

Take Command 9.0

*Published By
JP Software Inc.
P.O. Box 328
Chestertown, MD 21620 USA
<http://jpsoft.com>*

Table of Contents

Part I Overview	2
Part II What's New	3
Part III Take Command	16
1 Starting Take Command	17
Take Command Startup Options	17
TCC (4NT) Startup Options	19
TCSTART and TCEXIT	22
TCC Exit Codes	24
2 Configuration Options	24
Initialization (.INI) Files	24
Directives	26
Key Mapping Directives	28
General Input Keys	29
Backspace	29
BeginLine	29
Copy (directive)	29
Del (directive)	30
DelToBeginning	30
DelToEnd	30
DelWordLeft	30
DelWordRight	30
Down	30
EndLine	31
EraseLine	31
ExecLine	31
Ins	31
Left	31
NormalKey	31
Paste	32
Right	32
Up	32
WordLeft	32
WordRight	32
Command Line Editing Keys	32
AddFile	33
AliasExpand	33
CommandEscape	33

DelHistory	33
DirWinOpen	34
EndHistory	34
Help (directive)	34
HelpWord	34
HistWinOpen	34
LastHistory	34
LFNToggle	35
LineToEnd	35
NextFile	35
NextHistory	35
NormalEditKey	35
PopFile	35
PrevArgument	35
PrevFile	36
PrevHistory	36
RepeatFile	36
SaveHistory	36
VariableExpand	36
LIST Keys	36
ListBack	37
ListClipboard	37
ListContinue	37
ListExit	37
ListFind	37
ListFindRegex	38
ListFindRegexReverse	38
ListFindReverse	38
ListHex	38
ListHighBit	38
ListInfo	38
ListNext	38
ListOpen	39
ListPrevious	39
ListPrint	39
ListRefresh	39
ListUnicode	39
ListWrap	39
NormalListKey	40
Popup Window Keys	40
NormalPopupKey	40
PopupWinDel	40
PopupWinEdit	40

PopupWinEditWin	40
PopupWinExec	41
Advanced Directives	41
ClearKeyMap	41
Debug	41
INIQuery	42
LanguageDLL	42
MSAAMenu	43
UpdateINI	43
Take Command Configuration Dialog	43
Windows	43
Tabs	44
Advanced	45
Registration	46
TCC (4NT) Configuration Dialog	46
Startup	47
Windows	49
Command Line	50
Advanced	51
Internet	53
Debugger	54
Updates	55
Registration	55
3 The Take Command Interface	56
The Take Command Window	56
Menus	58
File	58
Edit	59
Tabs	60
View	60
Options	61
Window	61
Help	61
Tool Bars	62
Folders	63
List View	63
Tab Windows	64
Status Bar	64
Keyboard Shortcuts	64
Context Menus	65
Running DOS apps	66
Using the Scrollback Buffer	66
Highlighting & Copying Text	67

Resizing the Take Command Window.....	68
Drag & Drop.....	68
Take Command Dialogs.....	68
Run Program	69
Tab Toolbar	69
Skins and Themes.....	70
4 Directory Navigation	71
CDPATH feature	72
Extended Directory Searches	73
Automatic Directory Changes	76
Directory Aliases	76
5 File Selection	77
Wildcards	77
Ranges	80
Size Ranges.....	82
Date Ranges.....	82
Time Ranges.....	84
File Exclusion Ranges.....	85
Description Ranges.....	86
Attribute Switches	86
Multiple Filenames	87
Include Lists	88
Delayed Variable Expansion	89
LFN File Searches	89
File Lists	90
Switches for File Selection	91
6 Executable Extensions	91
7 Using Internet URLs	92
8 Using FTP and HTTP Servers	93
9 Input / Output Redirection	96
Redirection and Piping	97
Redirection.....	98
Piping	101
ANSI X3.64 Support	101
Keystack	102
Page and File Prompts	102
10 OpenAFS	103
11 The TCC (4NT) Command Line	103
Command Line Editing	104
Command History and Recall	106
Command History Window	107
Local and Global History Lists	108

Command Names & Parameters	109
Conditional Expressions	109
Filename Completion	113
Customizing Filename Completion	115
Filename Completion Window	116
Converting Between Long & Short Filenames	117
Appending Backslashes to Directory Names	117
Extended Parent Directory Names	118
Directory History Window	118
Variable Name Completion	119
Expanding and Disabling Aliases	120
Multiple Commands	120
Conditional Commands	120
Command Grouping	121
Starting Applications	122
Waiting for Applications to Finish	123
Escape Character	124
Command Parsing	124
Command Line Length Limits	126
Special Character Compatibility	126
Date Input Formats	127
Case Sensitivity	127
12 Aliases & Batch Files	128
Aliases	128
Batch Files	130
.BAT, .CMD & .BTM Files.....	131
Using .BAT Files Under TCC.....	131
Echoing in Batch Files.....	132
Special syntax for CMD.EXE compatibility.....	132
Batch File Parameters.....	133
Parameter Quoting.....	134
Using Environment Variables.....	135
Batch File Commands.....	135
Interrupting a Batch File.....	137
Detecting TCC and Take Command.....	137
Using Aliases in Batch Files.....	137
Debugging Batch Files.....	139
String Processing.....	139
Batch File Line Continuation.....	141
Batch File Compression.....	141
REXX Support.....	142
Perl support.....	142
Ruby support.....	143

EXTPROC and SHEBANG Support	143
13 TCC (4NT) Internal Commands	144
Commands by Name	144
Commands by Category	147
?	152
ACTIVATE	152
ALIAS	154
ASSOC	162
ATTRIB	163
BATCOMP	166
BDEBUGGER	166
BEEP	173
BREAK	174
BREAKPOINT	174
CALL	175
CANCEL	176
CD / CHDIR	177
CDD	178
CHCP	180
CLS	181
COLOR	181
COPY	182
DATE	189
DEBUGSTRING	190
DEFER	190
DEL / ERASE	190
DELAY	194
DESCRIBE	195
DETACH	197
DIR	198
DIRHISTORY	208
DIRS	209
DO	210
DRAWBOX	214
DRAWHLINE	215
DRAWVLINE	216
ECHO	217
ECHOERR	218
ECHOS	219
ECHOSERR	220
EJECTMEDIA	220
ENDLOCAL	220
ESET	222

EVENTLOG	223
EVENTMONITOR	224
EXCEPT	225
EXIT	227
FFIND	227
FIREWIREMONITOR	232
FOLDERMONITOR	232
FOR	234
FREE	241
FTYPE	241
FUNCTION	242
GLOBAL	245
GOSUB	247
GOTO	248
HEAD	249
HELP	251
HISTORY	251
IF	253
IFF	254
IFTP	255
INKEY	257
INPUT	259
JABBER	260
KEYBD	261
KEYS	261
KEYSTACK	262
LIST	264
LOADBTM	269
LOADMEDIA	269
LOG	269
MD / MKDIR	271
MEMORY	272
MKLINK	272
MKLNK	273
MOVE	274
MSGBOX	279
NETMONITOR	281
ON	282
OPTION	284
OSD	286
PATH	286
PAUSE	287
PDIR	288

PLAYAVI	291
PLAYSOUND	292
PLUGIN	293
POPD	294
POSTMSG	294
PRINT	295
PRIORITY	295
PROCESSMONITOR	296
PROMPT	297
PUSHD	299
QUERYBOX	300
QUIT	301
RD / RMDIR	301
REBOOT	303
RECYCLE	304
REM	304
REN / RENAME	305
RETURN	307
REXEC	308
RSHELL	309
SCREEN	310
SCRIPT	311
SCRPUT	311
SELECT	312
SENDMAIL	316
SERVICEMONITOR	318
SERVICES	319
SET	319
SETDOS	323
SETLOCAL	326
SHIFT	327
SHORTCUT	328
SHRALIAS	329
SMPP	330
SNMP	330
SNPP	331
START	331
SWITCH	335
SYNC	336
TAIL	339
TASKDIALOG	341
TASKEND	342
TASKLIST	342

TCFILTER	343
TCTOOLBAR	343
TEE	344
TEXT	345
TIME	347
TIMER	347
TITLE	349
TOUCH	349
TRANSIENT	352
TREE	352
TRUENAME	353
TYPE	354
UNALIAS	355
UNFUNCTION	356
UNSET	357
USBMONITOR	358
VER	359
VERIFY	360
VOL	360
VSCRPUT	360
WHICH	361
WINDOW	362
WMIQUERY	364
Y	364
14 Variables & Functions	365
System Variables	367
CDPATH	367
CMDLINE	368
COLORDIR	368
COMSPEC	368
FILECOMPLETION	368
HISTORYEXCLUDE	368
PATH	368
PATHEXT	369
PROMPT	369
RECYCLEEXCLUDE	369
TEMP	370
TCMD	370
TCMDVER	370
TITLEPROMPT	370
TMP	370
TREEEXCLUDE	370
VARIABLEEXCLUDE	370

CMD.EXE Compatibility Variables	371
COPYCMD variable	371
DIRCMD variable	371
String substitution	372
Internal Variables	372
Variables by Name	373
Variables by Category	376
! (Variable)	380
? variable	380
_? variable	380
= pseudovvariable	381
+ pseudovvariable	381
_4VER	381
_ACSTATUS	381
_ADMIN	381
_AFSWCELL	381
_ALT	381
_ANSI	382
_BATCH	382
_BATCHLINE	382
_BATCHNAME	382
_BATCHTYPE	382
_BATTERY	382
_BATTERYLIFE	382
_BATTERYPERCENT	382
_BDEBUGGER	382
_BG	382
_BOOT	383
_BUILD	383
_CAPSLOCK	383
_CDROMS	383
_CHILDPID	383
_CI	383
_CMDLINE	383
_CMDPROC	383
_CMDSPEC	383
_CO	383
_CODEPAGE	383
_COLUMN	384
_COLUMNS	384
_CONSOLEPIDS	384
_COUNTRY	384
_CPU	384

_CPUUSAGE.....	384
_CTRL	384
_CWD	384
_CWDS.....	385
_CWP	385
_CWPS.....	385
_DATE	385
_DATETIME.....	385
_DAY	385
_DETACHPID.....	385
_DISK	385
_DNAME.....	385
_DOS	385
_DOSVER.....	386
_DOW	386
_DOWF.....	386
_DOWI	386
_DOY	386
_DRIVES.....	386
_DST	386
_DVDS	386
_ECHO	386
_EDITMODE.....	386
_EXECSTR.....	386
_EXIT	386
_EXPANSION.....	387
_FG	387
_FTPERROR.....	387
_HDRIVES.....	387
_HLOGFILE.....	387
_HOST	387
_HOUR	387
_HWPROFILE.....	388
_IDLETICKS.....	388
_IDOW	388
_IDOWF.....	388
_IFTP	388
_IFTPS	388
_IMONTH.....	388
_IMONTHF.....	388
_ININAME	388
_IP	388
_ISODATE.....	388

_KBHIT	388
_LALT	388
_LASTDISK	389
_LCTRL	389
_LOGFILE	389
_LSHIFT	389
_MINUTE	389
_MONITORS	389
_MONTH	389
_MONTHF	389
_NUMLOCK	389
_OPENAFS	390
_OSBUILD	390
_PARENT	390
_PID	390
_PIPE	390
_PPID	390
_RALT	390
_RCTRL	390
_READY	390
_REGISTERED	390
_ROW	390
_ROWS	391
_RSHIFT	391
_RUBYTYPE	391
_RUBYVALUE	391
_SCROLLLOCK	391
_SECOND	391
_SELECTED	391
_SHELL	391
_SHELLS	391
_SHIFT	391
_SHORTCUT	392
_SHRALIAS	392
_STARTPATH	392
_STARTPID	392
_STDIN	392
_STDOUT	392
_STDERR	392
_STZN	392
_STZO	392
_SYSERR	392
_TCFILTER	392

_TCFOLDER.....	393
_TCTAB.....	393
_TIME	393
_TRANSIENT.....	393
_TZN	393
_TZO	393
_UNICODE.....	393
_UTCDATE.....	393
_UTCDATETIME.....	393
_UTCHOUR.....	393
_UTCISODATE.....	393
_UTCMINUTE.....	393
_UTCSECOND.....	393
_UTCTIME.....	394
_VIRTUALPC.....	394
_VMWARE.....	394
_VXPIXELS.....	394
_VYPIXELS.....	394
_WINDIR.....	394
_WINFGWINDOW.....	394
_WINNAME.....	394
_WINSYSDIR.....	394
_WINTICKS.....	394
_WINUSER.....	394
_WINVER.....	394
_WINTITLE.....	394
_WOW64.....	394
_XPIXELS.....	394
_YEAR	394
_YPIXELS.....	395
ERRORLEVEL.....	395
Variable Functions	395
Functions by Name.....	397
Functions by Category.....	401
Date Display Formats.....	406
@ABS	407
@AFSCELL.....	407
@AFSMOUNT.....	407
@AFSPATH.....	407
@AFSSYMLINK.....	407
@AFSVOLID.....	408
@AFSVOLNAME.....	408
@AGEDATE.....	408

@ALIAS.....	408
@ALTNAME.....	408
@ASCII.....	409
@ASSOC.....	409
@ATTRIB.....	409
@AVERAGE.....	410
@CAPI	410
@CAPS.....	411
@CDROM.....	411
@CEILING.....	411
@CHAR.....	411
@CLIP	412
@CLIPW.....	412
@COLOR.....	412
@COMMA.....	412
@COMPARE	413
@CONSOLE.....	413
@CONVERT.....	413
@COUNT.....	413
@CRC32.....	413
@CWD	414
@CWDS.....	414
@DATE.....	414
@DAY	414
@DEC	415
@DECIMAL.....	415
@DESCRIPT.....	415
@DEVICE.....	416
@DIGITS.....	416
@DIRSTACK.....	416
@DISKFREE.....	416
@DISKTOTAL.....	417
@DISKUSED.....	417
@DOMAIN.....	417
@DOW	417
@DOWF.....	418
@DOWI.....	418
@DOY	419
@DRIVETYPE.....	419
@DRIVETYPEEX.....	419
@ENUMSERVERS.....	419
@ENUMSHARES.....	420
@ERRTEXT.....	420

@EVAL.....	420
@EXEC.....	424
@EXECSTR.....	424
@EXETYPE.....	424
@EXPAND.....	425
@EXT	425
@FIELD.....	426
@FIELDS.....	427
@FILEAGE.....	427
@FILECLOSE.....	427
@FILEDATE.....	428
@FILENAME.....	428
@FILEOPEN.....	428
@FILEREAD.....	429
@FILEREADB.....	429
@FILES.....	430
@FILESEEK.....	431
@FILESEEKL.....	431
@FILESIZE.....	432
@FILETIME.....	432
@FILEWRITE.....	433
@FILEWRITEB.....	433
@FINDCLOSE.....	434
@FINDFIRST.....	434
@FINDNEXT.....	435
@FLOOR.....	435
@FORMAT.....	435
@FORMATN.....	436
@FORMATNC.....	436
@FSTYPE.....	436
@FTYPE.....	437
@FULL	437
@FUNCTION.....	437
@GETDIR.....	437
@GETFILE.....	438
@GETFOLDER.....	438
@GROUP.....	438
@HISTORY.....	439
@IDOW.....	439
@IDOWF.....	439
@IF	439
@INC	440
@INDEX.....	440

@INIREAD.....	441
@INIWRITE.....	441
@INODE.....	442
@INSERT.....	442
@INSTR.....	443
@INT	443
@IPADDRESS.....	443
@IPNAME.....	444
@ISALNUM.....	444
@ISALPHA.....	444
@ISASCII.....	444
@ISCNTRL.....	445
@ISDIGIT.....	445
@ISPRINT.....	445
@ISPROC.....	445
@ISPUNCT.....	445
@ISSPACE.....	446
@ISXDIGIT.....	446
@JUNCTION	446
@LABEL.....	446
@LCS	446
@LEFT	446
@LEN	447
@LFN	447
@LINE	447
@LINES.....	448
@LINKS.....	448
@LOWER.....	448
@LTRIM.....	448
@MAKEAGE.....	449
@MAKEDATE.....	449
@MAKETIME.....	449
@MAX	450
@MD5	450
@MIN	450
@MONTH.....	450
@NAME.....	451
@NUMERIC.....	451
@OPTION.....	452
@PATH.....	452
@OWNER.....	452
@PERL.....	452
@PING	453

@QUOTE.....	453
@RANDOM.....	453
@READSCR.....	453
@READY.....	453
@REGCREATE.....	454
@REGDELKEY.....	454
@REGEX.....	454
@REGEXINDEX.....	454
@REGEXIST.....	454
@REGEXSUB.....	455
@REGQUERY.....	455
@REGSET.....	455
@REGSETENV.....	455
@REGTYPE.....	455
@REMOTE.....	456
@REMOVABLE.....	456
@REPEAT.....	456
@REPLACE.....	456
@REVERSE.....	457
@REXX.....	457
@RIGHT.....	457
@RTRIM.....	457
@RUBY.....	458
@SCRIPT.....	458
@SEARCH.....	458
@SELECT.....	458
@SERIAL.....	459
@SERVER.....	459
@SFN.....	460
@SHA1.....	460
@SHA256.....	460
@SHA384.....	460
@SHA512.....	460
@SHFOLDER.....	461
@SIMILAR.....	462
@SNAPSHOT.....	462
@STRIP.....	462
@SUMMARY.....	462
@SUBSTR.....	463
@SUBST.....	463
@SYMLINK.....	463
@TIME.....	463
@TIMER.....	463

@TRIM	464
@TRUENAME	464
@TRUNCATE.....	464
@UNC	464
@UNICODE.....	464
@UNIQUE.....	465
@UNQUOTE.....	465
@UNQUOTES.....	465
@UPPER.....	465
@VERINFO.....	465
@WATTRIB.....	466
@WILD	466
@WINAPI.....	467
@WINCLASS.....	467
@WINEXENAME.....	467
@WININFO.....	467
@WINMEMORY	468
@WINMETRICS.....	468
@WINPOS.....	470
@WINSTATE.....	470
@WINSYSTEM.....	470
@WMI	472
@WORD.....	472
@WORDS.....	473
@WORKGROUP.....	473
@XMLCLOSE.....	473
@XMLNODES.....	473
@XMLOPEN.....	474
@XMLXPath.....	474
@YEAR.....	474

Part IV Troubleshooting 474

1 Registration	474
2 Troubleshooting, Service & Support	475
Technical Support	475
Contacting JP Software	477
3 Supported Platforms	477
4 Help File	477
5 Error Messages	478

Part V Reference Information 484

1 CMD.EXE Comparison	484
2 Limits	490

3	File Systems & File Name Conventions	490
	Drives & Volumes	491
	File Systems	491
	Directories & Subdirectories	492
	File Names	493
	File Attributes	494
	File Time Stamps	495
	NTFS File Streams	496
4	Regular Expression Syntax	496
5	XML in Take Command	502
6	Miscellaneous Reference Information	504
	Executable Files & File Searches	504
	Windows File Associations	506
	Popup Windows	507
	Windows System Errors	507
7	ASCII and Key Names	510
	ASCII Tables	511
	Key Names	515
8	ANSI X3.64 Command Reference	516
9	Colors, Color Names & Codes	518
Part VI	Glossary	519
1	Glossary - A	520
2	Glossary - B	521
3	Glossary - C	522
4	Glossary - D	525
5	Glossary - E	526
6	Glossary - F	528
7	Glossary - G	528
8	Glossary - H	529
9	Glossary - I	529
10	Glossary - J	530
11	Glossary - K	530
12	Glossary - L	530
13	Glossary - M	531
14	Glossary - N	531
15	Glossary - O	532
16	Glossary - P	532
17	Glossary - Q	533
18	Glossary - R	533

19 Glossary - S	534
20 Glossary - T	535
21 Glossary - U	536
22 Glossary - V	536
23 Glossary - W	536
24 Glossary - X	537
Part VII Copyright & Version	537
Index	538

ACKNOWLEDGEMENTS

We couldn't produce products like Take Command without the dedication and quality work of many people. A special thanks to all of you who helped make Take Command elegant, reliable, and friendly.

Copyright © 2008, JP Software Inc., All Rights Reserved. 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



Indispensable Windows Scripting & Console Tools
For 15 years & Over 250,000 Licenses

TAKE COMMAND 9.0

Welcome to our help! We have designed this help file to accompany our product **Take Command**. **Take Command** is designed for Windows XP, Windows 2003, Windows Vista, and Windows 2008. **Take Command** combines the best features of the GUI and character-mode interfaces. You can have multiple console applications open in tabbed windows, with a Windows Explorer-like interface available for those times when you need a visual look at your folders.

Take Command is composed of three elements which work closely together:

Take Command Environment - A rich development and operations environment that allows you to:

- Create and edit command scripts
- Debug scripts
- Run multiple console applications simultaneously in tabbed windows, including our own Take Command Console (**TCC**), CMD.EXE, PowerShell and bash. **Take Command** will display output much faster (up to 10x!) than running the application in a console window.
- Cut and paste text
- Drag and drop files into tab windows from an Explorer-like environment, other applications, or the desktop

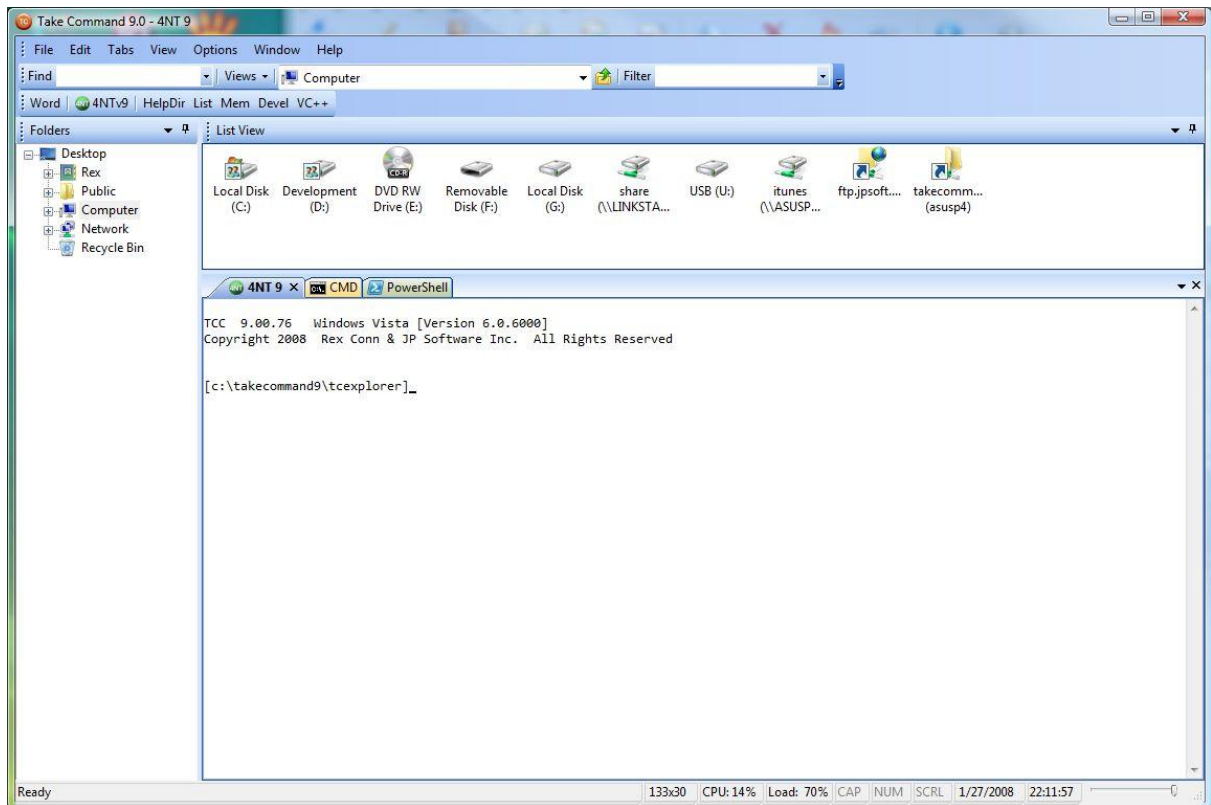
Take Command Console (TCC) - A command processor compatible with CMD.EXE (the default command processor in Windows XP / 2003 / Vista / 2008) but substantially enhanced with thousands of additional features. **TCC** provides the ability to:

- Interactively run commands, such as DIR, COPY, etc
- Interactively run batch script files, such as .CMD, .BAT or .BTM scripts
- Run batch scripts as background processes based on timed schedules or operational triggers, such as changes in the system environment

Take Command Language - A mature scripting language based on and compatible with CMD.EXE, but massively enhanced. It includes:

- 150+ internal commands
- 220+ functions
- 140+ variables
- Hundreds of additional options for CMD compatible commands
- Additional underlying capabilities, such as the ability to treat FTP and HTTP sites as if they were local disk drives

The following image shows how the pieces fit together. The overall environment surrounds a set of consoles, each with its own tab. Each console can run commands in the Take Command language (or other languages, such as PowerShell, if you open additional tab windows).



JP Software Inc.
P.O. Box 328, Chestertown, MD 21620, USA
phone: (800) 595-8197
email: support@jpsoft.com
web: <http://jpsoft.com/>

2 What's New

NEW VERSION OVERVIEW - Take Command 9.0

This new version of Take Command incorporates three of our products (Take Command, 4NT, and TCI) into a single product with a new interface. 4NT (now renamed **TCC**) can still be run separately from Take Command for those users who don't want to use the new GUI, or who need to invoke a non-interactive command processor from batch or make files.

This is a summary of the compatibility fixes and new features. For complete details, see the appropriate topics in this help file.

FEATURE LIST:

Take Command now supports multiple tabbed windows.

The maximum command line size has been increased to 65,535 characters.

The maximum single command size has been increased to 32,767 characters.

The maximum single argument size has been increased to 4,095 characters.

The maximum filename size has been increased to 4,095 characters.

The maximum number of command arguments has been increased to 4096.

Take Command now supports mouse select / copy screen regions

The commands that use /P (DIR, ALIAS, SET, etc.) will now overwrite the page prompt when they display the next line.

Added support for FTP access where the user does not have access to the root directory.

Added regular expression support to FTP / IFTP. The syntax for IFTP is:

```
"ftp:::regex"
```

FTP.CFG now supports multiple users/passwords for a single FTP site. You need to add an "alias" (enclosed in parentheses) for the ftp site following the name in the ftp.cfg file. For example:

```
jpsoft.com (jpadmin) Bob AdminPassword  
jpsoft.com (jppublic) anonymous Bob@jpsoft.com
```

You can then access the server as "ftp://jpadmin" or "ftp://jppublic".

Removed full-screen support in 4NT.

Directory aliases can now also be a single numeric character. For example:

```
alias 1:=c:\program files
```

Tweaked the parser a bit to better handle filenames with embedded back quotes.

If you try to reference a network disk that has become disconnected, Take Command will automatically try to reset the connection.

The password entries in the option dialogs (Email and Firewall) now display dots rather than showing the password in clear text. (It is still saved in the TCMD.INI file in clear text.)

Added support for key mapping the Alt punctuation keys `-[[]\;',./ (for example: Alt-.). Note that you cannot combine these key definitions with the Ctrl and/or Shift keys (Windows does not generate keystrokes for those key combinations).

The popup windows (history, directory history,
The default 4NT .INI file name has changed from 4NT.INI to TCMD.INI. (You can still override the name on the 4NT command line.)

Dropped the Spanish language dll (I haven't been able to find anybody to do the string translations).

Startup Options:

Added a new startup switch:

```
/l[ips]  
/li - don't load default .INI file  
/lp - don't load plugins
```

/ls - don't execute TCSTART

.INI directives:

DirJunctions=YES|no - display symbolic links in DIR and PDIR.

DirHistFile=file - load directory history at startup & save it at shutdown.

ErrorColors - display error messages (written to STDERR) in a different color.

MSAAMenu=yes|NO - If you're using a screen reader that has problems with the new **Take Command** detachable menu, set MSAAMenu to YES.

PrevArgument=Key Keystroke to recall the last argument from the previous command line (see Command Line Editing). The default is Ctrl-P.

UpdateIni=YES|no - option to allow or disallow changes to the .INI.

CDDWinColors, PopupWinColors - have been removed (obsolete).

PopupWinBegin, PopupWinEnd - have been removed (obsolete).

ScrollUp, ScrollDown, ScrollPgUp, ScrollPgDn - have been removed (obsolete).

ListColors, ListboxBarColors, ListInverseColors, SelectColors, SelectStatBarColors - have been removed from the OPTION dialog.

SaveDirCase - has been removed. (It now always defaults to preserving the directory case.)

Command Line Editing:

The command history and directory history popup windows have a toolbar which allows you to edit, remove, or move the history entries.

(4NT) The popup windows are now GUI (like Take Command) rather than character mode.

A Ctrl-0 through Ctrl-9 will insert the matching argument from the previous command line. For example:

```
c:\> echo now is the time for all good men
```

A Ctrl-0 will insert "echo" at the current cursor position; a Ctrl-4 will insert "time", etc.

A Ctrl-P will insert the last argument from the previous command line.

New Commands:

DEFER - execute a command line after the batch file exits. The syntax is:

DEFER command

EVENTMONITOR - Monitor event logs. The syntax is:

EVENTMONITOR [/C name] [server name /S"source" /T"type" /D"description" n command]

server UNC name of the computer with the log file, or LOCAL for the local computer
name log name
n Number of repetitions (or **FOREVER**)
command Command to execute when condition is triggered

[/C\(ancel\)](#) ^[224]
[/D"description"](#) ^[225]

[/S"source"](#) ^[225]
[/T"type"](#) ^[225]

FIREWIREMONITOR - monitor FireWire connections and execute a command when a device is connected or disconnected. The syntax is:

firewiremonitor [/C name] [name CONNECTED | DISCONNECTED n command...]

/C - cancel monitoring the FireWire device. If no name is provided, cancels all FireWire monitoring.

name - device name. The device name can either be the device ID name or the "friendly" name. The name can include wildcards.

n - number of repetitions (or "FOREVER")

FOLDERMONITOR - Monitor folder and/or file creation, modification, and deletion. The syntax is:

FOLDERMONITOR [/C name] [/S name /I"file" /E"file" CREATED DELETED MODIFIED RENAMED n command]

name Folder or file name
CREATED Execute the command if the folder or file is created
DELETED Execute the command if the folder or file is deleted
MODIFIED Execute the command if the folder or file is modified
RENAMED Execute the command if the folder or file is renamed
n Number of repetitions (or **FOREVER**)
command Command to execute when condition is triggered

[/C\(ancel\)](#) ^[234]
[/E\(xclude\)](#) ^[234]

[/I\(include\)](#) ^[234]
[/S\(ubdirectories\)](#) ^[232]

LOADMEDIA - like EJECTMEDIA, but closes the drive door.

NETMONITOR - monitor network connections and execute a command when a network is connected or disconnected. The syntax is:

NETMONITOR [/C name] [name CONNECTED | DISCONNECTED n command...]

name - network name ("LAN" for local network, "WAN" for dialup, or (wireless) network name.

/C - cancel monitoring the network. If no name is provided, cancels all network monitoring.

n - number of repetitions (or "FOREVER")

PROCESSMONITOR - monitor processes and execute a command when a process is started or ended. The syntax is:

PROCESSMONITOR [/C name] [name STARTED | PAUSED | STOPPED n command...]

name - the full process pathname (may include wildcards).

/C - cancel monitoring the specified process. If no name is provided, cancels all process monitoring.

n - number of repetitions (or "FOREVER")

SERVICEMONITOR - monitor Windows services and execute a command when a service is started, paused, or stopped. The syntax is:

SERVICEMONITOR [/C name] [name STARTED | PAUSED | STOPPED n command...]

name - the service name (may include wildcards).

/C - cancel monitoring the specified service. If no name is provided, cancels all service monitoring.

n - number of repetitions (or "FOREVER")

SERVICES - display, stop, or start system services. The syntax is:

SERVICES [/P /R /S] [name ...]

The **name** is the service name, not the display name. **name** can contain wildcards.

/P(ause) after each page

/R(un) the specified service(s)

/S(top) the specified service(s)

TASKDIALOG - Display a Windows Vista Task Dialog (requires Vista or later). The syntax is:

TASKDIALOG [/I /S /W] buttontype "title" "instruction" [text]

buttontype One or more of OK, CANCEL, YES, NO, RETRY, CANCEL, and/or CLOSE

title Text for the task dialog title

instruction Text for the main instruction

text Optional additional text that appears below the main instruction, in a smaller font

/I(nformation icon)

/S(top icon)

/W(arning icon)

TCFILTER - Display or set the filter for the Take Command list view. The syntax is:

TCFILTER [/C filter]

filter New filter value (i.e., *.doc)

/C(lear filter)

USBMONITOR - monitor USB connections and execute a command when a device is connected or disconnected. The syntax is:

USBMONITOR [/C name] [name CONNECTED | DISCONNECTED n command...]

name - device name. The device name can either be the device ID name or the "friendly" name. The name can include wildcards.

/C - cancel monitoring the USB device. If no name is provided, cancels all USB monitoring.

n - number of repetitions (or "FOREVER")

Commands:

ACTIVATE - added two new options:

DISABLE - disable the specified window (no keyboard or mouse input)

ENABLE - enable the specified window

ACTIVATE - RESTORE now supports restoring any app from the system tray.

ASSOC - now ignores remarks (lines prefixed with ":") in files read with /R.

CDD - Added several new options:

/NJ - skip junctions when indexing subdirectories.

/Sn - limit the directory recursion depth when indexing subdirectories.

/T - also change the TCMD Folders and List View directory.

/TO - only change the Take Command Folders and List View directory

COPY - added several new options:

/LD - When used with /S, if the source is a symbolic or hard link to a directory, COPY will create a copy of the link in the target directory instead of copying the subdirectory tree.

/MD - create the target directory if it doesn't exist. (Note that you *must* either terminate the target directory name with a trailing \ or specify a filename component; otherwise COPY cannot tell what you want for the directory and what you want for the filename!)

/ND - skip hidden subdirectories when used with /S

/NJ - skip junctions when used with /S

COPY - the /G (percent copied) option now also tells you the remaining time (with local, network, ftp, ftps, iftp, iftps, and tftp copies).

COPY - the /V(erify) option when combined with the /G (percent copied) option now also tells you the remaining time (with local, network, ftp, ftps, iftp, iftps, and tftp copies).

COPY - added support for connected web folder names in non-English environments. (For example, in French Windows uses "_fichiers" instead of ".files".)

DEL - added new options:

/B - if DEL can't delete the file (i.e., access denied) it will schedule it to be deleted at the next reboot.

/ND - skip hidden subdirectories when used with /S

/NF - don't display "bytes freed" summary

/NJ - skip junctions when used with /S

DIR - added option to skip junctions (/NJ) when used with /S.

DIR - added option to skip hidden subdirectories (/ND) when used with /S.

DIR - added option to display file times in UTC. The syntax is:

/T:[ACW][U]

For example to display the last write time in UTC:

dir /t:wu

DIR - added option (/NF) to suppress "bytes free" display.

DO - added option to skip junctions (/NJ) when used with /S.

DO - added option to skip hidden subdirectories (/ND) when used with /S.

DO - added an option to specify the start directory (primarily for use with /S). The syntax is:

/D"directoryname"

DO - the UNTIL DATETIME syntax now supports either YYYY-MM-DD HH:MM:SS or YYYYMMDDHHMMSS format.

FTYPE - now ignores remarks (lines prefixed with ":") in files read with /R.

IF ISFILE - supports wildcards with FTP files.

LIST - /X will now display 32 bytes per row if the screen column width is ≥ 144 .

LIST - added a new option (/XS) to display spaces rather than periods for non-printable characters when in hex mode. You can also toggle between spaces & periods with the 'S' key while displaying a file in hex mode.

LIST - added a new option (/U) which displays a ruler on the second line of the display.

LIST - the various popup windows (search, goto, etc.) are now GUI windows rather than character mode windows.

LOG - you can now specify up to four different log names:

The command log (default name **TCCommandLog**).

The history log (default name **TCHistoryLog**).

The error log (default name **TCErrrorLog**).

The "log all output" (default name **TCLogAll**).

(The logs can also be combined into common log names if desired with the LOG /W filename option.)

MKLINK - now copies an existing description to the link.

MKLNK - now copies an existing description to the link.

MOVE - added new options:

/B - if MOVE can't move the file (i.e., access denied) it will schedule it to be moved at the next reboot.

/LD - When used with /S, if the source is a symbolic or hard link to a directory, MOVE will create the link in the target directory instead of moving the subdirectory tree.

/MD - create the target directory if it doesn't exist. (Note that you **must** either terminate the target directory name with a trailing \ or specify a filename component; otherwise MOVE cannot tell what you want for the directory and what you want for the filename!)

/ND - skip hidden subdirectories when used with /S

/NJ - skip junctions when used with /S

MOVE - the /G (percent moved) option now also tells you the remaining time (with local, network, ftp, ftps, iftp, iftps, and tftp copies).

MOVE - the /V(erify) option when combined with the /G (percent moved) option now also tells you the remaining time (with local, network, ftp, ftps, iftp, iftps, and tftp copies).

ON - added several new tests:

ON CLOSE - invoked when the **Take Command** tab is closed.

ON LOGOFF - invoked when the user logs off.

ON SHUTDOWN - invoked when the system is shut down.

ON LBUTTON - invoked when the left mouse button is clicked.

ON MBUTTON - invoked when the middle mouse button is clicked.

ON RBUTTON - invoked when the right mouse button is clicked.

OPTION - The OPTION dialogs have been redone. With the exception of the debugging and key directives, all of the .INI directives are now configurable from OPTION.

OSD - added an option (/C) to close an existing OSD display. (Note that this will only close the most recent OSD; if you have multiple OSDs running you cannot select which one to close!)

OSD - added support for multiple lines of text.

OSD - now expands tabs (^t).

OSD - now accepts leading whitespace if the string is enclosed in single back quotes.

OSD - added an option (/V) for vertical display.

PDIR - added option to skip junctions (/NJ) when used with /S.

PDIR - added option to skip hidden subdirectories (/ND) when used with /S.

PDIR - added option to display file times in UTC. See DIR for the syntax.

PLAYSOUND - added two new options:

- /M - mute
- /U - unmute

PUSHD - now sets ERRORLEVEL (for compatibility with CMD.EXE).

QUERYBOX - added new option:

- /POS=top,left - set the dialog position. (The default is to center the QUERYBOX dialog within the 4NT / TCMD window.)

REN - added new options:

- /B - if REN can't rename the file (i.e., access denied) it will schedule it to be renamed at the next reboot.

- /MD - create the target directory if it doesn't exist. (Note that you **must** either terminate the target directory name with a trailing \ or specify a filename component; otherwise REN cannot tell what you want for the directory and what you want for the filename.)

SETDOS - added new argument for /X:

- /X-A - disable user-defined functions
- /X+A - enable user-defined functions

SENDMAIL - added support for multiple /H (header) arguments.

START - if you specify * as the password for /RUNAS, START will prompt you to enter the password. (Useful when you don't want to put the password in a batch file.)

START - added support for alternate Windows shells when using the "start c:\directory" syntax (which normally invokes an Explorer shell starting in the specified directory).

START - added new options:

- /MONITOR=n - start the process on a specific monitor (1 - n). (Note that this will only work with apps that do not try to position their window at startup.)

- /TAB - start the process in a new **Take Command** tab window.

START - removed the /CM option.

SYNC - added option to skip junctions (/NJ) when used with /S.

SYNC - added option to skip hidden subdirectories (/ND) when used with /S.

SYNC - the /G (percent copied) option now also tells you the remaining time (with local, network, ftp, ftps, iftp, iftps, and tftp copies).

TASKLIST - now adds a * after the PID of the current process.

TOUCH - added UTC time option to the /D /T switches. The syntax is:

```
/D:[ACW][U]  
/T:[ACW][U]
```

TREE - added option to skip junctions (/NJ) when used with /S.

UNALIAS - added option to preserve the specified aliases on a wildcard UNALIAS (i.e., "UNALIAS *" or "UNALIAS A*"). The syntax is:

```
unalias (aliasname1 aliasname2) *
```

UNFUNCTION - added option to preserve the specified variables on a wildcard UNFUNCTION. See UNALIAS for details.

UNSET - added option to preserve the specified variables on a wildcard UNSET. See UNALIAS for details.

WINDOW - removed the FS and WIN options.

WINDOW - added two new options:

DISABLE - disable the window (no keyboard or mouse input)

ENABLE - enable the window

Environment Variables:

TCMD creates two environment variables that can be queried by its child tab window processes:

TCMD - the full pathname of the Take Command executable

TCMDVER - the version & build number (i.e., 9.00.39).

Internal Variables:

_ADMIN - returns 1 if the current process is running as an administrator.

_CONSOLEPIDS - returns a space-delimited list of the PIDs of all processes attached to this console.

_MONITORS - returns the number of monitors.

_PARENT - returns the name of the parent process.

_SHORTCUT - full pathname of the shortcut file that started this process (or an empty string if it wasn't started by a shortcut).

_TCFOLDER - returns the selected folder in the Folders window if in a Take Command tab window.

_TCTAB - returns 1 if 4NT is in a Take Command tab window.

_UTCTIME - returns the current time in UTC.

_UTCDATE - returns the current UTC date.

_UTCISODATE - returns the current UTC date in ISO format (yyyy-mm-dd).

_UTCDATETIME - returns the current date and time in UTC.

_UTCHOUR - current UTC hour.

_UTCMINUTE - current UTC minute.

_UTCSECOND - current UTC second.

_VXPixels - virtual screen (multiple monitor) horizontal size in pixels.

_VYPixels - virtual screen (multiple monitor) vertical size in pixels.

Variable Functions:

@EVAL - added large number support. The maximum number size is now 20,000 digits (10,000 digits to the left of the decimal and 10,000 decimal places). If you want to use more than the default decimal values you'll need to change your EvalMax directive or use the "=x.y" format in @EVAL. (Note that this should also eliminate the rounding errors sometimes seen in floating point calculations in previous versions.)

@FILEAGE - will display the age in UTC if you append a "u" to the second argument.

@FILEDATE - will display the date in UTC if you append a "u" to the second argument.

@FILEREADB[n,length] - new function to return bytes from a file as a string of numeric ASCII values.

@FILES - added a new optional argument to scan the specified directory and its subdirectories. The syntax is:

%@files[/S[n] ...]

For example, to return the number of *.EXE files in the \Windows directory and all its subdirectories:

%@files[/s \windows*.exe]

@FILESIZE - added optional range support. See the @FILE syntax for details.

@FILESIZE - added a new optional argument to scan the specified directory and its subdirectories. The syntax is:

%@filesize[/S[n] ...]

For example, to return the size of all of the *.EXE files in the \Windows directory and all its subdirectories:

%@filesize[/s \windows*.exe]

@FILETIME - will display the time in UTC if you append a "u" to the second argument.

@FILEWRITEB - if the length argument is -1, @FILEWRITEB will read the string argument as a series of ASCII values in decimal or hex to write to the file. For example:

```
echo %@filewriteb[%file,-1,0xe0 0xF2 0xA9]
```

@FORMATNC - new function; like @FORMATN, but inserts thousands separators.

@FSTYPE - added support for UNC names.

@GETFOLDER - added optional second argument to set the text displayed above the tree list. For example:

```
echo %@getfolder[c:\windows,Browse the Windows Directories]
```

@ISPROC - new function which returns 1 if the specified process ID is an active process.

@REGTYPE - returns the registry variable type.

@REGSET - now allows setting REG_MULTI_SZ.

@SERVER - return information about the specified server. The syntax is:

```
@server[machinename,info]
```

where "info" is the type of information you want. The types are:

Name - return the server name

Comment - return the server comment

Version - the OS version (major version + minor version).

Users - the number of users who can attempt to log on the server.

Disconnect - the auto-disconnect time, in minutes.

Hidden - returns 1 if the server is hidden, 0 if it is visible

UserPath - the path to user directories

Type - return the type of the server. This is a combination of the following flags (you can use the .AND. operator in IF / IFF to test individual flags):

1	A LAN Manager workstation
2	A LAN Manager server
4	Any server running with Microsoft SQL Server
8	Primary domain controller
0x10	Backup domain controller
0x20	Server running the Timesource service
0x40	Apple File Protocol server
0x80	Novell server
0x100	LAN Manager 2.x domain member
0x200	Server sharing print queue
0x400	Server running dial-in service
0x800	UNIX / Linux server
0x1000	Windows Server 2003, Windows XP, Windows 2000, or Windows NT
0x2000	Server running Windows for Workgroups

0x4000	Microsoft File and Print for NetWare
0x8000	Windows server that is not a domain controller
0x10000	Server that can run the browser service
0x20000	Server running a browser service as backup
0x40000	Server running the master browser service
0x80000	Server running the domain master browser
0x40000	Windows 95/98/Me
0x1000000	Server clusters available in the domain
0x2000000	Terminal Server
0x4000000	Cluster virtual servers available in the domain
0x40000000	Servers maintained by the browser
0x80000000	Primary domain

@SHFOLDER - returns Windows folder locations (which vary in different versions of Windows and if the user has altered the defaults). The syntax is:

@SHFOLDER[n]

Where "n" is:

0	Desktop
2	Start Menu\Programs
5	My Documents
6	<user name>\Favorites
7	Start Menu\Programs\Startup
8	<user name>\Recent
9	<user name>\SendTo
11	<user name>\Start Menu
13	"My Music" folder
14	"My Videos" folder
16	<user name>\Desktop
19	<user name>\nethood
20	windows\fonts
21	templates
22	All Users\Start Menu
23	All Users\Start Menu\Programs
24	All Users\Startup
25	All Users\Desktop
26	<user name>\Application Data
27	<user name>\PrintHood
28	<user name>\Local Settings\Application Data (non roaming)
29	non localized startup
30	non localized common startup
31	common favorites
32	Internet cache
33	cookies
34	history
35	All Users\Application Data
36	Windows directory
37	Windows system directory
38	Program Files
39	Program Files\My Pictures
40	USERPROFILE
41	X86 system directory on RISC
42	x86 c:\Program Files on RISC
43	c:\Program Files\Common
44	x86 Program Files\Common on RISC
45	All Users\Templates

```

46 All Users\Documents
47 All Users\Start Menu\Programs\Administrative Tools
48 <user name>\Start Menu\Programs\Administrative Tools
53 All Users\My Music
54 All Users\My Pictures
55 All Users\My Video
56 Resource Directory
59 USERPROFILE\Local Settings\Application Data\Microsoft\CD Burning

```

@UNIQUE - added optional second argument to specify the prefix to use for the filename (Windows will only use the first three characters).

@XMLPATH - removed and replaced with the following (more powerful) XML functions.

@XMLCLOSE - close an XML file previously opened by @XMLOPEN. The syntax is:

```
@XMLCLOSE[]
```

@XMLOPEN - open an XML file for use by @XMLXPath and/or @XMLNODES. The syntax is:

```
@XMLOPEN[filename]
```

@XMLNODES - return the number of nodes (children) for the specified path in an XML file. The syntax is:

```
@XMLOPEN[["filename"],path]
```

If you don't specify a filename (which **must** be in double quotes), @XMLNODES will use the XML file previously opened by @XMLOPEN.

@XMLXPath - XML XPath query. (See the XML XPath docs for details on XPath syntax.) The syntax is:

```
@XMLXPath[["filename"],path]
```

If you don't specify a filename (which **must** be in double quotes), @XMLXPath will use the XML file previously opened by @XMLOPEN.

3 Take Command

This section provides a general description of **Take Command** and **TCC** operation.

- ▶ [Starting Take Command](#) ¹⁷
- ▶ [Configuration Options](#) ⁴³
- ▶ [The Take Command Interface](#) ⁵⁶
- ▶ [Directory Navigation](#) ⁷¹
- ▶ [File Selection](#) ⁷⁷
- ▶ [Executable Extensions](#) ⁹¹
- ▶ [Using Internet URLs](#) ⁹²
- ▶ [Using FTP and HTTP Servers](#) ⁹³
- ▶ [Input / Output Redirection](#) ⁹⁸
- ▶ [OpenAFS](#) ¹⁰³
- ▶ [The TCC Command Line](#) ¹⁰³
- ▶ [Aliases & Batch Files](#) ¹²⁸
- ▶ [TCC Internal Commands](#) ¹⁴⁴
- ▶ [Variables & Functions](#) ³⁶⁵

3.1 Starting Take Command

You will typically start **Take Command** from a Windows shortcut, located:

- on the desktop, or
- in the Quick Launch bar, or
- in the **Programs** section of the **Start** menu (including its **Startup** subdirectory).

You may also start it from the **Start / Run** dialog.

The installation software will optionally create both a **Take Command** folder or group (in the **Programs** section of the **Start** menu) and a desktop object (shortcut) which starts **Take Command**. Usually these are sufficient, but if you prefer, you can create multiple desktop objects or items to start **Take Command** with different startup commands or options, or to run different applications in the tab windows.

Each item or icon represents a different **Take Command** window. You can set any necessary command line parameters for **Take Command** such as a program to run in a tab window, and the name and path for the [.INI file](#)^[26]. See [Take Command Startup Options](#)^[17] for more information on startup command line options.

When you configure a **Take Command** 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 **Take Command**. For example:

Command Line:	C:\Program Files\JPSoft\TCMD\TCMD.EXE
Working directory:	C:\

You do not need to use the Change Icon button, because **TCMD.EXE** already contains icons.

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 **Take Command** startup command line does not need to contain any information. When invoked with an empty command line, **Take Command** will configure itself from the [TCMD.INI file](#)^[26], and then display a prompt and wait for you to type a command. However, you may add information to the [startup command line](#)^[17] that will affect the way **Take Command** operates.

3.1.1 Take Command Startup Options

The **Take Command** command line includes the program name with drive and path, followed by any options. For example:

```
"c:\program files\jpsoft\tcmd9\tcmd.exe" @c:\tcmd\tcmd.ini
```

There are several **Take Command** startup options. The complete syntax for the **Take Command** startup command line is (all on one line):

```
d:\path\tcmd.exe [[/]@d:\path\inifile] [//directive=value...] [/D
d:\path] [/N] [/T [d:\path\]program]
```

(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.)

The command line must start with the full **Take Command** path and executable name (**TCMD.EXE**):


```
d:\path\tcmd.exe
```

The additional items below may be included on the command line:

```
@d:\path\inifile OR
/@d:\path\inifile
```

This option sets the path and name of the [.INI file](#)²⁶. You don't need this option if

- 1) your [.INI file](#)²⁶ is named **TCMD.INI**, and
- 2) it is in one of the following directories:
 - 2.1) the same directory as **Take Command**
 - 2.2) the %localappdata% directory

 This option is most useful if you want to start the program with a specific and unique [.INI file](#)

To start **Take Command** without any [.INI file](#)²⁶, you can create an empty file and specify it as your [.INI file](#)²⁶.

To get around a Windows limitation that causes the displayed command line of a shortcut to be truncated when a parameter begins with @, you can use the alternative syntax

```
/@d:\path\inifile
```

Take Command will skip the leading forward slash.

Options:

//directive=value This option tells **Take Command** to treat the text appearing between the // and the next space or tab as an initialization directive. The directive should be in the same format as a line in [TCMD.INI](#)²⁶, but may not contain spaces, tabs, or comments. 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](#)²⁶.

/D Start **Folders** and **List View** in the specified directory.

/N Don't load **TCMD.INI** (useful when trying to isolate configuration problems).

/T You can specify the program to start in the first tab with the /T option:

```
d:\path\tcmd.exe /t d:\path\program
```

If there is already a **Take Command** session running, /T creates a new tab in the existing **Take Command** rather than starting a new session.

/T must be the last option on the command line (otherwise **Take Command** can't tell if additional options belong to **Take Command** or the program to start in the tab).

If you have [Startup Tabs](#)⁴⁴ defined, **Take Command** will display them

following the tab created by **IT**.

3.1.2 TCC (4NT) Startup Options

The command line that starts **TCC** will typically include the program name with drive and path, followed by any options. For example:

```
"c:\program files\jpsoft\tcmd9\4nt.exe" @c:\4nt\4nt.ini
```

Although the startup command line is usually very simple, you can add several options. You can do this manually in the Windows **RUN** dialog, in a Windows shortcut file (**.LNK**), at the **TCC** prompt or in a batch file (with or without using the internal [START](#)^[33] command). Each of these methods will start a new instance of the selected command processor, which will run in a new window, except when **TCC** is started from **TCC** (either at the command prompt or within a batch file) without the [START](#)^[33] command.

When you use a [pipe](#)^[10] in a command, either at the command prompt or in a batch file, **TCC** starts another instance of itself, using the same command line parameters (except as required for the pipe).

The complete syntax for the **TCC** startup command line is (all on one line):

```
d:\path\tcc.exe [d:\path] [/]@d:\path\inifile[/directive=value...][A /H /L:  
/LA /LD /LF /LH /Q /S /T:bf /U /V /X ][/C | /K][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. Some options are available only in specific 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 command line must start with the path and name of the executable program file (**TCC.EXE**):

```
d:\path\tcc.exe
```

The additional items below may be included on the command line:

```
d:\path
```

If included, this second copy *d:\path* of **TCC** path must be identical to *d:\path* in the command line segment above. It sets the drive and directory where the program is stored, called the **COMSPEC** path. This option is included for compatibility with other character mode command processors, but is not needed in normal use. **TCC** can find its own directory without a **COMSPEC** path.

```
@d:\path\inifile OR  
/@d:\path\inifile
```

This option sets the path and name of the [.INI file](#)^[26]. You don't need this option if

- 1) your [.INI file](#)^[26] is named *TCMD.INI*, and
- 2) it is in one of the following directories:
 - 2.1) the same directory as **Take Command**
 - 2.2) the %localappdata% directory

^[26]

This option is most useful if you want to start the program with a specific and unique [.INI file](#)

To start **TCC** without any [.INI file](#)^[26], you can create an empty file and specify it as your [.INI file](#)^[26].

To get around a Windows limitation that causes the displayed command line of a shortcut to be truncated when a parameter begins with @, you can use the alternative syntax

```
/@d:\path\inifile
```

TCC will skip the leading slash.

```
//directive=value
```

This option tells **TCC** to treat the text appearing between the // and the next space or tab as a directive. The directive should be in the same format as a line in the [.INI file](#)^[26], but may not contain spaces, tabs, or comments. 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](#)^[26].

Directives on the command line override any corresponding directive in the [.INI file](#)^[26].

/A This option causes the output of internal commands to a pipe or redirected to a file to be in ASCII when **TCC** starts. This is the default value, and isn't necessary unless you want to override a [Unicode Output](#)^[47] configuration option.

/D Disable execution of AutoRun commands from Registry. If /D is not specified when **TCC** starts, it will look for and execute the following registry variables:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\AutoRun
```

and / or

```
HKEY_CURRENT_USER\Software\Microsoft\Command Processor\AutoRun
```

See also the [AutoRun](#)^[47] configuration option.

/H Start **TCC** in a hidden window. The window will not appear on the task bar, or in the Alt-tab list of applications.

/I Don't load the .INI file, execute TCSTART, or load plugins.

/II Don't load the .INI file.

/IP Don't load plugins.

/IS Don't execute TCSTART.

/L: Forces the use of local lists for aliases, functions directory history and command history, overriding any configuration options. This method allows you to use global lists as the default, but start a specific session with local aliases, functions and histories. See the topics [ALIAS](#)^[154], [FUNCTION](#)^[242], and [Local and Global History Lists](#)^[108] for more details. Note the required trailing colon (!)

/LA Forces the use of local aliases.

- /LD** Forces the use of a local directory history.
- /LF** Forces the use of local functions.
- /LH** Forces the use of a local command history list.
- /Q** This option has no effect. It is included only for compatibility with **CMD.EXE**.
- /S** This option tells **TCC** that you do not want it to set up a Ctrl-C / Ctrl-Break handler. It is included for compatibility with **CMD.EXE**.

Warning: 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 **TCC** 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 **CMD.EXE**. See the **CMD.EXE** color codes in [Colors, Color Names & Codes](#) ^[518].

In most cases you should set default colors with the corresponding [Output Colors](#) ^[49] configuration option. If you use both, the /T switch overrides the configuration options.

- /U** This option causes the output of internal commands to a pipe or redirected to a file to be in Unicode when **TCC** starts. The command :

[OPTION](#) ^[284] //UnicodeOutput=yes | no

may be used at any time to switch between Unicode and ASCII output.

- /V** Tells **TCC** to handle the **CMD.EXE** syntax **!varname!** as a delayed expansion of **%varname**. Since **CMD.EXE**, unlike **TCC**, doesn't support delayed expansion of variable references in the **%varname%** format, it introduced a special **!varname!** notation. Using **/V** simply tells **TCC** to handle that syntax as an alternative to **%varname%** or **%varname** or **%[varname]**.

- /X** This option forces **TCC** to alter the operation of the [MD](#) ^[271] ([MKDIR](#) ^[271]) 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](#) ^[271] ([MKDIR](#) ^[271]) commands. This option is included for compatibility with **CMD.EXE**, where it also enables other options. However, in **TCC** those options are already enabled by default.

/C command or
/K command or
command

Only one of these options may be used to specify for **TCC** what it must do after startup, and what it should do after completing **command**. **Command** will be executed after the automatic **TCC** startup program [TCSTART](#) ^[22], but before a prompt is displayed. **Command** may be any valid alias, internal or external command, or batch file, including parameters.

All other startup options must be placed before **command**, because **TCC** will treat characters after **command** as parameters for **command** and not as additional startup options.

If **command** is preceded by **/C**, **TCC** will execute **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 **command** (with neither **/C** nor **/K**) at the end of the startup command line. It is included only for compatibility with **CMD.EXE**.

Example 1

Assume that **C:** is the boot drive, and you execute the command line below:

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

The events below will take place in the order shown:

- 1 **Windows** starts **c:\TCC\TCC.exe**
- 2 **TCC** initializes from
 - 1st choice: **c:\TCC\TCMD.INI**
 - 2nd choice: **TCMD.INI** in the **%localappdata%** directory.
- 3.1 If the initialization file was found, **and** it contains the directive **TCStartPath=c:\start** **and** one of the files
 - c:\start\tcstart.btm**
 - c:\start\tcstart.bat**
 - c:\start\tcstart.cmd**
 - c:\start\tcstart.exe**
 - c:\start\tcstart.com**
 exists, that file is executed by **TCC**.
- 3.2 If no initialization file was found in Step 2, **or** the initialization file either does not contain the **TCStartPath** directive, or the value of the directive is **c:\TCC**, and a **TCSTART** program is found in directory **c:\TCC**, it is executed by **TCC**
- 4 **TCC** executes **c:\4NT\start.btm** (or, if not found, it displays an error message).
- 5 **TCC** displays the command prompt, unless an [EXIT](#)^[22] command was executed in **c:\4NT\start.btm**, terminating **TCC**.

Example 2

The command line below, when executed by **TCC**, **CMD.EXE**, the RUN dialog, or a shortcut, will start **TCC**, select local aliases, execute any **TCSTART** file you have created, execute the file **PROCESS.BTM**, and exit. No prompt will be displayed by this session:

```
c:\tcmd9\tcc.exe /la /c c:\tcmd9\process.btm
```

3.1.3 TCSTART and TCEXIT

TCC Startup Programs

Each time **TCC** starts, it looks for a program named [TCSTART](#)^[22]. **TCSTART** is normally a batch file (**.BAT**, **.BTM**, or **.CMD**), but it can be any executable file. If you specify a path in the [TCSTART / TCEXIT](#)^[47] configuration option, the file must be in the specified directory. If the configuration option is not used, the **TCSTART** program, if any, in the same directory as your command processor is executed. Use of **TCSTART** is optional, and **TCC** will not display an error message if it cannot find the program. If you do not want to use a startup program, set the [TCSTART / TCEXIT](#)^[47] path to a directory which does not have one, or leave it unspecified, and make sure that no matching executable file is in **TCC**'s directory.

TCSTART is a convenient place to change the color or content of the prompt for each session, [LOG](#) ^[269] the start of a session, or execute other special startup or configuration commands. It is also one way to set [aliases](#) ^[154], [functions](#) ^[242], and [environment](#) ^[365] variables. See the section below on Pipes etc. about changing directories via *TCSTART*.

With the exception of some [initialization switches](#) ^[19], the entire startup command line passed to **TCC** is available to *TCSTART* as [batch file parameters](#) ^[133] (%1, %2, etc.). For example, to pause if any parameters are passed, you could include this command in *TCSTART*:

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

Pipes, Transient Sessions / Processes, and TCSTART

When you set up the *TCSTART* program, remember that it is executed every time the command processor starts, including when running a [pipe](#) ^[101] or when a transient copy of **TCC** is started with the [/c startup option](#) ^[19]. 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 *TCSTART* program changes to a different directory, the output file will be written there — not in *C:\DATA*. This is because **TCC** starts a second copy (instance) of itself to run the commands on the right hand side of the pipe, and that new copy executes *TCSTART* before processing the commands from the pipe.

The same thing occurs if you use a transient session (one started with the **/C** option) to run an individual command, then exit — the session will execute in the directory set by *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:\tc\tcmd.exe /c list myfile.txt
Working Directory: c:\data
```

Normally this shortcut would [LIST](#) ^[264] the file *C:\DATA\MYFILE.TXT*. However, if *TCSTART* changes the default to a different directory, **TCC** will look for *MYFILE.TXT* there — not in *C:\DATA*.

Similarly, any changes to environment variables, aliases, or other settings in *TCSTART* will affect all copies of **TCC**, including those used for pipes and transient sessions.

You can work around these potential problems with the [IF](#) ^[253] or [IFF](#) ^[254] commands and the [_PIPE](#) ^[390] and [_TRANSIENT](#) ^[393] internal variables. For example, to skip all *TCSTART* processing when running in a pipe or in a transient session, you could use a command like this at the beginning of *TCSTART*:

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

TCC Termination Program

Whenever a **TCC** session ends, it looks for a program named [TCEXIT](#) ^[22]. *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 *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 **TCC** exits, or to [LOG](#) ^[269] the end of the session. You can use a termination program even if you have no startup program.

No parameters are passed to the termination program.

3.1.4 TCC Exit Codes

If you start **TCC** 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 exits. This code indicates whether or not the operation performed was successful, with **0** indicating success and a non-zero value indicating a failure or other numeric result.

TCC'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.

You can also use the [EXIT](#)^[227] **n** command to explicitly set the exit code. This overrides the rules above, and sets the return code to the parameter of your [EXIT](#)^[227] command.

3.2 Configuration Options

Take Command and **TCC** offer a wide range of configuration options, allowing you to customize their operation for your needs and preferences. The **Take Command** menu entry **Options / Configure Take Command** invokes the [Take Command Configuration Dialog](#)^[43], and the **TCC OPTION**^[284] command invokes the [TCC Configuration Dialog](#)^[46].

We also discuss many ways of configuring **TCC** 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](#)^[128] and the [ALIAS](#)^[154] and [FUNCTION](#)^[242] commands).
- With [executable extensions](#)^[91] you can associate data files with the applications you use to open them.
- With the [FILECOMPLETION](#)^[368] environment variable or the [Filename Completion Options](#)^[50] configuration option, you can customize filename completion to match the command you are working with.
- With the [COLORDIR](#)^[368] environment variable or the [Directory Colors](#)^[49] configuration option you can set the colors used by the [DIR](#)^[198] command.
- With [command line options](#)^[19] you can specify where **TCC** looks for its startup files and how it operates for a specific instance.

3.2.1 Initialization (.INI) Files

Part of the power of **Take Command** is its flexibility, in allowing you to alter its configuration to match your style of computing. **Take Command**'s configuration is controlled through a file of initialization information.

See [Locating the .INI files](#)^[25] below to find out how **Take Command** locates its **TCMD.INI** file.

Modifying the TCMD.INI File

You can create, add to, and modify the **TCMD.INI** file with the **Configure Take Command** selection on the [Options menu](#)^[61], or (for the **TCC**-specific sections) with the [configuration dialog](#)^[46], available via the [OPTION](#)^[284] command.

Most of the changes you make in the [configuration dialog](#)^[46] or with the [OPTION](#)^[284] command take

effect immediately. A few (e.g., those associated with the startup screen size) only take effect when you start a new **Take Command** 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 of the configuration options. The [Advanced directives](#)^[41] and the [Key Mapping directives](#)^[28] do not have corresponding fields in the configuration dialogs, and must be entered manually.

Take Command reads its *TCMD.INI* file (see [Locating the .INI file](#)^[25]) 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 **Take Command**.

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 TCMD.INI File

Some settings in the *.INI* file are initialized when you install **Take Command**; others are modified as you use and when you exit **Take Command**.

Locating the TCMD.INI File

1) When starting **Take Command** or a **Take Command Console** primary shell:

- ▶ If there is an **@d:\path\infile** option on the startup command line, **Take Command** will use the path and file name specified there.
- ▶ Otherwise, the default *.INI* file name in the table below is used, and the search starts in the directory where the **Take Command** program file is stored. If the *.INI* file is not found, **Take Command** will look in the %LOCALAPPDATA% directory.

If no *.INI* file is found, all options are set to their default values. A new *.INI* file will be created, using the default location and name, as explained above.

2) When starting a **TCC** secondary shell:

TCC retrieves the primary shell's *.INI* file data, processes the [Secondary] section of the original *.INI* file if necessary, and then processes any **@d:\path\infile** option on the secondary shell command line.

See [Command Line Options](#)^[19] for more details about the startup command line.

TCMD.INI File Sections

The *TCMD.INI* file has a number of sections. Each section is identified by the section name in square brackets on a line by itself. **Take Command** stores the user-defined options in **[TakeCommand]**; **TCC** stores its user-defined options in **[4NT]**.

The **[Primary]** and **[Secondary]** sections include directives that are used only in **TCC** 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.

See [Primary](#)^[533] and [Secondary Shells](#)^[534] for an explanation of those terms

3.2.1.1 Directives

This topic contains general information on **Take Command** and **TCC** initialization. For information on specific directives see the separate topic for each type of directive:

- ▶ [Key Mapping Directives](#)^[28]
- ▶ [Advanced Directives](#)^[41]

These topics list the directives, with a short 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 short 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.

Syntax for 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.

Regardless of how long a string value is, for example the list for the [ColorDir](#)^[368] 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](#)^[126].

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](#)^[46] 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 **Take Command** or **TCC** 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](#)^[284] command or see the [INIQuery](#)^[42] directive.

The [SETDOS](#)^[323] command can override several of the `.INI` file directives. For example, the cursor shape used by **TCC** can be adjusted either with the `CursorIns` and `CursorOver` directives or the [SETDOS](#)^[323] /S command. The correspondence between a [SETDOS](#)^[323] option and a `.INI` directive is noted under both the individual help topic for that directive and under that option in the [SETDOS](#)^[323] help topic.

Secondary shells automatically inherit the configuration settings currently in effect in the previous shell. If values have been changed by [SETDOS](#)^[323] or [OPTION](#)^[284] 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.

If you want to force secondary shells to start with a specific value for a particular directive, regardless of any changes made 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 or listed below the directive's description.
- ▶ **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 must be set to one of the vertical bar separated values listed between the braces. The default value is shown in all upper case letters in the directive description, but in your file any one of the choices can be entered, using any case. (Do not enter the vertical bar.) 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](#)^[518] for the format of color names.
- ▶ **Name = *Key*** : This directive takes a key specification. See [Keys and Keynames](#)^[515] for the format of key names.
- ▶ **Name = *Path*** : This directive takes a path specification, without 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 the = is unnecessary (e.g. [ClearKeymap](#)^[41]).

Evaluation of Directives

The directives are evaluated sequentially from top to bottom within each section processed. When a directive is processed more than once during startup, it replaces any previous value(s).

Most [key mapping](#)^[28] and [advanced](#)^[41] directives are cumulative and may appear several times when several concurrent values are desired, such as when assigning several different keystrokes to the same function.

3.2.1.1.1 Key Mapping Directives

These directives allow you to change the keys used for **TCC** command line editing and other internal functions. They cannot be entered via the [configuration dialogs](#)^[46]; you must enter them manually (see the [.INI file](#)^[26] 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](#)^[29]
- ▶ [Command Line Editing Keys](#)^[32]
- ▶ [Popup Window Keys](#)^[40]
- ▶ [LIST Keys](#)^[36]

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](#)^[25] 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](#)^[33] 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](#)^[515] before using the key mapping directives.

Key assignments are processed before looking for keystroke aliases. For example, if you assign Shift-F1 to [HELP](#)^[25] and also assign Shift-F1 to a key alias, the key alias will be ignored.

Assigning a new keystroke for a function does not deassign the default keystroke for the same function. If you want to deassign one of the default keys, use the [NormalKey](#)^[31], [NormalEditKey](#)^[35], [NormalPopupKey](#)^[40] or [NormalListKey](#)^[40] directive. You must also deassign default keys before you can assign them to a different usage.

Note: if you assign the same key to two different functions, the first assignment found in the list will be used.

3.2.1.1.1.1 General Input Keys

These directives apply to all input. They are in effect whenever **TCC** requests input from the keyboard, including during [command line editing](#)^[104] and the [DESCRIBE](#)^[195], [ESET](#)^[222], [INPUT](#)^[259], [LIST](#)^[264], and [SELECT](#)^[312] commands. The general input keys are:

Backspace ^[29]	Deletes the character to the left of the cursor
BeginLine ^[29]	Moves the cursor to the start of the line
Copy ^[29]	Copies highlighted text to the keyboard
Del ^[30]	Deletes the character at the cursor
DelToBeginning ^[30]	Deletes from the cursor to the start of the line
DelToEnd ^[30]	Deletes from the cursor to the end of the line
DelWordLeft ^[30]	Deletes the word to the left of the cursor
DelWordRight ^[30]	Deletes the word to the right of the cursor
Down ^[30]	Moves the cursor or scrolls the display down
EndLine ^[31]	Moves the cursor to the end of the line
EraseLine ^[31]	Deletes the entire line
ExecLine ^[31]	Executes or accepts a line
Ins ^[31]	Toggles insert / overstrike mode
Left ^[31]	Moves the cursor or scrolls the display left
NormalKey ^[31]	Deassigns a key
Paste ^[32]	Pastes line from clipboard
Right ^[32]	Moves the cursor or scrolls the display right
Up ^[32]	Moves the cursor or scrolls the display up
WordLeft ^[32]	Moves the cursor left one word
WordRight ^[32]	Moves the cursor right one word

Backspace = Key

Default: Backspace

Adds **Key** to the list of keys available during command line entry to delete the character to the left of the cursor.

See other [General Input Keys](#)^[29].

BeginLine = Key

Default: Home

Specifies **key** during command line entry as a request to move the cursor to the beginning of the line.

See other [General Input Keys](#)^[29].

Copy = Key

Default: Ctrl-Y

Adds **Key** to the list of keys available during command line entry to copy the highlighted text to the clipboard.

See other [General Input Keys](#)^[29].

Del = *Key*

Default: Del

Deletes the character at the cursor.

See other [General Input Keys](#) ^[29].

DelToBeginning = *Key*

Default: Ctrl-Home

Deletes from the cursor to the start of the line.

See other [General Input Keys](#) ^[29].

DelToEnd = *Key*

Default: Ctrl-End

Deletes from the cursor to the end of the line.

See other [General Input Keys](#) ^[29].

DelWordLeft = *Key*

Default: Ctrl-L

Deletes the word to the left of the cursor.

See other [General Input Keys](#) ^[29].

DelWordRight = *Key*

Default: Ctrl-R, Ctrl-Bksp

Deletes the word to the right of the cursor. See [ClearKeyMap](#) ^[41] if you need to remove the default mapping of Ctrl-Bksp to this function.

See other [General Input Keys](#) ^[29].

Down = *Key*

Default: Down

Scrolls the display down one line in [LIST](#) ^[264]; moves the cursor down one line in [SELECT](#) ^[312] and in the command line history, directory history, or [@SELECT](#) ^[458] window.

See other [General Input Keys](#) ^[29].

EndLine = *Key*

Default: End

Moves the cursor to the end of the line.

See other [General Input Keys](#) ²⁹.

EraseLine = *Key*

Default: Esc

Deletes the entire line.

See other [General Input Keys](#) ²⁹.

ExecLine = *Key*

Default: Enter

Executes or accepts a line.

See other [General Input Keys](#) ²⁹.

Ins = *Key*

Default: Ins

Toggles insert / overstrike mode during line editing.

See other [General Input Keys](#) ²⁹.

Left = *Key*

Default: left arrow, ←

Specifies a key, such the using the key will move the cursor left.

See other [General Input Keys](#) ²⁹.

NormalKey = *Key*

Deassigns a general input key in order to disable the usual meaning of the key 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](#) ³⁰ directive.

See other [General Input Keys](#) ^[29].

Paste = Key

Default: Ctrl-V

Paste the first line of the clipboard to the input line at the cursor position.

See other [General Input Keys](#) ^[29].

Right = Key

Default: Right

Moves the cursor right one character on the input line; scrolls the display right 8 columns in [LIST](#) ^[264]; scrolls the display right 4 columns in the command line history, directory history, or [@SELECT](#) ^[458] window.

See other [General Input Keys](#) ^[29].

Up = Key

Default: Up

Scrolls the display up one line in [LIST](#) ^[264]; moves the cursor up one line in [SELECT](#) ^[312] and in the command line history, directory history, or [@SELECT](#) ^[458] window.

See other [General Input Keys](#) ^[29].

WordLeft = Key

Default: Ctrl-Left

Moves the cursor left one word; scrolls the display left 40 columns in [LIST](#) ^[264].

See other [General Input Keys](#) ^[29].

WordRight = Key

Default: Ctrl-Right

Moves the cursor right one word; scrolls the display right 40 columns in [LIST](#) ^[264].

See other [General Input Keys](#) ^[29].

3.2.1.1.1.2 Command Line Editing Keys

These directives apply only to [TCC command line editing](#) ^[104]. They are only effective at the prompt. The command line editing keys are:

AddFile ^[33]	Keeps filename completion entry and adds another
AliasExpand ^[33]	Expands aliases on the command line

CommandEscape 33 ↵	Allows direct entry of a keystroke
DelHistory 33 ↵	Deletes a history list entry
DirWinOpen 34 ↵	Opens the directory history window
EndHistory 34 ↵	Displays the last entry in the history list
Help 34 ↵	Invokes this help system
HelpWord 34 ↵	Invokes help for the word at the cursor
HistWinOpen 34 ↵	Opens the command history window
LastHistory 34 ↵	Recall the last history entry
LFNToggle 35 ↵	Toggles between long and short filenames
LineToEnd 35 ↵	Copies a line to the end of the history, then executes it
NextFile 35 ↵	Gets the next matching filename
NextHistory 35 ↵	Recalls the next command from the history
NormalEditKey 35 ↵	Deassigns a command line editing key
PopFile 35 ↵	Opens the filename completion window
PrevFile 36 ↵	Gets the previous matching filename
PrevHistory 36 ↵	Recalls the previous command from the history
RepeatFile 36 ↵	Repeats previous match during filename completion
SaveHistory 36 ↵	Saves the command line without executing it
VariableExpand 36 ↵	Expand variables on the command line

AddFile = *Key*

Default: F10

Adds **Key** to the list of keys available during command line entry to keep the current filename completion entry and insert the next matching filename.

See other [Command Line Editing Keys](#) | 32 ↵.

AliasExpand = *Key*

Default: Ctrl-F

Adds **Key** to the list of keys available during command line entry to expand all aliases in the current command line without executing them.

See other [Command Line Editing Keys](#) | 32 ↵.

CommandEscape = *Key*

Default: Alt-255

Adds **Key** to the list of keys available during command line entry to signify that the immediately subsequent keystroke is to be used literally, and not for command line editing control.

See other [Command Line Editing Keys](#) | 32 ↵.

DelHistory = *Key*

Default: Ctrl-D

Deletes the displayed history list entry and displays the previous entry.

See other [Command Line Editing Keys](#)^[32].

DirWinOpen = *Key*

Default: Ctrl-PgUp, F6

Opens the directory history window while at the command line.

See other [Popup Window Keys](#)^[40].

EndHistory = *Key*

Default: Ctrl-E

Displays the last entry in the history list.

See other [Command Line Editing Keys](#)^[32].

Help = *Key*

Default: F1

Displays the [Help File](#)^[477] topic for the current command. See also: the [HELP](#)^[251] command and the [HelpWord](#)^[34] directive.

See other [Command Line Editing Keys](#)^[32].

HelpWord = *Key*

Default: Ctrl-F1

Invokes the [HELP](#)^[477] facility for the word at the cursor.

See other [Command Line Editing Keys](#)^[32].

HistWinOpen = *Key*

Default: PgUp

Brings up the history window while at the command line.

See other [Popup Window Keys](#)^[40].

LastHistory = *Key*

Default: F3

Returns the last history entry. (Mostly useless; it is for compatibility with **CMD.EXE**.)

See other [Command Line Editing Keys](#)^[32].

LFNToggle = *Key*

Default: Ctrl-A

Toggles filename completion between long filename and short filename modes on LFN drives.

See other [Command Line Editing Keys](#)^[32].

LineToEnd = *Key*

Default: Ctrl-Enter

Copies the current command line to the end of the history list, then executes it.

See other [Command Line Editing Keys](#)^[32].

NextFile = *Key*

Default: F9, Tab

Gets the next matching filename during filename completion. See [ClearKeyMap](#)^[41] if you need to remove the default mapping of Tab to this function.

See other [Command Line Editing Keys](#)^[32].

NextHistory = *Key*

Default: Down

Recalls the next command from the command history.

See other [Command Line Editing Keys](#)^[32].

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](#)^[31] for an example.

PopFile = *Key*

Default: F7, Ctrl-Tab

Opens the filename completion window. Note that **Take Command** uses Ctrl-Tab to select windows. See [ClearKeyMap](#)^[41] if you need to remove the default mapping of Ctrl-Tab to this function.

See other [Command Line Editing Keys](#)^[32].

PrevArgument = *Key*

Default: Ctrl-P

Recall the last argument from the previous command line.

See other [Command Line Editing Keys](#)^[32].

PrevFile = Key

Default: F8, Shift-Tab

Gets the previous matching filename. See [ClearKeyMap](#)^[41] if you need to remove the default mapping of Shift-Tab to this function.

See other [Command Line Editing Keys](#)^[32].

PrevHistory = Key

Default: Up

Recalls the previous command from the command history.

See other [Command Line Editing Keys](#)^[32].

RepeatFile = Key

Default: F12

Repeats the previous matching filename during filename completion.

See other [Command Line Editing Keys](#)^[32].

SaveHistory = Key

Default: Ctrl-K

Saves the command line in the command history list without executing it.

See other [Command Line Editing Keys](#)^[32].

VariableExpand = Key

Default: Ctrl-X

Expands variables at the command prompt.

See other [Command Line Editing Keys](#)^[32].

3.2.1.1.1.3 LIST Keys

These directives are effective only inside the **TCC LIST**^[264] command.

ListBack ^[37]	Return to the previous file
ListClipboard ^[37]	Copy the current filename to the clipboard
ListContinue ^[37]	Continue to the next file

ListExit ³⁷	Exits the current file
ListFind ³⁷	Prompts and searches for a string
ListFindReverse ³⁸	Prompts and searches backwards
ListFindRegex ³⁸	Prompt and search for a regular expression
ListFindRegexReverse ³⁸	Prompt and search backwards for a regular expression
ListHex ³⁸	Toggles between hexadecimal and character display modes
ListHighBit ³⁸	Toggles LIST's "strip high bit" option
ListInfo ³⁸	Displays information about the current file
ListNext ³⁸	Finds the next matching string
ListOpen ³⁹	Displays the "open file" dialog
ListPrevious ³⁹	Finds the previous matching string
ListPrint ³⁹	Prints the file on the default printer
ListRefresh ³⁹	Refresh the display
ListUnicode ³⁹	Toggles Unicode display mode
ListWrap ³⁹	Toggles LIST's wrap option
NormalListKey ⁴⁰	Deassigns a LIST key

ListBack = Key

Default: B

Returns to the previous file.

See other [LIST Keys](#) ³⁶.

ListClipboard = Key

Default: Ctrl-B

Copy the current [LIST](#) ²⁶⁴ filename to the clipboard

See other [LIST Keys](#) ³⁶.

ListContinue = Key

Default: C

Go to the next file.

See other [LIST Keys](#) ³⁶.

ListExit = Key

Default: Esc

Exits from the [LIST](#) ²⁶⁴ command.

See other [LIST Keys](#) ³⁶.

ListFind = Key

Default: F

Prompts and searches for a string.

See other [LIST Keys](#) ³⁶.

ListFindRegex = *Key*

Default: R

Perform a regular expression search in [LIST](#)^[264].

See other [LIST Keys](#)^[36].

ListBack = *Key*

Default: Ctrl-R

Perform a backwards regular expression search in [LIST](#)^[264].

See other [LIST Keys](#)^[36].

ListFindReverse = *Key*

Default: Ctrl-F

Prompts and searches backward for a string.

See other [LIST Keys](#)^[36].

ListHex = *Key*

Default: X

Toggles between hexadecimal and character display modes.

See other [LIST Keys](#)^[36].

ListHighBit = *Key*

Default: H

Toggles LIST's "strip high bit" option, which can aid in displaying files from certain word processors.

See other [LIST Keys](#)^[36].

ListInfo = *Key*

Default: I

Displays information about the current file.

See other [LIST Keys](#)^[36].

ListNext = *Key*

Default: N

Finds the next matching string.

See other [LIST Keys](#)^[36].

ListOpen = *Key*

Default: O

Opens the common Windows "open file" dialog to select a new file to [LIST](#)^[264].

See other [LIST Keys](#)^[36].

ListPrevious = *Key*

Default: Ctrl-B

Finds the previous matching string.

See other [LIST Keys](#)^[36].

ListPrint = *Key*

Default: P

Prints the file on the default printer.

See other [LIST Keys](#)^[36].

ListRefresh = *Key*

Default: F5

Refresh the [LIST](#)^[264] display. (Useful when viewing a growing log file.)

See other [LIST Keys](#)^[36].

ListUnicode = *Key*

Default: U

Toggles the [LIST](#)^[264] display mode between Unicode and ASCII.

See other [LIST Keys](#)^[36].

ListWrap = *Key*

Default: W

Toggles [LIST](#)^[264]'s wrap option on and off. The wrap option wraps text at the right margin.

See other [LIST Keys](#)^[36].

NormalListKey = Key

Deassigns a [LIST](#)^[264] 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](#)^[31] for an example.

See other [LIST Keys](#)^[36].

3.2.1.1.1.4 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](#)^[458] window.

NormalPopupKey ^[40]	Deassigns a popup window key
PopupWinDel ^[40]	Deletes a line from within the popup window
PopupWinEdit ^[40]	Moves a line from the popup window to the prompt
PopupWinEditWin ^[40]	Edit a line in the popup window
PopupWinExec ^[41]	Selects the current item and closes the popup window

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](#)^[31] for an example.

See other [Popup Window Keys](#)^[40].

PopupWinDel = Key

Default: Ctrl-D

Deletes a line from within the command history or directory history window.

See other [Popup Window Keys](#)^[40].

PopupWinEdit = Key

Default: Ctrl-Enter

Moves a line from the command history or directory history window to the prompt for editing.

See other [Popup Window Keys](#)^[40].

PopupWinEdit = Key

Default: Ctrl-E

Edit a line in the command history or directory history window.

See other [Popup Window Keys](#) ^[40].

PopupWinExec = *Key*

Default: Enter

Selects the current item and closes the window.

See other [Popup Window Keys](#) ^[40].

3.2.1.1.2 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](#) ^[46]; you must enter them manually (see the [.INI file](#) ^[26] for details).

ClearKeyMap ^[41]	Clear default key mappings
Debug ^[41]	Set debugging options
INIQuery ^[42]	Query for each line in the .INI file
LanguageDL ^[42]	Set localized language DLL
UpdateINI ^[43]	Enable / disable changes to the .INI file.

3.2.1.1.2.1 ClearKeyMap

ClearKeyMap

Clears all current [key mappings](#) ^[28]. ClearKeyMap is a special directive which has no value or = after it. Use ClearKeyMap with caution - it deletes all of the default definitions, and also any definitions in your .INI file directives that are processed before ClearKeyMap is processed. It is useful only if you want to make available most of the keys which have default assignments for other purposes, e.g., for keystroke aliases. ClearKeyMap should appear before any other key mapping directives. You may restore default mappings to keys you want to retain using the appropriate key assignment directives, e.g., [NextFile](#) ^[35] = **Tab**.

To clear the default mappings for just a few keys, use the [NormalEditKey](#) ^[35], [NormalKey](#) ^[31], [NormalListKey](#) ^[40], and/or [NormalPopupKey](#) ^[40] directives.

See other [Advanced Directives](#) ^[41].

3.2.1.1.2.2 Debug

Debug = *n*

Default: 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 **Take Command**, then wait for a keystroke.
- 2 (default) Include the product name with every error message displayed by **Take Command**. This may be useful if you are unsure of the origin of a particular error message.

See also: the [batch file debugger](#)^[166], a separate and unrelated facility for stepping through batch files.

See other [Advanced Directives](#)^[41].

3.2.1.1.2.3 INIQuery

INIQuery = yes | NO

If set to **Yes**, a prompt will be displayed before execution of each subsequent line in the current *.INI* file. This allows you to modify certain directives when you start **Take Command** 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 *.INI* file (*i.e.*, cancels the INIQuery = Yes setting)

The **TCC** 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**, you can enter a new value for the directive, but not a new directive name.

See other [Advanced Directives](#)^[41].

3.2.1.1.2.4 LanguageDLL

LanguageDLL = *filename*

Specifies the filename of the language DLL **Take Command** and **TCC** should use (English.dll, French.dll, or German.dll). **Take Command** normally uses the language dll that matches the default Windows user language, but you can override it with this directive.

(In most cases you shouldn't set this directive -- if you use a non-default language dll, you will get a mix of one language from **Take Command** and another from Windows for the system error messages.)

See other [Advanced Directives](#)^[41].

3.2.1.1.2.5 MSAAMenu

MSAAMenu = yes | NO

Take Command checks to see if a screen reader is installed, and if so it sets MSAAMenu=Yes. This is to avoid problems with the **Take Command** detachable menus and some screen readers.

3.2.1.1.2.6 UpdateINI

UpdateINI = YES | no

Enable or disable changes to the .INI file. (Useful for administrators who want to prevent users from changing their configuration.)

3.2.2 Take Command Configuration Dialog

This dialog, available from the [Options menu](#)^[61] in **Take Command**, contains four pages or "tabs" of options that let you change the way **Take Command** looks and works.

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 section of the [TCMD.INI file](#)^[26] and will be in effect each time you start **Take Command**.

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

[Windows](#)^[43]
[Tabs](#)^[44]
[Advanced](#)^[45]
[Registration](#)^[46]

3.2.2.1 Windows

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[46] topic before continuing.

Take Command Window

Single Instance restricts **Take Command** to a single instance. (If you try to start another copy of **Take Command**, the previous instance will be brought to the foreground.)

Always on Top makes **Take Command** a topmost window (i.e., it will remain on top of all other non-topmost windows).

Minimize to Tray minimizes **Take Command** to the system tray instead of the task bar.

Startup Mode - The **Normal**, **Max**, **Min**, and **Custom** buttons select the initial state for the **Take Command** windows.

Transparency:

Transparency sets the transparency level for the **Take Command** window. The range is from 20 (nearly invisible) to 255 (completely opaque). You can define both the active transparency (when **Take Command** is the foreground window, and the inactive transparency (when **Take Command**

is in the background).

Cursor:

The **Pointer** and **I-Beam** buttons let you select the type of cursor which **Take Command** will use in the tab windows.

3.2.2.2 Tabs

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#) topic before continuing.

Tabs:

Tab Size sets the maximum length of the tab label text (in characters).

Tab Icons displays the application's icon in the tab label.

Top puts the tab labels at the top of the window, and **Bottom** puts them at the bottom.

Close Tab Button defines the close button (X) on the tab labels. You must restart **Take Command** for this option to take effect.

Windows:

The **Font** button opens a standard Windows font dialog that lets you select the font, point size, and font style for **Take Command** tab windows.

Colors lets you select the default foreground and background colors for text in the **Take Command** tab windows. (These colors will be overwritten if the application in the tab window sets its own colors.)

Buffer Rows sets the number of rows in the console-mode screen buffer.

Left Alt Key sends the left Alt key to **Take Command**, so you can use it to invoke the **Take Command** menu or scroll the window. (The right Alt key will be passed to the application in the tab window.)

Right Alt Key sends the right Alt key to **Take Command**, so you can use it to invoke the **Take Command** menu or scroll the window. (The left Alt key will be passed to the application in the tab window.)

Left Ctrl Key sends the left Ctrl key to **Take Command**. (The right Ctrl key will be passed to the application in the tab window.)

Right Ctrl Key sends the right Ctrl key to **Take Command**. (The left Ctrl key will be passed to the application in the tab window.)

COMSPEC:

Set the program you want to start in new tabs. If no COMSPEC option is set, **Take Command** will run the program defined in the **COMSPEC** environment variable. If no **COMSPEC** variable is set, **Take Command** will run its default command processor (**TCC**).

Startup Tabs:

The **Startup Tabs** specify programs to run in each tab at startup. You can specify an optional title,

the command line, an optional startup directory, and optionally the user context where the tab should run.

3.2.2.3 Advanced

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[46] topic before continuing.

Advanced Options:

Linux-style Selection : Copy selected text automatically to the clipboard when the left mouse button is released.

MouseWheel Support for TCC LIST : If enabled, **Take Command** will pass the mouse wheel messages to **TCC** when it detects that **TCC** is executing its internal [LIST](#)^[264] command.

Notify Windows Shell on File or Directory Change : Notify the system shell when changing files or directories.

Show File Extensions : If enabled, **List View** will always show file extensions even if they are disabled in the Windows Shell folder properties.

Show Hidden Files : If enabled, the **Folders** and **List View** windows will show hidden files.

Single Click to Open Folders : Single click to open an item in the **Folders** window. (This will also close any other expanded items.) If this option is not enabled, **Take Command** will open folders in the **Folders** window with a double click and will not close other expanded items.

Update Environment on System Change : If enabled, **Take Command** will monitor the WM_SETTINGCHANGE message and if the environment is specified, update the environment from the User, Volatile, and System registry entries. The updates are done whenever **Take Command** displays a prompt (to prevent the environment from changing in the middle of a batch file).

Win64 File System Redirection : If disabled, overrides the default Win64 behavior of remapping `windows\system32` calls to `windows\SysWOW64`.

Zone ID : Set the NTFS Zone ID security when running executables downloaded from the Internet. (Note that CMD.EXE never checks for the Zone ID, so setting it may introduce a minor incompatibility.)

Start Dir : The initial directory to display in the **Folders / List View** windows.

Regular Expression Syntax:

Sets the regular expression syntax.

Descriptions:

Enable Descriptions : Set description display for the List View window.

NTFS Descriptions : If set, **Take Command** uses the Comments field in the NTFS SummaryInformation stream for each file to hold its description, instead of the `DESCRIPT.ION` file. The advantages are that the description will always remain with the file regardless of

what program copies, moves, or renames it. The disadvantage is that you cannot attach a description to directories.

Maximum Length : Set the description length limit. The allowable range is 20 to 511 characters.

Filename : Sets the file name in which to store file descriptions. The default file name is DESCRIPT.ION.

3.2.2.4 Registration

There are no separate **trial** and **registered** versions of our products. Without registration, a trial version is fully functional for 30 days of use. After 30 days, you will be limited in the number of commands you can run in a session.

The Register tab allows you to register **Take Command**. When you purchase a new or upgrade copy of **Take Command**, you will receive an email with your name and registration key. Enter the registration information exactly as you received it in the email (preferably by cutting & pasting). Remember to save your registration key in a safe place in case you need to reinstall. If you have lost your registration key, you can request a replacement by contacting JP Software at support@jpsoft.com, or at one of the addresses listed at the start of this file.

3.2.3 TCC (4NT) Configuration Dialog

This dialog, available via the [OPTION](#) ^[284] command, contains several "pages" or "tabs" of options that let you change the way **TCC** looks and works.

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]**) of the TCMD.INI file 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 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 [Initialization \(.INI\) Files](#) ^[24] and [Directives](#) ^[26].

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

[Startup](#) ^[47]
[Windows](#) ^[49]
[Command Line](#) ^[50]
[Advanced](#) ^[51]
[Internet](#) ^[53]
[Batch Debugger](#) ^[54]
[Updates](#) ^[55]
[Registration](#) ^[55]

3.2.3.1 Startup

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[46] topic before continuing.

Logging:

Command : Save internal and external commands (after alias and variable expansion) executed either from the command prompt or a batch file to the log file. See [LOG](#)^[269] for more details.

Errors : Save error messages to the log file. If you enter a file name in the File field, that file will be used for error logging. See [LOG](#)^[269] for more details.

History : Save each command executed from the command prompt exactly as it was entered (before aliases and variable expansion) to the log file. If you enter a file name in the File field, that file will be used for history logging. See [LOG](#)^[269] for more details.

TCSTART / TCEXIT:

You can set the path to your [TCSTART / TCEXIT](#)^[22] files if they aren't in the same directory as **Take Command**.

Local Lists:

Local Aliases instructs **TCC** to use a local or (if unchecked) global alias list.

Local Functions instructs **TCC** to use a local or (if unchecked) global function list.

Local History instructs **TCC** to use a local or (if unchecked) global command history list.

Local Directory History instructs **TCC** to use a local or (if unchecked) global directory history.

Scripting:

REXX : Enable the internal [REXX](#)^[142] support (Open Object REXX and Regina REXX).

Perl : Enable the internal [Perl](#)^[142] (ActiveState 5.8) support.

Ruby : Enable the internal [Ruby](#)^[143] (1.8 or 1.9) support.

Note : you must restart the **TCC** tab window for the **REXX**, **Perl**, and **Ruby** options to take effect.

Search for SFNs : If enabled, filename searches will search for both long filenames and short filenames. See [LFN File Searches](#)^[89] for details.

PathExt : Determines whether **TCC** will use the PATHEXT environment variable. If disabled, the PATHEXT variable is ignored. If enabled, the [PATHEXT](#)^[369] variable will be used to determine extensions to look for when searching the PATH for an executable file. For details, see the [PATHEXT](#)^[369] variable and the [PATH](#)^[286] command. If you enable **PathExt** and then fail to set the PATHEXT variable, path searches will fail as there will be no extensions for which to search!

Delete to Recycle Bin : If enabled, files deleted by the [DEL / ERASE](#)^[190] commands and by [RD /S](#)^[301] are placed in the Windows Recycle Bin. If disabled, the files are deleted without being placed in the Recycle Bin. [DEL](#)^[190]s and [RD](#)^[301]s /K and /R switches allow you to override this setting for individual commands. The [RecycleExclude](#)^[369] environment variable can be used to exclude

specific files.

Prompt on Wildcard Deletes : Enable the confirmation prompt from [DEL](#)^[190] /Q when doing a wildcard-only or directory deletion. Use caution if you disable this option, as this will allow DEL /Q to delete an entire directory without prompting for confirmation. See [DEL](#)^[190] for additional details.

Copy Prompt on Overwrite : If enabled [COPY](#)^[182] and [MOVE](#)^[274] will prompt before overwriting an existing file if the command is being performed at the command prompt. (This duplicates the behavior of the current version of CMD.EXE.)

Default Batch Echo : Set the default batch echo mode. If enabled, all batch file commands are echoed unless [ECHO](#)^[217] is explicitly set off in the batch file. If disabled, no batch file commands are echoed unless [ECHO](#)^[217] is explicitly set on. See also: [SETDOS /V](#)^[323].

Protect Redirected Output Files : If enabled, standard output [redirection](#)^[98] will be prevented from overwriting an existing file, and will require that the output file already exist for append redirection. (You can override this option by adding the exclamation point to the output redirection symbol; i.e. >!.) See also: [SETDOS](#)^[323] /N.

Wait for External Apps : Determines whether **TCC** waits for an external program started from the command line to complete before redisplaying the prompt. See [Waiting for Applications to Finish](#)^[123] for details on the effects of this option.

Update Titles : **Take Command** normally changes the window titles 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, disable this option.

UNIX/Linux-style Paths : Enables the forward slash as a path separator in the command name (the first item on the command line). Note that setting UnixPaths to Yes does not change the switch character, it simply allows you to put forward slashes in the command name. When this option is enabled, command switches beginning with a forward slash must be preceded by a space to avoid confusion (this is a good general practice).

Zone ID : Set the NTFS Zone ID security when running executables downloaded from the Internet. (Note that CMD.EXE never checks for the Zone ID, so setting it may introduce a minor incompatibility.)

Notify Windows Shell on File or Directory Change : Notify the system shell when changing files or directories. The shell notification is done by the [ASSOC](#)^[162], [COPY](#)^[182], [DEL](#)^[190], [MD](#)^[271], [MOVE](#)^[274], and [RD](#)^[301] commands. Note that setting this option could introduce a slight incompatibility with CMD.EXE, which doesn't notify the system shell about anything.

Update Environment on System Change : If enabled, **TCC** will monitor the WM_SETTINGCHANGE message and if the environment is specified, update the environment from the User, Volatile, and System registry entries. The updates are done whenever **TCC** displays a prompt (to prevent the environment from changing in the middle of a batch file).

MouseWheel Support in LIST : Set mouse wheel support in [LIST](#)^[264]. Disable this option if you experience incompatibilities with other applications.

AutoRun : If enabled when a **TCC** tabbed window starts, execute the AutoRun registry variables (HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\AutoRun and/or HKEY_CURRENT_USER\Software\Microsoft\Command Processor\AutoRun).

Win64 File System Redirection : If disabled, overrides the default Win64 behavior of remapping `windows\system32` calls to `windows\SysWOW64`.

Show Symbolic Links : Displays the symbolic link in [DIR](#)^[198] or [PDIR](#)^[288] (Windows Vista or later only).

Cancel Batch File on Ctrl-C : Cancel batch file processing without the usual prompt when you press Control-C.

Duplicate CMD.EXE bugs : Tells the **TCC** parser to duplicate bugs in CMD.EXE. The only bug currently replicated is in the [IF](#)^[253] command.

Unicode Output : The **TCC** output files (such as redirected output) will be written in Unicode format.

3.2.3.2 Windows

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[46] topic before continuing.

Command Prompt Window

(These options are only used when **TCC** is not running in a **Take Command** tab window.)

The **Normal**, **Max**, **Min**, and **Custom** buttons select the initial state for the **TCC** window.

The **X**, **Y**, **Width**, and **Height** fields set the initial size and position of the **TCC** window. They are ignored unless the **Custom** button is also selected.

Colors:

ANSI Colors : Enable ANSI X3.64 string processing of the output of Take Command internal commands. Note that ANSI X3.64 processing of the output of external applications is not supported. See the [ANSI X3.64 Commands Reference](#)^[516] for a list of the ANSI X3.64 commands supported by **TCC**.

Colors : Select foreground and background colors for input, output, and error messages.

Directory Colors : Sets the directory colors used by [DIR](#)^[198] and [SELECT](#)^[312]. The format is the same as that used for the [COLORDIR](#)^[368] environment variable. See [Color-Coded Directories](#)^[203] for a detailed explanation.

Pop-Up Windows:

Set the size and position of :

- The [command](#)^[107] and [directory](#)^[118] history search windows.
- The directory popup window used by [extended directory searches](#)^[73].

The position can be either in column/row coordinates, relative to the upper left corner of the console window, or in screen coordinates relative to the upper left corner of the **Take Command** window. If the **width** is < 150, the position is assumed to be in column/row format.

Editor:

Editor : The pathname of the editing program to run from [LIST](#)^[264]. (The default is NOTEPAD.EXE.)

3.2.3.3 Command Line

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[46] topic before continuing.

Command History:

Minimum Length : Set 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 8192. You can prevent any command line from being saved in the history by beginning it with an "at" sign ("@"). See also the [HISTORY](#)^[106] command.

Copy to end : This option controls what happens when you re-execute a line from the command history. If this option is set, the line is appended to the end of the history list. The original copy of the command is always retained at its original position in the list. If this option is set, it will override **Move to End**.

Move to end : If enabled, a recalled line will be moved to the end of the command history. The difference between this directive and **Copy to end** is that **Copy to end** copies each recalled line to the end of the history but leaves the original in place. **Move to end** copies the line to the end of history and removes the original line. This directive has no effect if **Copy to end** is set.

Wrap : If enabled, the command history "wraps" when you reach the top or bottom of the list (so the list appears "circular"). If this option is disabled, history recall will stop (and beep) at the beginning and end of the list rather than wrapping.

Duplicates : Controls duplicate entry placement in the [history list](#)^[106].

- Off** Always add new entries
- First** Add new entry only if it does not match any old entries
- Last** Add new entry unconditionally, and delete matching older entries

History File : Load the specified history list file before running [TCSTART](#)^[22], and save the command history to the file after running [TCEXIT](#)^[22].

Directory History File : Load the specified directory history list file before running [TCSTART](#)^[22], and save the directory history to the file after running [TCEXIT](#)^[22].

Filename Completion:

Complete hidden files : If enabled, hidden and system files and directories will be displayed by [filename completion](#)^[368]. [CDD /S](#)^[180] will also index hidden directories if this option is set.

Add '\' to Directories : If enabled, a \ (backslash) is automatically appended to directory names (or / to FTP directories) in [filename completion](#)^[368].

Search PATH : If enabled, the directories in the [PATH](#)^[286] variable are searched if a match isn't found in the current directory.

Options : Sets the files returned during [filename completion](#)^[368] for selected commands. The format is the same as that used for the [FILECOMPLETION](#)^[368] environment variable. See [Customizing Filename Completion](#)^[115] for a detailed explanation of selective filename completion.

Server Completion : Configures server name completion (see [Filename Completion](#)^[113] for

information on how to use server name completion). **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.

Editing:

Edit Mode : Starts the command line editor in either **Insert** or **Overstrike** mode. If you specify **Initial Overstrike** or **Initial Insert**, 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. See also: [SETDOS](#)^[323] /M.

Overstrike Cursor : 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 **Insert Cursor** and **Overstrike Cursor** change. If you set **Insert Cursor** and **Overstrike Cursor** to -1, the cursor shape won't be modified at all. If you set them to 0, the cursor will be invisible. See also: [SETDOS](#) /S^[323].

Insert Cursor : 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%. See also: **Overstrike Cursor** (above) and [SETDOS](#)^[323] /S.

Extended Directory Search:

Search Level : Configure extended directory searches. **0** disables extended searches. For complete details on the meaning of the other settings see [Extended Directory Searches](#)^[73].

Path : The path to *JPSTREE.IDX*, the file used for the [extended directory search](#)^[73] database.

History Buffer Sizes

Command History : Set the amount of memory allocated to the [command history](#)^[106] list (in characters). The allowable range of values is from 4000 to 131071. If you use a [global history list](#)^[108], this value is ignored in all sessions except that which first establishes the global list. (To change the size, you will need to close all of the **TCC** windows, and any [SHRALIAS](#)^[329] session.)

Directory History : Set the amount of memory allocated to the [directory history](#)^[118] list (in characters). The allowable range is 1,024 to 32767. If you use a [global directory history](#)^[108] list, this value is ignored in all sessions except that which first establishes the global list. (To change the size, you will need to close all of the **TCC** windows, and [SHRALIAS](#)^[329] session.)

3.2.3.4 Advanced

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[46] topic before continuing.

Special Characters:

Separator : The character used to separate multiple commands on the same line. The default is an ampersand (&). It can be dynamically modified by the /C option of the [SETDOS](#)^[323] command. You cannot use any of the [redirection](#)^[98] characters (| > <) or any of the white space characters (space, tab, comma, or equal sign).

Escape : The character used to suppress the normal meaning of the following character. The default is a caret (^). See [Escape Character](#)^[124] for a description of the special escape sequences. You cannot use any of the [redirection](#)^[98] characters (|, >, or <) or the white space characters (space, tab, comma, or equal sign) as the escape character. See also: [SETDOS](#)^[323]/E.

Parameter : The character used after a percent sign to specify all or all remaining command line parameters in a batch file or alias (e.g., %\$ or %n\$; see [Batch File Parameters](#)^[133] and [ALIAS](#)^[154]). The default is the dollar sign [\$]. See also: [SETDOS](#)^[323]/P.

Switch : The character used to delineate options. The default is a forward slash (/). Note that this will only affect internal commands, not any external applications, and will probably break all of your existing batch files and aliases!

Localization:

Time : The format of time displays in the output of the [DATE](#)^[189], [DIR](#)^[198], [SELECT](#)^[312], [TIME](#)^[347] and [TIMER](#)^[347] commands, and in [LOG](#)^[269] files. It has no effect on [%_TIME](#)^[393], [%@MAKETIME](#)^[449], the \$t and \$T options of [PROMPT](#)^[297], or date and time [ranges](#)^[80].

Country	Formats the time according to the country code set for your system.
am/pm	Displays the time in 12-hour format with a trailing "a" for AM or "p" for PM.
24-hour	Display the time in 24-hour time format.

Decimal : Sets the character used as the decimal separator for [@EVAL](#)^[420], numeric [IF](#)^[253] and [IFF](#)^[254] tests, version numbers, and other similar uses. The only valid settings are period [.] , comma [,], and **Auto** (the default). A setting of **Auto** tells **TCC** to use the decimal separator associated with your current country code. If you change the decimal character you will need to adjust the thousands character so that the two characters are different. See also: [SETDOS](#)^[323]/G.

Thousands : Sets the character used as the thousands separator for numeric output. The only valid settings are period [.] , comma [,], and **Auto** (the default). **Auto** tells **TCC** to use the thousands separator associated with your current country code. If you change the thousands character you will need to adjust the decimal character so that the two characters are different. See also: [SETDOS](#)^[323]/G.

Default Beep:

Length : The default [BEEP](#)^[173] length in system clock ticks (approximately 1/18 of a second per tick). Also the default length for "error" beeps (for example, if you press an illegal key).

Frequency : The default frequency (in Hz) for the [BEEP](#)^[173] command. This is also the frequency for "error" beeps (for example, if you press an illegal key). To disable all error beeps set this to 0. If you do, the [BEEP](#)^[173] command will still be operable, but will not produce sound unless you explicitly specify the frequency and duration.

Tabs:

Tabs : Sets the tab stops for [LIST](#)^[264] output. The allowable range is 1 to 32.

Descriptions:

Enable Descriptions : Set description handling for the file processing commands (COPY, DEL, MOVE, REN, etc.). If disabled, **TCC** will not update the [description](#)^[195] file when files are moved, copied, deleted or renamed. See also: [SETDOS](#)^[323]/D.

NTFS Descriptions : If set, **TCC** uses the Comments field in the NTFS SummaryInformation stream for each file to hold its description, instead of the *DESCRIPT.ION* file. The advantages are that the description will always remain with the file regardless of what program copies, moves, or renames it. The disadvantage is that you cannot attach a description to directories.

Maximum Length : Set the description length limit for [DESCRIBE](#)^[195]. The allowable range is 20 to 511 characters.

Filename : Sets the file name in which to store file descriptions. The default file name is *DESCRIPT.ION*. See also: [SETDOS](#)^[323] /D.

@EVAL Precision

Minimum : The minimum number of digits after the decimal point in values displayed by [@EVAL](#)^[420]. The allowable range is 0 to 1000. This directive will be ignored if **Minimum** is larger than **Maximum**. You can override this setting with the construct [@EVAL](#)[expression=n,n]. See also: [SETDOS](#)^[323] /F.

Maximum : The maximum number of digits after the decimal point in values displayed by [@EVAL](#)^[420]. You can override this setting with the construct [@EVAL](#)[expression=n,n]. The allowable range is 0 to 1000; if you use the "=n,n" syntax the maximum is 10,000. See also: [SETDOS](#)^[323] /F.

Regular Expression Syntax

Sets the type of regular expression syntax to use.

3.2.3.5 Internet

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[46] topic before continuing.

SMTP:

Server : The SMTP server name to use in [SENDMAIL](#)^[316] for outgoing mail. (If not set, SENDMAIL will attempt to get the address from the registry.)

Address : The email address to use in [SENDMAIL](#)^[316] for outgoing mail. (If not set, SENDMAIL will attempt to get the current user's email address from the registry.)

User : The email username (if your SMTP server requires it for authentication).

Password : The email password of the user (if your SMTP server requires it for authentication).

Port : The SMTP port number for use by [SENDMAIL](#)^[316]. (The default port number is 25).

Firewall:

Type : The type of firewall in use on your network.

Host : The server name of the firewall (if any) for FTP and HTTP access.

User : The user name if the firewall requires authentication.

Password : The password if the firewall requires authentication.

RSHELL / REXEC:

Host : The name of the local host or user-assigned IP interface through which connections are initiated or accepted (for [REXEC](#)^[308] and [RSHELL](#)^[309]).

User : The name of the user on the local machine (for [RSHELL](#)^[309]).

Port : The port number (or communication endpoint) in the local machine to bind to for [REXEC](#)^[308] and [RSHELL](#)^[309].

JABBER:

Server is the [JABBER](#)^[260] server to log into.

User : The default user name for logging onto a Jabber server and sending IM's via the [JABBER](#)^[260] command.

Password : The logon password for the default Jabber user.

Time Server:

Time Server : The URL for the internet time server for [TIME /S](#)^[347]. If no server is specified, TIME uses clock.psu.edu.

HTTP Proxy:

Server : The proxy server address to use for HTTP calls

User : The user name if Basic authentication is to be used for the HTTP proxy.

Password : The user password if Basic authentication is to be used for the HTTP proxy.

Port : The proxy port number to use for HTTP calls.

FTP:

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

FTP.CFG : Specify the location and name of the file containing the FTP user names and passwords, and optionally the directory format for non-standard FTP servers. The default is FTP.CFG in the **Take Command** installation directory. See [Using FTP/HTTP Servers](#)^[93] for details.

Timeouts:

Set the timeout (inactivity) period in seconds for FTP, TFTP, and HTTP operations.

3.2.3.6 Debugger

If you are not familiar with the purpose or use of the configuration dialogs, review the main [configuration dialogs](#)^[46] topic before continuing.

Transparency : Set the default transparency level of the [debugger](#)^[166] windows (40 to 255). 40 is almost invisible; 255 is opaque (no transparency). You can also set the debugger window transparency with the slider control on the bottom left of the debugger window.

Extensions : Set the batch file extensions supported by the [debugger](#)^[166]. If you don't specify an extension when creating a new file, the debugger will use the first extension in the list. Extensions must be separated by a semicolon. This option will take effect the next time a debugger window is opened.

Auto Bracket Match : Enable on bracket matching - (), [], and { }. This option will take effect the next time a [debugger](#)^[166] window is opened.

Syntax Coloring : Display key words (as defined in *BATCH.BCP*) in color.

Auto Indent : Automatically indent new lines to match the previous line. This option will take effect the next time a [debugger](#)^[166] window is opened.

Smart Indent : Increase the indent for new lines if the command on the previous line matches one in the smart indent list in *BATCH.BCP*. This option will take effect the next time a [debugger](#)^[166] window is opened.

Case Fixing : Automatically change the case of key words to match that in *BATCH.BCP*. (Syntax Coloring must also be enabled so that Case Fixing can recognize the keywords.) This option will take effect the next time a debugger window is opened.

Show Line Numbers : Display line numbers before each line. This option will take effect the next time a [debugger](#)^[166] window is opened.

Backup Files : Save a backup copy of the file when you do a Save. The file will be saved with a *.BAK* extension. The option will take effect the next time a [debugger](#)^[166] window is opened.

Font : Display a font dialog to select the font style and size to use in the [debugger](#)^[166] windows.

Foreground RGB : Display a color picker dialog to select the foreground text color for [debugger](#)^[166] windows.

Background RGB : Display a color picker dialog to select the background text color for [debugger](#)^[166] windows.

3.2.3.7 Updates

The **Check for Updates** button queries the JP Software web server to see if there is an updated version of **Take Command** available. If there is, the new version information will be displayed and you can choose to download and automatically update your existing version.













3.2.3.8 Registration

There are no separate **trial** and **registered** versions of our products. Without registration, a trial version is fully functional for 30 days of use. After 30 days, you will be limited in the number of commands you can run in a session.

The Register tab allows you to register **Take Command**. When you purchase a new or upgrade copy of **Take Command**, you will receive an email with your name and registration key. Enter the registration information exactly as you received it in the email (preferably by cutting & pasting). Remember to save your registration key in a safe place in case you need to reinstall. If you have lost your registration key, you can request a replacement by contacting JP Software at support@jpsoft.com, or at one of the addresses listed at the start of this file.

3.3 The Take Command Interface



The *Take Command* Window

- ▶ [The Take Command Window](#) 
- ▶ [Menus](#) 
- ▶ [Tool Bars](#) 
- ▶ [Folders](#) 
- ▶ [List View](#) 
- ▶ [Tab Windows](#) 
- ▶ [Status Bar](#) 
- ▶ [Keyboard Shortcuts](#) 
- ▶ [Context Menus](#) 
- ▶ [Using the Scrollback Buffer](#) 
- ▶ [Highlighting and Copying Text](#) 
- ▶ [Take Command Dialogs](#) 

Starting Applications

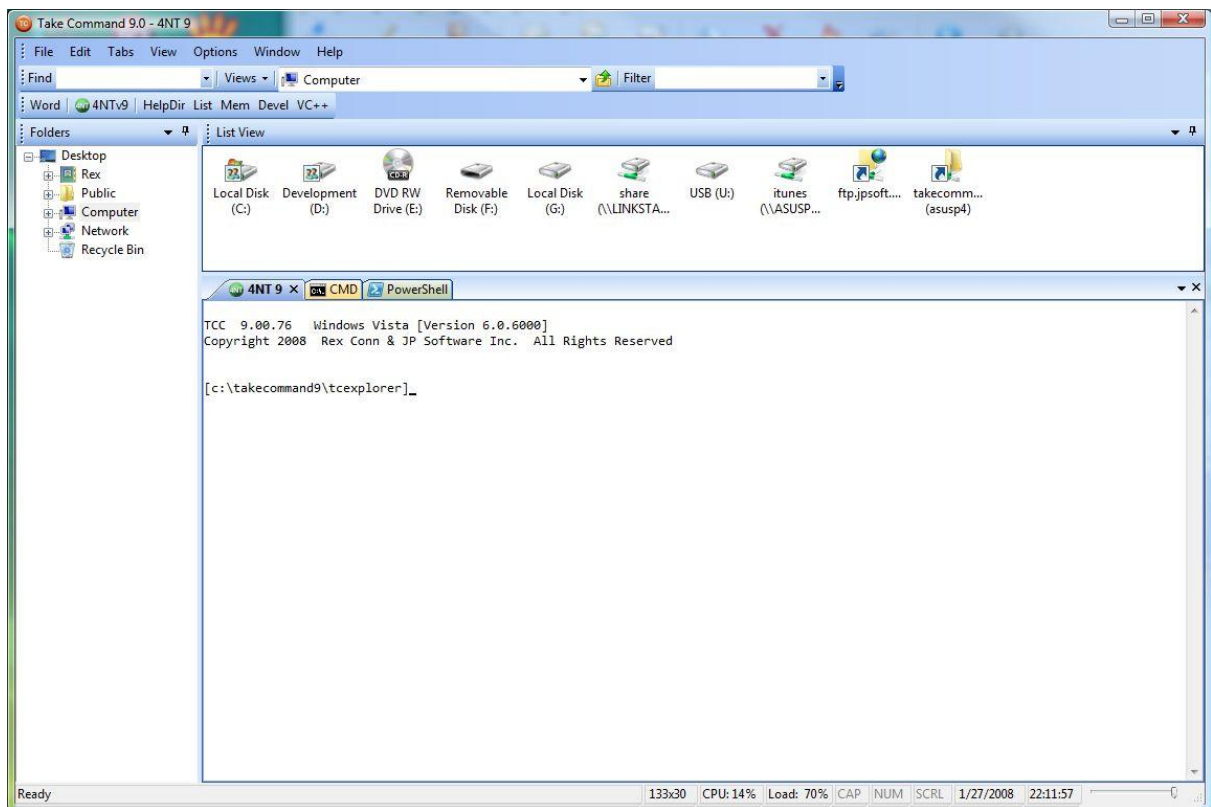
- ▶ [Starting Windows Applications](#) 

Take Command and the Windows Environment

- ▶ [Resizing the Take Command Window](#) 
- ▶ [Drag and Drop](#) 

3.3.1 The Take Command Window

The *Take Command* window has eight 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 also adjust the size of the **Take Command** window using standard window techniques, but see [Resizing the Take Command Window](#)^[68] for information about how **Take Command's** display changes when you do so.
2. See [Take Command Menus](#)^[58] for details about the **Menu Bar** and all of its menus.
3. The **Toolbar** 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](#)^[69] for instructions. You can customize and show or hide the Tool Bar with a choice on the [Options Menu](#)^[61].
4. The **Tab Toolbar** is an optional tool bar that you can use to execute internal or external commands, aliases, or batch files with the click of a mouse (or an accelerator key). You can define up to 50 tab toolbar buttons. To create buttons for the tab toolbar, select **Configure Tab Toolbar** from the [Options](#)^[61] menu. This selection displays the [tool bar dialog](#)^[69]. You can also configure the tab toolbar from **TCC** with the [TCTOOLBAR](#)^[343] command.
5. The **Folders** window shows a tree view of your desktop.
6. The **List View** window shows the contents of the item selected in the **Folders** window. You can display the **List View** window in a number of ways (Large Icons, Small Icons, List, and Details views).

7. The **Tab Windows** run the **Take Command Console**, or any other Windows console application (including CMD, PowerShell, or bash). You can use the scroll bars or the **Alt** cursor keys to view text that has scrolled through the window. You can also save the contents of a tab window and scrollbar buffer to a file, copy text from a tab Window to the clipboard, and copy text from the clipboard or from the tab window scrollbar buffer to the command line. See [Highlighting and Copying Text](#)^[67] for information about saving and retrieving text in the tab window and [The Command Line](#)^[103] for complete details about using the **Take Command** console command line.
8. Finally, the **Status Bar** at the bottom of the **Take Command** window displays information about your system:
- ▶ Tooltips for the menu selections
 - ▶ The tab window size (columns x rows)
 - ▶ The CPU usage (0 - 100%)
 - ▶ The memory load (0-100%)
 - ▶ The state of the Caps Lock key
 - ▶ The state of the Num Lock key
 - ▶ The state of the Scroll Lock key
 - ▶ The current date
 - ▶ The current time
 - ▶ A slider control to change the **Take Command** transparency

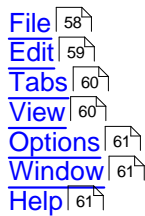
If you find the "I-Beam" cursor in the **Take Command** tab windows difficult to see, change it in the [Tabs](#)^[44] page of the configuration dialog to force the use of an arrow cursor in all parts of the window.

3.3.1.1 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 **File** menu). You can also select a menu by pressing **Alt** or **F10** and then moving the highlight with the cursor keys.

The **Take Command** menu bar is movable, dockable, and customizable. To move or dock the menu, move the mouse cursor over the grabber bar on the left side of the menu, press the left mouse button, and drag the menu to its new position. To customize the menu, click on **View / Toolbars and Menus / Customize**.

The menu bar entries allow you to select a variety of **Take Command** features:



3.3.1.1.1 File

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

New Tab

Opens the default **Take Command Console** command processor in a new tab window. (If you want to run a different application in a new tab, use the Run menu entry in the [Tabs](#)^[60] menu.)

Save to File...

Saves the contents of the current tab window's 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 current tab window's 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 current **Take Command** window (use this selection if the display appears incorrect).

Exit

Ends the current **Take Command** session.

3.3.1.1.2 Edit

The Edit menu allows you to copy text between the **Take Command** windows and the Windows clipboard. You can also access the clipboard in **TCC** with [redirection](#)^[98] to or from the CLIP: device, or with the [@CLIP](#)^[412] 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](#)^[67].

Copy

Copies selected text from the command line or scrollback buffer to the clipboard.

Paste

Copies text from the clipboard to the 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.

Copy + Paste

Copies the selected text from the scrollback buffer directly to the command line.

Copy + Paste + Run

Copies the selected text from the 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 scrollback buffer as selected text.

Find

Search the scrollback buffer for a string

Console Mouse


Send mouse moves and clicks to the console window. This option is specific to each tab, and is rarely necessary because few console apps use the mouse. You can also toggle this option with **Ctrl-M** in the tab window.

3.3.1.1.3 Tabs

New Tab

Opens the default command processor in a new tab window. **Take Command** defaults to the program specified in your COMSPEC environment variable; if that isn't found then **Take Command** will start **TCC.EXE**.

Run

Displays the [run dialog box](#)  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.

Attach Tabs

Displays a popup dialog of all of the console sessions that are **not** already displayed in a **Take Command** tab window, and allows you to select one or more to convert to a **Take Command** tab. You can select multiple sessions by clicking on individual entries and then on **OK**, or you can select a single session by double clicking on the entry.

Detach Tab

Disconnects the current tab window from **Take Command** and displays it on the desktop.

Rename Tab

Rename the current tab text (and the console title). If you set the title with this option, **Take Command** will not change the tab title if the application subsequently changes the console title. If you set the title to an empty string, the title will revert to that set by the console application.)

Close Tab

Closes the current tab window.

3.3.1.1.4 View

Toolbars and Menus

Customize the **Take Command** toolbars and menus.

Folder and List Views

Show or hide the **Folders** and **List View** windows. If you want to have more room for your tab windows, you can set the **Folders** and **List View** windows to AutoHide (i.e., they will be collapsed to a single tab label when not in use). If you don't want the view windows to appear at all, you can toggle them off with this option.

Status Bar

Show or hide the status bar.

Toggle Tabs Top/Bottom

Move the tab labels to the top or bottom of the tab window.

Large Icons

Display items in the List View using large icons.

List

Display items in the List View as a list.

Details

Display directory entries in the List View with a DIR-style display (Name, Size, Date/Time, Attributes, Type, and Description).

Set Filter

You can enter an expression to use to filter the directory entries displayed in the List View.

3.3.1.1.5 Options

Configure Take Command

Opens a [dialog](#)^[46] which you can use to change the **Take Command** configuration.

Configure Tab Toolbar

Opens a [dialog](#)^[69] in which you can define up to 50 buttons for the **Take Command** tab window [tool bar](#)^[62].

Skins

Opens the Skins and Themes dialog to select a custom skin. **Take Command** uses the standard Windows MSSTYLES format; thousands of skins are freely available on the Internet. You will need to copy your downloaded skin(s) to the **Styles** subdirectory in the **Take Command** installation directory.

Theme

Select a predefined theme for **Take Command**. This will change the color and appearance of the **Take Command** window and its components (such as the tab window labels).

3.3.1.1.6 Window

Toggle View Windows

Minimize or restore the Folder and List View windows.

New Horizontal Tab Group

Create a new tab bar by splitting the tabs window horizontally, and move the current tab window to the new tab bar.

New Vertical Tab Group

Create a new tab bar by splitting the tabs window vertically, and move the current tab window to the new tab bar.

This menu will also display the names of all of the tabs, and allow you to switch to a new tab by selecting it in the menu.

3.3.1.1.7 Help

See also: the [Help File](#)^[47] topic.

Contents

Displays the **Table of Contents** of the **Take Command** help file, 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, 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, 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 the JP Software 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.

Check for Updates

Query the JP Software web server to see if there is an updated version of *Take Command* available. If there is, the new version information will be displayed and you can choose to download and automatically update your existing version.

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.

Send Feedback to JP Software

Email us with suggestions or bug reports.

About Take Command

Displays the *Take Command* version, copyright, and license information.

3.3.1.2 Tool Bars

The *Take Command* window has two toolbars: the **Explorer toolbar** and the **Tab toolbar**.

You can hide or show the toolbars with the **View / Toolbars** menu entry, and you can detach and dock a toolbar by clicking on the left edge of the toolbar and dragging it to its new position. The new position of the toolbar will be saved when you close *Take Command* and restored when you restart.

Explorer Toolbar

The Explorer Toolbar has five default controls:

- The **Find** combo box searchrd for text in the current tab window's buffer.
- The **Views** button toggles the view type in the List View window between Large Icons, List, and Details). You can also select the view type from a drop-down list by clicking on the down arrow.
- The **Select Folder** combo box changes the current directory for the **Folders** and **List View** windows.
- The **Up** button changes to the parent directory in the **Folders** and **List View** windows.
- The **Filter** combo box allows you to enter [wildcards](#)^[77] or [regular expressions](#)^[80] to filter the directories and files displayed in the **List View** window. You can enter multiple wildcards by enclosing each argument in double quotes and separating them with a comma. For example:

