: %%fname
    if not exist fname error File does not exist
    in d:\letters c:\windows\wordpad.exe
case 2
    in d:\finance c:\windows\sol.exe
case 3
    in d:\comm c:\windows\iexplore.exe
case 4
    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**^[198] 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**^[264] 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**^[295] 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**^[287], **ENDLOCAL**^[200], and **UNALIAS**^[310] to clear out aliases before your batch file starts, and **SETDOS** to select the special characters you depend on, 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.

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.

Similarly, you can use the pseudovariables `%=` and `%+` to represent the command escape and command separator characters, respectively. There is no pseudovari-
parameter character, sorry.

5.2.10 Debugging Batch Files

4NT and Take Command include a built-in full-featured batch file debugger invoked with the BDEBUGGER command. The debugger gives you a detailed, step-by-step view of batch file execution, and will help solve particularly difficult batch file problems.

You can also do some simple debugging without loading the batch debugger.

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.

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 do not 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.

You can also use the TEE command to both view the output while the batch file is running and save it in a file for later examination.

5.2.11 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.

The command processor includes several features that make string processing easier. For example, you can use the INPUT, MSGBOX, and QUERYBOX commands for user input; the ECHO and ECHOERR, ECHOS and ECHOSERR, SCREEN, SCRPUT, and

[VSCRPUT](#)^[314] commands for output; and the [FOR](#)^[210] command or the [@FILEREAD](#)^[382] function to scan through the lines of a file. In addition, [variable functions](#)^[357] offer a wide range of [strings and character handling](#)^[359] capabilities.

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](#)^[287] command at the beginning of this batch file saves the environment and aliases. Then the [UNALIAS *](#)^[310] 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 a [INPUT](#)^[233] 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](#)^[416] function numbers the first word as 0, not as 1.

The [IFF](#)^[228] 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. (QUERYBOX 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](#)^[200] 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](#)^[61] 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](#)^[332] 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 when they are inside back-quotes.

If you use quote marks (ECHO "%STR"), the string entered by the user will be displayed properly, and so will the quotation marks. With quote marks, 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, quote marks, or redirection symbols can be virtually impossible to handle correctly.

One way to overcome these potential problems is to use the [SETDOS /X](#)^[283] command 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 the command processor, you may want to consider using the [REXX](#)^[334] language. Our products support external REXX programs for this purpose.

5.2.12 Batch File Line Continuation

The command processor will combine multiple lines in the batch file into a single line for processing when you include your [Escape Character](#)^[21] (the actual token or the symbolic "%="^[347] reference) as the very last character of each line to be combined (except the last). For example:

```
c:\> 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 [command line length limit](#)^[25].

5.2.13 Batch File Compression

You can compress your .BTM files with [BATCOMP](#)^[149]. That command condenses batch files by about a third and makes them unreadable with the [LIST](#)^[237] 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. You may also want to consider compressing batch files to save some disk space on the systems where compressed files are used.

The full syntax for the batch compression command is

```
BATCOMP [/Ekkkk /K][/Q][/O] InputFile [OutputFile]
```

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. For example, to compress *MYBATCH.CMD* and save the result as *MYBATCH.BTM*, you can use either one of these commands:

```
[c:\] batcomp mybatch.cmd
[c:\] batcomp mybatch.cmd 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.

By default, BATCOMP does not remove comment lines, i.e. lines starting with "REM" or ":", since in some instances, such as when a comment line occurs between a [TEXT](#)^[301] and *ENDTEXT*, the compressed file would behave differently from the original by not displaying the comment line. To override that default and force deletion of comment lines, use the */K* option. Lines starting with "REM >" (used to create a new, empty file, or to delete the old contents of one) are never considered comments to be removed.

The */Q* ("quiet") option suppresses informational messages from BATCOMP.

JP Software does not provide a utility to decompress batch files. If you use *BATCOMP*, 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 *.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](#)^[242] command.

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 *SOURCEMYBATCH.BTM* for the source file and *COMPMYBATCH.BTM* for the compressed version, or use *MYBATCHS.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).

If you plan to distribute batch files to users of different platforms, see [Special Character Compatibility](#)^[335] for important information on the command separator, escape character, and parameter character used in each product.

The current [BATCOMP](#)^[149] command replaces external program *BATCOM32.EXE* which was included in previous versions and is now obsolete.

5.2.14 Argument Quoting

As it [parses](#)^[29] the command line, the command processor looks for the ampersand [**&**] [command separator](#)^[24], [conditional commands](#)^[26] (**||** or **&&**), white space (spaces, tabs, and commas), percent signs [%] which indicate [variables](#)^[336] or [batch file](#)^[321] arguments to be expanded, and [redirection and piping](#)^[60] 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 single back quotes ['] or quote marks ["]. Although both back quotes and quote marks 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 quote marks. Redirection symbols inside quote marks are ignored. However, variable expansion **is** performed on expressions inside quote marks. The quote marks 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. The quotes keep characters together and reduce the number of arguments in the line.

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 that 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 in each case 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](#)^[138] for details.

You can disable and reenab back quotes and quote marks with the [SETDOS](#)^[283] /X command.

5.2.15 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 command processor batch language, especially if you need advanced string processing capabilities.

The REXX language is **not** built into the command processor, and must be obtained separately through add-on REXX software, such as Enterprise Alternatives' *Enterprise REXX*, Quercus's *Personal REXX*, IBM's *Object REXX*, or *Regina REXX*.

4NT and **Take Command** automatically recognize the presence of a REXX interpreter on your system. When the command processor loads, it asks Windows to locate specific REXX libraries associated with Enterprise REXX, Personal REXX, Object REXX, or Regina REXX. Specifically, it looks for *REGINA.DLL*, *WREXX32.DLL*, *RXREXX.DLL*, *REXX.DLL*, or *REXXAPI.DLL*. If a suitable library is found, the command processor checks to see if you are running a **.REX** or **.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 your REXX interpreter for processing.

(TC) When working with a REXX processor, Take Command automatically handles all input and output for the REXX program, and any standard REXX processor window for input and output is not displayed. 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.

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. The command processor uses the 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.

Also see the [@REXX](#)^[405] function.

5.2.16 EXTPROC Support

For compatibility with *CMD.EXE*, the command processor offers 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 the command processor as the external processor named on the EXTPROC line in the *.CMD* file. It will interpret the EXTPROC line as a command to reopen itself. The result will be an infinite loop that will continue until the computer runs out of resources and locks up.

5.3 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 [Multiple Commands](#)^[24]), the Escape Character (see [Escape Character](#)^[27]), and the Parameter Character (see [Batch File Parameters](#)^[32]).

The default values of each of these characters in each product is shown in the following chart.

	Separator	Escape	Parameter
4DOS	^	Ctrl-X	&
4NT / Take Command	&	^	\$
(symbolic reference)	%+	%=	

In your batch files and aliases, and even at the command line, you can smooth over these differences in three ways:

1. **Preferred:** use internal pseudovariables that contain the current special character, rather than using the character itself (see [%+](#)^[34] and [%=](#)^[34]). 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)
```

2. Select a consistent set of characters from the [Syntax tab](#)^[86] of the [configuration dialogs](#)^[82], or with [configuration directives](#)^[97] in the [.INI file](#)^[89]. For example, to set the 4NT or Take Command characters to match 4DOS, use these lines in the *.INI* file:

```
CommandSep = ^
EscapeChar = Ctrl-X
ParameterChar = &
```

3. In a batch file, use the [SETLOCAL](#)^[28] command to save the command separator, escape

character, and parameter character when the batch file starts. Then use `SETDOS`^[283] as described below to select the characters you want to use within the batch file. Use an `ENDLOCAL`^[206] command at the end of the batch file to restore the previous settings.

You can also use the `SETDOS`^[283] command 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 already 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 [`^`] 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 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%=Ctrl-X /p%=&
.....
endlocal
```

Warning: Whatever you do, make sure that each of the three special characters (*EscapeChar*, *CommandSep*, and *ParameterChar*) has a **unique** value. When you change one character, make sure the token you select is not already in use by another special character. A simple `SETDOS`^[283] can quickly remind you of your current settings. Using the same token for more than one special character will trigger erratic and unpredictable behavior.

6 Variables & Functions

The **environment** is a collection of information about your computer that every program receives. Each entry in the environment consists of a variable name and a string value. **The value string cannot be empty.** In fact, unsophisticated command processors remove the definition of a variable by setting its value to an empty string. **JP software** command processors allow you to use the `UNSET` command.

Usage

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, e.g., `%comspec`. If the name of the variable whose value you want to use is an expression, you can enclose the expression in brackets, e.g., `%[%n]`.

You can create, alter, view, and delete environment variables with the [SET](#) ^[28], [ESET](#) ^[20], and [UNSET](#) ^[31] commands, and in Take Command, with the [Environment popup window](#) ^[74] available from the Utilities menu.

Some environment variables have special meanings in the command processor: They are listed in [System Variables](#) ^[33].

The command processor also supports two special types of variables:

- ▶ [Internal variables](#) ^[34] are similar to environment variables, but are stored internally within the command processor, and are not visible in the environment. They provide information about your system for use in batch files and aliases.
- ▶ [Variable functions](#) ^[35] are referenced like environment variables, but perform additional functions like file handling, string manipulation and arithmetic calculations. In addition to the internal Variable Functions, you can use the [FUNCTION](#) ^[21] command to create your own user-defined functions.

Note: The command processor inherits its initial environment from the process which started it. That process might be **explorer.exe** or any other existing Windows process from which the current command processor session is launched. Note that if the starting process's environment is changed (through registry modifications, for example) while the command processor is already running, those changes will **not** be automatically reflected in the command processor's current environment. See the [SET](#) ^[28] command for details.

The [SET](#) ^[28] command is used to create, edit, or delete environment variables. 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:

```
del *.bak;*.bk!;*.bk
```

The variable names you use this way 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 8,191 characters.

In addition, the command processor uses the environment to keep track of the default directory on each drive. DOS keeps track of the default directory for each drive letter internally; Windows does not. The command processor overcomes this incompatibility 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 the command processor.

In 4NT and Take Command, the size of the environment is set automatically and expanded as necessary. You do not need to specify the size as you do under 4DOS or COMMAND.COM.

The trailing percent sign that was traditionally required for environment variable names is not usually required in the command processor, which accepts any character that cannot be part of a variable

name 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:

<u>original</u>	<u>expanded</u>	<u>method</u>
%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 [Argument Quoting](#)^[332] for details.

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
```

For compatibility with some peculiar syntax introduced in recent CMD.EXE versions, 4NT and TC now support:

%var:string1=string2%	Substitutes the second string for all instances of the first string in the variable.
%var:~x[,y]%	Returns the substring starting at the xth character position (base 0) and continuing for y characters. If y is not specified, returns the remainder of the string. If x is negative, starts from the end of the string.

For string manipulations, we suggest you rely instead on the much more flexible [Variable Functions](#)^[359].

Operating System Dependent Environment Variable Categories

SUBTOPIC UNDER CONSTRUCTION.

Windows 98 and ME allow definition of variables before the command processor starts, using **CONFIG.SYS** and **AUTOEXEC.BAT**. These variables are inherited by the command processor,

Windows NT, 2000, XP and 2003 allow definition of variables in different parts of the registry. These variables form part of the environment received by any program started directly by the OS. **MORE INFORMATION AT A LATTER DATE.**

6.1 System Variables

The variables below have special meaning for the command processor.

[CDPATH](#)^[339]
[CMDLINE](#)^[339]
[COLORDIR](#)^[339]
[COMSPEC](#)^[340]
[FILECOMPLETION](#)^[340]
[PATH](#)^[341]
[PATHEXT](#)^[341]
[PROMPT](#)^[341]
[TEMP](#)^[342]
[TITLEPROMPT](#)^[342]
[TMP](#)^[342]
[TREEEXCLUDE](#)^[342]

The variables below are used by some Microsoft command processors, but are ignored by JP software command processors. To see their usage by Microsoft and the alternate methods to achieve the same purpose in the command processor, review:

[COPYCMD](#)^[340]
[DIRCMD](#)^[340]

6.1.1 CDPATH (variable)

CDPATH tells the command processor where to search for directories specified by the [CD](#)^[159], [CDD](#)^[160], and [PUSHD](#)^[264] commands and in [automatic directory changes](#)^[22].

6.1.2 CMDLINE

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, it will not be put in CMDLINE, and any previous CMDLINE variable will be removed from the environment.

6.1.3 COLORDIR (variable)

COLORDIR controls directory display colors used by DIR. See [Color-Coded Directories](#)^[179] for a complete description of the format of this variable.

6.1.4 COMSPEC

COMSPEC contains the full path and name of the character-mode command processor, e.g. "c:\program files\jpsoft\4nt.exe".

4NT automatically sets COMSPEC to point to itself on startup. If this is not what you want, you can specifically SET COMSPEC to the program of your choice in 4START.BTM or some similarly convenient location. Note that some but not all programs rely on the contents of COMSPEC to locate the current character-mode command processor.

6.1.5 COPYCMD

A **COPYCMD** variable is used by some versions of *CMD.EXE* to hold default options for the [COPY](#)^[164] and [MOVE](#)^[246] commands. The command processor does not directly support this variable to which it attaches no special meaning, 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 or create a COPYCMD variable for compatibility with systems that do not use our command processors, you can define the alias to append the contents of that variable to the COPY command:

```
[c:\] set COPYCMD=/r
...
[c:\] alias COPY=*copy %copycmd`
```

and COPYCMD will be expanded to its current value at the time the alias is executed.

6.1.6 DIRCMD

A **DIRCMD** variable is used by some versions of *CMD.EXE* to hold default options for the DIR command. The command processor does not directly 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 or create a DIRCMD variable for compatibility with systems that do not use our command processors, you can define the alias to append the contents of that variable to the DIR command:

```
[c:\] set DIRCMD=/2 /p /v`
...
[c:\] alias DIR=*dir %dircmd`
```

and DIRCMD will be expanded to its current value at the time the alias is executed.

6.1.7 FILECOMPLETION (variable)

FILECOMPLETION sets the files made available during filename completion for selected commands. See [Customizing Filename Completion](#)^[19] for a complete description of the format of this variable.

6.1.8 HISTORYEXCLUDE

HistoryExclude specifies which commands should be excluded from the [History List](#)^[13]. The syntax is:

```
HistoryExclude=cmd1;cmd2;cmd3;...
```

For example, to exclude DEL, FREE, Notepad and user-defined alias MYDIR:

```
HistoryExclude=del;free;c:\windows\system32\notepad.exe;mydir
```

Also see [HISTORY](#)^[226].

6.1.9 PATH (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 the [PATH](#)^[254] command for a full description of this variable which can also be changes or modified with [SET](#)^[281] and [ESET](#)^[201].

Note: We strongly recommend that you always leave at least your "WINDOWS" and "SYSTEM(32)" directories in the PATH. The directory where 4NT or Take Command resides needs not be included in the PATH.

6.1.10 PATHEXT (variable)

PATHEXT can be used to select the extensions to look for when searching the [PATH](#)^[341] for an executable file. It consists of a list of extensions, separated by semicolons. For example, to replicate the default extension list used by the command processor:

```
set pathtext=.pif;.com;.exe;.btm;.bat;.cmd;.rex;.rexx
```

PATHEXT is ignored unless the [PathExt](#)^[108] setting is set to Yes in the [.INI file](#)^[89]. Once PATHEXT is enabled the standard path search for *.PIF*, *.COM*, *.EXE*, *.BTM*, *.BAT*, *.CMD*, *.REX*, and *.REXX* files is replaced by a search for files with the extensions listed in PATHEXT, in the order listed there.

Enabling PATHEXT affects **only** the standard path search, it does not affect the subsequent searches for files with [executable extensions](#)^[48]. 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](#)^[254] command.

Caution: If you set **PathExt=Yes** in the [.INI file](#)^[89], and then fail to set the PATHEXT variable, path searches will fail as there will be no extensions for which to search!

6.1.11 PROMPT (variable)

PROMPT defines the command-line prompt. It can be set or changed with the [PROMPT](#)^[262], [SET](#)^[281] and [ESET](#)^[201] commands. See the [PROMPT](#)^[262] command for details.

Also see: [TITLEPROMPT](#)^[342].

6.1.12 RECYCLEEXCLUDE

RECYCLEEXCLUDE can be used to exclude specified files from the Recycle Bin

The syntax is:

```
RecycleExclude=file1;file2;file3;...
```

For example, to exclude *.lib, *.obj, and *.bak files:

```
RecycleExclude=*.lib;*.obj;*.bak
```

Also see the [DEL/ERASE](#)^[172] command and the [RecycleBin](#)^[109] directive.

6.1.13 TEMP

TEMP specifies the directory where the command processor should store temporary files. If the TMP variable exists, the command processor will use its value. Some other programs also use TEMP to define where they should place their temporary files.

6.1.14 TITLEPROMPT

TITLEPROMPT can be used to specify the contents of the title bar under Windows. Modifying its value changes the displayed title immediately. Unsetting it does NOT affect the title. It may contain the special escape-sequences acceptable in [PROMPT](#)^[262], and all internal variables and functions can be used to generate it.

See also: [PROMPT](#)^[262], [ACTIVATE](#)^[136] and [WINDOW](#)^[316].

6.1.15 TMP

TMP is used by the command processor instead of **TEMP** if it is defined. Some other programs also use **TMP** to define where they should place their temporary files.

6.1.16 TREEEXCLUDE

TreeExclude specifies which drives and directories to ignore when updating the JPSTREE.IDX file. The syntax is:

```
TreeExclude=dir1;dir2;dir3;...
```

Any specified drive/directory and all of its subdirectories will be excluded from JPSTREE.IDX update. For example, to exclude everything in c:\windows, d:\temp\temp2, and everything on drive g:

```
TreeExclude=c:\windows;d:\temp\temp2;g:\
```

Also see: [Extended Directory Searches](#)^[56], [CDD](#)^[160].

6.2 Internal Variables

Internal variables are special variables built into the command processor to provide information about your system. They are not stored in the environment, but can be accessed as if they were environment variables in interactive commands, aliases, and batch files.

The values of these variables are stored internally in the command processor, and cannot be changed with the [SET](#)^[281], [UNSET](#)^[312], or [ESET](#)^[201] command. However, you can override any of these variables by defining a new environment variable with the same name. The internal variable can be made available again by unsetting the identically name environment variable. The names of ALL internal variables (except the pseudovariables errorlevel, ?, ??, +, and =) begin with an underscore

character to make it easier to distinguish them and to avoid accidentally overriding them.

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
```

Some internal 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.

Variables which return a file or directory name from a volume that supports long filenames return it in the same case as it is stored. Returned names are not quoted automatically, you must add the quotes yourself if they are required by the syntax in which you use them.

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](#)^[348] will not return accurate results if your operating system and Advanced Power Management drivers do not provide correct information on battery status to the command processor.

Note: CMD.EXE has several internal variable names which have no special meaning under 4NT or Take Command: CD, CMDCMDLINE, CMDEXTVERSION, DATE, RANDOM, TIME. However, for the sole purpose of CMD.EXE "emulation", the [IF DEFINED](#)^[227] test of these variables will always be true.

For a list of internal variables organized by general categories of use, see [Internal Variables by Category](#)^[343].

Examples

You can use internal variables in a wide variety of ways depending on your needs. Here are just a couple of examples. For a more comprehensive set of examples see the *EXAMPLES.BTM* file which came with your command processor.

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](#)^[228] 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
```

6.2.1 List by Category

► [Command processor status](#)^[343]

- ▶ [Compatibility](#)^[343]
- ▶ [Dates and times](#)^[343]
- ▶ [Drives and directories](#)^[343]
- ▶ [Error codes](#)^[343]
- ▶ [Hardware status](#)^[343]
- ▶ [Operating system and software status](#)^[343]
- ▶ [Screen, color, and cursor](#)^[343]

The list below gives a one-line description of all [Internal Variables](#)^[342] and a cross-reference which selects a separate help topic on that variable. Many variables are simple enough that the one-line description is probably sufficient, but in most cases you should check for any additional information in the cross-referenced explanation if you are not already familiar with a variable. You can also obtain help on any function with a "**HELP** *variablename*" command at the prompt. See the [HELP](#)^[225] command for details

Hardware status

acstatus ^[347]	AC line status
alt ^[348]	Alt key depressed
battery ^[348]	Battery status
batterylife ^[348]	Remaining battery life, seconds
batterypercent ^[348]	Remaining battery life, %
capslock ^[349]	CapsLock on
cpu ^[350]	CPU type
cpuusage ^[350]	CPU time usage (percent)
ctrl ^[350]	Ctrl key depressed
kbhit ^[350]	Returns 1 if and only if a keyboard input character is waiting
lalt ^[354]	left Alt key depressed
lctrl ^[354]	left Ctrl key depressed
lshift ^[354]	left Shift key depressed
numlock ^[354]	NumLock on
ralt ^[355]	right Alt key depressed
rctrl ^[355]	right Ctrl key depressed
rshift ^[355]	right Shift key depressed
scrolllock ^[355]	ScrollLock on
shift ^[355]	Shift key depressed

Operating system and software status

ansi ^[348]	ANSI X3.64 status
boot ^[349]	Boot drive letter, without a colon
codepage ^[350]	Current code page number
country ^[350]	Current country code
dos ^[351]	Operating system type
dosver ^[352]	Operating system version
host ^[353]	Host name of local computer.
hwprofile ^[353]	Windows hardware profile <i>if defined</i>
ip ^[354]	IP address(es) of local computer.
windir ^[356]	Windows directory pathname
winfgwindow ^[356]	Title of foreground window.
winname ^[356]	Name of local computer
winsysdir ^[356]	Windows system directory pathname
winticks ^[356]	Milliseconds since Windows was started
wintitle ^[356]	Current window title
winuser ^[356]	Name of current user.

winver^[357] Windows version number

Command processor status

4ver^[347] Command processor version
batch^[348] Batch nesting level
batchline^[348] Line number in current batch file.
batchname^[348] Full path and filename of current batch file.
batchtype^[348] Type of the current batch file
build^[349] Build number
childpid^[349] Process ID of most recent child process
cmdline^[349] Current command line
cmdproc^[349] Command processor name
cmdsproc^[349] Full pathname of command processor
detachpid^[351] Process ID of most recent detached process
dname^[351] Name of the description file.
echo^[352] Echo status
hlogfile^[353] Current history log file name
ininame^[353] Full pathname of the current INI file
logfile^[354] Current log file name
pid^[354] the command processor process ID (numeric)
pipe^[354] **1** if current process is running in a pipe; **0** otherwise
ppid^[355] Process ID of parent process
shell^[355] Shell level
shralias^[355] **1** if SHRALIAS is loaded
startpath^[356] Startup directory of current shell.
startpid^[356] Process ID of most recent STARTed process.
transient^[356] **1** if current process is a transient shell, **0** otherwise
unicode^[356] **1** if shell uses unicode for redirected output, **0** otherwise

Screen, color, and cursor

bq^[349] Background color at cursor position
ci^[349] Current text cursor shape in insert mode
co^[349] Current text cursor shape in overstrike mode
column^[350] Current cursor column
columns^[350] Virtual screen width
fg^[352] Foreground color at cursor position
row^[355] Current cursor row
rows^[355] Screen height
selected^[355] **(TC)** First line of highlighted text
xpixels^[357] Physical screen horizontal size in pixels
ypixels^[357] Physical screen vertical size in pixels

Drives and directories

cwd^[350] Current drive and directory
cwds^[351] Current drive and directory with trailing \
cwp^[351] Current directory
cwps^[351] Current directory with trailing \
disk^[351] Current drive
lastdisk^[354] Last valid drive

Dates and times

date^[351] Current date
datetime^[351] Current date and time, yyyyMMddhhmmss
day^[351] Current day of the month

<u>dow</u> ^[352]	Current day of the week, English, short
<u>dowf</u> ^[352]	Current day of the week, English, full
<u>dowi</u> ^[352]	Current day of the week as an integer
<u>day</u> ^[352]	Current day of the year
<u>hour</u> ^[353]	Current hour
<u>idow</u> ^[353]	Current day of the week, local language, short
<u>idowf</u> ^[353]	Current day of the week, local language, full
<u>imonth</u> ^[353]	Current month name, local language, short
<u>imonthf</u> ^[353]	Current month name, local language, full
<u>isodate</u> ^[354]	Current date in ISO 9601 format
<u>minute</u> ^[354]	Current minute
<u>month</u> ^[354]	Current month of the year as integer
<u>monthf</u> ^[354]	Current month of the year, English, full
<u>second</u> ^[355]	Current second
<u>time</u> ^[356]	Current time
<u>year</u> ^[357]	Current year

Error codes

<u>?</u> ^[346]	Exit code, last external program
<u>?</u> ^[346]	Exit code, last internal command
<u>errorlevel</u> ^[357]	Exit code, last external program
<u>ftperror</u> ^[352]	Last FTP error code
<u>syserr</u> ^[356]	Latest Windows error code

Compatibility

<u>=</u> ^[347]	Substitutes command processor escape character
<u>+</u> ^[347]	Substitutes command separator

6.2.2 ? (variable)

? is the exit code of the last **external** command. Many programs return 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.*

WARNING: In imitation of Microsoft's **CMD.EXE** some **internal** commands, e.g., **DIR**, also set this variable to the same value they set the variable ?^[346], which destroys the code from the last external command. To insure that you use the exit code from the external command you want to check, save the value in a variable immediately on command completion, and use the saved variable instead. We also strongly recommend that for **internal** commands you query the proper ?^[346] variable instead.

Alternate name: ERRORLEVEL^[357].

See also: ?^[346]

6.2.3 _?

_? contains the exit code of the last **internal** command. *You must use or save this value immediately, because it is set by every internal command.*

Result codes:

- 0** command was successful
- 1** a usage error occurred
- 2** another command processor error or an operating system error occurred

3 the command was interrupted by Ctrl-C or Ctrl-Break

This variable can also be set in a subroutine by the [RETURN](#)^[272] command.

Note that in imitation of **CMD.EXE** some internal commands, e.g., **DIR**, also set the variables [?](#)^[346] and [ERRORLEVEL](#)^[357] to the same value they set this variable. However, you are strongly urged to use **this** variable.

See also: [?](#)^[346]

6.2.4 =

= is the current [EscapeChar](#)^[103]. Use this pseudovariable, instead of the actual escape character, if you want your batch files and aliases to work in all JPsoft command processors or in other users' environment regardless of how the escape character is defined. For example, if the escape character is a caret [**^**] (the default in 4NT / 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
```

6.2.5 +

+ is the current [command separator](#)^[100]. Use this pseudovariable, instead of the actual command separator, if you want your batch files and aliases to work in all JPsoft command processors or in other users' environment regardless of how the command separator is defined.

WARNING: %+ should always be surrounded by spaces.

For example, if the command separator is an ampersand [**&**] (the default in 4NT / 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
```

6.2.6 _4VER

_4VER is the current command processor version (for example, **6.01**). The current decimal separator is used to separate the major and minor version numbers (see [DecimalChar](#)^[102] for details). ASCII and Unicode versions - starting with Version 5.00 - of 4NT and Take Command also append an **A** and **U**, respectively, e.g., **5.00U** for the first Unicode version.

See also: [BUILD](#)^[349].

6.2.7 _ACSTATUS

_ACSTATUS is the AC line status.

0	Offline
1	Online
unknown	Unknown

6.2.8 `_ALT`

`_ALT` is 1 if either Alt key is depressed, otherwise it is 0.

6.2.9 `_ANSI`

`_ANSI` is 1 if internal support for [ANSI Std. X3.64](#)^[64] is enabled, 0 if not.

SETDOS /A	ANSI Directive	<code>_ANSI</code>
0 (default)	Auto (default)	Result of test
1	Yes	1
2	No	0

6.2.10 `_BATCH`

`_BATCH` is the current batch file nesting level. It is 0 if no batch file is currently being processed.

6.2.11 `_BATCHLINE`

`_BATCHLINE` is the current line number in the current batch file. It is -1 if no batch file is currently being processed.

6.2.12 `_BATCHNAME`

`_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.

6.2.13 `_BATCHTYPE`

`_BATCHTYPE` is 0 if the current batch file is a normal (neither compressed nor encrypted) file, 1 if compressed, 2 if encrypted. It is -1 if no batch file is currently being processed.

6.2.14 `_BATTERY`

`_BATTERY` is the battery charge status:

1	High
2	Low
4	Critical
8	Charging
128	No battery
unknown	Unknown

6.2.15 `_BATTERYLIFE`

`_BATTERYLIFE` is either the number of seconds of battery life remaining, or `unknown`.

6.2.16 `_BATTERYPERCENT`

`_BATTERYPERCENT` is the percentage of battery charge remaining (0...100), or `unknown`.

6.2.17 **_BG**

_BG is a string containing the first three characters of the current background screen output color, for example, **B1a**. See [Colors, Color Names and Codes](#)^[461] for details.

6.2.18 **_BOOT**

_BOOT is the boot drive letter, without a colon.

6.2.19 **_BUILD**

_BUILD is the internal 4NT or Take Command build number. That number is essentially an arbitrary integer identifier and is reported solely for informational purposes.

See also: [_4VER](#)^[347].

6.2.20 **_CAPSLOCK**

_CAPSLOCK is 1 if the CapsLock key is toggled ON, and 0 otherwise.

6.2.21 **_CHILDPID**

_CHILDPID is the process ID of the most recent child process.

6.2.22 **_CI**

_CI is the insert mode cursor shape, as a percentage. See [SETDOS /S](#)^[283] and [CursorIns](#)^[101].

6.2.23 **_CMDLINE**

_CMDLINE is the current command line. This is most useful in key aliases. If you specify it on the command line, it returns the contents of the command line with the **%_cmdline** name removed.

6.2.24 **_CMDPROC**

_CMDPROC is the name of the current command processor, either **4NT** or **TCMD32**. 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 command processor. I

See also: [_4VER](#)^[347], [_BUILD](#)^[349], [_CMDSPEC](#)^[349], [_DOS](#)^[351], [_DOSVER](#)^[352], [_WINVER](#)^[357].

6.2.25 **_CMDSPEC**

_CMDSPEC is the full pathname of the command processor.

Example:

```
[C:\]echo %_cmdspec      <-- command
C:\JPSOFT\6U\4nt.exe    <-- response
```

6.2.26 **_CO**

_co is the overstrike mode cursor shape, as a percentage (see [SETDOS /S](#)^[283] and [CursorOver](#)^[101]).

6.2.27 `_CODEPAGE`

`_CODEPAGE` is the current code page number.

See also: [CHCP](#) [162].

6.2.28 `_COLUMN`

`_COLUMN` is the current cursor column. The leftmost column is numbered 0.

See also: [COLUMNS](#) [350], [ROW](#) [355], [ROWS](#) [355].

6.2.29 `_COLUMNS`

`_COLUMNS` is the current number of virtual screen columns, for example, 80. **(TC)** See [Resizing the Take Command Window](#) [77] for additional details on the virtual screen width.

See also: [COLUMN](#) [350], [ROW](#) [355], [ROWS](#) [355].

6.2.30 `_COUNTRY`

`_COUNTRY` is the current country code as reported by the operating system. This code is usually the same as the international dialing code for the country.

6.2.31 `_CPU`

`_CPU` is the CPU type:

```
486  i486
586  Pentium family
etc.
```

This variable merely queries Windows for the processor type. Compatible AMD or other processors will generally return the value corresponding to the Intel processor they most closely resemble.

To determine the type, revision, stepping level, and other such details for advanced processors use the [@WININFO](#) [411] function.

6.2.32 `_CPUUSAGE`

`_CPUUSAGE` indicates how busy the CPU is, in percent, 0 to 100, as reported by the operating system..

6.2.33 `_CTRL`

`_CTRL` is 1 if either Ctrl key is depressed, and 0 otherwise.

6.2.34 `_CWD`

`_CWD` is the current working directory in the format `d:\pathname`. If the current working directory is a root directory, the format is `d:\`.

See also: [CWDS](#) [351], [CWP](#) [351], [CWPS](#) [351], [@CWD](#) [368], [@CWDS](#) [369].

6.2.35 **_CWDS**

_CWDS is the current working directory in the format *d:\pathname*.

See also: [_CWD](#)^[350], [_CWP](#)^[351], [_CWPS](#)^[351], [@CWD](#)^[368], [@CWDS](#)^[369].

6.2.36 **_CWP**

_CWP is the current working directory in the format *\pathname*, without the drive.

See also: [_CWD](#)^[350], [_CWDS](#)^[351], [_CWPS](#)^[351], [@CWD](#)^[368], [@CWDS](#)^[369].

6.2.37 **_CWPS**

_CWPS is the current working directory in the format *\pathname*, without the drive.

See also: [_CWD](#)^[350], [_CWDS](#)^[351], [_CWP](#)^[351], [@CWD](#)^[368], [@CWDS](#)^[369].

6.2.38 **_DATE**

_DATE is the current system date, in the format determined by your country settings. The year will be in two-digit format for compatibility unless your country setting is *yyyy-mm-dd*.

See also: [_ISODATE](#)^[354].

6.2.39 **_DATETIME**

_DATETIME is the current date and time as 14-characters in the format *yyyyMMddhhmmss*. The date part is the same as [_isodate](#)^[354] without separators.

6.2.40 **_DAY**

_DAY is the current day of the month (1 to 31).

6.2.41 **_DETACHPID**

_DETACHPID is the process ID of the most recent process launched by the [DETACH](#)^[178] command.

6.2.42 **_DISK**

_DISK is the current disk drive letter, without a colon (for example, "C").

If the current directory is a UNC rather than a mapped drive, %_disk will return the sharename.

6.2.43 **_DNAME**

_DNAME is the name of the file used to store file descriptions. It can be changed with the [DescriptionName](#)^[102] directive in the [.INI file](#)^[89], or the [SETDOS /D](#)^[283] command.

6.2.44 **_DOS**

_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:

	4NT	Take Command
Windows 98	WIN98C	WIN98
Windows ME	WINMEC	WINME
Windows NT4	NT	WIN32
Windows 2000	WIN2K	WIN32
Windows XP	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. If you want the current command processor name, use [_CMDPROC](#)^[349]. To differentiate between different versions of Windows within 4NT / Take Command, use the [_WINVER](#)^[357] variable; to differentiate between different command processors, use the [_CMDPROC](#)^[349] variable.

6.2.45 [_DOSVER](#)

[_DOSVER](#) is the current operating system version. The current decimal character is used to separate the major and minor version numbers (see [DecimalChar](#)^[102]).

6.2.46 [_DOW](#)

[_DOW](#) is the first three characters of the current day of the week ("Mon", "Tue", "Wed", etc.).

6.2.47 [_DOWF](#)

[_DOWF](#) is the full day of the week for the current date ("Monday", "Tuesday", etc.).

6.2.48 [_DOWI](#)

[_DOWI](#) is the current day of the week as an integer (1 = Sunday, 2 = Monday, etc.).

6.2.49 [_DOY](#)

[_DOY](#) is the day of the year (1 to 366).

6.2.50 [_ECHO](#)

[_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 (see the [ECHO](#)^[198] command and the [BatchEcho](#)^[100] directive). The value returned by the [_ECHO](#) variable reflects the state applicable at the time the variable is queried.

6.2.51 [_FG](#)

[_FG](#) is a string containing the first three letters of the current foreground screen output color (for example, "Whi"). See [Colors, Color Names and Codes](#)^[467] for details.

6.2.52 [_FTPERROR](#)

[_FTPERROR](#) is the error code of the last error reported by [FTP](#)^[52]. Some of the possible codes 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)

6.2.53 **_HLOGFILE**

_HLOGFILE returns the name of the current history log file (or an empty string if LOG /H is OFF). See [LOG](#) ^[242] for information on logging.

6.2.54 **_HOST**

_HOST is the host name for the local computer.

6.2.55 **_HOUR**

_HOUR is the current hour (0 - 23).

6.2.56 **_HWPROFILE**

_HWPROFILE is the name of the current Windows hardware profile.

6.2.57 **_IDOW**

_IDOW is the 3-character abbreviation for the day of the week for the current date, in the current locale language.

6.2.58 **_IDOWF**

_IDOWF is the full name for the day of the week for the current date, in the current locale language.

6.2.59 **_IMONTH**

_IMONTH is the abbreviated name for the current month, in the current locale language.

6.2.60 **_IMONTHF**

_IMONTHF is the full name for the current month, in the current locale language.

6.2.61 **_ININAME**

_ININAME returns the fully qualified pathname of the INI file used by the current shell.

6.2.62 **_IP**

_IP is the IP address of the local computer. If the computer has more than one NIC, **_IP** returns a space-delimited list of all IP addresses.

6.2.63 **_ISODATE**

_ISODATE is the current system date, in ISO 9601 format: *yyyy-mm-dd*. Also see [_DATE](#)^[351] and [_DATETIME](#)^[351]..

6.2.64 **_KBHIT**

_KBHIT returns 1 if one or more keystrokes are waiting in the keyboard buffer, or 0 if the keyboard buffer is empty.

6.2.65 **_LALT**

_LALT returns 1 if the Left Alt key is depressed.

6.2.66 **_LASTDISK**

_LASTDISK is the last valid drive letter, without a colon.

6.2.67 **_LCTRL**

_LCTRL returns 1 if the Left Ctrl key is depressed.

6.2.68 **_LOGFILE**

_LOGFILE returns the name of the current log file (or an empty string if LOG is OFF). See [_LOG](#)^[242] for information on logging.

6.2.69 **_LSHIFT**

_LSHIFT returns 1 if the Left Shift key is depressed.

6.2.70 **_MINUTE**

_MINUTE is the current minute (0 - 59).

6.2.71 **_MONTH**

_MONTH is the current numeric month of the year (1 to 12).

6.2.72 **_MONTHF**

_MONTHF is the full name of the current month ("January", "February", etc.).

6.2.73 **_NUMLOCK**

_NUMLOCK returns 1 if the NumLock key is toggled ON.

6.2.74 **_PID**

_PID is the current process ID number.

6.2.75 **_PIPE**

_PIPE is "1" if the current process is running inside a pipe, and "0" otherwise.

6.2.76 _PPID

_PPID is the process ID number of the parent process.

6.2.77 _RALT

_RALT returns 1 if the Right Alt key is depressed.

6.2.78 _RCTRL

_RCTRL returns 1 if the Right Ctrl key is depressed.

6.2.79 _ROW

_ROW is the current cursor row (for example, "0" for the top of the window).

6.2.80 _ROWS

_ROWS is the current number of screen rows (for example, "25").

6.2.81 _RSHIFT

_RSHIFT returns 1 if the Right Shift key is depressed.

6.2.82 _SCROLLLOCK

_SCROLLLOCK returns 1 if the ScrollLock key is toggled ON.

6.2.83 _SECOND

_SECOND is the current second (0 - 59).

6.2.84 _SELECTED

_SELECTED (TC) returns the first line of text highlighted in the Take Command window. If no text has been highlighted, SELECTED returns an empty string.

6.2.85 _SHELL

_SHELL is the current shell instance identifier, one for each command processor. Each time a new instance of a command processor is started, and no other instance exists, _SHELL is set to 0; otherwise it is incremented by 1, independently of how the specific shell is started. When an instance of the command processor ends, _SHELL is reset to that of the highest still-active instance. If all instances have ended, the next new instance will be 0 again.

See [Primary and Secondary Shells](#)^[461] for an explanation of those terms.

6.2.86 _SHIFT

_SHIFT returns 1 if the either Shift key is depressed.

6.2.87 _SHRALIAS

_SHRALIAS returns 1 if [SHRALIAS](#)^[290] is loaded.

6.2.88 **_STARTPATH**

_STARTPATH returns the startup directory for the current shell (not necessarily the same as the location of the executable!)

6.2.89 **_STARTPID**

_STARTPID is the process ID of the most recent process launched by the [START](#)^[291] command.

6.2.90 **_SYSERR**

_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. See the [Windows System Errors](#)^[465] table in the Reference section for examples.

6.2.91 **_TIME**

_TIME contains the current system time in the format hh:mm:ss. The separator character may vary depending upon your country information.

6.2.92 **_TRANSIENT**

_TRANSIENT is "1" if the current shell is transient (started with a **/C**, see [Command Line Options](#)^[4] for details), or "0" otherwise.

6.2.93 **_UNICODE**

_UNICODE returns **1** if the shell is using Unicode for redirected output, **0** otherwise.

6.2.94 **_WINDIR**

_WINDIR returns the pathname of the Windows directory.

6.2.95 **_WINFGWINDOW**

_WINFGWINDOW returns the title of the foreground window.

6.2.96 **_WINNAME**

_WINNAME returns the computer name of the current system.

6.2.97 **_WINSYSDIR**

_WINSYSDIR returns the pathname of the Windows system directory.

6.2.98 **_WINTICKS**

_WINTICKS returns the number of milliseconds since Windows was started.

6.2.99 **_WINTITLE**

_WINTITLE returns the title of the current window.

6.2.100 **_WINUSER**

_WINUSER returns the name of the user currently logged on.

6.2.101 _WINVER

_WINVER returns the current Windows version number. The current decimal character is used to separate the major and minor version numbers (see [DecimalChar](#)^[102] for details).

6.2.102 _XPIXELS

_XPIXELS returns the physical screen horizontal size in pixels.

6.2.103 _YEAR

_YEAR is the current year (1980 to 2099).

6.2.104 _YPIXELS

_YPIXELS returns the physical screen vertical size in pixels.

6.2.105 ERRORLEVEL

ERRORLEVEL is an **alternate name** for the [?](#)^[346] variable, included only for compatibility with Microsoft, and is the exit code of the last **external** command. Many programs return 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 **ERRORLEVEL** is undefined..*

WARNING: In imitation of Microsoft's **CMD.EXE** some **internal** commands, e.g., **DIR**, also set this variable to the same value they set the variable [?](#)^[346], which destroys the code from the last external command. To insure that you use the exit code from the external command you want to check, save the value in a variable immediately on command completion, and use the saved variable instead. We also strongly recommend that for **internal** commands you query the proper [?](#)^[346] variable instead.

Alternate name: [?](#)^[346].

See also [?](#)^[346]

6.3 Variable Functions

Variable functions are very similar to 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), and no space is allowed between the function name and the "[]". All arguments of a variable function taken together as a group may not exceed 8,191 characters.

The variable functions are useful at the command prompt as well as in [aliases](#)^[318] and [batch files](#)^[327] to check on available system resources, manipulate strings and numbers, and work with files and filenames.

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](#)^[403] will not return accurate results if your operating system does not provide correct disk drive status information to the command processor.

The variable functions built into the command processor are listed in alphabetical order in subsequent topics. You can also obtain help from the command prompt on any function with a **HELP** **@functionname** or **HELP f_functionname** command, or by pressing [Ctrl-F1](#)^[126] when the cursor is on the function name. See the [HELP](#)^[225] command for details

Note: The [FUNCTION](#)^[218] command can be used to create, edit, or display user-defined variable functions, and the [UNFUNCTION](#)^[317] to delete them.

For a list of Variable Functions organized by general categories of use, see [Variable Functions by Category](#)^[359].

Memory Size / Disk Space / File Size Units and Report Format

Some variable functions, such as [@DISKFREE](#)^[377], accept an optional argument: **scale code**. These functions return a size of a disk or of an entity on the disk as a multiple of the specified scale factor from the table below. Lower case letters denote a power of 1,000, upper case letters a power of 1,024.

Code	Scale Factor		Code	Scale Factor		Unit Name
k	1,000	10**3	K	1,024	2**10	kilobyte
m	1,000,000	10**6	M	1,048,576	2**20	megabyte
g	1,000,000,000	10**9	G	1,073,741,824	2**30	gigabyte
t	1,000,000,000,000	10**12	T	1,099,511,627,776	2**40	terabyte

You can include **commas** (or the "thousands separator" character for your system) in the value from a function by appending the letter **c** to the scale code. For example, to add commas to a **b** (number of bytes) result, enter **bc** as the argument, e.g. "echo %@DISKFREE[C,bc]". To set the thousands separator see the [ThousandsChar](#)^[117] directive.

Notes

1) Disk manufacturers use the prefixes adopted from the metric system (kilo, mega, giga, tera) in their original, standard meaning (powers of 1,000), while memory manufacturers and Microsoft use the slightly larger powers of 1,024 (2**10).

2) The **scale code** is one of the few instances in which JP Software command processors are **case sensitive**.

Date Argument Format: See the [Date Formats](#)^[363] topic.

File Name Arguments

Filenames passed as variable function arguments must be in quotes if they contain white space 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.

Drive Letter Arguments

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.

Functions Accessing File Handles

The [@FILEREAD](#)^[382], [@FILEWRITE](#)^[385], [@FILEWRITEB](#)^[385], [@FILESEEK](#)^[383], [@FILESEEKL](#)^[383], and [@FILECLOSE](#)^[386] functions allow you to access files based on their *file handle*. These functions must be used only with file handles returned by [@FILEOPEN](#)^[381], unless otherwise noted under the individual functions. If you use them with any other file handle you may damage files.

File Attributes

Several functions accept a file attribute string to help determine which files to process. The rules for constructing the attribute string are the same as the ones for [Attribute Switches](#)^[39] in commands.

Examples

You can use variable functions in a wide variety of ways depending on your needs. Here are a couple of examples to give you an idea of what's possible. For a much more comprehensive set of examples, see the file *EXAMPLES.BTM*, which comes with the command processor.

The command below sets the prompt to show the amount of free memory (see [PROMPT](#)^[262] 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 such as CALC 3 * (4 + 5):