```
"*.cmd", "*.txt", "*.doc"
```

To enter a regular expression in the **Filter** combo box, precede it with two colons. For example:

::/c

You can also set the filter from **TCC** with the [TCFILTER](#)^[343] command.

You can customize the Explorer Toolbar by clicking on the button on the right edge and selecting **Add or Remove Buttons**, or by right-clicking on the blank space to the right of the toolbar and selecting **Customize**.

Tab Toolbar

The Tab Toolbar is an optional tool bar that you can use to execute internal or external commands, aliases, or batch files with the click of a mouse (or an accelerator key). You can define up to 50 tab toolbar buttons.

To create buttons for the tab toolbar, select **Configure Tab Toolbar** from the [Options](#)^[61] menu. This selection displays the [tool bar dialog](#)^[69]. You can also configure the tab toolbar from **TCC** with the [TCTOOLBAR](#)^[343] command.

3.3.1.3 Folders

The **Folders** window displays a tree view of the folders on your system. You can optionally select the folder by entering the name in the combo box on the [Explorer Toolbar](#)^[62].

You can detach, move and dock the **Folders** window by moving the mouse cursor to the upper left corner, left-clicking on the grabber bar, and holding the left mouse button down while dragging the window to its new location.

You can AutoHide the **Folders** window by clicking on the "push-pin" in the upper right corner, or selecting it from the drop-down menu. When in AutoHide, the window will be minimized to a single tab, and will automatically expand again when you move the mouse cursor over the tab.

You can hide the **Folders** and **List View** windows completely by toggling the **View / Folder and List Views** menu entry.

3.3.1.4 List View

The **List View** window displays the contents of the directory selected in the **Folders** window. You can choose the format (Large Icons, List, or Details) with the Views button on the [Explorer Toolbar](#)^[62].

You can optionally filter the contents of the **List View** window by entering a regular expression in the Filter combo box on the [Explorer Toolbar](#)^[62]. You can also set the filter from **TCC** with the [TCFILTER](#)^[343] command.

You can detach, move and dock the **List View** window by moving the mouse cursor to the upper left corner, left-clicking on the grabber bar, and holding the left mouse button down while dragging the window to its new location.

You can AutoHide the **List View** window by clicking on the "push-pin" in the upper right corner, or selecting it from the drop-down menu. When in AutoHide, the window will be minimized to a single tab, and will automatically expand again when you move the mouse cursor over the tab.

You can hide the **Folders** and **List View** windows completely by toggling the **View / Folder and List Views** menu entry.

3.3.1.5 Tab Windows

The **Take Command** tab windows allow you to run multiple console applications in their own tab inside a single **Take Command** session.

Although you can run any character-mode application in a tab window, the most common usage will be command processors or utilities. **Take Command** includes its own console-mode command processor (**TCC**, formerly known as **4NT**), but you can run any other command processor, including **CMD**, **PowerShell**, **bash**, etc. in a tab window.

You can use the scroll bars or the **Alt** cursor keys to view text that has scrolled through the window. You can also save the contents of a tab window and scrollbar buffer to a file, copy text from a tab Window to the clipboard, and copy text from the clipboard or from the tab window scrollbar buffer to the command line. See [Highlighting and Copying Text](#)^[67] for information about saving and retrieving text in the tab window and [The Command Line](#)^[103] for complete details about using the **Take Command** console command line.

You cannot run a GUI application in a tab window.

3.3.1.6 Status Bar

The **Take Command** window has a Status Bar that displays tooltips when you move the cursor over menu entries.

The status bar also displays the following information:

- ▶ The tab window size (columns x rows)
- ▶ The CPU usage (0 - 100%)
- ▶ The memory load (0 - 100%)
- ▶ The state of the Caps Lock key
- ▶ The state of the Num Lock key
- ▶ The state of the Scroll Lock key
- ▶ The current date (yyyy-mm-dd)
- ▶ The current time

You can hide the status bar fields (except for Caps Lock, Num Lock, and Scroll Lock) in the [Windows](#)^[43] configuration dialog.

There is a slider in the right corner that allows you to dynamically change the transparency level of the **Take Command** window. (You can also set the transparency in the **Options / Configure Take Command / Windows** dialog.)

3.3.1.7 Keyboard Shortcuts

Take Command offers a number of keyboard shortcuts to change windows or select entries in the **Folders** and **List View** windows. If you are in a tab window, you need to set the **Left Alt Key** or **Right Alt Key** and **Left Ctrl Key** or **Right Ctrl Key** options in the **Take Command tab options**^[44] dialog. (Otherwise the keystroke will be sent to the console application rather than being interpreted by **Take Command**.)

All Windows

Alt-F4	Closes the Take Command window
Ctrl-Tab	Pops up a window allowing you to select the Folder , List View , or any of the tab windows.
Alt-F6	Cycle through the Folders View, List View, and the active tab windows.

Tab Window

Ctrl-F4	Closes the active tab window
Ctrl-T	Open a new tab
Ctrl-C	Copy the selected text to the clipboard
Ctrl-V	Paste the clipboard contents to the command line
Alt-Left	Change to the previous tab window
Alt-Right	Change to the next tab window
Alt-Up	Scroll tab window buffer up one line
Alt-Down	Scroll tab window buffer down one line
Alt-PgUp	Scroll tab window buffer up one page
Alt-PgDn	Scroll tab window buffer down one page

Folders and/or View Windows

Alt-Enter	Display the Properties dialog for the selected folder or file
Ctrl-Enter	Open the selected folder or file
Shift-Enter	Explore the selected folder
Del	Delete the selected file(s) in the List View window to the recycle bin
Shift-Del	Permanently the selected file(s) in the List View window
F2	Rename the selected file(s) in the List View window

3.3.1.8 Context Menus

Take Command displays a variety of context menus when you click on the right mouse button, depending on the location of the mouse cursor. If the mouse cursor is on the:

Menu Bar

You can customize the menu and toolbars.

Toolbar

You can customize the menu and toolbars.

Folders caption bar

You can detach, dock, AutoHide, or Hide the **Folders** window.

Folders view

Displays the Windows Shell context menu.

List View caption bar

You can detach, dock, AutoHide, or Hide the **List View** window.

List View

Displays the Windows Shell context menu.

Tab Bar

You can create a new tab or move an existing console window to a **Take Command** tab window.

Tab Labels

You can attach, detach, rename, and close tabs;

Tab Windows

You can copy selected text to the clipboard or paste the clipboard contents to the application in the tab window. (Paste works by sending the contents of the clipboard to the tab window as a series of keystrokes, so it will work with any application that accepts input.)

3.3.1.9 Running DOS apps

If you want to run 16-bit DOS programs in **Take Command** tab windows, you need to be aware of some limitations in the 16-bit support in Windows.

32-bit and 64-bit console applications can be run in any size tab window (subject to any internal limitations in the program itself). However, when Windows starts a 16-bit DOS program, it will always force the console buffer to resize itself to 80x25, 80x43, or 80x50 (whichever is nearest to the current console size). This has three implications:

- If Windows increases the console window size, **Take Command** has to increase its window height to match, and
- You will lose everything in your scrollbar buffer when you start the DOS program, and
- You will not be able to scroll through the output of the DOS program.

When the DOS program exits, Windows will resize the screen buffer to its original size (but it keeps the screen window at the same size). If **Take Command** increased its window height, it will not resize itself to its previous size.

If you size your **Take Command** tab windows to 25, 43, or 50 lines, **Take Command** will not need to resize itself when running DOS programs.

You can also start a DOS program in a tab window (with the [Run](#)^[69] dialog). **Take Command** will not need to resize its window in this case, as the console manager will be forced to size the console downwards rather than upwards. Or you can run the DOS program in a separate console window (with the [Run](#)^[69] dialog or the [TCC START](#)^[33] command).

Whenever possible, we recommend you retire your DOS programs and replace them with Windows console applications!

3.3.1.10 Using the Scrollback Buffer

Take Command retains the text displayed on its tab windows 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 **Alt-Up** and **Alt-Down** keys to scroll the display one line at a time from the keyboard, and the **Alt-PgUp** and **Alt-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.

You can set the size of the scrollback buffer on the [Windows tab](#)^[49] of the [configuration dialogs](#)^[46].

3.3.1.11 Highlighting & Copying Text

While you are working at the **Take Command Console** 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 a **Take Command** tab 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 a **Take Command** tab window to the clipboard, first use the mouse to highlight the text, then right click and select **Copy**, or use the **COPY** command on the [Edit menu](#)^[59]. You can optionally combine multiple selected lines into a single line before placing it in the keyboard by holding down the **Ctrl** key and selecting **Copy** (or **Copy+Paste** or **Copy+Paste+Run**) from the right-click context menu or the Edit menu. If you have an existing multiline selection in the clipboard, you can copy it to a single line (with the CR/LF's replaced by a space) in the **Take Command** window by holding down the **Ctrl** key and selecting **Paste**. (If you hold down the **Ctrl** key and the selection wraps around the last screen column, the lines will be appended without an intervening space.)

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 **Take Command** tab windows contain text, they are not document windows 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 a **Take Command** tab 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.

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](#)^[68] facility to paste a filename from another application onto the command line, and you can access the clipboard with [redirection](#)^[98] to or from the CLIP: device, or with the [@CLIP](#)^[412] variable function.

3.3.1.12 Resizing the Take Command Window

You can resize the **Take Command** window at any time 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](#)^[198] (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 [_ROWS](#)^[397] 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. The [_COLUMNS](#)^[384] internal variable can be used to determine the current virtual screen width.

If you 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 shrink 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. Use the horizontal scroll bar to make this data visible.

When the font is changed, **Take Command** will recalculate the virtual screen width.

3.3.1.13 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 an application (or the **Take Command** folder or list view windows), and drop it anywhere inside a **Take Command** tab window. The full name of the file will be pasted 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](#)^[67] for details.

3.3.1.14 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](#) ^[62]).

Take Command uses standard Windows dialogs for tasks like printing, selecting a font, or browsing files and directories. Since 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](#) ^[58] you can use to access that dialog.

Run Program Dialog ^[69]	(Apps ^[60])
Properties Dialog ^[46]	(Options ^[61])
	Startup Tab ^[47]
	Windows Tab ^[43]
	Command Line Tab ^[50]
	AdvancedTab ^[51]
	Internet Tab ^[53]
	(Options ^[61])
Tool Bar Dialog ^[69]	

3.3.1.14.1 Run Program

The Run Program dialog, started from the [Tabs menu](#) ^[60], 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 **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.

You can enter an optional startup directory in the **Directory** edit box.

You can specify an optional user name and password in whose context the program should be run. (Depending on your user privileges, you may not be able to run the program in a **Take Command** tab window.)

Start in a New Tab will start a console application in a new **Take Command** tab window. (You cannot start a GUI application in a tab window.)

The **Normal**, **Minimized**, and **Maximized** 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.

3.3.1.14.2 Tab Toolbar

This dialog (called from the **Take Command** Options menu) allows you to define or modify buttons on the tool bar. The tool bar dialog will display 50 buttons (1-25 in the first row, and 26-50 in the second row).

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.

A toolbar button can either open a new tab, send keystrokes to the current tab, or change the directory displayed in the Folder and List views.

You can define a toolbar button to display either an icon, a text label, or both. You must specify either the **Icon** or **Label** fields. If you enter both, **Take Command** will display the text to the right of the icon on the button.

In the **Icon** field, enter the filename for the icon (.ico) that you want to display on the button. If you specify an .exe filename, **Take Command** will use the first icon in that file. You can use the **Browse** button to find the file.

In the **Label** field, enter the text that you want to display on the button.

In the **Command** field, you can enter either the command to be started in a new window ("Start a new window"), or the keystrokes to be sent to the current tab ("Send to current tab"). You can use the **Browse** button to find a file to be entered at the beginning of the Command field.

If the tab is sending keystrokes to the current tab, the text is in the same format as the [KEYSTACK](#)^[262] command in the **Take Command Console**:

If you're starting a new window, the **Directory** field will set the startup directory for the command. If you are changing the Folders directory, the **Directory** field specifies the new directory. You can use the **Browse** button to find the directory.

Keystroke Interpretation

Characters entered within double quotes, e.g., "**abc**" will be sent to the active console application as is. The only items allowed outside the quotes are key names, the **!** and **/W** options, and a repeat count.

If **keyname** is a number, it is interpreted as an ASCII character value.

Repetition. To send **keyname** several times, follow it with a space, left bracket [, the repetition count, and a right bracket]. For example, the command below will send the Enter key 4 times:

```
enter [4]
```

The repeat count works only with an individual **keyname**. It cannot be used with quoted strings. You must have a blank space between the **keyname** and the repetition count.

If you exit by choosing the **OK** button, any changes you have made will be saved in TCMD.INI, and reloaded automatically the next time you start **Take Command**. If you use the **Cancel** button, your changes will be discarded.

The tool bar can also be configured with the [TCTOOLBAR](#)^[343] command.

3.3.1.14.3 Skins and Themes

Take Command supports Windows *.msstyle skins to change the visual style of the **Take Command** window. (These are available from a number of web sites.) Once you have downloaded skins to the **Styles** subdirectory in your **Take Command** installation directory, you will see them when you select "Local Skins" on the "Skins and Themes" tab. (If you have any system-wide skins installed in your **Windows** directory, they will appear when you select "System Skins".)

Changing skins will only affect the appearance of **Take Command**; other programs on your system will be unaffected.

If you want to restore the default configuration, click on the **Default** button in the **Default Skins** group.

The **Controls** tab shows you how the selected skin will look. (You cannot change anything in this tab; use the **Skins and Themes** tab to change the appearance.)

3.4 Directory Navigation

TCC remembers 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. You change the current working directory with the [CD](#)^[177] command. **TCC** supports 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 **TCC** directory navigation features. It also provides detailed documentation on the enhanced directory search features: [Extended Directory Searches](#)^[73] and [CDPATH](#)^[72].

The **TCC** directory navigation features are in three groups: features which help **TCC** 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. **TCC** supports this method, and also offers two significant enhancements:

- ▶ The [CDPATH](#)^[72] variable allows you to enter a specific list of directories to be searched, rather than searching a database. Use [CDPATH](#)^[72] 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.
- ▶ [Extended Directory Searches](#)^[73] allows **TCC** to search a database of all the directories on your system to find the one you want.

Changing Directories

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

- ▶ [Automatic directory changes](#)^[76] allow you to type a directory name at the prompt and switch to it automatically, without typing an explicit [CD](#)^[177] or similar command.
- ▶ The [CD](#)^[177] command can change directories on a single drive, and can return to the most recently used directory.
- ▶ The [CDD](#)^[178] command changes drive and directory at the same time, and can return to the most recently used drive and directory.
- ▶ The [PUSHD](#)^[299] command changes the drive and directory like [CDD](#)^[178], and records the previous directory in a directory "stack." You can view the stack with the [DIRS](#)^[209] command or the [@DIRSTACK](#)^[416] function, and return to the directory on the top of the stack with [POPD](#)^[294].

[CDD](#)^[178], [PUSHD](#)^[299], and [automatic directory changes](#)^[76] can also change to network drives and directories mapped to drive letters and to ones specified with UNC names (see [File Systems](#)^[49] for details).

Returning to a Previous Directory

CMD.EXE does 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. **TCC** supports three additional, simpler methods for returning to a previous directory:

- ▶ The [CD](#) -^[177] and [CDD](#) -^[178] 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](#)^[118] 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](#)^[294] command returns to the last directory saved by [PUSHD](#)^[299]. The directory stack holds 2048 characters, enough for 40 to 80 typical drive and directory entries.

3.4.1 CDPATH feature

When you change directories with an [automatic directory change](#)^[76] or the [CD](#)^[177], [CDD](#)^[178], or [PUSHD](#)^[299] command, **TCC** must find the directory you want to change to. If it cannot find an exact match of the directory path and name, **TCC** tries to find the directory you requested via the [CDPATH](#)^[72], then via an [Extended Directory Search](#)^[73].

Enabling both [CDPATH](#)^[72] and [Extended Directory Searches](#)^[73] can yield confusing results. If you prefer to explicitly specify where **TCC** should look for directories, use [CDPATH](#)^[72]. If you prefer to have **TCC** look at all of the directory names on your disk, use Extended Directory Searches.

[CDPATH](#)^[72] is an environment variable, and is similar to the [PATH](#)^[368] variable used to search for executable files: it contains an explicit list of directories to search when attempting to find a new directory. **TCC** appends the specified directory name to each directory in [CDPATH](#)^[72] 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](#)^[72] list.

[CDPATH](#)^[72] 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](#)^[72] provides a quick way to find commonly used subdirectories in an explicit list of locations. You can create [CDPATH](#)^[72] with the [SET](#)^[319] command. The format of [CDPATH](#)^[72] is similar to that of [PATH](#)^[368]: 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](#)^[72] with this command:

```
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](#)^[72] 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`.

TCC looks first in the current directory, and attempts to find the `C:\WP\LETTERS\JANUARY\UTIL` subdirectory. Then it looks at [CDPATH](#)^[72], and appends `UTIL` to each entry in the [CDPATH](#)^[72] 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](#)^[72]. You can use `...` to find "uncles", i.e., a directory one level up (a sibling of the parent directory), thus a subdirectory of the directory 2 levels up.

3.4.2 Extended Directory Searches

When you change directories with an [automatic directory change](#)^[76], [CD](#)^[177], [CDD](#)^[178], or [PUSHD](#)^[299] command, **TCC** must find the directory you want to change to. To do so, it first 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, **TCC** changes to that directory, and does no further searching.

This 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 **TCC** to find the directory you want, even if you only enter a small part of its name.

When the first search method fails, **TCC** tries to find the directory you requested via the [CDPATH](#)^[72] variable, then via an Extended Directory Search. This section covers only Extended Directory Searches, which are more flexible and more commonly used than [CDPATH](#)^[72].

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](#)^[72] 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 [Windows tab](#)^[49] of the [configuration dialogs](#)^[46]. You can also change the keys used in the popup window with [key mapping directives](#)^[28].

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](#)^[178] 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:. Because of security restrictions in Windows Vista, the the default directory in Vista is defined as the value of the environment variable APPDATA (predefined by Windows). If you are running Vista and don't have APPDATA in your environment, the default directory will be the directory where **TCC** is installed. You can specify a different location for this file on the [Command Line tab](#) of the [configuration dialogs](#).

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 [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 Search Level on the [Command Line tab](#) of the [configuration dialogs](#).

- If Search Level = 0, extended searches are disabled, the *JPSTREE.IDX* 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 Search Level = 1 and an extended search is required, **TCC** will search the *JPSTREE.IDX* database for directory names which exactly match the name you specified.
- If Search Level = 2 and an extended search is required, **TCC** will search the database for exact matches first, just as when Search Level = 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 Search Level = 3 and an extended search is required, **TCC** will search the database for exact matches first, just as when Search Level = 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.

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

An extended directory search is not used if you specify a full directory path (one beginning with a

backslash \, or a drive letter and a backslash). If you use a name which begins with a drive letter (e.g. *C:MYDIR*), the extended search will examine only directories on that drive.

Forcing an Extended Search with Wildcards

Normally you type a specific directory name for **TCC** to locate, and the search proceeds as described in the preceding sections. However, you can also force **TCC** to perform an extended directory search by using [wildcard characters](#)^[77] in the directory name. If you use a wildcard, an extended search will occur whether or not extended searches have been enabled.

When **TCC** is changing directories and it finds *wildcards* in the directory name, it skips the explicit search and [CDPATH](#)^[72] 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](#)^[178] /s.

Internally, extended directory searches use wildcards to scan the directory database. If Search Level is set to 2, an extended search looks for the name you typed followed by an asterisk (i.e. *DIRNAME**). If Search Level 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 Search Level is set to 3, **TCC** will search the directory database for **ABC?DEF**.

Disabling Extended Searches in Batch Files

When writing batch files you may want to use the [CD](#)^[177] or [CDD](#)^[178] 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](#)^[177] or [CDD](#)^[178]. When this option is used and a directory does not exist below the current directory or on the [CDPATH](#)^[72], 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](#)^[299]. To perform the same function when using [PUSHD](#)^[299], save the current directory with [PUSHD](#)^[299] (without parameters) and then use [CDD](#)^[178] /N to change directories, for example:

```
pushd
cdd /n testdir
```


3.4.3 Automatic Directory Changes

Automatic directory changes are part of the comprehensive directory navigation features built into **TCC**. For a summary of these features, and more information on [Extended Directory Searches](#)^[73] and [CDPATH](#)^[72], see [Directory Navigation](#)^[71].

Automatic directory changes let you change directories quickly from the command prompt, without entering an explicit [CD](#)^[177] or [CDD](#)^[178] command. Simply type the name of the directory you want to change to at the prompt, with a terminating backslash (\) (either entered manually, or automatically via the [Add \ to Directories](#)^[50] configuration option). For example:

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

This can make directory changes very simple when it's combined with [Extended Directory Searches](#)^[73] or [CDPATH](#)^[72]. If you have enabled either of those features, **TCC** will use them in searching for a directory with an automatic directory change.

For example, suppose [Extended Directory Searches](#)^[73] 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]
```

This depends 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 matching **WIN** to choose from.

The text before the backslash can include a drive letter, a full path, a partial path, or a [UNC name](#)^[536] (see [File Systems](#)^[49] for details on UNC names). Commands like "...\" can be used to move up the directory tree quickly (see [Extended Parent Directory Names](#)^[118]).

If you enter a directory name without the trailing backslash, the parser will change to that directory if no internal or external command of that name is found (and before the [UNKNOWN_CMD](#)^[16] alias is executed.)

All directory changes, including automatic ones, save the current directory so it can be recalled with a [CDD](#)^[178] - or [CD](#)^[177] - 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.

3.4.4 Directory Aliases

Directory Aliases are a shorthand way of specifying pathnames. For example, if you define an alias:

```
alias pf:=c:\program files
```

You can then reference the files in **c:\program files\jpsoft** by entering **pf:\jpsoft**. Directory aliases work in places that accept filenames and directory names (internal command arguments or the first argument in a command line), including filename completion. You cannot use them in arguments to external applications, as **TCC** has no way of knowing what is a valid argument for external applications.

Directory aliases support alias arguments, but they do not support environment variable expansion.

3.5 File Selection

Most internal commands (like [COPY](#)^[182], [DIR](#)^[198], etc.) work on a file or a group of files. 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 **TCC** commands only, and generally cannot 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](#)^[77]
- ▶ [Ranges](#)^[80]
- ▶ [Attribute Switches](#)^[86]
- ▶ [Multiple Filenames](#)^[87]
- ▶ [Include Lists](#)^[88]
- ▶ [Extended Parent Directory Names](#)^[118]
- ▶ [Directory Aliases](#)^[76]
- ▶ [LFN File Searches](#)^[89]
- ▶ [@File Lists](#)^[90]
- ▶ [Command Switches for File Selection](#)^[91]

3.5.1 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.

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](#)^[85] (for internal commands), or [EXCEPT](#)^[225] (for external commands).

Most internal commands accept filenames with wildcards anywhere that a full filename can be used. There are two wildcard characters, the [asterisk](#)^[77] ***** and the [question mark](#)^[78] **?**. Additionally, you can specify a [set of characters](#)^[78]. Note the issues about [matching short file names](#)^[79].

WARNING: When you use a wildcard search for files to process in a command like [FOR](#)^[234] or [DO](#)^[210], and you create new filenames (whether by renaming existing files or by creating new files), the new filenames may match your selection wildcard, and cause you to process them again.

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:

```
dir *
```

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

```
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:

```
dir st*.d*
```

TCC 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*, but it will ignore *CLAIM.TXT*:

```
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*, *LATTER.DAT*, and *LITTER.DU*:

```
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 **TCC**.

"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:

```
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*.

Specific character set

In some cases, the *?* wildcard may be too general. **TCC** also allows you to 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:

```
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:

```
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:

```
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*:

```
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:

```
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,

```
dir letter[ ].doc
```

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

```
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](#)¹⁹⁸/I selects files to be displayed 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:

```
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](#)⁸⁸. For example, if you have a file on an LFN drive named *C:\DATA\LETTER1;V2* and you enter this command:

```
del \data\letter1;v2
```

you will not get the results you expect. Instead of deleting the named file, **TCC** will attempt to delete *LETTER1* and then *V2*, because the semicolon indicates an [include list](#)⁸⁸. 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:

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

Matching short file names

If the [Search for SFNs](#)^[47] configuration option is set, wildcard searches accept a match on either the LFN or the SFN to match the behavior of CMD.EXE. This may cause some files to be found because of SFN match only. In most situations this is not actually desirable, and can be avoided by disabling the option (the default).

Note: The wildcard expansion process will attempt to allow both CMD.EXE-style "extension" matching (only one extension, at the end of the word) and the advanced **TCC** filename matching (allowing things like *.*.abc) when an asterisk is encountered in the destination of a [COPY](#)^[182], [MOVE](#)^[274] or [REN/RENAME](#)^[305] command.

Regular Expressions

You can also use regular expressions for file name tests. (The type of regular expressions to use is specified by the [Regular Expressions Syntax](#)^[51] option.)

The syntax is:

```
::regex
```

For example:

```
dir ::ca[td]
```

Note that using regular expressions will slow your directory searches -- since Windows doesn't support them, the parser has to convert the filename to *, retrieve all filenames, and then match them to the expression.

If you have any special characters (whitespace, redirection characters, escape characters, etc.) in your regular expression, you will need to enclose it in double quotes. For example:

```
dir ::"^w{1,8}\.btm$"
```

For more information on the syntax, see [Regular Expression Syntax](#)^[496].

3.5.2 Ranges

Most internal commands which accept wild cards also allow size, date, time, exclusion, and description ranges to further define the files that you wish to work with. **TCC** will examine each file's properties to determine whether or not the file meets the range criteria that you have specified.

A size, date, time, or exclusion range specification begins with the switch character /, followed by a left square bracket [and a character that specifies the range type: **s** for size range, **d** for date range, **t** for time range, or **!** for exclusion 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 / [**s**100 , 200].

A description range begins with **/!**. See [Description Ranges](#)^[86] for the full syntax.

General Rules

You can reverse the range test by preceding the range argument with the **!** character. For example, to select files that are less than 100 bytes or more than 1000 bytes:

```
/![s100,1000]
```

If you combine different types of ranges, a file must satisfy all range specifications to be included. For example,

```
/[d2006-2-8,2008-2-9] /[s1024,2048]
```

means files last modified between February 8, 2006 and February 9, 2008, which are also between 1,024 and 2,048 bytes long.

You may not repeat the same range type in a command.

When you use range specifications in a command, they 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, 2007, you could use this command:

```
dir /[d2007-10-1,+0] *.c
```

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

```
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:

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

It can be tedious 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 another drive, you might define an alias like this:

```
alias workback=`select /[d-2] copy (*.dat) e:\datfiles\`
```

For more complex requirements, you may want to use internal variables (e.g. [_DATE](#)^[385] or [_TIME](#)^[393]) and built-in variable functions (e.g. [@DATE](#)^[414], [@TIME](#)^[463], [@MAKEDATE](#)^[449], [@MAKETIME](#)^[449], [@FILEDATE](#)^[428], [@FILETIME](#)^[432], or [@EVAL](#)^[420]). 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.

See the individual types for details on specifying ranges:

- ▶ [Size Ranges](#)^[82]
- ▶ [Date Ranges](#)^[82]
- ▶ [Time Ranges](#)^[84]
- ▶ [Exclusion Ranges](#)^[85]
- ▶ [Description Ranges](#)^[86]

Ranges can be used with many commands, including [ATTRIB](#)^[163], [COPY](#)^[182], [DEL](#)^[190], [DESCRIBE](#)^[195], [DIR](#)^[198], [DO](#)^[210], [EXCEPT](#)^[225], [FFIND](#)^[227], [FOR](#)^[234], [LIST](#)^[264], [MOVE](#)^[274], [RD](#)^[301], [REN](#)^[305], [SELECT](#)^[312], and [TYPE](#)^[354].

Ranges cannot be used with filename completion or in filename parameters for variable functions, except as described under the individual functions.

Do not use ranges with [@file](#) lists. See [@file lists](#)^[90] for details.

Date, Time, and Size 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 or 20,000 bytes long, as well as all sizes in between; a date range that selects files last modified between 2007-10-27 and 2007-10-30 will include files modified on each of those dates, and on the two days in between.

If you reverse range start and end values **TCC** 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]**.

3.5.2.1 Size Ranges

Size ranges select files whose size is between the inclusive limits specified. The second parameter of a size range is optional. If you use a single parameter, you will select all files of the specified size or larger. You can also precede the second parameter 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.

You can exclude a size range by preceding the range with the ! character.

When you use a size range in a command it should immediately follow the command name. See [General Rules for Using Ranges](#) ^[80] for additional details.

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

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
/[s1K,5K]	less than 1K or greater than 5K

3.5.2.2 Date Ranges

Date ranges select files dated at any time of day between the inclusive limits specified. For example, **/[d12-1-07,12-5-07]** selects files that were last modified on or after December 1, 2007, but not modified after December 5, 2007.

When you use a date range in a command, only other range specifications may be between the command name and the date range. See [General Rules for Using Ranges](#) ^[80] for additional details.

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, **/[d2006-12-31,2008-1-1]** selects files modified between December 31, 2006 and January 1, 2008.

If either parameter begins with a four digit year (which must 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.

The default time for the first date is the beginning of that day, and for the second date it is the end of that day. This is true even if the dates are in descending order, i.e., the first date is later than the second one. 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 **/[d2007-7-15@8:00a,2007-7-35@6:00p]** selects files that were modified at any time between 8:00:00 am on July 1, 2007 and 6:00:00 pm on July 3, 2007. 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 parameter in a date range, **TCC** substitutes the current date and time. For example, **/[d2007-10-1]** selects files dated between October 1, 2007 and the instant of command execution.

Instead of an explicit date, you may 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:

Specification	Selects Files
/[d2008-1-27,+3]	modified between January 27, 2008 and January 30, 2008
/[d2008-1-27,-3]	modified between January 24, 2008 and January 27, 2008
/[d-0]	modified today (from today minus zero days, to today)
/[d-1]	modified yesterday or today (from today minus one day, to today)
/[d-1,+0]	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:

```
dir /[d-3,1/1/80]
```

This reversed date range (with the later date given first) will be handled correctly by **TCC**. 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, **/[d2007-12-85@12:00,+2@12:00]** selects files that were last modified between noon on December 8 and noon on December 10, 2007. 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 specified.

You can exclude a date range by preceding the range with the **!** character.

Notes:

- If the second date is the termination date, and it includes an explicit termination time, it is considered an exact value. For example, in the last example the termination time was 6PM. Files with a timestamp of 6:00:01 PM or later are not included in the date range. This is different from the behavior of [time ranges](#)⁸⁴.

- If you include seconds in the times you specify, they will be silently ignored (no error or warning).
- If the first date is later than the second, any time of day modifiers for the first date are silently ignored.

Date types and selection

Windows file systems keep track of three dates for a file: when it was created, when it was last modified (written), and when it was last accessed. You 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, 2008 and February 7, 2008, inclusive, you would use **/[dc2008-02-1,2008-2-7]**. If you don't specify which date and time to use, **TCC** will use the date the file was last modified (written).

NOTE: On FAT32 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 parameter:	Beginning of the day (00:00:00)
Time of second parameter:	End of the day (23:59:59)
Missing second parameter:	Current date and time
Date type	Modification (write)

3.5.2.3 Time Ranges

Time ranges select files timed at any time between the two specified times of day. For example, to select files modified at or between noon and 2:00 PM on any day, 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.

When you use a time range in a command it should immediately follow the command name. See [General Rules for Using Ranges](#) for additional details.

If you omit the second parameter 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 parameters in a time range. The offset values are interpreted as minutes. Some examples:

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

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

You can exclude a time range by preceding the range with the **!** character.

Time types and selection

Windows keeps track of three times for a file: when it was created, when it was last modified (written), and when it was last accessed. You can specify which time is used in a time 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 time to use, **TCC** will use the time 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.

When you use a time range in a command it should immediately follow the command name. See [General Rules for Using Ranges](#)^[80] for additional details.

Defaults

Start time: Current time
End time: Current time
Time type: Modification (last write)

3.5.2.4 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. **TCC** examines each file name and excludes files that match the names you have specified in the exclusion range.

When you use an exclusion range in a command it should immediately follow the command name. See [General Rules for Using Ranges](#)^[80] for additional details.

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](#)^[77], but may not include path names or drive letters. You can exclude directories by appending a **** to the name.

The following example will display all files in the current directory except backup files (files with the extension **.BAK** or **.BK**):

```
dir /[!* .bak *.bk] *
```

You can combine file exclusion ranges with [date, time, and size ranges](#)^[80]. 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:

```
dir /[s10k] /[d-7] /[!* .c *.h] *
```

File exclusion ranges, a unique feature of **TCC**, work for internal commands. The [EXCEPT](#)^[225] command can also 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 [DO](#)^[210] or [FOR](#)^[234] command; however, the performance will 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. If no such file is found, it interprets the brackets as wildcards.

See also: [Include Lists](#) ^[88].

3.5.2.5 Description Ranges

Most internal commands which accept wildcards also accept description ranges to further define the files that you wish to work with.

When you use a description range in a command it should immediately follow the command name. See [General Rules for Using Ranges](#) ^[80] for additional details.

A description range is specified as `/!"text"` where *text* is the description to be matched. [Wildcards](#) ^[77] are supported. For example, `/!"*agua"` selects all files with the string **agua** somewhere in the file description. The search text must be enclosed in double quotes, and must immediately follow the `/!`, with no intervening spaces.

You can select all files that have a description with `/!"[?]"` (the `[?]` requires that the description contain at least one character, and the `*` allows any text).

You can select all files that do not have a description with `/!"[]"` (the `[]` requires that the first character, and therefore the descriptor itself, does not exist).

You can also search descriptions using [regular expressions](#) ^[496] with `/R" text"`.

If you precede the `I` or `R` with a `!`, the result is reversed. For example, `/!"*beta"` will select all of the files that do **not** have the word **beta** in their description.

See [DESCRIBE](#) ^[195] for details on file descriptions.

3.5.3 Attribute Switches

Most file commands in **TCC** include the `/A:` switch, which allows you to select files for the command to process based on their [attributes](#) ^[494]. These switches all use the format `/A:[-+]RHSAD`. The colon after `/A` is optional in [DIR](#) ^[198], [FFIND](#) ^[227], and [SELECT](#) ^[312], but is required in all other commands. 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](#) ^[532] volumes, the extended attributes below are also available.

- E** Encrypted
- C** Compressed
- I** Not content-indexed
- L** Symbolic link or 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, system, and/or read-only files in the default directory visible and writeable, but not modify the attributes of files which are neither hidden nor system nor read-only (thus not reporting files already in the desired state):

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

3.5.4 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](#) ^[77] 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:

```
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:

```
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](#) ^[182] or [MOVE](#) ^[274], 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](#) ^[77] and [include lists](#) ^[88], 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](#)^[90] as well as the [DO](#)^[210], [FOR](#)^[234] and [SELECT](#)^[312] commands for other ways of passing multiple file names to a command.

3.5.5 Include Lists

Any internal command that accepts [multiple filenames](#)^[87] will also accept one or more include lists. An include list is simply a group of filenames, with or without wildcards, separated by semicolons [;]. Only the first entry in each include list may specify a path. All files in an include list must be in the same directory. You may not add a space on either side of the semicolon. See the rule below to determine when a [semicolon is part of a file name](#)^[88] and when it is an include list separator.

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

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

to this using an include list:

```
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](#)^[182], 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](#)^[182] 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](#)^[198] and [SELECT](#)^[312] 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](#)^[312] list. You can see this difference clearly if you experiment with both techniques and the [DIR](#)^[198] command. For example,

```
dir \doc\*.txt *.doc
```

will list all the *.TXT* files in directory *\DOC* 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 in the current directory. However,

```
dir \doc\*.txt;*.doc
```

will display all the *.TXT* and *.DOC* files in directory *\DOC* in one list.

Like [extended wildcards](#)^[77] and [multiple filenames](#)^[87], include lists 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 2,047 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 **TCC** to treat it as an include list.

If a filename parameter includes a semicolon, **TCC** first attempts to find a filename containing an embedded semicolon. If found, that filename is used. If no file is found, the semicolon is considered to be an include list separator.

See also: [Exclusion Ranges](#)^[85].

3.5.6 Delayed Variable Expansion

Some of the internal commands ([COPY](#)^[182], [MOVE](#)^[274], [PDIR](#)^[288], [REN](#)^[305]) support delayed variable expansion for the target filename. The function argument must be an asterisk (*), which will be replaced by the name of each matching source file. The variable function name must be preceded by two %%'s; the first one will be removed before the command is called, and the second when the command calls the variable expansion routine. This allows much greater flexibility in building the target filenames.

For example, to copy all of your *.MP3 files, and append the string "_saved" to the filename part :

```
copy *.mp3 %>@name[*]_saved.mp3
```

3.5.7 LFN File Searches

There are some 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 SFNs, 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](#)^[531] (LFN) specified, and an [SFN](#)^[534] 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 **CMD.EXE** performs a wildcard search, it searches for both forms of each file name. 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
<i>Letter Home.DOC</i>	<i>LETTER~1.DOC</i>
<i>Letter02.DOC</i>	<i>LETTER02.DOC</i>

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 return LFN.

Because this dual search can result in some very unexpected or even disastrous results, **TCC** defaults to searching only for the LFN. You can change the default with the **Search for SFNs** option in the OPTION / Startup dialog.

Take extra care when you use wildcards to perform operations on LFN volumes if you have set **Search for SFNs**, because you may select more files than you intended. For example, Windows often generates short filenames that end with ~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!

3.5.8 File Lists

Many internal commands allow you to specify a file containing a list of all of the files you want to process in the command line (instead of enumerating them individually). You specify that a file is a file list by prefixing its name with the @ sign, e.g., [LIST](#)^[264] @XXX specifies that [LIST](#)^[264] is to operate on the files listed in the file XXX instead of on XXX itself.

A file list is simply a standard [text file](#)^[535] containing the names of the files to process, one per line. This allows you to create a list of files for processing using output from [DIR](#)^[198] /B, [DIR](#)^[198] /F, or [FFIND](#)^[227], a text editor, or any other method that produces a file in the proper format. Both absolute and relative paths may be included in the file. However, wildcards are ignored, and each line is processed literally, without any further checking. This means that if a command allows options to restrict operations based on age (/U, /C), ranges (/I..., /D..., /T...), attributes (/A:), or location (/S), those restrictions will be ignored when processing the @file contents.

Commands supporting the @File syntax include:

ATTRIB ^[163]	FOR ^[234]	SYNC ^[336]
COPY ^[182]	HEAD ^[249]	TAIL ^[339]
DEL / ERASE ^[190]	LIST ^[264]	TOUCH ^[349]
DESCRIBE ^[195]	MOVE ^[274]	TYPE ^[354]
DO ^[210]	RD / RMDIR ^[301]	
EXCEPT ^[225]	REN / RENAME ^[305]	

To use a 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*:

```
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:

```
copy @e:\lists\mylist.txt d:\save\
copy e:\lists\@mylist.txt d:\save\
```

To use appropriately formatted data on the Windows clipboard as an catalog file use @CLIP: as the file name, for example:

```
copy @clip: d:\save\
```

@File Lists and "@" Signs in File Names

Note that the @ sign is a rarely used, but legal filename character in Windows. 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.

3.5.9 Switches for File Selection

Many of the file processing commands ([ATTRIB](#)^[163], [COPY](#)^[182], [DEL](#)^[190], [DESCRIBE](#)^[195], [MOVE](#)^[274], [REN](#)^[305], [TYPE](#)^[354], 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. Make sure that any [range](#)^[80] selections precede the options below in the command line.

The common file selection switches include:

- /A:[[-+]rhsadeci]** Select files based on their attributes, for example **/A:RH** selects files which have the read-only and hidden attributes set. See [Attribute Switches](#)^[86] for details; see [File Attributes](#)^[494] for more information on attributes.
- /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.
- /S[n]** Process files in the current directory and all of its subdirectories.

3.6 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, **TCC** 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; or **.BTM**, **.BAT**, or **.CMD** to indicate a [batch file](#).^[130]

You can add to the default list of extensions, and have **TCC** 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](#)^[506] for details on this feature and its relationship to executable extensions. See also: [Executable Files and File Searches](#)^[504].

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 **TCC**, use the [SET](#)^[319] 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:


```
set .edt=c:\edit\editor.exe
```

If the command specified in an executable extension is a batch file or external program, **TCC** will search the [PATH](#)^[368] 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:

```
set .txt="c:\program files\accessories\wordpad.exe"
```

Now, if you have a file called *HELLO.TXT* and enter the command

```
hello
```

TCC 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
```

TCC 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](#)^[77], 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](#)^[357] to remove the corresponding variable.

3.7 Using Internet URLs

If you type an Internet URL (Uniform Resource Locator) which begins with **http:** or **https:** at the prompt, **TCC** 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 your browser.

See [Waiting for Applications to Finish](#)^[123] for information on problems with waiting for the browser to

finish after starting a URL.

3.8 Using FTP and HTTP Servers

TCC allows direct access to remote servers from internal commands such as [COPY](#)^[182], [DEL](#)^[190], [DIR](#)^[198], [MOVE](#)^[274], [MD](#)^[271], [RD](#)^[301], [REN](#)^[305], and [SELECT](#)^[312] via several protocols:

- ▶ [FTP](#)^[93] (basic FTP)
- ▶ [TFTP](#)^[96] (Trivial FTP)
- ▶ [FTPS](#)^[96] (SSL FTP)
- ▶ [HTTP](#)^[96] (basic Web access)
- ▶ [HTTPS](#)^[96] (SSL HTTP)

Note: Not all protocols are supported in every internal command. For example, **DIR** will not work with HTTP or HTTPS (because of limitations in the HTTP / HTTPS protocol).

• FTP support:

The basic filename syntax for anonymous connections is:

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

For example, to get a directory of the JP Software FTP site, you could use this command:

```
Dir ftp://jpsoft.com/*
```

If you don't specify a username and password, **TCC** will look for your FTP user names and passwords in the file *FTP.CFG* (which defaults to the **Take Command** directory). You can specify another directory with the [FTP.CFG](#)^[53] configuration option. You must add entries to the *FTP.CFG* file manually. The format for each line is:

```
url username [(alias)] password [directory template]
```

For example:

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

You can have multiple users for a single FTP site (for example, an admin user and a normal user). You need to add an alias (enclosed in parentheses) following the name of the ftp site. For example:

```
ftp://jpsoft.com (jpadmin) Bob AdminPassword
ftp://jpsoft.com (jppublic) anonymous Bob@jpsoft.com
```

You can then access the server as `ftp://jpadmin` or `ftp://jppublic`.

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

You can also specify an explicit username and password on the command line:

```
ftp://[username:password@]ftp.abc.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):

```
dir ftp://username:password@ftp.abc.com/mydir/ *
```

You can also use the internal [IFTP](#)^[255] command to start an FTP session with a server and then use a simplified syntax to manipulate files on the server.

TCC also supports symbolic hostnames (defined in \windows\system32\drivers\etc\hosts).

TCC 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:

```
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:

```
dir ftp://anonymous:email@domain.com@jpsoft.com/
```

TCC 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

```
dir ftp:myfile.txt
```

TCC 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), **TCC** will instead pass the exact string directly to the remote server.

TCC 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 **TCC** expects, the results may not be what you intend. For example, if the FTP server you are connecting to is case-sensitive, 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](#)^[182] /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](#)^[255] command, **TCC** 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](#)^[255] connection. You can determine if there is an active [IFTP](#)^[255] connection with the [_iftp](#)^[388] and [_iftps](#)^[388] variables.

Before you can use the built-in FTP support or the [IFTP](#)^[255] 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 set the [Proxy](#)^[53] configuration options.

Non-standard FTP servers:

TCC supports directory formats for the following:

- EPLF
- WFTP
- VMS (single-line filenames only)
- NetPresenz (Macintosh)
- Netware
- All known UNIX and Linux formats
- Windows FTP Server

If you have a non-standard FTP server that creates an unusual directory format, you can create an entry in your *FTP.CFG* file to allow **TCC** to parse the FTP server output. The format is described in the *FTP.CFG* following the host name, username, and password. The format characters are:

- I"text"** Do a wildcard comparison of "text" and the directory line; if it matches, discard the entire line. (This is to allow you to skip header & footer lines that would otherwise return garbage.)
- "text"** Compare (and skip) a literal string (does NOT support wildcard searches)
- <space>** Skip whitespace (spaces, tabs)
- !** Skip non-whitespace
- Ignore a single character
- F** Filename. If the F is followed by a . (i.e., "F."), the extension is the next non-whitespace string. The extension will be appended (preceded by a '.') to the filename.
- S** Subdirectory flag. If the S is followed by a =, the next character is the character in a "raw" directory listing that denotes a directory. If you don't specify a =, 'D' is assumed.
- T** Month as a string (i.e., "Jan", "Feb", etc.)
- U** **Linux**-style year (2004) or time (18:30) in the same field.
- Y** Year
- M** Month
- D** Day
- h** Hour
- m** Minute
- H** a or p (for am/pm)
- Z** File size. If the Z is followed by a =, the number following that is the block size.

(Note that upper/lower case is significant for the format characters.)

For example, the *FTP.CFG* entry for JPSOFT.COM can be described as:

```
jpsoft.com,anonymous,JPUser@,S!!!ZTDUF
```

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

See the [FTP](#)^[93] section above for general notes and requirements.

TFTP is only available with [COPY](#)^[182] (and with [MOVE](#)^[274] 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](#)^[93] 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](#)^[93] section above for general notes and requirements.

The **FTPS** syntax is:

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

For example:

```
copy ftps://jpsoft.com/tcmd/tcmd.exe
```

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

See the [FTP](#)^[93] section above for general syntax and requirements.

The **HTTPS** syntax is:

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

For example:

```
copy https://jpsoft.com/xyz.htm
```

3.9 Input / Output Redirection

This section covers features to change how **TCC** 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 **TCC**, also send special messages to the standard error device. Normally, the keyboard is used for standard input and the video display 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](#)^[198] 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](#),^[98] [Piping](#),^[101] and [Keystack](#).^[102] All three are explained in this section. In addition, **TCC** supports a subset of ANSI X3.64 control sequences in displayed text. The last topic in this section explains [ANSI X3.64 support](#)^[101] 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 display. 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](#)^[344] and [Y](#)^[364] commands are "pipe fittings" which add more flexibility to pipes.

- [Redirection and Piping](#)^[97]
- [ANSI X3.64 Support](#)^[101]
- [Keystack](#)^[102]
- [Page and File Prompts](#)^[102]

3.9.1 Redirection and Piping

This section covers redirection and piping. You can use these features to change how **TCC** 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](#)^[198] command, the task is easy. [DIR](#)^[198]'s output goes to the standard output device, and you have redirected standard output to the printer, so the [DIR](#)^[198] command prints filenames instead of displaying them on the screen. You can just as easily send the output of [DIR](#)^[198] (or any other command) to a file or a serial port.

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](#)^[344] and [Y](#)^[364] commands are "pipe fittings" which add more flexibility to pipes.

3.9.1.1 Redirection

Redirection can be used to reassign the standard input ([stdin](#)^[535]), standard output ([stdout](#)^[535]), and standard error ([stderr](#)^[535]) 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 (**COMn** 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 **TCC** (see below for additional redirection options using numeric file handles):

- ▶ [Input redirection](#)^[98]
- ▶ [Output redirection](#)^[98]
- ▶ [Special considerations for specific commands](#)^[99]
- ▶ [NoClobber](#)^[99]
- ▶ [Multiple redirections](#)^[99]
- ▶ [Creating an empty file](#)^[99]
- ▶ [Redirection by handle](#)^[99]
- ▶ ["Here-document" redirection](#)^[100]

Input redirection

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

Output redirection

	<i>overwrite</i>	<i>append</i>
standard output	> filename	>> filename
standard error	>&> filename	>>&> filename
merge standard output and standard error	>& 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](#)^[198] command to a file called *DIRLIST*, you could use a command line like this:

```
dir /b *.dat > dirlist
```

You can use any combination of input and output redirection for the same command, as appropriate for your purpose. For example, this command sends input to the external program **SORT** from the file *DIRLIST*, and sends output from **SORT** to the file *DIRLIST.SRT*:

```
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](#)^[198], 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](#)^[121] (e.g., ...) **> report**), all of the output from the command group is redirected, and redirection ends when the command group is done.

Special considerations for specific commands

You cannot redirect all output from the execution of a [DO](#)^[210] loop due to the restriction that the [DO](#)^[210] command and its matching [ENDDO](#)^[210] may not be part of a command group.

To redirect the output of a [TEXT](#)^[345] command, append the redirection syntax to the [TEXT](#)^[345] command.

When you execute a [FOR](#)^[234] or [GLOBAL](#)^[245] command, redirection is separately performed for each iteration, based on the directory current for that iteration. This can result in repeated overwriting of the output file, or the creation of a separate output file in each directory. To generate a single, cumulative output file, use [Command Grouping](#)^[121] as in the example below:

```
( for /r %f in (*.btm) echo %@full[%f] ) > c:\temp\btm1st
```

NoClobber

When output is directed to a file with **>**, **>&**, or **>&>**, and that file already exists, it will be overwritten. You can protect existing files by using the [SETDOS](#)^[323] /N1 command, the **Protect redirected output files** setting on the [Startup tab](#)^[47] of the configuration dialogs, or the [Protect redirected output file](#)^[47] option.

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

```
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 [Protect redirected output file](#)^[47], use **>!filename**.

Redirection by handle

In addition to the redirection options above, **TCC** also supports the **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:

```
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:

```
dir > outfile 2>&1
```

Notice that you can perform the same operations by using standard redirection features. The two examples above could be written as

```
dir > outfile >&> errfile
```

and

```
dir >& outfile
```

"Here-document" redirection

Wherever input redirection is supported, you can use a Linux-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 double quotes.

3.9.1.2 Piping

Piping is a special form of redirection, using an additional instance of **TCC** for each instance of the **piping** 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:

what is sent to pipe	command format
standard output only	<i>command1</i> <i>command2</i>
merge standard output and standard error	<i>command1</i> & <i>command2</i>

For example, to take the output of the [ALIAS](#)^[154] 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:

```
alias | sort
```

To do the same thing and then pipe the sorted list to the internal [LIST](#)^[264] command for full-screen viewing:

```
alias | sort | list /s
```

The [TEE](#)^[344] and [Y](#)^[364] 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.

Processing each line received from a pipe

To process each line of text sent by the left side of a pipe in **TCC**, you may use the syntax below:

```
dir | for %file in (@CON:) command %file
```

This example shows how to pass each line of piped data to a **command**.

WARNINGS: **TCC** implements pipes by starting a new process for the receiving program. This process goes through the standard secondary shell start-up procedure, including execution of the [TCSTART](#)^[22] 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 **TCC**, 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 [TCSTART](#)^[22] 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](#)^[264], each pipe adds another instance of **TCC**.

3.9.2 ANSI X3.64 Support

There is no support for ANSI X3.64 in Windows. For this reason, **TCC** contains its own limited ANSI X3.64 support (key substitutions are not supported, nor are double-width or double-height characters, or blinking characters). **TCC** interprets only its own output, not the output of external. In some cases

you can redirect the output of an application program to a temporary file, then send it through **TCC** ANSI X3.64 interpreter, e.g., by using the [TYPE](#)^[354] command. This will display ANSI X3.64 correctly, but will not work with an interactive application.

To utilize the **TCC** built-in ANSI X3.64 support you must enable it from the [Windows tab](#)^[49] of the [configuration dialogs](#)^[46], or with the [SETDOS](#)^[323] /A command. You can determine whether or not ANSI X3.64 support is enabled with the [ANSI](#)^[382] internal variable.

Several commands in **TCC** provide alternatives 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 **TCC** see the [ANSI X3.64 Command Reference](#)^[516].

3.9.3 Keystack

The [KEYSTACK](#)^[262] command overcomes two weaknesses of input redirection:

- 1) some programs ignore standard input and read the keyboard through Windows APIs, 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](#)^[262] lets you do exactly that.

[KEYSTACK](#)^[262] sends keystrokes to an application program. Once the [KEYSTACK](#)^[262] buffer is empty, the program will receive the rest of its input from the keyboard. [KEYSTACK](#)^[262] 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](#)^[262] buffer, enclose them in double quotes:

```
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](#)^[262]:

```
keystack F1 Enter
```

See [Keys and Key names](#)^[515] for details on how key names are entered. See the [KEYSTACK](#)^[262] command for information on using numeric key values along with or instead of key names, and other details about using the Keystack.

You must activate the window for the program that will receive the characters before you place them into the Keystack. See [KEYSTACK](#)^[262] for additional details; see [ACTIVATE](#)^[152] for information on activating a specific window.

3.9.4 Page and File Prompts

Page Prompts

Several **TCC** commands can generate prompts, which wait for you to press a key to view a new page or to perform a file activity. When **TCC** is displaying information in page mode, for example with a [DIR](#)^[198] /P or [SET](#)^[319] /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](#)^[190]/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	Yes	process this file
N	No	do not process this file
A	All	remaining files without further prompting
R	Remaining	files without further prompting

The **R** and **A** responses are equivalent; **A** 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](#)^[137].

3.10 OpenAFS

TCC has built-in support for OpenAFS. The parser will recognize Linux-style AFS names (i.e., **/afs/athena/user**) and convert them to Windows-compatible names (i.e., **\\afs\athena\user**). (It will also check for custom AFS mount points, and use that name instead of **afs**.)

See <http://www.openafs.org> for more information on OpenAFS.

3.11 The TCC (4NT) Command Line

A **TCC** window displays a 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](#)^[297] settings. (The default will look something like **[c:\]**). This is called the command line and the prompt is asking you to enter a command.

This section explains the features that will help you while you are entering commands, how keystrokes are interpreted when you enter them at the command line, and how to transfer text between **TCC** and other applications.

The keystrokes discussed here are the ones normally used by **TCC**. If you prefer using different keystrokes to perform these functions, you can assign new ones with [key mapping directives](#)^[28].

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

- ▶ [Command Line Editing](#)^[104]
- ▶ [Command History and Recall](#)^[106]
- ▶ [Command History Window](#)^[107]
- ▶ [Command Names and Parameters](#)^[109]
- ▶ [Filename Completion](#)^[113]

- ▶ [Filename Completion Window](#) ^[116]
- ▶ [Variable Completion](#) ^[119]
- ▶ [Automatic Directory Changes](#) ^[76]
- ▶ [Directory History Window](#) ^[118]
- ▶ [Multiple Commands](#) ^[120]
- ▶ [Expanding and Disabling Aliases](#) ^[120]
- ▶ [Command Line Length Limits](#) ^[126]
- ▶ [Command Grouping](#) ^[121]
- ▶ [Starting Applications](#) ^[122]
- ▶ [Command Parsing](#) ^[124]
- ▶ [Date Formats](#) ^[127]

Additional command line features are documented under [File Selection](#) ^[77] and under [Directory Navigation](#) ^[71].

3.11.1 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](#) ^[31] to execute it (or [Esc](#) ^[31] to erase it).

The command line as typed can contain up to a maximum of 32,767 characters, and it can expand to a maximum of 65,535 characters after variable, function and alias substitution. See [Command Line Length Limits](#) ^[126].

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 the default values, but most editing keys can be redefined via [Command Line Editing Keys](#) ^[32] or [General Input Keys](#) ^[29] directives.

Cursor Movement Keys:

Left ^[31]	Move the cursor left one character.
Right ^[32]	Move the cursor right one character.
Ctrl-Left ^[32]	Move the cursor left one word.
Ctrl-Right ^[32]	Move the cursor right one word.
Home ^[29]	Move the cursor to the beginning of the command.
End ^[31]	Move the cursor to the end of the command.

Insert and Delete Keys:

Ins ^[31]	Toggle between insert and overstrike mode (cursor shape indicates mode).
Del ^[30]	Delete the character under (or to the right of) the cursor, or the highlighted text.
Bksp ^[29]	Delete the character to the left of the cursor, or the highlighted text.
Ctrl-L ^[30]	Delete the word or partial word to the left of the cursor.
Ctrl-R ^[30] or Ctrl-Bksp ^[30]	Delete the word or partial word to the right of the cursor.
Ctrl-Home ^[30]	Delete from the beginning of the line to the cursor.
Ctrl-End ^[30]	Delete from the cursor to the end of the line.
Esc ^[31]	Delete the entire line.
Ctrl-V ^[32]	Paste the first line of text from the clipboard at the current cursor position.
Ctrl-C	Paste all of the text from the clipboard at the current cursor position.
Ctrl-P ^[35]	Paste the last argument from the previous command line.
Ctrl-0 to Ctrl-9	Paste the corresponding argument from the previous command line.

Highlighting:

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 ^[29]	Copy highlighted text to the clipboard

Execution:

Ctrl-K ^[36]	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.
Enter ^[31]	Execute the command line.

Miscellaneous:

F1 ^[25]	Get help for the command (first argument on the line)
Ctrl-F1 ^[34]	Get help for the current word.
Ctrl-F ^[33]	Expand an alias.
Ctrl-X ^[36]	Expand an environment variable.
Ctrl-A ^[35]	Toggle between LFN and SFN.
Alt-PgUp, Alt-PgDn, Alt-Home, Alt-End, Alt-Up, Alt-Down	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 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 **TCC** and other applications (see [Highlighting and Copying Text](#) ^[67] for additional details). You can also use [Drag and Drop](#) ^[68] to paste a filename from another application onto the command line.

Most of the command line editing capabilities are also available when you are prompted for a line of input. For example, you can use the command line editing keys when [DESCRIBE](#) ^[195] prompts for a file description, when [INPUT](#) ^[259] prompts for input from an alias or batch file, or when [LIST](#) ^[264] prompts you for a search string.

If you want your input at the command line to be in a different color, you can use the [Windows tab](#) ^[49] of the [configuration dialogs](#) ^[46].

TCC will prompt for additional command line text when you include the escape character as the very last character of a typed command line. The default escape character is the caret "**^**", but you can also use the symbolic "**%=**" representation for portability. For example:

```
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](#) ^[33]. 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 **TCC** 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](#)^[28], and **Alt-255** itself can be reassigned with the [CommandEscape](#)^[33] configuration option.

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, **TCC** 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](#)^[510] for some additional information.

3.11.2 Command History and Recall

Each time you execute a command, the entire command line is saved in a command history list. You can display the saved commands, search the list, modify commands, and rerun commands. The command history is available at the command prompt and in a special [command history window](#)^[107]. You can choose to use either a [local or global command history](#)^[108].

Command History Keys:

Up	Recall the previous (or most recent) command, or the most recent command that matches a partial command line.
Down	Recall the next (or oldest) command, or the oldest command that matches a partial command line.
PgUp	Display a popup window of the command history (or all entries matching 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 ^[368] environment variable.

Note: The keystrokes shown above are the default values. See [Key Mapping Directives](#)^[28] 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

```
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](#)^[198] command. Just press **Up** repeatedly to scan back through the history list. When the [DIR](#)^[198] command appears, press **Enter** to execute it again. You can also view the [command history in a window](#)^[107].

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](#)^[25] command.

The history list is normally "circular". If you move to the latest command in the list and then press **Up** once more, you'll see the oldest command in the list. Similarly, if you move to the first command in the list and then press **Up** 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.

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 **Up**. 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 **Up**. If you press **Up** 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** again.

You can specify the size of the command history list on the **Command Line** tab of the configuration dialog. When the list is full, the oldest commands are discarded to make room for new ones. You can also use the **Minimum Length** option 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 it in the contents of the [HistoryExclude](#)^[368] 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 reexecuted, 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**) 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** 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.

3.11.3 Command History Window

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

toolbar with buttons for editing, deleting, and moving lines.

To activate the command history window press **PgUp** or **PgDn** 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.)

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

Command History Window Keys:

Up	Scroll the display up one line.
Down	Scroll the display down one line.
Left ^[31]	Scroll the display left 4 columns.
Right ^[32]	Scroll the display right 4 columns.
PgUp	Scroll the display up one page.
PgDn	Scroll the display down one page.
Home	Go to the beginning of the list.
End	Go to the end of the list.
Ctrl-Enter ^[40]	Move the selected line to the command line for editing
Enter ^[41]	Execute the selected line
Ctrl-C	Copy the selected line to the clipboard
Ctrl-D ^[40] or Del	Delete the selected line from the list
Ctrl-E	Edit the selected line in the history window
Ctrl-Up	Move the selected line up one row
Ctrl-Down	Move the selected line down one row
Esc ^[31]	Close the window without making a selection.

Note: The keystrokes shown above are the default values. See [Key Mapping Directives](#) ^[28] for details on how to assign different keystrokes.

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 control the position and size of the history window with the corresponding items on the [Windows tab](#) ^[49] of the [configuration dialogs](#) ^[46]. You can also change the keys used in the window with [key mapping directives](#) ^[28].

3.11.4 Local and Global History Lists

The [command history](#) ^[25] and [directory history](#) ^[208] can be stored in either local or global lists.

With a local list, any changes made to the history will only affect the current **TCC** tab window. They will not be visible in other tabs or other copies of **TCC**.

With a global list, all **TCC** windows will share the same history, and any changes made to the history in one copy (e.g., by executing commands from the prompt) will affect all other copies. Global lists are the default.

You can control the type of history list with the [Local History](#) ^[47] and [Local Directory History](#) ^[47] options, and with the **/L**, **/LD** and **/LH** options of either the [START](#) ^[33] command or **TCC**.

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

If you want the histories to be retained in memory even when no **TCC** session is running, see the [SHRALIAS](#)^[329] command, which retains the global alias, user-defined function, command history, and directory history lists. [SHRALIAS](#)^[329] retains the lists in memory, but cannot preserve it when Windows itself is shut down or the user logs out. To save your histories for the next restart of 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](#)^[251] and [DIRHISTORY](#)^[208] commands

3.11.5 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:

```
dir
copy file1 file2 d:\
f:\util\mapmem /v
"c:\program files\JPSoft\tcmd9\tcc.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 options or parameters, and there are spaces between the options and parameters.

Some commands may work when options or parameters are entered directly after the command (without an intervening space, e.g. `dir/p`), or when several options or 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 Linux syntax where forward slashes are used in command paths, and want **TCC** to recognize this approach, you can set the [Unix/Linux-style Paths](#)^[47] option.

For more information on command entry see [Multiple Commands](#)^[120] and [Command Line Length Limits](#)^[126]. For details on how **TCC** handles the various elements it finds on the command line see [Command Parsing](#)^[124].

3.11.6 Conditional Expressions

The commands [DO](#)^[210] (when used with the UNTIL or WHILE keyword), [IF](#)^[253], [IFF](#)^[254]/ELSEIFF, and the variable function [@IF](#)^[439] evaluate a conditional expression, and perform a different action based on whether or not the expression is TRUE. The [SWITCH](#)^[335] command tests pairs of values for equality. Most of the [examples](#)^[113] below use the [IF](#)^[253] command, but 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](#)^[110]
- ▶ [status test](#)^[111]
- ▶ [logical expression](#)^[112]

Relational Expression

A relational expression compares two character strings, using one of the [relational operators](#)^[110] 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 double quotes are significant.

Numeric and String Comparison

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 uses the rules described for the [@NUMERIC](#)^[451] function to determine whether or not the strings are numeric, and only if both are numeric is a numeric comparison performed. If either value is non-numeric, a string comparison is used. To force a string comparison when both values may be numeric, use double quotes 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 numerically, you must convert them to decimal numbers using [@CONVERT](#)^[413]. This is not necessary if both are the same length - string comparison and numeric comparison yield the same result.

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
2 lt 19	true	numeric
"2" lt "19"	false	string

Relational Expression Formats

The format of a relational expression is one of

num1 [relational operator](#)^[110] *num2*
string1 [relational operator](#)^[110] *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

operator	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>
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 parameters in double quotes in order to avoid syntax errors which can occur if one of the parameter 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.

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

CMD.EXE	TCC
EQL or EQU	EQ
NEQ	NE
LSS	LT
LEQ	LE
GTR	GT
GEQ	GE

[Internal variables](#)^[372] and [variable functions](#)^[395] 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 Test

These conditions test operating system, file system or **TCC** 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.

Note: [GOSUB variables](#)^[247] and [internal variables](#)^[372] always fail the DEFINED test.

ERRORLEVEL [\[relational operator\]](#)^[110] 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](#)^[110] 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 filename, in which case the expression is true if any file matching the wildcard name exists. filename may include an absolute or relative path.

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

The special filename NUL is commonly used in CMD.EXE batch files to test the existence of a directory. The expression exist xxx\NUL is true only if xxx is a directory.

ISALIAS <i>aliasname</i>	If <i>aliasname</i> is defined as an alias, the expression is true.
ISAPP <i>appname</i>	If <i>appname</i> 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 ^[89] for details on the correspondence between short and long filenames). This test may require DEBUG privilege.
ISDIR <i>path</i> DIREXIST <i>path</i>	If the directory specified by <i>path</i> exists, the expression is true. Path may be either absolute or relative. DIREXIST may be used as a synonym for ISDIR.
ISFILE <i>filename</i>	If <i>filename</i> 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 <i>name</i>	If the user-defined function name is loaded, the expression is true.
ISINTERNAL <i>command</i>	If <i>command</i> is an active internal command, the expression is true. Commands can be activated and deactivated with the SETDOS ^[323] /I command.
ISLABEL <i>label</i>	If <i>label</i> exists in the current batch file, the expression is true. Labels may be one or more words long. Note that this test has nothing to do with disk partition labels.
ISPLUGIN <i>name</i>	If <i>name</i> is a plugin ^[293] variable, function, or command, the expression is true.
ISWINDOW "title"	If a window which matches the title exists, the expression is true. double quotes must be used around the title, which may contain wildcards and extended wildcards.
PLUGIN <i>module</i>	If the plugin module is loaded, the expression is true.

Logical Expressions

A logical expression is one of the following:

- ▶ a [relational expression](#) ^[110]
- ▶ a [status test](#) ^[111]
- ▶ the unary logical operator **NOT** followed by a [logical expression](#) ^[112]
- ▶ two [logical expression](#) ^[112]s connected by a binary logical operator

Logical operators

operator	type	usage	value is TRUE if
NOT	unary	NOT <i>cond</i>	<i>cond</i> is FALSE.
.AND.	binary	<i>cond1</i> .AND. <i>cond2</i>	both <i>cond1</i> and <i>cond2</i> are TRUE.
.OR.	binary	<i>cond1</i> .OR. <i>cond2</i>	at least one of <i>cond1</i> and <i>cond2</i> is TRUE.

.XOR.	binary	<i>cond1</i> .XOR. <i>cond2</i>	one of <i>cond1</i> and <i>cond2</i> 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 be used only 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
```

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 isfile 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
```

3.11.7 Filename Completion

Filename completion can help you by filling in a complete file name on the command line when you only remember or want to type part of it. Filename completion can be used at the command line, which

is explained here, and in a [filename completion window](#)^[116].

Filename Completion Keys:

F8 or Shift-Tab	Get the previous matching filename.
F9 or Tab	Get the next matching filename.
F10	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](#)^[28] 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:

```
copy au
```

and then press the **Tab** key or **F9** key. **TCC** will search the current directory for filenames that begin with *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 **TCC** 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 **F10** 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](#)^[77]. For example, you can copy the first matching *.TXT* file by typing

```
copy *.txt
```

and then pressing **Tab**.

If you don't specify part of a filename before pressing **Tab**, **TCC** 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. **Note:** you must terminate the command (e.g., by space) before file completion becomes available.

If you type a filename without an extension, **TCC** will append *** to the name on [LFN](#)^[53] drives, and **.** on drives which only support [short file names](#)^[53]. It will also place a *** after a partial extension. If you are typing a group of file names in an [include list](#)^[88], 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](#)^[91], 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](#)^[76]. 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 **TCC**, 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](#)^[198] /O:U would list them. That order is determined by the underlying file system.

TCC 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 [Server Completion](#)^[50] configuration option.

Filename completion will search the [PATH](#)^[286] for an executable filename if you have set the **Search Path** option in the **Command Line** configuration tab, and you are :

- (1) at the beginning of the command line, and
- (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](#)^[286].

If you are on an NTFS drive, you can also complete stream names. For example:

```
copy test:t
```

and then pressing **Tab** will search the file **test** for streams beginning with "t". Note that you cannot complete a filename and a stream name simultaneously (i.e., **t*:t***).

Several topics are related to filename completion. See:

- ▶ [Converting Between Long and Short Filenames](#)^[117]
- ▶ [Appending Backslashes to Directory Names](#)^[117]
- ▶ [Customizing Filename Completion](#)^[115]
- ▶ [Filename Completion Window](#)^[116]
- ▶ [Variable Name Completion](#)^[119]

3.11.8 Customizing Filename Completion

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

To customize filename completion you can use the [Filename Completion](#)^[50] configuration options. You can also use the [FILECOMPLETION](#)^[368] environment variable. If you use both, the environment variable will override the configuration option. You may find it useful to use the environment variable for experimenting, then create permanent settings with the configuration dialog .

The format for both the environment variable and the directive is:

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


where

cmd1 etc. are command names

ext1 etc. are file extensions (which may include wildcards) or one of the following file types:

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

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

The command name is the internal command, alias, or executable file name (without a path). For example, to have file completion return only directories for the [CD](#)^[177] 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:

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

To set the same results using the [FILECOMPLETION](#)^[368] environment variable:

```
set filecompletion=cd:dirs; winedit:c asm
```

With this setting in effect, if you type "CD " and then pressed **Tab**, **TCC** returns only directories, not files. If you type **WINEDIT** and press **Tab**, you will see only names of .C and .ASM files.

When testing for a customized filename match, **TCC** checks the actual command line you type (but without expanding any aliases). For example, if you use the definition above and have "W" aliased to "WINEDIT" and then enter a "W" command, filename completion – which refers only to "WINEDIT" – will be ignored. To use customized filename completion for aliases you must enter the alias name:

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

3.11.9 Filename Completion Window

You can view matching filenames in a filename completion window. To activate the window, press **F7** or **Ctrl-Tab** at the command line. You will see a popup window, 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](#)^[507] for details.

Filename Completion Window Keys:

F7 or Ctrl-Tab ^[35]	(from the command line) Open the window.
Up	Scroll the display up one line.
Down	Scroll the display down one line.
Left ^[31]	Scroll the display left 4 columns.
Right ^[32]	Scroll the display right 4 columns.
PgUp	Scroll the display up one page.
PgDn	Scroll the display down one page.
Home	Go to the beginning of the list.
End	Go to the end of the list.

[Enter](#) ^[41] or Double Click Insert the selected filename into the command line.

Note: The keystrokes shown above are the default values. See [Key Mapping Directives](#) ^[28] for details on how to assign different keystrokes.

See also: [Filename Completion](#) ^[113]

3.11.10 Converting Between Long & Short Filenames

On LFN drives, **TCC** will search for and display long filenames during filename completion. If you want to search for 8.3 short filenames (SFNs), press [Ctrl-A](#) ^[35] 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](#) ^[35] directive can be used to change the keystroke assigned to this feature.

You can press **Ctrl-A** at any time prior to beginning filename completion. The switch to SFN format remains in effect for the remainder of the current command line. When **TCC** 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.

3.11.11 Appending Backslashes to Directory Names

If you set the **Add \\ to Directories** option in the Command Line tab of the configuration dialogs, **TCC** will add a trailing backslash \\ to directory names. The character appended is a slash / for directory names in [FTP URLs](#) ^[93] or (if you have set the **UNIX/Linux-style Paths** option in the Startup tab) 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** keystrokes can build a complete path to the file you want to work with.

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

```
1  edit \da <F9>
2  edit \data\
3  edit \data\f <F9>
4  edit \data\frank.doc <F9>
5  edit \data\finance\
6  edit \data\finance\map <F9>
7  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.

3.11.12 Extended Parent Directory Names

TCC has an extended syntax for referencing parent directories, 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 your default directory is `C:\DATA\FINANCE\JANUARY`, you can copy the file `LETTERS.DAT` from 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](#)^[43] function to expand file and directory references when necessary:

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

3.11.13 Directory History Window

[The directory history window is part of a set of comprehensive directory navigation features built into **TCC**. For a summary of these features, and more information on enhanced directory navigation features, see [Directory Navigation](#)^[71].]

The directory history window includes a toolbar with buttons for editing, deleting, and moving lines.

Directory History Window Keys:

F6 or Ctrl-PgUp ^[34]	Open the window from the command line
Up	Scroll the display up one line.
Down	Scroll the display down one line.
Left ^[31]	Scroll the display left 4 columns.
Right ^[32]	Scroll the display right 4 columns.
PgUp	Scroll the display up one page.
PgDn	Scroll the display down one page.
Home	Go to the beginning of the list.
End	Go to the end of the list.
Ctrl-Enter ^[40]	Move the selected line to the command line for editing
Enter ^[41]	Change to the selected drive/directory
Ctrl-C	Copy the selected line to the clipboard
Ctrl-D ^[40] or Del	Delete the selected line from the list
Ctrl-E ^[40]	Edit the selected line in the directory history window
Ctrl-Up	Move the selected line up one row
Ctrl-Down	Move the selected line down one row
Esc ^[31]	Close the window without making a selection.

Note: The keystrokes shown above are the default values. See [Key Mapping Directives](#)^[28] 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](#)^[34] at the command line. You can then select a new directory with the [Enter](#)^[41] 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 [Command History Buffer Size](#)^[50] configuration option. You can change the keys used in the window with [key mapping directives](#)^[28].

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 **TCC**.

You can also view and manage the directory history list with the [DIRHISTORY](#)^[208] 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 be known only to the current copy of **TCC**. 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.

All copies of **TCC** using global directory history list will share a single copy of directory history. Any directory changes made in any of these copies of **TCC** will be recorded in that shared list, and be accessible by all of them. However, any additional copies of **TCC** which use local directory history will see their own local lists. Global lists are the default for **TCC**.

You can control the type of history list with the [Local Directory History](#)^[47] configuration option, with the /L and /LD [command line options](#)^[19], and with the /L and /LD options of the [START](#)^[331] command.

When you close all **TCC** 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 **TCC**. If you want the directory history list to be retained in memory even when no copy of **TCC** is running, you need to execute the [SHRALIAS](#)^[329] command, which performs this service for the global command history, directory history, user-defined functions, and aliases.

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.

3.11.14 Variable Name Completion

Variable name completion works like [filename completion](#)^[113]. If the parameter begins with a %, the completion routines will scan the environment for matching variable names. For example, if the [PROMPT](#)^[369] and [PATH](#)^[368] variables are in the environment, in that order, and no other variables start with p, the sequence below may be used to display the value of [PATH](#)^[368]:

```
echo %p<Tab>
echo %PROMPT<Tab>
echo %PATH<Enter>
```

Note: Variable name completion does not work for variable functions.

3.11.15 Expanding and Disabling Aliases

A few command line options are specifically related to aliases, and are documented briefly here for completeness. If you are not familiar with aliases, see [Aliases](#)^[128] and the [ALIAS](#)^[154] command for complete details.

You can expand an alias on the command line and view or edit the results by pressing [Ctrl-F](#)^[33] before the command is executed. This is 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:

```
*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 will endlessly try to redefine itself.

3.11.16 Multiple Commands

You will often 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 the [command separator](#)^[524] (by default, an ampersand &) or the %+ pseudovisible. For example, if you know you want to copy all of your .TXT files to D:\TEXT and then delete all of them beginning with 'A', you could enter the following command:

```
copy *.txt d:\text\ & del a*.txt
```

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 32,767 characters before alias and variable expansion, and 65,535 characters after expansion.

You can use multiple commands in [alias](#)^[154] definitions and [batch files](#)^[130] 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](#)^[323] command's /C option, or the [Separator character](#)^[51] configuration option.

3.11.17 Conditional Commands

When an internal command or external program finishes, it returns a result called the [exit code](#)^[24]. Conditional commands allow you to perform tasks based upon the previous command's [exit code](#)^[24]. Many programs return 0 if they are successful and a non-zero value if they encounter an error.

AND operator &&

If you separate two commands by && (AND), the second command will be executed only if the first command's [exit code](#)^[24] is 0. For example, the following command will only erase files if the BACKUP operation succeeds:

```
backup c:\ a: && del c:\*.bak;*.lst
```

OR operator ||

If you separate two commands by || (OR), the second command will be executed only if the first command's [exit code](#)^[24] is non-zero. For example, if the following BACKUP operation fails, then [ECHO](#)^[217] will display a message:

```
backup c:\ a: || echo Error in the backup!
```

All internal commands return an [exit code](#)^[24], 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](#)^[24]. To determine whether a particular external program returns a meaningful [exit code](#)^[24] use an ECHO %? command immediately after the program is finished. If the program's documentation does not discuss [exit code](#)^[24], you may need to experiment with a variety of conditions to see how the [exit code](#)^[24] changes.

3.11.18 Command Grouping

Command grouping allows you to group a set of commands together logically by enclosing them in parentheses.

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](#)^[305] commands in all subdirectories of your hard disk. You could do it like this:

```
global ren *.wxl *.wxo
global ren *.txl *.txo
```

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

```
global (ren *.wxl *.wxo & ren *.txl *.txo)
```

The two [REN](#)^[305] commands enclosed in the parentheses appear to [GLOBAL](#)^[245] as if they were a single command, so both commands are executed for every directory, but the directories are only scanned once, not twice, typically saving time.

This kind of command grouping is most useful with the [EXCEPT](#)^[225], [FOR](#)^[234], [GLOBAL](#)^[245], and [IF](#)^[253] 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. Examples 1 and 2 below will work properly, but Example 3 will not:

Example 1 (correct):

```
for %f in (1 2 3) (echo hello %f & echo goodbye %f)
```

Example 2 (correct):

```
for %f in (1 2 3) (
    echo hello %f
    echo goodbye %f
)
```

Example 3 (incorrect):

```
for %f in (1 2 3) (echo hello %f
    echo goodbye %f)
```

If the above examples are typed at the command line, **TCC** will 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](#)^[98] 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](#)^[217] 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](#)^[98] and [piping](#)^[101].

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 32,767 characters before alias and variable expansion, and 65,535 characters after expansion, whether the commands are entered from the prompt, an alias, or a batch file.

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](#)^[126].

3.11.19 Starting Applications

TCC 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), **TCC** 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.

TCC offers two methods to simplify and speed up access to your applications. One is to create an [alias](#)^[154], for example:

```
alias myapp d:\apps\myapp.exe
```

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. **TCC** will examine the file's extension and run the appropriate application, based on [executable extensions](#)^[91] or [Windows file associations](#)^[506].

For additional flexibility, you can also start applications with the [START](#)^[331] command. [START](#)^[331] 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, **TCC** searches for the application in the current directory, and then all directories on the PATH. **TCC** also searches the **Windows** and **Windows system** directories; see the [PATH](#)^[286] command for details. (If you do enter an explicit path, **TCC** will only look in the directory you specified.)

If you enter a file name with no extension, **TCC** will search each directory for a matching .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 via the [PathExt](#)^[47] configuration option. If no such file is found, **Take Command** will move on to the next directory in the search sequence.

Take Command Application Windows

Take Command runs console (character mode) applications either in a tab window within **Take Command** or in their own console window. **Take Command** always starts GUI applications in their own window.

3.11.20 Waiting for Applications to Finish

When you start a Windows GUI application from the prompt, **TCC** 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 **TCC** to wait for applications to finish before continuing by selecting the [Wait for External Apps](#)^[47] configuration option, or with the [START](#)^[331] command's /WAIT switch ([START](#)^[331] can also control many other aspects of how your applications are started).

TCC 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](#)^[331] command (without the /WAIT switch).

Due to the way Windows handles URLs, you cannot wait for the browser to finish when you enter an HTTP: URL at the prompt. In this situation, **TCC** always displays the next prompt immediately.

3.11.21 Escape Character

TCC recognizes a user-definable escape character. This character gives the character that follows a special meaning; it has a different purpose than 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 the default escape character, you can pick another character using the [SETDOS](#)^[323] /E command, or the [Escape character](#)^[51] configuration option. If you plan to share aliases or batch files between several **TCC** configurations, use the [%=](#)^[381] pseudovariable, which is accepted in all of them, regardless of the actual value assigned to the escape character. See the section on [Special Character Compatibility](#)^[126] for details about choosing compatible escape characters. Note that if you change the default, your batch files will not work under **CMD.EXE** and you won't be able to run third-party batch files.

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. The special characters which can follow the escape character are:

Codes for Escape Characters

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

```
echo 2 is %=> 4
```

The escape character has an additional use when it is the last character on any line of a batch file. **TCC** 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.

WARNING: Escape characters are considered to be normal characters on the right side of a pipe.

Note: The term **escape character** has two additional usages not related to the above description, as detailed in the description of the [PROMPT](#)^[297] command and in [ASCII, Key Codes and Key Names](#)^[510].

3.11.22 Command Parsing

Whenever you type something at the command line and press the [Enter](#)^[31] key, or include a command in a batch file, you pass a command to **TCC**, which must determine 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.

TCC goes through several steps when parsing a command line. Before it starts, it writes the entire command line (which may contain [multiple commands](#)^[120]) to the history log file if history logging has

been enabled (with the [LOG /H](#)^[269] command) and the command did not come from a batch file. The first command is then 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 tail

TCC 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, **TCC** tries to match the command name against its list of [aliases](#)^[128]. 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](#)^[33] 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 parameters, environment variables, internal variables, or variable functions in the command, and replace each one with its value (see "[Environment: Variables and Functions](#)"^[365]). This process is called variable expansion, and is not normally visible. However, you can view an expanded command line by pressing [Ctrl-X](#)^[36] after entering the command at the prompt.

The variable expansion process is modified for certain internal commands, such as [EXCEPT](#)^[225], [IF](#)^[253], and [GLOBAL](#)^[245]. 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 a plugin or internal command

Once it has finished variable expansion, **TCC** next tries to match the resulting command name with its list of [plugin](#)^[293] commands or [internal commands](#)^[144]. 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, **TCC** will echo the complete command to the screen (if command line echo has been enabled) and write it to the log file (if

command [logging](#)^[269] has been turned on).

6. Processing redirection and piping

Before it can actually execute your command, **TCC** must scan the command tail to see if it includes [redirection](#)^[98] or [piping](#)^[101]. 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, any remaining [Escape Characters](#)^[124] are processed. However, this might also already have taken place inside some of the variable functions (such as [@IF](#)^[439]) that are likely to pass escaped strings in their parameters. If you are referencing one of those in an [ECHO](#)^[217] or similar command, you need to escape twice ("^^") or use [SETDOS /X](#)^[323] to avoid premature evaluation. Carefully test those situations to make sure the results are as you intended.

8. Executing the command

Finally, it is time to execute the command. **TCC** will first look for a matching [plugin](#)^[293] command name; if it doesn't exist then it tries to match an internal command. Otherwise, **TCC** 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](#)^[504]).

9. Cleaning up

Once the internal command or external program has terminated, **TCC** 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 reenable several parts of command parsing (for example alias expansion, variable expansion, and redirection) with the [SETDOS /X](#)^[323] command.

3.11.23 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 32,767 characters long.

As **TCC** 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 65,535 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.

3.11.24 Special Character Compatibility

If you want to share aliases, user defined functions, and batch files with other users, you need to be aware of possible differences in three important characters: the Command Separator (see [Multiple Commands](#)^[120]), the Escape Character (see [Escape Character](#)^[124]), and the Parameter Character (see [Batch File Parameters](#)^[133]).

The default values of each of these characters is shown in the following chart.

Product	Separator	Escape	Parameter
4DOS (obsolete)	^	Ctrl-X	&
Take Command	&	^	\$
<i>pseudovvariable</i>	%+	%=	

In your batch files and aliases, and even at the command line, you can smooth over these differences in three ways:

1. Use internal pseudovvariables that contain the current special character, rather than using the character itself (see [%+](#)^[38†] and [%=](#)^[38†]). For example, this command:

```
if "%1" == "" (echo Parameter 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 Parameter missing! %+ quit)
```

2. Select a consistent set of characters from the [Advanced tab](#)^[51†] of the [configuration dialogs](#)^[46†].
3. In a batch file, use the [SETLOCAL](#)^[326†] command to save the command separator, escape character, and parameter character when the batch file starts. Then use [SETDOS](#)^[323†] as described below to select the characters you want to use within the batch file. Use an [ENDLOCAL](#)^[220†] command at the end of the batch file to restore the previous settings.

You can also use the [SETDOS](#)^[323†] command to change special characters on the command line.

3.11.25 Date Input Formats

Date Input Formats

Commands and functions which accept a date as a parameter expect the same field order displayed by the [DIR](#)^[198†] command and functions returning a date without a format code specifier. 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. Month and day may be entered without a leading zero. Most non-numeric printing characters are accepted as field separators. All three fields must be specified.

3.11.26 Case Sensitivity

With the following exceptions, **TCC** treats upper case and lower case letters identically:

The relational operator **EQC**

The character manipulation functions **@ascii**, **@unicode**, **@repeat**, **@replace**, **@similar**, **@strip** and **@wild**.

The codes used to specify units of storage size (**kKmMgGtT**) in:

- size ranges
- disk space and file size reporting functions

3.12 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](#)^[103], one that needs to be part of an exact sequence of other commands, 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.

- ▶ [Aliases](#)^[128]
- ▶ [Batch Files](#)^[130]
- ▶ [Special Character Compatibility](#)^[126]

3.12.1 Aliases

Much of the power of **TCC** 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. **TCC** also supports [Directory Aliases](#)^[76], a shorthand way of specifying pathnames.

This section shows you some examples of the power of aliases. See the [ALIAS](#)^[154] command for complete details about writing your own aliases.

The simplest type of alias gives a new name to an existing command. For example, you could create a command called **R** (for Root directory) to switch to the root directory this way:

```
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](#)^[177].

Aliases can also create customized versions of commands. For example, the [DIR](#)^[198] 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:

```
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):

```
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](#)^[154] 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](#)^[123] for additional details.

Aliases can be nested; that is, one alias can invoke another. For example, the alias above could also

be written as:

```
alias play=e:\music\player.exe
alias music=`pushd c:\sounds & play & popd`
```

If you enter *MUSIC* as a command, **TCC** executes the [PUSHD](#)^[299] 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](#)^[294] 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](#)^[190] command to prompt before it erases a file:

```
alias del=*del /p
```

An asterisk ***** is used in front of the second **DEL** to tell **TCC** to use the original internal command, not an alias. See [Temporarily Disabling Aliases](#)^[156] 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](#)^[195] 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:

```
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](#)^[195] command, and then uses a second alias to run *DESCRIBE.EXE* whenever you type **DESCRIBE**:

```
alias filedesc=*describe
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**:

```
alias @Shift-F5=*dir /2/p
```

This alias will put the [DIR](#)^[198] 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:

```
alias @@Shift-F5=*dir /2/p
```

The next example clears the window whenever you press **Ctrl-F2**:

```
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:

```
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:

```
alias in=`pushd %1 & %2$ & popd`
```

Now if you type:

```
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.

Alias Parameters

The above example uses two parameters: **%1** means the first parameter on the command line, and **%2\$** means the second and all subsequent parameters.

Aliases can use command line parameters or parameters like those in batch files. The command line parameters are numbered from **%0** to **%511**. (**%0** contains the alias name.) You can use double quotes to pass spaces, tabs, commas, and other special characters in an alias parameter; see [Parameter Quoting](#)^[134] for details. Alias examples in this section assume the **TCC** default of **ParameterChar=\$**.

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. **TCC** interprets it to mean "the entire command line, from parameter **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 parameter **%-n\$** means "the command line from parameter 1 to **n** - 1".

The special parameter **%#** contains the number of command line parameters.

Aliases cannot use indirect access to command parameters, e.g., **%[%n]** (where **n** is a parameter number) does not return the selected parameter.

See the [ALIAS](#)^[154] and [UNALIAS](#)^[355] commands for more information and examples.

3.12.2 Batch Files

A batch file is a file that contains a list of commands to execute. **TCC** reads and interprets each line as if it had been typed at the keyboard. Like [aliases](#)^[154], 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](#)^[131]
- ▶ [Using .BAT Files Under TCC](#)^[131]
- ▶ [Echoing in Batch Files](#)^[132]
- ▶ [Batch File Line Continuation](#)^[141]
- ▶ [Batch File Parameters](#)^[133]

- ▶ [Using Environment Variables](#) ^[135]
- ▶ [Batch File Commands](#) ^[135]
- ▶ [Interrupting a Batch File](#) ^[137]
- ▶ [Automatic Batch Files](#) ^[22]
- ▶ [Detecting Take Command](#) ^[137]
- ▶ [Using Aliases in Batch Files](#) ^[137]
- ▶ [Debugging Batch Files](#) ^[139]
- ▶ [String Processing](#) ^[139]
- ▶ [Batch File Compression](#) ^[141]
- ▶ [Parameter Quoting](#) ^[134]
- ▶ [Perl Support](#) ^[142]
- ▶ [REXX Support](#) ^[142]
- ▶ [Ruby Support](#) ^[143]
- ▶ [EXTPROC / Shebang Support](#) ^[143]

3.12.2.1 .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 can be used for self-modifying batch files (which are rare).

The batch file's extension determines its initial mode. Files with a *.BAT* or *.CMD* extension are run in the first mode. Files with a *.BTM* extension are run in the more efficient second mode. You can change the execution mode inside a batch file with the [LOADBTM](#) ^[269] command.

3.12.2.2 Using .BAT Files Under TCC

In most cases under **TCC** 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 Win98 to XP / 2003 / Vista / 2008. If you do, you need to be aware of the way **TCC** 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.

TCC does not use this system; it executes *.BAT* files directly, 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 **TCC** features. While we do not generally recommend using **4DOS** under Windows 2000 / XP / 2003, it works well in this specific situation.

When invoking DOS programs from a **TCC** batch file, we recommend that you enable the *CONFIG.NT* command **NTCMDPROMPT** without which Windows tends to "forget" to return control to a calling 32-bit program (such as **Take Command**) 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.

3.12.2.3 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](#)^[217] OFF and [ECHO](#)^[217] ON commands.
- ▶ The default setting can be changed with the [SETDOS](#)^[323] /V command, or the [Default Batch Echo](#)^[47] configuration option.

For example, the following line turns off echoing inside a batch file. The @ symbol keeps the batch file from displaying the ECHO OFF command itself:

```
@echo off
```

TCC also has a command line echo that is unrelated to the batch file echo setting. See [ECHO](#)^[217] for details about both settings.

3.12.2.4 Special syntax for CMD.EXE compatibility

For compatibility with CMD.EXE, **TCC** supports additional syntax to qualify references to parameters of batch files and the control variable of the [FOR](#)^[234] command when referenced by the **command** it executes. However, this syntax can usually be replaced by the more flexible [Variable Functions](#)^[395].

CMD.EXE 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[%n]
%~an	File attributes of %n	%@attrib[%n]
%~tn	File date and time of %n	%@filedate[%n] %@filetime[%n]
%~zn	File size of %n, bytes	%@filesize[%n]
%~\$PATH:n	Full name of the first match for %n in % PATH ^[368]	%@search[%n]

Notes

In the special case where the parameter to a %~ variable is 0, e.g., %~f0, the returned file name will always include the extension, as it does under CMD.EXE.

%~\$PATH:n returns an empty string if the file %n is not found in the path.

References qualified by the tilde ~ trigger an error message when used improperly, e.g. if attempting to display the size of a string parameter which is not the name of a file.

3.12.2.5 Batch File Parameters

Like [aliases](#) ^[154], user-defined [functions](#) ^[242] 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 parameters 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 **xyz/abc** is separated into parameters **foo** and **/abc**.

These parameters are numbered from **%1** to **%4095**. **%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 double quotes to pass spaces, tabs, commas, and other special characters in a batch file parameter; see [Parameter Quoting](#) ^[134] 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 use the special parameters shown in the table below:

parameter	value
%0	the name of the batch file as entered on the command line
%#	the number of command line parameters, modified by SHIFT ^[327]
%n\$	the command tail starting with parameter number n , modified by SHIFT ^[327]
%-n\$	the command tail from parameter 1 to n - 1
%\$	the complete command tail, modified by SHIFT ^[327]
%*	the complete command tail, unmodified by SHIFT ^[327]

For example, **%3\$** means the third and all subsequent parameters. The values of **%#**, **%n\$**, **%-n\$**, and **%\$** will change if you use the [SHIFT](#) ^[327] command. To emulate CMD.EXE, [SHIFT](#) ^[327] does not affect the value of **%***.

For example, if your batch file interprets the first parameter 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](#) ^[253] and [IFF](#) ^[254] commands.)

Batch files can also use [environment variables](#) ^[365], [internal variables](#) ^[372], and [variable functions](#) ^[395].

Batch file parameters may also use the special [CMD.EXE compatibility syntax](#)^[132].

3.12.2.5.1 Parameter Quoting

As **TCC** [parses](#)^[124] the command line, it looks for the [command separator](#)^[120], [conditional commands](#)^[120] (**|** and **&&**), white space (spaces, tabs, and commas), percent signs **%** which indicate [variables](#)^[365] or [batch file](#)^[130] parameters to be expanded, and [redirection and piping](#)^[97] characters **>**, **<**, and **|**.

Normally, these special characters cannot be passed to a command as part of a parameter. However, you can include any of the special characters in a parameter by enclosing the entire parameter in single back quotes **[]** or double quotes **[]**. Although both back quotes and double quotes will let you build parameters that include special characters, they do not work the same way.

No alias or variable expansion is performed on a parameter 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 when an expression is enclosed in double quotes. Redirection symbols inside double quotes are ignored. However, variable expansion **is** performed in expressions inside double quotes. The double quotes themselves will be passed to the command as part of the parameter.

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:

```
chkname `MY NAME`
```

Inside the batch file, **%1** will have the value **MY NAME**, including the space. The back quotes caused **TCC** to pass the string to the batch file as a single parameter. The quotes keep characters together and reduce the number of parameters 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:

```
set FORVAR=for
```

Now, if you enter the command

```
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:

```
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 parameters in the line.

The following example has 7 command line parameters, while the examples above only have 3:

```
quotes Now is the time %%forvar all good
```

(The double percent signs are needed in each case because the parameter 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 parameter can be enclosed in back quotes to prevent the expansion of replaceable parameters, variables, and multiple commands until the alias is invoked. See [ALIAS](#)^[154] for details.

You can disable and reenable back quotes and double quotes with the [SETDOS](#)^[323] /X command.

3.12.2.6 Using Environment Variables

Batch files can use [environment variables](#)^[365], [internal variables](#)^[372], [variable functions](#)^[395], or [user-defined functions](#)^[242]. You can use these variables and functions to determine system status (e.g., the CPU type), 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](#)^[319] 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](#)^[357] command to remove these variables from the environment at the end of your batch file. You can use [SETLOCAL](#)^[326] and [ENDLOCAL](#)^[220] 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](#)^[420] to add a number to a text string), you'll get an error message or a meaningless return value.

3.12.2.7 Batch File Commands

Some commands are particularly suited to batch file processing. Each command is explained in detail in the [Command Reference](#)^[147]. Here is a list of some of the commands you might find most useful:

ACTIVATE ^[152]	activates another window
BEEP ^[173]	produces a sound of any pitch and duration through the computer's speaker
BREAKPOINT ^[174]	set a breakpoint in the batch debugger
CALL ^[175]	executes one batch file from within another
CANCEL ^[176]	terminates all batch file processing
CLS ^[181]	clears the TCC window
COLOR ^[181]	sets the TCC display colors
DEBUGSTRING ^[190]	send text to the debugger
DEFER ^[190]	defers a command until the batch file ends
DO ^[210]	starts a loop. The loop can be based on a counter, or on a conditional expression, strings, or files. ENDDO terminates the loop

DRAWBOX ^[214]	draws a box on the screen
DRAWHLINE ^[215]	draws horizontal lines on the screen
DRAWVLINE ^[216]	draws vertical lines on the screen
ECHO ^[217]	sends text to the standard output device
ECHOS ^[219]	sends text to the standard output device
ECHOERR ^[217]	sends text to the standard error device
ECHOSERR ^[219]	sends text to the standard error device
ENDLOCAL ^[220]	restores the settings that were saved and allows specific variables to be exported (see SETLOCAL ^[326])
ENDTEXT ^[345]	ends the block of text started with TEXT ^[345]
EVENTLOG ^[223]	writes a string to the Windows application event log
FOR ^[234]	executes commands for each file that matches a set of wildcards, or each entry in a list
GOSUB ^[247]	executes a subroutine inside a batch file (see RETURN ^[307]).
GOTO ^[248]	branches to a different location in the batch file
IF ^[253]	execute commands based on a conditional expression
IFF ^[254]	
INKEY ^[257]	collects keyboard input and store it in environment variables
INPUT ^[259]	collects keyboard input and store it in environment variables
JABBER ^[260]	send an instant message (IM)
KEYSTACK ^[262]	sends keystrokes to applications
LOADBTM ^[269]	changes the batch file operating mode
MSGBOX ^[279]	displays a dialog box with standard buttons like Yes, No, OK, and Cancel, and returns the user's selection
ON ^[282]	initializes error handling for Ctrl-C / Ctrl-Break, or for program and command errors
OSD ^[286]	Display floating text on the desktop
PAUSE ^[287]	displays a message and waits for the user to press a key
PDIR ^[288]	creates a customized DIR-like display of directory contents
PLAYAVI ^[291]	plays Windows .AVI files
PLAYSOUND ^[292]	plays Windows sound files
POSTMSG ^[294]	send a message to a window
QUERYBOX ^[300]	displays a dialog box for text input
QUIT ^[301]	ends the current batch file and optionally returns an exit code
REM ^[304]	places a remark in a batch file
RETURN ^[307]	terminates a subroutine (see GOSUB ^[247])
SCREEN ^[310]	positions the cursor on the screen and optionally prints a message at the new location
SCRPUT ^[311]	displays a message in color
SENDMAIL ^[316]	sends an email message
SETLOCAL ^[326]	saves the current disk drive, default directory, environment, alias list, and special character settings (see ENDLOCAL ^[220]).
SHIFT ^[327]	changes the numbering of the batch file parameters
SMPP ^[330]	sends messages using the SMPP protocol
SNPP ^[331]	sends a message to an alphanumeric pager
START ^[331]	starts another session or window
SWITCH ^[335]	selects a group of statements to execute based on the value of a variable
TCTOOLBAR ^[343]	changes the TC tool bar buttons
TEXT ^[345]	displays a block of text (see ENDTEXT ^[345])
TIMER ^[347]	starts or reads a stopwatch
TITLE ^[349]	changes the window title
VSCRPUT ^[360]	displays a vertical message in color
WMIQUERY ^[364]	Query the Windows Management Instrumentation interface

These commands, along with the internal variables and variable functions, make the enhanced batch

file language extremely powerful. Your copy of **Take Command** includes a sample batch file (EXAMPLES.BTM), that demonstrates some of the things you can do with batch files.

3.12.2.8 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 **TCC** or an application program is running, how the application, if any, was written, whether [BREAK](#)^[174] is ON or OFF, and whether the [ON BREAK](#)^[282] command is in use.

If **TCC** detects a **Ctrl-C** or **Ctrl-Break** when ON BREAK is not in use, it displays 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](#)^[301] command; answering **A** is similar to the [CANCEL](#)^[176] command.

3.12.2.9 Detecting TCC and Take Command

From a batch file, you can determine if **TCC** is loaded by testing for the variable function [@EVAL](#)^[420], with a test like this:

```
if "%@eval[2 + 2]%" == "4" echo Take Command is loaded!
```

This test can never succeed in CMD.EXE. (Other variable functions could also be used for the same purpose.)

You can determine if **TCC** is running in a **Take Command** tab window with the internal variable [_TCTAB](#)^[393]:

```
if %_tctab == 1 echo TCC is running in a Take Command tab window!
```

3.12.2.10 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](#) [217] 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 parameters. It uses the first as a new working directory and changes to that directory with a [PUSHD](#) [299] 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](#) [335] 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](#) [177] to [CDD](#) [178] 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](#) [326], [ENDLOCAL](#) [220], and [UNALIAS](#) [355] to clear out aliases before your batch file starts, and [SETDOS](#) [323] to select the special characters you depend on, and restore them at the end, as we did in the previous example. Remember that [SETLOCAL](#) [326] and [ENDLOCAL](#) [220] 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.

You can use the pseudovariables [%=](#)^[381] and [%+](#)^[381] to represent the [command escape](#)^[124] and [command separator](#)^[51] characters, respectively. There is no pseudovvariable for the [parameter character](#)^[51].

3.12.2.11 Debugging Batch Files

Take Command includes a built-in full-featured batch file debugger invoked with the [BDEBUGGER](#)^[166] command. The debugger gives you a detailed, step-by-step view of batch file execution, and will help solve particularly difficult batch file problems.

3.12.2.12 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.

TCC includes several features that make string processing easier. For example, you can use the [INPUT](#)^[259], [MSGBOX](#)^[279], and [QUERYBOX](#)^[300] commands for user input; the [ECHO](#) and [ECHOERR](#)^[217], [ECHOS](#) and [ECHOSERR](#)^[219], [SCREEN](#)^[310], [SCRPUT](#)^[311], and [VSCRPUT](#)^[360] commands for output; and the [FOR](#)^[234] command or the [@FILEREAD](#)^[429] function to scan through the lines of a file. In addition, [variable functions](#)^[395] offer a wide range of [strings and character handling](#)^[404] 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](#)^[326] command at the beginning of this batch file saves the environment and aliases. Then the [UNALIAS *](#)^[355] 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](#)^[259] 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](#)^[472] function numbers the first word as 0, not as 1.

The [IFF](#)^[254] 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](#)^[220] 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](#)^[98] 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 [double quotes](#)^[134] 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 double quotes (ECHO "%STR"), the string entered by the user will be displayed properly, and so will the double quotes. With double quotes, the output would look like this:

```
"some > none"
```

As you can imagine, this kind of problem becomes much more difficult if you try to process text from a file. Special characters in the text can cause all kinds of confusion in your batch files. Text containing back-quotes, double quotes, or redirection symbols can be virtually impossible to handle correctly.

One way to overcome these potential problems is to use the [SETDOS /X](#)^[323] 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 **TCC**, you may want to consider using the [Perl](#)^[142], [REXX](#)^[142], or [Ruby](#)^[143] languages. Our products can execute Perl, REXX, and Ruby programs internally, and also support evaluating individual Perl, REXX, and Ruby

expressions internally.

3.12.2.13 Batch File Line Continuation

TCC will combine multiple lines in the batch file into a single line for processing when the [Escape Character](#) ^[124] (the actual token or the symbolic "[%="](#) ^[381]" reference) is the last character of each line to be combined (except the last). For example:

```
c:\> echo The quick brown fox jumped over the ^
sleeping ^
dog. > alphabet
```

You cannot use this technique to extend a batch file line beyond the normal [command line length limit](#) ^[126].

3.12.2.14 Batch File Compression

You can compress your *.BTM* files with [BATCOMP](#) ^[166]. That command compresses batch files by about a third and makes them unreadable with the [LIST](#) ^[264] command and similar utilities. Compressed batch files run at approximately the same speed as uncompressed *.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:

```
batcomp mybatch.cmd
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](#) ^[345] 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](#)^[269] 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](#)^[126] for important information on the command separator, escape character, and parameter character used in each product.

The [BATCOMP](#)^[166] command replaces the external program BATCOM32.EXE which was included in previous versions and is now obsolete.

3.12.2.15 REXX Support

REXX is a powerful file and text processing language developed by IBM, and available on many platforms. REXX is an ideal extension to the **TCC** batch language, especially if you need advanced string processing capabilities.

The REXX language is not built into **TCC**, and must be obtained separately through (free) add-on REXX software, such as Open Object REXX (<http://www.oorexx.org/>) or Regina REXX (<http://regina-rexx.sourceforge.net/>).

TCC automatically recognizes the presence of a REXX interpreter on your system. When **TCC** loads, it asks Windows to locate specific REXX libraries associated with Open 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, **TCC** 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, **TCC** passes the file to your REXX interpreter for processing.

(**TC**) When working with a REXX processor, **TCC** 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 **TCC**, 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 **TCC** to be executed (for example, if you execute a *DIR* command within a REXX script), the REXX software must use the correct address for **TCC**. **TCC** uses the address **CMD**.

For details on communication between REXX and **TCC**, or for more information on any aspect of REXX, see your REXX documentation.

See also: the [@REXX](#)^[457], [@PERL](#)^[452] and [@RUBY](#)^[458] functions.

3.12.2.16 Perl support

Perl is a powerful file and text processing language available on many platforms. Perl is an ideal extension to the **TCC** batch language, especially if you need advanced string processing capabilities.

The Perl language is not built into **TCC**, and must be obtained separately. The version supported by **TCC** is Active State Perl 5.8 (free from www.activestate.com).

TCC automatically recognizes the presence of a Perl interpreter on your system. If a suitable library is

found, **TCC** checks to see if you are running a **.PL** file. If so, **TCC** passes the file to your Perl interpreter for processing.

See also: the [@PERL](#)^[452], [@REXX](#)^[457] and [@RUBY](#)^[458] functions.

3.12.2.17 Ruby support

Ruby is a powerful object-oriented file and text processing language available on many platforms. Ruby is an ideal extension to the **TCC** batch language, especially if you need advanced string processing capabilities.

The Ruby language is not built into **TCC**, and must be obtained separately. The version supported by **TCC** is Ruby 1.8 (free from www.ruby-lang.org).

TCC automatically recognizes the presence of a Ruby interpreter on your system. If a suitable library is found, **TCC** checks to see if you are running a **.rb** file. If so, **TCC** passes the file to your Ruby interpreter for processing.

See also: the [@RUBY](#)^[458], [@REXX](#)^[457] and [@PERL](#)^[452] functions.

3.12.2.18 EXTPROC and SHEBANG Support

TCC offers an external processor 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. **TCC** will start the program and pass it the name of the **.CMD** file and any command line parameters 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
```

TCC 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 **TCC** 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.

TCC also provides **SHEBANG** support. It works identically to **EXTPROC**, but the first line begins with a **#!** .

Note that **EXTPROC** and **SHEBANG** only work with files with a **.CMD** extension, not **.BTM** or **.BAT**.

3.13 TCC (4NT) Internal Commands

TCC (formerly **4NT**) gives you instant access to more than 150 internal commands. (By contrast, Microsoft's [CMD.EXE](#)^[484] has fewer than 40 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](#)^[144]
- ▶ [Commands By Category](#)^[147]

Note: Remember that you can replace any internal command with an [ALIAS](#)^[154] or [plugin](#)^[293], or disable an internal command with [SETDOS /I](#)^[323].

3.13.1 Commands by Name

See also: [Internal Commands Listed by Category](#)^[147]

	Description
? ^[152]	Display list of internal commands or Prompt to execute a command
ACTIVATE ^[152]	Activate or set window state
ALIAS ^[154]	Define or display aliases
ASSOC ^[162]	Windows file associations
ATTRIB ^[163]	Change or display file attributes
BATCOMP ^[166]	Batch file compression
BDEBUGGER ^[166]	Batch file debugger
BEEP ^[173]	Beep the speaker
BREAK ^[174]	Define or display Ctrl-C state
BREAKPOINT ^[174]	Set a batch debugger breakpoint
CALL ^[175]	Call another batch file
CANCEL ^[176]	End batch file processing
CD ^[177]	Display or change directory
CDD ^[178]	Change drive and directory
CHCP ^[180]	Display or change code page
CHDIR ^[177]	Display or change directory
CLS ^[181]	Clear the display window
COLOR ^[181]	Change the display colors
COPY ^[182]	Copy files and/or directories
DATE ^[189]	Display or change date
DEBUGSTRING ^[190]	Send text to system debugger
DEFER ^[190]	Defer a command until batch file exit
DEL ^[190]	Delete files and/or directories
DELAY ^[194]	Wait for specified time
DESCRIBE ^[195]	Display or change descriptions
DETACH ^[197]	Start app detached
DIR ^[198]	Display files and/or directories
DIRHISTORY ^[208]	Display directory history list
DIRS ^[209]	Display directory stack

DO ^[210]	Create batch file loops
DRAWBOX ^[214]	Draw a box
DRAWHLINE ^[215]	Draw a horizontal line
DRAWVLINE ^[216]	Draw a vertical line
ECHO ^[217]	Echo a message
ECHOERR ^[217]	Echo a message to stderr ^[535]
ECHOS ^[219]	Echo a message with no CR/LF
ECHOSERR ^[219]	Echo with no CR/LF to stderr ^[535]
EJECTMEDIA ^[220]	Eject a removable drive
ENDLOCAL ^[220]	Restore from a SETLOCAL
ERASE ^[190]	Delete files and/or directories
ESET ^[222]	Edit variables or aliases
EVENTLOG ^[223]	Write Windows event log
EVENTMONITOR ^[224]	Monitor event log
EXCEPT ^[225]	Exclude files from a command
EXIT ^[227]	Exit TCC
FFIND ^[227]	Search for files or text
FIREWIREMONITOR ^[232]	Monitor FireWire devices
FOLDERMONITOR ^[232]	Monitor folders and/or files
FOR ^[234]	Repeat a command
FREE ^[241]	Display disk space
FTYPE ^[241]	Display or edit file types
FUNCTION ^[242]	Create or edit user functions
GLOBAL ^[245]	Run command in subdirectories
GOSUB ^[247]	Call batch subroutines
GOTO ^[248]	Branch in a batch file
HEAD ^[249]	Display beginning of file
HELP ^[251]	Help for internal commands
HISTORY ^[251]	Display or change history
IF ^[253]	Conditional command execution
IFF ^[254]	Conditional command execution
IFTP ^[255]	Open FTP connection
INKEY ^[257]	Get a single keystroke
INPUT ^[259]	Get a text string
JABBER ^[260]	Send an IM
KEYBD ^[261]	Set keyboard toggles
KEYS ^[261]	Enable or disable history list
KEYSTACK ^[262]	Send keystrokes to app
LIST ^[264]	Display content of files
LOADBTM ^[269]	Load batch file as .BTM
LOADMEDIA ^[269]	Close CD-ROM / DVD drive door
LOG ^[269]	Save log of commands
MD ^[271]	Create subdirectories
MEMORY ^[272]	Display memory statistics

MKDIR ^[271]	Create subdirectories
MKLNK ^[273]	Create NTFS hard or soft link
MOVE ^[274]	Move files or directories
MSGBOX ^[279]	Popup message box
NETMONITOR ^[282]	Monitor networks
ON ^[282]	Batch file error trapping
OPTION ^[284]	Configure the TCC console
OSD ^[286]	Display floating text
PATH ^[286]	Set or display PATH
PAUSE ^[287]	Wait for input
PDIR ^[288]	User-formatted DIR
PLAYAVI ^[291]	Display an .AVI file
PLAYSOUND ^[292]	Play a sound file
PLUGIN ^[293]	Load or unload plugin DLL
POPD ^[294]	Restore from directory stack
POSTMSG ^[294]	Send a message to a Window
PRINT ^[295]	Print a file
PRIORITY ^[295]	Set process priority
PROCESSMONITOR ^[296]	Monitor processes
PROMPT ^[297]	Change command line prompt
PUSHD ^[299]	Save directory to stack
QUERYBOX ^[300]	Popup input box
QUIT ^[301]	Exit batch file
RD ^[301]	Remove subdirectory
REBOOT ^[303]	Reboot system
RECYCLE ^[304]	Display or empty recycle bin
REXEC ^[308]	Remotely execute commands
REM ^[304]	Remark
REN ^[305]	Rename files or directories
RENAME ^[305]	Rename files or directories
RETURN ^[307]	Return from GOSUB
RMDIR ^[301]	Remove subdirectory
RSHELL ^[309]	Remotely execute commands
SCREEN ^[310]	Position cursor
SCRPUT ^[311]	Write directly to screen
SELECT ^[312]	Select files for a command
SENDMAIL ^[316]	Send email
SERVICEMONITOR ^[318]	Monitor Windows services
SERVICES ^[319]	Display, stop, or start system services
SET ^[319]	Set or display environment variables
SETDOS ^[323]	Set or display console options
SETLOCAL ^[326]	Save environment, aliases and functions
SHIFT ^[327]	Shift batch file parameters
SHORTCUT ^[328]	Create a Windows shortcut

SHRALIAS ^[329]	Share aliases
SMPP ^[330]	Simple message transfer
SNMP ^[330]	Send SNMP traps
SNPP ^[331]	Send message to pager
START ^[331]	Start a new session
SWITCH ^[335]	Batch file switch / case
SYNC ^[336]	Synchronize directories
TAIL ^[339]	Display end of file
TASKDIALOG ^[341]	Popup Vista task dialog
TASKEND ^[342]	End a task
TASKLIST ^[342]	Display Windows task list
TCFILTER ^[343]	Filter Take Command List View
TCTOOLBAR ^[343]	Edit Toolbar
TEE ^[344]	Pipe "tee-fitting"
TEXT ^[345]	Display text in batch file
TIME ^[347]	Set or display time
TIMER ^[347]	Stopwatch
TITLE ^[349]	Set window title
TOUCH ^[349]	Change file timestamps
TRANSIENT ^[352]	Toggle shell transient mode
TREE ^[352]	Display directory tree
TRUENAME ^[353]	Display true pathname
TYPE ^[354]	Display files
UNALIAS ^[355]	Remove aliases
UNFUNCTION ^[356]	Remove user-defined functions
UNSET ^[357]	Remove environment variable
USBMONITOR ^[358]	Monitor USB devices
VER ^[359]	Display version
VERIFY ^[360]	Display or set disk verification
VOL ^[360]	Display or set disk volume label
VSCRPUT ^[360]	Write text vertically
WHICH ^[361]	Display command information
WINDOW ^[362]	Window management
WMIQUERY ^[364]	WMI queries
Y ^[364]	Pipe "y-fitting"

3.13.2 Commands by Category

See also: [Internal Commands Listed by Name](#) ^[144]

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](#) ^[148]
- [Subdirectory management](#) ^[148]

- ▶ [Input and output](#) ^[149]
- ▶ [Window management commands](#) ^[149]
- ▶ [Commands primarily for use in or with batch files and aliases](#) ^[149]
- ▶ [Environment and path commands](#) ^[150]
- ▶ [System configuration and status](#) ^[150]
- ▶ [Monitoring commands](#) ^[151]
- ▶ [Other commands](#) ^[151]

File and directory management

	Description
ATTRIB ^[163]	Change or display file attributes
COPY ^[182]	Copy files and/or directories
DEL ^[190]	Delete files and/or directories
DESCRIBE ^[195]	Display or change descriptions
ERASE ^[190]	Delete files and/or directories
FFIND ^[227]	Search for files or text
HEAD ^[249]	Display beginning of file
IFTP ^[255]	Open FTP connection
LIST ^[264]	Display contents of files
MOVE ^[274]	Move files or subdirectories
RECYCLE ^[304]	Display or empty recycle bin
REN ^[305]	Rename files or directories
RENAME ^[305]	Rename files or directories
SELECT ^[312]	Select files for a command
SYNC ^[336]	Synchronize directories
TAIL ^[339]	Display end of file
TOUCH ^[349]	Change file dates/times
TREE ^[352]	Display directory tree
TRUENAME ^[353]	Display true pathname
TYPE ^[354]	Display files
Y ^[364]	Pipe "y-fitting"

Subdirectory management

	Description
CD ^[177]	Display or change directory
CDD ^[178]	Change drive and directory
CHDIR ^[177]	Display or change directory
DIR ^[198]	Display files and/or directories
DIRS ^[209]	Display directory stack
MD ^[277]	Create subdirectories
MKDIR ^[277]	Create subdirectories
MKLNK ^[273]	Create NTFS hard or soft link
PDIR ^[288]	User-formatted DIR
POPD ^[294]	Restore from directory stack
PUSHD ^[299]	Save directory to stack
RD ^[307]	Remove subdirectory
RMDIR ^[307]	Remove subdirectory

Input and output

	Description
DRAWBOX <small>[214]</small>	Draw a box
DRAWHLINE <small>[215]</small>	Draw a horizontal line
DRAWVLINE <small>[216]</small>	Draw a vertical line
ECHO <small>[217]</small>	Echo a message
ECHOERR <small>[217]</small>	Echo a message to stderr
ECHOS <small>[219]</small>	Echo a message with no CR/LF
ECHOSERR <small>[219]</small>	Echo with no CR/LF to stderr
INKEY <small>[257]</small>	Get a keystroke
INPUT <small>[259]</small>	Get an input line
KEYSTACK <small>[262]</small>	Send keystrokes to app
MSGBOX <small>[279]</small>	Popup message box
OSD <small>[286]</small>	Display floating text
PLAYAVI <small>[291]</small>	Play an .AVI file
PLAYSOUND <small>[292]</small>	Play a sound file
PRINT <small>[295]</small>	Print a file
QUERYBOX <small>[300]</small>	Popup input box
SCREEN <small>[310]</small>	Position cursor
SCRPUT <small>[311]</small>	Write directly to screen
SENDMAIL <small>[316]</small>	Send email
SMPP <small>[330]</small>	Send SMS message
SNMP <small>[330]</small>	Send SNMP trap
SNPP <small>[331]</small>	Send message to pager
TASKDIALOG <small>[341]</small>	Popup Vista task dialog
VSCRPUT <small>[360]</small>	Write text vertically

Window management commands

	Description
ACTIVATE <small>[152]</small>	Activate or set window state
POSTMSG <small>[294]</small>	Send a message to a Window
TITLE <small>[349]</small>	Set window title
WINDOW <small>[362]</small>	Window management

Commands primarily for use in or with batch files and aliases
 (some work only in batch files; see the individual commands for details)

	Description
ALIAS <small>[154]</small>	Define or display aliases
BATCOMP <small>[166]</small>	Batch file compression
BDEBUGGER <small>[166]</small>	Batch file debugger
BEEP <small>[173]</small>	Beep the speaker
BREAKPOINT <small>[174]</small>	Set a batch debugger breakpoint

CALL ^[175]	Call another batch file
CANCEL ^[176]	End batch file processing
DEBUGSTRING ^[190]	Send text to system debugger
DEFER ^[190]	Defer a command until the batch file exits
DELAY ^[194]	Wait for specified time
DO ^[210]	Batch file looping
ENDLOCAL ^[220]	Restore a SETLOCAL
EJECTMEDIA ^[220]	Eject a removable drive
FOR ^[234]	Repeat a command
FUNCTION ^[242]	Create or edit user functions
GLOBAL ^[245]	Run command in subdirectories
GOSUB ^[247]	Call batch subroutines
GOTO ^[248]	Go to a batch file label
IF ^[253]	Conditional command execution
IFF ^[254]	Conditional command execution
JABBER ^[260]	Send an IM
LOADBTM ^[269]	Load batch files as .BTM
LOADMEDIA ^[269]	Close CD-ROM / DVD drive door
ON ^[282]	Batch file error trapping
PAUSE ^[287]	Wait for input
QUIT ^[301]	Exit batch file
REM ^[304]	Remark
RETURN ^[307]	Return from GOSUB
SETLOCAL ^[326]	Save environment, aliases, and functions
SHIFT ^[327]	Shift batch file parameters
SWITCH ^[335]	Batch file switch / case
TEXT ^[345]	Display text in batch file
TRANSIENT ^[352]	Toggle shell transient mode
UNALIAS ^[355]	Remove aliases
UNFUNCTION ^[356]	Remove user-defined functions

Environment and path commands

	Description
ESET ^[222]	Edit variables or aliases
PATH ^[286]	Set or display PATH
SET ^[319]	Set or display environment variables
UNSET ^[357]	Remove environment variables

System configuration and status

	Description
ASSOC ^[162]	Windows file associations
BREAK ^[174]	Define or display Ctrl-C state
CHCP ^[180]	Display or change code page
CLS ^[181]	Clear the display window
COLOR ^[181]	Change the display colors

DATE <small>[189]</small>	Display or change date
DIRHISTORY <small>[208]</small>	Display directory history list
EVENTLOG <small>[223]</small>	Write to Windows event log
FREE <small>[241]</small>	Display disk space
FTYPE <small>[241]</small>	Display or edit file types
HISTORY <small>[251]</small>	Display or change history
KEYBD <small>[261]</small>	Set keyboard toggles
KEYS <small>[261]</small>	Enable or disable history list
LOG <small>[269]</small>	Save log of commands
MEMORY <small>[272]</small>	Display memory statistics
OPTION <small>[284]</small>	Configure the TCC console
PLUGIN <small>[293]</small>	Load or unload plugin DLL
PROMPT <small>[297]</small>	Change command line prompt
REBOOT <small>[303]</small>	Reboot system
SETDOS <small>[323]</small>	Internal options
SERVICES <small>[319]</small>	Display, stop, or start services
SHORTCUT <small>[328]</small>	Create a Windows shortcut
TASKEND <small>[342]</small>	End a task
TASKLIST <small>[342]</small>	Display Windows task list
TCFILTER <small>[343]</small>	Filter Take Command List View
TCTOOLBAR <small>[343]</small>	Edit Take Command toolbar
TIME <small>[347]</small>	Set or display time
VERIFY <small>[360]</small>	Display or set disk verification
VER <small>[359]</small>	Display version
VOL <small>[360]</small>	Display or set disk volume label

Monitoring commands

	Description
EVENTMONITOR <small>[224]</small>	Monitor event log
FIREWIREMONITOR <small>[232]</small>	Monitor FireWire devices
FOLDERMONITOR <small>[232]</small>	Monitor folders and/or files
NETMONITOR <small>[281]</small>	Monitor network connections
PROCESSMONITOR <small>[296]</small>	Monitor processes
SERVICEMONITOR <small>[318]</small>	Monitor Windows services
USBMONITOR <small>[358]</small>	Monitor USB devices

Other commands

	Description
? <small>[152]</small>	Display list of internal commands, or prompt to execute a command
DETACH <small>[197]</small>	Start app detached
EXCEPT <small>[225]</small>	Exclude files from a command
EXIT <small>[227]</small>	Exit TCC

HELP ^[251]	TCC help
SHRALIAS ^[329]	Share aliases & functions
START ^[331]	Start a new session
REXEC ^[308]	Remotely execute command
RSHELL ^[309]	Remotely execute command
TEE ^[344]	Pipe "tee-fitting"
TIMER ^[347]	Stopwatch
WHICH ^[361]	Display command information

3.13.3 ?

Purpose: Display a list of internal and plugin 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 and plugin commands. For help with any individual command, see the [HELP](#) ^[251] command. If you have disabled a command with [SETDOS /I](#) ^[323], it will not appear in the list.
2. The second function of ? is to prompt the user before executing a specific command 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 command line will be executed. If the user presses **N** or **n**, it will be ignored.

Example

```
? "Load the network" call netstart.btm
```

When this command is executed, you will see the prompt

```
Load the network (Y/N)?
```

If you answer Y, the [CALL](#) ^[175] command will be executed:

3.13.4 ACTIVATE

Purpose: Activate a window, set its state, or change its title.

Format: ACTIVATE [/R] "title" [MAX | MIN | RESTORE | CLOSE | ENABLE | DISABLE | TOPMOST | NOTOPMOST | TOP | BOTTOM | HIDE | /POS=left,top,width,height | /TRANS=n | TRAY | "newtitle"]

title	Current title of the window to be activated
left	New location of the left border of the window, in pixels
top	New location of the top border of the window, in pixels
width	New width of the window, in pixels
height	New height of the window, in pixels
newtitle	New title for window

/R(estore original window)

See also: [START](#)^[331], [TITLE](#)^[349], and [WINDOW](#)^[362].

Usage:

[ACTIVATE](#)^[152] activates, and optionally modifies, another session's window. It is not intended to modify the characteristics of the current **TCC** session (use [TITLE](#)^[349] or [WINDOW](#)^[362] for that purpose).

Title specifies the name of the target window to be activated. You can use [wildcards](#)^[77], including extended wildcards, in **title**. This is useful with applications that change their window title to reflect the file currently in use. **Title** must be enclosed in quotes.

Each execution of ACTIVATE allows you to modify one property of the target window. To perform multiple operations, use multiple ACTIVATE commands.

The options are:

MAX	Expands the window to its maximum size and activates it.
MIN	Reduces the window to an icon.
RESTORE	Activates the window at its default size and location.
CLOSE	Sends a "close" message to close the window.
ENABLE	Enable mouse and keyboard input
DISABLE	Disable mouse and keyboard input
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.
TRANS	Transparency level, where n=0 (invisible) to 255 (opaque) (does not work for console windows)
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).
TRAY	Move the specified window to the system tray.
"newtitle"	Changes the window title.

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 specify **newtitle**, it must be enclosed in double quotes (which will not appear as part of the title text).

ACTIVATE is often used before [KEYSTACK](#)^[262] 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, 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 you cannot make another window active except when the window which issues the command is itself active already. This is a Windows feature which helps to prevent windows which are not in the foreground from grabbing input intended for other windows.

Example

The example below first maximizes, and then renames the window originally called "Take Command":

```
activate "Take Command" max
activate "Take Command" "Take Command is Great!"
```

3.13.5 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: Display mode:
 ALIAS [/G /L /P] [wildname]

Definition mode:
 ALIAS /R [file...] | name[=]value

file	One or more input files to read alias definitions from
wildname	Name of alias whose definition is to be displayed (may contain * and ? wildcards)
name	Name for an alias, or for the key to execute the alias
value	Text to be substituted for the alias name or key

[/G\(lobal\)](#)^[162]
[/L\(ocal\)](#)^[162]

[/P\(ause\)](#)^[162]
[/R\(ead file\)](#)^[162]

See also: [UNALIAS](#)^[355], [ESET](#)^[222], and [Aliases](#)^[128].

Usage:

- ▶ [Overview](#)^[154]
- ▶ [Displaying Aliases](#)^[155]
- ▶ [Multiple Commands and Special Characters in Aliases](#)^[155]
- ▶ [Nested Aliases](#)^[156]
- ▶ [Temporarily Disabling Aliases](#)^[156]
- ▶ [Partial \(Abbreviated\) Alias Names](#)^[157]
- ▶ [Keystroke Aliases](#)^[157]
- ▶ [Saving and Reloading Your Aliases](#)^[159]
- ▶ [Alias Parameters](#)^[159]
- ▶ [Expanding Aliases at the Prompt](#)^[160]
- ▶ [Local and Global Aliases](#)^[160]
- ▶ [Retaining Global Aliases with SHRALIAS](#)^[161]
- ▶ [The PRE_INPUT, PRE_EXEC, and POST_EXEC Aliases](#)^[161]
- ▶ [The UNKNOWN_CMD Alias](#)^[161]
- ▶ [Warnings](#)^[162]

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 [DIR](#)^[198] /W, you could use the command:

```
alias d = dir /w
```

Now when you type a single **d** as a command, it will be translated into a **DIR /W** command.

If an ALIAS command specifies a **value**, and there was an alias already assigned to **name**, the old alias value is discarded.

If you define aliases for commonly used application programs, you can often remove the directories they're stored in from the [PATH](#)^[368]. For example, if you use Microsoft Word and had the **C:\WINWORD** directory in your path, you could define the following alias:

```
alias ww = c:\winword\winword.exe
```

With this alias defined, you can probably remove **C:\WINWORD** from your path. Word will now load more quickly than it would if **TCC** had to search the [PATH](#)^[368] for it. In addition, [PATH](#)^[368] 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](#)^[368] 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](#)^[368], you will need to define aliases for all the executable programs you commonly use which are stored in that directory.

TCC also supports [Directory Aliases](#)^[76], a shorthand way of specifying pathnames.

Aliases are stored in memory, and are not saved automatically when you turn off your computer or end your current **TCC** session. See below for information on saving and reloading your aliases.

Displaying Aliases

If you want to see a list of all currently defined aliases, type:

```
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:

```
alias list
```

You can also view the definitions for all aliases matching a specific pattern by specifying a single parameter containing wildcards (* or ?). For example:

```
alias *win*
```

will display all aliases containing the string **win**.

You can use the [/P](#)^[157] option to control display scrolling when displaying aliases.

Multiple Commands and Special Characters in Aliases

An alias can represent more than one command. For example:

```
alias letters = `cd \letters & tedit`
```

This alias creates a new command called **LETTERS**. The command first uses [CD](#)^[177] to change to a subdirectory called **LETTERS** of the directory current at the time of its execution, and then runs a program called **TEDIT**.

Aliases make extensive use of the [command separator](#)^[120], and the [parameter character](#)^[51], and may also use the [escape character](#)^[124].

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](#)^[46].

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 variable functions, or redirection, or piping. If you do not use back quotes, parameters, variables and functions are evaluated, and redirection or piping performed during the alias definition, and only the first command becomes part of the alias, the remaining ones are performed immediately. The back quotes prevent this premature expansion. 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](#)^[253] or [IFF](#)^[254] 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 in the current directory, and executes **Word** located in the **E:\WINWORD**. The second alias changes directories with the [PUSHD](#)^[299] command, runs the **WP** alias, and then returns to the original directory with the [POPD](#)^[294] command:

```
alias wp = e:\winword\winword.exe
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 **TCC** 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](#)^[198] command with the **/2** (two column) and **/P** (pause at the end of each page) options:

```
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 **TCC** would repeatedly re invoke the **DIR** alias, rather than running the [DIR](#)^[198] command. This would cause an "Alias loop" or "Command line too long" error. The asterisk forces interpretation of the second [DIR](#)^[198] 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](#)^[195] command will run anytime you type DESCRIBE. But two simple aliases will

give you access to both the *DESCRIBE.EXE* program and the [DESCRIBE](#)^[195] command:

```
alias describe = c:\winutil\describe.exe
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](#)^[195] command. The second line defines **FILEDESC** as a new name for the internal [DESCRIBE](#)^[195] command. The asterisk is needed in the second command to indicate that the following word means the internal command [DESCRIBE](#)^[195], 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 **TCC** to go straight to checking for an internal command, external command, or batch file.

You can prevent alias expansion by using an asterisk before a command that you enter at the command line or in a batch file. This can be useful when you want to be sure you are running the 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](#)^[198] 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 for that command.

You can disable aliases temporarily with the [SETDOS /X](#)^[323] 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:

```
alias wher*eis = dir /s /p
```

The new command, **WHEREIS**, can be invoked as **WHER**, **WHERE**, **WHEREI**, or **WHEREIS**. Now if you type:

```
where myfile.txt
```

The **WHEREIS** alias will be expanded to the command:

```
dir /s /p myfile.txt
```

Keystroke Aliases

There are two kinds of keystroke aliases: [insert-only](#)^[157] and [autoexecute](#)^[158].

Insert-only Keystroke Aliases

Assignment: To assign an insert-only alias to a keystroke, use the key name on the left side of the equal sign, preceded by one at sign @, and the value of the alias on the right side of the equal sign:

```
alias @key=value
```

Operation: When you press the key to which you assigned an insert-only alias, **TCC** displays and inserts the alias value in the current command line, at the current cursor position. If your command line

editing mode is overwrite, and the cursor is not at the end of the line, the alias value will overwrite part of the command line. You can continue to edit the command line, e.g., adding other parameters to the command. You must press **Enter** to execute the command.

Examples:

To assign the command **DIR /W** to the **F4** key, type:

```
alias @F4 = dir /w
```

To use it, press **F4** at the command prompt, and **DIR /w** will be placed on the command line for you. You can type additional parameters if you wish, and press **Enter** to execute the command. With the example alias, you can define the files that you want to display after pressing **F4** and before pressing **Enter** to execute the command.

You can also define a keystroke alias to insert a frequently used string into the middle of a command, e.g.,

```
alias @shift-F4 =%@expand[
```

which specific example can assist in processing wildcards for a program without such a feature.

Autoexecute Keystroke Aliases

Assignment: To assign an autoexecute alias to a keystroke, use the key name on the left side of the equal sign, preceded by two at signs @@, and the value of the alias on the right side of the equal sign:

```
alias @@key=value
```

Operation: When you press the key to which you assigned an autoexecute alias, **TCC** inserts the alias value in the current command line, at the current cursor position. If your command line editing mode is overwrite, and the cursor is not at the end of the line, the alias value will overwrite part of the command line. After the insertion/overwrite the command line is automatically executed.

Example: This command will assign an alias to the **F11** key that uses the [CDD](#)^[178] command to take you back to the previous default directory:

```
alias @@f11 = cdd -
```

Special Considerations for Keystroke Aliases

When you define keystroke aliases, the assignments will only be in effect at the command line, not inside application programs or batch files.

To insure that a keystroke alias, esp. an autoexecute one, is on the command line by itself, use the character defined by the [EraseLine](#)^[31] option (by default, the **Esc** key, best represented as **%=e**) as the first character of the alias value.

To force a visible indication that an autoexecute keystroke alias was used, include a descriptive [ECHO](#)^[217] command in the alias value.

Be careful not to assign aliases to keys that are already used at the command line (e.g., **F1** for [HELP](#)^[25]). 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 default use with the [NormalKey](#)^[31] or [NormalEditKey](#)^[35] options.

The **value** of an alias, including a keystroke alias, may contain only characters. It cannot contain

representations of keys such as **F1** .. **F12**, **Home**, etc.

See [Keys and Key Names](#)^[515] for a complete listing of key names and a description of the key name format.

Saving and Reloading Your Aliases

You can save your aliases to a file:

```
alias > alias.lst
```

You can then reload all the alias definitions in the file the next time you start up with the command:

```
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 **TCC** 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 **TCC**.

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.

To remove an alias, use the [UNALIAS](#)^[355] command.

Alias Parameters

Aliases can use command line parameters or parameters like those in batch files. The command line parameters are numbered from %0 to %511. (%0 contains the alias name.) You can use double quotes to pass spaces, tabs, commas, and other special characters in an alias parameter; see [Parameter Quoting](#)^[134] for details. (Alias examples in this section assume the **TCC** default of ParameterChar=\$.)

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. **TCC** interprets it to mean "the entire command line, from parameter **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 parameter **%-n\$** means "the command line from parameter 1 to **n** - 1".

The special parameter **%#** contains the number of command line parameters.

For example, the following alias will change directories, perform a command, and return to the original directory:

```
alias in `pushd %1 & %2$ & popd`
```

When this alias is invoked as:

```
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](#)^[254] command to redefine the defaults for [SET](#)^[319]. It should be entered on one line:

```
alias set = `iff %# == 0 then & *set /p & else & *set %$ & endiff`
```

This modifies the [SET](#)^[319] command so that if [SET](#)^[319] is entered with no parameters, it is replaced by SET /P (pause after displaying each page), but if [SET](#)^[319] is followed by a parameter, it behaves normally. Note the use of asterisks (*set) to prevent alias loops.

If an alias uses parameters, command line parameters will be deleted up to and including the highest referenced parameter. For example, if an alias refers only to %1 and %4, then the first and fourth parameters will be used, the second and third parameters will be discarded, and any additional parameters 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 parameters will be appended to the expanded command. A convenient way to prevent unwanted command line parameters 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:

```
alias calc = `echo The answer is: %@eval[%$]`
```

Now, if you enter:

```
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

Aliases can be stored in either a local or global list. The selection is made during **TCC** startup, using the **/L** or **/LA** [START](#)^[331] or [startup options](#)^[19], or by the [Local Aliases](#)^[47] configuration option, or interactively with the **ALIAS/G** and **ALIAS/L** options. The global alias list is limited to 128K characters; the local alias list is limited only by memory size.

With a local alias list, any changes made to the aliases will only affect the current copy of **TCC**. They will not be visible in other shells or other sessions.

With a global alias list, all copies of **TCC**, which are started with global alias list 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 **TCC**.

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.

When you use [SETLOCAL](#)^[326] / [ENDLOCAL](#)^[220] inside a batch file, changes in alias definitions are restored by the [ENDLOCAL](#)^[220]. However, if the session uses the global alias list, any concurrent sessions also using the global alias list are affected.

Whenever you start a second copy of **TCC** 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 **TCC** you can share the aliases among all running copies of **TCC**. When you close all **TCC** sessions, the memory for the global alias list is released, and a new, empty alias list is created the next time you start **TCC**.

If you want the alias list to be retained in memory even when no **TCC** session is running, you need to execute the [SHRALIAS](#)^[329] 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](#)^[329] from your [TCSTART](#)^[22] file.

[SHRALIAS](#)^[329] 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](#)^[159] above.

The PRE_INPUT, PRE_EXEC, and POST_EXEC Aliases

When at the command prompt (i.e., not executing a batch file), **TCC** will look for (and execute them if found) the following aliases:

PRE_INPUT - executed immediately before accepting input for a new command line.

PRE_EXEC - executed immediately after a command line is entered (before any expansion or redirection).

POST_EXEC - executed immediately after returning from a command and before displaying the prompt.

None of these aliases will be passed any arguments.

The UNKNOWN_CMD Alias

If you create an alias with the name **UNKNOWN_CMD**, it will be executed any time **TCC** would normally issue the "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 just display the command that caused the error:

```
alias unknown_cmd `echo Error in command "%$" `
```

If the **UNKNOWN_CMD** alias contains an unknown command, it will call itself repeatedly. If this occurs, **TCC** will loop up to 10 times, then display the **UNKNOWN_CMD loop** error.

Warnings

When you define an alias in the command line (i.e., without using the [/R](#)^[161] option), variables and functions not protected by back quotes or doubled % signs are immediately evaluated, and the result becomes part of the alias value.

Syntax errors in an alias are not detected until the alias is executed.

Options:

- /G** Switch from a local to a global alias list.
- /L** Switch from a global to a local alias list.
- /P** 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](#)^[102]).
- /R** 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 ***name*** and ***value***. Back quotes are not required around the value. Variables and functions referenced in the definitions remain in the definitions, to be evaluated each time the alias is executed. 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:

```
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 the definition, except the last, is terminated with an [escape character](#)^[124].

ALIAS /R will read from [stdin](#)^[535] if no filename is specified and input is redirected:

```
alias /r <
```

3.13.6 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\(ause\)](#)^[163]

[/R\(ead\)](#)^[163]

See also: [FTYPE](#)^[241], and [Executable Extensions](#)^[91].

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](#)^[506]. 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](#)^[241] 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** 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](#)^[102].
- /R** 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:

```
assoc /r assoc1.lst assoc2.lst
```

You can insert comments in the file by prefixing the line with a colon (:).

ASSOC /R will read from [stdin](#)^[535] if no filename is specified and input is redirected.

3.13.7 ATTRIB

Purpose: Change or view file and subdirectory attributes.

Format: ATTRIB [/A:[-+]rhsa] /D /E /I "text" /P /Q /S[n]] [+|- [AHIORST]] [@file] files ...

files A file, directory, or list of files or directories to process.

@file A text file containing the names of the files to process, one per line (see [@file lists](#)^[90] for details).

[/A:](#) (Attribute select)^[165]

[/P\(ause\)](#)^[165]

[/D\(irectories\)](#)^[165] [/Q\(quiet\)](#)^[165]
[/E \(No error messages\)](#)^[165] [/S\(ubdirectories\)](#)^[165]
[/!"text" \(match description\)](#)^[165]

Attribute flags:

Clear	Set	Attribute affected
-A	+A	archive
-H	+H	hidden
-I	+I	not content indexed
-O	+O	offline
-R	+R	read-only
-S	+S	system
-T	+T	temporary

File Selection

Supports [attribute switches](#)^[86], extended [wildcards](#)^[77], [ranges](#)^[80], [multiple file names](#)^[87], and [include lists](#)^[88]. Use wildcards with caution on LFN volumes; see [LFN File Searches](#)^[89] for details.

Usage:

Every file and subdirectory has attributes that can be turned on (set) or turned off (cleared): **Archive**, **Hidden**, **Not content indexed**, **Offline**, **Read-only**, **System**, and **Temporary**. For details on the meaning of each attribute, see [File Attributes](#)^[494].

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 **[+]-[AHIORST]** part of the format), or with the [DIR /T](#)^[198] 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*:

```
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*:

```
attrib +a *.txt +s -a test.com
```

When you use ATTRIB on an LFN drive, you must double 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.

NTFS also supports **D** (subdirectory), **V** (volume label), **E** (encrypted), **C** (compressed), **J** or **L** (junction / symbolic link) and **P** (sparse file) attributes. These attributes will be displayed by ATTRIB, but cannot be altered; they are designed to be controlled only by Windows.

ATTRIB will ignore underlines in the new attribute (the **[+]-[ADHIORST]** part of the command). For example, ATTRIB sees these 2 commands as identical:

```
attrib +a filename
```

```
attrib +__A_ filename
```

This allows you to use a string of attributes from either the [@ATTRIB](#)^[409] 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*:

```
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):

```
attrib -arhs abc* & attrib +%@attrib[file1] abc*
```

Options:

/A: Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[86] 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:

```
attrib /a:h -a *
```

Do not use **/A:** with [@file lists](#)^[90] for details.

/D If you use the **/D** option, ATTRIB will modify the attributes of directories in addition to files (yes, you can have a hidden directory):

```
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 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" Select files by matching text in their descriptions. The text can include [wildcards](#)^[77] and extended wildcards. The search text must be enclosed in double quotes, and must immediately follow the **/I**, 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](#)^[90] for details

/P 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](#)^[102]

/Q This option turns off ATTRIB's normal screen output. It is most useful in batch files.

/S 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](#)^[90] for details.

If you specify a number after the /S, ATTRIB will limit the subdirectory recursion to that number. For example, if you have a directory tree "a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.

3.13.8 BATCOMP

Purpose: Compress and optionally encrypt batch files. See [Batch File Compression](#)^[141] 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\(ncryption key\)](#)^[166]

[/K \(remove comments\)](#)^[166]

[/O\(verwrite\)](#)^[166]

[/Q\(quiet\)](#)^[166]

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 output file will not have one. In other words there is no BTM default if you just give a file name. 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 **TCC**. 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** Specify a key (string) to be used for encryption.
- /K** 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** Forces overwriting of any existing **OutputFile**.
- /Q** Suppresses all progress reports ([stdout](#)^[535]). Errors ([stderr](#)^[535]) are still shown.

3.13.9 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: BDEBUGGER [/A /B /C /E /F /I /K /N /W] batchfilename [parameters]

batchfilename Full name of the batch file to debug.
parameters parameters for the batch file

[/A \(list window\)](#) ^{|173|} [/I \(inactive\)](#) ^{|173|}
[/B \(atch variable window\)](#) ^{|173|} [/K \(eep debugger window\)](#) ^{|173|}
[/C \(reate new batch file\)](#) ^{|173|} [/N \(do Not hide the debugger window\)](#)
[/E \(nvironment variable window\)](#) ^{|173|} [/W \(atch window\)](#) ^{|173|}
[/F \(user-defined Functions window\)](#) ^{|173|}

Usage:

BDEBUGGER opens a special window in which a batch file can be examined, modified, and executed.

The debugger window includes a slider control on the lower left corner of the status bar to control the transparency level of the debugger window. If the debugger window is at least partially transparent, it will not be hidden while individual commands in the batch file are being executed (see the /N option).

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 Into	Execute the current line.
Step Over	Execute the current line but turn off the debugger for the duration of a CALL or GOSUB.
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 TCC (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 only set a breakpoint on an executable line (i.e., not on a blank line, comment, label, etc.),

You can get help for the currently selected (highlighted) command / variable / function by pressing Ctrl-F1, or right-clicking the mouse and selecting **Help** from the context menu.

You can change the line to be executed next when in debugging mode with either the "GOTO" dialog or by moving the caret to the line and either right clicking & selecting "Jump to This Line" or by pressing Ctrl-Shift-F11. Note that if you attempt to jump into or out of a DO loop or IFF block, bad

things will happen!

The debugger's Watch window allows you to monitor environment variables or to pause execution when a specified condition is met. The Watch window appears at the bottom of the debugger window. Enter the variable name or expression in the left column; the debugger will automatically display the current value in the right column. You can also add a variable to the Watch window by selecting it in the main debugger window, then clicking the right mouse button and selecting "Add to Watch". If the string in the left column is a single argument, it is assumed to be a variable name. Otherwise, it is assumed to be an expression. Expressions can be anything that IF can evaluate; for example:

```
%i = 3
ERRORLEVEL GT 12
```

Note that expressions require variable names to be prefixed by a %. If you're entering a single variable argument to monitor, do not use a %.

When the value of a monitored variable changes, the Watch window will change its background color (to a light red).

Alt-F11 will invoke the Evaluate Expression dialog. Ctrl-F9 will invoke the Evaluate Expression dialog for the current selection, or if no text is selected, for the current line.

You can open popup windows to display and/or modify aliases, batch variables, environment variables, and user functions. These windows will "dock" to the outer edges of the debugger window and will move with the debugger window. 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 environment variables window displays any variables changed during debugging in a different color. (The default is red; you can change it by adding a ChangeRGB=... line to BATCH.BCP with the appropriate RGB value.) You can specify variables to exclude from the environment variable window with the DebugVariableExclude variable. For example, to suppress the display of the processor and user variables:

```
set DebugVariableExclude=proc*;user*
```

Note that this option doesn't affect the existence of the variables, just whether they're displayed in the environment variable window.

If you press **Ctrl-C** or **Ctrl-Break** while debugging, you will see the prompt:

```
Cancel batch job filename (Y/N/A/D) :
```

Pressing **D** will return you to single-step mode in the debugger. (This allows you to interrupt a **run-to-breakpoint** without terminating the debugger and batch file.)

The default settings for the BDEBUGGER (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 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](#)^[169]
- ▶ [View commands](#)^[170]
- ▶ [Edit commands](#)^[170]
- ▶ [Mark / Clipboard commands](#)^[172]
- ▶ [Search commands](#)^[172]
- ▶ [File commands](#)^[172]
- ▶ [Bookmark commands](#)^[172]
- ▶ [Breakpoint commands](#)^[173]
- ▶ [Expression evaluation commands](#)^[173]

• 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: |

- 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.
Ctrl-Shift-Home	(18)	In addition to the caret movement this command will also extend the current selection to the new position.
Ctrl-End	(19)	This command will move the caret to the end of the text. When the caret is already at this location nothing happens.
Ctrl-Shift-End	(20)	In addition to the caret movement this command will also extend the current selection to the new position.
Ctrl-Tab	(21)	This command will move the caret to the previous tab-stop position. When the caret is located at the start of the line nothing happens.
Ctrl-Shift-Tab	(22)	In addition to the caret movement, this command will also extend the current selection to the new position.
PgUp	(23)	This command will move the caret one view up when it is located on the top line currently in the view. When the caret is not located on the top line of the view, it will be moved there.
Shift-PgUp	(24)	In addition to the caret movement this command will also extend the current selection to the new position.
PgDn	(25)	This command will move the caret one view down when it is located on the bottom line currently in the view. When the caret is not located on the bottom line of the view, it will be moved there.
Shift-PgDn	(26)	In addition to the caret movement, this command will also extend the current selection to the new position.

• 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 undone. You can re-do any number of changes up to the number of changes undone.
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 output is formatted using the user's default format.
Ins	(123) This command will toggle the current editing mode between overwrite

and insert.

- **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 <i>goto</i> dialog.
Ctrl-F	(301)	This command will show the <i>find</i> dialog.
Ctrl-H	(302)	This command will show the <i>replace</i> 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

Shift-F2 (255) searched starting at the first line.
 (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.

- **Expression evaluation commands**

Alt-F11 Invoke the Evaluate Expression dialog.
Ctrl-F9 Invoke the Evaluate Expression dialog for the current selection. If no text is selected, evaluate the current line.

Options:

/A Start with the alias window open.
/B Start with the batch variables window open.
/C If the specified batch file doesn't exist, create it without prompting.
/E Start with the environment variables window open.
/F Start with the user-defined functions window open.
/I Load the batch file, but do not start execution
/K Keep the batch debugger window open and switch to edit mode after the batch file exits.
/N By default, the debugger window hides itself while the debugged batch file is executing. **/N** keeps that window visible in the background (the **TCC** window is brought to the foreground).
/W Open the Watch window when the debugger starts.

3.13.10 BEEP

Purpose: Beep the speaker or play simple music.

Format: BEEP [frequency duration ...] [asterisk | exclamation | hand | question | ok]

frequency	The beep frequency in Hertz (cycles per second).
duration	The beep length in 1/18th second intervals.
asterisk	Plays the system default "asterisk" sound.
exclamation	Plays the system default "exclamation" sound.
hand	Plays the system default "hand" sound.
question	Plays the system default "question" sound.
ok	Plays the system default "ok" sound.

See also: the [Length](#)^[51] and [Frequency](#)^[51] configuration options.

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).

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

3.13.11 BREAK

Purpose: Enable or disable Ctrl-C and Ctrl-Break

Format: BREAK [ON | OFF]

Usage:

BREAK OFF will disable all **Ctrl-C** and **Ctrl-Break** handling in **TCC** (though not necessarily in child processes). In CMD.EXE, BREAK OFF doesn't actually do anything, so setting it in **TCC** will introduce a possible incompatibility with existing batch files.

3.13.12 BREAKPOINT

Purpose: Set a batch debugger breakpoint on the current line

Format: BREAKPOINT

Usage:

If the batch debugger is active, BREAKPOINT sets a breakpoint on the current line, stopping a "Step Out" sequence. If the batch debugger is not active, BREAKPOINT is ignored.

3.13.13 CALL

Purpose: Execute one batch file from within another.

Format: CALL file | :label [p1 [p2 ...]]

file	The batch file to execute.
:label	A label ^[530] in the current batch file.
p1, p2,...	Parameters for the batch file or subroutine

See also: [CANCEL](#) ^[176] and [QUIT](#) ^[301].

Usage:**Calling other batch files**

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 the batch file *A.BTM* uses **CALL B**, and *B.BTM* chains to the batch file *C.BTM*, on exit from *C.BTM* (without executing a [CANCEL](#) ^[176] command) processing of batch file *A.BTM* is resumed as if it had used **CALL C**.

File *A.BTM*:

```
...
call b
echo xxx
```

File *B.BTM*:

```
...
c
```

File *C.BTM*:

```
...
quit
```

In the example above, after execution of the [QUIT](#) ^[301] command in *C.BTM* the **ECHO xxx** command in *A.BTM* is executed next.

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](#)^[301] command), or it should terminate batch file processing with [CANCEL](#)^[176]. 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.

Calling a label

To provide compatibility with CMD.EXE, which does not support the [GOSUB](#)^[247] command for subroutines in the same batch file, you may create a subroutine starting with a [label](#)^[530] and terminated by any of the following:

- the end of the batch file
- [QUIT](#)^[301]
- [EXIT](#)^[227]
- [CANCEL](#)^[176]

Note that the last two do NOT return control to the CALL command. Do not use the [RETURN](#)^[307] command!

Parameters passed to the subroutine are accessible as %1, %2, etc., in the same manner as in a batch file.

Exit code

CALL returns an exit code which matches the batch file return code. You can test this exit code with the [%_?](#)^[380] or [%?](#)^[380] environment variables, and use it with [conditional commands](#)^[120] (&& and ||).

See also [GOSUB](#)^[247] and [user-defined functions](#)^[242].

3.13.14 CANCEL

Purpose: Terminate batch file processing.

Format: CANCEL [value]

value The numeric exit code to return to **TCC**.

See also: [CALL](#)^[175] and [QUIT](#)^[301].

Usage:

The CANCEL command ends all batch file processing, regardless of the batch file nesting level. Use [QUIT](#)^[301] 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](#) ^[253] command, and the [%?](#) ^[380] 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).

3.13.15 CD / CHDIR

Purpose: Display or change the current directory.

Format: CD [/D /N /X] [path | -]

path The directory to change to, optionally including a drive letter

[/D\(rive\)](#) ^[178]

[/X\(exclude\)](#) ^[178]

[/N\(o extended search\)](#) ^[178]

See also: [CDD](#) ^[178], [MD](#) ^[271], [PUSHD](#) ^[299], [RD](#) ^[301], and [Directory Navigation](#) ^[71].

Internet: Can be used with [FTP Servers](#) ^[93].

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:\FINANCEMYFILES*:

```
[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 (unless you use the [/D](#) ^[178] option). 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 [/D](#) ^[178] option, or preferably the [CDD](#) ^[178] command to change the current drive and directory at the same time.

If **path** contains white space or special characters (which is valid only for an [LFN](#) ^[531] drive), you must enclose it in double quotes .

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](#) ^[118]). 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 parameter 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](#) ^[72] and the [extended directory search](#) ^[73] database in order to find a matching directory and switch to it. You can disable this default extended search with [/N](#) ^[178]. You can also use wildcards in **path** to force an extended directory search. Read the section on [Directory Navigation](#) ^[71] for complete details on these and other directory navigation features.

CD saves the current directory before changing to a new directory. You can switch back to the previous directory by entering CD -. (There must be a space between the CD command and the hyphen.) You can switch back and forth between two directories by repeatedly entering CD -. The saved directory is the same for both the CD and [CDD](#)^[178] commands. Drive changes and [automatic directory changes](#)^[76] 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](#)^[118], 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](#)^[93] opened with [IFTP](#)^[255]. For example:

```
cd ftp:
ftp://jpsoft.com/

cd ftp:/pub
```

Note: FTP directory changes use neither the [CDPATH feature](#)^[72] nor the [Extended Directory Searches](#)^[73] database.

CD never changes the default drive, unless the [/D](#)^[178] option is specified. 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** Changes the current drive as well as directory. This option is included only for compatibility with the same option available in some versions of CMD.EXE. In most cases you should use [CDD](#)^[178], which performs the same function.
- /N** Skips the standard [extended directory search](#)^[73] 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.
- /X** Don't save the current directory to the Directory History list.

3.13.16 CDD

Purpose: Change the current disk drive and directory.

Format: CDD [/A /D[drive ...] /N[J] /S[n][drive ...] /U[drive...]/X] [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(II drives) ^[179]	/T (Also change Folders directory) ^[180]
/D(elete from JPSTREE.IDX) ^[179]	/TO Only change Folders directory) ^[180]
/N(o extended search) ^[179]	/U(pdate tree) ^[180]
/NJ (Skip junctions) ^[180]	/X (exclude from JPSTREE.IDX) ^[180]
/S(earch tree) ^[180]	

See also: [CD](#)^[177], [MD](#)^[277], [PUSHD](#)^[299], [RD](#)^[307], and [Directory Navigation](#)^[71].

Usage:

CDD is similar to the [CD](#) ^[177] command, except that it also changes the default disk drive if one is specified. For example, to change from the root directory on drive A to the subdirectory C:\WP:

```
[a:\] cdd c:\wp
[c:\wp]
```

If no drive / path argument is supplied, CDD displays the current drive and directory.

CDD can also be used to create and update the [Extended Directory Search](#) ^[73] 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](#) ^[49] 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](#) ^[72], then the [extended directory search](#) ^[73] 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](#) ^[74] 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](#) ^[76] 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](#) ^[118], which allows you to return quickly to a recently-used directory.

Windows limits the permissible length of the full subdirectory name (see the [Directories and Subdirectories](#) ^[492] topic for information on directory names).

When changing directories, **TCC** maintains the original case of each path element. This is necessary for a few programs which are case-sensitive in their use of directory names.

Options:

/A 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](#) ^[73] 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 Skips the standard [extended directory search](#) ^[73] 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.

/NJ Skips junctions when indexing directories (see /S).

/S Builds or rebuilds the [Extended Directory Search](#)^[73] 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 [Complete Hidden Files](#)^[50] option 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
```

If you specify a number after the /S, CDD will limit the subdirectory recursion to that number. For example, if you have a directory tree "a\b\c\d\e", /S2 will only index the "a", "b", and "c" directories.

/T Also change the current directory in the **Take Command** Folders and List View windows.

/T Change the current directory in the **Take Command** Folders and List View windows without changing the **TCC** current directory.

/U Updates the [Extended Directory Search](#)^[73] 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:\
```

/X Don't save the current directory to the Directory History list.

Note: The [TREEEXCLUDE](#)^[370] variable can be used to specify which drives and directories should be ignored when updating the directory database.

3.13.17 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.

```
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:

```
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.

3.13.18 CLS

Purpose: Clear the window and move the cursor to the upper left corner; optionally change the default display colors.

Format: CLS [/C /S] [[BRlght] *fg* ON [BRlght] *bg*

fg The new foreground color
bg The new background color

[/C\(lear buffer\)](#)^[181]

[/S\(croll buffer\)](#)^[181]

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:

```
cls
cls bright white on blue
```

CLS is often used in batch files before displaying text.

See [Colors and Color Names](#)^[518] for details about colors.

Options:

- /C** Clears the entire scrollbar buffer. If **/C** is not used, only the visible portion of the window is cleared.
- /S** 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.

3.13.19 COLOR

Purpose: Change the default display colors.

Format: COLOR [BRlght] *fg* ON [BRlght] *bg*

fg The new foreground color
bg The new background color

See also: [CLS](#)^[187] and [Colors and Color Names](#)^[518] 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:

```
color bright white on blue
```

TCC also supports the CMD.EXE syntax:

```
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.

If you have ANSI enabled and StdColors and/or InputColors set, they will override a COLOR command.

3.13.20 COPY

Purpose: Copy data between disks, directories, files, or physical hardware devices (such as your printer or serial port).

Format: COPY [/I"text"] [/A:... /C /D /E /F /FTP:A /G /H /J /K /L /M /MD /N[de]st] /O /P /Q /R /S[n] /T /U /V /X /Z] [@file] source [+] ... [/A/B] [TO:] target [...] [/A/B]

source A file or list of files or a device to copy from.

target 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](#)^[90] for details)

/A(SCII copy) ^[186]	/LD (create link) ^[188]
/A:... (Attribute select) ^[187]	/M(odified files) ^[188]
/B(inary copy) ^[187]	/MD (Create target directory) ^[188]
/C(hanged source files) ^[187]	/N (Disable) ^[188]
/D (Copy encrypted files) ^[187]	/O(nly if no target) ^[188]
/E (No error messages) ^[187]	/P(rompt) ^[188]
/F (No empty subdirectories) ^[187]	/Q(quiet) ^[188]
/FTP:A (ASCII copy) ^[187]	/R(eplace) ^[188]
/G (Display percentage) ^[187]	/S(ubdirectories) ^[188]
/H (Include hidden files) ^[187]	/T(otals) ^[189]
/I"text" (Match description) ^[187]	/U(pdate target) ^[189]
/J (Restartable) ^[187]	/V(erify) ^[189]
/K (Keep read-only attribute) ^[187]	/X (Clear archive) ^[189]
/L Copy symbolic links ^[188]	/Z (overwrite) ^[189]

See also: [ATTRIB](#)^[163], [MOVE](#)^[274], and [REN](#)^[305].

File Selection

Supports [attribute switches](#)^[86], extended [wildcards](#)^[77], [ranges](#)^[80], [multiple file names](#)^[87], [delayed variable expansion](#)^[89], and [include lists](#)^[88]. 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](#)^[89] for details.

Internet

Can be used with [FTP / FTPS / TFTP / HTTP / HTTPS Servers](#)^[93].

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*:

```
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*:

```
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](#)^[371] topic.

• Copying Files

You can copy several files at once by using [wildcards](#)^[77]:

```
copy *.txt e:\mydir
```

You can also list several *source* files in one command. The following command copies 3 specific files from the current directory to the *\MYDIR* directory on drive *E*:

```
copy file1 file2 file3 e:\mydir
```

COPY also understands [include lists](#)^[88], 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*:

```
copy e:\mydir\*.txt;*.doc;*.bat a:\
```

If there is only one parameter 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 from the current directory on drive *A* to the current directory on the current drive:

```
copy a:*.dat
```

If there are two or more parameters on the line separated by spaces, then COPY assumes that the last parameter is the *destination* and copies all *source* files to this new location. If the *destination* is a drive, directory, or device name, 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):

```
copy a:*.dat c:\mydir\  
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 single **destination** file, when you really meant to copy them individually to a **destination** directory.

To copy text to or from the clipboard use **CLIP:** as the device name. Using **CLIP:** with non-text data will produce unpredictable results. See [Redirection](#)^[98] for more information on **CLIP:**.

• Appending Files

A plus sign + tells COPY to append two or more **source** 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 *MEMO2* and *MEMO3* to *MEMO1* and leave the combined contents in the file named *MEMO1*:

```
copy memo1+memo2+memo3
```

To append the same three files but store the result in *BIGMEMO*:

```
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). Many FTP servers, including our own [ftp://jpsoft.com](http://ftp.jpsoft.com), use case sensitive file systems. For example:

```
copy ftp://ftp.abc.com/xyz/index index
```

Files copied to or from FTP/HTTP Servers are normally transferred in binary mode. To perform an ASCII transfer use the [/L](#)^[188] 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](#)^[187] option (percentage 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](#)^[255] 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](#)^[93] and [IFTP](#)^[255].

• NTFS File Streams

COPY supports file streams on NTFS drives. You can copy an individual stream by specifying the stream name, for example:

```
copy myfile:mystream stream.copy
```

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](#)^[496] 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:

```
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:

```
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](#)^[80] 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:

```
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:

```
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:](#)^[187].. Forces COPY to process **source** files with the attributes you specify after the :, or to

process all **source** files regardless of attributes, if [/A:](#)^[187] is used by itself.

[/H](#)^[187] 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](#)^[187] Retains the read-only attribute from each **source** file when creating the **destination** file. See [/K](#)^[187] below for a special note if you are running under Novell NetWare.

[/Z](#)^[189] Forces COPY to overwrite an existing **destination** file regardless of its attributes.

Use caution with [/A:](#)^[187], [/H](#)^[187], or [/K](#)^[187] 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:](#)^[187], [/H](#)^[187], or [/K](#)^[187], as **DESCRIPT.ION** is then treated like any other file.

You can copy files to multiple destinations with the TO: option. For example, to copy **letter.doc** to three different directories:

```
copy letter.doc TO: \save\ f:\backups\ q:\letters\
```

Note: The wildcard expansion process will attempt to allow both CMD.EXE-style "*extension*" matching (assumes only one extension, at the end of the word) and the advanced **TCC** string matching (allowing things like ***.*.abc**) when an asterisk is encountered in the **destination** of a COPY command.

COPY supports connected web folders. If an HTML file (i.e., with an .htm or .html extension) is copied, COPY will look for a folder in the same directory with the same name and an extension of ".files". If it is found, the .files directory will be copied to the target directory. You can disable connected web folders by setting the registry key:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\NoFileFolderConnection=0
```

Options

The [/A](#)^[186] (ASCII copy) and [/B](#)^[187] (binary copy) options apply to the preceding filename and to all subsequent filenames on the command line until the file name preceding the next [/A](#)^[186] or [/B](#)^[187], if any. All other options apply to all filenames on the command line, no matter where you put them.

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](#)^[188] and [/P](#)^[188], COPY will generally take a "conservative" approach and give priority to the option which generates more prompts or more information.

/A 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. If you use **/A** with a **destination** filename, a Control-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**, 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`^[187] option.

- /A:...** Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[86] 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`^[186] switch, above. Do not use **/A:** with **@file** lists. See [@file lists](#)^[90] for details. Hidden or system files selected by this option overwrite hidden or system files.
- /B** If you use **/B** with a **source** filename, the entire file is copied; <SUB> characters, if any, in the file are considered ordinary data to be copied. Using **/B** with a **destination** filename prevents addition of a <SUB> 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**.
- This option applies to the filename immediately preceding it, and to all subsequent filenames until the file name preceding the next `/A`^[186] or `/B` option.
- /C** Copy files only if the **destination** file exists and is older than the **source** (see also `/U`^[189]). This option is useful for updating the files in one directory from those in another without copying any files not already in the target directory. Before using **/C** in a network environment, be sure to read the note under `/U`^[189]. Do not use **/C** with **@file** lists. See [@file lists](#)^[90] for details.
- /D** (Windows 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.
- /F** When used with **/S**, COPY will not create any empty subdirectories.
- /FTP:A** Perform FTP transfers in ASCII mode, instead of the default binary mode.
- /G** Displays the percentage copied, the transfer rate (in Kbytes/second), and the estimated time remaining. Useful when copying large files across a network or via FTP to ensure the copy is proceeding. When `/V`^[189] is also used, reports percentage verified.
- /H** 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. See [Description Ranges](#)^[86] 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 read-only 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** (Windows Vista or later only) If the source is a symbolic link, copy the link to the target instead of the actual file.
- /LD** When used with /S, if the source is a symbolic or hard link to a directory, COPY will create the link in the target directory instead of copying the subdirectory tree.
- /M** 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](#)^[189] switch, or use [ATTRIB](#)^[163]. Do not use /M with **@file** lists. See [@file lists](#)^[90] for details.
- /MD** Create the target directory if it doesn't exist. Note that you **must** either terminate the target directory name with a trailing \ or specify a filename component; otherwise COPY cannot tell what you want for the directory and what you want for the filename.
- /N** 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](#)^[188].
- A **/N** with one of the following arguments has an alternate meaning:
- d** Skip hidden directories (when used with /S)
 - e** Don't display errors.
 - j** Skip junctions (when used with /S)
 - s** Don't display the summary.
 - t** Don't update the CD / CDD [extended directory search](#)^[73] database ([JPSTREE.IDX](#)).
- /O** Only copy the source file if the target file doesn't exist.
- /P** Ask the user to confirm each **source** file. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[102]. Note: the [Copy Prompt on Overwrite](#)^[47] configuration option can be used to force prompting at the command line only. See also: the [/Q](#)^[188] option below.
- /Q** Don't display filenames, percentage copied, total number of files copied, etc... When used in combination with the [/P](#)^[188] 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](#)^[189].
- /R** Prompt the user before overwriting an existing file. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[102]. See also: the [Copy Prompt on Overwrite](#)^[47] configuration option. (For compatibility with CMD.EXE, a /Y option on the command line is changed to /R.)
- /S** 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](#)^[73] 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](#)^[90] for details.

If you specify a number after the **/S**, COPY will limit the subdirectory recursion to that number. For example, if you have a directory tree "a\b\c\d\e", **/S2** will only affect the "a", "b", and "c" directories.

- /T** Turns off the display of filenames, like [/Q](#)^[188], but does display the total number of files copied.
- /U** 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](#)^[187]). 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](#)^[90] for details. When used with file systems that have different time resolutions (such as [FAT](#)^[528] and [NTFS](#)^[532]), **/U** will attempt to use the "coarsest" resolution of the two.
- /V** Verify each disk write by performing a true byte-by-byte comparison between the source and the newly-created target file. This option will significantly increase the time necessary to complete a COPY command.
- /X** Clear the archive attribute from the source file after a successful copy. This option is most useful if you are using COPY to maintain a set of backup files.
- /Z** Overwrite **destination** files regardless of their attributes. Without this option, COPY will fail with an "Access denied error" if the **destination** file has its read-only attribute set, or (depending on other options) its hidden or system attribute set. Required to overwrite read-only targets regardless of other options. Required to overwrite hidden or system targets unless the source also has the attribute, and either [/H](#)^[187] or [/A:](#)^[187] is used to select it.

3.13.21 DATE

Purpose: Display and optionally change the system date.

Format: DATE [/T] [*mm -dd -yy*]

mm The month (1 - 12)
dd The day (1 - 31)
yy The year (80 - 99 or a 4- digit year)

[/T \(Display only\)](#)^[190]

See also: [TIME](#)^[347].

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:

```
date
Fri Sep 1, 2006 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:

```
date 11-16-2006
```

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 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.

3.13.22 DEBUGSTRING

Purpose: Write text to the debugger for display.

Format: DEBUGSTRING string.

Usage:

If the application has no debugger, the system debugger displays the message. If the application has no debugger and the system debugger is not active, DEBUGSTRING does nothing.

3.13.23 DEFER

Purpose: Execute a command after the batch file exits

Format: DEFER command

Usage:

A batch file can have multiple DEFER commands. They will be executed in first in, first out order when the batch file exits.

If you have variables on the DEFER command line, they will be expanded before the DEFER command is processed, not when **command** is executed. To delay variable expansion until **command** is executed, use single back quotes around the variable names, or double the %'s before the variable names.

3.13.24 DEL / ERASE

Purpose: Erase one file, a group of files, or entire subdirectories.

Format: DEL [ranges] [/A:[-|+]*rhsadeci*jopt /E /F /I"text" /K /N[defjst] /P /Q /R /S[n] /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](#) for details).

/A: (Attribute select) ¹⁹³
/B (Delete after reboot) ¹⁹³
/E (No error messages) ¹⁹³

/Q(uiet) ¹⁹³
/R(ecycle bin) ¹⁹³
/S(ubdirectories) ¹⁹³

/F(orce delete)	/T(otal)
/I (match descriptions)	/W(ipe)
/K (no Recycle Bin)	/X (remove empty subdirectories)
/N (Disable)	/Y(es to all prompts)
/P(rompt)	/Z(ap hidden and read-only files)

File Selection

Supports [attribute switches](#), extended [wildcards](#), [ranges](#), [multiple file names](#), and [include lists](#). Use wildcards with caution on LFN volumes; see [LFN File Searches](#) for details.

Internet

Can be used with [FTP/HTTP Servers](#).

Usage

DEL and ERASE are synonyms. You can use either one. In the description below, every reference to DEL applies equally to ERASE.

Use the DEL command 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:

```
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:

```
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](#). For example, to delete all files in the current directory except those whose extension is *.TXT*, use a command like this:

```
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 **TCC** to follow CMD.EXE's approach and skip the confirmation prompt when /Q is used, set the [Prompt on Wildcard Deletes](#) configuration option. Use caution if you disable this option, 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.

When a file is deleted without using the Recycle Bin, 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 the [/W](#)^[194] option, which overwrites the file with zeros before deleting it. Use this option with caution. Once a file is obliterated, it is impossible to recover. Remember: [/W](#)^[194] overrides using the Recycle Bin.

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 [%_?](#)^[380] internal variable, and use it with [conditional commands](#)^[120] (&& and ||).

Use caution when using wildcards with DEL on LFN drives, because **TCC**'s wildcard matching can match both short and long filenames. This can delete files you did not expect; see [LFN File Searches](#)^[89] for additional details.

• Recycle Bin

When you delete files with DEL, **TCC** does not move the deleted files to the Windows Recycle Bin by default. You can change this default with the [Delete to Recycle Bin](#)^[47] configuration option. If you have disabled the recycle bin, you can override the setting and place deleted files in the recycle bin with the [/R](#)^[193] option:

```
del /r letters.txt
```

If you have enabled 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](#)^[193] option:

```
del /k letters.txt
```

You can also exclude files from the Recycle bin, even if [Delete to Recycle Bin](#)^[47] is enabled, or if the command use the [/R](#)^[193] option, with the [RecycleExclude](#)^[369] environment variable.

• FTP Usage

If you have appropriate permissions, you can delete files on [FTP servers](#)^[93]. For example:

```
del ftp://ftp.abc.com/index
```

You can also use the [IFTP](#)^[255] command to start an FTP session on a server and then use one of the following syntax examples:

```
del ftp:path/*.txt
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. You can delete an individual stream by specifying the stream name, for example:

```
del streamfile:s1
```

If no stream name is specified the entire file is deleted, including all streams.

See [NTFS File Streams](#)^[496] for additional details.

Options

- /A:** Delete only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[86] for information on the attributes which can follow **/A:**. Do not use **/A:** with @file lists. See [@file lists](#)^[90] for details.
- /B** If DEL can't delete the file (for example, if access is denied) it will schedule it to be deleted at the next reboot.
- /E** 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** This option has the same effect as [/Z](#)^[194] (see below): it deletes read-only, hidden, and system files as well as normal files.. It is included for compatibility with CMD.EXE.
- /I"text"** Select filenames by matching text in their descriptions. The text can include [wildcards](#)^[77] and extended wildcards. The search text must be enclosed in double quotes, 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** Physically delete files instead of sending them to the Windows Recycle Bin.
- /N** Do everything except actually delete the file(s). This is useful for testing the result of a DEL.

A **/N** with one of the following arguments has an alternate meaning:
 - d** Skip hidden directories (when used with **/S**)
 - e** Don't display errors
 - f** Don't display the bytes freed in the summary
 - j** Skip junctions (when used with **/S**)
 - s** Don't display the summary
 - t** Don't update the CD / CDD [extended directory search](#)^[73] database (*JPSTREE.IDX*)
- /P** Prompt the user to confirm each erasure. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[102].
- /Q** Don't display filenames as they are deleted, or the number of files deleted or bytes freed. If [Prompt on Wildcard Deletes](#)^[47] is disabled then **/Q** also disables the normal confirmation prompt when performing wildcard deletions (e.g. DEL *), for compatibility with CMD.EXE. Use caution if you disable [Prompt on Wildcard Deletes](#)^[47], as this will allow DEL /Q to delete an entire directory without prompting for confirmation. See also [/T](#)^[194].
- /R** Delete files to the Windows Recycle Bin.
- /S** 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](#)^[90] for details.

If you specify a number after the /S, DEL will limit the subdirectory recursion to that number. For example, if you have a directory tree "a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.

- /T** 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](#)^[193], the /T option will not speed up deletions under DOS.
- /W** 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](#)^[193].
- /X** Removes empty subdirectories (only useful when used with [/S](#)^[193]). If DEL deletes one or more directories, they will be removed automatically from the [extended directory search database](#)^[73].
- /Y** The reverse of [/P](#)^[193] — it assumes a Y response to everything, including deleting an entire subdirectory tree. **TCC** 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** 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](#)^[225] works by hiding files, /Z will override an [EXCEPT](#)^[225] command. However, files specified in a [file exclusion range](#)^[85] 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!):

```
del /s /x /y /z c:\util\
```

3.13.25 DELAY

Purpose: Pause for a specified length of time.

Format: DELAY [/B /M *time*]

time The number of seconds or milliseconds to delay.

[/B\(reak enabled\)](#)^[195]

[/M\(illiseconds\)](#)^[195]

Usage:

DELAY is useful in batch file loops while waiting for something to occur. For example, 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 **time** value can be as large as about 1 billion seconds (34 years!). If you don't enter a **time**, the default is 1 second.

TCC 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** Allows terminating a DELAY by pressing a key.
- /M** Count by milliseconds instead of seconds. Normally only used for delays of less than 1 second.

3.13.26 DESCRIBE

Purpose: Create, modify, or delete file and subdirectory descriptions.

Format: Creating or modifying descriptions
 DESCRIBE [ranges... /I"text"] [/A:atrlst] [@file] file [[/D]"description"]] ...]

Description file updating
 DESCRIBE /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](#)^[90] for details).
"description" The description to attach to the file.

[/A: \(Attribute select\)](#)^[197] [/I \(match description\)](#)^[197]
[/D\(escription follows\)](#)^[197] [/U\(pdate\) descriptions file](#)^[197]

See also: [@DESCRIPT](#)^[415], [DIR](#)^[198], and [SELECT](#)^[312]

File Selection

Supports [attribute switches](#)^[86], extended [wildcards](#)^[77], [ranges](#)^[80], [multiple file names](#)^[87], and [include lists](#)^[88]. Use wildcards with caution on LFN volumes; see [LFN File Searches](#)^[89] for details.

Usage:

DESCRIBE adds descriptions to files and subdirectories. (Volume root directories cannot have descriptions.) The descriptions are displayed by [DIR](#)^[198] in single-column mode and by [SELECT](#)^[312], and can be retrieved using the [@DESCRIPT](#)^[415] function. Descriptions let you identify your files in much more meaningful ways than you can in a filename alone.

You enter a description on the command line by typing the DESCRIBE, the filename, and the description in double quotes, like this:


```
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:

```
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](#)^[53] drive, you must quote **file** 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, **TCC** will treat the string as a quoted file name, not as description text as you intended. To resolve this problem use the [/D](#)^[197] 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):

```
describe data.tst /D"test file"    correct command
describe data.tst "test file"      incorrect command
```

On [LFN](#)^[53] drives you will not see file descriptions in a normal [DIR](#)^[198] display, because [DIR](#)^[198] must leave space for the long filenames. To view the descriptions, use [DIR](#)^[198] /Z to display the directory in FAT format. See [DIR](#)^[198] for more details.

Each description can be up to 511 characters long. You can change this limit with the [Maximum Length](#)^[51] configuration option. 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](#)^[198] output). DESCRIBE can edit descriptions longer than [Maximum Length](#)^[51] (up to a limit of 511 characters), but will not allow you to lengthen the existing text.

The descriptions are stored either in the NTFS SummaryInformation stream (if you have set the [NTFS Descriptions](#)^[51] configuration option), or in each directory in a hidden file called *DESCRIPT.ION*. Use the [ATTRIB](#)^[163] 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 **TCC** if you remove the hidden attribute.

You can change the description file name with the [Description Filename](#)^[45] configuration option or the [SETDOS /D](#)^[323] command, and retrieve it with the [%_DNAME](#)^[385] 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](#)^[182], [MOVE](#)^[274], [DEL](#)^[190], or [RENAME](#)^[305]), but not if you use an external program (such as XCOPY or Explorer). You can disable description processing with the [Enable Descriptions](#)^[45] configuration option, or with [SETDOS /D](#)^[323].

When you [COPY](#)^[182], [MOVE](#)^[274] or [REN](#)^[305] 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](#)^[182] and [MOVE](#)^[274] for additional details.