```
[c:\] alias calc `echo The answer is: %@eval[%$]`
```

(assuming a [ParameterChar](#)^[107] value of "\$").

6.3.1 List by Category

This list gives a one-line description of all built-in [Variable Functions](#)^[357], and a cross-reference which selects a separate help topic on that function where you will find the detailed syntax and description. You can also obtain help on any function with a "[HELP @functionname](#)" (or "[HELP f_functionname](#)") command at the prompt or by pressing [Ctrl-F1](#)^[120] when the cursor is on the function name. See the [HELP](#)^[225] command for details

- | | |
|---|---|
| ▶ Dates and times ^[359] | ▶ Network properties ^[359] |
| ▶ Drives and devices ^[359] | ▶ Numbers and arithmetic ^[359] |
| ▶ File content ^[359] | ▶ Strings and characters ^[359] |
| ▶ File names ^[359] | ▶ System status ^[359] |
| ▶ File properties ^[359] | ▶ Utility ^[359] |
| ▶ Input dialog boxes ^[359] | |

Note: this is merely an arbitrary classification and many functions have functionality that covers several categories.

System status

<u>@CLIP</u> ^[367]	Line <i>n</i> from clipboard
<u>@CLIPW</u> ^[367]	Write string to the clipboard
<u>@CONSOLE</u> ^[368]	Identify console sessions
<u>@ERRTEXT</u> ^[374]	Windows error description
<u>@READSCR</u> ^[403]	Read characters from the screen
<u>@REGCREATE</u> ^[403]	Create registry subkey
<u>@REGDELKEY</u> ^[403]	Delete a registry key and its subkeys
<u>@REGEXIST</u> ^[404]	Test if a registry key exists
<u>@REGQUERY</u> ^[404]	Read value from registry
<u>@REGSET</u> ^[404]	Write value to registry
<u>@REGSETENV</u> ^[404]	Write value to registry and broadcast change.
<u>@WINCLASS</u> ^[411]	Title of first window with classname
<u>@WINEXENAME</u> ^[411]	Executable name for window
<u>@WININFO</u> ^[411]	Current system information
<u>@WINMEMORY</u> ^[411]	Windows memory information
<u>@WINMETRICS</u> ^[412]	Windows system metrics
<u>@WINSTATE</u> ^[414]	Current state of window
<u>@WINSYSTEM</u> ^[414]	Set/get windows system parameters

Drives and devices

<u>@CDROM</u> ^[368]	CD-ROM drive detection (0 or 1)
<u>@CWD</u> ^[368]	Current Working Directory of drive
<u>@CWDS</u> ^[368]	Current Working Directory of drive with trailing backslash
<u>@DEVICE</u> ^[370]	Character device detection
<u>@DISKFREE</u> ^[371]	Free disk space
<u>@DISKTOTAL</u> ^[371]	Total disk space
<u>@DISKUSED</u> ^[372]	Used disk space
<u>@FSTYPE</u> ^[388]	File system type (FAT, NTFS, CDFS, etc.)
<u>@LABEL</u> ^[398]	Volume label
<u>@READY</u> ^[403]	Drive ready status (0 or 1)
<u>@REMOTE</u> ^[404]	Remote (network) drive detection (0 or 1)
<u>@REMOVABLE</u> ^[405]	Removable drive detection (0 or 1)

File content

<u>@CRC32</u> ^[368]	File CRC
<u>@FILECLOSE</u> ^[380]	Close a file handle
<u>@FILEOPEN</u> ^[381]	Open a file handle
<u>@FILEREAD</u> ^[382]	Read next line from a file handle
<u>@FILESEEK</u> ^[383]	Move a file handle pointer
<u>@FILESEEKL</u> ^[383]	Move a file handle pointer to a specified line
<u>@FILEWRITE</u> ^[385]	Write next line to a file handle
<u>@FILEWRITEB</u> ^[385]	Write data to a file handle
<u>@INIREAD</u> ^[391]	Return an entry from an <i>.INI</i> file
<u>@INIWRITE</u> ^[392]	Write an entry in an <i>.INI</i> file
<u>@LINE</u> ^[398]	Read a random line from a file
<u>@LINES</u> ^[398]	Count lines in a file
<u>@MD5</u> ^[400]	Computes MD5 hash for a string or file
<u>@TRUNCATE</u> ^[408]	Find "true" name for a file
<u>@VERINFO</u> ^[409]	Executable file version information

File names

<u>@ALTNAME</u> ^[364]	Returns the short name for the file.
----------------------------------	--------------------------------------

<u>@EXPAND</u> ^[37b]	Returns all names that match filename
<u>@EXT</u> ^[37b]	File extension
<u>@FILENAME</u> ^[387]	File name and extension
<u>@FULL</u> ^[38b]	Full file name with path
<u>@LFN</u> ^[397]	Returns long name for a short filename
<u>@NAME</u> ^[407]	File name without path or extension
<u>@PATH</u> ^[402]	File path without name
<u>@SFN</u> ^[407]	Returns short name for a long filename
<u>@SEARCH</u> ^[40b]	Path search
<u>@TRUENAME</u> ^[40b]	Find "true" name for a file
<u>@UNC</u> ^[40b]	Returns the UNC name of a file
<u>@UNIQUE</u> ^[40b]	Create file with unique name

File properties

<u>@ATTRIB</u> ^[36b]	Test or return file attributes
<u>@DESCRIPT</u> ^[37b]	File description
<u>@EXETYPE</u> ^[377]	Application type
<u>@FILEAGE</u> ^[38b]	File age (date and time)
<u>@FILEDATE</u> ^[38b]	File date
<u>@FILES</u> ^[382]	Count files matching a wildcard
<u>@FILESIZE</u> ^[384]	Size of files matching a wildcard
<u>@FILETIME</u> ^[384]	File time
<u>@FINDCLOSE</u> ^[38b]	Closes the search handle.
<u>@FINDFIRST</u> ^[38b]	Find first matching file
<u>@FINDNEXT</u> ^[38b]	Find next matching file
<u>@SEARCH</u> ^[40b]	Path search
<u>@TRUENAME</u> ^[40b]	Find "true" name for a file
<u>@UNIQUE</u> ^[40b]	Create file with unique name
<u>@VERINFO</u> ^[40b]	Executable file version information
<u>@WATTRIB</u> ^[41b]	Test or return file attributes, including Windows 2000 / XP attributes

Strings and characters

<u>@ASCII</u> ^[36b]	Numeric ASCII value for a character
<u>@CAPS</u> ^[36b]	Capitalize first character of each word
<u>@CHAR</u> ^[36b]	Character value for numeric ASCII
<u>@EXECSTR</u> ^[377]	Execute a command and return the first output line
<u>@FIELD</u> ^[37b]	Extract a field from a string
<u>@FIELDS</u> ^[37b]	Count fields in a string
<u>@FORMAT</u> ^[387]	Formats datstr according to fmstr
<u>@INDEX</u> ^[397]	Offset of string2 within string1
<u>@INSERT</u> ^[392]	Inserts string1 into string2
<u>@INSTR</u> ^[393]	Extract a substring
<u>@ISALNUM</u> ^[394]	Test for alphanumeric characters
<u>@ISALPHA</u> ^[394]	Test for alphabetic characters
<u>@ISASCII</u> ^[39b]	Test for ASCII characters
<u>@ISCNTRL</u> ^[39b]	Test for control characters
<u>@ISDIGIT</u> ^[39b]	Test for decimal digits
<u>@ISPRINT</u> ^[39b]	Test for printable characters
<u>@ISPUNCT</u> ^[39b]	Test for punctuation characters
<u>@ISSPACE</u> ^[39b]	Test for white space characters
<u>@ISXDIGIT</u> ^[39b]	Test for hexadecimal digits
<u>@LEFT</u> ^[397]	Returns the left end of string
<u>@LEN</u> ^[397]	Length of a string
<u>@LOWER</u> ^[39b]	Convert string to lower case
<u>@LTRIM</u> ^[39b]	Removes specified leading characters.
<u>@MD5</u> ^[40b]	Computes MD5 hash for a string or file

<u>@REPEAT</u> ^[405]	Repeat a character
<u>@REPLACE</u> ^[405]	Replace string1 with string2 in text
<u>@RIGHT</u> ^[405]	Returns the right end of string.
<u>@RTRIM</u> ^[405]	Removes specified trailing characters.
<u>@STRIP</u> ^[407]	Strips all characters in char from string
<u>@SUBST</u> ^[408]	Substitute a string within another string
<u>@SUBSTR</u> ^[407]	Extract a substring
<u>@TRIM</u> ^[405]	Remove blanks from a string
<u>@UNICODE</u> ^[409]	Numeric UNICODE value for a character
<u>@UPPER</u> ^[409]	Convert string to upper case
<u>@WILD</u> ^[410]	Compares strings using wildcards
<u>@WORD</u> ^[416]	Extract a word from a string
<u>@WORDS</u> ^[417]	Count words in a string

Numbers and arithmetic

<u>@ABS</u> ^[364]	Absolute value of n
<u>@CEILING</u> ^[365]	Smallest integer not less than n
<u>@COMMA</u> ^[365]	Insert commas into a number
<u>@CONVERT</u> ^[368]	Convert value from input base to output base
<u>@DEC</u> ^[370]	Decrement a numeric value by 1
<u>@DECIMAL</u> ^[370]	Decimal portion of a number
<u>@DIGITS</u> ^[370]	Tests if string is all digits
<u>@EVAL</u> ^[374]	Arithmetic calculations
<u>@FORMATN</u> ^[388]	Format a numeric value
<u>@FLOOR</u> ^[387]	Largest integer not larger than n
<u>@INC</u> ^[390]	Increment a numeric value by 1
<u>@INT</u> ^[393]	Integer part of a number
<u>@MAX</u> ^[400]	Largest integer in the list
<u>@MIN</u> ^[400]	Smallest integer in the list
<u>@NUMERIC</u> ^[401]	Test if a string is numeric
<u>@RANDOM</u> ^[402]	Generate a random integer

Dates and times

<u>@AGEDATE</u> ^[364]	Converts a <u>filetime</u> ^[384] into date and time
<u>@DAY</u> ^[369]	Returns day of month for date
<u>@DATE</u> ^[369]	Convert date to number of days
<u>@DOW</u> ^[372]	Returns day of week for date
<u>@DOWF</u> ^[372]	Full name of day of week
<u>@DOWI</u> ^[373]	Returns day of week as integer
<u>@DOY</u> ^[373]	Returns day of year for date
<u>@IDOW</u> ^[390]	Returns localized day of week for date
<u>@IDOWF</u> ^[390]	Full localized name of day of week for date
<u>@MAKEAGE</u> ^[399]	Convert date and time to file timestamp format
<u>@MAKEDATE</u> ^[399]	Convert number of days to date
<u>@MAKETIME</u> ^[400]	Convert number of seconds to time
<u>@MONTH</u> ^[400]	Returns month for date
<u>@TIME</u> ^[408]	Convert time to number of seconds
<u>@YEAR</u> ^[417]	Returns year for date

Input dialog boxes

<u>@GETDIR</u> ^[389]	Prompt for a directory name.
<u>@GETFILE</u> ^[389]	Prompt for a path and file name.
<u>@GETFOLDER</u> ^[389]	Folder name from tree view.

Network properties

@DOMAIN ^[372]	Domain name of a computer
@ENUMSERVERS ^[373]	Identify server names on a network
@ENUMSHARES ^[373]	Identify sharenames on a server
@IPADDRESS ^[394]	Returns the numeric IP for a host name
@IPNAME ^[394]	Returns the host name for a numeric IP
@PING ^[402]	Response time from a host
@WORKGROUP ^[417]	Workgroup name of a computer

Utility

@ALIAS ^[364]	Value of an alias
@CLIP ^[367]	Line <i>n</i> from clipboard
@CLIPW ^[367]	Write string to the clipboard
@COLOR ^[367]	RGB value of a color
@DIRSTACK ^[377]	Display directory stack entry
@ERRTEXT ^[374]	Windows error description
@EXEC ^[376]	Exit code of executing a command
@EXECSTR ^[377]	Output line of executing a command
@FUNCTION ^[388]	Definition of a function
@HISTORY ^[390]	A line or word from the command history
@IF ^[390]	Evaluates a conditional expression
@OPTION ^[407]	Current <i>.INI</i> file directive value
@READSCR ^[403]	Read characters from the screen
@REXX ^[405]	Value of executing an expression by REXX ^[334]
@SELECT ^[407]	Menu selection
@TIMER ^[408]	Get split time from timer.

6.3.2 Date Formats

Functions which accept a date as an argument use the date format and separators mandated by your country code (see [@COUNTRY](#) ^[350]), for example *dd.mm.yy* in Germany, *yy-mm-dd* in Japan, and *mm/dd/yy* in the USA. The year can be entered as a 4-digit or 2-digit value. Two-digit years from 80 to 99 are interpreted as 1980...1999; values from 0 to 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 ISO 9601 international format *yyyy-mm-dd*.

Some functions which **return** a date accept an **optional code** to specify the desired format of the date value:

Code	Date Format	Description
0 or none	variable	system default
1	mm/dd/yy	USA
2	dd/mm/yy	European
3	yy/mm/dd	Japanese
4	yyyy-mm-dd	ISO 9601

6.3.3 @ABS

@ABS[*n*]: Returns the absolute value of the number *n*.

Examples:

```
echo %@abs[-1]
echo %@abs[123]
```

6.3.4 @AGEDATE

@AGEDATE[*n*,*d*]: Converts *n*, a **FILETIME** numeric value, into a date and time pair, formatted according to the current country settings, or as explicitly specified by *d*. *n* is a 64-bit integer that represents a 64-bit integer that represents the time elapsed since 1601-01-01@00:00:00 (local time) with 100-ns resolution. The optional second argument *d* specifies the date format:

<u>Code</u>	<u>Date format</u>
0	system default
1	USA (mm/dd/yy)
2	European (dd/mm/yy)
3	Japan (yy/mm/dd)
4	ISO 9601 (yyyy-mm-dd)

Example:

See also: [Time Stamps](#)^[443], [@FILEAGE](#)^[380] and [@MAKEAGE](#)^[399].

6.3.5 @ALIAS

@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](#)^[283] /X. 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.

Examples:

```
alias foo=d:\path\myprog.exe -options
echo %@alias[foo]
```

6.3.6 @ALTNAME

@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*.

Examples:

```
echo %@altname["Long Name.exe"]
echo %@altname["C:\Program Files\Microsoft Office"]
echo %@altname["%CommonProgramFiles"]
```

6.3.7 @ASCII

@ASCII[c]: Returns the ASCII character code of the specified character(s) **c** as a numeric string. For example, "**@ascii[A]**" returns "65". If you enter more than one character, @ASCII returns the numeric value for each character in a space delimited string, e.g. "**%@ascii[abc]**" returns "97 98 99". You can put an [EscapeChar](#)^[103] before the actual character to allow quotes and other special characters in the argument (e.g., "**%@ASCII[^`]**").

Examples:

```
echo %@ascii[a]
echo %@ascii[A]
echo %@ascii[%=`]
```

Also see [ASCII, Key Codes, and ANSI Commands](#)^[445], and [ASCII & Unicode Versions](#)^[422]

Note: For Unicode products, the [@UNICODE](#)^[409] function will generally return more useful values.

6.3.8 @ATTRIB

@ATTRIB[filename[, -rhsadeciopt[, p]]]: If two or more arguments are specified, it returns a "1" if the specified file has the matching attribute(s); otherwise returns a "0". The basic attributes (FAT volumes) are:

- N** Normal (no attributes set)
- R** Read-only
- A** Archive
- H** Hidden
- S** System
- D** Directory

In addition, NTFS volumes allow display of the following extended attributes:

- E** Encrypted
- C** Compressed
- I** Not content-indexed
- J** Junction (reparse point)
- N** Normal
- O** Offline
- P** Sparse file
- T** Temporary

The extended attributes are displayed when @ATTRIB is invoked with a single argument, but they are suppressed when used for file selection (two or more arguments). To select files based on the extended attributes, see [@WATTRIB](#)^[410].

Also see [Attributes Switches](#)^[39].

The attributes (other than **N**) can be combined (for example **%@ATTRIB[MYFILE,HS]**). Normally @ATTRIB will only return a "1" if all of the attributes match. However, if a final **,p** is included (for **p**artial match), then @ATTRIB will return a "1" if any of the attributes match. For example, **%@ATTRIB[MYFILE,HS,p]** will return a "1" if **MYFILE** has the hidden, system, or both attributes. Without **,p** the function will return a "1" only if **MYFILE** has both attributes.

If you do not specify any attributes, @ATTRIB returns the attributes of the specified file in the format **RHSADECINOPT**, 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 returns an empty string.

The **filename** must be in quotes if it contains white space or special characters.

Examples:

```
echo %@attrib["C:\Program Files\My Program\myfile.exe",rhs,p]
echo Attributes for myfile.exe: %@attrib[myfile.exe]
```

6.3.9 @CAPS

@CAPS["xxx",text]: Capitalizes the first letter of each word in the string (words that do not start with a letter remain unchanged). You must specify the first argument, "xxx", to specify the separators that you wish to use. The list must be enclosed in quote-marks. If you want to use a quote mark as a separator, prefix it with your [Escape Character](#)^[27].

Examples:

```
echo %@caps[" ",i love 4nt and take command]
echo %@caps[" ",,peter,paul,mary]
echo %@caps[" %=","sacrebleu!", he said]
```

6.3.10 @CDROM

@CDROM[d:]: Returns "1" if the drive is a CD-ROM or "0" otherwise. The drive letter must be followed by a colon.

Examples:

```
echo %@cdrom[C:]
echo %@cdrom[%_disk:]
```

6.3.11 @CEILING

@CEILING[n]: Returns the value of the smallest integer that is not less than *n*.

Examples:

```
echo %@ceiling[3.14]
echo %@ceiling[-3.14]
echo %@ceiling[0]
echo %@ceiling[123]
```

Also see [@FLOOR](#)^[387].

6.3.12 @CHAR

@CHAR[n]: Returns the character corresponding to an [ASCII](#)^[446] or Unicode numeric value. If the argument is a set of numeric values, CHAR returns a string. For example **%@CHAR[65]** returns A;

%@CHAR[65 66 67] returns ABC.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Note: Not all characters are printable. High ASCII characters (128-255) and Unicode characters may vary depending on the font used.

Examples:

```
echo %@char[65]
echo %@char[65 97 66 98 67 99]
```

6.3.13 @CLIP

@CLIP[n]: Returns line *n* from the Windows text clipboard. The first line is numbered 0. The string ****EOC**** is returned for all line numbers beyond the end of the clipboard.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Examples:

```
echo %@clip[0]
if "%@clip[2]" eq "**EOC**" echo No more data in the clipboard
```

6.3.14 @CLIPW

@CLIPW[string]: Writes the string to the Windows text clipboard. Returns "0" if the operation was successful.

Examples:

```
if "%@clipw[save this line]" eq "0" echo Saved to the clipboard
```

6.3.15 @COLOR

@COLOR[r,g,b]: Displays the Windows color common dialog and returns the RGB value for the selected color as a string in the form "**r,g,b**" (e.g. "0,128,64"). To specify the initially selected color, use the desired **r** (red), **g** (green) and **b** (blue) parameters as arguments. If no arguments are provided, the initial selection will be black ("0,0,0"). The arguments are optional, but if one is used all three must be used.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Examples:

```
@color[]
@color[155,0,0]
```


6.3.16 @COMMA

@COMMA[*n*]: Returns the number with commas (or the appropriate [thousands separator](#)^[111] for your current country setting) inserted where appropriate.

Note: Some [variable functions](#)^[357] can directly generate a numeric result with appropriate thousand separators if you add a "c" to their scale parameter.

Examples:

```
echo %@comma[12345678]
echo %@comma[0.12345678]
echo %@comma[%_xpixels]
```

6.3.17 @CONSOLE

@CONSOLE[*title*]: Returns **1** if the specified window title belongs to a console window. The ***title*** may include [wildcards](#)^[36].

6.3.18 @CONVERT

@CONVERT[*input*, *output*, *value*]: Converts a numeric string (***value***) from one number base (***input***) to another (***output***). Valid bases range from 2 to 36. The ***value*** can be between 0 and 2**64-1. No error is returned if ***value*** is outside that range.

Examples:

```
echo binary 1010101 is decimal %@convert[2,10,1010101]
echo decimal 20 is hex %@convert[10,16,20]
echo hexadecimal FF is octal %@convert[16,8,FF]
echo this year is %@convert[10,2,%_year] in binary
```

6.3.19 @CRC32

@CRC32[*filename*]: Returns the CRC32 value (computed using the same algorithm as PKZIP or WINZIP) of the file specified by ***filename***, or **-1** if the file does not exist or cannot be opened.

Also see [@MD5](#)^[400].

Examples:

```
echo %@crc32["C:\My Files\Myprog.exe"]
echo %@crc32["%comspec"]
```

6.3.20 @CWD

@CWD[*d* :]: Returns the current working directory of the specified disk drive in the format *d*:*pathname*. If the current working directory is the root directory, the format is *d*:\ . The drive letter must be followed

by a colon.

In Windows/98/SE/ME this information is retained by the OS, In Windows NT/2000/XP/2003 this information is handled by your command processor.

Examples:

```
echo %@cwd[C:]
echo %@cwd[_disk:]
```

See also: [@CWDS](#)^[369].

6.3.21 @CWDS

@CWDS[d:]: Returns the current working directory of the specified disk drive in the format *d:\pathname*. The drive letter must be followed by a colon.

In Windows/98/SE/ME this information is retained by the OS, In Windows NT/2000/XP/2003 this information is handled by your command processor.

Examples:

```
echo %@cws[C:]
echo %@cws[_disk:]
```

See also: [@CWD](#)^[368].

6.3.22 @DATE

@DATE[date]: Returns the number of days since January 1, 1980 for the specified date. See [date formats](#)^[363] for information on acceptable argument formats.

Examples:

```
echo %@date[01-01-1981]
echo %@date[%_date]
```

6.3.23 @DAY

@DAY[date]: Returns the numeric day of the month for the specified date. See [date formats](#)^[363] for information on acceptable argument formats.

Examples:

```
echo %@day[01-01-1980]
echo %@day[%_date]
```

6.3.24 @DEC

@DEC[n]: Returns the same value as [@EVAL](#)^[374][n - 1]. It is most often used to decrement the value of a variable, e.g. "%@DEC[%var]". The variable itself is not changed. To do that, use a command such as:

```
set var=%@dec[%var]
```

6.3.25 @DECIMAL

@DECIMAL[n]: Returns the portion of the number to the right of the [decimal point](#)^[102] as an integer numeric string. Trailing zeros are used to pad to the [minimum width](#)^[103] specified for [@EVAL](#)^[374]. For example, "%@decimal[%@eval[1/2]]" is "5" if minimum width is 0, and "50000" if minimum width is 5.

Examples:

```
echo %@decimal[1234]
echo %@decimal[1.234]
echo %@decimal[12.34]
```

6.3.26 @DESCRIPT

@DESCRIPT[filename]: Returns the file description for the specified filename (see [DESCRIBE](#)^[176]).

The **filename** must be in quotes if it contains white space or special characters.

Examples:

```
echo %@descript["D:\My Path\Myfile.exe"]
echo %@descript["%comspec"]
```

6.3.27 @DEVICE

@DEVICE[name]: Returns **1** if the specified name is a character device (such as a printer or serial port), or **0** if not.

Examples:

```
echo %@device[lpt1]
echo %@device[com1]
echo %@device[com5]
echo %@device[clip:]
echo %@device[clip]
```

6.3.28 @DIGITS

@DIGITS[n]: Returns **1** if the string is composed of decimal digits only, otherwise it returns **0**. The [DecimalChar](#)^[102], the [ThousandsChar](#)^[111], and the sign characters (+ or -) are *not* digits, and if they are present in the string @DIGITS will return **0**.

Examples:

```
echo %@digits[12345]
echo %@digits[-12345]
```

```
echo %@digits[1.2345]
```

6.3.29 @DIRSTACK

@DIRSTACK[n]: Returns the name of the **n**th entry in the directory stack. The most recent entry is number "0". If no "n" argument is specified, returns the total number of entries in the stack. The directory stack is that established by calls to [PUSHD](#)^[264] / [POPD](#)^[261].

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

See also: [DIRS](#)^[190], [POPD](#)^[261], [PUSHD](#)^[264] and [Directory Navigation](#)^[54].

Examples:

```
echo %@dirstack[0]
echo %@dirstack[2]
echo %@dirstack[]
```

6.3.30 @DISKFREE

@DISKFREE[d:[,scale][c]]: Returns the amount of free disk space on the specified drive. If you're specifying a drive, the drive letter must be followed by a colon. Optionally, you can specify a directory or UNC name, and @DISKFREE will display the free disk space on the drive referenced by that name, which may be different from the drive *if the directory is a link* to a directory on another drive.

The optional second argument specifies the reporting scale (see [Memory Size / Disk Space / File Size Units and Report Format](#)^[357]). Adding the letter **c** requests the result be formatted using the "thousands separator" (see [ThousandsChar](#)^[111]). Also see [@DISKTOTAL](#)^[371] and [@DISKUSED](#)^[372].

Examples:

```
echo %@diskfree[c:]
echo %@diskfree[%_disk:,Kc]
```

6.3.31 @DISKTOTAL

@DISKTOTAL[d:[,scale][c]]: Returns the total disk space on the specified drive. If you're specifying a drive, the drive letter must be followed by a colon. Optionally, you can specify a directory or UNC name, and @DISKTOTAL will display the total disk space on the drive referenced by that name, which may be different from the drive *if the directory is a link* to a directory on another drive.

The optional second argument specifies the reporting scale (see [Memory Size / Disk Space / File Size Units and Report Format](#)^[357]). Adding the letter **c** requests the result be formatted using the "thousands separator" (see [ThousandsChar](#)^[111]). Also see [@DISKFREE](#)^[371] and [@DISKUSED](#)^[372].

Examples:

```
echo %@disktotal[c:]
```

```
echo %@disktotal[%_disk:,Kc]
```

6.3.32 @DISKUSED

@DISKUSED[d:[,scale][c]]: Returns the amount of disk space in use by files, directories, or marked bad on the specified drive. If you're specifying a drive, the drive letter must be followed by a colon. Optionally, you can specify a directory or UNC name, and @DISKUSED will display the used disk space on the drive referenced by that name, which may be different from the drive *if the directory is a link* to a directory on another drive.

The second argument specifies the reporting scale (see [Memory Size / Disk Space / File Size Units and Report Format](#)^[357]). Adding the letter **c** requests the result be formatted using the "thousands separator" (see [ThousandsChar](#)^[117]). Also see [@DISKFREE](#)^[371] and [@DISKTOTAL](#)^[371].

Examples:

```
echo %@diskused[c:]
echo %@diskused[%_disk:,Kc]
```

6.3.33 @DOMAIN

@DOMAIN[name]: Returns the *domain* of the computer specified by the DNS or NetBios *name*. If *name* is not specified, returns the *domain* of the local computer.

6.3.34 @DOSMEM

Replaced by [@WINMEMORY](#)^[2].

6.3.35 @DOW

@DOW[date]: (Day Of Week) Returns the first three characters of the English name of the day of the week for the specified date ("Mon", "Tue", "Wed", etc.). See [date formats](#)^[363] for information on acceptable argument formats.

Examples:

```
echo %@dow[01-01-1980]
echo %@dow[%_date]
```

Also see [@IDOW](#)^[390].

6.3.36 @DOWF

@DOWF[date]: (Day Of Week, Full) Returns the full English name of the day of the week for the specified date ("Monday", "Tuesday", etc.). See [date formats](#)^[363] for information on acceptable argument formats.

Examples:

```
echo %@dowf[01-01-1980]
echo %@dowf[%_date]
```

Also see [@IDOWF](#)^[390].

6.3.37 @DOWI

@DOWI[*date*]: Returns an integer representing the day of the week for the specified date (1 = Sunday, 2 = Monday, etc.). See [date formats](#)^[363] for information on acceptable argument formats.

Examples:

```
echo %@dowi[01-01-1980]
echo %@dowi[%_date]
```

6.3.38 @DOY

@DOY[*date*]: Returns the Day Of Year for the specified date (1-366). See [date formats](#)^[363] for information on acceptable argument formats.

Examples:

```
echo %@doy[02-02-2005]
echo %@doy[%_date]
```

6.3.39 @ENUMSERVERS

@ENUMSERVERS[*n*,*server*]: Enumerate the servers on the network. *n* is the entry number in the list of servers (first one is "0"). *server* is the machine name(s) to match and may contain [wildcards](#)^[36]. Returns a null string if there are fewer than "n" matching servers. This function can be repeatedly called, incrementing "n" each time to enumerate all available server names until it returns a null string.

Note: Windows may require a significant amount of time before returning data to this function when used on "large" networks. This is not anything over which 4NT/TC has any control.

Examples:

```
echo %@enumservers[0,\\SERVER]
for %i in (0 1 2) echo %@enumservers[%i,*]
```

6.3.40 @ENUMSHARES

@ENUMSHARES[*n*,*servershares*]**: Enumerate the share names for the specified server. *n* is the entry number in the list of shares (first one is "0"). *server* is the server name, and *shares* is the sharename(s) to match and may contain [wildcards](#)^[36]. Returns a null string if there are fewer than "n" matching shares. This function can be repeatedly called, incrementing "n" each time to enumerate all available shares until it returns a null string.

Examples:

```
echo %@enumshares[0,\\SERVER\DRIVE_C]
for %i in (0 1 2) echo %@enumshares[%i,\\SERVER\DRIVE_*]
```

6.3.41 @ERRTEXT

@ERRTEXT[*n*]: Returns the operating system error text for the specified code.

Examples:

```
echo %@errtext[2]
echo %@errtext[255]
echo %@errtext[%_syserr]
```

6.3.42 @EVAL

@EVAL[*expression*[=*x.y*]*H*]]: 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 (to control evaluation order), commas, hexadecimals and decimal separators. Parentheses can be nested. **@EVAL** will strip leading and trailing zeros from the result unless you use the output formatting operators.

- ▶ [Special interpretation of variable names](#)^[374]
- ▶ [Hexadecimal data](#)^[374]
- ▶ [Arithmetic operators](#)^[374]
- ▶ [Trigonometric and transcendental functions](#)^[374]
- ▶ [Order of precedence](#)^[374]
- ▶ [Display precision and output format](#)^[374]
- ▶ [Precision of internal calculations](#)^[374]
- ▶ [Examples](#)^[374]

Special interpretation of variable names

If the name of a variable starts with any character other than **x**, the percent sign % that must normally precede it to use its value instead of its name *may be dropped*.

Hexadecimal data

You can use hexadecimal numbers up to 16 digits long in expressions to be evaluated by *preceding the number with 0x*. For example:

```
[c:\] echo %@eval[0x10 + 0x10]
32
[c:\] echo %@convert[10, 16, %@eval[0x10 + 0x10]]
20
```

You can specify *hexadecimal output* with the special syntax **@eval[...=H]**. For example:

```
echo %@eval[3*6=H]
```

will output 12 (hex). The default is decimal output. To convert between and decimal and hexadecimal formats, see the [@CONVERT](#)^[368] function.

Arithmetic operators

- + (with one parameter) sign of numeric parameter (e.g. +3)
- + (with two parameters) addition
- (with one parameter) negation of symbolic parameter (e.g., -%n) or sign of numeric parameter (e.g. -1, +3)

-	(with two parameters) subtraction
*	multiplication
/	division
\	integer division (returns the integer part of the quotient)
MOD	modulo (returns the remainder when the first parameter is divided by the second)
%%	same as MOD
**	exponentiation
AND	bitwise and (returns 1 for each bit position where the corresponding bits in both parameters are 1)
&	same as AND
OR	bitwise or (returns 1 for each bit position where the corresponding bit in at least one parameter is 1)
	same as OR
XOR	bitwise exclusive or (returns 1 for each bit position where the corresponding bits of the two parameters are different)
^	same as XOR
SHL	arithmetic left shift of the first parameter, truncated toward zero to an integer, by the number of bits specified by the second parameter
<<	same as SHL
SHR	arithmetic right shift of the first parameter, truncated toward zero to an integer, by the number of bits specified by the second parameter
>>	same as SHR

NOTE: Under some conditions you must disable the normal interpretation of ">" and "<" as redirection symbols with [SETDOS /X-6](#)^[283] to use >> or << as shift operators. This is never required if you use the alternate operators SHL or SHR.

Trigonometric and transcendental functions

You can use as operands the trigonometric and transcendental functions below. *The argument of trigonometric functions is interpreted as radians.*

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

The special string **PI** is a shortcut for the value **3.14159265358979323846**.

Order of precedence

1. variables,
2. expressions in matching parentheses,
3. functions,
4. exponentiation;
5. multiplication, division, and MOD;
6. addition and subtraction;
7. AND, OR, XOR, SHL, and SHR.

When multiple consecutive expressions of a single precedence level are used, evaluation is left to right.

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.

Note: 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]
```

Display precision and output format

The maximum display 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 with the configuration dialogs, the [EvalMax](#)^[103] and [EvalMin](#)^[103] directives in the [.INI file](#)^[89] and with the [SETDOS](#)^[283] /F command. You can alter the decimal character with the configuration dialogs, the [DecimalChar](#)^[102] directive or the [SETDOS](#)^[283] /G command. Any of these changes only alter the way a result is displayed, not how it is computed internally.

You can alter the display precision for a single evaluation with the construct **@EVAL[expression=x.y]**. The **x** parameter specifies the minimum decimal precision (the minimum number of decimal places displayed); the **y** parameter sets the maximum decimal precision. You can use **=x,y** instead of **=x.y** if the comma is your decimal separator. You can specify either or both parameters. If **x > y**, **x** is ignored. If only **x** is specified, it is used as the value of both **x** and **y** (e.g. **=2** is equivalent to **=2.2**).