If you disable descriptions with the [SETDOS](#)^[323] /D0 option, DESCRIBE will return with an error message.

Options:

- /A:** Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[86] for information on the attributes which can follow /A:. Do not use /A: with @file lists. See [@file lists](#)^[90] for details.
- /D"description"**
The quoted string following the /D switch without any separation is used as a description, not a file name, avoiding ambiguity in the meaning of quoted strings. See the **Usage** section above for details.
- /I"text"**
Select files by matching text in their descriptions. The text can include [wildcards](#)^[77] and extended wildcards. The search text must be enclosed in double quotes, 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](#)^[90] for details.
- /U** Update the *DESCRIPT.ION* file (or the file specified by the [Description Filename](#)^[45] configuration option), 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.

3.13.27 DETACH

Purpose: Start a console (character-mode) application program in detached mode.

Format: DETACH [/Q] command

[/Q\(quiet\)](#)^[198]

command The name of a command to execute, including an optional drive and path specification and any parameters. The name must be enclosed in double quotes if it contains any spaces.

See also: [START](#)^[331] and [TASKEND](#)^[342].

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](#)^[98] 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, **TCC** will detach a copy of **TCC** to execute the command.

For example, the following command will detach a copy of **TCC** to run the batch file *XYZ.BTM*:

```
detach xyz.btm
```

You can also include any parameters or command line switches which the command knows how to interpret:

```
detach "xyz.btm Monday Nebraska"
```

Once the program has started, **TCC** 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](#)^[385] variable.

You can use the [TASKEND](#)^[342] command to stop a detached program which does not terminate on its own.

Options:

/Q Don't display the new process's ID.

3.13.28 DIR

Purpose: Display information about files and subdirectories.

Format: DIR [ranges] [options] [file...]

ranges one or more [ranges](#)^[80]
options one or more file selection or report format [options](#)^[205]
file The file, directory, or list of files or directories to display.

/1 ^[205]	1 column output	/L ^[206]	Lower case
/2 ^[205]	2 column output	/M ^[206]	suppress footer
/4 ^[205]	4 column output	/N ^[206]	New format or disable options
		[eshv]	
/: ^[205]	show streams	/O ^[206]	Order
/A ^[205]	Attribute select	/P ^[207]	Pause
/B ^[205]	Bare (name only)	/Q ^[207]	show owner
/C ^[206]	show Compression	/R ^[207]	disable wrap
/D ^[206]	Disable color coding	/S ^[207]	show Subdirectories to depth <i>n</i>
/E ^[206]	upper case	/T ^[207]	show aTtribute
/F ^[206]	Full path	/T: ^[207]	time type
/G ^[206]	allocated size	/U ^[208]	show summary information
/H ^[206]	Hide dots	/V ^[208]	Vertical sort
/I ^[206]	description range	/W ^[208]	Wide
"text"			
/J ^[206]	Justify names	/X ^[208]	show short names
/K ^[206]	suppress header	/Z ^[208]	use FAT format

See also: [ATTRIB](#)^[163], [DESCRIBE](#)^[195], [PDIR](#)^[288], [SELECT](#)^[312], and [SETDOS](#)^[323].

File Selection

Supports extended [wildcards](#)^[77], [ranges](#)^[80], [multiple file names](#)^[87], and [include lists](#)^[88].

Internet: Can be used with [FTP servers](#)^[93].

Usage:

DIR can be used to display information about files from one or more 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](#)^[288] command. DIR and PDIR are related, but 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](#)^[154] 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:

```
dir /2/p/v
```

To set up this format as the default, using an alias:

```
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](#)^[199]
- ▶ [Default DIR Output Format](#)^[200]
- ▶ [Switching Formats](#)^[201]
- ▶ [Multiple Column Displays](#)^[202]
- ▶ [Color-Coded Directories](#)^[203]
- ▶ [Redirected Output](#)^[204]
- ▶ [Other Notes](#)^[204]
- ▶ [Options](#)^[205]
- ▶ [FTP usage](#)^[204]

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:

```
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](#)^[77]. For example to display all of the .WKS files in the current directory:

```
dir *.wks
```

To display all .TXT files whose names begin with A, B, or 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:

```
dir *.wks *.wk1
```

If you use an [include list](#)^[88] 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:

```
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:

```
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:

```
dir /a:d
```

Finally, with the `/I` option, DIR can select files to display based on their descriptions (see [DESCRIBE](#)^[195] 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:

```
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:

```
dir /i"*test file*"
```

To display only those files which do not have descriptions, use:

```
dir /I"[]"
```

In addition, you can use [ranges](#)^[80] 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:

```
dir /[d-7]
dir /[*.*.bak]
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 C is unlabeled          Serial number is 3aaf:c891
Directory of  C:\release\version9\*

12/10/2007    0:39                <DIR>      .
12/10/2007    0:39                <DIR>      ..
11/25/2007   11:30                <DIR>      tcmd
12/09/2007   23:07           4,801,840  tcmd.exe
```

(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>          8-24-08   12:17
..               <DIR>          8-24-08   12:17
TEST             <DIR>          8-01-08   16:21
jptestree.idx    196967        8-28-08   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:

```
dir /o:d
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" follows "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 **TCC** on an LFN drive:

	default or /1 ^[205]	/2 or /4 columns	/W (wide)
Normal	date, time, size, LFN	2 - 4 columns, LFNs only	No. of columns based on longest LFN
/Z ^[208] (FAT)	truncated LFN, size, date, time	2 - 4 columns, truncated LFN plus date, time, size	5+ columns, truncated LFNs only
/X ^[208] (SFN)	date, time, size, SFN, LFN	2 - 4 columns, SFNs plus date, time, size	5+ columns, SFNs only
/X ^[208] /Z ^[208]	SFN, size, date, time	(Same as /X)	(Same as /X)

Color-Coded Directories

DIR 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](#)^[319] command to create an environment variable called [COLORDIR](#)^[368], or use the [Directory Colors](#)^[49] configuration option. If you use neither the variable nor the configuration option, DIR will use the default screen colors for all files.

If you use the COLORDIR variable, it will override the [Directory Colors](#)^[49] option. You may find it useful to use the COLORDIR variable for experimenting, then to set permanent directory colors with the [Directory Colors](#)^[49] option.

The format for both the COLORDIR environment variable and the [Directory Colors](#)^[49] option is:

```
ext ... :ColorName; ...
```

where "ext" is either a file extension (which may include wildcards) or one or more of the following file types:

type	files affected
ARCHIVE	Files with archive attribute set (modified since the last backup)
COMPRESSED	Compressed files
DIRS	Directories
ENCRYPTED	Encrypted files
HIDDEN	Hidden files
JUNCTION	Junctions or symbolic links
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](#)^[518] 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. `xyz.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 `.COM` and `.EXE` files in red on the current background, `.C` and `.ASM` files in bright cyan on the current background, read-only files in green on white, and everything else in the default color:

```
set colordir=com exe:red; c asm:bright cyan; rdonly:green on white
```

[Extended wildcards](#)^[77] can be used in directory color specifications. For example, to display `.BAK`, `.BAX`, and `.BAC` files in red, and everything else in the default color:

```
set colordir=BA[KXC]:red
```

You can combine attribute tests with the `.and.` / `.or.` / `.xor.` / `.not.` keywords. For example, to display directories that are also hidden in blue:

```
set colordir=dirs .and. hidden:blue
```

COLORDIR processes the line from left to right, and does not support parentheses.

Redirected Output

The output of the DIR command, like that of most other internal commands, can be [redirected](#)^[98] 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 or a character device, DIR will remove the color data as it sends the directory information to a file.

To redirect DIR output to the clipboard, use **CLIP:** as the output device name, for example:

```
dir *.exe > clip:
```

FTP Usage

You can display directories on [FTP servers](#)^[93]. For example:

```
dir ftp://jpsoft.com/tcmd
```

You can also use the [IFTP](#)^[255] 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](#)^[26].

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 NTFS disk compression, 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. **/C** and **/O:c** will be ignored for 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.

If the OFFLINE attribute is set, DIR will display the file size enclosed in parentheses (for compatibility with CMD.EXE).

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. In Windows Vista or later on NTFS drives, single column displays will also show the target of symbolic links following the filename.
- /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
  xyz:$DATA
  abc:$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:xyz
c:\test\myfile.dat:abc
```

- /A[:]** Display only those files that have the specified attribute(s) set. See [Attribute Switches](#)⁸⁶ for information on the attributes which can follow **/A:**.
- /B** 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** Display per-file and total compression percentage on NTFS drives with compression enabled. **/C** only works in single-column mode; it is ignored if **/2**, **/4**, or **/W** is used.
- /D** Temporarily disable directory color coding. May be required when color-coded directories are used and DIR output is redirected to a character device like a serial port (e.g., COM1). **/D** is not required when DIR output is redirected to a file.
- /E** Display filenames in upper case.
- /F** 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** Display the allocated disk space instead of the actual size of each file.
- /H** Suppress the display of the "." and ".." directories.
- /I "text"** Select filenames by matching text in their descriptions. The text can include [wildcards](#) and extended wildcards. The search text must be enclosed in double quotes, 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 (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** Display file and directory names in lower case.
- /M** Suppress the footer (file and byte count totals) display.
- /N** Use the long filename display format, even if the files are stored on a volume which does not support long filenames. See also **/Z**.

A **/N** with one of the following arguments has an alternate meaning:

- d** Skip hidden directories (when used with **/S**)
- e** Don't display an error message if no files match.
- f** Don't display "bytes free" in the summary
- j** Skip junctions (when used with **/S**)
- h** Don't display the header
- s** Don't display the summary.
- v** Don't display the volume information.

- /O** 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 sort key
 - 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)
- o** Sort by owner
- r** Reverse the sort order for all options
- s** Sort by size
- u** Unsorted

/P 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](#)^[102].

/Q Display the file or directory owner (NTFS and remote directories only).

/R 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 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**.

If you specify a number after the **/S**, DIR will limit the subdirectory recursion to that number. For example, if you have a directory tree "a\b\c\d\e", **/S2** will only affect the "a", "b", and "c" directories.

/T Display the filenames and attributes in the format **RHSADENTPCOIJ**, regardless of volume type:

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 or symbolic link

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, **TCC** will interpret the **/T** as the **/T:{acw}** time display switch (see below) and the following character as a time selector. For example:

```
dir /tz      incorrect, will display error
dir /t/z    correct
```

/T:a|c|w|u 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 (on VFAT volumes access time is always midnight).
- c** Creation date and time.
- w** Last modification (write) date and time (default).

If you append a **u** after the field, DIR will display the file time in UTC.

- /U** 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** Display the filenames sorted vertically rather than horizontally (use with the [/2](#)^[205], [/4](#)^[205] or [/W](#)^[208] options).
- /W** 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 **TCC** 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. On **NTFS** volumes this means case insensitive match, but on **VFAT** volumes this means case sensitive match (i.e., no lower case letters in the **SFM**). **/X** also selects short filenames in the [/2](#)^[205], [/4](#)^[205], [/B](#)^[205], [/W](#)^[208], 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](#)^[208] 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](#)^[51] exists, **/Z** defaults to using the name of the . and .. directories as description for those entries

3.13.29 DIRHISTORY

Purpose: Display, add to, clear, or read the directory history list.

Format: DIRHISTORY [/A directory /F /G /L /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) ^[209]	/N(o duplicates) ^[209]
/F(ree) ^[209]	/P(ause) ^[209]
/G(lobal) ^[209]	/R(ead) ^[209]
/L(ocal) ^[209]	

See also: [HISTORY](#)^[251].

Usage

Every time you change to a new directory or drive, **TCC** saves the previous directory in an internal

directory history list. The [directory history window](#)^[118] allows you to use the list to return to a previous directory. See also: [directory navigation](#)^[71].

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:

```
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 with the [Directory History Buffer Size](#)^[50] configuration option. The default size is 2,048 characters.

Your directory history list can be stored either locally (a separate history list for each copy of **TCC**) or globally (all copies of **TCC** share the same list). For details see the discussion of [local and global history lists](#)^[108]. If you use global lists, [SHRALIAS](#)^[329] can save the list when no copy of **TCC** is active, as long as you do not restart Windows.

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.

```
dirhistory > dirhist
.....
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.

TCC can also load and save the history list automatically if you use the [Directory History File](#)^[50] configuration option.

Options

- /A** Add a directory to the directory history list.
- /F** Erase all entries in the directory history list.
- /G** Switch from a local to a global directory history list.
- /L** Switch from a global to a local directory history list.
- /N** Removes duplicate entries (oldest first) from the directory history list.
- /P** 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](#)^[102].
- /R** Read the directory history from the specified file and append it to the list currently held in memory.

3.13.30 DIRS

Purpose: Display the current directory stack.

Format: DIRS

See also: [PUSHD](#)^[299], [POPD](#)^[294], [@DIRSTACK](#)^[416] and [Directory Navigation](#)^[71].

Usage:

The [PUSHD](#)^[299] command adds the current default drive and directory to the directory stack, a list maintained by **TCC**. The [POPD](#)^[294] command removes the top entry of the directory stack and makes that drive and directory the new default. The **DIRS** command displays the contents of the directory stack, with the most recent entries on top (*i.e.*, the next **POPD** will retrieve the first entry that **DIRS** displays).

For example, to change directories and then display the directory stack:

```
[c:\] pushd c:\database
[c:\database] pushd d:\wordp\memos
[d:\wordp\memos] dirs
c:\database
c:\
```

The directory stack holds 2048 characters, enough for 40 to 80 typical drive and directory entries.

3.13.31 DO

Purpose: Create loops in batch files.

Format: DO loop_control
 commands
[ITERATE](#)^[213]
 commands
[LEAVE](#)^[213]
 commands
 ENDDO

Loop_control formats

```
DO [211]count  

DO FOREVER [211]  

DO [211]varname = start TO end [BY step]  

DO WHILE [212]condition  

DO UNTIL [212]condition  

DO UNTIL DATETIME [212]date time  

DO FOR [212]n [SECONDS | MINUTES | HOURS]  

DO [212]varname IN [range...] /D"directory" [/I:"text" /S[n] /A:[-|+]rhsadecijopt] filesset  

DO [212]varname IN [/T"delimiters"] /L stringset  

DO [212]varname IN /C stringset  

DO [212]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 the variable name with %.
start, end, step	Integers in the range [-2 147 483 647, 2 147 483 647] or internal variables or variable functions that evaluate to such values, controlling the number of times the loop is executed.
condition	A conditional expression ^[109] to determine whether or not the loop should be executed

filename	A filename or list of filenames, possibly using wildcards ^[77]
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 ^[80]	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: ^[213]	(Attribute select)
/C ^[213]	Loop through each character in expression
/D"directory" ^[213]	Start directory
/I"text" ^[214]	(match description) Description range.
/L(iteral) ^[214]	members of set are strings, not filenames
/S ^[214]	Perform the loop in the current directory and all its subdirectories

Supports extended [wildcards](#) ^[77], [ranges](#) ^[80], and [include lists](#) ^[88] for the **set**. Use wildcards with caution on LFN volumes; see [LFN File Searches](#) ^[89] for details.

Usage

DO can only be used in [batch files](#) ^[130].

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. **TCC** 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. For example::

```
set ct=0
do 5
    beep
    set ct=%@inc[%ct]
enddo
```

- DO FOREVER creates an endless loop. You must use [LEAVE](#) ^[213] or [GOTO](#) ^[248] 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
enddolea
```

DO can also count down, rather than up. If **step** is negative, **varname** will be decreased by the absolute value of **step** 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](#)^[109] **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](#)^[109] 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). The date and time can be in either YYYY-MM-DD HH:MM:SS or YYYYMMDDHHMMSS format. (The date and/or time can be a variable.)
- DO FOR **n** SECONDS | MINUTES | HOURS executes the loop for the specified amount of time.
- 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 file or directory names. This is similar to the **set** used in the [FOR](#)^[234] command, but it can only include file and directory names, not arbitrary text strings. If **varname** already exists in the environment, it will be overwritten (unlike the control variable in [FOR](#))^[234]. 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 **TCC** has no control.

To use date, time, size, description, or file exclusion [ranges](#)^[80] for the **set** place them just before the filename(s), for example:

```
do x in [/d9-1-2004,9-31-2004] *.txt
```

- 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 /C **stringset** executes the commands between DO and ENDDO once for every character in **stringset** (including whitespace and special characters), 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 **TCC**, such as redirection and piping symbols, within the file (use [SETDOS](#)^[323] /X as needed).

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](#)^[98] for more information on **CLIP:**.

Special DO keywords: ITERATE and LEAVE

Two special keywords, ITERATE and LEAVE, may 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](#)^[253] or [IFF](#)^[254] 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](#)^[121], 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)      invalid command
line
```

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](#)^[248] to a line past the corresponding ENDDO. However, be sure to read the cautionary notes about [GOTO](#)^[248] and DO under the [GOTO](#)^[248] command before using GOTO in any other way inside any DO loop.

You cannot use [RETURN](#)^[307] to return from a [GOSUB](#)^[247] while inside a DO loop.

Note: Do not confuse the DO command with the unrelated optional **do** keyword of the [FOR](#)^[234] command.

Options:

- /A:** Select the files in a [DO](#)^[212] IN ... by their specified attribute(s). See [Attribute Switches](#)^[86] for information on the attributes which can follow **/A**.
- /C** For each loop, assign the next character (including whitespace and special characters) in the expression to the DO variable.
- /D"directory"** Set the start directory (for use with /S).

- /I" *text*"** Select files in a [DO](#) ^[212] IN ... by matching ***text*** in their descriptions. See [Description Ranges](#) ^[86] for details.
- /L** The parameters following [DO](#) ^[212] IN /L are strings, not filenames. Each parameter will be assigned in sequence, from left to right, to the loop control variable on consecutive passes through the loop.
- /N** Disable options:
- d** Skip hidden directories (when used with /S)
 - j** Skip junctions (when used with /S)
- /S** Perform the DO loop in the current directory and then on all of its subdirectories. (DO also supports /R as a synonym, for compatibility with FOR.)
- If you specify a number after the /S, DO will limit the subdirectory recursion to that number. For example, if you have a directory tree "a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.
- /T" *text*"** Specify the delimiters to be used when parsing a string set.

3.13.32 DRAWBOX

Purpose: Draw a box on the screen.

Format: DRAWBOX ulrow ulcol lrow lrcol 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
lrcol 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](#) ^[215] and [DRAWVLINE](#) ^[216].

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](#) ^[518] 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.

The maximum **row** value is determined by the current height of the **TCC** window. The maximum **column** value is determined by the current virtual screen width (see [Resizing the Take Command Window](#)^[68] 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 a Unicode or U.S. English extended ASCII character set. If you use an ASCII or raster font which does not include these line drawing characters, the box or lines will not be displayed correctly.

3.13.33 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](#)^[214] and [DRAWVLINE](#)^[216].

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.

The maximum *row* value is determined by the current height of the **TCC** window. The maximum *column* value is determined by the current virtual screen width (see Resizing the [Take Command Window](#)^[68] 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](#)^[518] for details about colors.

DRAWHLINE uses the standard line and box drawing characters in a Unicode or U.S. English extended ASCII character set. If you use an ASCII or raster font which does not include these line drawing characters, the box or lines will not be displayed correctly.

3.13.34 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](#)^[214] and [DRAWHLINE](#)^[215].

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.

The maximum *row* value is determined by the current height of the **TCC** window. The maximum *column* value is determined by the current virtual screen width (see [Resizing the Take Command Window](#)^[68] for more information).

See [Colors and Color Names](#)^[518] for details about colors.

DRAWVLINE uses the standard line and box drawing characters in a Unicode or U.S. English extended ASCII character set. If you use an ASCII or raster font which does not include these line drawing characters, the box or lines will not be displayed correctly.

3.13.35 ECHO

Purpose: Enable or disable batch file or command line echoing, display the echoing status on [stdout](#)^[535], or display a message on [stdout](#)^[535].

Format: ECHO [ON | OFF | message]

message Text to display.

See also the commands [ECHOS](#)^[219], [ECHOSERR](#)^[220], [ECHOERR](#)^[218], [SCREEN](#)^[310], [SCRPUT](#)^[311], [TEXT](#)^[345] and [VSCRPUT](#)^[360], and the internal variable [_ECHO](#)^[386].

Usage:

The ECHO command has two unrelated, independently functioning purposes:

- [Command line echoing](#)^[217]
- [Message display](#)^[217]

Command line echoing

When command line echoing is enabled, each command is displayed on [stdout](#)^[535] after it is fully parsed, aliases, functions, and variables expanded, but before it is executed.

Echoing control

TCC controls command line echoing in batch files and at the interactive prompt independently.

Executing ECHO ON at the command prompt enables, and ECHO OFF disables echoing at the command prompt. ECHO defaults to OFF at the command line. The command-line ECHO is most useful when you are learning how to use advanced features.

Similarly, executing ECHO ON in a batch file enables, and ECHO OFF disables echoing of batch file commands. ECHO defaults to ON in batch files. The current ECHO state is inherited by called batch files. You can change the default setting to OFF with the [SETDOS /V0](#)^[323] command, or the [Batch Echo](#)^[47] configuration option.

Regardless of the relevant echoing state, any command prefixed with the at-sign @ will not be echoed.

Echoing state display

To see the current echoing state, use the ECHO command with no parameters. 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](#)^[386].

Message display

If the ECHO command has a message (the whole [command tail](#)^[524], excluding redirection or piping, if any), and message is neither of the words ON or OFF (though it can include those words), message is fully parsed, then displayed on [stdout](#)^[535], regardless of the applicable echoing state. Any display sent to [stdout](#)^[535] after message has been displayed will start on a new line.

Display rules

- The first space after the command name is ignored.
- Trailing spaces in **message** are ignored.
- Functions and variables not enclosed between back quotes are evaluated.
- To include special characters, .e.g., < | >, in **message**, enclose them in double quotes or back quotes (see [Parameter Quoting](#)^[134]) or precede them with the [escape character](#)^[124], or use the /X option of the [SETDOS](#)^[323] command.
- To display % you may alternately use two % marks for each one to be displayed, e.g., %%
- To display trailing spaces, either enclose them in back quotes, or append a pair of back quotes behind them, e.g., **echo trailers** ``
- The [ASCII](#)^[511] NUL character cannot be included in **message**.
- If [stdout](#)^[535] is the console, after displaying **message** on the current line, the cursor will be moved to the beginning of the next line.
- If [stdout](#)^[535] is a file, the CR LF sequence will be appended to **message**.

To display a blank line, use one of the forms below:

```
echo ``      (two consecutive back quotes), or
echo.       (special syntax for compatibility with CMD.EXE).
```

Examples

This command will display a message:

```
echo Processing your print files...
```

The command

```
echo      This text is indented 3 spaces  ``
```

will display 3 leading and 3 trailing spaces.

3.13.36 ECHOERR

Purpose: Display a message to the standard error device ([stderr](#)^[535]).

Format: ECHOERR message

message Text to display.

See also: [ECHO](#)^[217], [ECHOS](#)^[219], and [ECHOSERR](#)^[220].

Usage:

ECHOERR (like [ECHO](#)^[217] in message display mode) parses and expands **message**, and displays it on [stderr](#)^[535] (usually the screen), instead of [stdout](#)^[535]. Even if [stdout](#)^[535] of a batch file is redirected or piped, ECHOERR will still display a screen message, unless [stderr](#)^[535] is redirected or piped (see [Redirection](#)^[97]). Any display sent to [stderr](#)^[535] after **message** has been displayed will start on a new line.

Display rules

- The first space after the command name is ignored.

- Trailing spaces in **message** are ignored.
- Functions and variables not enclosed between back quotes are expanded.
- To include special characters, .e.g., < | >, in **message**, enclose them in double quotes or back quotes (see [Parameter Quoting](#)^[134]) or precede them with the [escape character](#)^[124], or use the /X option of the [SETDOS](#)^[323] command.
- To display % you may alternately use two % marks for each one to be displayed, e.g., %%
- To display trailing spaces, either enclose them in back quotes, or append a pair of back quotes behind them, e.g., **echoerr trailers** ``
- The [ASCII](#)^[511] NUL character cannot be included in **message**.
- If [stderr](#)^[535] is the console, after displaying **message** on the current line, the cursor will be moved to the beginning of the next line.
- If [stderr](#)^[535] is a file, the CR LF sequence will be appended to **message**.

3.13.37 ECHOS

Purpose: Display a message to standard output ([stdout](#))^[535] without a trailing carriage return / line feed.

Format: ECHOS message

message Text to display.

See also: [ECHO](#)^[217], [ECHOERR](#)^[218], [ECHOSERR](#)^[220], [SCREEN](#)^[310], [SCRPUT](#)^[311], [TEXT](#)^[345], and [VSCRPUT](#)^[360].

Usage:

ECHOS, like [ECHO](#)^[217] in message display mode, parses, expands, and displays **message** on [stdout](#)^[535]. However, any display sent to [stdout](#)^[535] after **message** has been displayed will continue on the same line.

Display rules

- The first space after the command name is ignored.
- Trailing spaces in **message** are ignored.
- Functions and variables not enclosed between back quotes are evaluated.
- To include special characters, .e.g., < | >, in **message**, enclose them in double quotes or back quotes (see [Parameter Quoting](#)^[134]) or precede them with the [escape character](#)^[124], or use the /X option of the [SETDOS](#)^[323] command.
- To display % you may alternately use two % marks for each one to be displayed, e.g., %%
- To display trailing spaces, either enclose them in back quotes, or append a pair of back quotes behind them, e.g., **echo trailers** ``
- The [ASCII](#)^[511] NUL character cannot be included in **message**.
- ECHOS keeps the cursor on the same line, thus permitting building a line of display using multiple commands

ECHOS is useful for text output when you don't want to add a carriage return / linefeed pair at the end of the line. This is useful if your whole line of text requires more than one command to build, and also for controlling character devices.

3.13.38 ECHOSERR

Purpose: Display a message to the standard error device ([stderr](#)^[535]) without a trailing carriage return / line feed.

Format: ECHOSERR message

message Text to display.

See also: [ECHO](#)^[217], [ECHOS](#)^[219], and [ECHOERR](#)^[218].

Usage:

ECHOSERR acts as a combination of [ECHOS](#)^[219] and [ECHOERR](#)^[218]. It parses and expands **message**, and displays it on [stderr](#)^[535]. However, any display sent to [stderr](#)^[535] after **message** has been displayed will continue on the same line.

Display rules

- The first space after the command name is ignored.
- Trailing spaces in **message** are ignored.
- Functions and variables not enclosed between back quotes are evaluated.
- To include special characters, .e.g., < | >, in **message**, enclose them in double quotes or back quotes (see [Parameter Quoting](#)^[134]) or precede them with the [escape character](#)^[124], or use the /X option of the [SETDOS](#)^[323] command.
- To display % you may alternately use two % marks for each one to be displayed, e.g., %%
- To display trailing spaces, either enclose them in back quotes, or append a pair of back quotes behind them, e.g., `echo trailers ` ``
- The [ASCII](#)^[511] NUL character cannot be included in **message**.
- ECHOSERR keeps the cursor on the same line, thus permitting building a line of display using multiple commands

3.13.39 EJECTMEDIA

Purpose: Eject removable media in the specified drive(s)

Format: EJECTMEDIA drive ...

Usage:

EJECTMEDIA will eject removable media, such as CD-ROMs, DVDs, etc. (It is not intended for unmounting USB drives.)

See also [LOADMEDIA](#)^[269].

3.13.40 ENDLOCAL

Purpose: Restore the saved disk drive, directory, environment, alias list, and special characters, and exports selected variables.

Format: ENDLOCAL [/D] [exportvar ...]

[/D\(ont restore\)](#)^[221]

See also: [SETLOCAL](#)^[326].

Usage:

The [SETLOCAL](#)^[326] 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 user-defined function list. [ENDLOCAL](#) restores everything that was saved by the previous [SETLOCAL](#)^[326] 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](#)^[326] / [ENDLOCAL](#) may be nested within a single batch file up to 16 levels of nesting. You can also have multiple, separate [SETLOCAL](#)^[326] / [ENDLOCAL](#) pairs within a batch file, and nested batch files can each have their own [SETLOCAL](#)^[326] / [ENDLOCAL](#).

You cannot use [SETLOCAL](#)^[326] and [ENDLOCAL](#) in an alias or at the command line.

An [ENDLOCAL](#) is performed automatically at the end of a batch file, or when returning from a "[GOSUB](#)^[247] filename". If you invoke one batch file from another without using [CALL](#)^[175], 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](#)^[326].

- **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](#)^[326] / [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](#)^[227] below.

Options:

/D (Don't restore directory) Export the current directory: the original drive and directory saved by [SETLOCAL](#)^[326] will not be restored.

3.13.41 ESET

Purpose: Edit an environment variable, alias or function definition.

Format: ESET [/A /D /F /M /S /U /V /W] name

name The name of an environment variable, function or alias to edit.

/A(alias) ^[223]	/S(system variable) ^[223]
/D(default environment) ^[223]	/U(user variable) ^[223]
/F(function) ^[223]	/V(volatile variable) ^[223]
/M(master environment variable) ^[223]	/W(windowed Editing) ^[223]

See also: [ALIAS](#)^[154], [FUNCTION](#)^[242], [SET](#)^[319], [UNALIAS](#)^[355], [UNFUNCTION](#)^[356], and [UNSET](#)^[357].

Usage:

ESET allows you to edit an environment variable, alias or function definition using line editing commands (see [Command Line Editing](#)^[104]).

For example, to edit the executable file search path:

```
eset path
path=c:\;c:\dos;c:\util
```

To create and then edit an alias:

```
alias d = dir /d/j/p
eset d
d=dir /d/j/p
```

Unless a specific data type is specified by one of the option switches **/A**, **/D**, **/F**, **/M**, **/S**, **/U** or **/V**, ESET will search for **name** among environment variables first and then among aliases, thus if **name** is both a variable and an alias, ESET will edit the variable **name**, and ignore the alias **name**.

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 (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](#)^[247].

If you have enabled global aliases (see [ALIAS](#)^[154]), any changes made to an alias with ESET will immediately affect all other copies of **TCC** which are using the same alias list. Similarly, if you have enabled global functions (see [FUNCTION](#)^[242]), any changes made to a function using ESET /F will immediately affect all other copies of **TCC** 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:

```
eset /v myvar
```

Use caution when directly modifying registry variables as they may be essential to various Windows

processes and applications.

Options:

- /A** 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** Edit a user-defined function.
- /D** Edit a "default" variable in the registry (HKU\DEFAULT\Environment).
- /M** Edit the "master" environment (inherited by **TCC** at startup). Note that the master environment is only used if you run [START](#)^[33] with the **/I** option.
- /S** Edit a "system" variable in the registry (HKLM\System\CurrentControlSet\Control\Session Manager\Environment).
- /U** Edit a "user" variable in the registry (HKCU\Environment).
- /V** Edit a "volatile" variable in the registry (HKCU\Volatile Environment).
- /W** Edit the environment variable, alias, or function list in a popup window like that used by the [Batch File Debugger](#)^[166]. 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.

3.13.42 EVENTLOG

Purpose: Write a string to the Windows event log.

Format: EVENTLOG [/Cn /E /I /S source /W] *message*

message The text to write.

source The source for this message.

[/C\(ategory\)](#)^[224]

[/E\(rror\)](#)^[224]

[/I\(nformational\)](#)^[224]

[/S\(ource\)](#)^[224]

[/W\(arning\)](#)^[224]

See also: [HISTORY](#)^[251] and [LOG](#)^[269].

Usage:

EVENTLOG posts messages to the Windows application event log. Each message can be a maximum of 8,191 characters long. You cannot use the [command separator](#)^[51] character ([&]) or the [redirection](#)^[97] symbols (| > <) in an EVENTLOG message, unless you enclose the message in [quotes](#)^[134] or precede the special characters with the [escape character](#)^[124].

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:

/Cn Set the event category. The value can be from 0-999999; Windows defines 0-7 as:

- 0 - None
- 1 - Devices
- 2 - Disk
- 3 - Printers
- 4 - Services
- 5 - Shell
- 6 - System
- 7 - Network

/E Store the message as an error entry in the event log.

/I Store the message as an informational entry in the event log. This is the default if no switch is used.

/S 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 Store the message as a warning entry in the event log.

3.13.43 EVENTMONITOR

Purpose: Monitor event logs

Format: EVENTMONITOR [/C [name]]
EVENTMONITOR server name /S"source" /T"type" /D"description" n command

server	UNC name of the machine with the log file
name	log name
n	Number of repetitions (or FOREVER)
command	Command to execute when condition is triggered

[/C\(lear\)](#)^[224] [/S"source"](#)^[225]
[/D"description"](#)^[225] [/T"type"](#)^[225]

Usage:

If you don't enter any arguments, EVENTMONITOR will display the events it is currently monitoring.

The command line will be parsed and expanded before EVENTMONITOR is executed, so if you want to pass redirection characters or variables to **command** you will need to protect them (by enclosing in single back quotes, doubling the %'s, or using command grouping).

You can specify multiple **/D**, **/S**, and **/T** arguments. If you want to monitor multiple events in a log, put them into a single EVENTMONITOR command. EVENTMONITOR creates a separate thread for each EVENTMONITOR command, so if you have multiple commands you will be wasting CPU time, RAM, and risk having **command** executed simultaneously in different threads.

The monitoring runs asynchronously in a separate thread. When the condition is triggered, the

command will be executed immediately. This may cause problems if you try to write to the display or access files while the main **TCC** thread is also performing I/O. You may need to use [START](#)^[33] or [DETACH](#)^[19] in **command** to avoid conflicts.

EVENTMONITOR creates environment variables when an event is triggered that can be queried by **command**. The variables are deleted after **command** is executed.

_eventcomputer	The name of the computer than generated the event
_eventcount	The number of times the condition has been triggered
_eventdesc	The event description
_eventlog	The name of the event log
_eventsources	The name of the source that wrote the event
_eventtype	The event type (see /T ^[225] below)

Options:

/C	If name is specified, remove the monitor for that event. Otherwise, remove all event monitors.
/D	Description for the event to be monitored. Only events with a matching description will set the trigger. The description may contain regular expressions.
/S	Source for the event to be monitored. Only events with a matching source will set the trigger. The source may contain regular expressions.
/T	Type of event to be monitored. Only events with a matching type will set the trigger. The types of events are: Success Error Warning Information Audit_Success Audit_Failure

3.13.44 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 ^[90] for details).
command	The command to execute, including all appropriate parameters and switches.

[/I \(match description\)](#)^[227]

See also: [ATTRIB](#)^[163] and [File Exclusion Ranges](#)^[85].

File Selection

Supports extended [wildcards](#)^[77], [ranges](#)^[80], [multiple file names](#)^[87], and [include lists](#)^[88]. 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](#)^[89] 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:\MSC and C:\DOS, using the COPY command:

```
except (memo* *.wks) erase *
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:\:

```
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). Files which already had the hidden attribute, and are included in the set matching EXCEPT, will not be hidden after EXCEPT is completed. The hidden attribute of files not matching EXCEPT will not be changed.

Caution: EXCEPT will not work with programs or commands that ignore the hidden attribute or which work explicitly with hidden files, including [DEL](#)^[190] /Z, and the /H (process hidden files) switch available in internal file processing commands.

[File exclusion ranges](#)^[85] 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](#)^[121] 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*:

```
except (*.dat) (copy *.da* d:\save & ren *.da* *.sa*)
```

If you use filename completion (see [Filename Completion](#)^[113]) 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" Select files by matching text in their descriptions. The text can include [wildcards](#)^[77] and extended wildcards. The search text must be enclosed in double quotes, 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](#)^[90] for details.

3.13.45 EXIT

Purpose: Exit the current **TCC** session.

Format: EXIT [/B] [value]

value The numeric exit code to return.

[/B \(exit from batch file\)](#)^[227]

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:

```
exit
```

If you specify a value, EXIT will return that value to the program that started the command processor. For example:

```
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, rather than the shell. This switch is for compatibility with **CMD.EXE**. The [CANCEL](#)^[176] and [QUIT](#)^[301] commands are generally more flexible for use in batch files.

3.13.46 FFIND

Purpose: Search for files by name or contents.

Format: FFIND [/A[[:] [-]rhsadecijopt /B /C /D[/isf] /E["text"] /F /G /I /I"text" /K /L /M /N /O[[:] [-]acdeginsu] /P /R /S[n] /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\(tribute select\)](#)^[229]

[/B\(are\)](#)^[229]

[/C\(ase sensitive\)](#)^[229]

[/D\(rive\)](#)^[229]

[/M \(no footers\)](#)^[230]

[/N\(ot\)](#)^[230]

[/O\(rder\)](#)^[230]

[/P\(ause\)](#)^[230]

/E (upper case)	/R(everse search order)
/E"xx" (regular expression)	/S(ubdirectories)
/F (stop after match)	/T"xx" (text search string)
/G (goto directory)	/U (summary only)
/I(gnore wildcards)	/V (verbose)
/I"text" (match description)	/X["xx"] (hex display / search string)
/K (no headers)	/Y (prompt to stop after match)
/L(ine numbers)	/[+ -] skip matches

File Selection

Supports extended [wildcards](#), [ranges](#), [multiple file names](#), and [include lists](#).

Internet: Can be used with [FTP Servers](#).

Usage:

FFIND is a flexible search command that looks for files based on their names and their contents. Depending on the options you choose, FFIND can display filenames, matching text, or a combination of both in a variety of formats.

If you don't supply a file name, FFIND will read from standard input. (This allows you to pipe or redirect input to FFIND.)

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

```
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:

```
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:

```
ffind /r /t"Sincerely," *.txt
```

You can use **TCC** [extended wildcards](#) in the search string to increase the flexibility of FFIND's search. For example, the following command will find `.TXT` files which contain either the string `June` or `July`. It will also find `Juny` and `Jule`. The **/C** option makes the search case-sensitive:

```
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:

```
ffind /i/t"?" *.txt
```

Sometimes 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"):

```
ffind /x"41 63 65" *.txt
```

You can also search using Regular Expressions using the **/E** option. See [Regular Expression Syntax](#) ^[496] for supported expressions.

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. For example:

```
ffind /t"tcmd" 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](#) ^[93] and [IFTP](#) ^[255].

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:** Select only those files that have the specified attribute(s) set. See [Attribute Switches](#) ^[86] for information on the attributes which can follow **/A:**.
- /B** 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** 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** 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:

```
ffind /dcdeg ...
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:

```
ffind /s /dce *.btm d:\*.bat
```

- /E** Display filenames in upper case.
- /E"text"** Search for a [regular expression](#)^[496]. The regular expression must be contained in double quotes if it contains spaces, punctuation, or wildcard characters. See also **/T**.
- /F** Stops the search after the first match.
- /G** Change to the directory where the match was found (must be used with **/F**).
- /I** 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"** Select filenames by matching text in their descriptions. The text can include [wildcards](#)^[77] and extended wildcards. The search text must be enclosed in double quotes, 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** Suppress the display of the header or filename for each matching text line.
- /L** Include the line number for each text line displayed. FFIND numbers lines beginning with 1. 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](#)^[126], whichever comes first.
- /M** Suppress the footer (the number of files and number of matches) at the end of FFIND's display.
- /N** 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** 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)
 - o** Sort by owner
 - r** Reverse the sort order for all options
 - s** Sort by size
 - u** Unsorted
- /P** 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](#)^[102].

- /R** 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** 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**. If you specify a number following the **/S**, FFIND will limit the subdirectory recursion to the number specified. For example, if you have a directory tree "`\a\b\c\d\e`", **/S2** will only go to the "a", "b", and "c" directories.
- /T"text"** Specify the text search string. **/T** must be followed by a text string in double quotes (e.g., **/t"color"**). FFIND will perform a case-insensitive search unless you also use the **/C** option. For a hexadecimal search and/or hexadecimal display of the location where the search string is found, see **/X**. You can specify a search string with either **/T** or **/X**, but not both.
- /U** Display only the summary.
- /V** 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 **/E**, **/T** and **/X**.
- /X["xx.."]** 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":

```
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:

```
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.

3.13.47 FIREWIREMONITOR

Purpose: Monitor FireWire device connection and disconnection

Format: FIREWIREMONITOR [/C [name]]
FIREWIREMONITOR name CONNECTED | DISCONNECTED n command

name Device name
n Number of repetitions (or **FOREVER**)
command Command to execute when condition is triggered

[/C\(lear\)](#) ^[234]

Usage:

The FireWire device name can include wildcards. You can use either the device ID or the "friendly" name for the device.

The command line will be parsed and expanded before FIREWIREMONITOR is executed, so if you want to pass redirection characters or variables to **command** you will need to protect them (by enclosing in single back quotes, doubling the %'s, or using command grouping).

If you don't enter any arguments, FIREWIREMONITOR will display the FireWire devices it is currently monitoring.

The monitoring runs asynchronously in a separate thread. When the condition is triggered, the command will be executed immediately. This may cause problems if you try to write to the display or access files while the main **TCC** thread is also performing I/O. You may need to use [START](#) ^[337] or [DETACH](#) ^[197] in **command** to avoid conflicts.

FIREWIREMONITOR creates three environment variables when a device is connected or disconnected that can be queried by **command**. The variables are deleted after **command** is executed.

_firewiredeviceid The device ID
_firewirename The "friendly" name of the device
_firewirecount The number of times the condition has been triggered

Options:

/C If **name** is specified, remove the monitor for that FireWire device. Otherwise, remove all FireWire monitors.

3.13.48 FOLDERMONITOR

Purpose: Monitor folder and/or file creation, modification, and deletion

Format: FOLDERMONITOR [/C [folder]]
FOLDERMONITOR /S folder /I"file" /E"file" CREATED DELETED MODIFIED RENAMED
n command

folder	Folder (directory) name
CREATED	Execute the command if the folder or file is created
DELETED	Execute the command if the folder or file is deleted
MODIFIED	Execute the command if the folder or file is modified
RENAMED	Execute the command if the folder or file is renamed
n	Number of repetitions (or FOREVER)
command	Command to execute when condition is triggered

[/C\(lear\)](#)^[234]
[/E\(xclude\)](#)^[234]

[/I\(include\)](#)^[234]
[/S\(ubdirectories\)](#)^[232]

Usage:

If you don't enter any arguments, FOLDERMONITOR will display the folders and files it is currently monitoring, in the format:

```

    folder      (include/exclude)    condition    (n)    command

```

The command line will be parsed and expanded before FOLDERMONITOR is executed, so if you want to pass redirection characters or variables to **command** you will need to protect them (by enclosing in single back quotes, doubling the %'s, or using command grouping).

The **MODIFIED** condition is set if the file's size, attributes, or last access date and time are changed.

If you want to monitor multiple conditions for a file or folder, put them into a single FOLDERMONITOR command. FOLDERMONITOR creates a separate thread for each FOLDERMONITOR command, so if you have multiple commands you will be wasting CPU time, RAM, and risk having **command** executed simultaneously in different threads.

When the condition is triggered, the command will be executed immediately in the separate thread. This may cause problems if you try to write to the display or access files while the main **TCC** thread is also performing I/O. You may need to use [START](#)^[337] or [DETACH](#)^[197] in **command** to avoid conflicts.

FOLDERMONITOR creates several environment variables when a file or folder is created, deleted, modified, or renamed that can be queried by **command**. The variables are deleted after **command** is executed.

_folderaction The type of change to the file or folder. The possible values are:

```

    CREATED
    DELETED
    MODIFIED
    RENAMED

```

_foldercount The number of times the condition has been triggered

_foldername The name of the folder being monitored

_folderfile1 The name of the file or folder that was created/deleted/modified/renamed. If the file was renamed, **folderfile1** is the old name.

_folderfile2 If a file was renamed, **folderfile2** is the new name

For example, to monitor your **d:\results** directory and copy any new or modified files to a web page:

```
foldermonitor d:\results created modified forever copy "%_folderfile1"
```

"http://mycompany.com/results/"

Options:

- /C** If **name** is specified, remove the monitor for that folder. Otherwise, remove all folder / file monitors. **/C** cannot be combined with any other options.
- /E** Filename to be excluded. If you want to exclude multiple files, use multiple **/E** options. If you want to exclude a file in a specific subdirectory, the filename should include the relative path from the folder **name**. The name can include wildcards.
- /I** Filename to be included. If you want to include multiple files, use multiple **/I** options. If you want to include a file in a specific subdirectory, the filename should include the relative path from the folder **name**. The name can include wildcards.
- /S** Include subdirectories.

3.13.49 FOR

Purpose: Repeat a command for several values of a variable.

Format: File and string mode
 FOR [range...] [/I"text"] [/A:[-+]rhsadecijopt /D /F ["options"] /H /R [path] [/T"delimiters"]]
 %var IN ([@]set) DO command | (command ... [LEAVEFOR])

Counted mode
 FOR /L %var IN (start, step, end) DO command | (command ... [LEAVEFOR])

options	Parsing options for a "file parsing" FOR.
range	One or more range ^[240] specifications
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) ^[240]	/I description range ^[240]
/D(irectories only) ^[240]	/L (counted loop) ^[240]
/F(ile parsing) ^[240]	/R(ecursive) ^[240]
/H(ide dots) ^[240]	/T (delimiter list) ^[241]

File Selection

Supports [attribute switches](#) ^[86], extended [wildcards](#) ^[77], [ranges](#) ^[80], [multiple file names](#) ^[87], and [include lists](#) ^[88].

Ranges must appear immediately after the FOR keyword after alias expansions (if any), and only affect the selection of files specified using wildcards.

Use wildcards with caution on LFN volumes; see [LFN File Searches](#) ^[89] for details.

Usage:

FOR begins by creating a **set**. It then set the executes a command for every member of **set**. The

command can be an internal command, an alias, an external command, or a batch file. The members of **set** can be a list of file names, text strings, a group of numeric values, or text read from a list of files.

When **set** is made up of text or several separate file names (not an include list), the elements must be separated by spaces, tabs, or commas.

FOR includes a large number of options, some of which duplicate functions available in other internal commands. It also supports additional conventions not found in our other commands, included for compatibility with **CMD.EXE**.

The first three sections below ([Working with Files](#)^[235], [Working with Text](#)^[236], and [Retrieving Text from Files](#)^[236]) describe the FOR command and the enhancements to it which are included in **TCC**. The sections on [Parsing Text from Files](#)^[236] and [Counted FOR Loops](#)^[237] describe features added for compatibility with **CMD.EXE**. The sections [Directory Recursion](#)^[238] and [Output Redirection](#)^[238] warn of special considerations. The section entitled [Other Notes](#)^[238] contains information you may need if you use any aspect of the FOR command extensively.

Working with Files

Normally, **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](#)^[264] 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.)

Set can include multiple files and include lists, like this:

```
for %x in (d:\*.txt;*.doc;*.asc e:\test\*.txt;*.doc) type %x
```

FOR supports [wildcards and extended wildcards](#)^[77], as well as [extended parent directory](#)^[118] names, e.g., ... *.txt to process all of the **.TXT** files that are contained in the directory 2 levels above the current directory.

By default those members of **set** that include wildcards match only files, not directories.

When you use FOR on an LFN drive, you must quote any file names within set which contain white space or special characters. The same restriction may apply to names returned in the FOR variable, if you pass them to **TCC** 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 set.

If set includes filenames, the file list can be further refined by using date, time, size, description and file exclusion [ranges](#)^[80]. The range or ranges must be placed immediately after the word FOR. Ranges affect only those members of set which contain wildcards. For example, the FOR below will process all of the **.TXT** files that were created or updated on December 4, 2005, and of the file ABC.LST regardless of its timestamp:

```
for [/d12-4-2005,+0] %x in (*.txt abc.lst) ...
```

If **command** is an internal command that supports ranges, an independent range can also be used in **command** itself.

You can also refine the list by limiting it with the [/A:](#)^[240] option to select only files that have specific attributes.

When you use wildcards to specify **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 new file that could be included in **set**, it may or may not appear in a some later iteration of the same FOR command. Whether or not 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 a Windows constraint over which **TCC** 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 set for the current command.

Working with Text

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](#) ^[365] 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

If the name of a file in **set** is prefixed with @ ("at" sign), it is considered as an [@file list](#) ^[90]. FOR extracts each line from the file and places it in the FOR variable.

Warning: if the line contains characters which are syntactically significant for **TCC**, for example, one of the characters `<"[]|>`, it may have undesirable effects. You may use the /X option of [SETDOS](#) ^[323] to mitigate them.

If you use **@CON** as the filename, FOR will read from standard input (typically 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](#) ^[97] for more information on these features.

See [@file list](#) ^[90] for additional details.

Parsing Text from Files

Another 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](#) ^[240] option and include one or more file names in set. When you use this form of FOR, the variable name 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](#) ^[428] as the **set** to retrieve each line from the file, as described in the previous section, and use variable functions like [@FIELD](#) ^[426], [@INSTR](#) ^[443], [@LEFT](#) ^[446], [@RIGHT](#) ^[457], and [@WORD](#) ^[472] to parse the line (see [Variable Functions](#) ^[395] 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) ...
```

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.

usebackq : Duplicates the awkward **CMD.EXE** syntax. A back quoted string is executed as a command; a single quoted string is a literal string; and double quotes quote filenames in the file set. We don't recommend **usebackq** for batch files written for **TCC**, as **TCC** has much more elegant ways of doing the same things.

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 **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 for compatibility with **CMD.EXE**. In most cases, you will find the [DO](#)^[210] 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](#)^[240] option and then include three values, separated by commas, in **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.

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).

WARNING! You must not have white space between *start* and the subsequent comma, nor between *step* and its subsequent comma. White space after the comma is accepted.

Directory Recursion

By default, FOR works only with files in the current directory or a specified directory. Option switch [/R](#)^[240] specifies that the search should recursively process subdirectories. If you specify a directory name immediately after [/R](#)^[240], FOR will start in that directory and then search each of its subdirectories. If no directory is specified after the [/R](#), the search starts in the current default directory. If you do specify a directory, *and* its name includes any special characters, it must be enclosed in double quotes. For example, it must be quoted if it is specified with the aid of an environment variable, e.g., `%windir\command` .

There are two differences in the invocation of **command** caused by directory recursion:

- The loop control variable contains the full name of the matching file
- **command** is executed with the default directory set to the directory in which the file was found

This example processes all `.TXT` files in the current directory and its subdirectories:

```
for /r %x in (*.txt) ...
```

This example works with all of the `.BAK` files on drive **D**:

```
for /r d:\ %x in (*.bak) ...
```

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 file path, it will be created relative to the then current default directory. If you use directory recursion, this path will change for each directory processed. 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 (**var**) in the command. 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** which requires a single % if FOR is used at the command line, but requires %% if FOR is used in a batch file.) Note that you must have at least one % sign present.
- The variable name can be up to 80 characters long.
- If the FOR command is an alias, e.g., `alias for=*for /h, range`^[80] specifications will be ignored.
- The word DO is unnecessary but accepted. Do not confuse it with the completely unrelated [DO](#)^[210] command.
- If the name of the FOR variable **var** is a single character, for compatibility with **CMD.EXE**, it is created in the environment in a special way that does not overwrite an existing environment variable with the same name. Wherever **command** contains the % sign immediately followed by the character which is the name of the FOR variable, it is replaced by its value, regardless of any characters following it. For example, the following command tries to add **a:** and **b:** to the end of [PATH](#)^[368], but will not work as intended:

```
for %p in (a: b:) path %path;%p
path
b:ath;b:
```

The **%p** in **%path** was interpreted as the FOR variable **%p** followed by the text **ath**, 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, as shown in the examples below, any one of which accomplishes the original goal:

```
for %p in (a: b:) path %[path];%p
for %x in (a: b:) path %path;%x
for %px in (a: b:) path %path;%px
```

- If the name of the FOR variable contains more than one character, it is created in the environment, and erased when FOR is completed, whether or not a variable by that name existed before the FOR. It cannot be modified with the [SET](#)^[319], [ESET](#)^[222], or [UNSET](#)^[357] commands. If you already had a variable with that name, it will no longer be accessible. For example, a command that begins

```
for %path in ...
```

will write over your current [PATH](#)^[368] setting, then erase the [PATH](#)^[368] variable completely when FOR is done.

- **Command** may also use the FOR variable with the special syntax of **CMD.EXE** described in [Special syntax for CMD.EXE compatibility](#)^[132].
- 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](#)^[190] and one for [%@NAME](#)^[451]:

```
for %file in (*.txt) del "%@name[%file]".bak"
```

- You can use [command grouping](#)^[121] to execute multiple commands for each element in **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**:

```
[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](#)^[247] to execute a subroutine for every element in **set**. Within the subroutine, the FOR variable can be used just like environment variable. This is a convenient way to execute a complex sequence of commands for every element in **set** without [CALL](#)^[175]ing 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:

```
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.

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](#)^[210] command.

```
for %i in (*) (
    ...
    if "%i" == "xyz.abc" leavefor
    ...
)
```

Options:

/A: Process only those files that have the specified attribute(s). **/A:** will be used only when processing wildcard file names in **set**. It will be ignored for filenames without wildcards or other items in **set**. See [Attribute Switches](#)^[86] 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!
```

Default: **/A:-D-H-S**, i.e. include only **files** without the **hidden** and **system** attributes.

/D Only return subdirectories, excluding "." and ".." .

/F Return one or more words or tokens from each line of each file in **set**. The **/F** option can be followed by one or more options in a quoted string which control how the parsing is performed. See [Parsing Text From Files](#)^[236].

/H Suppresses the assignment of the "." and ".." directories to the FOR variable when directories are explicitly included using the **/A:**^[240] option.

/I "text" Select filenames by matching text in their descriptions. See [Description Ranges](#)^[86].

/L Interpret the three values in **set** as the **start**, **step**, and **end** values of a counted loop. See [Counted FOR Loops](#)^[237].

/R [path] Look in the current directory and all of its subdirectories for files in **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 double quotes (").

/T"text" Specify the delimiters to be used when parsing a string set.

3.13.50 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](#)^[272].

Usage:

A colon [:] is required after each drive letter. This example displays the status of drives A and C:

```
free a: c:
```

If the volume serial number is available, it will appear after the drive label or name.

FREE supports [OpenAFS](#)^[103] names.

3.13.51 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\(ause\)](#)^[242]

[/R\(ead from file\)](#)^[242]

See also: [ASSOC](#)^[162], and [Executable Extensions](#)^[91].

Usage

FTYPE allows you to display or update the command used to open a file of a specified type stored in the Windows registry.

FTYPE modifies the behavior of Windows file associations stored under the registry handle HKEY_CLASSES_ROOT, and discussed in more detail under [Windows File Associations](#)^[506]. 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. If the value contains the percent mark character %, the value stored will be type REG_EXPAND_SZ, otherwise it will be type REG_SZ.

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** 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](#)^[102].
- /R** 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](#)^[535]. The format of the file is the same as that of the FTYPE display.

You can insert comments in the file by prefixing the line with a colon (:).

3.13.52 FUNCTION

Purpose: Create, modify or display user-defined variable functions

Format: Display mode:
FUNCTION [/G /L /P] [wildname]

Direct definition mode:
FUNCTION name[=]definition

Definition file mode:
FUNCTION /R [file...]

file	One or more input files to read function definitions from.
wildname	Name of function whose definition is to be displayed (may contain * and ? wildcards)
name	The name of the function you want to define.
definition	The value or definition of what the function should return.

[/G\(lobal\)](#)^[245]
[/L\(ocal\)](#)^[244]

[/P\(ause\)](#)^[245]
[/R\(ead file\)](#)^[245]

See also: [UNFUNCTION](#)^[356] and [ESET](#)^[222].

Usage:

- ▶ [Overview](#)^[243]
- ▶ [Displaying Functions](#)^[243]
- ▶ [Defining Functions](#)^[243]

- ▶ [Deleting Functions](#) ^[244]
- ▶ [Local and Global Functions](#) ^[244]
- ▶ [Saving and Reloading Your Functions](#) ^[244]
- ▶ [Warnings](#) ^[245]

Overview

FUNCTION allows you to create or display user-defined variable functions that can be used anywhere [Variable Functions](#) ^[395] can be used. User-defined functions are powerful alternatives to [subroutines](#) ^[247].

Displaying Functions

If you invoke the FUNCTION command with no parameters, it will display the current function list:

```
function
```

If you include a **wildname**, which may include wildcards (* or ?), with no equal sign and no **definition**, FUNCTION will display the current values, if any, of all functions matching **wildname**, e.g.:

```
function *dx*
```

will display all functions which contain **dx** in their name.

You can use the [/P](#) ^[245] option to control display scrolling when displaying functions.

Defining Functions

If you include the equal sign and **definition**, FUNCTION will create or update the function referred to by **name**. Any previous **definition** associated with **name** is discarded. Instead of the = sign, you may use one or more spaces or tab characters to separate **name** and **definition**.

Once a function is defined, the definition may be edited using [ESET](#) ^[222] /F.

A function can optionally use references to parameters numbered from %0 to %511 which will be replaced with the matching parameter value when the function is called. %0 refers to the function name, %1 to the first parameter, etc. For example, the function

```
function leftmost=`%@left[1,%1]`
```

will return the leftmost character in its parameter, e.g. `%@leftmost[xyz]` will return `x`.

The parameter **%n\$** has a special meaning. **TCC** interprets it to mean "the entire command line, from parameter **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 parameter **%-n\$** means "the command line from parameter 1 to **n** - 1".

The special variable reference **%#** expands to the number of parameters passed to the function.

A function definition need not reference any parameters at all. For example:

```
function tomorrow=`%@makedate[%@inc[%@date[%_date]]]`
```

could be simply invoked as `%@tomorrow[]`.

To use the function **name** you invoke is as `%@name[parameters]`, where you must specify enough parameters to assign a value to the highest numbered parameter **referenced** in the function definition. It may have more parameters, which will be silently ignored.

The [Colors, Color Names and Codes](#)^[518] topic shows a simple example of the use of a function in a batch file.

Deleting Functions

The normal method is to use the [UNFUNCTION](#)^[356] command. However, it is also possible to delete a function by redefining it without a **definition**, e.g., the command

```
function fs=
```

deletes the function **fs**.

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 **TCC**. They will not be visible in other shells or other sessions.

With a global function list, all copies of **TCC** 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 **TCC**.

You can control the type of function list with the [Local Functions](#)^[47] configuration option, with the **/L** and **/LF** options of the [START](#)^[331] command, and with the **/L** and **/LF** [startup options](#)^[19].

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 **TCC** 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.

Saving and Reloading Your Functions

You can save your functions to a file (e.g., `FUNCTIONS.LST`) this way:

```
function > function.lst
```

You can then reload all the function definitions in the file the next time you start up with the command:

```
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 **TCC** 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 **TCC**.

When you define functions in a file that will be processed by the **FUNCTION /R** command, you do not need back quotes around definition, even if back quotes would normally be required when defining the same function at the command line or in a batch file.

Warnings

When you define a function in the command line (i.e., without using the [/R](#)^[245] option), variables and functions not protected by back quotes or doubled % signs are immediately evaluated, and the result becomes part of the function definition.

Syntax errors in a function definition are not detected until it is used.

Options:

- /G** Switch from a local to a global function list.
- /L** Switch from a global to a local function list.
- /P** Wait for a key to be pressed after each screen page before continuing the display.
- /R** This option loads a list of functions from a file. If no filename is specified and input is redirected, /R will read from [stdin](#)^[535]. The format of the file is the same as that of the FUNCTION display:

name=definition

where ***name*** is the name of the function and ***definition*** specifies how to determine its value. You may use the equal sign = or whitespace to separate ***name*** and ***definition***. Back-quotes are not required.

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:

```
function /r func1.lst func2.lst
```

Each definition in a FUNCTION /R file can be up to 8,191 characters long. The definitions can span multiple lines in the file if each line, except the last, is terminated with an [Escape Character](#)^[124].

If there is no filename parameter and input is redirected, FUNCTION /R will read from [stdin](#)^[535].

3.13.53 GLOBAL

Purpose: Execute a command in the current directory and its subdirectories.

Format: GLOBAL [/H /I /J /N /P /Q /Sn] command

command The command to execute, including parameters and switches.

/H(idden directories) ^[246]	/P(rompt) ^[247]
/I(gnore exit codes) ^[246]	/Q(quiet) ^[247]
/J (only junctions) ^[246]	/S(ubdirectory depth) ^[247]
/N(o junctions) ^[246]	

Usage:

GLOBAL performs **command** first in the current directory. Then it makes every subdirectory under the current directory the current working directory in turn, and performs **command** in that directory.

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\[\]](#)^[464], [@full\[\]](#)^[437], etc to make sure that files of the same name in different directories are correctly handled.

The example below 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](#)^[182] is the default internal command, the one found last will be left in **C:\TEMP**. Which one of multiple, identically named files is found last is unpredictable!

If you use the [/P](#)^[247] option, GLOBAL will prompt for each subdirectory before performing **command**. You can use this option if you want to perform **command** in most, but not all subdirectories of the current directory.

You can use [command grouping](#)^[127] 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 **D**. It then changes the extension of each of the copied files to **.SAV**:

```
global (copy *.txt d: & ren *.txt *.sav)
```

Output Redirection

The default output redirection (i.e., **global command > filename**) creates a new output file named **filename** as each directory visited. If **filename** does not include an absolute file path, these files will be created relative to the currently visited directory. If **filename** does include an absolute file path, that file will be overwritten as each directory is visited, and only the data from the last visited directory will survive.

The simplest way to force a single target file is to enclose the whole command line in parentheses, e.g.,:

```
(global command) > filename
```

Options:

- /H** Forces GLOBAL to look for hidden directories. If you don't use this switch, hidden directories and their subdirectories are ignored without error indication.
- /I** If this option is not specified, GLOBAL will terminate if **command** returns a non-zero exit code. Use **/I** if you want **command** to continue in additional subdirectories even if it returns an error in one subdirectory. GLOBAL will normally halt execution if **TCC** receives a **Ctrl-C** or **Ctrl-Break** even if you use **/I**.

Without this option, if GLOBAL is unable to change to a directory (for example, if user does not have access rights), GLOBAL 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.

- /J** Forces GLOBAL to only recurse through Junctions, not subdirectories.
- /N** Forces GLOBAL to ignore Junctions and only recurse through subdirectories.

- /P** Forces GLOBAL to prompt with each directory name before it performs **command** in that directory. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[102].
- /Q** Do not display the directory names as each directory is processed.
- /S** GLOBAL will limit the subdirectory recursion to the number specified. For example, if you have a directory tree "\ab\c\d\e", /S2 will only go to the "a", "b", and "c" directories.

3.13.54 GOSUB

Purpose: Execute a subroutine in the current batch file.

Format: GOSUB ["filename"] label [variables]

filename	The file containing the subroutine
label	The batch file label at the beginning of the subroutine.
variables	Optional GOSUB variables.

See also: [CALL](#)^[175], [GOTO](#)^[248], and [RETURN](#)^[307].

Usage:

GOSUB can only be used in batch files.

TCC 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](#)^[307] 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](#)^[307] call.

If you define GOSUB variables on the label but do not supply them on the GOSUB line, they will be set to an empty string.

GOSUB calls with variables are limited to a maximum of 22 levels deep. There is no 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](#)^[319], [ESET](#)^[222], or [UNSET](#)^[357] commands, and cannot be used with the DEFINED test of [IF](#)^[253], [IFF](#)^[254], or [@IF](#)^[439].

You cannot use [SET](#)^[319] 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.

You can call a subroutine in another file by specifying **filename** (the name must be enclosed in double quotes). This allows you to create libraries of subroutines, without having to duplicate them in each batch file. For example:

```
gosub "c:\library\batlib.btm" Evaluate [%1 %2 %3]
```

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. If the subroutine has executed a SETLOCAL without a matching ENDLOCAL, an ENDLOCAL will be executed before returning to the calling batch file.

You cannot [RETURN](#)^[307] from a GOSUB while inside a [DO](#)^[210] loop.

If **TCC** 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](#)^[307] command had been included as the last line of the file.

Subroutines can be nested.

See also: [user-defined functions](#)^[242].

3.13.55 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\(FF and DO continue\)](#)^[249]

See also: [GOSUB](#)^[247], [CALL](#)^[175].

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 following 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](#)^[254] statements and [DO](#)^[210] / 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](#)^[301] or [CANCEL](#)^[176] command to end a batch file.

Option:

/I 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](#)^[154] alias) on a subsequent ENDDO, ELSE, ELSEIFF, or ENDIFF statement.

3.13.56 HEAD

Purpose: Display the beginning of the specified file(s).

Format: HEAD [/A:[-][+]*rhsadecijopt*] /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](#)^[90] for details).

[/A: \(Attribute select\)](#)^[250]

[/C \(number of bytes\)](#)^[250]

[/I"text" \(match description\)](#)^[250]

[/N\(umber of lines\)](#)^[250]

[/P\(ause\)](#)^[251]

[/Q\(quiet\)](#)^[251]

[/V\(erbose\)](#)^[251]

See also: [LIST](#)^[264], [TAIL](#)^[339], and [TYPE](#)^[354].

File Selection

Supports [attribute switches](#)^[86], extended [wildcards](#)^[77], [ranges](#)^[80], [multiple file names](#)^[87], and [include lists](#)^[88].

Internet: Can be used with [FTP/HTTP Servers](#)^[93], 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*:

```
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](#)^[67] for additional information on CLIP:.

FTP Usage

HEAD can also display files on [FTP/HTTP Servers](#)^[93]. For example:

```
head ftp://jpsoft.com/index
```

NTFS File Streams

HEAD supports file streams on NTFS drives. You can type an individual stream by specifying the stream name, for example:

```
head streamfile:s1
```

Options:

- /A:** Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[86] for information on the attributes which can follow **/A:**. Do not use **/A:** with @file lists. See [@file lists](#)^[90] 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 double quotes, 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](#)^[90] for details.
- /N n** Display **n** lines. The default is 10.

- /P** Pause and prompt after displaying each page.
- /Q** 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** Display a header for each file.

3.13.57 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](#)^[477].

Usage:

Online help is available for all of **TCC**'s internal commands, variables, and other features.

The **TCC** help system (**tcmd.chm**) uses Microsoft's HTML Help Viewer (**HH.EXE**) included in all their current Windows bundles.

If you type the command **HELP** by itself (or press **F1**^[34] 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, internal variables, variable functions, and key mapping 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**^[34] (this can be useful when you need the syntax for a variable function).

3.13.58 HISTORY

Purpose: Display or modify the history list.

Format: HISTORY [/A *command* /F /G /L /N /P /R *filename*]

command
filename

A command to be added to the history list.

The name of a file containing entries to be added to the history list.

/A(dd) ^[252]	/N(o duplicates) ^[253]
/F(ree) ^[252]	/P(ause) ^[253]
/G(lobal) ^[253]	/R(ead) ^[253]
/L(ocal) ^[253]	

See also: [DIRHISTORY](#)^[208], [HistoryExclude](#)^[368] and [LOG](#)^[269].

Usage:

TCC keeps a list of the commands you have entered on the command line. See [Command History](#)

[and Recall](#)^[106] 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.

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 [Command History Buffer Size](#)^[50] configuration option). The default size is 8192 characters.

Your history list can be stored either locally (a separate history list for each copy of **TCC**) or globally (all copies of **TCC** share the same list). For full details see [local and global history](#)^[108].

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](#)^[269] command for this purpose.

You can disable the history list or specify a minimum command line length to save with the [Minimum Length](#)^[50] configuration option. 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](#)^[368] variable.

You can control whether duplicate entries will be saved in the history list with the [Duplicates](#)^[50] configuration option.

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:

```
history > histfile
history /f
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 *TCSTART.BTM* or other startup file, and the second in *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 **TCC** exits.

TCC can also load and save the history list automatically if you use the [History File](#)^[50] configuration option.

Options:

- /A** Add a command to the history list. This performs the same function as the **Ctrl-K** key at the command line.
- /F** Erase all entries in the command history list.

- /G** Switch from a local to a global history list.
- /L** Switch from a global to a local history list.
- /N** Removes duplicate entries (oldest first) from the history list.
- /P** Wait for a key after displaying each page of the list. Your options at the prompt are explained in detail under [Prompts](#)^[102].
- /R** Read the command history from the specified file and append it to the history list currently held in memory.

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](#)^[124]. However, you cannot use this method to exceed the command line length limit.

3.13.59 IF

Purpose: Execute a single command if a condition is true.

Format: IF [/I] condition command
IF [/I] condition (command1) ELSE (command2)

condition	A conditional expression ^[109]
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\(gnore case\)](#)^[254]

See also: [Conditional expressions](#)^[109], [IFF](#)^[254], [@IF](#)^[439].

Usage:

IF is most often used only aliases and batch files. It is always followed by a **condition** (see [Conditional expressions](#)^[109]), and then a **command**. First [condition](#)^[109] is evaluated, and if it is TRUE, **command** is executed. Otherwise, **command** is ignored.

If the condition is FALSE, **IF** returns a non-zero result, so it can be evaluated by one of the conditional expression operators (**||** or **&&**).

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](#)^[121]). If **condition** is TRUE, **command1** is executed, if FALSE, **command2** is executed. **Note:** this syntax is much less powerful than the [IFF](#)^[254] command, which is recommended.

When an IF test fails, the remainder of the command is discarded. Whether **TCC** continues with the next command on the line, or discards the rest of the line and goes to the next line is dependent upon the [Duplicate CMD.EXE Bugs](#)^[47] configuration option. CMD.EXE will discard all remaining commands on the line when an IF test fails, including those after a command separator or pipe character. If you do not want to reproduce CMD.EXE's behavior of an IF affecting all commands on a line, set **DuplicateBugs** to **No** in the .INI file.

For example, if [Duplicate CMD.EXE Bugs](#)^[47] is enabled (the default), the following command will display nothing, because the second ECHO command is discarded along with the first when the condition fails. If [Duplicate CMD.EXE Bugs](#)^[47] is disabled, it will display "hello":

```
[c:\] if 1 == 2 echo Wrong! & echo hello
```

Option:

- /I** This option is included only for compatibility with CMD.EXE. It has no effect in **TCC**, since all string comparisons are case-insensitive unless you specify a case-sensitive test (EQC).

3.13.60 IFF

Purpose: Perform one of several alternate sets of commands based on the values of conditional expressions.

Format: IFF condition1 THEN
 commandset1
 [ELSEIFF condition2 THEN
 commandset2]
 ...
 [ELSE
 commandset3]
 ENDIFF

condition1,2,3

commandset1

commandset2

commandset3

[Conditional expressions](#)^[109]

One or more commands to execute if **condition1** is TRUE

One or more commands to execute if **condition1** is FALSE, but **condition2** is TRUE.

One or more commands to execute if both **condition1** and **condition2** are FALSE.

See also: [IF](#)^[253] and [@IF](#)^[439].

Usage:

IFF is similar to [IF](#)^[253], but it can perform one **commandset** when a [conditional expression](#)^[109] 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](#)^[109]s, and execute the **commandset** associated with the first TRUE [conditional expression](#)^[109], 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](#)^[120]:

- 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 separators](#)^[120], 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](#)^[248] and IFF under the [GOTO](#)^[248] command before using a [GOTO](#)^[248] inside an IFF statement.

If you [pipe](#)^[97] 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 parameter is a subdirectory. If so, the alias deletes the subdirectory's files and removes it (enter this on one line):

```
alias prune `iff isdir %1 then & del /s /x /z %1 & else & echo %1 is not
a directory! & endiff`
```

3.13.61 IFTP

Purpose: Open or close an FTP/FTPS session

Format: IFTP [/S command /C /N /Pn /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) ^[257]	/Q(quiet) ^[257]
/N(o paths) ^[257]	/S(end) ^[257]
/P(assive) ^[257]	/V(erbose) ^[257]

Usage:

Most file processing commands and functions in **TCC** 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 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.

For example, to open an FTP connection using IFTP:

```
iftp ftp://user:pwd@jpsoft.com/dir1
```

For an FTPS connection, use something like:

```
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 double quotes 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 parameters 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](#)^[198] commands, assuming the IFTP session started above:

1. `dir "ftp:*.txt"`
2. `dir "ftp:dir2/*.txt"`
3. `dir "ftp:/*.txt"`
4. `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), **TCC** 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 **TCC** tab window. 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**:

```
dir "ftp://jpsoft.com/*"
```

An IFTP session remains open until you explicitly close it with this command:

```
iftp /c
```

Most FTP servers "time out" after a period of inactivity. **TCC** will attempt to detect if the connection has been closed by the server, and reconnect if you reference the IFTP session again. 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. You can determine if the connection is still active with the [_iftp](#)^[388] and [_iftps](#)^[388] variables.

IFTP and the other FTP features of **TCC** 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 **TCC** expects, commands

may not work as you intend. We urge you to test each server you use with nondestructive commands like [DIR](#)^[198] 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 Internet.

Options:

- /C** Use this switch, with no URL, to close an IFTP session (see the example above).
- /N** 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 for which the default processing fails to build a suitable name.
- /P** /P0 disables passive mode; /P1 enables it
- /Q** Turn off the display of the conversation with the FTP server.
- /S** Allows you to send commands directly to an FTP server. The connection must have already been opened by a previous IFTP command.
- /V** Display the dialog with the FTP server while opening the connection. This can be useful for debugging connection problems.

See [FTP Servers](#)^[93] for additional information on formatting and usage of FTP and FTPS references.

3.13.62 INKEY

Purpose: Get a single keystroke from the user and store it in an environment variable.

Format: INKEY [/C /D /K"keys" /P /M /Wait /X] [prompt] %%varname

prompt Optional text that is displayed as a prompt.
varname The variable that will hold the user's keystroke.
wait Time to wait for a keystroke, in seconds

/C ^[258] Clear buffer	/P ^[259] Password
/D ^[258] Digits only	/W ^[259] Wait
/K ^[258] valid keystrokes	/X ^[259] no carriage return
/M ^[259] Mouse buttons	

See also: [INPUT](#)^[259].

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](#)^[259] command, INKEY allows great flexibility in reading input from within a batch file or alias.

If **prompt** is included in an INKEY command, it is displayed while INKEY waits for input.

The following batch file fragment prompts for a character and stores it in the variable **NUM**:

```
inkey /D 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 [KEYSTACK](#)^[262]. You can supply a list of valid keystrokes with the [/K](#)^[258] 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).

A standard keystroke is stored directly in the environment variable. An extended keystroke (for example, a function key or a and cursor key) is stored as a string, consisting of a leading @, followed by its scan code as a decimal number, e.g., the **F1** key is stored as **@59**. The **Enter** key is stored as an extended keystroke **@28**. See [ASCII, Key Codes, and ANSI X3.64 Commands](#)^[510] for scan codes.

When the [/M](#)^[259] option enables recognition of mouse buttons, (and [/W](#)^[259] is not specified), the variable is set to a single character with one of the codes below:

<i>button</i>	<i>code</i>
left	240
middle	498
right	497

To test for a non-printing value returned by INKEY use the [@ASCII](#)^[409] function to get the numeric value of the key, or convert the expected value of the code to a code using [@CHAR](#)^[411]. For example, to test for **Esc**, which has an [ASCII](#)^[511] value of 27 or a left mouse button:

```
inkey Enter a key: %%key
if "%@ascii[%key]" == "27" echo Esc pressed
if %key EQ 240 echo Left mouse button clicked
```

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** with the [ON BREAK](#)^[282] command.

INKEY works within the command line window. If you prefer to use a dialog for user input, see the [MSGBOX](#)^[279] and [QUERYBOX](#)^[300] commands.

Options:

- /C** 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** Only accept numbers from **0** to **9**.
- /K "keys"** Specifies the permissible keystrokes. The list of valid keystrokes should be enclosed in double quotes. For alphabetic keys the validity test is not case sensitive. You can specify extended keys by enclosing their names in square brackets (within the quotes), for example:

```
inkey /k"ab[Ctrl-F9]" Enter A, B, Ctrl-F9 %%var
```

See [Keys and Key Names](#)^[515] 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, **TCC** will echo the keystroke if possible, beep, move the cursor back one character, and wait for another keystroke.

- /M** Enabled only if Windows' Quick Edit is disabled (alt-space -> Properties -> Options).
- /P** Prevents INKEY from echoing the character.
- /W** 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. If /W is specified, mouse buttons are ignored.

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** Prevents INKEY from displaying a carriage return and line feed after the user's entry.

3.13.63 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\(lear buffer\)](#)^[260]

[/D\(igits only\)](#)^[260]

[/E\(dit\)](#)^[260]

[/L\(ength\)](#)^[260]

[/N\(o colors\)](#)^[260]

[/P\(assword\)](#)^[260]

[/W\(ait\)](#)^[260]

[/X \(no carriage return\)](#)^[260]

See also: [SET](#)^[319], [INKEY](#)^[257], [KEYSTACK](#)^[262], [MSGBOX](#)^[279], and [QUERYBOX](#)^[300].

Usage:

INPUT optionally displays a prompt, then waits for your entry and stores it in an environment variable. INPUT is normally used in batch files and aliases to get multi-character input (for single keystroke input, see [INKEY](#)^[257]).

INPUT works within the command line window. If you prefer to use a dialog for user input, see the [MSGBOX](#)^[279] and [QUERYBOX](#)^[300] 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](#)^[262].

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](#) ^[282] command.

You can [pipe](#) ^[101] 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 **TCC** used to start the pipe.

Options:

- /C** Discard any keystrokes pending in the keyboard buffer before INPUT begins accepting characters.
- /D** Only accept numbers from 0 to 9.
- /E** 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** 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** Disables the use of input colors defined in the [Colors](#) ^[49] configuration options, and forces INPUT to use the default display colors.
- /P** Tells INPUT to echo asterisks, instead of the characters you type.
- /W** 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** Prevents INPUT from adding a carriage return and line feed after the user's entry.

3.13.64 JABBER

Purpose: Send an IM via the JABBER network.

Format: JABBER [/S(erver) /U(ser) /P(assword)] /B target[@server] message

message The message to send

[/B\(uddy\)](#) ^[261]
[/P\(assword\)](#) ^[261]

[/S\(erver\)](#) ^[261]
[/U\(sername\)](#) ^[261]

Usage:

If /S, /U, and/or /P are not specified, JABBER will use the default values defined in the .INI file (JabberServer, JabberUser, and JabberPassword).

JABBER is intended to send single short messages on an event (for example, when a large series of file transfers is completed), not as a general replacement for an interactive IM client.

Before using JABBER, you will need to create an account on a JABBER network server. See www.jabber.org for more information on the JABBER network and for open JABBER servers.

Options:

/B Address where the message will be sent
/P Logon password on the JABBER server
/S JABBER server to log onto
/U User logon name on the JABBER server

3.13.65 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\(aps lock\)](#)^[26↑] [/S\(croll lock\)](#)^[26↑]
[/N\(um lock\)](#)^[26↑]

Usage:

Most keyboards have 3 toggle keys, the Caps Lock, Num Lock, and Scroll Lock. KEYBD 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:

```
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 Turn the Caps Lock key on or off.
/N Turn the Num Lock key on or off.
/S Turn the Scroll Lock key on or off.

3.13.66 KEYS

Purpose: Enable, disable, or display the history list.

Format: KEYS [ON | OFF | LIST]

See also: [HISTORY](#)^[25↑].

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 [Command History Minimum Length](#)^[50] configuration option, and by using the [HISTORY](#)^[25] command. (CMD.EXE's KEYS command no longer has an effect, because command line editing is always enabled.)

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:

```
history
history /p
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:

```
keys off
```

You can turn the history back on with the command:

```
keys on
```

If you issue the KEYS command without any parameters, **TCC** will show you the current state of KEYS.

3.13.67 KEYSTACK

Purpose: Feed keystrokes to a program or command automatically.

Format: KEYSTACK [/R filename] [!] [/Wx] ["abc"] [keyname[n]] ...

/Wx	Delay in clock ticks before next insertion into the keystack.
"abc"	Literal characters to be placed in the Keystack.
keyname	Name of a key whose code is to be placed in the Keystack or its ASCII.
n	Number of times to repeat the immediately preceding named key.

[/R\(read file\)](#)^[264] [/W\(ait\)](#)^[264]

Usage:

Operation

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 **TCC** itself), you must start the program or [ACTIVATE](#)^[152] 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](#)^[33] command, then use the KEYSTACK command. If you start the program directly (without using [START](#)^[33]) 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 or not the [Wait for External Apps](#)^[47] configuration option is set. If it is **not** set (the default), you can use KEYSTACK immediately after an application is started. However, if [Wait for External Apps](#)^[47] is set, **TCC** will not execute any other operation until the program has finished, including the KEYSTACK command, and instead of the target program, the keystrokes will be sent to whatever program is running in the active window when KEYSTACK is executed.

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.

Keystroke Interpretation

Characters entered within double quotes (for example, "**abc**") will be sent to the target program as is. The only items allowed outside the quotes are key names, the **!** and [/W](#)^[264] options, and a repeat count.

If **keyname** is a single letter, it is inserted in the keystack buffer as if it had been quoted, without any spaces. For example, you could enter the string **abc** as **a b c**, instead of the quoted string method described above.

If **keyname** is a number, it is interpreted as a virtual key code (0 - 255).

Repetition. To send **keyname** several times, follow it with a space, left bracket **[**, the repetition count, and a right bracket **]**. For example, the command below will send the **Enter** key 4 times:

```
keystack enter [4]
```

The repeat count works only with an individual **keyname**. It cannot be used with quoted strings. You must have a blank space between the **keyname** and the repetition count.

See [Keys and key names](#)^[515] for a complete listing of key names and a description of the key name and numeric key code format.

Limitations

You can store a maximum of 4,095 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.

Note

You may need to experiment with your programs and insert delays (see the [/W](#)^[264] option) to find the window activation and keystroke sequence that works for a particular program.

Example

To start Word and open the last document you worked on, you could use the command:

```
start word & keystack /w54 alt-f "1"
```

This starts **Word**, delays about three seconds (54 clock ticks at about 1/18 second each) for **Word** to get started, places the keystrokes for **alt-F** (**File** pulldown menu), and 1 (open the most recently used file) into the buffer. **Word** receives these keystrokes and performs the appropriate actions. Notice that the two commands, [START](#)^[33] and **KEYSTACK** are issued on a single command line. This ensures that the keystrokes are sent to **Word**'s window, not back to **TCC**.

Option:

- /R** Read the KEYSTACK input from a file. (You can only read a single line.)
- /W** 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 **65535** (65,535 ticks is about 1 hour). Do not use the Thousands Separator in the number! You can use the **/W** option as many times as desired and at any point in the string of keystrokes except within double quotes. Some programs may need the delays provided by **/W** in order to receive keystrokes properly from KEYSTACK. The only way to determine what delay is needed is to experiment.

3.13.68 LIST

Purpose Display a text file, with forward and backward paging and scrolling.

Format LIST [range...] [/A:[-|+][rhsadecijopt](#) /B[-]n /C /Etext"/H /I /L[-]n /N /R /S /T"text" /U /W /X[s]]
[@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](#)^[90] for details).

range A file selection [range](#)^[80] ([date](#)^[82], [description](#)^[86], [exclusion](#)^[85], [size](#)^[82], [time](#)^[84])

[/A: \(Attribute select\)](#)^[267]

[/B\(yte offset\)](#)^[267]

[/C \(separate console\)](#)^[267]

[/E \(regular expression\)](#)^[268]

[/H\(igh bit off\)](#)^[268]

[/I\(gnore wildcards\)](#)^[268]

[/L\(ine offset\)](#)^[268]

[/N \(line numbers\)](#)^[268]

[/R\(everse\)](#)^[268]

[/S\(tandard input\)](#)^[268]

[/T \(search for Text\)](#)^[268]

[/U \(Ruler\)](#)^[269]

[/W\(rap\)](#)^[269]

[/X \(heXadecimal display mode\)](#)^[269]

See also: [HEAD](#)^[249], [TAIL](#)^[339], and [TYPE](#)^[354].

File Selection

Supports [attribute switches](#)^[86], extended [wildcards](#)^[77], [ranges](#)^[80], [multiple file names](#)^[87], and [include lists](#)^[88].

Internet

Can be used with [FTP/HTTP Servers](#)^[93].

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*:

list memo.doc

Note: LIST is primarily intended for displaying the contents of ASCII and Unicode 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 hexadecimal 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 **TCC** window. In **TCC**, the standard tool and scroll bars are replaced with the LIST tool and scroll bars. Use the scroll bars or the cursor pad to scroll through the file. You can select the LIST command either with the mouse (on the tool bar and scrollbars) or from the keyboard. LIST recognizes the following keys and buttons:

Key (Button)	Meaning
Home	Display the first page of the file
End	Display the last page of the file
Esc (ListExit ^[37])	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
F5 (ListRefresh ^[39])	Refresh the display
B (ListPrevious ^[39])	Go back to the previous file in the current group of files
Ctrl-B (ListClipboard ^[37])	Copy the current filename to the clipboard
C (ListContinue ^[37])	Continue with the next file
e	Edit the file (with the default editor; see the Editor ^[49] configuration option). If LIST is displaying a pipe, the contents are saved to the clipboard and the editor is started. (You will need to manually paste the clipboard contents.)
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 ^[268]) option
I (Info)	Display information on the current file (the full name, size, date, and time)
L (line numbers)	Toggle the line numbering option
N (ListNext ^[38])	Find next matching string
Ctrl-N	Find previous matching string in the file
O (ListOpen ^[39])	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)
R ListFindRegex ^[38]	Prompt and search for a regular expression
Ctrl-R	Prompt and search backwards for a regular expression
U (ListUnicode ^[39])	Toggle the Unicode display mode
W (Wrap)	Toggle the "line wrap" (/W ^[269]) option

X (Hex)Toggle the hex-mode display ([/X](#)^[269]) 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](#)^[511] 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](#)^[77] 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](#)^[268] 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](#)^[268] to ignore wildcards in the initial search string, and [/R](#)^[268] 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 also search using Regular Expressions using the **R** and **Ctrl-R** keys. See [Regular Expression Syntax](#)^[496] for supported expressions.

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**. 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](#)^[269] 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:

```
dir | list
```

Normally LIST will detect input from a [pipe](#)^[101] automatically, but if it does not, use [/S](#)^[268] 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](#)^[98] 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 will bring up a standard Windows print dialog which allows you to print selected text, the current page, or the entire file.

• FTP/HTTP Usage

LIST can display files on [FTP servers](#)^[93] as well as the contents of HTTP/HTTPS URLs. For example:

```
list ftp://jpsoft.com/index
list http://jpsoft.com/notfound.htm
```

You can also use the [IFTP](#)^[255] 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](#)^[93] and [IFTP](#)^[255].

• NTFS File Streams

LIST supports file streams on NTFS drives. You can list an individual stream by specifying the stream name, for example:

```
list streamfile:s1
```

If no stream name is specified the file's primary data is displayed.

See [NTFS File Streams](#)^[496] for additional details.

• Advanced Features

If you specify a directory name instead of a filename as a parameter, LIST will display each of the files in that directory.

If no filename is specified (and [stdin](#)^[535] is not redirected), LIST will open the common Windows "open file" dialog.

Most of the LIST keystrokes can be reassigned with [key mapping](#)^[36] directives.

By default, LIST sets tab stops every 8 columns. You can change this behavior with the [Tabs Width](#)^[51] configuration option.

Options

- | | |
|---------------|--|
| /A: | Select only those files that have the specified attribute(s) set. See Attribute Switches ^[86] for information on the attributes which can follow /A:. Do not use /A: with @file lists. See @file lists ^[90] for details. |
| /B[-]n | Start at byte n . If n is preceded by a minus sign -, 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. |
| /C | Display the file in a separate screen buffer and restore the original buffer upon exiting LIST. |

- /E** Search for a [regular expression](#)^[496] in the first **file**. This option is the same as pressing **R**, but it allows you to specify the search text on the command line. The regular expression must be contained in double quotes if it contains spaces, punctuation, or wildcard characters. See also **/T**.
- /H** 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 tool bar.
- /I** Only meaningful when used in conjunction with the **/T**^[268] "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**^[268], not subsequent searches started from within LIST.
- /I"text"** Select files by matching text in their descriptions. See [Description Ranges](#)^[86] for details.
- /L[-]n** Start at line **n**. If **n** is preceded by a minus sign -, 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.
- /N** Display line numbers. You can toggle the line numbers with the **L** key.
- /R** Only meaningful when used in conjunction with the **/T**^[268] "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** 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](#)^[198] you could use either of these commands:

```
dir | list
dir | list /s
```

- /T** 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 double quotes if it contains spaces, punctuation, or wildcard characters. For example, to search for the string **TC** in the file **README.DOC**, you can use this command:

```
list /t"Take Command" readme.doc
```

The search text may include [wildcards and extended wildcards](#)^[77]. For example, to search for the words **Hello** and **John** on the same line in the file **LETTER.DAT**:

```
list /t"Hello*John" letter.dat
```

When you display 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. See also: [/I](#)^[268] and [/R](#)^[268].

- /U** Display a ruler on the second line.
- /W** 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** 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.

You can display spaces rather than periods for non-printable characters by specifying the **/XS** option. You can also toggle between spaces and periods with the **S** key while displaying a file in hex mode.

3.13.69 LOADBTM

Purpose: Switch a batch file to or from BTM mode.

Format: LOADBTM [ON | OFF]

Usage:

TCC recognizes three kinds of [batch files](#)^[130]: **.CMD**, **.BAT**, and **.BTM**. Batch files with a **.BTM** extension will run faster than **.BAT** or **.CMD** files, as they are loaded into memory at startup and do not open and close the batch file for each line (as do **.BAT** and **.CMD** files).

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 parameter, 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.

3.13.70 LOADMEDIA

Purpose: Close the door of a removable media drive(s)

Format: LOADMEDIA drive ...

Usage:

LOADMEDIA will close the drive door (if the device allows it) of removable media, such as CD-ROMs, DVDs, etc.

See also [EJECTMEDIA](#)^[220].

3.13.71 LOG

Purpose: Save a log of commands to a disk file.

Format: LOG [/A /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.

[/A\(All\)](#)^[270]
[/E\(rrors\)](#)^[270]

[/H\(istory log\)](#)^[270]
[/W\(rite to\)](#)^[270]

See also: [HISTORY](#)^[251].

Usage:

The LOG command provides independent controls for two different methods of logging **TCC** activity:

- [Command Log](#)^[270]
- [History Log](#)^[270]

Command Log

Command logging creates a record of each internal and external command executed either from the command prompt or from a batch file in the format below:

```
[date time][id] command
```

where the **date** and **time** are formatted according to the country code set for your system, **id** is the process ID in hexadecimal format, and **command** is the actual command after any alias or variable expansion.

The LOG command controls the **command log** operation if it does not include the [/H](#)^[270] option. The default command log filename is **TCCCommandLog**.

History Log

History log creates a record of each command executed from the command prompt exactly as it was entered, before aliases and variables are expanded, without any additional information.

The LOG command controls the **history log** operation if it includes the [/H](#)^[270] option.

Notes

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:

- /A** This option saves all output to the **log all** file. The default filename is **TCLogAll**.
- /E** This option saves all error messages to the **error log**. The default filename is **TCErrLog**. See also: the [Error Logging](#)^[47] configuration option.
- /H** This option saves the commands to the **history log**. The default history log name is **TCHistoryLog**. For example, to turn on history logging and write to the file C:\LOG\HLOG:

```
log /h /w c:\log\hlog
```

/W This switch specifies a different filename for the LOG output. It also automatically performs a LOG ON command. For example, to turn command logging on and write the log to C:\LOG\LOGFILE:

```
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 terminate your **TCC** session. Turning LOG or LOG /H off or on does not change the file name.

3.13.72 MD / MKDIR

Purpose: Create a subdirectory.

Format: MD [/N[et] /S] *path*...
or
MKDIR [/N[et] /S] *path*...

path The name of one or more directories to create.

[/N\(o update\)](#) ^[272]

[/S\(subdirectories\)](#) ^[272]

See also: [RD](#) ^[301].

Internet: Can be used with [FTP Servers](#) ^[93].

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:

```
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:

```
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 [..\].

Windows limits the maximum length of the subdirectory name. See [Directories and Subdirectories](#) ^[492] 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](#) ^[73] database unless the **/N** option is specified.

You can create directories on FTP servers. For example:

```
md ftp://ftp.abc.com/data/index
```

Options:

- /N** If **/N** has no additional options, do not update the CD / CDD [extended directory search](#)^[73] database, *JPSTREE.IDX*. This is useful when creating a temporary directory which you do not want to appear in the extended search database. **/N** takes two optional arguments:
- e** Don't display errors. (Note that a **/Ne** alone will still update the [extended directory search](#)^[73] database.)
 - t** Don't update the [extended directory search](#)^[73] database. (This is the same as **/N** with no options.)
- /S** 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):

```
md /s \one\two\three
```

For compatibility with CMD.EXE, **/S** becomes the default if you enable **TCC** extensions with the **/X** switch on the **TCC** startup command line. See [Command Line Options](#)^[19] for details on **/X**.

3.13.73 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 and function 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.

3.13.74 MKLINK

Purpose: Create NTFS symbolic, hard, and soft links.

Format: MKLINK [/D /H /J /Q] Link Target

[/D](#)^[274] Create a directory symbolic link. (The default is to create a file symbolic link.)

[/H](#)^[273] Create a hard link (like MKLNK).

[/J](#)^[273] Create a junction.

[/Q](#)^[273] Don't display results.

Link The new symbolic link name

Target The pathname (full or relative) that the new link refers to.

Usage:

Due to Windows file system restrictions, creating symbolic links with MKLINK requires an NTFS volume and Windows Vista or later.

The file/directory names in **Link** and **Target** can be fully or partially qualified. MKLINK will also copy an existing description to the link.

See also [MKLNK](#)^[273].

Option:

- /D** Create a directory symbolic link. (The default is to create a file symbolic link.)
- /H** Create a hard link instead of a symbolic link.
- /J** Create a junction rather than a symbolic link.
- /Q** Don't display the result.

3.13.75 MKLNK

Purpose: Create or delete an NTFS hard or soft link.

Format: Create or update a link:
MKLNK [/A:[-]rhsadecijopt] parm1 [parm2]

Delete a link
MKLNK /D parm1

parm1 Name of an existing file ([hard link](#)^[529]) or directory (for [soft link](#)^[534]).
parm2 Name of the new directory entry (a file or directory reference) to be created.

[/A:](#)^[274] (Attribute select)
[/D](#)^[274] Delete a link

See also [MKLINK](#)^[272].

File Selection

For hard links, MKLNK supports [attribute switches](#)^[86], extended [wildcards](#)^[77], [ranges](#)^[80], [multiple file names](#)^[87], and [include lists](#)^[88]. 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](#)^[89] for details.

Usage:

Due to operating and file system restrictions, this command requires an NTFS volume.

The file/directory names in **parm1** and **parm2** can be fully or partially qualified, and may contain wildcards (hard links only). MKLINK will also copy an existing description to the link.

If a single argument is specified and it is a junction, MKLNK will display the directory name linked to the junction.

Hard Links

If **parm1** is a file, and **parm2** does not exist, MKLNK will create a [hard link](#)^[529], as described in the Glossary. If **parm2** exists, MKLNK reports an error.

MKLNK (and the underlying Windows API) may fail if the current directory is on a **subst** or **net use**

drive, or a **UNC** volume.

Soft Links

If **parm1** is a directory, and **parm2** does not exist, MKLNK will create a [soft link](#)^[534], also known as a "directory junction" or "reparse point". If **parm2** exists, and it is a [soft link](#)^[534], MKLNK updates it.

A soft link is an indirect or symbolic reference (**parm2**) to a directory that physically resides in another location (**parm1**). Note: deleting files from a soft link is equivalent to deleting the files from the original directory.

Note: Other operating systems, such as Linux, may also support "hard links" and "soft links", but the Windows implementation of these concepts may not behave in the same manner even though the names might be similar.

Option:

- /A:** Select only those files that have the specified attribute(s) set (hard links only). See [Attribute Switches](#)^[86] for information on the attributes which can follow **/A:**.
- /D** Remove an existing hard or soft link. For hard links, if no more links remain **/D** will not delete the file.

3.13.76 MOVE

Purpose: Move files to a new directory and drive.

Format: MOVE [/A:[-]rhsadecijopt /B /C /D /E /G /H /I"text" /J /L /LD /M /MD /N[de]st] /O /P /Q /R /S[n] /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](#)^[90] for details).

/A: (Attribute select) ^[277]	/N (Disable) ^[278]
/B (Move after reboot) ^[277]	/O (don't move if target exists) ^[278]
/C(hanged) ^[277]	/P(rompt) ^[278]
/D(irectory) ^[277]	/Q(quiet) ^[278]
/E (No error messages) ^[277]	/R(eplace) ^[278]
/G (display percent copied) ^[277]	/S(ubdirectory tree) ^[278]
/H(idden and system) ^[277]	/T(otal) ^[279]
/I"text" (match description) ^[278]	/U(pdate) ^[279]
/J (copy in restartable mode) ^[278]	/V(erify) ^[279]
/L (ASCII FTP transfer) ^[278]	/W(ipe) ^[279]
/LD (create link) ^[278]	/Y (force move of encrypted files) ^[279]
/M(odified files) ^[278]	/Z (overwrite) ^[279]
/MD (Create target directory) ^[278]	

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](#)^[305], but when the destination exists or two volumes are involved, it becomes a two-step command: a [COPY](#)^[182] to the target followed, if successful, by a [DEL](#)^[190] 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](#)^[182], [DEL](#)^[190] and [RENAME](#)^[305].

File Selection

Supports [attribute switches](#)^[86], extended [wildcards](#)^[77], [ranges](#)^[80], [multiple file names](#)^[87], [delayed variable expansion](#)^[89], and [include lists](#)^[88]. 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](#)^[89] for details.

Internet: Can be used with [FTP/TFTP/HTTP/HTTPS Servers](#)^[93].

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](#)^[182] and [RENAME](#)^[305], 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:

```
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](#)^[73] for details.

Be careful when you use MOVE with the [SELECT](#)^[312] 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).

When you specify a single subdirectory source and a single subdirectory target, the source directory tree will be moved to a subdirectory of the target directory. If the source is a subdirectory and the target doesn't exist, the target subdirectory will be created and the source tree moved to it. (These are both for compatibility with **CMD.EXE**.)

- **FTP Usage:**

You can move files to and from Internet URLs (FTP, TFTP and HTTP). For example:

```
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.

- **NTFS File Streams:**

MOVE supports file streams on NTFS drives. You can move an individual stream by specifying the stream name, for example:

```
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](#)^[496] 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 executing.

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.EXE-style "extension" matching (only one extension, at the end of the word) and the advanced **TCC string** matching (allowing things like *.*.abc) when an asterisk is encountered in the **destination** of a MOVE command.

MOVE supports connected web folders. If an HTML file (i.e., with an .htm or .html extension) is copied, MOVE will look for a folder in the same directory with the same name and an extension of ".files". If it is found, the .files directory will be moved to the target directory. You can disable connected web folders by setting the registry key:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\NoFileFolderConnection=0
```

Options:

- /A:** Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)⁸⁶ for information on the attributes which can follow **/A:**. See the cautionary note under **Advanced Features and Options** above before using **/A:** when both the **source** and **destination** directories contain file descriptions. Do not use **/A:** with @file lists. See [@file lists](#)⁹⁰ for details.
- /B** If MOVE can't move the file (i.e., access denied), it will schedule it to be moved at the next reboot.
- /C** 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](#)⁹⁰ for details.
- /D** 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** 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, the transfer rate (in Kbytes/second), and the estimated time remaining. This is useful when copying large files across networks or via FTP to show whether the move is proceeding.
- /H** Move all files, including hidden and system files. See the cautionary note under **Advanced Features and Options** above before using **/H** when both **source** and **destination** directories contain file descriptions.

- /I"text"** Select **source** files by matching text in their descriptions. The text can include wildcards. The search text must be enclosed in double quotes, 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](#)^[90] 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.
- /LD** When used with **/S**, if the source is a symbolic or hard link to a directory, MOVE will create the link in the target directory instead of moving the subdirectory tree.
- /M** Move only files that have the archive bit set. The archive bit will remain set after the MOVE. Do not use **/M** with @file lists. See [@file lists](#)^[90] for details.
- /MD** Create the target directory if it doesn't exist. (Note that you **must** either terminate the target directory name with a trailing \ or specify a filename component; otherwise MOVE cannot tell what you want for the directory and what you want for the filename!)
- /N** 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**.
- A **/N** with one of the following arguments has an alternate meaning:
- d** Skip hidden directories (when used with **/S**)
 - e** Don't display errors.
 - j** Skip junctions (when used with **/S**)
 - s** Don't display the summary.
 - t** Don't update the CD / CDD [extended directory search](#)^[73] database (**JPSTREE.IDX**).
- /O** Don't move the file(s) unless the target doesn't exist, i.e. do not overwrite an existing target..
- /P** Prompt the user to confirm each move. Your options at the prompt are explained in detail under [Prompts](#)^[102].
- /Q** 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** Prompt for a **Y** or **N** response before overwriting an existing **destination** file.
- /S** 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 **JPSTREE.IDX** 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](#)^[90] for details.

If you specify a number after the /S, MOVE will limit the subdirectory recursion to that number. For example, if you have a directory tree "a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.

- /T** 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** 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](#)^[90] 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 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** 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 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.

3.13.77 MSGBOX

Purpose: Display a Windows message box

Format: MSGBOX [/1["text"] /2["text"] /3["text"] /4["text"] /Dn /H /I /M /N /O /Px,y /Q /R /S /Tn /W] buttontype ["title"] prompt

buttontype One of **OK**, **OKCANCEL**, **YESNO**, **YESNOCANCEL**, **RETRYCANCEL**, **ABORTRETRYIGNORE**, **CANCELTRYCONTINUE**, or **CONTINUEABORT**

title Text for the title bar of the message box.

prompt Text that will appear inside the message box.

[/1 \(st button\)](#)^[28]

[/2 \(nd button\)](#)^[28]

[/3 \(rd button\)](#)^[28]

[/4 \(th button\)](#)^[28]

[/D\(isable temporarily\)](#)^[28]

[/H\(elp button\)](#)^[28]

[/I\(nformation icon\)](#)^[28]

[/M \(system modal\)](#)^[28]

[/N \(no sound\)](#)^[28]

[/O \(topmost window\)](#)^[28]

[/P \(screen coordinates\)](#)^[28]

[/Q\(uestion icon\)](#)^[28]

[/R\(ight justify buttons\)](#)^[28]

[/S\(top icon\)](#)^[28]

[/T\(imeout\)](#)^[28]

[/W\(arning icon\)](#)^[28]

See also: [INKEY](#)^[25], [INPUT](#)^[259], [QUERYBOX](#)^[300], and [TASKDIALOG](#)^[34].

Usage:

MSGBOX can display one of eight kinds of message boxes and wait for the user's response. You can

use **title** and **prompt** to display any text you wish. **TCC** will automatically size and center the box on the screen. The message box has up to three response buttons (plus an optional Help button), depending on its type, as shown below.

boxtype	button 1	button 2	button 3
OK	OK		
OKCANCEL	OK	Cancel	
YESNO	Yes	No	
YESNOCANCEL	Yes	No	Cancel
RETRYCANCEL	Retry	Cancel	
ABORTRETRYIGNORE	Abort	Retry	Ignore
CANCELTRYCONTINUE	Cancel	Try Again	Continue
CONTINUEABORT	Continue	Abort	

If the standard message box types don't meet your needs, you can create a custom message box with up to four buttons (plus an optional Help button), specifying the text that appears on each button.

The button the user chooses is indicated using the internal variable `%_?`^[380]. Be sure to save the return value in another variable or test it immediately; because the value of `%_?`^[380] changes with every internal command. The following list shows the value returned for each selection:

response	<code>%_?</code>^[380]
Yes or OK	10
No	11
Cancel	12
Retry	13
Try Again	14
Continue	15
Ignore	16
Abort	17
Help	18
timeout	20
custom button 1	21
custom button 2	22
custom button 3	23
custom button 4	24

If you define custom buttons, the button type argument will be ignored.

If there is an error in the MSGBOX command itself, `%_?`^[380] will be set as described in its documentation (see `_%?`^[380]).

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:
```

Since MSGBOX doesn't write to standard output, it disables redirection and piping to allow you to enter the redirection characters (<, >, and |) in your prompt text.

MSGBOX creates a popup dialog box. If you prefer to retrieve input from the command line, see the [INKEY](#)^[257] and [INPUT](#)^[259] commands.

Options:

- /1** If there is a text string following the option, set the custom text for the first button. Otherwise, set the first button as the default.
- /2** If there is a text string following the option, set the custom text for the second button. Otherwise, set the second button as the default.
- /3** If there is a text string following the option, set the custom text for the third button. Otherwise, set the third button as the default.
- /4** If there is a text string following the option, set the custom text for the fourth button. Otherwise, set the fourth button as the default.
- /Dn** Disable the message box buttons for **n** seconds at startup.
- /H** Display a help button.
- /I** Display an icon consisting of a lower case "i" in a circle in the message box.
- /M** The message box is created as a system modal window.
- /N** Don't play the default sound.
- /O** The message box is created as a topmost window.
- /Px,y** The initial x,y screen coordinates. If you don't use this option, MSGBOX will center its window in the **TCC** tab window.
- /Q** Display a question mark icon in the message box.
- /R** The buttons will be right-justified (as in XP Explorer).
- /S** Display a stop sign icon in the message box.
- /Tn** MSGBOX will wait a maximum of **n** seconds for a response. If the time limit expires, %_? will be set to 20. The time remaining before the window closes will be displayed in the default button.
- /W** Display an exclamation point icon in the message box.

3.13.78 NETMONITOR

Purpose: Monitor network connection and disconnection

Format: NETMONITOR [/C [name]]
 NETMONITOR name CONNECTED | DISCONNECTED n command

name Network name
n Number of repetitions (or **FOREVER**)
command Command to execute when condition is triggered

[/C\(lear\)](#)  ²⁸²

Usage:

The network name can be either **LAN** (for a local area network), **WAN** (dialup network), or the name of a wireless network. The network name can include wildcards.

The command line will be parsed and expanded before NETMONITOR is executed, so if you want to pass redirection characters or variables to **command** you will need to protect them (by enclosing in single back quotes, doubling the %'s, or using command grouping).

If you don't enter any arguments, NETMONITOR will display the networks it is currently monitoring.

The monitoring runs asynchronously in a separate thread. When the condition is triggered, the command will be executed immediately. This may cause problems if you try to write to the display or access files while the main **TCC** thread is also performing I/O. You may need to use [START](#)^[337] or [DETACH](#)^[197] in **command** to avoid conflicts.

NETMONITOR creates environment variables when a network is connected that can be queried by **command**. The variable is deleted after **command** is executed.

_netname	The name (SSID) of the network
_netcount	The number of times the condition has been triggered

Options:

/C	If name is specified, remove the monitor for that network. Otherwise, remove all network monitors.
-----------	---

3.13.79 ON

Purpose: Execute a command in a batch file when a specific condition occurs.

Format:

```
ON BREAK [command]
ON CLOSE [command]
ON ERROR [command]
ON ERRORLEVEL n [command]
ON ERRORMSG [command]
ON LOGOFF [command]
ON SHUTDOWN [command]
ON LBUTTON [command]
ON MBUTTON [command]
ON RBUTTON [command]
```

command command to execute when the event occurs

Usage:

ON sets a watch that remains in effect for the duration of the current batch file, or until replaced by another ON command. Whenever a **break** or **error** condition occurs after ON has been executed, the corresponding **command** is automatically executed.

Activation of ON BREAK

ON BREAK will execute **command** if the user presses **Ctrl-C** or **Ctrl-Break**.

Activation of ON CLOSE

ON CLOSE will execute **command** when **TCC** tab is closed.

Activation of ON ERROR and ON ERRORMSG

ON ERROR or ON ERRORMSG will execute **command** after any critical error, operating system error (such as a disk write error) or internal command error (such as a [COPY](#)^[182] command that fails to copy any files, or the use of an invalid command option).

ON ERROR executes **command** immediately after the error occurs, without displaying any **TCC** error message (Windows errors may still be displayed).

ON ERRORMSG first displays the appropriate error message, then executes **command**.

If both are specified, ON ERROR will take precedence, and ON ERRORMSG will be ignored.

Activation of ON ERRORLEVEL

ON ERRORLEVEL *n* will execute **command** when the internal ERRORLEVEL variable is greater than or equal to the integer specified by *n*. You can also use the IF ERRORLEVEL tests; for example:

```
ON ERRORLEVEL EQ 37 ...
```

Activation of ON LOGOFF

ON LOGOFF will execute **command** when the user logs off.

Activation of ON SHUTDOWN

ON SHUTDOWN will execute **command** when the system is being shut down.

Activation of ON LBUTTON

ON LBUTTON will execute **command** when the left mouse button is clicked.

Activation of ON MBUTTON

ON MBUTTON will execute **command** when the middle mouse button is clicked.

Activation of ON RBUTTON

ON RBUTTON will execute **command** when the right mouse button is clicked.

Scope

Each time an ON statement is defined, it defines a new command to be executed for that event, and any prior command is discarded.

ON BREAK or ON ERROR[MSG] without a command restores the **TCC** default handler.

An ON statement only affects the current batch file. When the batch file containing ON is exited for any reason, whether temporarily (e.g., by a [CALL](#)^[175] to another batch file) or permanently, the **TCC** default **break** and **error** handlers become effective. A [CALL](#)^[175]ed batch file may then use ON to define its own handlers. When control returns to the calling batch file, its **break** and **error** handlers that had been in effect at the [CALL](#)^[175] are reactivated.

Operation

The command can be any command that can be used on a batch file line by itself. Frequently, it is a [GOTO](#)^[248] or [GOSUB](#)^[247] 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](#)^[121] as the command if you want to execute multiple commands, for example:

```
on break (echo Oops, got a break! & quit)
```

ON BREAK, ON ERROR, ON ERRORLEVEL, ON ERRORMSG, ON LBUTTON, ON MBUTTON, and ON RBUTTON assume that you want to continue executing the batch file. After the command is executed, control automatically returns to the command in the batch file immediately after the one that was interrupted by the event. To avoid continuing the batch file after the event at the next command perform one of the following in **command**:

- transfer control with [GOTO](#)^[248],
- end the batch file with [QUIT](#)^[301] or [CANCEL](#)^[176]
- chain to another batch file (without using [CALL](#)^[175]).

When handling an error condition with ON ERROR[MSG], you may find it useful to use [internal variables](#)^[372], particularly [%_?](#)^[380] and [%_SYSERR](#)^[392], to help determine the cause of the error.

To force **TCC** to ignore break or error, use the [REM](#)^[304] command as your command.

Limitations

ON can only be used in batch files.

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, ON ERROR, ON ERRORLEVEL, or ON ERRORMSG is executing, the command will be restarted. This means you must use caution either to avoid or to handle any possible errors in the commands invoked by ON, since such errors can cause an infinite loop.

3.13.80 OPTION

Purpose: Modify or display **TCC** configuration.

Formats: [Invoking the OPTION dialog:](#)^[285]
OPTION

[Temporarily changing a few options:](#)^[285]
OPTION //directive=value ...

[Temporarily changing a list of options:](#)^[285]
OPTION @filename

[Displaying the current value of an option:](#)^[285]

OPTION directive

directive	Name of a directive to set, modify, or display.
value	A new value for that directive.
filename	A file containing directives to be immediately activated.

See also: [.INI file](#)^[26], [SETDOS](#)^[323]

Usage:

Invoking the OPTION Dialog

OPTION without parameters displays a property sheet which allows you to modify most of the configuration options stored in the [INI file](#)^[26].

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](#)^[46] for more information.

In some cases, changes you make in the **Startup** section of the OPTION dialogs will only take effect when you restart **TCC**. 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](#)^[181] command.

OPTION handles most standard directives. The [Key Mapping Directives](#)^[28] and [Advanced Directives](#)^[41] cannot be modified with the OPTION dialogs. These settings must be manually edited .INI file.

OPTION does not preserve inline 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.

Setting Individual Options Temporarily

If you follow the OPTION command with one or more sequences of a double slash mark //, each followed by a new **directive=value**, 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:

```
option //BatchEcho=No //InputColors=bri cya on bla
```

Option values may contain white space. However, you cannot enter an option value that contains the // string.

This feature is most useful for testing settings quickly, and in aliases or batch files that depend on certain options being in effect.

Changes made with // are temporary. They will not be saved in the .INI file.

Setting Many Options Temporarily

The command OPTION **@filename** allows you to temporarily modify multiple directive settings. The file specified by **filename** must be in the same format as an [.INI file](#)^[26]. Changes made with **@filename** are temporary. They will not be saved in the .INI file.

Displaying an option value

Specifying an option name alone will display the value of that option; e.g.:

```
option localHistory
localHistory=Yes
```

See also: the [@OPTION](#)^[452] function.

3.13.81 OSD

Purpose: Write floating text to the display

Format: OSD [/C /Font=n /N /POS=top,left /RGB=r,g,b /TIME=n /TOP /BOTTOM /LEFT /RIGHT /HCENTER /VCENTER /V] *text*

/C	Close the current OSD display
/Font=n	The font height (default 18)
/N	Don't wait for timeout before returning to the prompt
/POS=top,left	Screen coordinates for the top left corner of the text (default 10,10)
/RGB=r,g,b	Text color in RGB format (default 0,255,0)
/TIME=n	Time in seconds to display the text (default 10)
/TOP	Position the text at the top of the display
/BOTTOM	Position the text at the bottom of the display
/LEFT	Position the text at the left of the display
/RIGHT	Position the text at the right of the display
/HCENTER	Center the text horizontally
/VCENTER	Center the text vertically
/V	Display the text vertically
<i>text</i>	The text to display

Usage:

OSD displays text on the desktop without a surrounding window, like TV or monitor prompts.

If you want to display multiple lines, insert the LF escape sequence (^N) in your text. For example:

```
osd /pos=40,50 This is text with^Nmultiple lines.
```

If you specify the /V (vertical display) option, you cannot also display multiple lines of text.

You can combine the window positioning options. For example:

```
osd /hcenter /vcenter /n Your text here
```

OSD will strip leading whitespace in ***text***. If you want to display the leading whitespace in your ***text***, you will need to enclose it in single back quotes.

3.13.82 PATH

Purpose: Display or alter the list of directories that **TCC** 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](#)^[222] and [SET](#)^[319] (the PATH command is syntactically equivalent to SET PATH).

Usage:

When **TCC** 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.

TCC first searches the current directory, then 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.) These search procedures conform to the default search sequences used by Windows.

For example, after the following PATH command, **TCC** 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:

```
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](#)^[319], and edited with [ESET](#)^[222].

The [PATHEXT](#)^[369] environment variable, and the related [PathExt](#)^[47] configuration option, can be used to select the extensions to look for when searching the PATH for an executable file.

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. Some applications also use the PATH to search for their 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 (i.e., 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 **TCC** 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.

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.

3.13.83 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](#) command.

3.13.84 PDIR

Purpose: Display information about files and subdirectories in user-definable fields. It is a "programmable DIR" command.

Format: PDIR [ranges] [/A:[attrlist] /D /H /I"text" /K /M /N[dejj] /O[:][order] /P /S[n] /T:t /(...)] [file...]

attrlist	Selection attributes (see attribute switches for details)
order	Hierarchical list of sort keys
ranges	One or more date, description, exclusion, size, time ranges
file	One or more files to list
t	Timestamp type selection code

/A	Attribute select	/N	Disable options
/D	colorize	/O	Order
/H	do not Hide . and ..	/P	Page pause
/I	description range	/S	Subdirectories
"text"			
/K	show header	/T	Timestamp type
/M	show footer	[:t]	
		/(...)	output fields and format

See also: [DIR](#), [ATTRIB](#), [DESCRIBE](#), and [SELECT](#).

File Selection

Supports [attribute switches](#), extended [wildcards](#), [ranges](#), [multiple file names](#), and [include lists](#).

Internet

Can be used with [FTP/HTTP Servers](#).

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](#) command.

PDIR and [DIR](#)^[198] are related, but 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](#)^[154] command.

The `/(...)` option specifies which fields you want to display and how to format them. (You can have multiple `/(...)` options on a line.) The syntax is:

- a** Attributes
- c** Compression: Display the compression percentage on NTFS drives with compression enabled.
- d[...]** Date (you must specify at least one subfield, otherwise the field remains blank)
 - d** day (2 digits, leading zero)
 - m** month (2 digits, leading zero)
 - y** year (4 digits)
- f[...]** File or Directory name (case sensitive)
 - P** SFN path
 - p** LFN path
 - N** SFN filename
 - n** LFN filename (default)
- i** Description
- m** MD5 hash value (see the [@MD5](#)^[450] function)
- q** File or directory owner (NTFS only)
- r** CRC32 hash value (see the [@CRC32](#)^[413] function)
- s** stream names (NTFS only)
- sp** path and stream names as pathname+filename+streamname (NTFS only)
- t[...]** Time (you must specify at least one subfield, otherwise the field remains blank)
 - h** hours (2 digits, leading zero)
 - m** minutes (2 digits, leading zero)
 - s** seconds (2 digits, leading zero)
 - d** milliseconds (decimal separator and 3 digits)
- 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](#)^[395]). Note that the size will be truncated, not rounded.

@function[*]

call the specified variable [function](#)^[395] (internal or user-defined). To specify the current filename, use * as the parameter. 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 * parameter. If the parameter 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, except that escape characters are converted.

You can also specify a format, independently for each field, by prefixing the field character with its format specification:

```
[ - ]i.a
```

where

- specifies left justification instead of the default, right justification;
- i specifies the minimum field width, and
- a specifies the maximum field width.

If the first digit of *i* is **0**, the field will be padded with zeros instead of spaces.

Example

To display the CRC, the full LFN and the owner of each file:

```
pdir / (r f p n 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](#)^[198] command when a single filename is specified.

Most options are used to select the desired files/directories. (This is in contrast to the [DIR](#)^[198] command.) The special option [/\(...\)](#)^[291] is used to specify which characteristics of the selected files or directories should be displayed in which sequence and format.

- /A:...** Display only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[86] for information on the attributes which can follow **/A:.**
- /D** Colorize the directory listing. See [DIR](#)^[198] for more information on directory colorization.
- /H** Show the "." and ".." directory names (normally suppressed).
- /I "text"** Select filenames by matching text in their descriptions. See [Description Ranges](#)^[86] for details.
- /K** Show the header (disk and directory name) display.
- /M** Show the footer (file and byte count totals) display.
- /N** Turn off the specified options.

- d** Skip hidden directories (when used with /S)
- e** Don't display errors
- j** Skip junctions (when used with /S)

/O... The sorting order is applied to the listings of each subdirectory separately. Any combination of the sorting options may be used. If multiple options are specified, 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 (default). If **e** is also specified, sort by name only.
- 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); 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).
- o** Sort by owner
- r** Reverse the sort order for all options
- s** Sort by size
- u** Unsorted

/P 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](#)^[102].

/S Display file information from the current directory and all of its accessible subdirectories.

If you specify a number after the /S, PDIR will limit the subdirectory recursion to that number. For example, if you have a directory tree "a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.

/T:type 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 (NTFS volumes).
- c** Creation date and time.
- w** Last write date and time (default).

If /T is not specified, the default is /T:w.

If you append a **u** after the field, DIR will display the file time in UTC.

Note: If more than one time type is specified, the first one specified is used, and all subsequent ones ignored.

/(...) 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](#)^[198] command.

3.13.85 PLAYAVI

Purpose: Play Windows .AVI (video clip) files.

Format: PLAYAVI [/A /C /S /Vn] *filename*

filename The file to play

[/A\(synchronous\)](#)^[292] [/S\(ynchronous\)](#)^[292]
[/C\(enter\)](#)^[292] [/V\(olume\)](#)^[292]

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 **TCC** waits for the .AVI file to complete and its window to close before continuing with the next command in a batch file or alias, or prompting you for a new command. You can change this default behavior with the **/A** option.

Options:

- /A** Plays the .AVI file in asynchronous mode. Control returns to the **TCC** prompt immediately for a new command or to execute the next command in the current batch file or alias.
- /C** 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** Plays the .AVI file in synchronous mode (this is the default). **TCC** pauses until the file has finished playing and its window closes.
- /V** Sets the volume level. The range is 0 (silent) to 100.

3.13.86 PLAYSOUND

Purpose: Play MP3, .WAV, Midi, and other sound files.

Format: PLAYSOUND [/A /M /S /U /Vn] *filename*

filename The file to play

[/A\(synchronous\)](#)^[293] [/U\(n mute\)](#)^[293]
[/M\(ute\)](#)^[293] [/V\(volume\)](#)^[293]
[/S\(ynchronous\)](#)^[293]

Usage:

PLAYSOUND "plays" MP3, .WAV, Midi and other types of sound files for which Windows has an appropriate 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 **TCC** 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](#)^[293] switch, described below.

You can cancel the playing of a synchronous sound file by pressing Ctrl-Break while it is playing.

Options:

- /A** Plays the sound file in asynchronous mode. Control returns to the **TCC** prompt immediately for a new command or to execute the next command in the current batch file or alias.
- /M** Mute the volume.
- /S** Plays the sound file in synchronous mode (this is the default). **TCC** pauses until the file has finished playing and its window closes.
- /U** Unmute (restore the previous volume level).
- /V** Sets the volume level. The range is 0 (silent) to 100.

3.13.87 PLUGIN

Purpose: Load, unload, or display current plugins

Format: PLUGIN [/B /I plugin /L plugin /P /U plugin]

[/B \(full pathname\)](#)^[293] [/P\(ause\)](#)^[293]
[/I\(nfo\)](#)^[293] [/U\(nload\)](#)^[293]
[/L\(oad\)](#)^[293]

Usage:

Plugins allow you to write your own internal variables, variable functions, and internal commands, put them in a DLL, and have **TCC** load them at startup. Plugin names will override existing names, so you can extend and/or replace internal variables and commands. When **TCC** starts, it will automatically load any plugins in the default directory (the subdirectory PLUGINS\ in the **TCC** installation directory). The plugins will be loaded before the startup file ([TCSTART](#)^[22]) are executed.

You can also write keystroke plugins that will be called for every keystroke entered at the command line. A keystroke plugin can perform actions when a specific key is entered, or even change the key before passing it back to the command processor.

If no options are specified, PLUGIN will display the currently loaded plugins and their internal variables, variable functions, and commands.

See the Plugin SDK for more information on developing plugins.

Options:

- /B** Display the full pathnames of the plugins.
- /I** Display information about the specified plugin, including the name, author, author's email and web addresses, description, function list, version and build numbers. The **/I** option supports wildcards.
- /L** Loads the specified plugin. If the filename is *, load all plugins from the default directory (the subdirectory PLUGINS\ in the **TCC** installation directory).
- /P** 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](#)^[102].
- /U** Unloads the specified plugin. If the filename is *, unloads all plugins.

3.13.88 POPD

Purpose: Return to the disk drive and directory at the top of the directory stack..

Format: POPD [/X * n]

n The number of directories to pop

[/X \(exclude\)](#)^[294]

See also: [DIRS](#)^[209], [PUSHD](#)^[299], [@DIRSTACK](#)^[416] and [Directory Navigation](#)^[71].

Usage:

Each time you use the [PUSHD](#)^[299] command, it saves the current disk drive and directory on the internal directory stack. POPD restores the most recently saved drive and directory 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](#)^[118]. Read the section on [Directory Navigation](#)^[71] for complete details on this and other directory navigation features.

This example saves and changes the current disk drive and directory with [PUSHD](#)^[299], 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](#)^[209] 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.

Options:

/X Don't save the current directory to the Directory History list.

3.13.89 POSTMSG

Purpose: Post a message to a window

Format: POSTMSG "title" msg wparam lparam

title	The window title
msg	The message to send
wParam	wParam integer
lParam	lParam integer value

Usage:

POSTMSG allows you to send a Windows message to any window with a caption.

The **title** may contain wildcards, and POSTMSG will send the message to the first window with a matching title.

See the Windows SDK documentation for a list of possible messages and their parameters.

3.13.90 PRINT

Purpose: Print the specified file(s) using the application associated with each file's extension.

Format: PRINT [/A printer /D printer] filename ...

[/A\(dd\) printer](#)^[295] [/D\(elete\) printer](#)^[295]

Usage:

Except for plain text files, Windows files cannot be printed without sending them to an associated application for interpretation and formatting. Using the extension for each file you want to print, PRINT determines 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

```
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 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. 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](#)^[506] 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 expected.

Options:

/A Add a connection for the specified printer.

/D Remove the connection to the specified printer.

3.13.91 PRIORITY

Purpose: Display or set process priority, or suspend or resume a process.

Format: PRIORITY [/Q /R /S PID | "title" ABOVE | BELOW | NORMAL | HIGH | IDLE | REALTIME]

ABOVE	Above normal priority
BELOW	Below normal priority
NORMAL	Normal (default) priority
HIGH	High priority
IDLE	Idle priority (only executes when no higher priority task is scheduled)
REALTIME	Realtime priority

[/Q\(quiet\)](#)^[296] [/S\(suspend\)](#)^[296]
[/R\(esume\)](#)^[296]

Usage:

You can specify the process either by the PID or by the window title. If you don't specify either a PID or title, PRIORITY will adjust the priority of the current **TCC** process.

If you do not enter any arguments, PRIORITY displays all of the active processes, their current priority, the module names, and the window titles (if any).

Options:

/Q Don't display any suspend / resume messages.
/R Resume the process
/S Suspend the process

3.13.92 PROCESSMONITOR

Purpose: Monitor process start or end

Format: PROCESSMONITOR [/C [name]]
PROCESSMONITOR name STARTED | ENDED n command

name Full pathname of the process to monitor
n Number of repetitions (or **FOREVER**)
command Command to execute when condition is triggered

[/C\(lear\)](#)^[297]

Usage:

The process name can include wildcards.

The command line will be parsed and expanded before PROCESSMONITOR is executed, so if you want to pass redirection characters or variables to **command** you will need to protect them (by enclosing in single back quotes, doubling the %'s, or using command grouping).

If you don't enter any arguments, PROCESSMONITOR will display the processes it is currently monitoring.

The monitoring runs asynchronously in a separate thread. When the condition is triggered, the command will be executed immediately. This may cause problems if you try to write to the display or access files while the main **TCC** thread is also performing I/O. You may need to use [START](#)^[331] or [DETACH](#)^[197] in **command** to avoid conflicts.

PROCESSMONITOR creates three environment variables when a process is STARTED that can be queried by **command**. The variables are deleted after **command** is executed.

_processname The name of the process that was started
_processpid The PID of the process

_processcount The number of times the command has been triggered

For example, if you want to be alerted whenever "myapp" exits:

```
processmonitor myapp ended forever sendmail bob@abc.com Myapp Myapp just shut down!
```

Options:

/C If **name** is specified, remove the monitor for that process name. Otherwise, remove all active process monitors.

3.13.93 PROMPT

Purpose: Change the command line prompt.

Format: PROMPT [text]

text Text to be used as the new command line prompt.

See also: [ESET](#)^[222] and [SET](#)^[319] (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 a complex 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: **Fri 12-28-07** (the month, day, and year are formatted according to your current country settings)
- D** Current date, in the format: **Fri Dec 28, 2007**
- e** The ASCII ESC character (decimal 27), necessary for [ANSI](#)^[101] 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** Windows version number, in the format **6.0**.

- 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.
- +** Display one + character for each directory on the [PUSHD](#)^[299] 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 **TCC** prompt can be set in [TCSTART](#)^[22] or in any batch file that runs when **TCC** starts.

If you enter PROMPT with no parameters, 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](#)^[365] variable, [internal variable](#)^[372], or [variable function](#)^[395] 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:

```
prompt `[(@dosmem[K]K) $p]`
```

You can use this feature along with the [@EXEC](#)^[424] 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):

```
alias cbatt `if %_apmlife lt 30 beep 440 4 880 4 440 4 880 4`
prompt `%@exec[@cbatt]$p$g`
```

You can include [ANSI](#)^[101] escape sequences in the PROMPT by using the built-in ANSI X3.64 support in **TCC**. 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):

```
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 [TCSTART](#)^[22] file and use IF, IFF, or SWITCH statements to set the appropriate prompt for different shells.

3.13.94 PUSHD

Purpose: Save the current disk drive and directory, optionally changing to a new drive and directory.

Format: PUSHD [/X path]

path The name of the new default drive and directory.

[/X \(exclude\)](#)^[299]

See also: [DIRS](#)^[209], [POPD](#)^[294], [@DIRSTACK](#)^[416] and [Directory Navigation](#)^[71].

Usage:

PUSHD saves the current drive and directory to a "last in, first out" directory stack. The [POPD](#)^[294] 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](#)^[209] command displays the contents of the directory stack.

To save the current drive and directory, without changing directories, use the PUSHD command by itself, with no **path**.

If a **path** is specified as part of the PUSHD command, the current drive and directory are saved and PUSHD changes to the specified drive and directory. If the **path** includes a drive letter, PUSHD changes to the specified directory on the new drive without changing the current directory on the original drive.

This example saves the current directory and changes to C:\WORDP\MEMOS, then returns to the original directory:

```
[c:\] pushd \wordp\memos
[c:\wordp\memos] popd
[c:\]
```

When you use PUSHD to change to a directory on an LFN drive, you must quote the **path** name if it contains white space or special characters.

PUSHD can also change to a network drive and directory specified with a UNC name (see [File Systems](#)^[491] for details).

If PUSHD cannot change to the directory you have specified it will attempt to search the [CDPATH](#)^[72] and the [extended directory search](#)^[73] database. You can also use [wildcards](#)^[77] in the **path** to force an extended directory search. Read the section on [Directory Navigation](#)^[71] 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](#)^[118].

The directory stack can hold up to 2047 characters, or about 100 typical entries (depending on the length of the names). If you exceed this limit, the oldest entry is removed before adding a new entry.

Options:

/X Don't save the current directory to the Directory History list.

3.13.95 QUERYBOX

Purpose: Pops up a dialog box to get an input string from the user and save it in an environment variable.

Format: QUERYBOX [/D /E /Ln /P /POS=top,left /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) ^[300]	/P(assword) ^[300]
/E(dit existing value) ^[300]	/POS (ition) ^[300]
/L (maximum Length) ^[300]	/T(imeout) ^[300]

See also: [INKEY](#)^[257], [INPUT](#)^[259], and [MSGBOX](#)^[279].

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 text 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](#)^[282].

QUERYBOX returns a value of zero in the internal variable [%_?](#)^[380] 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	Only accepts numeric values.
/E	Allows you to edit an existing value. If there is no existing value for varname , QUERYBOX allows you to enter a new value.
/Ln	Sets the maximum number of characters which QUERYBOX will accept to n .
/P	Tells QUERYBOX to echo asterisks, instead of the characters you type.
/POS	Sets the dialog position. (If you don't specify a position, QUERYBOX will center the dialog in the TCC window.
/Tn	Wait for a maximum of n seconds for a response.

3.13.96 QUIT

Purpose: Terminate the current batch file.

Format: QUIT [value]

value The numeric exit code to return to **TCC** or to the previous batch file.

See also: [CANCEL](#)^[176] and [EXIT](#)^[227].

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](#)^[176] command.

If you specify a *value*, QUIT will set the [ERRORLEVEL](#)^[395] or exit code to that value. For information on exit codes see the [IF](#)^[253] command, and the [%?](#)^[380] 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.

3.13.97 RD / RMDIR

Purpose: Remove one or more subdirectories.

Format: RD [/I"text" /K /N[et] /Q /R /S] [@file] path...
or
RMDIR [/I"text" /K /N[et] /Q /R /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](#)^[90] for details).

/I (match descriptions) ^[302]	/Q(quiet) ^[302]
/K (no Recycle Bin) ^[302]	/R(ecycle bin) ^[302]
/N (disable options) ^[302]	/S(ubdirectories) ^[302]

See also: [MD](#)^[271].

File Selection

Supports extended [wildcards](#)^[77], [ranges](#)^[80], [multiple file names](#)^[87], and [include lists](#)^[88]. Use wildcards with caution on LFN volumes; see [LFN File Searches](#)^[89] for details.

Internet: Can be used with [FTP Servers](#)^[93].

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:

```
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](#) ^[190] 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](#) ^[73] 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](#) ^[93]. For example:

```
rd ftp://ftp.abc.com/data
```

Options:

- /I"text"** Select directories by matching text in their descriptions. The text can include [wildcards](#) ^[77] and extended wildcards. The search text must be enclosed in double quotes, 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](#) ^[90] for details.
- /K** When used with the **/S** option, this will physically delete files instead of sending them to the Windows Recycle Bin, even if you have the [Delete to Recycle Bin](#) ^[47] configuration option set.
- /N** This option takes two possible arguments:
- e** Don't display errors.
 - t** Don't update the CD / CDD [extended directory search](#) ^[73] database (**JPSTREE.IDX**).
- /Q** When used with the **/S** option, this will suppress the prompt before deleting the directories.
- /R** When used with the **/S** option, this will send the deleted files to the Windows Recycle Bin.
- /S** This option is included only for compatibility with CMD.EXE, and should be used with **EXTREME CARE!!** 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.** You cannot use wildcards with the **/S** option.

Note: Do not use /S with @file lists.

3.13.98 REBOOT

Purpose: Reboot the computer, log off Windows, or shut down.

Format: REBOOT [/H /K /L /M[0|1] /P /R /S /V /W]

/H(ibernate)	/R(eboot)
/K (lock)	/S(hutdown)
/L(ogoff)	/V(erify)
/M(onitor)	/W (standby)
/P(ower off)	

Usage:

REBOOT will log off or shut down the operating system, or completely restart your computer. It normally performs a warm reboot, or a 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:

```
reboot /v
```

TCC issues the standard commands to shut down other applications and the Windows before rebooting. Windows may prompt you for additional actions, or even ignore the request altogether depending on which processes are running.

Options:

- /H** Save everything in memory to your hard disk, and shutdown to save power. The desktop is restored to its original state when the computer is restarted.
- /K** Lock the workstation. To unlock, the user must log in.
- /L** 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.
- /M** Switch the display to low power (M0) or shut off the display (M1 -- will not work on all systems). This option will not reboot the computer unless you also include /R.
- /P** Log off Windows and turn off the computer.
- /R** Reboots the system. This is the default, but is required if you specify /M0 or /M1 and also want to reboot.
- /S** 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** Prompt for confirmation (**Y** or **N**) before acting.
- /W** Save power by turning off the monitor and hard disks. When the computer comes out of standby, the desktop is restored to its original state.

3.13.99 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\(delete\)](#)^[304]

[/E \(no error messages\)](#)^[304]

[/P\(prompt\)](#)^[304]

[/Q\(quiet\)](#)^[304]

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** Empty the recycle bin for the specified drive(s).
- /E** 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 the user to confirm each delete operation.
- /Q** Don't display the name of the recycle bin(s). This option is most often used in batch files.

3.13.10 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 8,191 characters long. **TCC** 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.

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:\xyz:

```
rem>xyz
```

(This capability is included for compatibility with and CMD.EXE. A simpler method for creating a zero-byte file with **TCC** is to use **>filename** as a command, with no actual command before the [>] redirection character.)

3.13.10 REN / RENAME

Purpose: Rename files or subdirectories.

Format: REN [/A:[-][+]*rhsadecijopt*] /B /E /I"text" /N[et] /P /Q /S /T [*@file*] *old_name...* *new_name*
or
RENAME [/A:[-][+]*rhsadecijopt*] /E /I"text" /N[et] /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](#) ^[90] for details).

/A: (Attribute select) ^[307]	/N (Disable) ^[307]
/B (Rename on reboot) ^[307]	/P(prompt) ^[307]
/E (No error messages) ^[307]	/Q(quiet) ^[307]
/I"text" (match description) ^[307]	/S(subdirectory) ^[307]
/MD (Create target directory) ^[307]	/T(otal) ^[307]

See also: [COPY](#) ^[182] and [MOVE](#) ^[274].

File Selection:

Supports [attribute switches](#) ^[86], extended [wildcards](#) ^[77], [ranges](#) ^[80], [multiple file names](#) ^[87], [delayed variable expansion](#) ^[89], and [include lists](#) ^[88]. Use wildcards with caution on LFN volumes; see [LFN File Searches](#) ^[89] for details.

Internet: Can be used with [FTP/HTTP Servers](#) ^[93] 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 *IWP*:

```
rename memo.txt mem.txt
rename /s \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:

```
ren config.nt autoexec.nt tcstart.btm *.sav
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**:

```
ren memo.txt \wp\memos\
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 parameter 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:

```
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](#)^[73] 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:\TCMD** directory contains a subdirectory **TEST**, you can rename TEST to be a subdirectory of the root directory like this:

```
[d:\tcmd] ren TEST \TEST\
```

REN does not change a file's attributes, except to set attribute **A**. 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. For example:

```
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](#)^[93] and [IFTP](#)^[255].

Note: The wildcard expansion process will attempt to allow both CMD.EXE-style "extension" matching (assumes only one extension, at the end of the word) and the advanced **TCC** string matching (allowing things like *.*.abc) when an asterisk is encountered in the destination of a REN command.

Options:

- /A:** Rename only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[86] for information on the attributes which can follow **/A:**. Do not use **/A:** with [@file lists](#)^[90]. See [@file lists](#)^[90] for details.
- /B** If REN can't rename the file (i.e., access denied), it will schedule it to be renamed at the next reboot.
- /E** Suppress all non-fatal error messages, such as "File Not Found." Fatal error messages, such as "Drive not ready," will still be displayed. This option is most useful in batch files.
- /I "text"** Select files by matching text in their descriptions. The text can include [wildcards](#)^[77] and extended wildcards. The search text must be enclosed in double quotes, 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](#)^[90]. See [@file lists](#)^[90] for details.
- /MD** Create the target directory if it doesn't exist. Note that you **must** either terminate the target directory name with a trailing \ or specify a filename component; otherwise REN cannot tell what you want for the directory and what you want for the filename.
- /N** 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.
- A **/N** with one of the following arguments has an alternate meaning:
- e** Don't display errors.
 - t** Don't update the CD / CDD [extended directory search](#)^[73] database (**JPSTREE.IDX**).
- /P** Prompt the user to confirm each rename operation. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[102].
- /Q** 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** 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 [GLOBAL](#)^[245] REN.
- /T** Don't display filenames as they are renamed, but report the number of files renamed. See also **/Q**.

3.13.10 RETURN

Purpose: Return from a GOSUB (subroutine) in a batch file.

Format: RETURN [value]

value The numeric exit code to return to **TCC**

See also: [GOSUB](#)^[247].

Usage:

TCC 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, **TCC** will display a "Missing GOSUB" error message.

You cannot execute a RETURN from inside a [DO](#) [210] loop.

The following batch file fragment calls a subroutine which displays the files in the current directory:

```
echo Calling a subroutine
gosub subr1
echo Returned from the subroutine
quit
:subr1
dir /a/w
return
```

If you specify a **value**, RETURN will set the internal 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 [?](#) [380] variable.

3.13.10 REXEC

Purpose: Remotely execute commands

Format: REXEC [/H host /U name /P password /Rn /Tn] host [/L userid] command ...

command The command to execute

/H(host name) [309]	/R(remote port) [309]
/L (user ID) [309]	/T (firewall type) [309]
/P(assword) [309]	/U(sername) [309]

Usage:

REXEC allows remote execution of commands on any system with the rexec service installed. Press Ctrl-C to disconnect from the other system.

If you don't specify a username, RSHELL will use the current username. You can provide a password on the command line by appending it to the username (i.e., "User:Password"). If you don't provide a password, REXEC will prompt for it.

If you want to do redirection on the remote system, enclose the argument list in double quotes. For example:

```
REXEC /H host /U user /P password "command | command2"
```

The double quotes will be removed before passing the commands to the remote system.

Note: Windows does not include the rshell service, so you will need to get one from a third-party and install it on the remote system before executing RSHELL.

Options:

- /H** Firewall host name
- /L** User name (ID)
- /P** Firewall user password
- /R** Remote port number
- /T** Firewall type, where *n* is:
 - 0 No firewall (default setting)
 - 1 Connect through a tunneling proxy
 - 2 Connect through a SOCKS4 Proxy
 - 3 Connect through a SOCKS5 Proxy
- /U** Firewall user name

3.13.10 RSHELL

Purpose: Remotely execute commands

Format: RSHELL [/H host /U name /P password /Rn /Tn] host [/L userid] command ...

command The command to execute

[/H\(ost name\)](#)^[309]
[/L \(user ID\)](#)^[309]
[/P\(assword\)](#)^[310]

[/R\(emote port\)](#)^[310]
[/T \(firewall type\)](#)^[310]
[/U\(sername\)](#)^[310]

Usage:

RSHELL allows remote execution of commands on any system with the rshell service installed. Press Ctrl-C to disconnect from the other system.

If you don't specify a username, RSHELL will use the current username.

If you want to do redirection on the remote system, enclose the argument list in double quotes. For example:

```
RSHELL /H host /U user /P password "command | command2"
```

The double quotes will be removed before passing the commands to the remote system.

Note: Windows does not include the rshell service, so you will need to get one from a third-party and install it on the remote system before executing RSHELL.

Options:

- /H** Firewall host name
- /L** User name

/P	Firewall user password								
/R	Remote port number								
/T	Firewall type, where n is: <table> <tr> <td>0</td><td>No firewall (default setting)</td></tr> <tr> <td>1</td><td>Connect through a tunneling proxy</td></tr> <tr> <td>2</td><td>Connect through a SOCKS4 Proxy</td></tr> <tr> <td>3</td><td>Connect through a SOCKS5 Proxy</td></tr> </table>	0	No firewall (default setting)	1	Connect through a tunneling proxy	2	Connect through a SOCKS4 Proxy	3	Connect through a SOCKS5 Proxy
0	No firewall (default setting)								
1	Connect through a tunneling proxy								
2	Connect through a SOCKS4 Proxy								
3	Connect through a SOCKS5 Proxy								
/U	Firewall user name								

3.13.10 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](#)^[217], [ECHOS and ECHOSERR](#)^[219], [SCRPUT](#)^[311], [TEXT](#)^[345], and [VSCRPUT](#)^[360].

Usage:

SCREEN allows you to create attractive screen displays in batch files. SCRPUT allows you 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](#)^[311] or [VSCRPUT](#)^[360]. SCREEN always leaves the cursor at the end of the displayed text.

The **row** and **column** values are zero-based, so on a 25 line by 80 column display, valid **rows** are 0 - 24 and valid **columns** are 0 - 79. SCREEN checks for a valid **row** and **column**, and displays a "Usage" error message if either value is out of range.

In **TCC**, the maximum **row** value is determined by the current height of the **TCC** tab window, and the maximum **column** value is determined by the current virtual screen width (see [Resizing the Take Command Window](#)^[68] 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 **TCC** tab window:

```
screen 999 999 Hello, World
```

3.13.10 SCRIPT

Purpose: Run a script using an Active Scripting engine.

Format: SCRIPT [/E engine] [filename ...]

[/E\(engine\)](#)^[311]

engine The name of the scripting engine

Usage:

If you don't specify any arguments, SCRIPT will display the installed engines.

See also the [@SCRIPT](#)^[458] variable function.

TCC has a COM interface to allow the script to call back into **TCC**. The methods are:

shell.exec("command") - execute the specified command (internal or external)

shell.write("string") - write the string to stdout

shell.writeLn("string") - write the string to stdout with a CR/LF

shell.alert("text") - pop up a message box

Options:

/E If the script doesn't have a recognized extension (i.e., **.vbs**, **.pls**, etc.) you will need to specify the engine SCRIPT should use to execute the script.

3.13.10 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](#)^[217], [ECHOS and ECHOSERR](#)^[219], [SCREEN](#)^[310], [TEXT](#)^[345], and [VSCRPUT](#)^[360].

Usage:

SCRPUT allows you to create attractive screen displays in batch files. SCRPUT allows you 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](#)^[518] 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. The maximum **row** value is determined by the current height of the **TCC** tab window. The maximum **column** value is determined by the current virtual screen width (see [Resizing the Take Command Window](#)^[68] for more information).

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 **TCC** tab 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
```

3.13.10{SELECT

Purpose: Interactively select files for a command.

Format: SELECT [/1 /A[:][-][+]*rhsadecijopt*] /C /D /E /H /I"text" /J /L /O[:][-]*acdeginorsu* /T:acw /X /Z] [*command*] ... (*files...*)...

command
files

The command to execute with the selected 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 ^[315]	/J(ustify names) ^[315]
/A(ttribute select) ^[315]	/L(ower case) ^[315]
/C(ompression) ^[315]	/O(rder) ^[315]
/D(isable color coding) ^[315]	/T(ime) ^[316]
/E (use upper case) ^[315]	/X (display short names) ^[316]
/H(ide dots) ^[315]	/Z (FAT format) ^[316]
/I"text" (match descriptions) ^[315]	

File Selection

Supports extended [wildcards](#)^[77], [ranges](#)^[80], [multiple file names](#)^[87], and [include lists](#)^[88]. Ranges **must** appear immediately after the SELECT keyword.

Internet: Can be used with FTP servers. See [Using FTP/HTTP Servers](#)^[93].

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 a parameter. 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 parameters.

SELECT can also select files on FTP servers. For example:

```
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](#)^[93] and [IFTP](#)^[255].

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](#)^[507]. 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
-	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	View the current highlighted file with LIST ^[264] . 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.

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:

```
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](#)^[88] inside the parentheses by inserting a semicolon:

```
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:

```
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](#)^[126]. 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:

```
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](#)^[80]. 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:

```
alias utils select (c:\util\*.com;*.exe;*.btm;*.bat)
```

If you want to use [filename completion](#)^[113] 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, **TCC** 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[:]** Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[86] for information on the attributes which can follow **/A:**.
- /C** 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](#)^[198] for more details on how compression ratios are calculated.
- /D** Temporarily turn off directory colorization.
- /E** Display filenames in upper case.
- /H** Suppress the display of the "." and ".." directory names.
- /I"text"** Display filenames by matching text in their descriptions. The text can include [wildcards](#)^[77] and extended wildcards. The search text must be enclosed in double quotes, 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 (align) filename extensions and display them in the FAT format.
- /L** Display file and directory names in lower case.
- /O** 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).
- o** Sort by owner
- r** Reverse the sort order for all options.
- s** Sort by size.
- u** Unsorted.

/T:acw 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.

3.13.10!SENDMAIL

Purpose: Send an email message.

Format: SENDMAIL [/A file1 [/A file2 ...] /D /Eaddress /H"header: value" /In /M /Pn /R /Sn /V] "address[,address...] [cc:address[,address] bcc:address[,address...]]" subject [text | @msgfile]

file1...	The attachment files.
address ^[317]	The destination email address.
subject ^[317]	The subject line.
text ^[316]	The message to send.
msgfile ^[317]	The file containing the message body.

/A ^[317] file	Attachment	/M ^[317]	CRAM-MD5 authentication
/D ^[317]	Delivery Confirmation	/P ^[318]	Priority
/E ^[317]	Reply-to address	n	
/H ^[317]	Send custom header	/R ^[318]	Send read receipt
/I ^[317] n	Importance	/Sn	Sensitivity
		V ^[316]	Verbose

See also: [SNPP](#)^[331] and [SMPP](#)^[330].

Usage:

SENDMAIL sends an email message from **TCC** 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 [SMTP](#)^[53] configuration options, 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](#)^[317], a [subject](#)^[317], and [message](#)^[317]. Optionally it may also have [attachments](#)^[317].

1. The **address** field contains one or more standard Internet email addresses:

```
sendmail abc@xyz.com ...
```

If **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 **cc** (**copy**) addresses by prefacing the desired target(s) with **cc:**; and **BCC** (**blind copy**) addresses by prefacing the desired target(s) with **bcc:**. For example:

```
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 it may be placed in a text file. To tell SENDMAIL to send the contents of a file as the message text, use @ sign, followed by the filename. You can use the same approach to send the text content of the clipboard (**@CLIP:**) or the console (**@CON:**):

```
sendmail abc@xyz.com Party @c:\messages\invitation.txt
sendmail abc@xyz.com Party @clip:
type myfile.txt | sendmail abc@xyz.com Party @con:
```

Options:

- /A file** 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 for each additional file to send. For example:

```
sendmail /a file1 /a "d:\path\My file2" abc@xyz.com ...
```

- /D** Request Delivery Notification.

- /E** Set the "reply to" address in the message header.

- /H** Set a custom header. The header will be appended to the message headers created from "to", "from", "subject", etc. The headers must of the format "header: value" as specified in RFC 822. You can specify multiple headers with multiple **/H** arguments.

- /In** Set the Importance where **n** is:

- 1 High
- 2 Normal (default)
- 3 Low

- /M** Use CRAM-MD5 authentication.

/Pn Set the Priority where **n** is:

- 0** Unspecified (default)
- 1** Normal
- 2** Urgent
- 3** Non Urgent

/R (Read receipt) : Send a read receipt.

/Sn Set the message sensitivity. The values are:

- 1** Personal
- 2** Private
- 3** CompanyConfidential

/V Show all the interaction with the server, except the message header and message body text.

3.13.11(SERVICEMONITOR

Purpose: Monitor service start, pause, and / or stop

Format: `SERVICEMONITOR [/C [name]]`
`SERVICEMONITOR name STARTED | PAUSED | STOPPED n command`

name Device name
n Number of repetitions (or **FOREVER**)
command Command to execute when condition is triggered

[/C\(lear\)](#)^[319]

Usage:

The service name can include wildcards.

The command line will be parsed and expanded before **SERVICEMONITOR** is executed, so if you want to pass redirection characters or variables to **command** you will need to protect them (by enclosing in single back quotes, doubling the %'s, or using command grouping).

If you don't enter any arguments, **SERVICEMONITOR** will display the services it is currently monitoring.

The monitoring runs asynchronously in a separate thread. When the condition is triggered, the command will be executed immediately. This may cause problems if you try to write to the display or access files while the main **TCC** thread is also performing I/O. You may need to use [START](#)^[337] or [DETACH](#)^[197] in **command** to avoid conflicts.

SERVICEMONITOR creates several environment variables when a service is started, paused, or stopped that can be queried by **command**. The variables are deleted after **command** is executed.

- _servicedisplay** Display name used by service control programs to identify the service
- _servicename** The name of the service in the service control manager database
- _servicecount** The number of times the command has been triggered

_servicestate The current state of the service. The possible values are:

- 1 The service is stopped
- 2 The service is starting
- 3 The service is stopping
- 4 The service is running
- 5 The service continue is pending
- 6 The service pause is pending
- 7 The service is paused

Options:

/C If **name** is specified, remove the monitor for that service. Otherwise, remove all service monitors.

3.13.11 SERVICES

Purpose: Display, stop, or start system services

Format: SERVICES [/P /R /S] [name ...]

[/P\(ause\)](#)^[319]
[/R\(un\)](#)^[319]

[/S\(top service\)](#)^[319]

Usage:

The **name** is the service name, not the display name. **name** can contain wildcards.

Options:

/P Pause after displaying each page.

/R Run the specified service(s).

/S Stop the specified service(s).

3.13.11 SET

Purpose: Display, create, modify, or delete environment variables.

Format: Display mode:
SET [/D /E /P /S /U /V /X] [wildname]

Definition mode:
SET [/A /D /S /U /V /E /R [file...]] | name=value | prompt]

Deletion mode:
SET [/D /S /U /V /E] name=

file One or more input files to read variable definitions from.
name The name of the environment variable
value The new value for the variable, separated from name by space[s]
prompt Optional input prompt for the **/P name=** option
wildname Name of variable[s] to be displayed. May contain * wildcard unless displaying registry variables

[/A](#) Arithmetic

[/S](#) System

/D ^[321]	Default	/U ^[322]	User
/E ^[322]	Environment, too	/V ^[323]	Volatile
/P ^[322]	Pause or Prompt	/X ^[323]	Override VariableExclude
/R ^[322]	Read from file(s)		

See also: [ESET](#)^[222] and [UNSET](#)^[357].

Usage:

Every program and command inherits an [environment](#)^[365], which is a list of pairs of variable **names** and **values**. Each **value** is a non-empty character string (i.e., there must be at least one character in it). Many programs use entries in the environment to modify their own actions. **TCC** itself uses several [environment variables](#)^[365].

If you simply type the SET command with no options or parameters, it will display all the names and values of all currently defined variables 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 only **name**, and there is no variable with that name, SET will display all environment variables whose names begin with **name**. For example, if there is no variable **pa**, the command below will display all variables whose names start with **pa**:

```
set pa
```

The above command is equivalent to the command

```
set pa*
```

If there is only a single parameter and it contains one or more wildcards (sorry, only * available), SET will display all matching environment variables. You cannot use wildcards to display the registry variables ([/D](#)^[322], [/S](#)^[322], [/U](#)^[323], and [/V](#)^[323]).

You can specify variables to exclude from the SET display with the VariableExclude variable. For example, to suppress the display of the processor and user variables:

```
set VariableExclude=proc*;user*
```

(Note that this option doesn't affect the existence of the variables, just whether they're displayed by a SET with no arguments.)

To add a variable to the environment, type SET, a space, the variable name, an equal sign, and the desired value:

```
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**.

Trailing whitespace in the SET command is ignored. To create a variable with trailing whitespace, use a pair of back quotes after the whitespace:

```
set mine=%@repeat[ ,20]``
```

makes **mine** 20 characters of spaces.

If you use SET to create a variable with the same name as one of the **TCC internal variables**^[372], 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. Once you delete your variable, the internal variable becomes accessible again.

To display the contents of a variable, type SET plus the variable name:

```
set mine
```

You can edit environment variables with the **ESET**^[222] command. To remove variables from the environment, use **UNSET**^[357], or type SET, followed by the variable name and an equal sign:

```
set mine=
```

The variable's **name** is limited to a maximum of 255 characters. **Name** and **value** together cannot be longer than 8,191 characters.

Note: You cannot use SET to modify **GOSUB variables**^[247].

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**^[322], **/S**^[322], **/U**^[323] and **/V**^[323] options respectively. For example, to display the contents of volatile variable **clientname**, use:

```
set /v clientname
```

Note that setting a registry variable using one of the options **/D**^[322], **/S**^[322], **/U**^[323] or **/V**^[323] **will not set** the variable in the local environment unless you also use the **/E**^[322] option.

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.

If the **Update Environment on System Change**^[47] configuration option is set, **TCC** will monitor the WM_SETTINGCHANGE message and update the environment from the User, Volatile, and System registry entries. The update is done whenever **TCC** displays the prompt (to prevent the environment from changing in the middle of a command).

Options:

- /A** Evaluate the arithmetic expression on the right of the equal sign, place the result in the environment, and display it. For example, this command adds 2 and 2, and places the

result in the environment variable **VAR**:

```
set /a var=2+2
```

/A interprets non numeric strings in **value** as environment variable names whether or not preceded by a percent sign %, and replaces them with their respective **values**. For example, this sequence will set **Y** to **4**:

```
set x=2
set /a y=x+2
```

You can use [@EVAL](#)^[420] to perform the same task; SET /A is included for compatibility with **CMD.EXE**. Unlike [@EVAL](#)^[420], use of the >> or << shift operators in SET /A requires disabling their interpretation as redirection symbols by using [SETDOS](#)^[323] /X-6.

/D Create/modify/delete a **default** variable in the registry (HKU\DEFAULT\Environment).

/E When used together with one of [/D](#)^[322], [/S](#)^[322], [/U](#)^[323], or [/V](#)^[323], set both the registry variable and the local environment variable.

/P 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](#)^[102].

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 [ESET](#)^[222] or [INPUT](#)^[259] command.

/R 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](#)^[126] for **TCC**. The file is in the same format as the SET display (i.e., **name=value**), so SET /R can accept as input a file generated by redirecting SET output. For example, the following commands will save the environment variables to a file, and then reload them from that file:

```
set > varlist
set /r varlist
```

You can load variables from multiple files by listing the filenames individually after the /R.

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](#)^[124]. However, you cannot use this method to exceed the command line length limit. You can also add comment lines to the file by starting each with a colon :. You can also use other special characters, e.g., redirection and pipe symbols (<> |), without the need for special handling (e.g., escaping). If you reference the value of another variable in **value** (e.g., **x=%path;c:\jpssoft**), evaluating that variable (**path** in the example) is postponed until at some future time a command line evaluates the current variable (**x** in the example), so that the command **echo %x** will display the **path** in effect when **echo** is executed, regardless of what **path** may have been when the original SET defined **x**.

If you do not specify a filename and input is redirected, **SET /R** will read from [stdin](#)^[535].

/S Create/modify/delete a **system** variable in the registry (HKLM\System\CurrentControlSet\Control\Session Manager\Environment).

- /U** Create/modify/delete a **user** variable in the registry (HKCU\Environment).
- /V** Create/modify/delete a **volatile** variable in the registry (HKCU\Volatile Environment).
- /X** Override the **VariableExclude** variable and display all matching variables.

3.13.11:SETDOS

Purpose: Display or set the **TCC** configuration.

Format: SETDOS [/A? /C? /D? /E? /Fn.n /G?? /I[+|-] *command* /M? /N? /P? /S?:? /V? /X[+|-]n]

/A(ANSI) ^[323]	/M(ode for editing) ^[325]
/C(ompound) ^[323]	/N(o clobber) ^[325]
/D(escriptions) ^[324]	/P(arameter character) ^[325]
/E(scape character) ^[324]	/S(hape of cursor) ^[325]
/F(ormat for @EVAL) ^[324]	/V(erbose) ^[325]
/G (numeric separators) ^[324]	/X (expansion, special characters) ^[325]
/I(nternal) ^[324]	

See also: [OPTION](#)^[284].

Usage:

SETDOS allows you to customize certain aspects of **TCC** to suit your personal tastes or the configuration of your system.

You can display the value of all SETDOS options by entering the SETDOS command with no parameters.

Most of the SETDOS options can also be changed in the [configuration dialogs](#)^[46]. The name of the corresponding configuration option is listed with each SETDOS option below; if none is listed, that option cannot be set from the configuration dialogs. You can also define the SETDOS options in your **TCSTART** or other startup file (see [Automatic Batch Files](#)^[22]), 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 [BDEBUGGER](#)^[166] command.

Inheritance

When a new instance of the command is started, it inherits the SETDOS characteristics set by the most recently started instance of **TCC**.

Options:

- /A** [ANSI] This option determines whether [ANSI X3.64 support](#)^[10] is enabled. **/A1** enables ANSI X3.64 string processing. The default of **/A0** disables ANSI X3.64 strings. See the [ANSI X3.64 Commands Reference](#)^[516] for a list of the ANSI X3.64 sequences supported by **TCC**. See also: the [ANSI Colors](#)^[49] configuration option and the [_ANSI](#)^[382] internal variable.
- /C** [Command Separator] 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](#)^[98] characters (**| > <**), or a space, tab, comma, or equal sign as the

command separator. The command separator is saved by [SETLOCAL](#)^[326] and restored by [ENDLOCAL](#)^[220]. The following example changes the separator character to a tilde [~]:

```
setdos /c~
```

- /D** [Descriptions and Description Name] This option controls whether file processing commands like [COPY](#)^[182], [DEL](#)^[190], [MOVE](#)^[274], and [REN](#)^[305] process file descriptions along with the files they belong to. **/D1** turns description processing on, which is the default. **/D0** turns description processing off. See also: the [Enable Descriptions](#)^[51] configuration option.

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:

```
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] This option sets the character used to suppress the normal meaning of the following character. Any character following the [escape character](#)^[124] will be passed unmodified to the command. The default escape character is a caret [^]. You cannot use any of the [redirection](#)^[98] characters (| > <) or a space, tab, comma, or equal sign as the escape character. The escape character is saved by [SETLOCAL](#)^[326] and restored by [ENDLOCAL](#)^[220]. Certain characters (**b**, **c**, **e**, **f**, **k**, **n**, **q**, **r**, **s**, and **t**) have special meanings when immediately preceded by the escape character.

- /F** [@EVAL maximum and minimum] This option lets you set the default decimal display precision for the [@EVAL](#)^[420] variable function. The maximum precision is 1,000 digits to the left of the decimal point and 1,000 digits to the right of the decimal point. (You can specify up to 10,000 digits in an @EVAL calculation by using the **=x,y** option.)

The format for this option is **/Fx.y**, where the x value sets the minimum number of digits to the right of the decimal point and the y value sets the maximum number of digits. You can use **=x,y** instead of **=x.y** if the comma is your decimal separator. Both values can range from 0 to 10. You can specify either or both values: **/F2.5**, **/F2**, and **/F.5** are all valid entries. If x is greater than y, it is ignored; if only x is specified, y is set to the same value (e.g. **/F2** is equivalent to **/F2.2**). See the [@EVAL Precision](#)^[51] configuration option to set the precision when **TCC** starts; see the [@EVAL](#)^[420] function if you want to set the display precision for a single computation.

- /G** [Decimal and thousands separator characters] This option sets the [Decimal](#)^[51] and [Thousands](#)^[51] 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](#)^[420], numeric [IF](#)^[253] and [IFF](#)^[254] tests, version numbers, and other similar uses. The thousands separator is used for numeric output, and is skipped when performing calculations in @EVAL.

- /I** 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 **TCC** to use an external command:

```
setdos /i-list
```

To re-enable all disabled commands use **/I***.

- /M** [Edit Mode] This option controls the initial line editing mode. To start in overstrike mode at the beginning of each command line, use **/M0** (the default). To start in insert mode, use **/M1**. See also: the [Edit Mode](#)^[50] configuration option.
- /N** [NoClobber] This option controls output [redirection](#)^[98]. **/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. See also: the [Protect Redirected Output File](#)^[47] configuration option.
- /P** [Parameter Character] This option sets the character used after a percent sign to specify all or all remaining command line parameters in a [batch file](#)^[130] or [alias](#)^[154]. The default value is the dollar sign **[\$]**. The parameter character is saved by [SETLOCAL](#)^[326] and restored by [ENDLOCAL](#)^[220].
- /S** [Insert and Overstrike Cursor] The cursor 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 can disable the cursor by specifying a size of 0:0.)
- If either value is -1, **TCC** 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 [Overstrike Cursor](#)^[50] and [Insert Cursor](#)^[50] configuration options.
- /V** [Batch Echo] This option controls the default for command echoing in batch files.
- /V0** disables echoing of batch file commands unless [ECHO](#)^[217] is explicitly set ON.
- /V1**, the default setting, enables echoing of batch file commands unless [ECHO](#)^[217] is explicitly set OFF. See also: the [Batch Echo](#)^[47] configuration option.
- /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 (in hex). All features are enabled when **TCC** 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 double quotes ["]) and square brackets
- 8 Escape character
- 9 [Include lists](#) ^[88]
- A [User-defined functions](#) ^[242]

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
```

A [SETLOCAL](#) ^[326] command will save the current SETDOS /X values for [ENDLOCAL](#) ^[220] to restore.

3.13.11 SETLOCAL

Purpose: Save a copy of the current disk drive, directory, environment, alias list, and special characters.

Format: SETLOCAL

See also: [ENDLOCAL](#) ^[220].

Usage:

SETLOCAL is valid only in batch files. It will save :

- the default disk drive and directory
- the environment,
- the alias list
- the special character set (command separator, escape character, parameter character, decimal separator, and thousands separator)
- the [SETDOS /X](#) ^[323] setting

After using SETLOCAL, you can change the values of any or all of the above, and later restore the original values with an [ENDLOCAL](#) ^[220] command, or just by exiting the batch file.

SETLOCAL does not save the command history or user functions.

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 SETLOCAL / ENDLOCAL pairs within a batch file, and nested batch files can each have their own SETLOCAL / ENDLOCAL pairs.

SETLOCAL does not override the [Local Aliases](#) ^[47] configuration option. Consequently changing aliases inside a SETLOCAL/ENDLOCAL pair affects the definition of aliases of other concurrently executing sessions of **TCC**.

You cannot use SETLOCAL in an alias or at the command line.

An ENDLOCAL is performed automatically at the end of a batch file, or when returning from a "[GOSUB](#) ^[247] filename". If you invoke one batch file from another without using [CALL](#) ^[175], 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 from inside a SETLOCAL / ENDLOCAL block. See [ENDLOCAL](#) ^[220] for details.

3.13.11!SHIFT

Purpose: Allows the use of more than 512 parameters in a batch file, or iterating through its parameters. This command can be used only in batch files.

Format: SHIFT [[-]*n* | /*n*]

n Number of positions to shift (an unsigned number), or the position of the parameter to be deleted.

Usage:

SHIFT is provided for compatibility with batch files written for CMD.EXE, where it was used to access more than the CMD.EXE limit of 10 parameters. **TCC** supports 4096 parameters (%0 to %4095), so you do not need to use SHIFT for batch files running exclusively under **TCC**.

SHIFT *n* moves each of the batch file parameters *n* positions to the left. The default value for *n* is 1. For example, SHIFT (with no parameters) makes the parameter %1 become to %0, the parameter %2 becomes %1, etc.

SHIFT -*n* moves parameters to the right, but it is limited to moving them back to their position on entry to the batch file.

This form of SHIFT also affects the special parameters %n\$, %\$ and %# (number of command parameters). However, for compatibility with **CMD.EXE**, this form of the SHIFT command does not alter the contents or order of the parameters returned by %*. See [Batch File Parameters](#) ^[133] for details.

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 the command below 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
```

SHIFT /n This form of the command irreversibly deletes parameter **%n** from the command tail, and shifts all parameters originally to its right 1 position to the left. For example,

```
shift /2
```

leaves parameters **%0** and **%1** unchanged, and moves the value of **%3** to position **%2**, **%4** to **%3**, etc.

This form of SHIFT also affects the special parameters **%n\$**, **%%\$** and **%%#** (number of batch file parameters), and unlike the first form, it also affects **%***. See [Batch File Parameters](#)^[133] for details.

3.13.11 SHORTCUT

Purpose: Create or display a shortcut.

Format: [Creation mode](#)^[328]
SHORTCUT command args dir desc link mode [iconfile [iconoffset [hotkey]]]

[Display mode](#)^[329]
SHORTCUT link

command	Command the shortcut executes
args	Command line parameters for command
dir	Starting directory
desc	Description
link	Filename of the .LNK or .PIF file.
mode	Initial window mode: 1=normal, 2=minimized, 3=maximized
iconfile	File containing the icon to use
iconoffset	Icon offset within iconfile
hotkey	Hotkey to invoke the shortcut

Usage:

Creation Mode

SHORTCUT creates a Windows shortcut file and places it in the specified directory. You can run any Windows shortcut from **TCC** by entering the name of the **.LNK** file on the command line.

SHORTCUT requires a minimum of 6 parameters. To leave a parameter blank, enter an empty string (2 double quotes "" in its place. Any parameter must be enclosed in double quotes if it includes white space or other special characters.

Command is the full path of the executable file to start, or the data file or folder to open. If it is a data file, its extension must be associated with an executable command (see [ASSOC](#)^[162]) for the shortcut to work.

The **args** parameter lists any command line parameters which you want to include when **command** is executed. For example, if **command** points to a batch file, you might want to include **%c** in **args** so that

TCC exits immediately when the batch file is completed.

The **dir** parameter is the path of the directory to which you want Windows to switch when the command starts. If you don't care which directory is used, you can omit this parameter by entering "" in its place.

Desc provides a description that is stored internally in the shortcut. It is displayed when the cursor is moved to the shortcut. If you omit the description, enter "" in its place.

The **link** parameter is the drive, path, name and extension of the shortcut file you want to create. The drive and path portion is interpreted according to the usual rules - missing elements default to the current defaults, path is relative to the current default unless it starts with \. 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 that Windows supports other extensions as well,

Note: If you want the shortcut to appear on the Windows desktop, you should include the full path to one of the desktop directory in the command. In most Windows configurations, that directory can be referenced symbolically as **%userprofile\Desktop**. Some Windows versions also include an **All Users\Desktop** directory.

The **mode** parameter 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.

The two (optional) parameters, **iconfile** and **iconoffset** allow you to specify the icon for the shortcut to use. (By default, SHORTCUT will use the default icon in the executable file.)

The final (optional) parameter **hotkey** specifies the keystroke which will call the shortcut. The keystroke should be entered in the same format as used in [KEYSTACK](#)^[262]; for example, **Ctrl-Alt-B**.

Display mode

If you provide a single parameter (a link file name), SHORTCUT will display the values for that link.

3.13.11 SHRALIAS

Purpose: Retains global command history, directory history, alias and user function lists in memory when **TCC** is not running.

Format: SHRALIAS [/U]

[/U\(nload\)](#)^[330]

Usage:

When you close all **TCC** 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 **TCC** 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 **TCC** is not running. SHRALIAS.EXE must be stored in the same directory as **TCC** 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.

If you have an environment variable named SHRALIAS_SAVE_PATH, SHRALIAS will save the alias, history, dirhistory, and function lists to the path specified by SHRALIAS_SAVE_PATH when SHRALIAS exits. The files will be saved in Unicode format as alias.sav, history.sav, dirhistory.sav, and function.sav.

SHRALIAS will not work unless you have at least one copy of **TCC** 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 **TCC** session which exits after starting SHRALIAS, the **TCC** 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](#) [194] command after SHRALIAS is loaded and before your session exits.

SHRALIAS will not work in detached sessions (i.e., those started with [DETACH](#) [197], or with the AT utility), due to security issues within Windows. Therefore the SHRALIAS command is ignored for detached sessions.

For more information about global histories, function and alias lists, see [Local and Global History Lists](#) [108], [Local and Global Functions](#) [242], [Local and Global Aliases](#) [154].

Option:

/U Shuts down SHRALIAS.EXE. All global command history, directory history, function and alias lists will be released from memory when the last copy of **TCC** exits unless SHRALIAS is loaded again before that time.

3.13.11!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](#) [316], [SNPP](#) [331].

Usage:

SMPP sends **message** through standard Internet Paging Gateways. Depending on your system configuration, you may need to start an Internet connection before using SMPP. See your service provider for specific requirements.

3.13.11!SNMP

Purpose: Send SNMP traps

Format: SNMP remotehost trapOID "value" [username password]

remotehost	Host name receiving the trap
trapOID	OID of the trap
value	Description
username	User name for SNMP v3 trap
password	Password for SNMP v3 trap

Usage:

SNMP normally sends an SNMPv2 trap. If you specify a user name and password it will send an SNMPv3 trap.

The following symbolic names are recognized and translated:

<u>Trap Name</u>	<u>OID</u>
coldStart	1.3.6.1.6.3.1.1.5.1
warmStart	1.3.6.1.6.3.1.1.5.2
linkDown	1.3.6.1.6.3.1.1.5.3
linkUp	1.3.6.1.6.3.1.1.5.4
authenticationFailure	1.3.6.1.6.3.1.1.5.5
egpNeighborLoss	1.3.6.1.6.3.1.1.5.6
enterpriseSpecific	1.3.6.1.6.3.1.1.5.7

3.13.12(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](#)^[316], [SMPP](#)^[330].

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 using SNPP.

3.13.12'START

Purpose: Start a program in another session or window.

Format: START ["title"] [/AFFINITY=n /ABOVENORMAL /BELOWNORMAL /HIGH /LOW /NORMAL /REALTIME /B /C /K /CM /Dpath /I /FS /INV /MAX /MIN /POS=x,y,width,height /L /LA /LD /LF /LH /MONITOR=n /RUNAS user password /SEPARATE /SHARED /SIZE=rows,cols /TAB /WAIT /WIN /PGM] "programe" [command]

title	Title to appear on title bar
path	Startup directory
programe	Program name (not the session name)
command	Command to be executed by programe

/ABOVENORMAL ^[333]	Priority	/LOW ^[334]	Priority
/AFFINITY ^[333]	Multiple CPUs	/MAX ^[334]	Maximized window
/B ^[333]	No new console	/MIN ^[334]	Minimized window
/BELOWNORMA ^[333]	Priority	/MONITOR ^[334]	Monitor to use
/L ^[333]		/NORMAL ^[334]	Priority
/C ^[333]	Close when done	/PGM ^[334]	Program name
/D ^[333]	Startup directory	/POS ^[334]	Position of window
/FS ^[334]	Full screen window	/REALTIME ^[334]	Priority
/HIGH ^[334]	Priority	/RUNAS ^[334]	Run as other user
/I ^[334]	Inherit environment	/SEPARATE ^[334]	Separate session
/INV ^[334]	Invisible window	/SHARED ^[334]	Shared session
/K ^[334]	Keep when done	/SIZE ^[334]	Screen buffer size
/L ^[334]	Local lists	/TAB ^[334]	Start in Take Command tab window
/LA ^[334]	Local aliases	/WAIT ^[334]	For session to finish
/LD ^[334]	Local directory history	/WIN ^[334]	Windowed session
/LF ^[334]	Local functions		
/LH ^[334]	Local history list		

See also: [DETACH](#)^[197].

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 **TCC** session. If you add a **command**, START will begin a new session or window and execute that command.

START will return to the **TCC** prompt immediately (or continue a batch file), without waiting for the program to complete, unless you use [/WAIT](#)^[334].

If **title** is included, will appear on the task list and **Alt-Tab** displays. instead of the program name, which is the default. **Title** must be enclosed in double quotes, and cannot exceed 127 characters.

START always assumes that the first quoted string on the command line is the **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), and you don't specify a **title**, START will interpret the first quoted string as the **title**, not the **command**. To address this, use the [/PGM](#)^[334] 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:

Valid

```
start /PGM "C:\Program Files\Proc.Exe"
start "test" "C:\Program Files\Proc.Exe"
```

Invalid

```
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.

Window controls: [/FS](#)^[334], [/MAX](#)^[334], [/MIN](#)^[334], and [/POS](#)^[334] allow you to start a character-mode windowed session in full screen mode, a maximized window, a minimized window, or a window with a specified position and size, respectively. [/INV](#)^[334] starts an invisible window. [/B](#)^[333] starts the program in the current console window. The default is [/WIN](#)^[334], which permits Windows to choose the position and size of the non-maximized window. If you start a graphics mode program, only [/MAX](#)^[334] and [/POS](#)^[334] are effective, and the position and size information associated with [/POS](#)^[334] is ignored. Windows will use the size, but not the position of the same program when last used in **RESTORE** mode. If you want to control the window size and placement of a graphics mode program, use the [/ACTIVATE](#)^[152] command after the window has been opened.

Session priority: The options [/ABOVENORMAL](#)^[333], [/BELOWNORMAL](#)^[333], [/HIGH](#)^[334], [/LOW](#)^[334], [/NORMAL](#)^[334] and [/REALTIME](#)^[334] allow you to select the new session's priority.

Program controls.

If **progrname** is in the "App Paths" registry, its associated "Path" value (if it exists) is inserted into the beginning of the [/PATH](#)^[368] in the environment inherited by the program.

If **progrname** is the name of a directory instead of an executable program, **TCC** will start your default Windows shell (usually Windows Explorer) in the specified directory.

Progrname inherits the environment as it exists when START is executed, unless [/I](#)^[334] is used to select the default environment.

If **progrname** specifies **TCC.EXE**, the options [/L](#)^[334], [/LA](#)^[334], [/LD](#)^[334], [/LF](#)^[334] and [/LH](#)^[334] provide control over the use of local or global lists. See details below.

The initial directory for **progrname** is the current default directory, unless otherwise specified using the [/D](#)^[333] option. **WARNING!** If **progrname** is a DOS program, this option may not work properly. To avoid such problems, you can change the default directory before START, and restore it afterwards.

If **progrname** is a 16-bit Windows application, by default it starts in a shared virtual machine. You may use the [/SEPARATE](#)^[334] option to force creation of a unique virtual machine.

When **command** is finished, [/C](#)^[333] closes the session (the default for Windows sessions), while [/K](#)^[334] keeps it and displays the prompt (the default for character mode sessions).

The Process ID of the detached session or program is returned in the [/STARTPID](#)^[392] internal variable.

Options:

/ABOVENORMAL	Set the priority above normal.
/AFFINITY=n	On multiple processor machines, set the processor affinity for this process. Acceptable values are 0 to <i>n</i> -1 (where <i>n</i> is the number of available processors)
/BELOWNORMAL	Set the priority below normal.
/B	The program is started without creating a new window or console, <i>i.e.</i> in the TCC 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 the session or window when the application ends.
/D	Specifies the startup directory. Include the directory name immediately after the /D , with no intervening spaces or punctuation. Due to limitations in the way Windows starts DOS programs, /D is ignored when starting DOS applications.

/FS	Start the console application in full-screen mode. (Not supported in Windows Vista or later.)
/HIGH	Start the window at high priority.
/I	Inherit the default (startup) environment, rather than the current environment.
/INV	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 ^[227] command to end the session.
/L	Start TCC with local alias, function, history and directory history lists. This option is equivalent to specifying all of /LA ^[334] , /LD ^[334] , /LF ^[334] , and /LH ^[334] (below).
/LA	Start TCC with a local alias list. See ALIAS ^[154] for information on local and global alias lists.
/LD	Start TCC with a local directory history list. See Local and Global History Lists ^[108] for information on local and global directory history lists.
/LF	Start TCC with a local function list. See FUNCTION ^[242] for information on local and global function lists.
/LH	Start TCC with a local history list. See Local and Global History Lists ^[108] for information on local and global history lists.
/LOW	Start the window at low priority.
/MAX	Start the session or window maximized.
/MIN	Start the session or window minimized.
/MONITOR=<i>n</i>	Start the program on the specified monitor (1 to <i>n</i>). This will only work with apps that do not try to position their window at startup, and you cannot combine this switch with /POS .
/NORMAL	Start the window at normal priority.
/PGM	The quoted string following this option is the program name. Any additional text beyond the quoted string is passed to the program as its parameters, 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=<i>left,top,width,height</i>	Start the window at the specified screen position. The top left corner of the screen is 0,0.
/REALTIME	Start the window at realtime priority.
/RUNAS	Run a command in the context of the specified user. The syntax is: /RUNAS user@domain password . If "domain" is not specified, the local database is checked for the username. If you specify * for the password, START will prompt you to enter the password. (Useful when you don't want to put the password in a batch file.)
/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, see /SHARED ^[334] .
/SHARED:	Start a 16-bit Windows application in the shared virtual machine (default). See also /SEPARATE ^[334] . Included only for compatibility with CMD.EXE.
/SIZE=<i>rows,columns</i>	Specifies the screen buffer size. Rows is the number of text rows and columns is the number of text columns. (This is not the size of the session's window.)
/TAB	Start the command in a new TCC tab window.
/WAIT	Wait for the new session or window to finish before continuing.
/WIN	Start the new console session as a window (this is the default.) See also /FS ^[334] and /B ^[333] .

3.13.12 SWITCH

Purpose: Select commands to execute in a batch file based on a value.

Format: SWITCH expression
CASE value1 [.OR. value2 [.OR. value3 ...]]
[commands]
CASE value4
[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](#)^[253] and [IFF](#)^[254].

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 **expression**, an optional DEFAULT statement, and terminated by an ENDSWITCH statement. Each CASE statement and the DEFAULT statement may be followed by one or more **commands**.

TCC evaluates **expression**, and sequentially compares it with the list of **values** in the CASE statements, starting with the first one. Comparison rules are the same ones used for the **EQ** relational operator; see [Numerical and String Comparisons](#)^[110] for details. If a match is found, the **commands** following the matched CASE statement are executed, and the batch file continues with the commands that follow ENDSWITCH. If there are any matches in subsequent CASE statements, they are ignored.

If during the search for a match the DEFAULT statement is encountered, the **commands**, if any, following it are executed, and the batch file continues with the commands that follow ENDSWITCH. Any CASE statements after the DEFAULT statement are ignored.

SWITCH commands can be nested.

You can exit from all SWITCH / ENDSWITCH processing by using [GOTO](#)^[248] to a line past the last ENDSWITCH.

Restrictions

Each SWITCH, CASE, DEFAULT and ENDSWITCH statement must be on a separate line, and may not be followed by a command separator. (This is the reason SWITCH cannot be used in aliases.) There is no restriction on grouping and command separator use in the **commands** for a CASE or DEFAULT.

You can link a list of values in a single CASE statement with .OR., but not with .AND. or .XOR..

Examples

The batch file fragment below displays one message if the user presses **A**, another if the user presses **B** or **C**, and a third one 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 **expression**. However, you can use other kinds of expressions if necessary. The first SWITCH statement 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
```

3.13.12 SYNC

Purpose: Synchronize two directories

Format: SYNC [/A:... /C /D /E /F /G /J /K /L /M /N[est] /O /P /Q /R /S[n] /T /V /W /X] dir1 dir2

dir1 First directory (and source for a /W)
dir2 Second directory (and target for a /W)

<u>/A:</u> ³³⁷ ...	Attribute switch	<u>/M</u> ³³⁶	Modified files (not Archived)
<u>/C</u> ³³⁶	Changed source files	<u>/N</u> ³³⁶	Disable
<u>/D</u> ³³⁶	Copy encrypted files	<u>/O</u> ³³⁶	Only if no target file
<u>/E</u> ³³⁶	No error messages	<u>/P</u> ³³⁶	Prompt
<u>/F</u> ³³⁶	No empty subdirectories	<u>/Q</u> ³³⁶	Quiet
<u>/G</u> ³³⁶	Display percentage completed	<u>/R</u> ³³⁶	Replace
<u>/H</u> ³³⁶	H(idden included)	<u>/S</u> ³³⁶	Subdirectories included
<u>/I"text"</u>	Match description	<u>/T</u> ³³⁶	Totals

/J ^[336]	Restartable copy	/V ^[336]	Verify
/K ^[336]	Keep RDONLY attribute	/W ^[336]	Delete non-matching target
/L ^[336]	ASCII-mode FTP transfer	/X ^[336]	(Clear archive bit)

See also: [COPY](#)^[182] and [MOVE](#)^[274].

File Selection

Supports extended [wildcards](#)^[77] and [ranges](#)^[80].

Internet: Can be used with [FTP servers](#)^[93].

Usage:

SYNC will synchronize two directories, copying the updated files from each directory to the other.

Options:

- /A:...** Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[86] for information on the attributes which can follow /A:. See the cautionary note under **Advanced Features** above before using /A: when both **dir1** and **dir2** contain file descriptions. Do not use /A: with **@file** lists. See [@file lists](#)^[90] for details. Hidden or system files selected by this option overwrite hidden or system files in the target directory.
- /C** Copy files only if the destination file exists and is older than the source file. This option is useful for updating the files in one directory from those in another without copying any files not already in the target directory. Do not use /C with **@file** lists. See [@file lists](#)^[90] for details.
- /D** (Windows XP+ Only) Force copy of an encrypted file even when the target will be decrypted.
- /E** 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.
- /F** When used with **/S**, SYNC will not create any empty subdirectories.
- /G** Displays the percentage copied, the transfer rate (in Kbytes/second), and the estimated time remaining. Useful when copying large files across a network or via FTP to ensure the copy is proceeding. When [/V](#)^[189] is also used, reports percentage verified.
- /H** 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 **dir1** and **dir2** contain file descriptions.
- /I "text"** Select source files by matching text in their descriptions. See [Description Ranges](#)^[86] for details.
- /J** Copy the files 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 read-only attribute) SYNC normally maintains the hidden and system attributes, sets the archive attribute, and removes the read-only attribute on the target file. **/K** tells

SYNC to also maintain the read-only attribute on the **destination** file.

- /L** Perform FTP transfers in ASCII mode, instead of the default binary mode.
- /M** 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](#)^[189] switch, or use [ATTRIB](#)^[163]. Do not use /M with [@file](#) lists. See [@file lists](#)^[90] for details.
- /N** Do everything except actually perform the copy. This option is useful for testing the result of a complex SYNC command. /N displays how many files would be copied. /N does not prevent creation of destination subdirectories when it is used with [/S](#)^[188].

A **/N** with one of the following arguments has an alternate meaning:
 - d** Skip hidden directories (when used with /S)
 - e** Don't display errors.
 - j** Skip junctions (when used with /S)
 - s** Don't display the summary.
 - t** Don't update the CD / CDD [extended directory search](#)^[73] database ([JPSTREE.IDX](#)).
- /P** Ask the user to confirm each source file. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[102]. See also: the [/Q](#)^[188] option below.
- /Q** Don't display filenames, percentage copied, total number of files copied, etc... When used in combination with the [/P](#)^[188] 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](#)^[189].
- /R** Prompt the user before overwriting an existing file. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[102].
- /S** Copy the subdirectory tree starting with the files in the source directory plus each subdirectory below that. If the destination subdirectories don't exist, SYNC will attempt to create them. If SYNC /S creates one or more destination directories, they will be added automatically to the [extended directory search](#)^[73] database.

If you attempt to use SYNC /S to copy a subdirectory tree into part of itself, SYNC will detect the resulting infinite loop, display an error message and exit. Do not use /S with [@file](#) lists. See [@file lists](#)^[90] for details.

If you specify a number after the /S, SYNC will limit the subdirectory recursion to that number. For example, if you have a directory tree "a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.
- /T** Turns off the display of filenames, like [/Q](#)^[188], but does display the total number of files copied.
- /V** Verify each disk write by performing a true byte-by-byte comparison between the source and the newly-created target file. This option will significantly increase the time necessary to complete a SYNC command.
- /W** Delete files in **dir2** that do not exist in **dir1**.
- /X** Clear the archive attribute from the source file after a successful copy.

3.13.12 TAIL

Purpose: Display the end of the specified file(s).

Format: TAIL [range ... [[/I](#) "text"]] [[/A](#): [attrlist] [/C](#) nn [/F](#) [/N](#)+x [/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 name of a file to display in each line (see [@file lists](#) for details).

/A : (Attribute select)	/N (umber of lines)
/C (number of bytes)	/P (ause)
/F (ollow)	/Q (uiet)
/I "text" (description range)	/V (erbose)
/N +x (skip x lines before display)	

See also: [HEAD](#), [LIST](#), and [TYPE](#).

File Selection

Supports extended [wildcards](#), [ranges](#), [multiple file names](#), and [include lists](#).

Internet: Can be used with [FTP servers](#), 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*:

```
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](#) for additional information on **CLIP:**.

• FTP Usage

TAIL can also display files on [FTP servers](#). For example:

```
tail "ftp://jpsoft.com/index"
```

You can also use the [IFTP](#) command to start an FTP session on a server, and then use an abbreviated syntax to specify the files and directories you want.

• NTFS File Streams

TAIL supports file streams on NTFS drives. You can type an individual stream by specifying the stream

name, for example:

```
tail streamfile:s1
```

• Pipes

TAIL can optionally be used with an input [pipe](#)^[107]. For example:

```
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 less than a million bytes when the input is piped.

Examples

<code>tail /n 5 xxx</code>	displays the last 5 lines of file xxx
<code>tail /n+20 /n 999999 xxx</code>	skip 20 lines, then display 999999 lines of xxx
<code>tail /n+1001 /n 1 xxx</code>	skip 1001 lines, then display 1 line of xxx

`set x=%@execstr[tail /n+1001 /n 1 xxx]` sets **x** to the contents of the 1002-nd line of xxx

`set x=%@execstr[tail /n 2 xxx]` sets **x** to the contents of the penultimate line of xxx

Options:

/A:[attributelist]

Select only those files that match the specified attribute(s). See [Attribute Switches](#)^[86] for details.

/Cnn[b/k/m]

Display **nn** bytes, 512-byte **b**locks, **k**ilobytes, or **m**egabytes, when there is no suffix, for suffix b, suffix k, or suffix m, respectively.

/F Continuously monitor the file and display new lines until the command is interrupted, e.g, using **Ctrl-C** or **Ctrl-Break**.

/I"text"

Select files by a [descriptor range](#)^[86]. See the link for details.

/N n Display **n** lines. The default is **10**. Space between the option switch /N and the number **n** is optional. If /N is specified without **n**, it is equivalent to specifying 0 lines to be displayed, and the command will not generate output, unless [/V](#)^[347] is also specified.

/N+x Skip **x** lines from the beginning of the file, than start displaying lines. If the **/N+** option is specified without specifying **x**, the option is ignored. This option does not affect the number of lines displayed (unless the start line is too close to the end of file)

Example: `TAIL /N+5 file` will display 10 lines (the default) after skipping 5 lines.

/P Pause and prompt after displaying each page.

/Q 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](#)^[347].

/V Display a header for each file.

3.13.12 TASKDIALOG

Purpose: Display a Windows Vista Task Dialog

Format: TASKDIALOG [/I /S /W] buttontype "title" "instruction" [text]

buttontype One or more of **OK**, **CANCEL**, **YES**, **NO**, **RETRY**, **CANCEL**, and/or **CLOSE**
title Text for the task dialog title
instruction Text for the main instruction
text Optional additional text that appears below the main instruction, in a smaller font

[/I\(nformation icon\)](#)^[281]
[/S\(top icon\)](#)^[281]

[/W\(arning icon\)](#)^[281]

See also: [INKEY](#)^[257], [INPUT](#)^[259], [MSGBOX](#)^[279] and [QUERYBOX](#)^[300].

Usage:

TASKDIALOG requires Windows Vista or later.

The button the user chooses is indicated using the internal variable [%_?](#)^[380]. Be sure to save the return value in another variable or test it immediately; because the value of [%_?](#)^[380] changes with every internal command. The following list shows the value returned for each button:

response	%_? ^[380]
Yes or OK	10
No	11
Cancel or Close	12
Retry	13

If there is an error in the TASKDIALOG command itself, [%_?](#)^[380] will be set to 2.

For example, to display a Yes / No message box and take action depending on the result, you could use commands like this:

```
taskdialog yes no "Copy" "Copy all files to A:?"
if %_? == 10 copy * a:
```

Since TASKDIALOG doesn't write to standard output, it disables redirection and piping to allow you to enter the redirection characters (<, >, and |) in your prompt text.

TASKDIALOG creates a popup dialog box. If you prefer to retrieve input from the command line, see the [INKEY](#)^[257] and [INPUT](#)^[259] commands.

Options:

/I Display an icon consisting of a lower case "i" in a circle in the message box.
/S Display a stop sign icon in the message box.
/W Display an exclamation point icon in the message box.

3.13.12 TASKEND

Purpose: End the specified process.

Format: TASKEND [/F] pid | name | "title"

pid	The process ID
name	The process name
title	Window title

[/F\(orce\)](#)^[342]

See also: [TASKLIST](#)^[342], [_PID](#)^[390], [_DETACHPID](#)^[385], [_WINTITLE](#)^[394]

Usage:

Windows applications (and Windows itself) run as one or more processes or tasks. You can use the TASKLIST command to display a list of currently-running tasks. TASKEND can be used to end a task.

When you use TASKEND, you must specify the task you want to end by process ID number, by name (usually the name of the executable file that started the task) or by window title. If you use the Window title to specify the task, you must enclose it in double quotes. You can use wild cards and extended wildcards in the window title.

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.

Using this command may require DEBUG privilege.

Option:

/F	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.
-----------	--

3.13.12 TASKLIST

Purpose: Display a processes list

Format: TASKLIST [/O /P] [name]

name	Process name or window title
-------------	------------------------------

[/O\(rder by PID\)](#)^[343]

[/P\(ause\)](#)^[343]

See also: [TASKEND](#)^[342].

Usage:

Windows programs 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.

TASKLIST will display a * after the process ID of the current process.

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](#)^[77].

Options:

- /O** Sort the output by Process ID (PID).
- /P** Wait for a key to be pressed after each screen page before continuing the display.

3.13.12 TCFILTER

Purpose: Display or change the List View filter

Format: TCFILTER [/C] filter

filter A regular expression new filter

[/C\(lear\)](#)^[342]

See also: [_TCFILTER](#)^[392]

Usage:

TCFILTER allows you to set the filter used by the **Take Command** List View to determine what file and folder names to display. For example, to only display files with a .DOC extension in the List View:

```
tcfilter *.doc
```

See [Regular Expression Syntax](#)^[496] for details on valid regular expressions.

Option:

- /C** Clear the current filter

3.13.12 TCTOOLBAR

Purpose: Change the tool bar buttons.

Format: TCTOOLBAR [/C /U /R filename] button [, flags, icon, label, command]

button	The button number (1 – 50)
flags	0=Start new tab, 1=Send to current tab
icon	Icon to display on button
label	The button label
command	The command to execute or keystrokes to send

[/C\(lear\)](#)^[344]

[/R\(ead file\)](#)^[344]

[/U\(pdate\)](#)^[344]

Usage:

TCTOOLBAR lets you configure the **Take Command** tab tool bar buttons (you can also use the [Configure Tool Bar dialog](#)^[69] available from the [Options](#)^[61] menu). The changes you make can be temporary or, with the **/U** option, written to the **TCMD.INI** file so that they will be loaded the next time

Take Command starts.

There are a maximum of 50 buttons on the tab tool bar. The **button** parameter must be a number from 1 to 50 to select the button you want to work with. If you enter a command like

```
tctoolbar 1
```

the button with that number will be removed from the tool bar. If you want to add or modify a button, you must include the **flags**, **icon** and/or **label**, and **command** parameters.

The **flags** parameter specifies what happens when you click the button. If **flags** is 0, **Take Command** will use **command** to start a new tab. If **flags** is 1, the **command** text (in [KEYSTACK](#)^[262] format) is sent to the current tab. You can optionally add 2 to the value of **flags** to insert a separator before the button.

The **icon** parameter allows you to specify the name of an icon file (or an executable file if you want to use its default icon). The icon will be displayed to the left of the button label. If you have entered a **label**, the **icon** parameter is optional.

The **label** parameter specifies the text that appears on the button. If the text contains white space or other special characters, it must be enclosed in double quotes. If you have entered an **icon** file, **label** is optional.

The **command** parameter contains the command to start a new tab (if **flags=0**), or the keystrokes to be sent to the current tab in [KEYSTACK](#)^[262] format (if **flags=1**) when the button is clicked.

Option:

- /C** Clear all entries from the toolbar.
- /R** Load the toolbar button definitions from the specified file. **/R** will **not** clear an existing toolbar; you must use **/C** for that. The file should be in the same format as the **[Buttons]** section in TCMD.INI:

```
Bn=flags,icon,label,command
```

n - the button number (1 - 50)

flags - 0=start new tab, 1=send keystrokes to current tab

icon - the icon to display on the label (leave empty for no icon)

label - the label to display on the button

command - the command to execute

- /U** Write the changed button definition to the **TCMD.INI** file so that it will be included the next time **Take Command** starts.

3.13.13(TEE

Purpose: Copy standard input to both standard output and a file.

Format: TEE [/A /D /T] file...

file One or more files that will receive the "tee-d" output.

[/A\(ppend\)](#)^[345]
[/D\(ate\)](#)^[345]

[/T\(ime\)](#)^[345]

See also: [Y](#)^[364], [piping](#)^[101] and [redirection](#)^[98].

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 to standard output, the other to the **file(s)** 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*:

```
ffind /t"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.

See [Piping](#)^[101] for more information on pipes.

Option:

- /A** Append to the file(s) rather than overwriting them.
- /D** Prefix each line with the current date (in yyyy-mm-dd format).
- /T** Prefix each line with the current time (in hh:mm:ss.ms format).

3.13.13 TEXT

Purpose: Display a block of text in a batch file.

Format: TEXT
 .
 .
 .
 ENDTEXT

See also: [ECHO](#)^[217], [ECHOS](#)^[219], [SCREEN](#)^[310], [SCRPUT](#)^[311], and [VSCRPUT](#)^[360].

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 lines in the batch file between itself and the terminating ENDTEXT. The display starts at the current display position, which allows you to start its display with other text, e.g., from the [ECHOS](#)^[219] command.

The lines between TEXT and ENDTEXT 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. This makes it easy to include special characters, e.g., < | > in the text. 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](#)^[181] or the [COLOR](#)^[181] command to set the default screen colors before executing TEXT.

Redirecting TEXT output

To redirect or pipe the entire block of text, use [redirection](#)^[98] or [piping](#)^[101] on the TEXT command itself as shown in the [Examples](#)^[346] below. As with any other command, this redirection is not affected by redirection of all output of the batch file by the command which started the batch file. Attempting to redirect or pipe the actual text lines is ignored. Attempting to redirect or pipe the ENDTEXT line is invalid.

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](#)^[282] 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):

```
@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
```

3.13.13 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\(erver time\)](#)^[347]

[/T \(Display only\)](#)^[347]

See also: [DATE](#)^[189].

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 Jan 31, 2008 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:

```
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 Sets the date and time from the specified internet time server. If no server is specified, TIME uses the server defined in the [Time Server](#)^[53] configuration option (the default is clock.psu.edu).

/T Displays the current time but does not prompt you for a new time. You cannot specify a new time on the command line with **/T**. If you do, the new time will be ignored.

3.13.13 TIMER

Purpose: TIMER is a system stopwatch.

Format: TIMER [ON | OFF | /Q /S /1 /2 /3]

ON Force the stopwatch to reset and start
OFF Force the stopwatch to stop

[/1](#)^[348] stopwatch #1 (default)

[/Q](#)^[349] quiet

[/2](#)^[348] stopwatch #2

[/S](#)^[349] split

[/3](#)^[348] stopwatch #3

 348**Usage:**

The TIMER command accepts its parameters in any order, and acts on the specified one of three possible timers (system stopwatches) by turning it on or off, or by displaying its current elapsed time. The TIMER command with neither of the keywords **ON** and **OFF** nor the **/S** option toggles the state of the timer.

If you execute TIMER or TIMER /S when the timer is off, or execute TIMER ON at any time, the current time of day is displayed, and the stopwatch starts from :

```
[c:\] timer
Timer 1 on: 12:21:46
```

If you execute TIMER /S when the timer is on, the elapsed time is displayed:

```
[c:\] timer /s
Timer 1 Elapsed time: 0:00:12.06
```

If you execute TIMER when it is on, or execute TIMER OFF, the stopwatch stops, the current time and the elapsed time are displayed, and the elapsed time is reset:

```
[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 more than 60 ms.

You can also retrieve the elapsed time of a timer using the [@TIMER\[\]](#)⁴⁶³⁾ function.

Options:

- /1** Use timer #1 (the default).
- /2** Use timer #2.
- /3** Use timer #3.

/Q Don't display any messages.

/S 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.

3.13.13 TITLE

Purpose: Change the window title.

Format: TITLE [/P] title

[/P\(prompt characters\)](#)^[349]

title The new window title.

See also: the [TITLEPROMPT](#)^[370] variable and the [ACTIVATE](#)^[152] and [WINDOW](#)^[362] commands.

Usage:

TITLE changes the text that appears in the caption bar at the top of the **TCC** window. 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":

```
title Title Test
```

Options:

/P Support the special characters in [PROMPT](#)^[297].

3.13.13 TOUCH

Purpose: Change a file's [time stamps](#)^[495], and optionally create a file.

Format: TOUCH [/A:[-][+][rhsdaecjot](#)] /C [/D[acw][date] /E /F /I"text" /N /Q /R[:acw]file /Sn /T[acw[u]][hh:mm[:ss[:dd]]] file...

file One or more files whose date and/or time stamps are to be changed.

[/A](#)^[350] Attribute select

:

[/C](#)^[350] Create file

[/D](#)^[350] Date

[/N](#)^[351] No action

[/Q](#)^[351] Quiet

[/R](#)^[351] Reference file

/E ^[350] No error messages	/S ^[351] Subdirectories
/F ^[350] Force read-only files	/T ^[351] Time
/I ^[351] Match descriptions	

File Selection:

Supports [attribute switches](#)^[86], extended [wildcards](#)^[77], [ranges](#)^[80], [multiple file names](#)^[87], subdirectories, [catalog files](#)^[90], and [include lists](#)^[88].

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.

By default, TOUCH affects only files. You must utilize the [/A](#)^[350] option to include directories. /A:D will select directories only.

Options:

- /A:** Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[86] for information on the attributes which can follow /A:.
- /C** Create **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** If neither [/R](#)^[351] nor [/D](#)^[350] are specified, the current date is used. If the [/D](#)^[350] option is specified without date, TOUCH will not modify the date even if [/R](#)^[351] is also specified. If the [/D](#)^[350] option is followed by date, and [/R](#)^[351] is not specified, date is used. The date must not be quoted. If both [/R](#)^[351] and [/D](#)^[350] with date are specified, the one specified later in the command takes effect.

On an LFN drive, you can specify which of the date fields should be set by appending **a**, **c**, or **w** to the **/D** option:

- a** Last access date
- c** Creation date
- w** Last modification (write) date

If you append a **u** to the date field, TOUCH will set the UTC date rather than the local date.

- /E** 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** The file systems normally do not permit changing timestamps of read only files. The [/F](#)^[350] option forces date and time change of read-only files by temporarily removing the read only attribute.

/I "text" Select files by matching text in their descriptions. See [Description Ranges](#)^[86] for details.

/N Display what would occur without actually doing it.

/Q Do not display normal messages.

/R The /R option permits duplication of the timestamp of **ref_file**. For example, if you recompile an old program (e.g., to obtain an intermediate file that has long been deleted) you may want to use the timestamp of the source file that was last changed as the timestamp of the newly built duplicate of the original object file to prevent a "make" from attempting to rebuild everything else in the project as shown in the example:

```
touch /r project.c project.obj
```

Another use could be to synchronize files without rendering the current version inaccessible during the synchronization:

```
touch /c /r c:\jpsoft\tcmd.chm %temp%\tcmd.chm
copy /u ftp://jpsoft.com/help/tcmd.chm %temp%\tcmd.chm
```

In the above example TOUCH creates an empty file with the timestamp of your already existing help file; [COPY](#)^[182] updates the empty file if a newer version is available (beware of timestamp synchronization across the Internet!).

On an LFN drive, you can specify which of the date/time fields should be used by appending **a**, **c**, or **w** to the **/R** option:

- a** Last access date and time (on VFAT volumes access time is always midnight).
- c** Creation date and time
- w** Last modification (write) date and time

/S TOUCH all matching files in the specified directory and its subdirectories. Do not use /S with @file lists. See [@file lists](#)^[90] for details.

If you specify a number after the /S, TOUCH will limit the subdirectory recursion to that number. For example, if you have a directory tree "a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.

/T If neither [/R](#)^[351] nor [/T](#)^[351] are specified, the current time is used. If the [/T](#)^[351] option is specified without time, TOUCH will not modify the time even if [/R](#)^[351] is also specified. If the [/T](#)^[351] option is followed by time, and [/R](#)^[351] is not specified, time is used. (Time must not be quoted). If both [/R](#)^[351] and [/T](#)^[351] with time are specified, the one specified later in the command takes effect.

On an LFN drive, you can specify which of the time fields should be set by appending **a**, **c**, or **w** to the **/T** option:

- a** Last access time (on VFAT volumes access time is always midnight).
- c** Creation time
- w** Last modification (write) time

If you append a **u** to the time field, TOUCH will set the UTC time rather than the local time.

3.13.13 TRANSIENT

Purpose: Toggle the shell's transient mode

Format: TRANSIENT [on | off]

Usage:

TRANSIENT allows you to change the shell's transient mode (i.e., whether it was started with a /C), so that you can make a transient session permanent (or vice versa).

3.13.13 TREE

Purpose: Display a graphical directory tree.

Format: TREE [[/A:[-|+]*rhs*adecijopt /A /B /D /F /H /Nj /P /S[n] /T[:a|c|w] /Z] *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: (Attribute select) ^[352]	/N (disable option) ^[353]
/A(SCII) ^[352]	/P(ause) ^[353]
/B(are) ^[353]	/S (file size) ^[353]
/D(escriptions) ^[353]	/T(ime and date) ^[353]
/F(iles) ^[353]	/Z (file size) ^[353]
/H(idden directories) ^[353]	

File Selection:

Supports [attribute switches](#)^[86], extended [wildcards](#)^[77], [ranges](#)^[80] (with /F), and [multiple file names](#)^[87].

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](#)^[352], or configure the **TCC** 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](#)^[264] by using standard [redirection](#)^[98] symbols. Be sure to review the [/A](#)^[352] option before attempting to print the TREE output. The options discussed below specify the amount of information included in the display.

Options:

/A	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.
/A:[..]	Select only those files that match the specified attribute(s). See Attribute Switches ^[86]

for details.

- /B** Display the full pathname of each directory, without any of the line-drawing characters.
- /D** Display file and directory descriptions.
- /F** 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** Display hidden as well as normal directories. If you combine **/H** and [/F](#)^[353], hidden files are also displayed.
- /N** Disables the specified options:
 - j** Skip junctions (when used with [/S](#)^[353])
- /P** 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](#)^[102].
- /S** If you specify a number after the **/S**, TREE will limit the subdirectory recursion to that number. For example, if you have a directory tree "a\b\c\d\e", **/S2** will only affect the "a", "b", and "c" directories.
- /T** Display the time and date for each directory. If you combine **/T** and [/F](#)^[353], 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](#)^[536] and [FAT32](#)^[528] volumes).
 - /T:c** Creation date and time.
 - /T:w** Last modification ("write") date and time (default).
- /Z** Display the size of each file. This option is only useful when combined with [/F](#)^[353].

3.13.13 TRUENAME

Purpose: Find the full, true path and file name for a file

Format: TRUENAME file

See also: The [@TRUENAME](#)^[464] variable function.

Usage:

Network reassignments, junctions, symbolic links, and the SUBST 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:\] truenam d:\test.exe
```

```
c:\util\test\test.exe
```

3.13.13 TYPE

Purpose: Display the contents of the specified file(s).