Precision of internal calculations

4NT and **TC** use 64-bit floating point arithmetic internally, i.e., each value (a "double") is represented by 64 bits: 1 for the sign, 11 for the exponent, 52 for the fraction part. The range of possible values is $\pm 1.7\text{E}308$ (at least 15 digits of precision). The effective precision of the result of **@EVAL** depends on which specific operations were performed on which values.

Examples

Expression	Value
@eval[3 / 6=2.4]	0.50
@eval[3 / 6=4.4]	0.5000
@eval[3 / 7]	0.4285714286
@eval[3 / 7=.4]	0.4286
@eval[3 / 7=2.2]	0.42
@eval[3 / 7=2]	0.42

Also see [@DEC](#)^[376] and [@INC](#)^[396].

6.3.43 @EXEC

@EXEC[command]: Execute **command** and return its numeric exit code.

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](#)^[262]. It is a "back door" entry into command processing and should be used with caution.

By default, @EXEC returns the result code from the command (see the [?](#)^[346] and [_?](#)^[346] variables), but if you preface the command name with an '@' then @EXEC returns an empty string.

Example:

```
PROMPT=%@exec[@color 15 on %@if[%@removable[%_disk] eq 0,2,4]%%+echos
[%_cwd%]%%+color 11 on 0]$s
```

Also see [@EXECSTR](#)^[377].

6.3.44 @EXECSTR

@EXECSTR[*command*]: Runs the specified ***command*** and returns the first line written to *stdout* by ***command***.

@EXECSTR is a "back door" entry into command processing and should be used with caution.

@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:

```
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]
```

@EXECSTR can also be used with internal commands:

```
echo Newest file is: %@execstr[*dir /a:-d /h /o:-t /f]
```

@EXECSTR involves several extensive internal processing stages. You might be able to use more complex command sequences (pipes, command groups, etc.) as its argument, but always *test carefully* first as the results may not always be what you expect. We recommend that you only use a single command (internal, external, batch file, etc.) argument.

Note: Remember that only the first line of standard output is returned. Since many internal and external commands start their text output on the second line, @EXECSTR[] may not return any useful information from those commands.

Also see [@EXEC](#)^[376].

6.3.45 @EXETYPE

@EXETYPE[*filename*]: Returns the application type:

0	Unknown
1	DOS app
2	PIF file
3	Win16
4	Win 3.x VxD
5	OS/2
6	Win32 GUI
7	Win32 console
8	Posix

Examples:

```
echo %@exetype["d:\path\myprog.exe"]
echo %@exetype["%comspec"]
```

6.3.46 @EXPAND

@EXPAND[*range...*] *filename*[,*[-rhsad]*]: Returns, on a single line, the names of all files and directories that are within the specified *range[s]*, **AND** match *filename*, **AND** have the specified attributes. *Filename* may contain [wildcards](#)^[36] and [include lists](#)^[46]. Returns an empty string if no files match. If the file list is longer than the allowed [command line length](#)^[25], it will be truncated without an error message. Each returned filename which contains whitespace or other special characters will be delimited by quote marks.

Filename must be in quotes if it contains white space or special characters.

The *range* and attribute arguments, if included, define properties of the files that will be included in the search as specified in [File Selection](#)^[36]. Multiple *range* arguments may be included, but not more than one each of [size range](#)^[42], [date range](#)^[42], and [time range](#)^[44]. *Range* arguments must precede *filename*. Each *range* argument is of the form

/[*a...*]

where *a* is one of *d*, *s* or *t*, optionally followed by range parameters. [Exclusion ranges](#)^[45] **are not supported**.

Examples:

```
echo %@expand[/[s2k,3k] *.txt]
    all files with extension txt in the current directory with size at least 2000 bytes and at most
3000 bytes
echo %@expand[* ,d]
    all subdirectories
echo %@expand[/d-365] %windir\w*.exe;w*.dll]
    all files at most 365 days old in the Windows directory, with extension EXE or DLL, and name
beginning with w.
```

6.3.47 @EXT

@EXT[*filename*]: Returns the extension from *filename*, without a leading period. On volumes which support long file names, the extension can be up to 255 characters long. On FAT drives it can be up to 3 characters long. *filename* must be in quotes if it contains white space or special characters.

On an LFN drive, the returned extension may contain white space or special characters. To avoid problems which could be caused by these characters, quote the returned extension before you pass it to other commands.

Examples:

```
echo %@ext[%@comspec]
echo "%@ext["LFN Names may have.very long extensions"]]"
```

6.3.48 @FIELD

@FIELD[["**xxx**",]**n**, **string**]: Returns the **n**th field in **string**. The first field is numbered **0**. If **n** is **negative** or it is preceded by a hyphen (minus sign), fields are counted backwards from the end of **string**, and the absolute value of **n** is used. You can specify the rightmost field by setting **n** to **-0**.

@FIELD^[379], **@FIELDS**^[379], **@WORD**^[416] and **@WORDS**^[417] default to spaces, tabs, and commas as separators. You can use the optional first argument, "**xxx**", to specify the separators that you wish to use. If you want to use a quote mark as a separator, prefix it with an **escape character**^[103]. If **string** is enclosed in quotation marks, you **must** specify the list of delimiters.

@FIELD^[379] and **@FIELDS**^[379] differ from **@WORD**^[416] and **@WORDS**^[417] in how multiple consecutive separators are counted. **@WORD**^[416] and **@WORDS**^[417] consider a sequence as a single separator. In contrast, **@FIELD**^[379] and **@FIELDS**^[379] count each occurrence of a separator individually, including counting individually each of the adjacent separators, which *allows you to rprocess empty fields in string*.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). To use hexadecimal form for a negative **n**, remember to use 32-bit 2's complement arithmetic. You cannot use the hexadecimal form to specify field **-0** (the rightmost field).

See also: **@WORD**^[416], **@WORDS**^[417], **@FIELDS**^[379].

Examples:

function	value
%@field[2,zero,one,two,three]	two
%@field[2,zero,,two,three]	two
%@field["\",2,C:\Program Files\My Dir\myapp.exe]	My Dir
%@field[-2,zero,one,two,three]	one

6.3.49 @FIELDS

@FIELDS[["**xxx**",]**string**]: Returns the number of fields in **string**.

@FIELD^[379], **@FIELDS**^[379], **@WORD**^[416] and **@WORDS**^[417] default to spaces, tabs, and commas as separators. You can use the optional first argument, "**xxx**", to specify the separators that you wish to use. If you want to use a quote mark as a separator, prefix it with an **escape character**^[103]. If **string** is enclosed in quotation marks, you **must** specify the list of delimiters.

@FIELD^[379] and **@FIELDS**^[379] differ from **@WORD**^[416] and **@WORDS**^[417] in how multiple consecutive separators are counted. **@WORD**^[416] and **@WORDS**^[417] consider a sequence as a single separator. In contrast, **@FIELD**^[379] and **@FIELDS**^[379] count each occurrence of a separator individually, including counting individually each of the adjacent separators, which *allows you to rprocess empty fields in*

string.

Also see [@WORD](#)^[416], [@WORDS](#)^[417], [@FIELD](#)^[379].

6.3.50 @FILEAGE

@FILEAGE[filename[,a|c|w]]: Returns the date and time of the file as a single numeric value, which is a 64-bit integer that represents the time elapsed since 1601-01-01@00:00:00 (local time) with 100-ns resolution.

Filename must be in quotes if it contains white space or special characters. The optional second argument selects which date field is returned for files on a [VFAT](#)^[485] or [NTFS](#)^[481] drive: **a** means the last access date, **c** means the creation date, and **w** means the last modification (write) date. The default is **w**.

Note: The value of this function in all versions of 4DOS and in versions of 4NT and TC up to 5.0 is a 32-bit number, which is a nonlinear representation of the file date and time the same information with a 2-s resolution, matching the FAT file timestamp format..

Examples:

```
echo %@fileage[d:\path\myfile.ext]
echo %@fileage["%comspec",c]
```

Also see [Time Stamps](#)^[443], [@AGEDATE](#)^[364] and [@MAKEAGE](#)^[399]

6.3.51 @FILECLOSE

@FILECLOSE[n]: Closes the file whose handle is **n**. You cannot close handles 0, 1 or 2. Returns **0** if the file was successfully closed, or **-1** if an error occurred.

This function should only be used with file handles returned by [@FILEOPEN](#)^[381]. If you use it with any other number you may damage other files opened by the command processor (or by the program which started the command processor).

Also see the related handle-based functions:

@FILEOPEN ^[381]	Open a file handle
@FILEREAD ^[382]	Read next line from a file handle
@FILESEEK ^[383]	Move a file handle pointer
@FILESEEKL ^[383]	Move a file handle pointer to a specified line
@FILEWRITE ^[385]	Write next line to a file handle
@FILEWRITEB ^[385]	Write data to a file handle

6.3.52 @FILEDATE

@FILEDATE[filename[,acw][,n]]: Returns the date a file was last modified, in the default country format (mm-dd-yy for the US). The **filename** must be in quotes if it contains white space or special characters. 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:

Code	Date format
0	system default
1	USA (mm/dd/yy)
2	European (dd/mm/yy)
3	Japan (yy/mm/dd)
4	ISO 9601 (yyyy-mm-dd)

Examples:

```
echo %@filedate["D:\my path\myfile.exe"]
echo %@filedate["comspec",c,4]
```

See [Time Stamps](#)^[443], [@FILETIME](#)^[384], [@FILEAGE](#)^[380].

6.3.53 @FILENAME

@FILENAME[*filename*]: Returns the name and extension of a file, without a path.

The ***filename*** must be in quotes if it contains white space or special characters. On an LFN drive, the returned filename 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.

Examples:

```
echo %@filename["D:\my path\myfile.exe"]
echo %@filename["comspec"]
```

6.3.54 @FILEOPEN

@FILEOPEN[*filename*, r[ead] | w[rite] | a[ppend][,b[t]]]: Opens the file in the specified mode and returns the file handle as an integer. 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](#)^[382] without a *length*, and to write text using [@FILEWRITE](#)^[385]. Binary mode should be used to read binary data with [@FILEREAD](#)^[382] with a *length*, and to write binary data with [@FILEWRITE](#)^[385]. Returns -1 if the file cannot be opened.

The ***filename*** must be in quotes if it contains white space or special characters.

To open a file for both reading and writing, open it in **append** mode, then use [@FILESEEK](#)^[383] to seek to the start of the file (or any other desired location) before performing additional operations.

@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 are inheritable. For more information on named pipes see your Windows documentation.

@FILEOPEN can open file streams on NTFS drives under Windows 2000 and above if the stream name is specified. See [NTFS File Streams](#)^[444] for additional details on file streams.

You must reference the file exclusively using the returned *file handle*, and you must close the file using the *file handle*. This is especially important when you are debugging a batch program using @FILEOPEN. If you suspect that file handles have been opened and not closed, you should exit the

command processor, and restart it.

Examples:

```
set h=%@fileopen["d:\path\myfile.txt",write]
echo writing %@filewrite[%h,this is a test] bytes
echo closing handle %#h: %@fileclose[%h]
```

Also see the related handle-based functions:

@FILECLOSE ^[380]	Close a file handle
@FILEREAD ^[382]	Read next line from a file handle
@FILESEEK ^[383]	Move a file handle pointer
@FILESEEKL ^[383]	Move a file handle pointer to a specified line
@FILEWRITE ^[385]	Write next line to a file handle
@FILEWRITEB ^[385]	Write data to a file handle

6.3.55 @FILEREAD

@FILEREAD[n[,length]]: Reads data from the file whose handle is *n*. Returns the string ****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.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

This function should only be used with file handles returned by [@FILEOPEN](#)^[381]!

Also see the related handle-based functions:

@FILECLOSE ^[380]	Close a file handle
@FILEOPEN ^[381]	Open a file handle
@FILESEEK ^[383]	Move a file handle pointer
@FILESEEKL ^[383]	Move a file handle pointer to a specified line
@FILEWRITE ^[385]	Write next line to a file handle
@FILEWRITEB ^[385]	Write data to a file handle

6.3.56 @FILES

@FILES[[range...] filename[, []-rhsad]]: Returns the number of files within *range* that match the *filename* and have the specified attributes. *Filename* may contain [wildcards](#)^[36] and [include lists](#)^[46]. 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](#)^[374].

Filename must be in quotes if it contains white space or special characters.

The *range* and attribute arguments, if included, define properties of the files that will be included in the search as specified in [File Selection](#)^[36]. Multiple *range* arguments may be included, but not more than one each of [size range](#)^[42], [date range](#)^[42], and [time range](#)^[44]. *Range* arguments must precede *filename*. Each *range* argument is of the form

/[**a...**]

where **a** is one of **d**, **s** or **t**, optionally followed by range parameters. [Exclusion ranges](#)^[45] **are not supported**.

Examples:

```
echo %@files[/[s2k,3k] *.txt]
    number of files with extension txt in the current directory with size at least 2000 bytes and at
    most 3000 bytes
echo %@files[* ,d]
    number of subdirectories
echo %@files[/[d-365] %windir\w*.exe;w*.dll]
    number of files at most 365 days old in the Windows directory, with extension EXE or DLL, and
    name beginning with w.
```

6.3.57 @FILESEEK

@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).

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

This function should only be used with file handles returned by [@FILEOPEN!](#)^[381]. If you use it with any other number you may damage other files opened by the command processor (or by the program which started the command processor).

Also see the related handle-based functions:

@FILECLOSE ^[380]	Close a file handle
@FILEOPEN ^[381]	Open a file handle
@FILEREAD ^[382]	Read next line from a file handle
@FILESEEKL ^[383]	Move a file handle pointer to a specified line
@FILEWRITE ^[385]	Write next line to a file handle
@FILEWRITEB ^[385]	Write data to a file handle

6.3.58 @FILESEEKL

@FILESEEKL[n,line,[1]]: 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. The optional third argument determines the starting point for the seek. If it is set to "**1**", seek will start from the current position in the file. If that third argument is missing or not equal to 1, seek will start from the beginning of the file.

@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.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

This function should only be used with file handles returned by [@FILEOPEN](#)^[381]! If you use it with any other number you may damage other files opened by the command processor (or by the program which started the command processor).

Also see the related handle-based functions:

@FILECLOSE ^[380]	Close a file handle
@FILEOPEN ^[381]	Open a file handle
@FILEREAD ^[382]	Read next line from a file handle
@FILESEEK ^[383]	Move a file handle pointer
@FILEWRITE ^[385]	Write next line to a file handle
@FILEWRITEB ^[385]	Write data to a file handle

6.3.59 @FILESIZE

@FILESIZE[filename[,scale[c][,a]]]: Returns the size of a file, or -1 if the file does not exist. If **filename** includes [wildcards](#)^[36] or an [include list](#)^[46], it 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.

Filename must be in quotes if it contains white space or special characters.

The second argument specifies the reporting scale (see [Memory Size / Disk Space / File Size Units and Report Format](#)^[357]). Adding the letter **c** requests the result be formatted using the "thousands separator" (see [ThousandsChar](#)^[111]).

Examples:

```
echo %@filesize[d:\path\myfile.ext]
echo %@filesize["%comspec",bc]
echo %@filesize["%comspec",bc,a]
```

6.3.60 @FILETIME

@FILETIME[filename[,acw][,s]]: Returns the time a file was last modified, in hh:mm format. The **filename** must be in quotes if it contains white space or special characters. 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, and without a seconds field, on non-NTFS drives (see [Time Stamp](#)^[443] for additional details).

Examples:

```
echo %@filetime["D:\my path\myfile.exe"]
echo %@filetime["comspec",c,4]
```

Also see [@FILEDATE](#)^[380], [@FILEAGE](#)^[380].

6.3.61 @FILEWRITE

@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.

This function should only be used with file handles returned by [@FILEOPEN](#)^[381]. If you use it with any other number you may damage other files opened by the command processor (or by the program which started the command processor).

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Also see the related handle-based functions:

@FILECLOSE ^[380]	Close a file handle
@FILEOPEN ^[381]	Open a file handle
@FILEREAD ^[382]	Read next line from a file handle
@FILESEEK ^[383]	Move a file handle pointer
@FILESEEKL ^[383]	Move a file handle pointer to a specified line
@FILEWRITEB ^[385]	Write data to a file handle

6.3.62 @FILEWRITEB

@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.

Note: (Unicode) Writes ASCII output when passed a *Unicode* string to more closely match the behavior of the ASCII versions. Note that if you're trying to write non-English (>128) characters with [@FILEWRITEB\[\]](#), the output will probably not match the input. See [ASCII & Unicode Versions](#)^[422].

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

This function should only be used with file handles returned by [@FILEOPEN](#)^[381]. If you use it with any other number you may damage other files opened by the command processor (or by the program which started the command processor).

Also see the related handle-based functions:

@FILECLOSE ^[380]	Close a file handle
@FILEOPEN ^[381]	Open a file handle
@FILEREAD ^[382]	Read next line from a file handle
@FILESEEK ^[383]	Move a file handle pointer
@FILESEEKL ^[383]	Move a file handle pointer to a specified line
@FILEWRITE ^[385]	Write next line to a file handle

6.3.63 @FINDCLOSE

@FINDCLOSE[[filename]]: Signals the end of a [@FINDFIRST](#)^[386] ... [@FINDNEXT](#)^[386] sequence. You must use this function to release the directory search handle. *Filename* is unnecessary, this function can be simply called as [%@FINDCLOSE\[\]](#) without arguments. [@FINDCLOSE](#) returns 0 if a [@FINDFIRST](#)^[386] ... [@FINDNEXT](#)^[386] sequence is in effect, a non-zero value otherwise.

Examples:

```
echo %@findfirst[*.exe]
echo %@findclose[]
```

6.3.64 @FINDFIRST

@FINDFIRST[*range...* *filename* [, *[-]rhsad*]]: Returns the name of the first file that matches *filename*, which may include [wildcards](#)^[36] and/or an [include list](#)^[46], and which file has the properties specified in the optional [range](#)^[40] and [attribute](#)^[39] arguments.

Filename must be in quotes if it contains white space or special characters.

The *range* and attribute arguments, if included, define properties of the files that will be included in the search as specified in [File Selection](#)^[36]. Multiple *range* arguments may be included, but not more than one each of [size range](#)^[42], [date range](#)^[42], and [time range](#)^[44]. *Range* arguments must precede *filename*. Each *range* argument is of the form

```
/[a...]
```

where *a* is one of *d*, *s* or *t*, optionally followed by range parameters. [Exclusion ranges](#)^[45] **are not supported**.

On an LFN or NTFS drive, the *returned* filename may contain white space or other special characters. Unlike [@EXPAND](#)^[378], no quote marks are added by this function. To avoid problems which could be caused by these characters, quote the returned name before you pass it to other commands. See the notes under [Variable Functions](#)^[357] for additional details.

@FINDFIRST[] locates the *first* file matching the requirements. To find more matching files, you must use [@FINDNEXT](#)^[386], and terminate the search with [@FINDCLOSE](#)^[385]. **Warning!** *If you fail to terminate the search, the command processor may hang.*

Warning: **@FINDFIRST**[] searches may not be nested!

Examples:

```
%@findfirst[/[d-30] *]
    locate files created 30 days ago or since
%@findfirst[/[s2k,3k] "%windir\*.exe",a]
    locate files with extension exe, the archive flag set, at least 2,000 bytes but not more than
    3,000 bytes long, in Windows' startup directory.
```

6.3.65 @FINDNEXT

@FINDNEXT[*filename* [, *[-]rhsad*]]: Returns the name of the next file that matches the *filename*(s) in the previous **@FINDFIRST** call. Returns an empty string when no more files match. **@FINDNEXT** should only be used after a successful call to [@FINDFIRST](#)^[386].

You do not need to include the *filename* parameter, because it must be the same as the one used in the previous **@FINDFIRST** call, unless you want to change *file attributes* for **@FINDNEXT**. *Filename*, if used, must be in quotes if it contains white space or special characters.

The attribute argument, if included, defines the attributes of the files that will be included in the search as specified in [Attribute Switches](#)^[39].

Range arguments may not be used in this function. The **range** arguments specified in the preceding **@FINDFIRST** call remain effective.

If you don't need to change the attribute parameters established by the preceding **@FINDFIRST**, you can simply use this function as **%@FINDNEXT[]** without arguments.

On an LFN or NTFS drive, the *returned* filename may contain white space or other special characters. Unlike **@EXPAND[]**^[378], no quote marks are added by this function. To avoid problems which could be caused by these characters, quote the returned name before you pass it to other commands. See the notes under **Variable Functions**^[357] for additional details.

@FINDFIRST[] locates the *first* file matching the requirements. To find more matching files, you must use **@FINDNEXT[]**^[386], and terminate the search with **@FINDCLOSE[]**^[385]. **Warning!** *If you fail to terminate the search, the command processor may hang.*

Examples:

```
echo %@findfirst[*]
echo %@findnext[]
echo %@findnext[* ,d]
echo %@findclose[]
```

6.3.66 @FLOOR

@FLOOR[n]: Returns the largest integer that is *not greater* than *n*.

Examples:

```
echo %@floor[3.14]
echo %@floor[-3.14]
echo %@floor[0]
echo %@floor[123]
```

Also see **@CEILING**^[366].

6.3.67 @FORMAT

@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[7,Hello]" returns " Hello"
"%@format[.3,Hello]" returns "Hel"
```

@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"
```

Also see **@FORMATN**^[388].

6.3.68 @FORMATN

@FORMATN**[[-]width[.precision],value]**: Format a numeric value. **Width** is a nonnegative integer specifying the minimum number of characters printed. If **Width** has a leading 0, the number will be left-padded with zeros. If the number of characters in the output value is less than the specified width, blanks are added to the left or the right of the values depending on whether the "-" flag (for left alignment) is specified, until the minimum width is reached. **Precision** specifies the number of digits after the decimal point. The **value** is rounded to the appropriate number of digits.

If you don't specify a precision, @FORMATN will default to 16 decimal places, and will usually not round the number appropriately. (For example, @FORMATN[3,3.4] will produce "3.3999999999999999".)

Also see [@FORMAT](#)^[387].

6.3.69 @FSTYPE

@FSTYPE[d:]: Returns the file system type for the specified drive. @FSTYPE returns **NTFS** for a drive that uses the Windows NTFS file system. It returns **FAT** for FAT12, FAT16, FAT32, and VFAT drives.

Examples:

```
echo %@fstype[c:]
echo %@fstype[_disk:]
```

6.3.70 @FULL

@FULL[filename]: Returns the full path and filename of a file. The **filename** must be in quotes if it contains white space or special characters. On an LFN drive, the returned filename 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. See the notes under [Variable Functions](#)^[357] for additional details.

Note: The @FULL function makes no assumption about the *existence* of a file or directory. Its **filename** argument can be any string and the function will attempt to turn it into a fully qualified "volume + path + name" specification, whether that full reference exists or not.

Examples:

```
echo %@full[foo.bar]
echo "%@full[.]"
echo "%@full["\Program Files"]"
```

6.3.71 @FUNCTION

@FUNCTION[name]: Returns the definition of the specified [user-defined function](#)^[218] **name** 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](#)^[283]. Otherwise, command separators, redirection characters, and other similar *punctuation* in the function may be interpreted as part of the current command, rather than part of a simple text string.

Example:

```
echo %@function[myfunction]
```

See the [FUNCTION](#)^[218] command.

6.3.72 @GETDIR

@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 **d:\path** must be in quotes if it contains white space or special characters. On an LFN drive, the returned name 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. See the notes under [Variable Functions](#)^[357] for additional details.

Examples:

```
cdd %@getdir["C:\program Files"]
%@getdir[]\
```

Note: @GETDIR deals with directories. All directories are folders, but not all folders are directories. To select a symbolic folder, see [@GETFOLDER](#)^[389].

6.3.73 @GETFILE

@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. 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 semicolons. 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. On an LFN or NTFS drive, the returned filename 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. See the notes under [Variable Functions](#)^[357] for additional details.

Examples:

```
echo %@getfile[*]
echo %@getfile["%windir",*.exe;*.com]
```

6.3.74 @GETFOLDER

@GETFOLDER[startdir]: Returns a folder selected from a tree view of available symbolic folders. If you don't specify a start folder, @GETFOLDER starts at "*My Computer*" or the equivalent symbolic folder in your specific Windows configuration.

Examples:

```
echo %@getfolder[]
echo %@getfolder["Control Panel"]
```

Note: @GETFOLDER deals with folders. All directories are folders, but not all folders are directories. To select a directory, see [@GETDIR](#)^[389].

6.3.75 @HISTORY

@HISTORY[x,y]: Returns a line or word from the [command history](#)^[13]. This function will prove most useful in keystroke aliases. "**x**" is the line to retrieve (current line = 0), and "**y**" is the specific word (first word = 0) desired within that line. If "**y**" is omitted, returns the entire line.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

6.3.76 @IDOW

@IDOW[date]: (International Day of Week) Returns the 3-character abbreviation for the day of the week for the specified date, in the *current locale* language. See [date formats](#)^[363] for information on acceptable argument formats.

Examples:

```
echo %@idow[01-01-1980]
echo %@idow[%_date]
```

Also see [@DOW](#)^[372].

6.3.77 @IDOWF

@IDOWF[date]: (International Day of Week, Full) Returns the full name for the day of the week for the specified date, in the *current locale* language. See [date formats](#)^[363] for information on acceptable argument formats.

Examples:

```
echo %@idowf[01-01-1980]
echo %@idowf[%_date]
```

Also see [@DOWF](#)^[372].

6.3.78 @IF

@IF[condition,string1,string2]: Evaluates *condition* according to the rules described in the topic [Conditional Expressions](#)^[31]. If *condition* is true, **@IF** returns *string1*. If it is false, **@IF** returns *string2*.

Examples

1) The expression `%@IF[2 == 2,Correct!,Oops!]` returns `Correct!`.

2) The command

```
echo Good %@if[%_hour ge 12,evening,morning]!
```

displays `Good morning!` in the AM hours and `Good evening!` in the PM hours.

6.3.79 @INC

@INC[n]: Returns the same value as [@EVAL](#)^[374] `[n + 1]`. It is most often used to increment the value of a variable, e.g. `"%@INC[%var]"`. The variable itself is not changed. To do that, use a command such as this:

```
set var=%@inc[%var]
```

6.3.80 @INDEX

@INDEX[*string1*,*string2*[,*-n*]]: Returns the offset of *string2* within *string1*, or -1 if *string2* is not found or if *string1* is empty. The first or leftmost position in *string1* is numbered 0. The optional third parameter *n* has three different interpretations:

- If *n* > 0, it specifies that the *n*th match from left to right is desired.
- If *n* < 0, it specifies that the *n*th match from right to left is desired.
- If *n*=0, the total number of matches is desired.

When *n* is omitted, the value returned is the offset of the *first* (leftmost) match.

Examples:

- locate the first "h" match:

```
echo %@index[This is a fine help file,h]
```

(found at offset 1, i.e. 2nd character)
- locate the second "s" match:

```
echo %@index[This is a fine help file,s,2]
```

(found at offset 6, i.e. 7th character)
- locate the last "i" match:

```
echo %@index[This is a fine help file,i,-1]
```

(found at offset 21, i.e. 22nd character)
- count the number of "f" matches:

```
echo %@index[This is a fine help file,f,0]
```

(2 matches found)
- tip: searching for a comma:

```
echo %@index["4NT, Take Command, 4DOS",^,,0]
```

(quote the string, and [escape](#)^[103] the comma literal)
- tip: searching for a quote:

```
echo %@index[contains a "quoted" word,^",0]
```

([escape](#)^[103] the quote literal)

6.3.81 @INIREAD

@INIREAD[*filename*,*section*,*entry*]: Returns the value of the *first* matching *entry* from the specified *file*, or an empty string if either *file* or the entry in *file* does not exist. *File* may, but need not be associated with your command processor. If *file* contains more than one section named *section*, only the first one is searched for *entry*.

File, *section*, and *entry* must be in quotes if they contain white space or special characters.

File selection

Both the name and extension of *file* must be specified. *This function does not apply a default extension.* If *file* does not explicitly include a path, @INIREAD uses %Windir, the Windows installation directory. To use the current directory, you must explicitly specify it, e.g., using .\ as the path.

Example

```
%@iniread[c:\tcmd\tcmd32.ini,TakeCommand,history]
```

returns the size of the command history if it is specified in *TCMD32.INI*. If you don't specify a path for the *.INI* file, @INIREAD will look in the %Windir and %Windir\System directories.

6.3.82 @INIWRITE

@INIWRITE[*file*,*section*,*entry*,*string*]: Creates, updates, or deletes an entry in the specified *file*, which may, but need not be associated with your command processor. If *file* does not exist, it will be created. @INIWRITE returns 0 for success or -1 for failure.

File, *section*, and *entry* must be in quotes if they contain white space or special characters.

File selection

Both the name and extension of *file* must be specified. *This function does not apply a default extension.* If *file* does not explicitly include a path, @INIWRITE uses %Windir, the Windows installation directory. To use the current directory, you must explicitly specify it, e.g., using *.* as the path.

Action

If *file* does not exist, it will be created. If *string* is empty, *file* will be empty, otherwise a section line and a directive line will be created.

The remaining descriptions relate to the case when *file* exists.

If more than one match for *section* exists in *file*, only the first one is searched for *entry*. If more than one match exists for *section* and *entry*, only the first match is one is affected. Searching starts at the beginning of the file, and stops on the first match.

If *string* is empty, the matching *section* and *entry*, if any, is deleted.

If *string* is not empty, and there is a matching *section* and *entry*, it is modified.

If *string* is not empty, and there is no matching *section* and *entry*, it is created.

Examples

```
echo %@iniwrite[c:\tcmd\tcmd32.ini,TakeCommand,history,8192]
```

will set the size of the command history to 2,048 bytes.

```
echo %@iniwrite[c:\tcmd\tcmd32.ini,TakeCommand,history,]
```

will remove the "history" entry from the file.

6.3.83 @INSERT

@INSERT[*offset*,*string1*,*string2*]: Inserts *string1* into *string2* starting at *offset*. The first offset in *string2* is 0. If *offset* is greater than the length of *string2*, *string1* will be appended to the end of *string2*. If *offset* is **negative**, its magnitude is used to count backward

form the end of *string2*, but not past its beginning. Setting *offset* to *-0* is the same as setting it to 0, i.e., *string1* will precede *string2* in the result. To include a comma in *string1*, precede it with your [EscapeChar](#)^[103].

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). To use hexadecimal form for a negative *n*, remember to use 32-bit 2's complement arithmetic.

Examples:

<i>function</i>	<i>value</i>
%@insert[1,arm,wing]	warming
%@insert[8,very ,this is useful]	this is very useful
%@insert[255,%=, very!,this is useful]	this is useful, very!
%@insert[-9,very ,this is useful]	this very is useful
%@insert[0,abcde,xyz]	abcdexyz

6.3.84 @INSTR

@INSTR[*start,length,string*]: Returns a substring, beginning at offset *start* and continuing for *length* characters. If *length* is **positive** or it is omitted, the offset is measured from the beginning (i.e., left end) of the string. If *length* is omitted, all of the *string* beginning at offset *start* is returned. If *length* is **negative**, the *offset* is measured leftward from the right end of the string, and magnitude of *length* is used. [@SUBSTR](#)^[407] is an older version of the same function.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). To use hexadecimal form for a negative *length*, remember to use 32-bit 2's complement arithmetic.

Examples:

<i>function</i>	<i>value</i>
=%@instr[8,3,this is useful]=	=use=
=%@instr[8,,this is useful]=	=useful=
=%@instr[8,-4,this is useful]=	=is u=
=%@instr[8,,commas, they don't matter]=	=they don't matter=

6.3.85 @INT

@INT[*n*]: Returns the integer part of the number *n*.

Examples:

```
echo %@int[1234]
echo %@int[1.234]
echo %@int[12.34]
```

6.3.86 @IPADDRESS

@IPADDRESS[host]: Returns the numeric IP address for the specified host name. If the **host** parameter is omitted, returns the current local IP. The IP is displayed in the common *dotted decimal* format ("nnn.nnn.nnn.nnn"). An invalid or unknown host name will return an error (see [@ERRTEXT](#)^[374] to decipher the error number if necessary).

Also see: [@IPNAME](#)^[394].

Example:

```
echo %@ipaddress[jpsoft.com]
echo %@ipaddress[]
```

6.3.87 @IPNAME

@IPNAME[numeric IP]: Returns the host name for the specified **numeric IP** address. An IP value of "0" returns the name of the current local host (usually the computer name). The IP number can be expressed in the common *dotted decimal* format ("nnn.nnn.nnn.nnn") or as a *packed decimal*. An invalid or unknown IP value will return an error (see [@ERRTEXT](#)^[374] to decipher the error number if necessary).

Also see: [@IPADDRESS](#)^[394].

Example:

```
echo %@ipname[192.220.109.228]
echo %@ipname[3235671524]
echo %@ipaddress[0]
```

6.3.88 @ISALNUM

@ISALNUM[string]: Returns "1" if **string** is entirely composed of alphabetic (a-z, A-Z) and/or numeric (0 - 9) characters. Returns "0" otherwise.

Also see: [@ISALPHA](#)^[394], [@ISASCII](#)^[395], [@ISCNTRL](#)^[395], [@ISDIGIT](#)^[395], [@ISPRINT](#)^[395], [@ISPUNCT](#)^[396], [@ISSPACE](#)^[396], [@ISXDIGIT](#)^[396].

Example:

```
echo %@isalnum[123abc]
echo %@isalnum[123 abc]
echo %@isalnum[123.456]
```

6.3.89 @ISALPHA

@ISALPHA[string]: Returns "1" if **string** is entirely composed of alphabetic (a-z, A-Z) characters. Returns "0" otherwise.

Also see: [@ISALNUM](#)^[394], [@ISASCII](#)^[395], [@ISCNTRL](#)^[395], [@ISDIGIT](#)^[395], [@ISPRINT](#)^[395], [@ISPUNCT](#)^[396], [@ISSPACE](#)^[396], [@ISXDIGIT](#)^[396].

Example:

```
echo %@isalnum[abc]
echo %@isalnum[ABC]
echo %@isalnum[A B C]
```

6.3.90 @ISASCII

@ISASCII[*string*]: Returns "1" if ***string*** is entirely composed of 7-bit ASCII characters (0x00 – 0x7F). Returns "0" otherwise.

Also see: [@ISALNUM](#)^[394], [@ISALPHA](#)^[394], [@ISCNTRL](#)^[395], [@ISDIGIT](#)^[395], [@ISPRINT](#)^[395], [@ISPUNCT](#)^[396], [@ISSPACE](#)^[396], [@ISXDIGIT](#)^[396].

Example:

```
echo %@isascii[abc]
echo %@isascii[abc 123]
echo %@isascii["abc",123]
```

6.3.91 @ISCNTRL

@ISCNTRL[*string*]: Returns "1" if ***string*** is entirely composed of ASCII control characters (0x00 – 0x1F or 0x7F). Returns "0" otherwise.

Also see: [@ISALNUM](#)^[394], [@ISALPHA](#)^[394], [@ISASCII](#)^[395], [@ISDIGIT](#)^[395], [@ISPRINT](#)^[395], [@ISPUNCT](#)^[396], [@ISSPACE](#)^[396], [@ISXDIGIT](#)^[396].

6.3.92 @ISDIGIT

@ISDIGIT[*string*]: Returns "1" if ***string*** is entirely composed of decimal digits (0– 9). Returns "0" otherwise.

Also see: [@ISALNUM](#)^[394], [@ISALPHA](#)^[394], [@ISASCII](#)^[395], [@ISCNTRL](#)^[395], [@ISPRINT](#)^[395], [@ISPUNCT](#)^[396], [@ISSPACE](#)^[396], [@ISXDIGIT](#)^[396].

Example:

```
echo %@isdigit[0]
echo %@isascii[123.456]
echo %@isascii[-123]
```

6.3.93 @ISPRINT

@ISPRINT[*string*]: Returns "1" if ***string*** is entirely composed of printable characters. Returns "0" otherwise.

Also see: [@ISALNUM](#)^[394], [@ISALPHA](#)^[394], [@ISASCII](#)^[395], [@ISCNTRL](#)^[395], [@ISDIGIT](#)^[395], [@ISPUNCT](#)^[396], [@ISSPACE](#)^[396], [@ISXDIGIT](#)^[396].

6.3.94 @ISPUNCT

@ISPUNCT[*string*]: Returns "1" if *string* is entirely composed of punctuation, i.e. printable characters which are not alphanumeric or space. Returns "0" otherwise.

Also see: [@ISALNUM](#)^[394], [@ISALPHA](#)^[394], [@ISASCII](#)^[395], [@ISCNTRL](#)^[395], [@ISDIGIT](#)^[395], [@ISPRINT](#)^[395], [@ISSPACE](#)^[396], [@ISXDIGIT](#)^[396].

Example:

```
echo %@ispunct[.]
echo %@ispunct[+]
echo %@ispunct[:-)]
```

6.3.95 @ISSPACE

@ISSPACE[*string*]: Returns "1" if *string* is entirely composed of white space characters (0x09 – 0x0D or 0x20). Returns "0" otherwise.

Also see: [@ISALNUM](#)^[394], [@ISALPHA](#)^[394], [@ISASCII](#)^[395], [@ISCNTRL](#)^[395], [@ISDIGIT](#)^[395], [@ISPRINT](#)^[395], [@ISPUNCT](#)^[396], [@ISXDIGIT](#)^[396].

Example:

```
echo %@isdigit[0]
echo %@isascii[123.456]
echo %@isascii[-123]
```

6.3.96 @ISXDIGIT

@ISXDIGIT[*string*]: Returns "1" if *string* is entirely composed of hexadecimal digits (0–9,A-F, a-f). Returns "0" otherwise.

Also see: [@ISALNUM](#)^[394], [@ISALPHA](#)^[394], [@ISASCII](#)^[395], [@ISCNTRL](#)^[395], [@ISDIGIT](#)^[395], [@ISPRINT](#)^[395], [@ISPUNCT](#)^[396], [@ISSPACE](#)^[396].

Example:

```
echo %@isxdigit[0]
echo %@isxdigit[7F]
echo %@isxdigit[0x7F]
```

6.3.97 @LABEL

@LABEL[d:]: Returns the volume label of the specified disk drive. The drive letter must be followed by a colon.

Examples:

```
echo %@label[C:]
echo %@label[%_disk:]
```

See also: [VOL](#)^[314].

6.3.98 @LEFT

@LEFT[*n*,*string*]: If *n* is positive, it returns the leftmost *n* characters of *string*. If *n* is greater than the length of *string*, it returns the entire *string*. If *n* is **negative**, it returns *string* after *dropping* its rightmost *-n* characters, unless *-n* is greater than the length of *string*, in which case it returns an empty string.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). To use hexadecimal form for a negative *n*, remember to use 32-bit 2's complement arithmetic.

Examples:

```
%@LEFT[2,jpsoft]      jp
%@LEFT[22,jpsoft]     jpsoft
%@LEFT[-2,jpsoft]      jpso
%@LEFT[-22,jpsoft]     empty string
```

6.3.99 @LEN

@LEN[*string*]: Returns the length of a string.

Examples:

```
echo %@len[this is a test]
echo %@len[%comspec]
```

6.3.100 @LFN

@LFN[*filename*]: Returns the long filename for a short ("8.3") *filename*. 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. If *filename* does not refer to an actual file, the results are unpredictable.