Format: TYPE [/A:[-][+]*rhsadecijopt*] /!"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](#)^[90] for details).

[/A: \(Attribute select\)](#)^[355] [/L \(line numbers\)](#)^[355]
[/!"text" \(match description\)](#)^[355] [/P \(pause\)](#)^[355]

See also: [HEAD](#)^[249], [TAIL](#)^[339], [LIST](#)^[264].

File Selection

Supports [attribute switches](#)^[86], extended [wildcards](#)^[77], [ranges](#)^[80], [multiple file names](#)^[87], and [include lists](#)^[88].

Internet: Can be used with [FTP and HTTP servers](#)^[93], e.g.

```
type "http://jpsoft.com/notfound.htm"
```

Usage:

The TYPE command displays a file. It is normally only useful for displaying text files (i.e. alphanumeric characters arranged in lines separated by CR/LF). Most text files use either ASCII or Unicode.

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:

```
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](#)^[98] 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](#)^[98] is useful if you want to add line numbers to a file, for example:

```
type /l myfile > myfile.num
```

• NTFS File Streams

TYPE supports file streams on NTFS drives. You can type an individual stream by specifying the stream name, for example:

```
type streamfile:s1
```

See [NTFS File Streams](#)^[496] for additional details.

Options:

- /A:** Select only those files that have the specified attribute(s) set. See [Attribute Switches](#)^[86] for information on the attributes which can follow **/A:**. Do not use **/A:** with @file lists. See [@file lists](#)^[90] for details.
- /I"text"** Select files by matching text in their descriptions. The text can include [wildcards](#)^[77] and extended wildcards. The search text must be enclosed in double quotes, 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](#)^[90] for details.
- /L** Display a line number preceding each line of text.
- /P** Prompt after displaying each page. Your options at the prompt are explained in detail under [Page and File Prompts](#)^[102].

3.13.14 UNALIAS

Purpose: Remove aliases from the alias list.

Format: UNALIAS [/Q /R file... (alias ...)] 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\(quiet\)](#)^[356]

[/R\(read file\)](#)^[356]

See also: [ALIAS](#)^[154] and [ESET](#)^[222].

Usage:

TCC 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.

For example, to remove the alias DDIR:

```
unalias ddir
```

To remove all the aliases:

```
unalias *
```

To remove all the aliases that begin with "DD":

```
unalias dd*
```

You can delete all matching aliases except for those specified by enclosing the exceptions in parentheses. For example, to remove all aliases beginning with "a" except for *alias1* and *alias2*:

```
unalias (alias1 alias2) a*
```

If you keep aliases in a file that can be loaded with the [ALIAS /R](#)^[154] command, you can remove the

aliases by using the UNALIAS /R command with the same file name:

```
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.

Options:

- /Q** 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 the list of aliases to remove from a file. The file format should be the same format as that used by the [ALIAS /R](#) ^[154] command. You can use multiple files with one UNALIAS /R command by placing the names on the command line, separated by spaces:

```
unalias /r alias1.lst alias2.lst
```

UNALIAS /R will read from [stdin](#) ^[535] if no filename is present and input is redirected.

3.13.14 UNFUNCTION

Purpose: Remove user-defined functions from the function list.

Format: UNFUNCTION [/Q /R *file...* (*function ...*)] *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\(quiet\)](#) ^[357]

[/R\(read file\)](#) ^[357]

See also: [FUNCTION](#) ^[242] and [ESET](#) ^[222].

Usage:

TCC 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.

To remove the function DDIR:

```
unfunction ddir
```

To remove all the functions:

```
unfunction *
```

To remove all the functions that begin with "DD":

```
unfunction dd*
```

You can delete all matching functions except for those specified by enclosing the exceptions in parentheses. For example, to remove all functions beginning with "f" except for *func1* and *func2*:

```
unfunction (func1 func2) f*
```

If you keep functions in a file that can be loaded with the [FUNCTION /R](#)^[242] command, you can remove the functions by using the UNFUNCTION /R command with the same file name:

```
unfunction /r function.lst
```

This is much faster than removing each function individually in a batch file, and can be more selective than using UNFUNCTION *.

Options:

- /Q** 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 the list of functions to remove from a file. The file format should be the same format as that used by the [FUNCTION /R](#)^[242] command. You can use multiple files with one UNFUNCTION /R command by placing the names on the command line, separated by spaces:

```
unfunction /r function1.lst function2.lst
```

UNFUNCTION /R will read from [stdin](#)^[535] if no filename is present and input is redirected.

3.13.14 UNSET

Purpose: Remove variables from the environment or the registry.

Format: UNSET [/D /E /Q /S /U /V /R file... (name ...)] name [name...]]
or
UNSET *

name One or more variables to removed (wildcards accepted except for registry variables).

file One or more files from which to read variables to be removed.

/D(efault) ^[358]	/S(ystem) ^[358]
/E(nvironment) ^[358]	/U(ser) ^[358]
/Q(quiet) ^[358]	/V(olatile) ^[358]
/R(ead) ^[358]	

See also: [ESET](#)^[222] and [SET](#)^[319].

Usage:

UNSET removes one or more variables from the environment or from the Windows Registry.

For example, to remove the environment variable **CMDLINE**:

```
unset cmdline
```

If you use the command **UNSET ***, all of the environment variables will be deleted:

```
unset *
```

You can delete all matching variables except for those specified by enclosing the exceptions in parentheses. For example, to remove all variables beginning with "v" except for *var1* and *var2*:

```
unset (var1 var2) v*
```

UNSET can be used in a batch file, in conjunction with the [SETLOCAL](#)^[326] and [ENDLOCAL](#)^[220] 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](#)^[319] command and the general discussion of the [environment](#)^[365].

Note: You cannot use UNSET with [GOSUB variables](#)^[247].

Use caution when removing environment variables, and especially when using **UNSET ***. Many programs will not work properly without certain environment variables; for example, **TCC** depends on **PATH**.

Registry Variables: Default, System, User, and Volatile registry variables can be manipulated with the UNSET command's [/D](#)^[358], [/S](#)^[358], [/U](#)^[358] and [/V](#)^[358] switches, respectively. To remove the variable from both the registry and from the local environment, use both the [/E](#)^[358] switch and the registry variable selection switch together. (You cannot use wildcards for the variable name.) For example, to remove the volatile variable **myvar** from both the registry and the local environment, use:

```
unset /v /e myvar
```

Use caution when directly removing registry variables as they may be essential to various Windows processes and applications.

Options:

- /D** Delete a default variable from the registry (HKCU\DEFAULT\Environment).
- /E** When used together with one of [/D](#)^[322], [/S](#)^[322], [/U](#)^[323], or [/V](#)^[323], unsets both the registry variable and the local environment variable.
- /Q** Prevents UNSET from displaying an error message if one or more of the variables do 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 environment variables to be UNSET from a file. This is much faster than using multiple UNSET commands in a batch file, and can be more selective than **UNSET ***. The file format may be the same as that used by the [SET](#)^[319] /R command (see [SET](#)^[319] for more details), or it could just be one variable per line, wildcards not processed.

UNSET /R will read from [STDIN](#)^[535] if no filename is present and input is redirected.
- /S** Delete a **system** variable from the registry (HKLM\System\CurrentControlSet\Control\Session Manager\Environment).
- /U** Delete a **user** variable from the registry (HKCU\Environment).
- /V** Delete a **volatile** variable from the registry (HKCU\Volatile Environment)

3.13.14 USBMONITOR

Purpose: Monitor USB device connection and disconnection

Format: USBMONITOR [/C [name]]

USBMONITOR name CONNECTED | DISCONNECTED n command

name Device name
n Number of repetitions (or **FOREVER**)
command Command to execute when condition is triggered

[/C\(lear\)](#) ^[359]

Usage:

The USB device name can include wildcards. You can use either the device ID or the "friendly" name for the device.

The command line will be parsed and expanded before USBMONITOR is executed, so if you want to pass redirection characters or variables to **command** you will need to protect them (by enclosing in single back quotes, doubling the %'s, or using command grouping).

If you don't enter any arguments, USBMONITOR will display the USB devices it is currently monitoring.

The monitoring runs asynchronously in a separate thread. When the condition is triggered, the command will be executed immediately. This may cause problems if you try to write to the display or access files while the main **TCC** thread is also performing I/O. You may need to use [START](#) ^[331] or [DETACH](#) ^[197] in **command** to avoid conflicts.

USBMONITOR creates two environment variables when a device is connected or disconnected that can be queried by **command**. The variables are deleted after **command** is executed.

_usbdeviceid The device ID
_usbname The "friendly" name of the device
_usbcount The number of times the command has been triggered

Options:

/C If **name** is specified, remove the monitor for that USB device. Otherwise, remove all USB monitors.

3.13.14 VER

Purpose: Display the **TCC** and operating system versions.

Format: VER [/R]

[/R\(evision\)](#) ^[360]

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 version numbers for both **TCC** and Windows:

```
[c:\] ver
Take Command 9.0.80   Windows Vista [Version 6.0.6000]
```

Option:

/R Display the **TCC** and operating system internal revision level (if any), plus your registered name.

3.13.14\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 under Windows. **TCC** supports VERIFY as a "do-nothing" command, for compatibility with CMD.EXE. This avoids **unknown command** errors in old batch files which use the VERIFY command. You can set verification for file copying with the [COPY /V](#) option.

If used without any parameters, VERIFY will display the state of the verify flag:

```
[c:\] verify
VERIFY is ON
```

3.13.14\VOL

Purpose: Display disk volume label(s).

Format: VOL [d:] ...

d: The drive or drives to search.

Usage:

Each disk may have a volume label, created when the disk is formatted or with the external LABEL command. Also, every 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](#) ⁴⁴⁶.

3.13.14\VSCRPUT

Purpose: Display text vertically in the specified color.

Format: VSCRPUT row col [BR!ght] fg ON [BR!ght] bg text

row	Starting row
col	Starting column
fg	Foreground text color
bg	Background text color
text	The text to display

See also: [SCRPUT](#)^[311].

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](#)^[518] 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 window 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.

The maximum **row** value is determined by the current height of the **TCC** window. The maximum **column** value is determined by the current virtual screen width (see [Resizing the Take Command Window](#)^[68] 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
```

3.13.14 WHICH

Purpose: Display the command type and what it would execute.

Format: WHICH [/A] command [command ...]

command One or more commands or files.

[/A\(II\)](#)^[362]

Usage:

WHICH displays information about internal and external commands, [Aliases](#)^[128] (including keystroke aliases), and files. When a file matches an applicable [Executable Extension](#)^[91] or [Windows File Association](#)^[506], 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\Microsoft
Office\OFFICE11\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 [Plugin](#)^[293] commands, [REXX](#)^[142] files, [EXTPROC](#)^[143] files, and associated files.

Note: WHICH does not support wildcard specifications unless you use the [/A](#)^[362] option. Parameters must be actual commands or actual 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 double quotes. A file specified without an explicit path must be on the current [PATH](#)^[368].

See [Executable Files and File Searches](#)^[504] for details on the order in which various locations are searched.

See also: [@SEARCH](#)^[458], [ASSOC](#)^[162], [FTYPE](#)^[241].

Option:

/A Display all matching commands. (Normally WHICH only displays the first match.)

3.13.14!WINDOW

Purpose: Minimize or maximize the current window, restore the default window size, or change the window title.

Format: WINDOW [MAX | MIN | RESTORE | HIDE | DISABLE | ENABLE | TRAY | TOPMOST | NOTOPMOST | TOP | BOTTOM | /POS=left,top,width,height | /SIZE=rows,columns | /TRANS=n | "newtitle"]

newtitle	A new title for the window
height	New height of window
width	New width of window
left	New position of the left border of window
top	New position of the top border window
rows	New height of window
columns	New width of window

[/POS\(ition\)](#)^[363]

[/SIZE \(of screen buffer\)](#)^[362]

See also: [ACTIVATE](#)^[152] and [TITLE](#)^[349].

Usage:

WINDOW is used to control the appearance and title of the current (**TCC**) 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).

If you are running in a **Take Command** tab window, the MAX, MIN, RESTORE, HIDE, TRAY, and /TRANS options will be sent to the main **Take Command** window, not the **TCC** window.

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:

Option	TC C	Take Command	Description
/POS= <i>left,top,width,height</i>	x	x	Set the window position and size on the desktop. The values are specified in pixels. Left and top refer to the position of the top left corner of the window relative to the top left corner (0,0) of the screen. The width and height values determine the window size. The window may be so sized and positioned that parts of it are beyond the physical area of the display. The / before the keyword is optional.
/TRANS= <i>n</i>		x	Set the transparency level of the Take Command window. n is an value from 0 (invisible) to 255 (opaque).
<i>newtitle</i>	x	x	Changes the window title. The title text must be enclosed in double quotes. (The quotes will not appear as part of the actual title as displayed.) Setting the title inside a batch file will only change the window title while the batch file is executing.
MAX	x	x	Expands the window to its maximum size.
MIN	x	x	Reduces the window to an icon.
RESTORE	x	x	Returns the window to its default size and location.
HIDE	x	x	Makes the window invisible. Use RESTORE to make the window visible.
TRAY	x	x	Moves the window to the taskbar tray.
/SIZE= <i>rows,columns</i>	x		Specify the TCC screen buffer size. Due to the design of Windows console sessions, you cannot use /SIZE to reduce the size of the screen buffer; it can only be increased. Does not affect window size.
DISABLE	x	x	Disable window input (mouse and keyboard)
ENABLE	x	x	Enable window input
TOPMOST		x	Keeps the Take Command window on top of all other windows until it closes, or NOTOPMOST is used.
NOTOPMOST		x	Allows other windows to overlay the Take Command window (this is the normal state for most windows).
TOP		x	Moves the Take Command window to the top of the window order, above all other non- TOPMOST windows.
BOTTOM		x	Moves the Take Command window to the bottom of the window order.

3.13.15 WMIQUERY

Purpose: Query the Windows Management Interface

Format: WMIQUERY [/A /B /C /H] namespace "query string" [index]

[/A\(All instances\)](#) ^[364] [/C\(classes\)](#) ^[364]
[/B\(blank\)](#) ^[364] [/H\(header\)](#) ^[364]

namespace	The namespace to query
"query string"	WQL query string
index	Class instance

Usage:

You can use a single period . for **namespace** to default to **root\cimv2**.

For more details on what is available, see the WMI and WQL documentation on MSDN (msdn.microsoft.com), and download the "WMI Code Creator" from Microsoft and browse the available namespaces, classes, and properties.

For example, to query the **name** property from the **Win32_Processor** class:

```
wmiquery root\cimv2 "SELECT name FROM Win32_Processor"
```

To query available classes:

```
wmiquery /A root "select name from __namespace"
```

Options:

- /A** Display all class instances starting at "index".
- /B** Separate class instances with a blank line.
- /C** Display all the matching class names for the specified namespace. **"query string"** is the filter to apply to the returned values; it can contain wildcards. For example:

```
wmiquery /c . "win32_q*"
```
- /H** Display a header for class instances.

3.13.15 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](#) ^[344], [piping](#) ^[101] and [redirection](#) ^[98].

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*:

```
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:

```
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](#)^[107] for more information on pipes.

3.14 Variables & Functions

The environment is a collection of information about your system that every program receives. Each entry in the environment consists of a variable name and a string value.

Usage

You can automatically substitute the text for the variable name in any command. To create the substitution, include a percent sign % and the 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](#)^[319], [ESET](#)^[222], and [UNSET](#)^[357] commands.

A few environment variables have special meanings for **TCC** (they are listed in [System Variables](#)^[367]).

TCC also supports two special types of variables:

- ▶ [Internal variables](#)^[372] are similar to environment variables, but are interpreted internally by **TCC**, and are not visible in the environment. They provide information about your system for use in batch files and aliases. Some of them provide access to information that may change even during the execution of a single command or batch file.
- ▶ [Variable functions](#)^[395] are referenced like environment variables, but perform additional actions like file handling, string manipulation and arithmetic calculations. In addition to the variable functions that are internal to **TCC**, you can use the [FUNCTION](#)^[242] command to create your own. These latter ones are referred to as user defined functions or UDFs.

Note: **TCC** inherits its initial environment from the process which started it. That process might be Explorer or another existing Windows process which launched the current **TCC** session. Note that if the starting process's environment is changed (through registry modifications, for example) while **TCC** is already running, those changes will not be automatically reflected in **TCC**'s current environment. See the [SET](#)^[319] command for details.

You use the [SET](#)^[319] command to create a new environment variable. [SET](#)^[319] can also modify or delete a single environment variable, or display the value of one or more environment variables. [ESET](#)^[222] allows you to edit an environment variable. [UNSET](#)^[357] deletes environment variables. For example, you can create a variable named **BACKUP** like this:

```
set BACKUP=*.bak;*.bk
```

If you then type:

```
del %BACKUP
```

it is equivalent to having type the command:

```
del *.bak;*.bk
```

The environment 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, **TCC** uses the environment to keep track of the default directory on each drive. Windows only tracks the default directory of the current drive; **TCC** overcomes this limitation by saving the default directory for each drive in the environment, using hidden variable names. 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](#)^[319] command.

The trailing percent sign that was traditionally required for environment variable names is not usually required by **TCC**, which accept 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 append 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:

```
set ONE=abcd
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 [Parameter Quoting](#) ^[134] for details.

Environment variables may contain alias names. **TCC** 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:

```
alias d2 dir /2
set cmd=d2
%cmd *.txt
```

For compatibility with some peculiar syntax introduced in recent **CMD.EXE** versions, **TCC** supports:

%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](#) ^[404].

3.14.1 System Variables

The variables below have special meaning for **TCC**:

CDPATH ^[367]	Directory navigation search list
CMDLINE ^[368]	Command line after full expansion
COLORDIR ^[368]	Directory colorization specification
COMSPEC ^[368]	Command processor specification
FILECOMPLETION ^[368]	File completion control variable
HISTORYEXCLUSION ^[368]	List of commands excluded from the command history ^[106]
PATH ^[368]	Executable program location search list
PATHEXT ^[369]	Ordered search list of extensions of executable programs
PROMPT ^[369]	Command prompt format specification
RECYCLEEXCLUDE ^[369]	List of files excluded from the recycle bin
TCMD ^[370]	Take Command's pathname
TCMDVER ^[370]	Take Command's version number
TEMP ^[370]	Directory for temporary files
TITLEPROMPT ^[370]	Command processor window title bar specification
TMP ^[370]	Directory for temporary files
TREEEXCLUDE ^[370]	List of directories excluded from JPSTREE.IDX
VARIABLEEXCLUDE ^[370]	Variables to exclude from SET list

3.14.1.1 CDPATH

CDPATH specifies where to search for directories specified in [CD](#) ^[177], [CDD](#) ^[178], and [PUSHD](#) ^[299] commands and in [automatic directory changes](#) ^[76]. See the [CDPATH feature](#) ^[72] for details.

This feature is maintained for backwards compatibility, but has largely been replaced by [Extended Directory Searches](#) ^[73].

3.14.1.2 CMDLINE

CMDLINE is set by **TCC** to the fully expanded text of the currently executing command line just before invoking any external command (.COM, .EXE, .BTM, .BAT or .CMD), unless the command line is prefaced with @ to prevent echoing, in which case **CMDLINE** will be removed.

3.14.1.3 COLORDIR

COLORDIR controls directory display colors used by [DIR](#)^[198]. See [Color-Coded Directories](#)^[203] for a complete description of the format of this variable.

3.14.1.4 COMSPEC

Many programs expect the value of **COMSPEC** to contain the full path and name of the current character-mode command processor, e.g.

```
c:\program files\jpsoft\tcmd9\tcc.exe
```

TCC automatically sets COMSPEC to point to TCC.EXE on startup. If you need to run a program from **TCC** which utilizes COMSPEC to locate a command processor to process commands or batch files that are not compatible with **TCC**, you may set COMSPEC to the command processor your program expects before you start it.

3.14.1.5 FILECOMPLETION

FILECOMPLETION specifies the files made available during filename completion for selected commands. See [Customizing Filename Completion](#)^[115] for a complete description of the format.

3.14.1.6 HISTORYEXCLUDE

HistoryExclude specifies which commands should be excluded from the [History List](#)^[106]. The syntax is:

```
HistoryExclude=cmd1[;cmd2[;cmd3[;...]]]
```

For example, to exclude the [DEL](#)^[190] and [FREE](#)^[241] commands, the **Notepad** program and the user-defined alias **MYDIR**:

```
HistoryExclude=del;free;c:\windows\system32\notepad.exe;mydir
```

See also: [HISTORY](#)^[251].

3.14.1.7 PATH

The **PATH** variable specifies the list of directories that **TCC** will search for executable files that aren't in the current directory. **PATH** is used by some application programs to find their own files. See the [PATH](#)^[286] command for a full description of this variable, which can also be changed or modified with [SET](#)^[319] and [ESET](#)^[222].

Note: We strongly recommend that you always leave at least your **WINDOWS** and **SYSTEM32** directories in the **PATH**. The directory where **TCC** resides need not be included in **PATH**.

3.14.1.8 PATHEXT

PATHEXT is expected to contain a list of extensions (including a leading period .), separated by semicolons. For example, to replicate the default extension list used by **TCC**:

```
set pathtext=.pif;.com;.exe;.btm;.bat;.cmd;.rex;.rexx
```

If you use a command in a batch file or at the command prompt and all of the following are true:

- the [PathExt](#)^[47] configuration option is set
- the command is not an alias
- the command is not an internal command
- the command is not a filename with an explicit extension (thus neither an executable extension nor a Windows file association is available)

then **TCC** will search each directory listed in [PATH](#)^[368] in turn for a file with its name matching the command and its extension matching one of the extensions in PATHEXT. The 1st directory in [PATH](#)^[368] is searched first, then the 2nd, looking in each for each of the extensions in PATHEXT in the order listed.

Enabling PATHEXT affects only the standard path search. It does not affect searches for files with explicit extensions, which must have either a standard extension (see list above), or one which is either an [executable extension](#)^[91] or a Windows association.

Caution: If you set the [PathExt](#)^[47] configuration option, and fail to set the PATHEXT variable, path searches without an explicit extension will fail as there will be no extensions for which to search! (Windows XP does define a default value for the PATHEXT variable.)

3.14.1.9 PROMPT

PROMPT defines the command line prompt. It can be set or changed with the [PROMPT](#)^[297], [SET](#)^[319] and [ESET](#)^[222] commands. See the [PROMPT](#)^[297] command for details.

See also: [TITLEPROMPT](#)^[370].

3.14.1.10 RECYCLEEXCLUDE

RECYCLEEXCLUDE specifies files to be excluded from the Recycle Bin.

The syntax is:

```
RecycleExclude=file1[;file2...]
```

***file1*, *file2*, ...** : file specifications, may include wildcards

For example, to exclude *.lib, *.obj, and *.bak files:

```
RecycleExclude=*.lib;*.obj;*.bak
```

See also: [DEL / ERASE](#)^[190] command and the [Delete to Recycle Bin](#)^[47] configuration option.

3.14.1.11 TEMP

TEMP specifies the directory where **TCC** should store temporary files, unless the [TMP](#) ^[370] variable exists. Many other programs also use TEMP to locate where they should place their temporary files.

3.14.1.12 TCMD

TCMD is the full pathname of the **Take Command** executable. It is only available to the applications running in **Take Command** tab windows.

3.14.1.13 TCMDVER

TCMDVER is the current major version, minor version, and build number of **Take Command**. (For example, "9.00.80".) It is only available to the applications running in **Take Command** tab windows.

3.14.1.14 TITLEPROMPT

TITLEPROMPT can be used to specify the contents of **TCC**'s window title. 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](#) ^[297], and all internal variables and functions can be used to generate it.

If you have specified a title for a startup tab in **Take Command**, it will override the **TITLEPROMPT** value.

See also: [ACTIVATE](#) ^[152], [PROMPT](#) ^[297], [TITLE](#) ^[349] and [WINDOW](#) ^[362].

3.14.1.15 TMP

If **TMP** is defined, it specifies the directory where **TCC** should store temporary files (overriding [TEMP](#) ^[370]). Some other programs also use TMP to define where they should place their temporary files.

3.14.1.16 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:\
```

Setting **TreeExclude** to the base directory of the target of a directory tree copy can speed up the copying considerably.

See also: [Extended Directory Searches](#) ^[73] and [CDD](#) ^[178].

3.14.1.17 VARIABLEEXCLUDE

VariableExclude specifies which environment variables should be excluded from the SET list. The syntax is:

```
VariableExclude=var1[;var2[;var3[;...]]]
```

For example, to exclude the SESSIONNAME, TMP, USERDOMAIN, and USERNAME variables:

```
SET VariableExclude=sessionname;tmp;userdomain;username
```

See also: [SET](#)^[319].

3.14.2 CMD.EXE Compatibility Variables

CMD.EXE has some built-in variables (i.e., which are treated as environment variables but which do not exist in the environment):

CD - the current directory (see also [_CWD](#)^[384]).

CMDCMDLINE - the command line that started the command processor.

CMDEXTVERSION - the command extensions internal version number.

DATE - the current system date (see also [_DATE](#)^[385]).

RANDOM - a random number between 0 and 32767 (see also [@RANDOM](#)^[453]).

TIME - the current system time (see also [_TIME](#)^[393]).

TCC supports all of these built-in variables. (In **TCC**, **CMDEXTVERSION** will always return **2**.)

The variables below are used by some Microsoft command processors, but are ignored by **TCC**. To see their usage by Microsoft and the alternate methods to achieve the same purpose in **TCC**, review:

COPYCMD ^[371]	CMD.EXE default options for COPY ^[182] command
DIRCMD ^[371]	CMD.EXE default options for DIR ^[198] command

3.14.2.1 COPYCMD variable

The **COPYCMD** variable is used by some versions of CMD.EXE to hold default options for the [COPY](#)^[182] command. **TCC** does not directly support this variable, i.e, its value has no affect on internal commands. In general, it is more efficient to define several aliases, each including a different combination of options. For example, if you want the COPY command to default to prompting you before overwriting an existing file, you could use this alias:

```
alias COPY=`*copy /r`
```

If you wish to use or create a COPYCMD variable for compatibility with CMD.EXE, you can define an alias to append the contents of that variable to the COPY command:

```
alias COPY=`*copy %copycmd`
```

Now each time the COPY alias is executed, the current value of COPYCMD will modify the execution of the COPY command.

3.14.2.2 DIRCMD variable

The **DIRCMD** variable is used by some versions of CMD.EXE to hold default options for the [DIR](#)^[198] command. **TCC** does not directly support this variable, i.e, its value has no affect on internal commands. In general, it is more efficient to define several aliases, each including a different combination of options. 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:

```
alias DIR=`*dir /2 /p /v`I
```


If you wish to use or create a DIRCMD variable for compatibility with CMD.EXE, you can define the alias to append the contents of that variable to the DIR command:

```
alias DIR=`*dir %dircmd`
```

Now each time the DIR alias is executed, the current value of DIRCMD will modify the execution of the DIR command.

3.14.2.3 String substitution

For compatibility with some peculiar syntax introduced in recent **CMD.EXE** versions, **TCC** supports:

<code>%var:string1=string2%</code>	Substitutes the second string for all instances of the first string in the variable.
<code>%var:~x[,y]%</code>	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](#)^[404].

3.14.3 Internal Variables

Internal variables are special variables built into **TCC** 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 **TCC**, and cannot be changed with the [SET](#)^[319], [UNSET](#)^[357], [ESET](#)^[222] or any other command. The DEFINED status test will always fail, too. 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:

```
echo %variablename
```

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](#)^[382] will not return accurate results if your operating system and Advanced Power Management drivers do not provide correct information on battery status to **TCC**.

For a list of internal variables organized by general categories of use, see [Internal Variables by Category](#)^[376].

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 **TCC**.

Store the current date and time in a file, then save the output of a DIR command in the same file:

```
echo Directory as of %_date %_time > dirsave
dir >> dirsave
```

Use the **IFF** command to check whether there are enough resources free before running an application:

```
iff %_GDIFREE lt 40 then
    echo Not enough GDI resources!
    quit
else
    d:\mydir\myapp
endiff
```

Call another batch file if today is Monday:

```
if "%_DOW" == "Mon" call c:\cleanup\weekly.bat
```

3.14.3.1 Variables by Name

+	Substitutes the TCC command separator
=	Substitutes the TCC escape character
!	Last argument of the previous command
?	Exit code, last external program
4ver	TCC version
_?	Exit code, last internal command
_acstatus	AC line status
_admin	1 if an administrator, else 0
_afswcell	OpenAFS workstation cell
_alt	Alt key depressed: 1, else 0
_ansi	ANSI X3.64 status
_batch	Batch nesting level
_batchline	Line number in current batch file.
_batchname	Full path and filename of current batch file.
_batchtype	Type of the current batch file
_battery	Battery status
_batterylife	Remaining battery life, seconds
_batterypercent	Remaining battery life, %
_bdebugger	Batch debugger active: 1, else 0
_bg	Background color at cursor position
_boot	Boot drive letter, without a colon
_build	Build number
_capslock	CapsLock on: 1, else 0
_cdroms	List of the CD-ROM drives
_childpid	Process ID of most recent child process
_ci	Current text cursor shape in insert mode

cmdline	Current command line
cmdproc	Command processor name
cmdsproc	Full pathname of command processor
co	Current text cursor shape in overstrike mode
codepage	Current code page number
column	Current cursor column
columns	Virtual screen width
consolepids	Process IDs attached to this console
country	Current country code
cpu	CPU type
cpuusage	CPU time usage (percent)
ctrl	Ctrl key depressed: 1, else 0
cwd	Current drive and directory
cwds	Current drive and directory with trailing \
cwp	Current directory
_cwps	Current directory with trailing \

_date	Current date
datetime	Current date and time, yyyyMMddhhmmss
day	Current day of the month
detachpid	Process ID of most recent detached process
disk	Current drive
dname	Name of the description file.
dos	Operating system type
dosver	Operating system version
dow	Current day of the week, English, short
dowf	Current day of the week, English, full
dowi	Current day of the week as an integer
doy	Current day of the year
drives	List of the existing drives
dst	Daylight savings time: 1, else 0
dvds	List of the DVD drives

_echo	Echo turned on: 1, else 0
_editmode	0 if in overstrike mode, 1 if in insert mode
errorlevel	Exit code, last external program
execstr	@EXECSTR return code
_exit	TCC exit return code
_expansion	SETDOS /X value

_fg	Foreground color at cursor position
_ftperror	Last FTP error code

hdrives	List of the fixed drives
_hlogfile	Current history log file name
host	Host name of local computer.
hour	Current hour
_hwprofile	Windows hardware profile if defined

_idleticks	Milliseconds since the last user input
idow	Current day of the week, local language, short
idowf	Current day of the week, local language, full
_iftp	IFTP session active: 1, else 0

iftps ^[388]	IFTPS session active: 1, else 0
imonth ^[388]	Current month name, local language, short
imonthf ^[388]	Current month name, local language, full
ininame ^[388]	Full pathname of the current INI file
ip ^[388]	IP address(es) of local computer.
isodate ^[388]	Current date in ISO 9601 format
_kbhit ^[388]	Keyboard input character is waiting: 1, else 0
_lalt ^[388]	left Alt key depressed: 1, else 0
lastdisk ^[389]	Last valid drive
lctrl ^[389]	Left Ctrl key depressed: 1, else 0
logfile ^[389]	Current log file name
_lshift ^[389]	Left Shift key depressed: 1, else 0
_minute ^[389]	Current minute
monitors ^[389]	Number of monitors
month ^[389]	Current month of the year as integer
monthf ^[389]	Current month of the year, English, full
_numlock ^[389]	NumLock on: 1, else 0
openafs ^[390]	OpenAFS installed: 1, otherwise 0
osbuild ^[390]	Windows build number
parent ^[390]	Name of the parent process
pid ^[390]	TCC process ID (numeric)
pipe ^[390]	Current process is running in a pipe: 1, else 0
ppid ^[390]	Process ID of parent process
_ralt ^[390]	Right Alt key depressed: 1, else 0
rctrl ^[390]	Right Ctrl key depressed: 1, else 0
ready ^[390]	List of accessible drives
registered ^[390]	Registered user name
row ^[390]	Current cursor row
rows ^[391]	Screen height
_rshift ^[391]	Right Shift key depressed: 1, else 0
scrolllock ^[391]	ScrollLock on: 1, else 0
second ^[391]	Current second
selected ^[391]	First line of highlighted text
shell ^[391]	Shell level
shells ^[391]	Shell level (old style)
shift ^[391]	Shift key depressed: 1, else 0
shortcut ^[392]	Pathname of shortcut that started this process
shralias ^[392]	SHRALIAS is loaded: 1, else 0
startpath ^[392]	Startup directory of current shell.
startpid ^[392]	Process ID of most recent STARTed process.
stdin ^[392]	STDIN redirected: 1, else 0
stdout ^[392]	STDOUT redirected: 1, else 0
stderr ^[392]	STDERR redirected: 1, else 0
stzn ^[392]	Name of time zone for standard time
stzo ^[392]	Offset in minutes from UTC for standard time

<u>syserr</u> ^[392]	Latest Windows error code
<u>tcfilter</u> ^[392]	Current filter in List view window
<u>tcfolder</u> ^[393]	Selected folder in Folders window
<u>tctab</u> ^[393]	Running inside Take Command: 1; else 0
<u>time</u> ^[393]	Current time
<u>transient</u> ^[393]	Current process is a transient shell: 1, else 0
<u>tzn</u> ^[393]	Name of current time zone
<u>tzo</u> ^[393]	Offset in minutes from UTC for current time zone
<u>unicode</u> ^[393]	Shell uses unicode for redirected output: 1, else 0
<u>utctime</u> ^[394]	Current UTC time
<u>utcdate</u> ^[393]	Current UTC date
<u>utcdatetime</u> ^[393]	Current UTC date and time
<u>utchour</u> ^[393]	Current UTC hour
<u>utcisodate</u> ^[393]	Current UTC date in ISO format
<u>utcisominute</u> ^[393]	Current UTC minute
<u>utcisosecond</u> ^[393]	Current UTC second
<u>virtualpc</u> ^[393]	Running inside VirtualPC: 1; else 0
<u>vmware</u> ^[394]	Running inside VMWare: 1; else 0
<u>vxpixels</u> ^[394]	Virtual screen horizontal size
<u>vypixels</u> ^[394]	Virtual screen vertical size
<u>windir</u> ^[394]	Windows directory pathname
<u>wingwindow</u> ^[394]	Title of foreground window.
<u>winname</u> ^[394]	Name of local computer
<u>winsysdir</u> ^[394]	Windows system directory pathname
<u>winticks</u> ^[394]	Milliseconds since Windows was started
<u>wintitle</u> ^[394]	Current window title
<u>winuser</u> ^[394]	Name of current user.
<u>winver</u> ^[394]	Windows version number
<u>wow64</u> ^[394]	Running inside WOW64: 1; else 0
<u>xpixels</u> ^[394]	Physical screen horizontal size in pixels
<u>year</u> ^[394]	Current year
<u>ypixels</u> ^[395]	Physical screen vertical size in pixels

3.14.3.2 Variables by Category

- ▶ [TCC status](#) ^[378]
- ▶ [Compatibility](#) ^[380]
- ▶ [Dates and times](#) ^[379]
- ▶ [Drives and directories](#) ^[379]
- ▶ [Error codes](#) ^[380]
- ▶ [Hardware status](#) ^[377]
- ▶ [Operating system and software status](#) ^[377]
- ▶ [Screen, color, and cursor](#) ^[378]

The list below gives a one-line description of all [Internal Variables](#)^[372] 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](#)^[25] command for details

Hardware status

acstatus ^[381]	AC line status
alt ^[381]	Alt key depressed
battery ^[382]	Battery status
batterylife ^[382]	Remaining battery life, seconds
batterypercent ^[382]	Remaining battery life, %
capslock ^[383]	CapsLock on: 1, otherwise 0
cpu ^[384]	CPU type
cpuusage ^[384]	CPU time usage (percent)
ctrl ^[384]	Ctrl key depressed: 1, otherwise 0
kbhit ^[385]	A keyboard input character is waiting: 1, otherwise 0
lalt ^[388]	left Alt key depressed: 1, otherwise 0
lctrl ^[389]	left Ctrl key depressed: 1, otherwise 0
lshift ^[389]	left Shift key depressed: 1, otherwise 0
numlock ^[389]	NumLock on: 1, otherwise 0
ralt ^[390]	right Alt key depressed: 1, otherwise 0
rctrl ^[390]	right Ctrl key depressed: 1, otherwise 0
rshift ^[391]	right Shift key depressed: 1, otherwise 0
scrolllock ^[391]	ScrollLock on: 1, otherwise 0
shift ^[391]	Shift key depressed: 1, otherwise 0

Operating system and software status

! ^[380]	Last argument of previous command
admin ^[381]	1 if administrator; else 0
ansi ^[382]	ANSI X3.64 status
boot ^[383]	Boot drive letter, without a colon
codepage ^[383]	Current code page number
country ^[384]	Current country code
dos ^[385]	Operating system type
dosver ^[386]	Operating system version
host ^[387]	Host name of local computer.
hwprofile ^[388]	Windows hardware profile if defined
idleticks ^[388]	Milliseconds since last user input
ip ^[388]	IP address(es) of local computer.
osbuild ^[390]	Windows build number
tctab ^[393]	Running inside Take Command: 1; else 0
virtualpc ^[394]	Running inside VirtualPC: 1; else 0
vmware ^[394]	Running inside VMWare: 1; else 0
windir ^[394]	Windows directory pathname
winfgwindow ^[394]	Title of foreground window.
winname ^[394]	Name of local computer
winsysdir ^[394]	Windows system directory pathname
winticks ^[394]	Milliseconds since Windows was started
wintitle ^[394]	Current window title
winuser ^[394]	Name of current user.
winver ^[394]	Windows version number
wow64 ^[394]	Running in Windows 64: 1; else 0

TCC status

4ver ³⁸¹	TCC version
batch ³⁸²	Batch nesting level
batchline ³⁸²	Line number in current batch file.
batchname ³⁸²	Full path and filename of current batch file.
batchtype ³⁸²	Type of the current batch file
bdebugger ³⁸²	Batch debugger active: 1, otherwise 0
build ³⁸³	Build number
childpid ³⁸³	Process ID of most recent child process
cmdline ³⁸³	Current command line
cmdproc ³⁸³	Command processor name
cmdsproc ³⁸³	Full pathname of command processor
detachpid ³⁸⁵	Process ID of most recent detached process
dname ³⁸⁵	Name of the description file.
echo ³⁸⁶	Echo status
editmode ³⁸⁶	Insert mode: 1; else 0
exit ³⁸⁶	TCC exit code
expansion ³⁸⁷	Current expansion mode (SETDOS /X)
hlogfile ³⁸⁷	Current history log file name
iftp ³⁸⁸	IFTP session active: 1, otherwise 0
iftps ³⁸⁸	IFTPS session active: 1, otherwise 0
iname ³⁸⁸	Full pathname of the current INI file
logfile ³⁸⁹	Current log file name
parent ³⁹⁰	Name of the parent process
pid ³⁹⁰	The TCC process ID (numeric)
pipe ³⁹⁰	Current process is running in a pipe: 1, otherwise 0
ppid ³⁹⁰	Process ID of parent process
registered ³⁹⁰	Registered user name
shell ³⁹¹	Shell level
shells ³⁹¹	Shell level (old style)
shortcut ³⁹²	Pathname of shortcut that started this process
shralias ³⁹²	SHRALIAS is loaded: 1, otherwise 0
startpath ³⁹²	Startup directory of current shell.
startpid ³⁹²	Process ID of most recent STARTed process.
stdin ³⁹²	STDIN redirected: 1, otherwise 0
stdout ³⁹²	STDOUT redirected: 1, otherwise 0
stderr ³⁹²	STDERR redirected: 1, otherwise 0
tcfilter ³⁹²	Current filter in List view window
tcfolder ³⁹³	Selected folder in Folders window
transient ³⁹³	Current process is a transient shell: 1, otherwise 0
_unicode ³⁹³	TC uses unicode for redirected output: 1, otherwise 0

Screen, color, and cursor

bg ³⁸²	Background color at cursor position
ci ³⁸³	Current text cursor shape in insert mode
co ³⁸³	Current text cursor shape in overstrike mode
column ³⁸⁴	Current cursor column
columns ³⁸⁴	Virtual screen width
fg ³⁸⁷	Foreground color at cursor position
monitor ³⁸⁹	Number of monitors
row ³⁹⁰	Current cursor row
rows ³⁹¹	Screen height
_selected ³⁹¹	(TC) First line of highlighted text

vxpixels <small>[394]</small>	Virtual screen horizontal size
vypixels <small>[394]</small>	Virtual screen vertical size
xpixels <small>[394]</small>	Physical screen horizontal size in pixels
ypixels <small>[395]</small>	Physical screen vertical size in pixels

Drives and directories

afswcell <small>[381]</small>	OpenAFS workstation cell
cdroms <small>[383]</small>	List of CD-ROM drives
cwd <small>[384]</small>	Current drive and directory
cwds <small>[385]</small>	Current drive and directory with trailing \
cwp <small>[385]</small>	Current directory
cwps <small>[385]</small>	Current directory with trailing \
disk <small>[385]</small>	Current drive
drives <small>[386]</small>	List of all available drives
dvds <small>[386]</small>	List of DVD drives
hdrives <small>[387]</small>	List of hard (fixed) drives
lastdisk <small>[389]</small>	Last valid drive
openafs <small>[390]</small>	OpenAFS service installed: 1, otherwise 0
ready <small>[390]</small>	List of ready (accessible) drives

Dates and times

date <small>[385]</small>	Current date
datetime <small>[385]</small>	Current date and time, yyyyMMddhhmmss
day <small>[385]</small>	Current day of the month
dow <small>[386]</small>	Current day of the week, English, short
dowf <small>[386]</small>	Current day of the week, English, full
dowi <small>[386]</small>	Current day of the week as an integer
doy <small>[386]</small>	Current day of the year
dst <small>[386]</small>	Daylight savings time: 1, else 0
hour <small>[387]</small>	Current hour
idow <small>[388]</small>	Current day of the week, local language, short
idowf <small>[388]</small>	Current day of the week, local language, full
imonth <small>[388]</small>	Current month name, local language, short
imonthf <small>[388]</small>	Current month name, local language, full
isodate <small>[388]</small>	Current date in ISO 9601 format
minute <small>[389]</small>	Current minute
month <small>[389]</small>	Current month of the year as integer
monthf <small>[389]</small>	Current month of the year, English, full
second <small>[391]</small>	Current second
stzn <small>[392]</small>	Name of time zone for standard time
stzo <small>[392]</small>	Offset in minutes from UTC for standard time
time <small>[393]</small>	Current time
tzn <small>[393]</small>	Name of current time zone
tzo <small>[393]</small>	Offset in minutes from UTC for current time zone
utctime <small>[394]</small>	Current UTC time
utcdate <small>[393]</small>	Current UTC date
utcdatetime <small>[393]</small>	Current UTC date and time
utcheour <small>[393]</small>	Current UTC hour
utcisodate <small>[393]</small>	Current UTC date in ISO format
utcminute <small>[393]</small>	Current UTC minute
utcsecond <small>[393]</small>	Current UTC second
year <small>[394]</small>	Current year

Error codes

? ^[380]	Exit code, last external program
? ^[380]	Exit code, last internal command
errorlevel ^[395]	Exit code, last external program
execstr ^[386]	Last @EXECSTR return code
ftperror ^[387]	Last FTP error code
syserr ^[392]	Latest Windows error code

Compatibility

= ^[381]	Substitutes the TCC escape character
+ ^[381]	Substitutes command separator

3.14.3.3 ! (Variable)

! returns the last argument of the previous command. The command is retrieved from the history list, so this will not work in a batch file -- it's intended for aliases and command line work.

3.14.3.4 ? variable

If an **external** command (i.e., a program) has an **exit code**, its value is stored in the ? variable when the program terminates. Additionally, some **internal** commands, e.g., [DIR](#)^[198] - to emulate Microsoft's **CMD.EXE** - also set this variable to the same value they set the variable [_?](#)^[380], an action 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, not that of a subsequent internal or external command, it is best to save the value of ? in another variable immediately on completion of the external command of interest, and use that variable instead. We also strongly recommend that for internal commands you query the [_?](#)^[380] variable instead.

Not all programs return an exit code. If a program does not explicitly return an exit code, the value of %? is undefined.

Alternate name: [ERRORLEVEL](#)^[395].

See also: [_?](#)^[380]

3.14.3.5 _? variable

_? 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, including the one used to save it.

Result codes:

- 0** command was successful
- 1** a usage error occurred
- 2** another **Take Command** 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](#)^[307] command.

Note that in imitation of CMD.EXE some internal commands, e.g., [DIR](#), also set the variables [?](#)^[380] and [ERRORLEVEL](#)^[395] to the same value they set this variable. However, you are strongly urged to use this variable.

See also: [?](#)^[380]

3.14.3.6 = pseudovariable

= is the current [Escape character](#)^[51]. Use this pseudovariable, instead of the actual escape character, if you want your batch files and aliases to work in other users' environment regardless of how the escape character is defined.

3.14.3.7 + pseudovariable

+ is the current [command separator](#)^[51]. Use this pseudovariable, instead of the actual command separator, if you want your batch files and aliases to work 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 **TCC**) 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
```

3.14.3.8 _4VER

_4VER returns the current **TCC** version (for example, 9.0). The current [Decimal character](#)^[51] is used to separate the major and minor version numbers.

See also: [_BUILD](#)^[383].

3.14.3.9 _ACSTATUS

_ACSTATUS returns the AC line status.

value	meaning
0	Offline
1	Online
unknown	Unknown

3.14.3.10 _ADMIN

_ADMIN returns 1 if the current process is running as an administrator.

3.14.3.11 _AFSWCELL

_AFSWCELL returns the OpenAFS workstation cell.

See <http://www.openafs.org> for more information on OpenAFS.

3.14.3.12 _ALT

_ALT returns the status of the **Alt** key:

value	status of selected key
1	at least one Alt key is depressed
0	neither is depressed

3.14.3.13 _ANSI

_ANSI returns 1 if internal support for [ANSI Std. X3.64](#)^[101] is enabled, 0 if not.

3.14.3.14 _BATCH

_BATCH returns the current batch file nesting level. It is 0 if no batch file is currently being processed.

3.14.3.15 _BATCHLINE

_BATCHLINE returns the current line number in the current batch file. It is -1 if no batch file is active.

3.14.3.16 _BATCHNAME

_BATCHNAME returns the full path and file name of the current batch file. It is an empty string if no batch file is active.

3.14.3.17 _BATCHTYPE

_BATCHTYPE returns the file type of the current batch file:

<i>value</i>	<i>meaning</i>
-1	not in a batch file
0	normal
1	compressed
2	encrypted

3.14.3.18 _BATTERY

_BATTERY returns the battery charge status:

<i>value</i>	<i>meaning</i>
1	High
2	Low
4	Critical
8	Charging
128	No battery
unknown	Unknown

3.14.3.19 _BATTERYLIFE

_BATTERYLIFE returns either the number of seconds of battery life remaining, or **unknown**.

3.14.3.20 _BATTERYPERCENT

_BATTERYPERCENT returns the percentage of battery charge remaining (**0...100**), or **unknown**.

3.14.3.21 _BDEBUGGER

_BDEBUGGER returns 1 if the batch debugger is actively debugging a file, or 0 if it is not.

3.14.3.22 _BG

_BG returns a string containing the first three characters of the current background screen output color (for example, **Bla**). See [Colors, Color Names and Codes](#)^[518] for details.

3.14.3.23 _BOOT

_BOOT returns the boot drive letter, without a colon.

3.14.3.24 _BUILD

_BUILD returns the internal **TCC** build number.

See also: [_4VER](#) [38].

3.14.3.25 _CAPSLOCK

_CAPSLOCK returns the current state of the **capslock** key:

<i>value</i>	<i>toggled status</i>
1	ON
0	OFF

3.14.3.26 _CDROMS

_CDROMS returns a space-delimited list of the CD-ROM drives on the system.

3.14.3.27 _CHILDPID

_CHILDPID returns the process ID of the most recent child process.

3.14.3.28 _CI

_CI returns the insert mode cursor shape, as a percentage (**0** to **100**).

See also [SETDOS /S](#) [323] and the [Insert Cursor](#) [50] configuration option.

3.14.3.29 _CMDLINE

_CMDLINE returns 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.

3.14.3.30 _CMDPROC

_CMDPROC returns the name of the current command processor (**TCC**). This variable is obsolete.

3.14.3.31 _CMDSPEC

_CMDSPEC returns the full pathname of the command processor.

3.14.3.32 _CO

_CO returns the overstrike mode cursor shape, as a percentage (**0** to **100**).

See also [SETDOS /S](#) [323] and the [Overstrike Cursor](#) [50] configuration option.

3.14.3.33 _CODEPAGE

_CODEPAGE returns the current Windows code page.

See also [CHCP](#) [180].

3.14.3.34 **_COLUMN**

_COLUMN is the current cursor column. The leftmost column is numbered **0**.

See also [_COLUMNS](#)^[384], [_ROW](#)^[390], and [_ROWS](#)^[391].

3.14.3.35 **_COLUMNS**

_COLUMNS returns the current number of virtual screen columns (for example, **80**).

See [Resizing the Take Command Window](#)^[68] for additional details on the virtual screen width.

See also [_COLUMN](#)^[384], [_ROW](#)^[390], and [_ROWS](#)^[391].

3.14.3.36 **_CONSOLEPIDS**

_CONSOLEPIDS returns a space-delimited list of the process IDs of all processes attached to this console.

3.14.3.37 **_COUNTRY**

_COUNTRY returns the current country code as reported by the operating system. This code is usually the same as the international dialing code for the country.

3.14.3.38 **_CPU**

_CPU returns 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.

This variable is obsolete. To determine the type, revision, stepping level, and other such details for advanced processors, use the [@WININFO](#)^[467] or [@WMI](#)^[472] function.

3.14.3.39 **_CPUUSAGE**

_CPUUSAGE returns the current CPU usage, as a percent (**0** to **100**).

3.14.3.40 **_CTRL**

_CTRL returns the status of the **Ctrl** keys:

<i>value</i>	<i>status of selected key</i>
1	at least one Ctrl key is depressed
0	neither is depressed

3.14.3.41 **_CWD**

_CWD returns 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](#)^[385], [_CWP](#)^[385], [_CWPS](#)^[385], [@CWD](#)^[414], and [@CWDS](#)^[414].

3.14.3.42 _CWDS

_CWDS returns the current working directory in the format **d:\pathname**.

See also [_CWD](#)^[384], [_CWP](#)^[385], [_CWPS](#)^[385], [@CWD](#)^[414], and [@CWDS](#)^[414].

3.14.3.43 _CWP

_CWP returns the current working directory in the format **\pathname** (without the drive letter).

See also [_CWD](#)^[384], [_CWDS](#)^[385], [_CWPS](#)^[385], [@CWD](#)^[414], and [@CWDS](#)^[414].

3.14.3.44 _CWPS

_CWPS returns the current working directory in the format **\pathname** (without the drive letter).

See also [_CWD](#)^[384], [_CWDS](#)^[385], [_CWP](#)^[385], [@CWD](#)^[414], and [@CWDS](#)^[414].

3.14.3.45 _DATE

_DATE returns 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](#)^[388].

3.14.3.46 _DATETIME

_DATETIME returns the current date and time in the format yyyyMMddhhmmss. The date part is the same as [_isodate](#)^[388] without separators.

3.14.3.47 _DAY

_DAY returns the current day of the month (1 to 31).

3.14.3.48 _DETACHPID

_DETACHPID returns the process ID of the most recent process launched by the [DETACH](#)^[197] command.

3.14.3.49 _DISK

_DISK returns the current disk drive letter, without a colon (for example, **C**).

If the current directory is a UNC, **%_disk** will return the sharename.

3.14.3.50 _DNAME

_DNAME returns the name of the file used to store file descriptions. It can be changed with the [Description Filename](#)^[51] configuration option, or the [SETDOS /D](#)^[323] command.

3.14.3.51 _DOS

_DOS returns the operating system type. **Take Command** returns a different value depending on the operating system, as follows:

Platform	Take Command
Windows XP	WINXP
Windows 2003	WIN2003
Windows Vista	WINVISTA
Windows 2008	WIN2008

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. See also the [_WINVER](#)^[394] variable.

3.14.3.52 _DOSVER

_DOSVER returns the current operating system version. The current [Decimal character](#)^[51] is used to separate the major and minor version numbers.

3.14.3.53 _DOW

_DOW returns the first three characters of the name of the current day of the week (**Mon, Tue, Wed**, etc.).

3.14.3.54 _DOWF

_DOWF returns the full name of the day of the week for the current date (**Monday, Tuesday**, etc.).

3.14.3.55 _DOWI

_DOWI returns the current day of the week as an integer (**1** = Sunday, **2** = Monday, etc.).

3.14.3.56 _DOY

_DOY returns the current day of the year (1 to 366).

3.14.3.57 _DRIVES

_DRIVES returns a space-delimited list of the existing drives in the format:

A: C: D: E:

3.14.3.58 _DST

_DST returns 1 if daylight savings time is in effect, or 0 if it is not.

3.14.3.59 _DVDS

_DVDS returns a space-delimited list of the DVD drives on the system.

3.14.3.60 _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](#)^[217] command and the [Batch Echo](#)^[47] configuration option). The value returned by the **_ECHO** variable reflects the state applicable at the time the variable is queried.

3.14.3.61 _EDITMODE

_EDITMODE returns 0 if the line editor is in overstrike mode, or 1 if it is in insert mode.

3.14.3.62 _EXECSTR

_EXECSTR returns the integer return code of the last [@EXECSTR](#)^[424] function.

3.14.3.63 _EXIT

_EXIT returns the reason for exiting **TCC**:

- 0** EXIT command
- 2** CLOSE_EVENT
- 5** LOGOFF_EVENT

6 SHUTDOWN_EVENT

3.14.3.64 _EXPANSION

_EXPANSION returns the current expansion mode (i.e., [SETDOS /X](#)^[323]). It returns the string **0** if everything is enabled, or a string of up to 9 characters of the disabled modes.

For example, if you disable nested variable expansion and redirection:

```
setdos /x-46
```

then %_expansion will return **46**.

3.14.3.65 _FG

_FG returns a string containing the first three letters of the current foreground screen output color (for example, "Whi"). See [Colors, Color Names and Codes](#)^[518] for details.

3.14.3.66 _FTPERROR

_FTPERROR returns the error code of the last error reported by [FTP](#)^[93]. 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)

3.14.3.67 _HDRIVES

_HDRIVES returns a space-delimited list of the hard (fixed) drives on the system.

3.14.3.68 _HLOGFILE

_HLOGFILE returns the name of the current history log file (or an empty string if LOG /H is OFF). See [LOG](#)^[269] for information on history logging.

3.14.3.69 _HOST

_HOST returns the host name for the local computer.

3.14.3.70 _HOUR

_HOUR returns the current hour (0 - 23).

3.14.3.71 _HWPFILE

_HWPFILE returns the name of the current Windows hardware profile.

3.14.3.72 _IDLETICKS

_IDLETICKS returns the number of milliseconds since the last user input.

3.14.3.73 _IDOW

_IDOW returns the 3-character abbreviation for the day of the week for the current date, in the current locale language.

3.14.3.74 _IDOWF

_IDOWF returns the full name for the day of the week for the current date, in the current locale language.

3.14.3.75 _IFTP

_IFTP returns **1** if an [IFTP](#) ^[255] session is active, **0** if it is not.

3.14.3.76 _IFTPS

_IFTPS returns **1** if an SSL [IFTP](#) ^[255] session is active, **0** if it is not.

3.14.3.77 _IMONTH

_IMONTH returns the abbreviated name for the current month, in the current locale language.

3.14.3.78 _IMONTHF

_IMONTHF returns the full name for the current month, in the current locale language.

3.14.3.79 _ININAME

_ININAME returns the fully qualified pathname of the INI file used by the current shell.

3.14.3.80 _IP

_IP returns 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.

3.14.3.81 _ISODATE

_ISODATE returns the current system date, in ISO 9601 format (**yyyy-mm-dd**).

See also [_DATE](#) ^[385] and [_DATETIME](#) ^[385].

3.14.3.82 _KBHIT

_KBHIT returns **1** if one or more keystrokes are waiting in the keyboard buffer, or **0** if the keyboard buffer is empty.

3.14.3.83 _LALT

_LALT returns the status of the left **Alt** key:

<i>value</i>	<i>key status</i>
--------------	-------------------

1	depressed
0	not depressed

See also [_ALT](#) ^[384].

3.14.3.84 _LASTDISK

_LASTDISK returns the last valid drive letter (without a colon).

3.14.3.85 _LCTRL

_LCTRL returns the status of the Left Ctrl key:

<i>value</i>	<i>key status</i>
1	depressed
0	not depressed

See also [_CTRL](#) ^[384].

3.14.3.86 _LOGFILE

_LOGFILE returns the name of the current log file (or an empty string if LOG is OFF). See [LOG](#) ^[269] for information on logging.

3.14.3.87 _LSHIFT

_LSHIFT returns the status of the left shift key:

<i>value</i>	<i>key status</i>
1	depressed
0	not depressed

See also [_SHIFT](#) ^[391].

3.14.3.88 _MINUTE

_MINUTE returns the current minute (0 - 59).

3.14.3.89 _MONITORS

_MONITORS returns the number of video displays.

3.14.3.90 _MONTH

_MONTH returns the current numeric month of the year (1 to 12).

3.14.3.91 _MONTHF

_MONTHF returns the full name of the current month (**January**, **February**, etc.).

3.14.3.92 _NUMLOCK

_NUMLOCK reports the current of the *numlock* key:

<i>value</i>	<i>toggled status</i>
1	ON
0	OFF

3.14.3.93 _OPENAFS

_OPENAFS returns **1** if the [OpenAFS](http://www.openafs.org) ^[103] service is active, **0** if it is not.

See <http://www.openafs.org> for more information on OpenAFS.

3.14.3.94 _OSBUILD

_OSBUILD returns the Windows build number.

3.14.3.95 _PARENT

_PARENT returns the name of the parent process (the process that started **TCC**).

3.14.3.96 _PID

_PID returns the current process ID number.

3.14.3.97 _PIPE

_PIPE returns **1** if the current process is running inside a pipe, and **0** otherwise.

3.14.3.98 _PPID

_PPID returns the process ID number of the parent process.

3.14.3.99 _RALT

_RALT returns the status of the right Alt key:

<i>value</i>	<i>key status</i>
1	depressed
0	not depressed

See also [_ALT](#) ^[381].

3.14.3.100 _RCTRL

_RCTRL returns the status of the right Ctrl key:

<i>value</i>	<i>key status</i>
1	depressed
0	not depressed

See also [_CTRL](#) ^[384].

3.14.3.101 _READY

_READY returns a space-delimited list of the currently ready (accessible) drives in the format :

C: D: E:

3.14.3.102 _REGISTERED

_REGISTERED returns the registered name of the user or an empty string if **Take Command** isn't registered.

3.14.3.103 _ROW

_ROW returns the current cursor row (for example, **0** for the top of the window).

3.14.3.10 _ROWS

_ROWS returns the current number of screen rows in the **TCC** window (for example, **25**).

3.14.3.10 _RSHIFT

_RSHIFT returns the status of the right Shift key:

<i>value</i>	<i>key status</i>
1	depressed
0	not depressed

See also [_SHIFT](#)^[391].

3.14.3.10 _RUBYTYPE

_RUBYTYPE returns the type of the Ruby VALUE returned by the last [@RUBY](#)^[458] call.

3.14.3.10 _RUBYVALUE

_RUBYVALUE returns the Ruby VALUE returned by the last [@RUBY](#)^[458] call.

3.14.3.10 _SCROLLLOCK

_SCROLLLOCK reports the current state of **scrolllock** mode, which can be toggled using the **scrolllock** key:

<i>value</i>	<i>toggled status</i>
1	ON
0	OFF

3.14.3.10 _SECOND

_SECOND is the current second (0 - 59).

3.14.3.11 _SELECTED

_SELECTED returns the first line of text highlighted in the **TCC** window. If no text has been highlighted, **SELECTED** returns an empty string.

3.14.3.11 _SHELL

_SHELL is the current shell instance identifier, one for each command processor. **_SHELL** will return 0 for a primary shell, or 1 (or higher) for a shell started by another **TCC** process (either directly or via a pipe).

3.14.3.11 _SHELLS

_SHELLS is the current shell identifier, one for each command processor. **_SHELLS** duplicates the behavior of **_SHELL** in older versions (7.01 and earlier) of **4NT** and **Take Command**.

3.14.3.11 _SHIFT

_SHIFT is the status the two **Shift** keys:

<i>value</i>	<i>status of selected key</i>
--------------	-------------------------------

1	at least one is depressed
0	neither is depressed

3.14.3.11!_SHORTCUT

_SHORTCUT returns the full pathname of the shortcut file that started this process. If the process was not started from a shortcut, **_SHORTCUT** returns an empty string.

3.14.3.11!_SHRALIAS

_SHRALIAS returns **1** if [SHRALIAS](#)^[329] is loaded, **0** if it is not.

3.14.3.11!_STARTPATH

_STARTPATH returns the startup directory for the current shell (not necessarily the same as the location of the executable!)

3.14.3.11!_STARTPID

_STARTPID returns the process ID of the most recent process launched by the [START](#)^[331] command.

3.14.3.11!_STDIN

_STDIN returns **1** if STDIN points to the console, or **0** if it has been redirected.

3.14.3.11!_STDOUT

_STDOUT returns **1** if STDOUT points to the console, or **0** if it has been redirected.

3.14.3.12!_STDERR

_STDERR returns **1** if STDERR points to the console, or **0** if it has been redirected.

3.14.3.12!_STZN

_STZN returns the name of standard time in the current time zone.

See also [_STZO](#)^[392], [_TZN](#)^[393], and [_TZO](#)^[393].

3.14.3.12!_STZO

_STZO returns the offset in minutes from UTC for standard time in the current time zone.

See also [_STZN](#)^[392], [_TZN](#)^[393], and [_TZO](#)^[393].

3.14.3.12!_SYSERR

_SYSERR returns 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](#)^[507] table in the Reference section for examples.

3.14.3.12!_TCFILTER

_TCFILTER returns the current filter in the **Take Command** List view window if **TCC** is running in a tab window, or an empty string if it is not.

3.14.3.12!_TCFOLDER

_TCFOLDER returns the selected folder in the *Take Command* Folders window if *TCC* is running in a tab window, or an empty string if it is not.

3.14.3.12!_TCTAB

_TCTAB returns **1** if this *TCC* process is running in a *Take Command* tab window, or **0** if it is not.

3.14.3.12!_TIME

_TIME returns the current system time in the format **hh:mm:ss**. The separator character may vary depending upon your country information.

3.14.3.12!_TRANSIENT

_TRANSIENT returns **1** if the current shell is transient (started with a */C*, see [Command Line Options](#) ^[19] for details), or **0** otherwise.

3.14.3.12!_TZN

_TZN returns the name of the current time zone.

See also [_STZN](#) ^[392], [_STZO](#) ^[392], and [_TZO](#) ^[393].

3.14.3.13!_TZO

_TZO returns the offset in minutes from UTC for the current time zone.

See also [_STZN](#) ^[392], [_STZO](#) ^[392], and [_TZN](#) ^[393].

3.14.3.13!_UNICODE

_UNICODE returns **1** if the shell is currently using Unicode for redirected output, **0** otherwise.

3.14.3.13!_UTCDATE

_UTCDATE returns the current UTC date in the user's default format.

3.14.3.13!_UTCDATETIME

_UTCDATETIME returns the current date and time in UTC.

3.14.3.13!_UTCHOUR

_UTCHOUR returns the current UTC hour.

3.14.3.13!_UTCISODATE

_UTCISODATE returns the current UTC date in ISO format (yyyy-mm-dd).

3.14.3.13!_UTCMINUTE

_UTCMINUTE returns the current UTC minute.

3.14.3.13!_UTCSECOND

_UTCSECOND returns the current UTC second.

3.14.3.13!_UTCTIME

_UTCTIME returns the current UTC time.

3.14.3.13!_VIRTUALPC

_VIRTUALPC returns 1 if **TCC** is running inside VirtualPC virtual machine.

3.14.3.14!_VMWARE

_VMWARE returns 1 if **TCC** is running inside a VMWare virtual machine.

3.14.3.14!_VXPIXELS

_VXPIXELS returns the horizontal size of the virtual screen (including multiple monitors) in pixels.

3.14.3.14!_VYPIXELS

_VXPIXELS returns the vertical size of the virtual screen (including multiple monitors) in pixels.

3.14.3.14!_WINDIR

_WINDIR returns the pathname of the Windows directory.

3.14.3.14!_WINFGWINDOW

_WINFGWINDOW returns the title of the foreground window.

3.14.3.14!_WINNAME

_WINNAME returns the computer name of the current system.

3.14.3.14!_WINSYSDIR

_WINSYSDIR returns the pathname of the Windows system directory.

3.14.3.14!_WINTICKS

_WINTICKS returns the number of milliseconds since Windows was started.

3.14.3.14!_WINUSER

_WINUSER returns the name of the user currently logged on.

3.14.3.14!_WINVER

_WINVER returns the current Windows version number. The current [Decimal character](#)^[51] is used to separate the major and minor version numbers.

3.14.3.15!_WINTITLE

_WINTITLE returns the title of the current window.

3.14.3.15!_WOW64

_WOW64 returns 1 if **TCC** is running in the WOW64 environment (64-bit Windows).

3.14.3.15!_XPIXELS

_XPIXELS returns the physical screen horizontal size in pixels.

3.14.3.15!_YEAR

_YEAR returns the current year (1980 to 2099).

3.14.3.15 _YPIXELS

_YPIXELS returns the physical screen vertical size in pixels.

3.14.3.15 ERRORLEVEL

ERRORLEVEL is an alternate name (included for compatibility with CMD.EXE) for the [_?](#)^[380] variable, 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: For compatibility with CMD.EXE, some internal commands, e.g., **DIR**, also set this variable to the same value as the variable [_?](#)^[380], which destroys the code from the last external command. If you need to preserve the return value of the external command, save the value in a variable immediately upon command completion, and use the saved variable instead. We also strongly recommend that for internal commands you query the [_?](#)^[380] variable instead.

See also [_?](#)^[380]

3.14.4 Variable Functions

Variable functions are very similar to internal variables, but they take one or more parameters (which can be environment variables or even other variable functions).

Variable functions are useful at the command prompt as well as in [aliases](#)^[128] and [batch files](#)^[130] to check on available system resources, manipulate strings and numbers, and work with files and filenames.

The variable functions built into **TCC** are listed in alphabetical order in subsequent topics. You can also obtain help from the command prompt on any function with a **HELP @functionname** command, or by pressing [Ctrl-F1](#)^[34] when the cursor is on the function name. See the [HELP](#)^[25] command for details

Note: The [FUNCTION](#)^[242] command can be used to create, edit, or display user-defined variable functions, and the [UNFUNCTION](#)^[356] to delete them.

For a list of Variable Functions organized by general categories of use, see [Variable Functions by Category](#)^[40].

Syntax

To have either a user-defined or a built-in variable function evaluated, its name must be preceded by a percent sign % (**%@EVAL**, **%@LEN**, etc.). All variable functions must have square brackets **[]** enclosing their parameter(s), if any. No space is allowed between the function name and the **[**. The combined parameters of a variable function may not exceed 8,191 characters.

Memory Size / Disk Space / File Size Units and Report Format

Some variable functions, such as [@DISKFREE](#)^[416], accept an optional parameter **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](#)^[514]) in the value returned 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 parameter, i.e.:

```
echo %@DISKFREE[C,bc]
```

Notes

- 1) Disk manufacturers use the prefixes adopted from the metric system (kilo, mega, giga, tera) in their original 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 **TCC** is case sensitive.

Date Parameter Format

See the [Date Formats](#)^[127] topic.

File Name Parameters

Filenames passed as variable function parameters must be enclosed in double 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 white space or special characters in the name properly.

Drive Letter Parameters

In variable functions which take a drive letter as a parameter, like [@DISKFREE](#)^[416] or [@READY](#)^[453], 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](#)^[429], [@FILEWRITE](#)^[433], [@FILEWRITEB](#)^[433], [@FILESEEK](#)^[431], [@FILESEEKL](#)^[431], and [@FILECLOSE](#)^[427] functions allow you to access files based on their file handle. These functions must be used only with file handles returned by [@FILEOPEN](#)^[428], 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](#)^[86] 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 more comprehensive set of examples, see the file **EXAMPLES.BTM**, which comes with **TCC**.

The command below sets the prompt to show the amount of free memory (see [PROMPT](#)^[297] for details on including variable functions in your prompt):

```
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)**:

```
alias calc `echo The answer is: %@eval[%$]`
```

3.14.4.1 Functions by Name

@ABS ^[407]	Absolute value of number
@AFSCCELL ^[407]	OpenAFS cell name
@AFSMOUNT ^[407]	OpenAFS mount point
@AFSPATH ^[407]	Path in OpenAFS: 1, otherwise 0
@AFSSYMLINK ^[407]	OpenAFS symbolic link
@AFSVOLID ^[408]	OpenAFS volume ID
@AFSVOLNAME ^[408]	OpenAFS volume name
@AGEDATE ^[408]	Converts an age ^[520] into date and time
@ALIAS ^[408]	Value of an alias
@ALTNAME ^[408]	Short name for the file.
@ASCII ^[409]	Set of ASCII-s for characters in string
@ASSOC ^[409]	File association
@ATTRIB ^[409]	Test or return file attributes
@AVERAGE ^[410]	Average of a list of numbers
@CAP ^[410]	Call a <code>_cdecl</code> function in a DLL
@CAPS ^[411]	Capitalize first character of each word
@CDROM ^[411]	CD-ROM drive: 1, otherwise 0
@CEILING ^[411]	Smallest integer not less than a number
@CHAR ^[411]	Character string, given a set of ASCII-s
@CLIP ^[412]	Specified line from clipboard
@CLIPW ^[412]	Write string to the clipboard
@COLOR ^[412]	RGB value of a color
@COMMA ^[412]	Insert commas into a number (thousands separators)
@COMPARE ^[413]	Two files are identical: 1, otherwise 0
@CONSOLE ^[413]	Identify console sessions
@CONVERT ^[413]	Convert value from input base to output base
@COUNT ^[413]	Number of times a character appears in a string
@CRC32 ^[413]	File CRC
@CWD ^[414]	Current Working Directory of specified drive
@CWDS ^[414]	Current Working Directory of specified drive, with trailing backslash
@DATE ^[414]	Convert date to number of days
@DAY ^[414]	Day of month for date

@DEC ⁴¹⁵	Decrement a numeric value by 1
@DECIMAL ⁴¹⁵	Decimal portion of a number
@DESCRIPT ⁴¹⁵	File description
@DEVICE ⁴¹⁶	Character device: 1, otherwise 0
@DIGITS ⁴¹⁶	String is all digits: 1, otherwise 0
@DIRSTACK ⁴¹⁶	Directory stack entry
@DISKFREE ⁴¹⁶	Free disk space
@DISKTOTAL ⁴¹⁷	Total disk space
@DISKUSED ⁴¹⁷	Used disk space
@DOMAIN ⁴¹⁷	Domain name of a computer
@DOW ⁴¹⁷	Short name of day of week for date
@DOWF ⁴¹⁸	Full name of day of week for date
@DOWI ⁴¹⁸	Day of week number for date
@DOY ⁴¹⁹	Day of year for date
@DRIVETYPE ⁴¹⁹	Type of a drive
@DRIVETYPEEX ⁴¹⁹	Type of a drive

@ENUMSERVERS ⁴¹⁹	Identify server names on a network
@ENUMSHARES ⁴²⁰	Identify sharenames on a server
@ERRTEXT ⁴²⁰	Windows error description
@EVAL ⁴²⁰	Arithmetic calculations
@EXEC ⁴²⁴	Execute a command and return its exit code
@EXECSTR ⁴²⁴	Execute a command and return the first output line
@EXETYPE ⁴²⁴	Application type
@EXPAND ⁴²⁵	All names that match filename
@EXT ⁴²⁵	File extension

@FIELD ⁴²⁶	Extract a field from a string
@FIELDS ⁴²⁷	Count fields in a string
@FILEAGE ⁴²⁷	File age ⁵²⁰ (date and time)
@FILECLOSE ⁴²⁷	Close a file handle
@FILEDATE ⁴²⁸	File date
@FILENAME ⁴²⁸	File name and extension
@FILEOPEN ⁴²⁸	Open a file handle
@FILEREAD ⁴²⁹	Read next line from a file
@FILEREADB ⁴²⁹	Read bytes from a file
@FILES ⁴³⁰	Number of files matching a wildcard
@FILESEEK ⁴³¹	Move a file handle pointer to specified file position
@FILESEEKL ⁴³¹	Move a file handle pointer to a specified line
@FILESIZE ⁴³²	Total size of files matching a wildcard
@FILETIME ⁴³²	File time
@FILEWRITE ⁴³³	Write next line to a file
@FILEWRITEB ⁴³³	Write data to a file
@FINDCLOSE ⁴³⁴	Closes the search handle.
@FINDFIRST ⁴³⁴	Find first matching file
@FINDNEXT ⁴³⁵	Find next matching file
@FLOOR ⁴³⁵	Largest integer not larger than a number
@FORMAT ⁴³⁵	Formats data string according to format string
@FORMATN ⁴³⁶	Format a numeric value
@FORMATNC ⁴³⁶	Format a numeric value and insert the thousands separator(s)
@FSTYPE ⁴³⁶	File system type (FAT, NTFS, CDFS, etc.)
@FTYPE ⁴³⁷	Open command string for file type
@FULL ⁴³⁷	Full file name with path
@FUNCTION ⁴³⁷	Definition of a function

@GETDIR ⁴³⁷	Prompt for a directory name.
@GETFILE ⁴³⁸	Prompt for a path and file name.
@GETFOLDER ⁴³⁸	Folder name from tree view.
@GROUP ⁴³⁸	User is member of group: 1, otherwise 0
@HISTORY ⁴³⁹	A line or word from the command history
@IDOW ⁴³⁹	Short local name of day of week for date
@IDOWF ⁴³⁹	Full local name of day of week for date
@IF ⁴³⁹	Evaluates a conditional expression
@INC ⁴⁴⁰	Increment a numeric value by 1
@INDEX ⁴⁴⁰	Offset of string2 within string1
@INIREAD ⁴⁴¹	Return an entry from an .INI file
@INIWRITE ⁴⁴¹	Write an entry in an .INI file
@INSERT ⁴⁴²	Inserts string1 into string2
@INODE ⁴⁴²	File Inode (in hex)
@INSTR ⁴⁴³	Extract a substring
@INT ⁴⁴³	Integer part of a number
@IPADDRESS ⁴⁴³	Returns the numeric IP for a host name
@IPNAME ⁴⁴⁴	Returns the host name for a numeric IP address
@ISALNUM ⁴⁴⁴	Test for alphanumeric characters
@ISALPHA ⁴⁴⁴	Test for alphabetic characters
@ISASCII ⁴⁴⁴	Test for ASCII characters
@ISCNTRL ⁴⁴⁵	Test for control characters
@ISDIGIT ⁴⁴⁵	Test for decimal digits
@ISPRINT ⁴⁴⁵	Test for printable characters
@ISPROC ⁴⁴⁵	Returns 1 if the process is active; otherwise 0
@ISPUNCT ⁴⁴⁵	Test for punctuation characters
@ISSPACE ⁴⁴⁶	Test for white space characters
@ISXDIGIT ⁴⁴⁶	Test for hexadecimal digits
@JUNCTION ⁴⁴⁶	Directory referenced by the junction
@LABEL ⁴⁴⁶	Volume label
@LCS ⁴⁴⁶	Longest common sequence in two strings
@LEFT ⁴⁴⁶	Left end of string
@LEN ⁴⁴⁷	Length of a string
@LFN ⁴⁴⁷	Long name for a short filename
@LINE ⁴⁴⁷	Specified line from a file
@LINES ⁴⁴⁸	Count of lines in a file
@LINKS ⁴⁴⁸	Number of NTFS hard links for the file
@LOWER ⁴⁴⁸	Convert string to lower case
@LTRIM ⁴⁴⁸	Left trim specified characters.
@MAKEAGE ⁴⁴⁹	Convert date and time to age ⁵²⁰
@MAKEDATE ⁴⁴⁹	Convert number of days to date
@MAKETIME ⁴⁴⁹	Convert number of seconds to time
@MAX ⁴⁵⁰	Largest integer in the list
@MD5 ⁴⁵⁰	MD5 hash of a string or file
@MIN ⁴⁵⁰	Smallest integer in the list
@MONTH ⁴⁵⁰	Month for date

@NAME ^[451]	File name without path or extension
@NUMERIC ^[451]	Test if a string is numeric
@OPTION ^[452]	Current configuration option value
@OWNER ^[452]	Return file owner
@PATH ^[452]	File path without name
@PERL ^[452]	Evaluate a Perl expression
@PING ^[453]	Response time from a host
@QUOTE ^[453]	Double quote the argument if necessary
@RANDOM ^[453]	Generate a random integer
@READSCR ^[453]	Read characters from the screen
@READY ^[453]	Drive ready: 1, otherwise 0
@REGCREATE ^[454]	Create registry subkey
@REGDELKEY ^[454]	Delete a registry key and its subkeys
@REGEX ^[454]	Match a regular expression
@REGEXINDEX ^[454]	Return the offset of a regular expression match
@REGEXIST ^[454]	Test if a registry key exists
@REGEXSUB ^[455]	Return nth matching regular expression group
@REGQUERY ^[455]	Read value from registry
@REGSET ^[455]	Write value to registry
@REGSETENV ^[455]	Write value to registry and broadcast change.
@REGTYPE ^[455]	Return type of registry variable
@REMOTE ^[456]	Remote (network) drive: 1, otherwise 0
@REMOVABLE ^[456]	Removable drive: 1, otherwise 0
@REPEAT ^[456]	Repeat a character
@REPLACE ^[456]	Replace string1 with string2 in text
@REVERSE ^[457]	Reverse a string
@REXX ^[457]	Value of executing an expression by REXX
@RIGHT ^[457]	Right end of string.
@RTRIM ^[457]	Removes specified trailing characters.
@RUBY ^[458]	Evaluate a Ruby expression
@SCRIPT ^[458]	Evaluate expression in an active scripting engine.
@SEARCH ^[458]	Path search
@SELECT ^[458]	Menu selection
@SERIAL ^[459]	Serial number of a disk
@SFN ^[460]	Short name for a long filename
@SHA1 ^[460]	SHA1 checksum for the file
@SHA256 ^[460]	SHA2-256 checksum for the file
@SHA384 ^[460]	SHA2-384 checksum for the file
@SHA512 ^[460]	SHA2-512 checksum for the file
@SHFOLDER ^[461]	Get Windows folder locations
@SIMILAR ^[462]	Compare two strings for similarity
@SNAPSHOT ^[462]	Save a window or desktop as a BMP
@STRIP ^[462]	Strips all characters in char from string
@SUBST ^[463]	Substitute a string within another string
@SUBSTR ^[463]	Extract a substring
@SUMMARY ^[462]	Query or set the NTFS SummaryInformation stream
@SYMLINK ^[463]	Target of a symbolic link

@TIME ^[463]	Convert a time of day to number of seconds
@TIMER ^[463]	Get split time from timer.
@TRIM ^[464]	Remove blanks from a string
@TRUENAME ^[464]	Find true name of a file
@TRUNCATE ^[464]	Truncate file at current position
@UNC ^[464]	UNC name of a file
@UNICODE ^[464]	Numeric UNICODE value for a character
@UNIQUE ^[465]	Create file with unique name
@UNQUOTE ^[465]	Remove double quotes from a filename
@UNQUOTES ^[465]	Remove leading and trailing double quotes
@UPPER ^[465]	Convert string to upper case
@VERINFO ^[465]	Executable file version information
@WATTRIB ^[466]	Test or return file attributes
@WILD ^[466]	Compares strings using wildcards
@WINAPI ^[467]	Call a Windows API function
@WINCLASS ^[467]	Title of first window with classname
@WINEXENAME ^[467]	Executable name for window
@WININFO ^[467]	Current system information
@WINMEMORY ^[468]	Windows memory information
@WINMETRICS ^[468]	Windows system metrics
@WINPOS ^[470]	Window position
@WINSTATE ^[470]	Current state of window
@WINSYSTEM ^[470]	Set/get windows system parameters
@WMI ^[472]	Query WMI
@WORD ^[472]	Extract a word from a string
@WORDS ^[473]	Count words in a string
@WORKGROUP ^[473]	Workgroup name of a computer
@XMLCLOSE ^[473]	Close an XML file previously opened by @XMLOPEN
@XMLNODES ^[473]	Return the number of nodes (children) for the specified path in an XML file
@XMLOPEN ^[474]	Open an XML file for use by @XMLXPath and/or @XMLNODES
@XMLXPath ^[474]	Return text of XML element
@YEAR ^[474]	Year for date

3.14.4.2 Functions by Category

This list gives a one-line description of all built-in [Variable Functions](#)^[395], 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** command at the prompt or by pressing [Ctrl-F1](#)^[34] when the cursor is on the function name. See the [HELP](#)^[25] command for details

- [Dates and times](#)^[405]
- [Drives and devices](#)^[402]
- [File content](#)^[402]
- [File names](#)^[403]
- [File properties](#)^[403]
- [Input dialog boxes](#)^[405]
- [Network properties](#)^[405]
- [Numbers and arithmetic](#)^[405]
- [Strings and characters](#)^[404]
- [System status](#)^[402]
- [Utility](#)^[406]

Note: many functions have functionality that covers several categories.

System status

@ASSOC	409	File association for the extension
@CLIP	412	Specified line from clipboard
@CLIPW	412	Write string to the clipboard
@CONSOLE	413	Identify console sessions
@ERRTEXT	420	Windows error description
@FTYPE	437	Open command string for the file type
@ISPROC	445	Returns 1 if a process is active; otherwise 0
@READSCR	453	Read characters from the screen
@REGCREATE	454	Create registry subkey
@REGDELKEY	454	Delete a registry key and its subkeys
@REGEXIST	454	Test if a registry key exists
@REGQUERY	455	Read value from registry
@REGSET	455	Write value to registry
@REGSETENV	455	Write value to registry and broadcast change.
@REGTYPE	455	Type of registry variable
@WINCLASS	467	Title of first window with classname
@WINEXENAME	467	Executable name for window
@WININFO	467	Current system information
@WINMEMORY	468	Windows memory information
@WINMETRICS	468	Windows system metrics
@WINPOS	470	Window position
@WINSTATE	470	Current state of window
@WINSYSTEM	470	Set/get windows system parameters

Directories, drives and devices

@CDROM	411	CD-ROM drive: 1, otherwise 0
@CWD	414	Current Working Directory of specified drive
@CWDS	414	Current Working Directory of specified drive, with trailing backslash
@DEVICE	416	Character device: 1, otherwise 0
@DISKFREE	416	Free disk space
@DISKTOTAL	417	Total disk space
@DISKUSED	417	Used disk space
@DRIVETYPE	419	Type of drive (hard drive, CD-ROM, etc.)
@DRIVETYPEEX	419	Type of drive (hard drive, CD-ROM, etc.)
@FSTYPE	436	File system type (FAT, NTFS, CDFS, etc.)
@JUNCTION	446	Directory referenced by the junction
@LABEL	446	Volume label
@READY	453	Drive ready: 1, otherwise 0
@REMOTE	456	Remote (network) drive: 1, otherwise 0
@REMOVABLE	456	Removable drive: 1, otherwise 0
@SERIAL	459	Serial number of a disk
@SHFOLDER	461	Windows folder locations
@SYMLINK	463	Target of a symbolic link

File content

@COMPARE	413	Compare two files
@CRC32	413	File CRC
@FILECLOSE	427	Close a file handle
@FILEOPEN	428	Open a file handle

@FILEREAD ⁴²⁹	Read next line from a file
@FILEREADB ⁴²⁹	Read bytes from a file
@FILESEEK ⁴³¹	Move a file handle pointer
@FILESEEKL ⁴³¹	Move a file handle pointer to a specified line
@FILEWRITE ⁴³³	Write next line to a file
@FILEWRITEB ⁴³³	Write data to a file handle
@INIREAD ⁴⁴¹	Return an entry from an .INI file
@INIWRITE ⁴⁴¹	Write an entry in an .INI file
@INODE ⁴⁴²	Inode value for a file
@LINE ⁴⁴⁷	Specified line from a file
@LINES ⁴⁴⁸	Count lines in a file
@LINKS ⁴⁴⁸	Number of NTFS hard links for a file
@MD5 ⁴⁵⁰	MD5 hash of a string or file
@SHA1 ⁴⁶⁰	SHA1 checksum for a file
@SHA256 ⁴⁶⁰	SHA2-256 checksum for a file
@SHA384 ⁴⁶⁰	SHA2-384 checksum for a file
@SHA512 ⁴⁶⁰	SHA2-512 checksum for a file
@SUMMARY ⁴⁶²	NTFS SummaryInformation stream for a file
@TRUNCATE ⁴⁶⁴	Truncate file at current position
@VERINFO ⁴⁶⁵	Executable file version information

File names

@ALTNAME ⁴⁰⁸	Short name for the file.
@EXPAND ⁴²⁵	All names that match filename
@EXT ⁴²⁵	File extension
@FILENAME ⁴²⁸	File name and extension
@FULL ⁴³⁷	Full file name with path
@LFN ⁴⁴⁷	Long name for a short filename
@NAME ⁴⁵¹	File name without path or extension
@PATH ⁴⁵²	File path without name
@QUOTE ⁴⁵³	Double quote a filename
@SFN ⁴⁶⁰	Short name for a long filename
@SEARCH ⁴⁵⁸	Path search
@TRUENAME ⁴⁶⁴	True name of a file
@UNC ⁴⁶⁴	UNC name of a file
@UNIQUE ⁴⁶⁵	Create file with unique name
@UNQUOTE ⁴⁶⁵	Remove double quotes from a filename
@UNQUOTES ⁴⁶⁵	Remove leading and trailing double quotes

File properties

@ATTRIB ⁴⁰⁹	Test or return file attributes
@DESCRIP ⁴¹⁵	File description
@EXETYPE ⁴²⁴	Application type
@FILEAGE ⁴²⁷	File age ⁵²⁰ (date and time)
@FILEDATE ⁴²⁸	File date
@FILES ⁴³⁰	Number of files matching a wildcard
@FILESIZE ⁴³²	Total size of files matching a wildcard
@FILETIME ⁴³²	File time
@FINDCLOSE ⁴³⁴	Closes the search handle.
@FINDFIRST ⁴³⁴	Find first matching file
@FINDNEXT ⁴³⁵	Find next matching file
@INODE ⁴⁴²	Inode value for a file

@LINKS ⁴⁴⁸	Number of NTFS hard links for a file
@OWNER ⁴⁵²	File owner
@SEARCH ⁴⁵⁸	Path search
@SUMMARY ⁴⁶²	NTFS SummaryInformation stream for a file
@TRUENAME ⁴⁶⁴	True name for a file
@UNIQUE ⁴⁶⁵	Create file with unique name
@VERINFO ⁴⁶⁵	Executable file version information
@WATTRIB ⁴⁶⁶	Test or return file attributes

Strings and characters

@ASCII ⁴⁰⁹	List of ASCII-s for characters in string
@CAPS ⁴¹¹	Capitalize first character of each word
@CHAR ⁴¹¹	Character string, given a set of ASCII-s
@COUNT ⁴¹³	Counts occurrences of a character in a string
@EXECSTR ⁴²⁴	Execute a command and return its first output line
@FIELD ⁴²⁶	Extract a field from a string
@FIELDS ⁴²⁷	Count fields in a string
@FORMAT ⁴³⁵	Formats data string according to format string
@INDEX ⁴⁴⁰	Offset of string2 within string1
@INSERT ⁴⁴²	Insert string1 into string2
@INSTR ⁴⁴³	Extract a substring
@ISALNUM ⁴⁴⁴	Test for alphanumeric characters
@ISALPHA ⁴⁴⁴	Test for alphabetic characters
@ISASCII ⁴⁴⁴	Test for ASCII characters
@ISCNTRL ⁴⁴⁵	Test for control characters
@ISDIGIT ⁴⁴⁵	Test for decimal digits
@ISPRINT ⁴⁴⁵	Test for printable characters
@ISPUNCT ⁴⁴⁵	Test for punctuation characters
@ISSPACE ⁴⁴⁶	Test for white space characters
@ISXDIGIT ⁴⁴⁶	Test for hexadecimal digits
@LCS ⁴⁴⁶	Longest common sequence in two strings
@LEFT ⁴⁴⁶	Left end of string
@LEN ⁴⁴⁷	Length of a string
@LOWER ⁴⁴⁸	Convert string to lower case
@LTRIM ⁴⁴⁸	Trims specified leading characters.
@MD5 ⁴⁵⁰	MD5 hash of a string or file
@REGEX ⁴⁵⁴	Return a Regular Expression test
@REGEXINDEX ⁴⁵⁴	Return the offset of a regular expression match
@REGEXSUB ⁴⁵⁵	Return the nth matching group of a regular expression test
@REPEAT ⁴⁵⁶	Repeat a character
@REPLACE ⁴⁵⁶	Replace string1 with string2 in text
@REVERSE ⁴⁵⁷	Reverse a string
@RIGHT ⁴⁵⁷	Right end of string
@RTRIM ⁴⁵⁷	Trims specified trailing characters.
@SIMILAR ⁴⁶²	Test similarity between two strings
@STRIP ⁴⁶²	Strips all characters in char from string
@SUBST ⁴⁶³	Substitute a string within another string
@SUBSTR ⁴⁶³	Older version of @INSTR ⁴⁴³ to extract a substring
@TRIM ⁴⁶⁴	Remove blanks from a string
@UNICODE ⁴⁶⁴	List of UNICODEs for characters in string
@UPPER ⁴⁶⁵	Convert string to upper case
@WILD ⁴⁶⁶	Compares strings using wildcards

@WORD ^[472]	Extract a word from a string
@WORDS ^[473]	Count words in a string

Numbers and arithmetic

@ABS ^[407]	Absolute value of n
@AVERAGE ^[410]	Average of a list
@CEILING ^[411]	Smallest integer not less than n
@COMMA ^[412]	Insert commas (thousands separators) into a numeric string
@CONVERT ^[413]	Convert value from input base to output base
@DEC ^[415]	Decrement a numeric value by 1
@DECIMAL ^[415]	Decimal fraction portion of a number
@DIGITS ^[416]	Tests if string is all digits
@EVAL ^[420]	Arithmetic calculations
@FORMATN ^[436]	Format a numeric value
@FORMATNC ^[436]	Format a numeric value and insert thousands separators
@FLOOR ^[435]	Largest integer not larger than n
@INC ^[440]	Increment a numeric value by 1
@INT ^[443]	Integer part of a number
@MAX ^[450]	Largest integer in the list
@MIN ^[450]	Smallest integer in the list
@NUMERIC ^[451]	Test if a string is numeric
@RANDOM ^[453]	Generate a random integer

Dates and times

@AGEDATE ^[408]	Converts an age ^[520] into date and time
@DAY ^[414]	Day of month for date
@DATE ^[414]	Convert date to number of days
@DOW ^[417]	Short name of day of week for date
@DOWF ^[418]	Full name of day of week
@DOWI ^[418]	Day of week as integer
@DOY ^[419]	Day of year for date
@IDOW ^[439]	Short localized name of day of week for date
@IDOWF ^[439]	Full localized name of day of week for date
@MAKEAGE ^[449]	Convert date and time to age ^[520]
@MAKEDATE ^[449]	Convert number of days to date
@MAKETIME ^[449]	Convert number of seconds to time
@MONTH ^[450]	Month in specified date
@TIME ^[463]	Convert time to number of seconds
@YEAR ^[474]	Year for date

Input dialog boxes

@GETDIR ^[437]	Prompt for a directory name.
@GETFILE ^[438]	Prompt for a path and file name.
@GETFOLDER ^[438]	Folder name from tree view.
@SELECT ^[458]	Menu selection

Network properties

@AFSCCELL ^[407]	OpenAFS cell name for a path
@AFSMOUNT ^[407]	OpenAFS mount point for a path
@AFSPATH ^[407]	Path is in OpenAFS: 1, otherwise 0

@AFSSYMLINK ⁴⁰⁷	OpenAFS symbolic link for a path
@AFSVOLID ⁴⁰⁸	OpenAFS volume ID for a path
@AFSVOLNAME ⁴⁰⁸	OpenAFS volume name for a path
@DOMAIN ⁴¹⁷	Domain name of a computer
@ENUMSERVERS ⁴¹⁹	Identify server names on a network
@ENUMSHARES ⁴²⁰	Identify sharenames on a server
@GROUP ⁴³⁸	User is member of group: 1, otherwise 0
@IPADDRESS ⁴⁴³	The numeric IP for a host name
@IPNAME ⁴⁴⁴	The host name for a numeric IP
@PING ⁴⁵³	Response time from a host
@WORKGROUP ⁴⁷³	Workgroup name of a computer

Utility

@ALIAS ⁴⁰⁸	Value of an alias
@CAP ⁴¹⁰	Call a _cdecl function in a DLL
@CLIP ⁴¹²	Specified line from clipboard
@CLIPW ⁴¹²	Write string to the clipboard
@COLOR ⁴¹²	RGB value of a color
@DIRSTACK ⁴¹⁶	Display directory stack entry
@ERRTEXT ⁴²⁰	Windows error description
@EXEC ⁴²⁴	Execute a command, returns its exit code
@EXECSTR ⁴²⁴	Execute a command, returns its first output line
@FUNCTION ⁴³⁷	Definition of a function
@HISTORY ⁴³⁹	A line or word from the command history
@IF ⁴³⁹	Value dependent on a conditional expression
@OPTION ⁴⁵²	Current configuration option value
@PERL ⁴⁵²	Evaluate a Perl expression
@READSCR ⁴⁵³	Read characters from the screen
@REXX ⁴⁵⁷	Evaluate a REXX¹⁴² expression
@RUBY ⁴⁵⁸	Evaluate a Ruby expression
@SCRIPT ⁴⁵⁸	Evaluate expression in active scripting engine
@SELECT ⁴⁵⁸	Menu selection
@SNAPSHOT ⁴⁶²	Save a window or the desktop to a BMP
@TIMER ⁴⁶³	Get split time from timer.
@WINAPI ⁴⁶⁷	Call a Windows API function
@WMI ⁴⁷²	Query WMI
@XMLCLOSE ⁴⁷³	Close an XML file previously opened by @XMLOPEN
@XMLNODES ⁴⁷³	Return the number of nodes (children) for the specified path in an XML file
@XMLOPEN ⁴⁷⁴	Open an XML file for use by @XMLXPath and/or @XMLNODES
@XMLXPath ⁴⁷⁴	Return text of XML element

3.14.4.3 Date Display Formats

All 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	<i>see below</i>	system default
1	mm/dd/yy	USA
2	dd/mm/yy	European
3	yy/mm/dd	Japanese

4	yyyy-mm-dd	ISO 9601
---	------------	----------

Field Order

For codes **1...4** the field order is as shown above. For code **0** the field order will also be one of those shown above. **TCC** determines which field is reported first by Windows in a short date, and selects the order from the table above with the same first field. All other aspects of the Windows short date format are ignored,

Field Width

Month and day are always 2 digits. Year is 2 digits for codes **1, 2** and **3**, and 4 digits for code **4**. For code **0** the year is 4 digits if it is the first field returned, and 2 digits if it is the last one.

Field Separator

Code **4** (ISO 9601) uses a hyphen as the separator character. For the other formats, the default Windows date separator is returned.

Setting the Windows Date Formats

The details below apply to Windows XP, but other versions of Windows are similar.

Start → Settings → Control Panel → Regional and Language Options → Customize → Date display the desired menu. The two relevant fields are **Short Date Format** and **Date Separator**.

3.14.4.4 @ABS

@ABS[*n*] : Returns the absolute value of the number *n*.

Examples:

```
echo %@abs[-1]
echo %@abs[123]
```

3.14.4.5 @AFSCCELL

@AFSCCELL[*path*] : Returns the [OpenAFS](http://www.openafs.org)^[103] cell name for the path.

See <http://www.openafs.org> for more information on OpenAFS.

3.14.4.6 @AFSMOUNT

@AFSMOUNT[*path*] : Returns the [OpenAFS](http://www.openafs.org)^[103] mount point for the pathname.

See <http://www.openafs.org> for more information on OpenAFS.

3.14.4.7 @AFSPATH

@AFSPATH[*path*] : Returns 1 if the path is in the [OpenAFS](http://www.openafs.org)^[103] file system.

See <http://www.openafs.org> for more information on OpenAFS.

3.14.4.8 @AFSSYMLINK

@AFSSYMLINK[*path*] : Returns the [OpenAFS](http://www.openafs.org)^[103] symbolic link for the path.

See <http://www.openafs.org> for more information on OpenAFS.

3.14.4.9 @AFSVOLID

@AFSVOLID[*path*] : Returns the [OpenAFS](#)^[103] volume ID for the path.

See <http://www.openafs.org> for more information on OpenAFS.

3.14.4.10 @AFSVOLNAME

@AFSVOLNAME[*path*] : Returns the [OpenAFS](#)^[103] volume name for the path.

See <http://www.openafs.org> for more information on OpenAFS.

3.14.4.11 @AGEDATE

@AGEDATE[*n*,*d*] : Converts an [age](#)^[520] *n* into a date and time pair, formatted according to the current country settings, or as explicitly specified by *d* (see [Date Display Formats](#)^[406]). The time is separated from the date by a comma, and is always in 24-hour format, displayed with 1 ms precision, as the examples show. The conversion does not take leap seconds into account.

Example:

```
for /l %n in (1,1,4) echo %n %@agedate[127551146920835000,%n]

1 03-12-05,15:24:52.083
2 12-03-05,15:24:52.083
3 05-03-12,15:24:52.083
4 2005-03-12,15:24:52.083
```

See also: [Time Stamps](#)^[495], [@FILEAGE](#)^[427] and [@MAKEAGE](#)^[449].

3.14.4.12 @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](#)^[323] /X. Otherwise, command separators, redirection characters, and other similar characters in the alias may be interpreted as part of the current command, rather than part of a simple text string.

Examples:

```
alias xyz=d:\path\myprog.exe -options
echo %@alias[xyz]
```

3.14.4.13 @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. If *filename* contains a \, @ALTNAME returns the [SFN](#)^[534] of the full path.

Examples:

```
echo %@altname["Long Name.exe"]
echo %@altname["C:\Program Files\Microsoft Office"]
echo %@altname["%CommonProgramFiles"]
```

3.14.4.14 @ASCII

@ASCII[*string*] : Returns the space separated list of ASCII values of the characters in ***string***. You can use the [Escape character](#)^[51] before a special character, e.g., a quote or greater than (>) sign, to include it in ***string***.

Note: The [@UNICODE](#)^[464] function will generally return more useful values.

Examples:

function	value
%@ascii[a]	97
%@ascii[A]	65
%@ascii[%=']	96
%@ascii[abc]	97 98 99

See also: [ASCII, Key Codes and Key Names](#)^[510].

3.14.4.15 @ASSOC

@ASSOC[*.ext*] : Returns the file association for the specified extension.

Example:

```
echo %@assoc[.doc]
```

3.14.4.16 @ATTRIB

@ATTRIB[*filename*[-*rhsadecijlopt*[*,p*]]] : If you do not specify any attributes, @ATTRIB returns the attributes of the specified file in the format **RHSADECIJNOPT**, rather than **0** or **1**. If two or more parameters are specified, @ATTRIB returns a **1** if the specified file has the matching attribute(s); otherwise it returns a **0**.

The basic attributes for 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 or symbolic link
L Junction or symbolic link
N Normal
O Offline
P Sparse file
T Temporary

The extended attributes are displayed when @ATTRIB is invoked with a single parameter, but they are suppressed when used for file selection (two or more parameters). To select files based on the

extended attributes, see [@WATTRIB](#)^[466].

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_____`. If the file does not exist, `@ATTRIB` returns an empty string.

The attributes (other than **N**) can be combined (for example `%@ATTRIB[MYFILE,HS]`). Normally `@ATTRIB` will only return **1** if all of the attributes match. However, if a final **,p** is included (partial match), then `@ATTRIB` will return **1** if any of the attributes match. For example, `%@ATTRIB[MYFILE,HS,p]` will return **1** if *MYFILE* has the hidden, system, or both attributes. Without **,p** the function will return **1** only if *MYFILE* has both attributes.

Filename must be in quotes if it contains white space or special characters.

See also: [Attributes Switches](#)^[86].

Examples:

```
echo %@attrib["C:\Program Files\My Program\myfile.exe",rhs,p]
echo Attributes for myfile.exe: %@attrib[myfile.exe]
```

3.14.4.17 @AVERAGE

@AVERAGE[...] : Returns the average of a list of numbers. The average is returned as a double; you can adjust the decimal precision by running the result through [@EVAL](#)^[420] (or [@INT](#)^[443]).

3.14.4.18 @CAPI

@CAPI[module,function[,integer | PING=n | PLONG=n | PDWORD=n | NULL | BUFFER | "string"]] : Returns the result of calling a function with a `_cdecl` type in a DLL.

module - name of the DLL containing the function

function - function name (case sensitive)

integer - an integer value to pass to the function

PINT - a pointer to the integer *n*

PLONG - a pointer to the long integer *n*

PDWORD - a pointer to the DWORD *n*

NULL - a null pointer (0)

BUFFER - `@CAPI` will pass an address for an internal buffer for the API to return a Unicode string value.

aBUFFER - `@CAPI` will pass an address for an internal buffer for the API to return an ASCII string value.

"string" - text argument (this must be enclosed in double quotes). If the argument is preceded by an 'a' (i.e., `a"Argument"`) then it is converted from Unicode to ASCII before calling the API. (Some Windows APIs only accept ASCII arguments.)

`@CAPI` supports a maximum of 8 arguments. The return value is either a string value returned by the API (if `BUFFER` or `aBUFFER` is specified), or the integer value returned by the API. The function must

be defined as `_cdecl`. If `@CAPI` can't find the specified function, it will append a "W" (for the Unicode version) to the function name and try again.

See also [@WINAPI](#)^[467].

3.14.4.19 @CAPS

@CAPS`["xxx"],text` : Capitalizes the first letter of each word in the string (words that do not start with a letter remain unchanged). The optional first parameter, **xxx**, specifies the separators that you wish to use. The list must be enclosed in double quotes. If you want to use a double quote as a separator, prefix it with the [Escape Character](#)^[124].

Examples:

```
echo %@caps[" ",i love take command]
echo %@caps[" ",,peter,paul,mary]
echo %@caps[" ^","sacrebleu!",, he said]
```

3.14.4.20 @CDROM

@CDROM`[d:]` : Returns **1** if the drive is an optical drive (CD-ROM, CD-RW, DVD, etc) or **0** otherwise. The drive letter must be followed by a colon.

Examples:

```
echo %@cdrom[C:]
echo %@cdrom[%_disk:]
```

3.14.4.21 @CEILING

@CEILING`[n]` : Returns the value of the smallest integer that is not less than **n**. `@CEILING` will perform an implicit [@EVAL](#)^[420] on its argument, so you can enter an arithmetic expression.

Examples:

```
echo %@ceiling[3.14]
echo %@ceiling[-3.14]
echo %@ceiling[0]
echo %@ceiling[123*37.36]
```

See also: [@FLOOR](#)^[435].

3.14.4.22 @CHAR

@CHAR`[n]` : Returns the character corresponding to a Unicode numeric value. If the parameter 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]
```

3.14.4.23 @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
```

3.14.4.24 @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
```

3.14.4.25 @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 *r* (red), *g* (green) and *b* (blue) parameters. If no parameters are provided, the initial selection will be black (**0,0,0**). The parameters 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]
```

3.14.4.26 @COMMA

@COMMA[*n*] : Returns the number with commas (or the appropriate [Thousands character](#)^[51] for your current country setting) inserted where appropriate.

Note: Some [variable functions](#)^[395] 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]
```

See also: [@CONVERT](#)^[413], [@FORMAT](#)^[435], [@FORMATN](#)^[436].

3.14.4.27 @COMPARE

@COMPARE[*file1,file2*] : Returns **1** if the two files are identical, or **0** if they differ. @COMPARE supports FTP filenames, but cannot compare two FTP files at the same time.

3.14.4.28 @CONSOLE

@CONSOLE[*title*] : Returns **1** if the specified window title belongs to a console window; **0** if it does not. The *title* may include [wildcards](#)^[77].

3.14.4.29 @CONVERT

@CONVERT[*input, output, value*] : Returns a numeric string *value* converted 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
```

See also: [@COMMA](#)^[412], [@FORMAT](#)^[435], [@FORMATN](#)^[436].

3.14.4.30 @COUNT

@COUNT[*c,string*] : Returns the number of times the character *c* appears in *string*.

Examples:

```
echo %@count[a,Another function example]
```

3.14.4.31 @CRC32

@CRC32[*filename*] : Returns the CRC32 value (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.

See also: [@SHA256](#)^[460], [@SHA384](#)^[460], [@SHA512](#)^[460], and [@MD5](#)^[450].

Examples:

```
echo %@crc32["C:\My Files\Myprog.exe"]
echo %@crc32["%comspec"]
```

3.14.4.32 @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.

Examples:

```
echo %@cwd[C:]
echo %@cwd[%_disk:]
```

See also: [@CWDS](#)^[414].

3.14.4.33 @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.

Examples:

```
echo %@cwds[C:]
echo %@cwds[%_disk:]
```

See also: [@CWD](#)^[414].

3.14.4.34 @DATE

@DATE[date[,format]] : Returns the number of days since January 1, 1980 for the specified date. See [date formats](#)^[127] for information on acceptable date formats. **Date** must be between 1980-01-01 and 2099-12-31 (inclusive).

@DATE accepts an optional second parameter specifying the date format:

- 0 default
- 1 USA (mm/dd/yy)
- 2 Europe (dd/mm/yy)
- 3 Japan (yy/mm/dd)
- 4 ISO (yyyy/mm/dd)

Examples:

```
echo %@date[01-01-2008]
echo %@date[%_date]
```

3.14.4.35 @DAY

@DAY[date[,format]] : Returns the numeric day of the month for the specified date. See [date formats](#)^[127] for information on acceptable date formats.

@DAY accepts an optional second parameter specifying the date format:

- 0 default
- 1 USA (mm/dd/yy)

- 2 Europe (dd/mm/yy)
- 3 Japan (yy/mm/dd)
- 4 ISO (yyyy/mm/dd)

Examples:

```
echo %@day[01-01-1980]
echo %@day[%_date]
```

3.14.4.36 @DEC

@DEC[*string*] : Returns :

- -1 if ***string*** is empty
- otherwise the same value as [@EVAL](#)^[420][*string* - 1]

If ***string*** is the name of an environment variable, its value is used whether or not it is preceded by a percent sign % without modifying the value of the variable. To actually decrement the value of the variable **var** use:

```
set var=%@dec[%var]
```

3.14.4.37 @DECIMAL

@DECIMAL[*number*]: Returns the portion of ***number*** to the right of the [Decimal character](#)^[51] as an integer numeric string. Trailing zeros are used to pad to the [Minimum Precision](#)^[51] specified for [@EVAL](#)^[420]. For example:

```
%@decimal[%@eval[1/2]]
```

is **5** if minimum width is 0, and **50000** if minimum width is 5.

@DECIMAL will perform an implicit @EVAL on its argument, so you can enter an arithmetic expression (including the @EVAL =min,max format string following the argument).

Examples:

function	value
%@decimal[1234]	0
%@decimal[1.234]	234
%@decimal[12.34]	34

3.14.4.38 @DESCRIPT

@DESCRIPT[*filename*]: Returns the file description for the specified filename (see [DESCRIBE](#)^[195]).

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"]
```

3.14.4.39 @DEVICE

@DEVICE[*name*]: Returns **1** if the specified name is a character device (such as a serial port), or **0** if not.

Examples:

```
echo %@device[lpt1]
echo %@device[com1]
echo %@device[com5]
echo %@device[clip]
```

3.14.4.40 @DIGITS

@DIGITS[*n*]: Returns **1** if the string is composed of decimal digits only, otherwise it returns **0**. The [Decimal character](#)^[51], the [Thousands character](#)^[51], 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]
```

3.14.4.41 @DIRSTACK

@DIRSTACK[*n*] : Returns the name of the *n*th entry in the directory stack. The oldest is number 0. If no *n* parameter is specified, returns the total number of entries in the stack. The directory stack is set by calls to [PUSHD](#)^[299] / [POPD](#)^[294].

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](#)^[209], [POPD](#)^[294], [PUSHD](#)^[299] and [Directory Navigation](#)^[71].

Examples:

```
echo %@dirstack[0]
echo %@dirstack[2]
echo %@dirstack[ ]
```

3.14.4.42 @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 parameter specifies the reporting scale (see [Memory Size / Disk Space / File Size Units and Report Format](#)^[395]). If the scale specification is suffixed with **c** the result will be formatted using the [thousands separator](#)^[51].

@DISKFREE supports [OpenAFS](#)^[103] names.

See also: [@DISKTOTAL](#)^[417] and [@DISKUSED](#)^[417].

Examples:

```
echo %@diskfree[c:]
echo %@diskfree[%_disk:,Kc]
```

3.14.4.43 @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 parameter specifies the reporting scale (see [Memory Size / Disk Space / File Size Units and Report Format](#)^[395]). If the scale specification is suffixed with **c** the result will be formatted using the [thousands separator](#)^[51].

@DISKTOTAL supports [OpenAFS](#)^[103] names.

See also: [@DISKFREE](#)^[416] and [@DISKUSED](#)^[417].

Examples:

```
echo %@disktotal[c:]
echo %@disktotal[%_disk:,Kc]
```

3.14.4.44 @DISKUSED

@DISKUSED[d:[,scale[c]]] : Returns the amount of disk space in use 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 disk space in use 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 parameter specifies the reporting scale (see [Memory Size / Disk Space / File Size Units and Report Format](#)^[395]). If the scale specification is suffixed with **c** the result will be formatted using the [thousands separator](#)^[51].

@DISKUSED supports [OpenAFS](#)^[103] names.

See also: [@DISKFREE](#)^[416] and [@DISKTOTAL](#)^[417].

Examples:

```
echo %@diskused[c:]
echo %@diskused[%_disk:,Kc]
```

3.14.4.45 @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.

3.14.4.46 @DOW

@DOW[date[,format]] : 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](#)^[127] for information on acceptable

date formats.

@DOW accepts an optional second parameter specifying the date format:

- 0** default
- 1** USA (mm/dd/yy)
- 2** Europe (dd/mm/yy)
- 3** Japan (yy/mm/dd)
- 4** ISO (yyyy/mm/dd)

Examples:

```
echo %@dow[01-01-1980]
echo %@dow[%_date]
```

See also: [@IDOW](#)^[439].

3.14.4.47 @DOWF

@DOWF[*date*,*format*] : Returns the full English name of the day of the week for the specified date ("Monday", "Tuesday", etc.). See [date formats](#)^[127] for information on acceptable parameter formats.

@DOWF accepts an optional second parameter specifying the date format:

- 0** default
- 1** USA (mm/dd/yy)
- 2** Europe (dd/mm/yy)
- 3** Japan (yy/mm/dd)
- 4** ISO (yyyy/mm/dd)

Examples:

```
echo %@dowf[01-01-1980]
echo %@dowf[%_date]
```

See also: [@IDOWF](#)^[439].

3.14.4.48 @DOWI

@DOWI[*date*,*format*] : Returns an integer representing the day of the week for the specified date (1 = Sunday, 2 = Monday, etc.). See [date formats](#)^[127] for information on acceptable date formats.

@DOWI accepts an optional second parameter specifying the date format:

- 0** default
- 1** USA (mm/dd/yy)
- 2** Europe (dd/mm/yy)
- 3** Japan (yy/mm/dd)
- 4** ISO (yyyy/mm/dd)

Examples:

```
echo %@dowi[01-01-1980]
echo %@dowi[%_date]
```

3.14.4.49 @DOY

@DOY[*date[,format]*] : Returns the day of year (1 - 366) for the specified date. See [date formats](#)^[127] for information on acceptable date formats.

@DOY accepts an optional second parameter specifying the date format:

- 0 default
- 1 USA (mm/dd/yy)
- 2 Europe (dd/mm/yy)
- 3 Japan (yy/mm/dd)
- 4 ISO (yyyy/mm/dd)

Examples:

```
echo %@doy[02-02-2005]
echo %@doy[%_date]
```

3.14.4.50 @DRIVETYPE

@DRIVETYPE[*drive*] : Return the type for the specified drive:

- 0 The drive type cannot be determined
- 1 The root path is invalid (no volume is mounted at the path)
- 2 Removable disk
- 3 Fixed disk
- 4 Remote (network) drive
- 5 CD-ROM
- 6 RAM disk

3.14.4.51 @DRIVETYPEEX

@DRIVETYPEEX[*drive*] : Return the type for the specified drive:

- 0 The drive type cannot be determined
- 1 The root path is invalid (no volume is mounted at the path)
- 2 Removable disk
- 3 Fixed disk
- 4 Remote (network) drive
- 5 CD-ROM
- 6 RAM disk
- 7 DVD
- 8 Tape

3.14.4.52 @ENUMSERVERS

@ENUMSERVERS[*n,server[,type]*] : Enumerate the servers on the network. *n* is the entry number in the list of servers (the first one is 0). *server* is the machine name(s) to match and it may contain [wildcards](#)^[77]. 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.

If *n* is -1, @ENUMSERVERS returns the number of matching servers.

@ENUMSERVERS takes an optional third argument to return only servers of that type. The possible types are:

WORKSTATION - All workstations.
SQLSERVER - Any server running Microsoft SQL Server
DOMAIN - Primary domain controller
DOMAINBACKUP - Backup domain controller
DOMAIN_ENUM - Primary domain
LOCAL - Servers maintained by the browser
AFP - Apple File Protocol servers
TIME - Servers running the Timesource service
PRINTQ - Server sharing print queue
TERMINAL - Terminal Servers
CLUSTER - Server clusters in the domain
VSCLUSTER - Cluster virtual servers in the domain
MASTER - Server running the master browser service

WARNING! Windows may require a significant amount of time before returning data to this function when used on large networks.

Examples:

```
echo %@enumservers[0,\\SERVER]
for %i in (0 1 2) echo %@enumservers[%i,*]
```

3.14.4.53 @ENUMSHARES

@ENUMSHARES[*n*,\\server\shares] : Enumerate the share names for the specified server. *n* is the entry number in the list of shares (the first one is **0**). **server** is the server name, and **shares** is the sharename(s) to match. **Shares** may contain [wildcards](#)^[77]. 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.

If the *n* is -1, @ENUMSHARES returns the number of matching sharenames.

Examples:

```
echo %@enumshares[0,\\SERVER\DRIVE_C]
for %i in (0 1 2) echo %@enumshares[%i,\\SERVER\DRIVE_*]
```

3.14.4.54 @ERRTEXT

@ERRTEXT[*n*] : Returns the operating system error text for the specified code.

Examples:

```
echo %@errtext[2]
echo %@errtext[255]
echo %@errtext[%_syserr]
```

3.14.4.55 @EVAL

@EVAL[*expression*[=*displayformat*]]: Evaluates a mathematical expression and returns its value in the format specified by **displayformat** or in the default format. [Parameter Interpretation](#)^[421] below describes what **expression** may contain. [Display precision and output format](#)^[423] below explains the result format.

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.

@EVAL supports very large numbers. The maximum size is 20,000 digits (10,000 digits to the left of the decimal point and 10,000 decimal places). If you want to use more than the default decimal values you'll need to change your [@Eval Precision](#)^[51] configuration options or use the "=x.y" format in @EVAL.

- ▶ [Parameter Interpretation](#)^[421]
- ▶ [Arithmetic operators](#)^[421]
- ▶ [Trigonometric and transcendental functions](#)^[422]
- ▶ [Order of precedence](#)^[423]
- ▶ [Precision of internal calculations](#)^[423]
- ▶ [Display precision and output format](#)^[423]
- ▶ [Examples](#)^[423]

Parameter Interpretation

Expression may contain environment and internal variables and variable functions. After all variables and functions have been expanded, it must be composed only of numeric strings and names of functions in [Trigonometric and transcendental functions](#)^[422], connected by [Arithmetic operators](#)^[421] and optionally grouped with parentheses.

@EVAL permits you to simplify **expression** by dropping the % percent mark in front of the names of environment variables. You must include % for internal variables and variable functions. @EVAL also permits you to use characters which normally have special meaning for **TCC** e.g., & < > ^ | without disabling their special meaning or quoting them.

Note: To ensure that **expression** is interpreted correctly, spaces should be placed on both sides of each operator, and parentheses used liberally. For example:

```
%@eval[(20 %% 3) + 4]
%@eval[12 and 65]
```

@EVAL also accepts numbers in the **e** exponent syntax; i.e. **1575e-2** = 15.75.

Number base

If a string starts with the characters **0x** it is interpreted as an integer in hexadecimal notation, and may be up to 16 digits long. Any other numeric string is considered to be a decimal number.

For example:

```
[c:\] echo %@eval[0x10 + 16]
32
```

You can specify hexadecimal output with the special syntax **@eval[...=H]**. For example:

```
echo %@eval[3*6=H]
```

will output 12 (hex). No leading 0x is included in the output. To convert between decimal and hexadecimal formats, see the [@CONVERT](#)^[413] function.

Arithmetic operators

Every operator accepts both integer and non-integer parameters, except as noted below.

Operators accepting fractional parameters

+	(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
**	exponentiation

Operators requiring integer parameters

\	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
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

Operators which truncate parameters to integer

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
~	Unary NOT

Trigonometric and transcendental functions

Expression may include the trigonometric and transcendental functions below. The argument 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 listed in [Trigonometric and transcendental functions](#)^[422]
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.

Precision of internal calculations

@EVAL supports numbers up to 20,000 digits; it is highly unlikely you'll need greater precision than this!

Display precision and output format

The maximum display precision is 10,000 digits to the left of the decimal point and 10,000 digits to the right. You can alter the default decimal precision with the [OPTION](#)^[284] command, the [@EVAL Precision](#)^[51] configuration options, and with the [SETDOS](#)^[323]/F command. You can change the decimal separator with the [decimal character](#)^[51] configuration option or the [SETDOS](#)^[323]/G command.

You can alter the display format for the current instance of @EVAL by specifying **displayformat**.

Hexadecimal display format specification

If **displayformat** is the letter **H**, output will be hexadecimal. If **displayformat** is **X**, the output will be hexadecimal with a leading **0x**.

Explicit precision specification

If **displayformat** is **i.a**, then:

- **i** must be a number which specifies the minimum decimal precision (the minimum number of decimal places displayed);
- **a** must be a number which sets the maximum decimal precision.
- the character separating **i** and **a** may be the comma if it is your decimal separator

You may specify either or both parameters **i** and **a**. If **i > a**, or if only **i** is specified, **i** is used as both the minimum and maximum precision, e.g. both **=2** and **=2.1** are equivalent to **=2.2**.

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

See also: [@DEC](#)^[415] and [@INC](#)^[440].

3.14.4.56 @EXEC

@EXEC[command] : Execute **command** and return its numeric exit code.

Command can be an alias, internal command, external command, **.BTM**, **.BAT**, or **.CMD** file.

By default, @EXEC returns the result code from **command** (see the [?](#)^[380] and [_?](#)^[380] variables). However, if in **command** you preface the command name with @ 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
```

See also: [@EXECSTR](#)^[424].

3.14.4.57 @EXECSTR

@EXECSTR[command] : Runs the specified **command** and returns the first line written to [stdout](#)^[535] by **command**.

@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:

```
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 parameter, 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.) parameter.

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.

See also: [@EXEC](#)^[424].

3.14.4.58 @EXETYPE

@EXETYPE[filename]: Returns the application type for an executable file:

Code	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" ]
```

3.14.4.59 @EXPAND

@EXPAND[\[range...\]](#) **filename**[\[, \[{+|-}\]rhsadecijopt\]](#) : Returns (in 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](#)^[77] and [include lists](#)^[88]. Returns an empty string if no files match. If the file list is longer than the allowed [command line length](#)^[126], it will be truncated without an error message. Each returned filename which contains white space or other special characters will be delimited by double quotes.

Filename must be in double quotes if it contains white space or special characters.

The **range** and attribute parameters, if included, define properties of the files that will be included in the result as specified in [File Selection](#)^[77]. Multiple **range** parameters may be included, but not more than one each of [description range](#)^[86], [size range](#)^[82], [date range](#)^[82], and [time range](#)^[84]. **Range** parameters must precede **filename**. [Exclusion ranges](#)^[85] 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**.

3.14.4.60 @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 quoted 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"]
```

3.14.4.61 @FIELD

@FIELD[\[*"sep_list"*,*n*,*string*\]](#) : Returns the *n*th field in **string**. The first field is numbered **0**. If *n* is negative, fields are counted backwards from the end of **string**. You can specify the rightmost field by setting *n* to **-0**.

You can specify a range of fields to return with the syntax:

@FIELD[\[*"sep_list"*,*start*\[-*end* | +*range*\],*string*\]](#)

Specify an inclusive range with a **-**. For example:

%@FIELD[\[2-4,A B C D E F G\]](#) will return "C D E". (Note that you cannot use inclusive ranges when starting from the end.)

You can specify a relative range with a **+**. For example:

%@FIELD[\[2+1,A B C D E F G\]](#) will return "C D".

The default list of separators for [@FIELD](#)^[426], [@FIELDS](#)^[427], [@WORD](#)^[472] and [@WORDS](#)^[473] consists of space, tab, and comma. You can use the optional first parameter, **sep_list**, to specify the separators that you wish to use. If you want to use a double quote as a separator, prefix it with an [escape character](#)^[51], e.g., **%="**. Alphabetic characters in **sep_list** are case sensitive. If you do not specify a separator list, **@FIELD** will skip any leading separators.

[@FIELD](#)^[426] and [@FIELDS](#)^[427] differ from [@WORD](#)^[472] and [@WORDS](#)^[473] in how multiple consecutive separators are counted. [@WORD](#)^[472] and [@WORDS](#)^[473] consider a sequence as a single separator, and ignore separators at either end of string. In contrast, [@FIELD](#)^[426] and [@FIELDS](#)^[427] count each occurrence of a separator individually, including those at either end of 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, e.g., **0xFFFFFFFF** for **-1**. There is no hexadecimal form to specify field **-0** (the rightmost field).

If **string** is double quoted, you must specify **sep_list**.

See also: [@WORD](#)^[472], [@WORDS](#)^[473], [@FIELDS](#)^[427].

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

3.14.4.62 @FIELDS

@FIELDS[*"sep_list",string*] : Returns the number of fields in **string**.

The default list of separators for [@FIELD](#)^[426], [@FIELDS](#)^[427], [@WORD](#)^[472] and [@WORDS](#)^[473] consists of space, tab, and comma. You can use the optional first parameter, **sep_list**, 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](#)^[51], e.g., %="". Alphabetic characters in **sep_list** are case sensitive. If you do not specify a separator list, @FIELD will skip any leading separators.

[@FIELD](#)^[426] and [@FIELDS](#)^[427] differ from [@WORD](#)^[472] and [@WORDS](#)^[473] in how multiple consecutive separators are counted. [@WORD](#)^[472] and [@WORDS](#)^[473] consider a sequence as a single separator, and ignore separators at either end of string. In contrast, [@FIELD](#)^[426] and [@FIELDS](#)^[427] count each occurrence of a separator individually, including those at either end of string.

If **string** is double quoted, you must specify **sep_list**.

See also: [@WORD](#)^[472], [@WORDS](#)^[473], [@FIELD](#)^[426].

3.14.4.63 @FILEAGE

@FILEAGE[*filename[,a|c|w|u]]* : Returns the date and time of the file as an [age](#)^[520].

Filename must be in quotes if it contains white space or special characters. The optional second parameter selects which date field is returned for files on a [VFAT](#)^[536] or [NTFS](#)^[532] drive: **a** means the last access date, **c** means the creation date, and **w** means the last modification (write) date. The default is **w**.

If you append a *u* to the second argument, @FILEAGE will display the age in UTC.

Examples:

```
echo %@fileage[d:\path\myfile.ext]
echo %@fileage["%comspec",c]
```

See also: [Time Stamps](#)^[495], [@AGEDATE](#)^[408] and [@MAKEAGE](#)^[449].

3.14.4.64 @FILECLOSE

@FILECLOSE[*n*] : Closes the file whose handle is **n**. 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](#)^[428]. If you use it with any other number you may damage other files opened by **TCC** (or by the program which started **TCC**).

See also the related handle-based functions:

@FILEOPEN ^[428]	Open a file handle
@FILEREAD ^[429]	Read next line from a file handle
@FILESEEK ^[431]	Move a file handle pointer
@FILESEEKL ^[431]	Move a file handle pointer to a specified line

[@FILEWRITE](#)^[433]
[@FILEWRITEB](#)^[433]
[@TRUNCATE](#)^[464]

Write next line to a file handle

Write data to a file handle

Truncate the file at the current position of the file handle pointer.

3.14.4.65 @FILEDATE

@FILEDATE*[filename[,a|c|w[u,d]]]* : Returns the date a file was last modified, in the default country format (mm-dd-yy for the US), or as explicitly specified by the optional third parameter **d** (see [Date Display Formats](#)^[406]). **Filename** must be in quotes if it contains white space or special characters. The optional second parameter 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.

If you append a *u* to the second argument, @FILEDATE will display the date in UTC.

Examples:

```
echo %@filedate["D:\my path\myfile.exe"]
echo %@filedate["comspec",c,4]
```

See [Time Stamps](#)^[495], [@FILETIME](#)^[432], [@FILEAGE](#)^[427].

3.14.4.66 @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"]
```

3.14.4.67 @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](#)^[429] without a **length**, and to write text using [@FILEWRITE](#)^[433]. Binary mode should be used to read binary data with [@FILEREAD](#)^[429] with a **length**, and to write binary data with [@FILEWRITEB](#)^[433]. Returns -1 if the file cannot be opened.

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](#)^[431] to position 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 if the stream name is specified. See [NTFS File Streams](#)^[496] 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 which uses @FILEOPEN. If you suspect that file handles have been opened and not closed, you should restart **TCC**.

Examples:

```
set h=%@fileopen["d:\path\myfile.txt",write]
echo writing %@filewrite[%h,this is a test] bytes
echo closing handle #h: %@fileclose[%h]
```

See also the related handle-based functions:

@FILECLOSE ^[427]	Close a file handle
@FILEREAD ^[429]	Read next line from a file handle
@FILESEEK ^[431]	Move a file handle pointer
@FILESEEKL ^[431]	Move a file handle pointer to a specified line
@FILEWRITE ^[433]	Write next line to a file handle
@FILEWRITEB ^[433]	Write data to a file handle
@TRUNCATE ^[464]	Truncate the file at the current position of the file handle pointer.

3.14.4.68 @FILEREAD

@FILEREAD^[429]**[*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](#)^[428]. If you use it with any other number you may damage other files opened by **TCC** (or by the program which started **TCC**).

Beware of characters with special meaning to **Take Command**, such as redirection and piping symbols, within the file. Use [SETDOS](#)^[323]/X with appropriate codes as needed.

See also the related handle-based functions:

@FILECLOSE ^[427]	Close a file handle
@FILEOPEN ^[428]	Open a file handle
@FILESEEK ^[431]	Move a file handle pointer
@FILESEEKL ^[431]	Move a file handle pointer to a specified line
@FILEWRITE ^[433]	Write next line to a file handle
@FILEWRITEB ^[433]	Write data to a file handle
@TRUNCATE ^[464]	Truncate the file at the current position of the file handle pointer.

3.14.4.69 @FILEREADB

@FILEREADB^[429]**[*n*,*length*]** : Reads *n* bytes of data from the file whose handle is *n*. Returns the string ****EOF**** if you attempt to read past the end of the file. The data will be returned as a string of space-separated numeric digits representing the ASCII value of each character.

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](#)^[428]. If you use it with any other number you may damage other files opened by **TCC** (or by the program which started **TCC**).

Beware of characters with special meaning to **TCC**, such as redirection and piping symbols, within the file. Use [SETDOS](#)^[323] /X with appropriate codes as needed.

See also the related handle-based functions:

@FILECLOSE ^[427]	Close a file handle
@FILEOPEN ^[428]	Open a file handle
@FILESEEK ^[431]	Move a file handle pointer
@FILESEEKL ^[431]	Move a file handle pointer to a specified line
@FILEWRITE ^[433]	Write next line to a file handle
@FILEWRITEB ^[433]	Write data to a file handle
@TRUNCATE ^[464]	Truncate the file at the current position of the file handle pointer.

3.14.4.70 @FILES

@FILES^[420]**[/S[n]] [range...] filename[, [{+|-}]rhsadecijopt]** : Returns the number of files within **range** that match **filename** and have the specified attributes. **Filename** may contain [wildcards](#)^[77] and [include lists](#)^[88]. Returns 0 if no files match. To check files in multiple directories use **@FILES** once for each, and add the results with [@EVAL](#)^[420].

Filename must be in double quotes if it contains white space or special characters.

The **range** and [attribute](#)^[86] parameters, if included, define properties of the files that will be included in the result as specified in [File Selection](#)^[77]. Multiple **range** parameters may be included, but not more than one each of [description range](#)^[86], [size range](#)^[82], [date range](#)^[82], and [time range](#)^[84]. **Range** parameters must precede **filename**. [Exclusion ranges](#)^[85] are not supported.

If you include the optional **/S** argument, **@FILES** will search the current directory and all of its subdirectories for matching files. If you specify a number after the **/S**, **@FILES** will limit the subdirectory recursion to that number. (For example, if you have a directory tree "a\b\c\d\e", /S2 will only affect the "a", "b", and "c" directories.)

If you are searching for subdirectories (i.e., by specifying "d" in the attribute argument), **@FILES** will not count the "." and ".." directory entries.

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
```

3.14.4.71 @FILESEEK

@FILESEEK[*n*,*offset*,*start*] Moves the file pointer of the file whose handle is *n* by *offset* bytes from the reference location specified via *start* (see the table below). The return value of @FILESEEK is the offset of the file pointer from the beginning of the file after the specified move. If *offset* is negative, the file pointer is moved from the reference location toward the beginning of the file. If *offset* is positive, the file pointer is moved from the reference location toward the end of the file. If *offset* is 0, the pointer is moved to the reference location.

If the function fails, the return value is -1.

<i>start</i>	<i>reference location</i>
0	beginning of file
1	current file pointer
2	end of 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](#)^[428]. If you use it with any other number you may damage other files opened by **TCC** (or by the program which started **TCC**).

Useful special cases

If you set *offset* to 0 :

- **@FILESEEK**[*n*,0,0] moves the file pointer to the beginning of file
- **@FILESEEK**[*n*,0,1] returns the current location of the file pointer without moving it.
- **@FILESEEK**[*n*,0,2] moves the file pointer to the end of file, and returns the current file size.

See *also* the related handle-based functions:

@FILECLOSE ^[427]	Close a file handle
@FILEOPEN ^[428]	Open a file handle
@FILEREAD ^[429]	Read next line from a file handle
@FILESEEKL ^[431]	Move a file handle pointer to a specified line
@FILEWRITE ^[433]	Write next line to a file handle
@FILEWRITEB ^[433]	Write data to a file handle
@TRUNCATE ^[464]	Truncate the file at the current position of the file handle pointer.

3.14.4.72 @FILESEEKL

@FILESEEKL[*n*,*line*[,1]] : Moves the file pointer to the specified *line* in the open 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 third parameter is optional, and determines the starting point for the seek. If not specified, or set to a value other than 1, @FILESEEKL starts at the beginning of the file. If set to 1, @FILESEEKL will start from the current position in the file.

If the function fails, the return value is -1.

@FILESEEKL must read each line of the file up to the target line in order to position the pointer, and can therefore cause significant delays if used in a 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](#)^[428]. If you use it with any other number you may damage other files opened by **TCC** (or by the program which started **TCC**).

See also the related handle-based functions:

@FILECLOSE ^[427]	Close a file handle
@FILEOPEN ^[428]	Open a file handle
@FILEREAD ^[429]	Read next line from a file handle
@FILESEEK ^[431]	Move a file handle pointer
@FILEWRITE ^[433]	Write next line to a file handle
@FILEWRITEB ^[433]	Write data to a file handle
@TRUNCATE ^[464]	Truncate the file at the current position of the file handle pointer.

3.14.4.73 @FILESIZE

@FILESIZE^[395][\[/S\[n\]\]](#) [\[range...\]](#) *filename*^[77][\[,scale\[c\]\]](#)^[88][\[,a\]\]](#) : Returns the size of a file, or -1 if the file does not exist. If **filename** includes [wildcards](#)^[77] or an [include list](#)^[88], it returns the combined size of all matching files. The optional third parameter **a** tells **@FILESIZE** to return the amount of space allocated for the file(s) on the disk. (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 parameter specifies the reporting scale (see [Memory Size / Disk Space / File Size Units and Report Format](#)^[395]). Adding the letter **c** requests the result be formatted using the [thousands separator](#)^[51].

The optional **range** parameter defines properties of the files that will be included in the result as specified in [File Selection](#)^[77]. Multiple **range** parameters may be included, but not more than one each of [description range](#)^[86], [size range](#)^[82], [date range](#)^[82], and [time range](#)^[84]. **Range** parameters must precede **filename**. [Exclusion ranges](#)^[85] are not supported.

If you include the optional **/S** argument, **@FILESIZE** will search the current directory and all of its subdirectories for matching files. If you specify a number after the **/S**, **@FILES** will limit the subdirectory recursion to that number. (For example, if you have a directory tree "a\b\c\d\e", **/S2** will only affect the "a", "b", and "c" directories.)

Examples:

```
echo %@filesize[d:\path\myfile.ext]
echo %@filesize["%comspec",bc]
echo %@filesize["%comspec",bc,a]
```

3.14.4.74 @FILETIME

@FILETIME^[495][\[filename\[,a\[c/w\[u\]\]\],s\]\]](#) : Returns the time of day a file was last modified, in hh:mm format. **Filename** must be in quotes if it contains white space or special characters. The optional second parameter 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 **s** as the optional third parameter. On non-NTFS drives, the last access time is always returned as 00:00, and without a seconds field (see [Time Stamp](#)^[495] for additional details).

If you append a *u* to the second argument, @FILETIME will display the time in UTC.

Examples:

```
echo %@filetime["D:\my path\myfile.exe"]
echo %@filetime["comspec",c,s]
```

See also: [@FILEDATE](#)^[428], [@FILEAGE](#)^[427].

3.14.4.75 @FILEWRITE

@FILEWRITE[*n,text*]: Writes a line to the file whose handle is *n*. Returns the number of characters written, or -1 if an error occurred. A CR/LF will be appended to *text*.

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](#)^[428]. If you use it with any other number you may damage other files opened by **TCC** (or by the program which started **TCC**).

Beware of characters with special meaning to **TCC**, such as redirection and piping symbols, within the file. Use [SETDOS](#)^[323]/X with appropriate codes as needed.

See also the related handle-based functions:

@FILECLOSE ^[427]	Close a file handle
@FILEOPEN ^[428]	Open a file handle
@FILEREAD ^[429]	Read next line from a file handle
@FILESEEK ^[431]	Move a file handle pointer
@FILESEEK ^[431]	Move a file handle pointer to a specified line
@FILEWRITEB ^[433]	Write data to a file handle
@TRUNCATE ^[464]	Truncate the file at the current position of the file handle pointer.

3.14.4.76 @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: Writes ASCII output when passed a Unicode string. Note that if you're trying to write non-English (>128) characters with @FILEWRITEB, the output will probably not match the input.

If the *length* argument is -1, @FILEWRITEB will read the string argument as a series of ASCII values in decimal or hex to write to the file. For example:

```
echo %@filewriteb[%file,-1,0xe0 0xF2 0xA9]
```

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](#)^[428]. If you use it with any other number you may damage other files opened by **TCC** (or by the program which started **TCC**).

Beware of characters with special meaning to **TCC**, such as redirection and piping symbols, within the file. Use [SETDOS](#)^[323]/X with appropriate codes as needed.

See also the related handle-based functions:

@FILECLOSE ^[427]	Close a file handle
@FILEOPEN ^[428]	Open a file handle
@FILEREAD ^[429]	Read next line from a file handle
@FILESEEK ^[431]	Move a file handle pointer
@FILESEEK ^[431]	Move a file handle pointer to a specified line
@FILEWRITE ^[433]	Write next line to a file handle
@TRUNCATE ^[464]	Truncate the file at the current position of the file handle pointer

3.14.4.77 @FINDCLOSE

@FINDCLOSE^[434]*[filename]*: Signals the end of a [@FINDFIRST](#)^[434] ... [@FINDNEXT](#)^[435] sequence. You must use this function to release the directory search handle. **Filename** is unnecessary, this function can be simply called as %**@FINDCLOSE**[] without parameters. **@FINDCLOSE** returns 0 if a [@FINDFIRST](#)^[434] ... [@FINDNEXT](#)^[435] sequence is in effect, a non-zero value otherwise.

Examples:

```
echo %@findfirst[*.exe]
echo %@findclose[]
```

3.14.4.78 @FINDFIRST

@FINDFIRST^[77]*[range... filename[, [+|-]rhsadecijopt]* : Returns the name of the first file that matches **filename**, which may include [wildcards](#)^[77] and/or an [include list](#)^[88], and which file has the properties specified in the optional [range](#)^[80] and [attribute](#)^[86] parameters.

Filename must be in quotes if it contains white space or special characters.

The **range** and attribute parameters, if included, define properties of the files that will be included in the search as specified in [File Selection](#)^[77]. Multiple **range** parameters may be included, but not more than one each of [size range](#)^[82], [date range](#)^[82], and [time range](#)^[84]. **Range** parameters must precede **filename**. Each **range** parameter is of the form

```
/[a...]
```

where **a** is one of **d**, **s** or **t**, optionally followed by range parameters.

On an LFN drive, the returned filename may contain white space or other special characters. Unlike [@EXPAND](#)^[425], no double quotes 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](#)^[395] for additional details.

@FINDFIRST[] locates the *first* file matching the requirements. To find more matching files, you must use [@FINDNEXT](#)^[435], and terminate the search with [@FINDCLOSE](#)^[434].

Warning: **@FINDFIRST** searches may not be nested!

Examples:

```
%@findfirst[/[d-30] *]
    locate files created no more than 30 days ago

%@findfirst[/[s2k,3k] "%windir\*.exe",a]
```


locate files with the extension **exe**, the archive flag set, and at least 2,000 bytes but not more than 3,000 bytes long, in the Windows directory.

3.14.4.79 @FINDNEXT

@FINDNEXT[\[filename\[, \[/\[-\]rhsadecijopt\]\]](#): 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](#)[\[434\]](#).

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 the file attributes for @FINDNEXT. **Filename**, if used, must be in quotes if it contains white space or special characters.

The attribute parameter, if included, defines the attributes of the files that will be included in the search as specified in [Attribute Switches](#)[\[86\]](#).

Range parameters may not be used in this function. The **range** parameters 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 parameters.

On an LFN drive, the returned filename may contain white space or other special characters. Unlike [@EXPAND](#)[\[425\]](#), no double quotes 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](#)[\[395\]](#) for additional details.

@FINDFIRST[] locates the first file matching the requirements. To find more matching files, you must use [@FINDNEXT](#)[\[435\]](#), and terminate the search with [@FINDCLOSE](#)[\[434\]](#).

Examples:

```
echo %@findfirst[*]
echo %@findnext[]
echo %@findnext[* ,d]
echo %@findclose[]
```

3.14.4.80 @FLOOR

@FLOOR[\[n\]](#): Returns the largest integer that is not greater than **n**. @FLOOR will perform an implicit [@EVAL](#)[\[420\]](#) on its argument, so you can enter an arithmetic expression.

Examples:

```
echo %@floor[3.14]
echo %@floor[-3.14]
echo %@floor[0]
echo %@floor[123]
```

See also: [@CEILING](#)[\[417\]](#).

3.14.4.81 @FORMAT

@FORMAT[\[format,string\]](#) : Reformats **string**, truncating it or padding it with spaces or zeros as necessary. **format** is of the format **[-]i.a**. If the optional minus sign is present, the result is left justified;

otherwise it is right justified. If *i* is specified, and its first digit is **0**, the padding character will be **0**, otherwise it will be a space. *i* is the minimum number of characters in the result, *a* is the maximum number of characters. If *a* is less than *i*, it will be ignored.

Examples

function	value
"%@format[7,Hello]"	" Hello"
"%@format[.3,Hello]"	"Hel"
"%@format[4,5]"	" 5"
"%@format[04,5]"	"0005"
"%@format[-04,5]"	"5000"

See also: [@COMMA](#)^[412], [@CONVERT](#)^[413], [@FORMATN](#)^[436].

3.14.4.82 @FORMATN

@FORMATN*[-]width[.precision],value* : Formats 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 may not round the number appropriately. (For example, @FORMATN[3,3.4] will produce "3.3999999999999999".)

See also: [@COMMA](#)^[412], [@CONVERT](#)^[413], [@FORMAT](#)^[435], [@FORMATNC](#)^[436].

3.14.4.83 @FORMATNC

@FORMATNC*[-]width[.precision],value* : Formats a numeric value and automatically inserts the thousands separator. **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.

See also: [@COMMA](#)^[412], [@CONVERT](#)^[413], [@FORMAT](#)^[435], [@FORMATN](#)^[436].

3.14.4.84 @FSTYPE

@FSTYPE*[d:]* : Returns the file system type for the specified drive or sharename. @FSTYPE returns **NTFS** for a drive that uses the Windows NTFS file system. It returns **FAT32** for FAT32 drives, and **FAT** for FAT12, FAT16, and VFAT drives.

You can specify either a drive name or a UNC name.

Examples:

```
echo %@fstype[c:]
echo %@fstype[%_disk:]
echo %@fstype[\\Music\iTunes]
```

3.14.4.85 @FTYPE

@FTYPE[xxx] : Returns the open command string for the specified file type.

Examples:

```
echo %@ftype[PerlScriptFile]
```

See also [@ASSOC](#)^[409] and [FTYPE](#)^[241].

3.14.4.86 @FULL

@FULL[filename] : Returns the full path and filename of a file. **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](#)^[395] for additional details.

Note: The @FULL function makes no assumption about the existence of a file or directory. The **filename** parameter 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[xyz.abc]
echo "%@full[.]"
echo "%@full["\Program Files"]"
```

3.14.4.87 @FUNCTION

@FUNCTION[name] : Returns the definition of the specified [user-defined function](#)^[242] **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](#)^[323]. 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](#)^[242] command.

3.14.4.88 @GETDIR

@GETDIR[d:\path[,title]] : 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.

d:\path must be in quotes if it contains white space or special characters. On an LFN drive, the returned path may contain white space or other special characters. To avoid problems which could be caused by these characters, quote the returned path before you pass it to other commands. See the notes under [Variable Functions](#)^[395] for additional details.

@GETDIR accepts an optional second parameter to set the title of the dialog box.

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](#) ^[438].

3.14.4.89 @GETFILE

@GETFILE[*d:\path\filename[,filter[,title]]*]: 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 parameter 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 parameters must be in quotes if they contain 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](#) ^[395] for additional details.

@GETFILE accepts an optional third parameter to set the title of the dialog box.

@GETFILE is still maintained for backwards compatibility, but it has been replaced by the more powerful [@GETFOLDER](#). ^[438]

Examples:

```
echo %@getfile[*]
echo %@getfile["%windir",*.exe;*.com]
```

3.14.4.90 @GETFOLDER

@GETFOLDER[*startdir[,title]*]: 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 Windows configuration.

The optional second argument sets the text to display above the tree view.

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](#) ^[437].

3.14.4.91 @GROUP

@GROUP[*server,group,user*]: Returns 1 if *user* is a member of the specified *group*. *server* specifies the DNS or NetBIOS name of the computer on which the function is to execute.

3.14.4.92 @HISTORY

@HISTORY*[x[,y]]* : Returns a line or word from the [command history](#)^[106]. (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, @HISTORY 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).

3.14.4.93 @IDOW

@IDOW*[date[,format]]* : Returns the 3-character abbreviation for the day of the week for the specified date, in the current locale language. See [date formats](#)^[127] for information on date formats.

@IDOW accepts an optional second parameter specifying the date format:

- 0 default
- 1 USA (mm/dd/yy)
- 2 Europe (dd/mm/yy)
- 3 Japan (yy/mm/dd)
- 4 ISO (yyyy/mm/dd)

Examples:

```
echo %@idow[01-01-1980]
echo %@idow[%_date]
```

See also: [@DOW](#)^[417].

3.14.4.94 @IDOWF

@IDOWF*[date[,format]]* : Returns the full name for the day of the week for the specified date, in the current locale language. See [date formats](#)^[127] for information on date formats.

@IDOWF accepts an optional second parameter specifying the date format:

- 0 default
- 1 USA (mm/dd/yy)
- 2 Europe (dd/mm/yy)
- 3 Japan (yy/mm/dd)
- 4 ISO (yyyy/mm/dd)

Examples:

```
echo %@idowf[01-01-2007]
echo %@idowf[%_date]
```

See also: [@DOWF](#)^[418].

3.14.4.95 @IF

@IF*[condition,string1,string2]*: Evaluates **condition** according to the rules described in [Conditional Expressions](#)^[109], and if **true**, it returns **string1**, otherwise it returns **string2**. Leading and trailing white space in **string1** and **string2** is retained. Either string may be empty or contain white space only.
WARNING: Both **string1** and **string2** are evaluated whether or not used. Do not use @IF if evaluating either one of the strings may fail; use the **IF** or **IFF** command instead.

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.

- 3) Assuming **A** and **C** are files in the current directory, but **B** is a subdirectory, the command:

```
for %x in (A B C) echo "%x" is %@if[isfile "%x", ,not] a file
```

will display

```
"A" is      a file
"B" is not  a file
"C" is      a file
```

3.14.4.96 @INC

@INC[*string*] returns

- 1 if ***string*** is empty
- otherwise the same value as [@EVAL](#)⁴²⁰[***string*** + 1]

If ***string*** is the name of an environment variable, its value is used whether or not it is preceded by a percent mark % without modifying the value of the variable. To actually increment the value of the variable **var** use

```
set var=%@inc[%var]
```

3.14.4.97 @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 or it is prefixed with the minus sign -, 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.

Tips

- searching for a **comma** :

1. quote ***string1*** (to prevent the expected comma making it appear as more than one parameter)

2. use [escape character](#)^[51] or its pseudovisible form %= in **string2** to escape the comma
`echo %@index["4NT, Take Command, 4DOS",%,,2]`

- searching for a **double quote** :

1. use [escape character](#)^[51] or its pseudovisible form %= in **string2** to escape the double quote
2. use the special form %=q to represent it in **string2**:
`echo %@index[contains a "quoted" word,%=q,0]`

See [Codes for Escapable Characters](#)^[124] for details.

Examples:

In all examples below

- **string1:** This is a fine help file
- **string2:** h

<i>n</i>	<i>result</i>	<i>purpose</i>
omitted	1	locate leftmost
0	2	count occurrences
1	1	locate leftmost
2	15	locate second leftmost
3	-1	locate third leftmost
-1	15	locate rightmost
-2	1	locate second rightmost
-3	-1	locate third rightmost

3.14.4.98 @INIREAD

@INIREAD[file,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. 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*.

3.14.4.99 @INIWRITE

@INIWRITE[file,section,entry,string]: Creates, updates, or deletes an entry in the specified **file**. 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 8,192 bytes.

```
echo %@iniwrite[c:\tcmd\tcmd32.ini,TakeCommand,history,]
```

will remove the **history** entry from the file.

3.14.4.10 @INODE

@INODE[filename] : Returns the inode (in hex) for the specified file.

When files are hard-linked to one another (see [MKLNK](#)^[273]), they share the same inode.

3.14.4.10 @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 value is used to count backward from 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 [escape character](#)^[51].

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 **offset**, remember to use 32-bit 2's complement arithmetic.

Examples:

function	value
%@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

3.14.4.10:@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 its length is specified by the value of **length** without the minus sign. [@SUBSTR](#)^[463] 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:

function	value
=%@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=

3.14.4.10:@INT

@INT[*n*]: Returns the integer part of the number **n**. **@INT** will perform an implicit [@EVAL](#)^[420] on its argument, so you can use an arithmetic expression for **n**.

Examples:

```
echo %@int[1234]
echo %@int[1.234]
echo %@int[12.34]
```

3.14.4.10:@IPADDRESS

@IPADDRESS[*hostname*]: Returns the numeric IP address for the specified **hostname**. If **hostname** is omitted, returns the current local IP address. The result is displayed in the standard format **nnn.nnn.nnn.nnn**. An invalid or unknown **hostname** will return an error (see [@ERREXT](#)^[420] to decipher the error number if necessary).

See also [@IPNAME](#)^[444].

Example:

```
echo %@ipaddress[jpsoft.com]
echo %@ipaddress[]
```


3.14.4.10!@IPNAME

@IPNAME[*numeric_IP*]: Returns the host name for the specified **numeric_IP** address. An IP address **0** returns the name of the current local host (usually the computer name). The IP address can be expressed in the common format **nnn.nnn.nnn.nnn** or as a packed decimal. An invalid or unknown IP address returns an error (see [@ERRTEXT](#)^[420] to decipher the error number if necessary).

See also: [@IPADDRESS](#)^[443].

Example:

```
echo %@ipname[192.220.109.228]
echo %@ipname[3235671524]
echo %@ipaddress[0]
```

3.14.4.10!@ISALNUM

@ISALNUM[*string*]: Returns **1** if **string** is entirely composed of alphabetic (a-z, A-Z) and/or numeric (0 - 9) characters; **0** otherwise.

See also: [@ISALPHA](#)^[444], [@ISASCII](#)^[444], [@ISCNTRL](#)^[445], [@ISDIGIT](#)^[445], [@ISPRINT](#)^[445], [@ISPUNCT](#)^[445], [@ISSPACE](#)^[446], [@ISXDIGIT](#)^[446].

Example:

```
echo %@isalnum[123abc]
echo %@isalnum[123 abc]
echo %@isalnum[123.456]
```

3.14.4.10!@ISALPHA

@ISALPHA[*string*]: Returns **1** if **string** is entirely composed of alphabetic (a-z, A-Z) characters; **0** otherwise.

See also: [@ISALNUM](#)^[444], [@ISASCII](#)^[444], [@ISCNTRL](#)^[445], [@ISDIGIT](#)^[445], [@ISPRINT](#)^[445], [@ISPUNCT](#)^[445], [@ISSPACE](#)^[446], [@ISXDIGIT](#)^[446].

Example:

```
echo %@isalpha[abc]
echo %@isalpha[ABC]
echo %@isalpha[A B C]
```

3.14.4.10!@ISASCII

@ISASCII[*string*]: Returns **1** if **string** is entirely composed of 7-bit ASCII characters (0x00 – 0x7F); **0** otherwise.

See also: [@ISALNUM](#)^[444], [@ISALPHA](#)^[444], [@ISCNTRL](#)^[445], [@ISDIGIT](#)^[445], [@ISPRINT](#)^[445], [@ISPUNCT](#)^[445], [@ISSPACE](#)^[446], [@ISXDIGIT](#)^[446].

Example:

```
echo %@isascii[abc]
echo %@isascii[abc 123]
echo %@isascii["abc",123]
```

3.14.4.10:@ISCNTRL

@ISCNTRL[*string*]: Returns **1** if ***string*** is entirely composed of ASCII control characters (0x00 – 0x1F or 0x7F); **0** otherwise.

See also: [@ISALNUM](#)^[444], [@ISALPHA](#)^[444], [@ISASCII](#)^[444], [@ISDIGIT](#)^[445], [@ISPRINT](#)^[445], [@ISPUNCT](#)^[445], [@ISSPACE](#)^[446], [@ISXDIGIT](#)^[446].

3.14.4.11:@ISDIGIT

@ISDIGIT[*string*] : Returns **1** if ***string*** is entirely composed of decimal digits (0– 9); **0** otherwise.

See also: [@ISALNUM](#)^[444], [@ISALPHA](#)^[444], [@ISASCII](#)^[444], [@ISCNTRL](#)^[445], [@ISPRINT](#)^[445], [@ISPUNCT](#)^[445], [@ISSPACE](#)^[446], [@ISXDIGIT](#)^[446].

Example:

```
echo %@isdigit[0]
echo %@isdigit[123.456]
echo %@isdigit[-123]
```

3.14.4.11:@ISPRINT

@ISPRINT[*string*]: Returns **1** if ***string*** is entirely composed of printable characters; **0** otherwise.

See also: [@ISALNUM](#)^[444], [@ISALPHA](#)^[444], [@ISASCII](#)^[444], [@ISCNTRL](#)^[445], [@ISDIGIT](#)^[445], [@ISPUNCT](#)^[445], [@ISSPACE](#)^[446], [@ISXDIGIT](#)^[446].

3.14.4.11:@ISPROC

@ISPROC[*pid*] : Returns 1 if the specified process ID is an active process, or 0 if it is not.

3.14.4.11:@ISPUNCT

@ISPUNCT[*string*]: Returns **1** if ***string*** is entirely composed of punctuation characters, i.e. printable characters which are not alphanumeric or space; **0** otherwise.

See also: [@ISALNUM](#)^[444], [@ISALPHA](#)^[444], [@ISASCII](#)^[444], [@ISCNTRL](#)^[445], [@ISDIGIT](#)^[445], [@ISPRINT](#)^[445], [@ISSPACE](#)^[446], [@ISXDIGIT](#)^[446].

Example:

```
echo %@ispunct[.]
echo %@ispunct[+]
echo %@ispunct[:-)]
```

3.14.4.11!@ISSPACE

@ISSPACE[*string*]: Returns **1** if ***string*** is entirely composed of white space characters (0x09 – 0x0D or 0x20); **0** otherwise.

See also: [@ISALNUM](#)^[444], [@ISALPHA](#)^[444], [@ISASCII](#)^[444], [@ISCNTRL](#)^[445], [@ISDIGIT](#)^[445], [@ISPRINT](#)^[445], [@ISPUNCT](#)^[445], [@ISXDIGIT](#)^[446].

3.14.4.11!@ISXDIGIT

@ISXDIGIT[*string*]: Returns **1** if ***string*** is entirely composed of hexadecimal digits (0–9, A–F, a–f); **0** otherwise.

See also: [@ISALNUM](#)^[444], [@ISALPHA](#)^[444], [@ISASCII](#)^[444], [@ISCNTRL](#)^[445], [@ISDIGIT](#)^[445], [@ISPRINT](#)^[445], [@ISPUNCT](#)^[445], [@ISSPACE](#)^[446].

Example:

```
echo %@isxdigit[0]
echo %@isxdigit[7F]
echo %@isxdigit[0x7F]
```

3.14.4.11!@JUNCTION

@JUNCTION[*dir*] : Returns the directory referenced by the specified junction.

3.14.4.11!@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](#)^[360].

3.14.4.11!@LCS

@LCS[*string1*,*string2*] : Returns a pointer to the Longest Common Sequence in ***string1*** and ***string2***.

3.14.4.11!@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.

Examples:

```
%@LEFT[2,jpsoft]      jp
%@LEFT[22,jpsoft]     jpsoft
%@LEFT[-2,jpsoft]      jpso
%@LEFT[-22,jpsoft]     empty string
```

3.14.4.12:@LEN

@LEN[*string*] : Returns the length of ***string***.

Examples:

```
echo %@len[this is a test]
echo %@len[%comspec]
```

3.14.4.12:@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](#)^[395] for additional details.

Examples:

```
echo %@lfn[xyz.abc]
echo "%@lfn[c:\progra~1]"
```

3.14.4.12:@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 [DO](#)^[210] command, or [@FILEOPEN](#)^[428] and a sequence of [@FILEREADS](#).^[429]

You can retrieve input from standard input if you specify **CON** as the filename. If you are [redirecting](#)^[98] input to @LINE using this feature, you must use [command grouping](#)^[127] or the redirection will not work

properly (you can [pipe](#)^[101] 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 if the stream name is specified. See [NTFS File Streams](#)^[496] for additional details on file streams.

Beware of characters with special meaning to **TCC**, such as redirection and piping symbols, within the file. Use [SETDOS](#)^[323]/X with appropriate codes as needed.

3.14.4.12:@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 if the stream name is specified. See [NTFS File Streams](#)^[496] for additional details on file streams.

3.14.4.12:@LINKS

@LINKS[filename] : Returns the number of hard links for the specified file (NTFS only).

See also [MKLNK](#)^[273].

3.14.4.12:@LOWER

@LOWER[string] : Returns the **string** converted to lower case.

Examples:

```
echo %@lower[ThiS iSS aTeSt]
echo %@lower[%path]
```

3.14.4.12:@LTRIM

@LTRIM[string1,string2] : - Returns **string2** with all the leading characters in **string1** removed. **String1** must be enclosed in double quotes if it contains any spaces, tabs, or commas.

Examples:

```
echo "%@ltrim[JP,JP Software]"
" Software"
```

3.14.4.12!@MAKEAGE

@MAKEAGE[*date*[,*time*[,*format*]]] : Converts **date** and **time** (if included) to an [age](#)^[520], a single value in the same format as [@FILEAGE](#)^[427].

@MAKEAGE accepts an optional third parameter specifying the date format:

- 0 default
- 1 USA (mm/dd/yy)
- 2 Europe (dd/mm/yy)
- 3 Japan (yy/mm/dd)
- 4 ISO (yyyy/mm/dd)

@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](#)^[408] is the inverse of this function.

Examples:

```
echo %@makeage[%_date]
echo %@makeage[%_date,%_time]
```

See also: [Time Stamps](#)^[495], [@FILEAGE](#)^[427], [@AGEDATE](#)^[408].

3.14.4.12!@MAKEDATE

@MAKEDATE[*n*[,*d*]]: Returns a date, formatted according to the current country settings, or as explicitly specified by *d* (see [Date Display Formats](#)^[406]). *n* is interpreted as the number of days since 1980-01-01, and must be in the range 0 to 43829 (corresponding to the date 2099-12-31). This is function is the inverse of [@DATE](#)^[414]. The optional second parameter specifies the 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 %@makedate[7924]
echo %@makedate[7924,4]
```

3.14.4.12!@MAKETIME

@MAKETIME[*n*] : Returns a time (formatted using the Time Separator specified in Regional Settings). *n* is interpreted as the number of seconds since midnight, and must not exceed 86399. This function is the inverse of [@TIME](#)^[463].

Examples:

```
echo %@maketime[45240]
echo %@maketime[79244]
```

3.14.4.13:@MAX

@MAX[*a,b,c,...*]: Returns the largest in the list of parameters. All parameters must be integers in the range **-2147483647** to **2147483647** and must be separated either by whitespace or by commas.

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]
5
```

3.14.4.13:@MD5

String mode: **@MD5**[*s,string*]
 File mode: **@MD5**[*f,filename*]

Returns the 32 hexadecimal digit MD5 hash of the character in **string** or of the contents of the file **filename**. The first parameter must be **s** for a string, and any leading or trailing whitespace characters in **string** are included.

Filename may be specified with or without an optional **f**. @MD5 returns **-1** if the file does not exist, or it cannot be read.

Since **Take Command** handles all internal strings as Unicode, @MD5 will return different results for a string and the identical string in an ASCII file.

See also: [@SHA256](#)^[460], [@SHA384](#)^[460], [@SHA512](#)^[460], and [@CRC32](#)^[413].

3.14.4.13:@MIN

@MIN[*a,b,c,...*] : Returns the smallest in the list of parameters. All parameters must be integers in the range **-2147483647** to **2147483647** and must be separated either by whitespace or by commas.

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]
-1
```

3.14.4.13:@MONTH

@MONTH[*date[,format]*] : Returns the month number for the specified date (1-12). See [date formats](#)^[127] for information on acceptable date formats.

@MONTH accepts an optional second parameter specifying the date format:

- 0** default
- 1** USA (mm/dd/yy)
- 2** Europe (dd/mm/yy)
- 3** Japan (yy/mm/dd)
- 4** ISO (yyyy/mm/dd)

Examples:

```
echo %@month[01-01-1980]
echo %@month[%_date]
```

3.14.4.13!@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](#)^[395] for additional details.

Note: The @NAME function makes no assumption about the existence of a file or directory. Its **filename** parameter can be any string and the function will attempt to extract from it a base name.

Examples:

```
echo %@name[xyz.abc]
echo "%@name[%_comspec]"
```

3.14.4.13!@NUMERIC

@NUMERIC[*[+/-]string*] : Returns **1** if **string** is numeric, and **0** otherwise.

To be numeric, the following must be true:

1. The first character may be a + or - sign,
2. The next character must be a decimal digit (0 to 9) or the [decimal separator](#)^[51].
3. The remainder of **string** must be composed entirely of decimal digits (0 to 9), the [thousands separator](#)^[51], and no more than a single decimal separator, with no thousands separators following the decimal separator.

Examples:

function	value
%@numeric[12345]	1
%@numeric[-12345]	1
%@numeric[.12345]	1
%@numeric[\$12.34]	0
%@numeric[5.00.125]	0
%@numeric[+5.00.125,5]	0
%@numeric[.00.125]	0
%@numeric[-5,.00.125]	0

3.14.4.13!@OPTION

@OPTION*[directive]* : Returns the current value of the requested configuration option. All directives which can be altered dynamically are supported. If **directive** is not supported, an error is returned.

For configuration directives, the current value returned may not match that stored in the *.INI* file.

For color directives, the current value is returned as a single number (0-255) combining foreground and background specifications. See [Colors, Color Names & Codes](#)^[518] for details.

Examples:

```
echo %@option[passiveftp]
echo %@option[stdcolors]
```

3.14.4.13!@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** parameter doesn't contain path information, you may expand it first with the [@FULL](#)^[437] 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](#)^[395] for additional details.

Note: The @PATH function makes no assumption about the existence of a file or directory. Its **filename** parameter can be any string, and the function will attempt to remove from it a "base name".

Examples:

command	result
echo "%@path["c:\program files\xyz.abc"]	"c:\program files\"
echo "%@path[xyz.abc]"	" "
echo "%@path[%_comspec]"	"c:\jpssoft\rlsdu\"

3.14.4.13!@OWNER

@OWNER*[filename]*: Returns the owner of the specified file (if any).

3.14.4.13!@PERL

@PERL*[expression]* : Executes the specified Perl expression (requires Active State Perl 5.8).

3.14.4.14:@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 **-2**. An invalid address returns **-3**.

Examples:

```
echo %@ping[microsoft.com]
echo %@ping[microsoft.com,10]
echo %@ping[microsoft.com,,16]
echo %@ping[192.168.1.100,2,512]
```

3.14.4.14:@QUOTE

@QUOTE[*string*] : Returns a double quoted argument if it contains any whitespace characters.

3.14.4.14:@RANDOM

@RANDOM[*min*, *max*]: Returns a pseudo random integer value between **min** and **max**, inclusive. The random number generator is initialized from the system clock the first time it is used after **TCC** starts and will therefore produce a different sequence of numbers each time you use it.

Examples:

```
echo %@random[0,1]
echo %@random[-10,10]
```

3.14.4.14:@READSCR

@READSCR[*row,col,length*]: Returns the text displayed in the **TCC** 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.

3.14.4.14:@READY

@READY[*d*]: Returns **1** if the specified drive is ready; otherwise returns **0**. The drive letter must be followed by a colon.

@READY does not support UNC names.

Examples:

```
echo %@ready[E:]
echo %@ready[%_boot:]
```

3.14.4.14!@REGCREATE

@REGCREATE[*HKEY... \subkey*]: Create a new registry subkey. The parameter 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 parameter (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.

See also: [@REGQUERY](#)^[455] (read a value), [@REGSET](#)^[455] (write a value), and [@REGSETENV](#)^[455] (write and broadcast a value).

3.14.4.14!@REGDELKEY

@REGDELKEY[*HKEY... key*]: Deletes the specified key and all of its subkeys. Returns **1** if the key was deleted, **0** otherwise. The expression must be enclosed in double quotes if it contains any separator characters (space, comma, or tab).

Note: use EXTREME caution with this function. It has the potential for causing irreparable damage to your registry and can even prevent Windows from booting!

See [@REGCREATE](#)^[454] for information on the format of the key name.

3.14.4.14!@REGEX

@REGEX[*expression, string*]: Returns the number of matching groups in the string. If no groups are specified, it returns **1** if the **expression** was found and **0** if it was not. The expression must be enclosed in double quotes if it contains any separator characters (space, comma, or tab). See [Regular Expression Syntax](#)^[496] for supported expressions.

3.14.4.14!@REGEXINDEX

@REGEXINDEX[*expression, string*]: Returns the offset of the first match. The **expression** must be enclosed in double quotes if it contains any separator characters (space, comma, or tab). See [Regular Expression Syntax](#)^[496] for supported expressions. (This function is basically a wildcard-enabled @INDEX.)

3.14.4.14!@REGEXIST

@REGEXIST[*HKEY... key*]: Returns **1** if the specified key exists, **0** otherwise

The expression must be enclosed in double quotes if it contains any separator characters (space, comma, or tab).

See [@REGCREATE](#)^[454] for information on the format of the key name.

3.14.4.15:@REGEXSUB

@REGEXSUB[*n,expression,string*] - returns the "nth" matching group in the string. (If you don't specify a group in **expression**, @REGEXSUB will return an empty string.) The expression must be enclosed in double quotes if it contains any separator characters (space, comma, or tab). See [Regular Expression Syntax](#)^[496] for supported expressions.

3.14.4.15:@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](#)^[454] (create a subkey) for information on the format of the key name. See also: [@REGSET](#)^[455] (write a value) and [@REGSETENV](#)^[455] (write and broadcast a value).

3.14.4.15:@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 parameter 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 or commas!

See [@REGCREATE](#)^[454] for information on the format of the key name. See also: [@REGQUERY](#)^[455] (read a value) and [@REGSETENV](#)^[455] (write and broadcast a value).

3.14.4.15:@REGSETENV

@REGSETENV[*HKEY... \subkey\value,type,data*] : The same as [@REGSET](#)^[455], 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](#)^[454] for information on the format of the key name. See also: [@REGQUERY](#)^[455] (read a value) and [@REGSET](#)^[455] (write a value).

3.14.4.15:@REGTYPE

@REGTYPE[*HKEY... \key*] : Returns the registry variable type. The possible values are:

0 - REG_NONE (No value type)

- 1 - REG_SZ (Unicode null terminated string)
- 2 - REG_EXPAND_SZ (Unicode null terminated string with environment variable references)
- 3 - REG_BINARY (Free form binary)
- 4 - REG_DWORD (32-bit number)
- 5 - REG_DWORD_BIG_ENDIAN (32-bit number)
- 6 - REG_LINK (Symbolic Link)
- 7 - REG_MULTI_SZ (Multiple Unicode strings)
- 8 - REG_RESOURCE_LIST (Resource list in the resource map)
- 9 - REG_FULL_RESOURCE_DESCRIPTOR (Resource list in the hardware description)
- 10 - REG_RESOURCE_REQUIREMENTS_LIST
- 11 - REG_QWORD (64-bit number)

3.14.4.15 @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]
```

3.14.4.15 @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]
```

3.14.4.15 @REPEAT

@REPEAT[char,count]: Returns the character **char** repeated **count** times (**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:

function	value
%@repeat[%char[95],10]	
7%@repeat[,7]spaces	7 spaces
%@repeat[x,10]	xxxxxxxxxx

3.14.4.15 @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]
```

3.14.4.15:@REVERSE

@REVERSE[*string*] : Reverses the order of the characters in **string**.

3.14.4.16:@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 **TCC** expands variables and functions before passing **expr** to REXX.

Examples:

```
echo %@rexx[ = 3 * 4 ]
set myprog=d:\path\xyz.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](#)^[142] for more information on the REXX language.

3.14.4.16:@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.

Examples:

function	value
%@RIGHT[2,jpsoft]	ft
%@RIGHT[22,jpsoft]	jpsoft
%@RIGHT[-2,jpsoft]	soft
%@RIGHT[-22,jpsoft]	empty string

3.14.4.16:@RTRIM

@RTRIM[*string1,string2*]: - Returns **string2** with any characters in **string1** removed from the right side of **string2**. **String1** must be enclosed in double quotes if it contains any spaces, tabs, or commas.

Examples:

```
echo "%@rtrim[98XP,Windows XP]"
"Windows "
```

3.14.4.16:@RUBY

@RUBY[*expression*] : Returns the string result of the Ruby expression. Note that the Ruby environment is persistent within a **TCC** tab window, so you can do things like:

```
%@ruby[b = 42]
%@ruby[p b]
```

which will print "42". The value returned by @RUBY is the value returned by the RUBY API `rb_eval_string`.

You can query the type of the value returned by the last @RUBY call with the [RUBYTYPE](#)^[391] internal variable, and the value returned by the last @RUBY call with the [RUBYVALUE](#)^[391] internal variable.

3.14.4.16:@SCRIPT

@SCRIPT[*engine,expression*] : Returns the integer result of expression in the specified active scripting engine. For example:

```
%@script[PerlScript,print "This message is from Perl!"]
```

See also the [SCRIPT](#)^[311] command.

3.14.4.16:@SEARCH

@SEARCH[*program[,path[,n]]*] : Searches for **program** using the specified **path**, or, if not specified, the **PATH** environment variable, appending an extension if one isn't specified. (See [Executable Files and File Searches](#)^[504] for details on the default extensions used when searching **PATH**, the order in which the search proceeds, and the search of the **WINDOWS** and **WINDOWS\SYSTEM** directories.) Returns the fully expanded name of **program**, including drive, path, base name, and extension, or an empty string if a match is not found. If [wildcards](#)^[77] are used in the **program**, @SEARCH will search for the first program file that matches the wildcard specification, and returns the drive and path for that file plus the wildcard filename (e.g., `E:\UTIL*.COM`).

Program and each directory specification in **path** must be in quotes if they contain white space or special characters.

@SEARCH accepts an optional third parameter specifying whether to search the current directory. If **n** is 0, @SEARCH will not look for the file in the current directory. If **n** is 1 (the default), @SEARCH will look in the current directory before searching the path.

Examples:

```
echo %@search[notepad]
echo %@search[msv*.dll,"d:\my dir\"]
```

3.14.4.16:@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](#)^[507] for details; to change the navigation keys see [Key Mapping directives](#)^[28]).

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 **filename**. To select from lines in the Windows clipboard, use **CLIP:** as **filename**.

If you use the optional 7th parameter **1** (immediately after the window title), the list will be sorted alphabetically.

Return value:

- the text of the line the scrollbar is on if you press **Enter**
- an empty string if you press **Esc**.

Examples:

```
call %@select["d:\path\my menu.txt",5,10,15,40,Select an option]
help %@word["=",0,%@select[d:\path\tcmd.ini,0,0,10,40,Select a
directive]]
```

3.14.4.16 @SERIAL

@SERIAL[d:]: Returns the serial number of the specified disk drive (in hex, i.e.: ABCD:0123). The drive letter must be followed by a colon.

Examples:

```
echo %@serial[C:]
echo %@serial[%_disk:]
```

See also: [@LABEL](#)⁴⁴⁶.

3.14.4.16 @SERVER

@SERVER[machinename,info]: Returns information about the specified server ***machinename***, where ***info*** is the type of information you want. The types are:

Name - return the server name

Comment - return the server comment

Version - the OS version (major version + minor version).

Users - the number of users who can attempt to log on the server.

Disconnect - the auto-disconnect time, in minutes.

Hidden - returns 1 if the server is hidden, 0 if it is visible

UserPath - the path to user directories

Type - return the type of the server. This is a combination of the following hex flags (you can use the .AND. operator in IF / IFF to test individual flags):

1	A LAN Manager workstation
2	A LAN Manager server
4	Any server running with Microsoft SQL Server
8	Primary domain controller
0x10	Backup domain controller
0x20	Server running the Timesource service
0x40	Apple File Protocol server
0x80	Novell server

0x100	LAN Manager 2.x domain member
0x200	Server sharing print queue
0x400	Server running dial-in service
0x800	Unix/Linux server
0x1000	Windows Server 2003, Windows XP, Windows 2000, or Windows NT
0x2000	Server running Windows for Workgroups
0x4000	Microsoft File and Print for NetWare
0x8000	Windows server that is not a domain controller
0x10000	Server that can run the browser service
0x20000	Server running a browser service as backup
0x40000	Server running the master browser service
0x80000	Server running the domain master browser
0x40000	Windows 95/98/Me
0x1000000	Server clusters available in the domain
0x2000000	Terminal Server
0x4000000	Cluster virtual servers available in the domain
0x40000000	Servers maintained by the browser
0x80000000	Primary domain

3.14.4.16:@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\xyz.abc"]
echo %@sfn[%_comspec]
```

3.14.4.17:@SHA1

@SHA1[filename] : Returns the SHA1 checksum of the specified file.

See also [@SHA256](#)^[460], [@SHA384](#)^[460], [@SHA512](#)^[460], [@MD5](#)^[450], and [@CRC32](#)^[413].

3.14.4.17:@SHA256

@SHA256[filename] : Returns the SHA2-256 checksum of the specified file.

See also [@SHA384](#)^[460], [@SHA512](#)^[460], [@MD5](#)^[450], and [@CRC32](#)^[413].

3.14.4.17:@SHA384

@SHA384[filename] : Returns the SHA2-384 checksum of the specified file.

See also [@SHA256](#)^[460], [@SHA512](#)^[460], [@MD5](#)^[450], and [@CRC32](#)^[413].

3.14.4.17:@SHA512

@SHA512[filename] : Returns the SHA2-512 checksum of the specified file.

See also [@SHA256](#)^[460], [@SHA384](#)^[460], [@MD5](#)^[450], and [@CRC32](#)^[413].

3.14.4.17.@SHFOLDER

@SHFOLDER[*n*] : Returns the full pathname for the specified Windows folder (which vary in different versions of Windows and if the user has altered the defaults).

n is a number from 0 to 59 that returns the following values:

- 0 - Desktop
- 2 - Start Menu\Programs
- 5 - My Documents
- 6 - <user name>\Favorites
- 7 - Start Menu\Programs\Startup
- 8 - <user name>\Recent
- 9 - <user name>\SendTo
- 11 - <user name>\Start Menu
- 13 - "My Music" folder
- 14 - "My Videos" folder
- 16 - <user name>\Desktop
- 19 - <user name>\nethood
- 20 - windows\fonts
- 21 - templates
- 22 - All Users\Start Menu
- 23 - All Users\Start Menu\Programs
- 24 - All Users\Startup
- 25 - All Users\Desktop
- 26 - <user name>\Application Data
- 27 - <user name>\PrintHood
- 28 - <user name>\Local Settings\Application Data (non roaming)
- 29 - non localized startup
- 30 - non localized common startup
- 31 - common favorites
- 32 - Internet cache
- 33 - cookies
- 34 - history
- 35 - All Users\Application Data
- 36 - Windows directory
- 37 - Windows system directory
- 38 - Program Files
- 39 - Program Files\My Pictures
- 40 - USERPROFILE
- 41 - X86 system directory on RISC
- 42 - x86 c:\Program Files on RISC
- 43 - c:\Program Files\Common
- 44 - x86 Program Files\Common on RISC
- 45 - All Users\Templates
- 46 - All Users\Documents
- 47 - All Users\Start Menu\Programs\Administrative Tools
- 48 - <user name>\Start Menu\Programs\Administrative Tools
- 53 - All Users\My Music
- 54 - All Users\My Pictures
- 55 - All Users\My Video
- 56 - Resource Directory
- 59 - USERPROFILE\Local Settings\Application Data\Microsoft\CD Burning

3.14.4.17!@SIMILAR

@SIMILAR[*string1,string2*] : Returns a value (0 - 100) reflecting the similarity between the two strings. **0** means the two strings have nothing in common; **100** means the strings are identical. Using the longer string as the first parameter usually results in lower similarity values and using the shorter results in higher values.

3.14.4.17!@SNAPSHOT

@SNAPSHOT[*DESKTOP | window[,n]*] : Save the desktop or a specific window to the clipboard as a BMP. The window argument can be either **DESKTOP** or a window title (which can include wildcards). The optional second argument specifies whether you want only the client area (0) or the entire window (1) to be saved. Returns **0** if successful.

3.14.4.17!@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 parameter in quotes. @STRIP will remove the quotes before processing the **string**.

3.14.4.17!@SUMMARY

@SUMMARY[*file,property[,value]*] : Read or set NTFS SummaryInformation data for the specified file. If it is a compound file, @SUMMARY will retrieve the data from the compound file object; otherwise @SUMMARY will retrieve the data from the SummaryInformation stream attached to the file. The valid SummaryInformation fields are:

- Title
- Subject
- Author
- Keywords
- Comments
- Template
- LastAuthor
- Revision Number
- Edit Time
- Last printed
- Created
- Last Saved
- Page Count
- Word Count
- Char Count
- AppName

Note that most files won't have any of these fields; the ones that do will usually only have some, not all.

To set SummaryInformation data, specify the value in the optional third parameter. For example, to set the **Title**:

@summary[foo.txt,Title,This is the Foo File]

3.14.4.17:@SUBSTR

@SUBSTR[*string*,*start*,*length*] : An older version of [@INSTR](#)^[443]. If the **length** is omitted, it will default to the remainder of **string**. If **string** includes commas, it must be quoted with double quotes ["] or back-quotes [,], or each comma must be preceded by an [Escape character](#)^[124]. The quotes count in calculating the position of the substring. @INSTR, which has **string** as its last parameter, 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]
```

See also: [@INSTR](#)^[443].

3.14.4.18:@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).

3.14.4.18:@SYMLINK

@SYMLINK[*link*] : Returns the target referenced by the specified symbolic link.

3.14.4.18:@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" are ignored. Any non-numeric character, except a right bracket] can be used to separate the hour, minute and second subfields.

Examples:

```
echo %@time[12:34:56]
echo %@time[%_time]
```

3.14.4.18:@TIMER

@TIMER[*n*,*precision*] : Returns the current split time for a stopwatch started with the [TIMER](#)^[347] command. The value of *n* specifies the timer to read and can be 1, 2, or 3.

@TIMER accepts an optional second argument to return the timer split as a floating-point numeric value suitable for arithmetic. The possible values are:

- s** split time in seconds (2 digit decimal precision)
- m** split time in minutes (4 digit decimal precision)
- h** split time in hours (5 digit decimal precision)

3.14.4.18 @TRIM

@TRIM[*string*] : Returns the string with the leading and trailing white space (space and tab characters) removed.

3.14.4.18 @TRUENAME

@TRUENAME[*filename*] : Returns the true, fully-expanded name for a file. @TRUENAME will "see through" junctions, symbolic links, a SUBST or network mapping. Wildcards cannot be used in the filename.

Note: The @TRUENAME function makes no assumption about the existence of a file or directory. Its *filename* parameter 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.

filename must be in quotes if it contains white space or special characters.

3.14.4.18 @TRUNCATE

@TRUNCATE[*handle*] : Truncate the file opened for write access by [@FILEOPEN](#)^[428] at the current position of the file pointer, where *handle* is the value returned by [@FILEOPEN](#)^[428].

See also the related handle-based functions:

@FILECLOSE ^[427]	Close a file handle
@FILEOPEN ^[428]	Open a file handle
@FILEREAD ^[429]	Read next line from a file handle
@FILESEEK ^[431]	Move a file handle pointer
@FILESEEK ^[431]	Move a file handle pointer to a specified line
@FILEWRITE ^[433]	Write next line to a file handle
@FILEWRITEB ^[433]	Write data to a file handle

3.14.4.18 @UNC

@UNC[*filename*] : Returns the UNC name for the specified file (or an error if the file has no UNC, e.g., a local file).

3.14.4.18 @UNICODE

@UNICODE[*string*] : Returns the space separated list of the Unicode values of the characters in *string*. You can use the [Escape character](#)^[124] before a special character (i.e., a quote or greater than (>) sign) in *string*.

See also: [@ASCII](#)^[409].

Examples:

function	value
%@unicode[a]	97
%@unicode[A]	65
%@unicode[%=`]	96
%@unicode[abc]	97 98 99

3.14.4.18:@UNIQUE

@UNIQUE*[path[,prefix]]* : Creates a zero-length file with a unique name in the specified directory, and returns its 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.

If **prefix** is specified, @UNIQUE will use the first three characters as the first three characters of the unique filename.

Because the file is created, if the [Protect Redirected Output File](#)^[47] configuration option is set, you must use the style >! redirection to avoid errors.

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 Windows. If you experience this problem you may need to use [DELAY](#)^[194], DELAY /M, or [BEEP](#)^[173] (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.

3.14.4.19:@UNQUOTE

@UNQUOTE*[string]* : Returns the argument with all double quotes removed.

3.14.4.19:@UNQUOTES

@UNQUOTES*[string]* : Returns the argument with leading and trailing double quotes removed.

3.14.4.19:@UPPER

@UPPER*[string]* : Returns **string** converted to upper case.

3.14.4.19:@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, **TCMD.EXE** returns values for:

```
CompanyName
FileDescription
FileVersion
InternalName
LegalCopyright
LegalTrademarks
OriginalFilename
```

```

ProductName
ProductVersion
Build

```

To return **CompanyName** :

```
echo %@verinfo[tcmd.exe,companyname,040904E4]
```

Note: Most, but not all, executables under Windows contain a **FileVersion** field. 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.

3.14.4.19!@WATTRIB

@WATTRIB[*filename*[-*attributes*[*p*]]]: This function is similar to **@ATTRIB**^[409], 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 or symbolic links
L Junction or symbolic links
C Compressed
O Offline
I Not content-indexed

See also: [Attributes Switches](#)^[86] and the [ATTRIB](#)^[163] command.

3.14.4.19!@WILD

@WILD[*string1*,*string2*] : Compares two strings and returns **1** if they match or **0** if they don't match. This function determines whether or not **string1** matches the pattern specified in **string2**, which may contain wildcards or [extended wildcards](#)^[77]. No wildcards are permitted in **string1**. The test is not case sensitive.

Examples

The examples below assume that the **PATH** variable contains:

```
c:\windows;c:\windows\system32;"c:\program files\util";d:\jpsoft
```

string1	string2	match condition	result
%path	*\UTIL*	string \u <i>t</i> i <i>l</i> anywhere	1
%path	*c	string ending with <i>c</i>	0
%path	*t	string ending with <i>t</i>	1
%path	c*	string starting with <i>c</i>	1
%path	t*	string starting with <i>t</i>	0
%path	*c*	string containing <i>c</i>	1
%path	*t*	string containing <i>t</i>	1
%path	*b*	string containing <i>b</i>	0
xyz	?	one character long string	0
x	?	one character long string	1

%path	c?*	leading c, followed by any one character, followed by 0 or more characters	1
%path	c*?	leading c, followed by zero or more characters, followed by any one character	1

3.14.4.19!@WINAPI

@WINAPI[module,function[,integer | PINT=n | PLONG=n | PDWORD=n | NULL | BUFFER | "string"] :
Returns the result of calling a Windows API function. The arguments are:

module - name of the DLL containing the function

function - function name (case sensitive)

integer - an integer value to pass to the function

PINT - a pointer to the integer *n*

PLONG - a pointer to the long integer *n*

PDWORD - a pointer to the DWORD *n*

NULL - a null pointer (0)

BUFFER - @WINAPI will pass an address for an internal buffer for the API to return a Unicode string value.

aBUFFER - @WINAPI will pass an address for an internal buffer for the API to return an ASCII string value.

"string" - text argument (this must be enclosed in double quotes). If the argument is preceded by an 'a' (i.e., a"Argument") then it is converted from Unicode to ASCII before calling the API. (Some Windows APIs only accept ASCII arguments.)

@WINAPI supports a maximum of 8 arguments. The return value is either a string value returned by the API (if BUFFER or aBUFFER is specified), or the integer value returned by the API. The function must be defined as WINAPI (__stdcall). If @WINAPI can't find the specified function, it will append a "W" (for the Unicode version) to the function name and try again.

See also [@CAPI](#)^[410].

3.14.4.19!@WINCLASS

@WINCLASS[classname] : Returns the window title of the first window with the specified class name, or an empty string if no windows match.

3.14.4.19!@WINEXENAME

@WINEXENAME[title]: Returns the executable name for the first window matching **title** (which can include [wildcards](#)^[77]), or an empty string if none.

3.14.4.19!@WININFO

@WININFO[n]: Returns information about the current system. *n* is a number specifying what information to return:

<i>n</i>	<i>Information returned</i>
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 586 Pentium:
5	Processor level
6	Processor revision
7	page size, bytes
8	virtual memory allocation granularity, bytes

3.14.4.20 @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

3.14.4.20 @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
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	Coordinate of left side of virtual screen
77	Coordinate of top of virtual screen

78	Width in pixels of virtual screen
79	Height in pixels of virtual screen
80	Number of monitors on desktop

3.14.4.20:@WINPOS

@WINPOS[title]: Returns the screen coordinates of the window with the specified title, in the format "*top,left,bottom,right*".

3.14.4.20:@WINSTATE

@WINSTATE[title] : Returns the window state of the first window matching **title** (which can include [wildcards](#) ⁽⁷⁷⁾). The return values are:

Value	Window state
0	Hidden
1	Normal
2	Minimized
3	Maximized

3.14.4.20:@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.

Where the selection is a state, the legal values are **0** for off/disabled, and **1** for on/enabled.

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. 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	Screen saver state
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	Menu fade animation state.
4116	4117	Selection fade effect state.
4118	4119	ToolTip animation state.
4120	4121	Type of ToolTip animation (1 for fade, 0 for slide)
4122	4123	Cursor shadow state.
4124	4125	State of the Mouse Sonar feature
4126	4127	Mouse clicklock state
4128	4129	Mouse vanish feature state
4130	4131	Whether native User menus have flat menu appearance.
4132	4133	Drop shadow effect state.
4158	4159	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	Caret width in edit controls

8200	8201	Time delay before the primary mouse button is locked.
8202	8203	Type of font smoothing (32769=standard anti-aliasing, 32770=ClearType).
8204	8205	Contrast value used in ClearType smoothing (1000-2200)
8206	8207	Width of the left and right edges of the focus rectangle
8208	8209	Height of the top and bottom edges of the focus rectangle
.		
GET	SET	Parameter to GET or SET

3.14.4.20:@WMI

@WMI[*namespace*,"*wql search*"[,*enum*]]: Returns the result of the WMI query.

The optional **enum** parameter specifies the property instance to return for classes that return multiple properties. You can omit the **enum** parameter if you're querying a single property and instance.

For details on what information is available, see the WMI and WQL documentation on MSDN (msdn.microsoft.com).

See also [WMIQUERY](#) ^[364].

Examples:

```
%@wmi[root\cimv2,"SELECT name FROM Win32_Processor"]
```

```
%@wmi[root\cimv2,"SELECT name, state FROM Win32_service",4]
```

3.14.4.20:@WORD

@WORD[["*sep_list*",]*n*,*string*]: Returns the *n*th word in **string**. The first (leftmost) word is numbered 0. If *n* is negative, 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**.

You can specify a range of words to return with the syntax:

```
@WORD[["sep_list"],start[-end | +range],string]
```

Specify an inclusive range with a **-**. For example:

```
%@word[2-4,A B C D E F G] will return "C D E". (Note that you cannot use inclusive ranges when starting from the end.)
```

You can specify a relative range with a **+**. For example:

```
%@word[2+1,A B C D E F G] will return "C D".
```

The default list of separators for [@FIELD](#) ^[426], [@FIELDS](#) ^[427], [@WORD](#) ^[472] and [@WORDS](#) ^[473] consists of space, tab, and comma. You can use the optional first parameter, *sep_list*, 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](#) ^[124]. Alphabetic characters in *sep_list* are case sensitive.

[@FIELD](#) ^[426] and [@FIELDS](#) ^[427] differ from [@WORD](#) ^[472] and [@WORDS](#) ^[473] in how multiple consecutive separators are counted. [@WORD](#) ^[472] and [@WORDS](#) ^[473] consider a sequence as a single separator, and ignore separators at either end of string. In contrast, [@FIELD](#) ^[426] and [@FIELDS](#) ^[427] count each occurrence of a separator individually, including those at either end of 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, e.g., **0xFFFFFFFF** for **-1**.

If **string** is double quoted, you must specify **sep_list**.

See also: [@WORDS](#)^[473], [@FIELD](#)^[426], [@FIELDS](#)^[427].

Examples:

function	value
%@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

3.14.4.20!@WORDS

@WORDS["sep_list",string]: Returns the number of words in **string**.

The default list of separators for [@FIELD](#)^[426], [@FIELDS](#)^[427], [@WORD](#)^[472] and [@WORDS](#)^[473] consists of space, tab, and comma. You can use the optional first parameter, **sep_list**, 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](#)^[124]. Alphabetic characters in **sep_list** are case sensitive.

[@FIELD](#)^[426] and [@FIELDS](#)^[427] differ from [@WORD](#)^[472] and [@WORDS](#)^[473] in how multiple consecutive separators are counted. [@WORD](#)^[472] and [@WORDS](#)^[473] consider a sequence as a single separator, and ignore separators at either end of **string**. In contrast, [@FIELD](#)^[426] and [@FIELDS](#)^[427] count each occurrence of a separator individually, including those at either end of **string**.

If **string** is double quoted, you must specify **sep_list**.

See also: [@WORD](#)^[472], [@FIELD](#)^[426], [@FIELDS](#)^[427].

3.14.4.20!@WORKGROUP

@WORKGROUP[name]: Returns the workgroup of the computer specified by the DNS or NetBios **name**. If **name** is not specified, @WORKGROUP returns the workgroup of the local computer.

3.14.4.20!@XMLCLOSE

@XMLCLOSE[]: Close an XML file previously opened by [@XMLOPEN](#)^[474].

For more details on XML support, see [XML in Take Command](#)^[502].

3.14.4.21!@XMLNODES

@XMLNODES["filename"],path]: Return the number of nodes (children) for the specified path in an XML file. The arguments are:

filename - name of XML file

path - one or more element accessors separated by a /.

If you don't specify a filename (which **must** be in double quotes), @XMLXPath will use the XML file previously opened by [@XMLOPEN](#)^[474].

For more details on XML support, see [XML in Take Command](#)^[502].

3.14.4.21: @XMLOPEN

@XMLOPEN[*filename*] - open an XML file for use by [@XMLXPath](#)^[474] and/or [@XMLNODES](#)^[473].

For more details on XML support, see [XML in Take Command](#)^[502].

3.14.4.21: @XMLXPath

@XMLXPath[["*filename*"],*path*] : XML XPath query. (See the [XML XPath docs](#) for details on XPath syntax.) The arguments are:

filename - name of XML file

path - one or more element accessors separated by a /.

If you don't specify a filename (which **must** be in double quotes), @XMLXPath will use the XML file previously opened by [@XMLOPEN](#)^[474].

For more details on XML support, see [XML in Take Command](#)^[502].

3.14.4.21: @YEAR

@YEAR[*date*[,*format*]]: Returns the year for the specified date. See [date formats](#)^[127] for valid formats.

@YEAR accepts an optional second parameter specifying the date format:

- 0 default
- 1 USA (mm/dd/yy)
- 2 Europe (dd/mm/yy)
- 3 Japan (yy/mm/dd)
- 4 ISO (yyyy/mm/dd)

4 Troubleshooting

- ▶ [Registration](#)^[474]
- ▶ [Troubleshooting Service and Support](#)^[475]
- ▶ [Supported Platforms](#)^[477]
- ▶ [Help File](#)^[477]
- ▶ [Error Messages](#)^[478]

4.1 Registration

There are no separate **trial** and **registered** versions of **Take Command**. Without registration, a trial version is fully functional for 30 days of use.

At any time you can apply your current personal registration information to a trial version in order to turn it into a registered product. Use the command [VER/R](#)^[359] from the prompt to verify the status of the copy you are currently running. (You can also view the [Help/About](#)^[61] menu entry.)

When you purchase a new or upgrade copy of **Take Command**, you will receive an email with your name and registration key. Start **Take Command**, click on the **Options** menu entry and then

Configure Take Command. Select the **Register** tab and enter the registration information exactly as you received it in the email. Remember to save your registration key in a safe place in case you need to reinstall. If you have lost your registration key, you can request a replacement by contacting JP Software at support@jpsoft.com, or at one of the addresses listed at the start of this file.

4.2 Troubleshooting, Service & Support

If you need help with **Take Command**, 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](#)^[477] 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 **Take Command**, review the [Technical Support](#)^[475] 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.

4.2.1 Technical Support

Support Plans

Standard, no-charge support is available electronically through our [Support Forum](#)^[475] (see below). 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?](#)^[476] section which outlines the basic data we need 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/>.

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 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 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.

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 **Take Command** to run properly, review the information on [Error Messages](#)^[478], 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](#)^[477] 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](#)^[359] command to determine the **Take Command** version and operating system version. This item is essential! Every question posted on the Forum should include a brief identification such as "Take Command 9.0.88 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.

- **If the problem seems related to startup and configuration issues**, what are the contents of any startup files you use (such as [TCSTART](#)^[22] or [TCEXIT](#)^[22], and the .INI file), any batch files they call, and any alias or environment variable files they load?
- **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 **Take Command** experiences an unrecoverable failure, it will attempt to send the relevant memory locations and other useful data to the Windows Error Reporting (WER) site.

4.2.2 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
Phone:	800-595-8197
Fax:	800-595-8197 (rings through to fax)
Online:	Web site: http://jpsoft.com/ FTP site: ftp://jpsoft.com Sales / Customer Service: support@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](#)^[475] for additional details, and for information on paid support options.

Note: Our server implements 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.

4.3 Supported Platforms

Take Command is a Win32 GUI application.

TCC is a Win32 console (character-mode) application.

Both are designed to run under Windows XP, Windows 2003, Windows Vista, and Windows 2008.

4.4 Help File

The installer for **Take Command** includes **tcmd.chm**, a help file in Window's compiled HTML format. That file includes description and syntax for all commands, variables and functions, as well as reference information to assist you in installing and using **Take Command** and developing batch files,

aliases and functions. **Take Command** uses the default Windows help system to display the contents of **tcmd.chm**. 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**^[34] or the **HELP**^[25] command, you can use standard Windows HTML Viewer (HH.EXE) keystrokes to navigate. For more information, see your Windows documentation.

The help is also available as PDF file. Please check our web site regularly to make sure you have the most recent version of the help files, since they are usually updated more frequently than the programs.

You can request help at the prompt from the **Help** menu, by typing **HELP**^[25] (or **HELP**^[25] plus a command name), or by pressing the **F1**^[34] key at any time when **TCC** is accepting keyboard input at the prompt. If you use the **HELP**^[25] 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**^[34], the help system will provide context sensitive help by using the first word on the line as the 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, internal variables, variable functions, and key mapping 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**^[34] key. This feature is especially useful when you need the syntax for a variable function.

If the topic you seek is not 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.

Quick Syntax Help:

If you just need a quick reminder of an internal command's syntax, type the name of the command at the prompt, followed by a slash and a question mark **/?** For example:

```
copy /?
```

TCC will display the syntax and the valid options for the command.

4.5 Error Messages

This section lists error messages generated by **Take Command**, and includes a recommended course of action for most errors. If you are unable to resolve the problem after reviewing these help files, contact JP Software for [technical support](#)^[475].

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. **Take Command** 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. See [Windows System Errors](#)^[507] for examples of system errors returned in the [_SYSERR](#)^[392] 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](#)^[166]) 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: **TCC** 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. **TCC** attempts to perform full path and filename expansion before copying to help ensure that files aren't inadvertently destroyed.

Can't create: **TCC** 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: **TCC** can't delete the specified file or directory. The disk is probably write protected.

Can't end current process: You attempted to terminate **TCC** with a [TASKEND](#)^[342] command. TASKEND can only be used to end other processes; to terminate **TCC**, use the [EXIT](#)^[227] command.

Can't get directory: **TCC** can't read the directory. The disk drive is probably not ready.

Can't make directory entry: **TCC** 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: **TCC** 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](#)^[455] 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: *Take Command* 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](#) ^[126] (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.

Directory stack empty: [POPD](#) ^[294] or [DIRS](#) ^[209] 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 *Take Command*, 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 *//inline* option to place an .INI directive on the [startup](#) ^[17] 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](#) ^[26]. The following message explains the error in more detail. Correct the line in error and restart *TCC* 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](#)^[162] command could not find a file association for the specified extension in the Windows registry.

File exists: The requested output file already exists, and **TCC** won't overwrite it.

File not found: **TCC** couldn't find the specified file. Check the spelling and path name.

File type not found: The [FTYPE](#)^[241] 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. See also: **Data error** above.

Include file not found: You used the Include directive in the [.INI file](#)^[26], 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](#)^[26], 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](#)^[166]) You are attempting to compress a file to itself.

Input file is already compressed: ([BATCOMP](#)^[166]) 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](#)^[141] 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](#)^[26].

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](#)^[26].

Invalid color: You gave an invalid value for a color directive in the [INI file](#)^[26].

Invalid count: The character repeat count for [KEYSTACK](#)^[262] is incorrect.

Invalid date: An invalid date was entered. Check the syntax and reenter.

Invalid directive name: **Take Command** can't recognize the name of a directive in the [INI file](#)^[26].

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](#)^[26], or you used an invalid key name in a keystroke [alias](#)^[154] 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](#)^[26].

Invalid parameter: **TCC** 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](#)^[26].

Invalid time: An invalid time was entered. Check the syntax and reenter.

Keystroke substitution table full: **TCC** ran out of room to store [keystroke substitutions](#)^[28] entered in the [.INI file](#)^[26]. Reduce the number of key substitutions or contact JP Software or your dealer for assistance.

Label not found: A [GOTO](#)^[246] or [GOSUB](#)^[247] 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](#)^[262] command is missing a closing parentheses around a character group. Correct the command.

Missing ENDTEXT: A [TEXT](#)^[345] command is missing a matching ENDTEXT. Check the batch file.

Missing GOSUB: **TCC** cannot perform the [RETURN](#)^[307] command in a batch file. You tried to do a RETURN without a [GOSUB](#)^[247], or your batch file has been corrupted.

Missing SETLOCAL: An [ENDLOCAL](#)^[220] was used without a matching [SETLOCAL](#)^[326].

No aliases defined: You tried to display aliases but no aliases have been defined.

No closing quote: **TCC** couldn't find a second matching back quote ['] or double-quote ["] on the command line.

No expression: The expression passed to the [%@EVAL](#)^[420] variable function is empty. Correct the expression and retry the operation.

No shared memory found: The [SHRALIAS](#)^[329] 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 **TCC** with at least one global list, then invoke SHRALIAS.

No SMTP server: [SENDMAIL](#)^[316] 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](#)^[301] 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](#)^[305]. You cannot rename a file to a different

disk drive.

Out of function space: You are attempting to create a [User-defined Function](#)^[242] 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: *Take Command* 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. Check your printer and add paper if necessary.

Overflow: An arithmetic overflow occurred in the [@EVAL](#)^[420] 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. See also: **Data error** above.

Sector not found: Disk error, usually caused by a bad or unformatted disk. See also: **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. See also: **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](#)^[329] 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](#)^[329] command to unload SHRALIAS.EXE, but it was never loaded. This message is informational and may not indicate an error condition.

String area overflow: *TCC* ran out of room to store the text from string directives in the [.INI file](#)^[26]. 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](#)^[262] buffer. Reduce the number of characters you are trying to send to the application at one time.

Syntax error: A command or [variable function](#)^[395] 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](#)^[420] variable function. Correct the expression and retry the operation.

UNKNOWN_CMD loop: The [UNKNOWN_CMD alias](#)^[154] 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 *TCC* 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](#)^[154]).

Unknown option name: ([OPTION](#)^[284]) You are attempting to modify or display an invalid or unknown option name.

Unknown process: [TASKEND](#)^[342] 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](#)^[152] 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. See also: **Data error** above.

5 Reference Information

- ▶ [CMD.EXE Comparison](#)^[484]
- ▶ [File Systems and File Name Conventions](#)^[490]
- ▶ [Miscellaneous Reference Information](#)^[504]
- ▶ [ASCII, Key Codes, and ANSI Commands](#)^[510]
- ▶ [Glossary](#)^[519]
- ▶ [Copyright and Version](#)^[537]

5.1 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.EXE command name matches an internal **TCC** command, the **TCC** command is almost always enhanced.

- ▶ [TCC command equivalents in CMD.EXE](#)^[484]
- ▶ [CMD.EXE command equivalents in TCC](#)^[487]
- ▶ [Command line editing](#)^[488]
- ▶ [Filename completion](#)^[489]
- ▶ [Command completion](#)^[489]
- ▶ [Redirection](#)^[489]
- ▶ [Wildcards](#)^[489]
- ▶ [Built-In Variables](#)^[489]
- ▶ [Batch File Structure](#)^[489]
- ▶ [Unique TCC features](#)^[489]

TCC command equivalent in CMD.EXE

TCC command	CMD.EXE command (nearest functionality)	CMD.EXE command type
? ^[380]	HELP	external

ACTIVATE ^[152]		
ALIAS ^[154]	DOSKEY	external
ASSOC ^[162]	ASSOC	internal
ATTRIB ^[163]	ATTRIB	external
BATCOMP ^[166]		
BDEBUGGER ^[166]		
BEEP ^[173]		
BREAK ^[174]	BREAK	internal
BREAKPOINT ^[174]		
CALL ^[175]	CALL	internal
CANCEL ^[176]		
CD ^[177]	CD	internal
CDD ^[178]		
CHCP ^[180]	CHCP	external
CHDIR ^[177]	CHDIR	internal
CLS ^[181]	CLS	internal
COLOR ^[181]	COLOR	internal
COPY ^[182]	COPY	internal
COPY ^[182]	XCOPY	external
DATE ^[189]	DATE	internal
DEL ^[190]	DEL	internal
DELAY ^[194]		
DESCRIBE ^[195]		
DETACH ^[197]		
DIR ^[198]	DIR	internal
DIRHISTORY ^[208]		
DIRS ^[209]		
DO ^[210]		
DRAWBOX ^[214]		
DRAWHLINE ^[215]		
DRAWVLINE ^[216]		
ECHO ^[217]	ECHO	internal
ECHOERR ^[218]		
ECHOS ^[219]		
ECHOSERR ^[220]		
ENDLOCAL ^[220]	ENDLOCAL	internal
ERASE ^[190]	ERASE	internal
ESET ^[222]		
EVENTLOG ^[223]		
EXCEPT ^[225]		
EXIT ^[227]	EXIT	internal
FFIND ^[227]	FIND	external
FFIND ^[227]	FINDSTR	external
FOR ^[234]	FOR	internal
FREE ^[241]		
FTYPE ^[241]	FTYPE	internal
FUNCTION ^[242]		
GLOBAL ^[245]		
GOSUB ^[247]	CALL :LABEL	internal
GOTO ^[248]	GOTO	internal
HEAD ^[249]		
HELP ^[251]	HELP	external
HISTORY ^[251]		

IF ^[253]	IF	internal
IFF ^[254]		
IFTP ^[255]		
INKEY ^[257]		
INPUT ^[259]		
KEYBD ^[261]		
KEYS ^[261]		
KEYSTACK ^[262]		
LIST ^[264]	MORE	external
LOADBTM ^[269]		
LOG ^[269]		
MD ^[271]	MD	internal
MEMORY ^[272]		
MKDIR ^[271]	MKDIR	internal
MKLNK ^[273]		
MOVE ^[274]	MOVE	internal
MSGBOX ^[279]		
ON ^[282]		
OPTION ^[284]		
OSD ^[286]		
PATH ^[286]	PATH	internal
PAUSE ^[287]	PAUSE	internal
PDIR ^[288]		
PLAYAVI ^[291]		
PLAYSOUND ^[292]		
POPD ^[294]	POPD	internal
PRINT ^[295]	PRINT	external
PRIORITY ^[295]		
PROMPT ^[297]	PROMPT	internal
PUSHD ^[299]	PUSHD	internal
QUERYBOX ^[300]		
QUIT ^[301]	GOTO :EOF	
RD ^[301]	RD	internal
REBOOT ^[303]		
RECYCLE ^[304]		
REM ^[304]	REM	internal
REN ^[305]	REN	internal
RENAME ^[305]	RENAME	internal
RETURN ^[307]	GOTO :EOF	
REXEC ^[308]		
RSHELL ^[309]		
RMDIR ^[301]	RMDIR	internal
SCREEN ^[310]		
SCRPUT ^[311]		
SELECT ^[312]		
SENDMAIL ^[316]		
SET ^[319]	SET	internal
SETDOS ^[323]		
SETLOCAL ^[326]	SETLOCAL	internal
SHIFT ^[327]	SHIFT	internal
SHORTCUT ^[328]		
SHRALIAS ^[329]		
SMPP ^[330]		

SNPP ³³¹		
START ³³¹	START	internal
SWITCH ³³⁵		
SYNC ³³⁶		
TAIL ³³⁹		
TASKEND ³⁴²		
TASKLIST ³⁴²		
TEE ³⁴⁴		
TEXT ³⁴⁵		
TIME ³⁴⁷	TIME	internal
TIMER ³⁴⁷		
TITLE ³⁴⁹	TITLE	internal
TOUCH ³⁴⁹		
TREE ³⁵²	TREE	external
TRUENAME ³⁵³		
TYPE ³⁵⁴	TYPE	internal
UNALIAS ³⁵⁵		
UNFUNCTION ³⁵⁶		
UNSET ³⁵⁷	SET VARNAME=	internal
VER ³⁵⁹	VER	internal
VERIFY ³⁶⁰	VERIFY	internal
VOL ³⁶⁰	VOL	internal
VSCRPUT ³⁶⁰		
WHICH ³⁶¹		
WINDOW ³⁶²		
Y ³⁶⁴		

CMD.EXE command equivalents in TCC

CMD.EXE command	CMD.EXE class	TCC command	comparison
ASSOC	internal	ASSOC ¹⁶²	enhanced
AT	external		
ATTRIB	external	ATTRIB ¹⁶³	enhanced
BREAK	internal	BREAK ¹⁷⁴	enhanced
CACLS	external		
CALL	internal	CALL ¹⁷⁵ , GOSUB ²⁴⁷	enhanced
CD	internal	CD ¹⁷⁷	enhanced
CHCP	external	CHCP ¹⁸⁰	identical
CHDIR	internal	CHDIR ¹⁷⁷	enhanced
CHKDSK	external		
CHKNTFS	external		
CLS	internal	CLS ¹⁸¹	enhanced
COLOR	internal	COLOR ¹⁸¹	enhanced
COMP	external		
COMPACT	external		
CONVERT	external		
COPY	internal	COPY ¹⁸²	enhanced
DATE	internal	DATE ¹⁸⁹	enhanced
DEL	internal	DEL ¹⁹⁰	enhanced
DIR	internal	DIR ¹⁹⁸ , PDIR ²⁸⁸	enhanced
DISKCOMP	external		

DISKCOPY	external		
DOSKEY	external	ALIAS <small>[154]</small>	enhanced
ECHO	internal	ECHO <small>[217]</small>	identical
ENDLOCAL	internal	ENDLOCAL <small>[220]</small>	enhanced
ERASE	internal	ERASE <small>[190]</small>	enhanced
EXIT	internal	EXIT <small>[227]</small>	enhanced
FC	external		
FIND	external	FFIND <small>[227]</small>	enhanced
FINDSTR	external	FFIND <small>[227]</small>	enhanced
FOR	internal	FOR <small>[234]</small>	enhanced
FORMAT	external		
FTYPE	internal	FTYPE <small>[241]</small>	enhanced
GOTO	internal	GOTO <small>[248]</small>	enhanced
GRAFTABL	external		
HELP	external	HELP <small>[251]</small>	enhanced
IF	internal	IF <small>[253]</small>	enhanced
LABEL	external		
MD	internal	MD <small>[271]</small>	enhanced
MKDIR	internal	MKDIR <small>[271]</small>	enhanced
MODE	external		
MORE	external	LIST <small>[264]</small>	enhanced
MOVE	internal	MOVE <small>[274]</small>	enhanced
PATH	internal	PATH <small>[286]</small>	identical
PAUSE	internal	PAUSE <small>[287]</small>	enhanced
POPD	internal	POPD <small>[294]</small>	enhanced
PRINT	external	PRINT <small>[295]</small>	enhanced
PROMPT	internal	PROMPT <small>[297]</small>	enhanced
PUSHD	internal	PUSHD <small>[299]</small>	enhanced
RD	internal	RD <small>[301]</small>	enhanced
RECOVER	external		
REM	internal	REM <small>[304]</small>	identical
REN	internal	REN <small>[305]</small>	enhanced
RENAME	internal	RENAME <small>[305]</small>	enhanced
REPLACE	external		
RMDIR	internal	RMDIR <small>[301]</small>	enhanced
SET	internal	SET <small>[319]</small>	enhanced
SETLOCAL	internal	SETLOCAL <small>[326]</small>	enhanced
SHIFT	internal	SHIFT <small>[327]</small>	enhanced
SORT	external		
START	internal	START <small>[331]</small>	enhanced
SUBST	external		
TIME	internal	TIME <small>[347]</small>	enhanced
TITLE	internal	TITLE <small>[349]</small>	enhanced
TREE	external	TREE <small>[352]</small>	enhanced
TYPE	internal	TYPE <small>[354]</small>	enhanced
VER	internal	VER <small>[359]</small>	enhanced
VERIFY	internal	VERIFY <small>[360]</small>	enhanced
VOL	internal	VOL <small>[360]</small>	identical
XCOPY	external	COPY <small>[182]</small>	enhanced

Command line editing

Filename completion**Command completion****Redirection****Wildcards**

CMD.EXE only supports the ? and * wildcards.

Built-In Variables

CMD.EXE has several built-in variables (i.e., which are treated as environment variables but which do not exist in the environment):

CD - current directory

CMDCMDLINE - command line that started CMD

CMDEXTVERSION - the command extensions internal version number

DATE - the current date (in the default short format)

RANDOM - a random number between 0 and 32767

TIME - current time

TCC supports all of these built-in variables. (In **TCC**, **CMDEXTVERSION** will always return **2**.)

Batch File Structure**Unique *Take Command* features****Batch debugger****Aliases****Internal functions****User defined functions****File selection****Ranges****Internet access and email****OpenAFS support****ANSI X3.64 support**

Directory navigation

Histories and logs

Intersession sharing

Perl, REXX, and Ruby support

5.2 Limits

Length Limits

entity	name	value	all
environment variable	1023	4,095	4,095
alias	1023	4,095	4,095
user defined function	1023	4,095	4,095

command type	before expansion	after expansion
command line	32,767	65,535
command group	32,767	65,535

Nesting Limits

command	depth
CALL <small>[175]</small>	32
DO <small>[210]</small>	no limit
FOR <small>[234]</small>	no limit
GOSUB <small>[247]</small> without parameters	no limit
GOSUB <small>[247]</small> with parameters	22
SETLOCAL <small>[326]</small>	16
IFF <small>[254]</small>	no limit

Miscellaneous Limits

entity	limit
character count in any function	4,095
number of batch file parameters	4,095
number of GOSUB parameters	255
file name	4,095
include list	4,095
single parameter	4,095
global alias list	131,072
global function list	65,536
key substitution table	128
directory stack	2,047

5.3 File Systems & File Name Conventions

You probably have 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. In Windows, 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. (Note that accessing Linux / UNIX FTP servers, the filenames **are** case sensitive.)

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](#)^[491]
- ▶ [File Systems](#)^[491]
- ▶ [Directories and Subdirectories](#)^[492]
- ▶ [File Names](#)^[493]
- ▶ [File Attributes](#)^[494]
- ▶ [Time Stamps](#)^[495]
- ▶ [NTFS Streams](#)^[496]

5.3.1 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 access network drives via their **UNC** (universal naming convention) name (e.g. `\\data\vol1\...`), without using a drive letter. See [File Systems](#)^[491] for more details.

Most systems also include optical drives (i.e. CD-ROM, CD-RW, and/or DVD). The optical drive is also assigned a drive letter (or several letters, for changers), typically using letters beyond that used by the last hard disk in the system, but before any 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 DVD drive, and **H:** and **I:** as network drives.

Each volume is formatted under a particular file system; see [File Systems](#)^[491] for details. Additional information about disk files and directories is available under [Directories and Subdirectories](#)^[492], [File Names](#)^[493], and [File Attributes](#)^[494].

5.3.2 File Systems

Take Command uses only documented Windows APIs to access the file systems, so it works with any file system supported by Windows.

Additional information about disk files and directories is available under [Drives and Volumes](#)^[491], [Directories and Subdirectories](#)^[492], [File Names](#)^[493], and [File Attributes](#)^[494].

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. **Take Command** 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, OS X, 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 **Take Command**.

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. **Take Command** will usually handle whatever naming conventions are supported by your network software, as long as the network software accurately reports the types of names it can handle.

In rare cases, **Take Command** 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\path\filename**, where **server** is the name of the network server where the files reside, and the **path\filename** portion is a directory name and file name which follow the conventions described under [Directories](#)^[492].

Take Command also allows you to use UNC directory names when changing directories (see [Directory Navigation](#)^[71] for more details).

OpenAFS

Take Command has built-in support for OpenAFS. The parser will recognize Linux-style AFS names (i.e., **/afs/athena/user**) and convert them to Windows-compatible names (i.e., **\\afs\athena\user**). (It will also check for custom AFS mount points, and use that name instead of **afs**.)

See <http://www.openafs.org> for more information on OpenAFS.

5.3.3 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](#)^[493]).

The drive and subdirectory portions of a file's name are 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 **TCC**, the path and filename can be up to 4095 characters, though most Windows applications (including **CMD.EXE** and **Explorer**) have trouble with path and filename lengths exceeding 260 characters. Shorter paths and names are advisable under Windows whenever feasible.

TCC 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. **TCC** allows you to use additional periods to specify directories further "up" the tree (see [Extended Parent Directory Names](#)^[118]).

Additional information about disk files and file systems is available under [Drives and Volumes](#)^[491], [File Systems](#)^[491], [File Names](#)^[493], and [File Attributes](#)^[494].

5.3.4 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 filenames (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 **TCC**, it is best to avoid using them in filenames. It is also better to use only those characters found in [ASCII](#)^[511], 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 **TCC**.

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](#)^[77] for details on avoiding problems with interpretation of those file names under **TCC**.

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 Linux-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 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 **TCC**, you must place double quotes 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:

```
copy let3 f:\LETTERS\Letter To Sara"
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*.

Additional information about disk files and file systems is available under [Drives and Volumes](#)^[491], [File Systems](#)^[491], [Directories and Subdirectories](#)^[492], [File Attributes](#)^[494], and [Time Stamps](#)^[495].

5.3.5 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. NTFS volumes support additional attributes: Encrypted, Compressed, Normal, Offline, Temporary, Not content-indexed, Sparse, and Junction / Symbolic Link / Reparse point. **Take Command** fully supports these [extended attributes](#)^[86].

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 **TCC** 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 **Take Command**, unless explicitly requested.. 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 subdirectory is created, *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 Windows utility LABEL (or equivalent third-party software). **Take Command** does not directly modify the volume label or any of its attributes, and provide read access only through the [VOL](#)^[360] command and the [@LABEL](#)^[446] variable function. All other commands ignore this directory entry.

Normal – this pseudo attribute is considered to be set if all other attributes (including the [extended attributes](#)^[86] available only on an NTFS volumes) are reset. It is not stored by the file system. When **Take Command** checks file attributes, it considers 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](#)^[163] and [DIR](#)^[198] commands, and by the [@ATTRIB](#)^[409] and [@WATTRIB](#)^[466] variable functions.

Attributes can be set, reset, and viewed with the [ATTRIB](#)^[163] command. The [DIR](#)^[198] command also has options to view the attribute status of files, and to view information about normally invisible hidden and system files and directories.

5.3.6 File 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 **TCC** file processing commands allow you to select files for processing based on their time stamps.

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. 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.
2. **Creation time** is the date and time the current instance of the file was created.
3. **Access time** is the date, and on NTFS volumes, the time, when the file was last accessed for either reading or writing.

Several **TCC** 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.

Time Stamp Resolution

The resolution of time stamps as well as the range of time instances representable vary with file systems. The table below shows some of them.

<i>file system</i>	<i>resolution</i>	<i>earliest time stamp</i>	<i>latest time stamp</i>
FAT/VFAT	2 s	1980-01-01 00:00:00 <i>local</i>	2107-12-31 23:59:58 <i>local</i>
NTFS	100 ns	1601-01-01 00:00:01 <i>UTC</i>	

UNIX/Linux	1 s	1970-01-01 00:00:00 UTC	
------------	-----	-------------------------	--

NTFS Timestamp Reports

These operating systems report timestamps in local time. However, conversion between UTC and local time is based on the difference between UTC and local time at the time of conversion, instead of that in effect when the file event occurred. Consequently, if daylight saving time is currently in effect, all file events around the year will be reported in DST. conversely, when DST is not in effect, all file events around the year will be reported in standard time. This method has the advantage that differences in event times can be calculated easily. However, the times reported will not be those when the event took place if the state DST at time of event is not the same as at the time of reporting.

The [TOUCH](#)^[349] command can be used to modify the timestamps of files and directories.

Additional information about disk files and file systems is available under [Drives and Volumes](#)^[491], [File Systems](#)^[491], [Directories and Subdirectories](#)^[492], and [File Names](#)^[493].

5.3.7 NTFS File Streams

The NTFS file system 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
```

You cannot use wildcards in stream names except when using [filename completion](#)^[368].

You can display stream names with the [DIR](#)^[198] `/:` option. The file processing commands [COPY](#)^[182], [DEL](#)^[190], [FFIND](#)^[227], [LIST](#)^[264], [MOVE](#)^[274] and [TYPE](#)^[354] support file streams when the stream name is explicitly specified; see the individual commands for additional details. Other file-related commands, such as ATTRIB 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](#)^[428], [@LINE](#)^[447], and [@LINES](#)^[448] also accept stream names.

5.4 Regular Expression Syntax

Oniguruma Regular Expressions Version 5.9.0 2007/05/31

This section covers the Ruby regular expression syntax. For information on Perl regular expression syntax, see your Perl documentation or <http://www.perl.com/doc/manual/html/pod/perlre.html>.

1. Syntax elements

\	escape (enable or disable meta character meaning)
	alternation
(...)	group
[...]	character class

2. Characters

\t	horizontal tab (0x09)
\v	vertical tab (0x0B)
\n	newline (0x0A)
\r	return (0x0D)
\b	back space (0x08)
\f	form feed (0x0C)
\a	bell (0x07)
\e	escape (0x1B)
\nnn	octal char (encoded byte value)
\xHH	hexadecimal char (encoded byte value)
\x{7HHHHHHH}	wide hexadecimal char (character code point value)
\cx	control char (character code point value)
\C-x	control char (character code point value)
\M-x	meta (x 0x80) (character code point value)
\M-\C-x	meta control char (character code point value)

(* \b is effective in character class [...] only)

3. Character types

.

any character (except newline)

\w

word character

Not Unicode:

alphanumeric, "_" and multibyte char.

Unicode:

General_Category -- (Letter|Mark|Number|Connector_Punctuation)

\W

non word char

\s

whitespace char

Not Unicode:

\t, \n, \v, \f, \r, \x20

Unicode:

0009, 000A, 000B, 000C, 000D, 0085(NEL),

General_Category -- Line_Separator

-- Paragraph_Separator

-- Space_Separator

\S

non whitespace char

\d

decimal digit char

Unicode: General_Category -- Decimal_Number

\D

non decimal digit char

\h

hexadecimal digit char [0-9a-fA-F]

\H

non hexadecimal digit char

Character Property

* \p{property-name}
 * \p{^property-name} (negative)
 * \P{property-name} (negative)

property-name:

+ works on all encodings
 Alnum, Alpha, Blank, Cntrl, Digit, Graph, Lower, Print, Punct, Space, Upper, XDigit,
 Word, ASCII,

4. Quantifier

greedy

? 1 or 0 times
 * 0 or more times
 + 1 or more times
 {n,m} at least n but not more than m times
 {n,} at least n times
 {,n} at least 0 but not more than n times ({0,n})
 {n} n times

reluctant

?? 1 or 0 times
 *? 0 or more times
 +? 1 or more times
 {n,m}? at least n but not more than m times
 {n,}? at least n times
 {,n}? at least 0 but not more than n times (== {0,n}?)

possessive (greedy and does not backtrack after repeated)

?+ 1 or 0 times
 *+ 0 or more times
 ++ 1 or more times

({n,m}+, {n,}+, {n}+ are possessive op. in ONIG_SYNTAX_JAVA only)

ex. /a*+/ === /(?!>a*)/

5. Anchors

^ beginning of the line
 \$ end of the line
 \b word boundary
 \B not word boundary
 \A beginning of string
 \Z end of string, or before newline at the end
 \z end of string
 \G matching start position (*)

6. Character class

`^...` negative class (lowest precedence operator)
`x-y` range from x to y
`[...]` set (character class in character class)
`..&&..` intersection (low precedence at the next of `^`)

ex. `[a-w&&[^c-g]z] ==> ([a-w] AND ([^c-g] OR z)) ==> [abh-w]`

* If you want to use '[', '-', ']' as a normal character in a character class, you should escape these characters by '\'.

POSIX bracket (`[[:xxxx:]]`, negate `[[:^xxxx:]]`)

Not Unicode Case:

<code>alnum</code>	alphabet or digit char
<code>alpha</code>	alphabet
<code>ascii</code>	code value: [0 - 127]
<code>blank</code>	<code>\t, \x20</code>
<code>cntrl</code>	
<code>digit0-9</code>	
<code>graph</code>	include all of multibyte encoded characters
<code>lower</code>	
<code>print</code>	include all of multibyte encoded characters
<code>punct</code>	
<code>space</code>	<code>\t, \n, \v, \f, \r, \x20</code>
<code>upper</code>	
<code>word</code>	alphanumeric, "_" and multibyte characters
<code>xdigit</code>	0-9, a-f, A-F

Unicode Case:

<code>alnum</code>	Letter Mark Decimal_Number
<code>alpha</code>	Letter Mark
<code>ascii</code>	0000 - 007F
<code>blank</code>	Space_Separator 0009
<code>cntrl</code>	Control Format Unassigned Private_Use Surrogate
<code>digit</code>	Decimal_Number
<code>graph</code>	<code>[[:^space:]] && ^Control && ^Unassigned && ^Surrogate</code>
<code>lower</code>	Lowercase_Letter
<code>print</code>	<code>[[:graph:]] [[:space:]]</code>
<code>punct</code>	Connector_Punctuation Dash_Punctuation Close_Punctuation Final_Punctuation Initial_Punctuation Other_Punctuation Open_Punctuation
<code>space</code>	Space_Separator Line_Separator Paragraph_Separator 0009 000A 000B 000C 000D 0085
<code>upper</code>	Uppercase_Letter
<code>word</code>	Letter Mark Decimal_Number Connector_Punctuation
<code>xdigit</code>	0030 - 0039 0041 - 0046 0061 - 0066 (0-9, a-f, A-F)

7. Extended groups

<code>(?#...)</code>	comment
<code>(?imx-imx)</code>	option on/off
	i: ignore case
	m: multi-line (dot(.) match newline)

(?imx-imx:subexp) x: extended form
option on/off for subexp

(?:subexp) not captured group
(subexp) captured group

(?=subexp) look-ahead
(?!subexp) negative look-ahead
(?<=subexp) look-behind
(?<!=subexp) negative look-behind

Subexp of look-behind must be fixed character length. But different character length is allowed in top level alternatives only.
ex. (?<=a|bc) is OK. (?<=aaa(?:b|cd)) is not allowed.

In negative-look-behind, captured group isn't allowed, but shy group(?:) is allowed.

(?>subexp) atomic group
don't backtrack in subexp.

(?<name>subexp) define named group
(All characters of the name must be a word character. And first character must not be a digit or upper case)
Not only a name but a number is assigned like a captured group.

Assigning the same name as two or more subexps is allowed. In this case, a subexp call can not be performed although the back reference is possible.

8. Back reference

\n back reference by group number (n >= 1)
\k<name> back reference by group name
\k'name' back reference by group name

In the back reference by the multiplex definition name, a subexp with a large number is referred to preferentially. (When not matched, a group of the small number is referred to.)

* Back reference by group number is forbidden if named group is defined in the pattern and ONIG_OPTION_CAPTURE_GROUP is not set.

Back reference with nest level

\k<name+n> n: 0, 1, 2, ...
\k<name-n> n: 0, 1, 2, ...
\k'name+n' n: 0, 1, 2, ...
\k'name-n' n: 0, 1, 2, ...

Destinate relative nest level from back reference position.

ex 1.

```
^A(?:<a>|.|(?:(<b>.)\g<a>\k<b+0>))\z/.match("reer")
```

ex 2.

```

r = Regexp.compile(<<'__REGEXP__'.strip, Regexp::EXTENDED)
(?<element> \g<stag> \g<content>* \g<etag> ){0}
(?<stag> < \g<name> \s* > ){0}
(?<name> [a-zA-Z_]+ ){0}
(?<content> [^<&]+ (\g<element> | [^<&]+)* ){0}
(?<etag> </ \k<name+1> > ){0}
\g<element>
__REGEXP__

p r.match('<foo>f<bar>bbb</bar>f</foo>').captures

```

9. Subexp call ("Tanaka Akira special")

```

\g<name>    call by group name
\g'name'    call by group name
\g<n>       call by group number (n >= 1)
\g'n'       call by group number (n >= 1)

```

* left-most recursive call is not allowed.

ex. (?<name>a\g<name>b) => error

(?<name>a\b\g<name>c) => OK

* Call by group number is forbidden if named group is defined in the pattern and ONIG_OPTION_CAPTURE_GROUP is not set.

* If the option status of called group is different from calling position then the group's option is effective.

ex. (?-i:\g<name>)(?i:(?<name>a)){0} match to "A"

10. Captured group

Behavior of the no-named group (...) changes with the following conditions. (But named group is not changed.)

case 1. /.../ (named group is not used, no option)

(...) is treated as a captured group.

case 2. /.../g (named group is not used, 'g' option)

(...) is treated as a no-captured group (?:...).

case 3. /..(?<name>..)/ (named group is used, no option)

(...) is treated as a no-captured group (?:...).
numbered-backref/call is not allowed.

case 4. /..(?<name>..)/G (named group is used, 'G' option)

(...) is treated as a captured group.
numbered-backref/call is allowed.

where

g: ONIG_OPTION_DONT_CAPTURE_GROUP
 G: ONIG_OPTION_CAPTURE_GROUP

A-1. Syntax dependent options

- + RUBY
 (?m): dot(.) match newline
- + PERL and JAVA
 (?s): dot(.) match newline
 (?m): ^ match after newline, \$ match before newline

A-2. Original extensions

- + hexadecimal digit char type \h, \H
- + named group (?<name>...)
- + named backref \k<name>
- + subexp call \g<name>, \g<group-num>

A-3. Missing features compared with perl 5.8.0

- + \N{name}
- + \l, \u, \L, \U, \X, \C
- + (?{code})
- + (??{code})
- + (?(condition)yes-pat|no-pat)
- * \Q...\E
 This is effective on PERL and JAVA.

5.5 XML in Take Command

TCC provides the ability to open, parse and close XML documents through the use of a subset of the XPath language. The syntax of the language is explained more clearly at the W3C site: http://www.w3schools.com/xpath/xpath_syntax.asp. We support a limited subset of the language explained below. Any functions of the language beyond what are listed below are unsupported, but may work.

The most common use of XML in **TCC** is to parse an XML formatted data file to extract elements for further processing in Take Command.

There are four **TCC** functions that provide XML support –

1. **@XMLOPEN** - open an XML file for use by @XMLXPath and/or @XMLNODES. The syntax is:

```
@XMLOPEN[filename]
Example: set a=%@XMLOPEN(bookstore.xls)
```

2. **@XMLCLOSE** - close an XML file previously opened by @XMLOPEN. The syntax is:

```
@XMLCLOSE[]
```

Example: set a=%@XMLCLOSE()

3. **@XMLNODES** - return the number of nodes (children) for the specified path in an XML file. The syntax is:

@XMLNODES[["filename"],path]

Example: set a=%@XMLNODES["bookstore.xls",/bookstore] – returns the number of books in the bookstore file (see example file below)

If you don't specify a filename (which **must** be in double quotes. @XMLNODES will use the XML file previously opened by @XMLOPEN.

4. **@XMLXPath** - XML XPath query. (See the XML XPath docs for details on XPath syntax.) The syntax is:

@XMLXPath[["filename"],path]

Example: echo %@XMLXPath[/bookstore/book] – lists all the sub-element values for the first book (see example file below)

If you don't specify a filename (which **must** be in double quotes), @XMLXPath will use the XML file previously opened by [@XMLOPEN](#).

Typical Use of The XML Functions

All discussions in this section refer to the following XML data file.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="jap">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
  <book>
    <title lang="ger">Day Watch</title>
    <price>14.99</price>
  </book>
  <book>
    <title lang="eng">Winston Churchill: An Autobiography</title>
    <price>49.99</price>
  </book>
</bookstore>
```

Each tag in this file is referred to as a node (e.g., bookstore, book, title). The identifiers within a tag are often referred to as an attribute (e.g., lang).

Typically you will use the four commands in the following order:

1. Open the file with @XMLOPEN
2. Get the number of child nodes (records) to process with @XMLNODES
3. Set up a loop from one to the number of records to process the records with @XMLXPath
4. Close the file with @XMLCLOSE

Below is a simple example of batch processing (without error handling J):

```
SET a=%@XMLOPEN[bookstore.xml]
SET b=%@XMLNODES[/bookstore]
DO i = 1 to %b
    SET Title= %@XMLXPath[/bookstore/book[%i]/title]
    SET Price= %@XMLXPath[/bookstore/book[%i]/price]
    ECHO %Title ` costs only ` %Price
ENDDO
SET c=%@XMLCLOSE[]
```

This example:

- Opens the bookstore data file,
- Evaluates the number of books in the bookstore as 4(since the only child of bookstore is books) ,
- Gets the title and price for each book
- Prints each title and price out
- Closes the data file.

XPath offers a large number of processing options in addition to the ones above. We have not tested their syntax. If you wish to test additional functionality and report back to us, we will add that information to our documentation.

5.6 Miscellaneous Reference Information

- [Executable Files and File Searches](#) ^[504]
- [Popup Windows](#) ^[507]
- [Windows System Errors](#) ^[507]
- [ASCII and Key Names](#) ^[515]
- [ANSI X3.64 Command Reference](#) ^[516]
- [Colors and Color Names](#) ^[518]

5.6.1 Executable Files & File Searches

When **TCC** can't find a matching internal command name, it tries to find an executable file whose name matches the command name. (Executable files are typically those with a **.COM** or **.EXE** extension.)

If **TCC** cannot find an executable program to run, it next looks for a matching [batch file](#) ^[130] name. **TCC** looks first for a **.BTM** file, then for a **.CMD** file, then for a **.BAT** file, and finally for a **.REX**, **.REXX**, **.PL**, or **.RB** file (if REXX, Perl, and/or Ruby are enabled).

You can change the list of extensions that are considered "executable", and the order in which they are searched, with the [PATHEXT](#) ^[369] environment variable, and the related [PathExt](#) ^[47] configuration option. PATHEXT is supported for compatibility reasons but should not generally be used as a substitute for [executable extensions](#) ^[91], which are more flexible.

Note: If the search for an external program or batch file fails, **TCC** checks to see if the command name matches the name of a file with an [executable extension](#) ^[91]. If an executable extension is found, **TCC** runs the program specified when the association was defined. If no executable extension is found, **TCC** 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, **TCC** calls the ShellExec Windows API. **TCC** 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.

TCC 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 **TCC** (and some applications) search for executable files. For example, if you wanted **TCC** 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
```

The directory names in the search path are separated by semicolons.

You can create or view the search path with the [PATH](#)^[286] command. You can use the [ESET](#)^[222] 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.

Take Command also searches the \WINDOWS\SYSTEM32 directory followed by the \WINDOWS directory. (The actual directory names may be different on your system. **TCC** 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, **TCC** 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, **TCC** 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](#)^[286] command for details.)

If you include an extension as part of the command name, **TCC** only searches for a file with that extension. Similarly, if you include a path as part of the command name, **TCC** 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 Linux syntax where forward slashes are used in command paths, and want **TCC** to recognize this approach, you can set the [Unix/Linux Paths](#)^[47] configuration option.

Once the file is found, **TCC** 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 **TCC**, 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 **TCC** (**.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 **Take Command** will request that Windows start the application for you. See [Windows File Associations](#)^[506] for additional details on how to control Windows file associations in **TCC**.

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

Command	TCC Search Sequence
WP	Search for any executable file whose base name is <i>WP</i> .
WP.EXE	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	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 **TCC** 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](#)^[154] 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, **TCC** displays an "Unknown command" error message and waits for your next instruction.

See also: the [WHICH](#)^[367] command.

5.6.1.1 Windows File Associations

Windows includes the ability to associate file extensions with specific applications; this feature is sometimes called "file associations". For example, a graphics program might be associated with files with a *.JPG* 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, **TCC** 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, **TCC** passes the command name to Windows to see if Windows can find an association for it.

TCC 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](#)^[162] command modifies or displays the associations between extensions and file types in the Windows registry. The [FTYPE](#)^[247] command modifies or displays the default command used to "open" a file of a specified type.

Executable extensions defined in **TCC** always take precedence over file associations defined in Windows. For example, if you associate the *.TXT* extension with your own editor using a **TCC** 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 **TCC**.

See also: [START](#)^[337], [ASSOC](#)^[162], [FTYPE](#)^[247], [Executable Extensions](#)^[91], [Executable Files and File Searches](#)^[504].

5.6.2 Popup Windows

Several features of **TCC** 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](#) ^[73], or a list created by the [SELECT](#) ^[312] command or the [@SELECT](#) ^[458] 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 with the [Pop-Up Windows](#) ^[49] configuration options. A few popup windows (e.g., the extended directory search window) have separate choices in the configuration dialogs. You can also change the keys used in popup windows with [key mapping directives](#) ^[28].

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

Up Arrow	Move the selection bar up one line
Down Arrow	Move the selection bar down one line
Left Arrow	Scroll the display left 1 column, if it is a scrolling display (i.e. if it has a horizontal scrollbar)
Right Arrow	Scroll the display right 1 column, if it is a scrolling display (i.e. if it has a horizontal scrollbar)
PgUp	Scroll the display up one page
PgDn	Scroll the display down one page
Home	Go to the beginning of the list
End	Go to the end of the list
Esc ^[31]	Close the window without making a selection
Enter ^[41]	Select the current item and close the window
Ctrl-E ^[40]	Edit the current selection
Ctrl-D ^[40]	Delete the current selection

Note: The keystrokes shown above are the defaults values. See [Key Mapping Directives](#) ^[28] 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, **TCC** beeps and does not move the cursor bar. To reset the search string, press Backspace.

5.6.3 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](#) ^[392], and the corresponding text is that provided by Microsoft.

Code	Error Message
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 (*tcmd.chm*) instead!

d.

5.7 ASCII and Key Names

For ASCII codes and key names see:

- ▶ [ASCII Table](#) ⁵¹¹
- ▶ [Keys & Key Names](#) ⁵¹⁵

The remainder of this section gives an explanation of the ASCII character sets and key names. For more information on **TCC's** ANSI X3.64 string support see [ANSI X3.64 Commands Reference](#)^[516]. 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 (determined by your keyboard hardware);
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.

If the key codes produced by your keyboard, the code page, and the font you choose are not fully compatible, 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.

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**.

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. See "[Command Line Editing](#)"^[104] for additional information.

5.7.1 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](#)^[510].

- [Control Characters 0 - 31, 127](#)^[512]
- [Printing Characters 32 - 47](#)^[512]
- [Printing Characters 48 - 63](#)^[513]

- [Printing Characters 64 - 79](#)^[513]
- [Printing Characters 80 - 95](#)^[514]
- [Printing Characters 96 - 111](#)^[514]
- [Printing Characters 112 - 126](#)^[514]

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	n/a	DEL	delete

Printing Characters 32 - 47

Dec	Hex	Char	Special character name
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

Dec	Hex	Char	Special character name
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

Dec	Hex	Char	Special character name
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

Dec	Hex	Char	Special character name
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

Dec	Hex	Char	Special character name
096	60	`	accent grave (back tick or back quote)
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

5.7.2 Key Names

Key names are used by **Take Command** in [tab toolbar](#)^[62] buttons, and in **TCC** to define keystroke [aliases](#)^[154], in [key mapping directives](#)^[28], and in the [KEYSTACK](#)^[262] command. The format of a key name is the same in all four cases:

[Prefix-]Keyname

The valid prefix and keyname combinations are shown in the table below. Names of keys must be spelled exactly as shown, except for case. Note that you cannot specify a punctuation key.

Prefix	Valid for keynames
none	A-Z, 0-9, F1-F12, Tab, Bksp, Enter, Up, Down, Left, Right, PgUp, PgDn, Home, End, Ins, Del, Esc
Alt-	A-Z, 0-9, F1-F12, Bksp, and the non-alphanumeric keys `-=[]\;',./
Ctrl-	A-Z, F1-F12, Tab, Bksp, Enter, Up, Down, Left, Right, PgUp, PgDn, Home, End, Ins, Del
Shift-	F1-F12, Tab

The prefix and key name must be separated by a hyphen (-). For example:

Alt-F10 ctrl-bksp

Some keys are intercepted by Windows and are not passed on to **Take Command**. For example, **Alt-Tab**, **Alt-Esc** and **Ctrl-Esc** typically pop up a task list, or are used in switching among multiple tasks. **Alt-space** brings down a menu to control window size and position, etc. Keys which are intercepted by the operating system (including menu accelerators, i.e. **Alt** plus another key) generally cannot be assigned to [aliases](#)^[154] or with [key mapping directives](#)^[28], because **Take Command** never receives these keystrokes. However, [KEYSTACK](#)^[262] can send them to Windows (though not to another application).

The above comments are based on common 101/102-key US-style keyboards. Some key combinations might not be available on some keyboards.

5.8 ANSI X3.64 Command Reference

TCC includes support for ANSI Std X3.64, allowing you to manipulate the cursor, screen color, and other display attributes through sequences of special characters embedded in the text displayed on the screen. These sequences are called "ANSI commands". (For a general description of this feature, see [ANSI Support](#)^[101].) **TCC** cannot provide ANSI X3.64 support to external applications.

TCC supports most common ANSI X3.64 screen commands, but does not provide the complete set of options supported by some operating system's ANSI X3.64 drivers (for example, **TCC** does not support ANSI X3.64 key substitutions; that functionality is already provided with [key aliases](#)^[128]). This section is a quick reference to the ANSI X3.64 commands supported by **TCC**.

ANSI X3.64 support within **TCC** can be enabled or disabled with the [ANSI Colors](#)^[49] configuration option, or the [SETDOS](#)^[323] /A command. You can test whether ANSI X3.64 support is enabled with the [_ANSI](#)^[382] 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 you can use the [ECHO](#)^[217] 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 internal ["escape" character](#)^[124] (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](#)^[297], using \$e to transmit the <ESC> character.

Commands

The internal **TCC** ANSI X3.64 interpreter 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. The default value for **row**, **rows**, **col**, and **cols** is 1.

<ESC>[rowsA	Cursor up by <i>rows</i>
<ESC>[rowsB	Cursor down by <i>rows</i>
<ESC>[colsC	Cursor right by <i>cols</i>
<ESC>[colsD	Cursor left by <i>cols</i>

<ESC>[row;colH	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>[row;colf	Set cursor position, same as "H" command
<ESC>[attr1;attr2;...m	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** Bright (high intensity) foreground color
- 2** Normal intensity foreground color
- 5** Bright (high intensity) background
- 7** Reverse video
- 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
```

5.9 Colors, Color Names & Codes

You can use color names in several configuration options and in some internal commands. The general form of a color specification is:

```
[BRiGht] fg ON [BRiGht] bg
```

where **fg** is the foreground or text color, and **bg** is the background 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	WHItE
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 **TCC** 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 options such as [ColorDIR](#)^[368], 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](#)^[181] command also supports the CMD.EXE style color specification **bf**, where **b** and **f** are CMD.EXE's codes for background and foreground colors, respectively (shown in the CMD.EXE columns of the table below). The numeric values of these codes are the same as the **TCC** 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		TCC name	TCC codes (decimal)		CMD.EXE codes* (hexadecimal)		ANSI X3.64 codes (decimal)	
normal	bright		norm al	bright	normal	bright	foreground	backgrou nd
black	gray	BLAck	0	8	0	8	30	40
blue	blue	BLUe	1	9	1	9	34	44
green	green	GREen	2	10	2	A	32	42

cyan	cyan	CYAn	3	11	3	B	36	46
red	pink	RED	4	12	4	C	31	41
magenta	magenta	MAGenta	5	13	5	D	35	45
brown	yellow	YELlow	6	14	6	E	33	43
white	white	WHItE	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**^[452] function returns the value of color configuration options by combining both foreground and background into a single number (0-255) using the following logic:

foreground value + (background value * 16) = code

For example, **bright white on red** (15 on 4) can be expressed as:

15 + (4 * 16) = 79

The following batch file translates a combined numeric color code:

```
@echo off
setlocal
function x=`%if[%1 gt 8,bri ,]%@word[%@eval[%1 %% 8],bla blu gre cya red
mag yel whi]`
:loop
input /c /d %=nColor code? %%c
if %c gt 255 .or. %c lt 0 quit
set f=%@eval[%c %% 16] & set b=%@eval[%c \ 16]
echos The color code %c is "%f on %b" ("%@x[%f] on %@x[%b]")
goto loop
```

Color Errors

A standard color specification allows sixteen foreground and sixteen background colors. 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 worse when running in windowed mode, because Windows 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 **Take Command** color specifications.

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 **Take Command** are typically discussed in more detail in other areas of this help file.

Select the glossary section you wish to review:

[A](#)^[520] [B](#)^[521] [C](#)^[522] [D](#)^[525] [E](#)^[526] [F](#)^[528] [G](#)^[528] [H](#)^[529] [I](#)^[529] [J](#)^[530] [K](#)^[530] [L](#)^[530] [M](#)^[531] [N](#)^[531] [O](#)^[532] [P](#)

[Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#)

6.1 Glossary - A

Access Date - In the Windows file systems it is the later of the dates on which a file was created or last accessed for reading. The [VFAT](#) and [NTFS](#) file systems save this file property.

Access Time - In the 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](#) 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.

Age - A special term used by the *Take Command* documentation to reference one of its methods to represent points in time (epochs). An **age** is the time elapsed since 1601-01-01 00:00:00 (local time) as a multiple of 100 ns.

Alias - A *TCC* feature which allows you to create shorthand name for a command or series of commands. For details see [Aliases](#).

Alias Argument - Same as [Alias Parameter](#).

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](#).

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](#). It is composed of various professional organizations, including EIA and IEEE, many of which also define standards.

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 the table below.

condition 1	condition 2	result
false	false	false
false	true	false
true	false	false
true	true	true

ANSI - [American National Standards Institute](#). The acronym **ANSI** in software development is often used as a short-hand reference to the standard [ANSI X3.64](#).

ANSI X3.64 - A standard specifying sequences of characters which control colors on the screen, manipulate the cursor and screen contents, and redefine keys. *TCC* includes support for a selected subset. This support may be enabled or disabled at any time. For more details see [ANSI X3.64 Command Reference](#).

Append - Concatenation of one file or string onto the end of another.

APM - See [Advanced Power Management](#).

Application - A program run from the command prompt or a batch file. Used broadly to mean any program other than **Take Command** 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](#)^[532].

ASCII - Acronym for *American National Standard Code for Information Interchange*, defined in the standard [ANSI](#)^[520] 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. See [ASCII file](#)^[521].

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](#)^[535].

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](#)^[533], [Hidden](#)^[529], [System](#)^[535], and [Archive](#)^[521]. Unmodifiable attributes include [Directory](#)^[525] and [Volume Label](#)^[536].

Automatic Directory Change - A **TCC** 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 **TCC** starts or terminates: [TCSTART](#)^[535] and [TCEXIT](#)^[535]. See [Automatic Startup and Termination Programs](#)^[22] for more details.

6.2 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. On most systems the BIOS is stored in read-only memory inside the PC.

BAT File - Same as [Batch File](#)^[521].

Batch File - A [text file](#)^[535] containing a sequence of commands for **TCC** 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**.

Batch File Argument - Same as [Batch File Parameter](#)^[521].

Batch File Parameter - A numbered variable (e.g. %2) used within a [batch file](#)^[521], 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](#)^[521].

Binary File - A file containing information which does not represent or cannot sensibly be interpreted as text. See also [ASCII File](#)^[521], [text file](#)^[535].

BIOS - [Basic Input Output System](#)^[521].

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. See also [Character Device](#)^[523].

Boot - Ungrammatical contraction of [Bootling](#)^[522], shortened from [Bootstrapping](#)^[522].

Boot Directory - The current directory at the time the system is started, usually the [root directory](#)^[534] of the [boot drive](#)^[522].

Boot Drive - The disk drive that the operating system is loaded from during the boot process, usually A: (the floppy disk) or C: (the hard disk), or on some systems a CD or DVD drive. In some diskless workstations it may be a network drive.

Bootling - shortened form of [Bootstrapping](#)^[522].

Bootstrapping - 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](#)^[523], [Cold Reboot](#)^[523], [Reboot](#)^[533], and [Warm Reboot](#)^[536].

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](#)^[521], unique to JP Software products, which is loaded into a [buffer](#)^[522] 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. **TCC** uses many buffers for its internal operations, e.g., to store [aliases](#)^[128].

6.3 Glossary - C

Carriage Return (CR) - A control character ([ASCII](#)^[511]: 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](#)^[535]s at the end of each line. See also [ASCII](#)^[511], [End of Line](#)^[526].

CD-ROM File System (CDFS) - The file system which supports CD-ROM / CD-RW 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](#)^[521], [Code Page](#)^[523], [Unicode](#)^[536].

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. See also [Block Device](#)^[522].

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. See also [Graphic Mode](#)^[529].

CMD File - A [batch file](#)^[521] designed for execution by **CMD.EXE** or **TCC**.

CMDLINE - An [environment variable](#)^[526] 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](#)^[524].

Cold Boot - The process of starting the computer by turning on its power. See also [Boot](#)^[522], [Cold Reboot](#)^[523], [Reboot](#)^[533], [Warm Reboot](#)^[536].

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](#)^[521]. See also [Boot](#)^[522], [Cold Boot](#)^[523], [Reboot](#)^[533], [Warm Reboot](#)^[536].

Command Completion - A **TCC** 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 **TCC** feature which displays commands as they are executed. Echoing can be turned on and off.

Command Grouping - A **TCC** feature which allows you to enclose a group of several commands within parentheses, and have them treated as a single command.

Command History - A **TCC** 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 **TCC** to display the [command history](#)^[523], allowing you to choose a previous command to modify and/or execute.

Command Interpreter - Same as [Command Processor](#)^[523].

Command Line Expansion - The process **TCC** 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 variables. See also [Parsing](#)^[532].

Command Processor - A program which interprets commands entered interactively or stored in a file (referred to as a [batch file](#)^[521] in the environment discussed here), and executes other programs. Also called [Command Interpreter](#)^[523].

Command Prompt - A prompt displayed on the screen by **TCC** when it is ready for another command

to be entered and executed.

Command Recall - See [Command History](#)^[523].

Command Separator - One or more characters used to separate multiple commands on the same command line. See [Multiple Commands](#)^[120].

Command Tail - The contents of the command line after removing the command name. See [Separating the command from its command tail](#)^[125].

Compound Command - See [Multiple Commands](#)^[120].

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](#)^[532] provides an attribute to indicate whether or not a file is compressed. Accessing compressed files requires slightly more processor time to perform the compression and expansion, but this is often compensated by the reduced data transfer time.

COMSPEC - An [environment variable](#)^[526] which defines where to find the character-mode command processor to start a secondary shell.

Condition - short for [conditional expression](#)^[109].

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](#)^[527].

Conditional Expression - See [Conditional Expression](#)^[109].

Console - See [Character Mode](#)^[523].

Console Mode - See [Character Mode](#)^[523].

Control Character - A character which does not have a printable or displayable representation, but, when encountered in a stream of characters, performs a specific action. In [ASCII](#)^[511] 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](#)^[511] can be generated by pressing the **Ctrl** key simultaneously with another key. For more information see [ASCII, Key Codes & ANSI Commands](#)^[510].

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](#)^[523].

CPU - See [Central Processing Unit](#)^[522].

CR - See [Carriage Return](#)^[522].

CRC - See [Cyclic Redundancy Code](#)^[525].

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**.

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](#)^[525].

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.

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.

6.4 Glossary - D

Date Range - A **TCC** feature which allows you to select files based on the date and time they were last modified. For details, see [Date Ranges](#)^[82].

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](#)^[528]. See also [Time Stamp](#)^[495].

Default Directory - Same as [Current Directory](#)^[525].

Default Drive - Same as [Current Drive](#)^[525].

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 **TCC**. A string of characters may be assigned to describe a file, using the [DESCRIBE](#)^[195] 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](#)^[182] or [MOVE](#)^[274]), 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](#)^[535].

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](#)^[523], and excludes [block devices](#)^[522].

Directive - The name of a configurable **Take Command** or **TCC** option and the value assigned to it. Most directives are set with the [OPTION](#)^[284] command. See also: [initialization file](#)^[530].

Directory :

1) A portion of a disk, identified by a name and a relationship to other directories in a [directory tree](#)^[526] structure, with the tree starting at the [root directory](#)^[534]. 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](#)^[521] of the entry for the **directory** in its [parent directory](#)^[532].

Directory History - A **TCC** 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 the [Directory History](#)^[526].

Directory Junction - same as [soft link](#)^[534].

Directory Stack - A **TCC** feature, implemented through the [PUSHD](#)^[299] and [POPD](#)^[294] commands, which allow you to save the current directory and return to it later. See also [Stack](#)^[535].

Directory Tree - The branching structure of directories on a disk, starting at the [root directory](#)^[534]. 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.

Domain - in mathematics the set of values for which a function is defined.

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.

6.5 Glossary - E

Echo - See [Command Echoing](#)^[523].

End of Line (EOL) - An indication of the end of line in [text file](#)^[535]s. Many conventions exist to represent EOL in ASCII files. The most common ones use one or two control characters: (1) [CR](#)^[522], (2) CR followed by [LF](#)^[530], (3) LF followed by CR, (4) LF. 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 **Take Command**. Others may be set at the command line using the [SET](#)^[319] 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](#)^[357] command, or by setting its value to an empty string. See also [Master Environment](#)^[531] and [Passed Environment](#)^[532].

Environment Variable - The name of a single entry in the [environment](#)^[526].

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](#)^[527].

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 **TCC** is the caret (^). This may be modified using the [Escape character](#)^[124] configuration option or by using the /E option of the [SETDOS](#)^[323] command.

2) A control character, symbol **ESC** ([ASCII](#)^[517]; 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 **Take Command**, 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 **TCC** 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 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>
false	false	false
false	true	true
true	false	true
true	true	false

Exit Code - The result code returned by an external command or an internal command. 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](#)^[526].

Expansion :

- 1) [Command Line Expansion](#)^[523]
- 2) The inverse of [compression](#)^[524].

Extended ASCII Character: A legally invalid phrase, used for a character whose code is in the range 128 to 255, used 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](#)^[527], [code page](#)^[528], and font.

Extended Directory Search - A **TCC** 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 keyboard which has no representation in the standard ASCII character set, such as a function key, cursor key, or **Alt** plus another key.

Extended Parent Directory Names - A **TCC** 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 **TCC** 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](#)^[537].

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, as distinguished from an [internal command](#)^[530], which is executed by **TCC**.

EXTPROC - A **TCC** feature which allows you to designate a specific external program to run a particular batch file.

6.6 Glossary - F

FAT - See [File Allocation Table](#)^[528].

FAT-Compatible File Name - See [SFN](#)^[534].

FAT File System - The file system used by PC-DOS, MS-DOS, and its emulators based on [FAT](#)^[528] 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](#)^[528] size.

FAT16 File System - Each entry in the [FAT](#)^[528] is 16 bits long, limiting table size to 65,536 allocation blocks.

FAT32 File System - Each entry in the [FAT](#)^[528] is 32 bits long, limiting table size to 4,294,967,296 allocation blocks, See also [VFAT File System](#)^[536].

FF - See [Form Feed](#)^[528].

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](#)^[521].

File Description - See [Description](#)^[525].

File Exclusion Range - A **TCC** feature which allows you to exclude files from processing by internal commands. See [File Exclusion Ranges](#)^[85].

Filename Completion - A **TCC** feature which allows you to type part of a filename on the command line, and have **TCC** 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.

6.7 Glossary - G

Global Aliases - A **TCC** option which allows you to store aliases in a global area accessible to all copies of **TCC**, so that any change made by one copy, even between a [SETLOCAL](#)^[326] - [ENDLOCAL](#)^[220] command pair, is immediately effective in all other copies. See also [Local Aliases](#)^[531].

Global Directory History - A **TCC** option which allows you to store the directory history in a global area accessible to all copies of **TCC**, so that any change made by one copy, even between a [SETLOCAL](#) [326] - [ENDLOCAL](#) [220] command pair, is immediately effective in all other copies. See also [Local Directory History](#) [531].

Global History - A **TCC** option which allows you to store the command history in a global area accessible to all copies of **TCC**, so that any change made by one copy, even between a [SETLOCAL](#) [326] - [ENDLOCAL](#) [220] command pair, is immediately effective in all other copies. See also [Local History](#) [531].

Global User Functions - A **TCC** option which allows you to store user-defined functions in a global area accessible to all copies of **TCC**, so that any change made by one copy, even between a [SETLOCAL](#) [326] - [ENDLOCAL](#) [220] command pair, is immediately effective in all other copies. See also [Local User Functions](#) [531].

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. See also [Character Mode](#) [523].

6.8 Glossary - H

Hard link - NTFS gives you the ability to create hard links, which are multiple directory entries referencing a single file body. Creating a hard link causes the system to create an additional directory entry that points to the same file (unlike a shortcut, which is actually an additional file). Files can have multiple hard links, and therefore a single file can appear in multiple directories, with multiple names in each.

A file with more than one hard link can be accessed by any of its directory entries. Each file on an NTFS volume has at least one hard link (connecting a directory entry to the file body). A file is deleted from the file system only after all its hard links (i.e., directory entries) have been deleted.

When a file with more than one directory entry is modified, the file properties (e.g., size, date stamps, etc.) of only the entry actually used for updating the file are updated instantly.

Hard links must be created on the same NTFS volume as the file.

Hidden - A file attribute indicating that the file should not be displayed with a normal [DIR](#) [198] command, and should not be made available to programs unless they specifically request access to hidden files. Often combined with the [System](#) [535] attribute.

History - See [Command History](#) [523].

History Window - See [Command History Window](#) [523] and [Directory History](#) [526].

6.9 Glossary - I

IFS - See [Installable File System](#) [529].

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 **TCC** simply as another drive .

Include List - A method of specifying several files or groups of files in the same directory, for use with

all internal commands which take file names as parameters.

Inheritance - A **TCC** feature which allows one instance of **TCC** to "inherit" the TCMD.INI file data, aliases, user defined functions, environment variables, command history, and directory history from a previous instance. More generally, a system which allows one program to pass information or settings on to another, often to a second instance of the same program.

Initialization Directive - See [Directive](#)^[525].

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 file for **Take Command** is **TCMD.INI**. For additional details, see [Startup Command](#)^[19] and [Initialization Files](#)^[26].

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. See also [Overstrike Mode](#)^[532].

Internal Command - A command which is part of **TCC**. See also [external command](#)^[528].

Internal Variables - Special variables created by **TCC** 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 Standardization Organization (ISO) - An international body promulgating standards.

ISO - See [International Standardisation Organization](#)^[530].

6.10 Glossary - J

Junction - same as [soft link](#)^[534]

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 value, a scan code, or an extended key code.

Key Mapping - A **TCC** feature which allows you to assign new keystrokes for command line functions such as manipulating the command history or completing file names.

Keyboard Buffer - A Windows 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.

6.12 Glossary - L

Label :

- 1) A location marker in a [batch file](#)^[521], with the format **:name**, allowing [CALL](#)^[175], [GOTO](#)^[248] and [GOSUB](#)^[247] commands to "jump" to that point in the file.
- 2) [Volume Label](#)^[536].

Line Feed (LF) - A control character ([ASCII](#)^[511]: 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](#)^[526]. It is not normally entered from the keyboard, but in many cases it can be generated, if necessary, by pressing **Ctrl-J**.

LF - See [Line Feed](#)^[530].

LFN - See [Long File Name](#)^[531].

LFN File System - A file system which supports [long file names](#)^[531]. 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](#)^[531], [Short File Name](#)^[534], [VFAT File System](#)^[536], and [NTFS](#)^[532].

Local Aliases - A **TCC** option which allows you to store aliases in a local area only accessible to the current copy of **TCC** so that a change made in the current copy of **TCC** does not affect other copies, and vice versa. See also [Global Aliases](#)^[528].

Local Directory History - A **TCC** option which allows you to store the directory history in a local area only accessible to the current copy of **TCC**, so that a change made in the current copy of **TCC** does not affect other copies, and vice versa. See also [Global Directory History](#)^[529].

Local History - A **TCC** option which allows you to store the command history in a local area only accessible to the current copy of **TCC**, so that a change made in the current copy of **TCC** does not affect other copies, and vice versa. See also [Global History](#)^[529].

Local User Functions - A **TCC** option which allows you to store user-defined functions in a local area only accessible to the current copy of **TCC**, so that a change made in the current copy of **TCC** does not affect other copies, and vice versa. See also [Global User Functions](#)^[529].

Logging - A **TCC** option, implemented via the [LOG](#)^[269] 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. See also [Short File Name](#)^[534].

6.13 Glossary - M

Master Environment - The master copy of the environment maintained by Windows.

Modulo - The remainder after an integer division. For example 11 modulo 3 is 2.

Multiple Commands - A **TCC** feature which allows multiple commands to be placed on a line, with each command separated by a [command separator](#)^[524].

6.14 Glossary - N

Name - The 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.

New Technology File System (NTFS): A file system distributed with Windows which allows longer file names, supports larger drives, and provides better performance than the [FAT file system](#)^[528].

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 **TCC** operating characteristics.

Option file - Alternate name for [Initialization File](#)^[530].

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 the table below.

<i>condition 1</i>	<i>condition 2</i>	<i>result</i>
false	false	false
false	true	true
true	false	true
true	true	true

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. See also [Insert Mode](#)^[530].

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.

Parent Directory - The directory in which a particular subdirectory is cataloged, often seen as the directory above a subdirectory.

Parsing - The process **TCC** performs to analyze the command line, perform alias, function and 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. See

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](#)^[368], which contains a series of path specifications used when searching for external commands and batch files.
- 3) The internal command [PATH](#)^[286], which redefines the value of the environment variable [PATH](#)^[368].

Pipe - A method for sending the standard output of one command to the standard input of another command, syntactically represented by a vertical bar "|" separating the commands. See also [Redirection](#)^[98].

Pixel - A single dot on the screen. The color of each pixel can be individually controlled.

Previous Working Directory - The working directory used most recently, just prior to selecting the [current working directory](#)^[525]. 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.

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 **Take Command**, use it in a similar manner to delimit text to be used literally (i.e., exactly as is).

6.18 Glossary - R

RAM - [Random Access Memory](#)^[533].

RAM Disk - Another name for [Virtual Disk](#)^[536].

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.

Range (file selection) - See [Date Range](#)^[82], [Description Range](#)^[86], [File Exclusion Range](#)^[85], [Size Range](#)^[82], [Time Range](#)^[84].

Range - the set of possible values of a function.

Read Only - A file attribute indicating that the file can be read, but not written or deleted by the operating system or **TCC** unless special commands are used.

Read Only Memory (ROM) - Physical memory used to store information which cannot be readily modified.

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](#)^[522], [Cold Reboot](#)^[523], and [Warm Reboot](#)^[536].

Redirection - A method for sending the output from a command to a file, or of providing the input for a command from a file. See also [Pipe](#)^[533].

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.

Reparse Point - same as [soft link](#)^[534]

REXX - A file and text processing language developed by IBM, and available on many PC and other platforms.

ROM - [Read Only Memory](#)^[533].

Root Directory - The first directory on a disk, from which all other directories are "descended". The root directory is referenced with a single backslash [\\].

6.19 Glossary - S

Search Path - A semicolon-delimited 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](#)^[530] 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](#)^[523], 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](#)^[528]: a [name](#)^[531] of 0 ..8 characters and an [extension](#)^[527] of 0 .. 3 characters, each consisting of only alphabetic and numeric characters plus the punctuation marks ! # \$ % & ' () - @ ^ _ ` { } and ~. See also [LFN](#)^[531].

VFAT file systems consider a file name that contains a lower case character to be a [LFN](#)^[531], even if it obeys all other rules to a SFN, and assign to it a SFN - the all upper case version of the LFN. NTFS does not assign an SFN to such files.

When you copy an LFN file to another LFN directory by its LFN (using the same name in the target directory), the SFN of the target file may be different from the SFN of the source file.

Size Range - A **TCC** feature which allows you to select files based on their size. For details see [Size Ranges](#)^[82].

Soft Link - also known as a directory junction or reparse point - is a special file. When created, its content is the name of a directory on any volume. Any reference to the contents of the soft link act on the contents of the directory it references, including deleting of files from the soft link, which deletes the files from the referenced directory.

Soft links can be chained, i.e., a soft link **A** can refer to another soft link **B**. Changing the referenced directory of **B** effectively changes the contents of **A**.

If the directory referenced by a soft link is deleted, references to the soft link's contents fail. If later another directory is created with the same name, the soft link will function again. Similarly, if the reference is to a removable volume, it will reference whatever volume is present at the time of access.

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](#) ^[525].

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 (stderr) - The file or character device where a program displays error messages. It defaults to the screen. It may be [redirected](#) ^[98] to a file, or [piped](#) ^[101] to another program.

Standard Input (stdin) - The file or character device whence a program obtains its normal input. It defaults to the keyboard. It may be [redirected](#) ^[98] to a file, or [piped](#) ^[101] to another program.

Standard Output (stdout) - The file or character device where a program displays its normal output. It defaults to the screen. It may be [redirected](#) ^[98] from a file, or [piped](#) ^[101] from another program.

Stderr - short for [Standard Error](#) ^[535].

Stdin - short for [Standard Input](#) ^[535].

Stdout - short for [Standard Output](#) ^[535].

Subdirectory - Any directory other than the [root directory](#) ^[534].

Subtree - See [Directory Tree](#) ^[526].

Switch - Alternate name for [Option](#) ^[532]; also the name of the program control command [SWITCH](#) ^[335].

System - A file attribute indicating that the file belongs to the operating system, and should not be accessed by other programs. Most files with this attribute also has the [Hidden](#) ^[529] attribute.

6.20 Glossary - T

Target: See [Destination](#) ^[525].

TCEXIT: A program which is executed whenever **TCC** exits. See [Automatic Startup and Termination Programs](#) ^[22] for more details.

TCSTART: A program which is executed whenever **TCC** starts. See [Automatic Startup and Termination Programs](#) ^[22] for more details.

Text file: A file containing only characters that are either displayable, or which affect the display format (format effectors). The characters of the file are represented by their character codes. **TCC** provides support for two systems of character codes, [ASCII](#) ^[511] and [Unicode](#) ^[536].

Time Range: A **TCC** feature which allows you to select files based on the time they were last modified, created, or accessed. See [Time ranges](#) ^[84] 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](#) ^[525].

Tree: See [Directory Tree](#) ^[526].

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>). Windows 2000 introduced an implementation of 16-bit *UNICODE* (65536 potential distinct values) intended to facilitate use of non-English languages. The first two bytes of Unicode text files in NTFS systems contain a signature representing the file encoding.

UDF - User Defined Function.

User Defined Function - A function defined by the user utilizing the [FUNCTION](#)^[242] command to manipulate strings, dates, and filenames; perform arithmetic; read and write files; and perform other similar functions. UDFs are invoked the same way as [Variable functions](#)^[395].

6.22 Glossary - V

Variable - See [Alias Parameter](#)^[520], 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 **TCC** to manipulate strings, dates, and filenames; perform arithmetic; read and write files; and perform other similar functions. Variable functions are similar to 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](#)^[528] which supports [long filenames](#)^[531].

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. Also called a **RAM Disk**.

Volume - See **Disk Drive**.

Volume Label - A character string stored on a disk drive or volume used to identify it. Windows uses a special, hidden file name in the disk root directory for this purpose.

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 parameters on the command line. The white space characters recognized by **TCC** are the space, 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**.

6.24 Glossary - X

X-3.64 - See [ANSI X-3.64](#)⁵²⁰

XOR - [Exclusive OR](#)⁵³⁷

7 Copyright & Version

Take Command 9.0 for Microsoft Windows XP / 2003 / Vista / 2008
Software: Copyright © 2008, Rex Conn and JP Software Inc.
All Rights Reserved.

Version 9.0 Help System
Help text: Copyright © 2008 JP Software Inc.
All Rights Reserved.

Language translations by Christian Albaret (French) and Hans-Peter Grözinger (German)

We gratefully acknowledge the contributions of Roger Byrne, Vincent Fatica and our other users.

This help material was last revised on Tuesday, January 29, 2008

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

- ! -

! 380
! range exclusion 80
!= inequality test operator 109

- \$ -

\$ metacharacter 297
\$ parameter 133

- & -

& ampersand 126
&& 120

- (-

() parentheses 109, 121

- * -

* (disable alias) 120
* (wildcard) 77
* parameter 133

- . -

.AND. 109
.BAT and 4NT 131
.BAT extension 131
.BTM extension 131
.CMD extension 131
.INI 388
.INI files 24
.OR. 109
.XOR. 109

- ? -

? (command) 152

? (variable) 380, 395
? (wildcard) 77

- @ -

@ at sign 90, 106
@ABS 407
@AFSCCELL 407
@AFSMOUNT 407
@AFSPATH 407
@AFSSYMLINK 407
@AFSVOLID 408
@AFSVOLNAME 408
@AGEDATE 408
@ALIAS 408
@ALTNAME 408
@ASCII 409
@ASSOC 162, 241, 409
@ATTRIB 86, 409
@AVERAGE 410
@CAPI 410
@CAPS 411
@CDROM 411
@CEILING 411
@CHAR 411
@CLIP 412
@CLIPW 412
@COLOR 412
@COMMA 412
@COMPARE 413
@CONSOLE 413
@CONVERT 413
@COUNT 413
@CRC32 413
@CWD 414
@CWDS 414
@DATE 414
@DAY 414
@DEC 415
@DECIMAL 415
@DESCRIPT 415
@DEVICE 416
@DIGITS 416
@DIRSTACK 416
@DISKFREE 416
@DISKTOTAL 417
@DISKUSED 417
@DOMAIN 417

@DOW	417	@IDOWF	439
@DOWF	418	@IF	109, 439
@DOWI	418	@INC	440
@DOY	419	@INDEX	440
@DRIVETYPE	419	@INIREAD	441
@DRIVETYPEEX	419	@INIWRITE	441
@ENUMSERVERS	419	@INODE	442
@ENUMSHARES	420	@INSERT	442
@ERRTEXT	420	@INSTR	443, 463
@EVAL	415, 420, 440	@INT	443
@EXEC	424	@IPADDRESS	443
@EXECSTR	386, 424	@IPNAME	444
@EXETYPE	424	@ISALNUM	444
@EXPAND	425	@ISALPHA	444
@EXT	425	@ISASCII	444
@FIELD	426	@ISCNTRL	445
@FIELDS	427	@ISDIGIT	445
@File List	90	@ISPRINT	445
@FILEAGE	427	@ISPROC	445
@FILECLOSE	427	@ISPUNCT	445
@FILEDATE	428	@ISSPACE	446
@FILENAME	428	@ISXDIGIT	446
@FILEOPEN	428	@JUNCTION	446
@FILEREAD	429	@LABEL	446
@FILEREADB	429	@LCS	446
@FILES	430	@LEFT	446
@FILESEEK	431	@LEN	447
@FILESEEKL	431	@LFN	447
@FILESIZE	432	@LINE	447
@FILETIME	432	@LINES	448
@FILEWRITE	433	@LINKS	448
@FILEWRITEB	433	@LOWER	448, 465
@FINDCLOSE	434	@LTRIM	448
@FINDFIRST	434	@MAKEAGE	449
@FINDNEXT	435	@MAKEDATE	449
@FLOOR	435	@MAKETIME	449
@FORMAT	435	@MAX	450
@FORMATN	436	@MD5	450
@FSTYPE	436	@MIN	450
@FTYPE	162, 241, 437	@MONTH	450
@FULL	437	@NAME	451
@FUNCTION	437	@NUMERIC	451
@GETDIR	437	@OPTION	452
@GETFILE	438	@OWNER	452
@GETFOLDER	438	@PATH	452
@GROUP	417, 438	@PERL	142, 452
@HISTORY	439	@PING	453
@IDOW	439	@QUOTE	453

@RANDOM 453
 @READSCR 453
 @READY 453
 @REGCREATE 454
 @REGDELKEY 454
 @REGEX 454, 496
 @REGEXINDEX 454, 496
 @REGEXIST 454
 @REGEXSUB 455, 496
 @REGQUERY 455
 @REGSET 455
 @REGSETENV 455
 @REGTYPE 455
 @REMOTE 456
 @REMOVABLE 456
 @REPEAT 456
 @REPLACE 456
 @REVERSE 457
 @REXX 142, 457
 @RIGHT 457
 @RTRIM 457
 @RUBY 143, 391, 458
 @SCRIPT 458
 @SEARCH 458
 @SELECT 312, 458
 @SERIAL 459
 @SERVER 459
 @SFN 460
 @SHA1 460
 @SHA256 460
 @SHA384 460
 @SHA512 460
 @SHFOLDER 461
 @SIMILAR 462
 @SNAPSHOT 462
 @STRIP 462
 @SUBST 463
 @SUBSTR 443, 463
 @SUMMARY 462
 @SYMLINK 463
 @TIME 463
 @TIMER 347, 463
 @TRIM 464
 @TRUENAME 353, 464
 @TRUNCATE 464
 @UNC 464
 @UNICODE 464
 @UNIQUE 465

@UNQUOTE 465
 @UNQUOTES 465
 @UPPER 465
 @VERINFO 465
 @WATTRIB 86, 466
 @WILD 466
 @WINAPI 294, 410, 467
 @WINCLASS 467
 @WINEXENAME 467
 @WININFO 384, 467
 @WINMEMORY 468
 @WINMETRICS 468
 @WINPOS 470
 @WINSTATE 470
 @WINSYSTEM 470
 @WMI 472
 @WORD 472
 @WORDS 473
 @WORKGROUP 473
 @XMLCLOSE 473
 @XMLNODES 473
 @XMLOPEN 474
 @XMLXPath 474
 @YEAR 474

- [-

[] (wildcard) 77

- \ -

\ backslash 117

■ ^ ■

^ caret 124, 126

■ _ ■

_? 380

_4VER 381

_ACSTATUS 381

_ADMIN 381

_AFSWCELL 381

_ALT 381

_ANSI 382

_BATCH 382

_BATCHLINE	382	_EXIT	386
_BATCHNAME	382	_EXPANSION	387
_BATCHTYPE	382	_FG	387
_BATTERY	382	_FTPERROR	387
_BATTERYLIFE	382	_HDRIVES	387
_BATTERYPERCENT	382	_HLOGFILE	387
_BDEBUGGER	382	_HOST	387
_BG	382	_HOUR	387
_BOOT	383	_HWPROFILE	388
_BUILD	383	_IDLETICKS	388
_CAPSLOCK	383	_IDOW	388
_CDROMS	383, 386	_IDOWF	388
_CHILDPID	383	_IFTP	388
_CI	383	_IFTPS	388
_CMDLINE	383	_IMONTH	388
_CMDPROC	383	_IMONTHF	388
_CMDSPEC	383	_ININAME	388
_CO	383	_IP	388
_CODEPAGE	383	_ISODATE	388
_COLUMN	384	_KBHIT	388
_COLUMNS	384	_LALT	388
_CONSOLEPIDS	384	_LASTDISK	389
_COUNTRY	384	_LCTRL	389
_CPU	384	_LOGFILE	389
_CPUUSAGE	384	_LSHIFT	389
_CTRL	384	_MINUTE	389
_CWD	384	_MONITORS	389
_CWDS	385	_MONTH	389
_CWP	385	_MONTHF	389
_CWPS	385	_NUMLOCK	389
_DATE	385	_OPENAFS	390
_DATETIME	385	_OSBUILD	390
_DAY	385	_PARENT	390
_DETACHPID	385	_PID	390
_DISK	385	_PIPE	390
_DNAME	195, 385	_PPID	390
_DOS	385	_RALT	390
_DOSVER	386	_RCTRL	390
_DOW	386	_READY	390
_DOWF	386	_REGISTERED	390
_DOWI	386	_ROW	390
_DOY	386	_ROWS	391
_DRIVES	386	_RSHIFT	391
_DST	386	_RUBYTYPE	391, 458
_DVDS	383, 386	_RUBYVALUE	391, 458
_ECHO	386	_SCROLLLOCK	391
_EDITMODE	386	_SECOND	391
_EXECSTR	386	_SELECTED	391

_SHELL 391
 _SHELLS 391
 _SHIFT 391
 _SHORTCUT 392
 _SHRALIAS 392
 _STARTPATH 392
 _STARTPID 392
 _STDERR 392
 _STDIN 392
 _STDOUT 392
 _STZN 392
 _STZO 392
 _SYSERR 392
 _TCFILTER 343, 392
 _TCFOLDER 393
 _TCTAB 393
 _TIME 393
 _TRANSIENT 393
 _TZN 392, 393
 _TZO 392, 393
 _UNICODE 393
 _UTCDATE 393
 _UTCDATETIME 393
 _UTCHOUR 393
 _UTCISODATE 393
 _UTCMINUTE 393
 _UTCSECOND 393
 _UTCTIME 394
 _VIRTUALPC 394
 _VMWARE 394
 _VXPIXELS 394
 _VYPIXELS 394
 _WINDIR 394
 _WINFGWINDOW 394
 _WINNAME 394
 _WINSYSDIR 394
 _WINTICKS 394
 _WINTITLE 394
 _WINUSER 394
 _WINVER 394
 _WOW64 394
 _XPIXELS 394
 _YEAR 394
 _YPIXELS 395

- | -

|| 120

- + -

+ plus sign 126, 381

- = -

= equal sign 126, 381

== equality test operator 109

- 1 -

16-bit Applications 66

- 4 -

4NT 9.0 3

4NT.GPF 475

4NT.INI 24, 388

4START 101

- A -

ABOVENORMAL 331

AC line status 381

ACTIVATE 152

AddFile 33

Administrator 381

Advanced 45, 51

Advanced .INI Directives 41

AFS 103

Alias 33, 154, 222, 355, 408

Aliases 76, 120, 128, 137, 154, 222, 329, 355

AliasExpand 33

Alphabetic 444

Alphanumeric 444

Alt Key 381, 388, 390

Alt-255 33, 104

Alt-Down 104

Alt-End 104

Alt-Home 104

Alt-PgDn 104

Alt-PgUp 104

Alt-Up 104

AND 120

ANSI 101, 510, 516, 518

ANSI X3.64 status 382
App Paths 504
AppendToDir 117
Archive 86, 494
Archive attribute 409
Argument 133, 134
ASCII 409, 444, 510, 511, 522, 530
ASCII Tables 511
ASSOC 162, 241, 409, 506
ATTRIB 86, 163
Attributes 86, 163, 409, 466, 494
Automatic directory change 76

- B -

Background Color 382, 518
Backspace 29, 104
Batch 382
Batch Debugger 139, 166, 174
Batch file name 382
Batch File Parameter 133
Batch File Parameters 327
Batch Files 128, 130, 131, 132, 133, 135, 137, 139, 141, 166, 382
Batch Line Number 382
BATCH.BCP 166
BATCOMP 141, 166
Battery 382
Battery charge 382
BDEBUGGER 139, 166, 174
BEEP 173, 292
BeginLine 29
BELOWNORMAL 331
Bksp 29, 104
BMP 462
Bookmarks 166
Boot drive 383
BOTTOM 152, 362
BREAK 137, 174, 282
BREAKPOINT 166, 174
Build 383

- C -

CALL 175
CANCEL 176, 301
Caps Lock 261, 383

CASE 335
Case Sensitivity 127
CD 177, 178
CDD 178
CDPATH 72, 367
CD-ROM 411
Cell Name 407
Character Device 416
CHCP 180
CHDIR 177
Child Process ID 383
ClearKeyMap 41
Clipboard 32, 67, 412
CLOSE 152
CLS 181
CM 331
CMD.EXE 132, 371, 484
CMD.EXE variables 372
CMDLINE 368
CMSTDIO 331
Code Page 180, 383
COLOR 181
Color Codes 518
Color Dialog 412
Color Names 518
Color settings 181
COLORDIR 368
Colors 382, 387
Columns 384
Command editing 104
Command groups 121
Command History 33, 34, 35, 36, 106, 107, 108, 251, 439
Command Line 50, 103, 104, 126, 383
Command Line Editing Keys 28, 32
Command names 109
Command parsing 124, 134
Command processor 383
Command processor exit codes 24
Command processor options 19
Command processor path 383
Command Processor Version 381
Command separator 381
CommandEscape 33
Commands 135, 144
Commands By Category 147
Commands By Name 144
CommandSep 120, 126, 381

- Comparison
 - case insensitive 109
 - case sensitive 109
 - numeric 109
 - string 109
 - Compatibility 126
 - Compiled batch files 166
 - Compound Character 120
 - Compressed 86, 494
 - Compressed attribute 409
 - Compressed batch file 382
 - Compression 141
 - Computer Name 394
 - COMSPEC 368
 - Conditional commands 120
 - Conditional expression 109
 - CONFIG.NT 131
 - Configuration 24, 284, 323
 - Configuration Dialog 43, 46
 - Console Window 413
 - Contact 477
 - Context Menus 65
 - Control Key 389, 390
 - COPY 182, 274, 371
 - Copy (directive) 29
 - COPYCMD 371
 - Copying Selected Text 67
 - CopyPrompt 182
 - Copyright 537
 - Country Code 384
 - CPU 384
 - Create Directory 271
 - Ctrl key 384
 - Ctrl-A 35, 104, 117
 - Ctrl-Bksp 30, 104
 - Ctrl-Break 104, 137, 174
 - Ctrl-C 137, 174
 - Ctrl-D 33, 40, 106
 - Ctrl-Down 35, 106
 - Ctrl-E 34, 40, 106
 - Ctrl-End 30, 104
 - Ctrl-Enter 35, 40, 106
 - Ctrl-F 33, 104
 - Ctrl-F1 34, 104, 251, 477
 - Ctrl-Home 30, 104
 - Ctrl-K 36, 104, 106
 - Ctrl-L 30, 104
 - CtrlI-C 104
 - Ctrl-Left 32, 104
 - Ctrl-PgUp 34
 - Ctrl-R 30, 104
 - Ctrl-Right 32, 104
 - Ctrl-shift right 104
 - Ctrl-shift-ins 104
 - Ctrl-shift-left 104
 - Ctrl-Shift-Tab 33
 - Ctrl-Tab 35
 - Ctrl-Up 36, 106
 - Ctrl-V 32
 - Ctrl-X 124, 126
 - Ctrl-Y 29, 104
 - Current command line 383
 - Current Working Directory 384, 385, 414
 - Cursor 383
 - Cursor Column 384
 - Cursor Position 310, 384
 - Cursor shape 383
 - CursorIns 383
 - CursorOver 383
- ## - D -
- Date 189, 347, 388, 414
 - Date Formats 127, 189, 406, 414
 - Date ranges 80, 82
 - DATETIME 210
 - Day
 - of month 385
 - of week 386
 - of week (full) 386
 - of week (integer) 386
 - of week (localized) 388
 - of year 386
 - Day of Month 414
 - Day of Week 417, 418, 439
 - Day of Year 419
 - Daylight Savings Time 386
 - Debug 41
 - Debugger 54
 - Debugging 139, 166
 - DEBUGSTRING 190
 - DEFAULT 335
 - Default Variables 222, 319, 357
 - DEFER 190
 - DEFINED 109
 - DEL 190

Del (directive) 30
 DELAY 194
 Delayed Variable Expansion 89
 Delete 104
 DelHistory 33
 DELIMS (FOR command) 234
 DelToBeginning 30
 DelToEnd 30
 DelWordLeft 30
 DelWordRight 30
 DESCRIBE 195, 385
 Description ranges 80
 DescriptionName 195, 385
 Descriptions 86
 Desktop 328
 Desktop Window 462
 DETACH 197, 385
 Detecting 137
 Dialog 68, 69
 DIR 198, 288, 371
 DIRCMD 371
 Directories 492
 Directory 86, 117, 271, 494
 Directory Aliases 76
 Directory attribute 409
 Directory Dialog 437
 Directory History 34, 108, 118, 208
 Directory Navigation 71, 73, 76, 209, 294, 299
 Directory Searches 72, 73, 177, 178, 370
 Directory Stack 71, 209, 294, 299, 416
 DIREXIST 109
 DIRHISTORY 208
 DIRS 209, 294, 299
 DirWinOpen 34
 Disable 323
 Disk serial number 459
 DO 109, 210
 DO (FOR command) 234
 DO UNTIL 109
 DO WHILE 109
 DOS Applications 66
 Down 30, 35, 106
 Drag and drop 68
 DRAWBOX 214
 DRAWHLIN 215, 216
 DRAWVLIN 215, 216
 Drive 491
 Drive Type 419

- E -

ECHO 132, 217, 219, 386
 ECHOERR 218, 220
 Echoing 132
 ECHOS 217, 219
 ECHOSERR 218, 220
 Edit Menu 59
 Editing 104
 Editing keys 104
 EJECTMEDIA 220
 ELSE 253, 254
 ELSEIFF 254
 Enable 323
 Encrypted 86, 494
 Encrypted attribute 409
 Encrypted batch file 382
 End 31, 104
 ENDDO 210
 EndHistory 34
 ENDIFF 254
 EndLine 31
 ENDLOCAL 220, 326
 ENDSWITCH 335
 ENDTEXT 345
 Enter 31, 41, 104
 Environment 222, 319, 357, 365
 Environment Variables 135
 EOL (FOR command) 234
 EQ 109
 EQC 109
 EQL 109
 EQU 109
 ERASE 190
 EraseLine 31
 Error 282, 392
 Error Messages 478
 Error Text 420
 ERRORLEVEL 109, 282, 380, 395
 ERRORMSG 282
 Errors 478, 507
 Esc 31
 Escape 33, 381
 Escape character 124
 EscapeChar 124, 126, 141, 381
 ESET 154, 222, 355, 368, 369
 EVENTLOG 223

EVENTMONITOR 224
EXCEPT 225
Exclusion ranges 80
Exclusive OR 537
ExecLine 31
Executable extensions 91
Executable Files 504
ExecWait 123
EXIST 109
EXIT 227, 386
Exit Code 24, 120, 380
Expand Aliases 33
Extended Directory Searches 177, 178, 299, 370
Extended Parent Directory Names 118
EXTPROC 143

- F -

F1 34, 104, 251, 477
F10 33
F12 36
F3 106
F6 34
F7 35
F8 36
F9 35
FAT 491
FAT32 491
FFIND 227
File completion 35, 36
File Dialog 438
File exclusion ranges 85
File Extension 425
File List 90
File Menu 58
File Names 490, 493
File Prompts 102
File Searches 89, 504
File selection 77, 91
File Streams 496
File Systems 490, 491
File Time Stamps 495
FILECOMPLETION 368
FileCompletion directive 115, 116
FILECOMPLETION variable 115, 116
Filename completion 113, 115, 116
Filename conversion 117
Filenames 117

FILL 214
FIREWIREMONITOR 232
Folder Dialog 438
Folder Locations 461
Folders 63
FOR 234
Foreground Color 387, 518
Foreground Window 394
FOREVER 210
Format Number 436
Format Text 435
FREE 241
FS 331
FTP 93, 255, 387
FTP.CFG 93
FTPS 93, 255, 387
FTYPE 162, 241, 437, 506
FUNCTION 242, 356
Functions 365, 395, 397
Functions Dialog 242

- G -

GE 109
General Input Keys 28, 29
GEQ 109
GLOBAL 108, 245
Glossary 519, 520, 521, 522, 525, 526, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537
GOSUB 247, 307
GOTO 248, 254
GT 109
GTR 109

- H -

Hard Link 272, 273
Hardware Profile 388
HEAD 249, 339, 354
HELP 34, 251, 477
Help Menu 61
HelpWord 34, 251, 477
here-document 98
Hidden 86, 494
Hidden attribute 409
HIDE 152, 362
HIGH 331

Highlighting Text 67
HistLogName 269
HistLogOn 269
History 33, 34, 35, 36, 251, 269, 368, 439
History Log File 387
HISTORYEXCLUDE 368
HistWinOpen 34
Home 29, 104
Host name 387
HTTP 92, 93
<http://jpsoft.com/> 61
HTTPS 92, 93

- I -

IF 109, 253, 439
IFF 109, 253, 254
IFTP 93, 255, 387, 388
IM 260
Include lists 88
Indirect file 90
INIQuery 42
Initialization 24
Initialization files 24, 26
INKEY 257, 259
Inode 442
INPUT 257, 259, 300
Ins 31
Insert 31, 104, 383
Insert cursor 383
Insert cursor shape 383
Instant Message 260
Internal Commands 144, 147
Internal Variables 372, 373, 376
Internet 53, 92, 255
Introduction 16
INV 331
IP 443, 444
IP Address 388
ipworks6.dll 93
ipwssl6.dll 93
ISALIAS 109
ISAPP 109
ISDIR 109
ISFILE 109
ISFUNCTION 109
ISINTERNAL 109
ISLABEL 109

ISO 9601 127
ISO date 385
ISWINDOW 109
ITERATE 210

- J -

JABBER 260
JavaScript 311
JP Software 477
jphelp.chm 477
JPSTREE.IDX 73, 177, 178, 299, 370
Junction (reparse point) 86, 272, 273, 446, 494
Junction (reparse point) attribute 409

- K -

Key Codes 510
Key Mapping Directives 28, 104, 106, 107, 113
Key Names 515
KEYBD 261
Keyboard 261, 388
Keyboard Shortcuts 64
Keypad 104, 510
KEYS 261, 515
KEYSTACK 102, 262

- L -

Label 360, 446, 494
Language 42
LanguageDLL 42
LastHistory 34
LE 109
LEAVE 210
LEAVEFOR (FOR command) 234
Left 31, 104
Length limits 126
LEQ 109
LFN 35, 89, 117, 493
LFNToggle 35, 117
Limits 490
Line Continuation 141
LineToEnd 35
Link 272, 273
LIST 37, 38, 39, 264
LIST Keys 28, 36

List View 63
ListBack 37
ListClipboard 37
ListContinue 37
ListExit 37
ListFind 37
ListFindRegex 38
ListFindRegexReverse 38
ListFindReverse 38
ListHex 38
ListHighBit 38
ListInfo 38
ListNext 38
ListOpen 39
ListPrevious 39
ListPrint 39
ListRefresh 39
ListUnicode 39
ListWrap 39
LOADBTM 269
Local 108
LocalAliases 154
LOG 269
Log File 389
Log Off 303
LogErrors 269
Logical expression 109
Logical operator 109
LogName 269
LogOn 269
Long File Name 89
Long Filename 117
Longest Common Sequence 446
LOW 331
Lower Case 448
LSS 109
LT 109

- M -

MailAddress 316
MailPassword 316
MailPort 316
MailServer 316
MailUser 316
MAX 152, 331, 362
MD 271
MEMORY 272

Menus 58, 59, 60, 61
Message Box 279
Metacharacters 297
MIN 152, 331, 362
Minute 389
MKDIR 271
MKLINK 272
MKLNK 273
Month 388, 389, 450
More? 121
Mount Point 407
MOVE 182, 274
MSAA 43
MSAAMenu 43
MSGBOX 279
Multiple Commands 120
Multiple filenames 87

- N -

Navigation 370
NE 109
NEQ 109
Nesting Level 382
Network Drive 456
NextFile 35
NextHistory 35
Normal 86, 331, 494
Normal attribute 409
NormalEditKey 35
NormalKey 31
NormalListKey 40
NormalPopupKey 40
NOT 109
Not content-indexed 86, 494
Not content-indexed attribute 409
NOTOPMOST 152, 362
NTCMDPROMPT 131
NTFS 491, 496
NTFS Links 448
NTFSDescriptions 195
Num Lock 261
NumLock 389

- O -

Offline 86, 494

Offline attribute 409
ON 137, 282
Online Help 477
OpenAFS 103, 381, 390, 407, 408, 491
OPTION 284, 452
Options 43, 44, 45, 47, 49, 50, 51, 53, 54
Options Menu 61
OR 120
OSD 286
Overstrike 31, 383
Overstrike cursor shape 383
Overview 2

- P -

Page and file prompts 102
Page prompts 102
Parameter 134
Parameter quoting 134
ParameterChar 126, 133
Parameters 109, 133
Parent Directory 118
Parent process 390
Parsing 124, 134
Paste 32, 67
Path 286, 368, 452
PATHEXT 369
PAUSE 287
PDIR 198, 288
Perl 142, 452
PerlScript 311
PGM 331
PgUp 34
Ping 453
Pipes 97
Piping 96, 97, 101
Pixels 394, 395
Platforms 477
PLAYAVI 291
PLAYSOUND 173, 292
PLUGIN 293
POPD 209, 294, 299
PopFile 35
Popup Window Keys 28, 40
Popup Windows 507
PopupWinDel 40
PopupWinEdit 40
PopupWinEditWin 40

PopupWinExec 41
POS 152, 331, 362
POST_EXEC 154
POSTMSG 294
PRE_EXEC 154
PRE_INPUT 154
precision 420
PrevFile 36
PrevHistory 36
Primary 26, 391
PRINT 295
PRIORITY 295
Process ID (PID) 197, 342, 383, 385, 390, 392
PROMPT 297, 369
PUSHD 209, 294, 299

- Q -

QUERYBOX 300
QUIT 176, 301
Quotes 465
Quoting 134

- R -

RAM 272
Random 453
Ranges 80, 82, 84, 85, 86
RD 301
Read-only 86, 494
Read-only attribute 409
REALTIME 331
REBOOT 303
Recall 106
RECYCLE 304
Recycle Bin 190, 301, 304, 369
RECYCLEEXCLUDE 369
Redirection 96, 97, 98, 392
Redirection and Piping 97
Reference 484, 504, 510
Registration 46, 55, 474
Registry 222, 319, 357
 Create 454
 Delete 454
 Exists 454
 Query 455
 Set 455

Registry 222, 319, 357
 Set (broadcast) 455
 Regular Expressions 454, 455, 496
 Relational expression 109
 Relational operator 109
 REM 304
 Remark 304
 Remote Drive 456
 Removable Drive 456
 Removable Media 220
 Remove Directory 301
 REN 274, 305
 RENAME 305
 RepeatFile 36
 Resizing 68
 RESTORE 152, 362
 RETURN 247, 307
 REXEC 308, 309
 REXX 142, 457
 Right 32, 104
 RMDIR 301
 Row 390
 Rows 391
 RSHELL 308, 309
 Ruby 143, 391, 458
 Run Program Dialog 69

- S -

Save Window 462
 SaveHistory 36
 Scan Codes 510
 SCREEN 310, 311
 Screen Reader 43
 Screen resizing 68
 Screen Size 394, 395
 SCRIPT 311
 ScrLk 391
 Scroll Lock 261, 391
 Scrollback Buffer 66
 SCRPUT 310, 311, 360
 Second 391
 Secondary 26, 391
 SELECT 312, 458
 SENDMAIL 316
 SEPARATE 331
 Servers 419
 SERVICEMONITOR 318
 SERVICES 319
 SET 222, 319, 357, 368, 369
 SETDOS 323, 383
 SETLOCAL 220, 326
 Setting colors 181
 Setup 474
 SFN 35, 89, 117, 408, 493
 SHADOW 214
 Shape 383
 SHARED 331
 Sharenames 420
 SHEBANG 143
 SHIFT 327
 Shift Key 389, 391
 Shift right 104
 Shift-left 104
 Shift-Tab 36
 Short file name 460
 Short Filename 117
 SHORTCUT 328
 Shortcuts 64, 328
 SHRALIAS 329, 392
 Shutdown 303
 SIZE 331, 362
 Size ranges 80, 82
 SKIP (FOR command) 234
 SMPP 330
 SMS 330
 SNMP 330
 SNPP 331
 Soft Link 272, 273
 Sparse file 86, 494
 Sparse file attribute 409
 Special Character Compatibility 124, 126
 Standard Error 97, 98, 101, 392
 Standard Input 97, 98, 101, 392
 Standard Output 97, 98, 101, 392, 424
 START 123, 331, 392
 Starting applications 122
 Startup 17, 47
 Startup command 19
 Startup Directory 392
 Startup drive 383
 Startup Options (Take Command) 17
 Startup options (TCC) 19
 Status Bar 64
 Status test 109
 stderr 98, 218, 220

stdin 98
 stdout 98, 217, 219
 Stopwatch 463
 Streams 496
 String Processing 139
 String substitution 372
 Subdirectories 492
 Subroutine 247, 307
 Substrings 372
 Supported Platforms 477
 SwapScrollKeys 106
 SWITCH 335
 Switches 91
 Symbolic Link 407
 symbolic link (reparse point) 463
 symbolic links 272
 SYNC 336
 Syntax Coloring 166
 System 86, 494
 System attribute 409
 System Errors 507
 System Metrics 468
 System Variables 222, 319, 357, 367

- T -

Tab 35
 Tab Configuration 44
 Tab Toolbar Dialog 69
 Tab Window 393
 Tab Windows 64
 Tabs menu 60
 TAIL 249, 339, 354
 Take Command 9.0 3
 Take Command Dialogs 68
 Take Command Interface 56
 Take Command Menus 58
 Take Command Tabs 44
 Take Command Window 56
 Take Command Windows Configuration 43
 TASKDIALOG 341
 TASKEND 342
 TASKLIST 342
 TC Scrollback Buffer Keys 28
 TCEXIT 22
 TCFILTER 343
 TCMD 370
 TCMD.INI 24

TCMD32.GPF 475
 TCMD32.INI 24, 388
 TCMDVER 370
 TCSTART 22, 101
 TCTOOLBAR 343
 Technical Support 475
 TEE 344, 364
 TEMP 370
 Temporary 494
 Temporary attribute 409
 Temporary file 86
 TEXT 345
 TFTP 93
 THEN 254
 Time 189, 347, 393, 463
 Time ranges 80, 84
 Time Stamps 495
 Time Zone 392, 393
 TIMER 347, 463
 TITLE 152, 349, 362, 370, 394
 TITLEPROMPT 370
 TMP 370
 TOKENS (FOR command) 234
 Tool Bar 62, 343
 Toolbar Dialog 69
 TOP 152, 362
 TOPMOST 152, 362
 TOUCH 349
 TRANS 362
 TRANSIENT 352
 Transient Shell 393
 traps 330
 TRAY 362
 TREE 352
 Tree view 63
 TREEEXCLUDE 73, 370
 Troubleshooting 474, 475
 TRUENAME 353, 464
 TYPE 339, 354

- U -

UNALIAS 154, 222, 355
 UNC 464, 491
 UNFUNCTION 242, 356
 Unicode 393, 464
 Unique File Name 465
 UNKNOWN_CMD 154, 355, 504

UNSET 222, 319, 357, 368, 369
UNTIL 210
Up 32, 36, 106
UpdateINI 43
Updates 55
Upper Case 465
URL 92
USBMONITOR 358
User 394
User Variables 222, 319, 357
Utilities Menu 61

- V -

Variable 397
Variable Expansion 36, 89
Variable Functions 395, 397
Variable Functions by Category 401
Variable name completion 119
VARIABLEEXCLUDE 370
VariableExpand 36
Variables 135, 222, 319, 357, 365, 367, 372, 373, 376
VBScript 311
VER 359
VERIFY 360
Version 359, 381, 386, 390, 465, 537
VFAT 491
Virtual Screen 394
VirtualPC 394
VMWare 394
VOL 360
Volatile Variables 222, 319, 357
Volume 360, 491
Volume ID 408
Volume Name 408
VSCRPUT 311, 360

- W -

WAIT 331
Waiting for applications 123
What's New 3
WHICH 361
WHILE 210
Wildcards 77, 466
WIN 331, 362

Window 152, 362
 Class 467
 Position 470
 State 470
Window Title 394
Windows 49
Windows API 467
Windows Directory 394
Windows File Associations 506
Windows Management Instrumentation 472
Windows Memory 468
Windows Parameters 470
Windows System Directory 394
Windows System Errors 507
Windows System Metrics 468
Windows Version 394
WMIQUERY 364
WordLeft 32
WordRight 32
Workgroup 473

- X -

X3.64 101, 510, 516, 537
XML 502
XOR 537

- Y -

Y 344, 364
Year 394, 474

- Z -

ZOOM 214