On an LFN drive, the returned name 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. See the notes under [Variable Functions](#)^[357] for additional details.

Examples:

```
echo %@lfn[foo.bar]
echo "%@lfn[c:\progra~1]"
```

6.3.101 @LINE

@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 **filename** must be in quotes if it contains white space or special characters.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

@LINE works with files having lines of no more than 8,191 characters. Longer lines will not be counted accurately.

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](#)^[210] command, or [@FILEOPEN](#)^[381] and a sequence of [@FILEREADS](#).^[382]

You can retrieve input from standard input if you specify CON as the filename. If you are [redirecting](#)^[61] input to @LINE using this feature, you must use [command grouping](#)^[27] or the redirection will not work properly (you can [pipe](#)^[64] to @LINE without a command group; this restriction applies only to input redirection). For example:

```
(echo %@line[con,0]) < myfile.dat
```

@LINE can retrieve data from file streams on NTFS drives under Windows 2000 and above if the stream name is specified. See [NTFS File Streams](#)^[444] for additional details on file streams.

6.3.102 @LINES

@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]].

The **filename** must be in quotes if it contains white space or special characters.

@LINES works with files having lines of no more than 8,191 characters; longer lines will not be counted accurately.

@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.

@LINES can count lines in file streams on NTFS drives under Windows 2000 and above if the stream name is specified. See [NTFS File Streams](#)^[444] for additional details on file streams.

6.3.103 @LOWER

@LOWER[string]: Returns the **string** converted to lower case.

Examples:

```
echo %@lower[ThIS iSS aTeSt]
```

```
echo %@lower[%path]
```

6.3.104 @LTRIM

@LTRIM[*string1*,*string2*]: - Returns ***string2*** with all the leading characters in ***string1*** removed. ***String1*** must be enclosed in quote marks if it contains any spaces, tabs, or commas.

Examples:

```
echo "%@ltrim[JP,JP Software]"
" Software"
```

6.3.105 @MAKEAGE

@MAKEAGE[*date*,*time*]: Returns ***date*** and ***time*** (if included) as a single value in the same **FILETIME** format as **@FILEAGE**^[380], the time elapsed since 1601-01-01 00:00:00 (local time) with 100 ns resolution in a 64-bit integer. **Note:** Previous 4NT and Take Command versions and the current 4DOS version use a different representation and resolution for this function. **@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!
```

@AGEDATE^[364] is the inverse of this function.

Examples:

```
echo %@makeage[%_date]
echo %@makeage[%_date,%_time]
```

See also: [Time Stamps](#)^[443], [@FILEAGE](#)^[380], [@AGEDATE](#)^[364].

6.3.106 @MAKEDATE

@MAKEDATE[*n*,*d*]: Returns a date (formatted according to the current country settings). ***n*** is the number of days since January 1, 1980. This is the inverse of [@DATE](#)^[369]. The optional second argument specifies the date format:

<u>Code</u>	<u>Date format</u>
0	system default
1	USA (mm/dd/yy)
2	European (dd/mm/yy)
3	Japan (yy/mm/dd)
4	ISO 9601 (yyyy-mm-dd)

Examples:

```
echo %@makedate[7924]
echo %@makedate[7924,4]
```


6.3.107 @MAKETIME

@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](#)^[408].

Examples:

```
echo %@maketime[45240]
echo %@maketime[7924,4]
```

6.3.108 @MAX

@MAX[a,b,c,...]: Returns the largest integer in the list. All elements in the list must be integers.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Example:

```
echo %@max[1,5,2,0,-1]
```

6.3.109 @MD5

@MD5[[s|f],[string|filename]]: Returns the MD5 hash for a file or string. The first parameter (*s* or *f*) determines if the second parameter should be treated as a string ("s") or as a file name ("f"). Note that any leading or trailing whitespace will NOT be removed from a string argument before computing the hash. @MD5 returns -1 if the specified file couldn't be opened.

Also see [@CRC32](#)^[368].

6.3.110 @MIN

@MIN[a,b,c,...]: Returns the smallest integer in the list. All elements in the list must be integers.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Example:

```
echo %@min[1,5,2,0,-1]
```

6.3.111 @MONTH

@MONTH[date]: Returns the month number for the specified date (1-12). See [date formats](#)^[363] for information on acceptable argument formats.

Examples:

```
echo %@month[01-01-1980]
echo %@month[%_date]
```

6.3.112 @NAME

@NAME[filename]: Returns the base name of a file, without the path or extension.

The **filename** must be in quotes if it contains white space or special characters. On an LFN drive, the returned filename 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. See the notes under [Variable Functions](#)^[357] for additional details.

Note: The @NAME function makes no assumption about the *existence* of a file or directory. Its **filename** argument can be any string and the function will attempt to extract from it a "base name".

Examples:

```
echo %@name[foo.bar]
echo "%@name[%_comspec]"
```

6.3.113 @NUMERIC

@NUMERIC[["+|-"]string]: Returns **1** if **string** is composed entirely of decimal digits (0 to 9), signs (+ or -), and the thousands ([ThousandsChar](#)^[117]) and decimal ([DecimalChar](#)^[102]) separators. Otherwise it 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.

Examples:

```
echo %@numeric[12345]
echo %@numeric[-12345]
echo %@numeric[.12345]
echo %@numeric[$12.34]
echo %@numeric[5.00.125]
```

6.3.114 @OPTION

@OPTION[string]: returns the current value of the requested [.INI file](#)^[89] directive. All directives which can be altered dynamically are supported except for key mapping directives.

For [Configuration Directives](#)^[97], the current value returned may not match that stored in the .INI file.

For [Color Directives](#)^[113], the current value is returned as a single number (0-255) combining foreground and background specifications. See [Colors, Color Names & Codes](#)^[461] for details.

Examples:

```
echo %@option[passiveftp]
echo %@option[stdcolors]
```

6.3.115 @PATH

@PATH[filename]: Returns the path portion of *filename*, if present, including the drive letter and a trailing backslash but not including the base name or extension. If the *filename* argument doesn't contain path information, you may expand it first with the [@FULL](#)^[388] function.

The *filename* must be in quotes if it contains white space or special characters. On an LFN or NTFS drive, the returned filename 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. See the notes under [Variable Functions](#)^[357] for additional details.

Note: The @PATH function makes no assumption about the *existence* of a file or directory. Its *filename* argument can be any string, and the function will attempt to remove from it a "base name".

Examples:

command	result
echo "%@path["c:\program files\foo.bar"]"	"c:\program files\"
echo "%@path[foo.bar]"	" "
echo "%@path[%_comspec]"	"c:\jpssoft\rlsdu\"

6.3.116 @PING

@PING[host[,timeout[,packetsize]]]: Returns the response time in milliseconds for the specified host. *Host* is the IP address or name, *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. The *timeout* defaults to 5 seconds, and *packetsize* defaults to 64 bytes. The minimum packet size is 8 bytes and the maximum is 2048 bytes.

A negative value indicates an error. If the request times out, @PING returns **-1**. An unreachable host returns a **-2**. An invalid address returns a **-3**.

Examples:

```
echo %@ping[microsoft.com]
echo %@ping[microsoft.com,10]
echo %@ping[microsoft.com,,16]
echo %@ping[192.168.01,2,512]
```

6.3.117 @RANDOM

@RANDOM[min, max]: Returns a pseudo random value between *min* and *max*, inclusive. *Min*, *max*, and the returned value are all integers. The random number generator is initialized from the system clock the first time it is used after the command processor starts and will therefore produce a different sequence of numbers each time you use it.

Examples:

```
echo %@random[0,1]
echo %@random[-10,10]
```

6.3.118 @READSCR

@READSCR[*row,col,length*]: Returns the text displayed in the command processor window at the specified location. The upper left corner of the window is location 0,0. The **row** and **column** can 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.

6.3.119 @READY

@READY[d:]: Returns "1" if the specified drive is ready; otherwise returns "0". The drive letter must be followed by a colon.

Examples:

```
echo %@ready[E:]
echo %@ready[%_boot:]
```

6.3.120 @REGCREATE

@REGCREATE[HKEY...\subkey]: Create a new registry subkey. The argument starts with the root key, which can be abbreviated:

Full root key	Short
HKEY_CLASSES_ROOT	HKCR
HKEY_CURRENT_USER	HKCU
HKEY_LOCAL_MACHINE	HKLM
HKEY_USERS	HKU
HKEY_CURRENT_CONFIG	HKCC

The remainder of the argument (after the backslash) specifies the new subkey. The entire name must be quoted if it contains any white space 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 the Windows error number if an error occurred.

Also see [@REGQUERY](#)^[404] (read a value), [@REGSET](#)^[404] (write a value), and [@REGSETENV](#)^[404] (write and broadcast a value).

6.3.121 @REGDELKEY

@REGDELKEY[HKEY...\key]: Deletes the specified **key** and all of its **subkeys**. Returns 1 if the key was deleted, 0 otherwise.

Note: use EXTREME caution with this function. It has the potential for causing irreparable damage to your registry and preventing Windows from booting!

See [@REGCREATE](#)^[403] for information on the format of the key name.

6.3.122 @REGEXIST

@REGEXIST[HKEY...\key]: Returns **1** if the specified key exists, **0** otherwise

Note: Remember to use quotes around any entry containing spaces!

See [@REGCREATE](#)^[403] for information on the format of the key name.

6.3.123 @REGQUERY

@REGQUERY[HKEY...\subkey\value]: Read a value from the registry. REGQUERY supports keys of type REG_DWORD, REG_QWORD, REG_EXPAND_SZ, REG_SZ, REG_DWORD_LITTLE_ENDIAN, and REG_QWORD_LITTLE_ENDIAN. If the key is of type REG_EXPAND_SZ, the value is returned without further expansion. If the value name does not exist, the function returns **-1**. If the value name is not supplied, REGQUERY returns the unnamed value for the specified key (the first value with a NULL name). To retrieve an unnamed value, add a trailing \ to the name.

Note: Remember to use quotes around any entry containing spaces or commas!

See [@REGCREATE](#)^[403] (create a subkey) for information on the format of the key name. Also see [@REGSET](#)^[404] (write a value) and [@REGSETENV](#)^[404] (write and broadcast a value).

6.3.124 @REGSET

@REGSET[HKEY...\subkey\value,type,data]: Write a value to the registry. REGSET supports keys of type REG_DWORD, REG_SZ, REG_EXPAND_SZ, and REG_DWORD_LITTLE_ENDIAN. "Type" is the value type (REG_DWORD, 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 the Windows error number if an error occurred.

Note: Remember to use quotes around any entry containing spaces!

See [@REGCREATE](#)^[403] for information on the format of the key name. Also see [@REGQUERY](#)^[404] (read a value) and [@REGSETENV](#)^[404] (write and broadcast a value).

6.3.125 @REGSETENV

@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 The Windows error number if an error occurred.

Note: Remember to use quotes around any entry containing spaces or commas!

See [@REGCREATE](#)^[403] for information on the format of the key name. Also see [@REGQUERY](#)^[404] (read a value) and [@REGSET](#)^[404] (write a value).

6.3.126 @REMOTE

@REMOTE[d:]: Returns **"1"** if the specified drive is a remote (network) drive; otherwise returns **"0"**. The drive letter must be followed by a colon.

Examples:

```
echo %@remote[e:]
echo %@remote[%_disk]
```

6.3.127 @REMOVABLE

@REMOVABLE[d:]: Returns "1" if the specified drive is removable (e.g. floppy disk, removable hard disk, USB storage device, etc.), "0" otherwise. The drive letter must be followed by a colon.

Examples:

```
echo %@removable[e:]
echo %@removable[%_disk]
```

6.3.128 @REPEAT

@REPEAT[char,count]: Returns the character *char* repeated *count* times, where *count* may not exceed 8,191.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Examples:

<i>function</i>	<i>value</i>
%@repeat[%@char[95],10]	_____
7%@repeat[,7]spaces	7 spaces
%@repeat[x,10]	xxxxxxxxxx

6.3.129 @REPLACE

@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.

Examples:

```
echo %@replace[\\,/,,"ftp:\\server\\etc"]
echo %@replace[%=,,,A better, command processor]
```

6.3.130 @REXX

@REXX[[:=]expr]: Calls the REXX interpreter to execute the *expression*. Returns the numeric code or string result from REXX. Console output from the REXX interpreter is suppressed while executing the expression. Note that the command processor expands variables and functions before passing *expr* to REXX.

Examples:

```
echo %@rexx[ = 3 * 4 ]
set myprog=d:\path\foo.exe
echo %@rexx[ address(%@name[%myprog]); return address() ]
```

Note: This function *requires* that a recognized REXX interpreter be installed and properly configured. See [REXX Support](#)^[334] for more information on the REXX language.

6.3.131 @RIGHT

@RIGHT[*n*,*string*]: If *n* is positive, it returns the rightmost *n* characters of *string*. If *n* is greater than the length of *string*, it returns the entire *string*. If *n* is **negative**, it returns *string* after *dropping* its leftmost *-n* characters, unless *-n* is greater than the length of *string*, in which case it returns an empty string.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). To use hexadecimal form for a negative *n*, remember to use 32-bit 2's complement arithmetic.

Examples:

<i>function</i>	<i>value</i>
%@RIGHT[2, jpsoft]	ft
%@RIGHT[22, jpsoft]	jpsoft
%@RIGHT[-2, jpsoft]	soft
%@RIGHT[-22, jpsoft]	empty string

6.3.132 @RTRIM

@RTRIM[*string1*,*string2*]: - Returns *string2* with all the trailing characters in *string1* removed. *String1* must be enclosed in quote marks if it contains any spaces, tabs, or commas.

Examples:

```
echo "%@rtrim[985NTPX,Windows 98]"
"Windows "
```

6.3.133 @SEARCH

@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 [Executable Files and File Searches](#)^[458] 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.) 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](#)^[36] 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).

The *filename* and *path* must be in quotes if they contain white space or special characters.

Examples:

```
echo %@search[notepad]
echo %@search[msv*.dll,"d:\my dir\"]
```

6.3.134 @SELECT

@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 standard popup window navigation keystrokes, including character matching (see [Popup Windows](#)^[464] for details; to change the navigation keys see [Key Mapping directives](#)^[114]). @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**. To select from lines in the Windows clipboard, use CLIP: as the **filename**. If you use the optional **1** argument after the window title, the list will be sorted alphabetically.

Examples:

```
call %@select["d:\path\my menu.txt",5,10,5,30,Select an option]
help %@word["=",0,%@select[d:\path\4nt.ini,0,0,10,30,Select a
directive]]
```

6.3.135 @SFN

@SFN[filename]: Returns the fully expanded short ("8.3") filename for a long **filename**. 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. If **filename** does not refer to an actual file, the results are unpredictable.

Examples:

```
echo %@sfn["c:\program files\foo.bar"]
echo %@sfn[%_comspec]
```

6.3.136 @STRIP

@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**.

Example:

```
echo %@strip[-:,%_isodate%%_time]
```

6.3.137 @SUBSTR

@SUBSTR[string,start,length]: An older version of [@INSTR](#)^[393]. If the **length** is omitted, it will default to the remainder of **string**. If **string** includes commas, it must be quoted with quote marks ["] or back-quotes [^], or each comma must be preceded by an [EscapeChar](#)^[103]. The quotes **count** in calculating the position of the substring. @INSTR, which has **string** as its last argument, does not

have this restriction.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Examples:

```
echo %@substr[this is useful,8]
echo %@substr[this is useful,8,-2]
echo %@substr["commas, they DO matter",9]
echo %@substr[commas%=, they DO matter,9]
```

6.3.138 @SUBST

@SUBST[n, string1, string2]: Substitutes **string1** starting at position **n** in **string2**.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

6.3.139 @TIME

@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.

Examples:

```
echo %@time[12:34:56]
echo %@time[%_time]
```

6.3.140 @TIMER

@TIMER[n]: Returns the current split time for a stopwatch started with the **TIMER**³⁰³ command. The value of **n** specifies the timer to read and can be 1, 2, or 3.

6.3.141 @TRIM

@TRIM[string]: Returns the string with the leading and trailing white space (space and tab characters) removed.

6.3.142 @TRUENAME

@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.

The **filename** must be in quotes if it contains white space or special characters.

6.3.143 @TRUNCATE

@TRUNCATE[handle]: truncate the file opened for **write access** by @FILEOPEN at the current position of the file pointer, where **handle** is the value returned by @FILEOPEN.

See also: [@FILEOPEN](#)^[381], [@FILEREAD](#)^[382], [@FILEWRITE](#)^[385] / [B](#)^[385], [@FILESEEK](#)^[383] / [L](#)^[383].

6.3.144 @UNC

@UNC[file]: Returns the UNC name for the specified file (or an error if the file has no UNC, e.g., a local file).

6.3.145 @UNICODE

@UNICODE[c]: (Unicode products) Returns the Unicode character code of the specified character(s) **c** as a numeric string. For example, "@unicode[A]" returns "65". If you enter more than one character, @UNICODE returns the numeric value for each character in a space delimited string, e.g., "%@unicode[abc]" returns "97 98 99". You can put an [EscapeChar](#)^[103] before the actual character to allow quotes and other special characters in the argument (e.g., "%@unicode[%=`]").

See also: [@ASCII](#)^[365], [ASCII & Unicode Versions](#)^[422].

6.3.146 @UNIQUE

@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 regardless of the type of drive on which the file is created. This function allows you to create a temporary file without overwriting an existing file.

The **path** must be in quotes if it contains white space or special characters.

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](#)^[175], DELAY /M, or [BEEP](#)^[156] (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.

6.3.147 @UPPER

@UPPER[string]: Returns the **string** converted to upper case.

6.3.148 @VERINFO

@VERINFO[filename[,info[,language]]]: Returns the version information for the specified file. The optional second parameter specifies the desired information and defaults to "FileVersion". The optional third parameter specifies the language/codepage pair (in hex). If that parameter is omitted, the code page for the default user language is assumed. If the requested information field is not provided in the specified file, returns a null string.

For example, the current 4NT.EXE returns values for:

```
CompanyName
FileDescription
FileVersion
InternalName
LegalCopyright
LegalTrademarks
OriginalFilename
ProductName
ProductVersion
```

such as:

```
echo %@verinfo[4nt.exe,companyname,040904E4]
```

Note: Most executables under Windows contain a "FileVersion" field, but not all do. The number, names and contents of the specific Information Fields and language/codepage pairs provided within a given application can potentially be *anything* the programmer decided to use.

6.3.149 @WATTRIB

@WATTRIB[filename[,attributes[,p]]]: This function is similar to [@ATTRIB](#)^[365], but supports file selection based on the following extended attributes available on NTFS volumes.

E	Encrypted
N	Normal
T	Temporary
P	Sparse file
J	Junction (reparse point)
C	Compressed
O	Offline
I	Not content-indexed

Also see [Attributes Switches](#)^[39] and the [ATTRIB](#)^[146] command

6.3.150 @WILD

@WILD[string1,string2]: Compares 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](#)^[36], but the first argument (**string1**) may not. The test is not case sensitive. This function is typically used to check if a string (**string1**, usually a variable reference) matches a specific pattern (**string2**).

Assuming the PATH variable contains:

```
c:\windows;c:\windows\system32;c:\program files\util;d:\jpsoft
```

the following are some examples of use:

%@wild[%path,*\UTIL*]	looking for string "util" anywhere, returns "1"
%@wild[%path,*c]	looking for a string ending with "c" returns "0"
%@wild[%path,*t]	looking for a string ending with "t" returns "1"
%@wild[%path,c*]	looking for a string starting with "c" returns "1"
%@wild[%path,t*]	looking for a string starting with "t" returns "0"
%@wild[%path,*c*]	looking for a string containing a "c" returns "1"
%@wild[%path,*t*]	looking for a string containing a "t" returns "1"
%@wild[%path,*b*]	looking for a string containing a "b" returns "0"

A lone "?" wildcard matches any single character. For example,

%@wild[foo,?]	returns "0" (too many characters, no match)
%@wild[x,?]	returns "1" (one character, matches the pattern)

Finally,

`%@wild[%path,c?*]` looks for a leading "c", followed by any one character, followed by 0 or more characters.

`%@wild[%path,c*?]` looks for a leading "c", followed by zero or more characters, followed by any one character.

See the [Extended Wildcards](#)^[36] topic for additional information on pattern matching.

6.3.151 @WINCLASS

@WINCLASS[classname]: Returns the window title of the first window with the specified class name.

6.3.152 @WINEXENAME

@WINEXENAME[title]: Returns the executable name for the first window matching the specified title (which can include [wildcards](#)^[36]).

6.3.153 @WININFO

@WININFO[n]: Returns information about the current system. *n* is a number specifying what information to return:

<i>n</i>	Information returned
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 486 i486 586 Pentium:
5	Processor level (Windows NT / 2000 / XP / 2003 only)
6	Processor revision (Windows NT / 2000 / XP / 2003 only)
7	page size, bytes
8	virtual memory allocation granularity, bytes

6.3.154 @WINMEMORY

@WINMEMORY[n]: Returns the requested Windows memory information. All values except memory load are returned in bytes. *n* is a number specifying what to return:

<i>n</i>	<i>Information returned</i>
0	Memory load, %
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

6.3.155 @WINMETRICS

@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.

Note: This function provides direct access to the ***GetSystemMetrics*** API. Not all available parameters are listed here and your Windows configuration may support additional parameters. See your Windows technical documentation for details.

<i>n</i>	<i>Information returned</i>
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	Is Pen Windows installed? 0 no 1 yes
42	Is DBCS version of USER.EXE installed? 0 no 1 yes
43	Number of buttons on mouse
44	(Win9x) Is security present? 0 no 1 yes
63	(Win9x) Is network is installed? 0 no 1 yes
67	(Win9x) How system was started 0 normal 1 fail-safe 2 fail-safe with 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	Is Windows set up for Arabic/Hebrew? 0 no 1 yes
75	Mouse has a wheel 0 no 1 yes
76	(Win98/ME, W2000/XP/2003) Coordinate of left side of virtual screen
77	(Win98/ME, W2000/XP/2003) Coordinate of top of virtual screen
78	(Win98/ME, W2000/XP/2003) Width in pixels of virtual screen
79	(Win98/ME, W2000/XP/2003) Height in pixels of virtual screen
80	(Win98/ME, W2000/XP/2003) Number of monitors on desktop

6.3.156 @WINSTATE

@WINSTATE[title]: Returns the window state of the first window with a matching title (which can include [wildcards](#)^[36]). The return values are:

<u>Value</u>	<u>Window state</u>
0	Hidden
1	Normal
2	Minimized
3	Maximized

6.3.157 @WINSYSTEM

@WINSYSTEM[n,v]: Sets or returns the value of the requested Windows system-wide parameters.

To retrieve a parameter, the format is **%@winsystem[n]** where "n" is the appropriate "GET" number from the table below.

To set a parameter, the format is **%@winsystem[n,v]** where "n" is the appropriate "SET" number from the table below and "v" is the desired new value for that parameter. The GET and SET numbers may be different or identical, depending on the specific parameter.

Where the selection is a state, the legal values are **0** for off/disabled, and **1** for on/disabled.

Where the selection is a width or height, the values are in pixels.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits).

Note: This function provides direct access to the **SystemParametersInfo** API. Not all available parameters are listed here and your Windows configuration may support additional parameters. See your Windows technical documentation for details, and use with caution.

GET	SET	Parameter to GET or SET
-----	-----	-------------------------

1	2	beep state
5	6	border width
10	11	keyboard repeat speed (0 to 31)
13	13	width of an icon cell
14	15	screen saver time-out (seconds)
16	17	state of the screen saver
22	23	keyboard repeat delay setting (0-3).
24	24	height of an icon cell
25	26	icon title wrapping state
27	28	pop-up menu alignment
37	38	full-window dragging state
56	57	Show Sounds accessibility flag
68	69	keyboard preference state (0=mouse, 1=keyboard)
70	71	screen reviewer utility state
74	75	font smoothing feature state
79	81	time-out for the low-power phase of screen saving (seconds)
80	82	time-out value for the power-off phase of screen saving (seconds)
83	85	low-power phase of screen saving state
84	86	power-off phase of screen saving state
89	90	locale identifier for the system default input language.
93	94	Mouse Trails feature state. (0 or 1= disabled, >1= number of cursors in the trail)
95	95	snap-to-default-button feature state
98	99	width of the mouse pointer WM_MOUSEHOVER message trigger rectangle
100	101	height of the mouse pointer WM_MOUSEHOVER message trigger rectangle
102	103	time in the hover rectangle for the mouse pointer to trigger a WM_MOUSEHOVER message (milliseconds)
104	105	number of lines to scroll when the mouse wheel is rotated
106	107	time that the system waits before displaying a shortcut menu when the mouse cursor is over a submenu item (milliseconds)
110	111	IME status window state - per user (0=invisible, 1=visible)
112	113	current mouse speed (1 to 20).

4096	4097	active window tracking state
4098	4099	menu animation feature state.
4100	4101	combo box animation state.
4102	4103	list box smooth-scrolling effect state.
4104	4105	gradient effect for window title bars.
4106	4107	menu access keys underline state.
4108	4109	active window tracking Z-order state.
4110	4111	hot-tracking state.
4114	4115	(W2000, XP, 2003) menu fade animation state.
4116	4117	(W2000, XP, 2003) selection fade effect state.
4118	4119	(W2000, XP, 2003) ToolTip animation state.
4120	4121	(W2000, XP, 2003) type of ToolTip animation (1 for fade, 0 for slide)
4122	4123	(W2000, XP, 2003) Get the cursor shadow state.
4124	4125	(WinMe, XP, 2003) state of the Mouse Sonar feature
4126	4127	(WinMe, XP, 2003) mouse clicklock state
4128	4129	(WinMe, XP, 2003) mouse vanish feature state
4130	4131	(XP, 2003) whether native User menus have flat menu appearance.
4132	4133	(XP, 2003) drop shadow effect state.
4158	4159	(W2000, XP, 2003) state of all UI effects.
8192	8193	time following user input during which the system will not allow applications to force themselves into the foreground (milliseconds)
8194	8195	active window tracking delay (milliseconds)
8196	8197	the number of times SetForegroundWindow will flash the taskbar button when rejecting a foreground switch request.
8198	8199	(W2000, XP, 2003) caret width in edit controls
8200	8201	(WinMe, XP, 2003) time delay before the primary mouse button is locked.
8202	8203	(XP, 2003) type of font smoothing (32769=standard anti-aliasing, 32770=ClearType).
8204	8205	(XP, 2003) contrast value used in ClearType smoothing (1000-2200)
8206	8207	(XP, 2003) width of the left and right edges of the focus rectangle
8208	8209	(XP, 2003) height of the top and bottom edges of the focus rectangle

GET	SET	Parameter to GET or SET
-----	-----	-------------------------

6.3.158 @WORD

@WORD [["xxx",], *n*, *string*]: Returns the *n*th word in *string*. The first word is numbered 0. If *n* is **negative** or it is preceded by a hyphen (minus sign), words are counted backwards from the end of *string*, and the absolute value of *n* is used. You can specify the rightmost word by setting *n* to -0.

@FIELD^[379], **@FIELDS**^[379], **@WORD**^[416] and **@WORDS**^[417] default to spaces, tabs, and commas as

separators. You can use the optional first argument, "**xxx**", to specify the separators that you wish to use. If you want to use a quote mark as a separator, prefix it with an [escape character](#)^[103]. If **string** is enclosed in quotation marks, you **must** specify the list of delimiters.

[@FIELD](#)^[379] and [@FIELDS](#)^[379] differ from [@WORD](#)^[416] and [@WORDS](#)^[417] in how multiple consecutive separators are counted. [@WORD](#)^[416] and [@WORDS](#)^[417] consider a sequence as a single separator. In contrast, [@FIELD](#)^[379] and [@FIELDS](#)^[379] count each occurrence of a separator individually, including counting individually each of the adjacent separators, which *allows you to rprocess empty fields in string*.

Numeric input may be entered in either decimal format (series of digits 0-9) or in hexadecimal format ("0x" followed by a sequence of 0-F hex digits). To use hexadecimal form for a negative **n**, remember to use 32-bit 2's complement arithmetic. You cannot use the hexadecimal form to specify word -0 (the rightmost word).

See also: [@WORDS](#)^[417], [@FIELD](#)^[379], [@FIELDS](#)^[379].

Examples:

<i>function</i>	<i>value</i>
%@WORD[2,NOW IS THE TIME]	THE
%@WORD[-0,NOW IS THE TIME]	TIME
%@WORD[-2,NOW IS THE TIME]	IS
%@WORD["=",1,2 + 2=4]	4

6.3.159 @WORDS

[@WORDS](#)["**xxx**" , **string**]: Returns the number of words in **string**.

[@FIELD](#)^[379], [@FIELDS](#)^[379], [@WORD](#)^[416] and [@WORDS](#)^[417] default to spaces, tabs, and commas as separators. You can use the optional first argument, "**xxx**", to specify the separators that you wish to use. If you want to use a quote mark as a separator, prefix it with an [escape character](#)^[103]. If **string** is enclosed in quotation marks, you **must** specify the list of delimiters.

[@FIELD](#)^[379] and [@FIELDS](#)^[379] differ from [@WORD](#)^[416] and [@WORDS](#)^[417] in how multiple consecutive separators are counted. [@WORD](#)^[416] and [@WORDS](#)^[417] consider a sequence as a single separator. In contrast, [@FIELD](#)^[379] and [@FIELDS](#)^[379] count each occurrence of a separator individually, including counting individually each of the adjacent separators, which *allows you to rprocess empty fields in string*.

See also: [@WORD](#)^[416], [@FIELD](#)^[379], [@FIELDS](#)^[379].

6.3.160 @WORKGROUP

[@WORKGROUP](#)[**name**]: Returns the workgroup of the computer specified by the DNS or NetBios **name**. If **name** is not specified, returns the workgroup of the local computer.

6.3.161 @YEAR

[@YEAR](#)[**date**]: Returns the year for the specified date. See [date formats](#)^[363] for information on acceptable argument formats.

7 Setup & Troubleshooting

- ▶ [Registration](#)^[418]
- ▶ [Troubleshooting Service and Support](#)^[418]
- ▶ [Supported Platforms](#)^[421]
- ▶ [ASCII and Unicode Versions](#)^[422]
- ▶ [Help File](#)^[423]
- ▶ [Error Messages](#)^[424]

7.1 Registration

There are no separate *trial* and **registered** versions of our products. Without registration, a trial version is fully functional for 30 days.

At any time you can apply your current personal registration information to a *trial* version in order to turn it into a **registered** product that bears your name and never expires. Use the command [VER /R](#)^[313] from the prompt to verify the status of the copy you are currently running. Under Take Command, you can also view the [Help/About](#)^[71] menu entry.

When you register, we email to the address you provided a small file (250 bytes or so) named **4NT.KEY** (for **4NT**) or **TCMD32.KEY** (for **Take Command**) which contains your personal registration data. That information must be entered into your Windows registry. Under most configurations, you merely need to double-click on the file's icon. On some systems, you may need to right-click on that icon and select "Install" or "Merge" from the context menu. You could also enter the information manually from any prompt by typing "regedit 4NT.KEY" or "regedit TMCD32.KEY". If you are running 4NT or Take Command at the time you load the KEY file, your registration information won't appear in that session until you exit and restart the command processor.

Remember to save your 4NT.KEY / TCMD32.KEY file in a safe place in case you need to reinstall 4NT or Take Command (such as when upgrading to a new machine). If you have lost your registration key, you can request a replacement by contacting JP Software at sales@jpsoft.com and giving us your full registered name, serial number, and any other data that might be necessary to locate your customer record. We may have to charge you a small fee for reissuing lost registration data.

Note: The email address you provide must be able to accept and deliver attachments with the *.KEY extension. If necessary, give us an alternative address to which the KEY files can be emailed.

Programs `br4nt.exe` and `brtc32.exe` used by previous versions are obsolete, and are neither required nor supported.

7.2 Troubleshooting, Service & Support

If you need help with the command processor, we encourage you to review our documentation and then contact us for assistance if required.

If you need help with sales, ordering, or shipments (including defective disks or other materials which were shipped to you), or with registration codes, please contact our Sales and Customer Service department. See [Contacting JP Software](#)^[421] for our email address, mail address, and telephone numbers. Note that Sales and Customer service staff cannot assist you with technical problems and conversely Technical Support representatives cannot answer your sales or registration questions.

If you need technical support for the command processor, review the [Technical Support](#)^[419] information section, which tells you what we need to know to provide you with accurate and timely support, then contact us via one of the methods described there. In most instances, our *Online Support Forum* is the fastest and most efficient way to address your technical questions and concerns.

7.2.1 Technical Support

Support Plans

Standard, **no-charge** support is available electronically through our **Support Forum** ([see below](#)^[419]).

We also offer a paid support option which includes automatic upgrades and support by private email or telephone.

For complete details on all support options, including plans currently offered and support terms and conditions, see our web site at <http://jpsoft.com/>.

Before you contact Technical Support, please review the "[What Information do we need?](#)"^[419] section which outlines the basic data needed to best address your questions and concerns.

Online Support

The primary venue for Technical Support is via our free online Support Forum, where our support personnel can read and respond to your messages, and other users can participate in and benefit from the exchange. The Forum is a lively community frequented by a number of experienced and helpful users. JP Software representatives read **every** Forum message and respond as promptly as reasonably possible whenever appropriate.

If you have any kind of Internet access, even if only email, chances are you **can** use the Forum which we make accessible as a mailing list, a news group, and a set of web pages. Forum Members must provide a valid email address and a full name to be able to **post**, but you do not need to join or provide any information to simply visit or search the Forum. For complete details and direct access links see the support area of our web site at <http://jpsoft.com/>.

In addition to the material you are currently reading, a number of other support resources are available from our web site, including documentation files, technical tips and discussions, other technical information, and links to other companies' sites. We update this information regularly, and we encourage you to check the Technical Support area of the web site to see if the information there will address any questions you have.

If you are unable to gain access to the forum, or you need to include confidential information in your support request, contact us via email at **support@jpsoft.com** ("plain text" only!) and we will assist you in resolving the problem with forum access, or assist you with your request privately if appropriate. Please do not use that address for standard support questions which can be posted on the forum. Inappropriate messages will be either automatically discarded or simply trigger a generic "*use the Forum*" auto responder.

If you are a paid support customer you should use the online Support Forum for routine questions. To create a private support incident refer to the materials sent to you with your subscription for contact information, or email **priority_support@jpsoft.com** and include your support ID (mail to this address may not be answered if it does not include a valid support ID).

Note that for security and "anti-spam" reasons, the support address filters out all non-text data (including screen shots and others) and such prohibited material will probably be lost before it ever reaches us. Technical support messages should be sent as standard ASCII text. Please do not transmit attached files, binary files, screen images, or any file over 10K bytes in size to any of our electronic technical support addresses unless asked to do so by our support staff. We will be unable

to respond to (and may not even receive) messages that do not met these basic criteria.

What Information do we need?

Before contacting us for support, please check this help file and other documentation for answers to your question. If you can't find what you need, try the Index. If you're having trouble getting the command processor to run properly, review the information on [Error Messages](#)^[424], and look through the Support Forum for any last-minute information.

If you need help with sales, ordering, shipments, registration keys, or other similar non-technical issues please contact our Sales and Customer Service department. Technical Support will not be able to assist you with those matters. Conversely, Customer Service is not equipped to answer your technical questions. See [Contacting JP Software](#)^[425] for our addresses.

Regardless of how you contact us for support, we can do a much better job of assisting you if you can give us some basic information, separate from your interpretations of or conclusions about the problem. Remember that we know NOTHING about your system or configuration unless you tell us, and we can't always make accurate guesses if you don't. The first four items listed below are essential for us to be able to understand and assist you with your problem:

- **What environment are you working in?** This includes the operating system version you are using, the version of the JP Software product involved, and related information such as network connections and the name and version number of any other software which appears to be involved in the problem. Use the [VER /R](#)^[313] command to determine the command processor version and operating system version. This item is essential! Every question posted on the Forum should include a brief identification such as "4NT 6.01U build 190 under XP+SP2" or something similar.
- **What exactly did you do?** A concise description of what steps you must take to make the problem appear is much more useful than a long analysis of what might be happening. In most cases, posting the exact command line(s) giving you trouble is the simplest approach.
- **What did you expect to happen?** Tell us the result you expected from the command or operation in question, so that we understand what you are trying to do. Something that seems "obvious" to you might not be so to others. For example, tell us "*I was expecting the file name to be in upper case*" or a similar brief explanation.
- **What actually happened?** At what point did the failure occur? If you saw an error message or other important or unusual information on the screen, what **exactly** did it say? Don't simply tell us "*it didn't work*". For example, if you were expecting output from a command and saw none, at least tell us that much.
- **Briefly, what techniques did you use to try to resolve the problem?** What results did you get? One "technique" that tends to solve many problems is to review the help for the command or feature in question and try it with the documented exact correct syntax, as opposed to some undocumented alternative you were under the impression should also "work".
- **If the problem seems related to startup and configuration issues,** what are the contents of any startup files you use (such as [TCSTART](#)^[6] or [4START](#)^[6], [TCEXIT](#)^[6] or [4EXIT](#)^[6], and the [.INI file](#)^[89]), any batch files they call, and any alias or environment variable files they load? You do *not* need to include the entire file, of course, but at least any entry you think might be relevant, such as a specific INI directive or a command.
- **Can you repeat the problem or does it occur randomly?** If it's random, does it seem related to the programs you're using when the problem occurs? Random or occasional

problems are very difficult to diagnose. Do your best to determine some sort of pattern or sequence of events that triggers the problem. If you can't reproduce it, chances are we won't be able to either. Note that mysterious unexplainable problems often permanently disappear after simply reloading the program or even rebooting the system.

- If the command processor experiences a **fatal failure**, such as a Windows "*General Protection Fault*" (GPF), "*Unhandled Exception*", or similar unrecoverable error, it will attempt to dump the relevant memory locations and other useful data to a **text file** (**4NT.GPF** or **TCMD.GPF**) into its directory if at all possible. That *.GPF file may be important for tracking down the exact internal location of the failure, whether in the command processor executable itself or (most often) in some Windows component that it happened to invoke, and Technical Support may ask you to send a copy of that text file to a specified address or post its contents on the Forum.

7.2.2 Contacting JP Software

You can contact JP Software at the following addresses and numbers. Our normal business hours are 9:00 AM to 5:00 PM weekdays, Eastern US time (except holidays).

Address: **JP Software Inc.**
P.O. Box 328
Chestertown, MD 21620
USA

Main number: **(410) 810-8818**
 Fax: **(410) 646-0026**
 Order Line: **(410) 810-8819**

Online: Web site: <http://jpsoft.com/>
 FTP site: <ftp://jpsoft.com>
 Sales / Customer Service: **sales@jpsoft.com**

Technical Support: Standard (no-charge) support: Available via our online **Support Forum**, accessible from the support area of our web site.

See [Technical Support](#)^[419] for additional details, and for information on paid support options.

Note: Our server implements aggressive "*anti-spam*" measures. Please make sure you are using the correct address with appropriate subject line and contents, else we might not receive your email message at all.

7.3 Supported Platforms

(TC) Take Command is a *win32 graphical* application **designed** to work with and **supported** under Windows 98, Windows ME, Windows NT 4.0, Windows 2000, Windows XP, and Windows 2003.

(4NT) 4NT is a *win32 console* application **designed** to run under Windows NT 4.0, Windows 2000, Windows XP, and Windows 2003. It is also **supported** under Windows 98 and ME. The subsection below covers the use of 4NT in the latter environment.

Note: see [ASCII and Unicode Versions](#)^[422].

Using 4NT Under Windows 98 and ME

This section explains the reasons for using 4NT under Windows 98 and ME instead of 4DOS, our

standard Windows 98/ME character-mode product.

In general you will find that 4NT works well under Windows 98 and ME, and offers certain advantages, but also has some limitations. The following are some of the areas where you may notice 4NT's particular benefits or limitations under Windows 98 and ME.

System startup: 4NT cannot be loaded as the DOS primary shell in *CONFIG.SYS*, so its capabilities will not be available when booting to a command prompt (without loading Windows). 4DOS can be loaded as the primary shell. This difference is irrelevant under Windows ME, where a command prompt boot option is not available.

Command line length: 4NT accepts typed commands up to 8,191 characters long, and can allow these lines to expand to up to 16,383 characters during processing. 4NT's support for much longer command lines may be important for software developers who must use long commands to set compiler environment variables and for other similar tasks.

Performance: Text output from 4NT is often somewhat slower than you may be used to from a DOS prompt (COMMAND.COM or 4DOS.COM) – for example, under Windows 98 and ME the DIR command in 4DOS will probably display a directory more quickly than in 4NT. This is due to the way Windows is designed; Windows 98 and ME produce faster output from 16-bit text mode programs like 4DOS, while Windows NT, 2000, and XP generally produce faster output from 32-bit programs like 4NT. This effect is less noticeable on faster systems, and is offset to some extent by improved 32-bit performance with other Windows APIs.

Features: 4NT offers certain features which are only available in a 32-bit program (for example, [FTP](#)^[52] support, and the [MSGBOX](#)^[251] and [QUERYBOX](#)^[265] commands). These features are not available in 16-bit command processors.

Windows Bugs: The portion of Windows 98 which provides support to 32-bit console applications like 4NT has a number of minor but annoying long-standing bugs. For example, the **Window Close** button does not work in 4NT and similar applications, and only a limited range of text window dimensions are supported. These problems are less prevalent in Windows ME.

32-bit console applications, such as 4NT, are executed under Win98 and ME by the Microsoft-provided conagent.exe ("Console Agent") program which translates 32-bit calls into 16-bit to communicate with the underlying DOS as required. If you need to alter the visual properties of your emulated 32-bit console sessions, you must change the properties for conagent.exe (generally found in the "system" directory), not 4nt.exe.

In summary, if you run Windows 98 or ME and need 4NT's features such as long command lines, FTP support, and additional commands, then it can be an excellent choice as long as you are willing to put up with slightly slower text output, and a few minor problems of integration with the Windows GUI.

Note: The "command.com" provided in NT-based operating systems is NOT equivalent to the DOS command processor bearing the same name. It is merely a wrapper that passes most all commands (except for a few like REM, SET and VER) to the underlying 32-bit command processor (CMD.EXE or 4NT.EXE).

7.4 ASCII & Unicode Versions

4NT and Take Command are available in both **ASCII** and **Unicode** versions.

The **Unicode** versions are designed for use under **Windows NT 4.0, 2000, XP, and 2003**.

The **ASCII** versions are designed for use under the DOS-based operating systems **Windows 98** and

ME.

You *can* use the ASCII versions under NT/W2000/XP (an unsupported combination!) but because of Windows limitations, they will not properly handle non-English languages and file names containing non-ASCII characters, and may not properly recognize some keystrokes. You *cannot* use the Unicode versions under Windows 98 or ME.

Both versions behave identically in general, with the major distinction that the ASCII version is limited to the 255 ASCII character set while the Unicode version can potentially handle all 65535 Unicode glyphs. Other functional differences will be noted in these help files under the appropriate topics.

The ASCII versions will identify themselves (see the **VER**^[313] command) with a trailing "A", e.g. "4NT 6.01A", while the Unicode versions use a trailing "U", e.g. "4NT 6.01U".

Note: To properly view Unicode glyphs, you must use a suitable Unicode font. In most Windows configurations, the Microsoft-provided "*Lucida Console*" will be an acceptable starting point but it only supports a limited subset of common Unicode glyphs. Some Unicode material may require a more complete font. The phrase "*Unicode support*" in Windows terminology refers to what is technically named "**UTF-16LE**" encoding.

7.5 Help File

The distribution packages for 4NT and Take Command contain **jphelp.chm**, a complete help file in Microsoft's "Compiled HTML" format designed for viewing under all supported Windows versions. That file includes description and syntax for all commands, variables and functions, as well as reference information to assist you in using the command processor and developing batch files, aliases and functions.

The same material is also available in alternative formats such as PDF. Please check our web site regularly to make sure you have the most recent version of the help files. We also provide a comprehensive set of searchable Online Web Help pages.

You can request help at the prompt from the Help menu (**TC**), by typing **HELP**^[225] (or **HELP** plus a command name), or by pressing the **F1**^[119] key at any time when the command processor is accepting keyboard input at the prompt. If you use the **HELP** command by itself you will be taken to an introductory page, but if you follow the command with a topic name (e.g. "help copy") you will see help on the requested topic if available.

If you type a command name on the line and then press **F1**^[119], the help system will provide "*context-sensitive*" help by using the first word on the line as a help topic. For example, if you press **F1** after entering each of the command lines shown below you will get the display indicated:

```
[c:\]                               Overview
[c:\] copy *.* a:                   Help on the COPY command
[c:\] c:\util\map                   "The page cannot be displayed"
```

You can use this feature to obtain help directly on any topic, not just on commands. All internal commands and most internal variables, functions and directives have their own topic, allowing you to directly query, for example, "help @eval" (help for the "@eval[]" function) or "help _disk" (help for the "_DISK" internal variable).

You can also invoke help for the word immediately above (or immediately to the left of) the cursor by pressing the **Ctrl-F1**^[120] key. This feature is especially useful when you need the syntax for a variable function.

If the topic you seek is not immediately listed, look for a suitable cross-reference from the **Index** tab or use the **Search** tab. The topics you use most often can be stored and recalled through the **Favorites** tab.

The command processor uses the default Windows help system to display the contents of the *jphelp.chm* help file. Under most configurations, Windows will remember the last used settings (window size, tab selected, etc.) for that file. Once you've started the help system with **F1**^[119] or the **HELP**^[225] command, you can use standard Windows HTML Viewer (*HH.EXE*) keystrokes to navigate. For more information, see your Windows documentation. **Note:** If you use both 4NT and Take Command, you may find it more convenient to only keep one copy of *jphelp.chm* in a common location, such as your "c:\windows\help" directory.

Quick Help:

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

```
copy /?
```

Then you will see "quick help" for that command, displayed in the command window.

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

7.6 Error Messages

This section lists error messages generated by the command processor, and includes a recommended course of action for most errors. If you are unable to resolve the problem after reviewing these help files carefully, then contact JP Software for [technical support](#)^[419].

Error messages relating to files are generally reports of errors returned by Windows. You may find some of these messages (for example, "Access denied") vague enough that they are not always helpful. The command processor includes the file name in file error messages, but is 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 Windows.

Not **all** errors potentially reported by Windows can be listed here, of course. See [Windows System Errors](#)^[465] for examples of system errors returned in the [_SYSERR](#)^[356] internal variable.

The following list includes most common error messages, in alphabetical order:

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 in a multitasking system.

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 debugging a batch file: You are attempting to invoke a nested instance of the Batch File Debugger ([BDEBUGGER](#)^[149]) while you are already in the debugger.

Already excluded files: You used more than one exclude range in a command. Combine the exclusions into a single range.

Bad disk unit: Generally caused by a disk drive hardware failure.

Batch file missing: the command processor can't find the batch (.BTM 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 directory is full.

Can't delete: the command processor can't delete the specified file or directory. The disk is probably write protected.

Can't end current process: You attempted to terminate the command processor with a [TASKEND](#)^[298] command. TASKEND can only be used to end other processes; to terminate the command processor, use the [EXIT](#)^[205] command.

Can't get directory: the command processor can't read the directory. The disk drive is probably not ready.

Can't install hook: the command processor can't install the operating system "hook" required by the [KEYSTACK](#)^[236] command. Contact JP Software for assistance.

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 query key type: The key name supplied to @REGQUERY refers to a key with a type that @REGQUERY does not support. See [@REGQUERY](#)^[404] for a list of supported key types.

Can't remove current directory: You attempted to remove the current directory, which Windows does not allow. Change to the parent directory and try again.

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 empty or not text format: You tried to retrieve some text from the Windows clipboard, but there is no text available. Correct the contents of the clipboard 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.

Command line too long: A single command or the entire command line exceeded the maximum [allowable length](#)^[25] (including during alias, variable, or function 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).

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: Windows 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]: (TC) A DDE transaction could not be completed by the [DDEEXEC](#)^[177] 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](#)^[267] or [DIRS](#)^[190] 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.

Divide by zero: The command or function you used tried to do a division by zero. If the data causing the problem is from your own input or batch file, change the input to avoid the divide by zero condition. If the data was generated internally by the command processor, contact JP Software for assistance.

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.

Error in command-line directive: You used the **//inl** option to place an **.INI** directive on the [startup](#)^[3] command line, 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](#)^[89]. 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: Windows 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: Windows 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 10 levels deep.

File Allocation Table bad: Windows 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.

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 not found: the command processor couldn't find the specified file. Check the spelling and path name.

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](#)^[89], 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](#)^[89], 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.

Input and output files must have different names: ([BATCOMP](#)^[149]) You are attempting to compress a file to itself.

Input file is already compressed: ([BATCOMP](#)^[149]) You are attempting to compress a batch file that has already been compressed.

Insufficient disk space: COPY or MOVE ran out of room on the destination drive. Remove some files and retry the operation.

Invalid batch file: The batch file is corrupted, or improperly [compressed](#)^[337] or encrypted. Retry with a new copy of the file.

Invalid character value: You gave an invalid value for a character directive in the [.INI file](#)^[89].

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](#)^[89].

Invalid color: You gave an invalid value for a color directive in the [.INI file](#)^[89].

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 the [.INI file](#)^[89].

Invalid drive: A bad or non-existent disk drive was specified.

Invalid key name: You tried to make an invalid key substitution in the [.INI file](#)^[89], or you used an invalid key name in a keystroke [alias](#)^[138] or command. Correct the error and retry the operation.

Invalid numeric value: You gave an invalid value for a numeric directive in the [.INI file](#)^[89].

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](#)^[89].

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](#)^[114] entered in the [.INI file](#)^[89]. Reduce the number of key substitutions or contact JP Software or your dealer for assistance.

Label not found: A GOTO or GOSUB referred to a non-existent label. Check your batch file.

Listbox is full: There is no more room in the Find Files / Text dialog's results box. Use a more selective search, or use the FFIND command rather than the dialog.

Missing close paren: A [KEYSTACK](#)^[236] command is missing a closing parentheses around a character group. Correct the command.

Missing ENDTEXT: A [TEXT](#)^[301] command is missing a matching ENDTEXT. Check the batch file.

Missing GOSUB: the command processor cannot perform the [RETURN](#)^[272] command in a batch file. You tried to do a RETURN without a [GOSUB](#)^[221], or your batch file has been corrupted.

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. Correct the expression and retry the operation.

No shared memory found: The SHRALIAS command could not find any global alias list, history list, or directory history list to retain, because you executed the command from a session with local lists. Start the command processor with at least one global list, then invoke SHRALIAS.

No SMTP server: [SENDMAIL](#)^[279] can't find an SMTP server. Check your INI file or mailer configuration (see SENDMAIL for additional details).

Not a directory: The name passed to [RD](#)^[267] is not a directory.

Not an alias: The specified alias is not in the alias list.

Not in environment: The specified variable is not in the environment.

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.

Out of function space: You are attempting to create a [User-defined Function](#)^[218] that would require more resources than what your system makes available. Shorten the function definition or delete functions you no longer need

Out of memory: the command processor or Windows had insufficient memory to execute the last command. Try to free some memory by closing other sessions. If the error persists, contact JP

Software for assistance.

Out of paper: Windows detected an out-of-paper condition on one of the printers (LPT1, LPT2, or LPT3). Check your printer and add paper if necessary.

Overflow: An arithmetic overflow occurred in the [@EVAL](#)^[374] variable function. Check the values being passed to @EVAL.

Read error: Windows encountered a disk read error; usually caused by a bad or unformatted disk. Also see **Data error** above.

Sector not found: Disk error, usually caused by a bad or unformatted disk. Also see **Data error** above.

Seek error: Windows 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 violation: You tried to access a file in use by another program in a multitasking system or on a network. 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.

SHRALIAS already loaded: You used the [SHRALIAS](#)^[290] command to load *SHRALIAS.EXE*, but it was already loaded. This message is informational and generally does not indicate an error condition.

SHRALIAS not loaded: You used the [SHRALIAS /U](#)^[290] 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.

String area overflow: the command processor ran out of room to store the text from string directives in the [.INI file](#)^[89]. Reduce the complexity of the *.INI* file or contact JP Software for assistance.

String too long: You tried to put more than 2038 characters into the [KEYSTACK](#)^[236] buffer. Reduce the number of characters you are trying to send to the application at one time.

Syntax error: A command or [variable function](#)^[357] was entered in an improper format. Check the syntax and correct the error.

Too many open files: Windows has run out of file handles.

Unbalanced parentheses: The number of left and right parentheses did not match in an expression passed to the [@EVAL](#)^[374] variable function. Correct the expression and retry the operation.

UNKNOWN_CMD loop: The [UNKNOWN_CMD alias](#)^[138] 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 the command processor 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 [ALIAS](#)^[138]).

Unknown option name: ([OPTION](#)^[253]) You are attempting to modify or display an invalid or unknown option name.

Unknown process: [TASKEND](#)^[298] cannot find the process you specified. If you are ending a process

using the title you may need to use wildcards to get a match on the title string. Correct the command and try again.

Variable loop: A nested environment variable refers to itself, or variables are nested more than 16 deep. Correct the error and retry the command.

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: Windows encountered a disk write error; usually caused by a bad or unformatted disk. Also see **Data error** above.

8 Reference Information

- ▶ [CMD.EXE Comparison](#) ^[430]
- ▶ [File Systems and File Name Conventions](#) ^[438]
- ▶ [Miscellaneous Reference Information](#) ^[458]
- ▶ [ASCII, Key Codes, and ANSI Commands](#) ^[445]
- ▶ [Glossary](#) ^[468]
- ▶ [Copyright and Version](#) ^[491]

8.1 CMD.EXE Comparison

THIS TOPIC IS STILL UNDER CONSTRUCTION. COMMENTS WELCOME.

The comparison of commands available is based on the version of CMD.EXE included with Windows XP Build 2600 Service Pack 2.

If the CMD command name matches an internal JP command, the JP command is either identical, or is more enhanced.

CMD "external commands" without a matching JP internal command can be used identically to CMD as long as CMD's directory is on the path (which is the default).

- ▶ [4NT/TC command equivalents in CMD](#) ^[430]
- ▶ [CMD command equivalents in 4NT/TC](#) ^[430]
- ▶ [Command line editing](#) ^[430]
- ▶ [Filename completion](#) ^[430]
- ▶ [Command completion](#) ^[430]
- ▶ [Redirection](#) ^[430]
- ▶ [Wildcards](#) ^[430]
- ▶ [Built-In Variables](#) ^[430]
- ▶ [Batch File Structure](#) ^[430]
- ▶ [Unique JPsoft command processor features](#) ^[430]

4NT/TC command equivalent in CMD

4NT/TC command	CMD command (nearest functionality)	CMD command type
? ^[346]	HELP	external

<u>ACTIVATE</u> ^[136]		
<u>ALIAS</u> ^[138]	DOSKEY	external
<u>ASSOC</u> ^[145]	ASSOC	internal
<u>ATTRIB</u> ^[146]	ATTRIB	external
<u>BATCOMP</u> ^[149]		
<u>BDEBUGGER</u> ^[149]		
<u>BEEP</u> ^[156]		
<u>BREAK</u> ^[157]	BREAK	internal
<u>CALL</u> ^[157]	CALL	internal
<u>CANCEL</u> ^[158]		
<u>CD</u> ^[159]	CD	internal
<u>CDD</u> ^[160]		
<u>CHCP</u> ^[162]	CHCP	external
<u>CHDIR</u> ^[159]	CHDIR	internal
<u>CLS</u> ^[163]	CLS	internal
<u>COLOR</u> ^[163]	COLOR	internal
<u>COPY</u> ^[164]	COPY	internal
<u>COPY</u> ^[164]	XCOPY	external
<u>DATE</u> ^[170]	DATE	internal
<u>DEL</u> ^[172]	DEL	internal
<u>DELAY</u> ^[175]		
<u>DESCRIBE</u> ^[176]		
<u>DETACH</u> ^[178]		
<u>DIR</u> ^[179]	DIR	internal
<u>DIRHISTORY</u> ^[189]		
<u>DIRS</u> ^[190]		
<u>DO</u> ^[194]		
<u>DRAWBOX</u> ^[195]		
<u>DRAWHLINE</u> ^[196]		
<u>DRAWVLINE</u> ^[197]		

<u>ECHO</u> ^[198]	ECHO	internal
<u>ECHOERR</u> ^[199]		
<u>ECHOS</u> ^[199]		
<u>ECHOSERR</u> ^[200]		
<u>ENDLOCAL</u> ^[200]	ENDLOCAL	internal
<u>ERASE</u> ^[172]	ERASE	internal
<u>ESET</u> ^[201]		
<u>EVENTLOG</u> ^[203]		
<u>EXCEPT</u> ^[204]		
<u>EXIT</u> ^[205]	EXIT	internal
<u>FFIND</u> ^[206]	FIND	external
<u>FFIND</u> ^[206]	FINDSTR	external
<u>FOR</u> ^[210]	FOR	internal
<u>FREE</u> ^[210]		
<u>FTYPE</u> ^[211]	FTYPE	internal
<u>FUNCTION</u> ^[210]		
<u>GLOBAL</u> ^[220]		
<u>GOSUB</u> ^[221]	CALL :LABEL	internal
<u>GOTO</u> ^[222]	GOTO	internal
<u>HEAD</u> ^[224]		
<u>HELP</u> ^[225]	HELP	external
<u>HISTORY</u> ^[226]		
<u>IF</u> ^[227]	IF	internal
<u>IFF</u> ^[228]		
<u>IFTP</u> ^[229]		
<u>INKEY</u> ^[231]		
<u>INPUT</u> ^[233]		
<u>KEYBD</u> ^[234]		
<u>KEYS</u> ^[235]		
<u>KEYSTACK</u> ^[236]		
<u>LIST</u> ^[237]	MORE	external
<u>LOADBTM</u> ^[242]		
<u>LOG</u> ^[242]		
<u>MD</u> ^[243]	MD	internal
<u>MEMORY</u> ^[245]		
<u>MKDIR</u> ^[245]	MKDIR	internal
<u>MKLNK</u> ^[245]		
<u>MOVE</u> ^[246]	MOVE	internal
<u>MSGBOX</u> ^[251]		

<u>ON</u> [252]		
<u>OPTION</u> [253]		
<u>PATH</u> [254]	PATH	internal
<u>PAUSE</u> [256]	PAUSE	internal
<u>PDIR</u> [256]		
<u>PLAYAVI</u> [260]		
<u>PLAYSOUND</u> [261]		
<u>POPD</u> [261]	POPD	internal
<u>PRINT</u> [262]	PRINT	external
<u>PROMPT</u> [262]	PROMPT	internal
<u>PUSHD</u> [264]	PUSHD	internal
<u>QUERYBOX</u> [265]		
<u>QUIT</u> [266]	GOTO :EOF	
<u>RD</u> [267]	RD	internal
<u>REBOOT</u> [268]		
<u>RECYCLE</u> [269]		
<u>REM</u> [269]	REM	internal
<u>REN</u> [270]	REN	internal
<u>RENAME</u> [270]	RENAME	internal
<u>RETURN</u> [272]	GOTO :EOF	
<u>RMDIR</u> [267]	RMDIR	internal
<u>SCREEN</u> [273]		
<u>SCRPUT</u> [274]		
<u>SELECT</u> [275]		
<u>SENDMAIL</u> [279]		
<u>SET</u> [281]	SET	internal
<u>SETDOS</u> [283]		
<u>SETLOCAL</u> [287]	SETLOCAL	internal
<u>SHIFT</u> [288]	SHIFT	internal
<u>SHORTCUT</u> [289]		
<u>SHRALIAS</u> [290]		
<u>SMPP</u> [290]		
<u>SNPP</u> [291]		
<u>START</u> [291]	START	internal
<u>SWITCH</u> [295]		

<u>TAIL</u> ^[29b]		
<u>TASKEND</u> ^[29b]		
<u>TASKLIST</u> ^[29b]		
<u>TEE</u> ^[30b]		
<u>TEXT</u> ^[30b]		
<u>TIME</u> ^[30b]	TIME	internal
<u>TIMER</u> ^[30b]		
<u>TITLE</u> ^[30b]	TITLE	internal
<u>TOUCH</u> ^[30b]		
<u>TREE</u> ^[30b]	TREE	external
<u>TRUENAME</u> ^[30b]		
<u>TYPE</u> ^[30b]	TYPE	internal
<u>UNALIAS</u> ^[31b]		
<u>UNFUNCTION</u> ^[31b]		
<u>UNSET</u> ^[31b]	SET VARNAME=	internal
<u>VER</u> ^[31b]	VER	internal
<u>VERIFY</u> ^[31b]	VERIFY	internal
<u>VOL</u> ^[31b]	VOL	internal
<u>VSCRPUT</u> ^[31b]		
<u>WHICH</u> ^[31b]		
<u>WINDOW</u> ^[31b]		
<u>Y</u> ^[31b]		

CMD command equivalents in 4NT/TC

CMD command	CMD class	4NT command	comparison
ASSOC	internal	<u>ASSOC</u> ^[14b]	enhanced
AT	external		
ATTRIB	external	<u>ATTRIB</u> ^[14b]	enhanced
BREAK	internal	<u>BREAK</u> ^[15b]	enhanced

CACLS	external		
CALL	internal	CALL ^[157] , GOSUB ^[224]	enhanced
CD	internal	CD ^[158]	enhanced
CHCP	external	CHCP ^[162]	identical
CHDIR	internal	CHDIR ^[159]	enhanced
CHKDSK	external		
CHKNTFS	external		
CLS	internal	CLS ^[163]	enhanced
COLOR	internal	COLOR ^[163]	enhanced
COMP	external		
COMPACT	external		
CONVERT	external		
COPY	internal	COPY ^[164]	enhanced
DATE	internal	DATE ^[170]	enhanced
DEL	internal	DEL ^[172]	enhanced
DIR	internal	DIR ^[179] , PDIR ^[256]	enhanced
DISKCOMP	external		
DISKCOPY	external		
DOSKEY	external	ALIAS ^[138]	enhanced
ECHO	internal	ECHO ^[198]	identical
ENDLOCAL	internal	ENDLOCAL ^[200]	enhanced
ERASE	internal	ERASE ^[172]	enhanced
EXIT	internal	EXIT ^[205]	enhanced
FC	external		
FIND	external	FFIND ^[206]	enhanced
FINDSTR	external	FFIND ^[206]	enhanced
FOR	internal	FOR ^[216]	enhanced
FORMAT	external		
FTYPE	internal	FTYPE ^[217]	enhanced
GOTO	internal	GOTO ^[222]	enhanced
GRAFTABL	external		
HELP	external	HELP ^[225]	enhanced
IF	internal	IF ^[227]	enhanced
LABEL	external		

MD	internal	MD ^[243]	enhanced
MKDIR	internal	MKDIR ^[243]	enhanced
MODE	external		
MORE	external	LIST ^[237]	enhanced
MOVE	internal	MOVE ^[246]	enhanced
PATH	internal	PATH ^[254]	identical
PAUSE	internal	PAUSE ^[256]	enhanced
POPD	internal	POPD ^[267]	enhanced
PRINT	external	PRINT ^[262]	enhanced
PROMPT	internal	PROMPT ^[262]	enhanced
PUSHD	internal	PUSHD ^[264]	enhanced
RD	internal	RD ^[267]	enhanced
RECOVER	external		
REM	internal	REM ^[269]	identical
REN	internal	REN ^[276]	enhanced
RENAME	internal	RENAME ^[276]	enhanced
REPLACE	external		
RMDIR	internal	RMDIR ^[267]	enhanced
SET	internal	SET ^[287]	enhanced
SETLOCAL	internal	SETLOCAL ^[287]	enhanced
SHIFT	internal	SHIFT ^[288]	identical
SORT	external		
START	internal	START ^[291]	enhanced
SUBST	external		
TIME	internal	TIME ^[302]	enhanced
TITLE	internal	TITLE ^[304]	enhanced
TREE	external	TREE ^[307]	enhanced
TYPE	internal	TYPE ^[308]	enhanced
VER	internal	VER ^[313]	enhanced
VERIFY	internal	VERIFY ^[314]	enhanced
VOL	internal	VOL ^[314]	identical
XCOPY	external	COPY ^[164]	enhanced

Command line editing

Filename completion

Command completion

Redirection

Wildcards

Built-In Variables

Batch File Structure

Unique JPssoft command processor features

Batch debugger

Aliases

Internal functions

User defined functions

File selection

Ranges

Internet access and email

ANSI X3.64 support

Directory navigation

Histories and logs

Intersession sharing

8.2 Limits

This topic is still under construction.

The categories of information below are expected to be collected and presented tabularly.

Length Limits

entity	name	value	all
environment variable	80	8,189	8,191
alias			
user defined function			

command type	before expansion	after expansion
single line command	8,191	16,383
multiline command		
command group	8,191	16,383

Nesting Limits

command	depth
CALL	
DO	
GOSUB	
SETLOCAL	

Miscellaneous Limits

entity	limit
character count in any function	8,191
number of batch file parameters	511
number of GOSUB parameters	
file name	2,047
include list	2,047
single argument	2,047

8.3 File Systems & File Name Conventions

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.

The topics below are roughly divided according to the different parts of a file name, and cover the file system structure and naming conventions:

- ▶ [Drives and Volumes](#) ⁴³⁹
- ▶ [File Systems](#) ⁴³⁹
- ▶ [Directories and Subdirectories](#) ⁴⁴⁰
- ▶ [File Names](#) ⁴⁴¹
- ▶ [File Attributes](#) ⁴⁴²
- ▶ [Time Stamps](#) ⁴⁴³
- ▶ [NTFS Streams](#) ⁴⁴⁴

8.3.1 Drives & 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 current operating systems can **partition** a large hard disk into multiple logical drives or **volumes** that are usually called **C:**, **D:**, **E:**, etc. Network systems (LANs) give additional drive letters to sections of the network file server drives. In addition, you can often access network drives via their **UNC (universal naming convention)** name (e.g. `\\data\vol1\...`), without using a drive letter. See [File Systems](#)^[439] 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. Like CD-ROM drives, RAM disks are usually assigned drive letters beyond the last hard disk in the system, but before network drives.

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 network drives.

Each volume is formatted under a particular file system; see [File Systems](#)^[439] for details. Additional information about disk files and directories is available under [Directories and Subdirectories](#)^[440], [File Names](#)^[441], and [File Attributes](#)^[442].

8.3.2 File Systems

Each disk volume is organized according to a file system. The file system determines how files are named, how they are organized on the disk, what information about each file is stored in the file directory, whether or not .

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 the command processor runs support four standard file systems: FAT, VFAT, FAT32, and NTFS. Throughout this manual, the term "LFN file system" is commonly used to describe the VFAT, FAT32, and NTFS systems as a group, where LFN stands for **Long File Name**. See [File Names](#)^[441] for details on the rules for naming files under each file system.

Additional file systems may be installed under some operating systems to support CD, DVD and network drives.

The file system type is determined when a hard disk volume is formatted and applies to the entire volume. For example, you might have a 600 GB hard disk divided into three 149.5 GB volumes and a 500 MB volume, with the first three volumes (C:, D:, and E:) formatted for NTFS, and the fourth formatted for one of FAT/VFAT/FAT32.

The command processor supports any standard file system installed under your operating system, and can access all files supported by the operating system.

Third-party software, such as LFNDOS and NTFSDOS, may make VFAT / FAT32 / NTFS file systems partially (i.e., read only) or fully accessible in operating systems which do not provide native support for them. If you use one of these packages, you must determine the extent they are compatible with your command processor.

Additional information about disk files and directories is available under [Drives and Volumes](#)^[439], [Directories and Subdirectories](#)^[440], [File Names](#)^[441], and [File Attributes](#)^[442].

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. The command processor supports all network file systems which are compatible with the underlying operating system. The networking software used to access remote systems (such as Unix, Linux, MacOS, etc..) which use different file systems typically emulates one of the common Windows file systems. Those emulations do not always provide a perfect duplicate of some functions (attributes, timestamps, etc.), an issue unrelated to the command processor per se.

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.

When you use a network file system, remember that the naming rules 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. The command processor 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 rare cases, the command processor 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.

Universal Naming Convention (UNC)

Some 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](#)^[440]. If *server* uses a Microsoft operating system, *volume* is the server's local drive letter, followed by the \$ (dollar) sign (i.e., replacing the : (colon) used for local volumes).

Despite its name, the UNC is a Microsoft convention.

The command processor also allows you to use UNC directory names when changing directories (see [Directory Navigation](#)^[54] for more details).

8.3.3 Directories & 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 that of file drawers containing groups of hanging folders, hanging folders containing smaller folders, and so on. (The terms directory and folder are *not* synonymous but often used as such in common Windows terminology. For accuracy, we use **directory** throughout these help files unless other folder types are also specifically applicable.)

Every drive has a root or base directory, and many 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 [File Names](#)^[441]).

The drive and subdirectory portions 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.

Under 4NT and Take Command, the path and filename can be up to 2047 characters, though many Windows applications (including `CME.EXE` and Explorer) have trouble with path and filename lengths exceeding 260 characters. Shorter paths and names are advisable under Windows whenever feasible.

The command processor maintains 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:\JPSOFT.DOC` refers to the file `JPSOFT.DOC` in the current directory on drive `E`; whereas `E:\JPSOFT.DOC` refers to the file `JPSOFT.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. The command processor allows you to use additional periods to specify directories further "up" the tree (see [Extended Parent Directory Names](#)^[47]).

Additional information about disk files and file systems is available under [Drives and Volumes](#)^[439], [File Systems](#)^[439], [File Names](#)^[441], and [File Attributes](#)^[442].

8.3.4 File Names

FAT File Names

Under the **FAT** file system, a 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. FAT filenames with an 8-character name and a 3-character extension are sometimes referred to as short file names (SFNs) to distinguish them from long file names (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 the command processor, it is best to avoid using them in filenames. It is also better to use only those characters found in ASCII[446], because changing font and/or code page may change drastically how they are displayed.`

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 the command processor.

Long File Names

VFAT, **FAT32** and **NTFS** allow using **long 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 [Wildcards](#)^[36] for details on avoiding problems with interpretation of those file names under the command processor.

LFNs 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. This is in contrast with Unix-style file systems, which are case sensitive, and permit **AA**, **Aa**, **aA**, and **aa** to be four different file names.

A file that has an LFN may have an additional, "FAT-compatible" name, which contains only those characters legal on a FAT volume, and which meets the 8-character name / 3-character extension limits. Programs which cannot handle long names (for example, DOS programs accessing an NTFS drive under Windows NT / 2000 / XP / 2003) generally can access files by using their FAT-compatible names. This name is assigned at the time the LFN is created in the specific directory, and to make it unique, it depends on what other SFNs exist in that directory at that instance. Consequently, when copying the file to another directory by its LFN the SFN generated in the target directory may be different from the SFN in the source directory.

When specifying an LFN-compatible file name, which includes spaces or other characters that would either not be allowed in a FAT name, or that may have syntactical significance for the command processor, you must place quote marks around the name in the command line. 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 the command processor. For more information on underlying issues related to fonts and character sets see [ASCII, Key Codes, and ANSI X3.64 Commands](#)^[445].

Additional information about disk files and file systems is available under [Drives and Volumes](#)^[439], [File Systems](#)^[439], [Directories and Subdirectories](#)^[440], [File Attributes](#)^[442], and [Time Stamps](#)^[443].

8.3.5 File Attributes

Each file has attributes, each of which defines a single characteristic of the file that can be either **set** or **reset**. Most file processing commands allow you to select files for processing based on their attributes.

The basic attributes *Archive*, *Read only*, *Hidden*, *System*, and *Directory* are present on all disk volumes. Windows 2000, XP, and 2003 support additional attributes: *Encrypted*, *Compressed*, *Normal*, *Offline*, *Temporary*, *Not content-indexed*, *Sparse*, and *Junction / Reparse point*. Except *Normal* (these attributes are available only on NTFS volumes. 4NT and Take command fully support these [extended attributes](#)^[39].

Archive - **set** by the operating system when the contents of the file are modified to indicate that it is a *candidate to be archived*, i.e., to be backed up. The attribute can be **reset** by any program to indicate that the file's contents have been archived. Most programs which can unset this attribute require that you use the explicit reset option, and default to retaining the status of this attribute. For example, the command processor command **COPY** requires the **/X** option to reset this attribute.

Read-only – if this attribute is set, the file can't be changed or erased accidentally. Most programs honor this attribute by default, which helps to protect important files from erasure and damage.

Either of the **Hidden** and **System** attributes, when **set**, prevent the file from appearing in directory listings and file searches, including those performed by file processing command of the command processor, *unless explicitly requested to be included*.. This both protects such files from accidental modification, and also speeds up user tasks not explicitly intended to process them.

Directory – this attribute is **set** by the operating system when a file is created to be a **subdirectory**, e.g., by the MKDIR command. The attribute cannot be reset. The operating system restricts all accesses to a directory file to directory manipulation operations.

Volume label – a special attribute of at most one directory entry in the root directory of a disk drive. The entry can be created, modified, or deleted only through the OS utility LABEL (or equivalent third-party software). JP Software command processors do not directly modify the *volume label* or any of its attributes, and provide read access only through the [VOL](#)^[314] command and the [@LABEL](#)^[396] variable function. All other commands ignore this directory entry.

Normal – this pseudoattribute is considered to be **set** if all other attributes (including the [extended attributes](#)^[39] available only on an NTFS volumes under Windows 2000 / XP / 2003) are **reset**. It is not stored by the file system.. When 4NT or Take Command check file attributes, they consider the Normal attribute as **set** if each of the other attributes is either reset, or unsupported by the combination of the file system and operating system.

The file attributes can also be accessed with the [ATTRIB](#)^[146] and [DIR](#)^[179] commands, and by the [@ATTRIB](#)^[366] and [@WATTRIB](#)^[410] variable functions.

Attributes can be set, reset, and viewed with the [ATTRIB](#)^[146] command. The [DIR](#)^[179] command also has options to view the attribute status of files, and to view information about normally *invisible* hidden and system files and directories.

8.3.6 Time Stamps

Each file has one or more time stamps. They are used by the operating system to record when the file was created, last modified, or last accessed. Most 4NT and Take Command file processing commands allow you to select files for processing based on their time stamp(s).

1. **Write time** is the date and time the file was last written, i.e., when its content was last modified, *On FAT volumes this is the only timestamp*. In all commands and functions this is the timestamp used unless you specify another. Always available. On FAT and VFAT volumes the resolution is 2 s. NTFS volumes have a 100 nanosecond resolution for the file creation and last write. (Unix and Linux systems use 1-s resolution.) *When a file is copied using the COPY command, even across a network, its write time is not changed. However, different file systems record time with different resolution, so minor changes may occur.* The COPY

command properly accommodates the different representations used by Unix/Linux systems, which record in UTC, and FAT / VFAT / NTFS file systems, which record in local time, as long as the local time zone is properly set by each system. Note that the write time of FAT / VFAT / NTFS subdirectories is never updated by the operating system, no matter how many times you added or deleted files. Beware: On the typical Unix/Linux system the *last modified* time of a copy of a file is the time it was copied, not when its *content* was last modified, due to the default settings in Unix and Linux. Since most web servers use Linux, this may make you falsely think you need a new download.

2. **Creation time** is the date and time the current instance of the file was created. When you copy an old file, you will see a creation time later than the write time.
3. **Access time** is the date, and on NTFS volumes, the time, when the file was last read. (NTFS volumes have a resolution of 1 hour.)

Several command processor commands and functions 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** creation time stamp,
- w** last write time stamp, and
- a** last access time stamp.

Note that FAT32 and VFAT volumes store the date but not the time of the *last access*. On these drives the time of last access will always be 00:00.

Note: The timestamp resolution on FAT volumes is limited by the file system. In general the creation time resolution is 10 milliseconds, the write time resolution is 2 seconds, and the access time resolution is one day.

The **TOUCH**^[304] command can be used to modify the timestamps of files (and directories under NT/W2000/XP).

Additional information about disk files and file systems is available under [Drives and Volumes](#)^[439], [File Systems](#)^[439], [Directories and Subdirectories](#)^[440], and [File Names](#)^[441].

8.3.7 NTFS File Streams

Microsoft's NTFS file system (in Windows 2000 and above) 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, FFIND, 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 also accept stream names.

DIR and other commands and functions which return file information (TREE, @FILESIZE, etc.) reference only the normal file data, and do not include stream data.

8.4 ASCII, Key Codes & ANSI Commands

For ASCII, key code, and ANSI X3.64 code reference tables, see:

- ▶ [ASCII Table](#)^[446]
- ▶ [Key Codes and Scan Codes Table](#)^[452]
- ▶ [ANSI X3.64 Commands Reference](#)^[456]

The remainder of this section gives a detailed explanation of character sets, ASCII, and key codes (for more information on 4NT's and Take Command's ANSI X3.64 string support see the separate [ANSI X3.64 Commands Reference](#)^[456] topic). If you are troubleshooting a keyboard or character display problem be sure to read all of the explanation below before referring to the tables.

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 document. 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:

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

The numeric key code is determined by your physical hardware including the language that your keyboard is produced for. 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, the code page, and the display font.

For an explanation of how key codes work, see the [Key Codes and Scan Codes Explanation](#)^[451]. For a list of key codes and scan codes for keys on the standard U.S. keyboard see the [Key Codes and Scan Codes Table](#)^[452].

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, but not in commonly-used U.S. English alphabetic, numeric, or punctuation characters.

Most systems 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. For a complete list of ASCII see the [ASCII Table](#)^[446].

ASCII, the **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange, was originally defined in the 1960s to replace a large variety of mostly 6-bit character codes, typically not capable of representing both upper and lower case letters in a context-free manner. The 7-bit codes in ASCII include the 26 upper case and the 26 lower case letters used in English, the 10 numerals, 32 punctuation marks used in U.S. publishing, and 33 control codes, designed for use in communication and printer control, and as data delimiters. ASCII was a superset of almost all characters previously used in computers. There were no graphic representations for the control characters included because the code set was designed for exchanging information between various hardware elements.

IBM in its design of the original PC used 8-bit storage elements to store each character, and used the word **byte** (which its mainframe division coined in the 1960s) to denote it. The official documentation of the Internet uses the word **octet** (from Latin *octo*, 8). This allowed incorporating the whole 128-member ASCII, and the addition of additional 128 extended codes for mathematical symbols, a few Greek letters, many letters used in European languages utilizing Latin-based alphabets, characters to draw boxes and lines, and some other miscellaneous symbols. This complete 256-character set was called the Original Equipment Manufacturer or **OEM** character set. For purposes of screen display IBM designed graphic representations for all of the control characters except 0 (NUL) and 255.

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 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).

The control codes can be entered on most keyboards by pressing the **Ctrl** key plus another character, or by pressing the special keys **Tab**, **Enter**, **Backspace**, and **Esc**. However, ASCII is not a complete character set for the IBM PC, because it defines only 128 of the 256 symbols usable on PC systems.

See your operating system documentation for more information about character sets, code pages, and country and language support. Refer to your operating system and/or font documentation for details on the full character set available in any particular font.

The tables in this section are based on U.S. English conventions. Your system may differ if it is configured for a different country or language. See your operating system documentation for more information about country and language support.

Note: You may also be able to use the "**Alt + keypad**" approach to generate ASCII values in both ASCII and Unicode products. See "[Command Line Editing](#)" for additional information.

8.4.1 ASCII Tables

These tables show the 128-character ASCII set for U.S. English systems. Most of the characters in code range 32..126 (the only codes for which ASCII specifies displayable symbols) will be the same on non-U.S. systems. The symbols associated with all other codes vary from font to font, as well as from country to country.

For more details on ASCII, character sets, and key codes, see the general information topic on [ASCII, Key Codes, and ANSI X3.64 Commands](#).

- [Control Characters 0 - 31, 127](#)
- [Printing Characters 32 - 47](#)
- [Printing Characters 48 - 63](#)

- [Printing Characters 64 - 79](#)^[44b]
- [Printing Characters 80 - 95](#)^[44b]
- [Printing Characters 96 - 111](#)^[44b]
- [Printing Characters 112 - 126](#)^[44b]

Control Characters 0 - 31, 127

ASCII (Dec)	ASCII (Hex)	Ctrl + Key	Acronym	Name
0	00	@	NUL	null
1	01	A	SOH	start of header
2	02	B	STX	start text
3	03	C	ETX	end text
4	04	D	EOT	end of transmission
5	05	E	ENQ	enquiry
6	06	F	ACK	acknowledge
7	07	G	BEL	bell
8	08	H	BS	backspace
9	09	I	HT	horizontal tab
10	0A	J	LF	linefeed
11	0B	K	VT	vertical tab
12	0C	L	FF	form feed
13	0D	M	CR	carriage return
14	0E	N	SO	shift out
15	0F	O	SI	shift in
16	10	P	DLE	data link escape
17	11	Q	DC1	device control 1
18	12	R	DC2	device control 2
19	13	S	DC3	device control 3
20	14	T	DC4	device control 4
21	15	U	NAK	negative acknowledge
21	16	V	SYN	synchronize
23	17	W	ETB	end text block
24	18	X	CAN	cancel
25	19	Y	EM	end of medium
26	1A	Z	SUB	substitute
27	1B	[ESC	escape
28	1C	\	FS	field separator
29	1D]	GR	group separator
30	1E	^	RS	record separator
31	1F	_	US	unit separator
127	7F	<i>n/a</i>	DEL	delete

Printing Characters 32 - 47

<u>Dec</u>	<u>Hex</u>	<u>Char</u>	<u>Special character name</u>
032	20	Space	space
033	21	!	exclamation mark
034	22	"	quote mark
035	23	#	number sign
036	24	\$	dollar (currency) sign
037	25	%	percent mark
038	26	&	ampersand
039	27	'	apostrophe
040	28	(left parenthesis
041	29)	right parenthesis
042	2A	*	asterisk
043	2B	+	plus sign
044	2C	,	comma
045	2D	-	hyphen (minus sign)
046	2E	.	period
047	2F	/	slash

Printing Characters 48 - 63

<u>Dec</u>	<u>Hex</u>	<u>Char</u>	<u>Special character name</u>
048	30	0	
049	31	1	
050	32	2	
051	33	3	
052	34	4	
053	35	5	
054	36	6	
055	37	7	
056	38	8	
057	39	9	
058	3A	:	colon
059	3B	;	semicolon
060	3C	<	less than sign
061	3D	=	equal sign
062	3E	>	greater than sign
063	3F	?	question mark

Printing Characters 64 - 79

<u>Dec</u>	<u>Hex</u>	<u>Char</u>	<u>Special character name</u>
064	40	@	at sign
065	41	A	
066	42	B	
067	43	C	
068	44	D	
069	45	E	
070	46	F	
071	47	G	
072	48	H	
073	49	I	
074	4A	J	
075	4B	K	
076	4C	L	
077	4D	M	
078	4E	N	
079	4F	O	

Printing Characters 80 - 95

<u>Dec</u>	<u>Hex</u>	<u>Char</u>	<u>Special character name</u>
080	50	P	
081	51	Q	
082	52	R	
083	53	S	
084	54	T	
085	55	U	
086	56	V	
087	57	W	
088	58	X	
089	59	Y	
090	5A	Z	
091	5B	[left bracket
092	5C	\	backslash
093	5D]	right bracket
094	5E	^	caret
095	5F	_	underscore

Printing Characters 96 - 111

<u>Dec</u>	<u>Hex</u>	<u>Char</u>	<u>Special character name</u>
096	60	`	accent grave (backtick)
097	61	a	
098	62	b	
099	63	c	
100	64	d	
101	65	e	
102	66	f	
103	67	g	
104	68	h	
105	69	i	
106	6A	j	
106	6B	k	
108	6C	l	
109	6D	m	
110	6E	n	
111	6F	o	

Printing Characters 112 - 126

Dec	Hex	Char	Special character name
112	70	p	
113	71	q	
114	72	r	
115	73	s	
116	74	t	
117	75	u	
118	76	v	
119	77	w	
120	78	x	
121	79	y	
122	7A	z	
123	7B	{	left brace
124	7C		vertical bar
125	7D	}	right brace
126	7E	~	tilde

8.4.2 Key Codes and Scan Codes Explanation

(This section explains how key codes and scan codes work. For a reference chart, see the [Key Codes and Scan Codes Table](#)^[452]. For more details on key codes, scan codes, and ASCII see the general information on [ASCII, Key Codes, and ANSI X3.64 Commands](#)^[445].)

When you press a single key or a key combination, Windows translates your keystroke into two numbers: a scan code, representing the actual key that was pressed, and an ASCII code, representing the ASCII value for that key. Windows returns these numbers the next time a program requests keyboard input. This section explains how key codes work; for information on using them with the command processor see the [key mapping directives](#)^[114] in the [.INI file](#)^[89], [keystroke aliases](#)^[138], and [INKEY](#)^[231].

Most command processor commands that use the numeric key codes listed here also use key names, which are usually more convenient to use than the numeric codes. See [Keys and Key Names](#)^[464] for more information on key names.

As PCs have evolved, the structure of keyboard codes has evolved somewhat haphazardly with them, resulting in a bewildering array of possible key codes. We'll give you a basic explanation of how key codes work. For a more in-depth discussion, refer to a BIOS or PC hardware reference manual.

The nuances of how your keyboard behaves depends on the keyboard manufacturer, the computer manufacturer who provides the built-in BIOS, and your operating system. As a result, we can't guarantee the accuracy of the information in the tables for every system, but the discussion and reference table should be accurate for most systems. Our discussion is based on the "enhanced" keyboard commonly used on 286, 386, 486, and Pentium computers, but virtually all of it is applicable to the 84-key keyboards on older systems. The primary difference is that older keyboards lack a separate cursor pad and only have 10 function keys.

All keys have a scan code, but not all have an ASCII code. For example, function keys and cursor keys are not part of the ASCII character set and have no ASCII, but they do have a scan code. Some keys have more than one ASCII, depending on what other keys they are combined with. The "**A**" key, for example, has ASCII 97 (lower case "a") if you press it by itself (with "caps lock" off). If you press it along with **Shift** (with "caps lock" off), the code changes to 65 (upper case "A"). If you press **Ctrl** and **A**, the code changes to 1. In all these cases, the scan code (30) is unchanged because you are pressing the same physical key.

Things are different if you press **Alt-A**. **Alt** keystrokes have no ASCII, so Windows returns an ASCII of 0, along with the **A** key's scan code of 30. This allows a program to detect all the possible variations of **A**, based on the combination of ASCII and scan code.

Some keys generate more than one scan code depending on whether **Shift**, **Ctrl**, or **Alt** is pressed. This allows a program to differentiate between two different keystrokes on the same key, neither of which has a corresponding ASCII. For example, **F1** has no ASCII, so it returns code 0, and the **F1** scan code of 59. **Shift-F1** also returns a code 0; if it also returned a scan code of 59, a program couldn't distinguish it from **F1**. The operating system translates scan codes for keys like **Shift-F1** (and **Ctrl-F1** and **Alt-F1**) so that each variation returns a different scan code along with an ASCII 0.

On the enhanced keyboard there's one more variation: non-ASCII keys on the cursor keypad (such as up-arrow) return the same scan code as the corresponding key on the numeric keypad, for compatibility reasons. If they also returned an ASCII code of 0, a program couldn't tell which key was pressed. Therefore, these keys return an ASCII code of 224 rather than 0. This means that older programs, which only look for an ASCII 0 to indicate a non-ASCII keystroke like up-arrow, may not detect these cursor pad keys properly.

The number of different codes returned by any given key varies from one (for the space bar) to four, depending on the key, the design of your keyboard, and the operating system. Some keys, like **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** keys, do not have any code representations at all. The same is true of keystrokes with more than one modifying key, like **Ctrl-Shift-A**. The operating system may perform special actions automatically when you press these keys (for example, it switches into Caps Lock mode when you press **Caps Lock**), but it does not report the keystrokes to whatever program is running. Programs which detect such keystrokes access the keyboard hardware directly, a subject which is beyond the scope of this documentation.

8.4.3 Key Codes and Scan Codes Table

(For more details on key codes, scan codes, and ASCII see the general information on [ASCII, Key Codes, and ANSI X3.64 Commands](#)^[44b], and the [Key Codes and Scan Codes Explanation](#)^[45h].)

The table below shows standard codes used on a U.S. keyboard. Other keyboards will vary, due to the changes required to support characters outside the U.S. English character set.

Key names prefaced by **np** are on the numeric keypad. Those prefaced by **cp** are on the cursor keypad between the main typing keys and the number keypad. The numeric keypad values are valid if Num Lock is turned off. If you need to specify a number key from the numeric keypad, use the scan code shown for the keypad and the ASCII code shown for the corresponding typewriter key. For example, the keypad "7" has a scan code of 71 (the np Home scan code) and an ASCII code of 54 (the ASCII code for "7").

The chart is blank for key combinations that do not have scan codes or ASCII codes, like **Ctrl-1** or **Alt-PgUp**.

Top Keyboard Row

Key Cap	Scan	ASCII	Shift-Scan	Shift-ASCII	Ctrl-Scan	Ctrl-ASCII	Alt-Scan
Esc	1	27	1	27	1	27	1
~	41	96	1	126			41
1	2	49	2	33			120
2	3	50	3	64	3	0	121
3	4	51	4	35			122
4	5	52	5	36			123
5	6	53	6	37			124
6	7	54	7	94	7	30	125
7	8	55	8	38			126
8	9	56	9	42			127
9	10	57	10	40			128
0	11	48	11	41			129
-	12	45	12	95	12	31	130
=	13	61	13	43			131
Backspace	14	8	14	8	14	127	14

Second Keyboard Row

Key Cap	Scan	ASCII	Shift-Scan	Shift-ASCII	Ctrl-Scan	Ctrl-ASCII	Alt-Scan
Tab	15	9	15	0	148	0	165
Q	16	113	16	81	16	17	16
W	17	119	17	87	17	23	17
E	18	101	18	69	18	5	18
R	19	114	19	82	19	18	19
T	20	116	20	84	20	20	20
Y	21	121	21	89	21	25	21
U	22	117	22	85	22	21	22
I	23	105	23	73	23	9	23
O	24	111	24	79	24	15	24
P	25	112	25	80	25	16	25
[26	91	26	123	26	27	26
]	27	93	27	125	27	29	27
\	43	92	43	124	43	28	43

Third Keyboard Row

<u>Key Cap</u>	<u>Scan</u>	<u>ASCII</u>	<u>Shift-Scan</u>	<u>Shift-ASCII</u>	<u>Ctrl-Scan</u>	<u>Ctrl-ASCII</u>	<u>Alt-Scan</u>
A	30	97	30	65	30	1	30
S	31	115	31	83	31	19	31
D	32	100	32	68	32	4	32
F	33	102	33	70	33	6	33
G	34	103	34	71	34	7	34
H	34	104	35	72	35	8	35
J	36	106	36	74	736	10	36
K	37	107	37	75	37	11	37
L	38	108	38	76	38	12	38
;	39	59	39	58			39
'	40	39	40	34			40
Enter	28	13	28	13	28	10	28

Bottom Keyboard Row

<u>Key Cap</u>	<u>Scan</u>	<u>ASCII</u>	<u>Shift-Scan</u>	<u>Shift-ASCII</u>	<u>Ctrl-Scan</u>	<u>Ctrl-ASCII</u>	<u>Alt-Scan</u>
Z	44	122	44	90	44	26	44
X	45	120	45	88	45	24	45
C	46	99	46	67	46	3	46
V	47	118	47	86	47	22	47
B	48	98	48	66	48	2	48
N	49	110	49	78	49	14	49
M	50	109	50	77	50	13	50
,	51	44	51	60			51
.	52	46	52	62			52
/	53	47	53	63			53
Space	57	32	57	32	57	32	57

Function Keys

<u>Key Cap</u>	<u>Scan</u>	<u>ASCII</u>	<u>Shift-Scan</u>	<u>Shift-ASCII</u>	<u>Ctrl-Scan</u>	<u>Ctrl-ASCII</u>	<u>Alt-Scan</u>
F1	59	0	84	0	94	0	104
F2	60	0	85	0	95	0	105
F3	61	0	86	0	96	0	106
F4	62	0	87	0	97	0	107
F5	63	0	88	0	98	0	108
F6	64	0	89	0	99	0	109
F7	65	0	90	0	100	0	110
F8	66	0	91	0	101	0	111
F9	67	0	92	0	102	0	112
F10	68	0	93	0	103	0	113
F11	133	0	135	0	137	0	139
F12	134	0	136	0	138	0	140

Numeric Key Pad

<u>Key Cap</u>	<u>Scan</u>	<u>ASCII</u>	<u>Shift-Scan</u>	<u>Shift-ASCII</u>	<u>Ctrl-Scan</u>	<u>Ctrl-ASCII</u>	<u>Alt-Scan</u>
/	224	47	224	47	149	0	164
*	55	42	55	42	150	0	55
-	74	45	74	45	142	0	74
7	71	0	71	55	119	0	
8	72	0	72	56	141	0	
9	73	0	73	57	132	0	
+	78	43	78	43	144	0	78
4	75	0	75	52	115	0	
5	76	0	76	53	143	0	
6	77	0	77	54	116	0	
1	79	0	79	49	117	0	
2	80	0	80	50	145	0	
3	81	0	81	51	118	0	
0	82	0	83	46	147	0	
Del	83	0	83	46	147	0	
Enter	224	13	224	13	224	10	166

Cursor Key Pad

Key Cap	Scan	ASCII	Shift-Scan	Shift-ASCII	Ctrl-Scan	Ctrl-ASCII	Alt-Scan
Home	71	224	71	224	119	224	151
Up	72	224	72	224	141	224	152
PgUp	73	224	73	224	132	224	153
Left	75	224	75	224	115	224	155
Right	77	224	77	224	116	224	157
End	79	224	79	224	117	224	159
Down	80	224	80	224	145	224	160
PgDn	81	224	81	224	118	224	161
Ins	82	224	82	224	146	224	162
Del	83	224	83	224	147	224	163

8.4.4 ANSI X3.64 Command Reference

The command processor's support for ANSI Std X3.64 allows you to manipulate the cursor, screen color, and other display attributes through sequences of special characters embedded in the text you display on the screen. These sequences are called "ANSI commands". (For a general description of this feature, see [ANSI Support](#)^[64].)

The command processor supports most common ANSI X3.64 screen commands, but does not provide the complete set of options supported by many operating system or third-party replacement ANSI X3.64 drivers (for example, 4NT and Take Command do not support ANSI X3.64 key substitutions). This section is a quick-reference to the ANSI X3.64 commands supported by 4NT and Take Command.

ANSI X3.64 support within the command processor can be enabled or disabled with the [ANSI](#)^[99] directive in the [INI file](#)^[89], the corresponding option on the [Colors tab](#)^[88] of the [configuration dialogs](#)^[82], or the [SETDOS](#)^[283] /A command. You can test whether ANSI X3.64 support is enabled with the [_ANSI](#)^[348] internal variable.

An ANSI X3.64 command string consists of three parts:

<ESC>[The ASCII character ESC, followed by a left bracket. These two characters must be present in all ANSI X3.64 strings.
parameters	Optional parameters for the command, usually numeric. If there are multiple parameters, they are separated by semicolons.
command	A single-letter command. (Case sensitive!)

For example, to position the cursor to row 7, column 12 the ANSI X3.64 command is:

```
<ESC>[7;12H
```

The **parameters** part of this command is "7;12" and the **command** part is "H".

To transmit ANSI X3.64 commands to the screen with the command processor you can use the [ECHO](#)^[198] command. The ESC character can be generated by inserting it into the string directly (if you are putting the string in a batch file and your editor will insert such a character), or by using the command processor's internal ["escape" character](#)^[27] (defaults: caret, [^]) followed by a lower-case "e".

For example, the sequence shown above could be transmitted from a batch file with either of these commands (the first uses an ESC character directly, represented below by "ESC"; the second uses ^e):

```
echo <ESC>[7;12H
echo ^e[7;12H
```

You can also include ANSI X3.64 commands in your [prompt](#)^[262], using \$e to transmit the <ESC> character.

Commands

The internal ANSI X3.64 interpreter of **4NT** and **Take Command** supports the subset of X3.64 commands below. Variable parameters are shown in lower-case italics, e.g., *row* and *attr*, and must be replaced with the appropriate decimal numeric value when using the commands. Default value for *row*, *rows*, *col*, and *cols* is 1.

<ESC>[<i>rows</i> A	Cursor up by <i>rows</i>
<ESC>[<i>rows</i> B	Cursor down by <i>rows</i>
<ESC>[<i>cols</i> C	Cursor right by <i>cols</i>
<ESC>[<i>cols</i> D	Cursor left by <i>cols</i>
<ESC>[<i>row</i> ; <i>col</i> H	Set cursor position (top left is row 1, column 1)
<ESC>[2J	Clear whole screen
<ESC>[K	Clear from cursor to end of line
<ESC>[<i>row</i> ; <i>col</i> f	Set cursor position, same as "H" command
<ESC>[<i>attr1</i> ; <i>attr2</i> ;... <i>m</i>	Set display attributes; see table of attribute values below
<ESC>[s	Save cursor position (may not be nested)
<ESC>[u	Restore saved cursor position

Display Attributes

The attribute values used for the **m** command are:

- 0** Restore all attributes to default
- 1** High intensity (bright) foreground color
- 2** Normal intensity foreground color
- 30..37** Foreground color
- 40..47** Background color

Foreground Code	Background Code	Color
30	40	Black
31	41	Red
32	42	Green
33	43	Yellow
34	44	Blue
35	45	Magenta
36	46	Cyan
37	47	White

Attribute settings are cumulative, and are independent of order (except code **0**, reset to default), and they can be combined into a single command (using the `;` concatenation operator), or split into separate commands.

Examples

Set bright red foreground without changing background:

```
echo %=e[31;1m
```

Set the display to bright cyan on blue, and clear the screen:

```
echo %=e[44;36;1m%=e[2J
```

Set up a prompt which saves the cursor position, displays the date and time on the top line in bright white on magenta, and then restores the cursor position and sets the color to bright cyan on blue, and displays the standard prompt:

```
prompt $e[s$e[1;1f$e[45;37;1m$e[K$d $t$e[u$e[44;36;1m$p$g
```

8.5 Miscellaneous Reference Information

- [Executable Files and File Searches](#)^[458]
- [Primary and Secondary Shells](#)^[461]
- [Colors and Color Names](#)^[461]
- [Keys and Key Names](#)^[464]
- [Popup Windows](#)^[464]
- [Windows System Errors](#)^[465]

8.5.1 Executable Files & File Searches

When 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.

If the command processor cannot find an executable 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, then for a *.BAT* file, and finally for a *.REX* file. If the command processor finds such a file, it then reads each line in the file as a new command.

You can change the list of extensions that are considered "executable", and the order in which they are searched, with the [PATHEXT](#)^[34] environment variable, and the related [PathExt](#)^[108] directive in the [.INI file](#)^[89]. PATHEXT is supported for compatibility reasons but should not generally be used as a substitute for [executable extensions](#)^[48], which are more flexible.

Note: 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](#)^[48]. If an executable extension is found, the command processor runs the program specified when the association was defined. If no executable extension is found, 4NT will look for a direct association for the extension in the registry and insert the associated string (usually the name of an application) at the beginning of the command line, then call the Windows *CreateProcess* API to execute that command. If the *CreateProcess* call fails, or if no association was found in the registry, then 4NT then calls the *ShellExec* Windows API. The command processor has no control over which action the above Windows APIs will take when presented with a file name. If you are concerned about what Windows might do with an "unknown" extension, create a specific executable extension.

The command processor first performs 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](#)^[254] command. You can use the [ESET](#)^[201] command 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 98 and ME, the command processor also searches the *WINDOWS* directory then the *WINDOWS\SYSTEM* directory; under Windows NT, 2000, XP, and 2003, the order is reversed, and the command processor searches the *WINDOWS\SYSTEM32* directory followed by the *WINDOWS* directory. (The actual directory names may be different on your system. The command processor will determine the correct names for the "Windows" and "Windows System" directories and use them.) This part of the search procedure conforms with the traditional search sequences used under each Windows operating system.

Note: If the file is not found on the PATH, the command processor then checks 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 or Windows file association) in the current subdirectory, then in the Windows directories if appropriate (see above), then in each directory in the search path, and then in the **App Paths** area of the registry. (You can change the search order so the current directory is not searched first; see the [PATH](#)^[254] command for details.)

If you include an extension as part of the command name, the command processor only searches 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:\wp\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](#)^[112] to Yes in the [.INI File](#)^[89].

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.

If you specify a file name including extension, and the file exists in the current directory (or you specify a path), but the file does not have an extension known to the command processor (`.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:

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

and the command processor will request that Windows start the application for you. See [Windows File Associations](#)^[461] for additional details on how to control Windows file associations in the command processor.

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):

Command	Command Processor Search Sequence
WP	Standard search for any executable file whose base name is <i>WP</i> .
WP.EXE	Standard search for <i>WP.EXE</i> ; will not find files with other extensions.
C:\WP\WP	Looks in the <i>C:\WP</i> directory for any executable file whose base name is <i>WP</i> . Does not check the standard search directories.
C:\WP\WP.EXE	Looks only for the file <i>C:\WP\WP.EXE</i> .
LAB.DOC	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.
C:\L\LAB.DOC	Looks only for the file <i>C:\L\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 or Windows file association 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 the [ALIAS](#)^[138] command 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.

Also see the [WHICH](#)^[315] command.

8.5.1.1 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 passes the command name to Windows to see if Windows can find an association for it.

The command processor offers 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](#)^[145] command modifies or displays the associations between extensions and file types in the Windows registry. The [FTYPE](#)^[217] command modifies or displays the default command used to "open" a file of a specified type.

Executable extensions defined in the command processor always take precedence over file associations defined in Windows. For example, if you associate the *.TXT* extension with your own editor using a command processor 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 the command processor.

Also see: [START](#)^[291], [ASSOC](#)^[145], [FTYPE](#)^[217], [Executable Extensions](#)^[48], [Executable Files and File Searches](#)^[458].

8.5.2 Primary & Secondary Shells

The command processor is commonly referred to as the **shell**. These help files often make reference to **primary shell** and **secondary shell**.

A **primary shell** is an instance of the command processor which is **not** the child of another instance of the **same** command processor. It is the "first" copy started by *some other process*. A common type of *primary shell* would be an instance of 4NT or Take Command launched by invoking (clicking on) a Windows shortcut. Note that a copy of Take Command launched from a 4NT prompt would also be a *primary shell* since its parent will be "some other process", not a previous Take Command instance.

A **Secondary Shell** is an instance of the command processor launched from a previous instance of that same command processor. For example, if you are at a 4NT prompt and invoke "d:\path\4nt.exe", then that new copy will be a Secondary Shell. In general, a Secondary shell inherits most settings from the parent Primary shell. A Secondary Shell can in turn launch additional Secondary shells, but only the original instance is a Primary Shell.

Under Windows NT, 2000, XP, and 2003, there is no "default command processor" per se and therefore no global Primary Shell. Each instance of 4NT or Take Command invoked by clicking on a shortcut or entering its name in a "run" dialog is a Primary. To facilitate communication between those several "parallel" instances of the command processor, we provide the ability to share resources via the [LocalAliases](#)^[96], [LocalHistory](#)^[96], [LocalDirHistory](#)^[96] and [LocalFunctions](#)^[96] directives and the [SHRALIAS](#)^[290] command. Any 4NT or Take Command instance launched from an existing copy 4NT or Take Command will be a Secondary. Note that Secondaries can be invoked explicitly (e.g. "d:\path\4nt.exe" from a 4NT prompt) or implicitly (to execute a pipe, for example).

8.5.3 Colors, Color Names & Codes

You can use color names in several of the directives in the [.INI file](#)^[89] and in many commands. The general form of a color specification is:

```
[BRiGht] fg ON [BRiGht] bg
```

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

Color Names

Color names as well as the attribute name **BRiGht** may be shortened to their first three letters. The available color names, shown below on approximations of the 8 basic background colors, are: **BLAck**, **BLUe**, **GREen**, **CYAn**, **RED**, **MAGenta**, **YELlow**, **WHItE**.

	BLUe	GREen	CYAn	RED	MAGenta	YELlow	te
BLAck	BLU	GREen	CYAn	RED	MAGenta	YELlow	WHItE
BLAck	BLUe	GRE	CYAn	RED	MAGenta	YELlow	WHItE
BLAck	BLUe	GREen	CYA	RED	MAGenta	YELlow	WHItE
BLAck	BLUe	GREen	CYAn	RED	MAGenta	YELlow	WHItE
BLAck	BLUe	GREen	CYAn	RED	MAG	YELlow	WHItE
BLAck	BLUe	GREen	CYAn	RED	MAGenta	YEL	WHItE
BLAck	BLUe	GREen	CYAn	RED	MAGenta	YELlow	WHI

Note: The colors (if any) represented by your viewer in the above table do not necessarily match the actual rendition provided by your display hardware and drivers at a command processor prompt. **BRiGht** backgrounds are generally always enabled under Windows.

Color Codes

You can also specify colors by numeric code (see table below) instead of by name. The numeric form is most useful in potentially long directives such as `ColorDIR[113]`, where using color names may take too much space. The codes are **decimal** numbers, with the codes for bright colors larger than those of the corresponding normal colors by 8.

The `COLOR[163]` command also supports the CMD.EXE style color specification `bf`, where *b* and *f* are CMD's codes for *background* and *foreground* colors, respectively, shown in the CMD columns of the table below. The numeric values of these codes are the same as the JP Software codes, but they are represented in **hexadecimal**.

ANSI X3.64 color codes are also shown in the table. Note that X3.64 support for the *bright* attribute is restricted to foreground. Note that the color codes are **decimal**, and the codes for *background* colors are larger than those of the corresponding *foreground* colors by 10.

SCREEN COLOR		JPSoft name	JP color codes (decimal)		CMD.EXE codes* (hexadecimal)		ANSI X3.64 codes (decimal)	
<i>normal</i>	<i>bright</i>		<i>normal</i>	<i>bright</i>	<i>normal</i>	<i>bright</i>	<i>foreground</i>	<i>background</i>
black	gray	BLA ck	0	8	0	8	30	40
blue	blue	BLU e	1	9	1	9	34	44
green	green	GRE en	2	10	2	A	32	42
cyan	cyan	CYA n	3	11	3	B	36	46
red	pink	RED	4	12	4	C	31	41
magenta	magenta	MAG enta	5	13	5	D	35	45
brown	yellow	YEL low	6	14	6	E	33	43
white	white	WHI te	7	15	7	F	37	47

* **Note:** The numeric values of the CMD.EXE and native color codes are identical, the difference is in representation only.

Use one number to substitute for the **[BR]ight fg** portion of the color name, and a second to substitute for the **[BR]ight bg** portion. For example, instead of **bright white on red** you could use **15 on 4** to save space in a ColorDir specification.

The **@OPTION**^[40h] function returns the value of color directives by combining both foreground and background into a **single number** (0-255) using the following logic:

$$\text{foreground value} + (\text{background value} * 16) = \text{code}$$

For example, *bright white on red* (15 on 4) can be expressed as:

$$15 + (4 * 16) = 79$$

The following shows a batch file which can be used to "translate" a combined numeric color code:

```

1. @echo off
2. setlocal
3. function x=`%if[%1 gt 8,bri ,]`%word[%@eval[%1 %% 8],bla blu gre cya
   red mag yel whi]`
4. :loop
5. input /c /d %=nColor code? %%c
6. if %c gt 255 .or. %c lt 0 quit
7. set f=%@eval[%c %% 16] %+ set b=%@eval[%c \ 16]
8. echos The color code %c is "%f on %b" ("%x[%f] on %x[%b]")
9. goto loop

```

(the line numbers are not part of the batch file and are merely to clarify any possible confusion caused by wrapping in your viewer).

Color Errors

A standard color specification allows sixteen foreground and eight background colors (sixteen if bright backgrounds are enabled). However, most video adapters and monitors do not provide true renditions of certain colors. For example, most users see normal "yellow" as brown, and bright yellow as yellow; many also see normal red as red, and "bright red" as pink. Color errors are often much worse when running in windowed mode, 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, and they cannot be corrected using the command processor color specifications.

8.5.4 Keys & Key Names

Key names are used to define keystroke [aliases](#)^[138], in several [.INI](#)^[89] directives, and with the [KEYSTACK](#)^[236] command. The format of a key name is the same in all three uses:

```
[Prefix-]Keyname
```

The possible key names are: **A - Z, 0 - 9, F1 - F12, Esc, Tab, Bksp, Enter, Ins, Del, Up, Down, Left, Right, PgUp, PgDn, Home** and **End**. 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.

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

```
Alt      followed by A - Z, 0 - 9, F1 - F12, or Bksp
Ctrl     followed by A - Z, F1 - F12, Tab, Bksp, Enter, Up, Down, Left,
        Right, Home, End, PgUp, PgDn, Ins, or Del
Shift    followed by F1 - F12 or Tab
```

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

```
Alt-F10  This is okay
Alt F10  The space will cause an error
```

Some keys are intercepted by Windows and are not passed on to the command processor. For example, Alt-Esc and Ctrl-Esc typically pop up a tasklist or are used in switching among multiple tasks. Keys which are intercepted by the operating system (including menu accelerators, *i.e.* **Alt** plus another key) generally cannot be assigned to [aliases](#)^[138] or with [key mapping directives](#)^[114], because the command processor never receives these keystrokes and therefore cannot act on them.

The above comments are based on common 101/102-key US-style keyboards. Some key combinations might not be available on some keyboards.

8.5.5 Popup Windows

Several features of the command processor 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](#)^[56], or a list created by the [SELECT](#)^[275] command or the [@SELECT](#)^[407] 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 where that window is discussed in detail.

You can control the position and size of most popup windows with the [configuration dialogs](#)^[82], or with the [PopupWinLeft](#)^[108], [PopupWinTop](#)^[108], [PopupWinWidth](#)^[108], and [PopupWinHeight](#)^[108] directives in the [.INI file](#)^[89]. A few popup windows (e.g., the extended directory search window) have their own specific [.INI](#) directives, and corresponding separate choices in the configuration dialogs. You can also change the keys used in most popup windows with [key mapping directives](#)^[114] in the [.INI](#) file.

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

Up Arrow ^[118]	Move the selection bar up one line.
Down Arrow ^[117]	Move the selection bar down one line.
Left Arrow ^[117]	Scroll the display left 1 column, if it is a scrolling display (<i>i.e.</i> if it has a horizontal scrollbar).
Right Arrow ^[118]	Scroll the display right 1 column, if it is a scrolling display (<i>i.e.</i> if it has a horizontal scrollbar).
PgUp ^[122]	Scroll the display up one page.
PgDn ^[122]	Scroll the display down one page.
Ctrl-PgUp ^[123] or Home ^[123]	Go to the beginning of the list.
Ctrl-PgDn ^[123] or End ^[123]	Go to the end of the list.
Esc ^[117]	Close the window without making a selection.
Enter ^[123]	Select the current item and close the window.

Note: The keystrokes shown above are the defaults values. See [Key Mapping Directives](#) ^[114] for details on how to assign different keystrokes.

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 search 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.

8.5.6 Windows System Errors

This list shows a typical set of error and system messages returned by Windows. The numbers are what might be returned in internal variable [_SYSERR](#) ^[356]. and the corresponding text is that provided by Microsoft.

Note: *This particular list happens to be from a Windows 2000 (US) configuration and your own installation will probably provide different messages, but most DOS and Windows versions use many of the same entries.*

<u>Code</u>	<u>Error Message</u>
0	The operation completed successfully.
1	Incorrect function.
2	The system cannot find the file specified.
3	The system cannot find the path specified.
4	The system cannot open the file.
5	Access is denied.
6	The handle is invalid.
7	The storage control blocks were destroyed.
8	Not enough storage is available to process this command.
9	The storage control block address is invalid.
10	The environment is incorrect.
11	An attempt was made to load a program with an incorrect format.
12	The access code is invalid.
13	The data is invalid.
14	Not enough storage is available to complete this operation.
15	The system cannot find the drive specified.
16	The directory cannot be removed.
17	The system cannot move the file to a different disk drive.
18	There are no more files.
19	The media is write protected.
20	The system cannot find the device specified.
21	The device is not ready.

- 22 The device does not recognize the command.
- 23 Data error (cyclic redundancy check).
- 24 The program issued a command but the command length is incorrect.
- 25 The drive cannot locate a specific area or track on the disk.
- 26 The specified disk or diskette cannot be accessed.
- 27 The drive cannot find the sector requested.
- 28 The printer is out of paper.
- 29 The system cannot write to the specified device.
- 30 The system cannot read from the specified device.
- 31 A device attached to the system is not functioning.
- 32 The process cannot access the file because it is being used by another process.
- 33 The process cannot access the file because another process has locked a portion of the file.
- 36 Too many files opened for sharing.
- 38 Reached the end of the file.
- 39 The disk is full.
- 50 The network request is not supported.
- 51 The remote computer is not available.
- 52 A duplicate name exists on the network.
- 53 The network path was not found.
- 54 The network is busy.
- 55 The specified network resource or device is no longer available.
- 56 The network BIOS command limit has been reached.
- 57 A network adapter hardware error occurred.
- 58 The specified server cannot perform the requested operation.
- 59 An unexpected network error occurred.
- 60 The remote adapter is not compatible.
- 61 The printer queue is full.
- 62 Space to store the file waiting to be printed is not available on the server.
- 63 Your file waiting to be printed was deleted.
- 64 The specified network name is no longer available.
- 65 Network access is denied.
- 66 The network resource type is not correct.
- 67 The network name cannot be found.
- 68 The name limit for the local computer network adapter card was exceeded.
- 69 The network BIOS session limit was exceeded.
- 70 The remote server has been paused or is in the process of being started.
- 71 No more connections can be made to this remote computer at this time because there are already as many connections as the computer can accept.
- 72 The specified printer or disk device has been paused.
- 80 The file exists.
- 82 The directory or file cannot be created.
- 83 Fail on INT 24.
- 84 Storage to process this request is not available.
- 85 The local device name is already in use.
- 86 The specified network password is not correct.
- 87 The parameter is incorrect.
- 88 A write fault occurred on the network.
- 89 The system cannot start another process at this time.
- 100 Cannot create another system semaphore.
- 101 The exclusive semaphore is owned by another process.
- 102 The semaphore is set and cannot be closed.
- 103 The semaphore cannot be set again.
- 104 Cannot request exclusive semaphores at interrupt time.
- 105 The previous ownership of this semaphore has ended.
- 107 The program stopped because an alternate diskette was not inserted.
- 108 The disk is in use or locked by another process.
- 109 The pipe has been ended.
- 110 The system cannot open the device or file specified.

- 111 The file name is too long.
- 112 There is not enough space on the disk.
- 113 No more internal file identifiers available.
- 114 The target internal file identifier is incorrect.
- 117 The IOCTL call made by the application program is not correct.
- 118 The verify-on-write switch parameter value is not correct.
- 119 The system does not support the command requested.
- 120 This function is not supported on this system.
- 121 The semaphore timeout period has expired.
- 122 The data area passed to a system call is too small.
- 123 The filename, directory name, or volume label syntax is incorrect.
- 124 The system call level is not correct.
- 125 The disk has no volume label.
- 126 The specified module could not be found.
- 127 The specified procedure could not be found.
- 128 There are no child processes to wait for.
- 130 Attempt to use a file handle to an open disk partition for an operation other than raw disk I/O.
- 131 An attempt was made to move the file pointer before the beginning of the file.
- 132 The file pointer cannot be set on the specified device or file.
- 133 A JOIN or SUBST command cannot be used for a drive that contains previously joined drives.
- 134 An attempt was made to use a JOIN or SUBST command on a drive that has already been joined.
- 135 An attempt was made to use a JOIN or SUBST command on a drive that has already been substituted.
- 136 The system tried to delete the JOIN of a drive that is not joined.
- 137 The system tried to delete the substitution of a drive that is not substituted.
- 138 The system tried to join a drive to a directory on a joined drive.
- 139 The system tried to substitute a drive to a directory on a substituted drive.
- 140 The system tried to join a drive to a directory on a substituted drive.
- 141 The system tried to SUBST a drive to a directory on a joined drive.
- 142 The system cannot perform a JOIN or SUBST at this time.
- 143 The system cannot join or substitute a drive to or for a directory on the same drive.
- 144 The directory is not a subdirectory of the root directory.
- 145 The directory is not empty.
- 146 The path specified is being used in a substitute.
- 147 Not enough resources are available to process this command.
- 148 The path specified cannot be used at this time.
- 149 An attempt was made to join or substitute a drive for which a directory on the drive is the target of a previous substitute.
- 150 System trace information was not specified in your CONFIG.SYS file, or tracing is disallowed.
- 151 The number of specified semaphore events for DosMuxSemWait is not correct.
- 152 DosMuxSemWait did not execute; too many semaphores are already set.
- 153 The DosMuxSemWait list is not correct.
- 154 The volume label you entered exceeds the label character limit of the target file system.
- 155 Cannot create another thread.
- 156 The recipient process has refused the signal.
- 157 The segment is already discarded and cannot be locked.
- 158 The segment is already unlocked.
- 159 The address for the thread ID is not correct.
- 160 The argument string passed to DosExecPgm is not correct.
- 161 The specified path is invalid.
- 162 A signal is already pending.
- 164 No more threads can be created in the system.
- 167 Unable to lock a region of a file.
- 170 The requested resource is in use.
- 173 A lock request was not outstanding for the supplied cancel region.
- 174 The file system does not support atomic changes to the lock type.
- 180 The system detected a segment number that was not correct.

- 183 Cannot create a file when that file already exists.
- 186 The flag passed is not correct.
- 187 The specified system semaphore name was not found.
- 196 The operating system cannot run this application program.
- 197 The operating system is not presently configured to run this application.
- 199 The operating system cannot run this application program.
- 200 The code segment cannot be greater than or equal to 64K.
- 203 The system could not find the environment option that was entered.
- 205 No process in the command subtree has a signal handler.
- 206 The filename or extension is too long.
- 207 The ring 2 stack is in use.
- 208 The global filename characters, * or ?, are entered incorrectly or too many global filename characters are specified.
- 209 The signal being posted is not correct.
- 210 The signal handler cannot be set.
- 212 The segment is locked and cannot be reallocated.
- 214 Too many dynamic-link modules are attached to this program or dynamic-link module.
- 215 Cannot nest calls to LoadModule.
- 230 The pipe state is invalid.
- 231 All pipe instances are busy.
- 232 The pipe is being closed.
- 233 No process is on the other end of the pipe.
- 234 More data is available.
- 240 The session was canceled.
- 254 The specified extended attribute name was invalid.
- 255 The extended attributes are inconsistent.

CAUTION: This is a **large** topic (>650 KB). The remainder of this list has been omitted in the PDF version of this file. Consider viewing the copy in your local help file (*jphelp.chm*) instead!

d.

8.6 Glossary

The glossary contains basic definitions for over 200 common terms listed in alphabetical order and is divided into sections by the first letter of each term. Concepts directly relevant to our command processors are typically discussed in more detail in other areas of this help file.

Select the glossary section you wish to review:

[4](#)^[468]
[A](#)^[469]
[B](#)^[470]
[C](#)^[471]
[D](#)^[474]
[E](#)^[476]
[F](#)^[477]
[G](#)^[478]
[H](#)^[479]
[I](#)^[479]
[K](#)^[480]
[L](#)^[480]
[M](#)^[481]
[N](#)^[481]
[O](#)^[481]
[P](#)^[482]
[Q](#)^[482]
[R](#)^[483]
[S](#)^[483]
[T](#)^[484]
[U](#)^[485]
[V](#)^[485]
[W](#)^[486]
[X](#)^[486]

8.6.1 Glossary - 4

4EXIT - A program which is executed whenever 4NT exits. See [Automatic Startup and Termination Programs](#)^[6] for more details.

4START - A program which is executed whenever 4NT starts. See [Automatic Startup and Termination Programs](#)^[6] for more details.

8.6.2 Glossary - A

Access Date - In the MS Windows file systems it is the latter of the dates on which a file was created or last accessed *for reading*. The [VFAT](#)^[485] and [NTFS](#)^[487] file systems save this file property.

Access Time - In the MS Windows file systems it is the time of day when the latter of the events of creating the file or last accessing the file *for reading* occurred. [NTFS](#)^[487] saves this file property.

Advanced Power Management (APM) - A standardized system used by manufacturers of battery-powered computers to control system power management, including shutdown of unused components or of the entire system based on usage patterns. **APM** can also report the source of system power (AC or battery), the battery status, and the remaining battery life.

Alias - A command processor feature which allows you to create shorthand name for a command or series of commands. For details see [Aliases](#)^[319].

Alias Argument - Same as [Alias Parameter](#)^[469].

Alias Parameter - A numbered variable (e.g. %2) included in an alias definition, allowing a different value to be used in the alias each time it is executed.

Alternate File Name - Same as [Short File Name](#)^[483].

American National Standards Institute (ANSI) - An organization which sets voluntary standards for a large variety of industrial products from boilers to photographic films, including computer-related systems, and is the U.S.A. representative in [ISO](#)^[479]. It is composed of various professional organizations, including EIA and IEEE, many of which also define standards. **ANSI's** previous names include **American Standards Association (ASA)** and **United States of America Standards Institute (USASI)**.

AND - A logical combination of two **true** or **false** conditions. The result is **true** if and only if both conditions are **true**, otherwise it is **false**. See table below.

<i>condition 1</i>	<i>condition 2</i>	<i>result</i>
<i>false</i>	<i>false</i>	<i>false</i>
<i>false</i>	<i>true</i>	<i>false</i>
<i>true</i>	<i>false</i>	<i>false</i>
<i>true</i>	<i>true</i>	<i>true</i>

ANSI - [American National Standards Institute](#)^[469]. The acronym **ANSI** in software development is often used as a short-hand reference to the standard [ANSI X3.64](#)^[469].

ANSI X3.64 - A standard specifying sequences of characters which control colors on the screen, manipulate the cursor and screen contents, and redefine keys. 4NT and Take Command include support for a selected subset. This support may be enabled or disabled at any time. For more details see [ANSI X3.64 Command Reference](#)^[456].

Append - Concatenation of one file or string onto the end of another. Not related to the MS-DOS/Windows external command named **APPEND** (not emulated by **4NT** or **TC**).

APM - See [Advanced Power Management](#)^[469].

Apostrophe - the character ' used in the English language to denote genitives and most contractions. Some programming languages, designed in the days when keypunches did not have [quote mark](#)^[482]s,

use it to delimit (denote the beginning and the end of) text strings to be used literally, instead of as part of the code. Due to the similarity of quoting text in natural language writings, they had been incorrectly referred to as "[quote mark](#)^[482]s".

Application - A program run from the command prompt or a batch file. Used broadly to mean any program other than the command processor itself; and more narrowly to mean a program with a specific purpose such as a spreadsheet or word processing program, as opposed to a utility.

Archive - 1) A file attribute indicating that the file has been modified since the last backup (most backup programs clear this attribute). 2) A single file (such as a **.ZIP** file) which contains a number of other files, optionally in compressed form.

Argument - Same as [Parameter](#)^[482].

ASCII - Acronym for *American National Standard Code for Information Interchange*, defined in the standard [ANSI](#)^[469] **INCITS 4-1986 (R1997) Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII) (formerly ANSI X3.4-1986 (R1997))**. This standard specifies a set of 128 characters (control characters and graphics characters such as letters, digits, and symbols) with their coded representation for use in information interchange among information processing systems, communication systems, and associated equipment. The ASCII versions of **4NT** and **TC** use this code for input, for output, and for the internal representation of command lines and batch files. See [ASCII file](#)^[469].

ASCII File - A file containing ASCII text, as opposed to a binary file which may contain numbers, or other information that cannot be sensibly interpreted as text. Often used in place of [text file](#)^[484].

Attribute - A property, or indication of a property, of a file which may be set (present) or cleared (absent). Most attributes can be changed, but some cannot. The standard modifiable attributes are [Read-Only](#)^[483], [Hidden](#)^[479], [System](#)^[483], and [Archive](#)^[469]. Unmodifiable attributes include [Directory](#)^[474] and [Volume Label](#)^[485].

AUTOEXEC - A batch file which is executed automatically each time Windows 98 starts. It defaults to **C:\AUTOEXEC.BAT**, but this can be modified in the **SHELL=** directive in [CONFIG.SYS](#)^[471]. Some Windows XP configurations can optionally extract [PATH](#)^[482] information from that file when found. The equivalent for DOS emulator sessions under Windows NT/2000/XP is **AUTOEXEC.NT**.

Automatic Directory Change - A command processor feature which allows you to change directories by typing the directory name terminated in a backslash [****] at the command prompt, without first entering a command.

Automatic Programs - Programs automatically executed when the OS starts, or the command processor starts or terminates: [AUTOEXEC](#)^[469], [4START](#)^[468], [4EXIT](#)^[468], [TCSTART](#)^[484], and [TCEXIT](#)^[484]. See [Automatic Startup and Termination Programs](#)^[6] for more details.

8.6.3 Glossary - B

Base Name - The file name without a drive, path, or extension. For example, in the file name **C:\DIR1\LETTER.DAT** the base name is **LETTER**.

Basic Input Output System (BIOS) - Software which provides basic low-level control of devices required to operate the system, such as the keyboard, floppy disk, and screen..It includes a hardware scheme to perform the [initial program load](#)^[479]. On most systems all of the BIOS is stored in read-only memory inside the PC.

BAT File - Same as [Batch File](#)^[470].

Batch File - A [text file](#)^[484] containing a sequence of commands for the command processor to execute. Batch files are used to save command sequences so that they can be reexecuted at any time, transferred to another system, etc. The extension of a batch file may be **.BAT**, **.CMD**, or **.BTM**, depending on the operating system and command processor you are using.

Batch File Argument - Same as [Batch File Parameter](#)^[470].

Batch File Parameter - A numbered variable (e.g. %2) used within a [batch file](#)^[470], allowing a different value to be used at that spot in the file each time it is executed.

Batch Program - A more descriptive name for [Batch File](#)^[470].

Binary File - A file containing information which does not represent or cannot sensibly be interpreted as text. See also [ASCII File](#)^[469], [text file](#)^[484].

BIOS - [Basic Input Output System](#)^[470]..

Block Device - A physical device for input or output, which interchanges data with the computer in large blocks. XXIth Century PCs can perform such transfers of data concurrently with other activities. Examples of **block devices** include disk-, tape-, and CD-ROM drives. Cf. [Character Device](#)^[471].

Boot - Ungrammatical contraction of [Bootling](#)^[470], shortened from [Bootstrapping](#)^[470].

Boot Directory - The current directory at the time the system is started, usually the [root directory](#)^[483] of the [boot drive](#)^[470].

Boot Drive - The disk drive that the operating system is loaded from during [IPL](#)^[479], usually A: (the floppy disk) or C: (the hard disk), or on some modern systems a CD or DVD drive. In some diskless workstations it may be a network drive.

Bootling - shortened form of [Bootstrapping](#)^[470].

Bootstrapping - Slang for [Initial Program Load](#)^[479], from the book "Baron Munchausen's Narrative of his Marvellous Travels and Campaigns in Russia", by Rudolf Erich Raspe (London, England, 1786), in particular, the story of the Baron's astounding ability to pull himself from the ocean by his own bootstraps. See also [Cold Boot](#)^[471], [Cold Reboot](#)^[471], [Reboot](#)^[483], and [Warm Reboot](#)^[486].

Break - A signal sent to a program to tell it to halt what it is doing. The **Ctrl-C** key or **Ctrl-Break** key is used to send this signal. Some external commands abort when they receive a break signal; others return to a previous screen or menu, or abort the current operation.

BTM File - A special type of [batch file](#)^[470], unique to JP Software products, which is loaded into a [buffer](#)^[470] in memory to speed up its execution.

Buffer - An area of memory set aside for temporary storage of information moved between concurrent programs or between programs and external media. For example, disk buffers are used to save information as it is transferred between your program and the disk, and the keyboard buffer holds keystrokes until a program can use them. The command processor uses many buffers for its internal operations, e.g., to store [alias](#)^[469]es.

8.6.4 Glossary - C

Carriage Return (CR) - A control character ([ASCII](#)^[446]: 13) specifying that the next character is to be displayed at the leftmost position of the current line. In older mechanical character printing devices the *carriage* was moved to cause printing in different character positions of the same line. In a PC the **CR** code is generated by pressing the **Enter** key on the keyboard, and stored in most [text file](#)^[484]s at the

end of each line. See also: [ASCII](#)^[446], [End of Line](#)^[476].

CD-ROM File System (CDFS) - The file system which supports CD-ROM drives. This is typically implemented as a distinct file system in 32-bit operating systems such as Windows XP. On other platforms it is implemented as a component of or addition to the underlying general file system for disk drives.

Central Processing Unit (CPU) - The part of the computer which historically performed all logic and most calculations. In PC-compatible systems, the CPU is on a single microprocessor chip. Many high-performance computers, including PCs, have more than one processing unit, often organized in such a way that none qualifies as the "central" processing unit.

Character Code - The code representing a specific character in the computer. See also: [ASCII](#)^[469], [Code Page](#)^[471], [Unicode](#)^[485].

Character Device - A physical device for input or output which must communicate with your computer one character at a time. Examples include the console, communications ports, and printers. Cf. [Block Device](#)^[470].

Character Mode - A display mode in which output is displayed one character at a time, and which cannot display graphics or pictures not included in the character font used. **Character Mode** displays are usually in a fixed font, typically with 80 columns in a line and 25 lines on the screen. Some systems allow you to increase or decrease the number of rows and columns to other fixed sizes. Cf. [Graphic Mode](#)^[478].

CMD File - A [batch file](#)^[470] designed for execution by some version of Microsoft's **CMD.EXE**.

CMDLINE - An [environment variable](#)^[476] used to extend the command line passed to another program beyond its normal length limits.

Code Page - A setting which tells Windows which character set to use, and how to retrieve and display date, time, and other information in the format appropriate to a particular country or region. See also [Country Settings](#)^[471].

Cold Boot - The process of starting the computer by turning on its power. See also [Boot](#)^[470], [Cold Reboot](#)^[471], [Reboot](#)^[483], [Warm Reboot](#)^[486]..

Cold Reboot - The process of restarting the computer in a way that physically resets most hardware devices, typically by pressing a reset button, or by emulating it utilizing a special feature of the [BIOS](#)^[470]. See also [Boot](#)^[470], [Cold Boot](#)^[471], [Reboot](#)^[483], [Warm Reboot](#)^[486]..

Command Completion - A command processor feature which allows you to recall a previous command by typing the first few letters of the command, then an up-arrow or down-arrow.

Command Echoing - A command processor feature which displays commands as they are executed. Echoing can be turned on and off.

Command Grouping - A command processor feature which allows you to enclose a group of several commands within parentheses, and have them treated as a single command for most purposes.

Command History - A command processor feature which retains the commands you have executed from the command prompt, so that they can be recalled, optionally modified, and reexecuted later.

Command History Window - A pop-up window used by the command processor to display the [command history](#)^[471], allowing you to choose a previous command to modify and/or execute.

Command Interpreter - Same as [Command Processor](#)^[471].

Command Line Expansion - The process the command processor goes through when it scans a command line and substitutes the appropriate actual values for aliases, alias parameters, batch file parameters, user defined functions, and environment variables. See also [Parsing](#)^[482].

Command Processor - A program which interprets commands entered interactively or stored in a file (referred to as a [batch file](#)^[470] in the environment discussed here), and executes other programs. Also called [Command Interpreter](#)^[471].

Command Prompt - An indication displayed on the screen by the command processor prompting you to enter another command to be executed.

Command Recall - See [Command History](#)^[471].

Command Separator - One or more characters used to separate multiple commands on the same command line. See [Multiple Commands](#)^[24].

Command Tail - The contents of the command line after removing the command name.

Compound Command - See [Multiple Commands](#)^[24].

Compression - A feature which compresses data as it is stored in a disk file, and expands it as it is read back, resulting in more efficient use of disk space. This feature may be provided by hardware, software, or a combination of both. The software may, but need not, be part of the operating system. [NTFS](#)^[481] 5 provides an attribute to indicate whether or not a file is compressed, and the versions of Windows NT which support NTFS 5 included the compression/expansion software. Accessing compressed files require slightly more processor time to perform the compression and expansion, but this is often compensated by the reduced data transfer time. More generally, compression is an approach to data storage which reduces repeated or redundant information in a set of data to minimize the space required to store it or the time required to transmit it.

COMSPEC - An [environment variable](#)^[476] which defines where to find the character-mode command processor to start a secondary shell under Windows 98. It is always set by **4NT** and **TC**. It is sometimes also used under Windows NT/2000/XP.

Condition - short for [conditional expression](#)^[31].

Conditional Commands - A command processor feature allowing commands to be executed or skipped depending on the results of a previous command. See also: [Exit Code](#)^[476].

Conditional Expression - See [Conditional Expression](#)^[31].

CONFIG.SYS - A file used under Windows 98 to specify which programs should be loaded when the system is started. The equivalent for DOS emulator sessions under Windows NT/2000/XP is **CONFIG.NT**.

Console - The PC keyboard and display.

Console Mode - See [Character Mode](#)^[471].

Control Character - A character which does not have a printable or displayable representation, but, when encountered in a stream of characters, requests the performance of a control action. In [ASCII](#)^[446] the codes for these characters are in the range [0,31] or it is 127; each character also has a two- or three-character symbol for representing it in plain text. The common PC keyboard for U.S. use has

special keys for some, i.e., **ESC**, **CR**, **BS**, **HT**, **DEL**. Each control character in [ASCII](#)^[446] can be generated by pressing the **Ctrl** key simultaneously with another key. For more information see [ASCII, Key Codes & ANSI Commands](#)^[445].

Coprocessor - Short for [Numeric Coprocessor](#)^[481].

Country Settings - The internal settings which tell the operating system how to interpret keyboard characters which vary from country to country, which character set to use, and how to retrieve and display date, time, and other information in the format appropriate to a particular country or region. See also [Code Page](#)^[471].

CPU - [Central Processing Unit](#)^[471].

CR - [Carriage Return](#)^[471]

CRC - [Cyclic Redundancy Code](#)^[471].

Critical Error - An error, usually related to a physical or hardware problem with input, output, or network access, which prevents a program from continuing.

Current Directory - The directory in which all file operations of a selected drive will take place unless otherwise specified by an explicit path. The current directory is typically displayed as part of the command prompt. Also called the **Current Working Directory**. In Windows 98 and earlier OS-s the operating system maintains this information independently for each drive. In the Windows NT family the OS only stores this information for the [current drive](#)^[471], but both **4NT** and **TC** do it for all drives.

Current Drive - The disk drive on which all file operations will take place unless otherwise specified. The current drive is typically displayed as part of the command prompt.

Current Working Directory - Alternate name for [Current Directory](#)^[471].

Cursor - A movable marker on the screen to show where text will be entered when you type at the keyboard, or which object on the screen will be affected when a mouse button is clicked. In character mode only the text cursor is available; graphical systems typically show both a mouse cursor and, when text can be entered, a separate text cursor.

Cyclic Redundancy Code (CRC) - A sequence of characters, associated with a block of data, which provide a reasonably unique signature, used to detect the possibility of the data being corrupted in storage or transmission.

8.6.5 Glossary - D

Date Range - A command processor feature which allows you to select files based on the date and time they were last modified. For details, see [Date Ranges](#)^[42].

Date Stamp - Information which may be stored in a file's directory entry to show the dates on which the file was created, last modified, and last accessed for reading. Only the modification date is available in the [FAT file system](#)^[471]. See also **Time Stamp**.

Default Directory - Same as [Current Directory](#)^[471].

Default Drive - Same as [Current Drive](#)^[471].

Deletion Tracking - An operating system or utility software feature designed to allow you to "undelete" or recover files which have recently been deleted. Delete tracking typically works by temporarily retaining the deleted files and / or information about the deleted files in a special area of the disk.

Description - A special feature of JPsoft command processors. A string of characters may be assigned to describe a file, using the [DESCRIBE](#)^[176] command, typically stored in the file `DESCRIPT.ION` in the same directory as the file itself.

Destination - In some file processing commands (e.g. [COPY](#)^[164] or [MOVE](#)^[246]), the name or directory the files should have after the command is completed. It is generally the last specification on the command line. See also [Source](#)^[483].

Detached Process - A program which is *detached* from the normal means of user input and output, and cannot use the keyboard, mouse, or video display.

Device Driver - A program which allows the operating system to communicate with a device. It must be loaded into memory before the system can access the attached devices. Device drivers are also used to manage memory or for other similar internal functions.

Device - A physical device for input or output such as the console, a communications port, or a printer. Sometimes *device* is used to refer to [character devices](#)^[471], and excludes [block devices](#)^[470].

Directive - The name of a configurable command processor option and the value assigned to it. All directives may be stored in an [Initialization File](#)^[479]. Most directives may also be effectuated using the [OPTION](#)^[253] command. See also: [initialization file](#)^[479].

Directory -

1) A portion of a disk, identified by a name and a relationship to other directories in a [directory tree](#)^[474] structure, with the tree starting at the [root directory](#)^[483]. A **directory** separates files on the disk into logical groups, but does not represent a physical division of the data on the disk.

2) The [attribute](#)^[469] of the entry for the **directory** in its [parent directory](#)^[482].

Directory History - A command processor feature which allows you to recall recently-used directory names in a popup window, and choose one to switch to.

Directory History Window - The name of the pop-up window displaying [Directory History](#)^[474].

Directory Stack - A command processor feature, implemented through the [PUSHD](#)^[264] and [POPD](#)^[261] commands, which allow you to save the current directory and return to it later. See also [Stack](#)^[483].

Directory Tree - The branching structure of directories on a disk, starting at the [root directory](#)^[483]. The *root* of the tree is usually considered as the *top* of the structure, so the actual structure can be visualized as an upside-down tree with the *root* at the top and branches going down. A portion or branch of the directory tree is sometimes called a *subtree*.

Double quote - A misuse of English, originating in the days when keypunches did not have quote marks, and apostrophes were used to represent them. Apostrophes had been incorrectly referred to as quote marks, and a pair of two apostrophes, often used inside an apostrophe-delimited ("quoted") string to represent a single apostrophe, as a "double quote".

Drive Letter - A letter used by some operating systems to designate a specific local disk volume, or part or all of a network server volume. In most cases drive letters range from A - Z, but some network operating systems allow the use of certain other characters as drive letters to support more than 26 disk volumes.

8.6.6 Glossary - E

Echo - See [Command Echoing](#)^[477].

End of Line (EOL) - An indication of the end of line in [text file](#)^[484]s. Many conventions exist to represent EOL in ASCII files. The most common ones use one or two control characters: (1) [CR](#)^[477], (2) CR followed by [LF](#)^[480], (3) LF followed by CR, (4) LF. PC-DOS/MS-DOS/Windows text files normally use (2). Unix/Linux and their variants use (4), and refer to the LF character as the new line (NL) character.

Environment - A set of variables and their values. Some variables are set before the start of the command processor. Others may be set at the command line using the [SET](#)^[287] command, or by your batch files. If a variable is defined, it has a value, which is a character string, and is at least 1 character long. A variable can be removed from the environment using the [UNSET](#)^[312] command, or by setting its value to an empty string. See also [Master Environment](#)^[487] and [Passed Environment](#)^[482].

Environment Variable - The name of a single entry in the [environment](#)^[476].

Error Level - A numeric value returned from an external command or, in some cases, from an internal command to indicate its result (e.g., success, failure, response to a question). Not all commands return an **error level**. Also known as [Exit Code](#)^[476].

Escape Character -

1) The command processor escape character, used to suppress the normal meaning of, or to give special meaning to the immediately following character. The default **escape character** in **4NT** and **TC** is the caret (^). This may be modified using the [EscapeChar](#)^[103] [directive](#)^[474] or by using the **/E** option of the [SETDOS](#)^[283] command.

2) A control character, symbol **ESC** ([ASCII](#)^[446]: 27).

Escape Sequence - A sequence of text characters which has a special meaning and is not treated as normal text. For example, the character sequence **<ESC> JK** (where **<ESC>** represents the ASCII "escape" character, decimal value 27) will cause an ANSI X3.64 driver to clear the screen from the cursor to the end of the current line, rather than simply displaying the string **ESC JK** on the screen. Similarly, in the command processor, the escape sequence **^F** on the command line is translated to a form feed, and is not treated as the literal characters **^F**.

Executable Extension - A command processor feature which allows you to specify the application to be executed when a file with a particular extension is named at the command prompt.

Executable File - A file, usually with the extension **.COM** or **.EXE**, which can be loaded into memory and run as a program.

Exclusive OR - See [XOR](#)^[486].

Exit Code - The result code returned by an external command or an internal command. Command processor internal commands return an **exit code** of 0 if successful, or non-zero if unsuccessful. External commands may, but need not return an **exit code**. See also [Errorlevel](#)^[476].

Expansion -

- 1) [Command Line Expansion](#)^[477]
- 2) The inverse of [compression](#)^[477].

Exclusive OR (XOR) - A logical combination of two *true* or *false* conditions. If both conditions are *false* or both conditions are *true* the result is **false**; if either condition is *true* and the other is *false* the result

is **true**. See below.

<i>condition 1</i>	<i>condition 2</i>	<i>result</i>
<i>false</i>	<i>false</i>	<i>false</i>
<i>false</i>	<i>true</i>	<i>true</i>
<i>true</i>	<i>false</i>	<i>true</i>
<i>true</i>	<i>true</i>	<i>false</i>

Extended ASCII Character: A legally invalid phrase, used for a character whose code is in the range [128,255], used on the PC as part of an extended set of 256 characters. The extension character set may include international language symbols, and box and line drawing characters. What is displayed or what is printed when such a character is used depends on the [country setting](#)^[474], [code page](#)^[474], and font.

Extended Directory Search - A command processor feature which maintains a directory search database or list, typically including all directories in your system, and allows you to change quickly to any directory in the list based on partial match of the directory you specify in the command.

Extended Key Code - The code for a key on the PC keyboard which has no representation in the standard ASCII character set, such as a function key, cursor key, or **Alt** plus another key. The extended key code for a key is often the same as the scan code for that key. See [Key Codes and Scan Codes Table](#)^[452].

Extended Parent Directory Names - A command processor feature which allows you to use additional periods in a directory name to represent directories which are successively higher in the directory tree.

Extended Wildcard - A command processor feature which extends the wildcard syntax and allows you to use multiple wildcard characters, and character ranges (e.g. [a-f] for the letters A through F). See also [Wildcard](#)^[486].

Extension - The portion of a file name following the last period. For example, in the file name C:\DIR1\LETTER.JOHN.DAT the extension is .DAT.

External Command - A program directly executable by the operating system, usually performing a utility function, as distinguished from an [internal command](#)^[479], which is executed by the command processor.

EXTPROC - A command processor feature which allows you to designate a specific external program to run a particular batch file instead of the command processor.

8.6.7 Glossary - F

FAT - [File Allocation Table](#)^[477].

FAT-Compatible File Name - See [SFN](#)^[483].

FAT File System - The file system used by PC-DOS, MS-DOS, and its emulators based on [FAT](#)^[477].to store files on diskettes and hard disks; also supported by Windows. Supports a single 2-s resolution timestamp of "last written" (modified). It is subclassified according to the size of word required to represent the largest possible [FAT](#)^[477] size. Three versions have been implemented: [12-bit](#)^[477], [16-bit](#)^[477], and [32-bit](#)^[477].

FAT12 File System - Each entry in the [FAT](#)^[477] is 12 bits long, limiting table size to 1,024 allocation

blocks.

FAT16 File System - Each entry in the [FAT](#)^[477] is 16 bits long, limiting table size to 65,536 allocation blocks.

FAT32 File System - Each entry in the [FAT](#)^[477] is 32 bits long, limiting table size to 4,294,967,296 allocation blocks, See also [VFAT File System](#)^[485].

FF - [Form Feed](#)^[477].

File Allocation Table (FAT) - A table used by the file system to keep track of disk space usage, allocation, ownership, and file content.

File Attribute - See [Attribute](#)^[469].

File Description - See [Description](#)^[474].

File Exclusion Range - A command processor feature which allows you to exclude files from processing by internal commands based on their names. See [File Exclusion Ranges](#)^[45].

Filename Completion - A command processor feature which allows you to type part of a filename on the command line, and have the command processor fill in the rest for you.

Font - a set of character shapes in a single style and size.

Form Feed (FF) - A control character (ASCII: 12), which typically causes a printer to skip to a new page. The **FF** character is not normally entered from the keyboard, but in many cases it can be generated, if necessary, by holding the **Alt** key, pressing the numeric pad keys 012, and releasing the **Alt** key, or by the **Ctrl-L** key combination.

Free Memory - Usually, the amount of total memory which is unoccupied and available for applications.

8.6.8 Glossary - G

Global Aliases - A command processor option which allows you to store aliases in a global area accessible to all copies of the command processor, so that any change made by one copy, even between a [SETLOCAL](#)^[287] - [ENDLOCAL](#)^[200] command pair, is immediately effective in all other copies. Cf. [Local Aliases](#)^[480].

Global Directory History - A command processor option which allows you to store the directory history in a global area accessible to all copies of the command processor, so that any change made by one copy, even between a [SETLOCAL](#)^[287] - [ENDLOCAL](#)^[200] command pair, is immediately effective in all other copies. Cf. [Local Directory History](#)^[480].

Global History - A command processor option which allows you to store the command history in a global area accessible to all copies of the command processor, so that any change made by one copy, even between a [SETLOCAL](#)^[287] - [ENDLOCAL](#)^[200] command pair, is immediately effective in all other copies. Cf. [Local History](#)^[480].

Global User Functions - A command processor option which allows you to store user-defined functions in a global area accessible to all copies of the command processor, so that any change made by one copy, even between a [SETLOCAL](#)^[287] - [ENDLOCAL](#)^[200] command pair, is immediately effective in all other copies. Cf. [Local User Functions](#)^[480].

Graphics Mode - A display mode in which output is displayed in any one of a range of fonts, typically in resizable windows with a variable number of text rows and columns, and which supports the display of graphics and pictures along with text. Cf. [Character Mode](#)^[477].

8.6.9 Glossary - H

Hidden - A file attribute indicating that the file should not be displayed with a normal [DIR](#)^[179] command, and should not be made available to programs unless they specifically request access to hidden files. Often combined with the [System](#)^[483] attribute.

History - See [Command History](#)^[477].

History Window - See [Command History Window](#)^[477] and [Directory History](#)^[474].

8.6.10 Glossary - I

IFS - [Installable File System](#)^[479].

Installable File System (IFS) - A file system for which device drivers can be loaded when required to support devices such as CD-ROM or network drives, or non-default disk formats. Installable file systems may be loaded at system startup, or loaded and unloaded dynamically while the system is running. They typically appear to the command processor simply as yet another drive .

Include List - A concise method of specifying several files or groups of files in the same directory, for use with all command processor commands which take file names as arguments.

Inheritance - A command processor feature which allows one copy of the command processor to "inherit" the .INI file data, aliases, user defined functions, environment variables, command history, and directory history from a previous copy. More generally, a system which allows one program to pass information or settings on to another, often to a second copy of the same program.

Initial Program Load (IPL) - the process of starting the computer and loading the operating system into memory. Also known as "boot", "booting", "bootstrapping".

Initialization Directive - See [Directive](#)^[474].

Initialization File or **.INI File** - A file which enumerates the initial settings of various command processor options, and which is automatically read by the command processor when it starts. The default initialization files are **4NT.INI** for **4NT** and **TCMD32.INI** for **TC**. For additional details, see [Startup Command](#)^[4] and [Initialization Files](#)^[89].

Insert Mode - When editing text, a mode in which newly typed characters are inserted into the line at the cursor position, rather than overwriting existing characters on the line. Cf: [Overstrike Mode](#)^[481].

Internal Command - A command which is part of the command processor. Cf [external command](#)^[476]..

Internal Variables - Special variables created by the command processor to provide information about your system. Internal variables are evaluated each time they are used, and are not actually stored in the environment. Internal variables are accessed using the same syntax as environment variables.

International Standardisation Organisation (ISO) - An international body promulgating standards.

ISO - [International Standardisation Organisation](#)^[479].

8.6.11 Glossary - K

Key Code - The code passed to a program when a key is pressed on the keyboard. Depending on the key that is pressed, and the software handling the keyboard, the code can be an ASCII, a scan code, or an extended key code.

Key Mapping - A command processor feature which allows you to assign new keystrokes for command line functions such as manipulating the command history or completing file names.

Keyboard Buffer - An operating system buffer which holds keystrokes you have typed that have not yet been used by the currently executing program.

Keystroke Alias - An alias assigned to a key, so that it can be invoked or recalled with a single keystroke.

8.6.12 Glossary - L

Label -

1) A location marker in a [batch file](#)^[470], with the format **:name**, allowing **GOTO** and **GOSUB** commands to "jump" to that point in the file.

2) [Volume Label](#)^[485].

Line Feed (LF) - A control character ([ASCII](#)^[446]: 10) indicating that the next character should be displayed in the current horizontal position but one line down. It is often used in text files as part of, or as the [End of Line](#)^[476]. It is not normally entered from the keyboard, but in many cases it can be generated, if necessary, by pressing **Ctrl-J**.

LF - [Line Feed](#)^[480].

LFN - [Long File Name](#)^[480].

LFN File System - A file system which supports [long file names](#)^[480]. An LFN file system may store both a long and short name for a file. The short name is sometimes called the *alternate* name. See also [Long File Name](#)^[480], [Short File Name](#)^[483], [VFAT File System](#)^[485], and [NTFS](#)^[481].

Local Aliases - A command processor option which allows you to store aliases in a local area only accessible to the current copy of the command processor, so that a change made in the current copy of the command processor does not affect other copies, and vice versa. Cf. [Global Aliases](#)^[478].

Local Directory History - A command processor option which allows you to store the directory history in a local area only accessible to the current copy of the command processor, so that a change made in the current copy of the command processor is not available in other copies, and vice versa. Cf. [Global Directory History](#)^[478].

Local History - A command processor option which allows you to store the command history in a local area only accessible to the current copy of the command processor, so that a change made in the current copy of the command processor is not available in other copies, and vice versa. Cf. [Global History](#)^[478].

Local User Functions - A command processor option which allows you to store user-defined functions in a local area only accessible to the current copy of the command processor, so that a change made in the current copy of the command processor does not affect other copies, and vice versa. Cf. [Global User Functions](#)^[478].

Logging - A command processor feature, implemented via the [LOG](#)^[242] command, which allows you to save a record of the commands you execute.

Long File Name (LFN) - A file name which does not conform to FAT file system restrictions, either because it is longer than the 8 character name plus 3 character extension, or because it contains spaces, multiple periods, or other characters not allowed in a FAT file name. Cf. [Short File Name](#)^[483].

8.6.13 Glossary - M

Master Environment - The master copy of the environment maintained by the OS. This is done completely differently by the Win98 and earlier OS-s and by the WinNT family of OS-s.

Modulo - The remainder after an integer division. For example 11 modulo 3 is 2, the remainder when 11 is divided by 3.

Multiple Commands - A command processor feature which allows multiple commands to be placed on a line, with each pair of consecutive commands separated by a [command separator](#)^[471]..

8.6.14 Glossary - N

Name - A part of the file name from its beginning up to, but not including, the last period in the file name.

Network - A system which allows several computers to be connected together to share files, printers, modems, or other resources, and to pass electronic mail or other information between the systems on the network.

Network File System - Software which runs over a network to allow access to files on the server. A network file system may support the same options as the file system used on local drives, or it may be more or less restrictive than the local file system about file names, disk volume capacity, and other similar features.

Numeric coprocessor - A device which performs floating point arithmetic, and is often external to the [CPU](#)^[471].

New Technology File System (NTFS): A file system distributed with Windows NT which allows longer file names, supports larger drives, and provides better performance than the [FAT file system](#)^[471].

8.6.15 Glossary - O

Operating System - A collection of software which provides access to various devices, including file systems and networks, services to other software, and ensures that programs don't interfere with each other while they are running.

Option -

1) A parameter for an internal command or application which specifies a particular behavior or setting. For example, the command "DIR /P" might be referred to as "having the /P option set". Alternate name: **switch**.

2) A command which allows modifying command processor operating characteristics.

Option file - Alternate name for [Initialization File](#)^[479].

OR - A logical combination of two **true** or **false** conditions. If both conditions are *false* the result is **false**; if either condition is *true* the result is **true**. See table below.

<i>condition 1</i>	<i>condition 2</i>	<i>result</i>
<i>false</i>	<i>false</i>	<i>false</i>
<i>false</i>	<i>true</i>	<i>true</i>
<i>true</i>	<i>false</i>	<i>true</i>
<i>true</i>	<i>true</i>	<i>true</i>

Overstrike Mode - When editing text, a mode in which newly typed characters overwrite existing characters on the line, rather than being inserted into the line at the cursor position. Cf.

[Insert Mode](#)^[479].

8.6.16 Glossary - P

Parameter - Additional information placed after a command or function name. For example, in the command `DIR XYZ`, `XYZ` is a parameter. Also used to refer to an alias parameter or batch file parameter. Mathematical name: **argument**.

Parent Directory - The directory in which a particular subdirectory is cataloged, often seen as the directory *above* a subdirectory.

Parsing - The process the command processor performs to analyze the command line, perform alias, function and environment variable expansion, and find the appropriate internal command or external command to execute. More generally, the process of breaking down a string or message into its individual components in order to process them properly.

Passed Environment - A copy of the environment created before running an application, so that any changes made by the application will not affect the environment of the application's parent.

Path -

- 1) A specification of all the directories required to locate a file. For example, the path for `C:\WPFILES\MYDIR\MEMO.TXT` is `C:\WPFILES\MYDIR\`.
- 2) The environment variable [PATH](#)^[341], which contains a series of path specifications used when searching for external commands and batch files.
- 3) The internal command [PATH](#)^[254], which redefines the value of the environment variable [PATH](#)^[341].

Pipe - A method for collecting the standard output of one command and passing it on as the standard input of another command, syntactically represented by a vertical bar "|" separating the commands. See also [Redirection](#)^[61].

Previous Working Directory - The working directory used most recently, just prior to selecting the [current working directory](#)^[471]. For example, if `C:\DATA` is the current working directory and you switch to `D:\UTIL`, `C:\DATA` becomes the previous working directory.

Primary Shell - The copy of the character-mode command processor which is loaded by the operating system when the system starts or a session opens.

8.6.17 Glossary - Q

Quote mark - The character " used by English and other natural languages to mark the beginning and end of a block of text copied from another source, and thus not subject to modification. Many programming languages, including **4NT** and **TC**, use it in a similar manner to delimit text to be used literally (i.e., exactly as is).

8.6.18 Glossary - R

RAM - [Random Access Memory](#)^[483].

RAM Disk - Technically incorrect name for [Virtual Disk](#)^[485].

Random Access Memory (RAM) - The physical memory used to store data while a computer is operating. The information in most types of RAM is lost when power is turned off. **Technically**, any memory device or system in which access time is independent of the order in which memory locations are accessed. Most **ROM**^[483]s used in PCs provide access times independently of access order, and are technically also RAMs. Conversely, many types of memory devices used in PCs, although customarily referred to as RAM, e.g., DRAM, are not truly RAMs. For example, DRAM devices provide faster response when consecutive accesses are made to locations within specific groups than to locations in different groups (although most PCs don't take advantage of this). There are other storage devices which provide read-write access, e.g., shift registers - for example, most of the recent modems contain a "first in, first out" buffer for incoming data, which are neither RAMs nor ROMs.

Range - See [Date Range](#)^[42], [Size Range](#)^[42], and [Time Range](#)^[44].

Read Only - A file attribute indicating that the file can be read, but not written or deleted by the operating system or the command processor unless special commands are used.

Read Only Memory (ROM) - A physical memory device used to store information which cannot be readily modified. . There are variants of **ROM** whose contents can be changed, but only through the use of special techniques. Many modern **BIOS ROMs** can be changed through the use of special programs. The opposite of **ROM** is read-write memory. Most ROM devices used in PCs are technically also [RAMs](#)^[483].

Reboot - The process of restarting the computer with software, with the keyboard (e.g. by pressing Ctrl-Alt-Del), by pressing a reset button, or by turning the power off and back on. See also [Boot](#)^[470], [Cold Reboot](#)^[471], [Initial Program Load](#)^[479] and [Warm Reboot](#)^[486].

Redirection - A method for collecting output from a command in a file, or of providing the input for a command from a file. See also [Pipe](#)^[482].

Registry - A hierarchically organized data file (or set of files) maintained by Windows to hold system parameters, hardware and software settings, and other similar information used by the operating system or by other software packages.

REXX - A file and text processing language developed by IBM, and available on many PC and other platforms.

ROM - [Read Only Memory](#)^[483].

Root Directory - The first directory on a disk, from which all other directories are "descended". The root directory is referenced with a single backslash [].

8.6.19 Glossary - S

Scan Code - The physical code for a key on the PC keyboard. For the original U.S. English keyboard layout the scan code represents the physical position of the key, starting with 1 for the key in the upper left corner (Esc), and increasing from left to right and top to bottom. This order will vary for more recent keyboards or those designed for other countries or languages. See [Key Codes and Scan Codes Table](#)^[452].

Search Path - A semicolon-separated list of directories in which to search for a file.

Secondary Shell - A copy of the command processor which is started by another program, rather than by the operating system.

Section - A part of an [Initialization File](#)^[479] starting with a **section name** enclosed in brackets, e.g., [primary].

Session - A general term for the individual windows or tasks started by a multitasking system.

Shell - The name used in Unix/Linux/etc. systems for [command processor](#)^[471], from a pictorial representation of the OS being encased by a shell.. Also used to refer to any program which gives access to operating system functions and features through a menu- or mouse-driven system, or which replaces the primary user interface of the operating system.

Short File Name (SFN) - A file name which follows the rules of the [FAT file system](#)^[477]: a [name](#)^[481] of 0 ..8 characters and an [extension](#)^[476] of 0 .. 3 characters, each consisting of only alphabetic and numeric characters plus the punctuation marks ! # \$ % & ' () - @ ^ _ ` { } and ~. When either **name** or **extension** is shorter than its maximum, internally to the file system it is padded with **invisible space** characters. See also [LFN](#)^[480].

Size Range - A command processor feature which allows you to select files based on their size. For details see [Size Ranges](#)^[42].

Source - In file processing commands (e.g. COPY or MOVE), the original files before any copying or modification has taken place, i.e., those specified earlier on the command line. See also [Destination](#)^[474].

Stack - An area of memory used by any program to store temporary data while the program is running; more generally, any such storage area where the last item stored is normally the first one removed.

Standard Error, Standard Input, and Standard Output - The file(s) or character device(s) where a program displays error messages, obtains its normal input, and displays its normal output, respectively. Unless [redirected](#)^[483], all three refer to the **console**.

Subdirectory - Any directory other than the [root directory](#)^[483].

Subtree - See [Directory Tree](#)^[474].

Switch - Alternate name for [Option](#)^[481].

System - A file attribute indicating that the file belongs to the operating system or command processor, and should not be accessed by other programs. Almost every file with this attribute also has the [Hidden](#)^[479] attribute.

8.6.20 Glossary - T

Target: See [Destination](#)^[474].

TCEXIT: A program which is executed whenever Take Command exits. See [Automatic Startup and Termination Programs](#)^[6] for more details.

TCSTART: A program which is executed whenever Take Command starts. See [Automatic Startup and Termination Programs](#)^[6] for more details.

Text file: A file containing only characters that are displayable, or which affect the display format (format effectors). The characters of the file are represented by their **character codes**. **4NT** and **TC** provide support for the two systems of character codes, [ASCII](#)^[446] and [Unicode](#)^[422].

Time Range: A command processor feature which allows you to select files based on the time they were last modified, created, or accessed. See [Time ranges](#)^[44] for more details.

Time Stamp: Information stored in a file's directory entry to show the times at which the file was created, last modified, and last accessed. Creation time is not available in the FAT file system; last access time is only available in the NTFS file system. See also [Date Stamp](#)^[474].

Tree: See [Directory Tree](#)^[474].

8.6.21 Glossary - U

UNC - Universal Naming Convention

Universal Naming Convention - A common method for accessing files on a network drive without using a "mapped" drive letter. Names specified this way are called UNC names, and 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.

UNICODE - A character set standard, designed and maintained by the non-profit *Unicode Consortium* (<http://unicode.org>). While DOS-based Microsoft operating systems (MS-DOS, Win9x/ME) relied on 8-bit *ASCII* (256 potential distinct values), their NT family (W2000, XP, etc.) introduced an implementation of 16-bit *UNICODE* (65536 potential distinct values) intended to facilitate use of non-English languages.

UDF - User Defined Function.

User Defined Function - A function defined by the user utilizing the [FUNCTION](#)^[218] command to manipulate strings, dates, and filenames; perform arithmetic; read and write files; and perform other similar functions. UDFs are invoked identically to [Variable functions](#)^[357].

8.6.22 Glossary - V

Variable - See [Alias Parameter](#)^[469], Batch File Parameter, Function Parameter, and Environment Variable.

Variable Expansion - The process of scanning a command line and replacing each environment variable name, alias-, function-, and batch file-parameter and function invocation with its value.

Variable Functions - Functions provided by the command processor to manipulate strings, dates, and filenames; perform arithmetic; read and write files; and perform other similar functions. Variable functions are similar to static environment variables or internal variables, but have parameters and can perform actions rather than just returning static information.

VFAT File System - An extension of the [FAT file system](#)^[477] which supports [long filenames](#)^[480].

Virtual Disk - A portion of memory, dedicated for this use through special purpose software, accessible by programs as if it were a physical disk drive, and storing data in files in a directory tree. IBM's VDISK.SYS and its Microsoft version, RAMDRIVE.SYS are typical implementations. Sometimes also called a **RAM Disk**.

Volume - See **Disk Drive**.

Volume Label - A character string stored on a disk drive or volume in a special manner used to identify it. Some operating systems, including all those from Microsoft, use a special, hidden file name cataloged in the disk directory for this purpose.

8.6.23 Glossary - W

Warm Reboot - The process of restarting the computer with software, or with the keyboard (e.g. by pressing Ctrl-Alt-Del), typically without physically resetting any hardware devices. See also **Cold Reboot**.

White Space Character - A character used to separate arguments on the command line. The white space characters recognized by the command processor are the space, horizontal tab, and comma.

Wildcard - A character ("*" or "?") used in a filename to specify the possibility that any single character ("?") or sequence of characters ("*") can occur at that point in the actual name. See also **Extended Wildcard**.

Windows NT File System - See [NTFS](#)^[481].

8.6.24 Glossary - X

X-3.64 - See [ANSI X-3.64](#)^[469].

XOR - [Exclusive OR](#)^[476].

9 What's New

WHATSNEW.TXT -- November 2004

JP Software Inc.
P.O. Box 328
Chestertown, MD 21620, USA

(410) 810-8818
fax (410) 810-0026

Email sales@jpsoft.com
support@jpsoft.com
Web <http://jpsoft.com>

NEW VERSION OVERVIEW - Take Command and 4NT 6.01

Since the last major release we've added numerous new features, commands, switches, variables, and other enhancements to 4NT and Take Command!

This is a summary of the compatibility fixes and new features. For complete details, see the appropriate topics in the online help.

New and modified features:

4NT and Take Command have been rewritten to share a common .DLL (TakeCmd.dll). This means they will use much less memory when you have multiple shells open.

The Caveman support in Take Command has improved compatibility and speed.

4NT and Take Command use a new IPWORKS version (IPWORKS6.DLL and IPWSSL6.DLL).

4NT - no longer needs a separate KEYSTACK.EXE (it's built into TakeCmd.dll).

4NT and Take Command now recognize Windows 2003.

Added FTPS (SSL FTP) support.

Added HTTPS (SSL HTTP) support.

Added TFTP (trivial FTP) support.

4NT and TC will now look for your FTP user names and passwords in the file "FTP.CFG", which is kept in the 4NT or TC directory by default (you can specify another directory with the FTPCFG .INI directive).

Added a new type of redirection (like the *nix "here-document") using the "program << word" syntax.

You can now preface an .INI filename on the startup line with a / to work around a Windows problem with not displaying the entire command line in a shortcut when an argument begins with @.

Increased key substitution table from 64 to 128 entries.

10 Order Form

You can at any time apply your current personal [registration information](#)^[418] to a "*trial*" version in order to turn it into a "*registered*" product that bears your name and never expires.

To **order** or **upgrade** any of our products, visit our web site and **Secure Online Store** or email /fax us a copy of the form shown below. See "[Contacting JP Software](#)"^[421] for our various addresses and phone/fax numbers.

JP Software
P.O. Box 328
Chestertown, MD 21620 USA

Standard Order Form [2004/8]
email to sales@jpsoft.com
or fax to (410) 810-0026

This order form covers our standard product line: new copies, upgrades, our CD Suite, and manuals.

For multi-system licenses, more shipping information, and answers to other ordering questions, see the PRODUCTS.TXT file. If you need additional information, or want to order directly from our web site:

For	See
-----	-----
On-line order entry	http://jpsoft.com/
Email orders / inquiries	sales@jpsoft.com
Phone orders / inquiries	(410) 810-8818
Fax orders	(410) 810-0026
US / international dealer list	http://jpsoft.com/
Detailed product information	PRODUCTS.TXT file
Support plans	SUPPORT.TXT file

=====

Name (first, last) _____

Company _____

Address _____

City, State/Prov, Post Code _____

Country _____

Phone _____ Fax _____

E-Mail _____

(Will only be used by JP Software to notify you of upgrades or contact you with other information related to our products.)

Register products to: ☐ Company ☐ Individual

Where did you hear about our products? _____

=====

	Total Price
New Copies and Product Packs	
<input type="checkbox"/> 4NT (\$69.95)	_____ [NC120]
<input type="checkbox"/> Take Command (\$69.95)	_____ [NC140]
<input type="checkbox"/> CD Suite (both products, \$99.95)	_____ [CD180]
Extended Support Annual Subscription (see SUPPORT.TXT; single system \$19.95 for 1 product, \$29.95 for CD Suite)	
<input type="checkbox"/> 4NT Extended Support	_____ [SU620]
<input type="checkbox"/> Take Command Extended Support	_____ [SU640]
<input type="checkbox"/> CD Suite (both products)	_____ [SU660]

11 Copyright & Version

4NT 6.01 for Microsoft Windows 98 / ME / NT / 2000 / XP / 2003
Take Command 6.01 for Microsoft Windows 98 / ME / NT / 2000 / XP / 2003
Software: Copyright © 1988-2004, Rex Conn and JP Software Inc.
All Rights Reserved.

Version 6.01 Help System
Help text: Copyright © 1993-2004 JP Software Inc.
All Rights Reserved.

Language.dll translations by
French Christian Albaret
German Hans-Peter Grözinger

We gratefully acknowledge the contributions of Roger Byrne and many of our other users.

This help material was last revised on Thursday, December 30, 2004.

4NT is JP Software Inc.'s trademark for its family of character-mode command processors. Take Command® is a registered trademark of JP Software Inc. JP Software, jpsoft.com, and all JP Software designs and logos are also trademarks of JP Software Inc. Other product and company names are trademarks of their respective owners.

Index

- ! -

! exclamation 4, 45, 61, 107
!= not equal 31

- \$ -

\$ dollar 107, 262, 322, 335

- & -

& ampersand 100, 107, 335
&& double ampersand 26

- (-

() parentheses 27, 31, 374

- * -

* asterisk 25, 36, 322

- . -

.AND. 31
.BAT 321
.BTM 321
.CMD 321
.INI File 6
.OR. 31, 295
.XOR. 31

- / -

/ slash 42, 45, 52, 99, 112, 170, 172, 179, 206, 210, 224, 229, 237, 267, 270, 275, 288, 296, 423
// double slash 253

- ? -

? 346

? (command) 136
? (variable) 346, 357
? question mark 36, 423

- @ -

@ at sign 13, 49, 52, 376
@ABS 364
@AGEDATE 364
@ALIAS 364
@ALTNAME 364
@ASCII 365
@ATTRIB 39, 365
@CAPS 366
@CDROM 366
@CEILING 366
@CHAR 366
@CLIP 367
@CLIPW 367
@COLOR 367
@COMMA 368
@CONSOLE 368
@CONVERT 368
@CRC32 368
@DATE 369
@DAY 369
@DEC 370
@DECIMAL 370
@DESCRIPT 370
@DEVICE 370
@DIGITS 370
@DIRSTACK 371
@DISKFREE 371
@DISKTOTAL 371
@DISKUSED 372
@DOW 372
@DOWF 372
@DOWI 373
@DOY 373
@ENUMSERVERS 373
@ENUMSHARES 373
@ERRTEXT 374
@EVAL 374
@EXEC 376
@EXECSTR 377
@EXETYPE 377
@EXPAND 378
@EXT 378

@FIELD	379	@ISXDIGIT	396
@FIELDS	379	@LABEL	396
@File Lists	49	@LEFT	397
@FILEAGE	380	@LEN	397
@FILECLOSE	380	@LFN	397
@FILEDATE	380	@LINE	398
@FILENAME	381	@LINES	398
@FILEOPEN	381	@LOWER	398
@FILEREAD	382	@LTRIM	399
@FILES	382	@MAKEAGE	399
@FILESEEK	383	@MAKEDATE	399
@FILESEEKLN	383	@MAKETIME	400
@FILESIZE	384	@MAX	400
@FILETIME	384	@MD5	400
@FILEWRITE	385	@MIN	400
@FILEWRITEB	385	@MONTH	400
@FINDCLOSE	385	@NAME	401
@FINDFIRST	386	@NUMERIC	401
@FINDNEXT	386	@OPTION	401
@FLOOR	387	@PATH	402
@FORMAT	387	@PING	402
@FORMATN	388	@RANDOM	402
@FSTYPE	388	@READSCR	403
@FULL	388	@READY	403
@FUNCTION	388	@REGCREATE	403
@GETDIR	389	@REGDELKEY	403
@GETFILE	389	@REGEXIST	404
@GETFOLDER	389	@REGQUERY	404
@HISTORY	390	@REGSET	404
@IDOW	390	@REGSETENV	404
@IDOWF	390	@REMOTE	404
@IF	31, 390	@REMOVABLE	405
@INC	390	@REPEAT	405
@INDEX	391	@REPLACE	405
@INIREAD	391	@REXX	405
@INIWRITE	392	@RIGHT	406
@INSERT	392	@RTRIM	406
@INSTR	393, 407	@SEARCH	406
@INT	393	@SELECT	407
@IPADDRESS	394	@SFN	407
@IPNAME	394	@STRIP	407
@ISALNUM	394	@SUBST	408
@ISALPHA	394	@SUBSTR	393, 407
@ISASCII	395	@TIME	408
@ISCNTRL	395	@TIMER	408
@ISDIGIT	395	@TRIM	408
@ISPUNCT	396	@TRUENAME	408
@ISSPACE	396	@UNC	409

@UNICODE 409
 @UNIQUE 409
 @UPPER 409
 @VERINFO 409
 @WATTRIB 39, 410
 @WILD 410
 @WINCLASS 411
 @WINEXENAME 411
 @wininfo 350, 411
 @WINMEMORY 411
 @WINMETRICS 412
 @WINSTATE 414
 @WINSYSTEM 414
 @WORD 416
 @WORDS 417
 @YEAR 417

- [-

[] brackets 176, 336, 357

- \ -

\ backslash 17, 19, 20, 22, 56, 59, 89, 99, 112,
 164, 243, 246, 270, 402, 403, 440, 458

- ^ -

^ caret 21, 100, 103, 335

- _ -

_? 346
 _4VER 347
 _ACSTATUS 347
 _ALT 348
 _ANSI 348
 _BATCH 348
 _BATCHLINE 348
 _BATCHNAME 348
 _BATCHTYPE 348
 _BATTERY 348
 _BATTERYLIFE 348
 _BATTERYPERCENT 348
 _BG 349
 _BOOT 349
 _BUILD 349

_CAPSLOCK 349
 _CHILDPID 349
 _CI 349
 _CMDLINE 349
 _CMDPROC 349
 _CMDSPEC 349
 _CO 349
 _CODEPAGE 350
 _COLUMN 350
 _COLUMNS 350
 _COUNTRY 350
 _CPU 350
 _CPUUSAGE 350
 _CTRL 350
 _CWD 350
 _CWDS 351
 _CWP 351
 _CWPS 351
 _DATE 351
 _DATETIME 351
 _DAY 351
 _DETACHPID 351
 _DISK 351
 _DNAME 102, 176, 351
 _DOS 351
 _DOSVER 352
 _DOW 352
 _DOWF 352
 _DOWI 352
 _DOY 352
 _ECHO 352
 _FG 352
 _FTPERROR 352
 _HLOGFILE 353
 _HOST 353
 _HOUR 353
 _HWPROFILE 353
 _IDOW 353
 _IDOWF 353
 _IMONTH 353
 _IMONTHF 353
 _ININAME 353
 _IP 354
 _ISODATE 354
 _KBHIT 354
 _LALT 354
 _LASTDISK 354
 _LCTRL 354

_LOGFILE 354
 _LSHIFT 354
 _MINUTE 354
 _MONTH 354
 _MONTHF 354
 _NUMLOCK 354
 _PID 354
 _PIPE 354
 _PPID 355
 _RALT 355
 _RCTRL 355
 _ROW 355
 _ROWS 355
 _RSHIFT 355
 _SCROLLLOCK 355
 _SECOND 355
 _SELECTED 355
 _SHELL 355
 _SHIFT 355
 _SHRALIAS 355
 _STARTPATH 356
 _STARTPID 356
 _SYSERR 356
 _TIME 356
 _TRANSIENT 356
 _UNICODE 356
 _WINDIR 356
 _WINFGWINDOW 356
 _WINNAME 356
 _WINSYSDIR 356
 _WINTICKS 356
 _WINTITLE 356
 _WINUSER 356
 _WINVER 357
 _XPIXELS 357
 _YEAR 357
 _YPIXELS 357

- | -

|| double "pipe" symbol 26

- ~ -

~ tilde 47, 52, 229, 322, 336, 441

- + -

+ plus sign 164, 335, 347

- = -

= equal sign 335, 347

== double equal 31

- 4 -

4DOS.HLP 423

4EXIT 6, 468

4HELP.EXE 423

4NT.GPF 419

4NT.KEY 418

4NTA.EXE 422

4NTLOG 106

4NTU.EXE 422

4START 6, 64, 468

4StartPath 93

- A -

ABOVENORMAL 291

AC line status 347

accent grave (backtick) 446

ACK 446

acknowledge 446

ACTIVATE 136

AddFile 119

Advanced Directives 125

Alias 119, 138, 201, 310

Aliases 25, 138, 201, 310, 318, 327

Aliases Window 74

AliasExpand 119

Alt key 348

Alt-255 11, 119

Alt-Down 11

Alt-End 11

Alt-Home 11

Alt-PgDn 11

Alt-PgUp 11

Alt-Up 11

ampersand 446

AmPm 99

AND 26
ANSI 60, 64, 99, 445, 456, 461
ANSI X3.64 status 348
apostrophe 446
App Paths 458
AppendToDir 19, 99
Apps Menu 69
Archive 39, 365, 442
Argument 322, 332
ASCII 422, 445, 446, 471, 480
ASSOC 145, 461
asterisk 446
at sign 446
ATTRIB 39, 146
Attributes 39, 146, 365, 378, 382, 410, 442
Automatic Directory Change 22
Automatic programs 6

- B -

Background 349
Background color 349
backslash 446
Backspace 11, 116, 446
Basics 3
Batch 348
Batch Debugger 74
Batch Debugger Tab 86
Batch file 348
Batch file name 348
Batch File Parameter 322
Batch Files 6, 149, 318, 321, 322, 324, 325, 327, 329, 331
BATCH.BCP 149
BatchEcho 100
BATCOMP 149, 331
Battery 348
Battery charge 348
Battery charge level 348
Battery life remaining 348
BDEBUGGER 149
BEEP 156
BeepFreq 100
BeepLength 100
BeginLine 116
BEL 446
bell 446
BELOWNORMAL 291

Bksp 11, 116
Bookmarks 149
Boot 349
Boot drive 349
Border 461
BOTTOM 136, 316
BREAK 157, 252, 327
Breakpoint 149
BS 446
Buffer 75, 121
Build 349
Buy 489

- C -

CALL 157
CAN 446
CANCEL 158, 446
Caps Lock 234, 349
CAPS LOCK key status 349
caret 446
carriage return 446
CASE 295
Caveman 78, 79
Caveman Options Tab 87
CD 159, 160
CDD 160
CDDWinColors 113
CDDWinHeight 100
CDDWinLeft 100
CDDWinTop 100
CDDWinWidth 100
CDPATH 59, 339
chaining 157
CHCP 162
CHDIR 159
Child Process ID 349
ClearKeyMap 126
Clipboard 76, 367
CLOSE 136
CLS 163
CM 291
CMDLINE 339
CMSTDIO 291
Code Page 162, 350
colon 446
COLOR 163
Color Codes 461

- Color Directives 113
- Color Names 461
- Color Options Tab 85
- ColorDIR 113, 339
- Colors 349, 352
 - Color Names and Codes 349, 352
- comma 446
- Command editing 11
- Command editing keys 11
- Command Grouping 27
- Command History 13, 15, 16
- Command Line 4, 10, 11, 25, 77, 349
- Command Line Editing Keys 118
- Command Names 17
- Command Parsing 29
- command processor 349
- Command processor exit 8
- Command processor exit codes 8
- Command processor options 4
- Command processor parameters 4
- Command processor path 349
- Command processor startup 3
- Command Processor Version 347
- Command separator 347
- CommandEscape 119
- Commands 127, 325
- Commands By Category 131
- Commands By Name 127
- CommandSep 24, 100, 335, 347
- COMPARISON 31
 - CASE INSENSITIVE 31
 - CASE SENSITIVE 31
 - NUMERIC 31
 - STRING 31
- Compatibility 335
- CompleteHidden 101
- Compound Character 24, 100
- Compressed 39, 365
- Compressed batch file 348
- Compression 331
- COMSPEC 340
- CONAGENT.EXE 421
- Conditional Commands 26
- CONDITIONAL EXPRESSION 31
- CONDITIONAL EXPRESSIONS 31
- CONFIG.NT 321
- Configuration 81, 253, 283
- Configuration Dialog 82
- Configuration Directives 97
- Console Applications 78
- Console Window 78
- ConsoleColumns 94
- ConsoleRows 94
- Contact 421
- Converting 20
- COPY 164, 340
- Copy (directive) 116
- COPYCMD 340
- Copying 76
- CopyPrompt 101
- Copyright 491
- CR 446
- Ctrl-C 11
- Ctrl key 350
- Ctrl-A 11, 20, 120
- Ctrl-Bksp 11, 117
- Ctrl-break 11, 101, 157, 327
- Ctrl-C 101, 157, 327
- Ctrl-D 13, 119, 123
- Ctrl-Del 101
- Ctrl-down 13, 120
- Ctrl-E 13, 119
- Ctrl-End 11, 116
- Ctrl-Enter 13, 120, 123
- Ctrl-F 11, 119
- Ctrl-F1 11, 120, 225, 423
- Ctrl-Home 11, 116
- Ctrl-Ins 101
- Ctrl-K 11, 13, 121
- ctrl-L 11, 116
- Ctrl-Left 11, 118
- Ctrl-PgDn 123
- Ctrl-PgUp 122, 123
- Ctrl-R 11, 117
- Ctrl-Right 11, 118
- Ctrl-Shift-Tab 119
- Ctrl-Tab 120
- Ctrl-up 13, 121
- Ctrl-V 101, 118
- Ctrl-X 21, 101, 103, 335
- Ctrl-Y 11, 116
- Ctrol-Shift right 11
- Ctrol-Shift-ins 11
- Ctrol-Shift-left 11
- CUA 101
- Current command line 349

Current working directory 350, 351
Cursor 95, 101, 102, 349
Cursor column 350
Cursor position 350
Cursor shape 349
CursorIns 101, 349
CursorOver 101, 349

- D -

data link escape 446
DATE 170
Date Formats 363
Date Ranges 40, 42, 378, 382
DATETIME 191
DC1 446
DC2 446
DC3 446
DC4 446
DDE Support 81, 171
DDEEXEC 81, 171
Debug 86, 126
Debugging 149, 329
DecimalChar 102
DEFAULT 295
Default Variables 201, 281, 312
DEFINED 31
DEL 172, 446
Del (directive) 116
DELAY 175
Delete 11, 446
DelGlobalQuery 102
DelHistory 119
DELIMS 210
DelToBeginning 116
DelToEnd 116
DelWordLeft 116
DelWordRight 117
DESCRIBE 102, 176, 351
DescriptionMax 102
DescriptionName 102, 176, 351
Descriptions 102
Desktop 289
Desktop Integration 8
DETACH 178, 351
Detecting 327
device control 1 446
device control 2 446

device control 3 446
device control 4 446
Dialog 71, 72, 73, 74
DIR 179, 256, 340
DIRCMD 340
Directories 440
Directory 19, 39, 365, 442
Directory History 16, 23
Directory Navigation 54, 190, 261, 264, 371
Directory Search 104
Directory Searches 56, 59, 342
Directory Stack 54, 190, 261, 264, 371
DIREXIST 31
DirHistory 94, 189
DIRS 190
DirWinOpen 122
Disable 283
DLE 446
DO 31, 191, 210
 UNTIL 31
 WHILE 31
dollar (currency) sign 446
Down 13, 117, 120
Drag and Drop 80
DRAWBOX 195
DRAWHLINE 196
DRAWVLINE 197
Drive 439
DuplicateBugs 94

- E -

ECHO 198, 199
ECHOERR 199, 200
Echoing 322
ECHOS 198, 199
ECHOSERR 199, 200
Edit Descriptions Dialog 74
Edit Menu 69
Editing 11
Editing keys 11
Editing Options Tab 84
EditMode 102
Editor 103
ELSE 227, 228
ELSEIF 228
EM 446
Enable 283

Encrypted 39, 365
 Encrypted batch file 348
 End 11, 117
 end of medium 446
 end of transmission 446
 end text 446
 end text block 446
 ENDDO 191
 EndHistory 119
 ENDIFF 228
 EndLine 117
 ENDLOCAL 200
 ENDSWITCH 295
 ENDTEXT 301
 ENQ 446
 enquiry 446
 Enter 11, 117, 123
 Environment 201, 281, 312, 336
 Environment Window 74
 EOL 210
 EOT 446
 EQ 31
 EQL 31
 EQU 31
 equal sign 446
 ERASE 172
 EraseLine 117
 ERROR 252
 Error Messages 424
 ERRORLEVEL 31, 346, 357
 ERRORMSG 252
 Errors 424, 465
 Esc 117, 446
 Escape 119, 347, 446
 Escape Character 21
 EscapeChar 21, 103, 331, 335, 347
 ESET 138, 201, 310, 341
 ETB 446
 ETX 446
 EvalMax 103
 EvalMin 103
 EVENTLOG 203
 EXCEPT 204
 exclamation mark 446
 Exclusive OR 486
 ExecLine 117
 Executable Extensions 48
 Executable Files 458

ExecWait 29, 103
 EXIT 205
 Exit Code 8, 26, 346
 Expand 119
 Explorer Shortcuts 8
 Extended Directory Searches 56, 159, 160, 264, 342
 Extended Parent Directory Names 47
 EXTPROC 334

- F -

F1 11, 119, 225, 423
 F10 119
 F12 121
 F3 13
 F6 122
 F7 120
 F8 121
 F9 120
 FAT 439
 FAT32 439
 FF 446
 FFIND 206
 field separator 446
 File Exclusion Ranges 45
 File Menu 68
 File Names 438, 441
 File Prompts 65
 File Searches 47, 458
 File Selection 36, 51
 File Streams 444
 File Systems 438, 439
 FILECOMPLETION 19, 21, 103, 340
 Filename Completion 17, 19, 21
 Filenames 20
 FILETIME 364, 380, 399
 FILL 195
 Find Files/Text Dialog 73
 FirewallHost 104
 FirewallPassword 104
 FirewallType 104
 FirewallUser 104
 Font 83
 FOR 210
 FOREVER 191
 form feed 446
 FREE 216

FS 291, 446
FTP 52, 104, 229, 352
FTP.CFG 52, 104
FTPCFG 104
FTPS 52
FTYPE 217, 461
FUNCTION 218, 311
Functions 336, 357
Functions Dialog 218
Functions Window 74
FuzzyCD 104

- G -

GE 31
General Input Keys 115
GEQ 31
giga 42
Global 16, 220
Glossary 468, 469, 470, 471, 474, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486
GOSUB 221, 272
GOTO 222, 228
GR 446
greater than sign 446
group separator 446
GT 31
GTR 31

- H -

HEAD 224
HELP 119, 225, 423
Help Menu 71
HelpWord 120, 225, 423
here-document 61
Hidden 39, 365, 442
HIDE 136, 316
HideConsole 94
HIGH 291
Highlighting 76
HistCopy 105
HistDups 85, 105
HistLogName 105, 242
HistLogOn 105, 242
HistMin 105
HistMove 105

History 25, 95, 105, 106, 226, 242, 341
History OptionsTab 85
HISTORYEXCLUDE 341
HistWinOpen 122
HistWrap 106
Home 11, 116
horizontal tab 446
HT 446
<http://jpsoft.com/> 71
HTTPS 52
hyphen (minus sign) 446

- I -

IBeamCursor 95
IF 31, 227, 390
IFF 31, 228
IFTP 52, 229, 352
Include 126
Include Lists 46
INIQuery 95
Initialization Directives 93
Initialization Files 89
INKEY 231
INPUT 233
InputColors 113
Ins 117
Insert 11, 102, 349
Insert cursor 349
Insert cursor shape 349
Internal command 346
Internal Commands 127
Internal Variables 342
Internet 51, 52, 229, 279, 291, 352
Internet Options Tab 87
INV 291
IP 394
ipworks6.dll 52
ipwssl6.dll 52
ISALIAS 31
ISAPP 31
ISDIR 31
ISFILE 31
ISFUNCTION 31
ISINTERNAL 31
ISO 9601 363
ISO date 351
ISWINDOW 31

ITERATE 191

- J -

JP Software 421

jphelp.chm 423

JPSTREE.IDX 56, 97, 159, 160, 264, 342

junction 245

Junction (reparse point) 39, 365

- K -

Key Codes 445, 451, 452

Key Mapping Directives 11, 13, 15, 17, 114

Key Names 464

KEYBD 234

Keypad 11, 445

KEYS 235, 464

Keystack 60, 65, 236

Keystrokes 25

kilo 42

KSTACK 65

- L -

Label 314, 396, 442

LE 31

LEAVE 191

LEAVEFOR 210

Left 11, 117

left brace 446

left bracket 446

left parenthesis 446

Length Limits 25

LEQ 31

less than sign 446

LF 446

LFN 20, 47, 441

LFNToggle 20, 120

Line Continuation 331

Line number 348

linefeed 446

LineToEnd 120

link 245

LIST 235, 237

LIST Keys 123

ListboxBarColors 113

ListColors 113

ListExit 124

ListFind 124

ListFindReverse 124

ListHex 124

ListHighBit 124

ListInfo 124

ListNext 124

ListOpen 124

ListPrevious 125

ListPrint 125

ListRowStart 106

ListStatBarColors 114

ListUnicode 125

ListWrap 125

LOADBTM 242

Local 16

LocalAliases 96

LocalDirHistory 96

LocalFunctions 96

LocalHistory 96

LOG 105, 106, 242

LogErrors 106, 242

LOGICAL EXPRESSION 31

LOGICAL EXPRESSIONS 31

LOGICAL OPERATOR 31

LOGICAL OPERATORS 31

LogName 106, 242

LogOn 106, 242

Long Filename 20, 47

LOW 291

LSS 31

LT 31

- M -

MailAddress 106

MailPassword 107

MailPort 107

MailServer 107

MailUser 107

MAX 136, 291, 316

MD 243

mega 42

MEMORY 245

Menus 68, 69, 70, 71

Metacharacters 262

MIN 136, 291, 316

Miscellaneous Options Tab 86
MKDIR 243
MKLNK 245
More? 27
MOVE 246
MSGBOX 251
Multiple Commands 24
Multiple Filenames 45

- N -

NAK 446
Name 348
named pipes 381
Navigation 22, 54, 56, 342
NE 31
negative acknowledge 446
NEQ 31
Nesting Level 348
New 487
NextFile 120
NextHistory 120
NextINIFile 126
NoClobber 107
Norma (attributeless) file 39
Normal 39, 291, 365, 442
NormalEditKey 120
NormalKey 117
NormalListKey 125
NormalPopupKey 122
NOT 31
Not content-indexed 39, 365
NOTOPMOST 136, 316
NTCMDPROMPT 321
NTFS 439, 444
NTP 97
NTVDM.EXE 421
NUL 446
null 446
Num Lock 234
number sign 446

- O -

OFF 157, 235, 242, 303, 314
Offline 39, 365
OK 251

OKCANCEL 251
ON 157, 235, 242, 252, 303, 314, 327
ON OFF 242
Online Help 423
OPTION 253
Options 4, 82, 83, 84, 85, 86, 87
Options Menu 70
OR 26
Order Form 489
Overstrike 102, 349
Overstrike cursor 349
Overstrike cursor shape 349
Overview 2

- P -

Page and File Prompts 65
ParameterChar 107, 322, 335
Parameters 17, 322
Parent Directory 47
parses 332
Parsing 29
PassiveFTP 108
Paste 118
PATH 254, 341
PathExt 108, 341
PAUSE 256
PauseOnError 96
PDIR 179, 256
percent mark 446
period 446
PGM 291
PgUp 122
Piping 60, 64
Platforms 421
PLAYAVI 260
PLAYSOUND 261
plus sign 446
Pointer 95
POPD 261
PopFile 120
Popup Window Keys 122
Popup Windows 464
PopupWinBegin 123
PopupWinColors 114
PopupWinDel 123
PopupWinEdit 123
PopupWinEnd 123

PopupWinExec 123
 PopupWinHeight 108
 PopupWinLeft 108
 PopupWinTop 108
 PopupWinWidth 108
 POS 136, 291, 316
 precision 103, 374
 PrevFile 121
 PrevHistory 121
 Primary 89, 355, 461
 Primary and Secondary Shells 461
 PRINT 262
 Process ID (PID) 178, 298, 299, 349, 351, 354, 355, 356
 PROMPT 262, 341
 Prompts 60
 Proxy 109
 ProxyPort 109
 ProxyUser 109
 PUSH 264

- Q -

QUERYBOX 265
 question mark 446
 QUIT 266
 quote mark 446
 Quoting 332

- R -

Ranges 40, 42, 44, 45, 378, 382
 RD 267
 Read-only 39, 365
 REALTIME 291
 REBOOT 268
 Recall 13
 record separator 446
 RECYCLE 269
 Recycle Bin 109, 172, 267, 269, 341
 RecycleBin 109
 RECYCLEXCLUDE 341
 Redirection 60, 61, 97
 Redirection and Piping 60
 Reference 430, 445, 458
 Register 418
 Registration 418

Registry 201, 281, 312, 403, 404, 418
 RELATIONAL EXPRESSION 31
 RELATIONAL EXPRESSIONS 31
 RELATIONAL OPERATOR 31
 RELATIONAL OPERATORS 31
 REM 269
 REN 270
 RENAME 270
 RepeatFile 121
 Resizing 77
 RESTORE 136, 316
 RETURN 221, 272
 REXX 334, 405
 Right 11, 118
 right brace 446
 right bracket 446
 right parenthesis 446
 RMDIR 267
 RS 446
 Run Program Dialog 72

- S -

SaveDirCase 126
 SaveHistory 121
 Scan Codes 445, 451, 452
 SCREEN 273
 ScreenBufSize 96
 ScreenColumns 109
 ScreenRows 109
 Scroll Lock 234
 Scrollback Buffer 75, 121
 ScrollDown 121
 Scrolling 25
 ScrollPgDn 121, 122
 ScrollPgUp 121, 122
 ScrollUp 121
 SCRPUT 274
 Secondary 89, 126, 355, 461
 SELECT 275
 SelectColors 114
 SelectStatBarColors 114
 semicolon 446
 SENDMAIL 106, 107, 279
 SEPARATE 291
 ServerCompletion 109
 SET 201, 281, 312, 341
 SETDOS 283, 349

- SETLOCAL 287
 - Setup 418
 - SFN 20, 441
 - SHADOW 195
 - Shape 349
 - SHARED 291
 - SHIFT 288
 - shift in 446
 - shift out 446
 - Shift right 11
 - Shift-Ins 101
 - Shift-left 11
 - Shift-Tab 121
 - Short Filename 20
 - SHORTCUT 289
 - Shortcuts 8, 289
 - SHRALIAS 290, 355
 - SI 446
 - SIZE 291, 316
 - Size Ranges 40, 42, 378, 382
 - SKIP 210
 - slash 446
 - SMPP 290
 - SMS 290
 - SNPP 291
 - SO 446
 - SOH 446
 - Space 446
 - Sparse file 39, 365
 - Special Character Compatibility 21, 100, 107, 335
 - SSLPort 110
 - SSLProvider 110
 - SSLStartMode 110
 - Standard Error 61, 64
 - Standard Input 61, 64
 - Standard Output 61, 64, 377
 - START 29, 291, 356
 - start of header 446
 - start text 446
 - Starting Applications 28
 - Startup 3
 - Startup drive 349
 - Startup OptionsTab 82
 - Status Bar 75
 - STATUS TEST 31
 - STATUS TESTS 31
 - StatusBarOn 110
 - StatusBarText 110
 - StdColors 114
 - stderr 61, 199, 200
 - stdin 61
 - stdout 61, 198, 199, 377
 - Streams 444
 - String Processing 329
 - STX 446
 - SUB 446
 - Subdirectories 440
 - subroutine 221, 272
 - substitute 446
 - Supported Platforms 421
 - SwapScrollKeys 13, 111
 - SWITCH 295
 - Switches 4, 39, 51
 - SYN 446
 - synchronize 446
 - Syntax Coloring 149
 - Syntax Options Tab 86
 - System 39, 365, 442
 - System Errors 465
 - System Variables 201, 281, 312, 339
- T -**
- Tab 120
 - Tables 446, 452
 - TabStops 111
 - TAIL 296
 - Take Command Dialogs 71
 - Take Command Interface 66
 - Take Command Menus 68
 - Take Command Window 66
 - TASKEND 298
 - TASKLIST 299
 - TC32LOG 106
 - TCEXIT 6
 - TCMD32.GPF 419
 - TCMD32.KEY 418
 - TCMD32A.EXE 422
 - TCMD32U.EXE 422
 - TCSTART 6
 - TCStartPath 96
 - TCTOOLBAR 299
 - Technical Support 418, 419
 - Tee 60, 300
 - TEMP 342
 - Temporary 365

Temporary file 39
tera 42
TEXT 301
TFTP 52
THEN 228
ThousandsChar 111
tilde 446
TIME 302
Time Ranges 40, 44, 378, 382
Time Stamps 443
TIMER 303
TimeServer 97
TITLE 136, 304, 316
TITLEPROMPT 342
TMP 342
TOKENS 210
Tool Bar 75, 299
Tool Bar Dialog 72
ToolBarOn 112
ToolBarText 112
TOP 136, 316
TOPMOST 136, 316
TOUCH 304
TRAY 316
TREE 307
TREEEXCLUDE 56, 342
TreePath 97
Troubleshooting 418
TRUENAME 308
TYPE 308

- U -

UNALIAS 138, 201, 310
UNC 439
underscore 446
UNFUNCTION 218, 311
UNICODE 97, 422
UnicodeOutput 97
unit separator 446
UnixPaths 112
UNKNOWN_CMD 138, 310, 458
UNSET 201, 281, 312, 341
UNTIL 191
Up 13, 118, 121
UpdateTitle 112
US 446
User Variables 201, 281, 312

Utilities Menu 70

- V -

Variable Completion 22
Variable Functions 357
Variable Functions by Category 359
Variables 201, 281, 312, 324, 336, 339, 342
Variables by Category 343
VER 313
VERIFY 314
Version 313, 347, 409, 422, 491
vertical bar 446
vertical tab 446
VFAT 439
VOL 314
Volatile Variables 201, 281, 312
Volume 314, 439
VSCRPUT 314
VT 446

- W -

WAIT 291
Waiting for Applications 29
What's New 487
WHICH 315
WHILE 191
Wildcards 36, 410
WIN 291, 316
Win32SFNSearch 112
WINDOW 136, 316
Window Options Tab 83
WindowHeight 97
Windows File Associations 461
Windows System Errors 465
WindowState 97
WindowWidth 97
WindowX 97
WindowY 97
WordLeft 118
WordRight 118

- X -

X3.64 60, 64, 445, 456
X-3.64 486

XOR 486

- Y -

Y 60, 317

YESNO 251

YESNOCANCEL 251

- Z -

ZOOM 195